

Lab 6 - Transform data with Azure Data Factory or Azure Synapse Pipelines

This lab teaches you 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.

After completing this lab, you 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

Lab setup and pre-requisites

Before starting this lab, you must complete **Lab 5: Ingest and load data into the Data Warehouse**.

This lab uses the dedicated SQL pool you created in the previous lab. You should have paused the SQL pool at the end of the previous lab, so resume it by following these instructions:

1. Open Synapse Studio (<https://web.azuresynthesize.net/>).
2. Select the **Manage** hub.
3. Select **SQL pools** in the left-hand menu. If the **SQLPool01** dedicated SQL pool is paused, hover over its name and select **▷**.

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

4. When prompted, select **Resume**. It will take a minute or two to resume the pool.
5. Continue to the next exercise while the dedicated SQL pool resumes.

Important: Once started, a dedicated SQL pool consumes credits in your Azure subscription until it is paused. If you take a break from this lab, or decide not to complete it; follow the instructions at the end of the lab to pause your SQL pool!

Exercise 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.

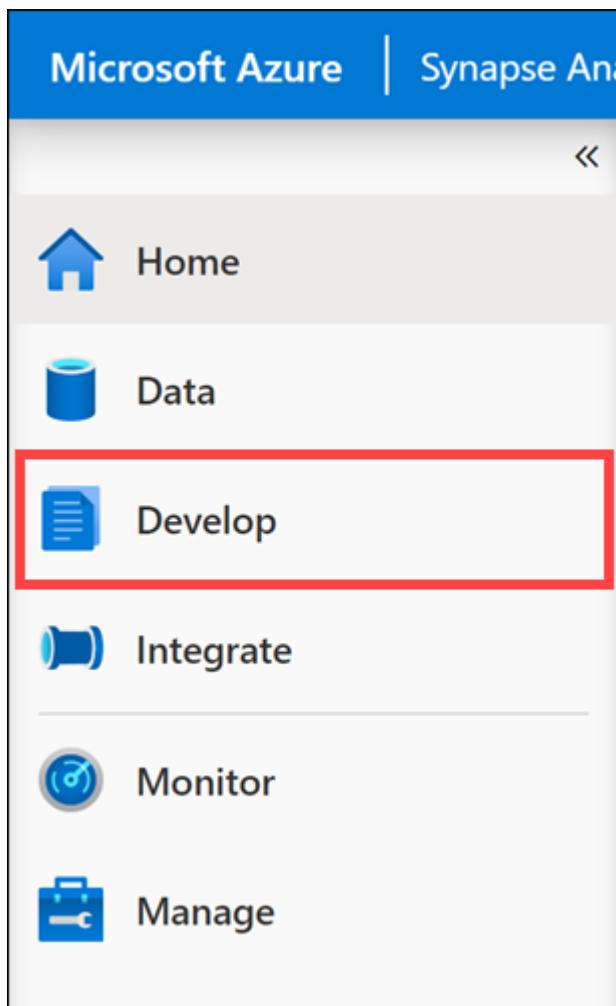
Additional benefits

- 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

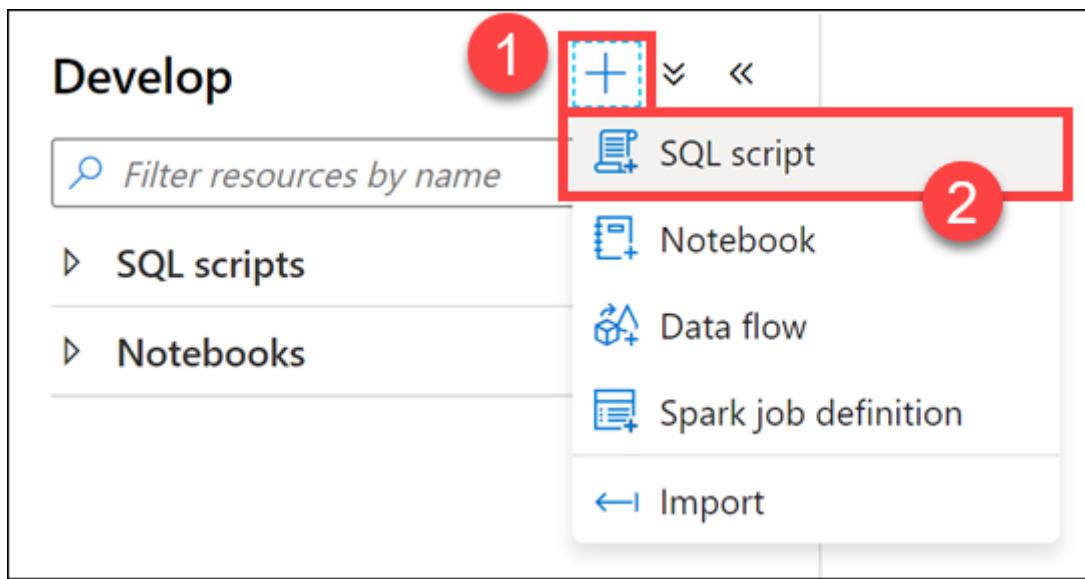
Task 1: Create SQL table

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. In Synapse Analytics Studio, navigate to the **Develop** hub.



2. In the + menu, select **SQL script**.



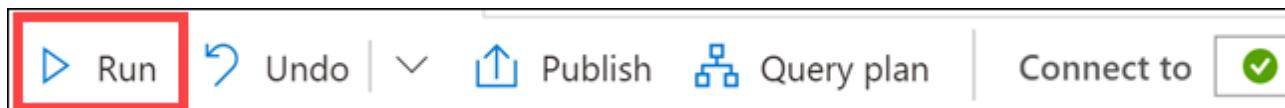
3. In the toolbar menu, connect to the **SQLPool01** database.



4. In the query window, replace the script with the following code 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
)
```

5. Select **Run** on the toolbar menu to run the script (you may need to wait for the SQL pool to resume).



6. 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
)
```

7. Run the script to create the table.

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.

The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, there's a vertical navigation menu with icons and labels: Home (house), Data (cylinder), Develop (document), Integrate (camera), Monitor (gauge), and Manage (briefcase). The 'Manage' option is highlighted with a red rectangular box. At the top, the title bar says 'Microsoft Azure | Synapse Analytics'.

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 a 'Linked services' option highlighted with a red box (Step 1). In the center, there's a 'New' button with a red box (Step 2). On the right, a grid of service icons is shown. The 'Azure Cosmos DB (SQL API)' icon is highlighted with a red box (Step 3). At the bottom right of the dialog, there's a 'Continue' button with a red box (Step 4).

3. Name the linked service **asacosmosdb01**, and then select the **asacosmosdbxxxxxxxx** Cosmos DB account name and the **CustomerProfile** database. Then select **Test connection** to ensure success, before clicking **Create**.

New linked service (Azure Cosmos DB (SQL API))

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

Name * 1
asacosmosdb01

Description

Connect via integration runtime * (i)
AutoResolveIntegrationRuntime

Connection string Azure Key Vault (i)

Account selection method (i)
 From Azure subscription Enter manually

Azure subscription (i)
Select all

Azure Cosmos DB account name * 2
asacosmosdbinaday84

Database name * 3
CustomerProfile

Additional connection properties
+ New

Annotations

Connection successful 4
 Connection successful Cancel

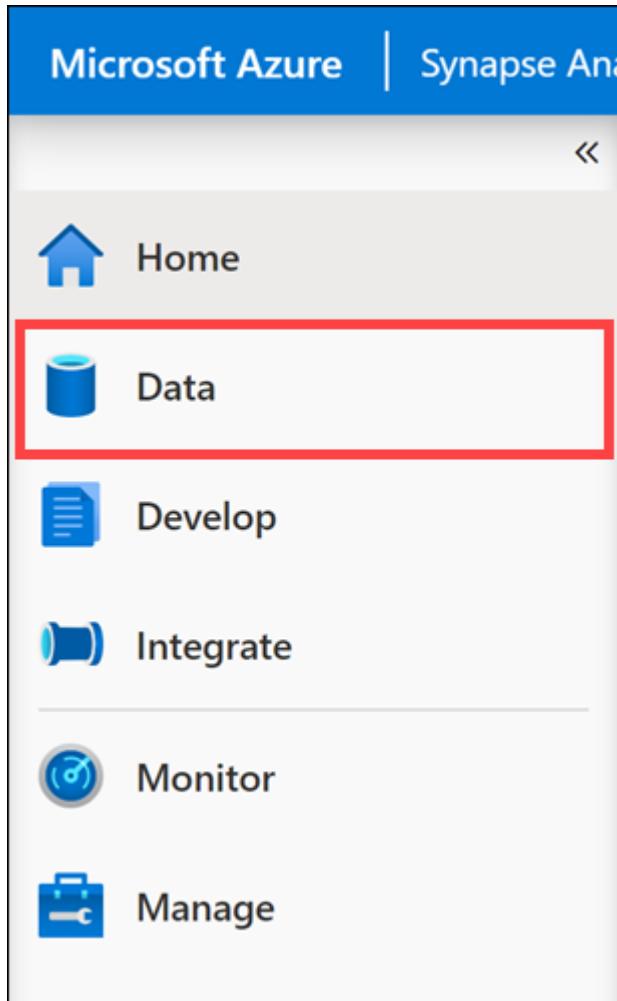
Task 3: Create data sets

User profile data comes from two different data sources, which we will create now. The customer profile data from an e-commerce system that provides top product purchases for each visitor of the site (customer) over

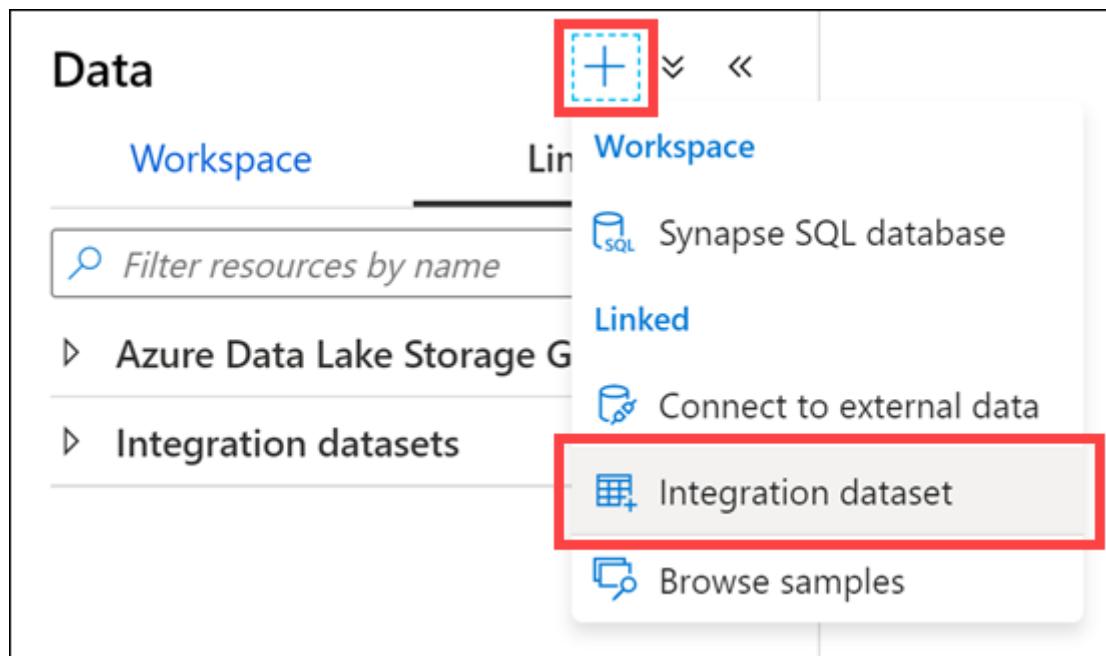
the past 12 months is stored within JSON files in the data lake. User profile data containing, among other things, product preferences and product reviews is stored as JSON documents in Cosmos DB.

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.

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



2. In the + menu, select **Integration dataset** to create a new dataset.



3. Select **Azure Cosmos DB (SQL API)**, then click **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

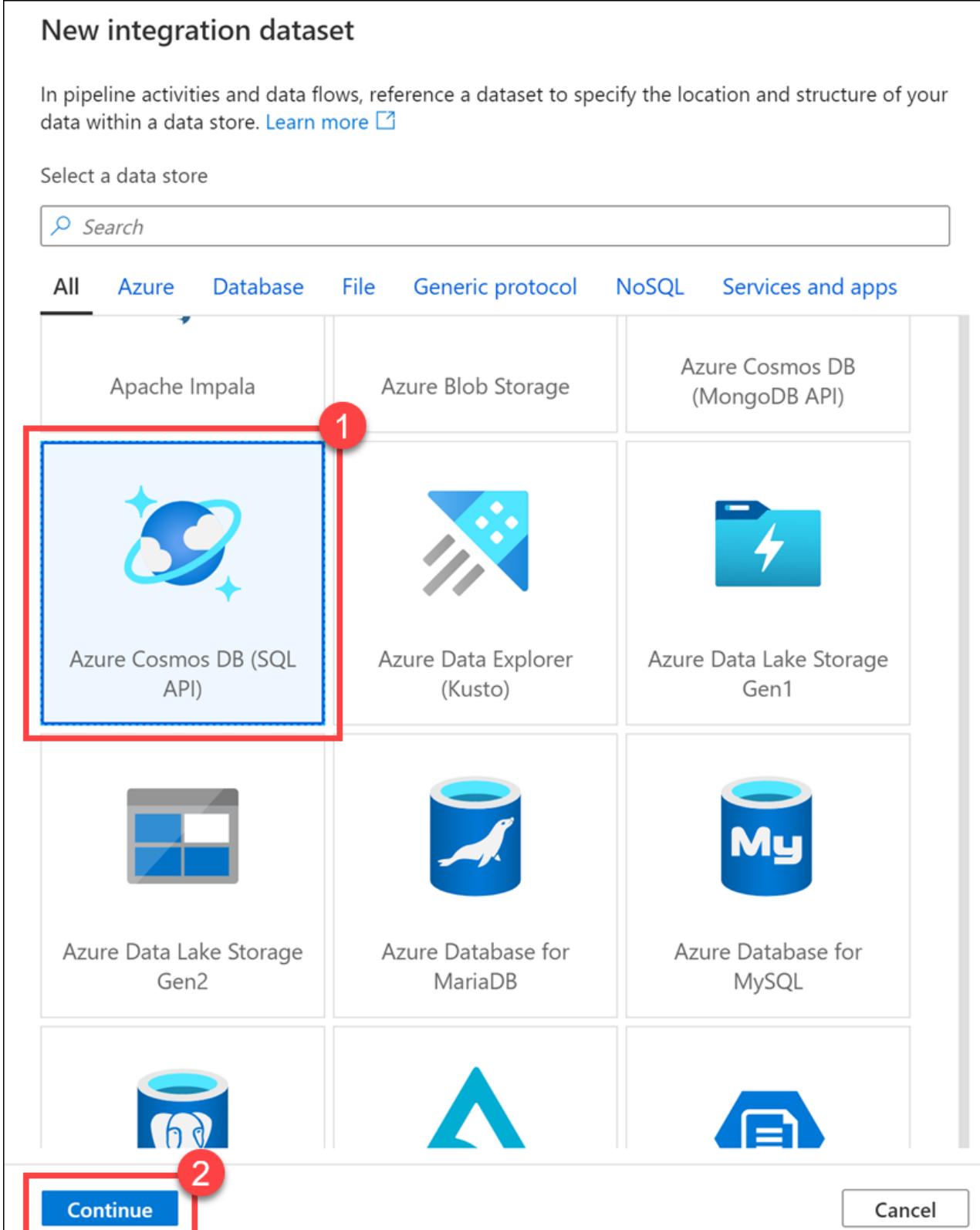
Search

All Azure Database File Generic protocol NoSQL Services and apps

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
 Azure Cosmos DB (MongoDB API)		

1 2

Continue **Cancel**



4. Configure the dataset as follows, then select **OK**:

- **Name:** Enter `asal400_customerprofile_cosmosdb`.
- **Linked service:** Select `asacosmosdb01`.
- **Collection:** Select `OnlineUserProfile01`.

Set properties

Choose a name for your dataset. This name can be updated at any time until it is published.

Name asal400_customerprofile_cosmosdb 1

Linked service * asacosmosdb01 2

Connect via integration runtime * AutoResolveIntegrationRuntime 3

Collection OnlineUserProfile01 4

From connection/store None

OK Back Cancel

5. After creating the dataset, select **Preview data** under its **Connection** tab.

CosmosDB Collection (SQL API)
asal400_customerprofile_cosmosdb

Connection Schema Parameters

Linked service * asacosmosdb01 5 Test connection Edit New

Integration runtime * AutoResolveIntegrationRuntime 6

Collection OnlineUserProfile01 7 Refresh Preview data 8

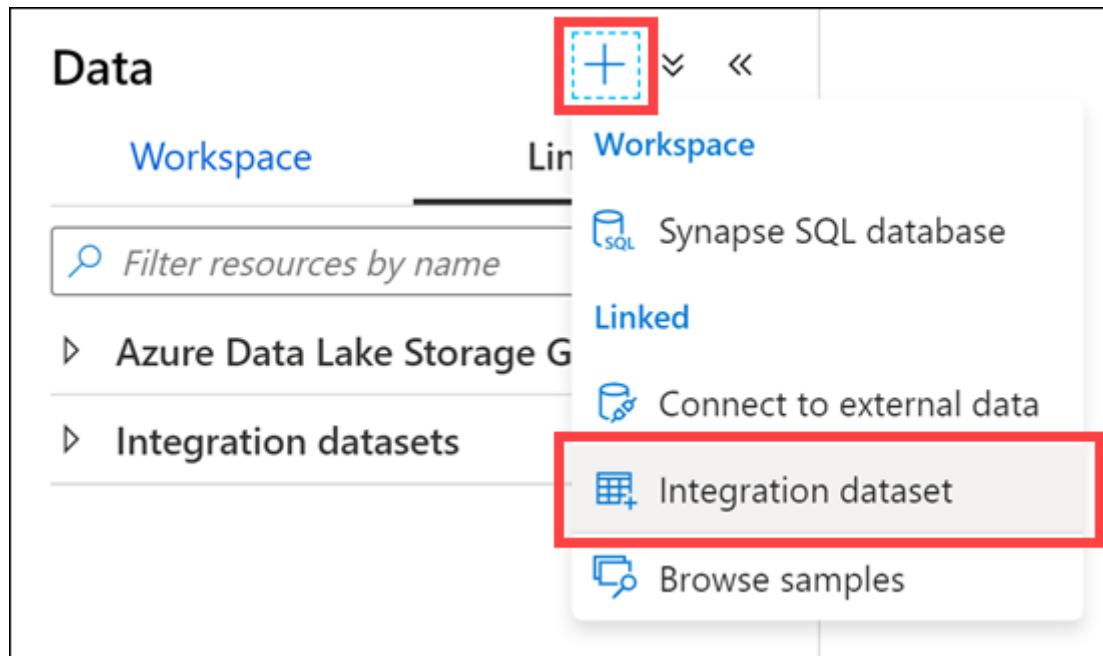
6. Preview data queries the selected Azure Cosmos DB collection and returns a sample of the documents within. The documents are stored in JSON format and include fields for **userId**, **cartId**, **preferredProducts** (an array of product IDs that may be empty), and **productReviews** (an array of written product reviews that may be empty).



The screenshot shows the 'Preview data' window in the Azure Data Explorer. It displays a JSON object representing a user profile. The object includes fields for userId, cartId, preferredProducts (a list of product IDs), and productReviews (a list of review objects). Each review contains productId, reviewText, and reviewDate.

```
[{"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": "2019-06-08T22:42:56.005Z"}, {"productId": 1360, "reviewText": "i use it daily when i'm in my courthouse.", "reviewDate": "2017-04-29T08:43:26.677Z"}]}
```

7. Close the preview. Then on the **Data** hub, in the + menu, select **Integration dataset** to create the second source data dataset we need.



8. Select **Azure Data Lake Storage Gen2**, then click **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

All Azure Database File Generic protocol NoSQL Services and apps

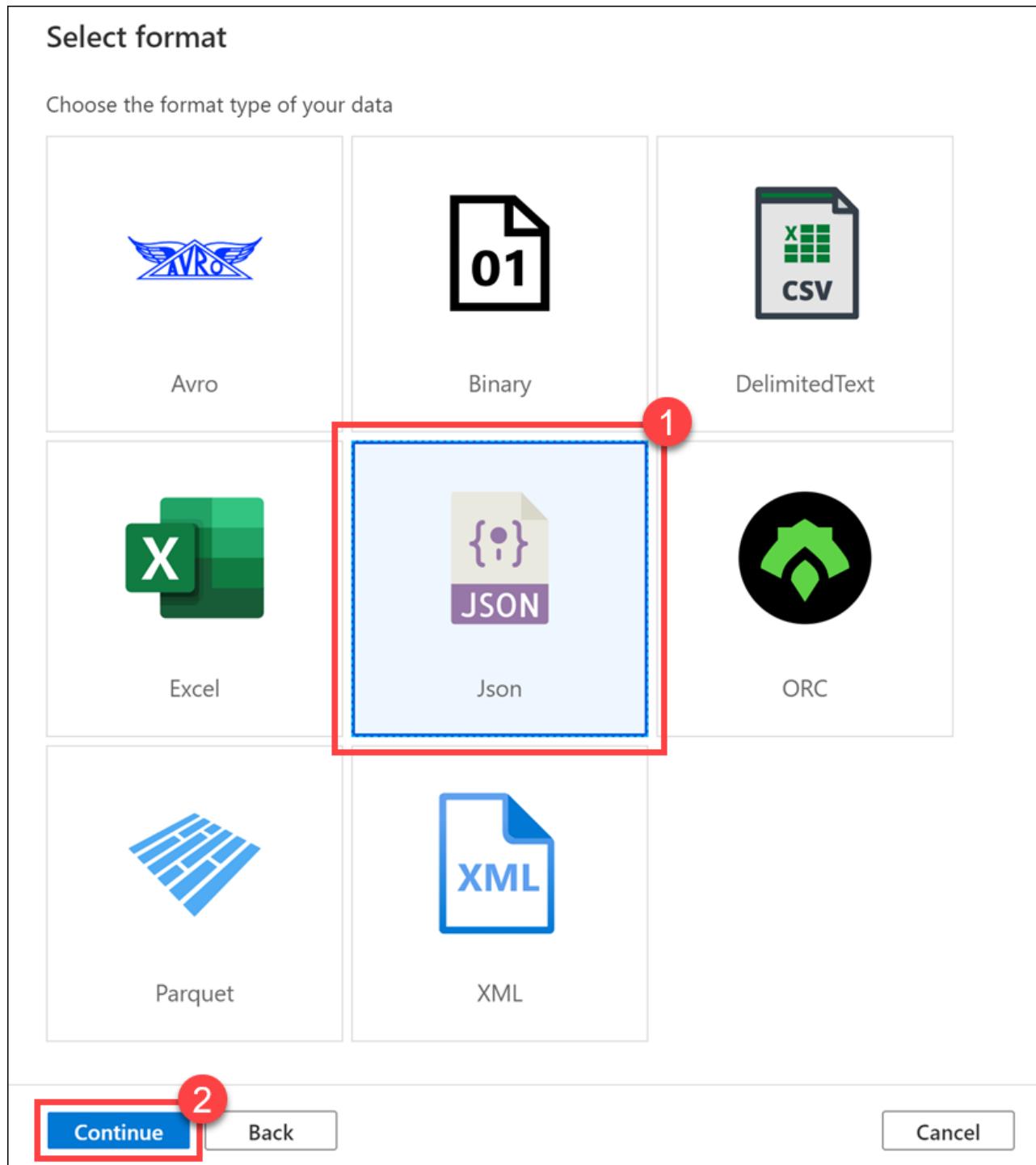
Azure Cosmos DB (SQL API)	Azure Data Explorer (Kusto)	Azure Data Lake Storage Gen1
Azure Data Lake Storage Gen2	Azure Database for MariaDB	Azure Database for MySQL
Azure Database for PostgreSQL	Azure Databricks Delta Lake	Azure File Storage

1

2

Continue **Cancel**

9. Select the **JSON** format, then select **Continue**.



10. Configure the dataset as follows, then select **OK**:

- **Name:** Enter `asal400_ecommerce_userprofiles_source`.
- **Linked service:** Select the `asadatalakxxxxxxxx` linked service.
- **File path:** Browse to the `wwi-02/online-user-profiles-02` path.
- **Import schema:** Select **From connection/store**.

Set properties

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

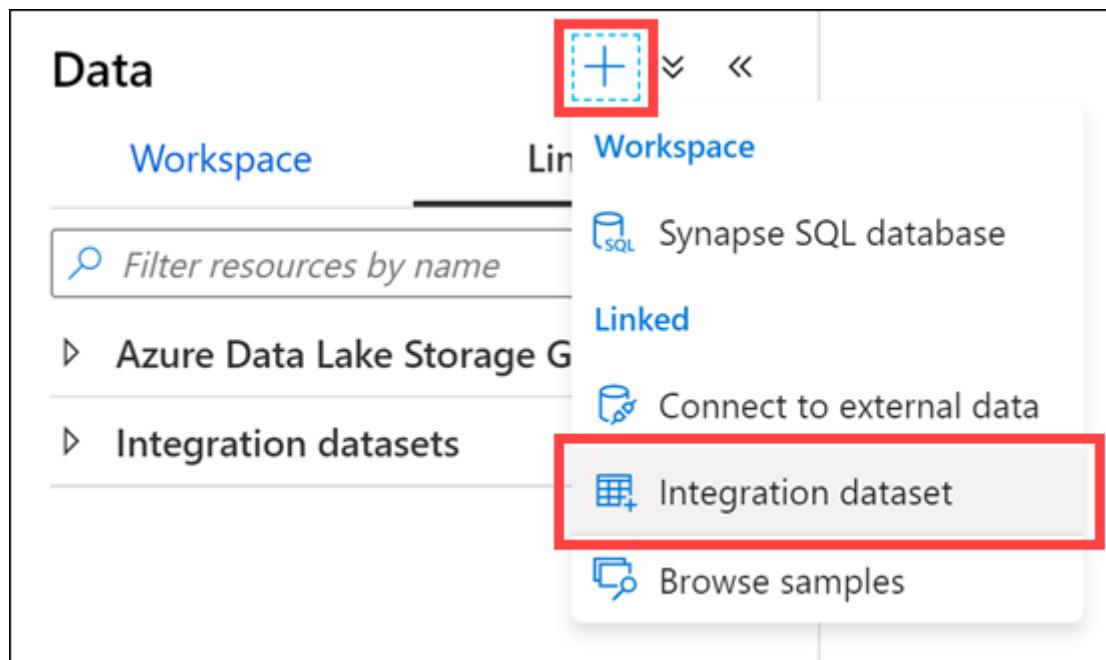
3 File path: wwi-02 / online-user-profiles-02 / File

4 Import schema: From connection/store (selected)

5 OK

Cancel

11. On the **Data** hub, in the + menu, select **Integration dataset** to create a third dataset that references the destination table for campaign analytics.



12. 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

All Azure Database File Generic protocol NoSQL Services and apps

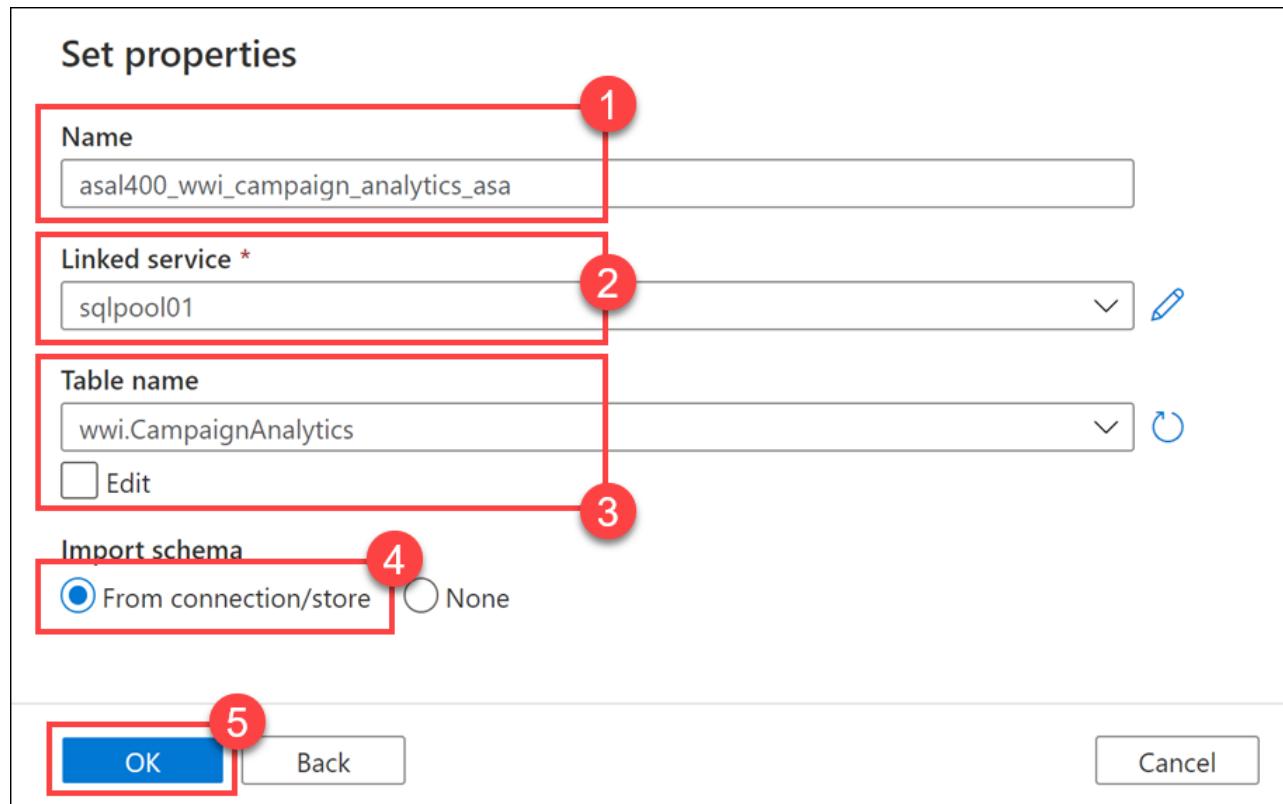
 Azure SQL Database	 Azure SQL Database Managed Instance	 Azure Search
 Azure Synapse Analytics	 Azure Synapse dedicated SQL pool	 Azure Table Storage
 Cassandra	 Common Data Service for Apps	 Concur (Preview)

1 2

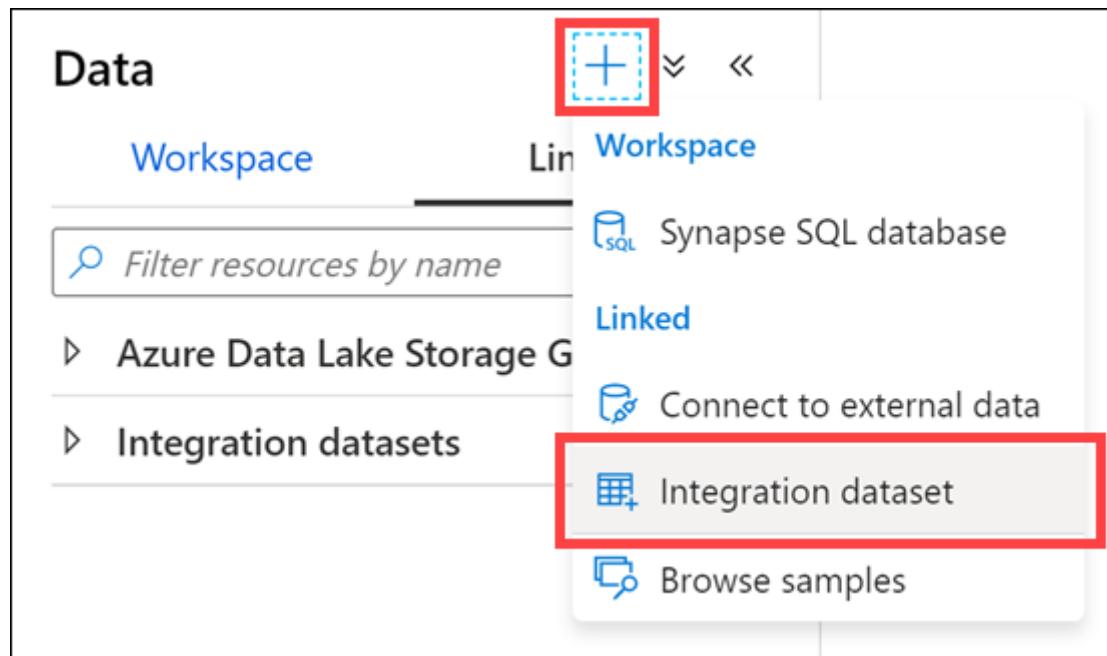
Cancel

13. Configure the dataset as follows, then select **OK**:

- **Name:** Enter `asal400_wwi_campaign_analytics_asa`.
- **Linked service:** Select the `SqlPool01`.
- **Table name:** Select `wwi.CampaignAnalytics`.
- **Import schema:** Select `From connection/store`.



14. On the **Data** hub, in the + menu, select **Integration dataset** to create a fourth dataset that references the destination table for top product purchases.



15. 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

All Azure Database File Generic protocol NoSQL Services and apps

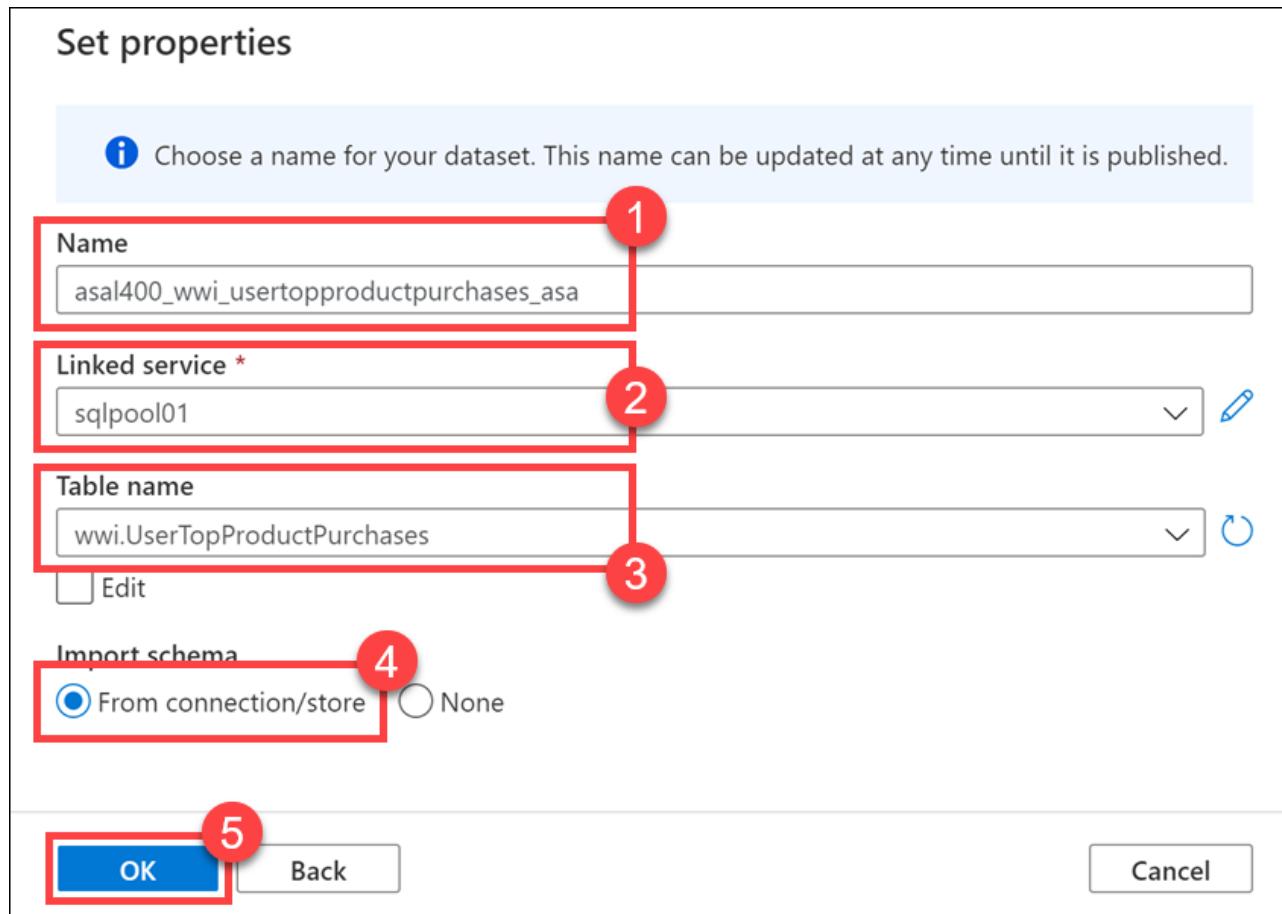
 Azure SQL Database	 Azure SQL Database Managed Instance	 Azure Search
 Azure Synapse Analytics	 Azure Synapse dedicated SQL pool	 Azure Table Storage
 Cassandra	 Common Data Service for Apps	 Concur (Preview)

1 2

Continue Cancel

16. Configure the dataset as follows, then select **OK**:

- **Name:** Enter `asal400_wwi_usertopproductpurchases_asa`.
- **Linked service:** Select the **SqlPool01**.
- **Table name:** Select **wwi.UserTopProductPurchases**.
- **Import schema:** Select **From connection/store**.



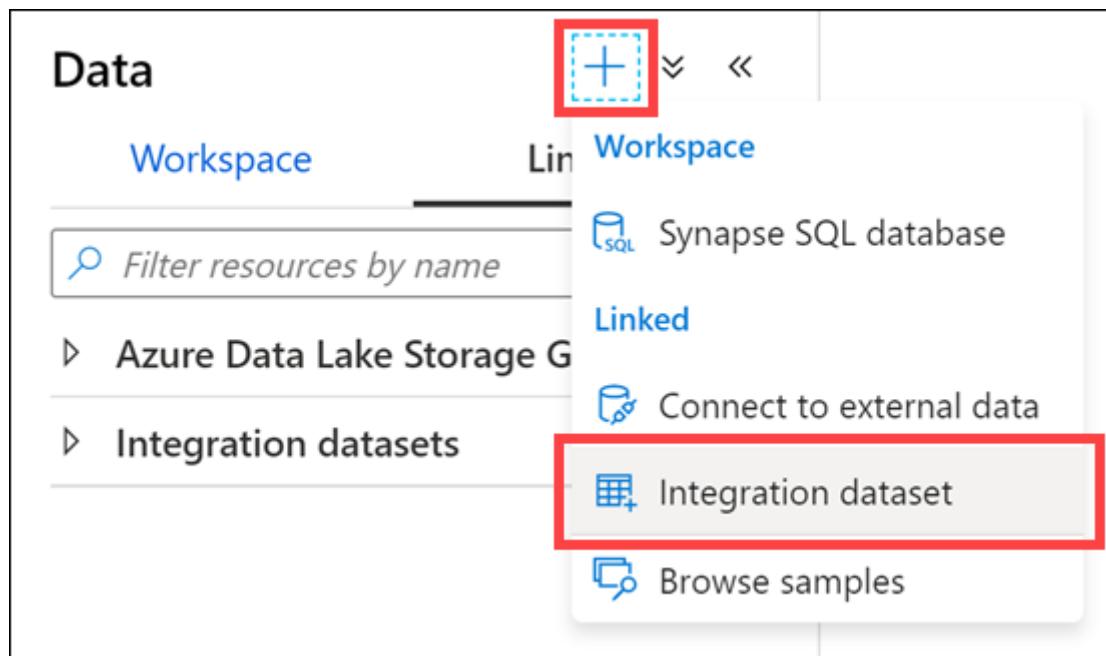
Task 4: Create campaign analytics dataset

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	DÃ©cor	Spring into Summer	\$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	DÃ©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 Summer	\$17\	637 \$6\	876		
Far West	US	Books	EnjoyTheMoment; \$11\	247 \$10\	339	San Diego	California	
South America	Mexico	DÃ©cor	Be Unique	\$8\	284 \$9\	840		

Issues include invalid characters in the revenue currency data, and misaligned columns.

- On the **Data** hub, in the + menu, select **Integration dataset** to create a new dataset.



2. Select **Azure Data Lake Storage Gen2**, 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

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

1

2

Continue **Cancel**

3. Select the **DelimitedText** format, then select **Continue**.

Select format

Choose the format type of your data

 Avro	 Binary	 DelimitedText 1
 Excel	 Json	 ORC
 Parquet	 XML	

2

Continue Back Cancel

4. Configure the dataset as follows, then select **OK**:

- **Name:** Enter `asal400_campaign_analytics_source`.
- **Linked service:** Select the `asadatalakxxxxxxxx` linked service.
- **File path:** Browse to `wwi-02/campaign-analytics/campaignanalytics.csv`.
- **First row as header:** Leave unchecked (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**.

Set properties

Choose a name for your dataset. This name can be updated at any time until it is published.

Name: asal400_campaign_analytics_source

Linked service: asadatalake264973

File path: wwi-02 / campaign-analytics / campaignanalytics.csv

First row as header:

Import schema: From connection/store From sample file None

OK Back Cancel

5. After creating the dataset, on its **Connection** tab, review the default settings. They should match the following configuration:

- **Compression type:** None.
- **Column delimiter:** Comma (,)
- **Row delimiter:** Default (\r\n, or \r\n)
- **Encoding:** Default(UTF-8)
- **Escape character:** Backslash (\)
- **Quote character:** Double quote (")
- **First row as header:** *Unchecked*
- **Null value:** *Empty*

The screenshot shows the 'Connection' pane in the Azure Data Factory interface. It includes fields for 'Linked service', 'File path', 'Compression type', 'Column delimiter', 'Row delimiter', 'Encoding', 'Escape character', 'Quote character', and 'First row as header'. The 'File path' field is set to 'wwi-02 / campaign-analytics / campaignanalytics.csv'. The 'Preview data' button is highlighted with a red box.

6. Select **Preview data** (close the **Properties** pane if it is in the way).

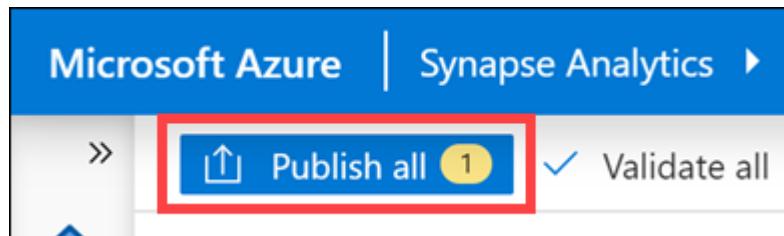
The preview displays a sample of the CSV file. You can see some of the issues shown at the beginning of this task. Notice that since we are not setting the first row as the header, the header columns appear as the first row. Also, notice that the city and state values do not appear. This is because of 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.

Preview data

Linked service: asadatalake264973
Object: campaignanalytics.csv

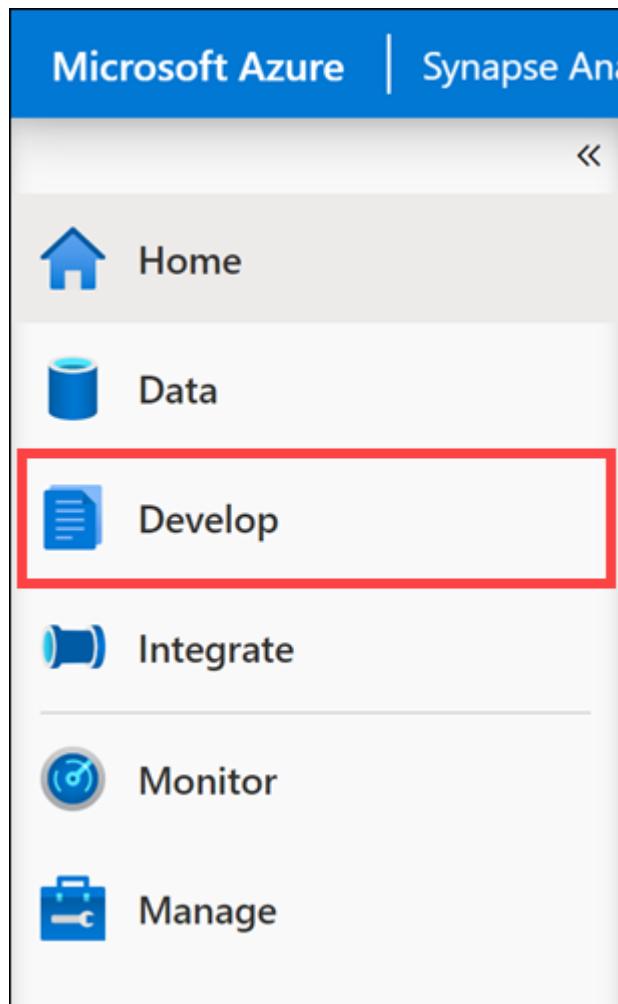
Prop_0	Prop_1	Prop_2	Prop_3	Prop_4	Prop_5	Prop_6	Prop_7
Region	Country	Product_Category	Campaign_Name	Revenue	Revenue_Target	City	State
Europe	Germany	Apparel and Footwear	Fun with Colors	\$14\	865.00	\$15\	960.00
Far West	US	Books	EnjoyTheMoment; BeUnique; TailoredForYou	\$14\	992.00	\$15\	699.00
Europe	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; TailoredForYou	\$11\	110.00	\$17\	200.00

7. Close the preview, and then select **Publish all** and click **Publish** to save your new resources.

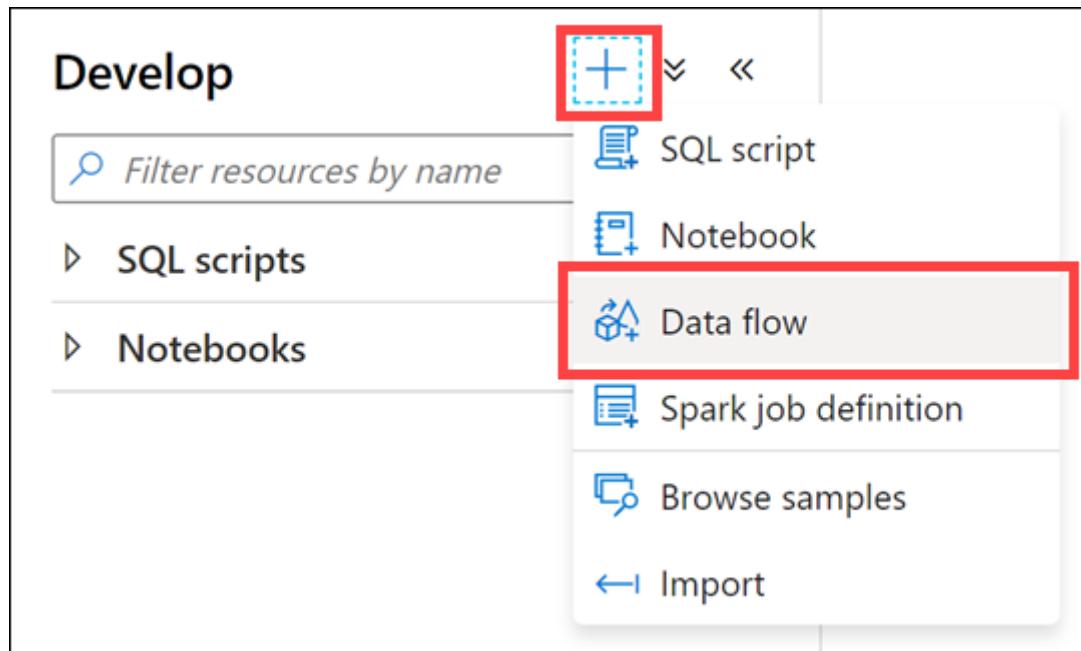


Task 5: Create campaign analytics data flow

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



2. In the + menu, select **Data flow** to create a new data flow (if a tip is displayed, close it.)



3. In the **General** settings of the **Properties** blade of the new data flow, change the **Name** to **asal400_lab2_writecampaignanalyticstoasa**.

Properties

General

Name *
asal400_lab2_writecampaignanalyticstoasa

Description

A screenshot of the 'Properties' blade for a new data flow. It shows the 'General' tab selected. There is a tip message: 'Choose a name for your data flow. This name can be updated at any time until it is published.' Below it, the 'Name' field is filled with 'asal400_lab2_writecampaignanalyticstoasa'. There is also a 'Description' field which is currently empty.

4. Select **Add Source** on the data flow canvas (again, if a tip is displayed, close it.)

Validate Data flow debug

Add Source

A screenshot of the data flow canvas. At the top, there are two checkboxes: 'Validate' (checked) and 'Data flow debug'. Below them is a large rectangular area with a dashed red border, which is labeled 'Add Source' in the center. The rest of the canvas is empty.

5. Under **Source settings**, configure the following:

- **Output stream name:** Enter **CampaignAnalytics**.
- **Source type:** Select **Integration 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 * [Learn more](#)

Source type *

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

Options

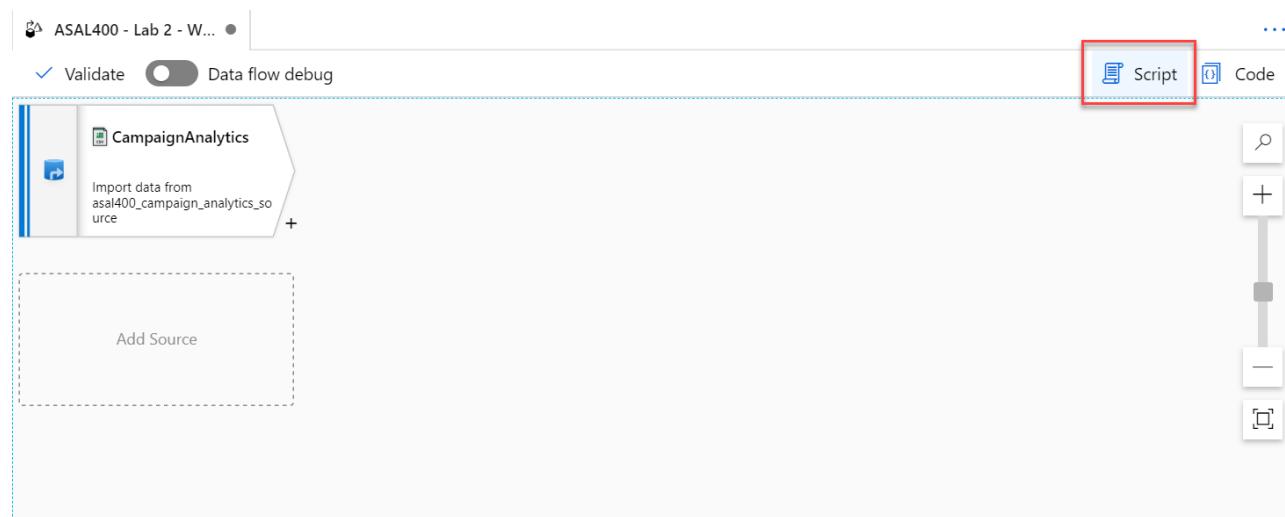
- Allow schema drift i
- Infer drifted column types i
- Validate schema i

Skip line count

Sampling * i Enable Disable

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, and to minimize resource consumption in the lab environment, we will bypass these features.

6. The data source has a schema we need to set. To do this, select **Script** above the design canvas.



The screenshot shows the Power BI Data Flow designer interface. At the top, there are several buttons: 'ASAL400 - Lab 2 - W...', 'Validate' (with a checkmark), 'Data flow debug' (with a toggle switch), and three tabs: 'Script' (which is highlighted with a red box), 'Code', and 'Projection'. Below these, the main workspace displays a single data source named 'CampaignAnalytics' with the description 'Import data from asal400_campaign_analytics_source'. A dashed box labeled 'Add Source' is visible. On the right side, there are standard canvas navigation controls: a search icon, a plus sign for adding new components, a minus sign for removing them, and a close/collapse icon.

7. Replace the script with the following to provide the column mappings, 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,
skipLines: 1) ~> CampaignAnalytics
```

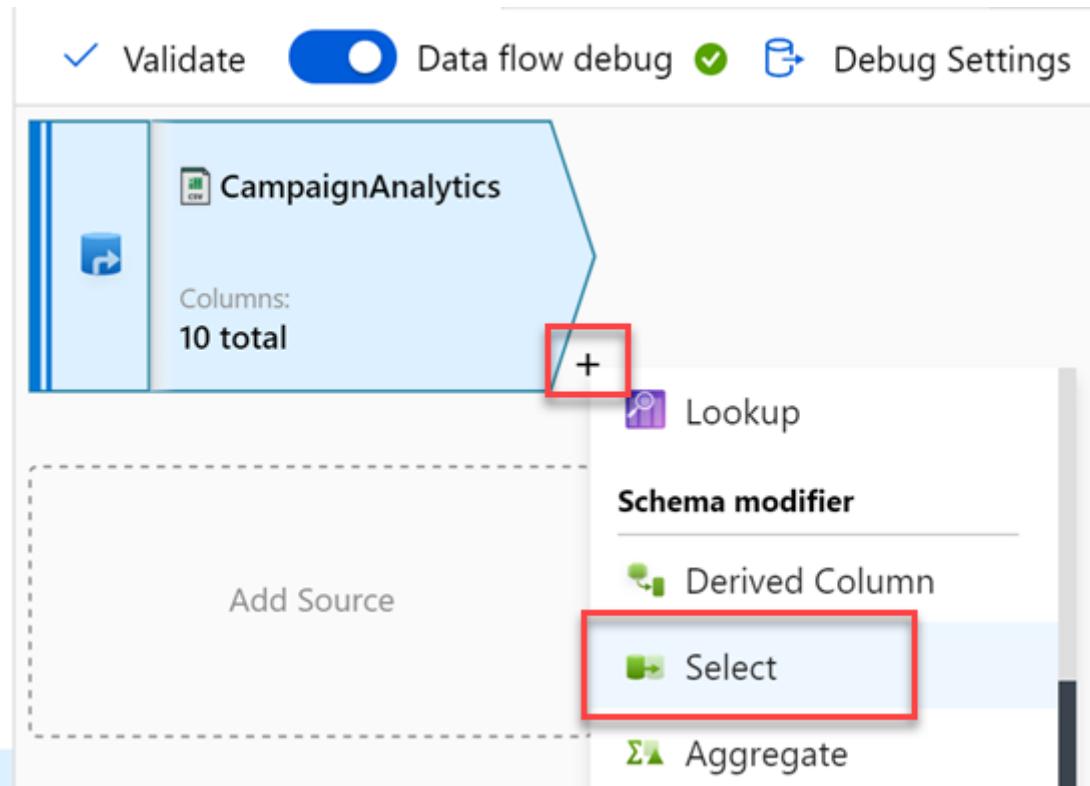
8. Select the **CampaignAnalytics** data source, then select **Projection**. The projection should display the following schema:

The screenshot shows the Data Preview tab of a pipeline configuration. The top navigation bar includes tabs for Source settings, Source options, **Projection**, Optimize, Inspect, and Data preview. The **Projection** tab is active.

The projection schema is displayed in a table format:

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** step, then select the **Select** schema modifier.



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 unselected, 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 ↗

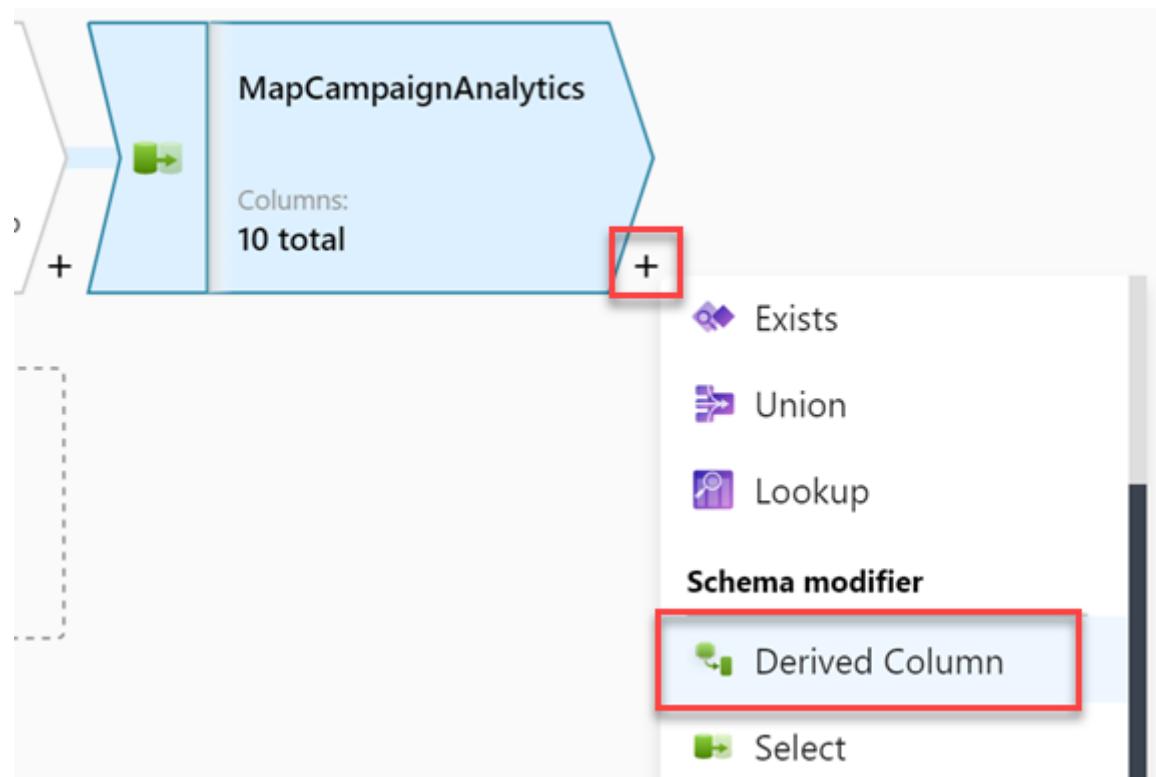
Incoming stream * CampaignAnalytics

Options Skip duplicate input columns ⓘ Skip duplicate output columns ⓘ

Input columns * Auto mapping ⓘ Reset + Add mapping Delete 10 mappings: All inf

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** step, then select the **Derived Column** schema modifier.



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
--------	------------

Column	Expression
Revenue	<code>toDecimal(replace(concat(toString(RevenuePart1), toString(Revenue)), '\\\\', ''), 10, 2, '\$###,###.##')</code>
RevenueTarget	<code>toDecimal(replace(concat(toString(RevenueTargetPart1), toString(RevenueTarget)), '\\\\', ''), 10, 2, '\$###,###.##')</code>

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 * ConvertColumnTypesAndValues Learn more ↗

Incoming stream * MapCampaignAnalytics

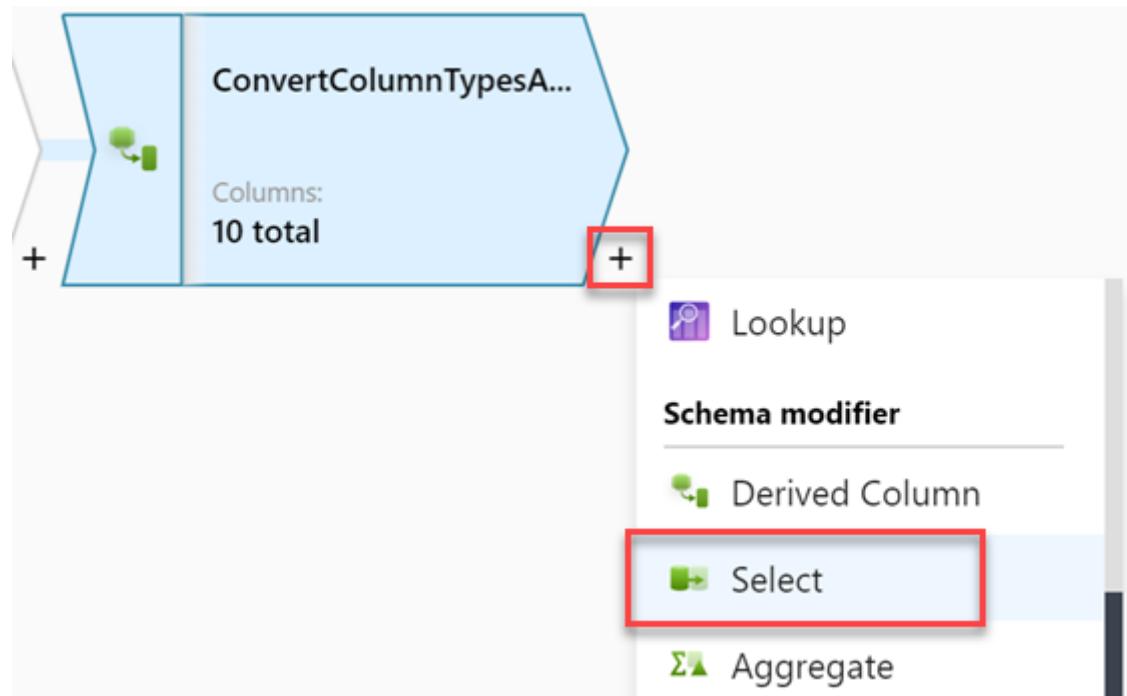
Columns * ⓘ

+ Add Duplicate Delete

Column	Expression
Revenue	<code>toDecimal(replace(concat(toString(RevenuePart1), t... e^x</code>
RevenueTarget	<code>toDecimal(replace(concat(toString(RevenueTargetP... e^x</code>

The expressions you defined will concatenate and clean-up the **RevenuePart1** and **Revenue** values and the **RevenueTargetPart1** and **RevenueTarget** values.

13. Select the + to the right of the **ConvertColumnTypesAndValues** step, then select the **Select** schema modifier from the context menu.



14. Under **Select settings**, configure the following:

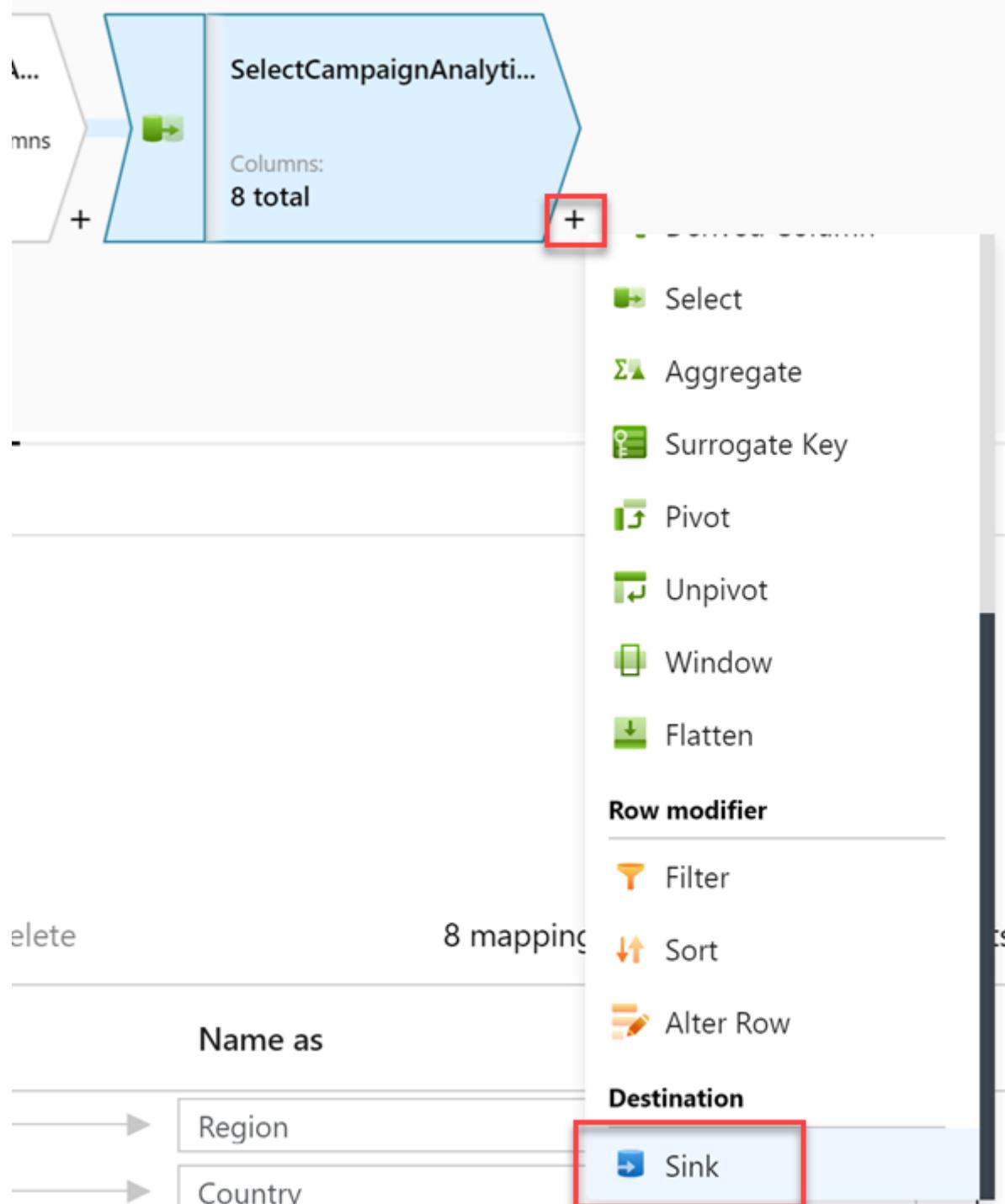
- **Output stream name:** Enter `SelectCampaignAnalyticsColumns`.
- **Incoming stream:** Select **ConvertColumnTypesAndValues**.
- **Options:** Check both options.

- **Input columns:** make sure **Auto mapping** is unchecked, then **Delete RevenuePart1** and **RevenueTargetPart1**. We no longer need these fields.

The screenshot shows the 'Select settings' blade for a data pipeline step. The 'Output stream name' is set to 'SelectCampaignAnalyticsColumns'. The 'Incoming stream' is 'ConvertColumnTypesAndValues'. Under 'Options', 'Skip duplicate input columns' and 'Skip duplicate output columns' are checked. The 'Input columns' section has 'Auto mapping' unchecked. A red box highlights the '8 mappings' message at the top right of the mapping table. The mapping table lists 8 columns from the input stream and their corresponding names in the output stream:

ConvertColumnTypesAndValues'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.



16. Under **Sink**, configure the following:

- **Output stream name:** Enter **CampaignAnalyticsASA**.
- **Incoming stream:** Select **SelectCampaignAnalyticsColumns**.
- **Sink type:** Select **Integration dataset**.
- **Dataset:** Select **asal400_wwi_campaign_analytics_asa**.
- **Options:** Check **Allow schema drift** and uncheck **Validate schema**.

Sink Settings Mapping Optimize Inspect Data preview

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

Incoming stream *

Sink type *

Dataset * [🔗 Test connection](#) [📝 Open](#) [✚ New](#)

Options Allow schema drift ⓘ Validate schema ⓘ

17. On the **Settings** tab, configure the following options:

- **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 ●

i We recommend enabling staging to improve performance with Azure Synapse Analytics datasets.

Update method Allow insert Allow delete Allow upsert Allow update

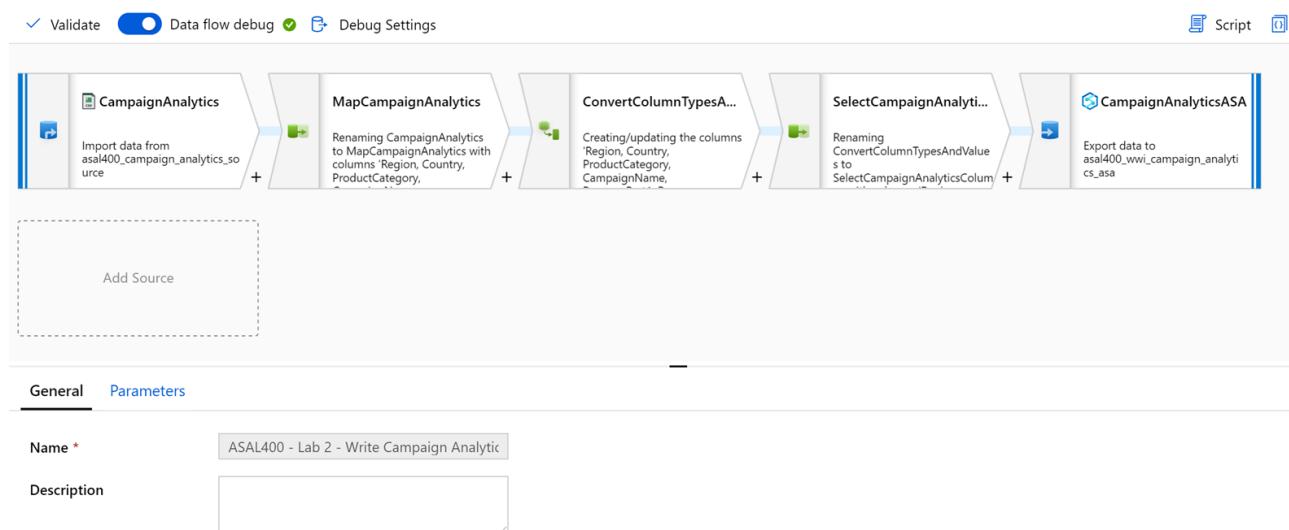
Table action None Recreate table Truncate table

Enable staging

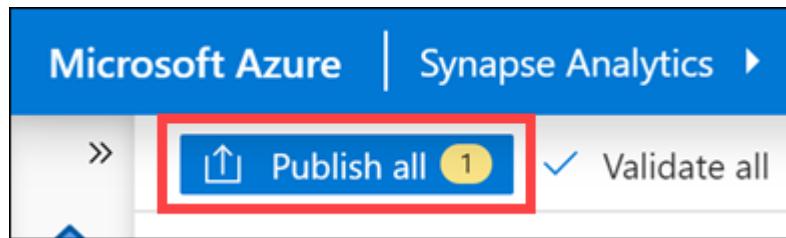
Batch size ⓘ

Pre SQL scripts

18. Your completed data flow should look similar to the following:



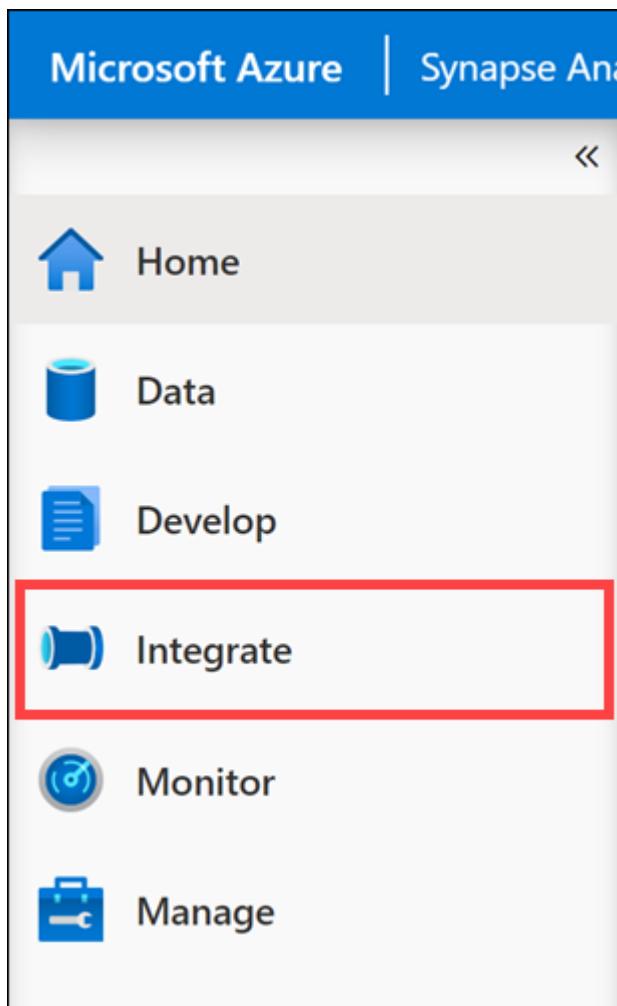
19. Select **Publish all** then **Publish** to save your new data flow.



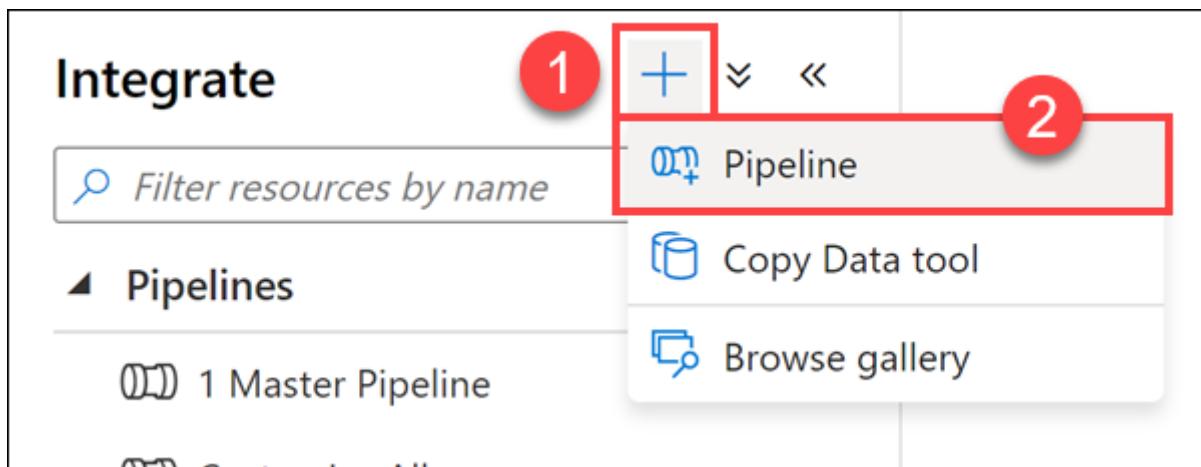
Task 6: Create campaign analytics data pipeline

In order to run the new data flow, you need to create a new pipeline and add a data flow activity to it.

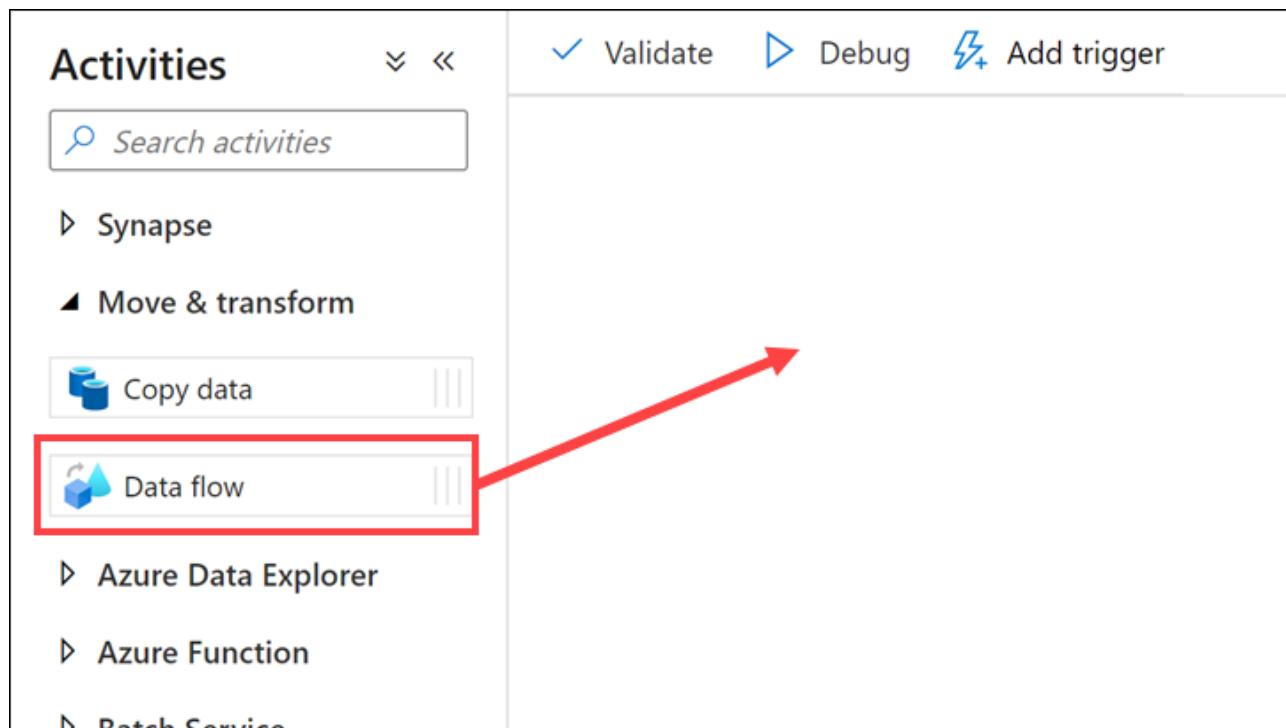
1. Navigate to the **Integrate** hub.



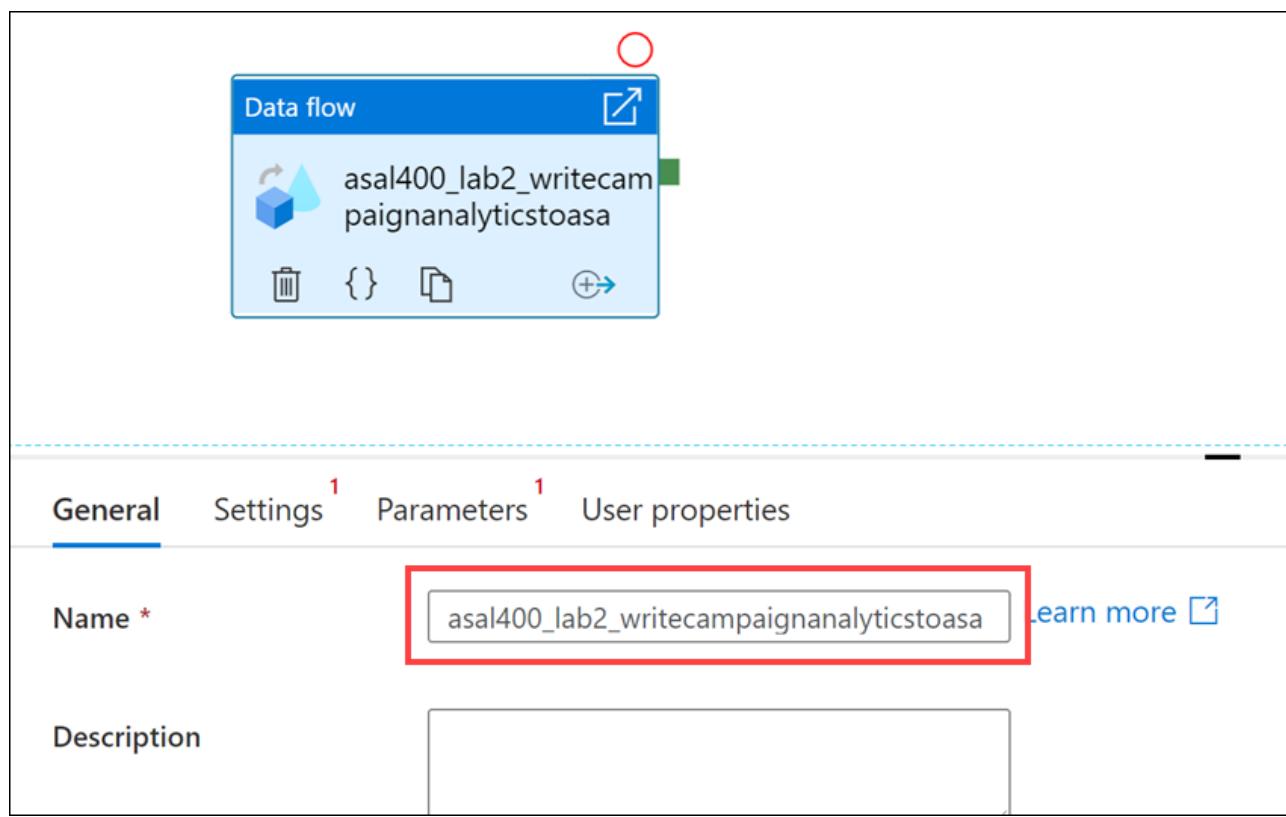
2. In the + menu, select **Pipeline** to create a new pipeline.



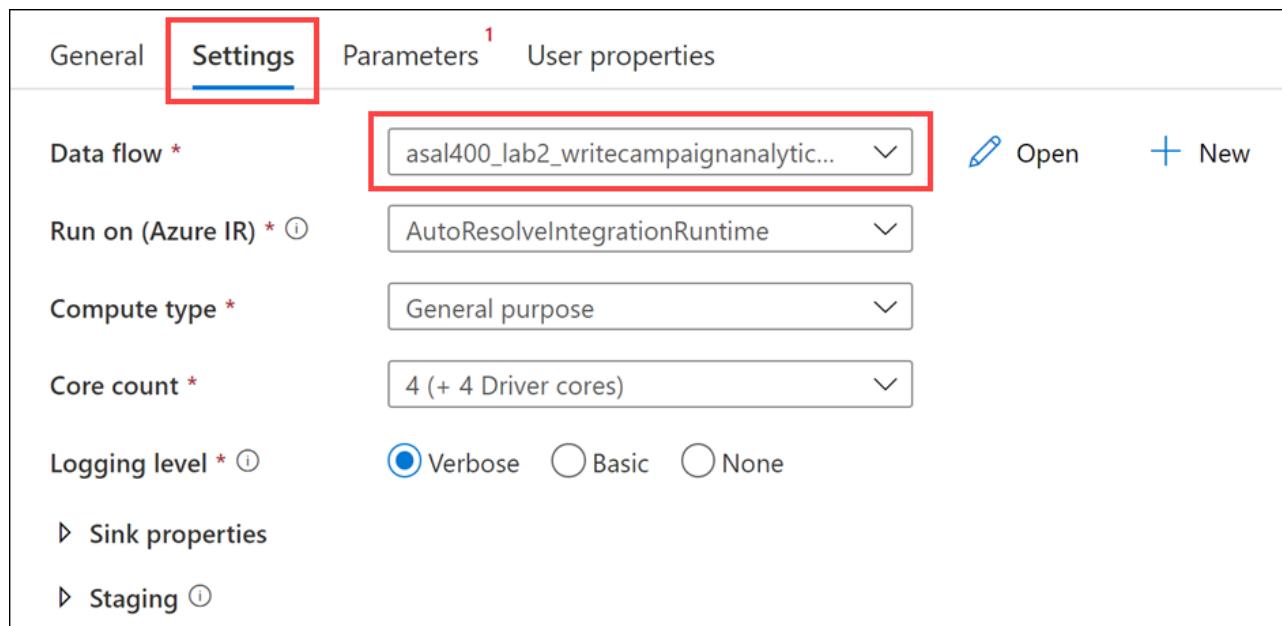
3. In the **General** section of the **Properties** blade for the new pipeline, enter the following **Name:** **Write Campaign Analytics to ASA**.
4. Expand **Move & transform** within the Activities list, then drag the **Data flow** activity onto the pipeline canvas.



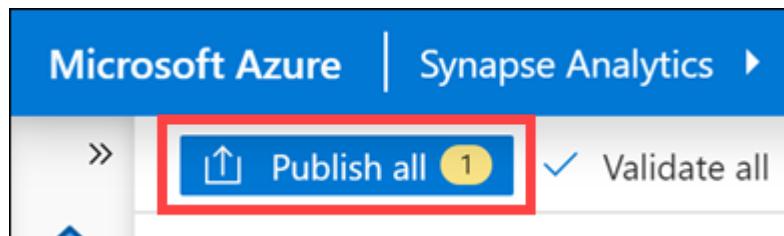
5. On the **General** tab for the data flow (beneath the pipeline canvas), set the **Name** to `asal400_lab2_writecampaignanalyticstoasa`.



6. Select the **Settings** tab; and then, in the **Data flow** list, select `asal400_lab2_writecampaignanalyticstoasa`.

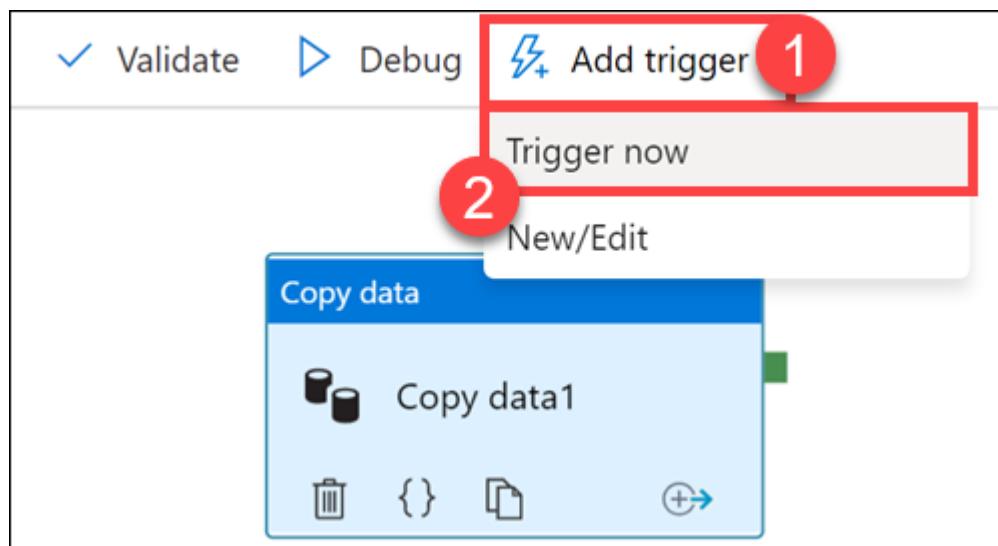


7. Select **Publish all** to save your new pipeline, and then select **Publish**.

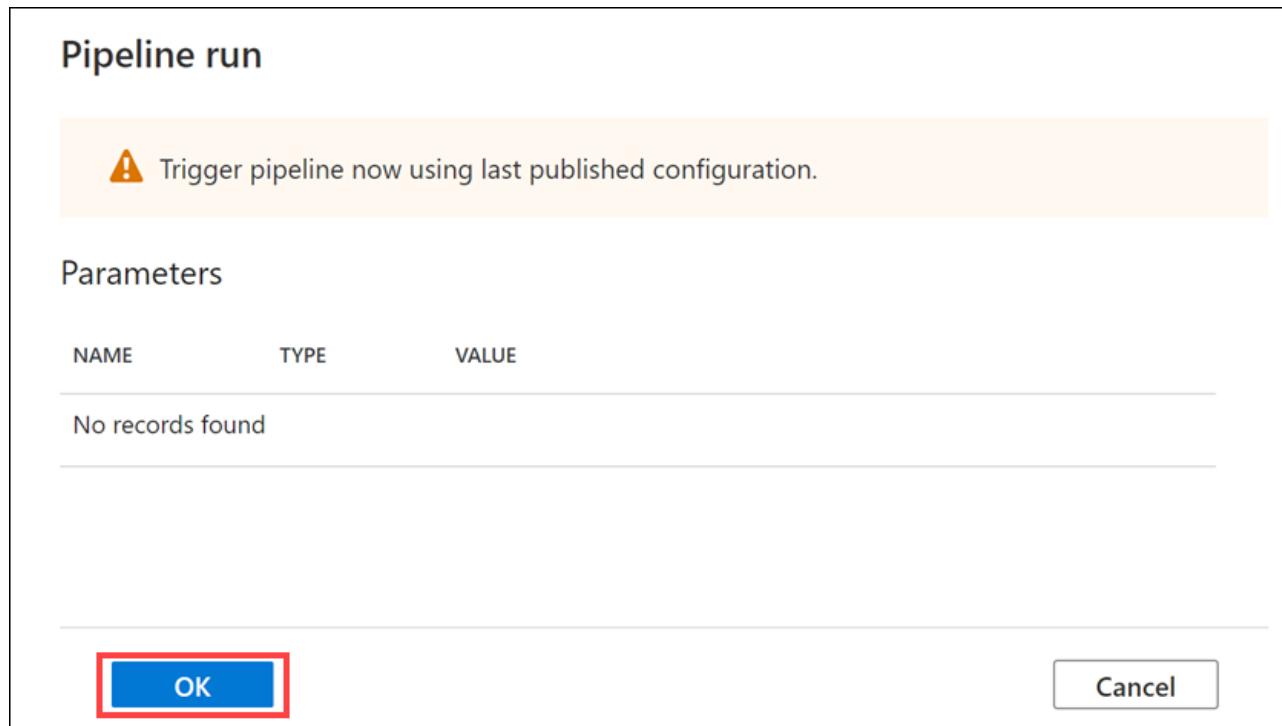


Task 7: Run the campaign analytics data pipeline

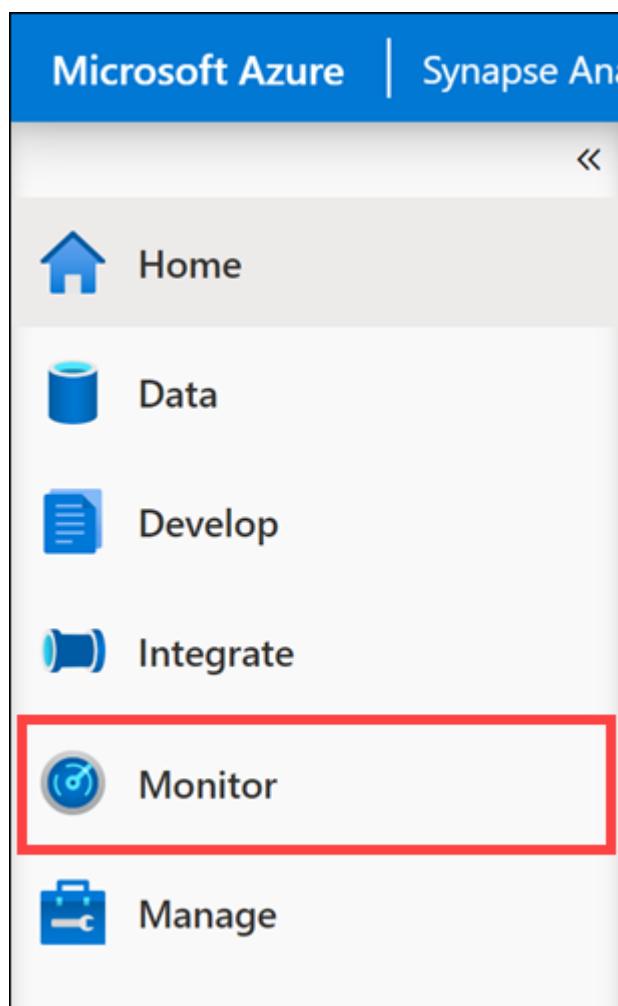
1. Select **Add trigger**, and then select **Trigger now** in the toolbar at the top of the pipeline canvas.



2. In the **Pipeline run** pane, select **OK** to start the pipeline run.



3. Navigate to the **Monitor** hub.



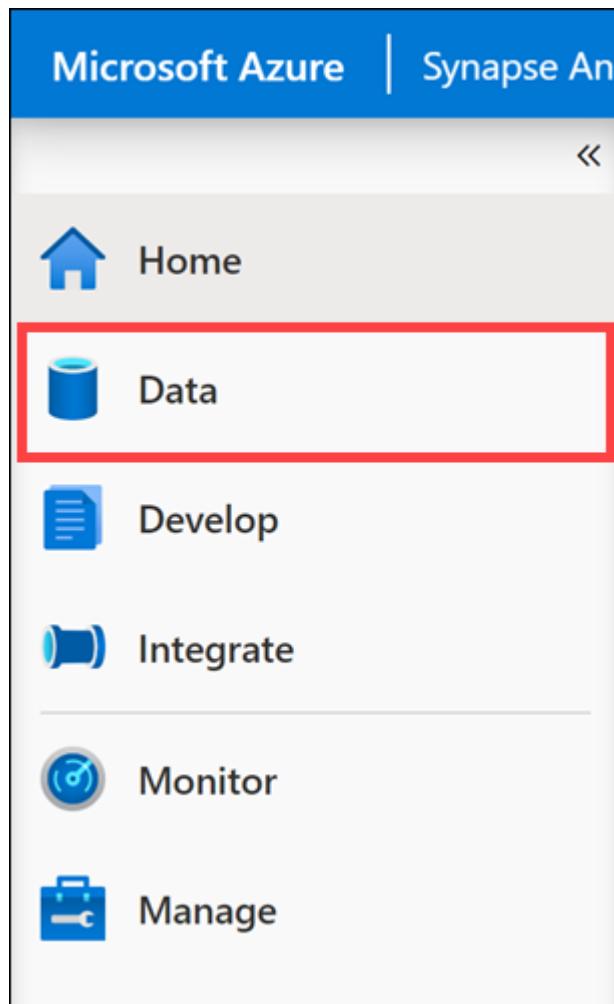
4. Wait for the pipeline run to successfully complete, which will take some time. You may need to refresh the view.

The screenshot shows the Microsoft Azure Synapse Analytics interface. On the left, there's a navigation sidebar with sections like Integration, Activities, and Data flow debug. The main area is titled 'Pipeline runs' and shows a table of recent runs. A single row is highlighted with a red box, representing a successful run named 'Write Campaign Analytics to ...' that started at 12/2/20, 5:18:10 PM and ended at 12/2/20, 5:23:48 PM, triggered by a manual trigger, and succeeded.

Task 8: 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 **SqlPool01** database underneath the **Workspace** section, then expand **Tables** (you may need to refresh to see the new tables).
3. Right-click the **wwi.CampaignAnalytics** table, then select **New SQL script** and **Select TOP 100 rows**.

The screenshot shows the Azure Data Explorer interface under the 'Data' section. A red box highlights the 'Workspace' tab. A search bar contains the placeholder 'Filter resources by name'. Below it, a tree view shows 'Databases' (2) and 'SQLPool01 (SQL)' which contains 'Tables' (2). One table, 'dbo.Category', is expanded to show its children: 'dbo.BookConsumption', 'dbo.BookList', 'dbo.Books', and 'dbo.Category'. Another table, 'wwi.CampaignAnalytics', is selected and highlighted with a red box. A context menu is open over this table, listing options: 'New SQL script' (highlighted with a red box), 'New notebook', 'New data flow', 'New integration dataset', and 'Refresh'. To the right of the menu, actions are listed: 'Select TOP 100 rows' (highlighted with a red box), 'CREATE', 'DROP', and 'DROP and CREATE'.

4. The properly transformed data should appear in the query results.

The screenshot shows a SQL query editor interface. At the top, there are navigation buttons for Run, Undo, Publish, Query plan, and connection settings (Connect to SQLPool01, Use database SQLPool01). Below the toolbar is a code editor window containing the following T-SQL script:

```

1  SELECT TOP (100) [Region]
2  ,[Country]
3  ,[ProductCategory]
4  ,[CampaignName]
5  ,[Revenue]
6  ,[RevenueTarget]
7  ,[City]
8  ,[State]
9  FROM [wwi].[CampaignAnalytics]

```

Below the code editor is a results pane with tabs for Results and Messages. The Results tab is selected, showing a table view of the query's output. The table has columns: Region, Country, ProductCategory, CampaignName, Revenue, RevenueTarget, City, and State. The data is as follows:

Region	Country	ProductCategory	CampaignName	Revenue	RevenueTarget	City	State
South America	Mexico	Apparel and Footwear	Enjoy the Moment	1398.00	5663.00	NU	NA
Europe	Germany	Décor	Tailored for You	7273.00	6184.00	NU	NA
Asia Pacific	China	Apparel and Footwear	Tailored for You	5434.00	10709.00	NU	NA
North & Central America	Canada	Exercise	Be Unique	8465.00	7654.00	NU	NA
South East	US	Team Sports	Enjoy the Moment	16523.00	17741.00	NU	NA
Far West	US	Books	EnjoyTheMoment; BeUn...	6815.00	14606.00	NU	NA
South America	Brazil	Furniture	Enjoy the Moment	11245.00	1841.00	NU	NA
Europe	France	Pillows & Cushions	Enjoy the Moment	15557.00	14176.00	NU	NA
Asia Pacific	Japan	Lighting	Tailored for You	15642.00	10214.00	NU	NA

At the bottom of the results pane, a message indicates the query was executed successfully.

5. Modify the query as follows and run the script:

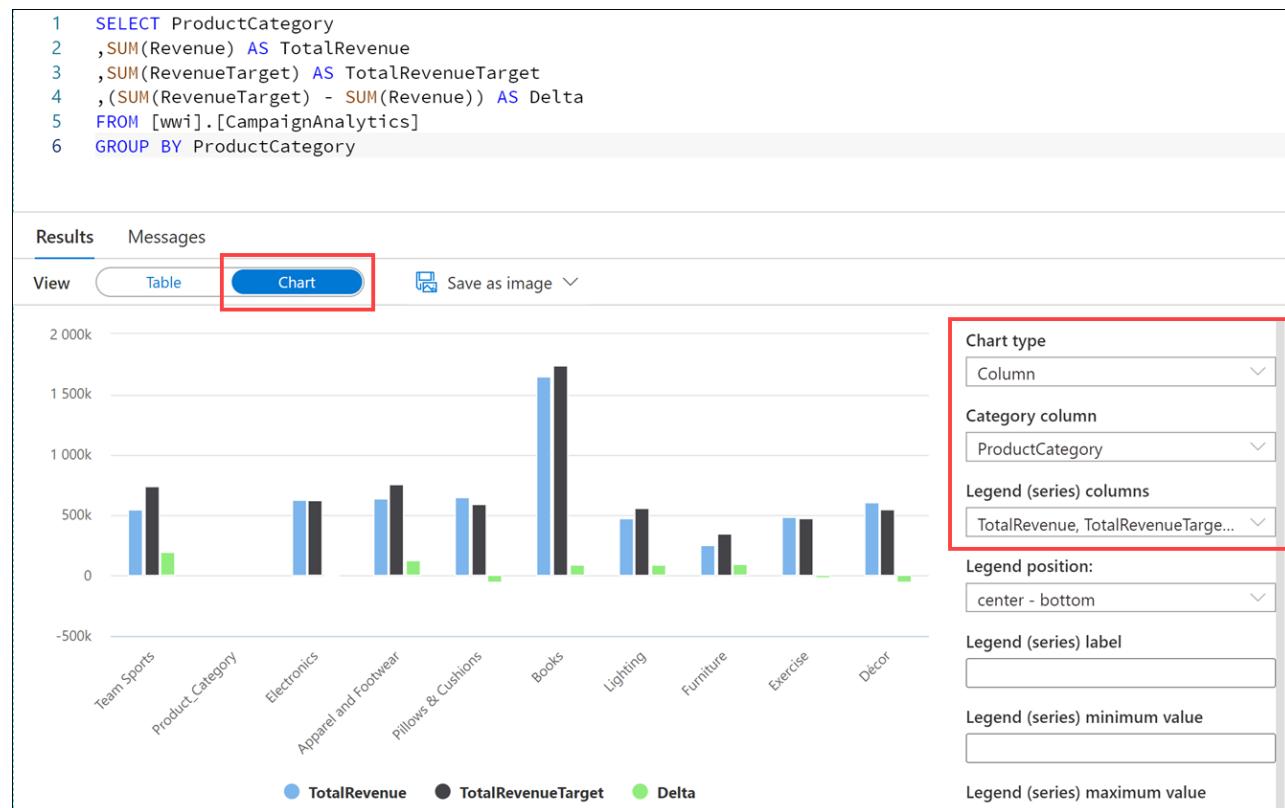
```

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:** Column.
- **Category column:** ProductCategory.
- **Legend (series) columns:** TotalRevenue, TotalRevenueTarget, and Delta.



Exercise 2 - Create Mapping Data Flow for top product purchases

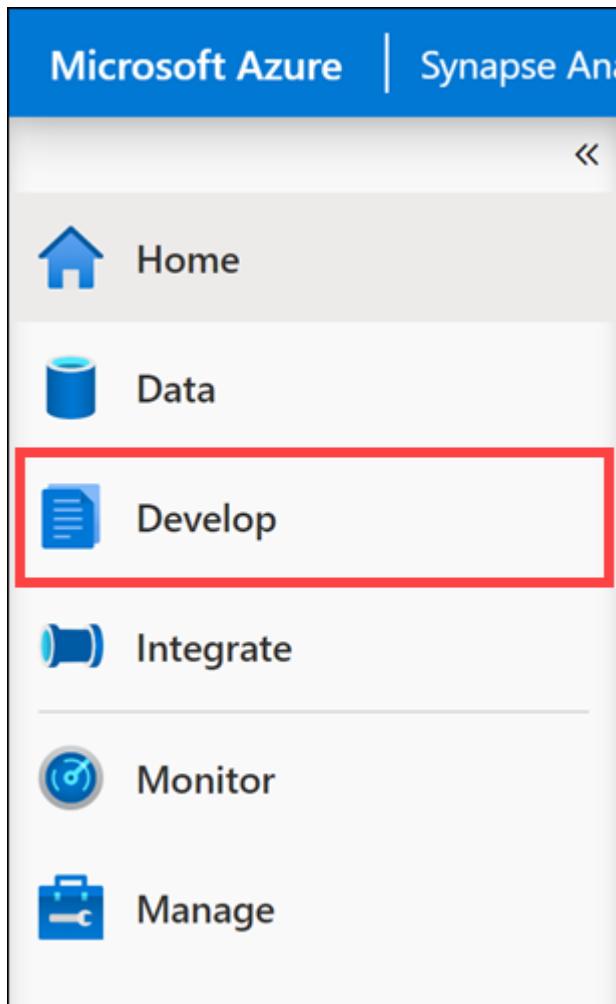
Tailwind Traders needs to combine top product purchases imported as JSON files from their eCommerce system with user preferred products from profile data stored as JSON documents in Azure Cosmos DB. They want to store the combined data in a dedicated SQL pool as well as their data lake for further analysis and reporting.

To do this, you will build a mapping data flow that performs the following tasks:

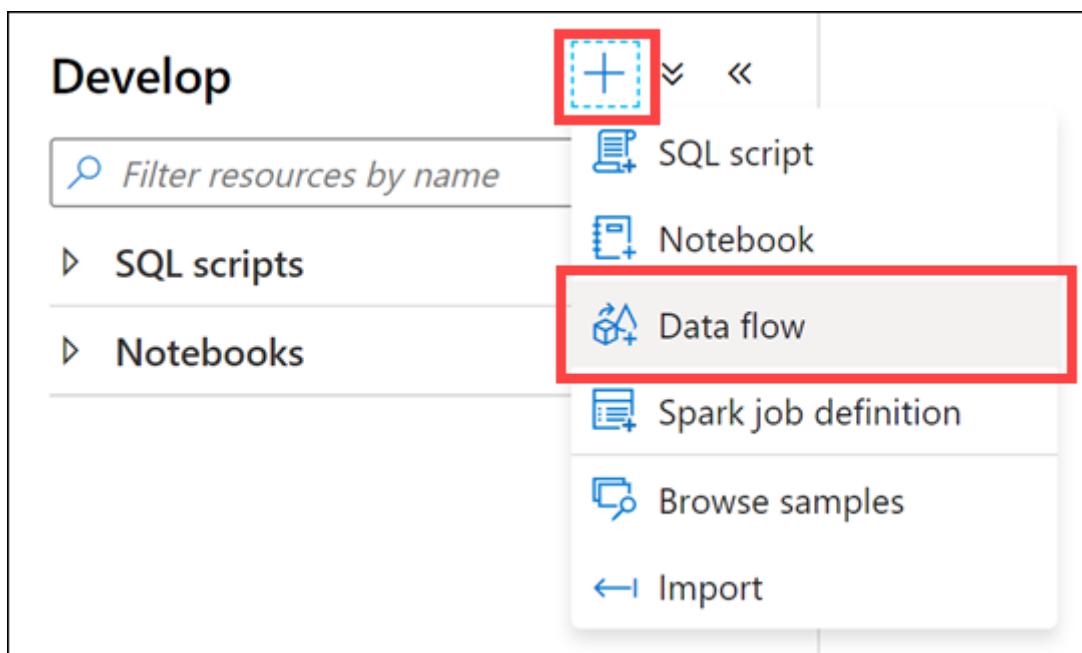
- Adds two ADLS Gen2 data sources for the JSON data
- Flattens the hierarchical structure of both sets of files
- Performs data transformations and type conversions
- Joins both data sources
- Creates new fields on the joined data set based on conditional logic
- Filters null records for required fields
- Writes to the dedicated SQL pool
- Simultaneously writes to the data lake

Task 1: Create Mapping Data Flow

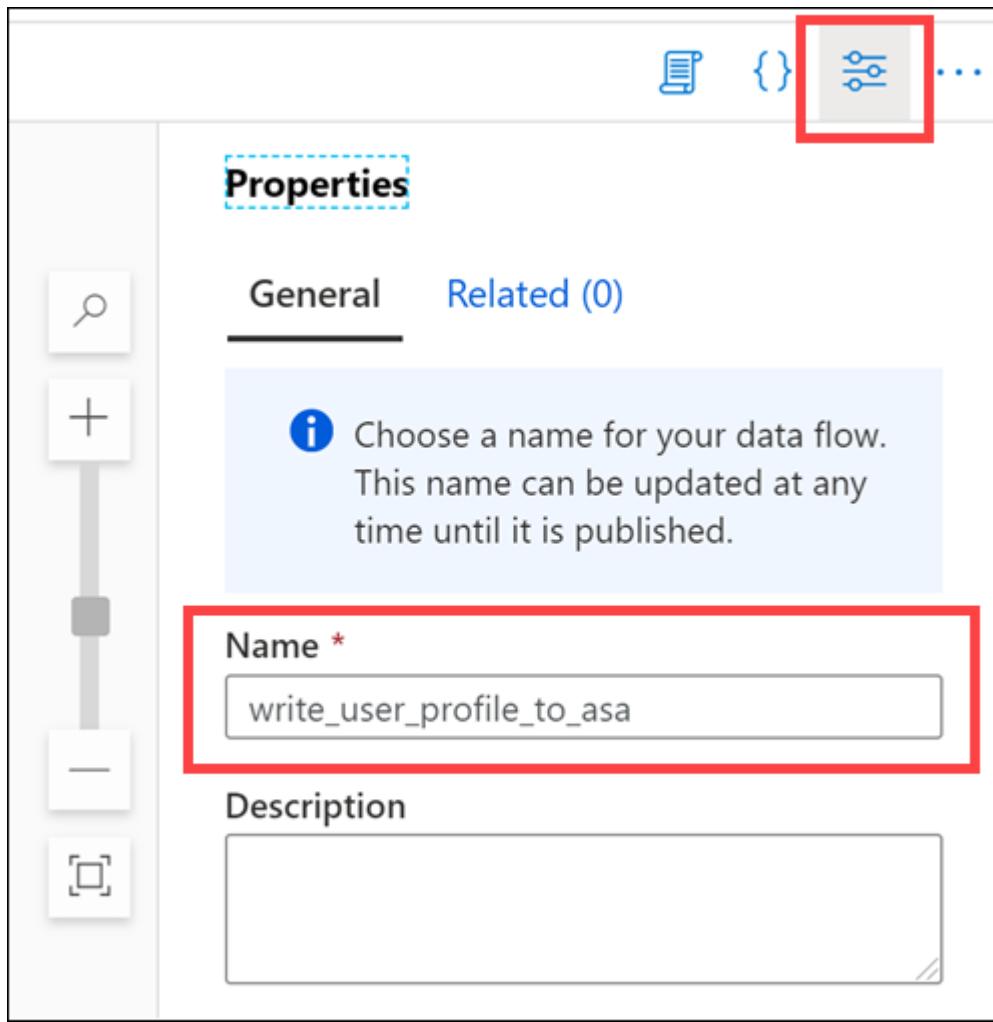
1. In Synapse Analytics Studio, navigate to the **Develop** hub.



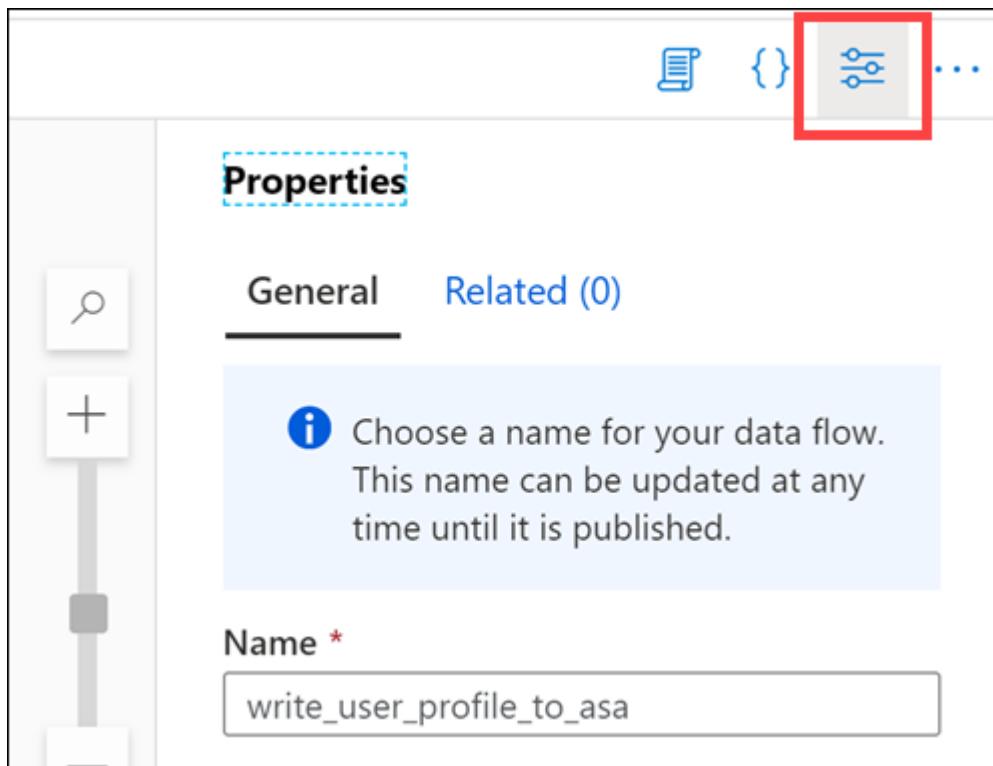
2. In the + menu, select **Data flow** to create a new data flow.



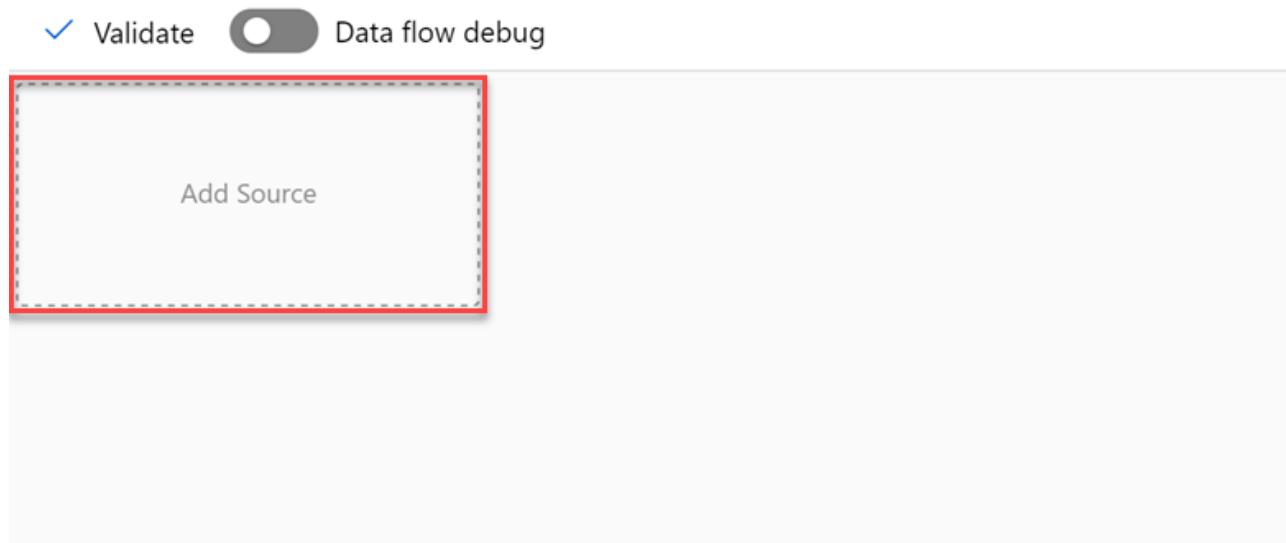
3. In the **General** section of the **Properties** pane of the new data flow, update the **Name** to the following:
write_user_profile_to_asa.



4. Select the **Properties** button to hide the pane.



5. Select **Add Source** on the data flow canvas.



6. Under **Source settings**, configure the following:

- **Output stream name:** Enter **EcommerceUserProfiles**.
- **Source type:** Select **Integration dataset**.
- **Dataset:** Select **asal400_ecommerce_userprofiles_source**.

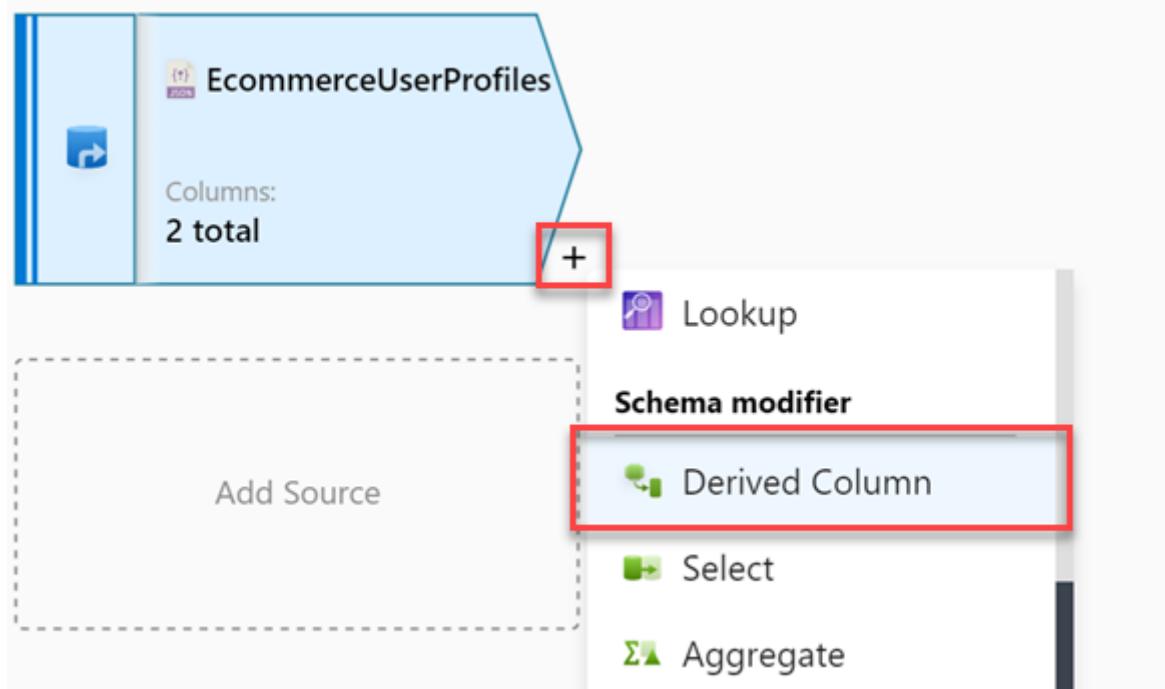
The screenshot shows the 'Source settings' tab selected in the top navigation bar. Below it, the configuration options are displayed:

- Output stream name ***: EcommerceUserProfiles
- Source type ***: Integration dataset (selected)
- Dataset ***: asal400_ecommerce_userprofiles_so... (dropdown menu)
- Options**:
 - Allow schema drift ⓘ
 - Infer drifted column types ⓘ
 - Validate schema ⓘ
- Sampling ***:
 - Enable
 - Disable

7. Select the **Source options** tab, then configure the following:

- **Wildcard paths:** Enter **online-user-profiles-02/*.json**

8. Select the + to the right of the **EcommerceUserProfiles** source, then select the **Derived Column** schema modifier.



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

- **Output stream name:** Enter `userId`.
- **Incoming stream:** Select **EcommerceUserProfiles**.
- **Columns:** Provide the following information:

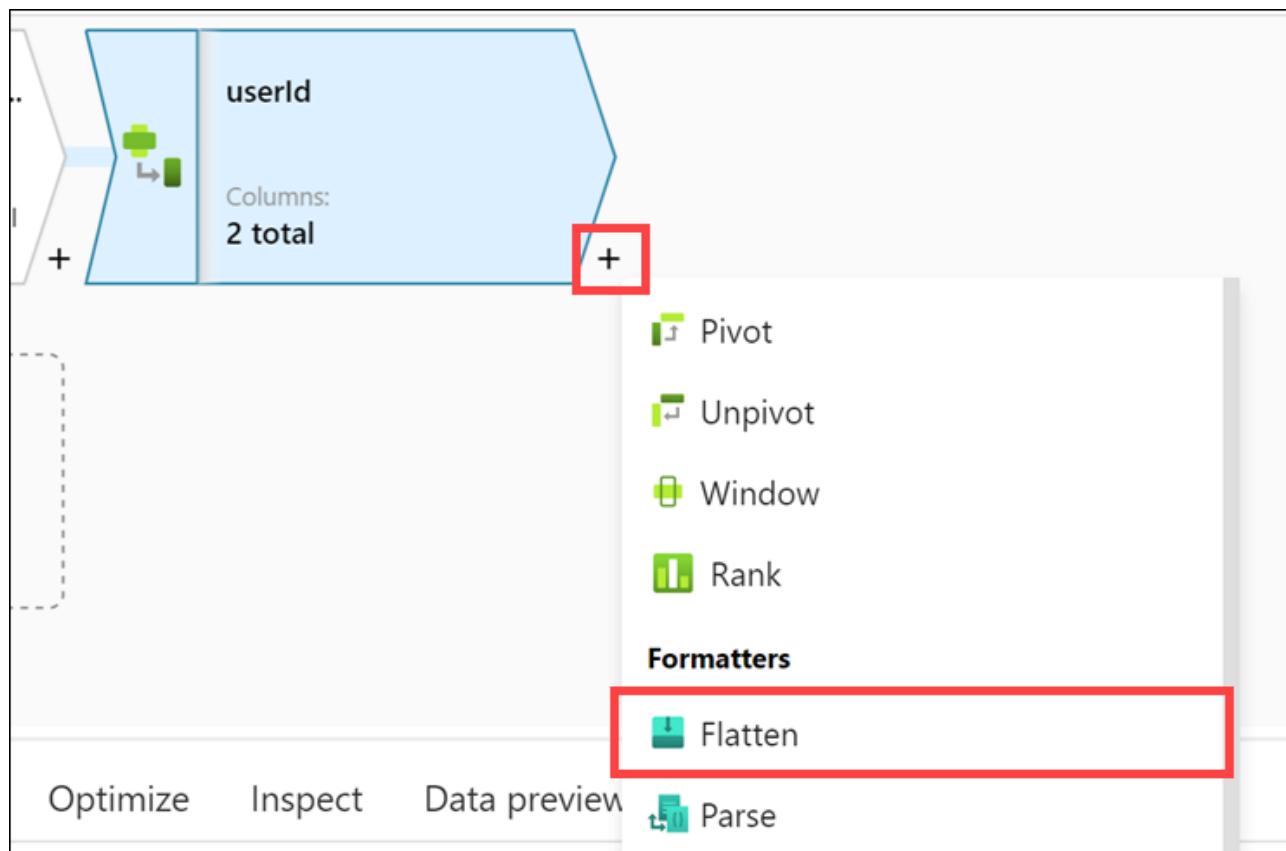
Column Expression

visitorId `toInteger(visitorId)`

The screenshot shows the "Derived column's settings" dialog. At the top, there are tabs for "Derived column's settings", "Optimize", "Inspect", and "Data preview". On the far right is a "Description" button. The "Output stream name" field contains "userId" and has a red box around it. The "Incoming stream" dropdown is set to "EcommerceUserProfiles". Below these are buttons for "+ Add", "Clone", "Delete", and "Open expression builder". The "Columns" section shows a table with one row. The first column is "Column" and the second is "Expression". The "Column" row contains "visitorId" and has a red box around it. The "Expression" row contains "`toInteger(visitorId)`" and also has a red box around it. There are "Add" and "Delete" buttons at the bottom right of the table.

This expression converts the **visitorId** column value to an integer data type.

10. Select the + to the right of the **userId** step, then select the **Flatten** formatter.



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

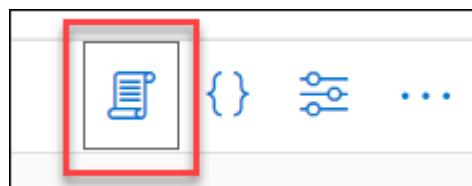
Input columns * **Add mapping**

Original Column	Name as
visitorId	visitorId
productId	productId
itemsPurchasedLast12Months	itemsPurchasedLast12Months

3 mappings: All inputs mapped

These settings provide a flattened representation of the data.

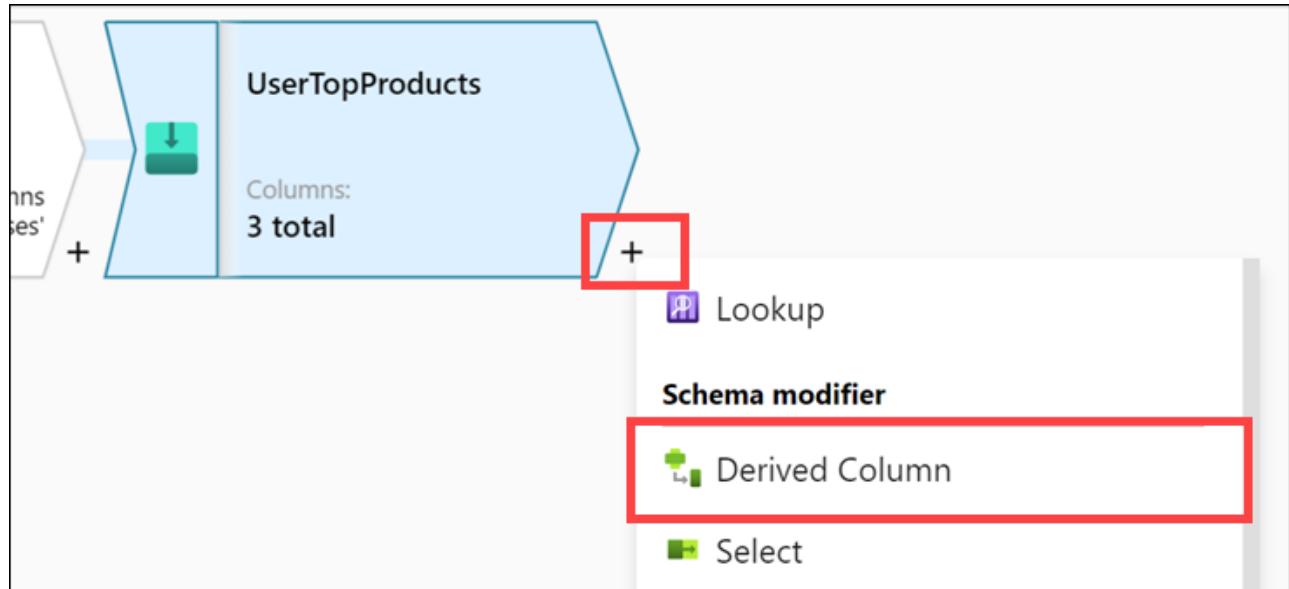
12. The user interface defines the mappings by generating a script. To view the script, select the **Script** button on the toolbar.



Verify that the script looks like this and then **Cancel** to return the graphical UI (if not, modify the script):

```
source(output(
    visitorId as string,
    topProductPurchases as (productId as string,
    itemsPurchasedLast12Months as string)[])
),
allowSchemaDrift: true,
validateSchema: false,
ignoreNoFilesFound: false,
documentForm: 'arrayOfDocuments',
wildcardPaths:['online-user-profiles-02/*.json']) ~>
EcommerceUserProfiles
EcommerceUserProfiles derive(visitorId = toInteger(visitorId)) ~> userId
userId foldDown(unroll(topProductPurchases),
mapColumn(
    visitorId,
    productId = topProductPurchases.productId,
    itemsPurchasedLast12Months =
    topProductPurchases.itemsPurchasedLast12Months
),
skipDuplicateMapInputs: false,
skipDuplicateMapOutputs: false) ~> UserTopProducts
```

13. Select the + to the right of the **UserTopProducts** step, then select the **Derived Column** schema modifier from the context menu.



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

- **Output stream name:** Enter `DeriveProductColumns`.
- **Incoming stream:** Select **UserTopProducts**.
- **Columns:** Provide the following information:

Column	Expression
productId	<code>toInteger(productId)</code>
itemsPurchasedLast12Months	<code>toInteger(itemsPurchasedLast12Months)</code>

The screenshot shows the 'Derived column's settings' dialog. At the top, tabs include 'Derived column's settings' (selected), 'Optimize', 'Inspect', and 'Data preview'. Below the tabs, the 'Output stream name' is set to 'DeriveProductColumns'. The 'Incoming stream' is set to 'UserTopProducts'. In the 'Columns' section, there's a red-bordered '+' button. The table lists columns with expressions: 'productId' with expression 'toInteger(productId)' and 'itemsPurchasedLast12Months' with expression 'toInteger(itemsPurchasedLast12Months)'. Each row has a red-bordered 'Delete' button.

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](#)

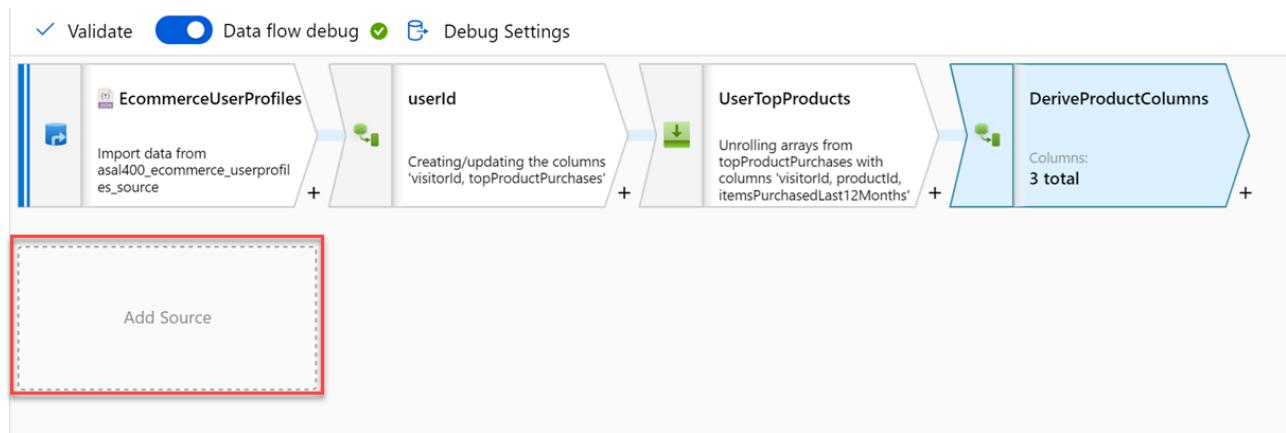
Columns * ⓘ

Column	Expression
<input type="text" value="productId"/>	<input type="text" value="tolong(productId)"/> 123 + ↻

[Add column](#) [Add column pattern](#)

These expressions convert the **productId** and **itemsPurchasedLast12Months** columns values to integers.

15. Select **Add Source** on the data flow canvas beneath the **EcommerceUserProfiles** source.



16. Under **Source settings**, configure the following:

- **Output stream name:** Enter **UserProfiles**.
- **Source type:** Select **Integration dataset**.
- **Dataset:** Select **asal400_customerprofile_cosmosdb**.

Source settings [Source options](#) [Projection](#) [Optimize](#) [Inspect](#) [Data preview](#)

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

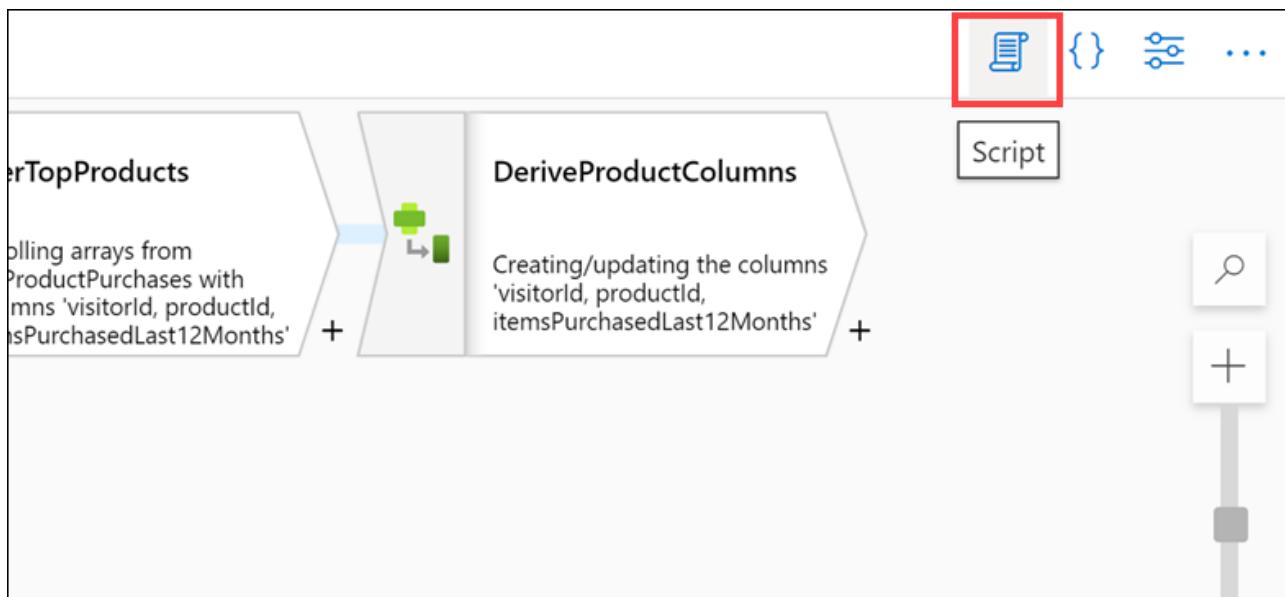
Source type *

Dataset * [▼](#) [🔗](#) [Test connection](#) [Open](#) [+](#) [New](#)

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

Sampling * ⓘ Enable Disable

17. 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.



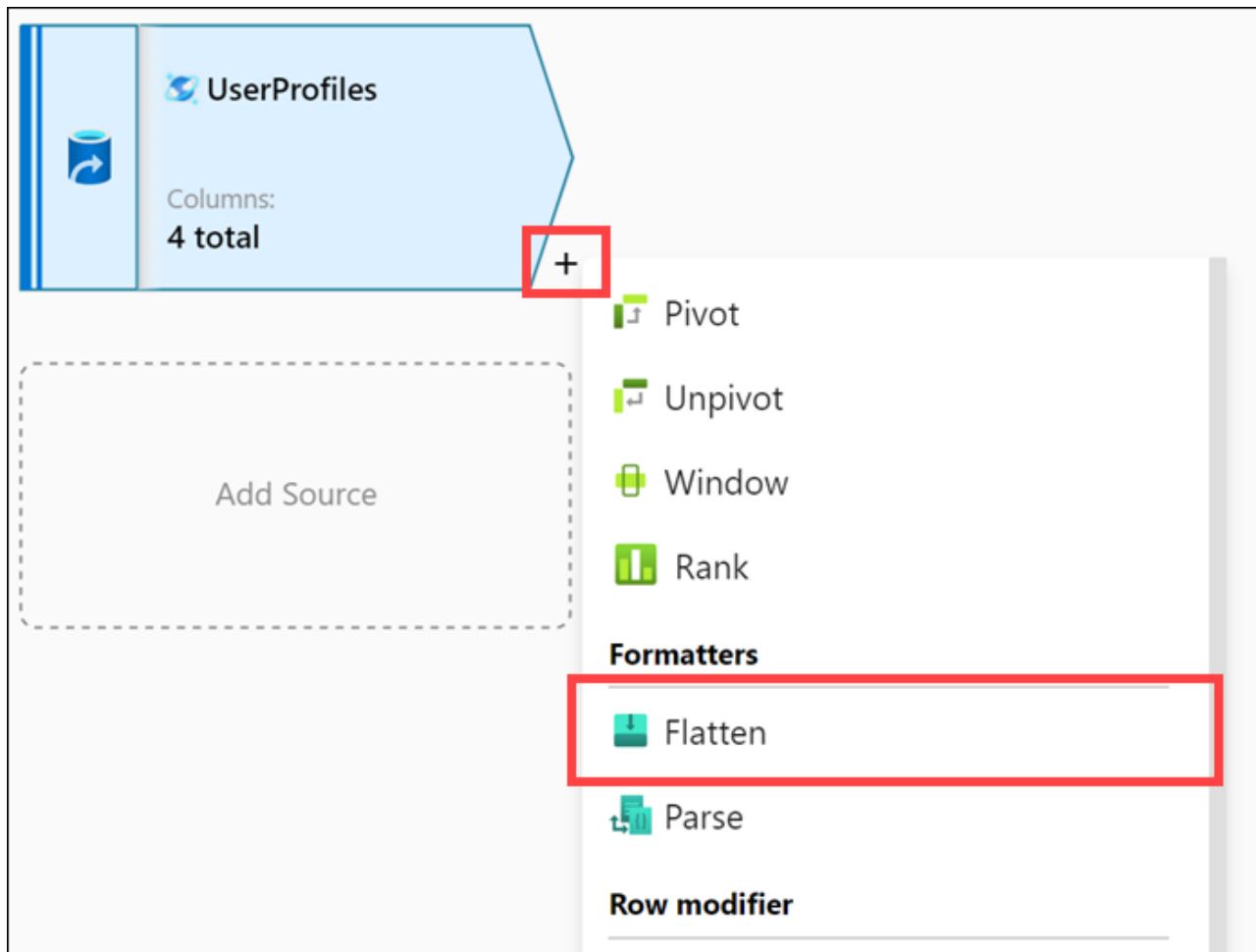
18. Locate the **UserProfiles** source in the script, which looks like this:

```
source(output(
    userId as string,
    cartId as string,
    preferredProducts as string[],
    productReviews as (productId as string, reviewText as string, reviewDate as string)[]
),
allowSchemaDrift: true,
validateSchema: false,
format: 'document') ~> UserProfiles
```

19. Modify script block as follows to set **preferredProducts** as an **integer[]** array and ensure the data types within the **productReviews** array are correctly defined. Then select **OK** to apply the script changes.

```
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
```

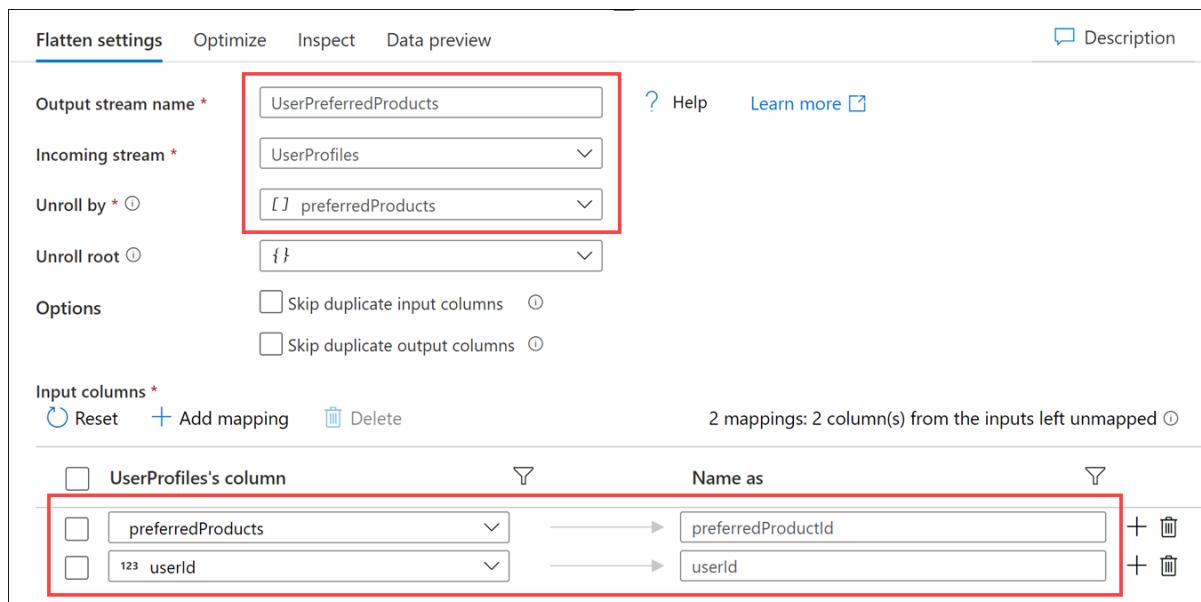
20. Select the + to the right of the **UserProfiles** source, then select the **Flatten** formatter.



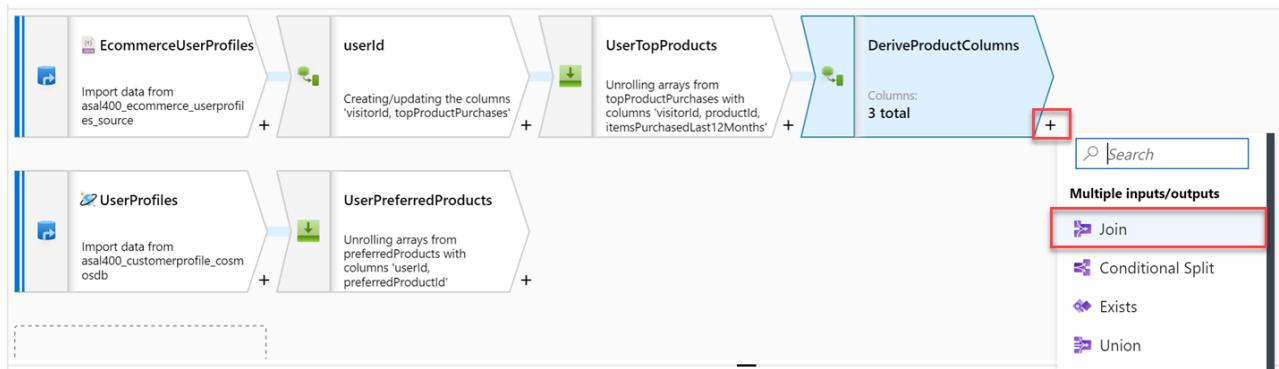
21. Under **Flatten settings**, configure the following:

- **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
[] preferredProducts	preferredProductId
userId	userId



22. Now it is time to join the two data sources. Select the **+** to the right of the **DeriveProductColumns** step, then select the **Join** option.



23. 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 ●

Output stream name * Learn more ↗

Left stream *

Right stream *

Join type * Full outer Inner Left outer Right outer Custom (cross)

Join conditions * ==

Description

24. 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 Auto ⓘ Fixed ⓘ Off ⓘ

Broadcast options * Left: 'DeriveProductColumns' ⓘ Right: 'UserPreferredProducts' ⓘ

Partition option * Use current partitioning Single partition Set partitioning

Partition type *

Number of partitions *

Column values to hash on * **Column** + ⚡

25. 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	

26. Select the + to the right of the **JoinTopProductsWithPreferredProducts** step, then select the **Derived Column** schema modifier.



27. 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
<code>isTopProduct</code>	<code>toBoolean(iif(isNull(productId), 'false', 'true'))</code>
<code>isPreferredProduct</code>	<code>toBoolean(iif(isNull(preferredProductId), 'false', 'true'))</code>
<code>productId</code>	<code>iif(isNull(productId), preferredProductId, productId)</code>
<code>userId</code>	<code>iif(isNull(userId), visitorId, userId)</code>

Derived column's settings Optimize Inspect Data preview ●

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

Incoming stream *

[+ Add](#) [Clone](#) [Delete](#) [Open expression builder](#)

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)

The derived column settings will provide the following result when the pipeline is run:

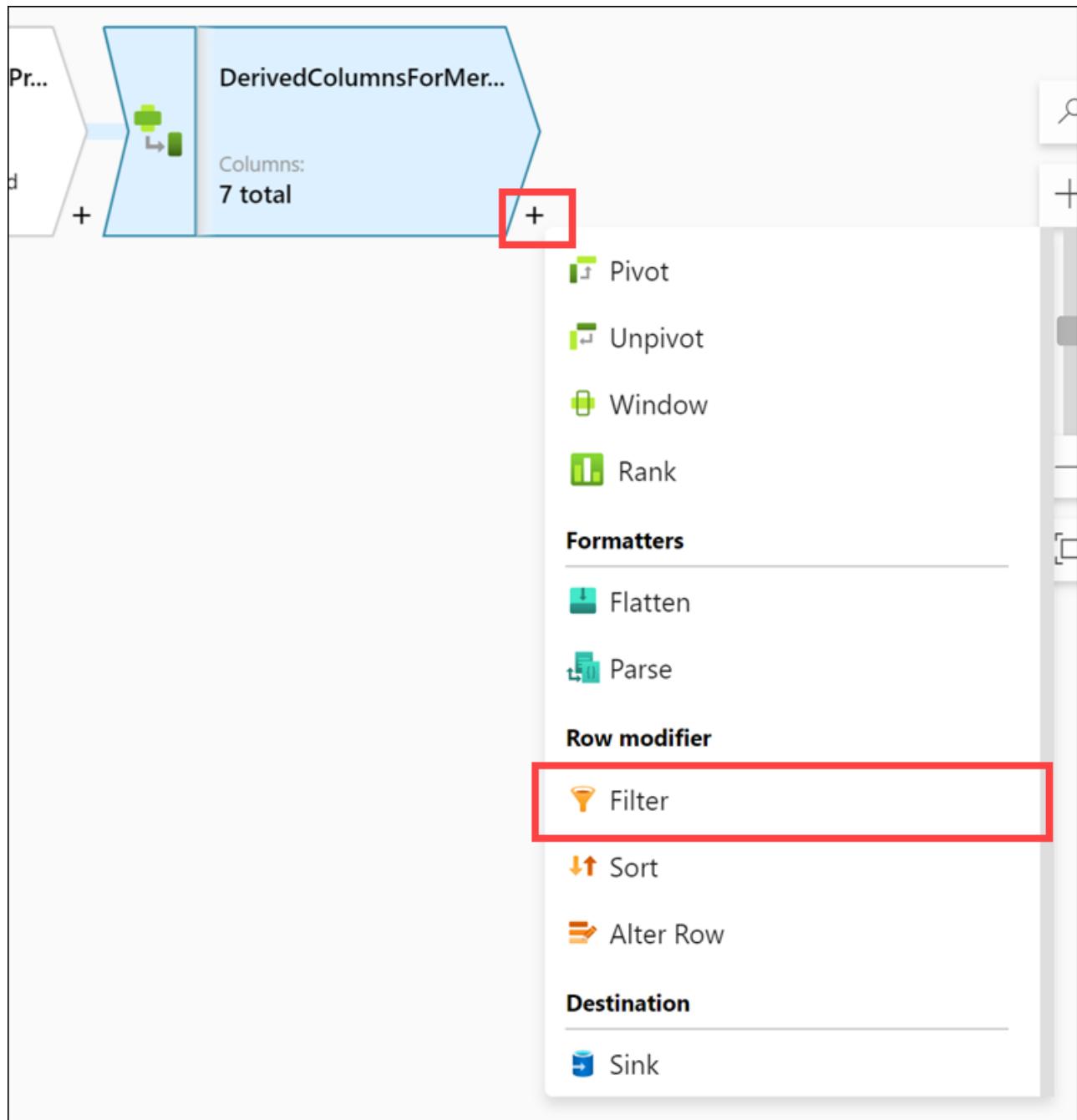
Derived column's settings Optimize Inspect Data preview ●

Number of rows [+ INSERT](#) 100 [* UPDATE](#) 0 [✖ DELETE](#) 0 [* UPSERT](#) 0 [🔍 LOOKUP](#) 0 **TOTAL** 15361

Refresh Typecast [▼](#) Modify [▼](#) Map drifted Statistics Remove

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	✓	✗

28. Select the + to the right of the **DerivedColumnsForMerge** step, then select the **Filter** row modifier.



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.

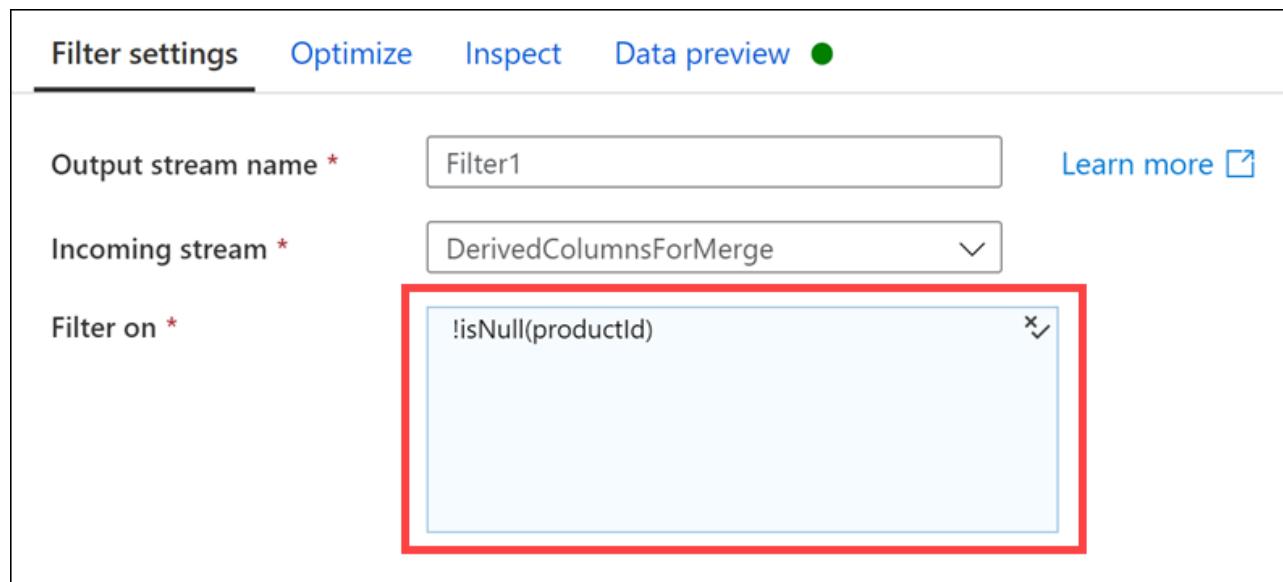
29. Set the **Filter on** expression to `!isNull(productId)`.

Filter settings Optimize Inspect Data preview ●

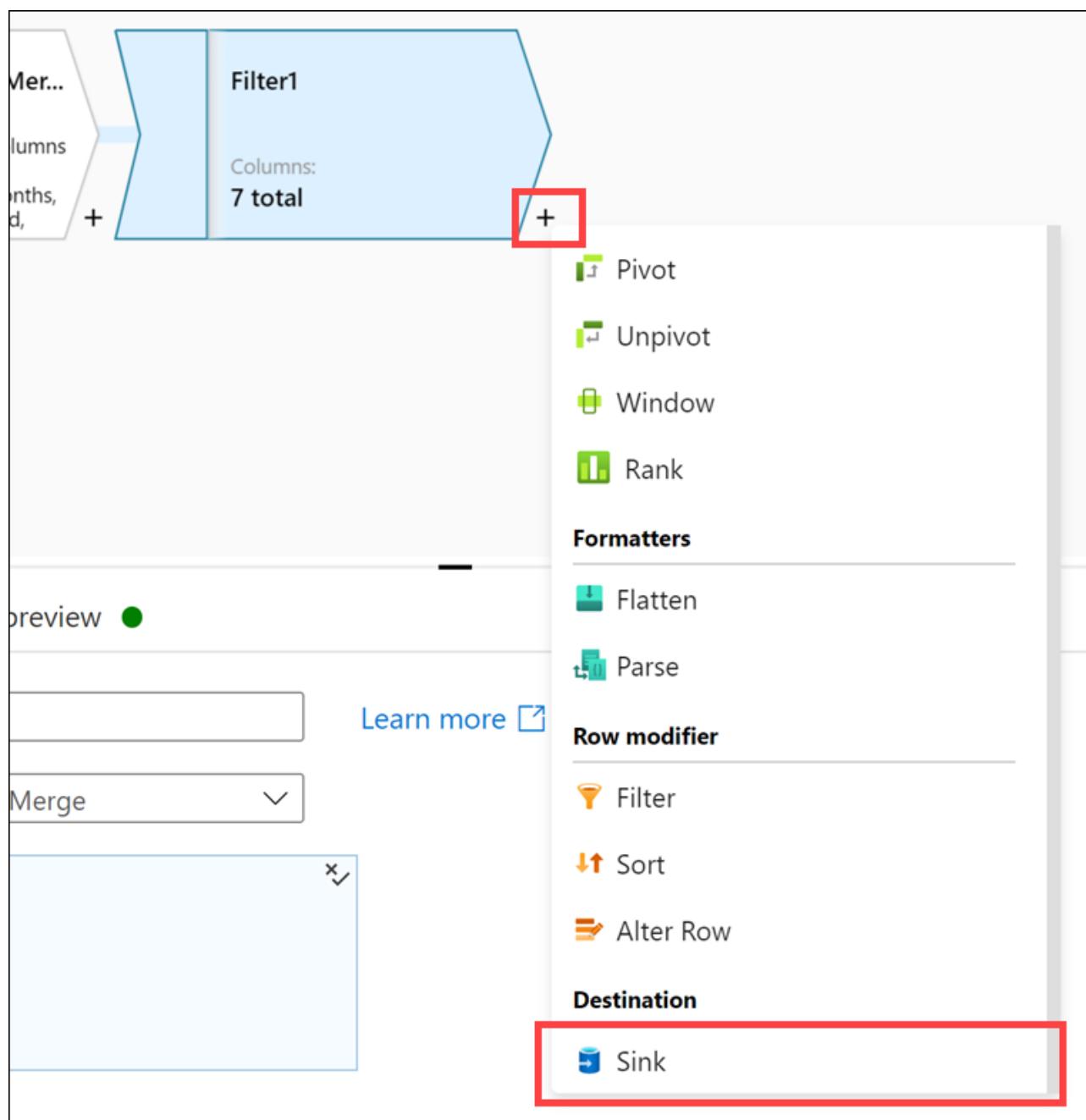
Output stream name * Filter1 [Learn more](#)

Incoming stream * DerivedColumnsForMerge

Filter on * !isNull(productId)



30. Select the + to the right of the **Filter1** step, then select the **Sink** destination from the context menu.



31. Under **Sink**, configure the following:

- **Output stream name:** Enter **UserTopProductPurchasesASA**.
- **Incoming stream:** Select **Filter1**.
- **Sink type:** select **Integration Dataset**.
- **Dataset:** Select **asal400_wwi_usertopproductpurchases_asa**.
- **Options:** Check **Allow schema drift** and uncheck **Validate schema**.

Sink Settings Mapping Optimize Inspect Data preview

Output stream name * UserTopProductPurchasesASA Learn more ↗

Incoming stream * Filter1

Sink type * Integration dataset Inline Workspace DB Cache

Dataset * asal400_wwi_usertopproductpurchas... Test connection Open New

Options Allow schema drift ⓘ Validate schema ⓘ

32. 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.

Sink **Settings** Mapping Optimize Inspect Data preview

Update method Allow insert Allow delete Allow upsert Allow update

Table action None Recreate table Truncate table

Enable staging

Batch size ⓘ

33. Select **Mapping**, then configure the following:

- **Auto mapping:** De-select 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 ⓘ

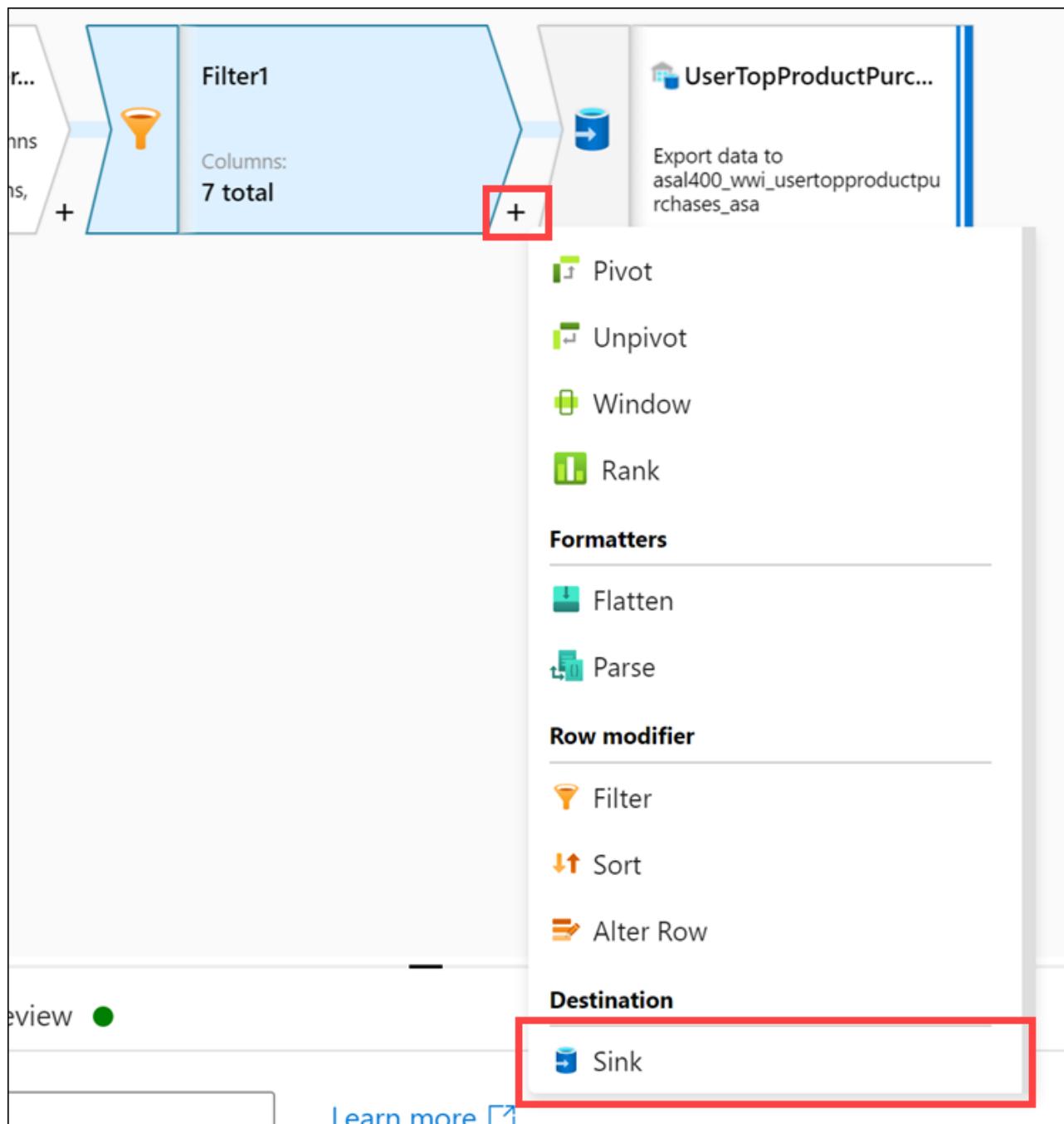
Skip duplicate output columns ⓘ

Auto mapping ⓘ (OFF) Reset Add mapping Delete Output format

5 mappings: All outputs mapped

Input columns	Output columns
123 userId	123 UserId
123 productId	123 ProductId
123 itemsPurchasedLast12Months	123 ItemsPurchasedLast12Months
✗ isTopProduct	✗ IsTopProduct
✗ isPreferredProduct	✗ IsPreferredProduct

34. Select the + to the right of the **Filter1** step, then select the **Sink** destination from the context menu to add a second sink.



35. 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 **asaworkspacexxxxxxx-WorkspaceDefaultStorage**.
- **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 *

Integration dataset	Inline	Workspace DB	Cache
---------------------	--------	--------------	-------

Inline dataset type * Delta

Linked service * asaworkspacec72egsr-WorkspaceDe... [Test connection](#) [Edit](#) [New](#)

Options

Allow schema drift ⓘ

Validate schema ⓘ

36. Select **Settings**, then configure the following:

- **Folder path:** Enter `wwi-02 / top-products` (type 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`.
- **Table action:** Select **Truncate**.
- **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

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 ⓘ

37. Select **Mapping**, then configure the following:

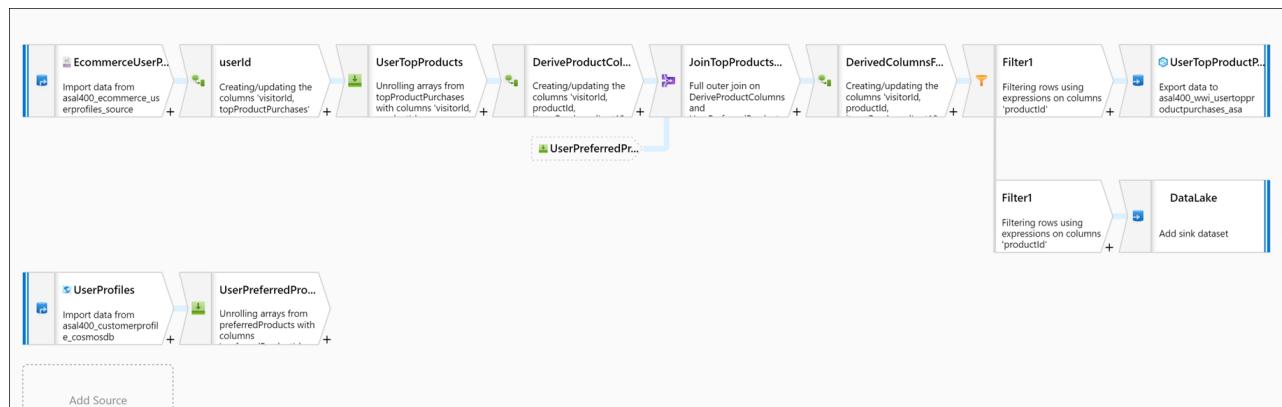
- **Auto mapping:** Uncheck this option.
- **Columns:** Define the following column mappings:

Input columns	Output columns
visitorId	visitorId
productId	productId
itemsPurchasedLast12Months	itemsPurchasedLast12Months
preferredProductId	preferredProductId
userId	userId
isTopProduct	isTopProduct
isPreferredProduct	isPreferredProduct

The screenshot shows the 'Mapping' tab in the Azure Synapse Analytics interface. Under 'Options', there are three checkboxes: 'Skip duplicate input columns', 'Skip duplicate output columns', and 'Auto mapping'. The 'Auto mapping' checkbox is highlighted with a red box. Below this is a table for mapping input columns to output columns. The input columns are: visitorId, productId, itemsPurchasedLast12Months, preferredProductId, userId, isTopProduct, and isPreferredProduct. The output columns are: visitorId, productId, itemsPurchasedLast12Months, preferredProductId, userId, isTopProduct, and isPreferredProduct. Each input column has a dropdown arrow next to it, and each output column has a trash can icon.

Notice that we have chosen to keep two additional 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.

38. Verify that your completed data flow looks similar to the following:



39. Select **Publish all**, then **Publish** to save your new data flow.



Exercise 3 - 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.

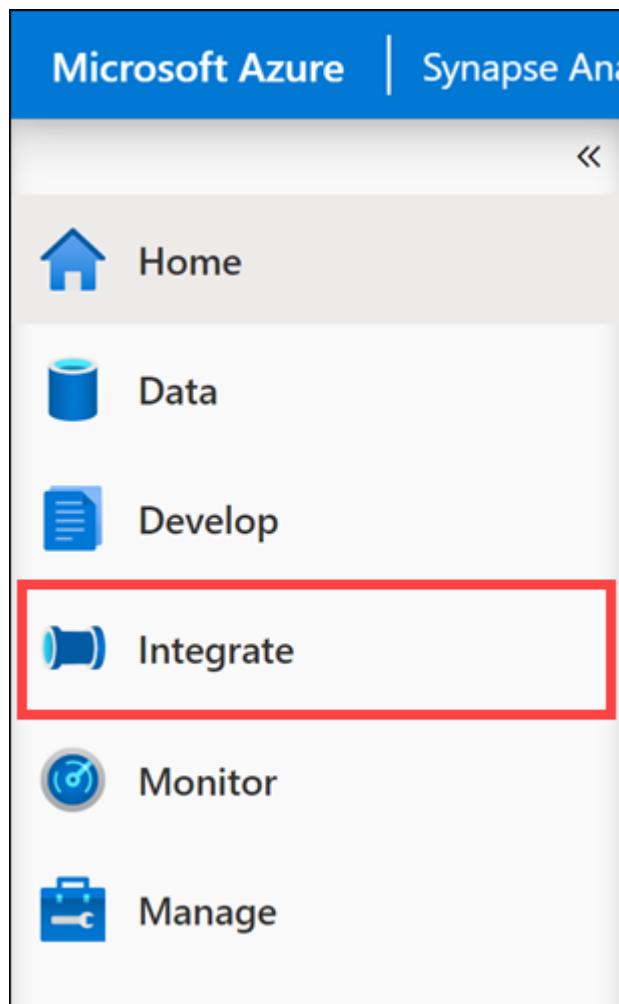
You recommend using Synapse Pipelines, which includes over 90 built-in connectors, can load data by manual execution of the pipeline or by orchestration, supports common loading patterns, enables fully parallel loading into the data lake or SQL tables, and shares a code base with ADF.

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.

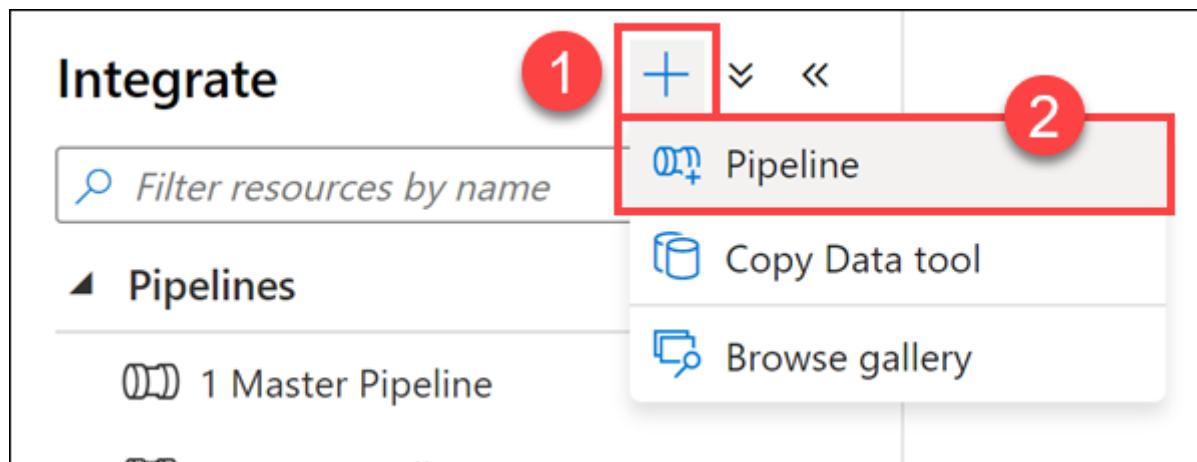
Task 1: Create pipeline

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.

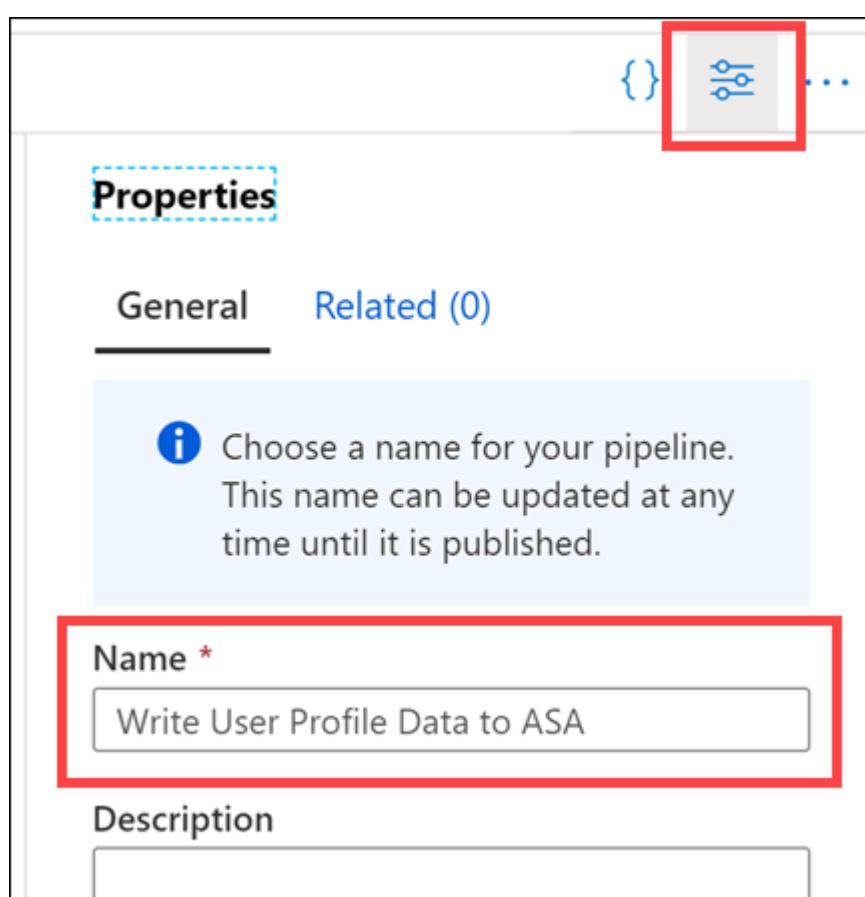
1. Navigate to the **Integrate** hub.



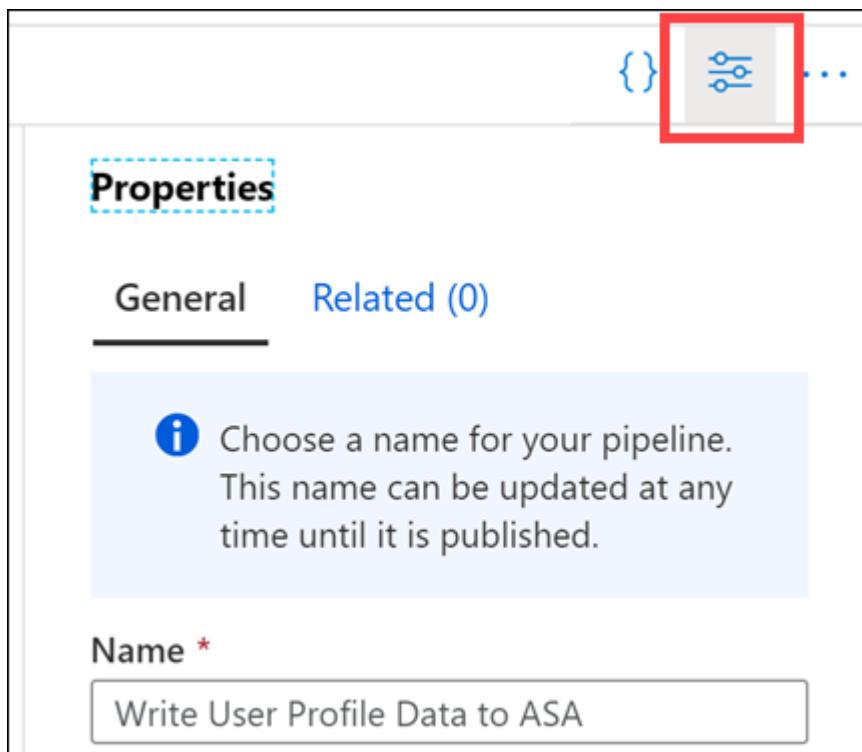
2. In the + menu, select **Pipeline**.



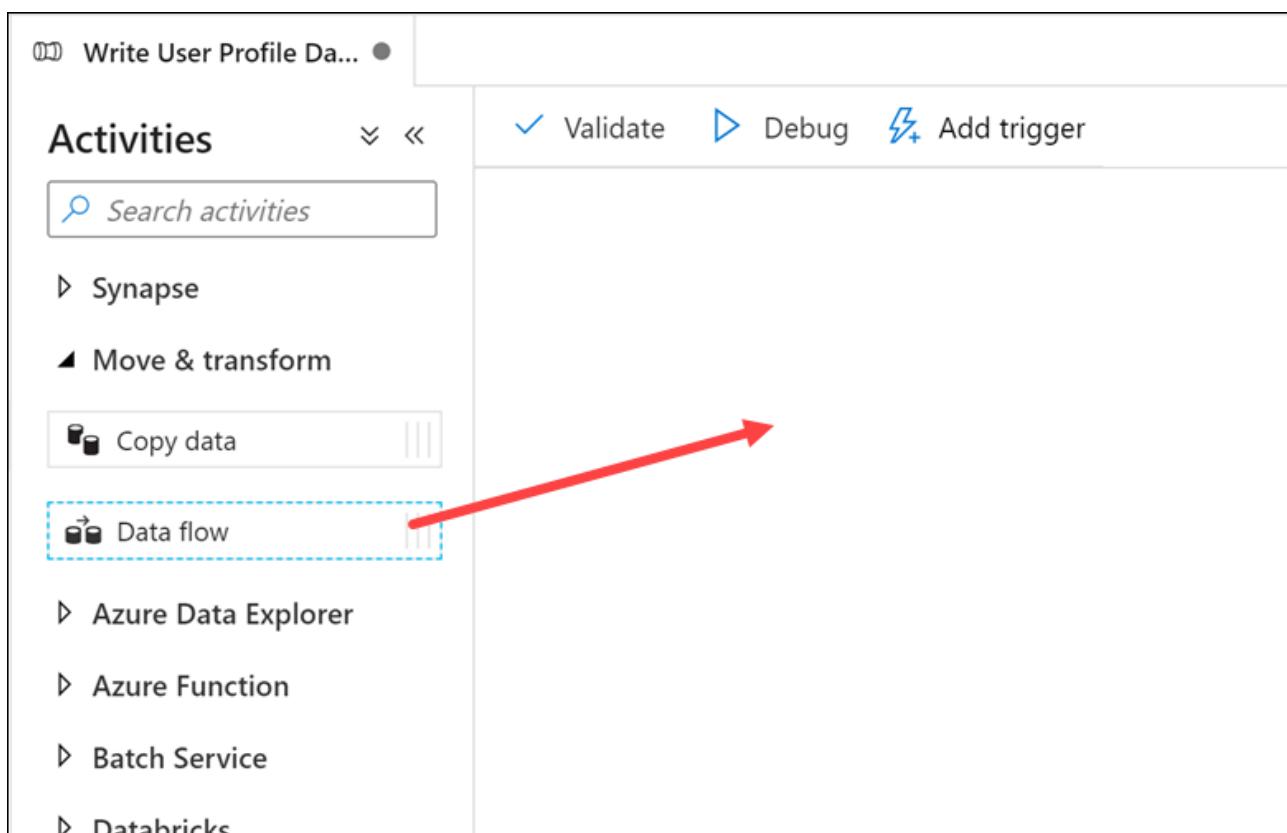
3. In the **General** section of the **Properties** pane of the new data flow, update the **Name** to **Write User Profile Data to ASA**.



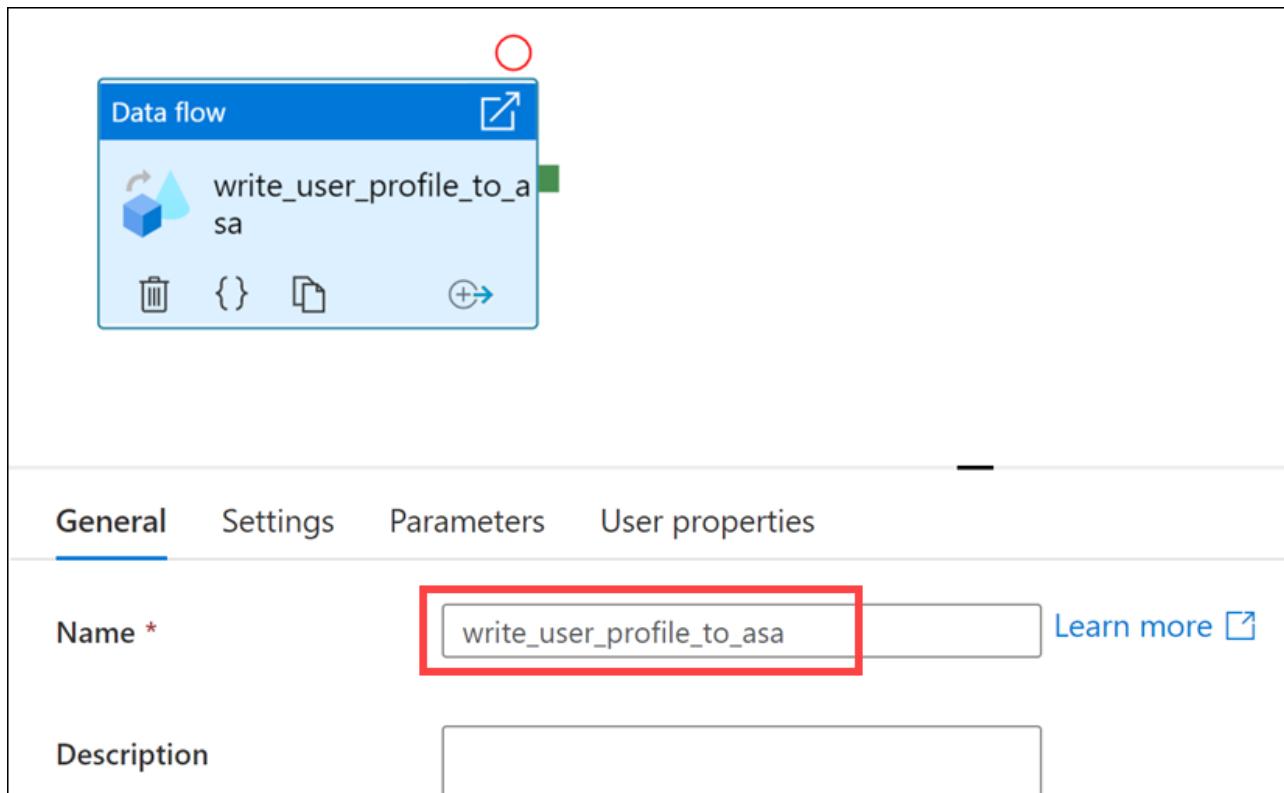
4. Select the **Properties** button to hide the pane.



5. Expand **Move & transform** within the Activities list, then drag the **Data flow** activity onto the pipeline canvas.



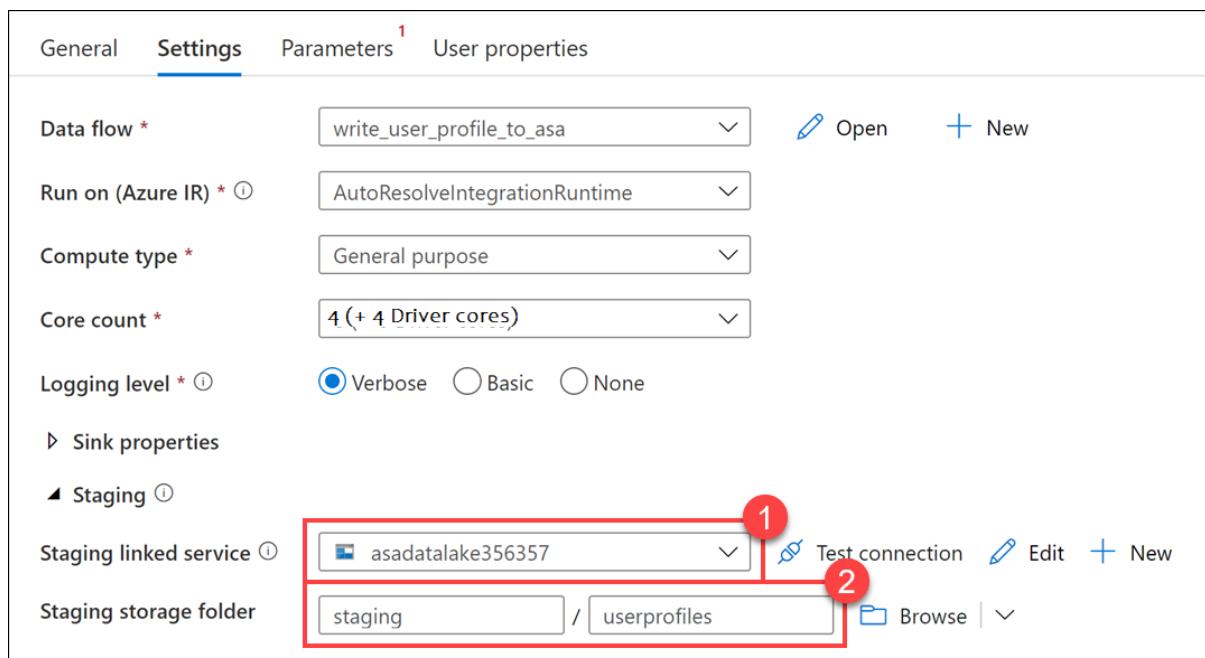
6. Under the **General** tab beneath the pipeline canvas set the **Name** to `write_user_profile_to_asa`.



7. On the **Settings** tab, select the **write_user_profile_to_asa** data flow, ensure **AutoResolveIntegrationRuntime** is selected. Choose the **Basic (General purpose)** compute type and set the core count to **4 (+ 4 Driver cores)**.

8. Expand **Staging** and configure the following:

- **Staging linked service:** Select the **asadatalakexxxxxxxx** linked service.
- **Staging storage folder:** Enter **staging / userprofiles** (the **userprofiles** folder will be automatically created for you during the first pipeline run).



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.

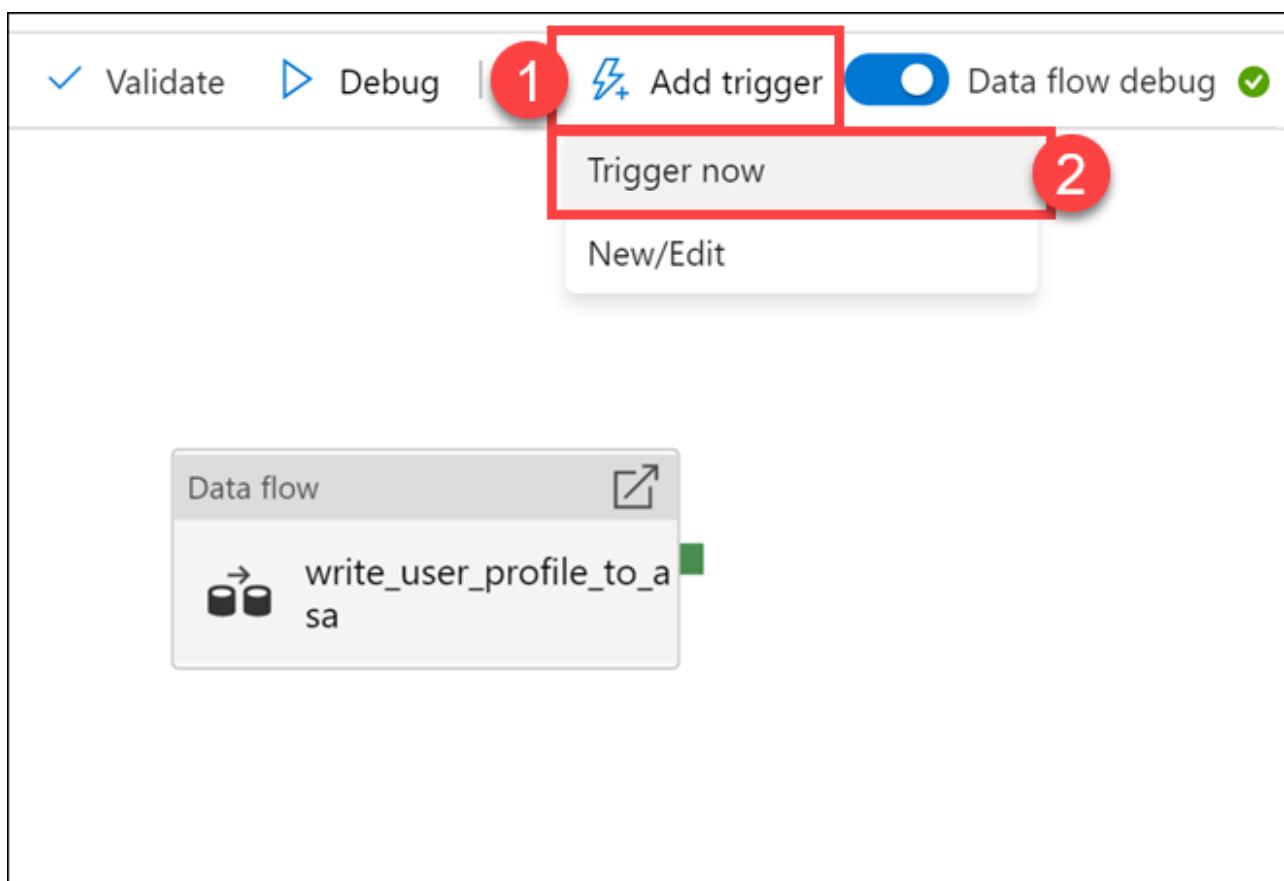


Task 2: Trigger, monitor, and analyze the user profile data pipeline

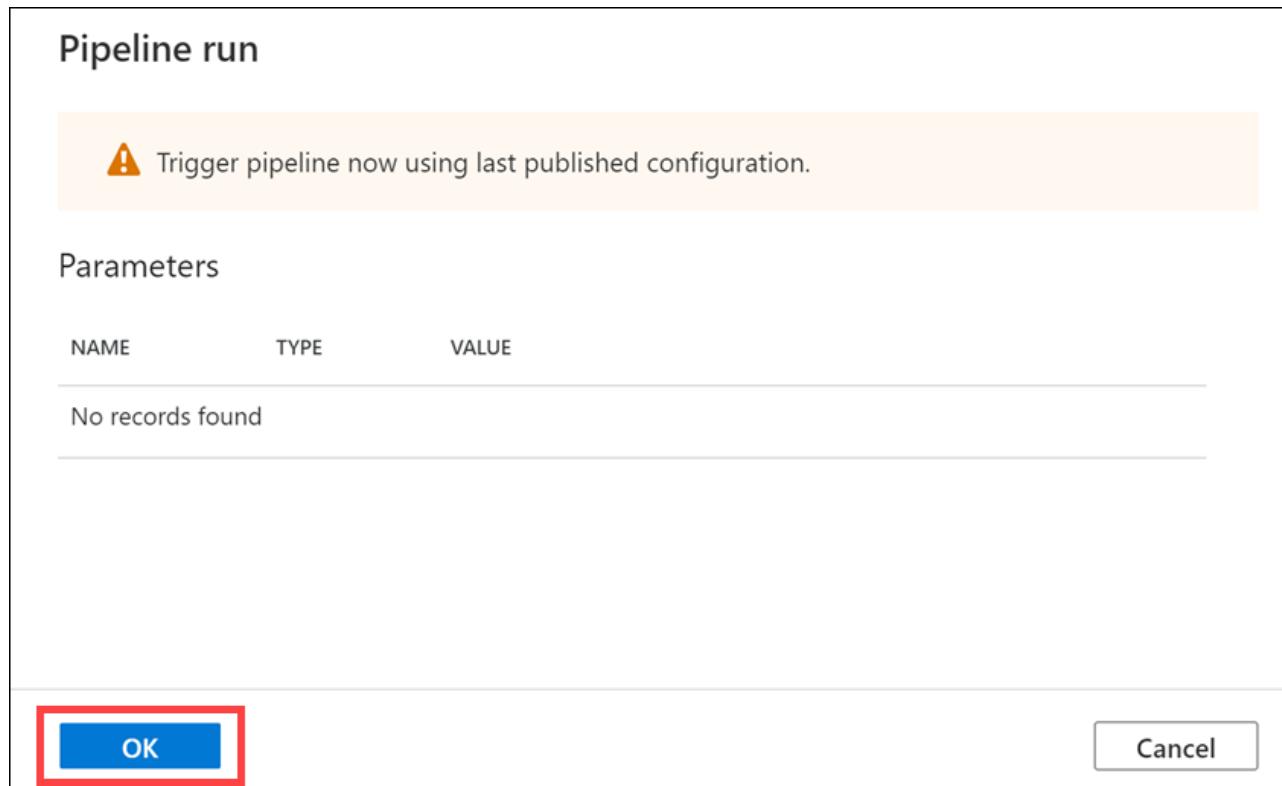
Tailwind Traders wants to monitor all pipeline runs and view statistics for performance tuning and troubleshooting purposes.

You have decided to show Tailwind Traders how to manually trigger, monitor, then analyze a pipeline run.

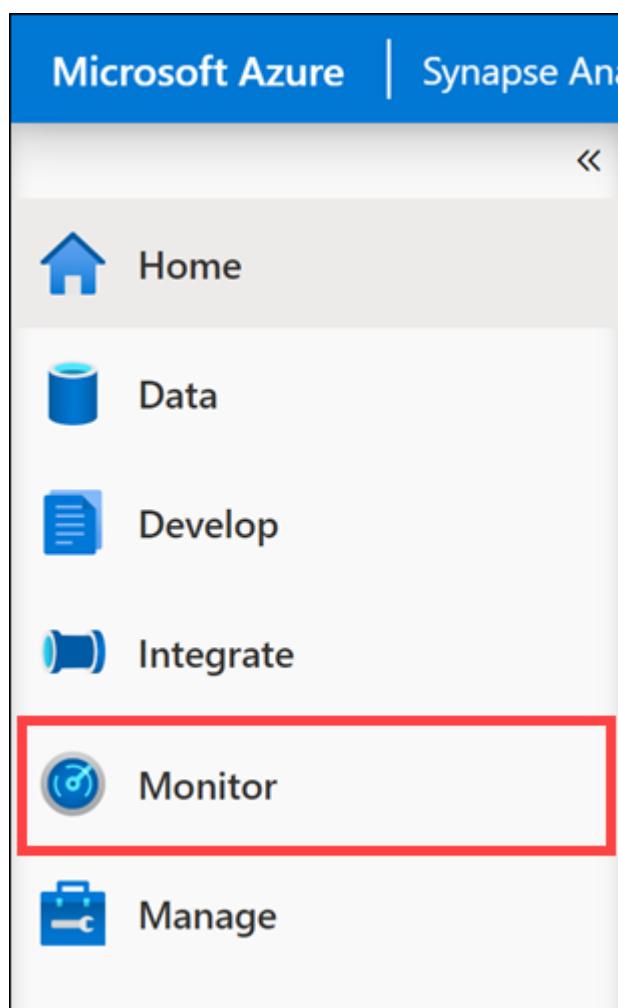
1. At the top of the pipeline, select **Add trigger**, then **Trigger now**.



2. There are no parameters for this pipeline, so select **OK** to run the trigger.



3. Navigate to the **Monitor** hub.



4. Select **Pipeline runs** and wait for the pipeline run to successfully complete, which may take some time.
You may need to refresh the view.

Pipeline runs

Triggered Debug Rerun Cancel Refresh 3 Edit columns List Gantt

Search by run ID or name Local time : Last 24 hours Pipeline name : All Status : All Runs : Latest runs Add filter Copy filters

Pipeline name	Run start ↑	Run end	Duration	Triggered by	Status	Error
User Profiles to Datalake	7/3/21, 4:56:50 PM	--	00:01:29	Manual trigger	In progress	
Write User Profile Data to ASA	7/3/21, 4:33:50 PM	7/3/21, 4:38:00 PM	00:04:09	Manual trigger	Succeeded	
Write Campaign Analytics to ...	7/3/21, 4:03:53 PM	7/3/21, 4:07:51 PM	00:03:58	Manual trigger	Succeeded	
Copy December Sales	7/3/21, 3:36:28 PM	7/3/21, 3:37:19 PM	00:00:50	Manual trigger	Succeeded	
Setup - Import User Profile Da...	7/2/21, 10:56:55 PM	7/2/21, 11:01:36 PM	00:04:40	Manual trigger	Succeeded	

5. Select the name of the pipeline to view the pipeline's activity runs.

Pipeline runs

Triggered Debug Rerun Cancel Refresh

Search by run ID or name Local time : Last 24 h

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.

Write User Profile Data to ASA

List Gantt

Rerun Rerun from activity Rerun from failed activity Refresh Edit pipeline

Data flow write_user_profile_to_asa

Activity runs

Pipeline run ID 0c17cf90-42e2-41fb-8ce5-35a203c66f28

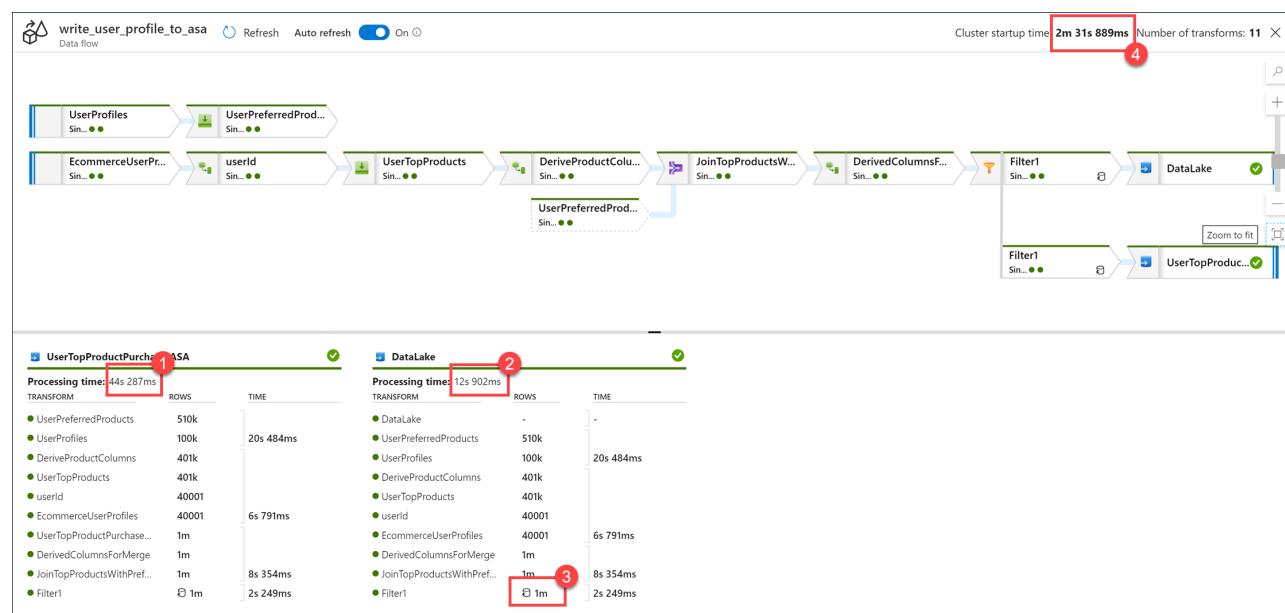
All status

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

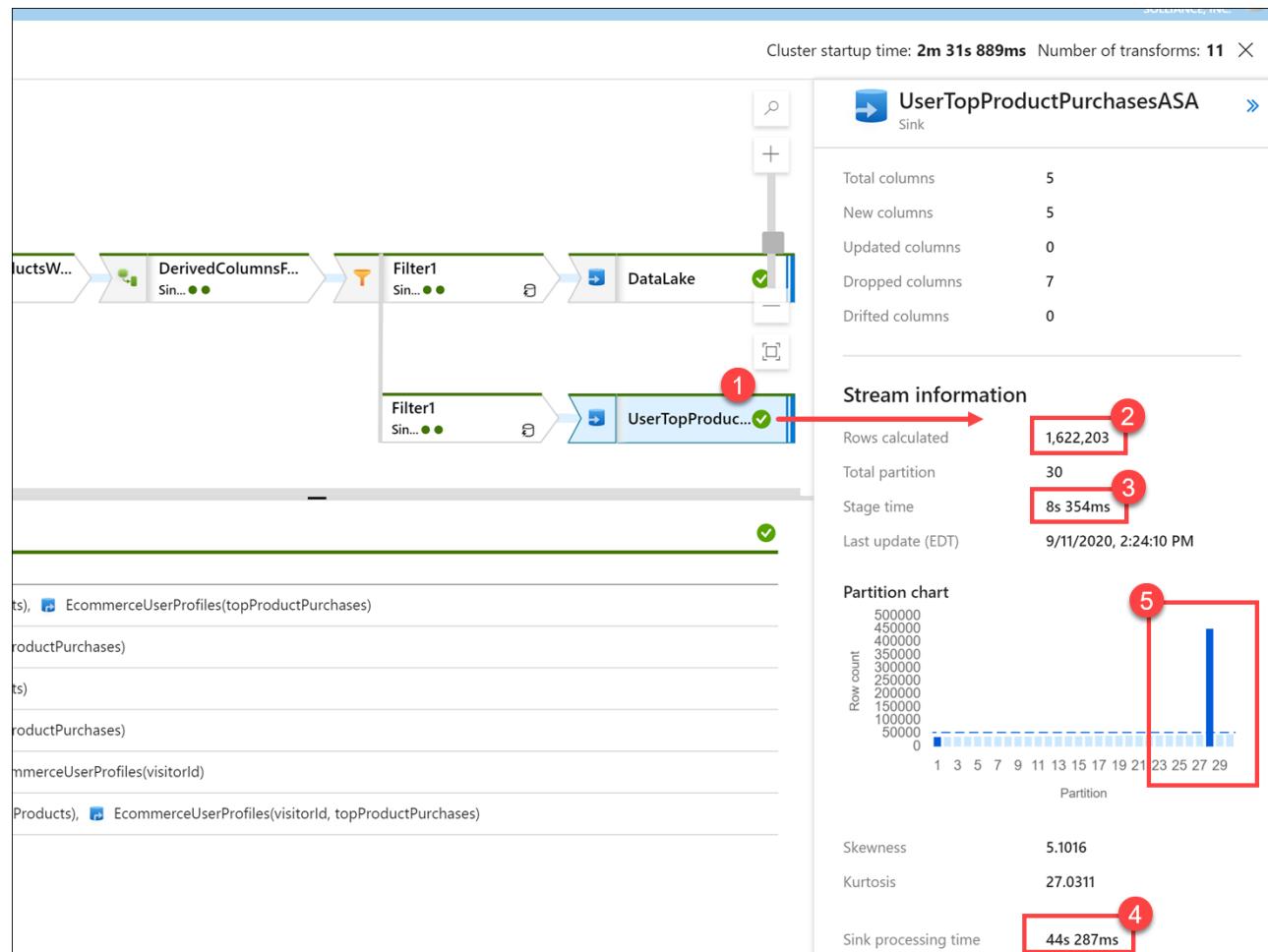
Data flow details

7. The data flow details displays the data flow steps and processing details. In the example below (which may be different from your results), processing time took around 44 seconds to process the SQL pool sink, and around 12 seconds to process the Data Lake sink. The **Filter1** output was around 1 million rows for both. You can see which activities took the longest to complete. The cluster startup time contributed over 2.5 minutes to the total pipeline run.



8. Select the **UserTopProductPurchasesASA** sink to view its details. In the example below (which may be different from your results), you can see that 1,622,203 rows were calculated with a total of 30 partitions. It took around eight seconds to stage the data 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* 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.



Important: Pause your SQL pool

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

1. In Synapse Studio, select the **Manage** hub.
2. Select **SQL pools** in the left-hand menu. Hover over the **SQLPool01** dedicated SQL pool and select **||**.

The screenshot shows the "Manage" hub in Azure Synapse Studio. On the left, a sidebar lists various categories: Analytics pools (with "SQL pools" highlighted by a red box and a red circle labeled "1"), External connections, Linked services, Azure Purview (Preview), Integration (with "Triggers" and "Integration runtimes" listed), and Security. The main area is titled "SQL pools" and contains the following content:

- SQL pools**: A brief description stating "The serverless SQL pool, Built-in, is immediately available for your workspace. Dedicated SQL pools can be configured".
- New**, **Refresh**, **System-assigned managed identity** buttons.
- Filter by name** input field.
- Showng 1-2 of 2 items (1 Serverless, 1 Dedicated)** message.
- Name**, **Type**, and **Status** table columns.
- A table listing two items:

Name	Type	Status
Built-in	Serverless	Online
SQLPool01	Dedicated	Online

 A red box labeled "2" surrounds the "Pause" button for the "SQLPool01" row.

3. When prompted, select **Pause**.