

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

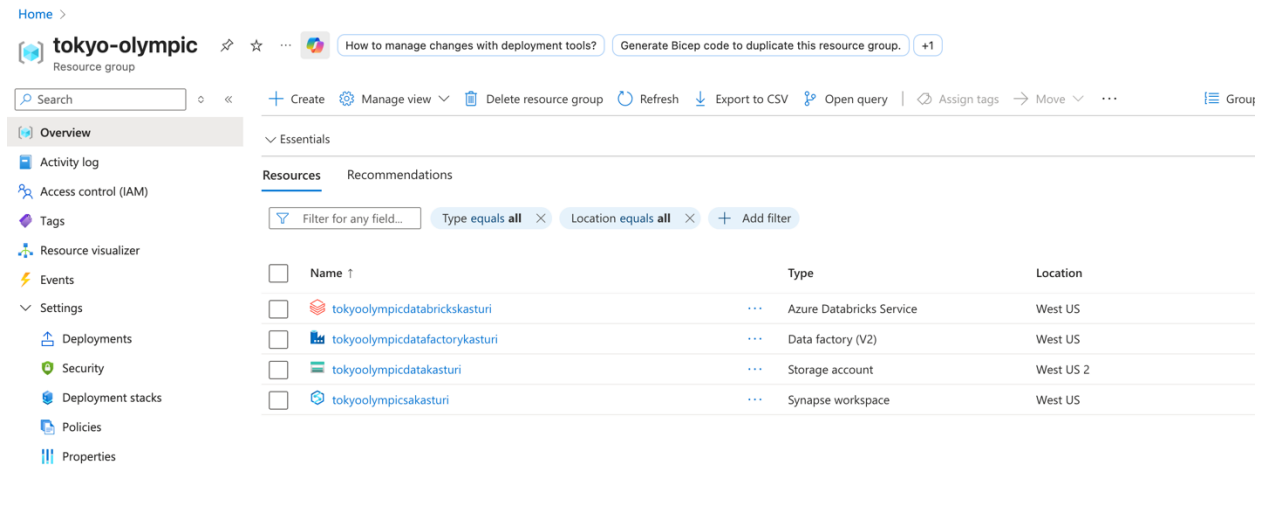
High-level Architecture

```
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

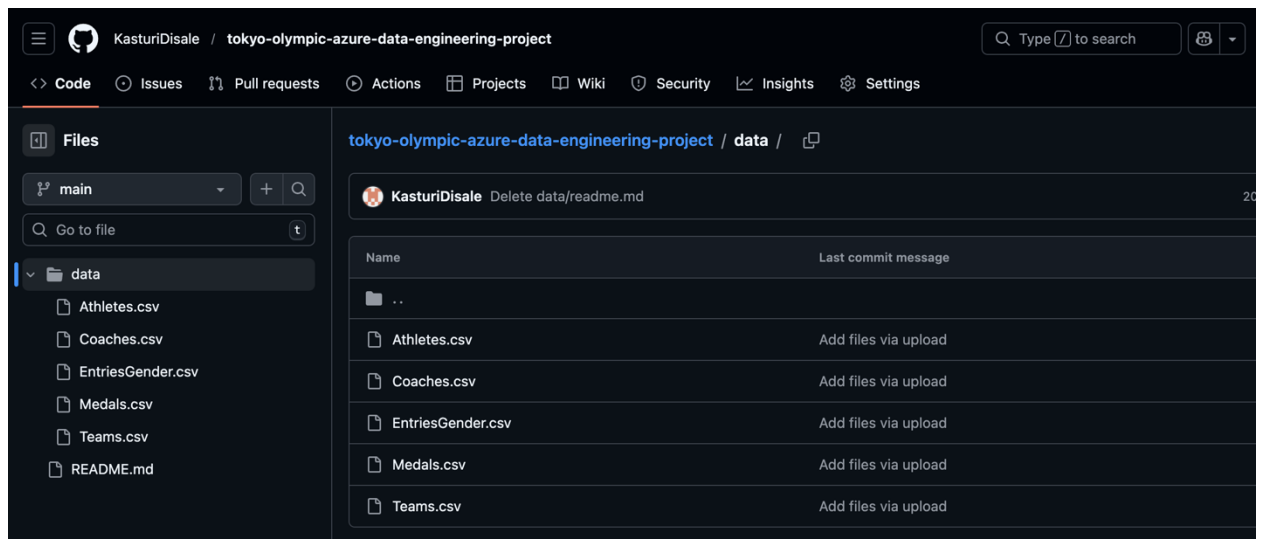
- Separation of **storage** and **compute**
- Lakehouse-style architecture (Raw → Curated)
- Serverless analytics for cost efficiency
- SQL-based semantic layer for BI tools

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

Home > tokyo-olympic > tokyoolympicdatakasturi

 **tokyoolympicdatakasturi** | Containers ☆ ...

Storage account

Search

Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Partner solutions
Resource visualizer
Data storage

Containers

+ Add container ↑ Upload ↺ Refresh | 🗑 Delete 🔒 Char

Search containers by prefix

Showing all 2 items

<input type="checkbox"/>	Name
<input type="checkbox"/>	\$logs
<input checked="" type="checkbox"/>	tokyo-olympic-data

Container Structure

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

Home > tokyo-olympic > tokyoolympicdatakasturi | Containers >

tokyo-olympic-data

Container

Search

⌵ ⌵

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

+ Add Directory

⬆ Upload

🔄 Refresh

🗑 Delete

📄 Copy

📄 Paste

🔍 Re

tokyo-olympic-data

Authentication method: Access key [\(Switch to Microsoft Entra user account\)](#)

Search blobs by prefix (case-sensitive)

Showing all 2 items

<input type="checkbox"/>	Name	Last modified
<input type="checkbox"/>	📁 raw-data	2/1/2026, 7:21:28 PM
<input type="checkbox"/>	📁 transformed-data	2/1/2026, 7:21:40 PM

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

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

Outcome

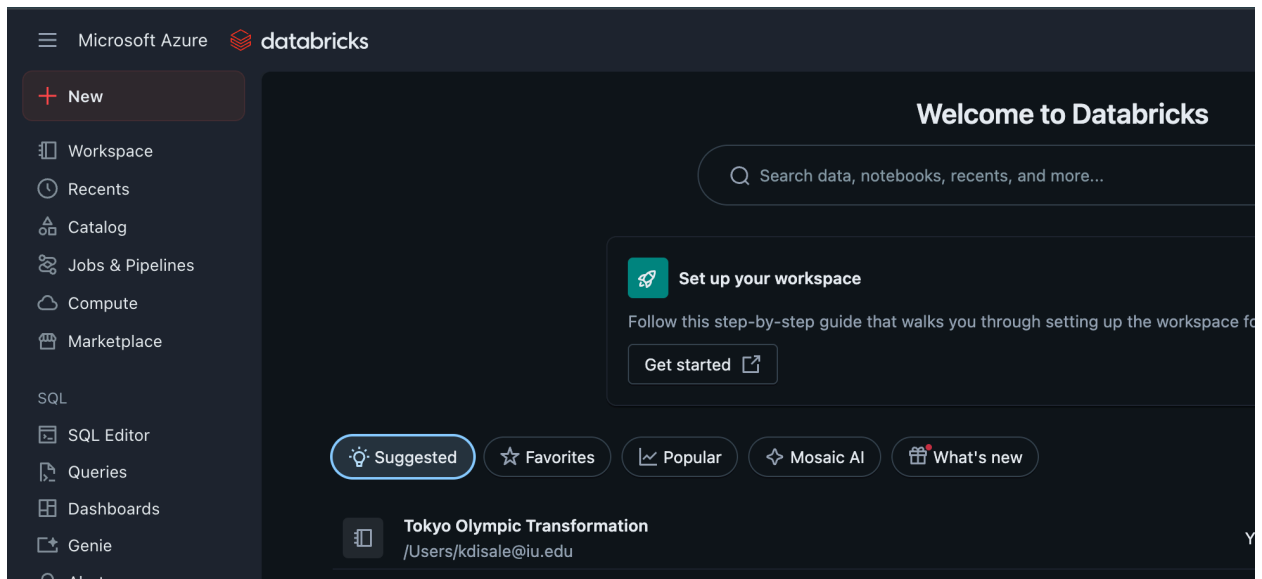
- Raw Tokyo Olympics CSV files successfully landed in ADLS Gen2
- Pipelines were parameterized and reusable

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 x +

File Edit View Run Help Python Tabs: ON ☆ Last edit was 17 hours ago Run all kdisa

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 of Iran	Karate
ABBASOV Islam	Azerbaijan	Wrestling
ABBINGH Lois	Netherlands	Handball
ABBOTT Emily	Australia	Rhythmic Gymnastics
ABBOTT Monica	United States of America	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

Home > tokyo-olympic > tokyoolympicdatakasturi | Containers >

tokyo-olympic-data

Container

Search

◊

«

+

Add Directory

↑

Upload

↺

Refresh

🗑️

Delete

📄

Copy

📄

Paste

🏷️

Rename

🔒

Acquire lease

🔗

Br...

📁 Overview

🔧 Diagnose and solve problems

👤 Access Control (IAM)

> Settings

tokyo-olympic-data > transformed-data

Authentication method: Access key ([Switch to Microsoft Entra user account](#))

🔍 Search blobs by prefix (case-sensitive)

Showing all 5 items

<input type="checkbox"/>	Name	Last modified	Access tier
<input type="checkbox"/>	📁 [..]		
<input type="checkbox"/>	📁 athletes	2/2/2026, 1:56:48 AM	
<input type="checkbox"/>	📁 coaches	2/2/2026, 1:57:30 AM	
<input type="checkbox"/>	📁 entriesgender	2/2/2026, 1:57:32 AM	
<input type="checkbox"/>	📁 medals	2/2/2026, 1:57:34 AM	
<input type="checkbox"/>	📁 teams	2/2/2026, 1:57:36 AM	

Home > tokyo-olympic > tokyoolympicdatakasturi | Containers >

tokyo-olympic-data

Container

Search

◊

«

+

Add Directory

↑

Upload

↺

Refresh

🗑️

Delete

📄

Copy

📄

Paste

🏷️

Ren...

📁 Overview

🔧 Diagnose and solve problems

👤 Access Control (IAM)

> Settings

tokyo-olympic-data > transformed-data > athletes

Authentication method: Access key ([Switch to Microsoft Entra user account](#))

🔍 Search blobs by prefix (case-sensitive)

Showing all 4 items

<input type="checkbox"/>	Name	Last modified	
<input type="checkbox"/>	📁 [..]		
<input type="checkbox"/>	📄 _SUCCESS	2/2/2026, 1:56:50 AM	I
<input type="checkbox"/>	📄 _committed_1405996304132860595	2/2/2026, 1:56:49 AM	I
<input type="checkbox"/>	📄 _started_1405996304132860595	2/2/2026, 1:56:48 AM	I
<input type="checkbox"/>	📄 part-00000-tid-1405996304132860595-d3d89d09-...	2/2/2026, 1:56:49 AM	I

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

The screenshot displays the Microsoft Azure Synapse Analytics workspace for 'tokyoolympicsakasturi'. The interface includes a left-hand navigation pane with icons for home, data, workspace, and linked resources. The 'Data' section is expanded, showing a 'Lake database' named 'Tokyo_olympic_DB' with a table 'athletes' and columns 'PersonName (string)', 'Country (string)', and 'Discipline (string)'. The main pane shows a SQL script named 'SQL script 1' with the following code:

```
1 -- Count the number of athletes from each country --
2
3 SELECT Country, COUNT(*) as TotalAtheletes
4 FROM athletes
5 GROUP BY Country
6 ORDER BY TotalAtheletes DESC;
7
8 -- Calculate the total medals won by each country --
9
10 SELECT TeamCountry,
11 SUM(Gold) as TotalGold,
12 SUM(Silver) as TotalSilver,
13 SUM(Bronze) as TotalBronze
14 FROM medals
15 GROUP BY TeamCountry
16 ORDER BY TotalGold DESC;
```

The 'Results' tab is active, showing a table with the following data:

Country	TotalAtheletes
United States of America	615
Japan	586
Australia	470
People's Republic of China	401
Germany	400

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

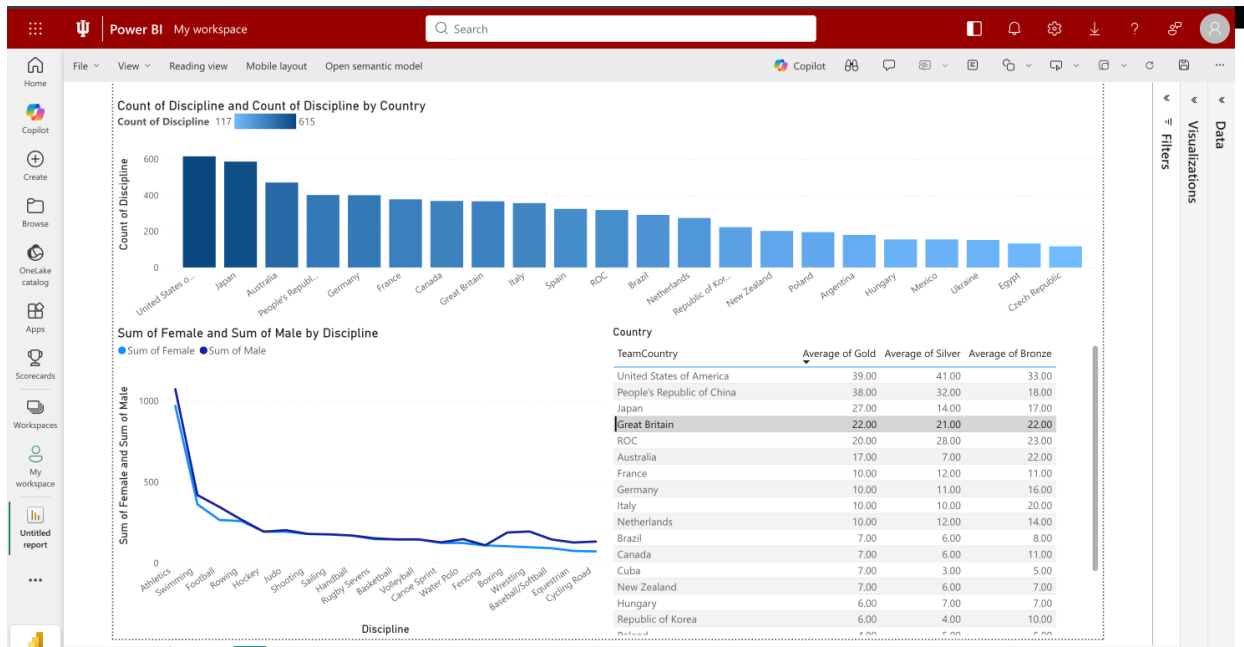
Apply

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>