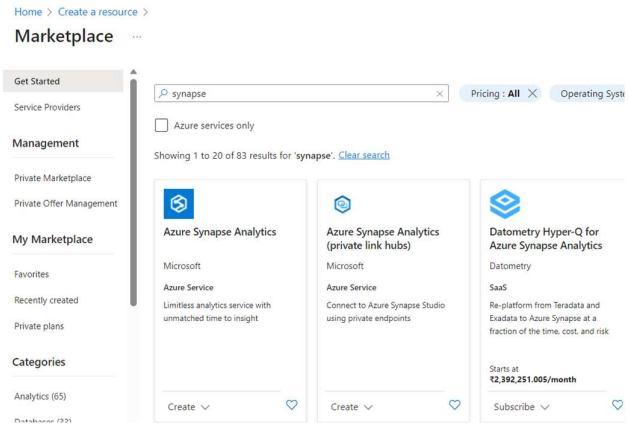
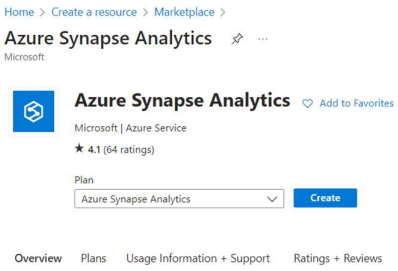
Context

Create Azure Synapse Analytics Workspace

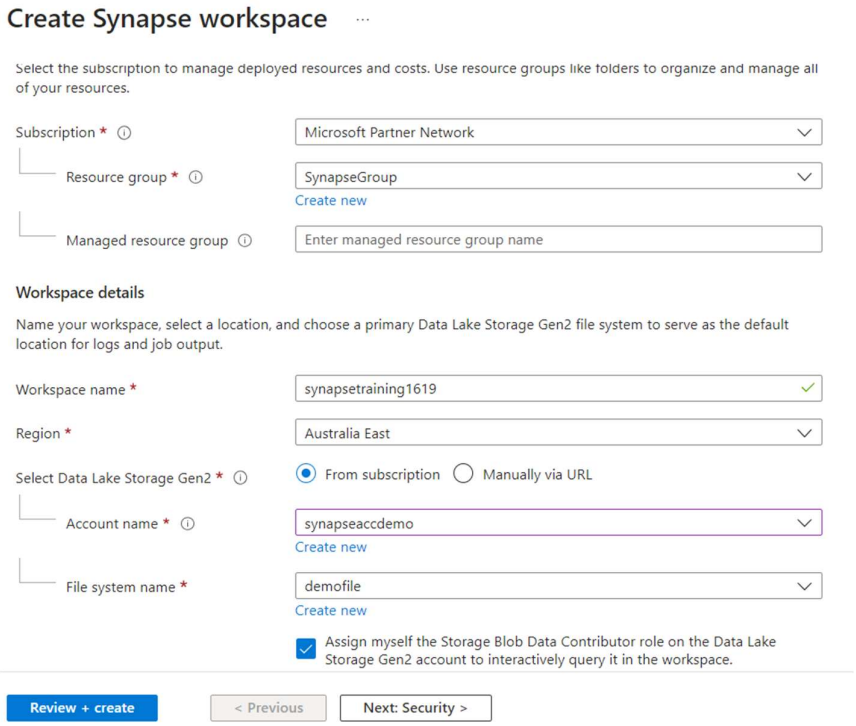
1. Go to Resources and search for Synapse then click on Azure Synapse Analytics.



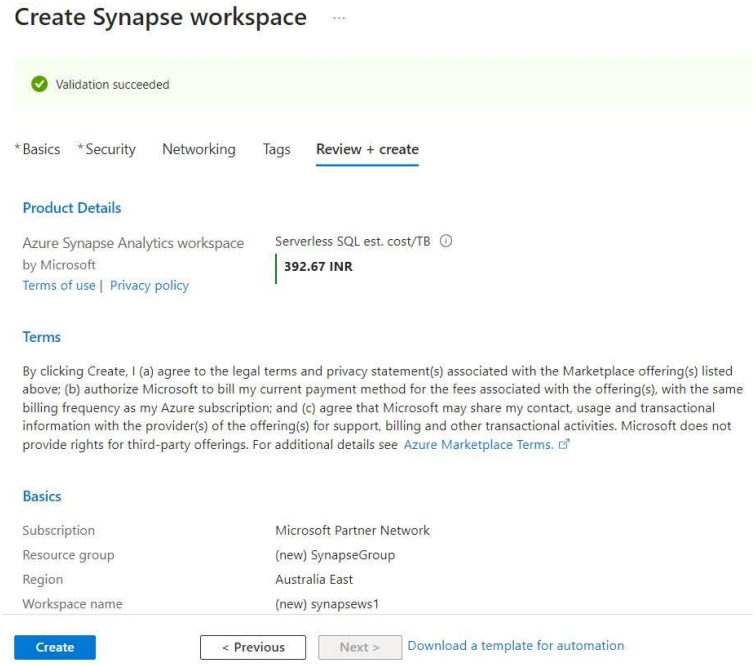
2. Click on Create.



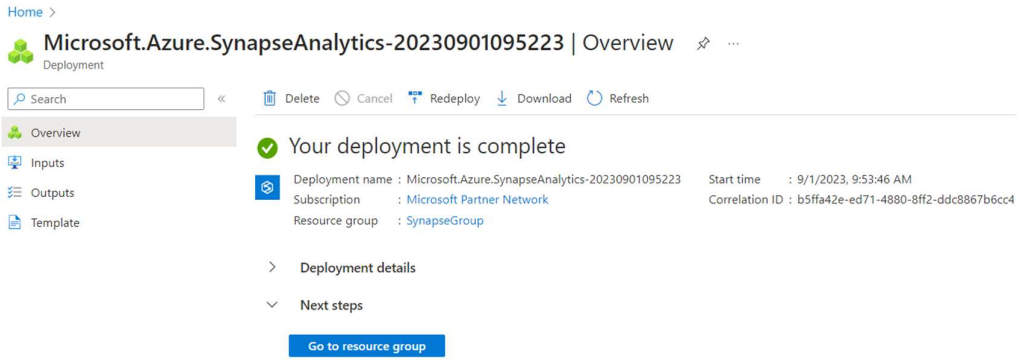
3. Under Overview, create a New Resource group, give the name, create a new account name, and create a new File System Name. Click on Review + Create.



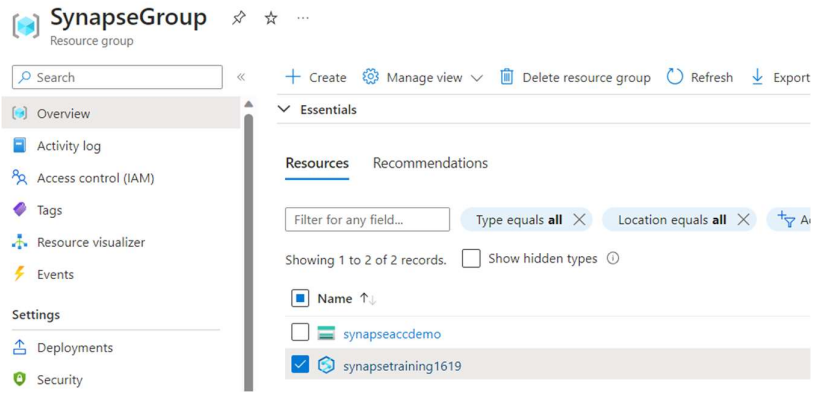
4. Click on Create.



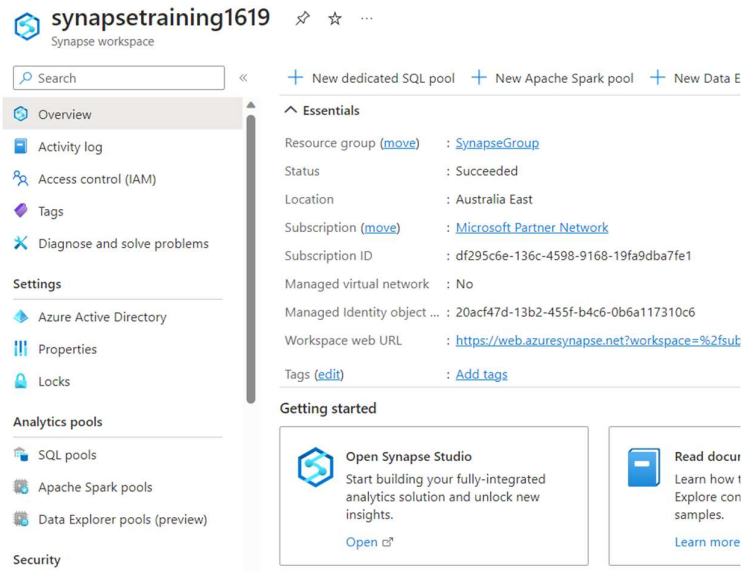
5. After creating you will see the below screen.
6. Click on Go to the resource group.



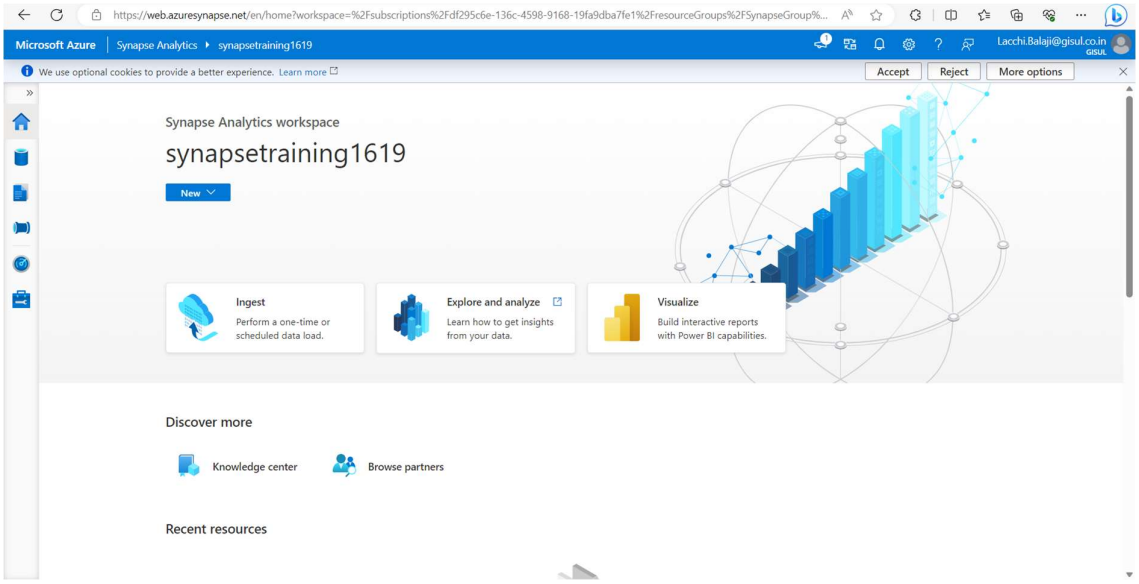
7. Here click on Synapse Workspace.



8. Now under Open Synapse Studio click on Open option.

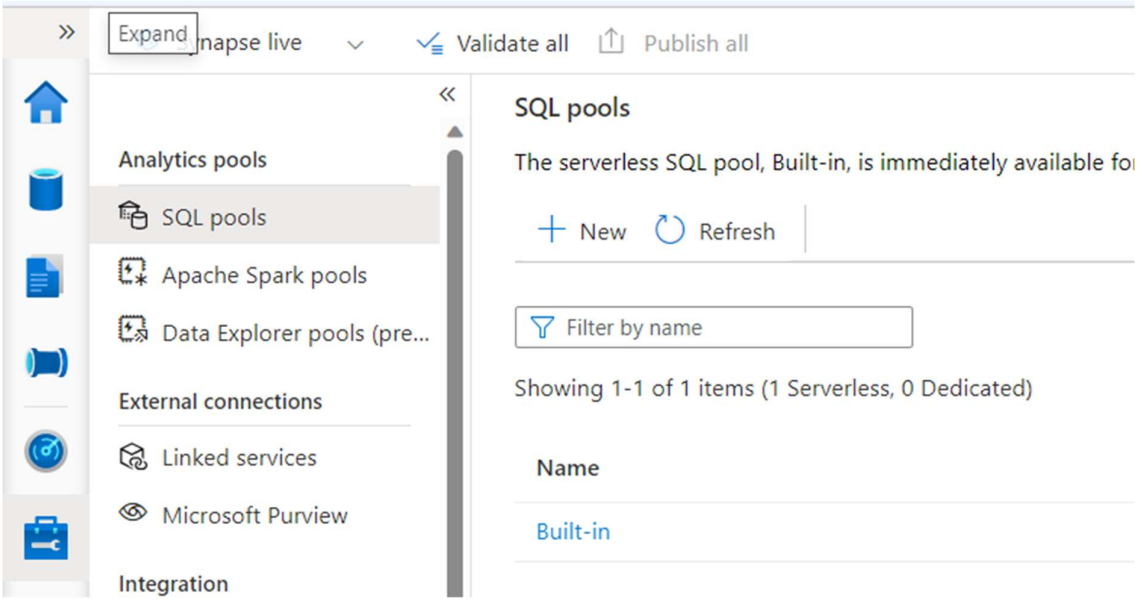


9. This is our synapse analytics workspace.

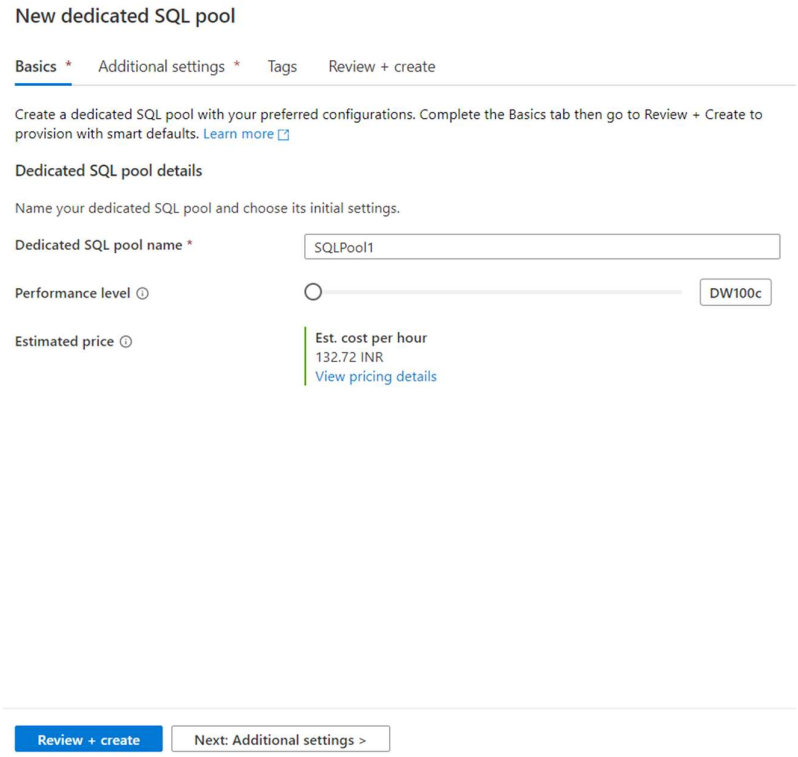


Create SQL Pool

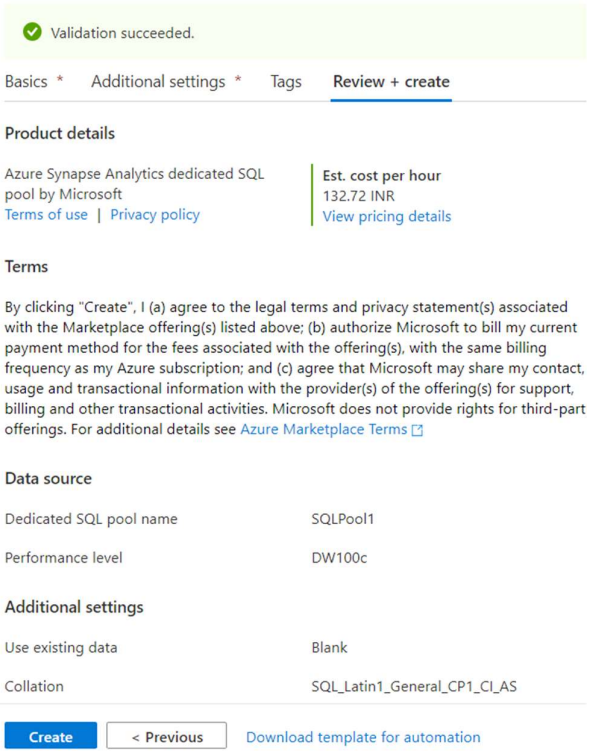
1. Under Manage, in the SQL Pool click on Plus symbol New.



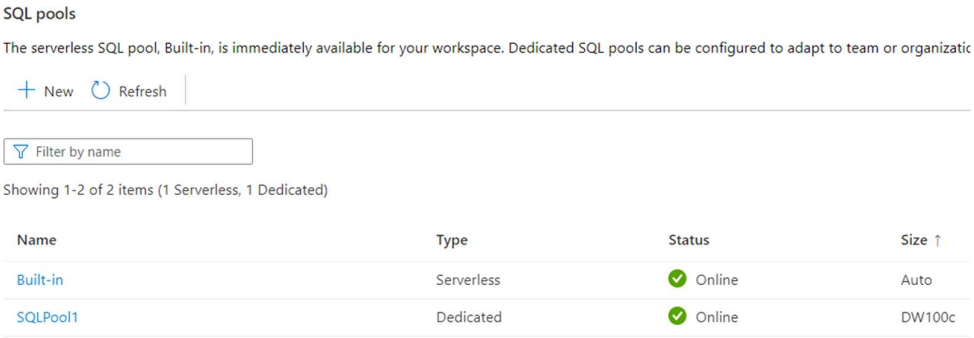
2. Give the name and set the performance level then click on Review + Create.



3. Now click on Create.

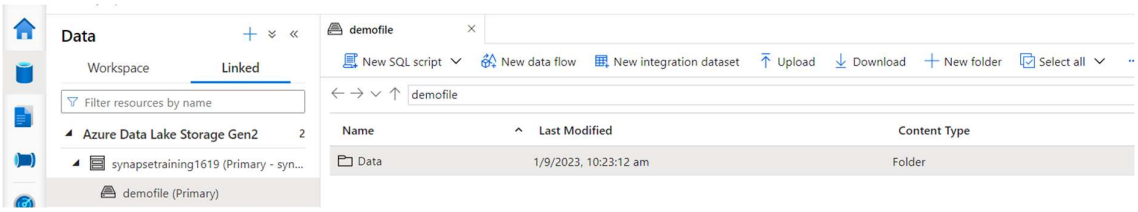


4. Here we have created a Dedicated SQLPool.

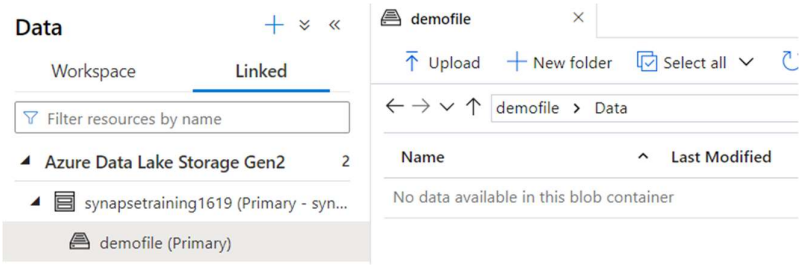


Analyze data with a dedicated SQL Pool

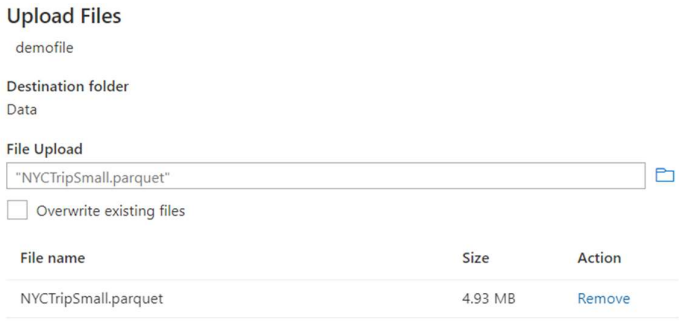
1. Under Data, in the Linked, go to demofile then create a folder name as Data using the New Folder option.



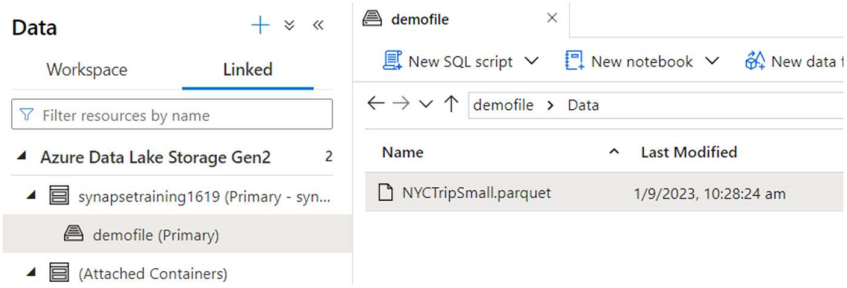
2. Go into the folder and click on the Upload option.



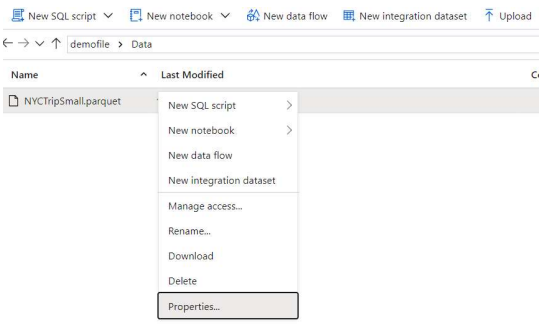
3. Select the file from your local system and click on Upload.



4. We are going to read this file using SQLPool.

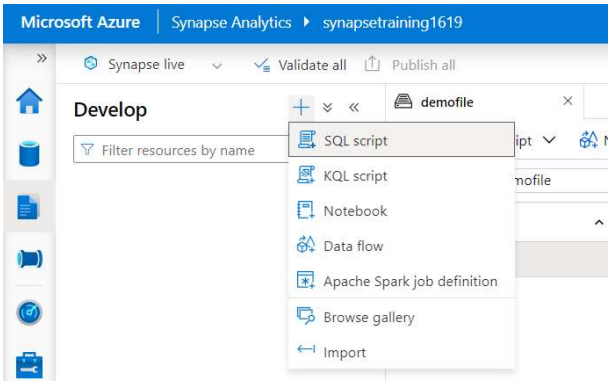


5. Right-click on it and click on Properties.



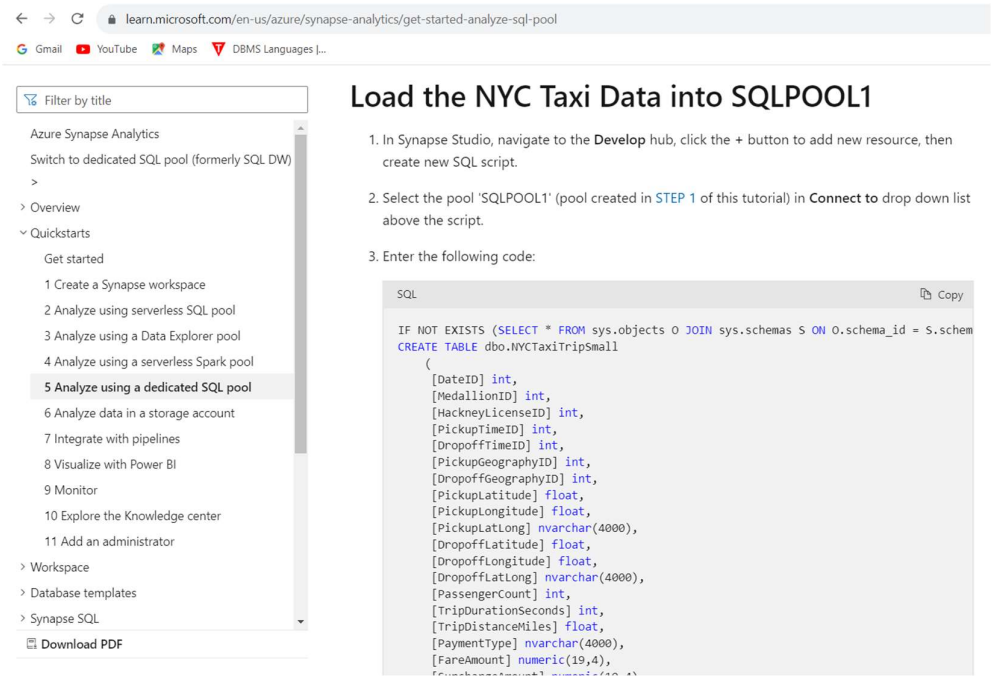
6. Copy the URL and paste it somewhere safe.

7. Under Develop click on the plus symbol and click on SQL script.

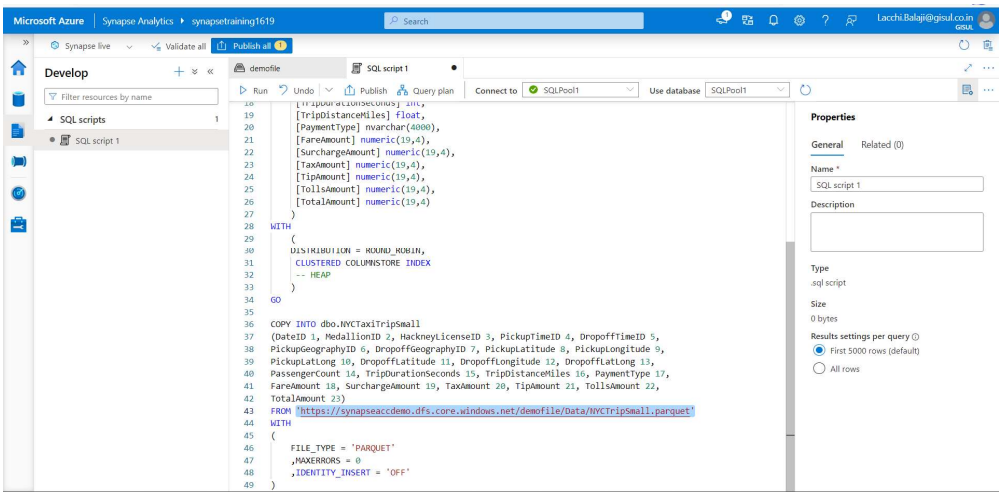


8. Under this link copy the below code.

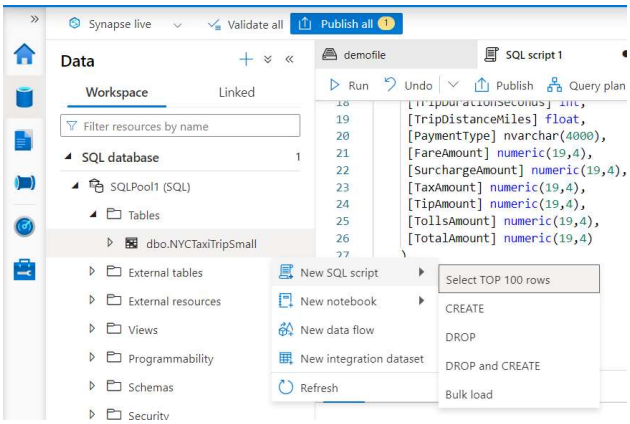
Link: https://learn.microsoft.com/en-us/azure/synapse-analytics/get-started-analyze-sql-pool



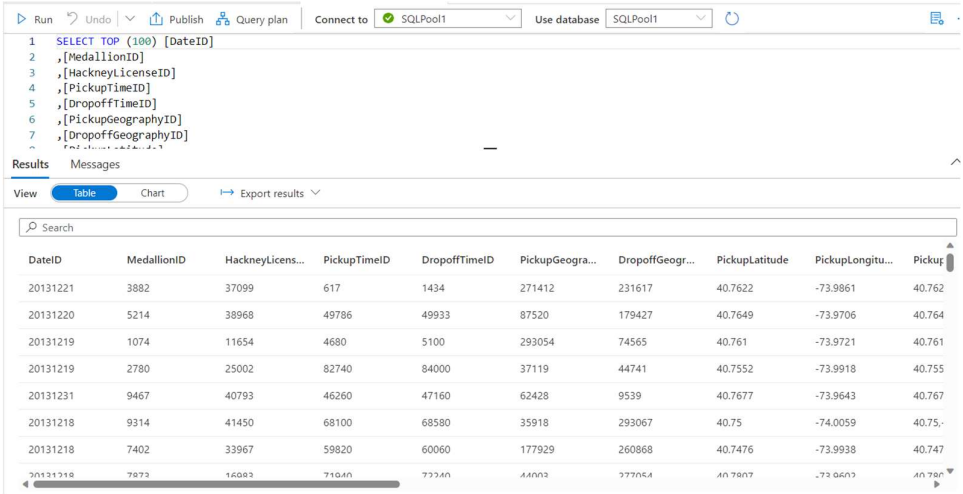
9. Paste it into SQL Script and connect to SQLPool1 then change the URL to the File URL that we copied in the previous step.
10. Next click on Run.



11. Now go to Data, In the Workspace under SQLPool click on the Tables and right click on the table go to New SQL Script, and click on Select TOP 100 rows.



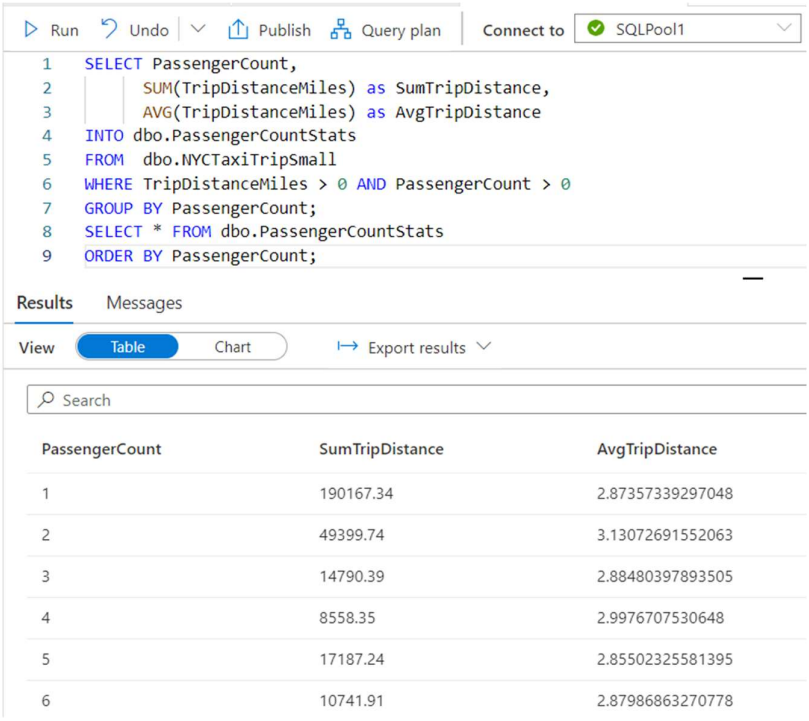
12. Here we can see the data inside that file.



13. Replace the text of the SQL script with this code and run it.

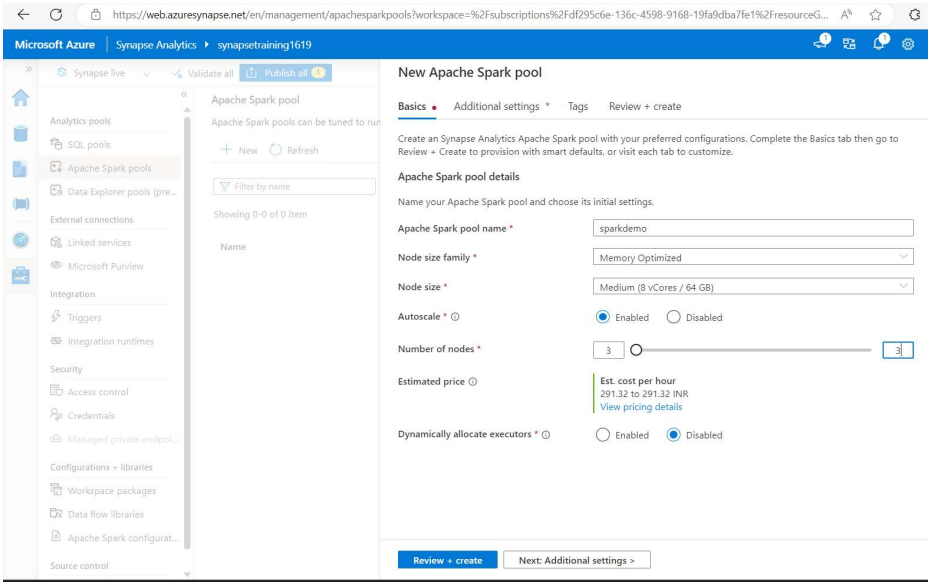
```
Query: SELECT PassengerCount,
    SUM(TripDistanceMiles) as SumTripDistance,
    AVG(TripDistanceMiles) as AvgTripDistance
INTO dbo.PassengerCountStats
FROM  dbo.NYCTaxiTripSmall
WHERE TripDistanceMiles > 0 AND PassengerCount > 0
GROUP BY PassengerCount;
SELECT * FROM dbo.PassengerCountStats
ORDER BY PassengerCount;
```

14. This query shows how the total trip distances and average trip distances relate to the number of passengers.

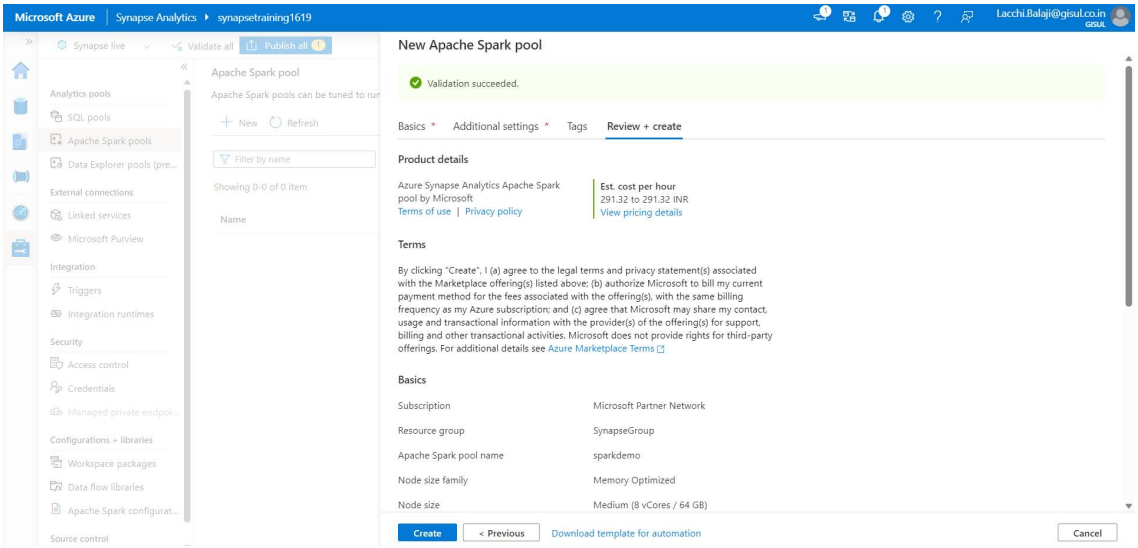


Create Pyspark Pool

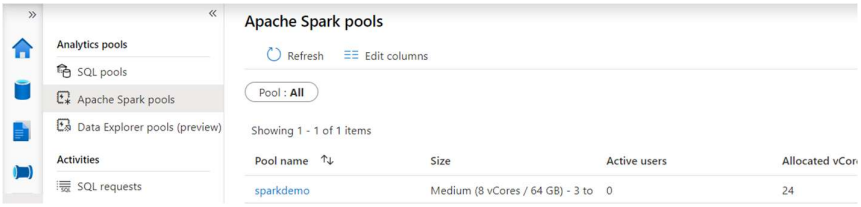
1. Under Manage, in the Apache Spark Pools click on the New plus symbol.
2. And give the properties as shown below. Now click on Review + Create.



3. Next click on Create.

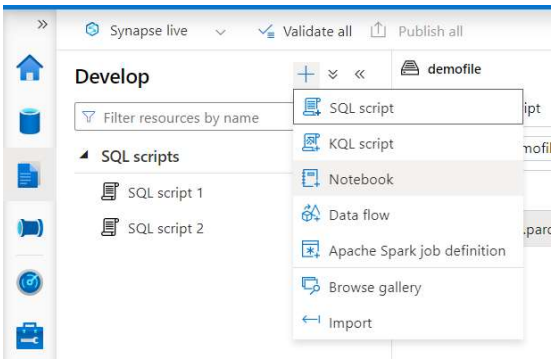


4. Here we have created our Pyspark pool.

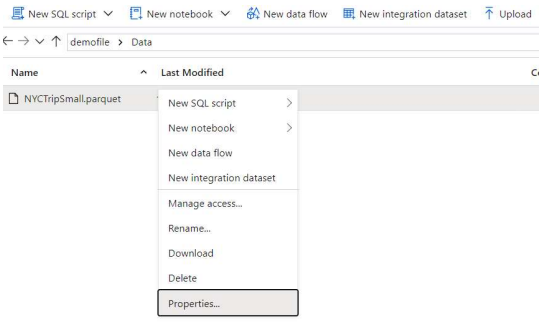


Analyze Data with Pyspark Pool

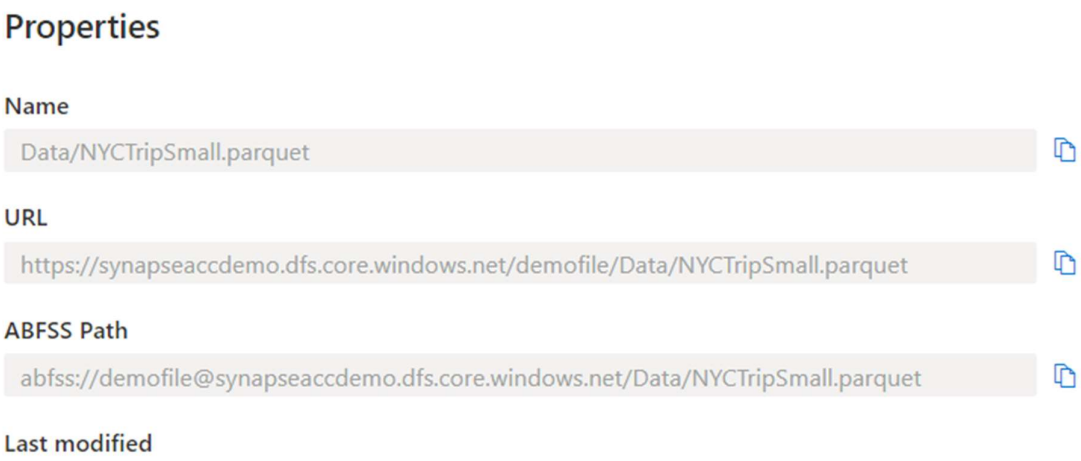
5. In the Develop click on the plus symbol and click on a notebook.



15. Right-click on the file and click on Properties.



16. Copy the ABFSS Path and paste it somewhere safe.



17. In the Notebook set the sparkdemo Attack.
18. And paste the below query. But in the query give the path you copied in the previous step.
19. Next click on the Run Cell option.

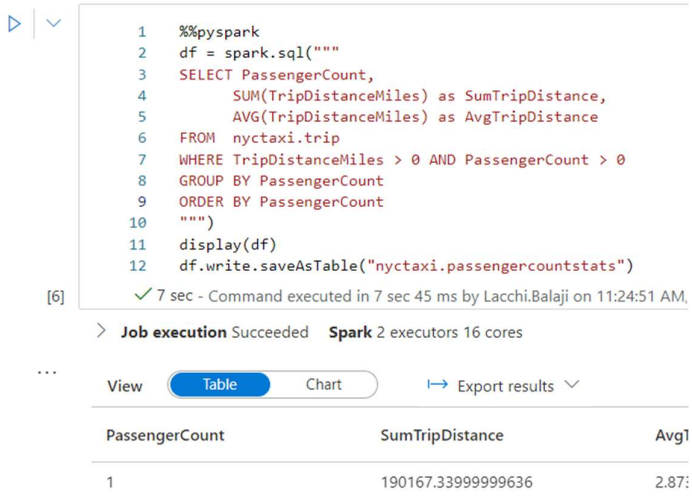
Query: %%pyspark
df =
spark.read.load(path='abfss://demofile@synapseaccdemo.dfs.core.windows.net/Data/NYCTripSmall.parquet')

20. Next click on the Plus symbol code and give the below code.

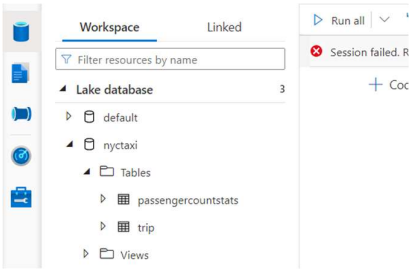
Code: spark.sql("create Database if not exists nyctaxi")
df.write.mode("overwrite").saveAsTable("nyctaxi.trip")

21. Click on the Plus symbol of the code and give the below code.

Code: %%pyspark
df = spark.sql("""
SELECT PassengerCount,
SUM(TripDistanceMiles) as SumTripDistance,
AVG(TripDistanceMiles) as AvgTripDistance
FROM nyctaxi.trip
WHERE TripDistanceMiles > 0 AND PassengerCount > 0
GROUP BY PassengerCount
ORDER BY PassengerCount
""")
display(df)
df.write.saveAsTable("nyctaxi.passengercountstats")

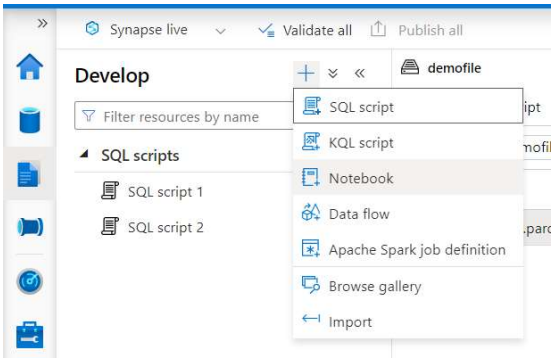


22. Under the pyspark pool we have a table.



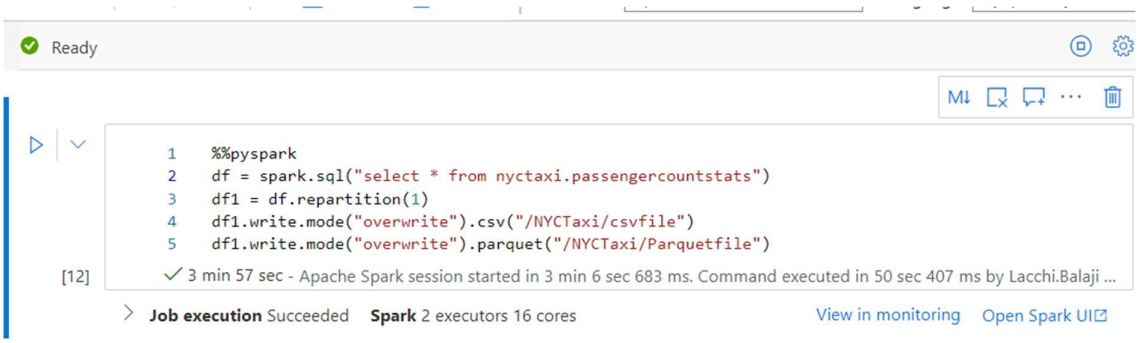
Analyze data in the Storage Account

1. In this example we are converting the passengercountstatus table into csv and store in the Folder.
2. Create a Notebook.

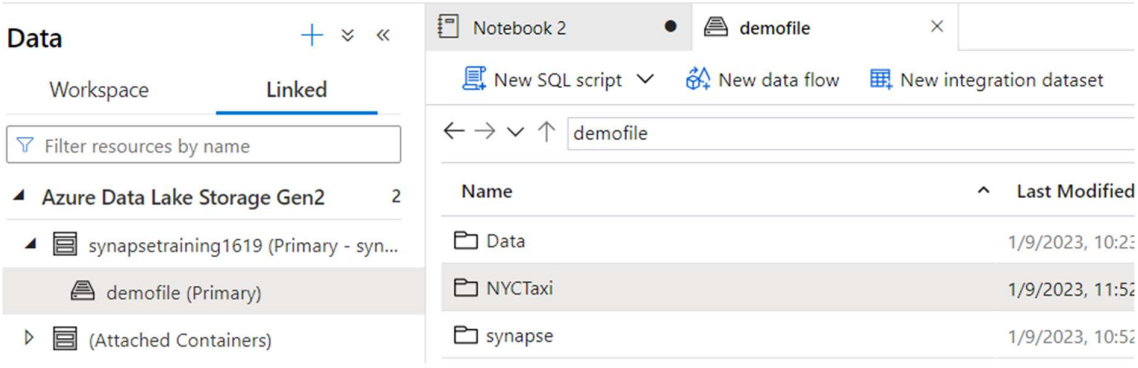


3. Give the below code and run it.

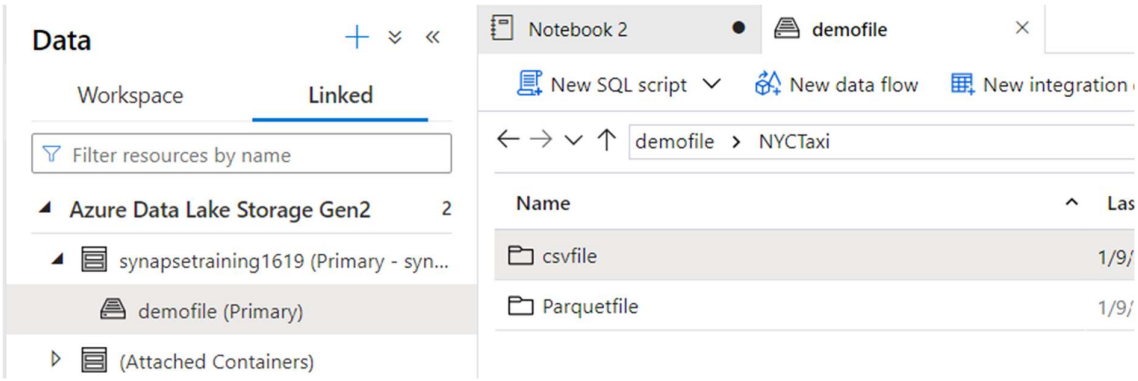
Code: %%pyspark
df = spark.sql("select * from nyctaxi.passengercountstats")
df1 = df.repartition(1)
df1.write.mode("overwrite").csv("/NYCTaxi/csvfile")
df1.write.mode("overwrite").parquet("/NYCTaxi/Parquetfile")



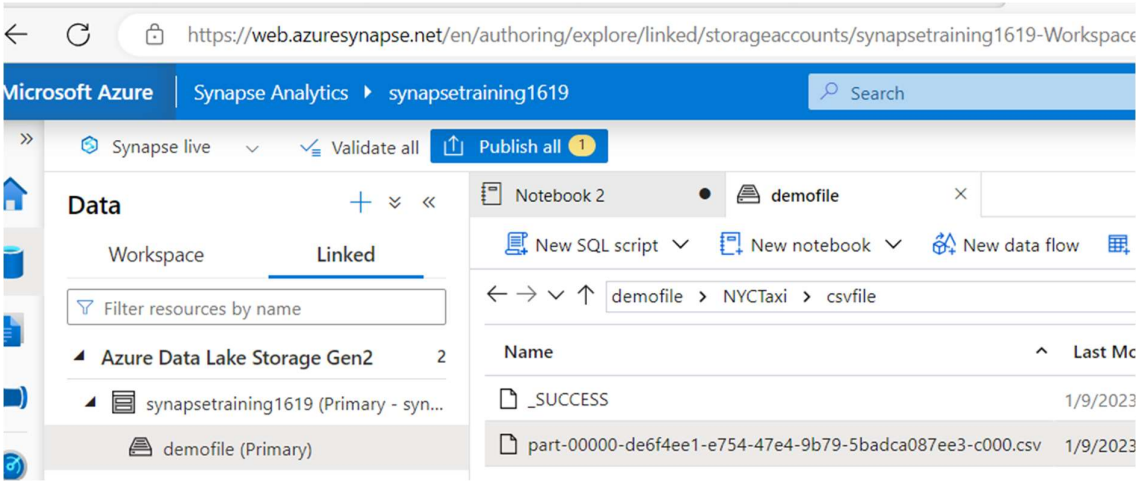
4. Now go to Data and check the folder.



5. Go into the folder and you will see a CSV folder.

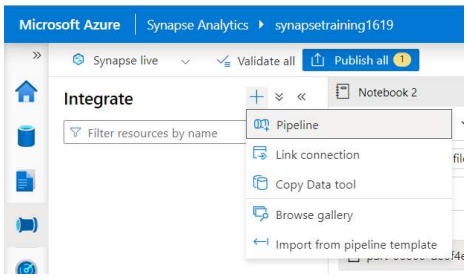


6. Go into the CSV folder and double-click on the file, the csv file automatically will download.

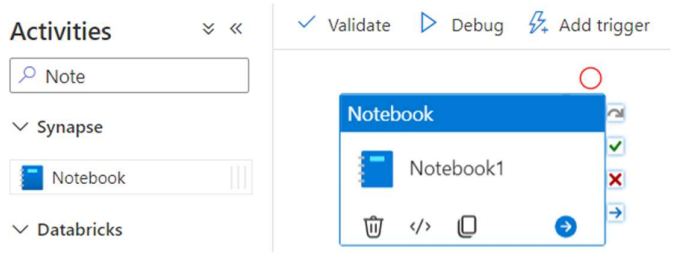


Integrate Pipelines

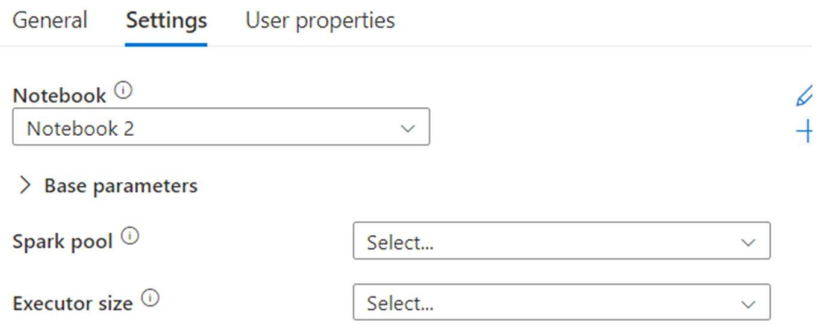
1. In the Integrate click on plus symbol and click on Pipeline.



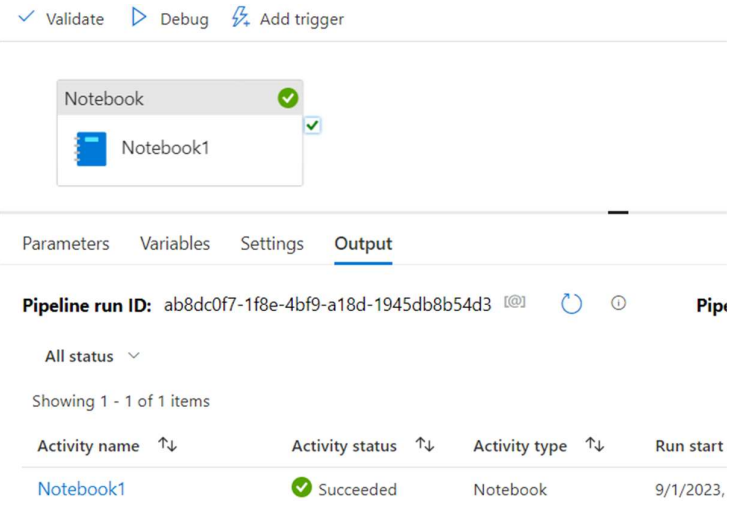
2. Drag and drop the Notebook.



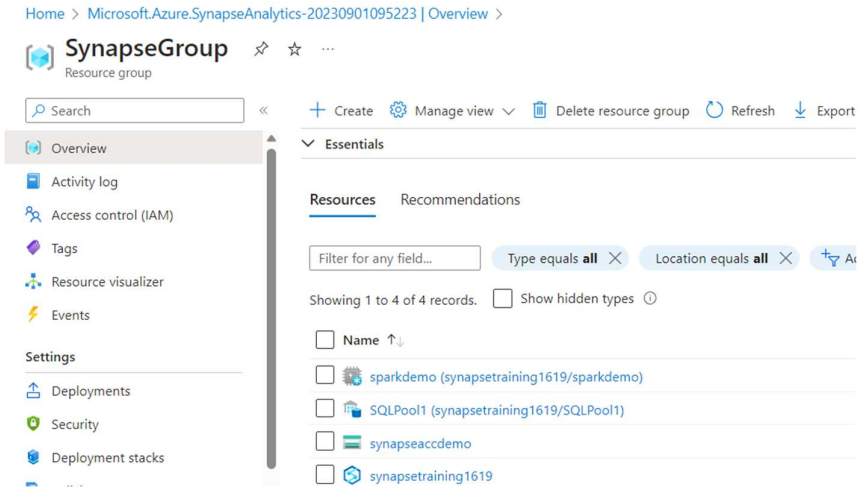
3. Select the Notebook that we created in the previous example.



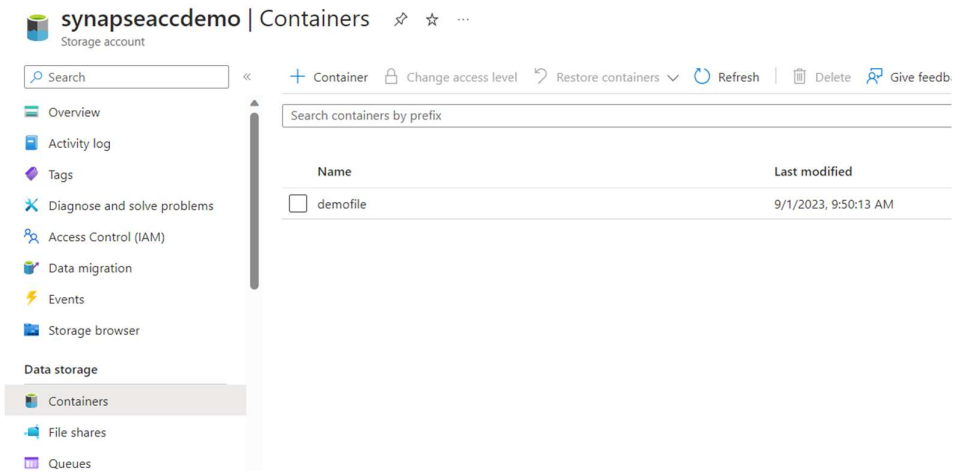
4. Here our pipeline executed successfully.



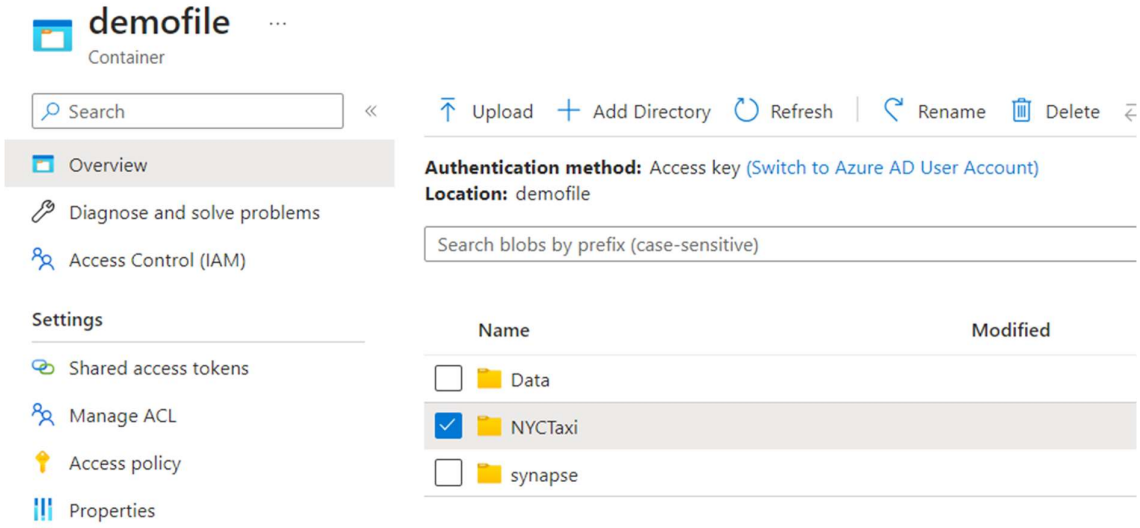
5. Click on the suynapseaccdemo storage account.



6. Under container click on Demo file.

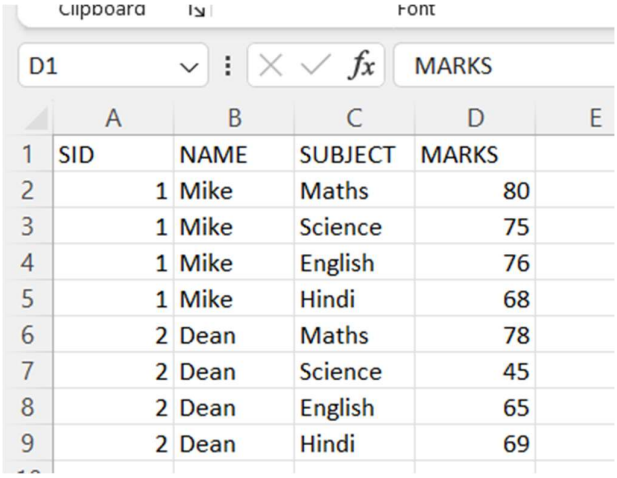


7. You will see the folder that we created in the notebook.



Mapping Data Flow

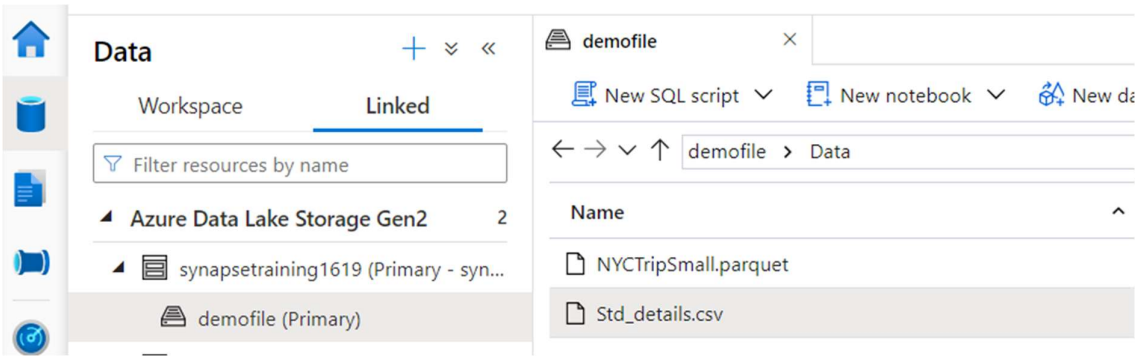
1. In this Example we are going to use the below csv file data.



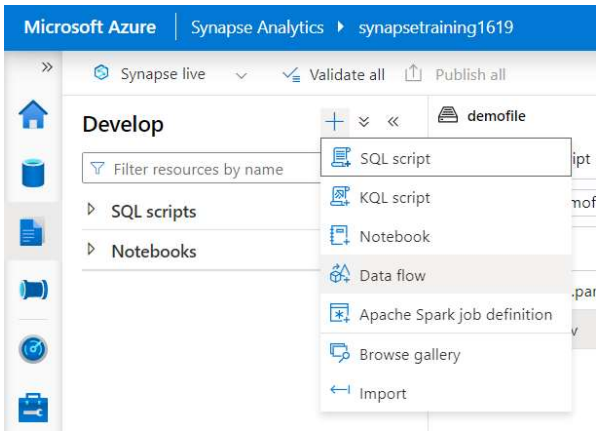
D1 | MARKS

	A	B	C	D	E
1	SID	NAME	SUBJECT	MARKS	
2	1	Mike	Maths	80	
3	1	Mike	Science	75	
4	1	Mike	English	76	
5	1	Mike	Hindi	68	
6	2	Dean	Maths	78	
7	2	Dean	Science	45	
8	2	Dean	English	65	
9	2	Dean	Hindi	69	

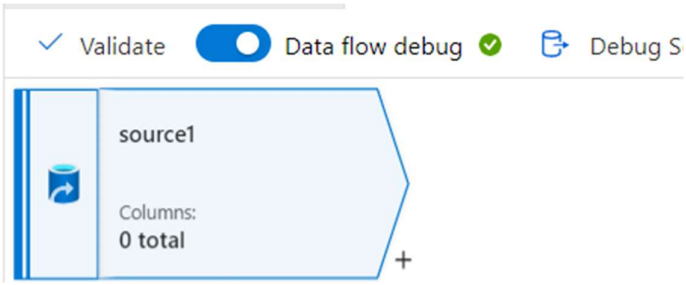
2. I uploaded my file into the Data folder.



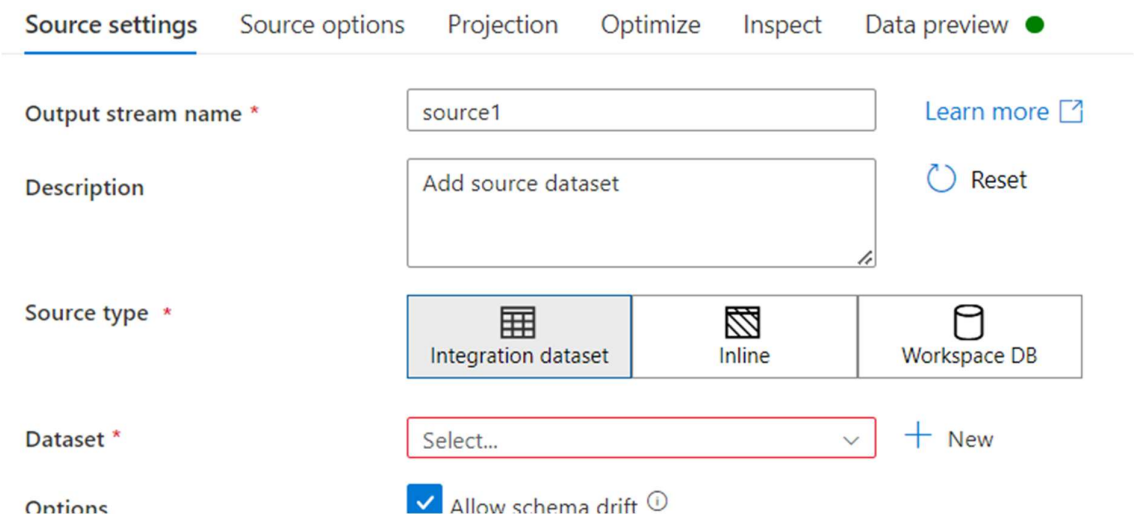
3. Now go to Develop and click on the plus symbol then click on Dataflow.



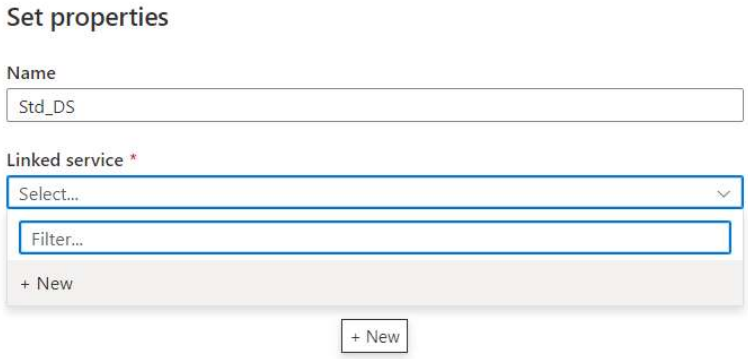
4. Add a Data source.



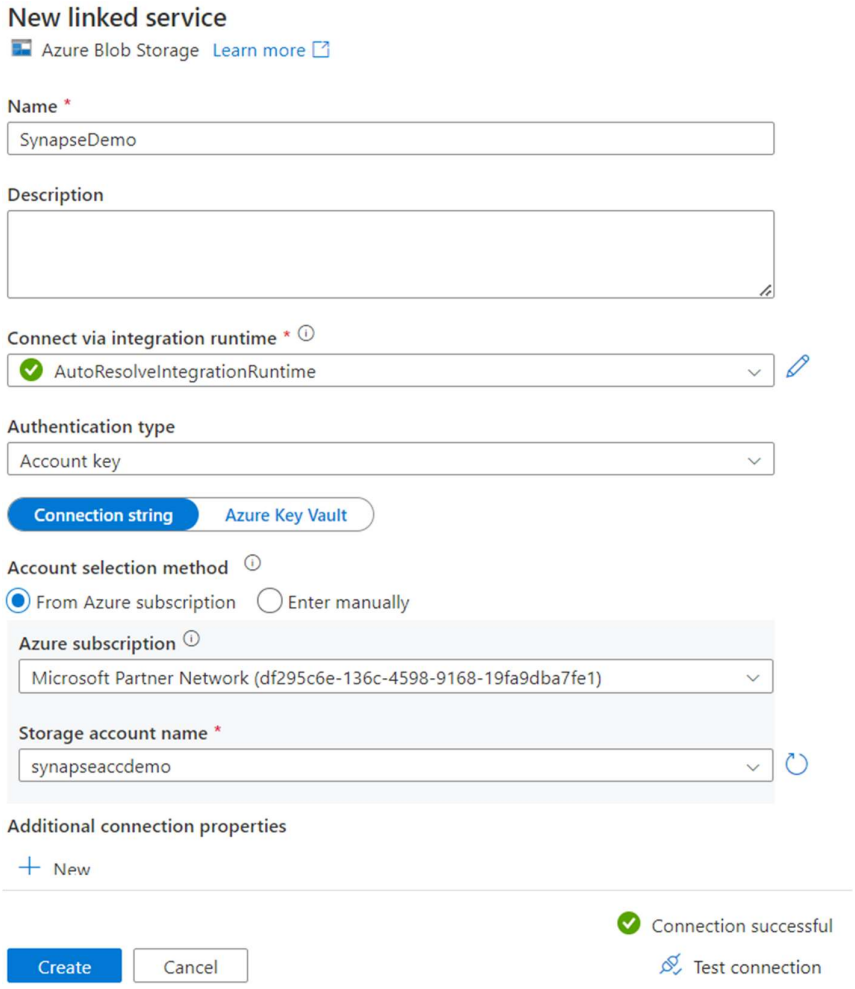
5. Click on the Plus symbol to create a Dataset.



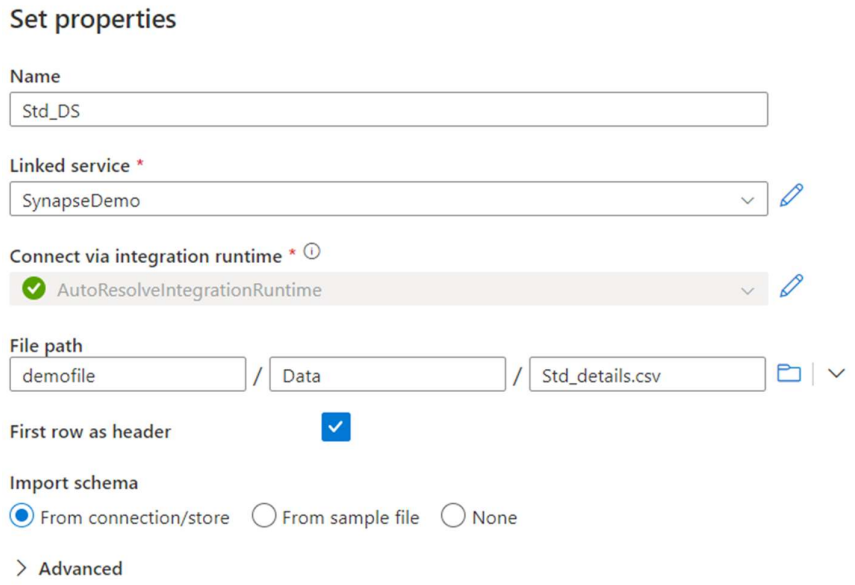
6. Select the Azure Blob Storage and click on continue.
7. Then select Delimited text and click on Continue.
8. Give the name and click on the plus symbol to create a New linked service.



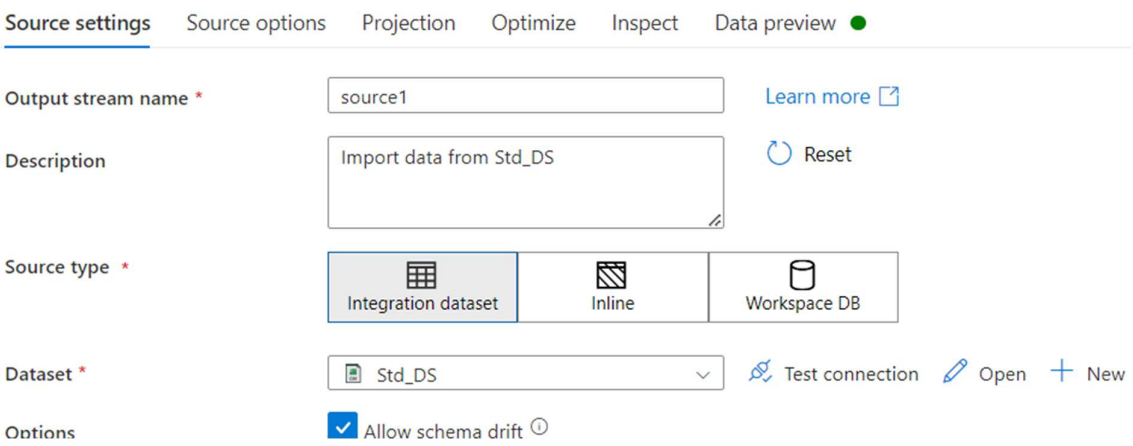
9. Set the below properties and click on Create.



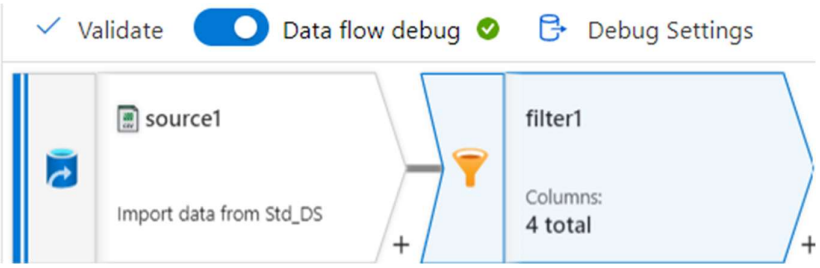
10. Select the file and click on OK.



11. Here we have assigned our source dataset.

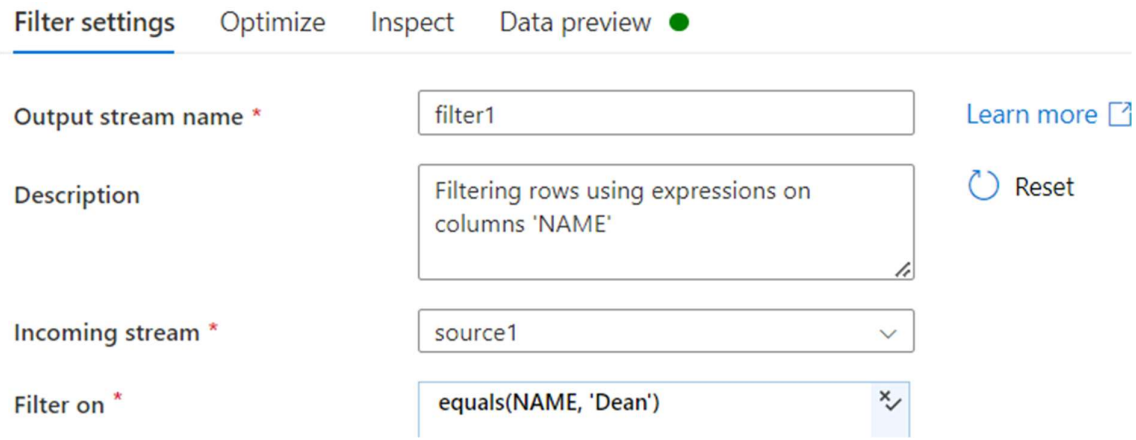


12. Add filter Transformation.

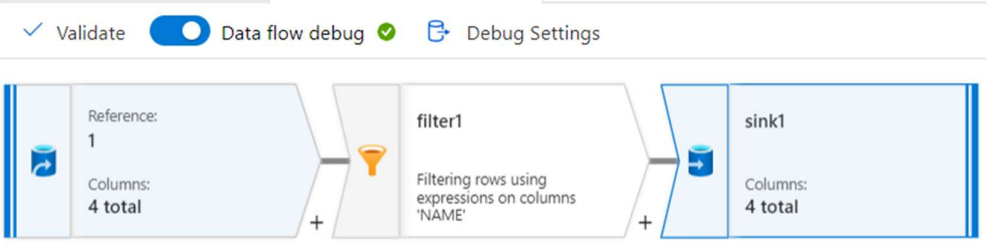


13. In the Filter Settings, under Filter On give the below expression.

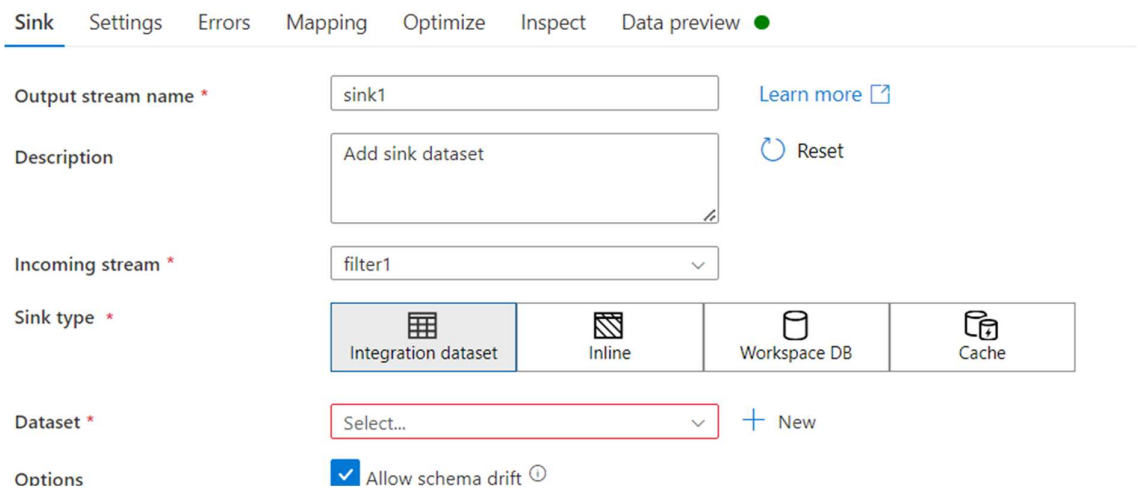
Exp: equals(NAME, 'Dean')



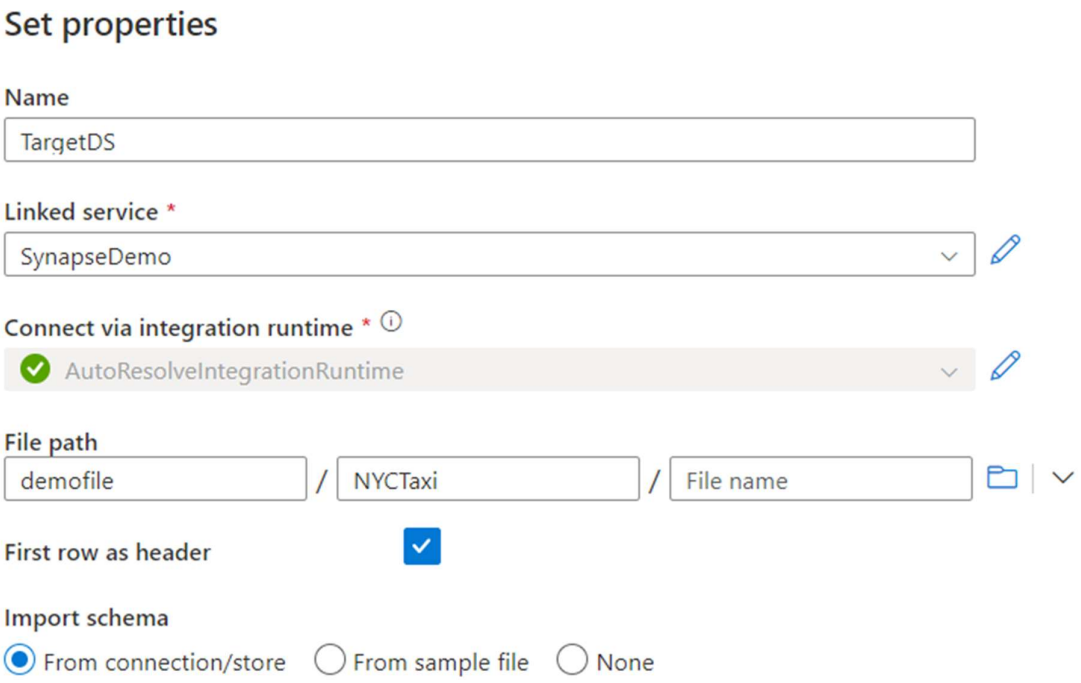
14. Add Sink Transformation.



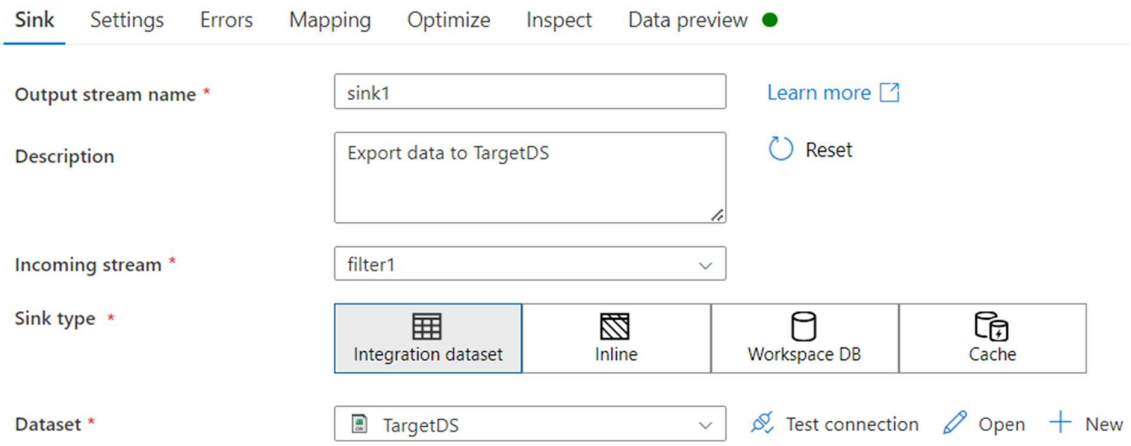
15. Click on the Plus symbol to create a Target Dataset.



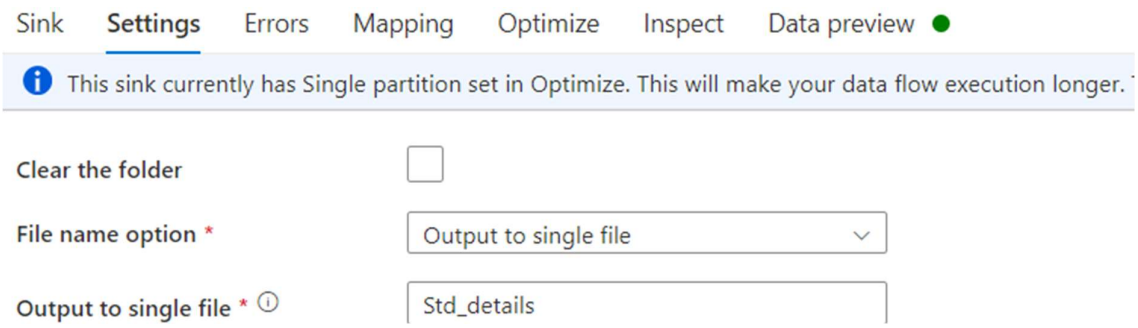
16. Select the Azure Blob Storage and click on continue.
17. Then select Delimited text and click on Continue.
18. Give the name, linked service, and file path. Don't give a file just give the location.



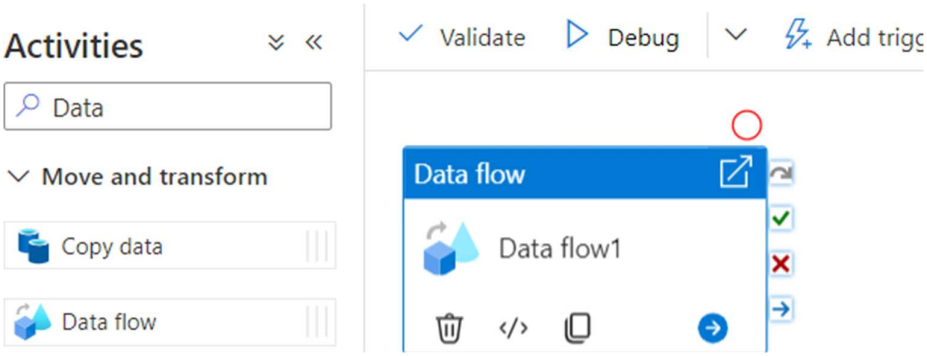
19. Here we assigned our Target dataset.



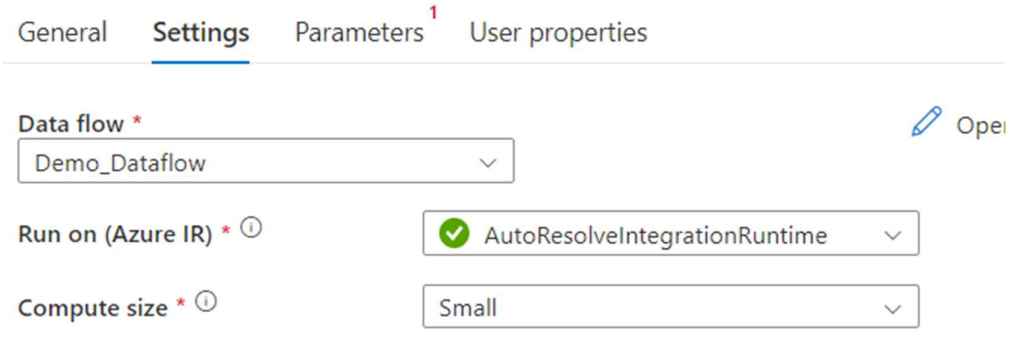
20. Under Settings select the file option and give the file name.



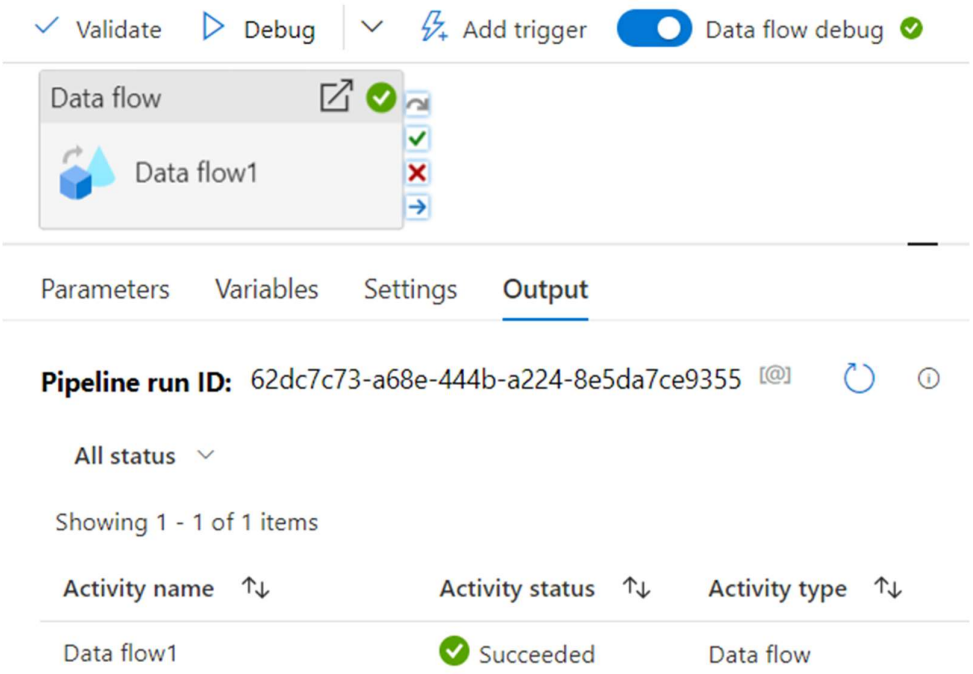
21. Create a Pipeline, Drag and Drop the data flow.



22. Under Settings select the Dataflow.

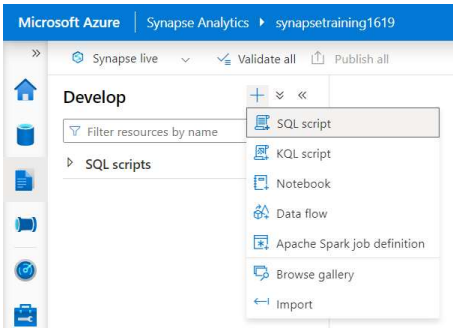


23. Validate it and click on Debug.
24. Here our pipeline executed successfully.



Querying and Analysing Data

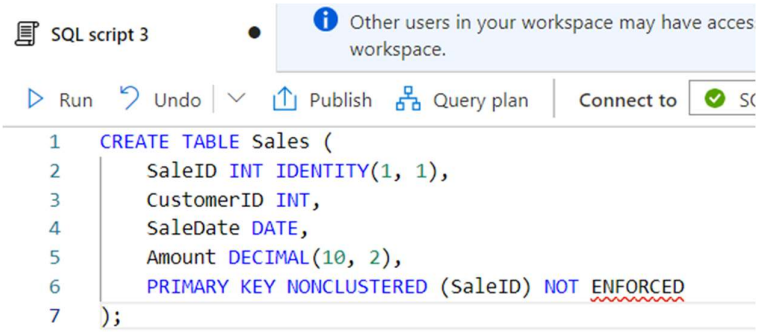
1. Under develop click on plus symbol and click on SQL Script.



2. Create a Table with the below query.

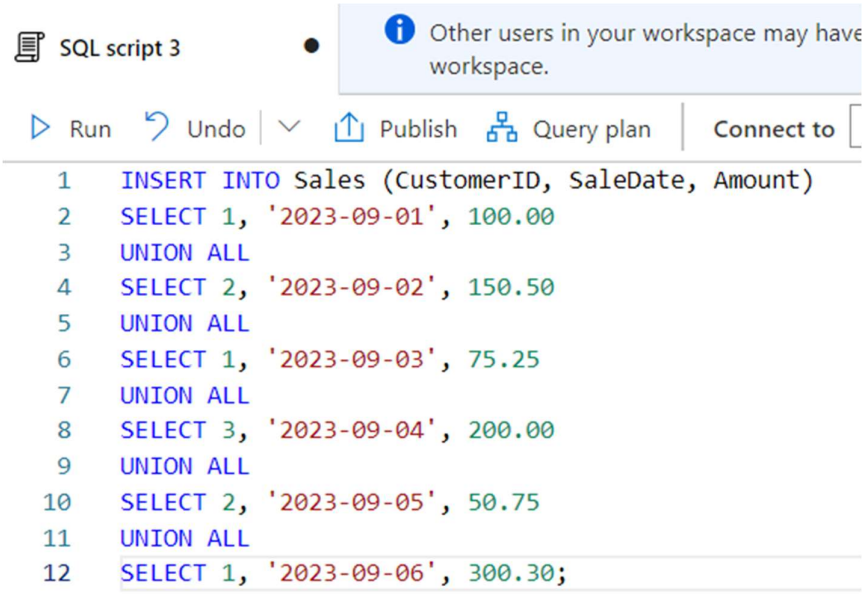
Query: CREATE TABLE Sales (
  SaleID INT IDENTITY(1, 1),
  CustomerID INT,
  SaleDate DATE,
  Amount DECIMAL(10, 2),
  PRIMARY KEY NONCLUSTERED (SaleID) NOT ENFORCED
);

3. Give the above query and click on Run.



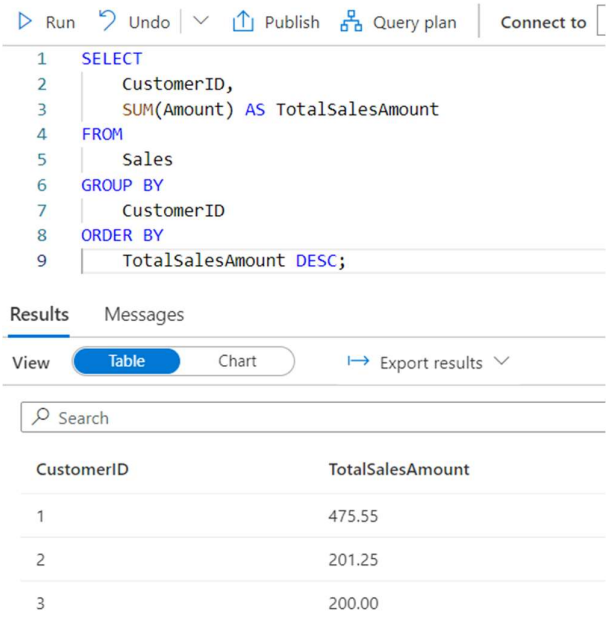
4. Insert sample data.

Query: INSERT INTO Sales (CustomerID, SaleDate, Amount)
SELECT 1, '2023-09-01', 100.00
UNION ALL
SELECT 2, '2023-09-02', 150.50
UNION ALL
SELECT 1, '2023-09-03', 75.25
UNION ALL
SELECT 3, '2023-09-04', 200.00
UNION ALL
SELECT 2, '2023-09-05', 50.75
UNION ALL
SELECT 1, '2023-09-06', 300.30;



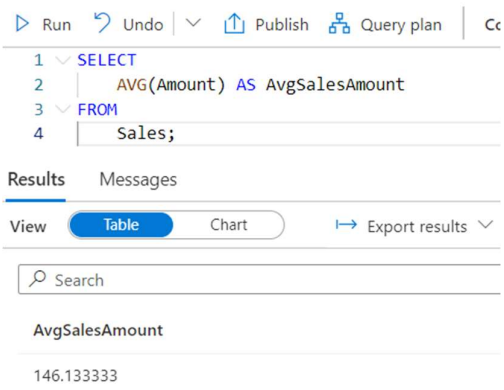
5. Now, let's perform some SQL queries to extract insights from the sample data.
6. Total Sales Amount by Customer.

Query: SELECT
  CustomerID,
  SUM(Amount) AS TotalSalesAmount
FROM
  Sales
GROUP BY
  CustomerID
ORDER BY
  TotalSalesAmount DESC;



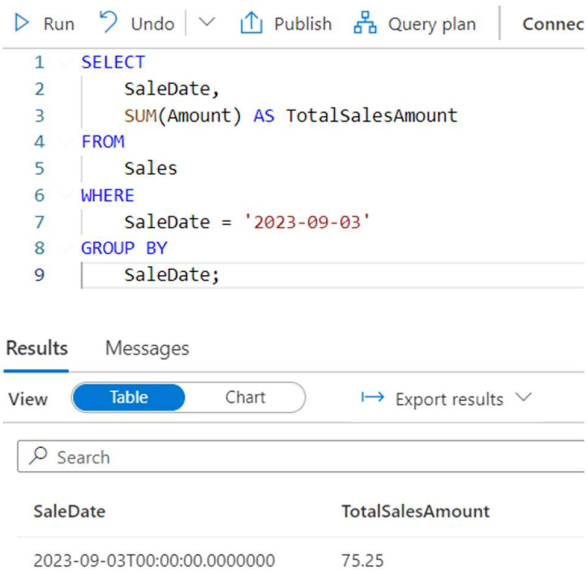
7. This query will give you the total sales amount for each customer, ordered from the highest total sales to the lowest.
8. Average Sales Amount per Transaction.

Query: SELECT
  AVG(Amount) AS AvgSalesAmount
FROM
  Sales;



9. This query calculates the average sales amount per transaction.
10. Sales on a Specific Date.

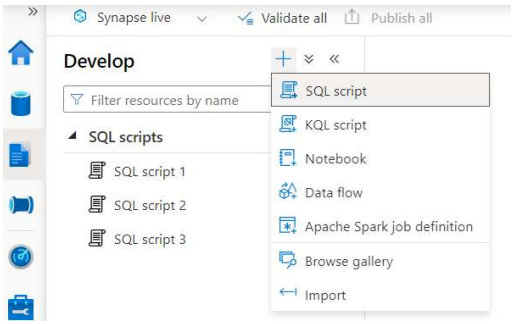
Query: SELECT
  SaleDate,
  SUM(Amount) AS TotalSalesAmount
FROM
  Sales
WHERE
  SaleDate = '2023-09-03'
GROUP BY
  SaleDate;



11. This query shows the total sales amount on a specific date (in this case, '2023-09-03').
12. You can execute these queries in Azure Synapse Studio or any SQL client tool connected to your Azure Synapse SQL Data Warehouse to analyze the sample data and gain meaningful insights.

Write and Execute Queries

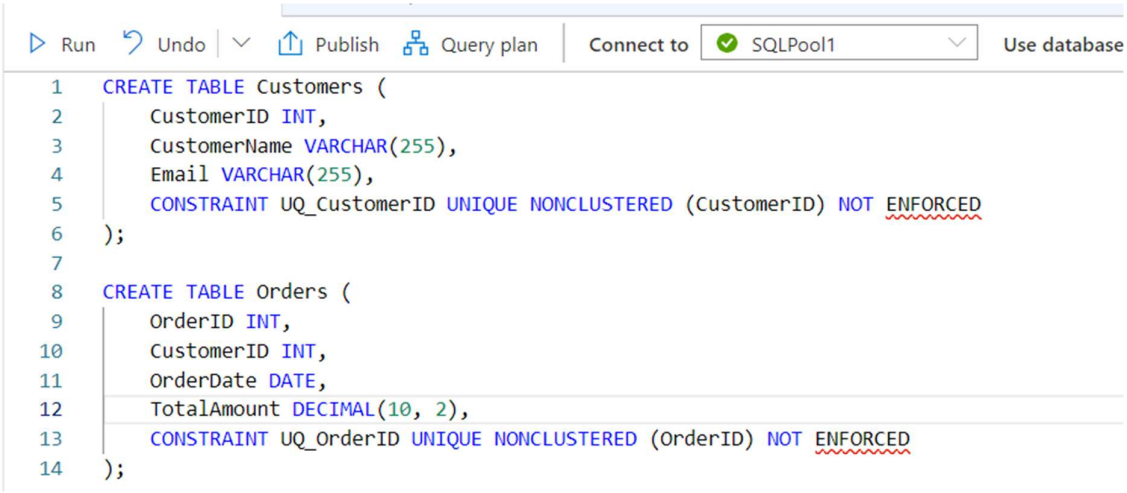
1. Create another SQL Script.



2. Create two sample tables: "Customers" and "Orders."

Query: CREATE TABLE Customers (
  CustomerID INT,
  CustomerName VARCHAR(255),
  Email VARCHAR(255),
  CONSTRAINT UQ_CustomerID UNIQUE NONCLUSTERED (CustomerID) NOT ENFORCED
);

CREATE TABLE Orders (
  OrderID INT,
  CustomerID INT,
  OrderDate DATE,
  TotalAmount DECIMAL(10, 2),
  CONSTRAINT UQ_OrderID UNIQUE NONCLUSTERED (OrderID) NOT ENFORCED
);

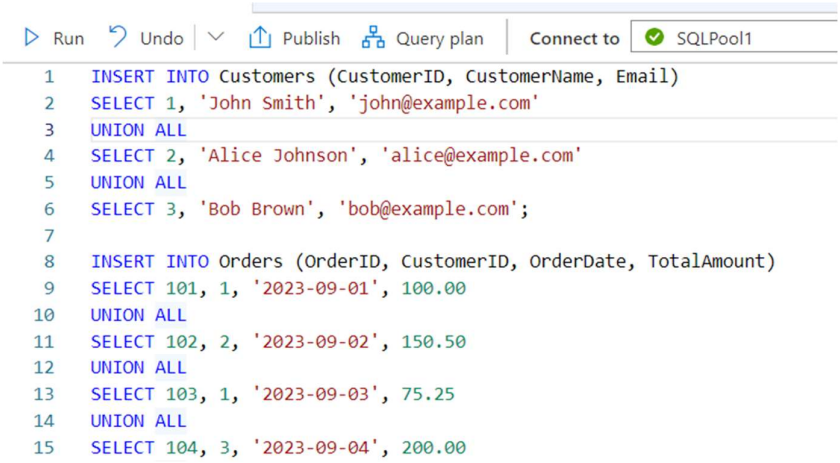


3. Now, let's insert some sample data into these tables.

Query: INSERT INTO Customers (CustomerID, CustomerName, Email)
SELECT 1, 'John Smith', 'john@example.com'
UNION ALL
SELECT 2, 'Alice Johnson', 'alice@example.com'
UNION ALL
SELECT 3, 'Bob Brown', 'bob@example.com';

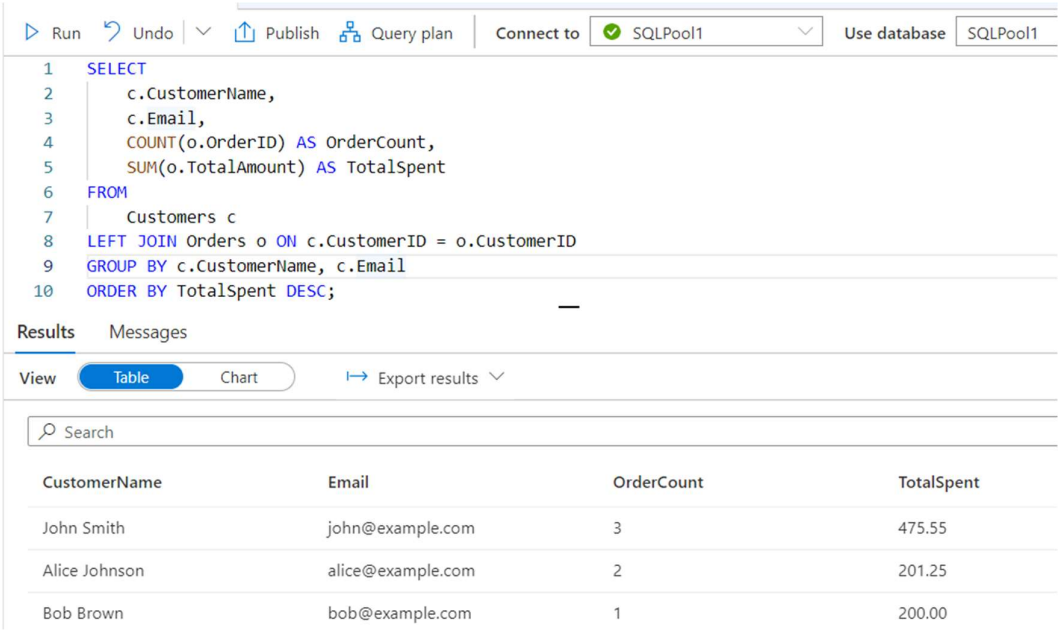
INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
SELECT 101, 1, '2023-09-01', 100.00
UNION ALL
SELECT 102, 2, '2023-09-02', 150.50
UNION ALL
SELECT 103, 1, '2023-09-03', 75.25
UNION ALL
SELECT 104, 3, '2023-09-04', 200.00
UNION ALL
SELECT 105, 2, '2023-09-05', 50.75

UNION ALL
SELECT 106, 1, '2023-09-06', 300.30;



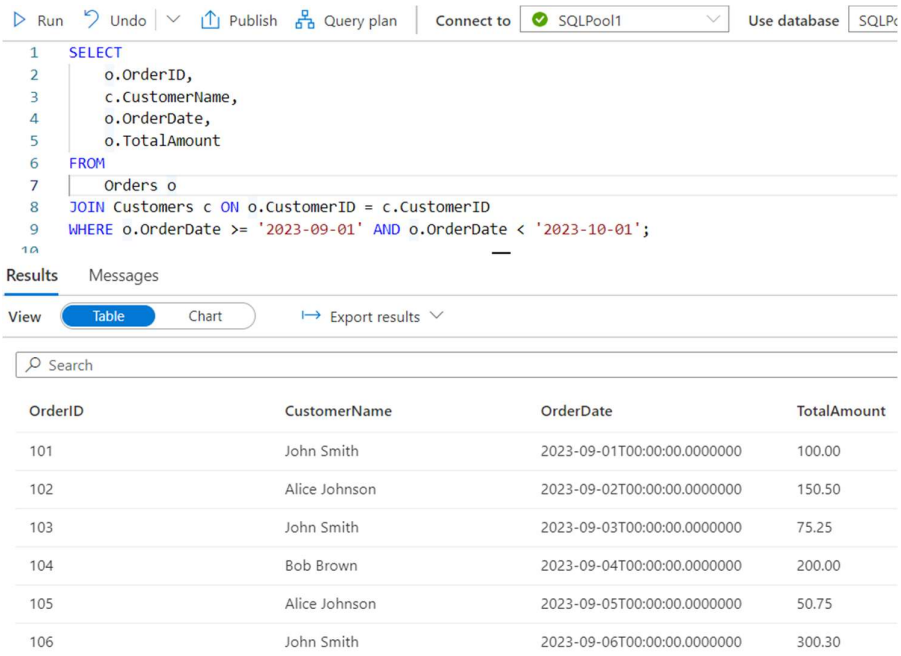
4. With the tables and data in place, let's write some SQL queries to analyze the data.
5. Retrieve Customer Information with Orders.
6. This query retrieves customer information along with the number of orders and the total amount spent by each customer.

Query: SELECT
  c.CustomerName,
  c.Email,
  COUNT(o.OrderID) AS OrderCount,
  SUM(o.TotalAmount) AS TotalSpent
FROM
  Customers c
LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerName, c.Email
ORDER BY TotalSpent DESC;



7. Retrieve Orders Placed in September 2023.
8. This query retrieves all orders placed in September 2023.

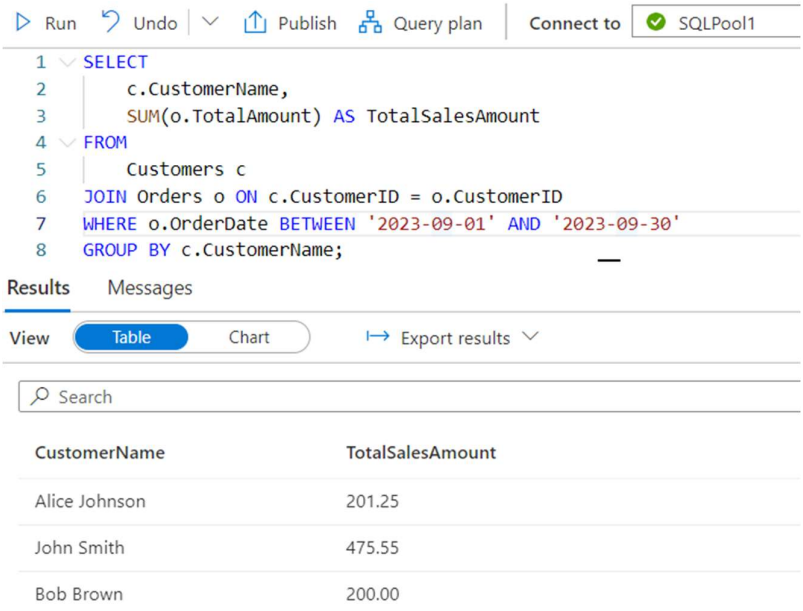
Query: SELECT
  o.OrderID,
  c.CustomerName,
  o.OrderDate,
  o.TotalAmount
FROM
  Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
WHERE o.OrderDate >= '2023-09-01' AND o.OrderDate < '2023-10-01';



9. Without Indexing.
10. This query represents the initial query without any indexing or optimization. It may perform slowly, especially on a large dataset.

Query: SELECT
  c.CustomerName,
  SUM(o.TotalAmount) AS TotalSalesAmount
FROM
  Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.OrderDate BETWEEN '2023-09-01' AND '2023-09-30'
GROUP BY c.CustomerName;

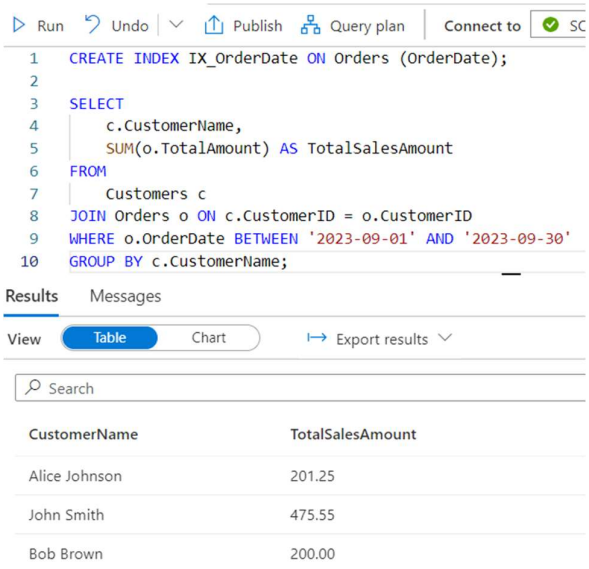
11. Give the Query and click on Run.



12. With Appropriate Indexing.
13. It adds an index on the OrderDate column in the Orders table. This index can significantly improve performance for date-range queries.

Query: CREATE INDEX IX_OrderDate ON Orders (OrderDate);

```
SELECT
  c.CustomerName,
  SUM(o.TotalAmount) AS TotalSalesAmount
FROM
  Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.OrderDate BETWEEN '2023-09-01' AND '2023-09-30'
GROUP BY c.CustomerName;
```



14. With Columnstore Index.
15. It introduces a clustered columnstore index on the Orders table. Columnstore indexes are optimized for large analytical queries and can provide excellent performance for aggregations.

Query: CREATE CLUSTERED COLUMNSTORE INDEX IX_Columnstore ON Orders;

```
SELECT
  c.CustomerName,
  SUM(o.TotalAmount) AS TotalSalesAmount
FROM
  Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
WHERE o.OrderDate BETWEEN '2023-09-01' AND '2023-09-30'
GROUP BY c.CustomerName;
```