

Contents

1 DP-203T00: Data Engineering on Azure	7
1.1 How should I use these files relative to the released MOC files?	7
1.2 What about changes to the student handbook?	7
1.3 How do I contribute?	7
1.4 Classroom Materials	7
1.5 Lab overview	7
1.5.1 Lab 1 - Explore compute and storage options for data engineering workloads	7
1.5.2 Lab 2 - Design and implement the serving layer	8
1.5.3 Lab 3 - Data engineering considerations for source files	8
1.5.4 Lab 4 - Run interactive queries using Azure Synapse Analytics serverless SQL pools	8
1.5.5 Lab 5 - Explore, transform, and load data into the Data Warehouse using Apache Spark	8
1.5.6 Lab 6 - Data Exploration and Transformation in Azure Databricks	8
1.5.7 Lab 7 - Ingest and load data into the data warehouse	8
1.5.8 Lab 8 - Transform data with Azure Data Factory or Azure Synapse Pipelines	8
1.5.9 Lab 9 - Integrate data from Notebooks with Azure Data Factory or Azure Synapse Pipelines	8
1.5.10 Lab 10 - Optimize query performance with dedicated SQL pools in Azure Synapse	8
1.5.11 Lab 11 - Analyze and Optimize Data Warehouse Storage	9
1.5.12 Lab 12 - Support Hybrid Transactional Analytical Processing (HTAP) with Azure Synapse Link	9
1.5.13 Lab 13 - End-to-end security with Azure Synapse Analytics	9
1.5.14 Lab 14 - Real-time Stream Processing with Stream Analytics	9
1.5.15 Lab 15 - Create a Stream Processing Solution with Event Hubs and Azure Databricks	9
1.5.16 Lab 16 - Build reports using Power BI integration with Azure Synapse Analytics	9
1.5.17 Lab 17 - Perform Integrated Machine Learning Processes in Azure Synapse Analytics	9
2 Microsoft Data Engineering four-day instructor-led training	9
2.1 Module directory	9
2.2 Modules that can share the same Synapse workspace	10
2.3 Modules that can share the same Azure Databricks workspace	10
3 Module 1 - Explore compute and storage options for data engineering workloads	11
3.1 Lab details	11
3.2 Lab 1 - Delta Lake architecture	11
3.2.1 Before the hands-on lab	11
3.2.1.1 Task 1: Create and configure the Azure Databricks workspace	11
3.2.2 Exercise 1: Complete the lab notebook	11
3.2.2.1 Task 1: Clone the Databricks archive	11
3.2.2.2 Task 2: Complete the following notebook	12
3.3 Lab 2 - Working with Apache Spark in Synapse Analytics	12
3.3.1 Before the hands-on lab	12
3.3.1.1 Task 1: Create and configure the Azure Synapse Analytics workspace	12
3.3.1.2 Task 2: Create and configure additional resources for this lab	12
3.3.2 Exercise 1: Load and data with Spark	13
3.3.2.1 Task 1: Index the Data Lake storage with Hyperspace	13
3.3.2.2 Task 2: Explore the Data Lake storage with the MSSparkUtil library	19
3.3.3 Resources	20
4 Module 2 - Design and Implement the serving layer	20
4.1 Lab details	20
4.1.1 Lab setup and pre-requisites	21
4.2 Exercise 0: Start the dedicated SQL pool	21
4.3 Exercise 1: Implementing a Star Schema	22
4.3.1 Task 1: Create star schema in SQL database	23
4.4 Exercise 2: Implementing a Snowflake Schema	31
4.4.1 Task 1: Create product snowflake schema in SQL database	32
4.4.2 Task 2: Create reseller snowflake schema in SQL database	35
4.5 Exercise 3: Implementing a Time Dimension Table	39
4.5.1 Task 1: Create time dimension table	39
4.5.2 Task 2: Populate the time dimension table	40

4.5.3	Task 3: Load data into other tables	43
4.5.4	Task 4: Query data	45
4.6	Exercise 4: Implementing a Star Schema in Synapse Analytics	47
4.6.1	Task 1: Create star schema in Synapse dedicated SQL	47
4.6.2	Task 2: Load data into Synapse tables	53
4.6.3	Task 3: Query data from Synapse	55
4.7	Exercise 5: Updating slowly changing dimensions with mapping data flows	57
4.7.1	Task 1: Create the Azure SQL Database linked service	58
4.7.2	Task 2: Create a mapping data flow	60
4.7.3	Task 3: Create a pipeline and run the data flow	81
4.7.4	Task 4: View inserted data	86
4.7.5	Task 5: Update a source customer record	88
4.7.6	Task 6: Re-run mapping data flow	89
4.7.7	Task 7: Verify record updated	92
4.8	Exercise 6: Cleanup	94
4.8.1	Task 1: Pause the dedicated SQL pool	94
5	Module 3 - Data engineering considerations for source files	95
5.1	Team whiteboard activity	95
5.2	Whiteboard	96
6	Module 4 - Run interactive queries using serverless SQL pools	96
6.1	Lab details	96
6.2	Lab setup and pre-requisites	97
6.3	Exercise 1: Querying a Data Lake Store using serverless SQL pools in Azure Synapse Analytics	97
6.3.1	Task 1: Query sales Parquet data with serverless SQL pools	98
6.3.2	Task 2: Create an external table for 2019 sales data	100
6.3.3	Task 3: Create an external table for CSV files	105
6.3.4	Task 4: Create a view with a serverless SQL pool	108
6.4	Exercise 2: Securing access to data through using a serverless SQL pool in Azure Synapse Analytics	113
6.4.1	Task 1: Create Azure Active Directory security groups	113
6.4.2	Task 2: Add group members	119
6.4.3	Task 3: Configure data lake security - Role-Based Access Control (RBAC)	122
6.4.4	Task 4: Configure data lake security - Access Control Lists (ACLs)	127
6.4.5	Task 5: Test permissions	130
7	Module 5 - Explore, transform, and load data into the Data Warehouse using Apache Spark	137
7.1	Lab details	138
7.2	Lab setup and pre-requisites	138
7.3	Exercise 0: Start the dedicated SQL pool	138
7.4	Exercise 1: Perform Data Exploration in Synapse Studio	140
7.4.1	Task 1: Exploring data using the data previewer in Azure Synapse Studio	140
7.4.2	Task 2: Using serverless SQL pools to explore files	152
7.4.3	Task 3: Exploring and fixing data with Synapse Spark	156
7.5	Exercise 2: Ingesting data with Spark notebooks in Azure Synapse Analytics	158
7.5.1	Task 1: Ingest and explore Parquet files from a data lake with Apache Spark for Azure Synapse	158
7.6	Exercise 3: Transforming data with DataFrames in Spark pools in Azure Synapse Analytics	164
7.6.1	Task 1: Query and transform JSON data with Apache Spark for Azure Synapse	164
7.7	Exercise 4: Integrating SQL and Spark pools in Azure Synapse Analytics	173
7.7.1	Task 1: Update notebook	173
7.8	Exercise 5: Cleanup	178
7.8.1	Task 1: Pause the dedicated SQL pool	178
8	Module 6 - Data exploration and transformation in Azure Databricks	180
8.1	Lab details	180
8.2	Lab 1 - Working with DataFrames	181
8.2.1	Before the hands-on lab	181
8.2.1.1	Task 1 - Create and configure the Azure Databricks workspace	181
8.2.2	Exercise 1: Complete the lab notebook	181
8.2.2.1	Task 1: Clone the Databricks archive	181

8.2.2.2	Task 2: Complete the Describe a DataFrame notebook	181
8.2.3	Exercise 2: Complete the Working with DataFrames notebook	182
8.2.4	Exercise 3: Complete the Display Function notebook	182
8.2.5	Exercise 4: Complete the Distinct Articles exercise notebook	182
8.3	Lab 2 - Working with DataFrames advanced methods	182
8.3.1	Exercise 2: Complete the lab notebook	182
8.3.1.1	Task 1: Clone the Databricks archive	182
8.3.1.2	Task 2: Complete the Date and Time Manipulation notebook	183
8.3.2	Exercise 3: Complete the Use Aggregate Functions notebook	183
8.3.3	Exercise 4: Complete the De-Duping Data exercise notebook	183
9	Module 7 - Ingest and load data into the Data Warehouse	183
9.1	Lab details	184
9.2	Lab setup and pre-requisites	184
9.3	Exercise 0: Start the dedicated SQL pool	184
9.4	Exercise 1: Import data with PolyBase and COPY using T-SQL	186
9.4.1	Task 1: Create staging tables	186
9.4.2	Task 2: Configure and run PolyBase load operation	188
9.4.3	Task 3: Configure and run the COPY statement	190
9.4.4	Task 4: Load data into the clustered columnstore table	190
9.4.5	Task 5: Use COPY to load text file with non-standard row delimiters	191
9.4.6	Task 6: Use PolyBase to load text file with non-standard row delimiters	192
9.5	Exercise 2: Petabyte-scale ingestion with Azure Synapse Pipelines	193
9.5.1	Task 1: Configure workload management classification	193
9.5.2	Task 2: Create pipeline with copy activity	198
9.6	Exercise 3: Cleanup	209
9.6.1	Task 1: Pause the dedicated SQL pool	209
10	Module 8 - Transform data with Azure Data Factory or Azure Synapse Pipelines	210
10.1	Lab details	210
10.2	Lab setup and pre-requisites	211
10.3	Exercise 0: Start the dedicated SQL pool	211
10.4	Lab 1: Code-free transformation at scale with Azure Synapse Pipelines	213
10.4.1	Exercise 1: Create artifacts	213
10.4.1.1	Task 1: Create SQL table	213
10.4.1.2	Task 2: Create linked service	215
10.4.1.3	Task 3: Create data sets	217
10.4.1.4	Task 4: Create campaign analytics dataset	228
10.4.2	Exercise 2: Create data pipeline to import poorly formatted CSV	233
10.4.2.1	Task 1: Create campaign analytics data flow	233
10.4.2.2	Task 2: Create campaign analytics data pipeline	244
10.4.2.3	Task 3: Run the campaign analytics data pipeline	246
10.4.2.4	Task 4: View campaign analytics table contents	248
10.4.3	Exercise 3: Create Mapping Data Flow for top product purchases	251
10.4.3.1	Task 1: Create Mapping Data Flow	251
10.5	Lab 2: Orchestrate data movement and transformation in Azure Synapse Pipelines	274
10.5.1	Exercise 1: Create, trigger, and monitor pipeline	274
10.5.1.1	Task 1: Create pipeline	274
10.5.1.2	Task 2: Trigger, monitor, and analyze the user profile data pipeline	278
10.5.2	Exercise 2: Cleanup	282
10.5.2.1	Task 1: Pause the dedicated SQL pool	283
11	Module 9 - Integrate data from notebooks with Azure Data Factory or Azure Synapse Pipelines	284
11.1	Lab details	284
11.2	Lab setup and pre-requisites	284
11.3	Exercise 1: Linked service and datasets	285
11.3.1	Task 1: Create linked service	285
11.3.2	Task 2: Create datasets	287
11.4	Exercise 2: Create mapping data flow and pipeline	293
11.4.1	Task 1: Retrieve the ADLS Gen2 linked service name	293

11.4.2	Task 2: Create mapping data flow	295
11.4.3	Task 3: Create pipeline	298
11.4.4	Task 4: Trigger the pipeline	301
11.5	Exercise 3: Create Synapse Spark notebook to find top products	303
11.5.1	Task 1: Create notebook	303
11.5.2	Task 2: Add the Notebook to the pipeline	311
12	Module 10 - Optimize query performance with dedicated SQL pools in Azure Synapse	322
12.1	Lab details	322
12.2	Lab setup and pre-requisites	323
12.3	Exercise 0: Start the dedicated SQL pool	323
12.4	Exercise 1: Understanding developer features of Azure Synapse Analytics	325
12.4.1	Task 1: Create tables and load data	325
12.4.2	Task 2: Using window functions	328
12.4.2.1	Task 2.1: OVER clause	328
12.4.2.2	Task 2.2: Aggregate functions	330
12.4.2.3	Task 2.3: Analytic functions	331
12.4.2.4	Task 2.4: ROWS clause	333
12.4.3	Task 3: Approximate execution using HyperLogLog functions	335
12.5	Exercise 2: Using data loading best practices in Azure Synapse Analytics	335
12.5.1	Task 1: Implement workload management	335
12.5.2	Task 2: Create a workload classifier to add importance to certain queries	336
12.5.3	Task 3: Reserve resources for specific workloads through workload isolation	341
12.6	Exercise 3: Optimizing data warehouse query performance in Azure Synapse Analytics	346
12.6.1	Task 1: Identify performance issues related to tables	346
12.6.2	Task 2: Improve table structure with hash distribution and columnstore index	355
12.6.3	Task 4: Improve further the table structure with partitioning	359
12.6.3.1	Task 4.1: Table distributions	361
12.6.3.2	Task 4.2: Indexes	361
12.6.3.3	Task 4.3: Partitioning	362
12.7	Exercise 4: Improve query performance	362
12.7.1	Task 1: Use materialized views	362
12.7.2	Task 2: Use result set caching	366
12.7.3	Task 3: Create and update statistics	372
12.7.4	Task 4: Create and update indexes	373
12.7.5	Task 5: Ordered Clustered Columnstore Indexes	374
12.8	Exercise 5: Cleanup	377
12.8.1	Task 1: Pause the dedicated SQL pool	377
13	Module 11 - Analyze and optimize Data Warehouse storage	379
13.1	Lab details	379
13.2	Lab setup and pre-requisites	380
13.3	Exercise 0: Start the dedicated SQL pool	380
13.4	Exercise 1 - Check for skewed data and space usage	381
13.4.1	Task 1 - Analyze the space used by tables	381
13.4.2	Task 2 - Use a more advanced approach to understand table space usage	386
13.5	Exercise 2 - Understand column store storage details	389
13.5.1	Task 1 - Create view for column store row group stats	389
13.5.2	Task 2 - Explore column store storage details	390
13.6	Exercise 3 - Study the impact of wrong choices for column data types	391
13.6.1	Task 1 - Create and populate tables with optimal column data types	391
13.6.2	Task 2 - Create and populate tables with sub-optimal column data types	392
13.6.3	Task 3 - Compare storage requirements	392
13.7	Exercise 4 - Study the impact of materialized views	394
13.7.1	Task 1 - Analyze the execution plan of a query	394
13.7.2	Task 2 - Improve the execution plan of the query with a materialized view	395
13.8	Exercise 5 - Avoid extensive logging	399
13.8.1	Task 1 - Explore rules for minimally logged operations	399
13.8.2	Task 2 - Optimizing a delete operation	400
13.9	Exercise 6: Cleanup	401
13.9.1	Task 1: Pause the dedicated SQL pool	401

14 Module 12 - Support Hybrid Transactional Analytical Processing (HTAP) with Azure Synapse Link	402
14.1 Lab details	402
14.2 Lab setup and pre-requisites	402
14.3 Exercise 1: Lab setup	403
14.3.1 Task 1: Create linked service	403
14.3.2 Task 2: Create dataset	405
14.4 Exercise 2: Configuring Azure Synapse Link with Azure Cosmos DB	409
14.4.1 Task 1: Enable Azure Synapse Link	410
14.4.2 Task 2: Create a new Azure Cosmos DB container	411
14.4.3 Task 3: Create and run a copy pipeline	414
14.5 Exercise 3: Querying Azure Cosmos DB with Apache Spark for Synapse Analytics	420
14.5.1 Task 1: Create a notebook	421
14.6 Exercise 4: Querying Azure Cosmos DB with serverless SQL pool for Azure Synapse Analytics	428
14.6.1 Task 1: Create a new SQL script	428
15 Module 13 - End-to-end security with Azure Synapse Analytics	434
15.1 Lab details	434
15.2 Resource naming throughout this lab	434
15.3 Lab setup and pre-requisites	435
15.4 Exercise 0: Start the dedicated SQL pool	435
15.5 Exercise 1 - Securing Azure Synapse Analytics supporting infrastructure	437
15.5.1 Task 1 - Observing the SQL Active Directory admin	437
15.5.2 Task 2 - Manage IP firewall rules	437
15.6 Exercise 2 - Securing the Azure Synapse Analytics workspace and managed services	439
15.6.1 Task 1 - Managing secrets with Azure Key Vault	439
15.6.2 Task 2 - Use Azure Key Vault for secrets when creating Linked Services	440
15.6.3 Task 3 - Secure workspace pipeline runs	442
15.6.4 Task 4 - Secure Azure Synapse Analytics dedicated SQL pools	449
15.7 Exercise 3 - Securing Azure Synapse Analytics workspace data	450
15.7.1 Task 1 - Column Level Security	450
15.7.2 Task 2 - Row level security	451
15.7.3 Task 3 - Dynamic data masking	453
15.8 Exercise 4: Cleanup	454
15.8.1 Task 1: Pause the dedicated SQL pool	454
15.9 Reference	456
15.10 Other Resources	456
16 Module 14 - Real-time stream processing with Stream Analytics	456
16.1 Lab details	456
16.2 Technology overview	457
16.2.1 Azure Stream Analytics	457
16.2.2 Azure Event Hubs	457
16.2.3 Power BI	457
16.3 Scenario overview	457
16.4 Lab setup and pre-requisites	458
16.5 Exercise 0: Start the dedicated SQL pool	458
16.6 Exercise 1: Configure services	459
16.6.1 Task 1: Configure Event Hubs	459
16.6.2 Task 2: Configure Synapse Analytics	463
16.6.3 Task 3: Configure Stream Analytics	465
16.7 Exercise 2: Generate and visualize data	479
16.7.1 Task 1: Run data generator	479
16.7.2 Task 2: Create Power BI dashboard	482
16.7.3 Task 3: View aggregate data in Synapse Analytics	499
16.8 Exercise 3: Cleanup	502
16.8.1 Task 1: Stop the data generator	502
16.8.2 Task 2: Stop the Stream Analytics job	503
16.8.3 Task 3: Pause the dedicated SQL pool	503
17 Module 15 - Create a stream processing solution with Event Hubs and Azure Databricks	504

17.1	Lab details	505
17.2	Concepts	505
17.3	Event Hubs and Spark Structured Streaming	505
17.4	Streaming concepts	505
17.5	Lab	505
17.5.1	Before the hands-on lab	505
17.5.1.1	Task 1: Create and configure the Azure Databricks workspace	506
17.5.2	Exercise 1: Complete the Structured Streaming Concepts notebook	506
17.5.2.1	Task 1: Clone the Databricks archive	506
17.5.2.2	Task 2: Complete the notebook	506
17.5.3	Exercise 2: Complete the Working with Time Windows notebook	506
17.5.4	Exercise 3: Complete the Structured Streaming with Azure EventHubs notebook	507
18	Module 16 - Build reports using Power BI integration with Azure Synapse Analytics	507
18.1	Lab details	507
18.2	Resource naming throughout this lab	507
18.3	Lab setup and pre-requisites	508
18.4	Exercise 0: Start the dedicated SQL pool	508
18.5	Exercise 1: Power BI and Synapse workspace integration	510
18.5.1	Task 1: Login to Power BI	510
18.5.2	Task 2: Create a Power BI workspace	510
18.5.3	Task 3: Connect to Power BI from Synapse	512
18.5.4	Task 4: Explore the Power BI linked service in Synapse Studio	515
18.5.5	Task 5: Create a new datasource to use in Power BI Desktop	517
18.5.6	Task 6: Create a new Power BI report in Synapse Studio	520
18.6	Exercise 2: Optimizing integration with Power BI	534
18.6.1	Task 1: Explore Power BI optimization options	534
18.6.2	Task 2: Improve performance with materialized views	534
18.6.3	Task 3: Improve performance with result-set caching	538
18.7	Exercise 3: Visualize data with SQL Serverless	540
18.7.1	Task 1: Explore the data lake with SQL Serverless	540
18.7.2	Task 2: Visualize data with SQL serverless and create a Power BI report	545
18.8	Exercise 4: Cleanup	558
18.8.1	Task 1: Pause the dedicated SQL pool	558
19	Module 17 - Perform integrated Machine Learning processes in Azure Synapse Analytics	560
19.1	Lab details	560
19.2	Pre-requisites	560
19.3	Before the hands-on lab	561
19.3.1	Task 1: Create and configure the Azure Synapse Analytics workspace	561
19.3.2	Task 2: Create and configure additional resources for this lab	561
19.4	Exercise 0: Start the dedicated SQL pool	561
19.5	Exercise 1: Create an Azure Machine Learning linked service	563
19.5.1	Task 1: Create and configure an Azure Machine Learning linked service in Synapse Studio	563
19.5.2	Task 2: Explore Azure Machine Learning integration features in Synapse Studio	569
19.6	Exercise 2: Trigger an Auto ML experiment using data from a Spark table	571
19.6.1	Task 1: Trigger a regression Auto ML experiment on a Spark table	571
19.6.2	Task 2: View experiment details in Azure Machine Learning workspace	575
19.7	Exercise 3: Enrich data using trained models	577
19.7.1	Task 1: Enrich data in a SQL pool table using a trained model from Azure Machine Learning	578
19.7.2	Task 2: Enrich data in a Spark table using a trained model from Azure Cognitive Services	584
19.7.3	Task 3: Integrate a Machine Learning-based enrichment procedure in a Synapse pipeline .	587
19.8	Exercise 4: Serve prediction results using Power BI	594
19.8.1	Task 1: Display prediction results in a Power BI report	594
19.9	Exercise 5: Cleanup	603
19.9.1	Task 1: Pause the dedicated SQL pool	603
19.10	Resources	605

1 DP-203T00: Data Engineering on Azure

Welcome to the course DP-203: Data Engineering on Azure. To support this course, we will need to make updates to the course content to keep it current with the Azure services used in the course. We are publishing the lab instructions and lab files on GitHub to allow for open contributions between the course authors and MCTs to keep the content current with changes in the Azure platform.

- **Are you a MCT?** - Have a look at our [GitHub User Guide for MCTs](#).

1.1 How should I use these files relative to the released MOC files?

- The instructor handbook and PowerPoints are still going to be your primary source for teaching the course content.
- These files on GitHub are designed to be used in conjunction with the student handbook, but are in GitHub as a central repository so MCTs and course authors can have a shared source for the latest lab files.
- the lab instructions for each module are found in the /Instructions/Labs folder. Each subfolder within this location refers to each module. For example, Lab01 relates to module01 etc. A README.md file exists in each folder with the lab instructions that the students will then follow.
- It will be recommended that for every delivery, trainers check GitHub for any changes that may have been made to support the latest Azure services, and get the latest files for their delivery.
- Please note that some of the images that you see in these lab instructions will not necessarily reflect the state of the lab environment that you will be using in this course. For example, while browsing for files in a data lake, you may see additional folders in the images that may not exist in your environment. This is by design, and your lab instructions will still work.

1.2 What about changes to the student handbook?

- We will review the student handbook on a quarterly basis and update through the normal MOC release channels as needed.

1.3 How do I contribute?

- Any MCT can submit issues to the code or content in the GitHub repro, Microsoft and the course author will triage and include content and lab code changes as needed.

1.4 Classroom Materials

It is strongly recommended that MCTs and Partners access these materials and in turn, provide them separately to students. Pointing students directly to GitHub to access Lab steps as part of an ongoing class will require them to access yet another UI as part of the course, contributing to a confusing experience for the student. An explanation to the student regarding why they are receiving separate Lab instructions can highlight the nature of an always-changing cloud-based interface and platform. Microsoft Learning support for accessing files on GitHub and support for navigation of the GitHub site is limited to MCTs teaching this course only.

1.5 Lab overview

The following is a summary of the lab objectives for each module:

1.5.1 Lab 1 - Explore compute and storage options for data engineering workloads

This lab teaches ways to structure the data lake, and to optimize the files for exploration, streaming, and batch workloads. The student will learn how to organize the data lake into levels of data refinement as they transform files through batch and stream processing. The students will also experience working with Apache Spark in Azure Synapse Analytics. They will learn how to create indexes on their datasets, such as CSV, JSON, and Parquet files, and use them for potential query and workload acceleration using Spark libraries including Hyperspace and MSSparkUtils.

1.5.2 Lab 2 - Design and implement the serving layer

This lab teaches how to design and implement data stores in a modern data warehouse to optimize analytical workloads. The student will learn how to design a multidimensional schema to store fact and dimension data. Then the student will learn how to populate slowly changing dimensions through incremental data loading from Azure Data Factory.

1.5.3 Lab 3 - Data engineering considerations for source files

In this lab, you will be directed by your instructor to work alone, or in groups for 20 minutes to read through the following information presented below. You will then answer the questions and present back to the classroom your findings based on the requirements.

1.5.4 Lab 4 - Run interactive queries using Azure Synapse Analytics serverless SQL pools

In this lab, students will learn how to work with files stored in the data lake and external file sources, through T-SQL statements executed by a serverless SQL pool in Azure Synapse Analytics. Students will query Parquet files stored in a data lake, as well as CSV files stored in an external data store. Next, they will create Azure Active Directory security groups and enforce access to files in the data lake through Role-Based Access Control (RBAC) and Access Control Lists (ACLs).

1.5.5 Lab 5 - Explore, transform, and load data into the Data Warehouse using Apache Spark

This lab teaches you how to explore data stored in a data lake, transform the data, and load data into a relational data store. You will explore Parquet and JSON files and use techniques to query and transform JSON files with hierarchical structures. Then you will use Apache Spark to load data into the data warehouse and join Parquet data in the data lake with data in the dedicated SQL pool.

1.5.6 Lab 6 - Data Exploration and Transformation in Azure Databricks

This lab teaches you how to use various Apache Spark DataFrame methods to explore and transform data in Azure Databricks. You will learn how to perform standard DataFrame methods to explore and transform data. You will also learn how to perform more advanced tasks, such as removing duplicate data, manipulate date/time values, rename columns, and aggregate data. They will provision the chosen ingestion technology and integrate this with Stream Analytics to create a solution that works with streaming data.

1.5.7 Lab 7 - Ingest and load data into the data warehouse

This lab teaches students how to ingest data into the data warehouse through T-SQL scripts and Synapse Analytics integration pipelines. The student will learn how to load data into Synapse dedicated SQL pools with PolyBase and COPY using T-SQL. The student will also learn how to use workload management along with a Copy activity in a Azure Synapse pipeline for petabyte-scale data ingestion.

1.5.8 Lab 8 - Transform data with Azure Data Factory or Azure Synapse Pipelines

This lab teaches students how to build data integration pipelines to ingest from multiple data sources, transform data using mapping data flows and notebooks, and perform data movement into one or more data sinks.

1.5.9 Lab 9 - Integrate data from Notebooks with Azure Data Factory or Azure Synapse Pipelines

In the lab, the students will create a notebook to query user activity and purchases that they have made in the past 12 months. They will then add the notebook to a pipeline using the new Notebook activity and execute this notebook after the Mapping Data Flow as part of their orchestration process. While configuring this the students will implement parameters to add dynamic content in the control flow and validate how the parameters can be used.

1.5.10 Lab 10 - Optimize query performance with dedicated SQL pools in Azure Synapse

In this lab, students will learn strategies to optimize data storage and processing when using dedicated SQL pools in Azure Synapse Analytics. The student will know how to use developer features, such as windowing and HyperLogLog functions, use data loading best practices, and optimize and improve query performance.

1.5.11 Lab 11 - Analyze and Optimize Data Warehouse Storage

This lab teaches you how to analyze and optimize the data storage of the Azure Synapse dedicated SQL pools. You will know techniques to understand table space usage and column store storage details. Next, you will know how to compare storage requirements between identical tables that use different data types. Finally, you will observe the impact materialized views have when executed in place of complex queries and learn how to avoid extensive logging by optimizing delete operations.

1.5.12 Lab 12 - Support Hybrid Transactional Analytical Processing (HTAP) with Azure Synapse Link

This lab teaches you how Azure Synapse Link enables seamless connectivity of an Azure Cosmos DB account to a Synapse workspace. You will understand how to enable and configure Synapse link, then how to query the Azure Cosmos DB analytical store using Apache Spark and SQL Serverless.

1.5.13 Lab 13 - End-to-end security with Azure Synapse Analytics

In this lab, students will learn how to secure a Synapse Analytics workspace and its supporting infrastructure. The student will observe the SQL Active Directory Admin, manage IP firewall rules, manage secrets with Azure Key Vault and access those secrets through a Key Vault linked service and pipeline activities. The student will understand how to implement column-level security, row-level security, and dynamic data masking when using dedicated SQL pools.

1.5.14 Lab 14 - Real-time Stream Processing with Stream Analytics

This lab teaches you how to process streaming data with Azure Stream Analytics. You will ingest vehicle telemetry data into Event Hubs, then process that data in real time, using various windowing functions in Azure Stream Analytics. You will output the data to Azure Synapse Analytics. Finally, you will learn how to scale the Stream Analytics job to increase throughput.

1.5.15 Lab 15 - Create a Stream Processing Solution with Event Hubs and Azure Databricks

This lab teaches you how to ingest and process streaming data at scale with Event Hubs and Spark Structured Streaming in Azure Databricks. You will learn the key features and uses of Structured Streaming. You will implement sliding windows to aggregate over chunks of data and apply watermarking to remove stale data. Finally, you will connect to Event Hubs to read and write streams.

1.5.16 Lab 16 - Build reports using Power BI integration with Azure Synapse Analytics

In this lab, the student will learn how to integrate Power BI with their Azure Synapse workspace to build reports in Power BI. The student will create a new data source and Power BI report in Azure Synapse Studio. Then the student will learn how to improve query performance with materialized views and result-set caching. Finally, the student will explore the data lake with serverless SQL pools and create visualizations against that data in Power BI.

1.5.17 Lab 17 - Perform Integrated Machine Learning Processes in Azure Synapse Analytics

In the lab, the students will explore the integrated, end-to-end Azure Machine Learning and Azure Cognitive Services experience in Azure Synapse Analytics. You will learn how to connect an Azure Synapse Analytics workspace to an Azure Machine Learning workspace using a Linked Service and then trigger an Automated ML experiment that uses data from a Spark table. You will also learn how to use trained models from Azure Machine Learning or Azure Cognitive Services to enrich data in a SQL pool table and then serve prediction results using Power BI.

2 Microsoft Data Engineering four-day instructor-led training

2.1 Module directory

- [Module 1 - Explore compute and storage options for data engineering workloads](#)
- [Module 2 - Design and Implement the serving layer](#)
- [Module 3 - Data engineering considerations for source files

- Module 4 - Run interactive queries using serverless SQL pools
- Module 5 - Explore, transform, and load data into the Data Warehouse using Apache Spark
- Module 6 - Data exploration and transformation in Azure Databricks
- Module 7 - Ingest and load data into the Data Warehouse
- Module 8 - Transform data with Azure Data Factory or Azure Synapse Pipelines
- Module 9 - Integrate data from notebooks with Azure Data Factory or Azure Synapse Pipelines
- Module 10 - Optimize query performance with dedicated SQL pools in Azure Synapse
- Module 11 - Analyze and optimize Data Warehouse storage
- Module 12 - Support Hybrid Transactional Analytical Processing (HTAP) with Azure Synapse Link
- Module 13 - End-to-end security with Azure Synapse Analytics
- Module 14 - Real-time stream processing with Stream Analytics
- Module 15 - Create a stream processing solution with Event Hubs and Azure Databricks
- Module 16 - Build reports using Power BI integration with Azure Synapse Analytics
- Module 17 - Perform integrated Machine Learning processes in Azure Synapse Analytics

2.2 Modules that can share the same Synapse workspace

- Synapse workspace named **asagworkspaceSUFFIX** (where SUFFIX is a unique id provided during setup):
 - Module 1
 - Module 17
- Synapse workspace named **asaworkspaceSUFFIX**:
 - Module 4
 - Module 5
 - Module 7
 - Module 8
 - Module 9
 - Module 10
 - Module 11
 - Module 12
 - Module 13
 - Module 16
- Requires own Synapse workspace (not shared):
 - Module 2
 - Module 14

2.3 Modules that can share the same Azure Databricks workspace

- Module 1
- Module 6
- Module 15

3 Module 1 - Explore compute and storage options for data engineering workloads

This module teaches ways to structure the data lake, and to optimize the files for exploration, streaming, and batch workloads. The student will learn how to organize the data lake into levels of data refinement as they transform files through batch and stream processing. Then they will learn how to create indexes on their datasets, such as CSV, JSON, and Parquet files, and use them for potential query and workload acceleration.

In this module, the student will be able to:

- Combine streaming and batch processing with a single pipeline
- Organize the data lake into levels of file transformation
- Index data lake storage for query and workload acceleration

3.1 Lab details

- [Module 1 - Explore compute and storage options for data engineering workloads](#)
 - [Lab details](#)
 - [Lab 1 - Delta Lake architecture](#)
 - * [Before the hands-on lab](#)
 - Task 1: Create and configure the Azure Databricks workspace
 - * [Exercise 1: Complete the lab notebook](#)
 - Task 1: Clone the Databricks archive
 - Task 2: Complete the following notebook
 - [Lab 2 - Working with Apache Spark in Synapse Analytics](#)
 - * [Before the hands-on lab](#)
 - Task 1: Create and configure the Azure Synapse Analytics workspace
 - Task 2: Create and configure additional resources for this lab
 - * [Exercise 1: Load and data with Spark](#)
 - Task 1: Index the Data Lake storage with Hyperspace
 - Task 2: Explore the Data Lake storage with the MSSparkUtil library
 - * [Resources](#)

3.2 Lab 1 - Delta Lake architecture

In this lab, you will use an Azure Databricks workspace and perform Structured Streaming with batch jobs by using Delta Lake. You need to complete the exercises within a Databricks Notebook. To begin, you need to have access to an Azure Databricks workspace. If you do not have a workspace available, follow the instructions below. Otherwise, you can skip to the bottom of the page to [Clone the Databricks archive](#).

3.2.1 Before the hands-on lab

Note: Only complete the [Before the hands-on lab](#) steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to [Exercise 1](#).

Before stepping through the exercises in this lab, make sure you have access to an Azure Databricks workspace with an available cluster. Perform the tasks below to configure the workspace.

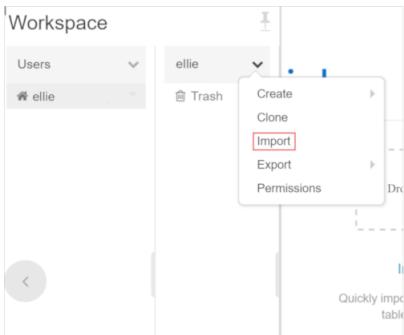
3.2.1.1 Task 1: Create and configure the Azure Databricks workspace

If you are not using a hosted lab environment, follow the [lab 01](#) setup instructions to manually create and configure the workspace.

3.2.2 Exercise 1: Complete the lab notebook

3.2.2.1 Task 1: Clone the Databricks archive

1. If you do not currently have your Azure Databricks workspace open: in the Azure portal, navigate to your deployed Azure Databricks workspace and select **Launch Workspace**.
2. In the left pane, select **Workspace > Users**, and select your username (the entry with the house icon).
3. In the pane that appears, select the arrow next to your name, and select **Import**.



1. In the **Import Notebooks** dialog box, select the URL and paste in the following URL:

<https://github.com/solliancenet/microsoft-learning-paths-databricks-notebooks/blob/master/data-engineer>

1. Select **Import**.
2. Select the **11-Delta-Lake-Architecture** folder that appears.

3.2.2.2 Task 2: Complete the following notebook

Open the **1-Delta-Architecture** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will explore combining streaming and batch processing with a single pipeline.

After you've completed the notebook, return to this screen, and continue to the next lab.

3.3 Lab 2 - Working with Apache Spark in Synapse Analytics

This lab demonstrates the experience of working with Apache Spark in Azure Synapse Analytics. You will learn how to connect an Azure Synapse Analytics workspace to an Azure Data Explorer workspace using a Linked Service and then load data from one of its databases using a Spark notebook. You will also learn how to use libraries like Hyperspace and MSSparkUtil to optimize the experience of working with Data Lake storage accounts from Spark notebooks. In addition to Data Explorer and Data Lake storage, the data enrichment process will also use historical data from a SQL Pool. In the end, you will learn how to publish the enriched data back into the Data Lake and consume it with the SQL Built-in Pool and Power BI.

After completing the lab, you will understand the main steps of an end-to-end data enrichment process that uses Spark in an Azure Synapse Analytics workspace.

3.3.1 Before the hands-on lab

Note: Only complete the **Before the hands-on lab** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 1.

Before stepping through the exercises in this lab, make sure you have properly configured your Azure Synapse Analytics workspace. Perform the tasks below to configure the workspace.

3.3.1.1 Task 1: Create and configure the Azure Synapse Analytics workspace

NOTE

If you have already created and configured the Synapse Analytics workspace while running one of the other labs available in this repo, you must not perform this task again and you can move on to the next task. The labs are designed to share the Synapse Analytics workspace, so you only need to create it once.

If you are not using a hosted lab environment, follow the instructions in [Deploy your Azure Synapse Analytics workspace](#) to create and configure the workspace.

3.3.1.2 Task 2: Create and configure additional resources for this lab

If you are not using a hosted lab environment, follow the instructions in [Deploy resources for Lab 02](#) to deploy additional resources for this lab. Once deployment is complete, you are ready to proceed with the exercises in this lab.

3.3.2 Exercise 1: Load and data with Spark

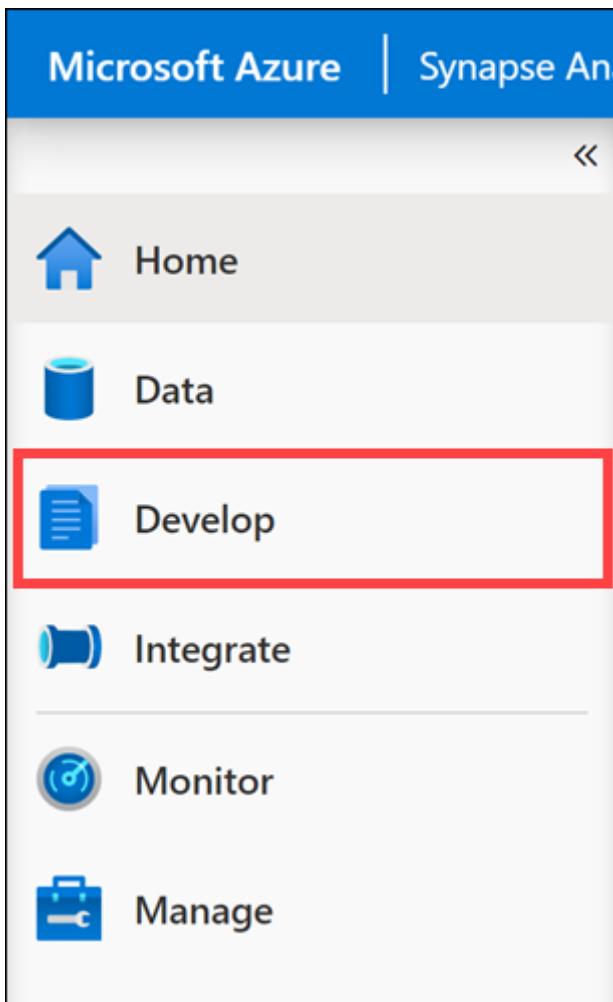
3.3.2.1 Task 1: Index the Data Lake storage with Hyperspace

When loading data from Azure Data Lake Gen 2, searching in the data in one of the most resource consuming operations. Hyperspace introduces the ability for Apache Spark users to create indexes on their datasets, such as CSV, JSON, and Parquet, and use them for potential query and workload acceleration.

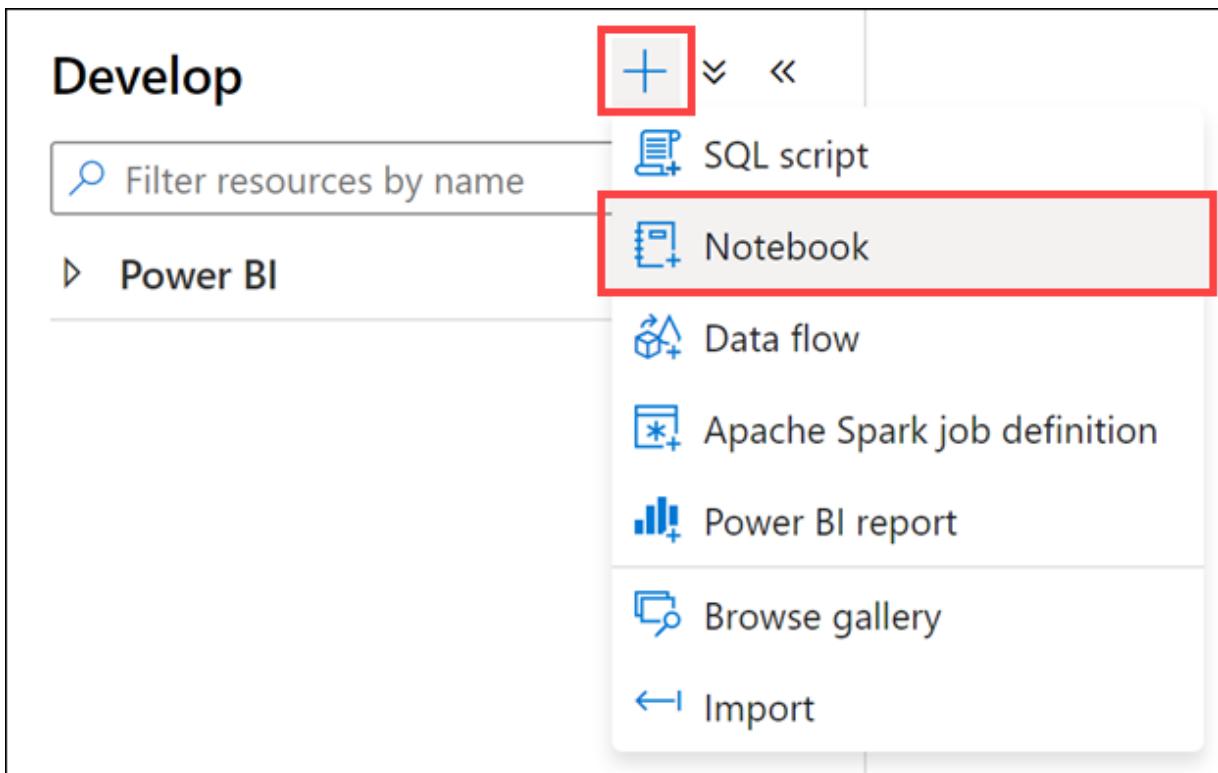
Hyperspace lets you create indexes on records scanned from persisted data files. After they're successfully created, an entry that corresponds to the index is added to the Hyperspace's metadata. This metadata is later used by Apache Spark's optimizer during query processing to find and use proper indexes. If the underlying data changes, you can refresh an existing index to capture that.

Also, Hyperspace allows users to compare their original plan versus the updated index-dependent plan before running their query.

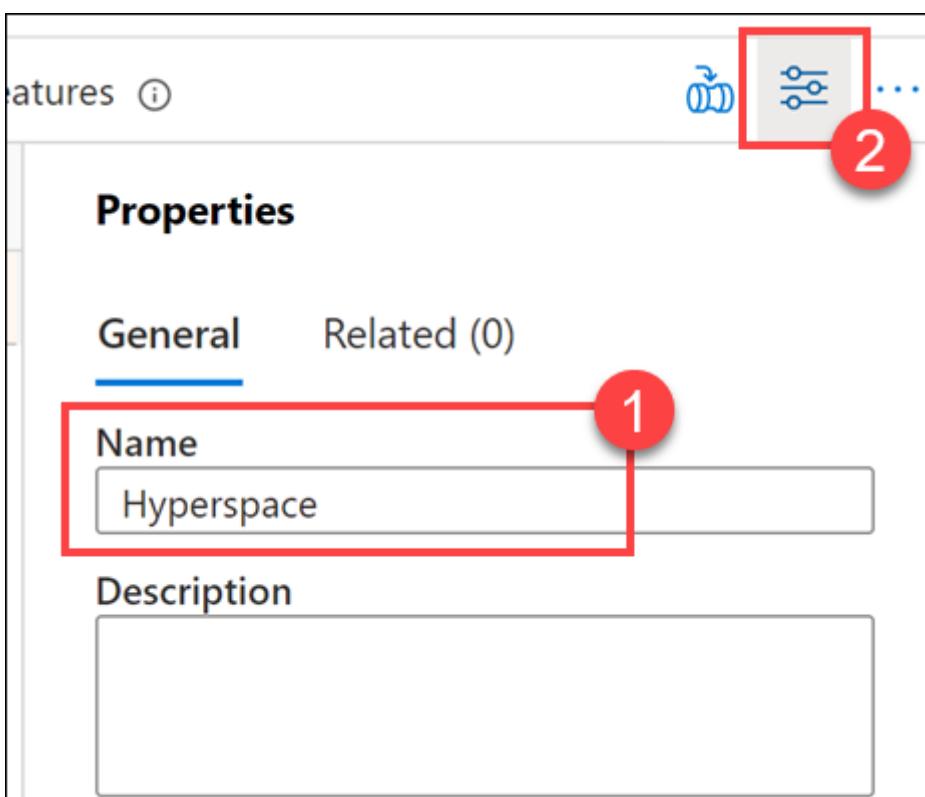
1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Develop** hub.



3. Select +, then **Notebook** to create a new Synapse notebook.



4. Enter **Hyperspace** for the notebook name (1), then select the **Properties** button above (2) to hide the properties pane.



5. Attach the notebook to the Spark cluster and make sure that the language is set to **PySpark (Python)**.



6. Add the following code to a new cell in your notebook:

```

from hyperspace import *
from com.microsoft.hyperspace import *
from com.microsoft.hyperspace.index import *

# Disable BroadcastHashJoin, so Spark will use standard SortMergeJoin. Currently, Hyperspace index
spark.conf.set("spark.sql.autoBroadcastJoinThreshold", -1)

# Replace the value below with the name of your primary ADLS Gen2 account for your Synapse workspace
datalake = 'REPLACE_WITH_YOUR_DATALAKE_NAME'

dfSales = spark.read.parquet("abfss://wwi-02@" + datalake + ".dfs.core.windows.net/sale-small/Year"
dfSales.show(10)

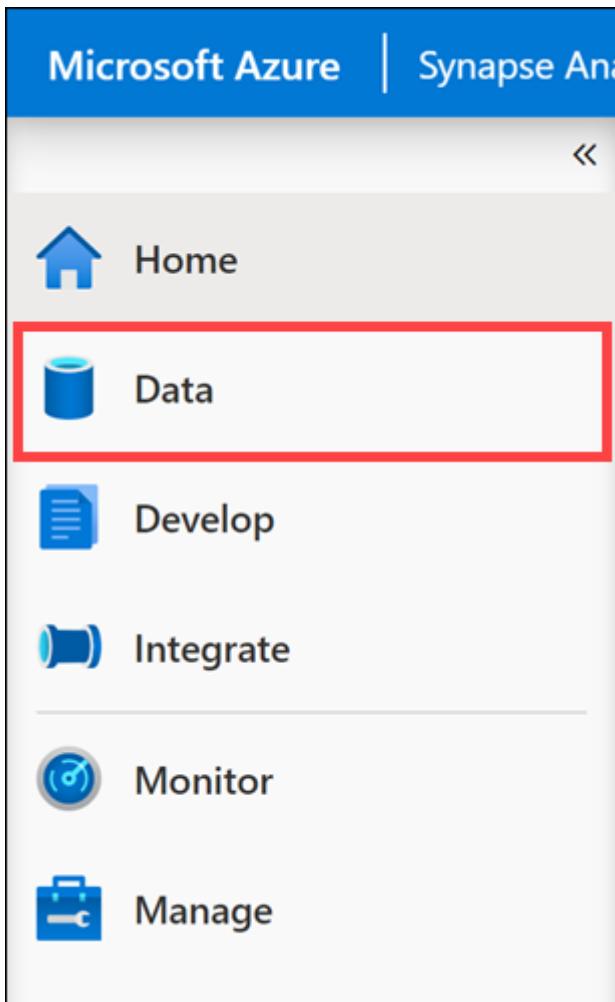
dfCustomers = spark.read.load("abfss://wwi-02@" + datalake + ".dfs.core.windows.net/data-generator"
dfCustomers.show(10)

# Create an instance of Hyperspace
hyperspace = Hyperspace(spark)

```

Replace the REPLACE_WITH_YOUR_DATALAKE_NAME value with the name of your primary ADLS Gen2 account for your Synapse workspace. To find this, do the following:

1. Navigate to the **Data** hub.



2. Select the **Linked** tab (1), expand the Azure Data Lake Storage Gen2 group, then make note of the primary ADLS Gen2 name (2) next to the name of the workspace.

Data

Workspace

Linked

Filter resources by name

▲ Azure Data Lake Storage Gen2

- ▶ asagaworkspace356342 (Primary - asagadatalake356342) ... 2
- ▶ asagadata...
- ▶ **Integration**

Linked service name:
asagaworkspace356342-
WorkspaceDefaultStorage
Account name: Primary -
asagadatalake356342

7. Run the new cell. It will load the two DataFrames with data from the data lake and initialize Hyperspace.

```

1  from hyperspace import *
2  from com.microsoft.hyperspace import *
3  from com.microsoft.hyperspace.index import *
4
5  # Disable BroadcastHashJoin, so Spark will use standard SortMergeJoin. Currently, Hyperspace indexes utilize SortMergeJoin to speed up query.
6  spark.conf.set("spark.sql.autoBroadcastJoinThreshold", -1)
7
8  # Replace [ ] value below with the name of your primary ADLS Gen2 account for your Synapse workspace
9  datalake = "asagadatalake356342"
10
11 dfSales = spark.read.parquet("abfss://wri-02@{}+datalake+*.dfs.core.windows.net/sale-small/Year=2019/Quarter=Q4/Month=12/*.parquet")
12 dfSales.show(10)
13
14 dfCustomers = spark.read.load("abfss://wri-02@{}+datalake+*.dfs.core.windows.net/data-generators/generator-customer-clean.csv", format="csv", header=True)
15 dfCustomers.show(10)
16
17 # Create an instance of Hyperspace
18 hyperspace = Hyperspace(spark)

Command executed in 265 909ms by odl_user_356342 on 04-29-2021 19:25:36.307-0400
> Job execution Succeeded Spark 2 executors 8 cores

```

TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount	TransactionDate	ProfitAmount	Hour	Minute	StoreId
b49627a7-e663-4d8...	53	2999	2 32..3300000000000000.. [64..6600000000000000.. 289191202 19..5800000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	2377	3 29..5600000000000000.. [88..6800000000000000.. 289191202 22..8600000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	735	2 26..7900000000000000.. [53..5800000000000000.. 289191202 15..6900000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	2999	4 32..3200000000000000.. [129..3200000000000000.. 289191202 39..1600000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	1533	3 26..1100000000000000.. [78..3300000000000000.. 289191202 21..1800000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	1871	2 25..3500000000000000.. [50..7000000000000000.. 289191202 17..4400000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	3998	1 23..0700000000000000.. [23..0700000000000000.. 289191202 6..8200000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	1380	4 38..0300000000000000.. [152..1200000000000000.. 289191202 40..9200000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	4337	3 34..4000000000000000.. [103..2000000000000000.. 289191202 25..4400000000000000.. 8 35 8893							
b49627a7-e663-4d8...	53	4593	2 23..2700000000000000.. [46..5400000000000000.. 289191202 9..4600000000000000.. 8 35 8893							

only showing top 10 rows

CustomerId	FirstName	MiddleInitial	LastName	FullName	Gender	Age	BirthDate	Address_PostaCode	Address_Street	Address_City	Address_Country	Mobile	Email
1	Conrad	N	Zemlak	Conrad N. Zemlak	Male	57	1963-03-23	16219	Dario View	DocksSide	Suriname	0925-586-826	Conrad_Zemlak_850...
2	Daren	H	Rolfson	Daren H. Rolfson	Male	61	1959-05-25	43387	Kathleen Oval	Lake Rileyton	Tanzania	0372-253-946	Daren_Rolfson_20...
3	Annie	J	Pouros	Annie J. Pouros	Female	51	1969-04-23	92666	Haegan Hill	Leltachester	Virgin Islands Br...	0834-564-182	Annie_Pouros_48ey...
4	Forrest	null	O'Kon	Forrest O'Kon	Male	54	1966-01-31	29524	Harry Spring	New Gregor	Netherlands	0761-327-187	Forrest_O'Kon_83...
5	Leroy	null	Leinke	Leroy Leinke	Male	59	1961-03-01	36967	Layla Ford	east Davontown	Suriname	0264-161-776	Leroy_Linke_2960u...
6	Luz	R	Borer	Luz R. Borer	Female	53	1967-04-24	51858	Marjorie Overpass	Carleeshire	Burkina Faso	0642-895-538	Luz_Borer_77eout...
7	Ira	M	Koelpin	Ira M. Koelpin	Male	52	1968-04-22	69710	Kendall Street	New Lola	Suriname	0114-516-158	Ira_Koelpin_06gm...
8	Salvatore	null	Schaden	Salvatore Schaden	Male	54	1965-12-30	45947	Juanan Crest	Thompsonhaven	Liechtenstein	0782-995-529	Salvatore_Schaden...
9	Robyn	K	Batz	Robyn K. Batz	Female	59	1968-08-17	67226	Derrick Flat	Lake Joeymaren	Denmark	0729-374-418	Robyn_Batz_10@mna...
18	Brandi	D	Kertzmann	Brandi D. Kertzmann	Female	43	1976-12-20	44794	Haiden Track	East Hyramhaven	Moldova	0739-782-555	Brandi_Kertzmann...

only showing top 10 rows

Note: You may select the Run button to the left of the cell, or enter **Shift+Enter** to execute the cell and create a new cell below.

The first time you execute a cell in the notebook will take a few minutes since it must start a new Spark cluster. Each subsequent cell execution should be much faster.

8. Add a new code cell to your notebook with the following code:

```
#create indexes: each one contains a name, a set of indexed columns and a set of included columns
indexConfigSales = IndexConfig("indexSALES", ["CustomerId"], ["TotalAmount"])
indexConfigCustomers = IndexConfig("indexCUSTOMERS", ["CustomerId"], ["FullName"])

hyperspace.createIndex(dfSales, indexConfigSales) # only create index once
```

```
hyperspace.createIndex(dfCustomers, indexConfigCustomers) # only create index once
hyperspace.indexes().show()
```

9. Run the new cell. It will create two indexes and display their structure.

```
[2]
1 #create indexes: each one contains a name, a set of indexed columns and a set of included columns
2 indexConfigSales = IndexConfig("indexSALES", ["CustomerId"], ["TotalAmount"])
3 indexConfigCustomers = IndexConfig("indexCUSTOMERS", ["CustomerId"], ["FullName"])
4
5 hyperspace.createIndex(dfSales, indexConfigSales)      # only create index once
6 hyperspace.createIndex(dfCustomers, indexConfigCustomers) # only create index once
7 hyperspace.indexes().show()
```

Command executed in 37s 440ms by odl_user_356342 on 04-29-2021 19:44:09.607 -04:00

> Job execution Succeeded Spark 2 executors 8 cores

name	indexedColumns	includedColumns	numBuckets	schema	indexLocation	state
indexCUSTOMERS	[CustomerId]	[FullName]	200	{"type": "struct", ...}	abfss://workspace...	ACTIVE
indexSALES	[CustomerId]	[TotalAmount]	200	{"type": "struct", ...}	abfss://workspace...	ACTIVE

10. Add another new code cell to your notebook with the following code:

```
df1 = dfSales.filter("""CustomerId = 200""").select("""TotalAmount""")
df1.show()
df1.explain(True)
```

11. Run the new cell. The output will show that the physical execution plan is not taking into account any of the indexes (performs a file scan on the original data file).

```
1 df1 = dfSales.filter("""CustomerId = 203""").select("""TotalAmount""")
2 df1.show()
3 df1.explain(True)

Command executed in 15s 308ms by ciprian on 11-30-2020 23:50:33.836 +02:00
View in monitoring Open Spark UI
```

> Job execution Succeeded Spark 2 executors 8 cores

TotalAmount
28.33000000000000...
110.9100000000000...
95.7200000000000...
103.8400000000000...
103.8400000000000...
92.2800000000000...
31.2800000000000...
31.2800000000000...

```
-- Parsed Logical Plan ==
'Project [unresolvedalias('TotalAmount, None)]
+- Filter (CustomerId#28 = 203)
   +- Relation[TransactionId#27,CustomerId#28,ProductId#29,Quantity#30,Price#31,TotalAmount#32,TransactionDate#33,ProfitAmount#34,Hour#35,Minute#36,StoreId#37] parquet

-- Analyzed Logical Plan ==
Totalamount: decimal(38,18)
Project [TotalAmount#32]
+- Filter (CustomerId#28 = 203)
   +- Relation[TransactionId#27,CustomerId#28,ProductId#29,Quantity#30,Price#31,TotalAmount#32,TransactionDate#33,ProfitAmount#34,Hour#35,Minute#36,StoreId#37] parquet

-- Optimized Logical Plan ==
Project [TotalAmount#32]
+- Filter (CustomerId#28 & (CustomerId#28 = 203))
   +- Relation[TransactionId#27,CustomerId#28,ProductId#29,Quantity#30,Price#31,TotalAmount#32,TransactionDate#33,ProfitAmount#34,Hour#35,Minute#36,StoreId#37] parquet

-- Physical Plan ==
*(1) Project [TotalAmount#32]
+- *(1) FileScan parquet [CustomerId#28,TotalAmount#32] Batched: true, Format: Parquet, Location: InMemoryFileIndex[abfss://wci-02@asadatalake01.dfs.core.windows.net/sale-small/Year=2019/Quarter=..., PartitionFilters: [], PushedFilters: [IsNotNull(CustomerId), EqualTo(CustomerId,203)], ReadSchema: struct<CustomerId:int,TotalAmount:decimal(38,18)>]
```

12. Now add another new cell to your notebook with the following code (notice the extra line at the beginning used to enable Hyperspace optimization in the Spark engine):

```
# Enable Hyperspace - Hyperspace optimization rules become visible to the Spark optimizer and explain
Hyperspace.enable(spark)
df1 = dfSales.filter("""CustomerId = 200""").select("""TotalAmount""")
df1.show()
df1.explain(True)
```

13. Run the new cell. The output will show that the physical execution plan is now using the index instead of the original data file.

```

1 # Enable Hypespace - Hypespace optimization rules become visible to the Spark optimizer and exploit existing Hypespace indexes to optimize user queries ...
2 Hypespace.enable(spark)
3 df1 = dfSales.filter("""CustomerId = 203""").select("""TotalAmount""")
4 df1.show()
5 df1.explain(True)

Command executed in 13s417ms by ciprian on 11-30-2020 23:54:56.589 +02:00
> Job execution Succeeded Spark 2 executors 8 cores
View in monitoring Open Spark UI


+-----+
| TotalAmount |
+-----+
| 28.33000000000000..|
| 110.310000000000..|
| 95.72000000000000..|
| 103.84000000000000..|
| 133.32000000000000..|
| 92.28000000000000..|
| 51.20000000000000..|
+-----+

== Parsed Logical Plan ==
Project [TotalAmount#32]
+> Filter (CustomerId#28 = 203)
+> Relation[TransactionId#27,CustomerId#28,ProductId#29,Quantity#30,Price#31,TotalAmount#32,TransactionDate#33,ProfitAmount#34,Hour#35,Minute#36,StoreId#37] parquet

== Analyzed Logical Plan ==
TotalAmount:decimal(38,10)
Project [TotalAmount#32]
+> Filter (CustomerId#28 = 203)
+> Relation[TransactionId#27,CustomerId#28,ProductId#29,Quantity#30,Price#31,TotalAmount#32,TransactionDate#33,ProfitAmount#34,Hour#35,Minute#36,StoreId#37] parquet

== Optimized Logical Plan ==
Project [TotalAmount#32]
+> Filter (isNotNull(CustomerId#28) && (CustomerId#28 = 203))
+> Relation[CustomerId#28,TotalAmount#32] parquet

== Physical Plan ==
*(1) Project [TotalAmount#32]
+> *(1) FileScan parquet [CustomerId#28,TotalAmount#32] Batched: true, Format: Parquet, Location: InMemoryFileIndex[abfss://workspace@asagadatalake01.dfs.core.windows.net/synapse/workspaces/asaga...], PartitionFilters: []
PushedFilters: [isNotNull(CustomerId), EqualTo(CustomerId,203)], ReadSchema: struct<CustomerID:int,TotalAmount:decimal(38,10)>


```

14. Hypespace provides an Explain API that allows you to compare the execution plans without indexes vs. with indexes. Add a new cell with the following code:

```

df1 = dfSales.filter("""CustomerId = 200""").select("""TotalAmount""")

spark.conf.set("spark.hyperspace.explain.displayMode", "html")
hyperspace.explain(df1, True, displayHTML)

```

15. Run the new cell. The output shows a comparison **Plan with indexes** vs. **Plan without indexes**. Observe how, in the first case the index file is used while in the second case the original data file is used.

```

1 df1 = dfSales.filter("""CustomerId = 203""").select("""TotalAmount""")
2
3 spark.conf.set("spark.hyperspace.explain.displayMode", "html")
4 hyperspace.explain(df1, True, displayHTML)

Command executed in 3s924ms by ciprian on 12-01-2020 00:00:07.054 +02:00
> Job execution Succeeded Spark 2 executors 8 cores
View in monitoring Open Spark UI


+-----+
| TotalAmount |
+-----+
| 28.33000000000000..|
| 110.310000000000..|
| 95.72000000000000..|
| 103.84000000000000..|
| 133.32000000000000..|
| 92.28000000000000..|
| 51.20000000000000..|
+-----+

==== Plan with indexes: ====
==== Plan without indexes: ====
==== Indexes used: ====
indexSALES:abfss://workspace@asagadatalake01.dfs.core.windows.net/synapse/workspaces/asagaworkspace01/warehouse/indexSALES/v__=0

==== Physical operator stats: ====
Physical Operator: Hyperspace Disabled/Hyperspace Enabled Difference
+-----+
| Physical Operator | Hyperspace Disabled | Hyperspace Enabled | Difference |
+-----+
| Filter | 1 | 1 | 0 |
| Project | 1 | 1 | 0 |
| Scan parquet | 1 | 1 | 0 |
| WholeStageCodegen | 1 | 1 | 0 |
+-----+


```

16. Let's investigate now a more complex case, involving a join operation. Add a new cell with the following code:

```

eqJoin = dfSales.join(dfCustomers, dfSales.CustomerId == dfCustomers.CustomerId).select(dfSales.T
hyperspace.explain(eqJoin, True, displayHTML)

```

17. Run the new cell. The output shows again a comparison **Plan with indexes** vs. **Plan without indexes**, where indexes are used in the first case and the original data files in the second.

```

1   eqJoin = dfSales.join(dfCustomers, dfSales.CustomerId == dfCustomers.CustomerId).select(dfSales.TotalAmount, dfCustomers.FullName)
2
3   hyperspace.explain(eqJoin, True, displayHTML)

```

Command executed in 5s 61ms by opan on 12-01-2020 00:21:16,535 +0200
> Job execution Succeeded Spark 2 executors 8 cores
View in monitoring Open

Plan with indexes:

```

Project [TotalAmount#32, FullName#97]
+-- SortMergeJoin [Customer#98, dfSales#100] on cast(CustomerId#98 as int), Inner
|   +- V2_Exchange_haskpartitoning[Customer#98, NULLS FIRST], False, 0
|   +- Exchange_haskpartitoning[Customer#98, ANS NULLS FIRST], False, 0
|   +- *{1} Filter isNotNull([CustomerId#98])
|   +- *{2} FileScan parquet [dfSales#100, TotalAmount#32] Batched: true, Format: Parquet, Location: InMemoryfileIndex[abfss://workspace@asagdatalake01.dfs.core.windows.net/synapse/asag...], PartitionFilters: [], PushedFilters: [isNotNull(CustomerId)], ReadSchema: struct<name: string, id: int, TotalAmount: double>
|   +- *{3} SortMergeJoin [Customer#98, dfSales#100] on cast(CustomerId#98 as int), False, 0
|   +- Exchange_haskpartitoning[cast(CustomerId#98 as int), 200], [id=41493]
|   +- *{4} Project [cast(CustomerId#98, FullName#97)]
|   +- *{5} FileScan parquet [Customer#98, FullName#97] Batched: true, Format: Parquet, Location: InMemoryfileIndex[abfss://workspace@asagdatalake01.dfs.core.windows.net/synapse/asag...], PartitionFilters: [], PushedFilters: [isNotNull(CustomerId)], ReadSchema: struct<name: string, id: int, FullName: string>

```

Plan without indexes:

```

Project [TotalAmount#32, FullName#97]
+-- SortMergeJoin [Customer#98, dfSales#100] on cast(CustomerId#98 as int), Inner
|   +- V2_Exchange_haskpartitoning[Customer#98, NULLS FIRST], False, 0
|   +- Exchange_haskpartitoning[Customer#98, ANS NULLS FIRST], False, 0
|   +- *{1} Filter isNotNull([CustomerId#98])
|   +- *{2} FileScan parquet [Customer#98, dfSales#100, TotalAmount#32] Batched: true, Format: Parquet, Location: InMemoryfileIndex[abfss://uni-02@asagdatalake01.dfs.core.windows.net/sale-snell/Year=2019/Quarter=...], PartitionFilters: [], PushedFilters: [isNotNull(CustomerId)], ReadSchema: struct<name: string, id: int, TotalAmount: double>
|   +- *{3} SortMergeJoin [Customer#98, dfSales#100] on cast(CustomerId#98 as int), False, 0
|   +- Exchange_haskpartitoning[cast(CustomerId#98 as int), 200], [id=41496]
|   +- *{4} Project [cast(CustomerId#98, FullName#97)]
|   +- *{5} FileScan csv [Customer#98, FullName#97] Batched: false, Format: CSV, Location: InMemoryfileIndex[abfss://uni-02@asagdatalake01.dfs.core.windows.net/data-generators/generator-cus...], PartitionFilters: [], PushedFilters: [isNotNull(CustomerId)], ReadSchema: struct<name: string, id: int, FullName: string>

```

Indexes used:

```

indexCUSTOMERS:abfss://workspace@asagdatalake01.dfs.core.windows.net/synapse/asagworkspace01/warehouse/indexes/indexCUSTOMERS/v_...
indexSALES:abfss://workspace@asagdatalake01.dfs.core.windows.net/synapse/asagworkspace01/warehouse/indexes/indexSALES/v_...

```

Physical operator status:

```

+-----+-----+
| Operator | Hyperspace Enabled/Hyperspace Enabled |
+-----+-----+
| Scan csv | 1 | 0 | -1 |
| Scan parquet | 1 | 1 | 1 |
| Filter | 2 | 1 | 0 |
| InputFile | 3 | 0 | 0 |
| Project | 4 | 0 | 0 |
| ShuffleExchange | 5 | 0 | 0 |
| Sort | 6 | 0 | 0 |
| SortMergeJoin | 7 | 1 | 0 |
| WholeStageCodegen | 8 | 0 | 0 |
+-----+-----+

```

In case you want to deactivate Hyperspace and cleanup the indexes, you can run the following code:

```
# Disable Hyperspace - Hyperspace rules no longer apply during query optimization. Disabling Hyperspace.disable(spark)
```

```
hyperspace.deleteIndex("indexSALES")
hyperspace.vacuumIndex("indexSALES")
hyperspace.deleteIndex("indexCUSTOMERS")
hyperspace.vacuumIndex("indexCUSTOMERS")
```

3.3.2.2 Task 2: Explore the Data Lake storage with the MSSparkUtil library

Microsoft Spark Utilities (MSSparkUtils) is a builtin package to help you easily perform common tasks. You can use MSSparkUtils to work with file systems, to get environment variables, and to work with secrets.

1. Continue with the same notebook from the previous task and add a new cell with the following code:

```
from notebookutils import mssparkutils

#
# Microsoft Spark Utilities
#
# https://docs.microsoft.com/en-us/azure/synapse-analytics/spark/microsoft-spark-utilities?pivotst...
#


# Azure storage access info
blob_account_name = datalake
blob_container_name = 'wwi-02'
blob_relative_path = '/'
linkedServiceName = datalake
blob_sas_token = mssparkutils.credentials.getConnectionStringOrCreds(linkedServiceName)

# Allow SPARK to access from Blob remotely
spark.conf.set('fs.azure.sas.%s.%s.blob.core.windows.net' % (blob_container_name, blob_account_name))

files = mssparkutils.fs.ls('/')
for file in files:
    print(file.name, file.isdir, file.isfile, file.path, file.size)

mssparkutils.fs.mkdirs('/SomeNewFolder')

files = mssparkutils.fs.ls('/')
```

```

for file in files:
    print(file.name, file.isDir, file.isFile, file.path, file.size)

```

- Run the new cell and observe how `mssparkutils` is used to work with the file system.

3.3.3 Resources

To learn more about the topics covered in this lab, use these resources:

- Apache Spark in Azure Synapse Analytics
- Announcing Azure Data Explorer data connector for Azure Synapse
- Connect to Azure Data Explorer using Apache Spark for Azure Synapse Analytics
- Azure Synapse Analytics shared metadata
- Introduction of Microsoft Spark Utilities
- Hyperspace - An open source indexing subsystem that brings index-based query acceleration to Apache Spark™ and big data workloads

4 Module 2 - Design and Implement the serving layer

This module teaches how to design and implement data stores in a modern data warehouse to optimize analytical workloads. The student will learn how to design a multidimensional schema to store fact and dimension data. Then the student will learn how to populate slowly changing dimensions through incremental data loading from Azure Data Factory.

In this module, the student will be able to:

- Design a star schema for analytical workloads (OLAP)
- Populate slowly changing dimensions with Azure Data Factory and mapping data flows

4.1 Lab details

- Module 2 - Design and Implement the serving layer
 - Lab details
 - Lab setup and pre-requisites
 - Exercise 0: Start the dedicated SQL pool
 - Exercise 1: Implementing a Star Schema
 - Task 1: Create star schema in SQL database
 - Exercise 2: Implementing a Snowflake Schema
 - Task 1: Create product snowflake schema in SQL database
 - Task 2: Create reseller snowflake schema in SQL database
 - Exercise 3: Implementing a Time Dimension Table
 - Task 1: Create time dimension table
 - Task 2: Populate the time dimension table
 - Task 3: Load data into other tables
 - Task 4: Query data
 - Exercise 4: Implementing a Star Schema in Synapse Analytics
 - Task 1: Create star schema in Synapse dedicated SQL
 - Task 2: Load data into Synapse tables
 - Task 3: Query data from Synapse
 - Exercise 5: Updating slowly changing dimensions with mapping data flows
 - Task 1: Create the Azure SQL Database linked service
 - Task 2: Create a mapping data flow
 - Task 3: Create a pipeline and run the data flow
 - Task 4: View inserted data
 - Task 5: Update a source customer record
 - Task 6: Re-run mapping data flow
 - Task 7: Verify record updated
 - Exercise 6: Cleanup
 - Task 1: Pause the dedicated SQL pool

4.1.1 Lab setup and pre-requisites

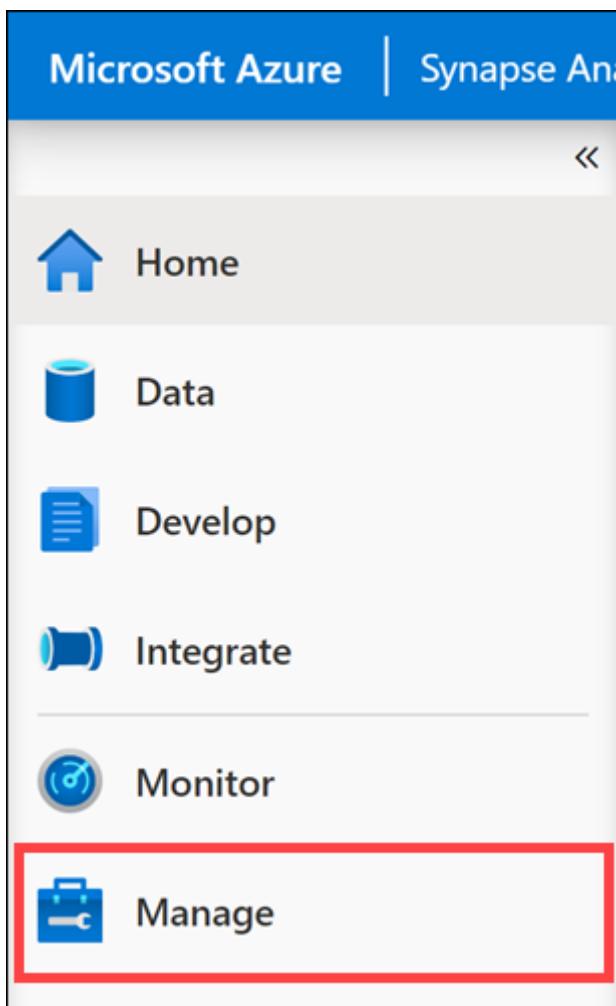
Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

1. If you have not already, follow the [lab setup instructions](#) for this module.
2. Install [Azure Data Studio](#) on your computer or lab virtual machine.

4.2 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The serverless SQL pool, Built-in, is immediately available for your workspace. Dedicated SQL pools can be configured to adapt to team or organizational needs.

SQL pools

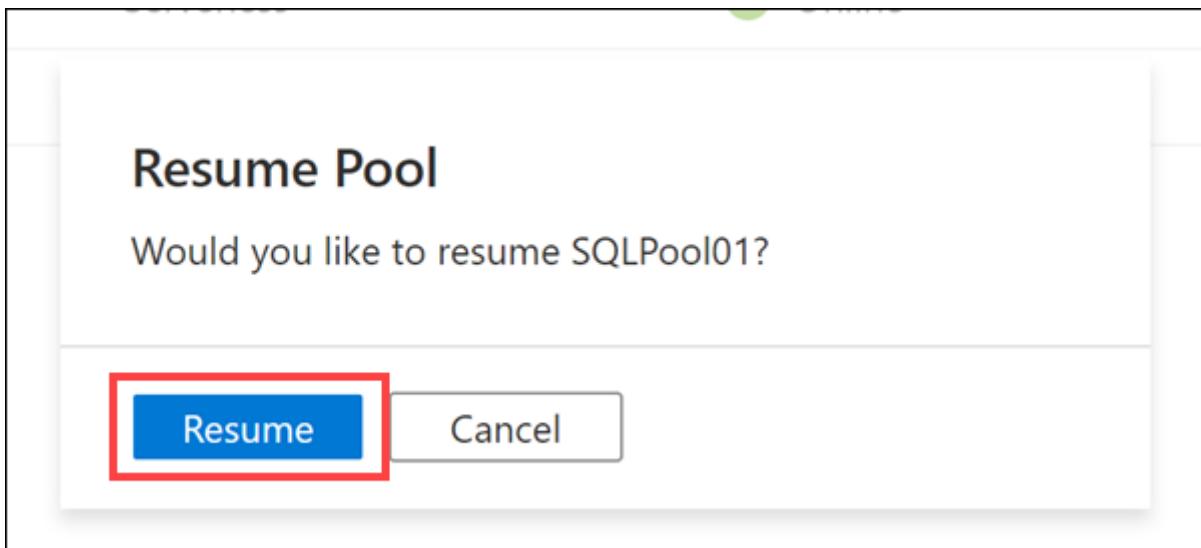
+ New Refresh System-assigned managed identity

Showing 1-2 of 2 items (1 Serverless, 1 Dedicated)

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused	DW100c

Resume

- When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

4.3 Exercise 1: Implementing a Star Schema

Star schema is a mature modeling approach widely adopted by relational data warehouses. It requires modelers to classify their model tables as either dimension or fact.

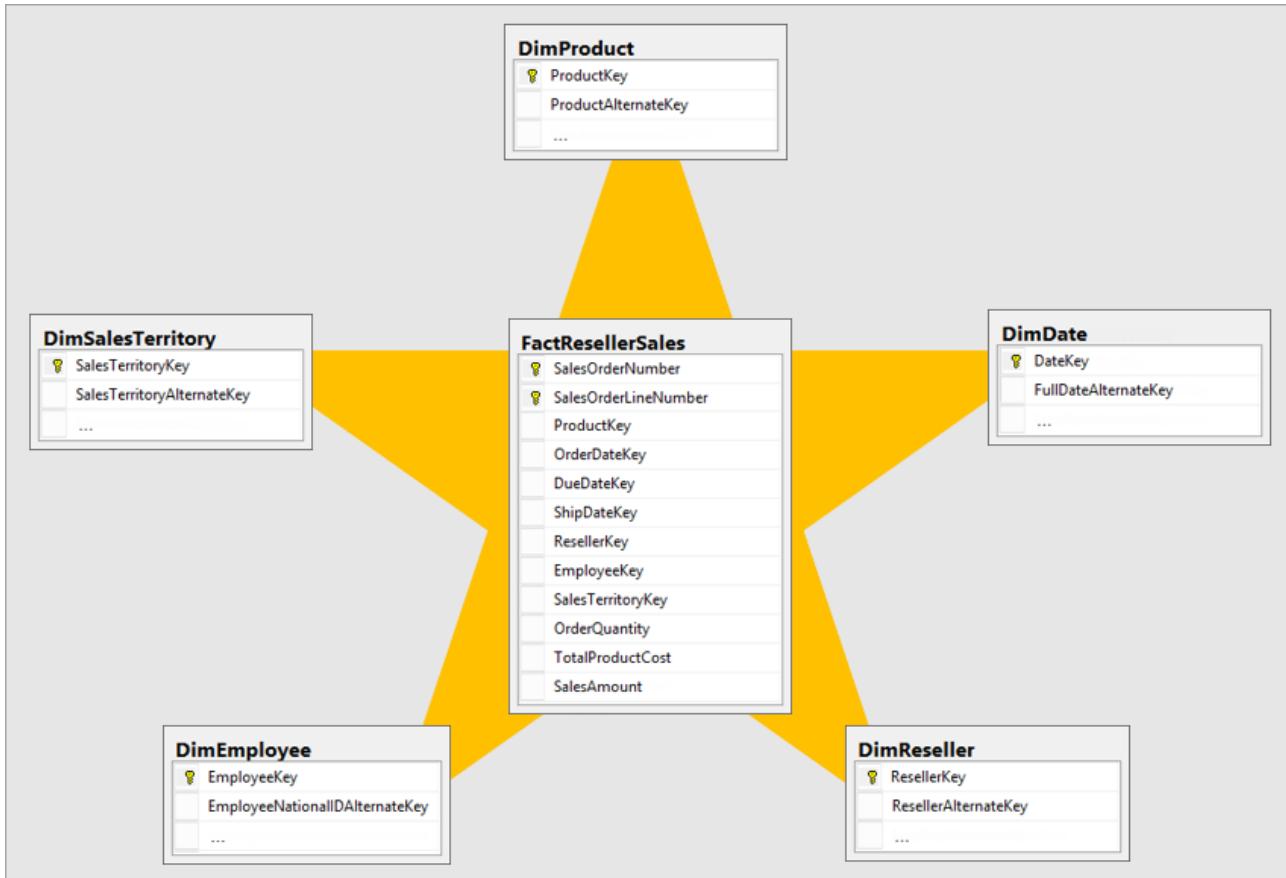
Dimension tables describe business entities—the things you model. Entities can include products, people, places, and concepts including time itself. The most consistent table you'll find in a star schema is a date dimension table. A dimension table contains a key column (or columns) that acts as a unique identifier, and descriptive columns.

Dimension tables contain attribute data that might change but usually changes infrequently. For example, a customer's name and address are stored in a dimension table and updated only when the customer's profile changes. To minimize the size of a large fact table, the customer's name and address don't need to be in every row of a fact table. Instead, the fact table and the dimension table can share a customer ID. A query can join the two tables to associate a customer's profile and transactions.

Fact tables store observations or events, and can be sales orders, stock balances, exchange rates, temperatures, etc. A fact table contains dimension key columns that relate to dimension tables, and numeric measure columns. The dimension key columns determine the dimensionality of a fact table, while the dimension key values determine the granularity of a fact table. For example, consider a fact table designed to store sale targets that has two dimension key columns **Date** and **ProductKey**. It's easy to understand that the table has two dimensions. The granularity, however, can't be determined without considering the dimension key values. In this example, consider that the values stored in the Date column are the first day of each month. In this case, the granularity is at month-product level.

Generally, dimension tables contain a relatively small number of rows. Fact tables, on the other hand, can contain a very large number of rows and continue to grow over time.

Below is an example star schema, where the fact table is in the middle, surrounded by dimension tables:



4.3.1 Task 1: Create star schema in SQL database

In this task, you create a star schema in SQL database, using foreign key constraints. The first step is to create the base dimension and fact tables.

1. Sign in to the Azure portal (<https://portal.azure.com>).
2. Open the resource group for this lab, then select the **SourceDB** SQL database.

Name ↑↓	Type ↑↓
<input type="checkbox"/> asagadatalakejdh013121	Storage account
<input type="checkbox"/> asagakeyvaultjdh013121	Key vault
<input type="checkbox"/> asagaworkspacejdh013121	Synapse workspace
<input type="checkbox"/> dp203sqljdh013121	SQL server
<input type="checkbox"/> SourceDB (dp203sqljdh013121/SourceDB)	SQL database
<input type="checkbox"/> SQLPool01 (asagaworkspacejdh013121/SQLPool01)	Dedicated SQL pool

3. Copy the **Server name** value on the Overview pane.

SourceDB (dp203sqljdh013121/SourceDB)

SQL database

Search (Ctrl+)

Copy Restore Export Set server firewall Delete Connect with... Feedback

Overview Activity log Tags Diagnose and solve problems Quick start

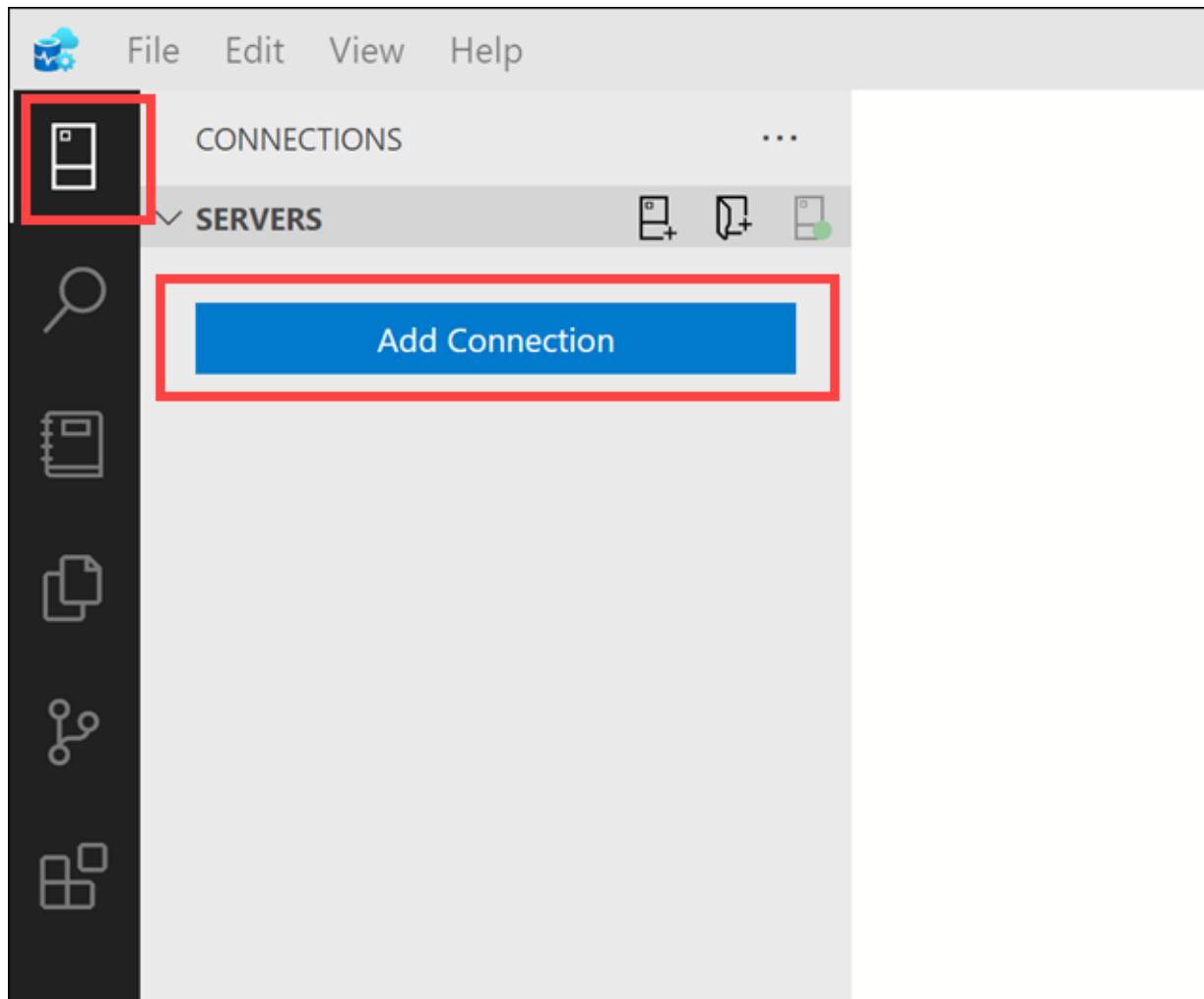
Resource group (change) : dp203-labs Status : Online Location : South Central US Subscription (change) : Show database connection strings Subscription ID : 00000000-0000-0000-0000-000000000000

Server name : dp203sqljdh013121.database.windows.net

Elastic pool : No elastic pool Connection strings : Show database connection strings Pricing tier : General Purpose: Gen5, 2 vCores Earliest restore point : 2021-01-31 17:36 UTC

4. Open Azure Data Studio.

5. Select **Servers** on the left-hand menu, then click **Add Connection**.



6. In the Connection Details form, provide the following information:

- **Server:** Paste the SourceDB server name value here.
- **Authentication type:** Select SQL Login.
- **User name:** Enter sqladmin.
- **Password:** Enter the password you supplied when deploying the lab environment, or which was provided to you as part of your hosted lab environment.
- **Remember password:** Checked.
- **Database:** Select SourceDB.

Connection

Recent Connections [Saved Connections](#) [Browse \(Preview\)](#)

No recent connection

Connection Details

Connection type

Microsoft SQL Server

Server

dp203sqljdh013121.database.windows.net

Authentication type

SQL Login

User name

sqladmin

Password

Remember password

Database

SourceDB

Server group

<Default>

Name (optional)

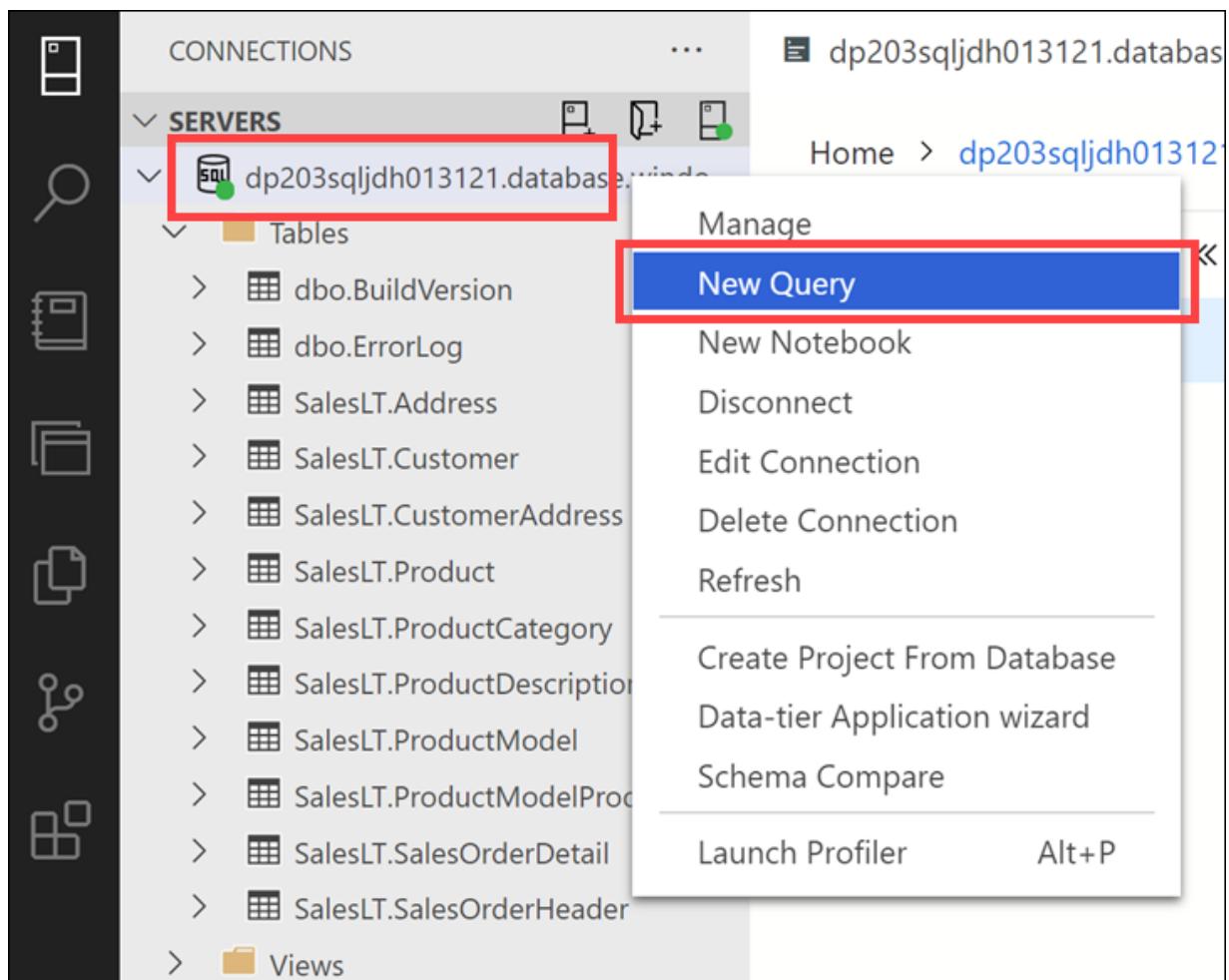
[Advanced...](#)

Connect

Cancel

7. Select **Connect**.

8. Select **Servers** in the left-hand menu, then right-click the SQL server you added at the beginning of the lab. Select **New Query**.



- Paste the following into the query window to create the dimension and fact tables:

```

CREATE TABLE [dbo].[DimReseller] (
    [ResellerKey] [int] IDENTITY(1,1) NOT NULL,
    [GeographyKey] [int] NULL,
    [ResellerAlternateKey] [nvarchar](15) NULL,
    [Phone] [nvarchar](25) NULL,
    [BusinessType] [varchar](20) NOT NULL,
    [ResellerName] [nvarchar](50) NOT NULL,
    [NumberEmployees] [int] NULL,
    [OrderFrequency] [char](1) NULL,
    [OrderMonth] [tinyint] NULL,
    [FirstOrderYear] [int] NULL,
    [LastOrderYear] [int] NULL,
    [ProductLine] [nvarchar](50) NULL,
    [AddressLine1] [nvarchar](60) NULL,
    [AddressLine2] [nvarchar](60) NULL,
    [AnnualSales] [money] NULL,
    [BankName] [nvarchar](50) NULL,
    [MinPaymentType] [tinyint] NULL,
    [MinPaymentAmount] [money] NULL,
    [AnnualRevenue] [money] NULL,
    [YearOpened] [int] NULL
);
GO

CREATE TABLE [dbo].[DimEmployee] (
    [EmployeeKey] [int] IDENTITY(1,1) NOT NULL,
    [ParentEmployeeKey] [int] NULL,
    [EmployeeNationalIDAlternateKey] [nvarchar](15) NULL,
    ...
);
GO

```

```

[ParentEmployeeNationalIDAlternateKey] [nvarchar](15) NULL,
[SalesTerritoryKey] [int] NULL,
[FirstName] [nvarchar](50) NOT NULL,
[LastName] [nvarchar](50) NOT NULL,
[MiddleName] [nvarchar](50) NULL,
[NameStyle] [bit] NOT NULL,
[Title] [nvarchar](50) NULL,
[HireDate] [date] NULL,
[BirthDate] [date] NULL,
[LoginID] [nvarchar](256) NULL,
[EmailAddress] [nvarchar](50) NULL,
[Phone] [nvarchar](25) NULL,
[MaritalStatus] [nchar](1) NULL,
[EmergencyContactName] [nvarchar](50) NULL,
[EmergencyContactPhone] [nvarchar](25) NULL,
[SalariedFlag] [bit] NULL,
[Gender] [nchar](1) NULL,
[PayFrequency] [tinyint] NULL,
[BaseRate] [money] NULL,
[VacationHours] [smallint] NULL,
[SickLeaveHours] [smallint] NULL,
[CurrentFlag] [bit] NOT NULL,
[SalesPersonFlag] [bit] NOT NULL,
[DepartmentName] [nvarchar](50) NULL,
[StartDate] [date] NULL,
[EndDate] [date] NULL,
[Status] [nvarchar](50) NULL,
[EmployeePhoto] [varbinary](max) NULL
);
GO

```

```

CREATE TABLE [dbo].[DimProduct](
    [ProductKey] [int] IDENTITY(1,1) NOT NULL,
    [ProductAlternateKey] [nvarchar](25) NULL,
    [ProductSubcategoryKey] [int] NULL,
    [WeightUnitMeasureCode] [nchar](3) NULL,
    [SizeUnitMeasureCode] [nchar](3) NULL,
    [EnglishProductName] [nvarchar](50) NOT NULL,
    [SpanishProductName] [nvarchar](50) NOT NULL,
    [FrenchProductName] [nvarchar](50) NOT NULL,
    [StandardCost] [money] NULL,
    [FinishedGoodsFlag] [bit] NOT NULL,
    [Color] [nvarchar](15) NOT NULL,
    [SafetyStockLevel] [smallint] NULL,
    [ReorderPoint] [smallint] NULL,
    [ListPrice] [money] NULL,
    [Size] [nvarchar](50) NULL,
    [SizeRange] [nvarchar](50) NULL,
    [Weight] [float] NULL,
    [DaysToManufacture] [int] NULL,
    [ProductLine] [nchar](2) NULL,
    [DealerPrice] [money] NULL,
    [Class] [nchar](2) NULL,
    [Style] [nchar](2) NULL,
    [ModelName] [nvarchar](50) NULL,
    [LargePhoto] [varbinary](max) NULL,
    [EnglishDescription] [nvarchar](400) NULL,
    [FrenchDescription] [nvarchar](400) NULL,
    [ChineseDescription] [nvarchar](400) NULL,
    [ArabicDescription] [nvarchar](400) NULL,
    [HebrewDescription] [nvarchar](400) NULL,
)

```

```

[ThaiDescription] [nvarchar](400) NULL,
[GermanDescription] [nvarchar](400) NULL,
[JapaneseDescription] [nvarchar](400) NULL,
[TurkishDescription] [nvarchar](400) NULL,
[StartDate] [datetime] NULL,
[EndDate] [datetime] NULL,
[Status] [nvarchar](7) NULL
);
GO

CREATE TABLE [dbo].[FactResellerSales](
[ProductKey] [int] NOT NULL,
[OrderDateKey] [int] NOT NULL,
[DueDateKey] [int] NOT NULL,
[ShipDateKey] [int] NOT NULL,
[ResellerKey] [int] NOT NULL,
[EmployeeKey] [int] NOT NULL,
[PromotionKey] [int] NOT NULL,
[CurrencyKey] [int] NOT NULL,
[SalesTerritoryKey] [int] NOT NULL,
[SalesOrderNumber] [nvarchar](20) NOT NULL,
[SalesOrderLineNumber] [tinyint] NOT NULL,
[RevisionNumber] [tinyint] NULL,
[OrderQuantity] [smallint] NULL,
[UnitPrice] [money] NULL,
[ExtendedAmount] [money] NULL,
[UnitPriceDiscountPct] [float] NULL,
[DiscountAmount] [float] NULL,
[ProductStandardCost] [money] NULL,
[TotalProductCost] [money] NULL,
[SalesAmount] [money] NULL,
[TaxAmt] [money] NULL,
[Freight] [money] NULL,
[CarrierTrackingNumber] [nvarchar](25) NULL,
[CustomerPONumber] [nvarchar](25) NULL,
[OrderDate] [datetime] NULL,
[DueDate] [datetime] NULL,
[ShipDate] [datetime] NULL
);
GO

```

10. Select **Run** or hit F5 to execute the query.

```

CREATE TABLE [dbo].[DimReseller](
    [ResellerKey] [int] IDENTITY(1,1) NOT NULL,
    [GeographyKey] [int] NULL,
    [ResellerAlternateKey] [nvarchar](15) NULL,
    [Phone] [nvarchar](25) NULL,
    [BusinessType] [varchar](20) NOT NULL,
    [ResellerName] [nvarchar](50) NOT NULL,
    [NumberEmployees] [int] NULL,
    [OrderFrequency] [char](1) NULL,
    [OrderMonth] [tinyint] NULL,
    [FirstOrderYear] [int] NULL,
    [LastOrderYear] [int] NULL,
    [ProductLine] [nvarchar](50) NULL,
    [AddressLine1] [nvarchar](60) NULL,
    [AddressLine2] [nvarchar](60) NULL,
    [AnnualSales] [money] NULL,
    [BankName] [nvarchar](50) NULL,
    [MinPaymentType] [tinyint] NULL,
    [MinPaymentAmount] [money] NULL,
    [AnnualRevenue] [money] NULL,
    [YearOpened] [int] NULL
);
GO
CREATE TABLE [dbo].[DimEmployee](

```

Now we have three dimension tables and a fact table. Together, these tables represent a star schema:

Column Name	Data Type	Allow Nulls
ProductKey	int	<input type="checkbox"/>
OrderDateKey	int	<input type="checkbox"/>
DueDateKey	int	<input type="checkbox"/>
ShipDateKey	int	<input type="checkbox"/>
ResellerKey	int	<input type="checkbox"/>
EmployeeKey	int	<input type="checkbox"/>
PromotionKey	int	<input type="checkbox"/>
CurrencyKey	int	<input type="checkbox"/>
SalesTerritoryKey	int	<input type="checkbox"/>
SalesOrderNumber	nvarchar(20)	<input type="checkbox"/>
SalesOrderLineNumber	tinyint	<input type="checkbox"/>
RevisionNumber	tinyint	<input checked="" type="checkbox"/>
OrderQuantity	smallint	<input checked="" type="checkbox"/>
UnitPrice	money	<input checked="" type="checkbox"/>
ExtendedAmount	money	<input checked="" type="checkbox"/>
UnitPriceDiscountPct	float	<input checked="" type="checkbox"/>
DiscountAmount	float	<input checked="" type="checkbox"/>
ProductStandardCost	money	<input checked="" type="checkbox"/>
TotalProductCost	money	<input checked="" type="checkbox"/>
SalesAmount	money	<input checked="" type="checkbox"/>
TaxAmt	money	<input checked="" type="checkbox"/>
Freight	money	<input checked="" type="checkbox"/>
CarrierTrackingNumber	nvarchar(25)	<input checked="" type="checkbox"/>
CustomerPONumber	nvarchar(25)	<input checked="" type="checkbox"/>
OrderDate	datetime	<input checked="" type="checkbox"/>
DueDate	datetime	<input checked="" type="checkbox"/>
ShipDate	datetime	<input checked="" type="checkbox"/>

Column Name	Data Type	Allow Nulls
ResellerKey	int	<input type="checkbox"/>
GeographyKey	int	<input checked="" type="checkbox"/>
ResellerAlternateKey	nvarchar(15)	<input checked="" type="checkbox"/>
Phone	nvarchar(25)	<input checked="" type="checkbox"/>
BusinessType	varchar(20)	<input type="checkbox"/>
ResellerName	nvarchar(50)	<input type="checkbox"/>
NumberEmployees	int	<input checked="" type="checkbox"/>
OrderFrequency	char(1)	<input checked="" type="checkbox"/>
OrderMonth	tinyint	<input checked="" type="checkbox"/>
FirstOrderYear	int	<input checked="" type="checkbox"/>
LastOrderYear	int	<input checked="" type="checkbox"/>
ProductLine	nvarchar(50)	<input checked="" type="checkbox"/>
AddressLine1	nvarchar(60)	<input checked="" type="checkbox"/>
AddressLine2	nvarchar(60)	<input checked="" type="checkbox"/>
AnnualSales	money	<input checked="" type="checkbox"/>
BankName	nvarchar(50)	<input checked="" type="checkbox"/>
MinPaymentType	tinyint	<input checked="" type="checkbox"/>
MinPaymentAmount	money	<input checked="" type="checkbox"/>
AnnualRevenue	money	<input checked="" type="checkbox"/>
YearOpened	int	<input checked="" type="checkbox"/>

Column Name	Data Type	Allow Nulls
EmployeeKey	int	<input type="checkbox"/>
ParentEmployeeKey	int	<input checked="" type="checkbox"/>
EmployeeNationalityAlt...	nvarchar(15)	<input checked="" type="checkbox"/>
ParentEmployeeNationalit...	nvarchar(15)	<input checked="" type="checkbox"/>
SalesTerritoryKey	int	<input checked="" type="checkbox"/>
FirstName	nvarchar(50)	<input type="checkbox"/>
LastName	nvarchar(50)	<input type="checkbox"/>
MiddleName	nvarchar(50)	<input checked="" type="checkbox"/>
NameStyle	bit	<input type="checkbox"/>
Title	nvarchar(50)	<input checked="" type="checkbox"/>
HireDate	date	<input checked="" type="checkbox"/>
BirthDate	date	<input checked="" type="checkbox"/>
LoginID	nvarchar(256)	<input checked="" type="checkbox"/>
EmailAddress	nvarchar(50)	<input checked="" type="checkbox"/>
Phone	nvarchar(25)	<input checked="" type="checkbox"/>
MaritalStatus	nchar(1)	<input checked="" type="checkbox"/>
EmergencyContactName	nvarchar(50)	<input checked="" type="checkbox"/>
EmergencyContactPhone	nvarchar(25)	<input checked="" type="checkbox"/>
SalaryFlag	bit	<input checked="" type="checkbox"/>
Gender	nchar(1)	<input checked="" type="checkbox"/>
PayFrequency	tinyint	<input checked="" type="checkbox"/>
BaseRate	money	<input checked="" type="checkbox"/>
VacationHours	smallint	<input checked="" type="checkbox"/>
SickLeaveHours	smallint	<input checked="" type="checkbox"/>
CurrentFlag	bit	<input type="checkbox"/>
SalesPersonFlag	bit	<input type="checkbox"/>
DepartmentName	nvarchar(50)	<input checked="" type="checkbox"/>
StartDate	date	<input checked="" type="checkbox"/>
EndDate	date	<input checked="" type="checkbox"/>
Status	nvarchar(50)	<input checked="" type="checkbox"/>

Column Name	Data Type	Allow Nulls
ProductKey	int	<input type="checkbox"/>
ProductAlternateKey	nvarchar(25)	<input checked="" type="checkbox"/>
ProductSubcategoryKey	int	<input checked="" type="checkbox"/>
WeightUnitMeasureCode	nchar(3)	<input checked="" type="checkbox"/>
SizeUnitMeasureCode	nchar(3)	<input checked="" type="checkbox"/>
EnglishProductName	nvarchar(50)	<input type="checkbox"/>
SpanishProductName	nvarchar(50)	<input type="checkbox"/>
FrenchProductName	nvarchar(50)	<input type="checkbox"/>
StandardCost	money	<input checked="" type="checkbox"/>
FinishedGoodFlag	bit	<input type="checkbox"/>
Color	nvarchar(15)	<input type="checkbox"/>
SafetyStockLevel	smallint	<input checked="" type="checkbox"/>
ReorderPoint	smallint	<input checked="" type="checkbox"/>
ListPrice	money	<input checked="" type="checkbox"/>
Size	nvarchar(50)	<input checked="" type="checkbox"/>
SizeRange	nvarchar(50)	<input checked="" type="checkbox"/>
Weight	float	<input checked="" type="checkbox"/>
DaysToManufacture	int	<input checked="" type="checkbox"/>
ProductLine	nchar(2)	<input checked="" type="checkbox"/>
DealerPrice	money	<input checked="" type="checkbox"/>
Class	nchar(2)	<input checked="" type="checkbox"/>
Style	nchar(2)	<input checked="" type="checkbox"/>
ModelName	nvarchar(50)	<input checked="" type="checkbox"/>
LargePhoto	varbinary(MAX)	<input checked="" type="checkbox"/>
EnglishDescription	nvarchar(400)	<input checked="" type="checkbox"/>
FrenchDescription	nvarchar(400)	<input checked="" type="checkbox"/>
ChineseDescription	nvarchar(400)	<input checked="" type="checkbox"/>
ArabicDescription	nvarchar(400)	<input checked="" type="checkbox"/>
HebrewDescription	nvarchar(400)	<input checked="" type="checkbox"/>
ThaiDescription	nvarchar(400)	<input checked="" type="checkbox"/>
GermanDescription	nvarchar(400)	<input checked="" type="checkbox"/>
JapaneseDescription	nvarchar(400)	<input checked="" type="checkbox"/>
TurkishDescription	nvarchar(400)	<input checked="" type="checkbox"/>
StartDate	datetime	<input checked="" type="checkbox"/>
EndDate	datetime	<input checked="" type="checkbox"/>
Status	nvarchar(7)	<input checked="" type="checkbox"/>

However, since we are using a SQL database, we can add foreign key relationships and constraints to define relationships and enforce the table values.

- Replace **and execute** the query with the following to create the DimReseller primary key and constraints:

```

-- Create DimReseller PK
ALTER TABLE [dbo].[DimReseller] WITH CHECK ADD
CONSTRAINT [PK_DimReseller_ResellerKey] PRIMARY KEY CLUSTERED
(
    [ResellerKey]
) ON [PRIMARY];
GO

-- Create DimReseller unique constraint
ALTER TABLE [dbo].[DimReseller] ADD CONSTRAINT [AK_DimReseller_ResellerAlternateKey] UNIQUE NONCLUSTERED
(
    [ResellerAlternateKey] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, 
GO

```

12. Replace **and execute** the query with the following to create the DimEmployee primary key:

```
-- Create DimEmployee PK
ALTER TABLE [dbo].[DimEmployee] WITH CHECK ADD
    CONSTRAINT [PK_DimEmployee_EmployeeKey] PRIMARY KEY CLUSTERED
    (
        [EmployeeKey]
    ) ON [PRIMARY];
GO
```

13. Replace **and execute** the query with the following to create the DimProduct primary key and constraints:

```
-- Create DimProduct PK
ALTER TABLE [dbo].[DimProduct] WITH CHECK ADD
    CONSTRAINT [PK_DimProduct_ProductKey] PRIMARY KEY CLUSTERED
    (
        [ProductKey]
    ) ON [PRIMARY];
GO

-- Create DimProduct unique constraint
ALTER TABLE [dbo].[DimProduct] ADD CONSTRAINT [AK_DimProduct_ProductAlternateKey_StartDate] UNIQUE
(
    [ProductAlternateKey] ASC,
    [StartDate] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, )
GO
```

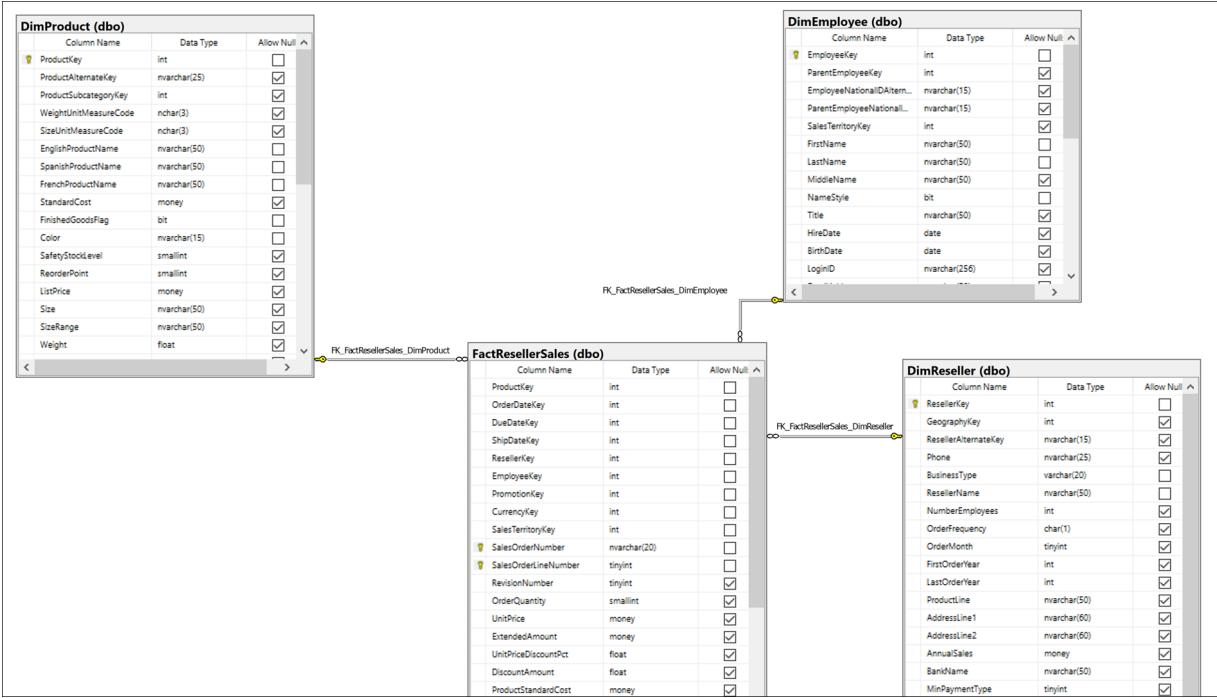
Now we can create the relationships between our fact and dimension tables, clearly defining the star schema.

14. Replace **and execute** the query with the following to create the FactResellerSales primary key and foreign key relationships:

```
-- Create FactResellerSales PK
ALTER TABLE [dbo].[FactResellerSales] WITH CHECK ADD
    CONSTRAINT [PK_FactResellerSales_SalesOrderNumber_SalesOrderLineNumber] PRIMARY KEY CLUSTERED
    (
        [SalesOrderNumber], [SalesOrderLineNumber]
    ) ON [PRIMARY];
GO

-- Create foreign key relationships to the dimension tables
ALTER TABLE [dbo].[FactResellerSales] ADD
    CONSTRAINT [FK_FactResellerSales_DimEmployee] FOREIGN KEY([EmployeeKey])
        REFERENCES [dbo].[DimEmployee] ([EmployeeKey]),
    CONSTRAINT [FK_FactResellerSales_DimProduct] FOREIGN KEY([ProductKey])
        REFERENCES [dbo].[DimProduct] ([ProductKey]),
    CONSTRAINT [FK_FactResellerSales_DimReseller] FOREIGN KEY([ResellerKey])
        REFERENCES [dbo].[DimReseller] ([ResellerKey]);
GO
```

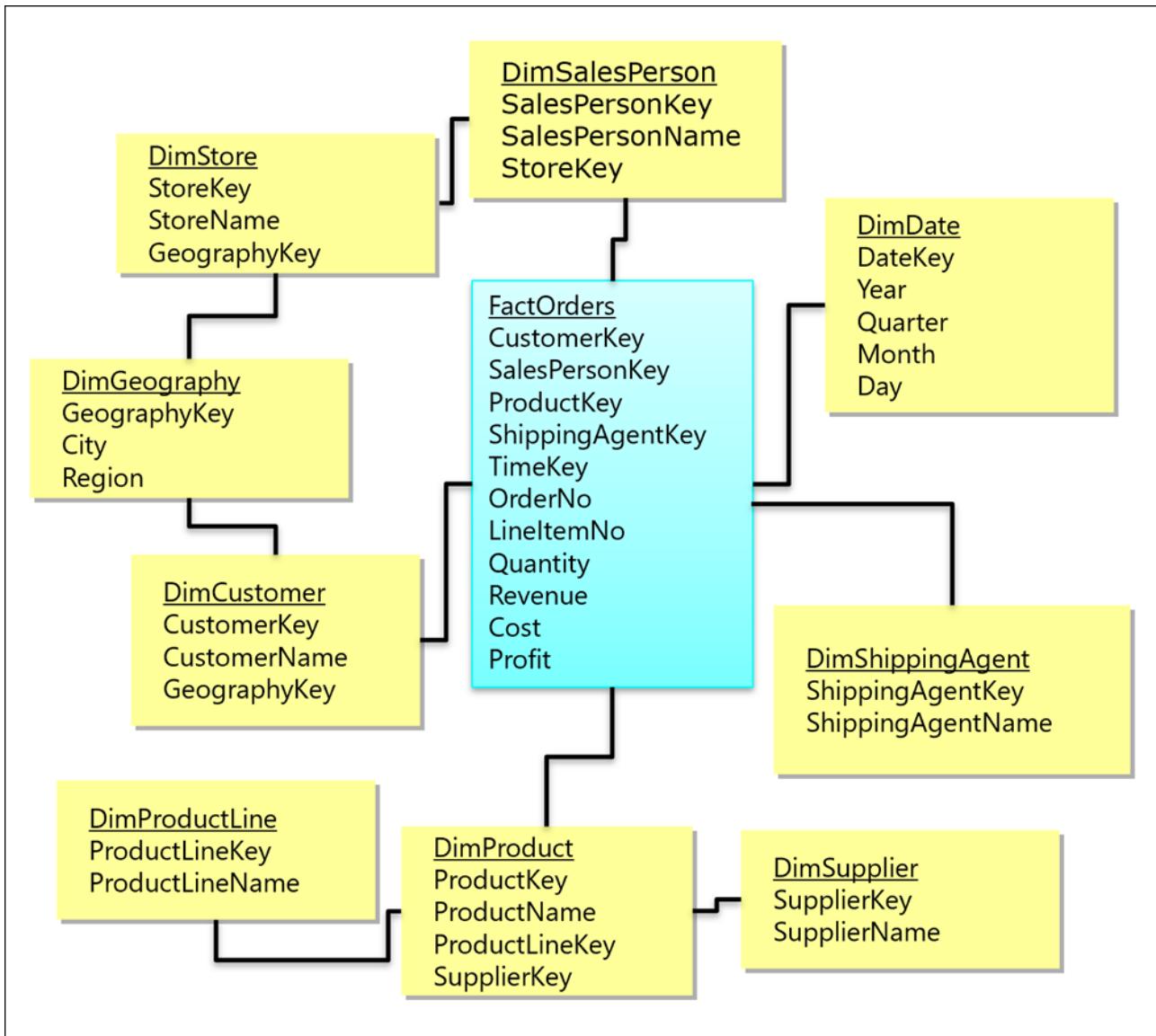
Our star schema now has relationships defined between the fact table and dimension tables. If you arrange the tables in a diagram, using a tool such as SQL Server Management studio, you can clearly see the relationships:



4.4 Exercise 2: Implementing a Snowflake Schema

A **snowflake schema** is a set of normalized tables for a single business entity. For example, Adventure Works classifies products by category and subcategory. Categories are assigned to subcategories, and products are in turn assigned to subcategories. In the Adventure Works relational data warehouse, the product dimension is normalized and stored in three related tables: **DimProductCategory**, **DimProductSubcategory**, and **DimProduct**.

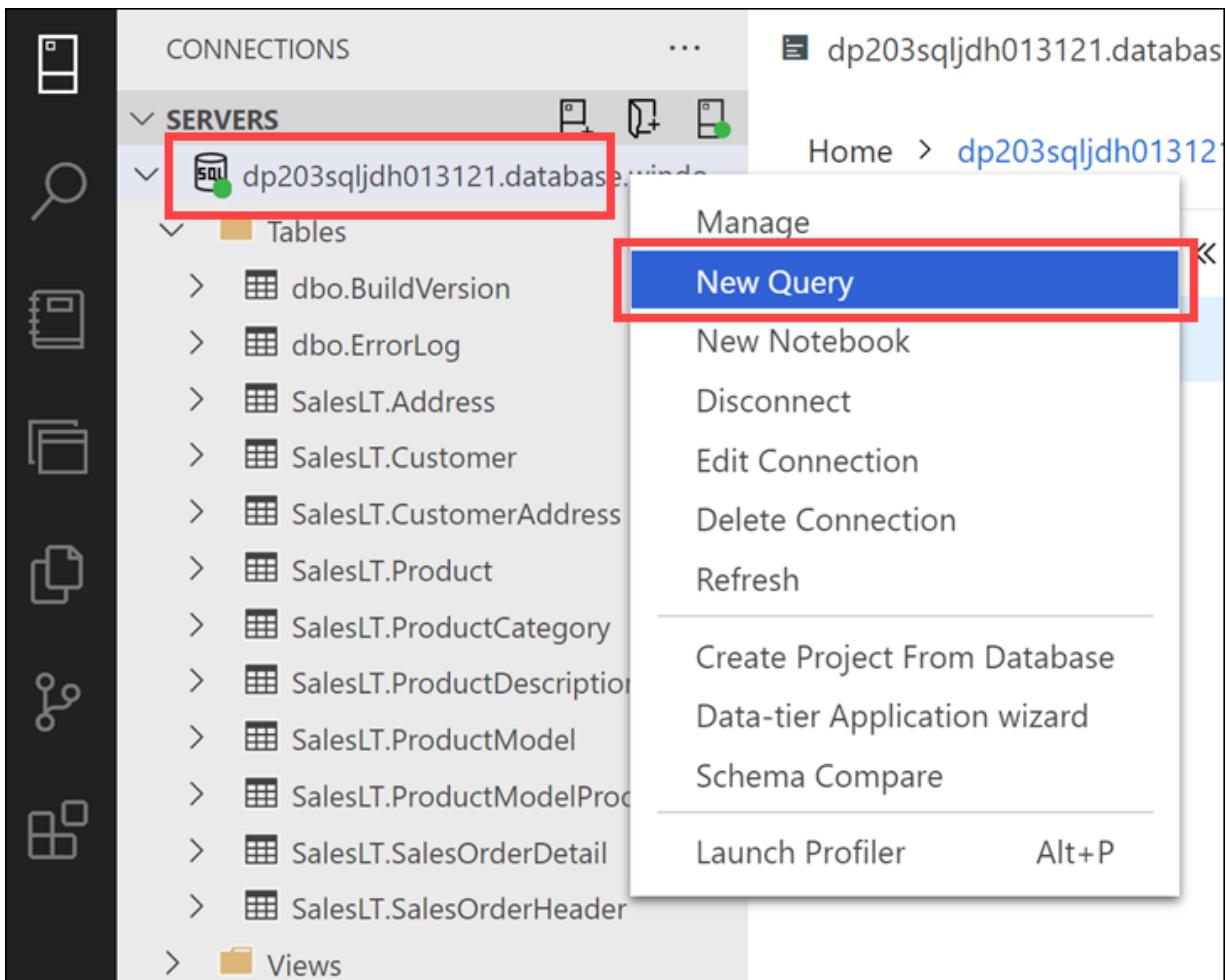
The snowflake schema is a variation of the star schema. You add normalized dimension tables to a star schema to create a snowflake pattern. In the following diagram, you see the yellow dimension tables surrounding the blue fact table. Notice that many of the dimension tables relate to one another in order to normalize the business entities:



4.4.1 Task 1: Create product snowflake schema in SQL database

In this task, you add two new dimension tables: **DimProductCategory** and **DimProductSubcategory**. You create a relationship between these two tables and the **DimProduct** table to create a normalized product dimension, known as a snowflake dimension. Doing so updates the star schema to include the normalized product dimension, transforming it into a snowflake schema.

1. Open Azure Data Explorer.
2. Select **Servers** in the left-hand menu, then right-click the SQL server you added at the beginning of the lab. Select **New Query**.



- Paste and execute the following into the query window to create the new dimension tables:

```

CREATE TABLE [dbo].[DimProductCategory] (
    [ProductCategoryKey] [int] IDENTITY(1,1) NOT NULL,
    [ProductCategoryAlternateKey] [int] NULL,
    [EnglishProductName] [nvarchar](50) NOT NULL,
    [SpanishProductName] [nvarchar](50) NOT NULL,
    [FrenchProductName] [nvarchar](50) NOT NULL
);
GO

CREATE TABLE [dbo].[DimProductSubcategory] (
    [ProductSubcategoryKey] [int] IDENTITY(1,1) NOT NULL,
    [ProductSubcategoryAlternateKey] [int] NULL,
    [EnglishProductSubcategoryName] [nvarchar](50) NOT NULL,
    [SpanishProductSubcategoryName] [nvarchar](50) NOT NULL,
    [FrenchProductSubcategoryName] [nvarchar](50) NOT NULL,
    [ProductCategoryKey] [int] NULL
);
GO

```

- Replace and execute the query with the following to create the DimProductCategory and DimProductSubcategory primary keys and constraints:

```

-- Create DimProductCategory PK
ALTER TABLE [dbo].[DimProductCategory] WITH CHECK ADD
CONSTRAINT [PK_DimProductCategory_ProductCategoryKey] PRIMARY KEY CLUSTERED
(
    [ProductCategoryKey]
) ON [PRIMARY];
GO

```

```

-- Create DimProductSubcategory PK
ALTER TABLE [dbo].[DimProductSubcategory] WITH CHECK ADD
CONSTRAINT [PK_DimProductSubcategory_ProductSubcategoryKey] PRIMARY KEY CLUSTERED
(
    [ProductSubcategoryKey]
) ON [PRIMARY];
GO

-- Create DimProductCategory unique constraint
ALTER TABLE [dbo].[DimProductCategory] ADD CONSTRAINT [AK_DimProductCategory_ProductCategoryAlternateKey] UNIQUE NONCLUSTERED
(
    [ProductCategoryAlternateKey] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
GO

-- Create DimProductSubcategory unique constraint
ALTER TABLE [dbo].[DimProductSubcategory] ADD CONSTRAINT [AK_DimProductSubcategory_ProductSubcategoryAlternateKey] UNIQUE NONCLUSTERED
(
    [ProductSubcategoryAlternateKey] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
GO

```

5. Replace **and execute** the query with the following to create foreign key relationships between DimProduct and DimProductSubcategory, and DimProductSubcategory and DimProductCategory:

```

-- Create foreign key relationship between DimProduct and DimProductSubcategory
ALTER TABLE [dbo].[DimProduct] ADD
CONSTRAINT [FK_DimProduct_DimProductSubcategory] FOREIGN KEY
(
    [ProductSubcategoryKey]
) REFERENCES [dbo].[DimProductSubcategory] ([ProductSubcategoryKey]);
GO

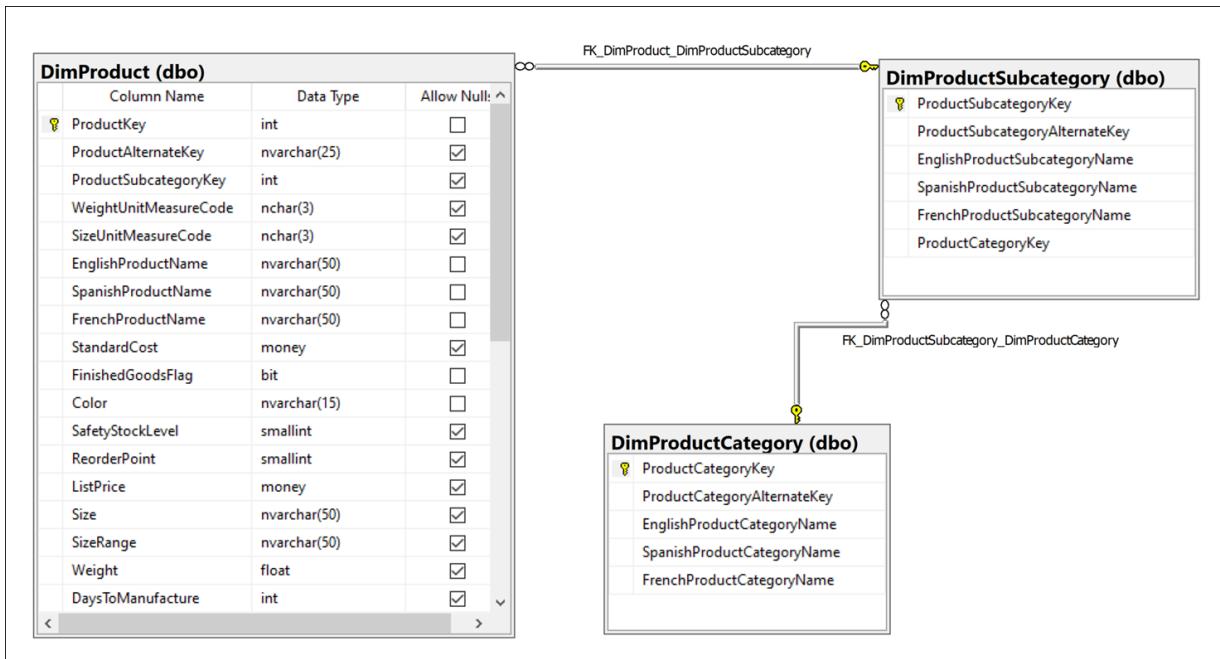
```

```

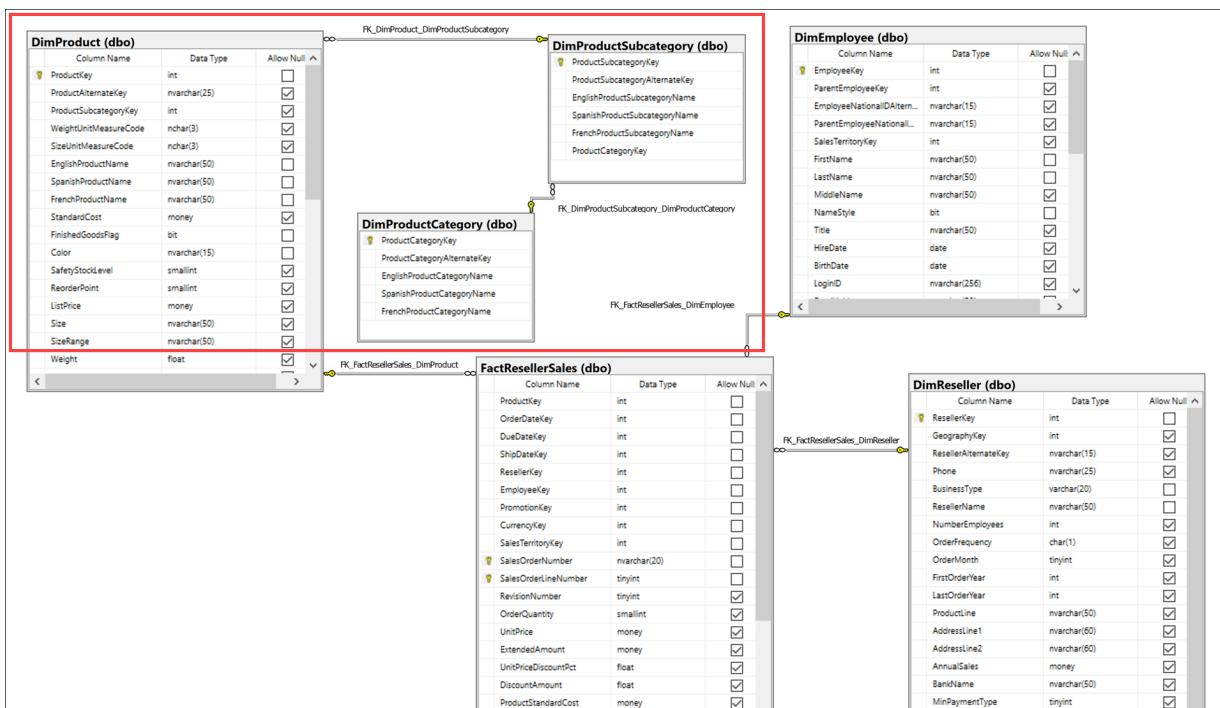
-- Create foreign key relationship between DimProductSubcategory and DimProductCategory
ALTER TABLE [dbo].[DimProductSubcategory] ADD
CONSTRAINT [FK_DimProductSubcategory_DimProductCategory] FOREIGN KEY
(
    [ProductCategoryKey]
) REFERENCES [dbo].[DimProductCategory] ([ProductCategoryKey]);
GO

```

You have created a snowflake dimension by normalizing the three product tables into a single business entity, or product dimension:



When we add the other tables into the diagram, we can see that the star schema is now transformed into a snowflake schema by normalizing the product tables. If you arrange the tables in a diagram, using a tool such as SQL Server Management studio, you can clearly see the relationships:



4.4.2 Task 2: Create reseller snowflake schema in SQL database

In this task, you add two new dimension tables: DimCustomer and DimGeography. You create a relationship between these two tables and the DimReseller table to create a normalized reseller dimension, or snowflake dimension.

- Paste and execute the following into the query window to create the new dimension tables:

```
CREATE TABLE [dbo].[DimCustomer]
[CustomerKey] [int] IDENTITY(1,1) NOT NULL,
[GeographyKey] [int] NULL,
[CustomerAlternateKey] [nvarchar](15) NOT NULL,
[Title] [nvarchar](8) NULL,
[FirstName] [nvarchar](50) NULL,
```

```

    [MiddleName] [nvarchar](50) NULL,
    [LastName] [nvarchar](50) NULL,
    [NameStyle] [bit] NULL,
    [BirthDate] [date] NULL,
    [MaritalStatus] [nchar](1) NULL,
    [Suffix] [nvarchar](10) NULL,
    [Gender] [nvarchar](1) NULL,
    [EmailAddress] [nvarchar](50) NULL,
    [YearlyIncome] [money] NULL,
    [TotalChildren] [tinyint] NULL,
    [NumberChildrenAtHome] [tinyint] NULL,
    [EnglishEducation] [nvarchar](40) NULL,
    [SpanishEducation] [nvarchar](40) NULL,
    [FrenchEducation] [nvarchar](40) NULL,
    [EnglishOccupation] [nvarchar](100) NULL,
    [SpanishOccupation] [nvarchar](100) NULL,
    [FrenchOccupation] [nvarchar](100) NULL,
    [HouseOwnerFlag] [nchar](1) NULL,
    [NumberCarsOwned] [tinyint] NULL,
    [AddressLine1] [nvarchar](120) NULL,
    [AddressLine2] [nvarchar](120) NULL,
    [Phone] [nvarchar](20) NULL,
    [DateFirstPurchase] [date] NULL,
    [CommuteDistance] [nvarchar](15) NULL
);
GO

```

```

CREATE TABLE [dbo].[DimGeography] (
    [GeographyKey] [int] IDENTITY(1,1) NOT NULL,
    [City] [nvarchar](30) NULL,
    [StateProvinceCode] [nvarchar](3) NULL,
    [StateProvinceName] [nvarchar](50) NULL,
    [CountryRegionCode] [nvarchar](3) NULL,
    [EnglishCountryRegionName] [nvarchar](50) NULL,
    [SpanishCountryRegionName] [nvarchar](50) NULL,
    [FrenchCountryRegionName] [nvarchar](50) NULL,
    [PostalCode] [nvarchar](15) NULL,
    [SalesTerritoryKey] [int] NULL,
    [IpAddressLocator] [nvarchar](15) NULL
);
GO

```

2. Replace **and execute** the query with the following to create the DimCustomer and DimGeography primary keys and a unique non-clustered index on the DimCustomer table:

```

-- Create DimCustomer PK
ALTER TABLE [dbo].[DimCustomer] WITH CHECK ADD
    CONSTRAINT [PK_DimCustomer_CustomerKey] PRIMARY KEY CLUSTERED
    (
        [CustomerKey]
    ) ON [PRIMARY];
GO

-- Create DimGeography PK
ALTER TABLE [dbo].[DimGeography] WITH CHECK ADD
    CONSTRAINT [PK_DimGeography_GeographyKey] PRIMARY KEY CLUSTERED
    (
        [GeographyKey]
    ) ON [PRIMARY];
GO

-- Create DimCustomer index

```

```
CREATE UNIQUE NONCLUSTERED INDEX [IX_DimCustomer_CustomerAlternateKey] ON [dbo].[DimCustomer] ([CustomerAlternateKey])
GO
```

3. Replace **and execute** the query with the following to create foreign key relationships between DimReseller and DimGeography, and DimGeography and DimCustomer:

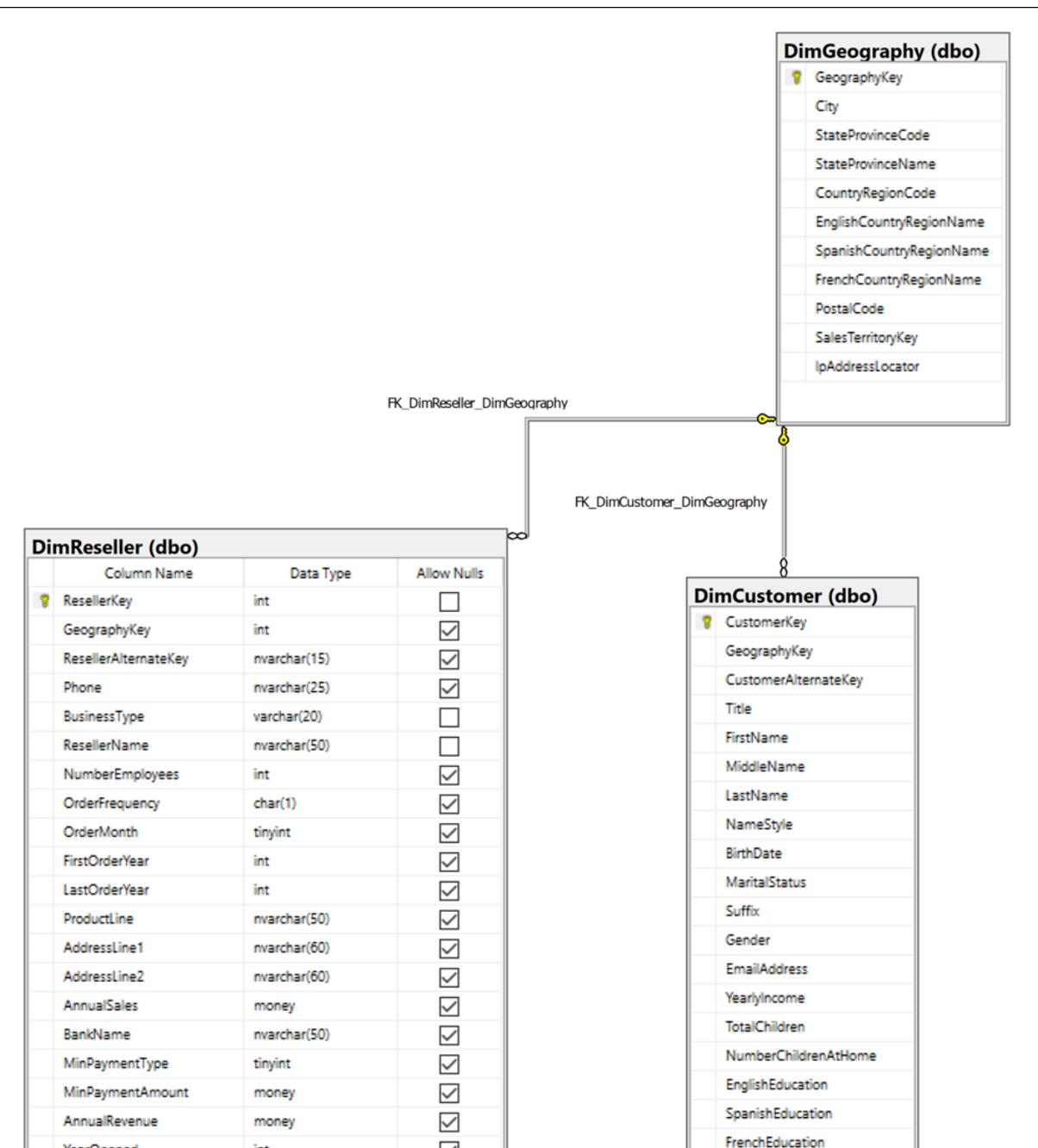
```
-- Create foreign key relationship between DimReseller and DimGeography
ALTER TABLE [dbo].[DimReseller] ADD
```

```
    CONSTRAINT [FK_DimReseller_DimGeography] FOREIGN KEY
    (
        [GeographyKey]
    ) REFERENCES [dbo].[DimGeography] ([GeographyKey]);
GO
```

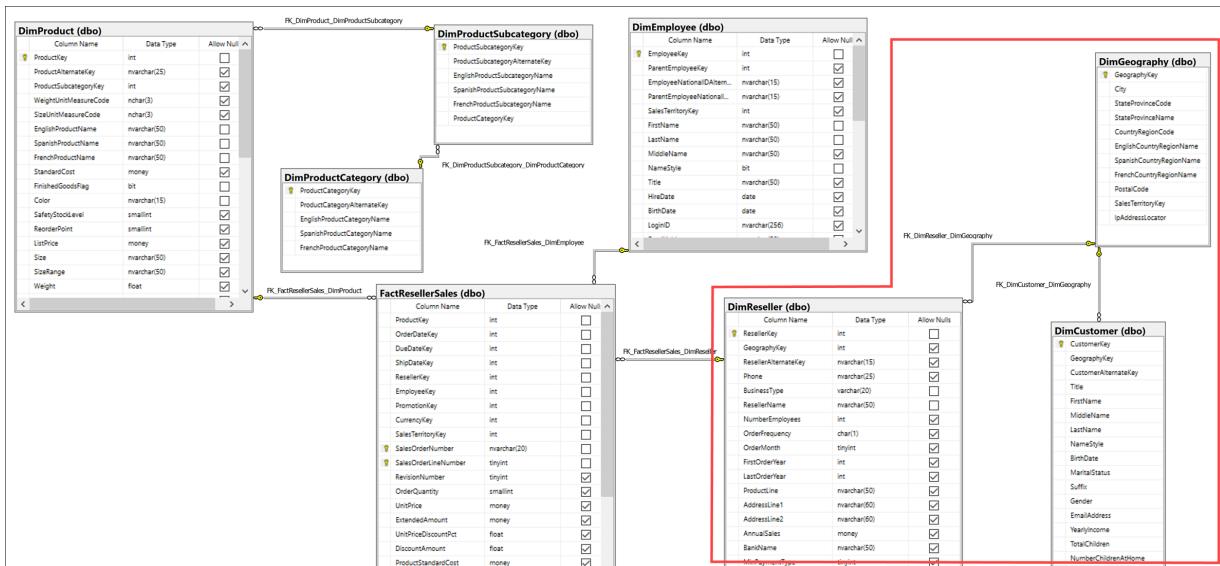
```
-- Create foreign key relationship between DimCustomer and DimGeography
ALTER TABLE [dbo].[DimCustomer] ADD
```

```
    CONSTRAINT [FK_DimCustomer_DimGeography] FOREIGN KEY
    (
        [GeographyKey]
    ) REFERENCES [dbo].[DimGeography] ([GeographyKey]);
GO
```

You now have a new snowflake dimension that normalizes reseller data with geography and customer dimensions.



Now let us look at how these new tables add another level of detail to our snowflake schema:



4.5 Exercise 3: Implementing a Time Dimension Table

A time dimension table is one of the most consistently used dimension tables. This type of table enables consistent granularity for temporal analysis and reporting and usually contains temporal hierarchies, such as Year > Quarter > Month > Day.

Time dimension tables can contain business-specific attributes that are useful references for reporting and filters, such as fiscal periods and public holidays.

This is the schema of the time dimension table that you will create:

Column	Data Type
DateKey	int
DateAltKey	datetime
CalendarYear	int
CalendarQuarter	int
MonthOfYear	int
MonthName	nvarchar(15)
DayOfMonth	int
DayOfWeek	int
DayName	nvarchar(15)
FiscalYear	int
FiscalQuarter	int

4.5.1 Task 1: Create time dimension table

In this task, you add the time dimension table and create foreign key relationships to the `FactRetailerSales` table.

1. Paste and execute the following into the query window to create the new time dimension table:

```
CREATE TABLE DimDate
(
    DateKey int NOT NULL,
    DateAltKey datetime NOT NULL,
    CalendarYear int NOT NULL,
    CalendarQuarter int NOT NULL,
    MonthOfYear int NOT NULL,
    [MonthName] nvarchar(15) NOT NULL,
    [DayOfMonth] int NOT NULL,
    [DayOfWeek] int NOT NULL,
    [DayName] nvarchar(15) NOT NULL,
    FiscalYear int NOT NULL,
    FiscalQuarter int NOT NULL)

```

GO

2. Replace **and execute** the query with the following to create the primary key and a unique non-clustered index on the DimDate table:

```
-- Create DimDate PK
```

```
ALTER TABLE [dbo].[DimDate] WITH CHECK ADD
CONSTRAINT [PK_DimDate_DateKey] PRIMARY KEY CLUSTERED
(
    [DateKey]
) ON [PRIMARY];
```

GO

```
-- Create unique non-clustered index
```

```
CREATE UNIQUE NONCLUSTERED INDEX [AK_DimDate_DateAltKey] ON [dbo].[DimDate]([DateAltKey]) ON [PRIMARY]
```

GO

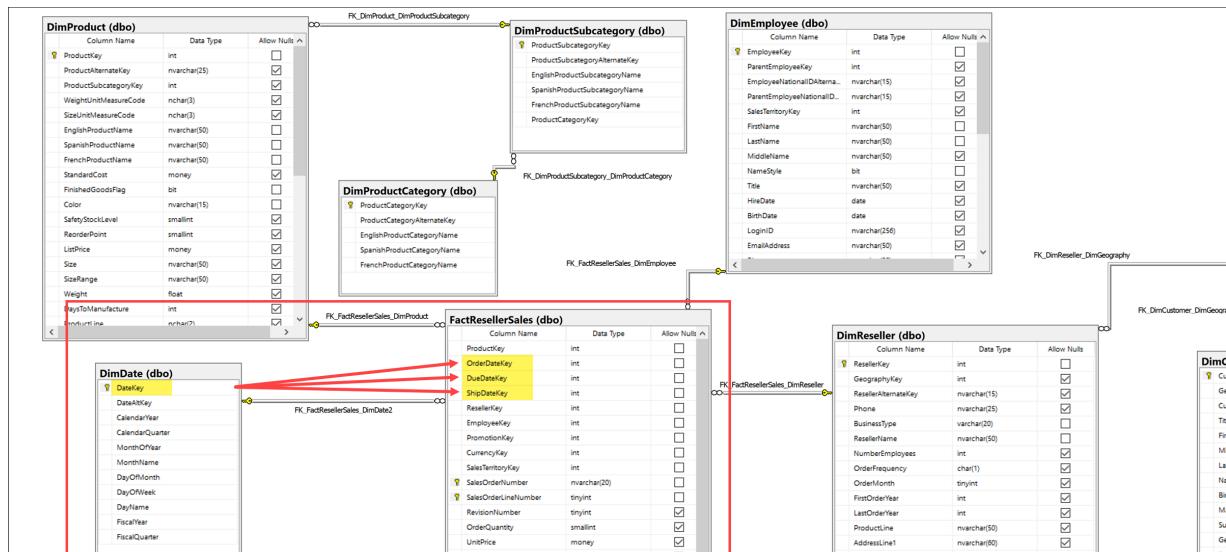
3. Replace **and execute** the query with the following to create foreign key relationships between FactRetailerSales and DimDate:

```
ALTER TABLE [dbo].[FactResellerSales] ADD
CONSTRAINT [FK_FactResellerSales_DimDate] FOREIGN KEY([OrderDateKey])
    REFERENCES [dbo].[DimDate] ([DateKey]),
CONSTRAINT [FK_FactResellerSales_DimDate1] FOREIGN KEY([DueDateKey])
    REFERENCES [dbo].[DimDate] ([DateKey]),
CONSTRAINT [FK_FactResellerSales_DimDate2] FOREIGN KEY([ShipDateKey])
    REFERENCES [dbo].[DimDate] ([DateKey]);
```

GO

Notice how the three fields refer to the primary key of the DimDate table.

Now our snowflake schema is updated to contain the time dimension table:



4.5.2 Task 2: Populate the time dimension table

You can populate time dimension tables in one of many ways, including T-SQL scripts using date/time functions, Microsoft Excel functions, importing from a flat file, or auto-generation by BI (business intelligence) tools. In this task, you populate the time dimension table using T-SQL, comparing generation methods along the way.

1. Paste **and execute** the following into the query window to create the new time dimension table:

```
DECLARE @StartDate datetime
DECLARE @EndDate datetime
SET @StartDate = '01/01/2005'
SET @EndDate = getdate()
DECLARE @LoopDate datetime
SET @LoopDate = @StartDate
```

```

WHILE @LoopDate <= @EndDate
BEGIN
INSERT INTO dbo.DimDate VALUES
(
    CAST(CONVERT(VARCHAR(8), @LoopDate, 112) AS int) , -- date key
    @LoopDate, -- date alt key
    Year(@LoopDate), -- calendar year
    datepart(qq, @LoopDate), -- calendar quarter
    Month(@LoopDate), -- month number of year
    datename(mm, @LoopDate), -- month name
    Day(@LoopDate), -- day number of month
    datepart(dw, @LoopDate), -- day number of week
    datename(dw, @LoopDate), -- day name of week
    CASE
        WHEN Month(@LoopDate) < 7 THEN Year(@LoopDate)
        ELSE Year(@Loopdate) + 1
    END, -- Fiscal year (assuming fiscal year runs from Jul to June)
    CASE
        WHEN Month(@LoopDate) IN (1, 2, 3) THEN 3
        WHEN Month(@LoopDate) IN (4, 5, 6) THEN 4
        WHEN Month(@LoopDate) IN (7, 8, 9) THEN 1
        WHEN Month(@LoopDate) IN (10, 11, 12) THEN 2
    END -- fiscal quarter
)
SET @LoopDate = DateAdd(dd, 1, @LoopDate)
END

```

In our environment, it took about **18 seconds** to insert the generated rows.

This query loops from a start date of January 1, 2005 until the current date, calculating and inserting values into the table for each day.

2. Replace **and execute** the query with the following to view the time dimension table data:

```
SELECT * FROM dbo.DimDate
```

You should see an output similar to:

	DateKey	DateAltKey	CalendarYear	CalendarQuarter	MonthOfYear	MonthName	DayOfMonth	DayOfWeek	DayName	FiscalYear	FiscalQuarter
1	20050101	2005-01-01 00:00:00.000	2005	1	1	January	1	7	Saturday	2005	3
2	20050102	2005-01-02 00:00:00.000	2005	1	1	January	2	1	Sunday	2005	3
3	20050103	2005-01-03 00:00:00.000	2005	1	1	January	3	2	Monday	2005	3
4	20050104	2005-01-04 00:00:00.000	2005	1	1	January	4	3	Tuesday	2005	3
5	20050105	2005-01-05 00:00:00.000	2005	1	1	January	5	4	Wednesday	2005	3
6	20050106	2005-01-06 00:00:00.000	2005	1	1	January	6	5	Thursday	2005	3
7	20050107	2005-01-07 00:00:00.000	2005	1	1	January	7	6	Friday	2005	3
8	20050108	2005-01-08 00:00:00.000	2005	1	1	January	8	7	Saturday	2005	3
9	20050109	2005-01-09 00:00:00.000	2005	1	1	January	9	1	Sunday	2005	3
10	20050110	2005-01-10 00:00:00.000	2005	1	1	January	10	2	Monday	2005	3
11	20050111	2005-01-11 00:00:00.000	2005	1	1	January	11	3	Tuesday	2005	3
12	20050112	2005-01-12 00:00:00.000	2005	1	1	January	12	4	Wednesday	2005	3
13	20050113	2005-01-13 00:00:00.000	2005	1	1	January	13	5	Thursday	2005	3
14	20050114	2005-01-14 00:00:00.000	2005	1	1	January	14	6	Friday	2005	3
15	20050115	2005-01-15 00:00:00.000	2005	1	1	January	15	7	Saturday	2005	3
16	20050116	2005-01-16 00:00:00.000	2005	1	1	January	16	1	Sunday	2005	3
17	20050117	2005-01-17 00:00:00.000	2005	1	1	January	17	2	Monday	2005	3
18	20050118	2005-01-18 00:00:00.000	2005	1	1	January	18	3	Tuesday	2005	3
19	20050119	2005-01-19 00:00:00.000	2005	1	1	January	19	4	Wednesday	2005	3
20	20050120	2005-01-20 00:00:00.000	2005	1	1	January	20	5	Thursday	2005	3
21	20050121	2005-01-21 00:00:00.000	2005	1	1	January	21	6	Friday	2005	3
22	20050122	2005-01-22 00:00:00.000	2005	1	1	January	22	7	Saturday	2005	3
23	20050123	2005-01-23 00:00:00.000	2005	1	1	January	23	1	Sunday	2005	3
24	20050124	2005-01-24 00:00:00.000	2005	1	1	January	24	2	Monday	2005	3
25	20050125	2005-01-25 00:00:00.000	2005	1	1	January	25	3	Tuesday	2005	3
26	20050126	2005-01-26 00:00:00.000	2005	1	1	January	26	4	Wednesday	2005	3
27	20050127	2005-01-27 00:00:00.000	2005	1	1	January	27	5	Thursday	2005	3
28	20050128	2005-01-28 00:00:00.000	2005	1	1	January	28	6	Friday	2005	3

3. Here is another way you can loop through dates to populate the table, this time setting both a start and end date. Replace **and execute** the query with the following to loop through dates within a given window (January 1, 1900 - December 31, 2050) and display the output:

```

DECLARE @BeginDate datetime
DECLARE @EndDate datetime

SET @BeginDate = '1/1/1900'
SET @EndDate = '12/31/2050'

CREATE TABLE #Dates ([date] datetime)

WHILE @BeginDate <= @EndDate
BEGIN
    INSERT #Dates
    VALUES
    (@BeginDate)

    SET @BeginDate = @BeginDate + 1
END
SELECT * FROM #Dates
DROP TABLE #Dates

```

In our environment, it took about 4 seconds to insert the generated rows.

This method works fine, but it has a lot to clean up, executes slowly, and has a lot of code when we factor adding in the other fields. Plus, it uses looping, which is not considered a best practice when inserting data using T-SQL.

4. Replace **and execute** the query with the following to improve the previous method with a **CTE** (common table expression) statement:

```

WITH mycte AS
(
    SELECT cast('1900-01-01' as datetime) DateValue
    UNION ALL

```

```

    SELECT DateValue + 1
    FROM mycte
    WHERE DateValue + 1 < '2050-12-31'
)

SELECT DateValue
FROM mycte
OPTION (MAXRECURSION 0)

```

In our environment, it took **less than one second** to execute the CTE query.

4.5.3 Task 3: Load data into other tables

In this task, you load the dimension and fact tables with data from a public data source.

- Paste **and execute** the following into the query window to create a master key encryption, database scoped credential, and external data source that accesses the public blob storage account that contains the source data:

```

IF NOT EXISTS (SELECT * FROM sys.symmetric_keys) BEGIN
    declare @password nvarchar(400) = CAST(newid() as VARCHAR(400));
    EXEC('CREATE MASTER KEY ENCRYPTION BY PASSWORD = ''' + @password + '''')
END

CREATE DATABASE SCOPED CREDENTIAL [dataengineering]
WITH IDENTITY='SHARED ACCESS SIGNATURE',
SECRET = 'sv=2019-10-10&st=2021-02-01T01%3A23%3A35Z&se=2030-02-02T01%3A23%3A00Z&sr=c&sp=rl&sig=Hui
GO

-- Create external data source secured using credential
CREATE EXTERNAL DATA SOURCE PublicDataSource WITH (
    TYPE = BLOB_STORAGE,
    LOCATION = 'https://solliancepublicdata.blob.core.windows.net/dataengineering',
    CREDENTIAL = dataengineering
);
GO

```

- Replace **and execute** the query with the following to insert data into the fact and dimension tables:

```

BULK INSERT[dbo]. [DimGeography] FROM 'dp-203/awdata/DimGeography.csv'
WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

BULK INSERT[dbo]. [DimCustomer] FROM 'dp-203/awdata/DimCustomer.csv'
WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

```

```

BULK INSERT[dbo]. [DimReseller] FROM 'dp-203/awdata/DimReseller.csv'
WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

BULK INSERT[dbo]. [DimEmployee] FROM 'dp-203/awdata/DimEmployee.csv'
WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

BULK INSERT[dbo]. [DimProductCategory] FROM 'dp-203/awdata/DimProductCategory.csv'
WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

BULK INSERT[dbo]. [DimProductSubcategory] FROM 'dp-203/awdata/DimProductSubcategory.csv'
WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

BULK INSERT[dbo]. [DimProduct] FROM 'dp-203/awdata/DimProduct.csv'
WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

BULK INSERT[dbo]. [FactResellerSales] FROM 'dp-203/awdata/FactResellerSales.csv'

```

```

WITH (
    DATA_SOURCE='PublicDataSource',
    CHECK_CONSTRAINTS,
    DATAFILETYPE='widechar',
    FIELDTERMINATOR='|',
    ROWTERMINATOR='\n',
    KEEPIDENTITY,
    TABLOCK
);
GO

```

4.5.4 Task 4: Query data

- Paste **and execute** the following query to retrieve reseller sales data from the snowflake schema at the reseller, product, and month granularity:

```

SELECT
    pc.[EnglishProductCategoryName]
    ,Coalesce(p.[ModelName], p.[EnglishProductName]) AS [Model]
    ,CASE
        WHEN e.[BaseRate] < 25 THEN 'Low'
        WHEN e.[BaseRate] > 40 THEN 'High'
        ELSE 'Moderate'
    END AS [EmployeeIncomeGroup]
    ,g.City AS ResellerCity
    ,g.StateProvinceName AS StateProvince
    ,r.[AnnualSales] AS ResellerAnnualSales
    ,d.[CalendarYear]
    ,d.[FiscalYear]
    ,d.[MonthOfYear] AS [Month]
    ,f.[SalesOrderNumber] AS [OrderNumber]
    ,f.SalesOrderLineNumber AS LineNumber
    ,f.OrderQuantity AS Quantity
    ,f.ExtendedAmount AS Amount
FROM
    [dbo].[FactResellerSales] f
INNER JOIN [dbo].[DimReseller] r
    ON f.ResellerKey = r.ResellerKey
INNER JOIN [dbo].[DimGeography] g
    ON r.GeographyKey = g.GeographyKey
INNER JOIN [dbo].[DimEmployee] e
    ON f.EmployeeKey = e.EmployeeKey
INNER JOIN [dbo].[DimDate] d
    ON f.[OrderDateKey] = d.[DateKey]
INNER JOIN [dbo].[DimProduct] p
    ON f.[ProductKey] = p.[ProductKey]
INNER JOIN [dbo].[DimProductSubcategory] psc
    ON p.[ProductSubcategoryKey] = psc.[ProductSubcategoryKey]
INNER JOIN [dbo].[DimProductCategory] pc
    ON psc.[ProductCategoryKey] = pc.[ProductCategoryKey]
ORDER BY Amount DESC

```

You should see an output similar to the following:

Results		Messages														
		EnglishProductName	Model	EmployeeIncomeGroup	ResellerCity	StateProvince	ResellerAnnualSales	CalendarYear	FiscalYear	Month	OrderNumber	LineNumber	Quantity	Amount		
1	Bikes	Touring-1000	Low	Sand City	California	3000000.0000	2013	2013	3	S055282	39	26	30992.9100			
2	Bikes	Mountain-100	Low	Minneapolis	Minnesota	3000000.0000	2011	2011	1	S043884	17	14	27607.9188			
3	Bikes	Touring-1000	Low	Sand City	California	3000000.0000	2012	2013	12	S051131	12	21	27536.0085			
4	Bikes	Road-350-W	Low	Union City	California	3000000.0000	2013	2013	2	S053460	53	30	25514.8500			
5	Bikes	Mountain-100	Low	Orlando	Florida	3000000.0000	2011	2011	1	S043875	12	13	25447.4246			
6	Bikes	Touring-1000	Low	Paris	Seine (Paris)	3000000.0000	2013	2013	4	S057954	26	19	24913.5315			
7	Bikes	Mountain-100	Low	Orlando	Florida	3000000.0000	2011	2011	1	S043875	10	12	23663.9304			
8	Bikes	Mountain-100	Low	La Mesa	California	3000000.0000	2011	2011	5	S044795	18	12	23663.9304			
9	Bikes	Touring-1000	Low	London	England	3000000.0000	2013	2013	1	S051823	10	18	23602.2930			
10	Bikes	Touring-1000	Low	Sand City	California	3000000.0000	2013	2013	3	S055282	16	18	23602.2930			
11	Bikes	Mountain-100	Low	Orlando	Florida	3000000.0000	2011	2011	5	S044518	8	12	23489.9304			
12	Bikes	Mountain-100	Low	Calgary	Alberta	3000000.0000	2011	2011	5	S044534	10	12	23489.9304			
13	Bikes	Mountain-100	Low	Gulfport	Mississippi	3000000.0000	2011	2012	10	S046090	13	12	23489.9304			
14	Bikes	Touring-1000	Low	Sand City	California	3000000.0000	2013	2013	3	S055282	36	17	22291.0545			
15	Bikes	Mountain-200	Low	Loveland	Colorado	3000000.0000	2013	2014	7	S063291	27	17	21691.9065			
16	Bikes	Mountain-100	Low	Orlando	Florida	3000000.0000	2011	2011	5	S044518	3	11	21532.4362			
17	Bikes	Mountain-100	Low	Park City	Utah	3000000.0000	2011	2011	3	S044100	9	11	21532.4362			
18	Bikes	Mountain-200	Low	London	England	3000000.0000	2013	2013	2	S053573	41	17	21458.1565			
19	Bikes	Road-350-W	Low	Union City	California	3000000.0000	2013	2014	11	S071783	7	25	21262.3750			
20	Bikes	Touring-1000	Low	Offenbach	Saarland	3000000.0000	2013	2013	3	S055254	17	16	20979.8160			
21	Bikes	Touring-1000	Low	Sand City	California	3000000.0000	2012	2013	12	S051131	54	16	20979.8160			
22	Bikes	Road-350-W	Low	Toronto	Ontario	1500000.0000	2012	2013	12	S051132	19	22	20581.9790			
23	Bikes	Mountain-100	Low	Orlando	Florida	3000000.0000	2011	2011	5	S044518	13	10	20399.9400			
24	Bikes	Mountain-100	Low	Moline	Illinois	1000000.0000	2011	2011	5	S044528	2	10	20399.9400			
25	Bikes	Mountain-100	Low	Moline	Illinois	1000000.0000	2011	2012	10	S046066	5	10	20399.9400			
26	Bikes	Mountain-100	Low	Moline	Illinois	1000000.0000	2011	2012	10	S046066	6	10	20399.9400			
27	Bikes	Mountain-100	Low	Orlando	Florida	3000000.0000	2011	2012	8	S045300	12	10	20249.9400			
28	Bikes	Mountain-100	Low	Minneapolis	Minnesota	3000000.0000	2011	2012	8	S045308	6	10	20249.9400			

2. Replace and execute the query with the following to limit the results to October sales between the 2012 and 2013 fiscal years:

```

SELECT
    pc.[EnglishProductName]
    ,Coalesce(p.[ModelName], p.[EnglishProductName]) AS [Model]
    ,CASE
        WHEN e.[BaseRate] < 25 THEN 'Low'
        WHEN e.[BaseRate] > 40 THEN 'High'
        ELSE 'Moderate'
    END AS [EmployeeIncomeGroup]
    ,g.City AS ResellerCity
    ,g.StateProvinceName AS StateProvince
    ,r.[AnnualSales] AS ResellerAnnualSales
    ,d.[CalendarYear]
    ,d.[FiscalYear]
    ,d.[MonthOfYear] AS [Month]
    ,f.[SalesOrderNumber] AS [OrderNumber]
    ,f.SalesOrderLineNumber AS LineNumber
    ,f.OrderQuantity AS Quantity
    ,f.ExtendedAmount AS Amount
FROM
    [dbo].[FactResellerSales] f
INNER JOIN [dbo].[DimReseller] r
    ON f.ResellerKey = r.ResellerKey
INNER JOIN [dbo].[DimGeography] g
    ON r.GeographyKey = g.GeographyKey
INNER JOIN [dbo].[DimEmployee] e
    ON f.EmployeeKey = e.EmployeeKey
INNER JOIN [dbo].[DimDate] d
    ON f.[OrderDateKey] = d.[DateKey]
INNER JOIN [dbo].[DimProduct] p
    ON f.[ProductKey] = p.[ProductKey]
INNER JOIN [dbo].[DimProductSubcategory] psc
    ON p.[ProductSubcategoryKey] = psc.[ProductSubcategoryKey]
INNER JOIN [dbo].[DimProductCategory] pc
    ON psc.[ProductCategoryKey] = pc.[ProductCategoryKey]
WHERE d.[MonthOfYear] = 10 AND d.[FiscalYear] IN (2012, 2013)
ORDER BY d.[FiscalYear]
```

You should see an output similar to the following:

Results		Messages											
	EnglishProductCategoryName	Model	EmployeeIncomeGroup	ResellerCity	StateProvince	ResellerAnnualSales	CalendarYear	FiscalYear	Month	OrderNumber	LineNumber	Quantity	Amount
1	Bikes	Mountain-100	Low	Waterloo	Ontario	3000000.0000	2011	2012	10	S046023	1	3	6074.9820
2	Bikes	Mountain-100	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	1	1	2039.9940
3	Bikes	Mountain-100	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	2	2	4049.9880
4	Clothing	Mountain Bike Socks	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	3	6	34.2000
5	Bikes	Mountain-100	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	4	1	2039.9940
6	Bikes	Mountain-100	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	5	1	2024.9940
7	Bikes	Mountain-100	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	6	1	2039.9940
8	Clothing	Mountain Bike Socks	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	7	2	11.4000
9	Bikes	Mountain-100	Low	Winnipeg	Manitoba	1500000.0000	2011	2012	10	S046024	8	1	2039.9940
10	Clothing	Long-Sleeve Logo Jersey	Low	Wasougal	Washington	1500000.0000	2011	2012	10	S046025	1	2	57.6808
11	Bikes	Road-650	Low	Wasougal	Washington	1500000.0000	2011	2012	10	S046025	2	1	419.4589
12	Bikes	Road-450	Low	Wasougal	Washington	1500000.0000	2011	2012	10	S046025	3	2	1749.5880
13	Bikes	Road-650	Low	Wasougal	Washington	1500000.0000	2011	2012	10	S046025	4	1	419.4589
14	Bikes	Mountain-100	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	1	4	8159.9760
15	Components	HL Mountain Frame	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	2	3	2429.2800
16	Components	HL Mountain Frame	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	3	2	1429.4086
17	Bikes	Mountain-100	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	4	3	6074.9820
18	Bikes	Mountain-100	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	5	2	4079.9880
19	Components	HL Mountain Frame	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	6	1	818.7000
20	Components	HL Mountain Frame	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	7	1	714.7043
21	Bikes	Mountain-100	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	8	2	4049.9880
22	Clothing	Mountain Bike Socks	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	9	3	17.1000
23	Clothing	Long-Sleeve Logo Jersey	Low	Zeeland	Michigan	3000000.0000	2011	2012	10	S046026	10	1	28.8404
24	Bikes	Road-650	Low	Warwick	Rhode Isla..	1500000.0000	2011	2012	10	S046027	1	2	838.9178
25	Bikes	Road-450	Low	Warwick	Rhode Isla..	1500000.0000	2011	2012	10	S046027	2	1	874.7940
26	Bikes	Road-650	Low	Warwick	Rhode Isla..	1500000.0000	2011	2012	10	S046027	3	1	419.4589
27	Bikes	Road-450	Low	Warwick	Rhode Isla..	1500000.0000	2011	2012	10	S046027	4	1	874.7940
28	Bikes	Road-650	Low	Warwick	Rhode Isla..	1500000.0000	2011	2012	10	S046027	5	1	419.4589
29	Clothing	Long-Sleeve Logo Jersey	Low	Warwick	Rhode Isla..	1500000.0000	2011	2012	10	S046027	6	1	28.8404
30	Components	ML Road Frame	Low	Winston-S...	North Caro...	800000.0000	2011	2012	10	S046028	1	1	356.8980

Notice how using the **time dimension table** makes filtering by specific date parts and logical dates (such as fiscal year) easier and more performant than calculating date functions on the fly.

4.6 Exercise 4: Implementing a Star Schema in Synapse Analytics

For larger data sets you may implement your data warehouse in Azure Synapse instead of SQL Server. Star schema models are still a best practice for modeling data in Synapse dedicated SQL pools. You may notice some differences with creating tables in Synapse Analytics vs. SQL database, but the same data modeling principles apply.

When you create a star schema or snowflake schema in Synapse, it requires some changes to your table creation scripts. In Synapse, you do not have foreign keys and unique value constraints like you do in SQL Server. Since these rules are not enforced at the database layer, the jobs used to load data are more responsible to maintain data integrity. You still have the option to use clustered indexes, but for most dimension tables in Synapse you will benefit from using a clustered columnstore index (CCI).

Since Synapse Analytics is a **massively parallel processing** (MPP) system, you must consider how data is distributed in your table design, as opposed to symmetric multiprocessing (SMP) systems, such as OLTP databases like Azure SQL Database. The table category often determines which option to choose for distributing the table.

Table category	Recommended distribution option
Fact	Use hash-distribution with clustered columnstore index. Performance improves when two hash tables are joined.
Dimension	Use replicated for smaller tables. If tables are too large to store on each Compute node, use hash-distribution.
Staging	Use round-robin for the staging table. The load with CTAS is fast. Once the data is in the staging table,

In the case of the dimension tables in this exercise, the amount of data stored per table falls well within the criteria for using a replicated distribution.

4.6.1 Task 1: Create star schema in Synapse dedicated SQL

In this task, you create a star schema in Azure Synapse dedicated pool. The first step is to create the base dimension and fact tables.

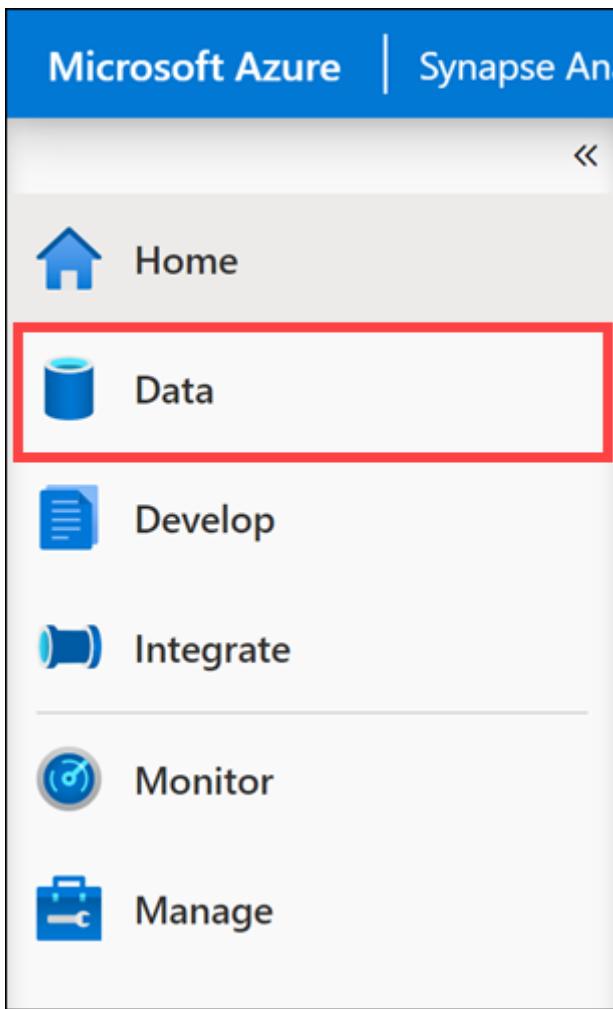
- Sign in to the Azure portal (<https://portal.azure.com>).
- Open the resource group for this lab, then select the **Synapse workspace**.

Name ↑↓	Type ↑↓
<input type="checkbox"/> asagadatalakejdh013121	Storage account
<input type="checkbox"/> asagakeyvaultjdh013121	Key vault
<input checked="" type="checkbox"/> asagaworkspacejdh013121	Synapse workspace
<input type="checkbox"/> dp203sqljdh013121	SQL server
<input type="checkbox"/> SourceDB (dp203sqljdh013121/SourceDB)	SQL database
<input type="checkbox"/> SQLPool01 (asagaworkspacejdh013121/SQLPool01)	Dedicated SQL pool

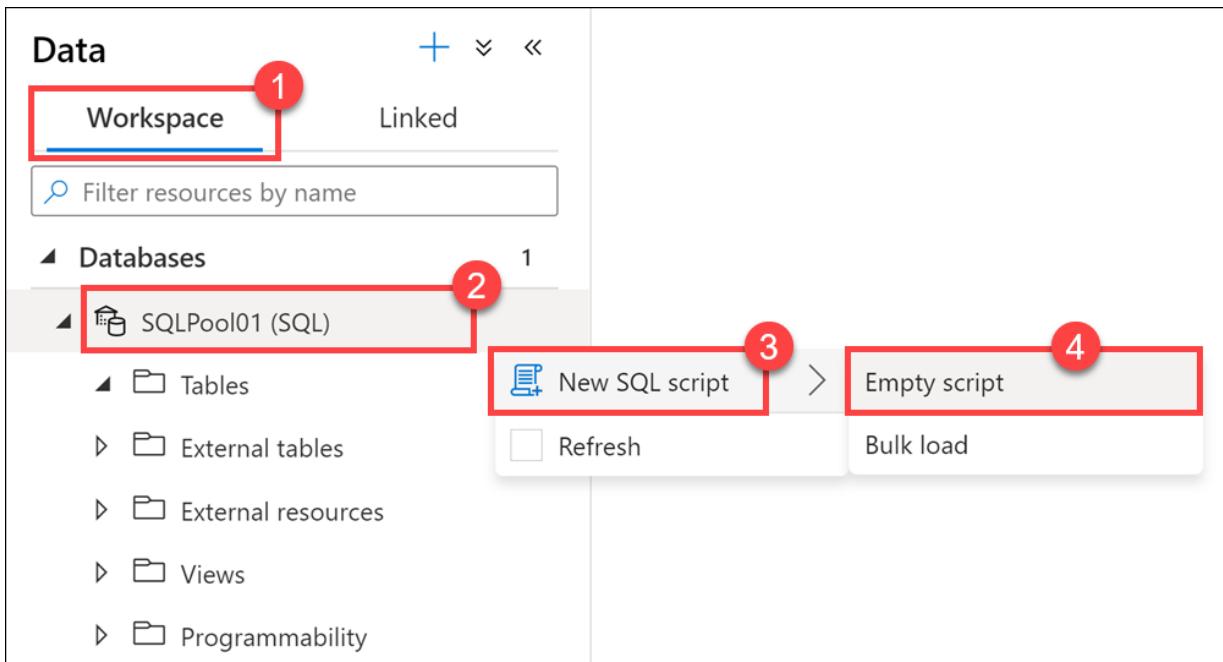
3. In your Synapse workspace Overview blade, select the **Open** link within **Open Synapse Studio**.

The screenshot shows the Azure Synapse workspace overview page. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, SQL Active Directory admin, Properties, Locks, Analytics pools (SQL pools, Apache Spark pools), and Security. The main content area displays workspace details: Resource group (dp203-labs), Status (Succeeded), Location (South Central US), Subscription (change), Subscription ID, Managed virtual network (No), Managed Identity object (No), Workspace web URL (<https://web.azure-synapse.net?workspace=%2fsu...>), and Tags (Click here to add tags). Below these details is a 'Getting started' section with two cards: 'Open Synapse Studio' (highlighted with a red box) and 'Read documentation'. The 'Open Synapse Studio' card contains the text: 'Start building your fully-integrated analytics solution and unlock new insights.' and a 'Open' button.

4. In Synapse Studio, navigate to the **Data** hub.



5. Select the **Workspace** tab (1), expand Databases, then right-click on **SQLPool01** (2). Select **New SQL script** (3), then select **Empty script** (4).



6. Paste the following script into the empty script window, then select **Run** or hit F5 to execute the query. You may notice some changes have been made to the original SQL star schema create script. A few notable changes are:
 - Distribution setting has been added to each table

- Clustered columnstore index is used for most tables.
- HASH function is used for Fact table distribution since it will be a larger table that should be distributed across nodes.
- A few fields are using varbinary data types that cannot be included in a clustered columnstore index in Azure Synapse. As a simple solution, a clustered index was used instead.

```

CREATE TABLE dbo.[DimCustomer] (
    [CustomerID] [int] NOT NULL,
    [Title] [nvarchar](8) NULL,
    [FirstName] [nvarchar](50) NOT NULL,
    [MiddleName] [nvarchar](50) NULL,
    [LastName] [nvarchar](50) NOT NULL,
    [Suffix] [nvarchar](10) NULL,
    [CompanyName] [nvarchar](128) NULL,
    [SalesPerson] [nvarchar](256) NULL,
    [EmailAddress] [nvarchar](50) NULL,
    [Phone] [nvarchar](25) NULL,
    [InsertedDate] [datetime] NOT NULL,
    [ModifiedDate] [datetime] NOT NULL,
    [HashKey] [char](66)
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);
GO

CREATE TABLE [dbo].[FactResellerSales] (
    [ProductKey] [int] NOT NULL,
    [OrderDateKey] [int] NOT NULL,
    [DueDateKey] [int] NOT NULL,
    [ShipDateKey] [int] NOT NULL,
    [ResellerKey] [int] NOT NULL,
    [EmployeeKey] [int] NOT NULL,
    [PromotionKey] [int] NOT NULL,
    [CurrencyKey] [int] NOT NULL,
    [SalesTerritoryKey] [int] NOT NULL,
    [SalesOrderNumber] [nvarchar](20) NOT NULL,
    [SalesOrderLineNumber] [tinyint] NOT NULL,
    [RevisionNumber] [tinyint] NULL,
    [OrderQuantity] [smallint] NULL,
    [UnitPrice] [money] NULL,
    [ExtendedAmount] [money] NULL,
    [UnitPriceDiscountPct] [float] NULL,
    [DiscountAmount] [float] NULL,
    [ProductStandardCost] [money] NULL,
    [TotalProductCost] [money] NULL,
    [SalesAmount] [money] NULL,
    [TaxAmt] [money] NULL,
    [Freight] [money] NULL,
    [CarrierTrackingNumber] [nvarchar](25) NULL,
    [CustomerPONumber] [nvarchar](25) NULL,
    [OrderDate] [datetime] NULL,
    [DueDate] [datetime] NULL,
    [ShipDate] [datetime] NULL
)
WITH
(
    DISTRIBUTION = HASH([SalesOrderNumber]),
    CLUSTERED COLUMNSTORE INDEX
)

```

```

);

GO

CREATE TABLE [dbo].[DimDate]
(
    [DateKey] [int] NOT NULL,
    [DateAltKey] [datetime] NOT NULL,
    [CalendarYear] [int] NOT NULL,
    [CalendarQuarter] [int] NOT NULL,
    [MonthOfYear] [int] NOT NULL,
    [MonthName] [nvarchar](15) NOT NULL,
    [DayOfMonth] [int] NOT NULL,
    [DayOfWeek] [int] NOT NULL,
    [DayName] [nvarchar](15) NOT NULL,
    [FiscalYear] [int] NOT NULL,
    [FiscalQuarter] [int] NOT NULL
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);
GO

CREATE TABLE [dbo].[DimReseller](
    [ResellerKey] [int] NOT NULL,
    [GeographyKey] [int] NULL,
    [ResellerAlternateKey] [nvarchar](15) NULL,
    [Phone] [nvarchar](25) NULL,
    [BusinessType] [varchar](20) NOT NULL,
    [ResellerName] [nvarchar](50) NOT NULL,
    [NumberEmployees] [int] NULL,
    [OrderFrequency] [char](1) NULL,
    [OrderMonth] [tinyint] NULL,
    [FirstOrderYear] [int] NULL,
    [LastOrderYear] [int] NULL,
    [ProductLine] [nvarchar](50) NULL,
    [AddressLine1] [nvarchar](60) NULL,
    [AddressLine2] [nvarchar](60) NULL,
    [AnnualSales] [money] NULL,
    [BankName] [nvarchar](50) NULL,
    [MinPaymentType] [tinyint] NULL,
    [MinPaymentAmount] [money] NULL,
    [AnnualRevenue] [money] NULL,
    [YearOpened] [int] NULL
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);
GO

CREATE TABLE [dbo].[DimEmployee](
    [EmployeeKey] [int] NOT NULL,
    [ParentEmployeeKey] [int] NULL,
    [EmployeeNationalIDAlternateKey] [nvarchar](15) NULL,
    [ParentEmployeeNationalIDAlternateKey] [nvarchar](15) NULL,
    [SalesTerritoryKey] [int] NULL,
    [FirstName] [nvarchar](50) NOT NULL,
    [LastName] [nvarchar](50) NOT NULL,

```

```

[MiddleName] [nvarchar](50) NULL,
[NameStyle] [bit] NOT NULL,
[Title] [nvarchar](50) NULL,
[HireDate] [date] NULL,
[BirthDate] [date] NULL,
[LoginID] [nvarchar](256) NULL,
[EmailAddress] [nvarchar](50) NULL,
[Phone] [nvarchar](25) NULL,
[MaritalStatus] [nchar](1) NULL,
[EmergencyContactName] [nvarchar](50) NULL,
[EmergencyContactPhone] [nvarchar](25) NULL,
[SalariedFlag] [bit] NULL,
[Gender] [nchar](1) NULL,
[PayFrequency] [tinyint] NULL,
[BaseRate] [money] NULL,
[VacationHours] [smallint] NULL,
[SickLeaveHours] [smallint] NULL,
[CurrentFlag] [bit] NOT NULL,
[SalesPersonFlag] [bit] NOT NULL,
[DepartmentName] [nvarchar](50) NULL,
[StartDate] [date] NULL,
[EndDate] [date] NULL,
[Status] [nvarchar](50) NULL,
[EmployeePhoto] [varbinary](max) NULL
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED INDEX (EmployeeKey)
);
GO

CREATE TABLE [dbo].[DimProduct](
    [ProductKey] [int] NOT NULL,
    [ProductAlternateKey] [nvarchar](25) NULL,
    [ProductSubcategoryKey] [int] NULL,
    [WeightUnitMeasureCode] [nchar](3) NULL,
    [SizeUnitMeasureCode] [nchar](3) NULL,
    [EnglishProductName] [nvarchar](50) NOT NULL,
    [SpanishProductName] [nvarchar](50) NULL,
    [FrenchProductName] [nvarchar](50) NULL,
    [StandardCost] [money] NULL,
    [FinishedGoodsFlag] [bit] NOT NULL,
    [Color] [nvarchar](15) NOT NULL,
    [SafetyStockLevel] [smallint] NULL,
    [ReorderPoint] [smallint] NULL,
    [ListPrice] [money] NULL,
    [Size] [nvarchar](50) NULL,
    [SizeRange] [nvarchar](50) NULL,
    [Weight] [float] NULL,
    [DaysToManufacture] [int] NULL,
    [ProductLine] [nchar](2) NULL,
    [DealerPrice] [money] NULL,
    [Class] [nchar](2) NULL,
    [Style] [nchar](2) NULL,
    [ModelName] [nvarchar](50) NULL,
    [LargePhoto] [varbinary](max) NULL,
    [EnglishDescription] [nvarchar](400) NULL,
    [FrenchDescription] [nvarchar](400) NULL,
    [ChineseDescription] [nvarchar](400) NULL,
    [ArabicDescription] [nvarchar](400) NULL,
)

```

```

[HebrewDescription] [nvarchar](400) NULL,
[ThaiDescription] [nvarchar](400) NULL,
[GermanDescription] [nvarchar](400) NULL,
[JapaneseDescription] [nvarchar](400) NULL,
[TurkishDescription] [nvarchar](400) NULL,
[StartDate] [datetime] NULL,
[EndDate] [datetime] NULL,
[Status] [nvarchar](7) NULL
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED INDEX (ProductKey)
);
GO

CREATE TABLE [dbo].[DimGeography] (
    [GeographyKey] [int] NOT NULL,
    [City] [nvarchar](30) NULL,
    [StateProvinceCode] [nvarchar](3) NULL,
    [StateProvinceName] [nvarchar](50) NULL,
    [CountryRegionCode] [nvarchar](3) NULL,
    [EnglishCountryRegionName] [nvarchar](50) NULL,
    [SpanishCountryRegionName] [nvarchar](50) NULL,
    [FrenchCountryRegionName] [nvarchar](50) NULL,
    [PostalCode] [nvarchar](15) NULL,
    [SalesTerritoryKey] [int] NULL,
    [IpAddressLocator] [nvarchar](15) NULL
)
WITH
(
    DISTRIBUTION = REPLICATE,
    CLUSTERED COLUMNSTORE INDEX
);
GO

```

```

1 CREATE TABLE dbo.[DimCustomer](
2     [CustomerID] [int] NOT NULL,
3     [Title] [nvarchar](8) NULL,
4     [FirstName] [nvarchar](50) NOT NULL,
5     [MiddleName] [nvarchar](50) NULL,
6     [LastName] [nvarchar](50) NOT NULL,
7     [Suffix] [nvarchar](10) NULL,
8     [CompanyName] [nvarchar](128) NULL,
9     [SalesPerson] [nvarchar](256) NULL,
10    [EmailAddress] [nvarchar](50) NULL,
11    [Phone] [nvarchar](25) NULL,
12    [InsertedDate] [datetime] NOT NULL,
13    [ModifiedDate] [datetime] NOT NULL,
14    [HashKey] [char](66)
15 )
16 WITH
17 (
18     DISTRIBUTION = REPLICATE,
19     CLUSTERED COLUMNSTORE INDEX
20 )
21

```

You will find Run in the top left corner of the script window.

4.6.2 Task 2: Load data into Synapse tables

In this task, you load the Synapse dimension and fact tables with data from a public data source. There are two ways to load this data from Azure Storage files using T-SQL: the COPY command or selecting from

external tables using Polybase. For this task you will use COPY since it is a simple and flexible syntax for loading delimited data from Azure Storage. If the source were a private storage account you would include a CREDENTIAL option to authorize the COPY command to read the data, but for this example that is not required.

1. Paste **and execute** the query with the following to insert data into the fact and dimension tables:

```
COPY INTO [dbo].[DimProduct]
FROM 'https://solliancepublicdata.blob.core.windows.net/dataengineering/dp-203/awdata/DimProduct.csv'
WITH (
    FILE_TYPE='CSV',
    FIELDTERMINATOR='|',
    FIELDQUOTE='',
    ROWTERMINATOR='\n',
    ENCODING = 'UTF16'
);
GO

COPY INTO [dbo].[DimReseller]
FROM 'https://solliancepublicdata.blob.core.windows.net/dataengineering/dp-203/awdata/DimReseller.csv'
WITH (
    FILE_TYPE='CSV',
    FIELDTERMINATOR='|',
    FIELDQUOTE='',
    ROWTERMINATOR='\n',
    ENCODING = 'UTF16'
);
GO

COPY INTO [dbo].[DimEmployee]
FROM 'https://solliancepublicdata.blob.core.windows.net/dataengineering/dp-203/awdata/DimEmployee.csv'
WITH (
    FILE_TYPE='CSV',
    FIELDTERMINATOR='|',
    FIELDQUOTE='',
    ROWTERMINATOR='\n',
    ENCODING = 'UTF16'
);
GO

COPY INTO [dbo].[DimGeography]
FROM 'https://solliancepublicdata.blob.core.windows.net/dataengineering/dp-203/awdata/DimGeography.csv'
WITH (
    FILE_TYPE='CSV',
    FIELDTERMINATOR='|',
    FIELDQUOTE='',
    ROWTERMINATOR='\n',
    ENCODING = 'UTF16'
);
GO

COPY INTO [dbo].[FactResellerSales]
FROM 'https://solliancepublicdata.blob.core.windows.net/dataengineering/dp-203/awdata/FactResellerSales.csv'
WITH (
    FILE_TYPE='CSV',
    FIELDTERMINATOR='|',
    FIELDQUOTE='',
    ROWTERMINATOR='\n',
    ENCODING = 'UTF16'
);
GO
```

2. To populate the time dimension table in Azure Synapse, it is fastest to load the data from a delimited file since the looping method used to create the time data runs slowly. To populate this important time dimension, paste **and execute** the following in the query window:

```
COPY INTO [dbo].[DimDate]
FROM 'https://solliancepublicdata.blob.core.windows.net/dataengineering/dp-203/awdata/DimDate.csv'
WITH (
    FILE_TYPE='CSV',
    FIELDTERMINATOR='|',
    FIELDQUOTE='',
    ROWTERMINATOR='0x0a',
    ENCODING = 'UTF16'
);
GO
```

4.6.3 Task 3: Query data from Synapse

1. Paste **and execute** the following query to retrieve reseller sales data from the Synapse star schema at the reseller location, product, and month granularity:

```
SELECT
    Coalesce(p.[ModelName], p.[EnglishProductName]) AS [Model]
    ,g.City AS ResellerCity
    ,g.StateProvinceName AS StateProvince
    ,d.[CalendarYear]
    ,d.[FiscalYear]
    ,d.[MonthOfYear] AS [Month]
    ,sum(f.OrderQuantity) AS Quantity
    ,sum(f.ExtendedAmount) AS Amount
    ,approx_count_distinct(f.SalesOrderNumber) AS UniqueOrders
FROM
    [dbo].[FactResellerSales] f
INNER JOIN [dbo].[DimReseller] r
    ON f.ResellerKey = r.ResellerKey
INNER JOIN [dbo].[DimGeography] g
    ON r.GeographyKey = g.GeographyKey
INNER JOIN [dbo].[DimDate] d
    ON f.[OrderDateKey] = d.[DateKey]
INNER JOIN [dbo].[DimProduct] p
    ON f.[ProductKey] = p.[ProductKey]
GROUP BY
    Coalesce(p.[ModelName], p.[EnglishProductName])
    ,g.City
    ,g.StateProvinceName
    ,d.[CalendarYear]
    ,d.[FiscalYear]
    ,d.[MonthOfYear]
ORDER BY Amount DESC
```

You should see an output similar to the following:

Results Messages

View Table Chart Export results ^

Search

Model	ResellerCity	StateProvince	CalendarYear	FiscalYear	Month	Quantity	Amount	UniqueOrders
Mountain-100	Seattle	Washington	2011	2011	3	272	552838.3680	2
Mountain-100	Toronto	Ontario	2011	2011	3	268	544738.3920	5
Mountain-100	Toronto	Ontario	2011	2012	8	256	520498.4640	4
Touring-1000	Sand City	California	2013	2013	3	372	483489.3960	1
Mountain-100	Orlando	Florida	2011	2011	5	236	473248.6024	1
Mountain-100	Orlando	Florida	2011	2011	1	224	448444.6760	1
Mountain-100	Toronto	Ontario	2011	2011	5	216	438898.7040	4
Mountain-100	Moline	Illinois	2011	2011	5	208	422698.7520	1
Mountain-100	Minneapolis	Minnesota	2011	2011	1	208	419250.7632	1
Mountain-100	Moline	Illinois	2011	2012	10	188	382498.8720	1
Mountain-100	Minneapolis	Minnesota	2011	2011	5	184	373858.8960	1
Mountain-100	La Mesa	California	2011	2011	5	180	362974.9296	1
Mountain-200	Toronto	Ontario	2012	2012	2	292	358314.1604	3
Mountain-100	Gulfport	Mississippi	2011	2011	5	176	358018.9440	1
Mountain-100	Minneapolis	Minnesota	2011	2012	10	176	357898.9440	1
Mountain-100	City Of Commerce	California	2011	2011	5	176	357178.9440	1
Mountain-200	Toronto	Ontario	2012	2012	5	292	357055.5944	3
Touring-1000	Sand City	California	2012	2013	12	292	350458.2900	1
Touring-1000	London	England	2013	2013	1	300	349981.4760	2

2. Replace **and execute** the query with the following to limit the results to October sales between the 2012 and 2013 fiscal years:

```

SELECT
    Coalesce(p.[ModelName], p.[EnglishProductName]) AS [Model]
    ,g.City AS ResellerCity
    ,g.StateProvinceName AS StateProvince
    ,d.[CalendarYear]
    ,d.[FiscalYear]
    ,d.[MonthOfYear] AS [Month]
    ,sum(f.OrderQuantity) AS Quantity
    ,sum(f.ExtendedAmount) AS Amount
    ,approx_count_distinct(f.SalesOrderNumber) AS UniqueOrders
FROM
    [dbo].[FactResellerSales] f
INNER JOIN [dbo].[DimReseller] r
    ON f.ResellerKey = r.ResellerKey
INNER JOIN [dbo].[DimGeography] g
    ON r.GeographyKey = g.GeographyKey
INNER JOIN [dbo].[DimDate] d
    ON f.[OrderDateKey] = d.[DateKey]
INNER JOIN [dbo].[DimProduct] p
    ON f.[ProductKey] = p.[ProductKey]
WHERE d.[MonthOfYear] = 10 AND d.[FiscalYear] IN (2012, 2013)
GROUP BY
    Coalesce(p.[ModelName], p.[EnglishProductName])
    ,g.City
    ,g.StateProvinceName
    ,d.[CalendarYear]
    ,d.[FiscalYear]
    ,d.[MonthOfYear]
ORDER BY d.[FiscalYear]

```

You should see an output similar to the following:

Results Messages

View Table Chart Export results ^

Search

Model	ResellerCity	StateProvince	CalendarYear	FiscalYear	Month	Quantity	Amount	UniqueOrders
Cycling Cap	Indianapolis	Indiana	2011	2012	10	8	414920	1
Cycling Cap	Winston-Salem	North Carolina	2011	2012	10	8	414920	1
HL Road Frame	Indianapolis	Indiana	2011	2012	10	16	12129.2144	1
LL Road Frame	Indianapolis	Indiana	2011	2012	10	36	6536.0568	2
LL Road Frame	Winston-Salem	North Carolina	2011	2012	10	24	4371.6576	1
Long-Sleeve Logo Jersey	Indianapolis	Indiana	2011	2012	10	32	922.8928	1
Long-Sleeve Logo Jersey	Winston-Salem	North Carolina	2011	2012	10	24	692.1696	1
ML Road Frame	Winston-Salem	North Carolina	2011	2012	10	4	1427.5920	1
Road-150	Winston-Salem	North Carolina	2011	2012	10	20	42939.2400	1
Road-450	Indianapolis	Indiana	2011	2012	10	12	10497.5280	1
Road-450	Winston-Salem	North Carolina	2011	2012	10	40	34991.7600	1
Road-650	Indianapolis	Indiana	2011	2012	10	48	20134.0272	1
Road-650	Winston-Salem	North Carolina	2011	2012	10	48	20134.0272	1
Sport-100	Indianapolis	Indiana	2011	2012	10	48	968.9520	2
Sport-100	Winston-Salem	North Carolina	2011	2012	10	24	484.4760	1
Cycling Cap	Toronto	Ontario	2011	2012	10	12	62.2380	1
LL Road Frame	Toronto	Ontario	2011	2012	10	44	7986.1328	3
Long-Sleeve Logo Jersey	Toronto	Ontario	2011	2012	10	12	346.0848	2
ML Road Frame	Toronto	Ontario	2011	2012	10	20	7137.9600	2

Notice how using the **time dimension table** makes filtering by specific date parts and logical dates (such as fiscal year) easier and more performant than calculating date functions on the fly.

4.7 Exercise 5: Updating slowly changing dimensions with mapping data flows

A **slowly changing dimension** (SCD) is one that appropriately manages change of dimension members over time. It applies when business entity values change over time, and in an ad hoc manner. A good example of a slowly changing dimension is a customer dimension, specifically its contact detail columns like email address and phone number. In contrast, some dimensions are considered to be rapidly changing when a dimension attribute changes often, like a stock's market price. The common design approach in these instances is to store rapidly changing attribute values in a fact table measure.

Star schema design theory refers to two common SCD types: Type 1 and Type 2. A dimension-type table could be Type 1 or Type 2, or support both types simultaneously for different columns.

Type 1 SCD

A **Type 1 SCD** always reflects the latest values, and when changes in source data are detected, the dimension table data is overwritten. This design approach is common for columns that store supplementary values, like the email address or phone number of a customer. When a customer email address or phone number changes, the dimension table updates the customer row with the new values. It's as if the customer always had this contact information.

Type 2 SCD

A **Type 2 SCD** supports versioning of dimension members. If the source system doesn't store versions, then it's usually the data warehouse load process that detects changes, and appropriately manages the change in a dimension table. In this case, the dimension table must use a surrogate key to provide a unique reference to a version of the dimension member. It also includes columns that define the date range validity of the version (for example, `StartDate` and `EndDate`) and possibly a flag column (for example, `IsCurrent`) to easily filter by current dimension members.

For example, Adventure Works assigns salespeople to a sales region. When a salesperson relocates, a new version of the salesperson must be created to ensure that historical facts remain associated with the former region. To support accurate historic analysis of sales by salesperson, the dimension table must store versions of salespeople and their associated region(s). The table should also include start and end date values to define the time validity. Current versions may define an empty end date (or 12/31/9999), which indicates that the row is the current version. The table must also define a surrogate key because the business key (in this instance, employee ID) won't be unique.

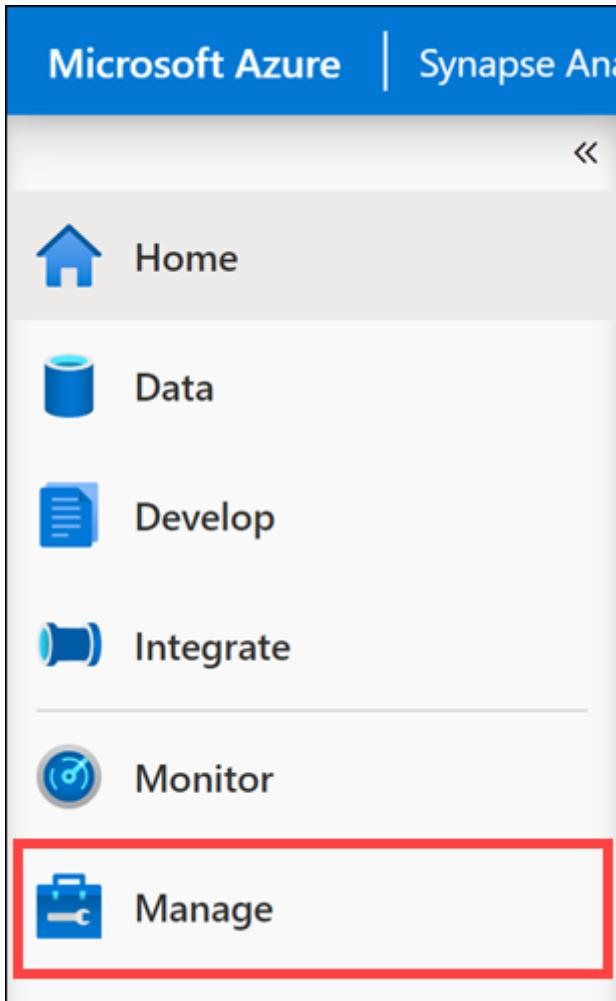
It's important to understand that when the source data doesn't store versions, you must use an intermediate system (like a data warehouse) to detect and store changes. The table load process must preserve existing data and detect changes. When a change is detected, the table load process must expire the current version. It records these changes by updating the `EndDate` value and inserting a new version with the `StartDate` value commencing from the previous `EndDate` value. Also, related facts must use a time-based lookup to retrieve the dimension key value relevant to the fact date.

In this exercise, you create a Type 1 SCD with Azure SQL Database as the source, and your Synapse dedicated SQL pool as the destination.

4.7.1 Task 1: Create the Azure SQL Database linked service

Linked services in Synapse Analytics enables you to manage connections to external resources. In this task, you create a linked service for the Azure SQL Database used as the data source for the DimCustomer dimension table.

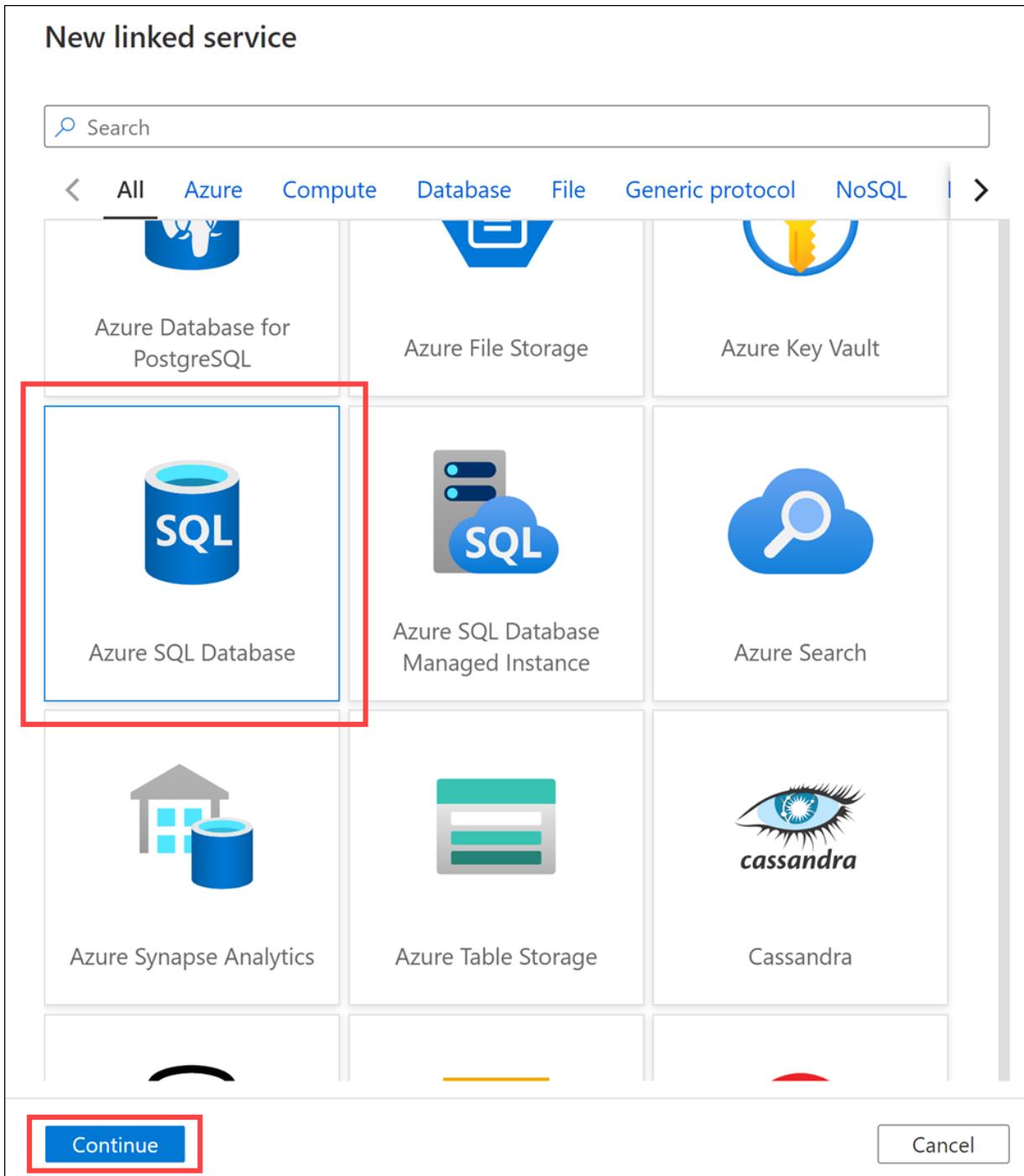
1. In Synapse Studio, navigate to the **Manage** hub.



2. Select **Linked services** on the left, then select **+ New**.

A screenshot of the 'Linked services' page in Synapse Studio. On the left, there's a sidebar with 'Analytics pools' (SQL pools, Apache Spark pools) and 'External connections' (Linked services, Azure Purview (Preview)). The 'Linked services' option is highlighted with a red box. The main area shows the 'Linked services' heading, a descriptive text about linked services, a '+ New' button (which is also highlighted with a red box), a 'Filter by name' input field, and a message indicating 1-2 items are showing.

3. Select **Azure SQL Database**, then select **Continue**.



4. Complete the new linked service form as follows:

- **Name:** Enter `AzureSqlDatabaseSource`
- **Account selection method:** Select **From Azure subscription**
- **Azure subscription:** Select the Azure subscription used for this lab
- **Server name:** Select the Azure SQL Server named `dp203sqlSUFFIX` (where SUFFIX is your unique suffix)
- **Database name:** Select `SourceDB`
- **Authentication type:** Select **SQL authentication**
- **Username:** Enter `sqladmin`
- **Password:** Enter the password you provided during the environment setup, or that was given to you if this is a hosted lab environment (also used at the beginning of this lab)

New linked service (Azure SQL Database)

i Choose a name for your linked service. This name cannot be updated later.

Name *

AzureSqlDataSource

Description

Connect via integration runtime * ⓘ

AutoResolveIntegrationRuntime



[Connection string](#) [Azure Key Vault](#)

Account selection method ⓘ

From Azure subscription Enter manually

Azure subscription



Server name *

dp203sqljdh013121



Database name *

SourceDB



Authentication type *

SQL authentication



User name *

sqladmin

[Password](#) [Azure Key Vault](#)

Password *

.....



Additional connection properties

+ New

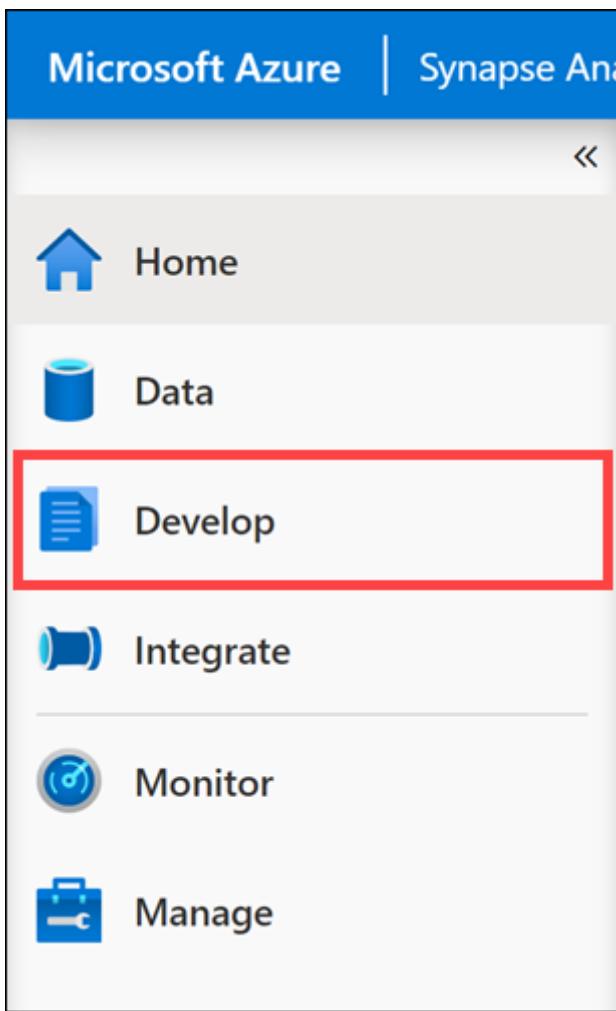
5. Select **Create**.

4.7.2 Task 2: Create a mapping data flow

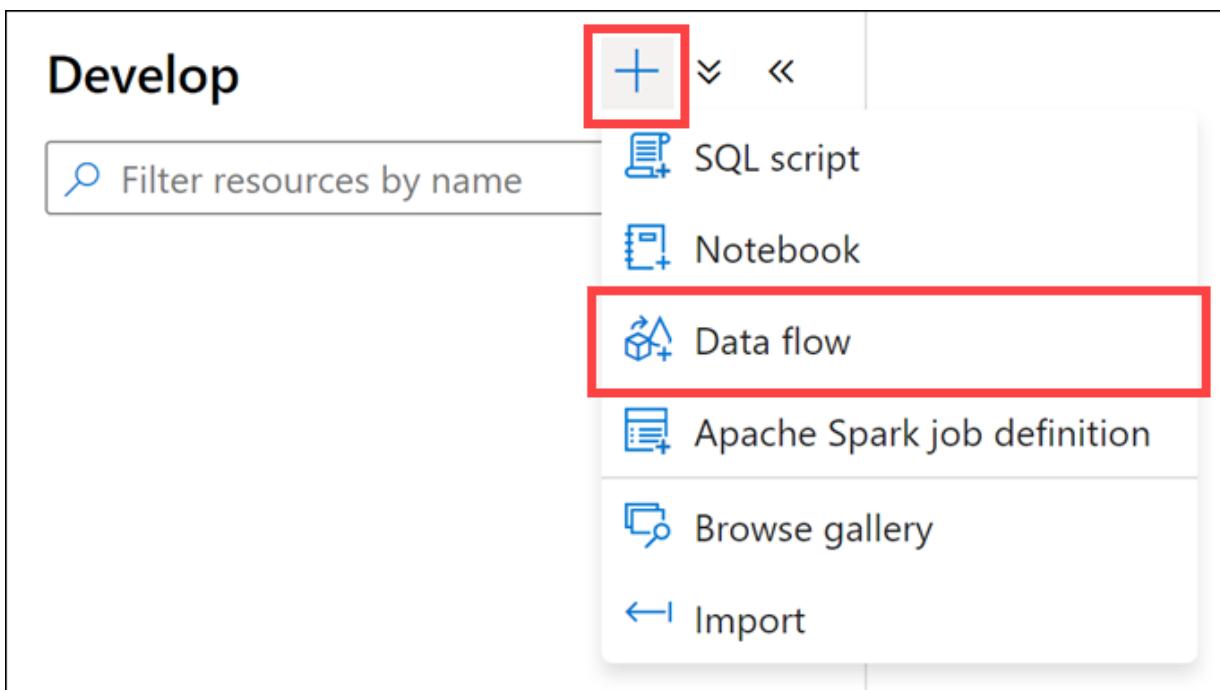
Mapping Data flows are pipeline activities that provide a visual way of specifying how to transform data, through a code-free experience. This feature offers data cleansing, transformation, aggregation, conversion, joins, data copy operations, etc.

In this task, you create a mapping data flow to create a Type 1 SCD.

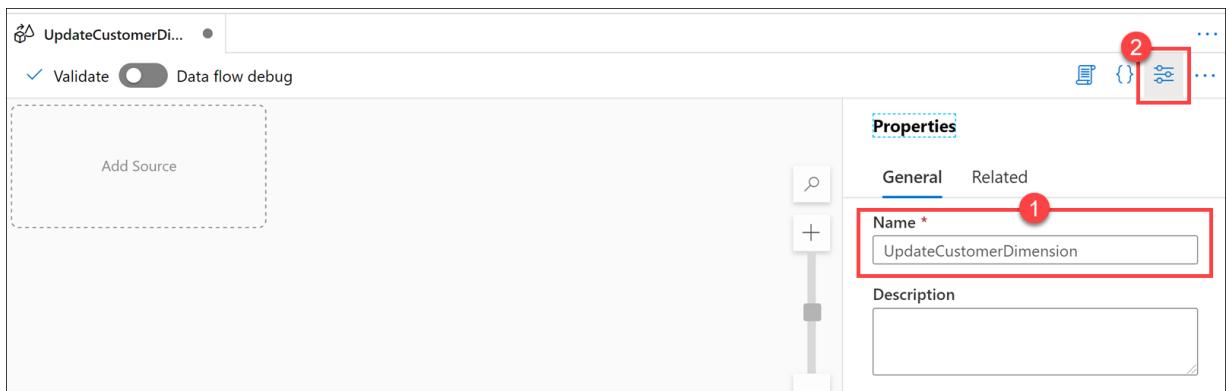
1. Navigate to the **Develop** hub.



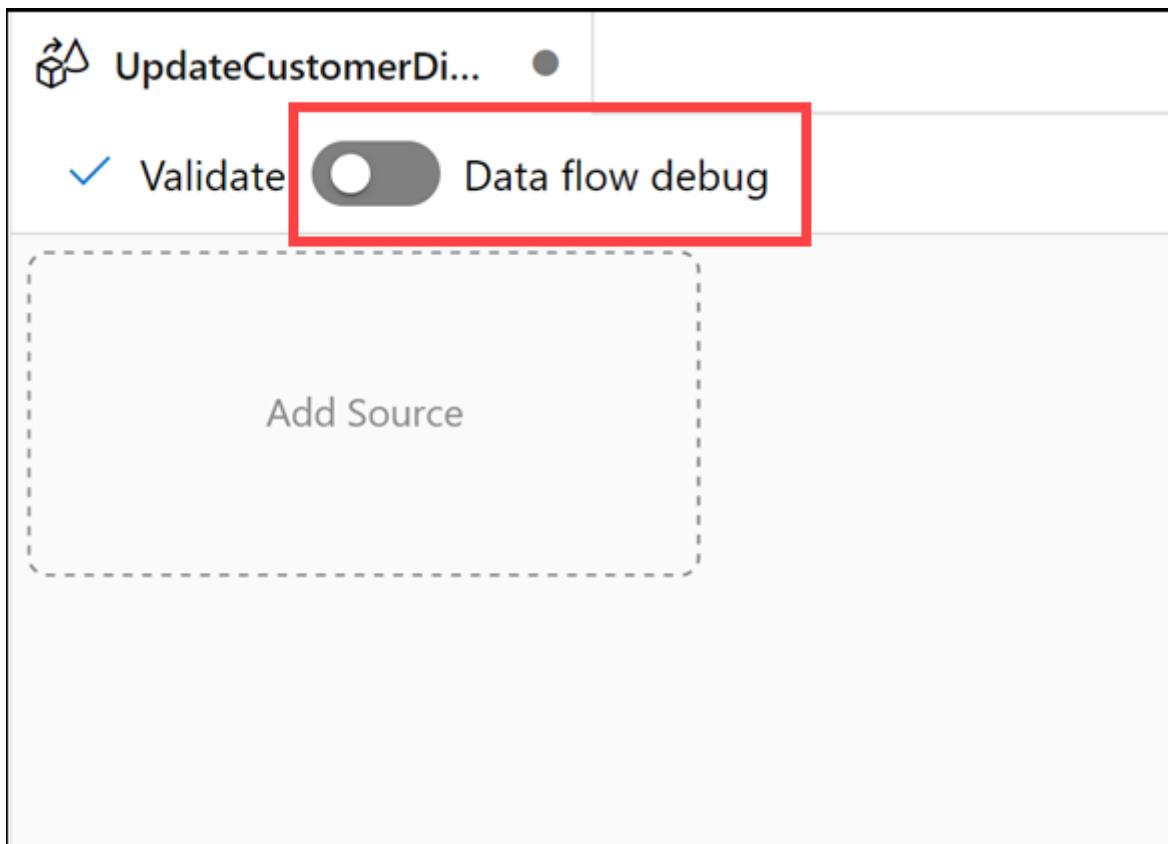
2. Select +, then select **Data flow**.



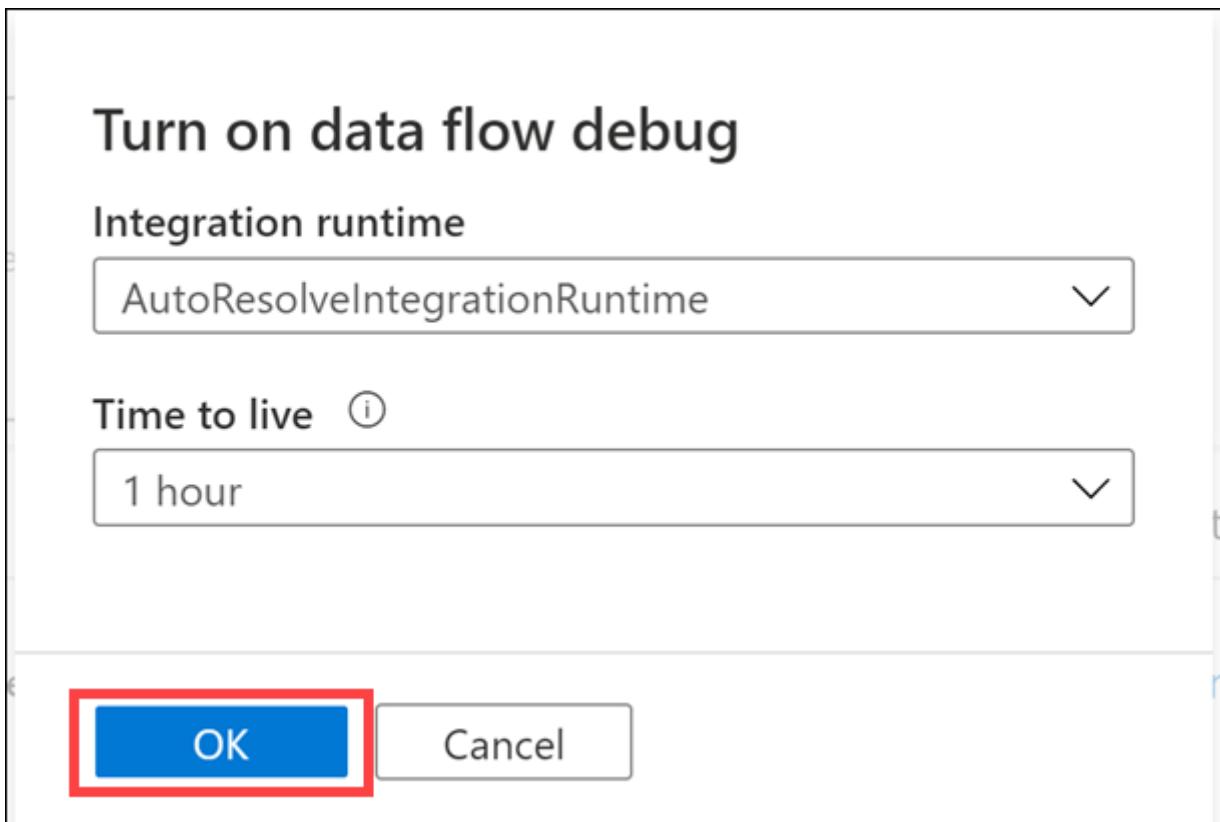
3. In the properties pane of the new data flow, enter `UpdateCustomerDimension` in the **Name** field (1), then select the **Properties** button (2) to hide the properties pane.



4. Select **Data flow debug** to enable the debugger. This will allow us to preview data transformations and debug the data flow before executing it in a pipeline.

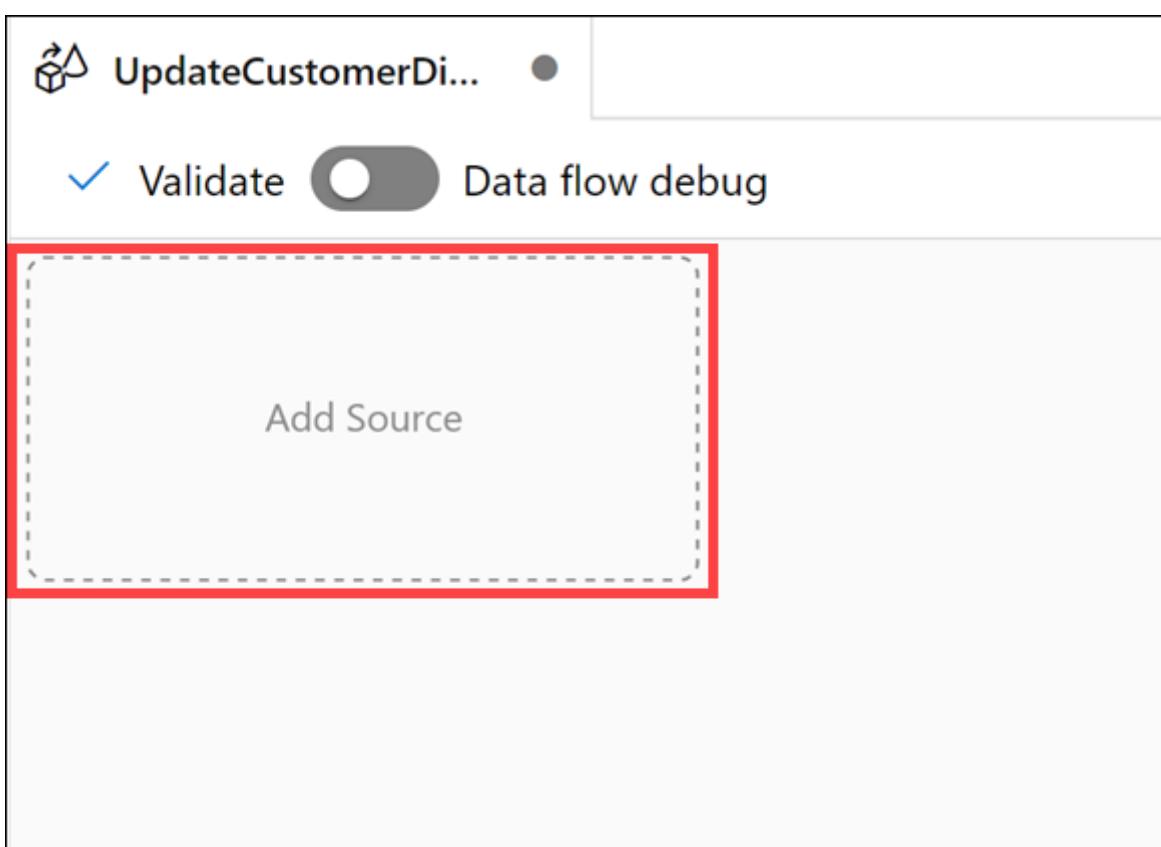


5. Select **OK** in the dialog that appears to turn on the data flow debug.



The debug cluster will start in a few minutes. In the meantime, you can continue with the next step.

6. Select **Add Source** on the canvas.



7. Under **Source settings**, configure the following properties:

- **Output stream name:** Enter `SourceDB`
- **Source type:** Select `Dataset`
- **Options:** Check `Allow schema drift` and leave the other options unchecked

- **Sampling:** Select **Disable**
- **Dataset:** Select **+** **New** to create a new dataset

Source settings Source options Projection Optimize Inspect Data preview

Output stream name * Learn more [↗](#)

Source type * Dataset Inline

Dataset *

Options

Allow schema drift ⓘ

Infer drifted column types ⓘ

Validate schema ⓘ

Sampling * ⓘ Enable Disable

8. In the new integration dataset dialog, select **Azure SQL Database**, then select **Continue**.

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

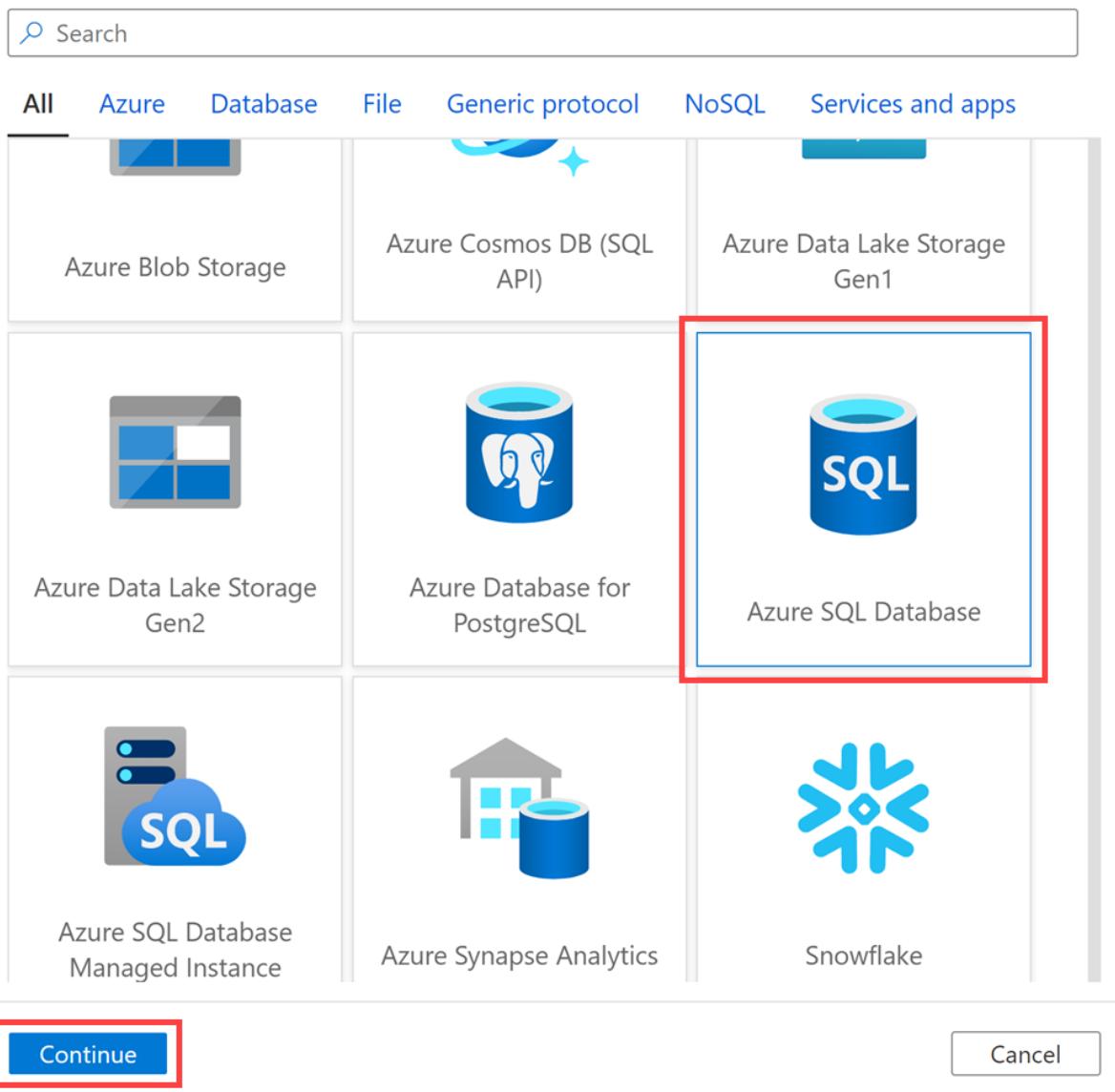
Select a data store

Search

All Azure Database File Generic protocol NoSQL Services and apps

Azure Blob Storage	Azure Cosmos DB (SQL API)	Azure Data Lake Storage Gen1
Azure Data Lake Storage Gen2	Azure Database for PostgreSQL	Azure SQL Database
Azure SQL Database Managed Instance	Azure Synapse Analytics	Snowflake

Continue **Cancel**



9. In the dataset properties, configure the following:

- **Name:** Enter SourceCustomer
- **Linked service:** Select AzureSqlDatabaseSource
- **Table name:** Select SalesLT.Customer

Set properties

Name
SourceCustomer

Linked service *
AzureSqlDataSource

Connect via integration runtime * ⓘ
AutoResolveIntegrationRuntime

Table name
SalesLT.Customer

Edit

Import schema
 From connection/store None

► Advanced

OK **Back** **Cancel**

10. Select **OK** to create the dataset.
11. The **SourceCustomer** dataset should now appear and be selected as the dataset for the source settings.

Source settings **Source options** **Projection** **Optimize** **Inspect** **Data preview** ●

Output stream name * SourceDB [Learn more](#) ↗

Source type * Dataset

Dataset * **SourceCustomer** [Test connection](#) [Open](#) [New](#)

Options
 Allow schema drift ⓘ
 Infer drifted column types ⓘ
 Validate schema ⓘ

Sampling * ⓘ Enable Disable

12. Select + to the right of the SourceDB source on the canvas, then select **Derived Column**.

The screenshot shows the Data Flow Designer interface. At the top, there is a component named "SourceDB" with a tooltip indicating "Columns: 15 total". Below it is a search bar and a list of "Multiple inputs/outputs" components: Join, Conditional Split, Exists, Union, and Lookup. A red box highlights the "Derived Column" option under the "Schema modifier" section. On the left, there is a panel titled "Add Source" and tabs for "Source settings" and "Source options".

13. Under **Derived column's settings**, configure the following properties:

- **Output stream name:** Enter `CreateCustomerHash`
- **Incoming stream:** Select `SourceDB`
- **Columns:** Enter the following:

Column	Expression
Type in HashKey	<code>sha2(256, iifNull>Title,'') +FirstName +iifNull(MiddleName,'') +LastName +iifNull(Suffix,'')</code>

The screenshot shows the "Derived column's settings" dialog. It includes fields for "Output stream name" (set to `CreateCustomerHash`) and "Incoming stream" (set to `SourceDB`). Below these are sections for "Columns" and "Expression". A red box highlights the "Expression" field, which contains the formula `sha2(256, Title+FirstName+MiddleName+LastName)`. The dialog also features tabs for "Optimize", "Inspect", "Data preview", and "Description".

14. Click in the **Expression** text box, then select **Open expression builder**.

The screenshot shows the 'Derived column's settings' interface. At the top, there are tabs for 'Derived column's settings', 'Optimize', 'Inspect', and 'Data preview'. A 'Description' button is located in the top right. Below these, the 'Output stream name' is set to 'CreateCustomerHash' and the 'Incoming stream' is 'SourceDB'. Under the 'Columns' section, there is a table with one row. The 'Column' is 'HashKey' and the 'Expression' is 'sha2(256, iifNull>Title, '') + FirstName + iifNull(MiddleName, '') + LastName + iifNull(Suffix, '') + iifNull(CompanyName, '')'. The 'Open expression builder' button is highlighted with a red box.

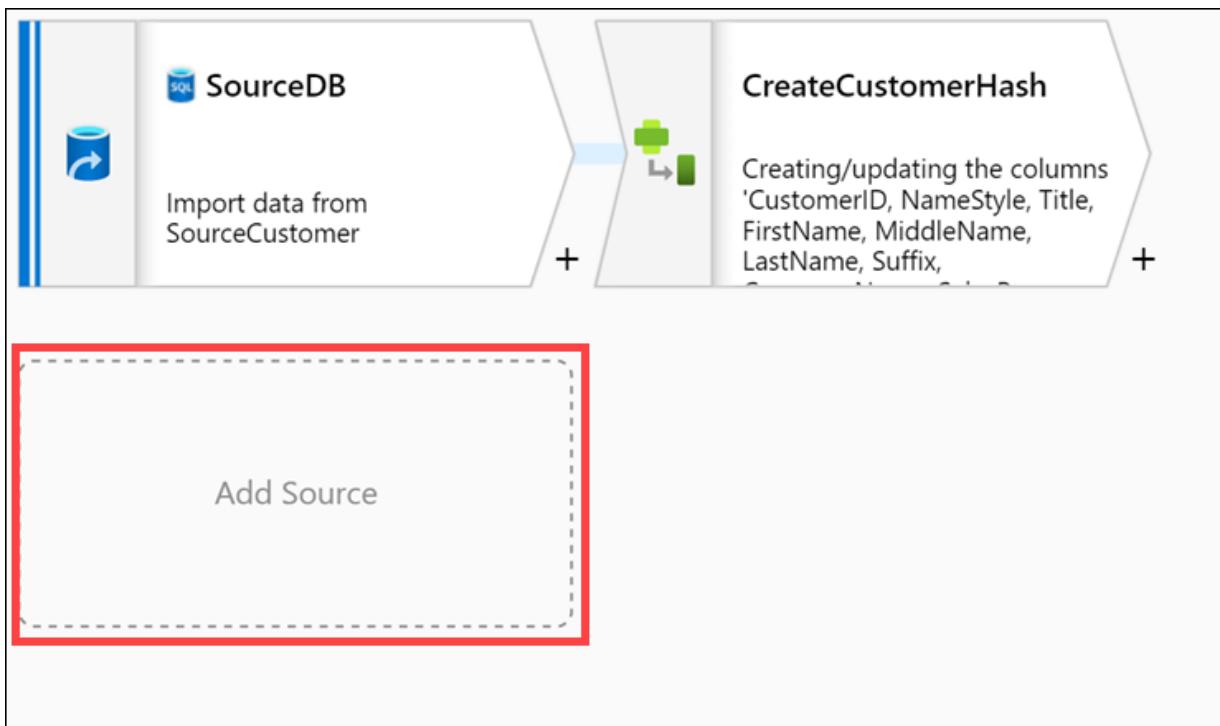
15. Select **Refresh** next to **Data preview** to preview the output of the HashKey column, which uses the sha2 function you added. You should see that each hash value is unique.

The screenshot shows the 'Visual expression builder' interface. On the left, there's a sidebar with 'Derived Columns' and a selected item 'abc HashKey'. The main area has tabs for 'Expression' (selected) and 'Data preview'. In the 'Expression' tab, the expression 'sha2(256, iifNull>Title, '') + FirstName + iifNull(MiddleName, '') + LastName + iifNull(Suffix, '') + iifNull(CompanyName, '')' is shown. In the 'Data preview' tab, there is a table with four columns: 'Output: HashKey' and three columns from the incoming stream: 'Title', 'FirstName', and 'LastName'. The table contains four rows of data. The 'Data preview' tab is highlighted with a red box.

Output: HashKey	Title	FirstName	MiddleName	LastName	Suffix
072ab0f11bf0e9d0d4ef609e16c383690d1e682bf2724ab254...	Mr.	Orlando	N.	Gee	NULL
1054ed4cc3a053679c80c18ec105ef1162174007af62c5f055e...	Mr.	Keith	NULL	Harris	NULL
6e8c44384bc55955a1cc304ee7102f30759fdbff2c3be4462c...	Ms.	Donna	F.	Carreras	NULL
0beb746affd6061903079e329029158c85cf5a1b9553a49ea5...	Ms.	Janet	M.	Gates	NULL

16. Select **Save and finish** to close the expression builder.

17. Select **Add Source** on the canvas underneath the **SourceDB** source. We need to add the **DimCustomer** table located in the Synapse dedicated SQL pool to use when comparing the existence of records and for comparing hashes.



18. Under **Source settings**, configure the following properties:

- **Output stream name:** Enter `SynapseDimCustomer`
- **Source type:** Select `Dataset`
- **Options:** Check `Allow schema drift` and leave the other options unchecked
- **Sampling:** Select `Disable`
- **Dataset:** Select `+ New` to create a new dataset

The screenshot shows the 'Source settings' tab of the dataset creation dialog. The 'Output stream name' field is set to `SynapseDimCustomer`. The 'Source type' section shows `Dataset` selected. The 'Dataset' dropdown menu has a red border around it, and the '`+ New`' button next to it is also highlighted with a red box. In the 'Options' section, the `Allow schema drift` checkbox is checked. The 'Sampling' section shows `Disable` selected.

19. In the new integration dataset dialog, select **Azure Synapse Analytics**, then select **Continue**.

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

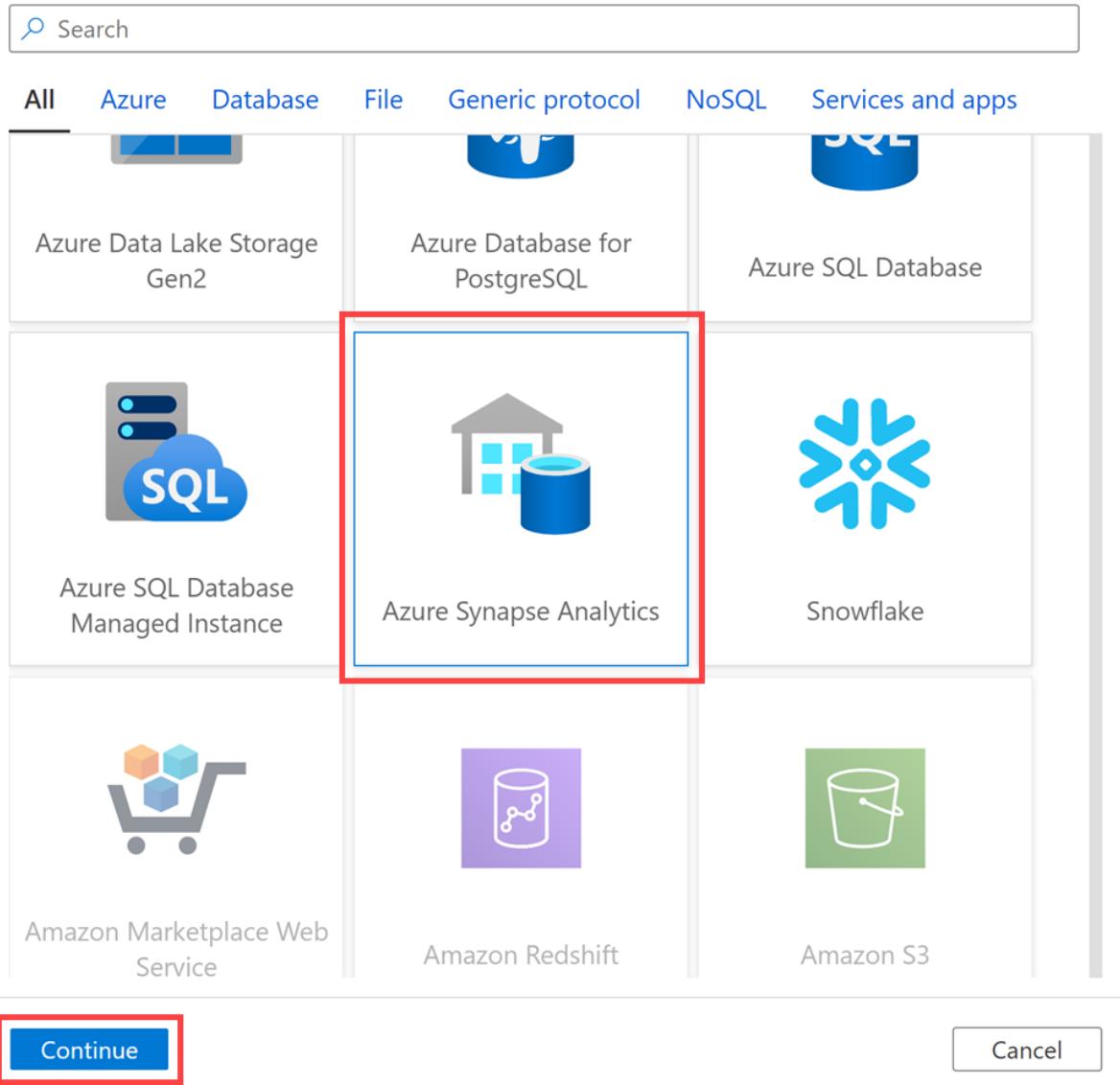
Select a data store

Search

All Azure Database File Generic protocol NoSQL Services and apps

Azure Data Lake Storage Gen2	Azure Database for PostgreSQL	Azure SQL Database
Azure SQL Database Managed Instance	Azure Synapse Analytics	Snowflake
Amazon Marketplace Web Service	Amazon Redshift	Amazon S3

Continue **Cancel**



20. In the dataset properties, configure the following:

- **Name:** Enter DimCustomer
- **Linked service:** Select the Synapse workspace linked service
- **Table name:** Select the Refresh button next to the dropdown

Set properties

Name
DimCustomer

Linked service *
asagaworkspacejdh013121-WorkspaceDefaultSqlServer

Connect via integration runtime * ⓘ
AutoResolveIntegrationRuntime

Table name
 Edit

Import schema
 From connection/store None

► Advanced

21. In the **Value** field, enter **SQLPool01**, then select **OK**.

Please provide actual value of the parameters to list tables

Parameters for linked service asagaworkspacejdh013121-
WorkspaceDefaultSqlServer

Name	Type	Value
DBName	String	<input type="text" value="SQLPool01"/>

22. Select **dbo.DimCustomer** under **Table name**, select **From connection/store** under **Import schema**, then select **OK** to create the dataset.

Set properties

Name
DimCustomer

Linked service *
asagaworkspacejdh013121-WorkspaceDefaultSqlServer

Connect via integration runtime * ⓘ
AutoResolveIntegrationRuntime

Table name
dbo.DimCustomer

Import schema
From connection/store

OK Back Cancel

23. The DimCustomer dataset should now appear and be selected as the dataset for the source settings.

Source settings Source options Projection Optimize Inspect Data preview ●

Output stream name * SynapseDimCustomer Learn more ↗

Source type * Dataset

Dataset * DimCustomer

Options Allow schema drift ⓘ
Infer drifted column types ⓘ
Validate schema ⓘ

Sampling * ⓘ Enable Disable

24. Select **Open** next to the DimCustomer dataset that you added.

Source settings Source options Projection Optimize Inspect Data preview ●

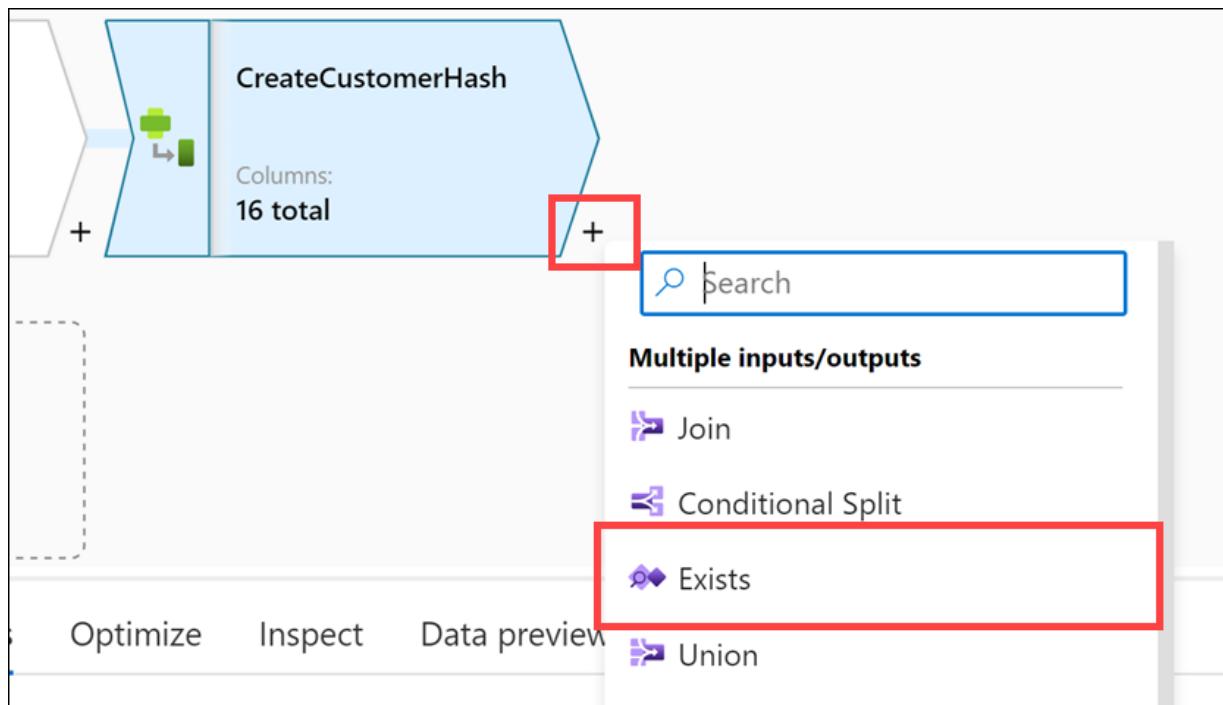
Output stream name *	SynapseDimCustomer	Learn more
Source type *	Dataset	Test connection Open New
Dataset *	DimCustomer	
Options	<input checked="" type="checkbox"/> Allow schema drift <small> ⓘ </small> <input type="checkbox"/> Infer drifted column types <small> ⓘ </small> <input type="checkbox"/> Validate schema <small> ⓘ </small>	
Sampling * ⓘ	<input type="radio"/> Enable <input checked="" type="radio"/> Disable	

25. Enter SQLPool01 in the **Value** field next to **DBName**.

Azure Synapse Analytics
DimCustomer

Connection	Schema	Parameters						
Linked service *	asagaworkspacejdh013121-Workspa... Test connection Edit New Learn more							
Linked service properties ⓘ <table border="1"> <thead> <tr> <th>NAME</th> <th>VALUE</th> <th>TYPE</th> </tr> </thead> <tbody> <tr> <td>DBName</td> <td>SQLPool01</td> <td>String</td> </tr> </tbody> </table>			NAME	VALUE	TYPE	DBName	SQLPool01	String
NAME	VALUE	TYPE						
DBName	SQLPool01	String						
Integration runtime *	AutoResolveIntegrationRuntime Edit							
Table	dbo	. DimCustomer Preview data						

26. Switch back to your data flow. *Do not* close the **DimCustomer** dataset. Select + to the right of the **CreateCustomerHash** derived column on the canvas, then select **Exists**.



27. Under **Exists settings**, configure the following properties:

- **Output stream name:** Enter Exists
- **Left stream:** Select CreateCustomerHash
- **Right stream:** Select SynapseDimCustomer
- **Exist type:** Select Doesn't exist
- **Exists conditions:** Set the following for Left and Right:

Left: CreateCustomerHash's column	Right: SynapseDimCustomer's column
HashKey	HashKey

Exists settings Optimize Inspect Data preview ● Description ^

Output stream name * Learn more ▾

Left stream *

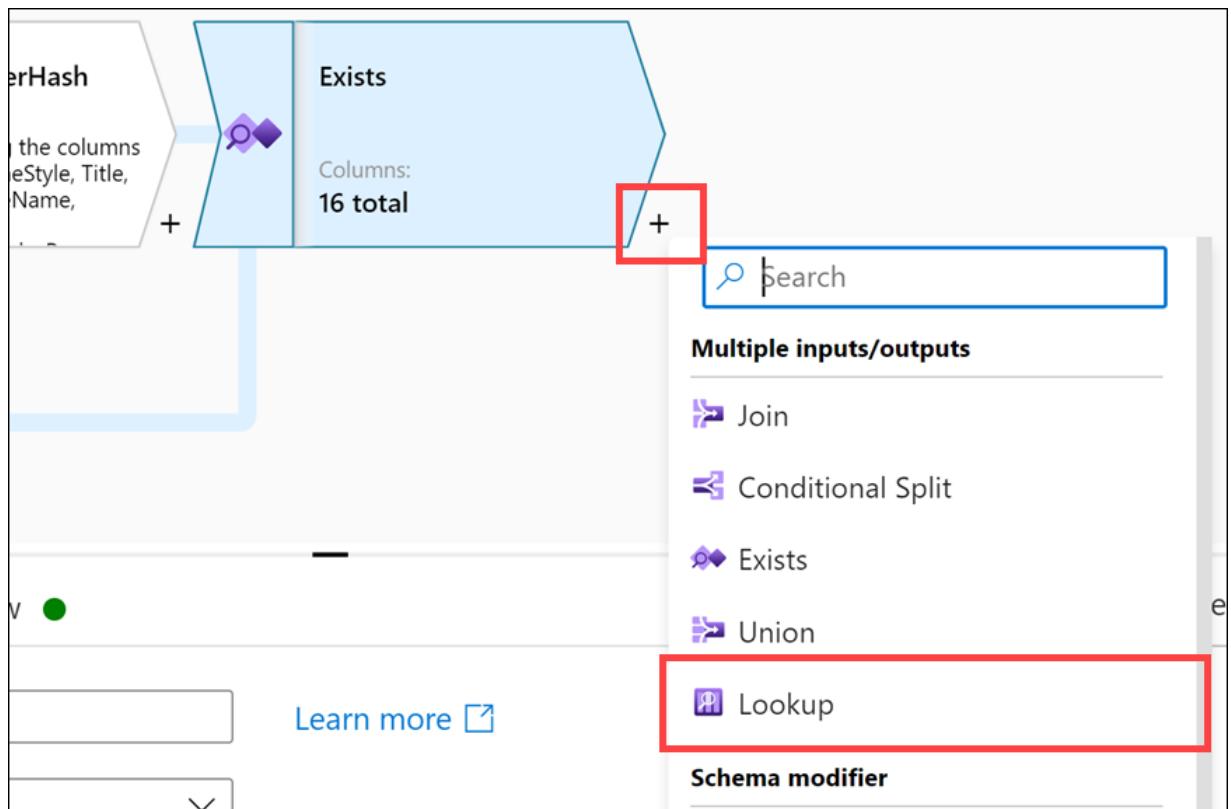
Right stream *

Exist type * Doesn't exist

Custom expression ⓘ

Exists conditions * == + -

28. Select **+** to the right of **Exists** on the canvas, then select **Lookup**.



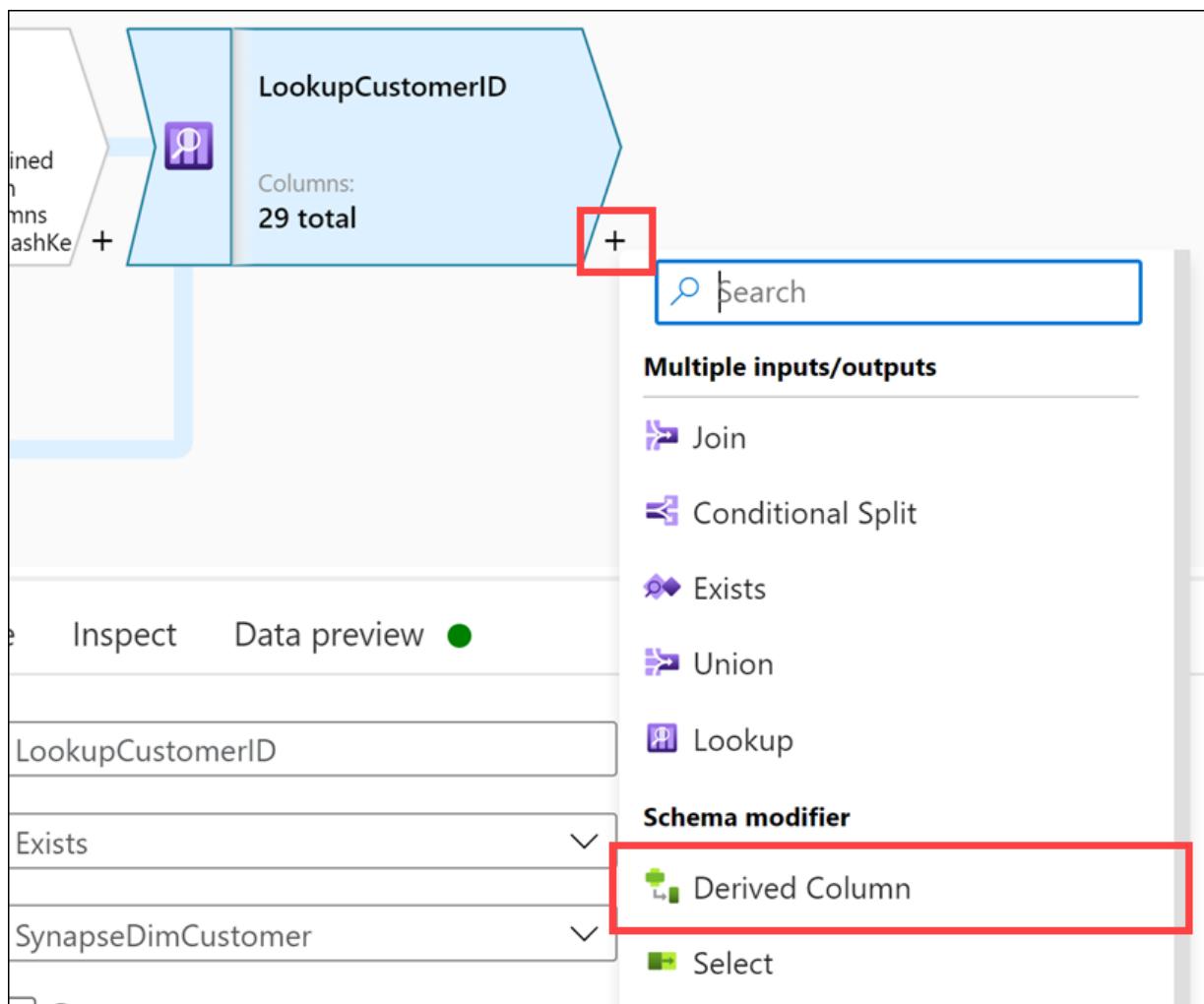
29. Under **Lookup settings**, configure the following properties:

- Output stream name:** Enter `LookupCustomerID`
- Primary stream:** Select `Exists`
- Lookup stream:** Select `SynapseDimCustomer`
- Match multiple rows:** Unchecked
- Match on:** Select `Any row`
- Lookup conditions:** Set the following for Left and Right:

Left: Exists's column	Right: SynapseDimCustomer's column
<code>CustomerID</code>	<code>CustomerID</code>

Output stream name *	<code>LookupCustomerID</code>					
Primary stream *	<code>Exists</code>					
Lookup stream *	<code>SynapseDimCustomer</code>					
Match multiple rows	<input type="checkbox"/>					
Match on *	<code>Any row</code>					
Lookup conditions *	<table border="1"> <tr> <td>Left: Exists's column</td> <td><code>123 CustomerID</code></td> <td><code>=</code></td> <td>Right: SynapseDimCustomer's column</td> <td><code>123 CustomerID</code></td> </tr> </table>	Left: Exists's column	<code>123 CustomerID</code>	<code>=</code>	Right: SynapseDimCustomer's column	<code>123 CustomerID</code>
Left: Exists's column	<code>123 CustomerID</code>	<code>=</code>	Right: SynapseDimCustomer's column	<code>123 CustomerID</code>		

30. Select `+` to the right of `LookupCustomerID` on the canvas, then select **Derived Column**.



31. Under **Derived column's settings**, configure the following properties:

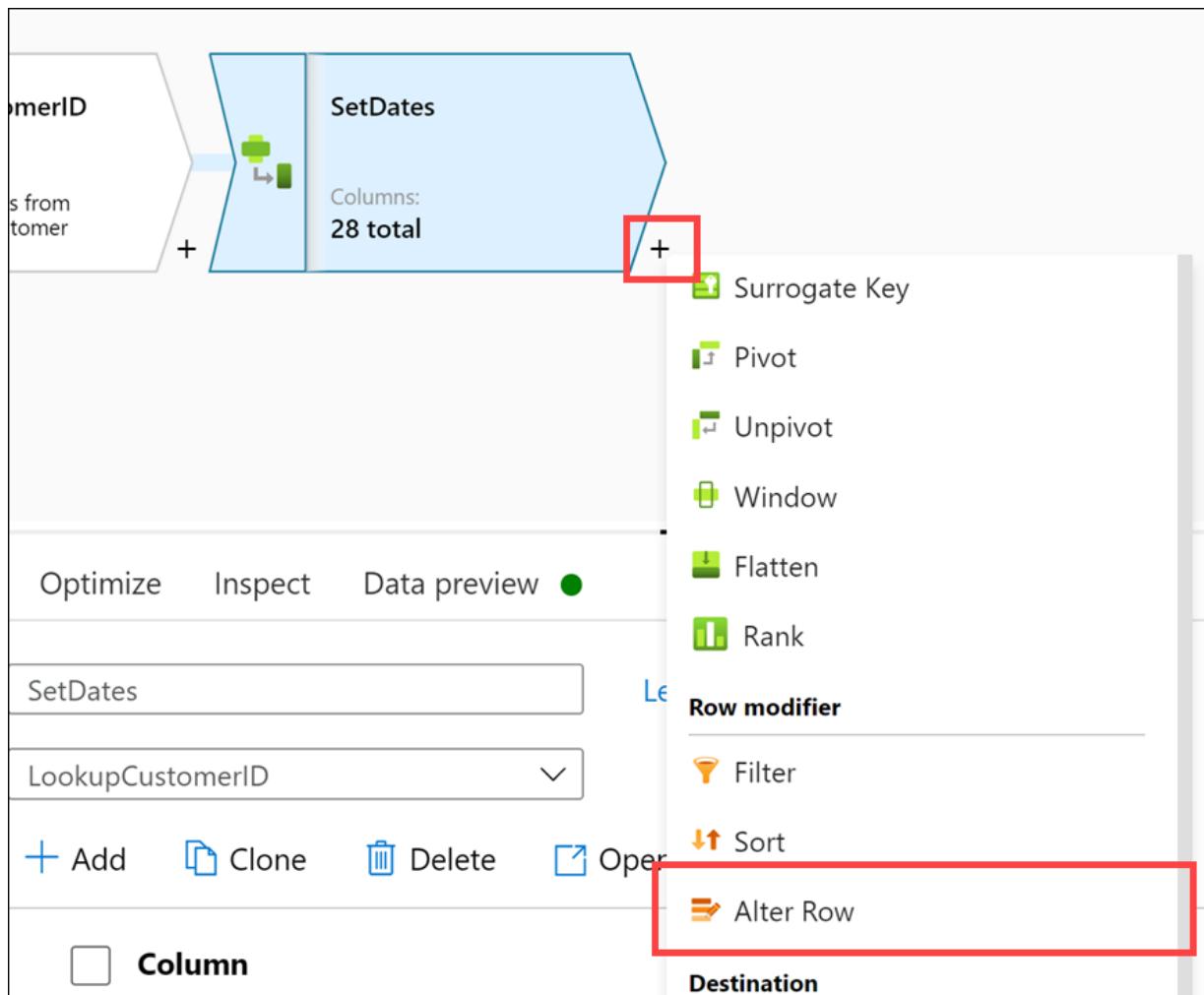
- **Output stream name:** Enter **SetDates**
- **Incoming stream:** Select **LookupCustomerID**
- **Columns:** Enter the following:

Column	Expression	Description
Select InsertedDate	iif(isNull(InsertedDate), currentTimestamp(), {InsertedDate})	If the InsertedDate value is null, return the current timestamp.
Select ModifiedDate	currentTimestamp()	Always update the ModifiedDate to the current timestamp.

The screenshot shows the 'Derived column's settings' dialog. It includes fields for 'Output stream name' (SetDates), 'Incoming stream' (LookupCustomerID), and a 'Columns' section with two entries: 'InsertedDate' and 'ModifiedDate'.

Note: To insert the second column, select **+ Add** above the Columns list, then select **Add column**.

32. Select **+** to the right of the **SetDates** derived column step on the canvas, then select **Alter Row**.



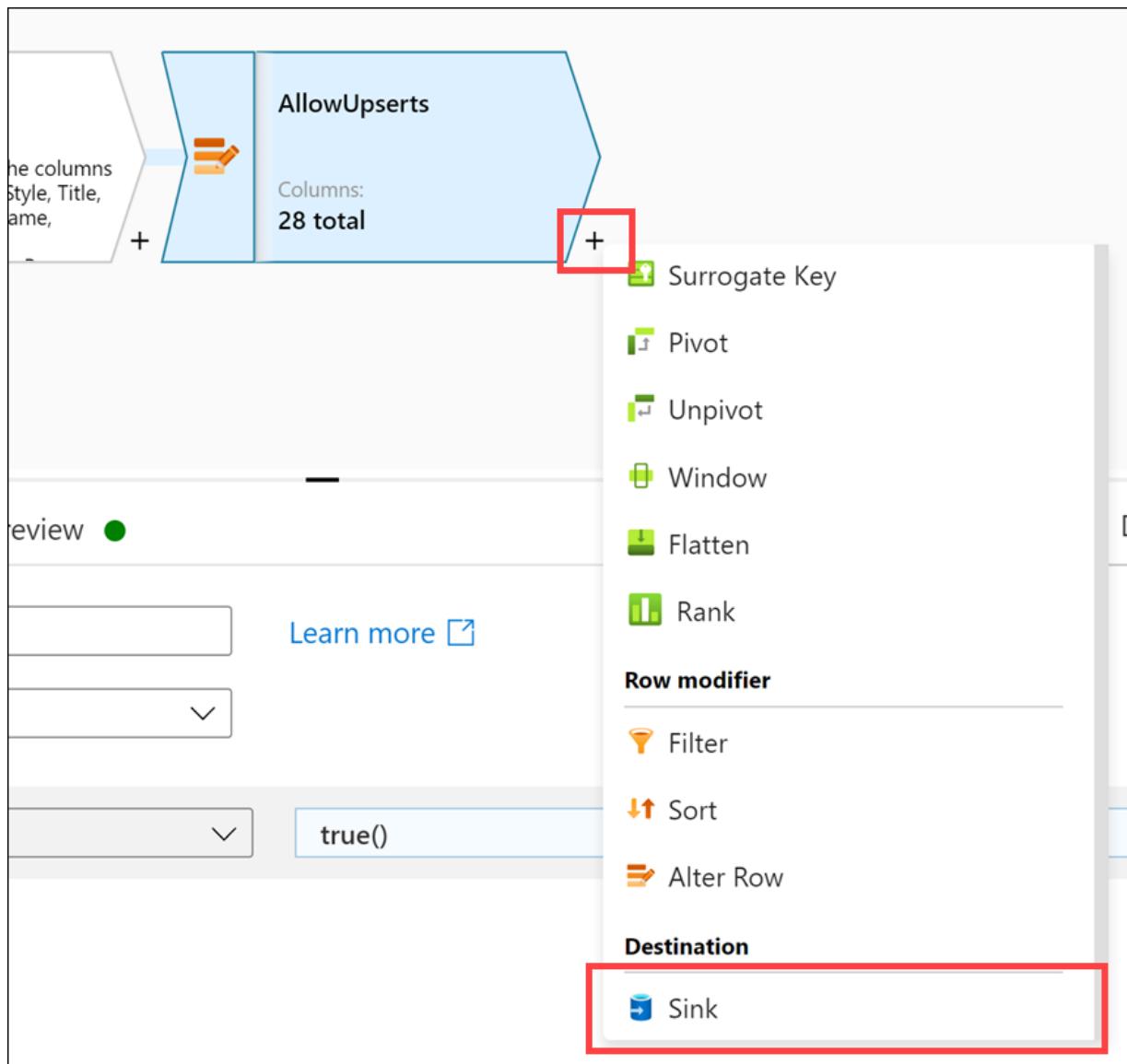
33. Under **Alter row settings**, configure the following properties:

- **Output stream name:** Enter **AllowUpserts**
- **Incoming stream:** Select **SetDates**
- **Alter row conditions:** Enter the following:

Condition	Expression	Description
Select Upsert if	true()	Set the condition to true() on the Upsert if condition to allow upserts. This ensures th

The screenshot shows the 'Alter row settings' dialog. In the 'Alter row conditions' section, there is a dropdown menu with an 'Upstart if' option and a 'true()' expression. This entire section is highlighted with a red box.

34. Select + to the right of the **AllowUpserts** alter row step on the canvas, then select **Sink**.



35. Under Sink, configure the following properties:

- **Output stream name:** Enter Sink
- **Incoming stream:** Select AllowUpserts
- **Sink type:** Select Dataset
- **Dataset:** Select DimCustomer
- **Options:** Check Allow schema drift and uncheck Validate schema

The screenshot shows the 'Sink' configuration dialog. It has tabs for Sink, Settings, Mapping, Optimize, Inspect, and Data preview. The 'Sink' tab is active. The 'Output stream name' field contains 'sink'. The 'Incoming stream' dropdown is set to 'AllowUpserts'. Under 'Sink type', the 'Dataset' radio button is selected, while 'Inline' and 'Cache' are unselected. The 'Dataset' dropdown is set to 'DimCustomer'. In the 'Options' section, the 'Allow schema drift' checkbox is checked, and the 'Validate schema' checkbox is unchecked. There are buttons for 'Test connection', 'Open', and 'New'.

36. Select the **Settings** tab and configure the following properties:

- **Update method:** Check **Allow upsert** and uncheck all other options
- **Key columns:** Select **List of columns**, then select **CustomerID** in the list
- **Table action:** Select **None**
- **Enable staging:** Unchecked

The screenshot shows the 'Settings' tab selected in the top navigation bar. A red box highlights the 'Settings' tab. Below it, a message says 'We recommend enabling staging to improve performance with Azure Synapse Analytics datasets.' Under 'Update method', 'Allow upsert' is checked (highlighted by a red box). Under 'Key columns', 'List of columns' is selected, and 'CustomerID' is listed in the dropdown (highlighted by a red box). Under 'Table action', 'None' is selected. Under 'Enable staging', the checkbox is unchecked. The 'Batch size' field is empty.

37. Select the **Mapping** tab, then uncheck **Auto mapping**. Configure the input columns mapping as outlined below:

Input columns	Output columns
SourceDB@CustomerID	CustomerID
SourceDB@Title	Title
SourceDB@FirstName	FirstName
SourceDB@MiddleName	MiddleName
SourceDB@LastName	LastName
SourceDB@Suffix	Suffix
SourceDB@CompanyName	CompanyName
SourceDB@SalesPerson	SalesPerson
SourceDB@EmailAddress	EmailAddress
SourceDB@Phone	Phone
InsertedDate	InsertedDate
ModifiedDate	ModifiedDate
CreateCustomerHash@HashKey	HashKey

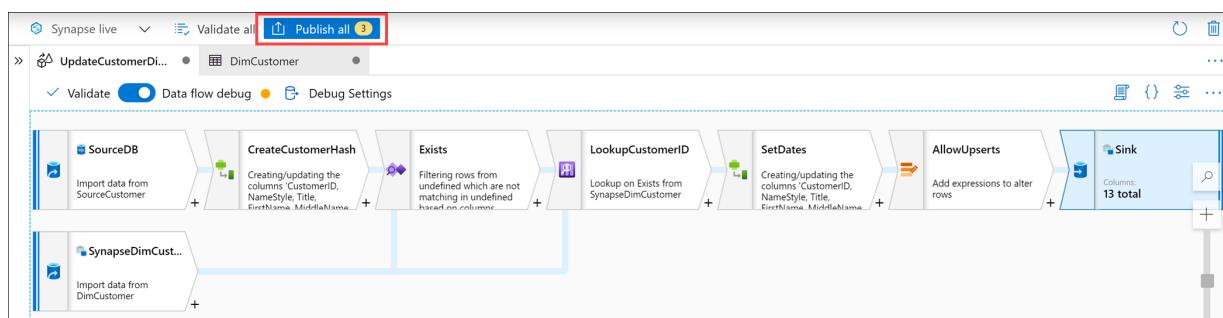
Sink Settings **Mapping** Optimize Inspect Data preview ● Description

Options Skip duplicate input columns ⓘ Skip duplicate output columns ⓘ Auto mapping ⓘ

13 mappings: All outputs mapped

Input columns	Output columns
123 SourceDB@CustomerID	123 CustomerID
abc SourceDB@Title	abc Title
abc SourceDB@FirstName	abc FirstName
abc SourceDB@MiddleName	abc MiddleName
abc SourceDB@LastName	abc LastName
abc SourceDB@Suffix	abc Suffix
abc SourceDB@CompanyName	abc CompanyName
abc SourceDB@SalesPerson	abc SalesPerson
abc SourceDB@EmailAddress	abc EmailAddress
abc SourceDB@Phone	abc Phone
InsertedDate	InsertedDate
ModifiedDate	ModifiedDate
abc CreateCustomerHash@HashKey	abc HashKey

38. The completed mapping flow should look like the following. Select **Publish all** to save your changes.



39. Select **Publish**.

Publish all

You are about to publish all pending changes to the live environment. [Learn more](#)

Pending changes (3)

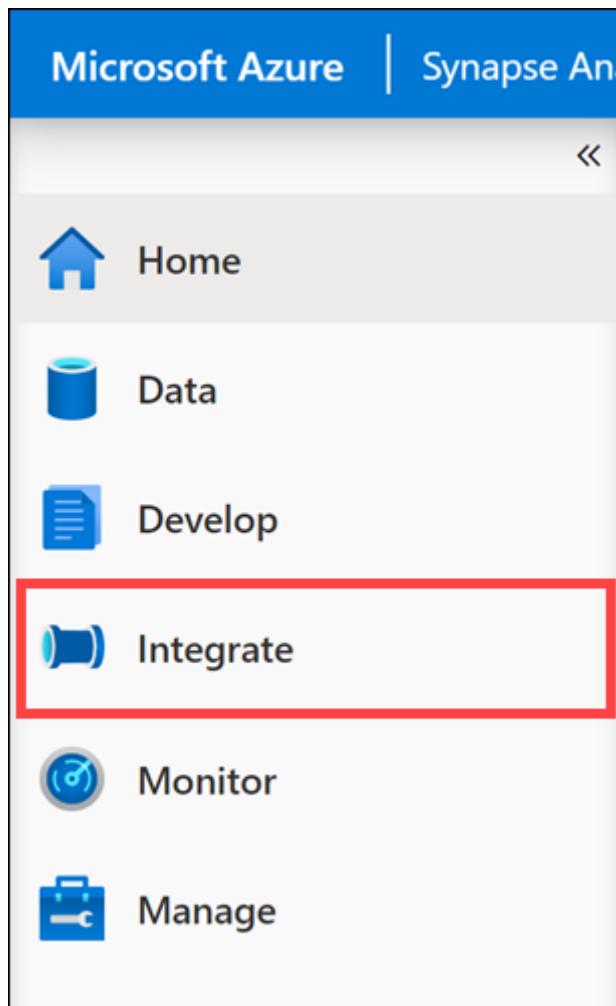
NAME	CHANGE	EXISTING
▲ Datasets		
SourceCustomer	(New)	-
DimCustomer	(New)	-
▲ Data flows		
UpdateCustomerDimension	(New)	-

Publish Cancel

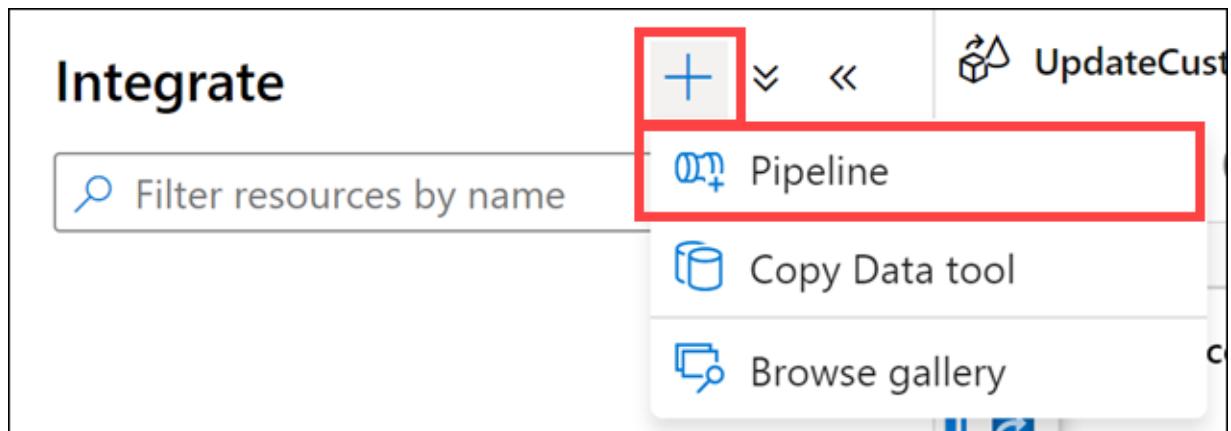
4.7.3 Task 3: Create a pipeline and run the data flow

In this task, you create a new Synapse integration pipeline to execute the mapping data flow, then run it to upsert customer records.

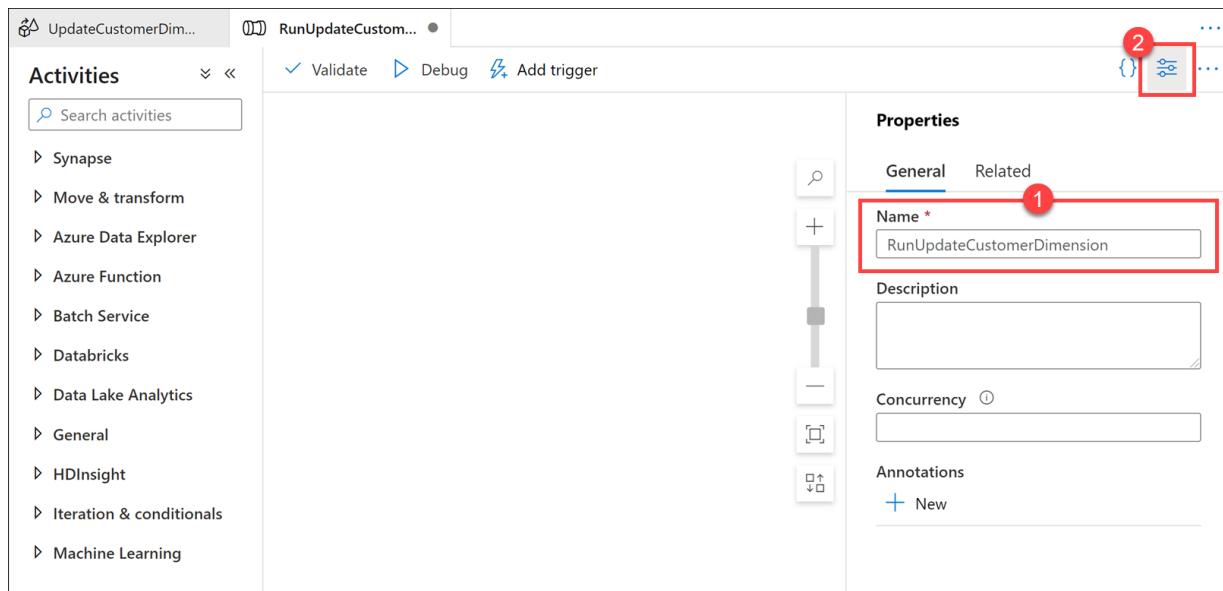
1. Navigate to the **Integrate** hub.



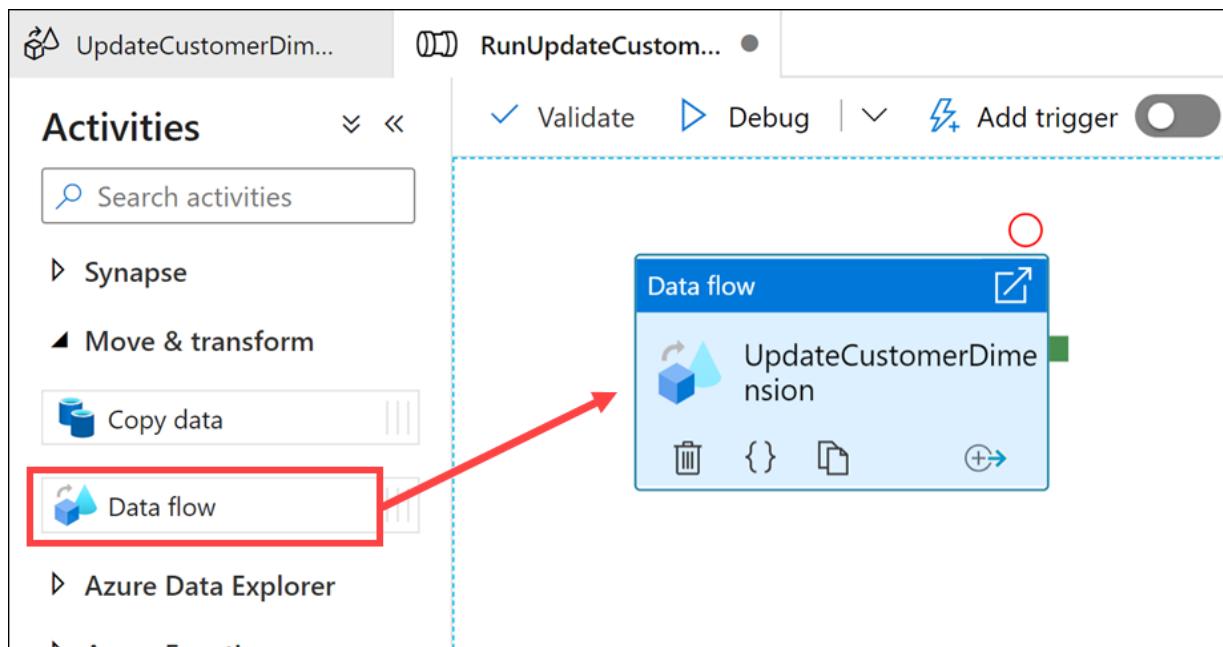
2. Select +, then select Pipeline.



3. In the properties pane of the new pipeline, enter RunUpdateCustomerDimension in the Name field (1), then select the Properties button (2) to hide the properties pane.



- Under the Activities pane to the left of the design canvas, expand **Move & transform**, then drag and drop the **Data flow** activity onto the canvas.



- Under the General tab, enter **UpdateCustomerDimension** for the name.

General Settings ¹ Parameters ¹ User properties

Name *	UpdateCustomerDimension	Learn more
Description		
Timeout ⓘ	7.00:00:00	
Retry ⓘ	0	
Retry interval ⓘ	30	
Secure output ⓘ	<input type="checkbox"/>	
Secure input ⓘ	<input type="checkbox"/>	

- Under the **Settings** tab, select the **UpdateCustomerDimension** data flow.

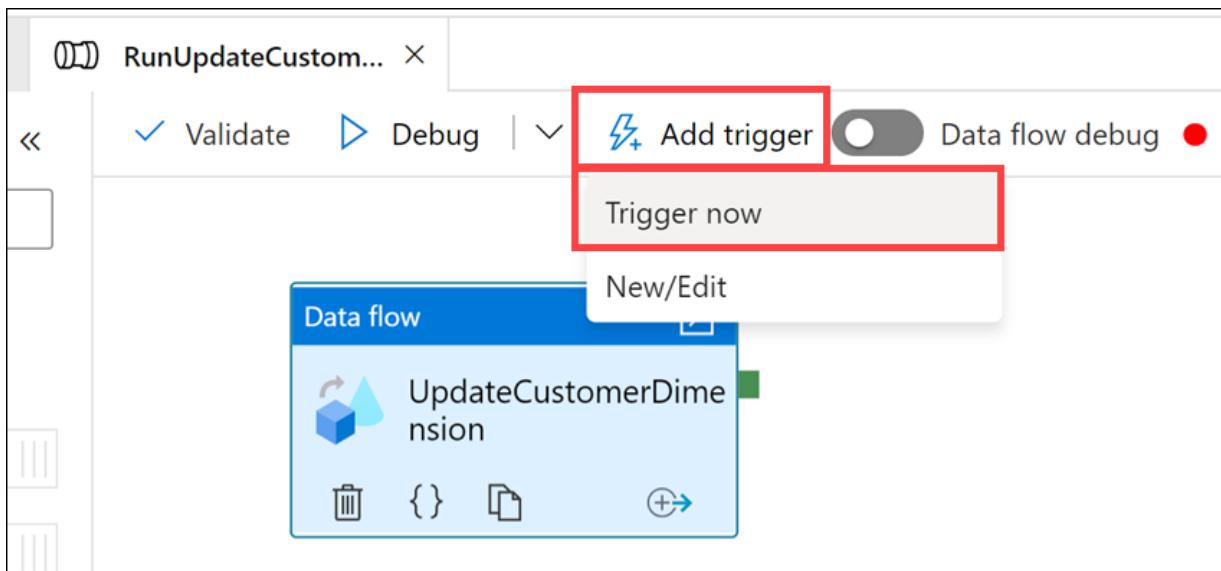
General **Settings** Parameters ¹ User properties

Data flow *	UpdateCustomerDimension	Open	New
Run on (Azure IR) * ⓘ	AutoResolveIntegrationRuntime	▼	
Compute type *	General purpose	▼	
Core count *	4 (+ 4 Driver cores)	▼	
Logging level * ⓘ	<input checked="" type="radio"/> Verbose <input type="radio"/> Basic <input type="radio"/> None		
Sink properties			
Staging			
Staging linked service ⓘ	Select...	New	
Staging storage folder	Container	/	Directory Browse

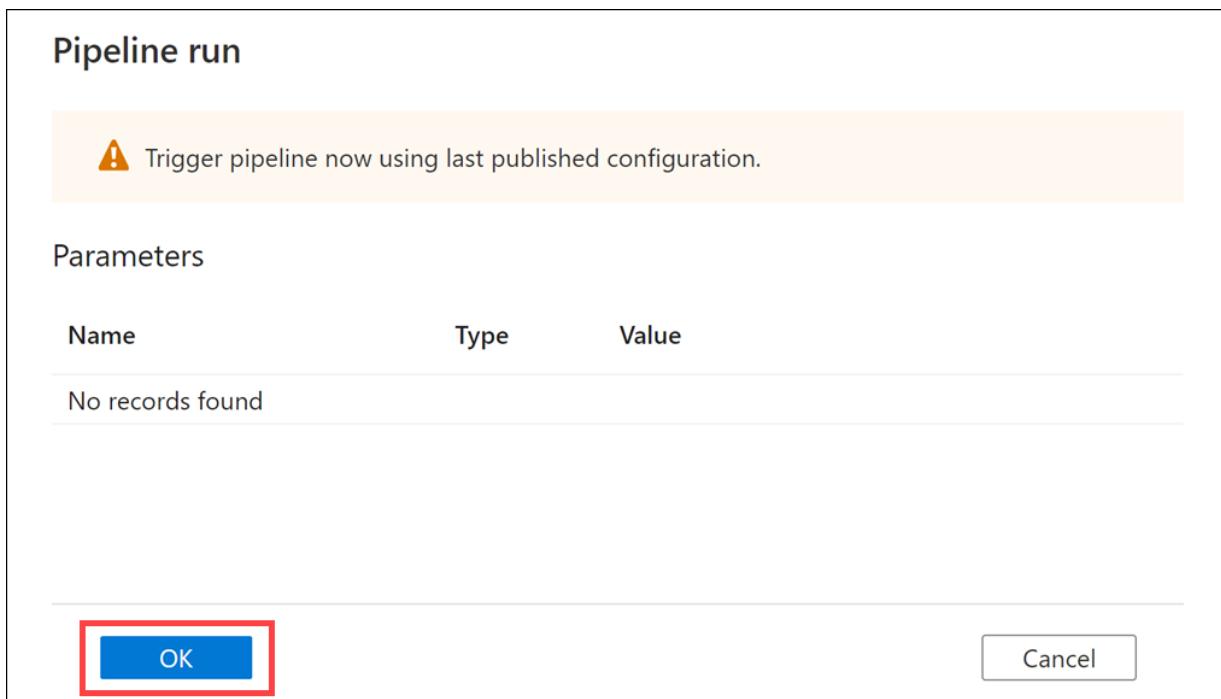
- Select **Publish all**, then select **Publish** in the dialog that appears.



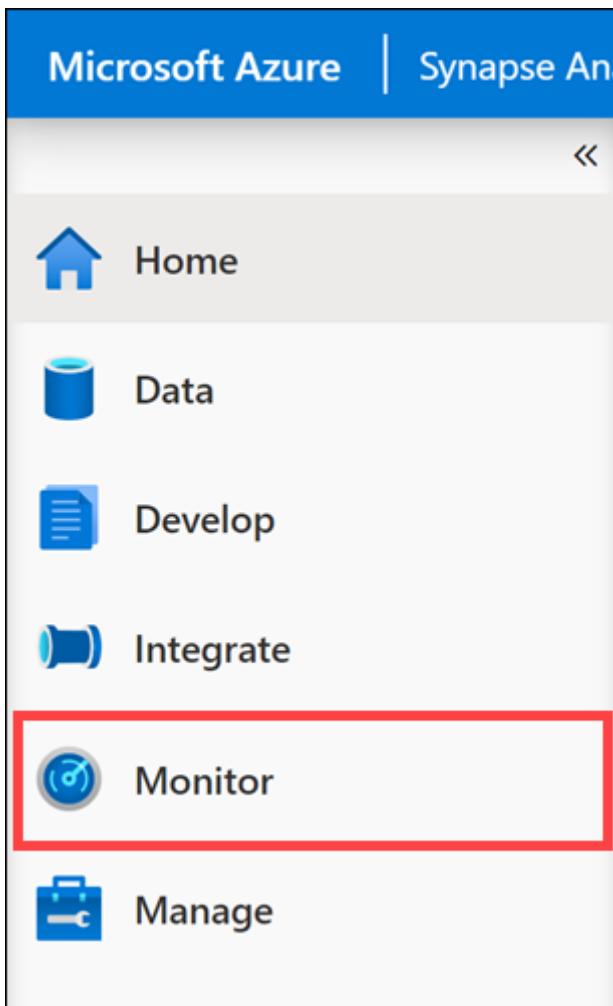
- After publishing completes, select **Add trigger** above the pipeline canvas, then select **Trigger now**.



9. Select **OK** in the Pipeline run dialog to trigger the pipeline.



10. Navigate to the **Monitor** hub.

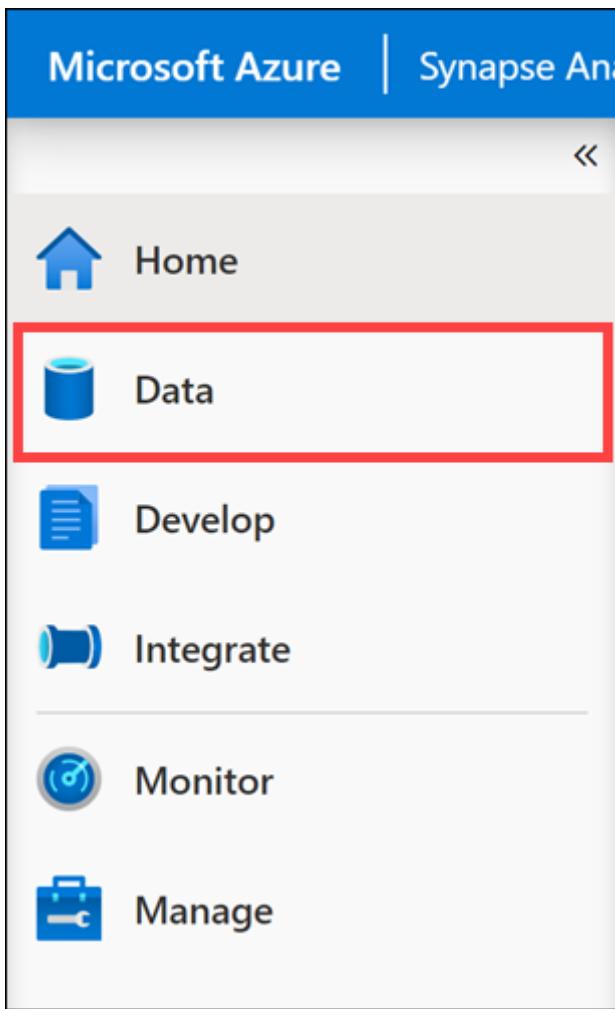


- Select **Pipeline runs** in the left-hand menu (1) and wait for the pipeline run to successfully complete (2). You may have to select **Refresh** (3) several times until the pipeline run completes.

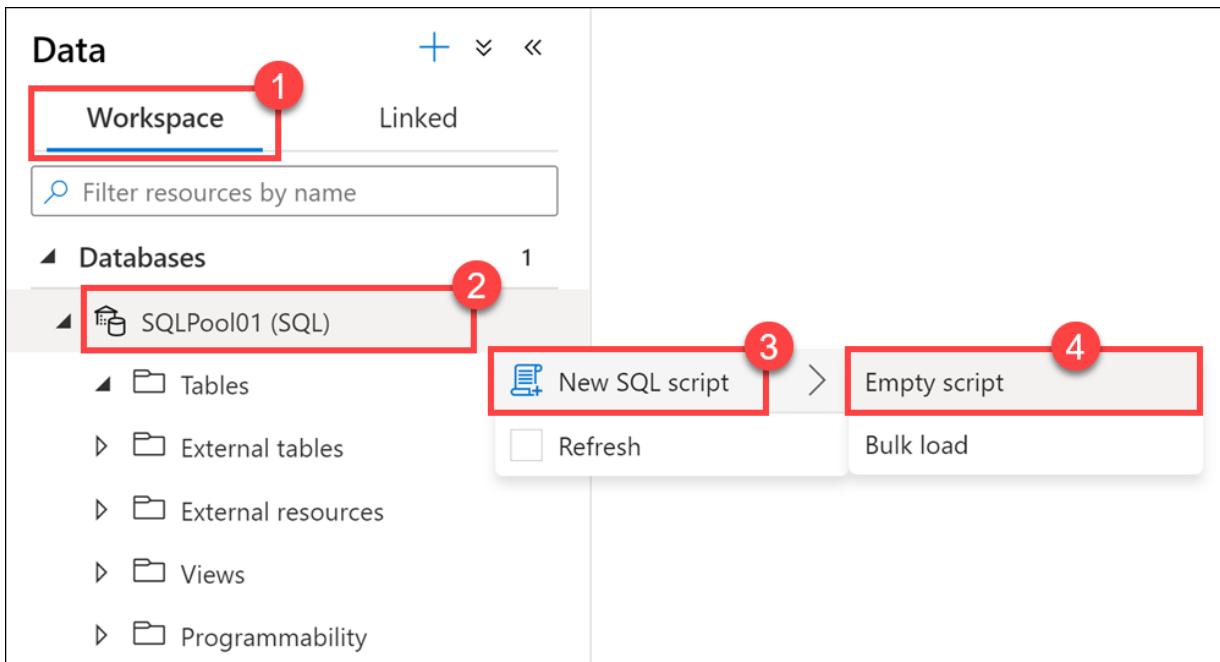
Pipeline name	Run start	Run end	Duration	Triggered by	Status
RunUpdateCustomerDimension	2/1/21, 3:16:32 AM	2/1/21, 3:21:50 AM	00:05:18	Manual trigger	✓ Succeeded

4.7.4 Task 4: View inserted data

1. Navigate to the **Data** hub.



2. Select the **Workspace** tab (1), expand Databases, then right-click on **SQLPool01** (2). Select **New SQL script** (3), then select **Empty script** (4).



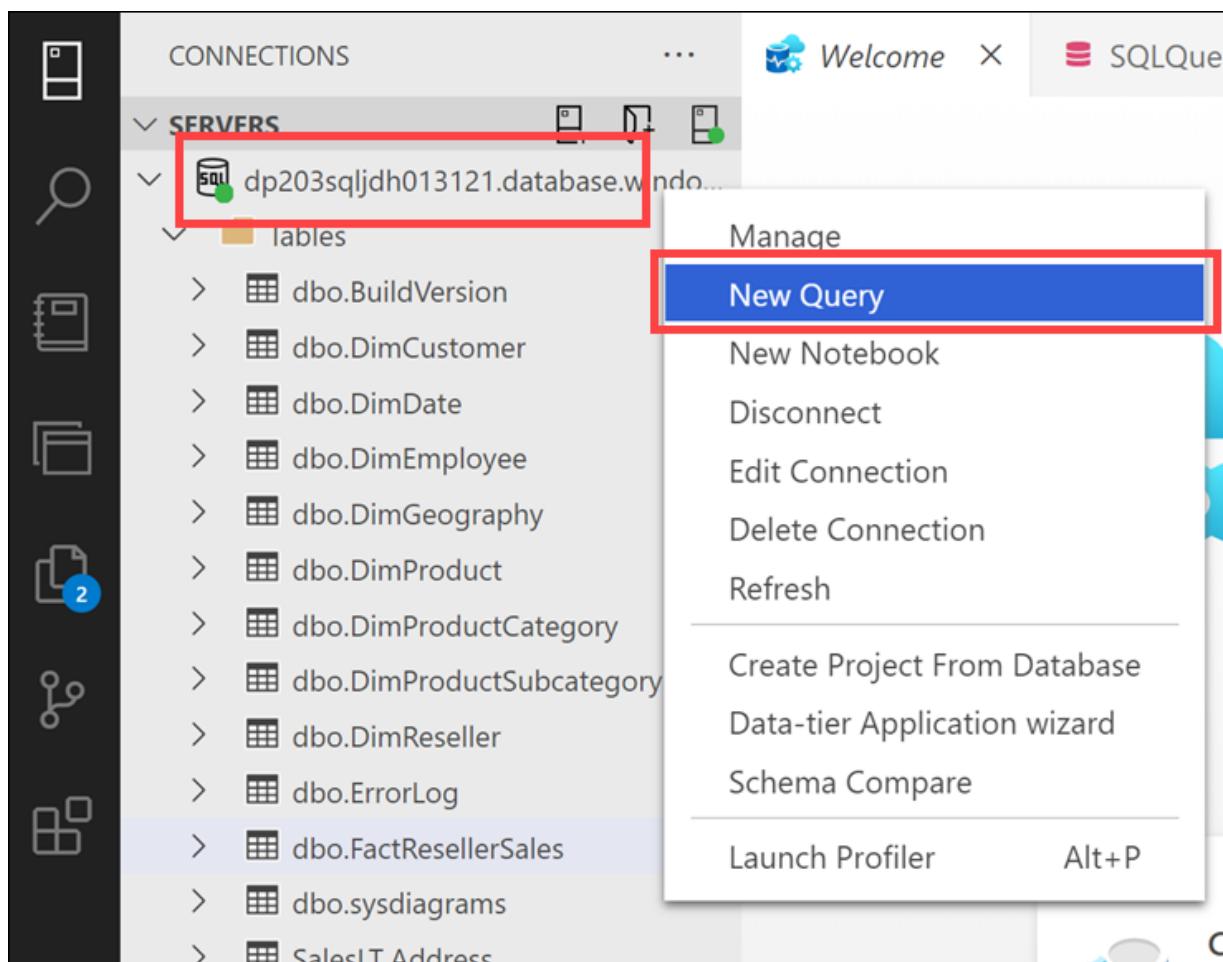
3. Paste the following in the query window, then select **Run** or hit F5 to execute the script and view the results:

```
SELECT * FROM DimCustomer
```

The screenshot shows the SQL Server Management Studio interface. In the top bar, there's a 'SQL script 1' tab, 'Run', 'Undo', 'Publish', 'Query plan', 'Connect to' dropdown set to 'SQLPool01', 'Use database' dropdown set to 'SQLPool01', and a refresh button. Below the top bar, a query window contains the command: '1 SELECT * FROM DimCustomer'. The results pane shows a table titled 'DimCustomer' with columns: CustomerID, Title, FirstName, MiddleName, LastName, Suffix, CompanyName, SalesPerson, EmailAddress, Phone, and Ins. The table data includes rows for customers like Danielle Johnson, Jean Jordan, David Lawrence, etc. At the bottom of the results pane, it says '00:00:01 Query executed successfully.'

4.7.5 Task 5: Update a source customer record

1. Open Azure Data Studio, or switch back to it if still open.
2. Select **Servers** in the left-hand menu, then right-click the SQL server you added at the beginning of the lab. Select **New Query**.



3. Paste the following into the query window to view the customer with a CustomerID of 10:

```
SELECT * FROM [SalesLT].[Customer] WHERE CustomerID = 10
```

- Select **Run** or hit F5 to execute the query.

```
1   SELECT * FROM [SalesLT].[Customer] WHERE CustomerID = 10
```

	CustomerID	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	CompanyName	SalesPerson	EmailAddress	Phone	PasswordHash
1	10	0	Ms.	Kathleen	M.	Garza	NULL	Rural Cycle Emporium	adventure-works\jose1	kathleen0@adventure-works.com	150-555-0127	Qa3aMCxNbVLGrc0b99K

The customer for Ms. Kathleen M. Garza is displayed. Let's change the customer's last name.

- Replace **and execute** the query with the following to update the customer's last name:

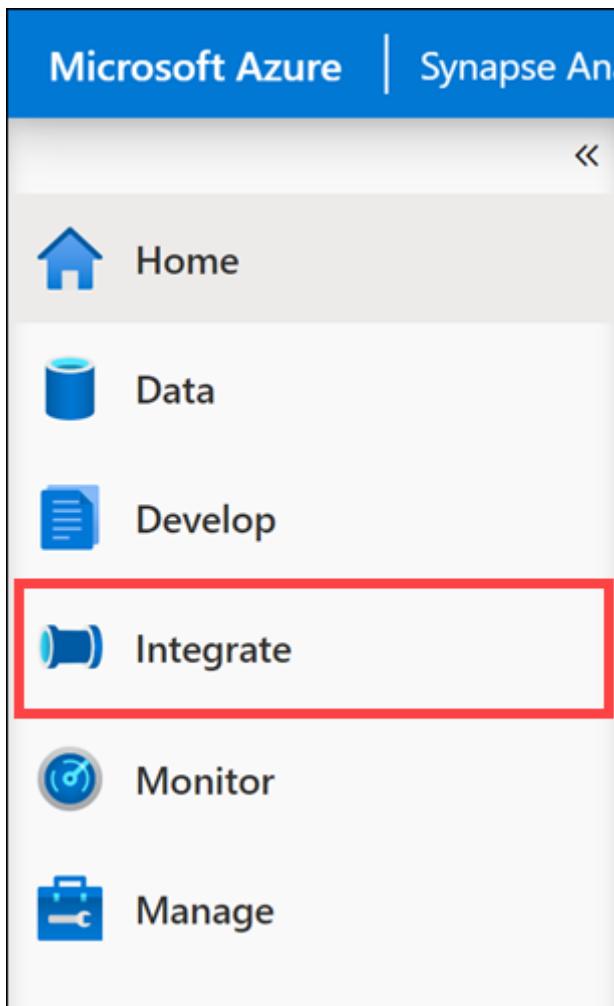
```
UPDATE [SalesLT].[Customer] SET LastName = 'Smith' WHERE CustomerID = 10
SELECT * FROM [SalesLT].[Customer] WHERE CustomerID = 10
```

```
1   UPDATE [SalesLT].[Customer] SET LastName = 'Smith' WHERE CustomerID = 10
2   SELECT * FROM [SalesLT].[Customer] WHERE CustomerID = 10
```

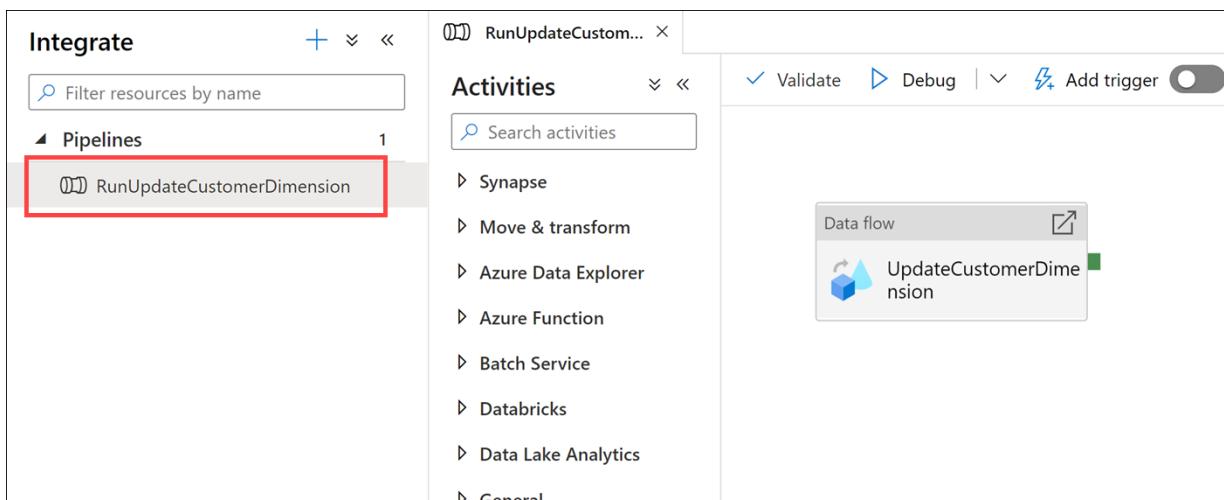
	CustomerID	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	CompanyName	SalesPerson
1	10	0	Ms.	Kathleen	M.	Smith	NULL	Rural Cycle Emporium	adventure-works\jose1

4.7.6 Task 6: Re-run mapping data flow

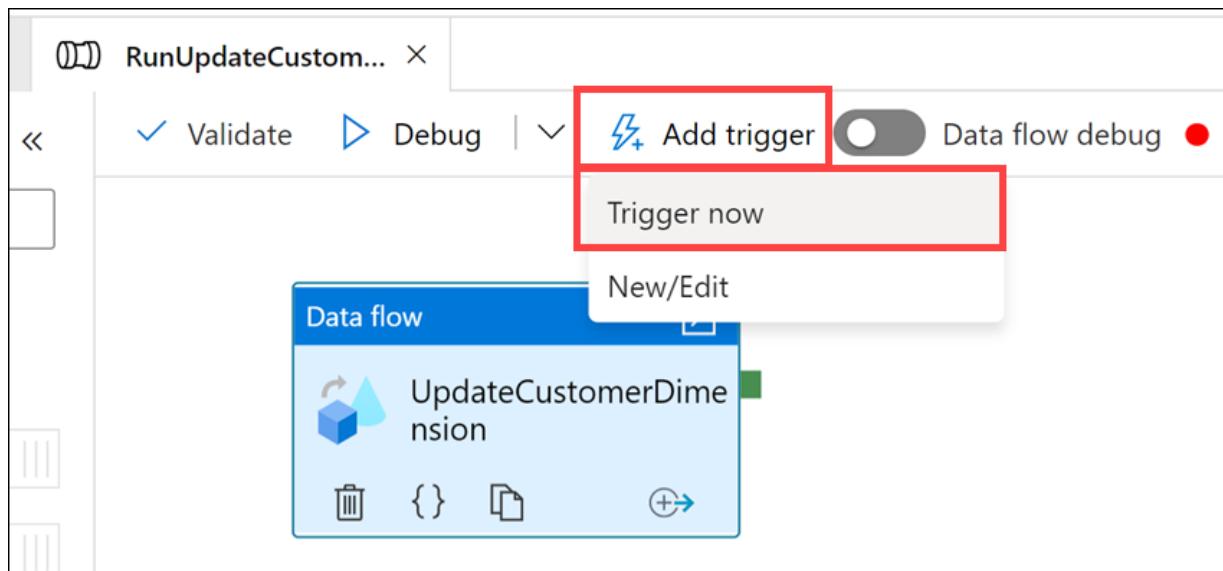
- Switch back to Synapse Studio.
- Navigate to the **Integrate** hub.



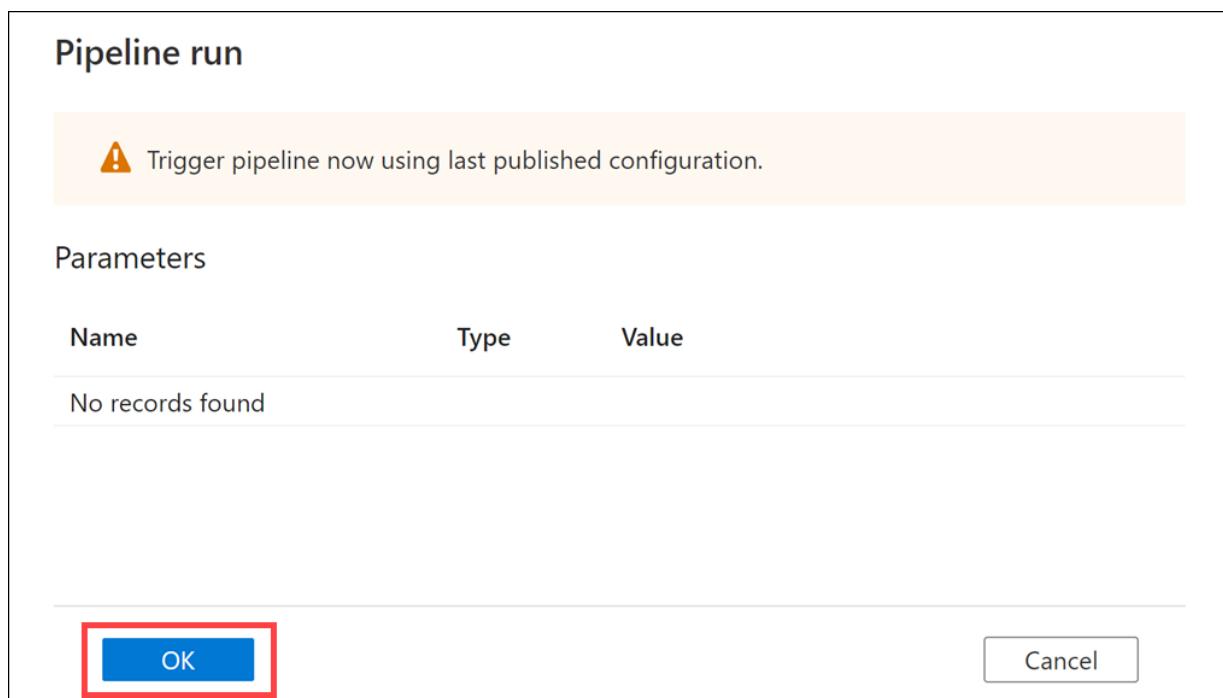
3. Select the **RunUpdateCustomerDimension** pipeline.



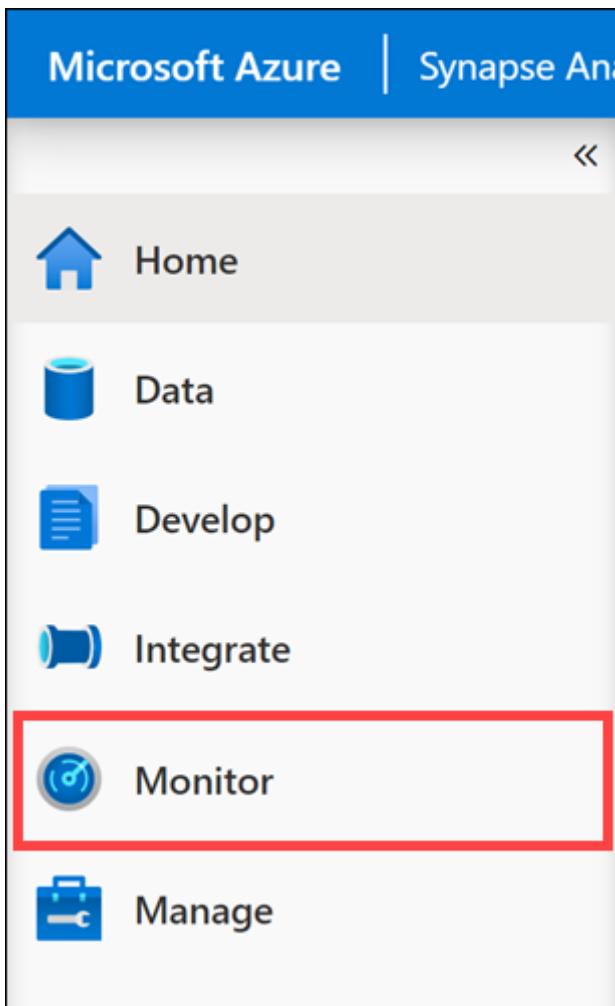
4. Select **Add trigger** above the pipeline canvas, then select **Trigger now**.



5. Select **OK** in the Pipeline run dialog to trigger the pipeline.



6. Navigate to the **Monitor** hub.

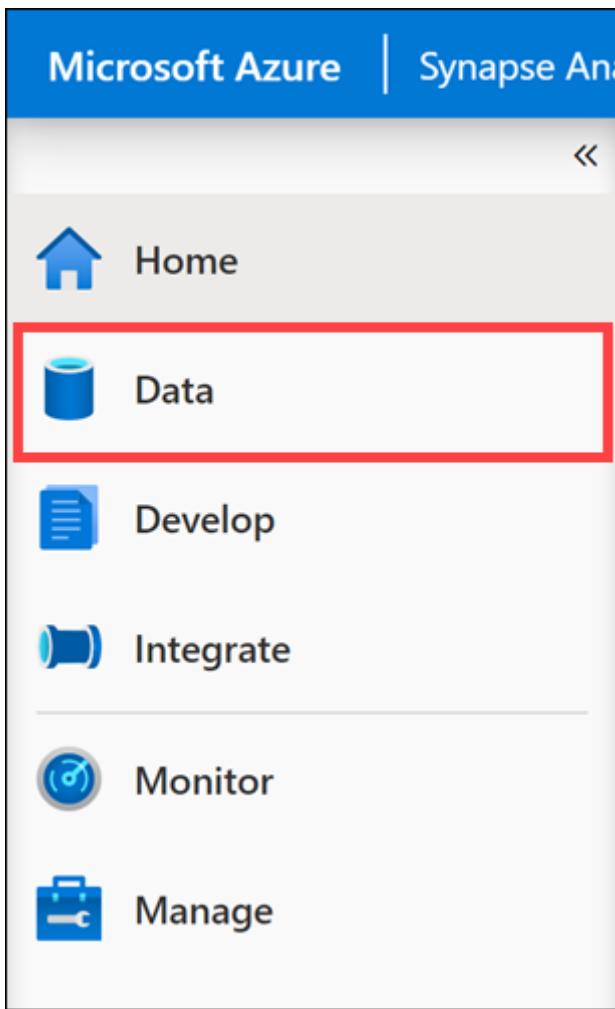


- Select **Pipeline runs** in the left-hand menu (1) and wait for the pipeline run to successfully complete (2). You may have to select **Refresh** (3) several times until the pipeline run completes.

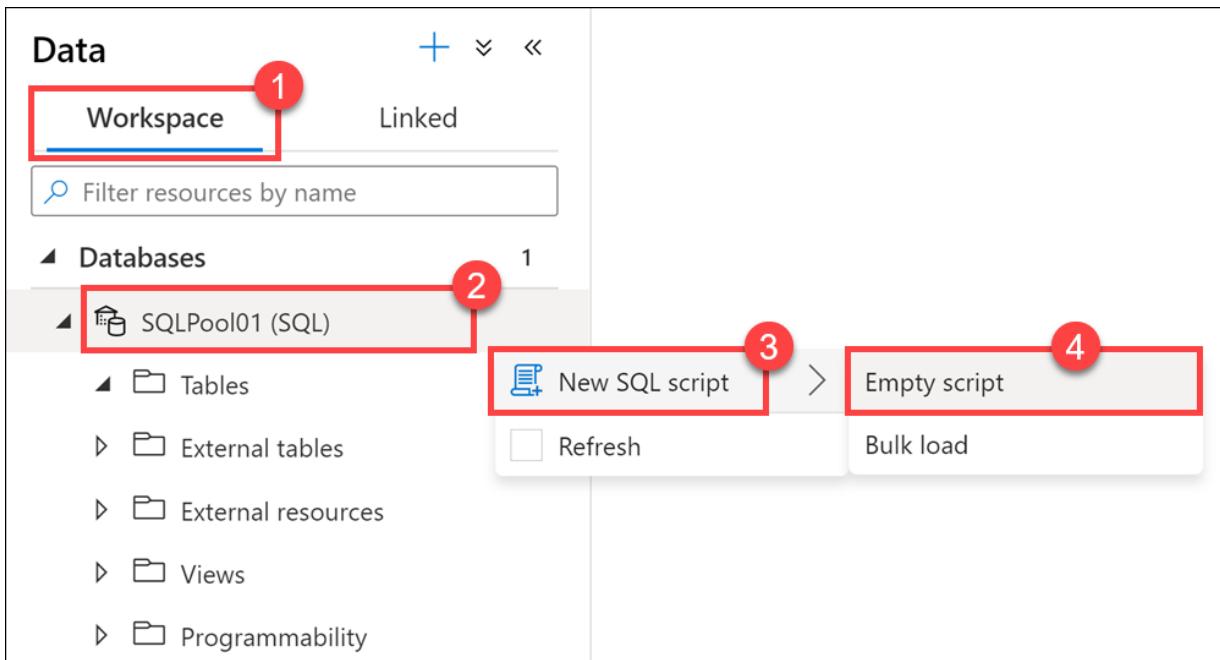
Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
RunUpdateCustomerDimension	2/1/21, 3:54:04 AM	2/1/21, 3:58:41 AM	00:04:37	Manual trigger	✓ Succeeded	Original
RunUpdateCustomerDimension	2/1/21, 3:16:32 AM	2/1/21, 3:21:50 AM	00:05:18	Manual trigger	✓ Succeeded	Original

4.7.7 Task 7: Verify record updated

1. Navigate to the **Data** hub.



2. Select the **Workspace** tab (1), expand Databases, then right-click on **SQLPool01** (2). Select **New SQL script** (3), then select **Empty script** (4).



3. Paste the following in the query window, then select **Run** or hit F5 to execute the script and view the results:

```
SELECT * FROM DimCustomer WHERE CustomerID = 10
```

The screenshot shows the Microsoft Synapse Studio interface. At the top, there's a toolbar with 'Run', 'Undo', 'Publish', 'Query plan', 'Connect to' (set to SQLPool01), 'Use database' (set to SQLPool01), and other standard icons. Below the toolbar is a query editor window containing the following SQL code:

```
1 SELECT * FROM DimCustomer WHERE CustomerID = 10
```

Below the query editor is a results pane. The 'Results' tab is selected, and the 'Table' view is chosen. A search bar labeled 'Search' is present. The results table has columns: CustomerID, Title, FirstName, MiddleName, LastName, Suffix, CompanyName, and SalesPerson. A single row is shown for CustomerID 10, with LastName 'Smith' highlighted by a red box.

CustomerID	Title	FirstName	MiddleName	LastName	Suffix	CompanyName	SalesPerson
10	Ms.	Kathleen	M.	Smith	NULL	Rural Cycle Em...	adventure-wor...

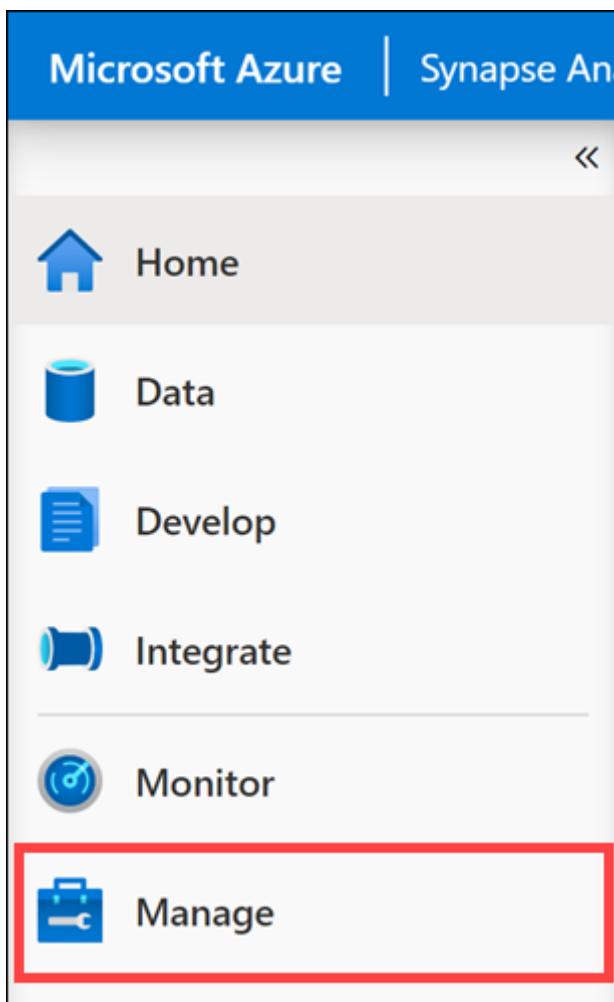
As we can see, the customer record successfully updated to modify the `LastName` value to match the source record.

4.8 Exercise 6: Cleanup

Complete these steps to free up resources you no longer need.

4.8.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

Analytics pools

- SQL pools** 1
- Apache Spark pools
- External connections
- Linked services
- Azure Purview (Preview)
- Integration
- Triggers
- Integration runtimes
- Security

SQL pools

The serverless SQL pool, Built-in, is immediately available for your workspace. Dedicated SQL pools can be configured.

+ New Refresh System-assigned managed identity

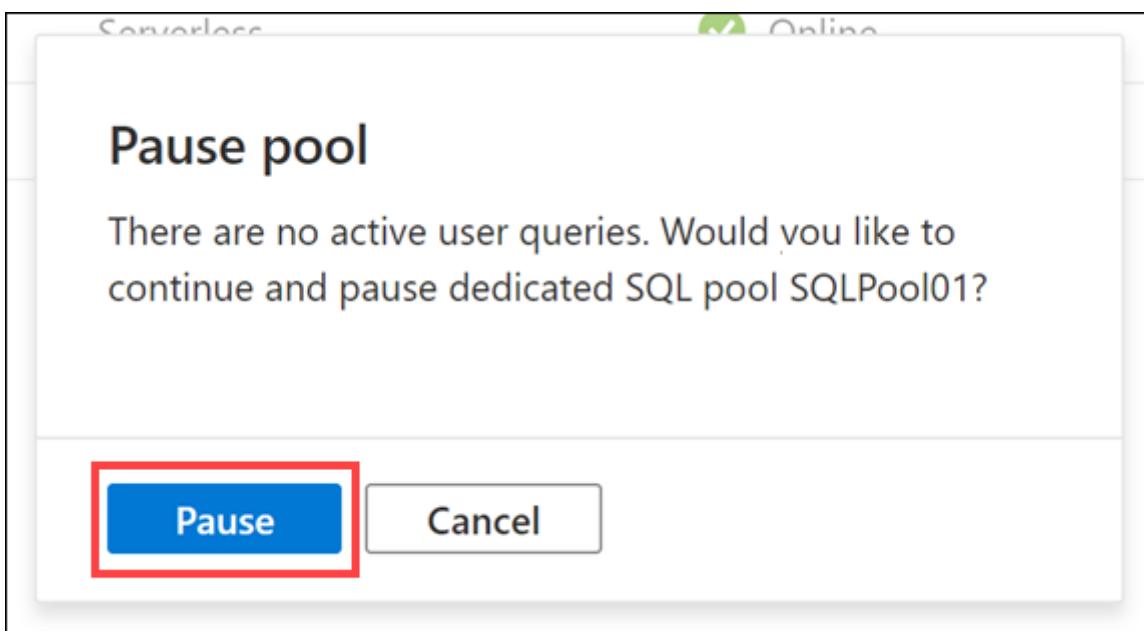
Filter by name

Showing 1-2 of 2 items (1 Serverless, 1 Dedicated)

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

Pause

- When prompted, select **Pause**.



5 Module 3 - Data engineering considerations for source files

In this lab, you will be directed by your instructor to work alone, or in groups for 20 minutes to read through the following information presented below. You will then answer the questions and present back to the classroom your findings based on the requirements.

5.1 Team whiteboard activity

Wide World Importers is ready to build a pipeline that copies their sales transactions from a table in an Oracle database to the data lake.

Requirements

- The pipeline that copies data will run on a scheduled basis, once per day.
- They would like to ingest this raw data applying the minimal amount of transformations to it.
- They want to ensure that their data lake always contains a copy of the original data, so that if their downstream processing has calculation or transformation errors, they can always re-compute from the original.
- Additionally, they want to avoid the file format prescribing what tools can be used to examine and process the data by making sure that the file format selected can be used by the broadest possible range of industry standard tools.
- The folder structure needs to be performant for typical exploratory and analytic queries for this type of data.

Example of the data

WWI has provided the following example of the data to you. You can assume they have hand selected rows that are *most* representative of the data.

SaleKey	CityKey	CustomerKey	BillToCustomerKey	StockItemKey	DeliveryDateKey	SalespersonKey	WWIIIn
294018	98706	0	0	25	2012-01-04	156	57894
294019	98706	0	0	216	2012-01-04	156	57894
294020	98706	0	0	168	2012-01-04	156	57894
294021	98706	0	0	100	2012-01-04	156	57894

5.2 Whiteboard

Open your whiteboard for the event, and in the area for Activity 1 provide your answers to the following challenges.

The following challenges are already present within the whiteboard template provided.

Challenges

1. What file format should they use for the raw data? Why did you recommend this file format, provide at least two reasons?
2. What specific settings should WWI use in configuring the way the dataset is serialized to disk (pay particular attention to the **Description** field)? Why did you suggest these?
3. Diagram the folder structure you would recommend they use in the hierarchical filesystem. Be sure to indicate filesystem, folders and files and describe how each layer (filesystem, folder and file) derives its name.
4. How does your folder structure support query performance for typical exploratory and analytic queries for this type of data?
5. WWI consider this data confidential, because if it were to fall into the hands of the competition it would cause irreparable harm. Diagram how you would deploy the data lake and secure access to the data lake endpoint? Be sure to illustrate how data flows between your Azure Synapse Analytics Workspace and the data lake and explain why this addresses WWI's requirements. Use the icons provided in the palette to diagram your solution.
6. WWI wants to enforce that any kind of modifications to sales data can happen in the current year only, while allowing all authorized users to query the entirety of data. Regarding the folder structure you previously recommended to WWI, how would accomplish this using RBAC and ACLs? Explain what actions would need to be taken at the start and end of the year 2020. Diagram your security groups, built-in roles and access permissions using the provided palette.

6 Module 4 - Run interactive queries using serverless SQL pools

In this module, students will learn how to work with files stored in the data lake and external file sources, through T-SQL statements executed by a serverless SQL pool in Azure Synapse Analytics. Students will query Parquet files stored in a data lake, as well as CSV files stored in an external data store. Next, they will create Azure Active Directory security groups and enforce access to files in the data lake through Role-Based Access Control (RBAC) and Access Control Lists (ACLs).

In this module, the student will be able to:

- Query Parquet data with serverless SQL pools
- Create external tables for Parquet and CSV files
- Create views with serverless SQL pools
- Secure access to data in a data lake when using serverless SQL pools
- Configure data lake security using Role-Based Access Control (RBAC) and Access Control Lists (ACLs)

6.1 Lab details

- [Module 4 - Run interactive queries using serverless SQL pools](#)

- [Lab details](#)
- [Lab setup and pre-requisites](#)
- [Exercise 1: Querying a Data Lake Store using serverless SQL pools in Azure Synapse Analytics](#)
 - * [Task 1: Query sales Parquet data with serverless SQL pools](#)
 - * [Task 2: Create an external table for 2019 sales data](#)
 - * [Task 3: Create an external table for CSV files](#)
 - * [Task 4: Create a view with a serverless SQL pool](#)
- [Exercise 2: Securing access to data through using a serverless SQL pool in Azure Synapse Analytics](#)
 - * [Task 1: Create Azure Active Directory security groups](#)
 - * [Task 2: Add group members](#)
 - * [Task 3: Configure data lake security - Role-Based Access Control \(RBAC\)](#)
 - * [Task 4: Configure data lake security - Access Control Lists \(ACLs\)](#)
 - * [Task 5: Test permissions](#)

Tailwind Trader's Data Engineers want a way to explore the data lake, transform and prepare data, and simplify their data transformation pipelines. In addition, they want their Data Analysts to explore data in the lake and Spark external tables created by Data Scientists or Data Engineers, using familiar T-SQL language or their favorite tools, which can connect to SQL endpoints.

6.2 Lab setup and pre-requisites

Note: Only complete the [Lab setup and pre-requisites](#) steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 1.

You must have permissions to create new Azure Active Directory security groups and assign members to them.

Complete the lab setup instructions for this module.

Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

6.3 Exercise 1: Querying a Data Lake Store using serverless SQL pools in Azure Synapse Analytics

Understanding data through data exploration is one of the core challenges faced today by data engineers and data scientists as well. Depending on the underlying structure of the data as well as the specific requirements of the exploration process, different data processing engines will offer varying degrees of performance, complexity, and flexibility.

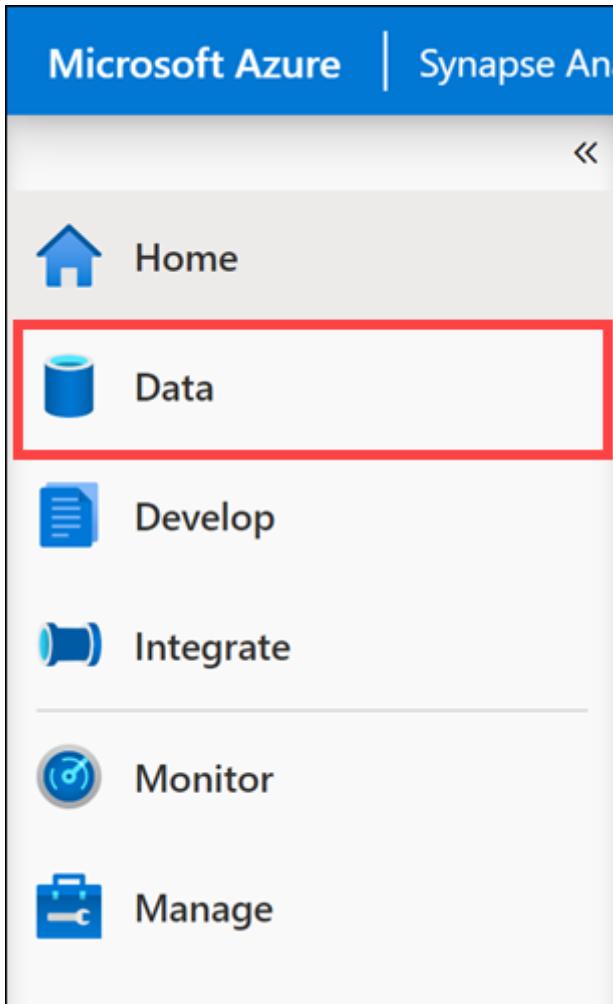
In Azure Synapse Analytics, you can use either SQL, Apache Spark for Synapse, or both. Which service you use mostly depends on your personal preference and expertise. When conducting data engineering tasks, both options can be equally valid in many cases. However, there are certain situations where harnessing the power of Apache Spark can help you overcome problems with the source data. This is because in a Synapse Notebook, you can import from a large number of free libraries that add functionality to your environment when working with data. There are other situations where it is much more convenient and faster using serverless SQL pool to explore the data, or to expose data in the data lake through a SQL view that can be accessed from external tools, like Power BI.

In this exercise, you will explore the data lake using both options.

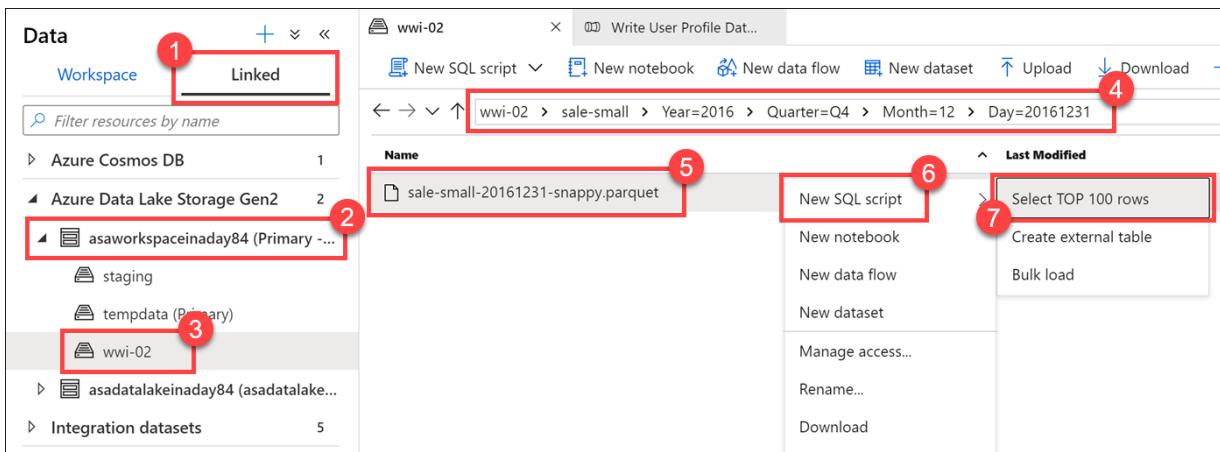
6.3.1 Task 1: Query sales Parquet data with serverless SQL pools

When you query Parquet files using serverless SQL pools, you can explore the data with T-SQL syntax.

1. Open Synapse Studio (<https://web.azuresynapse.net/>), and then navigate to the **Data** hub.



2. Select the **Linked** tab (1) and expand **Azure Data Lake Storage Gen2**. Expand the **asaworkspaceXX** primary ADLS Gen2 account (2) and select the **wwi-02** container (3). Navigate to the **sale-small/Year=2016/Quarter=Q4/Month=12/Day=20161231** folder (4). Right-click on the **sale-small-20161231-snappy.parquet** file (5), select **New SQL script** (6), then **Select TOP 100 rows** (7).



3. Ensure **Built-in** is selected (1) in the **Connect to** dropdown list above the query window, then run the query (2). Data is loaded by the serverless SQL endpoint and processed as if was coming from any regular relational database.

```

1 SELECT
2     TOP 100 *
3     FROM
4         OPENROWSET(
5             BULK 'https://asadatalakeinaday84.dfs.core.windows.net/wwi-02/sale-small/Year=2016/Quarter=Q4',
6             FORMAT='PARQUET'
7         ) AS [result]
8

```

The cell output shows the query results from the Parquet file.

TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount	TransactionDate	ProfitAmount	Hour
beb0650-b54f...	3	3208	4	33.8600000000...	135.4400000000...	20161231	43.2000000000...	6
beb0650-b54f...	3	1820	4	28.6500000000...	114.6000000000...	20161231	35.6800000000...	6
beb0650-b54f...	3	3372	2	23.0400000000...	46.0800000000...	20161231	13.0600000000...	6
beb0650-b54f...	3	1373	3	34.6500000000...	103.9500000000...	20161231	31.0200000000...	6
beb0650-b54f...	3	2760	1	36.1300000000...	36.1300000000...	20161231	7.2500000000...	6
beb0650-b54f...	3	2253	4	26.8000000000...	107.2000000000...	20161231	28.0000000000...	6
beb0650-b54f...	3	3498	4	25.0800000000...	100.3200000000...	20161231	28.6800000000...	6
beb0650-b54f...	3	2358	3	23.2900000000...	69.8700000000...	20161231	24.9300000000...	6
beb0650-b54f...	3	198	4	30.8200000000...	123.2800000000...	20161231	41.8000000000...	6

4. Modify the SQL query to perform aggregates and grouping operations to better understand the data. Replace the query with the following, making sure that the file path in OPENROWSET matches the current file path:

```

SELECT
    TransactionDate, ProductId,
    CAST(SUM(ProfitAmount) AS decimal(18,2)) AS [(sum) Profit],
    CAST(AVG(ProfitAmount) AS decimal(18,2)) AS [(avg) Profit],
    SUM(Quantity) AS [(sum) Quantity]
FROM
    OPENROWSET(
        BULK 'https://asadatalakeSUFFIX.dfs.core.windows.net/wwi-02/sale-small/Year=2016/Quarter=Q4',
        FORMAT='PARQUET'
    ) AS [r] GROUP BY r.TransactionDate, r.ProductId;

```

```

1  SELECT
2      TransactionDate, ProductId,
3      CAST(SUM(ProfitAmount) AS decimal(18,2)) AS [(sum) Profit],
4      CAST(AVG(ProfitAmount) AS decimal(18,2)) AS [(avg) Profit],
5      SUM(Quantity) AS [(sum) Quantity]
6  FROM
7      OPENROWSET(
8          BULK 'https://asadatalakeinaday84.dfs.core.windows.net/wwi-02/sale-small/Year=2016/Quarter=Q4/Month=12/Day=20161231/sale-sm'
9          FORMAT='PARQUET'
10     ) AS [r] GROUP BY r.TransactionDate, r.ProductId;
11

```

Results Messages

View Table Chart Export results ▾

TransactionDate	ProductId	(Sum) Profit	(Avg) Profit	(Sum) Quantity
20161231	3	2770.53	5.95	1169
20161231	5	10799.35	19.96	1355
20161231	15	8053.50	16.34	1239
20161231	21	11670.75	23.63	1197
20161231	22	8461.56	16.24	1284
20161231	33	9165.84	17.80	1266
20161231	41	14511.84	29.14	1234
20161231	42	11750.50	25.40	1206

5. Let's move on from this single file from 2016 and transition to a newer data set. We want to figure out how many records are contained within the Parquet files for all 2019 data. This information is important for planning how we optimize for importing the data into Azure Synapse Analytics. To do this, we'll replace the query with the following (be sure to update the name of your data lake in the BULK statement, by replacing [asadatalakeSUFFIX]):

```

SELECT
    COUNT(*)
FROM
    OPENROWSET(
        BULK 'https://asadatalakeSUFFIX.dfs.core.windows.net/wwi-02/sale-small/Year=2019/*/*/*/*',
        FORMAT='PARQUET'
    ) AS [r];

```

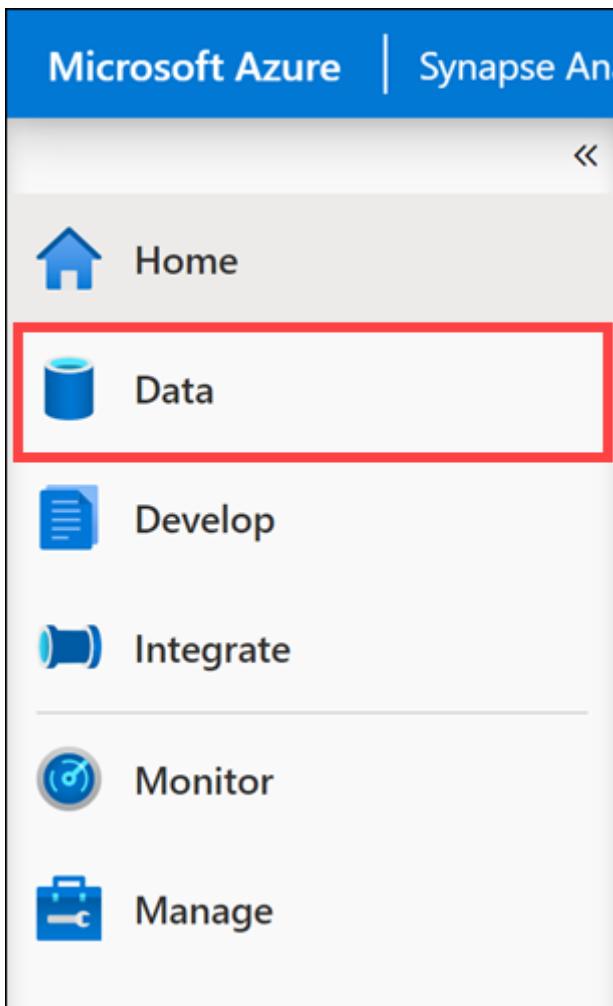
Notice how we updated the path to include all Parquet files in all subfolders of `sale-small/Year=2019`.

The output should be **339507246** records.

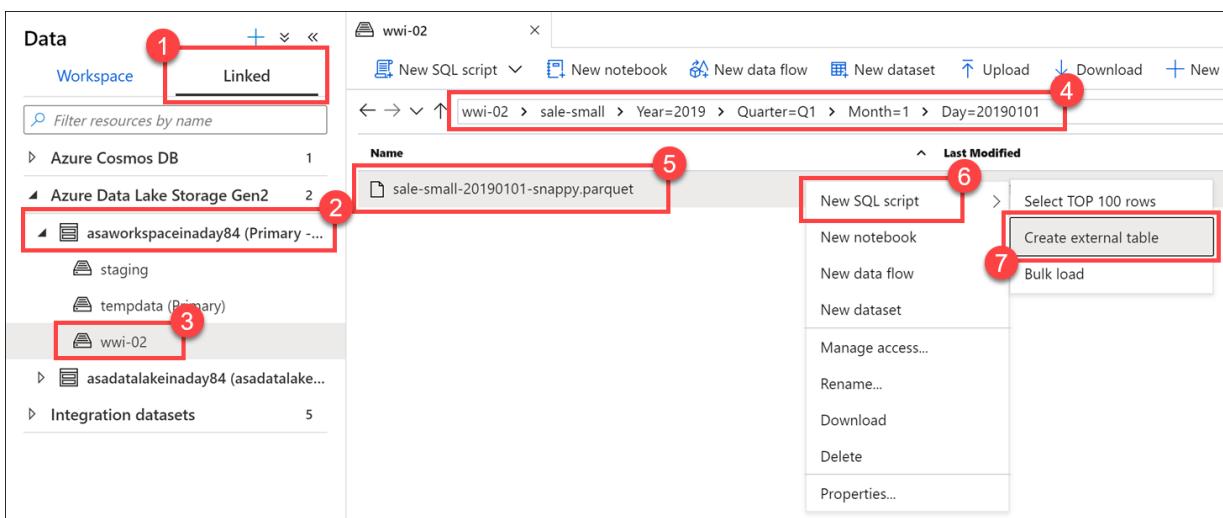
6.3.2 Task 2: Create an external table for 2019 sales data

Rather than creating a script with OPENROWSET and a path to the root 2019 folder every time we want to query the Parquet files, we can create an external table.

1. In Synapse Studio, navigate to the **Data** hub.



2. Select the **Linked** tab (1) and expand **Azure Data Lake Storage Gen2**. Expand the **asaworkspaceXX** primary ADLS Gen2 account (2) and select the **wwi-02** container (3). Navigate to the **sale-small/Year=2019/Quarter=Q1/Month=1/Day=20190101** folder (4). Right-click on the **sale-small-20190101-snappy.parquet** file (5), select **New SQL script** (6), then **Create external table** (7).



3. Make sure **Built-in** is selected for the **SQL pool** (1). Under **Select a database**, select **+ New** and enter **demo** (2). For **External table name**, enter **All2019Sales** (3). Under **Create external table**, select **Using SQL script** (4), then select **Create** (5).

Create external table

 sale-small-20190101-snappy.parque ...

External tables provide a convenient way to persist the schema of data residing in your data lake which can be reused for future adhoc analytics. [Learn more](#)

Select SQL pool * (i)

Built-in 1 

Select a database * (i)

demo 2 

External table name * (i)

All2019Sales 3

Create external table *

Automatically

Using SQL script 4

- i** This will include the create external table definition and the SELECT Top 100 in your SQL script. You will be required to run the SQL script to create the external table

5

Create

Cancel

Note: Make sure the script is connected to the serverless SQL pool (Built-in) (1) and the database is set to demo (2).



The generated script contains the following components:

- 1) The script begins with creating the `SynapseParquetFormat` external file format with a `FORMAT_TYPE` of `PARQUET`.
- 2) Next, the external data source is created, pointing to the `wwi-02` container of the data lake storage account.
- 3) The `CREATE EXTERNAL TABLE WITH` statement specifies the file location and refers to the new external file format and data source created above.
- 4) Finally, we select the top 100 results from the `2019Sales` external table.

```

1 IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormat') 1
2   CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
3     WITH ( FORMAT_TYPE = PARQUET)
4 GO
5
6 IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'wwi-02_asadatalakeinaday84_dfs_core_windows_net')
7   CREATE EXTERNAL DATA SOURCE [wwi-02_asadatalakeinaday84_dfs_core_windows_net]
8     WITH (
9       LOCATION    = 'https://asadatalakeinaday84.dfs.core.windows.net/wwi-02',
10      )
11 Go
12
13 CREATE EXTERNAL TABLE All2019Sales (
14   [TransactionId] varchar(8000),
15   [CustomerId] int,
16   [ProductId] smallint,
17   [Quantity] smallint,
18   [Price] numeric(38,18),
19   [TotalAmount] numeric(38,18),
20   [TransactionDate] int,
21   [ProfitAmount] numeric(38,18),
22   [Hour] smallint,
23   [Minute] smallint,
24   [StoreId] smallint
25   )
26   WITH (
27     LOCATION = 'sale-small/Year=2019/Quarter=Q1/Month=1/Day=20190101/sale-small-20190101-snappy.parquet',
28     DATA_SOURCE = [wwi-02_asadatalakeinaday84_dfs_core_windows_net],
29     FILE_FORMAT = [SynapseParquetFormat]
30   )
31 GO
32
33 SELECT TOP 100 * FROM All2019Sales 4
34 GO

```

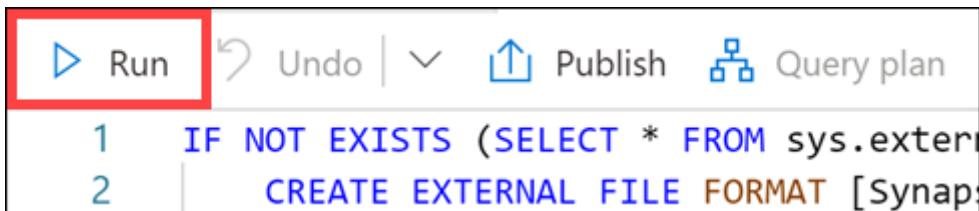
4. Replace the `LOCATION` value in the `CREATE EXTERNAL TABLE` statement with `sale-small/Year=2019/*/*/*/*.parquet`

```

13  CREATE EXTERNAL TABLE All2019Sales (
14      [TransactionId] varchar(8000),
15      [CustomerId] int,
16      [ProductId] smallint,
17      [Quantity] smallint,
18      [Price] numeric(38,18),
19      [TotalAmount] numeric(38,18),
20      [TransactionDate] int,
21      [ProfitAmount] numeric(38,18),
22      [Hour] smallint,
23      [Minute] smallint,
24      [StoreId] smallint
25  )
26  WITH (
27      LOCATION = 'sale-small/Year=2019/*/*/*/*.parquet',
28      DATA_SOURCE = [ww1-02_asadatalakeinaday84_dts_core_windows_net],
29      FILE_FORMAT = [SynapseParquetFormat]
30  )
31 GO
32
33 SELECT TOP 100 * FROM All2019Sales
34 GO

```

5. Run the script.



After running the script, we can see the output of the SELECT query against the All2019Sales external table. This displays the first 100 records from the Parquet files located in the YEAR=2019 folder.

```

27     LOCATION = 'sale-small/Year=2019/*/*/*.parquet',
28     DATA_SOURCE = [wwi-02_asadatalakeinaday84_dfs_core_windows_net],
29     FILE_FORMAT = [SynapseParquetFormat]
30   )
31 GO
32
33 SELECT TOP 100 * FROM All2019Sales
34 GO
35
36

```

Results Messages

Select Query 4 View Table Chart Export results

Search

TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount	TransactionDate	ProfitAmount	Hour
5fc634ab-e910...	1	2965	2	35.4200000000...	70.8400000000...	20190103	20.7000000000...	9
5fc634ab-e910...	1	3986	3	34.6400000000...	103.9200000000...	20190103	31.5900000000...	9
5fc634ab-e910...	1	4648	1	23.6000000000...	23.6000000000...	20190103	8.2500000000...	9
5fc634ab-e910...	1	1359	3	29.8200000000...	89.4600000000...	20190103	29.6700000000...	9
5fc634ab-e910...	1	2726	4	33.2200000000...	132.8800000000...	20190103	38.2000000000...	9
5fc634ab-e910...	1	384	4	31.6100000000...	126.4400000000...	20190103	29.3200000000...	9
5fc634ab-e910...	1	413	4	31.5900000000...	126.3600000000...	20190103	40.0000000000...	9
5fc634ab-e910...	1	2615	1	39.4000000000...	39.4000000000...	20190103	11.3700000000...	9
5fc634ab-e910...	1	2615	7	39.4000000000...	27.9800000000...	20190103	22.7400000000...	9

6.3.3 Task 3: Create an external table for CSV files

Tailwind Traders found an open data source for country population data that they want to use. They do not want to merely copy the data since it is regularly updated with projected populations in future years.

You decide to create an external table that connects to the external data source.

- Replace the SQL script with the following:

```

IF NOT EXISTS (SELECT * FROM sys.symmetric_keys) BEGIN
    declare @pasword nvarchar(400) = CAST(newid() as VARCHAR(400));
    EXEC('CREATE MASTER KEY ENCRYPTION BY PASSWORD = ''' + @pasword + '''')
END

CREATE DATABASE SCOPED CREDENTIAL [sqlondemand]
WITH IDENTITY='SHARED ACCESS SIGNATURE',
SECRET = 'sv=2018-03-28&ss=bf&srt=sco&sp=rl&st=2019-10-14T12%3A10%3A25Z&se=2061-12-31T12%3A10%3A00Z'
GO

-- Create external data source secured using credential
CREATE EXTERNAL DATA SOURCE SqlOnDemandDemo WITH (
    LOCATION = 'https://sqlondemandstorage.blob.core.windows.net',
    CREDENTIAL = sqlondemand
);
GO

CREATE EXTERNAL FILE FORMAT QuotedCsvWithHeader
WITH (
    FORMAT_TYPE = DELIMITEDTEXT,
    FORMAT_OPTIONS (
        FIELD_TERMINATOR = ',',
        STRING_DELIMITER = '',
        FIRST_ROW = 2
    )
);
GO

CREATE EXTERNAL TABLE [population]

```

```

(
    [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
    [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
    [year] smallint,
    [population] bigint
)
WITH (
    LOCATION = 'csv/population/population.csv',
    DATA_SOURCE = SqlOnDemandDemo,
    FILE_FORMAT = QuotedCsvWithHeader
);
GO

```

At the top of the script, we create a **MASTER KEY** with a random password (1). Next, we create a database-scoped credential for the containers in the external storage account (2), using a shared access signature (SAS) for delegated access. This credential is used when we create the **SqlOnDemandDemo** external data source (3) that points to the location of the external storage account that contains the population data:

```

1  IF NOT EXISTS (SELECT * FROM sys.symmetric_keys) BEGIN
2      declare @pasword nvarchar(400) = CAST(newid() as VARCHAR(400));
3      EXEC('CREATE MASTER KEY ENCRYPTION BY PASSWORD = ''' + @pasword + '''') ①
4  END
5
6  CREATE DATABASE SCOPED CREDENTIAL [sqlondemand] ②
7  WITH IDENTITY='SHARED ACCESS SIGNATURE'
8  SECRET = 'sv=2018-03-28&ss=bf&srt=sco&sp=r&st=2019-10-14T12%3A10%3A25Z&se=2061-12-31T12%3A10%3A00Z&sig=K1SU2ullCscyTS0An0n
9  GO
10
11 -- Create external data source secured using credential
12 CREATE EXTERNAL DATA SOURCE SqlOnDemandDemo WITH ( ③
13     LOCATION = 'https://sqlondemandstorage.blob.core.windows.net',
14     CREDENTIAL = sqlondemand
15 );
16 GO

```

Database-scoped credentials are used when any principal calls the OPENROWSET function with a DATA_SOURCE or selects data from an external table that doesn't access public files. The database scoped credential doesn't need to match the name of storage account because it will be explicitly used in the DATA SOURCE that defines the storage location.

In the next part of the script, we create an external file format called **QuotedCsvWithHeader**. Creating an external file format is a prerequisite for creating an External Table. By creating an External File Format, you specify the actual layout of the data referenced by an external table. Here we specify the CSV field terminator, string delimiter, and set the FIRST_ROW value to 2 since the file contains a header row:

```

17
18  CREATE EXTERNAL FILE FORMAT QuotedCsvWithHeader
19  WITH (
20      FORMAT_TYPE = DELIMITEDTEXT,
21      FORMAT_OPTIONS (
22          FIELD_TERMINATOR = ',',
23          STRING_DELIMITER = """",
24          FIRST_ROW = 2
25      )
26  );
27  GO

```

Finally, at the bottom of the script, we create an external table named **population**. The WITH clause specifies the relative location of the CSV file, points to the data source created above, as well as the **QuotedCsvWithHeader** file format:

```

28
29     CREATE EXTERNAL TABLE [population]
30     (
31         [country_code] VARCHAR (5) COLLATE Latin1_General_BIN2,
32         [country_name] VARCHAR (100) COLLATE Latin1_General_BIN2,
33         [year] smallint,
34         [population] bigint
35     )
36     WITH (
37         LOCATION = 'csv/population/population.csv',
38         DATA_SOURCE = SqlOnDemandDemo,
39         FILE_FORMAT = QuotedCsvWithHeader
40     );
41     GO

```

2. Run the script.



Please note that there are no data results for this query.

3. Replace the SQL script with the following to select from the population external table, filtered by 2019 data where the population is greater than 100 million:

```

SELECT [country_code]
    ,[country_name]
    ,[year]
    ,[population]
FROM [dbo].[population]
WHERE [year] = 2019 and population > 100000000

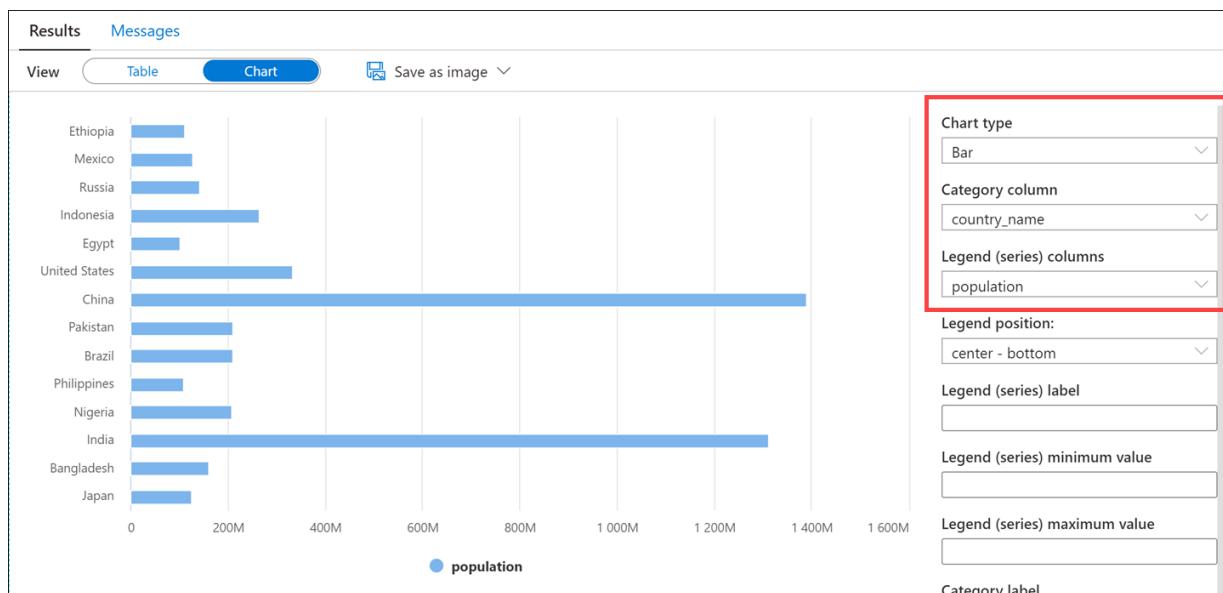
```

4. Run the script.



5. In the query results, select the **Chart** view, then configure it as follows:

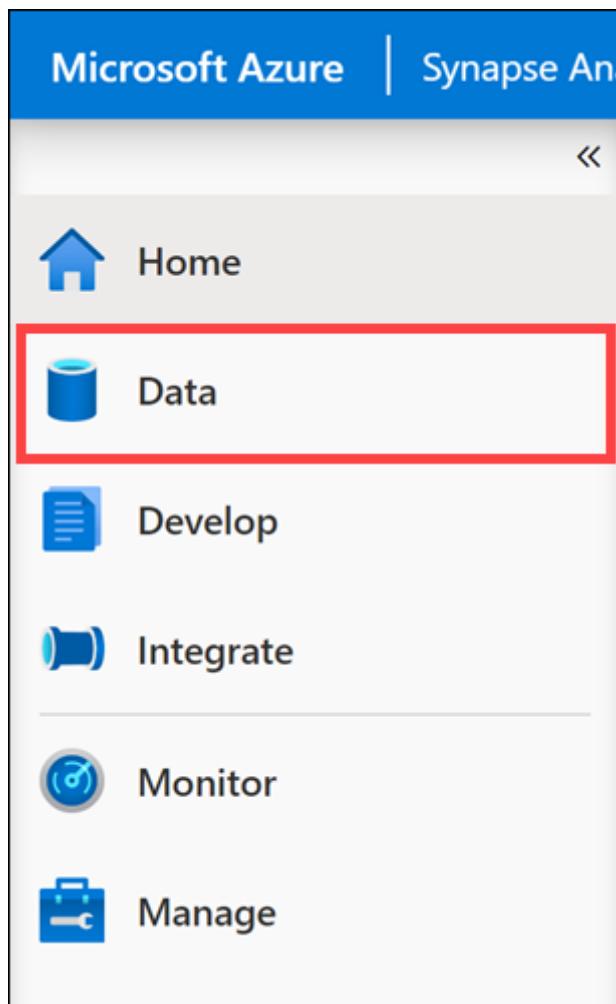
- **Chart type:** Select Bar.
- **Category column:** Select country_name.
- **Legend (series) columns:** Select population.
- **Legend position:** Select center - bottom.



6.3.4 Task 4: Create a view with a serverless SQL pool

Let's create a view to wrap a SQL query. Views allow you to reuse queries and are needed if you want to use tools, such as Power BI, in conjunction with serverless SQL pools.

1. In Synapse Studio, navigate to the **Data** hub.



2. Select the **Linked** tab (1) and expand **Azure Data Lake Storage Gen2**. Expand the **asaworkspaceXX** primary ADLS Gen2 account (2) and select the **wwi-02** container (3). Navigate to the **customer-info** folder (4). Right-click on the **customerinfo.csv** file (5), select **New SQL script** (6), then **Select**

TOP 100 rows (7).

The screenshot shows the Azure Data Studio interface. On the left, the 'Data' workspace is selected. A red box labeled 1 highlights the 'Linked' button. A red box labeled 2 highlights the 'asaworkspaceinaday84 (Primary ...)' dataset. A red box labeled 3 highlights the 'wwi-02' folder. In the center, a preview of 'customerinfo.csv' is shown with a red box labeled 4 highlighting the breadcrumb path 'wwi-02 > customer-info'. A red box labeled 5 highlights the file name 'customerinfo.csv'. A red box labeled 6 highlights the 'New SQL script' option in the context menu. A red box labeled 7 highlights the 'Select TOP 100 rows' option in the context menu.

3. Select **Run** to execute the script (1). Notice that the first row of the CSV file is the column header row (2).

The screenshot shows the Azure Data Studio query editor. A red box labeled 1 highlights the 'Run' button. The query code is as follows:

```

1 This is auto-generated code
2 SELECT
3   TOP 100 *
4   FROM
5     OPENROWSET(
6       BULK 'https://asadatalakeinaday84.dfs.core.windows.net/wwi-02/customer-info/customerinfo.csv',
7       FORMAT = 'CSV',
8       PARSER_VERSION='2.0'
9     ) AS [result]
10

```

The results pane shows a table with columns C1, C2, C3, C4, and C5. A red box labeled 2 highlights the header row. The data rows are as follows:

C1	C2	C3	C4	C5
UserName	Gender	Phone	Email	CreditCard
Jason.Green	Male	1-364-410-6690 x08562	Jason.Green@adventureworks.co...	9777-4287-8862-7386
Ben_Dickens	Male	551-530-7354 x0506	Ben_Dickens@adventureworks.c...	2992-5468-4860-3593
Alton_Vandervort	Male	811-714-3148	Alton_Vandervort@adventurewo...	2235-1309-9262-3979
Theresa16	Female	826-960-1591 x58508	Theresa16@adventureworks.com	4128-2405-5087-4621
Robert_Jerde95	Male	(888) 258-7303 x20068	Robert_Jerde95@adventurework...	4347-5516-7396-9369
Iana_Schmeler	Female	386-275-8812 v4018	Iana.Schmeler@adventureworks...	2146-3328-9686-9169

4. Update the script with the following and **make sure you replace YOUR_DATA_LAKE_NAME** (1) (your primary data lake storage account) in the OPENROWSET BULK path with the value in the in the previous select statement. Set the **Use database** value to **demo** (2) (use the refresh button to the right if needed):

```

CREATE VIEW CustomerInfo AS
  SELECT *
  FROM OPENROWSET(
    BULK 'https://YOUR_DATA_LAKE_NAME.dfs.core.windows.net/wwi-02/customer-info/customerinfo.csv',
    FORMAT = 'CSV',
    PARSER_VERSION='2.0',
    FIRSTROW=2
  )
  WITH (
    [UserName] VARCHAR (50),
    [Gender] VARCHAR (10),
    [Phone] VARCHAR (50),
    [Email] VARCHAR (100),
  )

```

```

    [CreditCard] VARCHAR (50)
) AS [r];
GO

```

```

SELECT * FROM CustomerInfo;
GO

```

```

1 CREATE VIEW CustomerInfo AS
2 |     SELECT *
3 |     FROM OPENROWSET(
4 |         BULK 'https://asadatalakeinaday84.dfs.core.windows.net/wwi-02/customer-info/customerinfo.csv',
5 |         FORMAT = 'CSV',
6 |         PARSER_VERSION='2.0',
7 |         FIRSTROW=2
8 |     )
9 |     WITH (
10 |         UserName1 VARCHAR (50)

```

5. Select **Run** to execute the script.

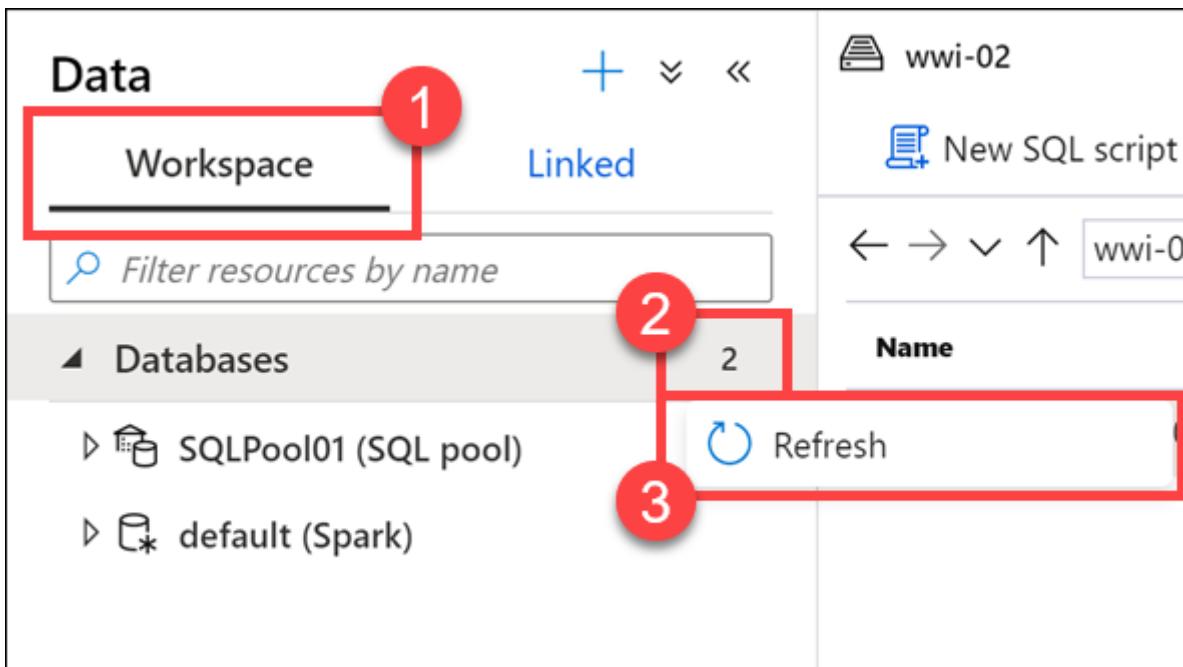


We just created the view to wrap the SQL query that selects data from the CSV file, then selected rows from the view:

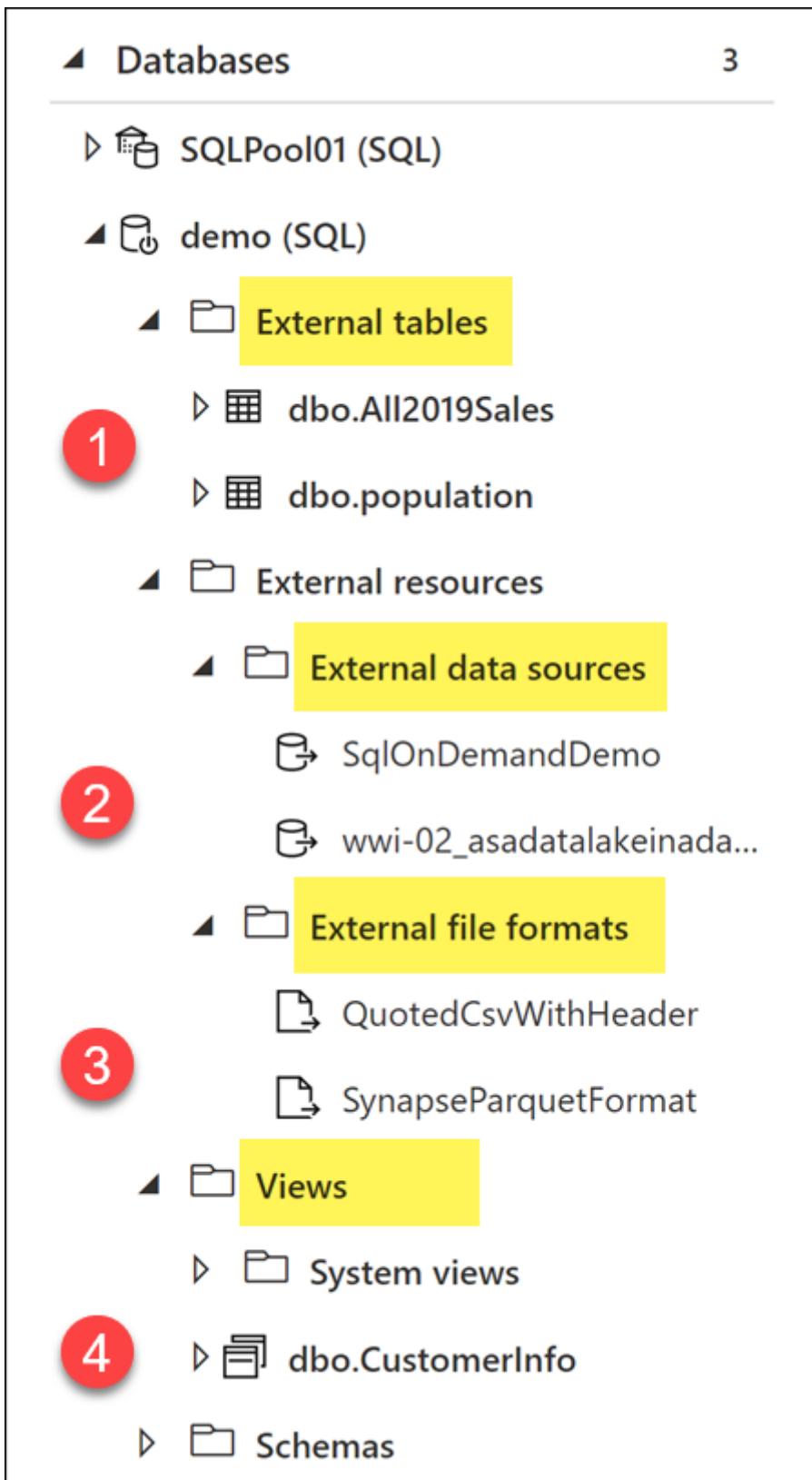
UserName	Gender	Phone	Email	CreditCard
Jason.Green	Male	1-364-410-6690 x08562	Jason.Green@adventureworks.co...	9777-4287-8862-7386
Ben_Dickens	Male	551-530-7354 x0506	Ben_Dickens@adventureworks.c...	2992-5468-4860-3593
Alton_Vandervort	Male	811-714-3148	Alton_Vandervort@adventurewo...	2235-1309-9262-3979
Theresa16	Female	826-960-1591 x58508	Theresa16@adventureworks.com	4128-2405-5087-4621
Robert_Jerde95	Male	(888) 258-7303 x20068	Robert_Jerde95@adventurework...	4347-5516-7396-9369
Jana.Schmeler	Female	386-275-8812 x4018	Jana.Schmeler@adventureworks....	2146-3328-9686-9169
Nathaniel_Botsford51	Male	(502) 878-3300	Nathaniel_Botsford51@adventur...	3779-7612-8624-1689

Notice that the first row no longer contains the column headers. This is because we used the **FIRSTROW=2** setting in the **OPENROWSET** statement when we created the view.

6. Within the **Data** hub, select the **Workspace** tab (1). Select the actions ellipses (...) to the right of the Databases group (2), then select **Refresh** (3).



7. Expand the demo SQL database.



The database contains the following objects that we created in our earlier steps:

- 1) **External tables:** All2019Sales and population.
- 2) **External data sources:** SqlOnDemandDemo and wwi-02_asadatalakeinadayXXX_dfs_core_windows_net.
- 3) **External file formats:** QuotedCsvWithHeader and SynapseParquetFormat.
- 4) **Views:** CustomerInfo.

6.4 Exercise 2: Securing access to data through using a serverless SQL pool in Azure Synapse Analytics

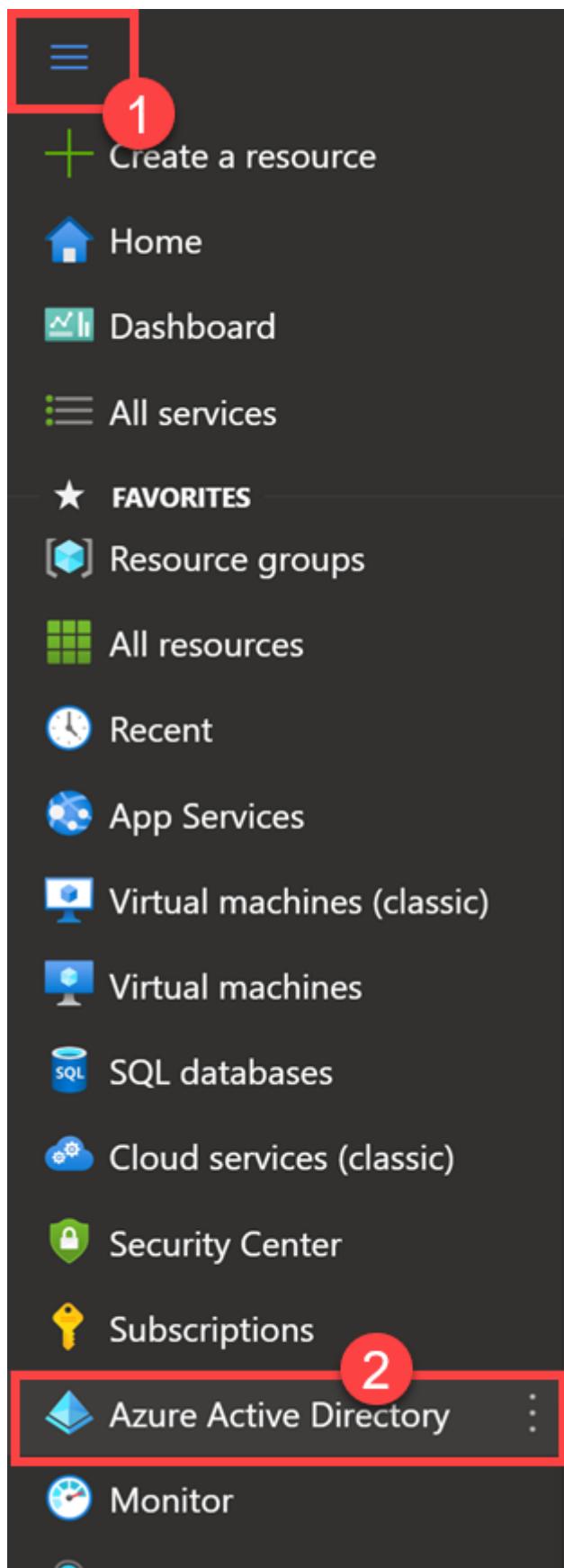
Tailwind Traders wants to enforce that any kind of modifications to sales data can happen in the current year only, while allowing all authorized users to query the entirety of data. They have a small group of admins who can modify historic data if needed.

- Tailwind Traders should create a security group in AAD, for example called `tailwind-history-owners`, with the intent that all users who belong to this group will have permissions to modify data from previous years.
- The `tailwind-history-owners` security group needs to be assigned to the Azure Storage built-in RBAC role `Storage Blob Data Owner` for the Azure Storage account containing the data lake. This allows AAD user and service principals that are added to this role to have the ability to modify all data from previous years.
- They need to add the user security principals who will have have permissions to modify all historical data to the `tailwind-history-owners` security group.
- Tailwind Traders should create another security group in AAD, for example called `tailwind-readers`, with the intent that all users who belong to this group will have permissions to read all contents of the file system (`prod` in this case), including all historical data.
- The `tailwind-readers` security group needs to be assigned to the Azure Storage built-in RBAC role `Storage Blob Data Reader` for the Azure Storage account containing the data lake. This allows AAD user and service principals that are added to this security group to have the ability to read all data in the file system, but not to modify it.
- Tailwind Traders should create another security group in AAD, for example called `tailwind-2020-writers`, with the intent that all users who belong to this group will have permissions to modify data only from the year 2020.
- They would create a another security group, for example called `tailwind-current-writers`, with the intent that only security groups would be added to this group. This group will have permissions to modify data only from the current year, set using ACLs.
- They need to add the `tailwind-readers` security group to the `tailwind-current-writers` security group.
- At the start of the year 2020, Tailwind Traders would add `tailwind-current-writers` to the `tailwind-2020-writers` security group.
- At the start of the year 2020, on the 2020 folder, Tailwind Traders would set the read, write and execute ACL permissions for the `tailwind-2020-writers` security group.
- At the start of the year 2021, to revoke write access to the 2020 data they would remove the `tailwind-current-writers` security group from the `tailwind-2020-writers` group. Members of `tailwind-readers` would continue to be able to read the contents of the file system because they have been granted read and execute (list) permissions not by the ACLs but by the RBAC built in role at the level of the file system.
- This approach takes into account that current changes to ACLs do not inherit permissions, so removing the write permission would require writing code that traverses all of its content removing the permission at each folder and file object.
- This approach is relatively fast. RBAC role assignments may take up to five minutes to propagate, regardless of the volume of data being secured.

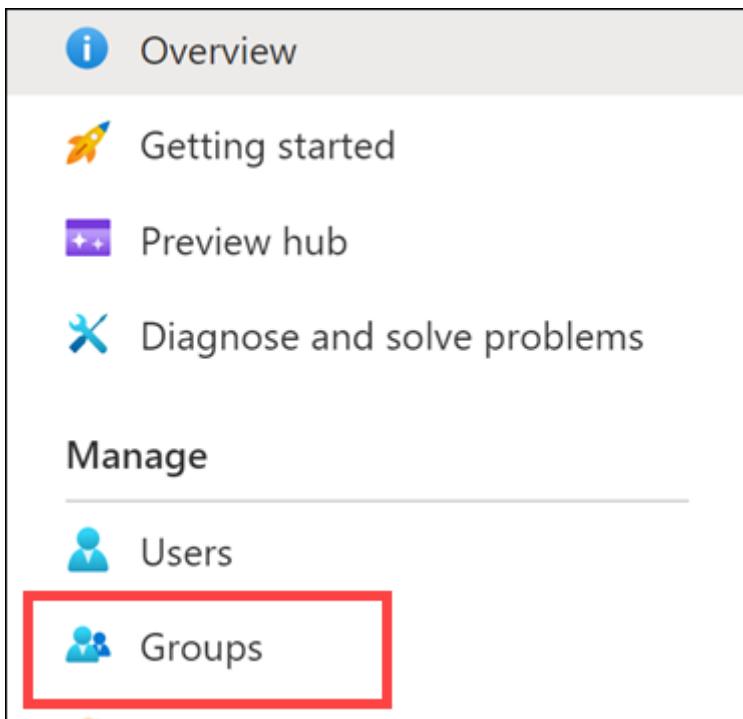
6.4.1 Task 1: Create Azure Active Directory security groups

In this segment, we will create security groups as described above. However, our data set ends in 2019, so we will create a `tailwind-2019-writers` group instead of 2021.

1. Switch back to the Azure portal (<https://portal.azure.com>) in a different browser tab, leaving Synapse Studio open.
2. Select the Azure menu (1), then select **Azure Active Directory** (2).



3. Select **Groups** in the left-hand menu.



4. Select **+ New group**.



5. Select **Security** from **Group type**. Enter **tailwind-history-owners-<suffix>** (where **<suffix>** is a unique value, such as your initials followed by two or more numbers) for the **Group name**, then select **Create**.

New Group

Group type * ⓘ
Security

Group name * ⓘ
tailwind-history-owners-jdh42

Group description ⓘ
Enter a description for the group

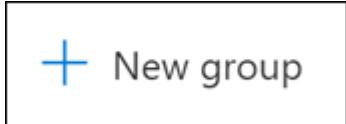
Membership type * ⓘ
Assigned

Owners
No owners selected

Members
No members selected

Create

6. Select **+** **New group**.



7. Select **Security** from **Group type**. Enter **tailwind-readers-<suffix>** (where **<suffix>** is a unique value, such as your initials followed by two or more numbers) for the **Group name**, then select **Create**.

New Group

Group type * ⓘ
Security

Group name * ⓘ
tailwind-readers-jdh42

Group description ⓘ
Enter a description for the group

Membership type * ⓘ
Assigned

Owners
No owners selected

Members
No members selected

Create

8. Select + New group.

+ New group

9. Select **Security** from **Group type**. Enter **tailwind-current-writers-<suffix>** (where <suffix> is a unique value, such as your initials followed by two or more numbers) for the **Group name**, then select **Create**.

New Group ...

Group type * ⓘ
Security

Group name * ⓘ
tailwind-current-writers-jdh42

Group description ⓘ
Enter a description for the group

Membership type * ⓘ
Assigned

Owners
No owners selected

Members
No members selected

Create

10. Select + New group.



11. Select **Security** from **Group type**. Enter **tailwind-2019-writers-<suffix>** (where <suffix> is a unique value, such as your initials followed by two or more numbers) for the **Group name**, then select **Create**.

New Group ...

Group type * ⓘ
Security

Group name * ⓘ
tailwind-2019-writers-jdh42

Group description ⓘ
Enter a description for the group

Membership type * ⓘ
Assigned

Owners
No owners selected

Members
No members selected

Create

6.4.2 Task 2: Add group members

To test out the permissions, we will add our own account to the `tailwind-readers-<suffix>` group.

1. Open the newly created `tailwind-readers-<suffix>` group.
2. Select **Members** (1) on the left, then select **+ Add members** (2).

tailwind-readers | Members

Group

« **Add members** 2 Remove Refresh

Overview Diagnose and solve problems

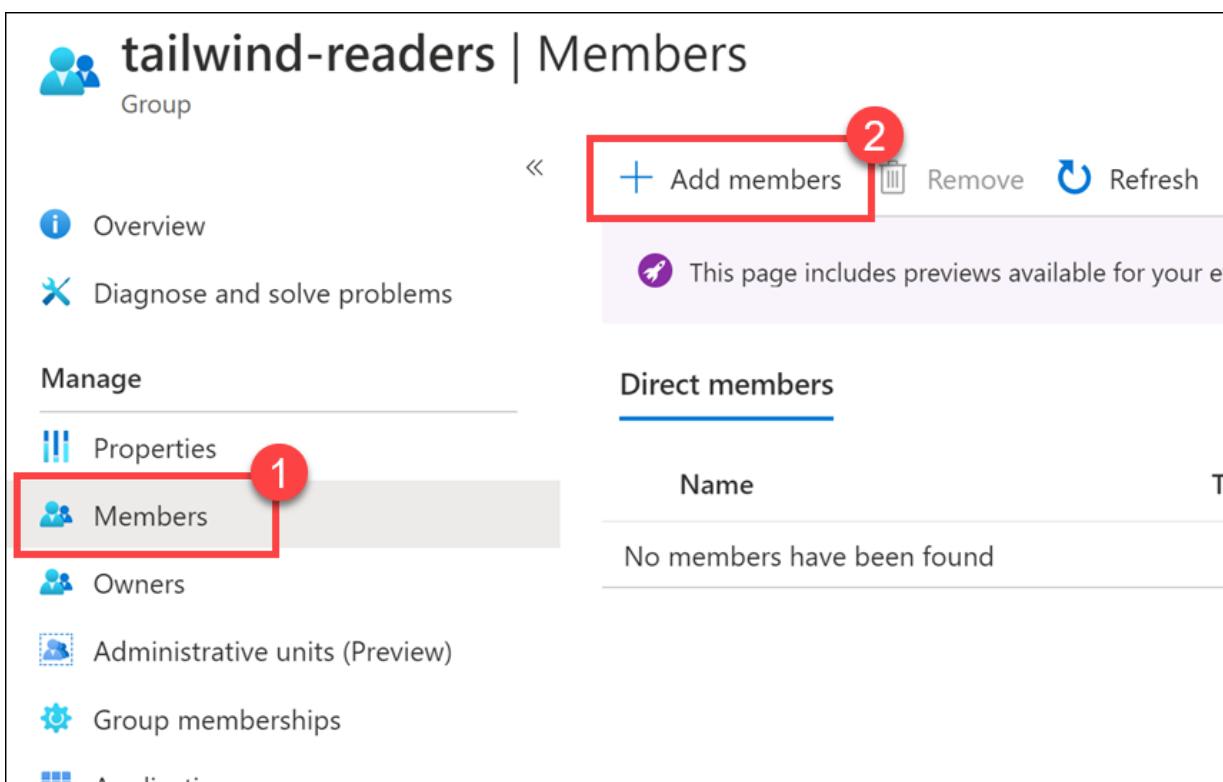
Manage

Properties 1 Members Owners Administrative units (Preview) Group memberships Applications

Direct members

Name

No members have been found



3. Add your user account that you are signed into for the lab, then select **Select**.

The screenshot shows a modal window titled "Add members". At the top left is a search bar with the placeholder "Search" and a magnifying glass icon. Below it is a search input field containing the name "joel". A list of results follows, showing a blurred profile picture, the name "Joel", the email "joel@...", and the status "Selected". Below this is a section titled "Selected items" containing a list of one item: a blurred profile picture, the name "Joel", the email "joel@...", and a "Remove" button. At the bottom is a large blue "Select" button.

4. Open the `tailwind-2019-writers-<suffix>` group.
5. Select **Members** (1) on the left, then select **+ Add members** (2).

The screenshot shows the "tailwind-2019-writers | Members" page. At the top left is a group icon and the title "tailwind-2019-writers | Members". Below it is a "Group" label. On the left, there's a sidebar with links: "Overview" (with a blue info icon), "Diagnose and solve problems" (with a blue cross icon), "Properties" (with a blue bar chart icon), "Members" (with a blue people icon, highlighted with a red box and a red circle containing the number 1), "Owners" (with a blue people icon), "Administrative units (Preview)" (with a blue document icon), "Group memberships" (with a blue gear icon), and "Applications" (with a blue grid icon). At the top right are buttons for "Add members" (with a plus sign icon, highlighted with a red box and a red circle containing the number 2), "Remove" (with a trash bin icon), and "Refresh" (with a circular arrow icon). A message box says "This page includes previews available for your". On the right, under "Direct members", there's a "Name" field and a message "No members have been found".

6. Search for tailwind, select the `tailwind-current-writers-<suffix>` group, then select **Select**.

Add members

X

Search ⓘ

 tailwind

TA

tailwind-2019-writers

TA

tailwind-current-writers
Selected

TA

tailwind-history-owners

TA

tailwind-readers

Selected items

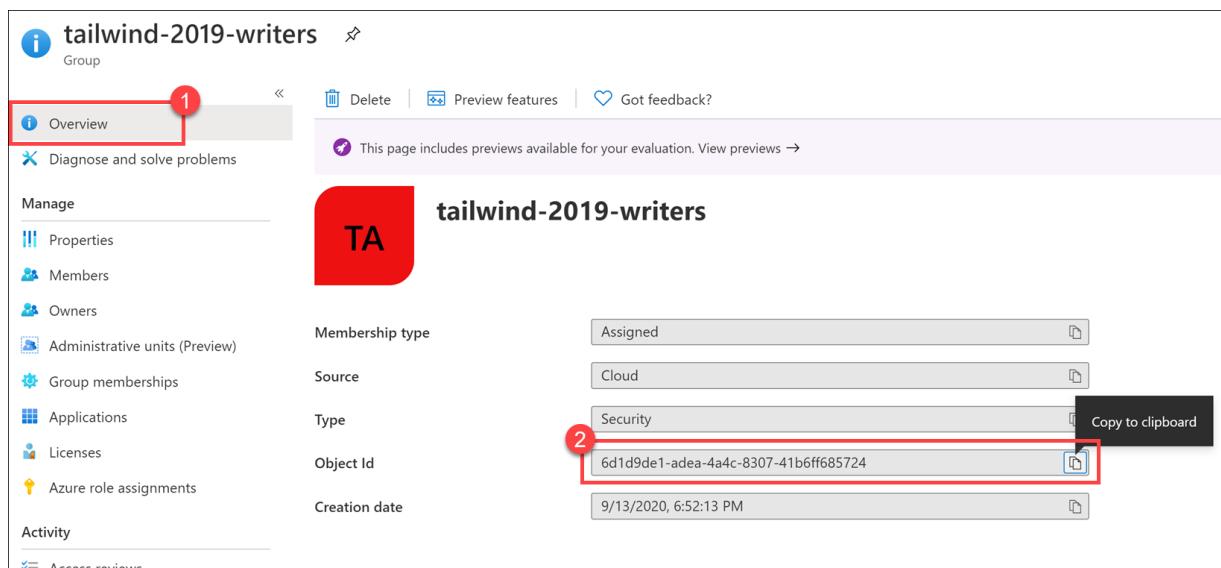
TA

tailwind-current-writers

[Remove](#)

[Select](#)

7. Select **Overview** in the left-hand menu, then **copy** the **Object Id**.



tailwind-2019-writers

Group

Overview

Diagnose and solve problems

Manage

- Properties
- Members
- Owners
- Administrative units (Preview)
- Group memberships
- Applications
- Licenses
- Azure role assignments

tailwind-2019-writers

TA

Membership type: Assigned

Source: Cloud

Type: Security

Object Id: 6d1d9de1-adea-4a4c-8307-41b6ff685724

Creation date: 9/13/2020, 6:52:13 PM

Note: Save the **Object Id** value to Notepad or similar text editor. This will be used in a later step when you assign access control in the storage account.

6.4.3 Task 3: Configure data lake security - Role-Based Access Control (RBAC)

1. Open the Azure resource group for this lab, which contains the Synapse Analytics workspace.
2. Open the default data lake storage account.

Showing 1 to 25 of 25 records. <input type="checkbox"/> Show hidden types ⓘ		No g
<input type="checkbox"/> Name ↑↓	Type ↑↓	
<input type="checkbox"/> amlworkspaceinaday84	Machine Learning	
<input type="checkbox"/> asaappinsightsinaday84	Application Insights	
<input type="checkbox"/> asacosmosdbinaday84	Azure Cosmos DB account	
<input type="checkbox"/> asadatalakeinaday84	Storage account	

3. Select **Access Control (IAM)** in the left-hand menu.

The screenshot shows the Azure Storage account 'asadatalakeinaday84' settings page. The left sidebar lists several options: Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data transfer, and Events. The 'Access Control (IAM)' option is highlighted with a red box. The main content area displays the storage account's name, type (Storage account), a search bar, and an 'Open in Explorer' button. A sidebar on the right provides information about classic alerts and includes links for Essentials, Resource group, Status, Location, and Subscription.

4. Select the **Role assignments** tab.

34 | Access Control (IAM)

The screenshot shows the 'Role assignments' tab selected in the Azure IAM interface. A red box highlights the 'Role assignments' button. Below it, a section titled 'Number of role assignments for this subscription' displays a progress bar.

5. Select + Add, then Add role assignment.

The screenshot shows the 'Add role assignment' dialog. A red box highlights the '+ Add' button. Below it, another red box highlights the 'Add role assignment' button. A third red box highlights the 'Add co-administrator' button. A black button labeled 'Add role assignment' is shown in the background.

6. For **Role**, select **Storage Blob Data Reader**. Search for **tailwind-readers** and select **tailwind-readers-<suffix>** from the results, then select **Save**.

Add role assignment

X

Role ⓘ

Storage Blob Data Reader ⓘ



Assign access to ⓘ

User, group, or service principal



Select ⓘ

tailwind-readers

No users, groups, or service principals found.

Selected members:

TA

tailwind-readers-jdh42

Remove

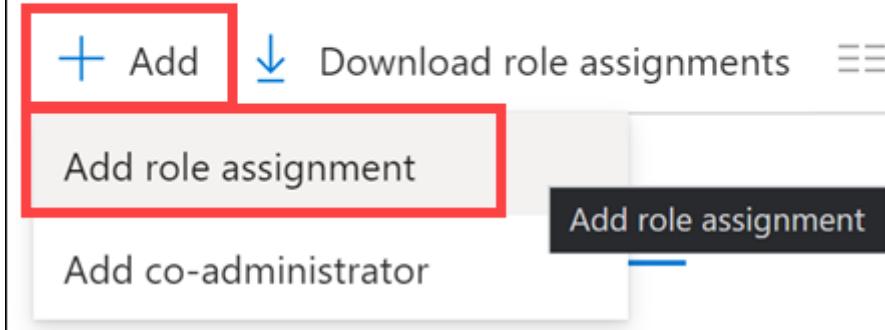
Save

Discard

Since our user account is added to this group, we will have read access to all files in the blob containers of this account. Tailwind Traders will need to add all users to the tailwind-readers-<suffix> security group.

7. Select + Add, then Add role assignment.

4 | Access Control (IAM)



8. For **Role**, select **Storage Blob Data Owner**. Search for **tailwind** and select **tailwind-history-owners-<suffix>** from the results, then select **Save**.

Add role assignment

X

Role ⓘ

Storage Blob Data Owner ⓘ

Assign access to ⓘ

User, group, or service principal

Select ⓘ

tailwind

TA

tailwind-2019-writers-jdh42

TA

tailwind-current-writers-jdh42

TA

tailwind-readers-jdh42

Selected members:

TA

tailwind-history-owners-jdh42

Remove

Save

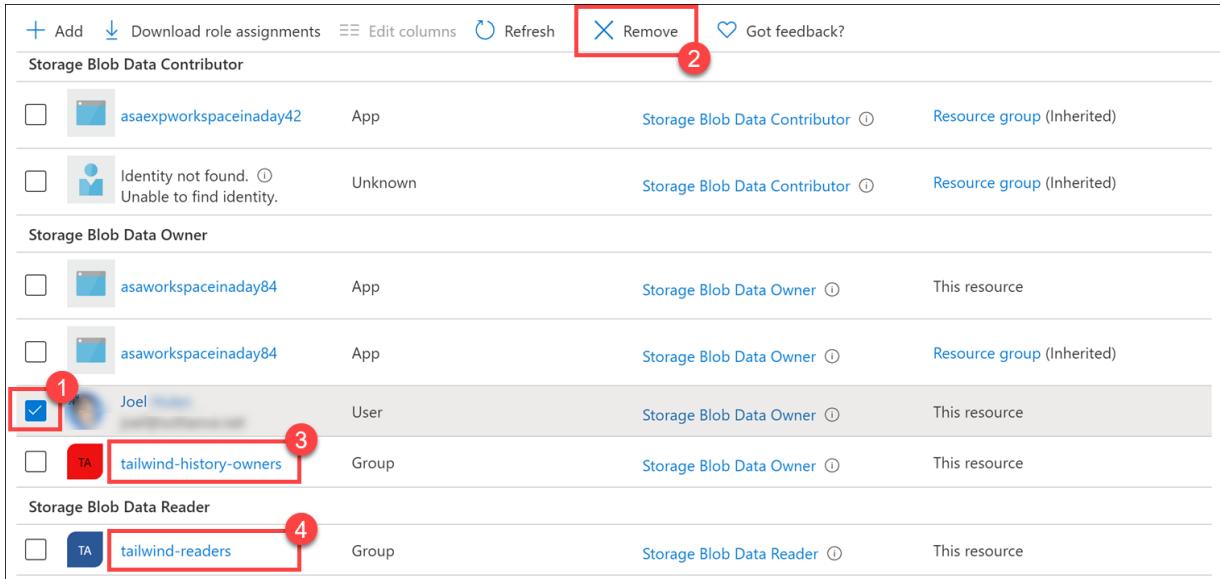
Discard

The tailwind-history-owners-<suffix> security group is now assigned to the Azure Storage built-in RBAC role Storage Blob Data Owner for the Azure Storage account containing the data lake. This allows Azure AD user and service principals that are added to this role to have the ability to modify all data.

Tailwind Traders needs to add the user security principals who will have have permissions to modify all

historical data to the `tailwind-history-owners-<suffix>` security group.

- In the **Access Control (IAM)** list for the storage account, select your Azure user account under the **Storage Blob Data Owner** role (1), then select **Remove** (2).



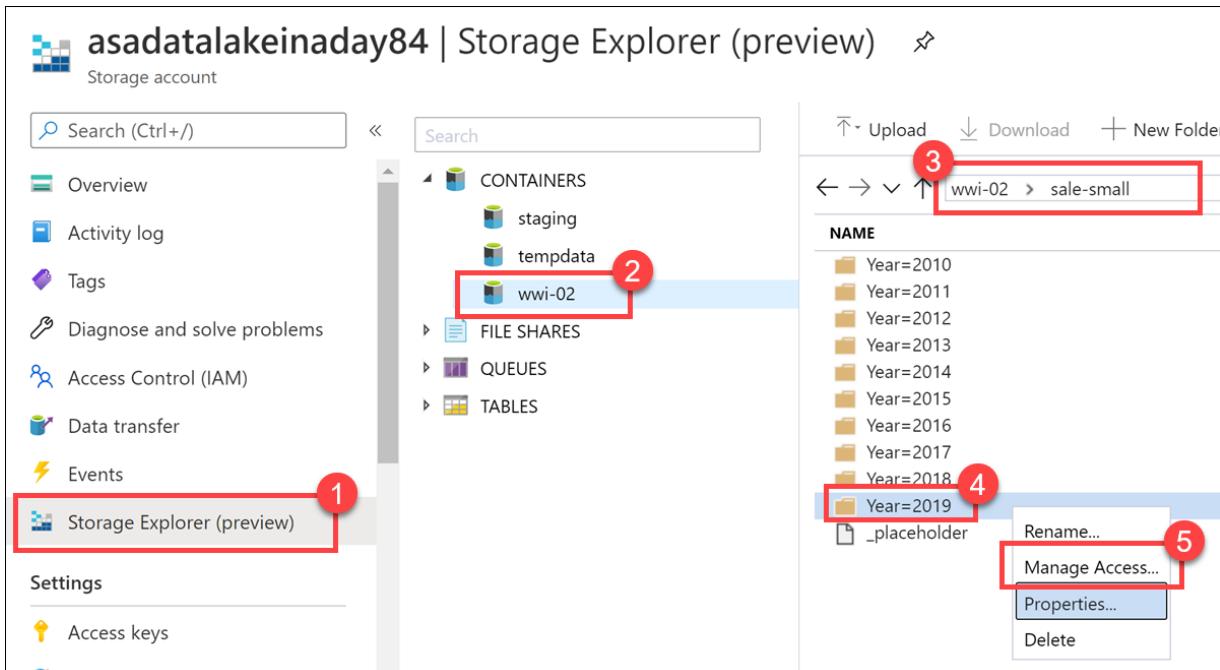
Storage Blob Data Contributor			
<input type="checkbox"/>  asaexpworkspaceinaday42	App	Storage Blob Data Contributor	Resource group (Inherited)
<input type="checkbox"/>  Identity not found. <small>(1)</small> Unable to find identity.	Unknown	Storage Blob Data Contributor	Resource group (Inherited)
Storage Blob Data Owner			
<input type="checkbox"/>  asaworkspaceinaday84	App	Storage Blob Data Owner	This resource
<input type="checkbox"/>  asaworkspaceinaday84	App	Storage Blob Data Owner	Resource group (Inherited)
<input checked="" type="checkbox"/>  Joel	User	Storage Blob Data Owner	This resource
<input type="checkbox"/> TA  tailwind-history-owners	Group	Storage Blob Data Owner	This resource
Storage Blob Data Reader			
<input type="checkbox"/> TA  tailwind-readers	Group	Storage Blob Data Reader	This resource

Notice that the `tailwind-history-owners-<suffix>` group is assigned to the **Storage Blob Data Owner** group (3), and `tailwind-readers-<suffix>` is assigned to the **Storage Blob Data Reader** group (4).

Note: You may need to navigate back to the resource group, then come back to this screen to see all of the new role assignments.

6.4.4 Task 4: Configure data lake security - Access Control Lists (ACLs)

- Select **Storage Explorer (preview)** on the left-hand menu (1). Expand CONTAINERS and select the **wwi-02** container (2). Open the **sale-small** folder (3), right-click on the **Year=2019** folder (4), then select **Manage Access..** (5).



- Paste the **Object Id** value you copied from the `tailwind-2019-writers-<suffix>` security group into the **Add user, group, or service principal** text box, then select **Add**.

Manage Access

X

Managing permissions for: wwi-02/sale-small/Year=2019

Users, groups, and service principals:

\$superuser (Owner)	
\$superuser (Owning Group)	
Other	
Mask	

Permissions for: \$superuser

Read Write Execute

Access



Default *



* This will automatically add these permissions to all new children of this directory. Learn more about default ACLs.

Add user, group, or service principal:

6d1d9de1-adea-4a4c-8307-41b6ff685724



Add

Save

[Cancel](#)

- Now you should see that the `tailwind-2019-writers-<suffix>` group is selected in the Manage Access dialog (1). Check the **Access** and **Default** check boxes and the **Read**, **Write**, and **Execute** checkboxes for each (2), then select **Save**.

Manage Access

X

Managing permissions for: wwi-02/sale-small/Year=2019

Users, groups, and service principals:

\$superuser (Owner)



\$superuser (Owning Group)



Other

Mask

tailwind-2019-writers

6d1d9de1-adea-4a4c-8307-41b6ff685724



1

Permissions for: 6d1d9de1-adea-4a4c-8307-41b6ff685724

Read

Write

Execute

Access

Default *

* This will automatically add these permissions to all new children of this directory. Learn more about default ACLs.

2

Add user, group, or service principal:

Enter a UPN or Object ID



Add

Save

Cancel

Now the security ACLs have been set to allow any users added to the `tailwind-current-<suffix>` security group to write to the `Year=2019` folder, by way of the `tailwind-2019-writers-<suffix>` group. These users can only manage current (2019 in this case) sales files.

At the start of the following year, to revoke write access to the 2019 data they would remove the `tailwind-current-writers-<suffix>` security group from the `tailwind-2019-writers-<suffix>` group. Members of `tailwind-readers-<suffix>` would continue to be able to read the contents of the file system because they have been granted read and execute (list) permissions not by the ACLs but by the RBAC built in role at the level of the file system.

Notice that we configured both the *access* ACLs and *default* ACLs in this configuration.

Access ACLs control access to an object. Files and directories both have access ACLs.

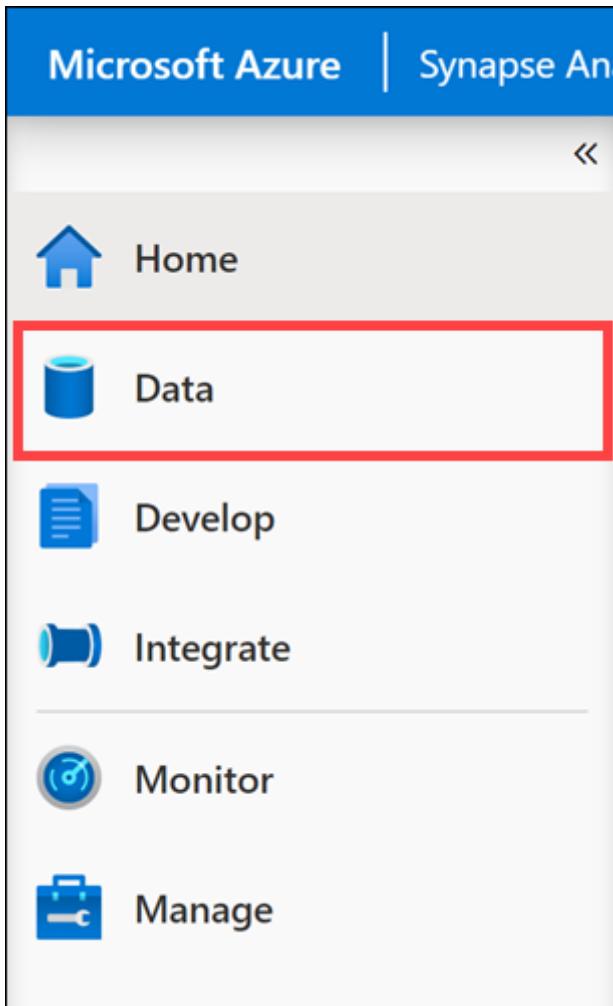
Default ACLs are templates of ACLs associated with a directory that determine the access ACLs for any

child items that are created under that directory. Files do not have default ACLs.

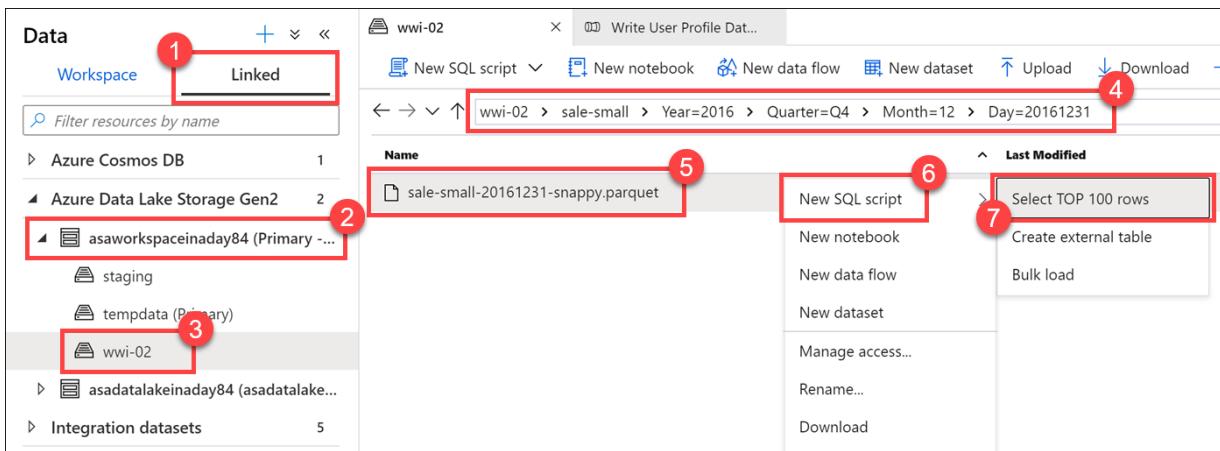
Both access ACLs and default ACLs have the same structure.

6.4.5 Task 5: Test permissions

1. In Synapse Studio, navigate to the **Data** hub.



2. Select the **Linked** tab (1) and expand **Azure Data Lake Storage Gen2**. Expand the **asaworkspaceXX** primary ADLS Gen2 account (2) and select the **wwi-02** container (3). Navigate to the **sale-small/Year=2016/Quarter=Q4/Month=12/Day=20161231** folder (4). Right-click on the **sale-small-20161231-snappy.parquet** file (5), select **New SQL script** (6), then **Select TOP 100 rows** (7).



3. Ensure **Built-in** is selected (1) in the **Connect to** dropdown list above the query window, then run the

query (2). Data is loaded by the serverless SQL pool endpoint and processed as if was coming from any regular relational database.

```

1 SELECT
2     TOP 100 *
3     FROM
4         OPENROWSET(
5             BULK 'https://asadatalakeinaday84.dfs.core.windows.net/wwi-02/sale-small/Year=2016/Quarter=Q4',
6             FORMAT='PARQUET'
7         ) AS [result]
8

```

The cell output shows the query results from the Parquet file.

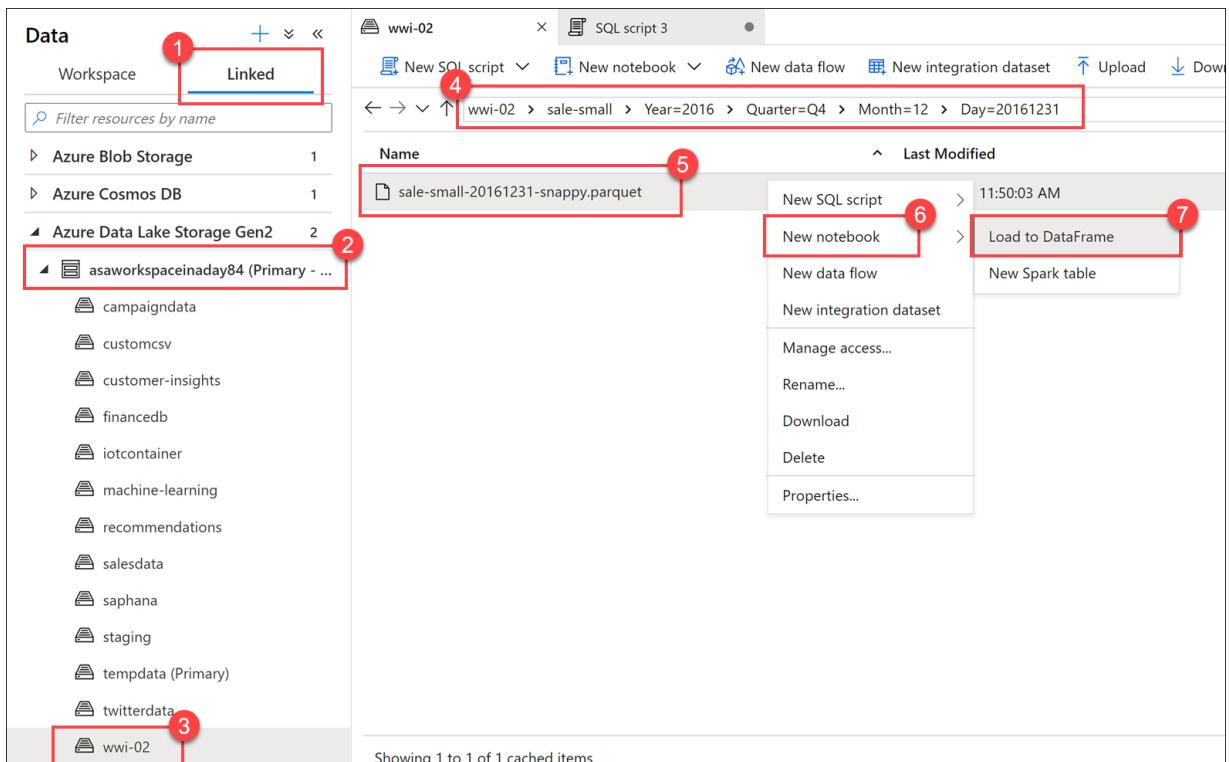
TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount	TransactionDate	ProfitAmount	Hour
bebd0650-b54f...	3	3208	4	33.8600000000...	135.4400000000...	20161231	43.2000000000...	6
bebd0650-b54f...	3	1820	4	28.6500000000...	114.6000000000...	20161231	35.6800000000...	6
bebd0650-b54f...	3	3372	2	23.0400000000...	46.0800000000...	20161231	13.0600000000...	6
bebd0650-b54f...	3	1373	3	34.6500000000...	103.9500000000...	20161231	31.0200000000...	6
bebd0650-b54f...	3	2760	1	36.1300000000...	36.1300000000...	20161231	7.2500000000...	6
bebd0650-b54f...	3	2253	4	26.8000000000...	107.2000000000...	20161231	28.0000000000...	6
bebd0650-b54f...	3	3498	4	25.0800000000...	100.3200000000...	20161231	28.6800000000...	6
bebd0650-b54f...	3	2358	3	23.2900000000...	69.8700000000...	20161231	24.9300000000...	6
bebd0650-b54f...	3	198	4	30.8200000000...	123.2800000000...	20161231	41.8000000000...	6

The read permissions to the Parquet file assigned to us through the `tailwind-readers-<suffix>` security group, which then is granted RBAC permissions on the storage account through the **Storage Blob Data Reader** role assignment, is what enabled us to view the file contents.

However, since we removed our account from the **Storage Blob Data Owner** role, and we did not add our account to the `tailwind-history-owners-<suffix>` security group, what if we try to write to this directory?

Let's give it a try.

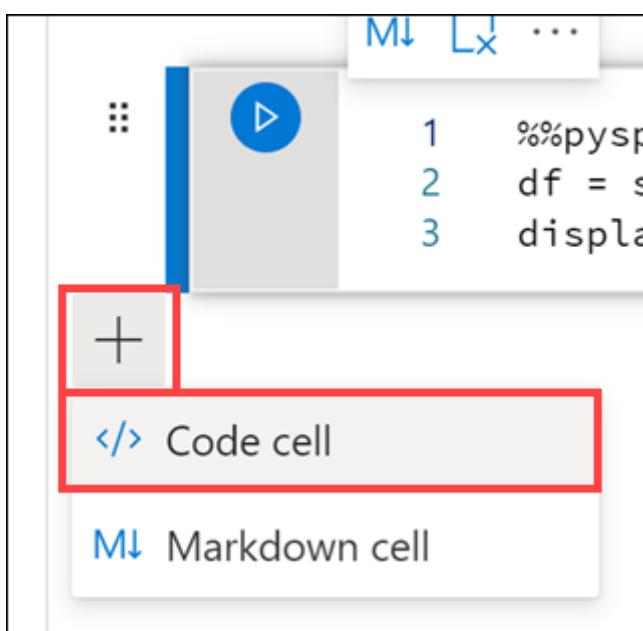
- In the **Data** hub, once again select the **Linked** tab (1) and expand **Azure Data Lake Storage Gen2**. Expand the `asaworkspaceXX` primary ADLS Gen2 account (2) and select the `wwi-02` container (3). Navigate to the `sale-small/Year=2016/Quarter=Q4/Month=12/Day=20161231` folder (4). Right-click on the `sale-small-20161231-snappy.parquet` file (5), select **New Notebook** (6), then select **Load to DataFrame** (7).



5. Attach your Spark pool to the notebook.



6. In the notebook, select +, then </> Code cell underneath Cell 1 to add a new code cell.



7. Enter the following in the new cell, then **copy the Parquet path from cell 1** and paste the value to replace REPLACE_WITH_PATH (1). Rename the Parquet file by adding -test to the end of the file name (2):

```
df.write.parquet('REPLACE_WITH_PATH')
```

```

1 %spark
2 df = spark.read.load('abfss://wwi-02@asadatalake356357.dfs.core.windows.net/sale-small/Year=2016/Quarter=Q4/Month=12/Day=20161231/sale-small-20161231-snappy.parquet', format='parquet')
3 display(df.limit(10))

```

```

1 df.write.parquet('abfss://wwi-02@asadatalake356357.dfs.core.windows.net/sale-small/Year=2016/Quarter=Q4/Month=12/Day=20161231/sale-small-20161231-snappy-test.parquet')

```

8. Select **Run all** in the toolbar to run both cells. After a few minutes when the Spark pool starts and the cells run, you should see the file data in the output from cell 1 (1). However, you should see a **403 error** in the output of cell 2 (2).

bebd0650-b54f-...	3	2253	4	26.8	107.2	20161231
bebd0650-b54f-...	3	3498	4	25.08	100.32	20161231
bebd0650-b54f-...	3	2358	3	23.29	69.87	20161231
bebd0650-b54f-...	3	198	4	30.82	123.28	20161231
bebd0650-b54f-...	3	190	1	24.65	24.65	20161231

Cell 2

```

1 df.write.parquet('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/sale-small/Year=2016, ...

```

Command executed in 2s 583ms by joel on 11-27-2020 10:43:03.494 -05:00

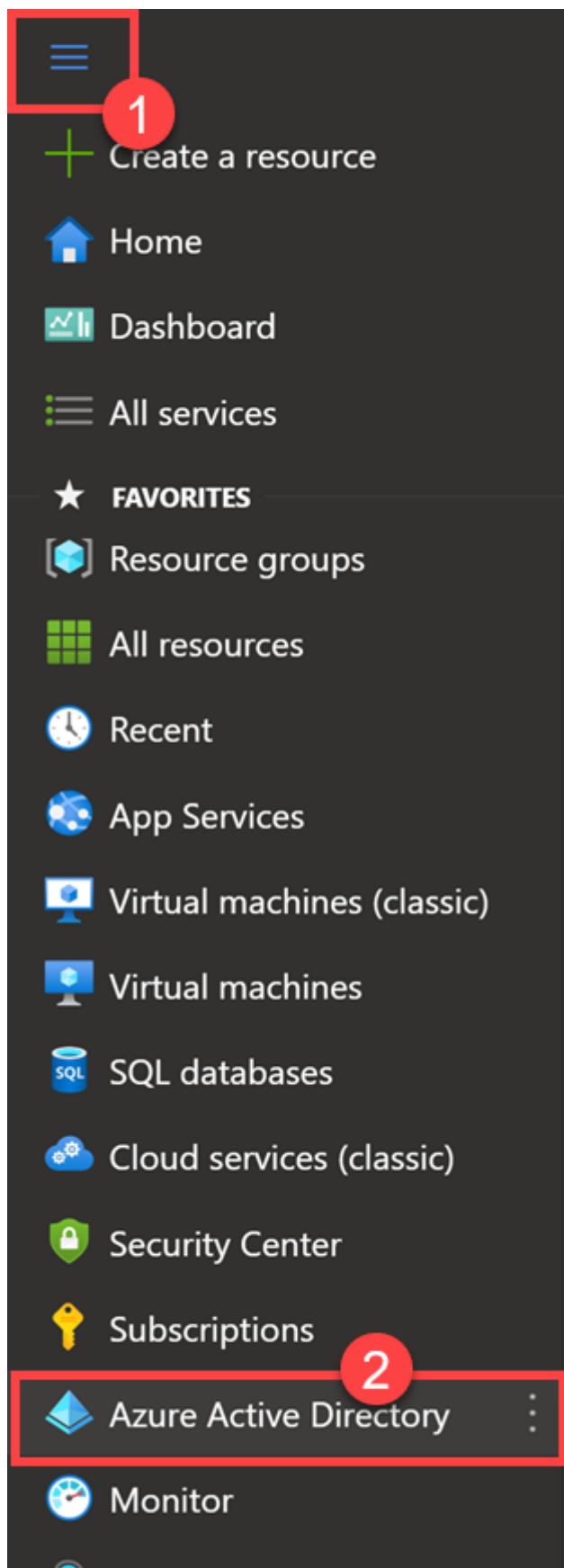
```

Py4JJavaError : An error occurred while calling o155.parquet.
: Operation failed: "This request is not authorized to perform this operation using this permission.", 403, PUT,
https://asadatalakeinaday84.dfs.core.windows.net/wwi-02/sale-
small/Year%3D2016/Quarter%3DQ4/Month%3D12/Day%3D20161231/sale-small-20161231-snappy-test.parquet/_temporary/0?
resource=directory&timeout=90, AuthorizationPermissionMismatch, "This request is not authorized to perform this
operation using this permission. RequestId:1ea8d457-401f-0078-27d3-c4ff84000000 Time:2020-11-27T15:43:02.4187420Z"
at org.apache.hadoop.fs.azurebfs.services.AbfsRestOperation.execute(AbfsRestOperation.java:166)
at org.apache.hadoop.fs.azurebfs.services.AbfsClient.createPath(AbfsClient.java:278)
at org.apache.hadoop.fs.azurebfs.AzureBlobFileSystemStore.createDirectory(AzureBlobFileSystemStore.java:419)
at org.apache.hadoop.fs.azurebfs.AzureBlobFileSystem.mkdirs(AzureBlobFileSystem.java:402)

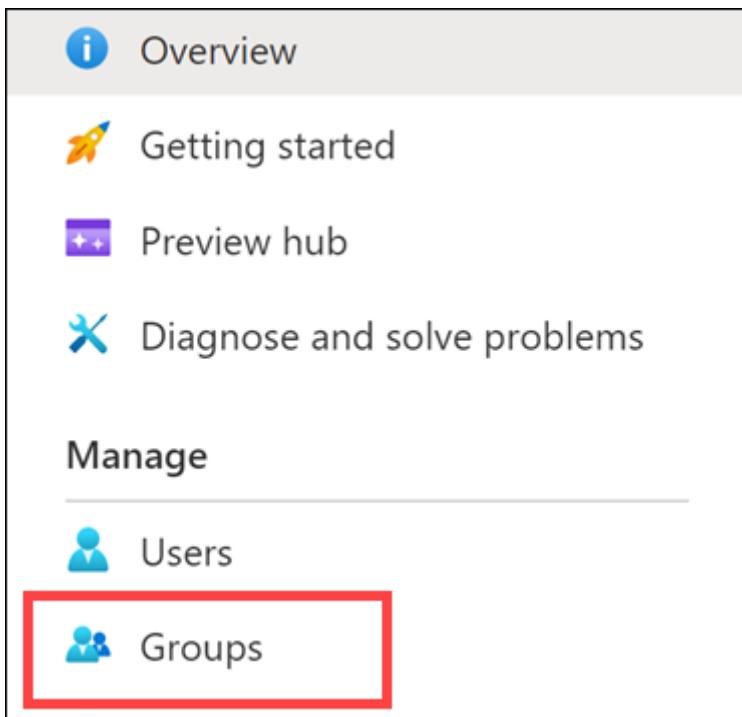
```

As expected, we do not have write permissions. The error returned by cell 2 is, **This request is not authorized to perform this operation using this permission.**, with a status code of 403.

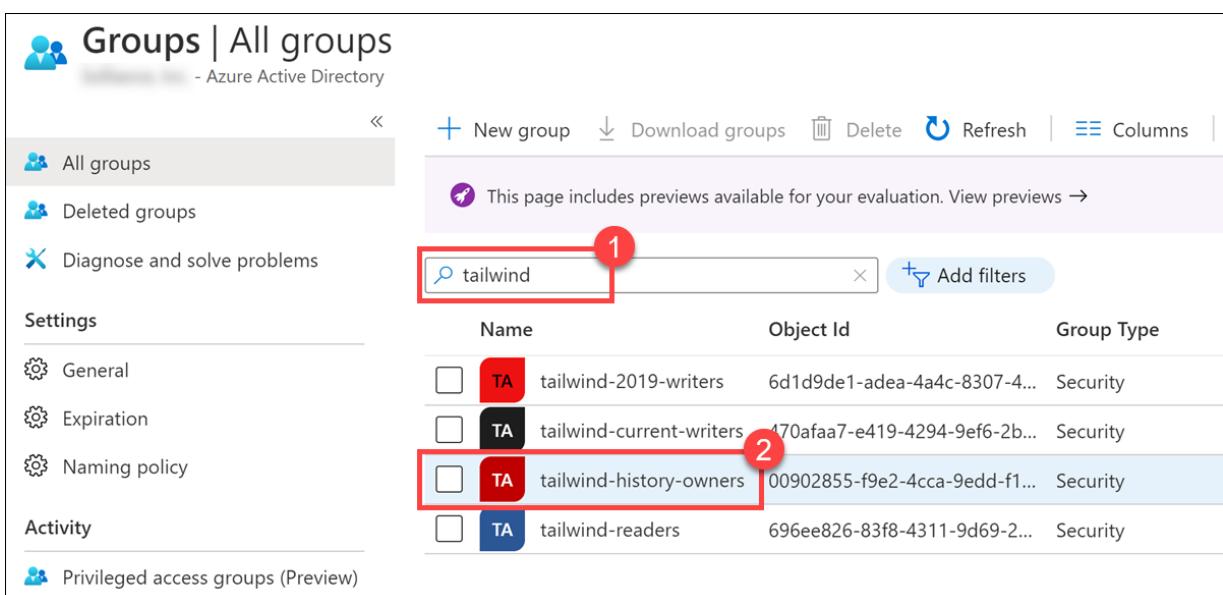
- Leave the notebook open and switch back to the Azure portal (<https://portal.azure.com>) in another tab.
- Select the Azure menu (1), then select **Azure Active Directory** (2).



11. Select **Groups** in the left-hand menu.



12. Type **tailwind** in the search box (1), then select **tailwind-history-owners-<suffix>** in the results (2).



The screenshot shows the 'Groups | All groups' page in Azure Active Directory. The search bar at the top contains the text 'tailwind' (1). The results table lists several groups:

Name	Object Id	Group Type
tailwind-2019-writers	6d1d9de1-adea-4a4c-8307-4...	Security
tailwind-current-writers	470afaa7-e419-4294-9ef6-2b...	Security
tailwind-history-owners	00902855-f9e2-4cca-9edd-f1...	Security
tailwind-readers	696ee826-83f8-4311-9d69-2...	Security

The group 'tailwind-history-owners' is highlighted with a red box and a red circle containing the number '2'.

13. Select **Members** (1) on the left, then select **+ Add members** (2).

tailwind-history-owners | Members

Group

« Add members Remove Refresh

Overview Diagnose and solve problems

Manage

Properties Members Owners Administrative units (Preview) Group memberships Applications

Direct members

Name

No members have been found

14. Add your user account that you are signed into for the lab, then select **Select**.

Add members

Search ⓘ

joel

Joel
joel@
Selected

Selected items

Joel
joel@

Select

15. Switch back to the open Synapse Notebook in Synapse Studio, then **Run** cell 2 once more (1). You should see a status of **Succeeded** (2) after a few moments.

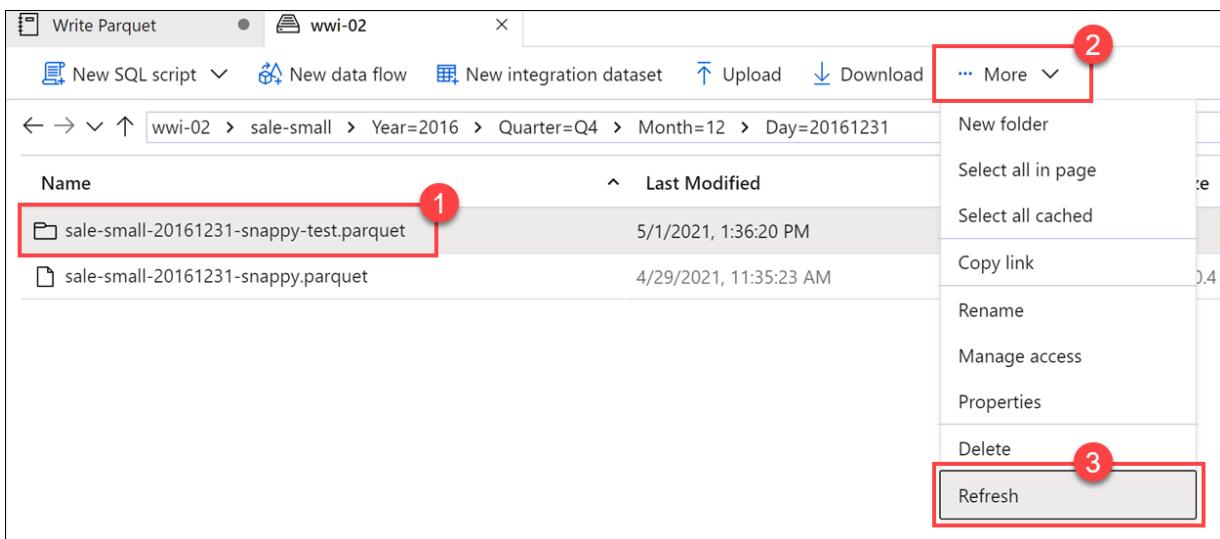
The screenshot shows a Jupyter Notebook cell titled "Cell 2". The cell contains a single line of Python code: `df.write.parquet('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/sale-small/Year=2016, ...`. A red box labeled "1" highlights the play button icon at the start of the cell. Another red box labeled "2" highlights the status message "Job execution Succeeded" and "Spark 2 executors 8 cores". Below the cell, a message says "Command executed in 4s 413ms by joel on 11-27-2020 10:47:31.987 -05:00". At the bottom right, there are links for "View in monitoring" and "Open Spark UI".

The cell succeeded this time because we added our account to the `tailwind-history-owners-<suffix>` group, which is assigned the **Storage Blob Data Owner** role.

Note: If you encounter the same error this time, stop the Spark session on the notebook, then select **Publish all**, then Publish. After publishing your changes, select your user profile on the top-right corner of the page and **log out**. Close the browser tab after logout is successful, then re-launch Synapse Studio (<https://web.azuresynapse.net/>), re-open the notebook, then re-run the cell. This may be needed because you must refresh the security token for the auth changes to take place.

Now let's verify that the file was written to the data lake.

16. Navigate back to the `sale-small/Year=2016/Quarter=Q4/Month=12/Day=20161231` folder. You should now see a folder for the new `sale-small-20161231-snappy-test.parquet` file we wrote from the notebook (1). If you don't see it listed here, select ... More in the toolbar (2), then select Refresh (3).



7 Module 5 - Explore, transform, and load data into the Data Warehouse using Apache Spark

This module teaches how to explore data stored in a data lake, transform the data, and load data into a relational data store. The student will explore Parquet and JSON files and use techniques to query and transform JSON files with hierarchical structures. Then the student will use Apache Spark to load data into the data warehouse and join Parquet data in the data lake with data in the dedicated SQL pool.

In this module, the student will be able to:

- Perform Data Exploration in Synapse Studio
- Ingest data with Spark notebooks in Azure Synapse Analytics
- Transform data with DataFrames in Spark pools in Azure Synapse Analytics
- Integrate SQL and Spark pools in Azure Synapse Analytics

7.1 Lab details

- Module 5 - Explore, transform, and load data into the Data Warehouse using Apache Spark
 - Lab details
 - Lab setup and pre-requisites
 - Exercise 0: Start the dedicated SQL pool
 - Exercise 1: Perform Data Exploration in Synapse Studio
 - * Task 1: Exploring data using the data previewer in Azure Synapse Studio
 - * Task 2: Using serverless SQL pools to explore files
 - * Task 3: Exploring and fixing data with Synapse Spark
 - Exercise 2: Ingesting data with Spark notebooks in Azure Synapse Analytics
 - * Task 1: Ingest and explore Parquet files from a data lake with Apache Spark for Azure Synapse
 - Exercise 3: Transforming data with DataFrames in Spark pools in Azure Synapse Analytics
 - * Task 1: Query and transform JSON data with Apache Spark for Azure Synapse
 - Exercise 4: Integrating SQL and Spark pools in Azure Synapse Analytics
 - * Task 1: Update notebook
 - Exercise 5: Cleanup
 - * Task 1: Pause the dedicated SQL pool

7.2 Lab setup and pre-requisites

Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Complete the **lab setup instructions** for this module.

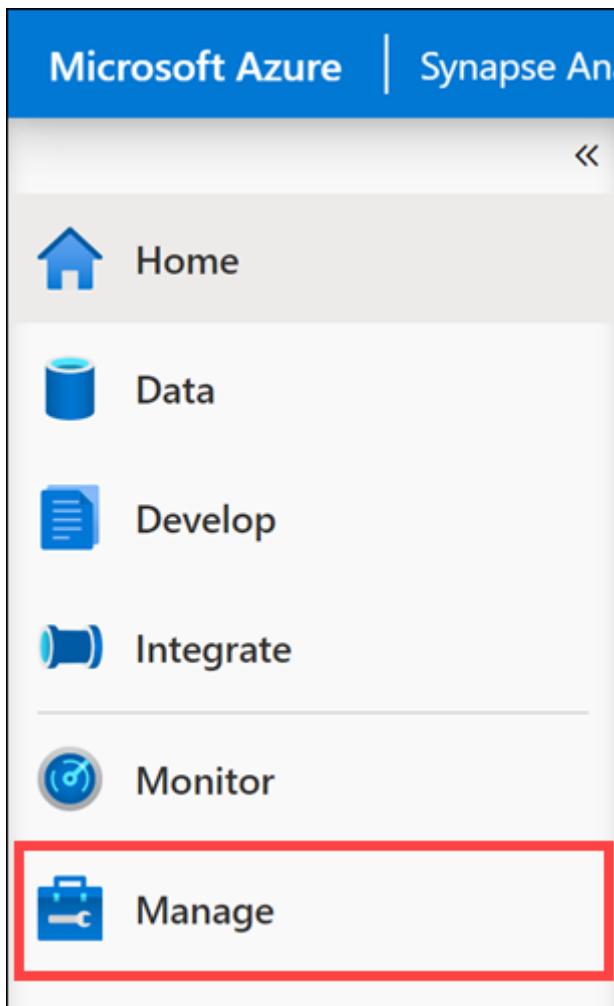
Note, the following modules share this same environment:

- Module 4
- Module 5
- Module 7
- Module 8
- Module 9
- Module 10
- Module 11
- Module 12
- Module 13
- Module 16

7.3 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the 'SQL pools' blade in the Azure portal. The left sidebar shows:

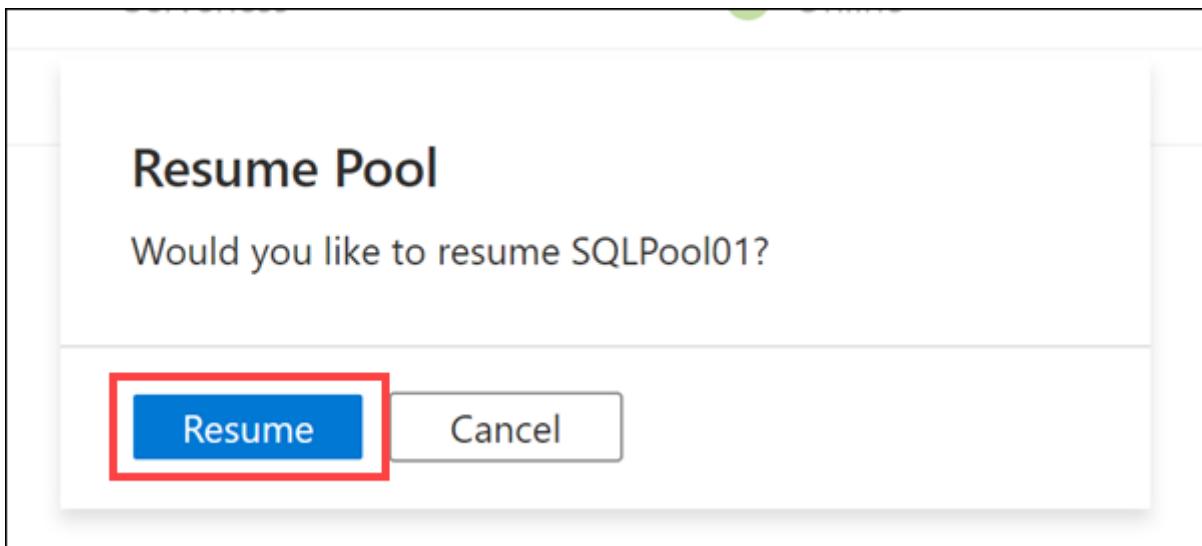
- Analytics pools
- SQL pools** (highlighted with a red box and a red number 1)
- Apache Spark pools
- External connections
- Linked services
- Azure Purview (Preview)
- Integration
- Triggers
- Integration runtimes
- Security
- Access control

The main area displays the 'SQL pools' table:

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused	DW100c

A red box highlights the 'Paused' status of the 'SQLPool01' row (2).

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

7.4 Exercise 1: Perform Data Exploration in Synapse Studio

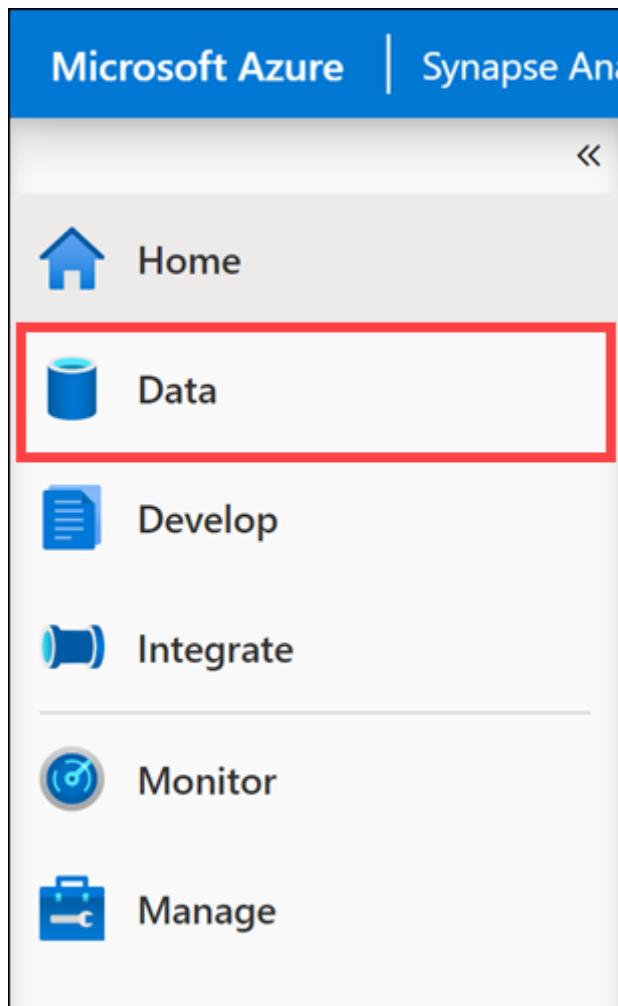
One of the first data engineering tasks typically performed during data ingestion is to explore the data that is to be imported. Data exploration allows engineers to understand better the contents of files being ingested. This process helps to identify any potential data quality issues that might hinder automated ingestion processes. Through exploration, we can gain insights into data types, data quality, and whether any processing needs to be performed on the files prior to importing the data into your data lake or using it for analytics workloads.

The engineers at Tailspin Traders have run into issues ingesting some of their sales data into the data warehouse, and have requested assistance in understanding how Synapse Studio can be used to help them resolve these issues. As the first step of this process, you need to explore the data to understand what is causing the issues they've encountered, and then provide them with a solution.

7.4.1 Task 1: Exploring data using the data previewer in Azure Synapse Studio

Azure Synapse Studio provides numerous ways to explore data, from a simple preview interface to more complicated programmatic options using Synapse Spark notebooks. In this exercise, you will learn how to use these features to explore, identify, and fix problematic files. You will be exploring CSV files stored in the wwi-02/sale-poc folder of the data lake and learning about how to identify and fix issues.

1. Open Synapse Studio (<https://web.azuresynapse.net/>), and then navigate to the **Data** hub.



The Data hub is where you access your provisioned SQL pool databases and SQL serverless databases in your workspace, as well as external data sources, such as storage accounts and other linked services.

2. We want to access files stored in the workspace's primary data lake, so select the **Linked** tab within the Data hub.



3. On the Linked tab, expand **Azure Data Lake Storage Gen2** and then expand the **Primary** data lake for the workspace.

Data + ⌂ <<

Workspace **Linked**

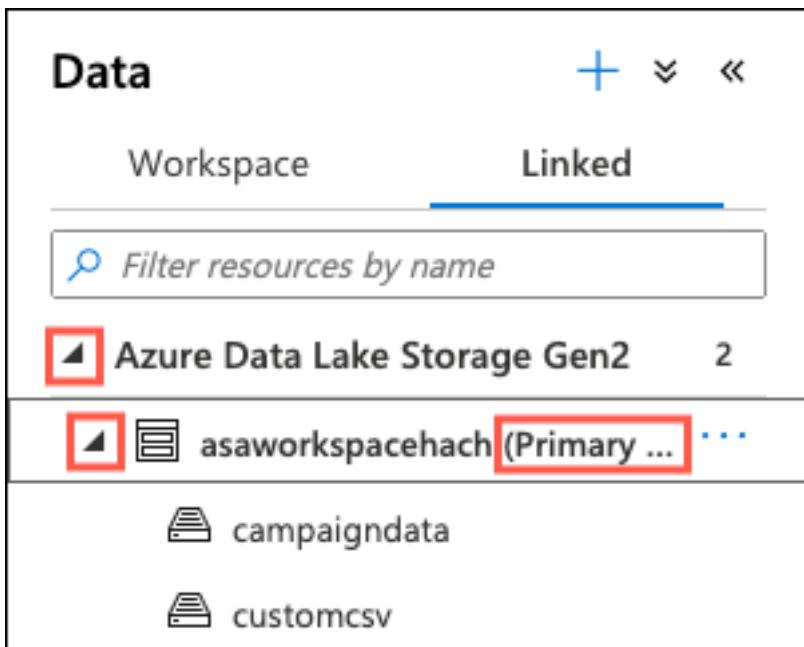
Filter resources by name

 Azure Data Lake Storage Gen2 2

  asaworkspacehach (Primary ... ⋮

 campaigndata

 customcsv

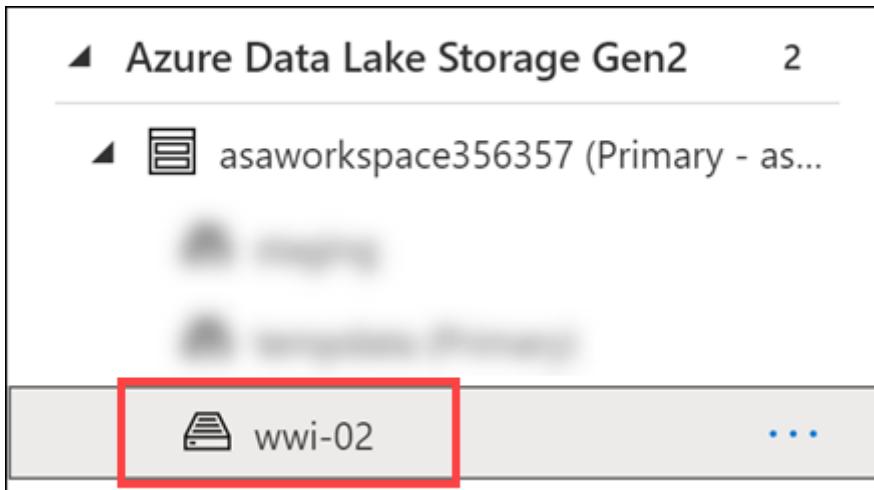


4. In the list of containers within the primary data lake storage account, select the `wwi-02` container.

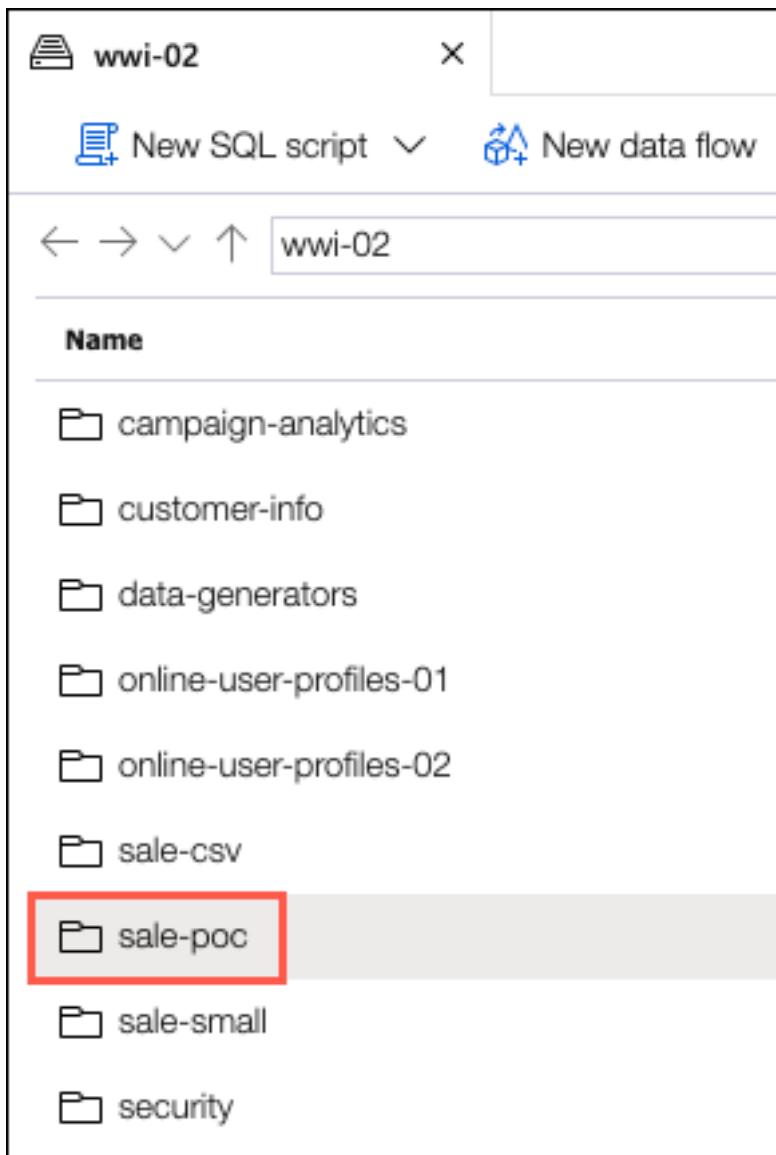
 Azure Data Lake Storage Gen2 2

  asaworkspace356357 (Primary - as... ⋮

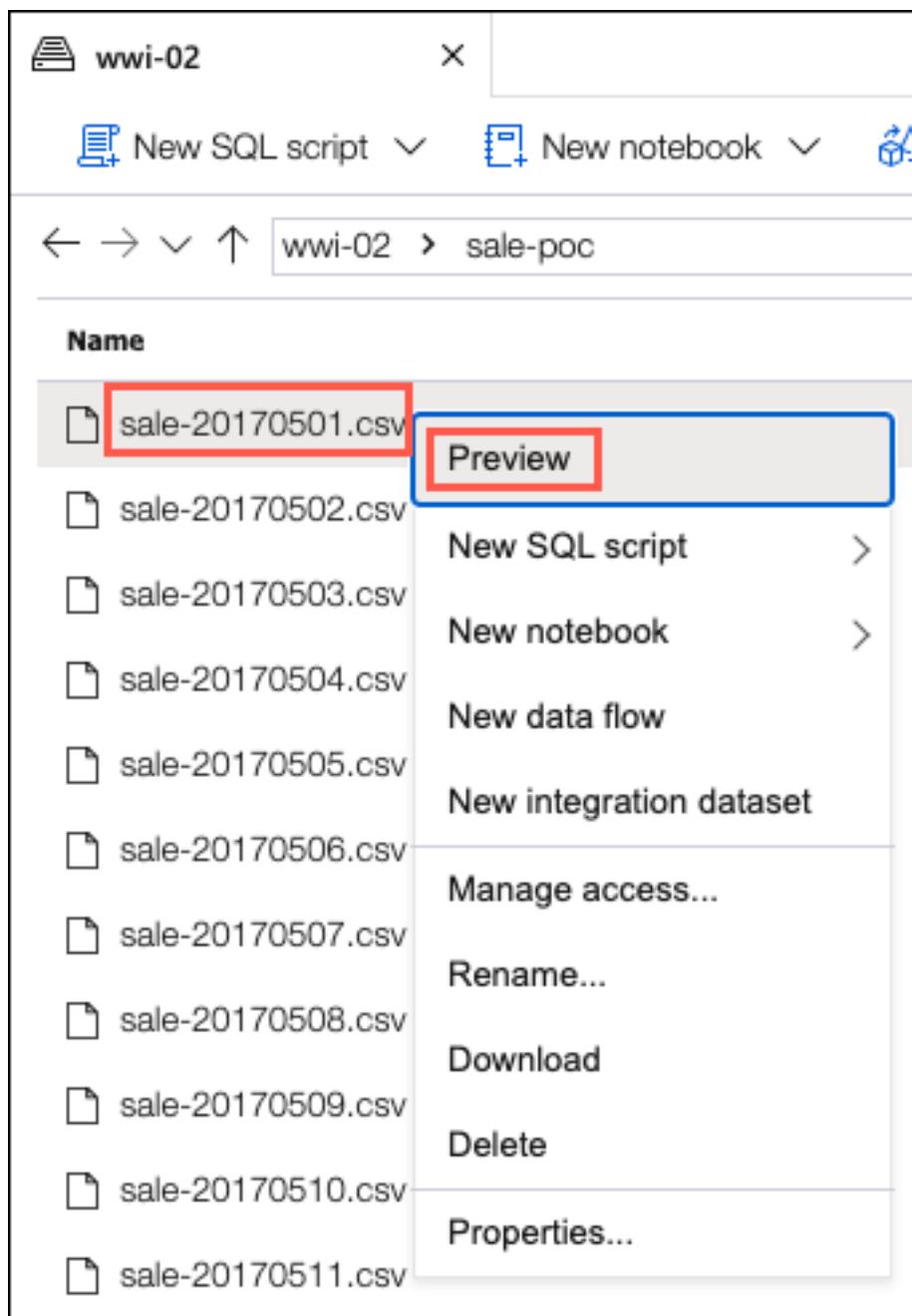
 wwi-02 ⋮



5. In the container explorer window, double-click on the `sale-poc` folder to open it.



6. The **sale-poc** contains sales data for the month of May, 2017. There are 31 files in the folder, one for each day of the month. These files were imported by a temporary process to account for an issue with Tailspin's import process. Let's now take a few minutes to explore some of the files.
7. Right-click the first file in the list, **sale-20170501.csv**, and select **Preview** from the context menu.



8. The **Preview** functionality in Synapse Studio provides an quick and code-free way to examine the contents of a file. This is an effective way of getting a basic understanding of the features (columns) and types of data stored within them for an individual file.

sale-20170501.csv

Path <https://asadatalakehach.dfs.core.windows.net/wwi-02/sale-poc/sale-20170501.csv>

Modified 10/15/2020, 4:54:08 PM

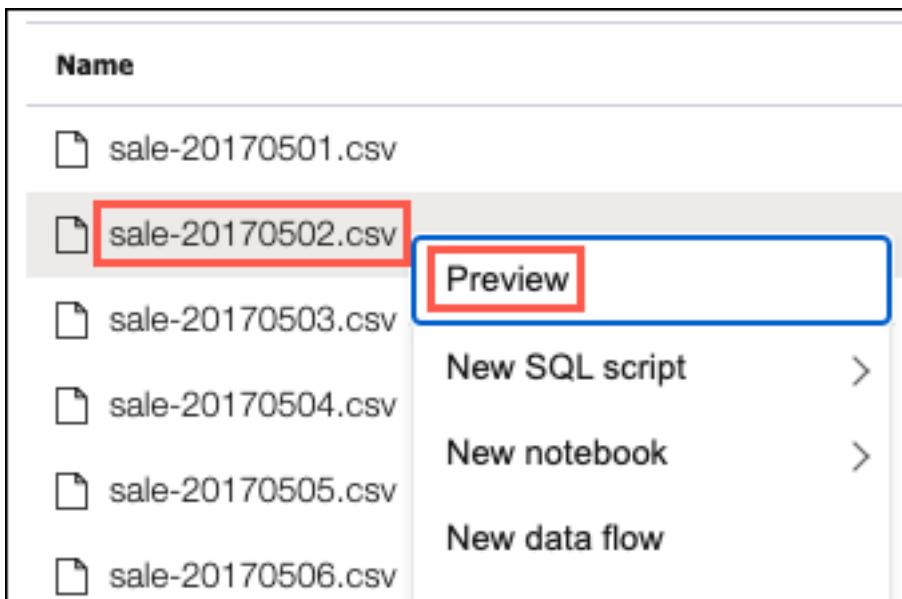
With column header On

TRANSACTIONID	CUSTOMERID	PRODUCTID	QUANTITY
e067fc11-e07d-4...	3	4581	4
e067fc11-e07d-4...	3	1365	4
e067fc11-e07d-4...	3	2641	4
e067fc11-e07d-4...	3	220	2
e067fc11-e07d-4...	3	110	3
e067fc11-e07d-4...	3	2	1
cdd2ed88-8aae-...	11	3323	1
cdd2ed88-8aae-...	11	4763	4
cdd2ed88-8aae-...	11	4070	1
cdd2ed88-8aae-...	11	582	3
cdd2ed88-8aae-...	11	194	1
cdd2ed88-8aae-...	11	70	1
cdd2ed88-8aae-...	11	32	3
b3a06a7b-6325...	41	3321	2
b3a06a7b-6325...	41	244	2
b3a06a7b-6325...	41	3009	1
b3a06a7b-6325...	41	126	3

OK

While in the Preview dialog for `sale-20170501.csv`, take a moment to scroll through the file preview. Scrolling down shows there are a limited number of rows included in the preview, so this is just a glimpse into the structure of the file. Scrolling to the right allows you to see the number and names of the columns contained in the file.

9. Select **OK** to close the preview of the `sale-20170501.csv` file.
10. When performing data exploration, it is important to look at more than just one file, as it helps to get a more representative sample of the data. Let's look at the next file in the `wwi-02\sale-poc` folder. Right-click the `sale-20170502.csv` file and select **Preview** from the context menu.



11. In the Preview dialog, you will immediately notice the structure of this file is different from the `sale-20170501.csv` file. No data rows appear in the preview and the column headers appear to contain data and not field names.

sale-20170502.csv

Path <https://asadatalakehach.dfs.core.windows.net/wwi-02/sale-poc/sale-20170502.csv>

Modified 10/15/2020, 4:54:09 PM

With column header On

586682A8-DB...	2	610	4
----------------	---	-----	---

OK

12. In the preview dialog, you have to option to turn off the **With column header** option. As it appears this file does not contain column headers, set this to off and inspect the results.

sale-20170502.csv

Path <https://asadatalakehach.dfs.core.windows.net/wwi-02/sale-poc/sale-20170502.csv>

Modified 10/15/2020, 4:54:09 PM

With column header Off

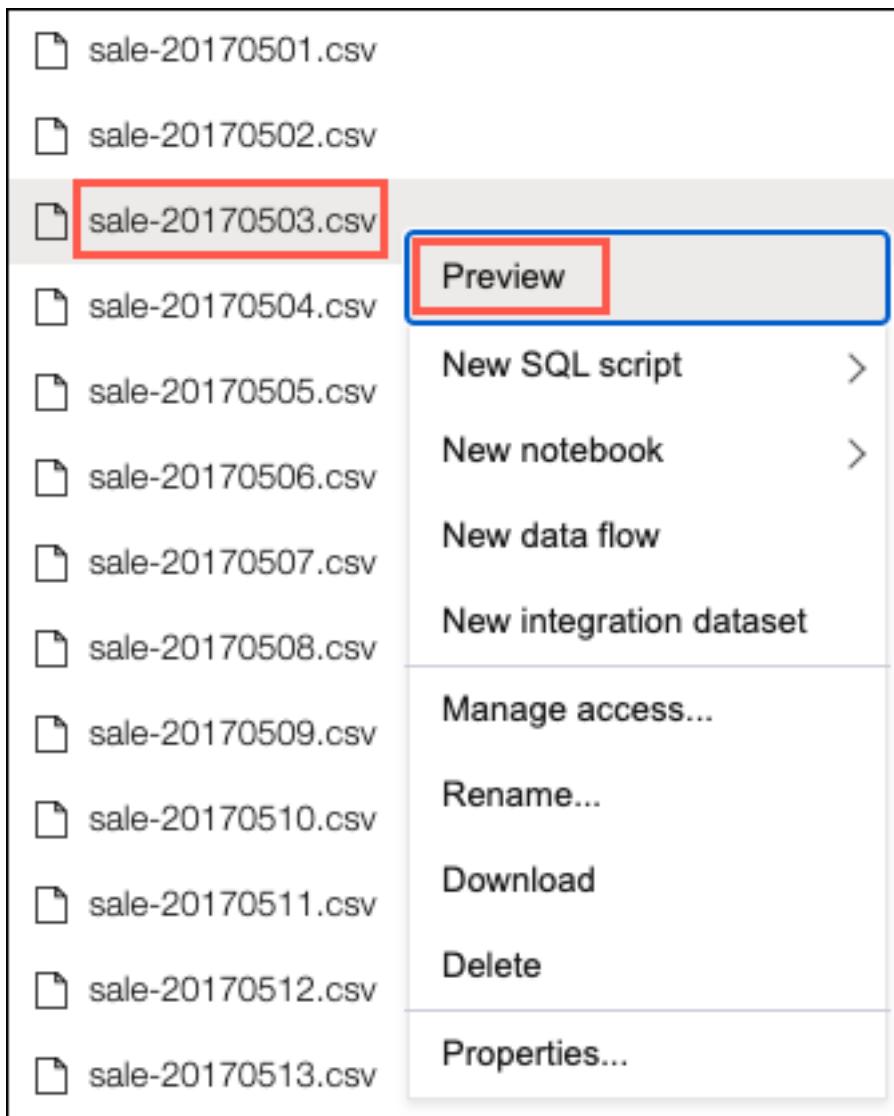
(NO COLUM... <th>(NO COLUM...<th>(NO COLUM...<th>(NO COLUM...</th></th></th>	(NO COLUM... <th>(NO COLUM...<th>(NO COLUM...</th></th>	(NO COLUM... <th>(NO COLUM...</th>	(NO COLUM...
--	---	------------------------------------	--------------

586682a8-db73-... <td>2<td>610<td>4</td></td></td>	2 <td>610<td>4</td></td>	610 <td>4</td>	4
--	--------------------------	----------------	---

OK

Setting the **With column headers** to off verifies that the file does not contain column headers. All columns have "(NO COLUMN NAME)" in the header. This setting moves the data down appropriately, and it appears this is only a single row. By scrolling to the right, you will notice that while there appears to only be a single row, there are many more columns than what we saw when previewing the first file. That file contained 11 columns.

13. Since we have seen two different file structures, let's look at another file to see if we can learn which format is more typical of the files within the `sale-poc` folder. Right-click the file named `sale-20170503.csv` and select **Preview**, as you have done previously.



14. The preview shows the `sale-20170503.csv` file appears to have a structure similar to that found in `20170501.csv`.

sale-20170503.csv

Path <https://asadatalakehach.dfs.core.windows.net/wwi-02/sale-poc/sale-20170503.csv>

Modified 10/15/2020, 4:54:08 PM

With column header On

TRANSACTIONID	CUSTOMERID	PRODUCTID	QUANTITY
bd85db4f-a973-...	16	2076	4
bd85db4f-a973-...	16	4655	2
bd85db4f-a973-...	16	699	1
bd85db4f-a973-...	16	699	1
bd85db4f-a973-...	16	2527	1
bd85db4f-a973-...	16	2490	1
bd85db4f-a973-...	16	3624	4
bd85db4f-a973-...	16	1972	2
bd85db4f-a973-...	16	1462	3
bd85db4f-a973-...	16	4791	1
bd85db4f-a973-...	16	124	2
bd85db4f-a973-...	16	40	3
22ac9d9c-49d6-...	22	487	1
22ac9d9c-49d6-...	22	1759	1
22ac9d9c-49d6-...	22	1415	3
22ac9d9c-49d6-...	22	1759	1
22ac9d9c-49d6-...	22	1523	1

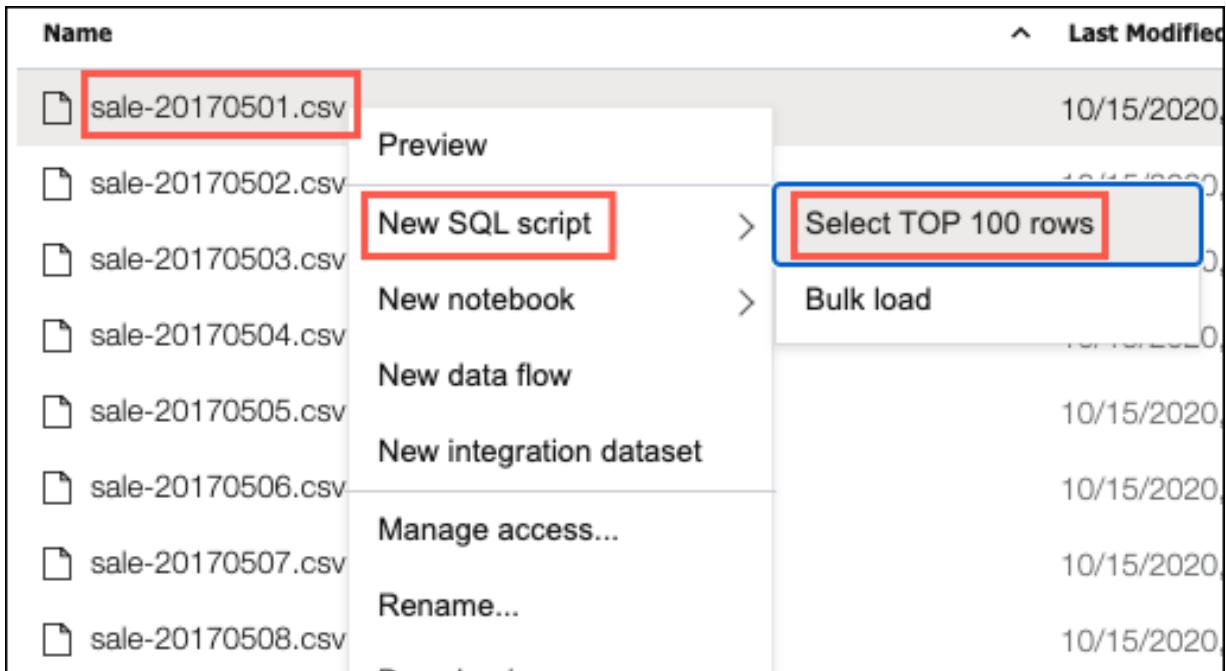
OK

15. Select **OK** to close the preview.
16. Now, take a few minutes to preview a few of the other files in the **sale-poc** folder. Do they have the same structure as the files for May 1st and 3rd?

7.4.2 Task 2: Using serverless SQL pools to explore files

The **Preview** functionality in Synapse Studio enables quick exploration of files, but doesn't allow us to look deeper into the data or gain much in the way of insights into files with issues. In this task, we will use the **serverless SQL pools (built-in)** functionality of Synapse to explore these files using T-SQL.

1. Right-click the **sale-20170501.csv** file again, this time selecting **New SQL Script** and **Select TOP 100 rows** from the context menu.



2. A new SQL script tab will open in Synapse Studio containing a SELECT statement to read the first 100 rows of the file. This provides another way to examine the contents of files. By limiting the number of rows being examined, we can speed up the exploration process, as queries to load all the data within the files will run slower.

```

1 -- This is auto-generated code
2 SELECT
3     TOP 100 *
4 FROM
5     OPENROWSET(
6         BULK 'https://asadatalake264973.dfs.core.windows.net/wwi-02/sale-poc/sale-20170501.csv',
7         FORMAT = 'CSV',
8         PARSE_VERSION='2.0'
9     ) AS [result]
10

```

T-SQL queries against files stored in the data lake leverage the **OPENROWSET** function. The **OPENROWSET** function can be referenced in the **FROM** clause of a query as if it were a table named **OPENROWSET**. It supports bulk operations through a built-in **BULK** provider that enables data from a file to be read and returned as a rowset. To learn more, you can review the [to OPENROWSET documentation](#).

3. Now, select **Run** on the toolbar to execute the query.



4. In the **Results** pane, observe the output.

Results Messages

View Table Chart Export results ^

Search

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount	TransactionDate	ProfitAmount	Hour	Minute	StoreId
e067fc11-e07d-4517-bc...	3	4581	4	20.8400000000...	91.6960000000...	20170501	26.0480000000...	2	30	7922
e067fc11-e07d-4517-bc...	3	1365	4	26.5200000000...	116.6880000000...	20170501	29.4360000000...	2	30	7922
e067fc11-e07d-4517-bc...	3	2641	4	29.7100000000...	130.7240000000...	20170501	37.4000000000...	2	30	7922
e067fc11-e07d-4517-bc...	3	220	2	27.6000000000...	60.7200000000...	20170501	15.3560000000...	2	30	7922
e067fc11-e07d-4517-bc...	3	110	3	28.4100000000...	93.7530000000...	20170501	33.0000000000...	2	30	7922
e067fc11-e07d-4517-bc...	3	2	1	39.7800000000...	43.7580000000...	20170501	11.5280000000...	2	30	7922
cdd2ed88-8aae-4295-8...	11	3323	1	30.5200000000...	33.5720000000...	20170501	10.2520000000...	20	43	3573
cdd2ed88-8aae-4295-8...	11	4763	4	21.2900000000...	93.6760000000...	20170501	29.2160000000...	20	43	3573
cdd2ed88-8aae-4295-8...	11	4070	1	35.7600000000...	39.3360000000...	20170501	10.0760000000...	20	43	3573

In the results, you will notice that the first row, containing the column headers, is rendered as a data row, and the columns are assigned names C1 - C11. You can use the FIRSTROW parameter of the OPENROWSET function to specify the number of the first row of the file to display as data. The default value is 1, so if a file contains a header row, the value can be set to 2 to skip the column headers. You can then specify the schema associated with the file using the WITH clause.

- Let's modify the query to tell it to skip the header row. In your query window, insert the following code snippet immediately after PARSER_VERSION='2.0':

```
, FIRSTROW = 2
```

- Next, insert the following SQL code to specify the schema between the final) and AS [result]:

```
WITH (
    [TransactionId] varchar(50),
    [CustomerId] int,
    [ProductId] int,
    [Quantity] int,
    [Price] decimal(10,3),
    [TotalAmount] decimal(10,3),
    [TransactionDate] varchar(8),
    [ProfitAmount] decimal(10,3),
    [Hour] int,
    [Minute] int,
    [StoreId] int
)
```

- Your final query should look similar to the following (where [YOUR-DATA-LAKE-ACCOUNT-NAME] is the name of your primary data lake storage account):

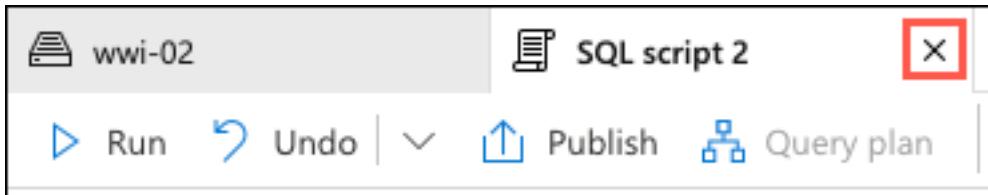
```
SELECT
    TOP 100 *
FROM
    OPENROWSET(
        BULK 'https://\[YOUR-DATA-LAKE-ACCOUNT-NAME\].dfs.core.windows.net/wwi-02/sale-poc/sale-20170501.csv',
        FORMAT = 'CSV',
        PARSER_VERSION='2.0',
        FIRSTROW = 2
    ) WITH (
        [TransactionId] varchar(50),
        [CustomerId] int,
        [ProductId] int,
        [Quantity] int,
        [Price] decimal(10,3),
        [TotalAmount] decimal(10,3),
        [TransactionDate] varchar(8),
        [ProfitAmount] decimal(10,3),
        [Hour] int,
        [Minute] int,
```

```
[StoreId] int
) AS [result]
```

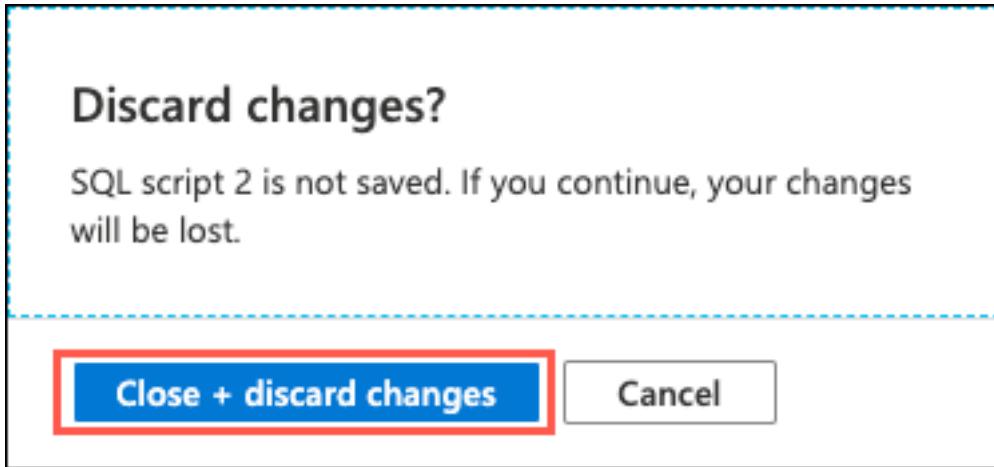
TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount	TransactionDate	ProfitAmount	Hour	Minute	StoreId
e067fc11-e07d-4517-bc93-f7dc4b44f35e	3	4581	4	20.840	91.696	20170501	26.048	2	30	7922
e067fc11-e07d-4517-bc93-f7dc4b44f35e	3	1365	4	26.520	116.688	20170501	29.436	2	30	7922
e067fc11-e07d-4517-bc93-f7dc4b44f35e	3	2641	4	29.710	130.724	20170501	37.400	2	30	7922
e067fc11-e07d-4517-bc93-f7dc4b44f35e	3	220	2	27.600	60.720	20170501	15.356	2	30	7922
e067fc11-e07d-4517-bc93-f7dc4b44f35e	3	110	3	28.410	93.753	20170501	33.000	2	30	7922

Using the `OPENROWSET` function, you can now use T-SQL syntax to further explore your data. For example, you can use a `WHERE` clause to check various fields for `null` or other values that might need to be handled prior to using the data for advanced analytics workloads. With the schema specified, you can refer to fields by name to make this processes easier.

- Close the SQL script tab by selecting the X to the left of the tab name.



- If prompted, select **Close + discard changes** in the Discard changes? dialog.



- We saw while using the **Preview** functionality that the `sale-20170502.csv` file is poorly formed. Let's see if we can learn more about the data in this file using T-SQL. Return to the `wwi-02` tab Right-click the `sale-20170502.csv` file and select **New SQL script** and **Select TOP 100 rows**.

The screenshot shows the Azure Data Studio interface. At the top, there's a toolbar with icons for New SQL script, New notebook, New data flow, and New integration. Below the toolbar, the current workspace is 'wwi-02 > sale-poc'. A list of files is displayed under 'Name' with a 'Last' column header. The files are:

- sale-20170502-fixed
- sale-20170501.csv
- sale-20170502.csv** (highlighted with a red box)
- sale-20170503.csv
- sale-20170504.csv
- sale-20170505.csv

A context menu is open over the 'sale-20170502.csv' file, showing options: Preview, New SQL script, New notebook, Select TOP 100 rows, Bulk load. The 'New SQL script' and 'Select TOP 100 rows' options are highlighted with red boxes.

- As you did previously, select **Run** on the toolbar to execute the query.



- Executing this query results in the error, `Error handling external file: 'Row larger than maximum allowed row size of 8388608 bytes found starting at byte 0.'`, being displayed in the Messages pane.

The 'Messages' pane shows the following log entry:

```

Started executing query at Line 1
Failed to execute query. Error: Error handling external file: 'Row larger than maximum allowed row size of 8388608 bytes found starting at byte 0.'. File: 'https://asadatalakehachdfs.core.windows.net/wwi-02/sale-poc/sale-20170502.csv'.
Total execution time: 00:00:01.351

```

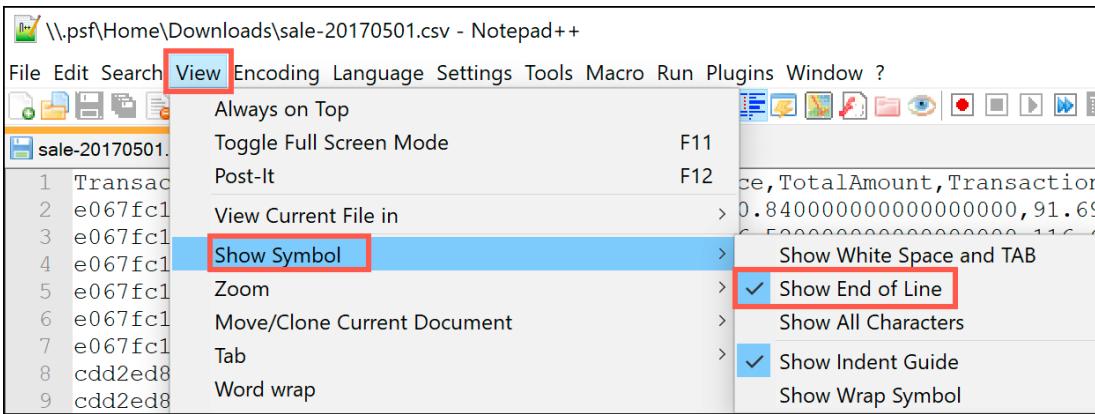
This error aligns with what we saw in the preview window for this file. In the preview we saw the data being separated into columns, but all of the data was in a single row. This implies the data is being split into columns using the default field delimiter (comma). What appears to be missing, however, is a row terminator, \r.

- At this point, we know the `sale-20170502.csv` file is a poorly formed CSV file, so we need to understand the format of the file better so we know how to fix the issues. T-SQL does not provide a mechanism to query the file for what the row terminator character is, so we can use a tool like [Notepad++](#) to discover this.

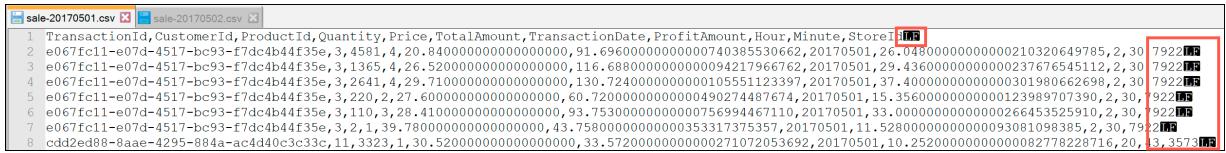
If you do not have Notepad++ installed, feel free to simply view the next three steps.

- By downloading the `sale-20170501.csv` and `sale-20170502.csv` files from your data lake, and then opening them in [Notepad++](#), you can view the end of line characters within the files.

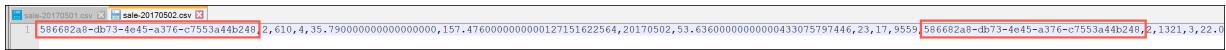
To display the row terminator symbols, open the **View** menu of Notepad++, then select **Show Symbol** and select **Show End of Line**.



15. Opening the `sale-20170501.csv` file in Notepad++ reveals that the well-formatted files contain a line feed (LF) character at the end of each line.



16. Opening the `sale-20170502.csv` file in Notepad++ reveals that there are no row terminator characters. The data is entered into the CSV file as a single row, with only commas separating each field value.



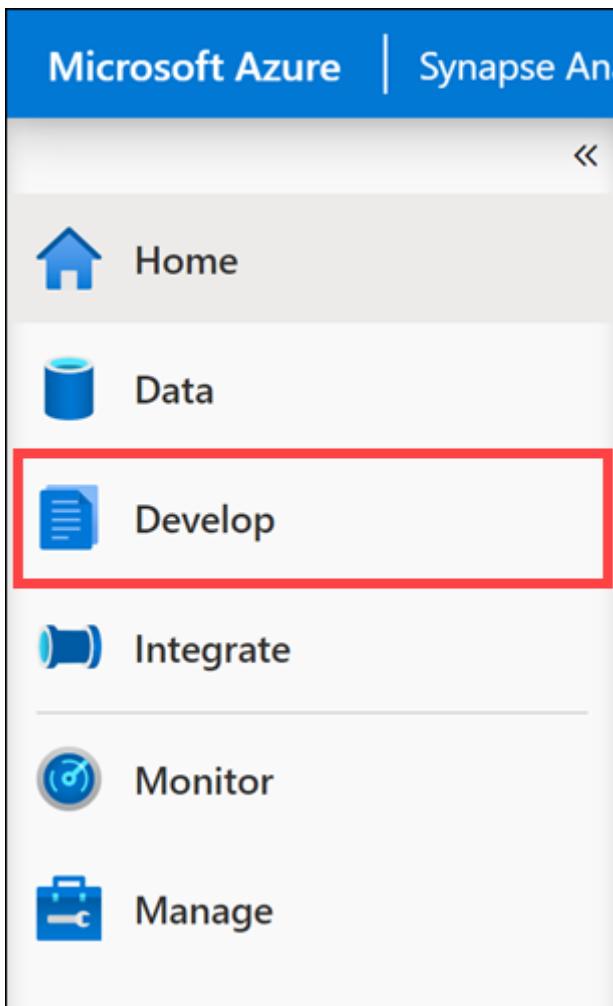
Notice that the data in this file is comprised of a single row, but we can see where the different rows should be. The TransactionId GUID value can be seen every eleventh field in the row. This would indicate that the file encountered some sort of error while processing, which resulted in the column headers and row delimiters being missing from the file.

17. In order for us to fix the file, we need to use code. T-SQL and Synapse Pipelines do not have the ability to efficiently handle this type of issue. To address the problems with this file, we need to use a Synapse Spark notebook.

7.4.3 Task 3: Exploring and fixing data with Synapse Spark

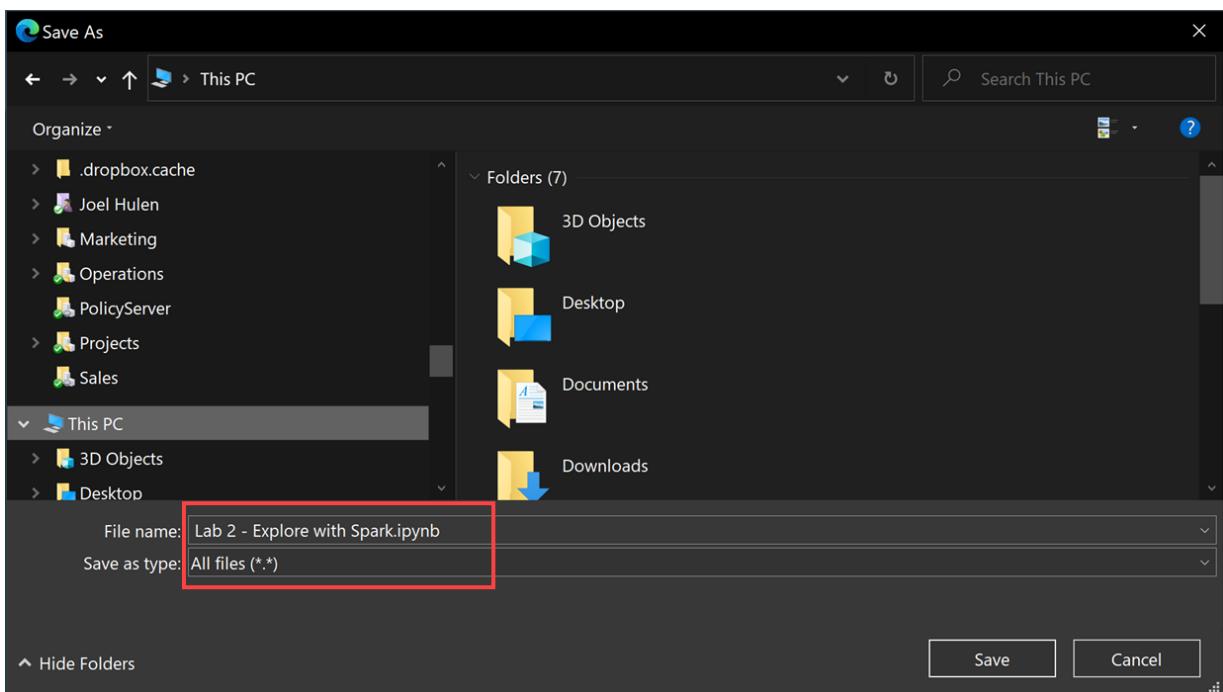
In this task, you will use a Synapse Spark notebook to explore a few of the files in the `wwi-02/sale-poc` folder in the data lake. You will also use Python code to fix the issues with the `sale-20170502.csv` file, so all the files in the directory can be ingested using a Synapse Pipeline later in this lab.

1. In Synapse Studio, open the **Develop** hub.



2. Download the Jupyter notebook for this exercise from <https://solliancepublicdata.blob.core.windows.net/notebooks/Lab%202%20Explore%20with%20Spark.ipynb>. This will download a file named `Lab 2 - Explore with Spark.ipynb`.

The link will open the contents of the file in a new browser window. Select **Save As** in the File menu. By default, the browser will attempt to save this as a text file. If you have the option, set **Save as type** to **All files (.)**. Make sure the file name ends with `.ipynb`.



3. On the Develop hub, select the Add New Resource (+) button and then select Import.

The screenshot shows the Microsoft Azure Synapse Analytics Develop hub. On the left, there is a vertical sidebar with icons: Home (selected), Database, Data Flow, Notebook, Power BI report, and a briefcase icon. The main area has a blue header bar with the Microsoft Azure logo, 'Synapse Analytics', and a workspace name. Below the header is a toolbar with 'Publish all', 'Validate all' (checked), 'Refresh', and 'Discard'. The main content area is titled 'Develop' and contains a search bar with 'Filter resources by name'. A list of resources is shown under 'SQL scripts': '1 SQL Query With Synaps', '2 JSON Extractor', '8 External Data To Synaps', 'Activity 03 - Data Warehc', and 'Lab 05 - Exercise 3 - Colu'. At the bottom right of this list, there is a red box around the 'Import' button, which has a blue icon and the word 'Import' next to it. Above the 'Import' button, there is a red box around the '+' icon in the top right corner of the resource list.

4. Select the **Lab 2 - Explore with Spark** you downloaded in step 2 and select Open.
5. Follow the instructions contained within the notebook to complete the remainder of this task. When you are done with the notebook, return to this guide and continue with the next section.
6. Once you have completed the **Lab 2 - Explore with Spark** notebook, click on the stop session button on the far right hand side of the toolbar to release the Spark cluster for the next exercise.

Tailwind Traders has unstructured and semi-structured files from various data sources. Their data engineers want to use their Spark expertise to explore, ingest, and transform these files.

You recommend using Synapse Notebooks, which are integrated in the Azure Synapse Analytics workspace and used from within Synapse Studio.

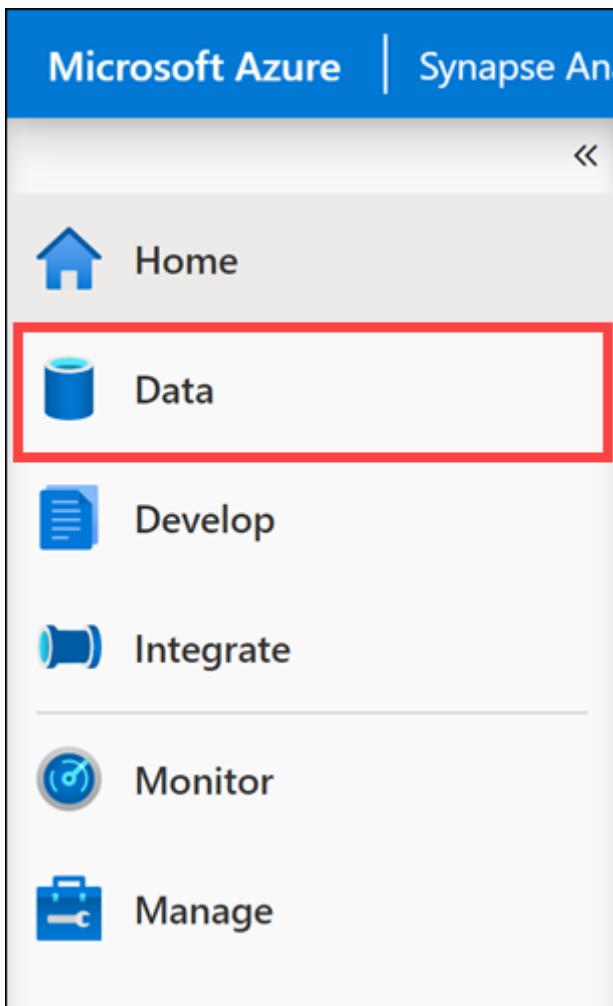
7.5 Exercise 2: Ingesting data with Spark notebooks in Azure Synapse Analytics

7.5.1 Task 1: Ingest and explore Parquet files from a data lake with Apache Spark for Azure Synapse

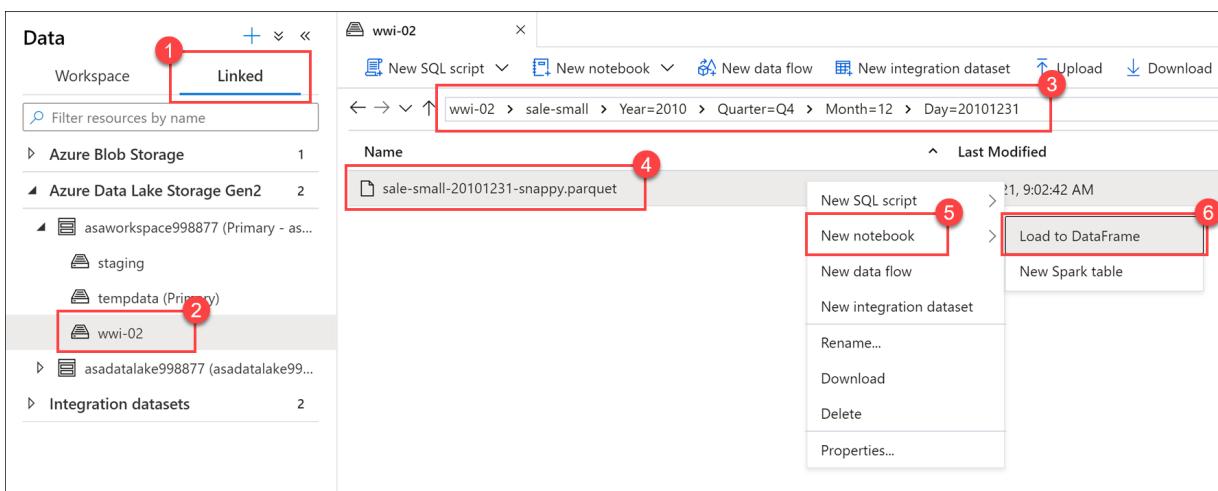
Tailwind Traders has Parquet files stored in their data lake. They want to know how they can quickly access the files and explore them using Apache Spark.

You recommend using the Data hub to view the Parquet files in the connected storage account, then use the *new notebook* context menu to create a new Synapse Notebook that loads a Spark dataframe with the contents of a selected Parquet file.

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Data** hub.

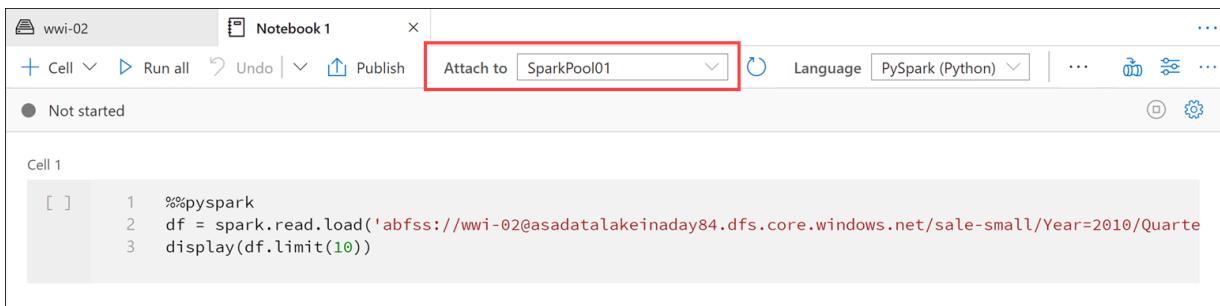


3. Select the **Linked** tab (1) and expand the **Azure Data Lake Storage Gen2** group, then expand the primary data lake storage account (*the name may differ from what you see here; it is the first storage account listed*). Select the **wwi-02** container (2) and browser to the **sale-small/Year=2010/Quarter=Q4/Month=12/Day=20101231** folder (3). Right-click the Parquet file (4), select **New notebook** (5), then select **Load to DataFrame *6**.

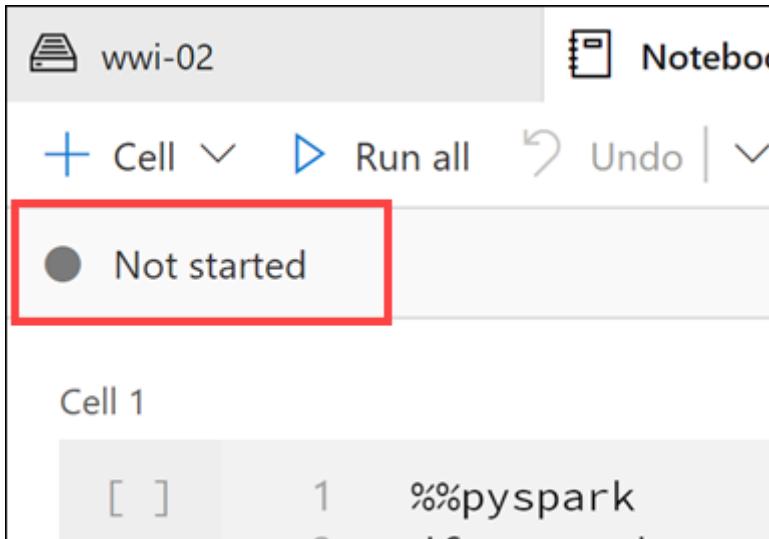


This generates a notebook with PySpark code to load the data in a Spark dataframe and display 10 rows with the header.

4. Make sure the Spark pool is attached to the notebook.



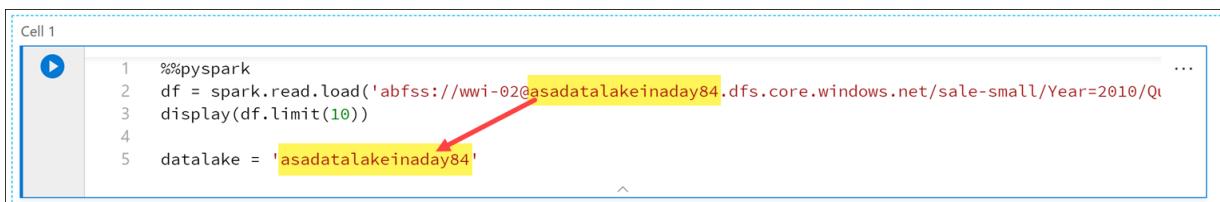
The Spark pool provides the compute for all notebook operations. If we look at the bottom of the notebook, we'll see that the pool has not started. When you run a cell in the notebook while the pool is idle, the pool will start and allocate resources. This is a one-time operation until the pool auto-pauses from being idle for too long.



The auto-pause settings are configured on the Spark pool configuration in the Manage hub.

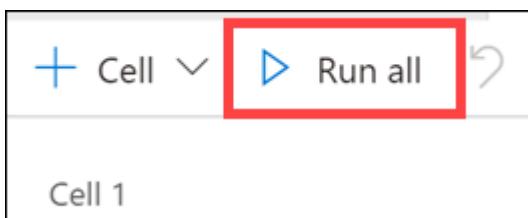
- Add the following beneath the code in the cell to define a variable named `datalake` whose value is the name of the primary storage account (**replace the REPLACE_WITH_YOUR_DATALAKE_NAME value with the name of the storage account in line 2**):

```
datalake = 'REPLACE_WITH_YOUR_DATALAKE_NAME'
```



This variable will be used in a couple cells later on.

- Select **Run all** on the notebook toolbar to execute the notebook.



Note: The first time you run a notebook in a Spark pool, Azure Synapse creates a new session. This can take approximately 3-5 minutes.

Note: To run just the cell, either hover over the cell and select the *Run cell* icon to the left of the cell, or select the cell then type **Ctrl+Enter** on your keyboard.

7. After the cell run is complete, change the View to **Chart** in the cell output.

The screenshot shows a Jupyter Notebook cell titled "Cell 1". The code cell contains the following Python code:

```

1 %%pyspark
2 df = spark.read.load('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/sale-small/Year=2010/'
3 display(df.limit(10))
4
5 datalake = 'asadatalakeinaday84'

```

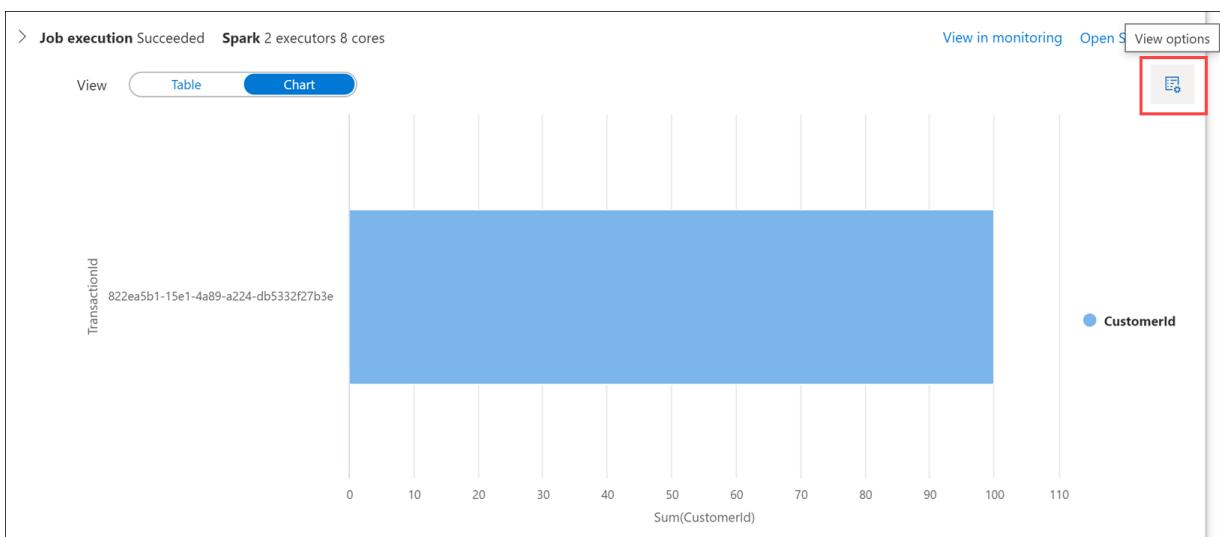
Below the code cell, a message indicates the command was executed in 3mins 7s 758ms by joel on 11-25-2020 20:14:16.976 -05:00. The job execution status is shown as "Succeeded" with 2 executors and 8 cores. There are links to "View in monitoring" and "Open Spark UI".

The main area displays a table of sales transaction data. The table has columns: TransactionId, CustomerId, ProductId, Quantity, Price, TotalAmount, TransactionDate, and ProfitA. The data shows various transactions for different products and customers, with total amounts ranging from 27.7 to 125.12.

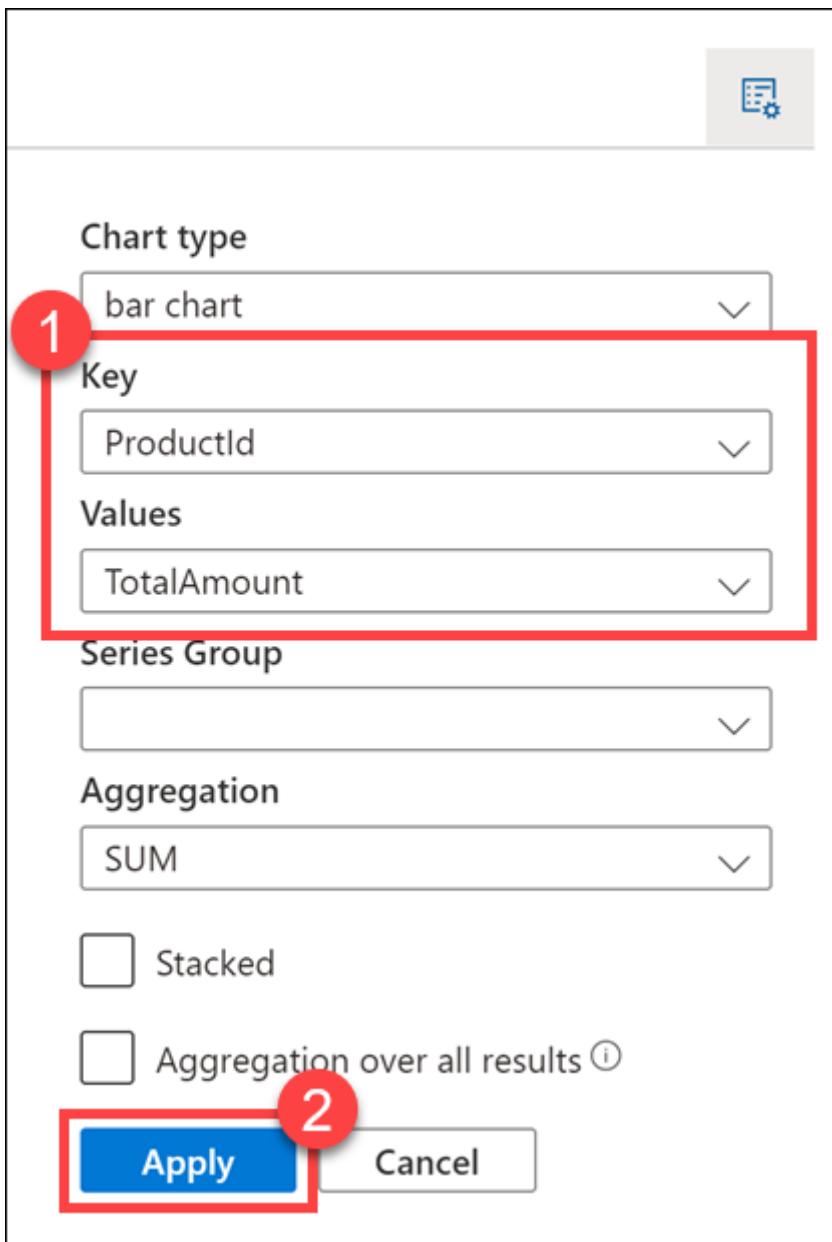
The "View" button is set to "Table", and the "Chart" button is highlighted with a red box.

By default, the cell outputs to a table view when we use the `display()` function. We see in the output the sales transaction data stored in the Parquet file for December 31, 2010. Let's select the **Chart** visualization to see a different view of the data.

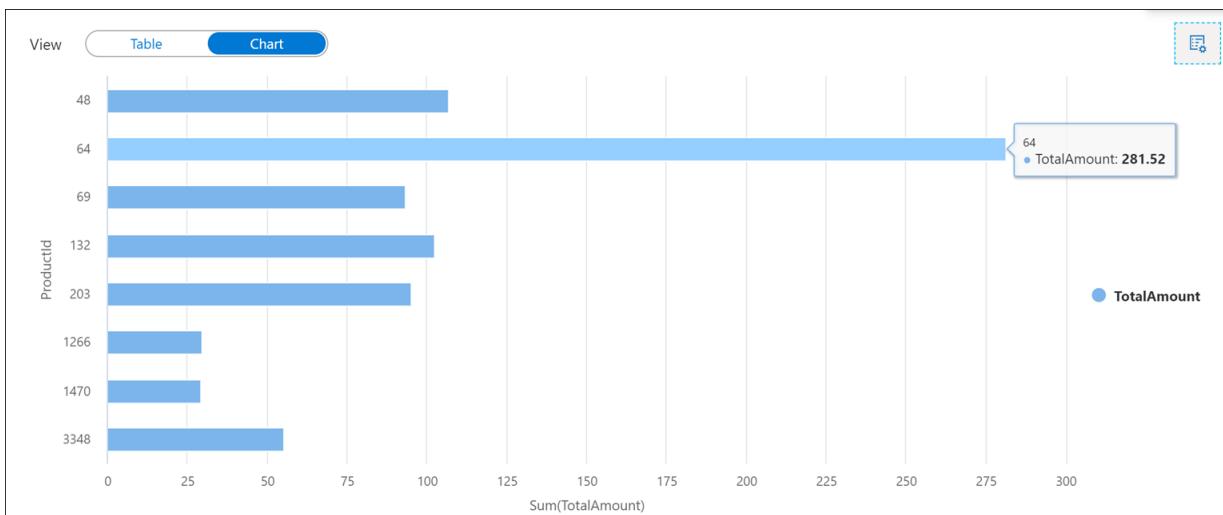
8. Select the **View options** button to the right.



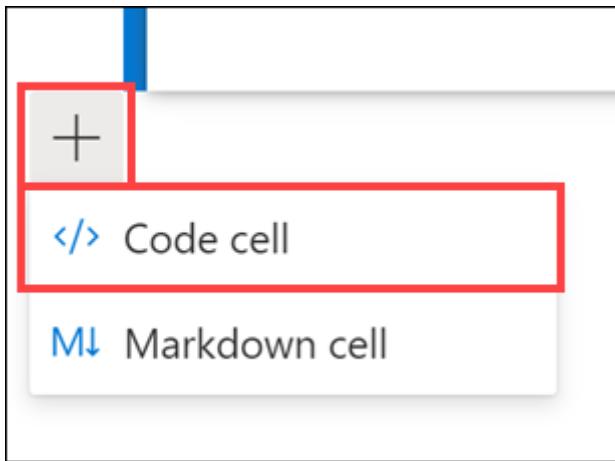
9. Set Key to **ProductId** and Values to **TotalAmount** (1), then select **Apply** (2).



10. The chart visualization is displayed. Hover over the bars to view details.



11. Create a new cell underneath by selecting +, then </> **Code cell** underneath the chart.



12. The Spark engine can analyze the Parquet files and infer the schema. To do this, enter the following in the new cell and **run** it:

```
df.printSchema()
```

Your output should look like the following:

```
root
|--- TransactionId: string (nullable = true)
|--- CustomerId: integer (nullable = true)
|--- ProductId: short (nullable = true)
|--- Quantity: short (nullable = true)
|--- Price: decimal(29,2) (nullable = true)
|--- TotalAmount: decimal(29,2) (nullable = true)
|--- TransactionDate: integer (nullable = true)
|--- ProfitAmount: decimal(29,2) (nullable = true)
|--- Hour: byte (nullable = true)
|--- Minute: byte (nullable = true)
|--- StoreId: short (nullable = true)
```

Spark evaluates the file contents to infer the schema. This automatic inference is usually sufficient for data exploration and most transformation tasks. However, when you load data to an external resource like a SQL table, sometimes you need to declare your own schema and apply that to the dataset. For now, the schema looks good.

13. Now let's use the dataframe to use aggregates and grouping operations to better understand the data. Create a new cell and enter the following, then **run** the cell:

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *

profitByDateProduct = (df.groupBy("TransactionDate", "ProductId")
    .agg(
        sum("ProfitAmount").alias("(sum)ProfitAmount"),
        round(avg("Quantity"), 4).alias("(avg)Quantity"),
        sum("Quantity").alias("(sum)Quantity"))
    .orderBy("TransactionDate"))
display(profitByDateProduct.limit(100))
```

We import required Python libraries to use aggregation functions and types defined in the schema to successfully execute the query.

The output shows the same data we saw in the chart above, but now with `sum` and `avg` aggregates (1). Notice that we use the `alias` method (2) to change the column names.

Cell 3

```

1  from pyspark.sql import SparkSession
2  from pyspark.sql.types import *
3  from pyspark.sql.functions import *
4
5  profitByDateProduct = (data_path.groupBy("TransactionDate","ProductId")
6      .agg(
7          sum("ProfitAmount").alias("(sum)ProfitAmount"),
8          round(avg("Quantity"), 4).alias("(avg)Quantity"),
9          sum("Quantity").alias("(sum)Quantity"))
10     .orderBy("TransactionDate"))
11 display(profitByDateProduct.limit(100))

```

Command executed in 6s 545ms by joel on 09-10-2020 15:13:18.982 -04:00

> Job execution Succeeded Spark 3 executors 12 cores [View in monitoring](#) [Open Spark UI](#)

View [Table](#) [Chart](#)

TransactionDate	ProductId	(sum)ProfitAmount	(avg)Quantity	(sum)Quantity
20101231	64	52975.23	2.5547	5467
20101231	3348	1409.01	2.4512	201
20101231	1470	1595.58	2.6364	174
20101231	1266	10731.05	2.5896	1199
20101231	48	39516.75	2.5158	4053
20101231	132	42468.96	2.4995	5154
20101231	66	1455.70	2.1026	1007

Ready [Stop session](#) | [Configure session](#)

7.6 Exercise 3: Transforming data with DataFrames in Spark pools in Azure Synapse Analytics

7.6.1 Task 1: Query and transform JSON data with Apache Spark for Azure Synapse

In addition to the sales data, Tailwind Traders has customer profile data from an e-commerce system that provides top product purchases for each visitor of the site (customer) over the past 12 months. This data is stored within JSON files in the data lake. They have struggled with ingesting, exploring, and transforming these JSON files and want your guidance. The files have a hierarchical structure that they want to flatten before loading into relational data stores. They also wish to apply grouping and aggregate operations as part of the data engineering process.

You recommend using Synapse Notebooks to explore and apply data transformations on the JSON files.

- Create a new cell in the Spark notebook, enter the following code and execute the cell:

```

df = (spark.read \
    .option('inferSchema', 'true') \
    .json('abfss://wwi-02@' + datalake + '.dfs.core.windows.net/online-user-profiles-02/*.json')
)

```

```

df.printSchema()

```

The `datalake` variable we created in the first cell is used here as part of the file path.

Your output should look like the following:

```

root
|-- topProductPurchases: array (nullable = true)
|   |-- element: struct (containsNull = true)
|   |   |-- itemsPurchasedLast12Months: long (nullable = true)
|   |   |-- productId: long (nullable = true)
|-- visitorId: long (nullable = true)

```

Notice that we are selecting all JSON files within the `online-user-profiles-02` directory. Each JSON file contains several rows, which is why we specified the `multiLine=True` option. Also, we set the `inferSchema` option to `true`, which instructs the Spark engine to review the files and create a schema based on the nature of the data.

2. We have been using Python code in these cells up to this point. If we want to query the files using SQL syntax, one option is to create a temporary view of the data within the dataframe. Execute the following in a new cell to create a view named `user_profiles`:

```
# create a view called user_profiles
df.createOrReplaceTempView("user_profiles")
```

3. Create a new cell. Since we want to use SQL instead of Python, we use the `%%sql` magic to set the language of the cell to SQL. Execute the following code in the cell:

```
%%sql
```

```
SELECT * FROM user_profiles LIMIT 10
```

Notice that the output shows nested data for `topProductPurchases`, which includes an array of `productId` and `itemsPurchasedLast12Months` values. You can expand the fields by clicking the right triangle in each row.

```

1 %%sql
2
3 SELECT * FROM user_profiles LIMIT 10

```

Command executed in 12s 994ms by joel on 09-10-2020 15:50:03.604 -04:00

> **Job execution** Succeeded **Spark** 3 executors 12 cores

View Table Chart

topProductPurchases	visitorId
▶ [{"schema": [{"name": "itemsPurcha"}]	117000
▶ [{"schema": [{"name": "itemsPurcha"}]	117001
▶ ▶ [{"schema": [{"name": "itemsPurcha"}]	117002
▶ ▶ ▶ 0: {"schema": [{"name": "itemsPui"}]	
▶ ▶ ▶ 1: {"schema": [{"name": "itemsPui"}]	
▶ ▶ ▶ schema: [{"name": "itemsPurcha"}]	
▶ ▶ ▶ ▶ 0: {"name": "itemsPurchasedLast12Mo	
▶ ▶ ▶ ▶ name: "itemsPurchasedLast12Mo	
▶ ▶ ▶ ▶ dataType: "{}"	
▶ ▶ ▶ ▶ nullable: "true"	
▶ ▶ ▶ ▶ ▼ metadata: {"map": {}}	
▶ ▶ ▶ ▶ ▶ map: "{}"	
▶ ▶ ▶ ▶ ▶ 1: {"name": "productId", "dat	
▶ ▶ ▶ ▶ ▶ values: "[15,2515]"	
▶ ▶ ▶ ▶ 2: {"schema": [{"name": "itemsPui"}]	
▶ ▶ ▶ ▶ 3: {"schema": [{"name": "itemsPui"}]	
▶ [{"schema": [{"name": "itemsPurcha"}]	117003

This makes analyzing the data a bit difficult. This is because the JSON file contents look like the following:

```
[
{
  "visitorId": 9529082,
  "topProductPurchases": [
    {
      "productId": 4679,
      "itemsPurchasedLast12Months": 26
    },
    {
      "productId": 1779,
      "itemsPurchasedLast12Months": 32
    }
  ]
}
```

```

},
{
  "productId": 2125,
  "itemsPurchasedLast12Months": 75
},
{
  "productId": 2007,
  "itemsPurchasedLast12Months": 39
},
{
  "productId": 1240,
  "itemsPurchasedLast12Months": 31
},
{
  "productId": 446,
  "itemsPurchasedLast12Months": 39
},
{
  "productId": 3110,
  "itemsPurchasedLast12Months": 40
},
{
  "productId": 52,
  "itemsPurchasedLast12Months": 2
},
{
  "productId": 978,
  "itemsPurchasedLast12Months": 81
},
{
  "productId": 1219,
  "itemsPurchasedLast12Months": 56
},
{
  "productId": 2982,
  "itemsPurchasedLast12Months": 59
}
]
},
{
  ...
},
{
  ...
}
]

```

4. PySpark contains a special `explode` function, which returns a new row for each element of the array. This will help flatten the `topProductPurchases` column for better readability or for easier querying. Execute the following in a new cell:

```

from pyspark.sql.functions import udf, explode

flat=df.select('visitorId', explode('topProductPurchases').alias('topProductPurchases_flat'))
flat.show(100)

```

In this cell, we created a new dataframe named `flat` that includes the `visitorId` field and a new aliased field named `topProductPurchases_flat`. As you can see, the output is a bit easier to read and, by extension, easier to query.

Cell 7

```

1   from pyspark.sql.functions import udf, explode
2
3   flat=df.select('visitorId',explode('topProductPurchases').alias('topProductPurchases_flat'))
4   flat.show(100)

```

Command executed in 2s 400ms by joel on 09-10-2020 15:55:17.768 -04:00

> Job execution Succeeded Spark 3 executors 12 cores

visitorId	topProductPurchases_flat
117000	[13, 3623]
117000	[5, 2321]
117001	[93, 713]
117001	[19, 2144]
117001	[30, 1094]
117001	[82, 3223]
117001	[42, 3328]
117001	[62, 2926]
117001	[63, 2651]
117001	[39, 341]
117001	[85, 4841]
117001	[67, 4289]
117001	[42, 1264]
117001	[43, 3608]
117001	[14, 504]
117001	[97, 2649]
117001	[44, 2873]
117001	[7, 4491]

5. Create a new cell and execute the following code to create a new flattened version of the dataframe that extracts the `topProductPurchases_flat.productId` and `topProductPurchases_flat.itemsPurchasedLast12Months` fields to create new rows for each data combination:

```
topPurchases = (flat.select('visitorId','topProductPurchases_flat.productId','topProductPurchases_flat.itemsPurchasedLast12Months')
    .orderBy('visitorId'))
```

```
topPurchases.show(100)
```

In the output, notice that we now have multiple rows for each `visitorId`.

Cell 8

```

1   topPurchases = (flat.select('visitorId','topProductPurchases_flat.productId','topProductPurchases_flat.itemsPurchasedLast12Months')
2   |   .orderBy('visitorId'))
3
4   topPurchases.show(100)

```

Command executed in 3s 712ms by joel on 09-10-2020 15:59:40.419 -04:00

> Job execution Succeeded Spark 3 executors 12 cores

visitorId	productId	itemsPurchasedLast12Months
80000	4198	92
80000	2488	31
80000	4136	30
80000	1362	18
80000	3122	33
80000	3270	5
80000	93	86
80000	102	42
80000	4206	21
80000	3538	65
80000	4745	38
80000	291	49
80000	290	83
80000	4074	48
80000	4024	35
80000	2481	63
80000	2859	54
80000	2069	93
80000	1330	78
80000	3794	90
80001	4105	11
80001	2249	75
80001	4684	9
80001	3729	56

6. Let's order the rows by the number of items purchased in the last 12 months. Create a new cell and

execute the following code:

```
# Let's order by the number of items purchased in the last 12 months
sortedTopPurchases = topPurchases.orderBy("itemsPurchasedLast12Months")

display(sortedTopPurchases.limit(100))
```

Cell 9

1 # Let's order by the number of items purchased in the last 12 months
2 sortedTopPurchases = topPurchases.orderBy("itemsPurchasedLast12Months")
3
4 display(sortedTopPurchases.limit(100))

Command executed in 6s 13ms by joel on 09-10-2020 16:02:31.831 -04:00

> Job execution Succeeded Spark 3 executors 12 cores

View **Table** Chart

visitorId	productId	itemsPurchasedLast12Months
118878	2895	1
88702	3331	1
118888	4497	1
118027	3405	1
118900	4338	1
118068	661	1
118900	4062	1
118088	4394	1
118906	226	1
118112	3509	1
118917	601	1

7. How do we sort in reverse order? One might conclude that we could make a call like this:
`topPurchases.orderBy("itemsPurchasedLast12Months desc")`. Try it in a new cell:

```
topPurchases.orderBy("itemsPurchasedLast12Months desc")
```

Cell 10

```
1 topPurchases.orderBy("itemsPurchasedLast12Months desc")
```

Command executed in 3s 488ms by joel on 09-10-2020 16:04:59.708 -04:00

AnalysisException : cannot resolve 'itemsPurchasedLast12Months desc' given input columns: [visitorId, productId, itemsPurchasedLast12Months];; 'Sort [itemsPurchasedLast12Months desc ASC NULLS FIRST], true
+- Sort [visitorId#394L ASC NULLS FIRST], true
 +- Project [visitorId#394L, topProductPurchases_flat#442.productId AS productId#454L, topProductPurchases_flat#442.itemsPurchasedLast12Months AS itemsPurchasedLast12Months#455L]
 +- Project [visitorId#394L, topProductPurchases_flat#442]
 +- Generate explode(topProductPurchases#393), false, [topProductPurchases_flat#442]
 +- Relation[topProductPurchases#393, visitorId#394L] json
Traceback (most recent call last):
File "/opt/spark/python/lib/pyspark.zip/pyspark/sql/dataframe.py", line 1098, in sort
 _jdf = self._jdf.sort(self._sort_cols, kwargs))
File "/opt/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 1257, in __call__
 answer, self.gateway_client, self.target_id, self.name)
File "/opt/spark/python/lib/pyspark.zip/pyspark/sql/utils.py", line 75, in deco
 raise AnalysisException(s.split(': ', 1)[1], stackTrace)
pyspark.sql.utils.AnalysisException: cannot resolve 'itemsPurchasedLast12Months desc' given input columns: [visitorId, productId, itemsPurchasedLast12Months..]

Notice that there is an `AnalysisException` error, because `itemsPurchasedLast12Months desc` does not match up with a column name.

Why does this not work?

- The `DataFrames` API is built upon an SQL engine.
- There is a lot of familiarity with this API and SQL syntax in general.
- The problem is that `orderBy(..)` expects the name of the column.
- What we specified was an SQL expression in the form of `requests desc`.
- What we need is a way to programmatically express such an expression.
- This leads us to the second variant, `orderBy(Column)` and more specifically, the class `Column`.

8. The `Column` class is an object that encompasses more than just the name of the column, but also column-level-transformations, such as sorting in a descending order. Execute the following code in a new cell:

```
sortedTopPurchases = (topPurchases
    .orderBy( col("itemsPurchasedLast12Months").desc() ))
```

```
display(sortedTopPurchases.limit(100))
```

Notice that the results are now sorted by the `itemsPurchasedLast12Months` column in descending order, thanks to the `desc()` method on the `col` object.

Cell 11

```
1 sortedTopPurchases = (topPurchases
2     .orderBy( col("itemsPurchasedLast12Months").desc() ))
3
4 display(sortedTopPurchases.limit(100))
```

Command executed in 5s 827ms by joel on 09-10-2020 16:10:15.556 -04:00

> Job execution Succeeded Spark 3 executors 12 cores



View

Table

Chart

visitorId	productId	itemsPurchasedLast12Months
84884	2834	99
101990	835	99
84902	1482	99
84011	340	99
84906	139	99
84024	3876	99
84915	4748	99
84060	484	99
84934	1359	99
84066	4467	99

9. How many *types* of products did each customer purchase? To figure this out, we need to group by `visitorId` and aggregate on the number of rows per customer. Execute the following code in a new cell:

```
groupedTopPurchases = (sortedTopPurchases.select("visitorId")
    .groupBy("visitorId")
    .agg(count("*").alias("total"))
    .orderBy("visitorId") )

display(groupedTopPurchases.limit(100))
```

Notice how we use the `groupBy` method on the `visitorId` column, and the `agg` method over a count of records to display the total for each customer.

Cell 12

```
1 groupedTopPurchases = (sortedTopPurchases.select("visitorId")
2     .groupBy("visitorId")
3     .agg(count("*").alias("total"))
4     .orderBy("visitorId") )
5
6 display(groupedTopPurchases.limit(100))
```

Command executed in 5s 21ms by joel on 09-10-2020 16:16:23.042 -04:00

> **Job execution** Succeeded **Spark** 3 executors 12 cores



View

Table

Chart

visitorId	total
80000	20
80001	20
80002	15
80003	12
80004	10
80005	13
80006	6
80007	18
80008	4

10. How many *total items* did each customer purchase? To figure this out, we need to group by `visitorId` and aggregate on the sum of `itemsPurchasedLast12Months` values per customer. Execute the following code in a new cell:

```
groupedTopPurchases = (sortedTopPurchases.select("visitorId", "itemsPurchasedLast12Months")
    .groupBy("visitorId")
    .agg(sum("itemsPurchasedLast12Months").alias("totalItemsPurchased"))
    .orderBy("visitorId") )

display(groupedTopPurchases.limit(100))
```

Here we group by `visitorId` once again, but now we use a `sum` over the `itemsPurchasedLast12Months` column in the `agg` method. Notice that we included the `itemsPurchasedLast12Months` column in the `select` statement so we could use it in the `sum`.

Cell 13

```

1 groupedTopPurchases = (sortedTopPurchases.select("visitorId", "itemsPurchasedLast12Months")
2     .groupBy("visitorId")
3     .agg(sum("itemsPurchasedLast12Months").alias("totalItemsPurchased"))
4     .orderBy("visitorId") )
5
6 display(groupedTopPurchases.limit(100))

```

Command executed in 5s 227ms by joel on 09-10-2020 16:21:05.194 -04:00

> Job execution Succeeded Spark 3 executors 12 cores

View Table Chart

visitorId	totalItemsPurchased
80000	1054
80001	834
80002	754
80003	684
80004	598
80005	615
80006	348
80007	932
80008	199

7.7 Exercise 4: Integrating SQL and Spark pools in Azure Synapse Analytics

Tailwind Traders wants to write to the SQL database associated with dedicated SQL pool after performing data engineering tasks in Spark, then reference that SQL database as a source for joining with Spark dataframes that contain data from other files.

You decide to use the Apache Spark to Synapse SQL connector to efficiently transfer data between Spark databases and SQL databases in Azure Synapse.

Transferring data between Spark databases and SQL databases can be done using JDBC. However, given two distributed systems such as Spark pools and SQL pools, JDBC tends to be a bottleneck with serial data transfer.

The Apache Spark pool to Synapse SQL connector is a data source implementation for Apache Spark. It uses the Azure Data Lake Storage Gen2 and PolyBase in dedicated SQL pools to efficiently transfer data between the Spark cluster and the Synapse SQL instance.

7.7.1 Task 1: Update notebook

- We have been using Python code in these cells up to this point. If we want to use the Apache Spark pool to Synapse SQL connector (`sqlanalytics`), one option is to create a temporary view of the data within the dataframe. Execute the following in a new cell to create a view named `top_purchases`:

```
# Create a temporary view for top purchases so we can load from Scala
topPurchases.createOrReplaceTempView("top_purchases")
```

We created a new temporary view from the `topPurchases` dataframe that we created earlier and which contains the flattened JSON user purchases data.

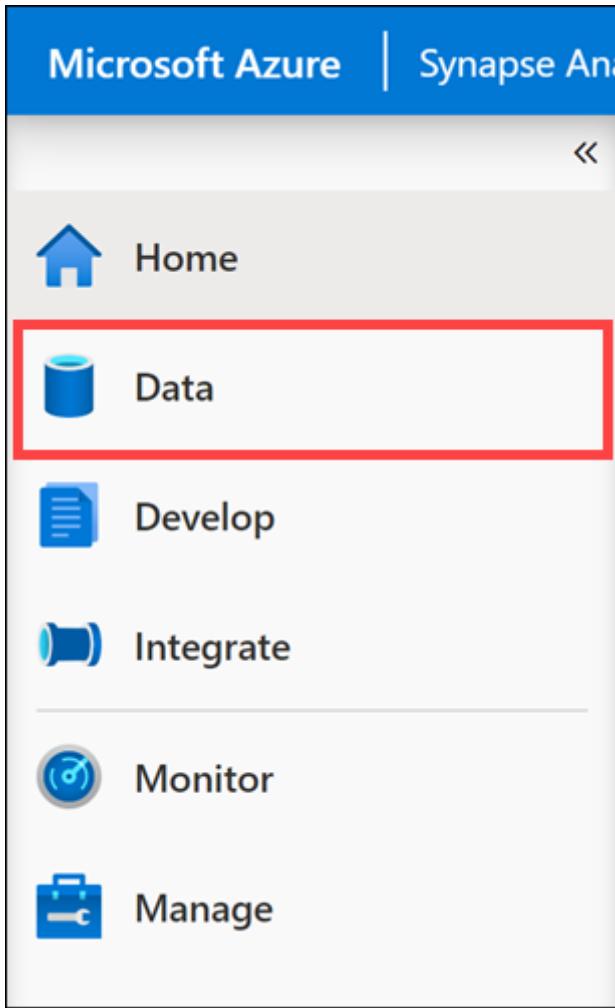
- We must execute code that uses the Apache Spark pool to Synapse SQL connector in Scala. To do this, we add the `%spark` magic to the cell. Execute the following in a new cell to read from the `top_purchases` view:

```
%%spark
// Make sure the name of the dedicated SQL pool (SQLPool01 below) matches the name of your SQL pool
val df = spark.sqlContext.sql("select * from top_purchases")
df.write.sqlAnalytics("SQLPool01.wwi.TopPurchases", Constants.INTERNAL)
```

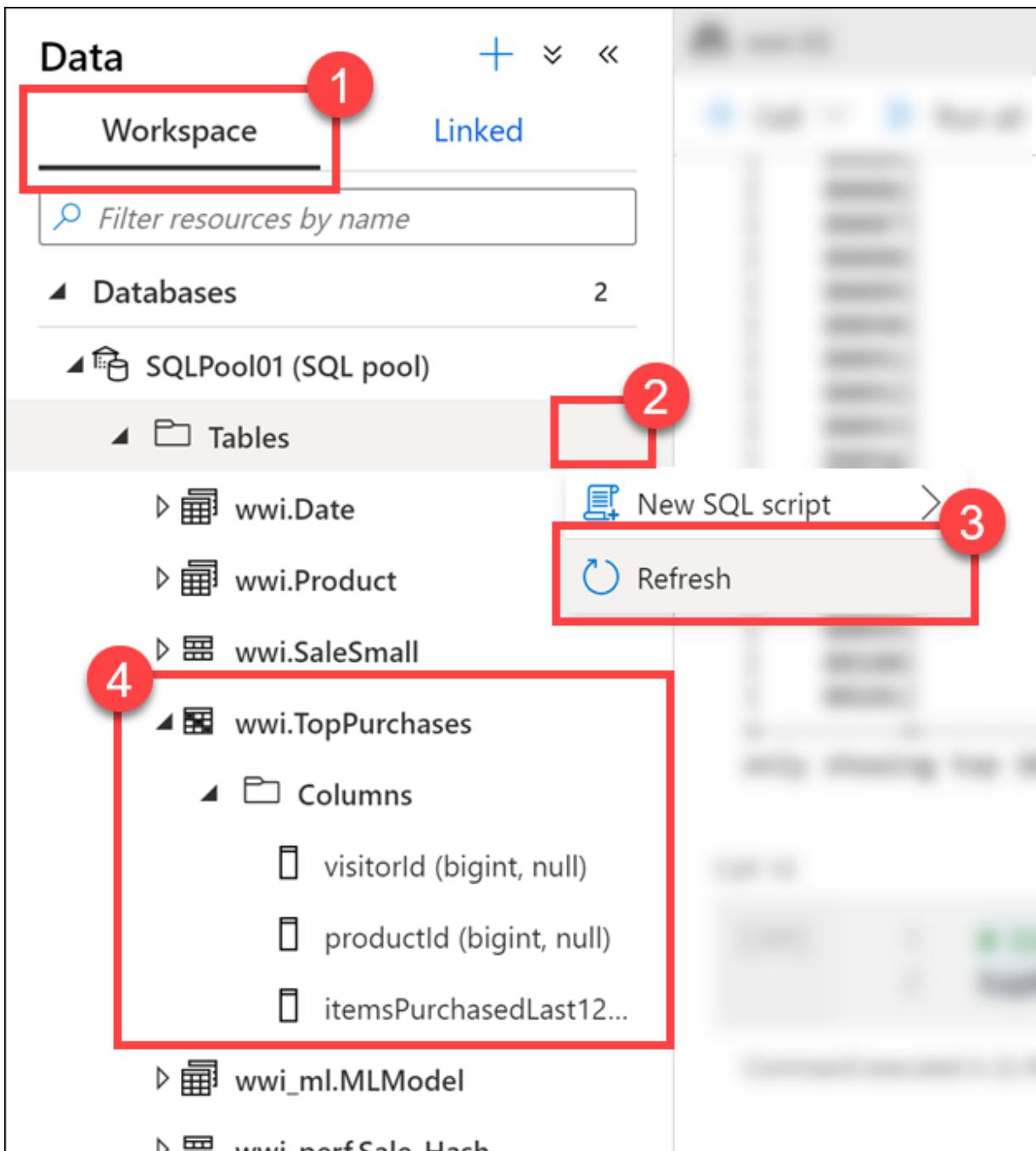
Note: The cell may take over a minute to execute. If you have run this command before, you will receive an error stating that "There is already an object named.." because the table already exists.

After the cell finishes executing, let's take a look at the list of SQL tables to verify that the table was successfully created for us.

3. Leave the notebook open, then navigate to the **Data** hub (if not already selected).



4. Select the **Workspace** tab (1), expand the SQL database, select the **ellipses (...)** on Tables (2) and select **Refresh (3)**. Expand the **wwi.TopPurchases** table and columns (4).



As you can see, the `wwi.TopPurchases` table was automatically created for us, based on the derived schema of the Spark dataframe. The Apache Spark pool to Synapse SQL connector was responsible for creating the table and efficiently loading the data into it.

5. **Return to the notebook** and execute the following in a new cell to read sales data from all the Parquet files located in the `sale-small/Year=2019/Quarter=Q4/Month=12/` folder:

```
dfsales = spark.read.load('abfss://wwi-020' + datalake + '.dfs.core.windows.net/sale-small/Year=2019/Quarter=Q4/Month=12/')
display(dfsales.limit(10))
```

Note: It can take over 3 minutes for this cell to execute.

The `datalake` variable we created in the first cell is used here as part of the file path.

```

Cell 16
1 dfsales = spark.read.load('abfss://ww1-02@' + datalake + '.dfs.core.windows.net/sale-small/year=2019/Quarter=Q4/Month=12/*.parquet', format='parquet') ...
2 display(dfsales.limit(10))

Command executed in 4s 570ms by joel on 09-10-2020 21:12:35.868 -04:00
> Job execution Succeeded Spark 3 executors 12 cores
View in monitoring Open Spark UI

```

TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount
0ce3b96c-2553-46db-823c-158d...	5	1327	3	28.77	86.31
0ce3b96c-2553-46db-823c-158d...	5	4257	2	22.95	45.9
0ce3b96c-2553-46db-823c-158d...	5	3388	2	19.88	39.76
0ce3b96c-2553-46db-823c-158d...	5	3861	2	27.86	55.72
0ce3b96c-2553-46db-823c-158d...	5	47	3	22.06	66.18
0ce3b96c-2553-46db-823c-158d...	5	214	1	36.24	36.24
0ce3b96c-2553-46db-823c-158d...	5	195	1	24.14	24.14
0ce3b96c-2553-46db-823c-158d...	5	134	3	29.16	87.48
0ce3b96c-2553-46db-823c-158d...	5	59	2	29.03	58.06
8469dff5-8251-4a3f-8550-5c532...	14	4213	3	37.87	113.61

Compare the file path in the cell above to the file path in the first cell. Here we are using a relative path to load **all December 2019 sales** data from the Parquet files located in `sale-small`, vs. just December 31, 2010 sales data.

Next, let's load the `TopSales` data from the SQL table we created earlier into a new Spark dataframe, then join it with this new `dfsales` dataframe. To do this, we must once again use the `%spark` magic on a new cell since we'll use the Apache Spark pool to Synapse SQL connector to retrieve data from the SQL database. Then we need to add the dataframe contents to a new temporary view so we can access the data from Python.

6. Execute the following in a new cell to read from the `TopSales` SQL table and save it to a temporary view:

```

%%spark
// Make sure the name of the SQL pool (SQLPool01 below) matches the name of your SQL pool.
val df2 = spark.read.sqlanalytics("SQLPool01.wwi.TopPurchases")
df2.createTempView("top_purchases_sql")

df2.head(10)

```

```

Cell 17
1 %%spark
2 // Make sure the name of the SQL pool (SQLPool01 below) matches the name of your SQL pool.
3 val df2 = spark.read.sqlanalytics("SQLPool01.wwi.TopPurchases")
4 df2.createTempView("top_purchases_sql")
5 df2.head(10)

Command executed in 10s 658ms by joel on 09-10-2020 20:28:20.300 -04:00
> Job execution Succeeded Spark 3 executors 12 cores
View in monitoring Open Spark UI

```

visitorId	productId	...
[112814,4817,81]	[112814,4336,91]	[112814,874,18]
[112814,128,96]	[112814,1184,53]	[112814,1807,90]
[112814,2973,60]	[112814,1852,83]	[112814,1191,70]
[112814,4845,4]		

The cell's language is set to Scala by using the `%spark` magic (1) at the top of the cell. We declared a new variable named `df2` as a new DataFrame created by the `spark.read.sqlanalytics` method, which reads from the `TopPurchases` table (2) in the SQL database. Then we populated a new temporary view named `top_purchases_sql` (3). Finally, we showed the first 10 records with the `df2.head(10)` line (4). The cell output displays the dataframe values (5).

7. Execute the following in a new cell to create a new dataframe in Python from the `top_purchases_sql` temporary view, then display the first 10 results:

```

dfTopPurchasesFromSql = sqlContext.table("top_purchases_sql")

display(dfTopPurchasesFromSql.limit(10))

```

Cell 18

```

1 dfTopPurchasesFromSql = sqlContext.table("top_purchases_sql")
2
3 display(dfTopPurchasesFromSql.limit(10))

```

Command executed in 10s 320ms by joel on 09-10-2020 20:40:45.435 -04:00

> Job execution Succeeded Spark 3 executors 12 cores

[View in monitoring](#) [Open Spark UI](#)

View [Table](#) [Chart](#)

visitorId	productId	itemsPurchasedLast12Months
118119	886	83
118119	2284	61
118119	1353	20
118119	4258	91
118119	2197	97
118119	3351	80
118119	3138	34
118119	2024	54
118120	2177	20
118120	831	28

8. Execute the following in a new cell to join the data from the sales Parquet files and the TopPurchases SQL database:

```

inner_join = dfsales.join(dfTopPurchasesFromSql,
    (dfsales.CustomerId == dfTopPurchasesFromSql.visitorId) & (dfsales.ProductId == dfTopPurchasesFromSql.productId))

inner_join_agg = (inner_join.select("CustomerId", "TotalAmount", "Quantity", "itemsPurchasedLast12Months")
    .groupBy(["CustomerId", "top_purchases_sql.productId"])
    .agg(
        sum("TotalAmount").alias("TotalAmountDecember"),
        sum("Quantity").alias("TotalQuantityDecember"),
        sum("itemsPurchasedLast12Months").alias("TotalItemsPurchasedLast12Months"))
    .orderBy("CustomerId"))

display(inner_join_agg.limit(100))

```

In the query, we joined the `dfsales` and `dfTopPurchasesFromSql` dataframes, matching on `CustomerId` and `ProductId`. This join combined the `TopPurchases` SQL table data with the December 2019 sales Parquet data (1).

We grouped by the `CustomerId` and `ProductId` fields. Since the `ProductId` field name is ambiguous (it exists in both dataframes), we had to fully-qualify the `ProductId` name to refer to the one in the `TopPurchases` dataframe (2).

Then we created an aggregate that summed the total amount spent on each product in December, the total number of product items in December, and the total product items purchased in the last 12 months (3).

Finally, we displayed the joined and aggregated data in a table view.

Note: Feel free to click on the column headers in the Table view to sort the result set.

Cell 20

```

1 inner_join = dfsales.join(dfTopPurchasesFromSql,
2     (dfsales.CustomerId == dfTopPurchasesFromSql.visitorId) & (dfsales.ProductId == dfTopPurchasesFromSql.productId))
3
4 inner_join_agg = (inner_join.select("CustomerId", "TotalAmount", "Quantity", "itemsPurchasedLast12Months", "top_purchases_sql.productId")
5     .groupBy(["CustomerId", "top_purchases_sql.productId"]))
6     .agg(
7         sum("TotalAmount").alias("TotalAmountDecember"),
8         sum("Quantity").alias("TotalQuantityDecember"),
9         sum("itemsPurchasedLast12Months").alias("TotalItemsPurchasedLast12Months"))
10    .orderBy("CustomerId"))
11
12 display(inner_join_agg.limit(100))
```

Command executed in 40s 941ms by joel on 09-10-2020 21:40:40.798 -04:00

> Job execution Succeeded Spark 3 executors 12 cores View in monitor

View Table Chart

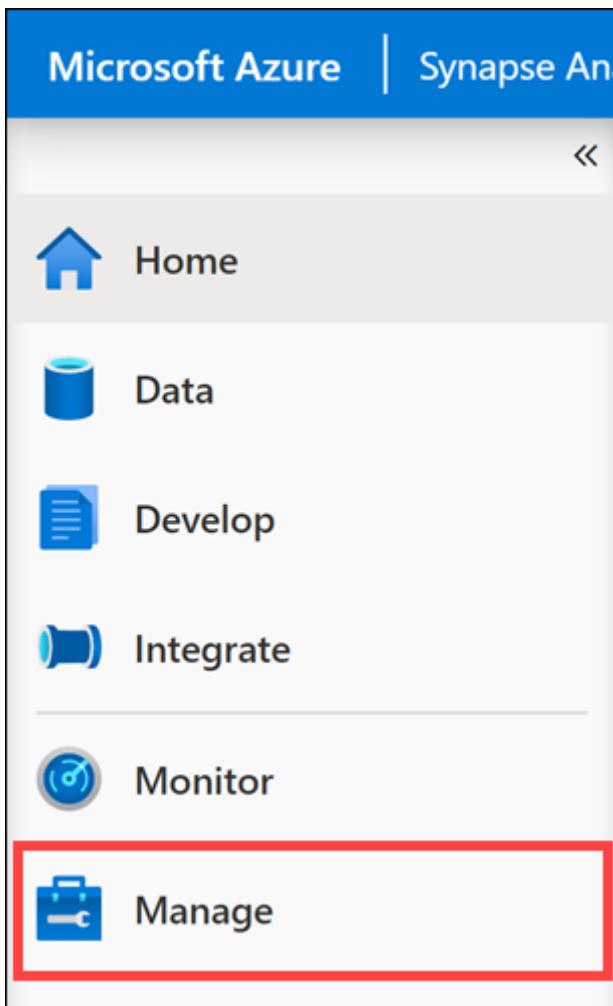
CustomerId	productId	TotalAmountDecember	TotalQuantityDecember	TotalItemsPurchasedLast12Months
80034	70	34.32	1	73
80097	1160	105.04	4	46
80126	30	62.4	3	95
80145	4475	21.66	1	40
80167	4097	69.99	3	23
80168	169	77.58	2	45

7.8 Exercise 5: Cleanup

Complete these steps to free up resources you no longer need.

7.8.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



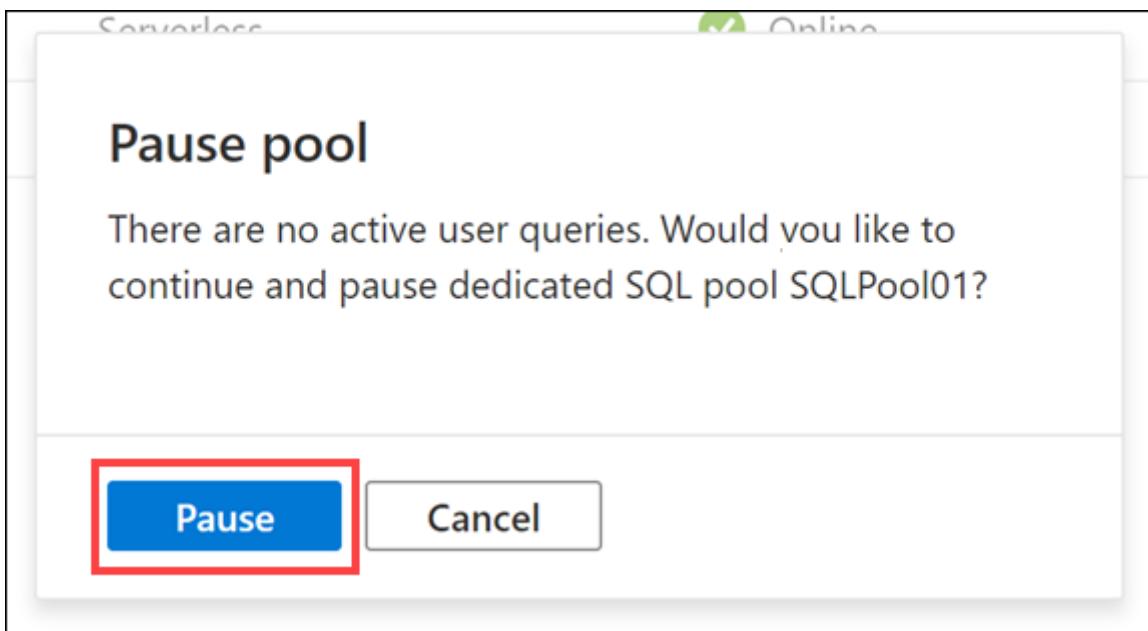
3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

The screenshot shows the 'SQL pools' blade in the Azure portal. The left sidebar lists various pool types: Analytics pools (with 'SQL pools' highlighted by a red box and a red number 1), External connections, Integration, and Security. The main area displays the 'SQL pools' table with the following data:

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

A red box highlights the 'Pause' button for the 'SQLPool01' row, and a red number 2 is placed above it.

4. When prompted, select **Pause**.



8 Module 6 - Data exploration and transformation in Azure Databricks

This module teaches how to use various Apache Spark DataFrame methods to explore and transform data in Azure Databricks. The student will learn how to perform standard DataFrame methods to explore and transform data. They will also learn how to perform more advanced tasks, such as removing duplicate data, manipulate date/time values, rename columns, and aggregate data.

In this module the student will be able to:

- Use DataFrames in Azure Databricks to explore and filter data
- Cache a DataFrame for faster subsequent queries
- Remove duplicate data
- Manipulate date/time values
- Remove and rename DataFrame columns
- Aggregate data stored in a DataFrame

8.1 Lab details

- [Module 6 - Data exploration and transformation in Azure Databricks](#)
 - [Lab details](#)
 - [Lab 1 - Working with DataFrames](#)
 - * [Before the hands-on lab](#)
 - [Task 1 - Create and configure the Azure Databricks workspace](#)
 - * [Exercise 1: Complete the lab notebook](#)
 - [Task 1: Clone the Databricks archive](#)
 - [Task 2: Complete the Describe a DataFrame notebook](#)
 - * [Exercise 2: Complete the Working with DataFrames notebook](#)
 - * [Exercise 3: Complete the Display Function notebook](#)
 - * [Exercise 4: Complete the Distinct Articles exercise notebook](#)
 - [Lab 2 - Working with DataFrames advanced methods](#)
 - * [Exercise 2: Complete the lab notebook](#)
 - [Task 1: Clone the Databricks archive](#)
 - [Task 2: Complete the Date and Time Manipulation notebook](#)
 - * [Exercise 3: Complete the Use Aggregate Functions notebook](#)
 - * [Exercise 4: Complete the De-Duping Data exercise notebook](#)

8.2 Lab 1 - Working with DataFrames

Your data processing in Azure Databricks is accomplished by defining DataFrames to read and process the Data. This lab will introduce how to read your data using Azure Databricks DataFrames. You need to complete the exercises within Databricks Notebooks. To begin, you need to have access to an Azure Databricks workspace. If you do not have a workspace available, follow the instructions below. Otherwise, you can skip ahead to the **Clone the Databricks archive** step.

8.2.1 Before the hands-on lab

Note: Only complete the **Before the hands-on lab** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 1.

Before stepping through the exercises in this lab, make sure you have access to an Azure Databricks workspace with an available cluster. Perform the tasks below to configure the workspace.

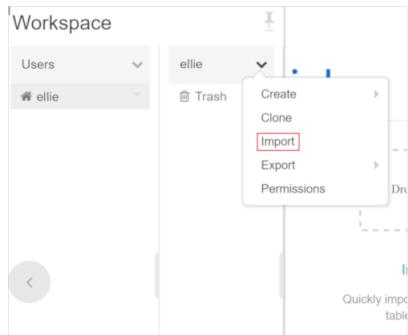
8.2.1.1 Task 1 - Create and configure the Azure Databricks workspace

Follow the [lab 06 setup instructions](#) to create and configure the workspace.

8.2.2 Exercise 1: Complete the lab notebook

8.2.2.1 Task 1: Clone the Databricks archive

1. If you do not currently have your Azure Databricks workspace open: in the Azure portal, navigate to your deployed Azure Databricks workspace and select **Launch Workspace**.
2. In the left pane, select **Workspace > Users**, and select your username (the entry with the house icon).
3. In the pane that appears, select the arrow next to your name, and select **Import**.



4. In the **Import Notebooks** dialog box, select the URL and paste in the following URL:

<https://github.com/solliancenet/microsoft-learning-paths-databricks-notebooks/blob/master/data-engineer>

1. Select **Import**.
2. Select the **04-Working-With-Dataframes** folder that appears.

8.2.2.2 Task 2: Complete the Describe a DataFrame notebook

Open the **1.Describe-a-dataframe** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will:

- Develop familiarity with the `DataFrame` APIs
- Learn the classes...
 - `SparkSession`
 - `DataFrame` (aka `Dataset[Row]`)
- Learn the action...
 - `count()`

After you've completed the notebook, return to this screen, and continue to the next step.

8.2.3 Exercise 2: Complete the Working with DataFrames notebook

In your Azure Databricks workspace, open the **04-Working-With-Dataframes** folder that you imported within your user folder.

Open the **2.Use-common-dataframe-methods** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will:

- Develop familiarity with the `DataFrame` APIs
- Use common DataFrame methods for performance
- Explore the Spark API documentation

After you've completed the notebook, return to this screen, and continue to the next step.

8.2.4 Exercise 3: Complete the Display Function notebook

In your Azure Databricks workspace, open the **04-Working-With-Dataframes** folder that you imported within your user folder.

Open the **3.Display-function** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will:

- Learn the transformations...
 - `limit(..)`
 - `select(..)`
 - `drop(..)`
 - `distinct()`
 - `dropDuplicates(..)`
- Learn the actions...
 - `show(..)`
 - `display(..)`

After you've completed the notebook, return to this screen, and continue to the next step.

8.2.5 Exercise 4: Complete the Distinct Articles exercise notebook

In your Azure Databricks workspace, open the **04-Working-With-Dataframes** folder that you imported within your user folder.

Open the **4.Exercise: Distinct Articles** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

In this exercise, you read Parquet files, apply necessary transformations, perform a total count of records, then verify that all the data was correctly loaded. As a bonus, try defining a schema that matches the data and update the read operation to use the schema.

Note: You will find a corresponding notebook within the `Solutions` subfolder. This contains completed cells for the exercise. Refer to the notebook if you get stuck or simply want to see the solution.

After you've completed the notebook, return to this screen, and continue to the next lab.

8.3 Lab 2 - Working with DataFrames advanced methods

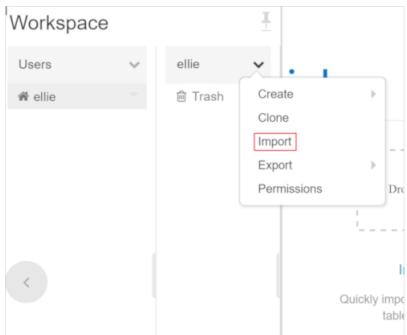
This lab builds on the Azure Databricks DataFrames concepts learned in the previous lab above by exploring some advanced methods data engineers can use to read, write, and transform data using DataFrames.

8.3.1 Exercise 2: Complete the lab notebook

8.3.1.1 Task 1: Clone the Databricks archive

1. If you do not currently have your Azure Databricks workspace open: in the Azure portal, navigate to your deployed Azure Databricks workspace and select **Launch Workspace**.
2. In the left pane, select **Workspace > Users**, and select your username (the entry with the house icon).

3. In the pane that appears, select the arrow next to your name, and select **Import**.



4. In the **Import Notebooks** dialog box, select the URL and paste in the following URL:

<https://github.com/solliancenet/microsoft-learning-paths-databricks-notebooks/blob/master/data-engineer>

1. Select **Import**.
2. Select the **07-Dataframe-Advanced-Methods** folder that appears.

8.3.1.2 Task 2: Complete the Date and Time Manipulation notebook

Open the **1.DateTime-Manipulation** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will:

- Explore more of the `...sql.functions` operations
 - Date & time functions

After you've completed the notebook, return to this screen, and continue to the next step.

8.3.2 Exercise 3: Complete the Use Aggregate Functions notebook

In your Azure Databricks workspace, open the **07-Dataframe-Advanced-Methods** folder that you imported within your user folder.

Open the **2.Use-Aggregate-Functions** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will learn various aggregate functions.

After you've completed the notebook, return to this screen, and continue to the next step.

8.3.3 Exercise 4: Complete the De-Duping Data exercise notebook

In your Azure Databricks workspace, open the **07-Dataframe-Advanced-Methods** folder that you imported within your user folder.

Open the **3.Exercise-Deduplication-of-Data** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

The goal of this exercise is to put into practice some of what you have learned about using DataFrames, including renaming columns. The instructions are provided within the notebook, along with empty cells for you to do your work. At the bottom of the notebook are additional cells that will help verify that your work is accurate.

Note: You will find a corresponding notebook within the **Solutions** subfolder. This contains completed cells for the exercise. Refer to the notebook if you get stuck or simply want to see the solution.

9 Module 7 - Ingest and load data into the Data Warehouse

This module teaches students how to ingest data into the data warehouse through T-SQL scripts and Synapse Analytics integration pipelines. The student will learn how to load data into Synapse dedicated SQL pools with PolyBase and COPY using T-SQL. The student will also learn how to use workload management along with a Copy activity in a Azure Synapse pipeline for petabyte-scale data ingestion.

In this module, the student will be able to:

- Perform petabyte-scale ingestion with Azure Synapse Pipelines
- Import data with PolyBase and COPY using T-SQL
- Use data loading best practices in Azure Synapse Analytics

9.1 Lab details

- Module 7 - Ingest and load data into the Data Warehouse
 - Lab details
 - Lab setup and pre-requisites
 - Exercise 0: Start the dedicated SQL pool
 - Exercise 1: Import data with PolyBase and COPY using T-SQL
 - * Task 1: Create staging tables
 - * Task 2: Configure and run PolyBase load operation
 - * Task 3: Configure and run the COPY statement
 - * Task 4: Load data into the clustered columnstore table
 - * Task 5: Use COPY to load text file with non-standard row delimiters
 - * Task 6: Use PolyBase to load text file with non-standard row delimiters
 - Exercise 2: Petabyte-scale ingestion with Azure Synapse Pipelines
 - * Task 1: Configure workload management classification
 - * Task 2: Create pipeline with copy activity
 - Exercise 3: Cleanup
 - * Task 1: Pause the dedicated SQL pool

9.2 Lab setup and pre-requisites

Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Complete the **lab setup instructions** for this module.

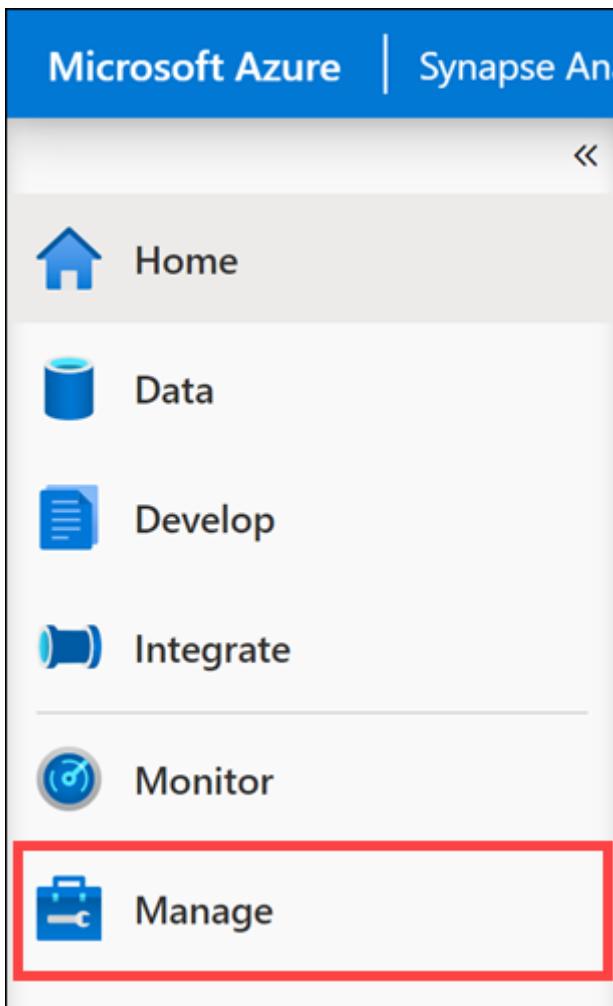
Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

9.3 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the "SQL pools" blade in the Azure portal. The left sidebar shows the following options:

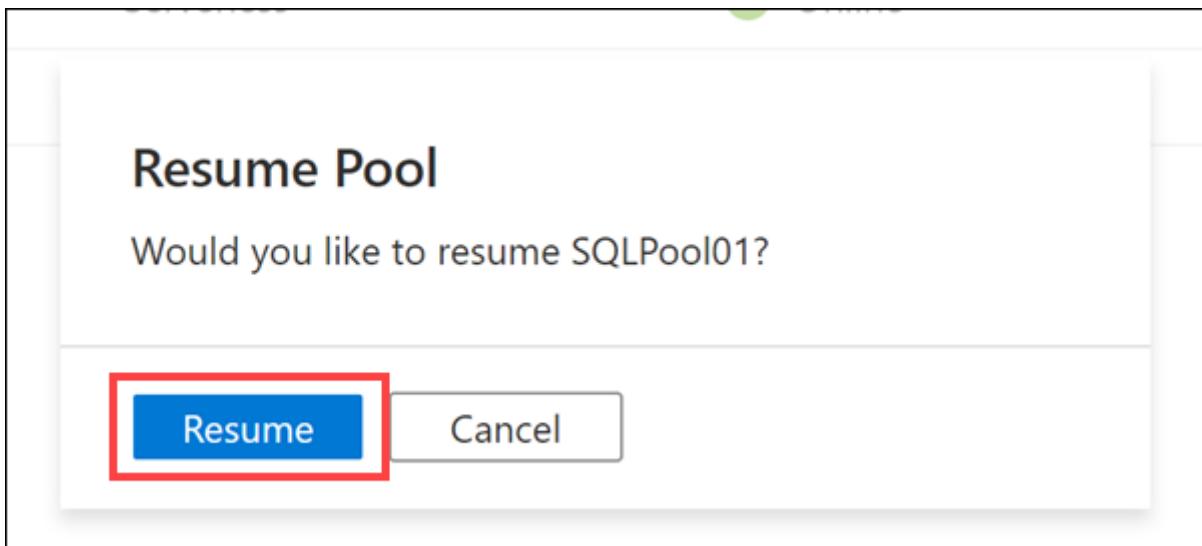
- Analytics pools
- SQL pools** (highlighted with a red box and a red number '1')
- Apache Spark pools
- External connections
- Linked services
- Azure Purview (Preview)
- Integration
- Triggers
- Integration runtimes
- Security
- Access control

The main area displays the "SQL pools" table:

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused	DW100c

A red box labeled '2' highlights the "Resume" button for the "SQLPool01" row.

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

9.4 Exercise 1: Import data with PolyBase and COPY using T-SQL

There are different options for loading large amounts and varying types of data into Azure Synapse Analytics, such as through T-SQL commands using a Synapse SQL Pool, and with Azure Synapse pipelines. In our scenario, Wide World Importers stores most of their raw data in a data lake and in different formats. Among the data loading options available to them, WWI's data engineers are most comfortable using T-SQL.

However, even with their familiarity with SQL, there are some things to consider when loading large or disparate file types and formats. Since the files are stored in ADLS Gen2, WWI can use either PolyBase external tables or the new COPY statement. Both options enable fast and scalable data load operations, but there are some differences between the two:

PolyBase	COPY
Needs CONTROL permission	Relaxed permission
Has row width limits	No row width limit
No delimiters within text	Supports delimiters in text
Fixed line delimiter	Supports custom column and row delimiters
Complex to set up in code	Reduces amount of code

WWI has heard that PolyBase is generally faster than COPY, especially when working with large data sets.

In this exercise, you will help WWI compare ease of setup, flexibility, and speed between these loading strategies.

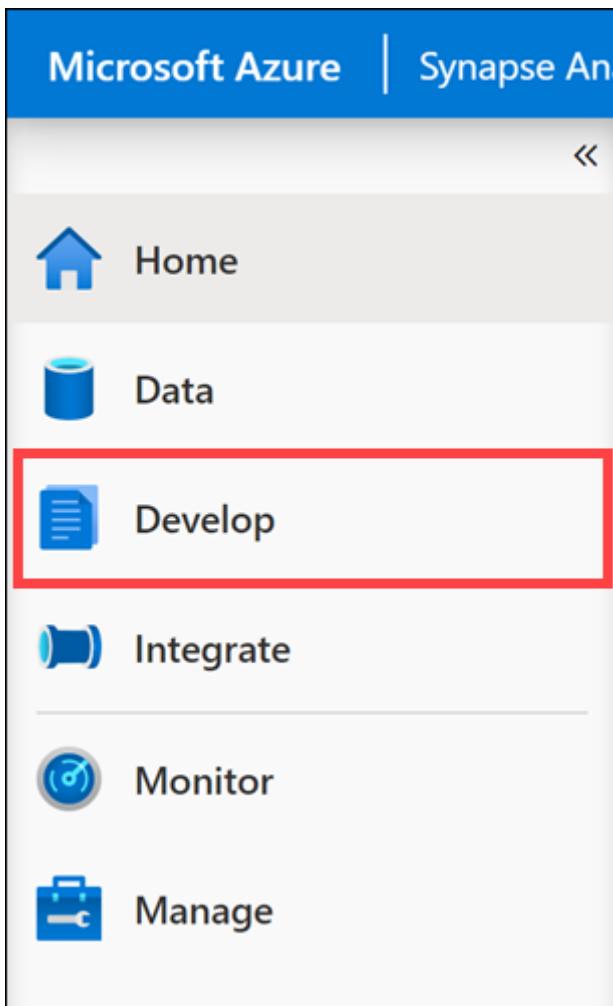
9.4.1 Task 1: Create staging tables

The `Sale` table has a columnstore index to optimize for read-heavy workloads. It is also used heavily for reporting and ad-hoc queries. To achieve the fastest loading speed and minimize the impact of heavy data inserts on the `Sale` table, WWI has decided to create a staging table for loads.

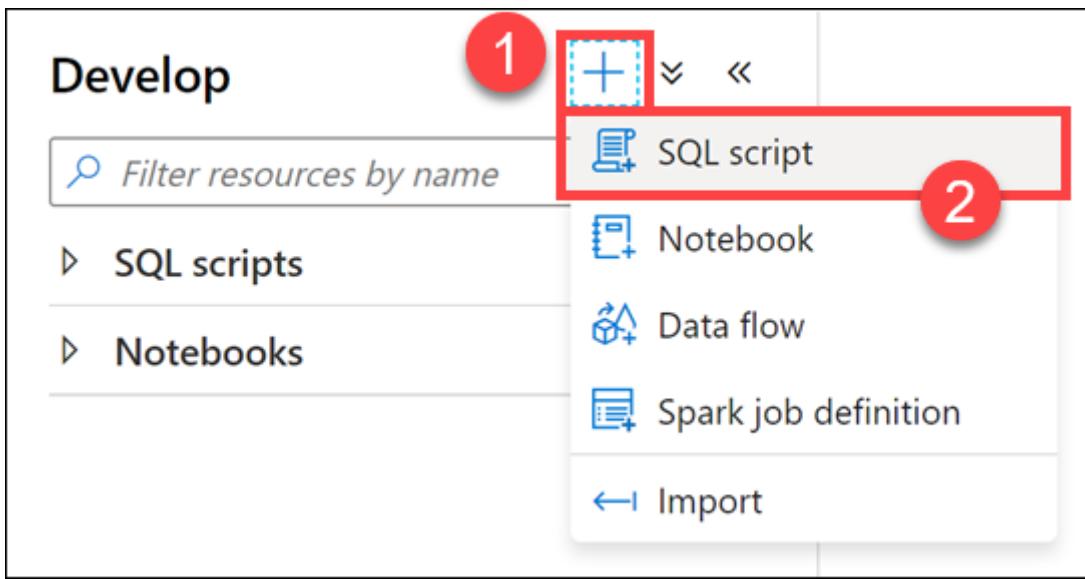
In this task, you will create a new staging table named `SaleHeap` in a new schema named `wwi_staging`. You will define it as a `heap` and use round-robin distribution. When WWI finalizes their data loading pipeline, they will load the data into `SaleHeap`, then insert from the heap table into `Sale`. Although this is a two-step process, the second step of inserting the rows to the production table does not incur data movement across the distributions.

You will also create a new `Sale` clustered columnstore table within the `wwi_staging` to compare data load speeds.

1. Open Synapse Analytics Studio (<https://web.azuresynapse.net/>), and then navigate to the **Develop** hub.



2. From the **Develop** menu, select the + button and choose **SQL Script** from the context menu.



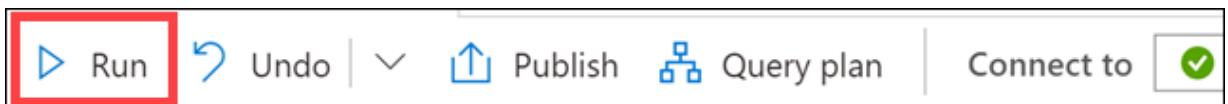
3. In the toolbar menu, connect to the **SQL Pool** database to execute the query.



4. In the query window, replace the script with the following to create the `wwi_staging` schema:

```
CREATE SCHEMA [wwi_staging]
```

5. Select **Run** from the toolbar menu to execute the SQL command.



Note: If you receive the following error, continue to the next step: Failed to execute query. Error: There is already an object named 'wwi_staging' in the database. CREATE SCHEMA failed due to previous errors.

6. In the query window, replace the script with the following to create the heap table:

```
CREATE TABLE [wwi_staging].[SaleHeap]
(
    [TransactionId] [uniqueidentifier] NOT NULL,
    [CustomerId] [int] NOT NULL,
    [ProductId] [smallint] NOT NULL,
    [Quantity] [smallint] NOT NULL,
    [Price] [decimal](9,2) NOT NULL,
    [TotalAmount] [decimal](9,2) NOT NULL,
    [TransactionDate] [int] NOT NULL,
    [ProfitAmount] [decimal](9,2) NOT NULL,
    [Hour] [tinyint] NOT NULL,
    [Minute] [tinyint] NOT NULL,
    [StoreId] [smallint] NOT NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    HEAP
)
```

7. Select **Run** from the toolbar menu to execute the SQL command.

8. In the query window, replace the script with the following to create the Sale table in the wwi_staging schema for load comparisons:

```
CREATE TABLE [wwi_staging].[Sale]
(
    [TransactionId] [uniqueidentifier] NOT NULL,
    [CustomerId] [int] NOT NULL,
    [ProductId] [smallint] NOT NULL,
    [Quantity] [smallint] NOT NULL,
    [Price] [decimal](9,2) NOT NULL,
    [TotalAmount] [decimal](9,2) NOT NULL,
    [TransactionDate] [int] NOT NULL,
    [ProfitAmount] [decimal](9,2) NOT NULL,
    [Hour] [tinyint] NOT NULL,
    [Minute] [tinyint] NOT NULL,
    [StoreId] [smallint] NOT NULL
)
WITH
(
    DISTRIBUTION = HASH ( [CustomerId] ),
    CLUSTERED COLUMNSTORE INDEX,
    PARTITION
    (
        [TransactionDate] RANGE RIGHT FOR VALUES (20100101, 20100201, 20100301, 20100401, 20100501)
    )
)
```

9. Select **Run** from the toolbar menu to execute the SQL command.

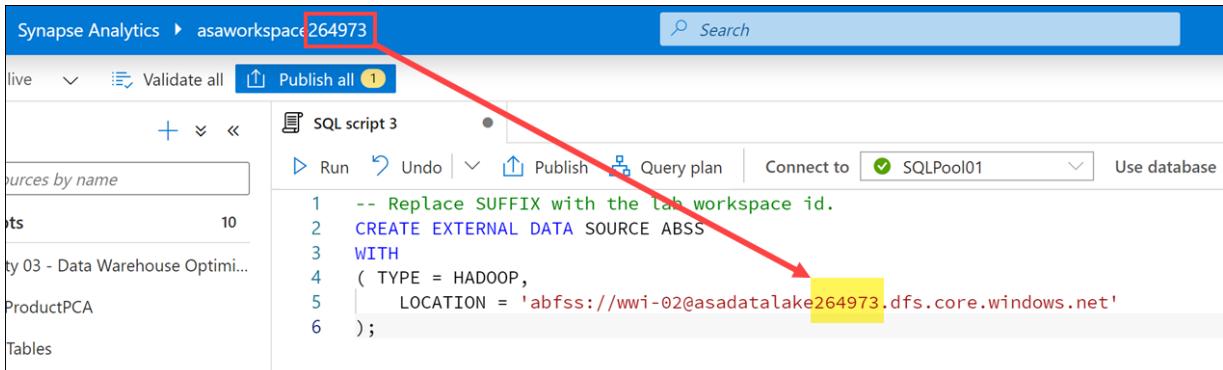
9.4.2 Task 2: Configure and run PolyBase load operation

PolyBase requires the following elements:

- An external data source that points to the `abfss` path in ADLS Gen2 where the Parquet files are located
 - An external file format for Parquet files
 - An external table that defines the schema for the files, as well as the location, data source, and file format
1. In the query window, replace the script with the following to create the external data source. Be sure to replace `SUFFIX` with the lab workspace id:

```
-- Replace SUFFIX with the lab workspace id.
CREATE EXTERNAL DATA SOURCE ABSS
WITH
( TYPE = HADOOP,
  LOCATION = 'abfss://wwi-02@asadatalakeSUFFIX.dfs.core.windows.net'
);
```

You can find the lab workspace id at the end of the Synapse Analytics workspace name, as well as your user name:



2. Select **Run** from the toolbar menu to execute the SQL command.
3. In the query window, replace the script with the following to create the external file format and external data table. Notice that we defined `TransactionId` as an `nvarchar(36)` field instead of `uniqueidentifier`. This is because external tables do not currently support `uniqueidentifier` columns:

```
CREATE EXTERNAL FILE FORMAT [ParquetFormat]
WITH (
  FORMAT_TYPE = PARQUET,
  DATA_COMPRESSION = 'org.apache.hadoop.io.compress.SnappyCodec'
)
GO

CREATE SCHEMA [wwi_external];
GO

CREATE EXTERNAL TABLE [wwi_external].Sales
(
  [TransactionId] [nvarchar](36) NOT NULL,
  [CustomerId] [int] NOT NULL,
  [ProductId] [smallint] NOT NULL,
  [Quantity] [smallint] NOT NULL,
  [Price] [decimal](9,2) NOT NULL,
  [TotalAmount] [decimal](9,2) NOT NULL,
  [TransactionDate] [int] NOT NULL,
  [ProfitAmount] [decimal](9,2) NOT NULL,
  [Hour] [tinyint] NOT NULL,
  [Minute] [tinyint] NOT NULL,
  [StoreId] [smallint] NOT NULL
)
WITH
(
  LOCATION = '/sale-small%2FYear%3D2019',
  DATA_SOURCE = ABSS,
```

```

    FILE_FORMAT = [ParquetFormat]
)
GO

```

Note: The /sale-small/Year=2019/ folder's Parquet files contain **339,507,246 rows**.

4. Select **Run** from the toolbar menu to execute the SQL command.
5. In the query window, replace the script with the following to load the data into the `wwi_staging.SalesHeap` table. This command takes approximately 10 minutes to execute:

```

INSERT INTO [wwi_staging].[SaleHeap]
SELECT *
FROM [wwi_external].[Sales]

```

While this is running, read the rest of the lab instructions to familiarize yourself with the content.

9.4.3 Task 3: Configure and run the COPY statement

Now let's see how to perform the same load operation with the COPY statement.

1. In the query window, replace the script with the following to truncate the heap table and load data using the COPY statement. As you did before, be sure to replace `SUFFIX` with the lab workspace id:

```

TRUNCATE TABLE wwi_staging.SaleHeap;
GO

```

```

-- Replace <PrimaryStorage> with the workspace default storage account name.
COPY INTO wwi_staging.SaleHeap
FROM 'https://asadatalakeSUFFIX.dfs.core.windows.net/wwi-02/sale-small%2FYear%3D2019'
WITH (
    FILE_TYPE = 'PARQUET',
    COMPRESSION = 'SNAPPY'
)
GO

```

2. Select **Run** from the toolbar menu to execute the SQL command. It takes a few minutes to execute this command. **Take note** of how long it took to execute this query.

While this is running, read the rest of the lab instructions to familiarize yourself with the content.

3. In the query window, replace the script with the following to see how many rows were imported:

```

SELECT COUNT(1) FROM wwi_staging.SaleHeap(nolock)

```

4. Select **Run** from the toolbar menu to execute the SQL command. You should see a result of 339507246.

Do the number of rows match for both load operations? Which activity was fastest? You should see that both copied the same amount of data in roughly the same amount of time.

9.4.4 Task 4: Load data into the clustered columnstore table

For both of the load operations above, we inserted data into the heap table. What if we inserted into the clustered columnstore table instead? Is there really a performance difference? Let's find out!

1. In the query window, replace the script with the following to load data into the clustered columnstore `Sale` table using the COPY statement. Be sure to replace `SUFFIX` with the id for your workspace. **DO NOT RUN** this command. In the interest of time, we will skip this command since it takes around 7 minutes to execute:

```

-- Replace SUFFIX with the workspace default storage account name.
COPY INTO wwi_staging.Sale
FROM 'https://asadatalakeSUFFIX.dfs.core.windows.net/wwi-02/sale-small%2FYear%3D2019'
WITH (
    FILE_TYPE = 'PARQUET',
    COMPRESSION = 'SNAPPY'
)

```

```
)  
GO
```

What were the results? Did the load operation take more or less time writing to `Sale` table vs. the heap (`SaleHeap`) table?

In our case, the results are as follows:

PolyBase vs. `COPY` (DW500) (*insert 2019 small data set (339,507,246 rows)*):

- `COPY` (Heap: **5:08**, clustered columnstore: **6:52**)
- PolyBase (Heap: **5:59**)

9.4.5 Task 5: Use `COPY` to load text file with non-standard row delimiters

One of the advantages `COPY` has over PolyBase is that it supports custom column and row delimiters.

WWI has a nightly process that ingests regional sales data from a partner analytics system and saves the files in the data lake. The text files use non-standard column and row delimiters where columns are delimited by a . and rows by a ,:

```
20200421.114892.130282.159488.172105.196533,20200420.109934.108377.122039.101946.100712,20200419.253714
```

The data has the following fields: `Date`, `NorthAmerica`, `SouthAmerica`, `Europe`, `Africa`, and `Asia`. They must process this data and store it in Synapse Analytics.

1. In the query window, replace the script with the following to create the `DailySalesCounts` table and load data using the `COPY` statement. As before, be sure to replace `SUFFIX` with the id for your workspace:

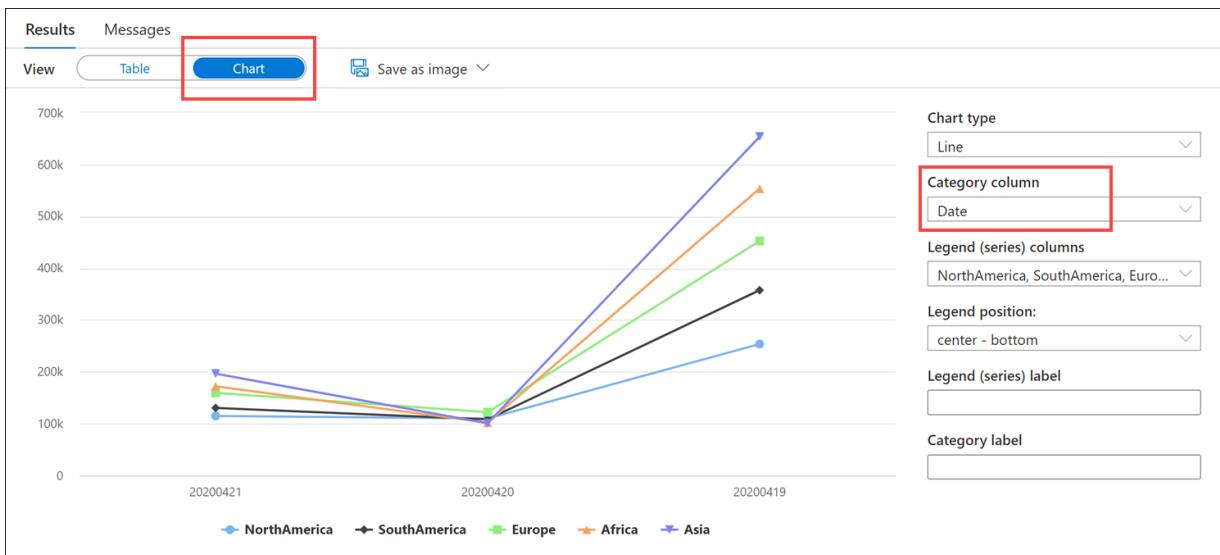
```
CREATE TABLE [wwi_staging].DailySalesCounts  
(  
    [Date] [int] NOT NULL,  
    [NorthAmerica] [int] NOT NULL,  
    [SouthAmerica] [int] NOT NULL,  
    [Europe] [int] NOT NULL,  
    [Africa] [int] NOT NULL,  
    [Asia] [int] NOT NULL  
)  
GO  
  
-- Replace <PrimaryStorage> with the workspace default storage account name.  
COPY INTO wwi_staging.DailySalesCounts  
FROM 'https://asadatalakeSUFFIX.dfs.core.windows.net/wwi-02/campaign-analytics/dailycounts.txt'  
WITH (  
    FILE_TYPE = 'CSV',  
    FIELDTERMINATOR='.',  
    ROWTERMINATOR=','  
)  
GO
```

Notice the `FIELDTERMINATOR` and `ROWTERMINATOR` properties that help us correctly parse the file.

2. Select **Run** from the toolbar menu to execute the SQL command.
3. In the query window, replace the script with the following to view the imported data:

```
SELECT * FROM [wwi_staging].DailySalesCounts  
ORDER BY [Date] DESC
```

4. Select **Run** from the toolbar menu to execute the SQL command.
5. Try viewing the results in a Chart and set the **Category column** to `Date`:



9.4.6 Task 6: Use PolyBase to load text file with non-standard row delimiters

Let's try this same operation using PolyBase.

1. In the query window, replace the script with the following to create a new external file format, external table, and load data using PolyBase:

```

CREATE EXTERNAL FILE FORMAT csv_dailysales
WITH (
    FORMAT_TYPE = DELIMITEDTEXT,
    FORMAT_OPTIONS (
        FIELD_TERMINATOR = '。',
        DATE_FORMAT = '',
        USE_TYPE_DEFAULT = False
    )
);
GO

CREATE EXTERNAL TABLE [wwi_external].[DailySalesCounts]
(
    [Date] [int] NOT NULL,
    [NorthAmerica] [int] NOT NULL,
    [SouthAmerica] [int] NOT NULL,
    [Europe] [int] NOT NULL,
    [Africa] [int] NOT NULL,
    [Asia] [int] NOT NULL
)
WITH
(
    LOCATION = '/campaign-analytics/dailycounts.txt',
    DATA_SOURCE = ABSS,
    FILE_FORMAT = csv_dailysales
)
GO
INSERT INTO [wwi_staging].[DailySalesCounts]
SELECT *
FROM [wwi_external].[DailySalesCounts]

```

2. Select **Run** from the toolbar menu to execute the SQL command.

You should see an error similar to: Failed to execute query. Error: HdfsBridge::recordReaderFillBuffer - Unexpected error encountered filling record reader buffer: HadoopExecutionException: Too many columns in the line..

Why is this? According to [PolyBase documentation](#):

The row delimiter in delimited-text files must be supported by Hadoop's LineRecordReader. That is, it must be either \r, \n, or \r\n. These delimiters are not user-configurable.

This is an example of where COPY's flexibility gives it an advantage over PolyBase.

9.5 Exercise 2: Petabyte-scale ingestion with Azure Synapse Pipelines

Tailwind Traders needs to ingest large volumes of sales data into the data warehouse. They want a repeatable process that can efficiently load the data. When the data loads, they want to prioritize the data movement jobs so they take priority.

You have decided to create a proof of concept data pipeline to import a large Parquet file, following best practices to improve the load performance.

There is often a level of orchestration involved when moving data into a data warehouse, coordinating movement from one or more data sources and sometimes some level of transformation. The transformation step can occur during (extract-transform-load - ETL) or after (extract-load-transform - ELT) data movement. Any modern data platform must provide a seamless experience for all the typical data wrangling actions like extractions, parsing, joining, standardizing, augmenting, cleansing, consolidating, and filtering. Azure Synapse Analytics provides two significant categories of features - data flows and data orchestrations (implemented as pipelines).

In this segment of the lab, we will focus on the orchestration aspect. The next segment will focus more on the transformation (data flow) pipelines.

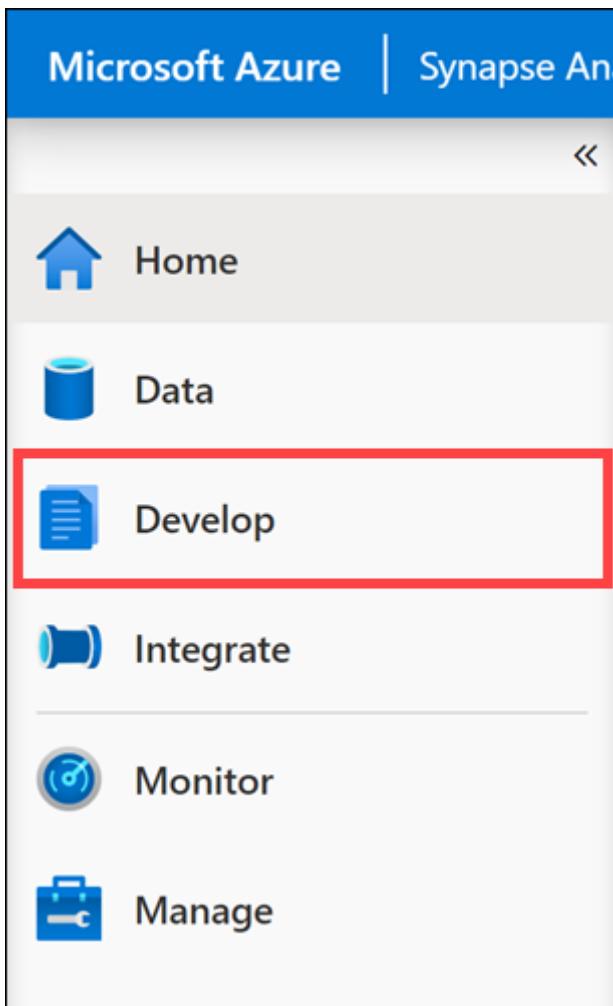
9.5.1 Task 1: Configure workload management classification

When loading a large amount of data, it is best to run only one load job at a time for fastest performance. If this isn't possible, run a minimal number of loads concurrently. If you expect a large loading job, consider scaling up your dedicated SQL pool before the load.

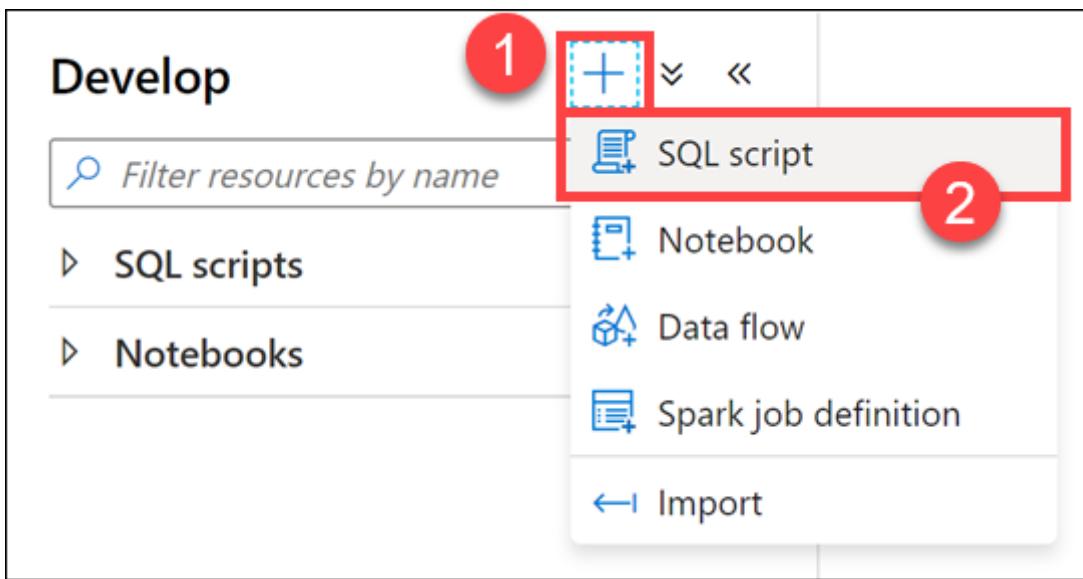
Be sure that you allocate enough memory to the pipeline session. To do this, increase the resource class of a user which has permissions to rebuild the index on this table to the recommended minimum.

To run loads with appropriate compute resources, create loading users designated for running loads. Assign each loading user to a specific resource class or workload group. To run a load, sign in as one of the loading users, and then run the load. The load runs with the user's resource class.

1. Open Synapse Studio (<https://web.azuresynapse.net/>), and then navigate to the **Develop** hub.



2. From the **Develop** menu, select the + button (1) and choose **SQL Script** from the context menu (2).



3. In the toolbar menu, connect to the **SQLPool01** database to execute the query.



4. In the query window, replace the script with the following to create a workload group, **BigDataLoad**, that uses workload isolation by reserving a minimum of 50% resources with a cap of 100%:

```
IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_classifiers WHERE group_name = 'BigDataLoad')
BEGIN
```

```

CREATE WORKLOAD GROUP BigDataLoad WITH
(
    MIN_PERCENTAGE_RESOURCE = 50 -- integer value
    ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 25 -- (guaranteed a minimum of 4 concurrency)
    ,CAP_PERCENTAGE_RESOURCE = 100
);
END

```

5. Select **Run** from the toolbar menu to execute the SQL command.

6. In the query window, replace the script with the following to create a new workload classifier, **HeavyLoader** that assigns the `asa.sql.import01` user we created in your environment to the **BigDataLoad** workload group. At the end, we select from `sys.workload_management_workload_classifiers` to view all classifiers, including the one we just created:

```

IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_classifiers WHERE [name] = 'HeavyLoader')
BEGIN
    CREATE WORKLOAD Classifier HeavyLoader WITH
    (
        Workload_Group ='BigDataLoad',
        MemberName='asa.sql.import01',
        IMPORTANCE = HIGH
    );
END

```

```
SELECT * FROM sys.workload_management_workload_classifiers
```

7. Select **Run** from the toolbar menu to execute the SQL command. You should see the new classifier in the query results:

```

1 IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_classifiers WHERE [name] = 'HeavyLoader')
2 BEGIN
3     CREATE WORKLOAD Classifier HeavyLoader WITH
4     (
5         Workload_Group ='BigDataLoad',
6         MemberName='asa.sql.import01',
7         IMPORTANCE = HIGH
8     );
9 END
10
11 SELECT * FROM sys.workload_management_workload_classifiers

```

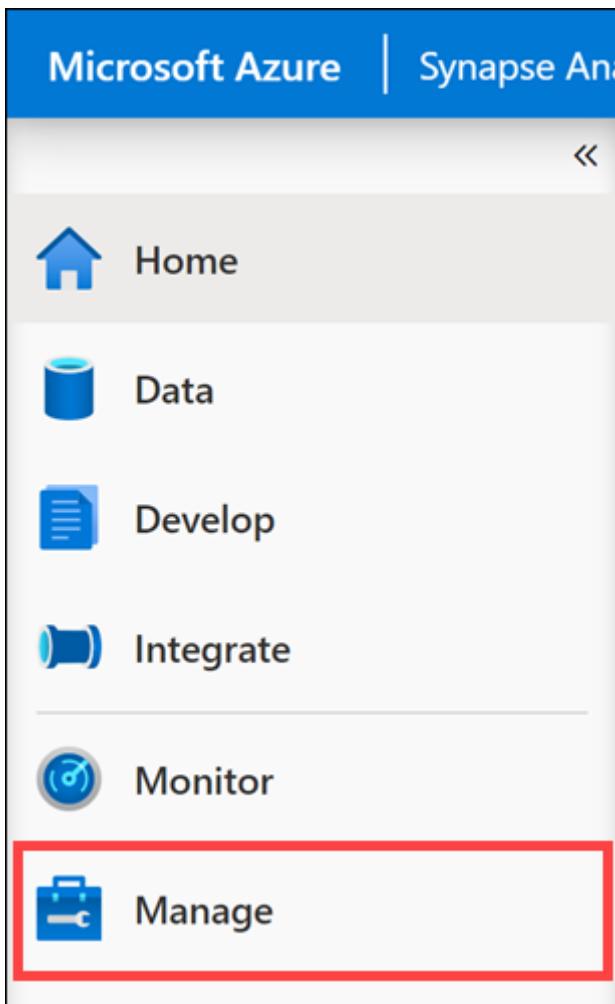
Results Messages

View Table Chart Export results ▾

Search

Classifier_id	Name	Group_name	Importance	Create_time	Modify_time	Is_enabled
8	staticrc40	staticrc40	normal	2021-02-26T11:00:00.000Z	2021-02-26T11:00:00.000Z	True
9	staticrc50	staticrc50	normal	2021-02-26T11:00:00.000Z	2021-02-26T11:00:00.000Z	True
10	staticrc60	staticrc60	normal	2021-02-26T11:00:00.000Z	2021-02-26T11:00:00.000Z	True
11	staticrc70	staticrc70	normal	2021-02-26T11:00:00.000Z	2021-02-26T11:00:00.000Z	True
12	staticrc80	staticrc80	normal	2021-02-26T11:00:00.000Z	2021-02-26T11:00:00.000Z	True
13	CEO	largerc	high	2021-02-26T21:00:00.000Z	2021-02-26T21:00:00.000Z	True
1014	CEODreamDemo	CEODemo	below_normal	2021-02-26T22:00:00.000Z	2021-02-26T22:00:00.000Z	True
1015	HeavyLoader	BigDataLoad	high	2021-02-27T15:00:00.000Z	2021-02-27T15:00:00.000Z	True

8. Navigate to the **Manage** hub.



9. Select **Linked services** in the left-hand menu (1), then select a linked service named **sqlpool01_import01** (2).

Linked services

Linked services are much like connection strings, which define the connection information required by your pipeline components.

New

Filter by name Annotations : Any

Showing 1 - 11 of 11 items

Name ↑↓	Type ↑↓
asadatalake022521	Azure Data Lake Storage Gen2
asakeyvault022521	Azure Key Vault
asapowerbi022521	Power BI
asastore022521	Azure Blob Storage
asaworkspace022521-WorkspaceDefa...	Azure Synapse Analytics
asaworkspace022521-WorkspaceDefa...	Azure Data Lake Storage Gen2
sqlpool01	Azure Synapse Analytics
sqlpool01_highperf	Azure Synapse Analytics
sqlpool01_import01	Azure Synapse Analytics
sqlpool01_workload01	Azure Synapse Analytics
sqlpool01_workload02	Azure Synapse Analytics

- Notice that the user name for the dedicated SQL pool connection is the **asa.sql.import01** user we added to the **HeavyLoader** classifier. We will use this linked service in our new pipeline to reserve resources for the data load activity.

Edit linked service (Azure Synapse Analytics)

Name *

Description



Connect via integration runtime *

 Connection string Azure Key Vault

Account selection method

 From Azure subscription Enter manually

Fully qualified domain name *

Database name *

Authentication type *



User name *

 Password Azure Key Vault

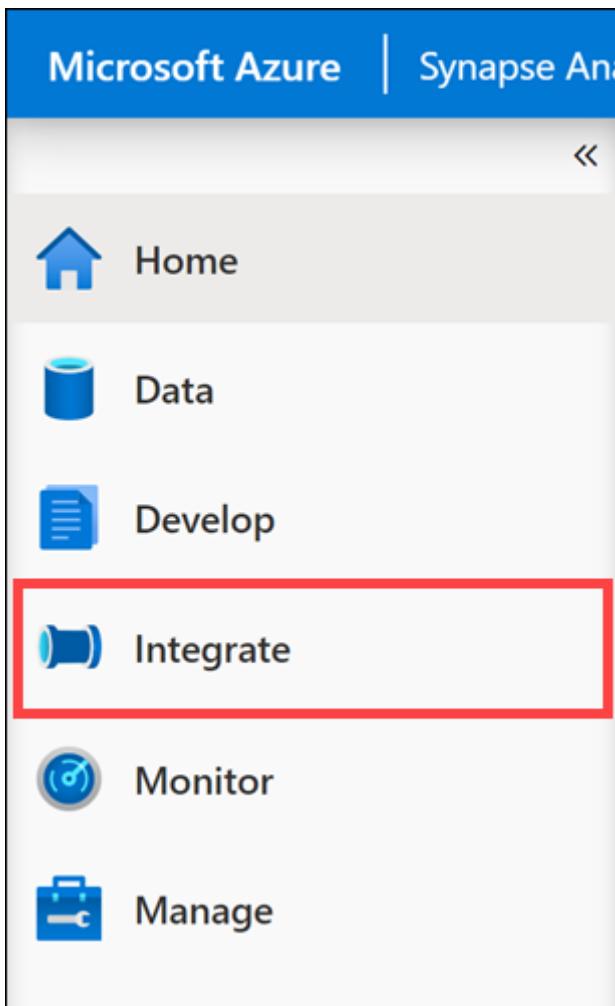
AKV linked service *



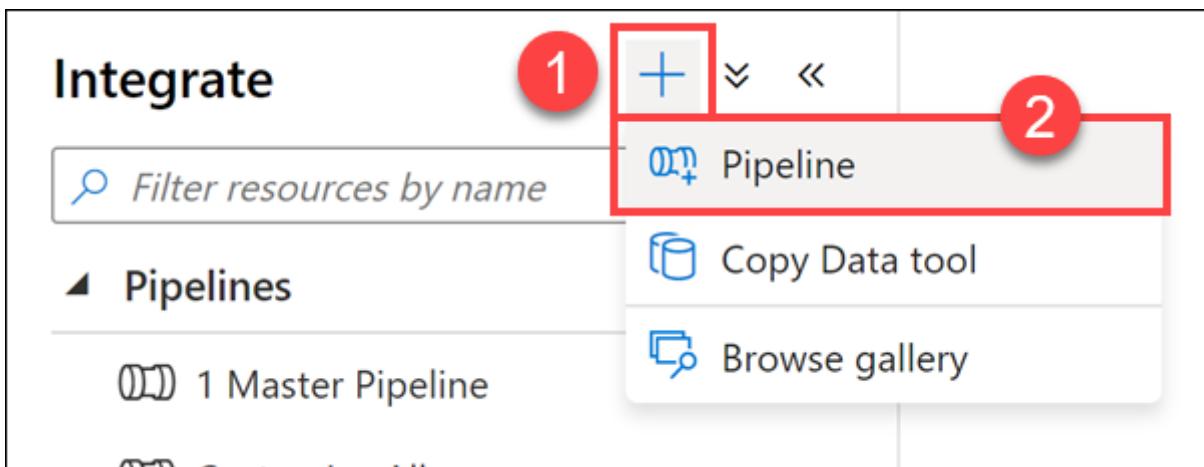
11. Select **Cancel** to close the dialog, and select **Discard changes** when prompted.

9.5.2 Task 2: Create pipeline with copy activity

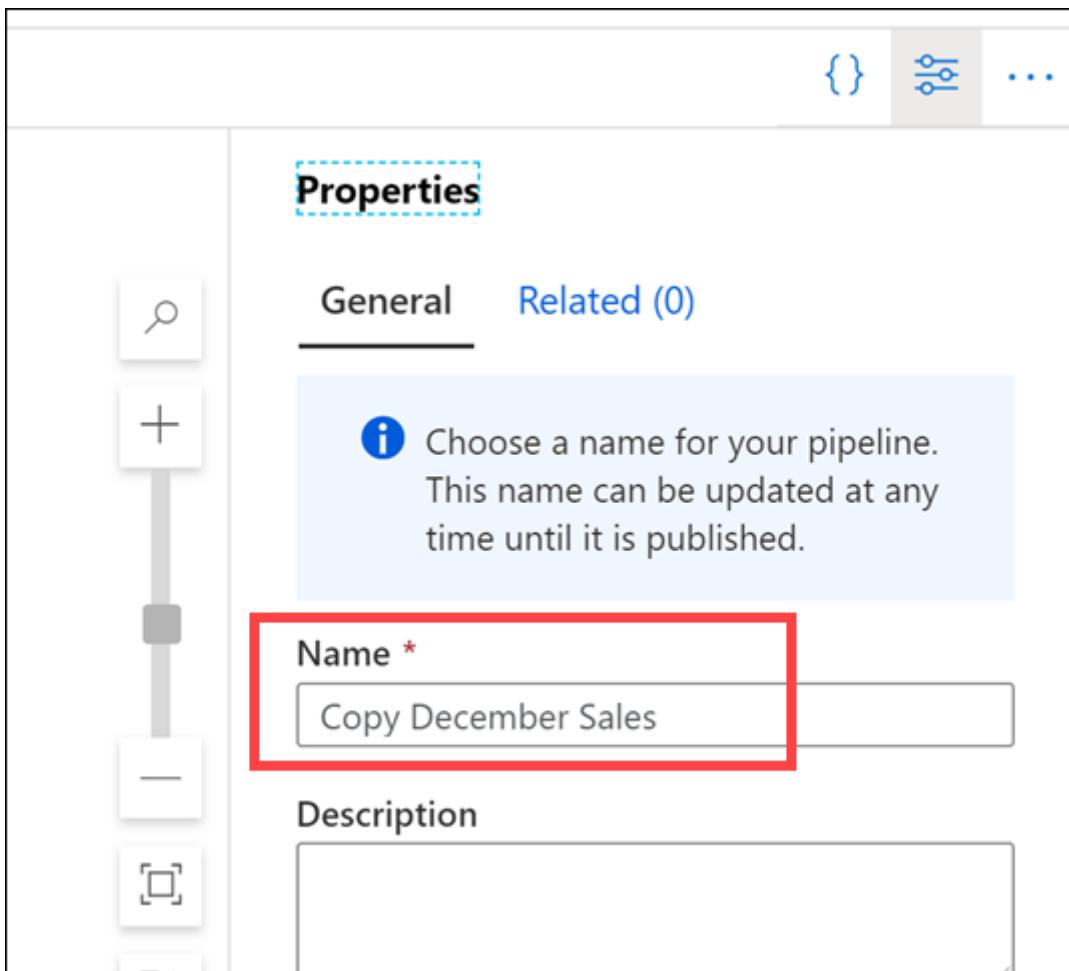
1. Navigate to the **Integrate** hub.



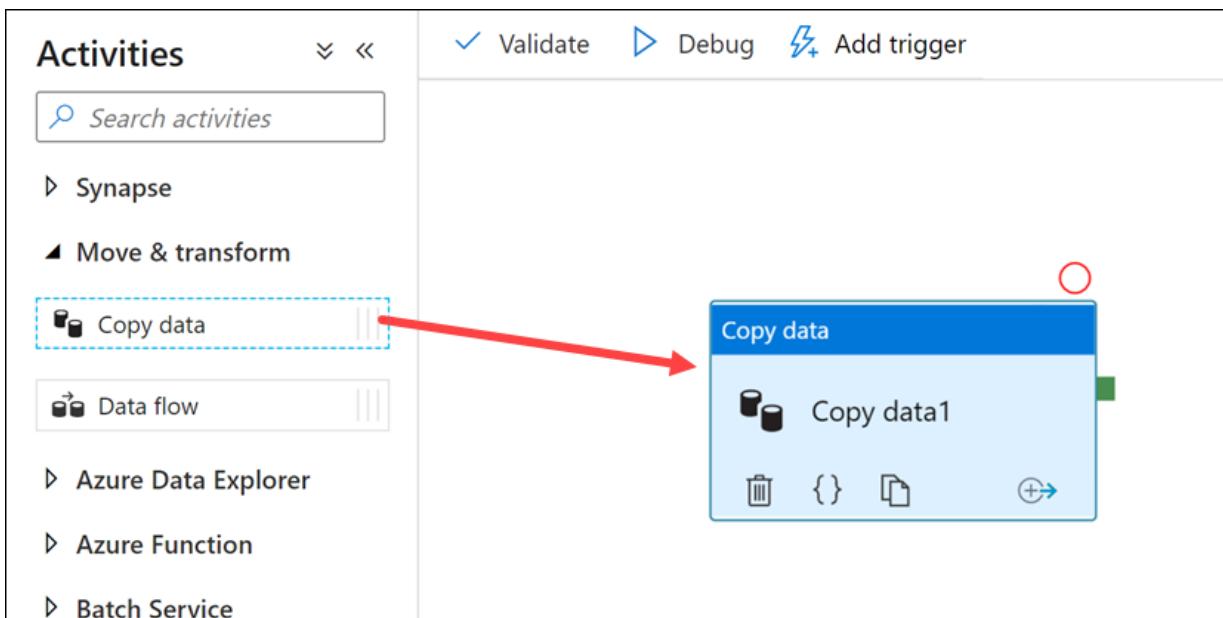
2. Select + (1) then Pipeline (2) to create a new pipeline.



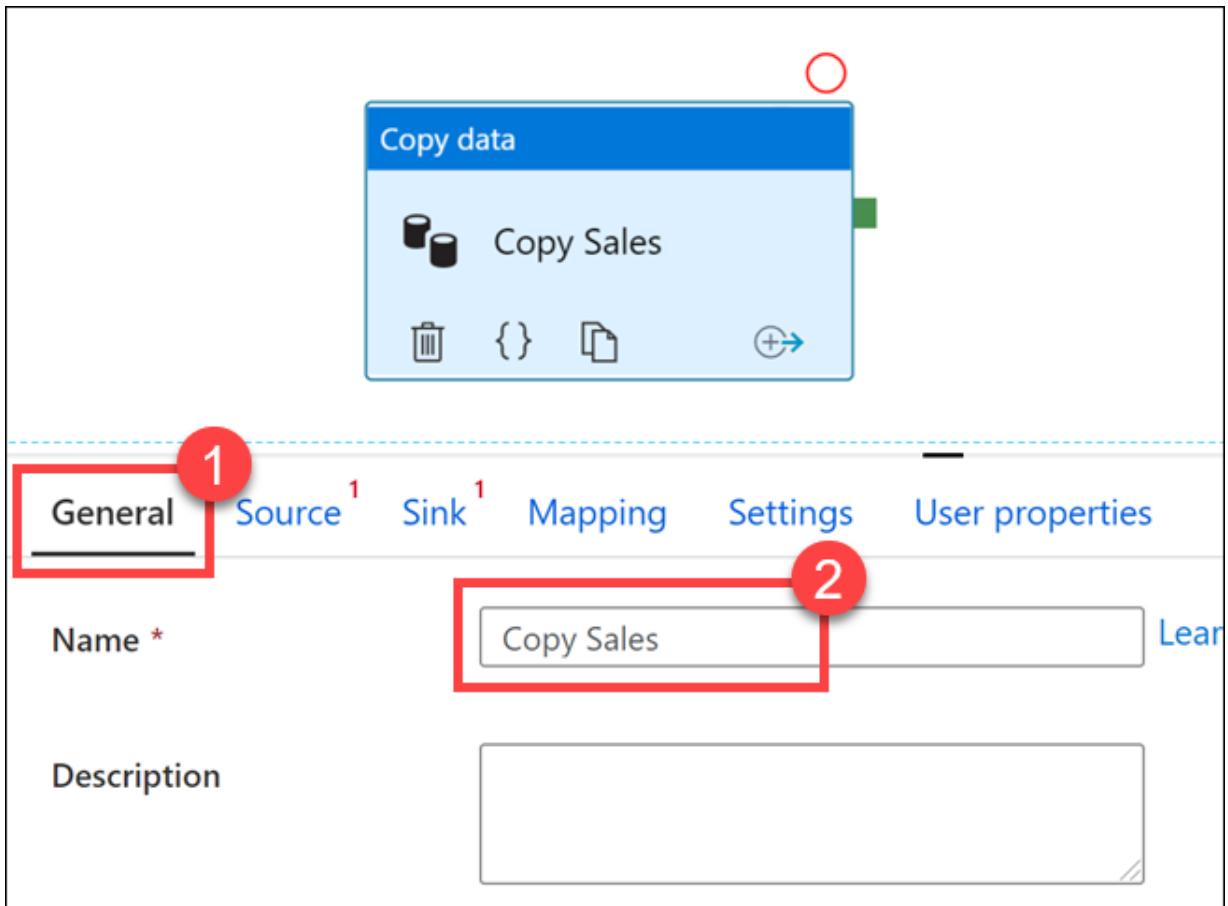
3. In the Properties pane for the new pipeline, enter the following Name: Copy December Sales.



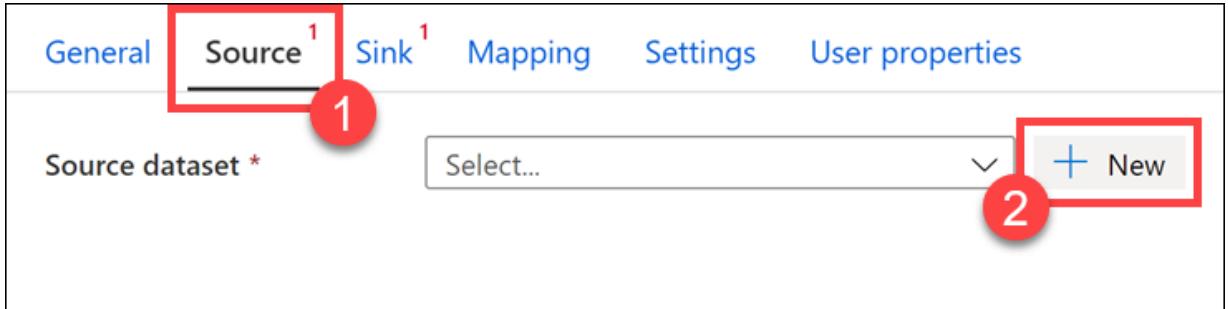
4. Expand **Move & transform** within the Activities list, then drag the **Copy data** activity onto the pipeline canvas.



5. Select the **Copy data** activity on the canvas, select the **General** tab (1), and set the **Name** to **Copy Sales** (2).



6. Select the **Source** tab (1), then select + New (2) next to Source dataset.



7. Select the **Azure Data Lake Storage Gen2** data store (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store.[Learn more](#)

Select a data store

The screenshot shows a grid of data storage options. The 'Azure Data Lake Storage Gen2' option is highlighted with a red box and circled with a red number 1. The 'Continue' button at the bottom left is also highlighted with a red box and circled with a red number 2.

All	Azure	Database	File	Generic protocol	NoSQL	Services and apps
Azure Cosmos DB (SQL API)	Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen1				
Azure Data Lake Storage Gen2	Azure Database for MariaDB	Azure Database for MySQL				
Azure Database for PostgreSQL	Azure Databricks Delta Lake	Azure File Storage				

8. Choose the **Parquet** format (1), then select **Continue** (2).

Select format

Choose the format type of your data



Avro



Binary



DelimitedText



Excel



Json



ORC



Parquet



XML

Continue

Back

Cancel

9. In the properties, set the name to **asal400_december_sales** (1) and select the **asadatalakeNNNNNN linked service** (2). Browse to the **wwi-02/campaign-analytics/sale-20161230-snappy.parquet** file location (3), select **From sample file** (4) for schema import. [Download this sample file](#) to your computer, then browse to it in the **Select file** field (5). Select **OK** (6).

Set properties

Choose a name for your dataset. This name can be updated at any time until it is published.

Name asal400_december_sales 1

Linked service * asadatalake 2

File path wwi-02 / campaign-analytics / sale-20161230-snappy.par 3

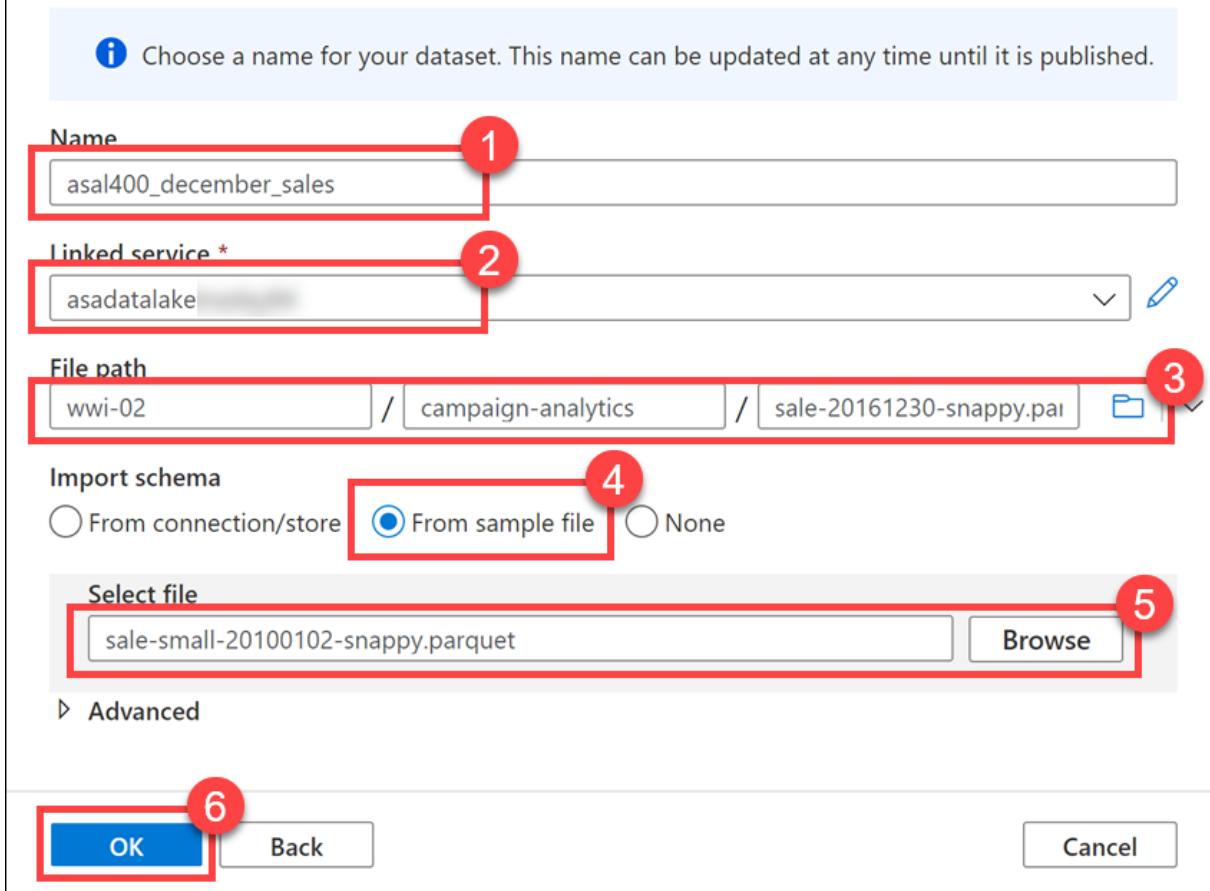
Import schema 4

From connection/store From sample file None

Select file sale-small-20100102-snappy.parquet 5

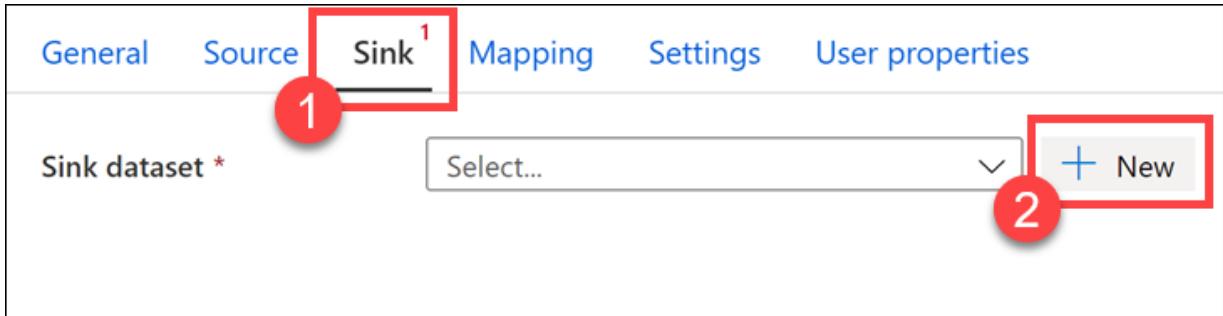
Advanced

OK 6 Back Cancel



We downloaded a sample Parquet file that has the exact same schema, but is much smaller. This is because the file we are copying is too large to automatically infer the schema in the copy activity source settings.

10. Select the **Sink** tab (1), then select **+ New** (2) next to Sink dataset.



11. Select the **Azure Synapse Analytics** data store (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store.[Learn more](#)

Select a data store

The screenshot shows a grid of data store options. The 'Azure Synapse Analytics' option is highlighted with a red box and circled with a red number 1. The 'Continue' button at the bottom left is also highlighted with a red box and circled with a red number 2.

All	Azure	Database	File	Generic protocol	Services and apps
Azure File Storage	Azure SQL Database	Azure SQL Database Managed Instance			
Azure Search	Azure Synapse Analytics	Azure Synapse dedicated SQL pool			
Azure Table Storage	Common Data Service for Apps	Dynamics 365			

Continue Cancel

12. In the properties, set the name to **asa1400_saleheap_asa** (1) and select the **sqlpool01_import01** linked service (2) that connects to Synapse Analytics with the **asa.sql.import01** user. For the table name, scroll the Table name dropdown and choose the **wwi_perf.Sale_Heap** table (3) then select **OK** (4).

Set properties

i Choose a name for your dataset. This name can be updated at any time until it is published.

Name	1
Linked service *	2
Table name	3
<input type="checkbox"/> Edit	
Import schema	
<input checked="" type="radio"/> From connection/store <input type="radio"/> None	
Advanced	
<input type="button" value="OK"/> 4 <input type="button" value="Back"/>	
<input type="button" value="Cancel"/>	

13. In the **Sink** tab, select the **Copy command** (1) copy method and enter the following in the pre-copy script to clear the table before import: `TRUNCATE TABLE wwi_perf.Sale_Heap` (2).

General	Source	Sink	Mapping	Settings	User properties
<p>Sink dataset *</p> <div style="display: flex; align-items: center;"> <div style="flex: 1;"> <input type="text" value="asal400_saleheap_asa"/> </div> <div style="flex: 1; text-align: right;"> Open New </div> </div> <p>Copy method</p> <div style="display: flex; align-items: center;"> <input type="radio"/> PolyBase <input checked="" type="radio"/> Copy command (Preview) <input type="radio"/> Bulk insert </div> <p>Allow copy command</p> <p>Default values</p> <p>Additional options</p> <p>Table option</p> <p>Pre-copy script</p> <p>Write batch timeout</p>					
<div style="border: 1px solid red; padding: 5px; margin-top: 10px;">1</div> <div style="border: 1px solid red; padding: 5px; margin-top: 10px;">2</div>					

The fastest and most scalable way to load data is through PolyBase or the COPY statement (1), and the COPY statement provides the most flexibility for high-throughput data ingestion into the SQL pool.

14. Select the **Mapping** tab (1) and select **Import schemas** (2) to create mappings for each source and destination field. Select **TransactionDate** in the source column (3) to map it to the **TransactionDateId** destination column.

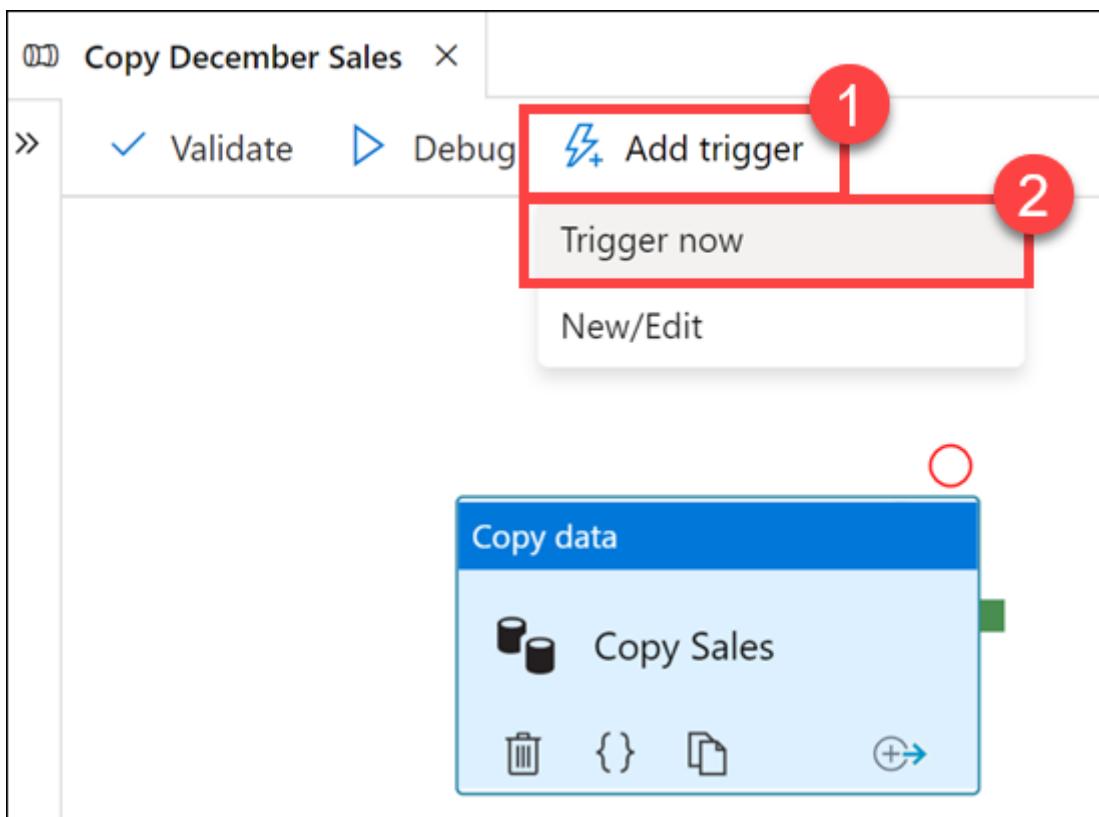
Source	Type	Destination	Type
TransactionId	UTF8	TransactionId	uniqueidentifier
CustomerId	INT32	CustomerId	int
ProductId	INT_16	ProductId	smallint
Quantity	INT_8	Quantity	tinyint
Price	DECIMAL	Price	decimal
TotalAmount	DECIMAL	TotalAmount	decimal
TransactionDate	INT32	TransactionDateId	int
ProfitAmount	DECIMAL	ProfitAmount	decimal
Hour	INT_8	Hour	tinyint
Minute	INT_8	Minute	tinyint
StoreId	INT_16	StoreId	smallint

15. Select the **Settings** tab (1) and set the **Data integration unit** to 8 (2). This is required due to the large size of the source Parquet file.

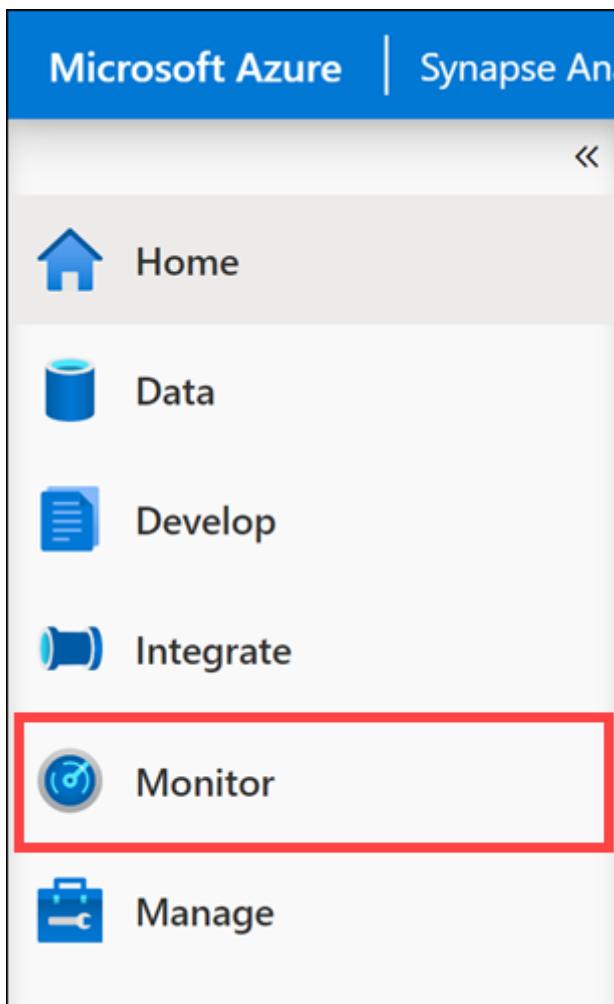
General	Source	Sink	Mapping	Settings	User properties
i You will be charged # of used DIUs * copy duration .25/DIU-hour. Local					
Data integration unit <input type="text" value="8"/> ▼					
<input type="checkbox"/> Edit					
Degree of copy parallelism <input type="text"/>					
<input checked="" type="checkbox"/> Edit					
Fault tolerance <input type="text"/>					
Enable staging <input type="checkbox"/>					

16. Select **Publish all**, then **Publish** to save your new resources.

17. Select **Add trigger** (1), then **Trigger now** (2). Select **OK** in the pipeline run trigger to begin.



18. Navigate to the **Monitor** hub.



19. Select **Pipeline Runs** (1). You can see the status (2) of your pipeline run here. Note that you may need to refresh the view (3). Once the pipeline run is complete, you can query the `wwi_perf.Sale_Heap` table to view the imported data.

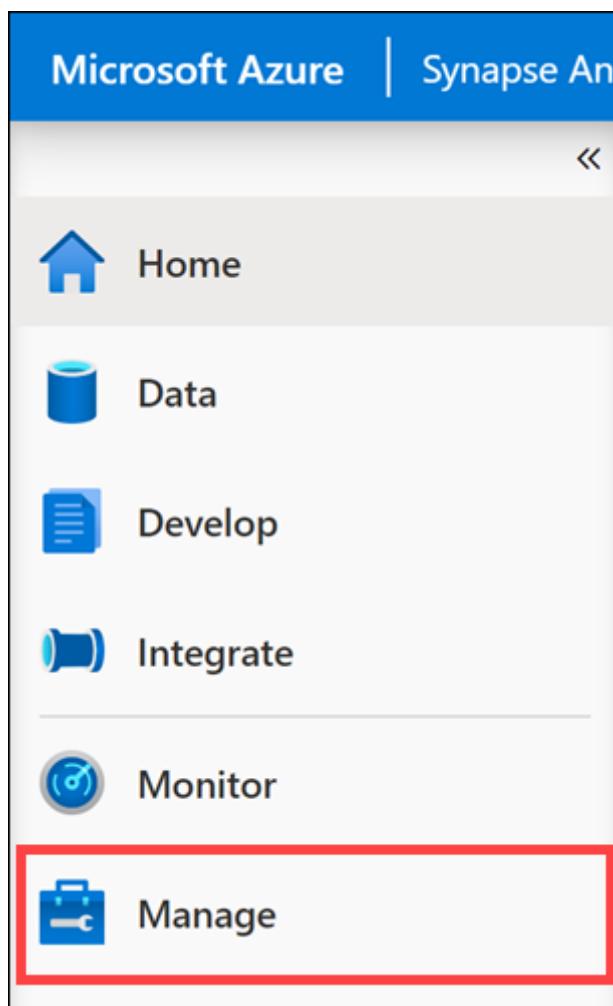
Pipeline name	Run start	Run end	Duration	Triggered by	Status
Copy December Sales	9/14/20, 7:09:04 PM	9/14/20, 7:10:04 PM	00:00:59	Manual trigger	Succeeded

9.6 Exercise 3: Cleanup

Complete these steps to free up resources you no longer need.

9.6.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

The serverless SQL pool, Built-in, is immediately available for your workspace. Dedicated SQL pools can be configured.

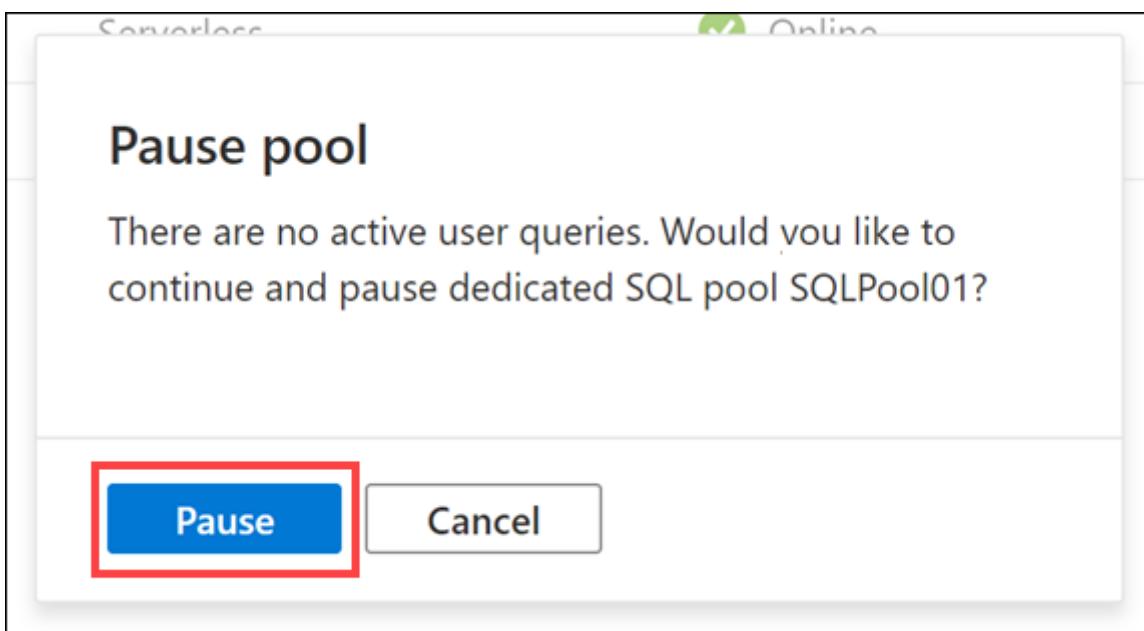
+ New Refresh System-assigned managed identity

Filter by name

Showing 1-2 of 2 items (1 Serverless, 1 Dedicated)

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

- When prompted, select **Pause**.



10 Module 8 - Transform data with Azure Data Factory or Azure Synapse Pipelines

This module teaches students how to build data integration pipelines to ingest from multiple data sources, transform data using mapping data flows and notebooks, and perform data movement into one or more data sinks.

In this module, the student will be able to:

- Execute code-free transformations at scale with Azure Synapse Pipelines
- Create data pipeline to import poorly formatted CSV files
- Create Mapping Data Flows

10.1 Lab details

- Module 8 - Transform data with Azure Data Factory or Azure Synapse Pipelines
 - Lab details
 - Lab setup and pre-requisites
 - Exercise 0: Start the dedicated SQL pool
 - Lab 1: Code-free transformation at scale with Azure Synapse Pipelines
 - Exercise 1: Create artifacts
 - Task 1: Create SQL table
 - Task 2: Create linked service

- Task 3: Create data sets
 - Task 4: Create campaign analytics dataset
- * Exercise 2: Create data pipeline to import poorly formatted CSV
 - Task 1: Create campaign analytics data flow
 - Task 2: Create campaign analytics data pipeline
 - Task 3: Run the campaign analytics data pipeline
 - Task 4: View campaign analytics table contents
- * Exercise 3: Create Mapping Data Flow for top product purchases
 - Task 1: Create Mapping Data Flow
- Lab 2: Orchestrate data movement and transformation in Azure Synapse Pipelines
 - * Exercise 1: Create, trigger, and monitor pipeline
 - Task 1: Create pipeline
 - Task 2: Trigger, monitor, and analyze the user profile data pipeline
 - * Exercise 2: Cleanup
 - Task 1: Pause the dedicated SQL pool

10.2 Lab setup and pre-requisites

Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Complete the **lab setup instructions** for this module.

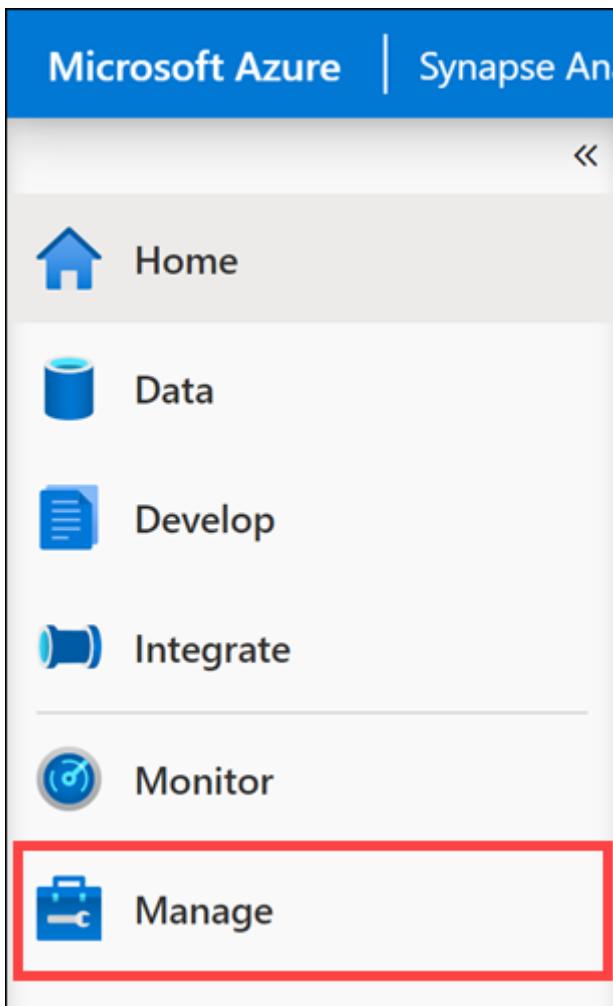
Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

10.3 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the "SQL pools" blade in the Azure portal. The left sidebar shows the following options:

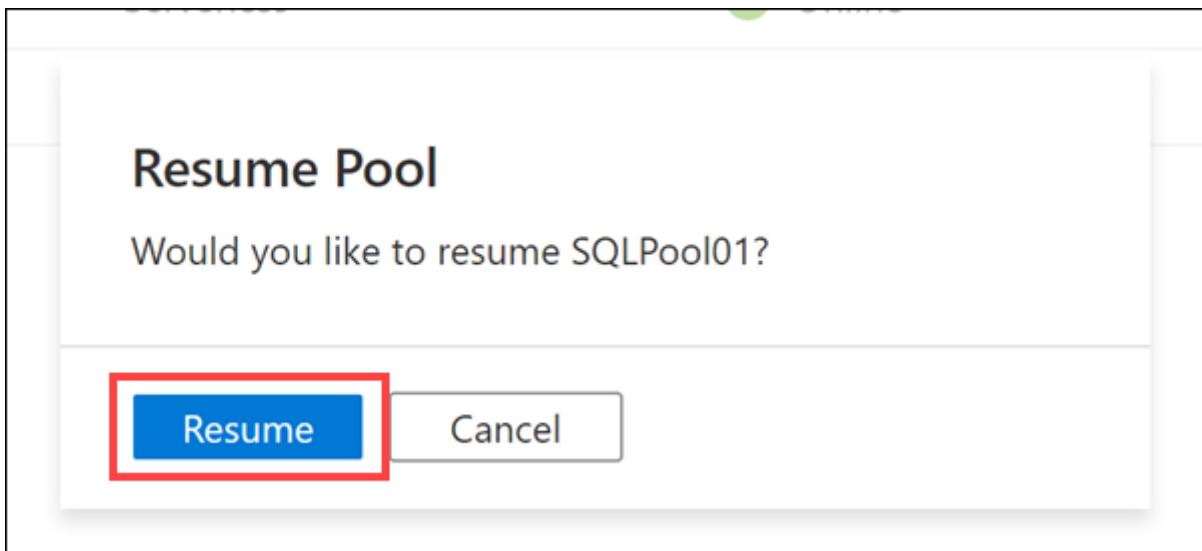
- Analytics pools
- SQL pools** (highlighted with a red box and a red number '1')
- Apache Spark pools
- External connections
- Linked services
- Azure Purview (Preview)
- Integration
- Triggers
- Integration runtimes
- Security
- Access control

The main area displays the "SQL pools" table:

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused	DW100c

A red box labeled '2' highlights the "Resume" button for the "SQLPool01" row.

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

10.4 Lab 1: Code-free transformation at scale with Azure Synapse Pipelines

Tailwind Traders would like code-free options for data engineering tasks. Their motivation is driven by the desire to allow junior-level data engineers who understand the data but do not have a lot of development experience build and maintain data transformation operations. The other driver for this requirement is to reduce fragility caused by complex code with reliance on libraries pinned to specific versions, remove code testing requirements, and improve ease of long-term maintenance.

Their other requirement is to maintain transformed data in a data lake in addition to the dedicated SQL pool. This gives them the flexibility to retain more fields in their data sets than they otherwise store in fact and dimension tables, and doing this allows them to access the data when they have paused the dedicated SQL pool, as a cost optimization.

Given these requirements, you recommend building Mapping Data Flows.

Mapping Data flows are pipeline activities that provide a visual way of specifying how to transform data, through a code-free experience. This feature offers data cleansing, transformation, aggregation, conversion, joins, data copy operations, etc.

Additional benefits

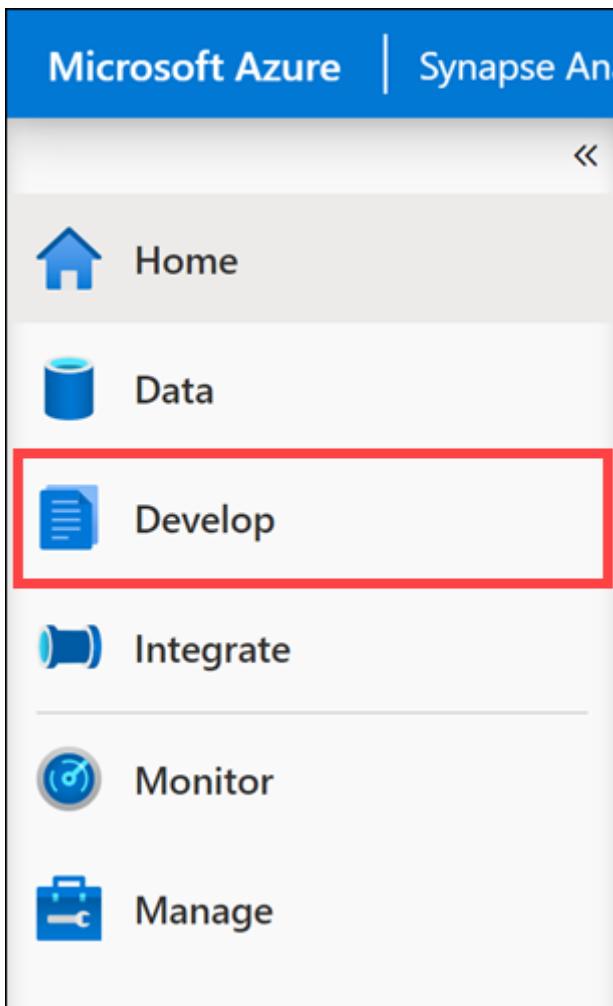
- Cloud scale via Spark execution
- Guided experience to easily build resilient data flows
- Flexibility to transform data per user's comfort
- Monitor and manage data flows from a single pane of glass

10.4.1 Exercise 1: Create artifacts

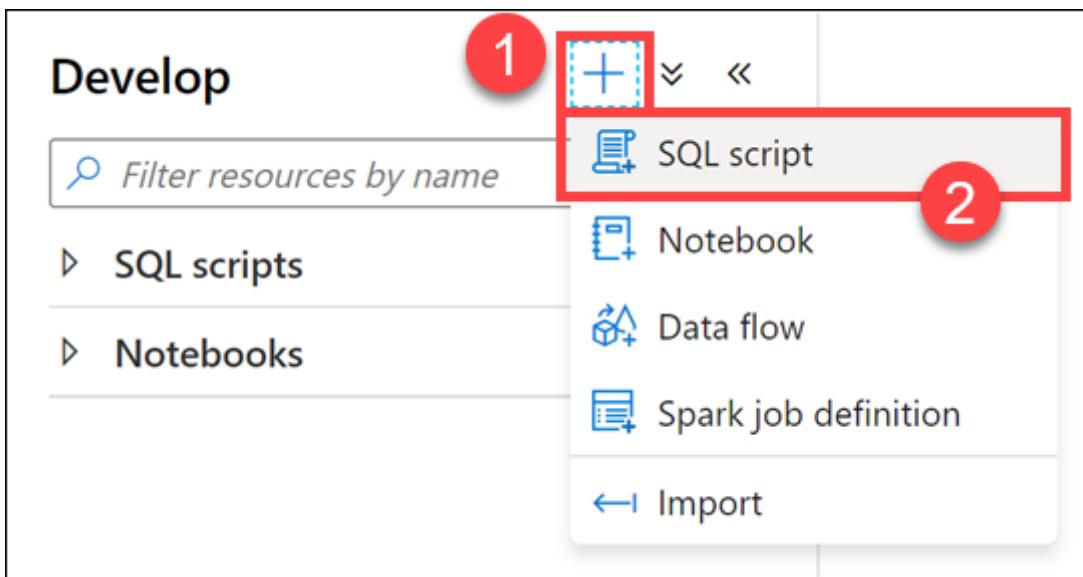
10.4.1.1 Task 1: Create SQL table

The Mapping Data Flow we will build will write user purchase data to a dedicated SQL pool. Tailwind Traders does not yet have a table to store this data. We will execute a SQL script to create this table as a pre-requisite.

1. Open Synapse Analytics Studio (<https://web.azuresynapse.net/>), and then navigate to the **Develop** hub.



2. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



3. In the toolbar menu, connect to the **SQLPool01** database to execute the query.



4. In the query window, replace the script with the following to create a new table that joins users' preferred products stored in Azure Cosmos DB with top product purchases per user from the e-commerce site, stored in JSON files within the data lake:

```
CREATE TABLE [wwi].[UserTopProductPurchases]
```

```

(
    [UserId] [int] NOT NULL,
    [ProductId] [int] NOT NULL,
    [ItemsPurchasedLast12Months] [int] NULL,
    [IsTopProduct] [bit] NOT NULL,
    [IsPreferredProduct] [bit] NOT NULL
)
WITH
(
    DISTRIBUTION = HASH ( [UserId] ),
    CLUSTERED COLUMNSTORE INDEX
)

```

- Select **Run** from the toolbar menu to execute the SQL command.



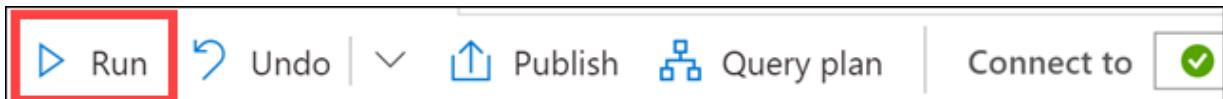
- In the query window, replace the script with the following to create a new table for the Campaign Analytics CSV file:

```

CREATE TABLE [wwi].[CampaignAnalytics]
(
    [Region] [nvarchar](50) NOT NULL,
    [Country] [nvarchar](30) NOT NULL,
    [ProductCategory] [nvarchar](50) NOT NULL,
    [CampaignName] [nvarchar](500) NOT NULL,
    [Revenue] [decimal](10,2) NULL,
    [RevenueTarget] [decimal](10,2) NULL,
    [City] [nvarchar](50) NULL,
    [State] [nvarchar](25) NULL
)
WITH
(
    DISTRIBUTION = HASH ( [Region] ),
    CLUSTERED COLUMNSTORE INDEX
)

```

- Select **Run** from the toolbar menu to execute the SQL command.

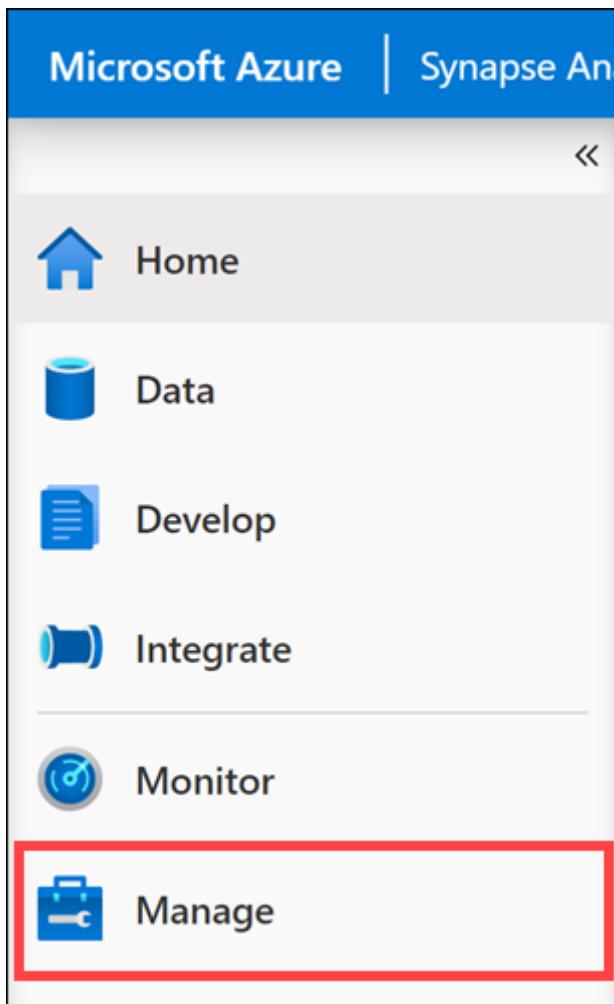


10.4.1.2 Task 2: Create linked service

Azure Cosmos DB is one of the data sources that will be used in the Mapping Data Flow. Tailwind Traders has not yet created the linked service. Follow the steps in this section to create one.

Note: Skip this section if you have already created a Cosmos DB linked service.

1. Navigate to the **Manage** hub.



2. Open **Linked services** and select **+ New** to create a new linked service. Select **Azure Cosmos DB (SQL API)** in the list of options, then select **Continue**.

The screenshot shows the 'New linked service' dialog in the Azure portal. On the left, the sidebar has 'Linked services' selected (marked with a red box and number 1). In the center, there's a yellow banner for Power BI integration. Below it, a search bar and a filter dropdown are shown. A grid of service icons is displayed, with 'Azure Cosmos DB (SQL API)' highlighted by a red box and number 3. At the bottom right of the dialog, a 'Continue' button is marked with a red box and number 4.

3. Name the linked service **asacosmosdb01** (1), select the **Cosmos DB account name** (**asacosmosdbSUFFIX**) and set the **Database name** value to **CustomerProfile** (2). Select **Test connection** to ensure success (3), then select **Create** (4).

New linked service (Azure Cosmos DB (SQL API))

Choose a name for your linked service. This name cannot be updated later.

Name * asacosmosdb01 1

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime ▼ edit

Connection string Azure Key Vault

Account selection method

From Azure subscription Enter manually

Azure subscription

Select all ▼

Azure Cosmos DB account name * asacosmosdbinaday84 2

Database name * CustomerProfile ▼ refresh

Additional connection properties

+ New

Annotations

I ..

4 Create Back

✓ Connection successful 🔗 Test connection Cancel

3

10.4.1.3 Task 3: Create data sets

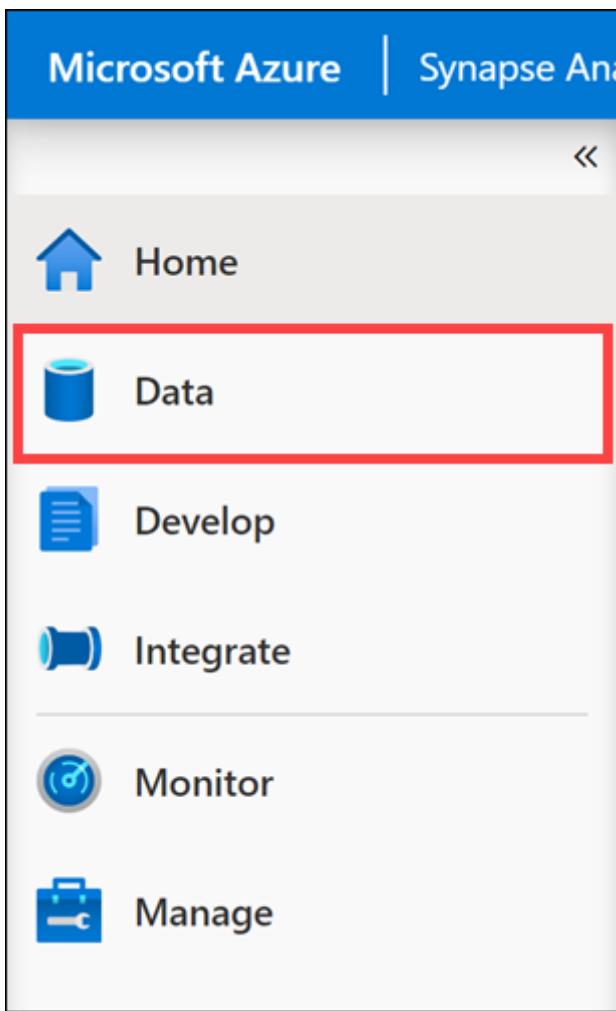
User profile data comes from two different data sources, which we will create now: `asal400_ecommerce_userprofiles_source` and `asal400_customerprofile_cosmosdb`. The customer profile data from an e-commerce system that provides top product purchases for each visitor of the site (customer) over the past 12 months is stored within JSON files in the data lake. User profile data containing, among other things, product preferences and product reviews is stored as JSON documents in Cosmos DB.

In this section, you'll create datasets for the SQL tables that will serve as data sinks for data pipelines you'll create later in this lab.

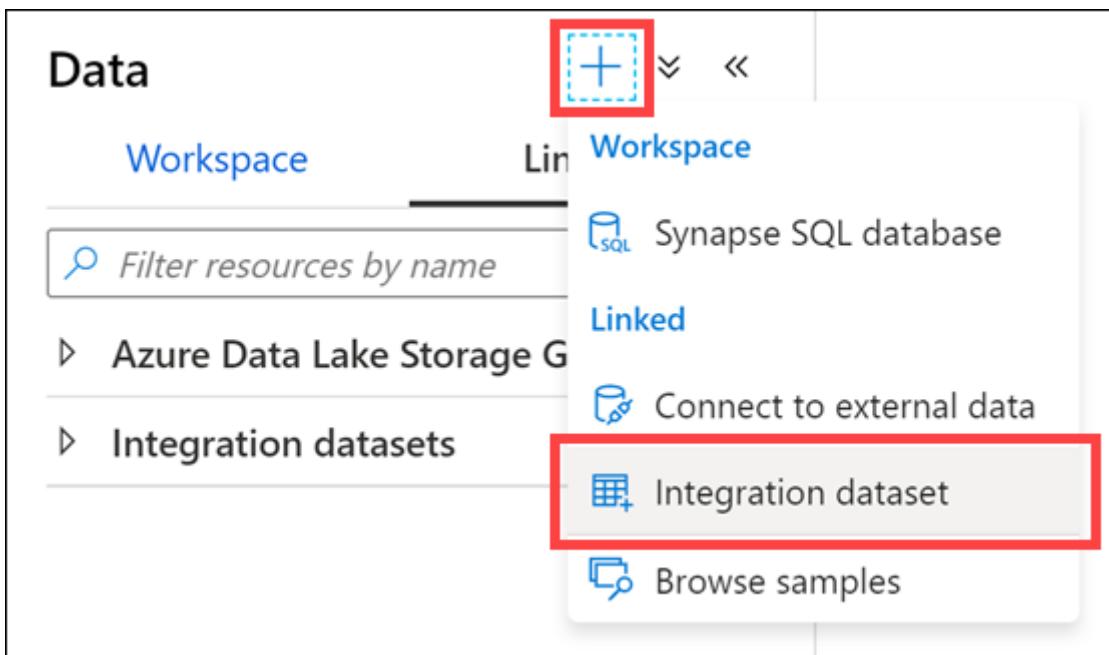
Complete the steps below to create the following two datasets: `asal400_ecommerce_userprofiles_source`

and asal400_customerprofile_cosmosdb.

1. Navigate to the **Data** hub.



2. Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



3. Select **Azure Cosmos DB (SQL API)** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

The screenshot shows the 'New integration dataset' dialog. At the top, there is a search bar with the placeholder 'Search'. Below it is a navigation bar with tabs: All, Azure, Database, File, Generic protocol, NoSQL, and Services and apps. The 'All' tab is selected. The main area displays a grid of data store icons. The 'Azure Cosmos DB (SQL API)' icon (a blue planet with a ring and stars) is highlighted with a red box and a red number '1' above it. To its right is the 'Apache Impala' icon (a blue cube with a white star). Below these are 'Azure Blob Storage' (a blue square with a white triangle), 'Azure Data Explorer (Kusto)' (a blue square with a white lightning bolt), and 'Azure Data Lake Storage Gen1' (a blue folder with a white lightning bolt). In the second row, there are three more icons: 'Azure Data Lake Storage Gen2' (a grey square with blue and white squares), 'Azure Database for MariaDB' (a blue cylinder with a white seal), and 'Azure Database for MySQL' (a blue cylinder with a white 'My'). In the bottom row, there are three more icons: 'Azure Synapse Analytics' (a blue cylinder with a white face), 'Azure Databricks' (a blue triangle with a white 'A'), and 'Azure Data Factory' (a blue square with a white document icon). A red box surrounds the 'Continue' button at the bottom left, and a red number '2' is placed above it. A 'Cancel' button is located at the bottom right.

4. Configure the dataset with the following characteristics, then select **OK** (4):

- **Name:** Enter `asal400_customerprofile_cosmosdb` (1).
- **Linked service:** Select the Azure Cosmos DB linked service (2).
- **Collection:** Select `OnlineUserProfile01` (3).

Set properties

Choose a name for your dataset. This name can be updated at any time until it is published.

1 Name
asal400_customerprofile_cosmosdb

2 Linked service *
asacosmosdb01

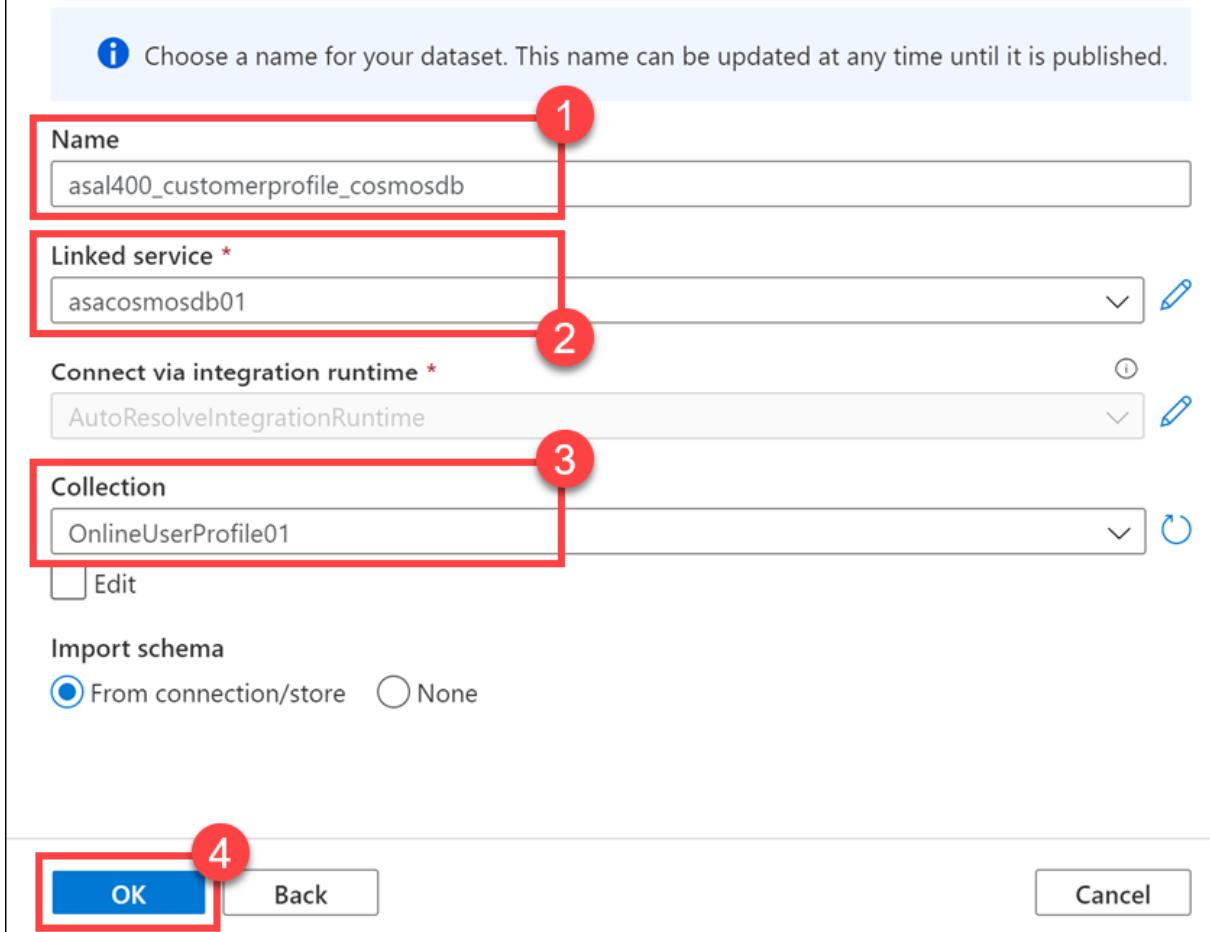
Connect via integration runtime *
AutoResolveIntegrationRuntime

3 Collection
OnlineUserProfile01

Edit

Import schema
 From connection/store None

4 OK Back Cancel



- After creating the dataset, select **Preview data** under its **Connection** tab.

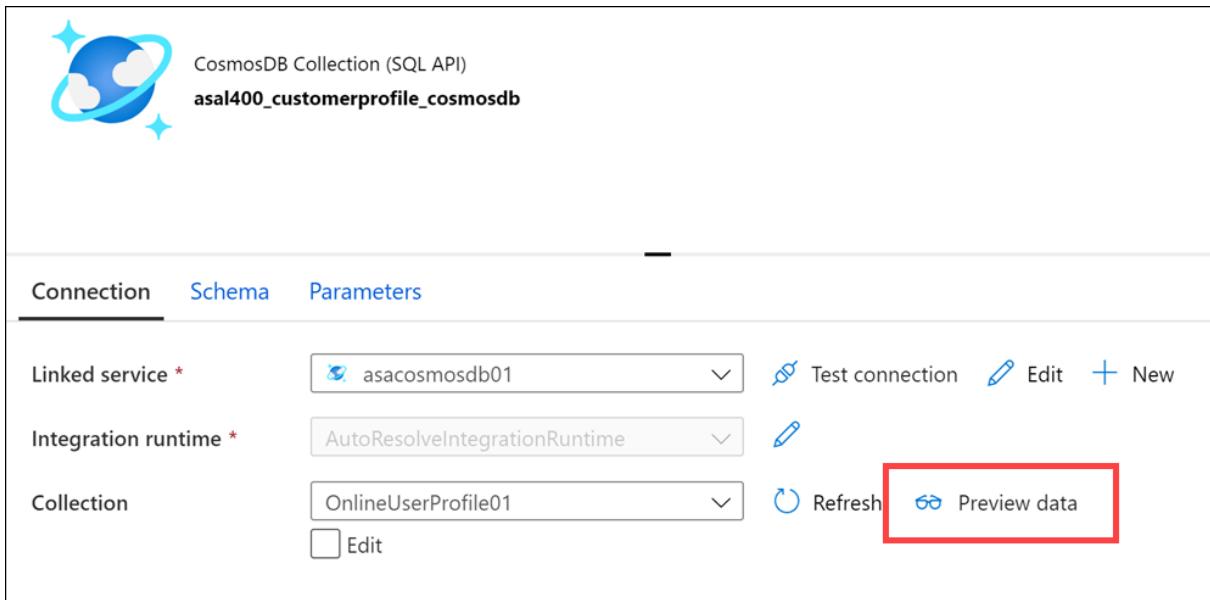
CosmosDB Collection (SQL API)
asal400_customerprofile_cosmosdb

Connection Schema Parameters

Linked service * asacosmosdb01 Test connection Edit + New

Integration runtime * AutoResolveIntegrationRuntime

Collection OnlineUserProfile01 Refresh **Preview data**



- Preview data queries the selected Azure Cosmos DB collection and returns a sample of the documents within. The documents are stored in JSON format and include a `userId` field, `cartId`, `preferredProducts` (an array of product IDs that may be empty), and `productReviews` (an array of written product reviews that may be empty).

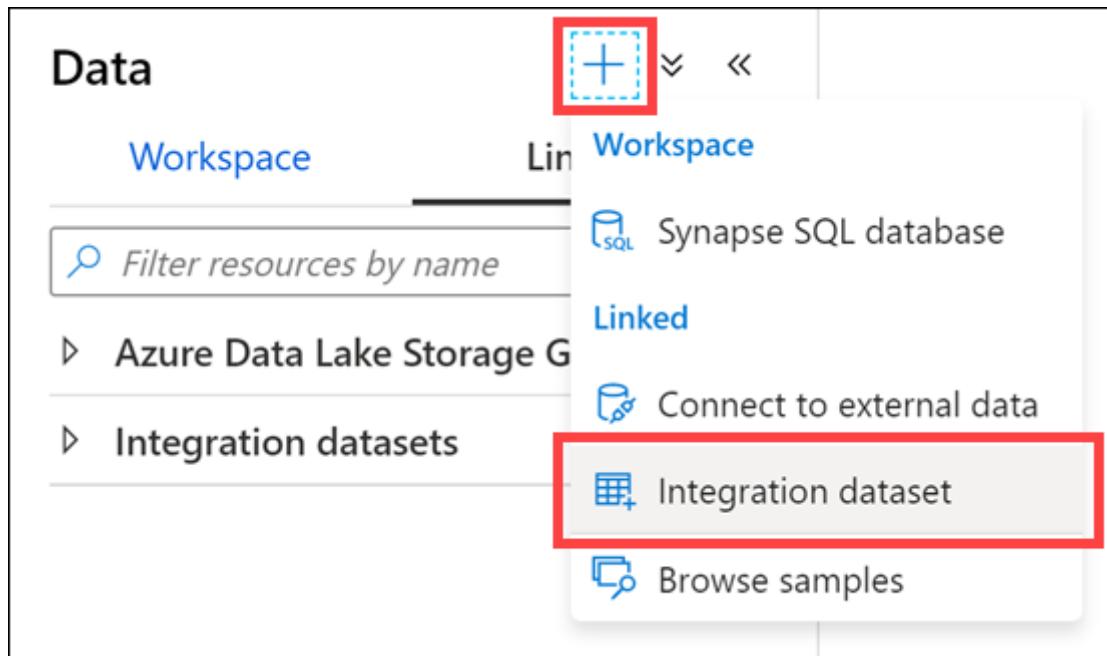
Preview data

Linked service: asacosmosdb01

Object: OnlineUserProfile01

```
[
  {
    "userId": 4193,
    "cartId": "2db8e371-dac3-4318-afc4-3d2fae5334e8",
    "preferredProducts": [
      1747,
      4740,
      315,
      4337,
      2913,
      4891
    ],
    "productReviews": [
      {
        "productId": 3052,
        "reviewText": "heard about this on brazilian radio, decided to give it a try.",
        "reviewDate": "2019-06-08T22:42:56.0052067+00:00"
      },
      {
        "productId": 1360,
        "reviewText": "i use it daily when i'm in my courthouse.",
        "reviewDate": "2017-04-29T08:43:26.6771565+00:00"
      }
    ]
  }
]
```

- Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



- Select **Azure Data Lake Storage Gen2** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store



All Azure Database File Generic protocol NoSQL Services and apps

 Azure Cosmos DB (SQL API)	 Azure Data Explorer (Kusto)	 Azure Data Lake Storage Gen1
 Azure Data Lake Storage Gen2	 Azure Database for MariaDB	 Azure Database for MySQL
 Azure Database for PostgreSQL	 Azure Databricks Delta Lake	 Azure File Storage
Continue		Cancel

1

2

9. Select the **JSON** format (1), then select **Continue** (2).

Select format

Choose the format type of your data



Avro



Binary



DelimitedText



Excel



Json



ORC



Parquet



XML

2
Continue

Back

Cancel

10. Configure the dataset with the following characteristics, then select **OK (5)**:

- **Name:** Enter `asal400_ecommerce_userprofiles_source` (1).
- **Linked service:** Select the `asadatalakeXX` linked service that already exists (2).
- **File path:** Browse to the `wwi-02/online-user-profiles-02` path (3).
- **Import schema:** Select `From connection/store` (4).

Set properties

i Choose a name for your dataset. This name can be updated at any time until it is published.

1 Name
asal400_ecommerce_userprofiles_source

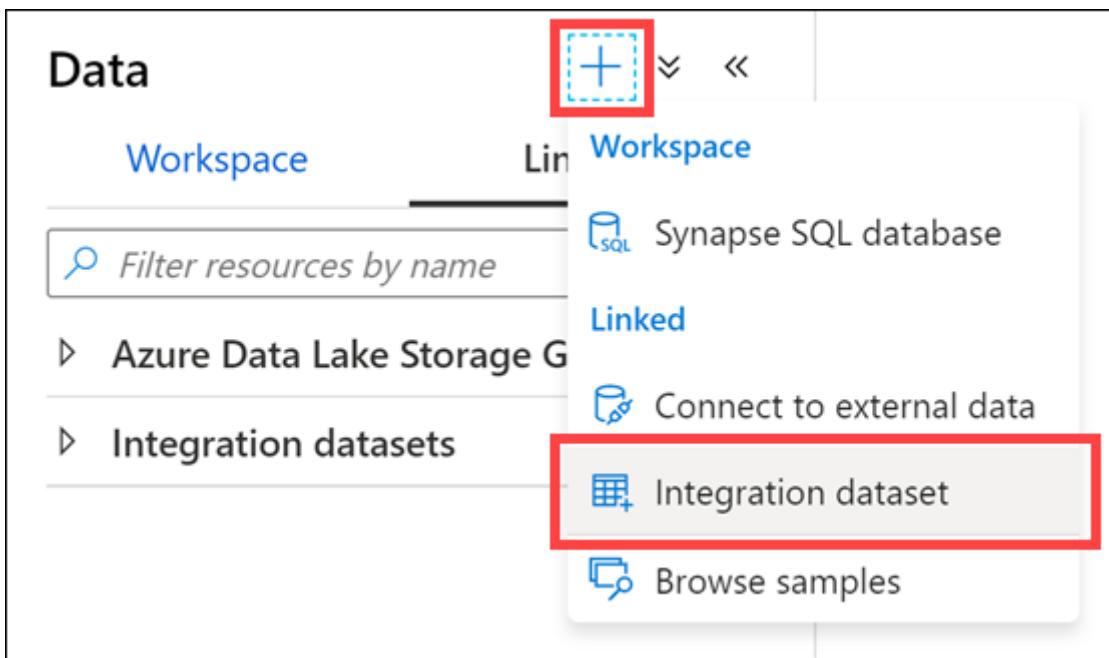
2 Linked service *
asadatalakeinaday84

3 File path
wwi-02 / online-user-profiles-02 / File

4 Import schema
 From connection/store From sample file None

5 OK Back Cancel

11. Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



12. Select **Azure Synapse Analytics** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

The screenshot shows a grid of data store options:

- Azure SQL Database**: SQL icon, blue background.
- Azure SQL Database Managed Instance**: SQL icon, grey background.
- Azure Search**: Magnifying glass icon.
- Azure Synapse Analytics**: House and cylinder icon, highlighted with a red box and a red circle labeled '1'.
- Azure Synapse dedicated SQL pool**: House and cylinder icon.
- Azure Table Storage**: Table icon.
- Cassandra**: Eye icon.
- Common Data Service for Apps**: Database icon.
- Concur (Preview)**: Yellow 'C' icon.

At the bottom left is a **Continue** button, which is also highlighted with a red box and a red circle labeled '2'. At the bottom right is a **Cancel** button.

13. Configure the dataset with the following characteristics, then select **OK** (5):

- **Name:** Enter `asal400_wwi_campaign_analytics_asa` (1).
- **Linked service:** Select the `SqlPool01` service (2).
- **Table name:** Select `wwi.CampaignAnalytics` (3).
- **Import schema:** Select `From connection/store` (4).

Set properties

1. Name: asal400_wwi_campaign_analytics_asa

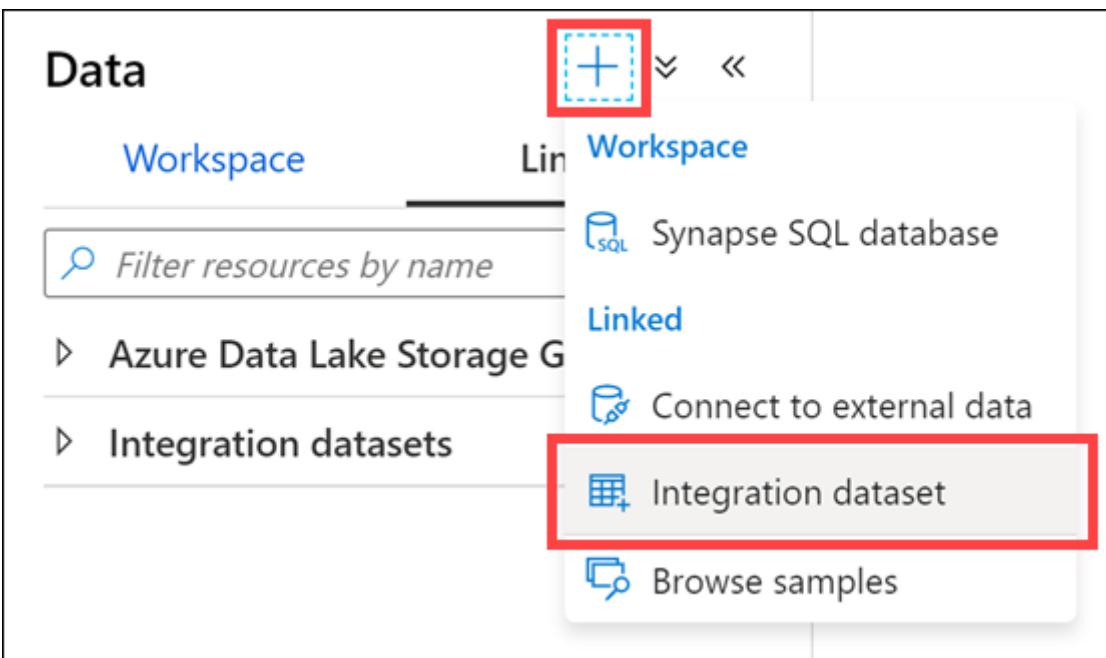
2. Linked service: sqlpool01

3. Table name: wwi.CampaignAnalytics

4. Import schema: From connection/store (radio button selected)

5. OK button

14. Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



15. Select **Azure Synapse Analytics** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

The screenshot shows a grid of data store options:

- Azure SQL Database
- Azure SQL Database Managed Instance
- Azure Search
- Azure Synapse Analytics (highlighted with a red box and a red circle labeled '1')
- Azure Synapse dedicated SQL pool
- Azure Table Storage
- Cassandra
- Common Data Service for Apps
- Concur (Preview)

At the bottom left is a blue 'Continue' button with a red box and a red circle labeled '2'. At the bottom right is a 'Cancel' button.

16. Configure the dataset with the following characteristics, then select **OK** (5):

- **Name:** Enter `asal400_wwi_usertopproductpurchases_asa` (1).
- **Linked service:** Select the `SqlPool01` service (2).
- **Table name:** Select `wwi.UserTopProductPurchases` (3).
- **Import schema:** Select `From connection/store` (4).

Set properties

i Choose a name for your dataset. This name can be updated at any time until it is published.

1 Name
asal400_wwi_usertopproductpurchases_asa

2 Linked service *
sqlpool01

3 Table name
wwi.UserTopProductPurchases

4 Import schema
 From connection/store None

5 OK Back Cancel

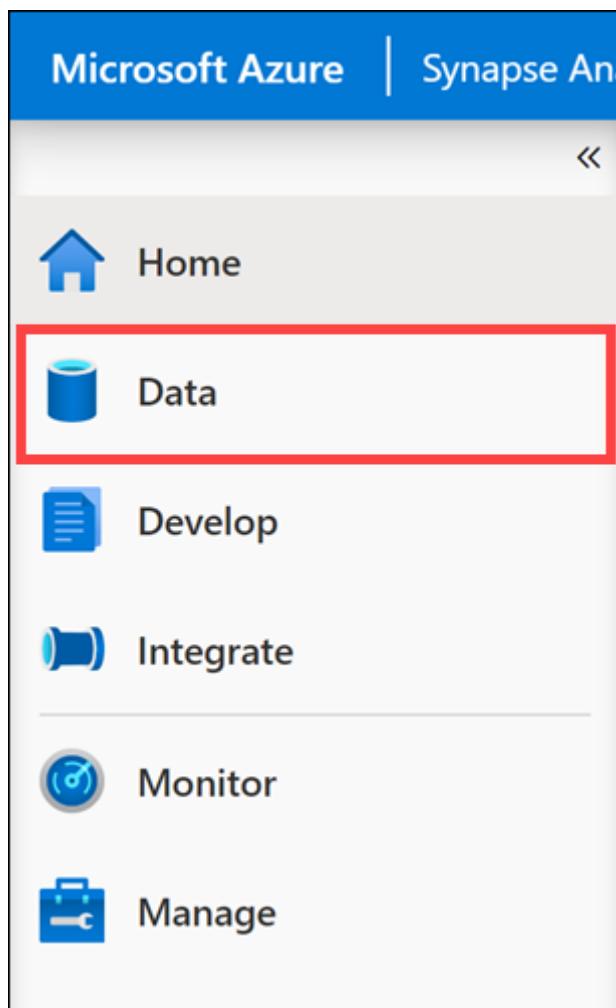
10.4.1.4 Task 4: Create campaign analytics dataset

Your organization was provided a poorly formatted CSV file containing marketing campaign data. The file was uploaded to the data lake and now it must be imported into the data warehouse.

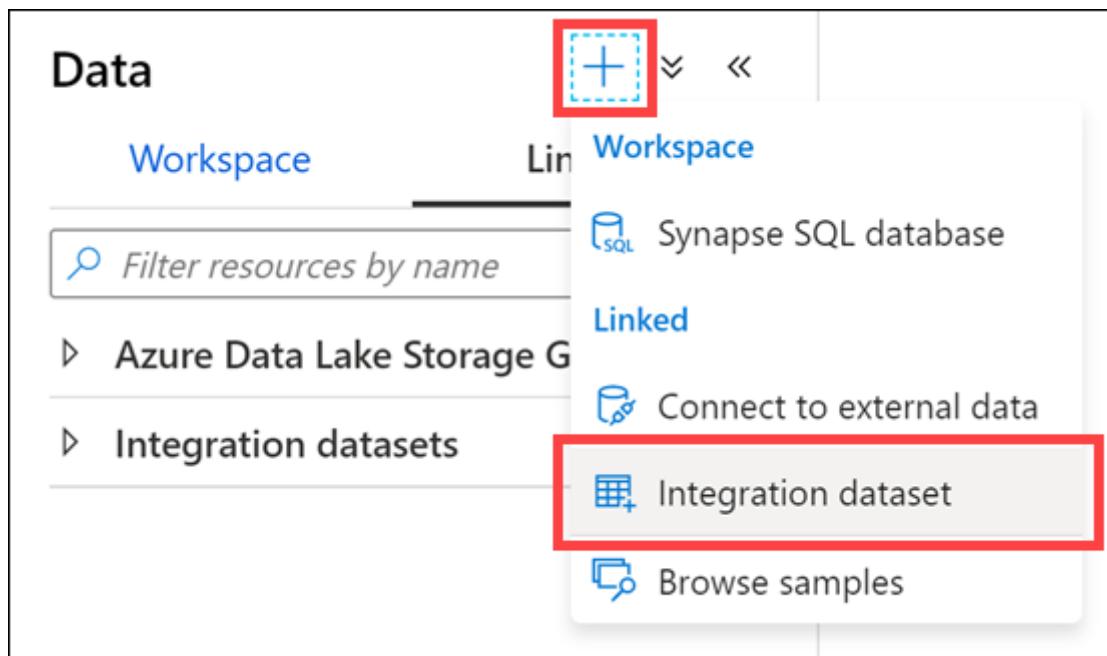
Region	Country	Product_Category	Campaign_Name	Revenue	Revenue_T	City	State	
Europe	Germany	Apparel and Footwear	Fun with Colors	\$14\	865 \$15\		960	
Far West	US	Books	EnjoyTheMoment; \$14\	992 \$15\		699	San Diego	California
Europe	Germany	Apparel and Footwear	Fall into Winter	\$5\	117 \$8\		713	
Far West	US	Books	EnjoyTheMoment; \$9\	935 \$15\		232	San Diego	California
Europe	France	Apparel and Footwear	Enjoy the Moment \$13\	221 \$8\		584		
Far West	US	Books	EnjoyTheMoment; \$15\	119 \$17\		269	San Diego	California
Europe	UK	Lighting	Fall into Winter	\$5\	117 \$9\		305	
Far West	US	Books	EnjoyTheMoment; \$15\	740 \$7\		685	San Diego	California
South America	Mexico	Electronics	Be Unique	\$16\	240 \$16\		38	
Far West	US	Books	EnjoyTheMoment; \$14\	778 \$13\		122	San Diego	California
North & Central America	USA	D&Cor	Spring into Summe\$6\	689 \$13\		88		
Far West	US	Books	EnjoyTheMoment; \$10\	296 \$7\		313	San Diego	California
South America	Mexico	Apparel and Footwear	Enjoy the Moment \$13\	98 \$5\		663		
Far West	US	Books	EnjoyTheMoment; \$14\	605 \$18\		971	San Diego	California
South America	Brazil	D&Cor	Fun with Colors	\$15\	142 \$7\		147	
Far West	US	Books	EnjoyTheMoment; \$14\	328 \$15\		577	San Diego	California
South America	Mexico	Exercise	Spring into Summe\$17\	637 \$6\		876		
Far West	US	Books	EnjoyTheMoment; \$11\	247 \$10\		339	San Diego	California
South America	Mexico	D&Cor	Be Unique	\$8\	284 \$9\		840	

Issues include invalid characters in the revenue currency data, and misaligned columns.

1. Navigate to the **Data** hub.



2. Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



3. Select **Azure Data Lake Storage Gen2** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

Search

All Azure Database File Generic protocol NoSQL Services and apps

 Azure Cosmos DB (SQL API)	 Azure Data Explorer (Kusto)	 Azure Data Lake Storage Gen1
 Azure Data Lake Storage Gen2	 Azure Database for MariaDB	 Azure Database for MySQL
 Azure Database for PostgreSQL	 Azure Databricks Delta Lake	 Azure File Storage
Continue		Cancel

1

2

4. Select the **DelimitedText** format (1), then select **Continue** (2).

Select format

Choose the format type of your data

 Avro	 Binary	 DelimitedText 1
 Excel	 Json	 ORC
 Parquet	 XML	

Continue 2 **Back** **Cancel**

5. Configure the dataset with the following characteristics, then select **OK** (6):

- **Name:** Enter `asal400_campaign_analytics_source` (1).
- **Linked service:** Select the `asadatalakeSUFFIX` linked service (2).
- **File path:** Browse to the `wwi-02/campaign-analytics/campaignanalytics.csv` path (3).
- **First row as header:** Leave unchecked (4). We are skipping the header because there is a mismatch between the number of columns in the header and the number of columns in the data rows.
- **Import schema:** Select `From connection/store` (5).

Set properties

1 Choose a name for your dataset. This name can be updated at any time until it is published.

2 Linked service *

3 File path

4 First row as header

5 Import schema

6 OK

6. After creating the dataset, navigate to its **Connection** tab. Leave the default settings. They should match the following configuration:

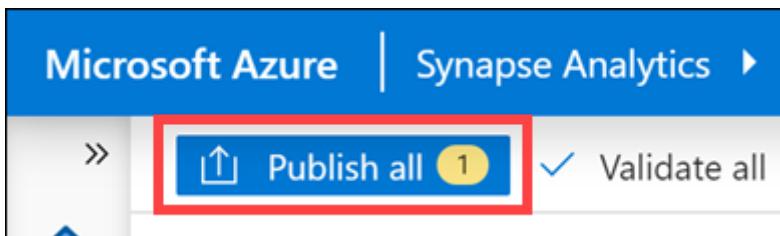
- Compression type:** Select none.
- Column delimiter:** Select Comma (,).
- Row delimiter:** Select Default (\r,\n, or \r\n).
- Encoding:** Select ‘Default(UTF-8)’.
- Escape character:** Select Backslash (\).
- Quote character:** Select Double quote (").
- First row as header:** Leave unchecked.
- Null value:** Leave the field empty.

Connection	Schema	Parameters
Linked service *	asadatalake356357	Test connection Edit New Learn more
File path *	wwi-02 / campaign-analytics / campaignanalytics.csv	Browse Preview data
Compression type	None	
Column delimiter ⓘ	Comma (,)	Edit
Row delimiter ⓘ	Default (\r,\n, or \r\n)	Edit
Encoding	Default(UTF-8)	
Escape character	Backslash (\)	Edit
Quote character	Double quote (")	Edit
First row as header	<input type="checkbox"/>	

7. Select **Preview data**.
8. Preview data displays a sample of the CSV file. You can see some of the issues shown in the screenshot at the beginning of this task. Notice that since we are not setting the first row as the header, the header columns appear as the first row. Also, notice that the city and state values seen in the earlier screenshot do not appear. This is because of the mismatch in the number of columns in the header row compared to the rest of the file. We will exclude the first row when we create the data flow in the next exercise.

Prop_0	Prop_1	Prop_2	Prop_3	Prop_4	Prop_5	Prop_6	Prop_7
Region	Country	Product_Category	Campaign_Name	Revenue	Revenue_Target	City	State
Europe	Germany	Apparel and Footwear	Fun with Colors	\$14\	865.00	\$15\	960.00
Far West	US	Books	EnjoyTheMoment; BeUnique; TailoredForYou	\$14\	992.00	\$15\	699.00
Europe	Germany	Apparel and Footwear	Fall into Winter	\$5\	117.00	\$8\	713.00
Far West	US	Books	EnjoyTheMoment; BeUnique; TailoredForYou	\$9\	935.00	\$15\	232.00
Europe	France	Apparel and Footwear	Enjoy the Moment	\$13\	221.00	\$8\	584.00
Far West	US	Books	EnjoyTheMoment; BeUnique;	110.00	110.00	110.00	110.00

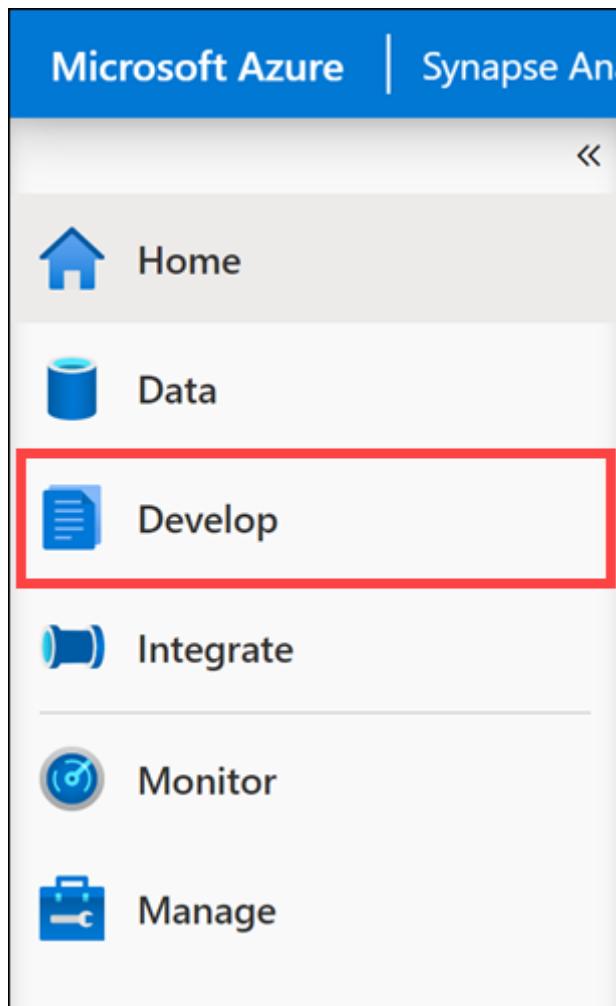
9. Select **Publish all** then **Publish** to save your new resources.



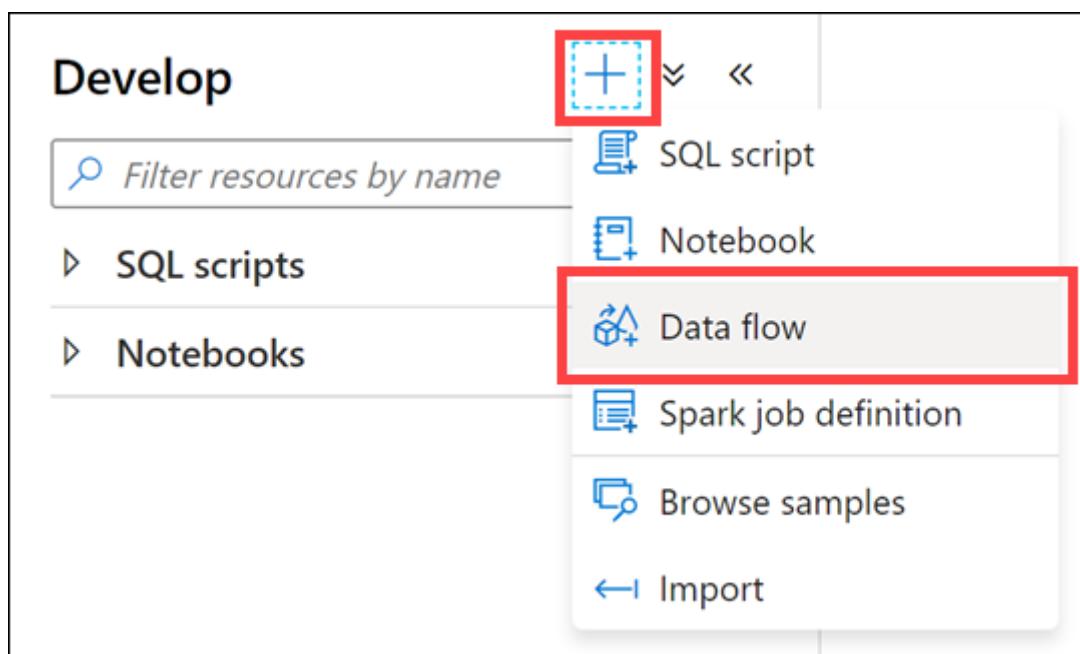
10.4.2 Exercise 2: Create data pipeline to import poorly formatted CSV

10.4.2.1 Task 1: Create campaign analytics data flow

1. Navigate to the **Develop** hub.



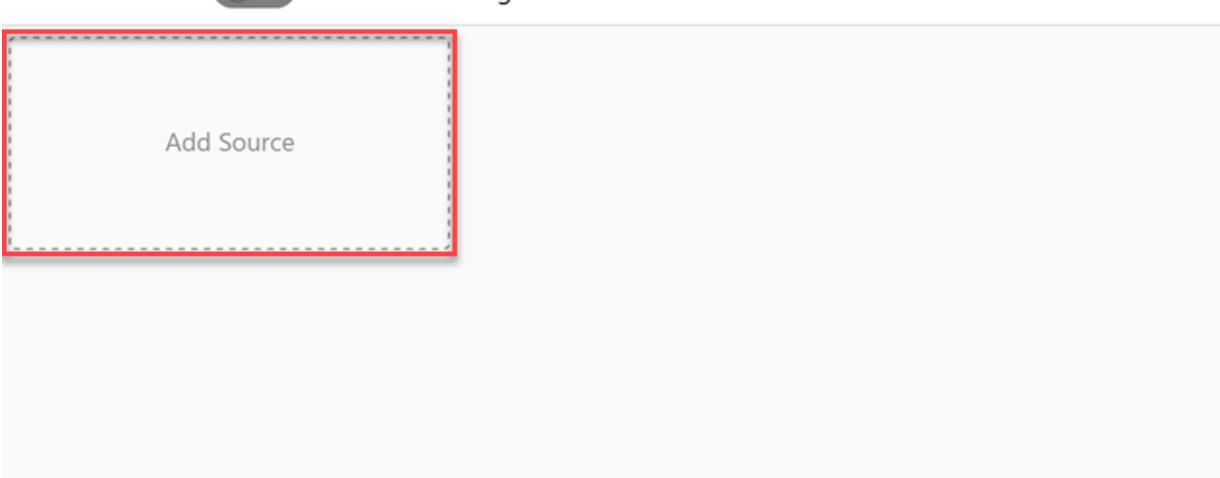
2. Select + then **Data flow** to create a new data flow.



3. In the **General** settings of the **Properties** blade of the new data flow, update the **Name** to the following:
asal400_lab2_writecampaignanalyticstoasa.



4. Select **Add Source** on the data flow canvas.

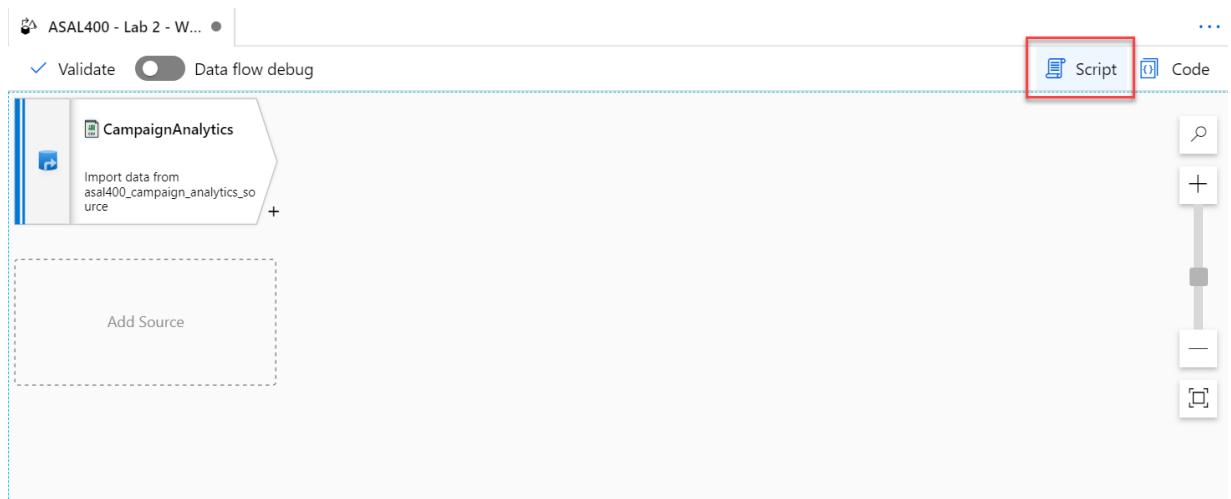


5. Under **Source settings**, configure the following:

- **Output stream name:** Enter CampaignAnalytics.
- **Source type:** Select Dataset.
- **Dataset:** Select asal400_campaign_analytics_source.
- **Options:** Select Allow schema drift and leave the other options unchecked.
- **Skip line count:** Enter 1. This allows us to skip the header row which has two fewer columns than the rest of the rows in the CSV file, truncating the last two data columns.
- **Sampling:** Select Disable.

Source settings	Source options	Projection	Optimize	Inspect	Data preview
Output stream name * CampaignAnalytics					Learn more
Source type *	<input checked="" type="button"/> Dataset	<input type="button"/> Inline			
Dataset *	<input type="button"/> asal400_campaign_analytics_source Test connection Open New				
Options	<input checked="" type="checkbox"/> Allow schema drift ① <input type="checkbox"/> Infer drifted column types ① <input type="checkbox"/> Validate schema ①				
Skip line count	1				
Sampling * ①	<input type="radio"/> Enable <input checked="" type="radio"/> Disable				

6. When you create data flows, certain features are enabled by turning on debug, such as previewing data and importing a schema (projection). Due to the amount of time it takes to enable this option, as well as environmental constraints of the lab environment, we will bypass these features. The data source has a schema we need to set. To do this, select **Script** above the design canvas.



7. Replace the script with the following to provide the column mappings (**output**), then select **OK**:

```
source(output
    {_col0_} as string,
    {_col1_} as string,
    {_col2_} as string,
    {_col3_} as string,
    {_col4_} as string,
    {_col5_} as double,
    {_col6_} as string,
    {_col7_} as double,
    {_col8_} as string,
    {_col9_} as string
),
allowSchemaDrift: true,
validateSchema: false,
ignoreNoFilesFound: false) ~> CampaignAnalytics
```

Your script should match the following:

Data flow name

ASAL400 - Lab 2 - Write Campaign Analytic

```
1 source(output(
2   |   {_col0_} as string,
3   |   {_col1_} as string,
4   |   {_col2_} as string,
5   |   {_col3_} as string,
6   |   {_col4_} as string,
7   |   {_col5_} as double,
8   |   {_col6_} as string,
9   |   {_col7_} as double,
10  |   {_col8_} as string,
11  |   {_col9_} as string
12  ),
13  allowSchemaDrift: true,
14  validateSchema: false,
15  skipLines: 1) ~> CampaignAnalytics
```

OK

Copy as single line

Cancel

8. Select the **CampaignAnalytics** data source, then select **Projection**. The projection should display the following schema:

Column name	Type	Format
col0	abc string	Specify format
col1	abc string	Specify format
col2	abc string	Specify format
col3	abc string	Specify format
col4	abc string	Specify format
col5	1.2 double	Specify format
col6	abc string	Specify format
col7	1.2 double	Specify format
col8	abc string	Specify format
col9	abc string	Specify format

9. Select the + to the right of the **CampaignAnalytics** source, then select the **Select** schema modifier from the context menu.

The screenshot shows the Alteryx workspace. At the top, there are several buttons: Validate (checked), Data flow debug (unchecked), and Debug Settings. Below this, the 'CampaignAnalytics' source is selected, showing its columns: 10 total. To the right of the source is a context menu with a red box highlighting the '+' icon. A dashed box encloses the 'Add Source' button and the context menu items. The context menu items include 'Schema modifier' (which is expanded), 'Derived Column', 'Select' (which is highlighted with a red box), and 'Aggregate'.

10. Under **Select settings**, configure the following:

- **Output stream name:** Enter **MapCampaignAnalytics**.
- **Incoming stream:** Select **CampaignAnalytics**.
- **Options:** Check both options.
- **Input columns:** make sure **Auto mapping** is unchecked, then provide the following values in the **Name as** fields:
 - Region
 - Country
 - ProductCategory
 - CampaignName
 - RevenuePart1
 - Revenue
 - RevenueTargetPart1

- RevenueTarget
- City
- State

Select settings Optimize Inspect Data preview ●

Output stream name * MapCampaignAnalytics [Learn more](#)

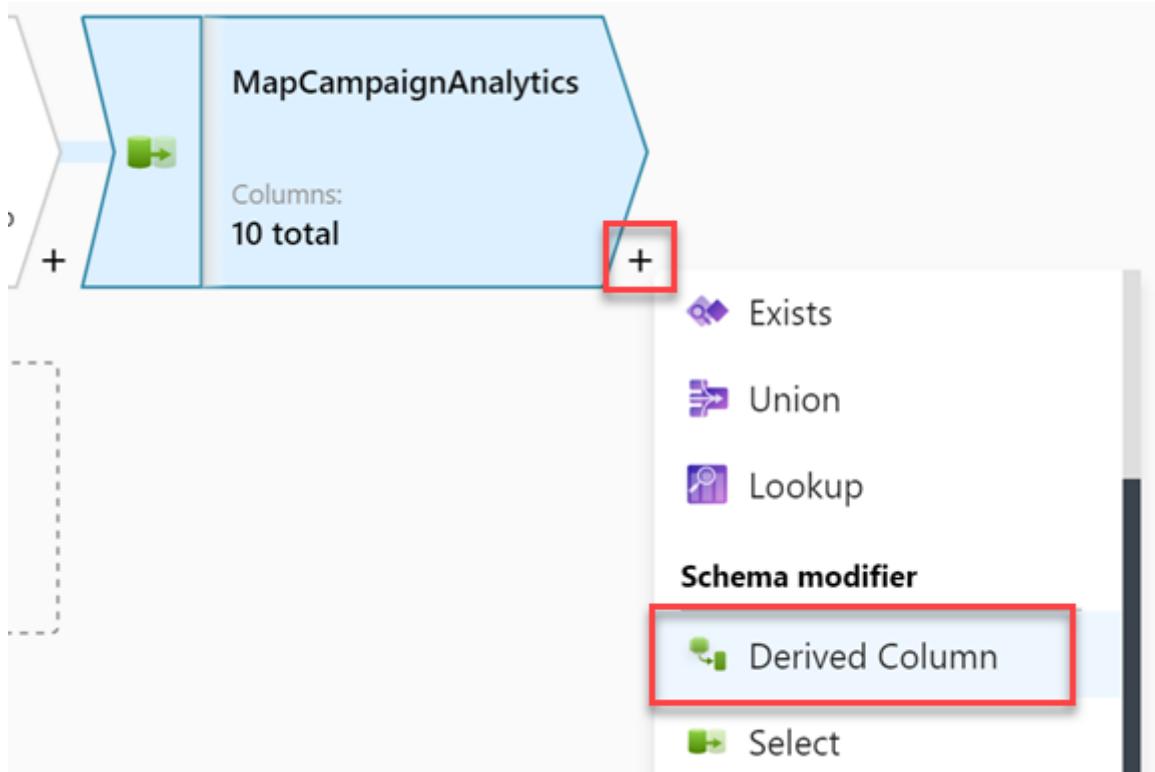
Incoming stream * CampaignAnalytics

Options
 Skip duplicate input columns
 Skip duplicate output columns

Input columns * [Auto mapping](#) [Reset](#) [Add mapping](#) [Delete](#) 10 mappings: All in

CampaignAnalytics's column	Name as
abc_col0_	Region
abc_col1_	Country
abc_col2_	ProductCategory
abc_col3_	CampaignName
abc_col4_	RevenuePart1
1.2_col5_	Revenue
abc_col6_	RevenueTargetPart1
1.2_col7_	RevenueTarget
abc_col8_	City
abc_col9_	State

11. Select the + to the right of the MapCampaignAnalytics source, then select the **Derived Column** schema modifier from the context menu.



12. Under **Derived column's settings**, configure the following:

- **Output stream name:** Enter ConvertColumnTypesAndValues.
- **Incoming stream:** Select MapCampaignAnalytics.
- **Columns:** Provide the following information:

Column	Expression
Revenue	toDecimal(replace(concat(toString(RevenuePart1), toString(Revenue)), '\\\\', ''), 10, 2, 'e')
RevenueTarget	toDecimal(replace(concat(toString(RevenueTargetPart1), toString(RevenueTarget)), '\\\\', ''), 10, 2, 'e')

Note: To insert the second column, select **+** **Add** above the Columns list, then select **Add column**.

Derived column's settings Optimize Inspect Data preview ●

Output stream name * ConvertColumnTypeAndValues Learn more ↗

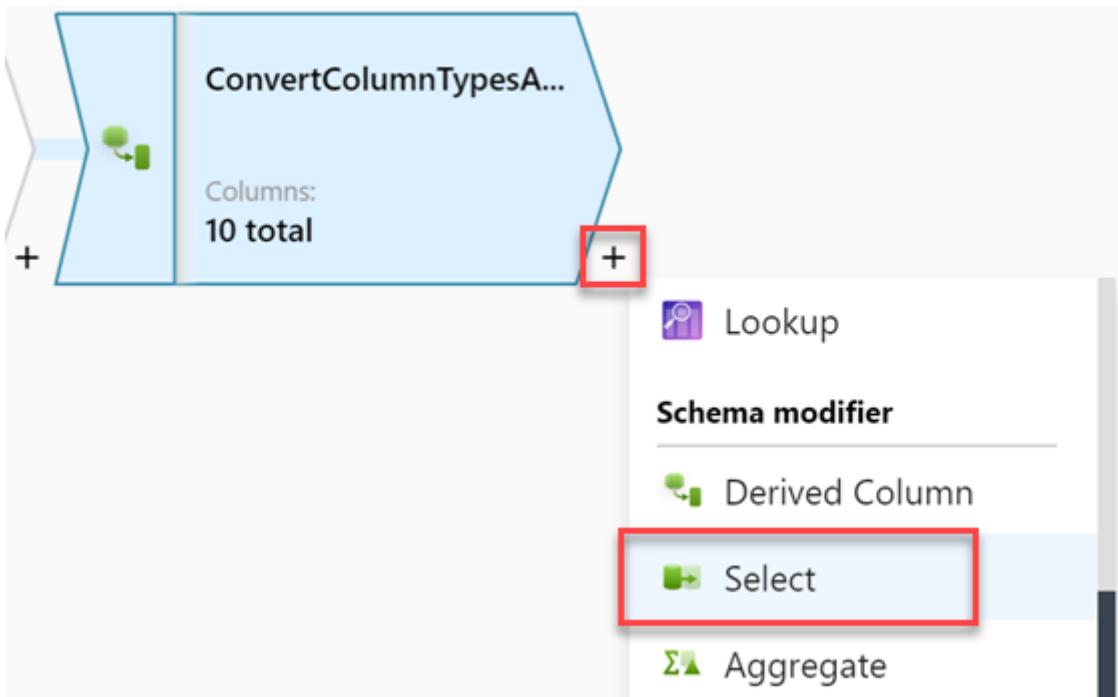
Incoming stream * MapCampaignAnalytics

Columns * ⓘ

+ Add Duplicate Delete

Column	Expression
Revenue	toDecimal(replace(concat(toString(RevenuePart1), t... e ^x))
RevenueTarget	toDecimal(replace(concat(toString(RevenueTargetP... e ^x))

13. Select the **+** to the right of the **ConvertColumnTypeAndValues** step, then select the **Select** schema modifier from the context menu.



14. Under **Select settings**, configure the following:

- **Output stream name:** Enter **SelectCampaignAnalyticsColumns**.
- **Incoming stream:** Select **ConvertColumnTypeAndValues**.
- **Options:** Check both options.
- **Input columns:** make sure **Auto mapping** is unchecked, then **Delete RevenuePart1** and **RevenueTargetPart1**. We no longer need these fields.

Select settings Optimize Inspect Data preview ●

Output stream name * SelectCampaignAnalyticsColumns [Learn more](#)

Incoming stream * ConvertColumnTypeAndValues

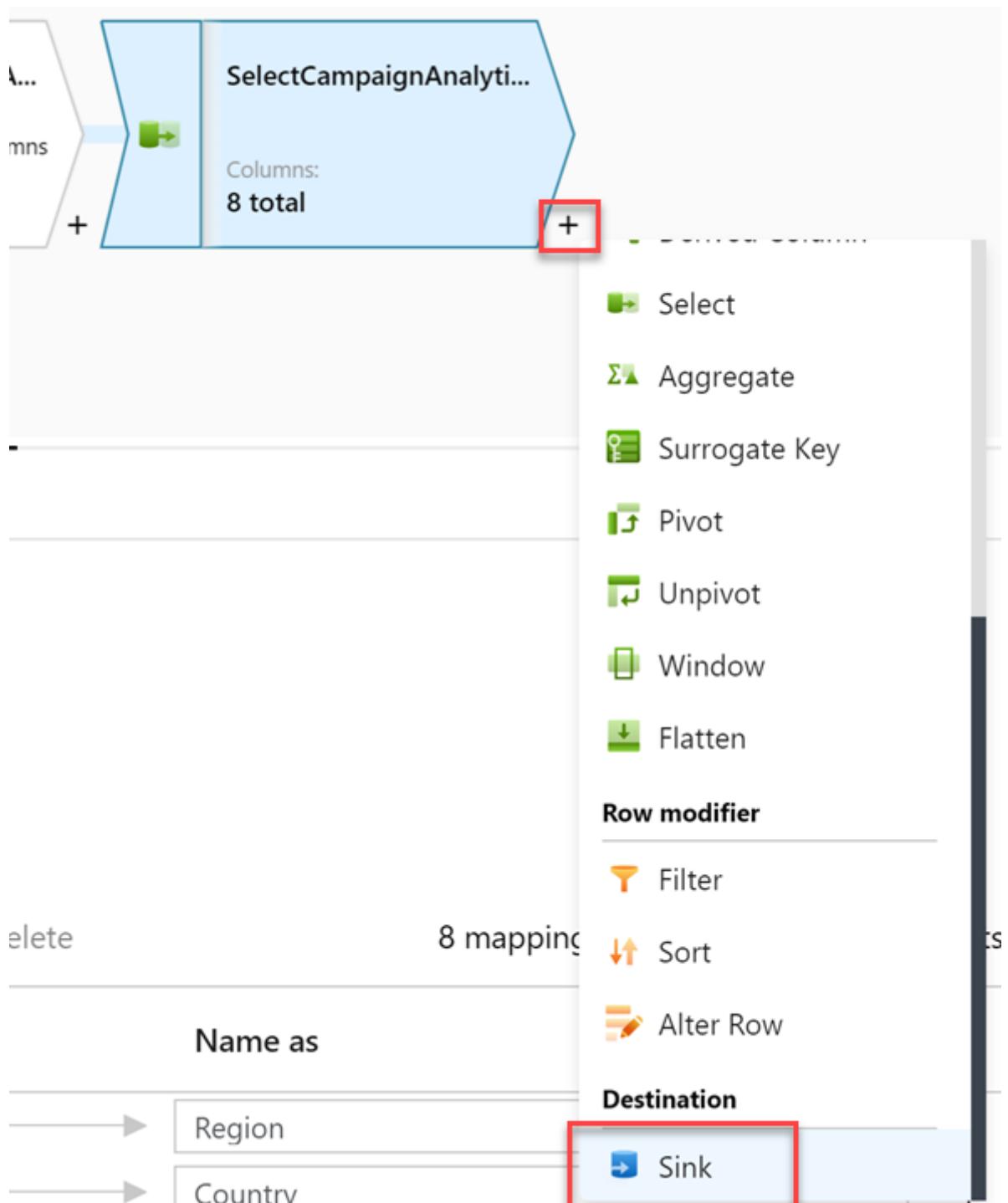
Options Skip duplicate input columns [○](#)
 Skip duplicate output columns [○](#)

Input columns * [Auto mapping](#) [Reset](#) [+ Add mapping](#) [Delete](#)

8 mappings: 2 column(s) from the inputs left unmapped

ConvertColumnTypeAndValues's column	Name as
abc Region	Region
abc Country	Country
abc ProductCategory	ProductCategory
abc CampaignName	CampaignName
e ^x Revenue	Revenue
e ^x RevenueTarget	RevenueTarget
abc City	City
abc State	State

15. Select the + to the right of the **SelectCampaignAnalyticsColumns** step, then select the **Sink** destination from the context menu.



16. Under **Sink**, configure the following:

- Output stream name:** Enter CampaignAnalyticsASA.
- Incoming stream:** Select SelectCampaignAnalyticsColumns.
- Sink type:** Select Dataset.
- Dataset:** Select asal400_wwi_campaign_analytics_asa, which is the CampaignAnalytics SQL table.
- Options:** Check Allow schema drift and uncheck Validate schema.

Sink Settings Mapping Optimize Inspect Data preview

Output stream name * CampaignAnalyticsASA [Learn more](#)

Incoming stream * SelectCampaignAnalyticsColumns

Sink type *

Dataset	Inline	Cache

Dataset * asal400_wwi_campaign_analytics_asa [Test connection](#) [Open](#) [New](#)

Options

- Allow schema drift [?](#)
- Validate schema [?](#)

17. Select **Settings**, then configure the following:

- **Update method:** Check `Allow insert` and leave the rest unchecked.
- **Table action:** Select `Truncate table`.
- **Enable staging:** Uncheck this option. The sample CSV file is small, making the staging option unnecessary.

Sink **Settings** Mapping Optimize Inspect Data preview

Info We recommend enabling staging to improve performance with Azure Synapse Analytics datasets.

Update method

- Allow insert
- Allow delete
- Allow upsert
- Allow update

Table action

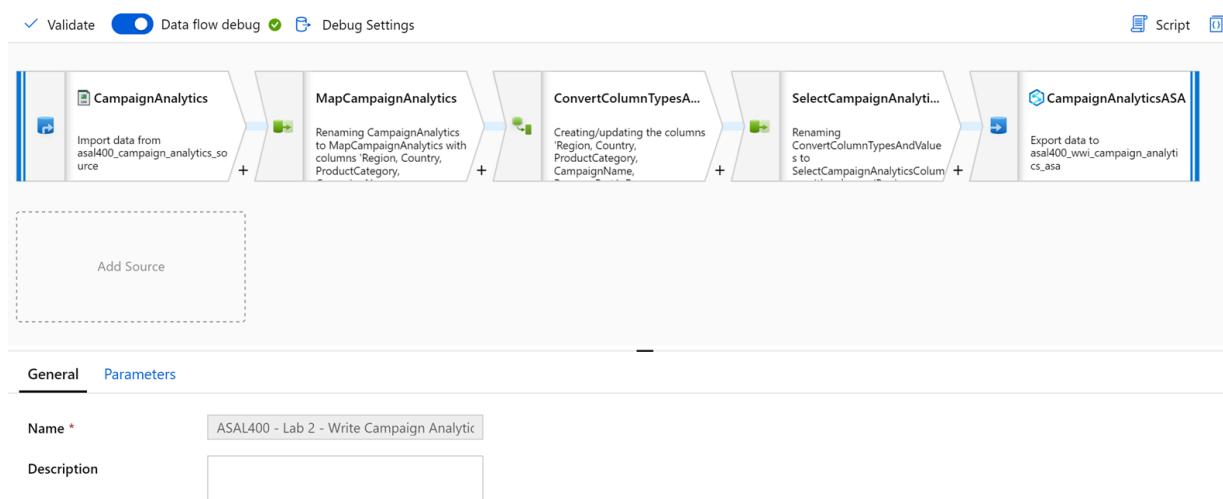
- None
- Recreate table
- Truncate table

Enable staging

Batch size ⓘ

Pre SQL scripts

18. Your completed data flow should look similar to the following:



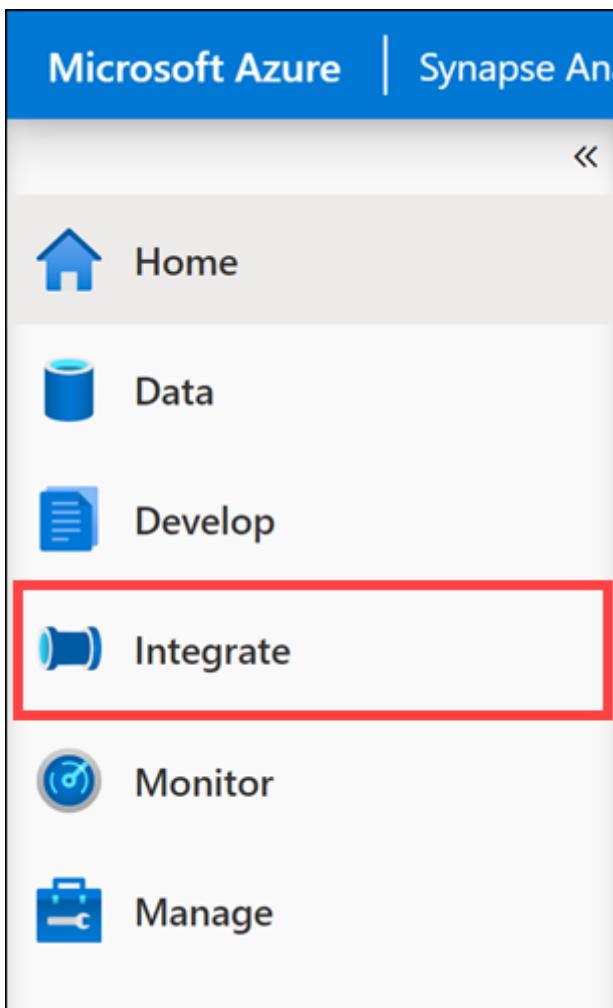
19. Select **Publish all** then **Publish** to save your new data flow.



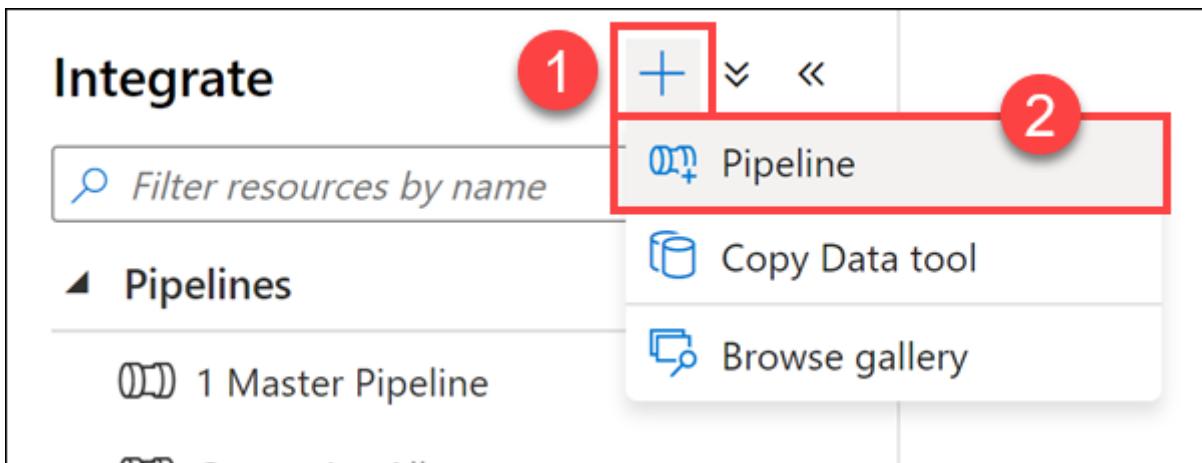
10.4.2.2 Task 2: Create campaign analytics data pipeline

In order to run the new data flow, you need to create a new pipeline and add a data flow activity to it.

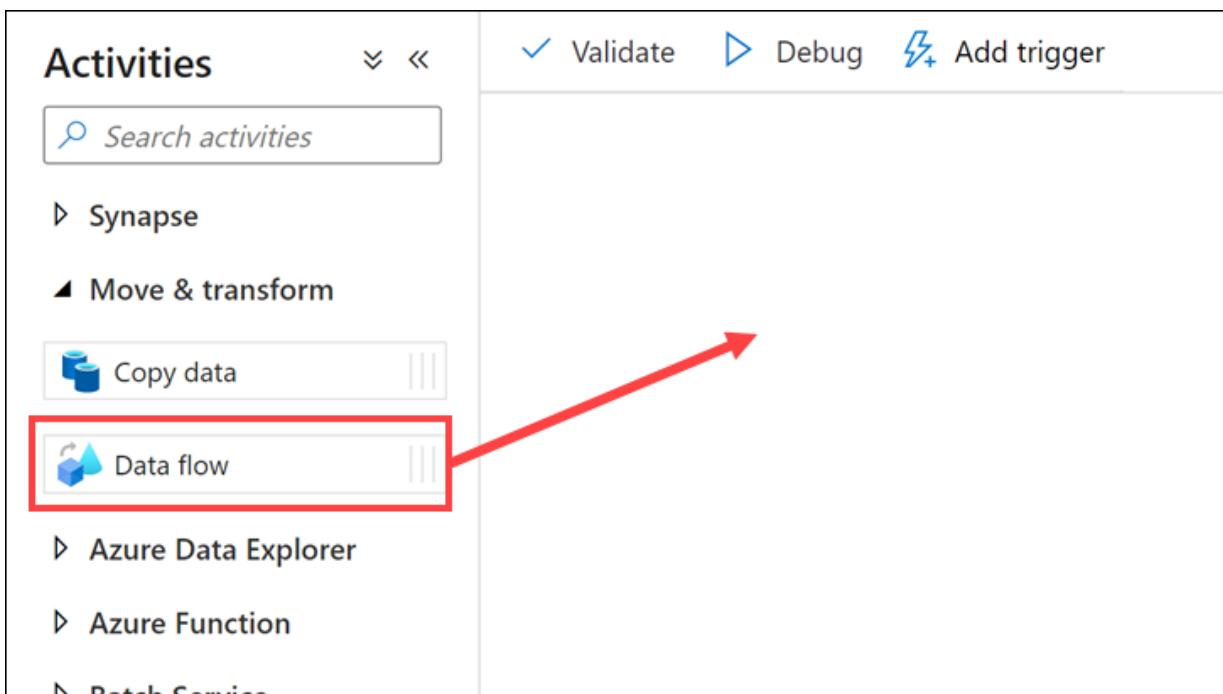
1. Navigate to the **Integrate** hub.



2. Select + then **Pipeline** to create a new pipeline.



3. In the **General** section of the **Properties** blade for the new pipeline, enter the following **Name**: Write Campaign Analytics to ASA.
4. Expand **Move & transform** within the Activities list, then drag the **Data flow** activity onto the pipeline canvas.



5. In the **General** section, set the **Name** value to asal400_lab2_writecampaignanalyticstoasa.

The screenshot shows the 'Data flow' blade in the Microsoft Azure portal. At the top, there's a blue header bar with the title 'Data flow' and a small circular icon with a red border. Below the header, there's a preview area showing a cube icon and the text 'asal400_lab2_writecam...'. Underneath this are four icons: a trash can, a copy symbol, a file, and a plus sign with a right arrow. Below the preview is a navigation bar with tabs: 'General' (underlined), 'Settings' (with a red box around it), 'Parameters' (with a red box around it), and 'User properties'. The main content area starts with a 'Name *' field containing 'asal400_lab2_writecampaignanalyticstoasa', which is also highlighted with a red box. To the right of this field is a 'learn more' link with a blue icon. Below the name field is a 'Description' field with an empty text area.

6. Select the **Settings** tab, then select **asal400_lab2_writecampaignanalyticstoasa** under **Data flow**.

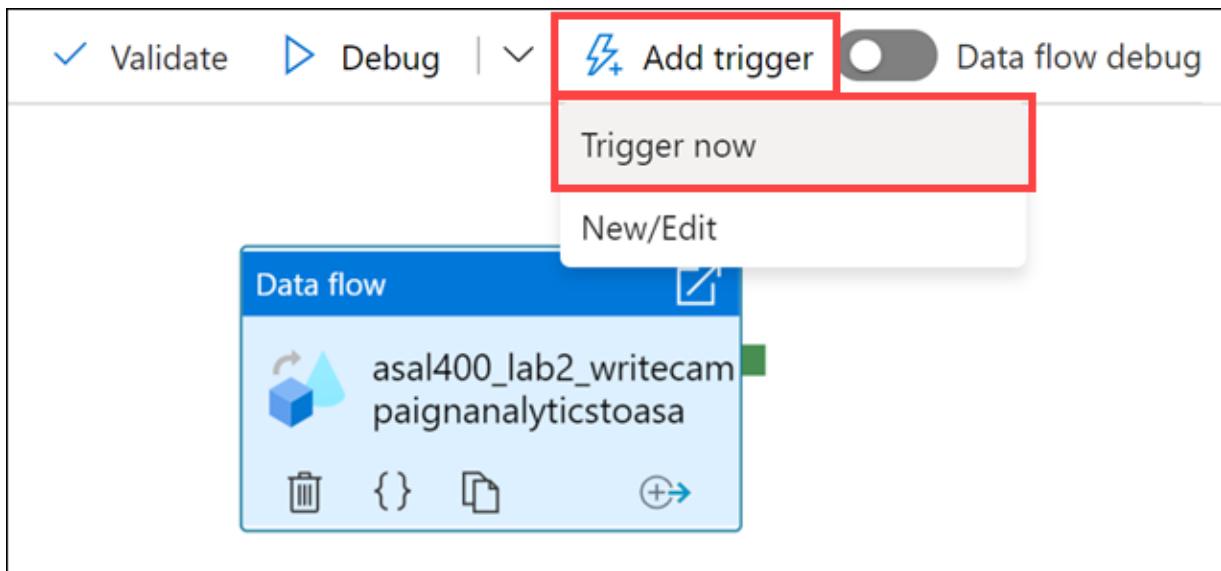
The screenshot shows the 'Settings' tab of the Data flow blade. The 'General' tab is selected and has a red box around its 'Settings' tab. The 'Data flow *' dropdown is set to 'asal400_lab2_writecampaignanalyticstoasa', which is also highlighted with a red box. To the right of the dropdown are two buttons: 'Open' with a blue pencil icon and 'New' with a blue plus sign icon. Below the dropdown are several configuration fields: 'Run on (Azure IR) *' (set to 'AutoResolveIntegrationRuntime'), 'Compute type *' (set to 'General purpose'), 'Core count *' (set to '4 (+ 4 Driver cores)'), and 'Logging level *' (radio buttons for 'Verbose' (selected), 'Basic', and 'None'). At the bottom left, there are two expandable sections: 'Sink properties' and 'Staging'.

7. Select **Publish all** to save your new pipeline.

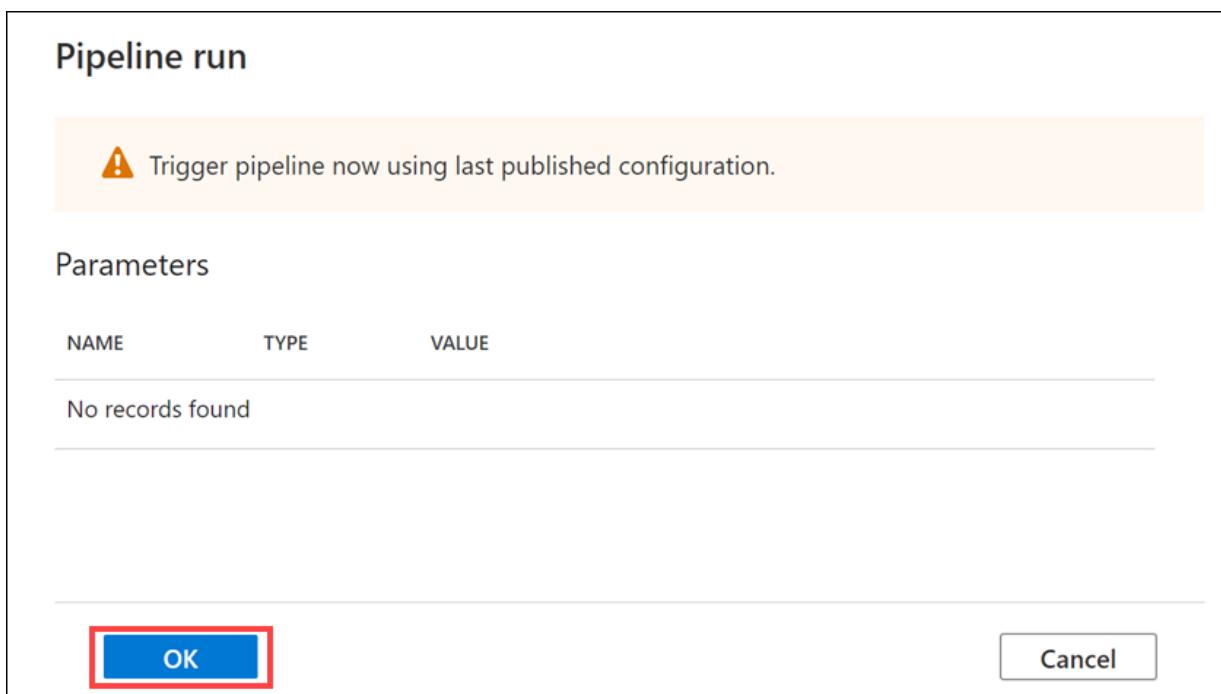
The screenshot shows the Microsoft Azure Synapse Analytics toolbar. It features a blue header with the text 'Microsoft Azure | Synapse Analytics'. Below the header are two buttons: 'Publish all' (highlighted with a red box) and 'Validate all'. The 'Publish all' button has a yellow circle with the number '1' on it, indicating one pending publication. There are also other toolbar icons like a double arrow and a triangle.

10.4.2.3 Task 3: Run the campaign analytics data pipeline

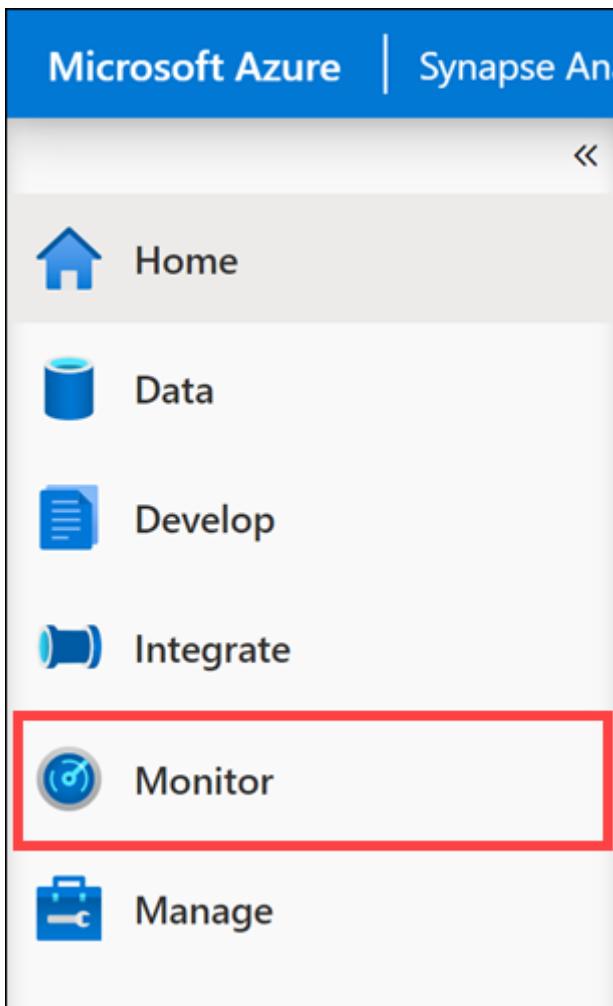
1. Select **Add trigger**, and then select **Trigger now** in the toolbar at the top of the pipeline canvas.



2. In the Pipeline run blade, select **OK** to start the pipeline run.



3. Navigate to the **Monitor** hub.



4. Wait for the pipeline run to successfully complete. You may need to refresh the view.

While this is running, read the rest of the lab instructions to familiarize yourself with the content.

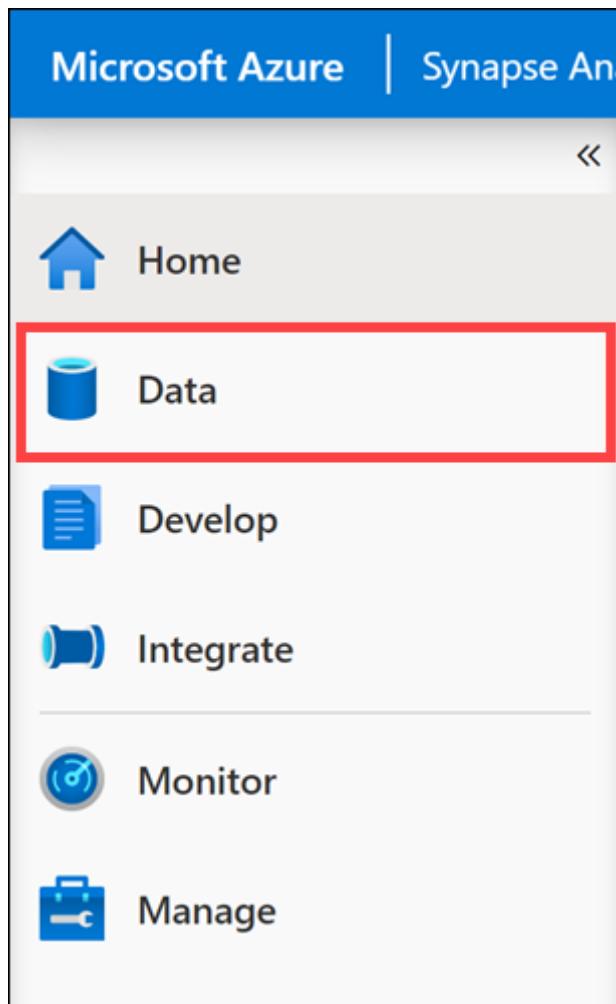
The screenshot shows the 'Pipeline runs' page in the Microsoft Azure Synapse Analytics portal. The left sidebar under 'Integration' has 'Pipeline runs' selected and highlighted with a red box. The main area displays a table of pipeline runs. One run is selected and highlighted with a red box, showing details: Pipeline name: Write Campaign Analytics to ..., Run start: 12/2/20, 5:18:10 PM, Run end: 12/2/20, 5:23:48 PM, Duration: 00:05:37, Triggered by: Manual trigger, Status: Succeeded (indicated by a green checkmark), and Run type: Original.

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
Write Campaign Analytics to ...	12/2/20, 5:18:10 PM	12/2/20, 5:23:48 PM	00:05:37	Manual trigger	Succeeded	Original

10.4.2.4 Task 4: View campaign analytics table contents

Now that the pipeline run is complete, let's take a look at the SQL table to verify the data successfully copied.

1. Navigate to the **Data** hub.



2. Expand the `SqlPool101` database underneath the **Workspace** section, then expand **Tables**.
3. Right-click the `wwi.CampaignAnalytics` table, then select the **Select TOP 1000 rows** menu item under the New SQL script context menu. You may need to refresh to see the new tables.

The screenshot shows the 'Data' workspace in Azure Data Studio. The left sidebar lists databases and tables. A context menu is open over the 'wwi.CampaignAnalytics' table, with several options highlighted by red boxes: 'New SQL script', 'Select TOP 100 rows', 'New notebook', 'New data flow', 'New integration dataset', and 'Refresh'.

4. The properly transformed data should appear in the query results.

The screenshot shows the Azure Data Studio query editor with the following SQL query:

```

1 SELECT TOP (100) [Region]
2 ,[Country]
3 ,[ProductCategory]
4 ,[CampaignName]
5 ,[Revenue]
6 ,[RevenueTarget]
7 ,[City]
8 ,[State]
9 | FROM [wwi].[CampaignAnalytics]

```

The results pane displays the following data:

Region	Country	ProductCategory	CampaignName	Revenue	RevenueTarget	City
South America	Mexico	Apparel and Footwear	Enjoy the Moment	1398.00	5663.00	NU
Europe	Germany	Décor	Tailored for You	7273.00	6184.00	NU
Asia Pacific	China	Apparel and Footwear	Tailored for You	5434.00	10709.00	NU
North & Central America	Canada	Exercise	Be Unique	8465.00	7654.00	NU
South East	US	Team Sports	Enjoy the Moment	16523.00	17741.00	NU
Far West	US	Books	EnjoyTheMoment; BeUn...	6815.00	14606.00	NU
South America	Brazil	Furniture	Enjoy the Moment	11245.00	1841.00	NU
Europe	France	Pillows & Cushions	Enjoy the Moment	15557.00	14176.00	NU
Asia Pacific	Japan	Lighting	Tailored for You	15642.00	10214.00	NU

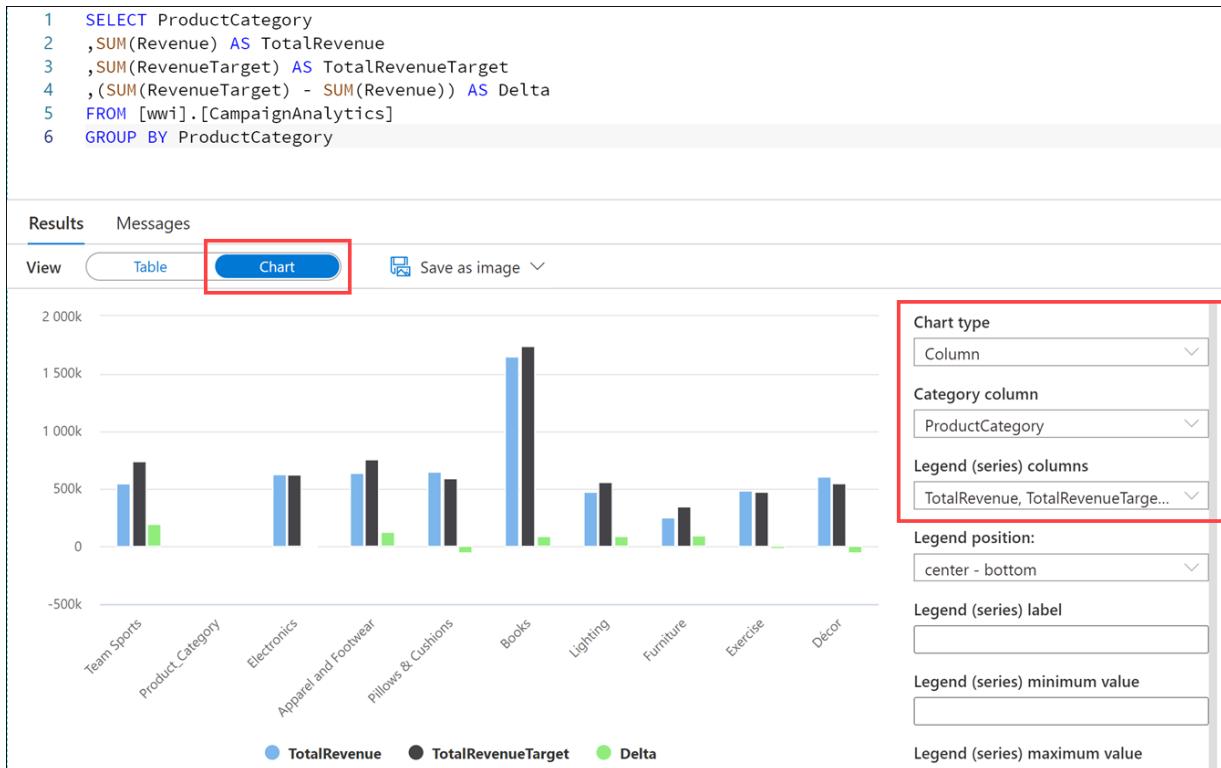
At the bottom, a message indicates: "00:00:00 Query executed successfully."

5. Update the query to the following and **Run**:

```
SELECT ProductCategory
, SUM(Revenue) AS TotalRevenue
, SUM(RevenueTarget) AS TotalRevenueTarget
, (SUM(RevenueTarget) - SUM(Revenue)) AS Delta
FROM [wwi].[CampaignAnalytics]
GROUP BY ProductCategory
```

6. In the query results, select the **Chart** view. Configure the columns as defined:

- **Chart type:** Select Column.
- **Category column:** Select ProductCategory.
- **Legend (series) columns:** Select TotalRevenue, TotalRevenueTarget, and Delta.



10.4.3 Exercise 3: Create Mapping Data Flow for top product purchases

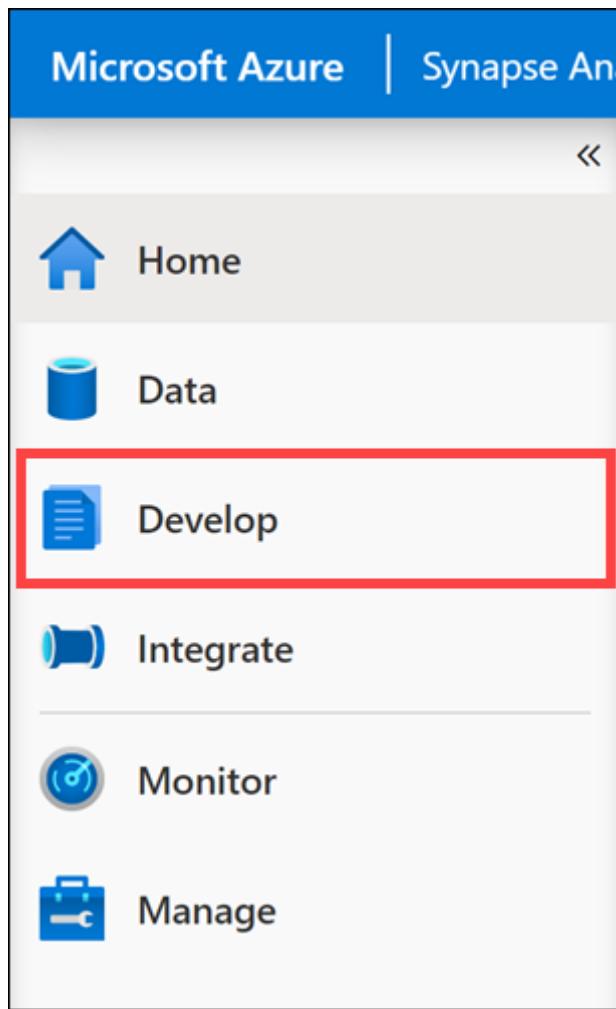
Tailwind Traders needs to combine top product purchases imported as JSON files from their eCommerce system with user preferred products from profile data stored as JSON documents in Azure Cosmos DB. They want to store the combined data in a dedicated SQL pool as well as their data lake for further analysis and reporting.

To do this, you will build a mapping data flow that performs the following tasks:

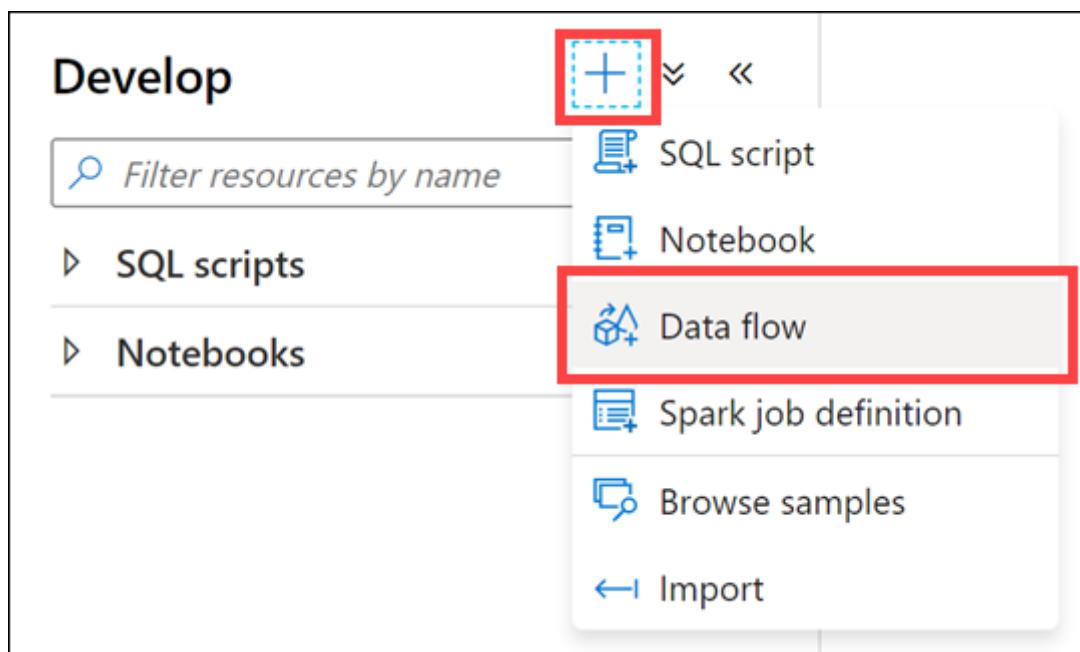
- Adds two ADLS Gen2 data sources for the JSON data
- Flattens the hierarchical structure of both sets of files
- Performs data transformations and type conversions
- Joins both data sources
- Creates new fields on the joined data set based on conditional logic
- Filters null records for required fields
- Writes to the dedicated SQL pool
- Simultaneously writes to the data lake

10.4.3.1 Task 1: Create Mapping Data Flow

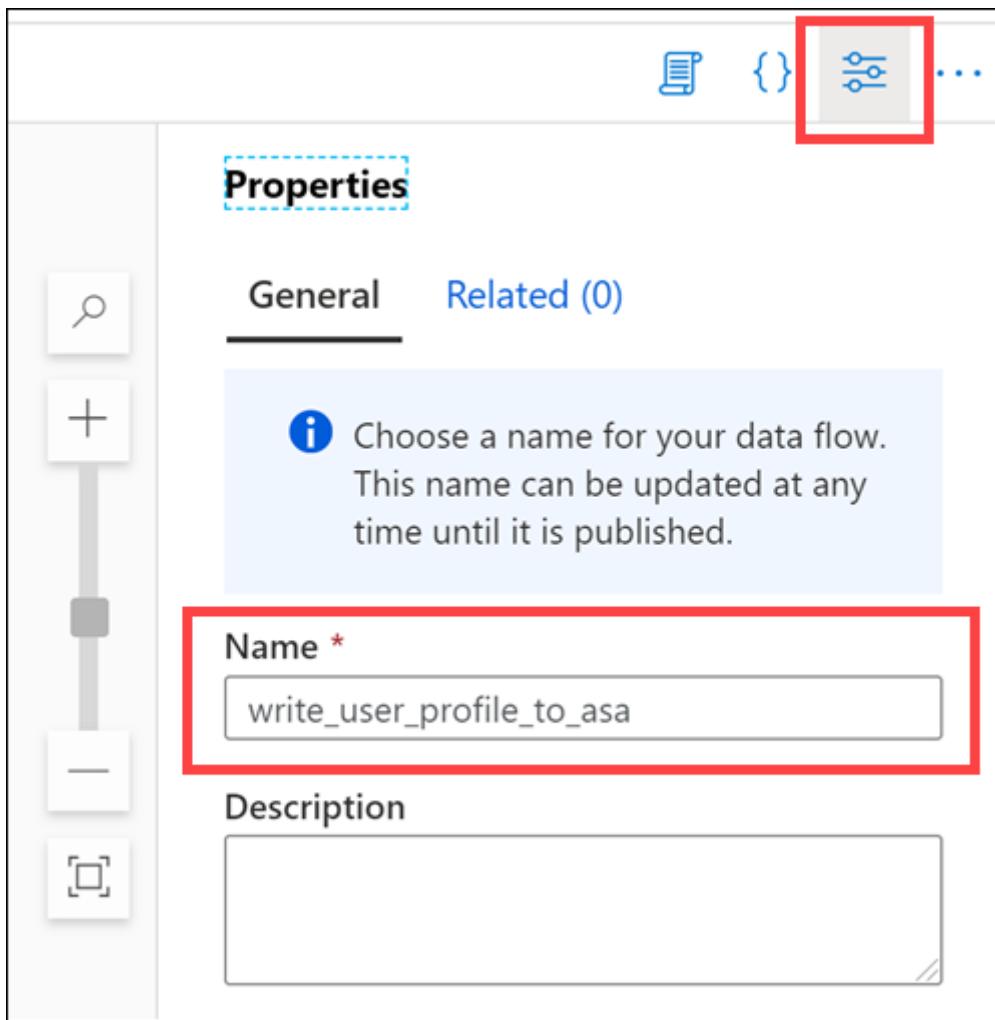
1. Navigate to the **Develop** hub.



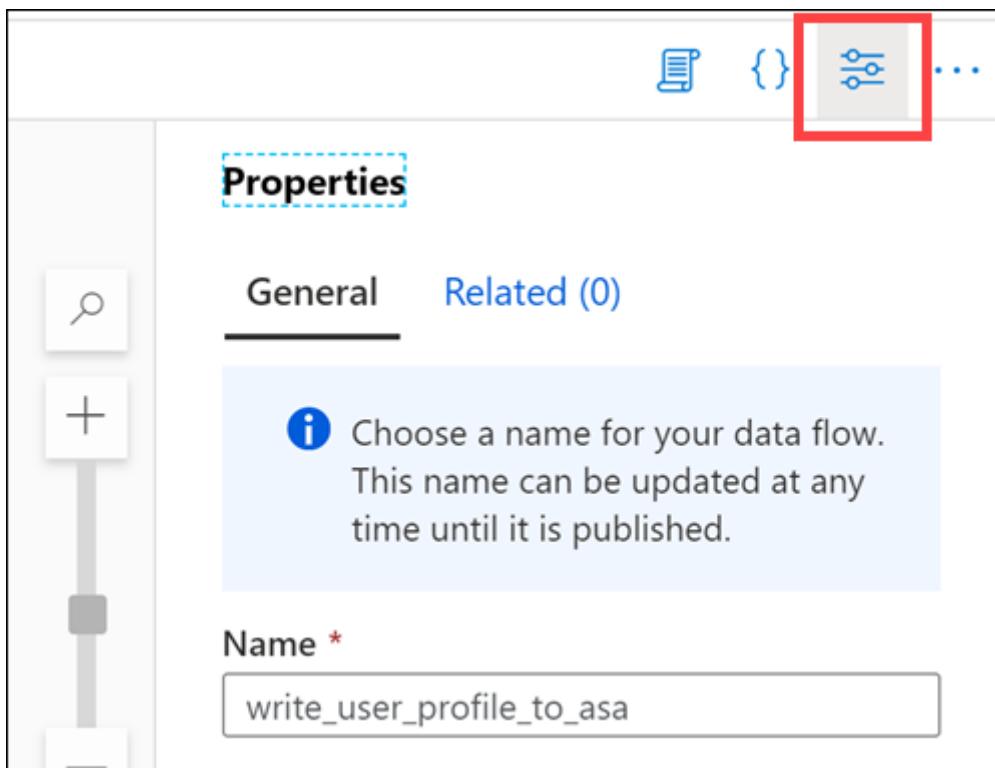
2. Select + then **Data flow** to create a new data flow.



3. In the **General** section of the **Profiles** pane of the new data flow, update the **Name** to the following:
`write_user_profile_to_asa`.

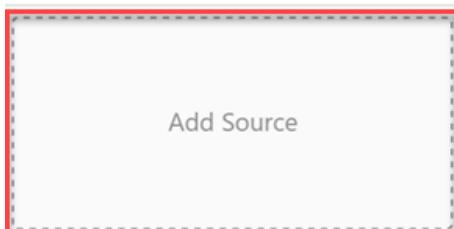


4. Select the **Properties** button to hide the pane.



5. Select **Add Source** on the data flow canvas.

Validate Data flow debug



6. Under **Source settings**, configure the following:

- **Output stream name:** Enter EcommerceUserProfiles.
- **Source type:** Select Dataset.
- **Dataset:** Select asal400_ecommerce_userprofiles_source.

Source settings	Source options	Projection	Optimize	Inspect	Data preview
Output stream name * EcommerceUserProfiles Learn more					
Source type * <input checked="" type="button"/> Dataset <input type="button"/> Inline					
Dataset * asal400_ecommerce_userprofiles_so.. Test connection					
Options <input checked="" type="checkbox"/> Allow schema drift i <input type="checkbox"/> Infer drifted column types i <input type="checkbox"/> Validate schema i					
Sampling * i <input type="radio"/> Enable <input checked="" type="radio"/> Disable					

7. Select the **Source options** tab, then configure the following:

- **Wildcard paths:** Enter online-user-profiles-02/*.json.
- **JSON Settings:** Expand this section, then select the **Array of documents** setting. This denotes that each file contains an array of JSON documents.

Source settings **Source options** Projection Optimize Inspect Data preview ●

Wildcard paths [+] [Delete]

Partition root path

Allow no files found

List of files

Column to store file name

After completion * No action Delete source files Move

Start time (UTC) End time (UTC)

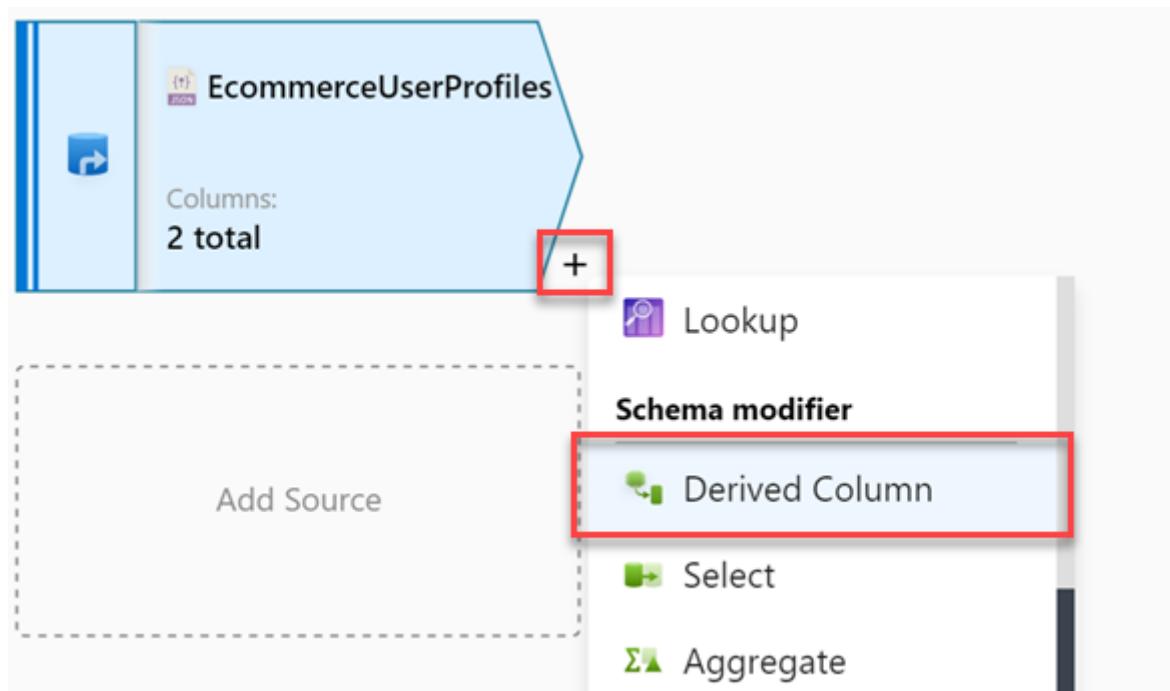
Filter by last modified

▲ JSON settings

Document form Single document Document per line Array of documents [i]

Unquoted column names

8. Select the + to the right of the **EcommerceUserProfiles** source, then select the **Derived Column** schema modifier from the context menu.



9. Under **Derived column's settings**, configure the following:

- **Output stream name:** Enter `userId`.
- **Incoming stream:** Select **EcommerceUserProfiles**.
- **Columns:** Provide the following information:

Column	Expression	Description
visitorId	<code>toInteger(visitorId)</code>	Converts the <code>visitorId</code> column from a string to an integer.

Derived column's settings Optimize Inspect Data preview Description

Output stream name * Learn more [\[\]](#)

Incoming stream * [\[\]](#)

[\[\] Add](#) [\[\] Clone](#) [\[\] Delete](#) [\[\] Open expression builder](#)

Columns [\[\]](#)

Column	Expression
visitorId	tolInteger(visitorId)

10. Select the **+** to the right of the **userId** step, then select the **Flatten** formatter from the context menu.

..

+ **userId**

Columns: 2 total

+ **Flatten**

Pivot

Unpivot

Window

Rank

Formatters

Parse

Optimize Inspect Data preview

11. Under **Flatten settings**, configure the following:

- Output stream name:** Enter **UserTopProducts**.
- Incoming stream:** Select **userId**.
- Unroll by:** Select **[] topProductPurchases**.
- Input columns:** Provide the following information:

userId's column	Name as
visitorId	visitorId
topProductPurchases.productId	productId
topProductPurchases.itemsPurchasedLast12Months	itemsPurchasedLast12Months

Select **+** **Add mapping**, then select **Fixed mapping** to add each new column mapping.

Output stream name * UserTopProducts

Incoming stream * userId

Unroll by * topProductPurchases

Unroll root

Options Skip duplicate input columns Skip duplicate output columns

Input columns * userId's column visitorId abc topProductPurchases.productId abc topProductPurchases.itemsPurchasedLast... + Add mapping Delete

3 mappings: All inputs mapped

Source Column	Mapped To
visitorId	visitorId
productId	productId
itemsPurchasedLast12Months	itemsPurchasedLast12Months

These settings provide a flattened view of the data source with one or more rows per visitorId, similar to when we explored the data within the Spark notebook in the previous module. Using data preview requires you to enable Debug mode, which we are not enabling for this lab. *The following screenshot is for illustration only:*

Flatten settings Optimize Inspect Data preview

Number of rows + INSERT 100 * UPDATE 0 × DELETE 0 ⚡ UPsert 0 🔎 LOOKUP 0

⟳ Refresh Typecast ↻ Modify ↻ Map drifted Statistics Remove

visitorId	productId	itemsPurchasedLast12Months
9611082	61	62
9611082	3003	20
9611082	2140	39
9611082	2918	62
9611082	4482	85
9611082	140	73
9611082	3892	33

IMPORTANT: A bug was introduced with the latest release, and the userId source columns are not being updated from the user interface. As a temporary fix, access the script for the data flow (located in the toolbar). Find the userId activity in the script, and in the mapColumn function, ensure you append the appropriate source field. For productId, ensure it is sourced from **topProductPurchases.productId**, and that itemsPurchasedLast12Months is sourced from **topProductPurchases.itemsPurchasedLast12Months**.



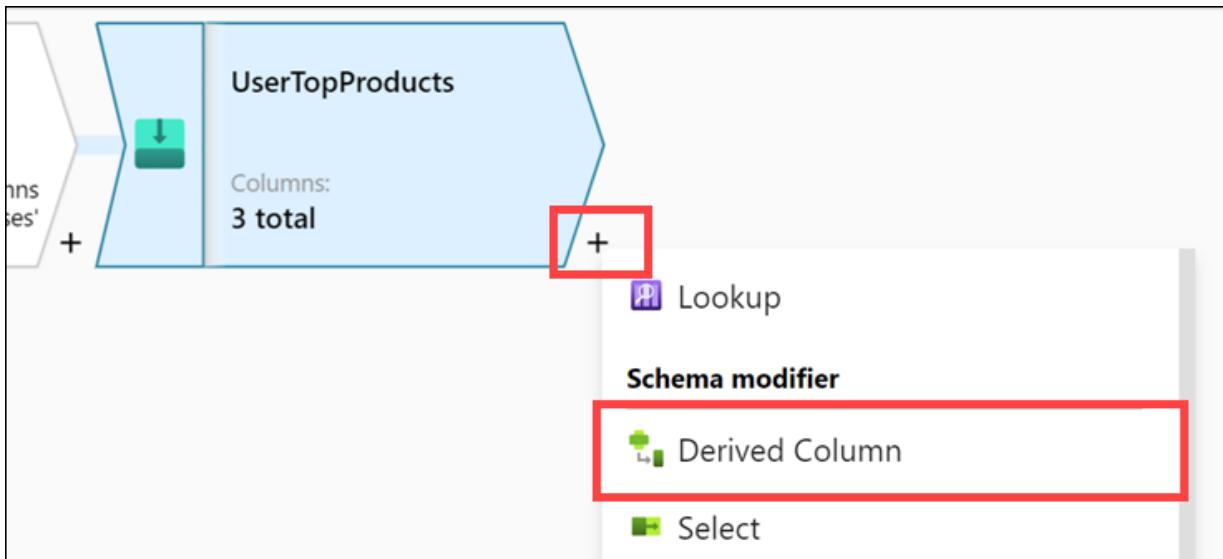
```
userId foldDown(unroll(topProductPurchases),
    mapColumn(
        visitorId,
        productId = topProductPurchases.productId,
        itemsPurchasedLast12Months = topProductPurchases.itemsPurchasedLast12Months
    )
)
```

```

source(output(
    visitorId as string,
    topProductPurchases as (productId as string, itemsPurchasedLast12Months as string)[])
),
allowSchemaDrift: true,
validateSchema: false,
ignoreNoFilesFound: false,
documentForm: 'singleDocument',
wildcardPaths:['online-user-profiles-02/*.json']) ~> EcommerceUserProfiles
source(output(
    userId as string,
    cartId as string,
    preferredProducts as string[],
    productReviews as string[]
),
allowSchemaDrift: true,
validateSchema: false,
ignoreNoFilesFound: false,
format: 'document') ~> UserProfiles
EcommerceUserProfiles derive(visitorId = toInteger(visitorId)) ~> userId
userId foldDown(unroll(topProductPurchases),
mapColumn(
    visitorId,
    productId = topProductPurchases.productId,
    itemsPurchasedLast12Months = topProductPurchases.itemsPurchasedLast12Months
),
skipDuplicateMapInputs: false,
skipDuplicateMapOutputs: false) ~> UserTopProducts

```

12. Select the + to the right of the UserTopProducts step, then select the **Derived Column** schema modifier from the context menu.



13. Under **Derived column's settings**, configure the following:

- Output stream name:** Enter DeriveProductColumns.
- Incoming stream:** Select UserTopProducts.
- Columns:** Provide the following information:

Column	Expression	Description
productId	toInteger(productId)	Converts the productId column from a string
itemsPurchasedLast12Months	toInteger(itemsPurchasedLast12Months)	Converts the itemsPurchasedLast12Months column from a string

Derived column's settings Optimize Inspect Data preview ●

Output stream name * Learn more [🔗](#)

Incoming stream * [▼](#)

[+ Add](#) [Clone](#) [Delete](#) [Open expression builder](#)

Columns * ⓘ

Column	Expression
<input type="checkbox"/> productId	<input type="text" value="toInteger(productId)"/> 123 + ↗ Delete ↗
<input type="checkbox"/> itemsPurchasedLast12Months	<input type="text" value="toInteger(itemsPurchasedLast12Months)"/> 123 + ↗ Delete ↗

Note: To add a column to the derived column settings, select **+** to the right of the first column, then select **Add column**.

Output stream name * Learn more [🔗](#)

Incoming stream * [▼](#)

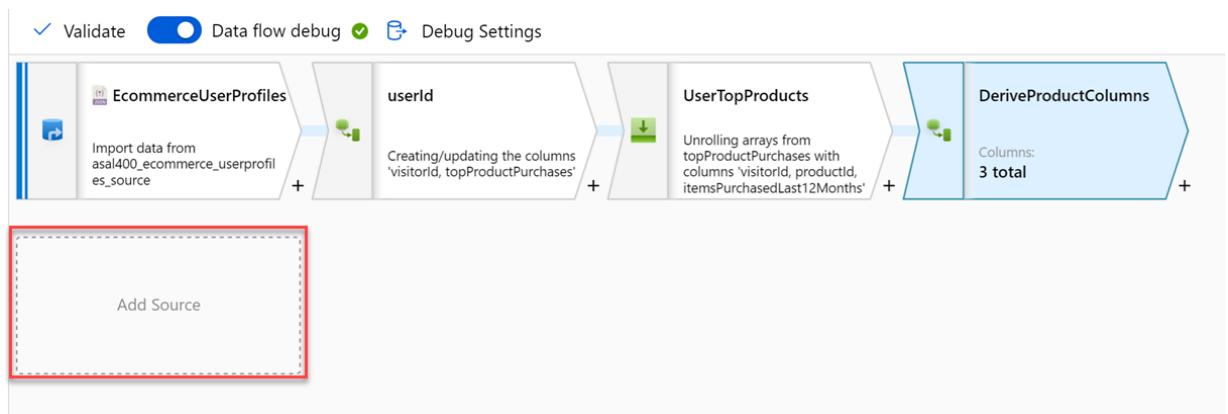
[+ Add](#) [Clone](#) [Delete](#) [Open expression builder](#)

Columns * ⓘ

Column	Expression
<input type="checkbox"/> productId	<input type="text" value="toInteger(productId)"/> 123 + ↗ Delete ↗

[Add column](#) [Add column pattern](#)

14. Select **Add Source** on the data flow canvas beneath the **EcommerceUserProfiles** source.



15. Under **Source settings**, configure the following:

- **Output stream name:** Enter **UserProfiles**.
- **Source type:** Select **Dataset**.
- **Dataset:** Select **asal400_customerprofile_cosmosdb**.

Source settings Source options Projection Optimize Inspect Data preview

Output stream name * [Learn more](#)

Source type * Dataset Inline

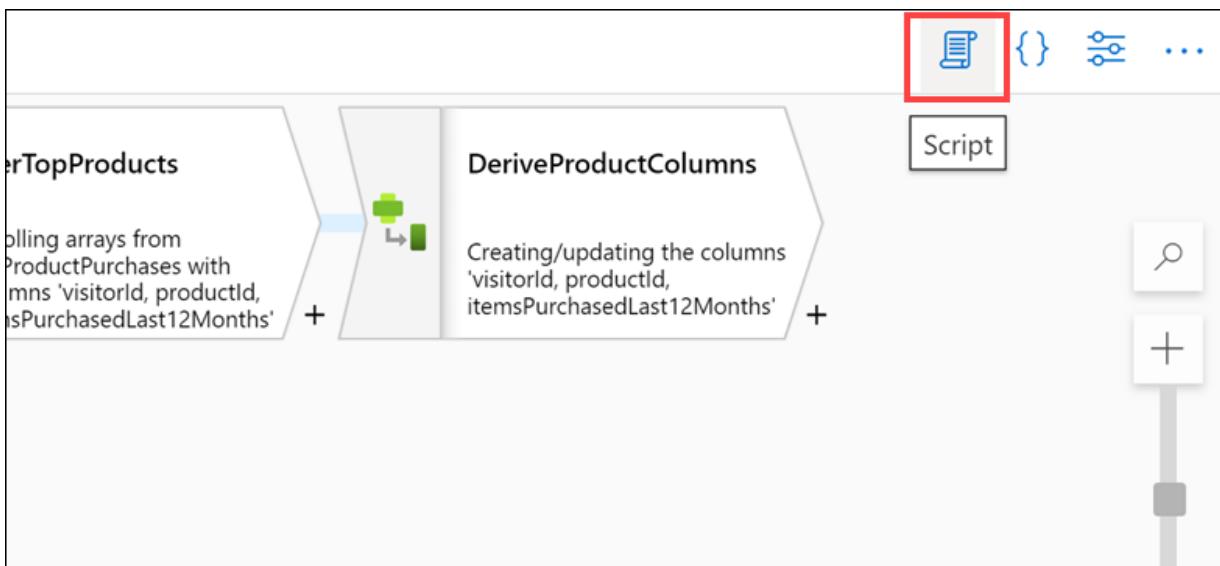
Dataset * [Test connection](#)

Options

- Allow schema drift [ⓘ](#)
- Infer drifted column types [ⓘ](#)
- Validate schema [ⓘ](#)

Sampling * ⓘ Enable Disable

16. Since we are not using the data flow debugger, we need to enter the data flow's Script view to update the source projection. Select **Script** in the toolbar above the canvas.



17. Locate the **UserProfiles** source in the script and replace its script block with the following to set `preferredProducts` as an `integer[]` array and ensure the data types within the `productReviews` array are correctly defined:

```
source(output(
    cartId as string,
    preferredProducts as integer[],
    productReviews as (productId as integer, reviewDate as string, reviewText as string) [],
    userId as integer
),
allowSchemaDrift: true,
validateSchema: false,
ignoreNoFilesFound: false,
format: 'document') ~> UserProfiles
```

```

1 source(output(
2     visitorId as string,
3     topProductPurchases as (productId as string, itemsPurchasedLast12Months as string)[]
4 ),
5     allowSchemaDrift: true,
6     validateSchema: false,
7     ignoreNoFilesFound: false,
8     documentForm: 'arrayOfDocuments',
9     wildcardPaths:['online-user-profiles-02/*.json']) ~> EcommerceUserProfiles
10 source(output(
11     cartId as string,
12     preferredProducts as integer[],
13     productReviews as (productId as integer, reviewDate as string, reviewText as string)[],
14     userId as integer
15 ),
16     allowSchemaDrift: true,
17     validateSchema: false,
18     ignoreNoFilesFound: false,
19     format: 'document') ~> UserProfiles
20 EcommerceUserProfiles derive(visitorId = toInteger(visitorId)) ~> userId
21 UserId foldDown(unroll(topProductPurchases),
22     mapColumn(
23         visitorId,
24         productId = topProductPurchases.productId,
25         itemsPurchasedLast12Months = topProductPurchases.itemsPurchasedLast12Months
26     ),
27     skipDuplicateMapInputs: false,
28     skipDuplicateMapOutputs: false) ~> UserTopProducts
29 UserTopProducts derive(productId = toInteger(productId),
30     itemsPurchasedLast12Months = toInteger(itemsPurchasedLast12Months)) ~> DeriveProductColumns

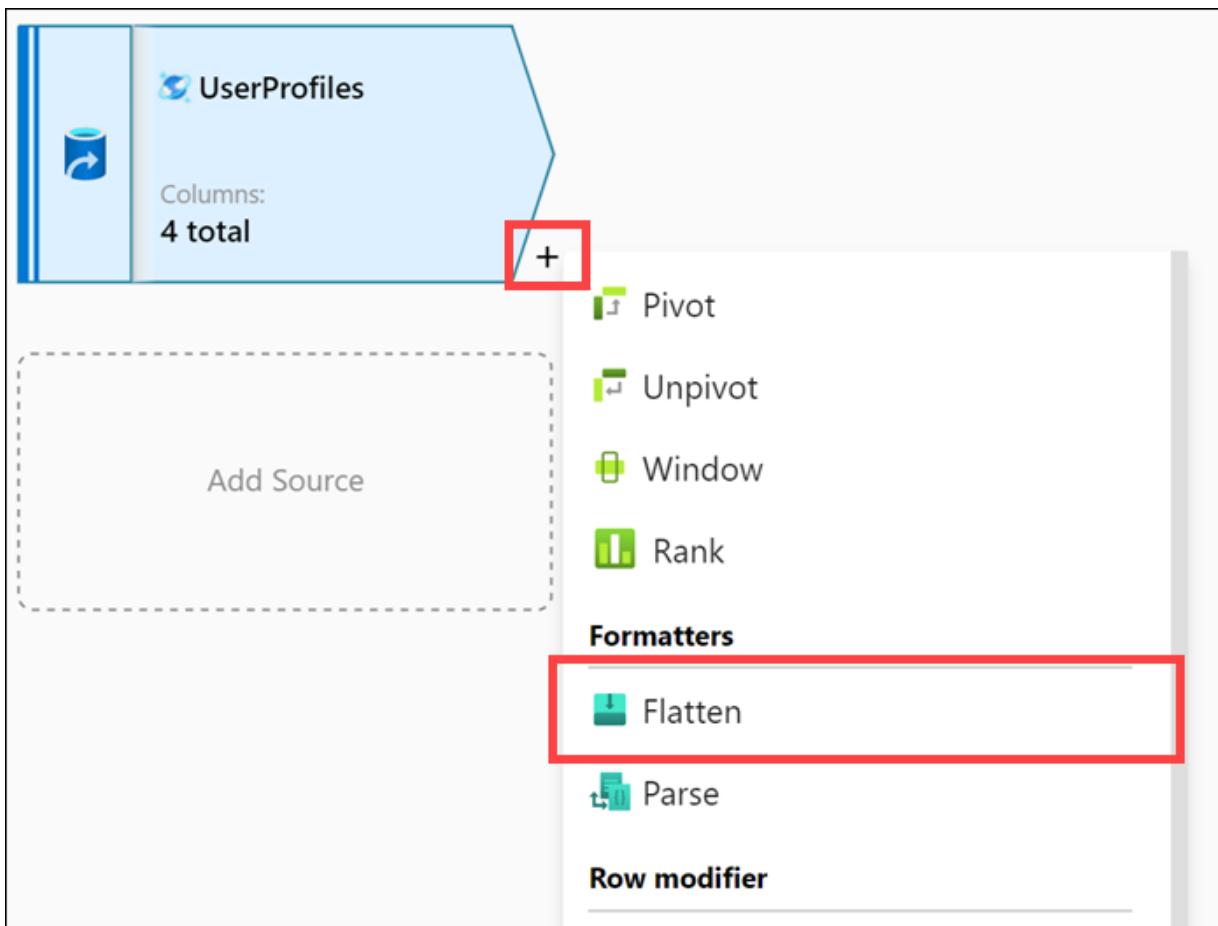
```

OK Copy as single line Cancel

18. Select **OK** to apply the script changes. The data source has now been updated with the new schema. The following screenshot shows what the source data looks like if you are able to view it with the data preview option. Using data preview requires you to enable Debug mode, which we are not enabling for this lab. *The following screenshot is for illustration only.*

Source settings						Source options	Projection	Optimize	Inspect	Data preview	
Number of rows	+ INSERT	100	*	UPDATE	0	×	DELETE	0	*	UPSERT	0
	↻ Refresh	□ Typecast	▼	Modify	▼	Map drifted	☒	Statistics	×	Remove	🔍 LOOKUP
↑↓	cartId	abc		preferredProducts	[]	123	preferredProducts[1]: 4444		userId	123	
+	406a06af-e54f-42e9-aad8-9a36f2c7f8ca	[...]				123	preferredProducts[2]: 330			9079954	
+	5c4dc5dc-a585-41ec-8149-9133caa3a73a	[...]								9079747	
+	d79f4b3a-6c66-497a-bf59-eca12dd4f16a	[...]								9079334	
+	acde5617-af9f-4ad9-98ed-e03c793e7bd0	[...]								9079190	
+	7f2742a6-4653-4905-80ad-2d64dae04253	[...]								9078634	
+	2d3c75f6-24ce-4307-9059-902736896c61	[...]								9078467	
+	44f4c3f9-d761-4d10-bbf8-84cd45c3a461	[...]								9078115	

19. Select the **+** to the right of the **UserProfiles** source, then select the **Flatten** formatter from the context menu.

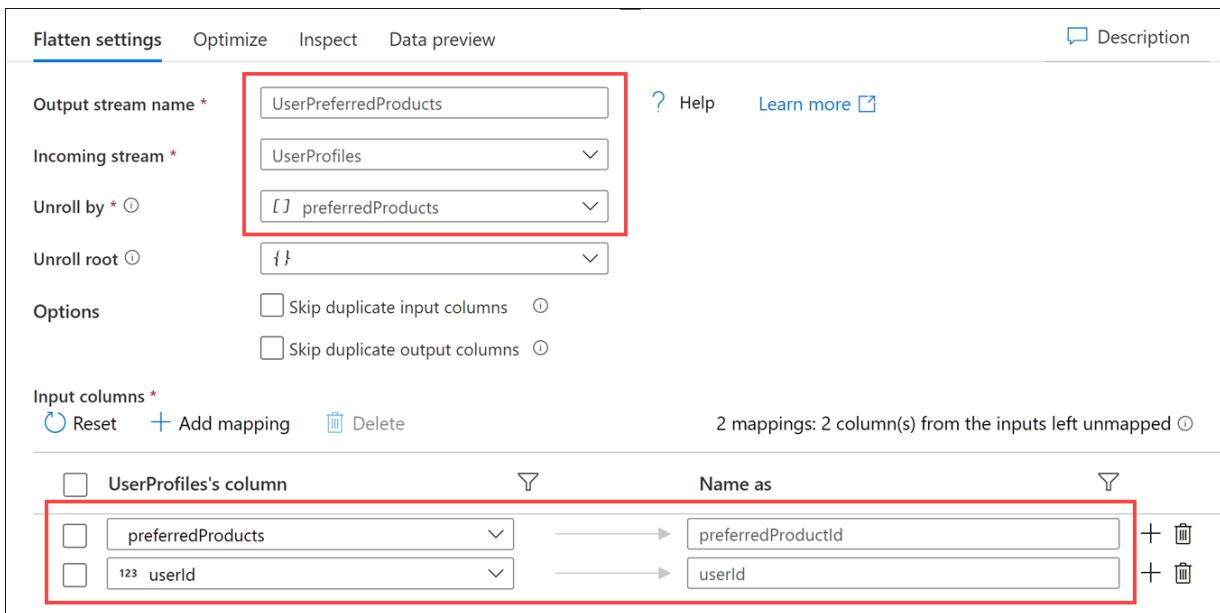


20. Under **Flatten settings**, configure the following:

- **Output stream name:** Enter `UserPreferredProducts`.
- **Incoming stream:** Select `UserProfiles`.
- **Unroll by:** Select `[] preferredProducts`.
- **Input columns:** Provide the following information. Be sure to **delete** `cartId` and `[] productReviews`:

UserProfiles's column	Name as
<code>[] preferredProducts</code>	<code>preferredProductId</code>
<code>userId</code>	<code>userId</code>

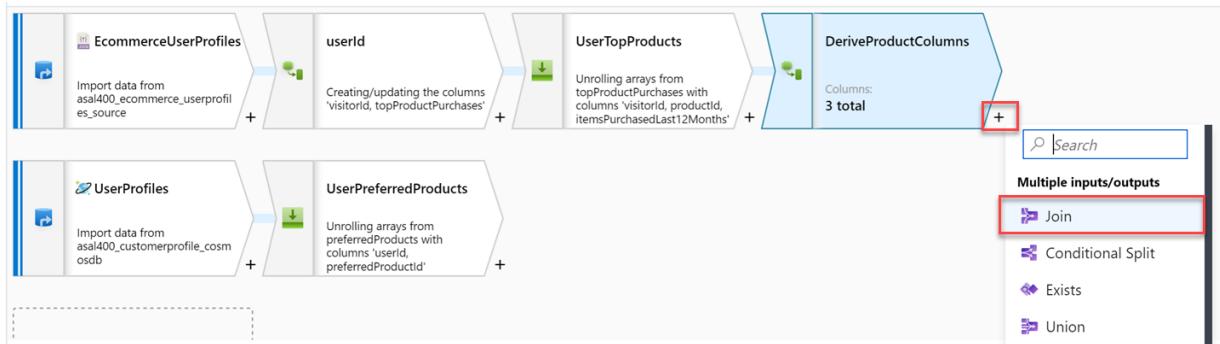
Select **+** **Add mapping**, then select **Fixed mapping** to add each new column mapping.



These settings provide a flattened view of the data source with one or more rows per `userId`. Using data preview requires you to enable Debug mode, which we are not enabling for this lab. *The following screenshot is for illustration only:*

Flatten settings	Optimize	Inspect	Data preview
Number of rows + INSERT 100	* UPDATE 0	✗ DELETE 0	* UPSERT 0
↻ Refresh	Typecast	Modify	Map drifted
↑↓	userId 123		preferredProductId 123
+ 9079747			4235
+ 9079747			3288
+ 9079747			2756
+ 9079334			4074
+ 9079334			272
+ 9079334			2164
+ 9079334			414

21. Now it is time to join the two data sources. Select the **+** to the right of the `DeriveProductColumns` step, then select the **Join** option from the context menu.



22. Under **Join settings**, configure the following:

- **Output stream name:** Enter `JoinTopProductsWithPreferredProducts`.
- **Left stream:** Select `DeriveProductColumns`.

- **Right stream:** Select UserPreferredProducts.
- **Join type:** Select Full outer.
- **Join conditions:** Provide the following information:

Left: DeriveProductColumns's column	Right: UserPreferredProducts's column
visitorId	userId

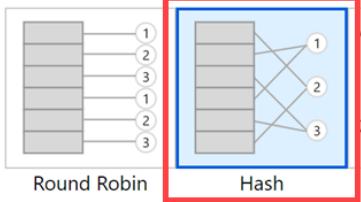
Join settings Optimize Inspect Data preview ● Description

Output stream name *	<input type="text" value="JoinTopProductsWithPreferredProducts"/>	Learn more ?
Left stream *	<input type="text" value="DeriveProductColumns"/>	
Right stream *	<input type="text" value="UserPreferredProducts"/>	
Join type *	<input checked="" type="radio"/> Full outer <input type="radio"/> Inner <input type="radio"/> Left outer <input type="radio"/> Right outer <input type="radio"/> Custom (cross)	
Join conditions *	Left: DeriveProductColumns's column <input type="text" value="123 visitorId"/> <input type="text" value="123 userId"/> = + -	

23. Select **Optimize** and configure the following:

- **Broadcast:** Select Fixed.
- **Broadcast options:** Check Left: 'DeriveProductColumns'.
- **Partition option:** Select Set partitioning.
- **Partition type:** Select Hash.
- **Number of partitions:** Enter 30.
- **Column:** Select productId.

Join settings **Optimize** Inspect Data preview ●

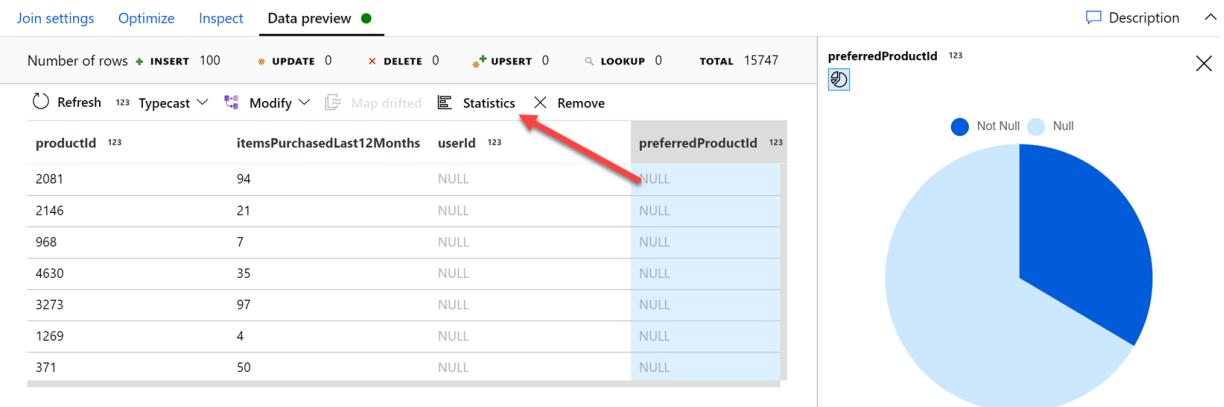
Broadcast	<input type="radio"/> Auto ? <input checked="" type="radio"/> Fixed ? <input type="radio"/> Off ?
Broadcast options *	<input checked="" type="checkbox"/> Left: 'DeriveProductColumns' ? <input type="checkbox"/> Right: 'UserPreferredProducts' ?
Partition option *	<input type="radio"/> Use current partitioning <input type="radio"/> Single partition <input checked="" type="radio"/> Set partitioning ?
Partition type *	 Hash
Number of partitions *	<input type="text" value="30"/>
Column values to hash on * Column	<input type="text" value="123 productId"/>

24. Select the **Inspect** tab to see the join mapping, including the column feed source and whether the column is used in a join.

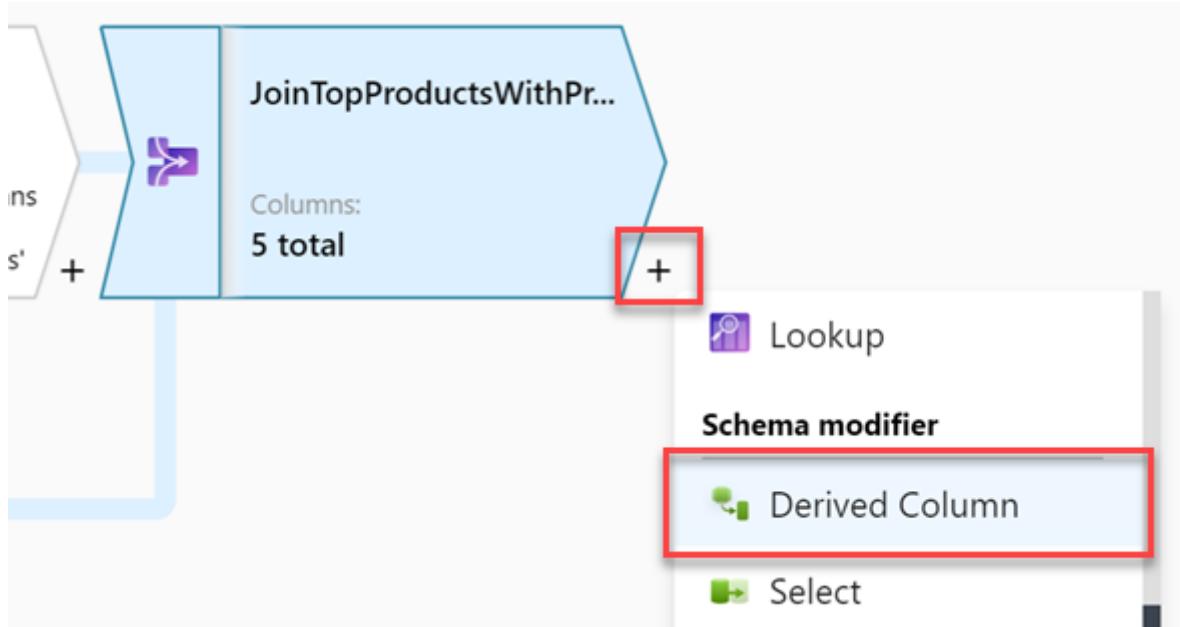
Join settings Optimize **Inspect** Data preview

Number of columns Left: DeriveProductColumns 3		Right: UserPreferredProducts 2		
Order ↑↓	Column ↑↓	Type ↑↓	Fed by ↑↓	Used in Join ↑↓
1	visitorId	123 integer	UserTopProducts	✓
2	productId	123 integer	DeriveProductColumns	
3	itemsPurchasedLast12Months	123 integer	DeriveProductColumns	
4	userId	123 integer	UserPreferredProducts	✓
5	preferredProductId	123 integer	UserPreferredProducts	

For illustrative purposes of data preview only: Since we are not turning on data flow debugging, do not perform this step. In this small sample of data, likely the `userId` and `preferredProductId` columns will only show null values. If you want to get a sense of how many records contain values for these fields, select a column, such as `preferredProductId`, then select **Statistics** in the toolbar above. This displays a chart for the column showing the ratio of values.



25. Select the + to the right of the `JoinTopProductsWithPreferredProducts` step, then select the **Derived Column** schema modifier from the context menu.



26. Under **Derived column's settings**, configure the following:

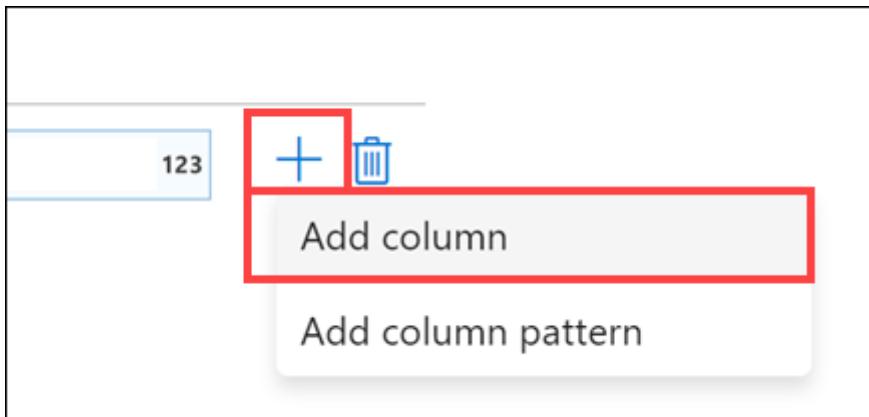
- Output stream name:** Enter `DerivedColumnsForMerge`.
- Incoming stream:** Select `JoinTopProductsWithPreferredProducts`.
- Columns:** Provide the following information (*type in the first two column names*):

Column	Expression	Description
isTopProduct	toBoolean(iif(isNull(productId), 'false', 'true'))	Returns true if productId is null.
isPreferredProduct	toBoolean(iif(isNull(preferredProductId), 'false', 'true'))	Returns true if preferredProductId is null.
productId	iif(isNull(productId), preferredProductId, productId)	Sets the productId output to preferredProductId if productId is null.
userId	iif(isNull(userId), visitorId, userId)	Sets the userId output to visitorId if userId is null.

The screenshot shows the 'Derived column's settings' tab in a data flow editor. It includes fields for 'Output stream name' (set to 'DerivedColumnsForMerge'), 'Incoming stream' (set to 'JoinTopProductsWithPreferredProdu...'), and buttons for 'Add', 'Clone', 'Delete', and 'Open expression builder'. Below these, a table lists four derived columns:

Column	Expression
isTopProduct	toBoolean(iif(isNull(productId), 'false', 'true'))
isPreferredProduct	toBoolean(iif(isNull(preferredProductId), 'false', 'true'))
productId	iif(isNull(productId), preferredProductId, productId)
userId	iif(isNull(userId), visitorId, userId)

Note: Remember, select +, then **Add column** to the right of a derived column to add a new column below.



The derived column settings provide the following result:

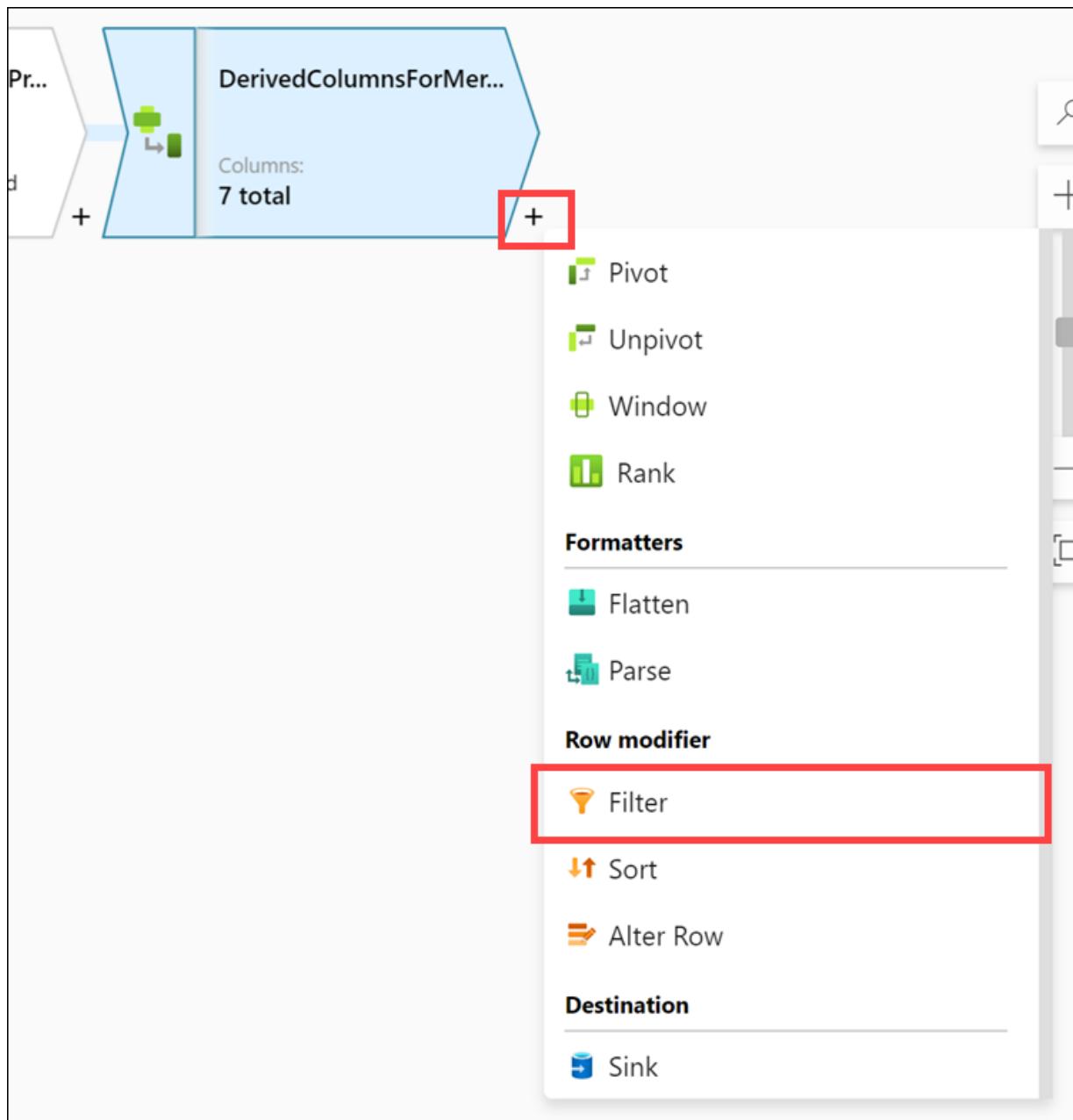
The screenshot shows the 'Data preview' tab with the following details:

- Number of rows: 15361
- Action buttons: INSERT 100, UPDATE 0, DELETE 0, UPSERT 0, LOOKUP 0, TOTAL 15361
- Toolbar buttons: Refresh, Typecast, Modify, Map drifted, Statistics, Remove

The data preview table has the following schema and data:

userId	preferredProductId	visitorId	productId	itemsPurchasedLast12Months	isTopProduct	isPreferredProduct
9591082	NULL	9591082	372	9	✓	✗
9591085	NULL	9591085	1731	87	✓	✗
9591088	NULL	9591088	4820	72	✓	✗
9591093	NULL	9591093	4101	35	✓	✗
9591095	NULL	9591095	4861	96	✓	✗
9591103	NULL	9591103	90	11	✓	✗
9591105	NULL	9591105	1146	35	✓	✗

- Select the + to the right of the **DerivedColumnsForMerge** step, then select the **Filter** destination from the context menu.



We are adding the Filter step to remove any records where the `ProductId` is null. The data sets have a small percentage of invalid records, and null `ProductId` values will cause errors when loading into the `UserTopProductPurchases` dedicated SQL pool table.

28. Set the **Filter on** expression to `!isNull(productId)`.

Filter settings Optimize Inspect Data preview ●

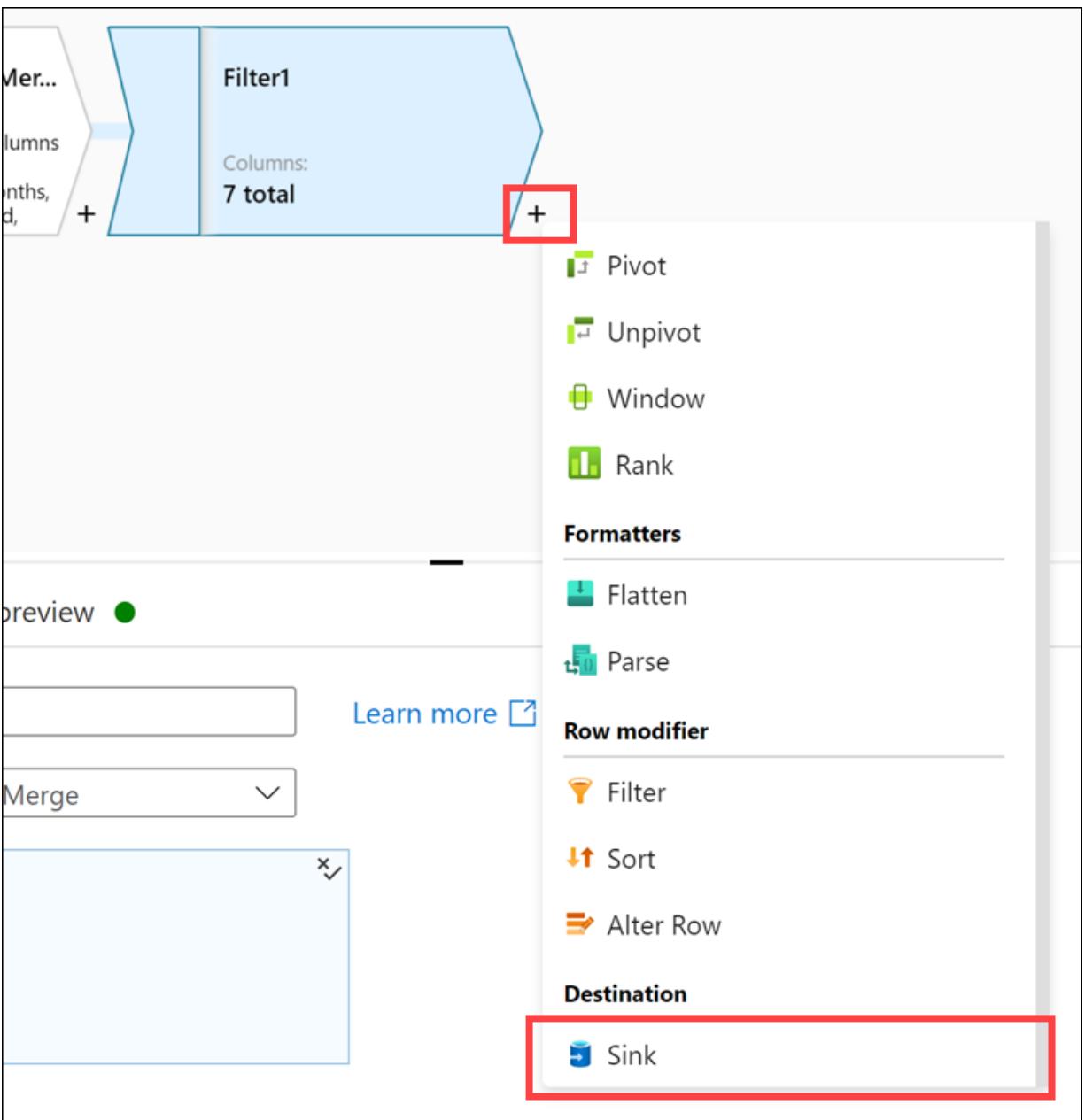
Output stream name * Filter1 [Learn more](#)

Incoming stream * DerivedColumnsForMerge

Filter on *

!isNull(productId)

29. Select the + to the right of the Filter1 step, then select the **Sink** destination from the context menu.



30. Under **Sink**, configure the following:

- **Output stream name:** Enter UserTopProductPurchasesASA.
- **Incoming stream:** Select Filter1.
- **Sink type:** select Dataset.
- **Dataset:** Select asal400_wwi_usertopproductpurchases_asa, which is the UserTopProductPurchases SQL table.
- **Options:** Check Allow schema drift and uncheck Validate schema.

The screenshot shows the 'Sink' configuration page. The tabs at the top are Sink, Settings, Mapping, Optimize, Inspect, and Data preview. The 'Sink' tab is selected. The configuration includes:

- Output stream name ***: UserTopProductPurchasesASA
- Incoming stream ***: Filter1
- Sink type ***: Dataset (selected)
- Dataset ***: asal400_wwi_usertopproductpurchases_asa (with a 'Test connection' button)
- Options**:
 - Allow schema drift
 - Validate schema

31. Select **Settings**, then configure the following:

- **Update method:** Check Allow insert and leave the rest unchecked.
- **Table action:** Select Truncate table.
- **Enable staging:** Check this option. Since we are importing a lot of data, we want to enable staging to improve performance.

The screenshot shows the 'Settings' configuration page. The tabs at the top are Sink, Settings, Mapping, Optimize, Inspect, and Data preview. The 'Settings' tab is selected. The configuration includes:

- Update method**: A group of checkboxes where 'Allow insert' is checked and highlighted with a red box.
- Table action**: A group of radio buttons where 'Truncate table' is selected and highlighted with a red box.
- Enable staging**: A checkbox which is checked and highlighted with a red box.
- Batch size**: An input field for specifying batch sizes.

32. Select **Mapping**, then configure the following:

- **Auto mapping:** Uncheck this option.
- **Columns:** Provide the following information:

Input columns	Output columns
userId	UserId
productId	ProductId
itemsPurchasedLast12Months	ItemsPurchasedLast12Months
isTopProduct	IsTopProduct
isPreferredProduct	IsPreferredProduct

Sink Settings **Mapping** Optimize Inspect Data preview ● Description

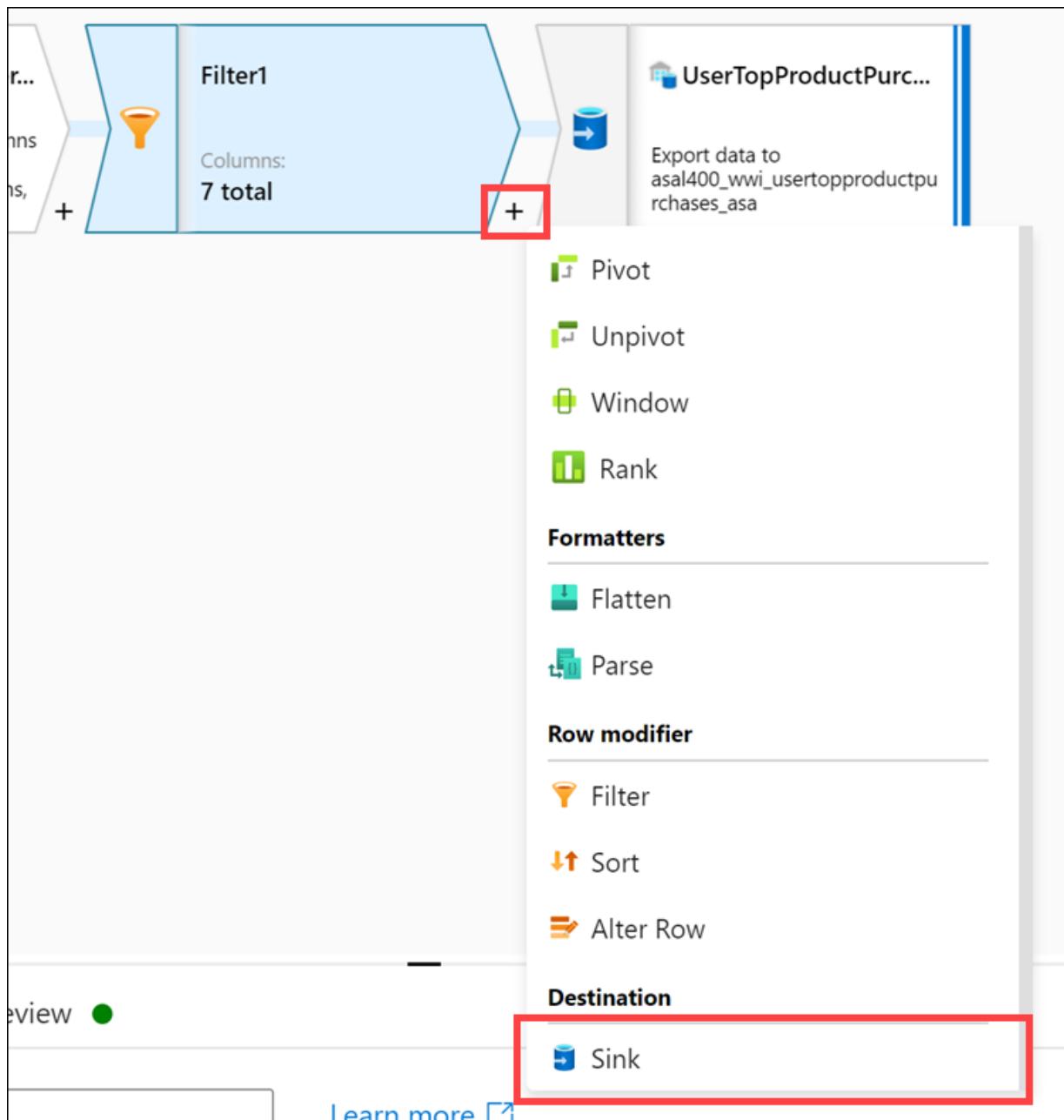
Options

Skip duplicate input columns ○
 Skip duplicate output columns ○

Auto mapping ○ ↻ Reset + Add mapping Delete Output format 5 mappings: All outputs mapped

Input columns	Y	Output columns	Y
123 userId	→	123 UserId	+ █
123 productId	→	123 ProductId	+ █
123 itemsPurchasedLast12Months	→	123 ItemsPurchasedLast12Months	+ █
✗ isTopProduct	→	✗ IsTopProduct	+ █
✗ isPreferredProduct	→	✗ IsPreferredProduct	+ █

33. Select the **+** to the right of the **Filter1** step, then select the **Sink** destination from the context menu to add a second sink.



34. Under **Sink**, configure the following:

- **Output stream name:** Enter DataLake.
- **Incoming stream:** Select Filter1.
- **Sink type:** select Inline.
- **Inline dataset type:** select Delta.
- **Linked service:** Select the default workspace data lake storage account (example: asaworkspaceinaday84-Wor...).
- **Options:** Check Allow schema drift and uncheck Validate schema.

Sink Settings Mapping Optimize Inspect Data preview

Output stream name * DataLake [Learn more](#)

Incoming stream * Filter1

Sink type *

- Dataset
- Inline
- Cache

Inline dataset type * Delta

Linked service * asaworkspacedeilt42-WorkspaceDef... [Test connection](#) [Edit](#) [New](#)

Integration runtime * AutoResolveIntegrationRuntime [Edit](#)

Options

- Allow schema drift ⓘ
- Validate schema ⓘ

35. Select **Settings**, then configure the following:

- **Folder path:** Enter `wwi-02/top-products` (**copy and paste** these two values into the fields since the `top-products` folder does not yet exist).
- **Compression type:** Select `snappy`.
- **Compression level:** Select `Fastest`.
- **Vacuum:** Enter `0`.
- **Truncate table:** Select.
- **Update method:** Check `Allow insert` and leave the rest unchecked.
- **Merge schema (under Delta options):** Unchecked.

Sink **Settings** Mapping Optimize Inspect Data preview

Folder path * wwi-02 / top-products [Browse](#)

Compression type snappy

Compression level Fastest

Vacuum 0

Table action None Overwrite ⓘ Truncate ⓘ

Update method

- Allow insert
- Allow delete
- Allow upsert
- Allow update

▲ Delta options

Merge schema

36. Select **Mapping**, then configure the following:

- **Auto mapping:** Uncheck this option.
- **Columns:** Provide the following information:

Input columns	Output columns
visitorId	visitorId
productId	productId
itemsPurchasedLast12Months	itemsPurchasedLast12Months
preferredProductId	preferredProductId
userId	userId
isTopProduct	isTopProduct
isPreferredProduct	isPreferredProduct

Sink Settings **Mapping** Optimize Inspect Data preview

Options

Skip duplicate input columns ⓘ

Skip duplicate output columns ⓘ

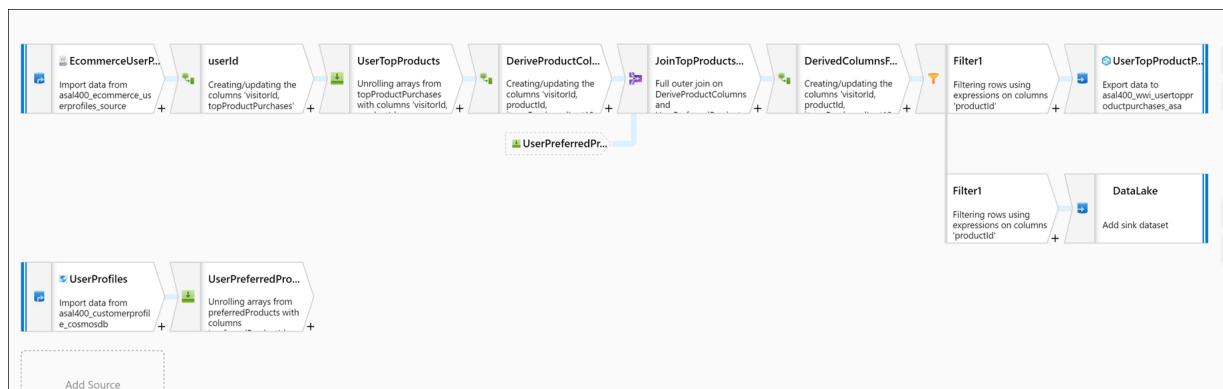
Auto mapping ⓘ

+ Add mapping Delete Reset Import schema View schema

Input columns	Output columns
visitorId	visitorId
productId	productId
itemsPurchasedLast12Months	itemsPurchasedLast12Months
preferredProductId	preferredProductId
userId	userId
isTopProduct	isTopProduct
isPreferredProduct	isPreferredProduct

Notice that we have chosen to keep more fields for the data lake sink vs. the SQL pool sink (**visitorId** and **preferredProductId**). This is because we aren't adhering to a fixed destination schema (like a SQL table), and because we want to retain the original data as much as possible in the data lake.

37. Your completed data flow should look similar to the following:



38. Select **Publish all**, then **Publish** to save your new data flow.



10.5 Lab 2: Orchestrate data movement and transformation in Azure Synapse Pipelines

Tailwind Traders is familiar with Azure Data Factory (ADF) pipelines and wants to know if Azure Synapse Analytics can either integrate with ADF or has a similar capability. They want to orchestrate data ingest, transformation, and load activities across their entire data catalog, both internal and external to their data warehouse.

You recommend using Synapse Pipelines, which includes over 90 built-in connectors, can load data by manual execution of the pipeline or by orchestration, supports common loading patterns, enables fully parallel loading into the data lake or SQL tables, and shares a code base with ADF.

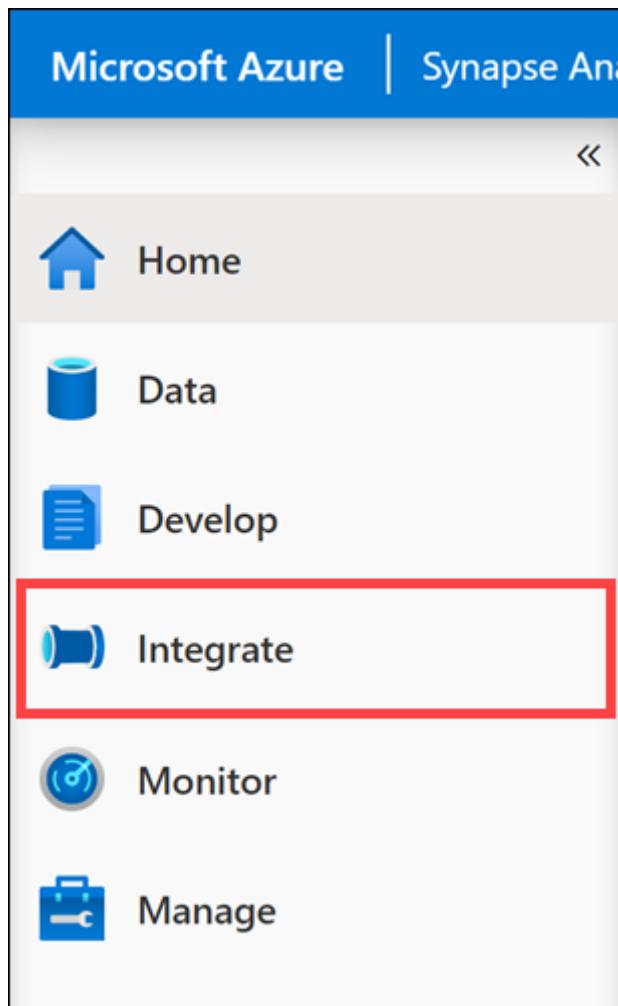
By using Synapse Pipelines, Tailwind Traders can experience the same familiar interface as ADF without having to use an orchestration service outside of Azure Synapse Analytics.

10.5.1 Exercise 1: Create, trigger, and monitor pipeline

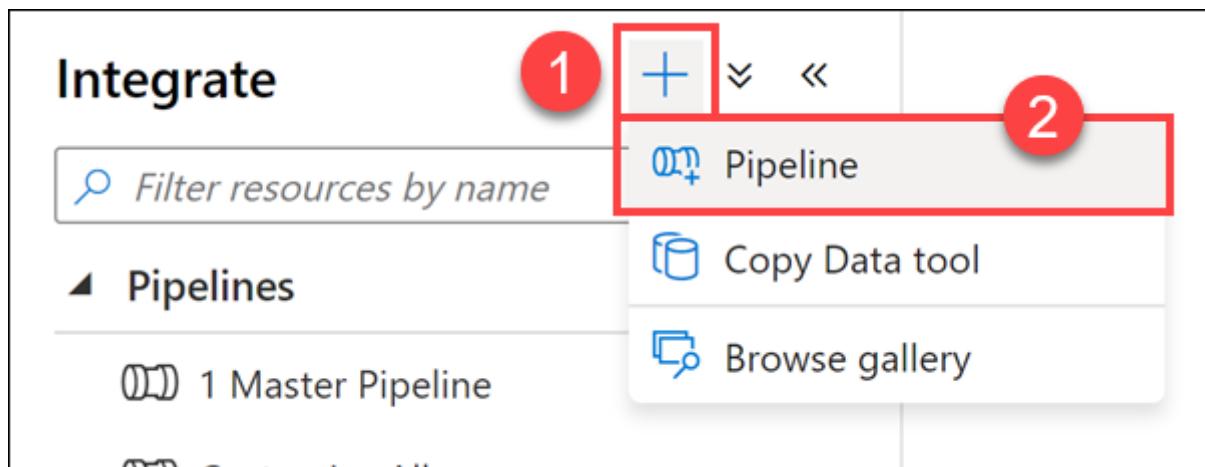
10.5.1.1 Task 1: Create pipeline

Let's start by executing our new Mapping Data Flow. In order to run the new data flow, we need to create a new pipeline and add a data flow activity to it.

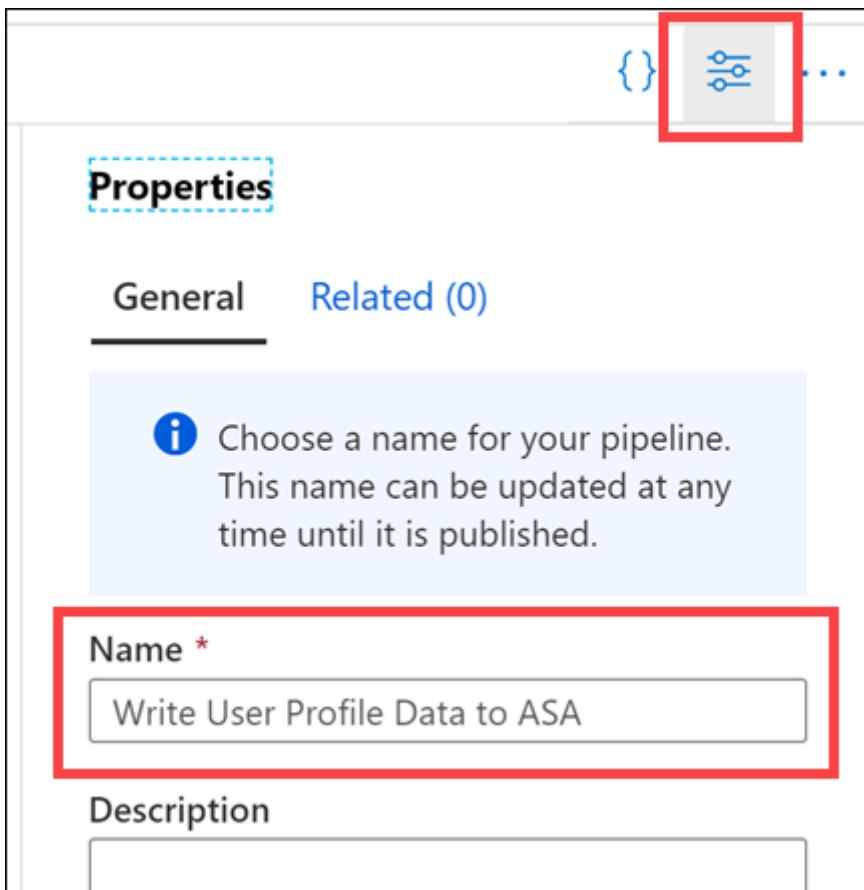
1. Navigate to the **Integrate** hub.



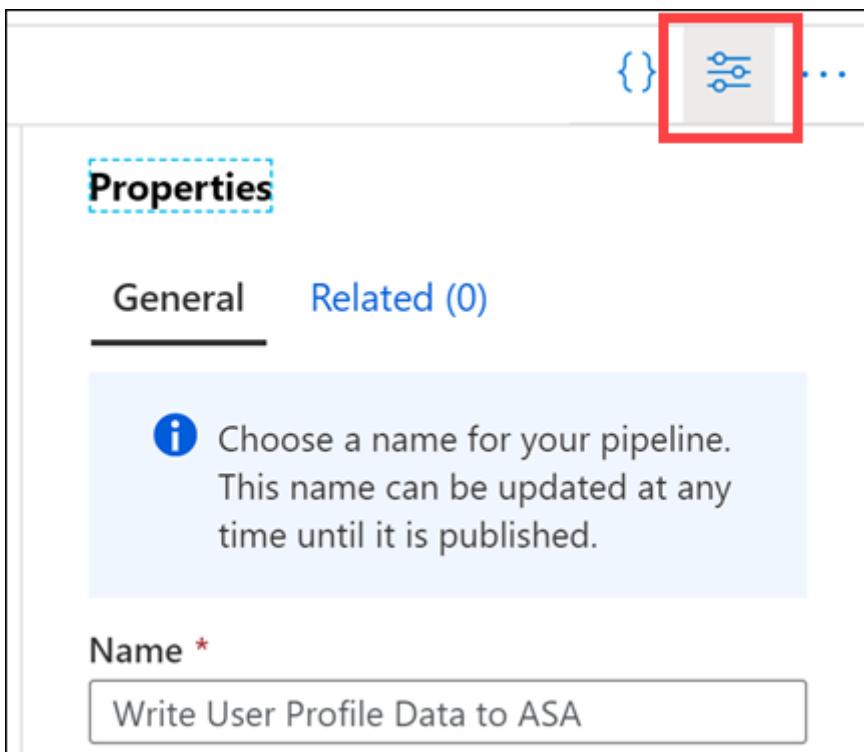
2. Select + (1), then Pipeline (2).



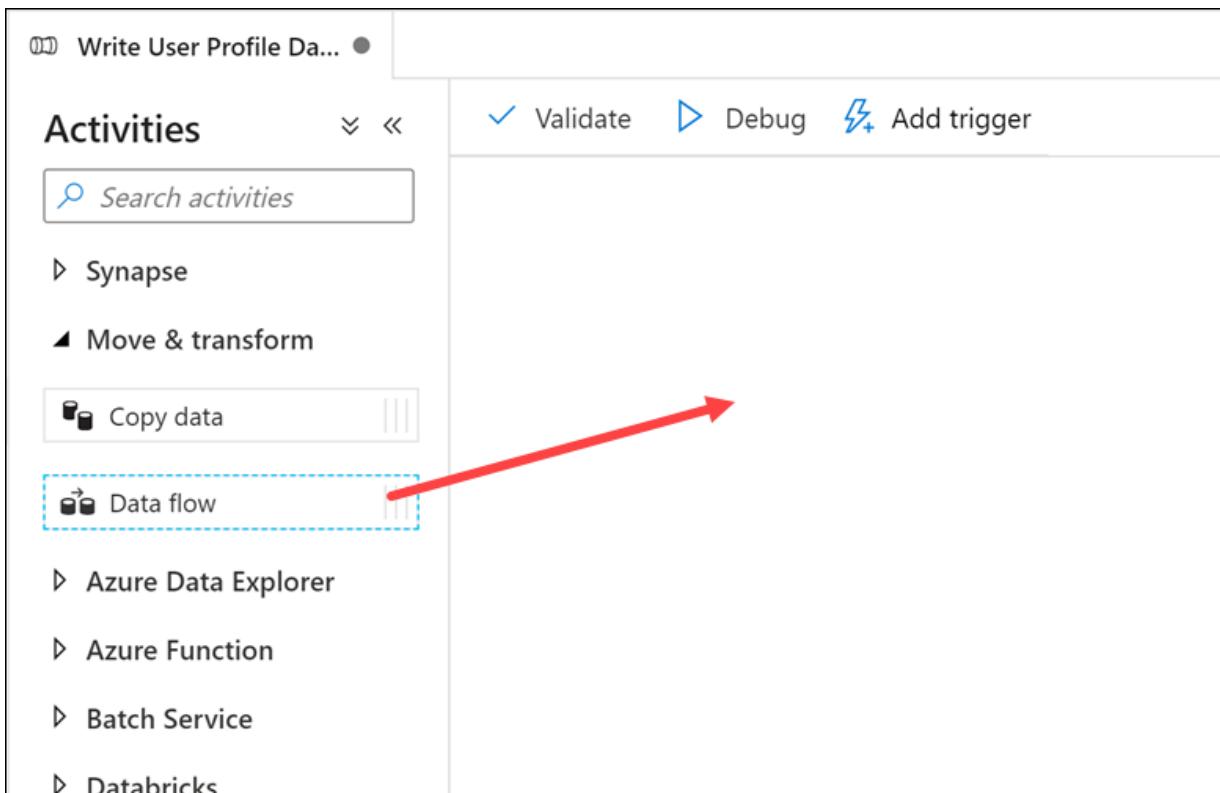
3. In the **General** section of the **Profiles** pane of the new data flow, update the **Name** to the following:
Write User Profile Data to ASA.



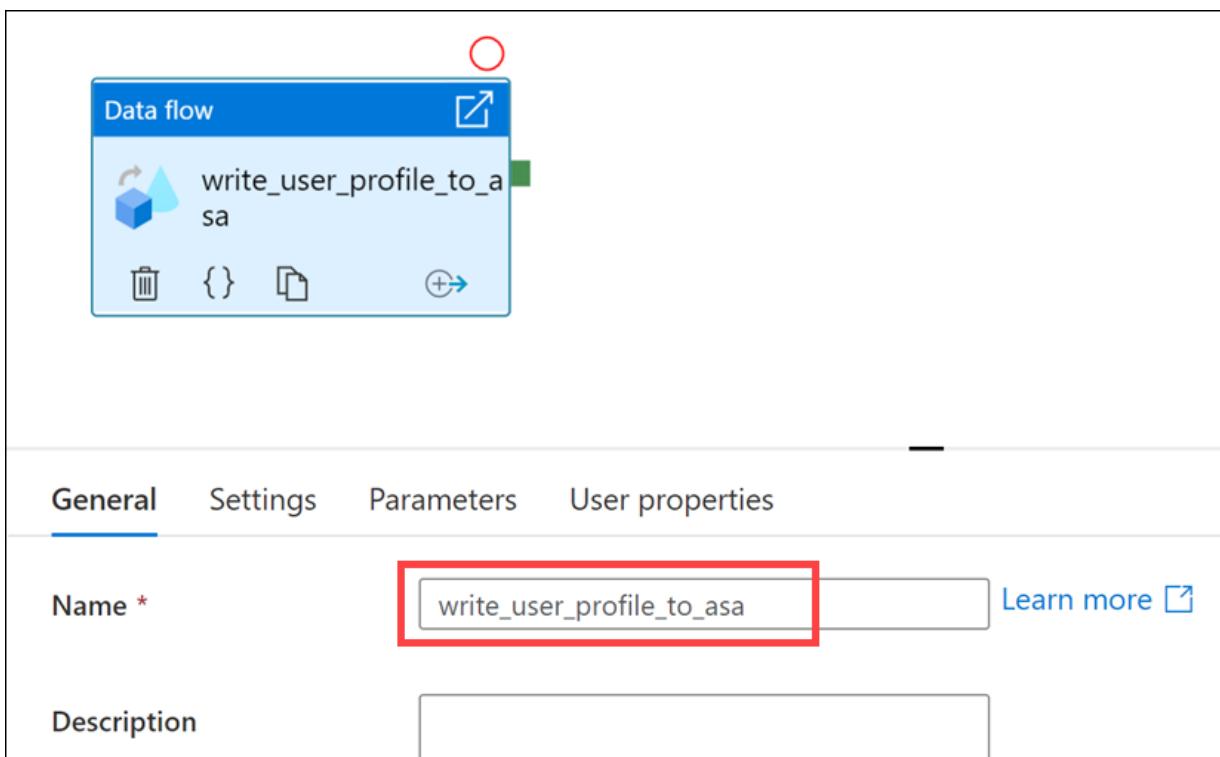
4. Select the **Properties** button to hide the pane.



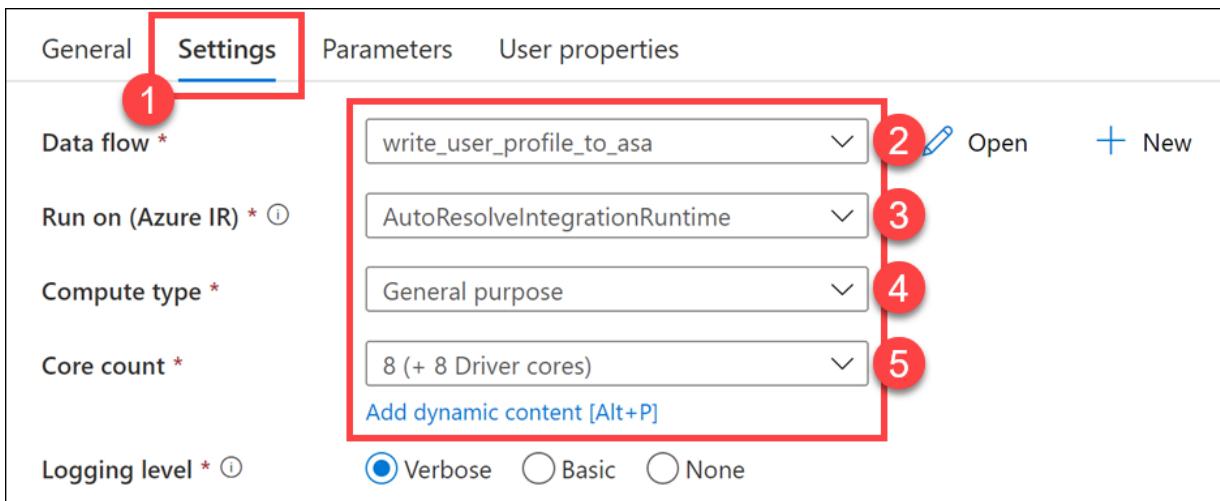
5. Expand **Move & transform** within the Activities list, then drag the **Data flow** activity onto the pipeline canvas.



6. Under the **General** tab, set the Name to `write_user_profile_to_asa`.



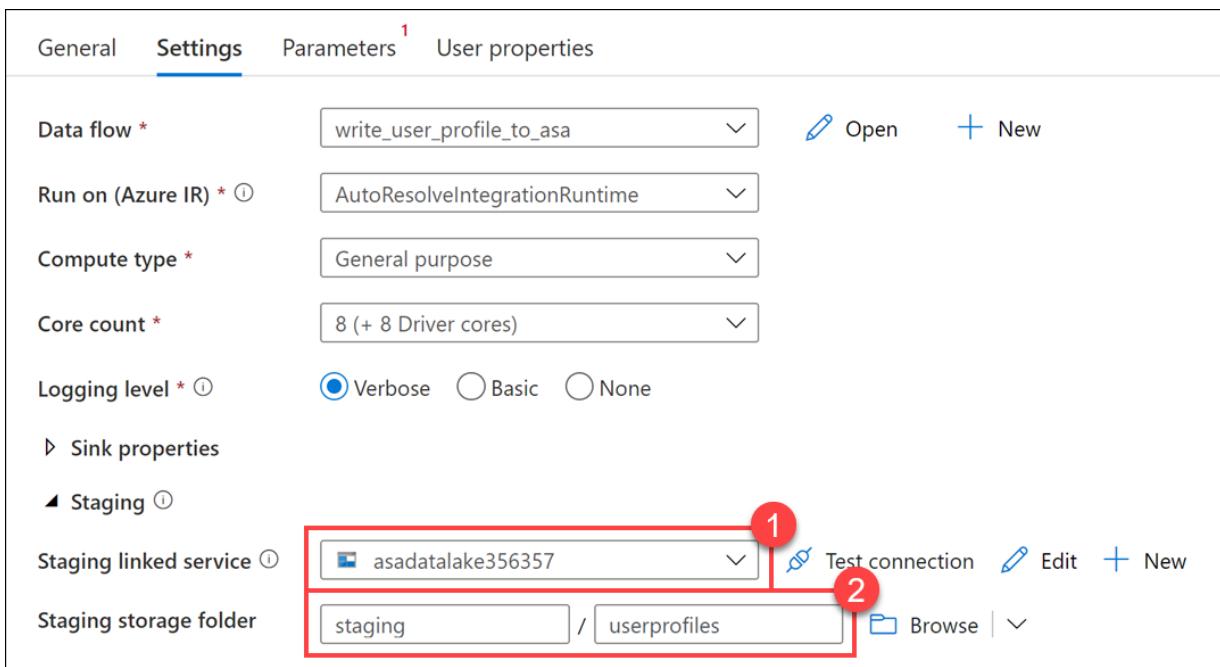
7. Select the **Settings** tab (1). Select `write_user_profile_to_asa` for **Data flow** (2), then ensure `AutoResolveIntegrationRuntime` is selected for **Run on (Azure IR)** (3). Choose the **General purpose Compute type** (4) and select `8 (+ 8 cores)` for the **Core count** (5).



8. Expand **Staging** and configure the following:

- **Staging linked service:** Select the `asadatalakeSUFFIX` linked service.
- **Staging storage folder:** Enter `staging/userprofiles`. The `userprofiles` folder will be automatically created for you during the first pipeline run.

Copy and paste the `staging` and `userprofiles` folder names into the two fields.



The staging options under PolyBase are recommended when you have a large amount of data to move into or out of Azure Synapse Analytics. You will want to experiment with enabling and disabling staging on the data flow in a production environment to evaluate the difference in performance.

9. Select **Publish all** then **Publish** to save your pipeline.

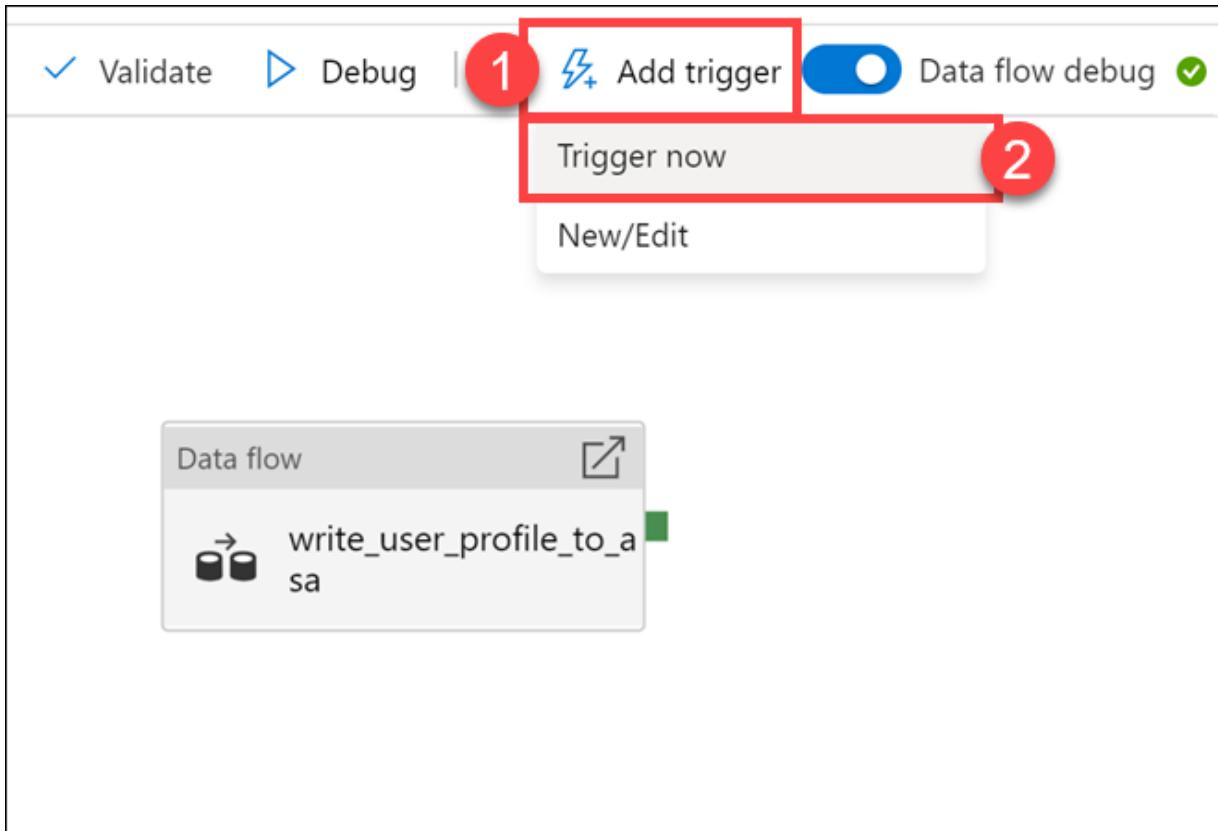


10.5.1.2 Task 2: Trigger, monitor, and analyze the user profile data pipeline

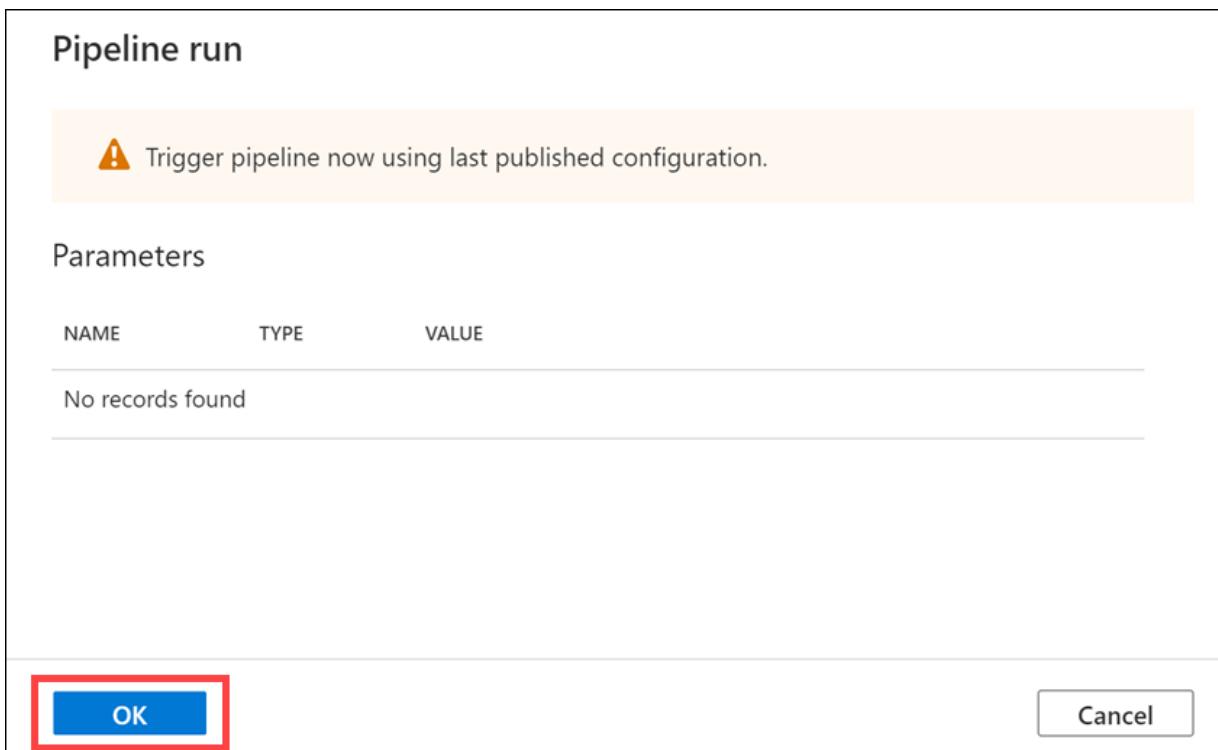
Tailwind Traders wants to monitor all pipeline runs and view statistics for performance tuning and troubleshooting purposes.

You have decided to show Tailwind Traders how to manually trigger, monitor, then analyze a pipeline run.

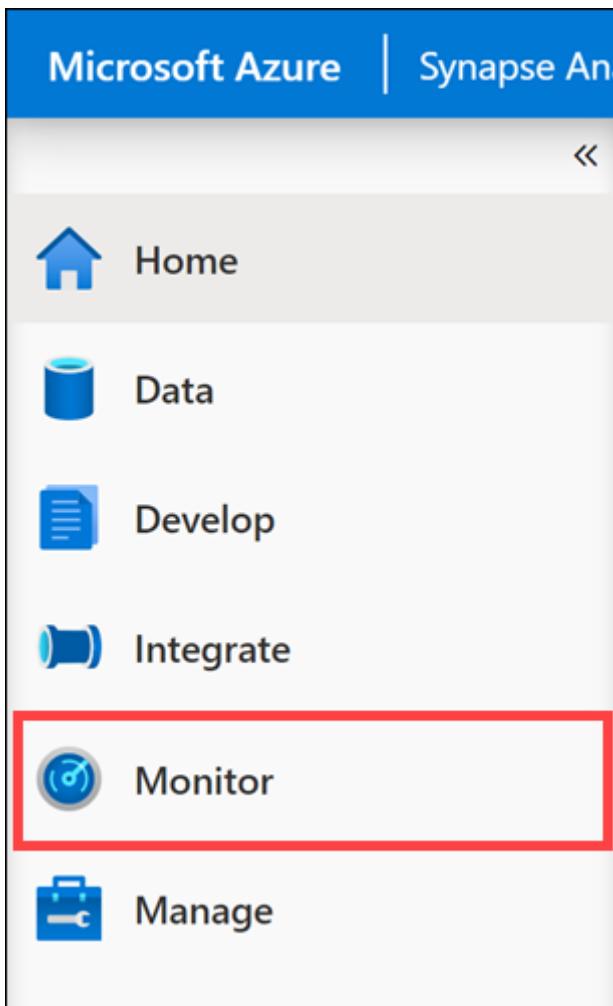
- At the top of the pipeline, select **Add trigger** (1), then **Trigger now** (2).



- There are no parameters for this pipeline, so select **OK** to run the trigger.



- Navigate to the **Monitor** hub.



4. Select **Pipeline runs** (1) and wait for the pipeline run to successfully complete (2). You may need to refresh (3) the view.

While this is running, read the rest of the lab instructions to familiarize yourself with the content.

The screenshot shows the 'Pipeline runs' page in the Azure Synapse Analytics portal. The left sidebar shows integration options like Pipeline runs, Trigger runs, and Integration runtimes. The main area displays a table of pipeline runs:

Run ID	Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
1	Write User Profile Data to ASA	1/19/21, 6:29:04 PM	1/19/21, 6:34:44 PM	00:05:40	Manual trigger	✓ Succeeded	Original
2	Copy December Sales	1/19/21, 5:35:06 PM	1/19/21, 5:35:50 PM	00:00:44	Manual trigger	✓ Succeeded	Original

5. Select the name of the pipeline to view the pipeline's activity runs.

Pipeline runs

Showing 1 - 8 items

Pipeline name	Run start ↑↓
Write User Profile Data to ASA	2/27/21, 2:04:35 PM

6. Hover over the data flow activity name in the **Activity runs** list, then select the **Data flow details** icon.

Activity runs

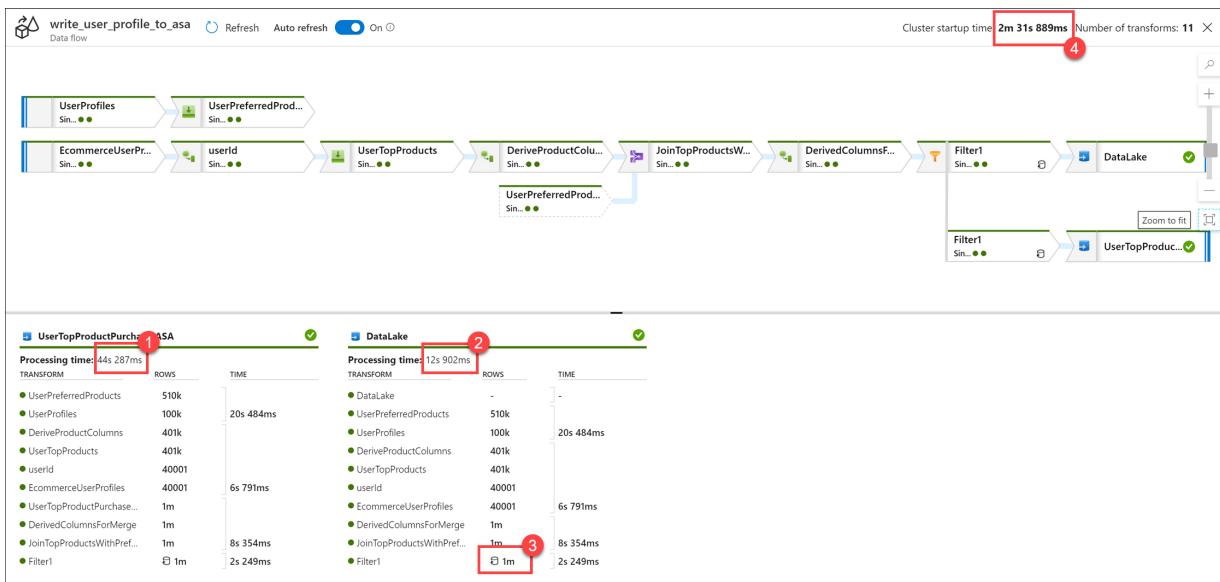
Pipeline run ID 0c17cf90-42e2-41fb-8ce5-35a203c66f28

All status ▾

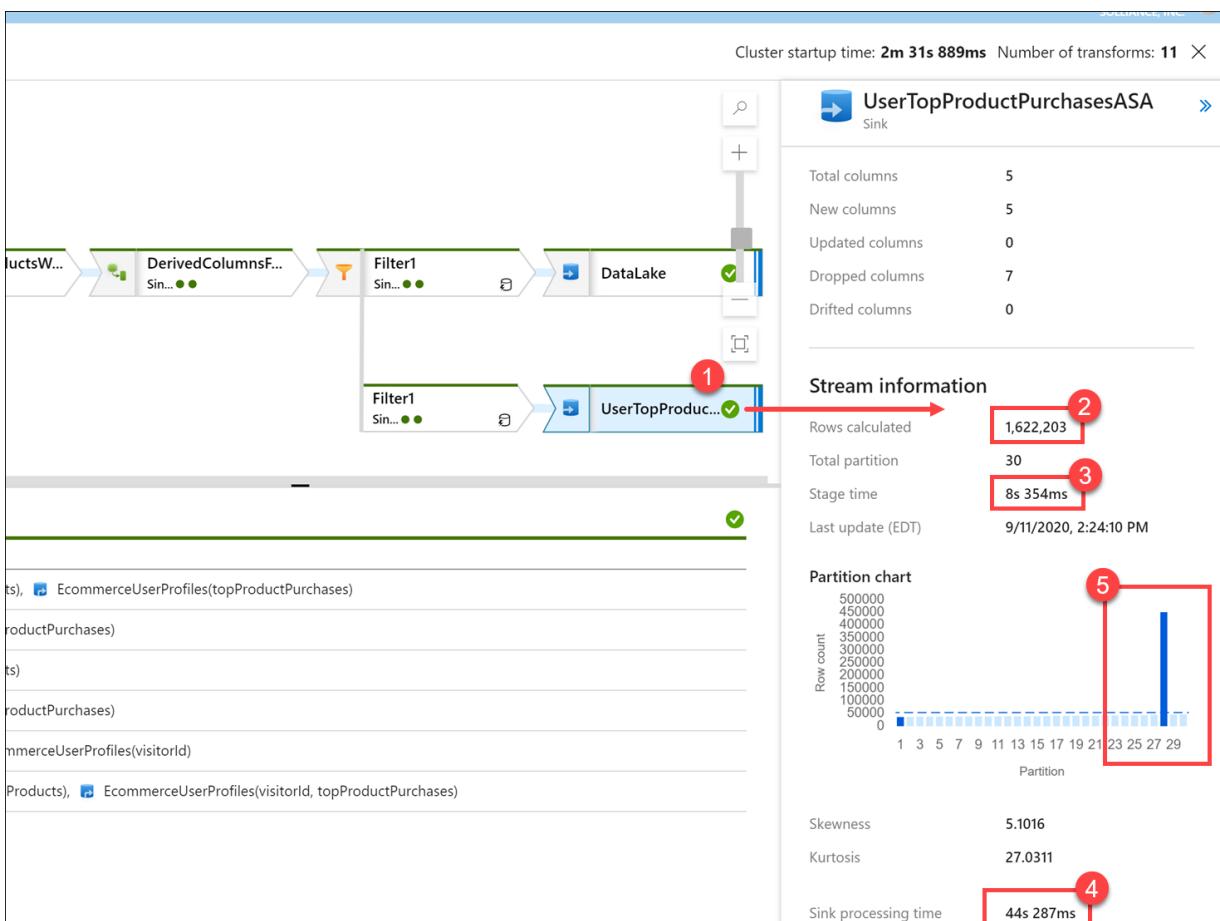
Showing 1 - 1 of 1 items

Activity name	Activity type	Run start ↑↓	Duration	Status	Integration runtime
write_user_pr... ↗ ↘	Data flow	2/27/21, 2:04:37 PM	00:09:06	Succeeded	DefaultIntegrationRuntime (Ea

7. The data flow details displays the data flow steps and processing details. In our example, processing time took around **44 seconds to process** the SQL pool sink (1), and around **12 seconds to process** the Data Lake sink (2). The Filter1 output was around **1 million rows** (3) for both. You can see which activities took the longest to complete. The cluster startup time contributed over **2.5 minutes** (4) to the total pipeline run.



8. Select the **UserTopProductPurchasesASA** sink (1) to view its details. We can see that **1,622,203 rows** were calculated (2) with a total of 30 partitions. It took around **eight seconds** to stage the data (3) in ADLS Gen2 prior to writing the data to the SQL table. The total sink processing time in our case was around **44 seconds** (4). It is also apparent that we have a **hot partition** (5) that is significantly larger than the others. If we need to squeeze extra performance out of this pipeline, we can re-evaluate data partitioning to more evenly spread the partitions to better facilitate parallel data loading and filtering. We could also experiment with disabling staging to see if there's a processing time difference. Finally, the size of the dedicated SQL pool plays a factor in how long it takes to ingest data into the sink.

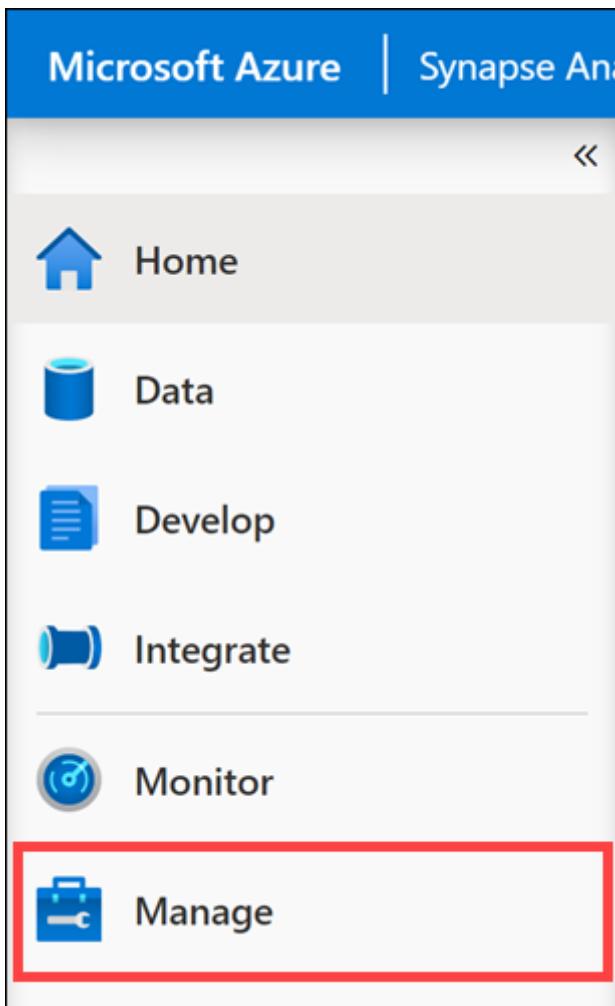


10.5.2 Exercise 2: Cleanup

Complete these steps to free up resources you no longer need.

10.5.2.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



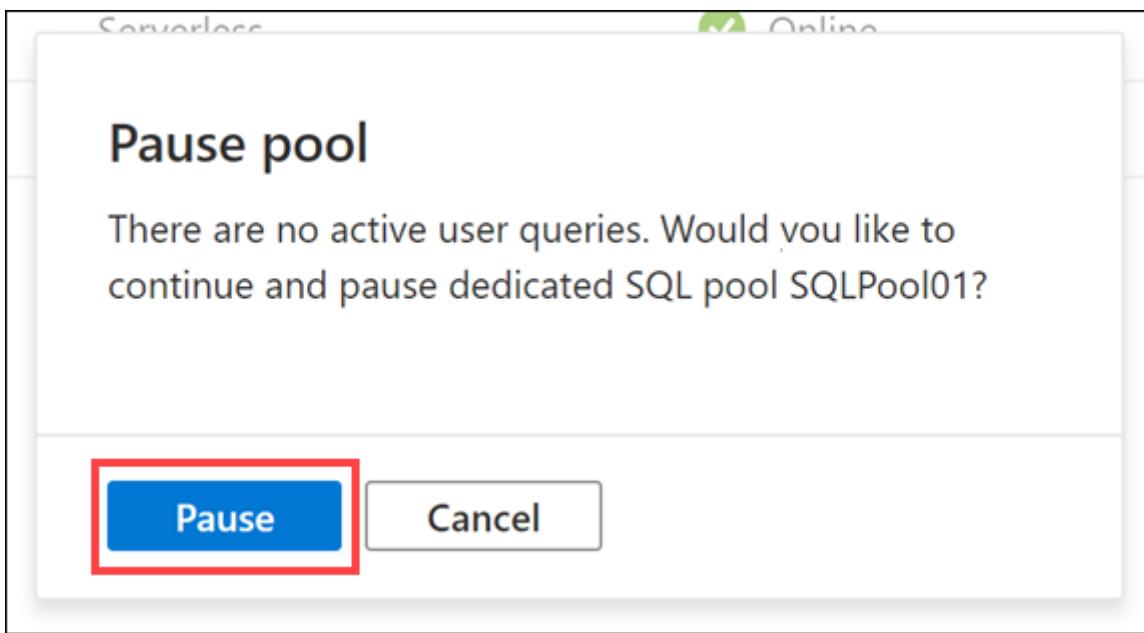
3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

This screenshot shows the "SQL pools" page from the Synapse Analytics portal. On the left, there's a sidebar with "Analytics pools" expanded, showing "SQL pools" (marked with a red box and a red number "1") and other options like "Apache Spark pools", "External connections", etc. The main content area has a heading "SQL pools" with a message about serverless and dedicated pools. It includes a "New" button, a "Refresh" button, and a "System-assigned managed identity" toggle. There's also a "Filter by name" input field. A table lists the available pools:

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

A red box highlights the "Pause" button for the "SQLPool01" row, and a red number "2" is placed above it. A red box also surrounds the "SQLPool01" row itself.

4. When prompted, select **Pause**.



11 Module 9 - Integrate data from notebooks with Azure Data Factory or Azure Synapse Pipelines

The student will learn how to create linked services, and orchestrate data movement and transformation in Azure Synapse Pipelines.

In this module, the student will be able to:

- Orchestrate data movement and transformation in Azure Synapse Pipelines

11.1 Lab details

- [Module 9 - Integrate data from notebooks with Azure Data Factory or Azure Synapse Pipelines](#)
 - [Lab details](#)
 - [Lab setup and pre-requisites](#)
 - [Exercise 1: Linked service and datasets](#)
 - * [Task 1: Create linked service](#)
 - * [Task 2: Create datasets](#)
 - [Exercise 2: Create mapping data flow and pipeline](#)
 - * [Task 1: Retrieve the ADLS Gen2 linked service name](#)
 - * [Task 2: Create mapping data flow](#)
 - * [Task 3: Create pipeline](#)
 - * [Task 4: Trigger the pipeline](#)
 - [Exercise 3: Create Synapse Spark notebook to find top products](#)
 - * [Task 1: Create notebook](#)
 - * [Task 2: Add the Notebook to the pipeline](#)

TODO: Include the data sources, mapping data flow, and pipeline from Module 9 in the setup for Module 10.

11.2 Lab setup and pre-requisites

Note: Only complete the [Lab setup and pre-requisites](#) steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 1.

[Complete the lab setup instructions](#) for this module.

Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)

- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

11.3 Exercise 1: Linked service and datasets

Note: Complete this exercise if you **have not** completed Module 8, or if you do not have the following Synapse artifacts:

- Linked services:
 - `asacosmosdb01`
- Datasets:
 - `asal400_ecommerce_userprofiles_source`
 - `asal400_customerprofile_cosmosdb`

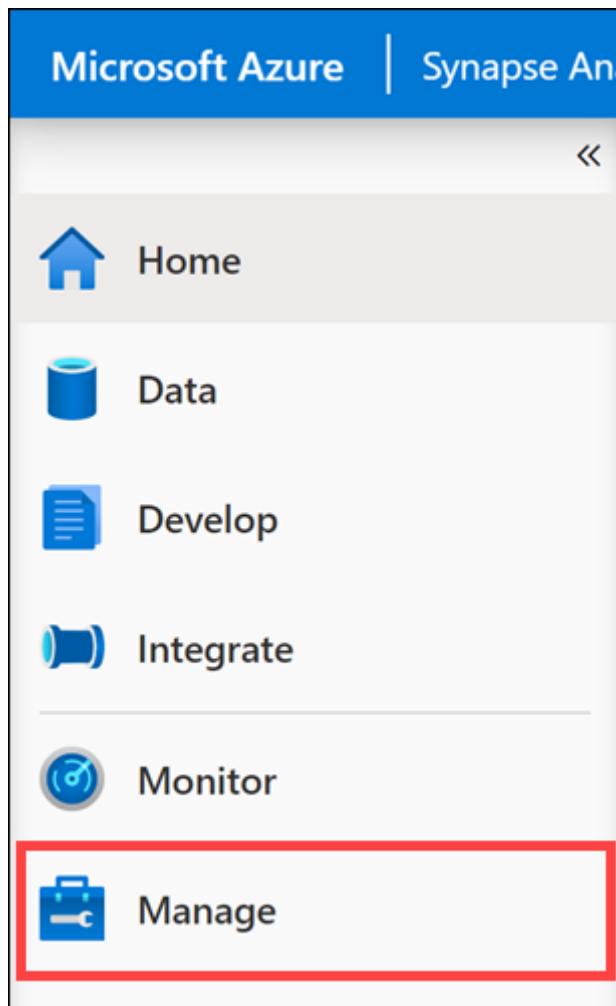
If you completed Module 8 or already have these artifacts, skip ahead to Exercise 2.

11.3.1 Task 1: Create linked service

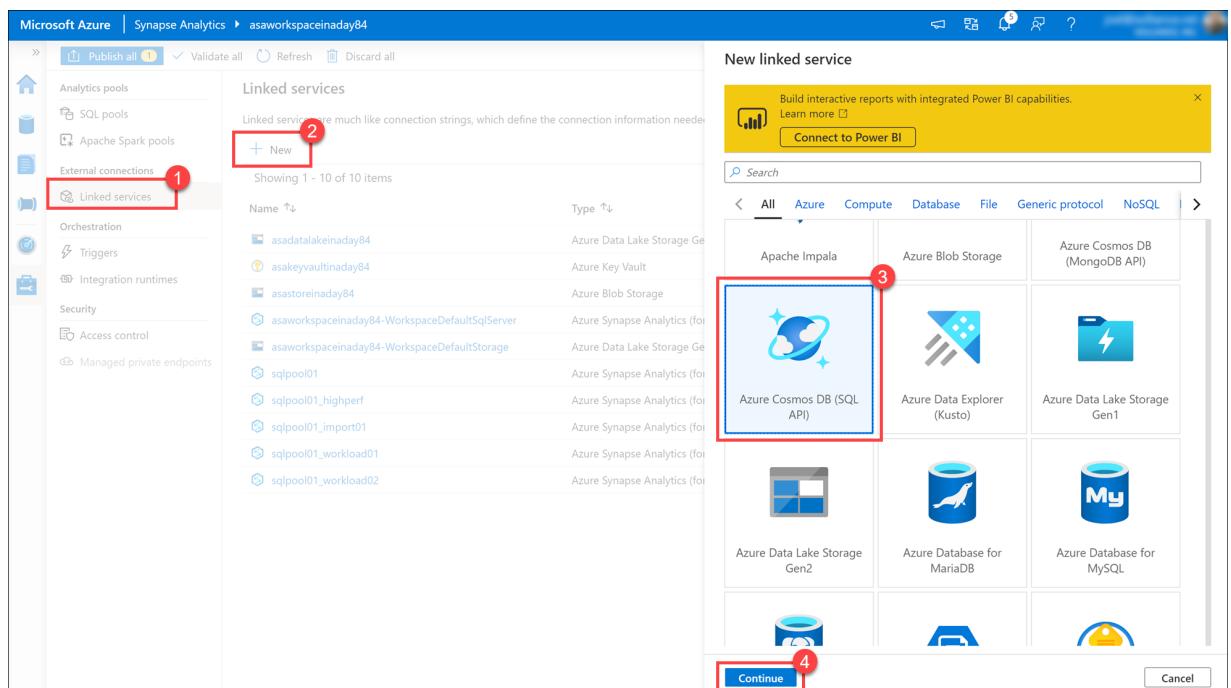
Complete the steps below to create an Azure Cosmos DB linked service.

Note: Skip this section if you have already created a Cosmos DB linked service.

1. Open Synapse Studio (<https://web.azuresynapse.net/>), and then navigate to the **Manage** hub.



2. Open **Linked services** and select **+ New** to create a new linked service. Select **Azure Cosmos DB (SQL API)** in the list of options, then select **Continue**.



3. Name the linked service **asacosmosdb01** (1), select the **Cosmos DB account name** (**asacosmosdbSUFFIX**) and set the **Database name** value to **CustomerProfile** (2). Select **Test connection** to ensure success (3), then select **Create** (4).

New linked service (Azure Cosmos DB (SQL API))

Choose a name for your linked service. This name cannot be updated later.

Name * asacosmosdb01 1

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime ▽ edit

Connection string Azure Key Vault

Account selection method

From Azure subscription Enter manually

Azure subscription

Select all ▽

Azure Cosmos DB account name * asacosmosdbinaday84 2

Database name * CustomerProfile ▽ refresh

Additional connection properties

+ New

Annotations

1 ..

Create Back 3

Connection successful 4

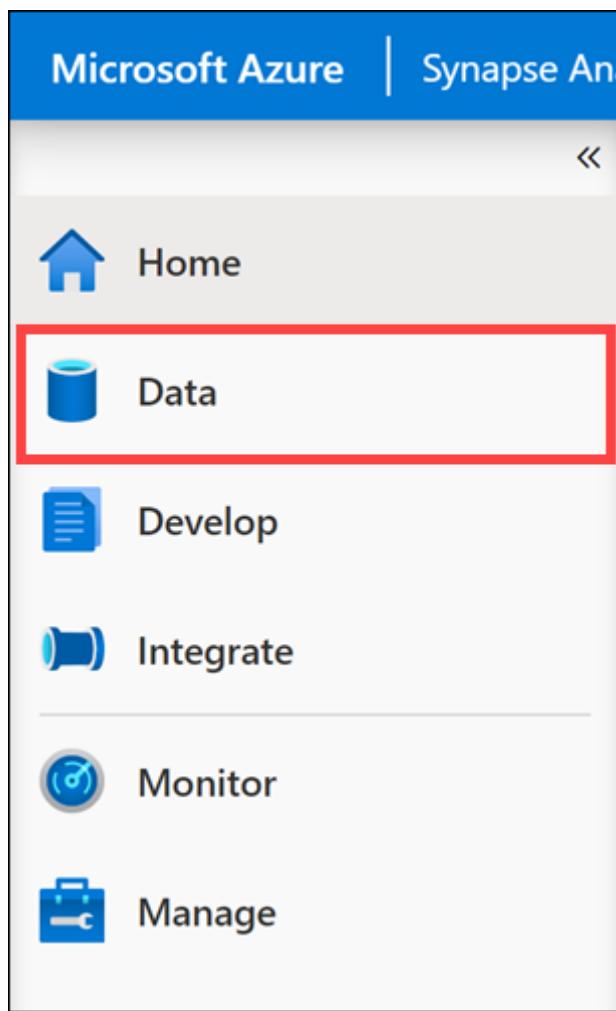
Test connection Cancel

11.3.2 Task 2: Create datasets

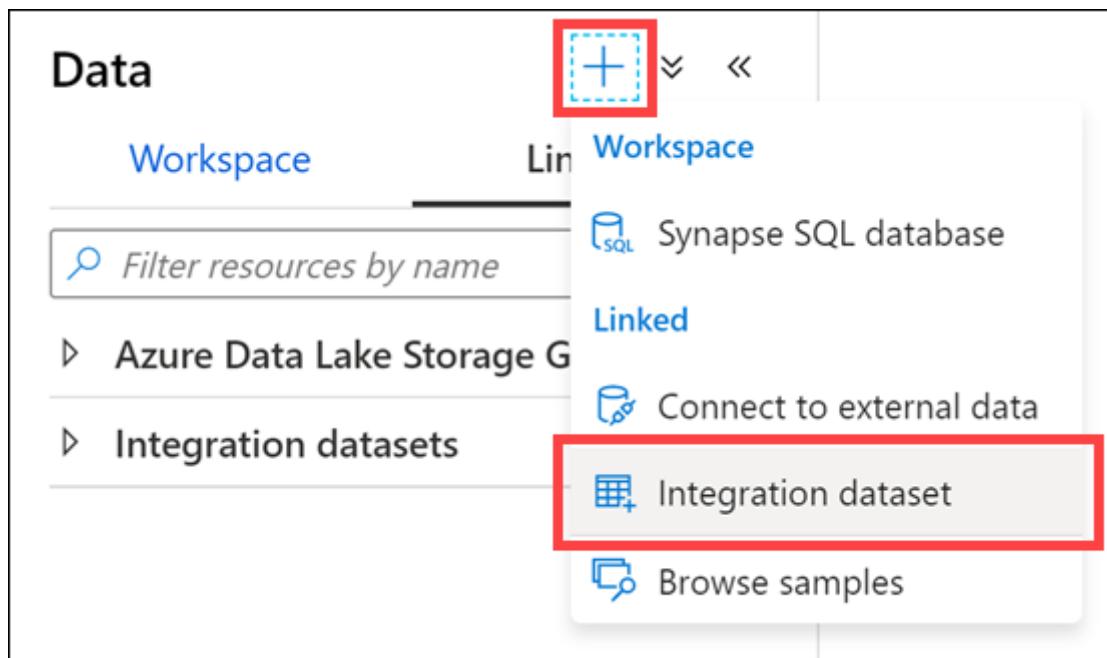
Complete the steps below to create the `asal400_customerprofile_cosmosdb` dataset.

Note to presenter: Skip this section if you have already completed Module 4.

1. Navigate to the **Data** hub.



2. Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



3. Select **Azure Cosmos DB (SQL API)** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

The screenshot shows the 'New integration dataset' dialog. At the top, there's a search bar with a magnifying glass icon and the word 'Search'. Below it is a navigation bar with tabs: All, Azure, Database, File, Generic protocol, NoSQL, and Services and apps. The 'All' tab is selected. The main area displays a grid of data store icons. The first icon in the top-left corner, representing 'Azure Cosmos DB (SQL API)', is highlighted with a red box and a red circle containing the number '1'. To its right is the 'Apache Impala' icon. In the second row, the third icon from the left, representing 'Azure Data Lake Storage Gen1', is highlighted with a red box and a red circle containing the number '2'. Other icons include 'Azure Blob Storage', 'Azure Data Explorer (Kusto)', 'Azure Data Lake Storage Gen1', 'Azure Data Lake Storage Gen2', 'Azure Database for MariaDB', and 'Azure Database for MySQL'. At the bottom left is a 'Continue' button with a red box and a red circle containing the number '2'. At the bottom right is a 'Cancel' button.

4. Configure the dataset with the following characteristics, then select **OK** (4):

- **Name:** Enter `asal400_customerprofile_cosmosdb` (1).
- **Linked service:** Select the Azure Cosmos DB linked service (2).
- **Collection:** Select `OnlineUserProfile01` (3).

Set properties

1 Choose a name for your dataset. This name can be updated at any time until it is published.

2 Choose a linked service.

3 Choose a collection.

4 Click OK.

Name
asal400_customerprofile_cosmosdb

Linked service *
asacosmosdb01

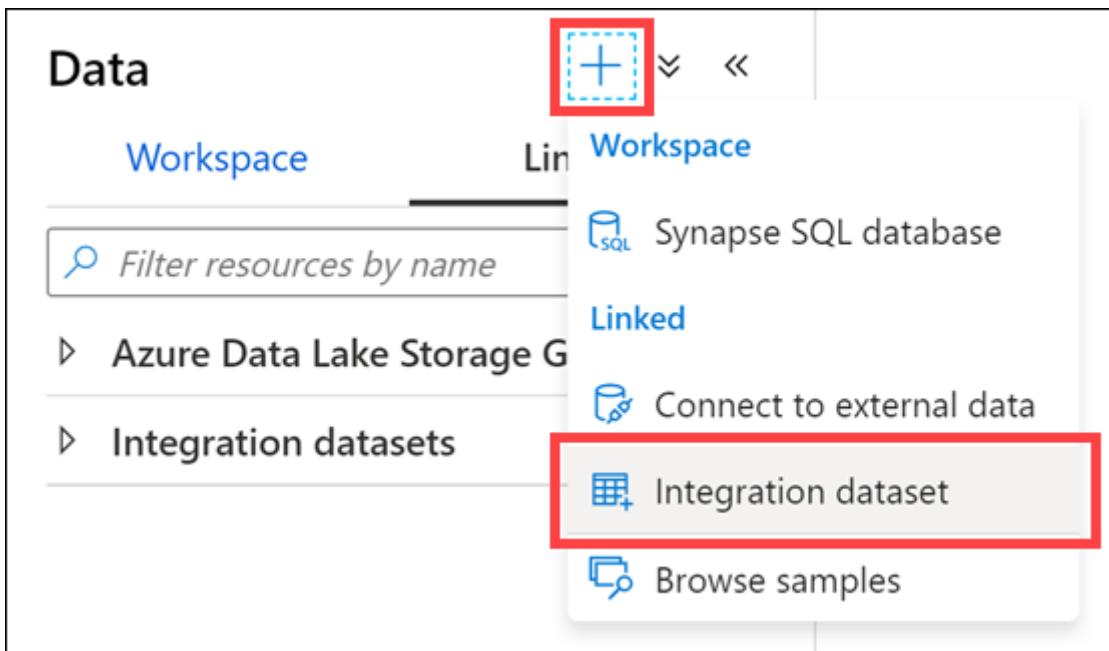
Connect via integration runtime *
AutoResolveIntegrationRuntime

Collection
OnlineUserProfile01

Import schema
 From connection/store None

OK Back Cancel

5. Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



6. Select **Azure Data Lake Storage Gen2** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

Search

All Azure Database File Generic protocol NoSQL Services and apps

 Azure Cosmos DB (SQL API)	 Azure Data Explorer (Kusto)	 Azure Data Lake Storage Gen1
 Azure Data Lake Storage Gen2	 Azure Database for MariaDB	 Azure Database for MySQL
 Azure Database for PostgreSQL	 Azure Databricks Delta Lake	 Azure File Storage
Continue		Cancel

1

2

7. Select the **JSON** format (1), then select **Continue** (2).

Select format

Choose the format type of your data



Avro



Binary



DelimitedText



Excel



Json



ORC



Parquet



XML

2
Continue

Back

Cancel

8. Configure the dataset with the following characteristics, then select **OK (5)**:

- **Name:** Enter `asal400_ecommerce_userprofiles_source` (1).
- **Linked service:** Select the `asadatalakeXX` linked service that already exists (2).
- **File path:** Browse to the `wwi-02/online-user-profiles-02` path (3).
- **Import schema:** Select `From connection/store` (4).

Set properties

1 Choose a name for your dataset. This name can be updated at any time until it is published.

2 Linked service *

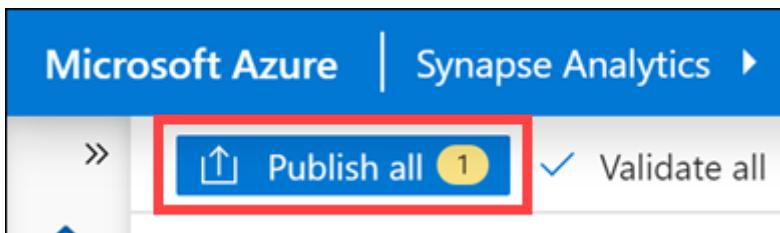
3 File path

4 Import schema

5 OK

The dialog box has a red border and several fields are highlighted with red boxes and numbered circles: 1 points to the 'Name' input field; 2 points to the 'Linked service' dropdown; 3 points to the 'File path' input field; 4 points to the 'Import schema' radio buttons; and 5 points to the 'OK' button.

9. Select **Publish all** then **Publish** to save your new resources.

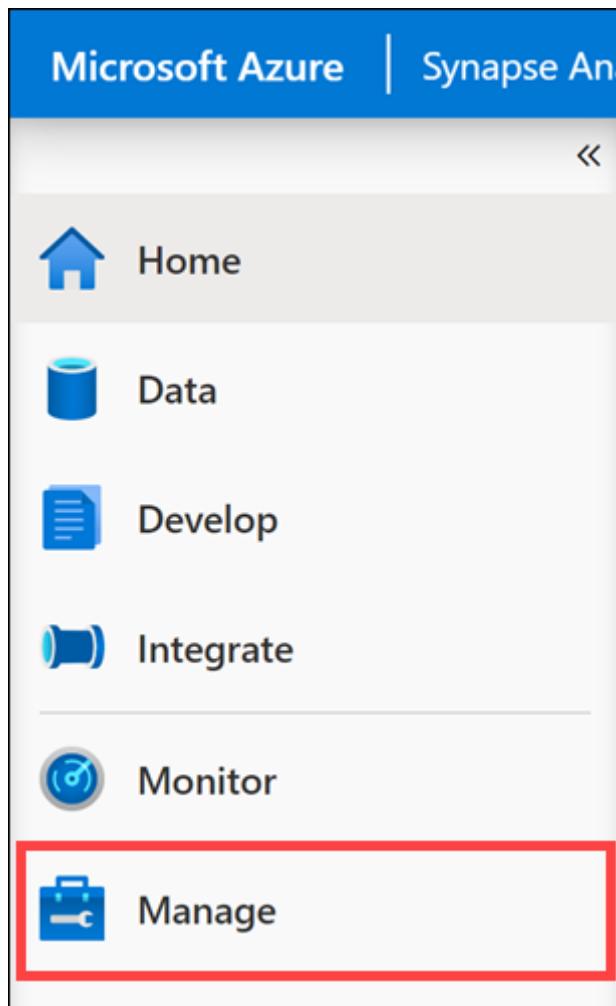


11.4 Exercise 2: Create mapping data flow and pipeline

In this exercise, you create a mapping data flow that copies user profile data to the data lake, then create a pipeline that orchestrates executing the data flow, and later on, the Spark notebook you create later in this lab.

11.4.1 Task 1: Retrieve the ADLS Gen2 linked service name

1. Navigate to the **Manage** hub.



2. Select **Liked services** on the left-hand menu. Locate an **Azure Data Lake Storage Gen2** linked service in the list, hover over the service, then select **{ } Code**.

The screenshot shows the 'Linked services' page in the Azure portal. On the left, there is a sidebar with sections for Analytics pools, External connections (with 'Linked services' highlighted by a red box), Integration, and Security. The main area displays a table of linked services. One row is selected, showing 'asadatalake356357' as the name, 'Azure Data Lake Storage Gen2' as the type, and a 'Code' button. A red box highlights the '{ }' icon next to the service name.

Name	Type
asadatalake356357	Azure Data Lake Storage Gen2
asacosmosdb01	Azure Cosmos DB (SQL API)
asakeyvault356357	Azure Key Vault
asastore356357	Azure Blob Storage

3. Copy the **name** of the linked service, then select **Cancel** to close the dialog. Save this value to Notepad or similar text editor to use later.

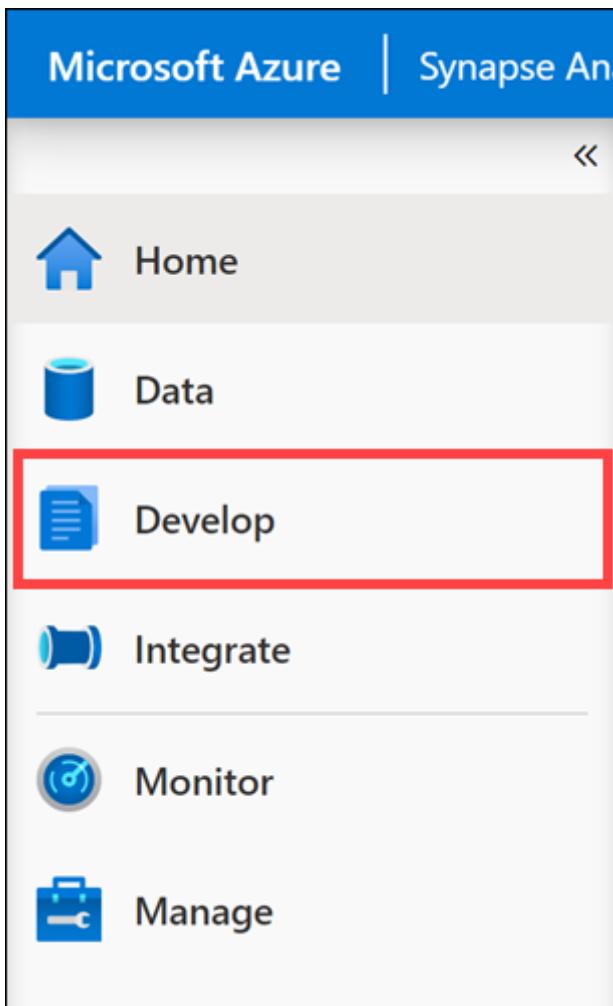
Copy to clipboard

```
1  {
2      "name": 'asadatalake356357',
3      "type": "Microsoft.Synapse/workspaces/linkedservices",
4      "properties": {
5          "annotations": [],
6          "type": "AzureBlobFS",
7          "typeProperties": {
8              "url": "https://asadatalake356357.dfs.core.windows.net",
9              "encryptedCredential": "ew0KICAiVmVyc2lvbiI6ICIyMDE3LTExLTM"
10         }
11     }
12 }
```

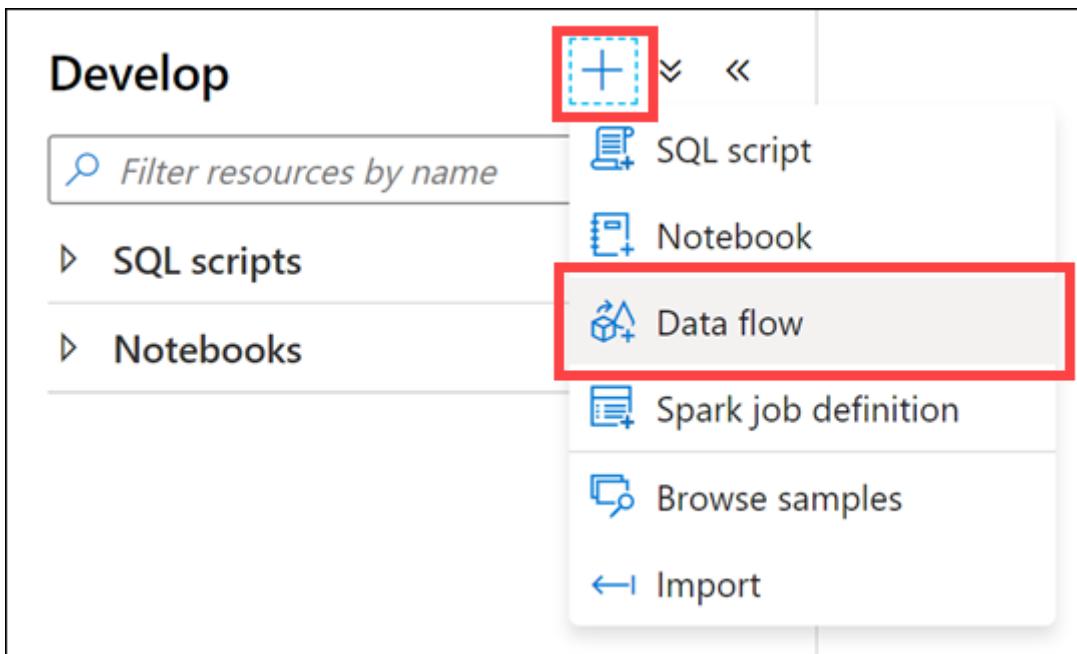
Apply Cancel

11.4.2 Task 2: Create mapping data flow

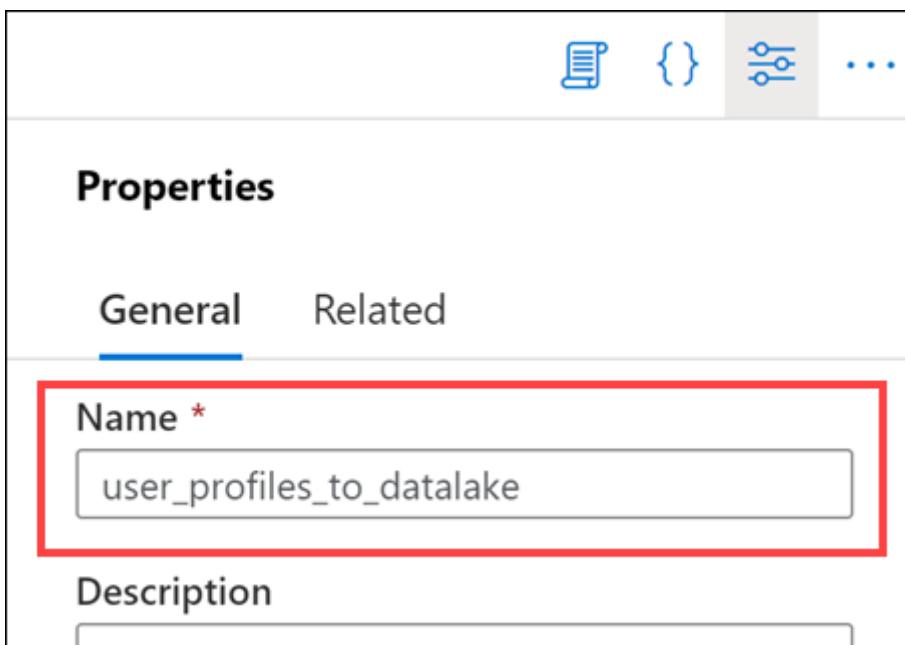
1. Navigate to the **Develop** hub.



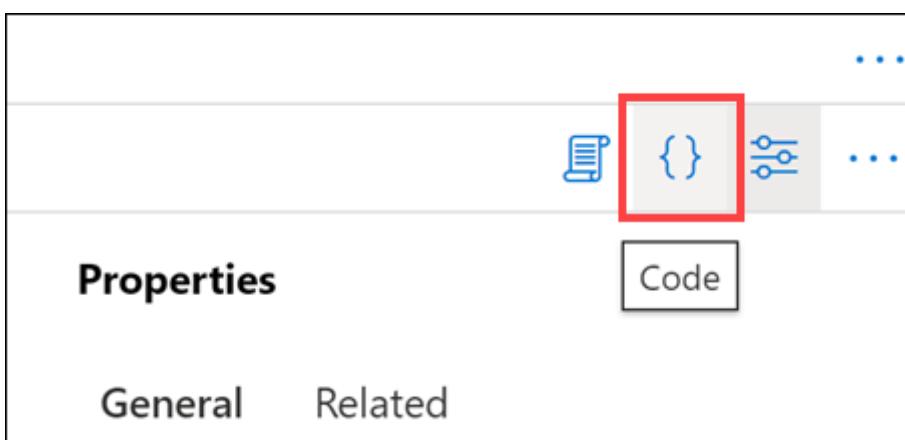
2. Select + then **Data flow** to create a new data flow.



3. In the **General** settings of the **Properties** blade of the new data flow, update the **Name** to the following: `user_profiles_to_datalake`. Make sure the name exactly matches. Otherwise, you will receive an error when you close the code view in a few steps.



4. Select the {} **Code** button at the top-right above the data flow properties.



5. Replace the existing code with the following:

```
{
  "name": "user_profiles_to_datalake",
  "properties": {
    "type": "MappingDataFlow",
    "typeProperties": {
      "sources": [
        {
          "dataset": {
            "referenceName": "asal400_ecommerce_userprofiles_source",
            "type": "DatasetReference"
          },
          "name": "EcommerceUserProfiles"
        },
        {
          "dataset": {
            "referenceName": "asal400_customerprofile_cosmosdb",
            "type": "DatasetReference"
          },
          "name": "UserProfiles"
        }
      ],
      "sinks": [
        {
          "linkedService": {
            "referenceName": "INSERT_YOUR_DATALAKE_SERVICE_NAME",
            "type": "LinkedServiceReference"
          },
          "name": "DataLake"
        }
      ],
      "transformations": [
        {
          "name": "userId"
        },
        {
          "name": "UserTopProducts"
        },
        {
          "name": "DerivedProductColumns"
        },
        {
          "name": "UserPreferredProducts"
        },
        {
          "name": "JoinTopProductsWithPreferredProducts"
        },
        {
          "name": "DerivedColumnsForMerge"
        },
        {
          "name": "Filter1"
        }
      ],
      "script": "source(output(\n\t\tvisitorId as string,\n\t\ttopProductPurchases as (produ
    }
  }
}
```

6. Replace **INSERT_YOUR_DATALAKE_SERVICE_NAME** on line 25 with the name of your ADLS Gen2 linked service that you copied in the previous task (Task 1) above.

```

18
19           },
20           "name": "UserProfiles"
21       }
22   ],
23   "sinks": [
24     {
25       "linkedService": {
26         "referenceName": "INSERT_YOUR_DATALAKE_SERVICE_NAME",
27         "type": "LinkedServiceReference"
28       },
29       "name": "DataLake"
30     }
31   ],
32   "transformations": [

```

The value should now include the name of your linked service:

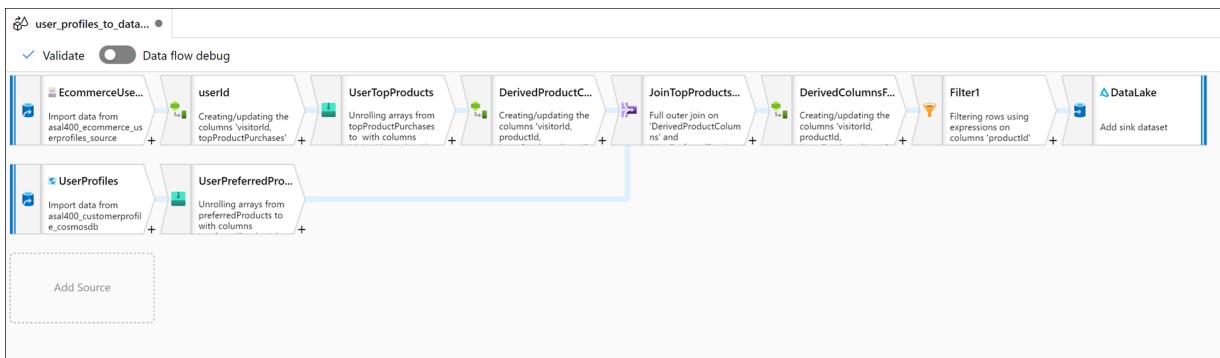
```

18
19           },
20           "name": "UserProfiles"
21       }
22   ],
23   "sinks": [
24     {
25       "linkedService": {
26         "referenceName": "asadatalake      ",
27         "type": "LinkedServiceReference"
28       },
29       "name": "DataLake"
30     }
31   ],
32   "transformations": [

```

7. Select **OK**.

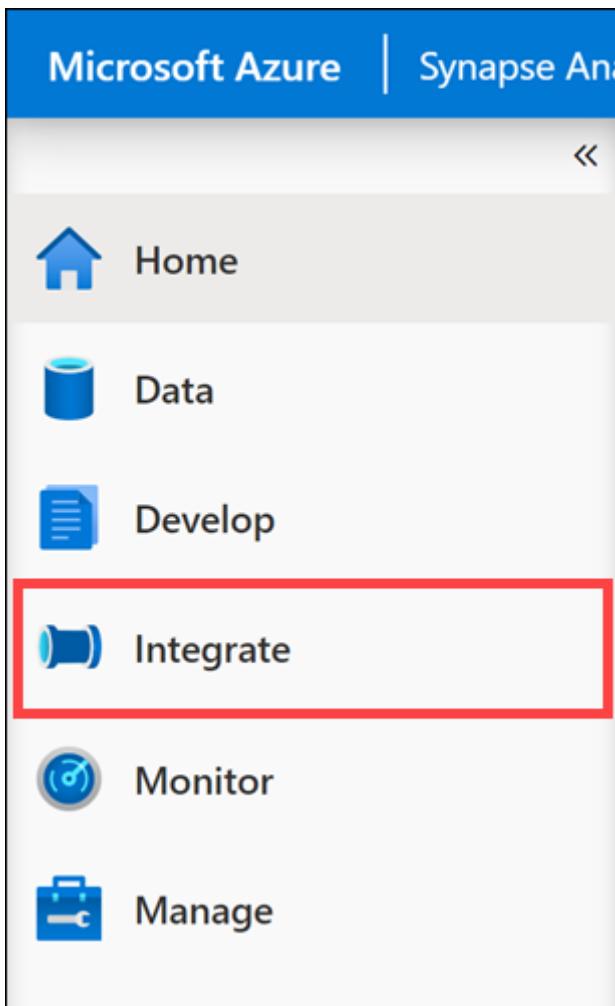
8. The data flow should look like the following:



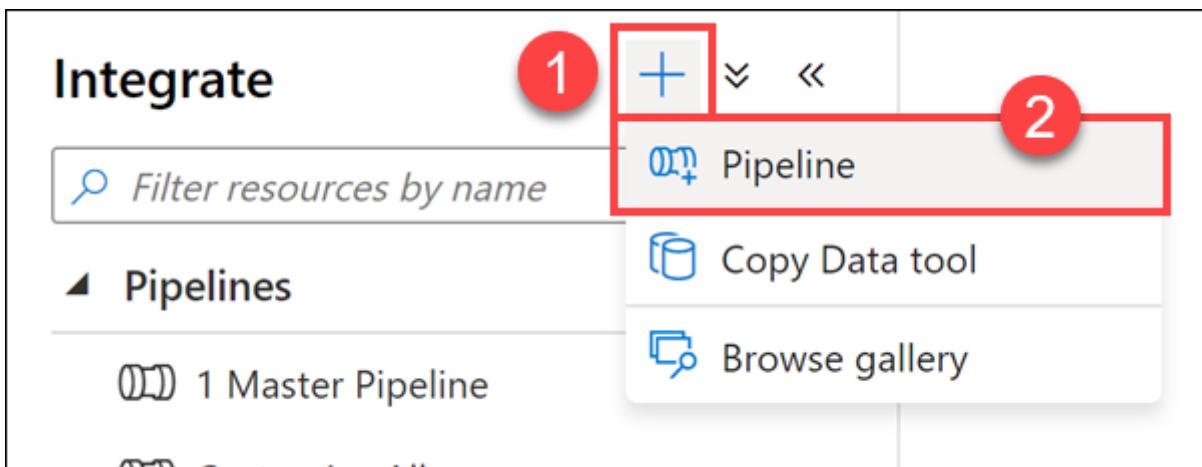
11.4.3 Task 3: Create pipeline

In this step, you create a new integration pipeline to execute the data flow.

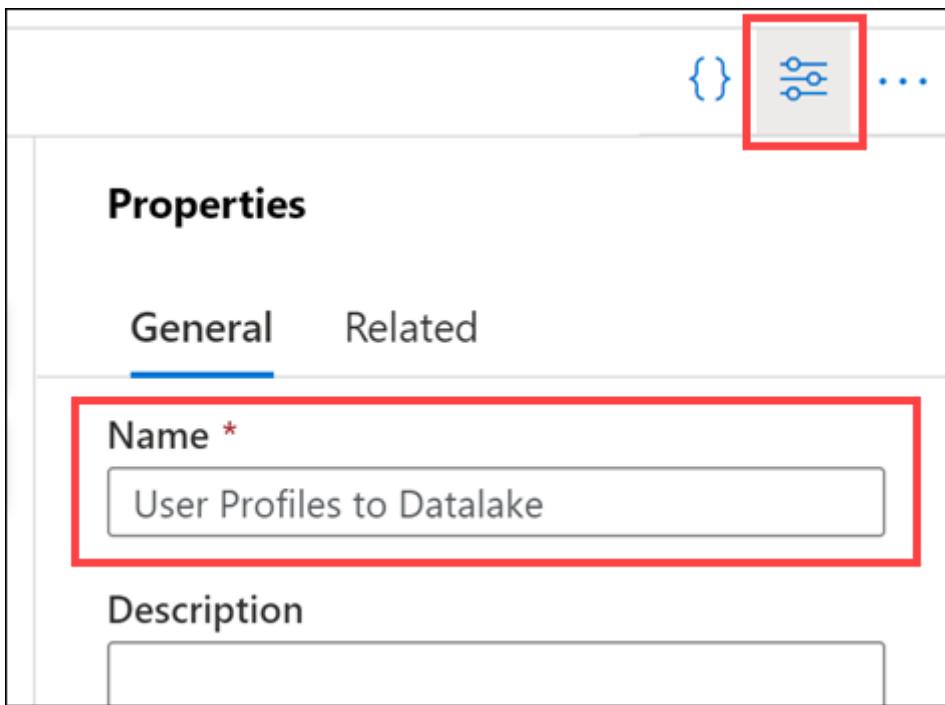
1. Navigate to the **Integrate** hub.



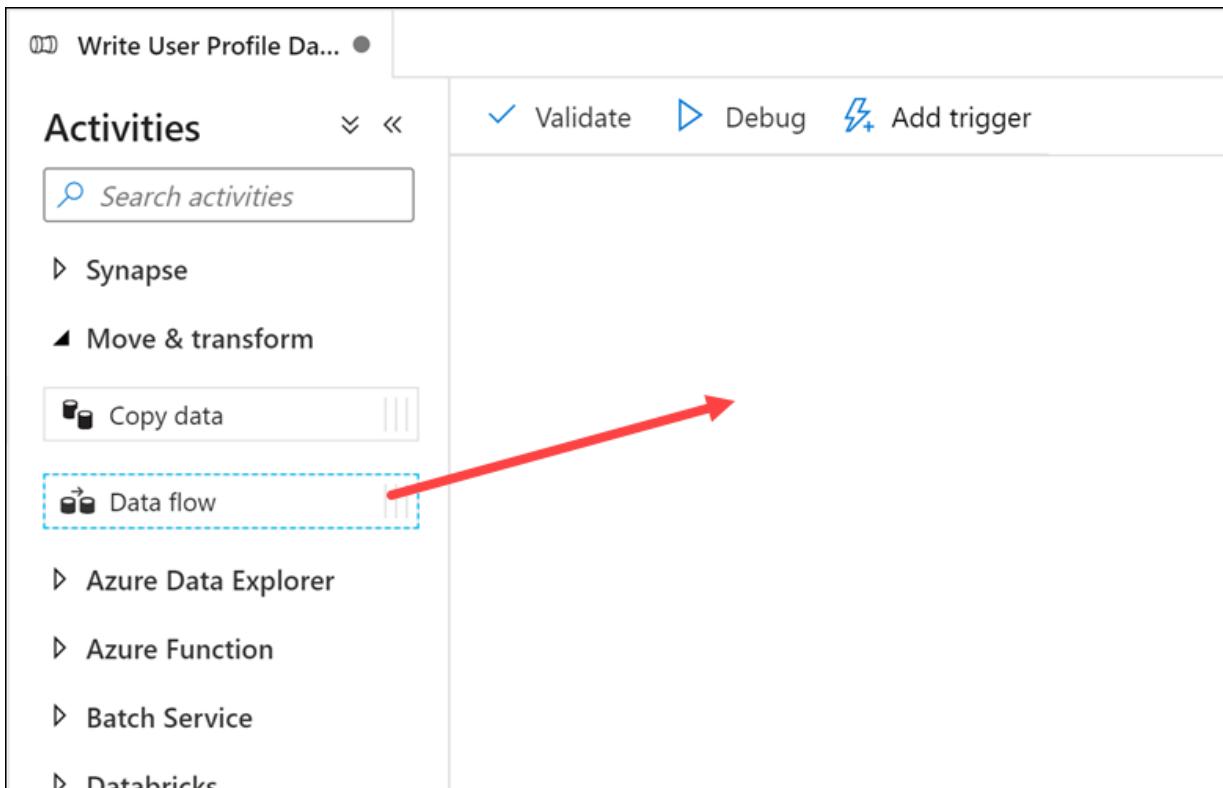
2. Select + (1), then Pipeline (2).



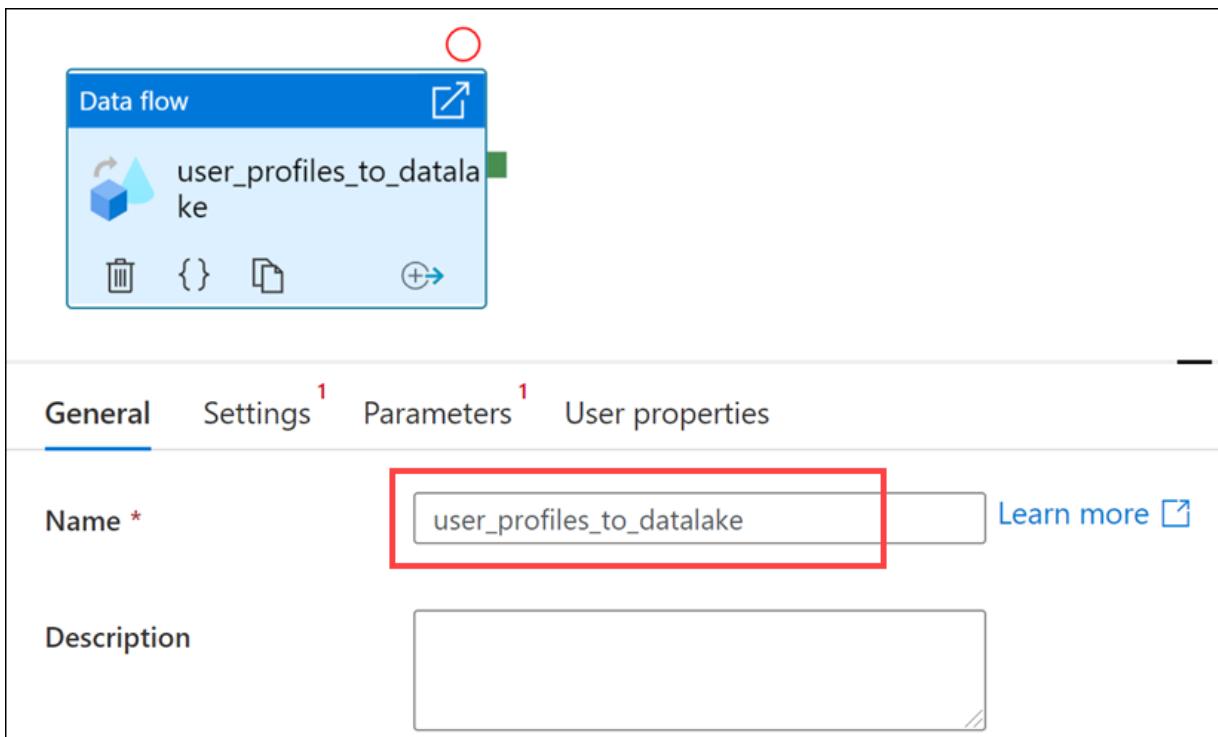
3. In the **General** section of the **Profiles** pane of the new data flow, update the **Name** to the following: User Profiles to Datalake. Select the **Properties** button to hide the pane.



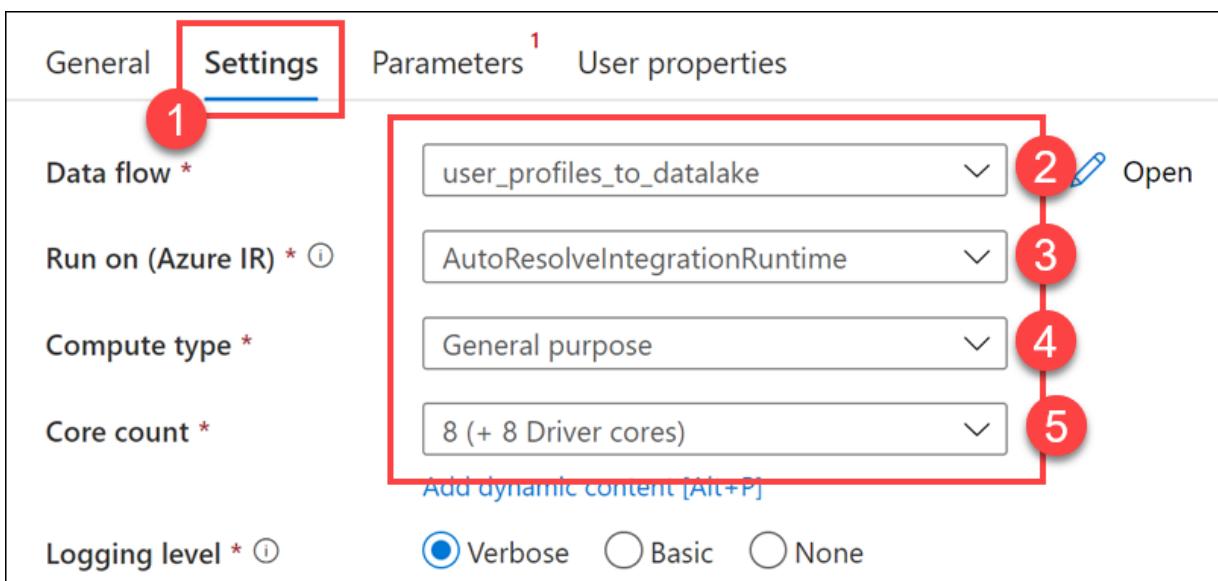
4. Expand **Move & transform** within the Activities list, then drag the **Data flow** activity onto the pipeline canvas.



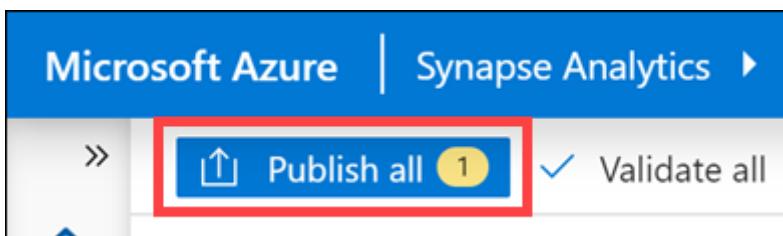
5. Under the **General** tab, set the Name to `user_profiles_to_datalake`.



6. Select the **Settings** tab (1). Select `user_profiles_to_datalake` for **Data flow** (2), then ensure `AutoResolveIntegrationRuntime` is selected for **Run on (Azure IR)** (3). Choose the **General purpose Compute type** (4) and select `8 (+ 8 cores)` for the **Core count** (5).

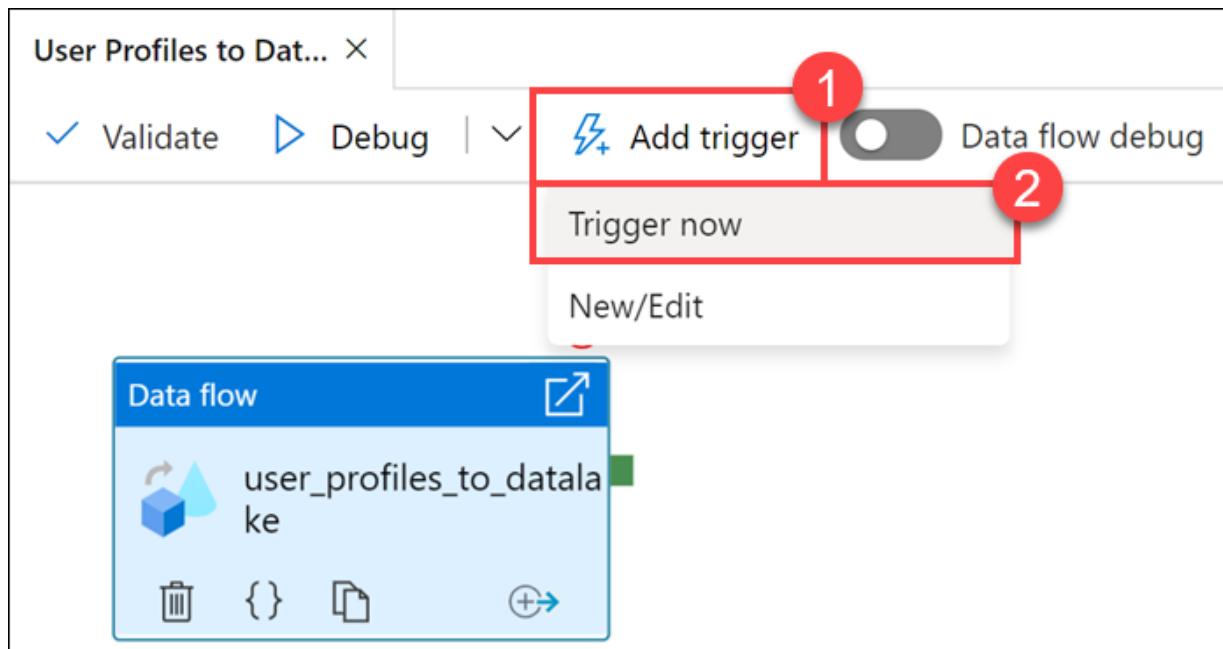


7. Select **Publish all** then **Publish** to save your pipeline.

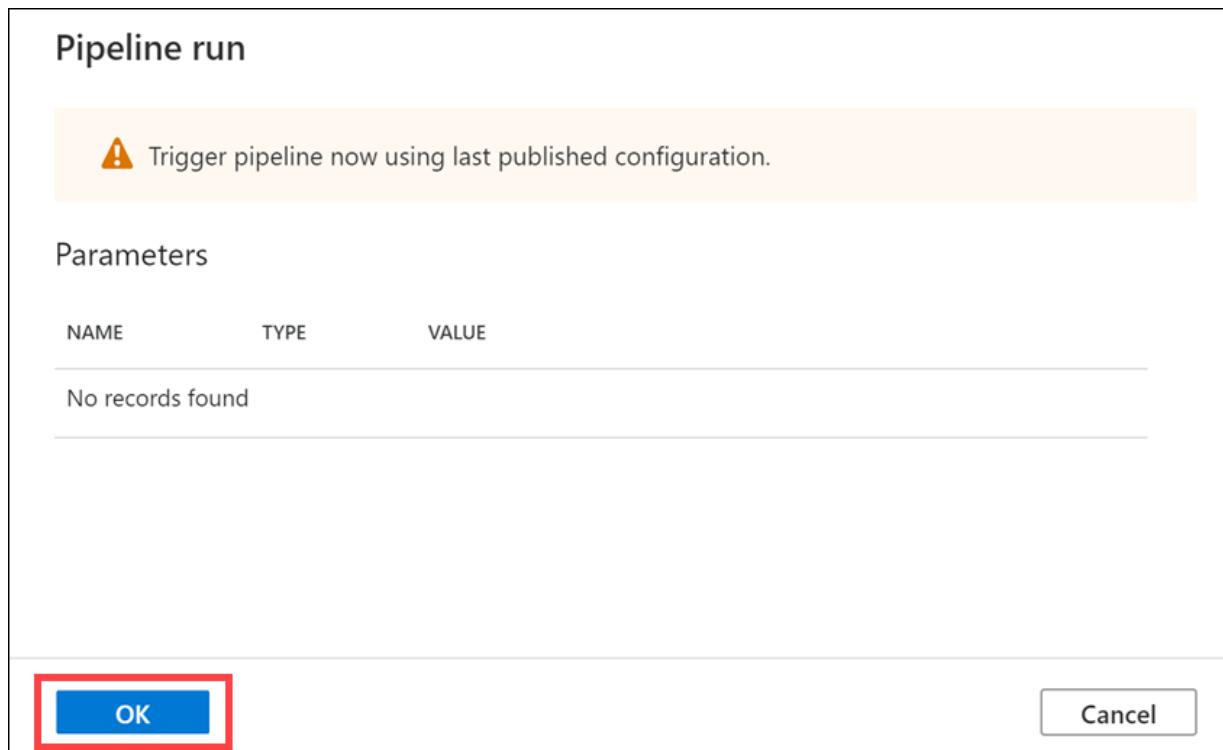


11.4.4 Task 4: Trigger the pipeline

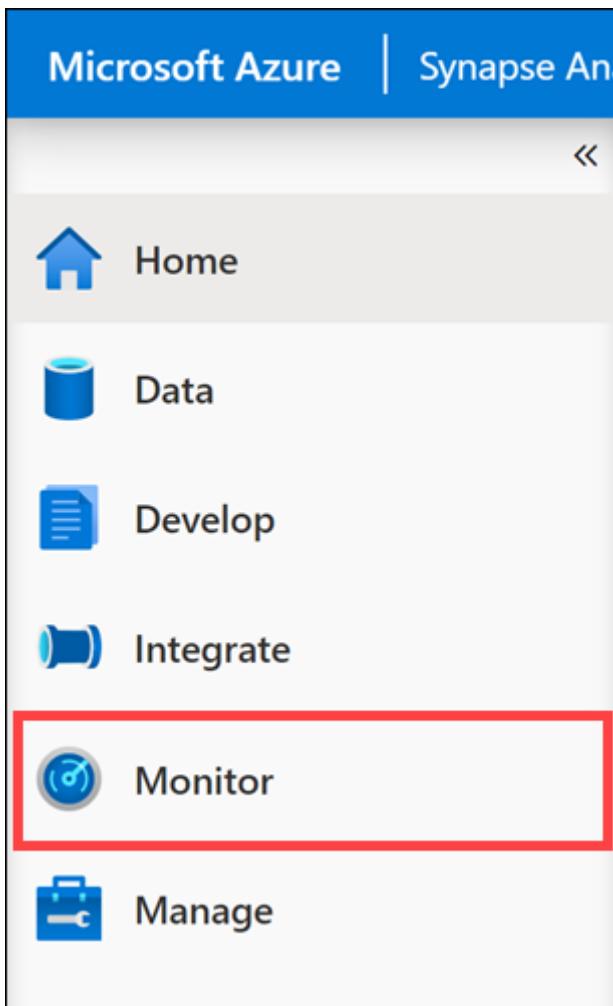
- At the top of the pipeline, select **Add trigger** (1), then **Trigger now** (2).



2. There are no parameters for this pipeline, so select **OK** to run the trigger.



3. Navigate to the **Monitor** hub.



- Select **Pipeline runs** (1) and wait for the pipeline run to successfully complete (2). You may need to refresh (3) the view.

While this is running, read the rest of the lab instructions to familiarize yourself with the content.

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
User Profiles to Datalake	5/2/21, 1:52:35 PM	5/2/21, 1:57:15 PM	00:04:39	Manual trigger	✓ Succeeded	Original

11.5 Exercise 3: Create Synapse Spark notebook to find top products

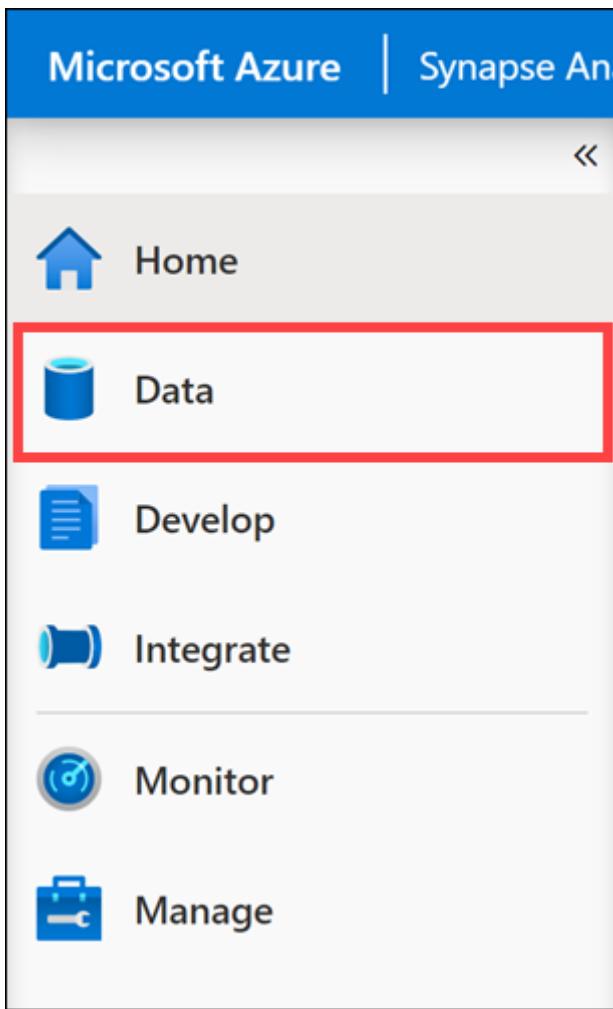
Tailwind Traders uses a Mapping Data flow in Synapse Analytics to process, join, and import user profile data. Now they want to find the top 5 products for each user, based on which ones are both preferred and top, and have the most purchases in the past 12 months. Then, they want to calculate the top 5 products overall.

In this segment of the lab, you will create a Synapse Spark notebook to make these calculations.

We will access the data from the data lake that was added as a second sink in the data flow, removing the dedicated SQL pool dependency.

11.5.1 Task 1: Create notebook

- Open Synapse Analytics Studio (<https://web.azuresynapse.net/>), and then navigate to the **Data** hub.



2. Select the **Linked** tab (1) and expand the **primary data lake storage account** (2) underneath the **Azure Data Lake Storage Gen2**. Select the **wwi-02** container (3) and open the **top-products** folder (4). Right-click on any Parquet file (5), select the **New notebook** menu item (6), then select **Load to DataFrame** (7). If you don't see the folder, select **Refresh** above.

Name	Last Modified	Content Type
_delta_log	11/26/2020, 12:18:24 AM	Folder
part-00000-80db06cf-0892-41c7-9c21-44bae30ec46c-c000.snappy.parquet	11/26/2020, 12:18:24 AM	New SQL script
part-00003-7c06fb2c-c86e-4cdd-bf57-d697ebb6c712-c000.snappy.parquet	11/26/2020, 12:18:24 AM	New notebook
part-00011-b3d96b81-8369-4231-a5d1-08a93dfe1653-c000.snappy.parquet	11/26/2020, 12:18:24 AM	Load to DataFrame
part-00012-68f66638-1619-4e8d-b941-69bba4d878b5-c000.snappy.parquet	11/26/2020, 12:18:24 AM	New data flow
part-00017-61a9807f-d1c0-4163-b9f6-3d51d69b1c46-c000.snappy.parquet	11/26/2020, 12:18:24 AM	New integration dataset
part-00020-cf5b45e6-d864-4e28-9a59-f15ff5cdb7b6-c000.snappy.parquet	11/26/2020, 12:18:24 AM	Manage access...
part-00023-674b18e2-3ae0-42df-bdb6-13281d9dd9e2-c000.snappy.parquet	11/26/2020, 12:18:24 AM	Rename...
part-00028-6afb8a30-a957-4a07-b8b0-c341ed3f834c-c000.snappy.parquet	11/26/2020, 12:18:24 AM	Download
part-00029-657c43be-b834-440d-8a96-5e7ead097f78-c000.snappy.parquet	11/26/2020, 12:18:24 AM	Delete
		Properties...

3. Make sure the notebook is attached to your Spark pool.

```

Cell 1
1 %%pyspark
2 df = spark.read.load('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top-products/part-00000-80d
3 display(df.limit(10))

```

4. Replace the Parquet file name with ***.parquet (1)** to select all Parquet files in the top-products folder. For example, the path should be similar to: abfss://wwi-02@YOUR_DATALAKE_NAME.dfs.core.windows.net/top-prod

```

Cell 1
1 %%pyspark
2 df = spark.read.load('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top-products/*.parquet', format='parquet')
3 display(df.limit(10))

```

5. Select **Run all** on the notebook toolbar to execute the notebook.

visitorId	productId	itemsPurchasedLast12Months	preferredProductId	userId	isTopProduct	isPreferredProduct
undefined	2717	undefined	2717	148	false	true
undefined	4002	undefined	4002	148	false	true
undefined	1716	undefined	1716	148	false	true
undefined	4520	undefined	4520	148	false	true
undefined	951	undefined	951	148	false	true
undefined	1817	undefined	1817	148	false	true
undefined	2634	undefined	2634	463	false	true

Note: The first time you run a notebook in a Spark pool, Synapse creates a new session. This can take approximately 3-5 minutes.

Note: To run just the cell, either hover over the cell and select the *Run cell* icon to the left of the cell, or select the cell then type **Ctrl+Enter** on your keyboard.

6. Create a new cell underneath by selecting the + button and selecting the </> **Code cell** item. The + button is located beneath the notebook cell on the left.

7. Enter and execute the following in the new cell to populate a new dataframe called `topPurchases`, create a new temporary view named `top_purchases`, and show the first 100 rows:

```

topPurchases = df.select(
    "UserId", "ProductId",
    "ItemsPurchasedLast12Months", "IsTopProduct",
    "IsPreferredProduct")

# Populate a temporary view so we can query from SQL
topPurchases.createOrReplaceTempView("top_purchases")

```

`topPurchases.show(100)`

The output should look similar to the following:

UserId	ProductId	ItemsPurchasedLast12Months	IsTopProduct	IsPreferredProduct
148	2717	null	false	true
148	4002	null	false	true
148	1716	null	false	true
148	4520	null	false	true
148	951	null	false	true
148	1817	null	false	true
463	2634	null	false	true
463	2795	null	false	true
471	1946	null	false	true
471	4431	null	false	true
471	566	null	false	true
471	2179	null	false	true
471	3758	null	false	true
471	2434	null	false	true
471	1793	null	false	true
471	1620	null	false	true
471	1572	null	false	true
833	957	null	false	true
833	3140	null	false	true
833	1087	null	false	true

8. Execute the following in a new cell to create a new DataFrame to hold only top preferred products where both `IsTopProduct` and `IsPreferredProduct` are true:

```

from pyspark.sql.functions import *

topPreferredProducts = (topPurchases
    .filter( col("IsTopProduct") == True)
    .filter( col("IsPreferredProduct") == True)
    .orderBy( col("ItemsPurchasedLast12Months").desc() ))

```

`topPreferredProducts.show(100)`

Cell 3

```
1  from pyspark.sql.functions import *
2
3  topPreferredProducts = (topPurchases
4      .filter( col("IsTopProduct") == True)
5      .filter( col("IsPreferredProduct") == True)
6      .orderBy( col("ItemsPurchasedLast12Months").desc() ))
7
8  topPreferredProducts.show(100)
```

UserId	ProductId	ItemsPurchasedLast12Months	IsTopProduct	IsPreferredProduct
90779	4086	99	true	true
99537	3865	99	true	true
90779	4086	99	true	true
85007	521	99	true	true
90779	4086	99	true	true
89537	1473	99	true	true
96220	677	99	true	true
89537	1473	99	true	true
96220	677	99	true	true
89537	1473	99	true	true
96220	677	99	true	true
90804	1709	99	true	true
96220	677	99	true	true

9. Execute the following in a new cell to create a new temporary view by using SQL:

```
%sql
```

```
CREATE OR REPLACE TEMPORARY VIEW top_5_products
AS
    select UserId, ProductId, ItemsPurchasedLast12Months
    from (select *,
        row_number() over (partition by UserId order by ItemsPurchasedLast12Months desc) as seqnum
    from top_purchases
    ) a
    where seqnum <= 5 and IsTopProduct == true and IsPreferredProduct = true
    order by a.UserId
```

Note that there is no output for the above query. The query uses the `top_purchases` temporary view as a source and applies a `row_number()` over method to apply a row number for the records for each user where `ItemsPurchasedLast12Months` is greatest. The `where` clause filters the results so we only retrieve up to five products where both `IsTopProduct` and `IsPreferredProduct` are set to true. This gives us the top five most purchased products for each user where those products are also identified as their favorite products, according to their user profile stored in Azure Cosmos DB.

10. Execute the following in a new cell to create and display a new DataFrame that stores the results of the `top_5_products` temporary view you created in the previous cell:

```
top5Products = sqlContext.table("top_5_products")
```

```
top5Products.show(100)
```

You should see an output similar to the following, which displays the top five preferred products per user:

Cell 5

```
[26] 1 top5Products = sqlContext.table("top_5_products")
      2
      3 top5Products.show(100)
```

UserId	ProductId	ItemsPurchasedLast12Months
80000	2069	93
80000	2069	93
80000	2069	93
80000	2069	93
80000	2069	93
80001	1812	93
80001	1812	93
80001	1812	93
80001	1812	93
80001	1812	93
80002	1256	90
80002	1256	90
80002	4987	88
80002	3190	92
80002	3190	92
80003	295	91
80003	638	97
80003	620	97

11. Execute the following in a new cell to compare the number of top preferred products to the top five preferred products per customer:

```
print('before filter: ', topPreferredProducts.count(), ', after filter: ', top5Products.count())
```

The output should be similar to before filter: 997873 , after filter: 85020.

12. Calculate the top five products overall, based on those that are both preferred by customers and purchased the most. To do this, execute the following in a new cell:

```
top5ProductsOverall = (top5Products.select("ProductId", "ItemsPurchasedLast12Months")
    .groupBy("ProductId")
    .agg( sum("ItemsPurchasedLast12Months").alias("Total") )
    .orderBy( col("Total").desc() )
    .limit(5))

top5ProductsOverall.show()
```

In this cell, we grouped the top five preferred products by product ID, summed up the total items purchased in the last 12 months, sorted that value in descending order, and returned the top five results. Your output should be similar to the following:

ProductId	Total
347	4523
4833	4314
3459	4233
2486	4135
2107	4113

```
+-----+-----+
```

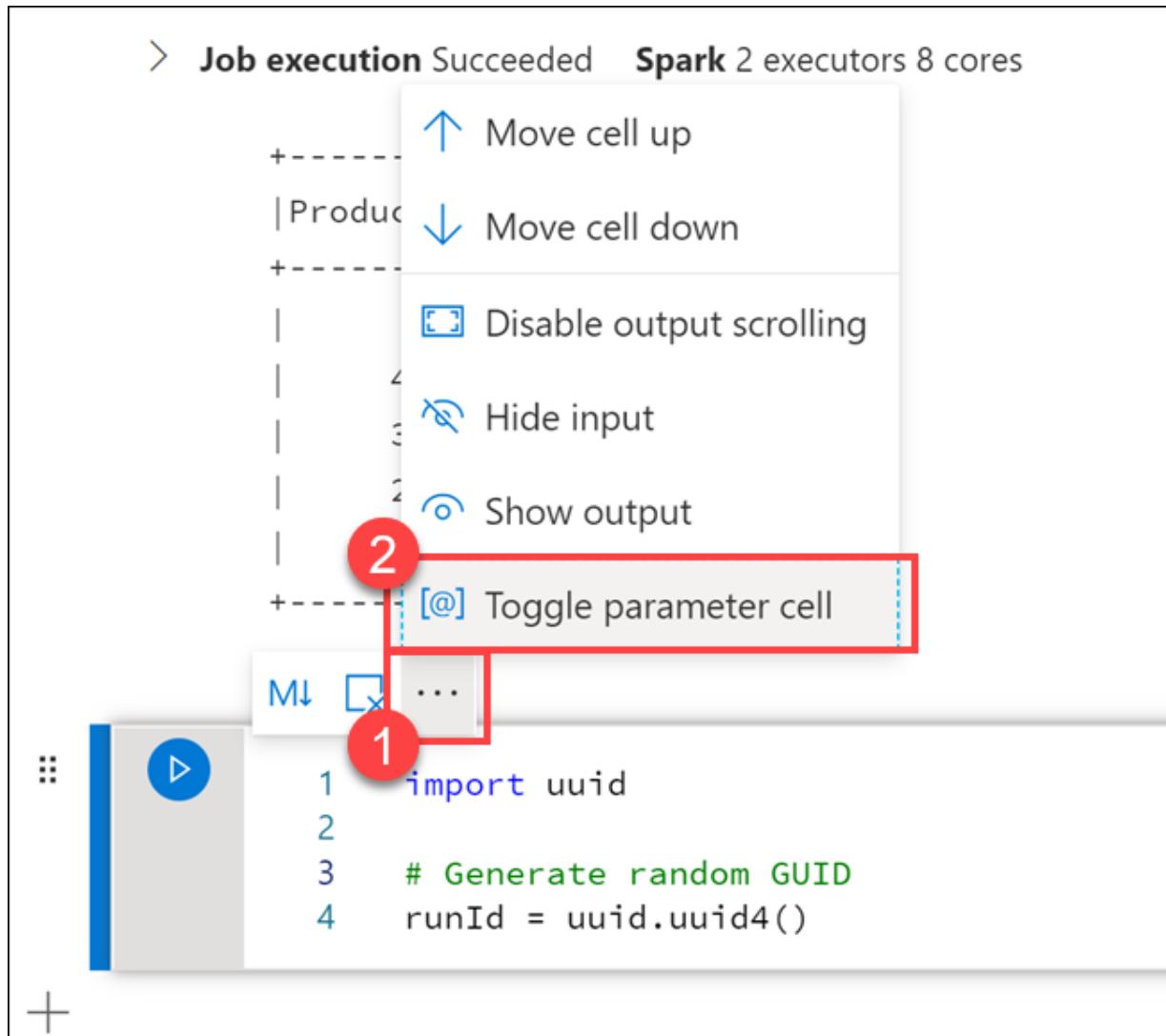
13. We are going to execute this notebook from a pipeline. We want to pass in a parameter that sets a `runId` variable value that will be used to name the Parquet file. Execute the following in a new cell:

```
import uuid

# Generate random GUID
runId = uuid.uuid4()
```

We are using the `uuid` library that comes with Spark to generate a random GUID. We want to override the `runId` variable with a parameter passed in by the pipeline. To do this, we need to toggle this as a parameter cell.

14. Select the actions ellipses (...) above the cell (1), then select **Toggle parameter cell** (2).



After toggling this option, you will see the **Parameters** tag on the cell.



15. Paste the following code in a new cell to use the `runId` variable as the Parquet filename in the `/top5-products/` path in the primary data lake account. Replace `YOUR_DATALAKE_NAME` in the path with the name of your primary data lake account. To find this, scroll up to **Cell 1** at the top of the

page (1). Copy the data lake storage account from the path (2). Paste this value as a replacement for **YOUR_DATALAKE_NAME** in the path (3) inside the new cell, then execute the cell.

```
%%pyspark
```

```
top5ProductsOverall.write.parquet('abfss://wwi-02@YOUR_DATALAKE_NAME.dfs.core.windows.net/top5-prod...
```

The screenshot shows a Jupyter Notebook interface with three cells:

- Cell 1:** Contains Python code using the %%pyspark magic command to read a Parquet file from a specific path: `abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top-products/*.parquet`.
- Cell 2:** Contains Python code using the `uuid` module to generate a random GUID.
- Cell 9:** Contains the same Parquet write code as Cell 1, but with the path modified to use the generated GUID from Cell 2: `abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top5-products/' + str(runId) ...`.

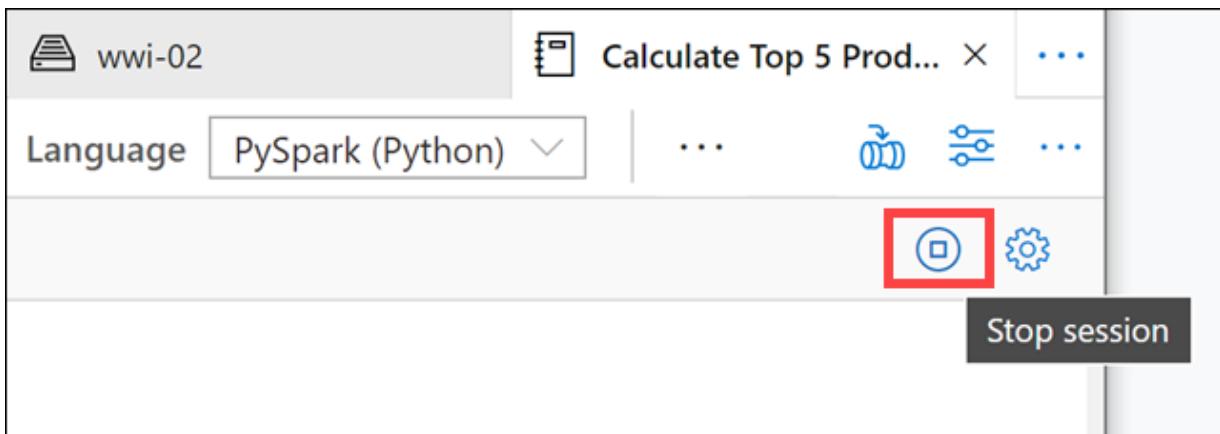
A red arrow points from the highlighted path in Cell 1 to the highlighted path in Cell 9, indicating the replacement of the placeholder with the generated GUID.

16. Verify that the file was written to the data lake. Navigate to the **Data** hub and select the **Linked** tab (1). Expand the primary data lake storage account and select the **wwi-02** container (2). Navigate to the **top5-products** folder (3). You should see a folder for the Parquet file in the directory with a GUID as the file name (4).

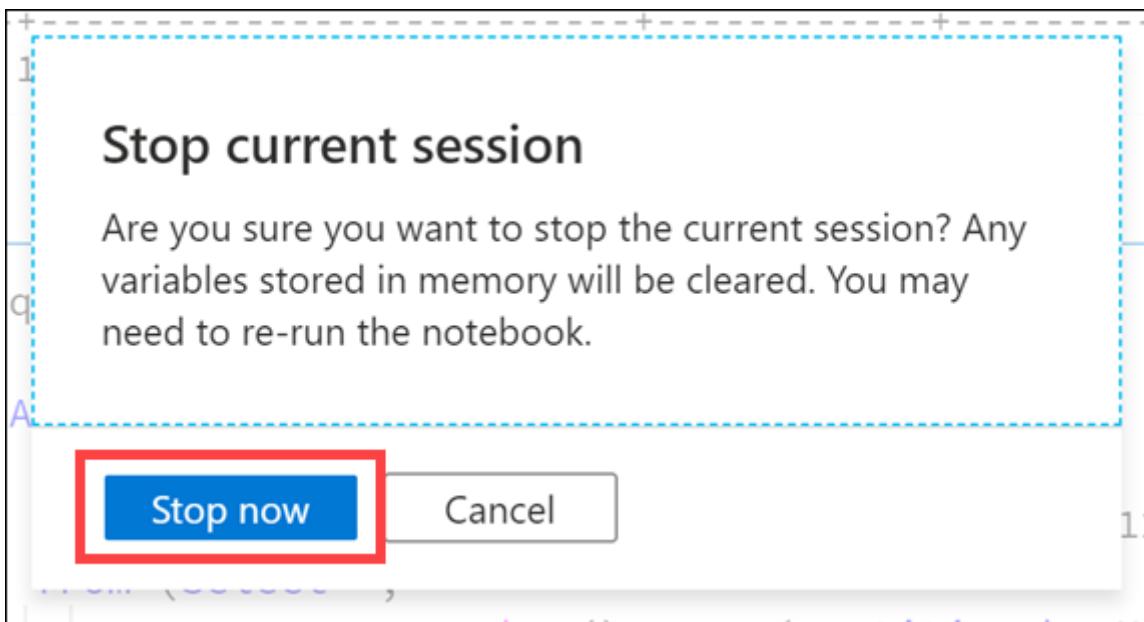
The screenshot shows the Azure Data Explorer 'Data' hub with the 'Linked' tab selected (1). The 'wwi-02' container is expanded (2), showing its contents. The 'top5-products' folder is selected (3), and the Parquet file '12fe8c61-0998-4124-bf3c-fa751d8ce088.parquet' is highlighted (4).

The Parquet write method on the dataframe in the Notebook cell created this directory since it did not previously exist.

17. Return to the notebook. Select **Stop session** on the upper-right of the notebook. We want to stop the session to free up the compute resources for when we run the notebook inside the pipeline in the next section.



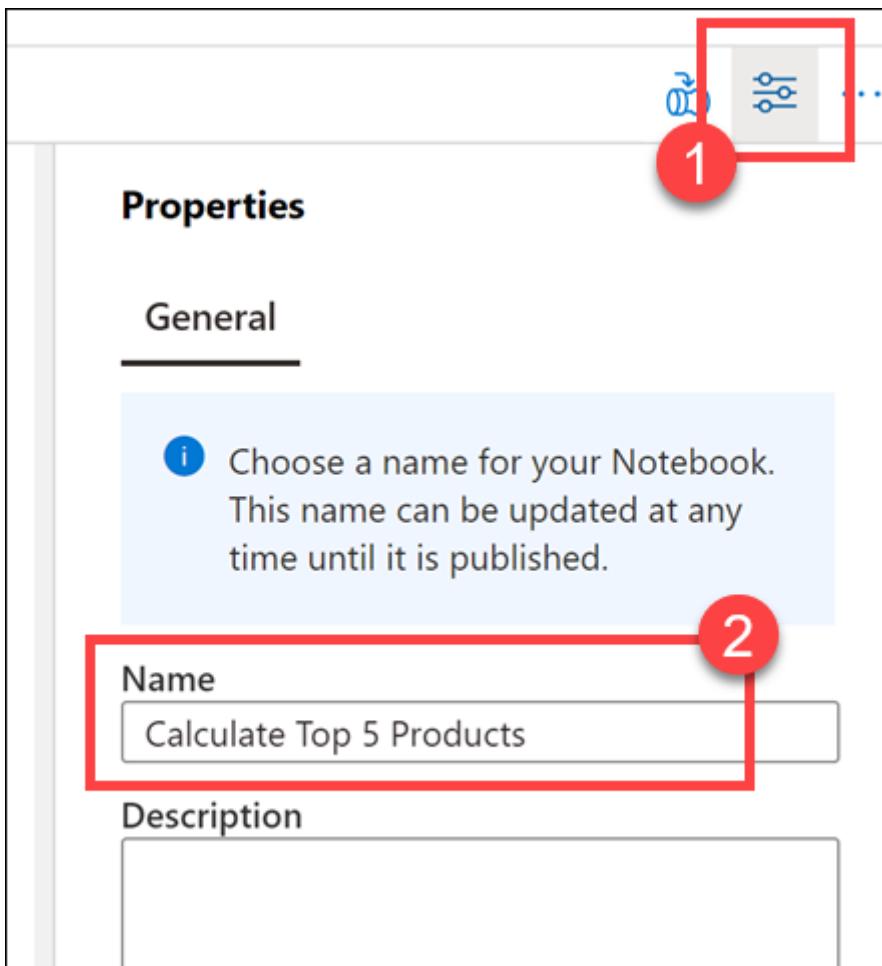
18. Select **Stop now** in the Stop current session.



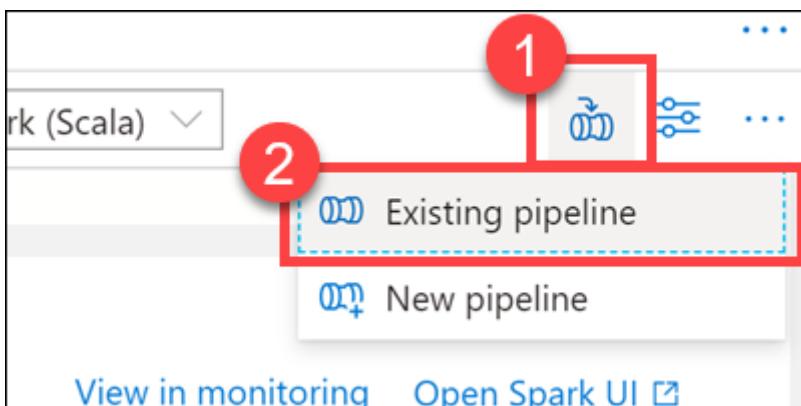
11.5.2 Task 2: Add the Notebook to the pipeline

Tailwind Traders wants to execute this notebook after the Mapping Data Flow runs as part of their orchestration process. To do this, we will add this notebook to our pipeline as a new Notebook activity.

1. Return to the notebook. Select the **Properties** button (1) at the top-right corner of the notebook, then enter Calculate Top 5 Products for the Name (2).



2. Select the **Add to pipeline** button (1) at the top-right corner of the notebook, then select **Existing pipeline** (2).



3. Select the **User Profiles to Datalake** pipeline (1), then select **Add *2**.

Add to pipeline

 Calculate Top 5 Products

Pipelines

Choose a pipeline

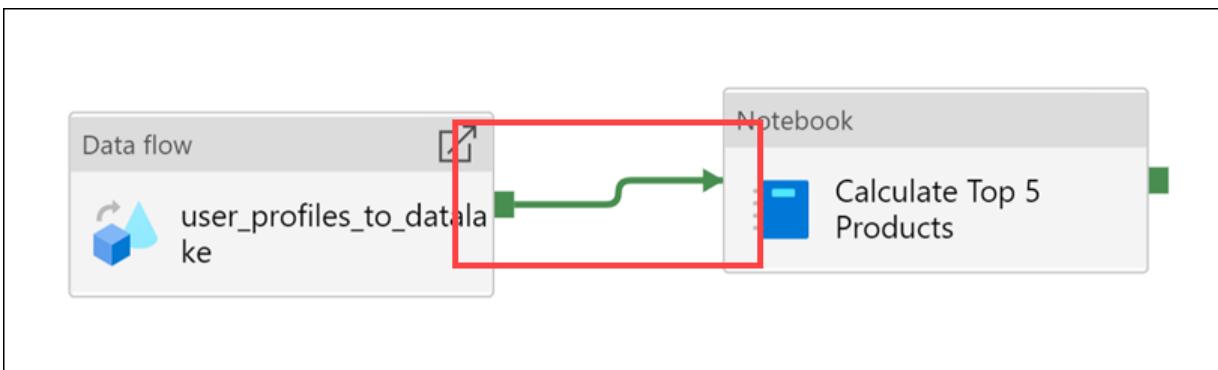
-  Lab 08 - Execute Business Analyst Querie
-  Lab 08 - Execute Data Analyst and CEO C
-  Write User Profile Data to ASA
-  User Profiles to Datalake

1

2

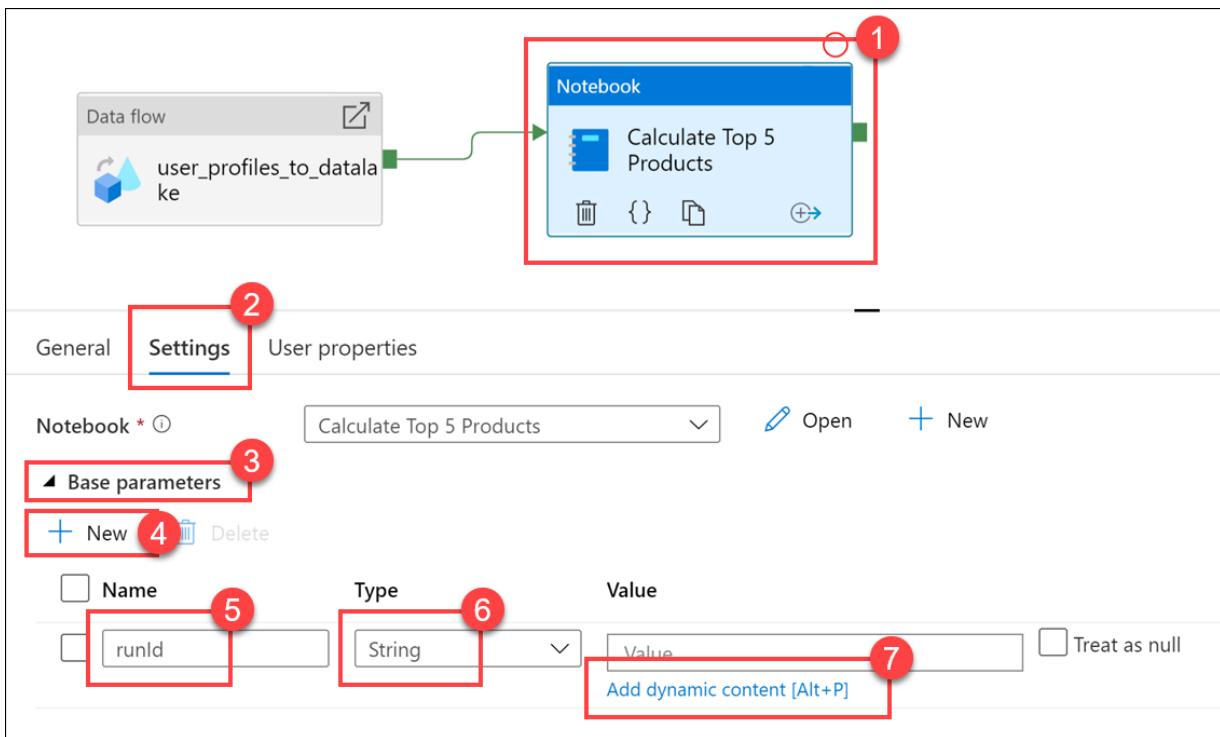
 Add Cancel

4. Synapse Studio adds the Notebook activity to the pipeline. Rearrange the **Notebook activity** so it sits to the right of the **Data flow activity**. Select the **Data flow activity** and drag a **Success** activity pipeline connection **green box** to the **Notebook activity**.



The Success activity arrow instructs the pipeline to execute the Notebook activity after the Data flow activity successfully runs.

5. Select the **Notebook activity** (1), select the **Settings** tab (2), expand **Base parameters** (3), and select **+ New** (4). Enter **runId** in the **Name** field (5). Select **String** for the **Type** (6). For the **Value**, select **Add dynamic content** (7).



6. Select **Pipeline run ID** under **System variables** (1). This adds `@pipeline().RunId` to the dynamic content box (2). Select **Finish** (3) to close the dialog.

Add dynamic content

The screenshot shows the 'Add dynamic content' dialog box. At the top, there is an input field containing the expression '@pipeline().RunId' (circled with a red number 2). Below the input field is a 'Clear contents' link and a 'Filter...' search bar. The main content area contains several system variables listed under sections like 'System variables' and 'Functions'. One variable, 'Pipeline run ID', is highlighted with a red box and circled with a red number 1. The 'Finish' button at the bottom left is also circled with a red box and has a red number 3 above it. A 'Cancel' button is located at the bottom right.

@pipeline().RunId 2

Clear contents

Filter... +

Use [expressions](#), [functions](#) or refer to [system variables](#).

▲ System variables

- Pipeline Name
Name of the pipeline
- Pipeline run ID** 1
ID of the specific pipeline run
- Pipeline trigger ID
ID of the trigger that invokes the pipeline
- Pipeline trigger name
Name of the trigger that invokes the pipeline
- Pipeline trigger time
Time when the trigger that invoked the pipeline. The trigger time is the actual fired time, not the sc...
- Pipeline trigger type
Type of the trigger that invoked the pipeline (Manual, Scheduler)
- Workspace name
Name of the workspace the pipeline run is running within

▲ Functions

- ▼ Expand all
- ▷ Collection Functions

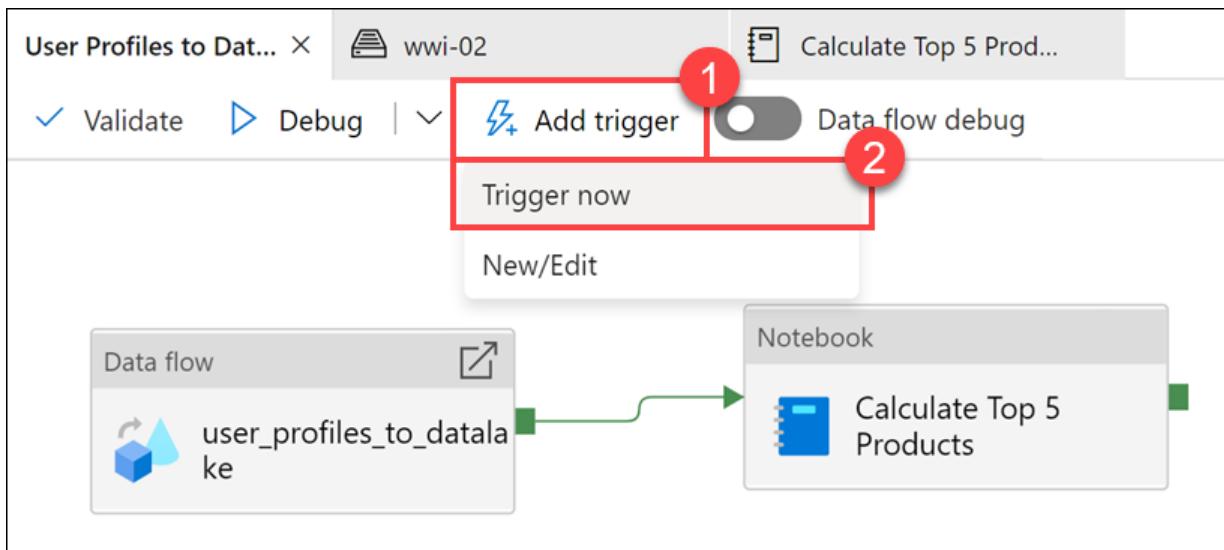
3 Finish Cancel

The Pipeline run ID value is a unique GUID assigned to each pipeline run. We will use this value for the name of the Parquet file by passing this value in as the `runId` Notebook parameter. We can then look through the pipeline run history and find the specific Parquet file created for each pipeline run.

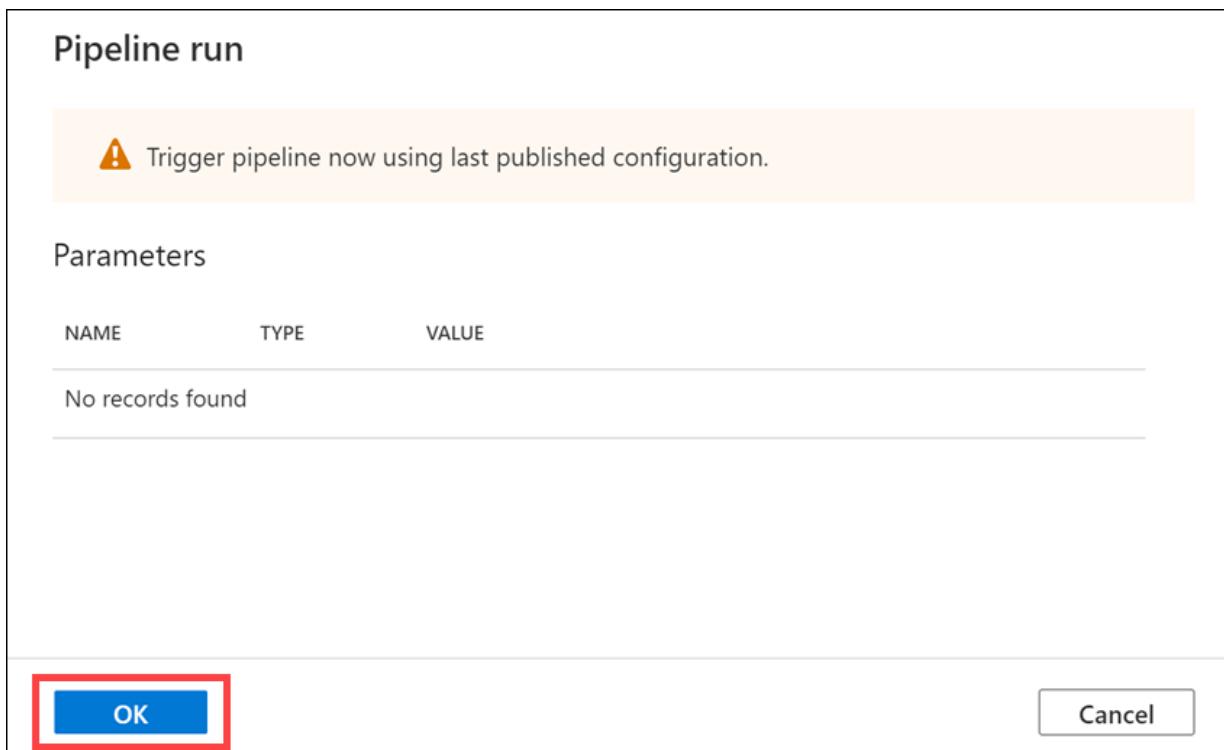
7. Select **Publish all** then **Publish** to save your changes.



8. OPTIONAL - Pipeline run now takes >10 minutes - After publishing is complete, select Add trigger (1), then Trigger now (2) to run the updated pipeline.



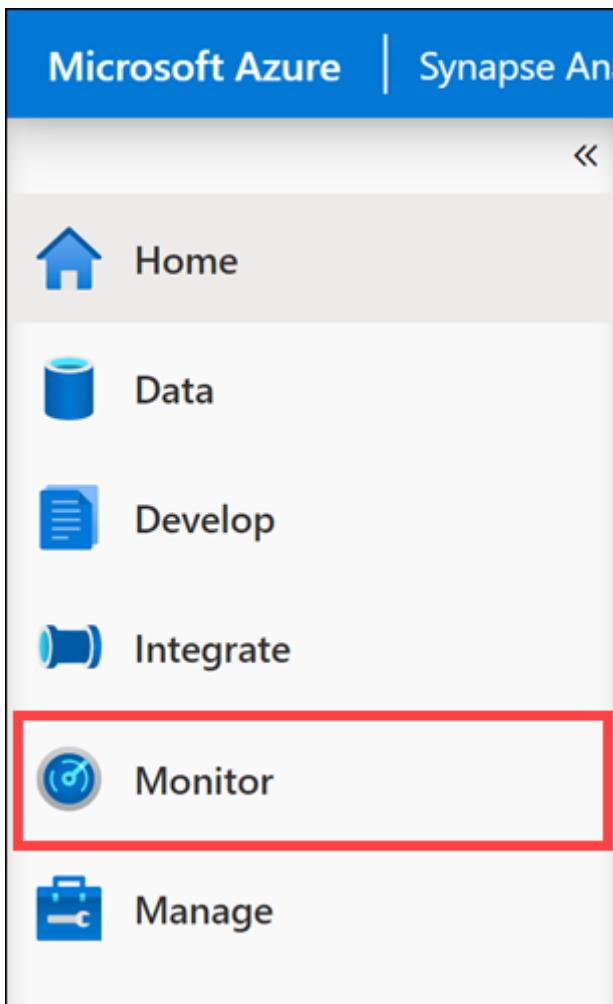
9. Select OK to run the trigger.



OK

Cancel

10. Navigate to the **Monitor** hub.



11. Select **Pipeline runs** (1) and wait for the pipeline run to successfully complete (2). You may need to refresh (3) the view.

Pipeline name	Run start	Run end	Duration	Triggered by	Status
User Profiles to Datalake	5/2/21, 2:32:48 PM	5/2/21, 2:44:27 PM	00:11:38	Manual trigger	Succeeded
User Profiles to Datalake	5/2/21, 1:52:35 PM	5/2/21, 1:57:15 PM	00:04:39	Manual trigger	Succeeded

It can take over 10 minutes for the run to complete with the addition of the notebook activity.
While this is running, read the rest of the lab instructions to familiarize yourself with the content.

12. Select the name of the pipeline to view the pipeline's activity runs.

Pipeline runs

Triggered Debug | Rerun Cancel

Search by run ID or name Local time

Add filter

Showing 1 - 5 items

Pipeline name
 User Profiles to Datalake

13. This time, we see both the **Data flow** activity, and the new **Notebook** activity (1). Make note of the **Pipeline run ID** value (2). We will compare this to the Parquet file name generated by the notebook. Select the **Calculate Top 5 Products** Notebook name to view its details (3).

User Profiles to Datalake

List Gantt

Rerun Rerun from activity Rerun from failed activity Refresh Edit pipeline

Activity runs

Pipeline run ID 06acd59f-e3f5-4c0f-9d6a-f5531cbb2ded

All status ▾

Showing 1 - 2 of 2 items

Activity name	Activity type	Run start ↑	Duration	Status	Integration
Calculate Top 5 Products	Notebook	5/2/21, 2:37:59 PM	00:06:27	✓ Succeeded	DefaultInteg
user_profiles_to_datalake	Data flow	5/2/21, 2:32:51 PM	00:05:07	✓ Succeeded	DefaultInteg

14. Here we see the Notebook run details. You can select the **Playback** button (1) to watch a playback of the

progress through the **jobs** (2). At the bottom, you can view the **Diagnostics** and **Logs** with different filter options (3). Hover over a stage to view its details, such as the duration, total tasks, data details, etc. Select the **View details** link on the **stage** to view its details (5).

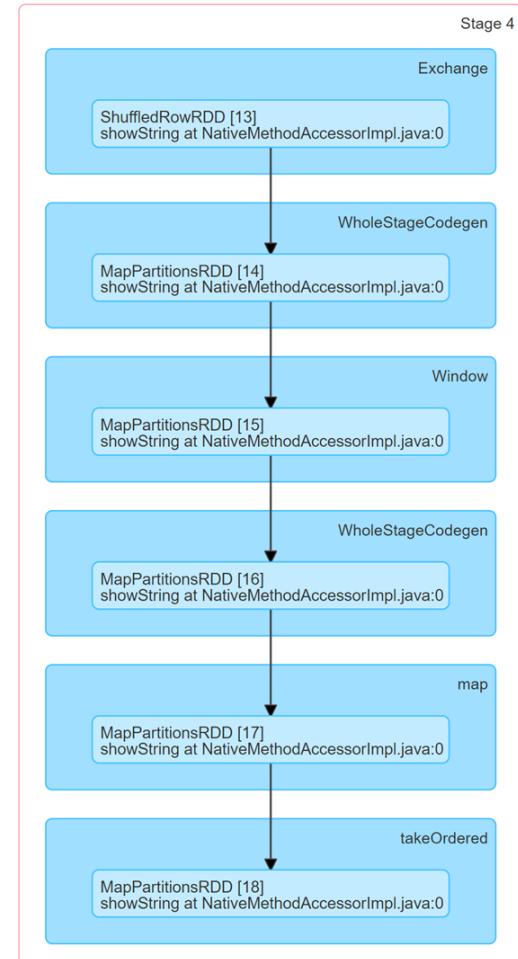
The screenshot shows the Spark Application UI for job ID 3bf888e7-0b00-49f2-b3ed-2b165736673b. The top navigation bar includes Refresh, Compare applications, and Spark history server. Below it, the job summary shows 2562 completed tasks, a stopped status, and a total duration of 6m 24s. A playback slider is set at 0 ms / 2 min 19 sec 143 ms. The main area displays the DAG visualization with stages 0 through 6. Stage 4 is expanded, showing its details: 200 tasks, 3 sec 504 ms duration, 1,622,203 rows, 10.1 MB read, and 0 bytes written. The 'Diagnostics' section lists Failed jobs, Data skew, Time skew, and Executor utilization. Stage 4's 'View details' link is highlighted with a red box and number 5. Stage 4 is also labeled with a green checkmark and 'Stage 4' (4). Stage 4's expanded details are also labeled with a red box and number 4.

15. The Spark application UI opens in a new tab where we can see the stage details. Expand the **DAG Visualization** to view the stage details.

Details for Stage 4 (Attempt 0)

Total Time Across All Tasks: 18 s
Locality Level Summary: Node local: 200
Shuffle Read: 10.1 MB / 1622203

DAG Visualization

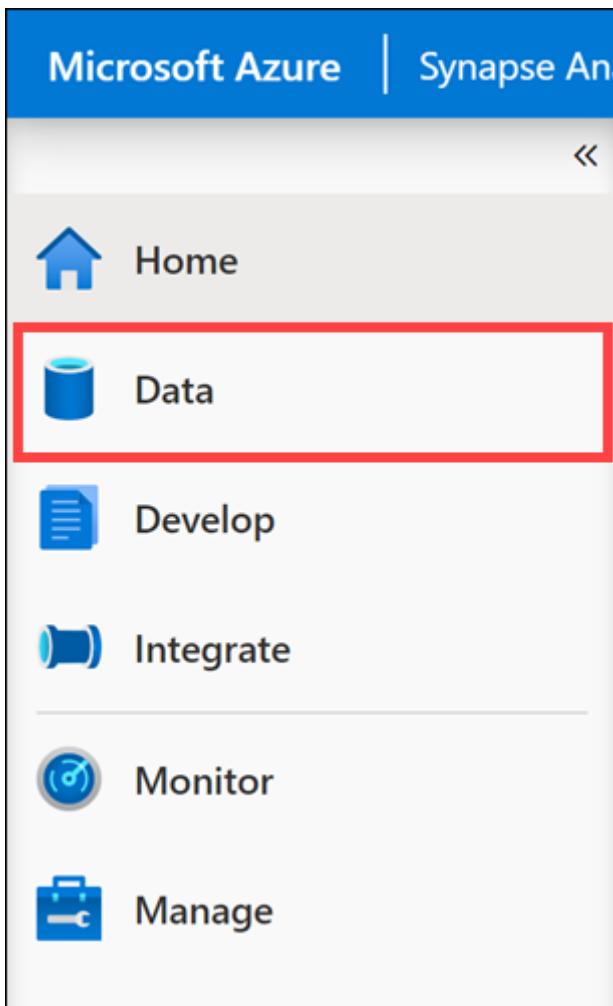


- ▶ Show Additional Metrics
- ▶ Event Timeline

Summary Metrics for 200 Completed Tasks

Metric	Min	25th percentile	Median
Duration	20 ms	42 ms	57 ms
GC Time	0 ms	0 ms	0 ms
Shuffle Read Size / Records	43.9 KB / 6471	50.2 KB / 7648	51.5 KB / 8057

16. Navigate back to the **Data** hub.

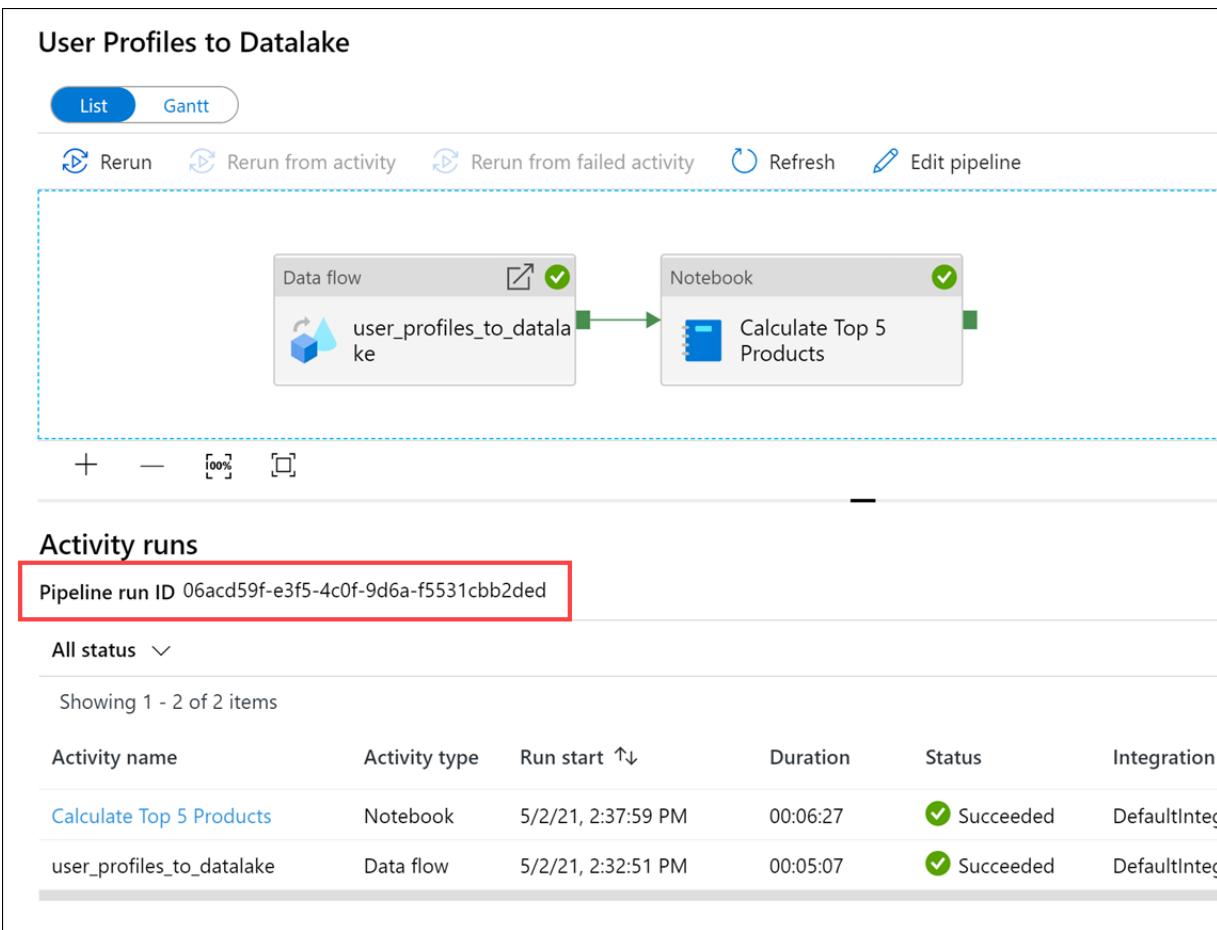


17. Select the **Linked** tab (1), select the **wwi-02** container (2) on the primary data lake storage account, navigate to the **top5-products** folder (3), and verify that a folder exists for the Parquet file whose name matches the **Pipeline run ID**.

The screenshot shows the Azure Data Lake Storage Gen2 Data view. On the left, under the "Linked" tab (1), the "wwi-02" container (2) is selected. On the right, the "top5-products" folder (3) is selected. In the "Name" column (4), a file named "06acd59f-e3f5-4c0f-9d6a-f5531cbb2ded.parquet" is listed, which matches the Pipeline run ID.

Name
06acd59f-e3f5-4c0f-9d6a-f5531cbb2ded.parquet
1db0b1ee-458a-48e2-b7df-446ce3c06996.parquet

As you can see, we have a file whose name matches the **Pipeline run ID** we noted earlier:



These values match because we passed in the Pipeline run ID to the `runId` parameter on the Notebook activity.

12 Module 10 - Optimize query performance with dedicated SQL pools in Azure Synapse

In this module, students will learn strategies to optimize data storage and processing when using dedicated SQL pools in Azure Synapse Analytics. The student will know how to use developer features, such as windowing and HyperLogLog functions, use data loading best practices, and optimize and improve query performance.

In this module, the student will be able to:

- Understand developer features of Azure Synapse Analytics
- Optimize data warehouse query performance in Azure Synapse Analytics
- Improve query performance

12.1 Lab details

- Module 10 - Optimize query performance with dedicated SQL pools in Azure Synapse
 - Lab details
 - Lab setup and pre-requisites
 - Exercise 0: Start the dedicated SQL pool
 - Exercise 1: Understanding developer features of Azure Synapse Analytics
 - * Task 1: Create tables and load data
 - * Task 2: Using window functions
 - Task 2.1: OVER clause
 - Task 2.2: Aggregate functions
 - Task 2.3: Analytic functions
 - Task 2.4: ROWS clause
 - * Task 3: Approximate execution using HyperLogLog functions
 - Exercise 2: Using data loading best practices in Azure Synapse Analytics

- * Task 1: Implement workload management
- * Task 2: Create a workload classifier to add importance to certain queries
- * Task 3: Reserve resources for specific workloads through workload isolation
- Exercise 3: Optimizing data warehouse query performance in Azure Synapse Analytics
 - * Task 1: Identify performance issues related to tables
 - * Task 2: Improve table structure with hash distribution and columnstore index
 - * Task 4: Improve further the table structure with partitioning
 - Task 4.1: Table distributions
 - Task 4.2: Indexes
 - Task 4.3: Partitioning
- Exercise 4: Improve query performance
 - * Task 1: Use materialized views
 - * Task 2: Use result set caching
 - * Task 3: Create and update statistics
 - * Task 4: Create and update indexes
 - * Task 5: Ordered Clustered Columnstore Indexes
- Exercise 5: Cleanup
 - * Task 1: Pause the dedicated SQL pool

12.2 Lab setup and pre-requisites

Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Complete the lab setup instructions for this module.

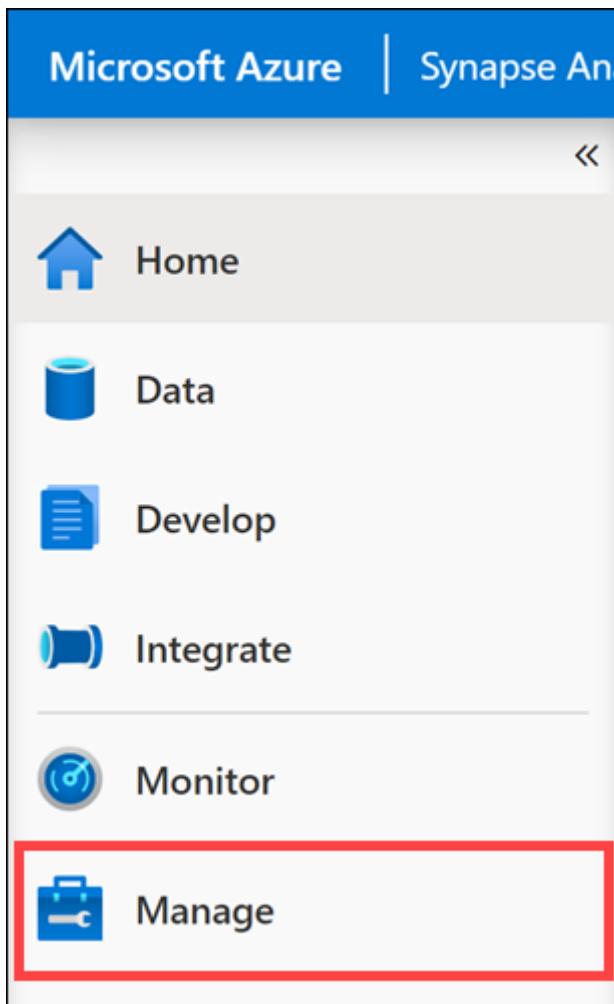
Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

12.3 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the "SQL pools" blade in the Azure portal. The left sidebar shows the following options:

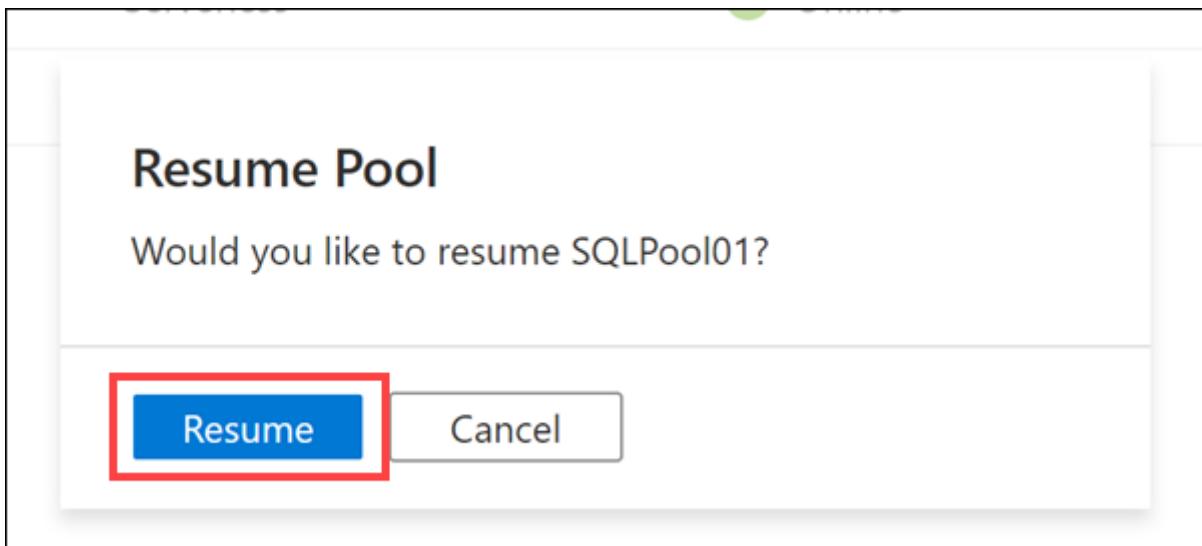
- Analytics pools
- SQL pools** (highlighted with a red box and a red number '1')
- Apache Spark pools
- External connections
- Linked services
- Azure Purview (Preview)
- Integration
- Triggers
- Integration runtimes
- Security
- Access control

The main area displays the "SQL pools" table:

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused	DW100c

A red box labeled '2' highlights the "Resume" button for the "SQLPool01" row.

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.



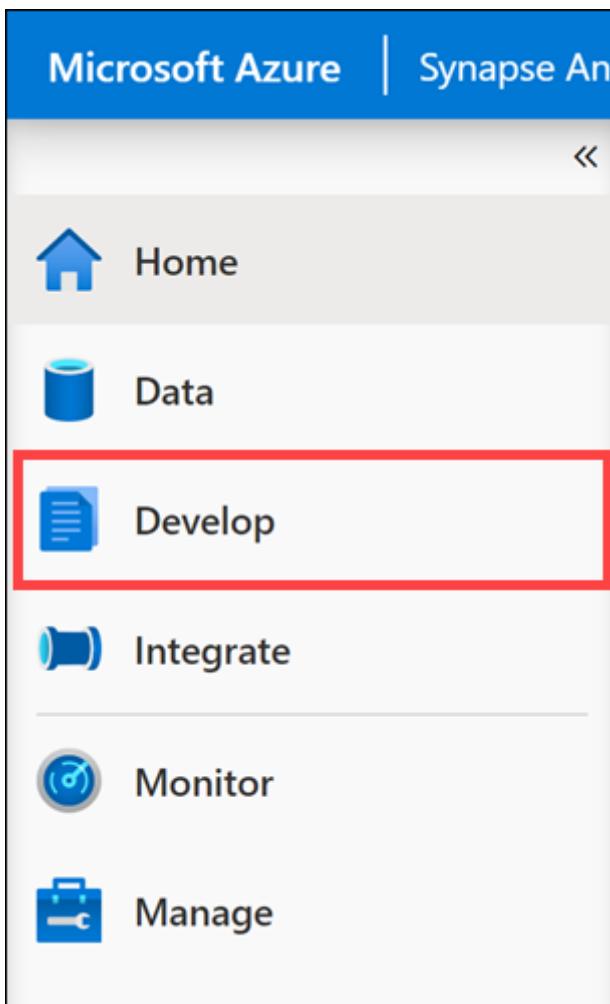
Wait until the dedicated SQL pool resumes.

12.4 Exercise 1: Understanding developer features of Azure Synapse Analytics

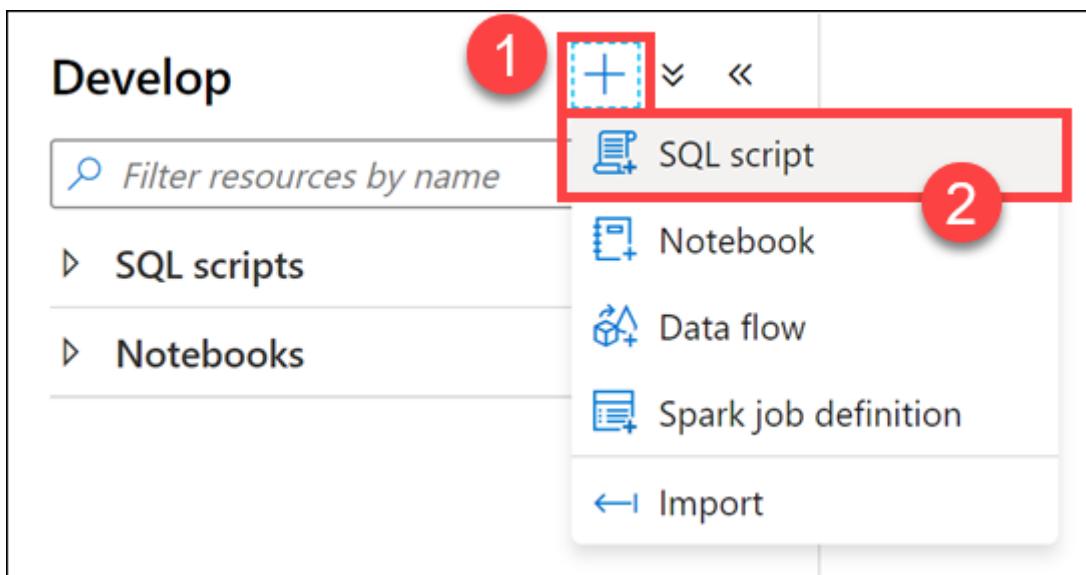
12.4.1 Task 1: Create tables and load data

Before you begin, we need to create a few new tables and load them with data.

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Develop** hub.



3. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



4. In the toolbar menu, connect to the **SQLPool01** database to execute the query.



5. In the query window, replace the script with the following to use the **OVER** clause with data from the `wwi_security.Sale` table:

```

IF OBJECT_ID(N'[dbo].[Category]', N'U') IS NOT NULL
DROP TABLE [dbo].[Category]

CREATE TABLE [dbo].[Category]
(
    [ID] [float] NOT NULL,
    [Category] [varchar](255) NULL,
    [SubCategory] [varchar](255) NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
GO

IF OBJECT_ID(N'[dbo].[Books]', N'U') IS NOT NULL
DROP TABLE [dbo].[Books]

CREATE TABLE [dbo].[Books]
(
    [ID] [float] NOT NULL,
    [BookListID] [float] NULL,
    [Title] [varchar](255) NULL,
    [Author] [varchar](255) NULL,
    [Duration] [float] NULL,
    [Image] [varchar](255) NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
GO

IF OBJECT_ID(N'[dbo].[BookConsumption]', N'U') IS NOT NULL

```

```

DROP TABLE [dbo].[BookConsumption]

CREATE TABLE [dbo].[BookConsumption]
(
    [BookID] [float] NULL,
    [Clicks] [float] NULL,
    [Downloads] [float] NULL,
    [Time Spent] [float] NULL,
    [Country] [varchar](255) NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
GO

IF OBJECT_ID(N'[dbo].[BookList]', N'U') IS NOT NULL
DROP TABLE [dbo].[BookList]

CREATE TABLE [dbo].[BookList]
(
    [ID] [float] NOT NULL,
    [CategoryID] [float] NULL,
    [BookList] [varchar](255) NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
GO

COPY INTO Category
FROM 'https://solliancepublicdata.blob.core.windows.net/cdp/csv/Category.csv'
WITH (
    FILE_TYPE = 'CSV',
    FIRSTROW = 2
)
GO

COPY INTO Books
FROM 'https://solliancepublicdata.blob.core.windows.net/cdp/csv/Books.csv'
WITH (
    FILE_TYPE = 'CSV',
    FIRSTROW = 2
)
GO

COPY INTO BookConsumption
FROM 'https://solliancepublicdata.blob.core.windows.net/cdp/csv/BookConsumption.csv'
WITH (
    FILE_TYPE = 'CSV',
    FIRSTROW = 2
)
GO

COPY INTO BookList
FROM 'https://solliancepublicdata.blob.core.windows.net/cdp/csv/BookList.csv'
WITH (
    FILE_TYPE = 'CSV',

```

```
FIRSTROW = 2  
)  
GO
```

6. Select **Run** from the toolbar menu to execute the SQL command.



After a few seconds, you should see that the query successfully completed.

12.4.2 Task 2: Using window functions

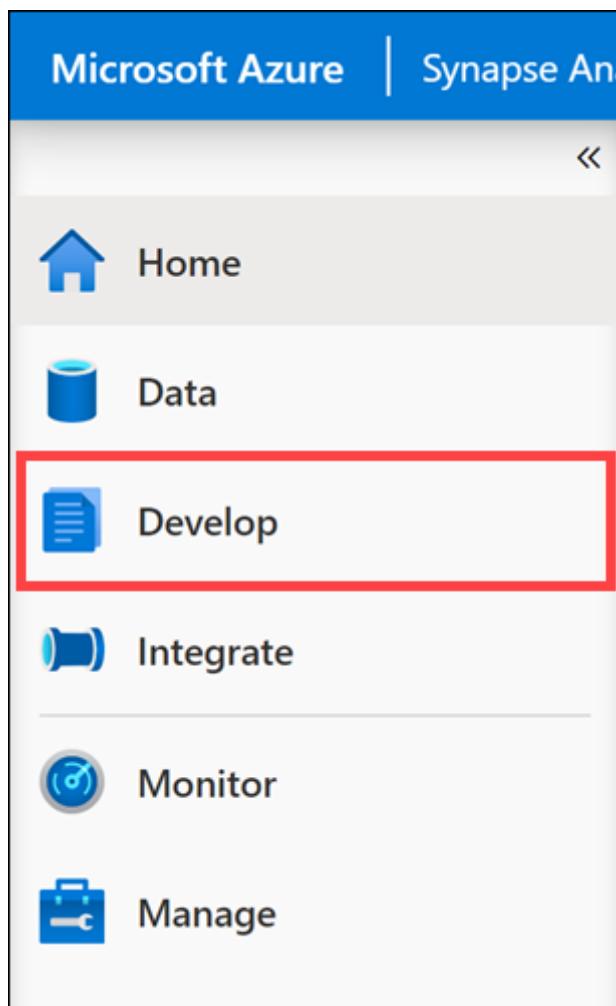
Tailwind Traders is looking for ways to more efficiently analyze their sales data without relying on expensive cursors, subqueries, and other outdated methods they use today.

You propose using window functions to perform calculations over a set of rows. With these functions, you treat groups of rows as an entity.

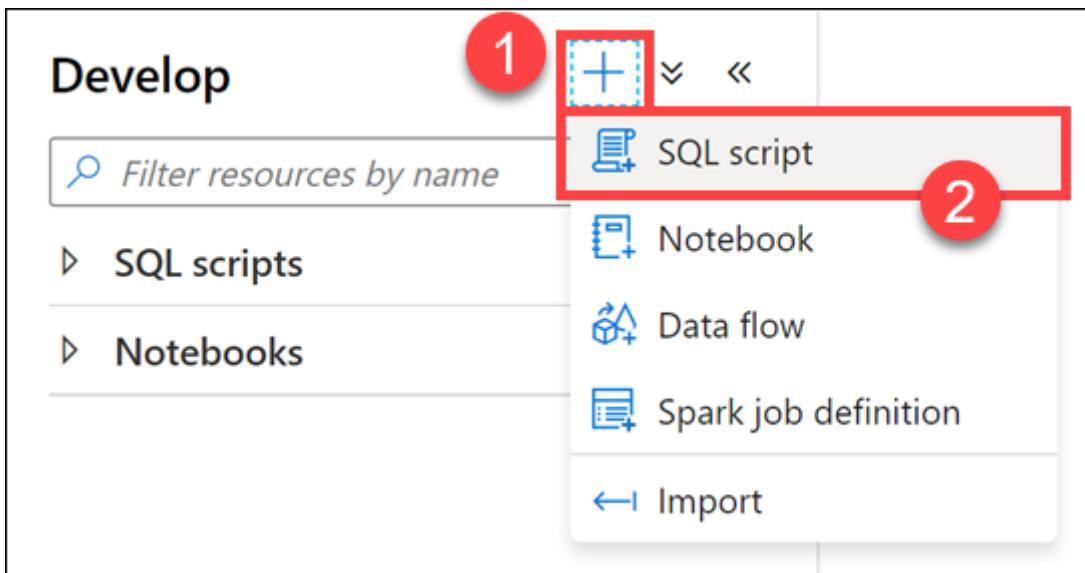
12.4.2.1 Task 2.1: OVER clause

One of the key components of window functions is the **OVER** clause. This clause determines the partitioning and ordering of a rowset before the associated window function is applied. That is, the **OVER** clause defines a window or user-specified set of rows within a query result set. A window function then computes a value for each row in the window. You can use the **OVER** clause with functions to compute aggregated values such as moving averages, cumulative aggregates, running totals, or a top N per group results.

1. Select the **Develop** hub.



2. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



3. In the toolbar menu, connect to the **SQLPool01** database to execute the query.



4. In the query window, replace the script with the following to use the **OVER** clause with data from the `wwi_security.Sale` table:

```

SELECT
    ROW_NUMBER() OVER(PARTITION BY Region ORDER BY Quantity DESC) AS "Row Number",
    Product,
    Quantity,
    Region
FROM wwi_security.Sale
WHERE Quantity <> 0
ORDER BY Region;

```

5. Select **Run** from the toolbar menu to execute the SQL command.



When we use `PARTITION BY` with the `OVER` clause (1), we divide the query result set into partitions. The window function is applied to each partition separately and computation restarts for each partition.

```

1  SELECT
2      ROW_NUMBER() OVER(PARTITION BY Region ORDER BY Quantity DESC) AS "Row Number",
3      Product,
4      Quantity,
5      Region
6  FROM wwi_security.Sale
7  WHERE Quantity <> 0
8  ORDER BY Region;

```

Results Messages

View **Table** Chart Export results ▾

Search

Row Number	Product	Quantity	Region
16	Wheel	5	Far West
17	Video Games	5	Far West
18	Video Games	5	Far West
19	Wireless headphones	3	Far West
20	Wireless headphones	3	Far West
21	Key Chain	2	Far West
22	Key Chain	2	Far West
1	Sunglasses	52	South East
2	Beach hat	52	South East
	Beach hat	52	

The script we executed uses the OVER clause with ROW_NUMBER function (1) to display a row number for each row within a partition. The partition in our case is the Region column. The ORDER BY clause (2) specified in the OVER clause orders the rows in each partition by the column Quantity. The ORDER BY clause in the SELECT statement determines the order in which the entire query result set is returned.

Scroll down in the results view until the **Row Number** count (3) starts over with a **different region** (4). Since the partition is set to Region, the ROW_NUMBER resets when the region changes. Essentially, we've partitioned by region and have a result set identified by the number of rows in that region.

12.4.2.2 Task 2.2: Aggregate functions

Now let's use aggregate functions with our window by expanding on our query that uses the OVER clause.

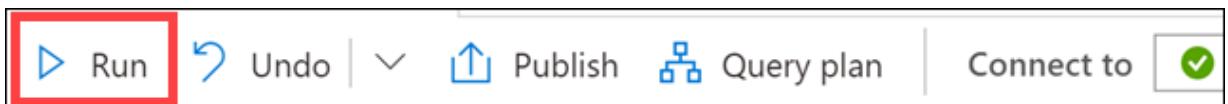
- In the query window, replace the script with the following to add aggregate functions:

```

SELECT
    ROW_NUMBER() OVER(PARTITION BY Region ORDER BY Quantity DESC) AS "Row Number",
    Product,
    Quantity,
    SUM(Quantity) OVER(PARTITION BY Region) AS Total,
    AVG(Quantity) OVER(PARTITION BY Region) AS Avg,
    COUNT(Quantity) OVER(PARTITION BY Region) AS Count,
    MIN(Quantity) OVER(PARTITION BY Region) AS Min,
    MAX(Quantity) OVER(PARTITION BY Region) AS Max,
    Region
FROM wwi_security.Sale
WHERE Quantity <> 0
ORDER BY Region;

```

- Select **Run** from the toolbar menu to execute the SQL command.



In our query, we added the SUM, AVG, COUNT, MIN, and MAX aggregate functions. Using the OVER clause is more efficient than using subqueries.

```

5   SUM(Quantity) OVER(PARTITION BY Region) AS Total,
6   AVG(Quantity) OVER(PARTITION BY Region) AS Avg,
7   COUNT(Quantity) OVER(PARTITION BY Region) AS Count,
8   MIN(Quantity) OVER(PARTITION BY Region) AS Min,
9   MAX(Quantity) OVER(PARTITION BY Region) AS Max,
10  Region
11  FROM wwi_security.Sale
12  WHERE Quantity > 0
13  ORDER BY Region;

```

Row Number	Product	Quantity	Total	Avg	Count	Min	Max	Region
17	video Games	5	328	14	22	2	32	Far West
18	Video Games	5	328	14	22	2	32	Far West
19	Wireless headphones	3	328	14	22	2	32	Far West
20	Wireless headphones	3	328	14	22	2	32	Far West
21	Key Chain	2	328	14	22	2	32	Far West
22	Key Chain	2	328	14	22	2	32	Far West
1	Sunglasses	52	962	32	30	5	52	South East
2	Beach hat	52	962	32	30	5	52	South East
3	Beach hat	52	962	32	30	5	52	South East
4	Sunglasses	52	962	32	30	5	52	South East

12.4.2.3 Task 2.3: Analytic functions

Analytic functions calculate an aggregate value based on a group of rows. Unlike aggregate functions, however, analytic functions can return multiple rows for each group. Use analytic functions to compute moving averages, running totals, percentages, or top-N results within a group.

Tailwind Traders has book sales data they import from their online store and wish to compute percentages of book downloads by category.

To do this, you decide to build window functions that use the PERCENTILE_CONT and PERCENTILE_DISC functions.

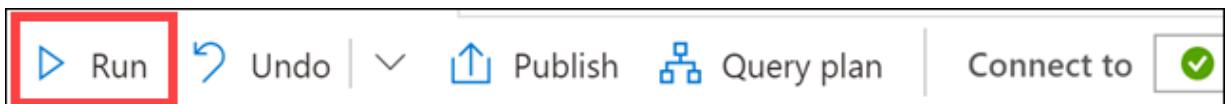
1. In the query window, replace the script with the following to add aggregate functions:

```

-- PERCENTILE_CONT, PERCENTILE_DISC
SELECT DISTINCT c.Category
,PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY bc.Downloads)
OVER (PARTITION BY Category) AS MedianCont
,PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY bc.Downloads)
OVER (PARTITION BY Category) AS MedianDisc
FROM dbo.Category AS c
INNER JOIN dbo.BookList AS bl
ON bl.CategoryID = c.ID
INNER JOIN dbo.BookConsumption AS bc
ON bc.BookID = bl.ID
ORDER BY Category

```

2. Select **Run** from the toolbar menu to execute the SQL command.



```

1 -- PERCENTILE_CONT, PERCENTILE_DISC
2 SELECT DISTINCT c.Category
3 ,PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY bc.Downloads)
4 ,OVER (PARTITION BY Category) AS MedianCont
5 ,PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY bc.Downloads)
6 OVER (PARTITION BY Category) AS MedianDisc
7 FROM dbo.Category AS c
8 INNER JOIN dbo.BookList AS bl
9 ON bl.CategoryID = c.ID
10 INNER JOIN dbo.BookConsumption AS bc
11 ON bc.BookID = bl.ID
12 ORDER BY Category
13

```

Results Messages

View **Table** Chart Export results

Search

Category	MedianCont	MedianDisc
Favorites	304.75	231.833333333333
Fiction	1091.58333333333	935
Romance	78.583333333333	59.16666666666667
Sci-fi	54.75	44.3333333333333
Trending Now	187	153.333333333333

In this query, we use **PERCENTILE_CONT** (1) and **PERCENTILE_DISC** (2) to find the median number of downloads in each book category. These functions may not return the same value. **PERCENTILE_CONT** interpolates the appropriate value, which may or may not exist in the data set, while **PERCENTILE_DISC** always returns an actual value from the set. To explain further, **PERCENTILE_DISC** computes a specific percentile for sorted values in an entire rowset or within a rowset's distinct partitions.

The 0.5 value passed to the **percentile functions** (1 & 2) computes the 50th percentile, or the median, of the downloads.

The **WITHIN GROUP** expression (3) specifies a list of values to sort and compute the percentile over. Only one **ORDER BY** expression is allowed, and the default sort order is ascending.

The **OVER** clause (4) divides the **FROM** clause's result set into partitions, in this case, **Category**. The percentile function is applied to these partitions.

3. In the query window, replace the script with the following to use the LAG analytic function:

```
--LAG Function
SELECT ProductId,
       [Hour],
       [HourSalesTotal],
       LAG(HourSalesTotal,1,0) OVER (ORDER BY [Hour]) AS PreviousHouseSalesTotal,
       [HourSalesTotal] - LAG(HourSalesTotal,1,0) OVER (ORDER BY [Hour]) AS Diff
FROM (
    SELECT ProductId,
           [Hour],
```

```

SUM(TotalAmount) AS HourSalesTotal
FROM [wwi_perf].[Sale_Index]
WHERE ProductId = 3848 AND [Hour] BETWEEN 8 AND 20
GROUP BY ProductID, [Hour]) as HourTotals

```

Tailwind Traders wants to compare the sales totals for a product over an hourly basis over time, showing the difference in value.

To accomplish this, you use the LAG analytic function. This function accesses data from a previous row in the same result set without the use of a self-join. LAG provides access to a row at a given physical offset that comes before the current row. We use this analytic function to compare values in the current row with values in a previous row.

4. Select **Run** from the toolbar menu to execute the SQL command.

The screenshot shows a SQL query editor interface. At the top, there are buttons for Run (highlighted with a red box), Undo, Publish, Query plan, and Connect to. The main area contains a numbered SQL script:

```

1  --LAG Function
2  SELECT ProductId,
3      [Hour],
4      [HourSalesTotal],
5      LAG(HourSalesTotal,1,0) OVER (ORDER BY [Hour]) AS PreviousHouseSalesTotal,
6      [HourSalesTotal] - LAG(HourSalesTotal,1,0) OVER (ORDER BY [Hour]) AS Diff
7  FROM (
8      SELECT ProductId,
9          [Hour],
10         SUM(TotalAmount) AS HourSalesTotal
11    FROM [wwi_perf].[Sale_Index]
12   WHERE ProductId = 3848 AND [Hour] BETWEEN 8 AND 20
13   GROUP BY ProductID, [Hour]) as HourTotals

```

The results table below shows the output for ProductId 3848 from hour 8 to 14. Red annotations highlight the 'Diff' column (column 3) and the 'PreviousHouseSalesTotal' column (column 2). Red arrows point from the 'PreviousHouseSalesTotal' values in one row to the 'Diff' value in the next row, illustrating the calculation of the difference between consecutive hours.

ProductId	Hour	HourSalesTotal	PreviousHouseSalesTotal	Diff
3848	8	100531.88	0.00	100531.88
3848	9	98647.60	100531.88	-1884.28
3848	10	104688.38	98647.60	6040.78
3848	11	101917.38	104688.38	-2771.00
3848	12	94546.52	101917.38	-7370.86
3848	13	104078.76	94546.52	9532.24
3848	14	95128.43	104078.76	-8950.33

In this query, we use the **LAG function (1)** to return the **difference in sales (2)** for a specific product over peak sales hours (8-20). We also calculate the difference in sales from one row to the next (3). Notice that because there is no lag value available for the first row, the default of zero (0) is returned.

12.4.2.4 Task 2.4: ROWS clause

The ROWS and RANGE clauses further limit the rows within the partition by specifying start and end points within the partition. This is done by specifying a range of rows with respect to the current row either by logical association or physical association. Physical association is achieved by using the ROWS clause.

Tailwind Traders wants to find the books that have the fewest downloads by country, while displaying the total number of downloads for each book within each country in ascending order.

To achieve this, you use ROWS in combination with UNBOUNDED PRECEDING to limit the rows within the Country partition, specifying that the window start with the first row of the partition.

1. In the query window, replace the script with the following to add aggregate functions:

```
-- ROWS UNBOUNDED PRECEDING
SELECT DISTINCT bc.Country, b.Title AS Book, bc.Downloads
    ,FIRST_VALUE(b.Title) OVER (PARTITION BY Country
        ORDER BY Downloads ASC ROWS UNBOUNDED PRECEDING) AS FewestDownloads
FROM dbo.BookConsumption AS bc
INNER JOIN dbo.Books AS b
    ON b.ID = bc.BookID
ORDER BY Country, Downloads
```

2. Select **Run** from the toolbar menu to execute the SQL command.

The screenshot shows a SQL query window with the following interface elements:

- Toolbar:** Includes buttons for Run (highlighted with a red box), Undo, Publish, Query plan, and Connect to.
- Query Editor:** Displays the SQL code with annotations:
 - A red box highlights the entire query.
 - A yellow box highlights the `FIRST_VALUE(b.Title)` part of the analytic function.
 - A red circle with the number "1" is placed over the `ROWS UNBOUNDED PRECEDING` clause.
 - A red box highlights the last two columns of the result table.
 - A red circle with the number "2" is placed over the row for Sweden.
- Results:** A table showing the output of the query. The columns are Country, Book, Downloads, and FewestDownloads.

Country	Book	Downloads	FewestDownloads
Germany	LOVE LIKE THAT (The Romance Chri...	1295.00000000000	Harry Potter - The Ultimate Quiz Book
Germany	A Reaper at the Gates	1418	Harry Potter - The Ultimate Quiz Book
Germany	Harry Potter - The Ultimate Book...	1565.66666666667	Harry Potter - The Ultimate Quiz Book
Germany	Harry Potter - The Amazing Quiz ...	1721.66666666667	Harry Potter - The Ultimate Quiz Book
Germany	101 Amazing Harry Potter Facts	2410.33333333333	Harry Potter - The Ultimate Quiz Book
Germany	Harry Potter - The Ultimate Quiz	2812.66666666667	Harry Potter - The Ultimate Quiz Book
Sweden	Burn for Me	1.199999999999999	Burn for Me
Sweden	Diamond Fire	6	Burn for Me
Sweden	Game Of Thrones The Quiz Book...	8.19999999999999	Burn for Me
Sweden	Harry Potter - The Complete Qui...	11	Burn for Me
Sweden	The World of Ice and Fire: The U...	15.6	Burn for Me

In this query, we use the `FIRST_VALUE` analytic function to retrieve the book title with the fewest downloads, as indicated by the `ROWS UNBOUNDED PRECEDING` clause over the Country partition (1). The `UNBOUNDED PRECEDING` option set the window start to the first row of the partition, giving us the title of the book with the fewest downloads for the country within the partition.

In the result set, we can scroll through the list of books by country, sorted by number of downloads in ascending order. Here we see that for Germany, **Harry Potter - The Ultimate Quiz Book** (not to be confused with **Harry Potter - The Ultimate Quiz**, which had the most) had the fewest downloads, and **Burn for Me** had the fewest in Sweden (2).

12.4.3 Task 3: Approximate execution using HyperLogLog functions

As Tailwind Traders starts to work with very large data sets, they struggle with slow running queries. For instance, obtaining a distinct count of all customers in the early stages of data exploration slows down the process. How can they speed up these queries?

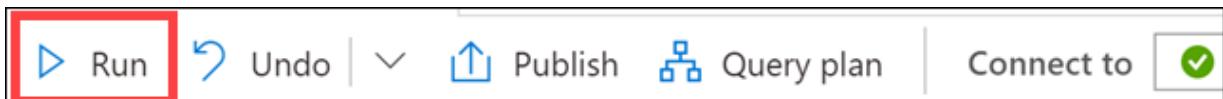
You decide to use approximate execution using HyperLogLog accuracy to reduce query latency in exchange for a small reduction in accuracy. This tradeoff works for Tailwind Trader's situation where they just need to get a feel for the data.

To understand their requirements, let's first execute a distinct count over the large `Sale_Heap` table to find the count of distinct customers.

1. In the query window, replace the script with the following:

```
SELECT COUNT(DISTINCT CustomerId) from wwi_poc.Sale
```

2. Select **Run** from the toolbar menu to execute the SQL command.



The query takes between 50 and 70 seconds to execute. That is expected, as distinct counts are one of the most difficult types of queries to optimize.

The result should be 1,000,000.

3. In the query window, replace the script with the following to use the HyperLogLog approach:

```
SELECT APPROX_COUNT_DISTINCT(CustomerId) from wwi_poc.Sale
```

4. Select **Run** from the toolbar menu to execute the SQL command.



The query should take much less time to execute. The result isn't quite the same, for example, it may be 1,001,619.

`APPROX_COUNT_DISTINCT` returns a result with a **2% accuracy** of true cardinality on average.

This means, if `COUNT (DISTINCT)` returns 1,000,000, HyperLogLog will return a value in the range of 999,736 to 1,016,234.

12.5 Exercise 2: Using data loading best practices in Azure Synapse Analytics

12.5.1 Task 1: Implement workload management

Running mixed workloads can pose resource challenges on busy systems. Solution Architects seek ways to separate classic data warehousing activities (such as loading, transforming, and querying data) to ensure that enough resources exist to hit SLAs.

Workload management for dedicated SQL pools in Azure Synapse consists of three high-level concepts: Workload Classification, Workload Importance and Workload Isolation. These capabilities give you more control over how your workload utilizes system resources.

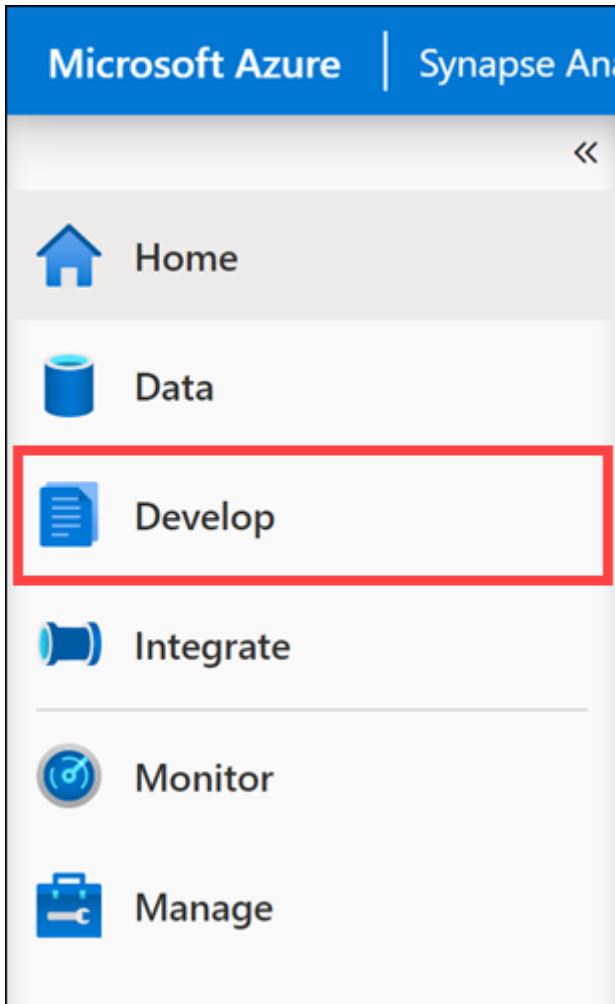
Workload importance influences the order in which a request gets access to resources. On a busy system, a request with higher importance has first access to resources. Importance can also ensure ordered access to locks.

Workload isolation reserves resources for a workload group. Resources reserved in a workload group are held exclusively for that workload group to ensure execution. Workload groups also allow you to define the amount of resources that are assigned per request, much like resource classes do. Workload groups give you the ability to reserve or cap the amount of resources a set of requests can consume. Finally, workload groups are a mechanism to apply rules, such as query timeout, to requests.

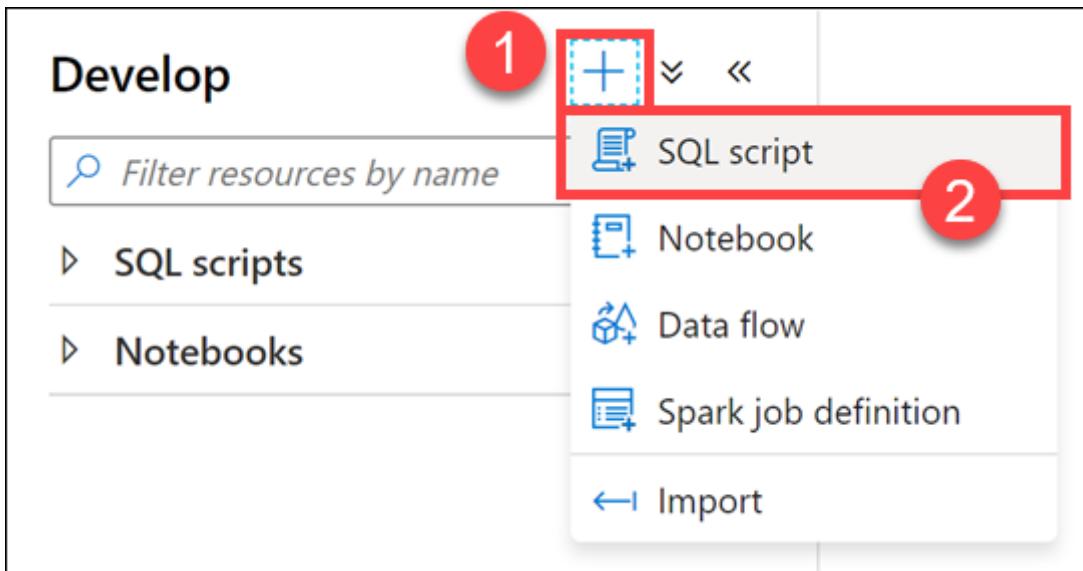
12.5.2 Task 2: Create a workload classifier to add importance to certain queries

Tailwind Traders has asked you if there is a way to mark queries executed by the CEO as more important than others, so they don't appear slow due to heavy data loading or other workloads in the queue. You decide to create a workload classifier and add importance to prioritize the CEO's queries.

1. Select the **Develop** hub.



2. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



3. In the toolbar menu, connect to the **SQLPool01** database to execute the query.



4. In the query window, replace the script with the following to confirm that there are no queries currently being run by users logged in as `asa.sql.workload01`, representing the CEO of the organization or `asa.sql.workload02` representing the data analyst working on the project:

--First, let's confirm that there are no queries currently being run by users logged in workload01

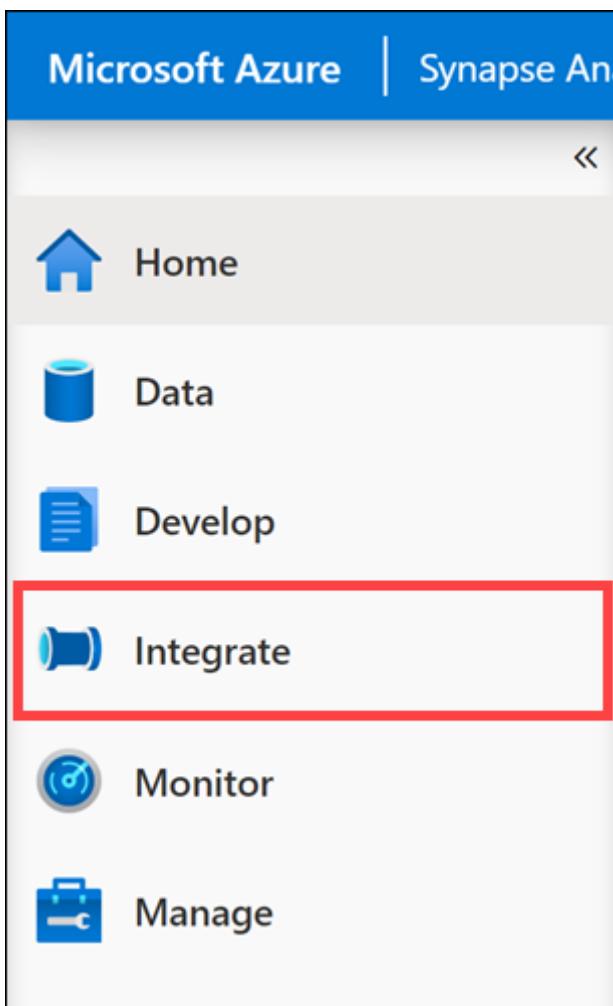
```
SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended')
--and submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,s.login_name
```

5. Select **Run** from the toolbar menu to execute the SQL command.

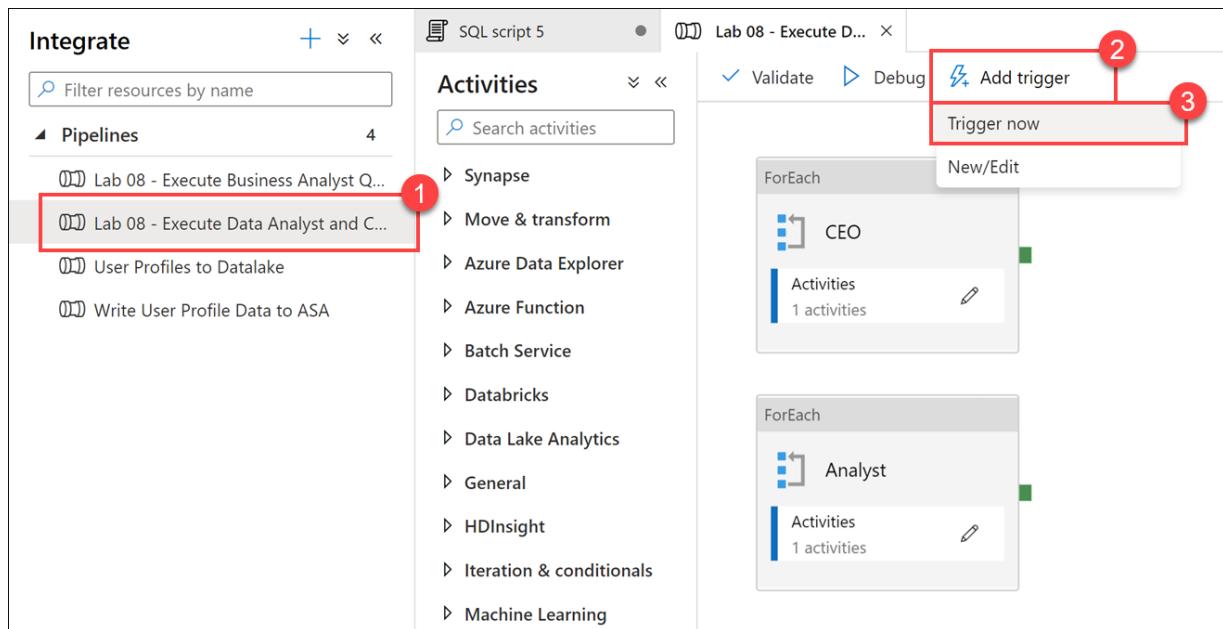


Now that we have confirmed that there are no running queries, we need to flood the system with queries and see what happens for `asa.sql.workload01` and `asa.sql.workload02`. To do this, we'll run a Synapse Pipeline which triggers queries.

6. Select the **Integrate** hub.



7. Select the **Lab 08 - Execute Data Analyst and CEO Queries** pipeline (1), which will run / trigger the `asa.sql.workload01` and `asa.sql.workload02` queries. Select **Add trigger** (2), then **Trigger now** (3). In the dialog that appears, select **OK**.



Note: Leave this tab open since you will come back to this pipeline again.

- Let's see what happened to all the queries we just triggered as they flood the system. In the query window, replace the script with the following:

```
SELECT s.login_name, r.[Status], r.Importance, submit_time, start_time ,s.session_id FROM sys.dm_pdw_exec_requests s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended') and submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,status
```

- Select **Run** from the toolbar menu to execute the SQL command.



You should see an output similar to the following:

3 WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and Importance
 4 is not NULL AND r.[status] in ('Running','Suspended') and submit_time>dateadd(minute,-
 5 ORDER BY submit_time ,status

Results [Messages](#)

View [Table](#) [Chart](#) [Export results](#)

Search

Login_name	Status	Importance	Subm
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload01	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-
asa.sql.workload02	Running	normal	2020-

Note: This query could take over a minute to execute. If it takes longer than this, cancel the query and run it again.

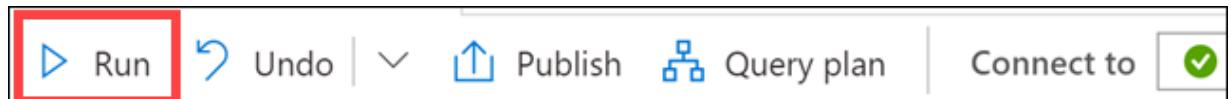
Notice that the **Importance** level for all queries is set to **normal**.

10. We will give our **asa.sql.workload01** user queries priority by implementing the **Workload Importance** feature. In the query window, replace the script with the following:

```
IF EXISTS (SELECT * FROM sys.workload_management_workload_classifiers WHERE name = 'CEO')
BEGIN
    DROP WORKLOAD CLASSIFIER CEO;
END
CREATE WORKLOAD CLASSIFIER CEO
    WITH (WORKLOAD_GROUP = 'largerc'
    , MEMBERNAME = 'asa.sql.workload01', IMPORTANCE = High);
```

We are executing this script to create a new **Workload Classifier** named **CEO** that uses the **largerc** Workload Group and sets the **Importance** level of the queries to **High**.

11. Select **Run** from the toolbar menu to execute the SQL command.



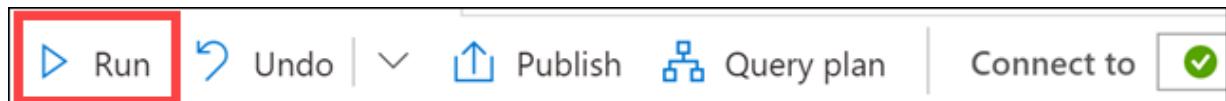
12. Let's flood the system again with queries and see what happens this time for **asa.sql.workload01** and **asa.sql.workload02** queries. To do this, we'll run a Synapse Pipeline which triggers queries. **Select** the **Integrate** Tab, **run** the **Lab 08 - Execute Data Analyst and CEO Queries** pipeline, which will run / trigger the **asa.sql.workload01** and **asa.sql.workload02** queries.
13. In the query window, replace the script with the following to see what happens to the **asa.sql.workload01** queries this time:

```

SELECT s.login_name, r.[Status], r.Importance, submit_time, start_time ,s.session_id FROM sys.dm_pdw_exec_requests r
JOIN sys.dm_pdw_exec_requests s ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended') and submit_time>dateadd(minute,-2,getdate())
ORDER BY submit_time ,status desc

```

14. Select **Run** from the toolbar menu to execute the SQL command.



You should see an output similar to the following:

3 WHERE s.login_name IN ('asa.sql.workload01','asa.sql.workload02') and Importance
 4 is not NULL AND r.[status] in ('Running','Suspended') and submit_time>dateadd(m
 5 ORDER BY submit_time ,status desc

Results Messages

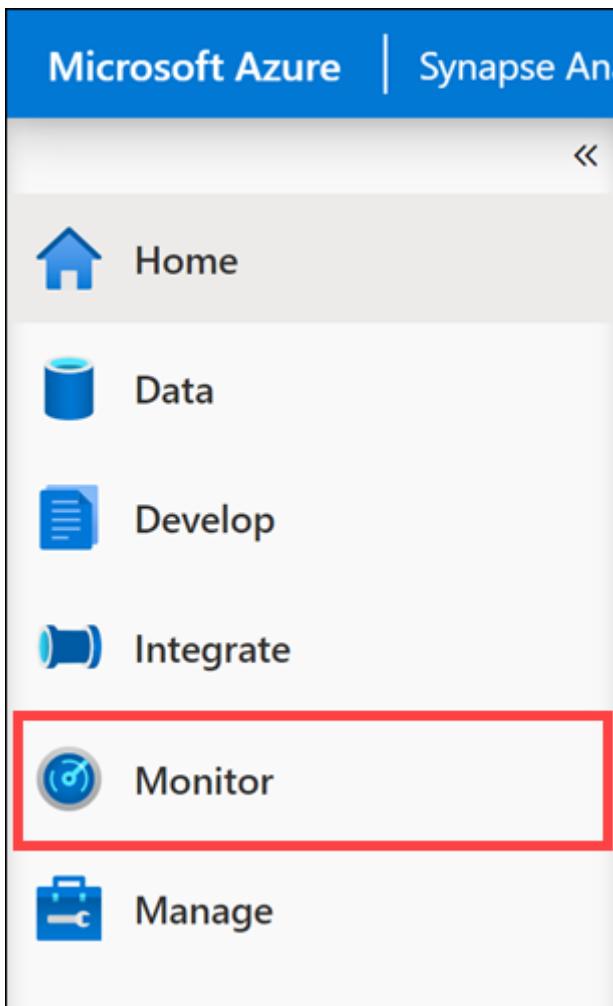
View **Table** Chart Export results ▾

Search

Login_name	Status	Importance
asa.sql.workload01	Suspended	high
asa.sql.workload01	Suspended	high
asa.sql.workload02	Suspended	normal
asa.sql.workload01	Suspended	high
asa.sql.workload01	Suspended	high
asa.sql.workload02	Suspended	normal
asa.sql.workload01	Suspended	high
asa.sql.workload02	Suspended	normal

Notice that the queries executed by the `asa.sql.workload01` user have a **high** importance.

15. Select the **Monitor** hub.



- Select **Pipeline runs** (1), and then select **Cancel recursive** (2) for each running Lab 08 pipelines, marked **In progress** (3). This will help speed up the remaining tasks.

Pipeline runs

Pipeline name	Run start ↑	Run end	Duration	Triggered by	Status
Lab 08 - Execute D...	2/26/21, 4:40:02 PM	--	00:00:35	Manual trigger	In progress
Lab 08 - Execute Data Analyst ...	Cancel recursive 7 PM	--	00:02:30	Manual trigger	In progress
Setup - Import User Profile Da...	2/26/21, 8:55:28 AM	2/26/21, 9:00:35 AM	00:05:06	Manual trigger	Succeeded
Setup - Load SQL Pool	2/26/21, 7:16:19 AM	2/26/21, 7:17:01 AM	00:00:41	Manual trigger	Succeeded
Setup - Load SQL Pool (global)	2/26/21, 6:19:07 AM	2/26/21, 7:01:04 AM	00:41:56	Manual trigger	Succeeded

Please note: If you see that any of these pipeline activities failed, that is OK. The activities within have a 2-minute timeout so they do not disrupt the queries run during this lab.

12.5.3 Task 3: Reserve resources for specific workloads through workload isolation

Workload isolation means resources are reserved, exclusively, for a workload group. Workload groups are containers for a set of requests and are the basis for how workload management, including workload isolation, is configured on a system. A simple workload management configuration can manage data loads and user queries.

In the absence of workload isolation, requests operate in the shared pool of resources. Access to resources in the shared pool is not guaranteed and is assigned on an importance basis.

Given the workload requirements provided by Tailwind Traders, you decide to create a new workload group called **CEO** to reserve resources for queries executed by the CEO.

Let's start by experimenting with different parameters.

1. In the query window, replace the script with the following:

```
IF NOT EXISTS (SELECT * FROM sys.workload_management_workload_groups WHERE name = 'CEO Demo')
BEGIN
    Create WORKLOAD GROUP CEO Demo WITH
    ( MIN_PERCENTAGE_RESOURCE = 50          -- integer value
    ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 25 --
    ,CAP_PERCENTAGE_RESOURCE = 100
    )
END
```

The script creates a workload group called CEO Demo to reserve resources exclusively for the workload group. In this example, a workload group with a MIN_PERCENTAGE_RESOURCE set to 50% and REQUEST_MIN_RESOURCE_GRANT_PERCENT set to 25% is guaranteed 2 concurrency.

Note: This query could take up to a minute to execute. If it takes longer than this, cancel the query and run it again.

2. Select **Run** from the toolbar menu to execute the SQL command.

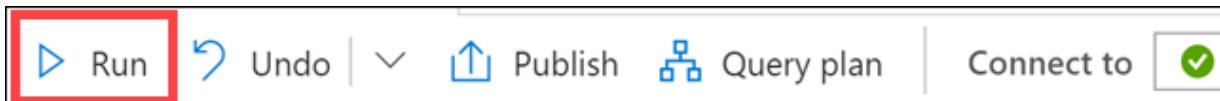


3. In the query window, replace the script with the following to create a Workload Classifier called CEO Dream Demo that assigns a workload group and importance to incoming requests:

```
IF NOT EXISTS (SELECT * FROM sys.workload_management_classifiers WHERE name = 'CEO Dream Demo')
BEGIN
    Create Workload Classifier CEO Dream Demo with
    ( Workload_Group ='CEO Demo', MemberName='asa.sql.workload02', IMPORTANCE = BELOW_NORMAL);
END
```

This script sets the Importance to **BELLOW_NORMAL** for the `asa.sql.workload02` user, through the new `CEO Dream Demo` Workload Classifier.

4. Select **Run** from the toolbar menu to execute the SQL command.



5. In the query window, replace the script with the following to confirm that there are no active queries being run by `asa.sql.workload02` (suspended queries are OK):

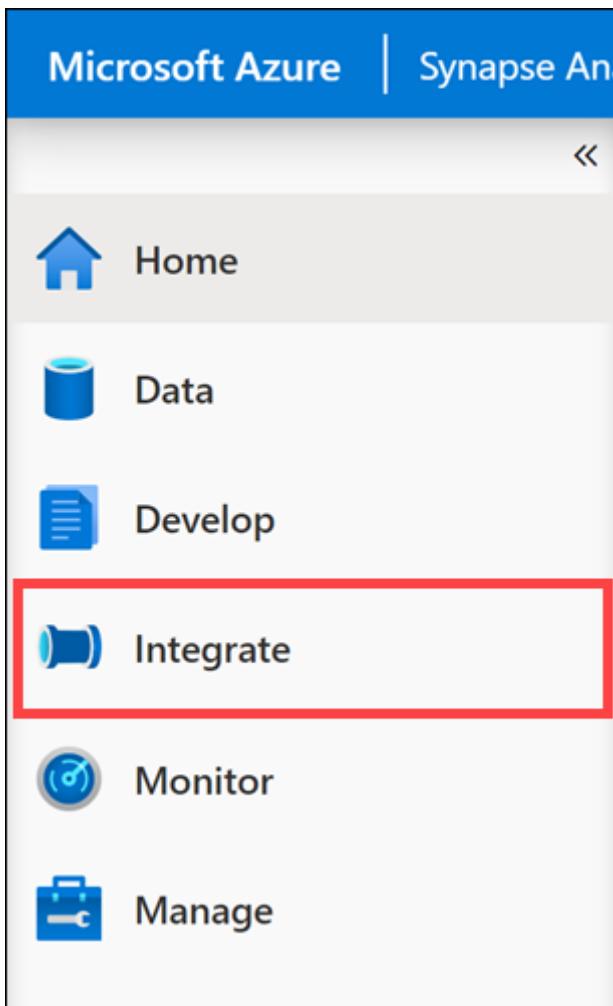
```
SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended')
ORDER BY submit_time, status
```

Note: If there are still active queries, wait a minute or two. They are configured to time out after two minutes since canceling the pipeline does not always cancel the queries.

6. Select **Run** from the toolbar menu to execute the SQL command.



7. Select the **Integrate** hub.



8. Select the **Lab 08 - Execute Business Analyst Queries Pipeline** (1), which will run / trigger `asa.sql.workload02` queries. Select **Add trigger** (2), then **Trigger now** (3). In the dialog that appears, select **OK**.

Note: Leave this tab open since you will come back to this pipeline again.

9. In the query window, replace the script with the following to see what happened to all the `asa.sql.workload02` queries we just triggered as they flood the system:

```
SELECT s.login_name, r.[Status], r.Importance, submit_time,
start_time ,s.session_id FROM sys.dm_pdw_exec_sessions s
JOIN sys.dm_pdw_exec_requests r ON s.session_id = r.session_id
WHERE s.login_name IN ('asa.sql.workload02') and Importance
is not NULL AND r.[status] in ('Running','Suspended')
ORDER BY submit_time, status
```

10. Select **Run** from the toolbar menu to execute the SQL command.

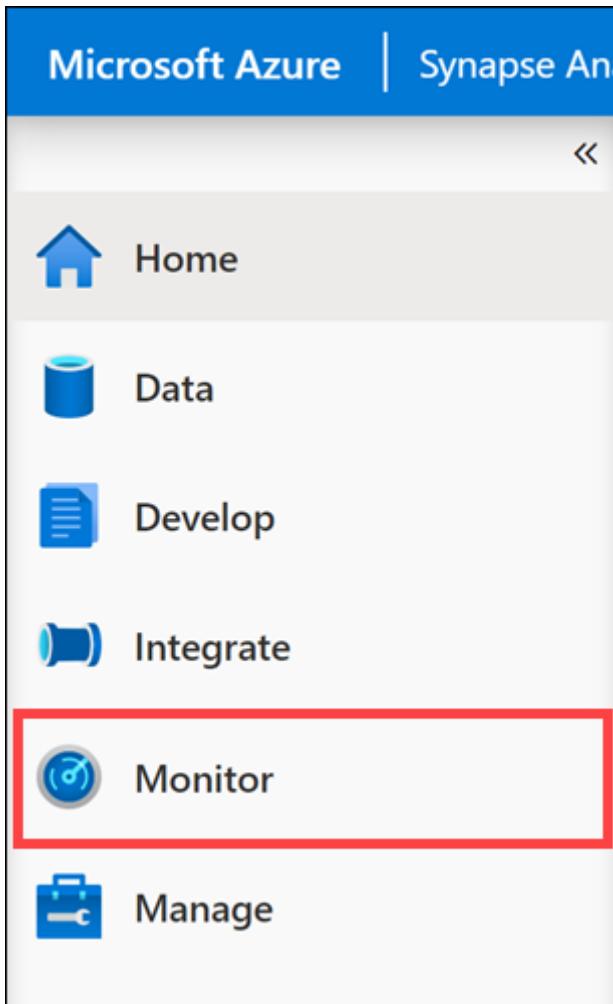


You should see an output similar to the following that shows the importance for each session set to `below_normal`:

Login_name	Status	Importance	Submit_time
asa.sql.workload02	Running	below_normal	2020
asa.sql.workload02	Running	below_normal	2020
asa.sql.workload02	Running	below_normal	2020
asa.sql.workload02	Running	below_normal	2020

Notice that the running scripts are executed by the `asa.sql.workload02` user (1) with an Importance level of `below_normal` (2). We have successfully configured the business analyst queries to execute at a lower importance than the CEO queries. We can also see that the `CEOdreamDemo` Workload Classifier works as expected.

11. Select the **Monitor** hub.



12. Select **Pipeline runs** (1), and then select **Cancel recursive** (2) for each running Lab 08 pipelines, marked **In progress** (3). This will help speed up the remaining tasks.

Pipeline runs

Triggered Debug Rerun Cancel Refresh Edit columns

End time : Last 24 hours (9/7/20 11:38 PM - 9/8/20 11:38 PM) Time zone : Eastern Time (US & Canada) (UT... Status : All status Runs : Latest runs

Showing 1 - 10 items

Pipeline name	Run start ↑	Run end	Duration	Triggered by	Status
Lab 08 - Execute B...	9/9/20, 10:03:02 AM	--	00:05:24	Manual trigger	2 In progress
Lab 08 - Execute Data Analyst...	9/8/20, 11:04 PM	9/8/20, 11:39:22 PM	00:20:18	Manual trigger	Cancelled
Lab 08 - Execute Data Analyst...	9/8/20, 11:16:35 PM	9/8/20, 11:39:19 PM	00:22:44	Manual trigger	Cancelled

13. Return to the query window under the **Develop** hub. In the query window, replace the script with the following to set 3.25% minimum resources per request:

```
IF EXISTS (SELECT * FROM sys.workload_management_workload_classifiers WHERE group_name = 'CEO Demo')
BEGIN
    Drop Workload Classifier CEO Dream Demo
    DROP WORKLOAD GROUP CEO Demo
END
--- Creates a workload group 'CEO Demo'.
Create WORKLOAD GROUP CEO Demo WITH
(
    MIN_PERCENTAGE_RESOURCE = 26 -- integer value
    ,REQUEST_MIN_RESOURCE_GRANT_PERCENT = 3.25 -- factor of 26 (guaranteed more than 4 concurrent users)
    ,CAP_PERCENTAGE_RESOURCE = 100
)
--- Creates a workload Classifier 'CEO Dream Demo'.
Create Workload Classifier CEO Dream Demo with
```

```
(Workload_Group = 'CEO Demo', MemberName='asa.sql.workload02', IMPORTANCE = BELOW_NORMAL);
```

Note: If this query takes more than 45 seconds to run, cancel it, then run it again.

Note: Configuring workload containment implicitly defines a maximum level of concurrency.

With a CAP_PERCENTAGE_RESOURCE set to 60% and a REQUEST_MIN_RESOURCE_GRANT_PERCENT set to 1%, up to a 60-concurrency level is allowed for the workload group. Consider the method included below for determining the maximum concurrency:

[Max Concurrency] = [CAP_PERCENTAGE_RESOURCE] / [REQUEST_MIN_RESOURCE_GRANT_PERCENT]

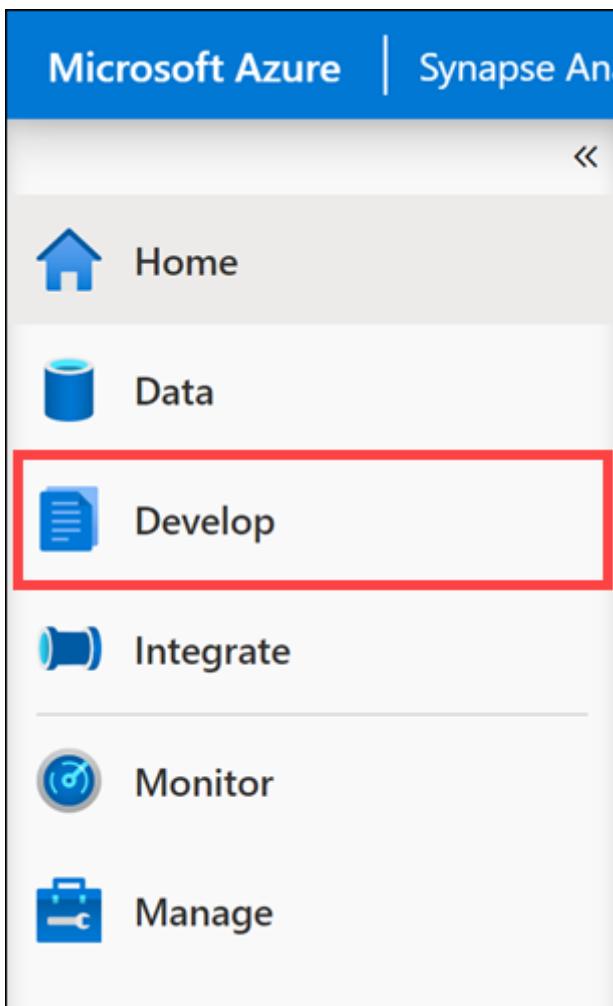
14. Select **Run** from the toolbar menu to execute the SQL command.



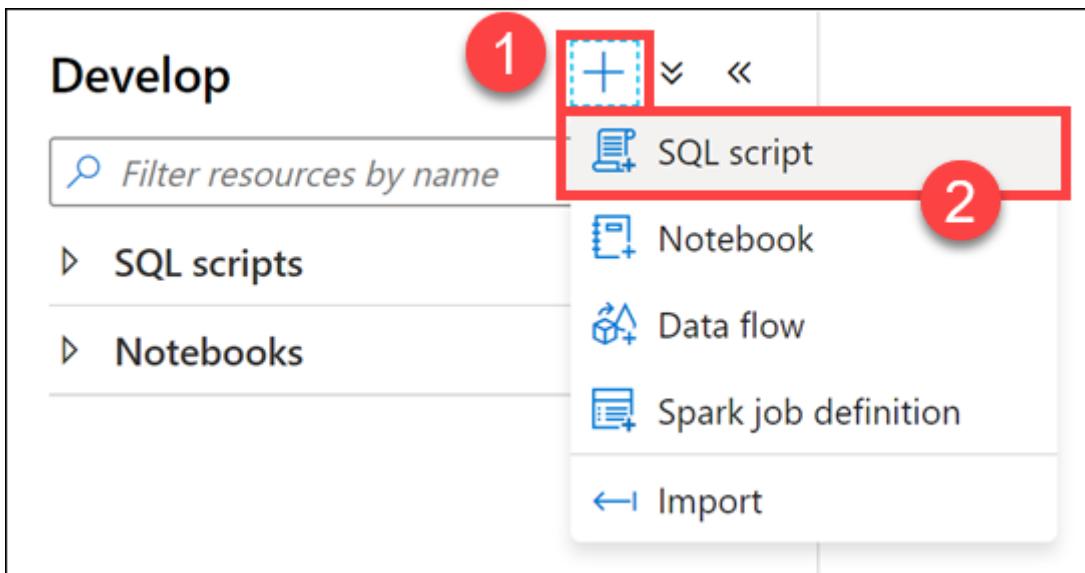
12.6 Exercise 3: Optimizing data warehouse query performance in Azure Synapse Analytics

12.6.1 Task 1: Identify performance issues related to tables

1. Select the **Develop** hub.



2. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



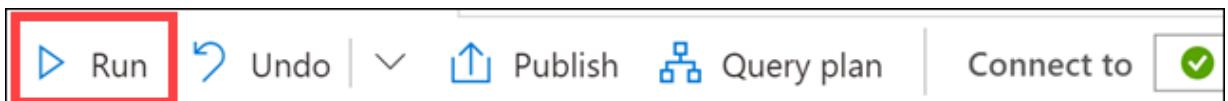
3. In the toolbar menu, connect to the **SQLPool01** database to execute the query.



4. In the query window, replace the script with the following to count the number of records in a heap table:

```
SELECT  
    COUNT_BIG(*)  
FROM  
    [wwi_poc].[Sale]
```

5. Select **Run** from the toolbar menu to execute the SQL command.



The script takes up to **15 seconds** to execute and returns a count of ~982 million rows in the table.

If the script is still running after 45 seconds, click on Cancel.

Note: Do not execute this query ahead of time. If you do, the query may run faster during subsequent executions.

The screenshot shows a SQL query editor interface. At the top, there is a toolbar with icons for Run, Undo, Publish, Query plan, and Connect to. The 'Connect to' dropdown is set to 'SQLPool01'. Below the toolbar, a code editor window displays the following SQL script:

```
1 SELECT
2     COUNT_BIG(*)
3 FROM
4     [wwi_poc].[Sale]
```

Below the code editor, there are tabs for 'Results' and 'Messages', with 'Results' being the active tab. Under 'View', the 'Table' option is selected. There is also a 'Chart' button and an 'Export results' dropdown. A search bar labeled 'Search' is present. The results section shows a single row of data:

(No column name)
981995895

6. In the query window, replace the script with the following (more complex) statement:

```
SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,SUM(S.TotalAmount) as TotalAmount
    FROM
        [wwi_poc].[Sale] S
    GROUP BY
        S.CustomerId
) T
OPTION (LABEL = 'Lab: Heap')
```

7. Select **Run** from the toolbar menu to execute the SQL command.



The script takes up to **60 seconds** to execute and returns the result. There is clearly something wrong with the **Sale_Heap** table that induces the performance hit.

If the script is still running after 90 seconds, click on Cancel.

The screenshot shows a SQL query editor interface. At the top, there are navigation buttons for Run, Undo, Publish, Query plan, and Connect to. The 'Connect to' dropdown is set to 'SQLPool01'. Below the toolbar, the query is displayed:

```

1  SELECT TOP 1000 * FROM
2  (
3      SELECT
4          S.CustomerId
5          ,SUM(S.TotalAmount) as TotalAmount
6      FROM
7          [wwi_poc].[Sale] S
8      GROUP BY
9          S.CustomerId
10 ) T
11 OPTION (LABEL = 'Lab: Heap')

```

Below the query, there are tabs for Results and Messages, with Results selected. Under Results, there are buttons for View, Table (which is selected), Chart, and Export results. A search bar is also present.

CustomerId	TotalAmount
105703	64408.68
489764	33954.76
941393	91254.13
954534	96395.03
863554	63202.45
228662	49787.00
614320	52678.57
316648	53469.57
876242	26022.77

At the bottom of the results pane, a message is displayed in a red-bordered box:

✓ 00:00:51 Query executed successfully.

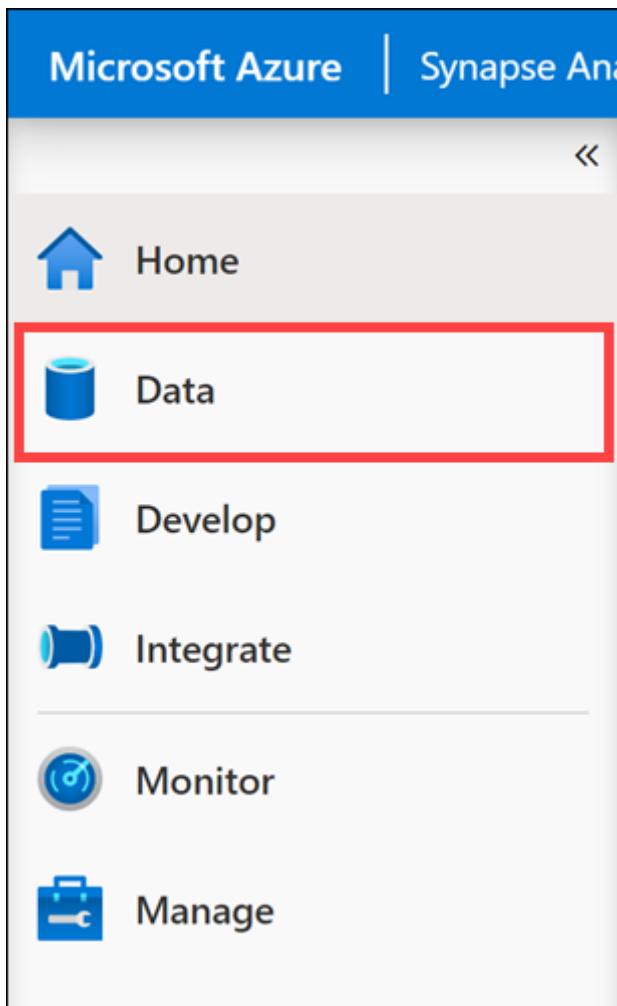
Note the `OPTION` clause used in the statement. This comes in handy when you're looking to identify your query in the `sys.dm_pdw_exec_requests` DMV.

```

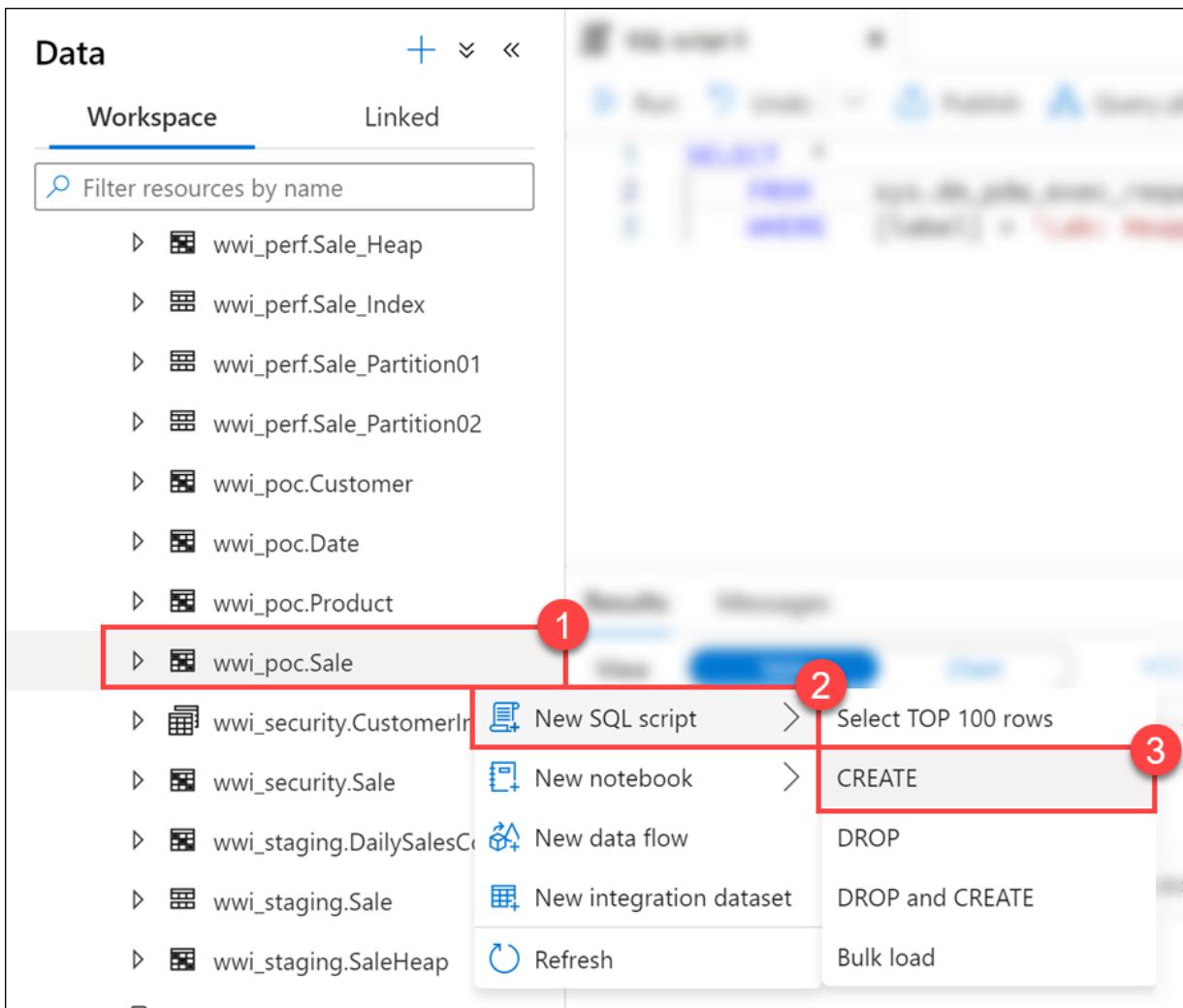
SELECT *
FROM sys.dm_pdw_exec_requests
WHERE [label] = 'Lab: Heap';

```

8. Select the **Data** hub.



9. Expand the **SQLPool01** database and its list of tables. Right-click **wwi_poc.Sale** (1), select **New SQL script** (2), then select **CREATE** (3).



10. Take a look at the script used to create the table:

```

CREATE TABLE [wwi_poc].[Sale]
(
    [TransactionId] [uniqueidentifier] NOT NULL,
    [CustomerId] [int] NOT NULL,
    [ProductId] [smallint] NOT NULL,
    [Quantity] [tinyint] NOT NULL,
    [Price] [decimal](9,2) NOT NULL,
    [TotalAmount] [decimal](9,2) NOT NULL,
    [TransactionDateId] [int] NOT NULL,
    [ProfitAmount] [decimal](9,2) NOT NULL,
    [Hour] [tinyint] NOT NULL,
    [Minute] [tinyint] NOT NULL,
    [StoreId] [smallint] NOT NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    HEAP
)

```

Note: *Do not run this script!* It is just for demonstration purposes to review the schema.

You can immediately spot at least two reasons for the performance hit:

- The ROUND_ROBIN distribution
- The HEAP structure of the table

NOTE

In this case, when we are looking for fast query response times, the heap structure is not a good choice as we will see in a moment. Still, there are cases where using a heap table can help performance rather than hurting it. One such example is when we're looking to ingest large amounts of data into the SQL database associated with the dedicated SQL pool.

If we were to review the query plan in detail, we would clearly see the root cause of the performance problem: inter-distribution data movements.

- Run the same script as the one you've run at step 2, but this time with the `EXPLAIN WITH_RECOMMENDATIONS` line before it:

```
EXPLAIN WITH_RECOMMENDATIONS
SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,SUM(S.TotalAmount) AS TotalAmount
    FROM
        [wwi_poc].[Sale] S
    GROUP BY
        S.CustomerId
) T
```

The `EXPLAIN WITH_RECOMMENDATIONS` clause returns the query plan for an Azure Synapse Analytics SQL statement without running the statement. Use `EXPLAIN` to preview which operations will require data movement and to view the estimated costs of the query operations. By default, you will get the execution plan in XML format, which you can export to other formats like CSV or JSON. **Do not** select `Query Plan` from the toolbar as it will try to download the query plan and open it in SQL Server Management Studio.

Your query should return something similar to:

```
<data><row><explain><?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="4" number_distributions="60" number_distributions_per_node="15">
<sql>SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,SUM(S.TotalAmount) AS TotalAmount
    FROM
        [wwi_poc].[Sale] S
    GROUP BY
        S.CustomerId
) T</sql>
<materialized_view_candidates>
    <materialized_view_candidates with_constants="False">CREATE MATERIALIZED VIEW View1 WITH (DIST
SELECT [S].[CustomerId] AS [Expr0]
    ,SUM([S].[TotalAmount]) AS [Expr1]
FROM [wwi_poc].[Sale] [S]
GROUP BY [S].[CustomerId]</materialized_view_candidates>
</materialized_view_candidates>
<dsql_operations total_cost="4.0656044" total_number_operations="5">
    <dsql_operation operation_type="RND_ID">
        <identifier>TEMP_ID_56</identifier>
    </dsql_operation>
    <dsql_operation operation_type="ON">
        <location permanent="false" distribution="AllDistributions" />
    <sql_operations>
        <sql_operation type="statement">CREATE TABLE [qtabledb].[dbo].[TEMP_ID_56] ([CustomerId] I
    </sql_operations>
    </dsql_operation>
    <dsql_operation operation_type="SHUFFLE_MOVE">
        <operation_cost cost="4.0656044" accumulative_cost="4.0656044" average_rowsize="13" output_rows
        <source_statement>SELECT [T1_1].[CustomerId] AS [CustomerId], [T1_1].[col] AS [col] FROM (SELE
```

```

OPTION (MAXDOP 4, MIN_GRANT_PERCENT = [MIN_GRANT], DISTRIBUTED_MOVE(N''))</source_statement>
<destination_table>[TEMP_ID_56]</destination_table>
<shuffle_columns>CustomerId;</shuffle_columns>
</dsql_operation>
<dsql_operation operation_type="RETURN">
<location distribution="AllDistributions" />
<select>SELECT [T1_1].[CustomerId] AS [CustomerId], [T1_1].[col] AS [col] FROM (SELECT TOP (CA
OPTION (MAXDOP 4, MIN_GRANT_PERCENT = [MIN_GRANT])</select>
</dsql_operation>
<dsql_operation operation_type="ON">
<location permanent="false" distribution="AllDistributions" />
<sql_operations>
    <sql_operation type="statement">DROP TABLE [qtabledb].[dbo].[TEMP_ID_56]</sql_operation>
</sql_operations>
</dsql_operation>
</dsql_operations>
</dsql_query></explain></row></data>

```

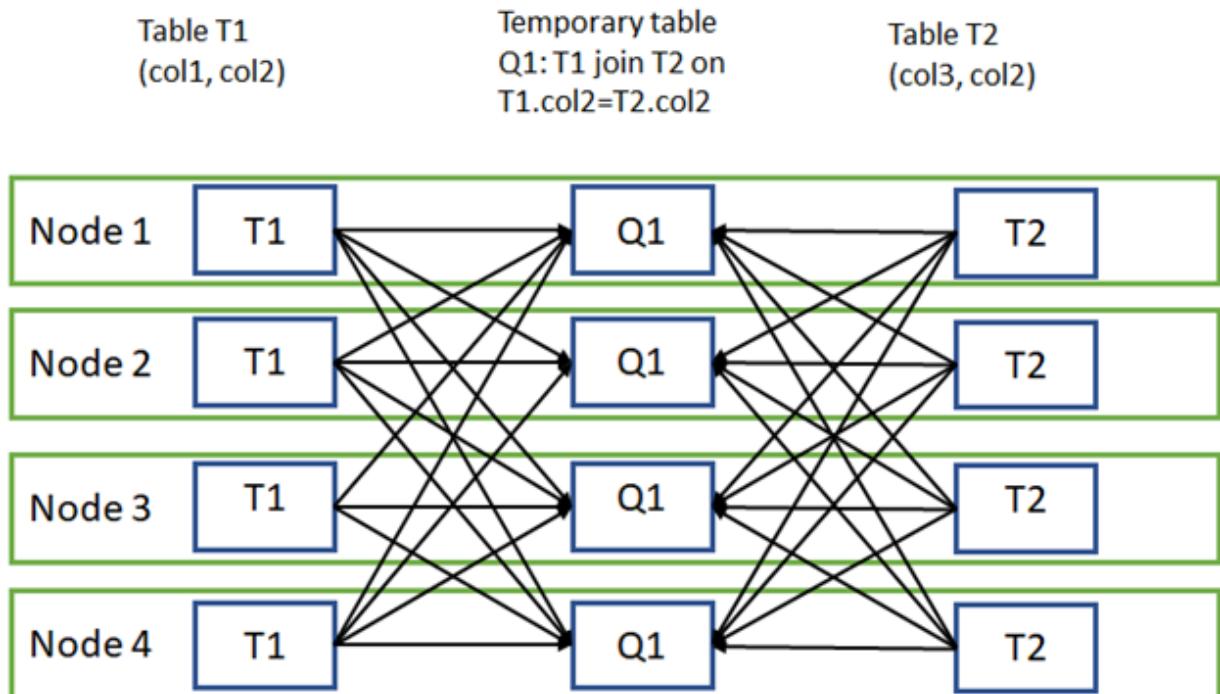
Notice the details of the internal layout of the MPP system:

```
<dsql_query number_nodes="4" number_distributions="60" number_distributions_per_node="15">
```

This layout is given by the current Date Warehouse Units (DWU) setting. In the setup used for the example above, we were running at DW2000c which means that there are 4 physical nodes to service the 60 distributions, giving a number of 15 distributions per physical node. Depending on your own DWU settings, these numbers will vary.

The query plan indicates data movement is required. This is indicated by the SHUFFLE_MOVE distributed SQL operation.

Data movement is an operation where parts of the distributed tables are moved to different nodes during query execution. This operation is required where the data is not available on the target node, most commonly when the tables do not share the distribution key. The most common data movement operation is shuffle. During shuffle, for each input row, Synapse computes a hash value using the join columns and then sends that row to the node that owns that hash value. Either one or both sides of join can participate in the shuffle. The diagram below displays shuffle to implement join between tables T1 and T2 where neither of the tables is distributed on the join column col2.



Let's dive now into the details provided by the query plan to understand some of the problems our current approach has. The following table contains the description of every operation mentioned in the query plan:

Operation	Operation Type	Description
1	RND_ID	Identifies an object that will be created. In our case, it's the TEMP_ID_76 internal table.
2	ON	Specifies the location (nodes or distributions) where the operation will occur. AllDistr...
3	SHUFFLE_MOVE	The list of shuffle columns contains only one column which is CustomerId (specified via <lo...
4	RETURN	Data resulting from the shuffle operation will be collected from all distributions (see <lo...
5	ON	The TEMP_ID_76 will be deleted from all distributions.

It becomes clear now what is the root cause of the performance problem: the inter-distribution data movements. This is actually one of the simplest examples given the small size of the data that needs to be shuffled. You can imagine how much worse things become when the shuffled row size becomes larger.

You can learn more about the structure of the query plan generated by the EXPLAIN statement [here](#).

- Besides the EXPLAIN statement, you can also understand the plan details using the sys.dm_pdw_request_steps DMV.

Query the sys.dm_pdw_exec_requests DMW to find your query id (this is for the query you executed previously at step 6):

```
SELECT
*
FROM
    sys.dm_pdw_exec_requests
WHERE
    [label] = 'Lab: Heap'
```

The result contains, among other things, the query id (Request_id), the label, and the original SQL statement:

Results	Messages										
request_id	session_id	status	submit_time	start_time	end_compile_time	end_time	total_elapsed_time	label	error_id	database_id	command
QID5418	SID366	Completed	2021-05-02T20...	2021-05-02T20...	2021-05-02T20...	2021-05-02T20...	50251	Lab: Heap	NULL	75	SELECT TOP 10...

- With the query id (QID5418 in this case, **substitute with your id**) you can now investigate the individual steps of the query:

```
SELECT
*
FROM
    sys.dm_pdw_request_steps
WHERE
    request_id = 'QID5418'
ORDER BY
    step_index
```

The steps (indexed 0 to 4) are matching operations 2 to 6 from the query plan. Again, the culprit stands out: the step with index 2 describes the inter-partition data movement operation. By looking at the TOTAL_ELAPSED_TIME column one can clearly tell the largest part of the query time is generated by this step. **Take note of the step index** for the next query.

Results	Messages										
request_id	step_index	plan_node_id	operation_type	distribution_type	location_type	status	error_id	start_time	end_time	total_elapsed_time	row_count
QID5418	0	-1	RandomIDOper...	Unspecified	Control	Complete	NULL	2021-05-02T20...	2021-05-02T20...	0	-1
QID5418	1	-1	OnOperation	AllDistributions	Compute	Complete	NULL	2021-05-02T20...	2021-05-02T20...	203	-1
QID5418	2	9	ShuffleMoveOp...	AllDistributions	Compute	Complete	NULL	2021-05-02T20...	2021-05-02T20...	44392	58798030
QID5418	3	23	ReturnOperation	AllDistributions	Compute	Complete	NULL	2021-05-02T20...	2021-05-02T20...	5375	1000
QID5418	4	-1	OnOperation	AllDistributions	Compute	Complete	NULL	2021-05-02T20...	2021-05-02T20...	234	-1

- Get more details on the problematic step using the following SQL statement (substitute the request_id and step_index with your values):

```

SELECT
*
FROM
    sys.dm_pdw_sql_requests
WHERE
    request_id = 'QID5418'
    AND step_index = 2

```

The results of the statement provide details about what happens on each distribution within the SQL pool.

request_id	step_index	pdw_node_id	distribution_id	status	error_id	start_time	end_time	total_elapsed_t...	row_count	spid	command
QID5418	2	28	1	Complete	NULL	2021-05-02T20...	2021-05-02T20...	34954	-1	691	EXEC sp_set_distributed_query_c...
QID5418	2	28	2	Complete	NULL	2021-05-02T20...	2021-05-02T20...	42454	-1	833	EXEC sp_set_distributed_query_c...
QID5418	2	28	3	Complete	NULL	2021-05-02T20...	2021-05-02T20...	39829	-1	444	EXEC sp_set_distributed_query_c...
QID5418	2	28	4	Complete	NULL	2021-05-02T20...	2021-05-02T20...	40845	-1	617	EXEC sp_set_distributed_query_c...
QID5418	2	28	5	Complete	NULL	2021-05-02T20...	2021-05-02T20...	43485	-1	794	EXEC sp_set_distributed_query_c...
QID5418	2	28	6	Complete	NULL	2021-05-02T20...	2021-05-02T20...	35876	-1	673	EXEC sp_set_distributed_query_c...
QID5418	2	28	7	Complete	NULL	2021-05-02T20...	2021-05-02T20...	35517	-1	820	EXEC sp_set_distributed_query_c...
QID5418	2	28	8	Complete	NULL	2021-05-02T20...	2021-05-02T20...	43564	-1	367	EXEC sp_set_distributed_query_c...
QID5418	2	28	9	Complete	NULL	2021-05-02T20...	2021-05-02T20...	35673	-1	502	EXEC sp_set_distributed_query_c...
QID5418	2	28	10	Complete	NULL	2021-05-02T20...	2021-05-02T20...	39360	-1	184	EXEC sp_set_distributed_query_c...

15. Finally, you can use the following SQL statement to investigate data movement on the distributed databases (substitute the `request_id` and `step_index` with your values):

```

SELECT
*
FROM
    sys.dm_pdw_dms_workers
WHERE
    request_id = 'QID5418'
    AND step_index = 2
ORDER BY
    distribution_id

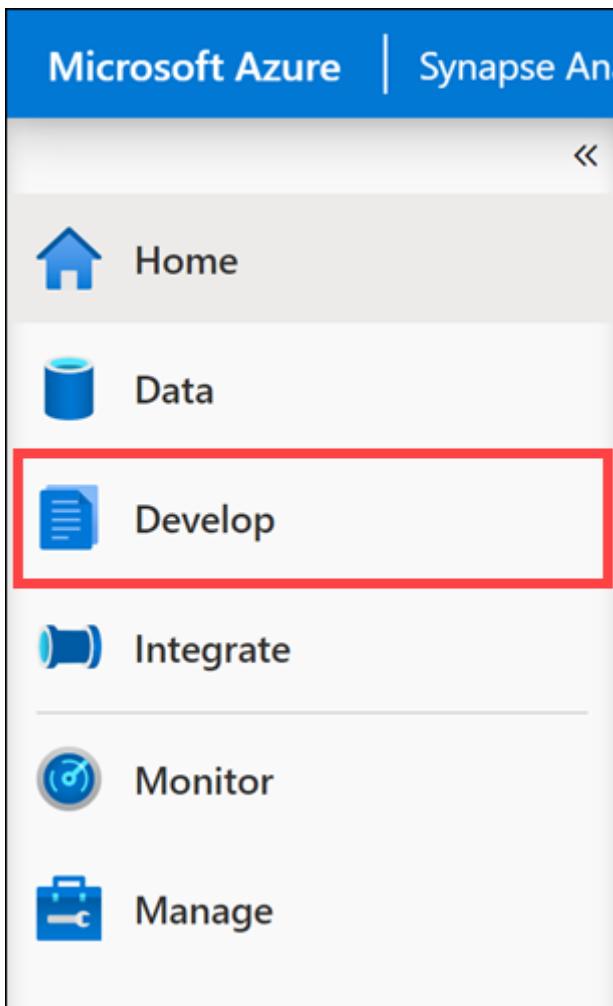
```

The results of the statement provide details about data being moved at each distribution. The `ROWS_PROCESSED` column is especially useful here to get an estimate of the magnitude of the data movement happening when the query is executed.

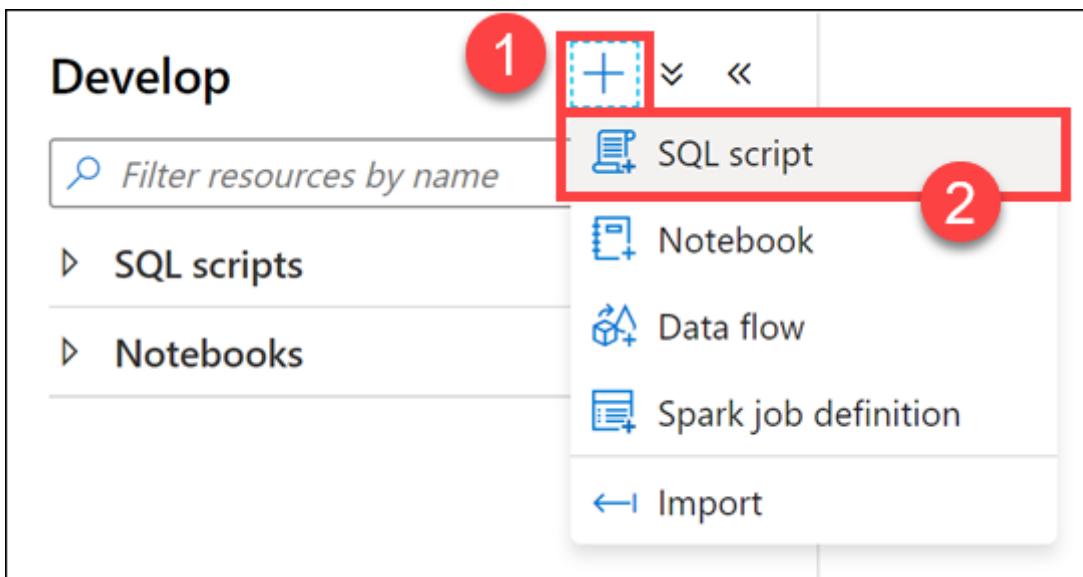
request_id	step_index	dms_step_index	pdw_node_id	distribution_id	type	status	bytes_per_sec	bytes_processed	rows_processed	start_time	end_time	total_elapsed_t...	cpu_time	query_time
QID5418	2	NULL	28	1	HASH_READER	StepComplete	192045	6702403	980028	2021-05-02T20...	2021-05-02T20...	34900	NULL	31887
QID5418	2	NULL	28	1	WRITER	NULL	55732	6652588	973343	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	2	WRITER	NULL	56990	6802807	993749	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	2	HASH_READER	StepComplete	157963	6702844	979963	2021-05-02T20...	2021-05-02T20...	42433	NULL	39553
QID5418	2	NULL	28	3	HASH_READER	StepComplete	168317	6702407	979988	2021-05-02T20...	2021-05-02T20...	39820	NULL	37417
QID5418	2	NULL	28	3	WRITER	NULL	56308	6721370	983356	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	4	WRITER	NULL	56250	6714463	982278	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	4	HASH_READER	StepComplete	164223	6702443	980053	2021-05-02T20...	2021-05-02T20...	40813	NULL	39443
QID5418	2	NULL	28	5	HASH_READER	StepComplete	154366	6703518	980154	2021-05-02T20...	2021-05-02T20...	43426	NULL	42043
QID5418	2	NULL	28	5	WRITER	NULL	56000	6684633	978165	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	6	WRITER	NULL	56172	6705161	981348	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	6	HASH_READER	StepComplete	186838	6703749	980087	2021-05-02T20...	2021-05-02T20...	35880	NULL	33013
QID5418	2	NULL	28	7	HASH_READER	StepComplete	189107	6700636	979696	2021-05-02T20...	2021-05-02T20...	35433	NULL	32956
QID5418	2	NULL	28	7	WRITER	NULL	56236	6712779	981045	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	8	WRITER	NULL	55901	6672803	976718	2021-05-02T20...	2021-05-02T20...	3214	NULL	43023
QID5418	2	NULL	28	8	HASH_READER	StepComplete	153798	6700525	979691	2021-05-02T20...	2021-05-02T20...	43567	NULL	42317
QID5418	3	NULL	28	9	HASH_READER	StepComplete	187010	6704448	980000	2021-05-02T20...	2021-05-02T20...	35663	NULL	33013

12.6.2 Task 2: Improve table structure with hash distribution and columnstore index

- Select the Develop hub.



2. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



3. In the toolbar menu, connect to the **SQLPool01** database to execute the query.



4. In the query window, replace the script with the following to create an improved version of the table using CTAS (Create Table As Select):

```
CREATE TABLE [wwi_perf].[Sale_Hash]
WITH
```

```

(
    DISTRIBUTION = HASH ( [CustomerId] ),
    CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT
*
FROM
[wwi_poc].[Sale]

```

5. Select **Run** from the toolbar menu to execute the SQL command.



The query will take around **10 minutes** to complete. While this is running, read the rest of the lab instructions to familiarize yourself with the content.

NOTE

CTAS is a more customizable version of the SELECT...INTO statement. SELECT...INTO doesn't allow you to change either the distribution method or the index type as part of the operation. You create the new table by using the default distribution type of ROUND_ROBIN, and the default table structure of CLUSTERED COLUMNSTORE INDEX.

With CTAS, on the other hand, you can specify both the distribution of the table data as well as the table structure type.

6. In the query window, replace the script with the following to see performance improvements:

```

SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,SUM(S.TotalAmount) AS TotalAmount
    FROM
        [wwi_perf].[Sale_Hash] S
    GROUP BY
        S.CustomerId
) T

```

7. Select **Run** from the toolbar menu to execute the SQL command.



You should see a performance improvement executing against the new Hash table compared to the first time we ran the script against the Heap table. In our case, the query executed in about 8 seconds.

Results Messages

View Table **Chart** Export results ▾

Search

CustomerId	TotalAmount
350763	77364.89
865644	60326.00
948822	66208.58
183569	52247.55
9932	82073.40
314574	64704.83
446476	68789.38
24502	117986.33
314472	45026.04

00:00:08 Query executed successfully.

8. Run the following EXPLAIN statement again to get the query plan (do not select **Query Plan** from the toolbar as it will try do download the query plan and open it in SQL Server Management Studio):

```
EXPLAIN
SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,SUM(S.TotalAmount) AS TotalAmount
    FROM
        [wwi_perf].[Sale_Hash] S
    GROUP BY
        S.CustomerId
) T
```

The resulting query plan is clearly much better than the previous one, as there is no more inter-distribution data movement involved.

```
<data><row><explain><?xml version="1.0" encoding="utf-8"?>
```

```

<dsql_query number_nodes="5" number_distributions="60" number_distributions_per_node="12">
<sql>SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,SUM(S.TotalAmount) as TotalAmount
    FROM
        [wwi_perf].[Sale_Hash] S
    GROUP BY
        S.CustomerId
) T</sql>
<dsql_operations total_cost="0" total_number_operations="1">
    <dsql_operation operation_type="RETURN">
        <location distribution="AllDistributions" />
        <select>SELECT [T1_1].[CustomerId] AS [CustomerId], [T1_1].[col] AS [col] FROM (SELECT TOP (CAST(OPTION (MAXDOP 4)</select>
        </dsql_operation>
    </dsql_operations>
</dsql_query></explain></row></data>

```

9. Try running a more complex query and investigate the execution plan and execution steps. Here is an example of a more complex query you can use:

```

SELECT
    AVG(TotalProfit) as AvgMonthlyCustomerProfit
FROM
(
    SELECT
        S.CustomerId
        ,D.Year
        ,D.Month
        ,SUM(S.TotalAmount) as TotalAmount
        ,AVG(S.TotalAmount) as AvgAmount
        ,SUM(S.ProfitAmount) as TotalProfit
        ,AVG(S.ProfitAmount) as AvgProfit
    FROM
        [wwi_perf].[Sale_Partition01] S
        join [wwi].[Date] D on
            D.DateId = S.TransactionDateId
    GROUP BY
        S.CustomerId
        ,D.Year
        ,D.Month
) T

```

12.6.3 Task 4: Improve further the table structure with partitioning

Table partitions enable you to divide your data into smaller groups of data. Partitioning can benefit data maintenance and query performance. Whether it benefits both or just one is dependent on how data is loaded and whether the same column can be used for both purposes, since partitioning can only be done on one column.

Date columns are usually good candidates for partitioning tables at the distributions level. In the case of Tailwind Trader's sales data, partitioning based on the `TransactionDateId` column seems to be a good choice.

The dedicated SQL pool already contains two versions of the `Sale` table that have been partitioned using `TransactionDateId`. These tables are `[wwi_perf].[Sale_Partition01]` and `[wwi_perf].[Sale_Partition02]`. Below are the CTAS queries that have been used to create these tables.

1. In the query window, replace the script with the following CTAS queries that create the partition tables (**do not execute**):

```

CREATE TABLE [wwi_perf].[Sale_Partition01]
WITH
(

```

```

DISTRIBUTION = HASH ( [CustomerId] ),
CLUSTERED COLUMNSTORE INDEX,
PARTITION
(
    [TransactionDateId] RANGE RIGHT FOR VALUES (
        20190101, 20190201, 20190301, 20190401, 20190501, 20190601, 20190701, 20190801, 20190901
    )
)
AS
SELECT
*
FROM
[wwi_perf].[Sale_Heap]
OPTION (LABEL = 'CTAS : Sale_Partition01')

CREATE TABLE [wwi_perf].[Sale_Partition02]
WITH
(
    DISTRIBUTION = HASH ( [CustomerId] ),
    CLUSTERED COLUMNSTORE INDEX,
    PARTITION
    (
        [TransactionDateId] RANGE RIGHT FOR VALUES (
            20190101, 20190401, 20190701, 20191001
        )
    )
)
AS
SELECT *
FROM
[wwi_perf].[Sale_Heap]
OPTION (LABEL = 'CTAS : Sale_Partition02')

```

Note

These queries have already been run on the dedicated SQL pool. **Do not** execute the script.

Notice the two partitioning strategies we've used here. The first partitioning scheme is month-based and the second is quarter-based (**3**).

```

1 CREATE TABLE [wwi_perf].[Sale_Partition01]
2 WITH
3 (
4     DISTRIBUTION = HASH ( [CustomerId] ), 1
5     CLUSTERED COLUMNSTORE INDEX, 2
6     PARTITION
7     (
8         [TransactionDateId] RANGE RIGHT FOR VALUES (
9             20190101, 20190201, 20190301, 20190401, 20190501, 20190601, 20190701, 20190801, 20190901, 20191001, 20191101, 20191201)
10    )
11 )
12 AS
13 SELECT
14     *
15 FROM
16     [wwi_perf].[Sale_Heap]
17 OPTION (LABEL = 'CTAS : Sale_Partition01')
18
19 CREATE TABLE [wwi_perf].[Sale_Partition02]
20 WITH
21 (
22     DISTRIBUTION = HASH ( [CustomerId] ), 1
23     CLUSTERED COLUMNSTORE INDEX, 2
24     PARTITION
25     (
26         [TransactionDateId] RANGE RIGHT FOR VALUES (
27             20190101, 20190401, 20190701, 20191001)
28    )
29 )
30 AS
31 SELECT *
32 FROM
33     [wwi_perf].[Sale_Heap]
34 OPTION (LABEL = 'CTAS : Sale Partition02')

```

12.6.3.1 Task 4.1: Table distributions

As you can see, the two partitioned tables are hash-distributed (1). A distributed table appears as a single table, but the rows are actually stored across 60 distributions. The rows are distributed with a hash or round-robin algorithm.

The types of distributions are:

- **Round-robin distributed:** Distributes table rows evenly across all distributions at random.
- **Hash distributed:** Distributes table rows across the Compute nodes by using a deterministic hash function to assign each row to one distribution.
- **Replicated:** Full copy of table accessible on each Compute node.

A hash-distributed table distributes table rows across the Compute nodes by using a deterministic hash function to assign each row to one distribution.

Since identical values always hash to the same distribution, the data warehouse has built-in knowledge of the row locations.

Dedicated SQL pool uses this knowledge to minimize data movement during queries, which improves query performance. Hash-distributed tables work well for large fact tables in a star schema. They can have very large numbers of rows and still achieve high performance. There are, of course, some design considerations that help you to get the performance the distributed system is designed to provide.

Consider using a hash-distributed table when:

- The table size on disk is more than 2 GB.
- The table has frequent insert, update, and delete operations.

12.6.3.2 Task 4.2: Indexes

Looking at the query, also notice that both partitioned tables are configured with a **clustered columnstore index** (2). There are different types of indexes you can use in dedicated SQL pool:

- **Clustered Columnstore index (Default Primary):** Offers the highest level of data compression and best overall query performance.
- **Clustered index (Primary):** Is performant for looking up a single to few rows.
- **Heap (Primary):** Benefits from faster loading and landing temporary data. It is best for small lookup tables.

- **Nonclustered indexes (Secondary)**: Enable ordering of multiple columns in a table and allows multiple nonclustered on a single table. These can be created on any of the above primary indexes and offer more performant lookup queries.

By default, dedicated SQL pool creates a clustered columnstore index when no index options are specified on a table. Clustered columnstore tables offer both the highest level of data compression as well as the best overall query performance. They will generally outperform clustered index or heap tables and are usually the best choice for large tables. For these reasons, clustered columnstore is the best place to start when you are unsure of how to index your table.

There are a few scenarios where clustered columnstore may not be a good option:

- Columnstore tables do not support `varchar(max)`, `nvarchar(max)`, and `varbinary(max)`. Consider heap or clustered index instead.
- Columnstore tables may be less efficient for transient data. Consider heap and perhaps even temporary tables.
- Small tables with less than 100 million rows. Consider heap tables.

12.6.3.3 Task 4.3: Partitioning

Again, with this query, we partition the two tables differently (3) so we can evaluate the performance difference and decide which partitioning strategy is best long-term. The one we ultimately go with depends on various factors with Tailwind Trader's data. You may decide to keep both to optimize query performance, but then you double the data storage and maintenance requirements for managing the data.

Partitioning is supported on all table types.

The **RANGE RIGHT** option that we use in the query (3) is used for time partitions. RANGE LEFT is used for number partitions.

The primary benefits to partitioning is that it:

- Improves efficiency and performance of loading and querying by limiting the scope to a subset of data.
- Offers significant query performance enhancements where filtering on the partition key can eliminate unnecessary scans and eliminate I/O (input/output operations).

The reason we have created two tables with different partition strategies (3) is to experiment with proper sizing.

While partitioning can be used to improve performance, creating a table with too many partitions can hurt performance under some circumstances. These concerns are especially true for clustered columnstore tables, like we created here. For partitioning to be helpful, it is important to understand when to use partitioning and the number of partitions to create. There is no hard and fast rule as to how many partitions are too many, it depends on your data and how many partitions you are loading simultaneously. A successful partitioning scheme usually has tens to hundreds of partitions, not thousands.

Supplemental information:

When creating partitions on clustered columnstore tables, it is important to consider how many rows belong to each partition. For optimal compression and performance of clustered columnstore tables, a minimum of 1 million rows per distribution and partition is needed. Before partitions are created, dedicated SQL pools already divides each table into 60 distributed databases. Any partitioning added to a table is in addition to the distributions created behind the scenes. Using this example, if the sales fact table contained 36 monthly partitions, and given that dedicated SQL pool has 60 distributions, then the sales fact table should contain 60 million rows per month, or 2.1 billion rows when all months are populated. If a table contains fewer than the recommended minimum number of rows per partition, consider using fewer partitions in order to increase the number of rows per partition.

12.7 Exercise 4: Improve query performance

12.7.1 Task 1: Use materialized views

As opposed to a standard view, a materialized view pre-computes, stores, and maintains its data in a dedicated SQL pool just like a table. Here is a basic comparison between standard and materialized views:

Comparison	View	Materialized View
View definition	Stored in Synapse Analytics.	Stored in Synapse Analytics.
View content	Generated each time when the view is used.	Pre-processed and stored.

Comparison	View	Materialized View
Data refresh	Always updated	Always updated
Speed to retrieve view data from complex queries	Slow	Fast
Extra storage	No	Yes
Syntax	CREATE VIEW	CREATE MATERIALIZED VIEW

1. Execute the following query to get an approximation of its execution time:

```
SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,D.Year
        ,D.Quarter
        ,SUM(S.TotalAmount) as TotalAmount
    FROM
        [wwi_perf].[Sale_Partition02] S
        join [wwi].[Date] D on
            S.TransactionDateId = D.DateId
    GROUP BY
        S.CustomerId
        ,D.Year
        ,D.Quarter
) T
```

2. Execute this query as well (notice the slight difference):

```
SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,D.Year
        ,D.Month
        ,SUM(S.ProfitAmount) as TotalProfit
    FROM
        [wwi_perf].[Sale_Partition02] S
        join [wwi].[Date] D on
            S.TransactionDateId = D.DateId
    GROUP BY
        S.CustomerId
        ,D.Year
        ,D.Month
) T
```

3. Create a materialized view that can support both queries above:

```
CREATE MATERIALIZED VIEW
    wwi_perf.mvCustomerSales
WITH
(
    DISTRIBUTION = HASH( CustomerId )
)
AS
SELECT
    S.CustomerId
    ,D.Year
    ,D.Quarter
    ,D.Month
    ,SUM(S.TotalAmount) as TotalAmount
    ,SUM(S.ProfitAmount) as TotalProfit
FROM
    [wwi_perf].[Sale_Partition02] S
```

```

join [wwi].[Date] D on
    S.TransactionDateId = D.DateId
GROUP BY
    S.CustomerId
    ,D.Year
    ,D.Quarter
    ,D.Month

```

4. Run the following query to get an estimated execution plan (do not select **Query Plan** from the toolbar as it will try to download the query plan and open it in SQL Server Management Studio):

```

EXPLAIN
SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,D.Year
        ,D.Quarter
        ,SUM(S.TotalAmount) as TotalAmount
    FROM
        [wwi_perf].[Sale_Partition02] S
        join [wwi].[Date] D on
            S.TransactionDateId = D.DateId
    GROUP BY
        S.CustomerId
        ,D.Year
        ,D.Quarter
) T

```

The resulting execution plan shows how the newly created materialized view is used to optimize the execution. Note the `FROM [SQLPool01].[wwi_perf].[mvCustomerSales]` in the `<dsql_operations>` element.

```

<?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="5" number_distributions="60" number_distributions_per_node="12">
<sql>SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,D.Year
        ,D.Quarter
        ,SUM(S.TotalAmount) as TotalAmount
    FROM
        [wwi_perf].[Sale_Partition02] S
        join [wwi].[Date] D on
            S.TransactionDateId = D.DateId
    GROUP BY
        S.CustomerId
        ,D.Year
        ,D.Quarter
) T</sql>
<dsql_operations total_cost="0" total_number_operations="1">
    <dsql_operation operation_type="RETURN">
        <location distribution="AllDistributions" />
        <select>SELECT [T1_1].[CustomerId] AS [CustomerId], [T1_1].[Year] AS [Year], [T1_1].[Quarter]
OPTION (MAXDOP 6)</select>
    </dsql_operation>
</dsql_operations>
</dsql_query>

```

5. The same materialized view is also used to optimize the second query. Get its execution plan:

```

EXPLAIN
SELECT TOP 1000 * FROM

```

```

(
    SELECT
        S.CustomerId
        ,D.Year
        ,D.Month
        ,SUM(S.ProfitAmount) as TotalProfit
    FROM
        [wwi_perf].[Sale_Partition02] S
        join [wwi].[Date] D on
            S.TransactionDateId = D.DateId
    GROUP BY
        S.CustomerId
        ,D.Year
        ,D.Month
) T

```

The resulting execution plan shows the use of the same materialized view to optimize execution:

```

<?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="5" number_distributions="60" number_distributions_per_node="12">
<sql>SELECT TOP 1000 * FROM
(
    SELECT
        S.CustomerId
        ,D.Year
        ,D.Month
        ,SUM(S.ProfitAmount) as TotalProfit
    FROM
        [wwi_perf].[Sale_Partition02] S
        join [wwi].[Date] D on
            S.TransactionDateId = D.DateId
    GROUP BY
        S.CustomerId
        ,D.Year
        ,D.Month
) T</sql>
<dsql_operations total_cost="0" total_number_operations="1">
    <dsql_operation operation_type="RETURN">
        <location distribution="AllDistributions" />
        <select>SELECT [T1_1].[CustomerId] AS [CustomerId], [T1_1].[Year] AS [Year], [T1_1].[Month] AS
OPTION (MAXDOP 6)</select>
    </dsql_operation>
</dsql_operations>
</dsql_query>

```

Note

Even if the two queries have different aggregation levels, the query optimizer is able to infer the use of the materialized view. This happens because the materialized view covers both aggregation levels (**Quarter** and Month) as well as both aggregation measures (**TotalAmount** and **ProfitAmount**).

6. Check the materialized view overhead:

```
DBCC PDW_SHOWMATERIALIZEDVIEWOVERHEAD ( 'wwi_perf.mvCustomerSales' )
```

The results show that **BASE_VIEW_ROWS** are equal to **TOTAL_ROWS** (and hence **OVERHEAD_RATIO** is 1). The materialized view is perfectly aligned with the base view. This situation is expected to change once the underlying data starts to change.

7. Update the original data the materialized view was built on:

```

UPDATE
    [wwi_perf].[Sale_Partition02]
SET

```

```

    TotalAmount = TotalAmount * 1.01
    ,ProfitAmount = ProfitAmount * 1.01

```

```

WHERE
    CustomerId BETWEEN 100 and 200

```

8. Check the materialized view overhead again:

```
DBCC PDW_SHOWMATERIALIZEDVIEWOVERHEAD ( 'wwi_perf.mvCustomerSales' )
```

Results	Messages		
View	Table	Chart	Export results
<input type="text"/> Search			
OBJECT_ID	BASE_VIEW_ROWS	TOTAL_ROWS	OVERHEAD_RATIO
544720993	118109924	118121863	1.0001010838054499

There is now a delta stored by the materialized view which results in TOTAL_ROWS being greater than BASE_VIEW_ROWS and OVERHEAD_RATIO being greater than 1.

9. Rebuild the materialized view and check that the overhead ration went back to 1:

```
ALTER MATERIALIZED VIEW [wwi_perf].[mvCustomerSales] REBUILD
```

```
DBCC PDW_SHOWMATERIALIZEDVIEWOVERHEAD ( 'wwi_perf.mvCustomerSales' )
```

Results	Messages		
View	Table	Chart	Export results
<input type="text"/> Search			
OBJECT_ID	BASE_VIEW_ROWS	TOTAL_ROWS	OVERHEAD_RATIO
544720993	118109924	118109924	1.0000000000000000

12.7.2 Task 2: Use result set caching

Tailwind Trader's downstream reports are used by many users, which often means the same query is being executed repeatedly against data that does not change often. What can they do to improve the performance of these types of queries? How does this approach work when the underlying data changes?

They should consider result-set caching.

Cache the results of a query in the dedicated Azure Synapse SQL pool storage. This enables interactive response times for repetitive queries against tables with infrequent data changes.

The result-set cache persists even if dedicated SQL pool is paused and resumed later.

Query cache is invalidated and refreshed when the underlying table data or query code changes.

Result cache is evicted regularly based on a time-aware least recently used algorithm (TLRU).

- In the query window, replace the script with the following to check if result set caching is on in the current dedicated SQL pool:

```

SELECT
    name
    ,is_result_set_caching_on
FROM
    sys.databases

```

- Select **Run** from the toolbar menu to execute the SQL command.



Look at the output of the query. What is the `is_result_set_caching_on` value for `SQLPool01`? In our case, it is set to `False`, meaning result set caching is currently disabled.

```

1  SELECT
2      name
3      ,is_result_set_caching_on
4  FROM
5      sys.databases

```

Results Messages

View Table Chart Export results

Search

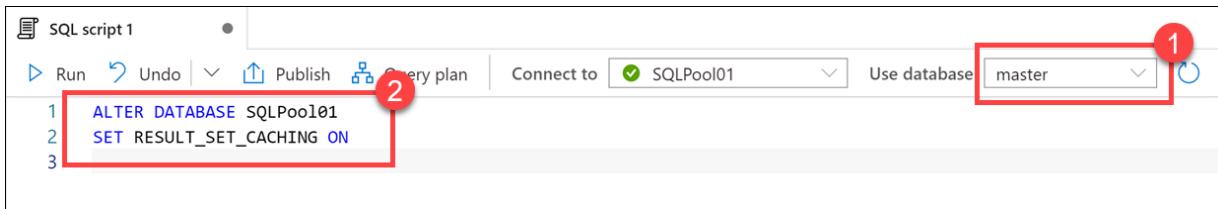
Name	Is_result_set_caching_on
master	False
SQLPool01	False

- In the query window, change the database to **master** (1), then replace the script (2) with the following to activate result set caching:

```

ALTER DATABASE SQLPool01
SET RESULT_SET_CACHING ON

```



- Select **Run** from the toolbar menu to execute the SQL command.



Important

The operations to create a result set cache and retrieve data from the cache happen on the control node of a dedicated SQL pool instance. When result set caching is turned ON, running queries that return a large result set (for example, >1GB) can cause high throttling on the control node and slow down the overall query response on the instance. Those queries are commonly used during data exploration or ETL operations. To avoid stressing the control node and cause performance issue, users should turn OFF result set caching on the database before running those types of queries.

- In the toolbar menu, connect to the **SQLPool01** database for the next query.



- In the query window, replace the script with the following query and immediately check if it hit the cache:

```

SELECT
    D.Year
    ,D.Quarter
    ,D.Month
    ,SUM(S.TotalAmount) as TotalAmount
    ,SUM(S.ProfitAmount) as TotalProfit
FROM
    [wwi_perf].[Sale_Partition02] S
    join [wwi].[Date] D on
        S.TransactionDateId = D.DateId
GROUP BY
    D.Year
    ,D.Quarter
    ,D.Month
OPTION (LABEL = 'Lab: Result set caching')

SELECT
    result_cache_hit
FROM
    sys.dm_pdw_exec_requests
WHERE
    request_id =
    (
        SELECT TOP 1
            request_id
        FROM
            sys.dm_pdw_exec_requests
        WHERE
            [label] = 'Lab: Result set caching'
        ORDER BY
            start_time desc
    )
)

```

7. Select **Run** from the toolbar menu to execute the SQL command.



As expected, the result is **False (0)**.

Result_cache_hit	0

Still, you can identify that, while running the query, dedicated SQL pool has also cached the result set.

8. In the query window, replace the script with the following to get the execution steps:

```

SELECT
    step_index
    ,operation_type
    ,location_type
    ,status
    ,total_elapsed_time
    ,command
FROM
    sys.dm_pdw_request_steps
WHERE
    request_id =
    (
        SELECT TOP 1
            request_id
        FROM
            sys.dm_pdw_exec_requests
        WHERE
            [label] = 'Lab: Result set caching'
        ORDER BY
            start_time desc
    )

```

9. Select **Run** from the toolbar menu to execute the SQL command.



The execution plan reveals the building of the result set cache:

Step_index	Operation_type	Location_type	Status	Total_elapsed_time	Command
0	RandomIDOperation	Control	Complete	0	TEMP_ID_21
1	OnOperation	Compute	Complete	31	CREATE TABLE [qtabledb].[dbo].[TEMP_ID_21] ([Dateld] INT NOT NULL, [Month] TI...
2	BroadcastMoveOperation	Compute	Complete	78	SELECT [T1_1].[Dateld] AS [Dateld], [T1_1].[Month] AS [Month], [T1_1].[Quarter] AS...
3	RandomIDOperation	Control	Complete	0	TEMP_ID_22
4	OnOperation	Compute	Complete	171	CREATE TABLE [qtabledb].[dbo].[TEMP_ID_22] ([Month] TINYINT NOT NULL, [Quar...
5	ShuffleMoveOperation	Compute	Complete	4156	SELECT [T1_1].[Month] AS [Month], [T1_1].[Quarter] AS [Quarter], [T1_1].[Year] AS [...
6	PartitionMoveOperation	Compute	Complete	187	SELECT [T1_1].[Year] AS [Year], [T1_1].[Quarter] AS [Quarter], [T1_1].[Month] AS [M...
7	ReturnOperation	Control	Complete	218	select * from [DWResultCacheDb].[dbo].[iq_{20F95277-EA59-41D0-8E3E-C36F11F5...
8	OnOperation	Compute	Complete	109	DROP TABLE [qtabledb].[dbo].[TEMP_ID_22]
9	OnOperation	Compute	Complete	31	DROP TABLE [qtabledb].[dbo].[TEMP_ID_21]

You can control at the user session level the use of the result set cache.

10. In the query window, replace the script with the following to deactivate and activate the result cache:

```
SET RESULT_SET_CACHING OFF
```

```

SELECT
    D.Year
    ,D.Quarter
    ,D.Month
    ,SUM(S.TotalAmount) as TotalAmount
    ,SUM(S.ProfitAmount) as TotalProfit
FROM
    [wwi_perf].[Sale_Partition02] S
    join [wwi].[Date] D on
        S.TransactionDateId = D.DateId
GROUP BY

```

```

D.Year
,D.Quarter
,D.Month
OPTION (LABEL = 'Lab: Result set caching off')

SET RESULT_SET_CACHING ON

SELECT
    D.Year
    ,D.Quarter
    ,D.Month
    ,SUM(S.TotalAmount) as TotalAmount
    ,SUM(S.ProfitAmount) as TotalProfit
FROM
    [wwi_perf].[Sale_Partition02] S
    join [wwi].[Date] D on
        S.TransactionDateId = D.DateId
GROUP BY
    D.Year
    ,D.Quarter
    ,D.Month
OPTION (LABEL = 'Lab: Result set caching on')

SELECT TOP 2
    request_id
    ,[label]
    ,result_cache_hit
FROM
    sys.dm_pdw_exec_requests
WHERE
    [label] in ('Lab: Result set caching off', 'Lab: Result set caching on')
ORDER BY
    start_time desc

```

11. Select **Run** from the toolbar menu to execute the SQL command.



The result of `SET RESULT_SET_CACHING OFF` in the script above is visible in the cache hit test results (The `result_cache_hit` column returns 1 for cache hit, 0 for cache miss, and *negative values* for reasons why result set caching was not used.):

Results		Messages			
Select	Query 3	View	Table	Chart	Export results
<input type="text"/> Search					
Request_id	Label	Result_cache_hit			
QID3985	Lab03: Result set caching on	1			
QID3983	Lab03: Result set caching off	-2			

12. In the query window, replace the script with the following to check the space used by the result cache:

```
DBCC SHOWRESULTCACHESPACEUSED
```

13. Select **Run** from the toolbar menu to execute the SQL command.



We can see the amount of space reserved, how much is used by data, the amount used for the index, and how much unused space there is for the result cache in the query results.

1 DBCC SHOWRESULTCACHESPACEUSED			
RESERVED_SPACE	DATA_SPACE	INDEX_SPACE	UNUSED_SPACE
20608	18488	1424	696

14. In the query window, replace the script with the following to clear the result set cache:

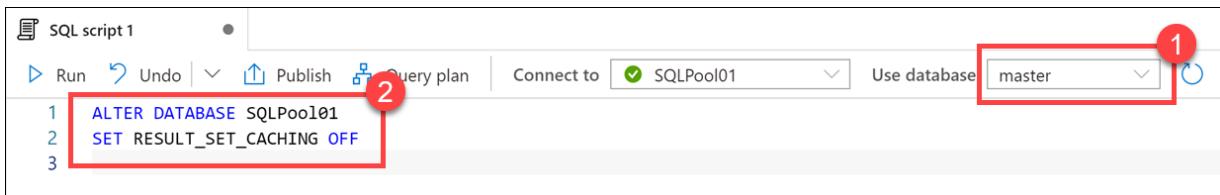
```
DBCC DROPRERESULTSETCACHE
```

15. Select **Run** from the toolbar menu to execute the SQL command.



16. In the query window, change the database to **master** (1), then replace the script (2) with the following to disable result set caching:

```
ALTER DATABASE SQLPool01
SET RESULT_SET_CACHING OFF
```



17. Select **Run** from the toolbar menu to execute the SQL command.



Note

Make sure you disable result set caching on the dedicated SQL pool. Failing to do so will have a negative impact on the remainder of the demos, as it will skew execution times and defeat the purpose of several upcoming exercises.

The maximum size of result set cache is 1 TB per database. The cached results are automatically invalidated when the underlying query data change.

The cache eviction is managed by dedicated SQL pool automatically following this schedule:

- Every 48 hours if the result set hasn't been used or has been invalidated.
- When the result set cache approaches the maximum size.

Users can manually empty the entire result set cache by using one of these options:

- Turn OFF the result set cache feature for the database
- Run DBCC DROPRERESULTSETCACHE while connected to the database

Pausing a database won't empty the cached result set.

12.7.3 Task 3: Create and update statistics

The more the dedicated SQL pool resource knows about your data, the faster it can execute queries. After loading data into the dedicated SQL pool, collecting statistics on your data is one of the most important things you can do for query optimization.

The dedicated SQL pool query optimizer is a cost-based optimizer. It compares the cost of various query plans, and then chooses the plan with the lowest cost. In most cases, it chooses the plan that will execute the fastest.

For example, if the optimizer estimates that the date your query is filtering on will return one row it will choose one plan. If it estimates that the selected date will return 1 million rows, it will return a different plan.

1. Check if statistics are set to be automatically created in the database:

```
SELECT name, is_auto_create_stats_on
FROM sys.databases
```

2. See statistics that have been automatically created (change the database back to your dedicated SQL Pool):

```
SELECT *
FROM
    sys.dm_pdw_exec_requests
WHERE
    Command like 'CREATE STATISTICS%'
```

Notice the special name pattern used for automatically created statistics:

REQUEST_ID	SESSION_ID	STATUS	SUBMIT_TIME	START_TIME	END_COMPILE_TIME	END_TIME	TOTAL_ELAPSED_TIME	LABEL	ERROR_ID	DATABASE_ID	COMMAND
QID476406	SID8627	Completed	2020-04-24T15:5...	2020-04-24T15:5...	2020-04-24T15:5...	1140	NULL	NULL	9	CREATE STATISTICS [S_WA_Sys_00000004_184C96B4] ON [wwi].[Date].[Quarter];	
QID476472	SID8643	Completed	2020-04-24T16:2...	2020-04-24T16:2...	2020-04-24T16:2...	1187	NULL	NULL	9	CREATE STATISTICS [S_WA_Sys_00000006_0F824689] ON [wwi_perf].[Sale_Partition02].[TotalAmount];	
QID476473	SID8643	Completed	2020-04-24T16:2...	2020-04-24T16:2...	2020-04-24T16:2...	1093	NULL	NULL	9	CREATE STATISTICS [S_WA_Sys_00000008_0F824689] ON [wwi_perf].[Sale_Partition02].[ProfitAmount];	

3. Check if there are any statistics created for CustomerId from the wwi_perf.Sale_Has table:

```
DBCC SHOW_STATISTICS ('wwi_perf.Sale_Hash', CustomerId) WITH HISTOGRAM
```

You should get an error stating that statistics for CustomerId does not exist.

4. Create statistics for CustomerId:

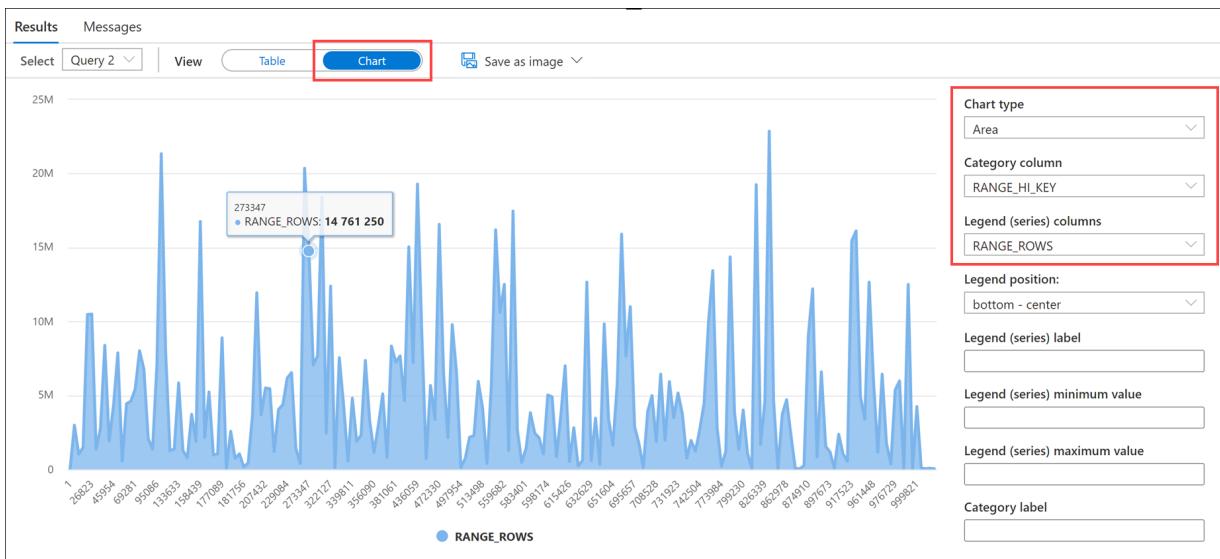
```
CREATE STATISTICS Sale_Hash_CustomerId ON wwi_perf.Sale_Hash (CustomerId)
```

Display the newly created statistics:

```
DBCC SHOW_STATISTICS([wwi_perf.Sale_Hash], 'Sale_Hash_CustomerId')
```

In the results pane, switch to Chart display and configure the properties as follows:

- **Chart type:** Area
- **Category column:** RANGE_HI_KEY
- **Legend (series) columns:** RANGE_ROWS



You now have a visual on the statistics created for the `CustomerId` column.

Important

The more SQL pool knows about your data, the faster it can execute queries against it. After loading data into SQL pool, collecting statistics on your data is one of the most important things you can do to optimize your queries.

The SQL pool query optimizer is a cost-based optimizer. It compares the cost of various query plans, and then chooses the plan with the lowest cost. In most cases, it chooses the plan that will execute the fastest.

For example, if the optimizer estimates that the date your query is filtering on will return one row it will choose one plan. If it estimates that the selected date will return 1 million rows, it will return a different plan.

12.7.4 Task 4: Create and update indexes

Clustered Columnstore Index vs. Heap vs. Clustered and Nonclustered

Clustered indexes may outperform clustered columnstore indexes when a single row needs to be quickly retrieved. For queries where a single or very few row lookup is required to perform with extreme speed, consider a cluster index or nonclustered secondary index. The disadvantage to using a clustered index is that only queries that benefit are the ones that use a highly selective filter on the clustered index column. To improve filter on other columns a nonclustered index can be added to other columns. However, each index which is added to a table adds both space and processing time to loads.

1. Retrieve information about a single customer from the table with CCI:

```
SELECT
*
FROM
[wwi_perf].[Sale_Hash]
WHERE
CustomerId = 500000
```

Take a note of the execution time.

2. Retrieve information about a single customer from the table with a clustered index:

```
SELECT
*
FROM
[wwi_perf].[Sale_Index]
WHERE
CustomerId = 500000
```

The execution time is similar to the one for the query above. Clustered columnstore indexes have no significant advantage over clustered indexes in the specific scenario of highly selective queries.

3. Retrieve information about multiple customers from the table with CCI:

```
SELECT
*
FROM
[wwi_perf].[Sale_Hash]
WHERE
CustomerID between 400000 and 400100
```

and then retrieve the same information from the table with a clustered index:

```
SELECT
*
FROM
[wwi_perf].[Sale_Index]
WHERE
CustomerID between 400000 and 400100
```

Run both queries several times to get a stable execution time. Under normal conditions, you should see that even with a relatively small number of customers, the CCI table starts yielding better results than the clustered index table.

4. Now add an extra condition on the query, one that refers to the `StoreId` column:

```
SELECT
*
FROM
[wwi_perf].[Sale_Index]
WHERE
CustomerID between 400000 and 400100
and StoreId between 2000 and 4000
```

Take a note of the execution time.

5. Create a non-clustered index on the `StoreId` column:

```
CREATE INDEX Store_Index on wwi_perf.Sale_Index (StoreId)
```

The creation of the index should complete in a few minutes. Once the index is created, run the previous query again. Notice the improvement in execution time resulting from the newly created non-clustered index.

Note

Creating a non-clustered index on the `wwi_perf.Sale_Index` is based on the already existing clustered index. As a bonus exercise, try to create the same type of index on the `wwi_perf.Sale_Hash` table. Can you explain the difference in index creation time?

12.7.5 Task 5: Ordered Clustered Columnstore Indexes

By default, for each table created without an index option, an internal component (index builder) creates a non-ordered clustered columnstore index (CCI) on it. Data in each column is compressed into a separate CCI rowgroup segment. There's metadata on each segment's value range, so segments that are outside the bounds of the query predicate aren't read from disk during query execution. CCI offers the highest level of data compression and reduces the size of segments to read so queries can run faster. However, because the index builder doesn't sort data before compressing them into segments, segments with overlapping value ranges could occur, causing queries to read more segments from disk and take longer to finish.

When creating an ordered CCI, the Synapse SQL engine sorts the existing data in memory by the order key(s) before the index builder compresses them into index segments. With sorted data, segment overlapping is reduced allowing queries to have a more efficient segment elimination and thus faster performance because the number of segments to read from disk is smaller. If all data can be sorted in memory at once, then segment overlapping can be avoided. Due to large tables in data warehouses, this scenario doesn't happen often.

Queries with the following patterns typically run faster with ordered CCI:

- The queries have equality, inequality, or range predicates
- The predicate columns and the ordered CCI columns are the same.
- The predicate columns are used in the same order as the column ordinal of ordered CCI columns.

1. Run the following query to show the segment overlaps for the Sale_Hash table:

```

select
    OBJ.name as table_name
    ,COL.name as column_name
    ,NT.distribution_id
    ,NP.partition_id
    ,NP.rows as partition_rows
    ,NP.data_compression_desc
    ,NCSS.segment_id
    ,NCSS.version
    ,NCSS.min_data_id
    ,NCSS.max_data_id
    ,NCSS.row_count
from
    sys.objects OBJ
    JOIN sys.columns as COL ON
        OBJ.object_id = COL.object_id
    JOIN sys.pdw_table_mappings TM ON
        OBJ.object_id = TM.object_id
    JOIN sys.pdw_nodes_tables as NT on
        TM.physical_name = NT.name
    JOIN sys.pdw_nodes_partitions NP on
        NT.object_id = NP.object_id
        and NT.pdw_node_id = NP.pdw_node_id
        and substring(TM.physical_name, 40, 10) = NP.distribution_id
    JOIN sys.pdw_nodes_column_store_segments NCSS on
        NP.partition_id = NCSS.partition_id
        and NP.distribution_id = NCSS.distribution_id
        and COL.column_id = NCSS.column_id
where
    OBJ.name = 'Sale_Hash'
    and COL.name = 'CustomerId'
    and TM.physical_name not like '%HdTable%'
order by
    NT.distribution_id

```

Here is a short description of the tables involved in the query:

Table Name	Description
sys.objects	All objects in the database. Filtered to match only the Sale_Hash table.
sys.columns	All columns in the database. Filtered to match only the CustomerId column of
sys.pdw_table_mappings	Maps each table to local tables on physical nodes and distributions.
sys.pdw_nodes_tables	Contains information on each local table in each distribution.
sys.pdw_nodes_partitions	Contains information on each local partition of each local table in each distribu
sys.pdw_nodes_column_store_segments	Contains information on each CCI segment for each partition and distribution

With this information on hand, take a look at the result:

Results Messages

View Table Chart Export results

Search

TABLE_NAME	COLUMN_NAME	DISTRIBUTION_ID	PARTITION_ID	PARTITION_ROWS	DATA_COMPRESSION_DESC	SEGMENT_ID	VERSION	MIN_DATA_ID	MAX_DATA_ID	ROW_COUNT
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	1	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	2	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	3	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	4	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	5	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	6	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	7	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	8	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	9	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	10	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	11	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	12	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	13	1	10	999996	1048576
Sale_Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	14	1	10	999996	1048576
Sale Hash	Customerid	1	72057594203209728	48215156	COLUMNSTORE	15	1	10	999996	1048576

Browse through the result set and notice the significant overlap between segments. There is literally overlap in customer ids between every single pair of segments (CustomerId values in the data range from 1 to 1,000,000). The segment structure of this CCI is clearly inefficient and will result in a lot of unnecessary reads from storage.

- Run the following query to show the segment overlaps for the Sale_Hash_Ordered table:

```
select
    OBJ.name as table_name
    ,COL.name as column_name
    ,NT.distribution_id
    ,NP.partition_id
    ,NP.rows as partition_rows
    ,NP.data_compression_desc
    ,NCSS.segment_id
    ,NCSS.version
    ,NCSS.min_data_id
    ,NCSS.max_data_id
    ,NCSS.row_count
from
    sys.objects OBJ
    JOIN sys.columns as COL ON
        OBJ.object_id = COL.object_id
    JOIN sys.pdw_table_mappings TM ON
        OBJ.object_id = TM.object_id
    JOIN sys.pdw_nodes_tables as NT on
        TM.physical_name = NT.name
    JOIN sys.pdw_nodes_partitions NP on
        NT.object_id = NP.object_id
        and NT.pdw_node_id = NP.pdw_node_id
        and substring(TM.physical_name, 40, 10) = NP.distribution_id
    JOIN sys.pdw_nodes_column_store_segments NCSS on
        NP.partition_id = NCSS.partition_id
        and NP.distribution_id = NCSS.distribution_id
        and COL.column_id = NCSS.column_id
where
    OBJ.name = 'Sale_Hash_Ordered'
    and COL.name = 'CustomerId'
    and TM.physical_name not like '%HdTable%'
order by
    NT.distribution_id
```

The CTAS used to create the wwi_perf.Sale_Hash_Ordered table was the following (**Do not execute**):

```
CREATE TABLE [wwi_perf].[Sale_Hash_Ordered]
WITH
(
    DISTRIBUTION = HASH ( [CustomerId] ),
    CLUSTERED COLUMNSTORE INDEX ORDER( [CustomerId] )
)
AS
```

```

SELECT
*
FROM
    [wwi_perf].[Sale_Heap]
OPTION (LABEL = 'CTAS : Sale_Hash', MAXDOP 1)

```

Notice the creation of the ordered CCI with MAXDOP = 1. Each thread used for ordered CCI creation works on a subset of data and sorts it locally. There's no global sorting across data sorted by different threads. Using parallel threads can reduce the time to create an ordered CCI but will generate more overlapping segments than using a single thread. Currently, the MAXDOP option is only supported in creating an ordered CCI table using CREATE TABLE AS SELECT command. Creating an ordered CCI via CREATE INDEX or CREATE TABLE commands does not support the MAXDOP option.

The results show significantly less overlap between segments:

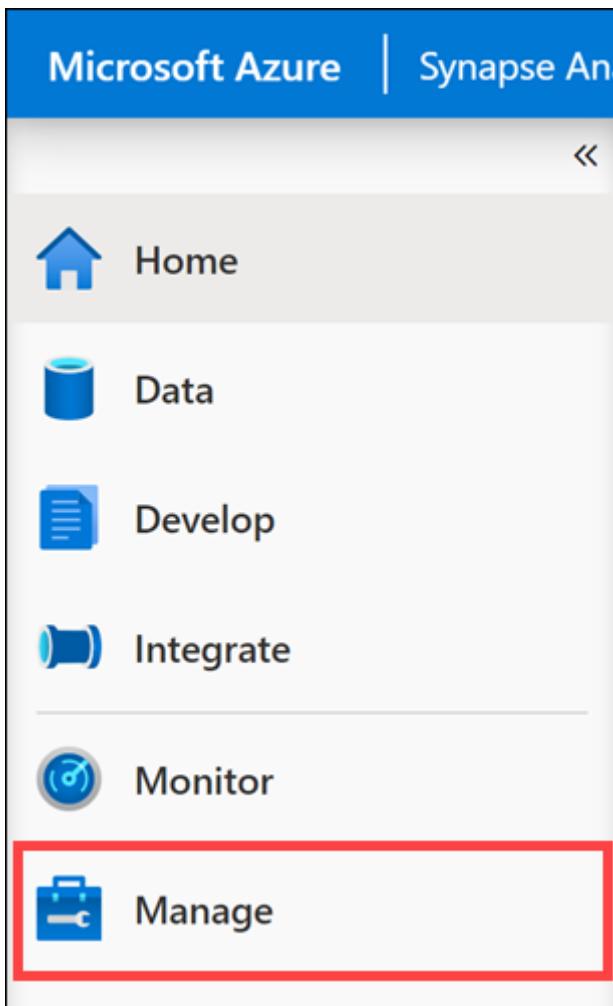
TABLE_NAME	COLUMN_NAME	DISTRIBUTION_ID	PARTITION_ID	PARTITION_ROWS	DATA_COMPRESSION_DESC	SEGMENT_ID	VERSION	MIN_DATA_ID	MAX_DATA_ID	ROW_COUNT
Sale_Hash_Ordered	CustomerID	0	72057594261405696	48215156	COLUMNSTORE	0	1	347705	1048576	
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	1	1	347705	692049	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	2	1	10	999996	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	3	1	30939	377130	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	4	1	377130	723958	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	5	1	10	999996	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	6	1	63020	410614	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	7	1	410614	756816	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	8	1	10	999996	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	9	1	94801	439760	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	10	1	439760	785796	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	11	1	10	999996	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	12	1	127555	474518	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	13	1	474518	817481	1048576
Sale_Hash_Ordered	CustomerID	1	72057594261405696	48215156	COLUMNSTORE	14	1	10	999996	1048576

12.8 Exercise 5: Cleanup

Complete these steps to free up resources you no longer need.

12.8.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



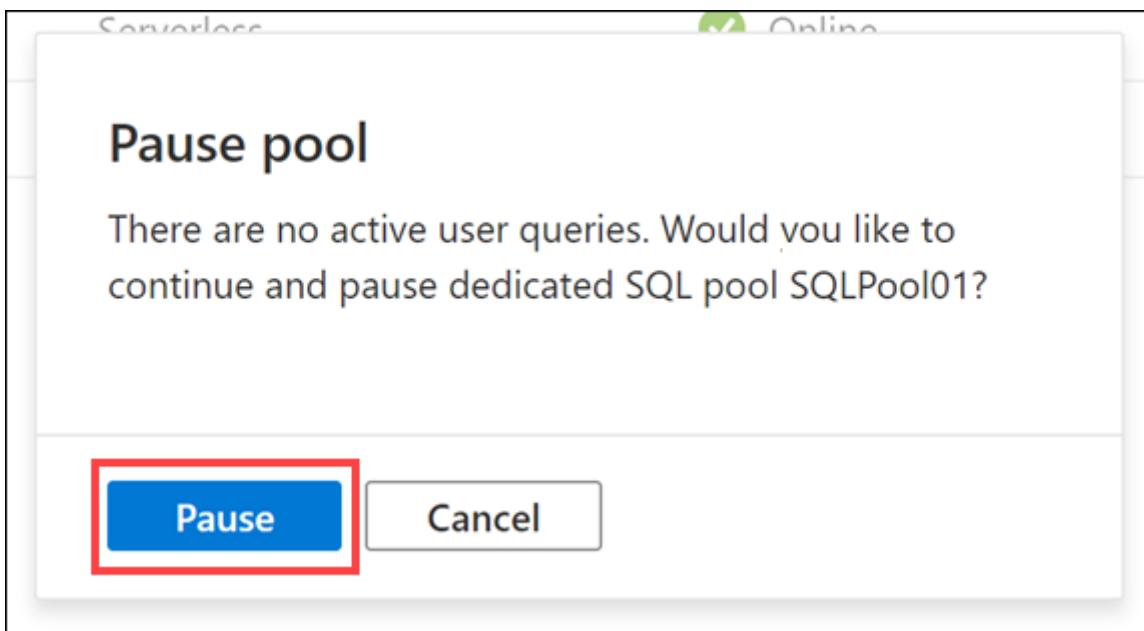
3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

The screenshot shows the 'SQL pools' blade in the Azure portal. The left sidebar lists various pool types: Analytics pools (with 'SQL pools' highlighted by a red box and a red number 1), External connections, Integration, and Security. The main area displays the 'SQL pools' table with the following data:

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

A red box highlights the 'Pause' button for the 'SQLPool01' row, and a red number 2 is placed above it.

4. When prompted, select **Pause**.



13 Module 11 - Analyze and optimize Data Warehouse storage

In this module, students will learn how to analyze then optimize the data storage of the Azure Synapse dedicated SQL pools. The student will know techniques to understand table space usage and column store storage details. Next the student will know how to compare storage requirements between identical tables that use different data types. Finally, the student will observe the impact materialized views have when executed in place of complex queries and learn how to avoid extensive logging by optimizing delete operations.

In this module, the student will be able to:

- Check for skewed data and space usage
- Understand column store storage details
- Study the impact of materialized views
- Explore rules for minimally logged operations

13.1 Lab details

- [Module 11 - Analyze and optimize Data Warehouse storage](#)
 - [Lab details](#)
 - [Lab setup and pre-requisites](#)
 - [Exercise 0: Start the dedicated SQL pool](#)
 - [Exercise 1 - Check for skewed data and space usage](#)
 - * [Task 1 - Analyze the space used by tables](#)
 - * [Task 2 - Use a more advanced approach to understand table space usage](#)
 - [Exercise 2 - Understand column store storage details](#)
 - * [Task 1 - Create view for column store row group stats](#)
 - * [Task 2 - Explore column store storage details](#)
 - [Exercise 3 - Study the impact of wrong choices for column data types](#)
 - * [Task 1 - Create and populate tables with optimal column data types](#)
 - * [Task 2 - Create and populate tables with sub-optimal column data types](#)
 - * [Task 3 - Compare storage requirements](#)
 - [Exercise 4 - Study the impact of materialized views](#)
 - * [Task 1 - Analyze the execution plan of a query](#)
 - * [Task 2 - Improve the execution plan of the query with a materialized view](#)
 - [Exercise 5 - Avoid extensive logging](#)
 - * [Task 1 - Explore rules for minimally logged operations](#)
 - * [Task 2 - Optimizing a delete operation](#)
 - [Exercise 6: Cleanup](#)
 - * [Task 1: Pause the dedicated SQL pool](#)

13.2 Lab setup and pre-requisites

Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Complete the **lab setup instructions** for this module.

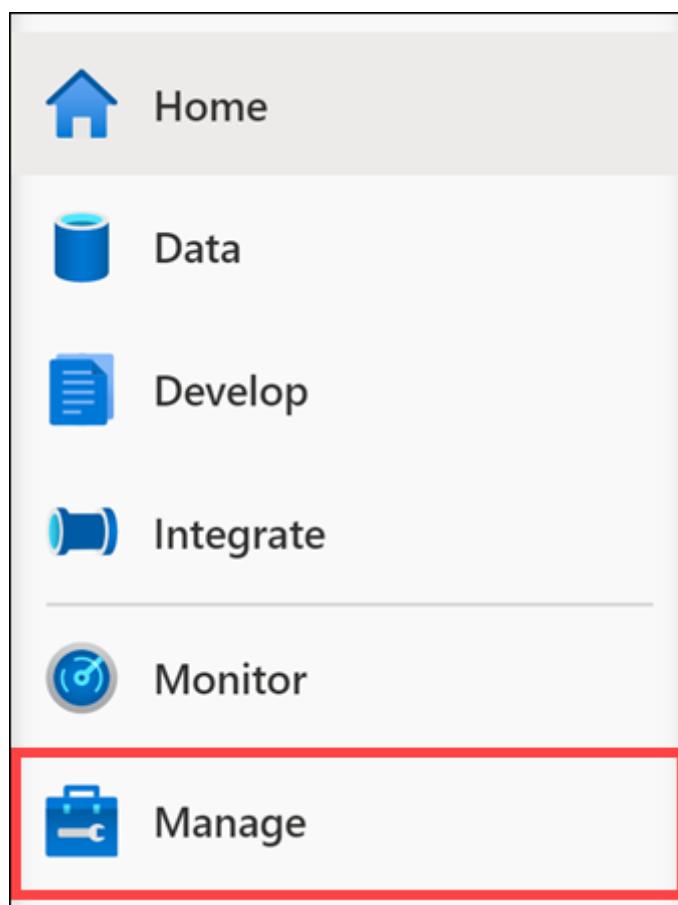
Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

13.3 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

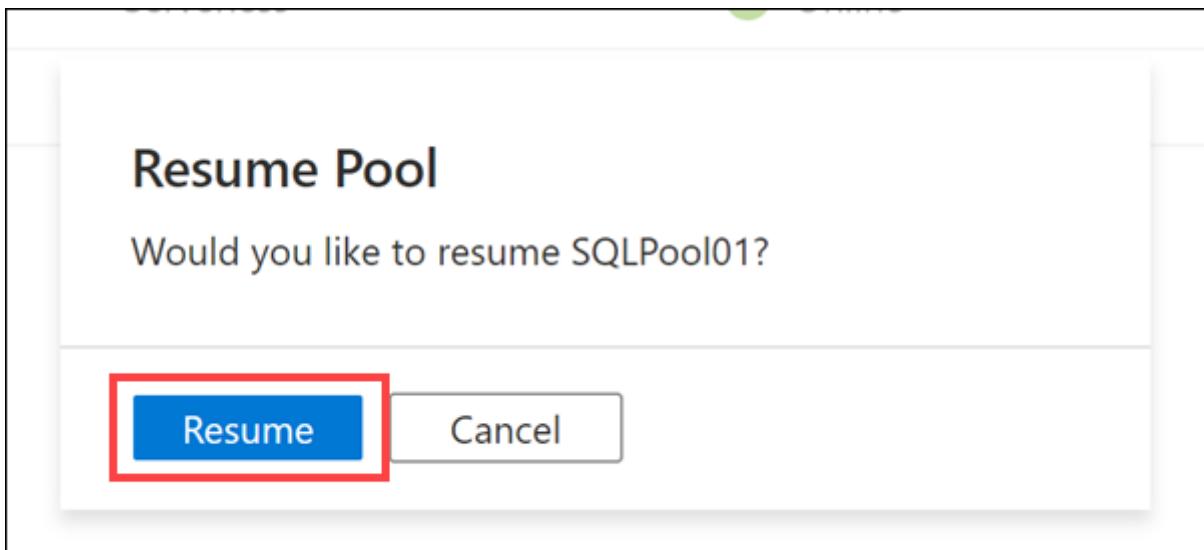
1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the 'SQL pools' page in the Azure Synapse Studio. On the left, there's a sidebar with various options like Analytics pools, External connections, Integration, Security, and Access control. The 'SQL pools' option is selected and highlighted with a red box and the number '1'. The main area shows a table with one item: 'SQLPool01'. The table has columns for Name, Type, Status, and Size. The 'Status' column shows 'Online' with a green checkmark and 'Paused' with a grey circle. A red box labeled '2' surrounds the 'Resume' button in the 'Actions' column for the 'SQLPool01' row.

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.

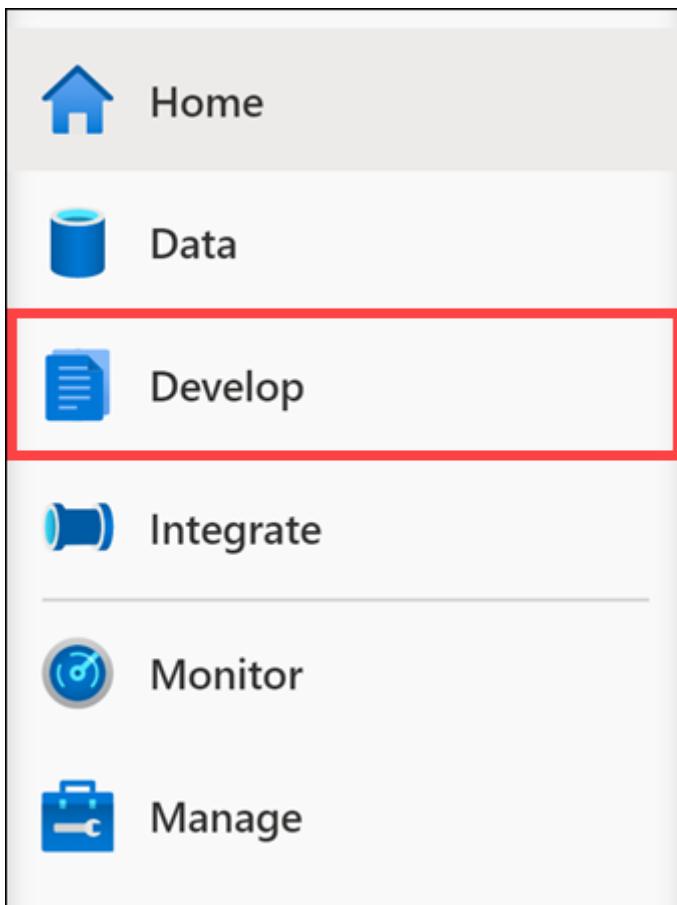


Continue to the next exercise while the dedicated SQL pool resumes.

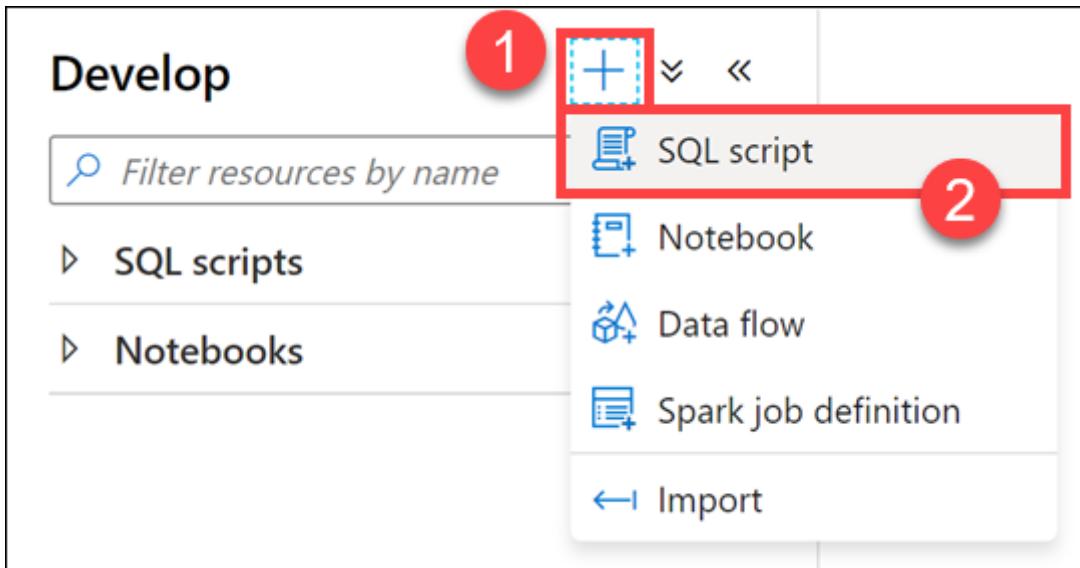
13.4 Exercise 1 - Check for skewed data and space usage

13.4.1 Task 1 - Analyze the space used by tables

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Develop** hub.



3. From the **Develop** menu, select the + button (1) and choose **SQL Script** (2) from the context menu.



4. In the toolbar menu, connect to the **SQLPool01** dedicated SQL pool as well as the **SQLPool01** database.



5. Paste the following script to create a hash-distributed Clustered Columnstore Index (CCI) table using CTAS (Create Table As Select) if it does not already exist:

```
IF OBJECT_ID(N'[wwi_perf].[Sale_Hash]', N'U') IS NULL
BEGIN
    CREATE TABLE [wwi_perf].[Sale_Hash]
    WITH
    (
```

```

        DISTRIBUTION = HASH ( [CustomerId] ),
CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT
*
FROM
[wwi_poc].[Sale]
END

```

6. Select **Run** from the toolbar menu to execute the SQL command.



If the table does not already exist, the query will take around **10 minutes** to complete. While this is running, read the rest of the lab instructions to familiarize yourself with the content.

NOTE

CTAS is a more customizable version of the SELECT...INTO statement. SELECT...INTO doesn't allow you to change either the distribution method or the index type as part of the operation. You create the new table by using the default distribution type of ROUND_ROBIN, and the default table structure of CLUSTERED COLUMNSTORE INDEX.

With CTAS, on the other hand, you can specify both the distribution of the table data as well as the table structure type.

7. In the query window, replace the script with the following Database Console Command (DBCC):

```
DBCC PDW_SHOWSPACEUSED('wwi_perf.Sale_Hash');
```

Results						
Messages						
View	Table	Chart	Export results			
<input type="text"/> Search						
ROWS	RESERVED_SPACE	DATA_SPACE	INDEX_SPACE	UNUSED_SPACE	PDW_NODE_ID	DISTRIBUTION_ID
16315504	311160	311104	0	56	1	1
16498478	314384	314328	0	56	1	2
16403834	312640	312584	0	56	1	3
16461768	313696	313640	0	56	1	4
16375412	312056	312000	0	56	1	5
16379175	312256	312200	0	56	1	6
16348103	311536	311480	0	56	1	7
16417438	312928	312872	0	56	1	8
16422836	312312	312256	0	56	1	9
16333578	311648	311592	0	56	1	10
16426341	312760	312704	0	56	1	11
16334437	311392	311336	0	56	1	12
16276619	310192	310136	0	56	1	13
16478492	313664	313608	0	56	1	14

8. Analyze the number of rows in each distribution. Those numbers should be as even as possible. You can see from the results that rows are equally distributed across distributions. Let's dive a bit more into this analysis. Use the following query to get customers with the most sale transaction items:

```
SELECT TOP 1000
CustomerId,
```

```

    count(*) as TransactionItemsCount
FROM
    [wwi_perf].[Sale_Hash]
GROUP BY
    CustomerId
ORDER BY
    count(*) DESC

```

Results Messages

View **Table** Chart [Export results](#)

Search

CustomerId	TransactionItemsCount
592004	3377
559682	3370
322127	3311
420390	3106
549076	3102
673714	3069
436059	3053
598065	3047
185880	3037
705332	3034
915880	3027
902934	2987
519689	2971
325395	2970

Now find the customers with the least sale transaction items:

```

SELECT TOP 1000
    CustomerId,
    count(*) as TransactionItemsCount
FROM
    [wwi_perf].[Sale_Hash]
GROUP BY
    CustomerId
ORDER BY
    count(*) ASC

```

Results Messages

View **Table** Chart [Export results](#)

Search

CustomerId	TransactionItemsCount
98718	62
207561	69
606484	70
639663	71
712472	74
725622	75
630462	75
710198	76
897971	79
637624	80
169153	80
363701	80
242199	83
714291	84

Notice that in our environment, the largest number of transaction items is 3,377 and the smallest is 62.

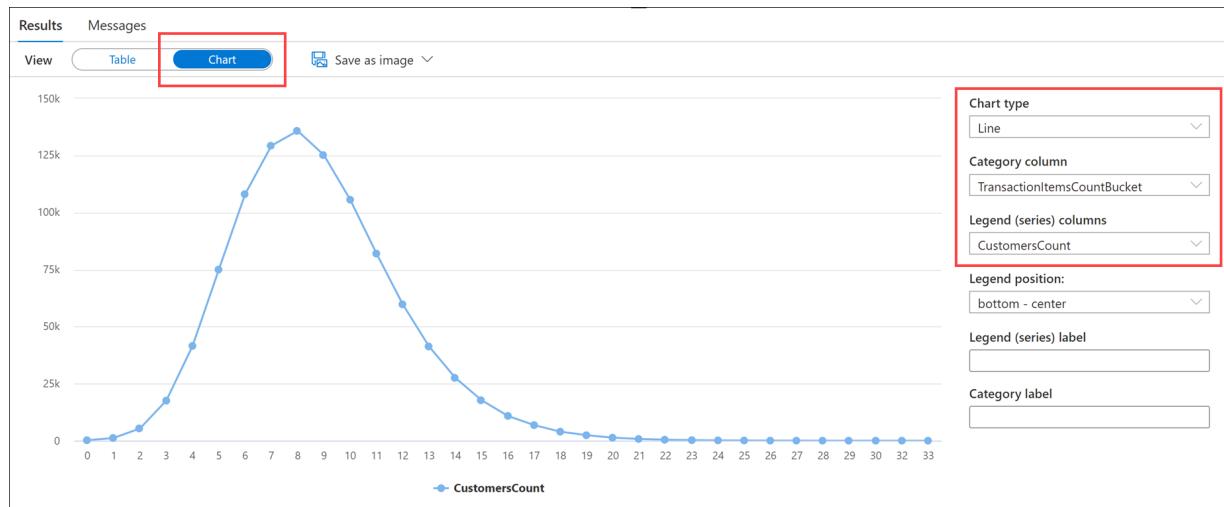
Let's find now the distribution of per-customer transaction item counts. Run the following query:

```

SELECT
    T.TransactionItemsCountBucket
    ,count(*) as CustomersCount
FROM
(
    SELECT
        CustomerId,
        (count(*) - 62) / 100 as TransactionItemsCountBucket
    FROM
        [wwi_perf].[Sale_Hash]
    GROUP BY
        CustomerId
) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
    T.TransactionItemsCountBucket

```

In the **Results** pane, switch to the **Chart** view and configure it as follows (see the options set on the right side):



Without diving too much into the mathematical and statistical aspects of it, this histogram displays the reason why there is virtually no skew in the data distribution of the **Sale_Hash** table. If you haven't figured it out yet, the reason we are talking about is the quasi-normal distribution of the per-customer transaction items counts.

13.4.2 Task 2 - Use a more advanced approach to understand table space usage

- Run the following script to create the **vTableSizes** view:

```

CREATE VIEW [wwi_perf].[vTableSizes]
AS
WITH base
AS
(
SELECT
    GETDATE()
    , DB_NAME()
    , s.name
    , t.name
    , QUOTENAME(s.name)+'. '+QUOTENAME(t.name)
    , nt.[name]
    , ROW_NUMBER() OVER(PARTITION BY nt.[name] ORDER BY (SELECT NULL))
    , tp.[distribution_policy_desc]
    AS [execution_time]
    AS [database_name]
    AS [schema_name]
    AS [table_name]
    AS [two_part_name]
    AS [node_table_name]
    AS [node_table_name_desc]
)

```

```

        , c.[name]
        , nt.[distribution_id]
        , i.[type]
        , i.[type_desc]
        , nt.[pdw_node_id]
        , pn.[type]
        , pn.[name]
        , di.name
        , di.position
        , nps.[partition_number]
        , nps.[reserved_page_count]
        , nps.[reserved_page_count] - nps.[used_page_count]
        , nps.[in_row_data_page_count]
            + nps.[row_overflow_used_page_count]
            + nps.[lob_used_page_count]
        , nps.[reserved_page_count]
            - (nps.[reserved_page_count] - nps.[used_page_count])
            - ([in_row_data_page_count]
                + [row_overflow_used_page_count]+[lob_used_page_count])
        , nps.[row_count]
AS [distribution_column]
AS [distribution_id]
AS [index_type]
AS [index_type_desc]
AS [pdw_node_id]
AS [pdw_node_type]
AS [pdw_node_name]
AS [dist_name]
AS [dist_position]
AS [partition_nmbr]
AS [reserved_space_page]
AS [unused_space_page]
AS [data_space_page_co
AS [index_space_page_c
AS [row_count]

FROM
    sys.schemas s
INNER JOIN sys.tables t
    ON s.[schema_id] = t.[schema_id]
INNER JOIN sys.indexes i
    ON t.[object_id] = i.[object_id]
    AND i.[index_id] <= 1
INNER JOIN sys.pdw_table_distribution_properties tp
    ON t.[object_id] = tp.[object_id]
INNER JOIN sys.pdw_table_mappings tm
    ON t.[object_id] = tm.[object_id]
INNER JOIN sys.pdw_nodes_tables nt
    ON tm.[physical_name] = nt.[name]
INNER JOIN sys.dm_pdw_nodes pn
    ON nt.[pdw_node_id] = pn.[pdw_node_id]
INNER JOIN sys.pdw_distributions di
    ON nt.[distribution_id] = di.[distribution_id]
INNER JOIN sys.dm_pdw_nodes_db_partition_stats nps
    ON nt.[object_id] = nps.[object_id]
    AND nt.[pdw_node_id] = nps.[pdw_node_id]
    AND nt.[distribution_id] = nps.[distribution_id]
LEFT OUTER JOIN (select * from sys.pdw_column_distribution_properties where distribution_ordinal =
    ON t.[object_id] = cdp.[object_id]
LEFT OUTER JOIN sys.columns c
    ON cdp.[object_id] = c.[object_id]
    AND cdp.[column_id] = c.[column_id]
WHERE pn.[type] = 'COMPUTE'
)
, size
AS
(
SELECT
    [execution_time]
    , [database_name]
    , [schema_name]
    , [table_name]
    , [two_part_name]
    , [node_table_name]
    , [node_table_name_seq]
    , [distribution_policy_name]
    , [distribution_column]

```

```

, [distribution_id]
, [index_type]
, [index_type_desc]
, [pdw_node_id]
, [pdw_node_type]
, [pdw_node_name]
, [dist_name]
, [dist_position]
, [partition_nmbr]
, [reserved_space_page_count]
, [unused_space_page_count]
, [data_space_page_count]
, [index_space_page_count]
, [row_count]
, ([reserved_space_page_count] * 8.0) AS [reserved_space_KB]
, ([reserved_space_page_count] * 8.0)/1000 AS [reserved_space_MB]
, ([reserved_space_page_count] * 8.0)/1000000 AS [reserved_space_GB]
, ([reserved_space_page_count] * 8.0)/1000000000 AS [reserved_space_TB]
, ([unused_space_page_count] * 8.0) AS [unused_space_KB]
, ([unused_space_page_count] * 8.0)/1000 AS [unused_space_MB]
, ([unused_space_page_count] * 8.0)/1000000 AS [unused_space_GB]
, ([unused_space_page_count] * 8.0)/1000000000 AS [unused_space_TB]
, ([data_space_page_count] * 8.0) AS [data_space_KB]
, ([data_space_page_count] * 8.0)/1000 AS [data_space_MB]
, ([data_space_page_count] * 8.0)/1000000 AS [data_space_GB]
, ([data_space_page_count] * 8.0)/1000000000 AS [data_space_TB]
, ([index_space_page_count] * 8.0) AS [index_space_KB]
, ([index_space_page_count] * 8.0)/1000 AS [index_space_MB]
, ([index_space_page_count] * 8.0)/1000000 AS [index_space_GB]
, ([index_space_page_count] * 8.0)/1000000000 AS [index_space_TB]
FROM base
)
SELECT *
FROM size

```

Take a moment to analyze the script above. You have encountered already some of the tables in the previous lab. Here is a short description of the tables and DMVs involved in the query:

Table Name	Description
sys.schemas	All schemas in the database.
sys.tables	All tables in the database.
sys.indexes	All indexes in the database.
sys.columns	All columns in the database.
sys.pdw_table_mappings	Maps each table to local tables on physical nodes and distributions.
sys.pdw_nodes_tables	Contains information on each local table in each distribution.
sys.pdw_table_distribution_properties	Holds distribution information for tables (the type of distribution tables have).
sys.pdw_column_distribution_properties	Holds distribution information for columns. Filtered to include only columns u
sys.pdw_distributions	Holds information about the distributions from the SQL pool.
sys.dm_pdw_nodes	Holds information about the nodes from the SQL pool. Filtered to include only
sys.dm_pdw_nodes_db_partition_stats	Returns page and row-count information for every partition in the current data

- Run the following script to view the details about the structure of the tables in the `wwi_perf` schema, as well as the `[wwi_poc].[Sale]` table, which was used as the source for the `Sale_Hash` table.:

```

SELECT
    schema_name
,   table_name
,   distribution_policy_name
,   distribution_column
,   index_type_desc

```

```

    , COUNT(distinct partition_nmbr) as nbr_partitions
    , SUM(row_count) as table_row_count
    , SUM(reserved_space_GB) as table_reserved_space_GB
    , SUM(data_space_GB) as table_data_space_GB
    , SUM(index_space_GB) as table_index_space_GB
    , SUM(unused_space_GB) as table_unused_space_GB
FROM [wwi_perf].[vTableSizes]
WHERE schema_name = 'wwi_perf' OR (schema_name = 'wwi_poc' AND table_name = 'sale')
GROUP BY database_name
, schema_name
, table_name
, distribution_policy_name
, distribution_column
, index_type_desc
ORDER BY table_reserved_space_GB desc

```

Analyze the results:

schema_name	table_name	distribution_policy_name	distribution_column	index_type_desc	nbr_partitions	table_row_count	table_reserved_space_GB	table_data_space_GB	table_index_sp...
wwi_poc	Sale	ROUND_ROBIN	NULL	HEAP	1	981995895	48.478784000	48.470176000	0.001920000
wwi_perf	Sale_Index	HASH	CustomerId	CLUSTERED	1	679014492	19.394688000	19.345112000	0.045648000
wwi_perf	Sale_Hash	HASH	CustomerId	CLUSTERED COLUMNSTORE	1	981995895	18.714432000	18.711072000	0.000000000
wwi_perf	Sale_Partition01	HASH	CustomerId	CLUSTERED COLUMNSTORE	13	339886790	6.796848000	6.748352000	0.007440000
wwi_perf	Sale_Partition02	HASH	CustomerId	CLUSTERED COLUMNSTORE	5	339507246	6.426032000	6.385728000	0.004168000
wwi_perf	Sale_Hash_Ordered	HASH	CustomerId	CLUSTERED COLUMNSTORE	1	339507246	6.038952000	6.035592000	0.000000000
wwi_perf	Sale_Heap	ROUND_ROBIN	NULL	HEAP	1	1325209	0.062432000	0.059992000	0.000480000

Notice the significant difference between the space used by CLUSTERED COLUMNSTORE and HEAP or CLUSTERED tables. This provides a clear indication on the significant advantages columnstore indexes have. **Make note of the row counts.**

13.5 Exercise 2 - Understand column store storage details

13.5.1 Task 1 - Create view for column store row group stats

- Run the following query to create the vColumnStoreRowGroupStats:

```

create view [wwi_perf].[vColumnStoreRowGroupStats]
as
with cte
as
(
select tb.[name] AS [logical_table_name]
, rg.[row_group_id] AS [row_group_id]
, rg.[state] AS [state]
, rg.[state_desc] AS [state_desc]
, rg.[total_rows] AS [total_rows]
, rg.[trim_reason_desc] AS trim_reason_desc
, mp.[physical_name] AS physical_name
FROM sys.[schemas] sm
JOIN sys.[tables] tb ON sm.[schema_id] = tb.[schema_id]
JOIN sys.[pdw_table_mappings] mp ON tb.[object_id] = mp.[object_id]
JOIN sys.[pdw_nodes_tables] nt ON nt.[name] = mp.[physical_name]
JOIN sys.[dm_pdw_nodes_db_column_store_row_group_physical_stats] rg ON rg.[object_id] = nt.[distribution_id]
AND rg.[pdw_node_id] = nt.[distribution_id]

```

```
)
select *
from cte;
```

In this query we are using the `sys.dm_pdw_nodes_db_column_store_row_group_physical_stats` DMV which provides current rowgroup-level information about all of the columnstore indexes in the current database.

The `state_desc` column provides useful information on the state of a row group:

Name	Description
INVISIBLE	A rowgroup which is being compressed.
OPEN	A deltastore rowgroup that is accepting new rows. It is important to remember that an open rowgroup is still being updated.
CLOSED	A deltastore rowgroup that contains the maximum number of rows, and is waiting for the tuple mover process to move the data to the columnstore.
COMPRESSED	A row group that is compressed with columnstore compression and stored in the columnstore.
TOMBSTONE	A row group that was formerly in the deltastore and is no longer used.

The `trim_reason_desc` column describes the reason that triggered the COMPRESSED rowgroup to have less than the maximum number of rows:

Name	Description
UNKNOWN_UPGRADED_FROM_PREVIOUS_VERSION	Occurred when upgrading from the previous version of SQL Server.
NO_TRIM	The row group was not trimmed. The row group was compressed with the default settings.
BULKLOAD	The bulk-load batch size limited the number of rows. This is what you should expect when using BULKLOAD.
REORG	Forced compression as part of REORG command.
DICTIONARY_SIZE	Dictionary size grew too large to compress all of the rows together.
MEMORY_LIMITATION	Not enough available memory to compress all the rows together.
RESIDUAL_ROW_GROUP	Closed as part of last row group with rows < 1 million during index build or shrink.

13.5.2 Task 2 - Explore column store storage details

- Explore the statistics of the columnstore for the `Sale_Partition01` table using the following query:

```

SELECT
*
FROM
[wwi_perf].[vColumnStoreRowGroupStats]
WHERE
Logical_Table_Name = 'Sale_Partition01'
```

- Explore the results of the query:

Logical_table_name	Row_group_id	State	State_desc	Total_rows	Trim_reason_desc	Physical_name
Sale_Partition01	0	3	COMPRESSED	149822	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_23
Sale_Partition01	1	3	COMPRESSED	113318	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_25
Sale_Partition01	5	3	COMPRESSED	184746	AUTO_MERGE	Table_5a8866f1e9624ff49526ebf0251a91e7_26
Sale_Partition01	3	3	COMPRESSED	112351	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_27
Sale_Partition01	4	3	COMPRESSED	78682	REORG	Table_5a8866f1e9624ff49526ebf0251a91e7_28
Sale_Partition01	4	3	COMPRESSED	82663	REORG	Table_5a8866f1e9624ff49526ebf0251a91e7_30
Sale_Partition01	3	3	COMPRESSED	149717	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_31
Sale_Partition01	1	3	COMPRESSED	110873	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_33
Sale_Partition01	0	3	COMPRESSED	125922	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_34
Sale_Partition01	2	3	COMPRESSED	105846	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_35
Sale_Partition01	2	3	COMPRESSED	139054	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_36
Sale_Partition01	1	1	OPEN	78635	NULL	Table_5a8866f1e9624ff49526ebf0251a91e7_37
Sale_Partition01	0	3	COMPRESSED	119465	BULKLOAD	Table_5a8866f1e9624ff49526ebf0251a91e7_38
Sale_Partition01	4	3	COMPRESSED	94810	REORG	Table_5a8866f1e9624ff49526ebf0251a91e7_39

Browse through the results and get an overview of the rowgroup states. Notice the COMPRESSED and OPEN states of some of the row groups.

- Explore the statistics of the columnstore for the `Sale_Hash_Ordered` table using the same query:

```
SELECT
*
FROM
[wwi_perf].[vColumnStoreRowGroupStats]
WHERE
Logical_Table_Name = 'Sale_Hash_Ordered'
```

- Explore the results of the query:

Logical_table_name	Row_group_id	State	State_desc	Total_rows	Trim_reason_desc	Physical_name
Sale_Hash_Ordered	5	3	COMPRESSED	397731	BULKLOAD	Table_426afa17d3444d18b5173cbec7ae54a5_58
Sale_Hash_Ordered	5	3	COMPRESSED	454797	BULKLOAD	Table_426afa17d3444d18b5173cbec7ae54a5_59
Sale_Hash_Ordered	5	3	COMPRESSED	402974	BULKLOAD	Table_426afa17d3444d18b5173cbec7ae54a5_60
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_1
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_2
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_3
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_4
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_5
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_6
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_7
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_8
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_9
Sale_Hash_Ordered	4	3	COMPRESSED	1048576	NO_TRIM	Table_426afa17d3444d18b5173cbec7ae54a5_10

There is a significant difference in the rowgroup states from the previous one. This highlights one of the potential advantages of ordered CCIs.

13.6 Exercise 3 - Study the impact of wrong choices for column data types

13.6.1 Task 1 - Create and populate tables with optimal column data types

Use the following query to create two tables (`Sale_Hash_Projection` and `Sale_Hash_Projection2`) which contain a subset of the columns from `Sale_Hash_Ordered`:

```
CREATE TABLE [wwi_perf].[Sale_Hash_Projection]
WITH
(
    DISTRIBUTION = HASH ( [CustomerId] ),
    HEAP
)
AS
SELECT
    [CustomerId]
    ,[ProductId]
    ,[Quantity]
FROM
    [wwi_perf].[Sale_Hash_Ordered]

CREATE TABLE [wwi_perf].[Sale_Hash_Projection2]
WITH
(
    DISTRIBUTION = HASH ( [CustomerId] ),
    CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT
    [CustomerId]
    ,[ProductId]
```

```

, [Quantity]
FROM
[wwi_perf].[Sale_Hash_Ordered]
```

The query should finish execution in less than four minutes. While this is running, read the rest of the lab instructions to familiarize yourself with the content.

13.6.2 Task 2 - Create and populate tables with sub-optimal column data types

Use the following query to create two additional tables (Sale_Hash_Projection_Big and Sale_Hash_Projection_Big2) that have the same columns, but with different (sub_optimal) data types:

```

CREATE TABLE [wwi_perf].[Sale_Hash_Projection_Big]
WITH
(
    DISTRIBUTION = HASH ( [CustomerId] ),
    HEAP
)
AS
SELECT
    [CustomerId]
    ,CAST([ProductId] as bigint) as [ProductId]
    ,CAST([Quantity] as bigint) as [Quantity]
FROM
[wwi_perf].[Sale_Hash_Ordered]

CREATE TABLE [wwi_perf].[Sale_Hash_Projection_Big2]
WITH
(
    DISTRIBUTION = HASH ( [CustomerId] ),
    CLUSTERED COLUMNSTORE INDEX
)
AS
SELECT
    [CustomerId]
    ,CAST([ProductId] as bigint) as [ProductId]
    ,CAST([Quantity] as bigint) as [Quantity]
FROM
[wwi_perf].[Sale_Hash_Ordered]
```

The query should finish execution in about the same amount of time. While this is running, continue reading the lab instructions.

13.6.3 Task 3 - Compare storage requirements

- Verify that the four tables have the same number of rows (there should be 339,507,246 rows in each):

```

SELECT 'Sale_Hash_Projection', COUNT_BIG(*) FROM [wwi_perf].[Sale_Hash_Projection]
UNION
SELECT 'Sale_Hash_Projection2', COUNT_BIG(*) FROM [wwi_perf].[Sale_Hash_Projection2]
UNION
SELECT 'Sale_Hash_Projection_Big', COUNT_BIG(*) FROM [wwi_perf].[Sale_Hash_Projection_Big]
UNION
SELECT 'Sale_Hash_Projection_Big2', COUNT_BIG(*) FROM [wwi_perf].[Sale_Hash_Projection_Big2]
```

- Run the following query to compare the storage requirements for the three tables:

```

SELECT
    table_name
    , distribution_policy_name
    , distribution_column
    , index_type_desc
    , COUNT(distinct partition_nmbr) as nbr_partitions
    , SUM(row_count) as table_row_count
```

```

        ,      SUM(reserved_space_GB)           as table_reserved_space_GB
        ,      SUM(data_space_GB)              as table_data_space_GB
        ,      SUM(index_space_GB)            as table_index_space_GB
        ,      SUM(unused_space_GB)           as table_unused_space_GB
FROM
    [wwi_perf].[vTableSizes]
WHERE
    schema_name = 'wwi_perf'
    and table_name in ('Sale_Hash_Projection', 'Sale_Hash_Projection2',
                        'Sale_Hash_Projection_Big', 'Sale_Hash_Projection_Big2')
GROUP BY
    database_name
    ,     schema_name
    ,     table_name
    ,     distribution_policy_name
    ,     distribution_column
    ,     index_type_desc
ORDER BY
    table_data_space_GB desc

```

3. Analyze the results:

table_name	distribution_p...	distribution_co...	index_type_desc	nbr_partitions	table_row_count	table_reserved_space_GB	table_data_space_GB
Sale_Hash_Projection_Big	HASH	CustomerId	HEAP	1	339507246	8.156880000	8.150248000
Sale_Hash_Projection	HASH	CustomerId	HEAP	1	339507246	8.157448000	8.150216000
Sale_Hash_Projection_Big2	HASH	CustomerId	CLUSTERED CO...	1	339507246	1.011488000	1.008128000
Sale_Hash_Projection2	HASH	CustomerId	CLUSTERED CO...	1	339507246	1.008824000	1.005464000

With a small number of rows (around 340 million), you can see some space differences caused by the BIGINT column type versus SMALLINT and TINYINT.

We ran this same query after loading 2.9 billion rows, and the results were more pronounced. There are two important conclusions to draw here:

- In the case of HEAP tables, the storage impact of using BIGINT instead of SMALLINT (for ProductId) and TINYINT (for QUANTITY) is almost 1 GB (0.8941 GB). We're talking here about only two columns and a moderate number of rows (2.9 billion).
- Even in the case of CLUSTERED COLUMNSTORE tables, where compression will offset some of the differences, there is still a difference of 12.7 MB.

Minimizing the size of data types shortens the row length, which leads to better query performance. Use the smallest data type that works for your data:

- Avoid defining character columns with a large default length. For example, if the longest value is 25 characters, then define your column as VARCHAR(25).
- Avoid using [NVARCHAR][NVARCHAR] when you only need VARCHAR.
- When possible, use NVARCHAR(4000) or VARCHAR(8000) instead of NVARCHAR(MAX) or VARCHAR(MAX).

Note

If you are using PolyBase external tables to load your SQL pool tables, the defined length of the table row cannot exceed 1 MB. When a row with variable-length data exceeds 1 MB, you can load the row with BCP, but not with PolyBase.

13.7 Exercise 4 - Study the impact of materialized views

13.7.1 Task 1 - Analyze the execution plan of a query

1. Run again the query to find the number of customers in each bucket of per-customer transaction items counts:

```
SELECT
    T.TransactionItemsCountBucket
    ,count(*) as CustomersCount
FROM
(
    SELECT
        CustomerId,
        (count(*) - 184) / 100 as TransactionItemsCountBucket
    FROM
        [wwi_perf].[Sale_Hash]
    GROUP BY
        CustomerId
) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
    T.TransactionItemsCountBucket
```

2. Improve the query by adding support to calculate the lower margin of the first per-customer transactions items count bucket:

```
SELECT
    T.TransactionItemsCountBucket
    ,count(*) as CustomersCount
FROM
(
    SELECT
        CustomerId,
        (
            COUNT(*) -
            (
                SELECT
                    MIN(TransactionItemsCount)
                FROM
                (
                    SELECT
                        COUNT(*) as TransactionItemsCount
                    FROM
                        [wwi_perf].[Sale_Hash]
                    GROUP BY
                        CustomerId
                ) X
            )
        ) / 100 as TransactionItemsCountBucket
    FROM
        [wwi_perf].[Sale_Hash]
    GROUP BY
        CustomerId
) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
    T.TransactionItemsCountBucket
```

13.7.2 Task 2 - Improve the execution plan of the query with a materialized view

1. Run the query with the EXPLAIN directive (note the WITH_RECOMMENDATIONS option as well):

```
EXPLAIN WITH_RECOMMENDATIONS
SELECT
    T.TransactionItemsCountBucket
    ,count(*) as CustomersCount
FROM
(
    SELECT
        CustomerId,
        (
            COUNT(*) -
            (
                SELECT
                    MIN(TransactionItemsCount)
                FROM
                (
                    SELECT
                        COUNT(*) as TransactionItemsCount
                    FROM
                    [
                        [wwi_perf].[Sale_Hash]
                    ]
                    GROUP BY
                        CustomerId
                ) X
            )
        ) / 100 as TransactionItemsCountBucket
    FROM
        [
            [wwi_perf].[Sale_Hash]
        ]
        GROUP BY
            CustomerId
    ) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
    T.TransactionItemsCountBucket
```

2. Analyze the resulting execution plan. Take a close look to the <materialized_view_candidates> section which suggests possible materialized views you can create to improve the performance of the query.

```
<?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="5" number_distributions="60" number_distributions_per_node="12">
<sql>SELECT
    T.TransactionItemsCountBucket
    ,count(*) as CustomersCount
FROM
(
    SELECT
        CustomerId,
        (
            COUNT(*) -
            (
                SELECT
                    MIN(TransactionItemsCount)
                FROM
                (
                    SELECT
                        COUNT(*) as TransactionItemsCount
                    FROM
                    [
                        [wwi_perf].[Sale_Hash]
                    ]
                    GROUP BY
                        CustomerId
                ) X
            )
        ) / 100 as TransactionItemsCountBucket
    FROM
        [
            [wwi_perf].[Sale_Hash]
        ]
        GROUP BY
            CustomerId
    ) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
    T.TransactionItemsCountBucket
```

```

        ) X
    )
) / 100 as TransactionItemsCountBucket
FROM
    [wwi_perf].[Sale_Hash]
GROUP BY
    CustomerId
) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
    T.TransactionItemsCountBucket</sql>
<materialized_view_candidates>
    <materialized_view_candidates with_constants="False">CREATE MATERIALIZED VIEW View1 WITH (DIST
SELECT [SQLPool01].[wwi_perf].[Sale_Hash].[CustomerId] AS [Expr0],
    COUNT(*) AS [Expr1]
FROM [wwi_perf].[Sale_Hash]
GROUP BY [SQLPool01].[wwi_perf].[Sale_Hash].[CustomerId]</materialized_view_candidates>
</materialized_view_candidates>
<dsql_operations total_cost="0.0242811172881356" total_number_operations="9">
    <dsql_operation operation_type="RND_ID">
        <identifier>TEMP_ID_99</identifier>
    </dsql_operation>
    <dsql_operation operation_type="ON">
        <location permanent="false" distribution="AllComputeNodes" />
        <sql_operations>
            <sql_operation type="statement">CREATE TABLE [qtabledb].[dbo].[TEMP_ID_99] ([col] INT ) WI
        </sql_operations>
    </dsql_operation>
    <dsql_operation operation_type="BROADCAST_MOVE">
        <operation_cost cost="0.00096" accumulative_cost="0.00096" average_rowsize="4" output_rows="1" />
        <source_statement>SELECT [T1_1].[col] AS [col] FROM (SELECT MIN([T2_1].[col]) AS [col] FROM (S
OPTION (MAXDOP 6, MIN_GRANT_PERCENT = [MIN_GRANT], DISTRIBUTED_MOVE(N''))</source_statement>
        <destination_table>[TEMP_ID_99]</destination_table>
    </dsql_operation>
    <dsql_operation operation_type="RND_ID">
        <identifier>TEMP_ID_100</identifier>
    </dsql_operation>
    <dsql_operation operation_type="ON">
        <location permanent="false" distribution="AllDistributions" />
        <sql_operations>
            <sql_operation type="statement">CREATE TABLE [qtabledb].[dbo].[TEMP_ID_100] ([col] INT, [co
        </sql_operations>
    </dsql_operation>
    <dsql_operation operation_type="SHUFFLE_MOVE">
        <operation_cost cost="0.0233211172881356" accumulative_cost="0.0242811172881356" average_rowsi
        <source_statement>SELECT [T1_1].[col1] AS [col], [T1_1].[col] AS [col1] FROM (SELECT COUNT_BIG(
(SELECT COUNT(CAST ((0) AS INT)) AS [col] FROM [SQLPool01].[wwi_perf].[Sale_Hash] AS T4_1 GROUP BY
ON (0 = 0)) AS T2_1 GROUP BY [T2_1].[col]) AS T1_1
OPTION (MAXDOP 6, MIN_GRANT_PERCENT = [MIN_GRANT], DISTRIBUTED_MOVE(N''))</source_statement>
        <destination_table>[TEMP_ID_100]</destination_table>
        <shuffle_columns>col;</shuffle_columns>
    </dsql_operation>
    <dsql_operation operation_type="RETURN">
        <location distribution="AllDistributions" />
        <select>SELECT [T1_1].[col1] AS [col], [T1_1].[col] AS [col1] FROM (SELECT CONVERT (INT, [T2_1]
OPTION (MAXDOP 6, MIN_GRANT_PERCENT = [MIN_GRANT]))</select>
    </dsql_operation>
    <dsql_operation operation_type="ON">
        <location permanent="false" distribution="AllDistributions" />
        <sql_operations>

```

```
<sql_operation type="statement">DROP TABLE [qtabledb].[dbo].[TEMP_ID_100]</sql_operation>
</sql_operations>
</dsql_operation>
<dsql_operation operation_type="ON">
<location permanent="false" distribution="AllComputeNodes" />
<sql_operations>
    <sql_operation type="statement">DROP TABLE [qtabledb].[dbo].[TEMP_ID_99]</sql_operation>
</sql_operations>
</dsql_operation>
</dsql_operations>
</dsql_query>
```

3. Create the suggested materialized view:

```
CREATE MATERIALIZED VIEW
    mvTransactionItemsCounts
WITH
(
    DISTRIBUTION = HASH([CustomerId])
)
AS
SELECT
    CustomerId
    ,COUNT(*) AS ItemsCount
FROM
    [wwi_perf].[Sale_Hash]
GROUP BY
    CustomerId
```

4. Check the execution plan again:

```
EXPLAIN WITH_RECOMMENDATIONS
SELECT
    T.TransactionItemsCountBucket
    ,count(*) as CustomersCount
FROM
(
    SELECT
        CustomerId,
        (
            COUNT(*) -
            (
                SELECT
                    MIN(TransactionItemsCount)
                FROM
                (
                    SELECT
                        COUNT(*) as TransactionItemsCount
                    FROM
                        [wwi_perf].[Sale_Hash]
                    GROUP BY
                        CustomerId
                ) X
            )
        ) / 100 as TransactionItemsCountBucket
    FROM
        [wwi_perf].[Sale_Hash]
    GROUP BY
        CustomerId
) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
```

T.TransactionItemsCountBucket

The resulting execution plan indicates now the use of the `mvTransactionItemsCounts` (the `BROADCAST_MOVE` distributed SQL operation) materialized view which provides improvements to the query execution time:

```

<?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="5" number_distributions="60" number_distributions_per_node="12">
<sql>SELECT
    T.TransactionItemsCountBucket
    ,count(*) as CustomersCount
FROM
(
    SELECT
        CustomerId,
        (
            COUNT(*) -
            (
                SELECT
                    MIN(TransactionItemsCount)
                FROM
                (
                    SELECT
                        COUNT(*) as TransactionItemsCount
                    FROM
                        [wwi_perf].[Sale_Hash]
                    GROUP BY
                        CustomerId
                ) X
            )
        ) / 100 as TransactionItemsCountBucket
    FROM
        [wwi_perf].[Sale_Hash]
    GROUP BY
        CustomerId
) T
GROUP BY
    T.TransactionItemsCountBucket
ORDER BY
    T.TransactionItemsCountBucket</sql>
<materialized_view_candidates>
    <materialized_view_candidates with_constants="False">CREATE MATERIALIZED VIEW View1 WITH (DIST
SELECT [SQLPool01].[wwi_perf].[Sale_Hash].[CustomerId] AS [Expr0] ,
    COUNT(*) AS [Expr1]
FROM [wwi_perf].[Sale_Hash]
GROUP BY [SQLPool01].[wwi_perf].[Sale_Hash].[CustomerId]</materialized_view_candidates>
</materialized_view_candidates>
<dsql_operations total_cost="0.0242811172881356" total_number_operations="9">
    <dsql_operation operation_type="RND_ID">
        <identifier>TEMP_ID_111</identifier>
    </dsql_operation>
    <dsql_operation operation_type="ON">
        <location permanent="false" distribution="AllComputeNodes" />
    <sql_operations>
        <sql_operation type="statement">CREATE TABLE [qtabledb].[dbo].[TEMP_ID_111] ([col] INT ) W
    </sql_operations>
    </dsql_operation>
    <dsql_operation operation_type="BROADCAST_MOVE">
        <operation_cost cost="0.00096" accumulative_cost="0.00096" average_rowsize="4" output_rows="1">
            <source_statement>SELECT [T1_1].[col] AS [col] FROM (SELECT MIN([T2_1].[col]) AS [col] FROM (S
OPTION (MAXDOP 6, MIN_GRANT_PERCENT = [MIN_GRANT], DISTRIBUTED_MOVE(N''))</source_statement>
        <destination_table>[TEMP_ID_111]</destination_table>
    </dsql_operation>
</dsql_operations>

```

```

    </dsql_operation>
    <dsql_operation operation_type="RND_ID">
        <identifier>TEMP_ID_112</identifier>
    </dsql_operation>
    <dsql_operation operation_type="ON">
        <location permanent="false" distribution="AllDistributions" />
        <sql_operations>
            <sql_operation type="statement">CREATE TABLE [qtabledb].[dbo].[TEMP_ID_112] ([col] INT, [co</pre>
```

13.8 Exercise 5 - Avoid extensive logging

13.8.1 Task 1 - Explore rules for minimally logged operations

The following operations are capable of being minimally logged:

- CREATE TABLE AS SELECT (CTAS)
- INSERT..SELECT
- CREATE INDEX
- ALTER INDEX REBUILD
- DROP INDEX
- TRUNCATE TABLE
- DROP TABLE
- ALTER TABLE SWITCH PARTITION

Minimal logging with bulk load

CTAS and INSERT...SELECT are both bulk load operations. However, both are influenced by the target table definition and depend on the load scenario. The following table explains when bulk operations are fully or minimally logged:

Primary Index	Load Scenario	Logging Mode
Heap	Any	Minimal

Primary Index	Load Scenario	Logging Mode
Clustered Index	Empty target table	Minimal
Clustered Index	Loaded rows do not overlap with existing pages in target	Minimal
Clustered Index	Loaded rows overlap with existing pages in target	Full
Clustered Columnstore Index	Batch size \geq 102,400 per partition aligned distribution	Minimal
Clustered Columnstore Index	Batch size $<$ 102,400 per partition aligned distribution	Full

It is worth noting that any writes to update secondary or non-clustered indexes will always be fully logged operations.

IMPORTANT

A Synapse Analytics SQL pool has 60 distributions. Therefore, assuming all rows are evenly distributed and landing in a single partition, your batch will need to contain 6,144,000 rows or larger to be minimally logged when writing to a Clustered Columnstore Index. If the table is partitioned and the rows being inserted span partition boundaries, then you will need 6,144,000 rows per partition boundary assuming even data distribution. Each partition in each distribution must independently exceed the 102,400 row threshold for the insert to be minimally logged into the distribution.

Loading data into a non-empty table with a clustered index can often contain a mixture of fully logged and minimally logged rows. A clustered index is a balanced tree (b-tree) of pages. If the page being written to already contains rows from another transaction, then these writes will be fully logged. However, if the page is empty then the write to that page will be minimally logged.

13.8.2 Task 2 - Optimizing a delete operation

1. Check the number of transaction items for customers with ids lower than 900000 using the following query:

```
SELECT
    COUNT_BIG(*) AS TransactionItemsCount
FROM
    [wwi_perf].[Sale_Hash]
WHERE
    CustomerId < 900000
```

2. Implement a minimal logging approach to delete transaction items for customers with ids lower than 900000. Use the following CTAS query to isolate the transaction items that should be kept:

```
CREATE TABLE [wwi_perf].[Sale_Hash_v2]
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    HEAP
)
AS
SELECT
    *
FROM
    [wwi_perf].[Sale_Hash]
WHERE
    CustomerId >= 900000
```

The query should execute within about 90 seconds. All that would remain to complete the process would be to delete the `Sale_Heap` table and rename `Sale_Heap_v2` to `Sale_Heap`.

3. Compare the previous operation with a classical delete:

```
DELETE
    [wwi_perf].[Sale_Hash]
WHERE
    CustomerId < 900000
```

Note

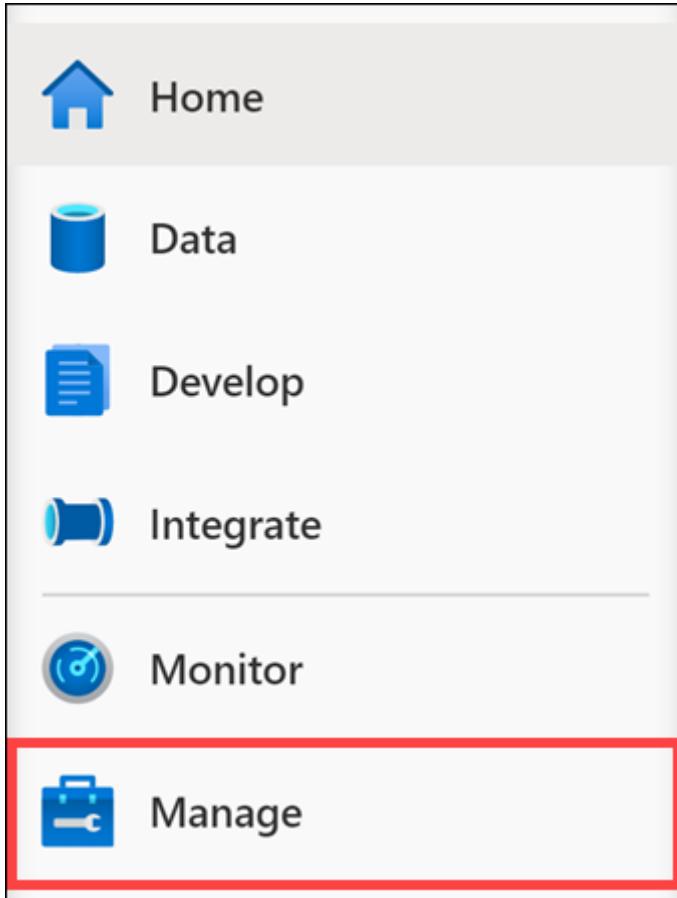
The query will run for a potentially long time (>12 minutes). Once the time exceeds significantly the time to run the previous CTAS query, you can cancel it (as you can already see the benefit of the CTAS-based approach).

13.9 Exercise 6: Cleanup

Complete these steps to free up resources you no longer need.

13.9.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



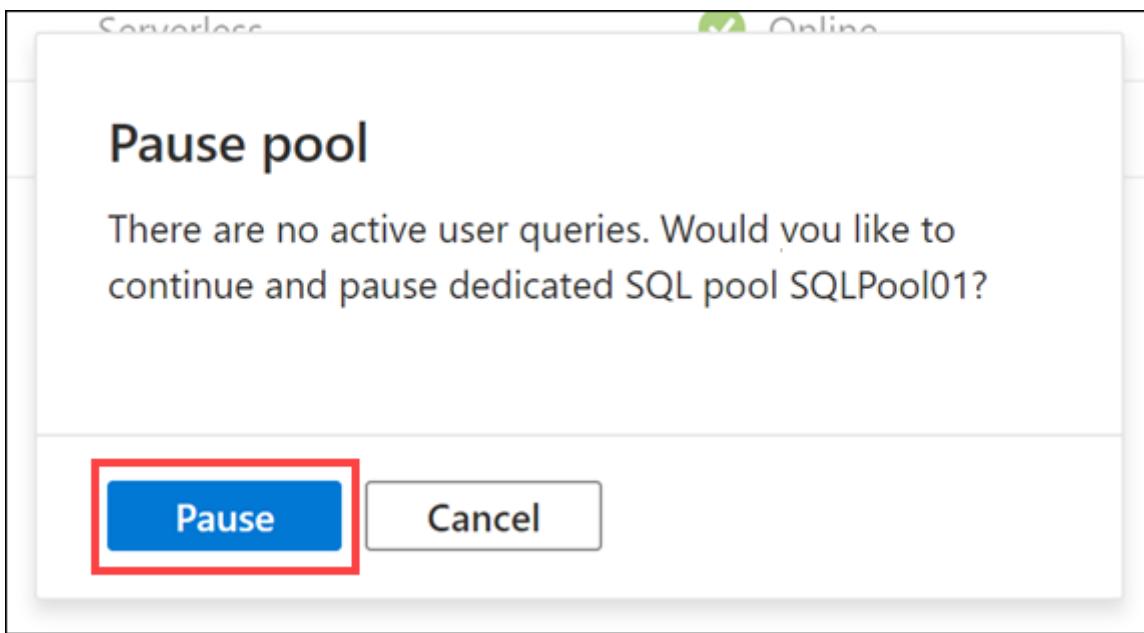
3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

A screenshot of the 'SQL pools' blade in Synapse Studio. On the left, there's a sidebar with 'Analytics pools' (selected), 'SQL pools' (highlighted with a red box and labeled '1'), 'Apache Spark pools', 'External connections', 'Linked services', 'Azure Purview (Preview)', 'Integration' (with 'Triggers' and 'Integration runtimes' listed), and 'Security'. The main area shows a table titled 'SQL pools' with the following data:

Name	Type	Status
Built-in	Serverless	✓ Online
SQLPool01	Dedicated	✓ Online

The 'SQLPool01' row has a 'Pause' button at the bottom right, which is highlighted with a red box and labeled with a red circle containing the number '2'.

4. When prompted, select **Pause**.



14 Module 12 - Support Hybrid Transactional Analytical Processing (HTAP) with Azure Synapse Link

In this module, students will learn how Azure Synapse Link enables seamless connectivity of an Azure Cosmos DB account to a Synapse workspace. The student will understand how to enable and configure Synapse link, then how to query the Azure Cosmos DB analytical store using Apache Spark and SQL Serverless.

In this module, the student will be able to:

- Configure Azure Synapse Link with Azure Cosmos DB
- Query Azure Cosmos DB with Apache Spark for Synapse Analytics
- Query Azure Cosmos DB with serverless SQL pool for Azure Synapse Analytics

14.1 Lab details

- [Module 12 - Support Hybrid Transactional Analytical Processing \(HTAP\) with Azure Synapse Link](#)
 - [Lab details](#)
 - [Lab setup and pre-requisites](#)
 - [Exercise 1: Lab setup](#)
 - * [Task 1: Create linked service](#)
 - * [Task 2: Create dataset](#)
 - [Exercise 2: Configuring Azure Synapse Link with Azure Cosmos DB](#)
 - * [Task 1: Enable Azure Synapse Link](#)
 - * [Task 2: Create a new Azure Cosmos DB container](#)
 - * [Task 3: Create and run a copy pipeline](#)
 - [Exercise 3: Querying Azure Cosmos DB with Apache Spark for Synapse Analytics](#)
 - * [Task 1: Create a notebook](#)
 - [Exercise 4: Querying Azure Cosmos DB with serverless SQL pool for Azure Synapse Analytics](#)
 - * [Task 1: Create a new SQL script](#)

14.2 Lab setup and pre-requisites

Note: Only complete the [Lab setup and pre-requisites](#) steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 1.

Complete the lab setup instructions for this module.

Note, the following modules share this same environment:

- [Module 4](#)

- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

14.3 Exercise 1: Lab setup

14.3.1 Task 1: Create linked service

Complete the steps below to create an Azure Cosmos DB linked service.

Note: Skip this section if you have already created the following within this environment in a previous module:

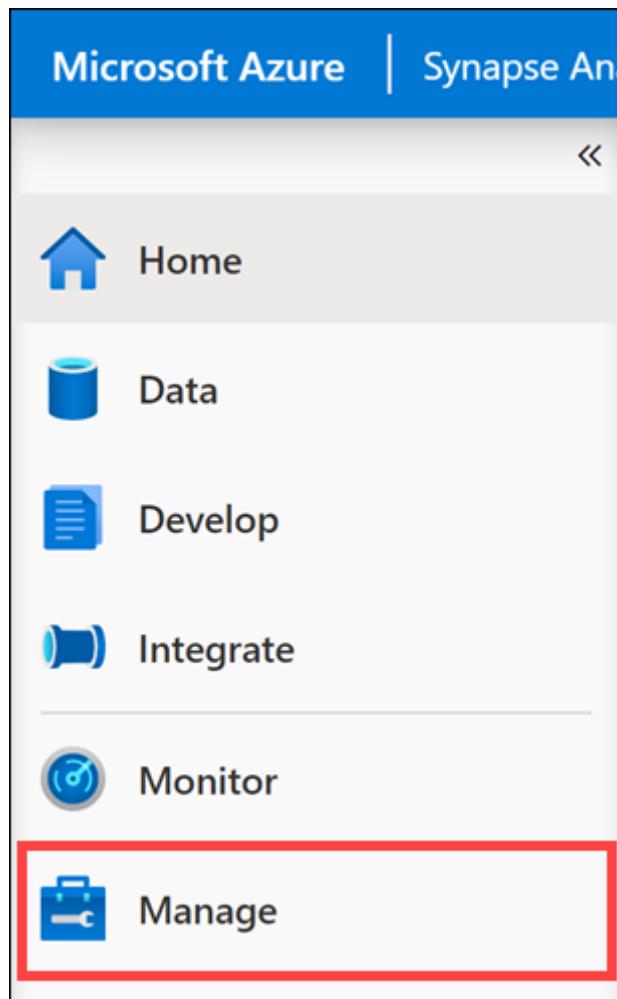
Linked service:

- `asacosmosdb01` (Cosmos DB)

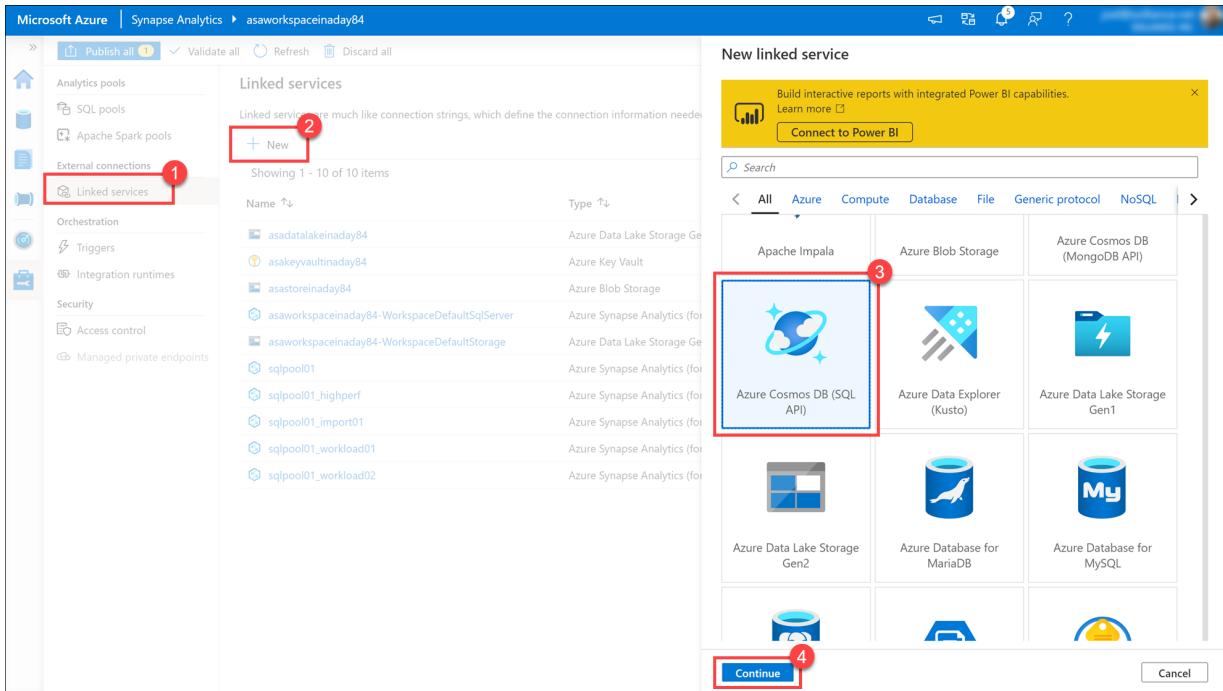
Integration dataset:

- `asal400_customerprofile_cosmosdb`

1. Open Synapse Studio (<https://web.azuresynapse.net/>), and then navigate to the **Manage** hub.



2. Open **Linked services** and select **+ New** to create a new linked service. Select **Azure Cosmos DB (SQL API)** in the list of options, then select **Continue**.



3. Name the linked service **asacosmosdb01** (1), select the **Cosmos DB account name** (**asacosmosdbSUFFIX**) and set the **Database name** value to **CustomerProfile** (2). Select **Test connection** to ensure success (3), then select **Create** (4).

New linked service (Azure Cosmos DB (SQL API))

Choose a name for your linked service. This name cannot be updated later.

Name * asacosmosdb01 1

Description

Connect via integration runtime * (i)

AutoResolveIntegrationRuntime ▼ edit

Connection string Azure Key Vault

Account selection method (i)

From Azure subscription Enter manually

Azure subscription (i)
Select all ▼

Azure Cosmos DB account name * (i) asacosmosdbinaday84 2

Database name * (i) CustomerProfile ▼ refresh

Additional connection properties

+ New

Annotations

I ..

Create Back 4

Connection successful 3

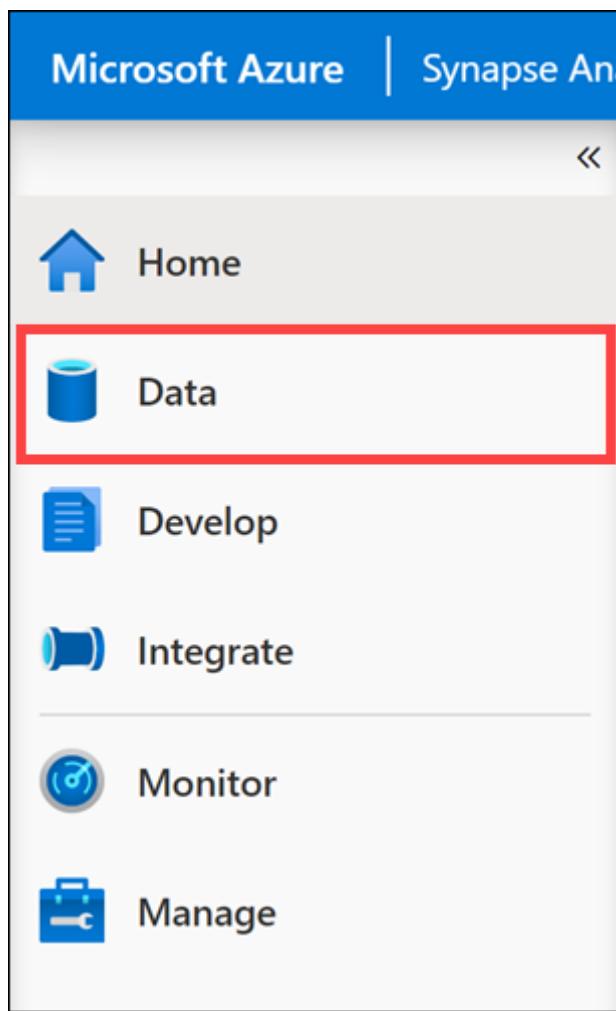
Test connection Cancel

14.3.2 Task 2: Create dataset

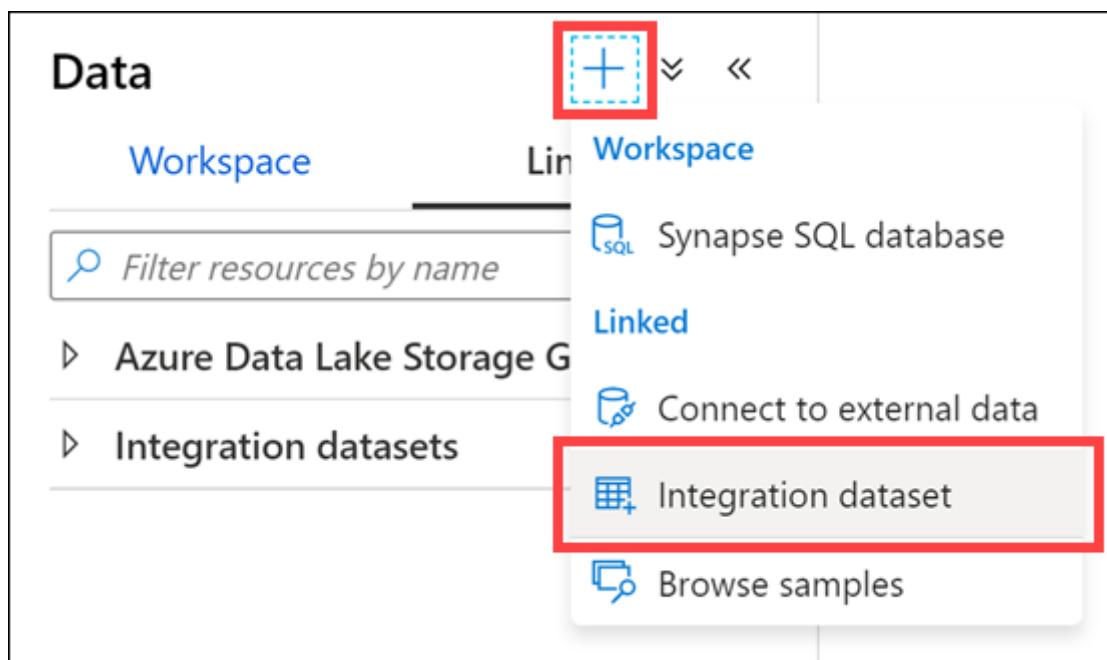
Complete the steps below to create the `asal400_customerprofile_cosmosdb` dataset.

Note to presenter: Skip this section if you have already completed Module 4.

1. Navigate to the **Data** hub.



2. Select + in the toolbar (1), then select **Integration dataset** (2) to create a new dataset.



3. Select **Azure Cosmos DB (SQL API)** from the list (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

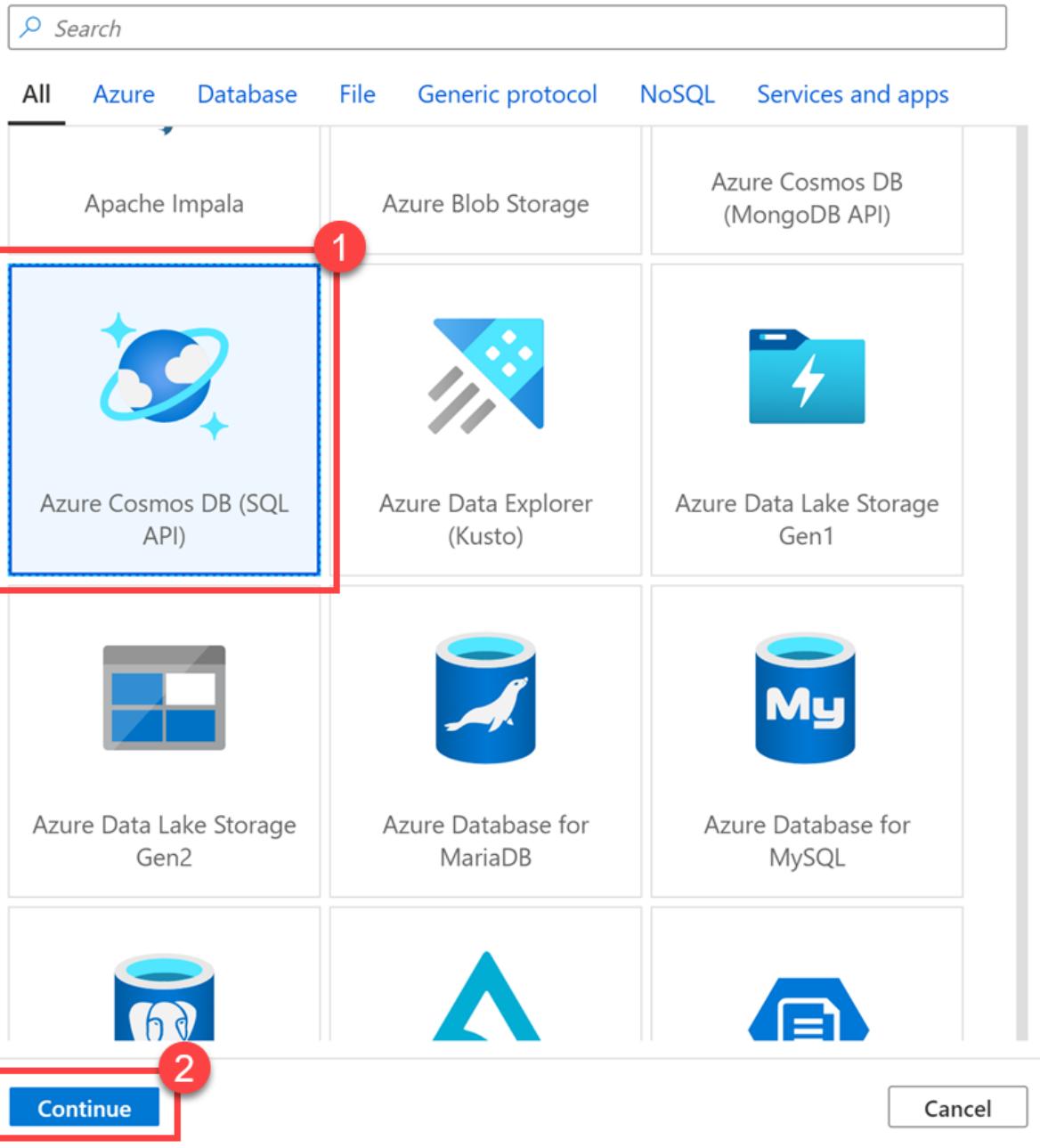
Search

All Azure Database File Generic protocol NoSQL Services and apps

Apache Impala	Azure Blob Storage	Azure Cosmos DB (MongoDB API)
 Azure Cosmos DB (SQL API)		
	Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen1
Azure Data Lake Storage Gen2	Azure Database for MariaDB	Azure Database for MySQL
		
Continue		Cancel

1

2



4. Configure the dataset with the following characteristics, then select **OK** (4):

- **Name:** Enter `asal400_customerprofile_cosmosdb` (1).
- **Linked service:** Select the Azure Cosmos DB linked service (2).
- **Collection:** Select `OnlineUserProfile01` (3).

Set properties

Choose a name for your dataset. This name can be updated at any time until it is published.

1 Name
asal400_customerprofile_cosmosdb

2 Linked service *
asacosmosdb01

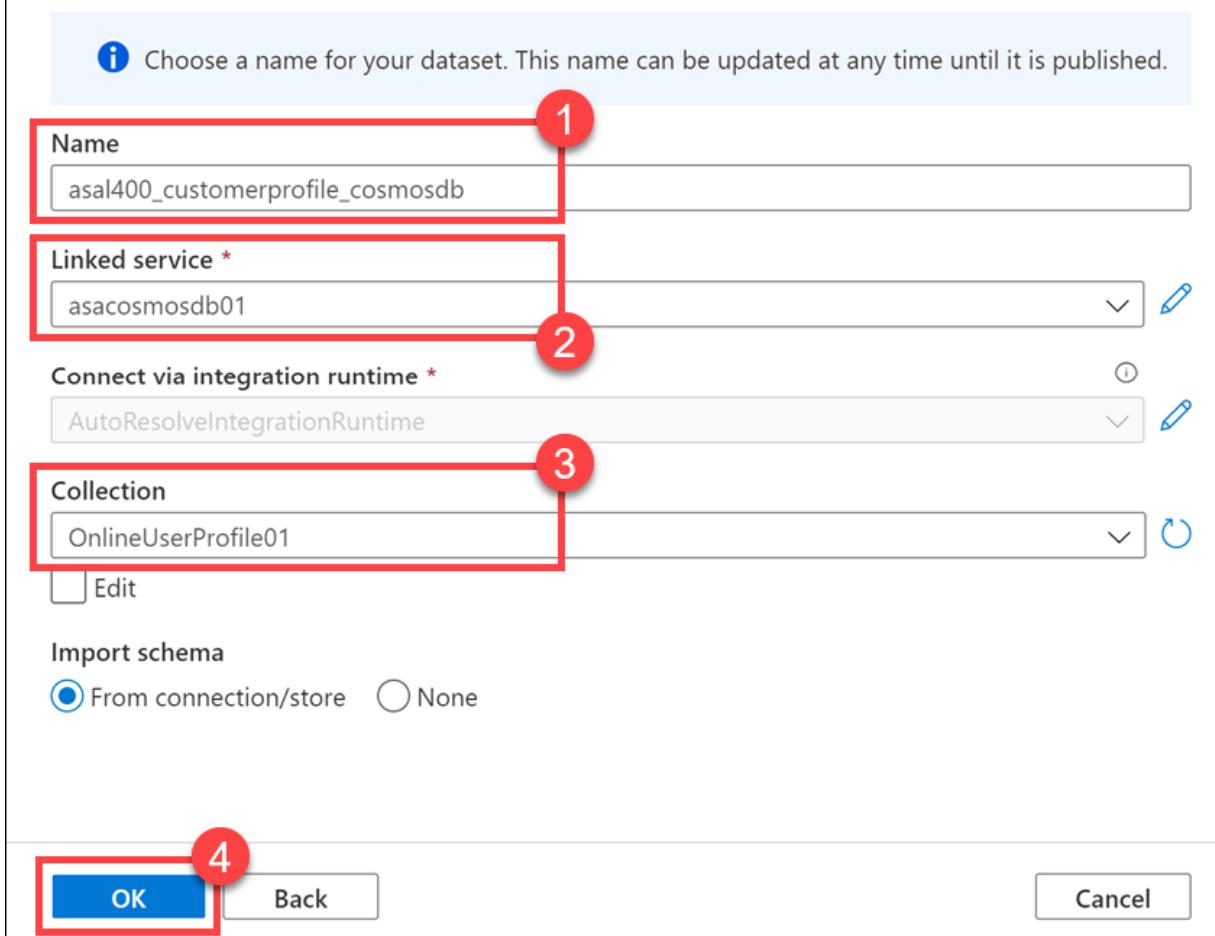
Connect via integration runtime *
AutoResolveIntegrationRuntime

3 Collection
OnlineUserProfile01

Edit

Import schema
 From connection/store None

4 OK Back Cancel



- After creating the dataset, select **Preview data** under its **Connection** tab.

CosmosDB Collection (SQL API)
asal400_customerprofile_cosmosdb

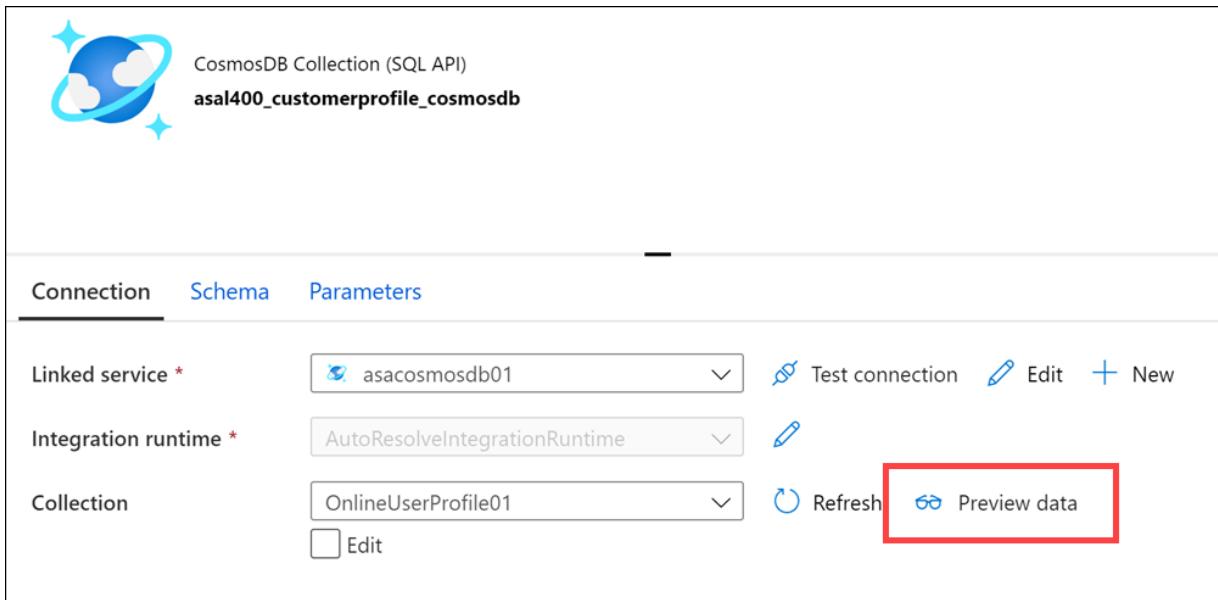
Connection Schema Parameters

Linked service * asacosmosdb01 Test connection Edit New

Integration runtime * AutoResolveIntegrationRuntime

Collection OnlineUserProfile01 Refresh Preview data

Edit



- Preview data queries the selected Azure Cosmos DB collection and returns a sample of the documents within. The documents are stored in JSON format and include a `userId` field, `cartId`, `preferredProducts` (an array of product IDs that may be empty), and `productReviews` (an array of written product reviews that may be empty).

The screenshot shows the 'Preview data' interface in Azure Synapse Analytics. It displays a JSON object representing a user profile from a linked service named 'asacosmosdb01'. The object is named 'OnlineUserProfile01' and contains fields such as 'userId', 'cartId', 'preferredProducts' (an array of product IDs), and 'productReviews' (an array of review objects). One review entry is shown in detail, including its product ID, review text ("heard about this on brazilian radio, decided to give it a try."), and review date ("2019-06-08T22:42:56.0052067+00:00").

```

Linked service: asacosmosdb01
Object: OnlineUserProfile01

[
  {
    "userId": 4193,
    "cartId": "2db8e371-dac3-4318-afc4-3d2fae5334e8",
    "preferredProducts": [
      1747,
      4740,
      315,
      4337,
      2913,
      4891
    ],
    "productReviews": [
      {
        "productId": 3052,
        "reviewText": "heard about this on brazilian radio, decided to give it a try.",
        "reviewDate": "2019-06-08T22:42:56.0052067+00:00"
      },
      {
        "productId": 1360,
        "reviewText": "i use it daily when i'm in my courthouse.",
        "reviewDate": "2017-04-29T08:43:26.6771565+00:00"
      }
    ]
  }
]

```

- Select **Publish all** then **Publish** to save your new resources.



14.4 Exercise 2: Configuring Azure Synapse Link with Azure Cosmos DB

Tailwind Traders uses Azure Cosmos DB to store user profile data from their eCommerce site. The NoSQL document store provided by the Azure Cosmos DB SQL API provides the familiarity of managing their data using SQL syntax, while being able to read and write the files at a massive, global scale.

While Tailwind Traders is happy with the capabilities and performance of Azure Cosmos DB, they are concerned about the cost of executing a large volume of analytical queries over multiple partitions (cross-partition queries) from their data warehouse. They want to efficiently access all the data without needing to increase the Azure Cosmos DB request units (RUs). They have looked at options for extracting data from their containers to the data lake as it changes, through the Azure Cosmos DB change feed mechanism. The problem with this approach is the extra service and code dependencies and long-term maintenance of the solution. They could perform bulk exports from a Synapse Pipeline, but then they won't have the most up-to-date information at any given moment.

You decide to enable Azure Synapse Link for Cosmos DB and enable the analytical store on their Azure Cosmos DB containers. With this configuration, all transactional data is automatically stored in a fully isolated column store. This store enables large-scale analytics against the operational data in Azure Cosmos DB, without impacting the transactional workloads or incurring resource unit (RU) costs. Azure Synapse Link for Cosmos DB creates a tight integration between Azure Cosmos DB and Azure Synapse Analytics, which enables Tailwind Traders to run near real-time analytics over their operational data with no-ETL and full performance isolation from their transactional workloads.

By combining the distributed scale of Cosmos DB's transactional processing with the built-in analytical store and the computing power of Azure Synapse Analytics, Azure Synapse Link enables a Hybrid Transactional/Analytical Processing (HTAP) architecture for optimizing Tailwind Trader's business processes. This integration eliminates ETL processes, enabling business analysts, data engineers & data scientists to self-serve and run near real-time BI, analytics, and Machine Learning pipelines over operational data.

14.4.1 Task 1: Enable Azure Synapse Link

1. Navigate to the Azure portal (<https://portal.azure.com>) and open the resource group for your lab environment.
2. Select the **Azure Cosmos DB** account.

Name ↑↓	Type ↑↓
amlworkspaceinaday84	Machine Learning
asaappinsightsinaday84	Application Insights
asacosmosdbinaday84	Azure Cosmos DB account
asadatalakeinaday84	Storage account

3. Select **Features** in the left-hand menu (1), then select **Azure Synapse Link** (2).

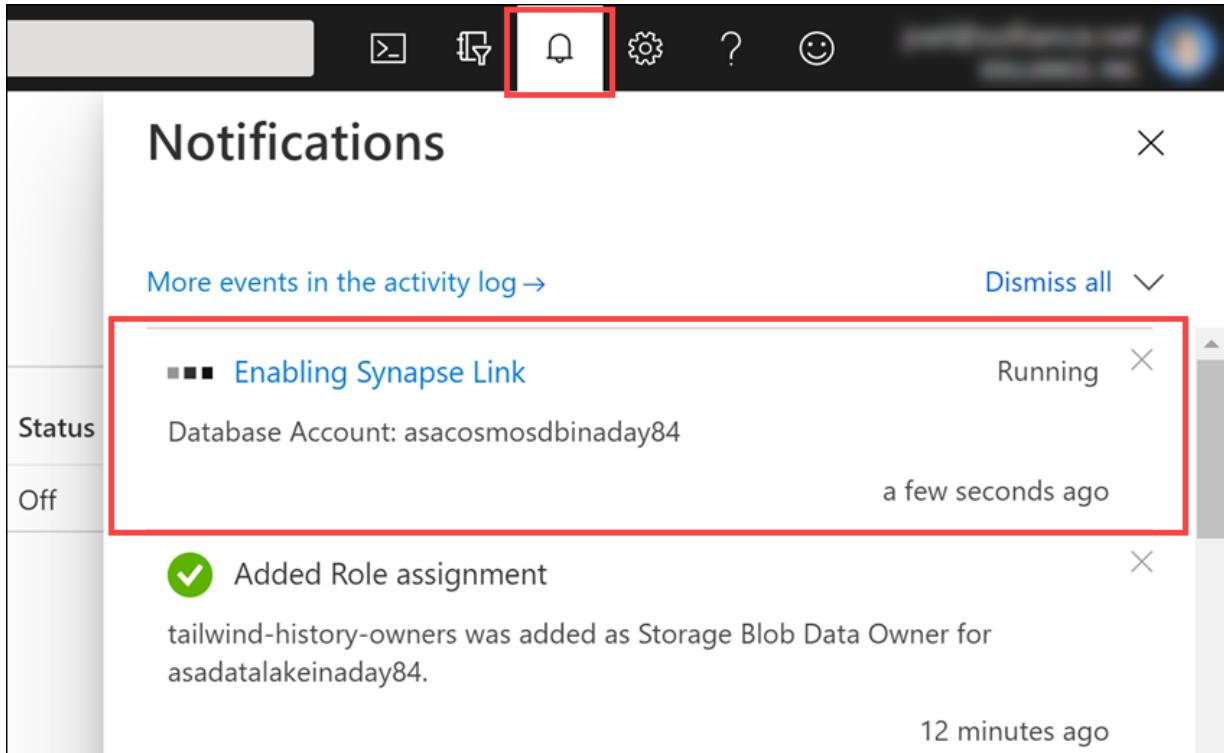
The screenshot shows the 'asacosmosdbinaday84 | Features' page for an Azure Cosmos DB account. On the left, there is a sidebar with 'Settings' and three options: 'Features' (marked with a red box and number 1), 'Data Explorer', and 'Replicate data globally'. The main area has a search bar and a refresh button. A table lists features with columns 'Feature' and 'Status'. The 'Azure Synapse Link' row is highlighted with a red box and number 2, showing it is currently 'Off'.

4. Select **Enable**.

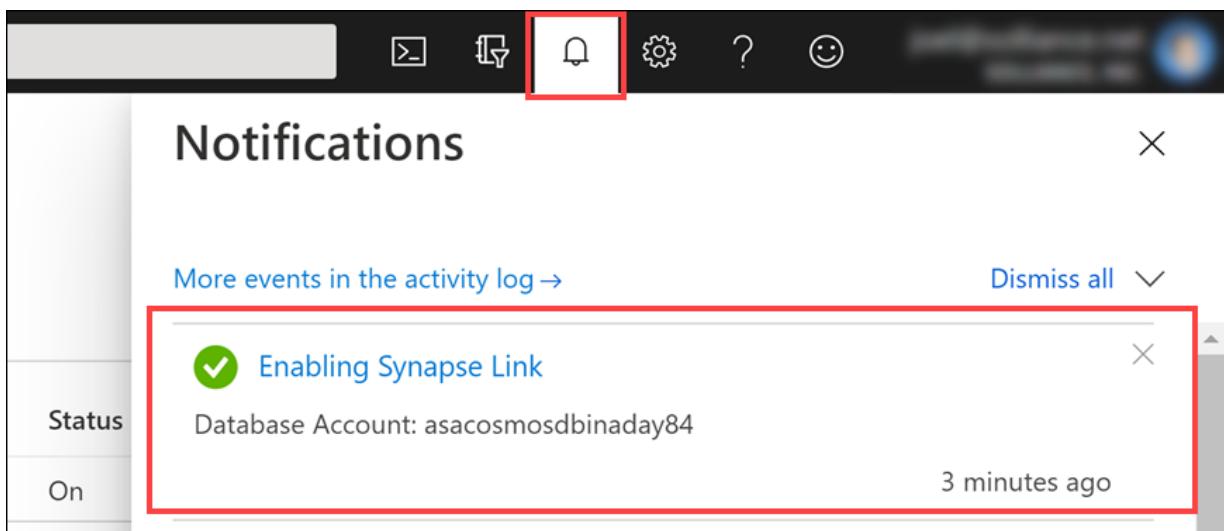
The screenshot shows the 'Azure Synapse Link' dialog box. It contains a descriptive text about the feature, followed by a detailed explanation of its benefits. At the bottom, there are two buttons: 'Learn More' and 'Enable' (which is highlighted with a red box), and a 'Close' button.

Before we can create an Azure Cosmos DB container with an analytical store, we must first enable Azure Synapse Link.

5. You must wait for this operation to complete before continuing, which should take about a minute. Check the status by selecting the Azure **Notifications** icon.



You will see a green checkmark next to "Enabling Synapse Link" when it successfully completes.

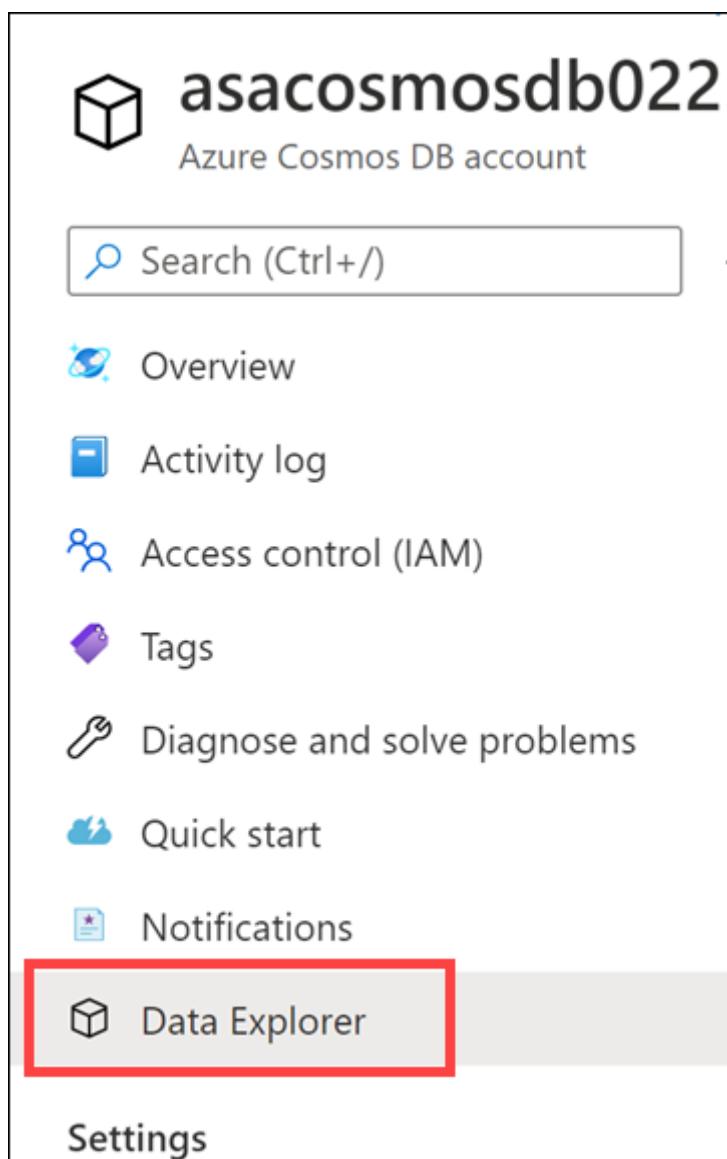


14.4.2 Task 2: Create a new Azure Cosmos DB container

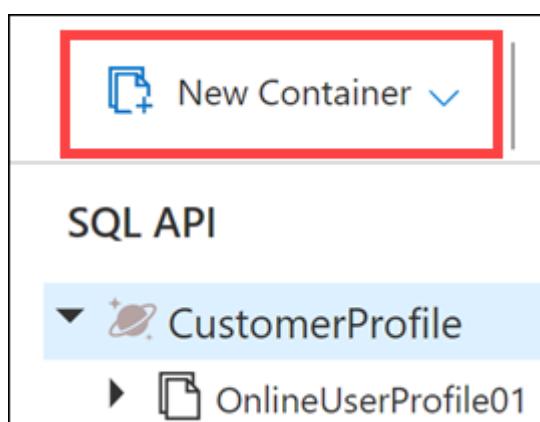
Tailwind Traders has an Azure Cosmos DB container named `OnlineUserProfile01`. Since we enabled the Azure Synapse Link feature *after* the container was already created, we cannot enable the analytical store on the container. We will create a new container that has the same partition key and enable the analytical store.

After creating the container, we will create a new Synapse Pipeline to copy data from the `OnlineUserProfile01` container to the new one.

1. Select **Data Explorer** on the left-hand menu.



2. Select New Container.



3. For Database id, select Use existing, then select CustomerProfile (1). Enter UserProfileHTAP for the Container id (2), then enter /userId for the Partition key (3). For Throughput, select Autoscale (4), then enter 4000 for the Max RU/s value (5). Finally, expand Advanced and set Analytical store to On (6), then select OK.

New Container

X

* Database id ⓘ

Create new Use existing

CustomerProfile

1

* Container id ⓘ

UserProfileHTAP

2

* Partition key ⓘ

/userId

3

* Container throughput (autoscale) ⓘ

Autoscale Manual

Estimate your required RU/s with [capacity calculator](#).

Container max RU/s ⓘ

4000

4

Your container throughput will automatically scale from **400 RU/s (10% of max RU/s) - 4000 RU/s** based on usage.

Estimated monthly cost (USD) ⓘ: **\$35.04 - \$350.40** (1 region, 400 - 4000 RU/s, \$0.00012/RU)

Unique keys ⓘ

+ Add unique key

Advanced

My partition key is larger than 100 bytes

Analytical store ⓘ

On Off

6

Here we set the **partition key** value to **customerId**, because it is a field we use most often in queries and contains a relatively high cardinality (number of unique values) for good partitioning performance. We set the throughput to Autoscale with a maximum value of 4,000 request units (RUs). This means that the container will have a minimum of 400 RUs allocated (10% of the maximum number), and will scale up to a maximum of 4,000 when the scale engine detects a high enough demand to warrant increasing the throughput. Finally, we enable the **analytical store** on the container, which allows us to take full advantage of the Hybrid Transactional/Analytical Processing (HTAP) architecture from within Synapse Analytics.

Let's take a quick look at the data we will copy over to the new container.

4. Expand the **OnlineUserProfile01** container underneath the **CustomerProfile** database, then select **Items** (1). Select one of the documents (2) and view its contents (3). The documents are stored in JSON format.

The screenshot shows the Azure Cosmos DB SQL API interface. On the left, the navigation pane shows the CustomerProfile database with the OnlineUserProfile01 container expanded. The 'Items' item under the OnlineUserProfile01 container is highlighted with a red box and labeled '1'. A specific document row is selected with a red box and labeled '2'. The document's JSON content is displayed on the right, with line numbers 1 through 26. A red box highlights the 'productId' field in the JSON, and a red circle labeled '3' points to the account name 'asacosmosdbinaday84' in the top-left corner of the interface.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
{
    "userId": 4180,
    "cartId": "9a1b857d-45ee-44e3-95f2-3d8dbe24699c",
    "preferredProducts": [
        1907,
        668,
        3325,
        1684,
        4166,
        4095
    ],
    "productReviews": [
        {
            "productId": 2823,
            "reviewText": "this Buckinghamshire is perplexed.",
            "reviewDate": "2019-09-07T10:05:10.4820458+03:00"
        },
        {
            "productId": 396,
            "reviewText": "talk about surprise!!!",
            "reviewDate": "2019-11-17T06:39:19.5688794+02:00"
        },
        {
            "productId": 4912,
            "reviewText": "The box this comes in is 5 kilometer by 6 yard and",
            "reviewDate": "2018-01-16T01:47:31.9624911+02:00"
        }
    ]
}

```

5. Select **Keys** in the left-hand menu (1), then copy the **Primary Key** value (2) and save it to Notepad or similar for later reference. Copy the Azure Cosmos DB **account name** in the upper-left corner (3) and also save it to Notepad or similar text editor for later.

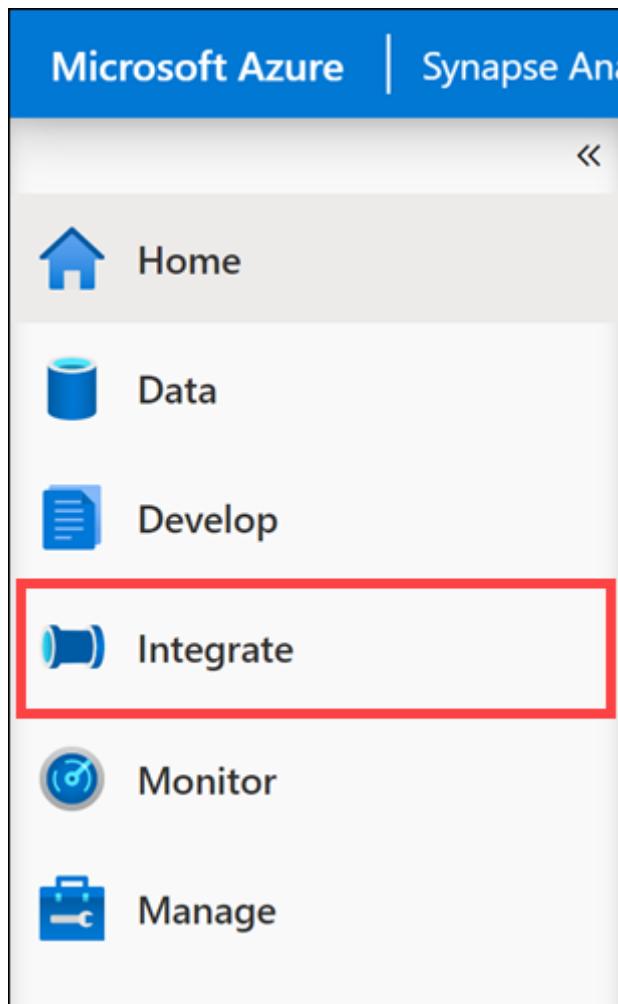
The screenshot shows the 'Keys' blade for the 'asacosmosdbinaday84' account. The 'Keys' item in the left sidebar is highlighted with a red box and labeled '1'. The 'PRIMARY KEY' field is selected with a red box and labeled '2'. A context menu is open over the 'PRIMARY KEY' field, with the 'Copied' option highlighted. The account name 'asacosmosdbinaday84' is visible in the top-left corner of the blade.

Note: Take note of these values. You will need this information when creating the SQL view toward the end of the demo.

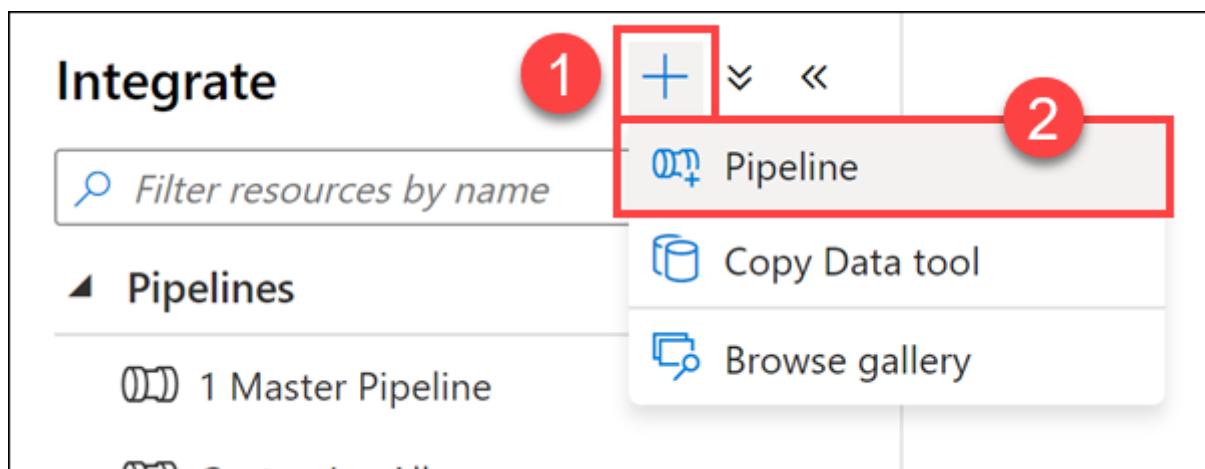
14.4.3 Task 3: Create and run a copy pipeline

Now that we have the new Azure Cosmos DB container with the analytical store enabled, we need to copy the contents of the existing container by using a Synapse Pipeline.

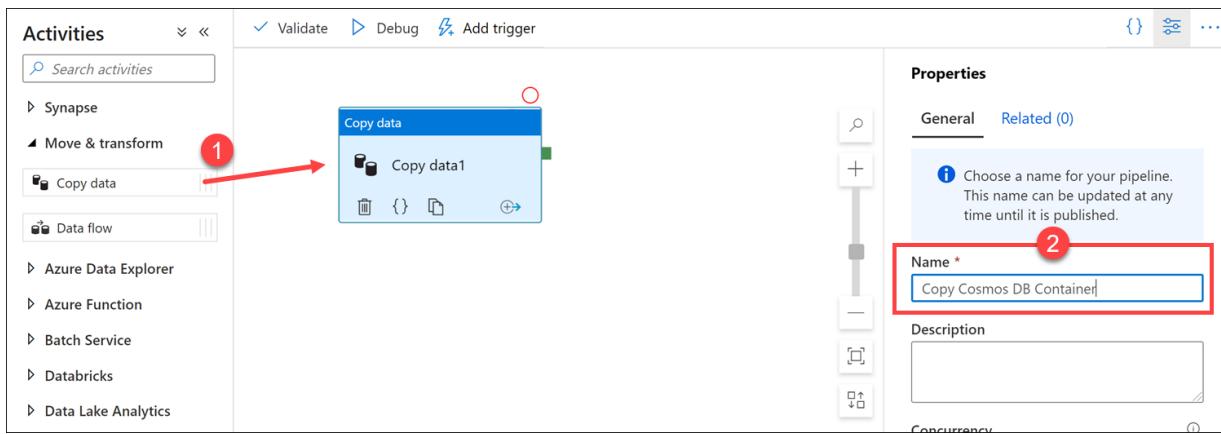
1. Open Synapse Studio (<https://web.azuresynapse.net/>), and then navigate to the **Integrate** hub.



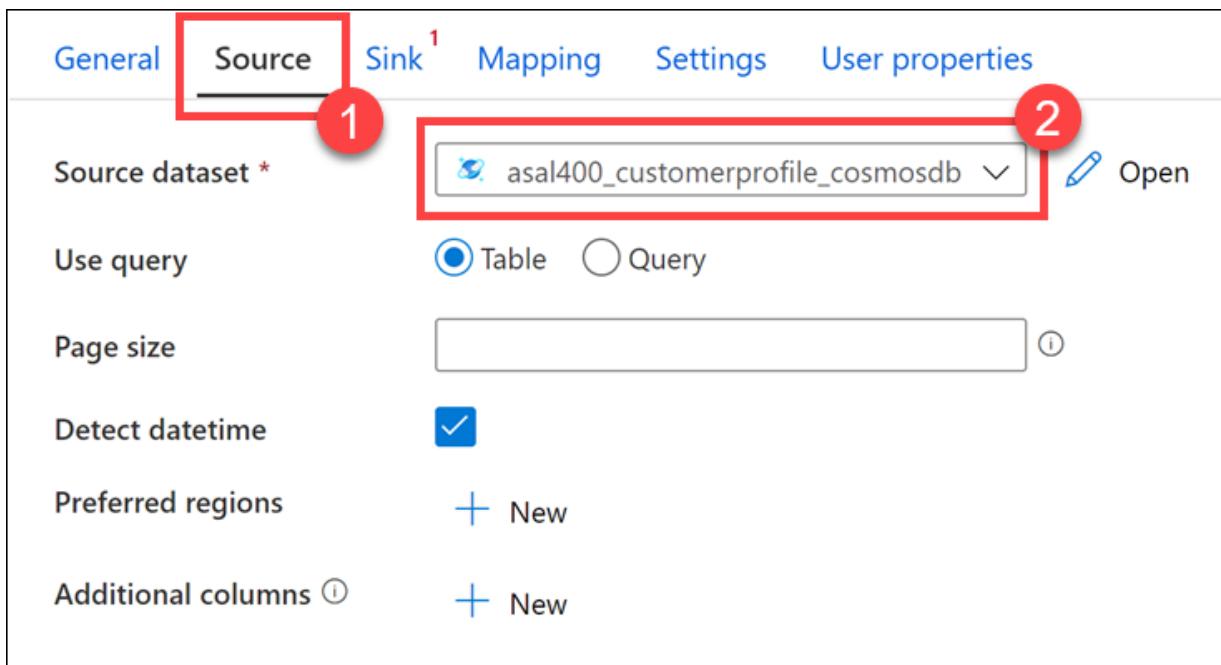
2. Select + (1), then Pipeline (2).



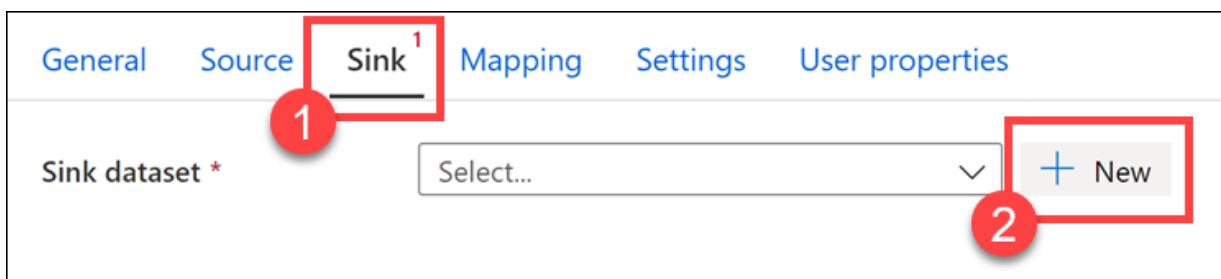
3. Under Activities, expand the Move & transform group, then drag the **Copy data** activity onto the canvas (1). Set the **Name** to **Copy Cosmos DB Container** in the Properties blade (2).



4. Select the new Copy activity that you added to the canvas, then select the **Source** tab (1). Select the `asal400_customerprofile_cosmosdb` source dataset from the list (2).



5. Select the **Sink** tab (1), then select **+ New** (2).



6. Select the **Azure Cosmos DB (SQL API)** dataset type (1), then select **Continue** (2).

New integration dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store.[Learn more](#)

Select a data store

The screenshot shows a grid of nine data store options. The 'Services and apps' tab is selected. The 'Azure Cosmos DB (SQL API)' option is highlighted with a red box and circled with a red number 1. The 'Continue' button at the bottom left is also highlighted with a red box and circled with a red number 2.

All	Azure	Database	File	Generic protocol	Services and apps
					Azure Cosmos DB (MongoDB API)
Azure Blob Storage					Azure Cosmos DB (SQL API)
Azure Data Explorer (Kusto)			Azure Data Lake Storage Gen1		Azure Data Lake Storage Gen2
MySQL			MongoDB		Apache Hadoop
Continue	2				Cancel

7. For **Name**, enter `cosmos_db_htap` (1). Select the `asacosmosdb01` (2) **Linked service**. Select the `UserProfileHTAP` (3) **Collection**. Select **From connection/store** under **Import schema** (4), then select **OK** (5).

Set properties

Name cosmos_db_htap 1

Linked service * asacosmosdb01 2

Connect via integration runtime * AutoResolveIntegrationRuntime i e

Collection UserProfileHTAP 3

Edit r

Import schema From connection/store 4 None

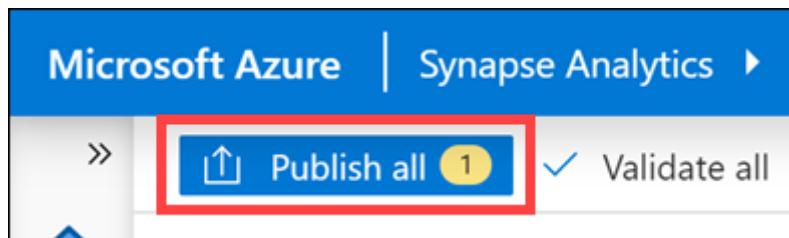
Advanced

OK Back Cancel

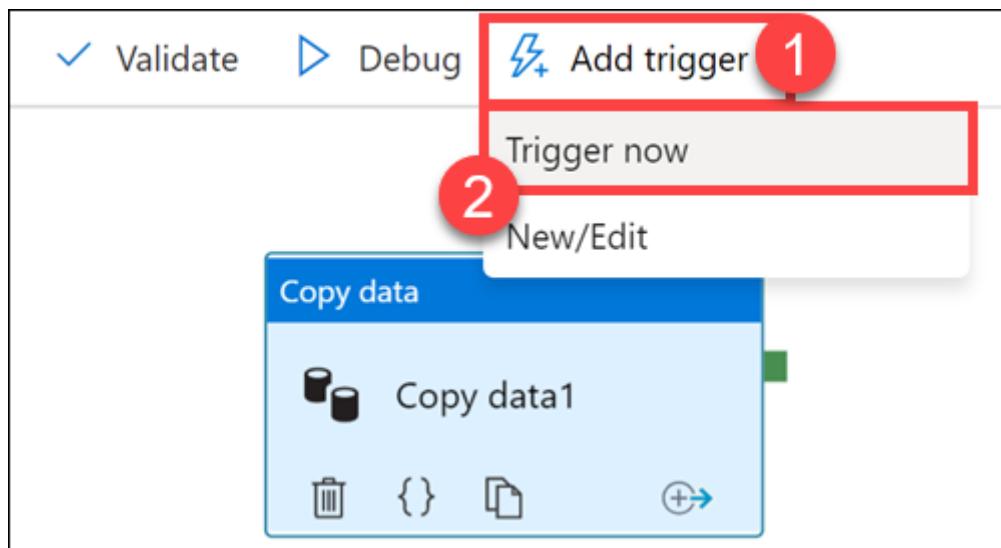
- Underneath the new sink dataset you just added, select the **Insert** write behavior.

General	Source	Sink	Mapping	Settings	User properties
<p>Sink dataset * cosmos_db_htap Open</p> <p>Write behavior Insert v</p> <p>Write batch timeout</p> <p>Write batch size</p> <p>Max concurrent connections i</p> <p>Disable performance metrics <input type="checkbox"/> i</p>					

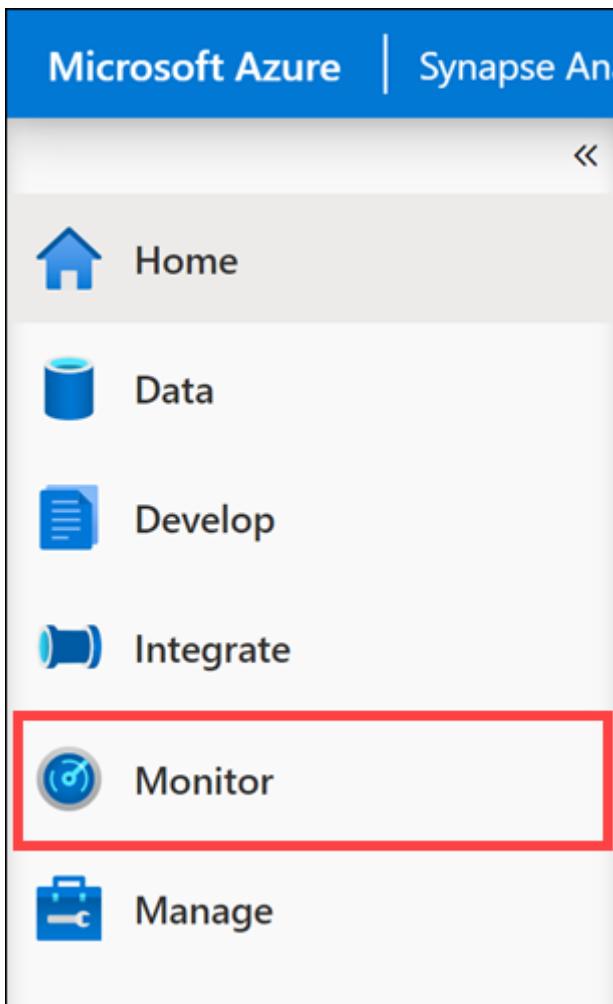
- Select **Publish all**, then **Publish** to save the new pipeline.



10. Above the pipeline canvas, select **Add trigger** (1), then **Trigger now** (2). Select **OK** to trigger the run.



11. Navigate to the **Monitor** hub.



12. Select **Pipeline runs** (1) and wait until the pipeline run has successfully completed (2). You may have to select **Refresh** (3) a few times.

Pipeline name	Run start	Run end	Duration	Triggered by	Status
Copy Cosmos DB Container	11/27/20, 11:13:08 AM	11/27/20, 11:16:58 AM	00:03:49	Manual trigger	Succeeded

This may take **around 4 minutes** to complete. While this is running, read the rest of the lab instructions to familiarize yourself with the content.

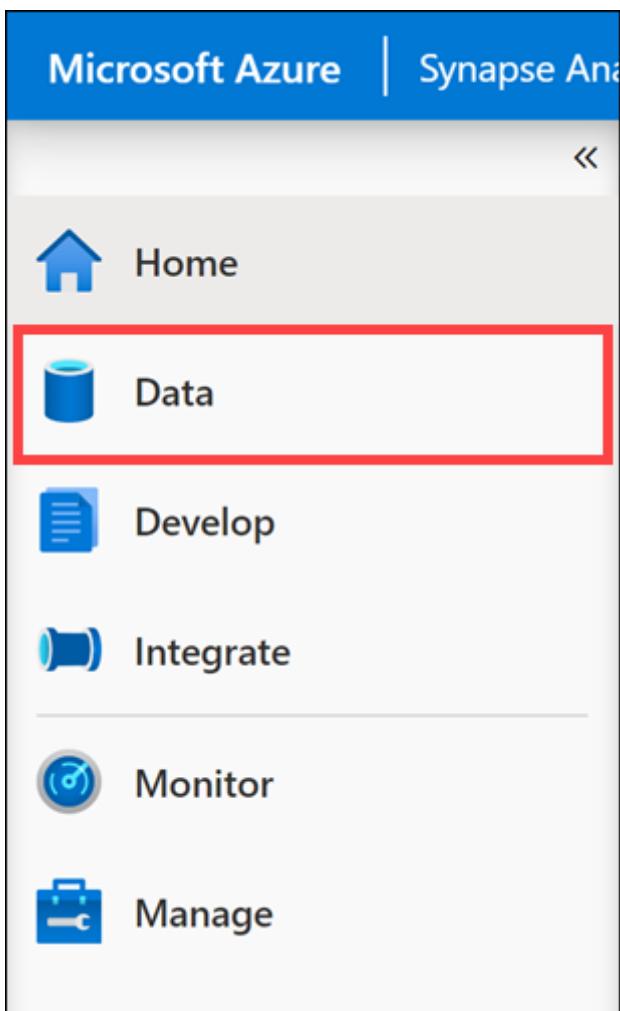
14.5 Exercise 3: Querying Azure Cosmos DB with Apache Spark for Synapse Analytics

Tailwind Traders wants to use Apache Spark to run analytical queries against the new Azure Cosmos DB container. In this segment, we will use built-in gestures in Synapse Studio to quickly create a Synapse Notebook that loads data from the analytical store of the HTAP-enabled container, without impacting the transactional store.

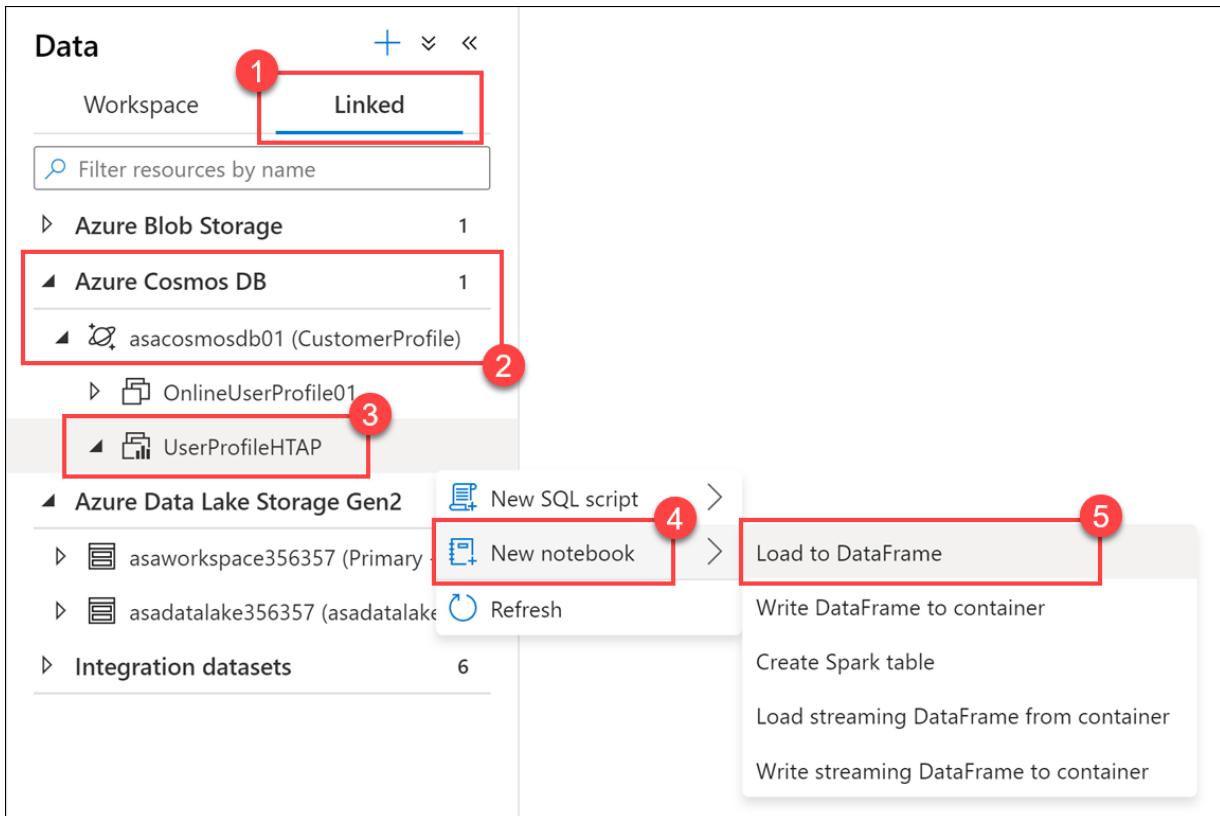
Tailwind Traders is trying to solve how they can use the list of preferred products identified with each user, coupled with any matching product IDs in their review history, to show a list of all preferred product reviews.

14.5.1 Task 1: Create a notebook

1. Navigate to the **Data** hub.



2. Select the **Linked** tab (1) and expand the **Azure Cosmos DB** section, then the **asacosmosdb01 (CustomerProfile)** linked service (2). Right-click on the **UserProfileHTAP** container (3), select the **New notebook** gesture (4), then select **Load to DataFrame** (5).



Notice that the **UserProfileHTAP** container that we created has a slightly different icon than the other container. This indicates that the analytical store is enabled.

3. In the new notebook, select your Spark pool in the **Attach to** dropdown list.



4. Select **Run all (1)**.

The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with a 'Run all' button (labeled 1) which is highlighted with a red box. Below the toolbar is a cell labeled 'Cell 1'. The code in the cell is:

```

1 # Read from Cosmos DB analytical store into a Spark DataFrame and display 10 rows from the DataFrame ...
2 # To select a preferred list of regions in a multi-region Cosmos DB account, add .option("spark.cosmos.preferredRegion"
3
4 df = spark.read\
5     .format("cosmos.olap")\ 2
6     .option("spark.synapse.linkedService", "asacosmosdb01")\ 3
7     .option("spark.cosmos.container", "UserProfileHTAP")\ 4
8     .load()
9
10 display(df.limit(10))

```

Four red arrows point to specific parts of the code: arrow 2 points to the 'format' method, arrow 3 points to the first 'option' method, and arrow 4 points to the second 'option' method. The code is followed by a message: 'Command executed in 3mins 33s 728ms by joel on 09-16-2020 00:22:16.665 -04:00'. Below the code cell, it says 'Job execution Succeeded' and 'Spark 2 executors 8 cores'. There are links to 'View in monitoring' and 'Open Spark UI'. The result section shows a table with the following data:

_rid	_ts	userId	cartId	preferredProducts
GpNQALvEY79PNQAAAAAAA==	1600214048	33304	af8aa699-c371-460f-ba4e-be8e4...	► "[2549,1841,2089,2874
GpNQALvEY79QNQAAAAAAA==	1600214048	18933	1c438ce1-d26c-4283-a8d7-7fe8...	► "[1935,1738,957,1992,2
GpNQALvEY79RNQAAAAAAA==	1600214048	35087	6a25d03b-f6d1-41a3-8e0d-2e3a...	► "[3704,1323,1614,438,3
GpNQALvEY79SNQAAAAAAA==	1600214048	22258	2c430f0f-769a-4d61-b67b-f5852...	► "[1999,4564,2629,4986
GpNQALvEY79TNQAAAAAAA==	1600214048	14753	a2c755bd-c8de-49e4-b5d8-33df...	► "[3433,4541,622,4900,2
GpNQALvEY79UNQAAAAAAA==	1600214048	4302	34e6d8a7-8985-4450-9f26-35ea...	"[]"

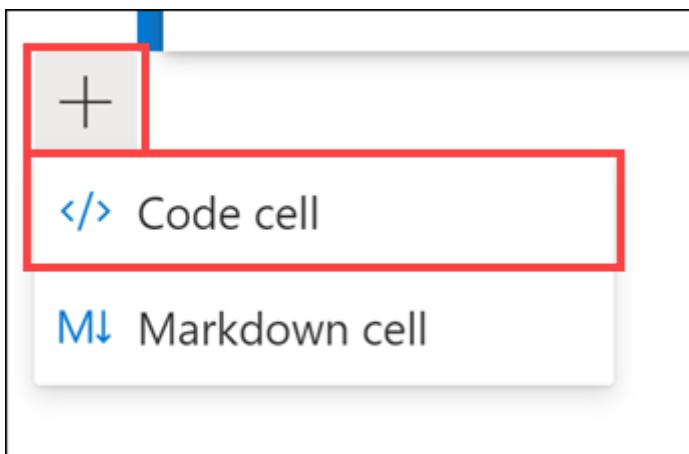
It will take a few minutes to start the Spark session the first time.

In the generated code within Cell 1, notice that the `spark.read` format is set to `cosmos.olap` (2). This instructs Synapse Link to use the container's analytical store. If we wanted to connect to the transactional store instead, like to read from the change feed or write to the container, we'd use `cosmos.oltp` instead.

Note: You cannot write to the analytical store, only read from it. If you want to load data into the container, you need to connect to the transactional store.

The first option configures the name of the Azure Cosmos DB linked service (3). The second option defines the Azure Cosmos DB container from which we want to read (4).

5. Select the + button underneath the cell you executed, then select </> Code cell. This adds a new code cell beneath the first one.



6. The DataFrame contains extra columns that we don't need. Let's remove the unwanted columns and create a clean version of the DataFrame. To do this, enter the following in the new cell and run it:

```
unwanted_cols = {'_attachments', '_etag', '_rid', '_self', '_ts', 'collectionType', 'id'}
```

```
# Remove unwanted columns from the columns collection
cols = list(set(df.columns) - unwanted_cols)
```

```
profiles = df.select(cols)

display(profiles.limit(10))
```

The output now only contains the columns that we want. Notice that the `preferredProducts` (1) and `productReviews` (2) columns contain child elements. Expand the values on a row to view them. You may recall seeing the raw JSON format in the `UserProfiles01` container within the Azure Cosmos DB Data Explorer.

Cell 2

```
1 unwanted_cols = {'_attachments','_etag','_rid','_self','_ts','collectionType','id'}
2
3 # Remove unwanted columns from the columns collection
4 cols = list(set(df.columns) - unwanted_cols)
5
6 profiles = df.select(cols)
7
8 display(profiles.limit(10))
```

Command executed in 4s 513ms by joel on 09-16-2020 00:36:56.940 -04:00

> Job execution Succeeded Spark 2 executors 8 cores [View in monitoring](#)

View [Table](#) [Chart](#)

preferredProducts

- ▼ "[2549,1841,2089,2874,185,2842,4!]
- 0: "2549"
- 1: "1841"
- 2: "2089"
- 3: "2874"
- 4: "185"
- 5: "2842"
- 6: "4577"
- 7: "1448"
- 8: "1294"

1

userId

- 33304

2

productReviews

- ▼ "[{"productId":2901,"reviewText":"c
- 0: {"productId":2901,"reviewTex
- 1: {"productId":738,"reviewTex
- 2: {"productId":1986,"reviewTex
- 3: {"productId":1163,"reviewTex

2

cartId

- af8aa699-c371-460f-ba4e-be8e4...

7. We should know how many records we're dealing with. To do this, enter the following in a new cell and **run it**:

```
profiles.count()
```

You should see a count result of 100 000

8. We want to use the `preferredProducts` column array and `productReviews` column array for each user and create a graph of products that are from their preferred list that match with products that they have reviewed. To do this, we need to create two new DataFrames that contain flattened values from those two columns so we can join them in a later step. Enter the following in a new cell and **run it**:

```
from pyspark.sql.functions import udf, explode
```

```
preferredProductsFlat=profiles.select('userId', explode('preferredProducts')).alias('productId'))  
productReviewsFlat=profiles.select('userId', explode('productReviews')).alias('productReviews'))  
display(productReviewsFlat.limit(10))
```

In this cell, we imported the special PySpark [explode function](#), which returns a new row for each element of the array. This function helps flatten the `preferredProducts` and `productReviews` columns for better readability or for easier querying.

Cell 4

```

1 from pyspark.sql.functions import udf, explode
2
3 preferredProductsFlat=profiles.select('userId',explode('preferredProducts').alias('productId'))
4 productReviewsFlat=profiles.select('userId',explode('productReviews').alias('productReviews'))
5 display(productReviewsFlat.limit(10))

```

Command executed in 2s 677ms by joel on 09-16-2020 01:13:23.535 -04:00

> Job execution Succeeded Spark 2 executors 8 cores [View](#)

View [Table](#) [Chart](#)

userId	productReviews
52563	<ul style="list-style-type: none"> ▼ {"productId":2793,"reviewText":"he productId: "2793" reviewText: ""heard about this on reviewDate: ""2019-05-19T00:30:00Z"}
52563	<ul style="list-style-type: none"> ► {"productId":1486,"reviewText":"M productId: "1486" reviewText: "My wife loves this product. reviewDate: ""2019-05-19T00:30:00Z"}
52563	<ul style="list-style-type: none"> ► {"productId":4959,"reviewText":"he productId: "4959" reviewText: ""he loves it. reviewDate: ""2019-05-19T00:30:00Z"}

Observe the cell output where we display the `productReviewFlat` DataFrame contents. We see a new `productReviews` column that contains the `productId` we want to match to the preferred products list for the user, as well as the `reviewText` that we want to display or save.

- Let's look at the `preferredProductsFlat` DataFrame contents. To do this, enter the following in a new cell and **run** it:

```
display(preferredProductsFlat.limit(20))
```

Cell 5



1 display(preferredProductsFlat.limit(20))

Command executed in 2s 738ms by joel on 09-16-2020 01:30:00.037 -04:00

> **Job execution Succeeded** **Spark** 2 executors 8 cores



View

Table

Chart

userId	productId
52563	1544
52563	3766
52563	4032
52563	1012
52563	3990
52563	393
40088	4484
40088	4109

Since we used the `explode` function on the preferred products array, we have flattened the column values to `userId` and `productId` rows, ordered by user.

10. Now we need to further flatten the `productReviewFlat` DataFrame contents to extract the `productReviews.productId` and `productReviews.reviewText` fields and create new rows for each data combination. To do this, enter the following in a new cell and **run** it:

```
productReviews = (productReviewsFlat.select('userId', 'productReviews.productId', 'productReviews.reviewText')
    .orderBy('userId'))
```

```
display(productReviews.limit(10))
```

In the output, notice that we now have multiple rows for each `userId`.

Cell 6

```

1 productReviews = (productReviewsFlat.select('userId', 'productReviews.productId', 'productReviews.reviewText')
2     .orderBy('userId'))
3
4 display(productReviews.limit(10))

```

Command executed in 2s 676ms by joel on 09-16-2020 01:13:30.418 -04:00

> Job execution Succeeded Spark 2 executors 8 cores

[View in monitoring](#) [Open in notebook](#)

View [Table](#) [Chart](#)

userId	productId	reviewText
1	3397	It only works when I'm Juan de N...
1	3687	I saw one of these in Nauru and I...
1	2864	i use it this time when i'm in my ...
1	3286	I saw one of these in Saint Pierre ...
1	4953	My neighbor Georgine has one o...
2	374	one of my hobbies is cooking. an...
2	4450	My baboon loves to play with it.
2	4151	My neighbor Georgine has one o...
2	1808	It only works when I'm Samoa

11. The final step is to join the `preferredProductsFlat` and `productReviews` DataFrames on the `userId` and `productId` values to build our graph of preferred product reviews. To do this, enter the following in a new cell and **run** it:

```

preferredProductReviews = (preferredProductsFlat.join(productReviews,
    (preferredProductsFlat.userId == productReviews.userId) &
    (preferredProductsFlat.productId == productReviews.productId))
)
display(preferredProductReviews.limit(100))

```

Note: Feel free to click on the column headers in the Table view to sort the result set.

Cell 7

```

1 preferredProductReviews = (preferredProductsFlat.join(productReviews,
2   (preferredProductsFlat.userId == productReviews.userId) &
3   (preferredProductsFlat.productId == productReviews.productId))
4 )
5
6 display(preferredProductReviews.limit(100))

```

Command executed in 2s 707ms by joel on 09-16-2020 01:26:23.275 -04:00

> Job execution Succeeded Spark 2 executors 8 cores



View

Table

Chart

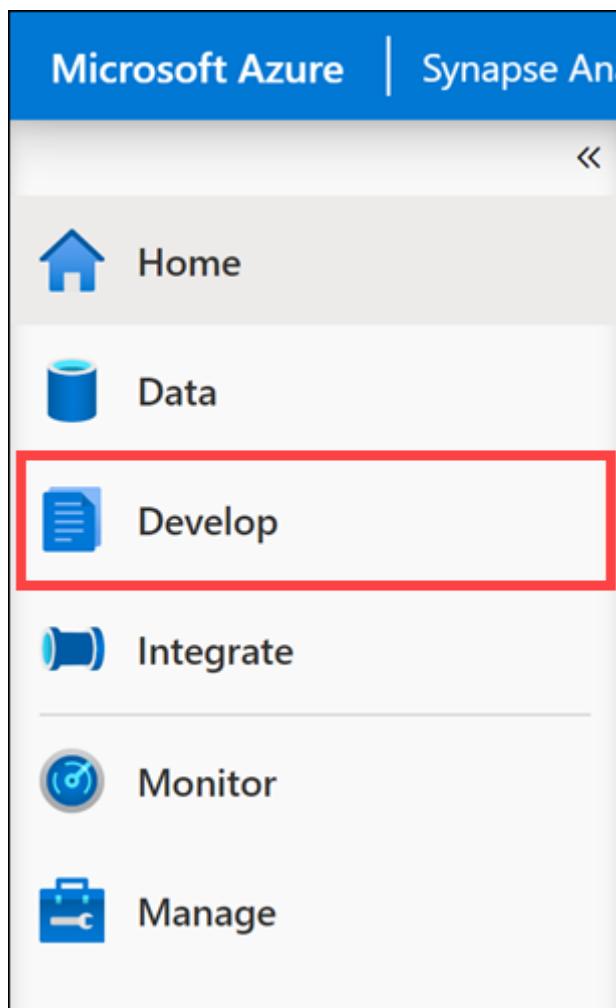
userId ↑	productId	reviewText
2382	1954	My raven loves to play with it.
2662	174	talk about contentment!!!
2863	4003	talk about contempt!!!
3496	2044	The box this comes in is 3 centim...
3616	3035	The box this comes in is 3 kilome...
3892	2297	My Shih-Tzu loves to play with it.
3971	2624	My goldfinch loves to play with it.
4684	2739	This deliverables works certainly ...
5014	319	this Graphic Interface is vertical

14.6 Exercise 4: Querying Azure Cosmos DB with serverless SQL pool for Azure Synapse Analytics

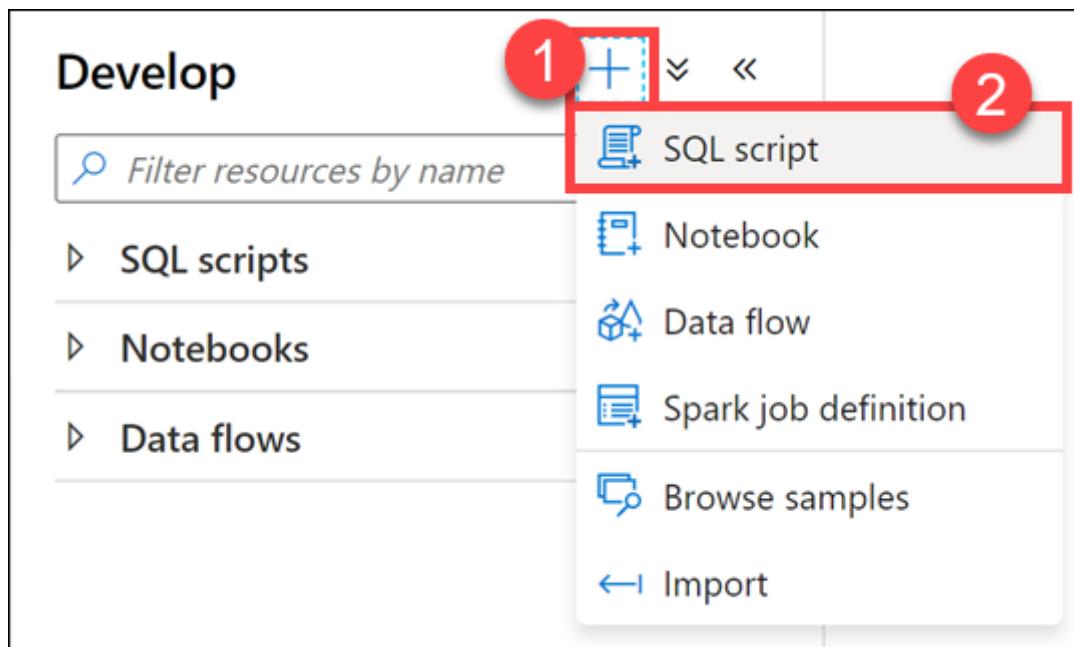
Tailwind Traders wants to explore the Azure Cosmos DB analytical store with T-SQL. Ideally, they can create views that can then be used for joins with other analytical store containers, files from the data lake, or accessed by external tools, like Power BI.

14.6.1 Task 1: Create a new SQL script

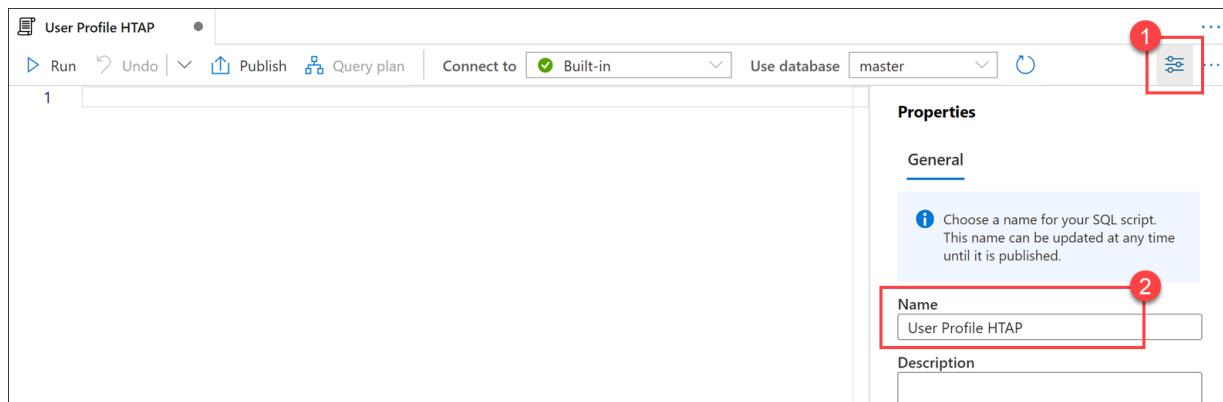
1. Navigate to the **Develop** hub.



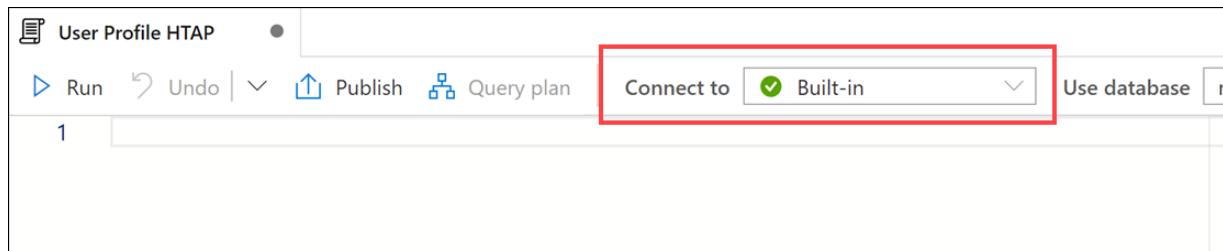
2. Select + (1), then SQL script (2).



3. When the script opens, you will see the Properties pane to the right (1). Enter **User Profile HTAP** for the **Name** (2), then select the **Properties** button to close the pane (1).



4. Verify that the serverless SQL pool (**Built-in**) is selected.



5. Paste the following SQL query. In the OPENROWSET statement, replace **YOUR_ACCOUNT_NAME** with the Azure Cosmos DB account name and **YOUR_ACCOUNT_KEY** with the Azure Cosmos DB Primary Key value you copied in step 5 above after you created the container.

```

USE master
GO

IF DB_ID (N'Profiles') IS NULL
BEGIN
    CREATE DATABASE Profiles;
END
GO

USE Profiles
GO

DROP VIEW IF EXISTS UserProfileHTAP;
GO

CREATE VIEW UserProfileHTAP
AS
SELECT
    *
FROM OPENROWSET(
    'CosmosDB',
    N'account=YOUR_ACCOUNT_NAME;database=CustomerProfile;key=YOUR_ACCOUNT_KEY',
    UserProfileHTAP
)
WITH (
    userId bigint,
    cartId varchar(50),
    preferredProducts varchar(max),
    productReviews varchar(max)
) AS profiles
CROSS APPLY OPENJSON (productReviews)
WITH (
    productId bigint,
    reviewText varchar(1000)
)

```

```
) AS reviews
```

```
GO
```

Your completed query should look similar to the following:

The screenshot shows a SQL query in a code editor with five numbered callouts (1-5) highlighting specific parts of the code:

- Callout 1: Points to the line `CREATE VIEW UserProfileHTAP`.
- Callout 2: Points to the line `FROM OPENROWSET(`, which includes the account details and container name.
- Callout 3: Points to the `WITH` clause and its associated column definitions.
- Callout 4: Points to the `CROSS APPLY OPENJSON (productReviews)` part of the query.
- Callout 5: Points to the message log at the bottom of the editor.

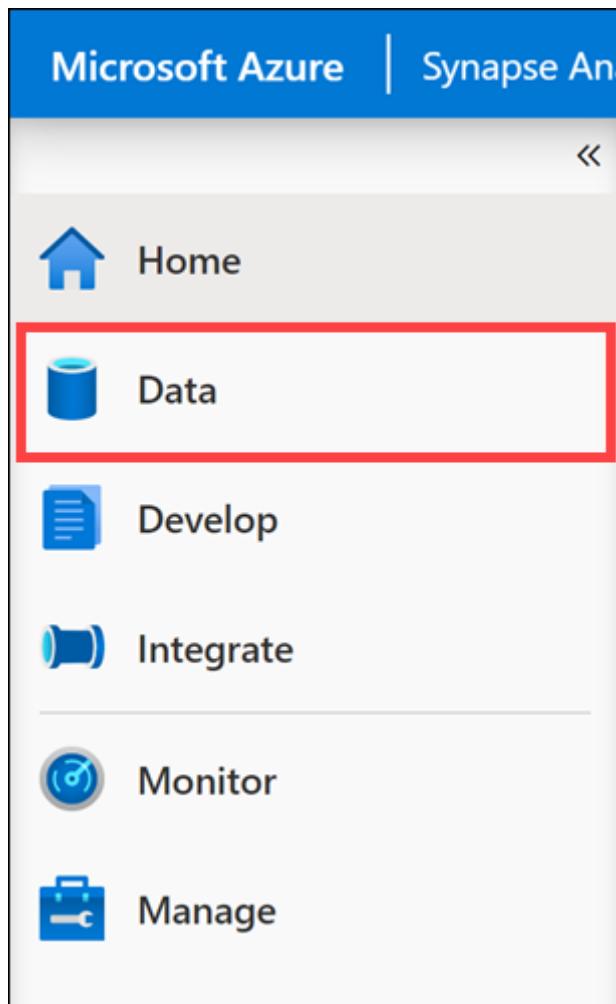
Below the code editor, there are two tabs: **Results** and **Messages**. The **Messages** tab is selected, showing the following output:

```
11:46:22 AM Started executing query at Line 1
(Changed database context to 'Profiles'. Changed database context to 'master'.)
Total execution time: 00:00:04.304
```

At the bottom of the message log, there is a green checkmark icon followed by the text: **00:00:04 Query executed successfully.**

The query starts out by creating a new serverless SQL pool database named **Profiles** if it does not exist, then executes `USE Profiles` to run the rest of the script contents against the **Profiles** database. Next, it drops the `UserProfileHTAP` view if it exists. Finally, it performs the following:

- 1. Creates a SQL view named `UserProfileHTAP`.
 - 2. Uses the `OPENROWSET` statement to set the data source type to `CosmosDB`, sets the account details, and specifies that we want to create the view over the Azure Cosmos DB analytical store container named `UserProfileHTAP`.
 - 3. The `WITH` clause matches the property names in the JSON documents and applies the appropriate SQL data types. Notice that we set the `preferredProducts` and `productReviews` fields to `varchar(max)`. This is because both of these properties contain JSON-formatted data within.
 - 4. Since the `productReviews` property in the JSON documents contain nested subarrays, we want to "join" the properties from the document with all elements of the array. Synapse SQL enables us to flatten the nested structure by applying the `OPENJSON` function on the nested array. We flatten the values within `productReviews` like we did using the Python `explode` function earlier in the Synapse Notebook.
 - 5. The output shows that the statements successfully executed.
6. Navigate to the **Data** hub.



7. Select the **Workspace** tab (1) and expand the Databases group. Expand the **Profiles** SQL on-demand database (2). If you do not see this on the list, refresh the Databases list. Expand Views, then right-click on the **UserProfileHTAP** view (3). Select **New SQL script** (4), then **Select TOP 100 rows** (5).

The screenshot shows the Azure Data Explorer interface. The left sidebar is titled "Data" and contains a "Workspace" section (marked 1) which is selected. Below it is a search bar ("Filter resources by name"). Under "Databases" (marked 4), there are two entries: "SQLPool01 (SQL)" and "demo (SQL)" (marked 2). "demo (SQL)" is expanded to show "Profiles (SQL)", "External tables", "External resources", "Views", "System views" (marked 3), "Schemas", "Security", and "default (Spark)". In the "Views" section, "dbo.UserProfileHTAP" is selected. The right pane shows a command bar with "New SQL script" (marked 4) and "Select TOP 100 rows" (marked 5).

8. Run the query and take note of the results.

The screenshot shows the results of the query execution. The query is:

```

1  SELECT TOP (100) [userId]
2  ,[cartId]
3  ,[preferredProducts]
4  ,[productReviews]
5  ,[productId]
6  ,[reviewText]
7  | FROM [dbo].[UserProfileHTAP]

```

The results are displayed in a table format. The columns are: CartId (1), PreferredProducts (2), ProductReviews, ProductId, and ReviewText. Red arrows point from the numbered callouts to the corresponding columns in the results table.

CartId	PreferredProducts	ProductReviews	ProductId	ReviewText
65080	[1462,1232,3641,3535,460]	[{"productId":2990,"reviewText":"I saw one of these in ..."}]	2990	I saw one of these in Haiti and I bou...
65080	[1462,1232,3641,3535,460]	[{"productId":2990,"reviewText":"I saw one of these in ..."}]	2006	This Wisconsin works quite well. It p...
65080	[1462,1232,3641,3535,460]	[{"productId":2990,"reviewText":"I saw one of these in ..."}]	2799	The box this comes in is 5 light-year...
65080	[1462,1232,3641,3535,460]	[{"productId":2990,"reviewText":"I saw one of these in ..."}]	4798	The box this comes in is 3 yard by 6...
61840	[2117,565,4794]	[{"productId":2749,"reviewText":"this New Zealand Do..."}]	2749	this New Zealand Dollar is ghetto.
61840	[2117,565,4794]	[{"productId":2749,"reviewText":"this New Zealand Do..."}]	3525	I saw one of these in Juan de Nova I...
61840	[2117,565,4794]	[{"productId":2749,"reviewText":"this New Zealand Do..."}]	1562	talk about pleasure!
61840	[2117,565,4794]	[{"productId":2749,"reviewText":"this New Zealand Do..."}]	1756	heard about this on ndombolo radi...

00:00:10 Query executed successfully.

The **preferredProducts** (1) and **productReviews** (2) fields are included in the query, which both contain JSON-formatted values. Notice how the CROSS APPLY OPENJSON statement in the view successfully flattened the nested subarray values in the **productReviews** (2) field by extracting the **productId** and **reviewText** values into new fields.

15 Module 13 - End-to-end security with Azure Synapse Analytics

In this module, students will learn how to secure a Synapse Analytics workspace and its supporting infrastructure. The student will observe the SQL Active Directory Admin, manage IP firewall rules, manage secrets with Azure Key Vault and access those secrets through a Key Vault linked service and pipeline activities. The student will understand how to implement column-level security, row-level security, and dynamic data masking when using dedicated SQL pools.

In this module, the student will be able to:

- Secure Azure Synapse Analytics supporting infrastructure
- Secure the Azure Synapse Analytics workspace and managed services
- Secure Azure Synapse Analytics workspace data

15.1 Lab details

- [Module 13 - End-to-end security with Azure Synapse Analytics](#)
 - [Lab details](#)
 - [Resource naming throughout this lab](#)
 - [Lab setup and pre-requisites](#)
 - [Exercise 0: Start the dedicated SQL pool](#)
 - [Exercise 1 - Securing Azure Synapse Analytics supporting infrastructure](#)
 - * [Task 1 - Observing the SQL Active Directory admin](#)
 - * [Task 2 - Manage IP firewall rules](#)
 - [Exercise 2 - Securing the Azure Synapse Analytics workspace and managed services](#)
 - * [Task 1 - Managing secrets with Azure Key Vault](#)
 - * [Task 2 - Use Azure Key Vault for secrets when creating Linked Services](#)
 - * [Task 3 - Secure workspace pipeline runs](#)
 - * [Task 4 - Secure Azure Synapse Analytics dedicated SQL pools](#)
 - [Exercise 3 - Securing Azure Synapse Analytics workspace data](#)
 - * [Task 1 - Column Level Security](#)
 - * [Task 2 - Row level security](#)
 - * [Task 3 - Dynamic data masking](#)
 - [Exercise 4: Cleanup](#)
 - * [Task 1: Pause the dedicated SQL pool](#)
 - [Reference](#)
 - [Other Resources](#)

This lab will guide you through several security-related steps that cover an end-to-end security story for Azure Synapse Analytics. Some key take-aways from this lab are:

1. Leverage Azure Key Vault to store sensitive connection information, such as access keys and passwords for linked services as well as in pipelines.
2. Introspect the data that is contained within the SQL Pools in the context of potential sensitive/confidential data disclosure. Identify the columns representing sensitive data, then secure them by adding column-level security. Determine at the table level what data should be hidden from specific groups of users then define security predicates to apply row level security (filters) on the table. If desired, you also have the option of applying Dynamic Data Masking to mask sensitive data returned in queries on a column by column basis.

15.2 Resource naming throughout this lab

For the remainder of this guide, the following terms will be used for various ASA-related resources (make sure you replace them with actual names and values):

Azure Synapse Analytics Resource	To be referred to
Workspace resource group	<code>WorkspaceResourceGroup</code>
Workspace / workspace name	<code>Workspace</code>
Primary Storage Account	<code>PrimaryStorage</code>
Default file system container	<code>DefaultFileSystem</code>
SQL Pool	<code>SqlPool01</code>
SQL Serverless Endpoint	<code>SqlServerless01</code>
Active Directory Principal of New User	<code>user@domain.com</code>

Azure Synapse Analytics Resource	To be referred to
SQL username of New User	<code>newuser</code>
Azure Key Vault	<code>KeyVault01</code>
Azure Key Vault Private Endpoint Name	<code>KeyVaultPrivateEndpointName</code>
Azure Subscription	<code>WorkspaceSubscription</code>
Azure Region	<code>WorkspaceRegion</code>

15.3 Lab setup and pre-requisites

Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Complete the **lab setup instructions** for this module.

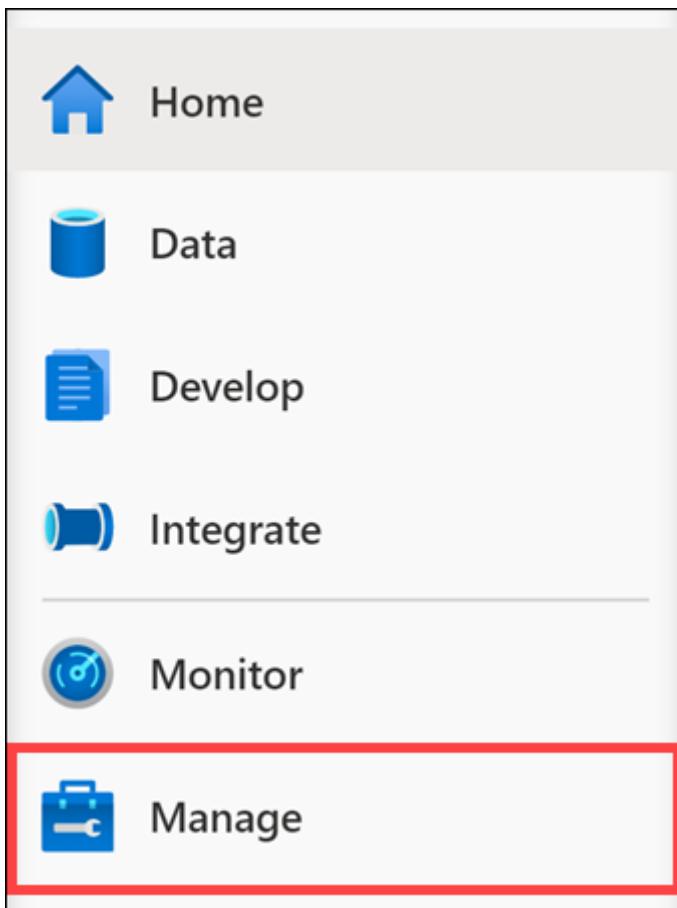
Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

15.4 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



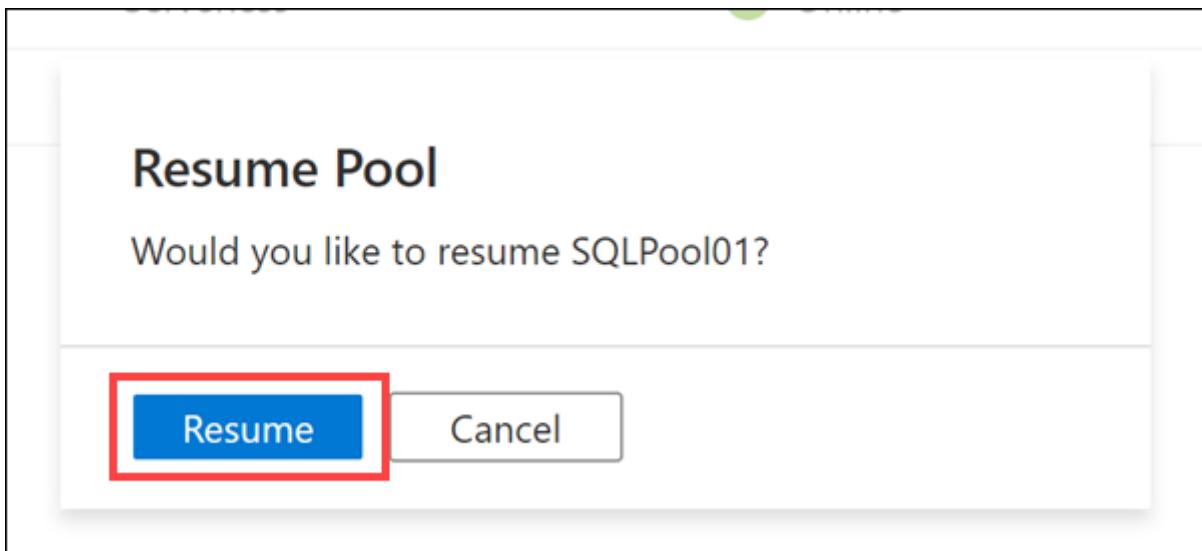
3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the 'SQL pools' blade in the Azure Data Studio interface. On the left, there is a sidebar with the following options: Analytics pools, SQL pools (highlighted with a red box and a red number '1'), Apache Spark pools, External connections, Linked services, Azure Purview (Preview), Integration, Triggers, Integration runtimes, Security, and Access control. The main area displays the 'SQL pools' section with the following details:

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused (highlighted with a red box)	DW100c

At the bottom of the table, there is a 'Resume' button, which is also highlighted with a red box and a red number '2'.

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

15.5 Exercise 1 - Securing Azure Synapse Analytics supporting infrastructure

Azure Synapse Analytics (ASA) is a powerful solution that handles security for many of the resources that it creates and manages. In order to run ASA, however, some foundational security measures need to be put in place to ensure the infrastructure that it relies upon is secure. In this exercise, we will walk through securing the supporting infrastructure of ASA.

15.5.1 Task 1 - Observing the SQL Active Directory admin

The SQL Active Directory Admin can be a user (the default) or group (best practice so that more than one user can be provided these permissions) security principal. The principal assigned to this will have administrative permissions to the SQL Pools contained in the workspace.

1. In the **Azure Portal** (<https://portal.azure.com>), browse to your lab resource group, and from the list of resources open your Synapse workspace (do not launch Synapse Studio).
2. From the left menu, select **SQL Active Directory admin** and observe who is listed as a SQL Active Directory Admin. Is it a user or group?

asaworkspace | SQL Active Directory admin

Search (Ctrl+ /)

Set admin Remove admin Save

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Settings

SQL Active Directory admin (highlighted with a red border)

Active Directory admin ⓘ

Azure Active Directory authentication allows you to centrally manage identity and access to the SQL pools.

Learn more ⓘ

✓ [redacted]

15.5.2 Task 2 - Manage IP firewall rules

Having robust Internet security is a must for every technology system. One way to mitigate internet threat vectors is by reducing the number of public IP addresses that can access the Azure Synapse Analytics Workspace through the use of IP firewall rules. The Azure Synapse Analytics workspace will then delegate those same rules to all managed public endpoints of the workspace, including those for SQL pools and SQL Serverless endpoints.

1. In the **Azure Portal**, open the Synapse workspace (do not launch Studio).
2. From the left menu of the **Azure Synapse Analytics** page, select **Firewalls**.

The screenshot shows the 'asaworkspace' Synapse workspace settings page. At the top, there's a search bar labeled 'Search (Ctrl+ /)'. Below it are three main sections: 'Properties', 'Locks', and 'Analytics pools' (which includes 'SQL pools' and 'Apache Spark pools'). A horizontal line separates these from the 'Security' section. Under 'Security', there are four options: 'Encryption' (with a shield icon), 'Firewalls' (with a blue square icon), 'Managed identities' (with a yellow lightning bolt icon), and 'Private endpoint connections' (with a blue double-headed arrow icon). The 'Firewalls' option is highlighted with a red rectangular box.

3. Notice that an IP Firewall rule of **Allow All** has already been created for you in the lab environment. If you wanted to add your specific IP address you would instead select **+ Add Client IP** from the toolbar menu (you do not need to do this now).

The screenshot shows the 'Firewalls' settings for the 'asaworkspace' Synapse workspace. The left sidebar includes options like Properties, Locks, Analytics pools (SQL pools, Apache Spark pools), Security (Encryption, Firewalls, Managed identities, Private endpoint connections), and a search bar. The 'Firewalls' tab is selected and highlighted with a red box. At the top right, there's a 'Save' and 'Discard' button, and a prominent red box surrounds the '+ Add client IP' button. Below it, a message states: 'The IPs listed below will have full access to Synapse workspace 'asaworkspace''. There's a toggle switch for 'Allow Azure services and resources to access this workspace' set to 'ON'. The main table lists a single rule: 'allowAll' with Start IP '0.0.0.0' and End IP '255.255.255.255'. An ellipsis button is also present.

Note: When connecting to Synapse from your local network, certain ports need to be open. To support the functions of Synapse Studio, ensure outgoing TCP ports 80, 443, and 1143, and UDP port 53 are open.

15.6 Exercise 2 - Securing the Azure Synapse Analytics workspace and managed services

15.6.1 Task 1 - Managing secrets with Azure Key Vault

When dealing with connectivity to external data sources and services, sensitive connection information such as passwords and access keys should be properly handled. It is recommended that this type of information be stored in an Azure Key Vault. Leveraging Azure Key Vault not only protects against secrets being compromised, it also serves as a central source of truth; meaning that if a secret value needs to be updated (such as when cycling access keys on a storage account), it can be changed in one place and all services consuming this key will start pulling the new value immediately. Azure Key Vault encrypts and decrypts information transparently using 256-bit AES encryption, which is FIPS 140-2 compliant.

1. In the **Azure Portal**, open the resource group for this lab, and from the list of resources, select the **Key vault** resource.

The screenshot shows a list of resources in the Azure portal. At the top, there are filters for 'Type == (all)', 'Location == (all)', and an 'Add filter' button. Below, it says 'Showing 1 to 9 of 9 records.' and has a 'Show hidden types' checkbox. The list includes several resources: 'amlworkspace212045' (Machine Learning), 'asaappinsights212045' (Application Insights), 'asacosmosdb212045' (Azure Cosmos DB account), 'asadatalake212045' (Storage account), 'asastore212045' (Storage account), and 'asakeyvault212045' (Key vault), which is highlighted with a red box. Each resource has a checkbox to its left and a 'Type' column to its right.

2. From the left menu, under Settings, select **Access Policies**.
3. Observe that Managed Service Identity (MSI) representing your Synapse workspace (it has a name similar to `asaworkspacennnnnnn`) has already been listed under Application and it has 4 selected Secret Permissions.

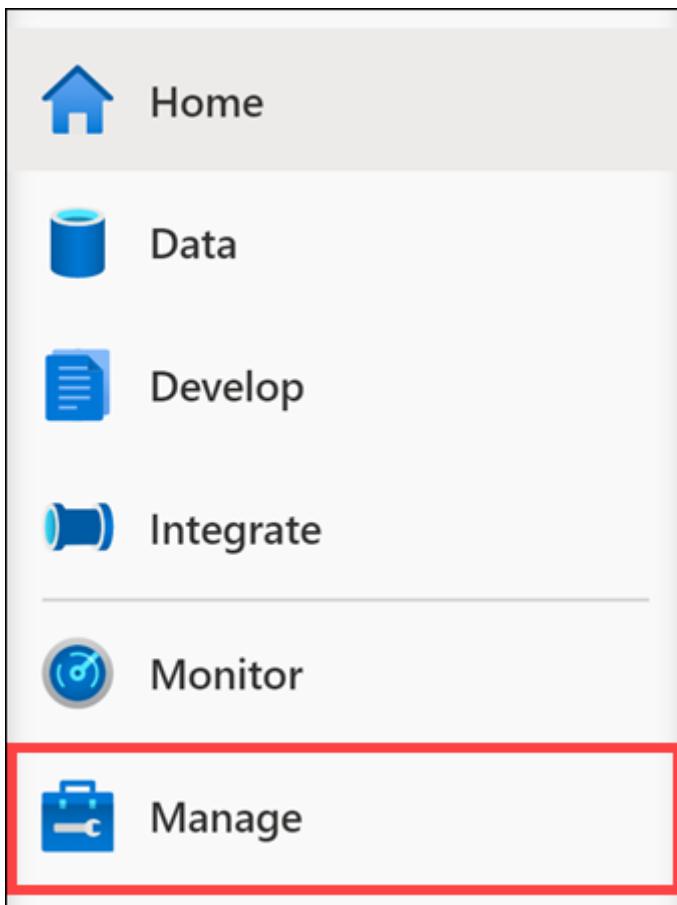
4. Select the drop-down that reads **4 selected** under **Secret Permissions**, observe that Get (which allows your workspace to retrieve the values of secrets from Key Vault) and List (which allows your workspace to enumerate secrets) are set.

15.6.2 Task 2 - Use Azure Key Vault for secrets when creating Linked Services

Linked Services are synonymous with connection strings in Azure Synapse Analytics. Azure Synapse Analytics linked services provides the ability to connect to nearly 100 different types of external services ranging from Azure Storage Accounts to Amazon S3 and more. When connecting to external services, having secrets related to connection information is almost guaranteed. The best place to store these secrets is the Azure Key Vault. Azure Synapse Analytics provides the ability to configure all linked service connections with values from Azure Key Vault.

In order to leverage Azure Key Vault in linked services, you must first add **asakeyvaultXX** as a linked service in Azure Synapse Analytics.

1. Navigate to **Azure Synapse Studio** (<https://web.azuresynapse.net/>) and sign in with the same user account you did in the Azure portal.
2. Select the **Manage** hub from the left menu.



- Beneath **External Connections**, select **Linked Services**, observe that a Linked Service pointing to your Key Vault has been provided in the environment.

The screenshot shows the 'Linked services' blade in the Azure Synapse Analytics portal. The left sidebar shows categories like 'Analytics pools', 'External connections' (with 'Linked services' selected), 'Integration', 'Security', and 'Access control'. The main area displays a table of linked services:

Name	Type	Related
asacosmosdb01	Azure Cosmos DB (SQL API)	2
asadatalake356357	Azure Data Lake Storage Gen2	3
asakeyvault356357	Azure Key Vault	5
asastore356357	Azure Blob Storage	0
asaworkspace356357-WorkspaceDefaultSq...	Azure Synapse Analytics	0

Since we have the Azure Key Vault set up as a linked service, we can leverage it when defining new linked services. Every New linked service provides the option to retrieve secrets from Azure Key Vault. The form requests the selection of the Azure Key Vault linked service, the secret name, and (optional) specific version of the secret.

New linked service (Oracle)

i Choose a name for your linked service. This name cannot be updated later.

Name *

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

(i)



Connection string

Azure Key Vault

AKV linked service *

(i)



Edit connection

Secret name *

(i)

Secret version

Use the latest version if left blank

(i)

Annotations

+ New

Parameters

Advanced (i)

15.6.3 Task 3 - Secure workspace pipeline runs

It is recommended to store any secrets that are part of your pipeline in Azure Key Vault. In this task you will retrieve these values using a Web activity, just to show the mechanics. The second part of this task demonstrates using a Web activity in the pipeline to retrieve a secret from the Key Vault.

1. Return to the Azure portal.
2. Open the `asakeyvaultXX` Azure Key Vault resource, and select **Secrets** from the left menu. From the top toolbar, select + **Generate/Import**.

The screenshot shows the 'Secrets' blade in the Azure Key Vault. On the left, there's a navigation menu with icons for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and Events (preview). Below that is a 'Settings' section with 'Keys', 'Secrets' (which is highlighted with a red box), and Certificates. At the top right, there are buttons for 'Generate/Import' (highlighted with a red box), 'Refresh', and 'Restore Backup'. A search bar is at the top left.

3. Create a secret, with the name **PipelineSecret** and assign it a value of **IsNotASecret**, and select the **Create** button.

Create a secret

The screenshot shows the 'Create a secret' form. It has fields for 'Name' (PipelineSecret) and 'Value' (.....). There are also sections for 'Content type (optional)', 'Set activation date?', and 'Set expiration date?'. At the bottom, there's a toggle for 'Enabled?' with options 'Yes' (selected) and 'No'.

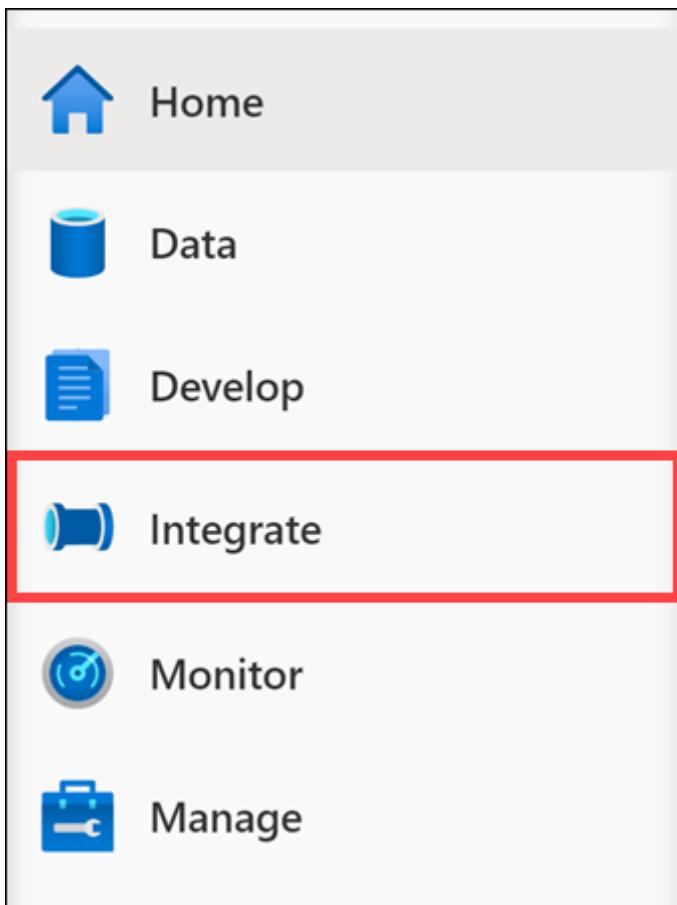
Upload options	Manual
Name *	PipelineSecret
Value *
Content type (optional)	
Set activation date?	<input type="checkbox"/>
Set expiration date?	<input type="checkbox"/>
Enabled?	<input checked="" type="radio"/> Yes <input type="radio"/> No

4. Open the secret that you just created, drill into the current version, and copy the value in the Secret

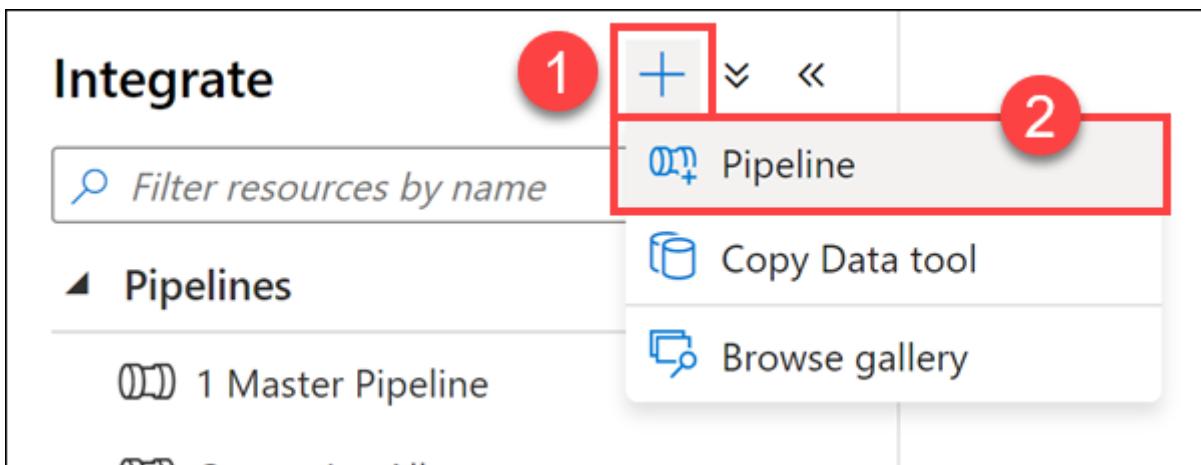
Identifier field. Save this value in a text editor, or retain it in your clipboard for a future step.

The screenshot shows the 'Properties' section of an Azure Key Vault secret. At the top, there are 'Save' and 'Discard' buttons. Below that, the 'Properties' section includes 'Created' (12/3/2020, 9:01:16 PM) and 'Updated' (12/3/2020, 9:01:16 PM). The 'Secret Identifier' field contains a URL: <https://asakeyvault... .vault.azure.net/secrets/PipelineSecret/56e20...>. To the right of this URL is a 'Copy to clipboard' button with a copy icon, which has a red box drawn around it. Below the URL, there's a 'Settings' section with options for activation and expiration dates, both with checkboxes. Under 'Enabled?', the 'Yes' button is selected. At the bottom, there's a 'Tags' section showing '0 tags'.

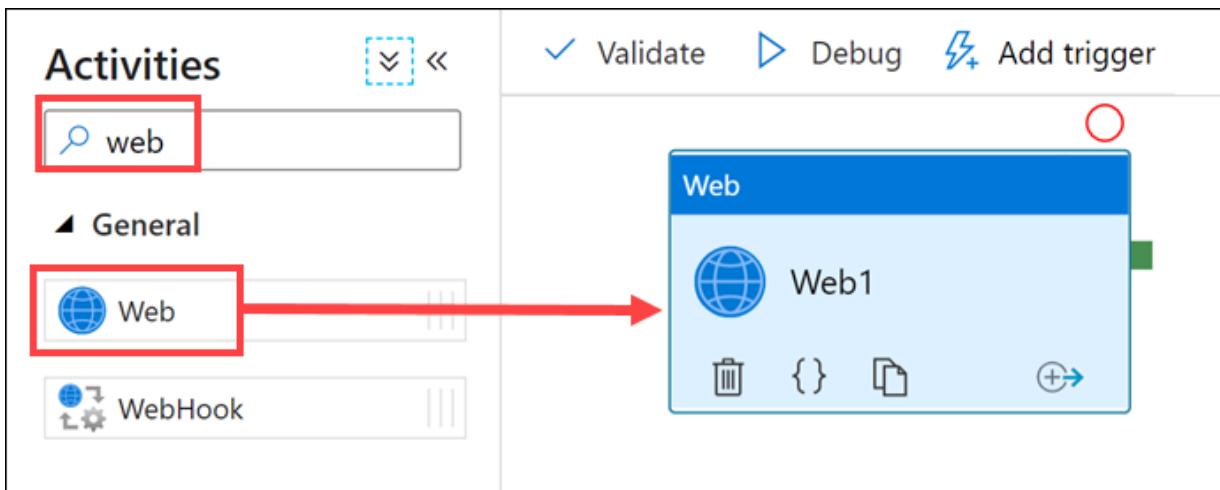
5. Switch back to Synapse Studio, then select the **Integrate** hub from the left menu.



6. From the **Integrate** blade, select the + button and add a new **Pipeline**.

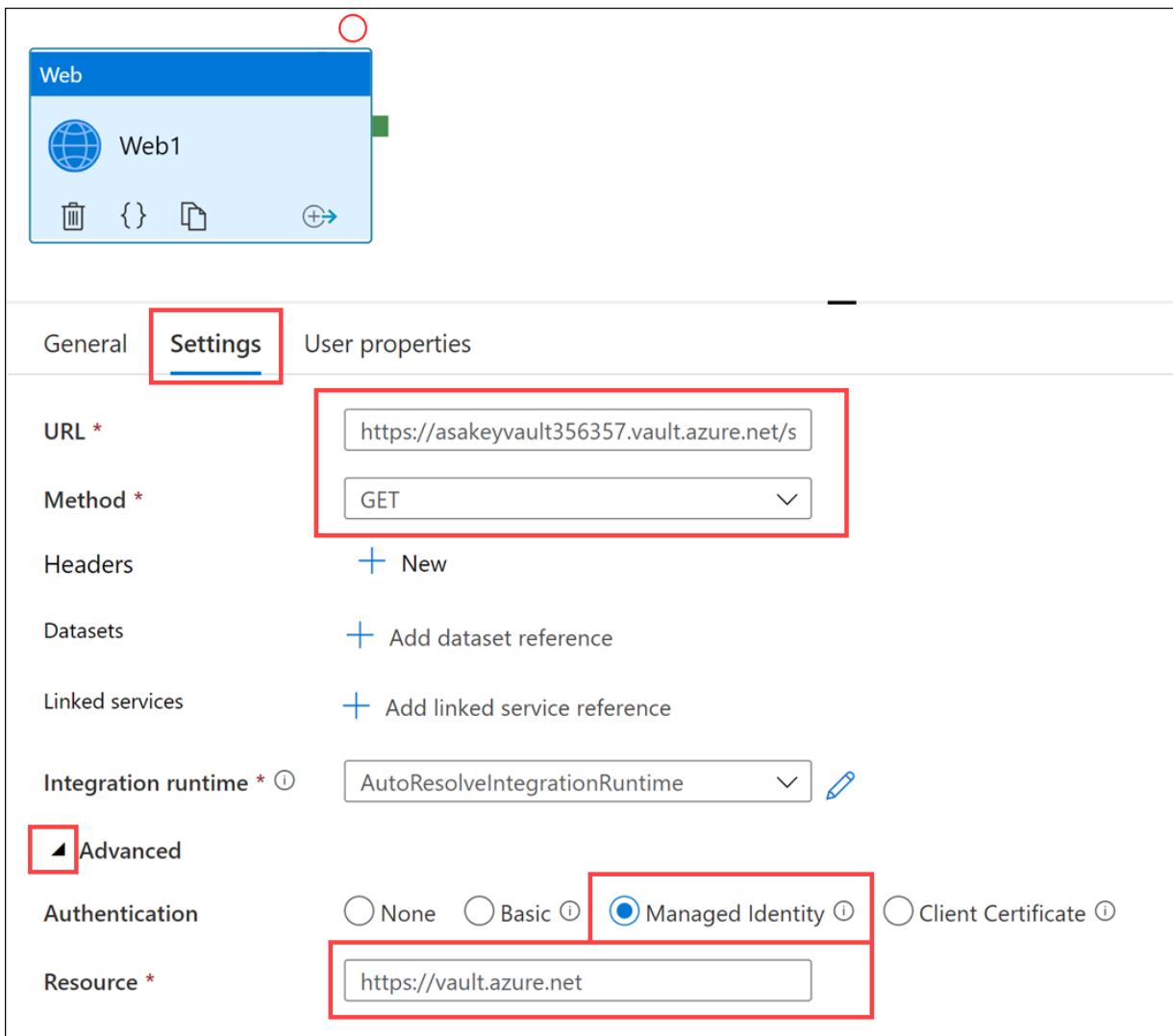


7. On the **Pipeline** tab, in the **Activities** pane search for **Web** and then drag an instance of a **Web** activity to the design area.

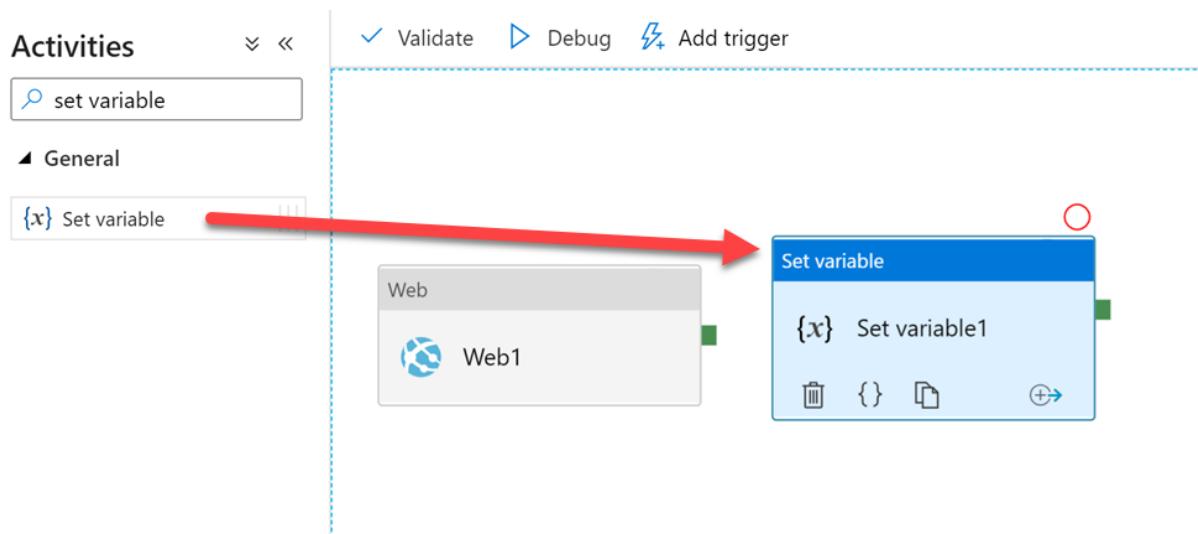


8. Select the **Web1** web activity, and select the **Settings** tab. Fill out the form as follows:

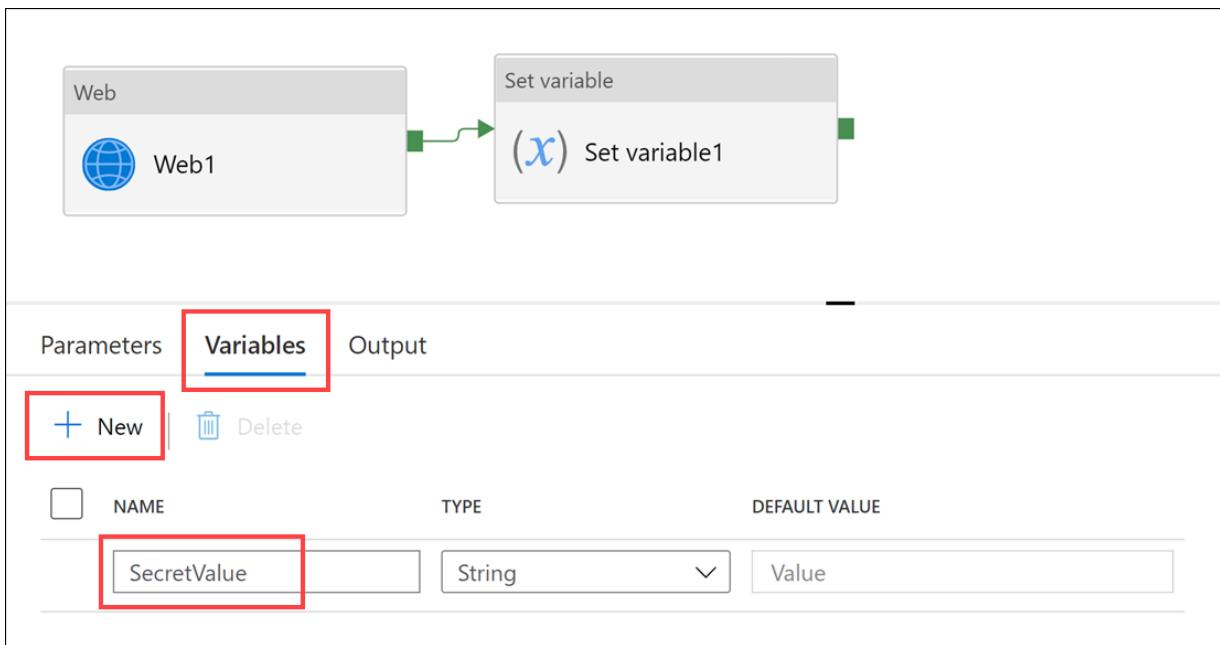
1. **URL:** Paste the Key Vault Secret Identifier value you copied in step 4 above, then **append ?api-version=7.1** to the end of this value. For example, it should look something like: <https://asakeyvaultNNNNN.vault.azure.net/secrets/PipelineSecret/f808d4fa99d84861872010f6c8d25c>
2. **Method:** Select **Get**.
3. Expand the **Advanced** section, and for **Authentication** select **MSI**. We have already established an Access Policy for the Managed Service Identity of our Synapse workspace, this means that the pipeline activity has permissions to access the key vault via an HTTP call.
4. **Resource:** Enter <https://vault.azure.net>



9. From the Activities pane, add a **Set variable** activity to the design surface of the pipeline.



10. On the design surface of the pipeline, select the **Web1** activity and drag a **Success** activity pipeline connection (green box) to the **Set variable1** activity.
11. With the pipeline selected in the designer (e.g., neither of the activities are selected), select the **Variables** tab and add a new **String** parameter named **SecretValue**.



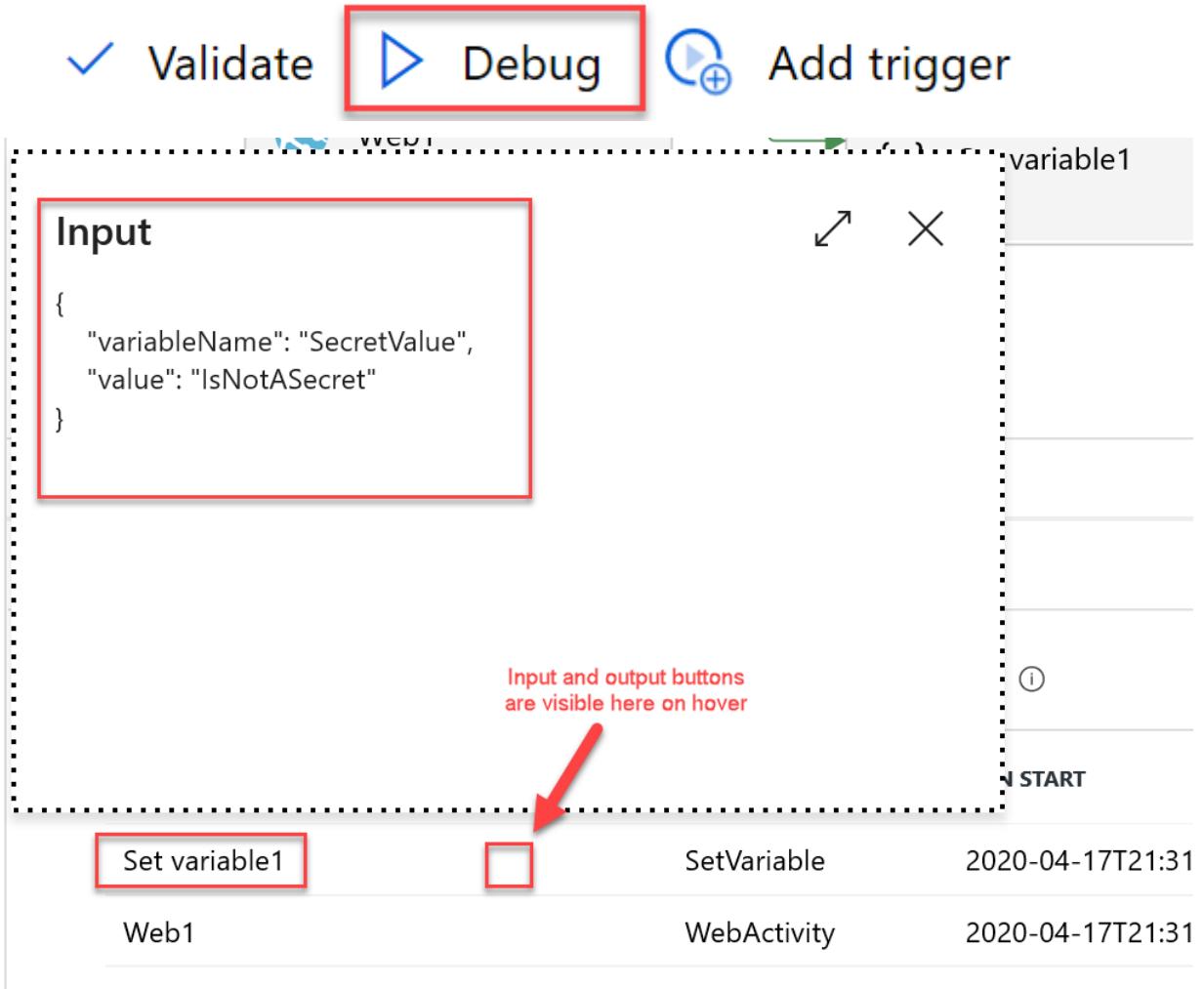
12. Select the **Set variable1** activity and select the **Variables** tab. Fill out the form as follows:

- Name:** Select **SecretValue** (the variable that we just created on our pipeline).
 - Value:** Enter `@activity('Web1').output.value`
- Validate
 Debug
 Add trigger

The screenshot shows the 'Set variable1' activity selected. A red circle highlights the 'Variables' tab in the properties panel. The 'Value' field contains the expression `@activity('Web1').output.value`.

Name *	SecretValue
Value *	<code>@activity('Web1').output.value</code>

13. Debug the pipeline by selecting **Debug** from the toolbar menu. When it runs observe the inputs and outputs of both activities from the **Output** tab of the pipeline.

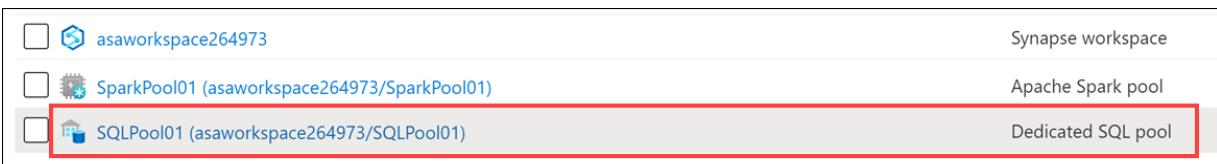


Note: On the **Web1** activity, on the **General** tab there is a **Secure Output** checkbox that when checked will prevent the secret value from being logged in plain text, for instance in the pipeline run, you would see a masked value ***** instead of the actual value retrieved from the Key vault. Any activity that consumes this value should also have their **Secure Input** checkbox checked.

15.6.4 Task 4 - Secure Azure Synapse Analytics dedicated SQL pools

Transparent Data Encryption (TDE) is a feature of SQL Server that provides encryption and decryption of data at rest, this includes: databases, log files, and back ups. When using this feature with Synapse Analytics dedicated SQL pools, it will use a built-in symmetric Database Encryption Key (DEK) that is provided by the pool itself. With TDE, all stored data is encrypted on disk, when the data is requested, TDE will decrypt this data at the page level as it's read into memory, and vice-versa encrypting in-memory data before it gets written back to disk. As with the name, this happens transparently without affecting any application code. When creating a dedicated SQL pool through Synapse Analytics, Transparent Data Encryption is not enabled. The first part of this task will show you how to enable this feature.

1. In the **Azure Portal**, open your resource group, then locate and open the **SqlPool01** dedicated SQL pool resource.



2. On the **SQL pool** resource screen, select **Transparent data encryption** from the left-hand menu. **DO NOT** turn on data encryption.

SQLPool01 ([/SQLPool01](#)) | Transparent data encryption

Dedicated SQL pool

Search (Ctrl+ /)

Save Discard Refresh Feedback

Properties Locks

Security

- Auditing
- Data Discovery & Classification
- Dynamic Data Masking
- Security Center
- Transparent data encryption**

Common Tasks

Encrypts your databases, backups, and logs at rest without any changes to your application. This setting applies only to this dedicated SQL pool.

Learn more [»](#)

Data encryption

ON OFF

Encryption status

Unencrypted

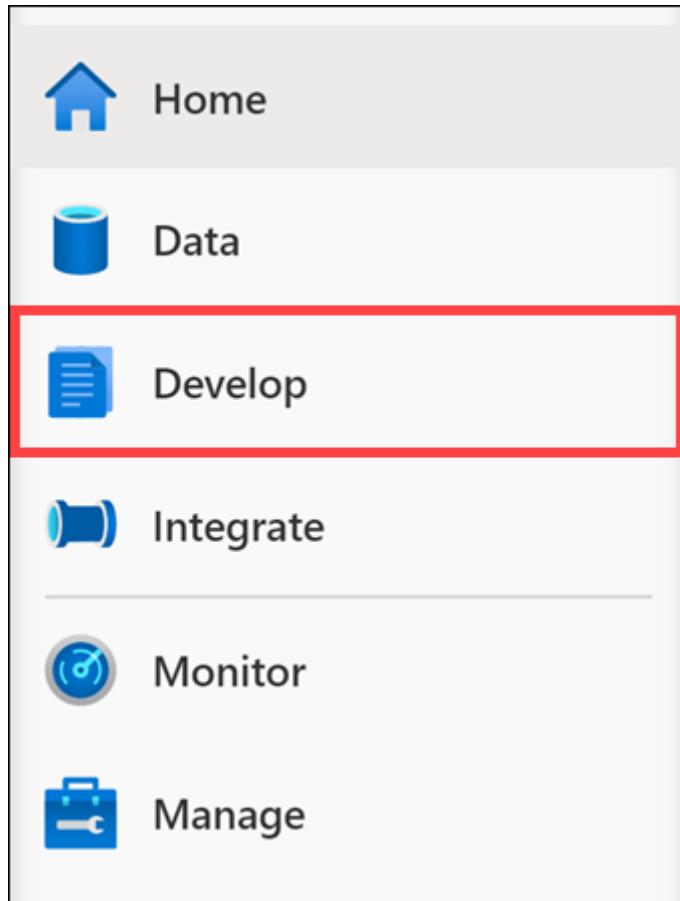
By default, this option is turned off. When you enable data encryption on this dedicated SQL pool, the pool is taken offline for a few minutes while TDE is applied.

15.7 Exercise 3 - Securing Azure Synapse Analytics workspace data

15.7.1 Task 1 - Column Level Security

It is important to identify data columns of that hold sensitive information. Types of sensitive could be social security numbers, email addresses, credit card numbers, financial totals, and more. Azure Synapse Analytics allows you define permissions that prevent users or roles select privileges on specific columns.

1. In **Azure Synapse Studio**, select **Develop** from the left menu.



2. From the **Develop** menu, expand the **SQL scripts** section, and select **Lab 05 - Exercise 3 - Column**

Level Security.

The screenshot shows the 'Develop' section of Azure Synapse Studio. At the top, there is a search bar with the placeholder 'Filter resources by name'. Below it, a heading 'SQL scripts' is followed by the number '10'. A list of scripts is displayed, with the first item, 'Lab 05 - Exercise 3 - Column Level Security', highlighted with a red box. The other items in the list are 'Lab 05 - Exercise 3 - Dynamic Data Masking' and 'Lab 05 - Exercise 3 - Row Level Security'. The background of the list area is blurred.

3. In the toolbar menu, connect to the database on which you want to execute the query, SQLPool01.



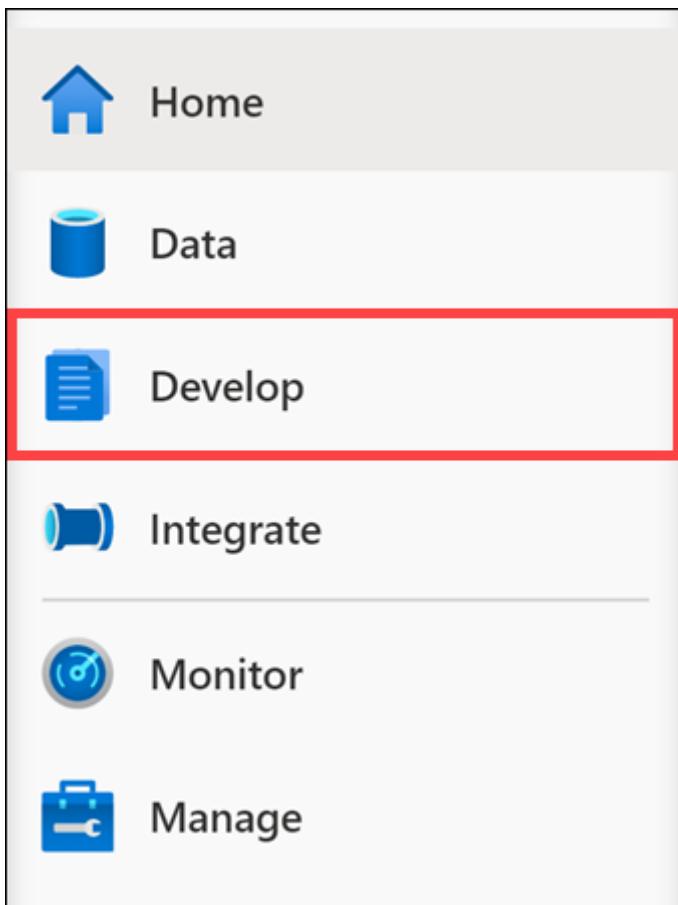
4. In the query window, **run each step individually** by highlighting the statement(s) in the step in the query window, and selecting the **Run** button from the toolbar (or enter F5).



5. You may now close the script tab, when prompted choose to **Discard all changes**.

15.7.2 Task 2 - Row level security

1. In **Azure Synapse Studio**, select **Develop** from the left menu.



2. From the **Develop** menu, expand the **SQL scripts** section, and select **Lab05 - Exercise 3 - Row Level Security**.

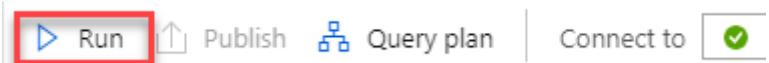
A screenshot of the 'Develop' menu's 'SQL scripts' section. The section title is 'SQL scripts' with a count of '10'. Below it is a list of three items:

- Lab 05 - Exercise 3 - Column Level Security
- Lab 05 - Exercise 3 - Dynamic Data Masking
- Lab 05 - Exercise 3 - Row Level Security (highlighted with a red box)

3. In the toolbar menu, connect to the database on which you want to execute the query, SQLPool01.



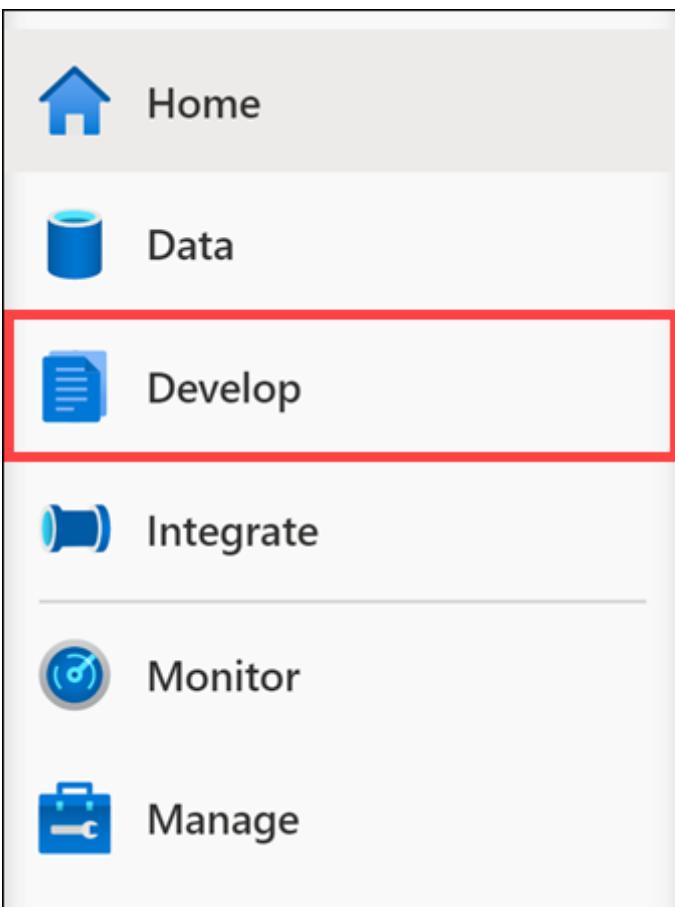
4. In the query window, **run each step individually** by highlighting the statement(s) for the step in the query window, and selecting the **Run** button from the toolbar (or enter F5).



5. You may now close the script tab, when prompted choose to **Discard all changes**.

15.7.3 Task 3 - Dynamic data masking

1. In **Azure Synapse Studio**, select **Develop** from the left menu.



2. From the **Develop** menu, expand the **SQL scripts** section, and select **Lab05 - Exercise 3 - Dynamic Data Masking**.

The screenshot shows the 'Develop' hub in Azure Synapse Studio. At the top, there's a search bar with the placeholder 'Filter resources by name'. Below it, a section titled 'SQL scripts' shows a list of 10 items. The third item in the list, 'Lab 05 - Exercise 3 - Dynamic Data Masking', is highlighted with a red rectangular box around its icon and text.

3. In the toolbar menu, connect to the database on which you want to execute the query, SQLPool01.



4. In the query window, **run each step individually** by highlighting the statement(s) for the step in the query window, and selecting the **Run** button from the toolbar (or enter F5).



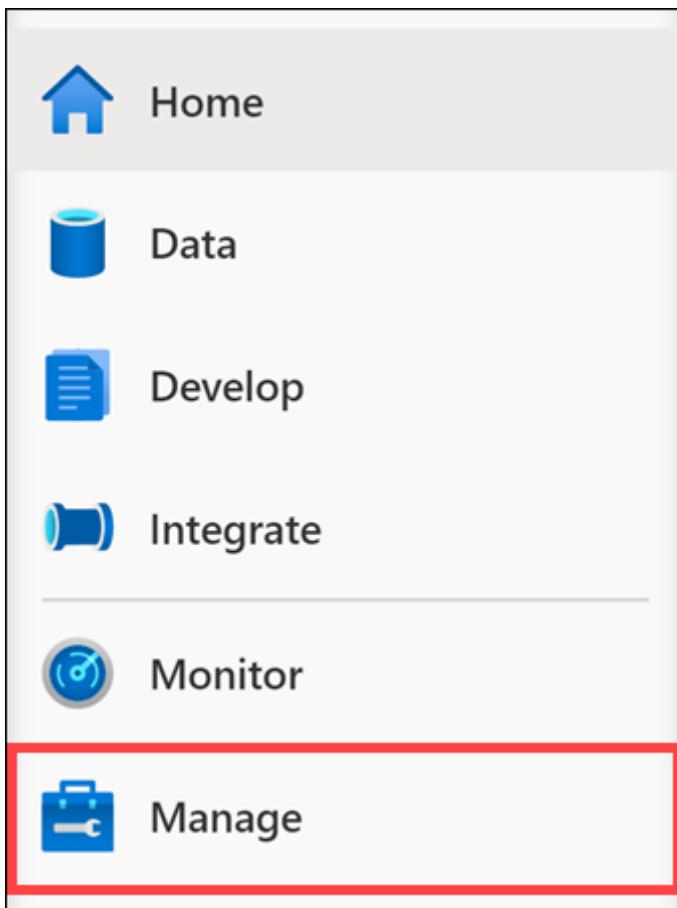
5. You may now close the script tab, when prompted choose to **Discard all changes**.

15.8 Exercise 4: Cleanup

Complete these steps to free up resources you no longer need.

15.8.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



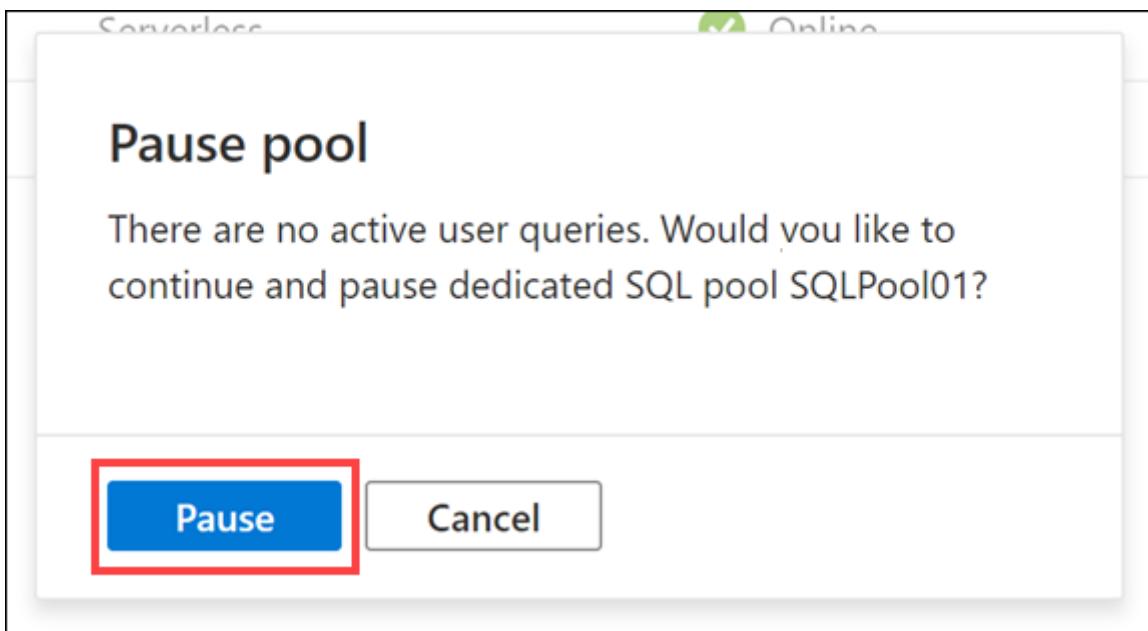
3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

The screenshot shows the 'SQL pools' blade in the Azure portal. On the left, there's a sidebar with sections like Analytics pools, External connections, Integration, and Security. The 'SQL pools' link is highlighted with a red box and a number '1'. The main area shows a table titled 'SQL pools' with the following data:

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

A red box highlights the 'Pause' button next to the 'SQLPool01' row, with a number '2' indicating the step to click it.

4. When prompted, select **Pause**.



15.9 Reference

- IP Firewalls
- Synapse Workspace Managed Identity
- Synapse Managed VNet
- Synapse Managed Private Endpoints
- Secure your Synapse Workspace
- Connect to your Synapse Workspace using private links
- Create a Managed private endpoint to your data source
- Granting Permissions to Workspace Managed Identity

15.10 Other Resources

- Managing access to workspaces, data and pipelines
- Analyze with Apache Spark
- Visualize data with Power BI
- Control storage account access for SQL on-demand

16 Module 14 - Real-time stream processing with Stream Analytics

In this module, students will learn how to process streaming data with Azure Stream Analytics. The student will ingest vehicle telemetry data into Event Hubs, then process that data in real time, using various windowing functions in Azure Stream Analytics. They will output the data to Azure Synapse Analytics. Finally, the student will learn how to scale the Stream Analytics job to increase throughput.

In this module, the student will be able to:

- Use Stream Analytics to process real-time data from Event Hubs
- Use Stream Analytics windowing functions to build aggregates and output to Synapse Analytics
- Scale the Azure Stream Analytics job to increase throughput through partitioning
- Repartition the stream input to optimize parallelization

16.1 Lab details

- [Module 14 - Real-time stream processing with Stream Analytics](#)
 - [Lab details](#)
 - [Technology overview](#)
 - * [Azure Stream Analytics](#)
 - * [Azure Event Hubs](#)
 - * [Power BI](#)
 - [Scenario overview](#)

- Lab setup and pre-requisites
- Exercise 0: Start the dedicated SQL pool
- Exercise 1: Configure services
 - * Task 1: Configure Event Hubs
 - * Task 2: Configure Synapse Analytics
 - * Task 3: Configure Stream Analytics
- Exercise 2: Generate and visualize data
 - * Task 1: Run data generator
 - * Task 2: Create Power BI dashboard
 - * Task 3: View aggregate data in Synapse Analytics
- Exercise 3: Cleanup
 - * Task 1: Stop the data generator
 - * Task 2: Stop the Stream Analytics job
 - * Task 3: Pause the dedicated SQL pool

16.2 Technology overview

16.2.1 Azure Stream Analytics

As more and more data is generated from a variety of connected devices and sensors, transforming this data into actionable insights and predictions in near real-time is now an operational necessity. [Azure Stream Analytics](#) seamlessly integrates with your real-time application architecture to enable powerful, real-time analytics on your data no matter what the volume.

Azure Stream Analytics enables you to develop massively parallel Complex Event Processing (CEP) pipelines with simplicity. It allows you to author powerful, real-time analytics solutions using very simple, declarative [SQL like language](#) with embedded support for temporal logic. Extensive array of [out-of-the-box connectors](#), advanced debugging and job monitoring capabilities help keep costs down by significantly lowering the developer skills required. Additionally, Azure Stream Analytics is highly extensible through support for custom code with [JavaScript User Defined functions](#) further extending the streaming logic written in SQL.

Getting started in seconds is easy with Azure Stream Analytics as there is no infrastructure to worry about, and no servers, virtual machines, or clusters to manage. You can instantly [scale-out the processing power](#) from one to hundreds of streaming units for any job. You only pay for the processing used per job.

[Guaranteed event delivery](#) and an enterprise grade SLA, provide the three 9's of availability, making sure that Azure Stream Analytics is suitable for mission critical workloads. Automated checkpoints enable fault tolerant operation with fast restarts with no data loss.

Azure Stream Analytics can be used to allow you to quickly build real-time dashboards with Power BI for a live command and control view. [Real-time dashboards](#) help transform live data into actionable and insightful visuals, and help you focus on what matters to you the most.

16.2.2 Azure Event Hubs

[Azure Event Hubs](#) is a big data pipeline that can ingest millions of events per second. It facilitates the capture, retention, and replay of telemetry and event stream data, using standard protocols such as HTTPS, AMQP, AMQP over websockets, and Kafka. The data can come from many concurrent sources and up to 20 consumer groups can allow applications to read entire event hub independently at their own pace.

16.2.3 Power BI

[Power BI](#) is a business analytics service that delivers insights to enable fast, informed decisions. Enabling you to transform data into stunning visuals and share them with colleagues on any device. Power BI provides a rich canvas on which to visually [explore and analyze your data](#). The ability to collaborate on and share customized [dashboards](#) and interactive reports is part of the experience, enabling you to scale across your organization with built-in governance and security.

16.3 Scenario overview

Contoso Auto is collecting vehicle telemetry and wants to use Event Hubs to rapidly ingest and store the data in its raw form, then do some processing in near real-time. In the end, they want to create a dashboard that automatically updates with new data as it flows in after being processed. What they would like to see on the dashboard are various visualizations of detected anomalies, like engines overheating, abnormal oil pressure, and

aggressive driving, using components such as a map to show anomalies related to cities, as well as various charts and graphs depicting this information in a clear way.

In this experience, you will use Azure Event Hubs to ingest streaming vehicle telemetry data as the entry point to a near real-time analytics pipeline built on Event Hubs, Azure Stream Analytics, and Power BI. Azure Stream Analytics extracts the vehicle sensor data from Event Hubs, performs aggregations over windows of time, then sends the aggregated data to Azure Synapse Analytics and Power BI for data visualization and analysis. A vehicle telemetry data generator will be used to send vehicle telemetry data to Event Hubs.

16.4 Lab setup and pre-requisites

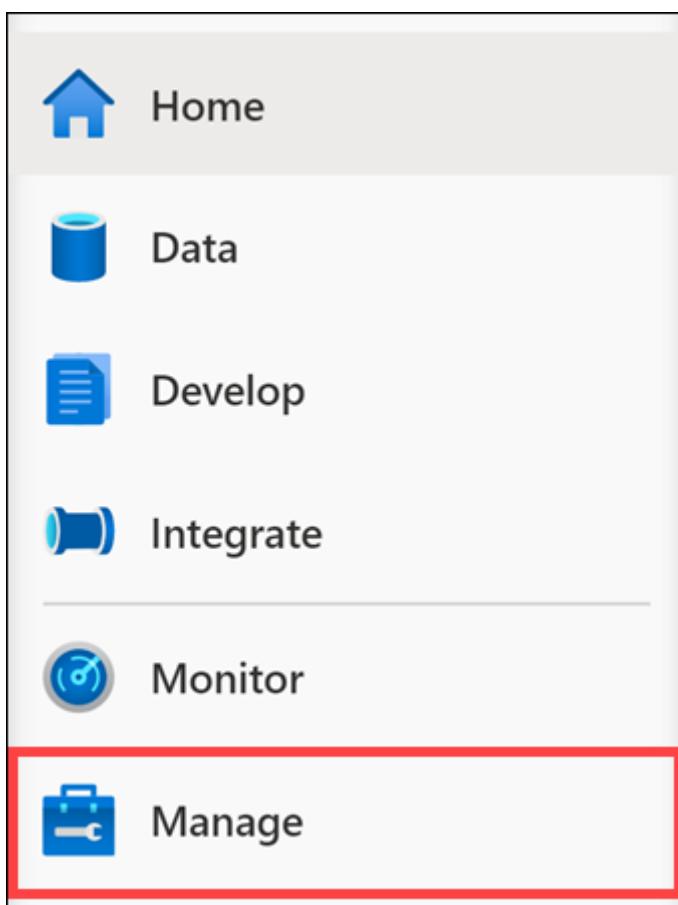
Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

- Azure subscription
- Power BI account (sign up at <https://powerbi.microsoft.com>)
- [Lab environment setup](#)

16.5 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

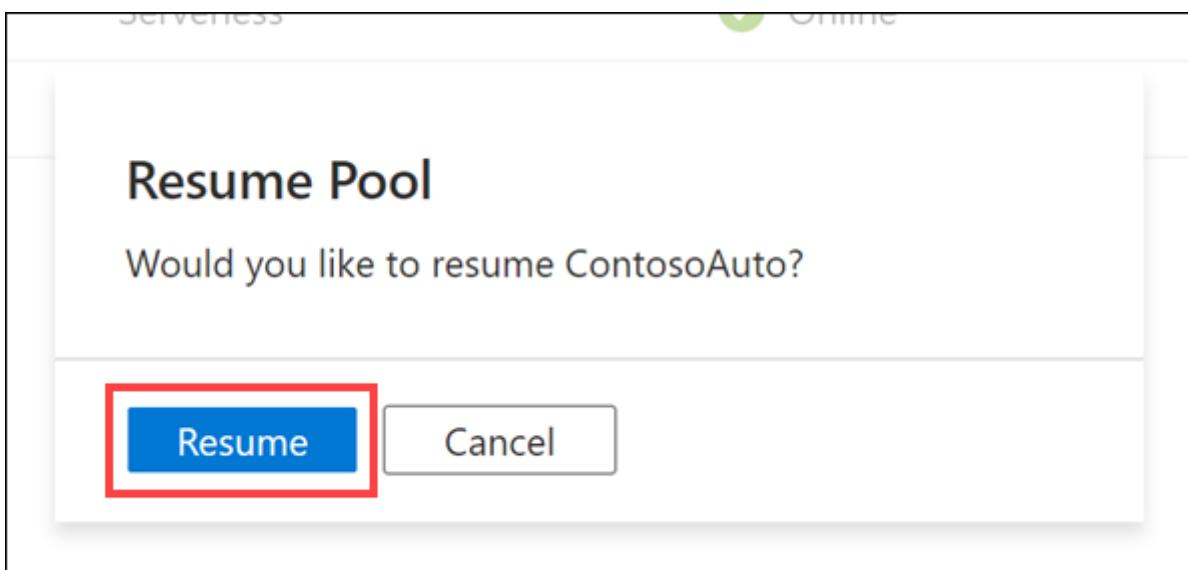
1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the 'SQL pools' section of the Azure portal. On the left, there's a sidebar with various options like 'Analytics pools', 'External connections', 'Integration', and 'Security'. The 'SQL pools' option is selected and highlighted with a red box and the number '1'. In the main area, there's a table titled 'SQL pools' with one item listed: 'Built-in' (Serverless, Online). Below it is another row for 'ContosoAuto' (Dedicated, Paused). A red box labeled '2' highlights the 'Resume' button in the 'ContosoAuto' row.

- When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

16.6 Exercise 1: Configure services

16.6.1 Task 1: Configure Event Hubs

Azure Event Hubs is a Big Data streaming platform and event ingestion service, capable of receiving and processing millions of events per second. We are using it to temporarily store vehicle telemetry data that is processed and ready to be sent to the real-time dashboard. As data flows into Event Hubs, Azure Stream Analytics will query the data, applying aggregates and tagging anomalies, then send it to Azure Synapse Analytics and Power BI.

In this task, you will create and configure a new event hub within the provided Event Hubs namespace. This will be used to capture vehicle telemetry after it has been processed and enriched by the Azure function you will create later on.

1. Navigate to the [Azure portal](#).
2. Select **Resource groups** from the left-hand menu. Then select the resource group named **ms-dataengineering-14**.
3. Select the **Event Hubs Namespace** (`eventhubYOUR_UNIQUE_ID`) from the list of resources in your resource group.

		Type ↑↓
<input type="checkbox"/>	Name ↑↓	
<input type="checkbox"/>	asadatalakede44	Storage account
<input type="checkbox"/>	asade44	Stream Analytics job
<input type="checkbox"/>	asaworkspacede44	Synapse workspace
<input type="checkbox"/>	ContosoAuto (asaworkspacede44/ContosoAuto)	Dedicated SQL pool
<input type="checkbox"/>	eventhubde44	Event Hubs Namespace

4. Within the Event Hubs Namespace blade, select **Event Hubs** within the left-hand menu.

The screenshot shows the Azure portal interface for an Event Hubs Namespace. On the left, there is a navigation menu with three items: 'Automation script', 'Entities', and 'Monitoring'. The 'Entities' item is expanded, revealing a list of resources. One resource, 'Event Hubs', is highlighted with a red box. Below the entities list is another section titled 'Monitoring' with a 'Alerts' option.

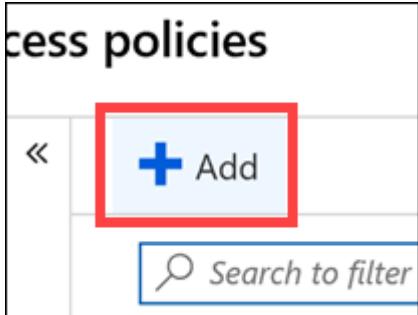
5. Select the **telemetry** event hub from the list.

The screenshot shows the 'Event Hub' blade. At the top, there are buttons for '+ Event Hub' and 'Refresh'. Below is a search bar with the placeholder 'Search to filter items...'. The main area is a table with two columns: 'NAME' and 'STATUS'. A single row is visible, containing the name 'telemetry' and the status 'Active'. The 'telemetry' cell is highlighted with a red box.

6. Select **Shared access policies** from the left-hand menu.

The screenshot shows the 'Settings' blade. It has four options: 'Shared access policies', 'Properties', and 'Locks'. The 'Shared access policies' option is highlighted with a red box.

7. Select **+ Add** in the top toolbar to create a new shared access policy.



8. In the **Add SAS Policy** blade, configure the following:

- **Name:** Enter "Read".
- **Managed:** Unchecked.
- **Send:** Unchecked.
- **Listen:** Checked.

A screenshot of the "Add SAS Policy" blade. The title is "Add SAS Policy" and it says "Event Hubs" below it. There is a close button "X" in the top right corner. The form has several fields:

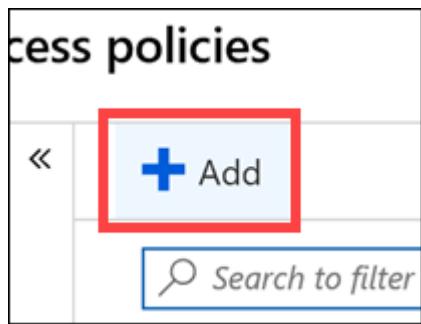
- A required field "Policy name" with the value "Read" entered. This field is highlighted with a red border.
- A checkbox for "Manage" which is unchecked.
- A checkbox for "Send" which is unchecked.
- A checkbox for "Listen" which is checked and highlighted with a red border.

At the bottom of the form is a blue "Create" button.

It is a best practice to create separate policies for reading, writing, and managing events. This follows the principle of least privilege to prevent services and applications from performing unauthorized operations.

9. Select **Create** on the bottom of the form when you are finished entering the values.

10. Select **+ Add** in the top toolbar to create a new shared access policy.



11. In the **Add SAS Policy** blade, configure the following:

- **Name:** Enter "Write".
- **Managed:** Unchecked.
- **Send:** Checked.
- **Listen:** Unchecked.

A screenshot of the "Add SAS Policy" blade. The title is "Add SAS Policy" and the subtitle is "Event Hubs".

- A field labeled "Policy name" with the value "Write" is highlighted with a red box. A green checkmark is visible to the right of the input field.
- A checkbox labeled "Manage" is unchecked.
- A checkbox labeled "Send" is checked and highlighted with a red box.
- A checkbox labeled "Listen" is unchecked.

At the bottom of the form is a blue "Create" button.

12. Select **Create** on the bottom of the form when you are finished entering the values.

13. Select your **Write** policy from the list. Copy the **Connection string - primary key** value by selecting the **Copy** button to the right of the field. **SAVE THIS VALUE** in Notepad or similar text editor for later.

16.6.2 Task 2: Configure Synapse Analytics

Azure Synapse is an end-to-end analytics platform which combines SQL data warehousing, big data analytics, and data integration into a single integrated environment. It empowers users to gain quick access and insights across all of their data, enabling a whole new level of performance and scale that is simply unmatched in the industry.

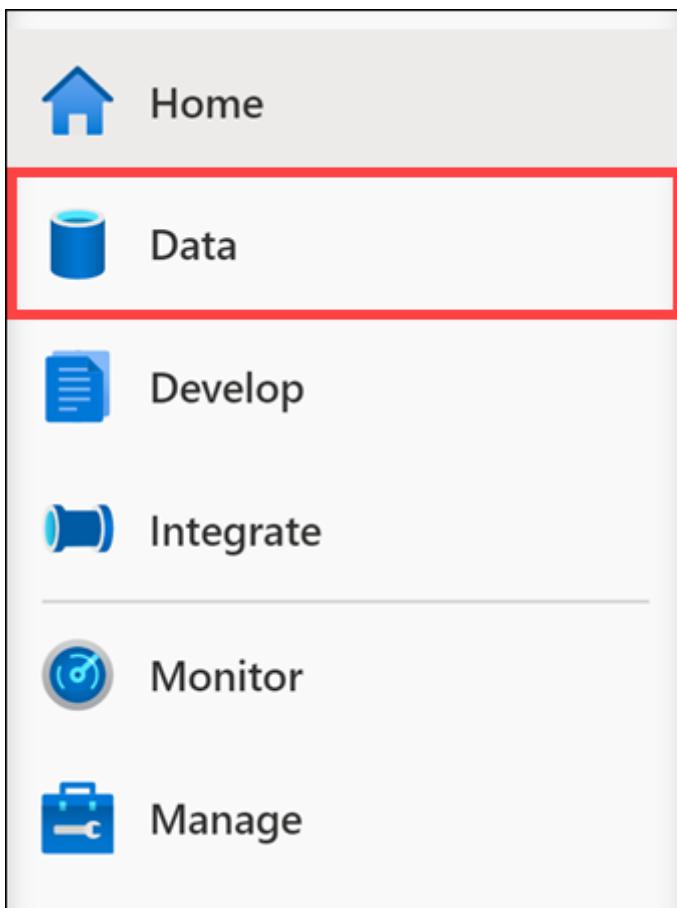
In this task, you will create a table in a Synapse dedicated SQL pool to store aggregate vehicle data provided by a Stream Analytics job that processes vehicle telemetry ingested by Event Hubs.

1. Navigate to the [Azure portal](#).
2. Select **Resource groups** from the left-hand menu. Then select the resource group named **ms-dataengineering-14**.
3. Select the **Synapse workspace** (`asaworkspaceYOUR_UNIQUE_ID`) from the list of resources in your resource group.

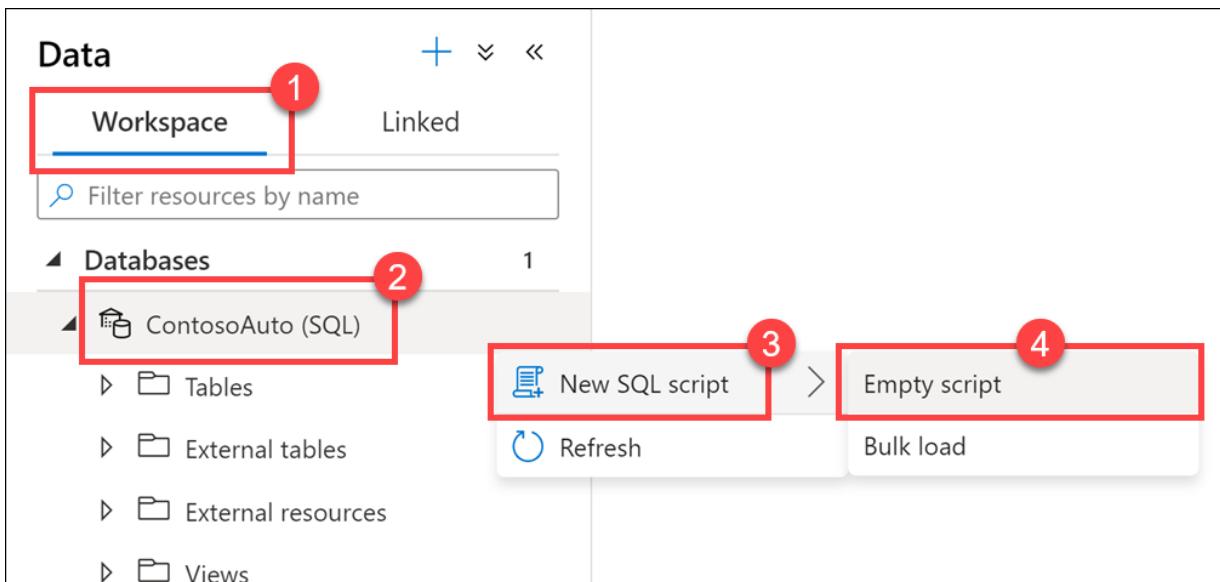
4. Select **Open** within the **Open Synapse Studio** box inside the Overview pane.

The screenshot shows the Azure Synapse Studio Overview page for workspace **asaworkspacede44**. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (SQL Active Directory admin, Properties, Locks), Analytics pools (SQL pools, Apache Spark pools), Security, and Encryption. The main content area displays workspace details such as Resource group (ms-dataengineering-14), Status (Succeeded), Location (East US), Subscription (change), Subscription ID, Managed virtual network (No), Managed Identity object (None), Workspace web URL (<https://web.azure.synapse.net?workspace=%2fschemas%2fcontosoauto>), and Tags (change). It also shows Firewalls settings, Primary ADLS Gen2 account endpoint, Primary ADLS Gen2 file system endpoint, SQL admin username (asa.sql.admin), SQL Active Directory ad..., Dedicated SQL endpoint (asaworkspacede44.sql.azuresynapse.net), Serverless SQL endpoint (asaworkspacede44-ondemand.sql.azuresynapse.net), and Development endpoint. A 'Getting started' section includes a 'Open Synapse Studio' button, which is highlighted with a red box.

5. Within Synapse Studio, select **Data** in the left-hand menu to navigate to the Data hub.



6. Select the **Workspace** tab (1), expand Databases and right-click **ContosoAuto** (2). Select **New SQL script** (3), then select **Empty script** (4).



7. Make sure the script is connected to ContosoAuto, then replace the script with the following and select Run to create a new table:

```

CREATE TABLE dbo.VehicleAverages
(
    [AverageEngineTemperature] [float] NOT NULL,
    [AverageSpeed] [float] NOT NULL,
    [Snapshot] [datetime] NOT NULL
)
WITH
(
    DISTRIBUTION = ROUND_ROBIN,
    CLUSTERED COLUMNSTORE INDEX
)
GO

```

```

SQL script 1
Run Undo Publish Query plan Connect to ContosoAuto ...
1 CREATE TABLE dbo.VehicleAverages
2 (
3     [AverageEngineTemperature] [float] NOT NULL,
4     [AverageSpeed] [float] NOT NULL,
5     [Snapshot] [datetime] NOT NULL
6 )
7 WITH
8 (
9     DISTRIBUTION = ROUND_ROBIN,
10    CLUSTERED COLUMNSTORE INDEX
11 )
12 GO

```

16.6.3 Task 3: Configure Stream Analytics

Azure Stream Analytics is an event-processing engine that allows you to examine high volumes of data streaming from devices. Incoming data can be from devices, sensors, web sites, social media feeds, applications, and more. It also supports extracting information from data streams, identifying patterns, and relationships. You can then use these patterns to trigger other actions downstream, such as create alerts, feed information to a reporting tool, or store it for later use.

In this task, you will configure Stream Analytics to use the event hub you created as a source, query and analyze

that data, then send it to Power BI for reporting and aggregated data to Azure Synapse Analytics.

1. Navigate to the [Azure portal](#).
2. Select **Resource groups** from the left-hand menu. Then select the resource group named **ms-dataengineering-14**.
3. Select the **Stream Analytics job** (**asaYOUR_UNIQUE_ID**) from the list of resources in your resource group.

Name ↑↓	Type ↑↓
asadalake44	Storage account
asade44	Stream Analytics job
asaworkspacede44	Synapse workspace
ContosoAuto (asaworkspacede44/ContosoAuto)	Dedicated SQL pool
eventhubde44	Event Hubs Namespace

4. Within the Stream Analytics job, select **Storage account settings** in the left-hand menu, then select **Add storage account**. Since we will use Synapse Analytics as one of the outputs, we need to first configure the job storage account.

The screenshot shows the 'Storage account settings' page for the Stream Analytics job 'asade44'. The left sidebar lists 'Inputs', 'Functions', 'Query', and 'Outputs'. The main area shows 'Storage account settings' with a red box around the 'Add storage account' button. The bottom section shows 'Configure' with 'Environment' selected, and 'Storage account settings' is also highlighted with a red box. Other options like 'Scale' are visible.

5. In the **Storage account settings** form, configure the following:

- **Select storage account from your subscriptions:** Selected.
- **Subscription:** Make sure the subscription you are using for this lab is selected.
- **Storage account:** Select the storage account named **asadatalakeYOUR_UNIQUE_ID**.
- **Authentication mode:** Select "Connection string".

 Save

i This storage account is used for storing content related to your job.

Storage account settings

Provide Blob storage/ADLS Gen2 settings manually

Select Blob storage/ADLS Gen2 from your subscriptions

Subscription

Storage account *

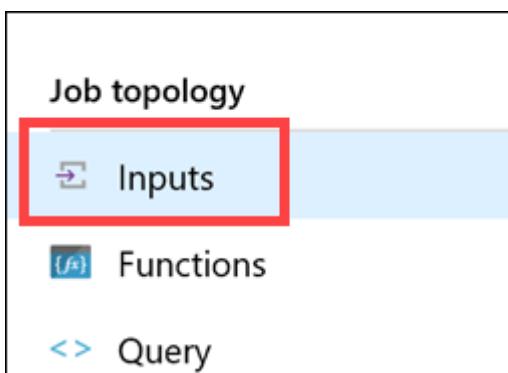
asadatalake356363

Authentication mode

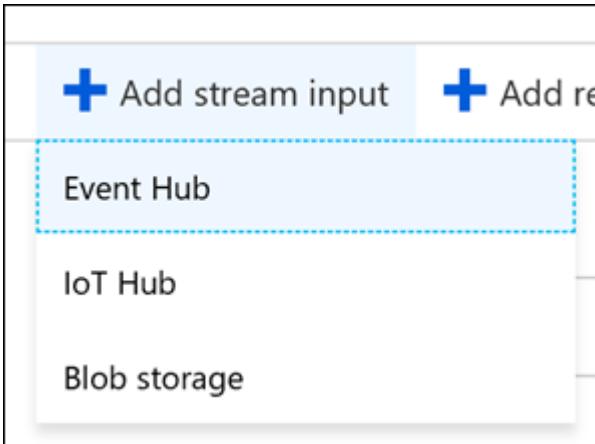
Connection string

6. Select **Save**, then **Yes** when prompted to save the storage account settings.

7. Within the Stream Analytics job, select **Inputs** within the left-hand menu.



8. Select **+ Add stream input** in the top toolbar, then select **Event Hub** to create a new Event Hub input.



9. In the **New Input** blade, configure the following:

- **Name:** Enter "eventhub".
- **Select Event Hub from your subscriptions:** Selected.
- **Subscription:** Make sure the subscription you are using for this lab is selected.
- **Event Hub namespace:** Select the Event Hub namespace you are using for this lab.
- **Event Hub name:** Select **Use existing**, then select **telemetry**, which you created earlier.
- **Event Hub consumer group:** Select **Use existing**, then select **\$Default**.
- **Authentication mode:** Select **Connection string**.
- **Event Hub policy name:** Select **Use existing**, then select **Read**.
- Leave all other values at their defaults.

Event Hub

X

New input

Input alias *

eventhub



Provide Event Hub settings manually

Select Event Hub from your subscriptions

Subscription

Event Hub namespace * ⓘ

eventhubde44



Event Hub name * ⓘ

Create new Use existing

telemetry



Event Hub consumer group * ⓘ

Create new Use existing

\$Default



Authentication mode

Connection string



Event Hub policy name * ⓘ

Create new Use existing

Read



Event Hub policy key

.....

Event serialization format * ⓘ

JSON



Encoding ⓘ

UTF-8

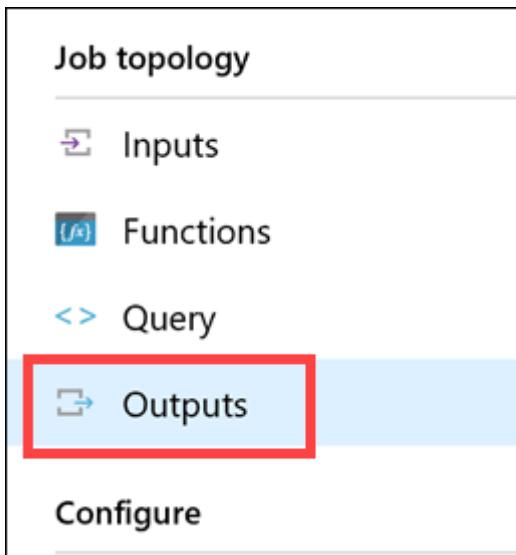


Event compression type ⓘ

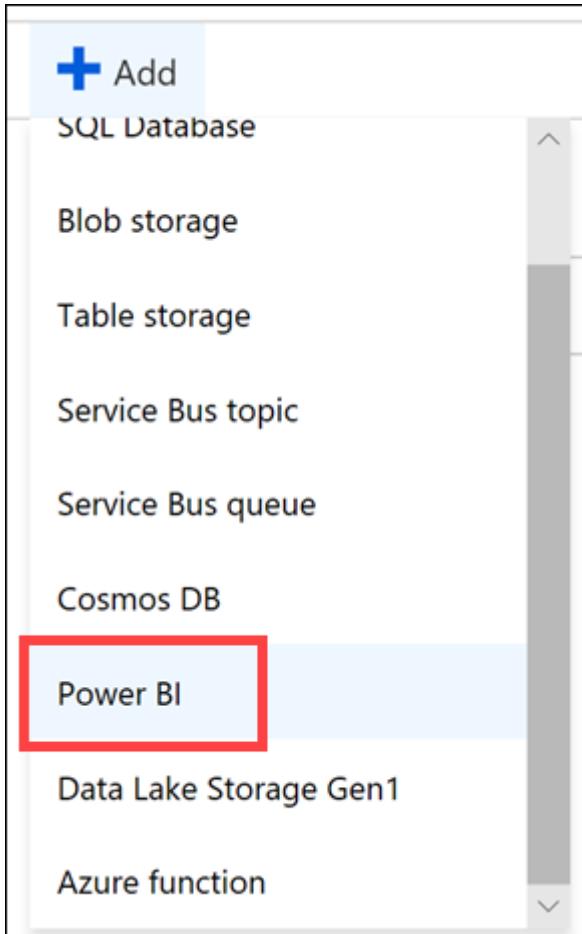
None



10. Select **Save** on the bottom of the form when you are finished entering the values.
11. Within the Stream Analytics job blade, select **Outputs** within the left-hand menu.



12. Select **+ Add** in the top toolbar, then select **Power BI** to create a new Power BI output.



13. In the **New Output** blade, select the **Authorize** button to authorize a connection from Stream Analytics to your Power BI account.

Power BI

X

New output

Authorize connection

You'll need to authorize with Power BI to configure your output settings.

[Authorize](#)

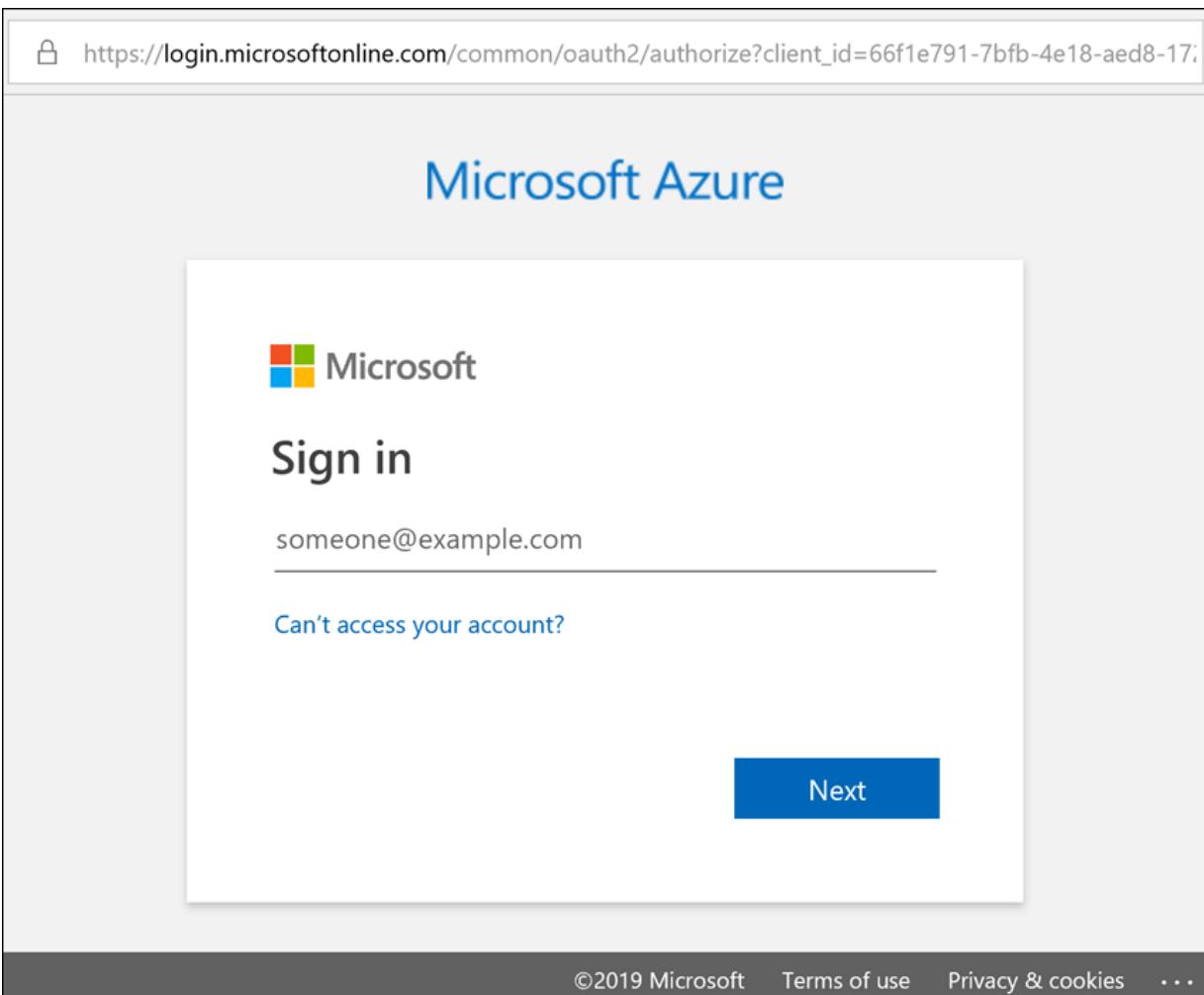
Don't have a Microsoft Power BI account yet?

[Sign up](#)

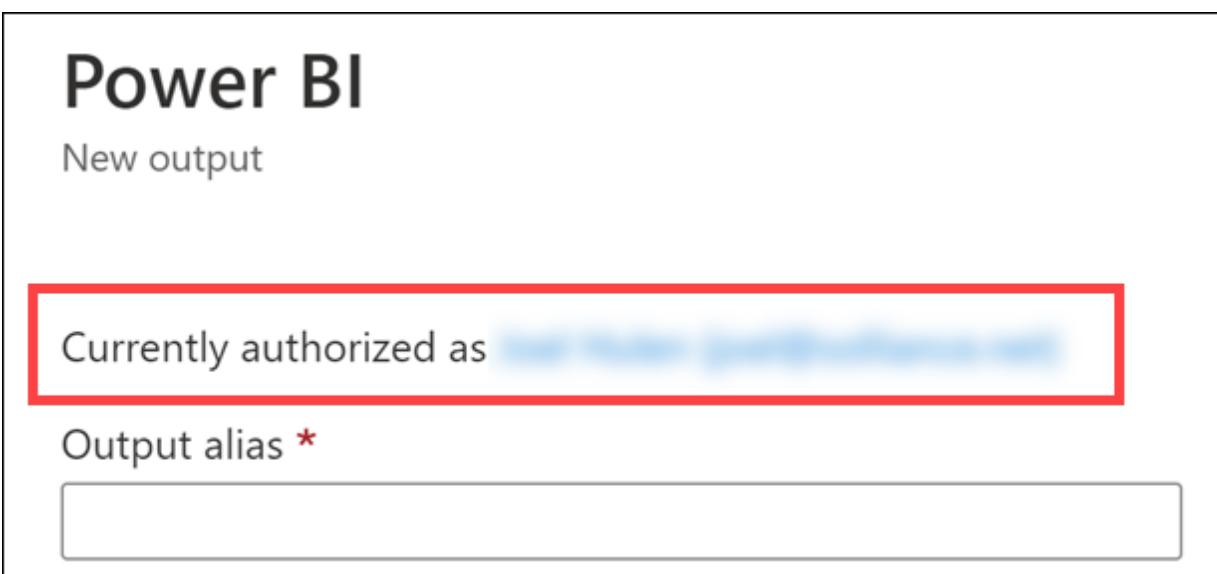


You are granting temporary access in order to get Power BI dashboard metadata.

14. When prompted, sign in to your Power BI account, which is the same username and password you were provided with and used to login to the Azure Portal.



15. After successfully signing in to your Power BI account, the New Output blade will update to show you are currently authorized.



16. In the **New Output** blade, configure the following:
 - **Output alias:** Enter `powerBIArtists`.
 - **Authentication mode:** Select "User token".
 - **Group workspace:** Select "My Workspace" (if you do not see this option, select the "User token" authentication mode first).
 - **Dataset name:** Enter `ContosoAutoVehicleAnomalies`.
 - **Table name:** Enter `Alerts`.

Power BI

X

New output

Currently authorized as [REDACTED]

Output alias *

powerBIArtists



Group workspace *

My workspace



Authentication mode

User token



Note: You are granting this output permanent access to your Power BI dashboard. Should you need to revoke this access in the future you can do one of the following:

1. Change the user account password.
2. Delete this output.
3. Delete this job.

Dataset name * ⓘ

ContosoAutoVehicleAnomalies



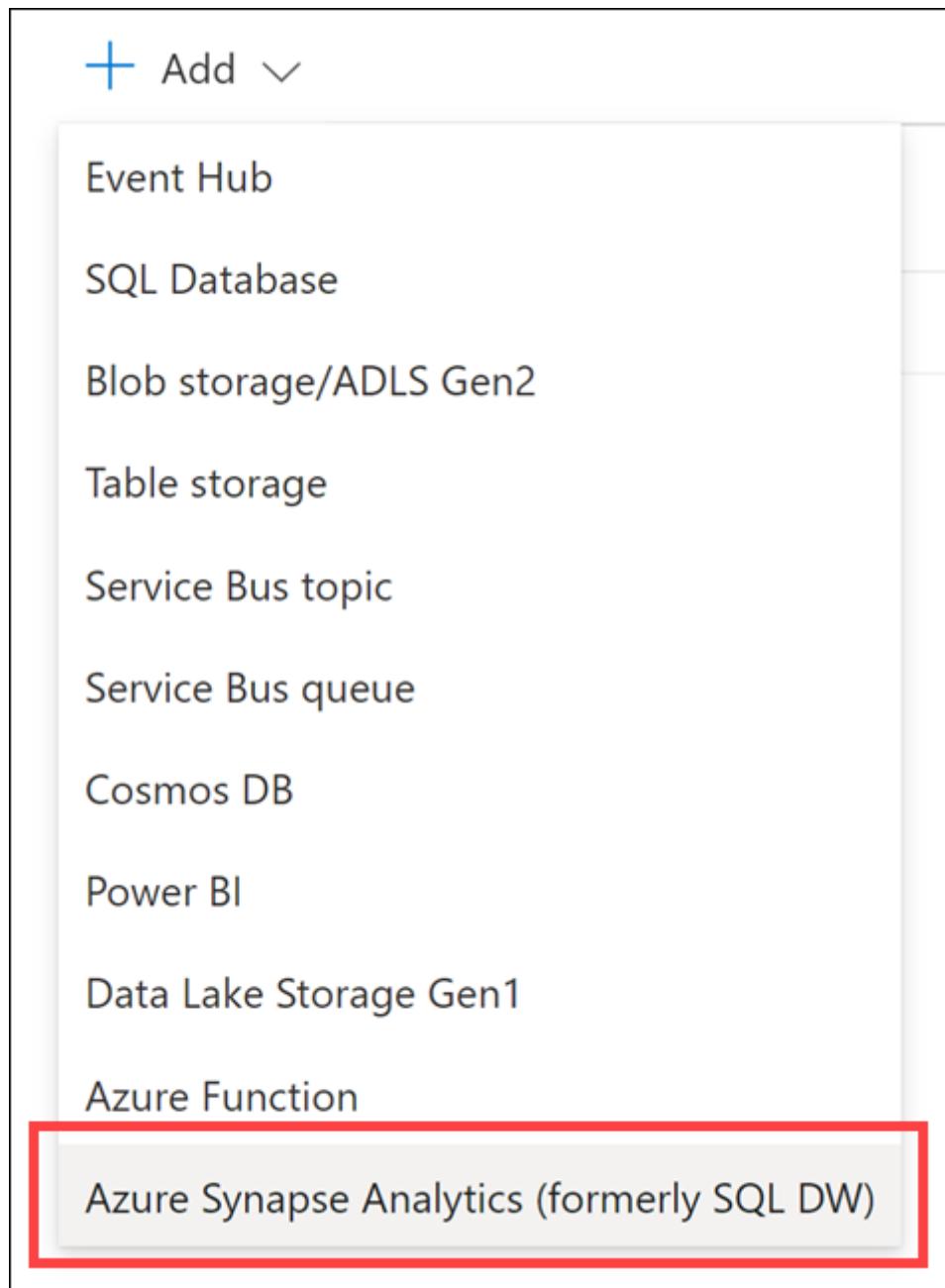
Table name *

Alerts



17. Select **Save** on the bottom of the form when you are finished entering the values.
18. Select **+** **Add** in the top toolbar, then select **Azure Synapse Analytics (formerly SQL DW)** to

create a new Synapse Analytics output.



19. In the **New Output** blade, configure the following:

- **Output alias:** Enter "synapse".
- **Select Azure Synapse Analytics from your subscriptions:** Selected.
- **Subscription:** Select the subscription you are using for this lab.
- **Database:** Select "ContosoAuto". Make sure your correct Synapse workspace name appears under "Server name".
- **Table:** Enter `dbo.VehicleAverages`
- **Authentication mode:** Select "Connection string".
- **Username:** Enter `asa.sql.admin`
- **Password:** Enter the SQL admin password value you entered when deploying the lab environment, or which was provided to you as part of your hosted lab environment. **Note:** This password is most likely not the same as the password you used to sign in to the Azure portal.

Azure Synapse Analytics (for...

X

New output

Output alias *

synapse



Provide SQL Database settings manually

Select SQL Database from your subscriptions

Subscription

Database *

ContosoAuto



Authentication mode

Connection string



Username *

asa.sql.admin



Password *



Server name

asaworkspace356363

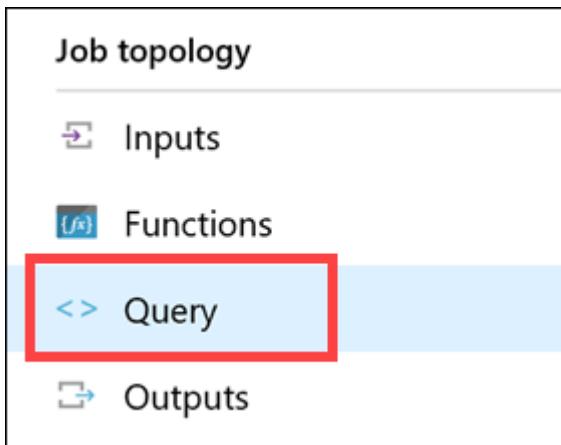
Table *

dbo.VehicleAverages



Save

20. Select **Save** on the bottom of the form when you are finished entering the values.
21. Within the Stream Analytics job blade, select **Query** within the left-hand menu.



22. Clear the edit **Query** window and paste the following in its place:

```

WITH
    Averages AS (
        select
            AVG(engineTemperature) averageEngineTemperature,
            AVG(speed) averageSpeed
        FROM
            eventhub TIMESTAMP BY [timestamp]
        GROUP BY
            TumblingWindow(Duration(second, 2))
    ),
    Anomalies AS (
        select
            t.vin,
            t.[timestamp],
            t.city,
            t.region,
            t.outsideTemperature,
            t.engineTemperature,
            a.averageEngineTemperature,
            t.speed,
            a.averageSpeed,
            t.fuel,
            t.engineoil,
            t.tirepressure,
            t.odometer,
            t.accelerator_pedal_position,
            t.parking_brake_status,
            t.headlamp_status,
            t.brake_pedal_status,
            t.transmission_gear_position,
            t.ignition_status,
            t.windshield_wiper_status,
            t.abs,
            (case when a.averageEngineTemperature >= 405 OR a.averageEngineTemperature <= 15 then 1 else 0 end) as oilanomaly,
            (case when (t.transmission_gear_position = 'first' OR
            t.transmission_gear_position = 'second' OR
            t.transmission_gear_position = 'third') AND
            t.brake_pedal_status = 1 AND
            t.accelerator_pedal_position >= 90 AND
            a.averageSpeed >= 55 then 1 else 0 end) as aggressivedriving
    )

```

```

from eventhub t TIMESTAMP BY [timestamp]
INNER JOIN Averages a ON DATEDIFF(second, t, a) BETWEEN 0 And 2
),
VehicleAverages AS (
    select
        AVG(engineTemperature) averageEngineTemperature,
        AVG(speed) averageSpeed,
        System.TimeStamp() as snapshot
FROM
    eventhub TIMESTAMP BY [timestamp]
GROUP BY
    TumblingWindow(Duration(minute, 2))
)
-- INSERT INTO POWER BI
SELECT
    *
INTO
    powerBIArtists
FROM
    Anomalies
where aggressivedriving = 1 OR enginetempanomaly = 1 OR oilanomaly = 1
-- INSERT INTO SYNAPSE ANALYTICS
SELECT
    *
INTO
    synapse
FROM
    VehicleAverages

```

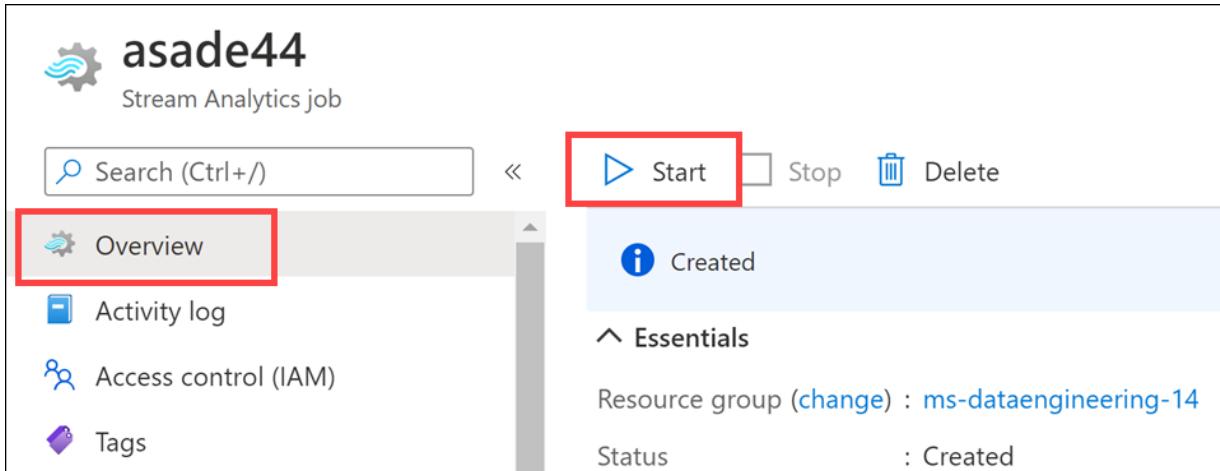
The screenshot shows the Azure Data Explorer interface with a query editor. On the left, there's a sidebar with 'Inputs (1)' containing 'eventhub' and 'Outputs (2)' containing 'powerBIArtists' and 'synapse'. The main area has tabs for 'Test query', 'Save query', and 'Discard changes'. The 'Test query' tab contains the DAX code from the previous snippet. A red box highlights the first 25 lines of the code, which define the 'Averages' and 'Anomalies' tables. Below the code, there are tabs for 'Input preview' and 'Test results'.

The query averages the engine temperature and speed over a two second duration. Then it selects all telemetry data, including the average values from the previous step, and specifies the following anomalies as new fields:

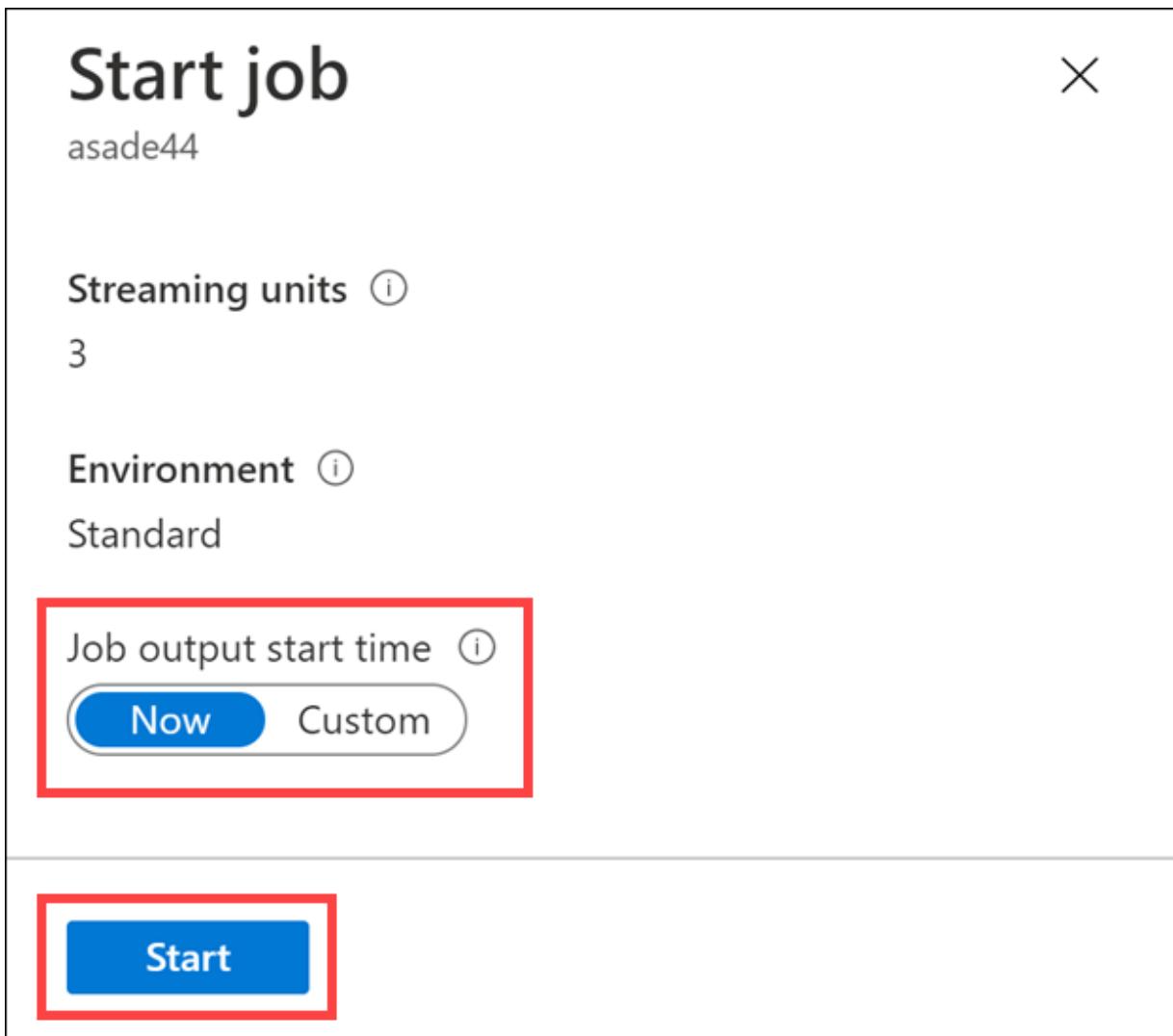
- enginetempanomaly:** When the average engine temperature is ≥ 405 or ≤ 15 .
- oilanomaly:** When the engine oil ≤ 1 .
- aggressivedriving:** When the transmission gear position is in first, second, or third, and the brake pedal status is 1, the accelerator pedal position ≥ 90 , and the average speed is ≥ 55 .

The query outputs all fields from the anomalies step into the `powerBIArtists` output where `aggressivedriving = 1` or `enginetempanomaly = 1` or `oilanomaly = 1` for reporting. The query also aggregates the average engine temperature and speed of all vehicles over the past two minutes, using `TumblingWindow(Duration(minute, 2))`, and outputs these fields to the `synapse` output.

23. Select **Save query** in the top toolbar when you are finished updating the query.
24. Within the Stream Analytics job blade, select **Overview** within the left-hand menu. On top of the Overview blade, select **Start**.



25. In the Start job blade that appears, select **Now** for the job output start time, then select **Start**. This will start the Stream Analytics job so it will be ready to start processing and sending your events to Power BI later on.



16.7 Exercise 2: Generate and visualize data

16.7.1 Task 1: Run data generator

The data generator console application creates and sends simulated vehicle sensor telemetry for an array of vehicles (denoted by VIN (vehicle identification number)) directly to Event Hubs. For this to happen, you first need to configure it with the Event Hub connection string.

In this task, you will configure and run the data generator. The data generator saves simulated vehicle telemetry data to Event Hubs, prompting your Stream Analytics job to aggregate and analyze the enriched data and send it to Power BI and Synapse Analytics. The final step will be to create the Power BI report in the task that follows.

1. On your lab VM or computer, download the [TransactionGeneratorExecutable.zip](#) file.
2. Extract the zip file to your machine, making note of the extraction location.
3. Open the folder containing the extracted files, then open either the `linux-x64`, `osx-x64`, or `win-x64` subfolder, based on your environment.
4. Within the appropriate subfolder, open the `appsettings.json` file. Paste your `telemetry` Event Hub connection string value next to `EVENT_HUB_CONNECTION_STRING`. Make sure you have quotes ("") around the value, as shown. **Save** the file.

```

1  {
2      "EVENT_HUB_CONNECTION_STRING": "Endpoint=sb://eventhubde44.servicebus.windows.net/;SharedAccessKeyName=Write;SharedAccessKey=0u52KB/R1KGy/avjytFMg7F2s",
3
4      "SECONDS_TO_LEAD": "0",
5      "SECONDS_TO_RUN": "1800"
6  }
7

```

`SECONDS_TO_LEAD` is the amount of time to wait before sending vehicle telemetry data. Default value is 0.

`SECONDS_TO_RUN` is the maximum amount of time to allow the generator to run before stopping transmission of data. The default value is 1800. Data will also stop transmitting when you enter Ctrl+C while the generator is running, or if you close the window.

5. Execute the data generator using one of the following methods, based on your platform:

1. Windows:

- Simply execute **TransactionGenerator.exe** inside the `win-x64` folder.

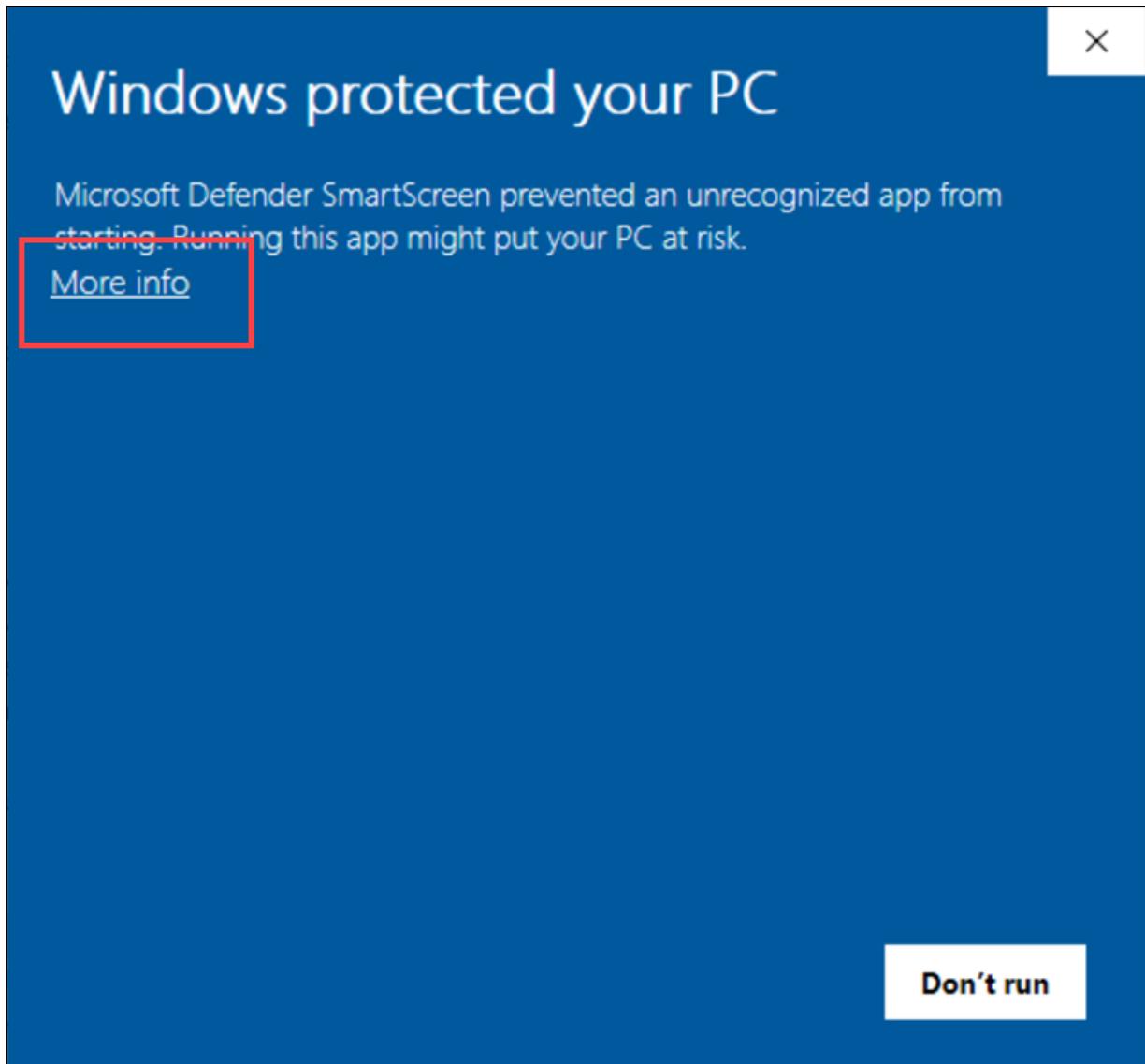
2. Linux:

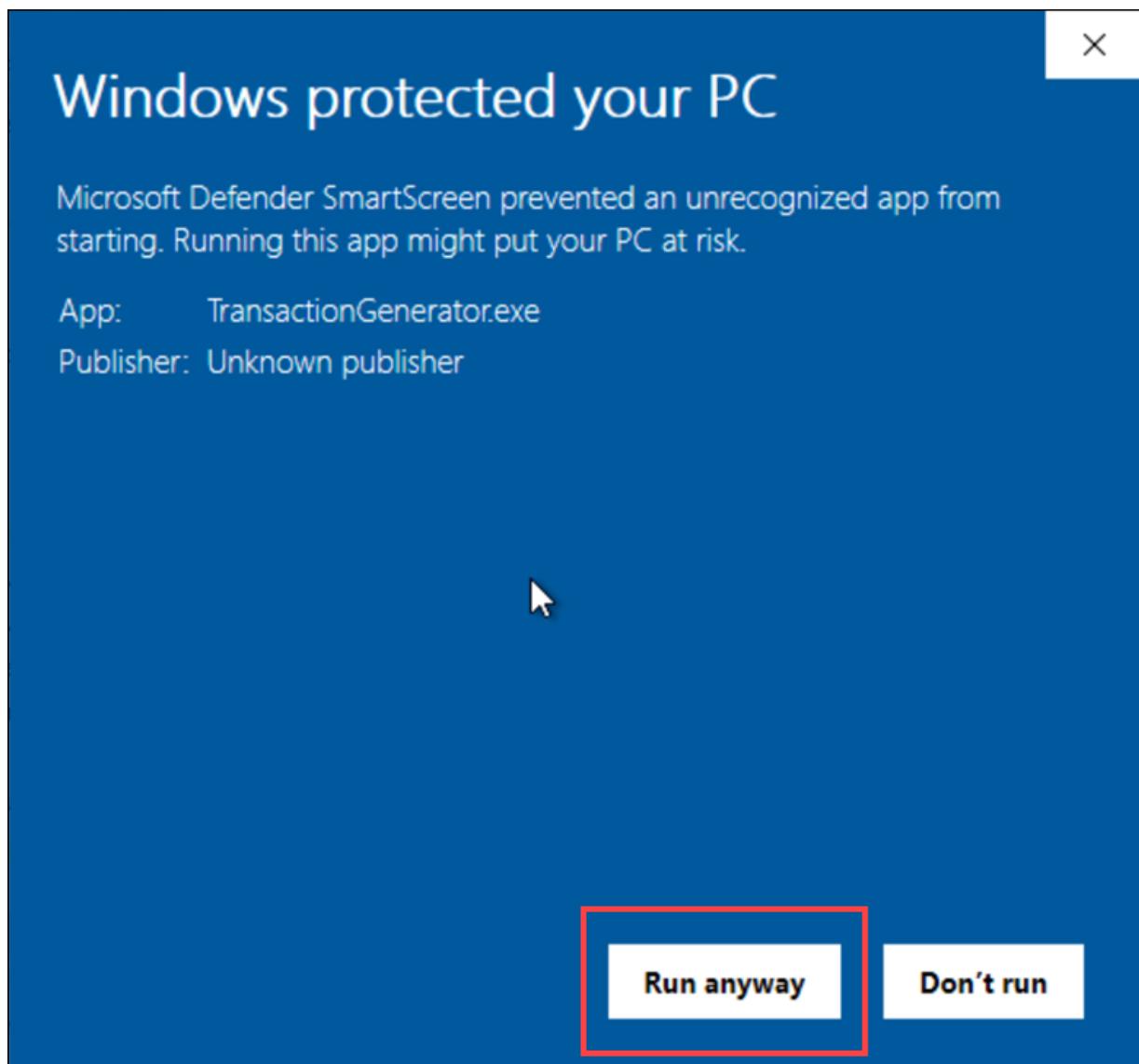
- Navigate to the `linux-x64` folder.
- Run `chmod 777 DataGenerator` to provide access to the binary.
- Run `./DataGenerator`.

3. MacOS:

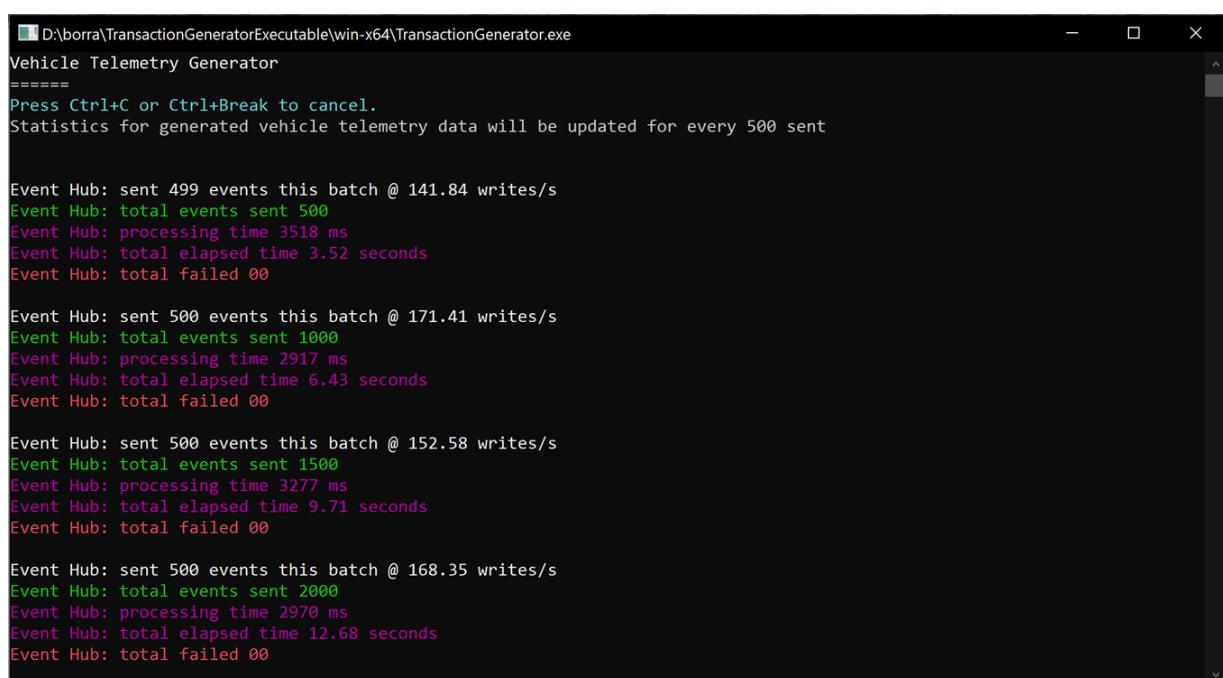
- Open a new terminal.
- Navigate to the `osx-x64` directory.
- Run `./DataGenerator`.

6. If you are using Windows and receive a dialog after trying to execute the data generator, select **More info**, then **Run anyway**.





7. A new console window will open, and you should see it start to send data after a few seconds. Once you see that it is sending data to Event Hubs, *minimize* the window and keep it running in the background.



```
D:\borra\TransactionGeneratorExecutable\win-x64\TransactionGenerator.exe
Vehicle Telemetry Generator
=====
Press Ctrl+C or Ctrl+Break to cancel.
Statistics for generated vehicle telemetry data will be updated for every 500 sent

Event Hub: sent 499 events this batch @ 141.84 writes/s
Event Hub: total events sent 500
Event Hub: processing time 3518 ms
Event Hub: total elapsed time 3.52 seconds
Event Hub: total failed 00

Event Hub: sent 500 events this batch @ 171.41 writes/s
Event Hub: total events sent 1000
Event Hub: processing time 2917 ms
Event Hub: total elapsed time 6.43 seconds
Event Hub: total failed 00

Event Hub: sent 500 events this batch @ 152.58 writes/s
Event Hub: total events sent 1500
Event Hub: processing time 3277 ms
Event Hub: total elapsed time 9.71 seconds
Event Hub: total failed 00

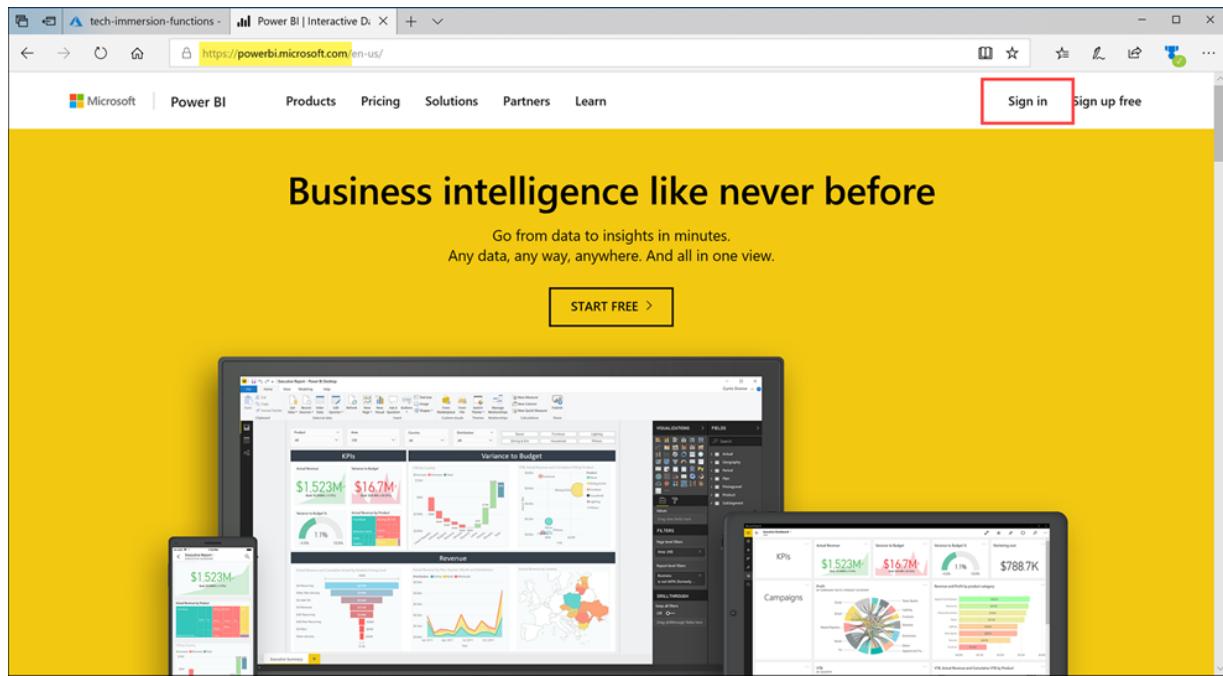
Event Hub: sent 500 events this batch @ 168.35 writes/s
Event Hub: total events sent 2000
Event Hub: processing time 2970 ms
Event Hub: total elapsed time 12.68 seconds
Event Hub: total failed 00
```

After every 500 records are requested to be sent, you will see output statistics.

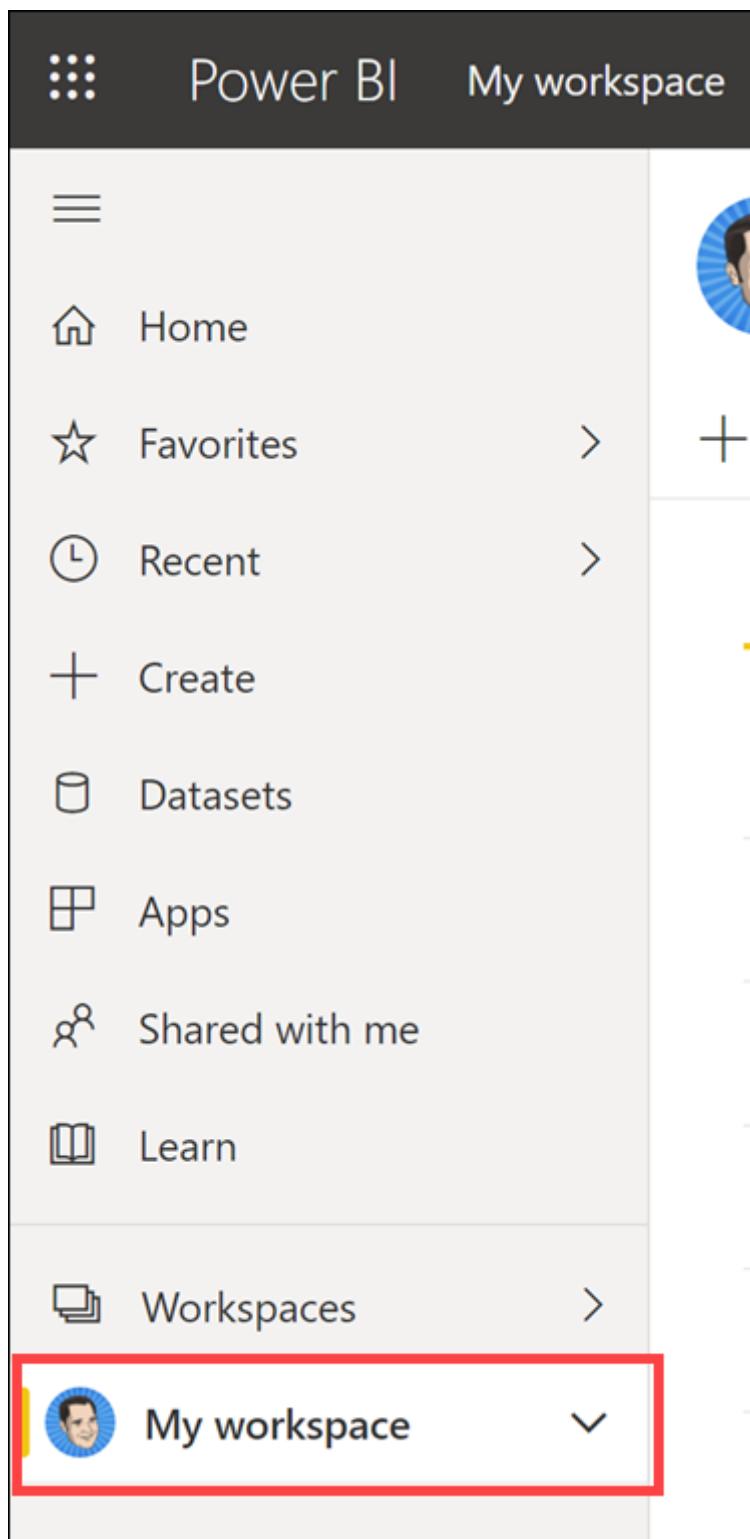
16.7.2 Task 2: Create Power BI dashboard

In this task, you will use Power BI to create a report showing captured vehicle anomaly data. Then you will pin that report to a live dashboard for near real-time updates.

1. Open your web browser and navigate to <https://powerbi.microsoft.com/>. Select **Sign in** on the upper-right.



2. Enter your Power BI credentials you used when creating the Power BI output for Stream Analytics.
3. After signing in, select **My Workspace** on the left-hand menu.



4. Select the **Datasets + dataflows** tab on top of the workspace. Locate the dataset named **ContosoAutoVehicleAnomalies**, then select the **Create Report** action button to the right of the name. If you do not see the dataset, you may need to wait a few minutes and refresh the page.

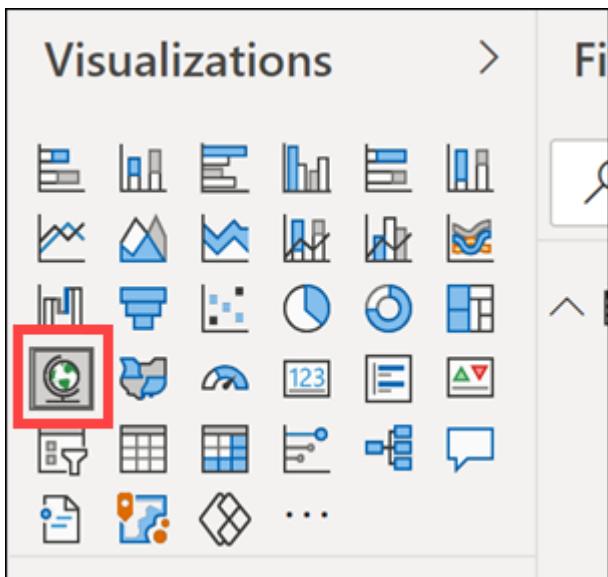
The screenshot shows the Power BI workspace interface. At the top, there's a navigation bar with 'My workspace' and a user profile icon. Below it, a search bar and a 'New' button are visible. The main area has tabs: 'All', 'Content', and 'Datasets + dataflows', with 'Datasets + dataflows' being the active tab and highlighted with a red box. A table lists datasets, showing columns for Name, Type, Owner, and Refreshed. One dataset, 'ContosoAutoVehicleAnomalies', is selected and has a context menu open. The 'Create report' option in this menu is also highlighted with a red box.

Note: It can take several minutes for the dataset to appear. You may need to periodically refresh the page before you see the Datasets tab.

5. You should see a new blank report for VehicleAnomalies with the field list on the far right.

The screenshot shows a Power BI report canvas. On the left is a navigation pane with options like Home, Favorites, Recent, Create, Datasets, Apps, Shared with me, Learn, Workspaces, and My workspace. The main area is titled 'Build visuals with your data' and says 'Select or drag fields from the Fields pane onto the report canvas.' To the right is a 'Fields' pane with a search bar and a tree view of fields. Under the 'Alerts' category, many fields are listed, including abs, accelerator_ped., aggressivedriving, averageenginetemp..., averagespeed, brake_pedal_st..., city, engineoil, enginetempano..., enginetemperat..., fuel, headlamp_status, ignition_status, odometer, oilanomaly, outsidetemperat..., parking_brake_s..., region, speed, timestamp, tirepressure, transmission_ge..., vin, and windshield_wipe...'. The 'Alerts' category is highlighted with a red box.

6. Select the **Map** visualization within the Visualizations section on the right.



7. Drag the **city** field to **Location**, and **aggressivedriving** to **Size**. This will place points of different sizes over cities on the map, depending on how many aggressive driving records there are.

The screenshot shows the Power BI interface with two main panes: 'Visualizations' on the left and 'Fields' on the right.

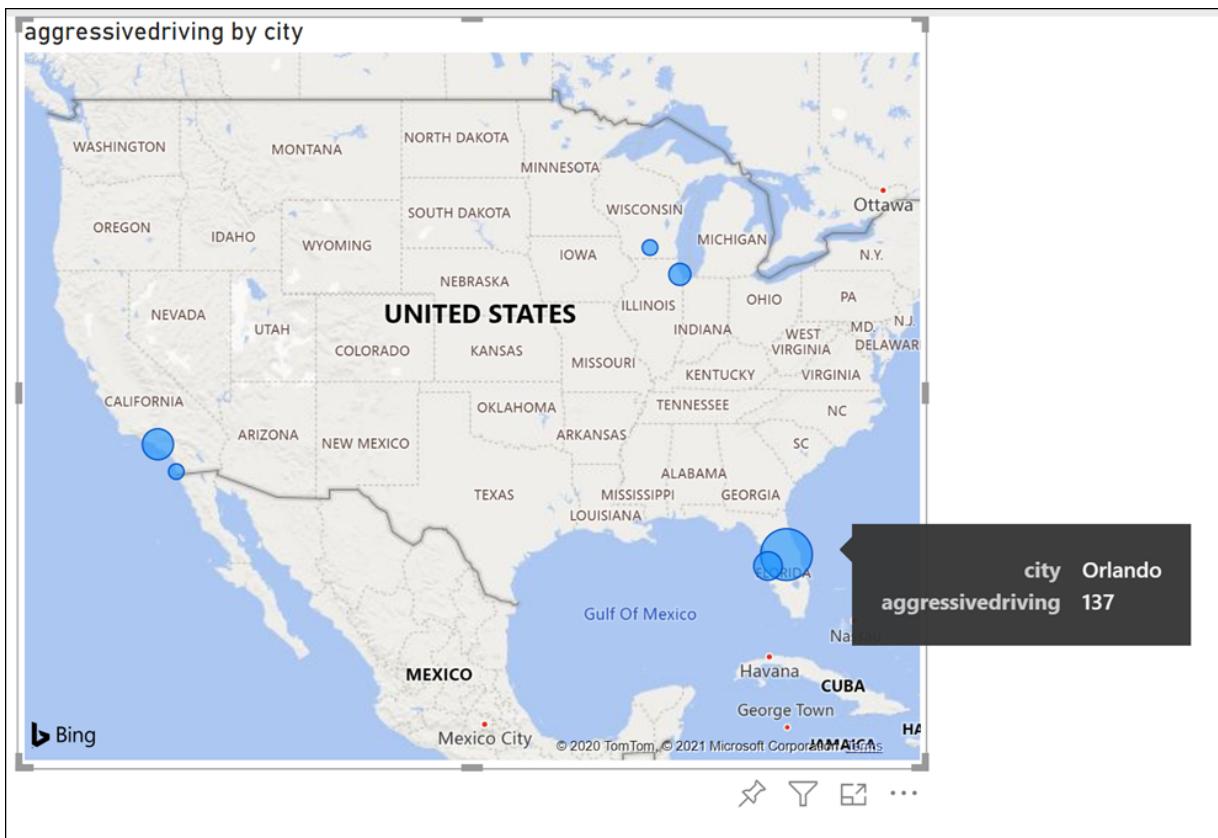
Visualizations pane:

- Icon grid: A grid of icons representing various visualization types like charts, maps, and tables.
- Location icon: Selected.
- City field input: 'city' (with a red arrow pointing to it).
- Legend section: 'Add data fields here'.
- Latitude section: 'Add data fields here'.
- Longitude section: 'Add data fields here'.
- Size section: 'aggressivedriving' (with a red arrow pointing to it).
- ToolTips section: 'Add data fields here'.

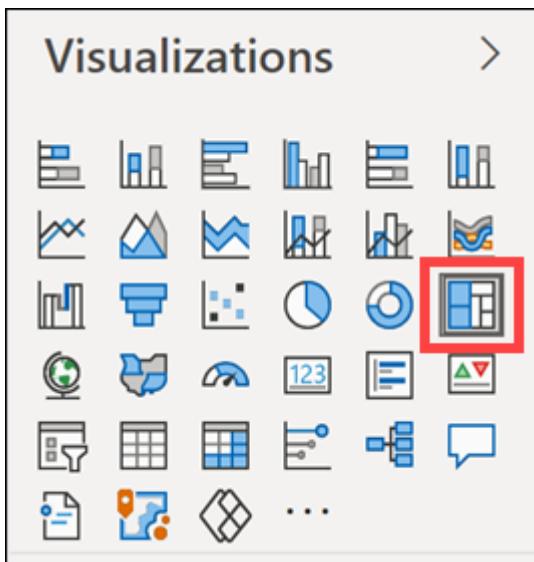
Fields pane:

- Search bar: 'Search'.
- Alerts group: Contains 'aggressivedriving' (selected), 'city' (selected), and other fields like 'abs', 'accelerator_ped...', 'averageenginete...', etc.
- Other fields listed: 'engineoil', 'enginetempano...', 'enginetemperat...', 'fuel', 'headlamp_status', 'ignition_status', 'odometer', 'oilanomaly', 'outsidetemperat...', 'parking_brake_s...', 'region'.

8. Your map should look similar to the following:



9. Select a blank area on the report to deselect the map. Now select the **Treemap** visualization.



10. Drag the **enginetemperature** field to **Values**, then drag the **transmission_gear_position** field to **Group**. This will group the engine temperature values by the transmission gear position on the treemap so you can see which gears are associated with the hottest or coolest engine temperatures. The treemap sizes the groups according to the values, with the largest appearing on the upper-left and the lowest on the lower-right.

Visualizations

Group

transmission_gear_posit

Details

Add data fields here

Values

enginetemperature

Tooltips

Add data fields here

Drill through

Cross-report

Off

Keep all filters

On

Add drill-through fields here

Fields

^ **Alerts**

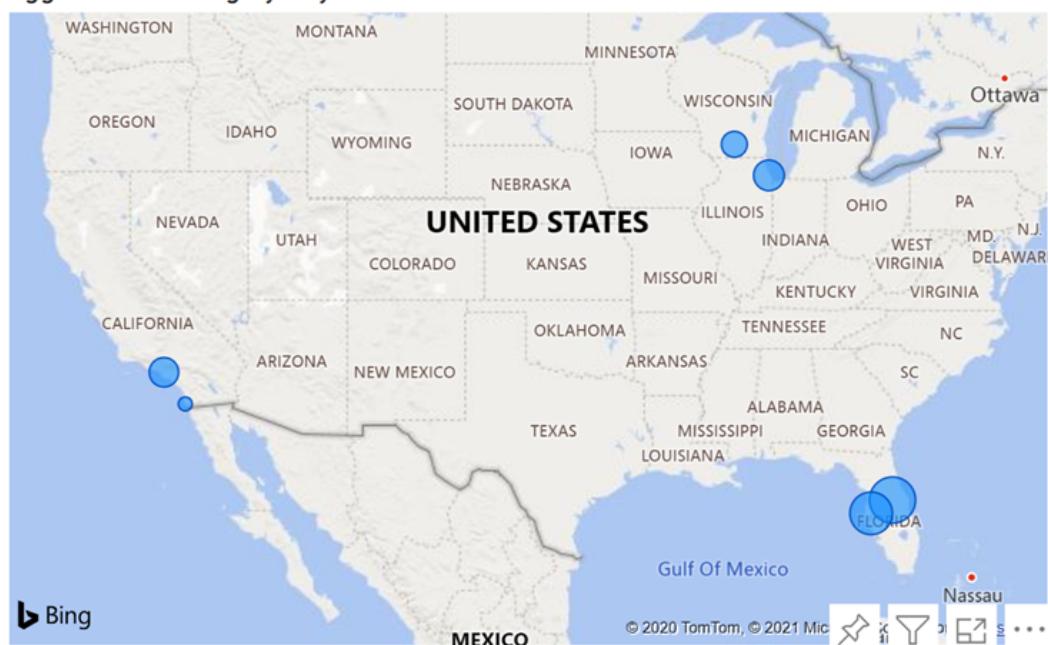
- Σ abs
- Σ accelerator_ped...
- Σ aggressivedriving
- Σ averageenginete...
- Σ averagespeed
- Σ brake_pedal_sta...
- city
- Σ engineoil
- Σ enginetempano...
- Σ enginetemperat...
- Σ fuel
- Σ headlamp_status
- Σ ignition_status
- Σ odometer
- Σ oilanomaly
- Σ outsidetemperat...
- Σ parking_brake_s...
- region
- Σ speed
- timestamp
- Σ tirepressure
- transmission_ge...
- vin

11. Select the down arrow next to the **enginetemperature** field under **Values**. Select **Average** from the menu to aggregate the values by average instead of the sum.

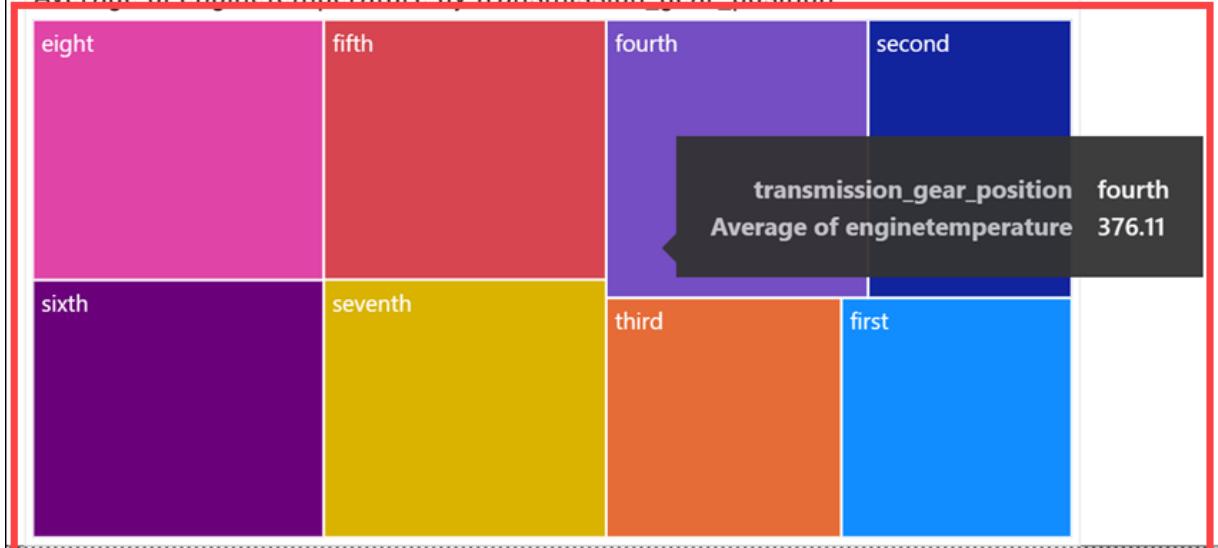
The screenshot shows the Power BI Fields pane. On the left, there's a grid of visualization icons. Below them are buttons for Group, Details, Add data fields here, Values, and Tooltips. The 'Values' button is highlighted with a yellow bar. In the center, under 'Values', there's a dropdown menu for the field 'enginetemperature'. The menu items are: Remove field, Rename for this visual, Move to, Sum, Average (which is selected and highlighted with a red box), Minimum, Maximum, Count (Distinct), Count, Standard deviation, Variance, Median, Show value as, and Ignition status. At the bottom of the pane, there's a checkbox for 'Σ ignition_status'.

12. Your treemap should look similar to the following:

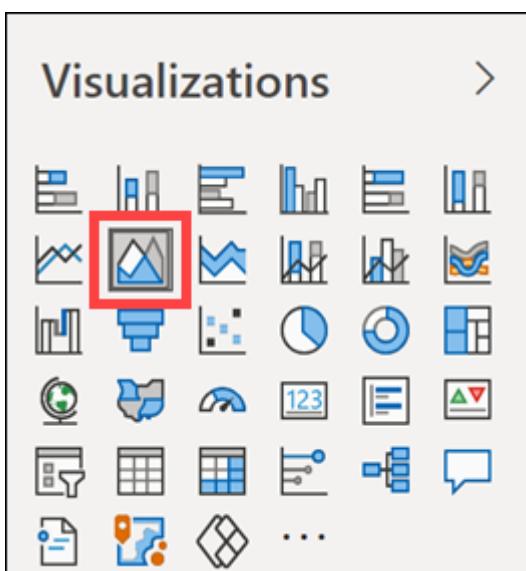
aggressivedriving by city



Average of enginetemperature by transmission_gear_position



13. Select a blank area on the report to deselect the treemap. Now select the **Area chart** visualization.



14. Drag the **region** field to **Legend**, the **speed** field to **Values**, and the **timestamp** field to **Axis**. This will display an area chart with different colors indicating the region and the speed at which drivers travel over time within that region.

Visualizations

Axis

timestamp

Legend

region

Values

speed

Secondary values

Add data fields here

Tooltips

Add data fields here

Drill through

Cross-report

Off

Fields

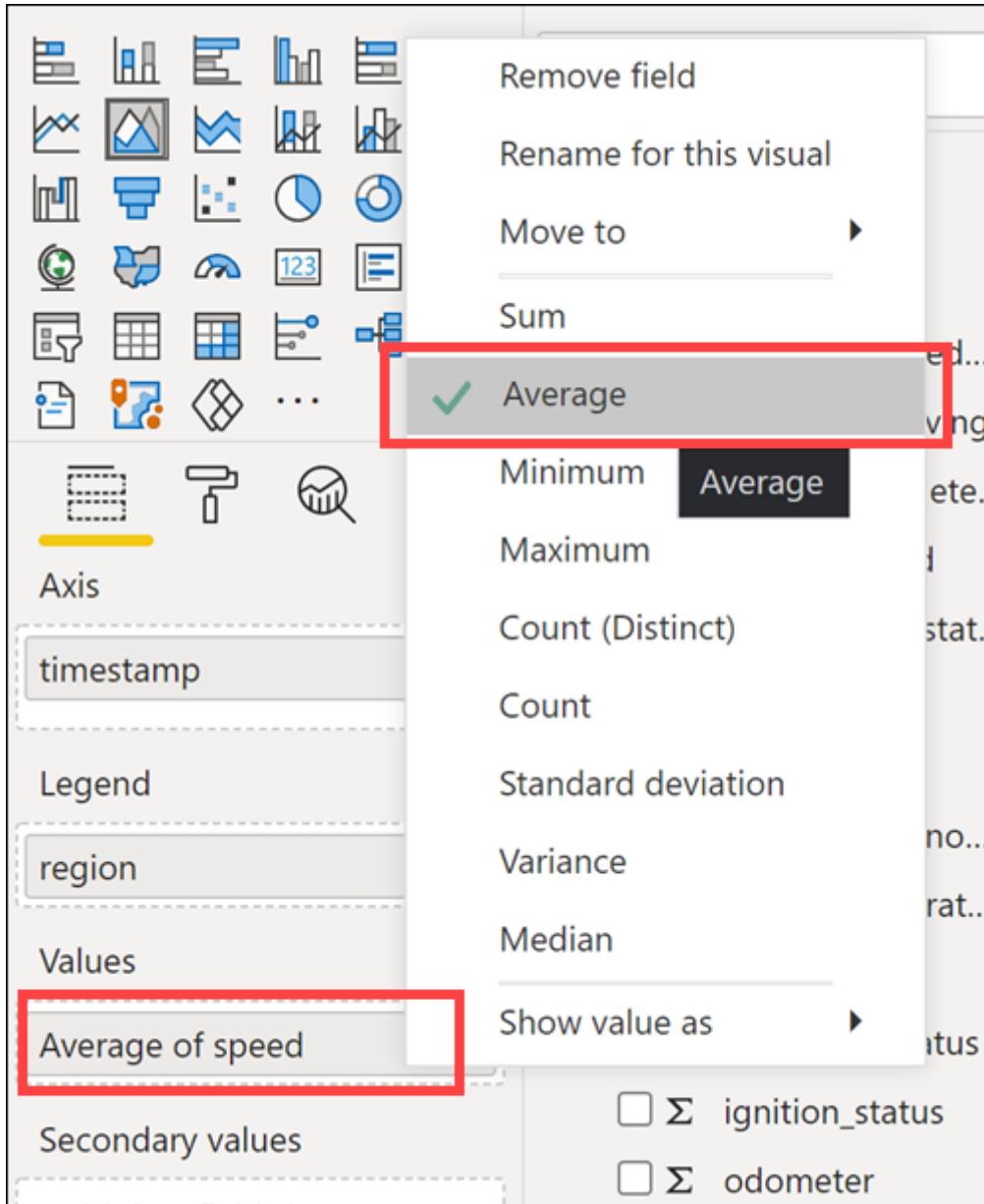
Search

Alerts

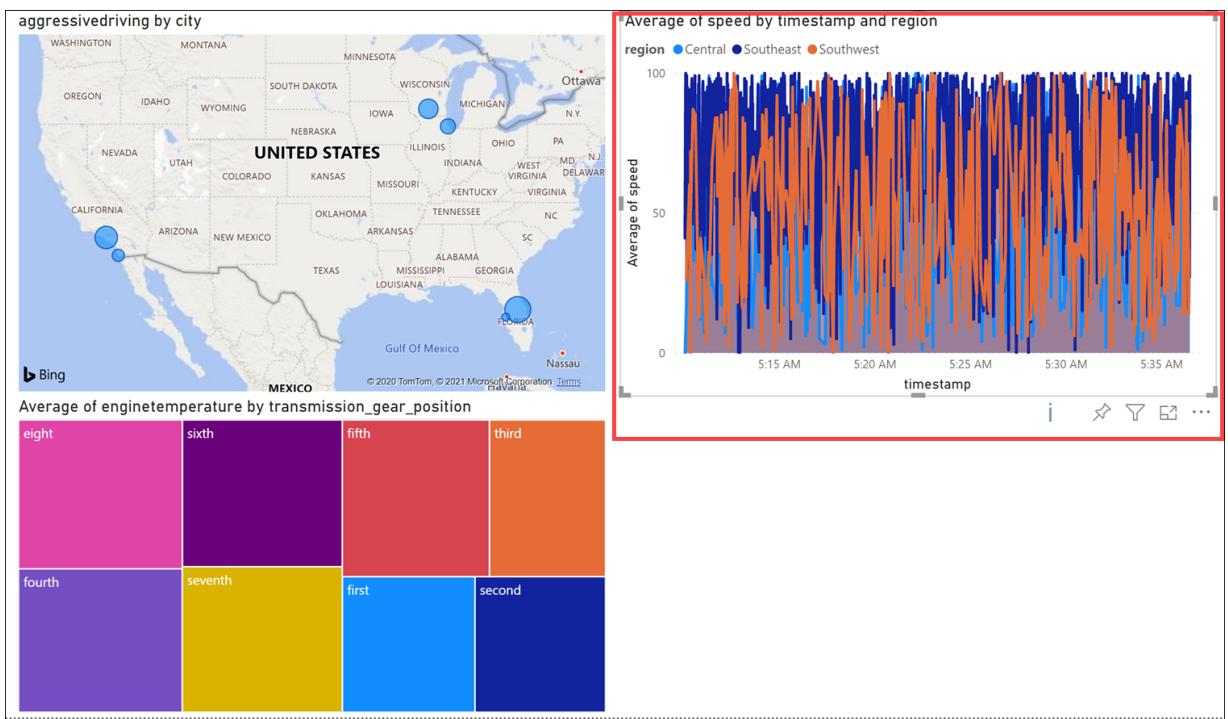
- Σ abs
- Σ accelerator_ped...
- Σ aggressivedriving
- Σ averageenginete...
- Σ averagespeed
- Σ brake_pedal_stat...
- city
- Σ engineoil
- Σ enginetempano...
- Σ enginetemperat...
- Σ fuel
- Σ headlamp_status
- Σ ignition_status
- Σ odometer
- Σ oilanomaly
- Σ outsidetemperat...
- Σ parking_brake_st...
- region
- Σ speed
- timestamp
- Σ tirepressure
- transmission_ge...

492

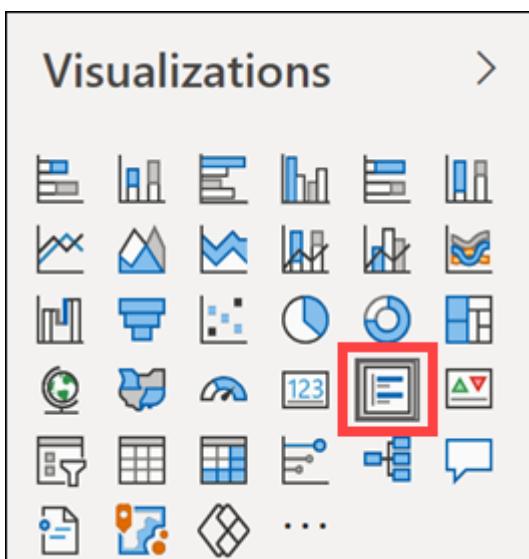
15. Select the down arrow next to the **speed** field under **Values**. Select **Average** from the menu to aggregate the values by average instead of the sum.



16. Your area chart should look similar to the following:



17. Select a blank area on the report to deselect the area chart. Now select the **Multi-row card** visualization.

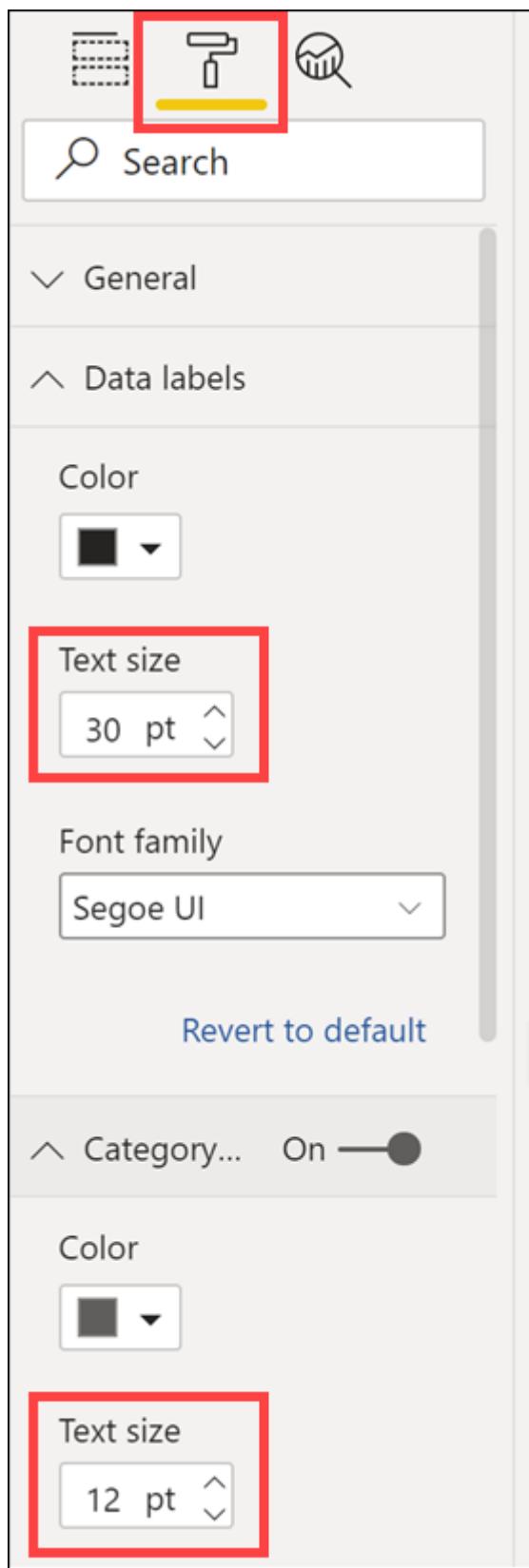


18. Drag the **aggressivedriving** field, **enginetempanomaly**, and **oilanomaly** fields to **Fields**.

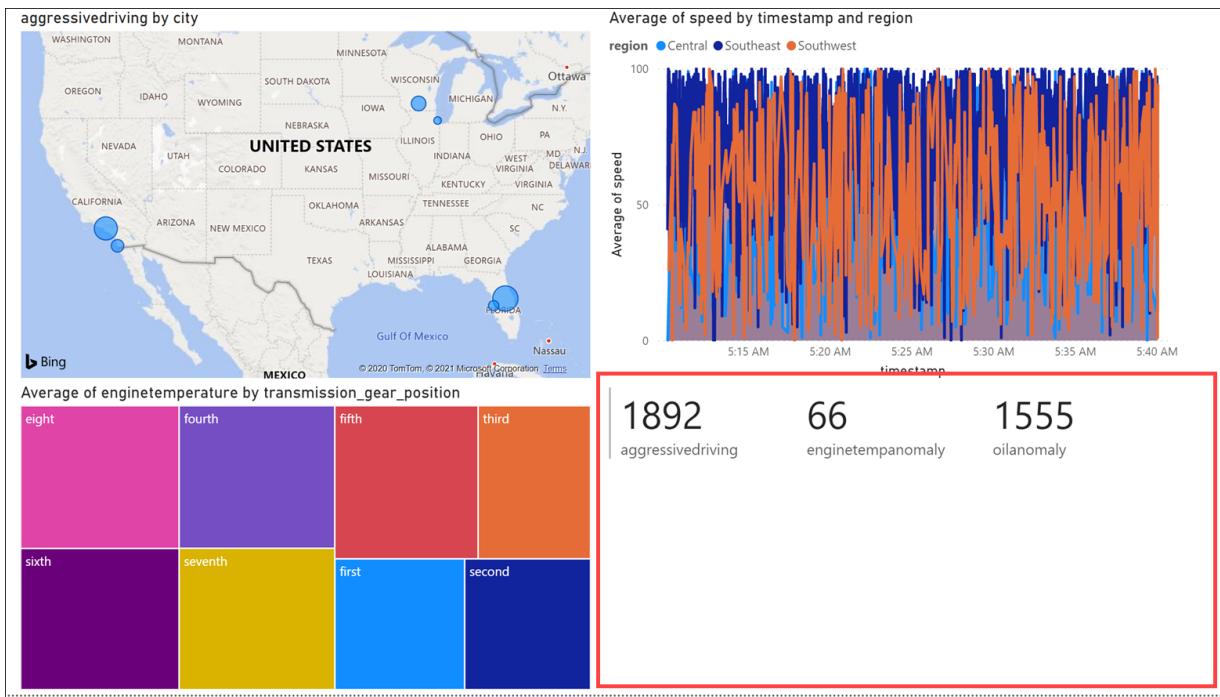
The screenshot shows the Power BI Fields pane on the right and the Visualizations pane on the left. In the Fields pane, under the 'Alerts' category, several fields are listed with checkboxes. Three specific fields are highlighted with yellow checkmarks: 'aggressivedriving', 'enginetempanomaly', and 'oilanomaly'. Red arrows point from these three highlighted fields in the Fields pane down to their corresponding entries in the 'Fields' section of the multi-row card settings in the Visualizations pane. The 'Fields' section contains three rows, each with a dropdown menu icon, a field name, and a close button ('X'). The first row is 'aggressivedriving', the second is 'enginetempanomaly', and the third is 'oilanomaly'. The 'Drill through' section below the card settings includes options: 'Cross-report', 'Off', 'Keep all filters', and 'On'. A dashed box labeled 'Add drill-through fields here' is at the bottom.

Field	Status
abs	<input type="checkbox"/>
accelerator_ped...	<input type="checkbox"/>
aggressivedriving	<input checked="" type="checkbox"/>
averageenginete...	<input type="checkbox"/>
averagespeed	<input type="checkbox"/>
brake_pedal_stat...	<input type="checkbox"/>
city	<input type="checkbox"/>
engineoil	<input type="checkbox"/>
enginetempano...	<input checked="" type="checkbox"/>
enginetemperat...	<input type="checkbox"/>
fuel	<input type="checkbox"/>
headlamp_status	<input type="checkbox"/>
ignition_status	<input type="checkbox"/>
odometer	<input type="checkbox"/>
oilanomaly	<input checked="" type="checkbox"/>
outsidetemperat...	<input type="checkbox"/>
parking_brake_st...	<input type="checkbox"/>

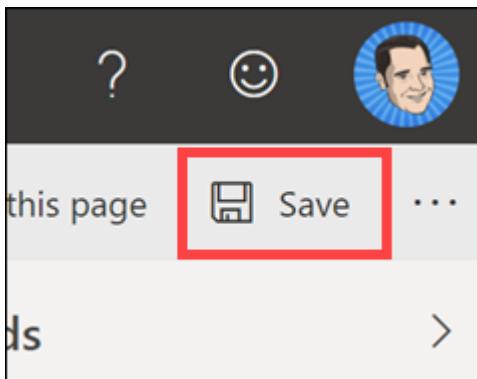
19. Select the **Format** tab in the multi-row card settings, then expand **Data labels**. Set the **Text size** to 30. Expand **Category labels** and set the **Text size** to 12.



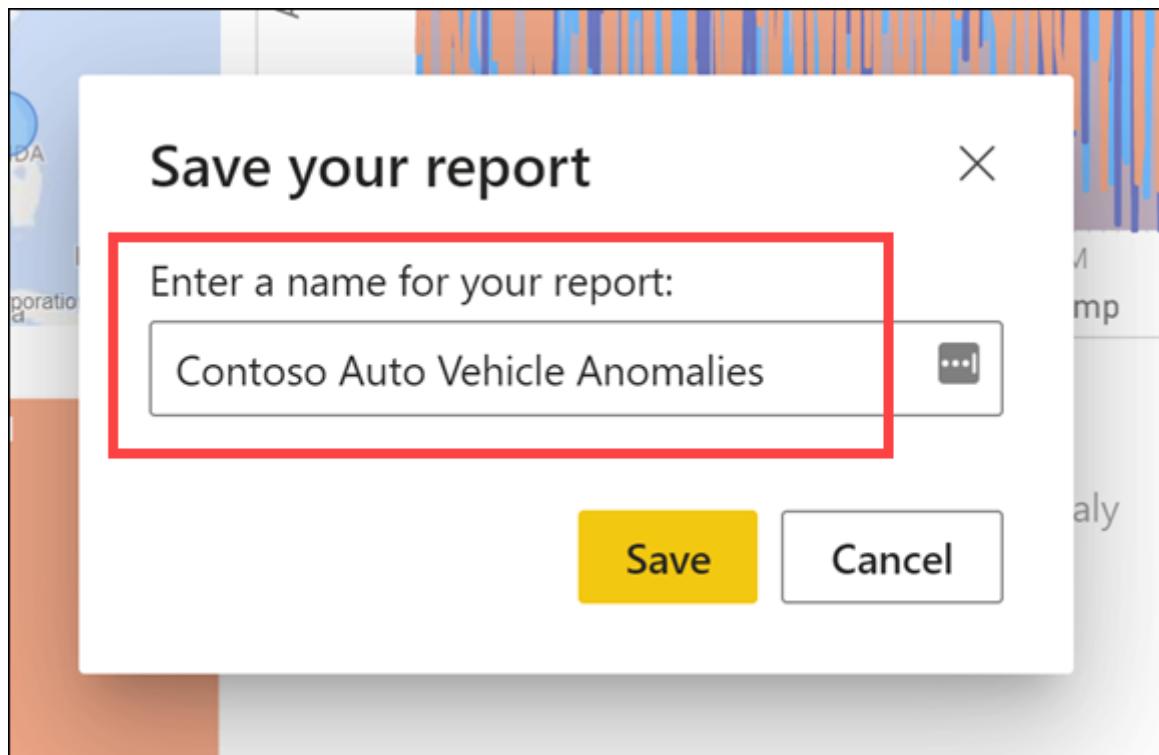
20. Your multi-row card should look similar to the following:



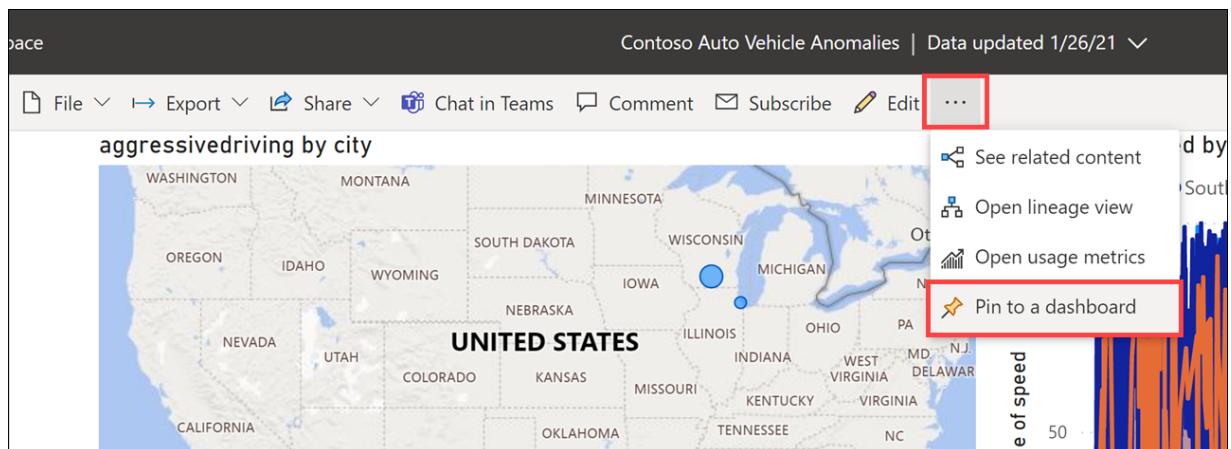
21. Select **Save** on the upper-right of the page.



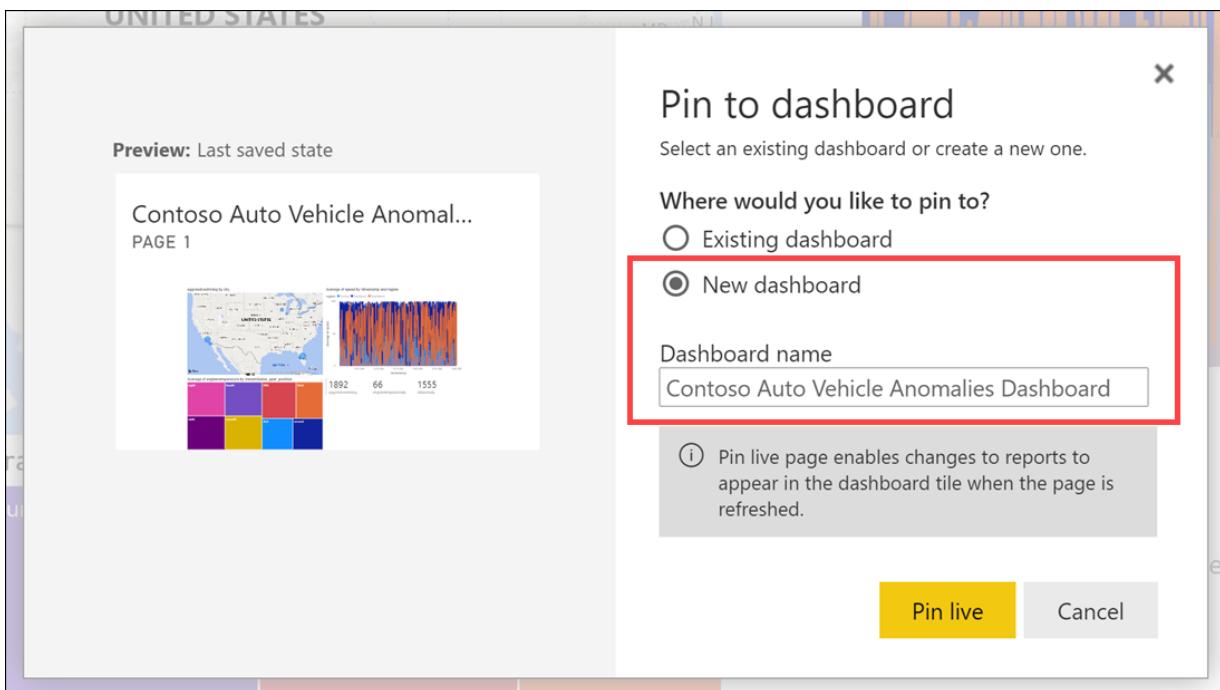
22. Enter a name, such as "Contoso Auto Vehicle Anomalies", then select **Save**.



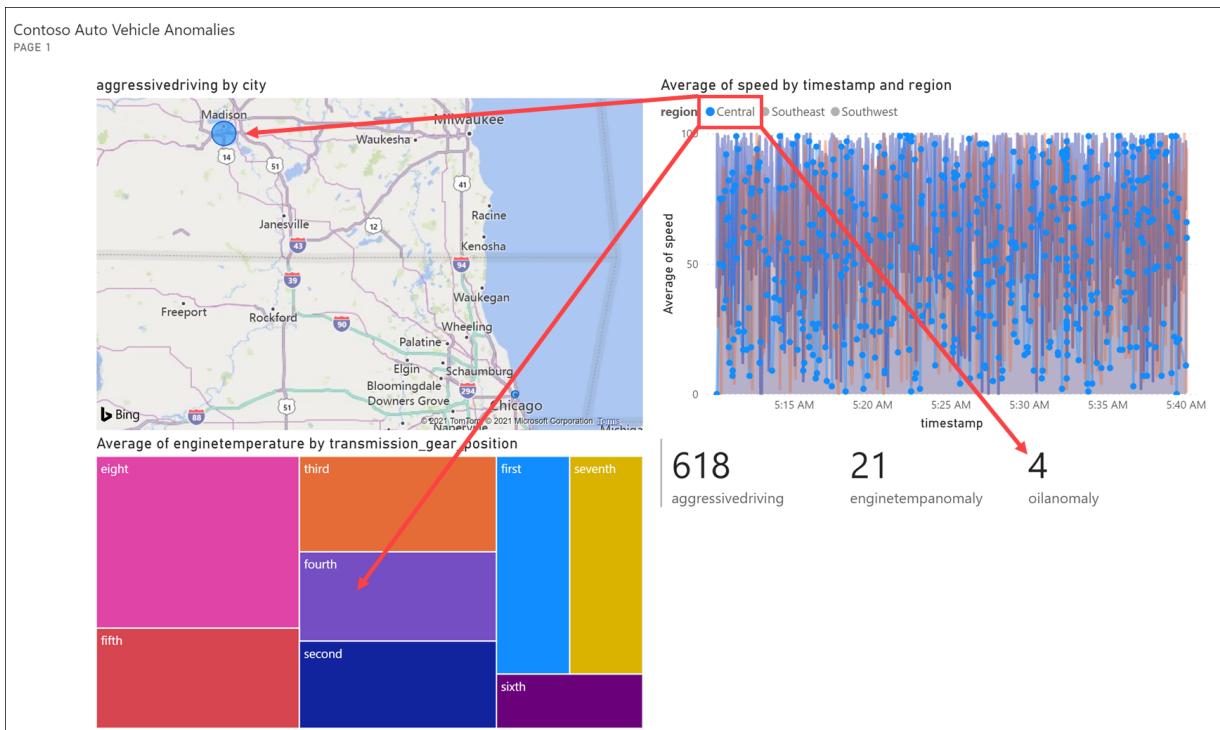
23. Now let's add this report to a dashboard. Select **Pin to a dashboard** at the top of the report (you may have to select the ellipses ...).



24. Select **New dashboard**, then enter a name, such as "Contoso Auto Vehicle Anomalies Dashboard". Select **Pin live**. When prompted select the option to view the dashboard. Otherwise, you can find the dashboard under My Workspace on the left-hand menu.



25. The live dashboard will automatically refresh and update while data is being captured. You can hover over any point on a chart to view information about the item. Select one of the regions in the legend above the average speed chart. All other charts will filter by that region automatically. Click on the region again to clear the filter.



16.7.3 Task 3: View aggregate data in Synapse Analytics

As you recall, when you created the query in Stream Analytics, you aggregated the engine temperature and vehicle speed data over two-minute intervals and saved the data to Synapse Analytics. This capability demonstrates the Stream Analytics query's ability to write data to multiple outputs at varying intervals. Writing to a Synapse Analytics dedicated SQL pool enables us to retain the historic and current aggregate data as part of the data warehouse without requiring an ETL/ELT process.

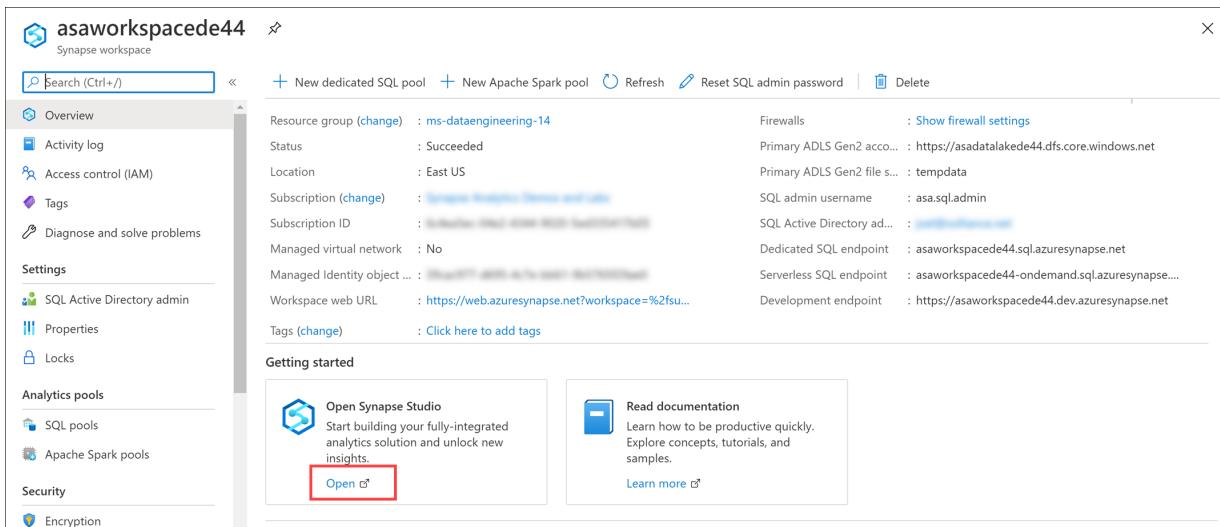
In this task, you will view the anomaly data within Synapse Analytics.

1. If you have not yet done so, **stop** the **TransactionGenerator**.

2. Navigate to the [Azure portal](#).
3. Select **Resource groups** from the left-hand menu. Then select the resource group named **ms-dataengineering-14**.
4. Select the **Synapse workspace** (`asaworkspaceYOUR_UNIQUE_ID`) from the list of resources in your resource group.

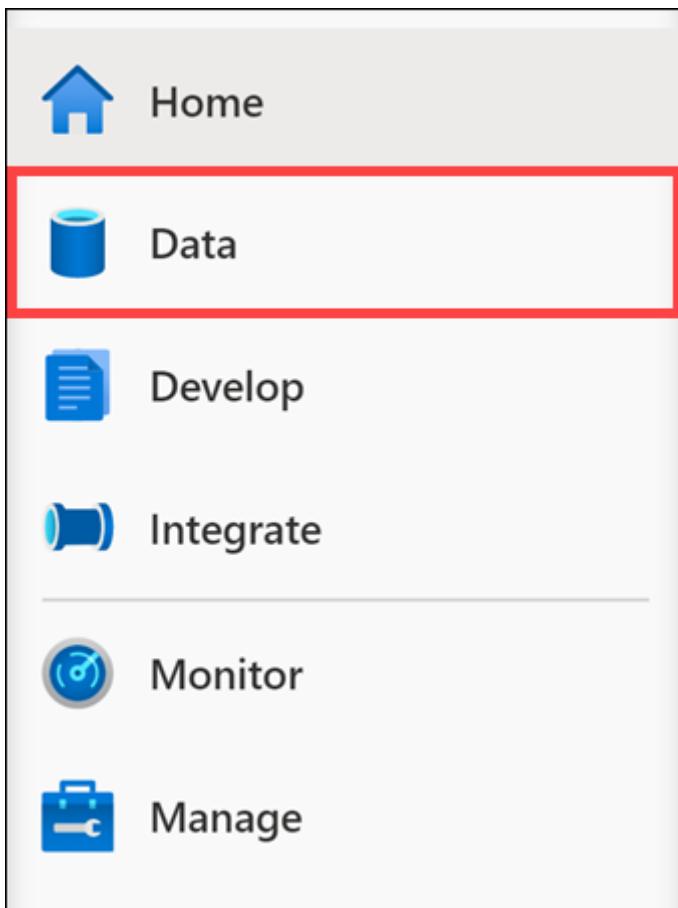
<input type="checkbox"/> Name ↑↓	Type ↑↓
<input type="checkbox"/>  asadatalakede44	Storage account
<input type="checkbox"/>  asade44	Stream Analytics job
<input checked="" type="checkbox"/>  asaworkspacede44	Synapse workspace
<input type="checkbox"/>  ContosoAuto (asaworkspacede44/ContosoAuto)	Dedicated SQL pool
<input type="checkbox"/>  eventhubde44	Event Hubs Namespace

5. Select **Open** within the **Open Synapse Studio** box inside the Overview pane.



The screenshot shows the Azure portal overview for the 'asaworkspacede44' Synapse workspace. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (SQL Active Directory admin, Properties, Locks), Analytics pools (SQL pools, Apache Spark pools), and Security (Encryption). The main pane displays workspace details such as Resource group (ms-dataengineering-14), Status (Succeeded), Location (East US), Subscription (change), Subscription ID, Managed virtual network (No), Managed Identity object (redacted), Workspace web URL (<https://web.azure.synapse.net?workspace=%2fsu...>), and Tags (change). In the 'Getting started' section, there are two boxes: 'Open Synapse Studio' (with a red box around the 'Open' button) and 'Read documentation' (with a 'Learn more' link).

6. Within Synapse Studio, select **Data** in the left-hand menu to navigate to the Data hub.



7. Select the **Workspace** tab (1), expand the ContosoAuto database, expand **Tables**, then right-click on the **dbo.VehicleAverages** table (2). If you do not see the table listed, refresh the tables list. Select **New SQL script** (3), then **Select TOP 100 rows** (4).

A screenshot of the Power BI workspace under the 'Data' tab. Step 1 highlights the 'Workspace' tab. Step 2 highlights the 'dbo.VehicleAverages' table in the 'Tables' section of the 'ContosoAuto' database. Step 3 highlights the 'New SQL script' option in the context menu. Step 4 highlights the 'Select TOP 100 rows' option in the context menu dropdown.

8. View the query results and observe the aggregate data stored in **AverageEngineTemperature** and **AverageSpeed**. The **Snapshot** value changes in two-minute intervals between these records.

SQL script 2

Run Undo Publish Query plan Connect to ContosoAuto Use database ContosoAuto

```

1  SELECT TOP (100) [AverageEngineTemperature]
2  ,[AverageSpeed]
3  ,[Snapshot]
4  | FROM [dbo].[VehicleAverages]

```

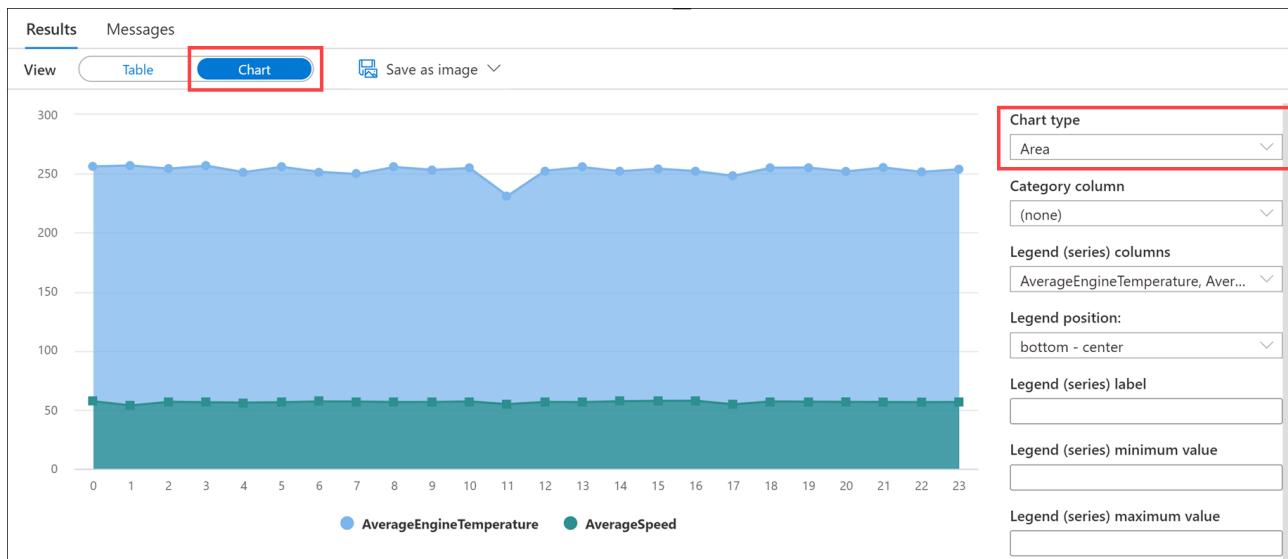
Results Messages

View Table Chart Export results

Search

AverageEngineTemperature	AverageSpeed	Snapshot
256.166471865086	57.1262314242778	2021-01-26T05:32:00.0000000
256.908745247148	53.6653992395437	2021-01-26T05:42:00.0000000
254.266666666667	56.6684144818976	2021-01-26T05:04:00.0000000
256.703214890017	56.2942470389171	2021-01-26T05:12:00.0000000
251.264119322943	56.0288252052958	2021-01-26T05:20:00.0000000
255.816524310118	56.2789093298292	2021-01-26T05:22:00.0000000
251.562739543409	57.0508248625229	2021-01-26T05:16:00.0000000
249.6855	56.89175	2021-01-26T04:30:00.0000000
255.688074874939	56.5436501533	2021-01-26T05:18:00.0000000
253.208571428571	56.5671428571429	2021-01-26T04:28:00.0000000

- Select the **Chart** view in the Results output, then set the chart type to **Area**. This visualization shows the average engine temperature correlated with the average speed over time. Feel free to experiment with the chart settings.



16.8 Exercise 3: Cleanup

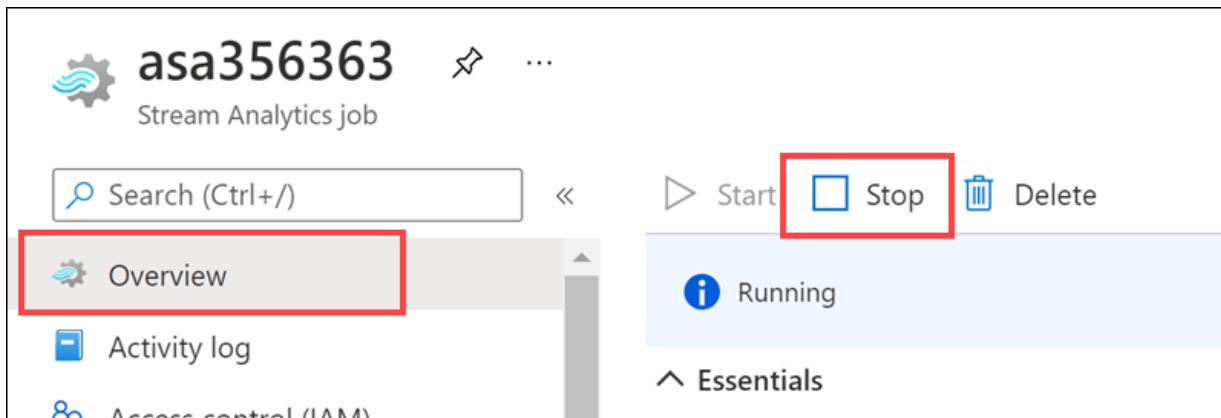
Complete these steps to stop the data generator and free up resources you no longer need.

16.8.1 Task 1: Stop the data generator

- Go back to the console/terminal window in which your data generator is running. Close the window to stop the generator.

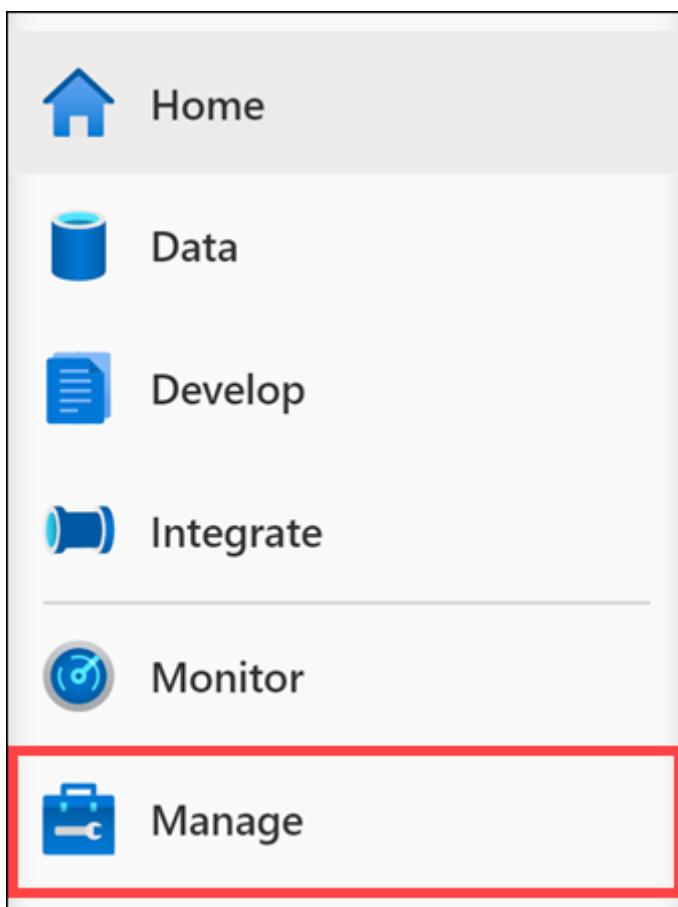
16.8.2 Task 2: Stop the Stream Analytics job

1. Navigate to the Stream Analytics job in the Azure portal.
2. In the Overview pane, select **Stop**, then select **Yes** when prompted.



16.8.3 Task 3: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

The serverless SQL pool, Built-in, is immediately available for your workspace. Dedicated SQL pools can be configured

+ New Refresh System-assigned managed identity

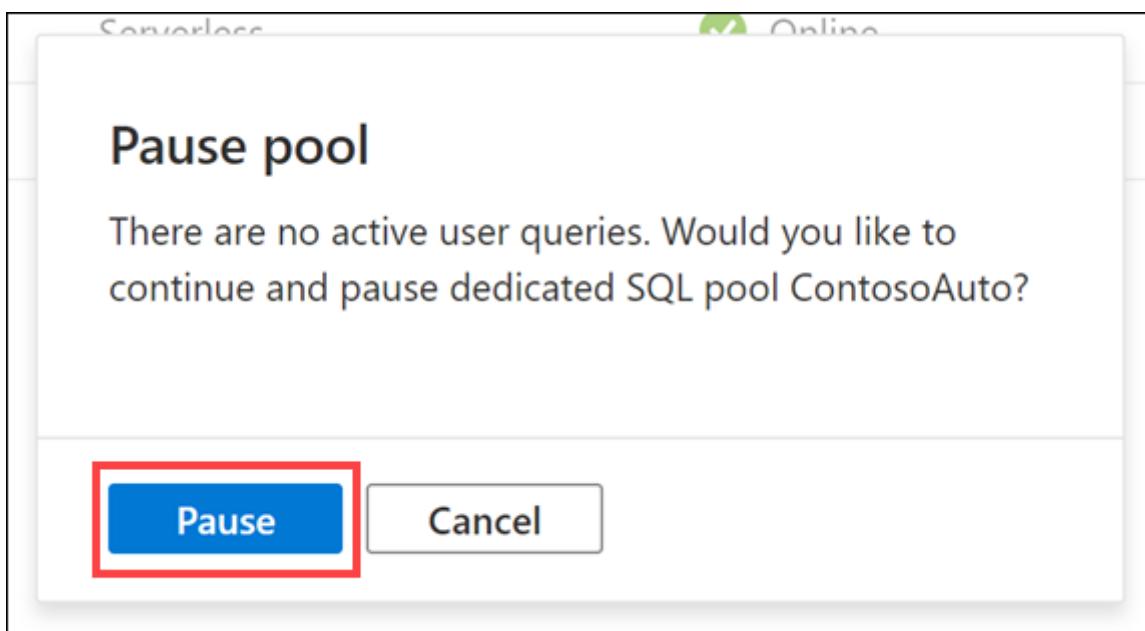
Filter by name

Showing 1-2 of 2 items (1 Serverless, 1 Dedicated)

Name	Type	Status
Built-in	Serverless	Online
ContosoAuto	Dedicated	Online

Pause

- When prompted, select **Pause**.



SOS

and other diagnostic tools now ship of band and work with any version of the .NET Core runtime.

SOS has moved to the diagnostics repo here: <https://github.com/dotnet/diagnostics.git>.

Instructions to install SOS: <https://github.com/dotnet/diagnostics#installing-sos>. SOS and other diagnostic tools now ship of band and work with any version of the .NET Core runtime.

SOS has moved to the diagnostics repo here: <https://github.com/dotnet/diagnostics.git>.

Instructions to install SOS: <https://github.com/dotnet/diagnostics#installing-sos>. SOS and other diagnostic tools now ship of band and work with any version of the .NET Core runtime.

SOS has moved to the diagnostics repo here: <https://github.com/dotnet/diagnostics.git>.

Instructions to install SOS: <https://github.com/dotnet/diagnostics#installing-sos>.

17 Module 15 - Create a stream processing solution with Event Hubs and Azure Databricks

In this module, students will learn how to ingest and process streaming data at scale with Event Hubs and Spark Structured Streaming in Azure Databricks. The student will learn the key features and uses of Structured Streaming. The student will implement sliding windows to aggregate over chunks of data and apply watermarking to remove stale data. Finally, the student will connect to Event Hubs to read and write streams.

In this module, the student will be able to:

- Know the key features and uses of Structured Streaming
- Stream data from a file and write it out to a distributed file system
- Use sliding windows to aggregate over chunks of data rather than all data
- Apply watermarking to remove stale data
- Connect to Event Hubs read and write streams

17.1 Lab details

- Module 15 - Create a stream processing solution with Event Hubs and Azure Databricks
 - Lab details
 - Concepts
 - Event Hubs and Spark Structured Streaming
 - Streaming concepts
 - Lab
 - * Before the hands-on lab
 - Task 1: Create and configure the Azure Databricks workspace
 - * Exercise 1: Complete the Structured Streaming Concepts notebook
 - Task 1: Clone the Databricks archive
 - Task 2: Complete the notebook
 - * Exercise 2: Complete the Working with Time Windows notebook
 - * Exercise 3: Complete the Structured Streaming with Azure EventHubs notebook

17.2 Concepts

Apache Spark Structured Streaming is a fast, scalable, and fault-tolerant stream processing API. You can use it to perform analytics on your streaming data in near real time.

With Structured Streaming, you can use SQL queries to process streaming data in the same way that you would process static data. The API continuously increments and updates the final data.

17.3 Event Hubs and Spark Structured Streaming

Azure Event Hubs is a scalable real-time data ingestion service that processes millions of data in a matter of seconds. It can receive large amounts of data from multiple sources and stream the prepared data to Azure Data Lake or Azure Blob storage.

Azure Event Hubs can be integrated with Spark Structured Streaming to perform processing of messages in near real time. You can query and analyze the processed data as it comes by using a Structured Streaming query and Spark SQL.

17.4 Streaming concepts

Stream processing is where you continuously incorporate new data into Data Lake storage and compute results. The streaming data comes in faster than it can be consumed when using traditional batch-related processing techniques. A stream of data is treated as a table to which data is continuously appended. Examples of such data include bank card transactions, Internet of Things (IoT) device data, and video game play events.

A streaming system consists of:

- Input sources such as Kafka, Azure Event Hubs, IoT Hub, files on a distributed system, or TCP-IP sockets
- Stream processing using Structured Streaming, `foreach` sinks, memory sinks, etc.

17.5 Lab

You need to complete the exercises within Databricks Notebooks. To begin, you need to have access to an Azure Databricks workspace. If you do not have a workspace available, follow the instructions below. Otherwise, you can skip ahead to the `Clone the Databricks archive` step.

17.5.1 Before the hands-on lab

Note: Only complete the `Before the hands-on lab` steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 1.

Before stepping through the exercises in this lab, make sure you have access to an Azure Databricks workspace with an available cluster. Perform the tasks below to configure the workspace.

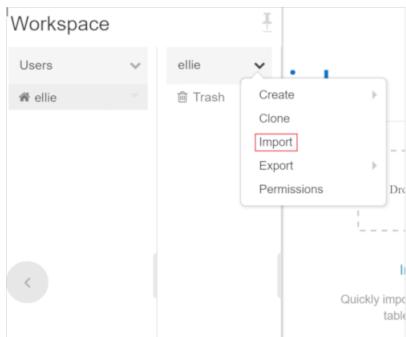
17.5.1.1 Task 1: Create and configure the Azure Databricks workspace

Follow the [lab 15 setup instructions](#) to create and configure the workspace.

17.5.2 Exercise 1: Complete the Structured Streaming Concepts notebook

17.5.2.1 Task 1: Clone the Databricks archive

1. If you do not currently have your Azure Databricks workspace open: in the Azure portal, navigate to your deployed Azure Databricks workspace and select **Launch Workspace**.
2. In the left pane, select **Workspace > Users**, and select your username (the entry with the house icon).
3. In the pane that appears, select the arrow next to your name, and select **Import**.



4. In the **Import Notebooks** dialog box, select the URL and paste in the following URL:

<https://github.com/solliancenet/microsoft-learning-paths-databricks-notebooks/blob/master/data-engineer>

1. Select **Import**.
2. Select the **10-Structured-Streaming** folder that appears.

17.5.2.2 Task 2: Complete the notebook

Open the **1.Structured-Streaming-Concepts** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will:

- Stream data from a file and write it out to a distributed file system
- List active streams
- Stop active streams

After you've completed the notebook, return to this screen, and continue to the next step.

17.5.3 Exercise 2: Complete the Working with Time Windows notebook

In your Azure Databricks workspace, open the **10-Structured-Streaming** folder that you imported within your user folder.

Open the **2.Time-Windows** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will:

- Use sliding windows to aggregate over chunks of data rather than all data
- Apply watermarking to throw away stale old data that you do not have space to keep
- Plot live graphs using `display`

After you've completed the notebook, return to this screen, and continue to the next step.

17.5.4 Exercise 3: Complete the Structured Streaming with Azure EventHubs notebook

In your Azure Databricks workspace, open the **10-Structured-Streaming** folder that you imported within your user folder.

Open the **3.Streaming-With-Event-Hubs-Demo** notebook. Make sure you attach your cluster to the notebook before following the instructions and running the cells within.

Within the notebook, you will:

- Connect to Event Hubs and write a stream to your event hub
- Read a stream from your event hub
- Define a schema for the JSON payload and parse the data do display it within a table

18 Module 16 - Build reports using Power BI integration with Azure Synapse Analytics

In this module, the student will learn how to integrate Power BI with their Synapse workspace to build reports in Power BI. The student will create a new datasource and Power BI report in Synapse Studio. Then the student will learn how to improve query performance with materialized views and result-set caching. Finally, the student will explore the data lake with serverless SQL pools and create visualizations against that data in Power BI.

In this module, the student will be able to:

- Integrate a Synapse workspace and Power BI
- Optimize integration with Power BI
- Improve query performance with materialized views and result-set caching
- Visualize data with SQL serverless and create a Power BI report

18.1 Lab details

- Module 16 - Build reports using Power BI integration with Azure Synapse Analytics
 - Lab details
 - Resource naming throughout this lab
 - Lab setup and pre-requisites
 - Exercise 0: Start the dedicated SQL pool
 - Exercise 1: Power BI and Synapse workspace integration
 - * Task 1: Login to Power BI
 - * Task 2: Create a Power BI workspace
 - * Task 3: Connect to Power BI from Synapse
 - * Task 4: Explore the Power BI linked service in Synapse Studio
 - * Task 5: Create a new datasource to use in Power BI Desktop
 - * Task 6: Create a new Power BI report in Synapse Studio
 - Exercise 2: Optimizing integration with Power BI
 - * Task 1: Explore Power BI optimization options
 - * Task 2: Improve performance with materialized views
 - * Task 3: Improve performance with result-set caching
 - Exercise 3: Visualize data with SQL Serverless
 - * Task 1: Explore the data lake with SQL Serverless
 - * Task 2: Visualize data with SQL serverless and create a Power BI report
 - Exercise 4: Cleanup
 - * Task 1: Pause the dedicated SQL pool

18.2 Resource naming throughout this lab

For the remainder of this guide, the following terms will be used for various ASA-related resources (make sure you replace them with actual names and values):

Azure Synapse Analytics Resource	To be referred to
Workspace / workspace name	Workspace
Power BI workspace name	Synapse 01

Azure Synapse Analytics Resource	To be referred to
SQL Pool	SqlPool101
Lab schema name	pbi

18.3 Lab setup and pre-requisites

Note: Only complete the **Lab setup and pre-requisites** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Install Power BI Desktop on your lab computer or VM.

Complete the lab setup instructions for this module.

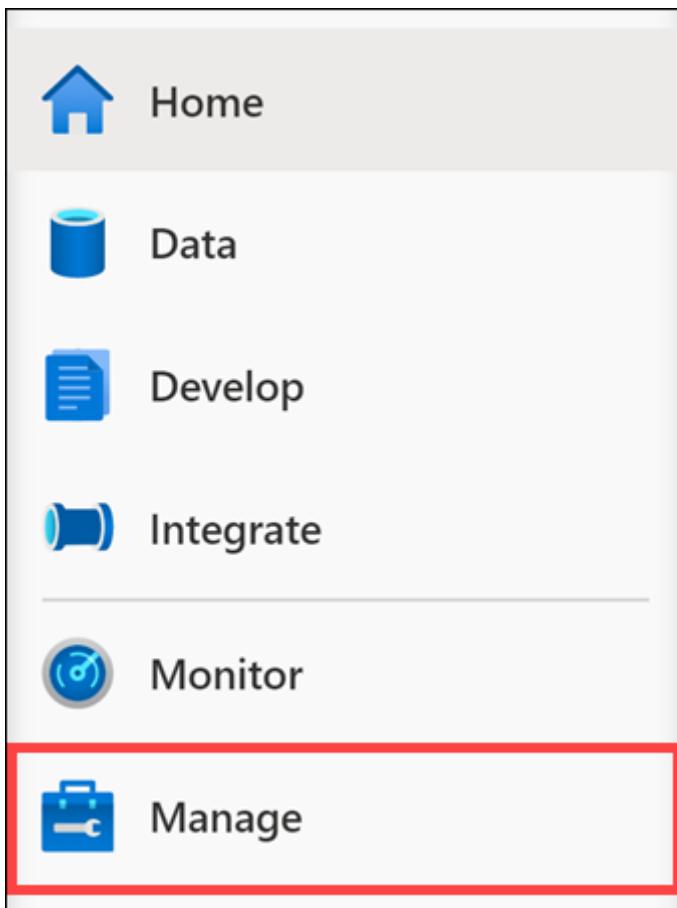
Note, the following modules share this same environment:

- [Module 4](#)
- [Module 5](#)
- [Module 7](#)
- [Module 8](#)
- [Module 9](#)
- [Module 10](#)
- [Module 11](#)
- [Module 12](#)
- [Module 13](#)
- [Module 16](#)

18.4 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



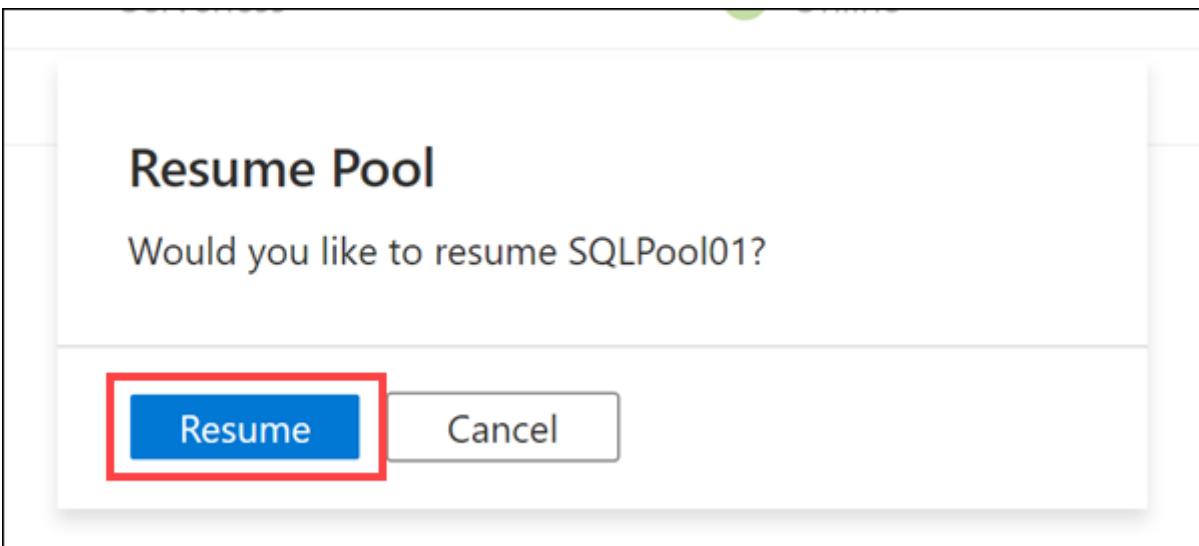
3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the 'SQL pools' blade in the Azure Data Studio interface. On the left, there is a sidebar with the following options: Analytics pools, SQL pools (highlighted with a red box and a red number '1'), Apache Spark pools, External connections, Linked services, Azure Purview (Preview), Integration, Triggers, Integration runtimes, Security, and Access control. The main area displays the 'SQL pools' section with the following details:

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused (highlighted with a red box and a red number '2'))	DW100c

At the bottom of the table, there is a 'Resume' button.

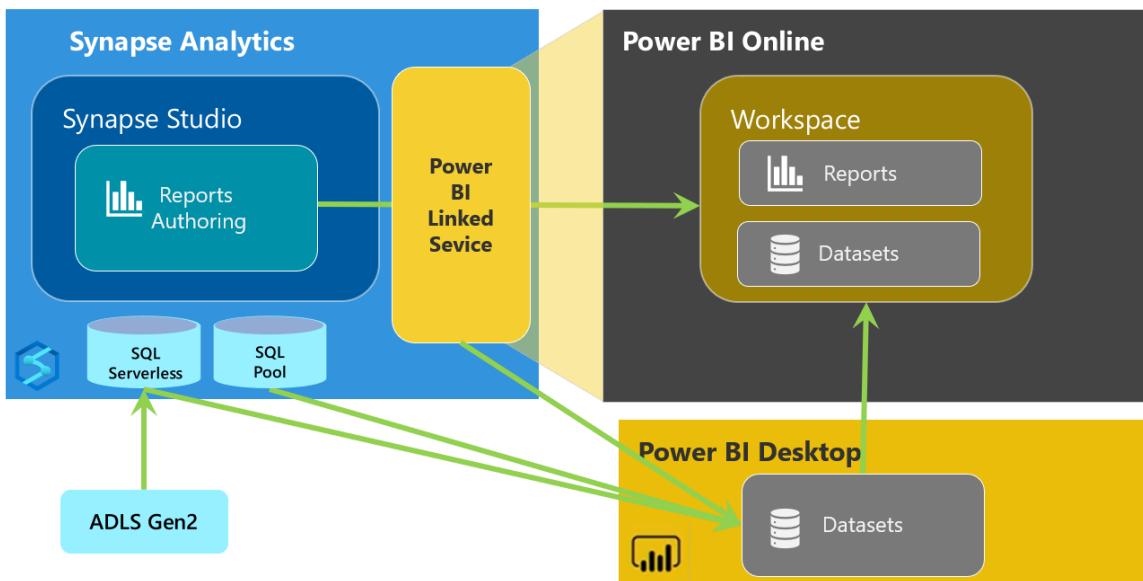
4. When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

18.5 Exercise 1: Power BI and Synapse workspace integration

Synapse Analytics and Power BI integration

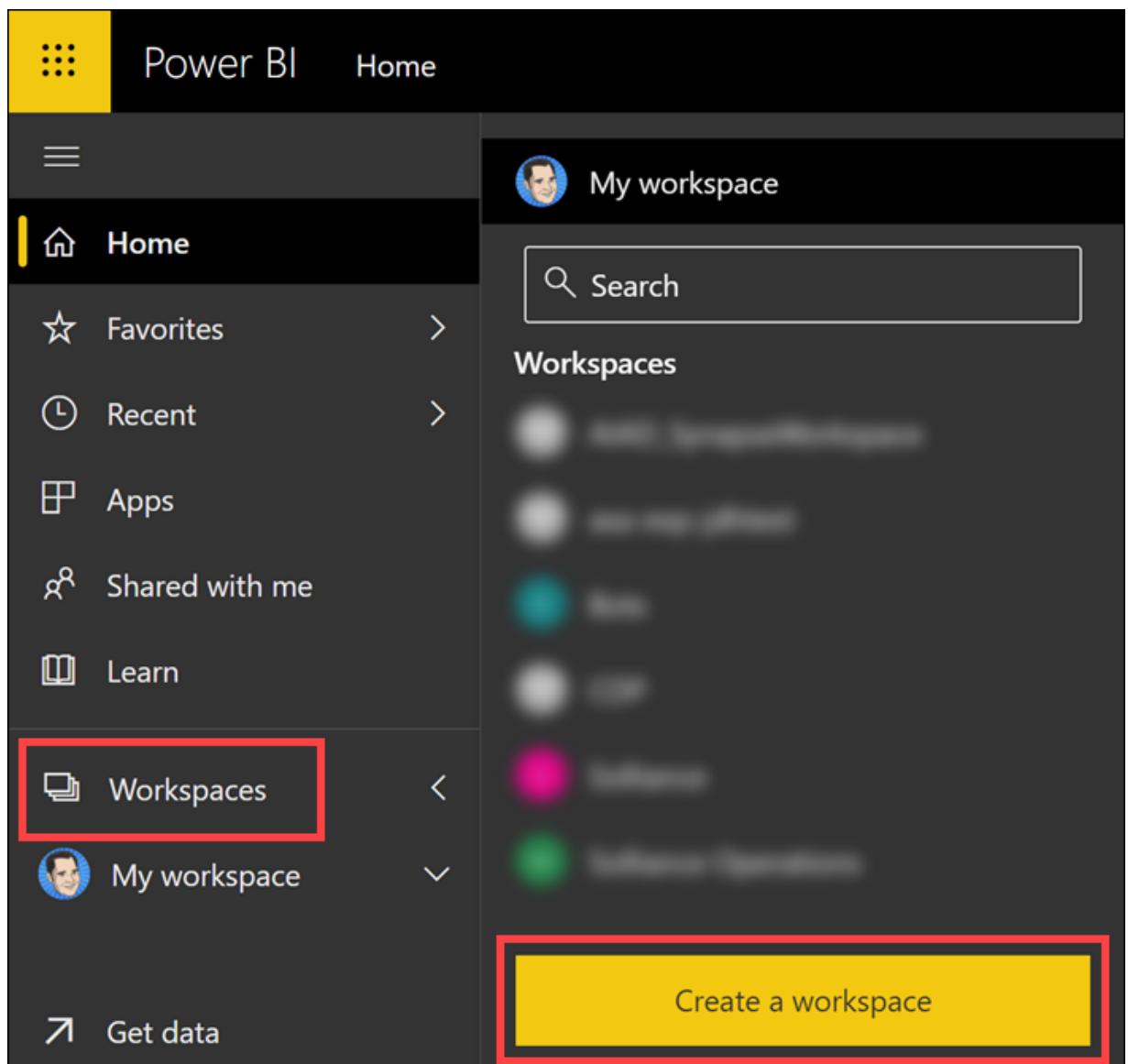


18.5.1 Task 1: Login to Power BI

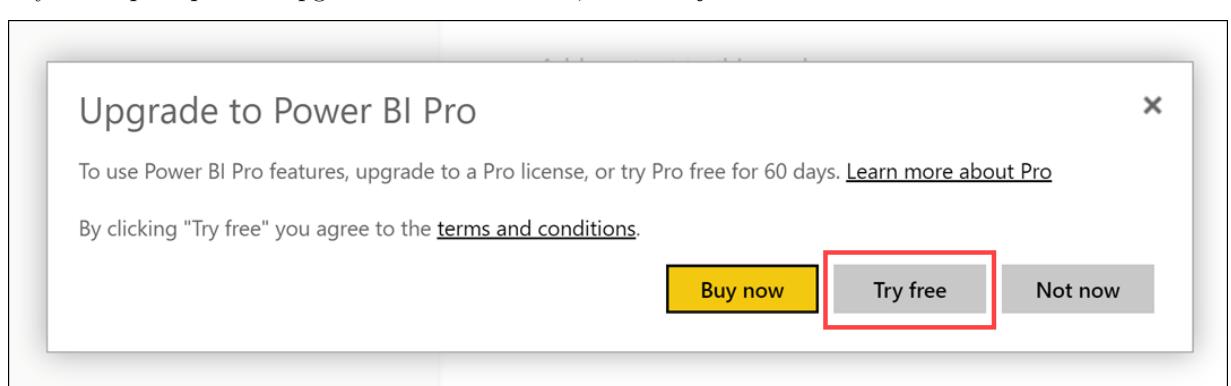
1. In a new browser tab, navigate to <https://powerbi.microsoft.com/>.
2. Sign in with the same account used to sign in to Azure by selecting the **Sign in** link on the upper-right corner.
3. If this is your first time signing into this account, complete the setup wizard with the default options.

18.5.2 Task 2: Create a Power BI workspace

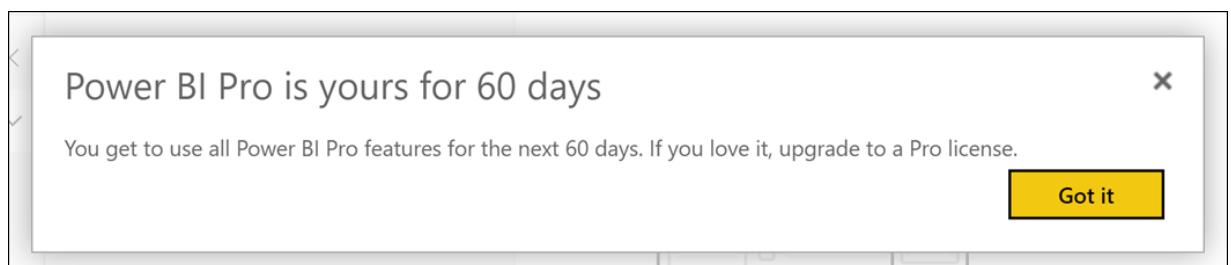
1. Select **Workspaces**, then select **Create a workspace**.



2. If you are prompted to upgrade to Power BI Pro, select **Try free**.



Select **Got it** to confirm the pro subscription.



- Set the name to **synapse-training**, then select **Save**. If you receive a message that **synapse-training** is not available, append your initials or other characters to make the workspace name unique in your organization.

Create a workspace

YOU'RE CREATING AN UPGRADED WORKSPACE
Enjoy new features, better sharing options, and improved security controls.
[Revert to classic](#) | [Learn more](#)

Workspace image

 Upload
Delete

Workspace name

Available

Description

Describe this workspace

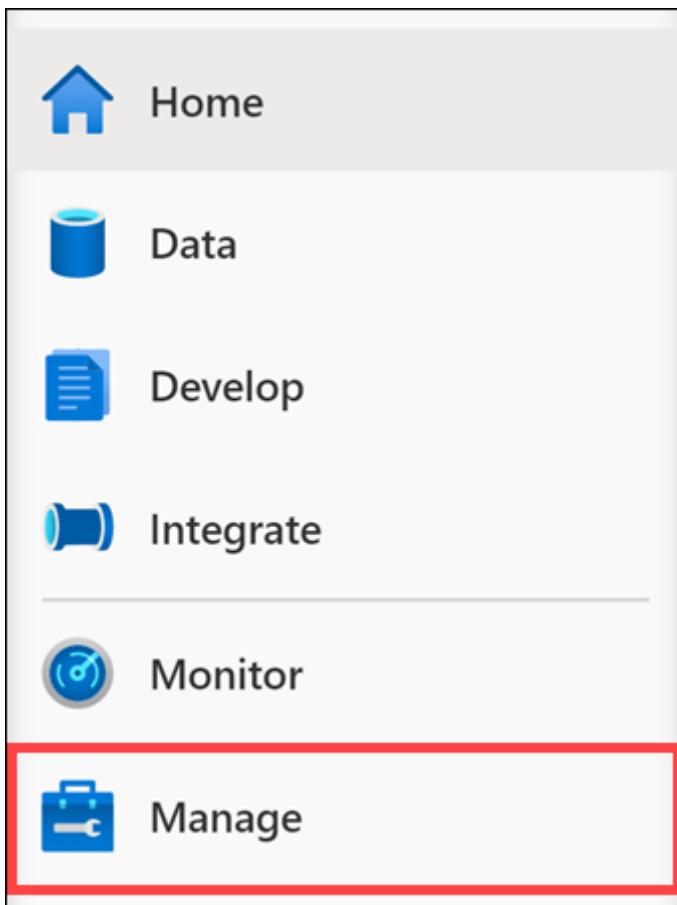
[Learn more about workspace settings](#)

Advanced ▾

Save Cancel

18.5.3 Task 3: Connect to Power BI from Synapse

- Open Synapse Studio (<https://web.azuresynapse.net/>), and then navigate to the **Manage hub**.

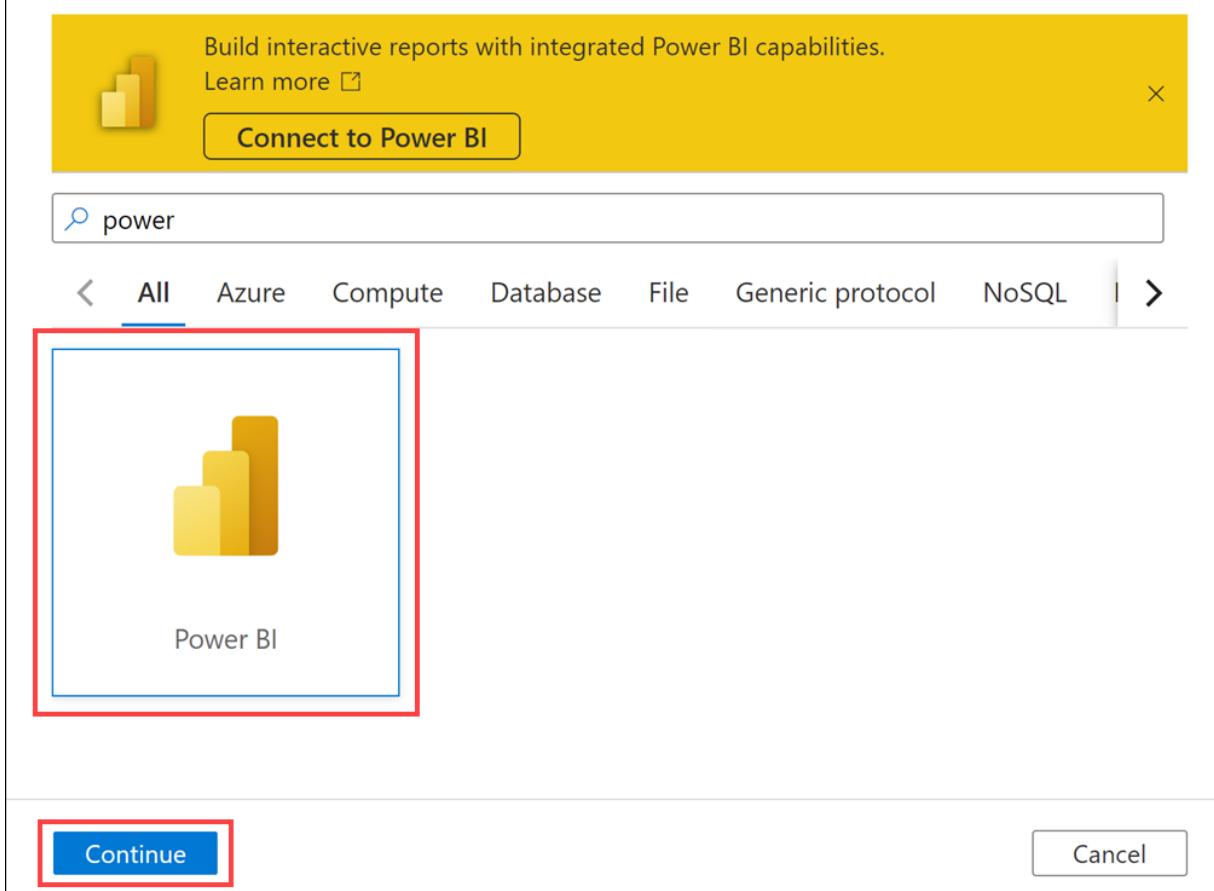


2. Select **Linked services** on the left-hand menu, then select **+ New**.

The screenshot shows the 'Linked services' page. On the left, there is a sidebar with categories: Analytics pools, External connections (which has 'Linked services' selected and highlighted with a red box), Integration, and Security. On the right, the main area is titled 'Linked services' and contains the following text: 'Linked services are much like connection string'. Below this is a button labeled '+ New' which is also highlighted with a red box. There is a 'Filter by keyword' input field and a partially visible 'Ann' button. At the bottom, it says 'Showing 1 - 2 of 2 items' and lists two entries: 'synapselabretailjdhasws-Workspace...' and 'synapselabretailjdhasws-Workspace...'. The first entry has a blue hexagonal icon next to it.

3. Select **Power BI**, then select **Continue**.

New linked service



4. In the dataset properties form, complete the following:

Field	Value
Name	enter <code>handson_powerbi</code>
Workspace name	select <code>synapse-training</code>

New linked service (Power BI)

i Choose a name for your linked service. This name cannot be updated later.

Name *

handson_powerbi

Description

Tenant

Workspace name *

synapse-training (

Edit

Annotations

+ New

Name

▷ Advanced ⓘ

Create

Back

Cancel

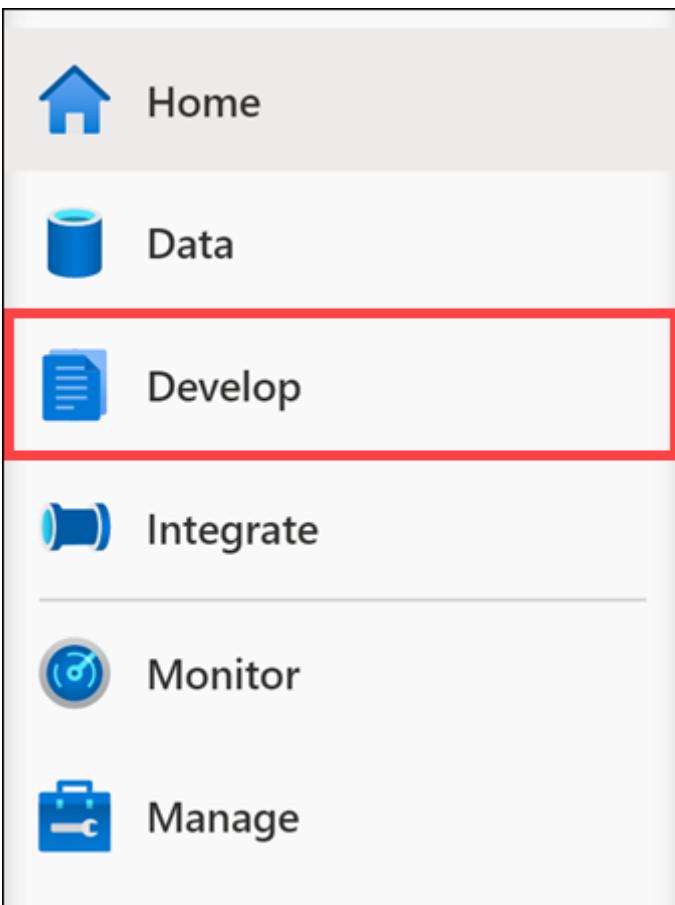
5. Select **Create**.

6. Select **Publish all**, then **Publish**.



18.5.4 Task 4: Explore the Power BI linked service in Synapse Studio

1. In **Azure Synapse Studio** and navigate to the **Develop** hub using the left menu option.



2. Expand Power BI, expand `handson_powerbi` and observe that you have access to your Power BI datasets and reports, directly from Synapse Studio.

The screenshot shows the 'Develop' tab in the Power BI service. At the top right are navigation icons: a blue plus sign, a downward arrow, and a double-left arrow. Below them is a search bar with the placeholder 'Filter resources by name'. The main list contains the following items:

- SQL scripts (12)
- Notebooks (5)
- Data flows (2)
- Power BI (1)**
 - handson_powerbi**
 - Power BI datasets
 - Power BI reports

A red box highlights the 'Power BI' category and its sub-item 'handson_powerbi'.

New reports can be created by selecting + at the top of the **Develop** tab. Existing reports can be edited by selecting the report name. Any saved changes will be written back to the Power BI workspace.

18.5.5 Task 5: Create a new datasource to use in Power BI Desktop

1. Beneath **Power BI**, under the linked Power BI workspace, select **Power BI datasets**.
2. Select **New Power BI dataset** from the top actions menu.

The screenshot shows the 'Develop' tab in the Power BI service. The left pane lists resources, and the right pane shows a detailed view of the 'Power BI datasets' section for the 'handson_powerbi' workspace.

Actions:

- + New Power BI dataset
- Refresh

Power BI datasets (handson_powerbi)

This is a read-only view of datasets existing in handson_powerbi

Showing 0 item

Name

A red box highlights the 'Power BI datasets' item in the left list, and another red box highlights the '+ New Power BI dataset' button in the top right actions area.

3. Select **Start** and make sure you have Power BI Desktop installed on your environment machine.

The diagram illustrates the process of creating a Power BI dataset. It consists of four circular nodes connected by arrows. The first node contains a blue cylinder icon representing a data source. The second node contains a blue document icon with a downward arrow, representing the download or import of data. The third node contains a yellow laptop icon with a bar chart, representing the Power BI desktop application. The fourth node is a portrait of a woman with long dark hair sitting at a desk, working on a laptop, representing the user building reports.

Let's get started with Microsoft Power BI

Create a Power BI dataset from a data source and publish it to Power BI to build reports in Azure Synapse Studio.

To begin, you'll need Power BI Desktop installed on your local machine

[Install Power BI Desktop](#)

[View documentation](#)

Start **Cancel**

4. Select **SQLPool01**, then select **Continue**.



Select a data source

Select a SQL pool below to use as a data source. You'll be able to select tables from this pool when creating your dataset.

Name

 SQLPool01

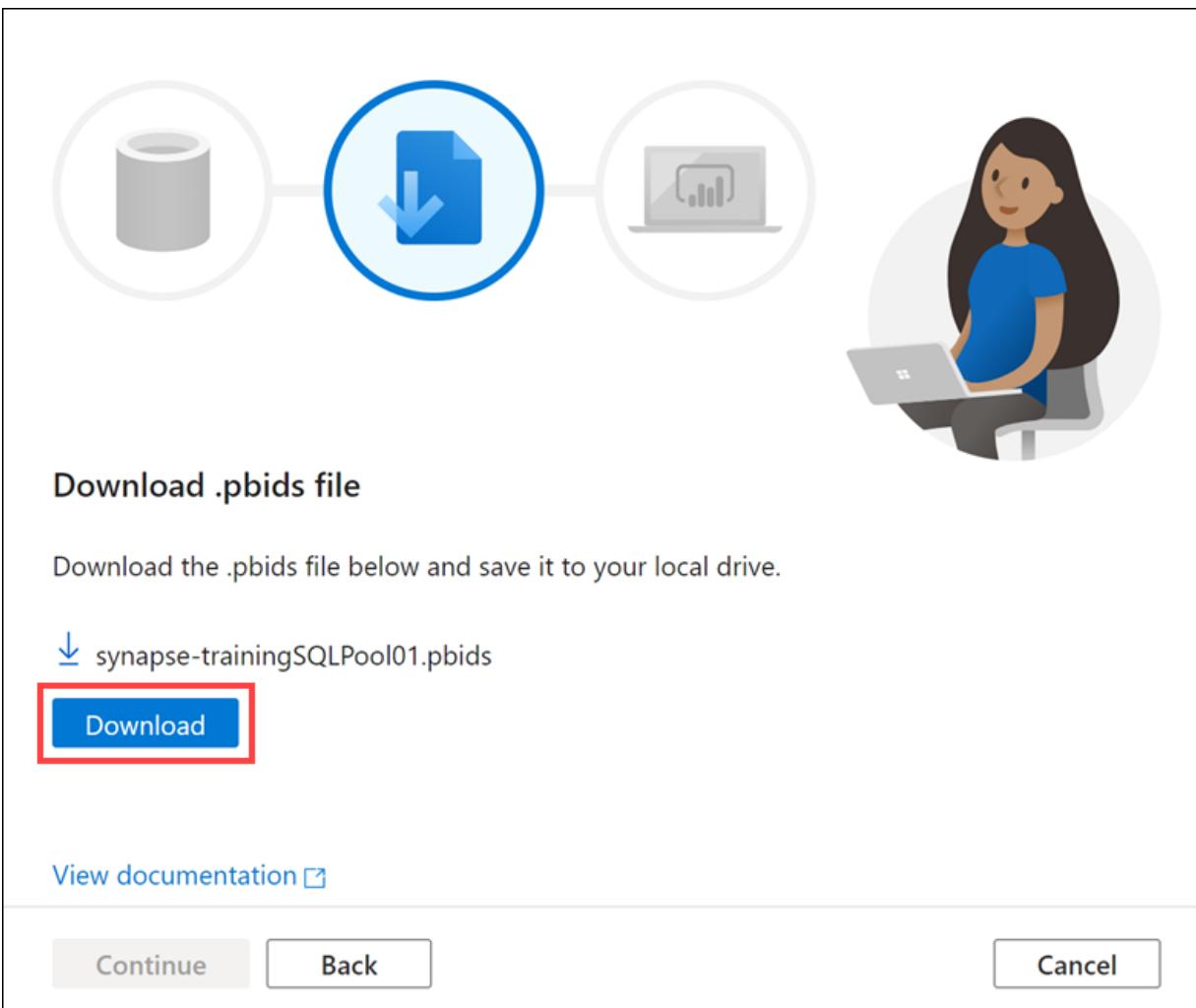
[View documentation](#) 

Continue

Back

Cancel

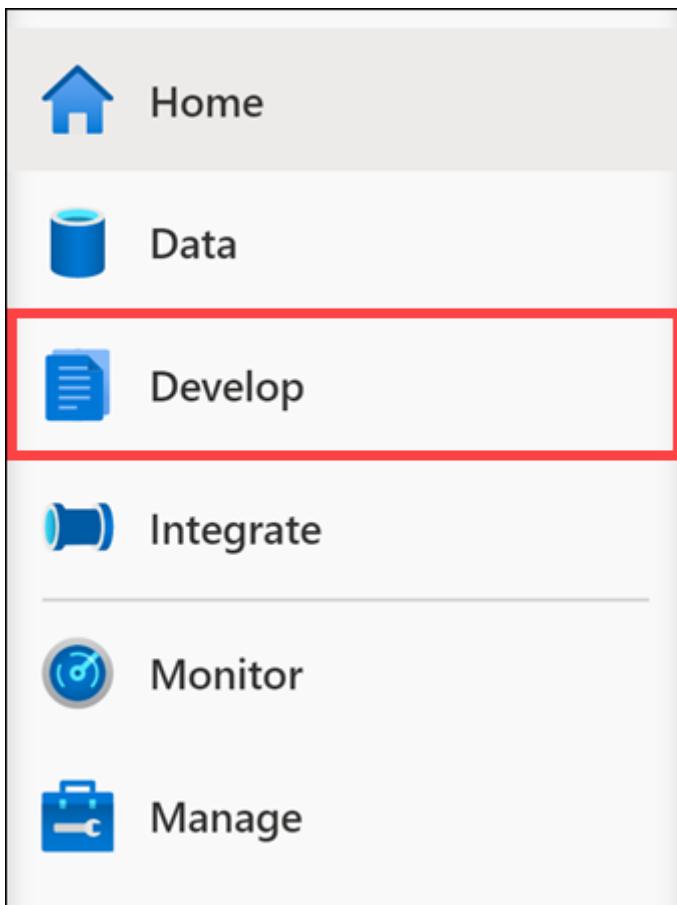
5. Next, select **Download** to download the .pbids file.



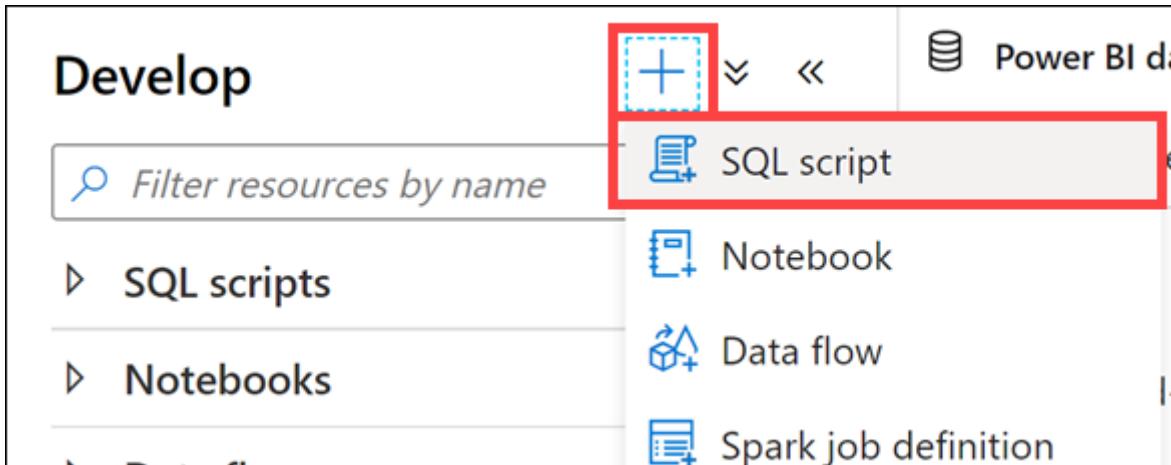
6. Select **Continue**, then **Close and refresh** to close the publishing dialog.

18.5.6 Task 6: Create a new Power BI report in Synapse Studio

1. In **Azure Synapse Studio**, select **Develop** from the left menu.



2. Select +, then SQL script.



3. Connect to **SQLPool01**, then execute the following query to get an approximation of its execution time (may be around 1 minute). This will be the query we'll use to bring data in the Power BI report you'll build later in this exercise.

```
SELECT count(*) FROM
(
    SELECT
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
        ,avg(FS.TotalAmount) as AvgTotalAmount
        ,avg(FS.ProfitAmount) as AvgProfitAmount
        ,sum(FS.TotalAmount) as TotalAmount
)
```

```

        ,sum(FS.ProfitAmount) as ProfitAmount
    FROM
        wwi.SaleSmall FS
        JOIN wwi.Product P ON P.ProductId = FS.ProductId
        JOIN wwi.Date D ON FS.TransactionDateId = D.DateId
    GROUP BY
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
    ) T

```

You should see a query result of 194683820.

The screenshot shows a SQL query editor interface. At the top, there are several buttons: Run (highlighted with a red box and number 2), Undo, Publish, Query plan, Connect to (highlighted with a red box and number 1), Use database, and a refresh icon. The 'Connect to' dropdown is set to 'SQLPool01'. Below the toolbar, the SQL code is displayed:

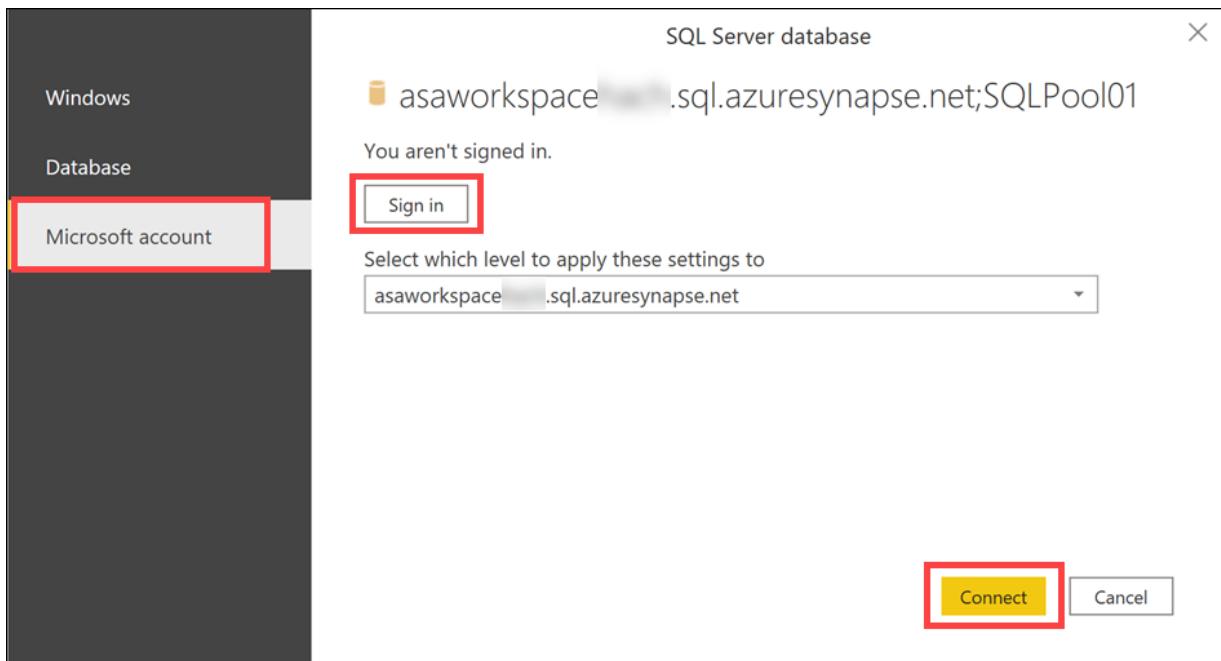
```

1 SELECT count(*) FROM
2 (
3     SELECT
4         FS.CustomerID
5         ,P.Seasonality
6         ,D.Year
7         ,D.Quarter
8         ,D.Month
9         ,avg(FS.TotalAmount) as AvgTotalAmount
10        ,avg(FS.ProfitAmount) as AvgProfitAmount
11        ,sum(FS.TotalAmount) as TotalAmount
12        ,sum(FS.ProfitAmount) as ProfitAmount
13    FROM
14        wwi.SaleSmall FS
15        JOIN wwi.Product P ON P.ProductId = FS.ProductId
16        JOIN wwi.Date D ON FS.TransactionDateId = D.DateId

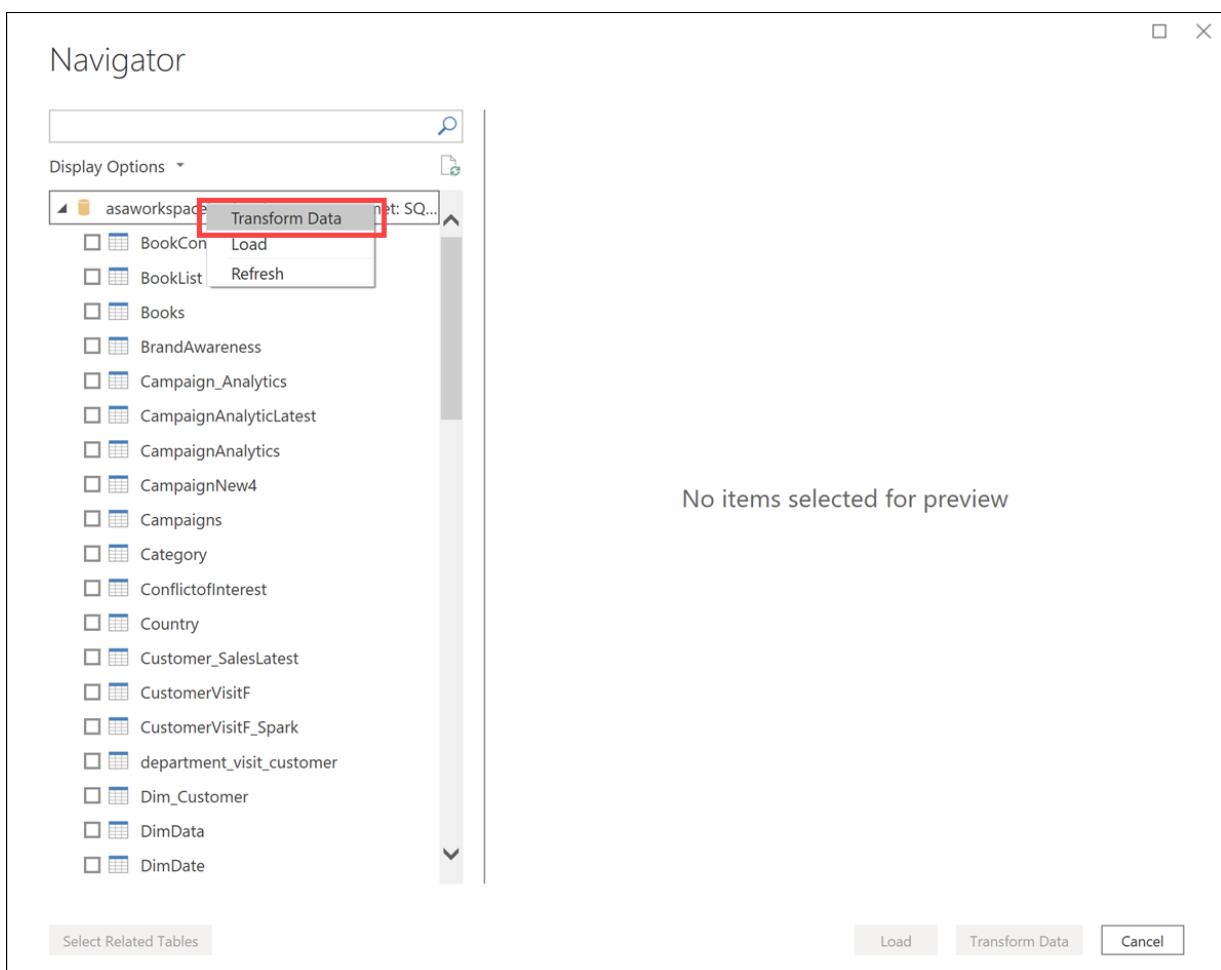
```

Below the code, there are tabs for 'Results' (selected) and 'Messages'. Under 'View', the 'Table' option is selected. The results section shows a single row with the column '(No Column Name)' containing the value '194683820' (highlighted with a red box and number 3).

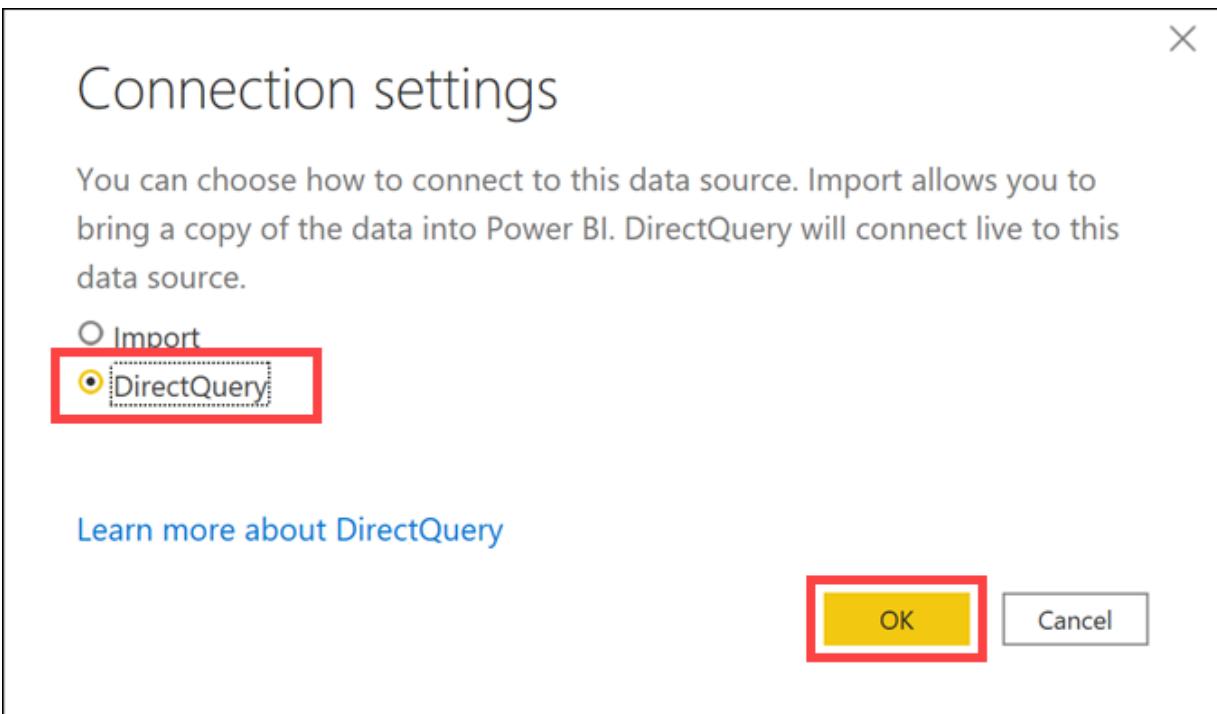
4. To connect to your datasource, open the downloaded .pbids file in Power BI Desktop. Select the **Microsoft account** option on the left, **Sign in** (with the same credentials you use for connecting to the Synapse workspace) and click **Connect**.



5. In the Navigator dialog, right-click on the root database node and select **Transform data**.



6. Select the **DirectQuery** option in the connection settings dialog, since our intention is not to bring a copy of the data into Power BI, but to be able to query the data source while working with the report visualizations. Click **OK** and wait a few seconds while the connection is configured.



7. In the Power Query editor, open the settings page of the **Source** step in the query. Expand the **Advanced options** section, paste the following query and click **OK**.

SQL Server database

Server: asaworkspacechach.sql.azuresynapse.net
Database (optional): SQLPool01

Advanced options

Command timeout in minutes (optional):

SQL statement (optional, requires database):

```
SELECT * FROM
(
    SELECT
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
        ,avg(FS.TotalAmount) as AvgTotalAmount
        ,avg(FS.ProfitAmount) as AvgProfitAmount
        ,sum(FS.TotalAmount) as TotalAmount
        ,sum(FS.ProfitAmount) as ProfitAmount
)
```

Include relationship columns
 Navigate using full hierarchy
 Enable SQL Server Failover support

OK Cancel

```

SELECT * FROM
(
    SELECT
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
        ,avg(FS.TotalAmount) as AvgTotalAmount
        ,avg(FS.ProfitAmount) as AvgProfitAmount
        ,sum(FS.TotalAmount) as TotalAmount
        ,sum(FS.ProfitAmount) as ProfitAmount
)
FROM
    wwi.SaleSmall FS
    JOIN wwi.Product P ON P.ProductId = FS.ProductId

```

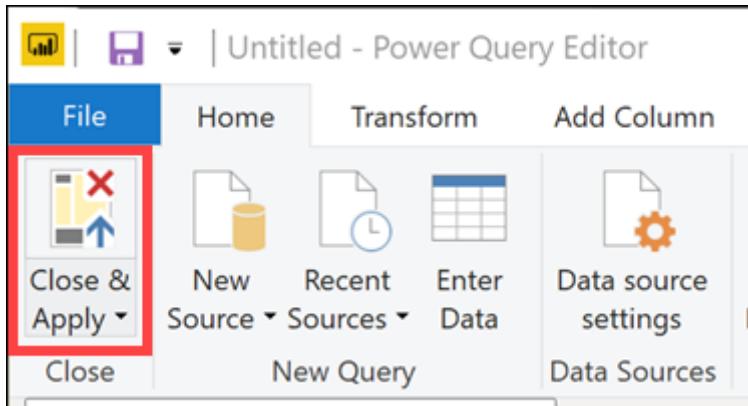
```

        JOIN wwi.Date D ON FS.TransactionDateId = D.DateId
GROUP BY
    FS.CustomerID
    ,P.Seasonality
    ,D.Year
    ,D.Quarter
    ,D.Month
) T

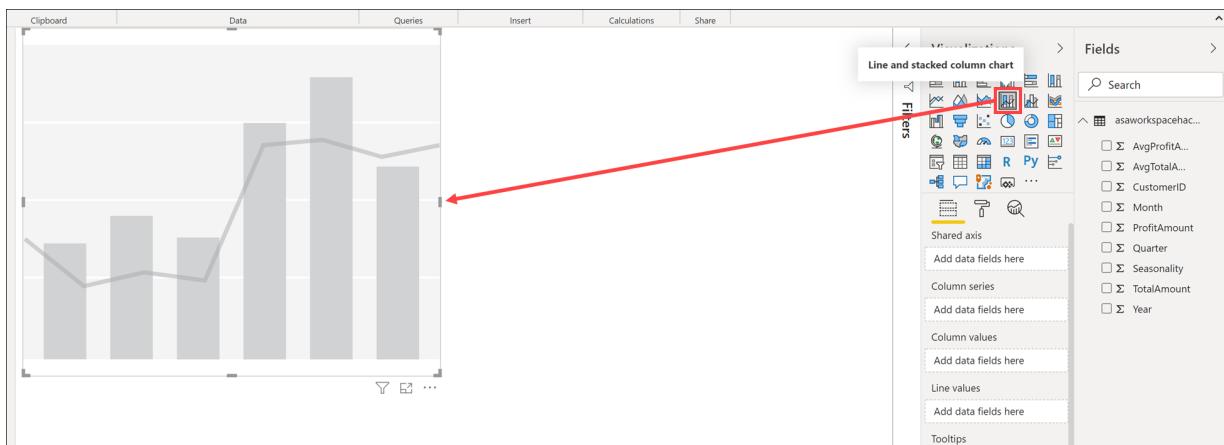
```

Note that this step will take at least 40-60 seconds to execute since it submits the query directly on the Synapse SQL Pool connection.

- Select **Close & Apply** on the topmost left corner of the editor window to apply the query and fetch the initial schema in the Power BI designer window.

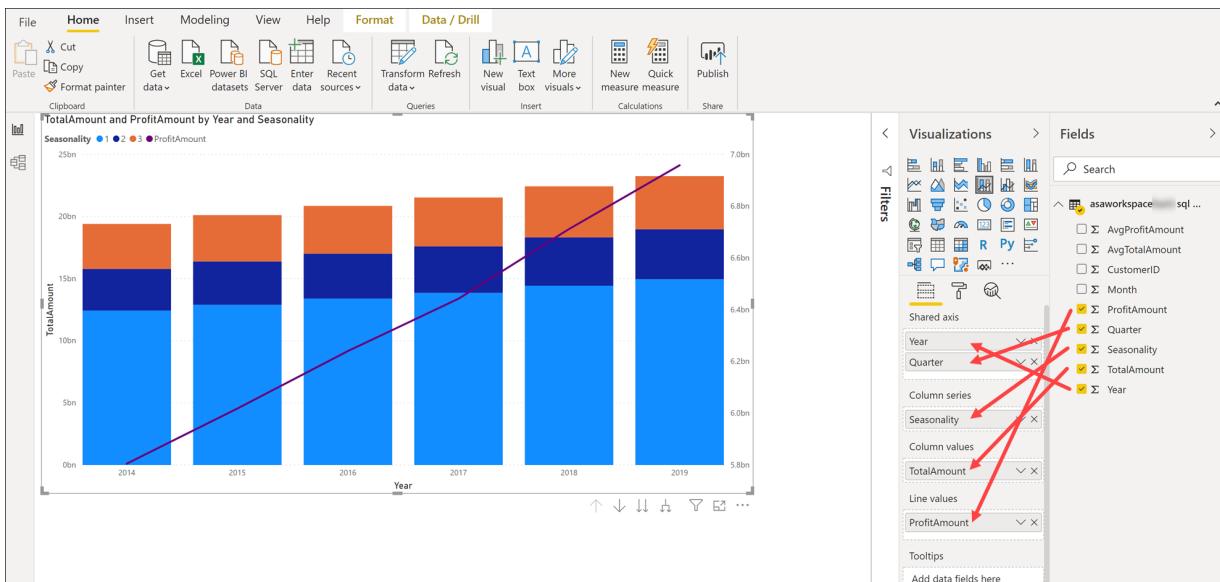


- Back to the Power BI report editor, expand the **Visualizations** menu on the right, then select the **Line and stacked column chart** visualization.



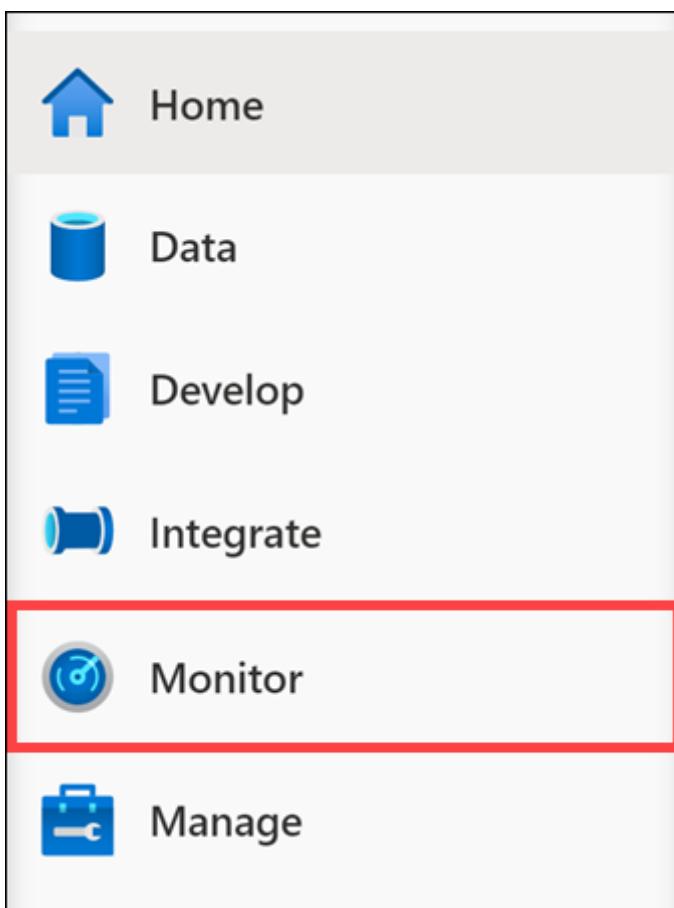
- Select the newly created chart to expand its properties pane. Using the expanded **Fields** menu, configure the visualization as follows:

- Shared axis:** Year, Quarter
- Column series:** Seasonality
- Column values:** TotalAmount
- Line values:** ProfitAmount



It will take around 40-60 seconds for the visualization to render, due to the live query execution on the Synapse dedicated SQL pool.

11. You can check the query executed while configuring the visualization in the Power BI Desktop application. Switch back to Synapse Studio, then select the **Monitor** hub from the left-hand menu.



12. Under the **Activities** section, open the **SQL requests** monitor. Make sure you select **SQLPool01** in the Pool filter, as by default Built-In is selected.

SQL requests					
Integration					
Activities					
Apache Spark applications					
SQL requests					
Data flow debug					
Analytics pools					
SQL pools					
Apache Spark pools					
Request ID ↑	Request content ↑	Submit time ↑	Duration	Submitter ↑	
QID6547	SELECT TOP (... More	2023-10-09T10:00:00Z	21 sec	odl_user_356	
QID6546	SELECT TOP (... More	2023-10-09T10:00:00Z	21 sec	odl_user_356	
QID6545	USE [SQLPool... More	2023-10-09T10:00:00Z	0s	odl_user_356	
QID6543	SELECT TOP (... More	2023-10-09T10:00:00Z	6 sec	odl_user_356	
QID6542	USE [SQLPool... More	2023-10-09T10:00:00Z	0s	odl_user_356	
QID6541	SELECT TOP (... More	2023-10-09T10:00:00Z	15 sec	odl_user_356	

13. Identify the query behind your visualization in the topmost requests you see in the log and observe the duration which is about 20-30 seconds. Select **More** on a request to look into the actual query submitted from Power BI Desktop.

Showing 1 - 100 of 138 items				
Request ID ↑↓	Request content ↑↓	Submit time ↑↓	Duration	
QID6547	SELECT TOP (... More	2023-10-09T10:00:00Z	21 sec	
QID6546	SELECT TOP (... More	2023-10-09T10:00:00Z	21 sec	
QID6545	USE [SQLPool... More	2023-10-09T10:00:00Z	0s	
QID6543	SELECT TOP (... More	2023-10-09T10:00:00Z	6 sec	

Request content

QID6547

Pool :

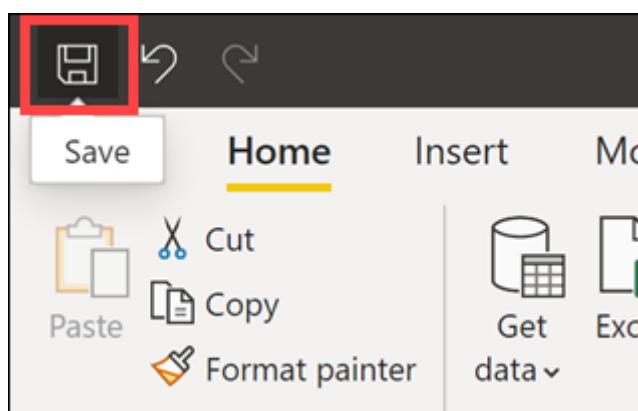
Duratio
21 sec
21 sec
0s
6 sec
0s
15 sec
0s
13 sec
0s
3 sec
0s
0s
0s
1 min
0s

```

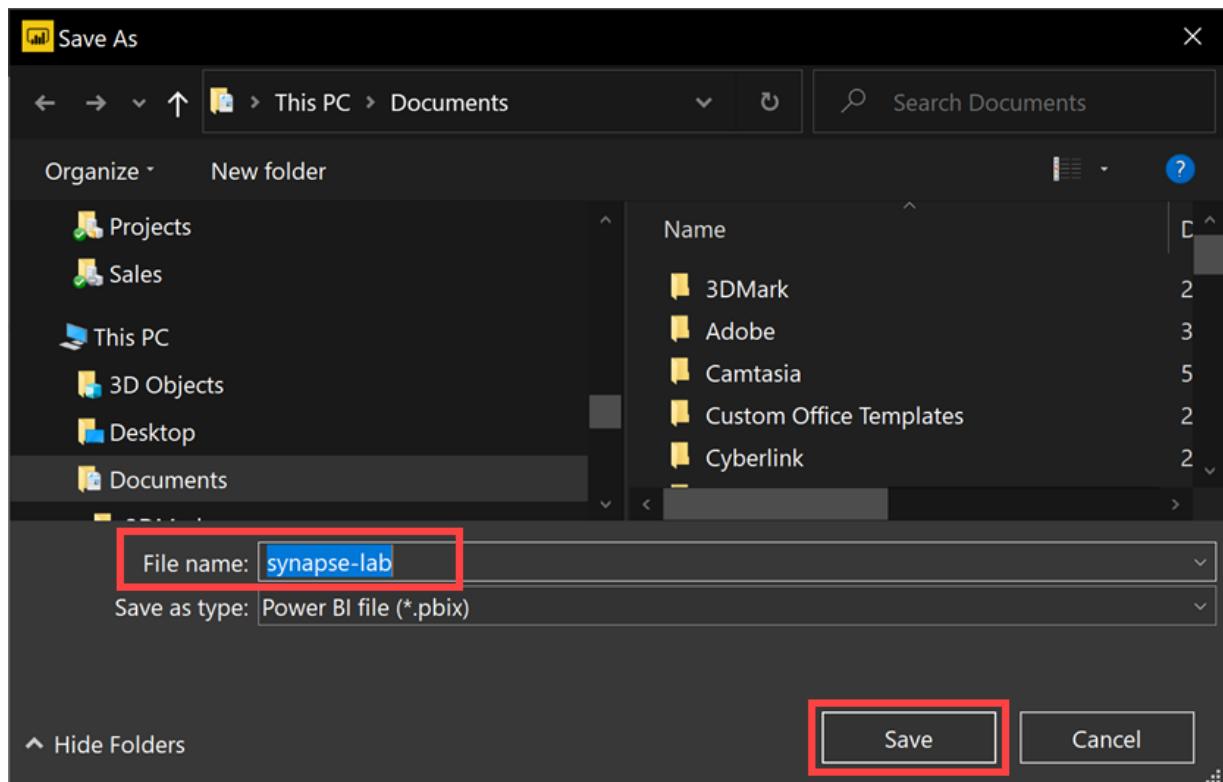
SELECT
TOP (10000001) [t0].[Year],SUM([t0].[ProfitAmount])
AS [a0]
FROM
(
(
SELECT * FROM
(
SELECT
    FS.CustomerID
    ,P.Seasonality
    ,D.Year
    ,D.Quarter
    ,D.Month
    ,avg(FS.TotalAmount) as AvgTotalAmount
    ,avg(FS.ProfitAmount) as AvgProfitAmount
    ,sum(FS.TotalAmount) as TotalAmount
    ,sum(FS.ProfitAmount) as ProfitAmount
FROM
    wwi.SaleSmall FS
    JOIN wwi.Product P ON P.ProductId = FS.ProductId
    JOIN wwi.Date D ON FS.TransactionDateId = D.DateId
GROUP BY
    FS.CustomerID
    ,P.Seasonality
    ,D.Year
    ,D.Quarter
    ,D.Month
) T
)
)
```

Close

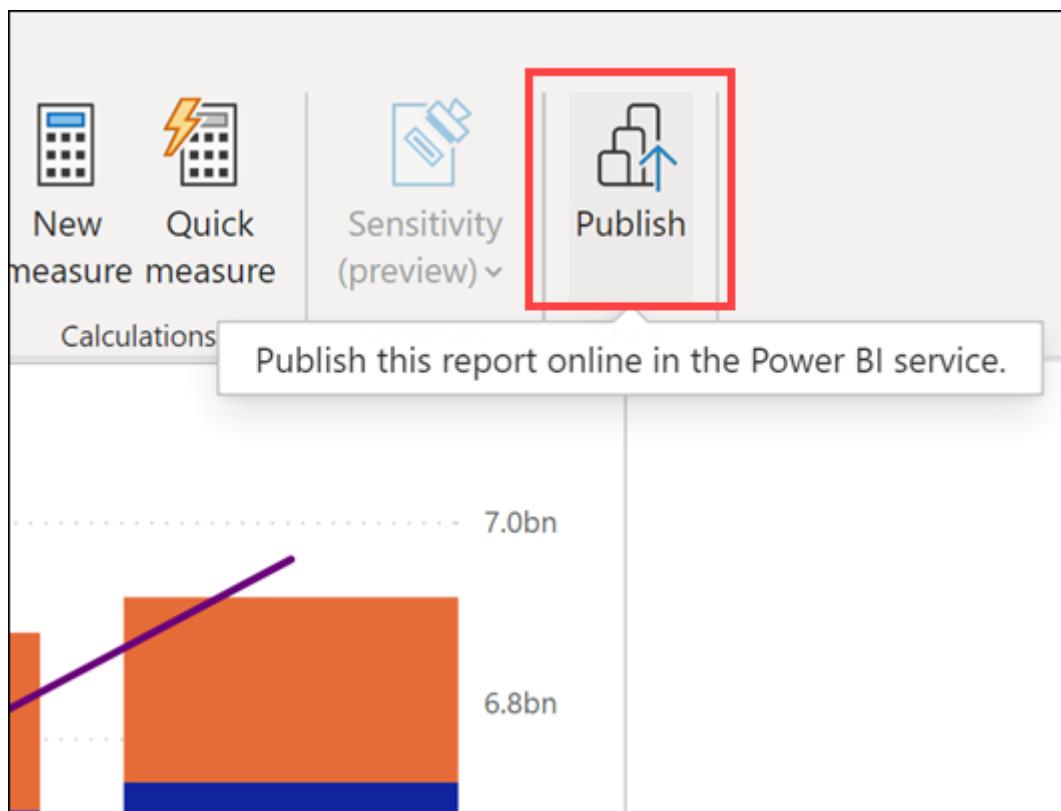
14. Switch back to the Power BI Desktop application, then click **Save** in the top-left corner.



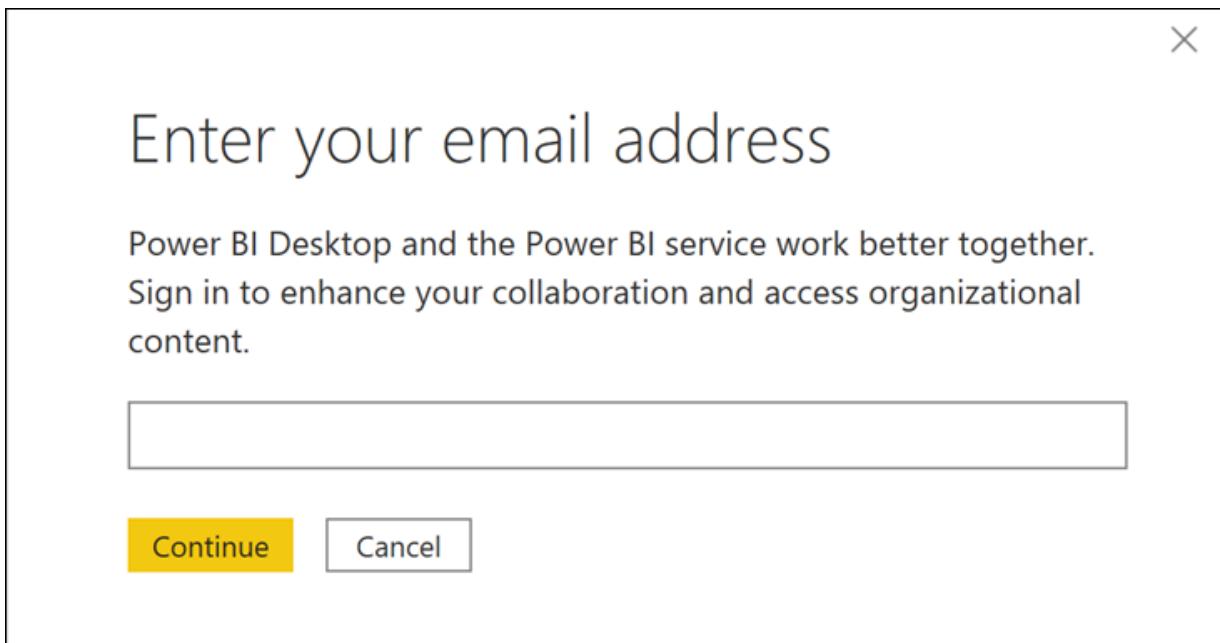
15. Specify a file name, such as **synapse-lab**, then click **Save**.



16. Click **Publish** above the saved report. Make sure that, in Power BI Desktop, you are signed in with the same account you use in the Power BI portal and in Synapse Studio. You can switch to the proper account from the right topmost corner of the window.

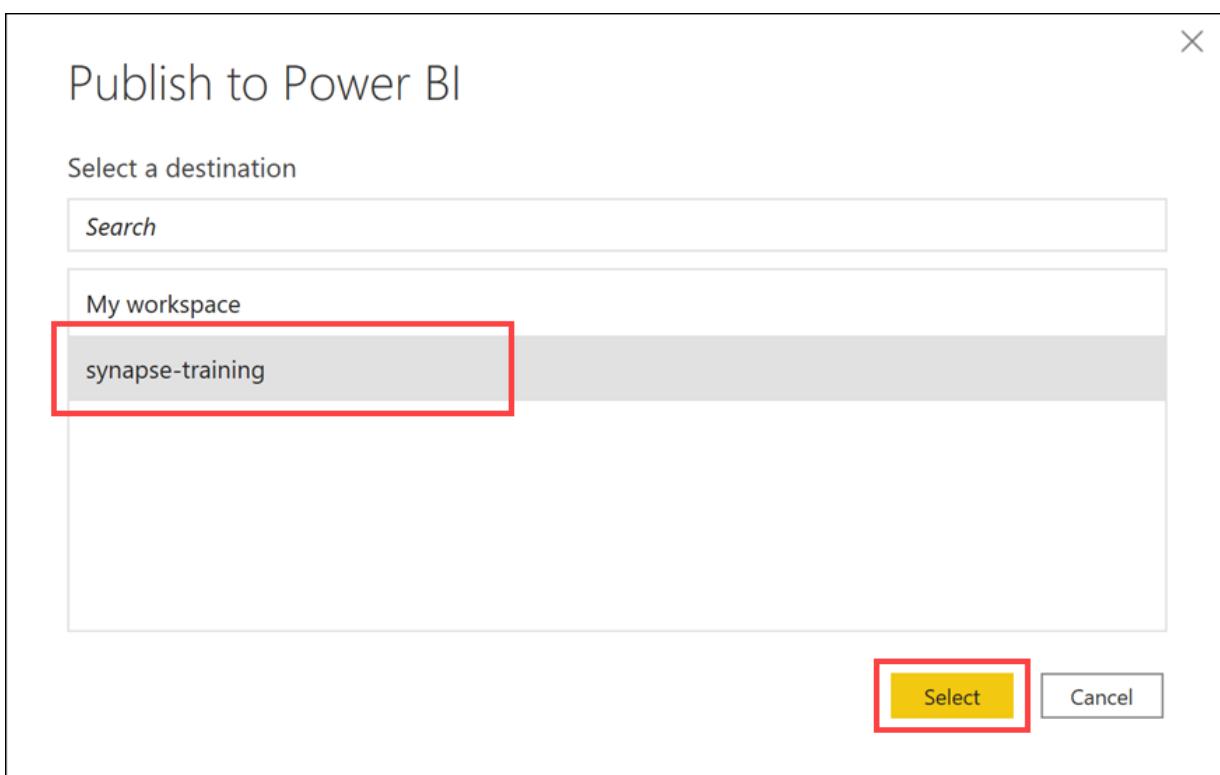


If you are not currently signed in to Power BI desktop, you will be prompted to enter your email address. Use the account credentials you are using for connecting to the Azure portal and Synapse Studio in this lab.



Follow the prompts to complete signing in to your account.

17. In the **Publish to Power BI** dialog, select the workspace you linked to Synapse (for example, **synapse-training**), then click **Select**.



18. Wait until the publish operation successfully completes.

Publishing to Power BI

✓ Success!

[Open 'synapse-lab.pbix' in Power BI](#)



Did you know?

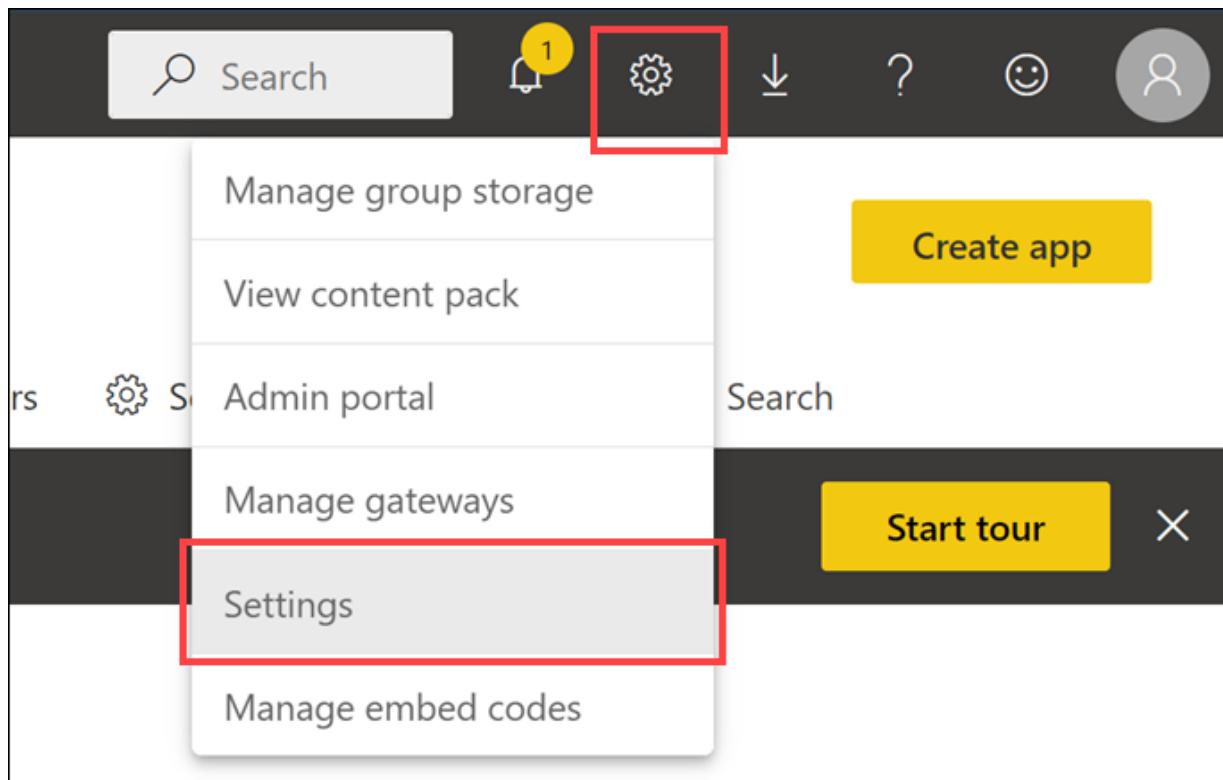
You can create a portrait view of your report, tailored for mobile phones.
On the **View** tab, select **Mobile Layout**. [Learn more](#)

[Got it](#)

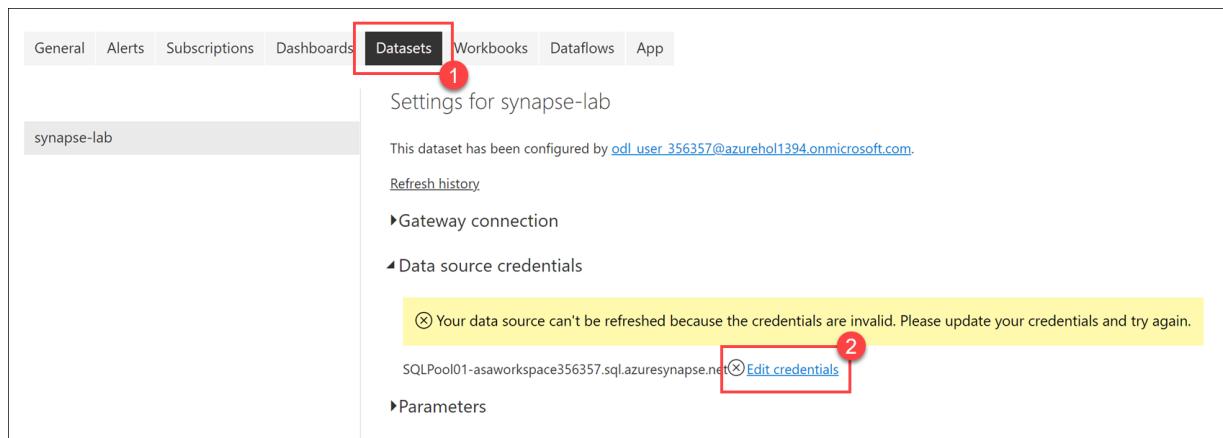
19. Switch back to the Power BI service, or navigate to it in a new browser tab if you closed it earlier (<https://powerbi.microsoft.com/>).
20. Select the **synapse-training** workspace you created earlier. If you already had it open, refresh the page to see the new report and dataset.

	Name	Type	Owner
	synapse-lab	Report	synapse-training
	synapse-lab	Dataset	synapse-training

21. Select the **Settings** gear icon on the upper-right of the page, then select **Settings**. If you do not see the gear icon, you will need to select the ellipses (...) to view the menu item.



22. Select the **Datasets** tab. If you see an error message under **Data source credentials** that your data source can't be refreshed because the credentials are invalid, select **Edit credentials**. It may take a few seconds for this section to appear.



23. In the dialog that appears, select the **OAuth2** authentication method, then select **Sign in**. Enter your credentials if prompted.

Configure synapse-lab

X

server

asaworkspace356357.sql.azuresynapse.net



database

SQLPool01

Authentication method

OAuth2



Privacy level setting for this data source

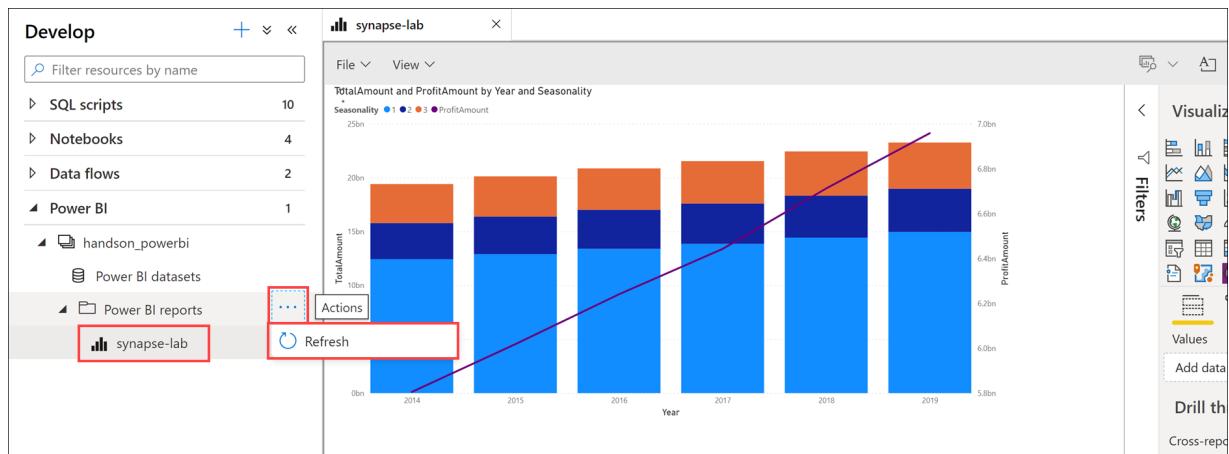


Report viewers can only access this data source with their own Power BI identities using DirectQuery. [Learn more](#)

Sign in

Cancel

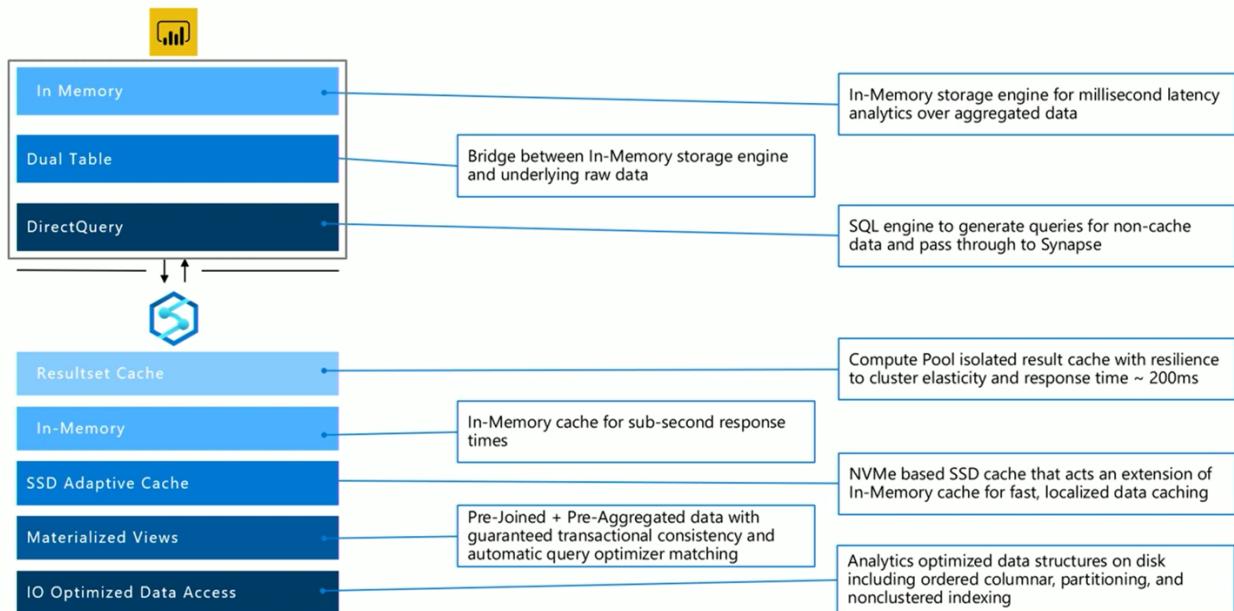
24. Now you should be able to see this report published in Synapse Studio. Switch back to Synapse Studio, select the **Develop** hub and refresh the Power BI reports node.



18.6 Exercise 2: Optimizing integration with Power BI

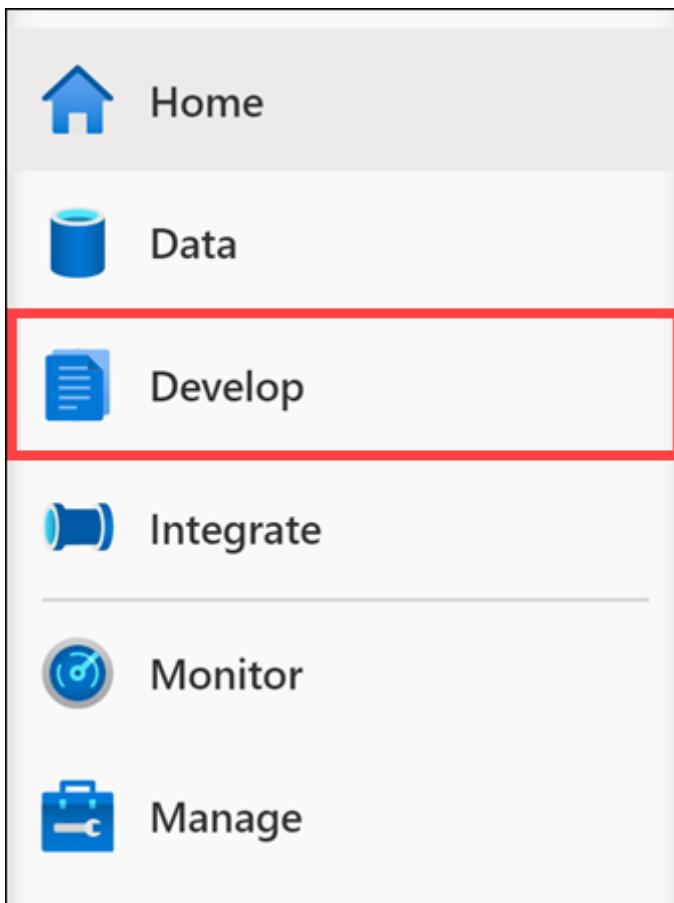
18.6.1 Task 1: Explore Power BI optimization options

Let's recall the performance optimization options we have when integrating Power BI reports in Azure Synapse Analytics, among which we'll demonstrate the use of Result-set caching and materialized views options later in this exercise.

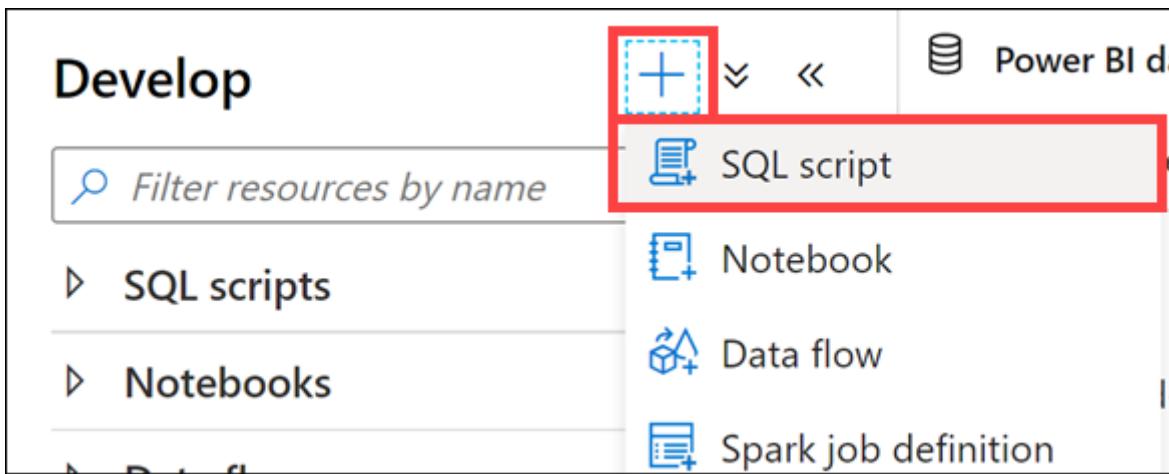


18.6.2 Task 2: Improve performance with materialized views

1. In [Azure Synapse Studio](#), select **Develop** from the left-hand menu.



2. Select **+**, then **SQL script**.



3. Connect to **SQLPool01**, then execute the following query to get an estimated execution plan and observe the total cost and number of operations:

```
EXPLAIN
SELECT * FROM
(
    SELECT
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
        ,avg(FS.TotalAmount) as AvgTotalAmount
        ,avg(FS.ProfitAmount) as AvgProfitAmount
        ,sum(FS.TotalAmount) as TotalAmount
        ,sum(FS.ProfitAmount) as ProfitAmount
    FROM
        wwi.SaleSmall FS
    JOIN wwi.Product P ON P.ProductId = FS.ProductId
    JOIN wwi.Date D ON FS.TransactionDateId = D.DateId
    GROUP BY
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
) T
```

4. The results should look similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<dsql_query number_nodes="1" number_distributions="60" number_distributions_per_node="60">
    <sql>SELECT count(*) FROM
    (
        SELECT
            FS.CustomerID
            ,P.Seasonality
            ,D.Year
            ,D.Quarter
            ,D.Month
            ,avg(FS.TotalAmount) as AvgTotalAmount
            ,avg(FS.ProfitAmount) as AvgProfitAmount
            ,sum(FS.TotalAmount) as TotalAmount
            ,sum(FS.ProfitAmount) as ProfitAmount
        FROM
            wwi.SaleSmall FS
        JOIN wwi.Product P ON P.ProductId = FS.ProductId
```

```

        JOIN wwi.Date D ON FS.TransactionDateId = D.DateId
    GROUP BY
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
    ) T</sql>
    <dsq_operations total_cost="10.61376" total_number_operations="12">

```

5. Replace the query with the following to create a materialized view that can support the above query:

```

IF EXISTS(select * FROM sys.views where name = 'mvCustomerSales')
    DROP VIEW wwi_perf.mvCustomerSales
GO

CREATE MATERIALIZED VIEW
    wwi_perf.mvCustomerSales
WITH
(
    DISTRIBUTION = HASH( CustomerId )
)
AS
SELECT
    FS.CustomerID
    ,P.Seasonality
    ,D.Year
    ,D.Quarter
    ,D.Month
    ,avg(FS.TotalAmount) as AvgTotalAmount
    ,avg(FS.ProfitAmount) as AvgProfitAmount
    ,sum(FS.TotalAmount) as TotalAmount
    ,sum(FS.ProfitAmount) as ProfitAmount
FROM
    wwi.SaleSmall FS
    JOIN wwi.Product P ON P.ProductId = FS.ProductId
    JOIN wwi.Date D ON FS.TransactionDateId = D.DateId
GROUP BY
    FS.CustomerID
    ,P.Seasonality
    ,D.Year
    ,D.Quarter
    ,D.Month
GO

```

This query will take between 60 and 150 seconds to complete.

We first drop the view if it exists, in case it already exists from an earlier lab.

6. Run the following query to check that it actually hits the created materialized view.

```

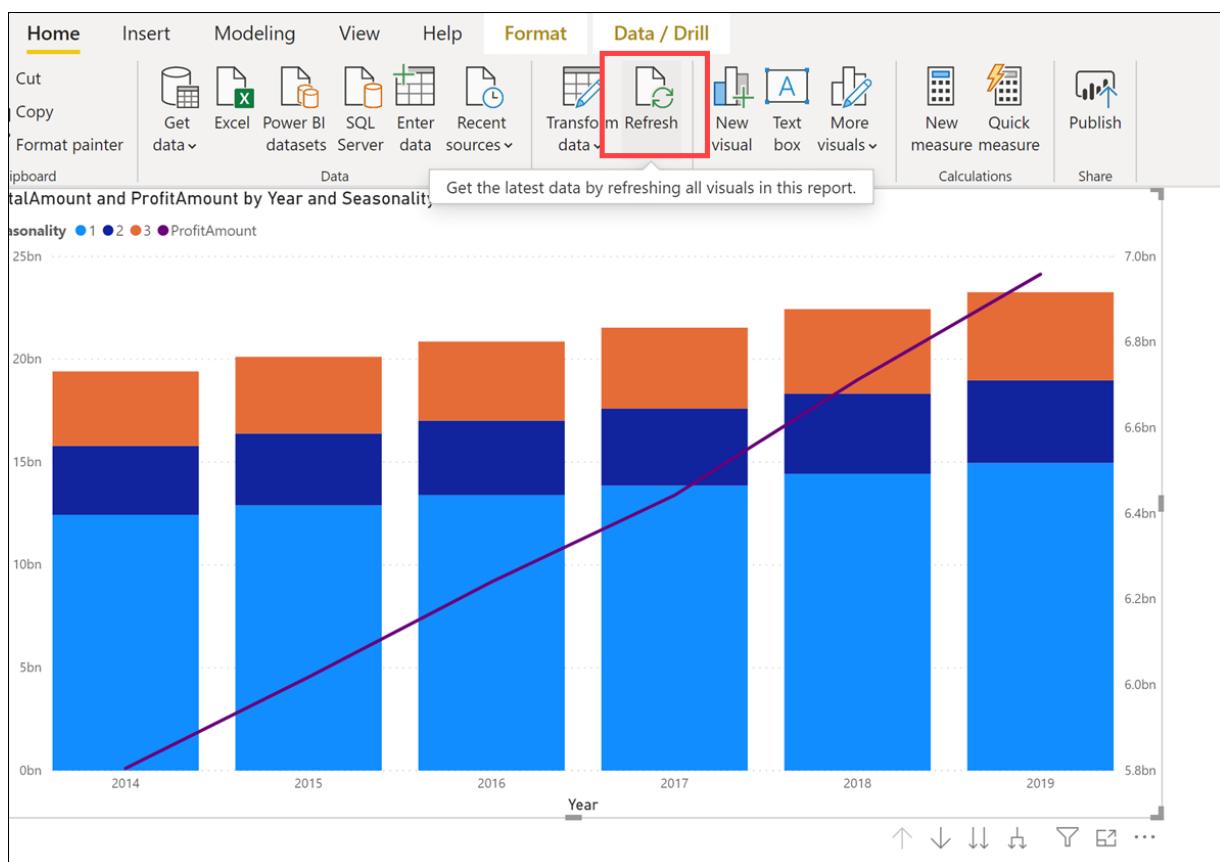
EXPLAIN
SELECT * FROM
(
    SELECT
        FS.CustomerID
        ,P.Seasonality
        ,D.Year
        ,D.Quarter
        ,D.Month
        ,avg(FS.TotalAmount) as AvgTotalAmount
        ,avg(FS.ProfitAmount) as AvgProfitAmount
        ,sum(FS.TotalAmount) as TotalAmount
        ,sum(FS.ProfitAmount) as ProfitAmount

```

FROM

```
wwi.SaleSmall FS  
JOIN wwi.Product P ON P.ProductId = FS.ProductId  
JOIN wwi.Date D ON FS.TransactionDateId = D.DateId  
GROUP BY  
FS.CustomerID  
,P.Seasonality  
,D.Year  
,D.Quarter  
,D.Month  
) T
```

7. Switch back to the Power BI Desktop report, then click on the **Refresh** button above the report to submit the query. The query optimizer should use the new materialized view.



Notice that the data refresh only takes a few seconds now, compared to before.

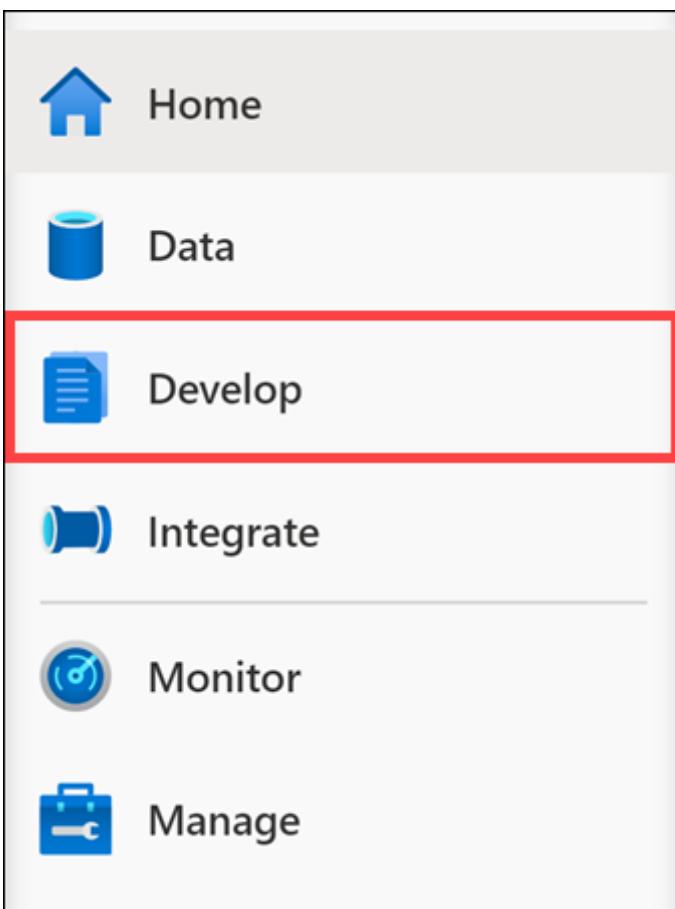
8. Check the duration of the query again in Synapse Studio, in the monitoring hub, under SQL requests. Notice that the Power BI queries using the new materialized view run much faster (Duration ~ 10s).

The screenshot shows the Synapse Studio monitoring hub under the "SQL requests" section. On the left, there is a navigation sidebar with categories like Integration, Activities, and a selected "SQL requests" item. The main area displays a table of SQL requests with the following columns: Request ID, Request content, Submit time, and Duration. The "Pool : SQLPool01" filter is highlighted with a red box. The "Duration" column is also highlighted with a red box. The table shows three rows of data:

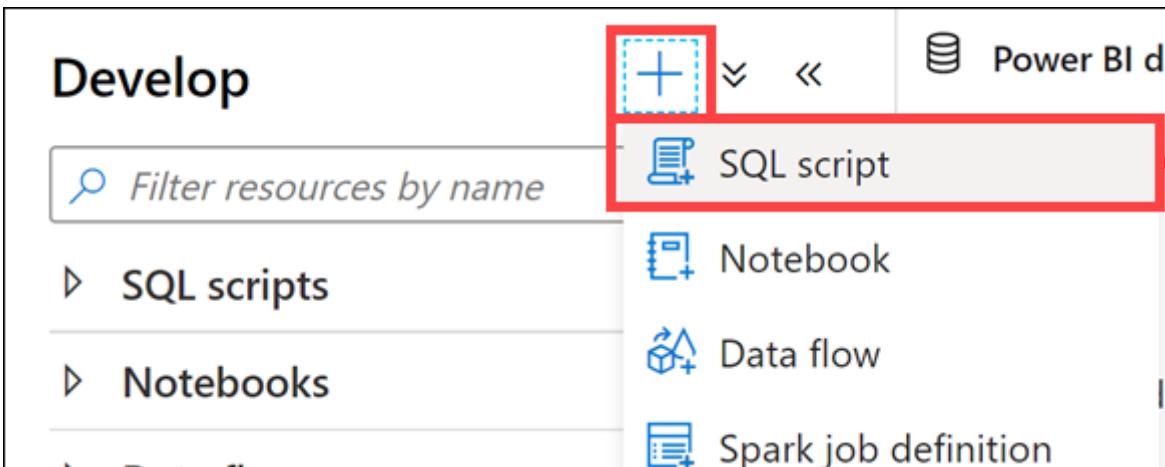
Request ID	Request content	Submit time	Duration
QID6813	SELECT TOP (...)	2023-10-12T10:00:00Z	9 sec
QID6812	SELECT TOP (...)	2023-10-12T10:00:00Z	11 sec
QID6811	USE [SQLPool...]	2023-10-12T10:00:00Z	0s

18.6.3 Task 3: Improve performance with result-set caching

1. In **Azure Synapse Studio**, select **Develop** from the left-hand menu.



2. Select +, then **SQL script**.



3. Connect to **SQLPool01**, then execute the following query to check if result set caching is turned on in the current SQL pool:

```
SELECT
    name
    ,is_result_set_caching_on
FROM
    sys.databases
```

4. If **False** is returned for **SQLPool01**, execute the following query to activate it (you need to run it on the **master** database):

```
ALTER DATABASE [SQLPool01]
```

```
SET RESULT_SET_CACHING ON
```

Connect to **SQLPool01** and use the **master** database:

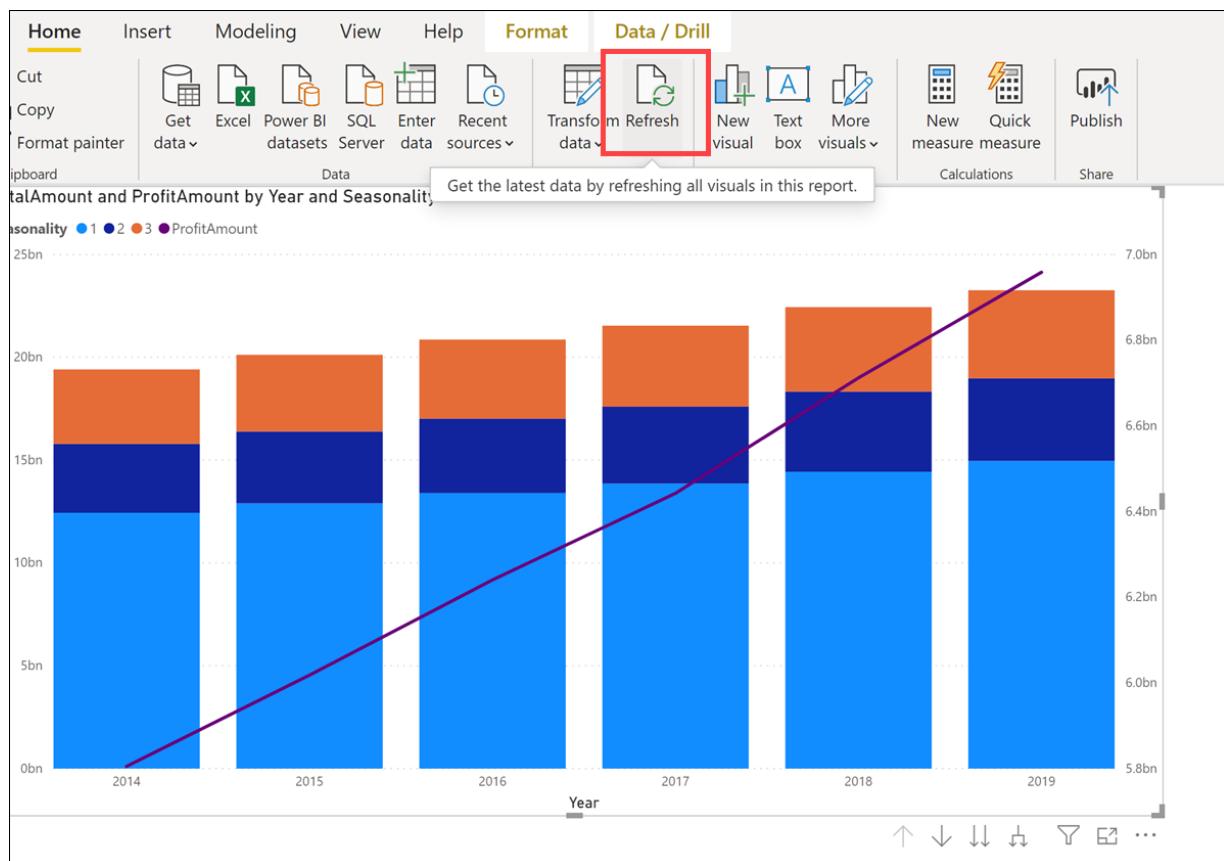
The screenshot shows the Synapse Studio interface with the 'Connect to' dropdown set to 'SQLPool01' and the 'Use database' dropdown set to 'master'. The status bar at the bottom displays the message 'Properties'.

This process takes a couple of minutes to complete. While this is running, continue reading to familiarize yourself with the rest of the lab content.

Important

The operations to create result set cache and retrieve data from the cache happen on the control node of a Synapse SQL pool instance. When result set caching is turned ON, running queries that return large result set (for example, >1GB) can cause high throttling on the control node and slow down the overall query response on the instance. Those queries are commonly used during data exploration or ETL operations. To avoid stressing the control node and cause performance issue, users should turn OFF result set caching on the database before running those types of queries.

5. Next move back to the Power BI Desktop report and hit the **Refresh** button to submit the query again.



6. After the data refreshes, hit **Refresh once more** to ensure we hit the result set cache.
7. Check the duration of the query again in Synapse Studio, in the Monitoring hub - SQL Requests page. Notice that now it runs almost instantly (Duration = 0s).

SQL requests

Showing 1 - 100 of 354 items

Request ID ↑	Request content ↑↓	Submit time ↑↓	Duration
QID6877	SELECT TOP (... More	2023-09-01T10:00:00Z	0s
QID6876	SELECT TOP (... More	2023-09-01T10:00:00Z	0s
QID6875	select * from (...) More	2023-09-01T10:00:00Z	0s

8. Return to the SQL script (or create a new one if you closed it) and run the following on the **master** database while connected to the dedicated SQL pool to turn result set caching back off:

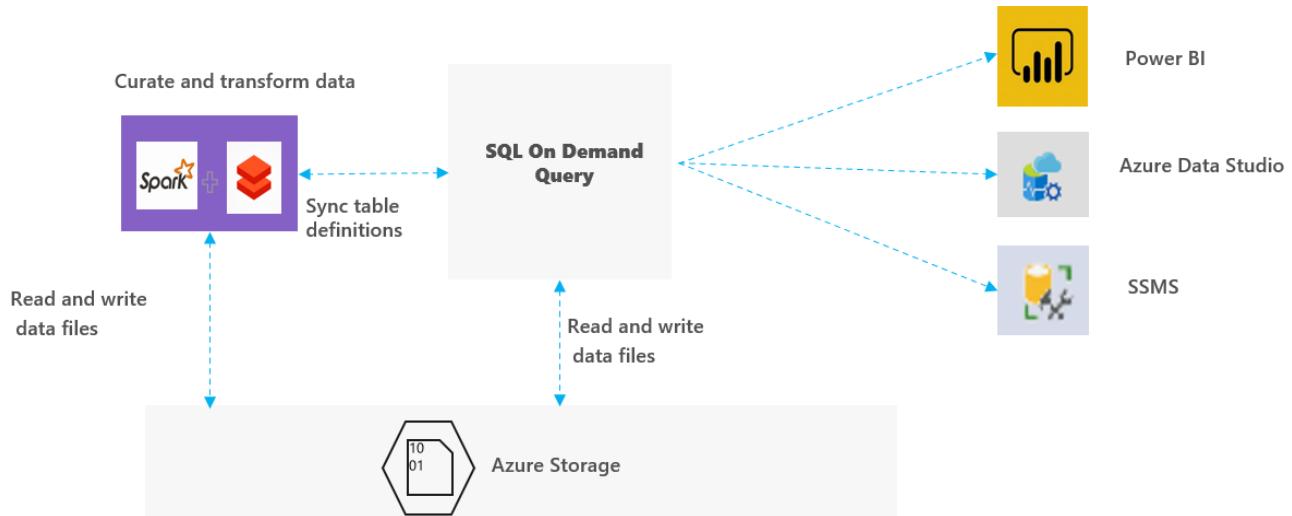
```
ALTER DATABASE [SQLPool01]
SET RESULT_SET_CACHING OFF
```

SQL script 7

Run Undo Publish Query plan Connect to SQLPool01 Use database master

```
1 ALTER DATABASE [SQLPool01]
2 SET RESULT_SET_CACHING OFF
```

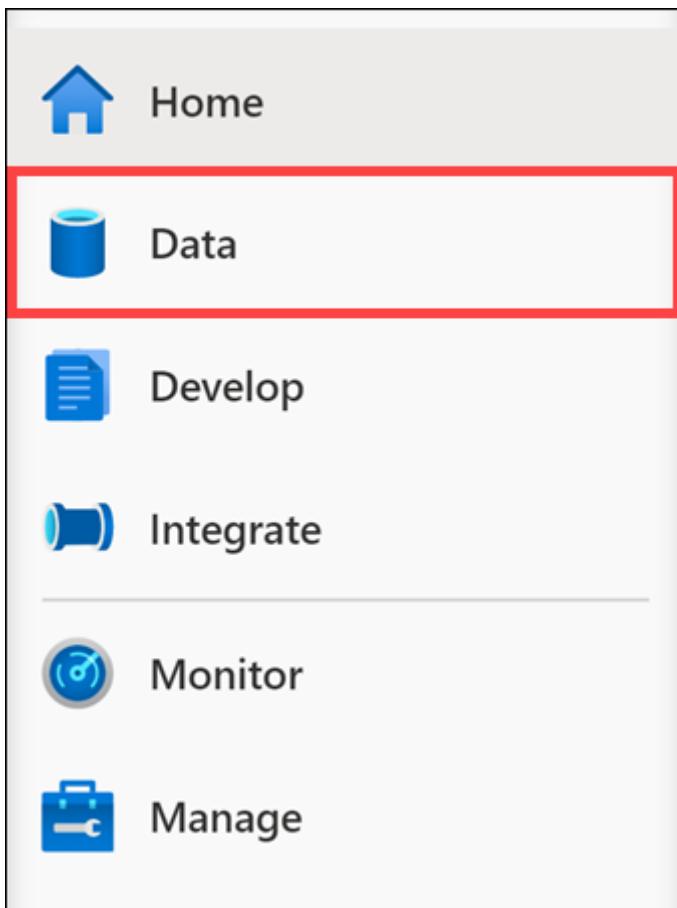
18.7 Exercise 3: Visualize data with SQL Serverless



18.7.1 Task 1: Explore the data lake with SQL Serverless

First, let's prepare the Power BI report query by exploring the data source we'll use for visualizations. In this exercise, we'll use the SQL-on demand instance from your Synapse workspace.

1. In [Azure Synapse Studio](#), navigate to the **Data** hub.



2. Select the **Linked** tab (1). Under the **Azure Data Lake Storage Gen2** group, select the Primary Data Lake (first node) (2) and select the **wwi-02** container (3). Navigate to **wwi-02/sale-small/Year=2019/Quarter=Q1/Month=1/Day=20190101** (4). Right-click on the Parquet file (5), select **New SQL script** (6), then **Select TOP 100 rows** (7).

The screenshot shows the Data Explorer in Azure Data Studio. The left pane displays a tree view of Azure Data Lake Storage Gen2 resources, including workspaces like 'asaworkspace' and containers like 'wwi-02'. The right pane shows a detailed view of the 'wwi-02' container, specifically the 'sale-small' folder under 'Year=2019/Quarter=Q1/Month=1/Day=20190101'. A Parquet file named 'sale-small-20190101-snappy.parquet' is selected. A context menu is open over this file, with steps numbered 5 through 7 indicating the sequence of actions: 5. Right-click on the file; 6. Select 'New SQL script'; 7. Select 'Select TOP 100 rows'.

3. Run the generated script to preview data stored in the Parquet file.

```

1 SELECT
2     TOP 100 *
3 FROM
4     OPENROWSET(
5         BULK 'https://asadatalake264973.dfs.core.windows.net/wwi-02/sale-small/Year=2019/Quarter=Q1/Month=1/Day=2019'
6         FORMAT='PARQUET'
7     ) AS [result]
8

```

Results Messages

Select Query 2 View Table Chart Export results ▾

TransactionId	CustomerId	ProductId	Quantity	Price	TotalAmount	TransactionDate	ProfitAmount	Ho...
9992daf3-b3bf...	13	4169	4	23.9800000000...	95.9200000000...	20190101	24.0400000000...	8
9992daf3-b3bf...	13	4169	4	23.9800000000...	95.9200000000...	20190101	24.0400000000...	8
9992daf3-b3bf...	13	3246	1	28.7200000000...	28.7200000000...	20190101	7.3000000000...	8
9992daf3-b3bf...	13	611	1	23.3300000000...	23.3300000000...	20190101	6.5900000000...	8
9992daf3-b3bf...	13	153	3	27.4300000000...	82.2900000000...	20190101	21.0000000000...	8
9992daf3-b3bf...	13	134	2	29.1600000000...	58.3200000000...	20190101	18.2600000000...	8
9992daf3-b3bf...	13	251	1	22.9800000000...	22.9800000000...	20190101	6.2100000000...	8
9992daf3-b3bf...	13	98	2	27.7500000000...	55.5000000000...	20190101	15.5800000000...	8
9992daf3-b3bf...	13	205	2	39.1800000000...	78.3600000000...	20190101	26.3400000000...	8
5ced6131-97b2...	23	4162	2	19.1300000000...	38.2600000000...	20190101	12.6600000000...	5

- Copy the name of the primary data lake storage account from the query and save it in Notepad or similar text editor. You need this value for the next step.

```

1 SELECT
2     TOP 100 *
3 FROM
4     OPENROWSET(
5         BULK 'https://asadatalake...dfs.core.windows.net/wwi-02/sale-small/Year=2019/Quarter=Q1/Month=1/Day=2019'
6         FORMAT='PARQUET'
7     ) AS [result]
8

```

- Now let's prepare the query we want to use next in the Power BI report. The query will extract the sum of amount and profit by day for a given month. Let's take for example January 2019. Notice the use of wildcards for the filepath that will reference all the files corresponding to one month. Paste and run the following query on the SQL-on-demand instance and replace YOUR_STORAGE_ACCOUNT_NAME with the name of the storage account you copied above:

```

DROP DATABASE IF EXISTS demo;
GO

CREATE DATABASE demo;
GO

USE demo;
GO

CREATE VIEW [2019Q1Sales] AS
SELECT
    SUBSTRING(result.filename(), 12, 4) as Year
    ,SUBSTRING(result.filename(), 16, 2) as Month
    ,SUBSTRING(result.filename(), 18, 2) as Day
    ,SUM(TotalAmount) as TotalAmount
    ,SUM(ProfitAmount) as ProfitAmount
    ,COUNT(*) as TransactionsCount
FROM

```

```

OPENROWSET(
    BULK 'https://YOUR_STORAGE_ACCOUNT_NAME.dfs.core.windows.net/ww1-02/sale-small/Year=2019/Q1/Quarter=Q1/Month=1/*/*.parquet'
    FORMAT='PARQUET'
) AS [result]
GROUP BY
    [result].filename()
GO

```

You should see a query output similar to the following:

```

1  DROP DATABASE IF EXISTS demo;
2  GO
3
4  CREATE DATABASE demo;
5  GO
6
7  USE demo;
8  GO
9
10 CREATE VIEW [2019Q1Sales] AS
11  SELECT
12      SUBSTRING(result.filename(), 12, 4) as Year
13      ,SUBSTRING(result.filename(), 16, 2) as Month
14      ,SUBSTRING(result.filename(), 18, 2) as Day
15      ,SUM(TotalAmount) as TotalAmount
16      ,SUM(ProfitAmount) as ProfitAmount
17      ,COUNT(*) as TransactionsCount
18  FROM
19  OPENROWSET(
20      BULK 'https://asadatalake264973.dfs.core.windows.net/ww1-02/sale-small/Year=2019/Quarter=Q1/Month=1/*/*.parquet',
21      FORMAT='PARQUET'
22  ) AS [result]
23  GROUP BY
24  |    [result].filename()
25  GO

```

Results	Messages
	11:04:28 PM Started executing query at Line 1 (Potential conversion error while reading VARCHAR column 'TransactionId' from UTF8 encoded text. Change database collation to a UTF8 collation or specify explicit column schema in WITH clause columns. Changed database context to 'demo'.) Total execution time: 00:00:04.661

00:00:04 Query executed successfully.

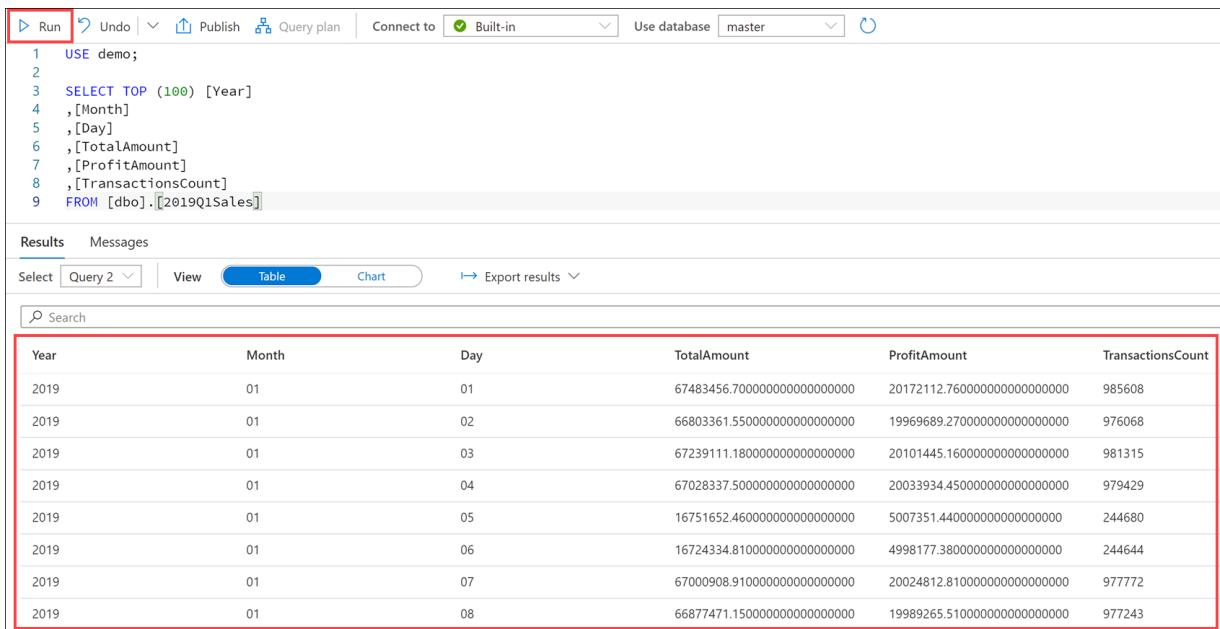
- Replace the query with the following to select from the new view you created:

```

USE demo;

SELECT TOP (100) [Year]
,[Month]
,[Day]
,[TotalAmount]
,[ProfitAmount]
,[TransactionsCount]
FROM [dbo].[2019Q1Sales]

```



```

1 USE demo;
2
3 SELECT TOP (100) [Year]
4 ,[Month]
5 ,[Day]
6 ,[TotalAmount]
7 ,[ProfitAmount]
8 ,[TransactionsCount]
9 FROM [dbo].[2019Q1Sales]

```

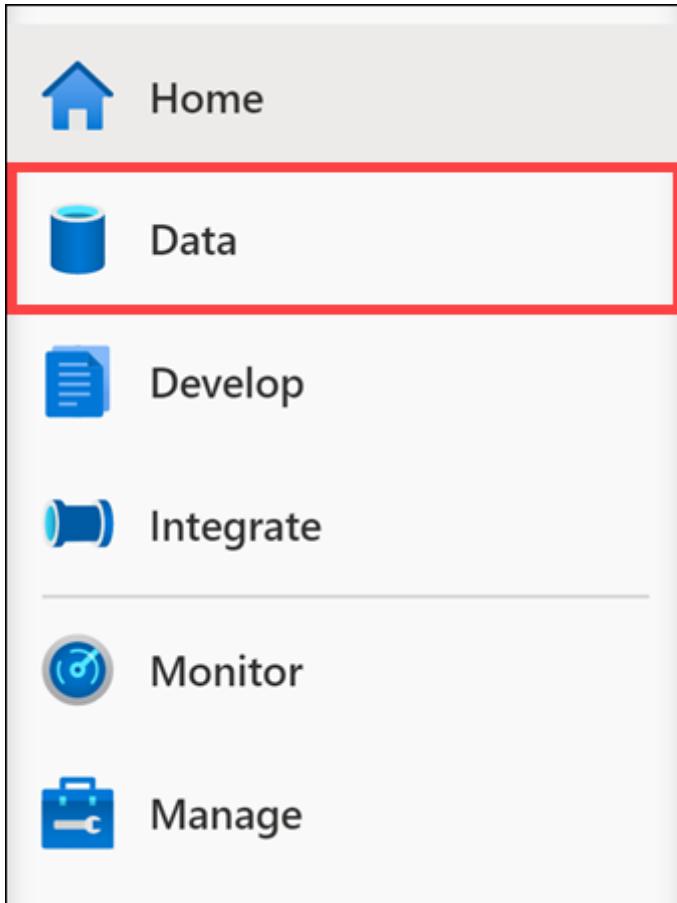
Results Messages

Select Query 2 View Table Chart Export results

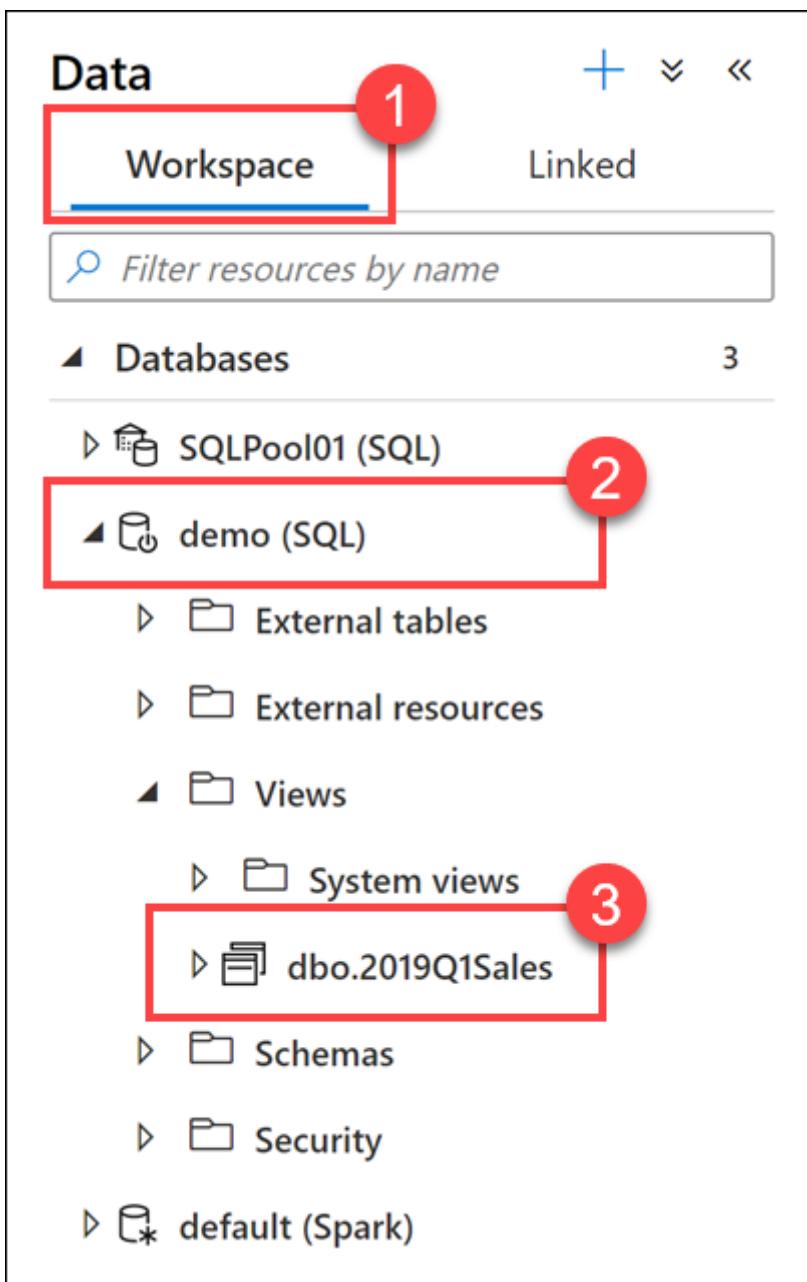
Search

Year	Month	Day	TotalAmount	ProfitAmount	TransactionsCount
2019	01	01	67483456.70000000000000000000	20172112.76000000000000000000	985608
2019	01	02	66803361.55000000000000000000	19969689.27000000000000000000	976068
2019	01	03	67239111.18000000000000000000	20101445.16000000000000000000	981315
2019	01	04	67028337.50000000000000000000	20033934.45000000000000000000	979429
2019	01	05	16751652.46000000000000000000	5007351.44000000000000000000	244680
2019	01	06	16724334.81000000000000000000	4998177.38000000000000000000	244644
2019	01	07	67000908.91000000000000000000	20024812.81000000000000000000	977772
2019	01	08	66877471.15000000000000000000	19989265.51000000000000000000	977243

7. Navigate to the **Data** hub in the left-hand menu.



8. Select the **Workspace** tab (1), then right-click the **Databases** group and select **Refresh** to update the list of databases. Expand the Databases group and expand the new **demo (SQL on-demand)** database you created (2). You should see the **dbo.2019Q1Sales** view listed within the Views group (3).



Note

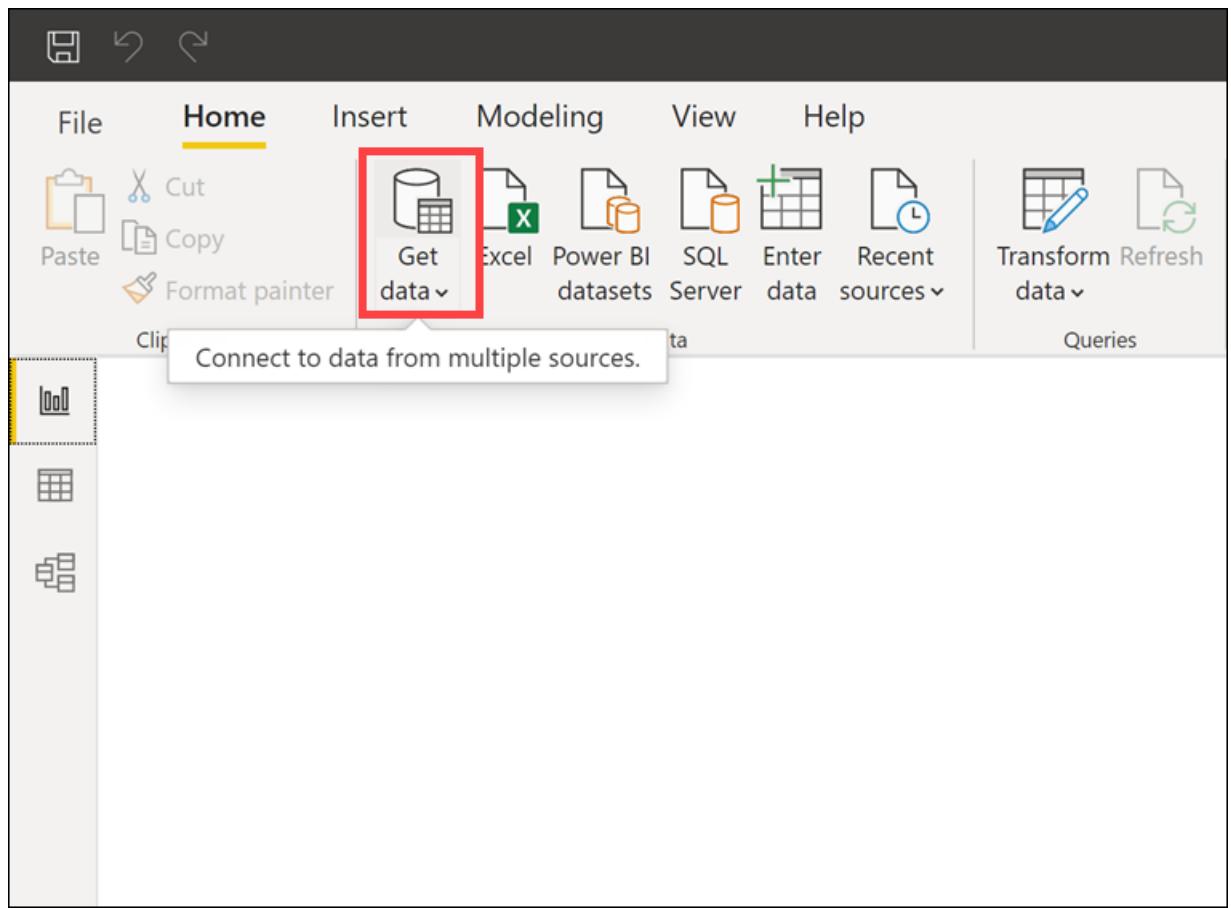
Synapse serverless SQL databases are used only for viewing metadata, not for actual data.

18.7.2 Task 2: Visualize data with SQL serverless and create a Power BI report

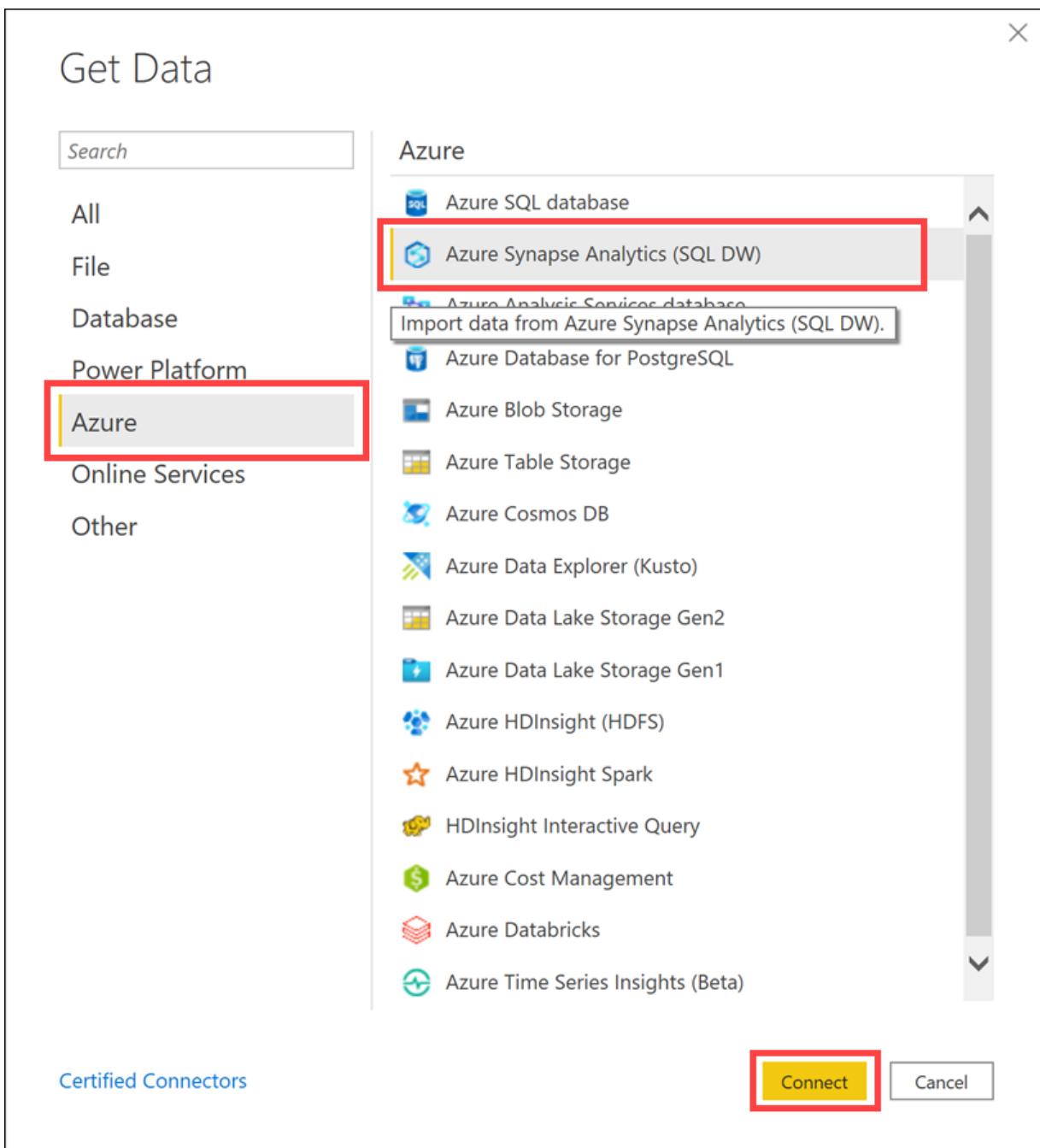
- In the [Azure Portal](#), navigate to your Synapse Workspace. In the **Overview** tab, copy the **Serverless SQL endpoint**:

Setting	Value
Resource group (Change)	: [REDACTED]
Status	: [REDACTED]
Location	: [REDACTED]
Subscription (change)	: [REDACTED]
Subscription ID	: [REDACTED]
Managed virtual network	: No
Managed Identity object ...	: [REDACTED]
Workspace web URL	: https://web.azureusynapse.net?workspace=%2fsubscriptions%...
Firewalls	: Show firewall settings
Dedicated SQL endpoint	: asaworkspace1.sql.azuresynapse.net
Serverless SQL endpoint	: asaworkspace1-on-demand.sql.azuresynapse.net
Development endpoint	: https://asaworkspace1.dev.azuresynapse.net

- Switch back to Power BI Desktop. Create a new report, then click **Get data**.

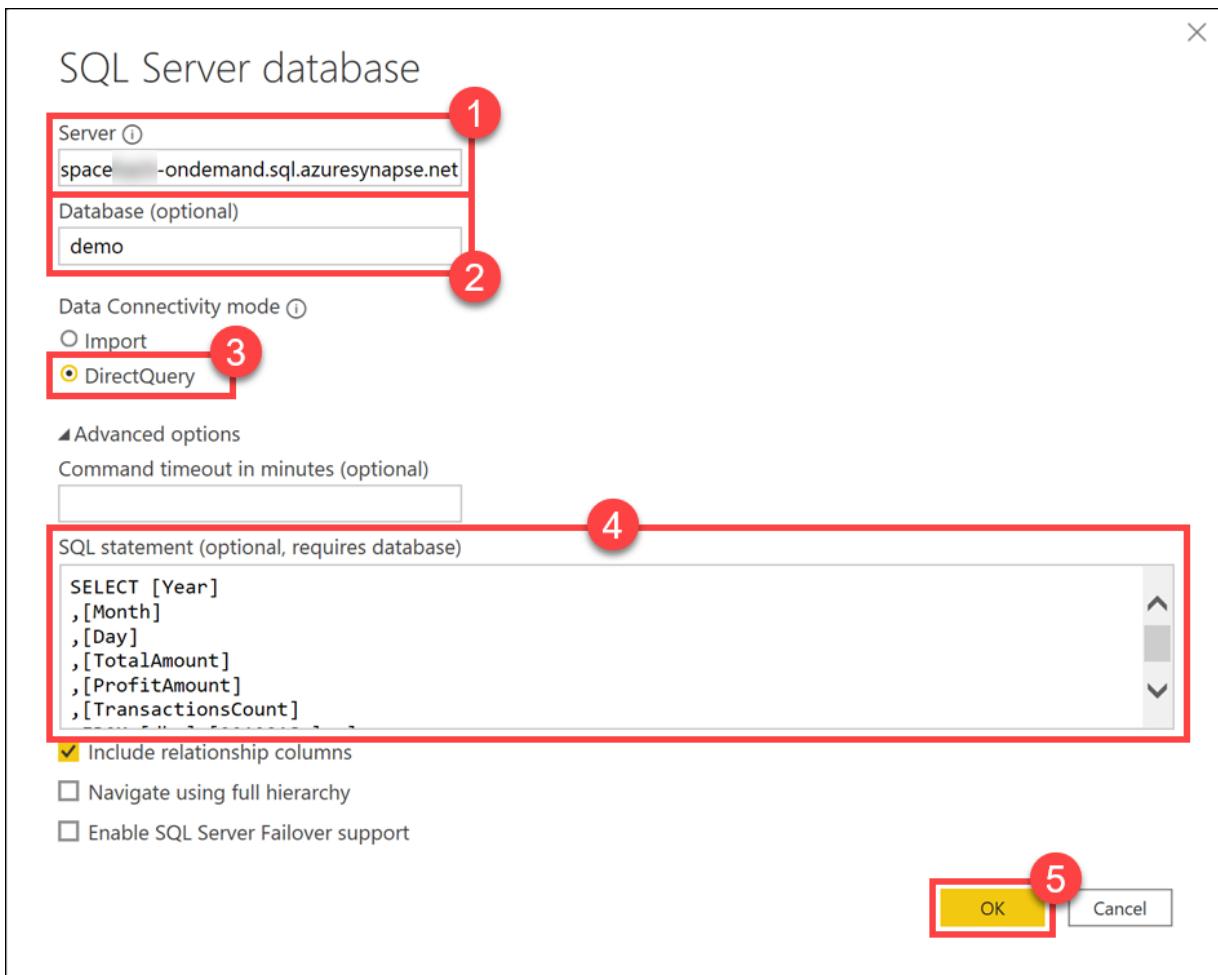


3. Select **Azure** on the left-hand menu, then select **Azure Synapse Analytics (SQL DW)**. Finally, click **Connect**:

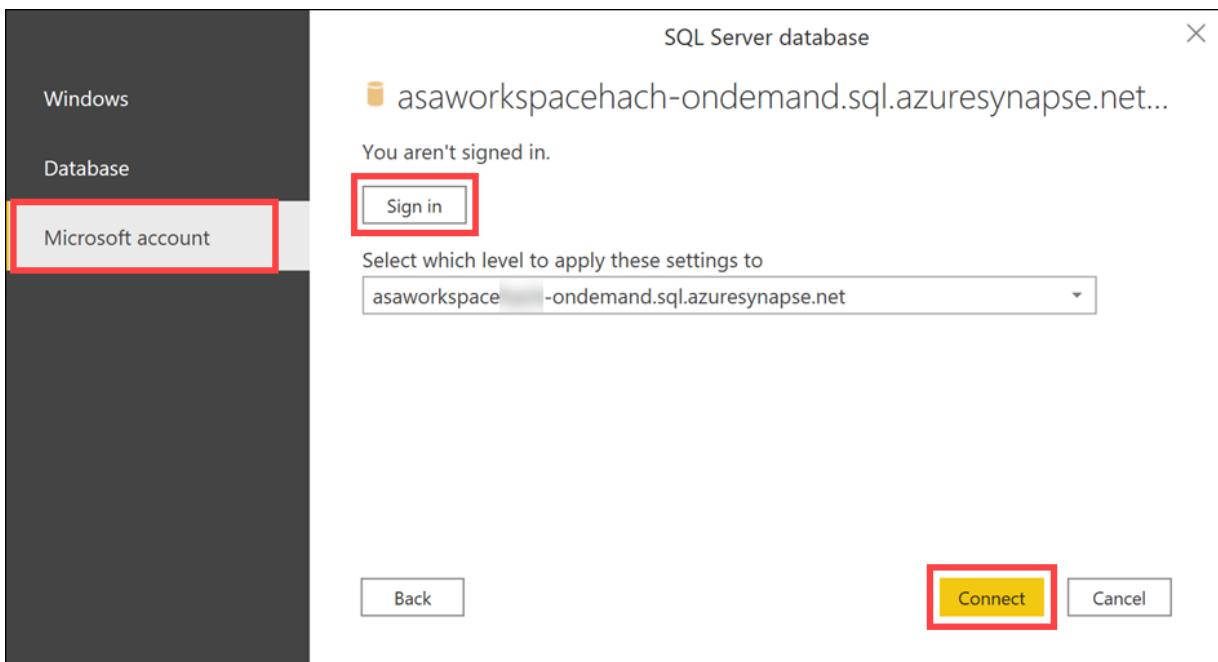


4. Paste the endpoint to the Serverless SQL endpoint identified on the first step into the **Server** field (1), enter **demo** for the **Database** (2), select **DirectQuery** (3), then **paste the query below** (4) into the expanded **Advanced options** section of the SQL Server database dialog. Finally, click **OK** (5).

```
SELECT TOP (100) [Year]
,[Month]
,[Day]
,[TotalAmount]
,[ProfitAmount]
,[TransactionsCount]
FROM [dbo].[2019Q1Sales]
```



5. (If prompted) Select the **Microsoft account** option on the left, **Sign in** (with the same credentials you use for connecting to the Synapse workspace) and click **Connect**.



6. Select **Load** in the preview data window and wait for the connection to be configured.

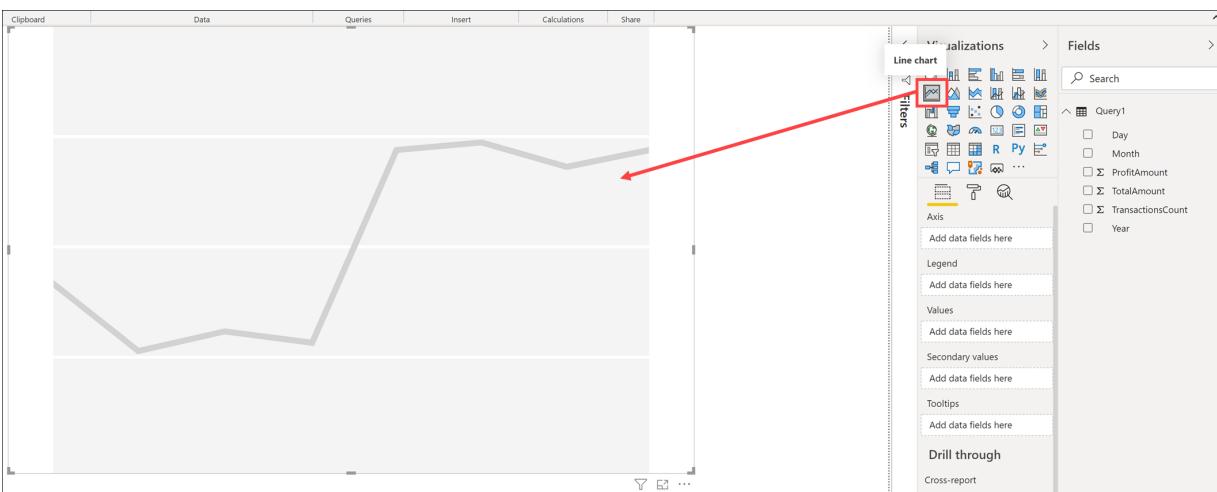
asaworkspace [REDACTED]-ondemand.sql.azuresynapse.net: demo

Year	Month	Day	TotalAmount	ProfitAmount	TransactionsCount
2019	01	12	16756025.38	5011485.96	244544
2019	01	11	66940179.17	20001784.11	977868
2019	01	03	67239111.18	20101445.16	981315
2019	01	10	67132263.31	20070031.54	979252
2019	01	26	16944412.83	5065306.11	247486
2019	01	07	67000908.91	20024812.81	977772
2019	01	17	66767008.65	19953601.19	976619
2019	01	27	16923618.38	5057844.72	247010
2019	01	20	16822669.83	5028092.81	245715
2019	01	09	67124352.12	20062412.13	980116
2019	01	04	67028337.5	20033934.45	979429
2019	01	02	66803361.55	19969689.27	976068
2019	01	29	67354791.26	20134734.01	984338
2019	01	14	66989079.41	20021974.54	977938
2019	01	13	16706214.33	4993204.31	243483
2019	01	23	67197864.66	20086095.45	982519
2019	01	22	67332112.27	20125594.29	983780
2019	01	24	67265525.77	20111449.35	982327
2019	01	25	66865687.57	19984963.44	976185
2019	01	18	67293166.56	20112519.18	982882

The data in the preview has been truncated due to size limits.

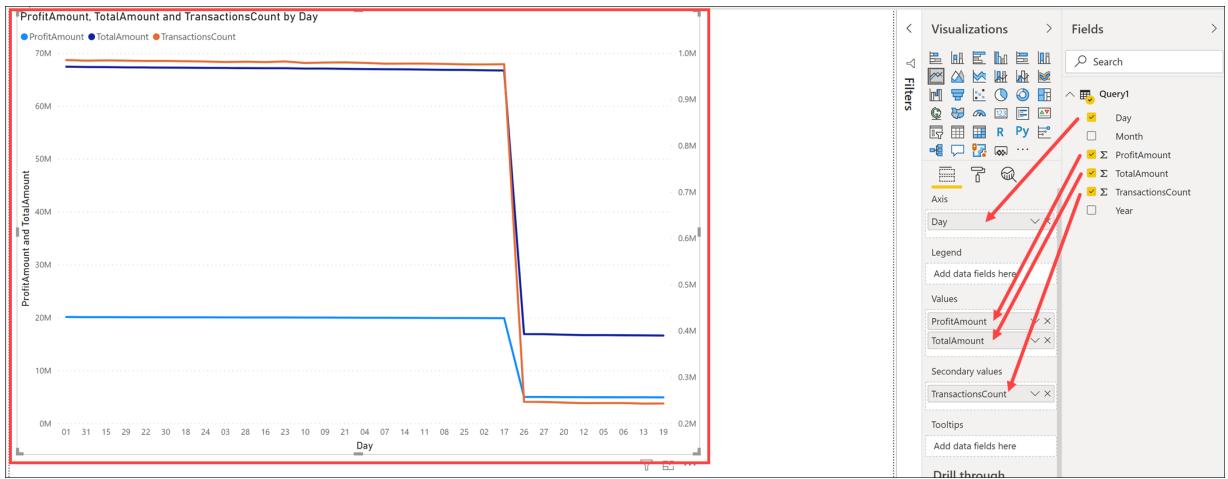
Load Transform Data Cancel

7. After the data loads, select **Line chart** from the **Visualizations** menu.



8. Select the line chart visualization and configure it as follows to show Profit, Amount, and Transactions count by day:

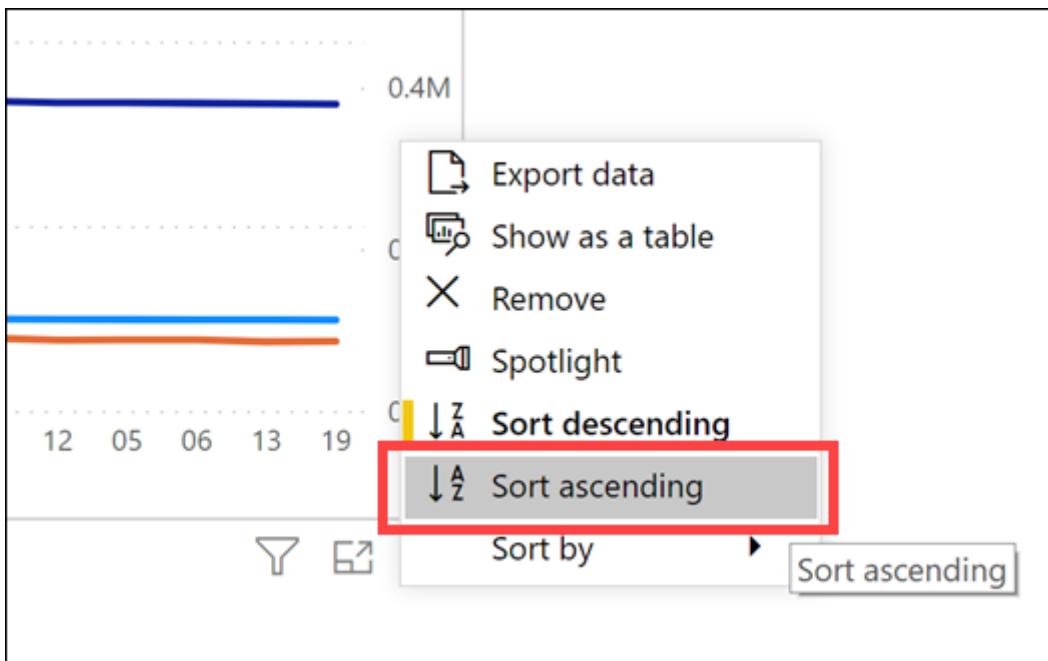
- **Axis:** Day
- **Values:** ProfitAmount, TotalAmount
- **Secondary values:** TransactionsCount



9. Select the line chart visualization and configure it to sort in ascending order by the day of transaction. To do this, select **More options** next to the chart visualization.



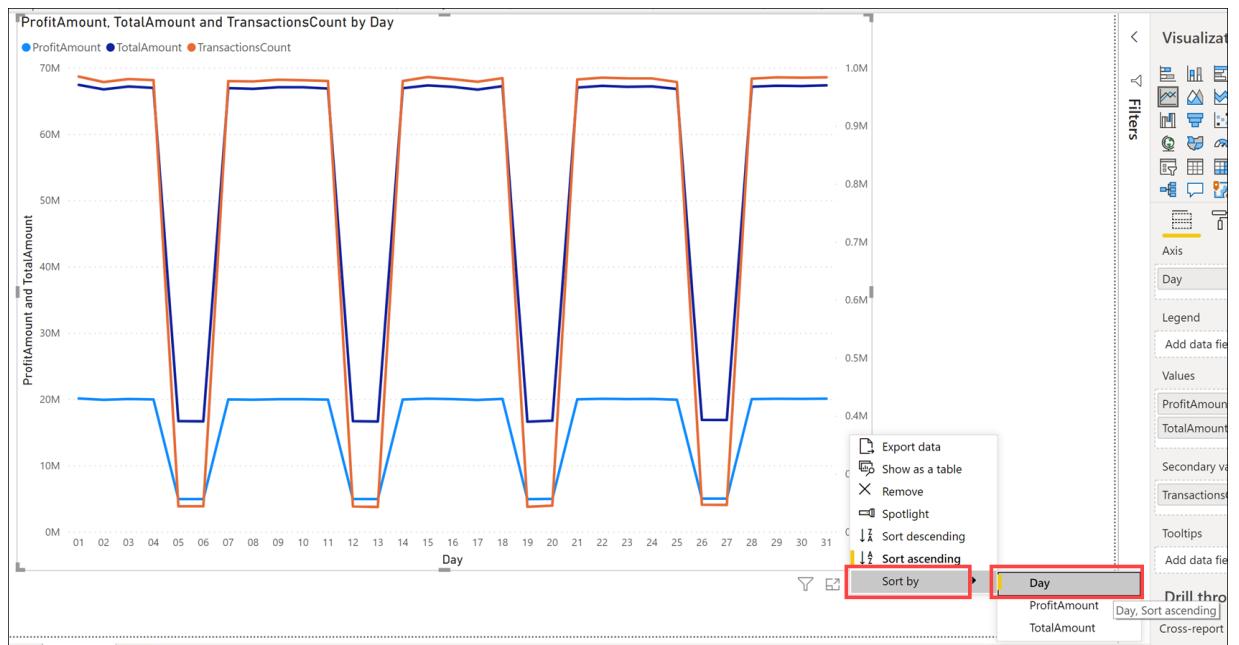
Select **Sort ascending**.



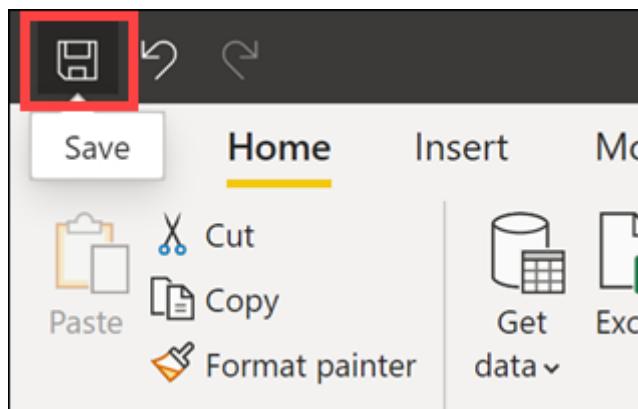
Select **More options** next to the chart visualization again.



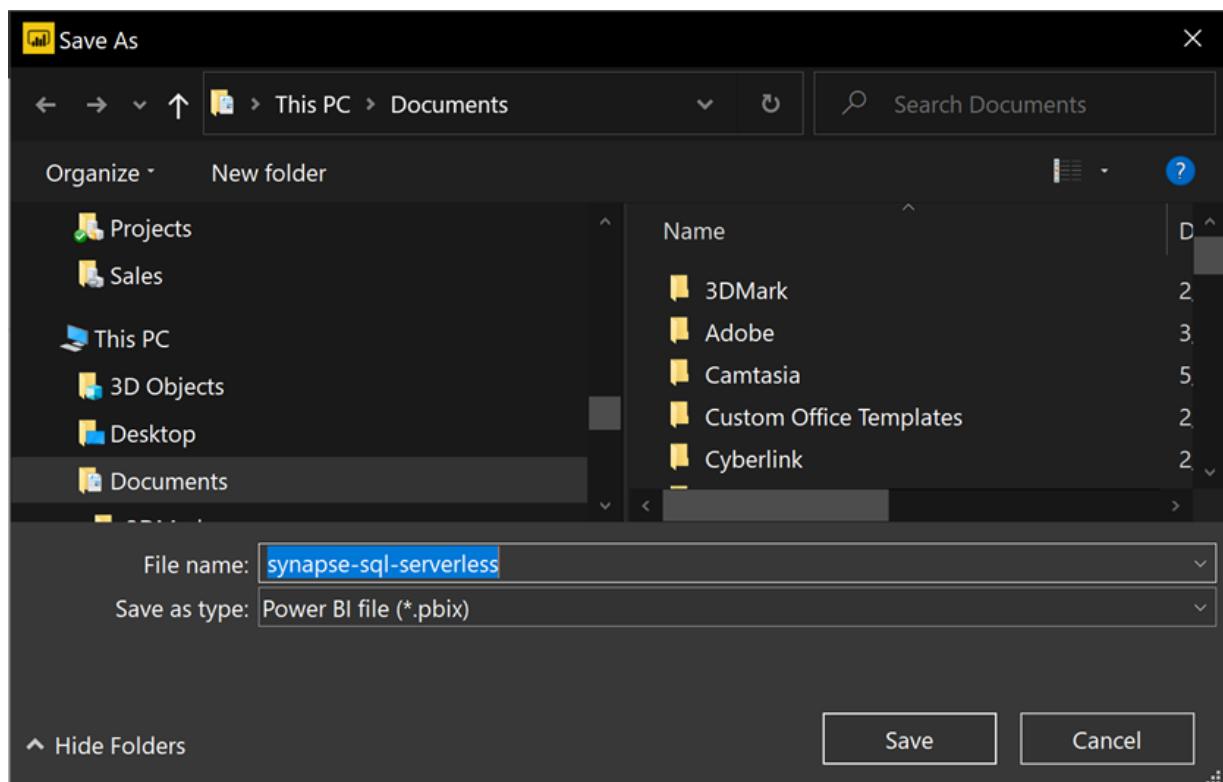
Select **Sort by**, then **Day**.



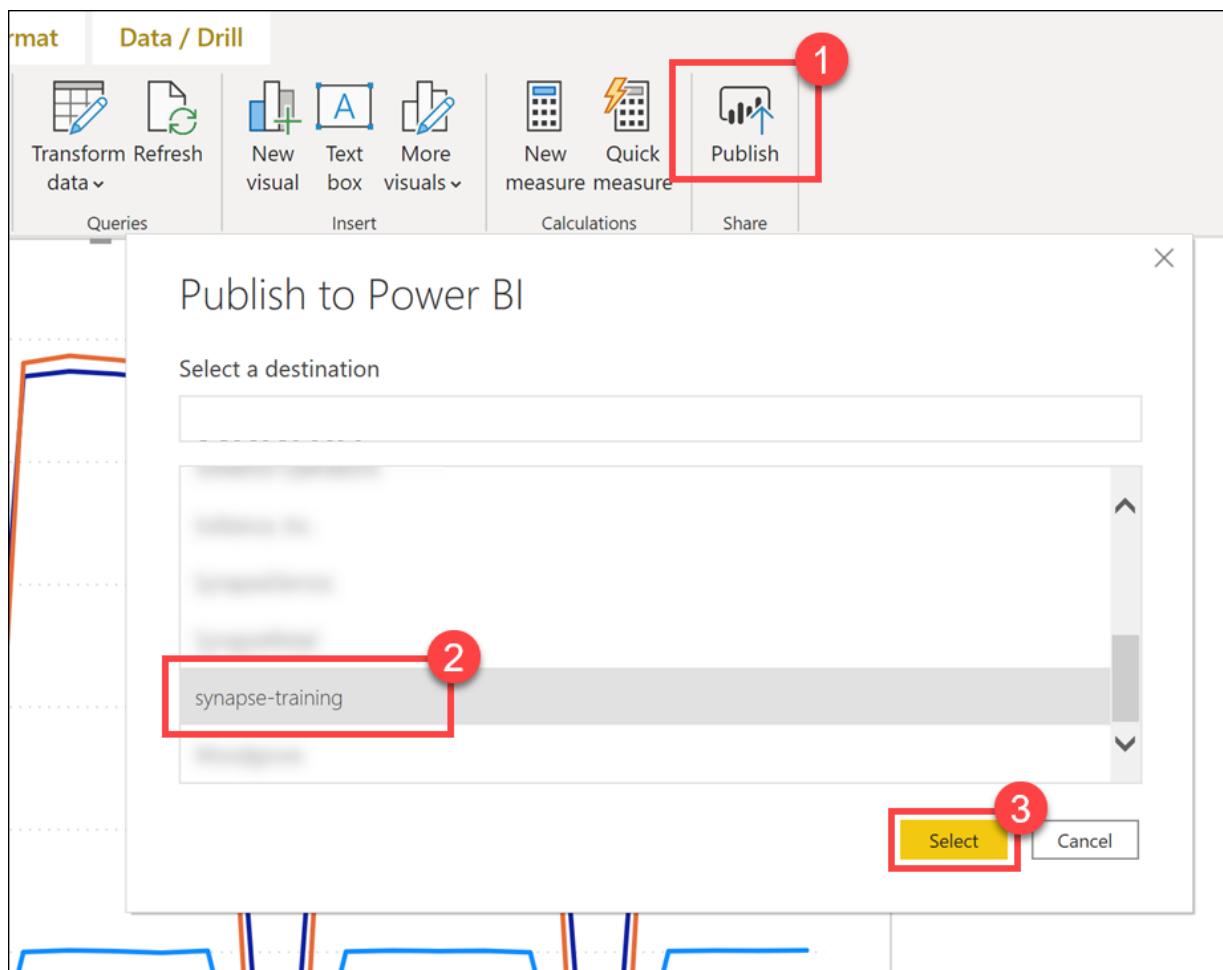
10. Click **Save** in the top-left corner.



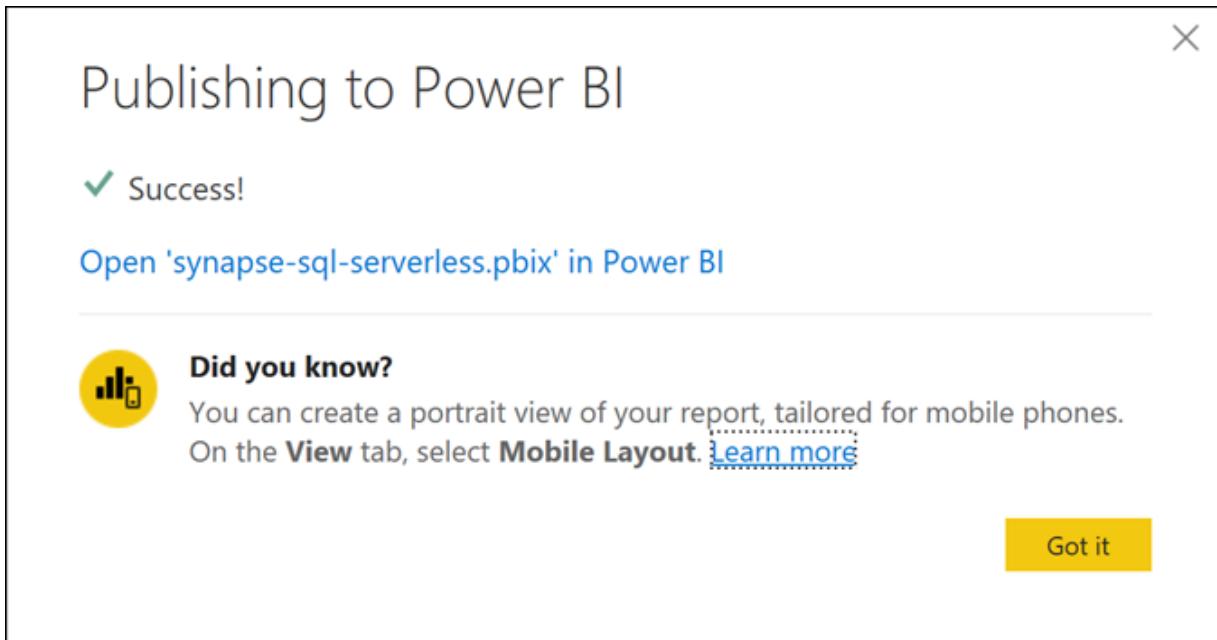
11. Specify a file name, such as `synapse-sql-serverless`, then click **Save**.



12. Click **Publish** above the saved report. Make sure that, in Power BI Desktop, you are signed in with the same account you use in the Power BI portal and in Synapse Studio. You can switch to the proper account from the right topmost corner of the window. In the **Publish to Power BI** dialog, select the workspace you linked to Synapse (for example, **synapse-training**), then click **Select**.



- Wait until the publish operation successfully completes.

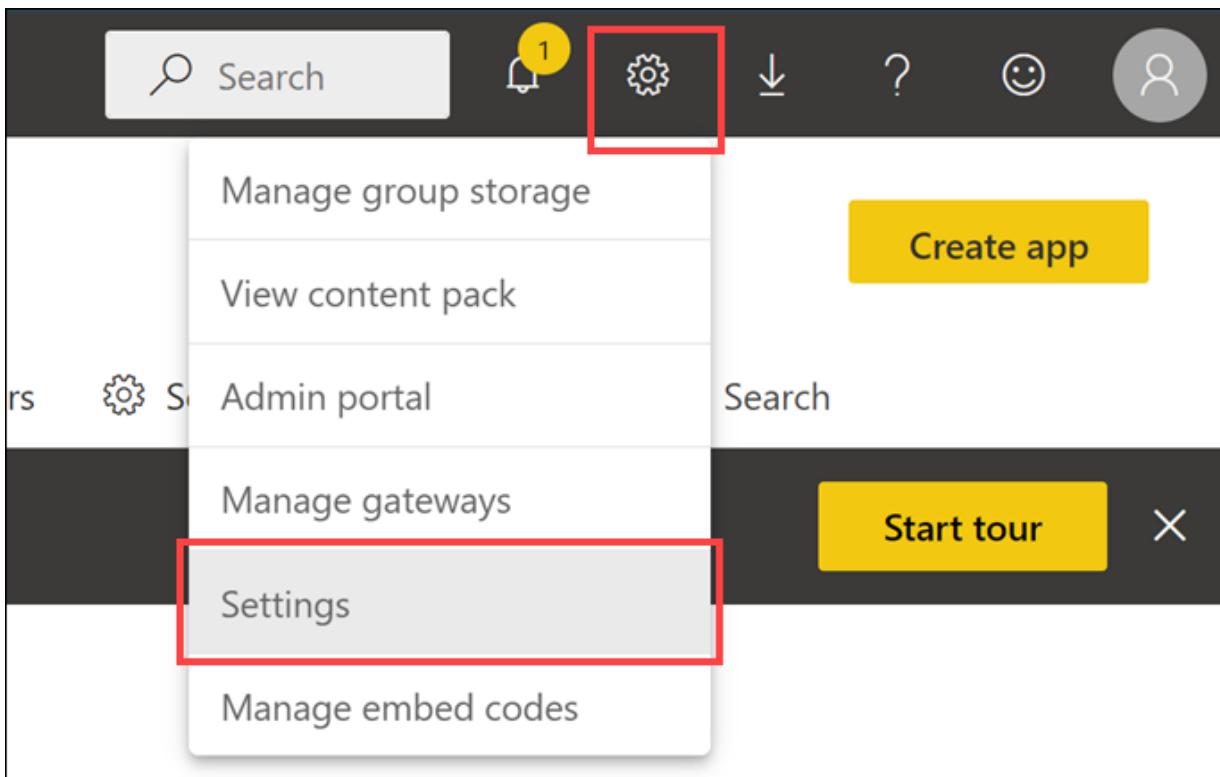


- Switch back to the Power BI service, or navigate to it in a new browser tab if you closed it earlier (<https://powerbi.microsoft.com/>).
- Select the **synapse-training** workspace you created earlier. If you already had it open, refresh the page to see the new report and dataset.

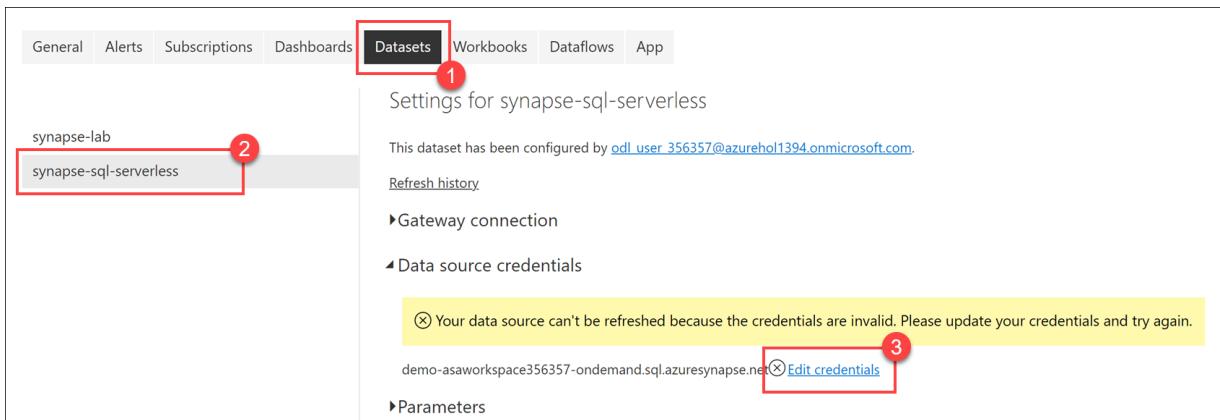
The screenshot shows the Power BI service interface. The left sidebar has a "Workspaces" section with a dropdown menu. The "synapse-training" workspace is selected and highlighted with a red box. The main area shows the "synapse-training" workspace with a "Welcome to workspaces" message. Below it is a table listing datasets and reports. The table has columns for Name, Type, and Owner. Two items are highlighted with a red box: a report named "synapse-sql-serverless" and a dataset named "synapse-sql-serverless".

Name	Type	Owner
synapse-lab	Report	synapse-training
synapse-lab	Dataset	synapse-training
synapse-sql-serverless	Report	synapse-training
synapse-sql-serverless	Dataset	synapse-training

- Select the **Settings** gear icon on the upper-right of the page, then select **Settings**. If you do not see the gear icon, you will need to select the ellipses (...) to view the menu item.



17. Select the **Datasets** tab (1), then select the **synapse-sql-serverless** dataset (2). If you see an error message under **Data source credentials** that your data source can't be refreshed because the credentials are invalid, select **Edit credentials** (3). It may take a few seconds for this section to appear.



18. In the dialog that appears, select the **OAuth2** authentication method, then select **Sign in**. Enter your credentials if prompted.

Configure synapse-lab

X

server

asaworkspace356357.sql.azureSynapse.net



database

SQLPool01

Authentication method

OAuth2



Privacy level setting for this data source

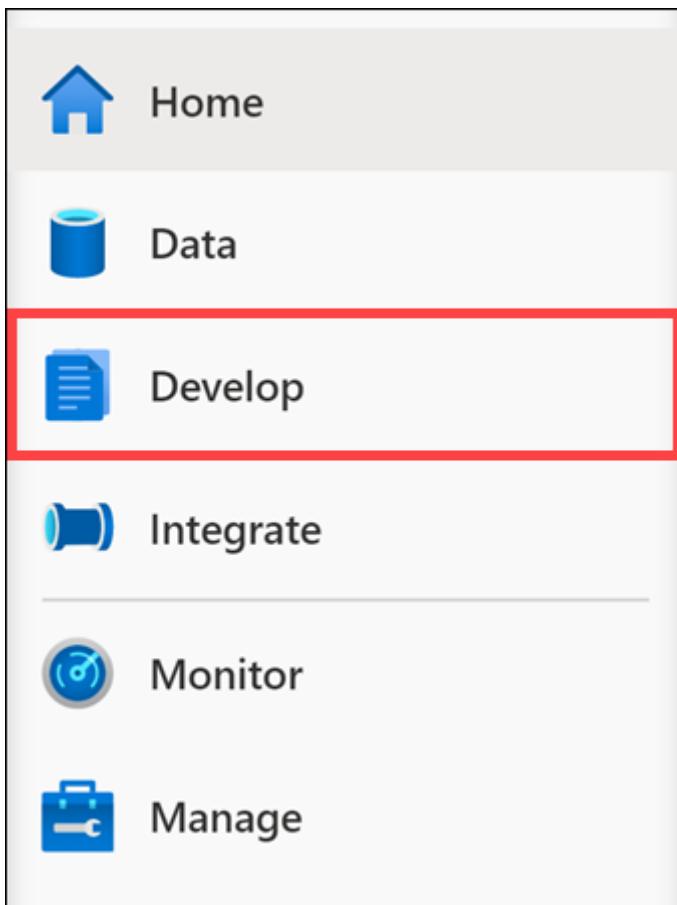


Report viewers can only access this data source with their own Power BI identities using DirectQuery. [Learn more](#)

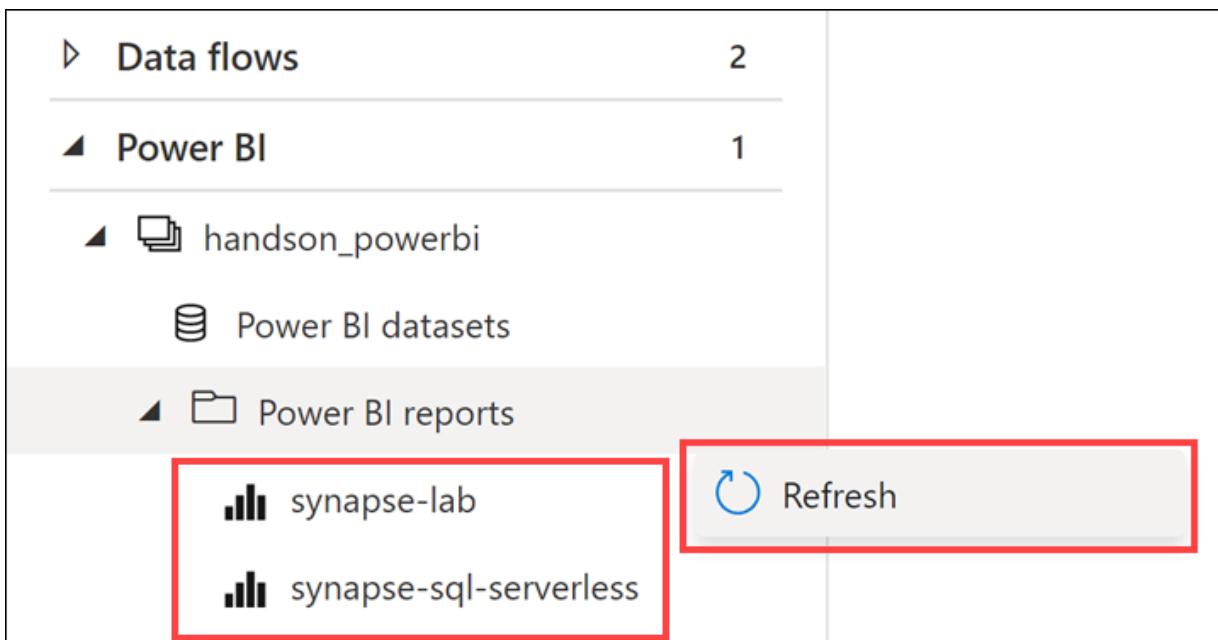
Sign in

Cancel

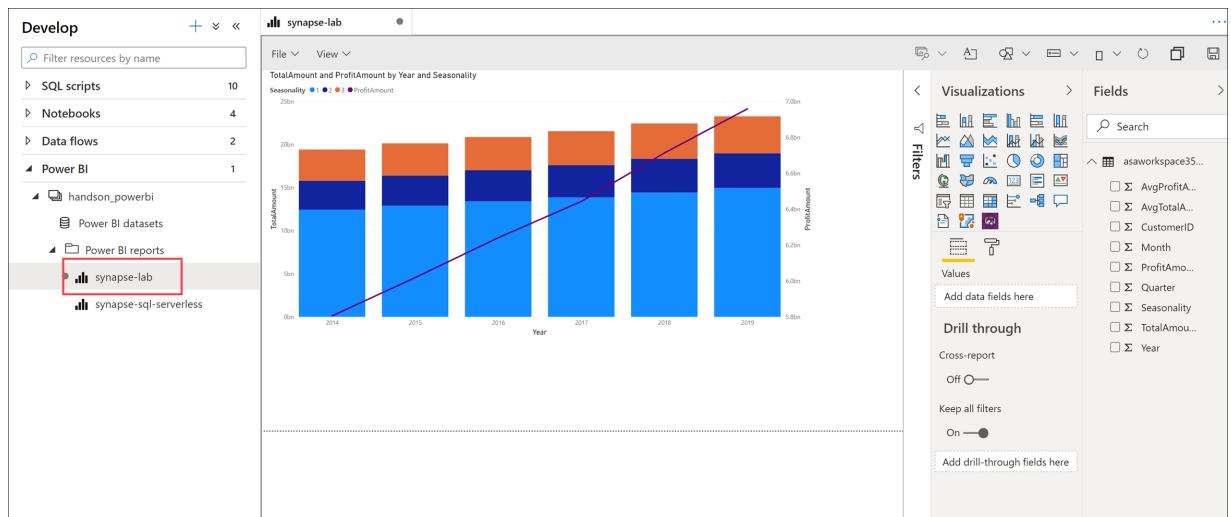
19. In Azure Synapse Studio, navigate to the **Develop** hub.



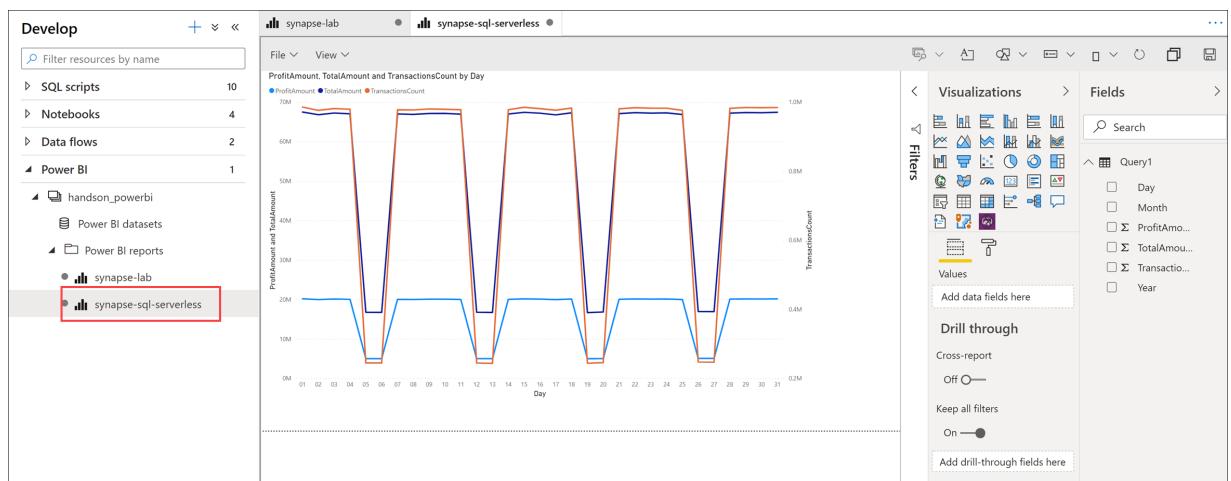
20. Expand the Power BI group, expand your Power BI linked service (for example, handson_powerbi), right-click on **Power BI reports** and select **Refresh** to update the list of reports. You should see the two Power BI reports you created in this lab (`synapse-lab` and `synapse-sql-serverless`).



21. Select the `synapse-lab` report. You can view and edit the report directly within Synapse Studio!



22. Select the **synapse-sql-serverless** report. You should be able to view and edit this report as well.

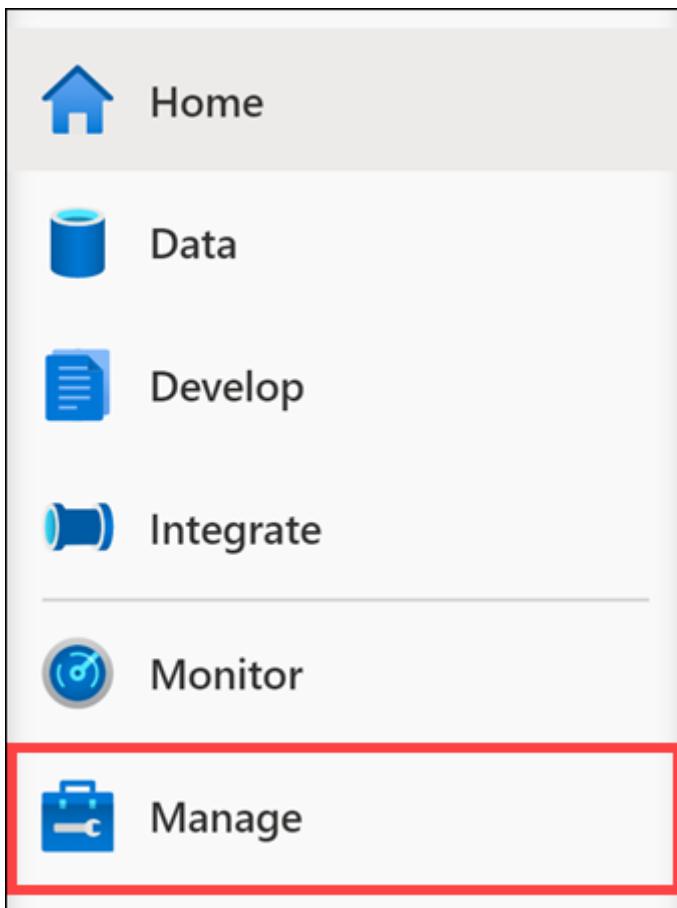


18.8 Exercise 4: Cleanup

Complete these steps to free up resources you no longer need.

18.8.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.

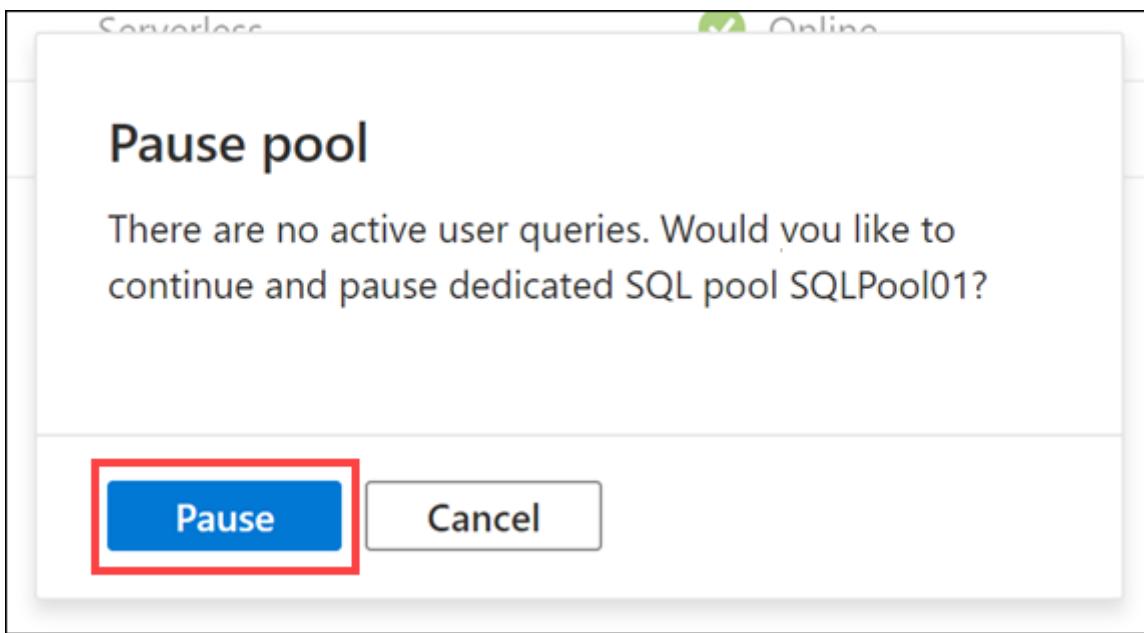


3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

The screenshot shows the 'SQL pools' blade in the Azure portal. On the left, there's a sidebar with 'Analytics pools' (selected), 'SQL pools' (highlighted with a red box and labeled 1), 'Apache Spark pools', 'External connections', 'Linked services', 'Azure Purview (Preview)', 'Integration', 'Triggers', 'Integration runtimes', and 'Security'. The main area is titled 'SQL pools' and contains the following text: 'The serverless SQL pool, Built-in, is immediately available for your workspace. Dedicated SQL pools can be configured.' Below this are buttons for '+ New', 'Refresh', and 'System-assigned managed identity'. A 'Filter by name' input field is also present. A table lists items: Name (Built-in, SQLPool01), Type (Serverless, Dedicated), and Status (Online, Online). The 'SQLPool01' row has a red box around its 'Pause' button (labeled 2).

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

4. When prompted, select **Pause**.



19 Module 17 - Perform integrated Machine Learning processes in Azure Synapse Analytics

This lab demonstrates the integrated, end-to-end Azure Machine Learning and Azure Cognitive Services experience in Azure Synapse Analytics. You will learn how to connect an Azure Synapse Analytics workspace to an Azure Machine Learning workspace using a Linked Service and then trigger an Automated ML experiment that uses data from a Spark table. You will also learn how to use trained models from Azure Machine Learning or Azure Cognitive Services to enrich data in a SQL pool table and then serve prediction results using Power BI.

After completing the lab, you will understand the main steps of an end-to-end Machine Learning process that build on top of the integration between Azure Synapse Analytics and Azure Machine Learning.

19.1 Lab details

- [Module 17 - Perform integrated Machine Learning processes in Azure Synapse Analytics](#)
 - [Lab details](#)
 - [Pre-requisites](#)
 - [Before the hands-on lab](#)
 - * [Task 1: Create and configure the Azure Synapse Analytics workspace](#)
 - * [Task 2: Create and configure additional resources for this lab](#)
 - [Exercise 0: Start the dedicated SQL pool](#)
 - [Exercise 1: Create an Azure Machine Learning linked service](#)
 - * [Task 1: Create and configure an Azure Machine Learning linked service in Synapse Studio](#)
 - * [Task 2: Explore Azure Machine Learning integration features in Synapse Studio](#)
 - [Exercise 2: Trigger an Auto ML experiment using data from a Spark table](#)
 - * [Task 1: Trigger a regression Auto ML experiment on a Spark table](#)
 - * [Task 2: View experiment details in Azure Machine Learning workspace](#)
 - [Exercise 3: Enrich data using trained models](#)
 - * [Task 1: Enrich data in a SQL pool table using a trained model from Azure Machine Learning](#)
 - * [Task 2: Enrich data in a Spark table using a trained model from Azure Cognitive Services](#)
 - * [Task 3: Integrate a Machine Learning-based enrichment procedure in a Synapse pipeline](#)
 - [Exercise 4: Serve prediction results using Power BI](#)
 - * [Task 1: Display prediction results in a Power BI report](#)
 - [Exercise 5: Cleanup](#)
 - * [Task 1: Pause the dedicated SQL pool](#)
 - [Resources](#)

19.2 Pre-requisites

Install Power BI Desktop on your lab computer or VM.

19.3 Before the hands-on lab

Note: Only complete the **Before the hands-on lab** steps if you are **not** using a hosted lab environment, and are instead using your own Azure subscription. Otherwise, skip ahead to Exercise 0.

Before stepping through the exercises in this lab, make sure you have properly configured your Azure Synapse Analytics workspace. Perform the tasks below to configure the workspace.

19.3.1 Task 1: Create and configure the Azure Synapse Analytics workspace

NOTE

If you have already created and configured the Synapse Analytics workspace while running one of the other labs available in this repo, you must not perform this task again and you can move on to the next task. The labs are designed to share the Synapse Analytics workspace, so you only need to create it once.

If you are not using a hosted lab environment, follow the instructions in [Deploy your Azure Synapse Analytics workspace](#) to create and configure the workspace.

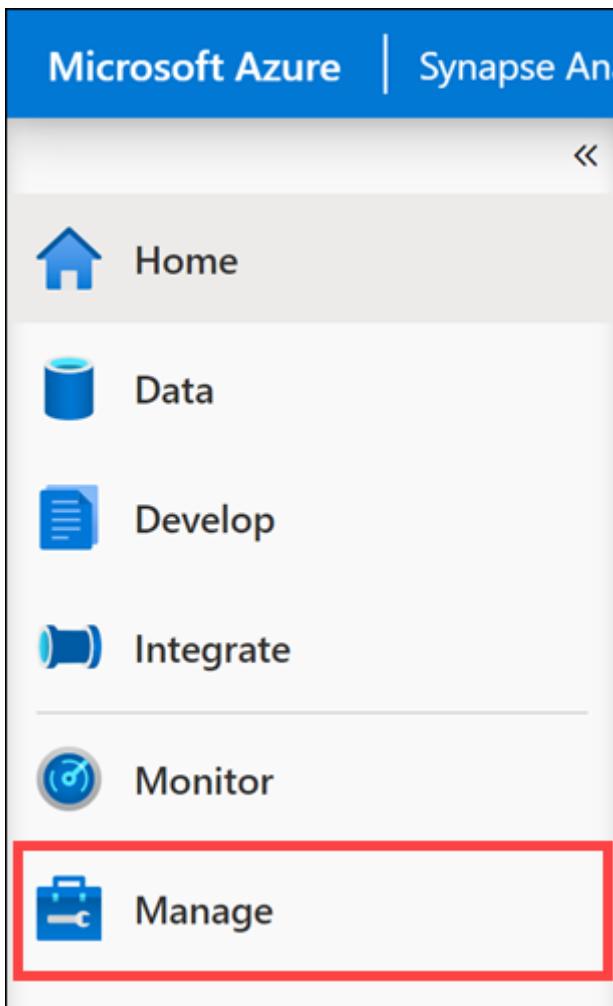
19.3.2 Task 2: Create and configure additional resources for this lab

If you are not using a hosted lab environment, follow the instructions in [Deploy resources for Lab 01](#) to deploy additional resources for this lab. Once deployment is complete, you are ready to proceed with the exercises in this lab.

19.4 Exercise 0: Start the dedicated SQL pool

This lab uses the dedicated SQL pool. As a first step, make sure it is not paused. If so, start it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the **Manage** hub.



3. Select **SQL pools** in the left-hand menu (1). If the dedicated SQL pool is paused, hover over the name of the pool and select **Resume** (2).

The screenshot shows the "SQL pools" blade in the Azure portal. The left sidebar shows:

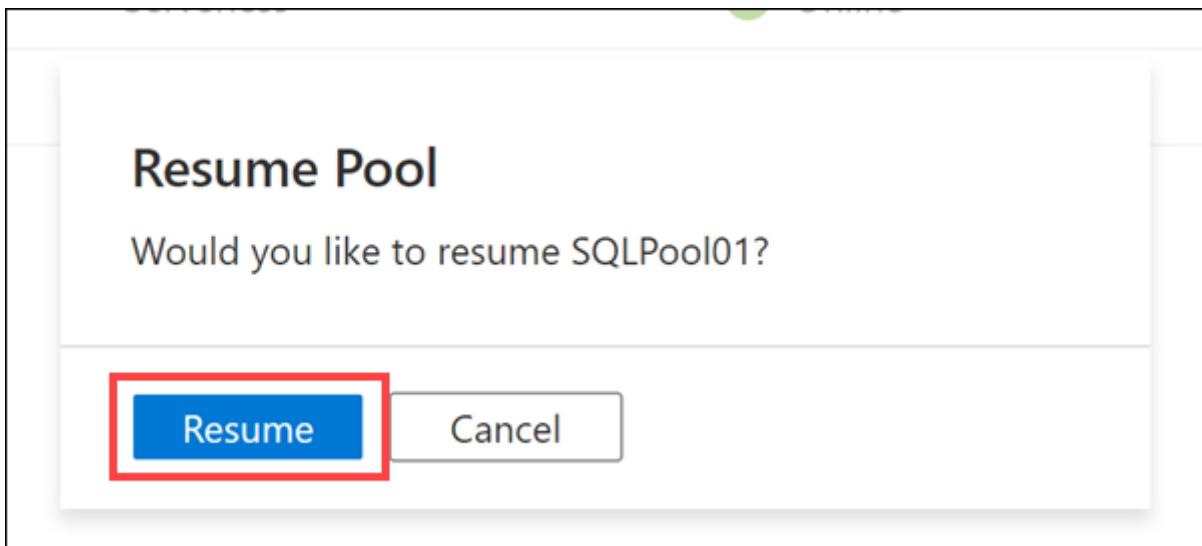
- Analytics pools
- SQL pools** (highlighted with a red box and a red number '1')
- Apache Spark pools
- External connections
- Linked services
- Azure Purview (Preview)
- Integration
- Triggers
- Integration runtimes
- Security
- Access control

The main area displays the "SQL pools" table:

Name	Type	Status	Size
Built-in	Serverless	Online	Auto
SQLPool01	Dedicated	Paused (highlighted with a red box and a red number '2')	DW100c

Buttons at the bottom include "New", "Refresh", and "System-assigned managed identity". A "Filter by name" input field is also present.

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.



Continue to the next exercise while the dedicated SQL pool resumes.

19.5 Exercise 1: Create an Azure Machine Learning linked service

In this exercise, you will create and configure an Azure Machine Learning linked service in Synapse Studio. Once the linked service is available, you will explore the Azure Machine Learning integration features in Synapse Studio.

19.5.1 Task 1: Create and configure an Azure Machine Learning linked service in Synapse Studio

The Synapse Analytics linked service authenticates with Azure Machine Learning using a service principal. The service principal is based on an Azure Active Directory application named **Azure Synapse Analytics GA Labs** and has already been created for you by the deployment procedure. The secret associated with the service principal has also been created and saved in the Azure Key Vault instance, under the **ASA-GA-LABS** name.

NOTE

In the labs provided by this repo, the Azure AD application is used in a single Azure AD tenant which means it has exactly one service principal associated to it. Consequently, we will use the terms Azure AD application and service principal interchangeably. For a detailed explanation on Azure AD applications and security principals, see [Application and service principal objects in Azure Active Directory](#).

1. To view the service principal, open the Azure portal and navigate to your instance of Azure Active directory. Select the **App registrations** section and you should see the **Azure Synapse Analytics GA Labs SUFFIX** (where SUFFIX is your unique suffix used during lab deployment) application under the **Owned applications** tab.

Display name	Application (client) ID	Created on	Certificates & secrets
AS Azure Synapse Analytics GA Labs 297032	f6a4b9b8-224d-405d-ad23-44e2c3cb5bb0	2/1/2021	Current

2. Select the application to view its properties and copy the value of the Application (client) ID property (you will need it in a moment to configure the linked service).

Azure Synapse Analytics GA Labs 297032

Search (Ctrl+ /) Delete Endpoints Preview features

Overview Quickstart Integration assistant

Display name : Azure Synapse Analytics GA Labs 297032

Application (client) ID :

Directory (tenant) ID : Object ID :

Starting June 30th, 2020 we will no longer add any new features to Azure AD. Microsoft will no longer provide feature updates. Applications will need to be upgraded.

Call APIs

3. To view the secret, open the Azure Portal and navigate to the Azure Key Vault instance that has been created in your resource group. Select the **Secrets** section and you should see the ASA-GA-LABS secret:

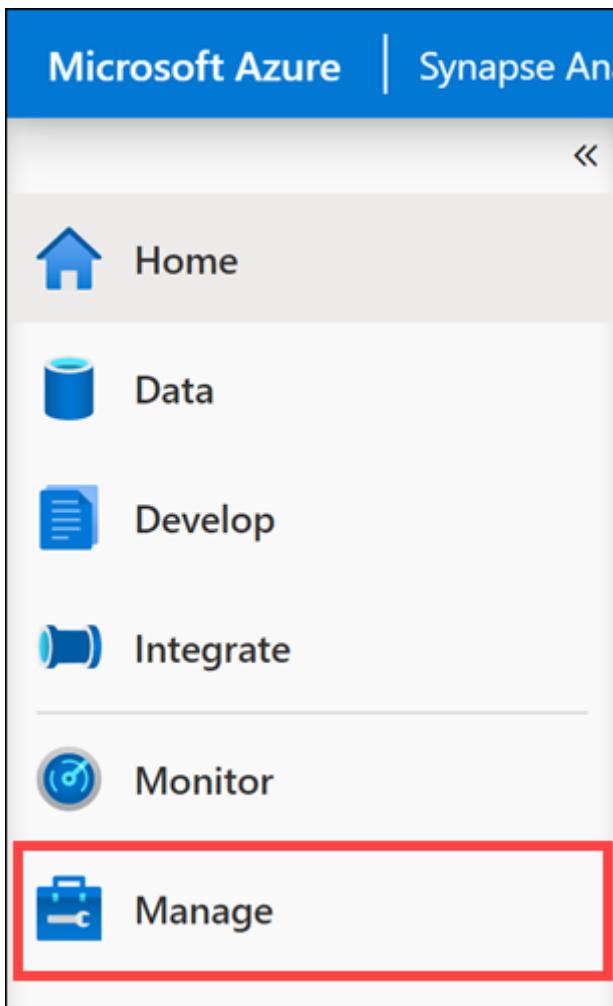
Name	Type	Status
ASA-GA-COGNITIVE-SERVICES		✓ Enabled
ASA-GA-LABS		✓ Enabled
d8f2df2f-50f2-4859-8ad4-dcf...		✓ Enabled
d8f2df2f-50f2-4859-8ad4-dcf...		✓ Enabled
SQL-USER-ASA	text/plain	✓ Enabled

4. First, you need to make sure the service principal has permissions to work with the Azure Machine Learning workspace. Open the Azure Portal and navigate to the Azure Machine Learning workspace that has been created in your resource group. Select the **Access control (IAM)** section on the left, then select **+ Add** and **Add role assignment**. In the **Add role assignment** dialog, select the **Contributor** role, select **Azure Synapse Analytics GA Labs SUFFIX** (where SUFFIX is your unique suffix used during lab deployment) service principal, and the select **Save**.

The screenshot shows the Azure portal interface for managing access control (IAM) for a specific resource. On the left, there's a sidebar with various navigation options like Overview, Activity log, and Access control (IAM). The main area displays the 'Access control (IAM)' blade for the resource 'asagamachinelearning356342'. A modal window titled 'Add role assignment' is open, allowing the user to assign a 'Contributor' role to the service principal 'azure synapse'. The 'Save' button at the bottom of the modal is highlighted with a red box.

You are now ready to create the Azure Machine Learning linked service.

5. Open Synapse Studio (<https://web.azuresynapse.net/>).
6. Select the **Manage** hub.



7. Select **Linked services**, and then select **+ New**. In the search field from the **New linked service** dialog, enter **Azure Machine Learning**. Select the **Azure Machine Learning** option and then select **Continue**.

The screenshot shows the 'New linked service' dialog. On the left, the sidebar has a red box around the 'Linked services' option under 'Integration'. Step 1 points to this. Step 2 points to the '+ New' button next to the search bar. Step 3 points to the search bar where 'Azure Machine Learning' is typed. Step 4 points to the 'Azure Machine Learning' icon in the list. Step 5 points to the 'Continue' button at the bottom right.

8. In the **New linked service (Azure Machine Learning)** dialog, provide the following properties:
- Name: enter **asagamachinelearning01**.

- Azure subscription: make sure the Azure subscription containing your resource group is selected.
- Azure Machine Learning workspace name: make sure your Azure Machine Learning workspace is selected.
- Notice how `Tenant identifier` has been already filled in for you.
- Service principal ID: enter the application client ID that you copied earlier.
- Select the `Azure Key Vault` option.
- AKV linked service: make sure your Azure Key Vault service is selected.
- Secret name: enter `ASA-GA-LABS`.

New linked service (Azure Machine Learning)

i Choose a name for your linked service. This name cannot be updated later.

Name *

asagamachinelearning01

Description

Connect via integration runtime * ⓘ

AutoResolveIntegrationRuntime



Azure Machine Learning workspace selection method ⓘ

From Azure subscription Enter manually

Azure subscription ⓘ

Select all

Azure Machine Learning workspace name *

asagamachinelearning356342

Tenant *

[REDACTED]

Service principal ID *

d7f0dfb6-[REDACTED]

Service principal key

Azure Key Vault

AKV linked service * ⓘ

asagakeyvault356342



Secret name * ⓘ

ASA-GA-LABS

Secret version ⓘ

Use the latest version if left blank

Connection successful

Test connection

Cancel

Create

Back

9. Next, select **Test connection** to make sure all settings are correct, and then select **Create**. The Azure Machine Learning linked service will now be created in the Synapse Analytics workspace.

IMPORTANT

The linked service is not complete until you publish it to the workspace. Notice the indicator

near your Azure Machine Learning linked service. To publish it, select **Publish all** and then **Publish**.

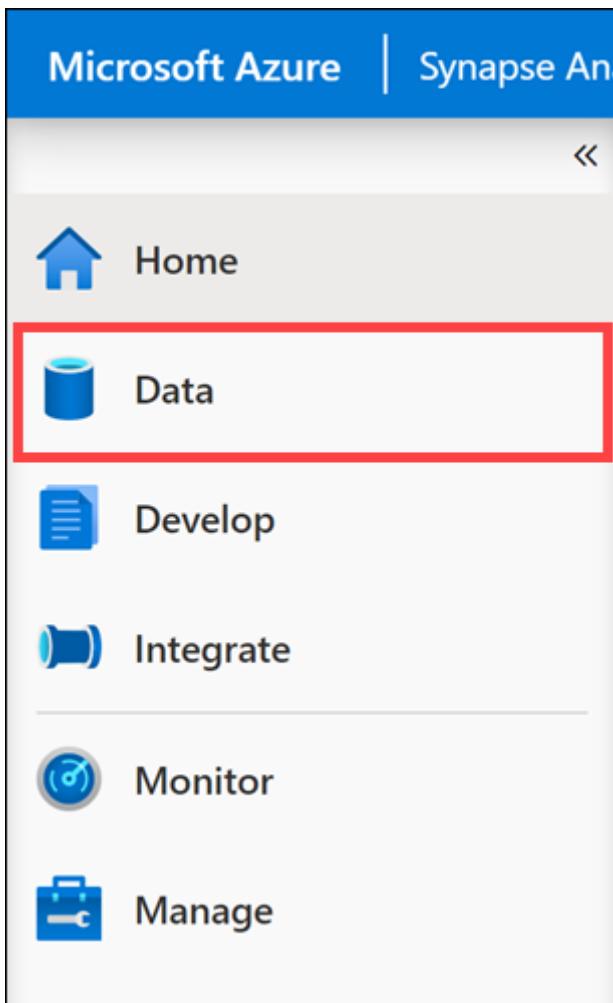
The screenshot shows the Azure Synapse Analytics studio interface. On the left, there's a sidebar with various icons and sections: Home, Analytics pools, SQL pools, Apache Spark pools, External connections, Linked services (which is currently selected and highlighted with a red box), Integration, Triggers, Integration runtimes, Security, Access control, Credentials, and Managed private endpoints. At the top, there are three buttons: 'Validate all' (with a checkmark icon), 'Publish all' (with a blue arrow icon and a red box around it), and 'Discard all'. Below the sidebar, the main area is titled 'Linked services' with the sub-instruction: 'Linked services are much like connection strings, which define the connection information needed for Azure Synapse Analytics to connect to external systems.' There's a '+ New' button and a 'Filter by keyword' input field. A 'Annotations : Any' button is also present. The table below lists seven items:

Name	Type
asagadatalake01	Azure Data Lake Storage Gen2
asagakeyvault01	Azure Key Vault
asagamachinelearning01	Azure Machine Learning
asagaworkspace01-WorkspaceDefaultSqlServer	Azure Synapse Analytics (formerly SQL DW)
asagaworkspace01-WorkspaceDefaultStorage	Azure Data Lake Storage Gen2
sqlpool01	Azure Synapse Analytics (formerly SQL DW)
sqlpool01_highperf	Azure Synapse Analytics (formerly SQL DW)

19.5.2 Task 2: Explore Azure Machine Learning integration features in Synapse Studio

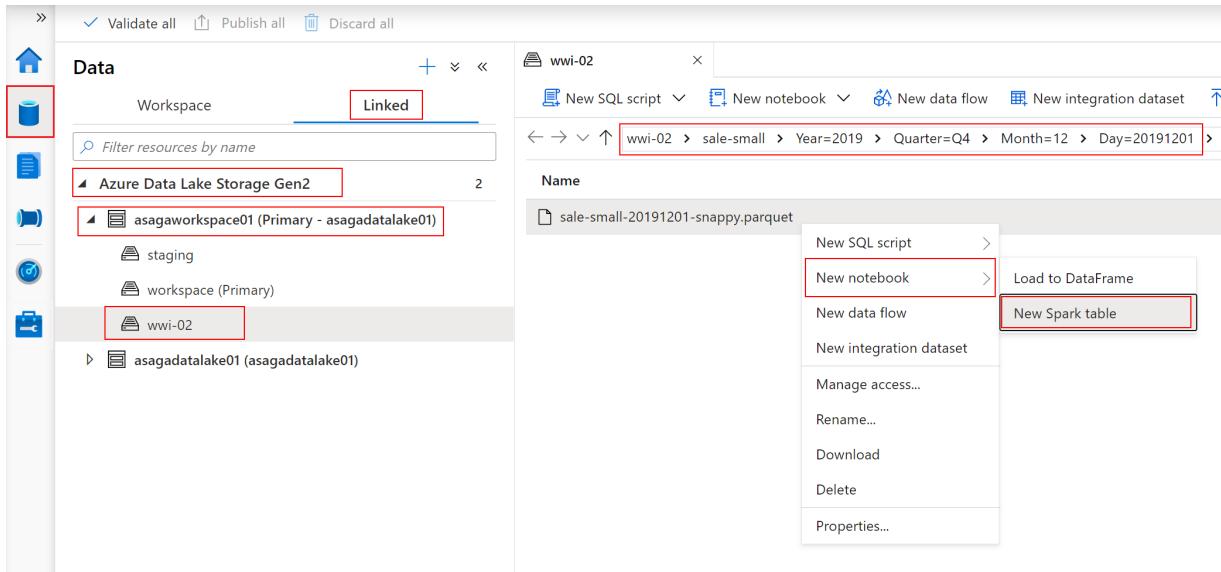
First, we need to create a Spark table as a starting point for the Machine Learning model training process.

1. Select the **Data** hub.



2. Select the **Linked** tab
3. In the primary **Azure Data Lake Storage Gen 2** account, select the **wwi-02** file system, and then select

the `sale-small-20191201-snappy.parquet` file under `wwi-02\sale-small\Year=2019\Quarter=Q4\Month=12\Day=20191201`. Right click the file and select New notebook -> New Spark table.



4. Attach your Spark cluster to the notebook and ensure the language is set to PySpark (Python).



5. Replace the content of the notebook cell with the following code and then run the cell:

```
import pyspark.sql.functions as f

df = spark.read.load('abfss://wwi-02@<data_lake_account_name>.dfs.core.windows.net/sale-small/Year
                     format='parquet')
df_consolidated = df.groupBy('ProductId', 'TransactionDate', 'Hour').agg(f.sum('Quantity')).alias('TotalQuantity')
df_consolidated.write.mode("overwrite").saveAsTable("default.SaleConsolidated")
```

NOTE:

Replace `<data_lake_account_name>` with the actual name of your Synapse Analytics primary data lake account.

The code takes all data available for December 2019 and aggregates it at the `ProductId`, `TransactionDate`, and `Hour` level, calculating the total product quantities sold as `TotalQuantity`. The result is then saved as a Spark table named `SaleConsolidated`. To view the table in the Datahub, expand the `default` (Spark) database in the `Workspace` section. Your table will show up in the `Tables` folder. Select the three dots at the right of the table name to view the `Machine Learning` option in the context menu.

The screenshot shows the Azure Machine Learning studio interface. On the left, the Data hub is open, displaying databases and tables. In the center, a notebook titled 'wwi-02' is running, with Cell 1 containing Python code for reading a parquet file from a Databricks file system and writing it back to a table. A context menu is open over the 'saleconsolidated' table in the 'default (Spark)' database. The menu items 'Machine Learning' and 'Enrich with new model' are highlighted with red boxes.

The following options are available in the **Machine Learning** section:

- Enrich with new model: allows you to start an AutoML experiment to train a new model.
- Enrich with existing model: allows you to use an existing Azure Cognitive Services model.

19.6 Exercise 2: Trigger an Auto ML experiment using data from a Spark table

In this exercise, you will trigger the execution of an Auto ML experiment and view its progress in Azure Machine learning studio.

19.6.1 Task 1: Trigger a regression Auto ML experiment on a Spark table

1. To trigger the execution of a new AutoML experiment, select the Data hub and then select the ... area on the right of the **saleconsolidated** Spark table to activate the context menu.

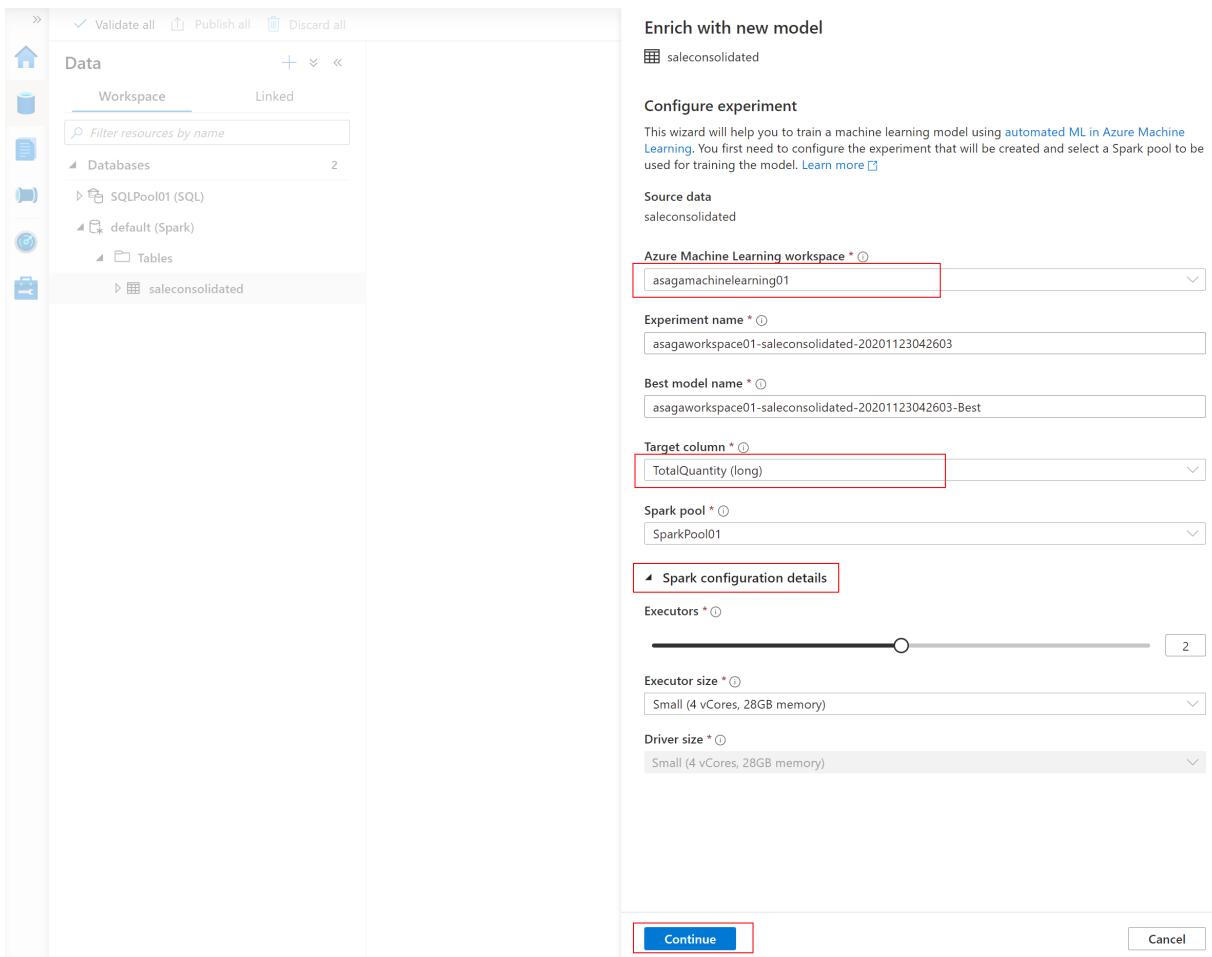
The screenshot shows the Azure Machine Learning studio interface. On the left, the Data hub is open, displaying databases and tables. In the center, a context menu is open over the 'saleconsolidated' table in the 'default (Spark)' database. The 'Actions' option in the menu is highlighted with a red box.

2. From the context menu, select **Enrich with new model**.

The screenshot shows the Azure Data Studio interface. On the left, the 'Data' sidebar is open, showing 'Workspace' selected. Under 'Databases', 'default (Spark)' is selected, and under 'Tables', 'saleconsolidated' is selected. In the main area, 'Cell 1' contains a Python script for reading data from a parquet file and writing it back to the same table. A context menu is open over the table name 'saleconsolidated', with the 'Machine Learning' option selected. A sub-menu appears with two options: 'Enrich with existing model' and 'Enrich with new model', both of which are highlighted with red boxes.

The **Enrich with new model** dialog allow you to set the properties for the Azure Machine Learning experiment. Provide values as follows:

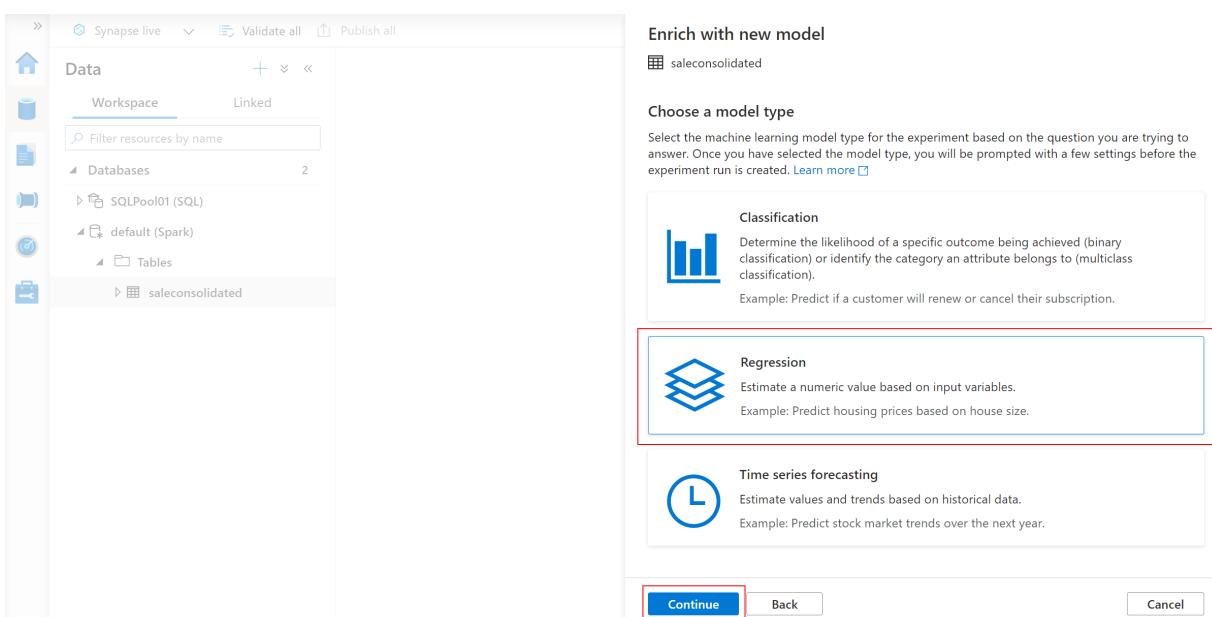
- **Azure Machine Learning workspace:** leave unchanged, should be automaticall populated with your Azure Machine Learning workspace name.
- **Experiment name:** leave unchanged, a name will be automatically suggested.
- **Best model name:** leave unchanged, a name will be automatically suggested. Save this name as you will need it later to identify the model in the Azure Machine Learning Studio.
- **Target column:** Select `TotalQuantity(long)` - this is the feature you are looking to predict.
- **Spark pool:** leave unchanged, should be automaticall populated with your Spark pool name.



Notice the Apache Spark configuration details:

- The number of executors that will be used
- The size of the executor

3. Select **Continue** to advance with the configuration of your Auto ML experiment.
4. Next, you will choose the model type. In this case, the choice will be **Regression** as we try to predict a continuous numerical value. After selecting the model type, select **Continue** to advance.

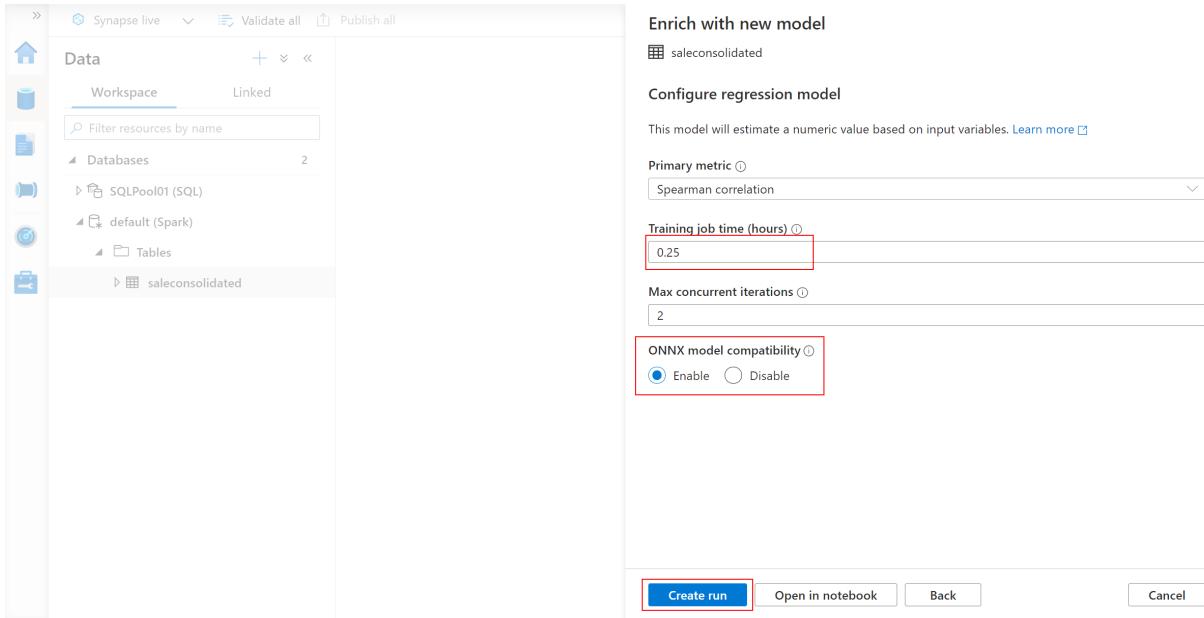


5. On the **Configure regression model** dialog, provide values as follows:

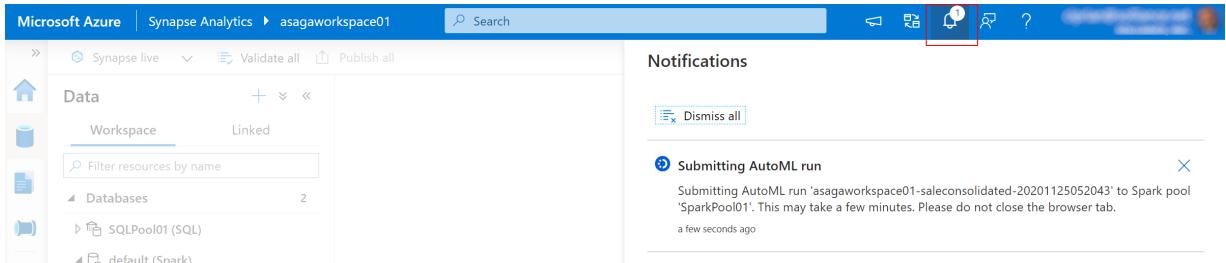
- **Primary metric:** leave unchanged, **Spearman correlation** should be suggested by default.

- **Training job time (hours)**: set to 0.25 to force the process to finish after 15 minutes.
- **Max concurrent iterations**: leave unchanged.
- **ONNX model compatibility**: set to **Enable** - this is very important as currently only ONNX models are supported in the Synapse Studio integrated experience.

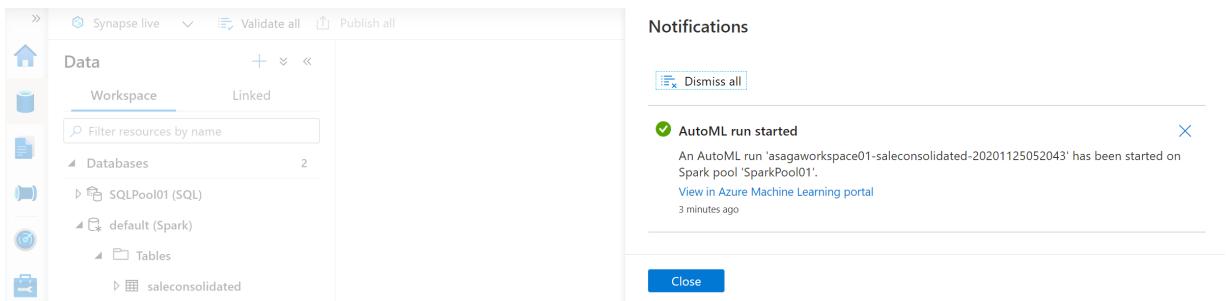
6. Once you have set all the values, select **Create run** to advance.



As your run is being submitted, a notification will pop up instructing you to wait until the Auto ML run is submitted. You can check the status of the notification by selecting the **Notifications** icon on the top right part of your screen.



Once your run is successfully submitted, you will get another notification that will inform you about the actual start of the Auto ML experiment run.



NOTE

Alongside the **Create run** option you might have noticed the **Open in notebook** option. Selecting that option allows you to review the actual Python code that is used to submit the Auto ML run. As an exercise, try re-doing all the steps in this task, but instead of selecting **Create run**, select **Open in notebook**. You should see a notebook similar to this:

```

import azurerm.core
from azurerm.core import Experiment, Workspace, Dataset, Datastore
from azurerm.train.automl import AutoMLConfig
from azurerm.data.dataset_factory import TabularDatasetFactory

subscription_id = "6cc6a5ec-04d2-4144-9820-5e33354170b5"
resource_group = "SynapseAnalytics-CA"
workspace_name = "asageworkspace356342"
experiment_name = "asageworkspace356342-saleconsolidated-20201125054045"
ws = Workspace(subscription_id=subscription_id, resource_group=resource_group, workspace_name=workspace_name)
experiment = Experiment(ws, experiment_name)

df = spark.sql("SELECT * FROM default.saleconsolidated")
datastore = Datastore.get_default(ws)
dataset = TabularDatasetFactory.register_spark_dataframe(df, datastore, name=experiment_name + ".dataset")

automl_config = AutoMLConfig(spark_context = sc,
                             task = "classification",
                             training_data = dataset,
                             label_column_name = "TotalQuantity",
                             primary_metric = "spearman_correlation",
                             experiment_timeout_hours = 0.25,
                             max_concurrent_iterations = 2,
                             enable_omni_compatible_models = True)

run = experiment.submit(automl_config)

displayHTML("<a href={} target='_blank'>Your experiment in Azure Machine Learning portal: {}</a>".format(run.get_portal_url(), run.id))

```

Take a moment to read through the code that is generated for you.

19.6.2 Task 2: View experiment details in Azure Machine Learning workspace

- To view the experiment run you just started, open the Azure Portal, select your resource group, and then select the Azure Machine Learning workspace from the resource group.

Showing 1 to 13 of 13 records. <input type="checkbox"/> Show hidden types ⓘ		No g
<input type="checkbox"/> Name ↑↓	Type ↑↓	
<input type="checkbox"/> asagacognitiveservices356342	Cognitive Services	
<input type="checkbox"/> asagacontainerregistry356342	Container registry	
<input type="checkbox"/> asagadataexpl356342	Azure Data Explorer Cluster	
<input type="checkbox"/> asagatalake356342	Storage account	
<input type="checkbox"/> asagakeyvault356342	Key vault	
<input type="checkbox"/> asagamachinelearning356342	Machine learning	
<input type="checkbox"/> asagastorage356342	Storage account	
<input type="checkbox"/> asagaworkspace356342	Synapse workspace	
<input type="checkbox"/> databricksdemoeventhubs356342	Event Hubs Namespace	
<input type="checkbox"/> SparkPool01 (asagaworkspace356342/SparkPool01)	Apache Spark pool	
<input type="checkbox"/> SQLPool01 (asagaworkspace356342/SQLPool01)	Dedicated SQL pool	

- Locate and select the **Launch studio** button to start Azure Machine Learning Studio.

The screenshot shows the 'Overview' page of an Azure Machine Learning workspace. On the left, there's a sidebar with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings, Private endpoint connections, Properties, Locks, and Monitoring. The main area has a 'Essentials' section with details such as Resource group, Location, Subscription, Studio web URL, Storage, Registry, Key Vault, and Application Insights. Below this is a section titled 'Manage your machine learning lifecycle' with a 'Launch studio' button, which is highlighted with a red box.

- In Azure Machine Learning Studio, select the **Automated ML** section on the left and identify the experiment run you have just started. Note the experiment name, the **Running status**, and the **local** compute target.

The screenshot shows the 'Automated ML' section. It displays a table of recent runs. The columns include Run, Run ID, Experiment, Status, Submitted time, Duration, Submitted by, Compute target, and Tags. One row is highlighted, showing 'Run 1' with 'AutoML_6573db19-17ef-46ce-b091-df0...', 'Experiment asagaworkspace356342-saleconsolidated-20210430045928', 'Status Running', 'Submitted time Apr 30, 2021 1:55 PM', 'Duration 6m 57s', 'Submitted by ODL_User 356...', and 'Compute target local'. The 'Compute target' column is highlighted with a red box.

The reason why you see **local** as the compute target is because you are running the AutoML experiment on the Spark pool inside Synapse Analytics. From the point of view of Azure Machine Learning, you are not running your experiment on Azure Machine Learning's compute resources, but on your "local" compute resources.

- Select your run, and then select the **Models** tab to view the current list of models being built by your run. The models are listed in descending order of the metric value (which is **Spearman correlation** in this case), the best ones being listed first.

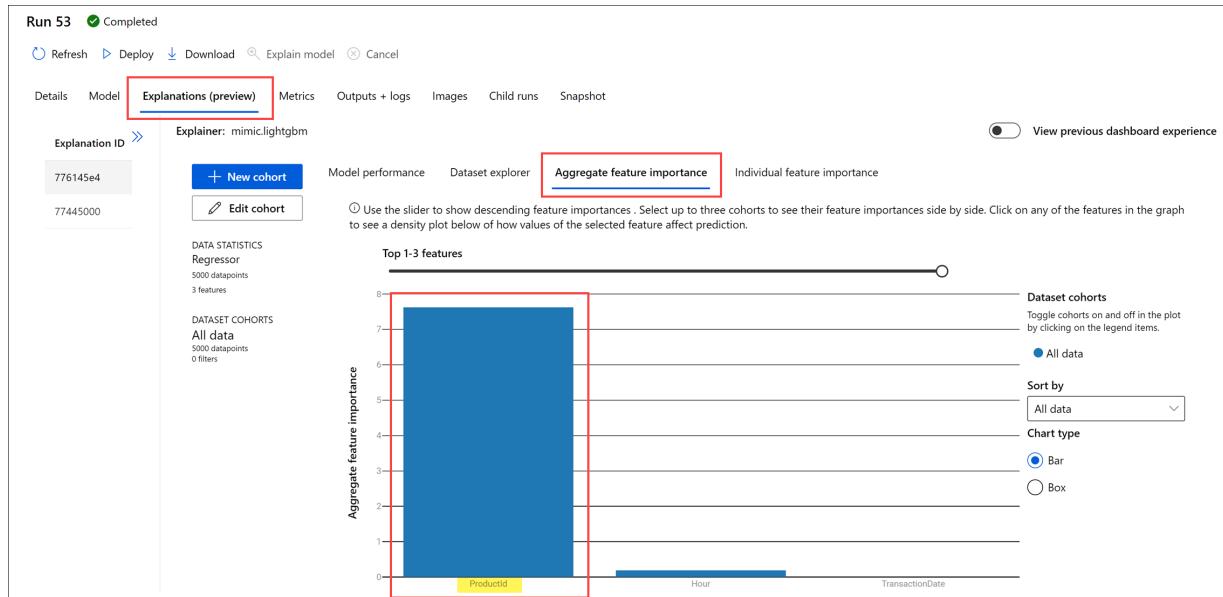
The screenshot shows the details of 'Run 1'. It includes a navigation bar with Home, Automated ML, and the specific run ID. Below this is a summary section for 'Run 1' (Running) with Refresh and Cancel buttons. The 'Models' tab is selected and highlighted with a red box. Underneath are buttons for Deploy, Download, and Explain model. The main area is a table of models, with the 'Models' tab also highlighted with a red box. The columns are Algorithm name, Explained, Spearman correlation (sorted by降序), Sampling, Created, and Duration. The table lists several models, each with a 'View explanation' link. The top model has a Spearman correlation of 0.53343.

Algorithm name	Explained	Spearman correlation ↓	Sampling ⓘ	Created	Duration
StandardScalerWrapper, XGBoostRegressor	View explanation	0.53343	100.00 %	Apr 30, 2021 2:05 PM	12s
StandardScalerWrapper, XGBoostRegressor	View explanation	0.51993	100.00 %	Apr 30, 2021 2:05 PM	14s
StandardScalerWrapper, XGBoostRegressor	View explanation	0.51808	100.00 %	Apr 30, 2021 2:00 PM	14s
StandardScalerWrapper, XGBoostRegressor	View explanation	0.49974	100.00 %	Apr 30, 2021 2:00 PM	11s
MaxAbsScaler, RandomForest	View explanation	0.45735	100.00 %	Apr 30, 2021 1:58 PM	11s
StandardScalerWrapper, XGBoostRegressor	View explanation	0.44157	100.00 %	Apr 30, 2021 2:02 PM	11s
StandardScalerWrapper, XGBoostRegressor	View explanation	0.43706	100.00 %	Apr 30, 2021 2:02 PM	10s
StandardScalerWrapper, GradientBoosting	View explanation	0.42192	100.00 %	Apr 30, 2021 2:01 PM	35s
MaxAbsScaler, RandomForest	View explanation	0.38909	100.00 %	Apr 30, 2021 2:01 PM	12s
StandardScalerWrapper, XGBoostRegressor	View explanation	0.38765	100.00 %	Apr 30, 2021 1:57 PM	12s
StandardScalerWrapper, DecisionTree	View explanation	0.38365	100.00 %	Apr 30, 2021 1:58 PM	7s
StandardScalerWrapper, XGBoostRegressor	View explanation	0.37059	100.00 %	Apr 30, 2021 1:58 PM	10s

- Select the best model (the one at the top of the list) then click on **View Explanations** to open the

Explanations (preview) tab to see the model explanation.

6. Select the **Aggregate feature importance** tab. You are now able to see the global importance of the input features. For your model, the feature that influences the most the value of the predicted value is **ProductId**.



7. Next, select the **Models** section on the left in Azure Machine Learning Studio and see your best model registered with Azure Machine Learning. This allows you to refer to this model later on in this lab.

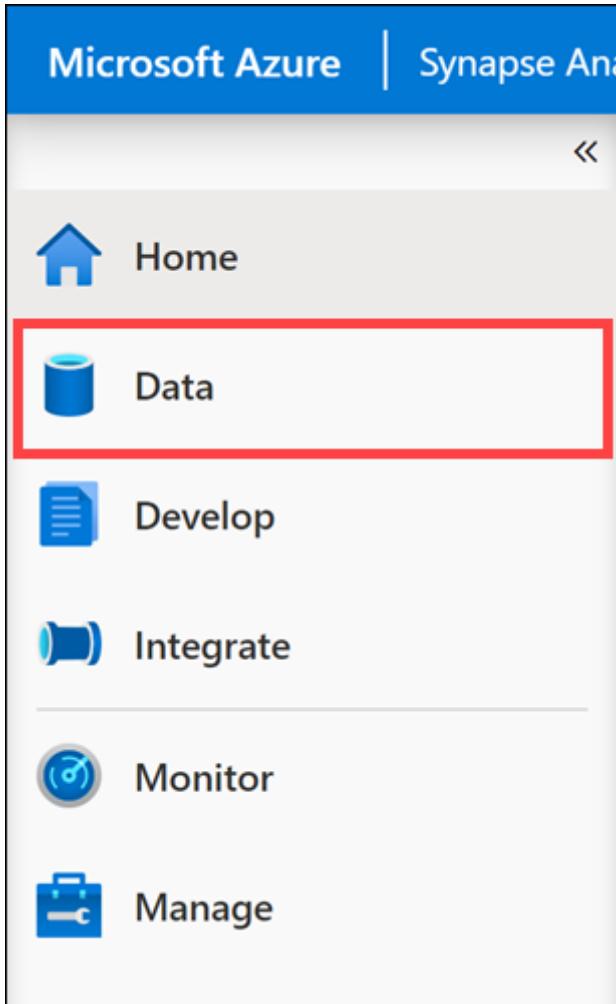
The screenshot shows the 'Models' section in the left sidebar. The main area is titled 'Model List' and displays a single model entry. The columns are 'Name', 'Version', and 'Experiment'. The 'Name' column contains the text 'asagaworkspace356342-saleco...' which is highlighted with a red box. The 'Version' column shows the value '1'. The 'Experiment' column is partially visible. At the top of the page, there are buttons for 'Register model', 'Delete', 'Deploy', 'Edit columns', and 'Refresh', with 'Refresh' also highlighted with a red box. On the far left, there is a vertical sidebar with several icons: a list icon, a gear icon, a cube icon (highlighted with a red box), and a cloud icon.

19.7 Exercise 3: Enrich data using trained models

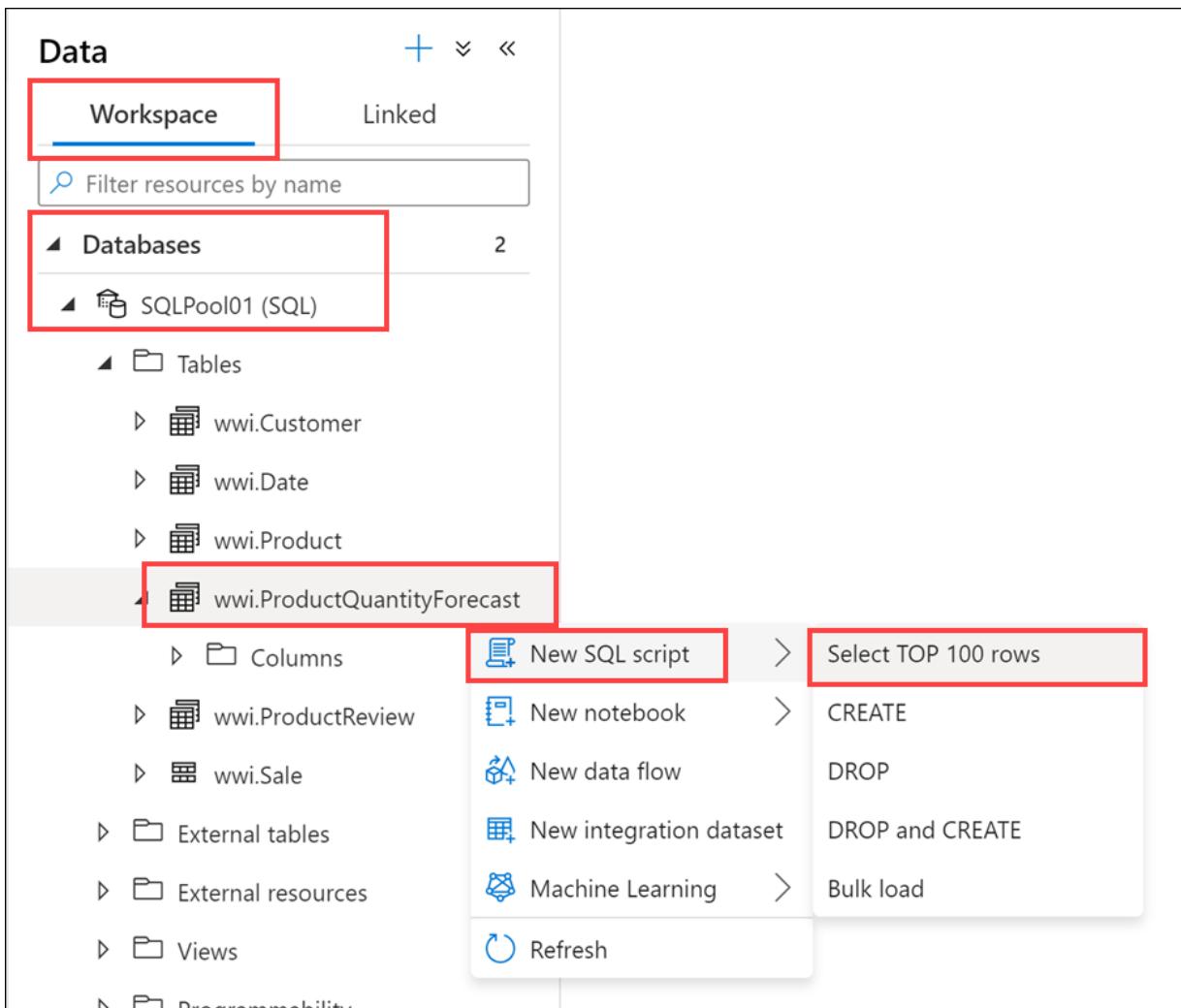
In this exercise, you will use existing trained models to perform predictions on data. Task 1 uses a trained model from Azure Machine Learning services while Task 2 uses one from Azure Cognitive Services. Finally, you will include the predictiton stored procedure created in Task 1 into a Synapse pipeline.

19.7.1 Task 1: Enrich data in a SQL pool table using a trained model from Azure Machine Learning

1. Switch back to Synapse Studio and select the **Data** hub.

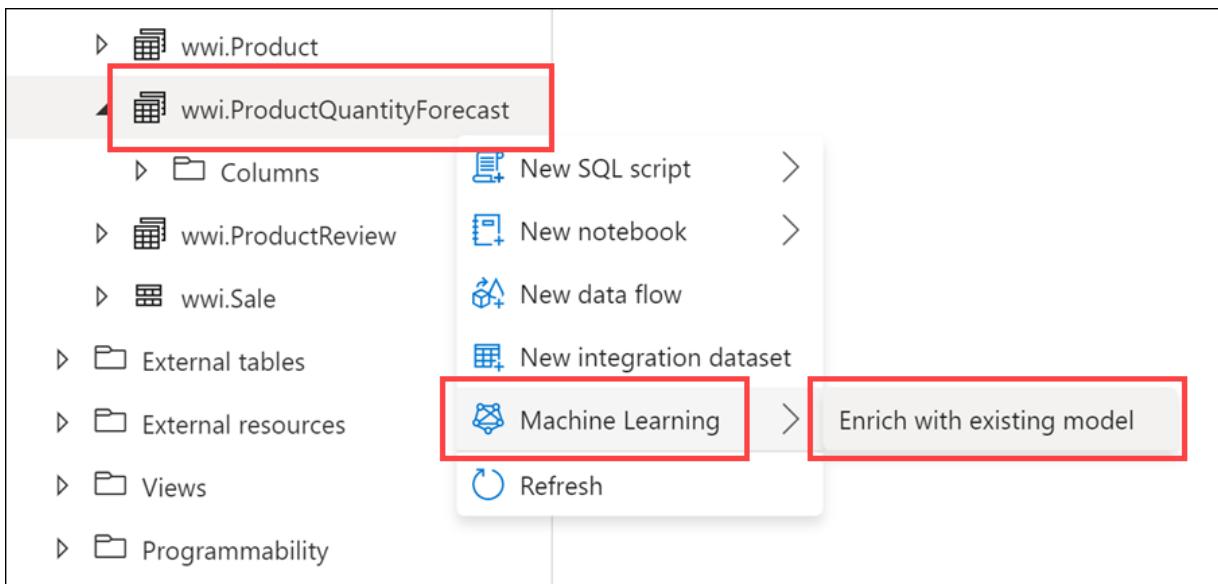


2. Select the **Workspace** tab, and then locate the `wwi.ProductQuantityForecast` table in the `SQLPool01` (SQL) database (under **Databases**). Activate the context menu by selecting **... from the right side of the table name**, and then select **New SQL script > Select TOP 100 rows**. The table contains the following columns:
 - **ProductId**: the identifier of the product for which we want to predict
 - **TransactionDate**: the future date for which we want to predict
 - **Hour**: the hour from the future date for which we want to predict
 - **TotalQuantity**: the value we want to predict for the specified product, day, and hour.



Notice that TotalQuantity is zero in all rows as this is the placeholder for the predicted values we are looking to get.

3. To use the model you just trained in Azure Machine Learning, activate the context menu of the `wwi.ProductQuantityForecast`, and then select `Machine Learning > Enrich with existing model`.



4. This will open the `Enrich with existing model` dialog where you can select your model. Select the most recent model and then select `Continue`.

Enrich with existing model

 wwi.ProductQuantityForecast

Select the model you want to use to enrich the selected dataset. [Learn more](#)

Azure Machine Learning

Azure Machine Learning workspace *

asagamachinelearning356342

Name	Versi...	Created	Created by	Fra...
asagaworkspace356342-saleconsolidate...	1	18:14:50 04/3...	ODL_User ...	Cust...

Continue

Cancel

5. Next, you will manage the input and output column mappings. Because the column names from the target table and the table used for model training match, you can leave all mappings as suggested by default. Select **Continue** to advance.

Enrich with existing model

wwi.ProductQuantityForecast

Map the source table columns to the expected model inputs. [Learn more](#)

Input mapping *

[+ New](#) | [Delete](#)

<input type="checkbox"/>	Source column	Model input	Input type	+	Delete
<input type="checkbox"/>	ProductId	ProductId	bigint	+	Delete
<input type="checkbox"/>	Hour	Hour	bigint	+	Delete

Output mapping *

[+ New](#) | [Delete](#)

<input type="checkbox"/>	Model output	Output type	+	Delete
<input type="checkbox"/>	variable_out1	real	+	Delete

[Continue](#)

[Back](#)

[Cancel](#)

6. The final step presents you with options to name the stored procedure that will perform the predictions and the table that will store the serialized form of your model. Provide the following values:

- **Stored procedure name:** [wwi].[ForecastProductQuantity]
- **Select target table:** Create new
- **New table:** [wwi].[Model]

Select **Deploy model + open script** to deploy your model into the SQL pool.

Enrich with existing model

wwi.ProductQuantityForecast

Stored procedure

A stored procedure will be created once you run the generated script. Specify a name for this stored procedure. [Learn more](#)

Stored procedure name *

[wwi].[ForecastProductQuantity]

Target table

Create a new table or use an existing table to store the model. [Learn more](#)

Select target table *

Existing table Create new

New table *

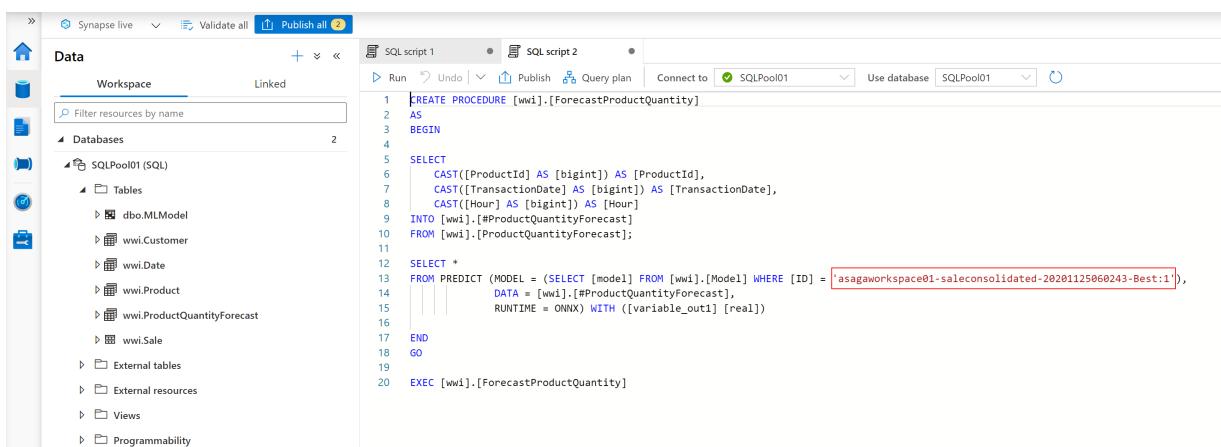
[wwi].[Model]

Deploy model + open script

Back

Cancel

- From the new SQL script that is created for you, copy the ID of the model:



```
CREATE PROCEDURE [wwi].[ForecastProductQuantity]
AS
BEGIN
    SELECT *
    FROM PREDICT (MODEL = (SELECT [model] FROM [wwi].[Model] WHERE [ID] = 'asagaworkspace01-saleconsolidated-20201125060243-Best:1'),
    DATA = [wwi].[#ProductQuantityForecast],
    RUNTIME = ONNX) WITH ([variable_out1] [real])
END
GO
EXEC [wwi].[ForecastProductQuantity]
```

- The T-SQL code that is generated will only return the results of the prediction, without actually saving them. To save the results of the prediction directly into the [wwi].[ProductQuantityForecast] table, replace the generated code with the following, then execute the script (after replacing <your_model_id>):

```
CREATE PROC [wwi].[ForecastProductQuantity] AS
BEGIN

    SELECT
        CAST([ProductId] AS [bigint]) AS [ProductId],
        CAST([TransactionDate] AS [bigint]) AS [TransactionDate],
        CAST([Hour] AS [bigint]) AS [Hour]
    INTO #ProductQuantityForecast
    FROM [wwi].[ProductQuantityForecast]
```

```

WHERE TotalQuantity = 0;

SELECT
    ProductId
    , TransactionDate
    , Hour
    , CAST(variable_out1 as INT) as TotalQuantity
INTO
    #Pred
FROM PREDICT (MODEL = (SELECT [model] FROM wwi.Model WHERE [ID] = '<your_model_id>'),
    DATA = #ProductQuantityForecast,
    RUNTIME = ONNX) WITH ([variable_out1] [real])

MERGE [wwi].[ProductQuantityForecast] AS target
    USING (select * from #Pred) AS source (ProductId, TransactionDate, Hour, TotalQuantity)
ON (target.ProductId = source.ProductId and target.TransactionDate = source.TransactionDate and target.Hour = source.Hour)
WHEN MATCHED THEN
    UPDATE SET target.TotalQuantity = source.TotalQuantity;
END
GO

```

In the code above, make sure you replace <your_model_id> with the actual ID of the model (the one you copied in the previous step).

NOTE:

Our version of the stored procedure uses the MERGE command to update the values of the TotalQuantity field in-place, in the wwi.ProductQuantityForecast table. The MERGE command has been recently added to Azure Synapse Analytics. For more details, read [New MERGE command for Azure Synapse Analytics](#).

9. You are now ready to perform the forecast on the TotalQuantity column. Replace the SQL script and run the following statement:

```

EXEC
    wwi.ForecastProductQuantity
SELECT
    *
FROM
    wwi.ProductQuantityForecast

```

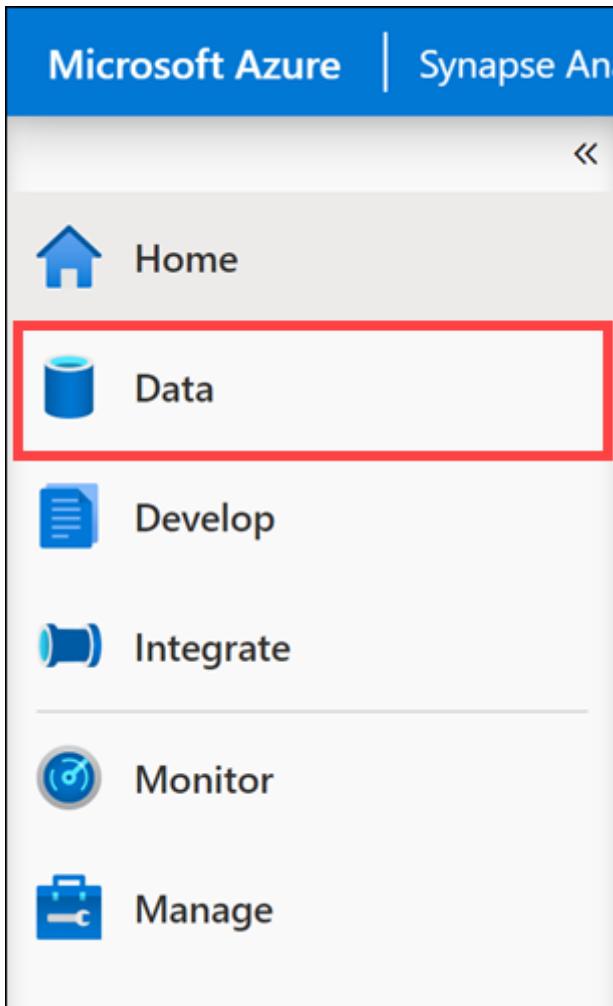
Notice how the values in the TotalQuantity column have changed from zero to non-zero predicted values:

ProductId	TransactionDate	Hour	TotalQuantity
1000	20201209	10	10
600	20201209	10	11
1100	20201209	10	9
500	20201209	10	12
100	20201209	10	90
400	20201209	10	11
900	20201209	10	12
800	20201209	10	13
300	20201209	10	9
200	20201209	10	91
1200	20201209	10	10
700	20201209	10	10

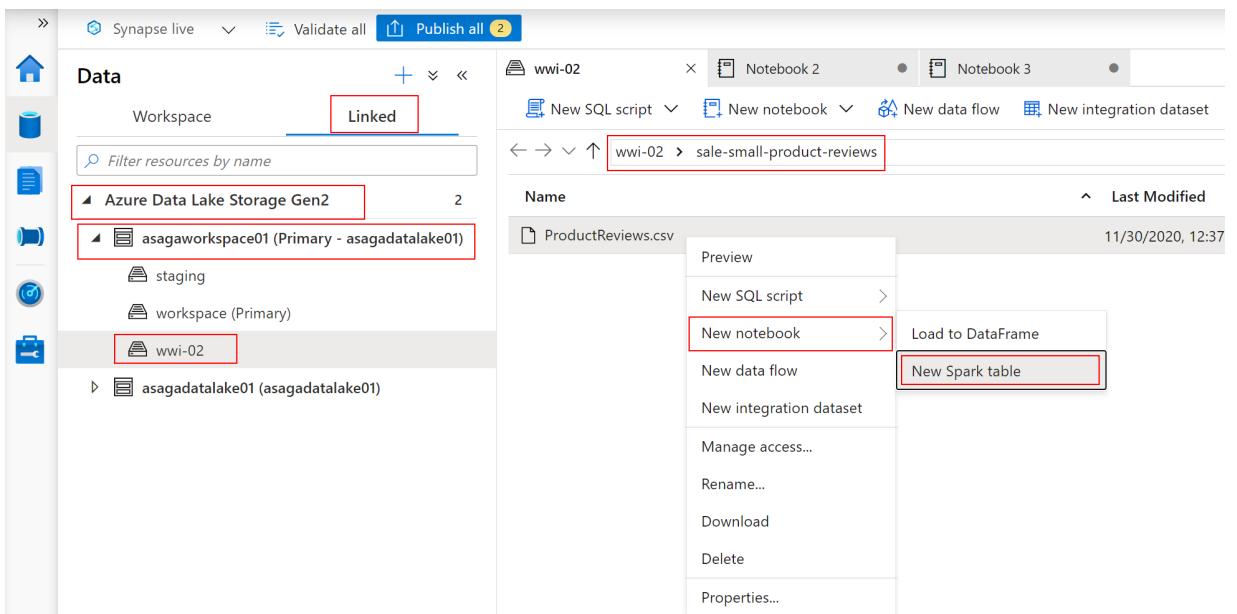
19.7.2 Task 2: Enrich data in a Spark table using a trained model from Azure Cognitive Services

First, we need to create a Spark table to be used as the input for the Cognitive Services model.

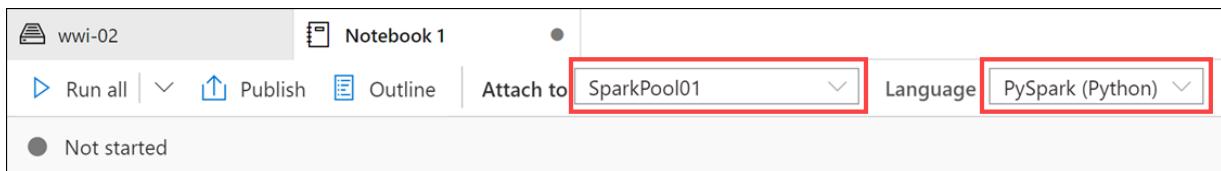
1. Select the **Data** hub.



2. Select the **Linked** tab. In the primary Azure Data Lake Storage Gen 2 account, select the `wwi-02` file system, and then select the `ProductReviews.csv` file under `wwi-02\sale-small-product-reviews`. Right click the file and select **New notebook** -> **New Spark table**.



3. Attach the Apache Spark pool to the notebook and make sure PySpark (Python) is the selected language.



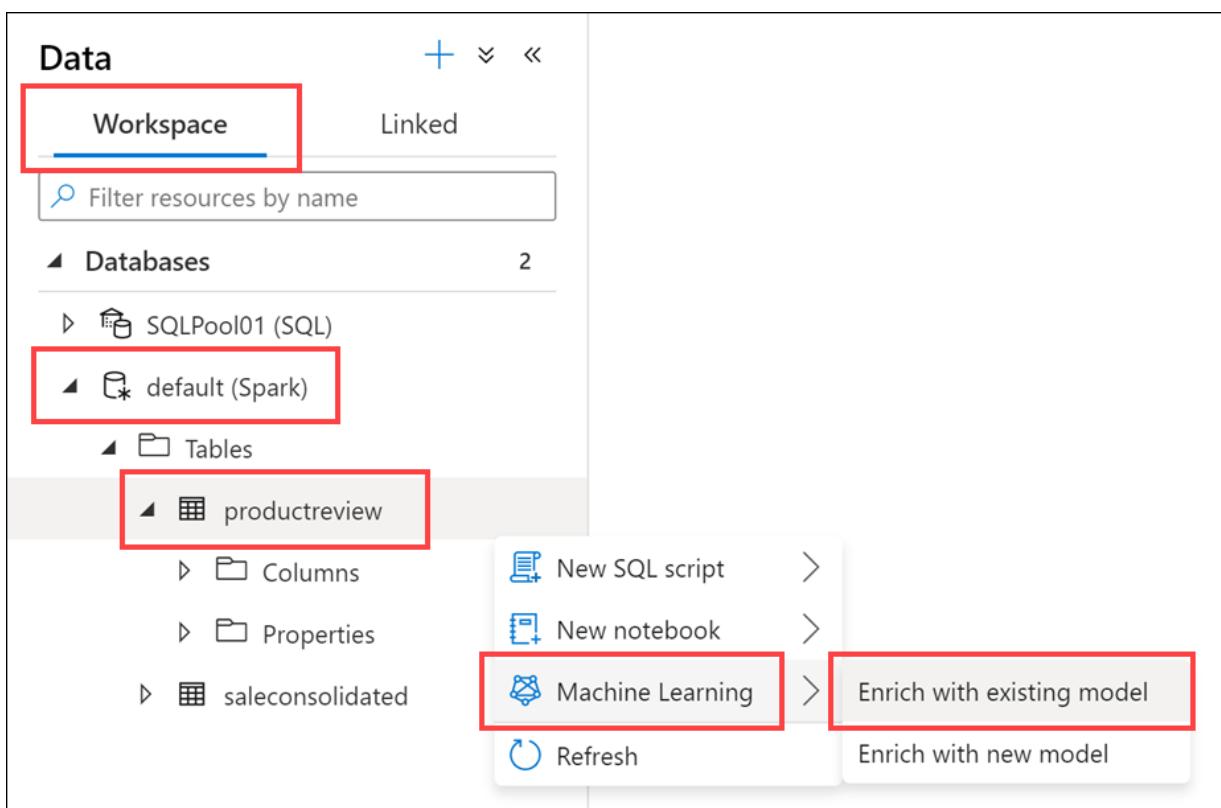
4. Replace the content of the notebook cell with the following code and then run the cell:

```
%%pyspark
df = spark.read.load('abfss://wwi-02@<data_lake_account_name>.dfs.core.windows.net/sale-small-prod',
                     header=True
)
df.write.mode("overwrite").saveAsTable("default.ProductReview")
```

NOTE:

Replace <data_lake_account_name> with the actual name of your Synapse Analytics primary data lake account.

5. To view the table in the Data hub, expand the **default (Spark)** database in the **Workspace** section. Your **productreview** table will show up in the **Tables** folder. Select the three dots at the right of the table name to view the **Machine Learning** option in the context menu and then select **Machine Learning > Enrich with existing model**.



6. In the **Enrich with existing model** dialog, select **Text Analytics - Sentiment Analysis** under **Azure Cognitive Services** and then select **Continue**.

The screenshot shows the Azure Synapse Analytics workspace interface. On the left, there's a sidebar with icons for Home, Databases, Tables, and Notebooks. The main area shows a dataset named 'wwi-02' with a table named 'ProductReviews.csv'. To the right, a modal window titled 'Enrich with existing model' is open. It shows a list of pre-trained Azure Cognitive Services models: 'productreview', 'Anomaly Detector', and 'Text Analytics - Sentiment Analysis'. The 'Text Analytics - Sentiment Analysis' option is highlighted with a red box. At the bottom of the modal are 'Continue', 'Back', and 'Cancel' buttons.

7. Next, provide values as follows:

- **Azure subscription:** select the Azure subscription of your resource group.
- **Cognitive Services account:** select the Cognitive Services account that has been provisioned in your resource group. The name should be `asagacognitiveservices<unique_suffix>`, where `<unique_suffix>` is the unique suffix you provided when deploying the Synapse Analytics workspace.
- **Azure Key Vault linked service:** select the Azure Key Vault linked services that has been provisioned in your Synapse Analytics workspace. The name should be `asagakeyvault<unique_suffix>`, where `<unique_suffix>` is the unique suffix you provided when deploying the Synapse Analytics workspace.
- **Secret name:** enter `ASA-GA-COGNITIVE-SERVICES` (the name of the secret that contains the key for the specified Cognitive Services account).

8. Select Continue to move next.

The screenshot shows the 'Enrich with existing model' dialog with the following inputs filled:

- Azure subscription:** Synapse Analytics Demos and Labs
- Cognitive Services account ***: asagacognitiveservices01
- Azure Key Vault linked service ***: asagakeyvault01
- Secret name ***: ASA-GA-COGNITIVE-SERVICES

 The 'Continue' button is highlighted with a red box at the bottom of the dialog.

9. Next, provide values as follows:

- **Language:** select English.
- **Text column:** select ReviewText (string)

10. Select Open notebook to view the generated code.

NOTE:

When you created the `ProductReview` Spark table by running the notebook cell, you started a Spark session on that notebook. The default settings on your Synapse Analytics workspace will not allow you to start a new notebook that runs in parallel with that one. You will need to copy the contents of the two cells that contain to Cognitive Services integration code into that notebook and run them on the Spark session that you have already started. After copying the two cells, you should see a screen similar to this:

```

Cell 1
1 # pyspark
2 df = spark.read.load("abfss://ww1-02@segadatalake01.dfs.core.windows.net/sale-small-product-reviews/ProductReviews.csv", format="csv")
3 df.createOrReplaceTempView("ProductReview")
4 df.write.mode("overwrite").saveAsTable("default.ProductReview")
5
Command executed in 45s 79ms by odl on 11-10-2020 14:32:37.345 +02:00
> Job execution Succeeded: Spark 2 executors 8 cores

Cell 2
[ ] 1 import mmlspark
2
3 if mmlspark.__spark_package_version__ < "1.0.0-rc3":
4     raise Exception("This notebook is not compatible with the current version of mmlspark: {}. Please upgrade to 1.0.0-rc3 or higher.".format(
5     mmlspark.__spark_package_version__))
6

Cell 3
[ ] 1 from mmlspark.cognitive import *
2 from notebookutils import mmlsparkutils
3 from pyspark.sql.functions import explode
4
5 # Fetch the subscription key (or a general Cognitive Service key) from Azure Key Vault
6 service_key = mmlsparkutils.credentials.getSecret("asagkeyvault01", "ASA-GA-COGNITIVE-SERVICES", "asagkeyvault01")
7
8 # Load the data into a Spark DataFrame
9 df = spark.sql("SELECT * FROM default.productreview")
10
11 sentiment = (TextSentiment()
12     .setLocation("westus2")
13     .setSubscriptionKey(service_key)
14     .setOutputCol("output")
15     .setErrorCol("error")
16     .setLanguage("en")
17     .setTextCol("ReviewText"))
18
19 results = sentiment.transform(df)
20
21 # Show the results
22 display(results)
23     .select("ReviewText", explode("output").alias("exploded"), "error")\
24     .select("ReviewText", "exploded.*", "error")\
25     .limit(10)

```

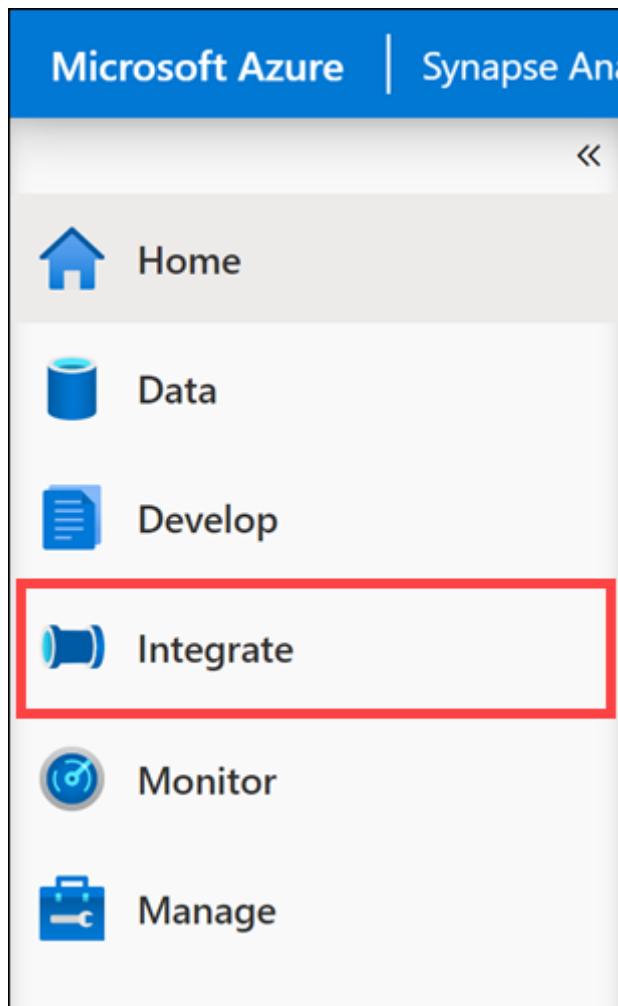
NOTE: To run the notebook generated by Synapse Studio without copying its cells, you can use the **Apache Spark applications** section of the **Monitor** hub where you can view and cancel running Spark session. For more details on this, see [Use Synapse Studio to monitor your Apache Spark applications](#). In this lab, we used to copy cells approach to avoid the extra time required to cancel the running Spark session and start a new one afterwards.

Run cells 2 and 3 in the notebook to get the sentiment analysis results for your data.

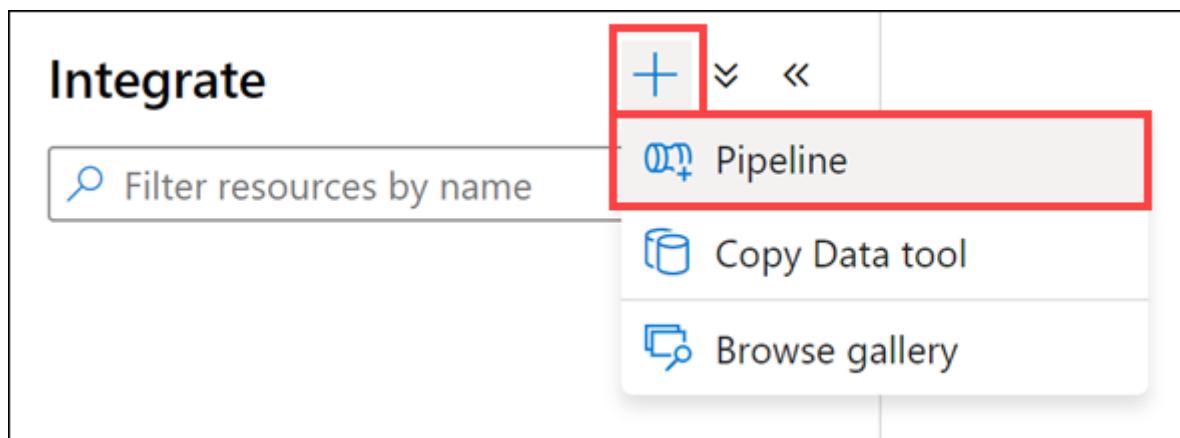
ReviewText	sentiment	statistics	documentScores	sentences
The product was standard, and it seems suitable to use. I am unworried about the way it...	positive	undefined	undefined	► "[{"sentiment":"positive","confidence":0.99,"ReviewText":"The product was standard, and it seems suitable to use. I am unworried about the way it..."}]
The product was just what I needed, and it seems very easy to use. I like the way it handles...	positive	undefined	undefined	► "[{"sentiment":"positive","confidence":0.99,"ReviewText":"The product was just what I needed, and it seems very easy to use. I like the way it handles..."}]
The product was standard, but it seems difficult to use. I do not like the way it handles...	negative	undefined	undefined	► "[{"sentiment":"negative","confidence":0.99,"ReviewText":"The product was standard, but it seems difficult to use. I do not like the way it handles..."}]
The product was not particularly great, but it seems fair to use. I am untroubled about th...	mixed	undefined	undefined	► "[{"sentiment":"neutral","confidence":0.99,"ReviewText":"The product was not particularly great, but it seems fair to use. I am untroubled about th..."}]
The product was not particularly great, and it seems difficult to use. I do not like the way...	negative	undefined	undefined	► "[{"sentiment":"negative","confidence":0.99,"ReviewText":"The product was not particularly great, and it seems difficult to use. I do not like the way..."}]
The product was normal, but it seems uncomfortable to use. I do not like the way it han...	negative	undefined	undefined	► "[{"sentiment":"negative","confidence":0.99,"ReviewText":"The product was normal, but it seems uncomfortable to use. I do not like the way it han..."}]
The product was perfect, and it seems adequate to use. I love the way it handles. Overall,...	positive	undefined	undefined	► "[{"sentiment":"positive","confidence":0.99,"ReviewText":"The product was perfect, and it seems adequate to use. I love the way it handles. Overall,..."}]
The product was not particularly great, but it seems fair to use. I am unworried about th...	negative	undefined	undefined	► "[{"sentiment":"negative","confidence":0.99,"ReviewText":"The product was not particularly great, but it seems fair to use. I am unworried about th..."}]

19.7.3 Task 3: Integrate a Machine Learning-based enrichment procedure in a Synapse pipeline

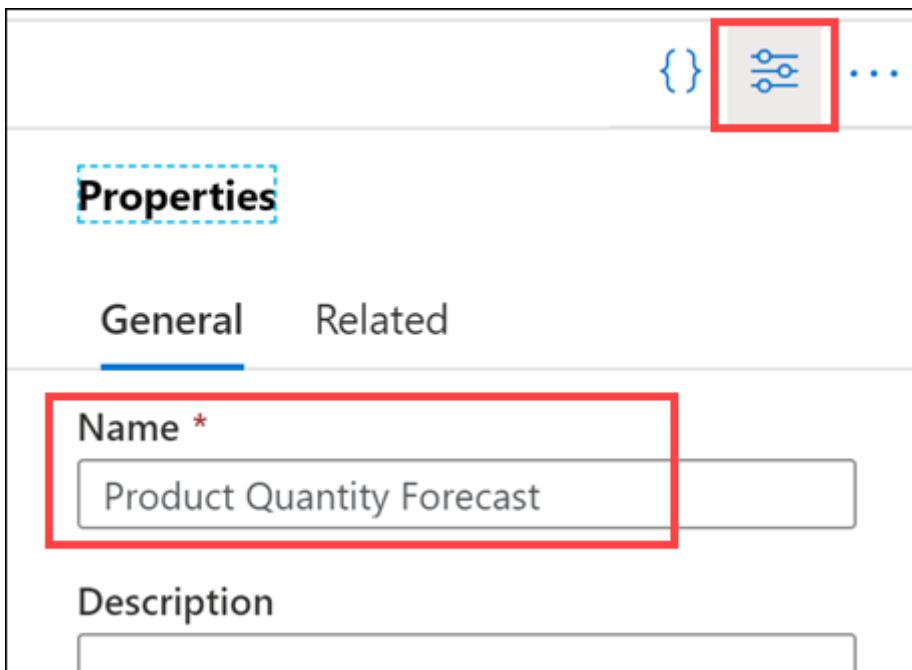
- Select the **Integrate** hub.



2. Select +, then **Pipeline** to create a new Synapse pipeline.



3. Enter **Product Quantity Forecast** as the name of the pipeline in the properties pane, then select the **Properties** button to close the pane.



4. From the Move & transform section, add a Copy data activity and name it Import forecast requests.

5. In the Source section of the copy activity properties, provide the following values:

- **Source dataset:** select the `wwi02_sale_small_product_quantity_forecast_adls` dataset.
- **File path type:** select Wildcard file path
- **Wildcard paths:** enter `sale-small-product-quantity-forecast` in the first textbox, and `*.csv` in the second.

General **Source** Sink ¹ Mapping Settings User properties

Source dataset * [Open](#) [New](#) [Preview data](#) [Learn more](#)

File path type File path in dataset Wildcard file path List of files

Wildcard paths wwi-02 / /

Start time (UTC) End time (UTC)

Filter by last modified [View](#)

Recursively

6. In the **Sink** section of the copy activity properties, provide the following values:

- **Sink dataset:** select the `wwi02_sale_small_product_quantity_forecast_asa` dataset.

General Source **Sink** Mapping Settings User properties

Sink dataset * [Open](#) [New](#)

Copy method PolyBase [View](#) Copy command [View](#) Bulk insert

Allow PolyBase

Reject type

Reject value

Use type default

7. In the **Mapping** section of the copy activity properties, select **Import schemas** and check field mappings between source and sink.

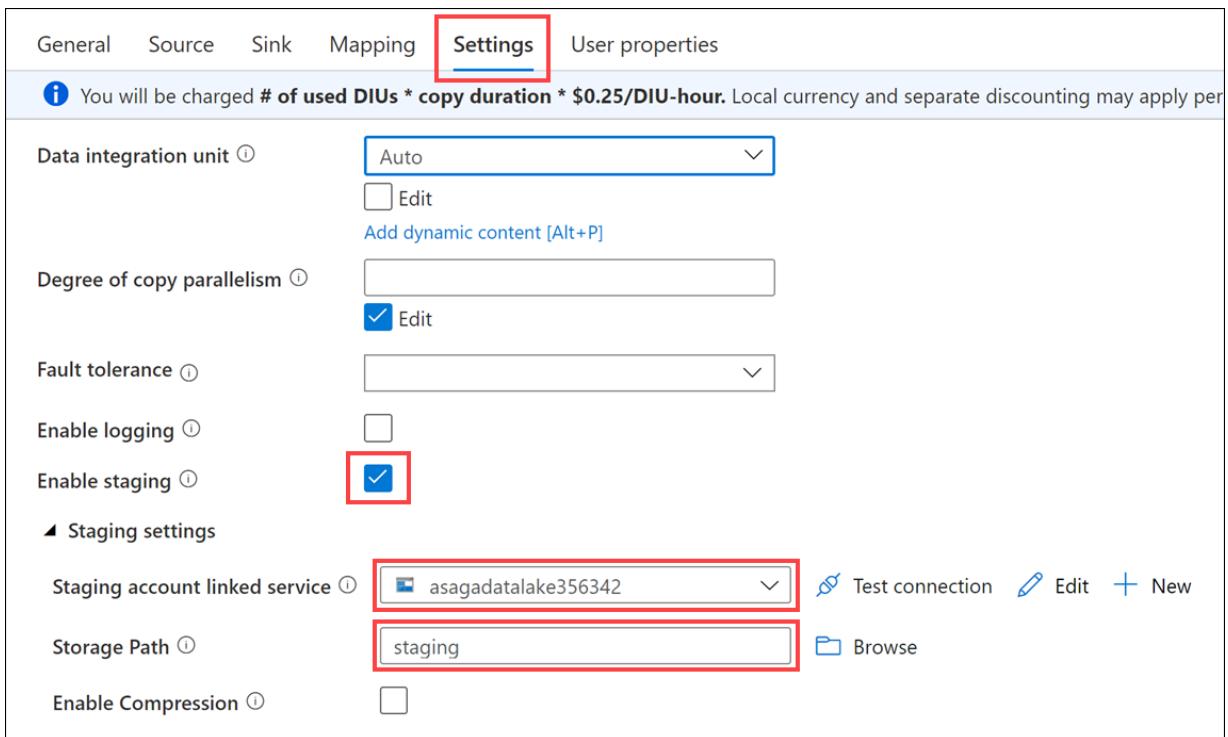
General Source Sink **Mapping** Settings User properties

[Import schemas](#) [Preview source](#) [New mapping](#) [Clear](#) [Reset](#) [Delete](#)

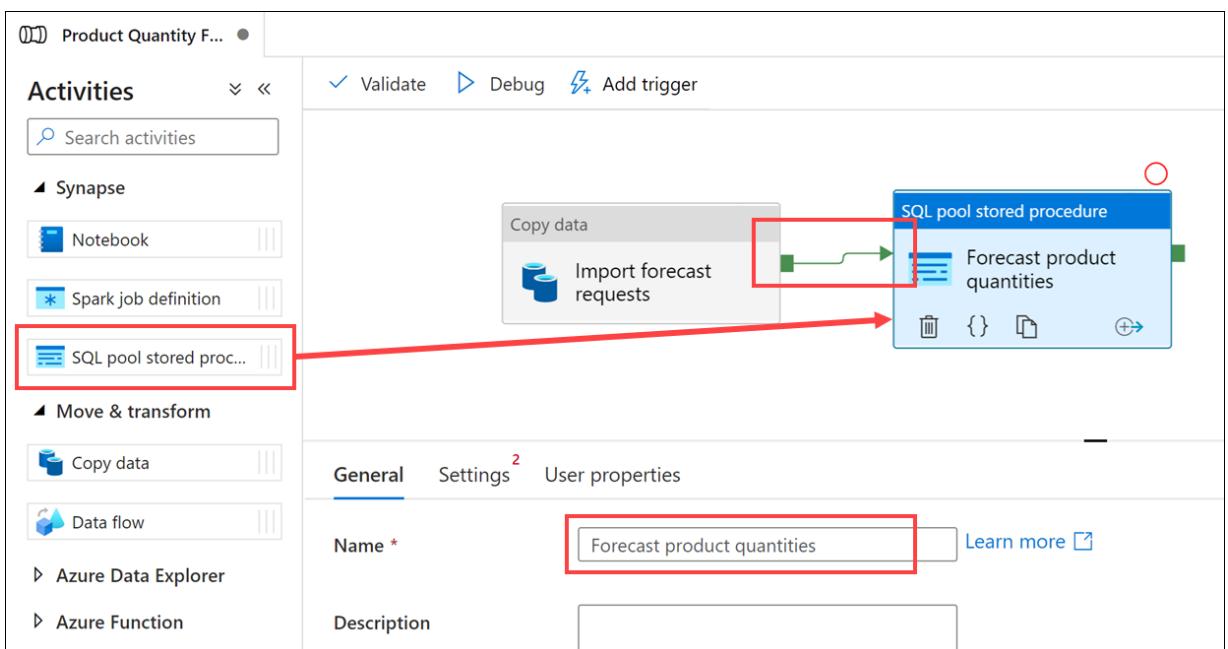
Source	Type	Destination	Type
ProductId	String	ProductId	int
TransactionDate	String	TransactionDate	int
Hour	String	Hour	int
TotalQuantity	String	TotalQuantity	int

8. In the **Settings** section of the copy activity properties, provide the following values:

- **Enable staging:** select the option.
- **Staging account linked service:** select the `asagadatalake<unique_suffix>` linked service (where `<unique_suffix>` is the unique suffix you provided when deploying the Synapse Analytics workspace).
- **Storage path:** enter `staging`.

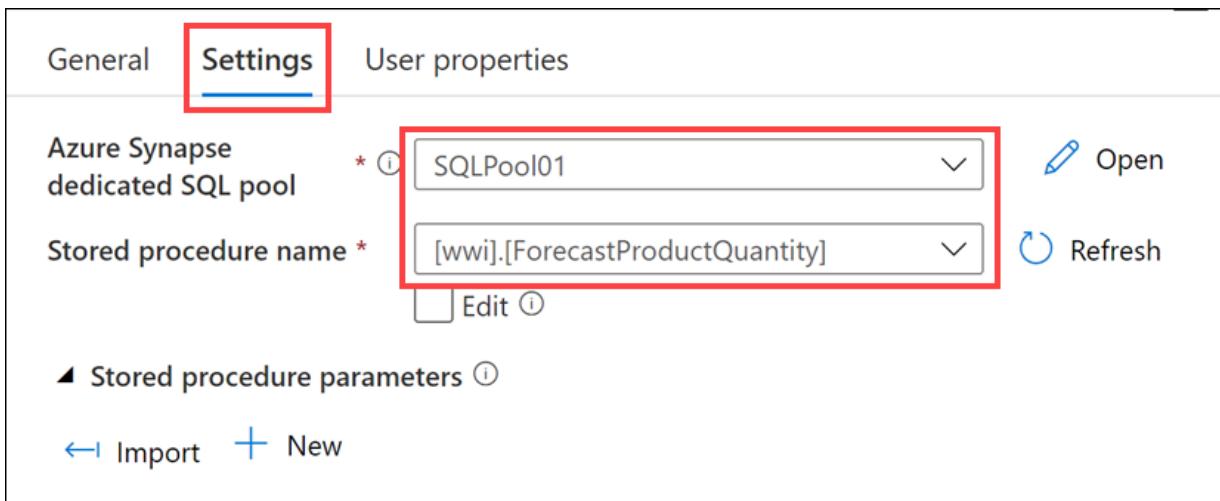


9. From the Synapse section, add a **SQL pool stored procedure** activity and name it **Forecast product quantities**. Connect the two pipeline activities to ensure the stored procedure runs after the data import.

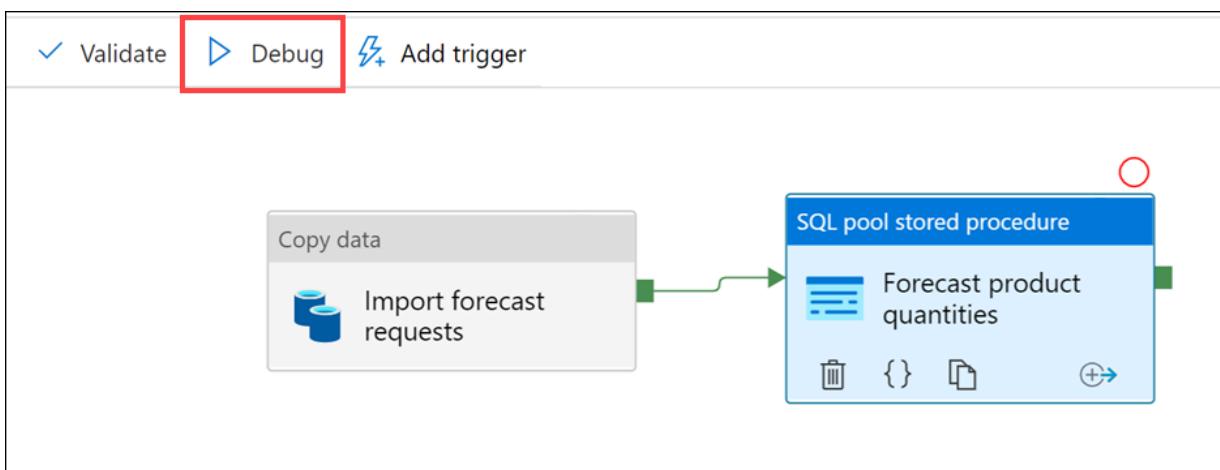


10. In the **Settings** section of the stored procedure activity properties, provide the following values:

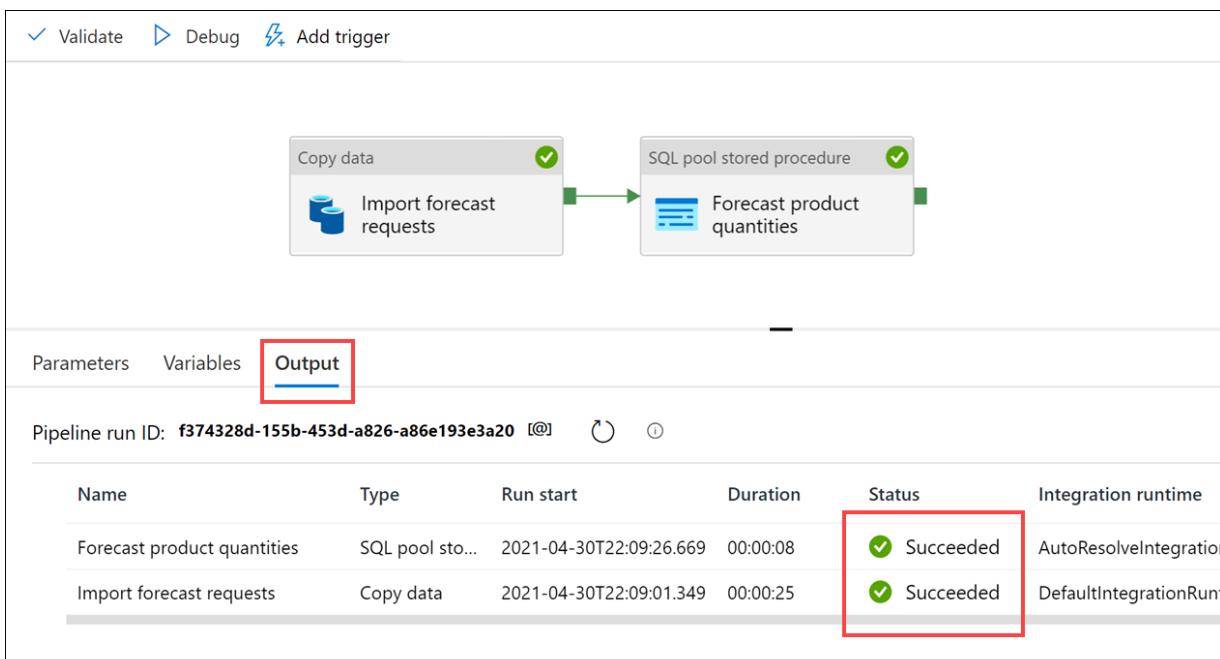
- **Azure Synapse dedicated SQL pool:** select your dedicated SQL pool (eg. `SQLPool01`).
- **Stored procedure name:** select `[wwi].[ForecastProductQuantity]`.



11. Select **Debug** to make sure the pipeline works correctly.

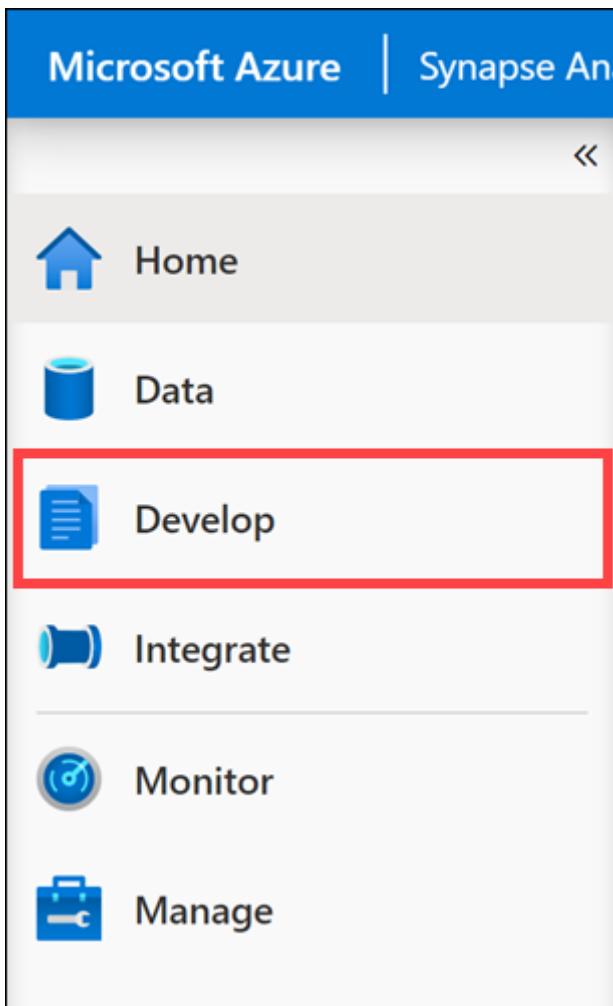


The pipeline's **Output** tab will show the debug status. Wait until the status for both activities is **Succeeded**.



12. Select **Publish all**, then **Publish** to publish the pipeline.

13. Select the **Develop** hub.



14. Select +, then select SQL script.

The screenshot shows the 'Develop' blade in the Azure portal. At the top, it says 'Develop' and has a search bar labeled 'Filter resources by name'. Below that is a section for 'Power BI'. To the right, there is a list of resource types: '+', 'Product Quantitative', 'SQL script' (which is highlighted with a red box), 'Notebook', 'Data flow', 'Apache Spark job definition', 'Power BI report', 'Browse gallery', and 'Import'. The '+' icon is also highlighted with a red box.

15. Connect to your dedicated SQL pool, then execute the following script:

```
SELECT
```

```

*
FROM
wwi.ProductQuantityForecast

```

In the results you should now see forecasted values for Hour = 11 (which are corresponding to rows imported by the pipeline):

ProductId	TransactionDate	Hour	TotalQuantity
1000	20201209	10	8
200	20201209	10	56
800	20201209	10	8
100	20201209	11	49
1100	20201209	10	6
200	20201209	11	56
300	20201209	11	7
400	20201209	11	5
500	20201209	11	8
600	20201209	11	8
700	20201209	11	6

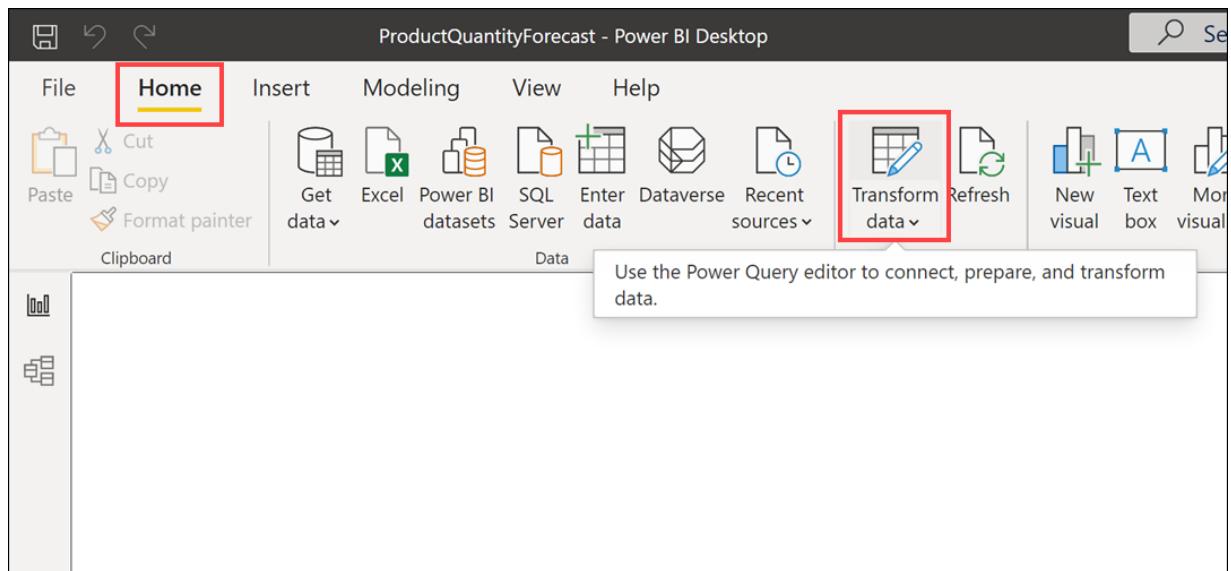
19.8 Exercise 4: Serve prediction results using Power BI

In this exercise you will view the prediction results in a Power BI report. You will also trigger the forecast pipeline with new input data and view the updated quantites in the Power BI report.

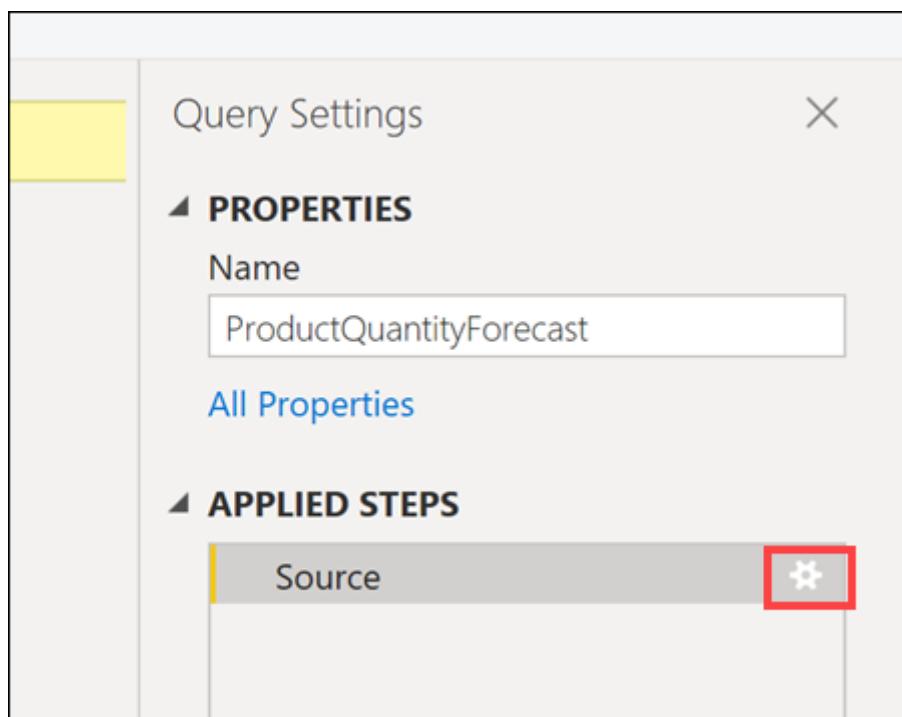
19.8.1 Task 1: Display prediction results in a Power BI report

First, you will publish a simple Product Quantity Forecast report to Power BI.

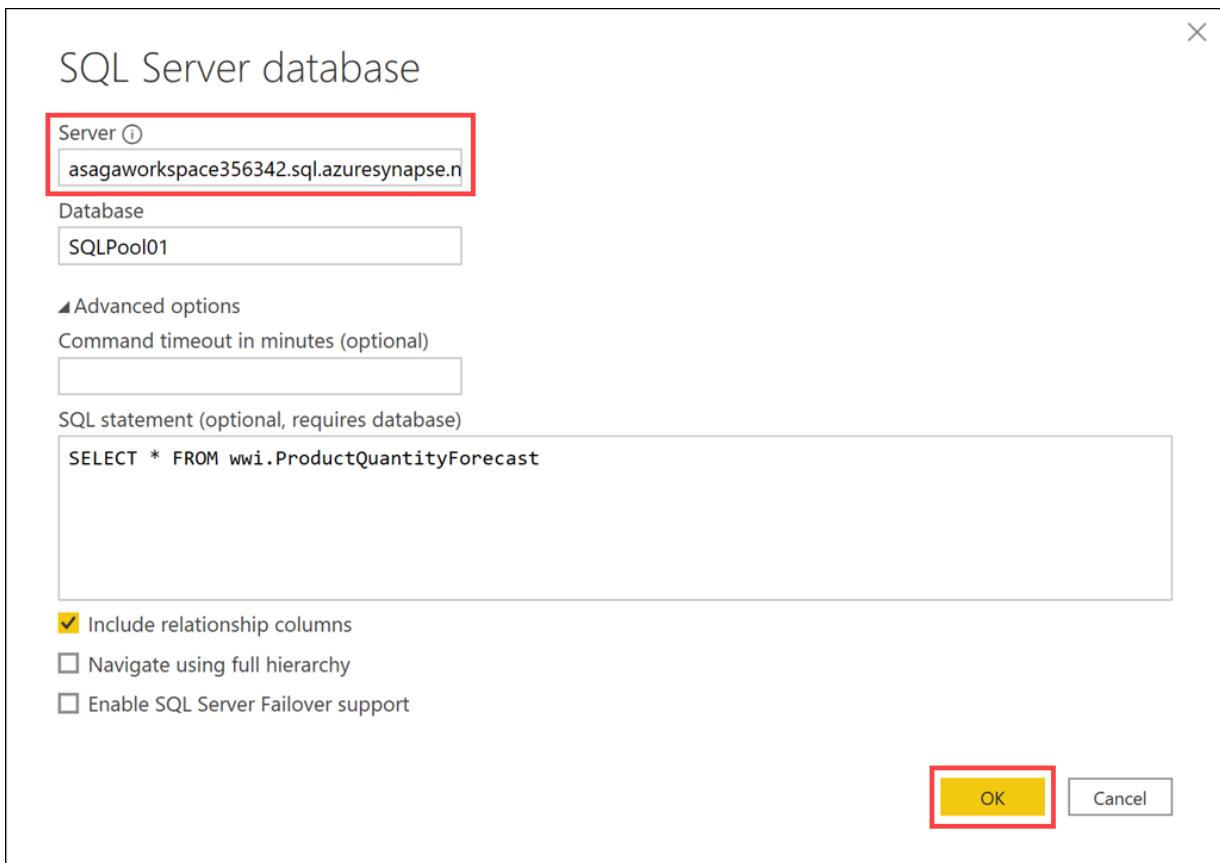
1. Download the `ProductQuantityForecast.pbix` file from the GitHub repo: [ProductQuantityForecast.pbix](#) (select `Download` on the GitHub page).
2. Open the file with Power BI Desktop (ignore the warning about missing credentials). Also, if you are first prompted to update credentials, ignore the messages and close the pop-ups without updating the connection information.
3. In the Home section of the report, select `Transform data`.



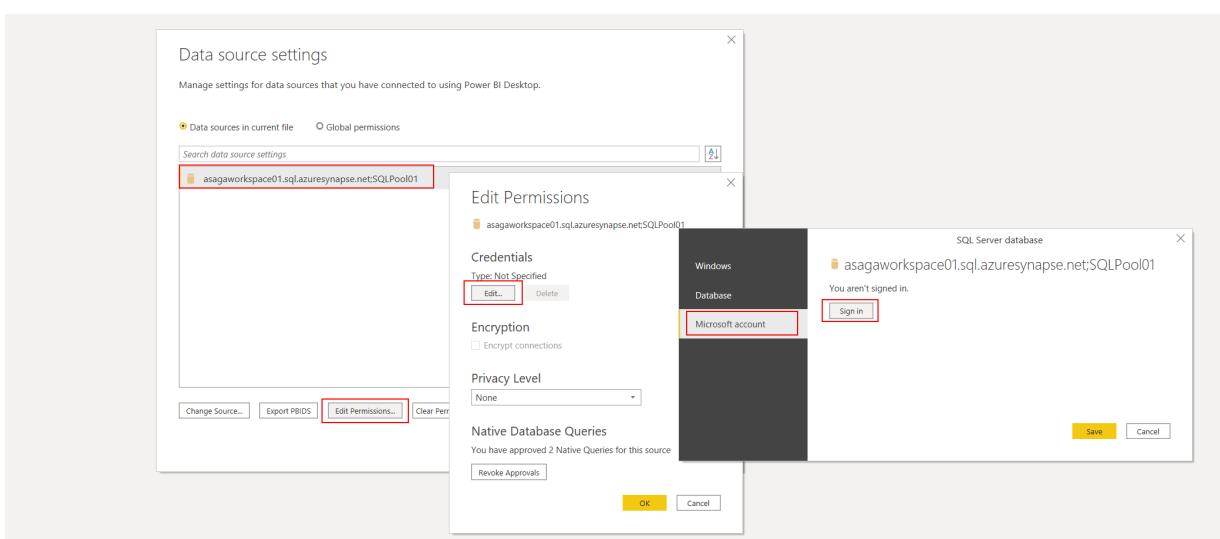
4. Select the gear icon on the Source entry in the APPLIED STEPS list of the ProductQuantityForecast query.



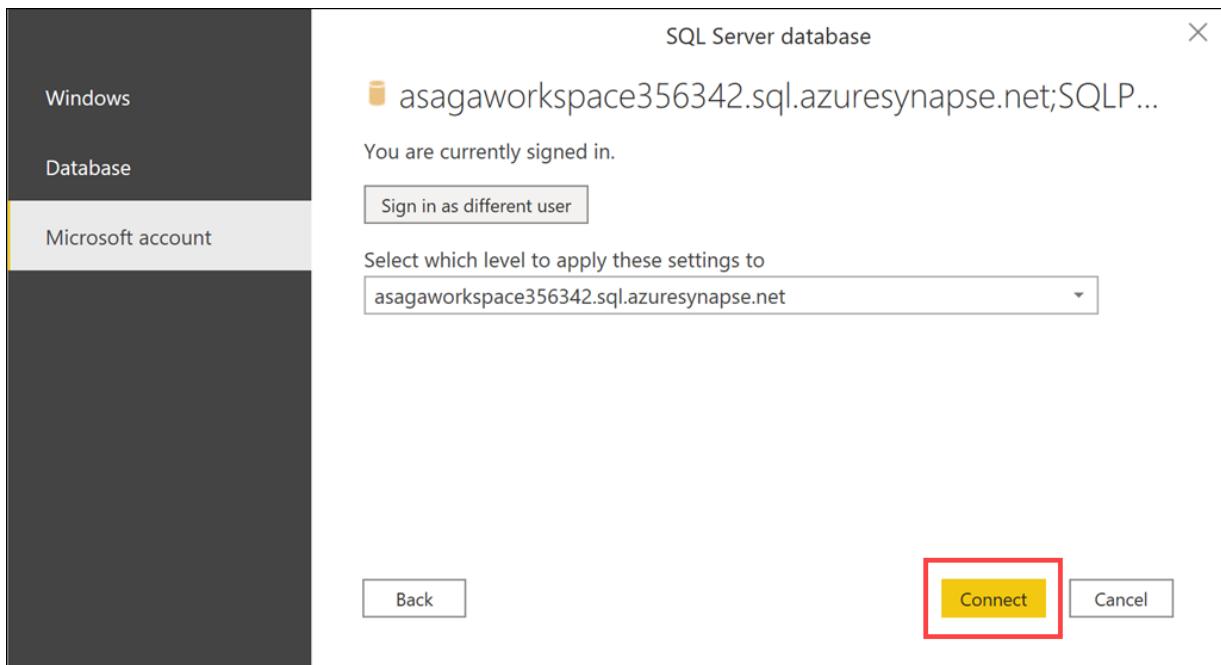
5. Change the name of the server to `asagaworkspace<unique_suffix>.sql.azuresynapse.net` (where `<unique_suffix>` is the unique suffix for your Synapse Analytics workspace), then select **OK**.



6. The credentials window will pop up and prompt you to enter the credentials to connect to the Synapse Analytics SQL pool (in case it doesn't, select **Data source settings** on the ribbon, select your data source, select **Edit Permissions...**, and then select **Edit...** under **Credentials**).
7. In the credentials window, select **Microsoft account** and then select **Sign in**. Use your Power BI Pro account to sign in.



8. After signing in, select **Connect** to establish the connection to your dedicated SQL pool.



9. Close all open popup windows, then select **Close & Apply**.

The screenshot shows the Power Query Editor window with a table of data. The table has columns: ProductId, TransactionDate, Hour, and TotalQuantity. The data looks like this:

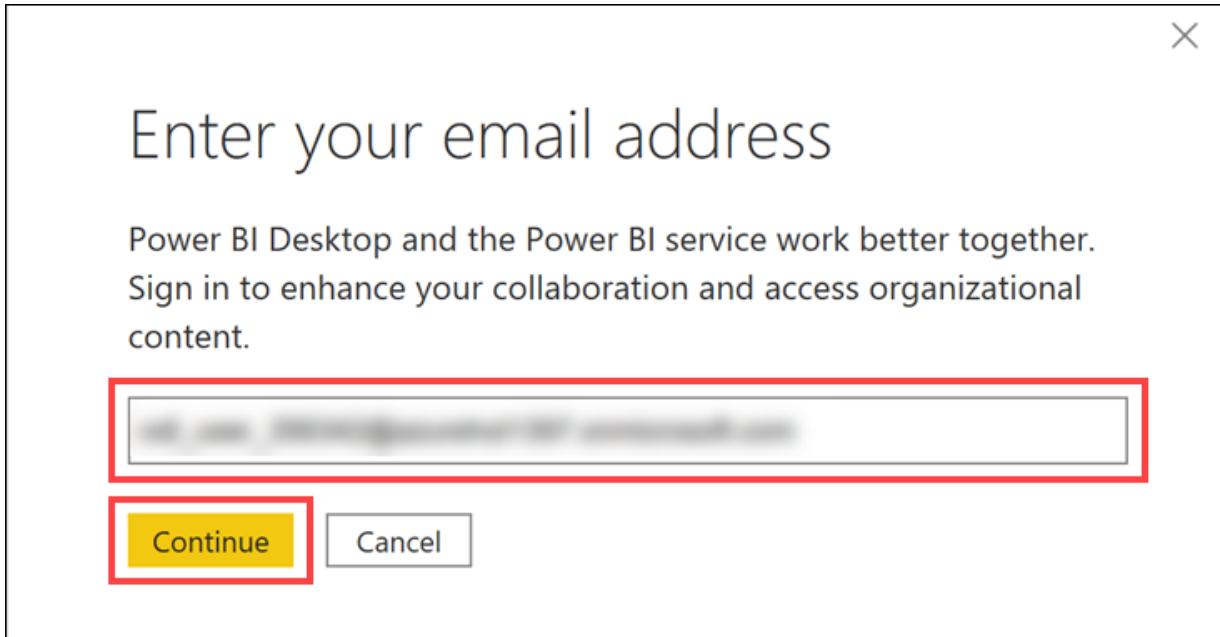
	ProductId	TransactionDate	Hour	TotalQuantity
1	400	20201209	10	5
2	200	20201209	10	56
3	700	20201209	10	6
4	100	20201209	10	49
5	100	20201209	11	49
6	200	20201209	11	56
7	300	20201209	11	7
8	400	20201209	11	5
9	500	20201209	11	8
10	600	20201209	11	8

The "File" ribbon tab is selected, and the "Close & Apply" button is highlighted with a red box. A tooltip says "Close the Query Editor window and apply any pending changes."

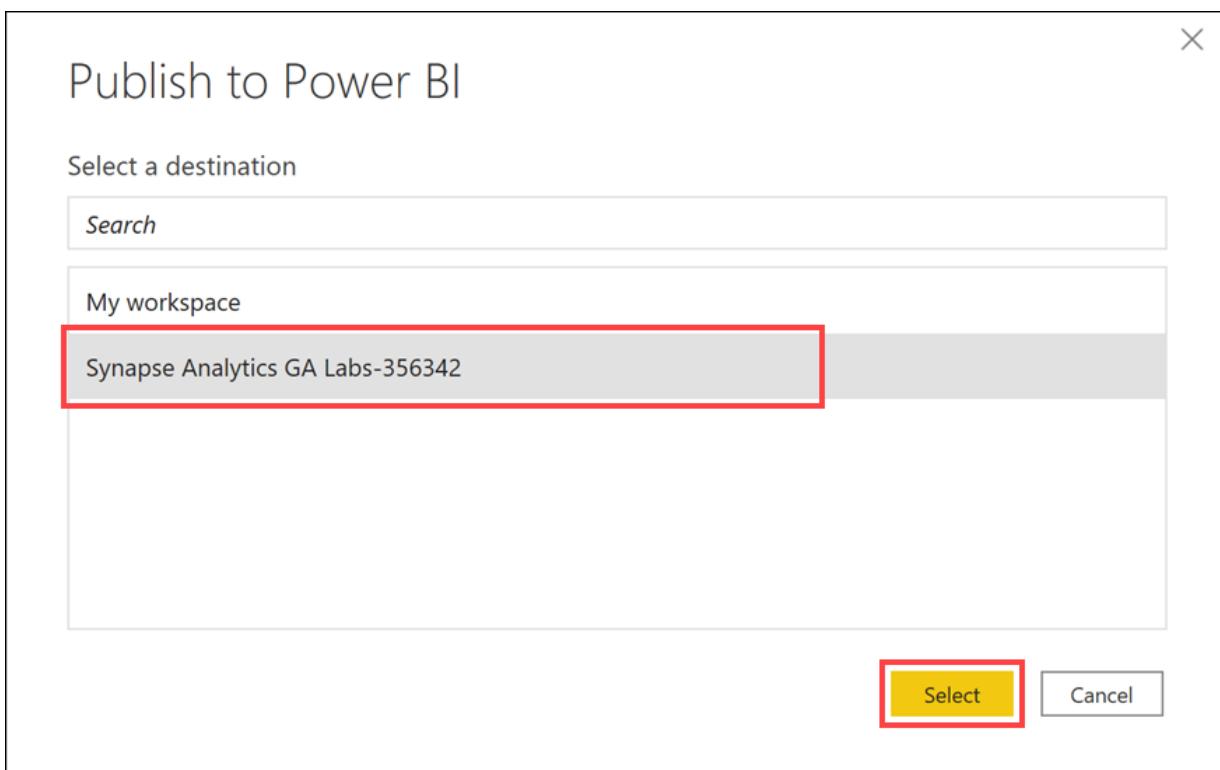
10. After the report loads, select **Publish** on the ribbon. When prompted to save your changes, select **Save**.

The screenshot shows the Power BI Desktop interface. The ribbon is visible at the top with tabs: File, Home, Insert, Modeling, View, Help. The "Home" tab is selected. The "Clipboard" section shows a bar chart titled "TotalQuantity by ProductId". The chart has "TotalQuantity" on the Y-axis (0, 50, 100) and "ProductId" on the X-axis (0, 200, 400, 600, 800, 1000, 1200). The "Publish" button in the ribbon is highlighted with a red box. A tooltip says "Publish this report online in the Power BI service."

11. When prompted, enter the email address of your Azure account you are using for this lab, then select **Continue**. Enter your password or select your user from the list when prompted.



12. Select the Synapse Analytics Power BI Pro workspace created for this lab, then select **Save**.



Wait until the publish succeeds.



Publishing to Power BI

✓ Success!

[Open 'ProductQuantityForecast.pbix' in Power BI](#)

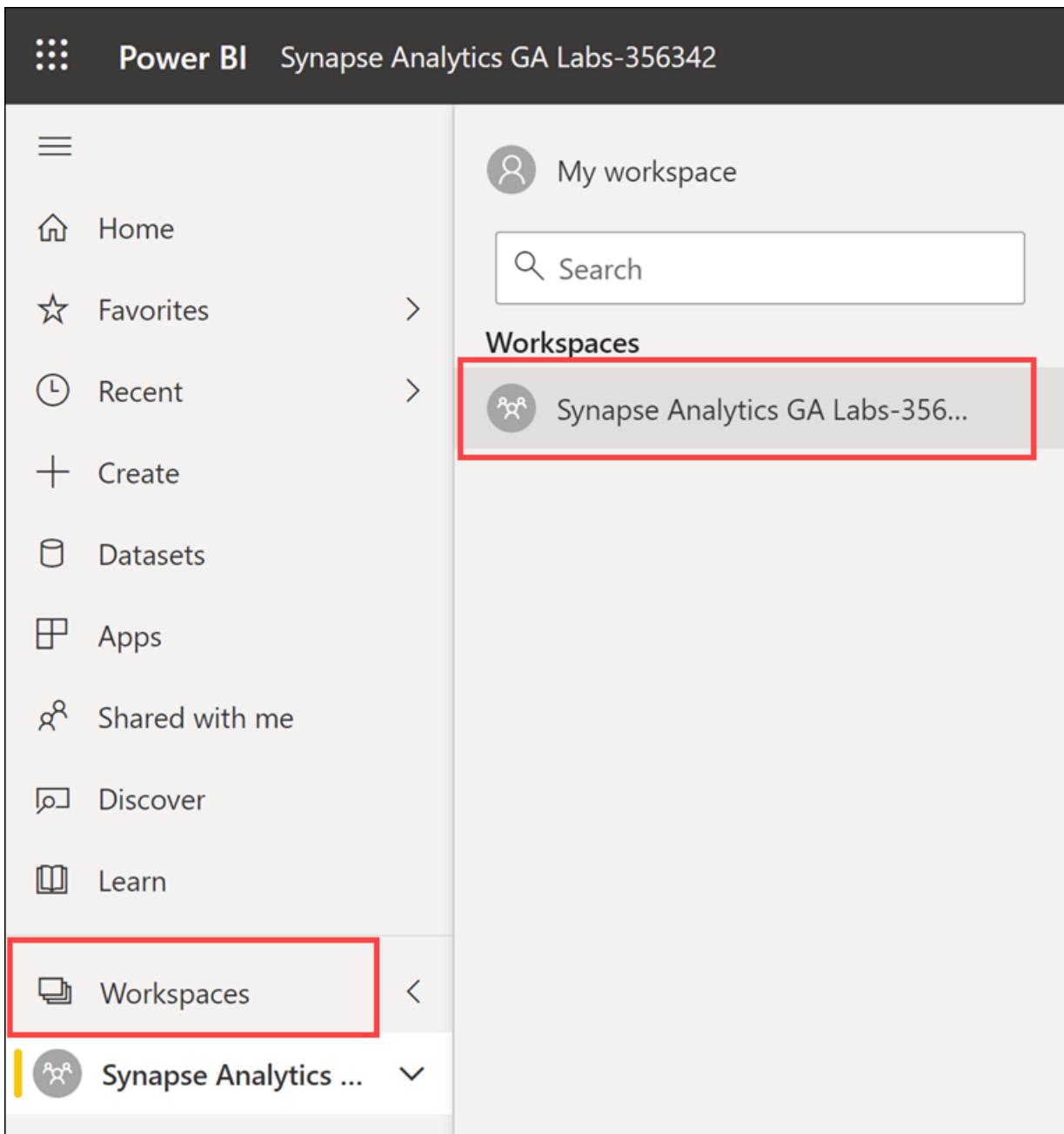


Did you know?

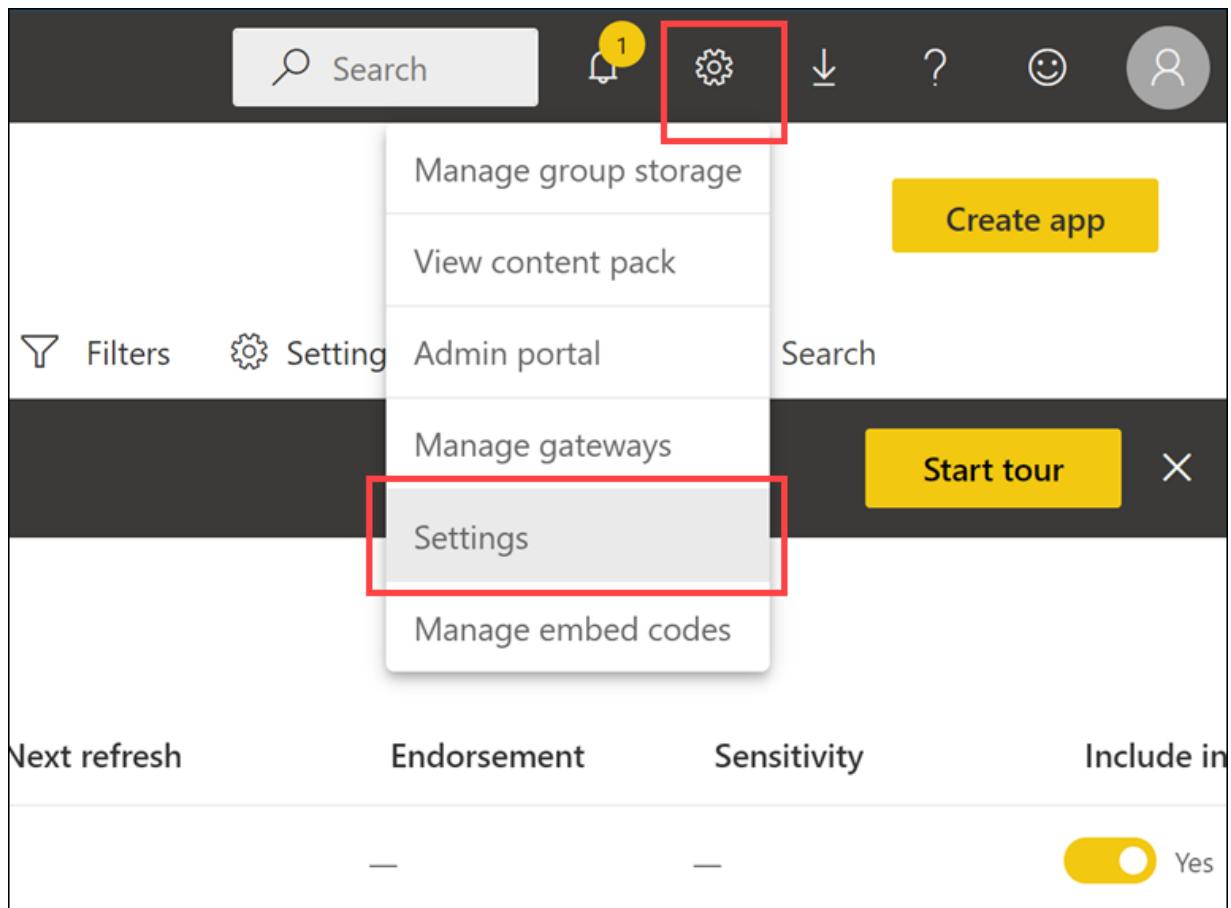
You can create a portrait view of your report, tailored for mobile phones.
On the **View** tab, select **Mobile Layout**. [Learn more](#)

[Got it](#)

13. In a new web browser tab, navigate to <https://powerbi.com>.
14. Select **Sign in** and enter your Azure credentials you are using for the lab, when prompted.
15. Select **Workspaces** on the left-hand menu, then select the Synapse Analytics Power BI workspace created for this lab. This is the same workspace you published to a moment ago.



16. Select **Settings** in the top-most menu, then select **Settings**.



17. Select the **Datasets** tab, then select **Edit credentials**.

A screenshot of the 'Datasets' tab in the Power BI service. The tab bar includes General, Alerts, Subscriptions, Dashboards, Datasets (highlighted with a red box), Workbooks, Dataflows, and App. On the left, a card for 'ProductQuantityForecast' shows it has been configured by 'odl_user_356342@azurehol1397.onmicrosoft.com'. The main area shows a 'Gateway connection' section with a warning message: 'Your data source can't be refreshed because the credentials are invalid. Please update your credentials and try again.' Below this are sections for 'Data source credentials', 'Parameters', and 'Scheduled refresh'. A red box highlights the 'Edit credentials' link next to the warning message.

18. Select **OAuth2** under **Authentication method**, then select **Sign in**. When prompted, enter your credentials.

Configure ProductQuantityFo...

X

server

asagaworkspace356342.sql.azuresynapse.net



database

SQLPool01

Authentication method

OAuth2



Privacy level setting for this data source

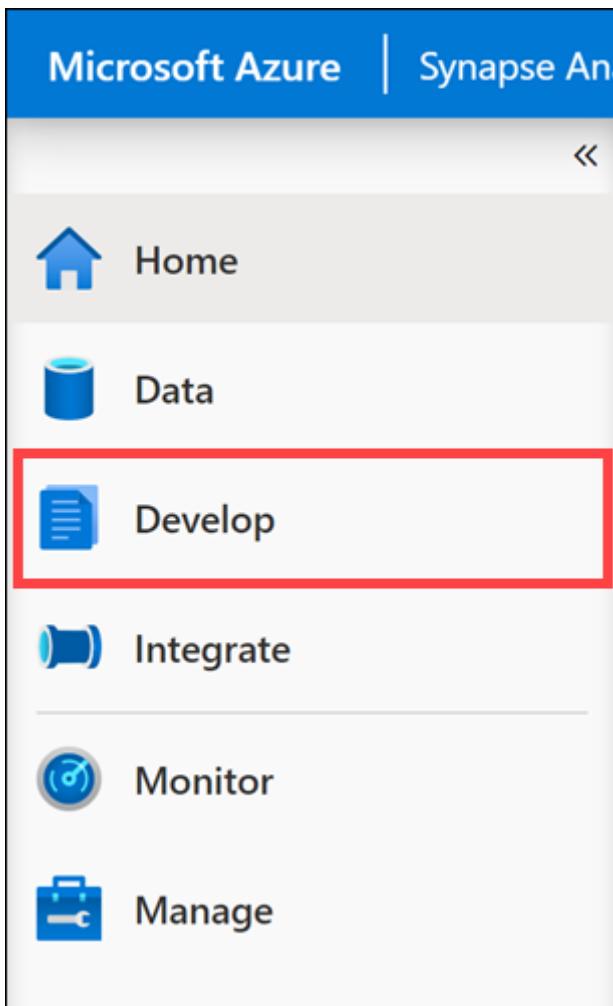


Report viewers can only access this data source with their own Power BI identities using DirectQuery. [Learn more](#)

Sign in

Cancel

19. To view the results of the report, switch back to Synapse Studio.
20. Select the Develop hub on the left-hand side.



21. Expand the Power BI section, and select the ProductQuantityForecast report under the Power BI reports section from your workspace.

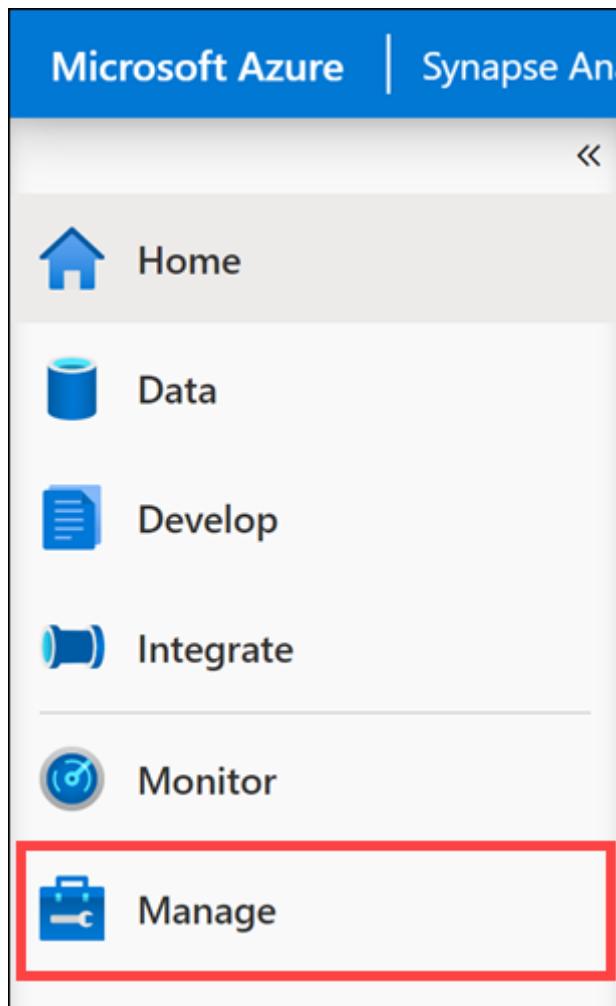


19.9 Exercise 5: Cleanup

Complete these steps to free up resources you no longer need.

19.9.1 Task 1: Pause the dedicated SQL pool

1. Open Synapse Studio (<https://web.azuresynapse.net/>).
2. Select the Manage hub.



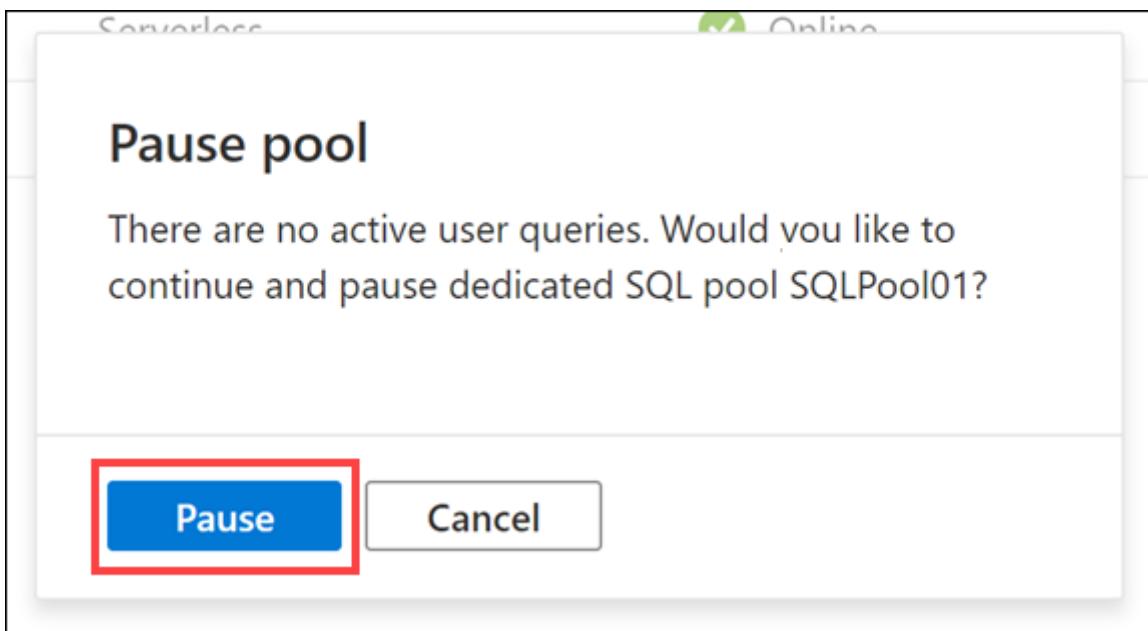
3. Select **SQL pools** in the left-hand menu (1). Hover over the name of the dedicated SQL pool and select **Pause** (2).

The screenshot shows the 'SQL pools' blade in the Azure portal. The left sidebar lists various pool types: Analytics pools (with 'SQL pools' highlighted by a red box and a red number 1), External connections, Integration, and Security. The main area displays the 'SQL pools' table with the following data:

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

A red box highlights the 'Pause' button for the 'SQLPool01' row, and a red number 2 is placed above it.

4. When prompted, select **Pause**.



19.10 Resources

To learn more about the topics covered in this lab, use these resources:

- Quickstart: Create a new Azure Machine Learning linked service in Synapse
- Tutorial: Machine learning model scoring wizard for dedicated SQL pools