Tokyo Olympics 2020 Data Engineering Project (End-to-End)

Project Overview

This project demonstrates an end-to-end data engineering pipeline using Azure cloud services and Databricks to process and analyze Tokyo Olympics 2020 datasets. The primary objective was to extract raw Olympic data, transform and cleanse it, and load it into an Azure SQL Database for further analysis and reporting.

Tools and Technologies Used

- **Azure Data Lake Storage Gen2**: For storing raw datasets.
- **Azure Databricks**: For data transformation using PySpark.
- **Azure SQL Database**: Final destination for transformed data.
- **Azure Data Factory (ADF)**: Used for orchestration and data ingestion (optional future enhancement).
- **PySpark**: Used for data manipulation and transformation.
- **JDBC**: For writing transformed data into Azure SQL Database.

Datasets Processed

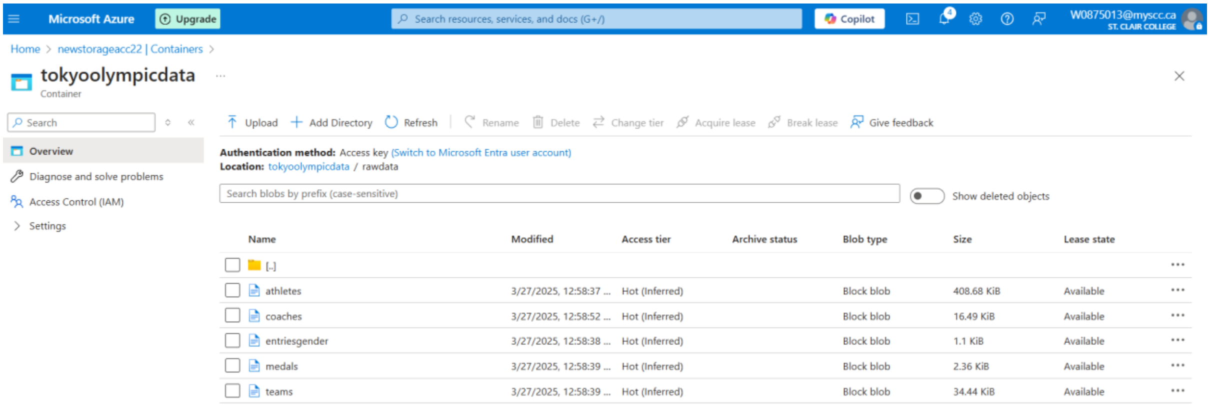
- Athletes
- Teams
- Entries by Gender
- Coaches
- Medals

Steps Implemented

1. Data Ingestion

- Uploaded Tokyo Olympics datasets (CSV files) to Azure Data Lake Storage Gen2.
- Created a bronze layer folder for raw files.

- Created a pipeline using copy activity to transfer files to transformeddata folder.



Home > newstorageacc22 | Containers >

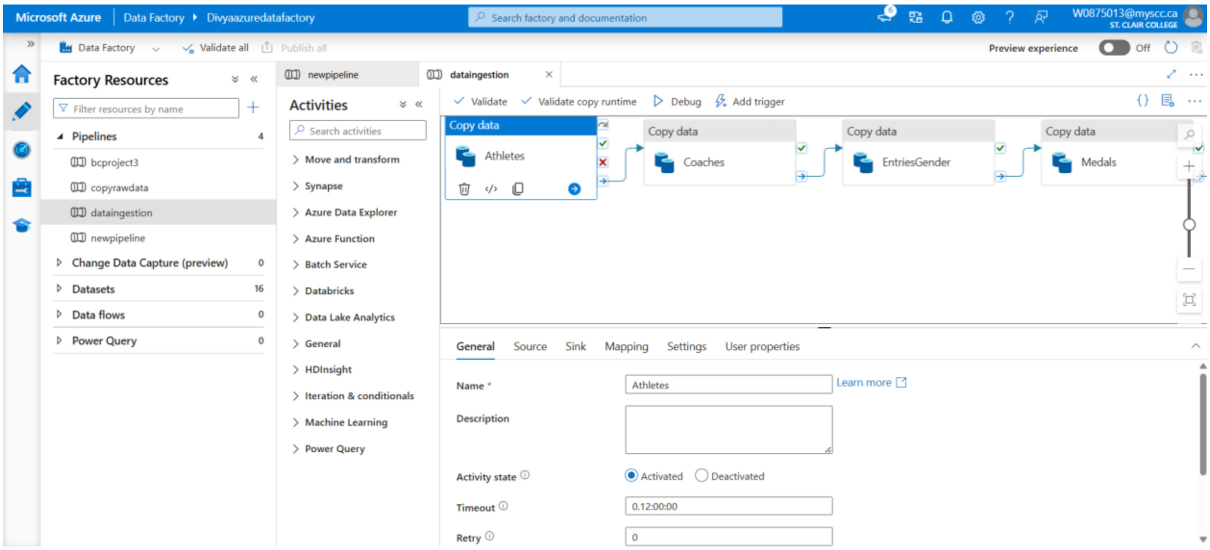
tokyoolympicdata

Container

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: tokyoolympicdata / rawdata

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
[..]						
athletes	3/27/2025, 12:58:37 ...	Hot (Inferred)		Block blob	408.68 KiB	Available
coaches	3/27/2025, 12:58:52 ...	Hot (Inferred)		Block blob	16.49 KiB	Available
entriesgender	3/27/2025, 12:58:38 ...	Hot (Inferred)		Block blob	1.1 KiB	Available
medals	3/27/2025, 12:58:39 ...	Hot (Inferred)		Block blob	2.36 KiB	Available
teams	3/27/2025, 12:58:39 ...	Hot (Inferred)		Block blob	34.44 KiB	Available



Factory Resources	
Pipelines	4
bcproject3	
copyrawdata	
dataingestion	
newpipeline	
Change Data Capture (preview)	0
Datasets	16
Data flows	0
Power Query	0

Name * Athletes

Description

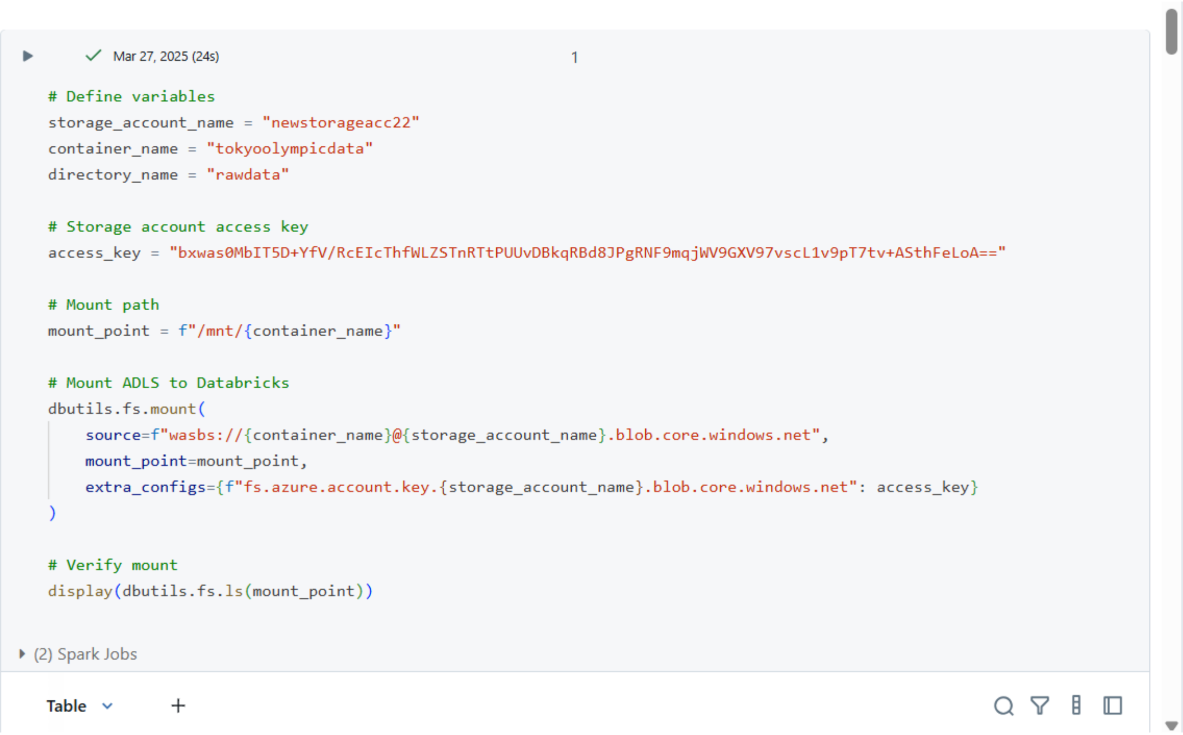
Activity state ● Activated ○ Deactivated

Timeout 0.12:00:00

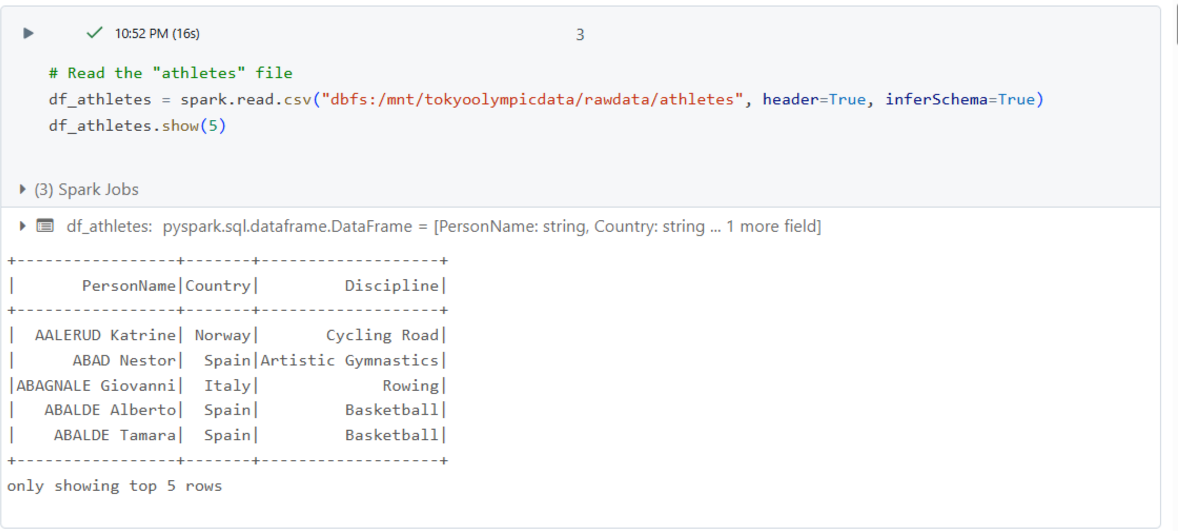
Retry 0

2. Data Transformation with Databricks

- Read raw CSVs using PySpark into DataFrames.
- Applied necessary transformations like schema definition, null handling, and finding the top 10 countries with the highest number of gold medals
- Finding the total number of athletes per country
- Validated each DataFrame to ensure data quality and schema consistency.



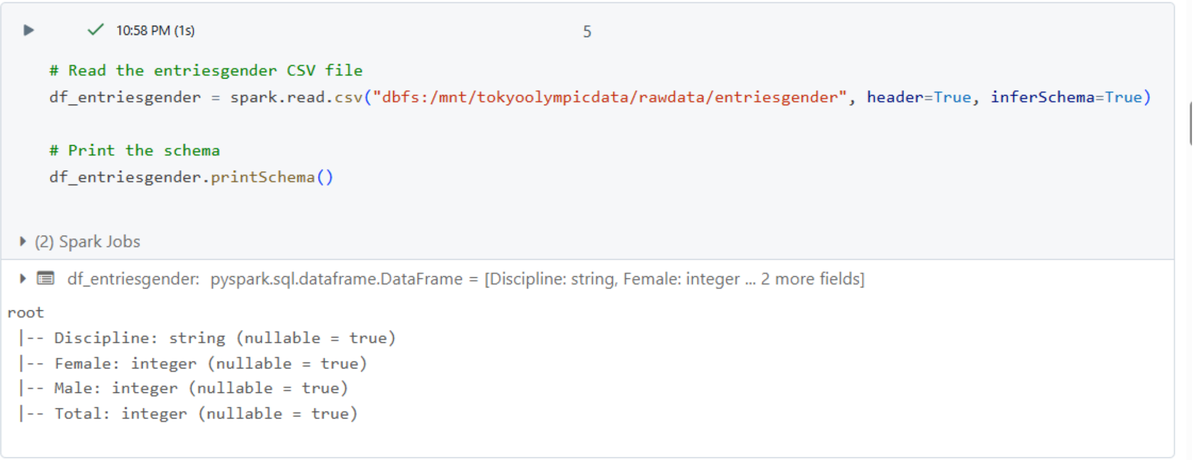
```
# Define variables
storage_account_name = "newstorageacc22"
container_name = "tokyoolympicdata"
directory_name = "rawdata"

# Storage account access key
access_key = "bxwas0MbIT5D+YfV/RcEIcThfWLZSTnRTtPUUvDBkqRBd8JPgRNF9mqjWV9GXV97vscL1v9pT7tv+ASthFeLoA=="

# Mount path
mount_point = f"/mnt/{container_name}"

# Mount ADLS to Databricks
dbutils.fs.mount(
    source=f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",
    mount_point=mount_point,
    extra_configs={f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": access_key}
)

# Verify mount
display(dbutils.fs.ls(mount_point))
```



```
# Read the "athletes" file
df_athletes = spark.read.csv("dbfs:/mnt/tokyoolympicdata/rawdata/athletes", header=True, inferSchema=True)
df_athletes.show(5)
```

df_athletes: pyspark.sql.dataframe.DataFrame = [PersonName: string, Country: string ... 1 more field]

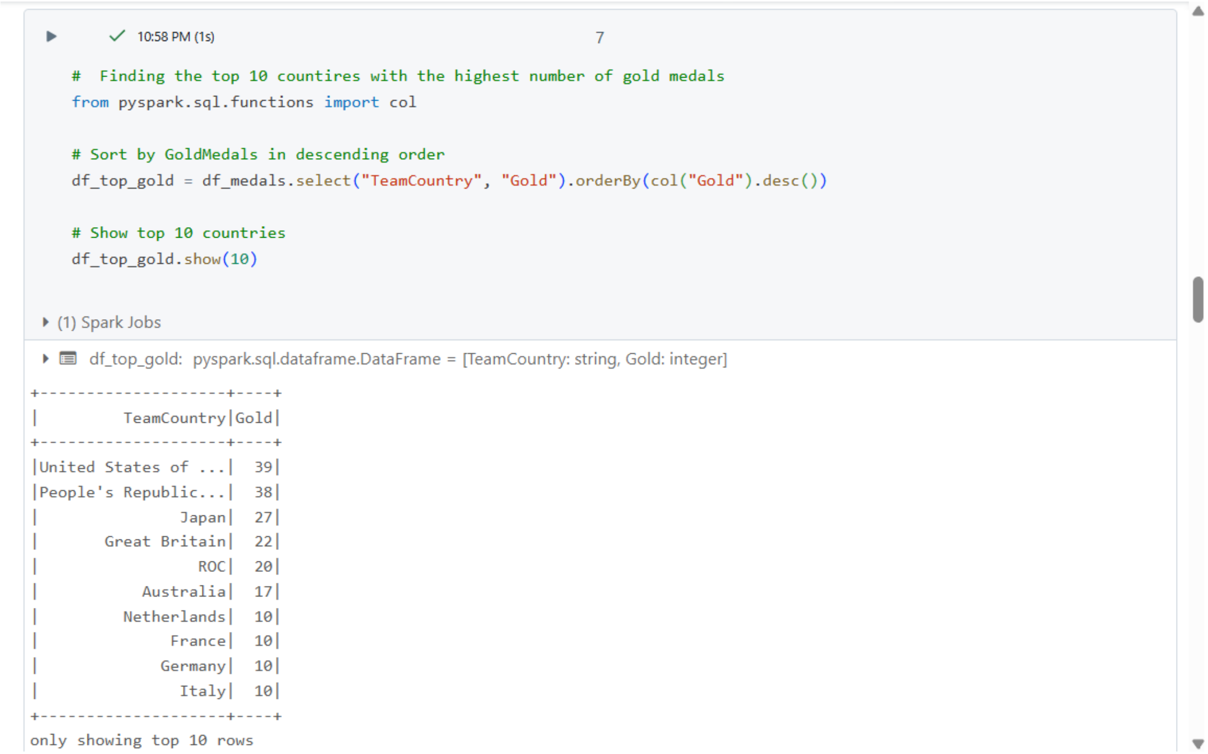
```
+-----------------+-------+------------------+
|       PersonName|Country|        Discipline|
+-----------------+-------+------------------+
|  AALERUD Katrine| Norway|      Cycling Road|
|      ABAD Nestor|  Spain|Artistic Gymnastics|
|ABAGNALE Giovanni|  Italy|            Rowing|
|   ABALDE Alberto|  Spain|        Basketball|
|    ABALDE Tamara|  Spain|        Basketball|
+-----------------+-------+------------------+
only showing top 5 rows
```



```
# Read the entriesgender CSV file
df_entriesgender = spark.read.csv("dbfs:/mnt/tokyoolympicdata/rawdata/entriesgender", header=True, inferSchema=True)

# Print the schema
df_entriesgender.printSchema()
```

df_entriesgender: pyspark.sql.dataframe.DataFrame = [Discipline: string, Female: integer ... 2 more fields]

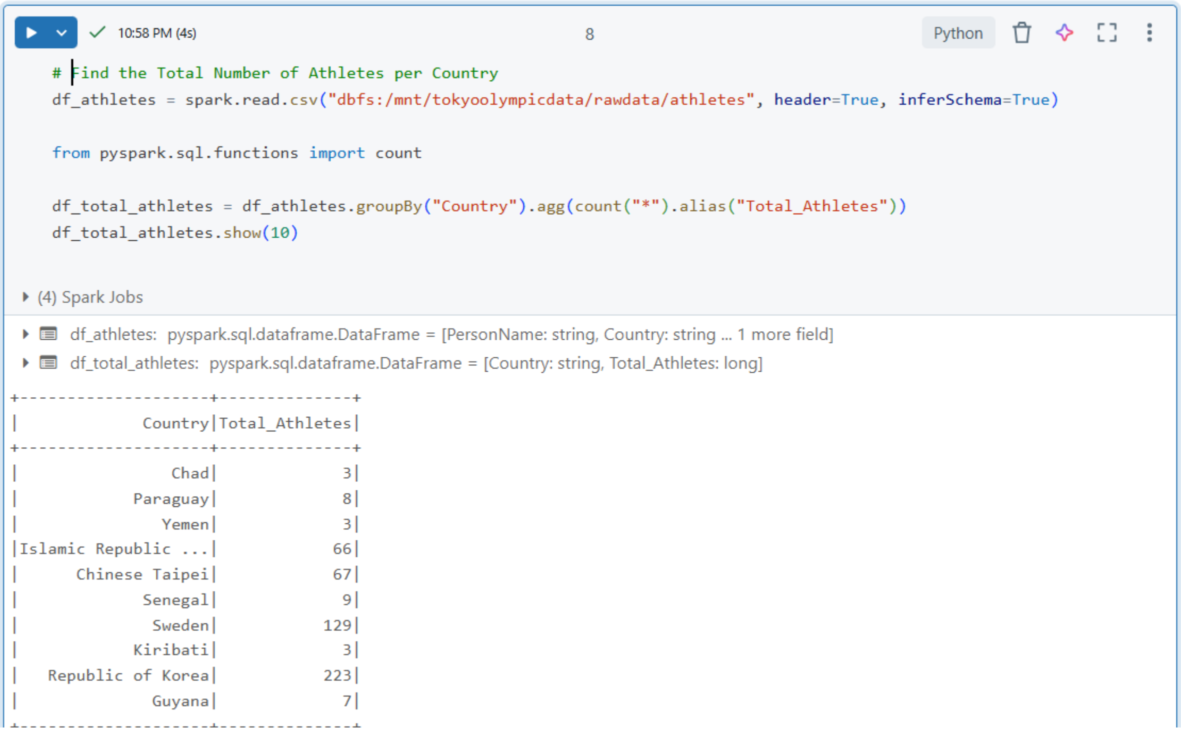
```
root
 |-- Discipline: string (nullable = true)
 |-- Female: integer (nullable = true)
 |-- Male: integer (nullable = true)
 |-- Total: integer (nullable = true)
```



```
#  Finding the top 10 countires with the highest number of gold medals
from pyspark.sql.functions import col

# Sort by GoldMedals in descending order
df_top_gold = df_medals.select("TeamCountry", "Gold").orderBy(col("Gold").desc())

# Show top 10 countries
df_top_gold.show(10)
```

df_top_gold: pyspark.sql.dataframe.DataFrame = [TeamCountry: string, Gold: integer]

```
+--------------------+----+
|         TeamCountry|Gold|
+--------------------+----+
|United States of ...|  39|
|People's Republic...|  38|
|               Japan|  27|
|       Great Britain|  22|
|                 ROC|  20|
|           Australia|  17|
|         Netherlands|  10|
|              France|  10|
|             Germany|  10|
|               Italy|  10|
+--------------------+----+
only showing top 10 rows
```



```
# Find the Total Number of Athletes per Country
df_athletes = spark.read.csv("dbfs:/mnt/tokyoolympicdata/rawdata/athletes", header=True, inferSchema=True)

from pyspark.sql.functions import count

df_total_athletes = df_athletes.groupBy("Country").agg(count("*").alias("Total_Athletes"))
df_total_athletes.show(10)
```

df_athletes: pyspark.sql.dataframe.DataFrame = [PersonName: string, Country: string ... 1 more field]

df_total_athletes: pyspark.sql.dataframe.DataFrame = [Country: string, Total_Athletes: long]

```
+--------------------+--------------+
|             Country|Total_Athletes|
+--------------------+--------------+
|                Chad|             3|
|            Paraguay|             8|
|               Yemen|             3|
|Islamic Republic ...|            66|
|      Chinese Taipei|            67|
|             Senegal|             9|
|              Sweden|           129|
|            Kiribati|             3|
|   Republic of Korea|           223|
|              Guyana|             7|
+--------------------+--------------+
```

3. JDBC Connection Setup

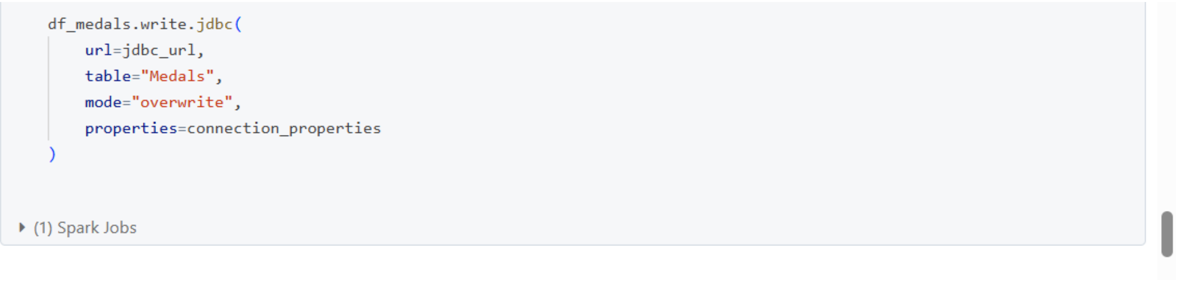
jdbc_url = "jdbc:sqlserver://divyaproject.database.windows.net:1433;database=bcproject3;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;"

```
connection_properties = {
    "user": "Yourusername",
    "password": "Your password",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}
```

4. Data Loading to Azure SQL Database

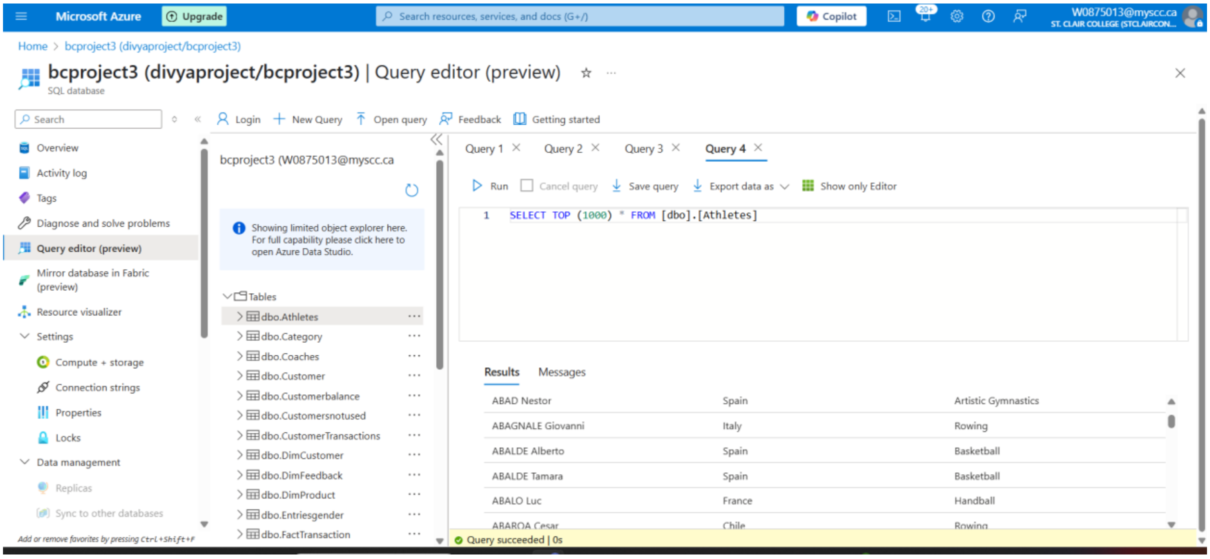
Transformed DataFrames were written to Azure SQL Database using the .write.jdbc() method:

```
df_athletes.write.jdbc(url=jdbc_url, table="Athletes", mode="overwrite", properties=connection_properties)

df_teams.write.jdbc(url=jdbc_url, table="Teams", mode="overwrite", properties=connection_properties)

df_entriesgender.write.jdbc(url=jdbc_url, table="EntriesGender", mode="overwrite", properties=connection_properties)

df_coaches.write.jdbc(url=jdbc_url, table="Coaches", mode="overwrite", properties=connection_properties)

df_medals.write.jdbc(url=jdbc_url, table="Medals", mode="overwrite", properties=connection_properties)
```



```
df_medals.write.jdbc(
    url=jdbc_url,
    table="Medals",
    mode="overwrite",
    properties=connection_properties
)
```

Output from the SQL Database for Athletes file:



Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

```
1    SELECT TOP (1000) * FROM [dbo].[Athletes]
```

ABAD Nestor	Spain	Artistic Gymnastics
ABAGNALE Giovanni	Italy	Rowing
ABALDE Alberto	Spain	Basketball
ABALDE Tamara	Spain	Basketball
ABALO Luc	France	Handball
ABAROA Cesar	Chile	Rowing

Query succeeded | 0s

- Similarly, other tables like medals, teams, coaches and entriesgender were created.

Outcome

- Successfully built and tested an ETL pipeline that processes and stores Tokyo Olympics data into an Azure SQL Database.
- Created a scalable and repeatable process that can be extended to other Olympic datasets.