
Microsoft Azure Solutions Business
Welcome to
Azure Synapse Technical Boot Camp
Day 1

A look into Day 1

Kick-Off	6:30-6:35	Welcome	Main Call
	6:35-7:00	Azure Synapse Analytics 101	
	7:00-7:15	Demo Walkthrough	
Ingest	7:15-7:30	Break	
	7:30-8:30	Data Loading & Data Lake Organization	
	8:30-9:30	Activity: Data Lake Design & Security Considerations	Table Group Call
	9:30-9:45	Break	
	9:45-10:30	Build Hands-on: Data Integration Part 1	
Transform	10:30-11:30	Break	Main Call
	11:30-12:00	Data Transformations	
	12:00-12:30	Activity: Data Engineering Discussion	
	12:30-1:30	Build Hands-on: Data Integration Part 2	Table Group Call
		End of Day 1	

Today we will be learning and collaborating across three spaces:

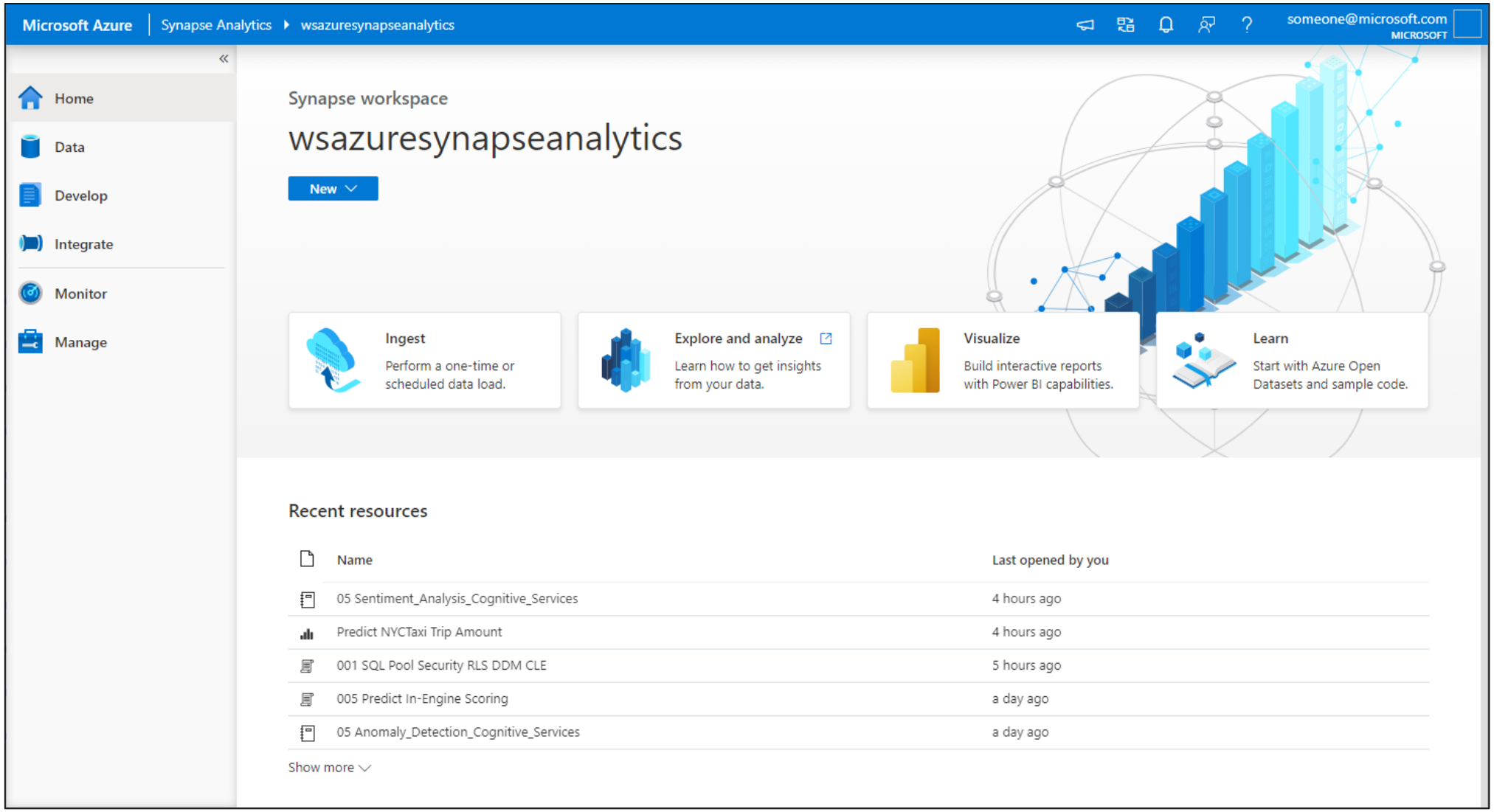
- Main call (this meeting)
- Table Group channel within the event Team
- CloudLabs Learner Portal & Synapse environment

Presentation/ Whole Group | Lab | Activity/ Discussion/ Group Work | Announcements

Azure Synapse 101

Studio

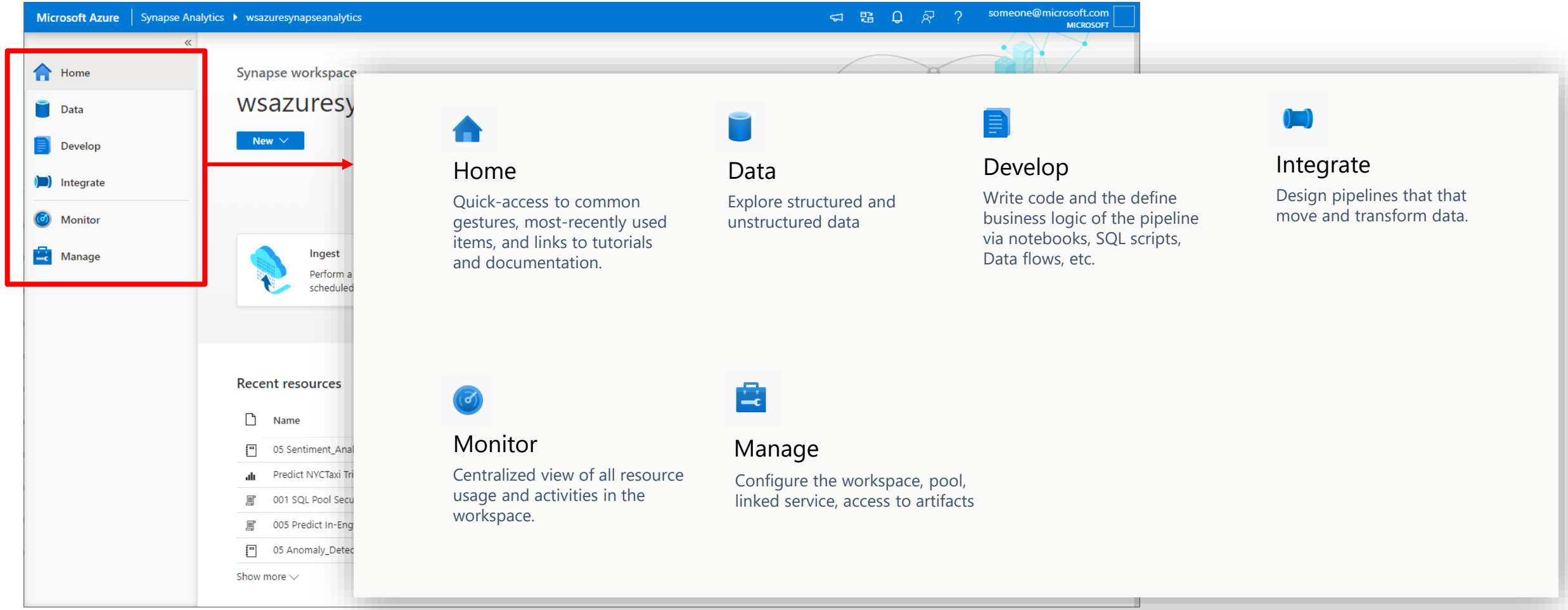
A single place for Data Engineers, Data Scientists, and IT Pros to collaborate on enterprise analytics



Synapse Studio

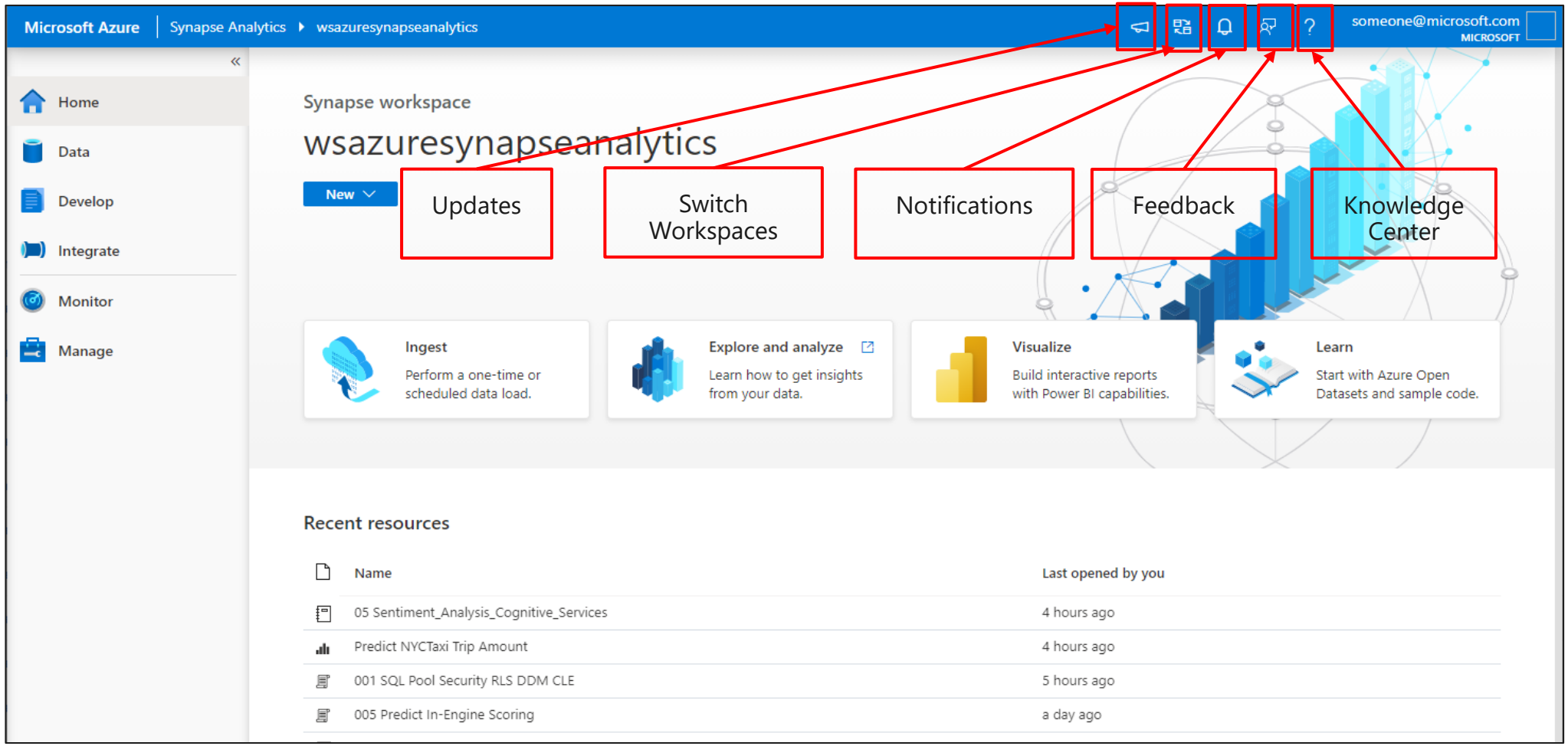
Synapse Studio divided into **Activity hubs**.

These organize the tasks needed for building analytics solution.



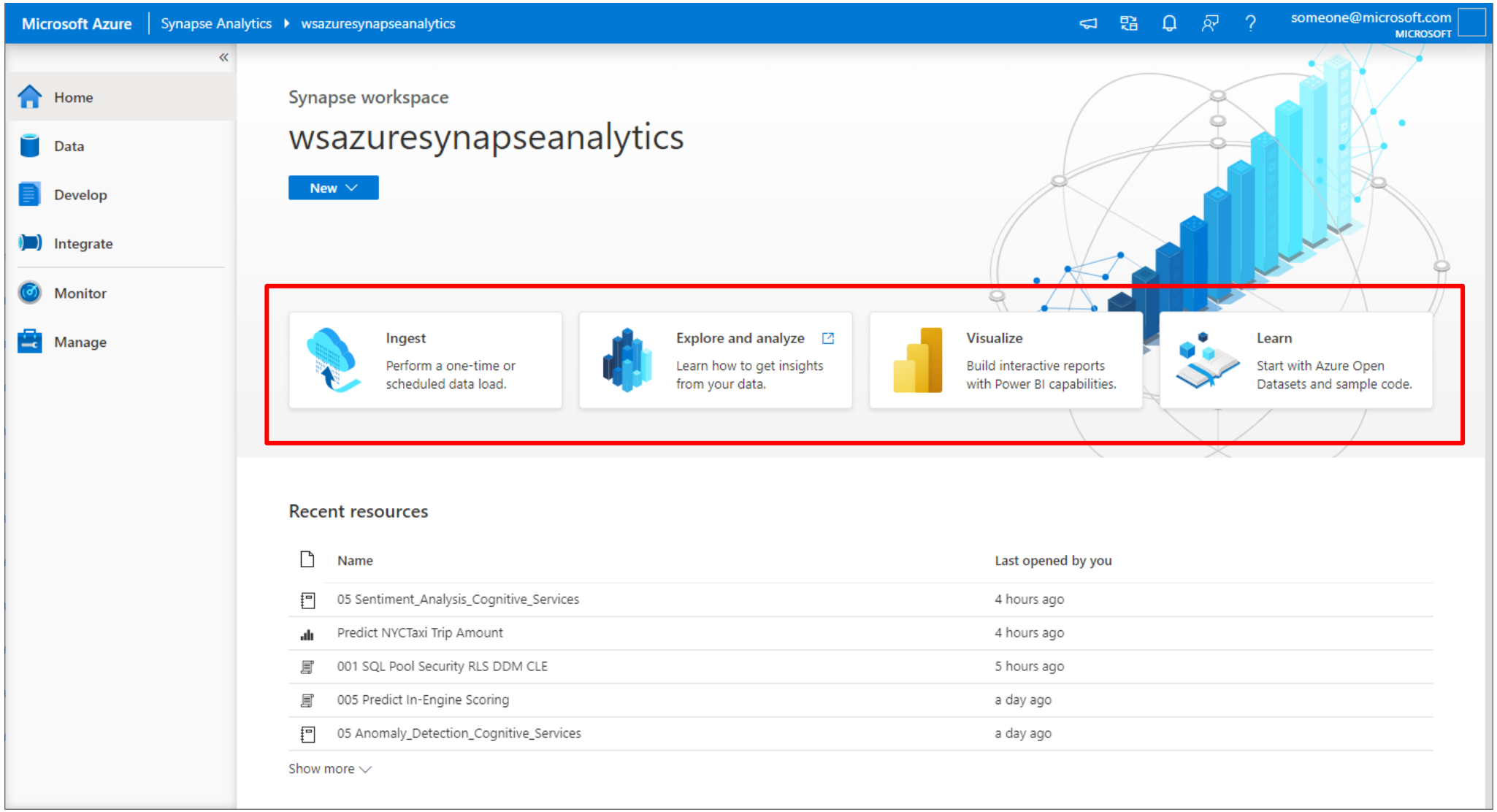
Home Hub

Ease of access to get updates, to switch workspace, to get notifications and to provide feedback



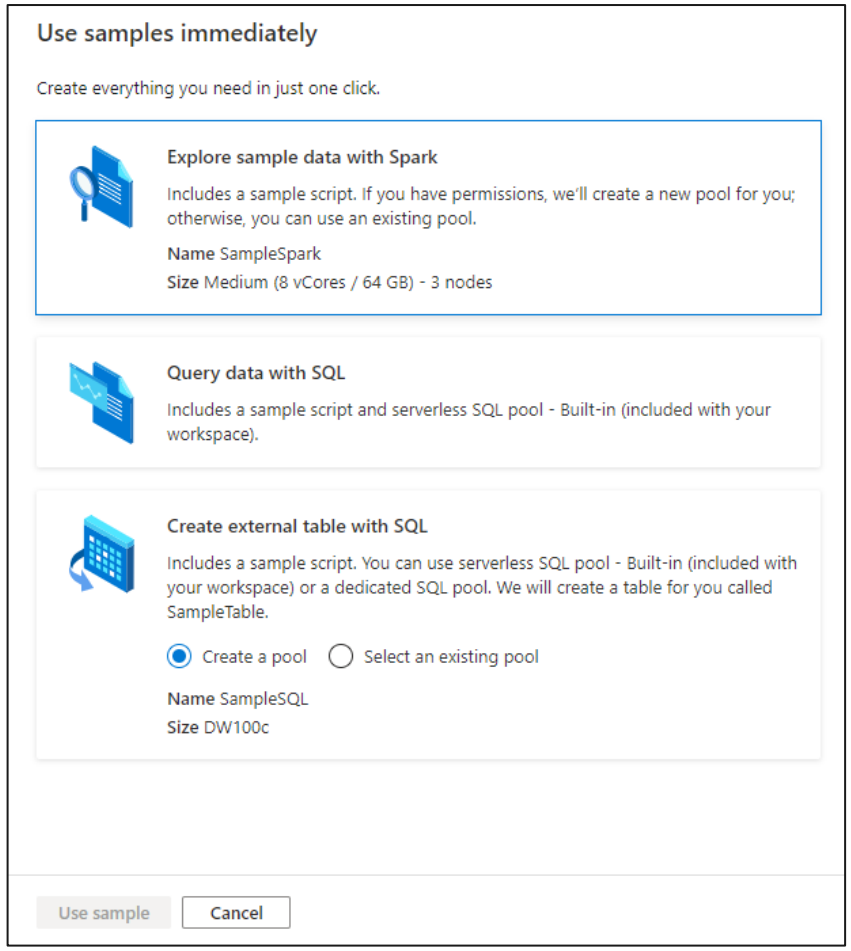
Home Hub

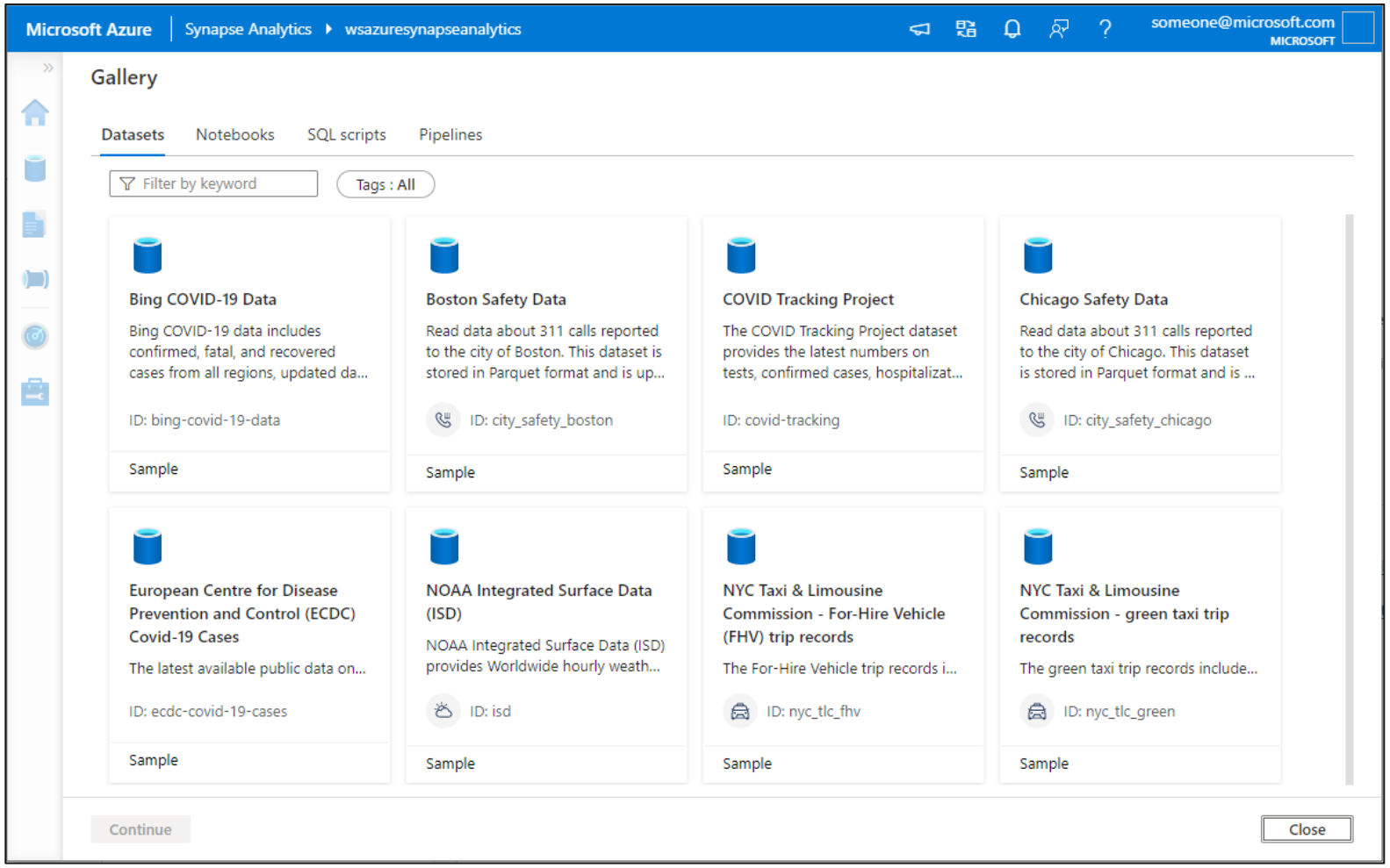
It is a starting point for the activities with key links to tasks, artifacts, documentation and sample artifacts for learning purpose



Knowledge center

Knowledge center offers open datasets, sample notebooks, SQL scripts and pipeline templates for easy start and learning



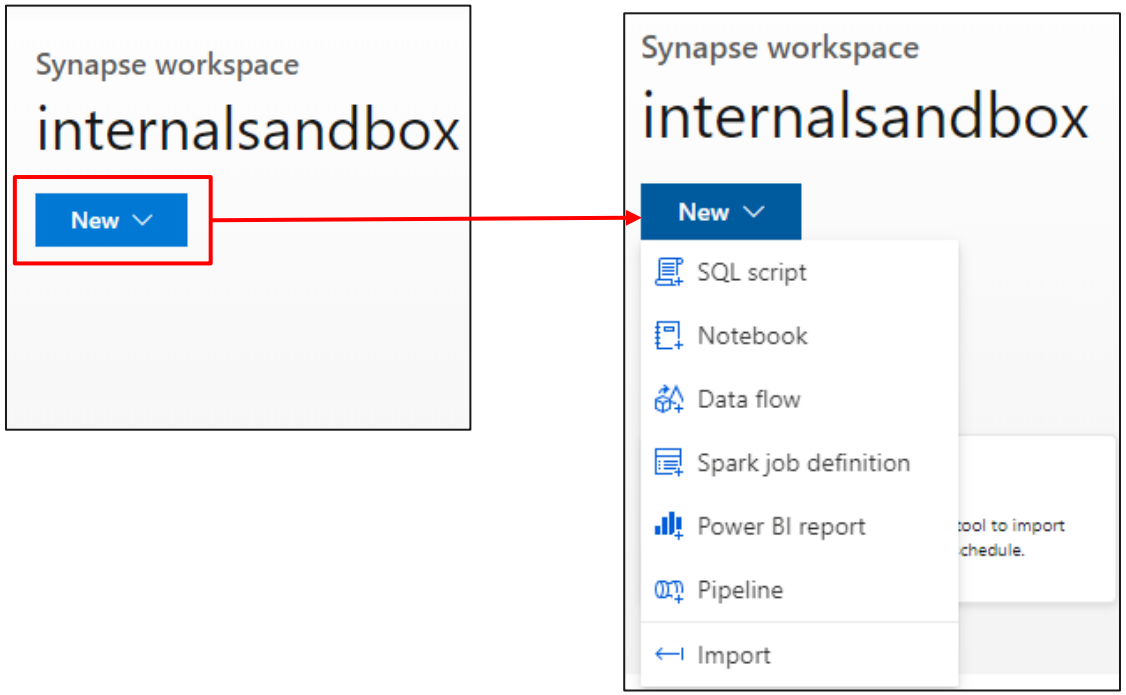


Home Hub

Overview

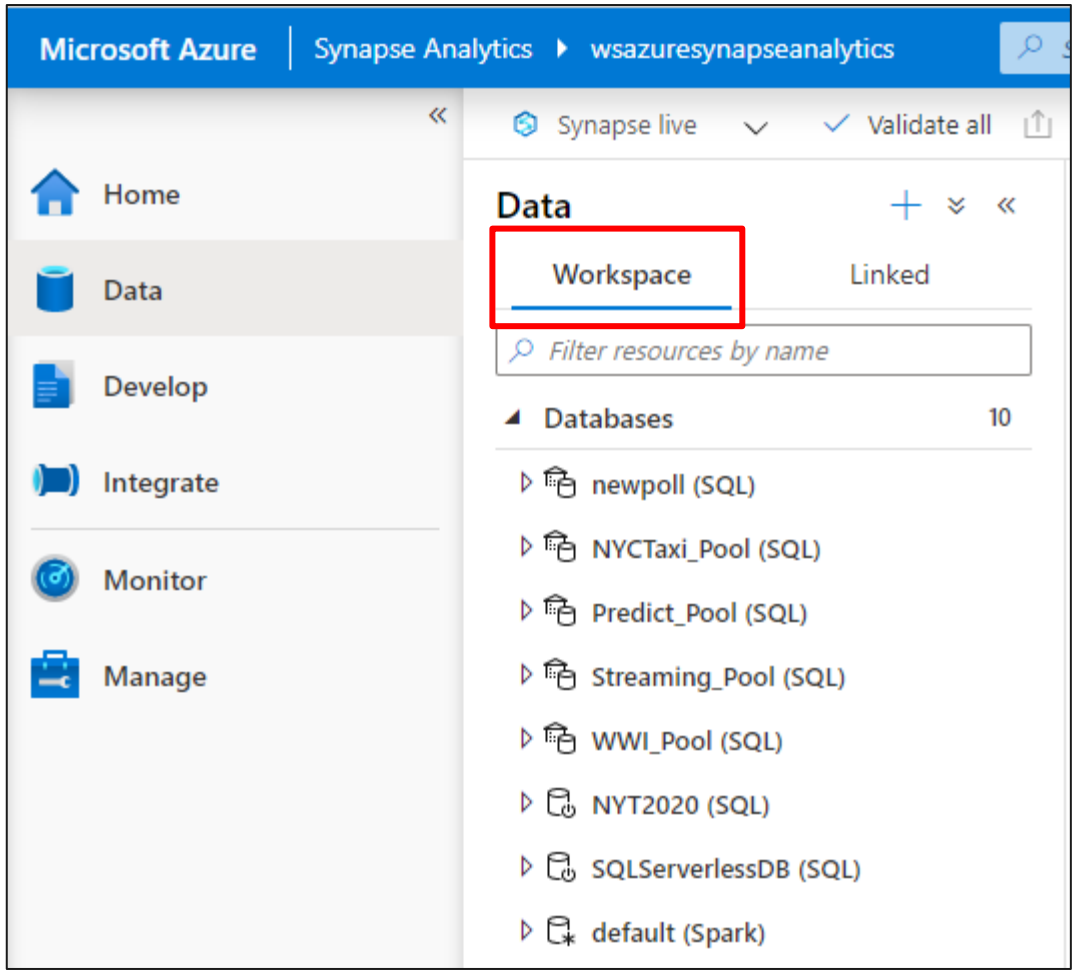
New dropdown – offers quickly start work item

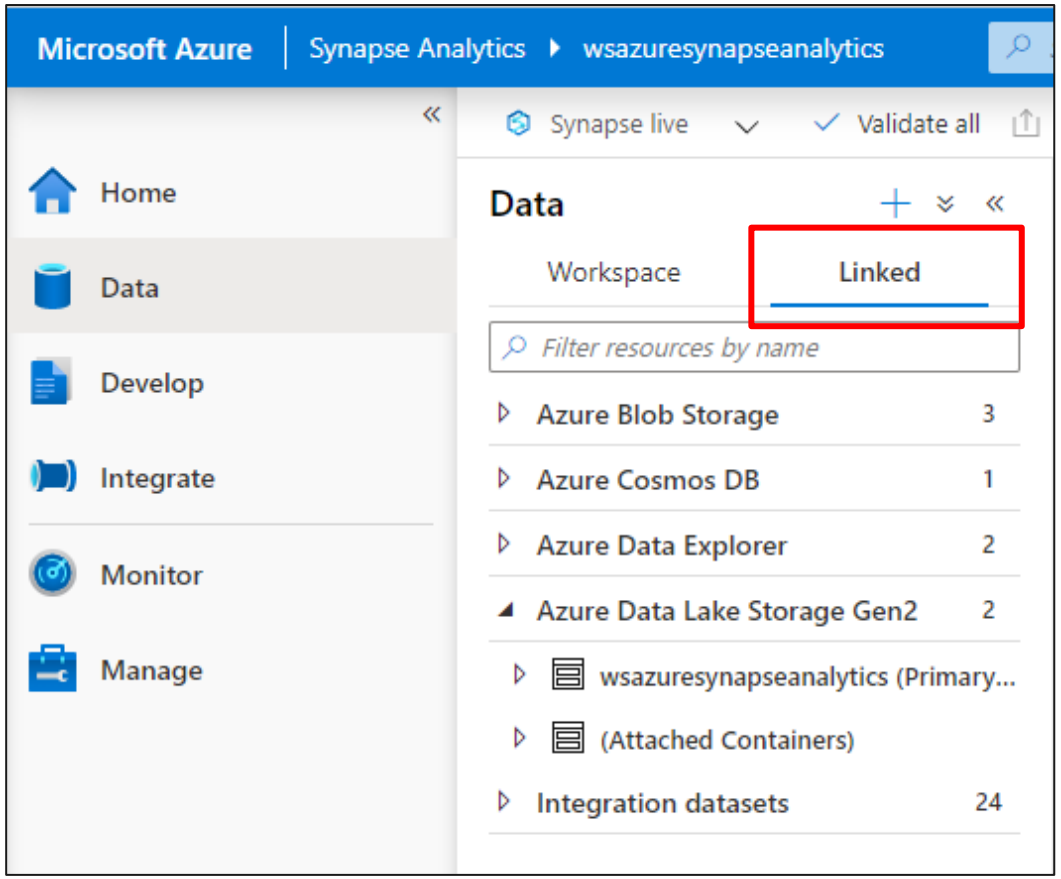
Recent & Pinned – Lists recently opened code artifacts. Pin selected ones for quick access



Data Hub

Explore data inside the workspace and in linked storage accounts





Data Hub – Linked Storage

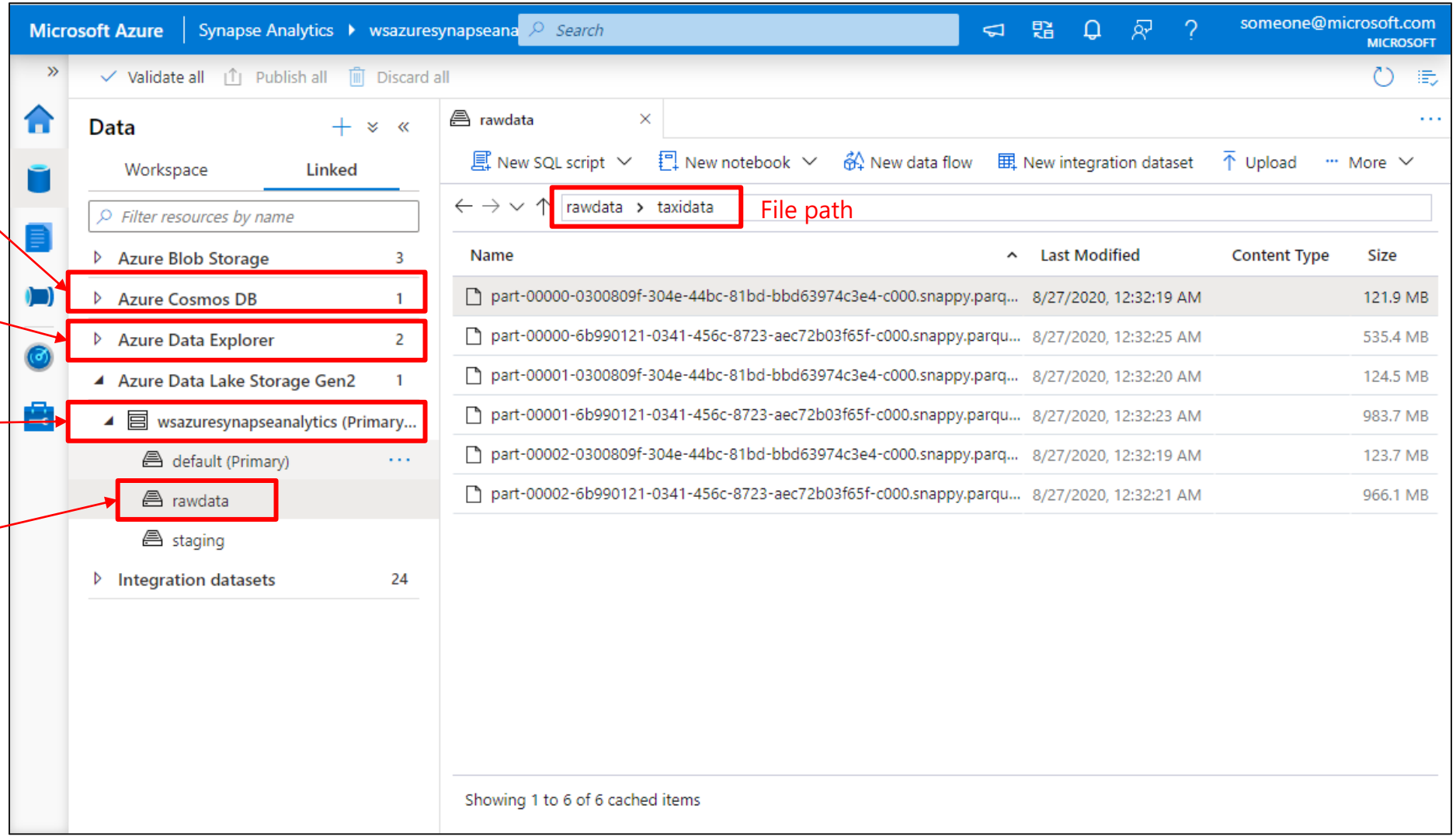
Browse Azure Data Lake Storage Gen2 accounts – filesystems, Azure Data Explorer – clusters, Azure Cosmos DB -containers

Linked Cosmos DB Analytical Store

Linked Azure Data Explorer

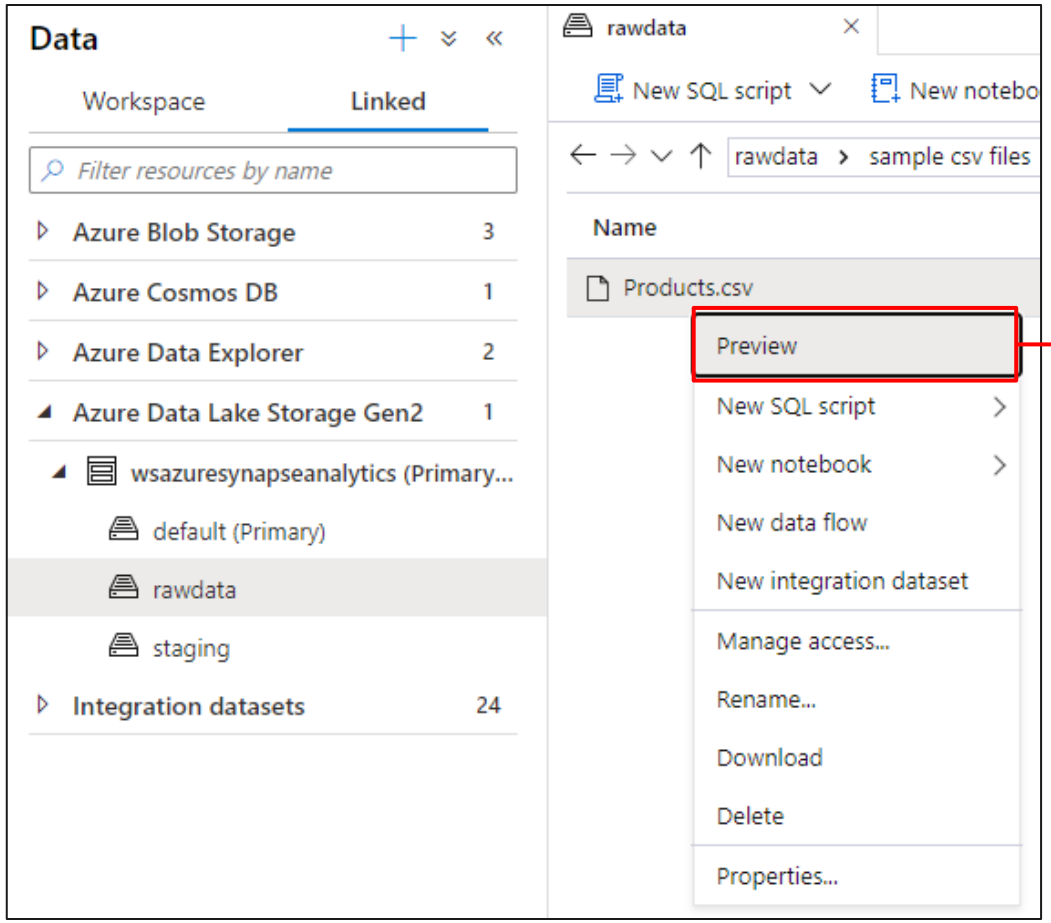
Linked ADLS Gen2 Account

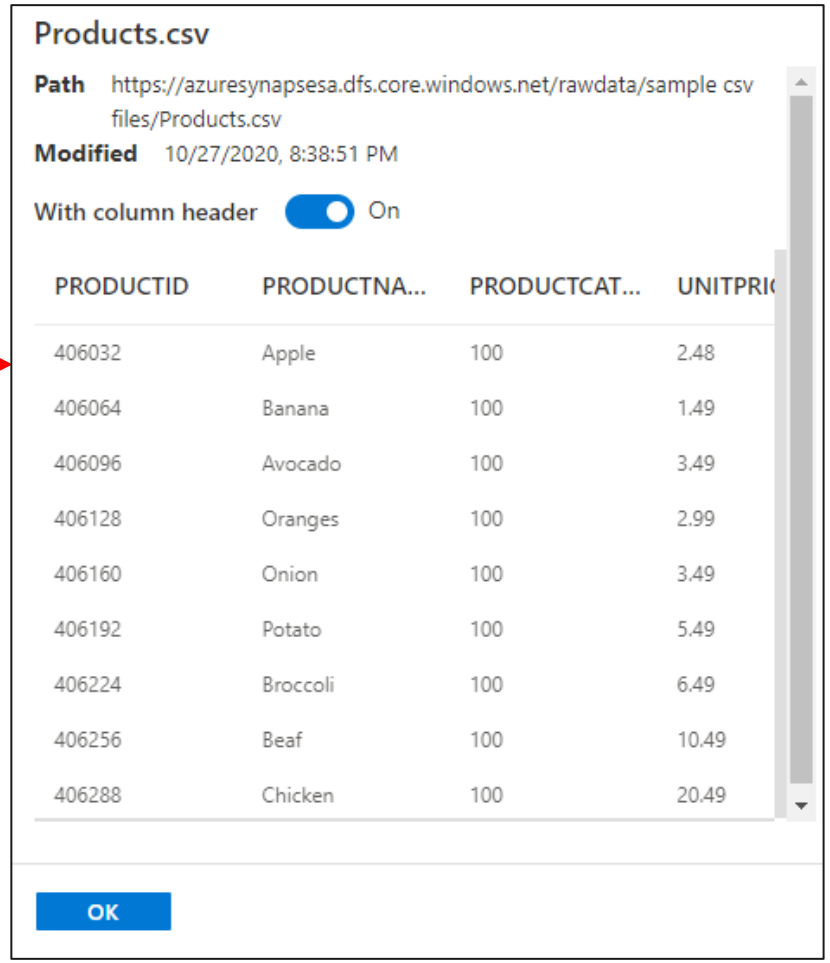
Container (filesystem)



Data Hub – Storage accounts

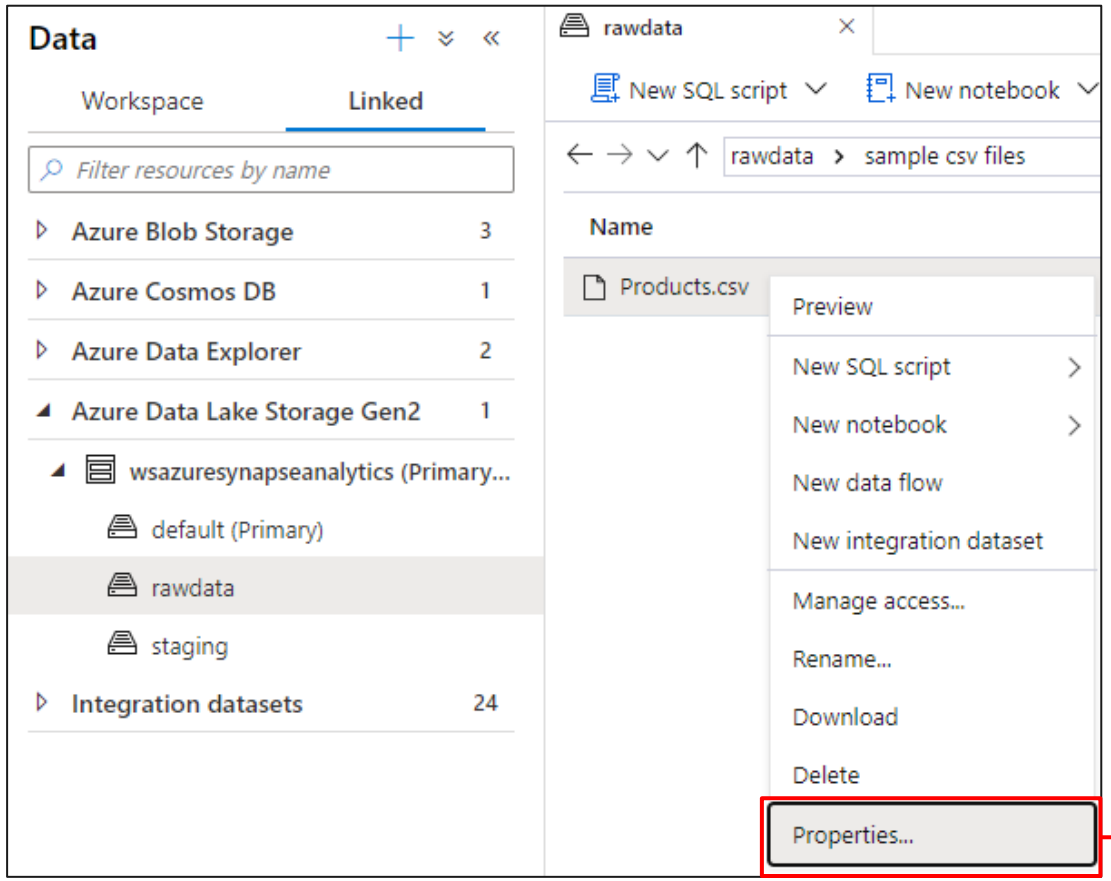
Preview a sample of your data

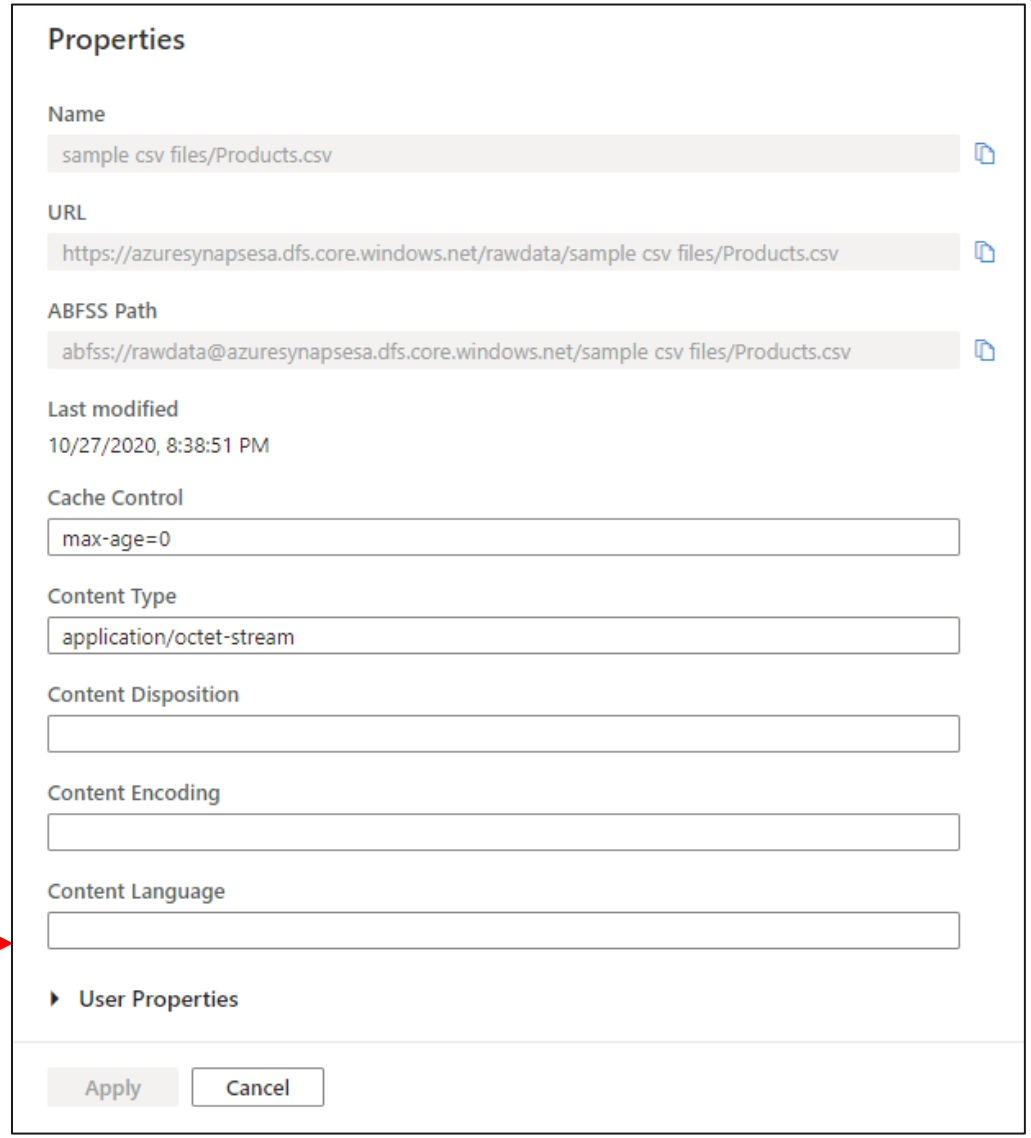




Data Hub – Storage accounts

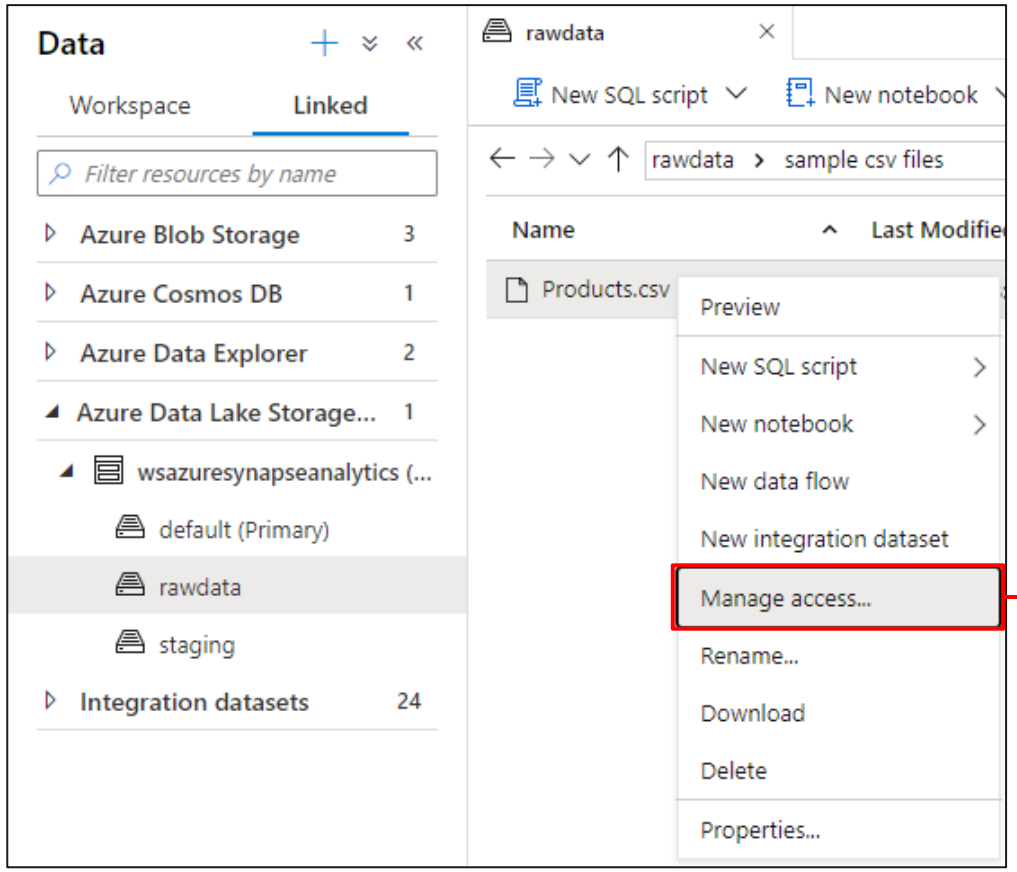
See basic file properties

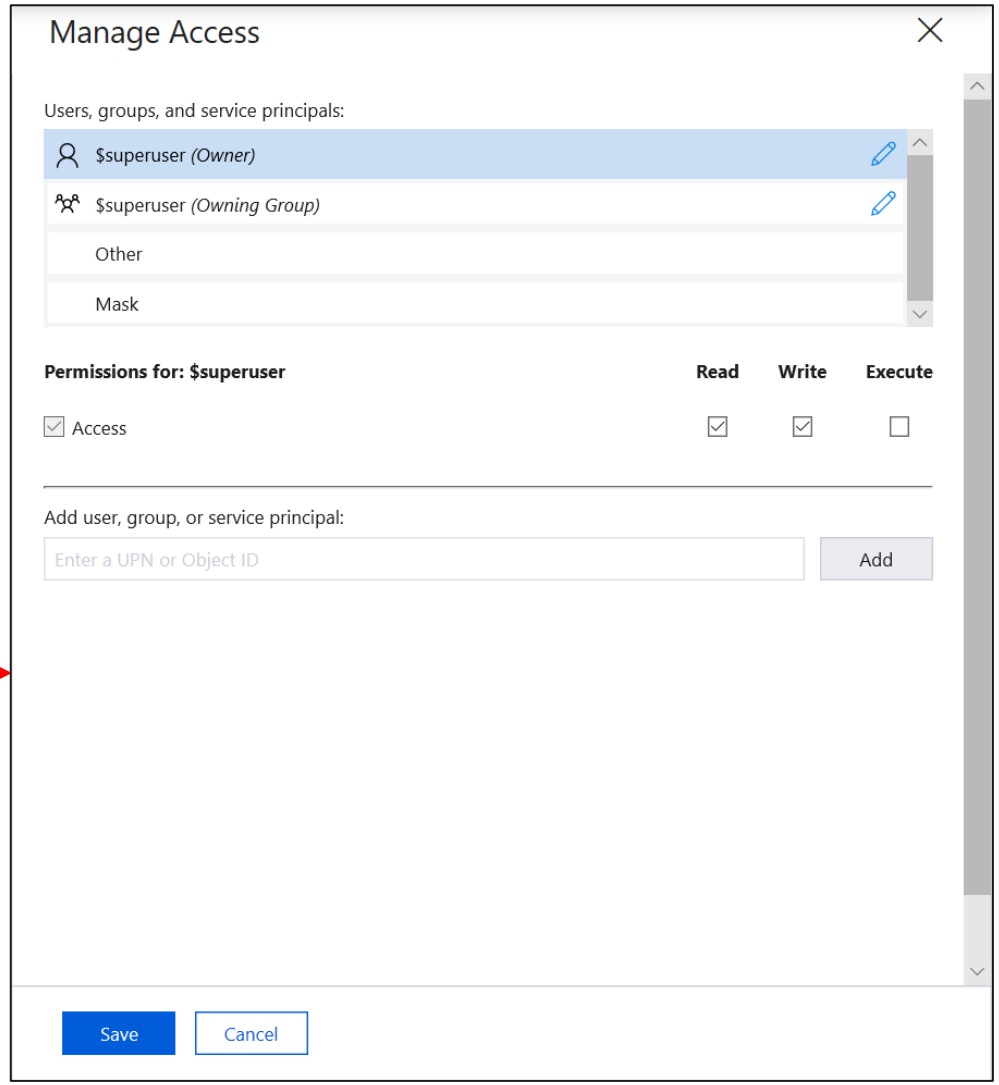




Data Hub – Storage accounts

Manage Access - Configure standard POSIX ACLs on files and folders

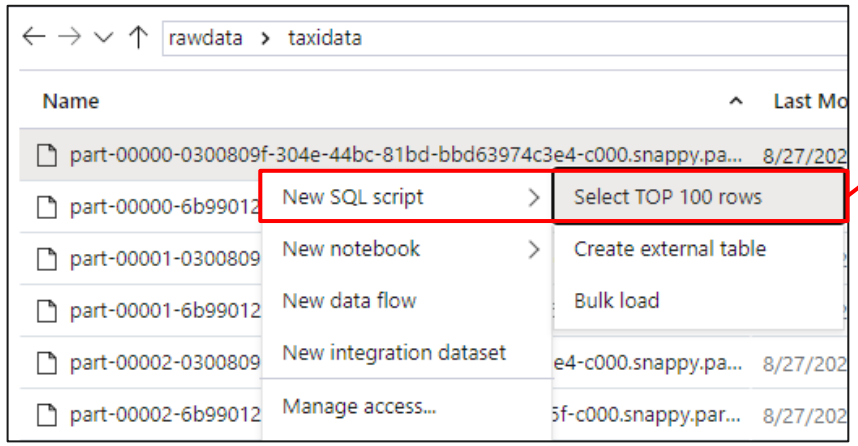


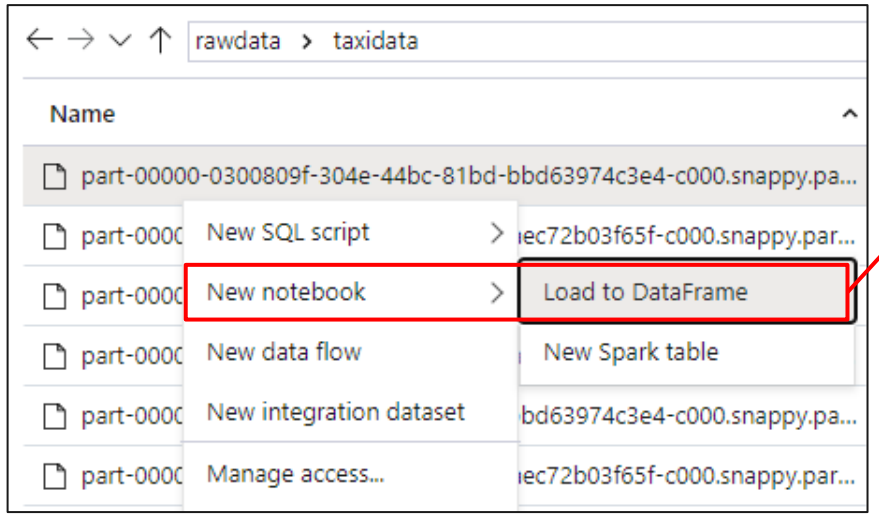


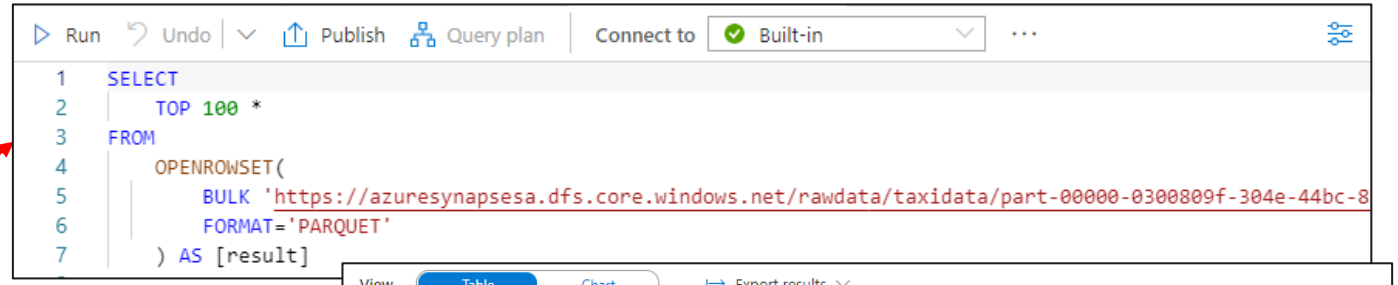
Data Hub – Storage accounts

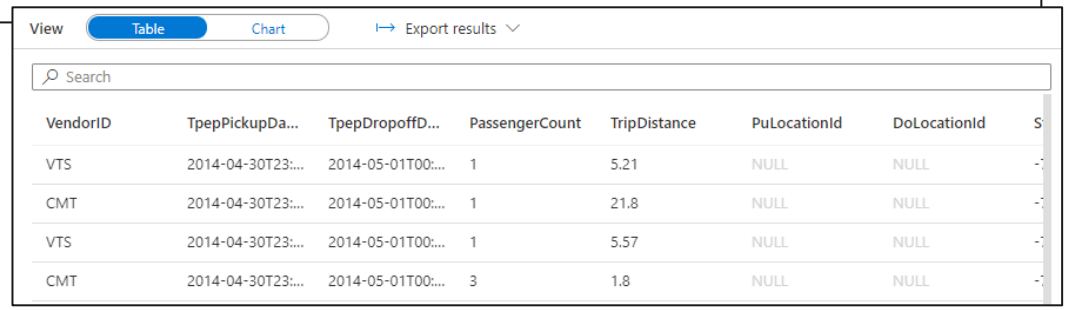
Two simple gestures to start analyzing with SQL scripts or with notebooks.

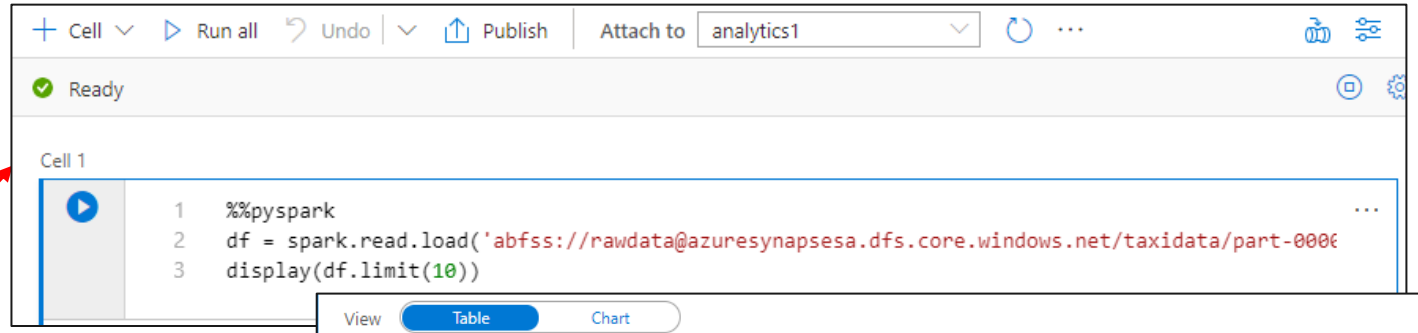
T-SQL or PySpark auto-generated.

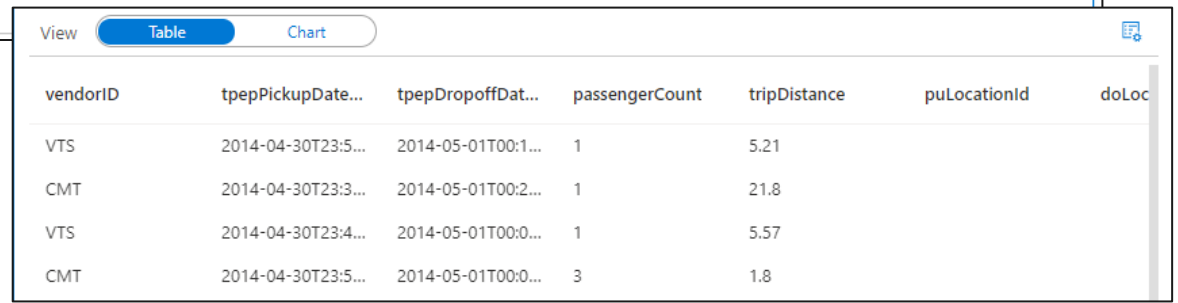






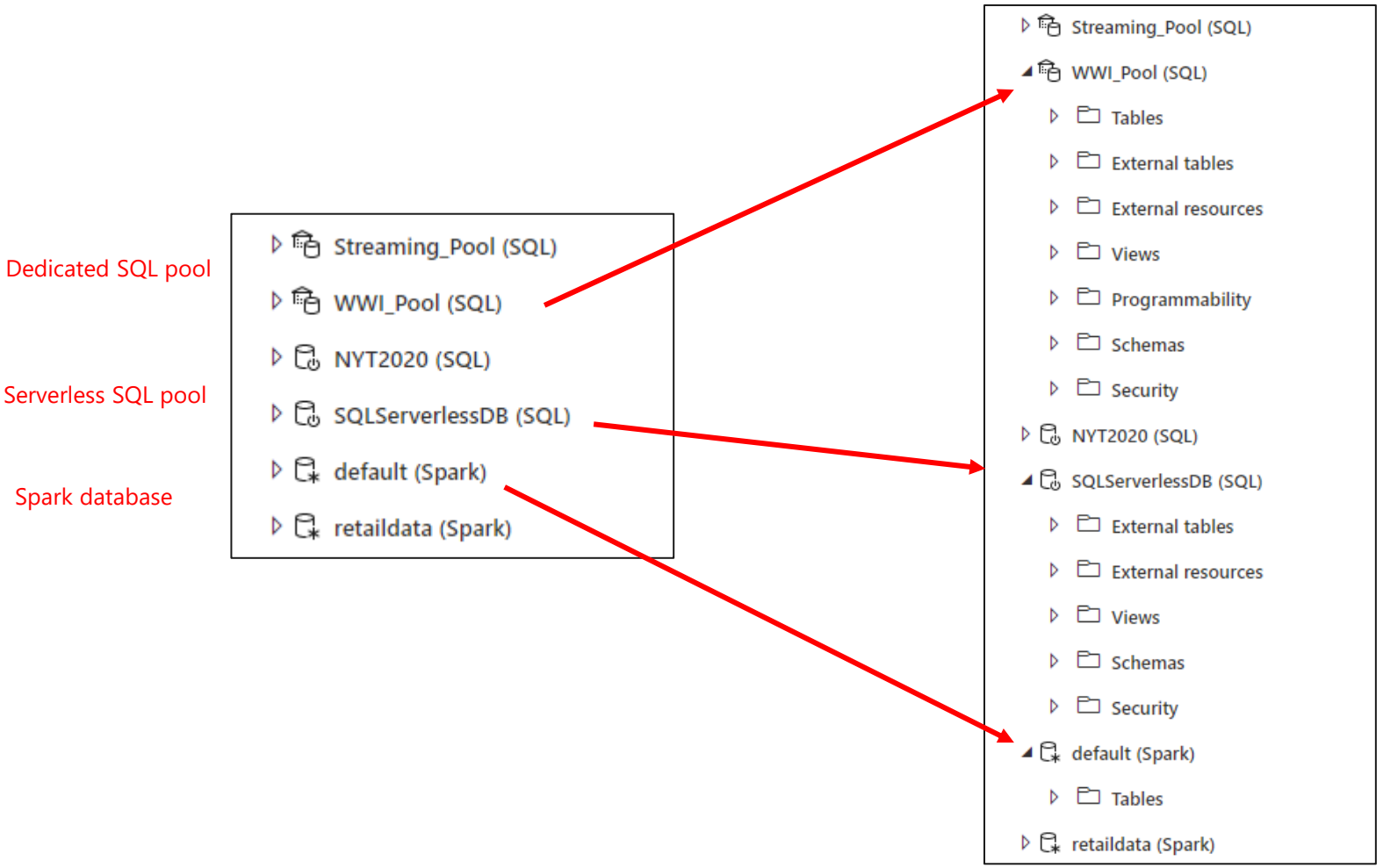






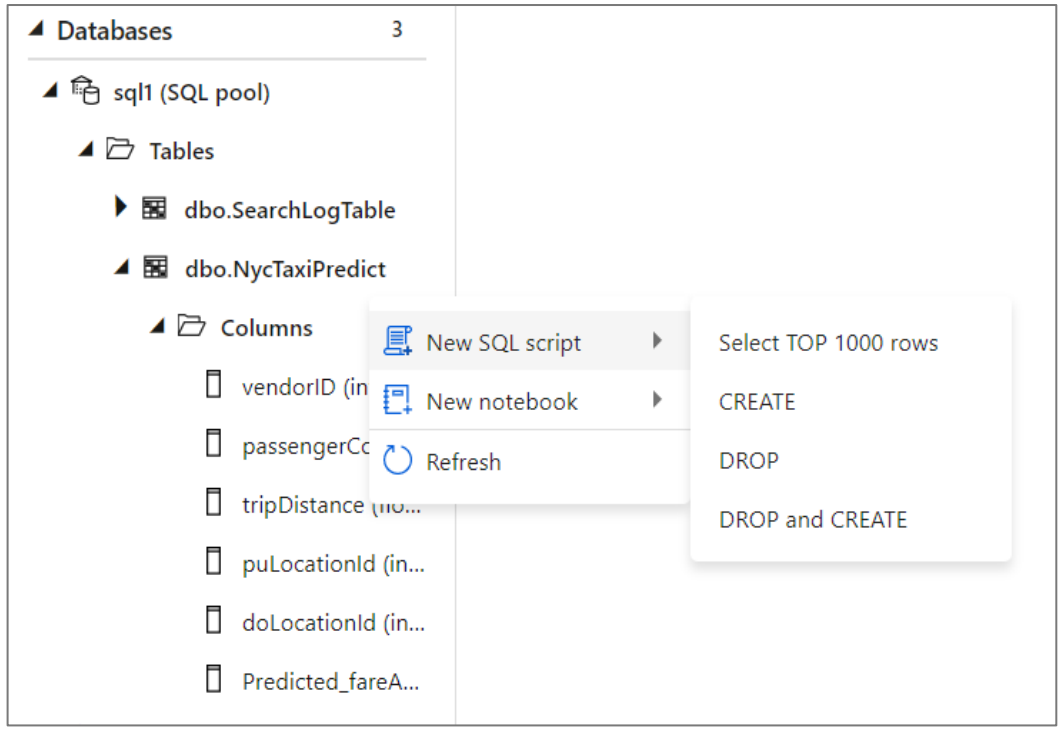
Data Hub – Databases

Explore the different kinds of databases that exist in a workspace.

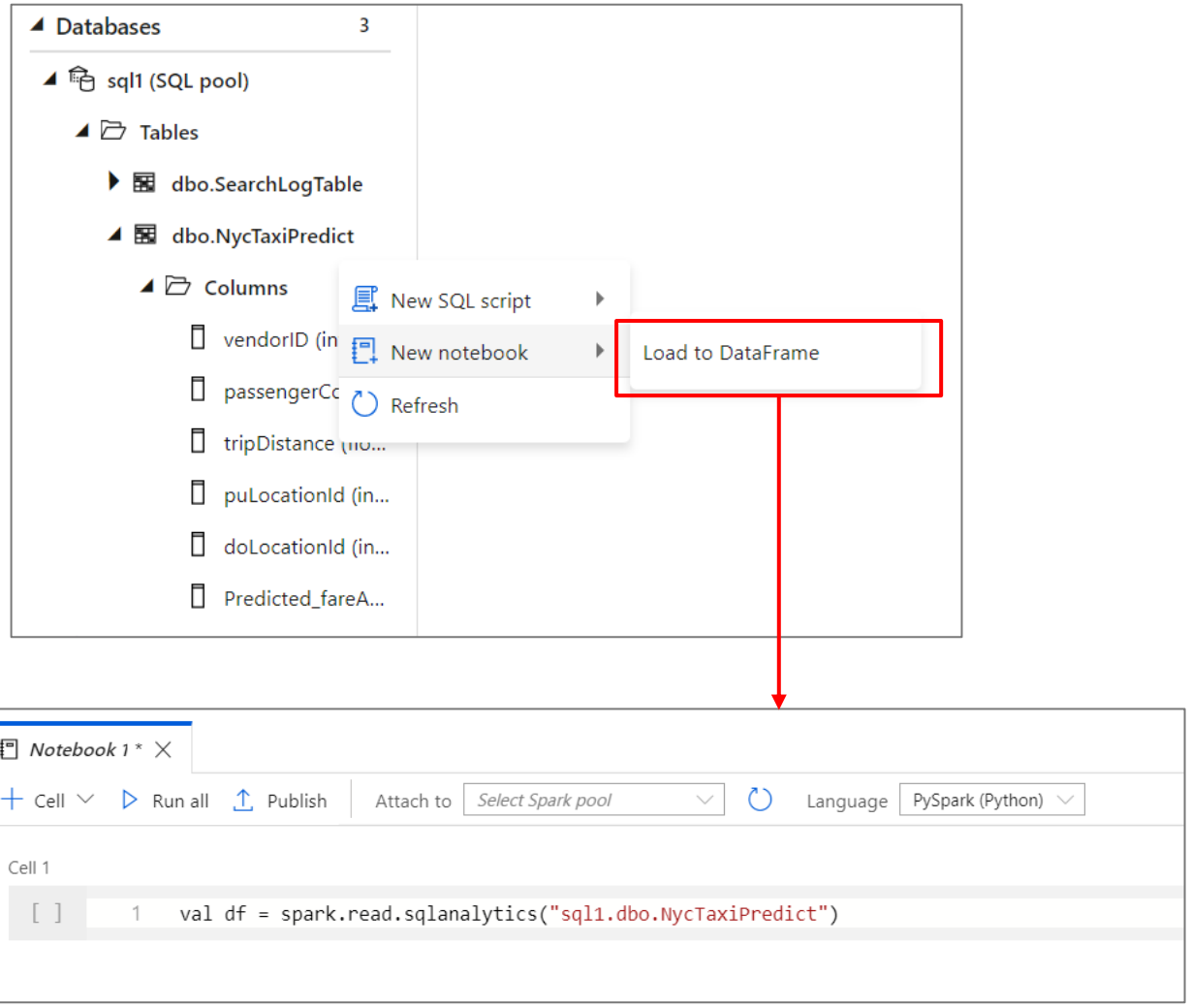


Data Hub – Databases

Familiar gesture to generate T-SQL scripts from SQL metadata objects such as tables.

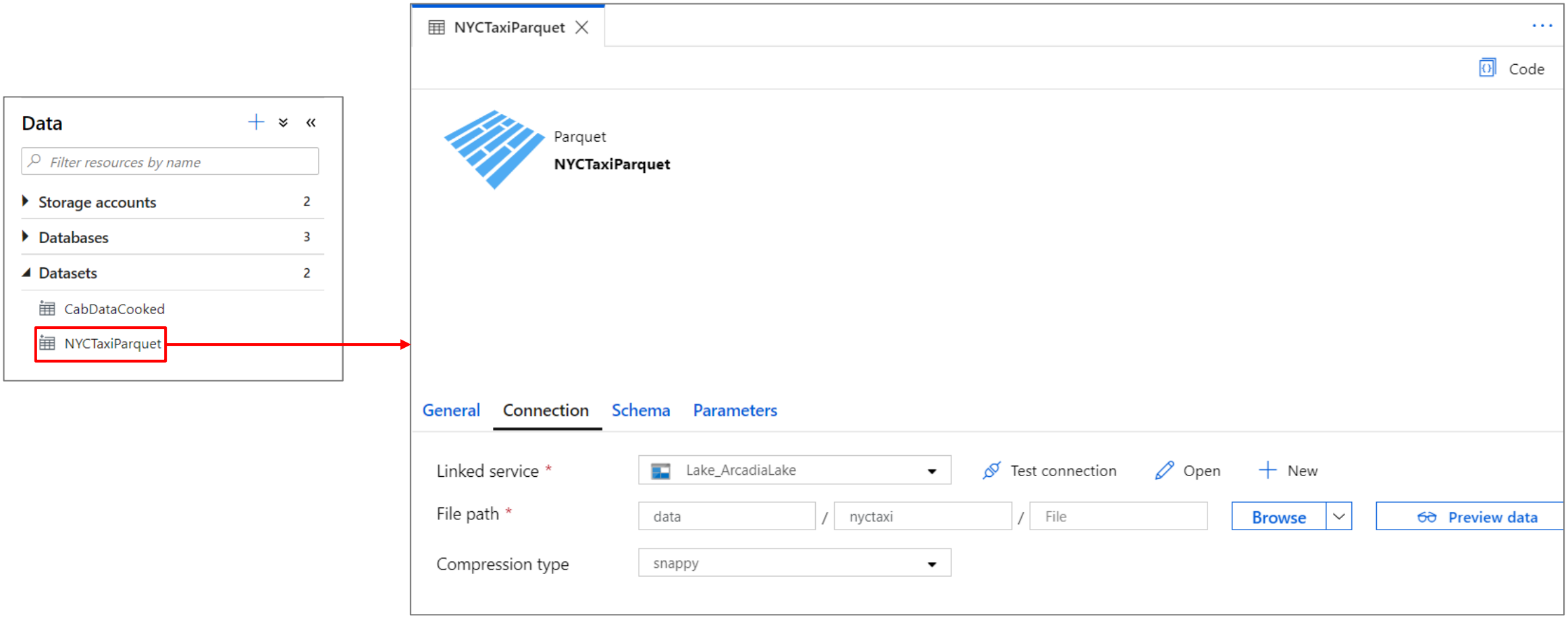


Starting from a table, auto-generate a single line of PySpark code that makes it easy to load a SQL table into a Spark dataframe



Data Hub – Datasets

Orchestration datasets describe data that is persisted. Once a dataset is defined, it can be used in pipelines and sources of data or as sinks of data.



Develop Hub

Overview

It provides development experience to query, analyze, model data

Benefits

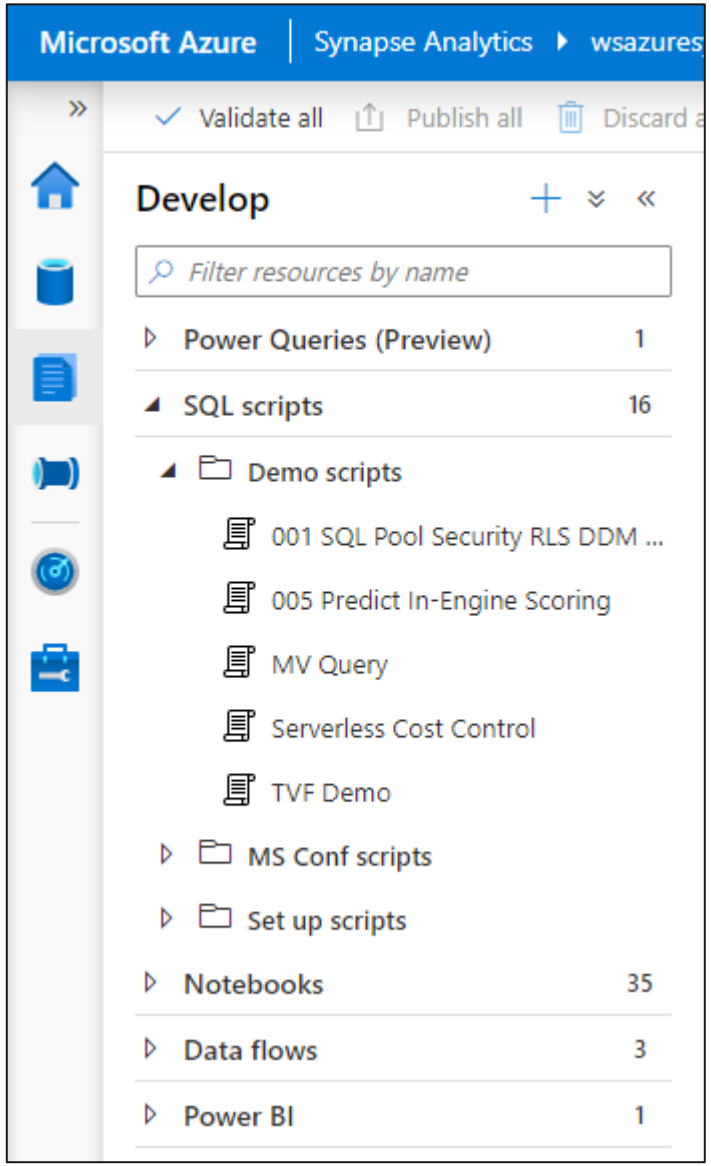
Multiple languages to analyze data under one umbrella

Switch over notebooks and scripts without loosing content

Code intellisense offers reliable code development

Create insightful visualizations

Organize artifacts in folders and sub-folders



Develop Hub - SQL scripts

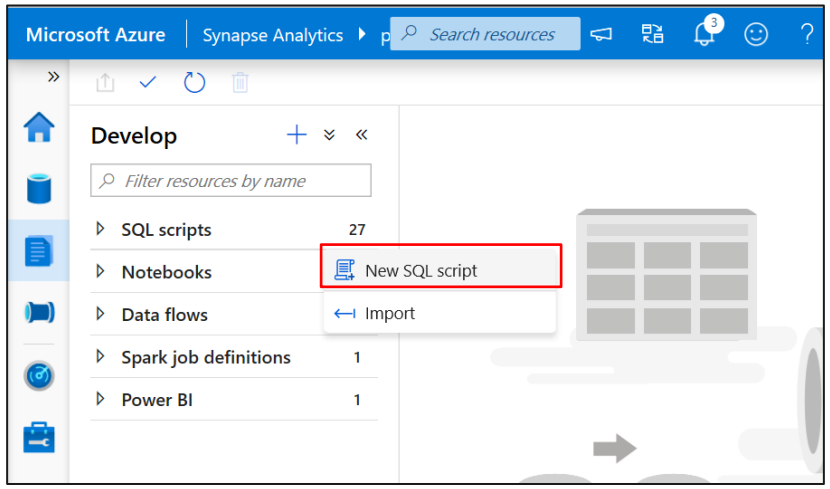
SQL Script

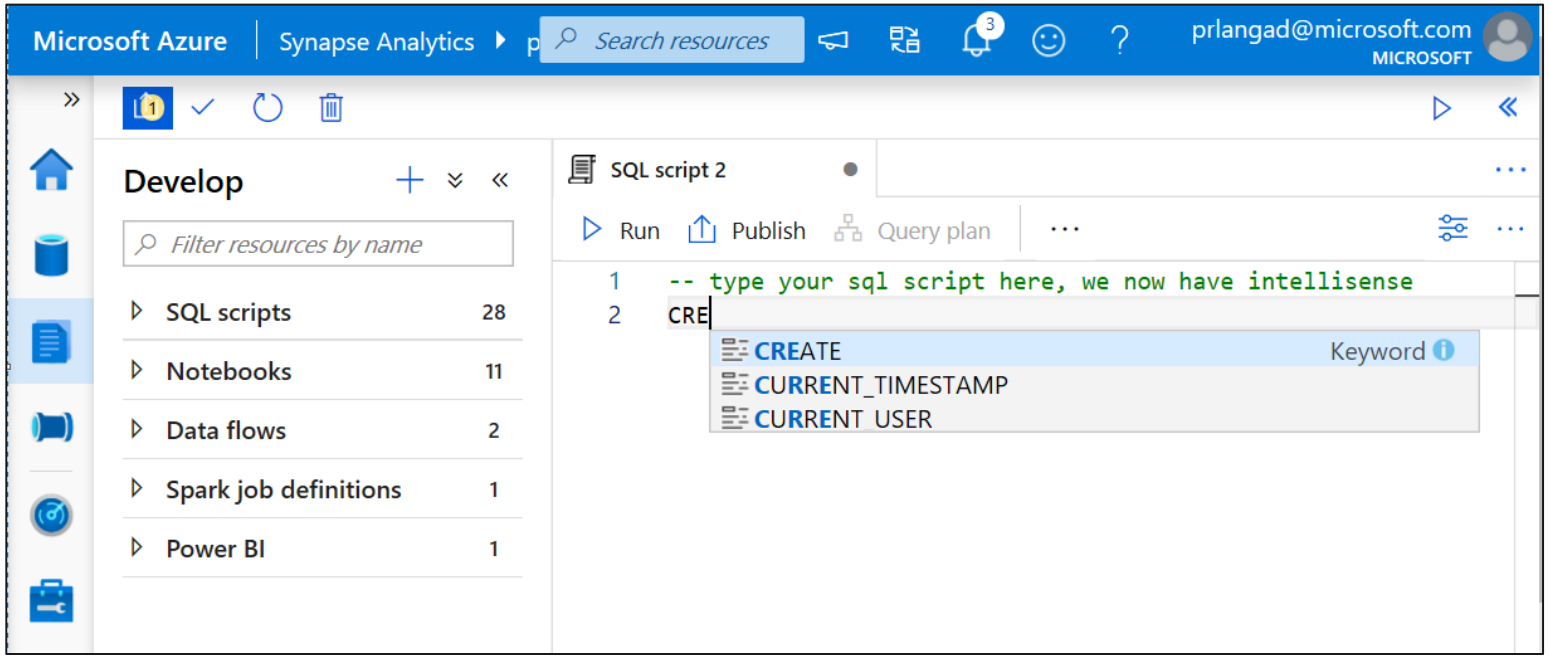
Authoring SQL Scripts

Execute SQL script on dedicated SQL pool or serverless SQL pool

Commit individual SQL script or multiple SQL scripts through Commit all feature

Language support and intellisense

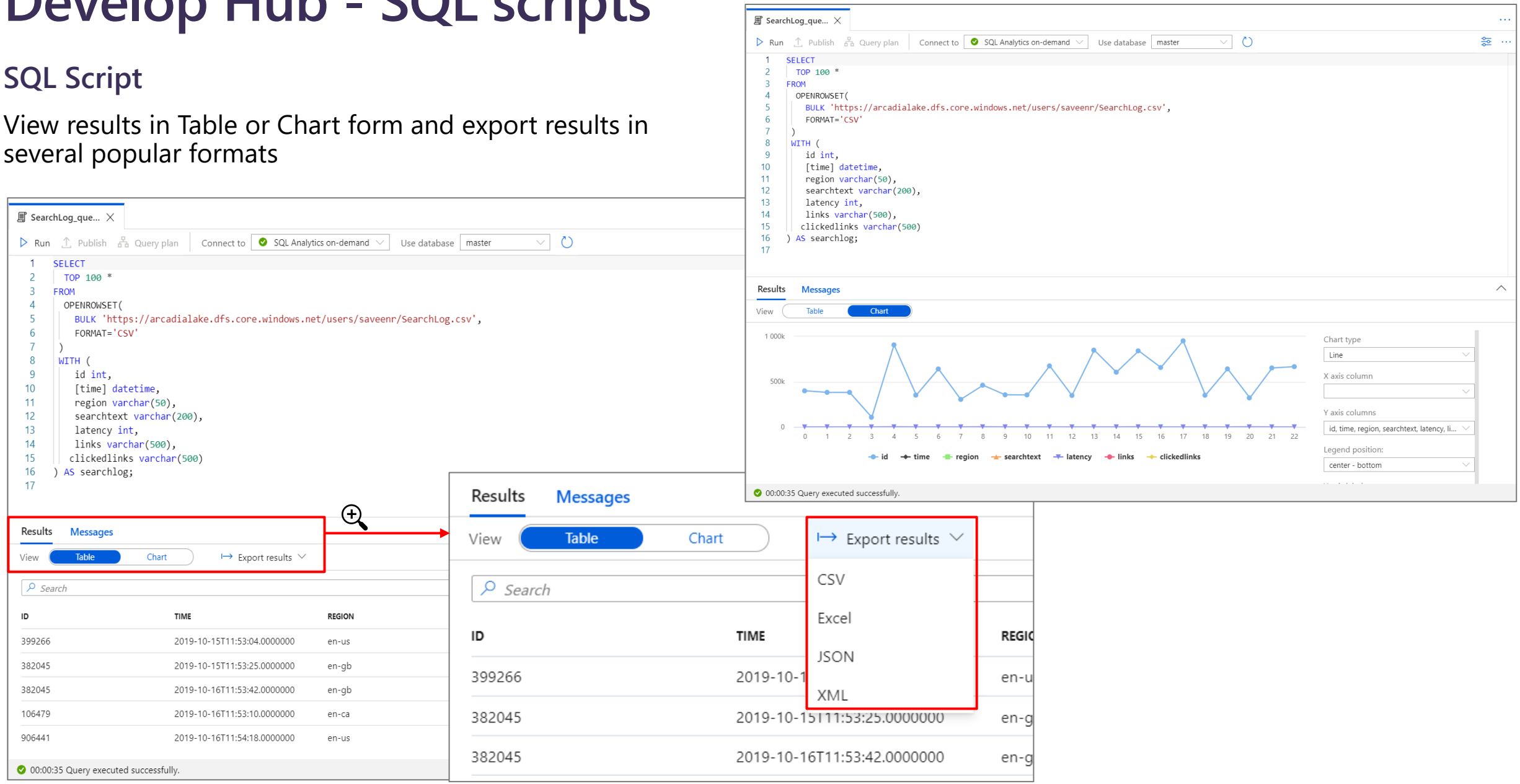




Develop Hub - SQL scripts

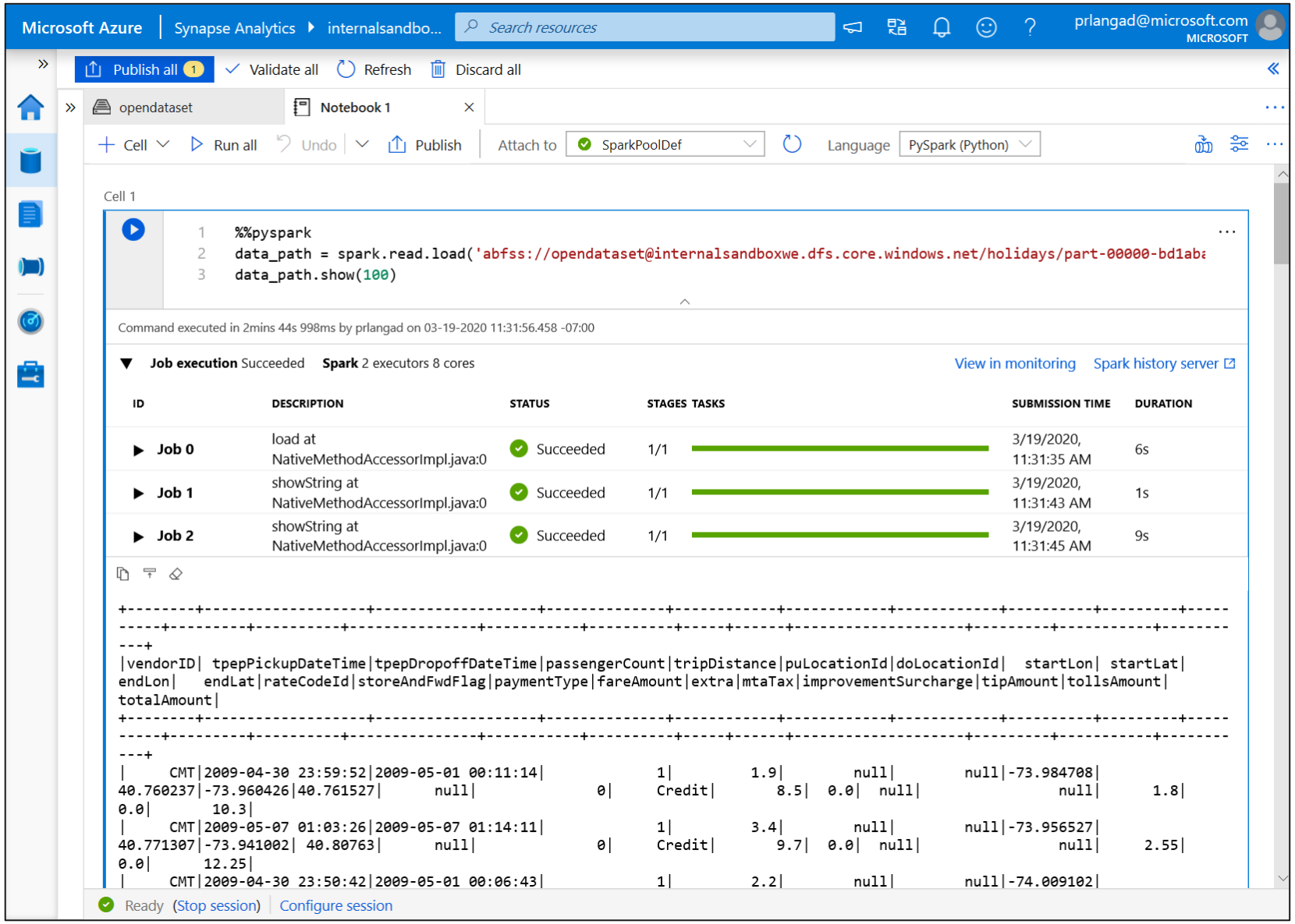
SQL Script

View results in Table or Chart form and export results in several popular formats



Develop Hub - Notebooks

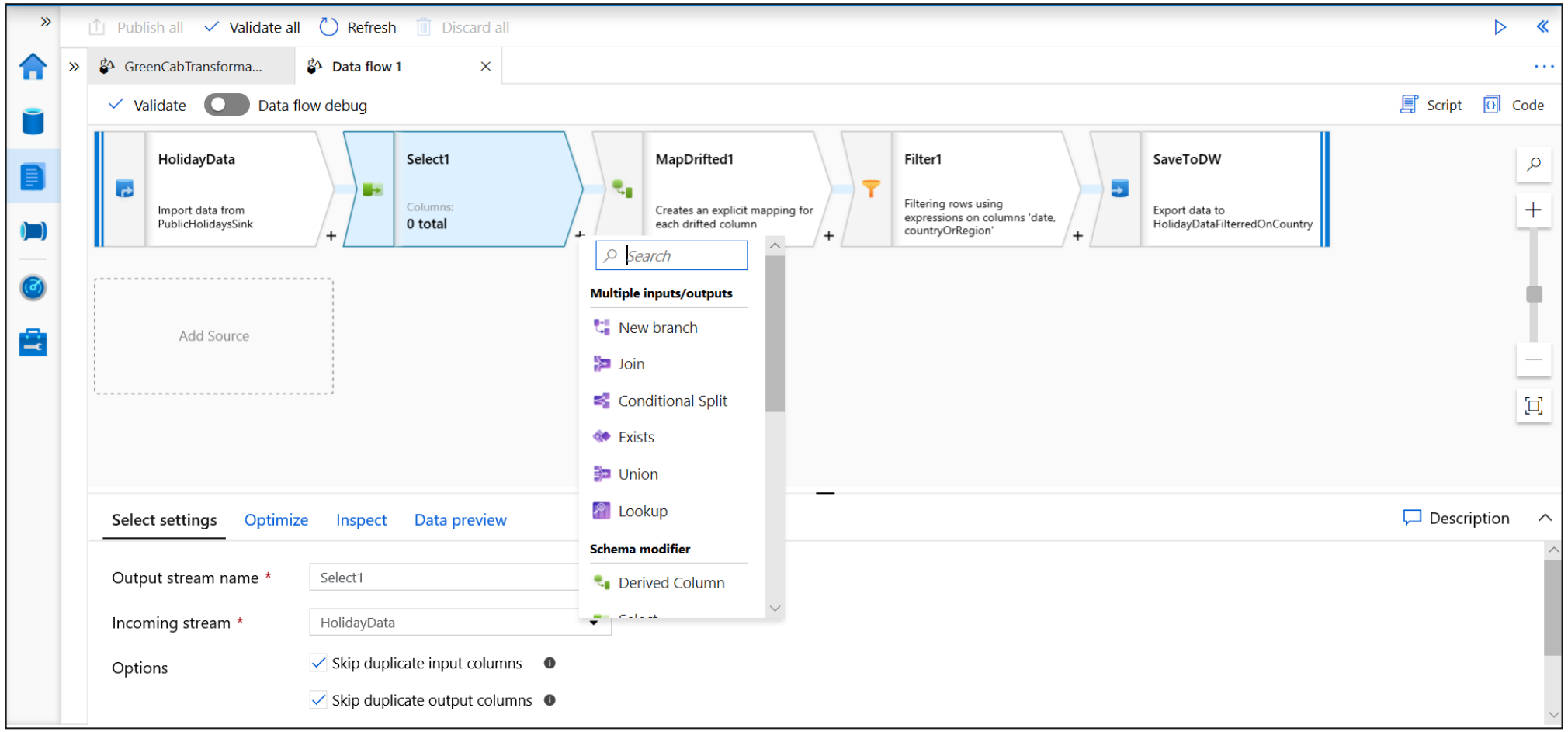
As notebook cells run, the underlying Spark application status is shown. Providing immediate feedback and progress tracking.



Develop Hub - Data Flows

Data flows are a visual way of specifying how to transform data.

Provides a code-free experience.



Develop Hub – Power BI

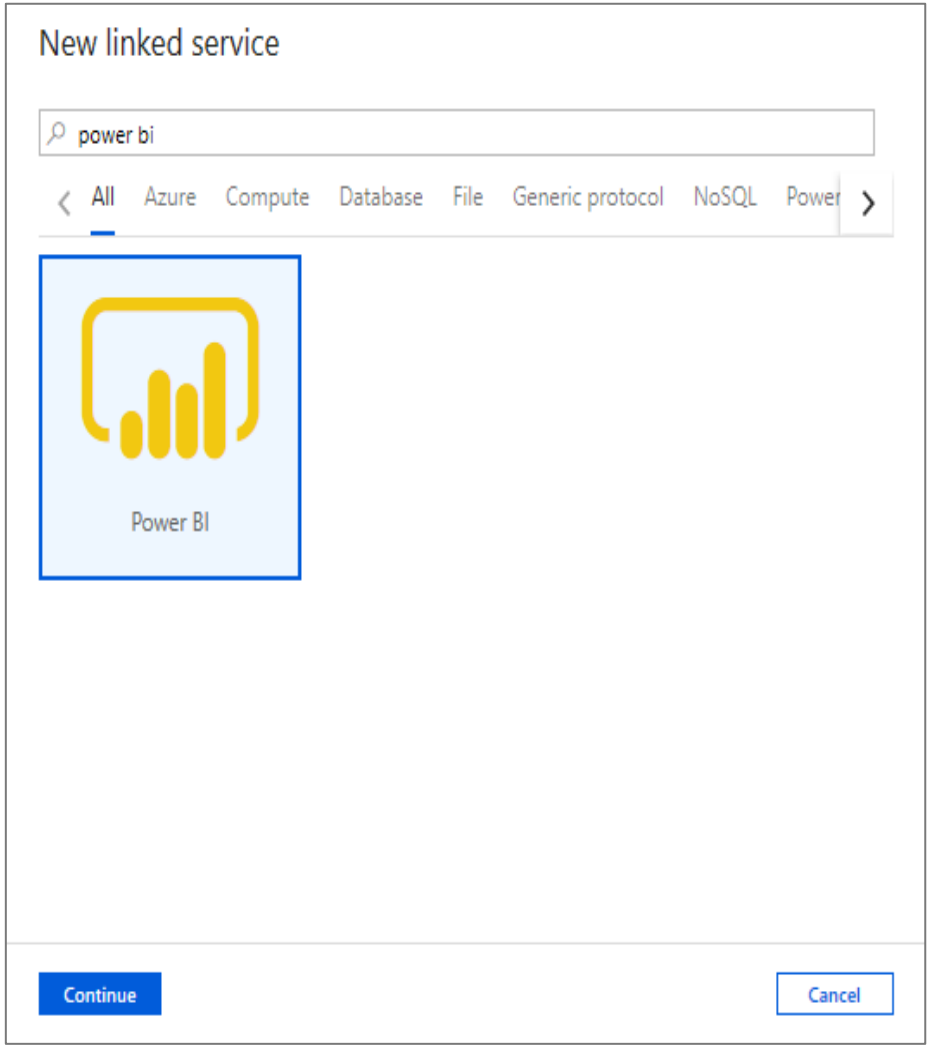
Overview

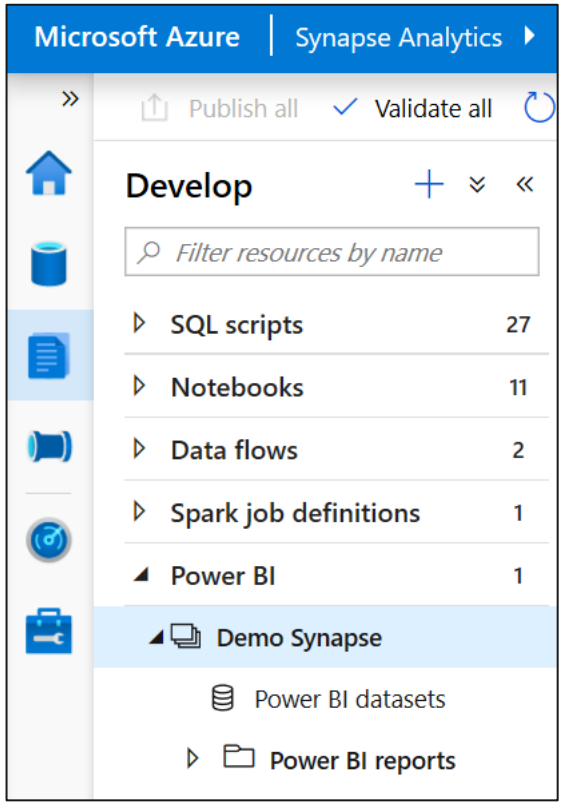
Create Power BI reports in the workspace

Provides access to published reports in the workspace

Update reports real time from Synapse workspace to get it reflected on Power BI service

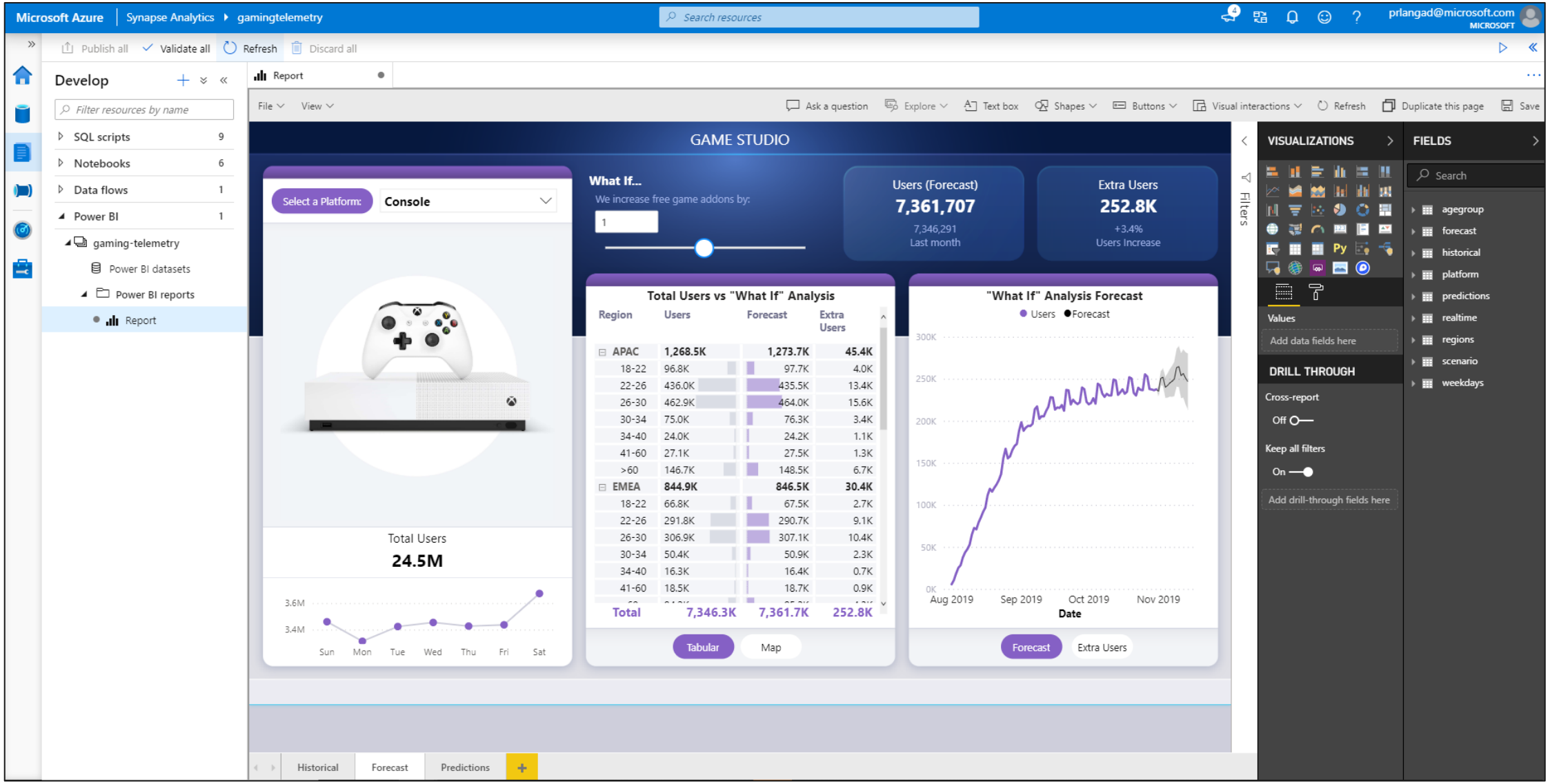
Visually explore and analyze data





Develop Hub – Power BI

View published reports in Power BI workspace

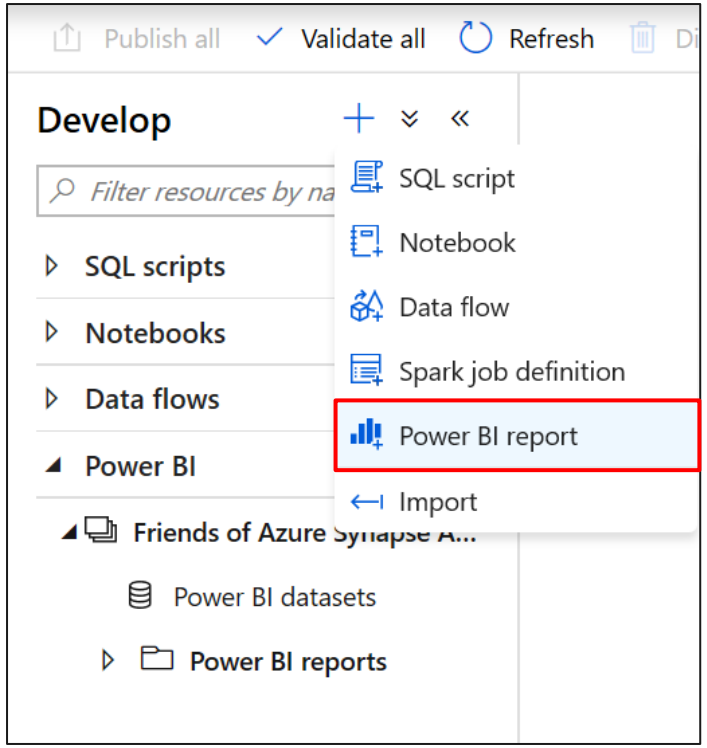


Develop Hub – Power BI

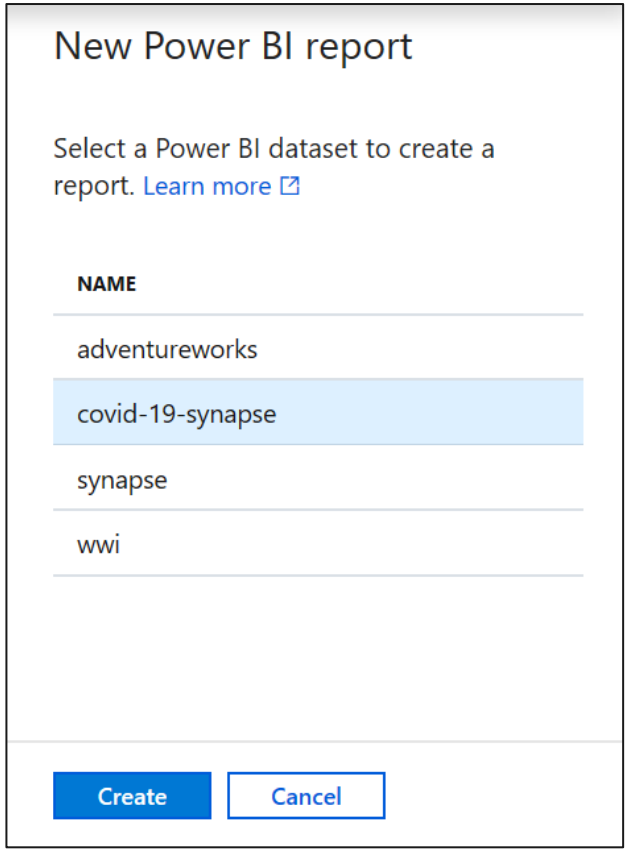
Create new reports from existing published Power BI datasets

Create new Power BI datasets

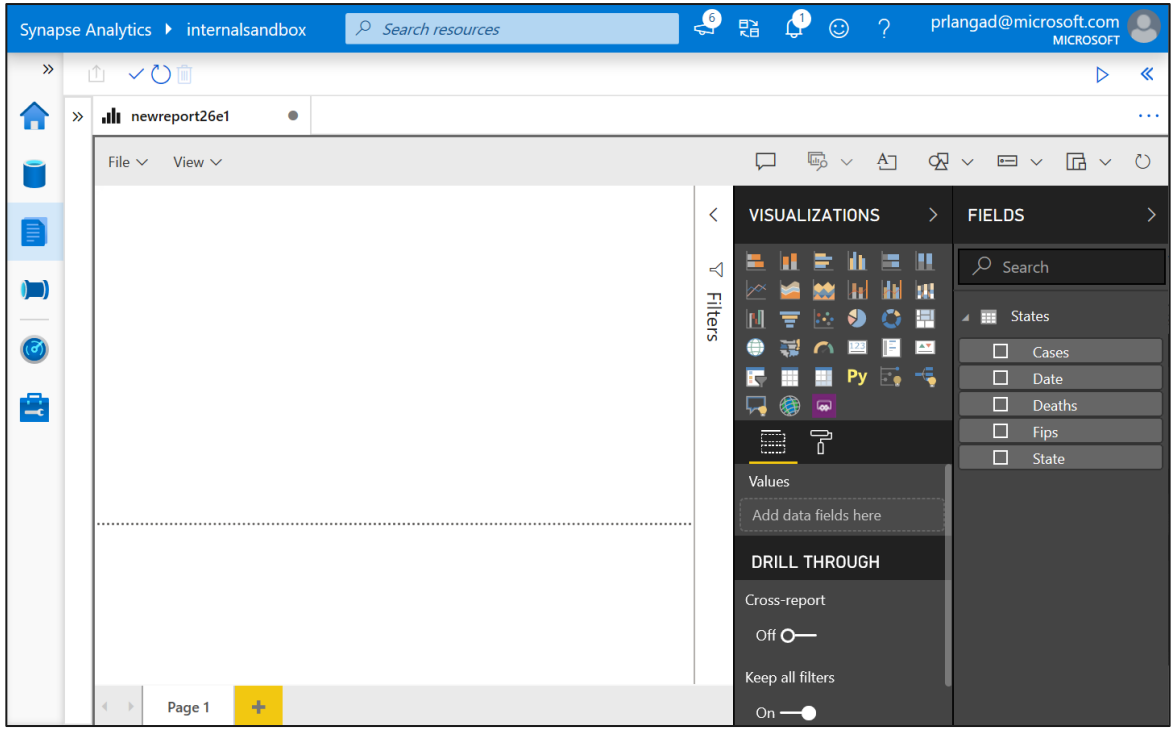
1



2

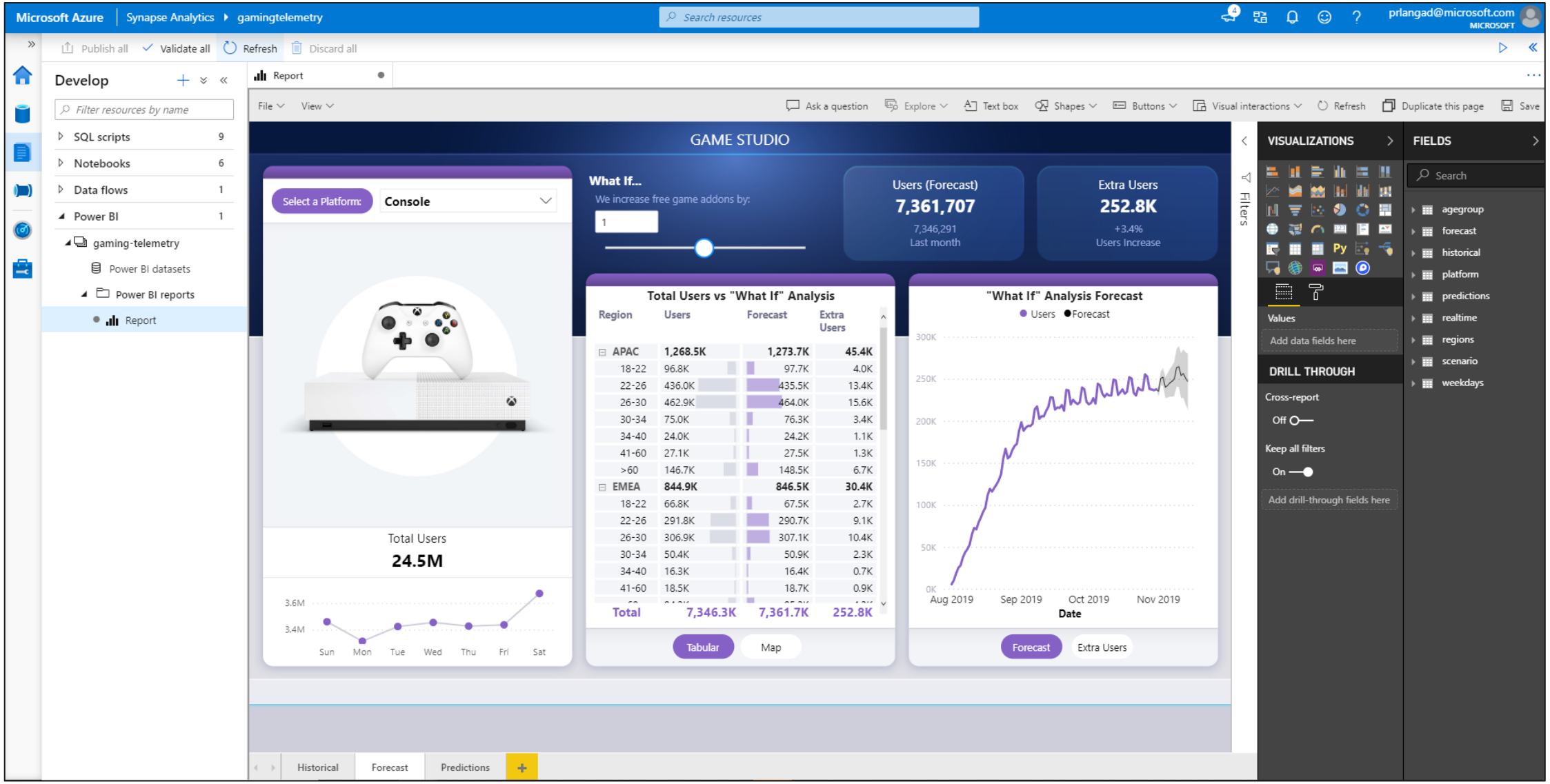


3



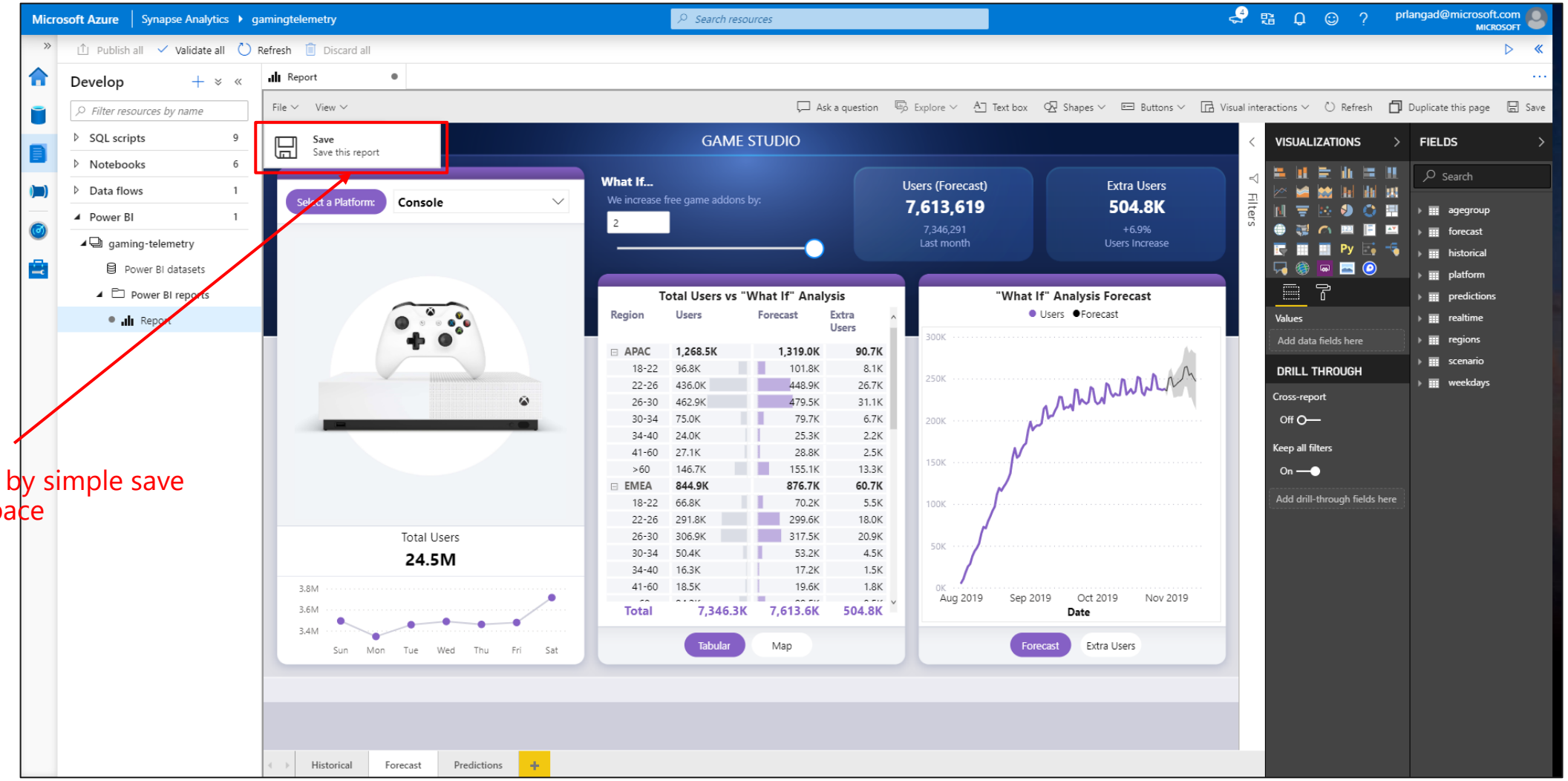
Develop Hub – Power BI

Edit reports in Synapse workspace



Develop Hub – Power BI

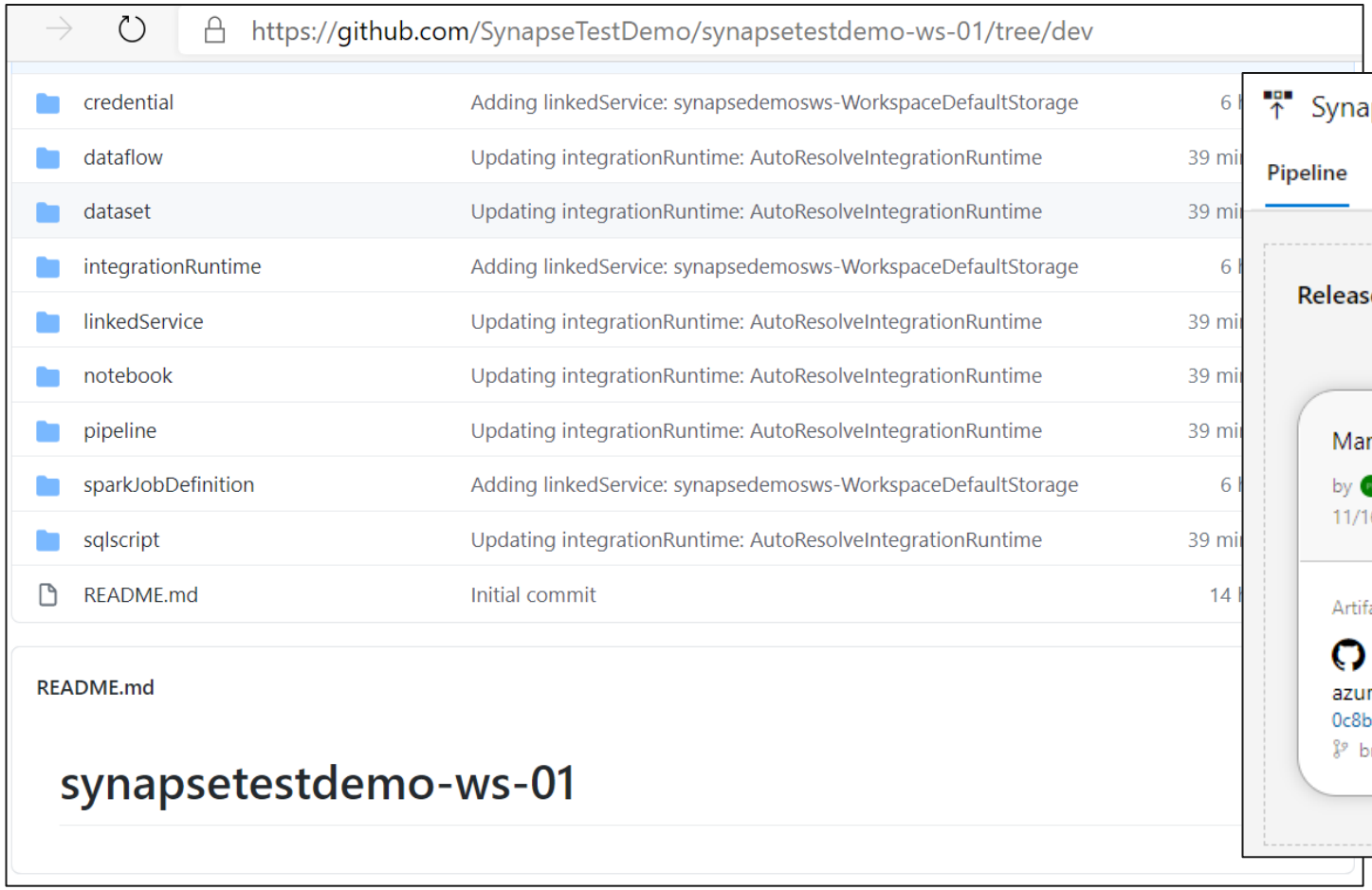
Publish edited reports in Synapse workspace to Power BI workspace

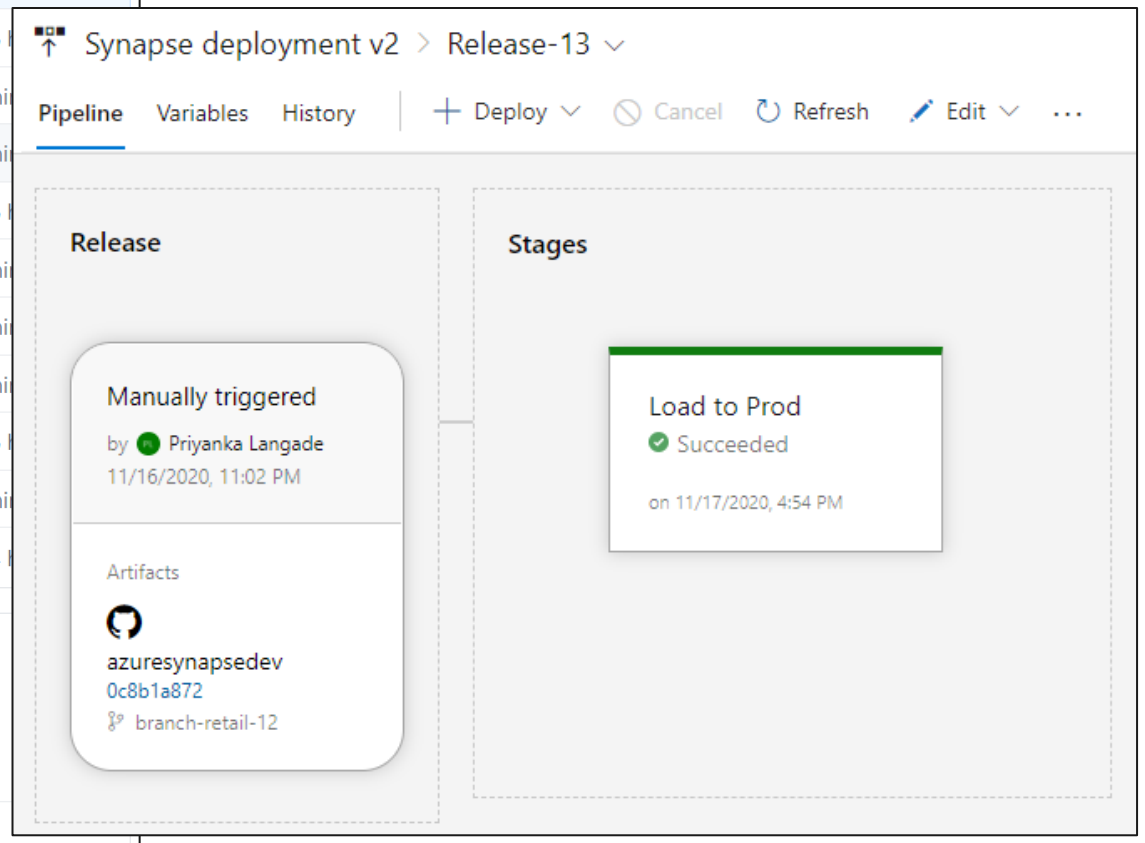


Publish changes by simple save report in workspace

Develop – CI/CD

Commit artifacts to source-controlled repository and operationalize release pipelines with Synapse deployment task

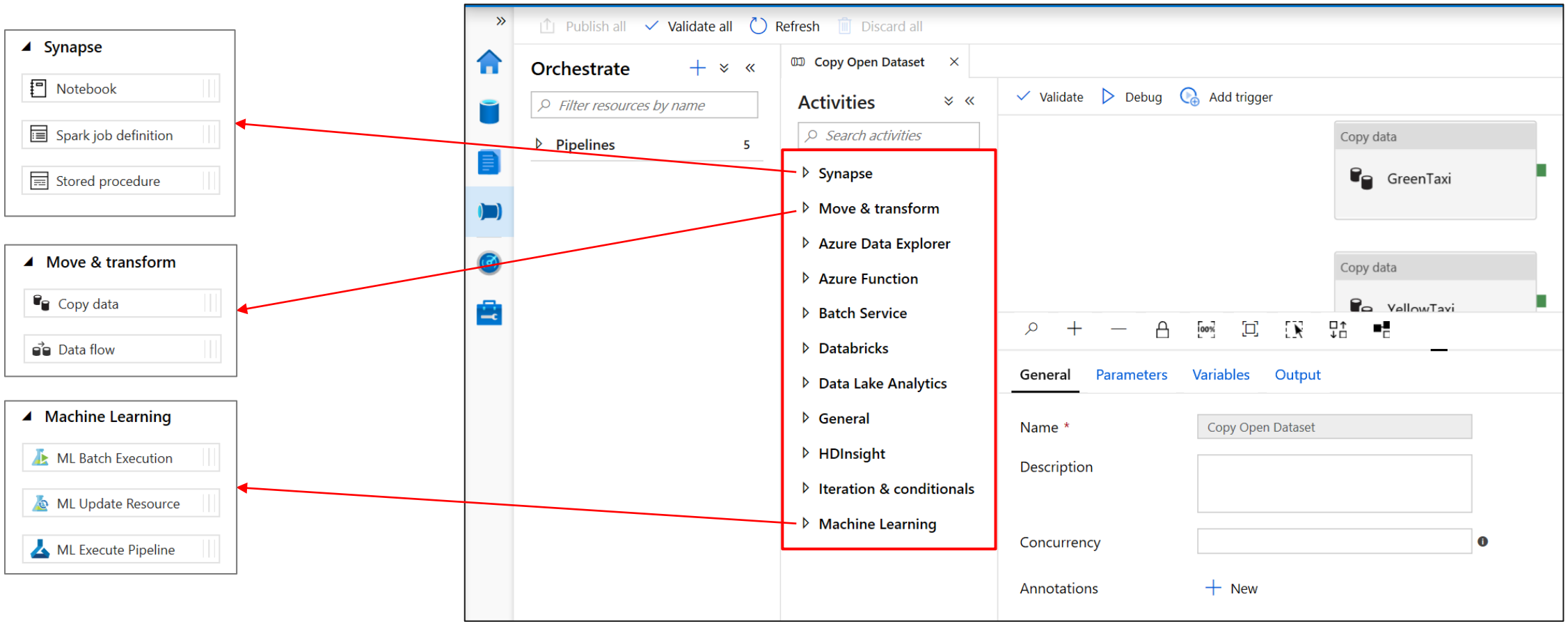




Integrate Hub

It provides ability to create pipelines to ingest, transform and load data with 90+ inbuilt connectors.

Offers a wide range of activities that a pipeline can perform.



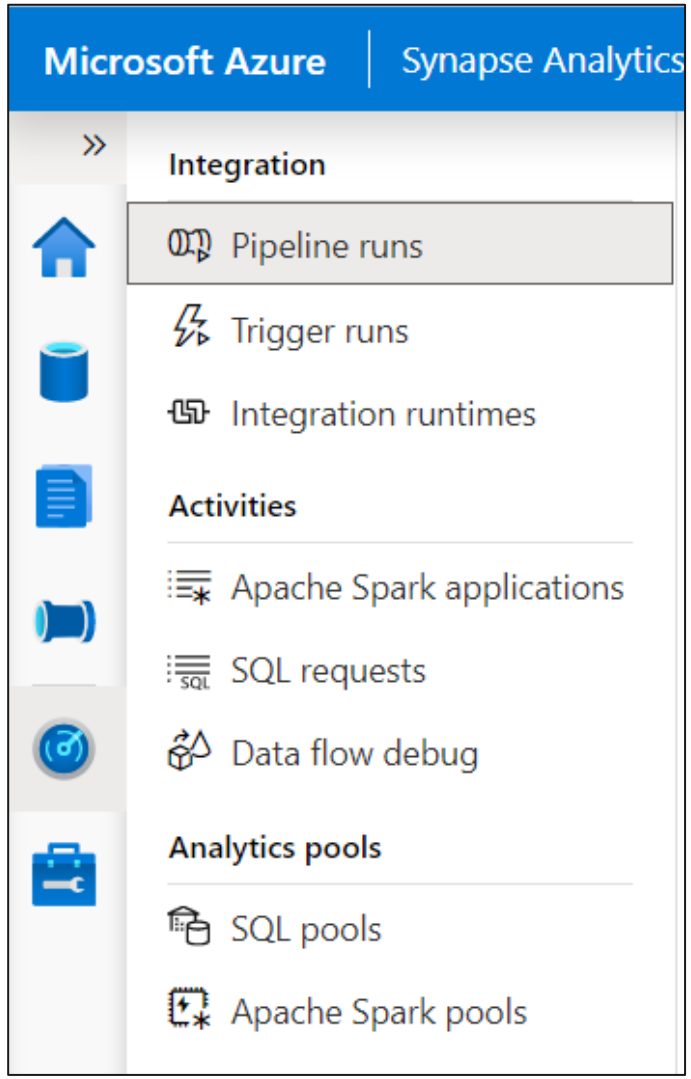
Monitor Hub

Overview

This feature provides single pane of glass to monitor orchestration, activities for Apache Spark Application and SQL requests.

Benefits

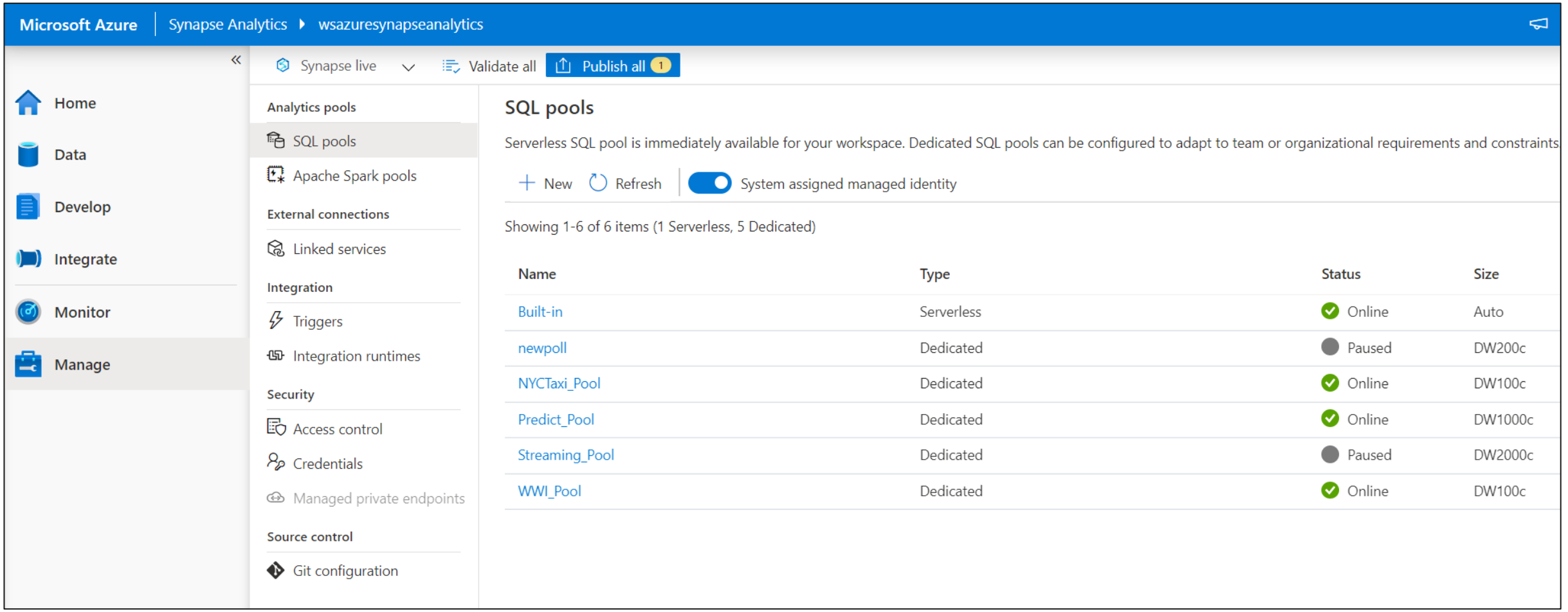
Offers additional filters to monitor specific activities or orchestration



Manage Hub

Overview

This feature provides ability to manage Analytics pools, Linked Services, Integration, Security and Source Control.



Manage – Linked services

Overview

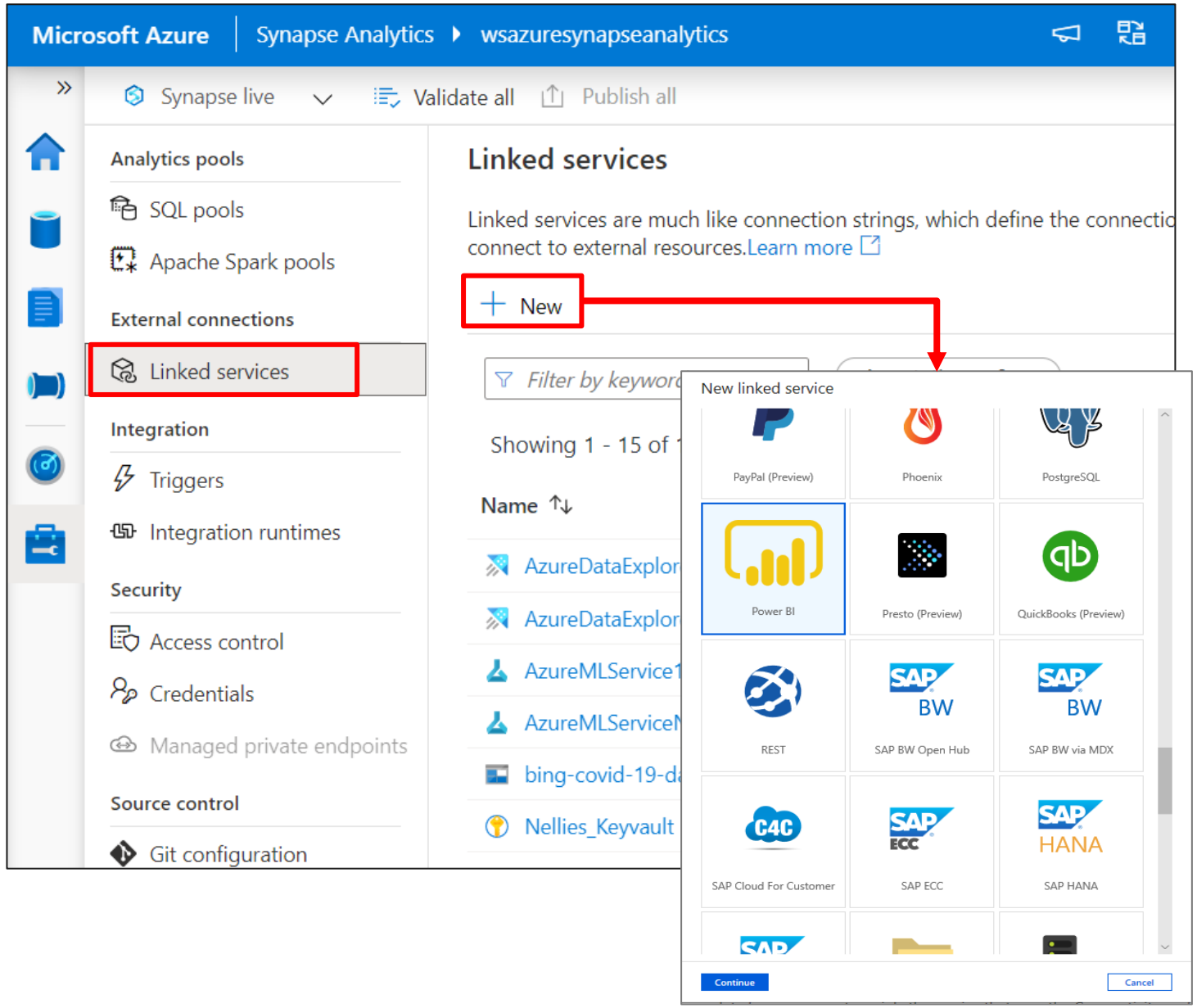
It defines the connection information needed to connect to external resources.

Benefits

Offers pre-build 90+ connectors

Easy cross platform data migration

Represents data store or compute resources



Manage – Triggers

Overview

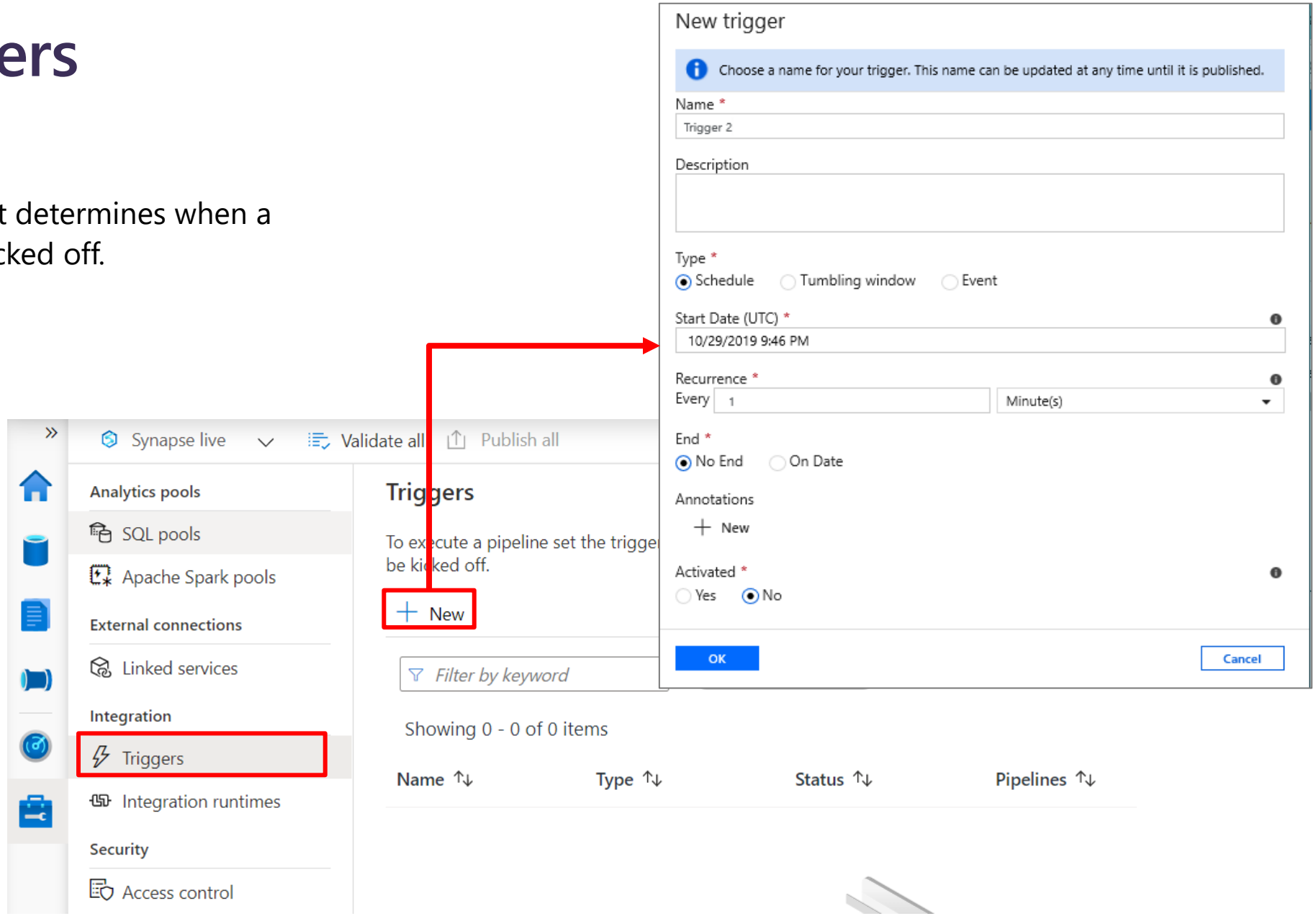
It defines a unit of processing that determines when a pipeline execution needs to be kicked off.

Benefits

Create and manage

- Schedule trigger
- Tumbling window trigger
- Event trigger

Control pipeline execution



Manage – Access Control

Overview

It provides access control management to workspace resources and artifacts for admins

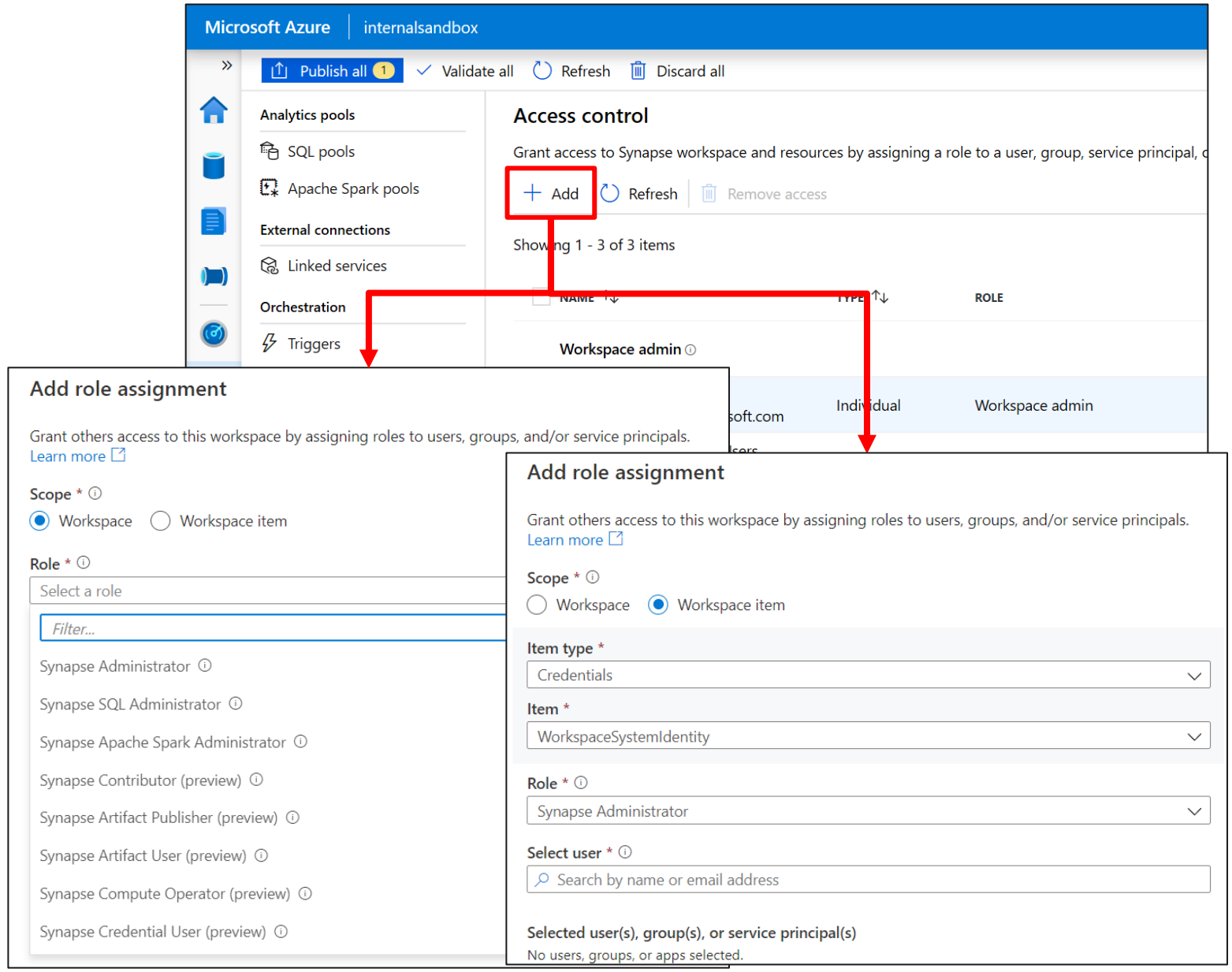
Benefits

Share workspace with the team

Increases productivity

Assign granular level permissions

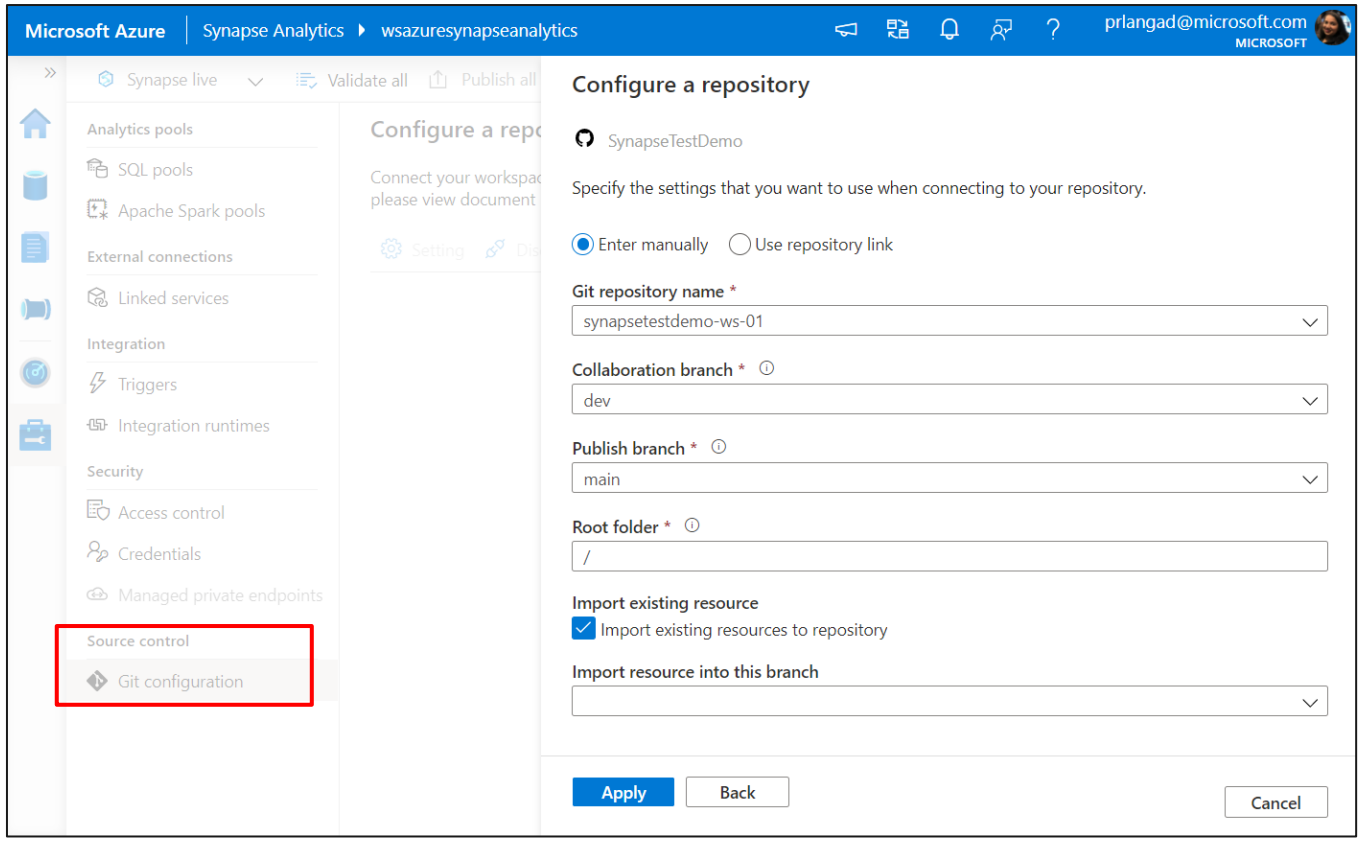
Manage permissions on Spark pools, Integration Runtimes, Linked services, Credentials

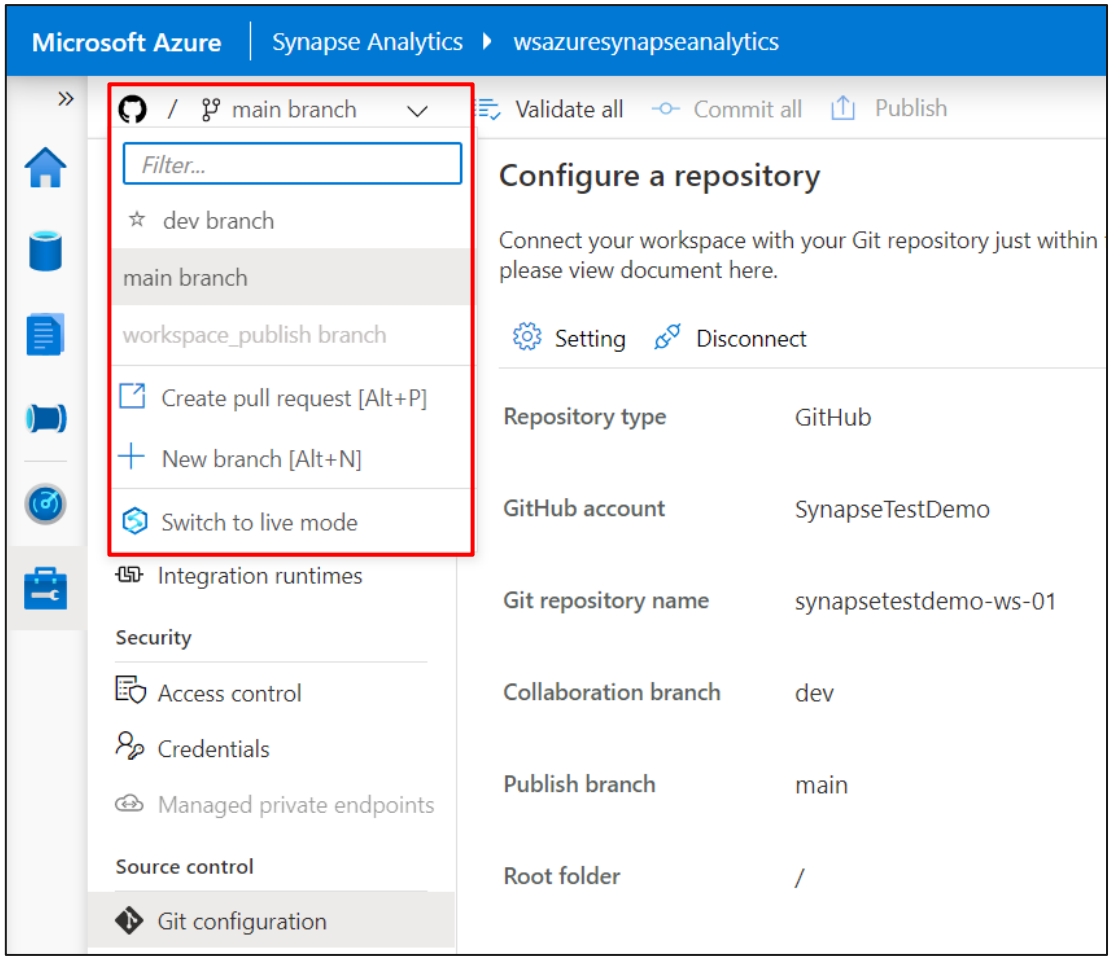


Manage – Source Control

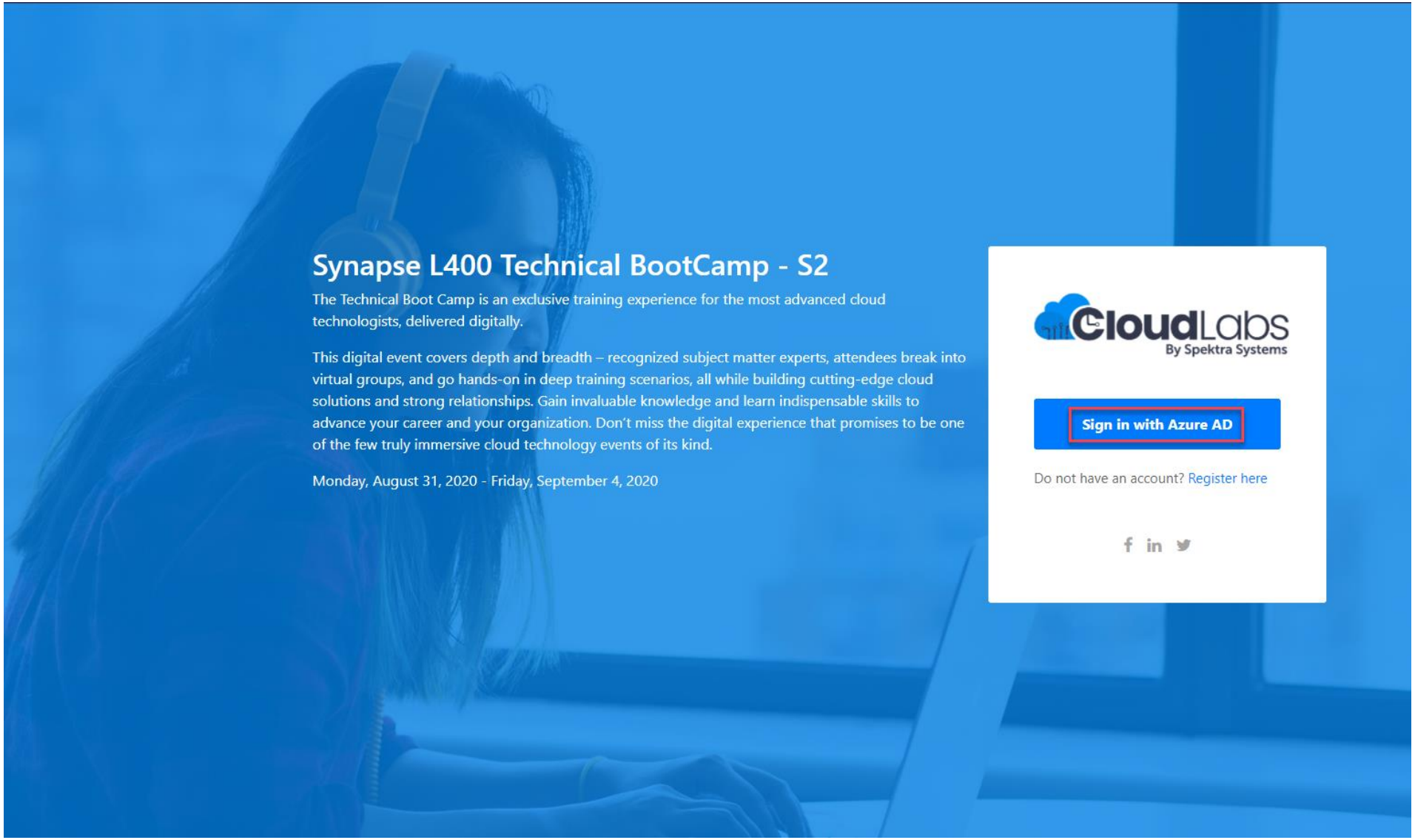
Overview

Associate Synapse workspace with a Git repository, Azure DevOps, or GitHub

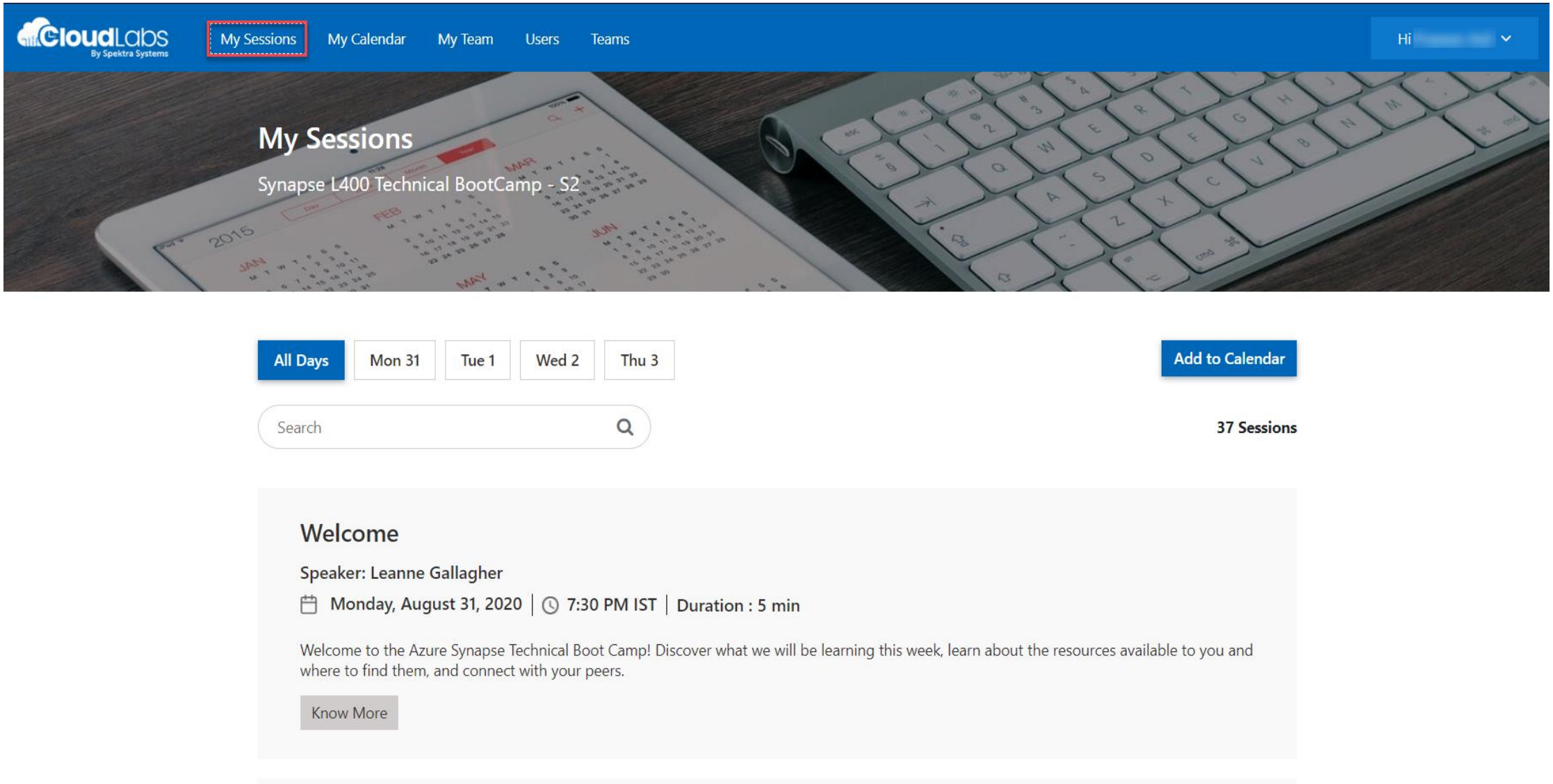




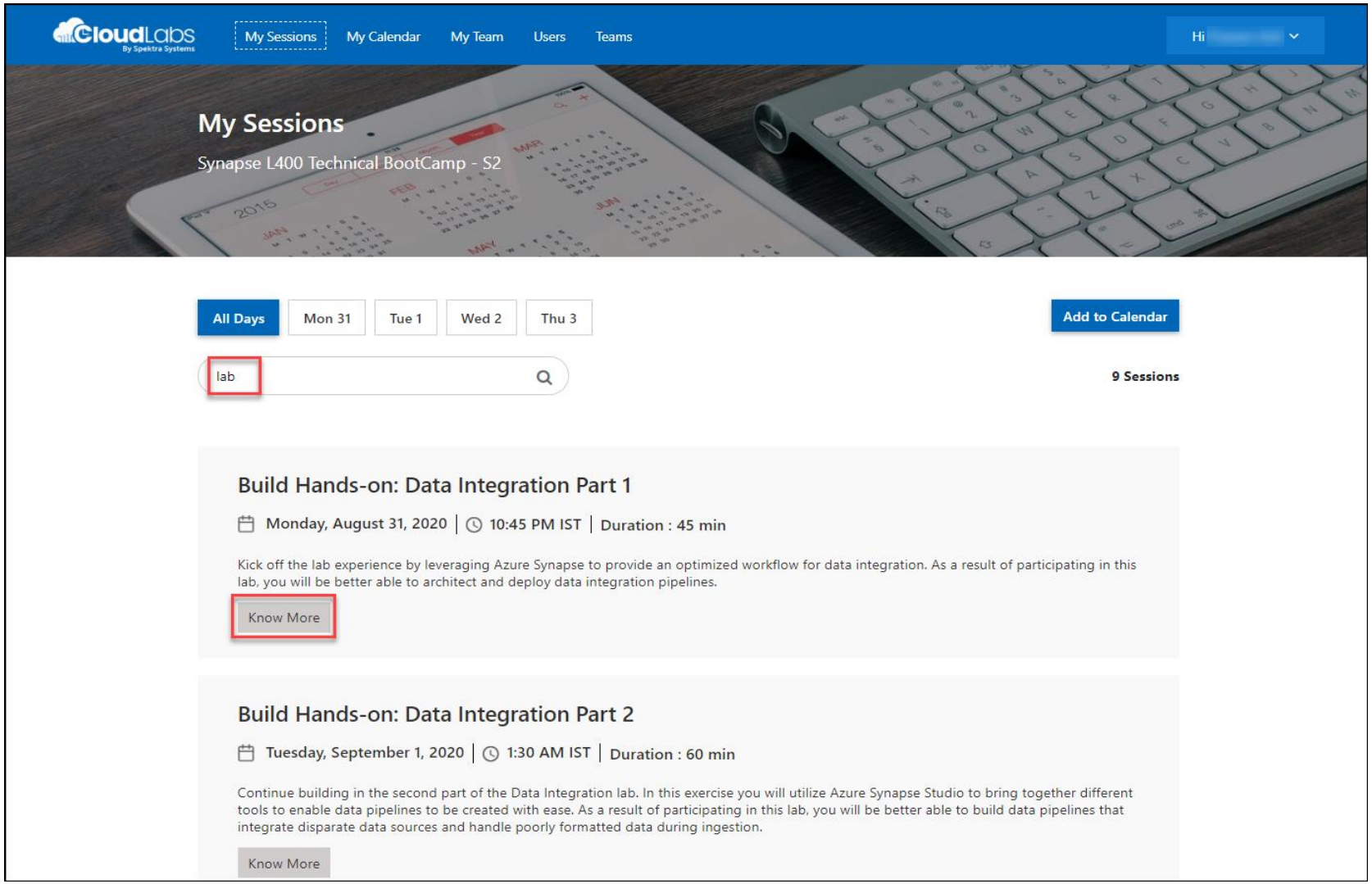
Login to the portal



Sessions for Bootcamp



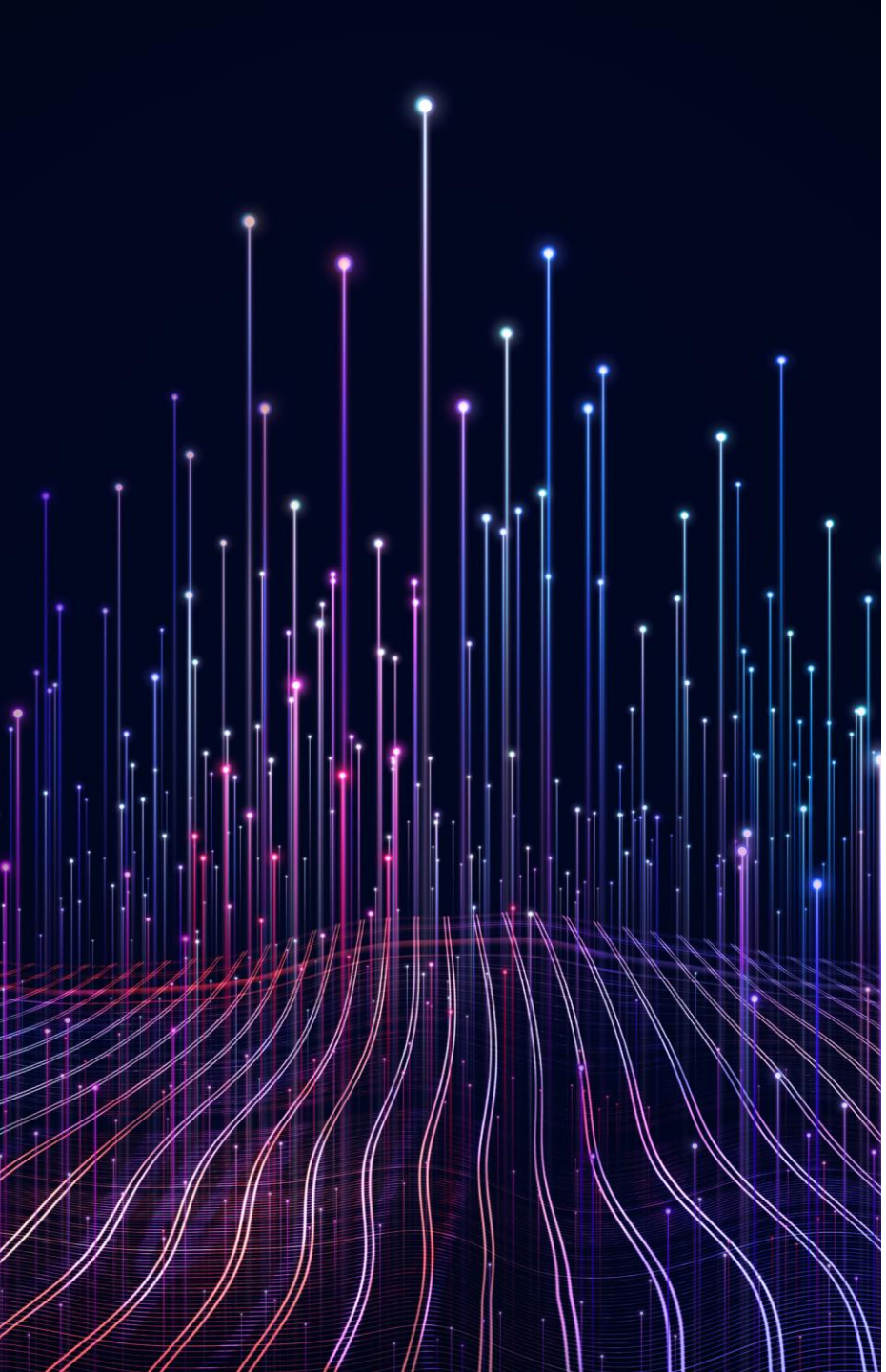
Search for lab



- Search for lab
- Click on Know More

Microsoft Azure Solutions Business

Data Loading & Data Lake Organization



Agenda

1 **Integration (Orchestration)**

Synapse pipelines

2 **Ingest files to tables**

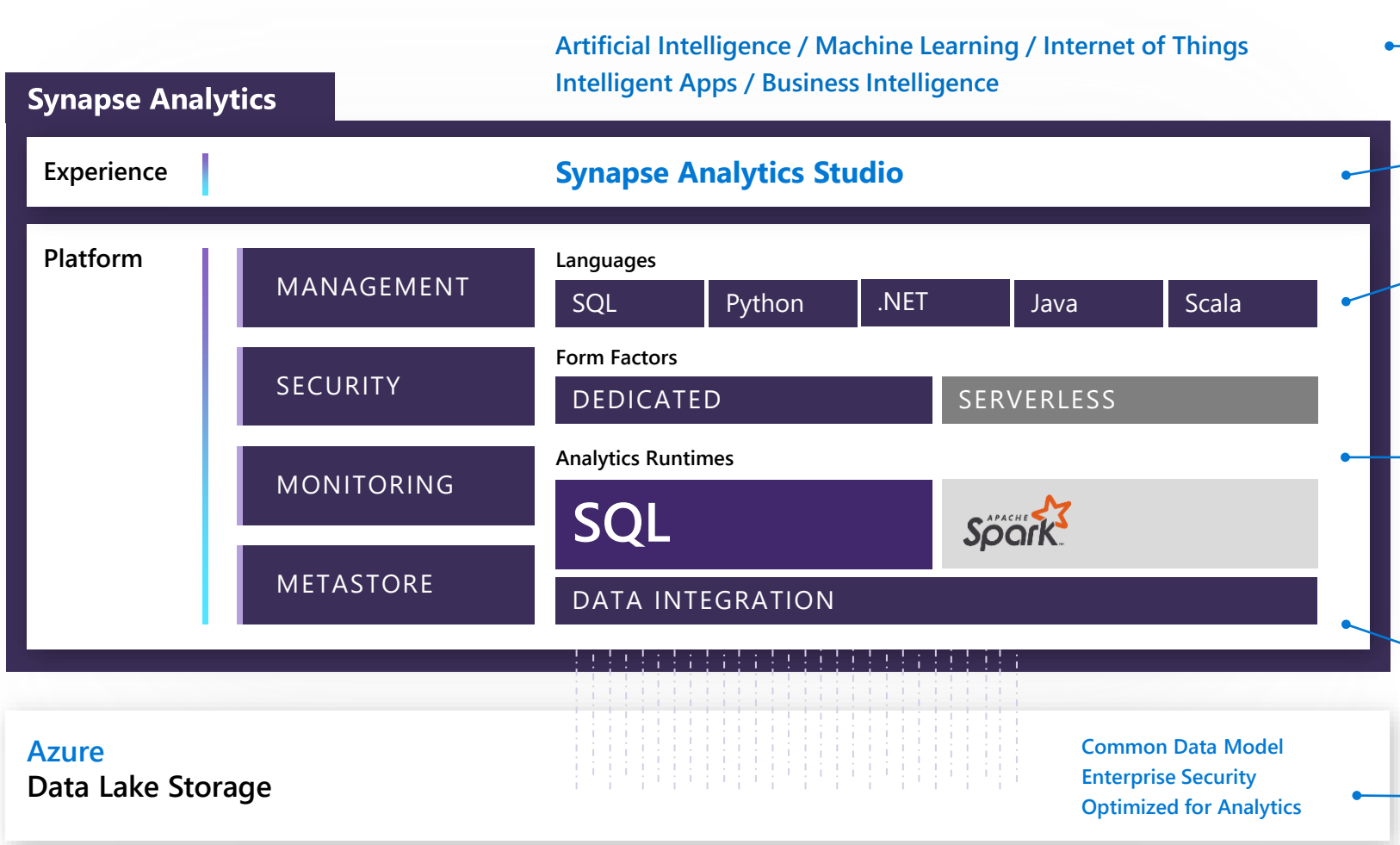
Copy versus CTAS

3 **Best practices**

Various ingest and storage best practices

Azure Synapse Analytics

Limitless analytics service with unmatched time to insight



Designed for analytics **workloads at any scale**

SaaS **developer experiences** for code free and code first

Multiple **languages** suited to different analytics workloads

Integrated analytics runtimes available dedicated and serverless

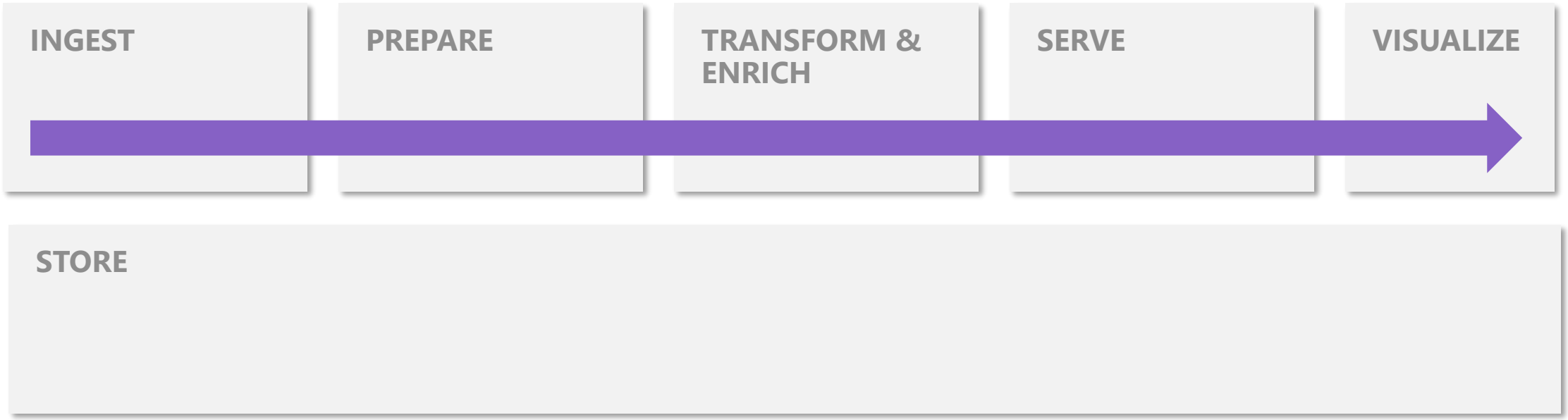
Synapse SQL offering T-SQL for batch, streaming and interactive processing

Apache Spark for big data processing with Python, Scala and .NET

Integrated **platform services** for, management, security, monitoring, and meta-store

Data **lake integrated** and Common Data Model aware

Modern Data Warehouse



Ingest - Integration with Pipelines

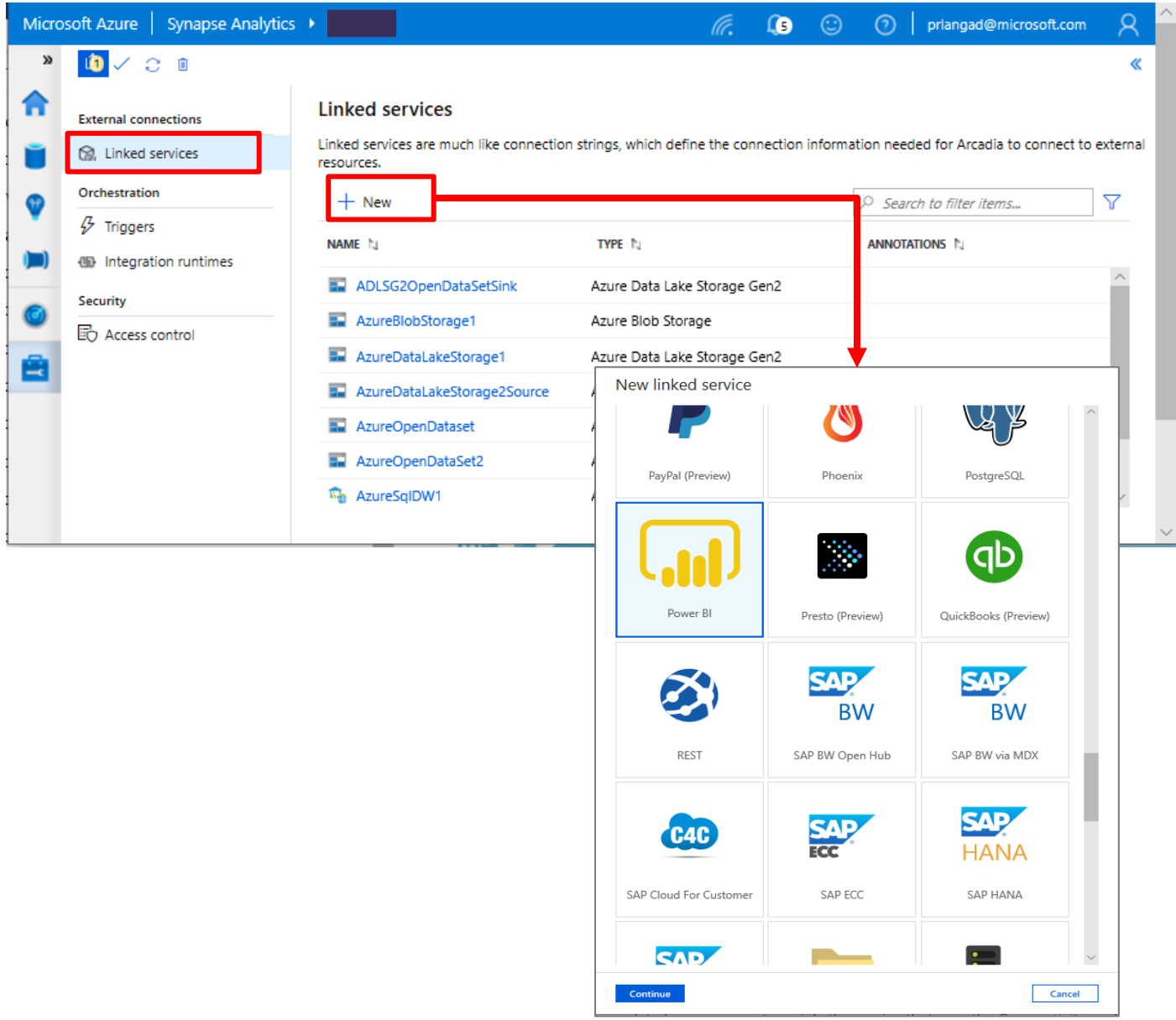
Linked services

Overview

Linked services define the connection information needed to connect to external resources.

Benefits

- Offers pre-build 90+ connectors
- Easy cross platform data migration
- Represents data store *or* compute resources



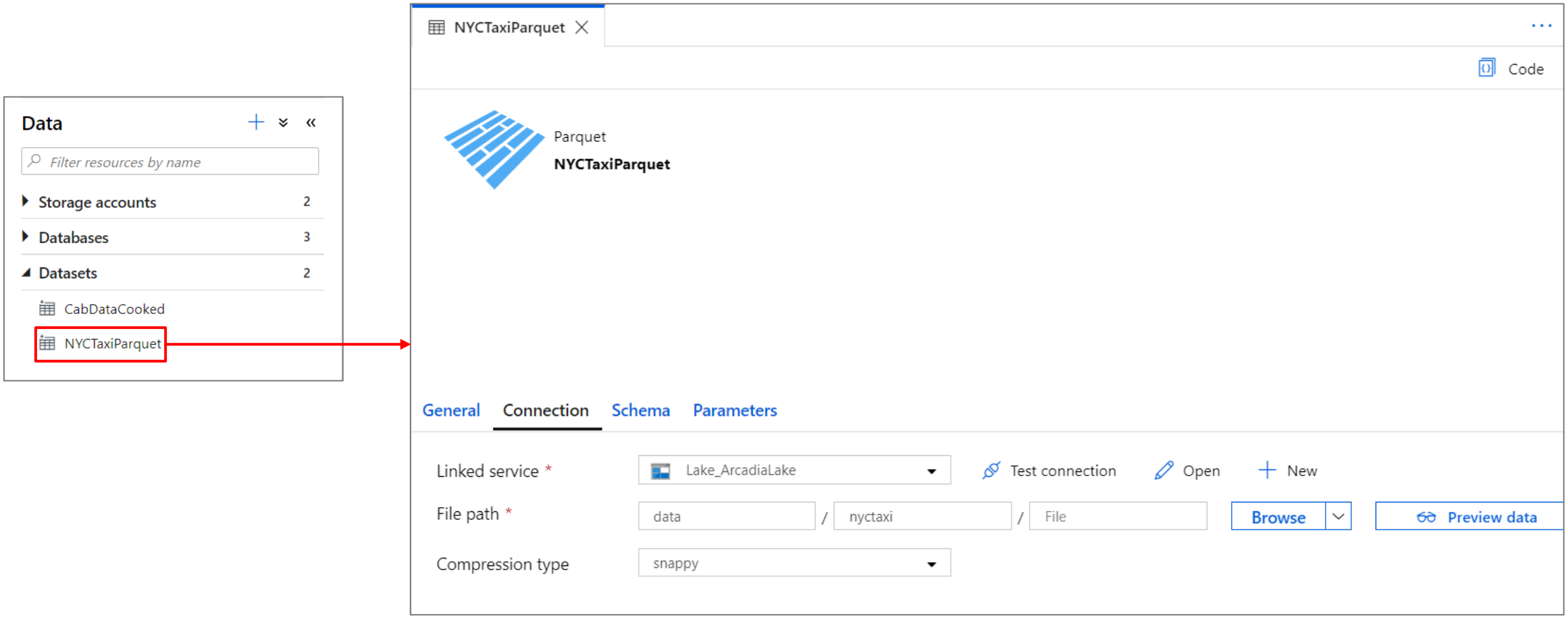
90+ Connectors out of the box

Azure (15)	Database & DW (26)		File Storage (6)	File Formats(6)	NoSQL (3)	Services and App (28)		Generic (4)
Blob storage	Amazon Redshift	Oracle	Amazon S3	AVRO	Cassandra	Amazon MWS	Oracle Service Cloud	Generic HTTP
Cosmos DB - SQL API	DB2	Phoenix	File system	Binary	Couchbase	CDS for Apps	PayPal	Generic OData
Cosmos DB - MongoDB API	Drill	PostgreSQL	FTP	Delimited Text	MongoDB	Concur	QuickBooks	Generic ODBC
Data Explorer	Google BigQuery	Presto	Google Cloud Storage	JSON		Dynamics 365	Salesforce	Generic REST
Data Lake Storage Gen1	Greenplum	SAP BW Open Hub	HDFS	ORC		Dynamics AX	SF Service Cloud	
Data Lake Storage Gen2	HBase	SAP BW via MDX	SFTP	Parquet		Dynamics CRM	SF Marketing Cloud	
Database for MariaDB	Hive	SAP HANA				Google AdWords	SAP C4C	
Database for MySQL	Apache Impala	SAP table				HubSpot	SAP ECC	
Database for PostgreSQL	Informix	Spark				Jira	ServiceNow	
File Storage	MariaDB	SQL Server				Magento	Shopify	
SQL Database	Microsoft Access	Sybase				Marketo	Square	
SQL Database MI	MySQL	Teradata				Office 365	Web table	
SQL Data Warehouse	Netezza	Vertica				Oracle Eloqua	Xero	
Search index	Snowflake					Oracle Responsys	Zoho	
Table storage								

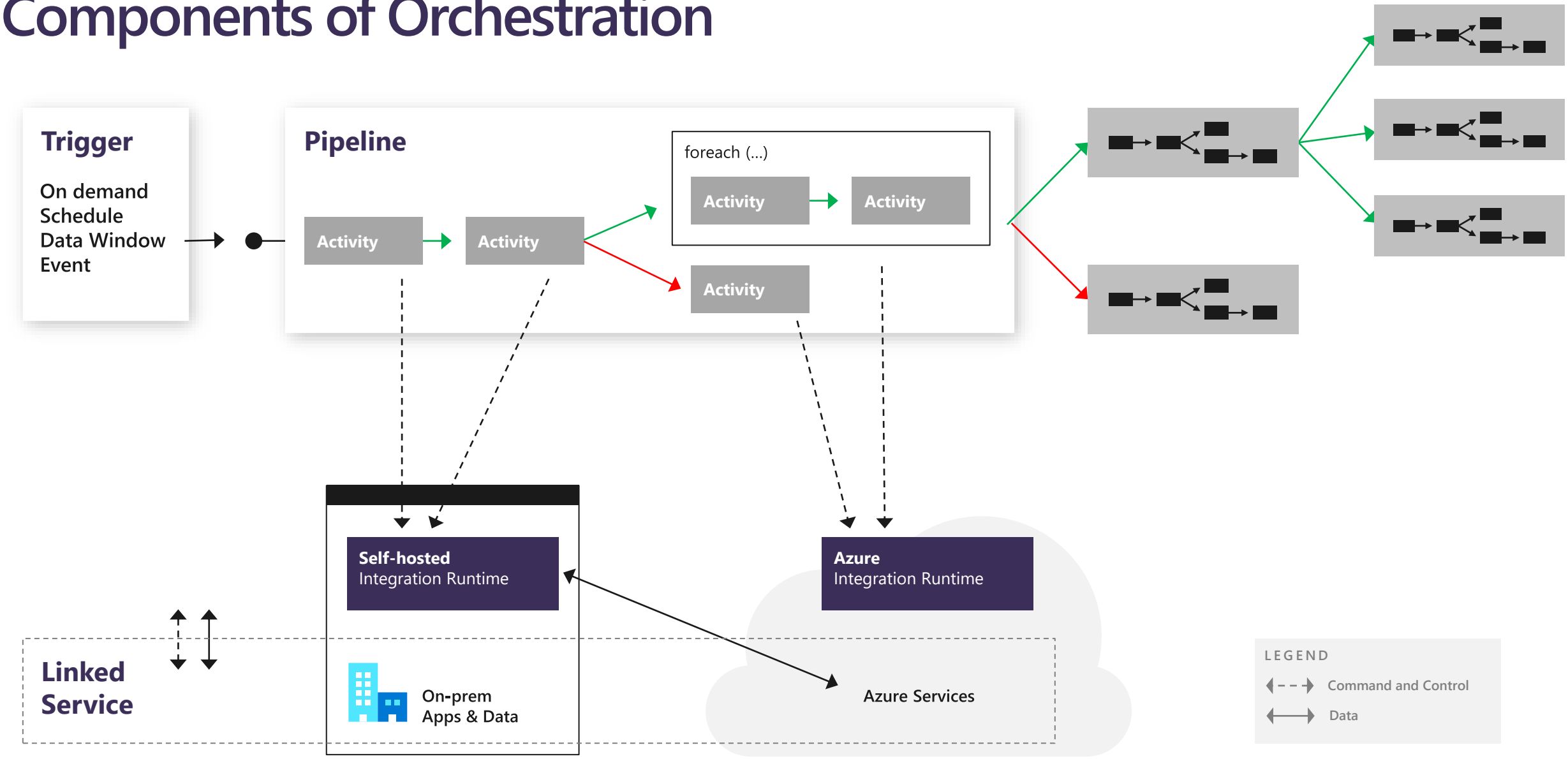
Datasets

Orchestration datasets describe data that is persisted.

Once a dataset is defined, it can be used in pipelines and sources of data or as sinks of data.



Components of Orchestration

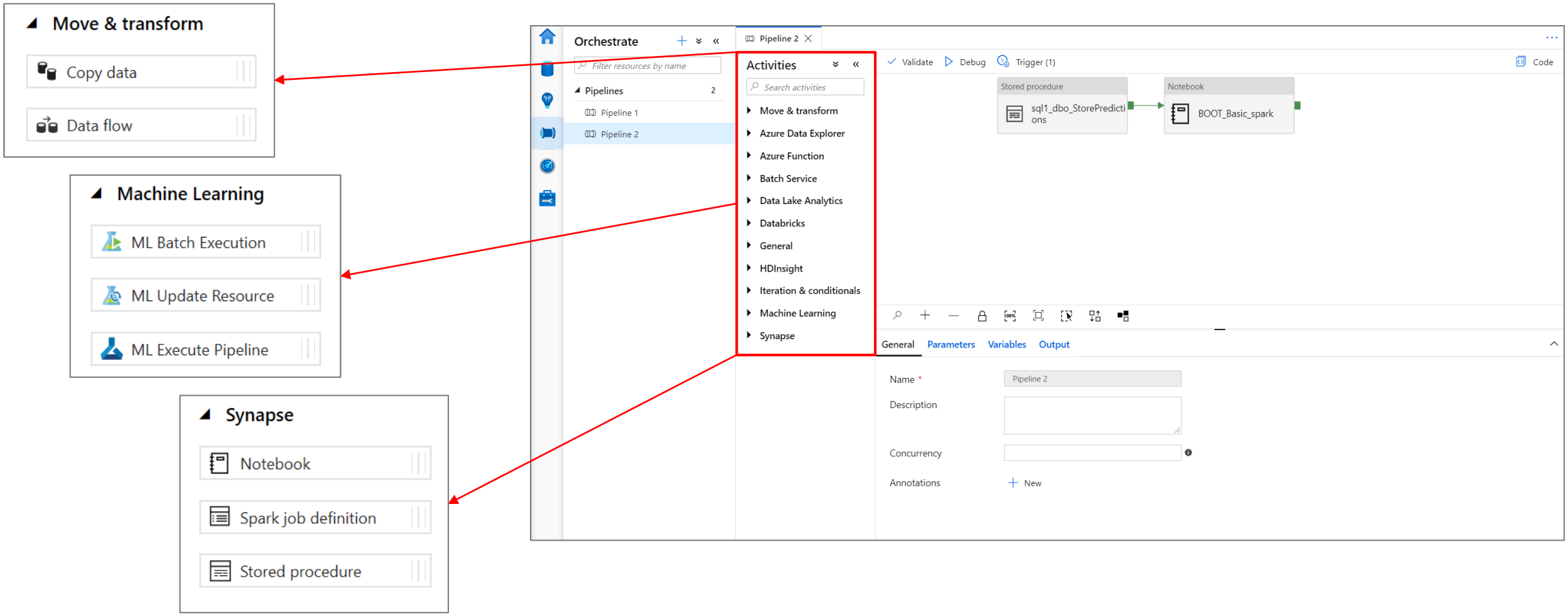


Synapse Pipelines shares codebase with Azure Data Factory

Pipelines

Create pipelines to ingest, transform and load data with 90+ inbuilt connectors.

Offers a wide range of activities that a pipeline can perform.



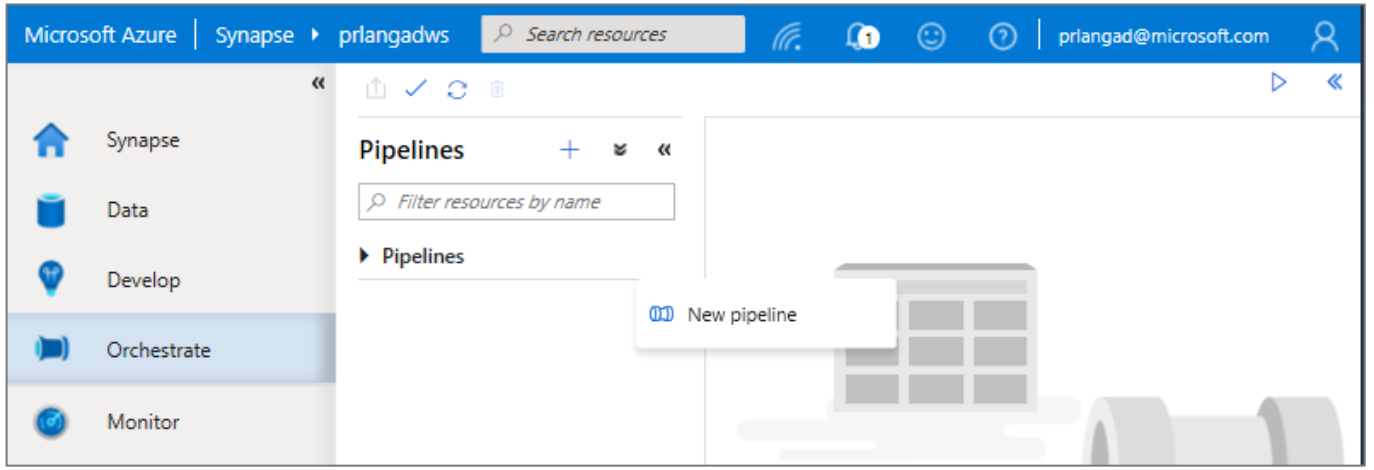
Pipelines

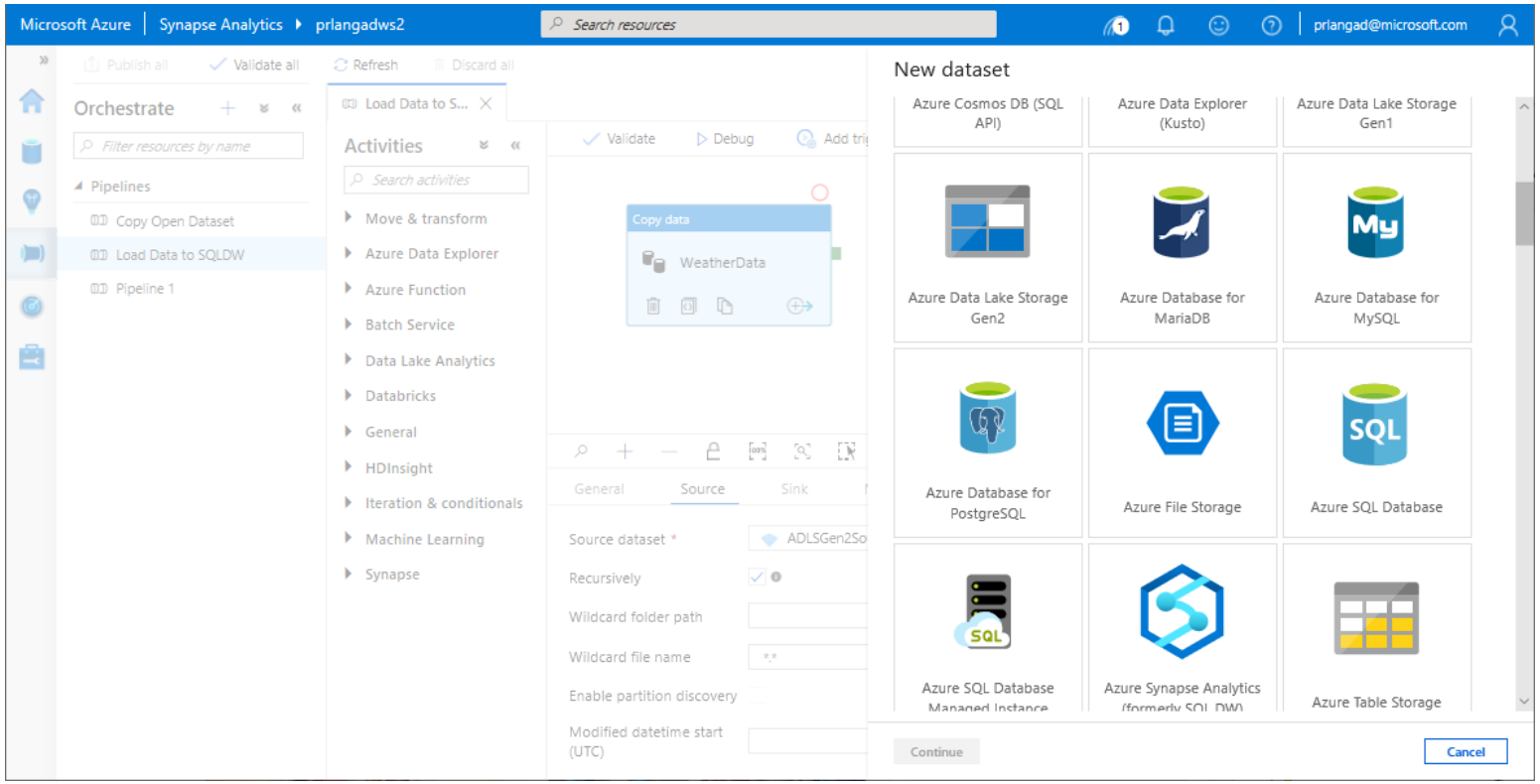
Overview

- Provide ability to load data from storage account to desired linked service.
- Load data by manual execution of pipeline or by orchestration.

Benefits

- Supports common loading patterns.
- Fully parallel loading into data lake or SQL tables.
- Graphical development experience.

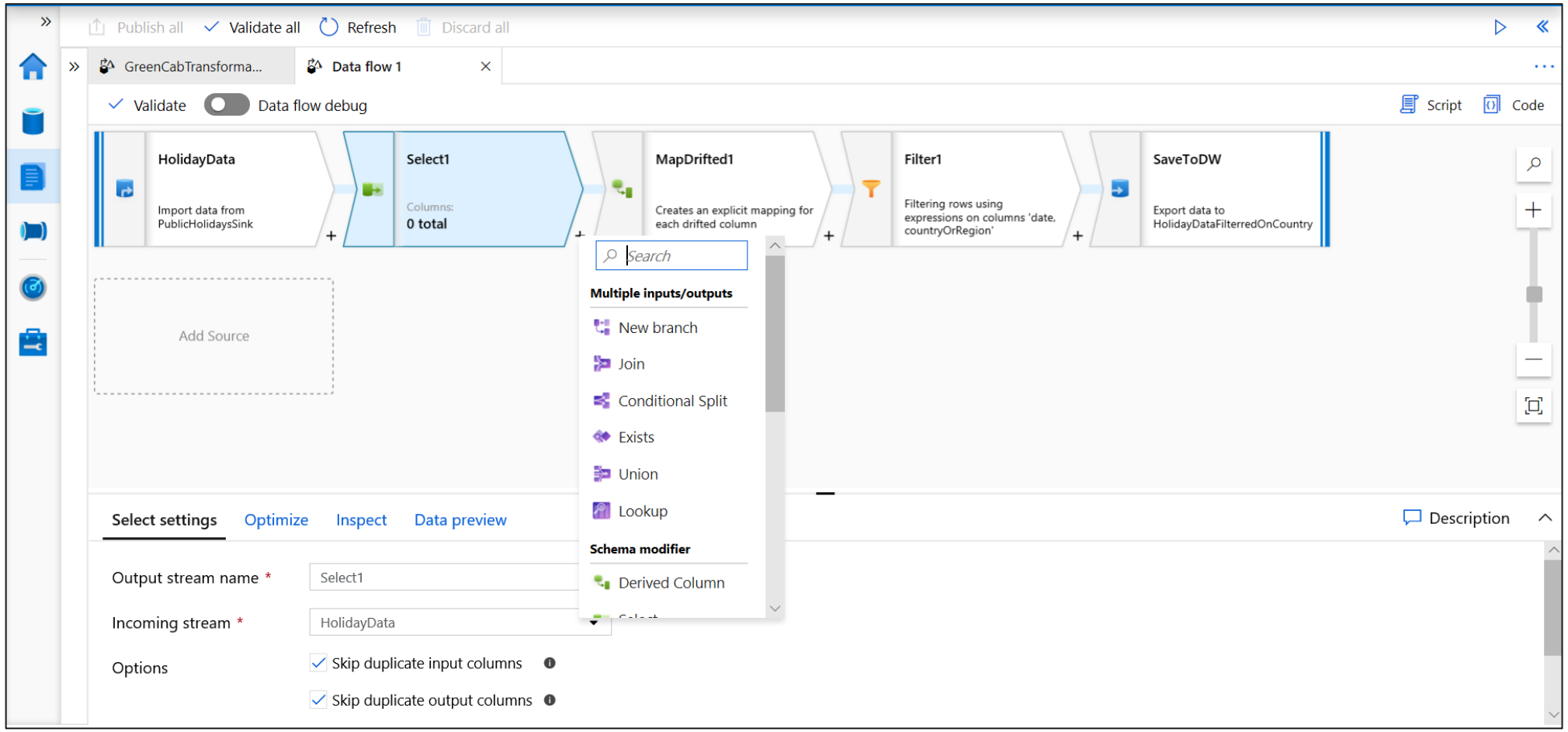




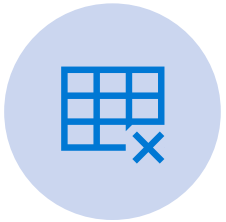
Develop Hub - Data Flows

Data flows are a visual way of specifying how to transform data.

Provides a code-free experience.



Dataflow Capabilities



Handle upserts, updates, deletes on sql sinks

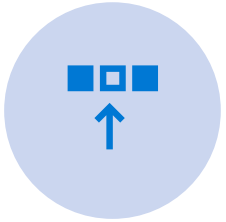
Add new partition methods

Add schema drift support

Add file handling (move files after read, write files to file names described in rows etc)

New inventory of functions (for e.g Hash functions for row comparison)

Commonly used ETL patterns(Sequence generator/Lookup transformation/SCD…)



Data lineage – Capturing sink column lineage & impact analysis(invaluable if this is for enterprise deployment)

Implement commonly used ETL patterns as templates(SCD Type1, Type2, Data Vault)

Triggers

Overview

Triggers represent a unit of processing that determines when a pipeline execution needs to be kicked off.

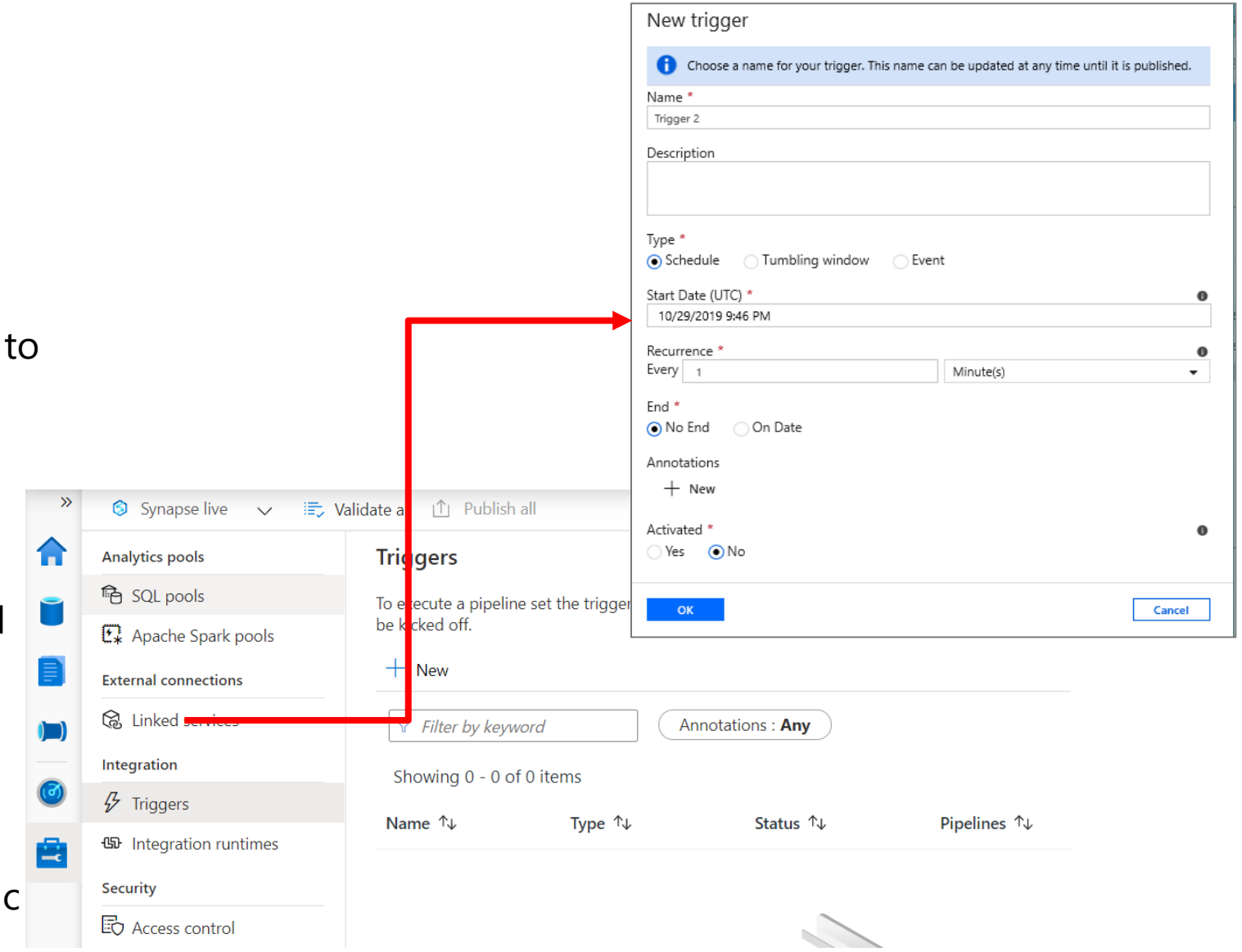
Data Integration offers 3 trigger types as –

1. Schedule – gets fired at a schedule with information of start date, recurrence, end date

2. Event – gets fired on specified Storage event

3. Tumbling window – gets fired at a periodic time interval from a specified start date, while retaining state

It also provides ability to monitor pipeline runs and control trigger execution.



Manage – Integration runtimes

Overview

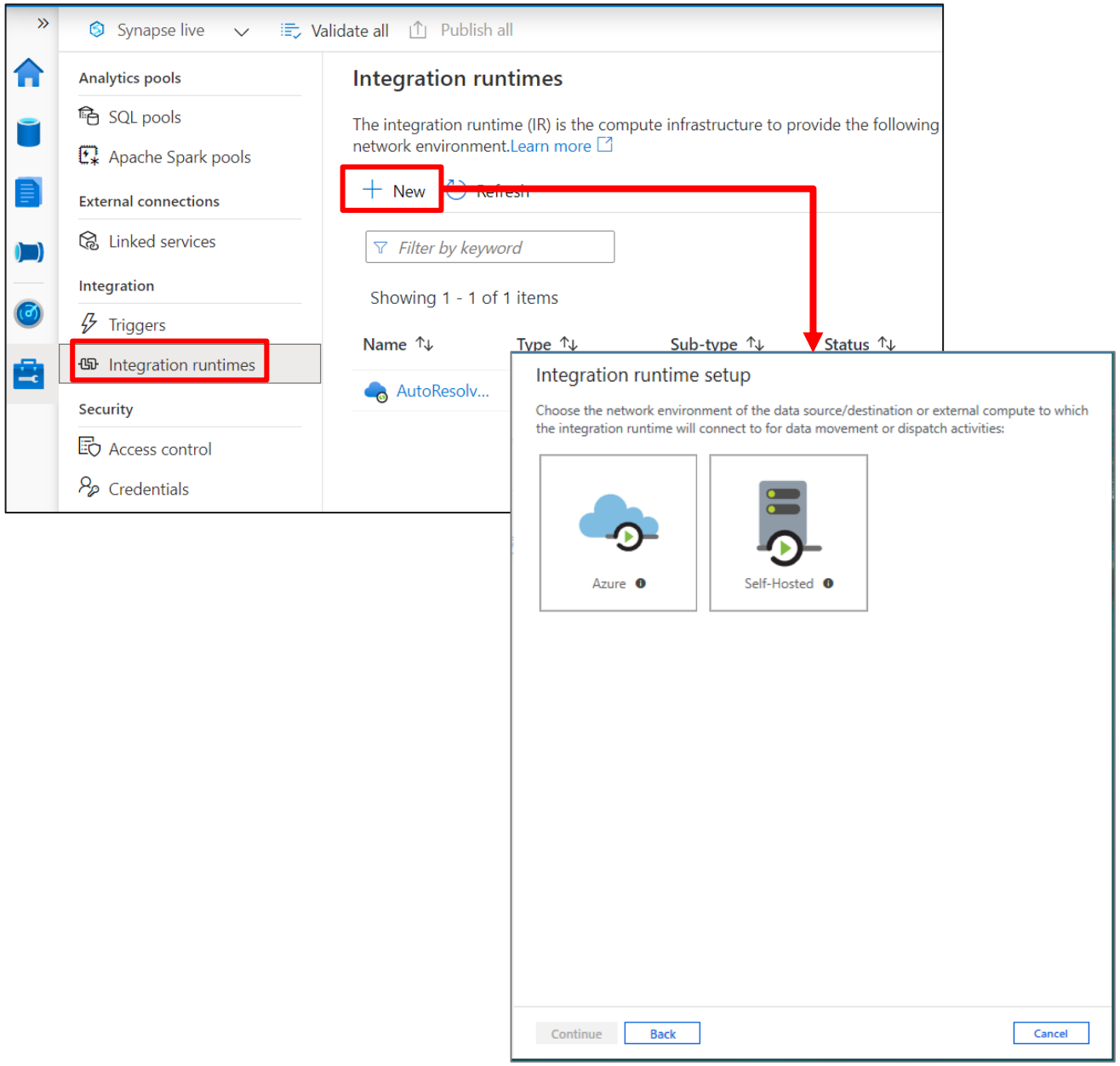
Integration runtimes are the compute infrastructure used by Pipelines to provide the data integration capabilities across different network environments. An integration runtime provides the bridge between the activity and linked services.

Benefits

Offers Azure Integration Runtime or Self-Hosted Integration Runtime

Azure Integration Runtime – provides fully managed, serverless compute in Azure

Self-Hosted Integration Runtime – use compute resources in on-premises machine or a VM inside private network



Data Movement with Integration Runtimes

Scalable

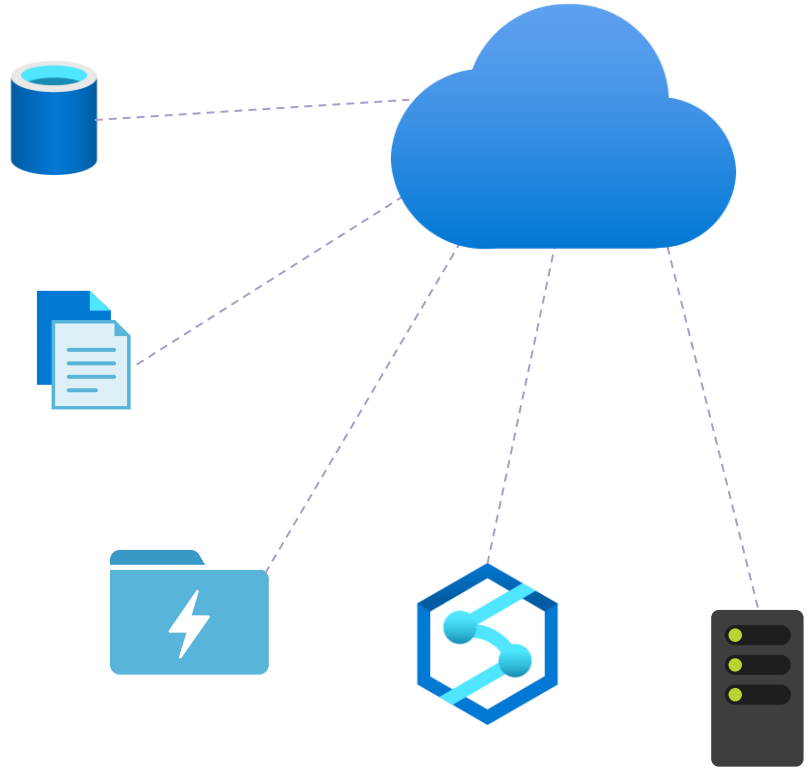
- per job elasticity
- Up to 4 GB/s

Simple

- Visually author or via code (Python, .Net, etc.)
- Serverless, no infrastructure to manage

Access all your data

- 90+ connectors provided and growing (cloud, on premises, SaaS)
- Data Movement as a Service: 25 points of presence worldwide
- Self-hostable Integration Runtime for hybrid movement



Pop Quiz 1

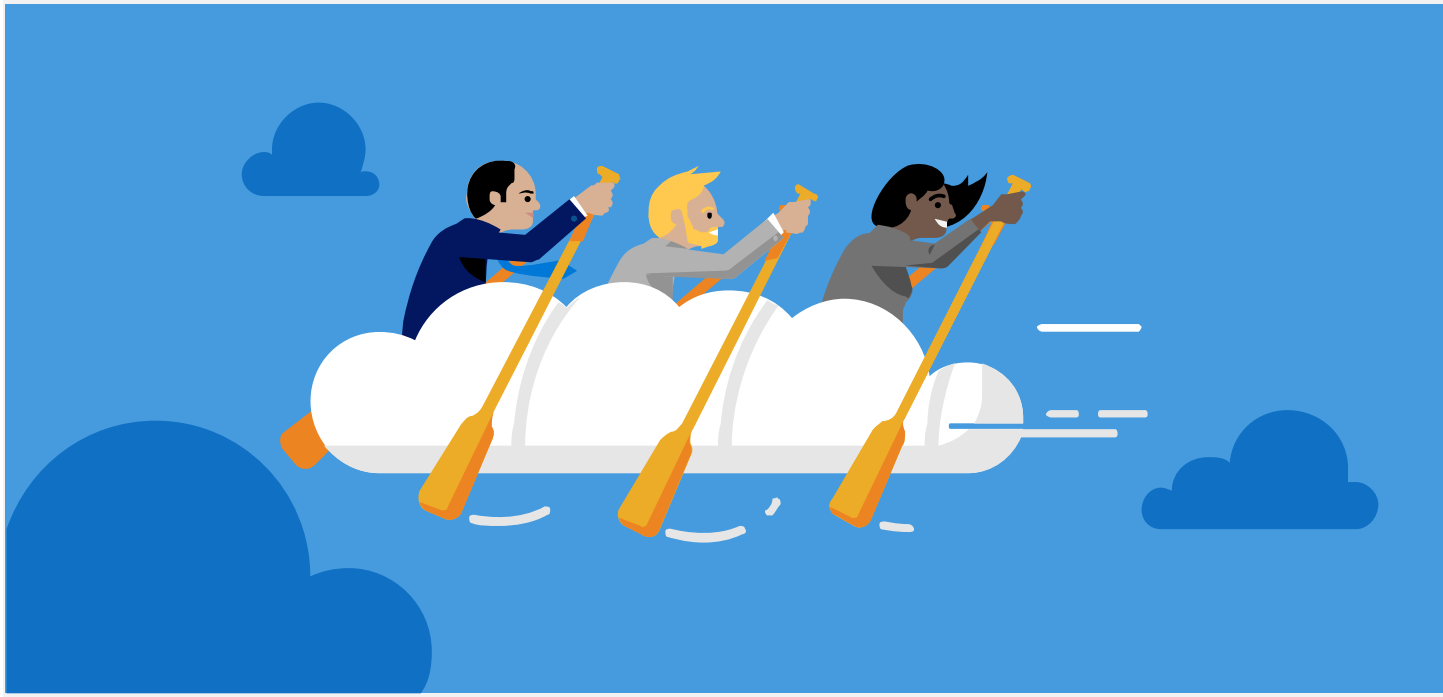
Which one of these is NOT a component of a Synapse pipeline?

A) I.R.

B) Linked Service

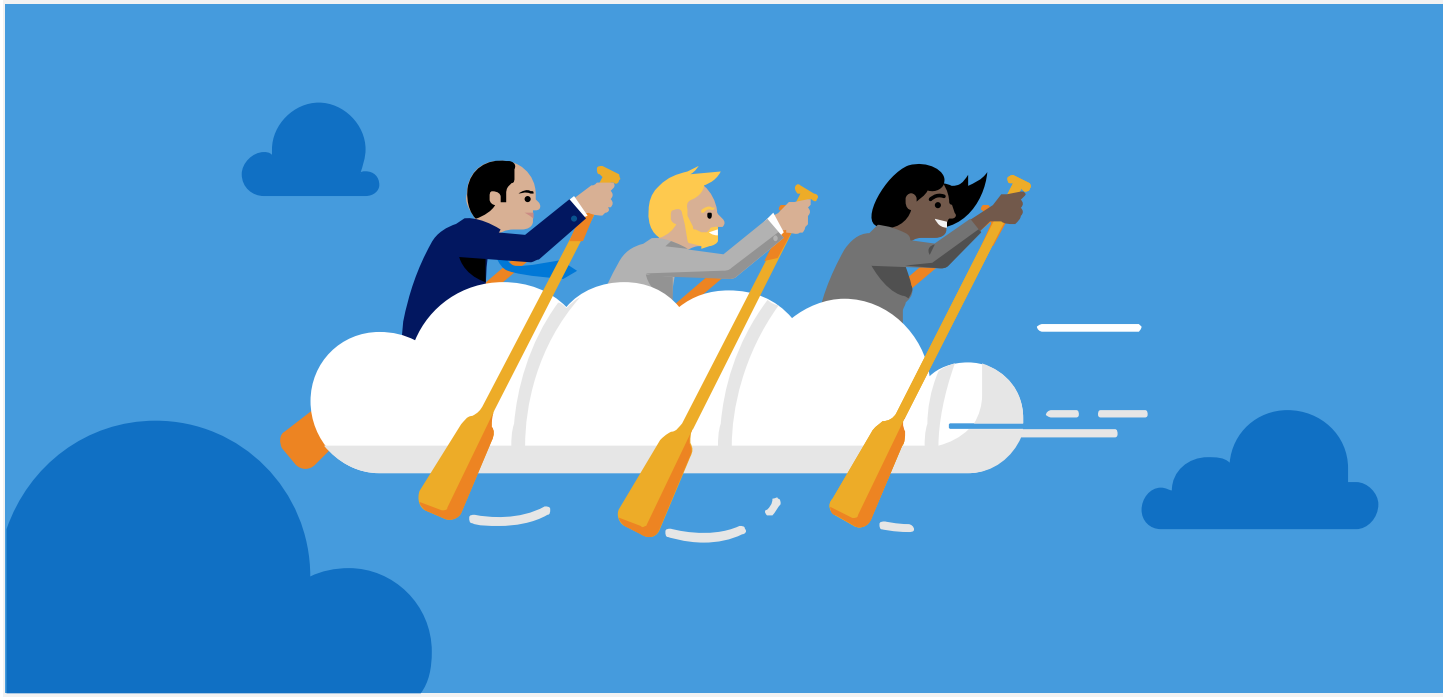
C) Table

D) Activity



Pop Quiz 1

Which one of these is NOT a component of a Synapse pipeline?



A) I.R.

B) Linked Service

C) Table

D) Activity

Files and Tables

COPY command

Overview

Copies data from source to destination

Benefits

- Retrieves data from all files from the folder and all its subfolders.
- Supports multiple locations from the same storage account, separated by comma
- Supports Azure Data Lake Storage (ADLS) Gen 2 and Azure Blob Storage.
- Supports CSV, PARQUET, ORC file formats

```
COPY INTO test_1
FROM 'https://XYZ.blob.core.windows.net/customerdatasets/test_1.txt'
WITH (
   FILE_TYPE = 'CSV',
   CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>'),
   FIELDQUOTE = '"',
   FIELDTERMINATOR=';',
   ROWTERMINATOR='0X0A',
   ENCODING = 'UTF8',
   DATEFORMAT = 'ymd',
   MAXERRORS = 10,
   ERRORFILE = '/errorsfolder/'--path starting from the storage container,
   IDENTITY_INSERT
   )
```

```
COPY INTO test_parquet
FROM 'https://XYZ.blob.core.windows.net/customerdatasets/test.parquet'
WITH (
   FILE_FORMAT = myFileFormat
   CREDENTIAL=(IDENTITY= 'Shared Access Signature',
SECRET='<Your_SAS_Token>')
)
```

Create External Table As Select (Polybase)

Overview

- Creates an external table and then exports results of the SELECT statement. These operations will import data into the database for the duration of the query

Steps:

1. Create Master Key
2. Create Credentials
3. Create External Data Source
4. Create External Data Format
5. Create External Table

```
-- Create a database master key if one does not already exist
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'S0me!nfo'
;
-- Create a database scoped credential with Azure storage account key as the secret.
CREATE DATABASE SCOPED CREDENTIAL AzureStorageCredential
WITH
    IDENTITY   = '<my_account>'
,   SECRET     = '<azure_storage_account_key>'
;
-- Create an external data source with CREDENTIAL option.
CREATE EXTERNAL DATA SOURCE MyAzureStorage
WITH
(   LOCATION   = 'wasbs://daily@logs.blob.core.windows.net/'
,   CREDENTIAL = AzureStorageCredential
,   TYPE       = HADOOP
)
-- Create an external file format
CREATE EXTERNAL FILE FORMAT MyAzureCSVFormat
WITH (FORMAT_TYPE = DELIMITEDTEXT,
     FORMAT_OPTIONS(
        FIELD_TERMINATOR = ',',
        FIRST_ROW = 2)
--Create an external table
CREATE EXTERNAL TABLE dbo.FactInternetSalesNew
WITH(
       LOCATION = '/files/Customer',
       DATA_SOURCE = MyAzureStorage,
       FILE_FORMAT = MyAzureCSVFormat
   )
AS SELECT T1.* FROM dbo.FactInternetSales T1 JOIN dbo.DimCustomer T2
ON ( T1.CustomerKey = T2.CustomerKey )
OPTION ( HASH JOIN );
```

Polybase vs Copy

Polybase

- GA, stable
- Needs CONTROL permission
- Enables querying via external tables
- Challenges:
  - Row width (1 MB)
  - Delimiters in text
  - Fixed line delimiter
  - Code complexity

Copy

- Relaxed permission
- No row width limit
- Supports delimiters in text
- Supports custom column and row delimiters

Best Practices for Files and Tables

Question...

How many different methods of loading ADLS can you think of?

What about a Synapse SQL Pool?

Ingest Flat files to tables

Ingest flat file data into Azure Storage (Azure Data Lake Store Gen2)

- When your data sources are on-premises, you need to move the data to Azure Storage before ingestion.
- Data in other cloud platforms needs to be moved to Azure Storage before ingestion.

Load from flat files as relational tables within the data warehouse

Ingest - Structuring ADLS Gen2

- Separate storage accounts for each environment: dev, test, & production.

- Use a common folder structure to organize data by degree of refinement.

Ingest from on-premises data sources

Fastest is done by batch:

- Extract from data source to multiple CSV/Parquet files
- Use AzCopy to upload to ADLS

Alternative is query-insert:

- Set up SSIS self-hosted integration runtime on-premises
- Use Synapse Pipeline to extract/copy
- Use Synapse Pipeline to execute load procedure

Large Migrations:

- Use Azure Data Box where available

Ingest from Cloud Data Sources

Options:

- Extract using Synapse Pipelines
- Write to ADLS as Parquet files
- AzCopy is a fast move for files from S3 to ADLS

Ingest File Data Sources

Look out for these file format challenges…

Invalid file format

- Multiple row types
- Ragged columns

Row size > 1Mb

Datetime format/s (e.g., use of nanosecond date time)

NULL value literal/s

Free form text

Parquet partitions

XML data

Use of non-standard line delimiters (e.g., CR)

…and try these Solutions

- Use Spark to pre-process and fix data errors
- Flatten and parse XML in Spark
- Use COPY to ingest complex CSV instead of Polybase

Ingest and Store – Formats

For batch flat files, Azure Synapse Analytics supports CSV, Parquet, ORC, and JSON formats.

Ingest streaming data messages/events via Event Hub or IoT Hub.

Parquet format recommended for storing ingested data at various levels of refinement.

Ingest - When to BCP / Bulk Copy

Green fields: Never

- Network unreliability, no retries
- Needs VM in cloud, performance dependent on VM configuration
- Doesn’t support ADLS
- Reduces concurrency
- Control-gated performance limitation, can not scale with DWU

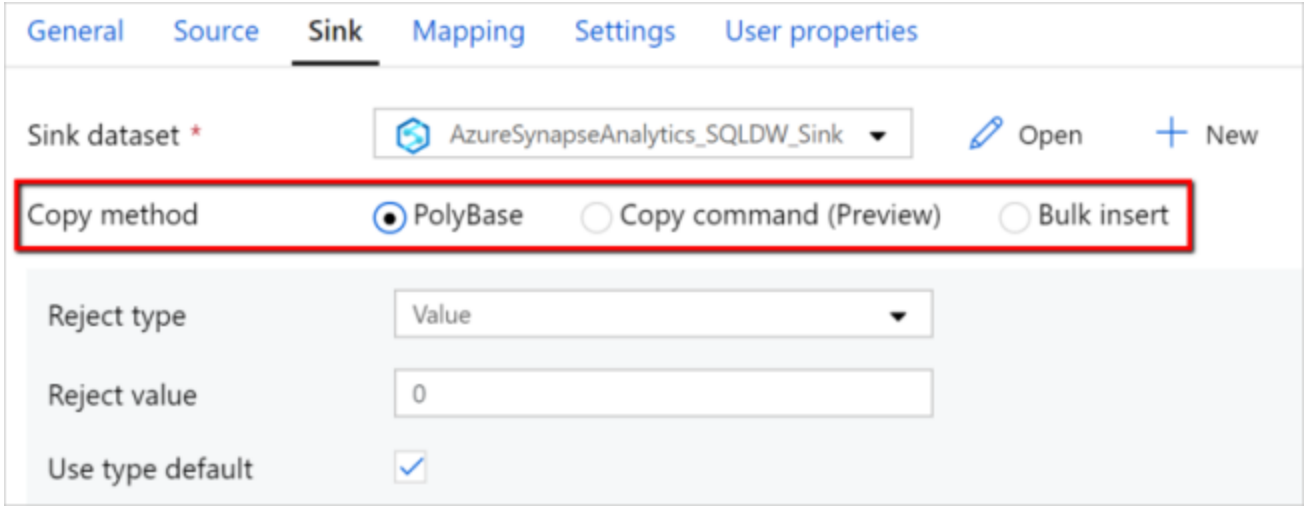
Migrations:

- Use Synapse Pipeline or AzCopy
- Bulk Copy will work, but it will be slower than other methods

Ingest – Synapse Pipelines

- Un-check USE TYPE DEFAULT, it is not a best practice.

- Land data in ADLS Gen2, then ingest using Polybase / COPY.
  - This means you can re-ingest the same data set without having to repeat extracts, and better demonstrate ingestion performance.



Ingest and Store – Loading staging tables

Indexing

Use Heap tables

Speed load performance by staging data in heap tables and temporary tables prior to running transformations.

Only load to a CCI table if the test requires a load to a single table, then complex end-user queries against that table.

Ingest and Store – Loading staging tables

Distribution

Use Round Robin Distribution for:

- Potentially useful tables created from raw input.
- Temporary staging tables used in data preparation.

Other distribution considerations:

- Never load to a REPLICATED table
- Load to a ROUND_ROBIN table if the test is ONLY raw ingestion performance, or if the table is very small
- Load to a HASH table if the task is a pipeline with subsequent transformations using the loaded table

Ingest – Scaling to shorten duration

Ingestion duration is correlated with the number of DWU's allocated to the SQL Pool.

For every *doubling* of the DWU's you *halve* the ingestion time.

2d = t/2

d: DWU

T: ingestion time

Only applies from DWU500c – DWU30000c

Export to files with CETAS

CETAS = parallel operation that creates external table metadata and exports the SELECT query results to a set of files in your storage account.

Store frequently used parts of queries, like joined reference tables, to a new set of files. You can then join to this single external table instead of repeating common joins in multiple queries.

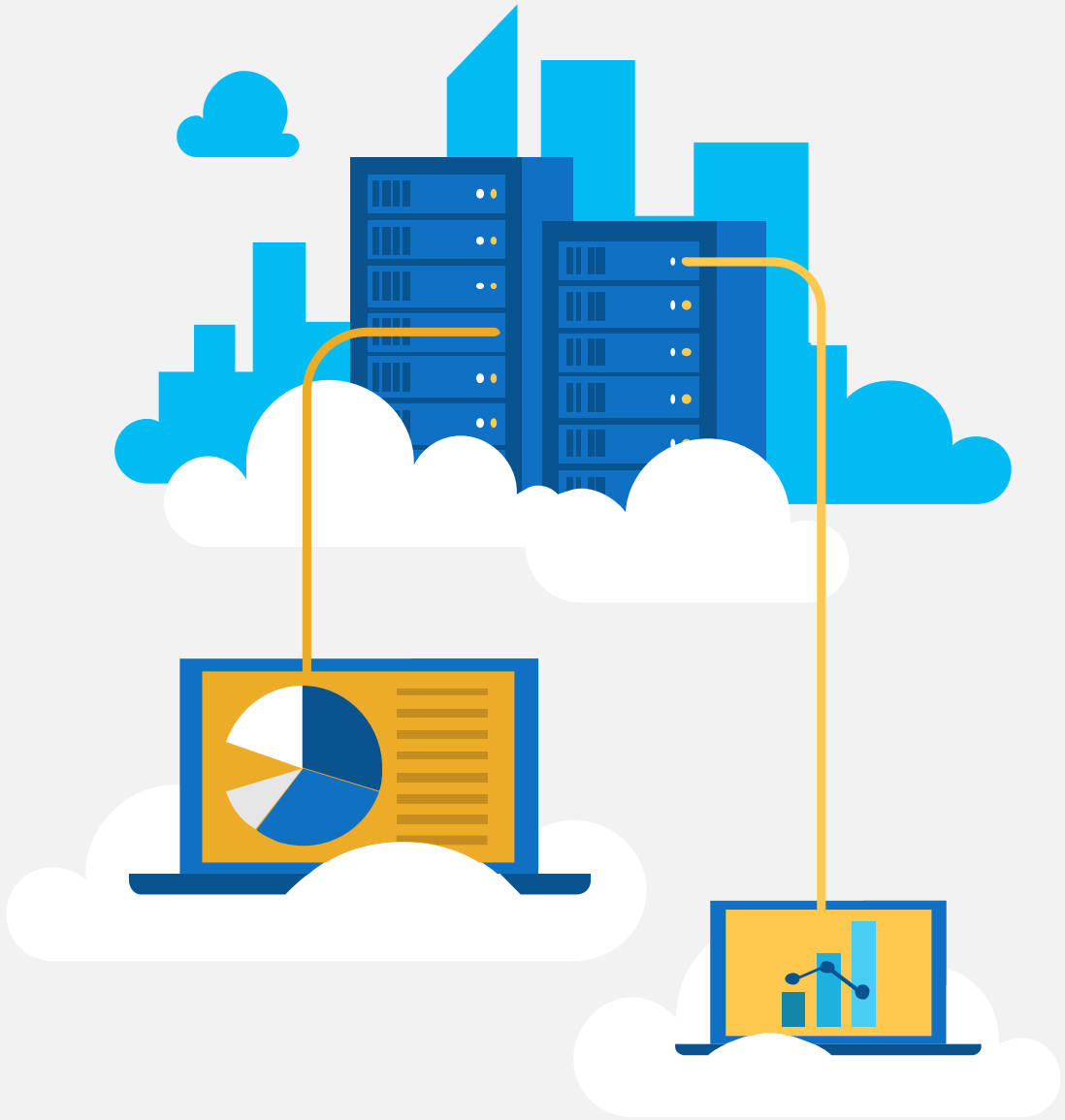
As CETAS generates Parquet files, statistics will be automatically created when the first query targets this external table, resulting in improved performance.

Pop Quiz 2

True or False: Both COPY command AND Polybase require CONTROL permission

TRUE

FALSE

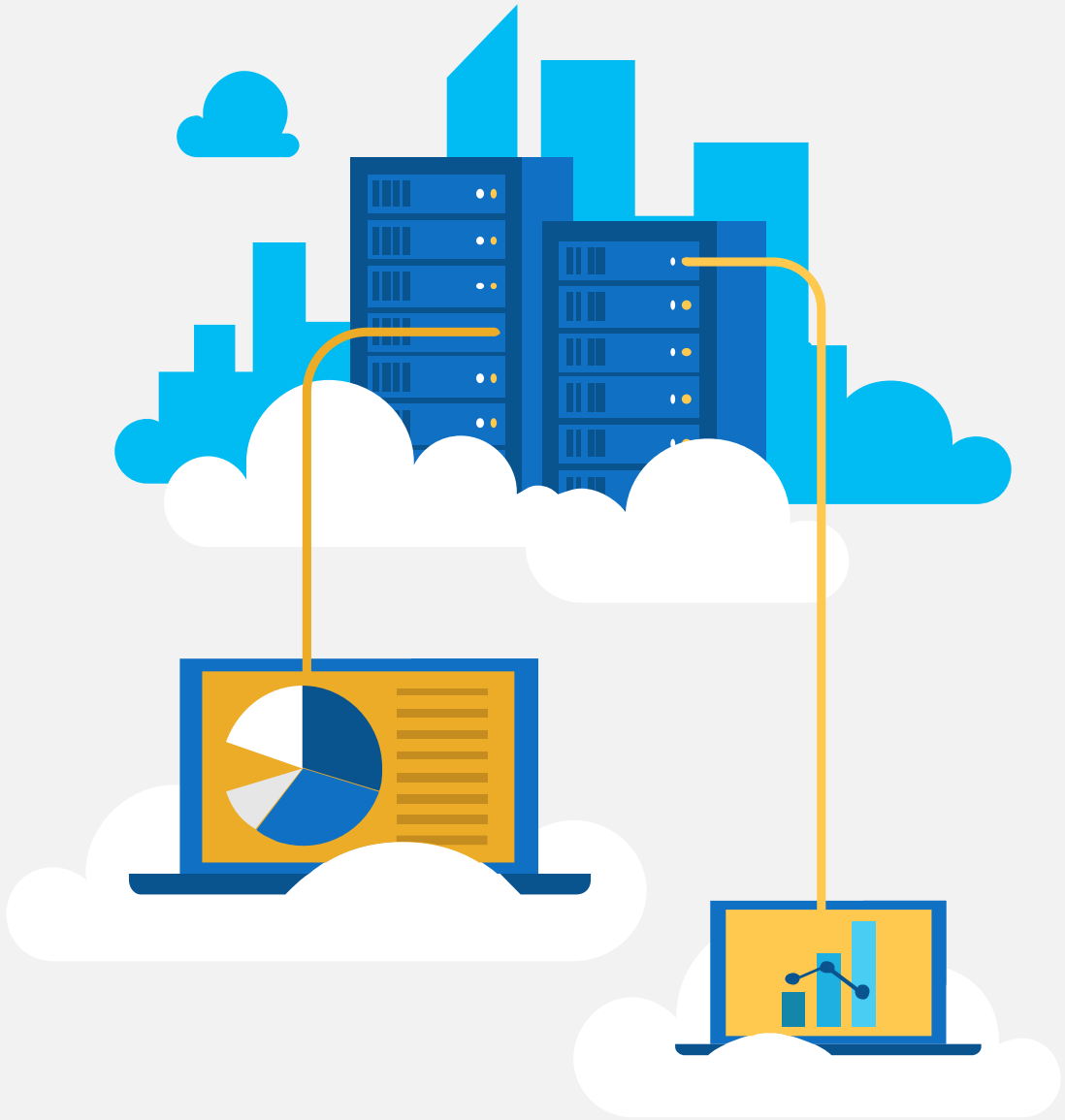


Pop Quiz 2

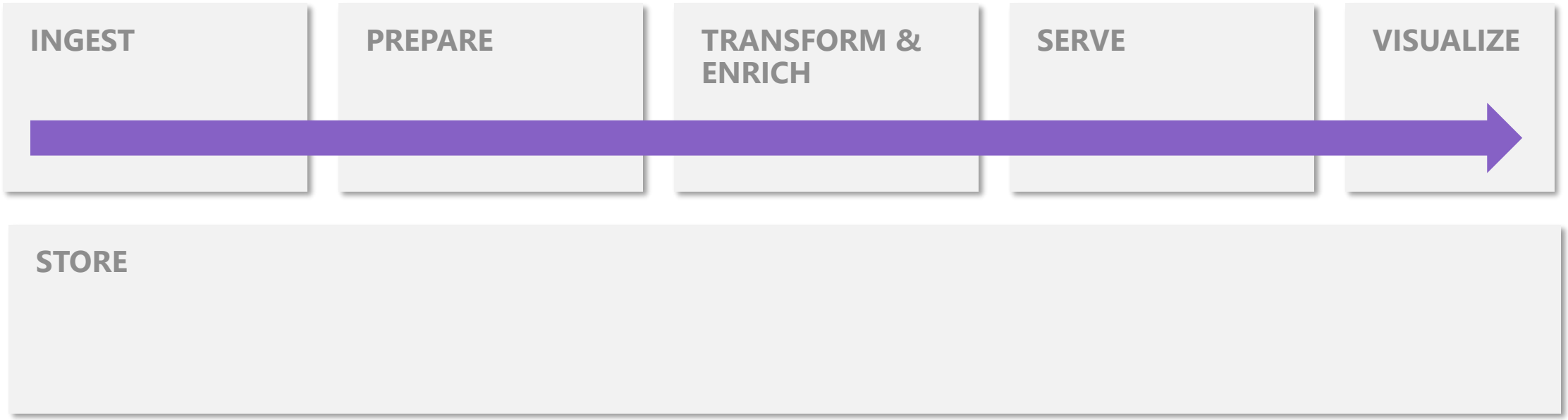
True or False: Both COPY command AND Polybase require CONTROL permissions

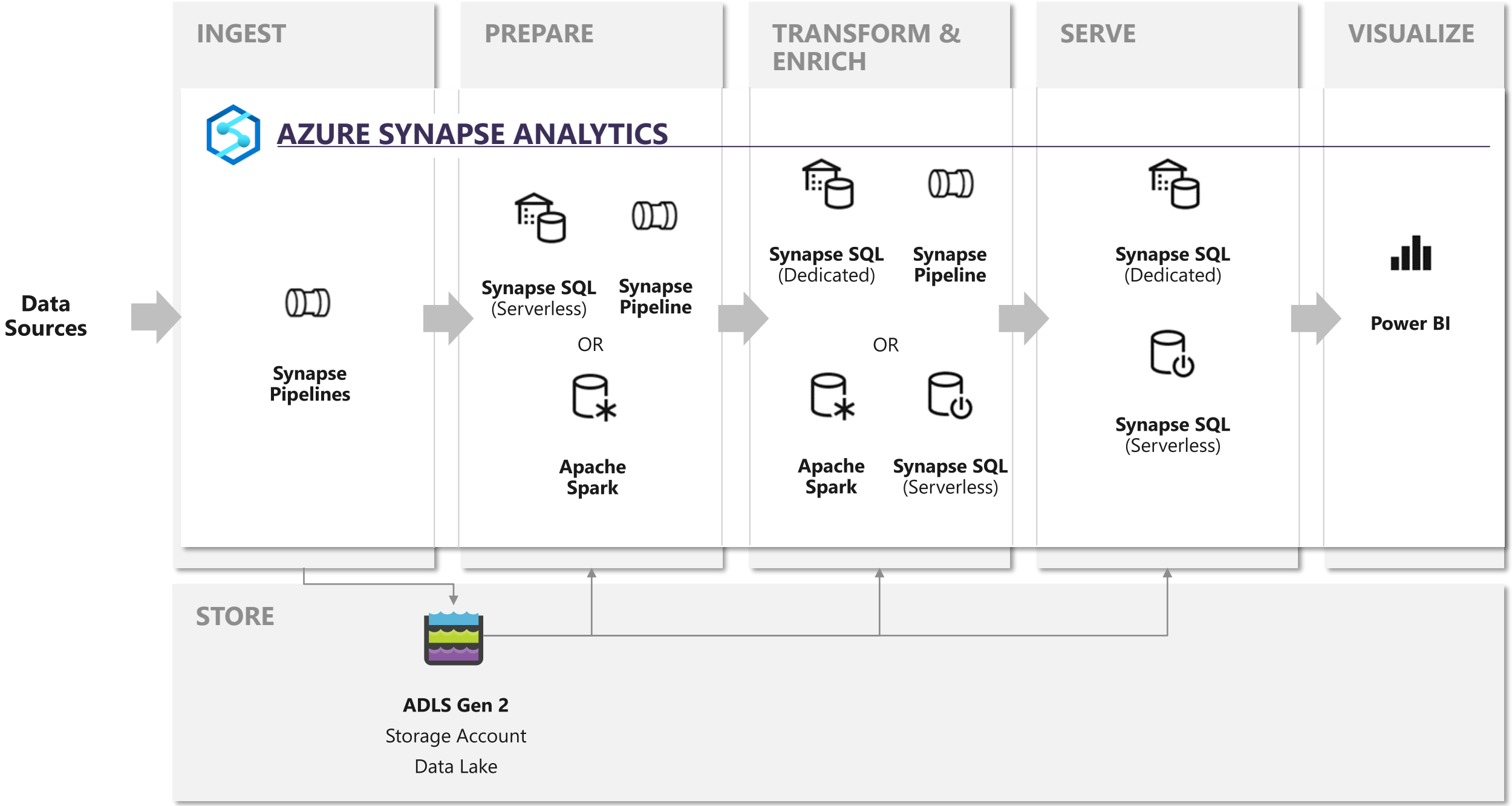
TRUE

FALSE



Modern Data Warehouse



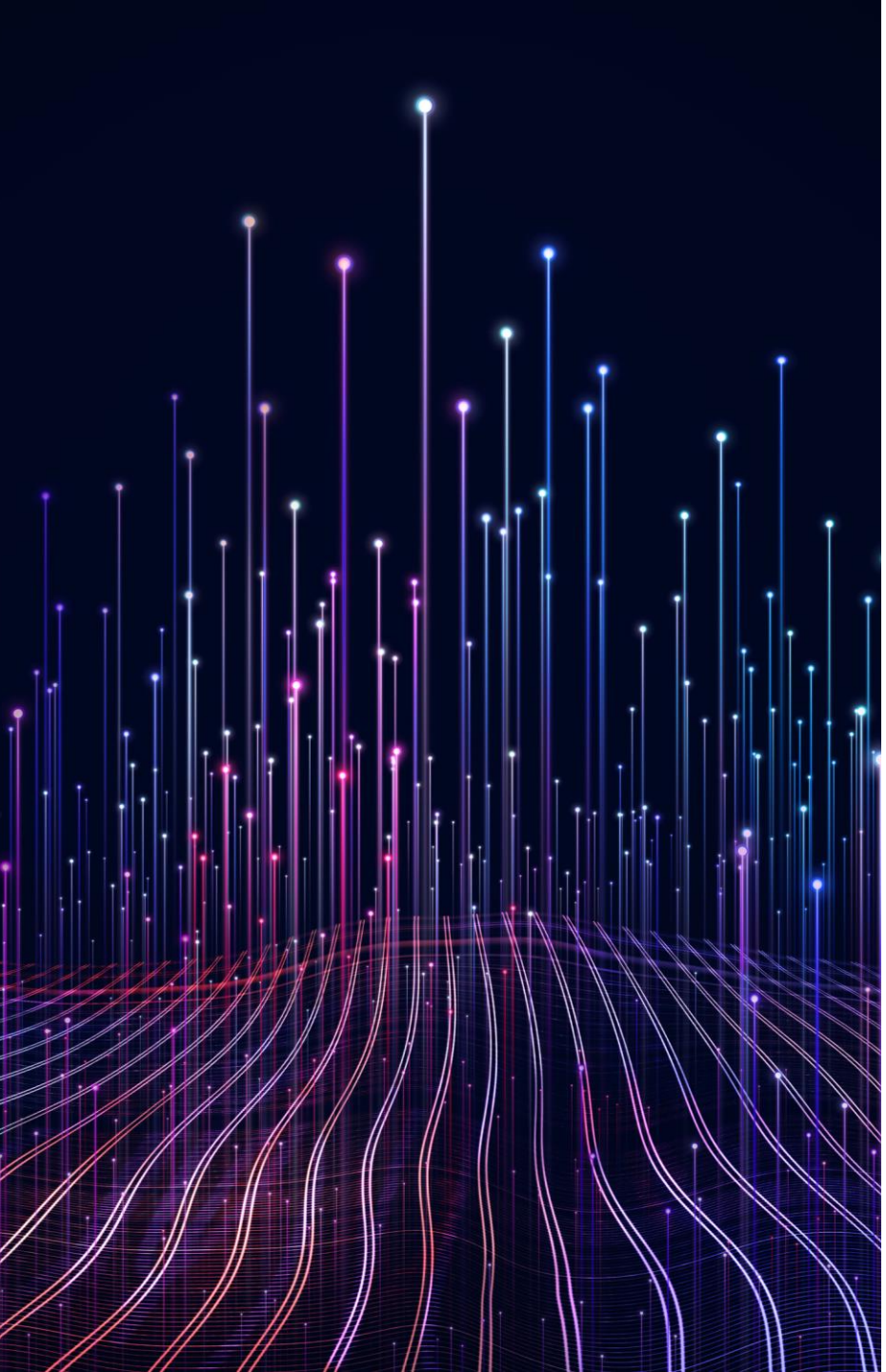

INGEST
PREPARE
TRANSFORM & ENRICH
SERVE
VISUALIZE
AZURE SYNAPSE ANALYTICS
Data Sources
Synapse Pipelines
Synapse SQL (Serverless)
Synapse Pipeline
OR
Apache Spark
Synapse SQL (Dedicated)
Synapse Pipeline
OR
Apache Spark
Synapse SQL (Serverless)
Synapse SQL (Dedicated)
Synapse SQL (Serverless)
Power BI
STORE
ADLS Gen 2
Storage Account
Data Lake

Microsoft Azure Solutions Business

Thank you

Microsoft Azure Solutions Business

Data Transformations



Agenda

1 **Preparing to transform**

Understanding and exploring the data.

2 **Apply transformations**

Apply coded and code-free transformations.

3 **Serverless transforms**

Use Azure Synapse serverless SQL to transform data with SQL scripts.

4 **Transform with Spark**

Here we have an example of what the agenda item would look like.

5 **Best practices**

Best practices for data transformation.

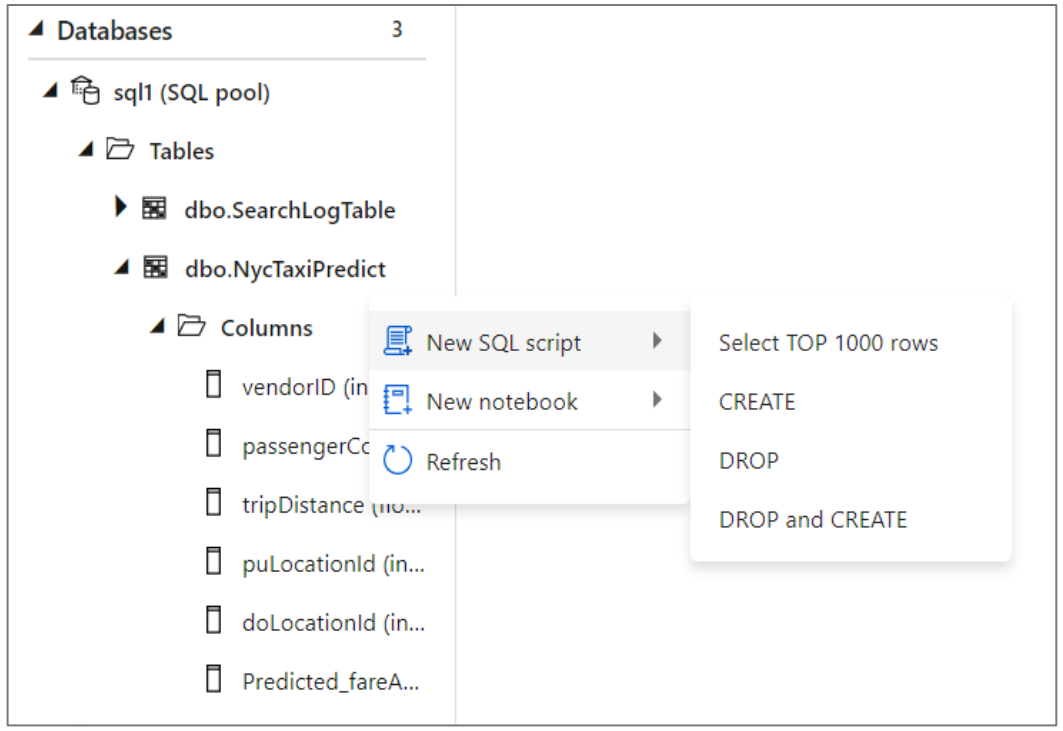
Typical Data Transformations

- Create persistent staging area / data vault
- Standardize data from different sources
- Remove duplicate rows
- Impute missing values
- Calculate derived values
- Prepare data for facts and dimensions

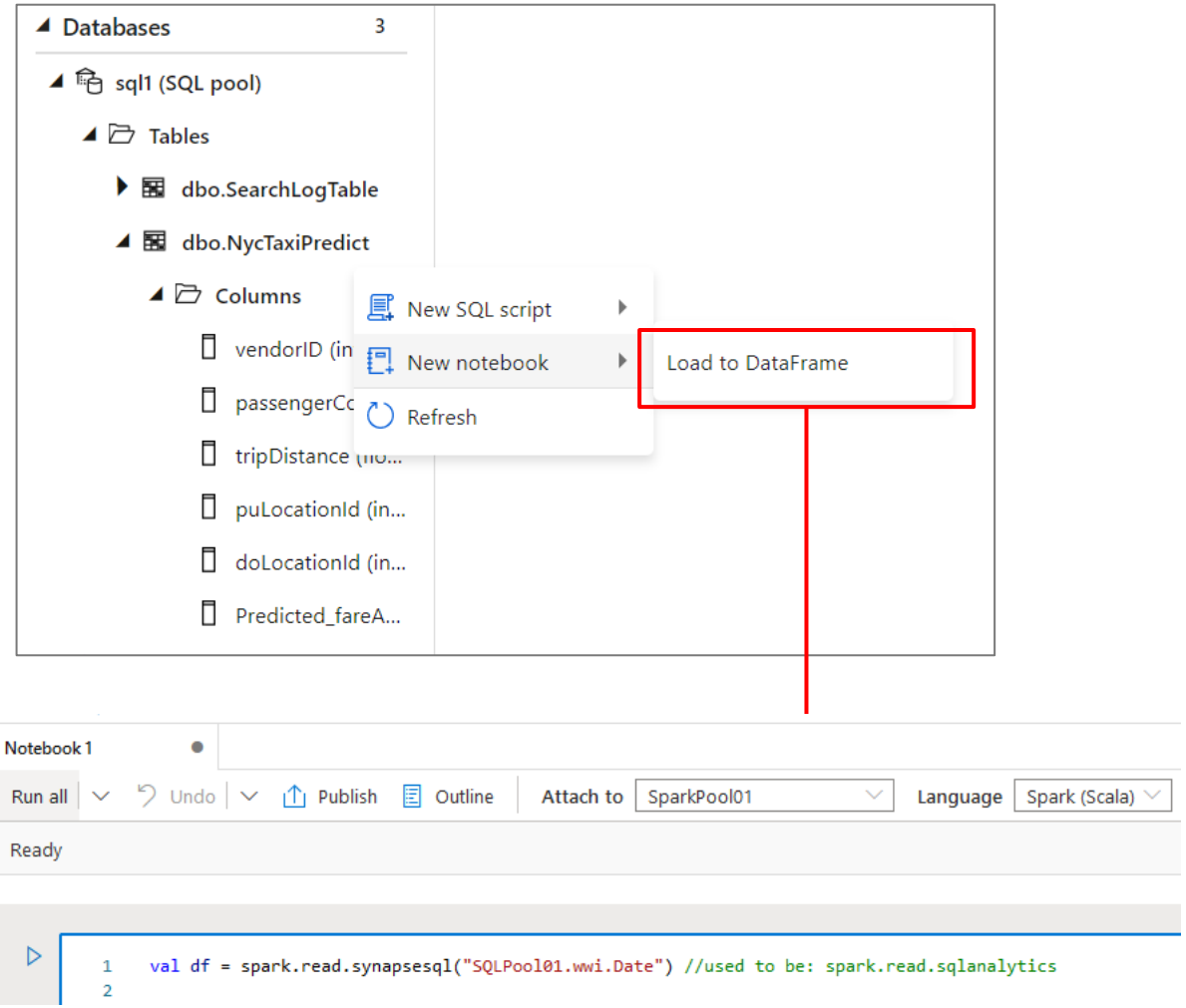
Applying transformations

Code based transformations

Familiar gesture to generate T-SQL scripts from SQL metadata objects such as tables.

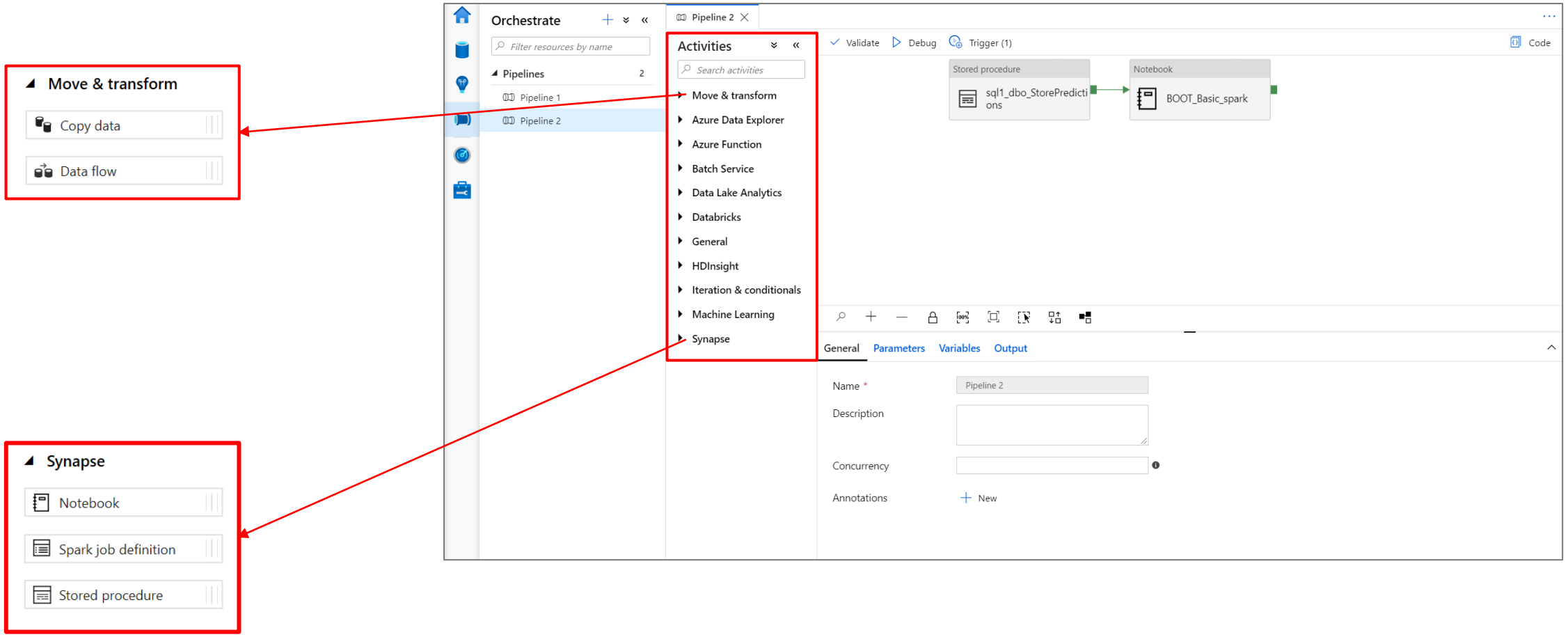


Starting from a table, auto-generate a single line of PySpark code that makes it easy to load a SQL table into a Spark dataframe and author transforms in a notebook.



Transform with Pipelines

Orchestrate transformations with Synapse Pipelines.



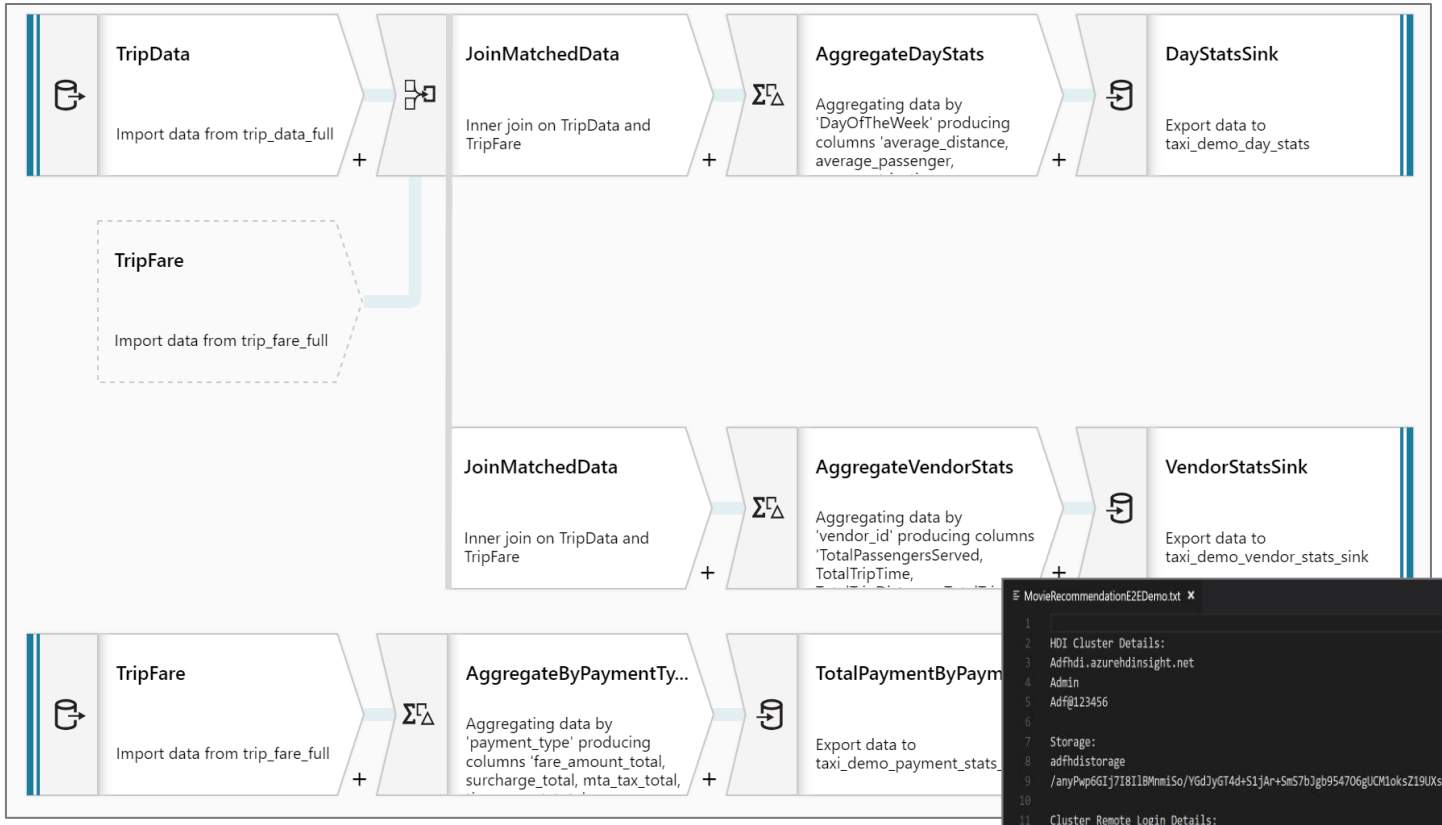
No Code Transform with Data Flows

Overview

It offers data cleansing, transformation, aggregation, conversion, etc

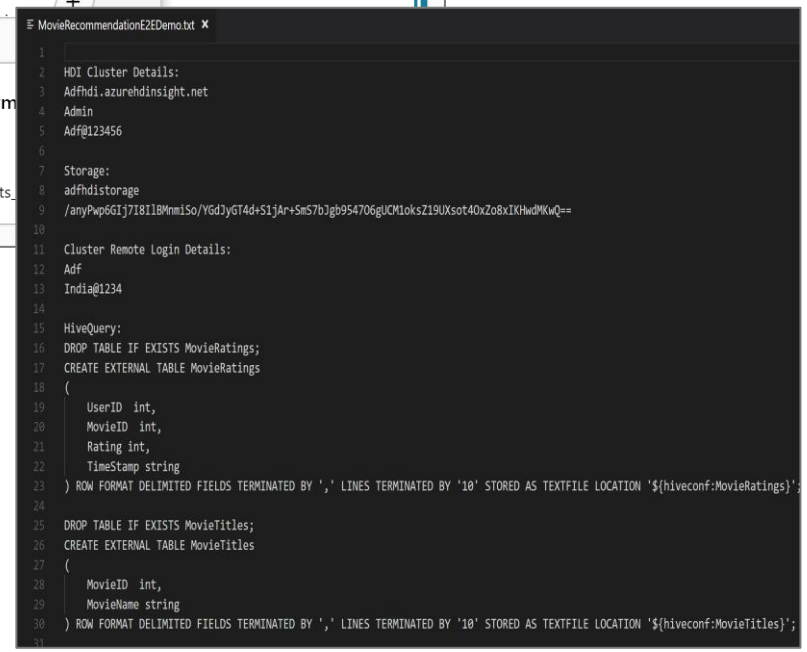
Benefits

- Cloud scale via Spark execution
- Guided experience to easily build resilient data flows
- Flexibility to transform data per user's comfort
- Monitor and manage dataflows from a single pane of glass



This…

NOT this…



Transform with serverless SQL

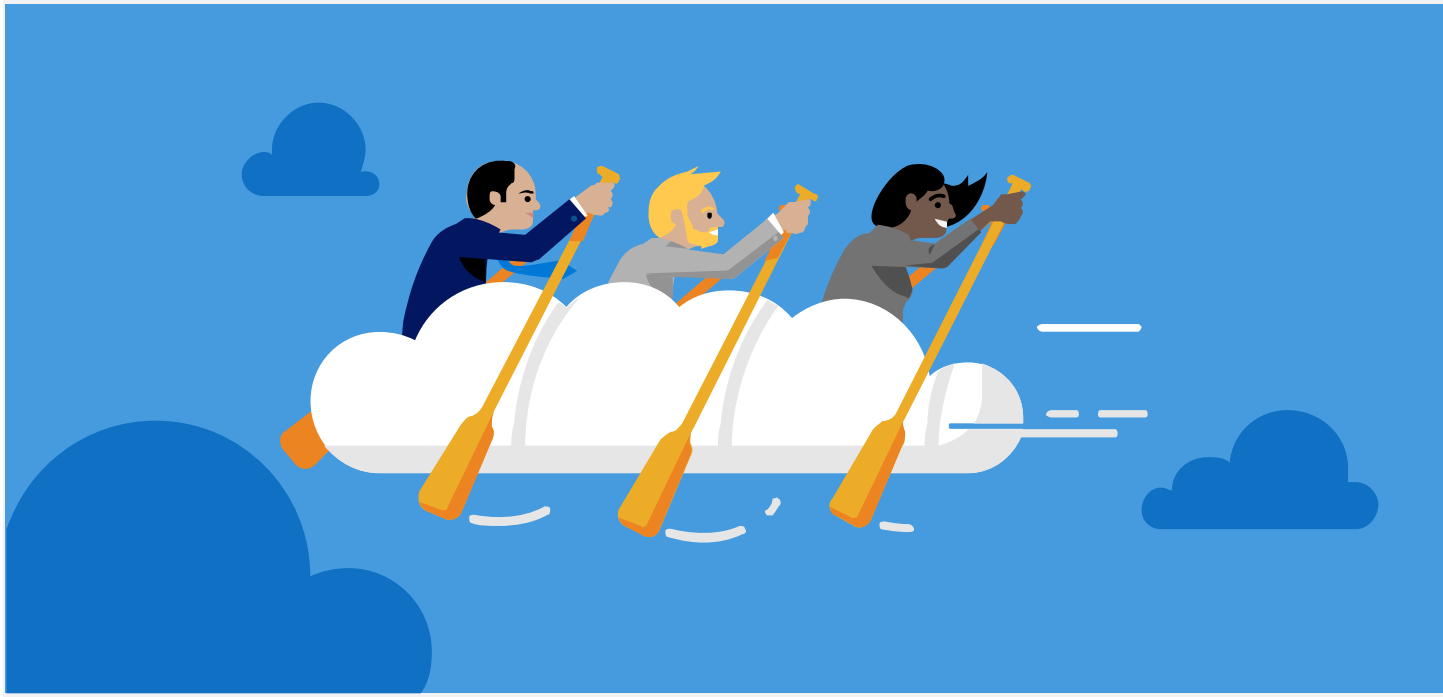
Pop Quiz 1

What's the largest scale TPC-H workload serverless SQL has successfully run?

A) 100TB

B) 1PB

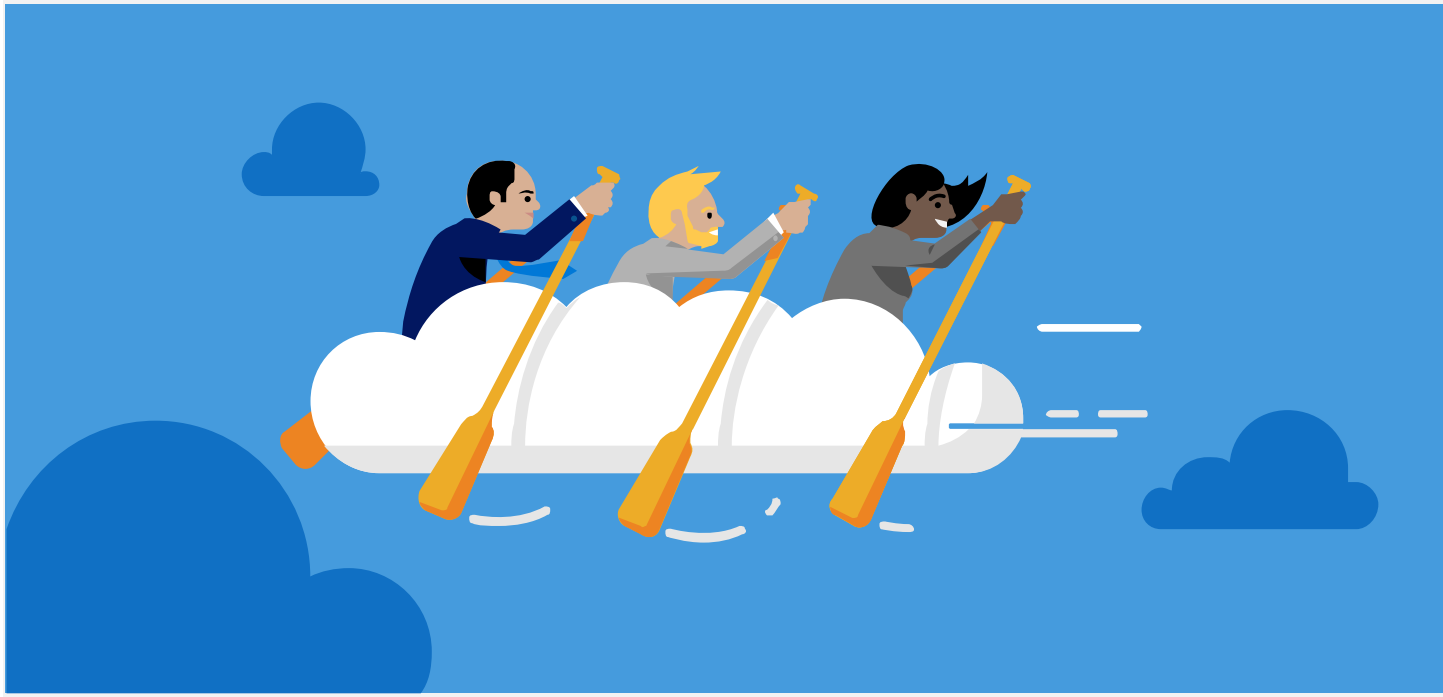
C) 10PB



Pop Quiz 1

What's the largest scale TPC-H workload a serverless SQL pool has successfully run?

A)	**B)**	C)
100TB	**1PB**	10PB



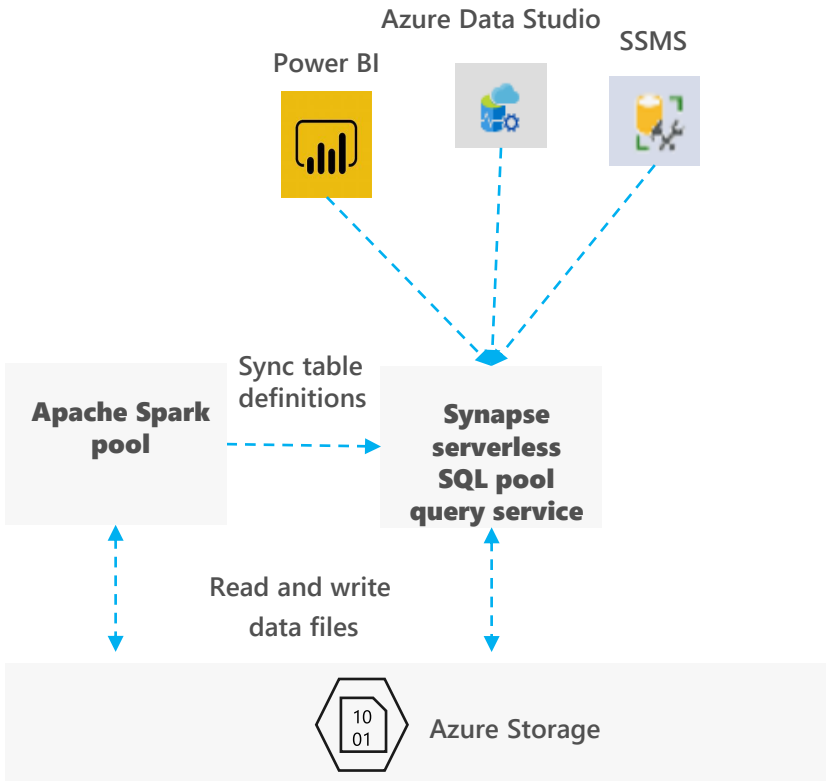
serverless SQL pool

Overview

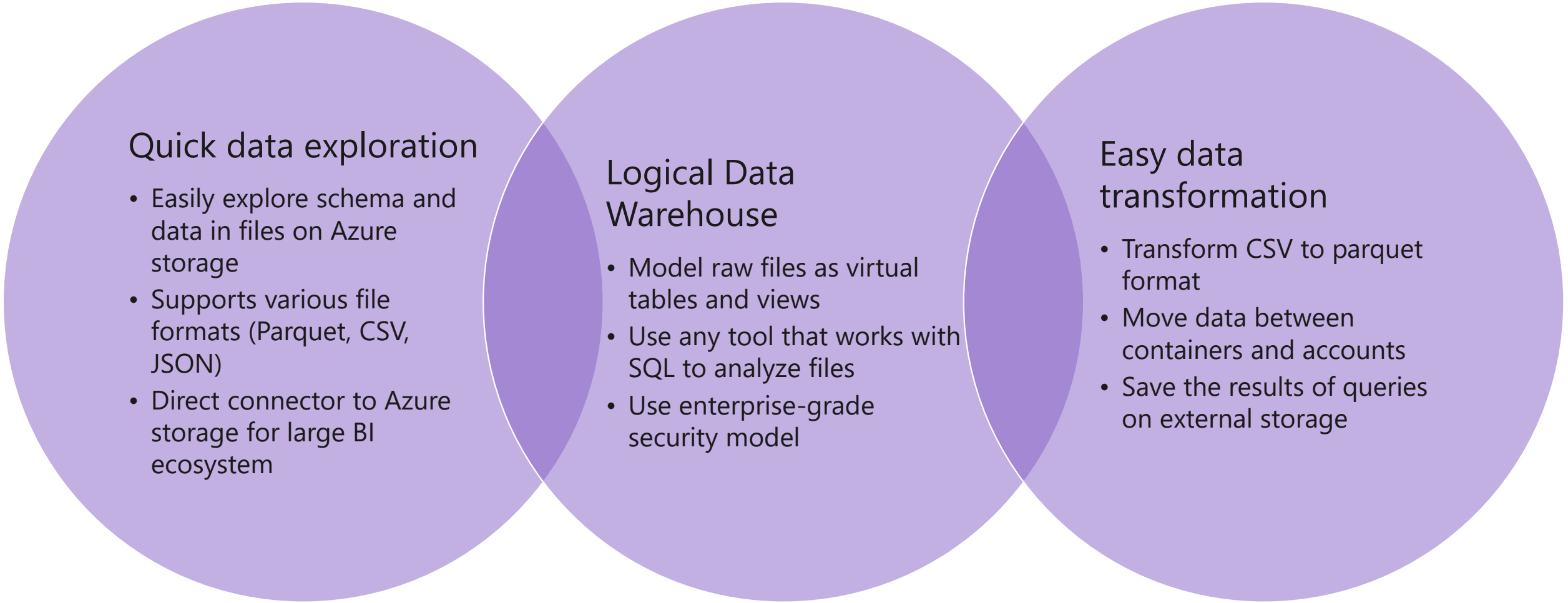
An interactive query service that enables you to use standard T-SQL queries over files in Azure storage.

Benefits

- Use SQL to work with files on Azure storage
  - Directly query files on Azure storage using T-SQL
  - Logical Data Warehouse on top of Azure storage
  - Easy data transformation of Azure storage files
- Supports any tool or library that uses T-SQL to query data
- Automatically synchronize tables from Spark
- Serverless
  - No infrastructure, no upfront cost, no resource reservation
  - Pay only for query execution (per data processed)

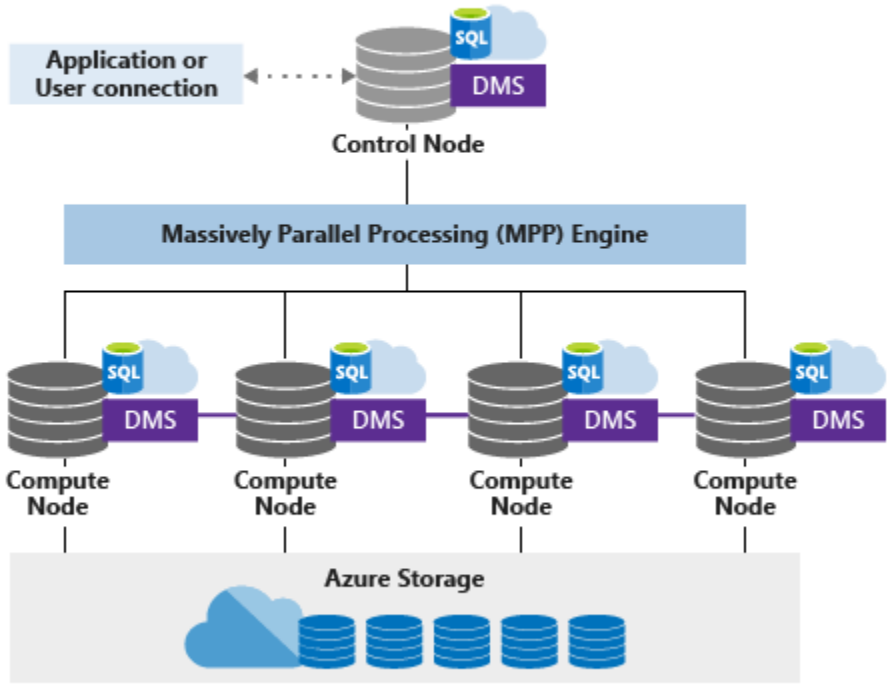


Recommended usage scenarios

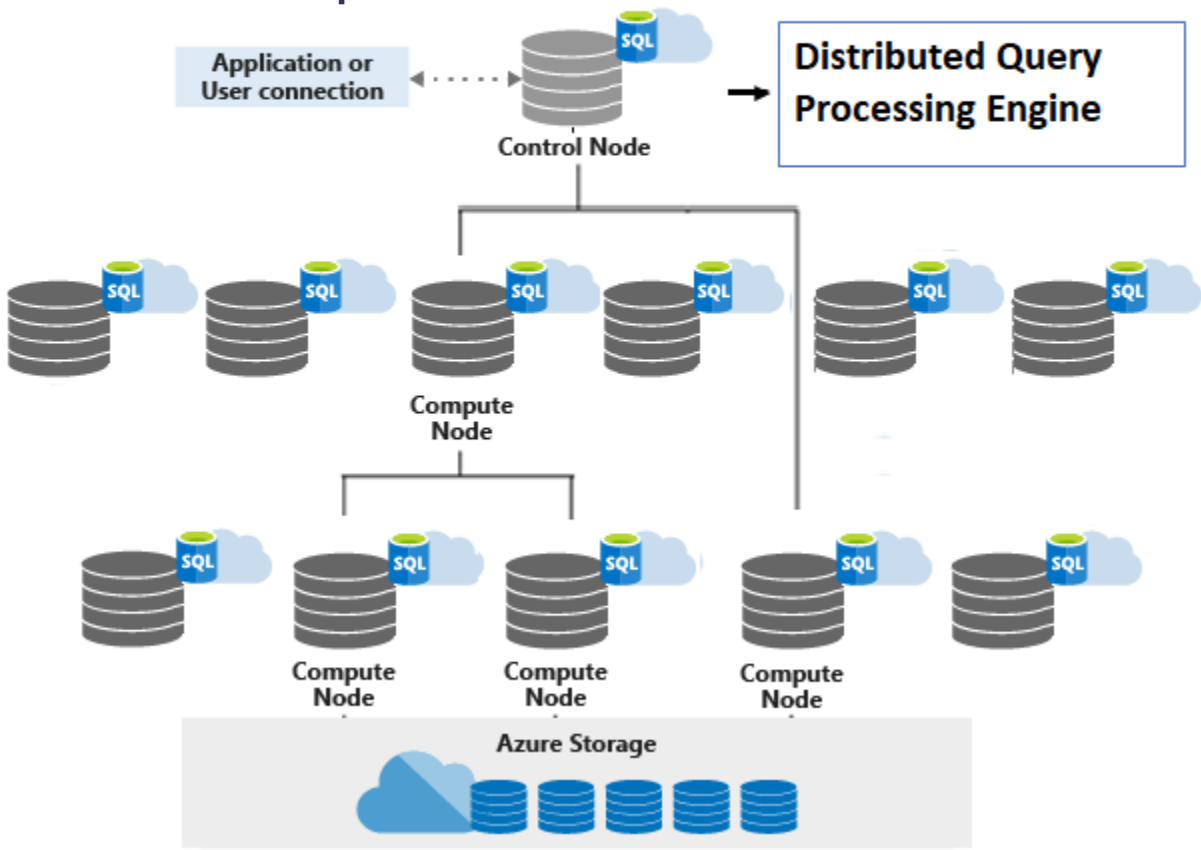


serverless SQL pool

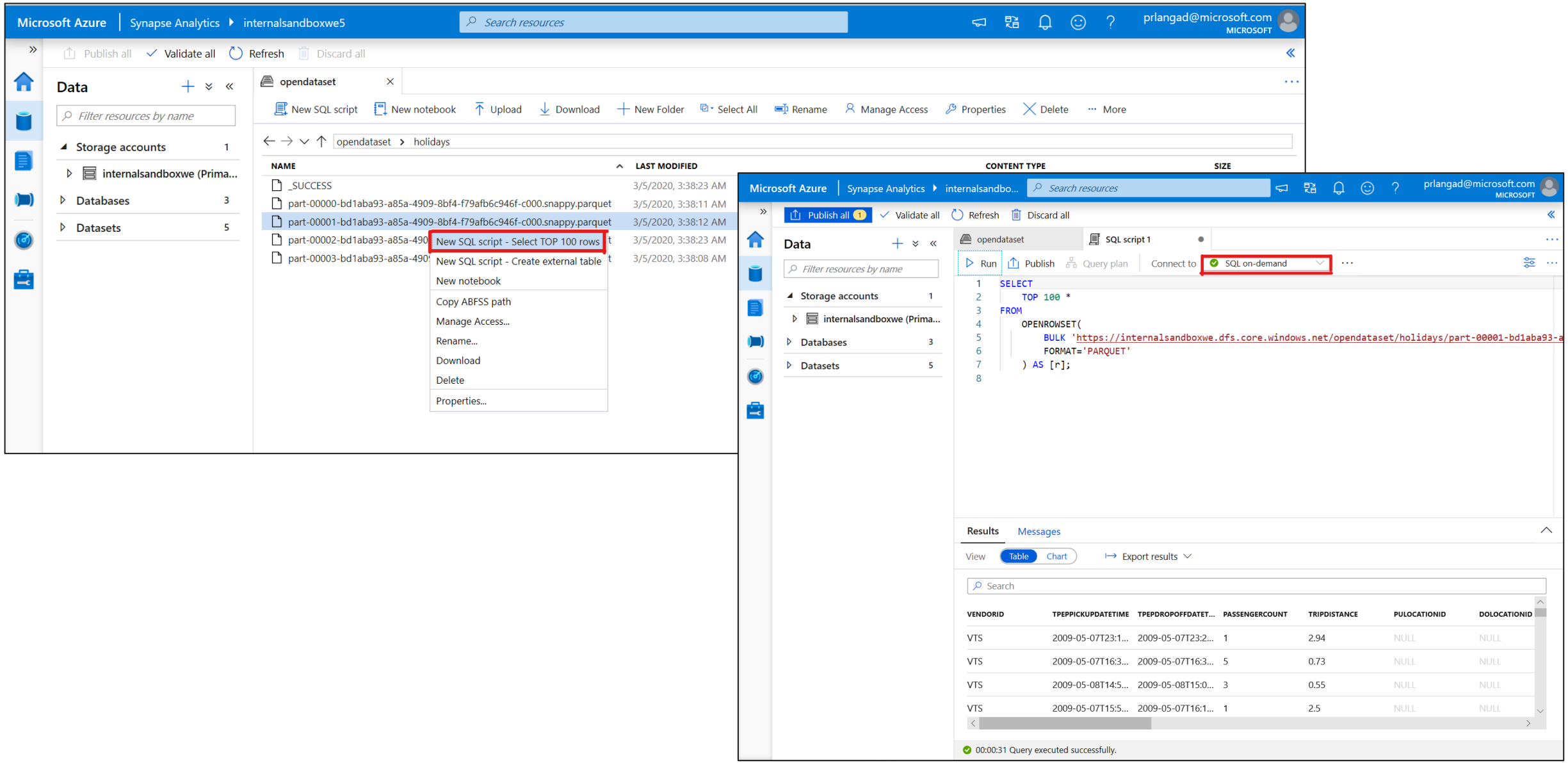
dedicated SQL pool



serverless SQL pool



Easily explore files on storage



Easily query files in various formats

Overview

Use OPENROWSET function to access data stored in various file formats

Benefits

Enables you to read CSV, parquet, and JSON files

Provides unified T-SQL interface for all file types

Use standard SQL language to transform and analyze returned data

- Use JSON functions to get the data from underlying files.
- Use JSON functions to get data from PARQUET nested types

```
SELECT TOP 10 *
FROM OPENROWSET(
      BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.csv',
      FORMAT = 'CSV')
WITH (
        country_code VARCHAR(4),
        country_name VARCHAR(50),
        year INT,
        population INT
        ) AS nyc
```

```
SELECT TOP 10 *
FROM OPENROWSET(
      BULK 'https://XYZ.blob.core.windows.net/parquet/taxi/*.parquet',
      FORMAT = 'PARQUET') AS nyc
```

```
SELECT TOP 10 *
  JSON_VALUE(jsonContent, '$.countryCode') AS country_code,
  JSON_VALUE(jsonContent, '$.countryName') AS country_name,
  JSON_VALUE(jsonContent, '$.year')  AS year
  JSON_VALUE(jsonContent, '$.population')  AS population
FROM  OPENROWSET(
      BULK 'https://XYZ.blob.core.windows.net/json/taxi/*.json',
      FORMAT='CSV',
      FIELDTERMINATOR ='0x0b',
      FIELDQUOTE = '0x0b',
      ROWTERMINATOR = '0x0b'
   )
   WITH ( jsonContent varchar(MAX)   ) AS json_line
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Automatic schema inference

Overview

OPENROWSET will automatically determine columns and types of data stored in external file.

Benefits

No need to up-front analyze file structure to query the file

OPENROWSET identifies columns and their types based on underlying file metadata.

Perfect solution for data exploration where schema is unknown.

The functionality is available for both parquet & CSV files.

```
SELECT TOP 10 *
FROM OPENROWSET(
        BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.parquet',
        FORMAT = 'PARQUET') AS nyc
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

```
SELECT
  TOP 100 *
FROM
  OPENROWSET(
    BULK 'https://azuresynapsesa.dfs.core.windows.net/default/RetailData/StoreDemoGraphics.csv',
    FORMAT = 'CSV',
    PARSER_VERSION='2.0',
HEADER_ROW = TRUE) AS [result]
```

StoreId	RatioAge60	CollegeRatio	Income	HighIncome15...	LargeHH	MinoritiesRatio	More1FullTime...	DistanceNeare...	SalesN
2	0.232864734	0.248934934	10.55320518	0.463887065	0.103953406	0.114279949	0.303585347	2.110122129	1.1428
5	0.117368032	0.32122573	10.92237097	0.535883355	0.103091585	0.053875277	0.410568032	3.801997814	0.6818

Defined the query result schema inline

Overview

Specify columns and types at query time.

Benefits

Define result schema at query time in WITH clause.

No need for external format files.

Explicitly define exact return types, their sizes, and collations.

Improve performance by column elimination in parquet files.

```
SELECT TOP 10 *
FROM OPENROWSET(
        BULK 'https://XYZ.blob.core.windows.net/csv/taxi/*.csv',
        FORMAT = 'CSV')
WITH (
        country_code VARCHAR(4),
        country_name VARCHAR(50),
        year INT,
        population INT
        ) AS nyc
```

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Customize the content parsing to fit your case

Overview

Uses OPENROWSET function to access data from various types of CSV files.

Benefits

Ability to read CSV files with custom format

- With or without header row
- Handle any new-line terminator (Windows or Unix style)
- Use custom field terminator and quote character
- Read UTF-8 and UTF-18 encoded files
- Use only a subset of columns by specifying column position after column types

```
SELECT *
FROM OPENROWSET(
    BULK 'https://XYZ.blob.core.windows.net/csv/population/population.csv',
    FORMAT = 'CSV',
    FIELDTERMINATOR =',',
    ROWTERMINATOR = '\n'
)
WITH (
    [country_code] VARCHAR (5) 2,
    [country_name] VARCHAR (100) 4,
    [year] smallint 7,
    [population] bigint 9
) AS [r]
WHERE
    country_name = 'Luxembourg'
    AND year = 2017
```

Second, fourth, seventh and ninth columns are returned

	country_code	country_name	year	population
1	LU	Luxembourg	2017	594130

Easily query multiple files, with wildcards

Overview

Uses OPENROWSET function to access data from multiple files or folders using wildcards in path

Benefits

Offers reading multiple files/folders through usage of wildcards

Offers reading specific file/folder

Supports use of multiple wildcards

```
SELECT YEAR(pickup_datetime) as [year],
        SUM(passenger_count) AS passengers_total,
        COUNT(*) AS [rides_total]
FROM OPENROWSET(
       BULK 'https://XYZ.blob.core.windows.net/csv/taxi/year=*/month=1/*.parquet',
       FORMAT = 'PARQUET')  AS nyc
GROUP BY YEAR(pickup_datetime)
ORDER BY YEAR(pickup_datetime)
```

	year	passengers_total	rides_total
1	2001	14	10
2	2002	29	16
3	2003	22	16
4	2008	378	188
5	2009	594	353
6	2016	102093687	61758523
7	2017	184464988	113496932
8	2018	86272771	53925040
9	2019	37	29
…	2020	6	6

Query partitioned data, using the folder structure

Overview

Uses OPENROWSET function to access data partitioned in sub-folders

Benefits

Use filepath() function to access actual values from file paths.

Eliminate sub-folders/partitions before the query starts execution

Query Spark/Hive partitioned data sets

```
SELECT
    r.filepath(1) AS [year]
   ,r.filepath(2) AS [month]
   ,COUNT_BIG(*) AS [rows]
FROM OPENROWSET(
      BULK 'https://XYZ.blob.core.windows.net/year=*/month=*/*.parquet',
      FORMAT = 'PARQUET') AS [r]
WHERE r.filepath(1) IN ('2017')
   AND r.filepath(2) IN ('10', '11', '12')

GROUP BY    r.filepath() ,r.filepath(1) ,r.filepath(2)
ORDER BY    filepath
```

year	month	rows
2017	10	9768815
2017	11	9284803
2017	12	9508276

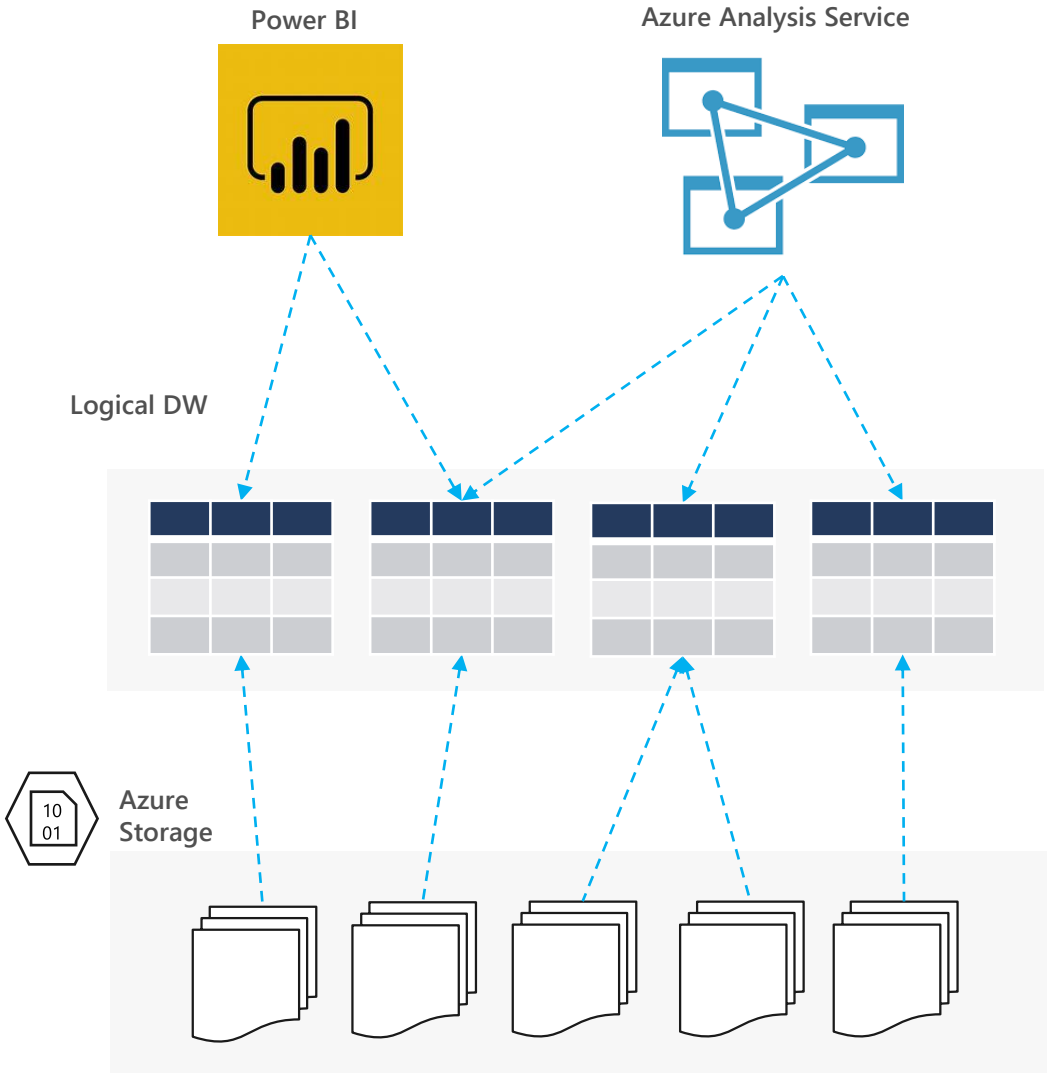
Synapse serverless SQL pool as a logical data warehouse

Overview

Logical relational layer on top of physical files in Azure Storage.

Benefits

- Abstract physical storage and file formats using well understandable relational concepts such as tables and views.
- Direct connector to Azure storage for large ecosystem of BI tools
- BI tools that use SQL can work with files on storage
  - Analytic tools use external tables that represent proxy to actual files.
  - No need for custom connectors in BI tools.
- Provides complex data processing (joining and aggregation) on top of raw files.
- Apply enterprise-ready security model and access control using battle-tested SQL Server permission model on top of Azure storage files



Logical Data Warehouse views

Overview

serverless SQL pool logical data warehouse views are created on external files placed in customer Azure storage

Benefits

Create SQL views on externally stored data

Access files using the view from various tools and language

Leverage rich T-SQL language to process and analyze data in external files exposed via views

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP VIEW IF EXISTS  populationView
GO

CREATE VIEW populationView AS
SELECT *
FROM OPENROWSET(
      BULK 'https://XYZ.blob.core.windows.net/csv/population/*.csv',
      FORMAT = 'CSV',
      FIELDTERMINATOR =',',
      ROWTERMINATOR = '\n'
   )
WITH (
   [country_code] VARCHAR (5) ,
   [country_name] VARCHAR (100),
   [year] smallint,
   [population] bigint
) AS [r]
```

```
SELECT
   country_name, population
FROM populationView
WHERE
   [year] = 2019
ORDER BY
   [population] DESC
```

	country_name	population
1	China	1389618778
2	India	1311559204
3	United States	331883986
4	Indonesia	264935824
5	Pakistan	210797836
6	Brazil	210301591
7	Nigeria	208679114
8	Bangladesh	161062905
9	Russia	141944641
10	Mexico	127318112

Logical Data Warehouse - tables

Overview

Create external tables that reference external files in your serverless SQL pool logical data warehouse

Benefits

Create external tables that reference set of files on Azure storage.

Join and transform multiple tables in the same query.

Enables you to analyze external files with the same experience that you have in classic databases.

Manage column statistics in external tables.

Manage access rights per table.

Create PowerBI reports on the views created on external data

```
USE [mydbname]
GO

DROP TABLE IF EXISTS dbo.Population
GO
CREATE EXTERNAL TABLE dbo.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 = MyAzureStorage,
      FILE_FORMAT = MyAzureCSVFormat
)
```

```
CREATE STATISTICS stat_country_name
ON dbo.Population(country_name);
```

```
SELECT
   country_name, population
FROM population
WHERE year = 2019
ORDER BY population DESC
```

	country_name	population
1	China	1389618778
2	India	1311559204
3	United States	331883986
4	Indonesia	264935824
5	Pakistan	210797836
6	Brazil	210301591
7	Nigeria	208679114
8	Bangladesh	161062905
9	Russia	141944641
10	Mexico	127318112

Easy data transformation

Overview

Easily perform data transformations of Azure Storage files using SQL queries

Optimize data pipeline - achieve more using serverless SQL pool

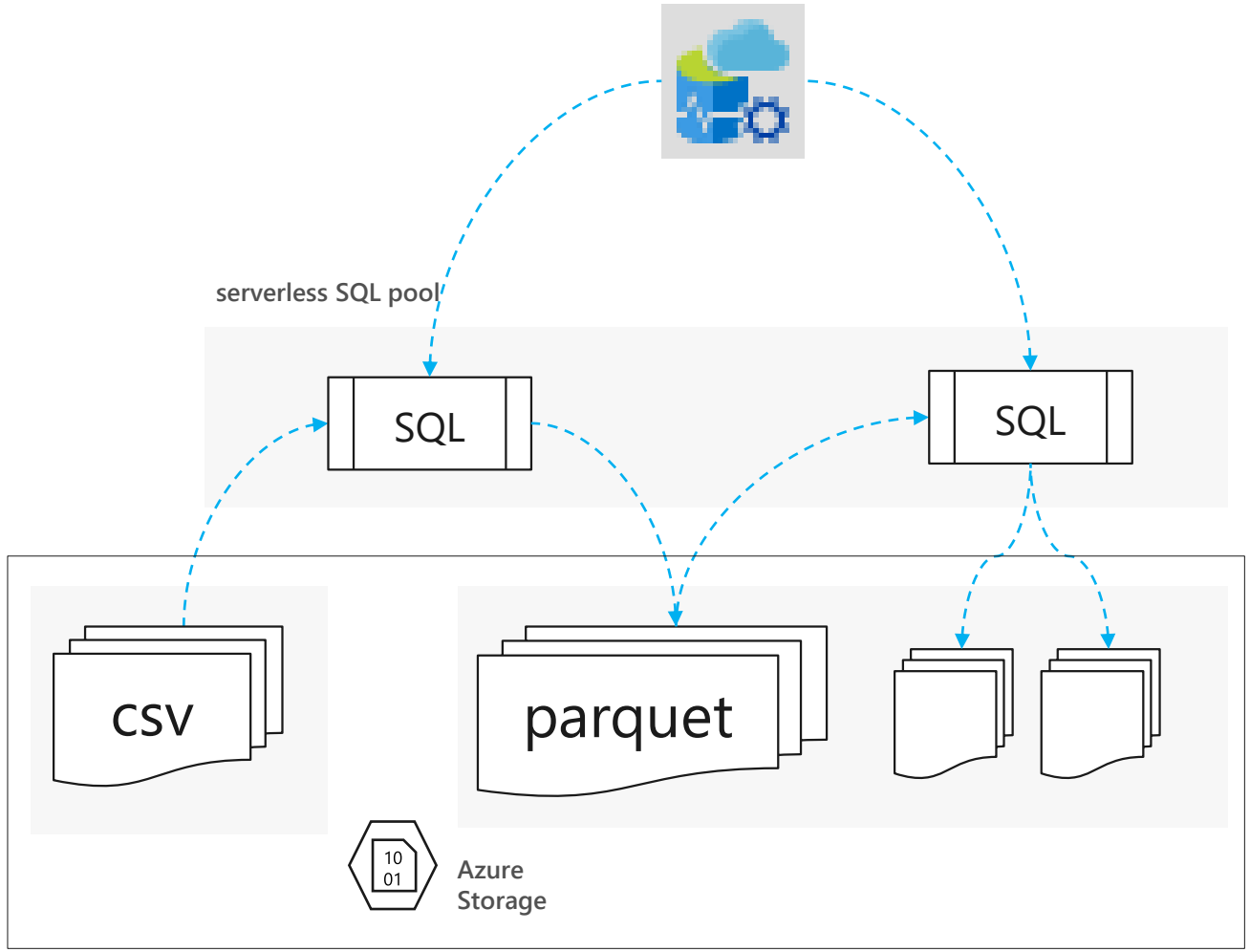
Benefits

Single statement transformations:

- convert CSV or JSON files to Parquet
- copy files from one storage account to another
- re-partition data to new location(s)
- store results of your query on Azure Storage

SQL ETL pipelines

- Use SQL commands to transform data
- Chain SQL statement for build ETL process
- Materialize reports created on the current snapshot of data



Easy data transformation with CETAS

Overview

Create external tables as select (CETAS) enables you to easily transform data and store the results of query on Azure storage

Benefits

Select any data set and store it in parquet format.

Pre-calculate and store results of query and store them permanently on Azure storage.

Use saved data using external table.

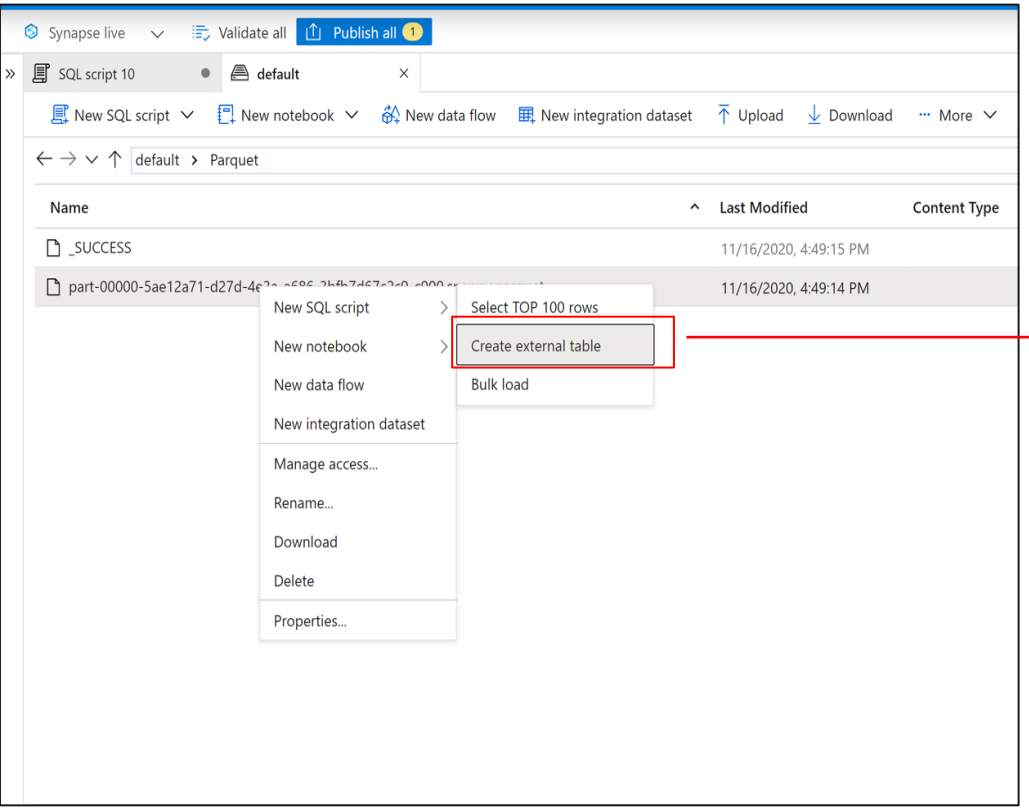
Improve performance of your reports by permanently storing the result based on current snapshot of data as parquet files.

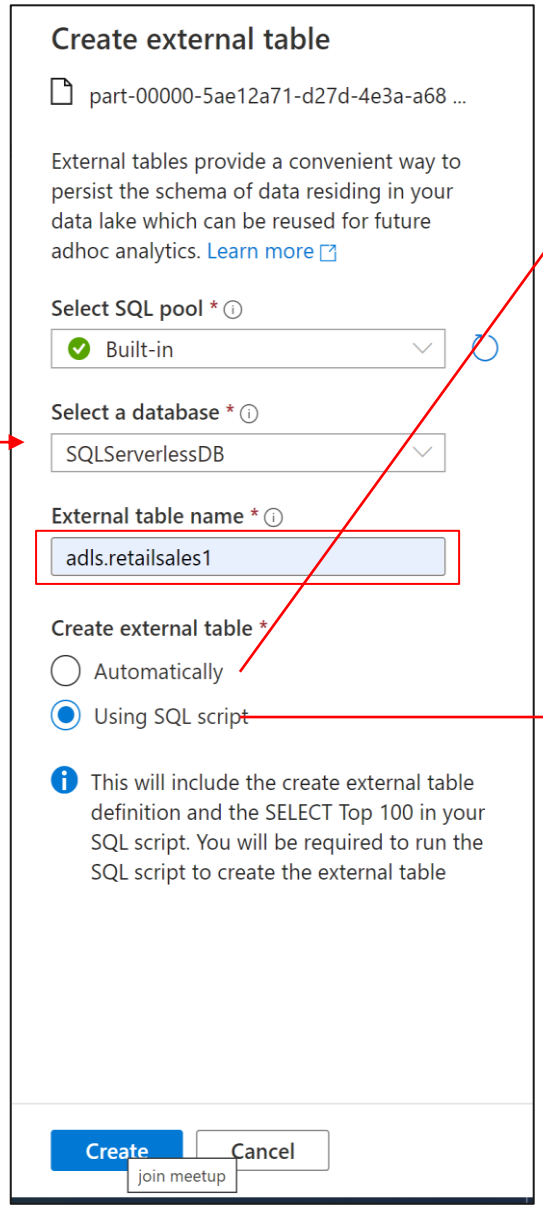
```
-- copy CSV dataset into parquet data set
CREATE EXTERNAL TABLE parquet.Population
WITH(
      LOCATION = '/parquet/population',
      DATA_SOURCE = MyAzureStorage,
      FILE_FORMAT = MyAzureParquetFormat  )
AS
SELECT *
FROM csv.Population

-- pre-create report using new parquet data-set
CREATE EXTERNAL TABLE parquet.PopulationByMonth2017
WITH(
      LOCATION = '/parquet/population/bymonth/2017',
      DATA_SOURCE = MyAzureStorage,
      FILE_FORMAT = MyAzureParquetFormat  )
AS
SELECT month = p.month, population = COUNT ( p.population )
FROM parquet.Population p
WHERE p.year = 2017
GROUP BY p.month

-- Reporting tools can now directly read data from pre-created report
SELECT *
FROM parquet.PopulationByMonth2017
```

UI based data transformation






```
SELECT TOP 100 * FROM adls.retailsale
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormat')
    CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
    WITH ( FORMAT_TYPE = PARQUET)
GO

IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'default_azuresynapsesa_dfs_core_windows_net')
    CREATE EXTERNAL DATA SOURCE [default_azuresynapsesa_dfs_core_windows_net]
    WITH (
        LOCATION   = 'https://azuresynapsesa.dfs.core.windows.net/default',
    )
Go

CREATE EXTERNAL TABLE adls.retailsale (
    [storeId] varchar(8000),
    [productCode] varchar(8000),
    [quantity] varchar(8000),
    [logQuantity] varchar(8000),
    [advertising] varchar(8000),
    [price] varchar(8000),
    [weekStarting] varchar(8000),
    [id] varchar(8000)
    )
    WITH (
    LOCATION = 'Parquet/part-00000-5ae12a71-d27d-4e3a-a686-3bfb7d67c2c9-c000.snappy.parquet',
    DATA_SOURCE = [default_azuresynapsesa_dfs_core_windows_net],
    FILE_FORMAT = [SynapseParquetFormat]
    )
GO

SELECT TOP 100 * FROM adls.retailsale
```

Automatic syncing of Spark tables

Overview

Tables created in Spark pool are automatically created as external tables that reference external files in your serverless SQL pool logical data warehouse

Benefits

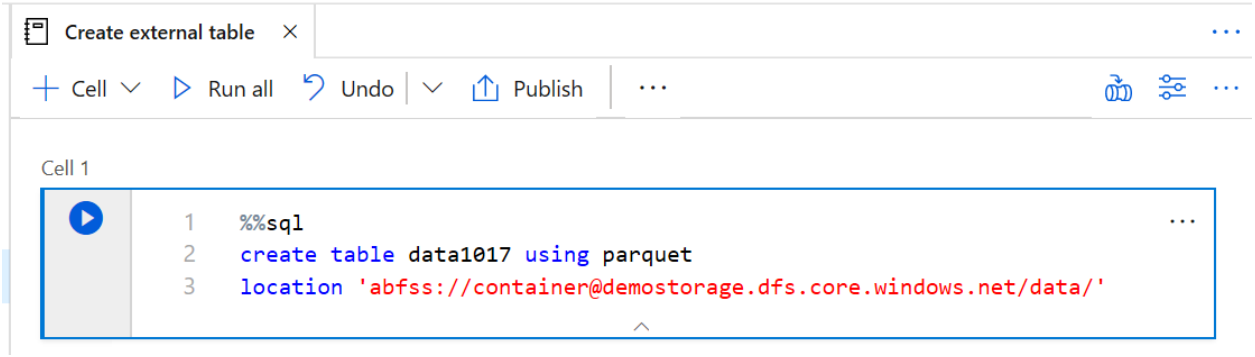
Tables designed using Spark languages are immediately available in serverless SQL pool.

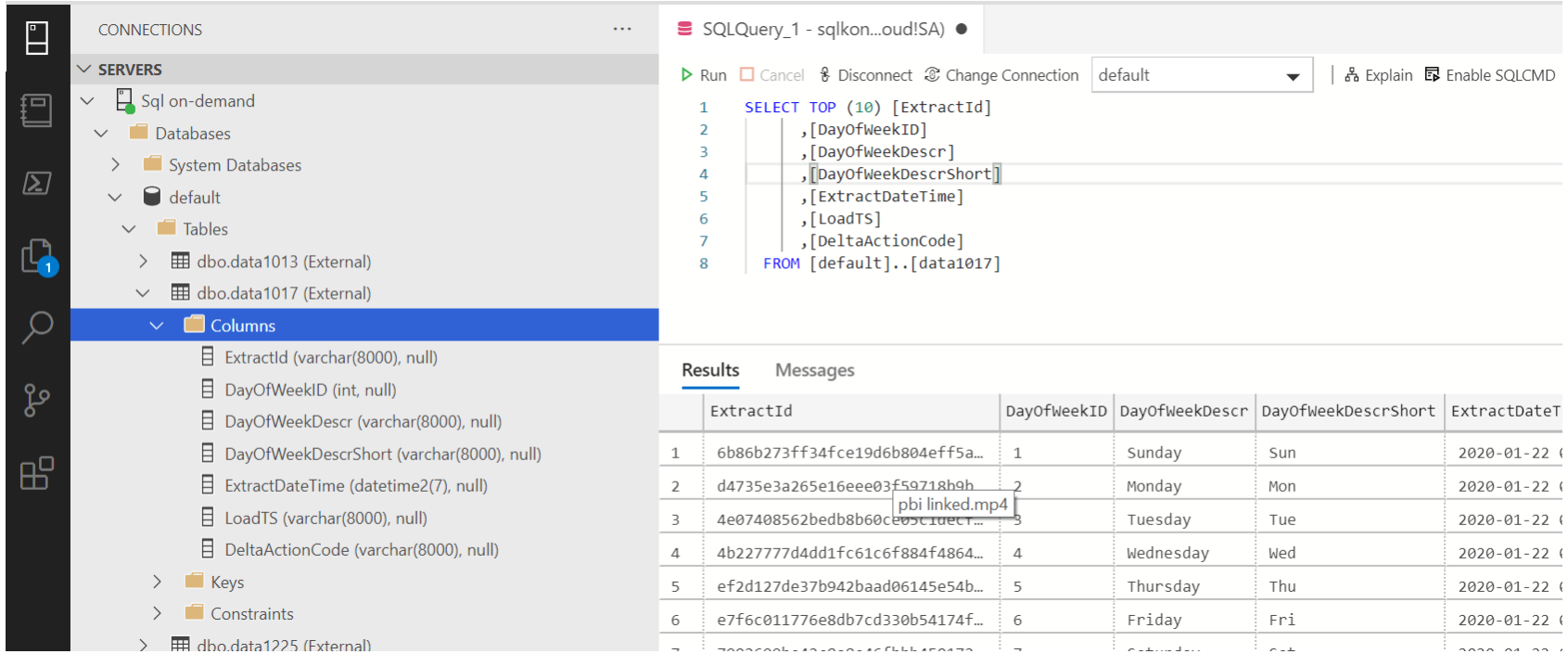
Schema definition matches original

Spark table updates are applied in serverless SQL pool

No need to manually create SQL tables that match Spark tables

Spark and serverless SQL pool tables reference the same external files.





Metastore

Overview

It offers the different computational engines of a workspace to share databases and Parquet-backed tables between its Apache Spark pools, serverless SQL pool, and dedicated SQL pool.

Benefits

- The shared metadata model supports the modern data warehouse pattern.
- The Spark created databases and all their tables become visible in any of the Azure Synapse workspace Spark pool instances and can be used from any of the Spark jobs provided necessary permissions are provided.
- Databases are created automatically in the serverless SQL pool metadata.
- The external and managed tables created by Spark job are made accessible as external tables in the serverless SQL pool metadata in the dbo schema of the corresponding database.
- Spark created databases and their Parquet-backed tables will be mapped into the SQL pools for which metadata synchronization enabled.

Transform with Spark

Transforming with Spark – Querying SQL Pools

Existing Approach

```
val jdbcUsername = "<SQL DB ADMIN USER>"
val jdbcPwd = "<SQL DB ADMIN PWD>"
val jdbcHostname = "servername.database.windows.net”
val jdbcPort = 1433
val jdbcDatabase ="<AZURE SQL DB NAME>“


val jdbc_url =
s"jdbc:sqlserver://${jdbcHostname}:${jdbcPort};database=${jdbcDatabase};
encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.databas
e.windows.net;loginTimeout=60;“

val connectionProperties = new Properties()

connectionProperties.put("user", s"${jdbcUsername}")
connectionProperties.put("password", s"${jdbcPwd}")

val sqlTableDf = spark.read.jdbc(jdbc_url, “dbo.Tbl1", connectionProperties)
```

New Approach Using Scala

```
// Construct a Spark DataFrame from SQL Pool table
var df = spark.read.sqlanalytics("sql1.dbo.Tbl1")

// Write the Spark DataFrame into SQL Pool table
df.write.sqlanalytics(“sql1.dbo.Tbl2”)
```

Using Python

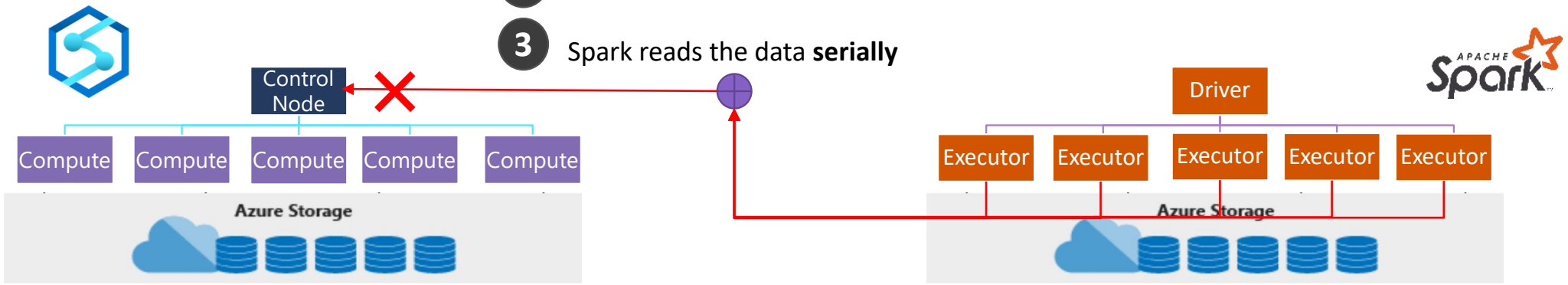
```
%%spark
var df = spark.read.sqlanalytics("sql1.dbo.Tbl1")
df.createOrReplaceTempView("tbl1")


%%pyspark
sample = spark.sql("SELECT * FROM tbl1")
sample.createOrReplaceTempView("tblnew")


 %%spark
 var df = spark.sql("SELECT * FROM tblnew")
 df.write.sqlanalytics(“sql1.dbo.tbl2",
 Constants.INTERNAL)
```

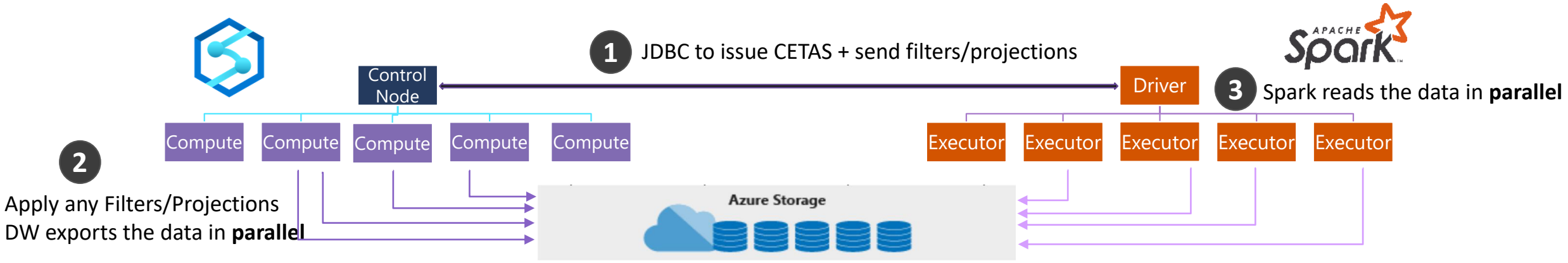
Existing Approach: JDBC

1. JDBC to open connection
2. Apply any Filters/Projections
3. Spark reads the data **serially**



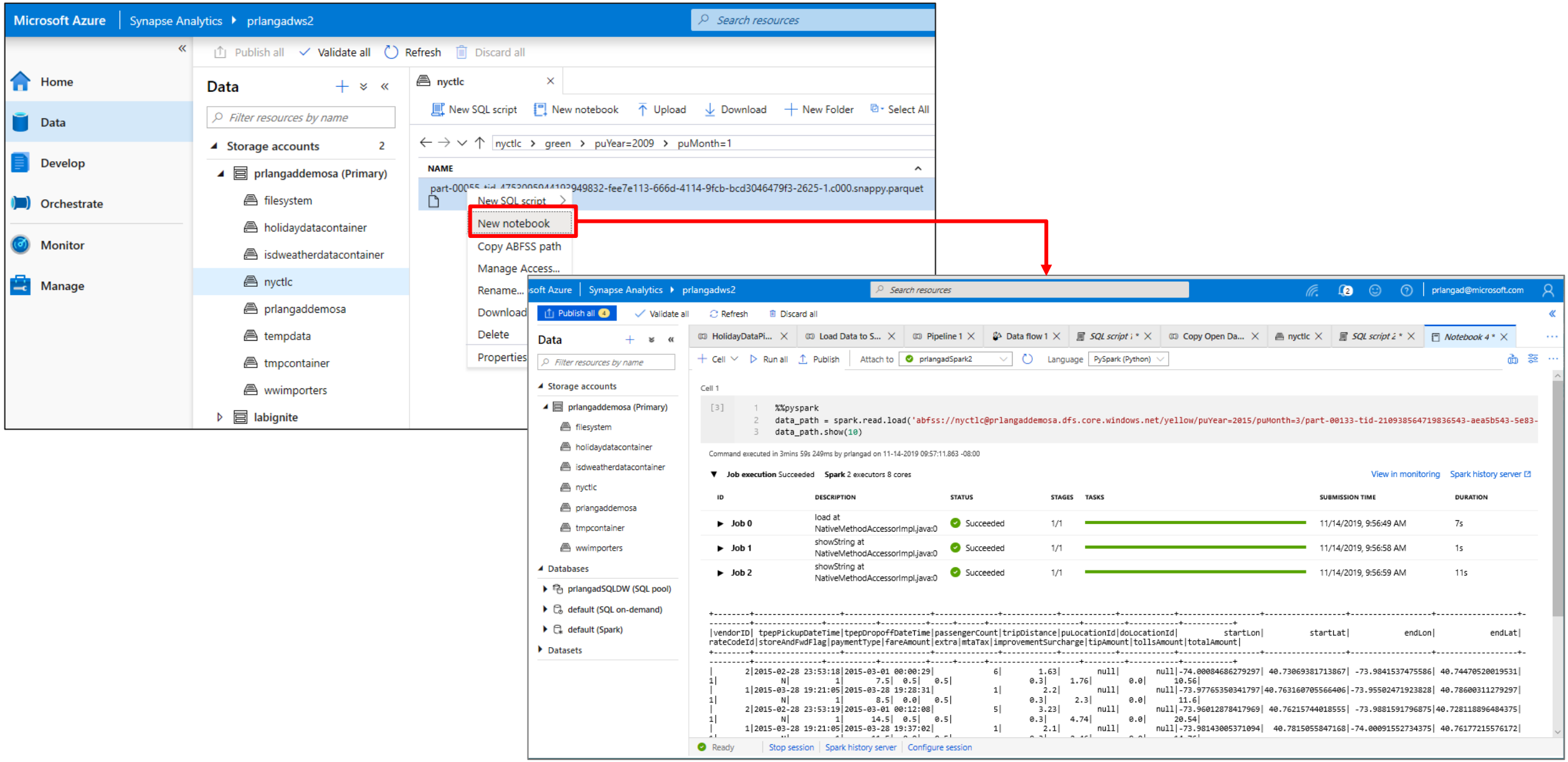
New Approach: JDBC and Polybase

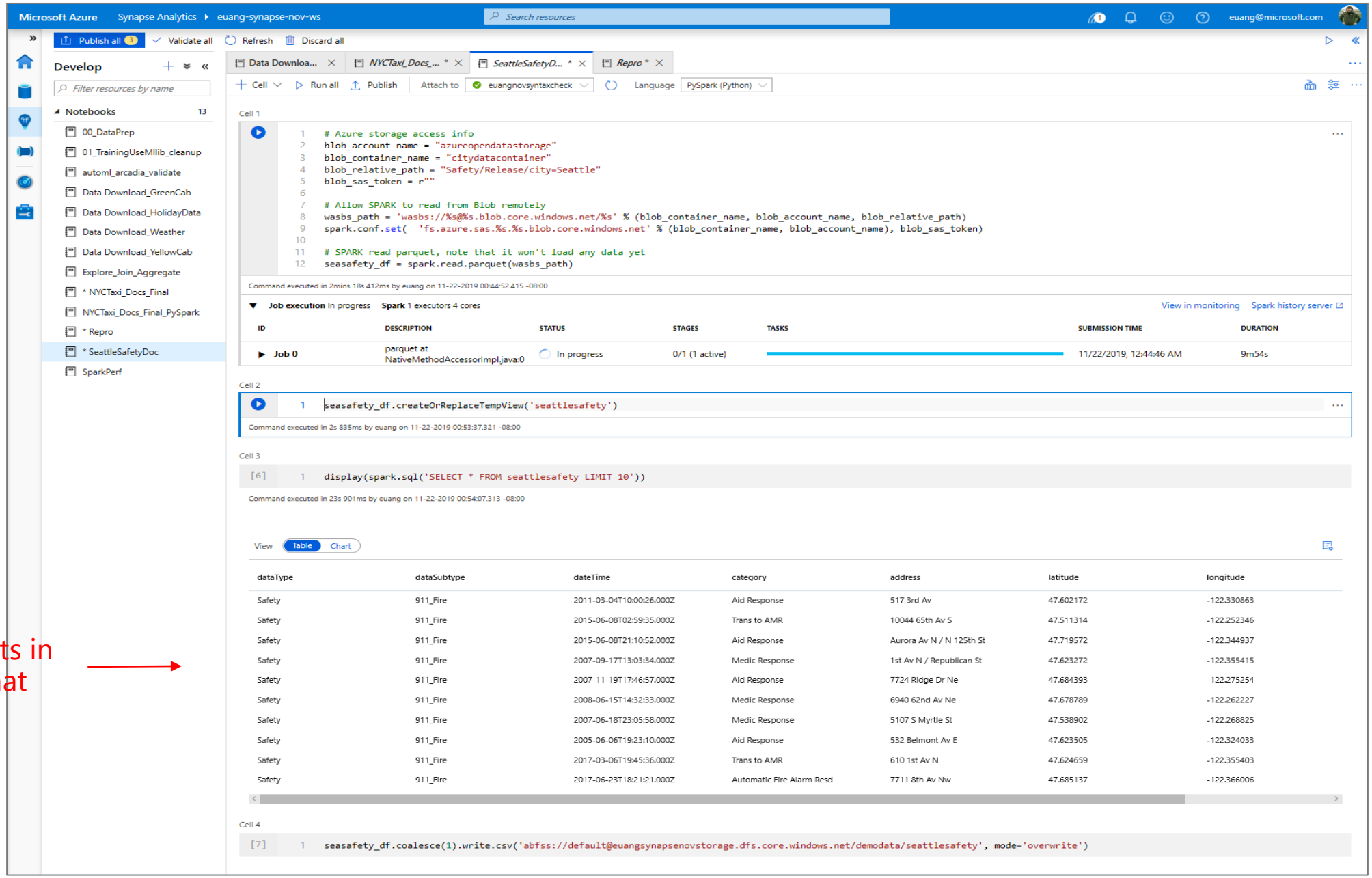
1. JDBC to issue CETAS + send filters/projections
2. Apply any Filters/Projections
   DW exports the data in **parallel**
3. Spark reads the data in **parallel**



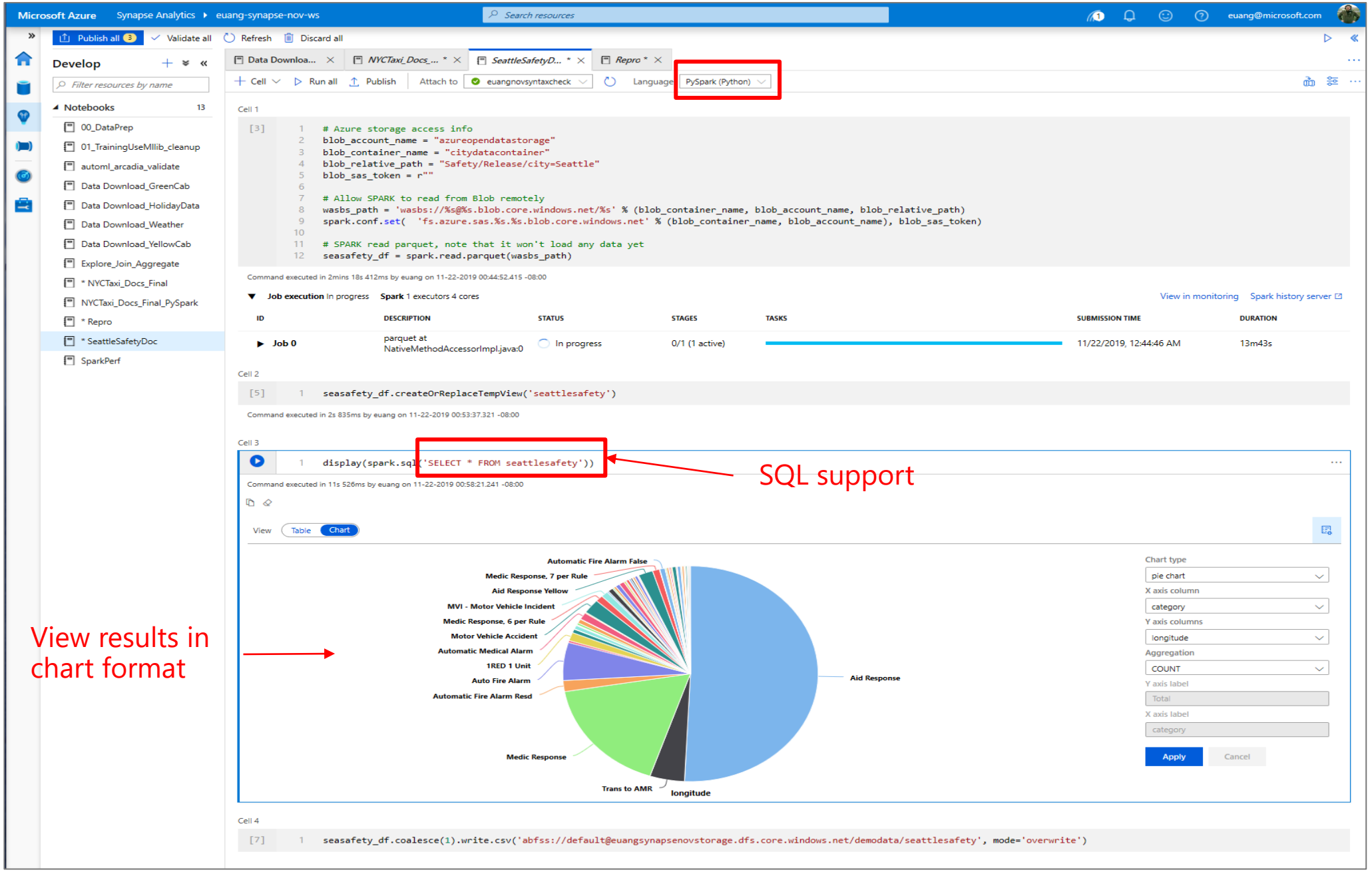
User Provisioned Workspace-Default Data Lake

Create Notebook on files in storage





View results in table format


Microsoft Azure
Synapse Analytics
euang-synapse-nov-ws
Search resources
euang@microsoft.com
Publish all 3
Validate all
Refresh
Discard all
Develop
Filter resources by name
Notebooks 13
00_DataPrep
01_TrainingUseMllib_cleanup
automl_arcadia_validate
Data Download_GreenCab
Data Download_HolidayData
Data Download_Weather
Data Download_YellowCab
Explore_Join_Aggregate
* NYCTaxi_Docs_Final
NYCTaxi_Docs_Final_PySpark
* Repro
* SeattleSafetyDoc
SparkPerf
Data Downloa...
NYCTaxi_Docs_... *
SeattleSafetyD... *
Repro *
Cell
Run all
Publish
Attach to
euangnovsyntaxcheck
Language
PySpark (Python)
Cell 1
[3]
# Azure storage access info
blob_account_name = "azureopendatastorage"
blob_container_name = "citydatacontainer"
blob_relative_path = "Safety/Release/city=Seattle"
blob_sas_token = r""
# Allow SPARK to read from Blob remotely
wasbs_path = 'wasbs://%s@%s.blob.core.windows.net/%s' % (blob_container_name, blob_account_name, blob_relative_path)
spark.conf.set('fs.azure.sas.%s.%s.blob.core.windows.net' % (blob_container_name, blob_account_name), blob_sas_token)
# SPARK read parquet, note that it won't load any data yet
seasafety_df = spark.read.parquet(wasbs_path)
Command executed in 2mins 18s 412ms by euang on 11-22-2019 00:44:52.415 -08:00
Job execution In progress
Spark 1 executors 4 cores
View in monitoring
Spark history server
ID
DESCRIPTION
STATUS
STAGES
TASKS
SUBMISSION TIME
DURATION
Job 0
parquet at NativeMethodAccessorImpl.java:0
In progress
0/1 (1 active)
11/22/2019, 12:44:46 AM
13m43s
Cell 2
[5]
seasafety_df.createOrReplaceTempView('seattlesafety')
Command executed in 2s 835ms by euang on 11-22-2019 00:53:37.321 -08:00
Cell 3
display(spark.sql('SELECT * FROM seattlesafety'))
Command executed in 11s 526ms by euang on 11-22-2019 00:58:21.241 -08:00
View
Table
Chart
Automatic Fire Alarm False
Medic Response, 7 per Rule
Aid Response Yellow
MVI - Motor Vehicle Incident
Medic Response, 6 per Rule
Motor Vehicle Accident
Automatic Medical Alarm
1RED 1 Unit
Auto Fire Alarm
Automatic Fire Alarm Resd
Medic Response
Trans to AMR
longitude
Aid Response
Chart type
pie chart
X axis column
category
Y axis columns
longitude
Aggregation
COUNT
Y axis label
Total
X axis label
category
Apply
Cancel
Cell 4
[7]
seasafety_df.coalesce(1).write.csv('abfss://default@euangsynapsenovstorage.dfs.core.windows.net/demodata/seattlesafety', mode='overwrite')
SQL support
View results in chart format

Microsoft Azure | Synapse Analytics | euang-synapse-nov-ws

Search resources

euang@microsoft.com

Publish all | Validate all | Refresh | Discard all

Develop

Filter resources by name

Notebooks 13

- 00_DataPrep
- 01_TrainingUseMllib_cleanup
- automl_arcadia_validate
- Data Download_GreenCab
- Data Download_HolidayData
- Data Download_Weather
- Data Download_YellowCab
- Explore_Join_Aggregate
- * NYCTaxi_Docs_Final
- NYCTaxi_Docs_Final_PySpark
- Repro
- SeattleSafetyDoc
- SparkPerf

Data Downloa... | NYCTaxi_Docs_... *

Cell | Run all | Publish | Attach to: Select Spark pool | Language: PySpark (Python)

```
# Creating a temp table allows easier manipulation during the session, they are not persisted between sessions,
# for that write the data to storage like above.
sampled_taxi_df.createOrReplaceTempView("nytaxi")
```

Exploratory Data Analysis

Look at the data and evaluate its suitablility for use in a model, do this via some basic charts focussed on tip values and relatoinships.

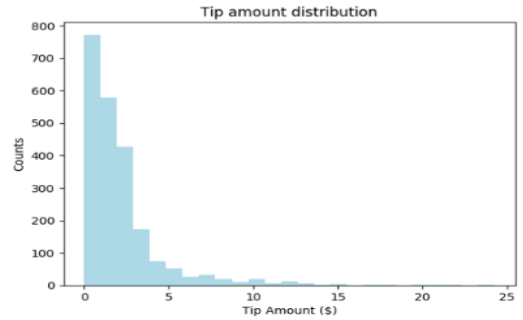
Cell 9

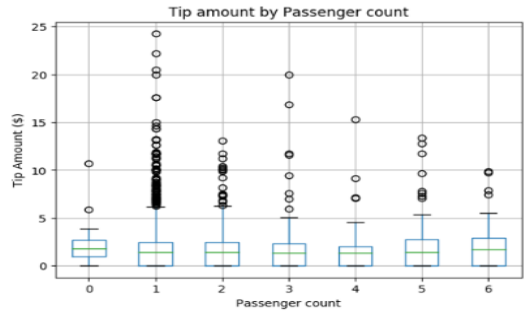
```
#The charting package needs a Pandas dataframe or numpy array do the conversion
sampled_taxi_pd_df = sampled_taxi_df.toPandas()

# Look at tips by amount count histogram
ax1 = sampled_taxi_pd_df['tipAmount'].plot(kind='hist', bins=25, facecolor='lightblue')
ax1.set_title('Tip amount distribution')
ax1.set_xlabel('Tip Amount ($)')
ax1.set_ylabel('Counts')
plt.suptitle('')
plt.show()

# How many passengers tip'd by various amounts
ax2 = sampled_taxi_pd_df.boxplot(column=['tipAmount'], by=['passengerCount'])
ax2.set_title('Tip amount by Passenger count')
ax2.set_xlabel('Passenger count')
ax2.set_ylabel('Tip Amount ($)')
plt.suptitle('')
plt.show()

# Look at the relationship between fare and tip amounts
ax = sampled_taxi_pd_df.plot(kind='scatter', x= 'fareAmount', y = 'tipAmount', c='blue', alpha = 0.10, s=2.5*(sampled_taxi_pd_df['passengerCount']))
ax.set_title('Tip amount by Fare amount')
ax.set_xlabel('Fare Amount ($)')
ax.set_ylabel('Tip Amount ($)')
plt.axis([-2, 80, -2, 20])
plt.suptitle('')
plt.show()
```





Tip amount by Fare amount

Exploratory data analysis with graphs – histogram, boxplot etc

Not Started | Configure session

Best practices

Serverless SQL Pools

- Co-locate storage and serverless SQL pools
- Consider Azure Storage throttling
- Prepare files for querying (CSV, JSON -> Parquet)
- Push wildcards to lower levels in the path
- Use appropriate data types and check inferred data types
- Use filename and filepath functions to target specific partitions
- Use PARSER_VERSION 2.0 to query CSV files
- Use CETAS to enhance query performance and joins
- Choose SAS credentials over Azure AD pass-through (for now)

CCI vs Heap

- Transformations using Heap tables are generally faster than CCI. This is because rows need to be assembled from column stores on read tables, and columnar compression is needed on targets.

- The wider the table, and the more text fields it contains, the faster Heap is over CCI.

- Use Heap tables at transformation layer, use CCI tables where appropriate at presentation layer

CCI Best Practice

- MAX data types not supported
- At least 1 million rows * 60 distributions * number of partitions
- At least 100k rows per batch, up to 1million
- Load using at least LARGERC or STATICRC60
  - Create a loading user
- Minimal UPDATE and DELETE (or REBUILD frequently)

Automatic statistics management – Dedicated SQL

Overview

Statistics are automatically created and maintained for dedicated SQL pool. Incoming queries are analyzed, and individual column statistics are generated on the columns that improve cardinality estimates to enhance query performance.

Statistics are automatically updated as data modifications occur in underlying tables. By default, these updates are synchronous but can be configured to be asynchronous.

Statistics are considered out of date when:

- There was a data change on an empty table
- The number of rows in the table at time of statistics creation was 500 or less, and more than 500 rows have been updated
- The number of rows in the table at time of statistics creation was more than 500, and more than 500 + 20% of rows have been updated

```
-- Turn on/off auto-create statistics settings
ALTER DATABASE {database_name}
SET AUTO_CREATE_STATISTICS { ON | OFF }

-- Turn on/off auto-update statistics settings
ALTER DATABASE {database_name}
SET AUTO_UPDATE_STATISTICS { ON | OFF }

-- Configure synchronous/asynchronous update
ALTER DATABASE {database_name}
SET AUTO_UPDATE_STATISTICS_ASYNC { ON | OFF }

-- Check statistics settings for a database
SELECT      is_auto_create_stats_on,
            is_auto_update_stats_on,
            is_auto_update_stats_async_on
FROM        sys.databases
```

Statistics (serverless SQL)

- Automatic creation available only for Parquet and CSV support
- Same goes for recreation of statistics
- Only single-column statistics are currently supported
- CSV sampling not supported yet (only FULLSCAN)

CTAS vs Insert / Update / Delete / Merge

- Prefer CTAS when you update or delete more than 10% of rows

- Prefer CTAS when you are updating or deleting a clustered Columnstore index, and do not have time for an offline rebuild

UPDATE FROM and DELETE FROM

- Azure Synapse Analytics does not currently support (*) joins in UPDATE FROM and DELETE FROM queries.

- Implement the join as a temporary / transient table, then UPDATE / DELETE from that table

- (*) Coming soon

Simple is better than clever

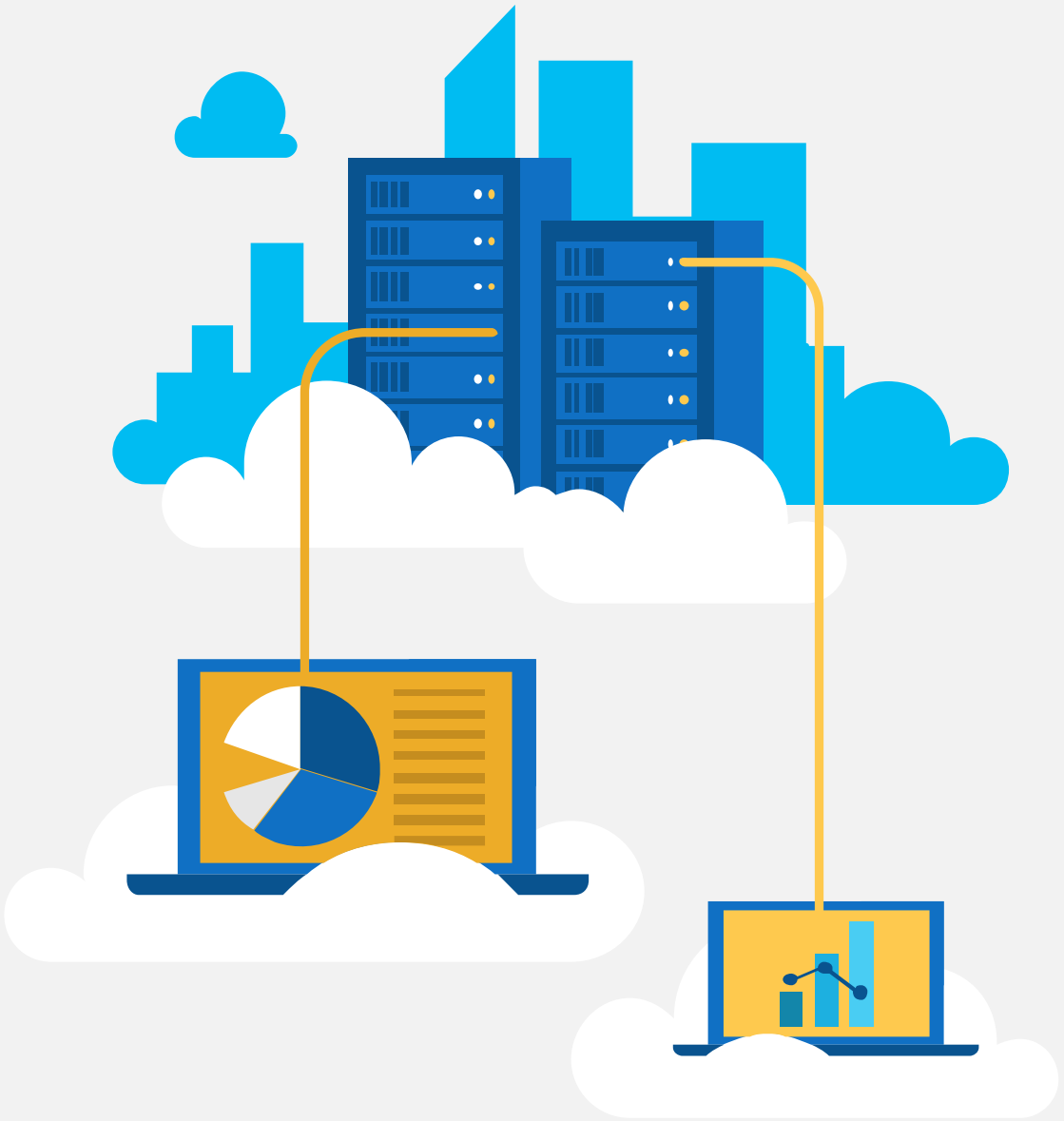
- Persist standard columns early, to avoid calculations and functions in WHERE clause
- Unroll CTEs and JOIN sub-selects to transient / temporary tables to manage distribution

- Simple queries are easier to tune and debug

Pop Quiz 2

What is the optimal size for a rowgroup in columnstore format in a Synapse SQL Pool?

A)	B)	C)
99,999	60,000,000	1,048,576



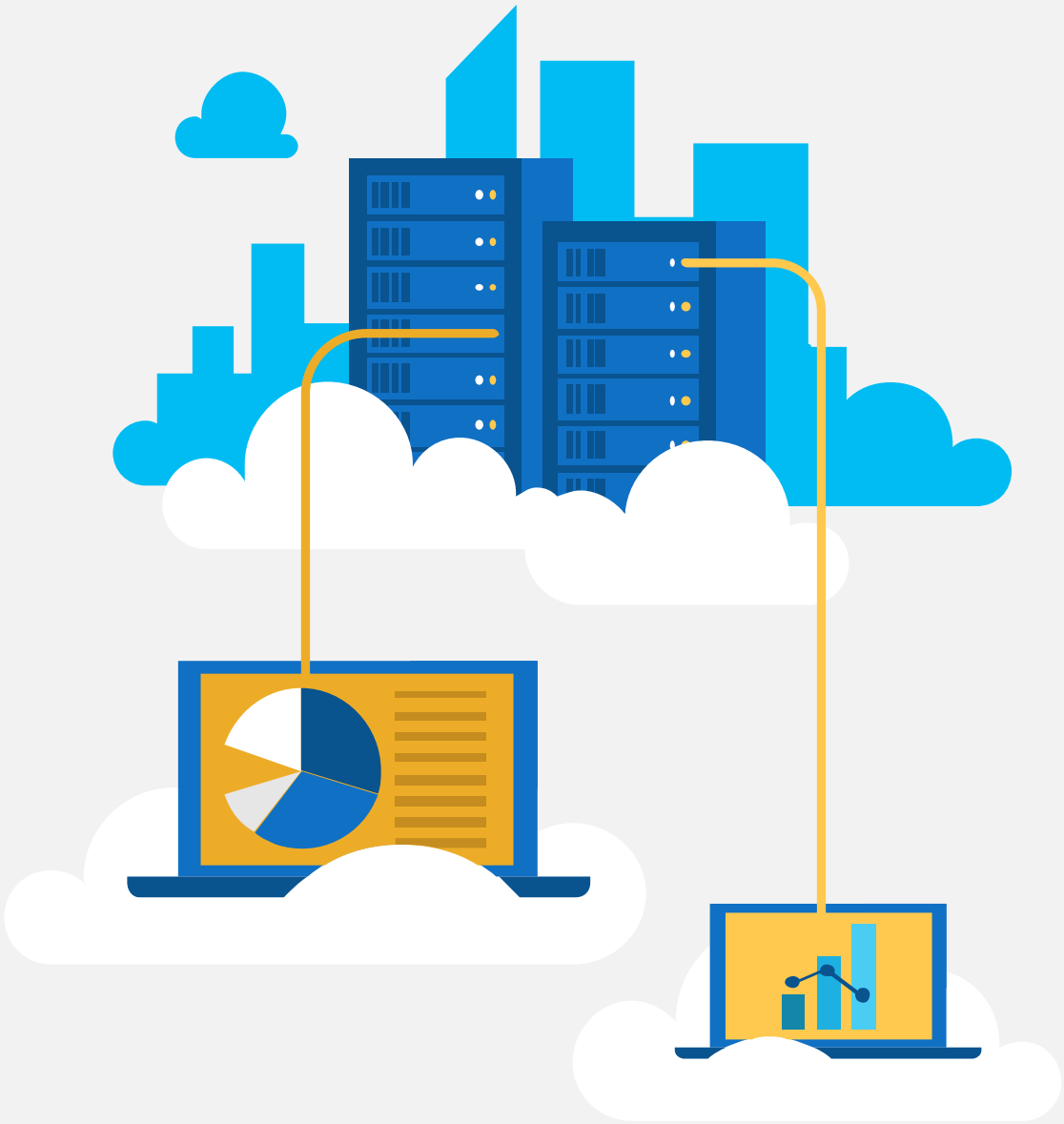
Pop Quiz 2

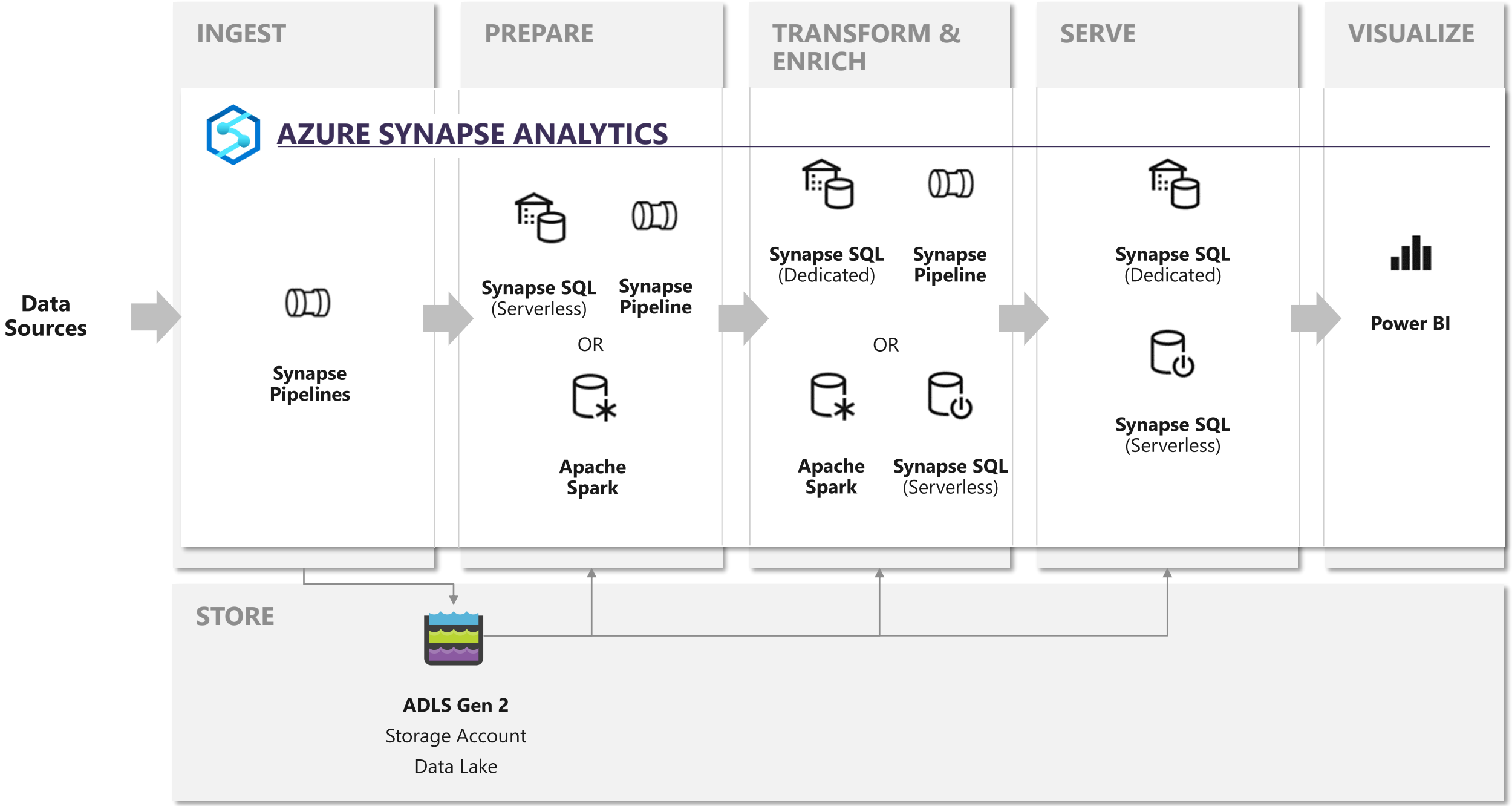
What is the optimal size for a rowgroup in columnstore format in a Synapse SQL Pool?

A) 99,999

B) 60,000,000

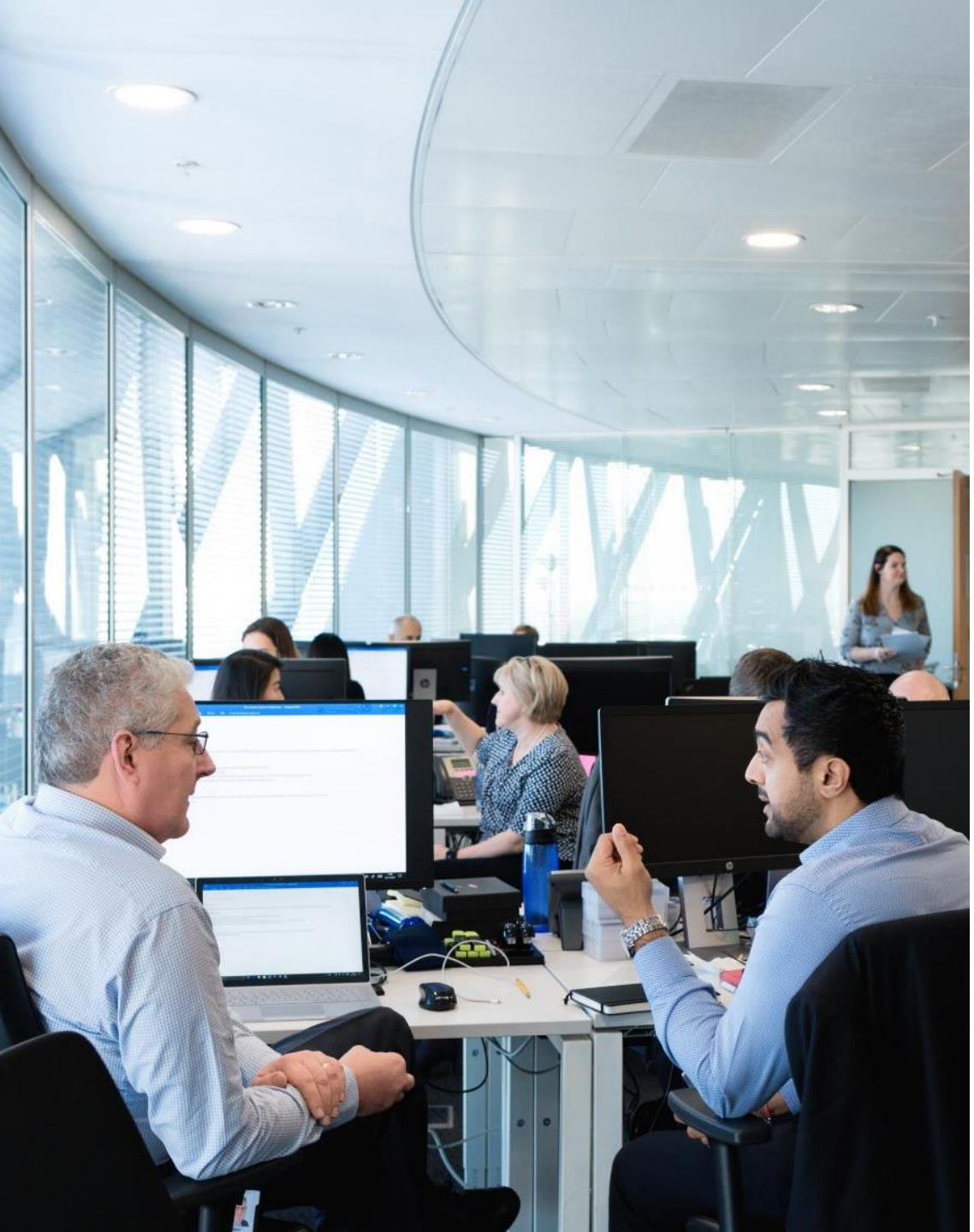
C) 1,048,576




INGEST
PREPARE
TRANSFORM & ENRICH
SERVE
VISUALIZE
AZURE SYNAPSE ANALYTICS
Data Sources
Synapse Pipelines
Synapse SQL (Serverless)
Synapse Pipeline
OR
Apache Spark
Synapse SQL (Dedicated)
Synapse Pipeline
OR
Apache Spark
Synapse SQL (Serverless)
Synapse SQL (Dedicated)
Synapse SQL (Serverless)
Power BI
STORE
ADLS Gen 2
Storage Account
Data Lake

Microsoft Azure Solutions Business

Thank you



Full Group Activity:
Data Engineering Discussion

Objective: As a result of participating in this activity, you will better be able to decide on which Azure Synapse Analytics component to use for specific data engineering scenarios.

What you will be doing: The facilitator will be posting questions to the entire group using an interactive tool called Mentimeter. The answers you post using Mentimeter will then drive the Data Engineering Discussion.

Total Activity Time: 30 minutes

Microsoft Azure
Solutions Business

Mentimeter Poll

Scan QR Code

or

Go to www.menti.com and use code

6142 6491

Closing

Thank you for your participation in today's Azure Synapse Technical Boot Camp!

TODAY

We learned:

- ✓ Best practices for rapid and reliable data ingestion into a Data Warehouse
- ✓ A well architected data lake is built upon scalable and secure partitioning structure
- ✓ Established best practices for data transformation within the data engineering pipeline in order to efficiently go from raw to structured data for analysis

TOMORROW

We will learn to:

- Implement optimization strategizes for data warehouse using SQL
- Apply security concepts to a customer scenario

Microsoft Azure Solutions Business