### **Final Project - Comparative Document: Azure Data Solutions Based on Use Cases**

### **Scenario and Solutions:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S No.** | **Scenario** | **Data Services used** | **Purpose** |
| 1. | **IoT Real-Time Data Ingestion and Analysis** | Event Hub, Stream Analytics, Cosmos DB | To monitor and predict average temperature detected by IoT sensors |
| 2. | **Batch Processing of Structured Data:**   1. **with loading data into synapse** 2. **without loading data into synapse** | ADLS, Dedicated SQL Pool, Serverless SQL Pool | Monthly reporting on historical data |
| 3. | **To conduct large-scale transformations on data (batch mode) providing high performance, scalability and multiple language support features.** | ADLS, Azure Synapse Spark Pool, Azure Synapse Pipelines | Regional and Monthly Sales performance Analysis of a company |

Let’s see a detailed comparison of various Azure data services like Azure Synapse Serverless SQL Pool, Dedicated SQL Pool, Spark Pool, Cosmos DB, Event hub and Stream Analytics and find which services are ideal for different data scenarios based on some factors.

### **Scenario 1: IoT Real-Time Data Ingestion and Analysis**

**Use Case:** A company needs to build a solution that will process real-time streaming data from 3 IoT sensors in order to detect the average temperature and humidity of the room.

#### **Solution/Tools used:**

|  |  |  |
| --- | --- | --- |
| **Tool used:** | **Purpose:** | **Why we choose this tool in this solution:** |
| **Azure Event Hub:** | Ingest real-time streaming data from multiple IoT sensors. | * To ingest large volumes of real-time streaming data from multiple sensors. * A scalable and reliable way to handle real time data without delays. |
| **Azure Stream Analytics:** | Process and analyze streaming data in near real-time and use queries to detect average temperature and humidity of the room. | * For timely processing of streaming data in near real-time, fast decision-making and responsive actions. |
| **Azure Cosmos DB:** | Store the processed results (avg temp and humidity) in containers for further process. | * Provides Low-latency/real time data access. * High scalability to handle large amounts of incoming sensor data. |

#### **Steps to follow:**

**1. Set Up Azure Event Hub for Data Ingestion:**

* **Create an** **Event Hub Namespace and an Event Hub within the Namespace:**
  + Event Hub Namespace: finalprojectnamespace
  + Event Hub: finalprojecteventhub

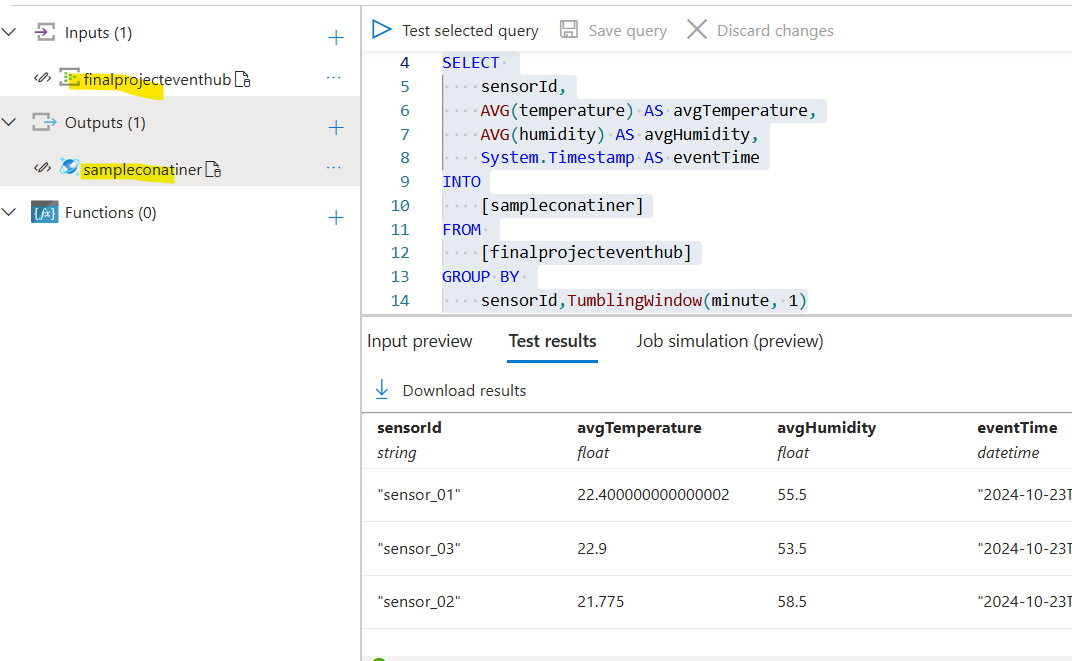
**2. Set Up Cosmos DB for storing output/processed Data:**

* **Create a Cosmos DB Account with a Database and Container:**
  + Cosmos DB Account: finalrojectcosmosaccount
  + Database and Container: sampledb / sampleconatiner
  + Set the partition key as: sensorId

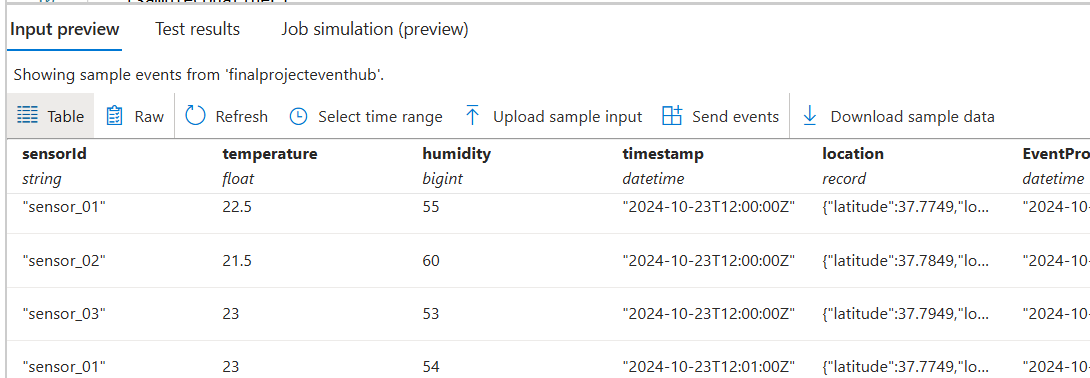
**3. Configure Azure Stream Analytics for Real-Time Data Processing:**

* **Create a Stream Analytics Job and define input, output and the query to process data:**
  + Job: finalprojectstreamanalyticsjob
  + Input: finalprojecteventhub
  + Output: sampleconatiner in cosmos DB
  + Query: Write query to find the average temperature and humidity of the room.
  + Test and save the query.

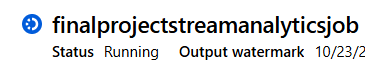
Query test results:



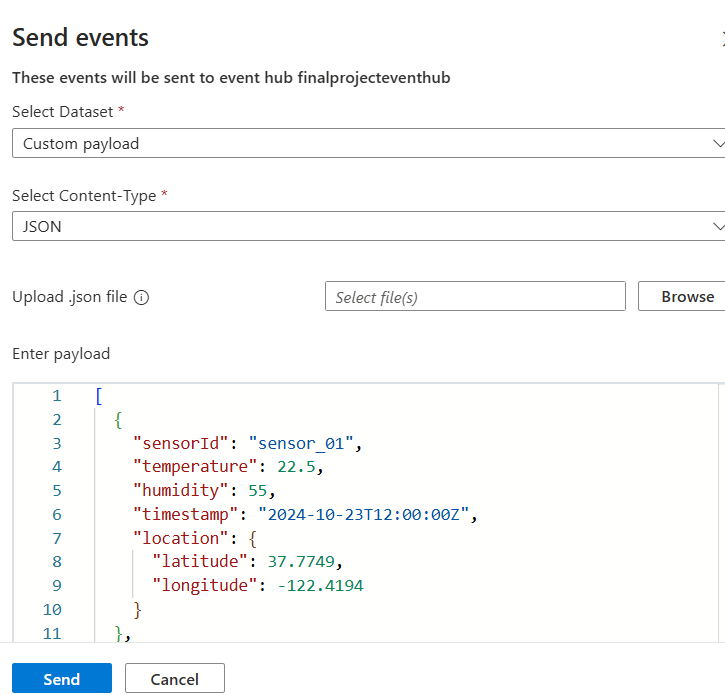
Input preview for reference:



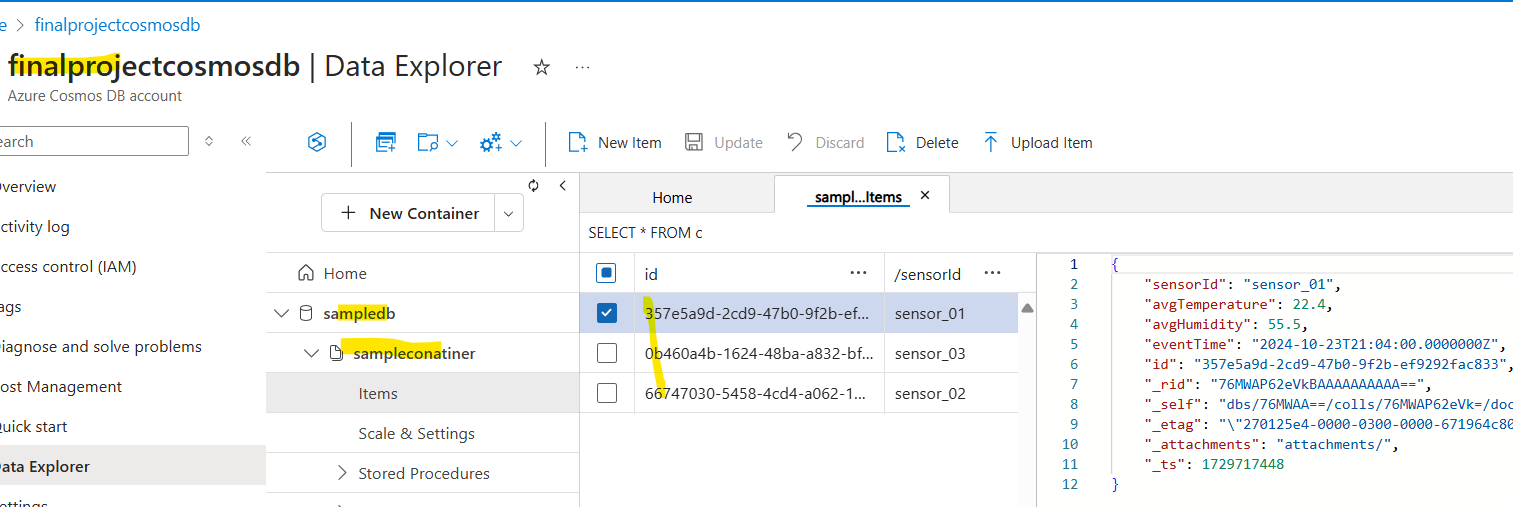
* + After testing, start the job



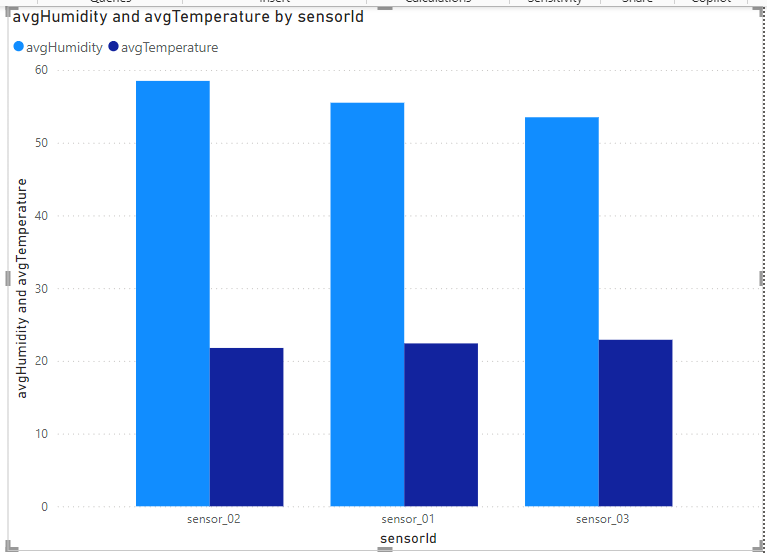
* + Send event (data from sensors) to event hub:



* + Data was processed by stream analytics and loaded into the cosmos DB container: partitioned by sensorId



* + Visualize the data in cosmos DB container using Power BI:



**Conclusion:** Processed real-time streaming data from 3 sensors and detected the average temperature and humidity of the room by using event hub and stream analytics. Visualized this data using Power BI.

### **Scenario 2: Batch Processing of Large Structured Datasets**

**Use Case:** A financial services company needs to analyze their sales data in different regions in the month January to understand the sales trend.

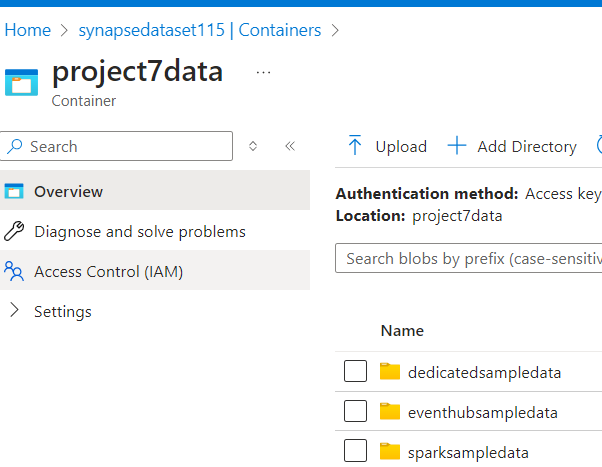
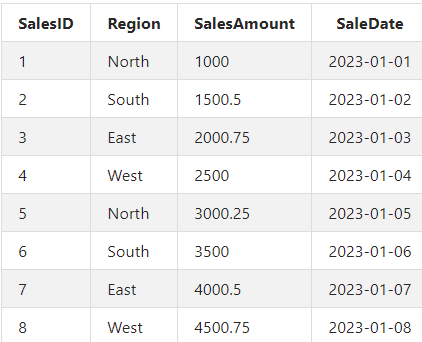
#### **Solution/Tools:**

|  |  |  |
| --- | --- | --- |
| **Tool used:** | **Purpose:** | **Why we choose this tool in this solution:** |
| **Azure Data Lake Storage (ADLS)**: | Store raw sales transaction data in CSV format. | * Scalable * Cost-effective * Handles large volumes of data |
| **Azure Synapse Dedicated SQL Pool**: | Perform batch processing and large-scale querying on the structured data. | * Improves query performance * Can handle complex queries across large datasets * Massively Parallel Processing (MPP) - Distributes data and processing across multiple nodes (60) * Scalability * Provide features like SCD tables, Distribution methods etc.. |
| **Azure Synapse Serverless SQL Pool**: | Query raw data directly from ADLS without ETL. | * Cost effective * Scalability * Serverless * Enable querying of large volumes of data directly from Data Lake |

#### **Steps to follow:**

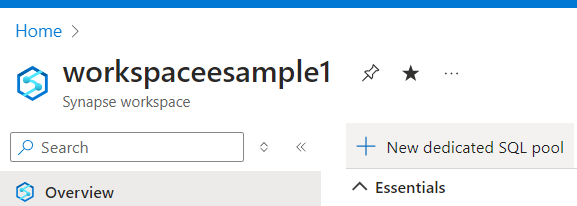
1. **Configure an ADLS account and upload the sales data file:**

* ADLS: synapsedataset115, container: project7data
* Location: dedicatedsampledata/data/sales\_data.csv

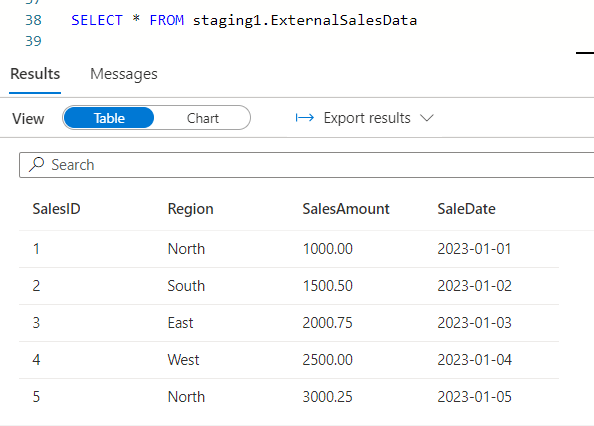
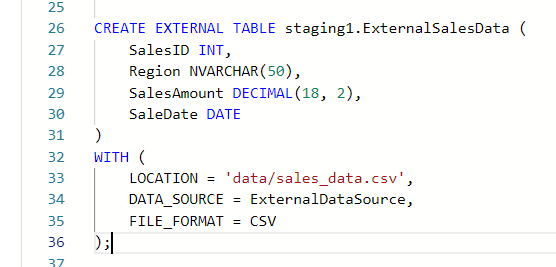
 

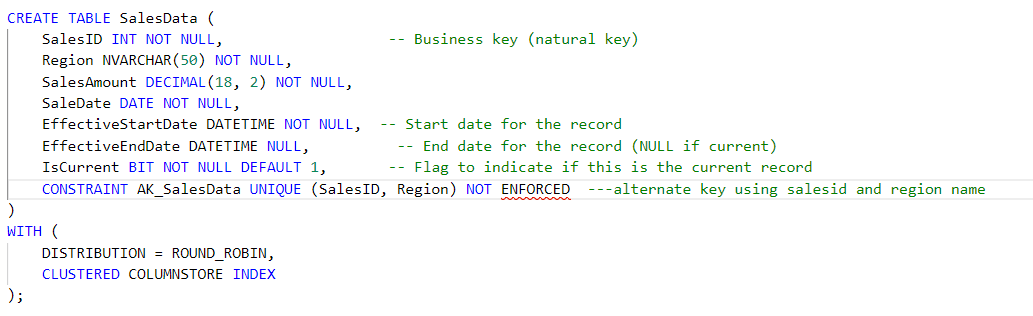
1. **Set up a Dedicated SQL Pool in the synapse workspace:**

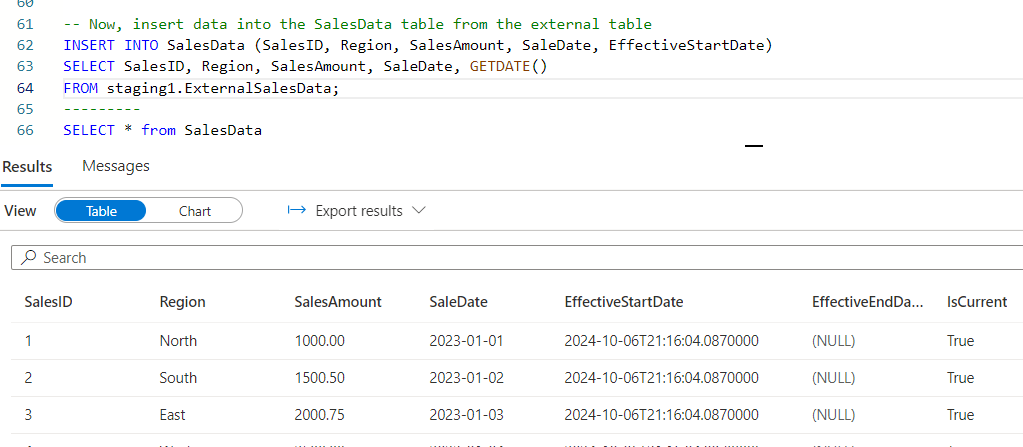
* Dedicated pool: samplededicatedsqlpool
* Workspace: workspacesample1



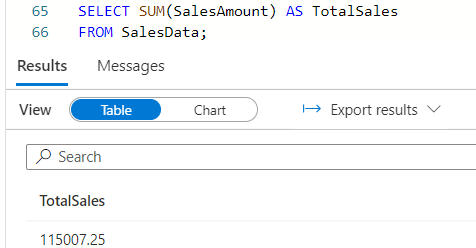
* Create an external table ‘ExternalSalesData’ to read sales data from ADLS:

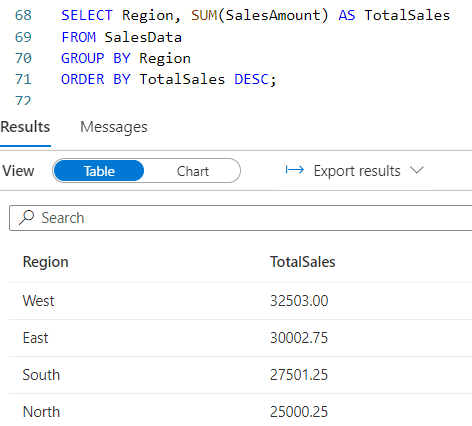


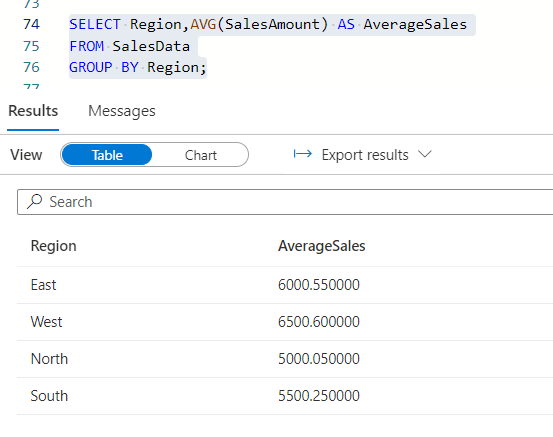
* Create ‘SalesData’ table with following features: Type 2 SCD, alternate key
* Insert data to SalesData table from the external table:



* Perform some transformations on sales data:

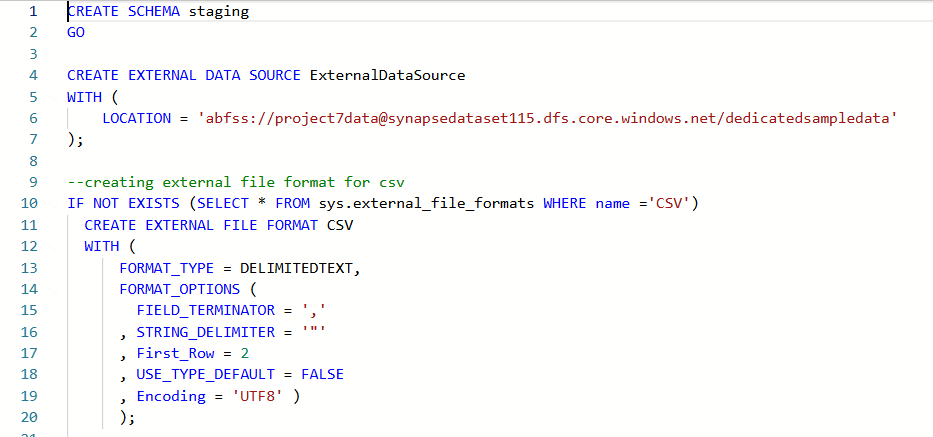
 TOTAL SALES

 REGION WISE TOTAL SALES

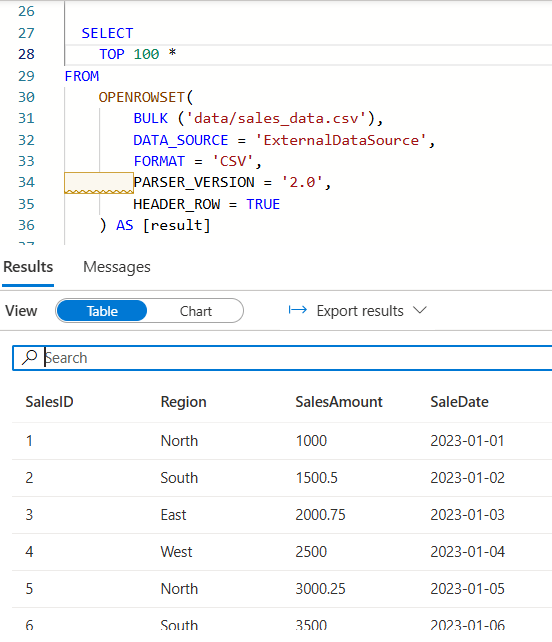
 REGION WISE AVG SALES

1. **Use Serverless SQL Pool to query data directly from ADLS without loading it into Synapse:**

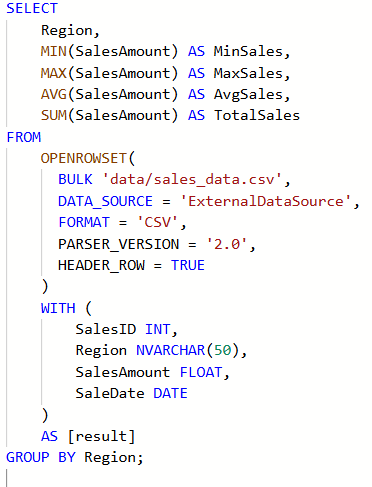
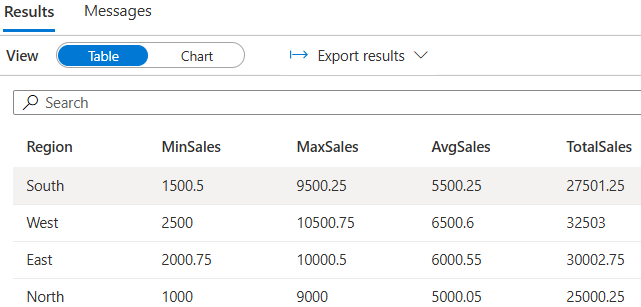
* Use the Built- in serverless pool
* Workspace: workspacesample1
* Create external data source and file format:



* Read the sales data in ADLS folder without loading it to the synapse using select statement:

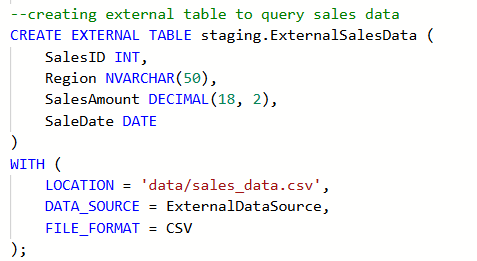


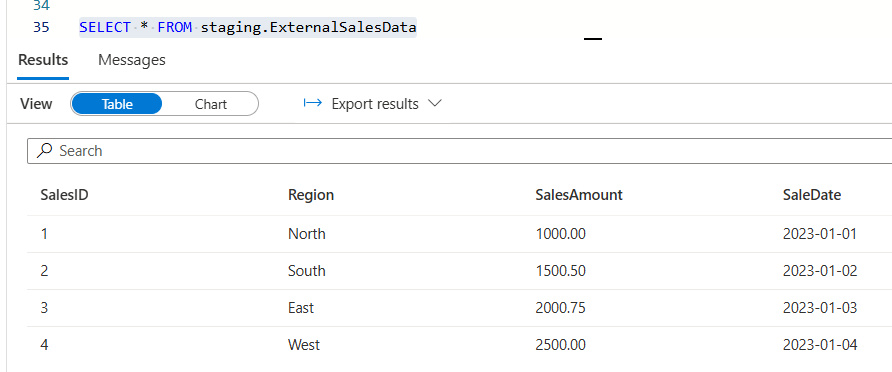
* Now, if we need to know the min, max, avg and total sales in each region without loading data to synapse, use the following code:

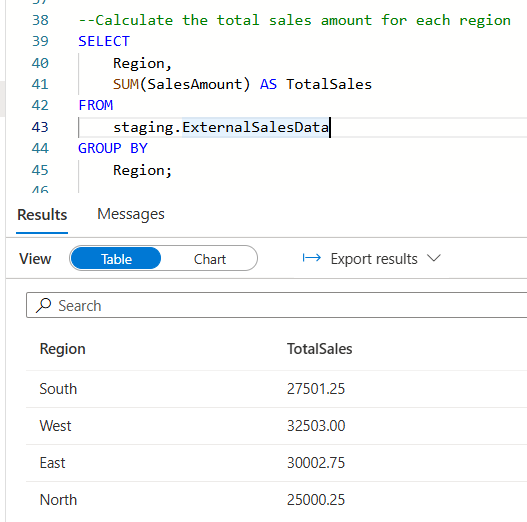
* Create an external table ‘ExternalSalesData’ to query the sales data stored in adls:

(This external table can be used to integrate with power BI to build data visualizations)

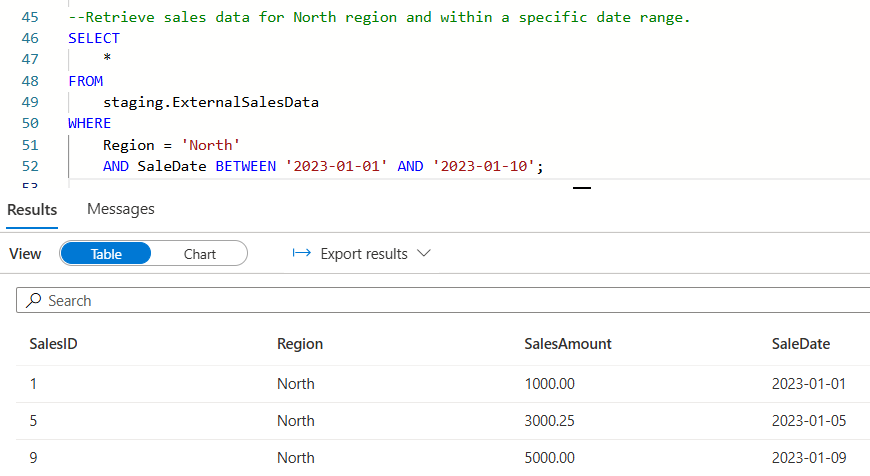


****

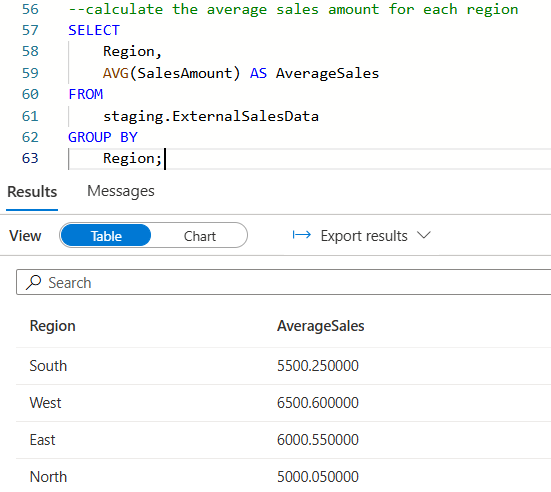
* Perform some transformations on sales data:
* Total sales by region:

****

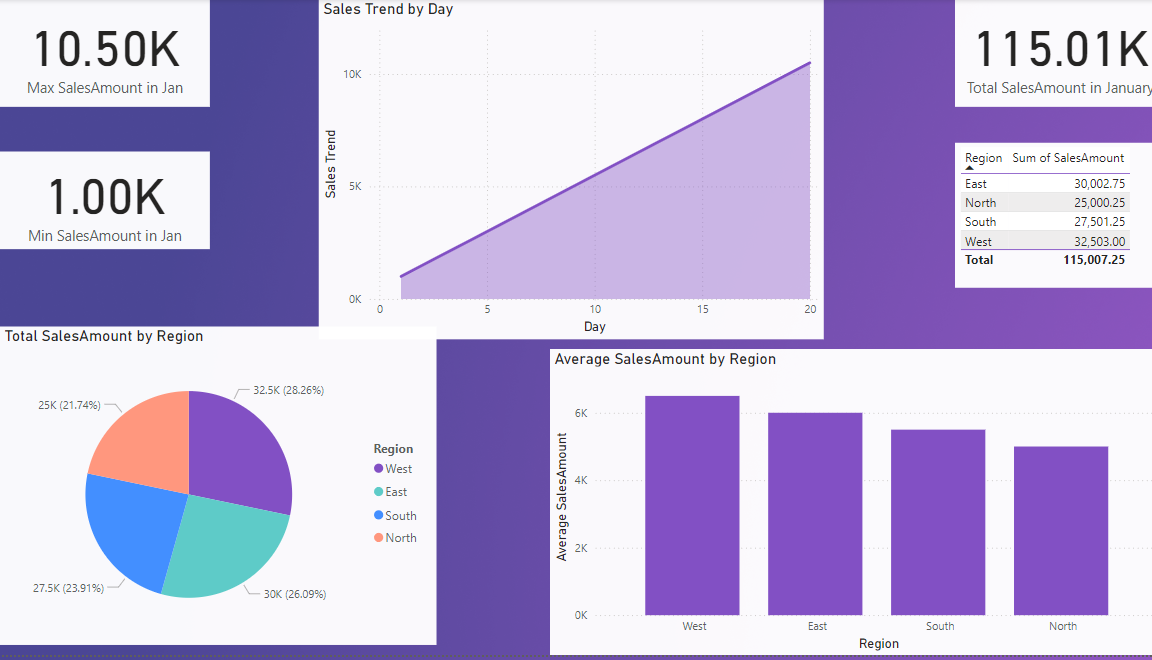
* Retrieve sales data for North region within a specific data range:

****

* Average sales amount by region:

****

* Visualizing the Sales trend of the company in the month January using Power BI:



**Conclusion:** Analyzed the region wise total and average sales of the company in the month January using dedicated and serverless SQL pools and visualized using Power BI.

### **Scenario 3: Regional and Monthly Sales performance Analysis:**

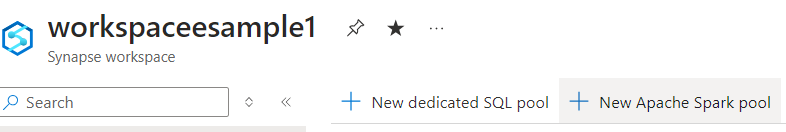
**Use Case:** An e-commerce business sells products online and wants to understand its monthly sales patterns and evaluate the sales performance across regions using a high performance and multiple language compatible azure service.

#### **Solution/Tools used:**

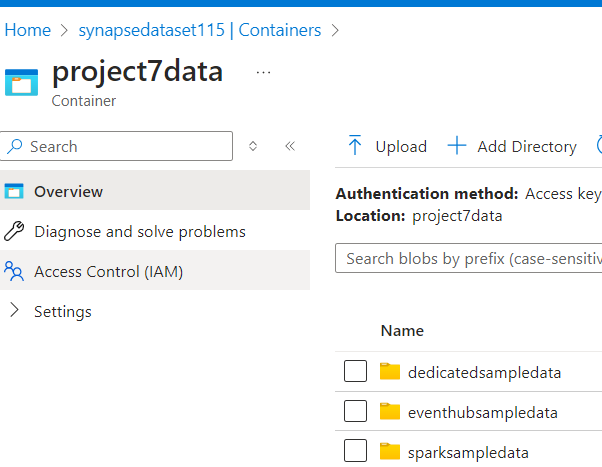
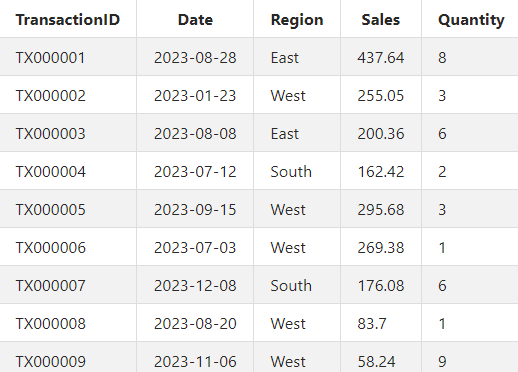
|  |  |  |
| --- | --- | --- |
| **Tool used:** | **Purpose:** | **Why we choose this tool in this solution:** |
| **Azure Synapse Spark Pool**: | Perform transformations on datasets to analyze sales trend over the year. | * Process large amounts of data efficiently (real-time or batch mode) * Distributing computing power * Compatible with multiple programming languages |
| **Azure Data Lake Storage (ADLS):** | Store raw and processed Sales transactions data | * Scalable * Cost-effective * Handles large volumes of data |
| **Azure Synapse Pipelines:** | ToIntegrate the notebook with pipeline as an activity. Pipelines can be used to automate the process in future by scheduling it | * Data integration * We can automate pipelines and reduce manual intervention * Cost-effective * Scalable |

**Steps to follow:**

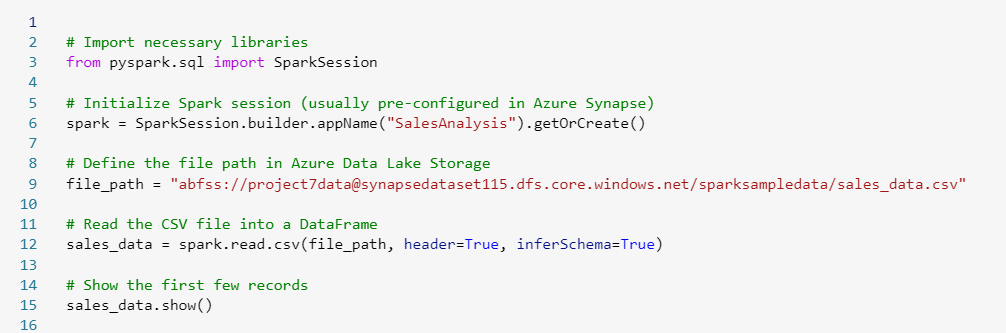
1. Set up the Spark Pool ‘samplesparkpool’ in synapse workspace:

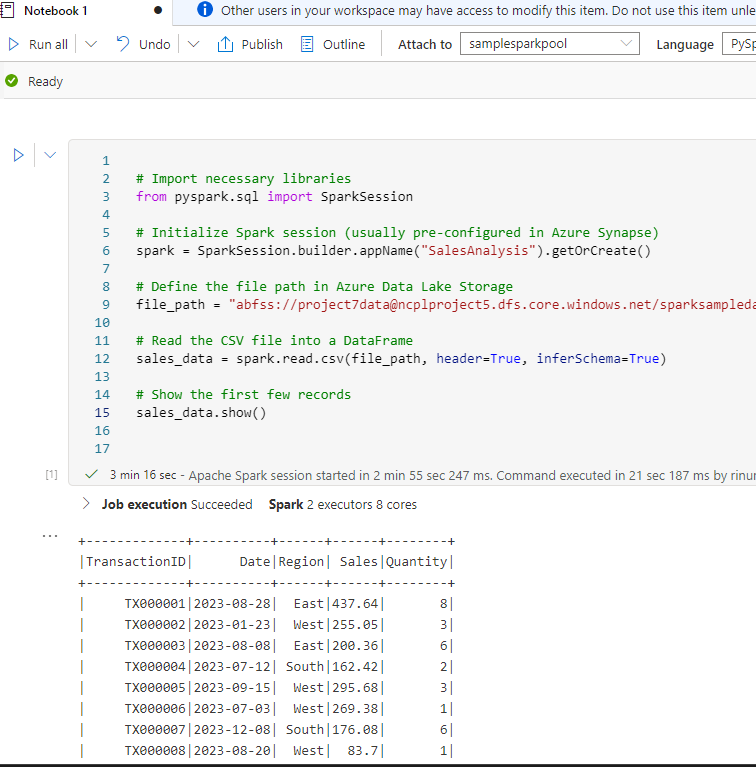


1. Upload a Sales data file in ADLS: sparksampledata/sales\_data.csv

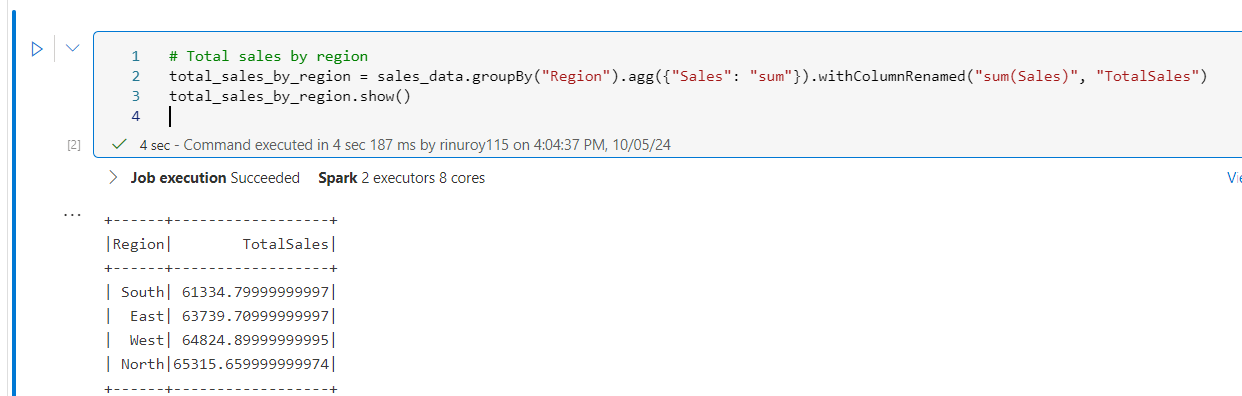
1. Write Python scripts to read the sales data into data frame(sales\_data) by creating a notebook in spark pool:



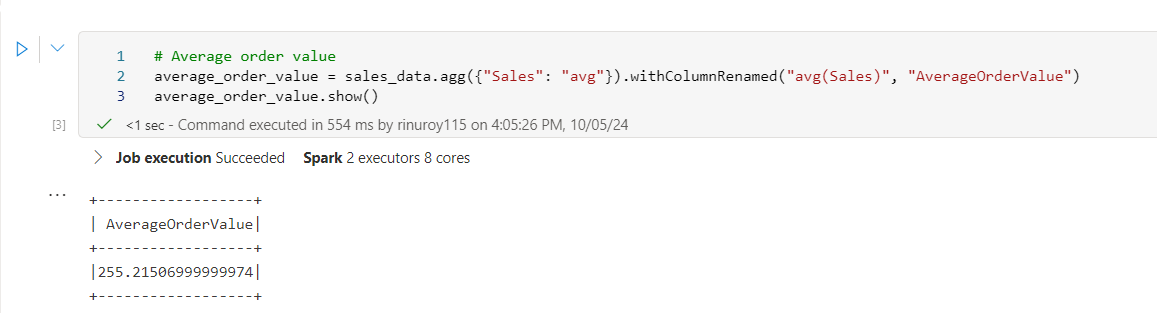


1. Performing some transformations on data frame(sales\_data) in spark pool:

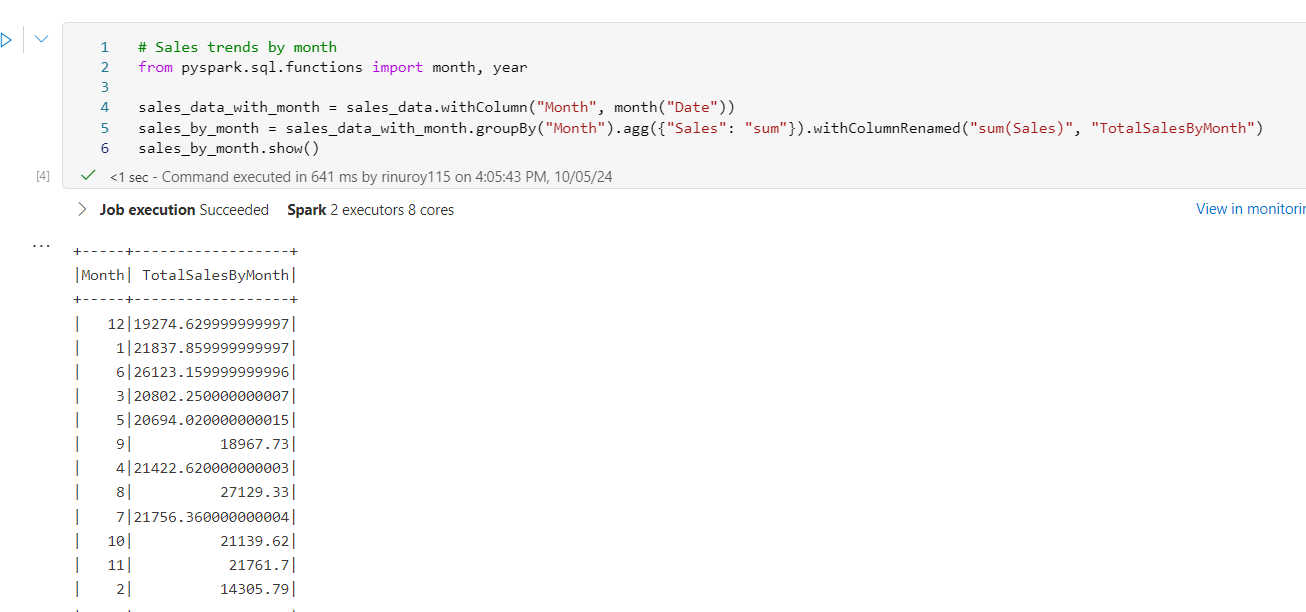
* Total Sales by Region:



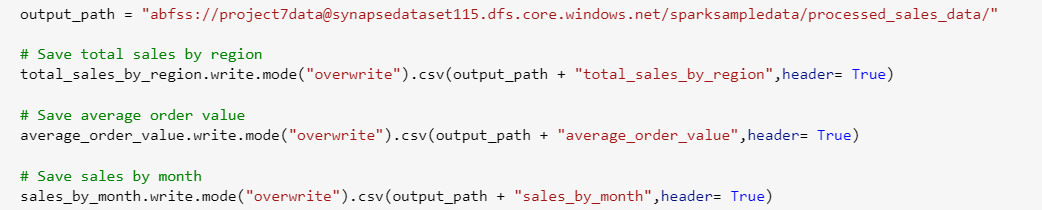
* Average Sales value:



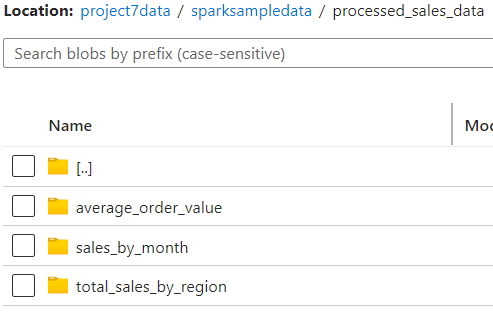
* Total Sales by Month:



1. Write/save the output of transformations to ADLS location mentioned:



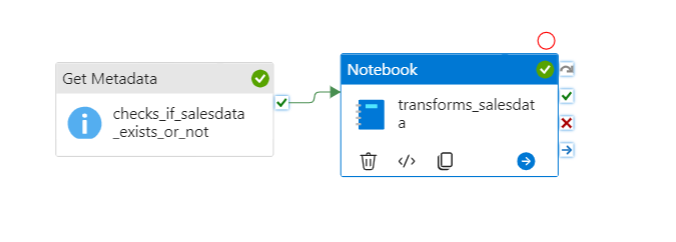




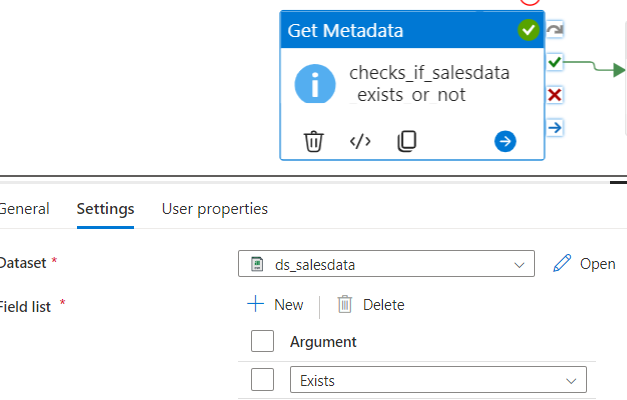
**Next create a pipeline named ‘Pipeline\_transformations’ under the Integrate tab:**

****

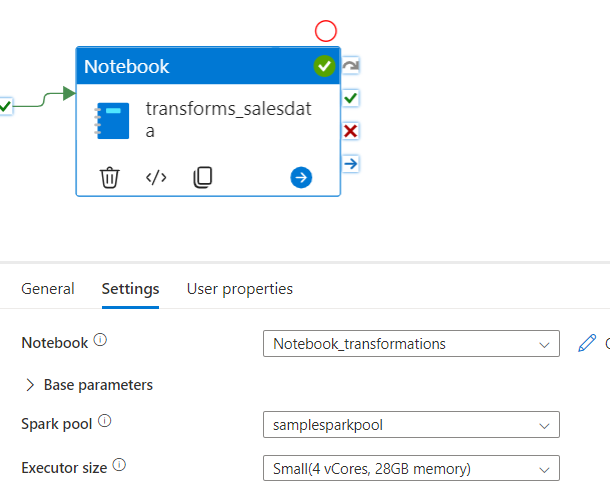
1. Integrate the notebook with pipeline as an activity: pipeline has 2 activities



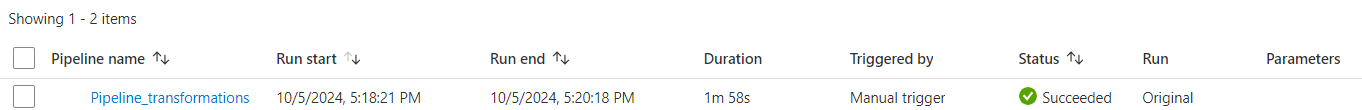
* Get metadata activity: to check if file exists or not in ADLS location



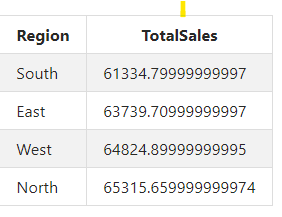
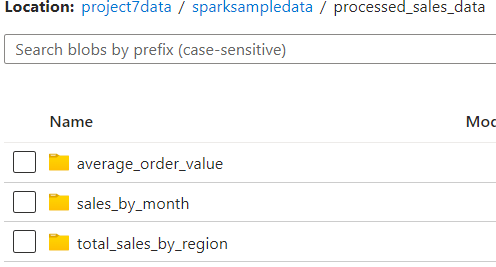
* Notebook activity: to perform transformations on the data using spark pool



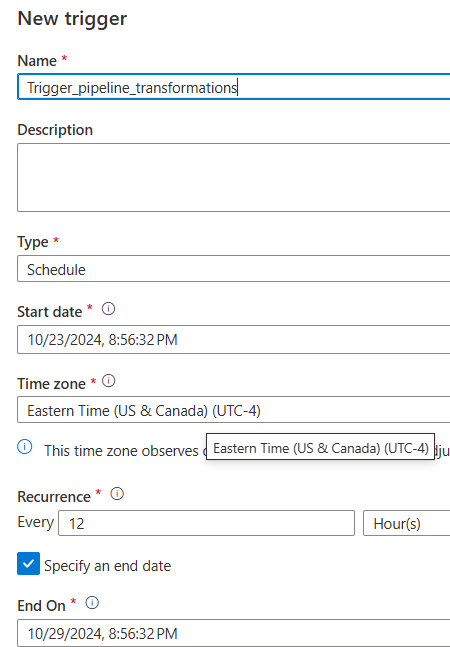
* Triggered the pipeline and it got completed successfully:



* Output: data got saved to mentioned ADLS location



* We can add a schedule trigger to automate/schedule the pipeline run as follow:



**Conclusion:** Analyzed monthly sales patterns and evaluated the sales performance across regions using spark pool and integrated the notebook with a pipeline.

**ERRORS FACED:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Issue** | **Supporting Images** | **Solution Found** |
| 1 | Could not load data from cosmos DB containers to Power BI |  | Transformed the data types using Power BI and loaded the data and it worked. |
| 2 | Mentioned Incorrect container name |  | Corrected the container name and it worked |
| 3 | Got an error while adding alternate key to SalesData table: |  | Added the syntax ‘NOT ENFORCED’ and it worked. |
| 4 | Forgot to mention Header = True in a statement and data was saved without column names in ADLS |  | Mentioned Header = True in the statement and it worked |
|  |  |  |  |

**Insights Gained:**

* If using Azure IoT Hub as an intermediary, sensors can communicate with IoT Hub using various protocols (HTTP, MQTT, AMQP), and then IoT Hub can route messages to Event Hub.
* Azure provides Software Development Kits (SDKs) for various programming languages, which make it easier to connect IoT devices to Azure Event Hub.
* The Internet of Things (IoT) refers to a network of physical devices that are connected to the internet, allowing them to collect and exchange data.
* Choose Cosmos DB if you need real-time access, scalability, and flexibility in data structure.
* Choose Dedicated SQL Pool if your focus is on large-scale analytics with complex querying over structured data.