Ingestion and Transformation of Olympics Data

Title:

This project is designed to showcase how various Azure services can be utilized to ingest, store, transform Olympics-related data. We have employed the following Azure services to achieve these goals:

- **Data Ingestion: Configure and run the ADF pipelines to ingest the Olympics data into Azure Data Lake Storage Gen2.**

- **Data Transformation: Execute the PySpark notebooks in Azure Databricks to perform data transformation tasks**

Project Overview

The main objective of this project is to ingest Olympic data into Azure Data Lake Storage Gen2 using Azure Data Factory (ADF) pipelines and perform data transformation tasks using PySpark notebooks in Azure Databricks. The processed data will be stored back in Azure Data Lake Storage Gen2 for further analysis.

Workflow:

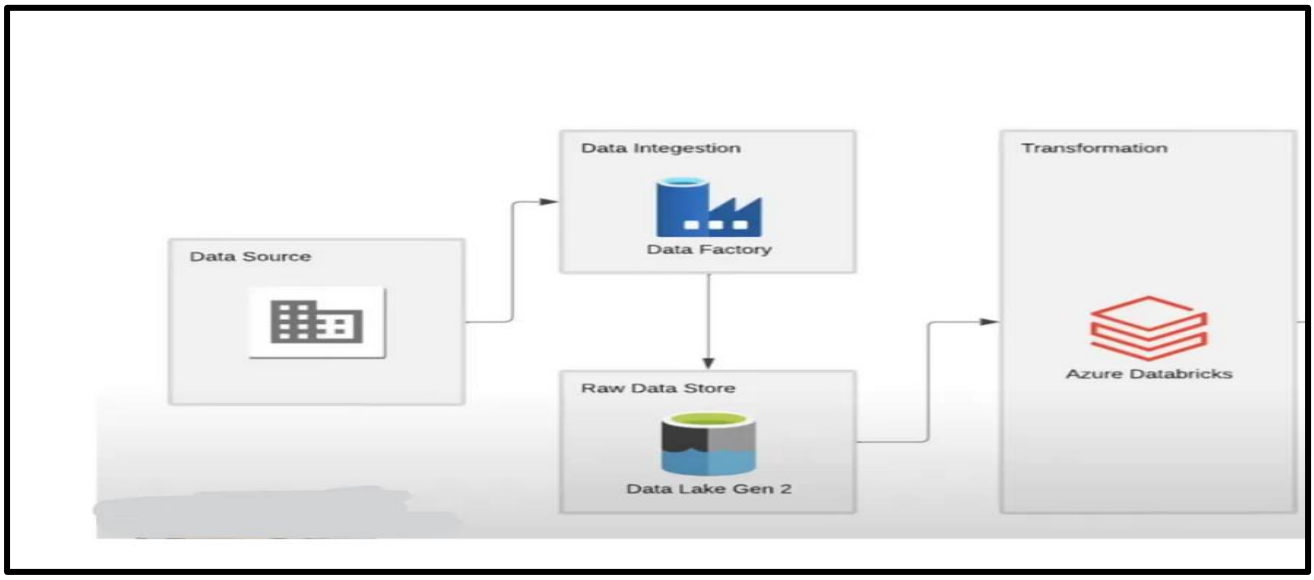
1. **Data Ingestion:**
    - Raw Olympic data from various sources is ingested into Azure Data Lake Storage Gen2 using ADF pipelines.
    - ADF is configured to handle data movement and transformation activities, ensuring data integrity and security during the ingestion process.
2. **Data Transformation:**
    - Azure Databricks is employed to execute PySpark notebooks that perform data transformations on the ingested Olympic data.
    - PySpark notebooks read data from Azure Data Lake Storage Gen2, apply necessary transformations, and store the processed data back in the data lake.

Azure Resources/Tools Used:

- **Azure Data Factory (ADF)**: Used for data ingestion, Azure Data Factory allows us to efficiently collect data from various sources and move it to the desired destination. In this project, ADF is responsible for bringing in Olympics data from external sources.
- **Azure Data Lake Storage Gen2**: This is where the ingested data is stored. Azure Data Lake Storage Gen2 provides a scalable and secure platform for storing large volumes of data. It enables us to manage, access, and analyse data effectively.
- **Azure Databricks**: For data transformation, we leverage Azure Databricks with PySpark. Databricks provides a collaborative environment for data engineers and data scientists to work together on big data projects. In our project, we use PySpark to clean, reshape, and process the raw Olympics data into a more usable format.
- **Python with PySpark:** PySpark is the Python API for Apache Spark, which is a fast and general-purpose cluster computing system. PySpark would be used within Azure Databricks notebooks for developing and executing data transformation tasks.

- **Microsoft Excel:** Microsoft Excel might be used for data preparation or as a data source for creating input datasets for the project.

Architectural diagram



Execution Overview

To ingest Olympics data into Azure Data Lake Storage Gen2, Azure Data Factory (ADF) pipelines are configured and executed. These pipelines are designed to efficiently transfer data from various sources to the designated Azure Data Lake Storage Gen2 destination. The configuration includes defining the data sources, mapping transformations, and specifying the target location within the Data Lake.

Once the data is ingested, the next step involves executing PySpark notebooks in Azure Databricks to perform data transformation tasks. These PySpark notebooks contain the logic and code necessary to clean, process, and transform the raw Olympics data into a structured and analyzable format. The notebooks leverage the distributed computing capabilities of Apache Spark to handle large datasets efficiently.

This end-to-end process ensures a seamless flow from data ingestion using ADF pipelines to data transformation using Azure Databricks, ultimately preparing the Olympics data for further analysis and in sights within the Azure ecosystem. The combination of these Azure services facilitates a scalable and flexible approach to managing and processing large volumes of data for analytics and reporting purposes.

How it works

Data Ingestion:

1) Source Data Identification:

- Identify the sources of Olympics data.
- These sources could include databases, Azure storage accounts, or any other data repositories where the Olympics data is stored.

2) **Azure Data Lake Storage Gen2 Configuration:**

- Set up Azure Data Lake Storage Gen2 (ADLS Gen2) as the destination for storing the Olympics data.
- Go to Azure Portal and search for Storage Accounts.
- Click on + to create. Add the Subscription , Resource group name , storage account name
- Enable Hierarchical Namespace to organize data in a hierarchical structure, improving performance and manageability in Advanced setting
- Click on Create
- Deployment is in progress.
- Once the deployment is complete Click on Go to Resource.

3) Azure Data Factory (ADF) Pipeline Configuration:

- Create a Azure data factory by selecting the Azure subscription, Resource group and give a name for data factory and Click on Create.
- Once the deployment is complete, Click on Go to Resource
- Click on Launch Studio.
- Click on Ingest Data
- Give the details of Source and Destination to perform Copy Activity.

4) **Run the Pipeline:**

- Once the pipeline is configured, trigger the pipeline execution manually or let it run based on the defined schedule.
- Monitor the pipeline execution for any errors or issues and troubleshoot as needed.
- Verify that the data has been successfully ingested into ADLS Gen2.

Data Transformation:

1) Azure Databricks Environment Setup:

- Set up an Azure Databricks workspace in the Azure portal.
- Configure clusters within the Databricks workspace with appropriate specifications for running PySpark notebooks.

- Specify the cluster type, instance types, number of instances, and auto-scaling settings based on the workload requirements.
- It is a best practice to use personal compute to reduce the costing.

2) PySpark Notebook Development:

- Create a new notebook within the Azure Databricks workspace.
- Set the configuration to Azure Data Lake Storage gen2 account by providing name of the storage account,name of container and key.
- Develop PySpark notebooks to perform various data transformation tasks on the ingested Olympics data. Tasks may include aggregation, filtering, distinct operations, data normalization, joining datasets, etc.
- Write PySpark code to implement the desired data transformation logic.
- Ensure that the code is well-documented and includes comments for clarity and maintainability.
- Test the PySpark code within the notebook to verify that it produces the expected output.

3) Notebook Execution and Scheduling:

- Execute the developed PySpark notebooks within the Azure Databricks environment to perform data transformation tasks on the ingested Olympics data.
- Schedule the execution of notebooks at specified intervals or triggers to ensure regular updates to the transformed data.
- Configure notebook parameters and input/output paths as necessary for scheduled execution.
- Review the transformed data output to ensure it meets the desired transformation requirements.

Tasks performed

Data Ingestion with ADF Pipelines:

- Configure Azure Data Factory (ADF) pipelines to ingest Olympics data.
- Define the necessary data sources and sinks, specifying Azure Data Lake Storage Gen2 as the destination.
- Set up activities within the pipeline to orchestrate the data movement.
- Schedule or trigger the pipeline execution as per our requirements.

Data Transformation with PySpark Notebooks in Azure Databricks:

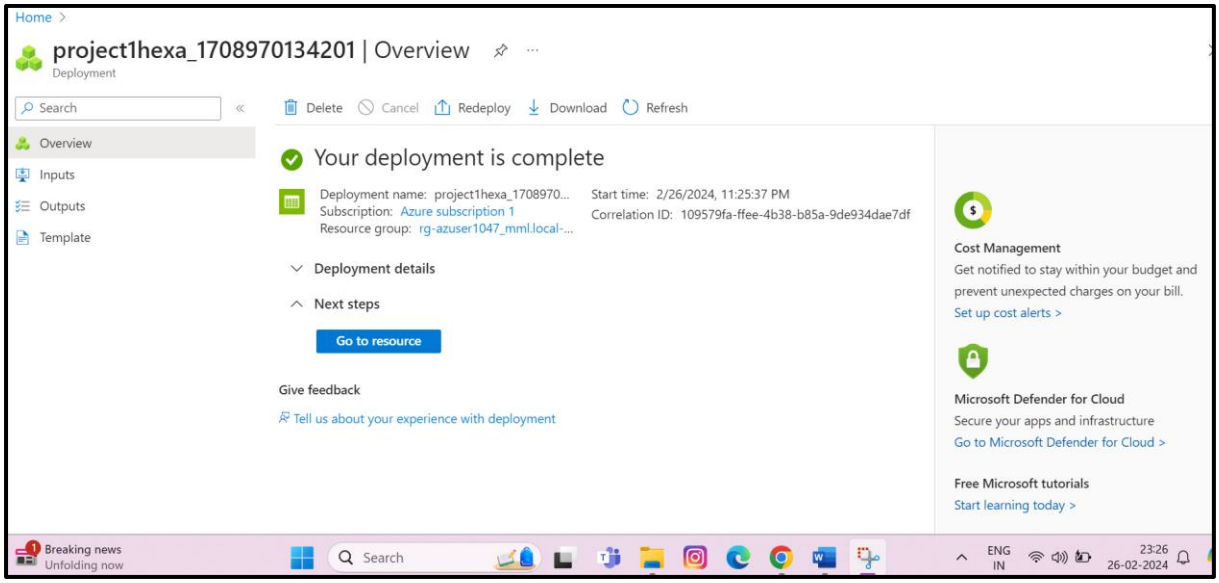
- Utilize Azure Databricks for data transformation using PySpark.
- Create Azure Databricks workspace in Azure portal.
- Develop PySpark notebooks to perform specific transformation tasks like sorting, filtering, aggregations, distinct columns etc., on the ingested Olympics data.
- Leverage the scalability of Databricks for efficient distributed processing.

- Execute the notebooks, ensuring they handle the data according to your transformation logic.

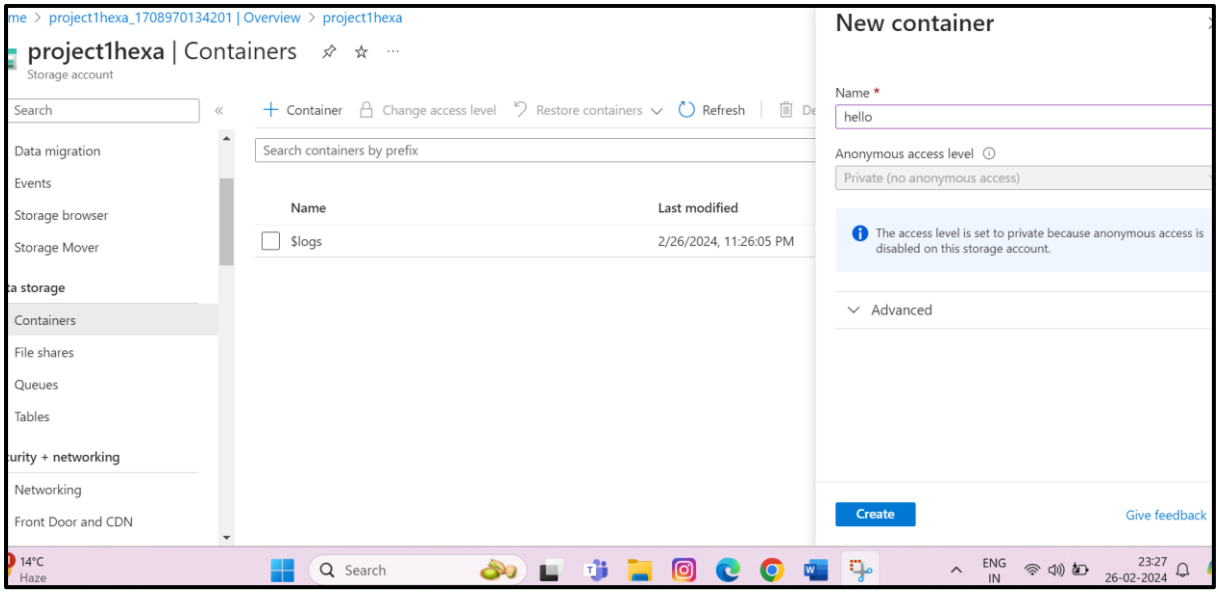
RESULTS

Setting Up Source and destination Storage Accounts

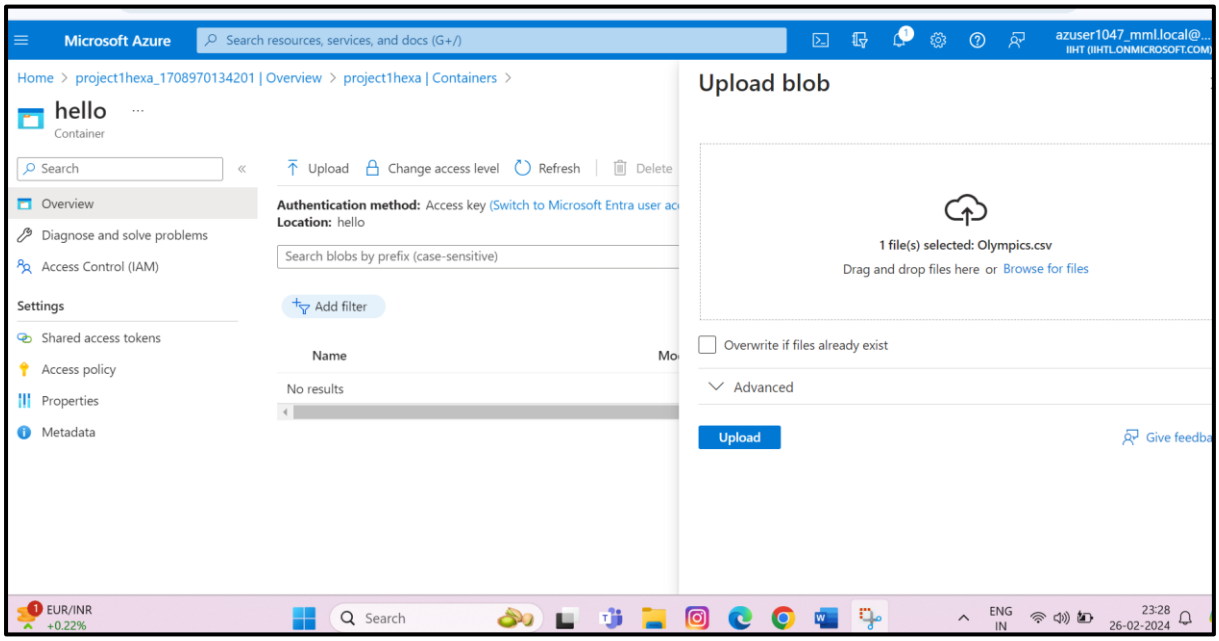
First a Blob storage is created:



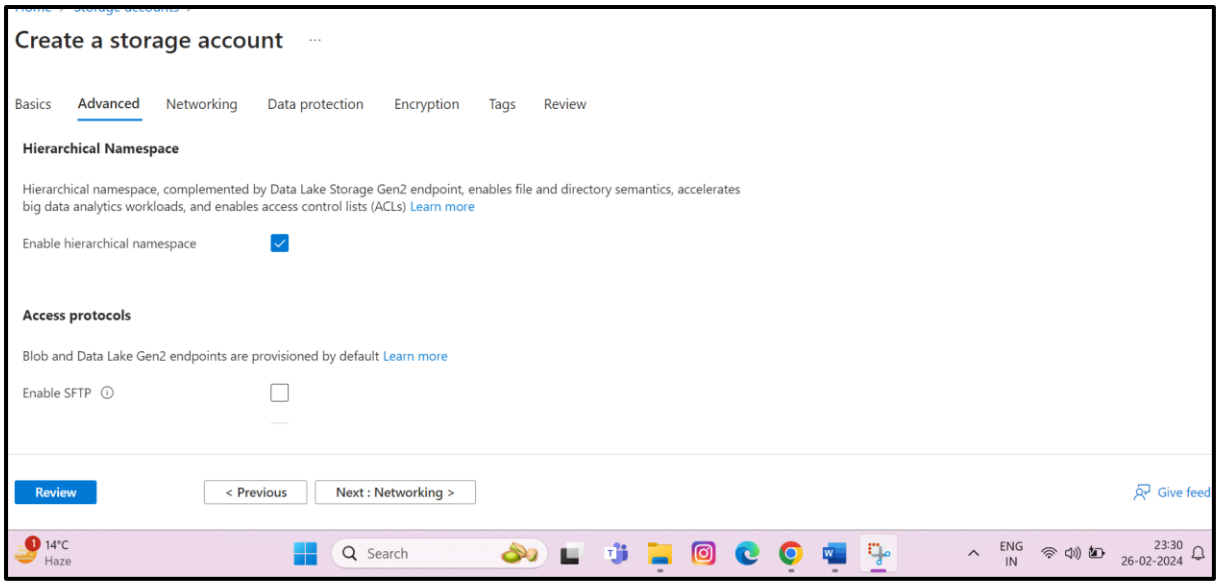
Inside the blob storage a Container is created:



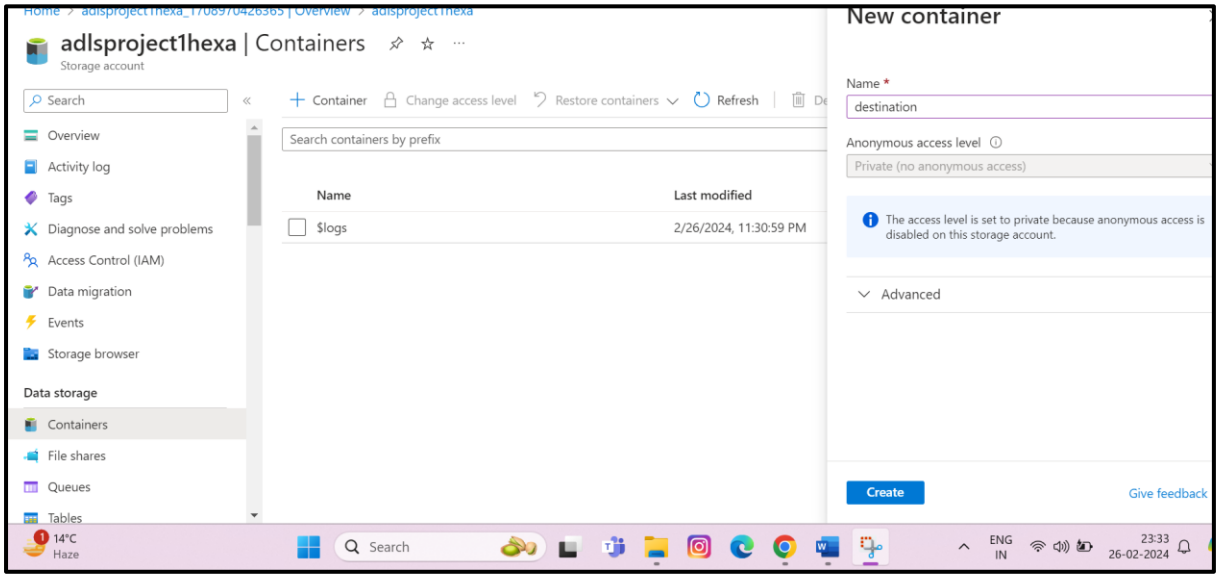
After that the source file is uploaded in the container:



Now creating an Aure Data Lake Storage GEN2(ADLS) account by enabling hierarchical namespace:

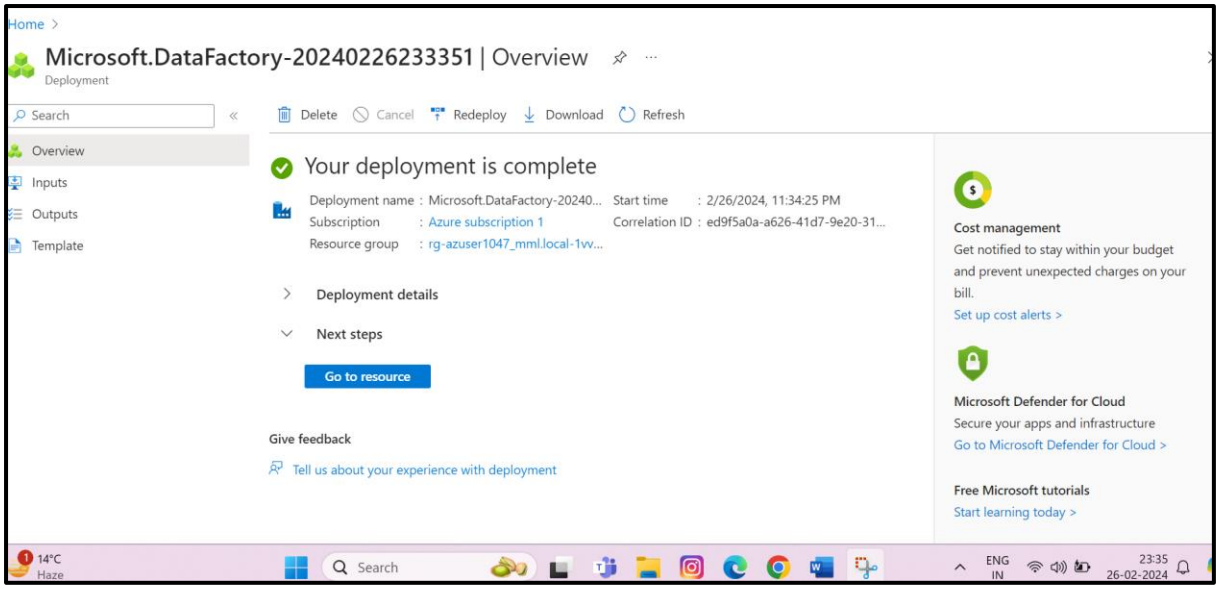


Creating a container in ADLS where file will be copied through copy action:

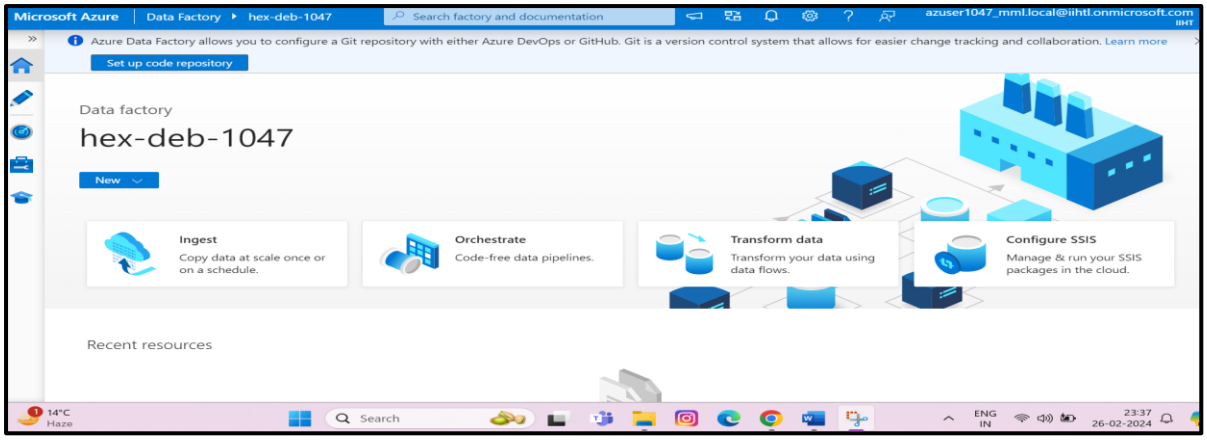


Data Ingestion

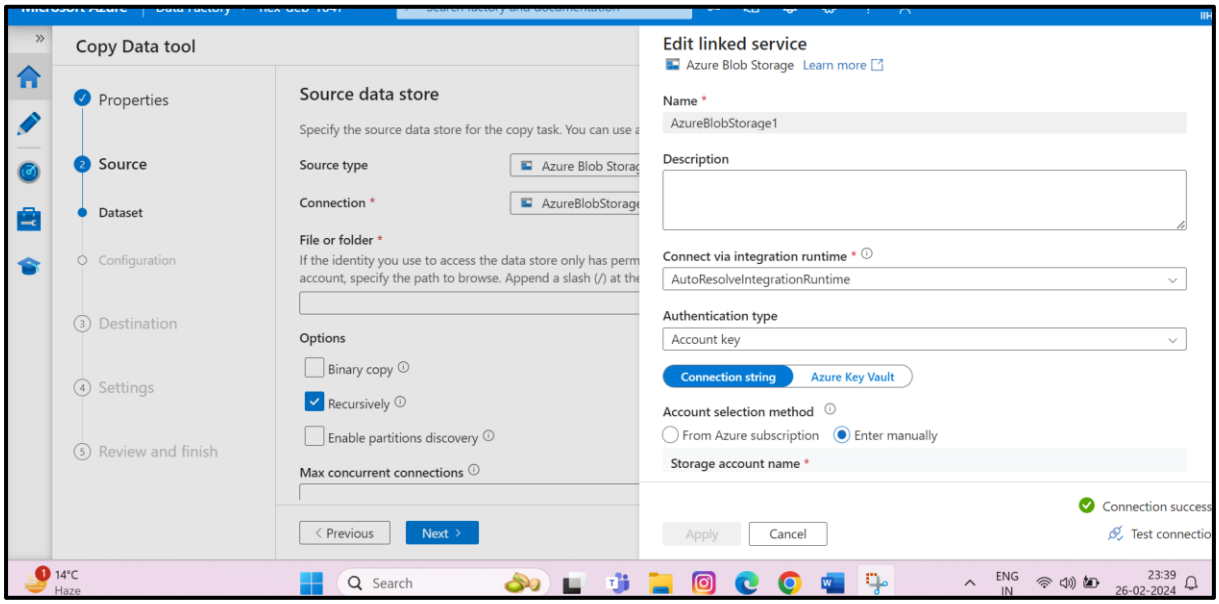
Data factory created successfully:



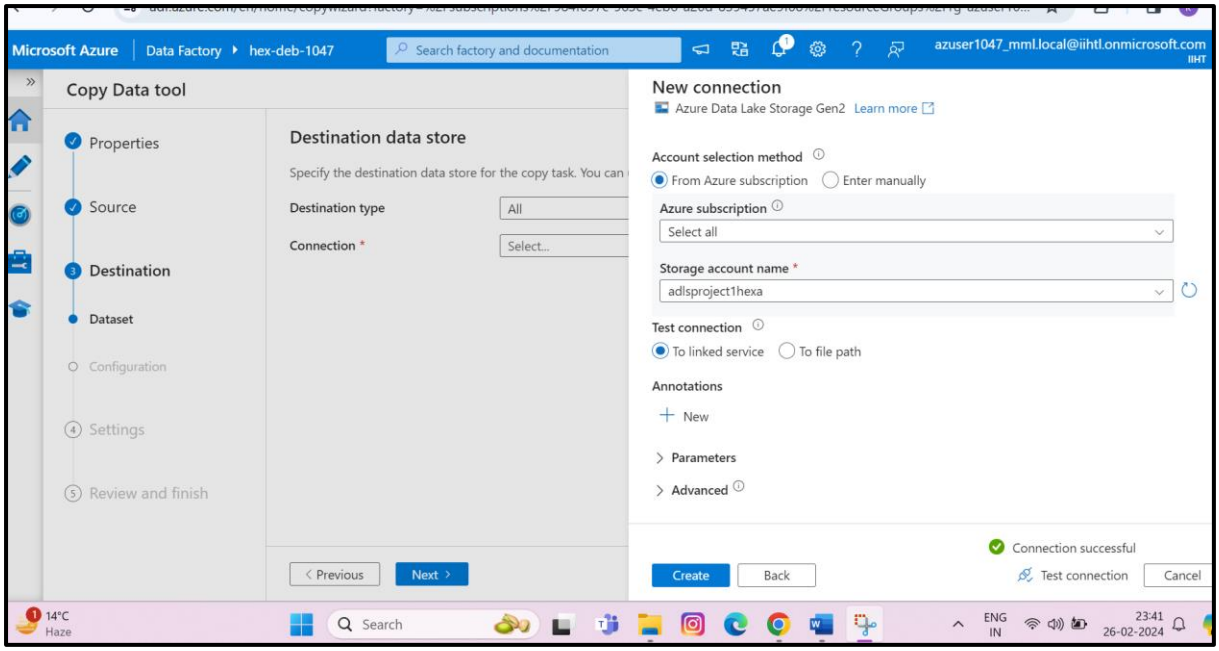
ADF portal:



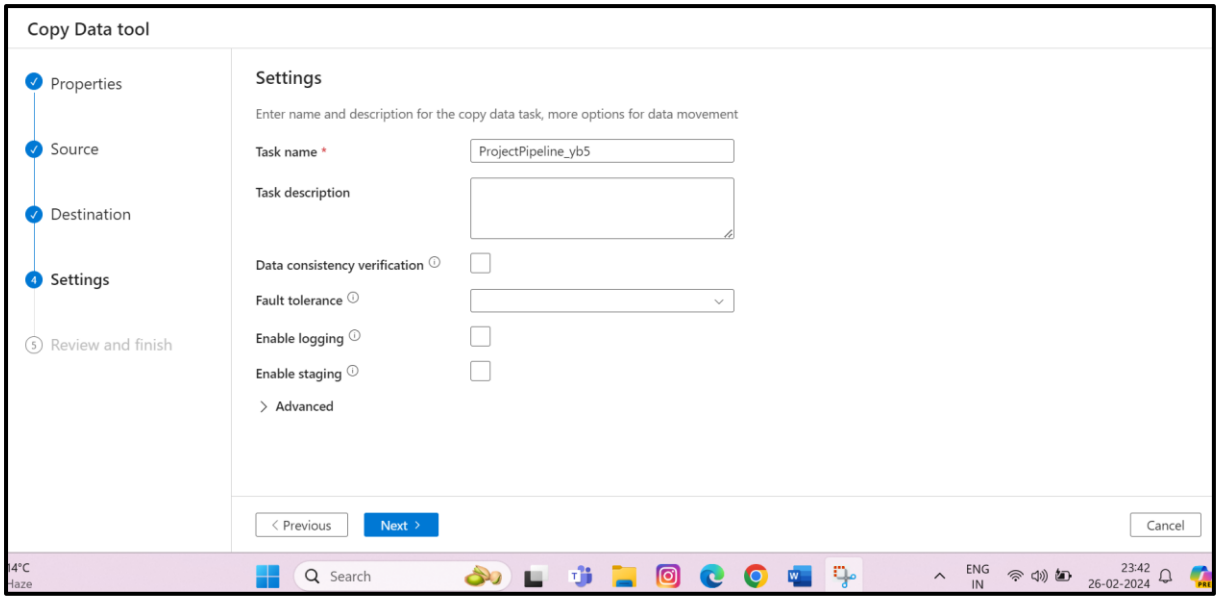
Connecting to source linked service i.e. blob storage:



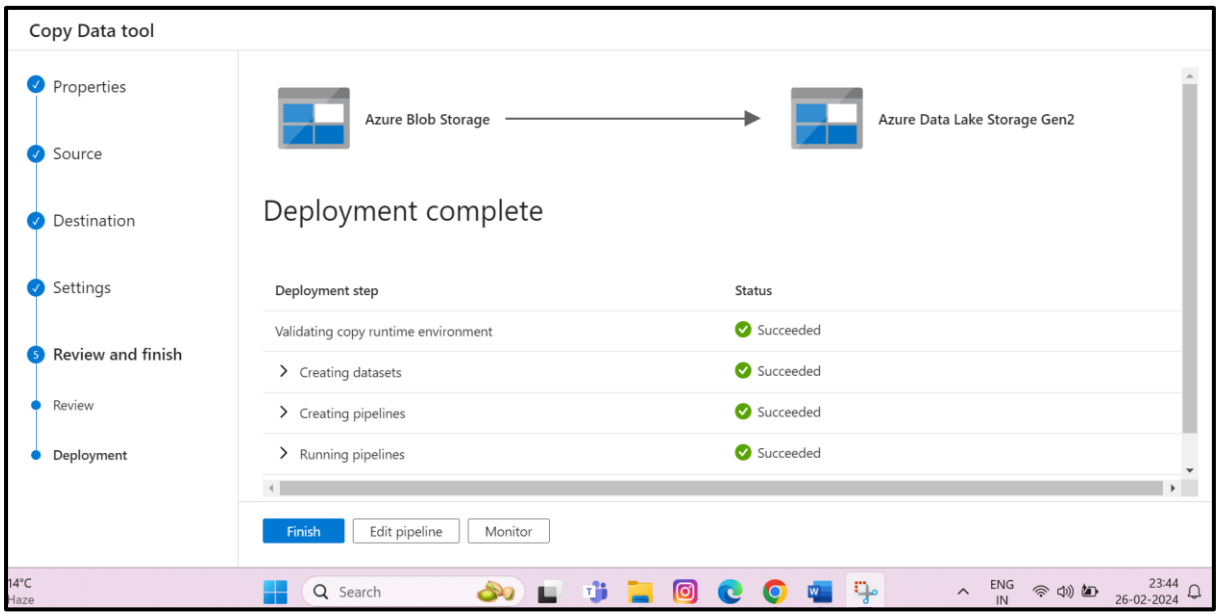
Adding destination storage account i.e. data lake store gen2:



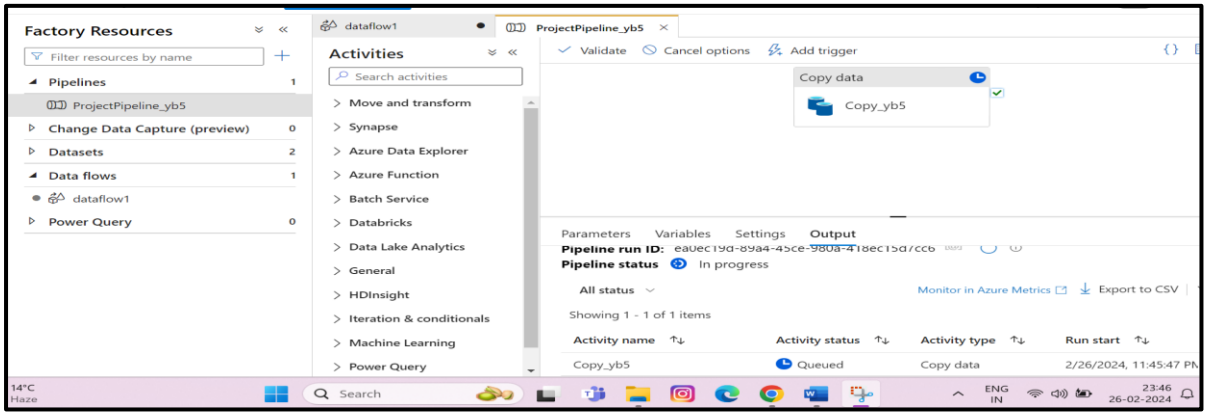
Pipeline Creation:



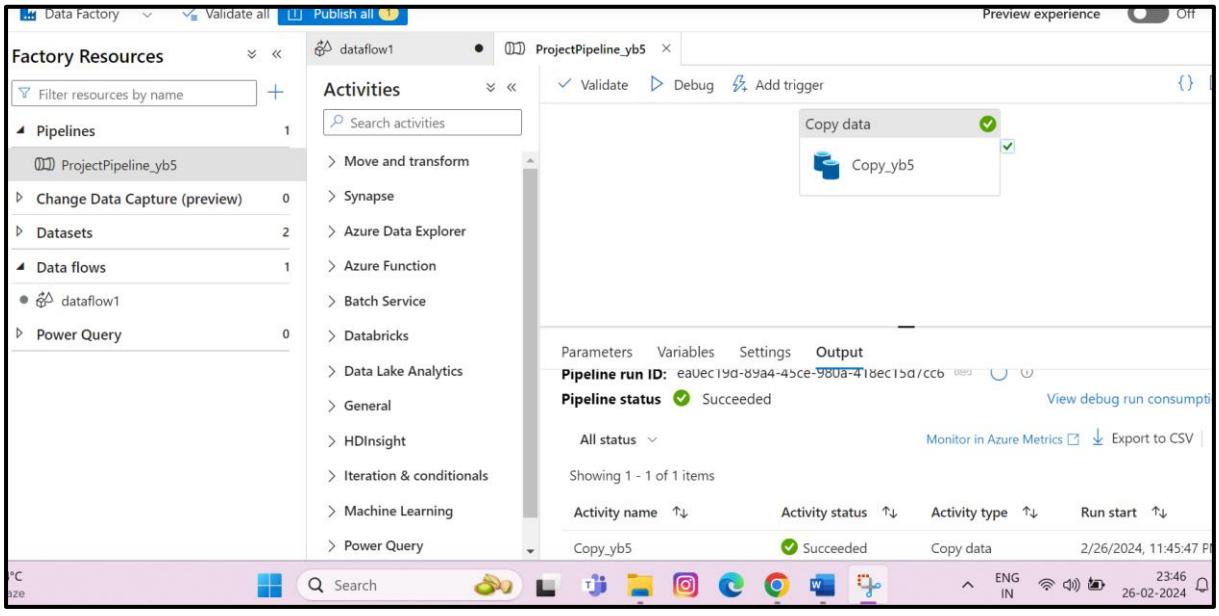
Deployment successful:



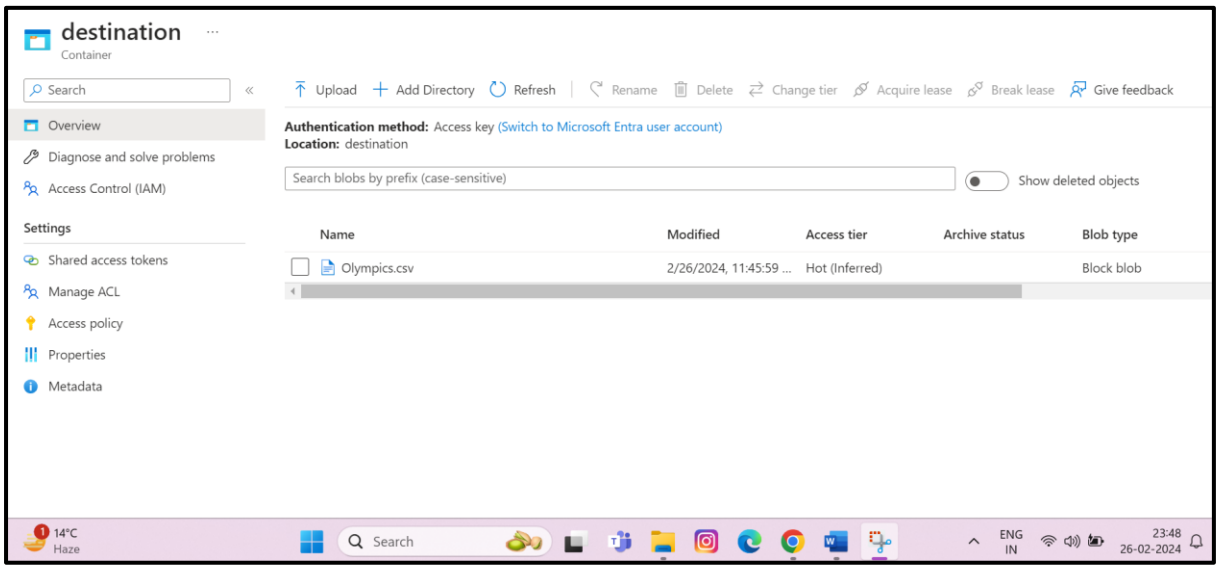
Debugging the pipeline by going to the author tab:



Debugging complete and copy action succeeded:

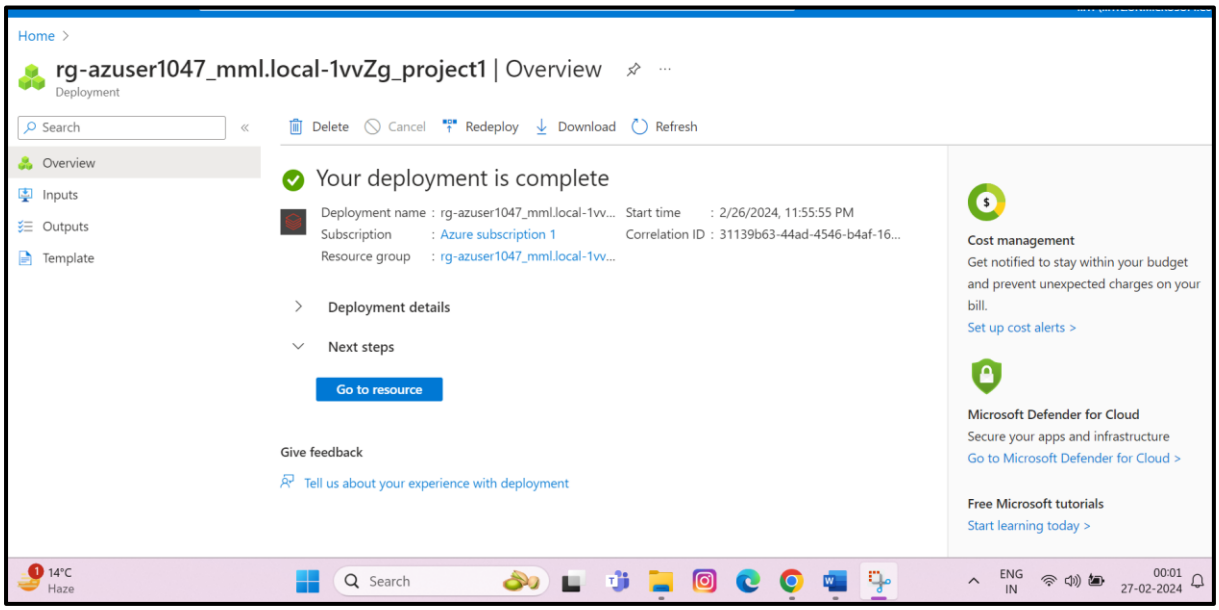


File is copied in the destination folder successfully:

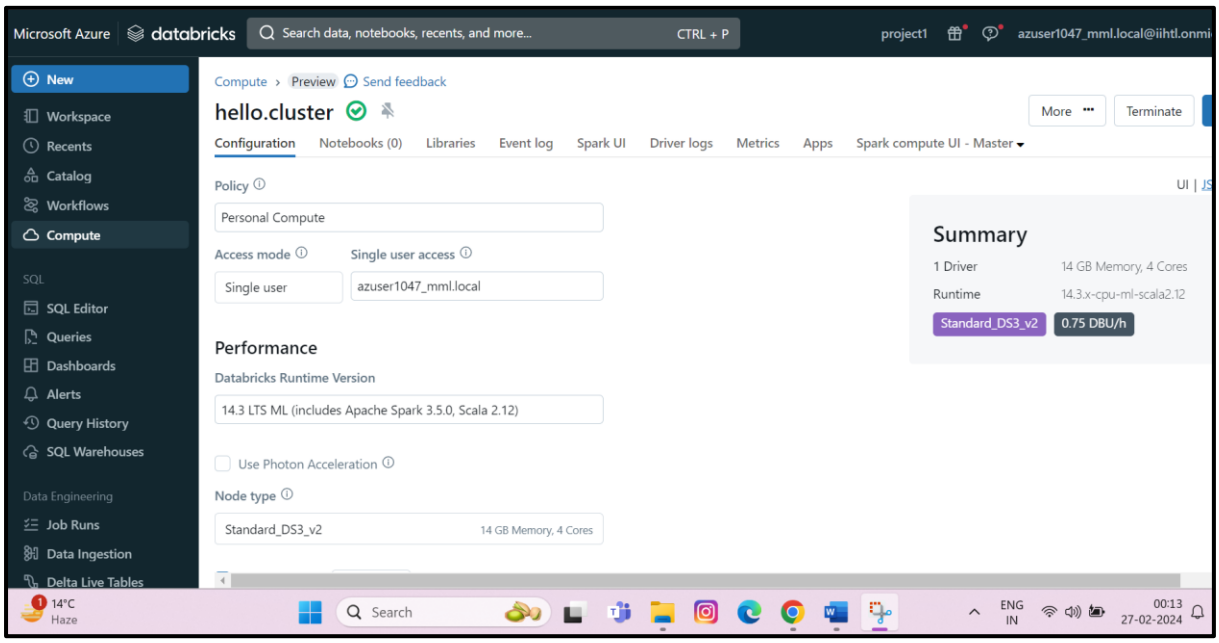


Data Transformations

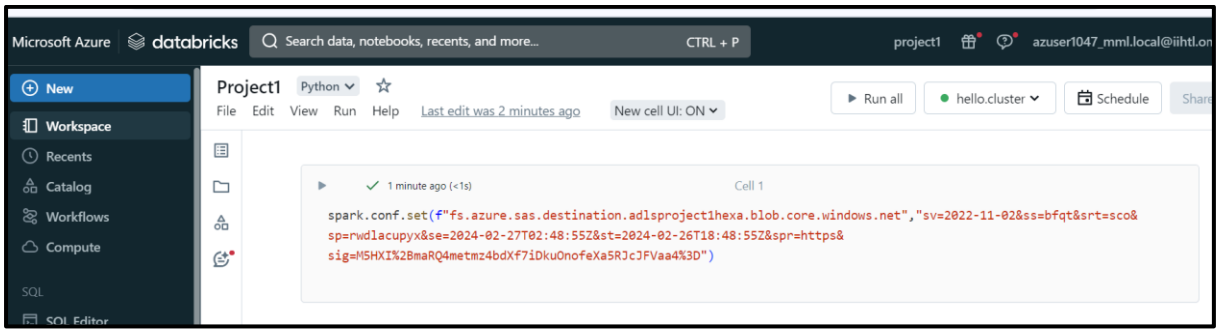
Creating a Databricks workspace:



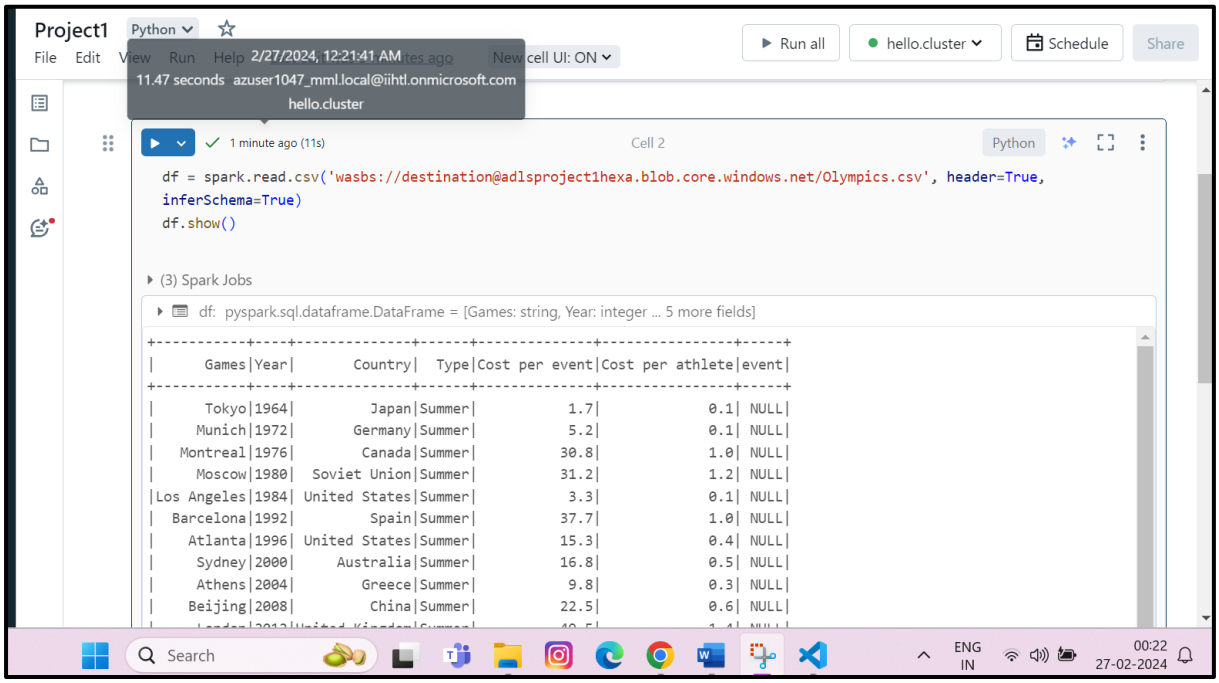
Created a cluster:



Connecting Databricks to source file through ADLS for transformations:

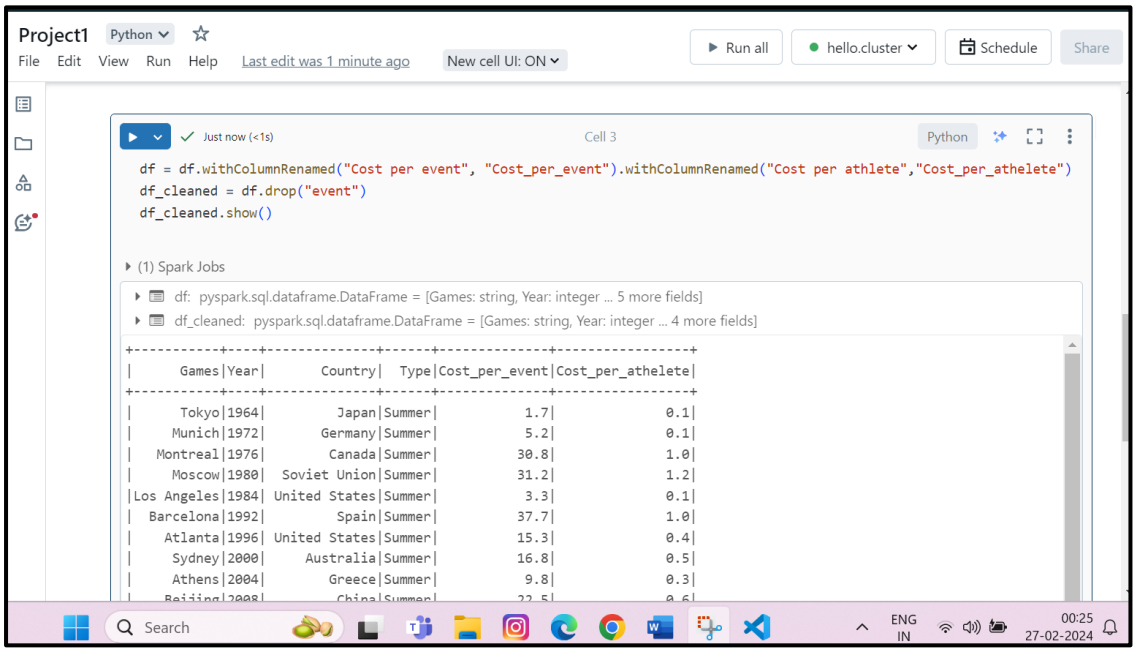


Displaying the schema:

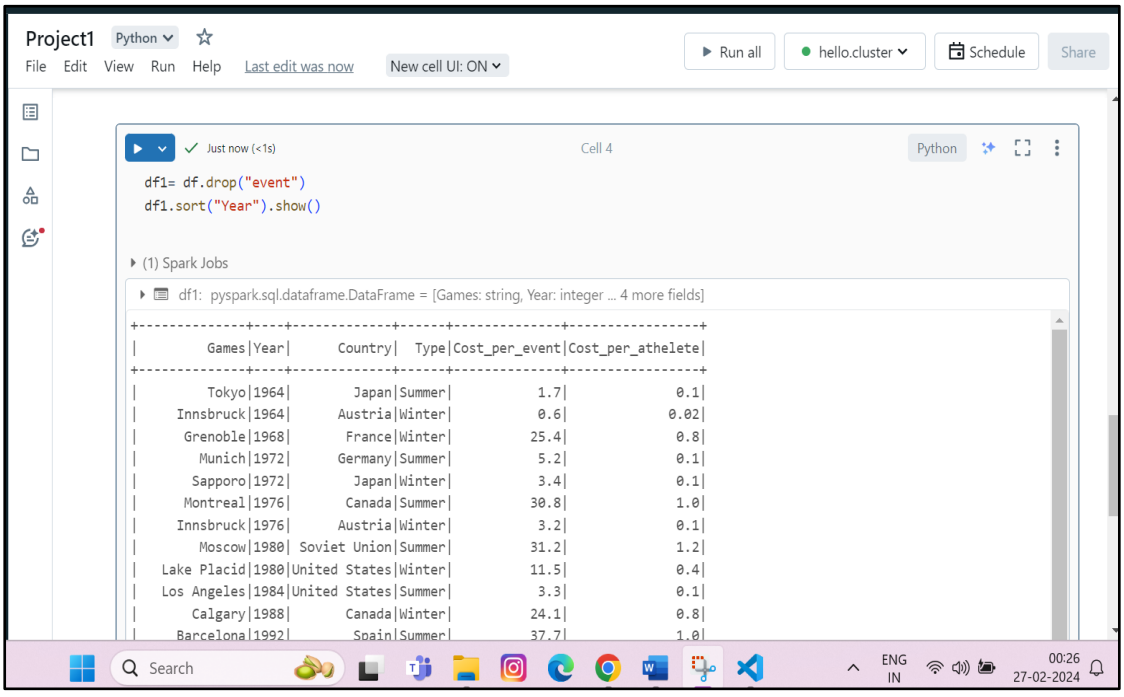


Transformations Done On the Schema

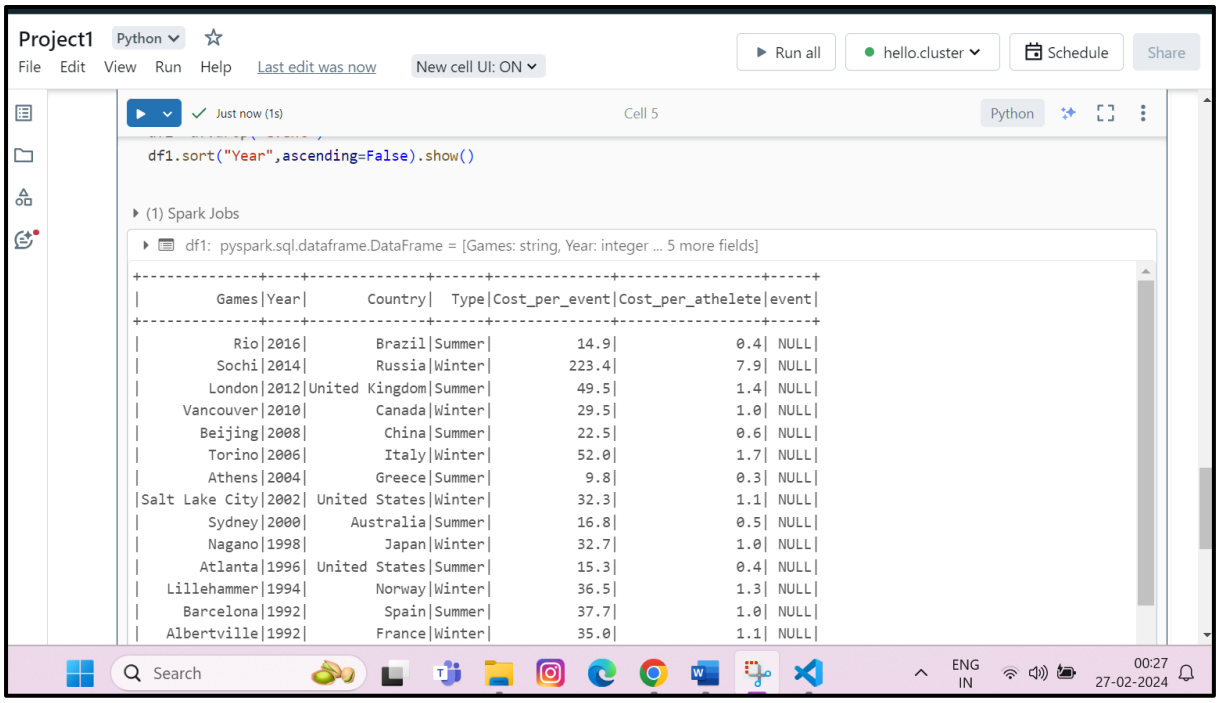
- **Renaming columns and dropping null values:**



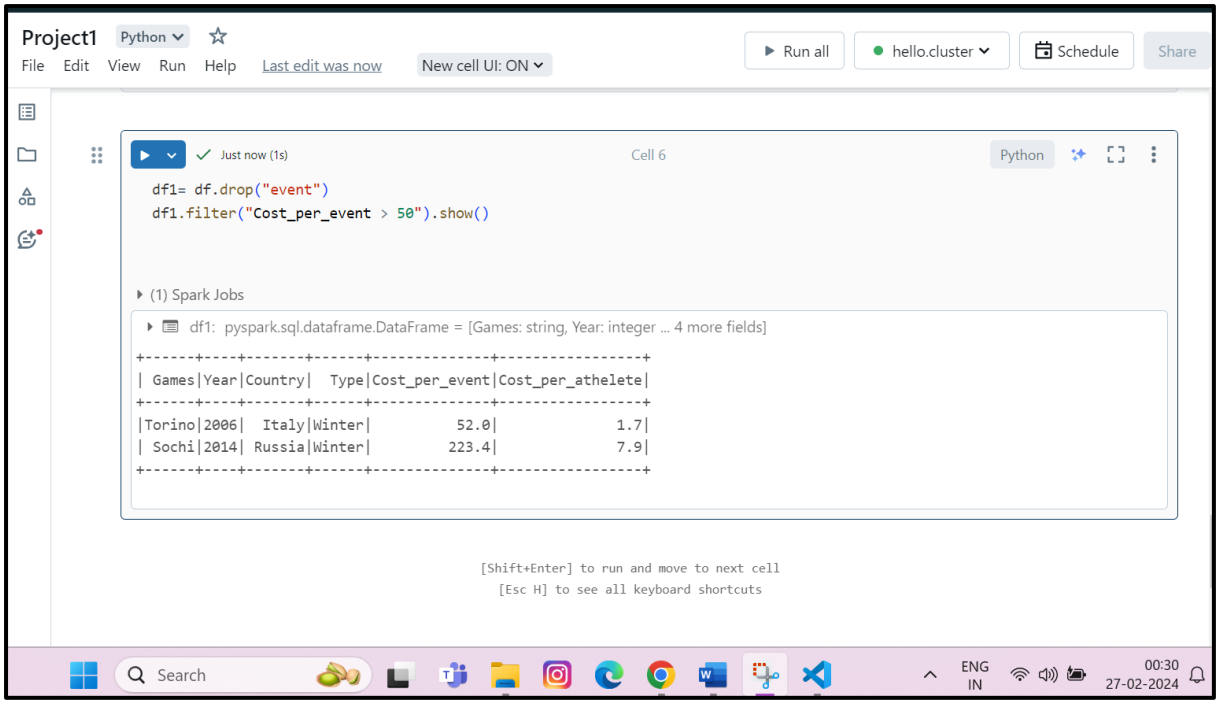
- **Sort in ascending order:**



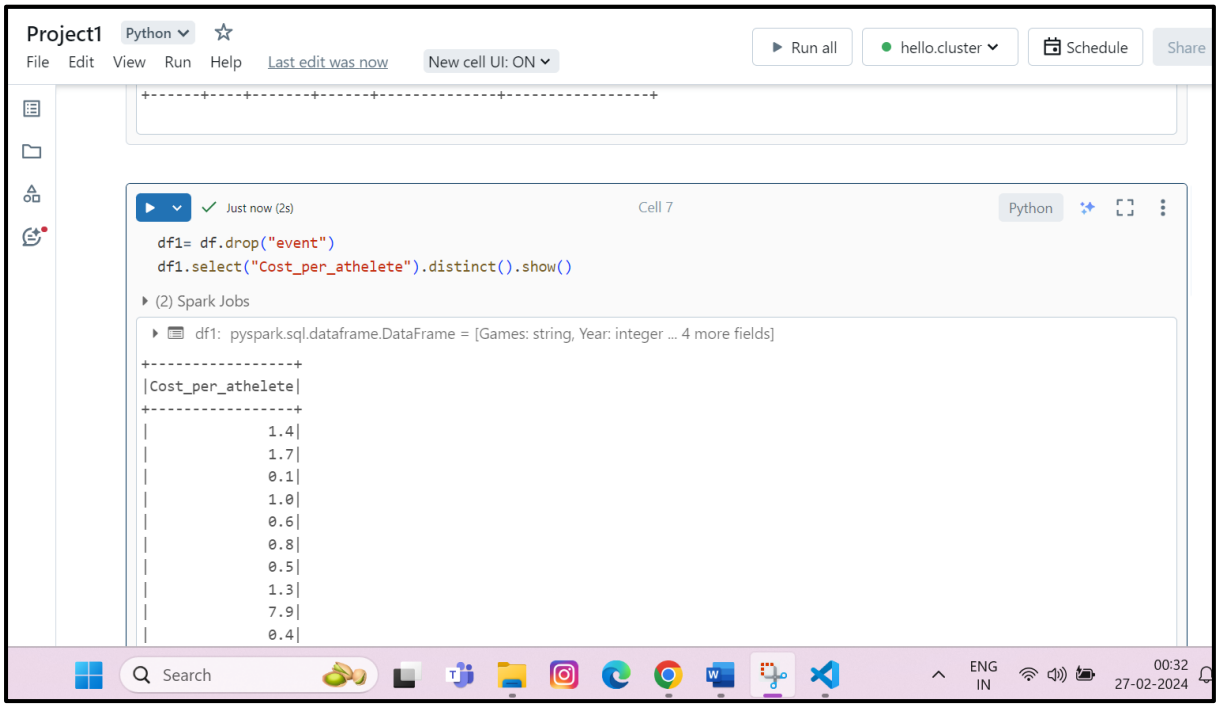
- **Sort in Descending Order:**



- **Filtering:**

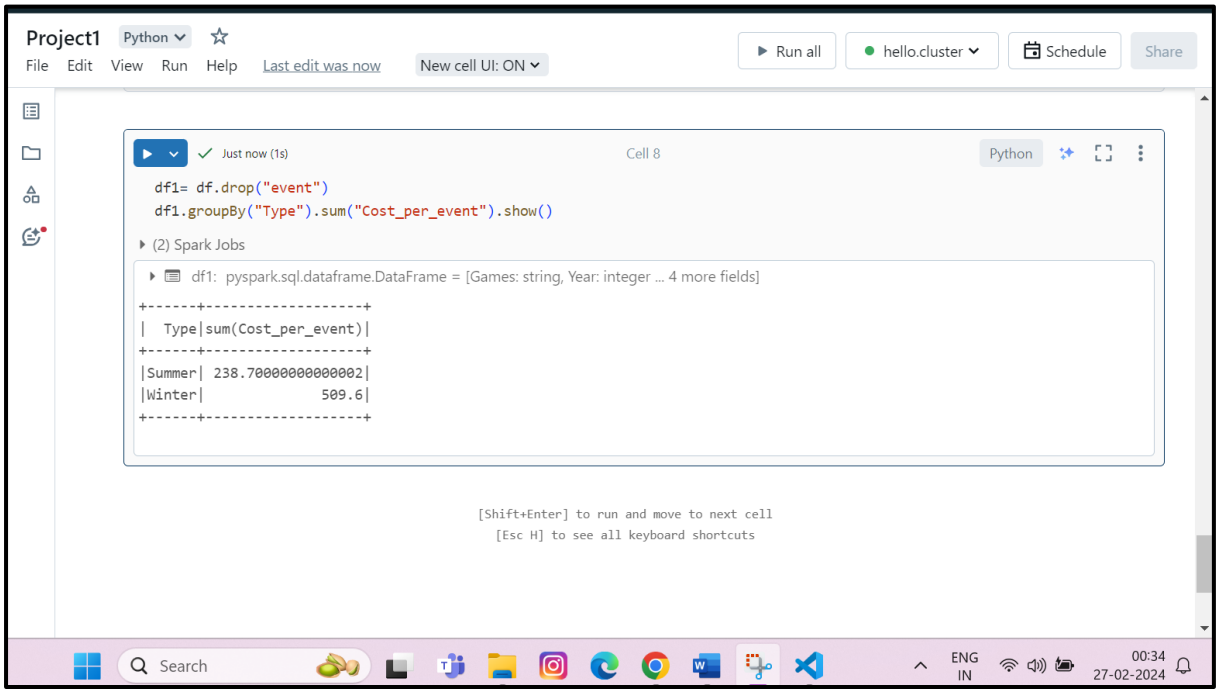


- **Distinct:**

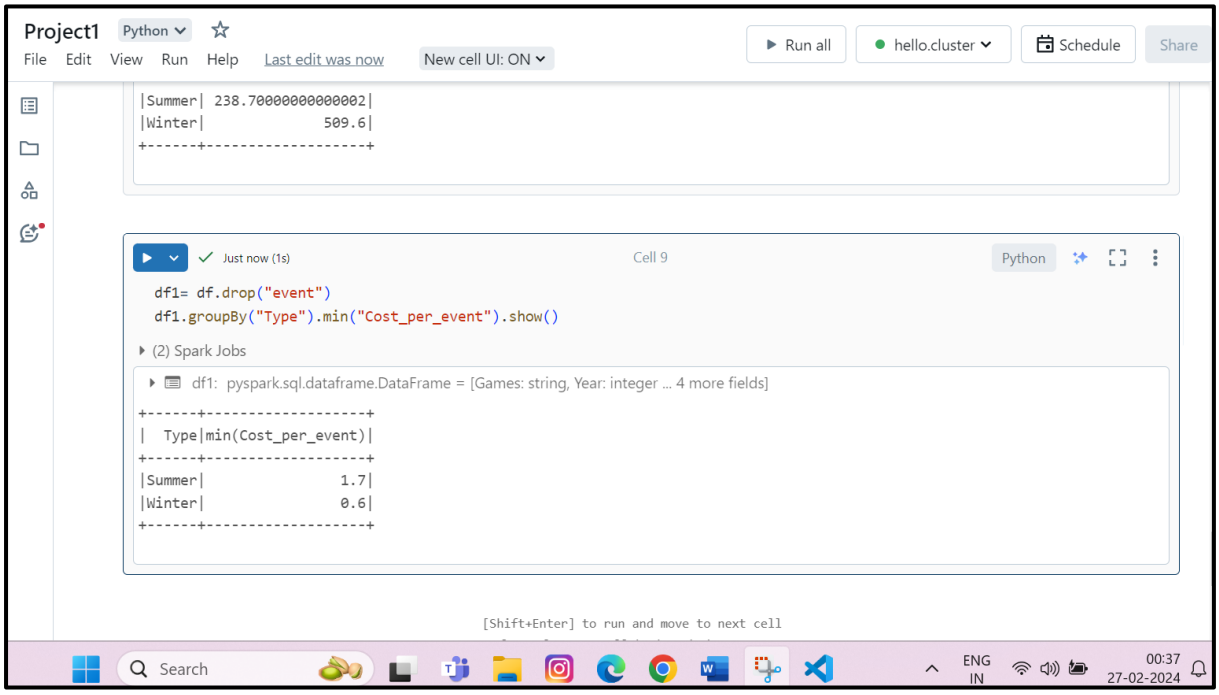


GroupBy with Aggregate Functions

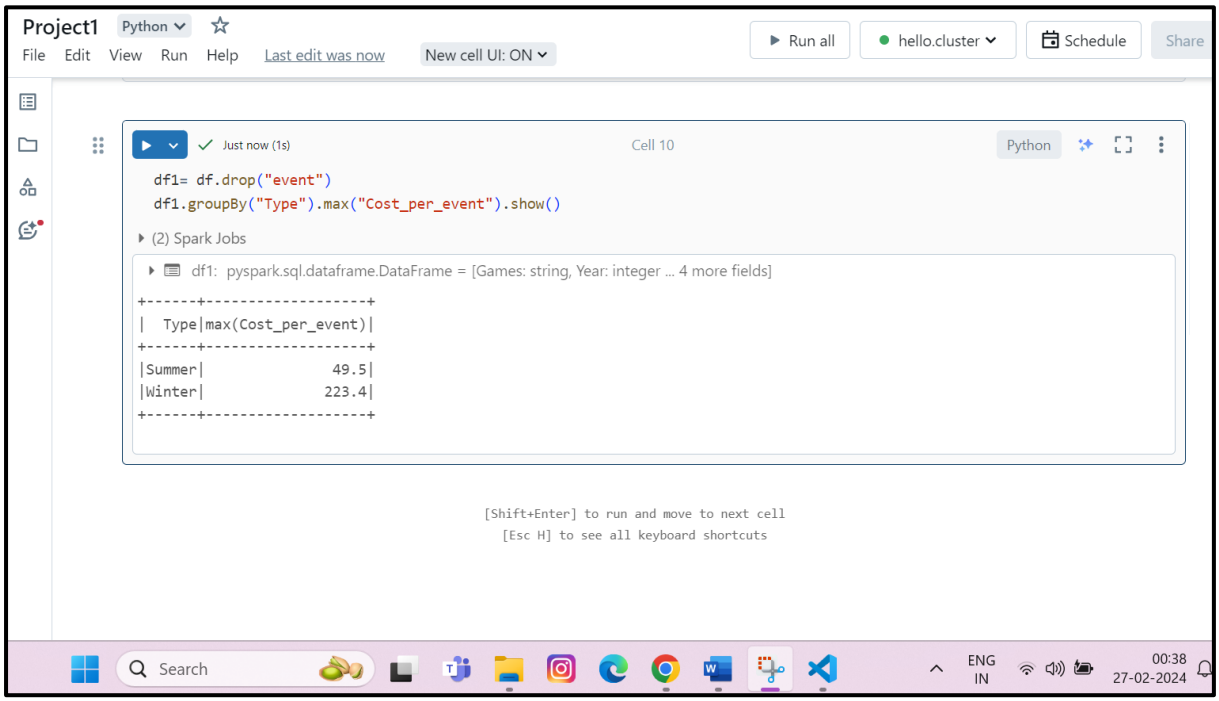
- **Sum:**



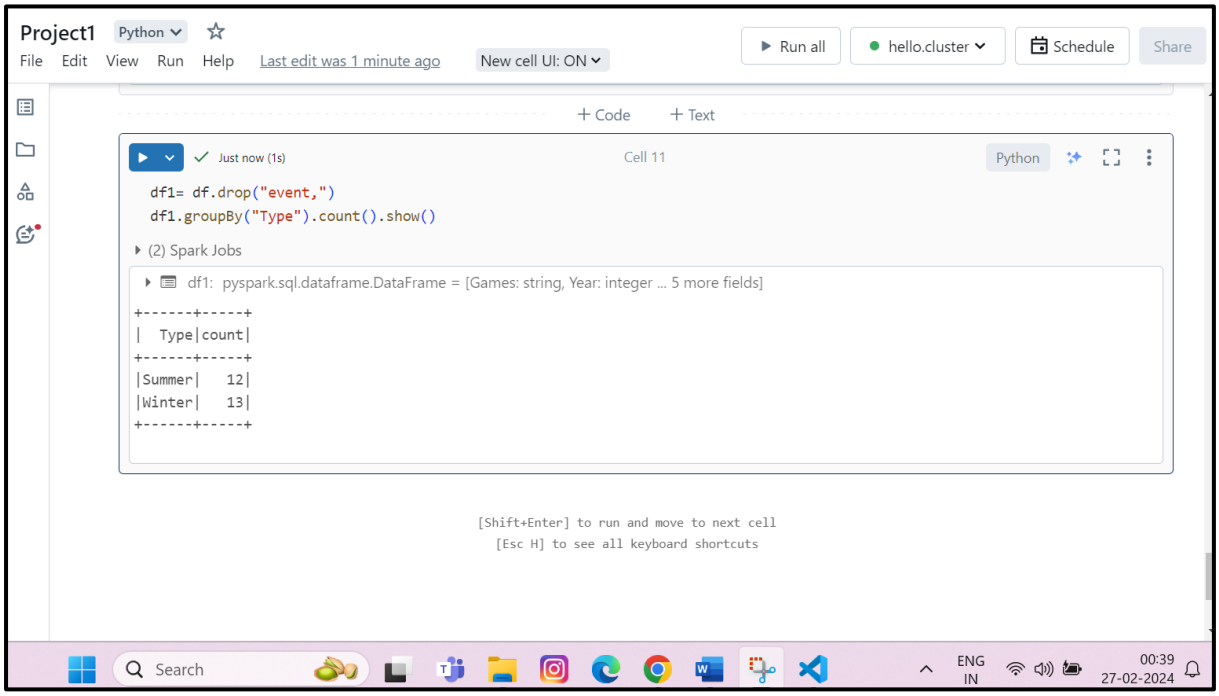
- **Min:**


Project1
Python
File Edit View Run Help Last edit was now New cell UI: ON
Run all
hello.cluster
Schedule
Share
|Summer| 238.70000000000002|
|Winter| 509.6|
Just now (1s)
Cell 9
Python
df1= df.drop("event")
df1.groupBy("Type").min("Cost_per_event").show()
(2) Spark Jobs
df1: pyspark.sql.dataframe.DataFrame = [Games: string, Year: integer ... 4 more fields]
| Type|min(Cost_per_event)|
|Summer| 1.7|
|Winter| 0.6|
[Shift+Enter] to run and move to next cell
Search
ENG IN
00:37
27-02-2024

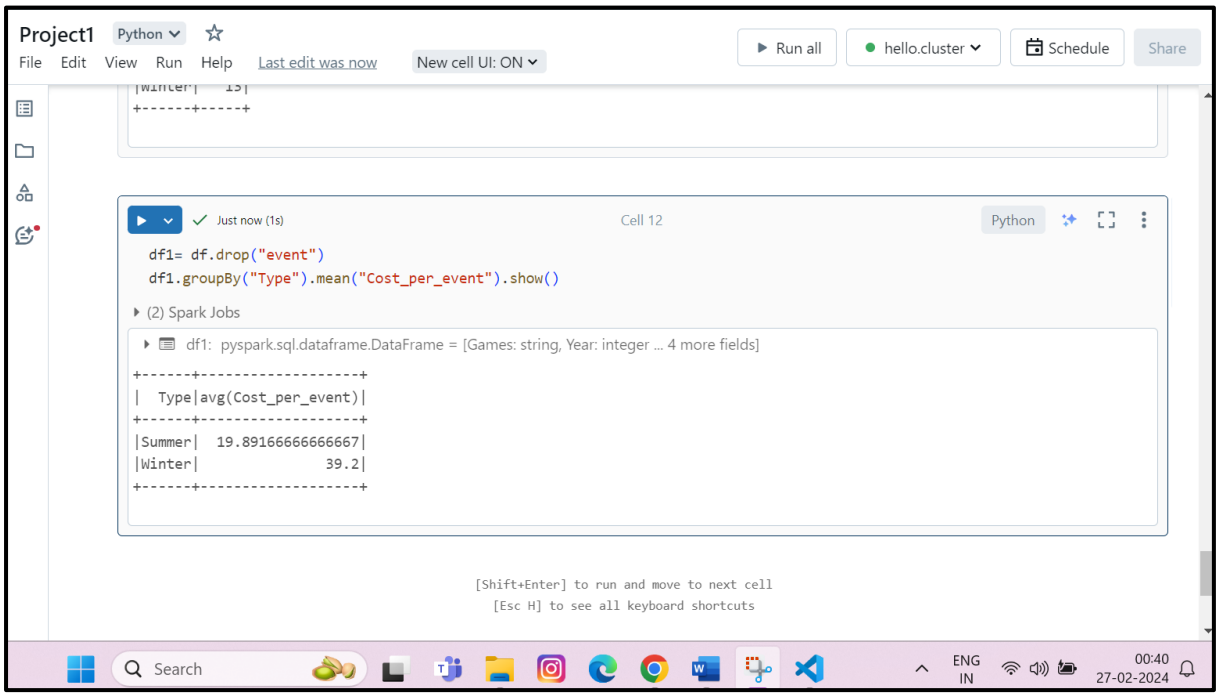
- **Max:**


Project1
Python
File Edit View Run Help Last edit was now New cell UI: ON
Run all
hello.cluster
Schedule
Share
Just now (1s)
Cell 10
Python
df1= df.drop("event")
df1.groupBy("Type").max("Cost_per_event").show()
(2) Spark Jobs
df1: pyspark.sql.dataframe.DataFrame = [Games: string, Year: integer ... 4 more fields]
| Type|max(Cost_per_event)|
|Summer| 49.5|
|Winter| 223.4|
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
Search
ENG IN
00:38
27-02-2024

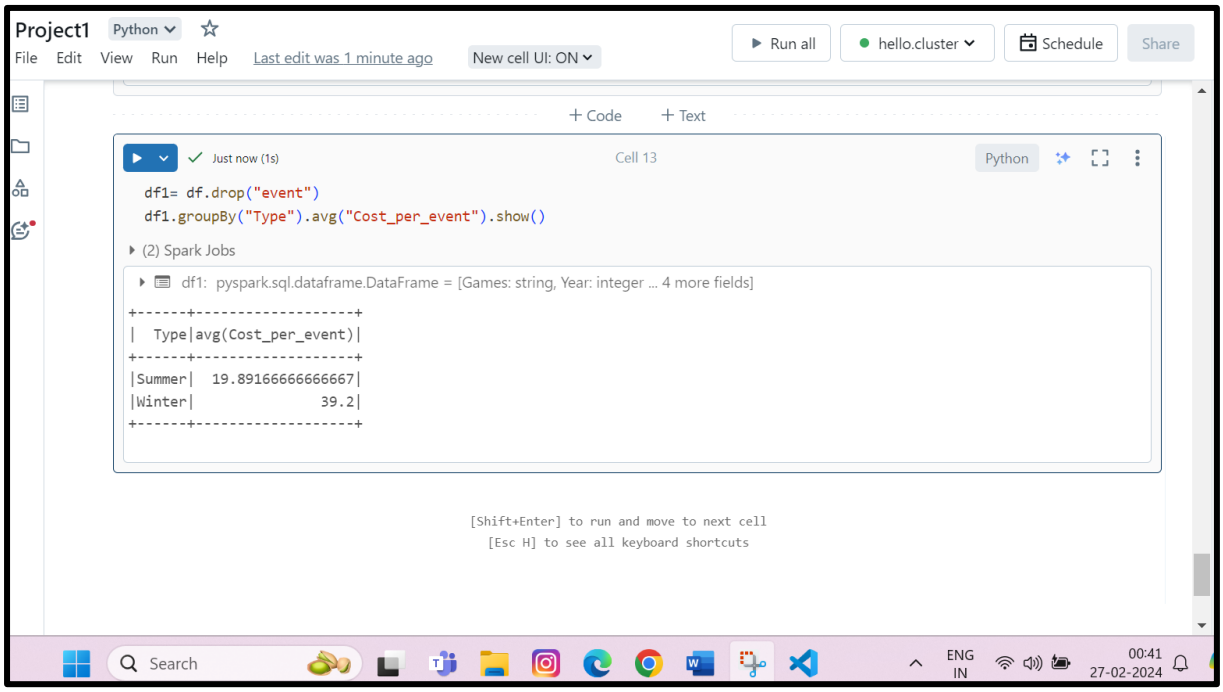
- **Count:**


Project1
Python
File Edit View Run Help
Last edit was 1 minute ago
New cell UI: ON
Run all
hello.cluster
Schedule
Share
+ Code + Text
Just now (1s)
Cell 11
Python
df1= df.drop("event,")
df1.groupBy("Type").count().show()
(2) Spark Jobs
df1: pyspark.sql.dataframe.DataFrame = [Games: string, Year: integer ... 5 more fields]
+------+-----+
| Type|count|
+------+-----+
|Summer| 12|
|Winter| 13|
+------+-----+
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
Search
ENG
IN
00:39
27-02-2024

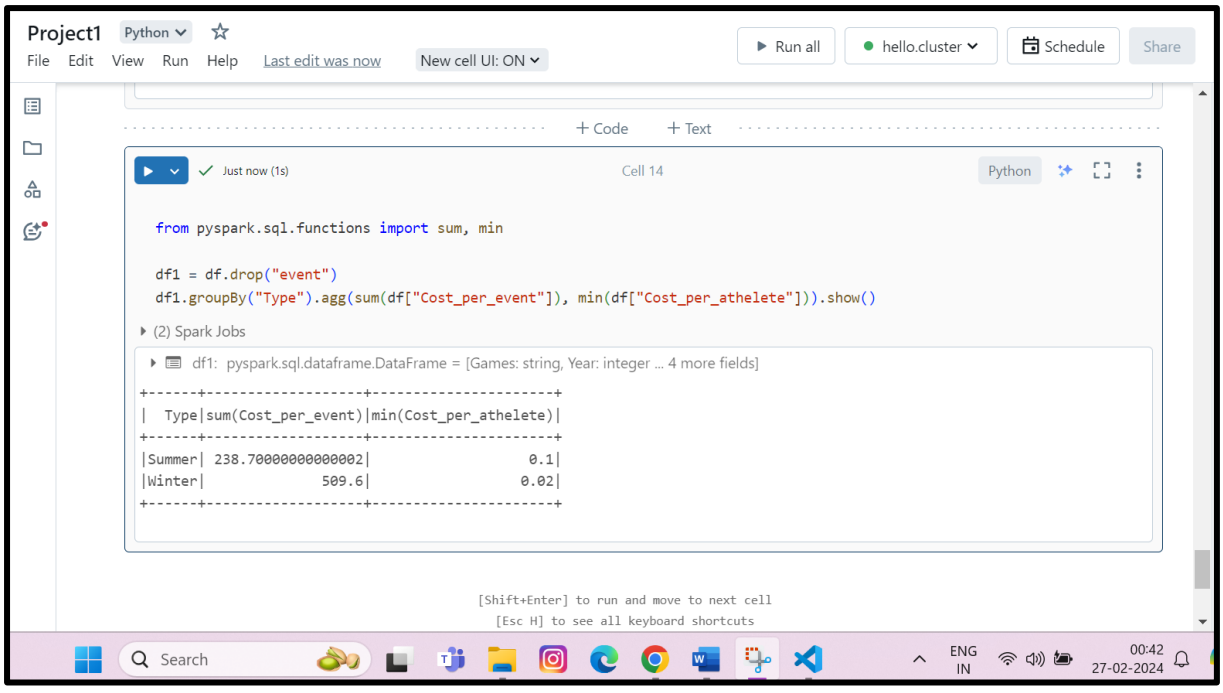
- **Mean:**


Project1
Python
File Edit View Run Help
Last edit was now
New cell UI: ON
Run all
hello.cluster
Schedule
Share
Just now (1s)
Cell 12
Python
df1= df.drop("event")
df1.groupBy("Type").mean("Cost_per_event").show()
(2) Spark Jobs
df1: pyspark.sql.dataframe.DataFrame = [Games: string, Year: integer ... 4 more fields]
+------+-------------------+
| Type|avg(Cost_per_event)|
+------+-------------------+
|Summer| 19.89166666666667|
|Winter| 39.2|
+------+-------------------+
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
Search
ENG
IN
00:40
27-02-2024

- **Avg:**


Project1
Python
File Edit View Run Help
Last edit was 1 minute ago
New cell UI: ON
Run all
hello.cluster
Schedule
Share
+ Code + Text
Just now (1s)
Cell 13
Python
df1= df.drop("event")
df1.groupBy("Type").avg("Cost_per_event").show()
(2) Spark Jobs
df1: pyspark.sql.dataframe.DataFrame = [Games: string, Year: integer ... 4 more fields]
+------+-------------------+
| Type|avg(Cost_per_event)|
+------+-------------------+
|Summer| 19.89166666666667|
|Winter| 39.2|
+------+-------------------+
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
Search
ENG
IN
00:41
27-02-2024

- **Agg**


Project1
Python
File Edit View Run Help
Last edit was now
New cell UI: ON
Run all
hello.cluster
Schedule
Share
+ Code + Text
Just now (1s)
Cell 14
Python
from pyspark.sql.functions import sum, min
df1 = df.drop("event")
df1.groupBy("Type").agg(sum(df["Cost_per_event"]), min(df["Cost_per_athelete"])).show()
(2) Spark Jobs
df1: pyspark.sql.dataframe.DataFrame = [Games: string, Year: integer ... 4 more fields]
+------+-------------------+----------------------+
| Type|sum(Cost_per_event)|min(Cost_per_athelete)|
+------+-------------------+----------------------+
|Summer| 238.70000000000002| 0.1|
|Winter| 509.6| 0.02|
+------+-------------------+----------------------+
[Shift+Enter] to run and move to next cell
[Esc H] to see all keyboard shortcuts
Search
ENG
IN
00:42
27-02-2024

CONCLUSION

In conclusion, the integration of Azure Data Factory (ADF) pipelines for data ingestion and Azure Databricks for data transformation provides a comprehensive solution for handling Olympics data within the Azure ecosystem. ADF simplifies and automates the process of bringing in data from diverse sources and depositing it into Azure Data Lake Storage Gen2. This serves as a reliable and scalable data repository.

On the other hand, Azure Databricks, with its PySpark capabilities, facilitates efficient and parallelized data transformations, ensuring that the raw Olympics data is transformed into a structured and analyzable format. The combination of these services not only streamlines the end-to-end data processing workflow but also takes advantage of the cloud's scalability and computational power.

By leveraging these Azure services, organizations can achieve a seam less and robust pipeline for managing, ingesting, and transforming large volumes of data, allowing for enhanced analytics, reporting, and insights. This approach aligns with modern data engineering best practices and empowers users to derive valuable insights from the Olympics data while benefitting from the flexibility and scalability offered by the Azure cloud platform.