Tokyo Olympics Azure Data Engineering Project

1. Project Overview

This project demonstrates an end-to-end **Azure Data Engineering pipeline** built using modern, cloud-native services. The goal was to ingest raw Tokyo Olympics data, process and transform it using Apache Spark, store it in a scalable data lake, expose it through Azure Synapse Analytics, and finally build interactive dashboards in **Power BI** for analytics and reporting.

The project closely follows the architecture and implementation shown in the referenced tutorial videos (Part 1 and Part 2), with additional integration of **Power BI** as the final consumption layer.

---

2. Architecture Overview

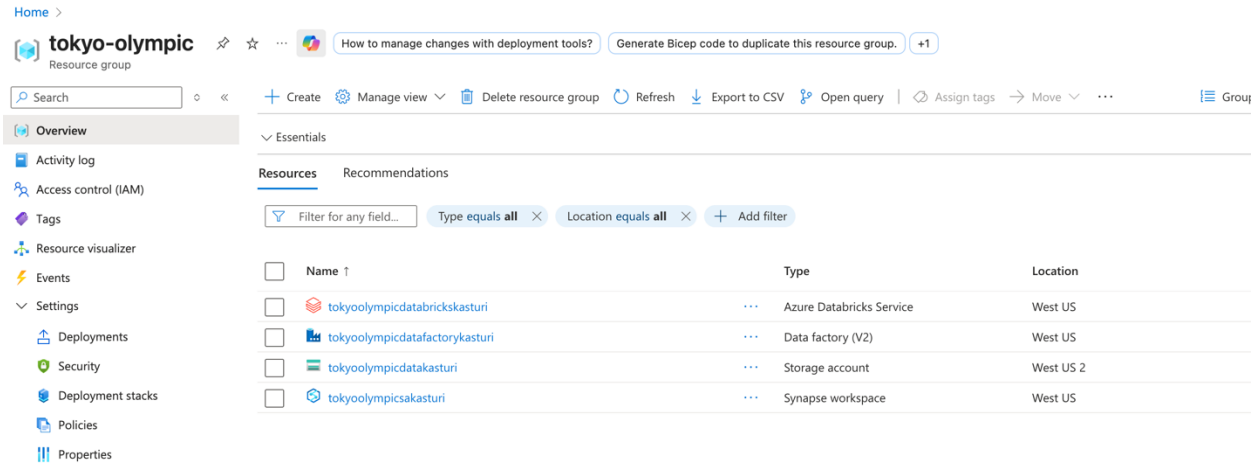
High-level Architecture

```
GitHub (Raw CSV data)
        ↓
Azure Data Factory (Ingestion)
        ↓
Azure Data Lake Storage Gen2 (Raw Zone)
        ↓
Azure Databricks (Apache Spark – Transformations)
        ↓
Azure Data Lake Storage Gen2 (Transformed / Curated Zone)
        ↓
Azure Synapse Analytics (Serverless SQL + Lake Database)
        ↓
Power BI (Dashboards & Reports)
```

Key Design Principles

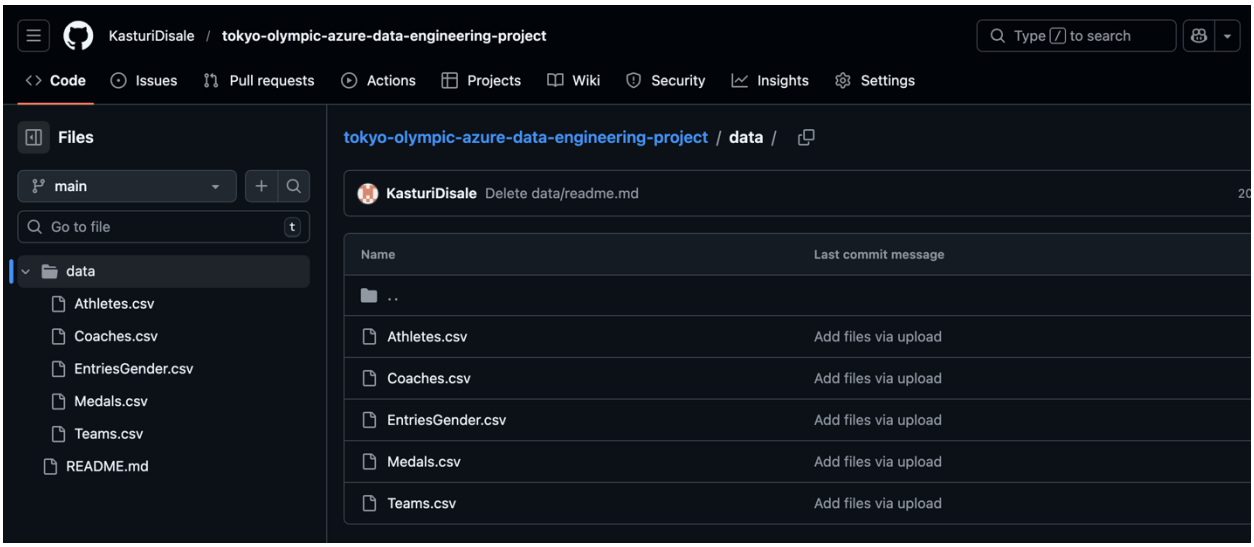
- Separation of **storage** and **compute**
- Lakehouse-style architecture (Raw → Curated)
- Serverless analytics for cost efficiency
- SQL-based semantic layer for BI tools

Created a resource storage for the whole project:



3. Data Source

- Dataset: **Tokyo Olympics Data**
- Format: CSV
- Source: GitHub repository (accessed via raw file URLs)



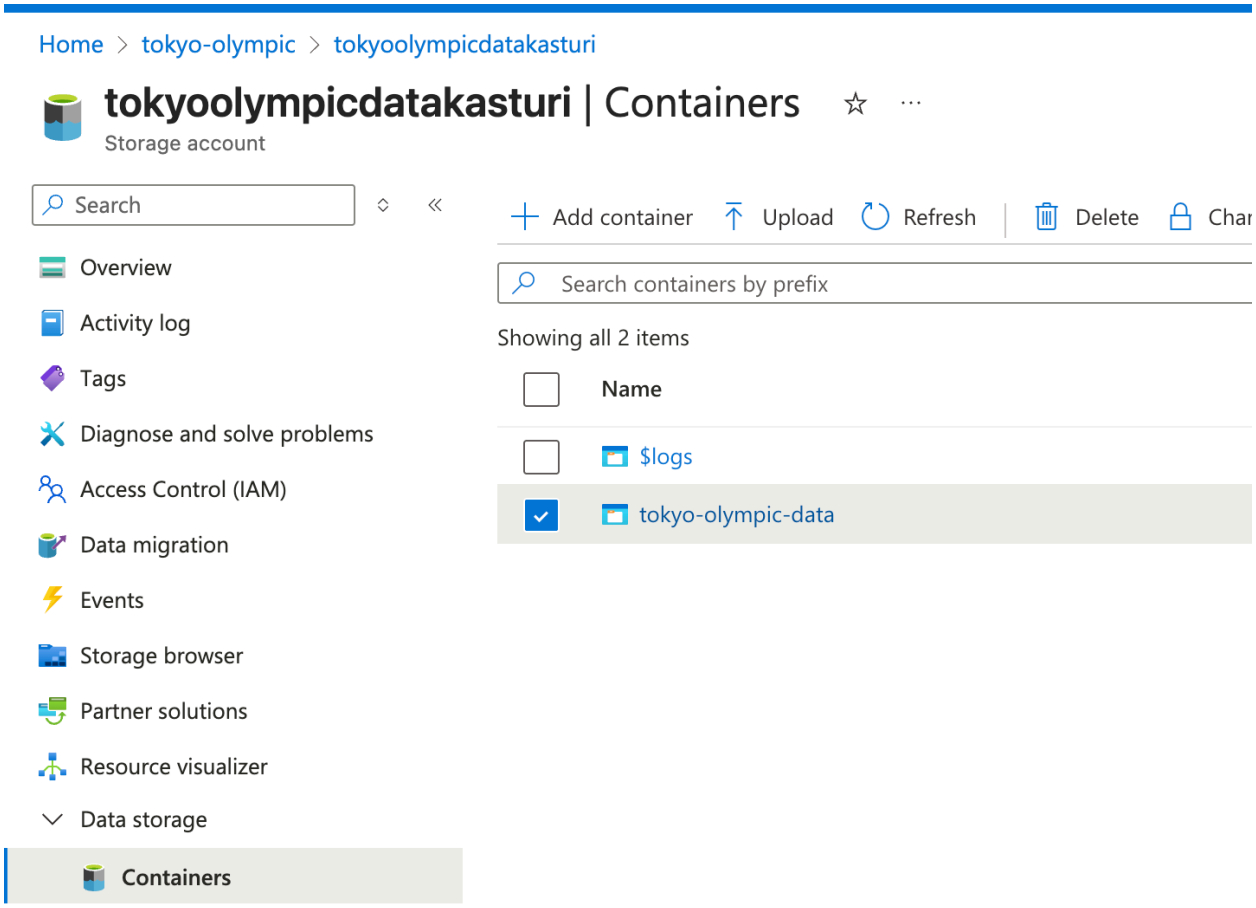
- Example entities:
  - Athletes
  - Teams
  - Coaches
  - Medals
  - Entries Gender

These CSV files served as the raw input for the pipeline.

4. Azure Data Lake Storage Gen2 Setup

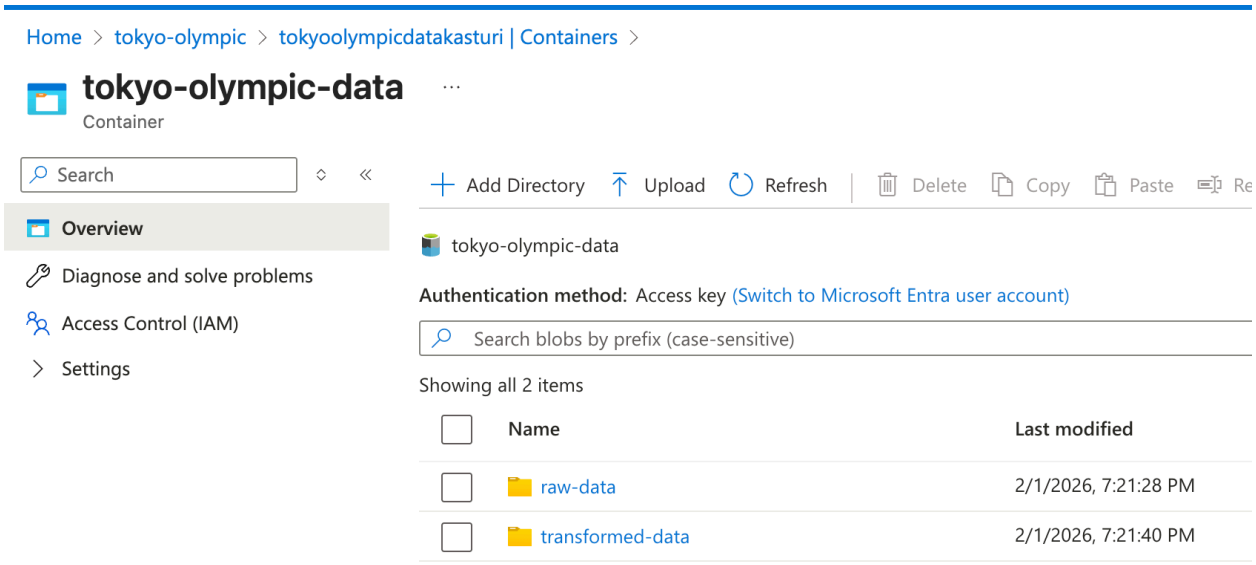
Storage Account

- Type: Azure Data Lake Storage Gen2
- Used as the central data lake for the project



Container Structure

- Raw Data: To store the raw data files from GitHub.
- Transformed Data: To store the transformed data files.



This layered approach ensures data traceability, reprocessing capability, and clean separation between raw and processed datasets.

---

5. Data Ingestion using Azure Data Factory

Azure Data Factory (ADF) was used to ingest raw CSV data from GitHub into ADLS Gen2.

Steps Performed

1. Created **Linked Services** for:
    - HTTP (GitHub raw URLs)
    - Azure Data Lake Storage Gen2
2. Built **pipelines** using Copy Data activities
3. Configured datasets for CSV files
4. Loaded data into the **raw data** folder

Outcome

- Raw Tokyo Olympics CSV files successfully landed in ADLS Gen2
- Pipelines were parameterized and reusable

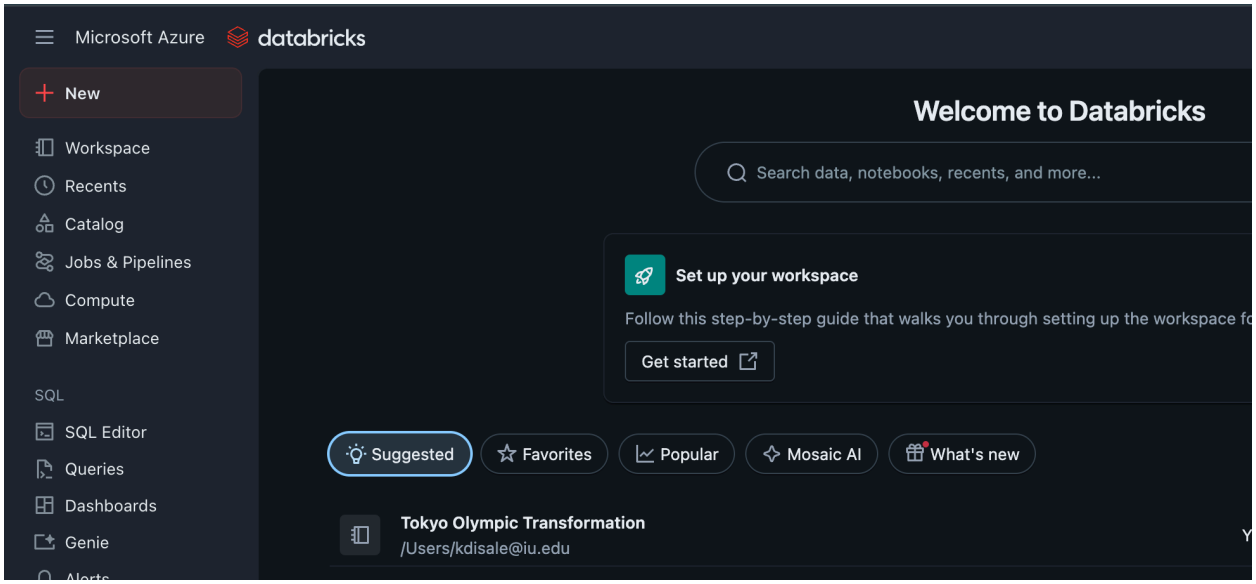
---

6. Data Transformation using Azure Databricks

Azure Databricks with **Apache Spark** was used for data cleansing, transformation, and enrichment.

Databricks Setup

- Workspace created in Azure
- Apache Spark cluster configured
- Secure access to ADLS Gen2 using Service Principal / OAuth



Transformation Logic

Using PySpark:

- Read raw CSV data from ADLS Gen2
- Applied schema inference and data type corrections
- Handled null values and inconsistent columns
- Renamed columns for consistency
- Wrote transformed data in **transformed data folder**

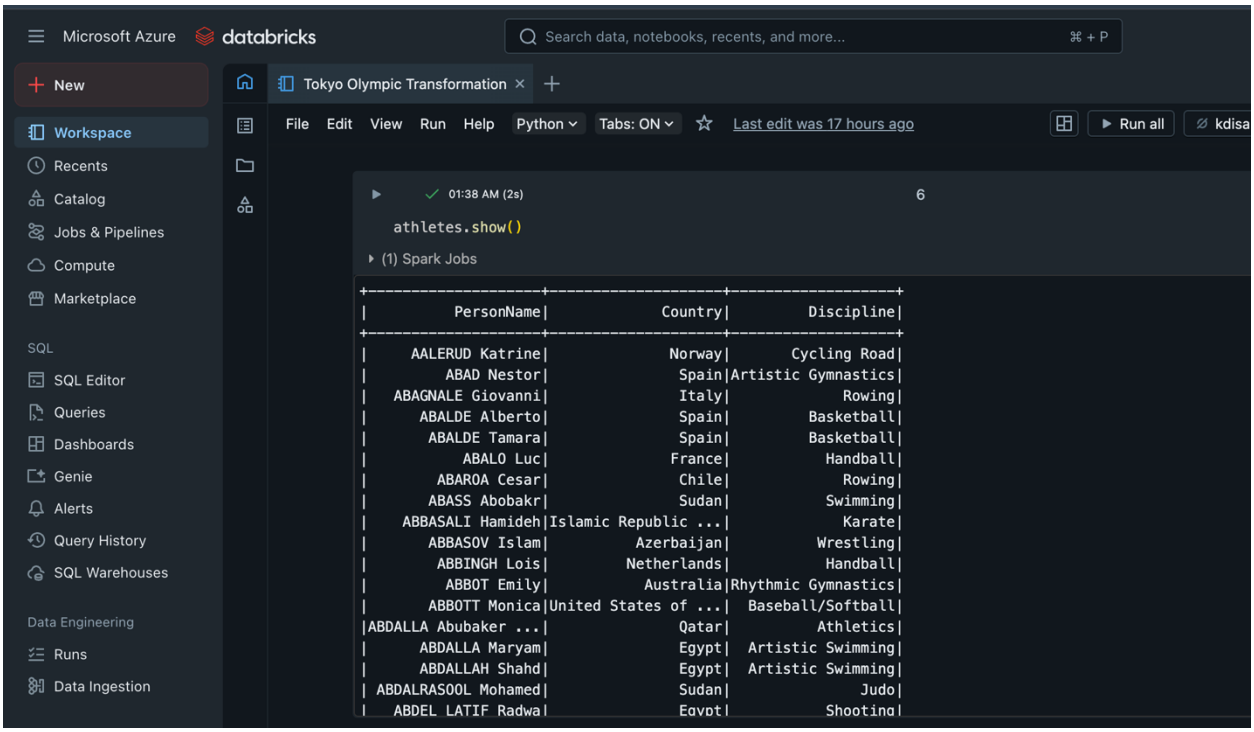
Example Spark Operations

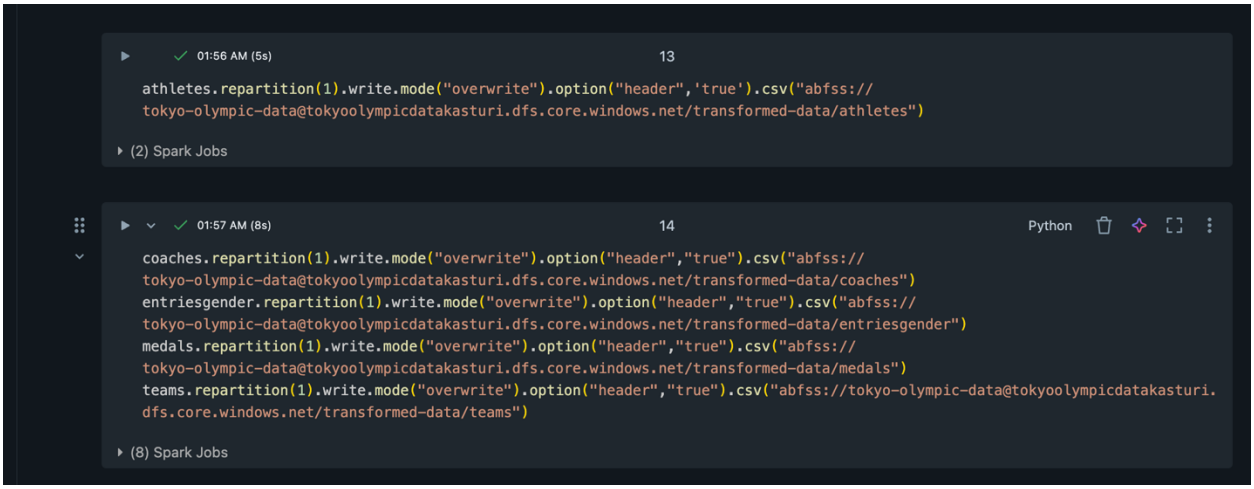
- `spark.read.csv()`
- Column transformations
- Data validation
- `write.mode("overwrite").parquet()`

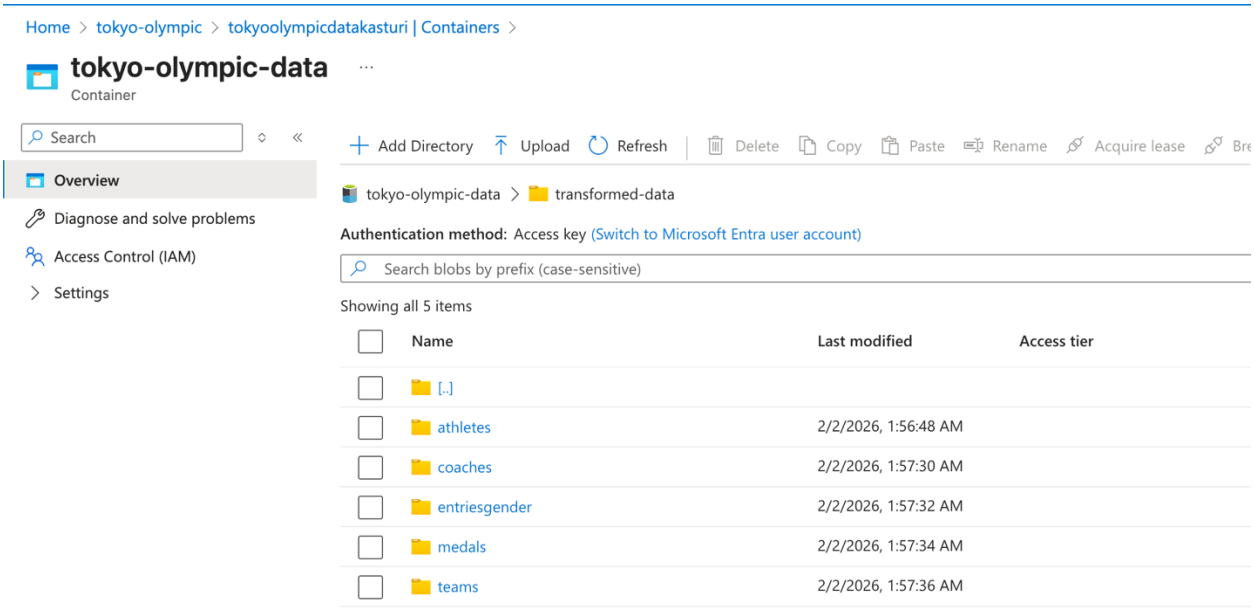
Outcome

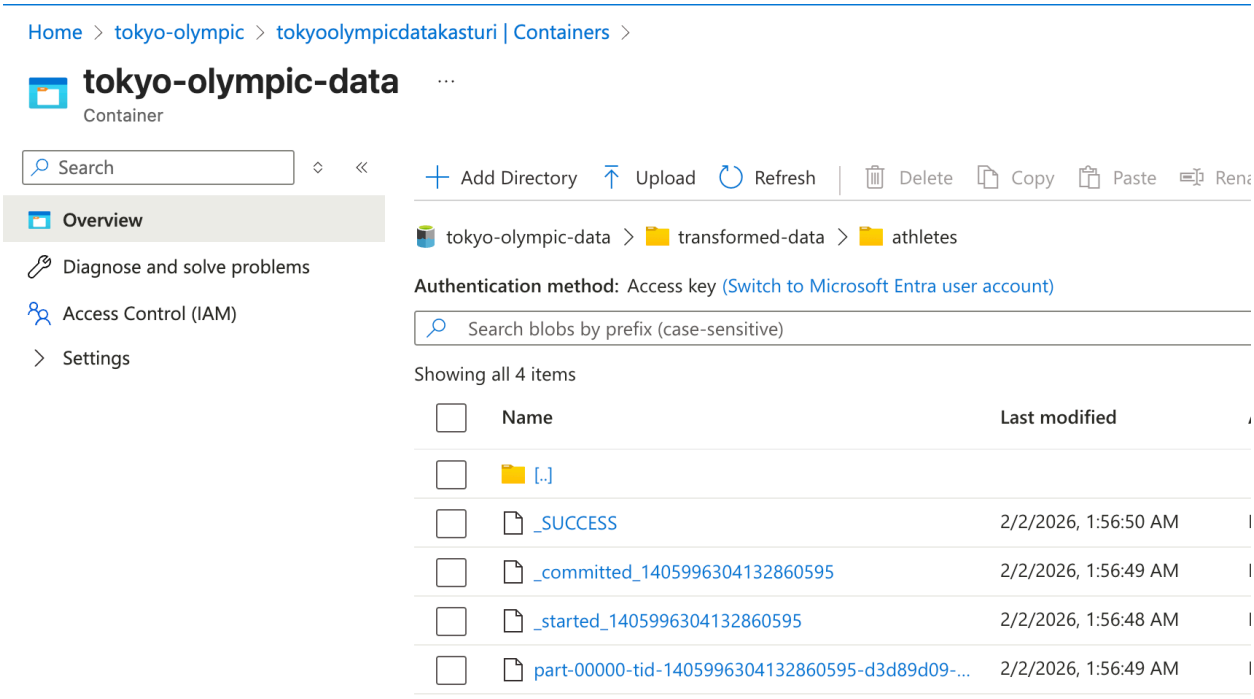
- High-quality, analytics-ready datasets stored in ADLS Gen2

Code Link


Microsoft Azure
databricks
Search data, notebooks, recents, and more...
⌘ + P
New
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Runs
Data Ingestion
Tokyo Olympic Transformation
File Edit View Run Help Python Tabs: ON
Last edit was 17 hours ago
Run all
01:38 AM (2s)
6
athletes.show()
(1) Spark Jobs
+--------------------+--------------------+--------------------+
| 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
ABALO Luc	France	Handball
ABAROA Cesar	Chile	Rowing
ABASS Abobakr	Sudan	Swimming
ABBASALI Hamideh	Islamic Republic ...	Karate
ABBASOV Islam	Azerbaijan	Wrestling
ABBINGH Lois	Netherlands	Handball
ABBOT Emily	Australia	Rhythmic Gymnastics
ABBOTT Monica	United States of ...	Baseball/Softball
ABDALLA Abubaker ...	Qatar	Athletics
ABDALLA Maryam	Egypt	Artistic Swimming
ABDALLAH Shahd	Egypt	Artistic Swimming
ABDALRASOOL Mohamed	Sudan	Judo
ABDEL LATIF Radwa	Egypt	Shooting


01:56 AM (5s)
13
athletes.repartition(1).write.mode("overwrite").option("header",'true').csv("abfss://
tokyo-olympic-data@tokyoolympicdatakasturi.dfs.core.windows.net/transformed-data/athletes")
(2) Spark Jobs
01:57 AM (8s)
14
Python
coaches.repartition(1).write.mode("overwrite").option("header","true").csv("abfss://
tokyo-olympic-data@tokyoolympicdatakasturi.dfs.core.windows.net/transformed-data/coaches")
entriesgender.repartition(1).write.mode("overwrite").option("header","true").csv("abfss://
tokyo-olympic-data@tokyoolympicdatakasturi.dfs.core.windows.net/transformed-data/entriesgender")
medals.repartition(1).write.mode("overwrite").option("header","true").csv("abfss://
tokyo-olympic-data@tokyoolympicdatakasturi.dfs.core.windows.net/transformed-data/medals")
teams.repartition(1).write.mode("overwrite").option("header","true").csv("abfss://tokyo-olympic-data@tokyoolympicdatakasturi.
dfs.core.windows.net/transformed-data/teams")
(8) Spark Jobs





7. Azure Synapse Analytics Integration

Azure Synapse Analytics was used as the **query and semantic layer** on top of the data lake.

Synapse Workspace

- Workspace Name: `tokyoolympicsakasturi`
- SQL Pool: **Built-in Serverless SQL Pool**

Lake Database

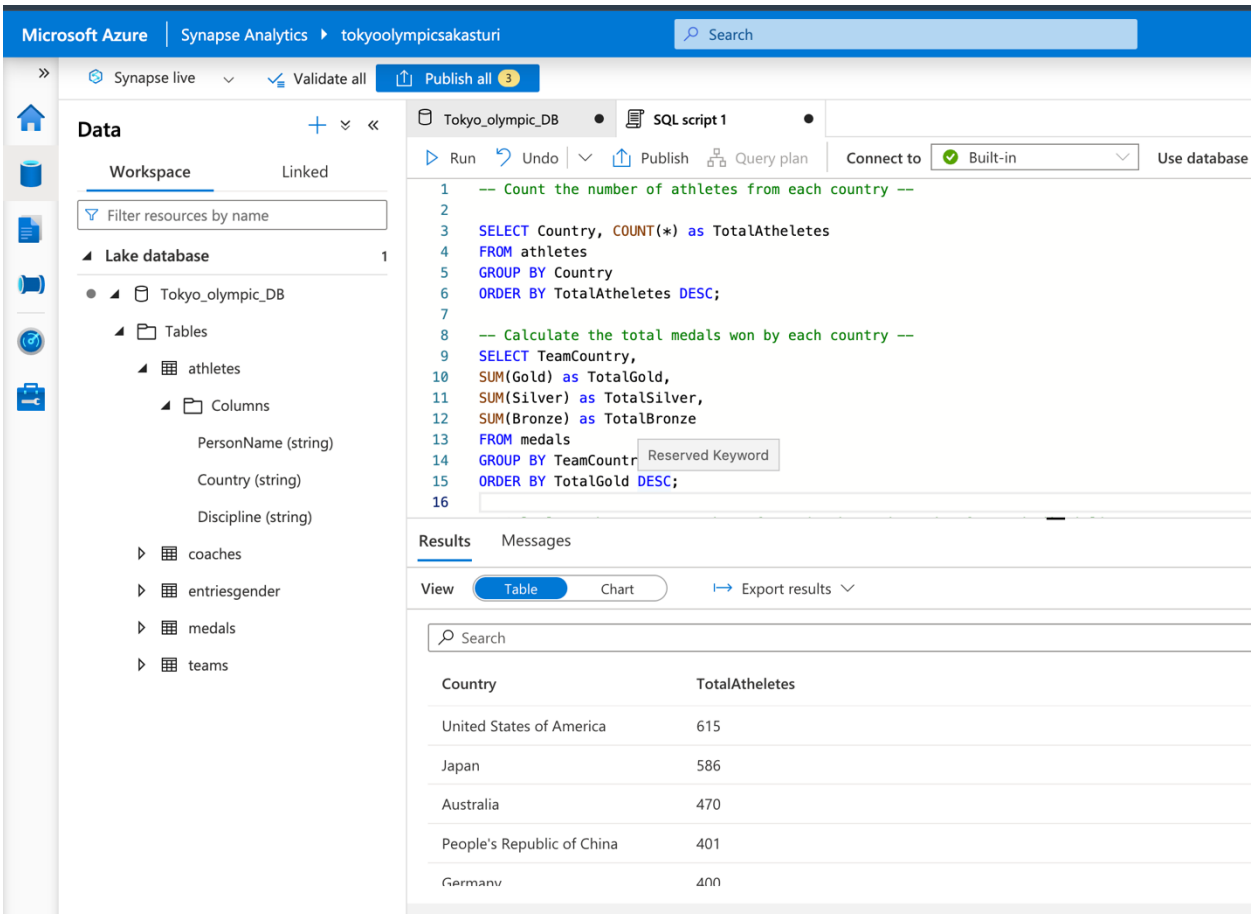
- Created a Lake Database: `Tokyo_olympic_DB`
- Tables mapped to curated Parquet files in ADLS Gen2

Benefits of Serverless SQL

- No infrastructure management
- Pay-per-query pricing
- Direct querying of data lake files

Outcome

- Structured tables/views available for BI consumption
- SQL-based access to lake data



---

8. Power BI Integration

Power BI was used as the visualization and reporting layer.

Connection Method

- Connector: **Azure Synapse Analytics**
- Server: `tokyoolympicsakasturi.sql.azuresynapse.net`
- Database: `master`
- Authentication: **Microsoft / Organizational Account**

Power BI connects to the **serverless SQL pool**, which exposes lake data through tables and views.

Data Modeling

- Imported curated tables
- Built relationships (star schema where applicable)
- Created calculated measures and KPIs

Dashboards Created

- Medal count by country
- Athletes distribution by sport
- Gender analysis
- Top-performing countries

Settings for Tokyo-Olympic-Dashboard

View semantic model

Last refresh succeeded: 2/2/2026, 5:55:34 PM

Refresh history

Semantic model description

Describe the contents of this semantic model.

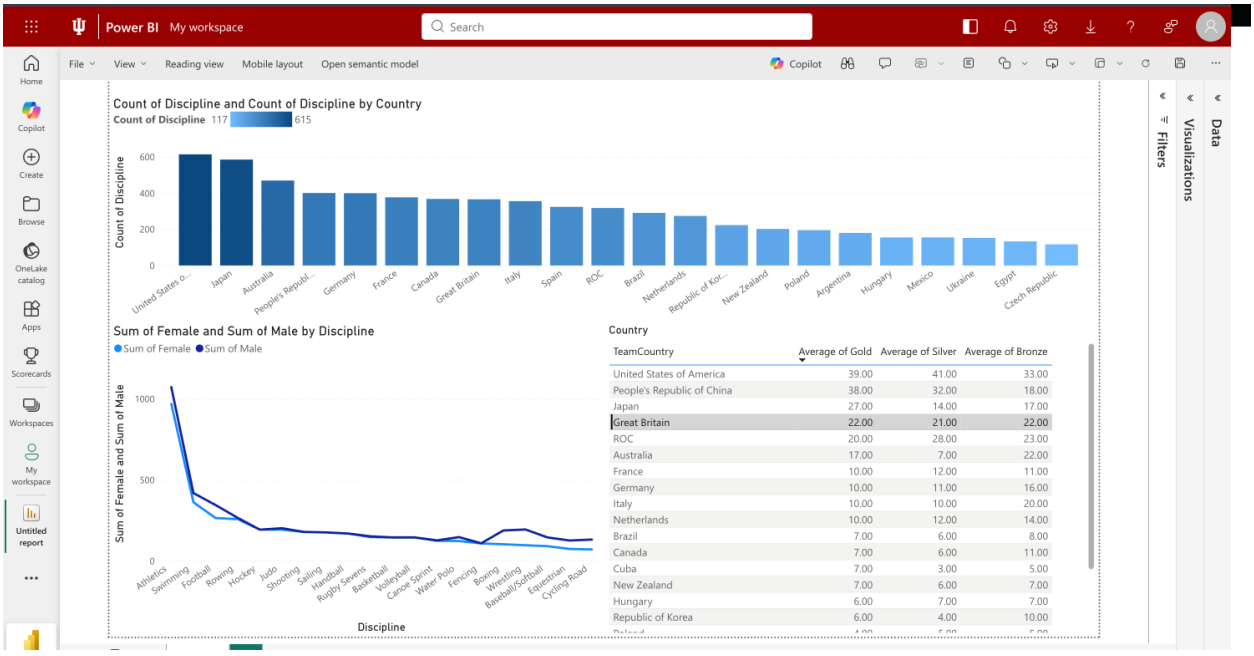
500 characters left

Apply Discard

Gateway and cloud connections

Data source credentials

Synapse Managed by connection Synapse kdisale Show in lineage view



9. Security & Access Management

- Azure AD–based authentication
- Role-based access control (RBAC)
- Storage Blob Data Reader for data access
- Synapse SQL permissions for querying

Secrets and credentials were managed securely and not hardcoded in production-ready setups.

---

10. Tools & Technologies Used

- Azure Data Factory
- Azure Data Lake Storage Gen2
- Azure Databricks (Apache Spark)
- Azure Synapse Analytics (Serverless SQL)
- Power BI
- GitHub
- Python (PySpark)
- SQL

---

11. Key Learnings

- End-to-end data engineering pipeline design
- Lakehouse architecture implementation
- Spark-based data transformations
- Serverless analytics using Synapse SQL
- Power BI integration with Azure data platforms
- Azure security and authentication best practices

---

12. Conclusion

This project successfully demonstrates a **real-world Azure Data Engineering workflow**, from raw data ingestion to business intelligence reporting. It highlights best practices in data lake design, scalable data processing, serverless analytics, and dashboard development. The architecture is modular, scalable, and aligned with industry standards, making it suitable for both learning and portfolio demonstration purposes.

---

13. References

- Part 1 Tutorial: https://www.youtube.com/watch?v=IaA9YNlg5hM
- Part 2 Tutorial: https://www.youtube.com/watch?v=nW0ffUW2vw4