

Exercise - Integrate a Notebook within Azure Synapse Pipelines

18 minutes

In this unit, you create an Azure Synapse Spark notebook to analyze and transform data loaded by a mapping data flow, and store the data in a data lake. You create a parameter cell that accepts a string parameter that defines the folder name for the data the notebook writes to the data lake.

You then add this notebook to a Synapse pipeline, and pass the unique pipeline run ID to the notebook parameter so that you can later correlate the pipeline run with the data saved by the notebook activity.

Finally, you use the **Monitor** hub in Synapse Studio to monitor the pipeline run, obtain the run ID, and then locate the corresponding files stored in the data lake.

About Apache Spark and notebooks

Apache Spark is a parallel processing framework that supports in-memory processing to boost the performance of big-data analytic applications. Apache Spark in Azure Synapse Analytics is one of Microsoft's implementations of Apache Spark in the cloud.

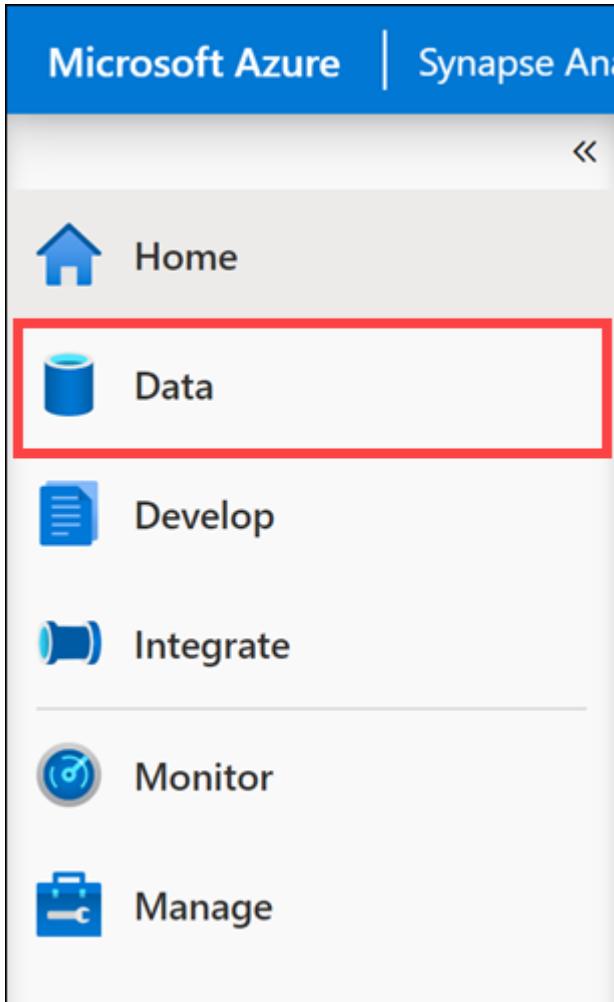
An Apache Spark notebook in Synapse Studio is a web interface for you to create files that contain live code, visualizations, and narrative text. Notebooks are a good place to validate ideas and use quick experiments to get insights from your data. Notebooks are also widely used in data preparation, data visualization, machine learning, and other Big Data scenarios.

Create a Synapse Spark notebook

Suppose you created a mapping data flow in Synapse Analytics to process, join, and import user profile data. Now, you want to find the top five products for each user, based on which ones are both preferred and top choice, and have the most purchases in the past 12 months. Then, you want to calculate the top five products overall.

In this exercise, you create a Synapse Spark notebook to make these calculations.

1. Open Synapse Analytics Studio (<https://web.azuresynthesize.net/>), and go to the Data hub.



2. Select the **Linked** tab (1), and expand the **primary data lake storage account** (2) underneath the **Azure Data Lake Storage Gen2**. Select the **wwi-02** container (3), and open the **top-products** folder (4). Right-click on any Parquet file (5), select the **New notebook** menu item (6), and then select **Load to DataFrame** (7). If you don't see the folder, select Refresh.

The screenshot shows the Azure Data Lake Storage Gen2 Data Explorer interface. The left sidebar lists workspaces, and the main area shows a folder structure under 'wwi-02'. A context menu is open over a specific Parquet file ('part-00000-80db06cf-0892-41c7-9c21-44bae30ec46c-c000.snappy.parquet'). The menu items are numbered 5 through 7:

5. New SQL script
6. New notebook
7. Load to DataFrame

3. Make sure the notebook is attached to your Spark pool.

The screenshot shows a Jupyter Notebook titled 'Notebook 1'. The toolbar includes a dropdown for attaching to a Spark pool, which is currently set to 'SparkPool01'. The code cell contains the following PySpark code:

```
Cell 1
[ ] 1 %%pyspark
2 df = spark.read.load('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top-products/part-00000-80
3 display(df.limit(10))
```

4. Replace the Parquet file name with *.parquet (1) to select all Parquet files in the top-products folder. For example, the path should be similar to: abfss://wwi-02@YOUR_DATA LAKE_NAME.dfs.core.windows.net/top-products/*.parquet.

The screenshot shows the same Jupyter Notebook cell after the update. The code now reads all Parquet files in the 'top-products' folder:

```
Cell 1
[ ] 1 %%pyspark
2 df = spark.read.load('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top-products/*.parquet', format='parquet')
3 display(df.limit(10))
```

5. Select Run all on the notebook toolbar to execute the notebook.

Cell 1

```
1 %%pyspark
2 df = spark.read.load('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top-products/*.parquet')
3 display(df.limit(10))
```

Command executed in 2mins 35s 588ms by joel on 11-26-2020 00:53:24.571 -05:00

> Job execution Succeeded Spark 2 executors 8 cores

[View in monitoring](#) [Open Spark UI](#)

View [Table](#) [Chart](#)

visitorId	productId	itemsPurchased...	preferredProduc...	userId	isTopProduct	isPreferredProd...
2717		2717	148	false	true	
4002		4002	148	false	true	
1716		1716	148	false	true	
4520		4520	148	false	true	
951		951	148	false	true	
1817		1817	148	false	true	
2634		2634	463	false	true	
2795		2795	463	false	true	

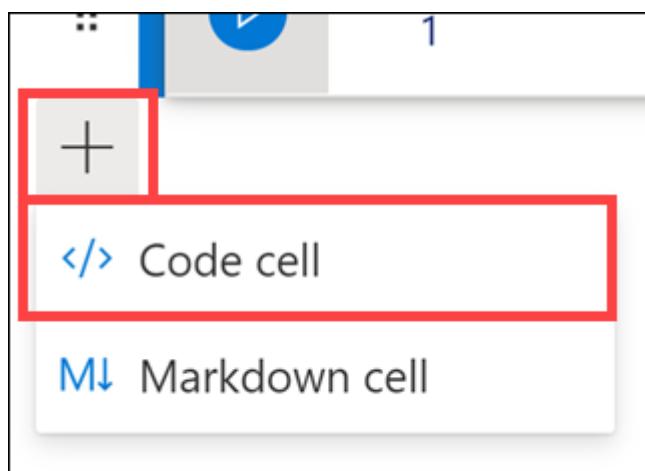
ⓘ Note

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

ⓘ Note

To run just the cell, either hover over the cell and select the *Run cell* icon to the left of the cell, or select the cell, then enter **Ctrl+Enter**.

6. Create a new cell underneath by selecting the + button, and selecting the **Code cell** item. The + button is located beneath the notebook cell on the left. Alternatively, you can also expand the + Cell menu in the Notebook toolbar, and select the **Code cell** item.



7. Run the following command in the new cell to populate a new dataframe called `topPurchases`, create a new temporary view named `top_purchases`, and show the first 100 rows:

```
Python
```

```
topPurchases = df.select(  
    "UserId", "ProductId",  
    "ItemsPurchasedLast12Months", "IsTopProduct",  
    "IsPreferredProduct")  
  
# Populate a temporary view so we can query from SQL  
topPurchases.createOrReplaceTempView("top_purchases")  
  
topPurchases.show(100)
```

The output should look similar to the following:

```
text
```

```
+-----+-----+-----+-----+-----+  
|-----+-----+-----+-----+-----+  
|UserId|ProductId|ItemsPurchasedLast12Months|IsTopProduct|IsPreferred-  
|-----+-----+-----+-----+-----+  
|-----+-----+-----+-----+-----+  
| 148| 2717| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 148| 4002| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 148| 1716| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 148| 4520| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 148| 951| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 148| 1817| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 463| 2634| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 463| 2795| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 471| 1946| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 471| 4431| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 471| 566| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 471| 2179| null| false|  
| true| |-----+-----+-----+-----+-----+  
| 471| 3758| null| false|  
| true| |-----+-----+-----+-----+-----+
```

471	2434	null	false
true			
471	1793	null	false
true			
471	1620	null	false
true			
471	1572	null	false
true			
833	957	null	false
true			
833	3140	null	false
true			
833	1087	null	false
true			

8. Run the following command in a new cell to create a new temporary view by using SQL:

SQL

%%sql

```
CREATE OR REPLACE TEMPORARY VIEW top_5_products
AS
    select UserId, ProductId, ItemsPurchasedLast12Months
    from (select *,
                row_number() over (partition by UserId order by Items-
PurchasedLast12Months desc) as seqnum
            from top_purchases
        ) a
    where seqnum <= 5 and IsTopProduct == true and IsPreferredProduct
= true
    order by a.UserId
```

① Note

There is no output for this query.

The query uses the `top_purchases` temporary view as a source and applies a `row_number()` over method to apply a row number for the records for each user where `ItemsPurchasedLast12Months` is greatest. The `where` clause filters the results so we only retrieve up to five products where both `IsTopProduct` and `IsPreferredProduct` are set to true. This gives us the top five most purchased products for each user where those products are *also* identified as their favorite products, according to their user profile stored in Azure Cosmos DB.

9. Run the following command in a new cell to create and display a new DataFrame that stores the results of the `top_5_products` temporary view you created in the previous cell:

Python

```
top5Products = sqlContext.table("top_5_products")  
top5Products.show(100)
```

You should see an output similar to the following, which displays the top five preferred products per user:

Cell 5

```
[26] 1 top5Products = sqlContext.table("top_5_products")  
2  
3 top5Products.show(100)
```

UserId	ProductId	ItemsPurchasedLast12Months
80000	2069	93
80000	2069	93
80000	2069	93
80000	2069	93
80000	2069	93
80000	2069	93
80001	1812	93
80001	1812	93
80001	1812	93
80001	1812	93
80001	1812	93
80002	1256	90
80002	1256	90
80002	4987	88
80002	3190	92
80002	3190	92
80003	295	91
80003	638	97
80003	638	97

10. Calculate the top five products overall, based on those that are both preferred by customers and purchased the most. To do this, run the following command in a new cell:

Python

```
top5ProductsOverall = (top5Products.select("ProductId", "ItemsPurchasedLast12Months")  
.groupBy("ProductId")  
.agg( sum("ItemsPurchasedLast12Months").alias("Total") )  
.orderBy( col("Total").desc() )  
.limit(5))  
  
top5ProductsOverall.show()
```

In this cell, we grouped the top five preferred products by product ID, summed up the total items purchased in the last 12 months, sorted that value in descending order, and returned the top five results. Your output should be similar to the following:

text

ProductId	Total
2107	4538
4833	4533
347	4523
3459	4233
4246	4155

Create a parameter cell

Azure Synapse pipelines look for the parameters cell, and treat this cell as defaults for the parameters passed in at execution time. The execution engine will add a new cell beneath the parameters cell with input parameters to overwrite the default values. When a parameters cell isn't designated, the injected cell will be inserted at the top of the notebook.

1. We are going to execute this notebook from a pipeline. We want to pass in a parameter that sets a `runId` variable value that will be used to name the Parquet file. Run the following command in a new cell:

Python

```
import uuid

# Generate random GUID
runId = uuid.uuid4()
```

We are using the `uuid` library that comes with Spark to generate a random GUID. We want to override the `runId` variable with a parameter passed in by the pipeline. To do this, we need to toggle this as a parameter cell.

2. Select the actions ellipses (...) on the top-right corner of the cell (1), and then select **Toggle parameter cell** (2).

The screenshot shows a Jupyter Notebook cell labeled "Cell 8" containing Python code to generate a random UUID. A context menu is open over the cell, with the "Parameters" option highlighted by a red box and the number "2". Other options in the menu include "Run cell" (Ctrl+Enter), "Run cells above", "Run cells below", "Move cell up" (Ctrl+Shift+↑), "Move cell down" (Ctrl+Shift+↓), "Collapse output", "Delete cell", and "D, D".

```

7 top5ProductsOverall.show()

+-----+-----+
|ProductId|Total|
+-----+-----+
| 2107 | 4538 |
| 347 | 4523 |
| 4833 | 4377 |
| 3459 | 4233 |
| 2486 | 4135 |
+-----+-----+

```

```

Cell 8
1 import uuid
2
3 # Generate random GUID
4 runId = uuid.uuid4()

```

After toggling this option, you will see the **Parameters** tag on the cell.

The screenshot shows the same Jupyter Notebook cell as before, but now the "Parameters" tag is visible at the top right of the cell area, indicated by a red box and the number "2".

```

Cell 8
1 import uuid
2
3 # Generate random GUID
4 runId = uuid.uuid4()

```

3. Paste the following code in a new cell to use the `runId` variable as the Parquet filename in the `/top5-products/` path in the primary data lake account. Replace `YOUR_DATALAKE_NAME` in the path with the name of your primary data lake account. To find this, scroll up to **Cell 1** at the top of the page (1). Copy the data lake storage account from the path (2). Paste this value as a replacement for `YOUR_DATALAKE_NAME` in the path (3) inside the new cell, then run the command in the cell.

The screenshot shows a Jupyter Notebook cell with the following code:

```

Python
%%pyspark

top5ProductsOverall.write.parquet('abfss://wwi-02@YOUR_DATALAKE_-
NAME.dfs.core.windows.net/top5-products/' + str(runId) + '.parquet')

```

The screenshot shows three Jupyter Notebook cells. Cell 1 (top) contains the %%pyspark magic command and the write parquet command with a placeholder path. Cell 2 (middle) shows the path with a yellow highlight and the number "2", indicating the part to copy. Cell 3 (bottom) shows the path with a yellow highlight and the number "3", indicating where to paste the copied value. Red arrows point from the highlighted text in Cell 2 to the placeholder in Cell 3.

```

Cell 1
[22] 1 %%pyspark
2 data_path = spark.read.load('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top-products/*.parquet', format='parquet')
3 display(data_path.limit(10))

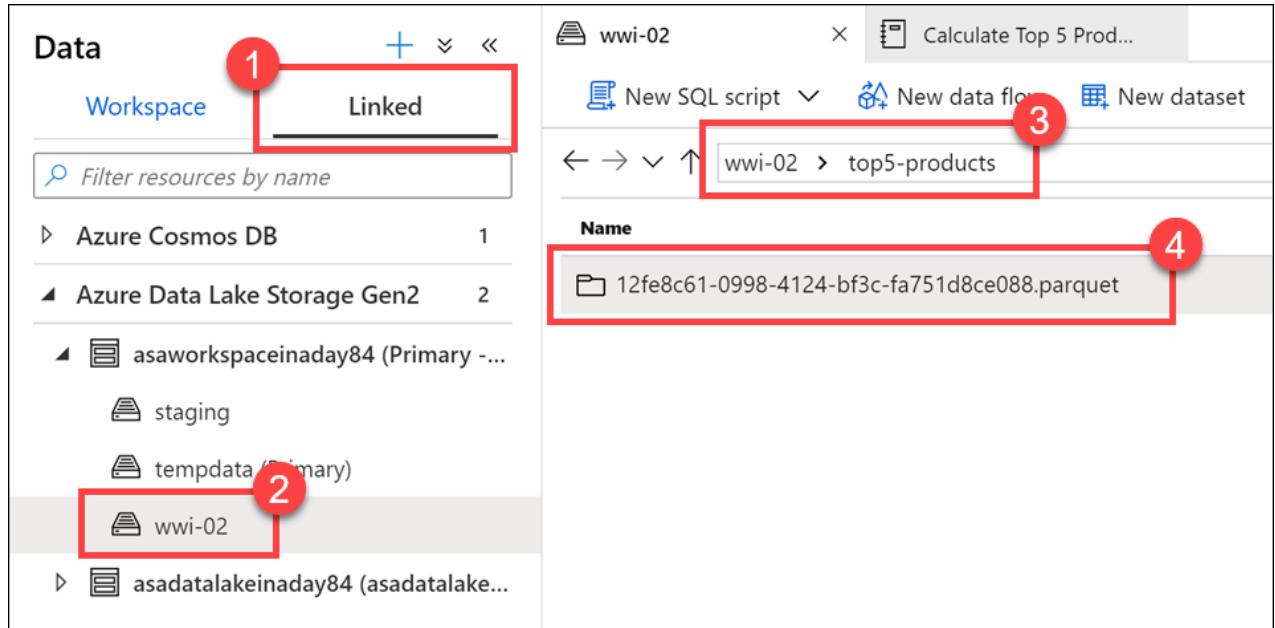
[29] 1 import uuid
2
3 # Generate random GUID
4 runId = uuid.uuid4()

Cell 9
1 top5ProductsOverall.write.parquet('abfss://wwi-02@asadatalakeinaday84.dfs.core.windows.net/top5-products/' + str(runId) ...

```

4. Verify that the file was written to the data lake. Go to the **Data** hub, and select the **Linked** tab (1). Expand the primary data lake storage account, and then select the `wwi-02`

container (2). Go to the top5-products folder (3). You should see a folder for the Parquet file in the directory with a GUID as the filename (4).

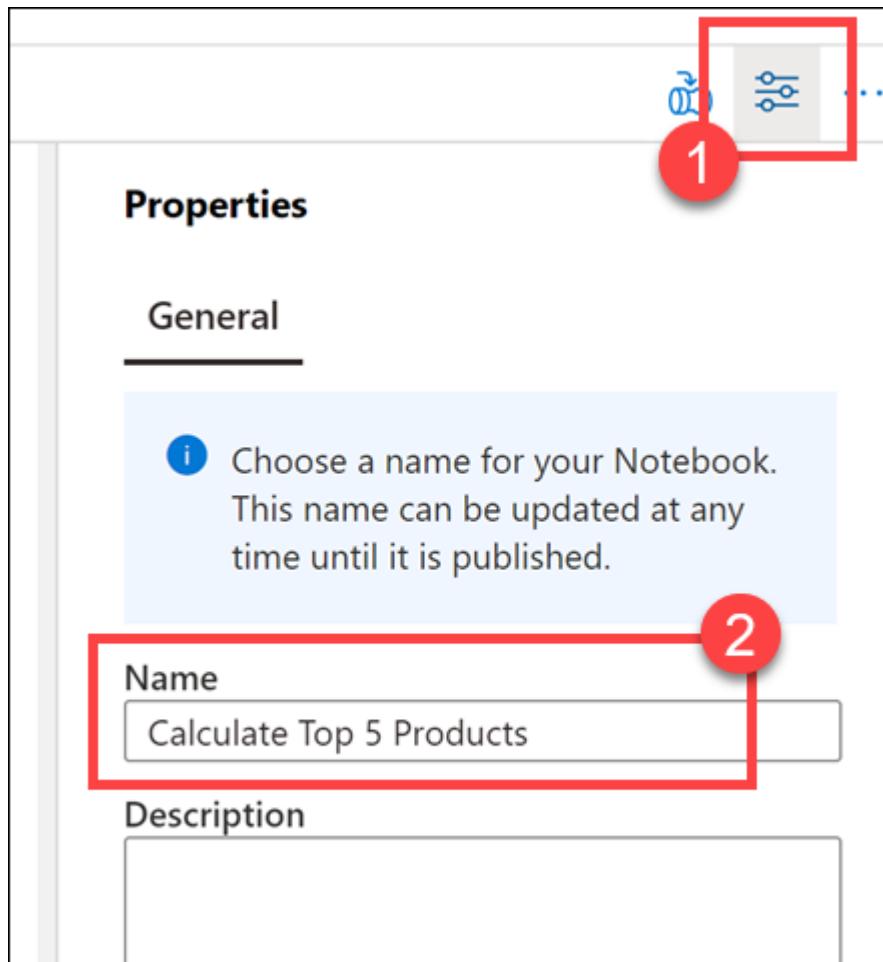


The Parquet write method on the dataframe in the Notebook cell created this directory because it did not previously exist.

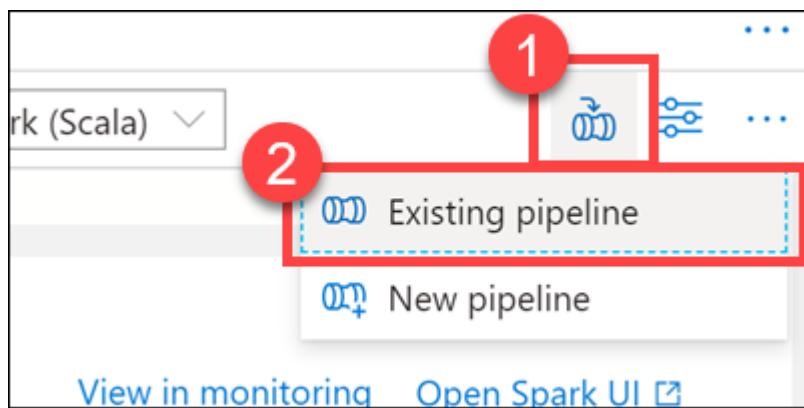
Add the Notebook to a Synapse pipeline

Referring back to the Mapping Data Flow we described at the beginning of the exercise, suppose you want to execute this notebook after the Data Flow runs as part of your orchestration process. To do this, you add this notebook to a pipeline as a new Notebook activity.

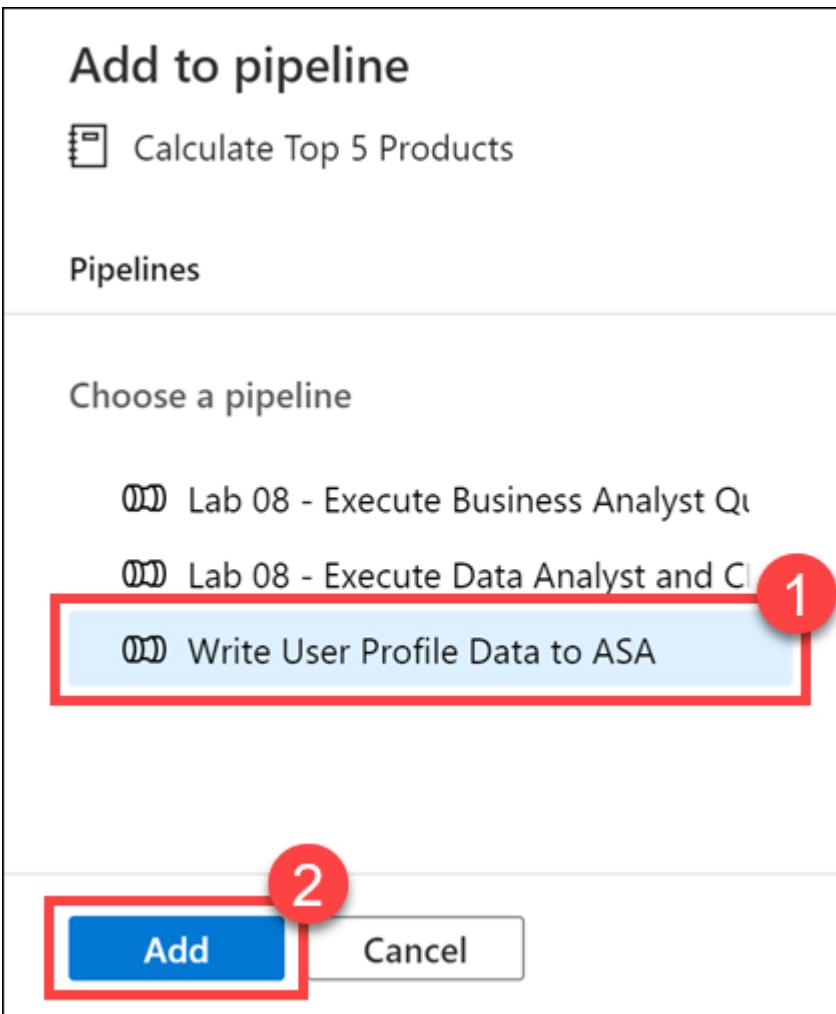
1. Return to the notebook. Select **Properties** (1) at the top-right corner of the notebook, and then enter Calculate Top 5 Products for the **Name** (2).



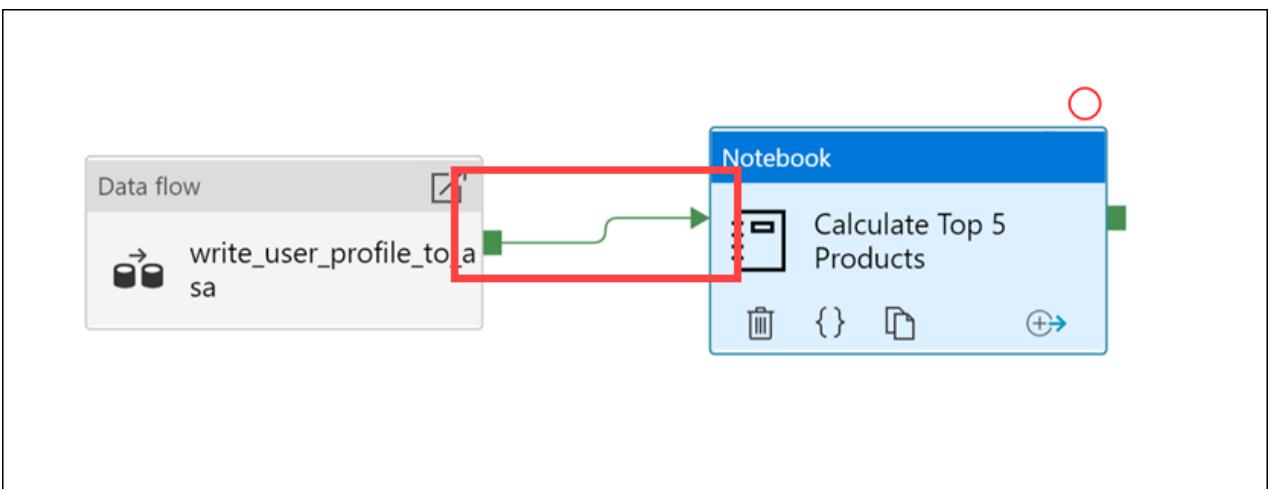
2. Select **Add to pipeline** (1) at the top-right corner of the notebook, and then select **Existing pipeline** (2).



3. Select the **Write User Profile Data to ASA** pipeline (1), and then select **Add ***(2).

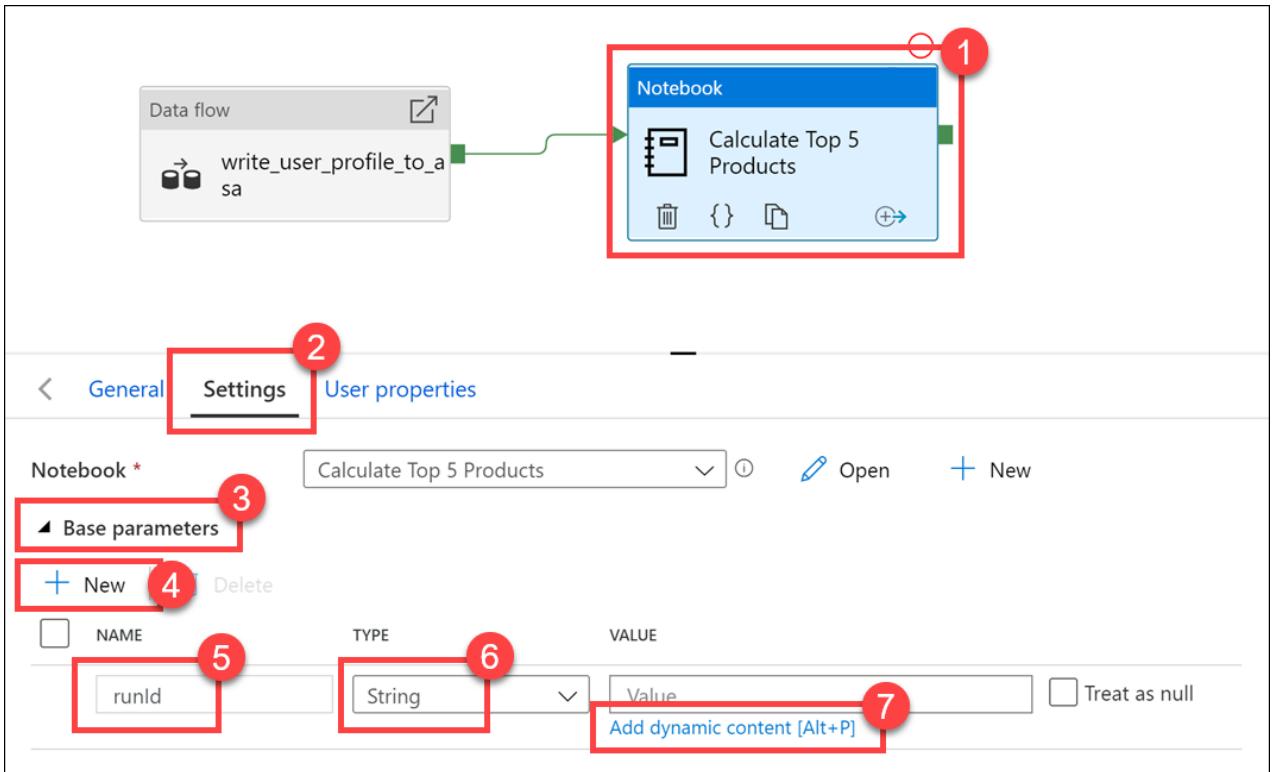


4. Synapse Studio adds the Notebook activity to the pipeline. Rearrange the **Notebook activity** so it sits to the right of the **Data flow activity**. Select the **Data flow activity**, and drag a **Success** activity pipeline connection **green box** to the **Notebook activity**.



The Success activity arrow instructs the pipeline to run the Notebook activity after the Data flow activity successfully runs.

5. Select the **Notebook activity** (1), then select the **Settings** tab (2), expand **Base parameters** (3), and then select **+ New** (4). Enter **runId** in the **Name** field (5). Select **String** for the **Type** (6). For the **Value**, select **Add dynamic content** (7).



6. Select Pipeline run ID under System variables (1). This adds `@pipeline().RunId` to the dynamic content box (2). Select Finish (3) to close the dialog box.

Add dynamic content

The Pipeline run ID value is a unique GUID assigned to each pipeline run. We will use this value for the name of the Parquet file by passing this value in as the `runId` Notebook parameter. We can then look through the pipeline run history and find the specific Parquet file created for each pipeline run.

7. Select **Publish all** then **Publish** to save your changes.

2

Clear contents

Filter...

+

Use [expressions](#), [functions](#) or refer to [system variables](#).

▲ System variables

- Pipeline Name
Name of the pipeline
- Pipeline run ID**
ID of the specific pipeline run
- Pipeline trigger ID
ID of the trigger that invokes the pipeline
- Pipeline trigger name
Name of the trigger that invokes the pipeline
- Pipeline trigger time
Time when the trigger that invoked the pipeline. The trigger time is the actual fired time, not the sc...
- Pipeline trigger type
Type of the trigger that invoked the pipeline (Manual, Scheduler)
- Workspace name
Name of the workspace the pipeline run is running within

▲ Functions

▼ [Expand all](#)

▷ [Collection Functions](#)

3

Finish

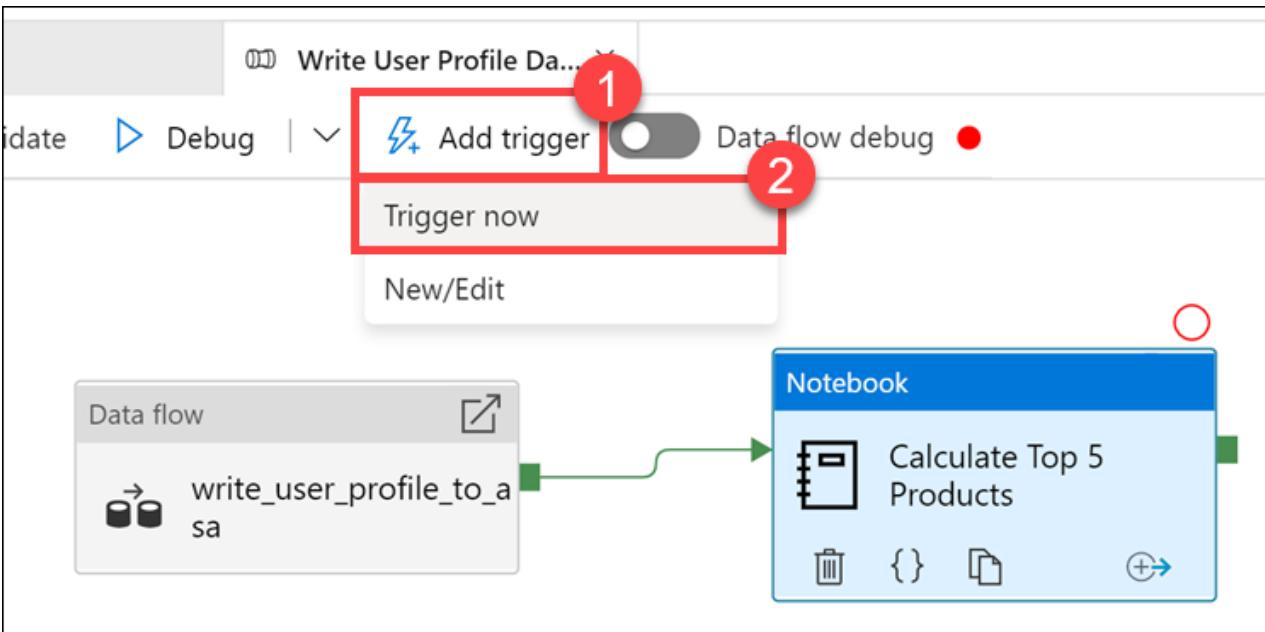
Cancel

The Pipeline run ID value is a unique GUID assigned to each pipeline run. We will use this value for the name of the Parquet file by passing this value in as the `runId` Notebook parameter. We can then look through the pipeline run history and find the specific Parquet file created for each pipeline run.

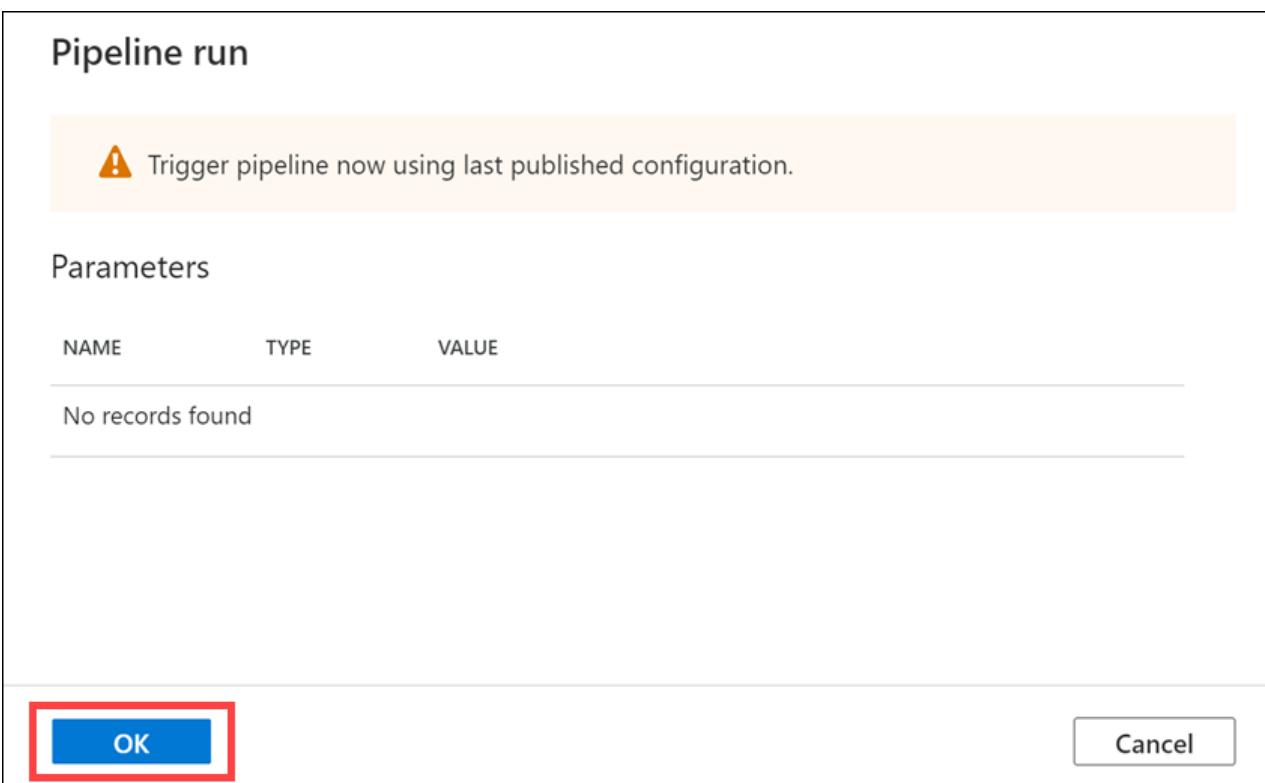
7. Select **Publish all** then **Publish** to save your changes.



8. After publishing is complete, select **Add trigger** (1), then **Trigger now** (2) to run the updated pipeline.



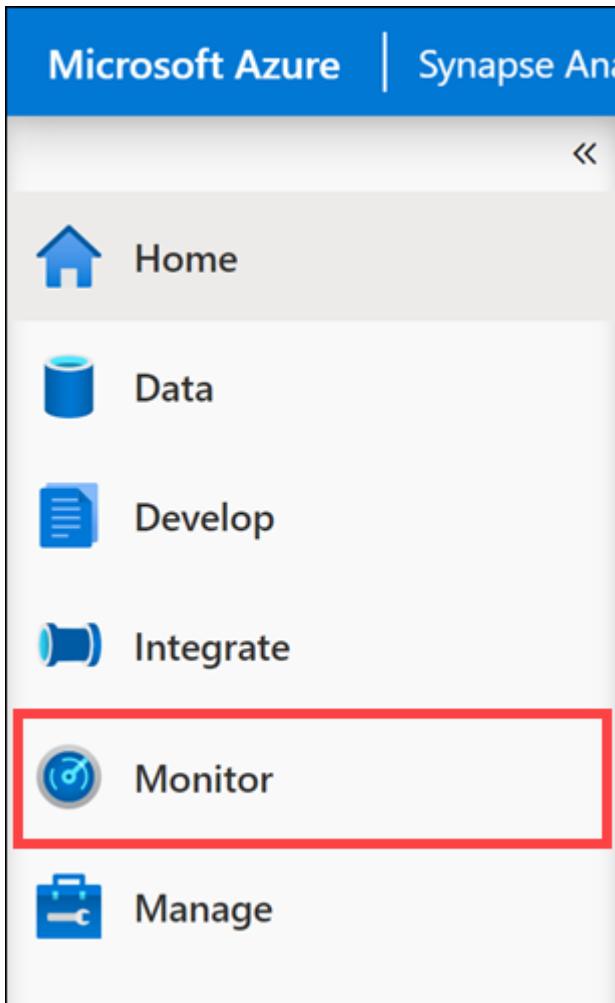
9. Select **OK** to run the trigger.



Monitor the pipeline run

The **Monitor** hub lets you monitor current and historical activities for SQL, Apache Spark, and Pipelines.

1. Go to the **Monitor** hub.



2. Select **Pipeline runs** (1), and wait for the pipeline run to successfully complete (2). You may need to refresh (3) the view.

A screenshot of the 'Pipeline runs' page. On the left is a sidebar with 'Integration' (selected), 'Pipeline runs' (1, highlighted with a red box and a red number), 'Trigger runs', 'Integration runtimes', 'Activities', 'Apache Spark applications', 'SQL requests', and 'Data flow debug'. The main area is titled 'Pipeline runs' and shows a table of runs. The table has columns: Pipeline name, Run start, Run end, Duration, Triggered by, Status, and Run. There are two rows of data:

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Run
Write User Profile Data t...	11/26/20, 3:33:48 AM	11/26/20, 3:45:43 AM	00:11:55	Manual trigger	✓ Succeeded	Original
Write User Profile Data t...	11/26/20, 3:16:33 AM	11/26/20, 3:21:26 AM	00:04:53	Manual trigger	✓ Succeeded	Original

At the top of the page, there are filters for 'Triggered', 'Debug', 'Rerun', 'Cancel', 'Refresh' (3, highlighted with a red box and a red number), 'List' (selected), 'Gantt', and search fields for 'Search by run ID or name' and 'Eastern Time (US & C... : Last 24 hours)'. There are also buttons for 'Pipeline name : All', 'Status : All', 'Runs : Latest runs', and 'Copy filters'.

3. 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

Eastern Time (US & C... : Last 24 h

Add filter

Showing 1 - 11 items



Pipeline name

Run start ↑↓



Write User Profile Data to ASA

11/26/20, 3:33:48 AM

- Notice both the Data flow activity and the new Notebook activity (1). Make note of the Pipeline run ID value (2). We will compare this to the Parquet file name generated by the notebook. Select the Calculate Top 5 Products Notebook name to view its details (3).

Write User Profile Data to ASA

List Gantt

Rerun Rerun from activity Rerun from failed activity Refresh

Activity runs

Pipeline run ID 16bb8447-824a-4147-b7ea-348505ebdc44

All status

Showing 1 - 2 of 2 items

Activity name	Activity type	Run start ↑	Duration	Status	Integration runtime
Calculate Top 5 Products	SynapseNotebook	11/26/20, 3:39:57 AM	00:05:45	Succeeded	DefaultIntegrationRuntime (East US 2)
write_user_profile_to_asa	ExecuteDataFlow	11/26/20, 3:33:50 AM	00:06:07	Succeeded	DefaultIntegrationRuntime (East US 2)

- Here, we see the Notebook run details. You can select **Playback** (1) to watch a playback of the progress through the jobs (2). At the bottom, you can view the **Diagnostics and Logs** with different filter options (3). To the right, we can view the run details, such as the

duration, Livy ID, Spark pool details, and so on. Select the **View details** link on a job to view its details (5).

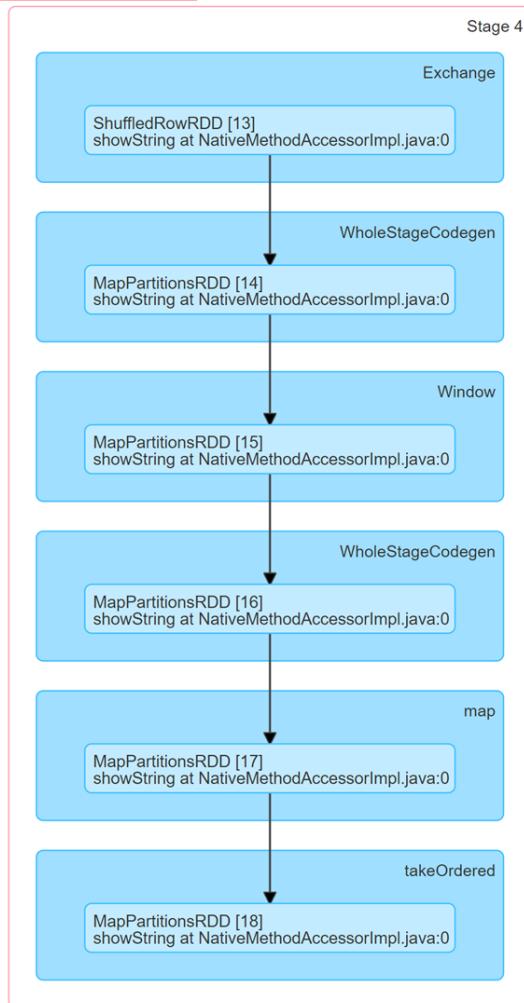
The screenshot shows the Spark Application UI. At the top, it displays the job ID **bf5a1022-c80d-4f98-b60f-7b2ce74606ff**, completed tasks (2562 of 2562), status (Stopped), and total duration (5m 45s). Below this is a toolbar with **Cancel**, **Refresh**, and **Spark history server** buttons. The main area is titled "Attempts 1 of 1" and shows a "Progress" bar at 0ms / 2min 5s 494ms. A red box labeled **1** highlights the **Playback** button. Stage details for Stages 0 through 5 are listed below, each with a "View details" link. Stage 4 is expanded, showing its tasks and metrics, with a red box labeled **5** highlighting the "View details" link. To the right, a sidebar titled "Details" contains sections for **Summary**, **Application** (with Application ID **application_1606380084300_0001**), **Queued duration** (0s), **Running duration** (5m 45s), **Livy ID** (4), **Submitter** (ee20d9e7-6295-4240-ba3f-c3784616c565), **Submit time** (11/26/20 3:39:58 AM), **Executors** (2), and **Spark pool** (Name: SparkPool01). At the bottom left, a "Diagnostics" section is open, showing items like Failed jobs, Data skew, Time skew, and Executor utilization, each with a green checkmark icon. A red box labeled **3** highlights the "Logs" tab.

6. The Spark application UI opens in a new tab where we can see the stage details. Expand the **DAG Visualization** to view the stage details.

Details for Stage 4 (Attempt 0)

Total Time Across All Tasks: 18 s
Locality Level Summary: Node local: 200
Shuffle Read: 10.1 MB / 1622203

▼ DAG Visualization

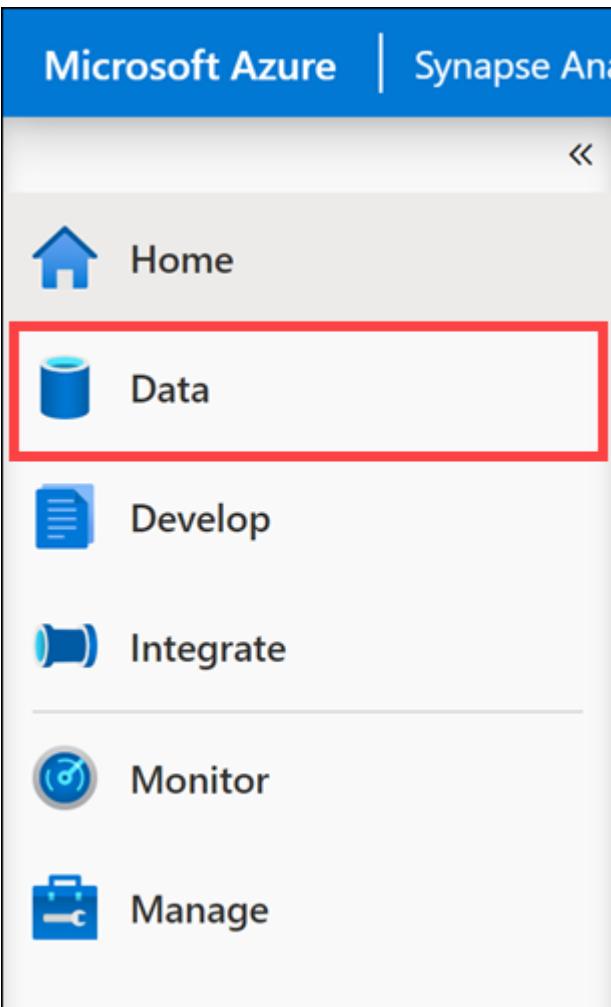


- ▶ Show Additional Metrics
- ▶ Event Timeline

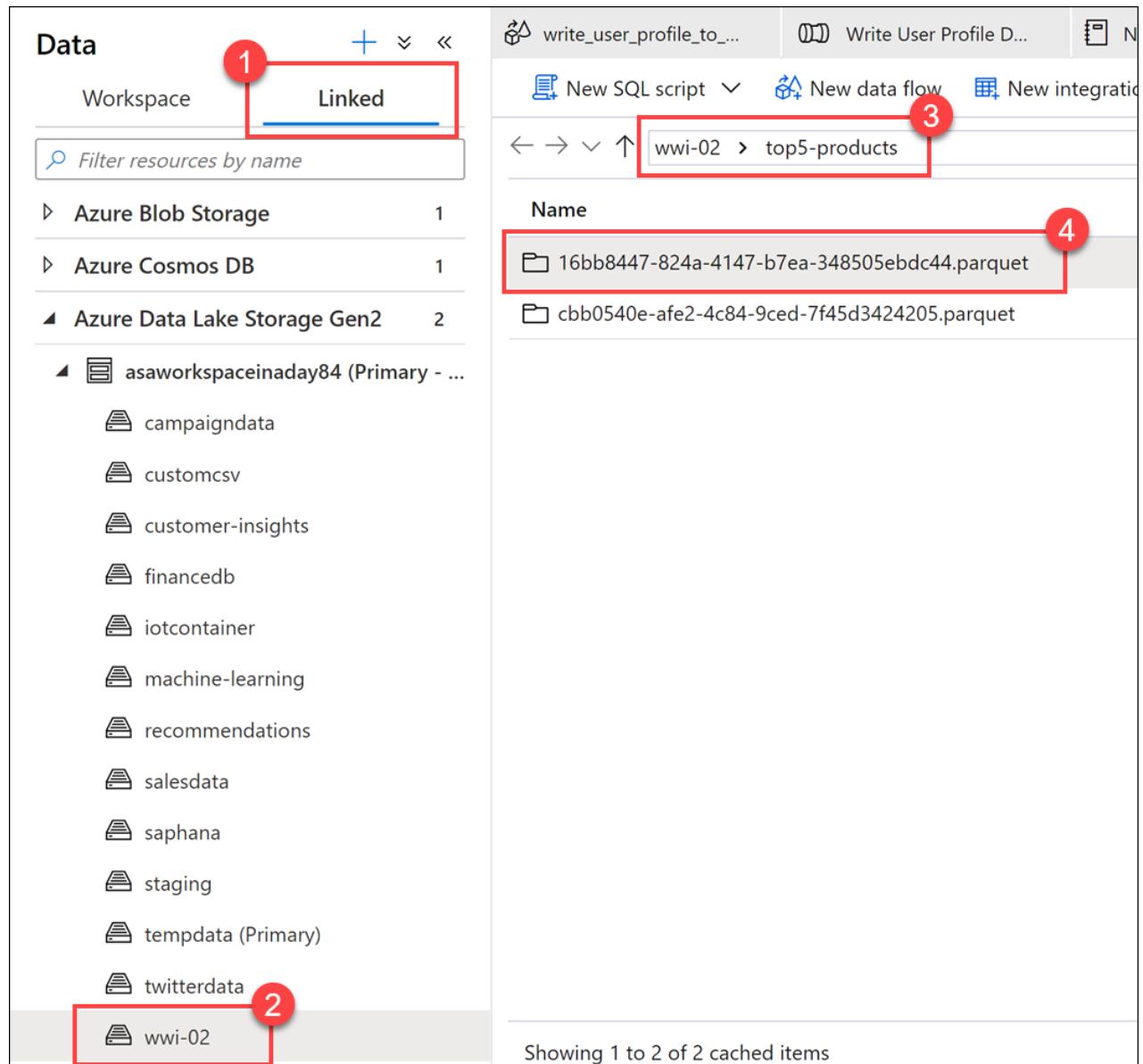
Summary Metrics for 200 Completed Tasks

Metric	Min	25th percentile	Median
Duration	20 ms	42 ms	57 ms
GC Time	0 ms	0 ms	0 ms
Shuffle Read Size / Records	43.9 KB / 6471	50.2 KB / 7648	51.5 KB / 8057

7. Go back to the **Data** hub.



8. Select the **Linked** tab (1), then select the **wwi-02** container (2) on the primary data lake storage account, go to the **top5-products** folder (3), and verify that a folder exists for the Parquet file whose name matches the Pipeline run ID.



As you can see, we have a file whose name matches the **Pipeline run ID** we previously noted:

Write User Profile Data to ASA

List Gantt

Rerun Rerun from activity Rerun from failed activity Refresh

```
graph LR; A[Data flow: write_user_profile_to_asa] --> B[Notebook: Calculate Top 5 Products]
```

Activity runs

Pipeline run ID 16bb8447-824a-4147-b7ea-348505ebdc44

All status ▾

Showing 1 - 2 of 2 items

Activity name	Activity type	Run start ↑↓	Duration	Status	Integration runtime
Calculate Top 5 Products	SynapseNotebook	11/26/20, 3:39:57 AM	00:05:45	✓ Succeeded	DefaultIntegrationRuntime (East US 2)
write_user_profile_to_asa	ExecuteDataFlow	11/26/20, 3:33:50 AM	00:06:07	✓ Succeeded	DefaultIntegrationRuntime (East US 2)

These values match because we passed in the Pipeline run ID to the `runId` parameter on the Notebook activity.