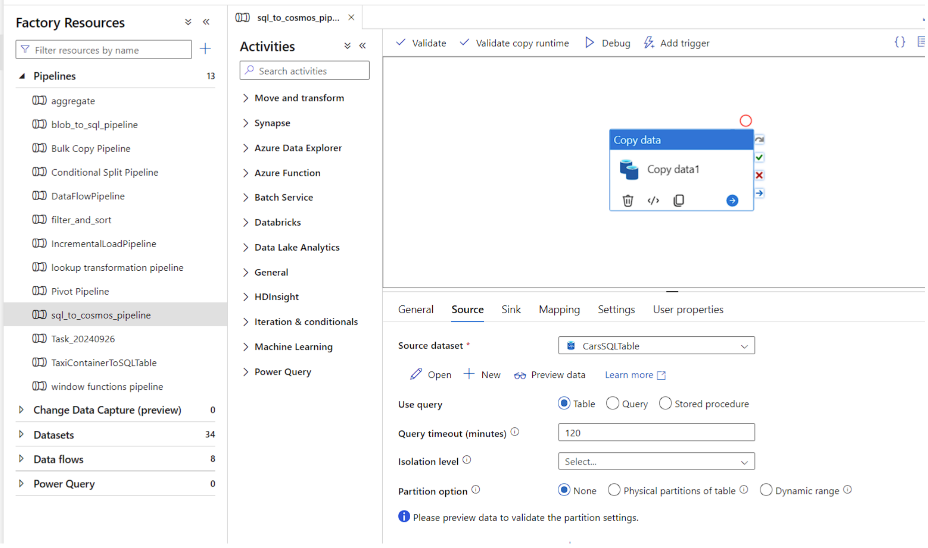
Assignment 10 - Azure Data Factory

Q1. Design an ADF pipeline to copy data from an on-premise Azure SQL database to Azure Cosmos DB, ensuring data consistency and performance optimization. Pick correct options of partitioning for better performance.

a. Main component of the pipeline is the copy data tool. Selected source is an SQL table named CarsSQLTable containing various details of different models of cars from multiple manufacturers.



Source dataset * CarsSQLTable

Use query ● Table ○ Query ○ Stored procedure

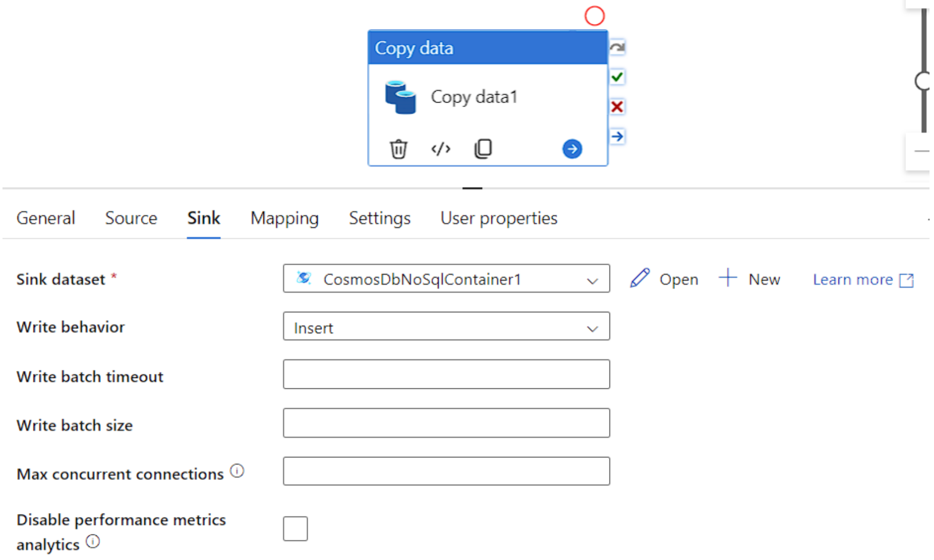
Query timeout (minutes) 120

Isolation level Select...

Partition option ● None ○ Physical partitions of table ○ Dynamic range

Please preview data to validate the partition settings.

b. Sink for the Copy Data tool is a Container in Azure CosmosDB.



General | Source | Sink | Mapping | Settings | User properties

Sink dataset * CosmosDbNoSqlContainer1 — Open — New — Learn more

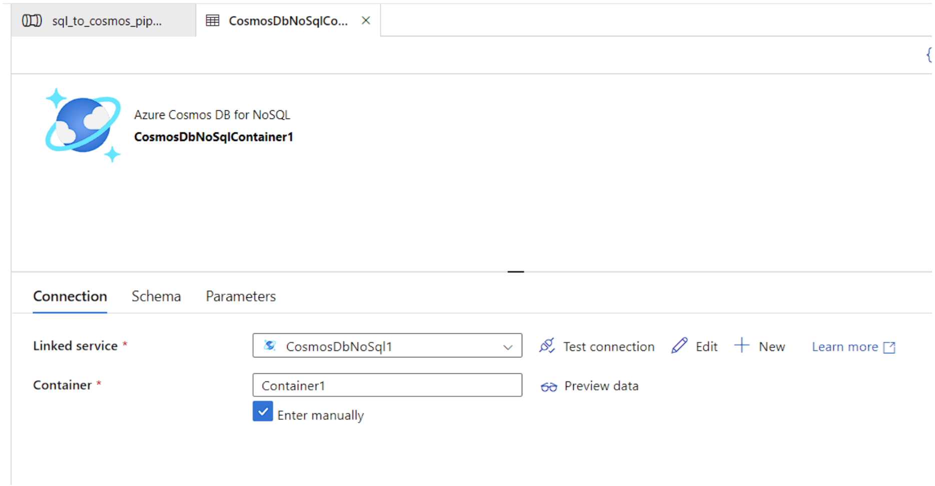
Write behavior: Insert

Write batch timeout:

Write batch size:

Max concurrent connections:

Disable performance metrics analytics: □



Azure Cosmos DB for NoSQL

CosmosDbNoSqlContainer1

Connection | Schema | Parameters

Linked service * CosmosDbNoSql1 — Test connection — Edit — New — Learn more

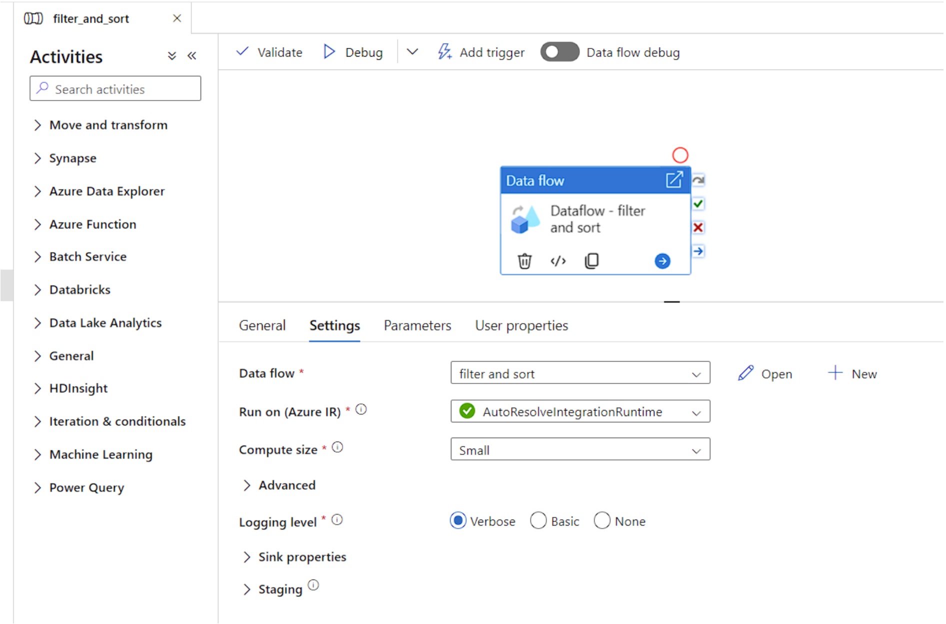
Container * Container1 — Preview data

☑ Enter manually

Multiple partitioning options are availlable while choosing source - here, Noone is choosen due to the smaller size of availabe data which will not provide significant performance benefits. For bigger datasets, using partitions will help provide performance benefits by parallelizing data extraction

Q2. Create Pipeline using Azure Data Flow in Azure Data Factory to apply Filter and Sort transformations on datasets.

a. Main component of the pipeline is a dataflow task.



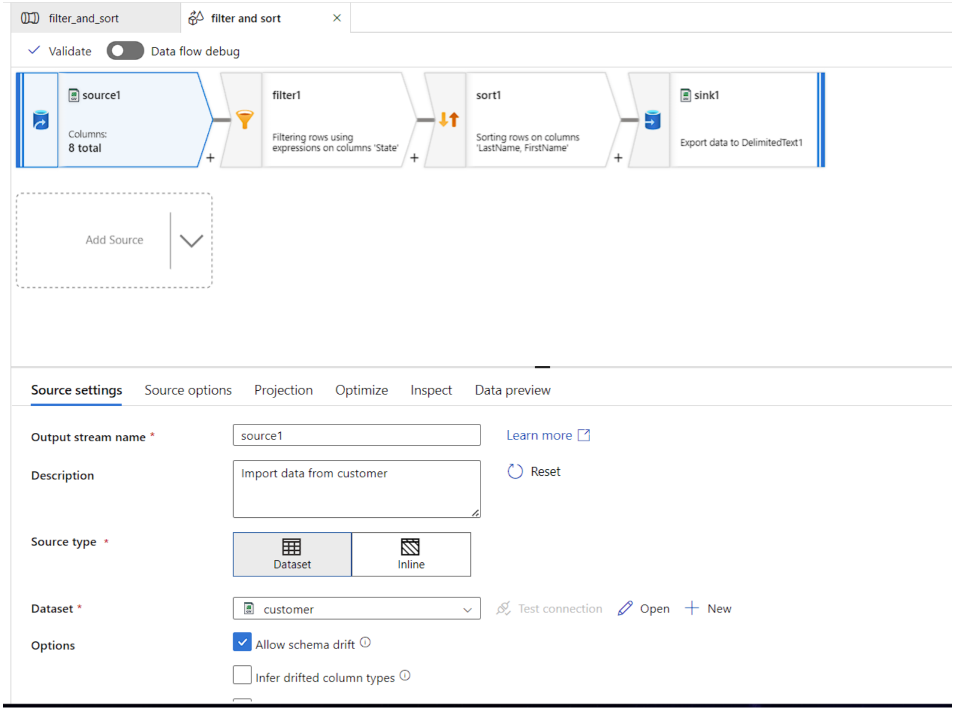
Data flow * filter and sort

Run on (Azure IR) * AutoResolveIntegrationRuntime

Compute size * Small

Logging level * ● Verbose ○ Basic ○ None

b. Dataflow first involves taking the data from a FlatFile source named Customers.



Output stream name * source1

Description Import data from customer

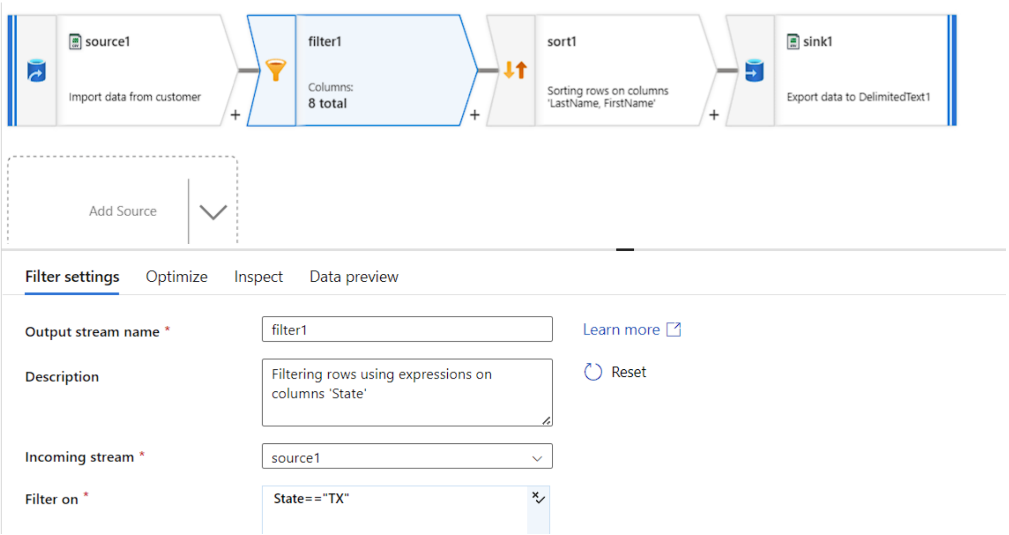
Source type * Dataset

Dataset * customer

Options ☑ Allow schema drift

□ Infer drifted column types

c. Filter involves taking only rows from the data where State equals 'TX'



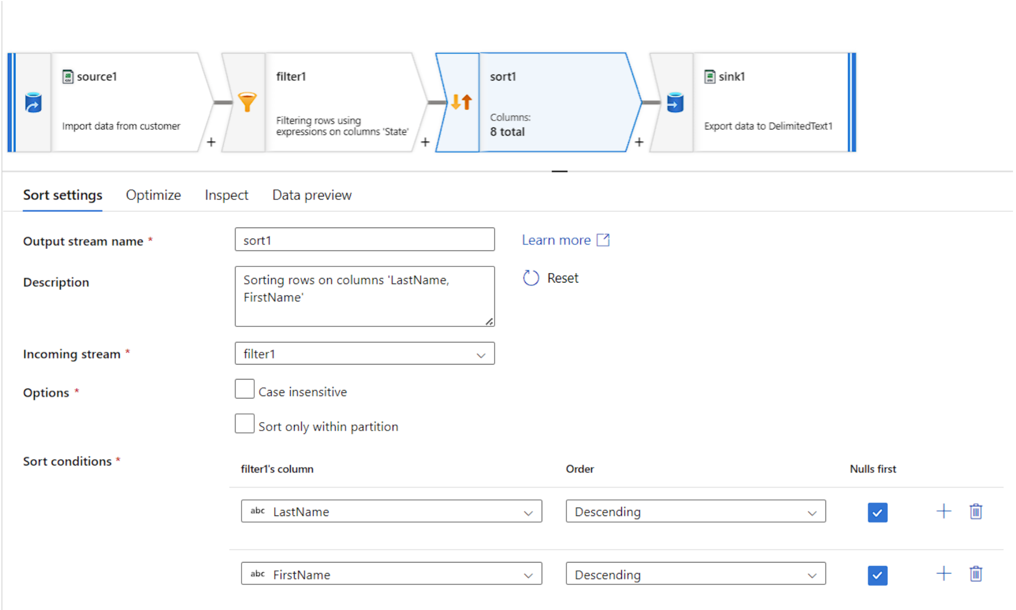
Output stream name * filter1

Description: Filtering rows using expressions on columns 'State'

Incoming stream * source1

Filter on * State=="TX"

d. Sort involves sorting the output from Filter step based on descending order of LastName and FirstName columns


source1
Import data from customer
filter1
Filtering rows using expressions on columns 'State'
sort1
Columns:
8 total
sink1
Export data to DelimitedText1
Sort settings
Optimize
Inspect
Data preview
Learn more
Reset

Output stream name * sort1

Description: Sorting rows on columns 'LastName, FirstName'

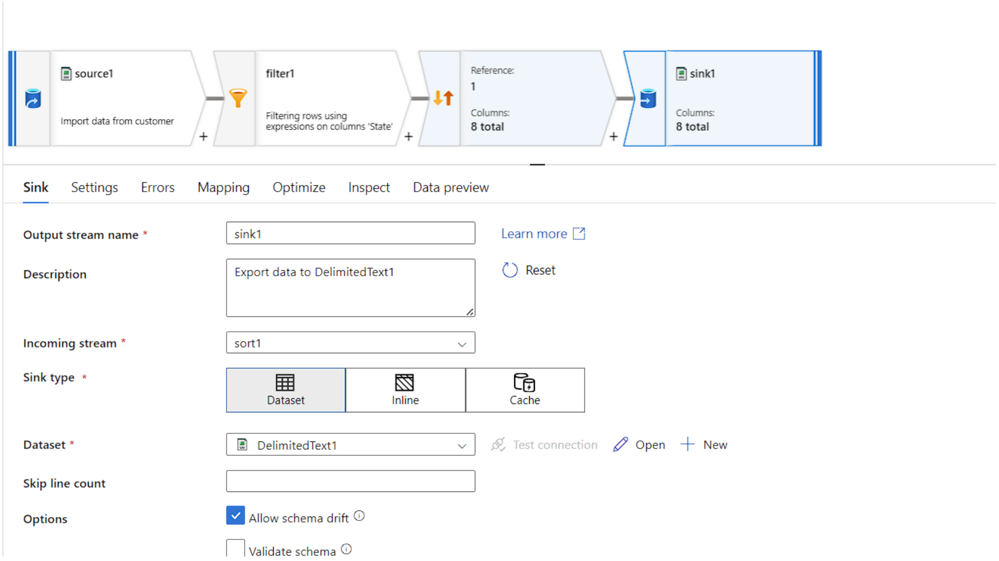
Incoming stream * filter1

Options *
□ Case insensitive
□ Sort only within partition

Sort conditions *

filter1's column	Order	Nulls first
abc LastName	Descending	☑
abc FirstName	Descending	☑

e. Final destination is a CSV file.



Output stream name * sink1

Description: Export data to DelimitedText1

Incoming stream * sort1

Sink type * Dataset

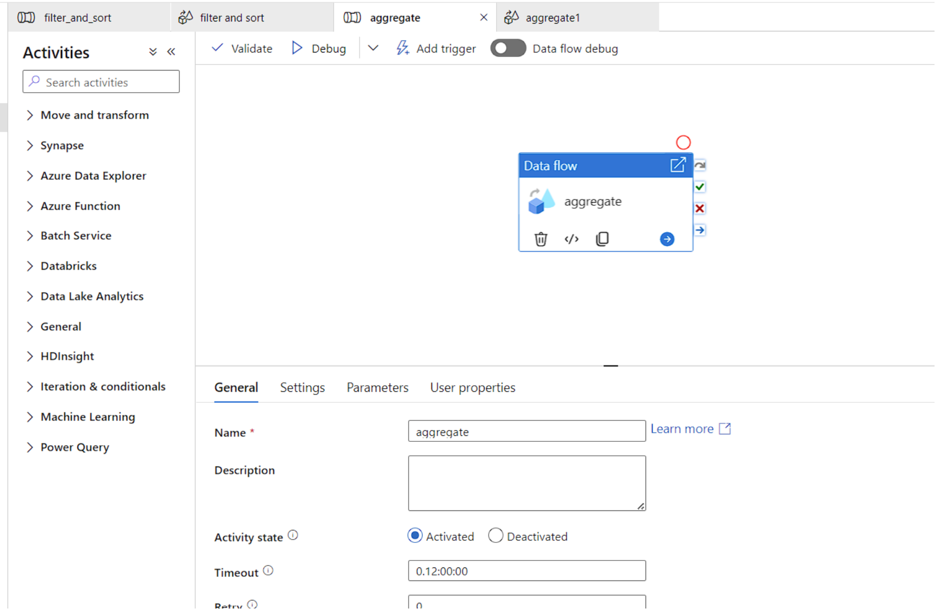
Dataset * DelimitedText1

Skip line count

Options
☑ Allow schema drift
□ Validate schema

Q3. Design an ADF pipeline to implement aggregate operations, such as sum, average, max, min and count, within an Azure Data Flow.

a. Main component of PIpeline is a dataflow task.



Name * aggregate

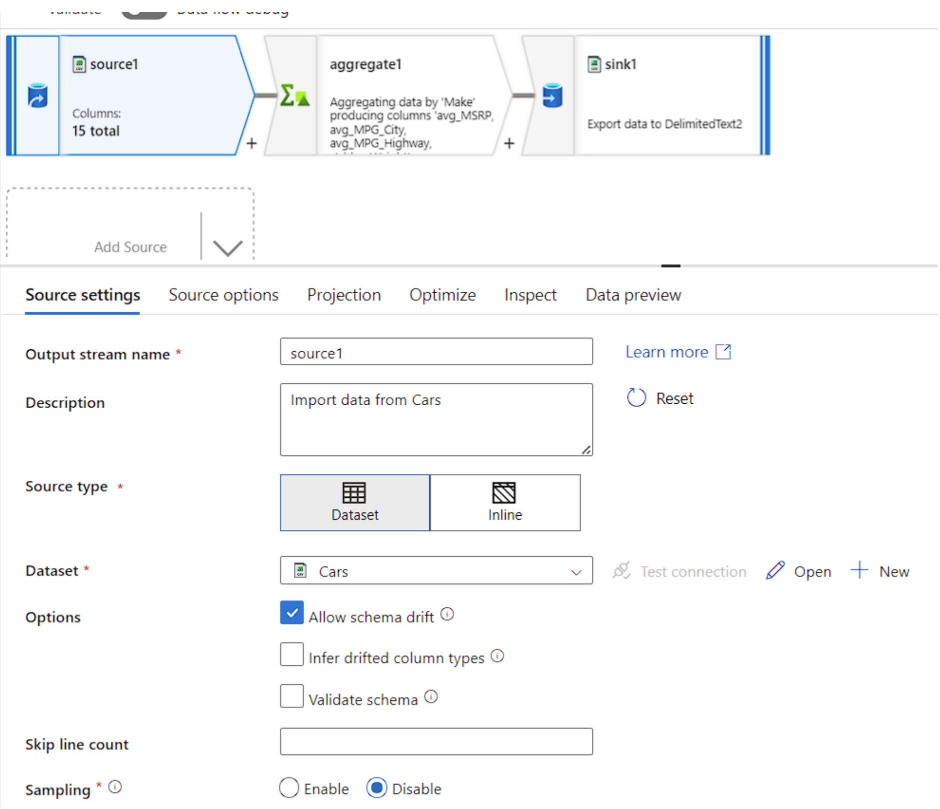
Description

Activity state ● Activated ○ Deactivated

Timeout 0.12:00:00

Retry 0

b. Data source here is a CSV file named Cars. Change datatype of each column to the appropriate type from the projection section.



Output stream name * source1

Description: Import data from Cars

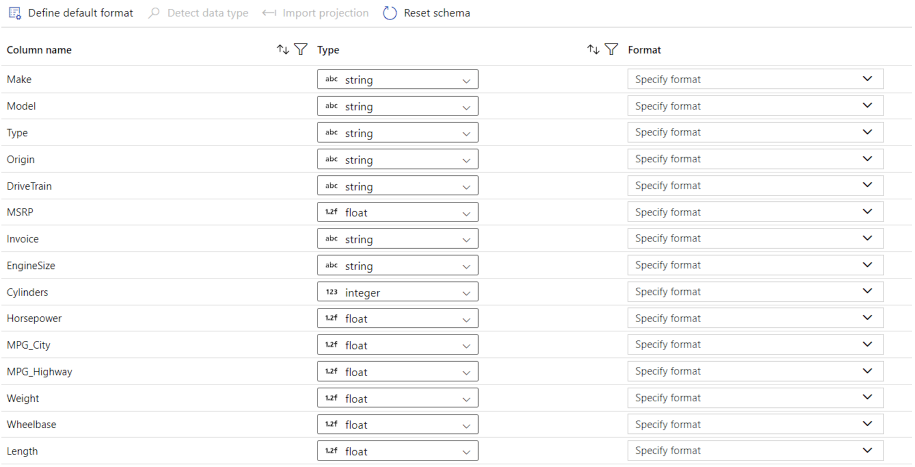
Source type *: Dataset (selected) | Inline

Dataset *: Cars

Options:
☑ Allow schema drift
□ Infer drifted column types
□ Validate schema

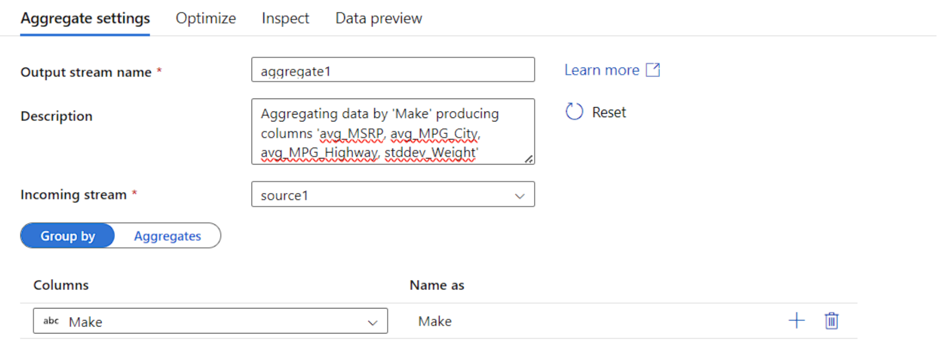
Skip line count:

Sampling *: ○ Enable ● Disable



Column name	Type	Format
Make	string	Specify format
Model	string	Specify format
Type	string	Specify format
Origin	string	Specify format
DriveTrain	string	Specify format
MSRP	float	Specify format
Invoice	string	Specify format
EngineSize	string	Specify format
Cylinders	integer	Specify format
Horsepower	float	Specify format
MPG_City	float	Specify format
MPG_Highway	float	Specify format
Weight	float	Specify format
Wheelbase	float	Specify format
Length	float	Specify format

c. Aggregating data by 'Make' producing columns 'avg_MSRP, avg_MPG_City, avg_MPG_Highway, stddev_Weight'

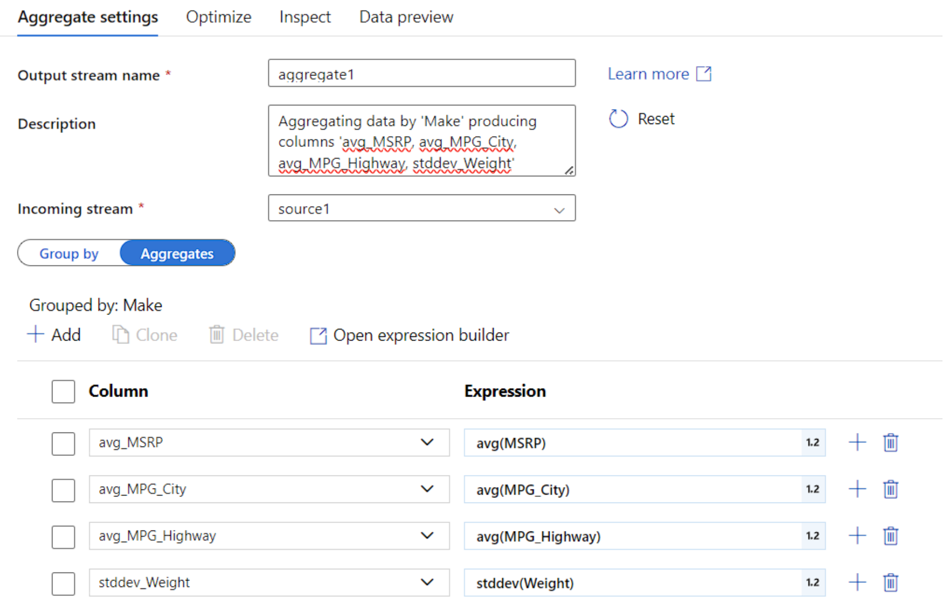


Aggregate settings | Optimize | Inspect | Data preview

Output stream name *	aggregate1	Learn more
Description	Aggregating data by 'Make' producing columns 'avg_MSRP, avg_MPG_City, avg_MPG_Highway, stddev_Weight'	Reset
Incoming stream *	source1	

Group by | Aggregates

Columns	Name as
abc Make	Make



Aggregate settings | Optimize | Inspect | Data preview

Output stream name *	aggregate1	Learn more
Description	Aggregating data by 'Make' producing columns 'avg_MSRP, avg_MPG_City, avg_MPG_Highway, stddev_Weight'	Reset
Incoming stream *	source1	

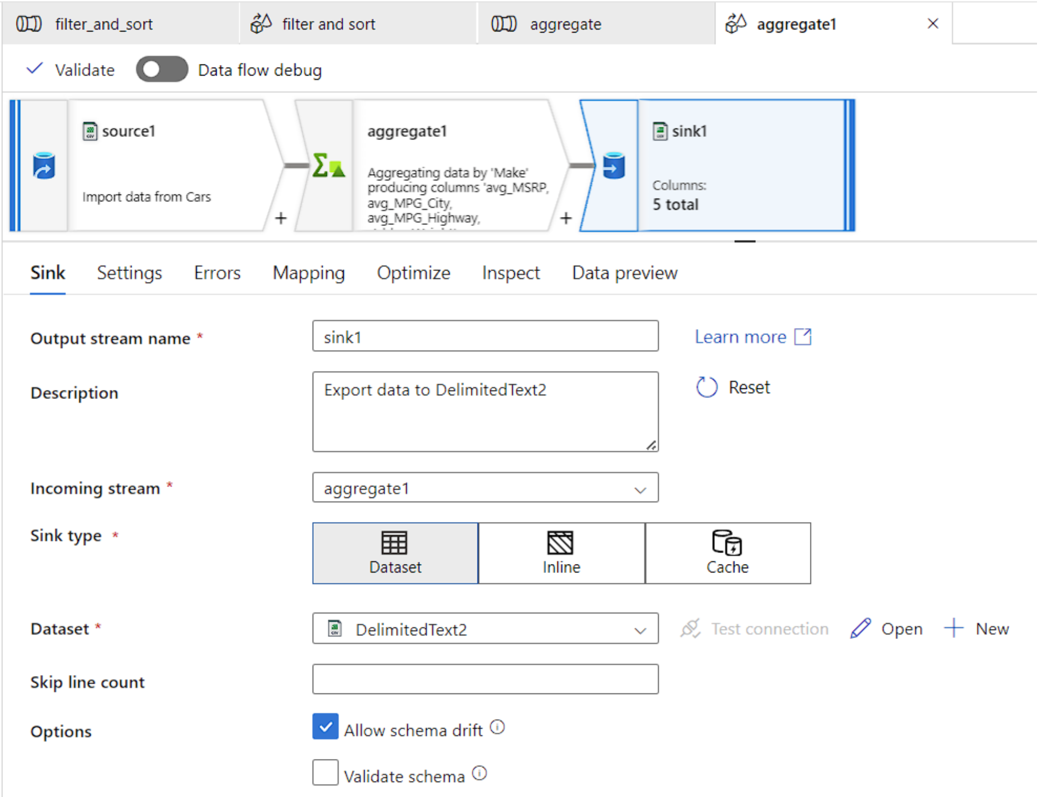
Group by | Aggregates

Grouped by: Make

+ Add | Clone | Delete | Open expression builder

□ Column	Expression
□ avg_MSRP	avg(MSRP)
□ avg_MPG_City	avg(MPG_City)
□ avg_MPG_Highway	avg(MPG_Highway)
□ stddev_Weight	stddev(Weight)

d. Final output is stored to a CSV file



filter_and_sort | filter and sort | aggregate | aggregate1 ×

✓ Validate | Data flow debug

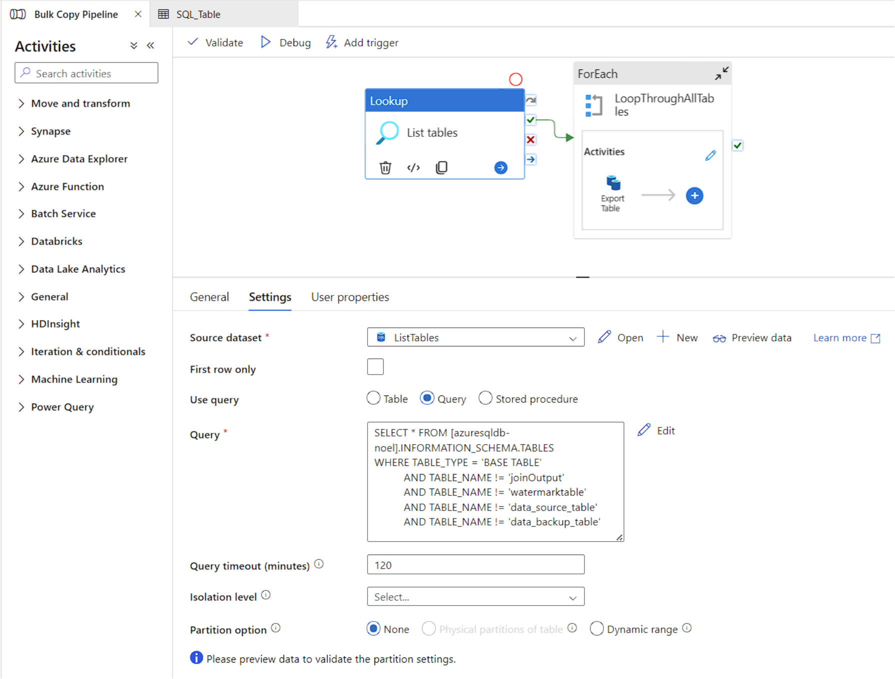
source1
Import data from Cars
aggregate1
Aggregating data by 'Make' producing columns 'avg_MSRP, avg_MPG_City, avg_MPG_Highway,
sink1
Columns:
5 total

Sink | Settings | Errors | Mapping | Optimize | Inspect | Data preview

Output stream name *	sink1	Learn more
Description	Export data to DelimitedText2	Reset
Incoming stream *	aggregate1	
Sink type *	Dataset | Inline | Cache	
Dataset *	DelimitedText2	Test connection | Open | + New
Skip line count		
Options	☑ Allow schema drift □ Validate schema	

4. Create best approach to bulk copy data from multiple homogenous sources into Azure SQL Database using ADF pipelines. Show usage of Lookup, For Each Loop and Expressions in Azure Data Factory.

a. Lookup component uses a query to get list of all tables in the specified database. The source dataset has only a linked service and no table connection as it is used just to have a connection the the database.



Source dataset * ListTables

First row only □

Use query ○ Table ● Query ○ Stored procedure

Query *

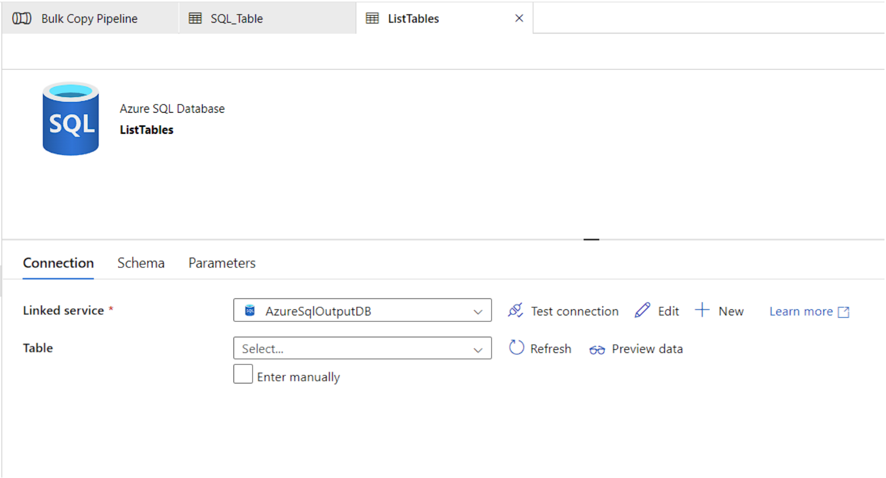
```
SELECT * FROM [azuresqldb-
noel].INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
        AND TABLE_NAME != 'joinOutput'
        AND TABLE_NAME != 'watermarktable'
        AND TABLE_NAME != 'data_source_table'
        AND TABLE_NAME != 'data_backup_table'
```

Query timeout (minutes) 120

Isolation level Select...

Partition option ● None ○ Physical partitions of table ○ Dynamic range

Please preview data to validate the partition settings.



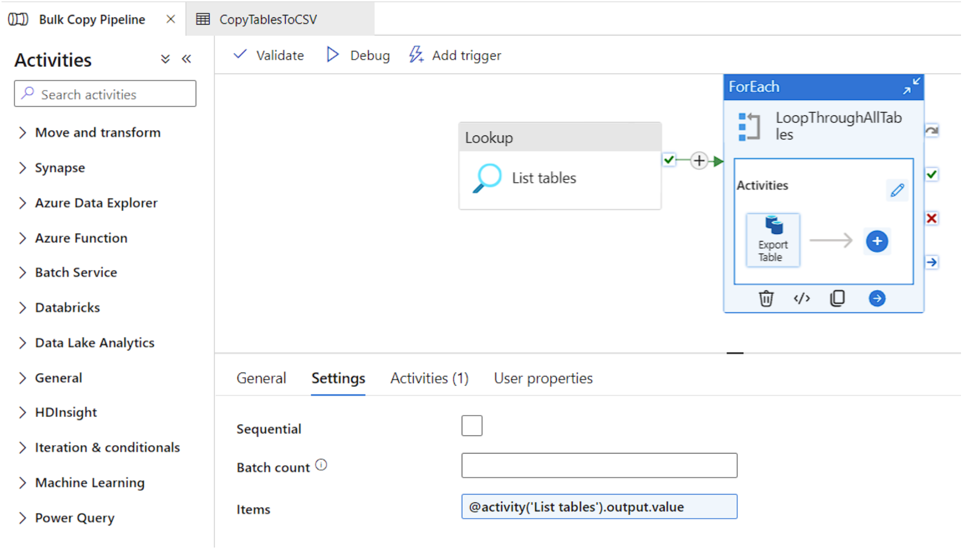
Azure SQL Database
ListTables

Linked service * AzureSqlOutputDB

Table Select...

□ Enter manually

b. ForEach loop gives the name of the tables to the 'Export Table' activity inside it using the expression `@activity('List tables').output.value`

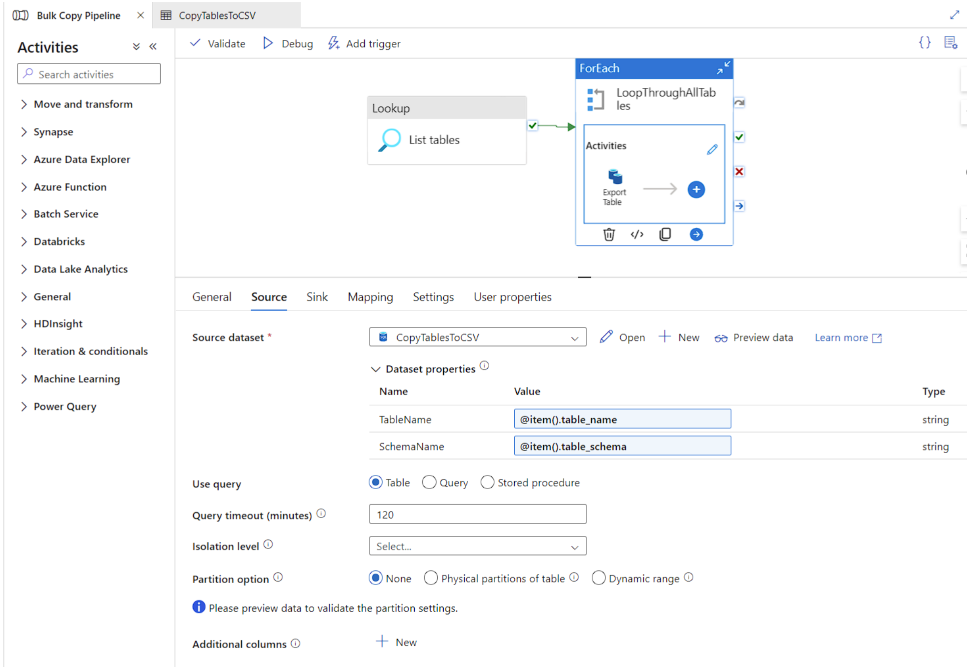


Sequential □

Batch count

Items @activity('List tables').output.value

c. In the Export Table activity, source dataset has two properties - TableName and SchemaName. These were used to access each table in database.



Source dataset * CopyTablesToCSV

Dataset properties

Name	Value	Type
TableName	@item().table_name	string
SchemaName	@item().table_schema	string

Use query ● Table ○ Query ○ Stored procedure

Query timeout (minutes) 120

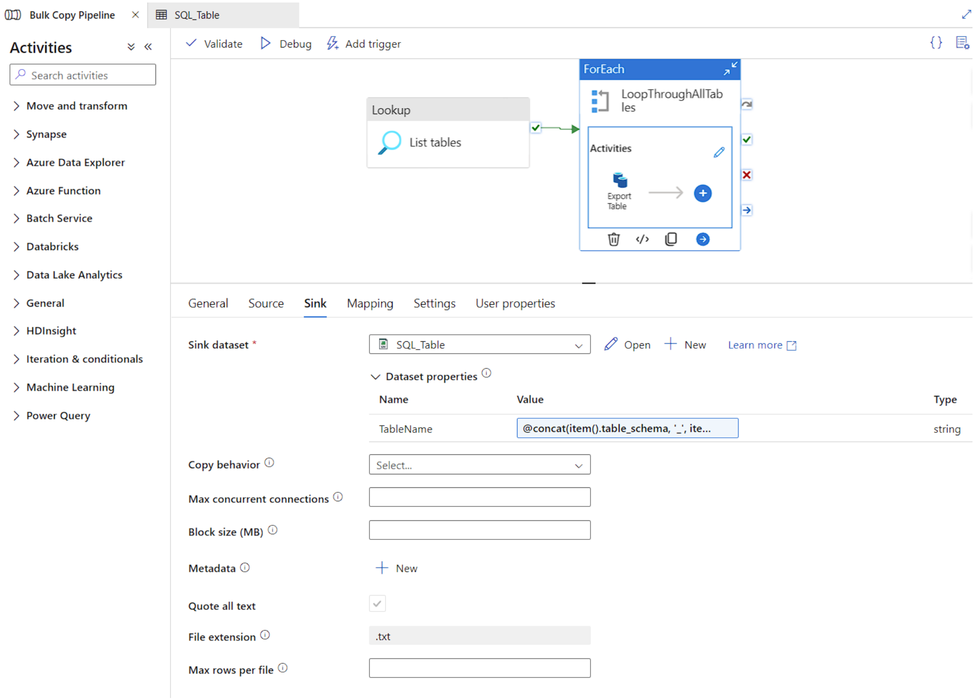
Isolation level Select...

Partition option ● None ○ Physical partitions of table ○ Dynamic range

Please preview data to validate the partition settings.

Additional columns + New

d. Sink dataset is a blob storage, where we store as records of each table in CSV files. Dataset has a property named TableName with value set to `@concat(item().table_schema, '_', item().table_name, '.csv')`



Sink dataset * SQL_Table

Dataset properties

Name	Value	Type
TableName	@concat(item().table_schema, '_', ite...	string

Copy behavior: Select...

Max concurrent connections:

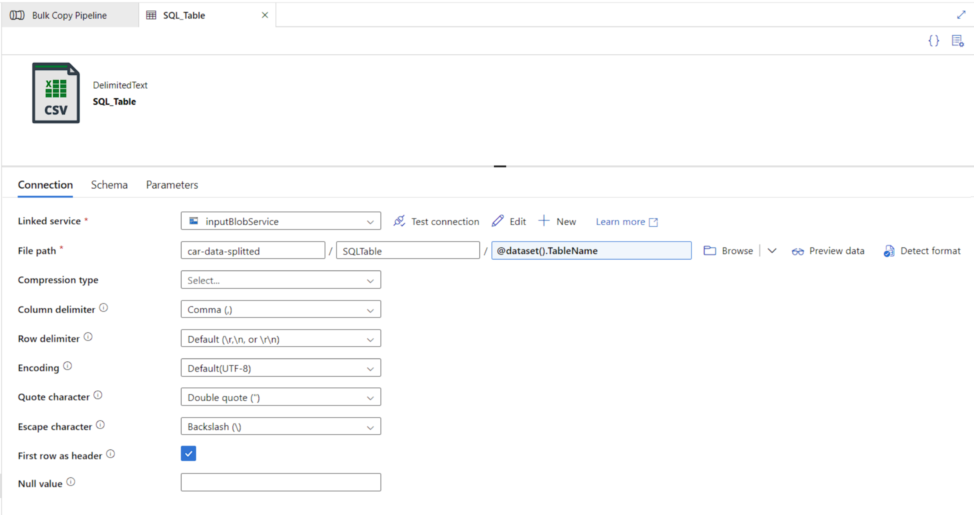
Block size (MB):

Metadata: + New

Quote all text: ✓

File extension: .txt

Max rows per file:



DelimitedText
SQL_Table

Linked service * inputBlobService

File path * car-data-splitted / SQLTable / @dataset().TableName

Compression type: Select...

Column delimiter: Comma (,)

Row delimiter: Default (\r,\n, or \r\n)

Encoding: Default(UTF-8)

Quote character: Double quote (")

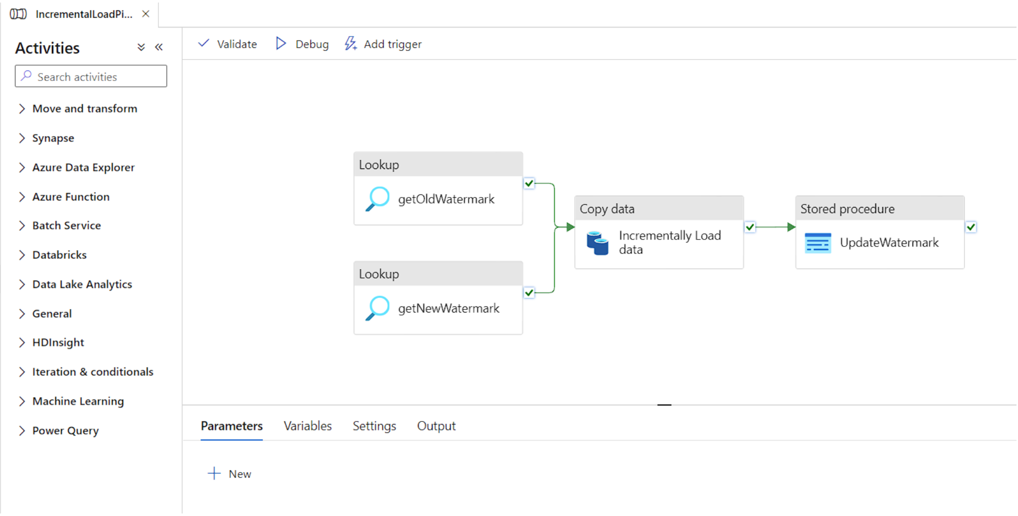
Escape character: Backslash (\)

First row as header: ☑

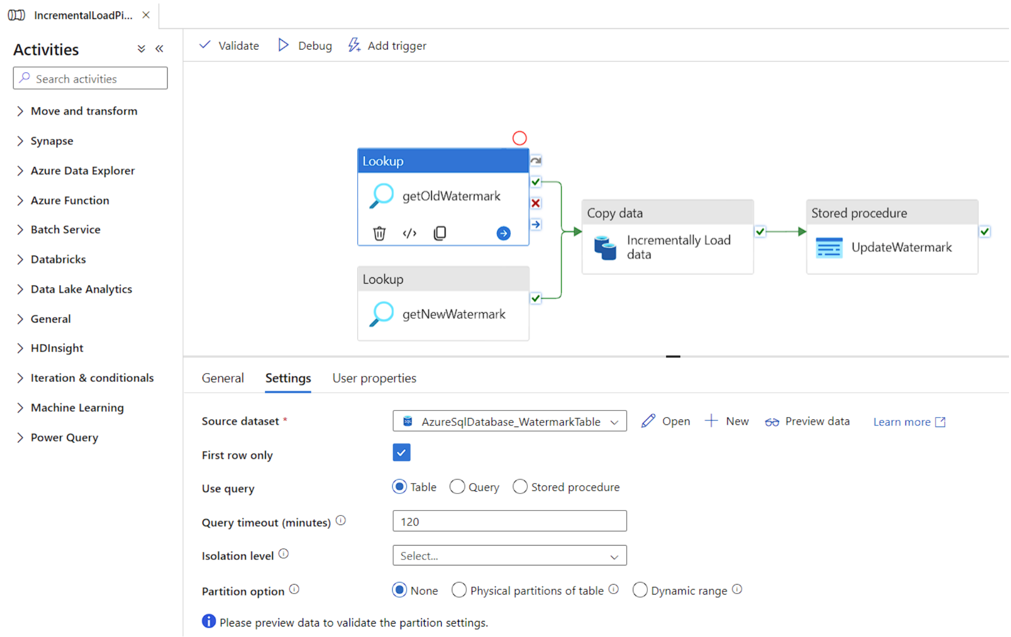
Null value:

5. Implement incremental load Pipeline in Azure Data Factory for handling datasets, ensuring efficient insert/upsert/updates to the target storage without re-inserting the entire dataset?

a. Main pipeline



b. The lookup activity named 'getOldWatermark' is used to retreive latest lookup value stored in the watermark table- which signifies when incremental loading last took place.



Source dataset * AzureSqlDatabase_WatermarkTable

First row only ☑

Use query ● Table ○ Query ○ Stored procedure

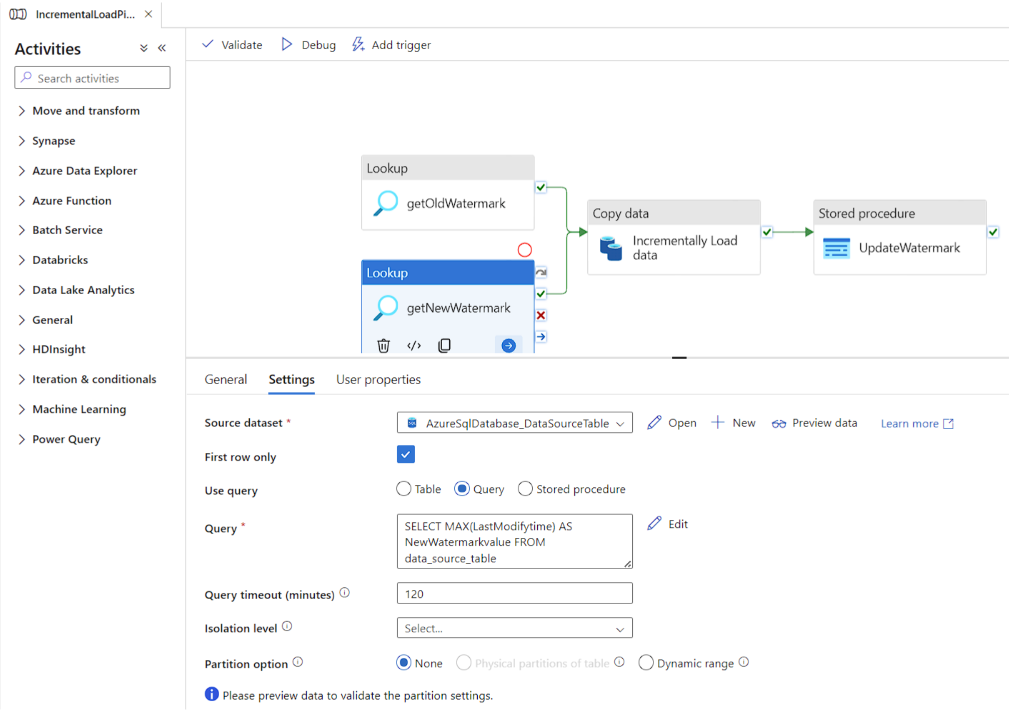
Query timeout (minutes) 120

Isolation level Select...

Partition option ● None ○ Physical partitions of table ○ Dynamic range

Please preview data to validate the partition settings.

c. The lookup activity 'getNewWatermark' uses a query to retreive the time when Source Dataset was last modified.



Source dataset * AzureSqlDatabase_DataSourceTable

First row only ☑

Use query ○ Table ● Query ○ Stored procedure

Query * SELECT MAX(LastModifytime) AS NewWatermarkvalue FROM data_source_table

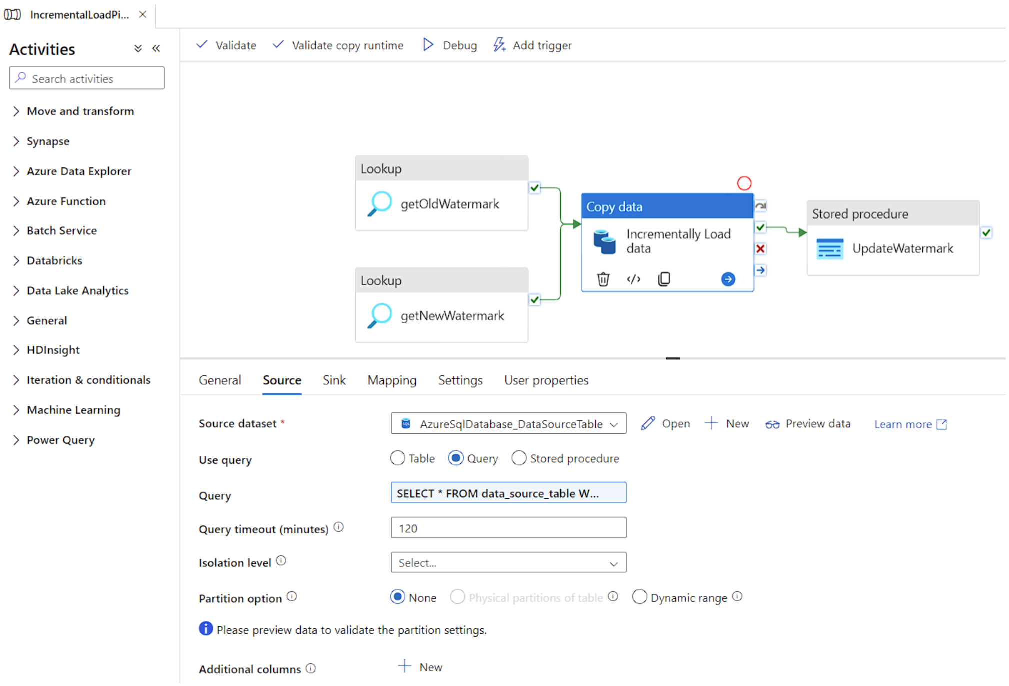
Query timeout (minutes) 120

Isolation level Select...

Partition option ● None ○ Physical partitions of table ○ Dynamic range

Please preview data to validate the partition settings.

d. Copy Data tool uses an SQL query inorder to retreive al rows where watermark value is greater than old watermark, but less than or equals newer watermark



Source dataset * AzureSqlDatabase_DataSourceTable

Use query ○ Table ● Query ○ Stored procedure

Query SELECT * FROM data_source_table W...

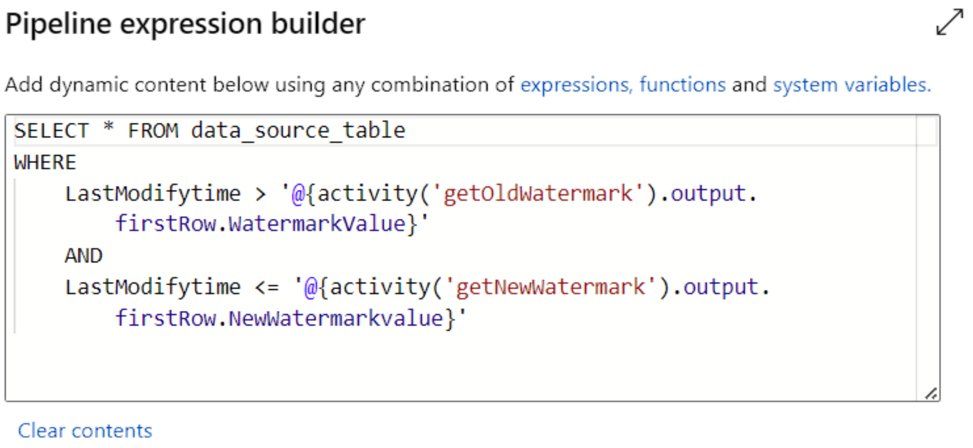
Query timeout (minutes) 120

Isolation level Select...

Partition option ● None ○ Physical partitions of table ○ Dynamic range

Please preview data to validate the partition settings.

Additional columns + New

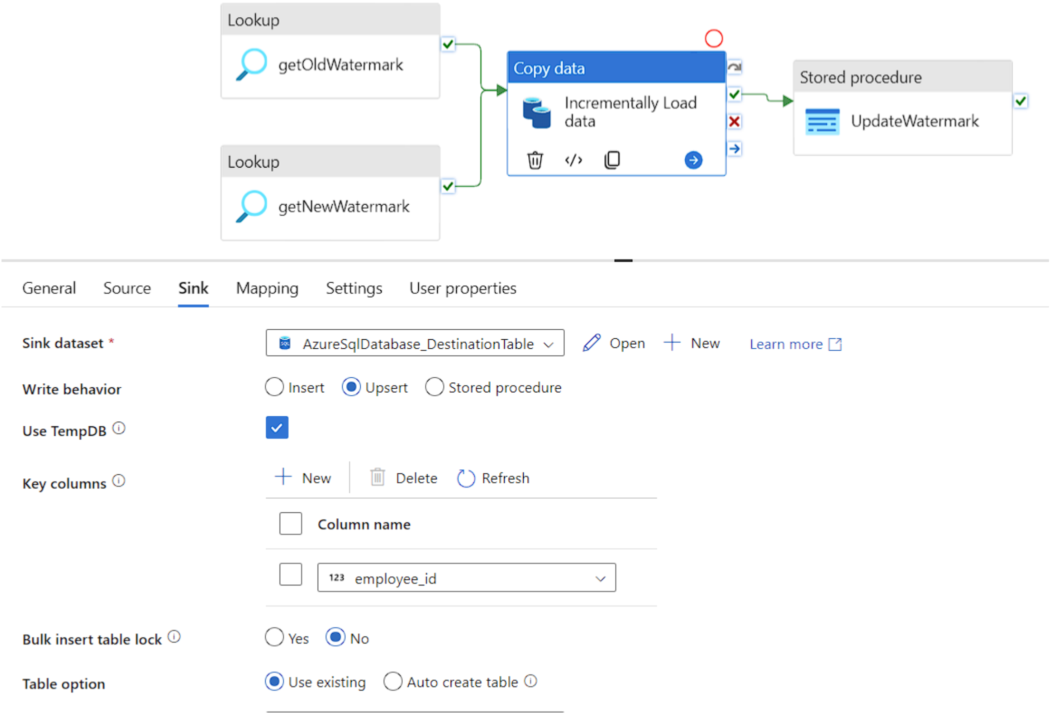


Pipeline expression builder

Add dynamic content below using any combination of expressions, functions and system variables.

```
SELECT * FROM data_source_table
WHERE
    LastModifytime > '@{activity('getOldWatermark').output.
        firstRow.WatermarkValue}'
    AND
    LastModifytime <= '@{activity('getNewWatermark').output.
        firstRow.NewWatermarkvalue}'
```

e. THe sink dataset is the destination table to which we are incrementally loading data. Upsert option is used to continuously input new data without getting overridden. Primary key for the table is employee_id.



General | Source | **Sink** | Mapping | Settings | User properties

Sink dataset * AzureSqlDatabase_DestinationTable | Open | New | Learn more

Write behavior ○ Insert ● Upsert ○ Stored procedure

Use TempDB ☑

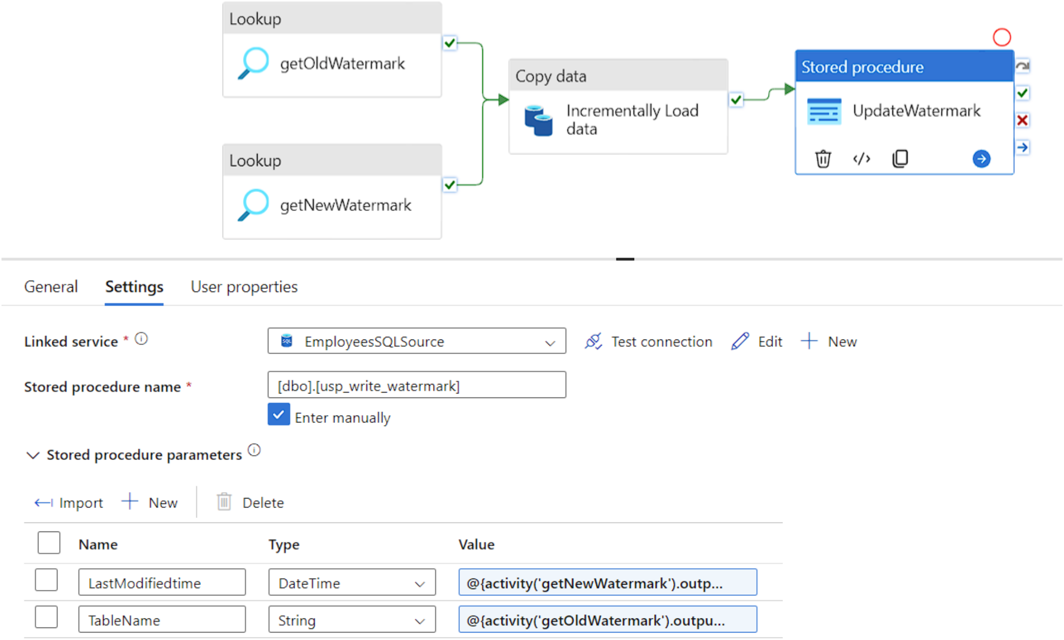
Key columns: New | Delete | Refresh

□	Column name
□	123 employee_id

Bulk insert table lock ○ Yes ● No

Table option ● Use existing ○ Auto create table

f. A stored procedure is used to write new watermark value and table name to the watermark table. 'LastModifiedtime' from getNewWatermark activity and 'TableName' are passed are parameters to the stored procedure.



General | **Settings** | User properties

Linked service * EmployeesSQLSource | Test connection | Edit | New

Stored procedure name * [dbo].[usp_write_watermark]

☑ Enter manually

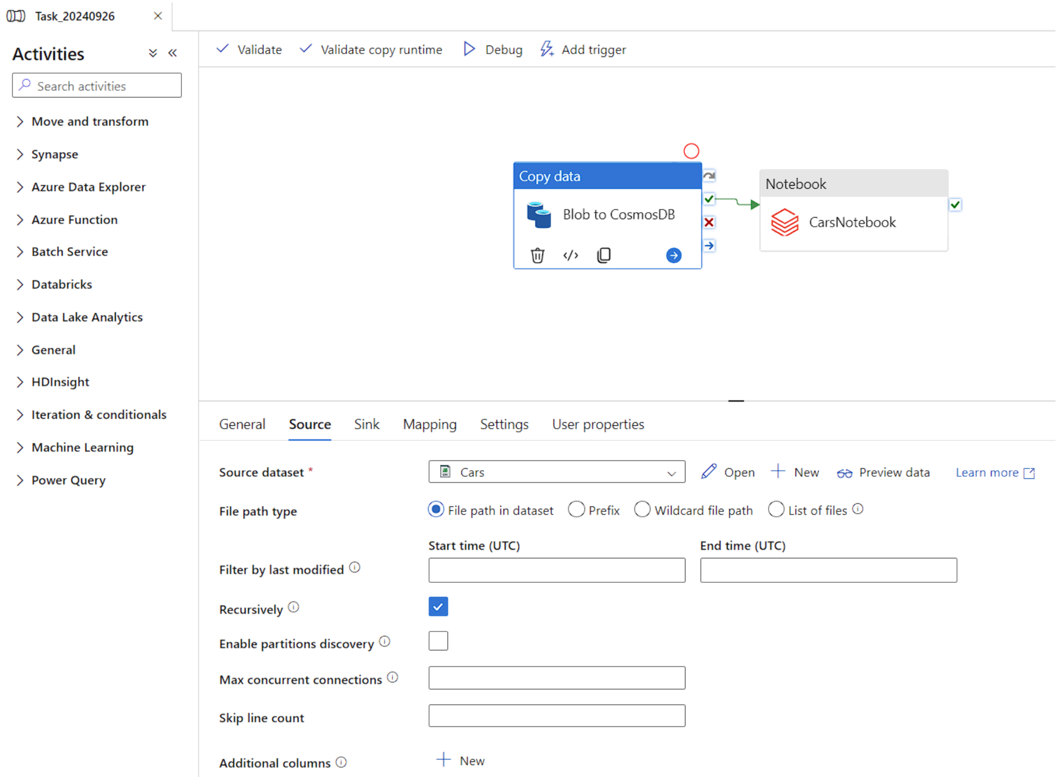
Stored procedure parameters

Import | New | Delete

□	Name	Type	Value
□	LastModifiedtime	DateTime	@{activity('getNewWatermark').outp...
□	TableName	String	@{activity('getOldWatermark').outpu...

6. What are the key steps to connect Azure Databricks to Cosmos DB for real-time analytics and data transformation using spark and Databricks.

a. Transfer data from source to CosmonDB storage. Source here is a CSV file in blob storage, that needs to be moved to Cosmos using the Copy Data Tool.



Source dataset * Cars

File path type ● File path in dataset ○ Prefix ○ Wildcard file path ○ List of files

Filter by last modified Start time (UTC) End time (UTC)

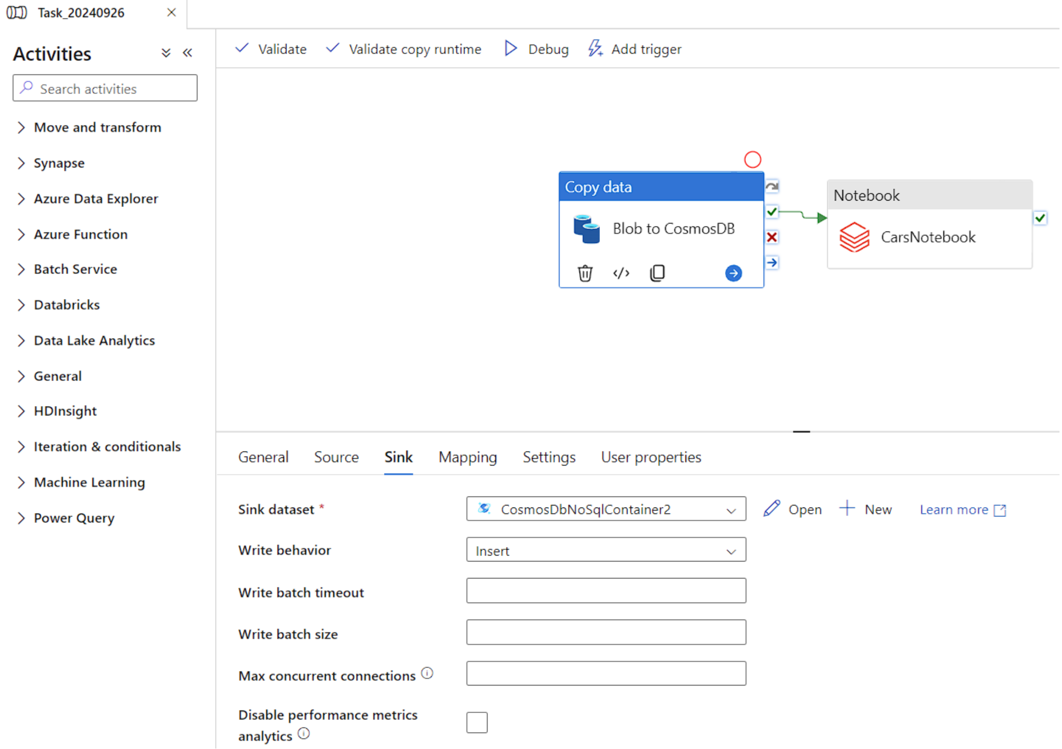
Recursively ☑

Enable partitions discovery □

Max concurrent connections

Skip line count

Additional columns + New



Sink dataset * CosmosDbNoSqlContainer2

Write behavior Insert

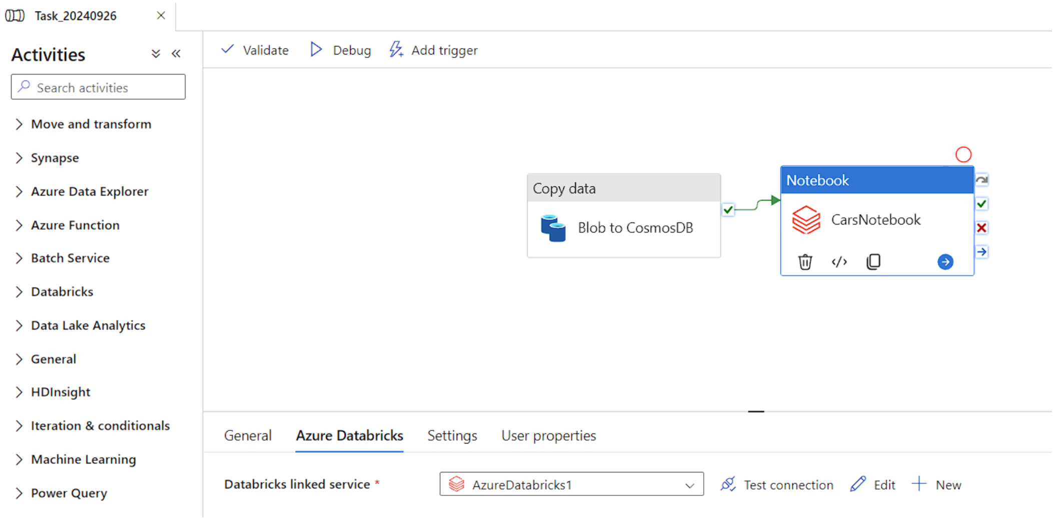
Write batch timeout

Write batch size

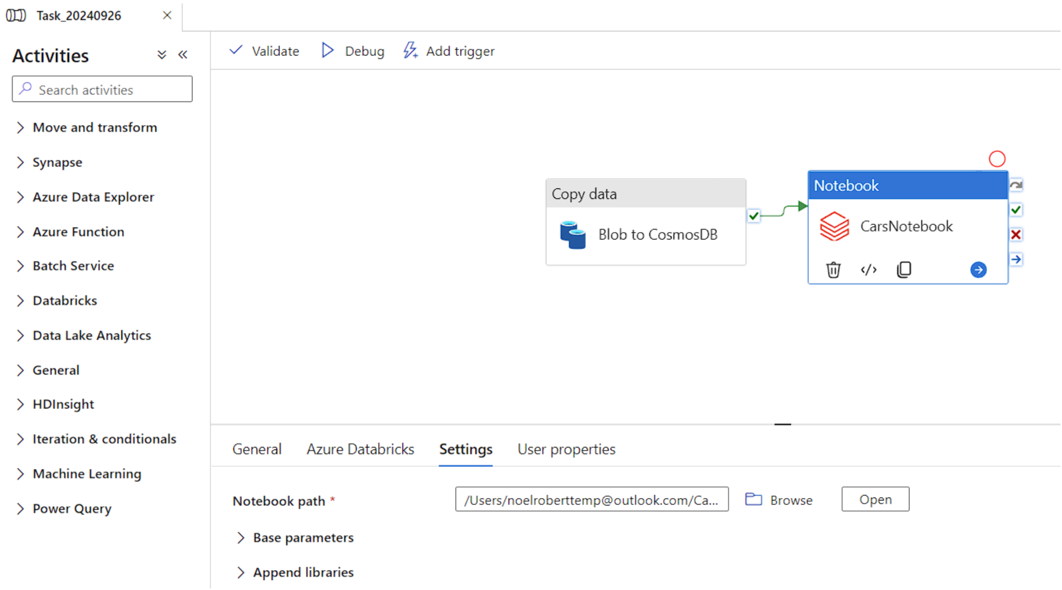
Max concurrent connections

Disable performance metrics analytics □

b. Create a linked service for connecting to databricks notebook. Go to Settings section to connect to a notebook.

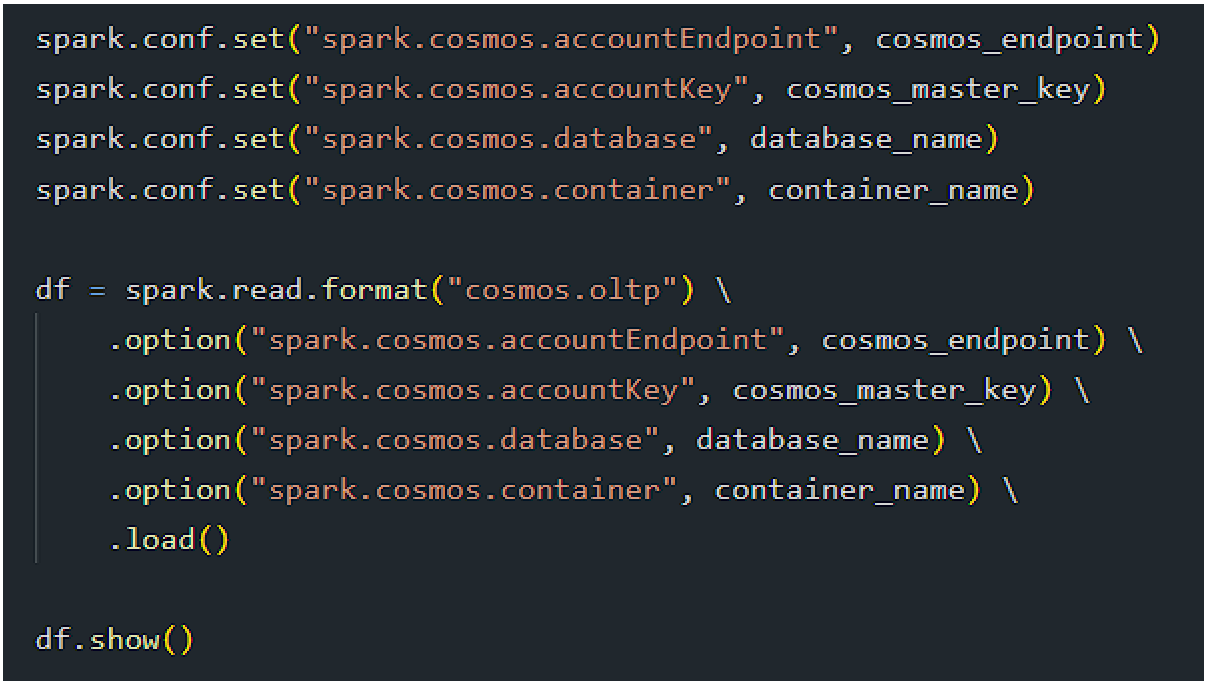


Databricks linked service * AzureDatabricks1



Notebook path * /Users/noelroberttemp@outlook.com/Ca...

c. In the databricks notebook, first install driver to connect the cluster to CosmosDB. Define endpoint URL, primary key, database name and container name. Read from cosmos as given below.



```
spark.conf.set("spark.cosmos.accountEndpoint", cosmos_endpoint)
spark.conf.set("spark.cosmos.accountKey", cosmos_master_key)
spark.conf.set("spark.cosmos.database", database_name)
spark.conf.set("spark.cosmos.container", container_name)

df = spark.read.format("cosmos.oltp") \
    .option("spark.cosmos.accountEndpoint", cosmos_endpoint) \
    .option("spark.cosmos.accountKey", cosmos_master_key) \
    .option("spark.cosmos.database", database_name) \
    .option("spark.cosmos.container", container_name) \
    .load()

df.show()
```