Azure Data Engineering Project: End-to-End Data Pipeline for Olympic Data

Project Overview

The goal is to build a data pipeline to extract, transform, and load Olympic data for analysis using various Azure services. The data source is a Kaggle dataset containing information about the 2021 Tokyo Olympics, uploaded to a GitHub repository for easier access. The data includes information about athletes, coaches, entries by gender, medals, and teams. The project involves the following steps:

Data Ingestion: Extracting the raw data from the GitHub repository using Azure Data Factory.

Raw Data Storage: Storing the extracted raw data in Azure Data Lake Storage Gen2.

Data Transformation: Using Azure Databricks to perform data cleaning and transformation using Apache Spark.

Transformed Data Storage: Loading the transformed data back into Azure Data Lake Storage Gen2.

Data Analysis: Using Azure Synapse Analytics to analyze the transformed data using SQL queries and visualization tools.

This project was inspired and guided by **Darshil Parmar's YouTube tutorial**, which provided excellent insights into using **Azure Cloud Services** for building data engineering pipelines.

Technologies Used

- **Azure Data Factory**: To build ETL pipelines for extracting and loading data.
- **Azure Databricks**: To perform data transformation using PySpark.
- **Azure Data Lake Storage Gen2**: To store raw and transformed data.
- **Azure Synapse Analytics**: For data analysis and SQL queries.
- **Power BI/Tableau**: To create interactive dashboards and data visualizations.
- **GitHub**: For hosting the dataset files.

Dataset Overview

The dataset used in this project is sourced from Kaggle and contains detailed information about the **Tokyo 2021 Olympics**. It includes data such as:

- Athlete information (names, countries, sports, etc.)
- Medals awarded (Gold, Silver, Bronze)
- Entries by gender and discipline
- Country-specific medal tallies

The data is stored in CSV files and hosted on a **GitHub repository**.

Here is the Github repository link: [ravalikolloju4/tokyo-olympic]

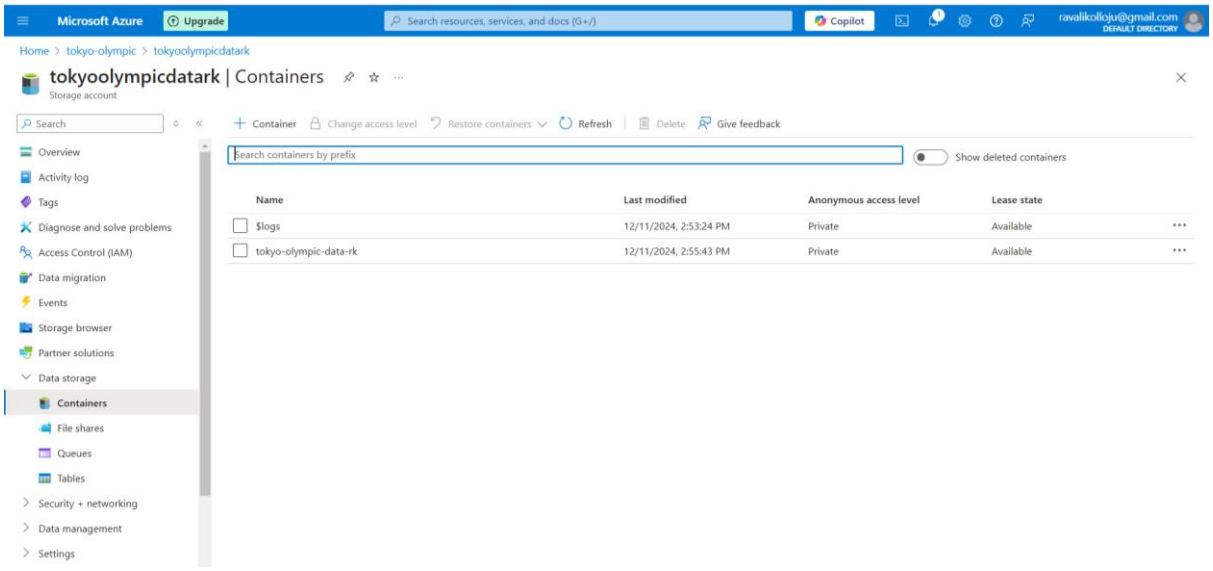
Step-by-Step Guide:

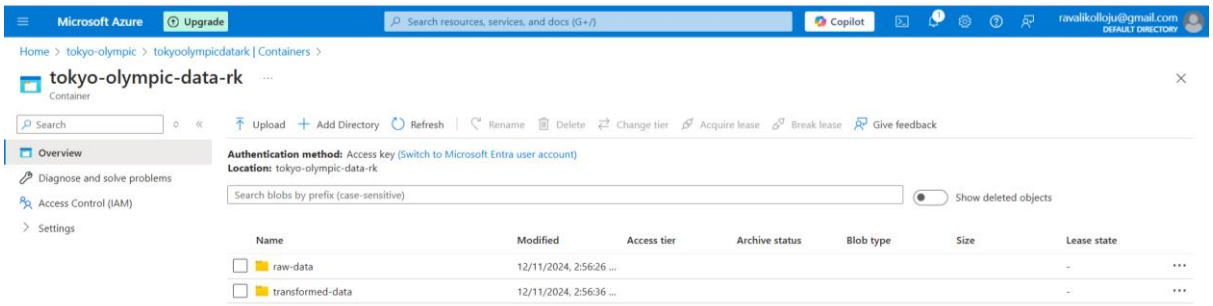
1. Set up Azure Data Lake Storage Gen2:

Create a storage account in Azure portal, naming it uniquely and selecting the nearest region.

Enable hierarchical namespace to store data in a hierarchical format like a local file system.

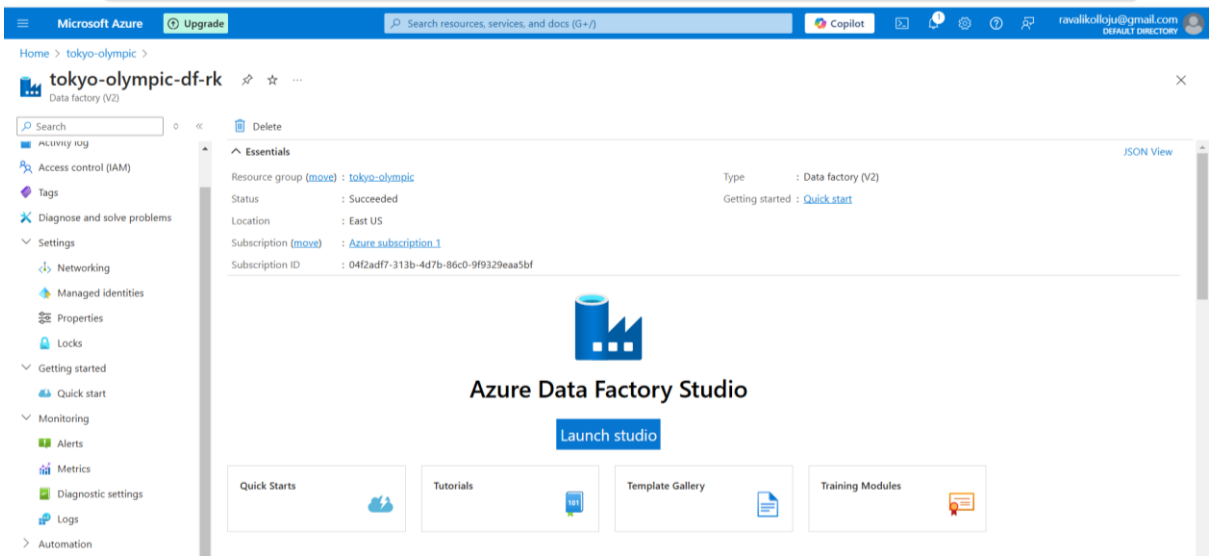
Create two containers within the storage account: "raw data" to store the initial extracted data and "transformed data" for the processed data.





2. Create an Azure Data Factory:

Create a data factory in the Azure portal, specifying the resource group and region. Launch the data factory studio to start building data pipelines.



3. Ingest Data using Azure Data Factory:

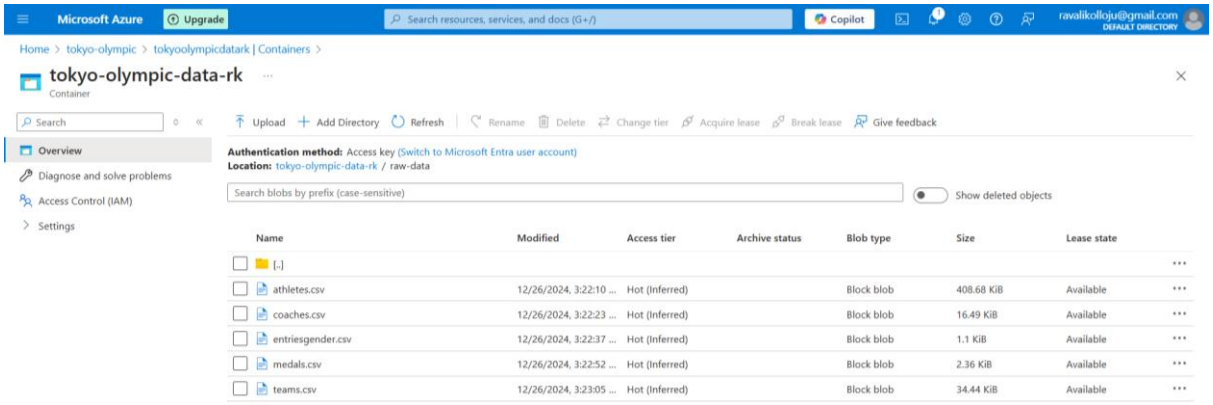
- Create a new pipeline within the data factory and name it appropriately e.g., "data ingestion".
- Add a copy activity to the pipeline to move data from the source to the destination.

Configure the copy activity's source:

- ✓ Create a new linked service of type "HTTP" to connect to the GitHub repository.
- ✓ Provide the raw URL of each CSV file from the repository as the base URL for each linked service.
- ✓ Set authentication type to "Anonymous" as the repository is public.
- ✓ Specify the file format as "Delimited Text" and "CSV"

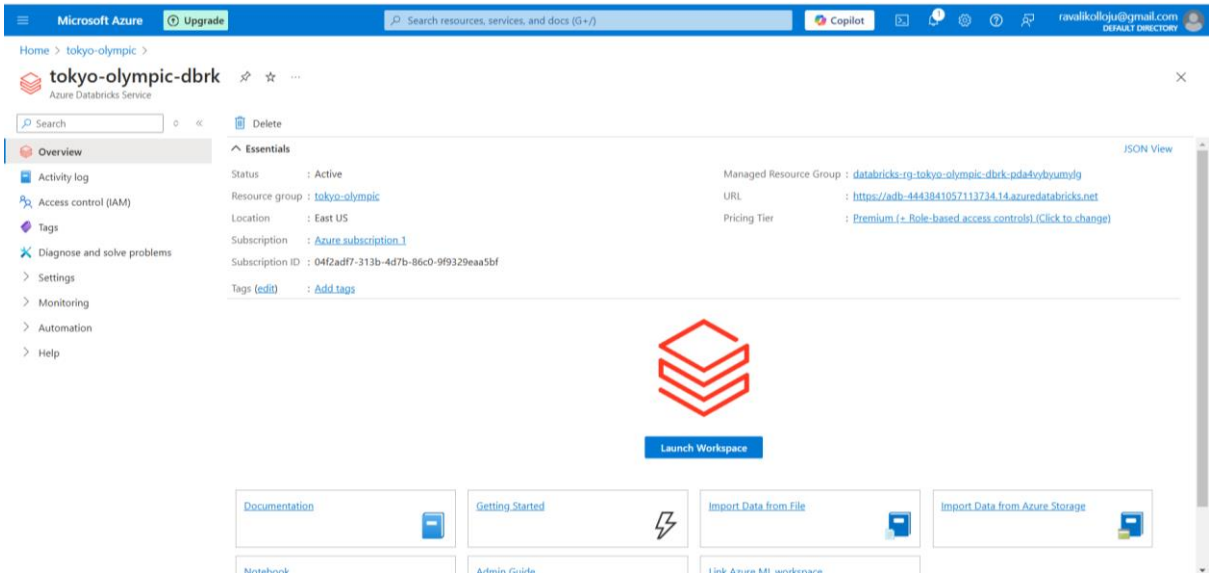
Configure the copy activity's sink:

- ✓ Create a linked service of type "Azure Data Lake Storage Gen2" to connect to the storage account.
- ✓ Select the appropriate storage account and container ("raw data").
- ✓ Specify the file path and name for each file.
- ✓ Validate the pipeline settings and debug to test its functionality.
- ✓ Run the pipeline to ingest all five CSV files into the "raw data" container.



4. Set up Azure Databricks:

- Create a Databricks workspace in the Azure portal, naming it and selecting the desired pricing tier and node type.
- Launch the Databricks workspace to access notebooks and clusters.



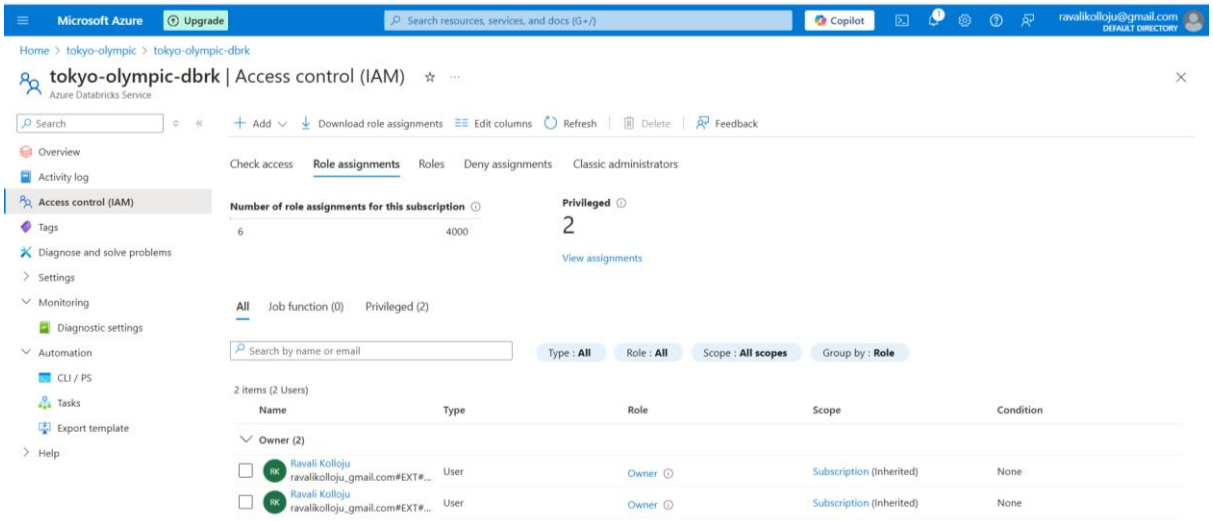
5. Connect Azure Databricks to Azure Data Lake Storage:

Create an app registration in the Azure portal and note the client ID, tenant ID, and secret key generated.

In a Databricks notebook, define a configuration object with the app registration credentials, storage account name, and container name.

Use the dbutils.fs.mount command to mount the "raw data" container to the Databricks file system, providing the configuration object and mount point.

Grant the app registration "Storage Blob Contributor" role access to the storage account for necessary permissions.



6. Transform Data using Azure Databricks:

Create a new notebook in Databricks and attach it to the cluster.

Read the CSV files from the mounted data lake using spark.read.format("csv").option("header", "true").load(), specifying the correct file path.

Analyze the schema of each data frame using printSchema() and identify columns requiring data type conversions.

Use the withColumn() function to create new columns or overwrite existing ones, applying data type conversions using cast() with the desired type.

Perform additional transformations as needed, like filtering, grouping, and aggregation.

7. Load Transformed Data to Azure Data Lake Storage:

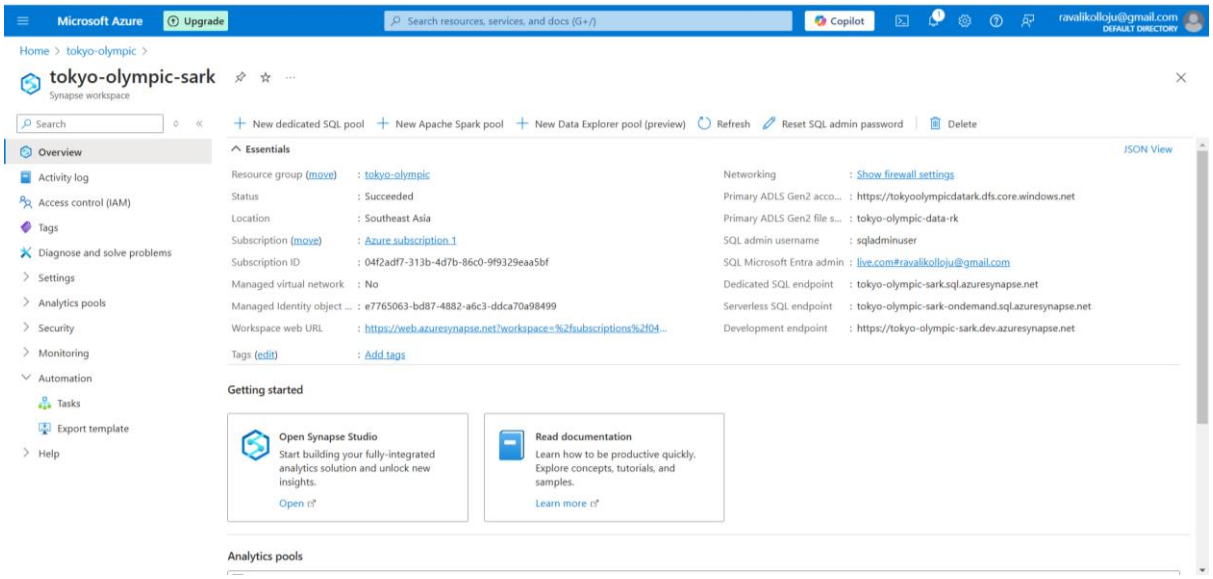
Write the transformed data frames back to the data lake using write.mode("overwrite").option("header", "true").csv(), specifying the "transformed data" container and file path.

Use repartition() to control the number of output files generated.

8. Set up Azure Synapse Analytics:

Create a Synapse workspace in the Azure portal, linking it to the existing storage account.

Launch the Synapse Studio to access its features.



9. Load Data into Azure Synapse Analytics:

Create a new database in the Synapse workspace.

Create tables for each transformed dataset by selecting "From Data Lake".

Specify the linked service, file path, and format for each table.

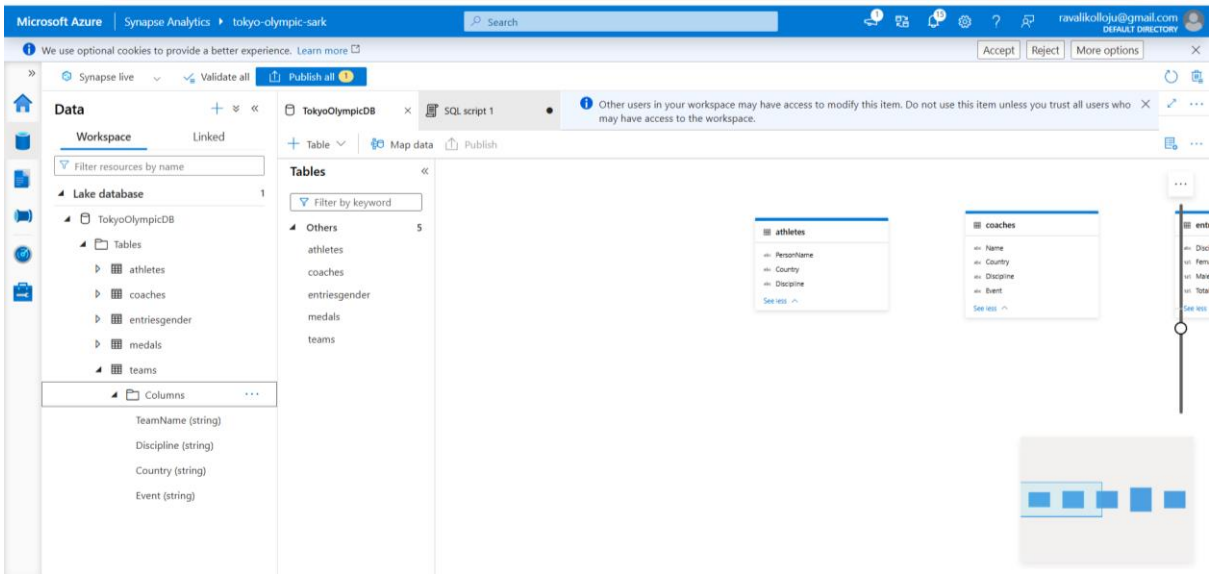
Review the detected schema and data types, making adjustments if necessary.

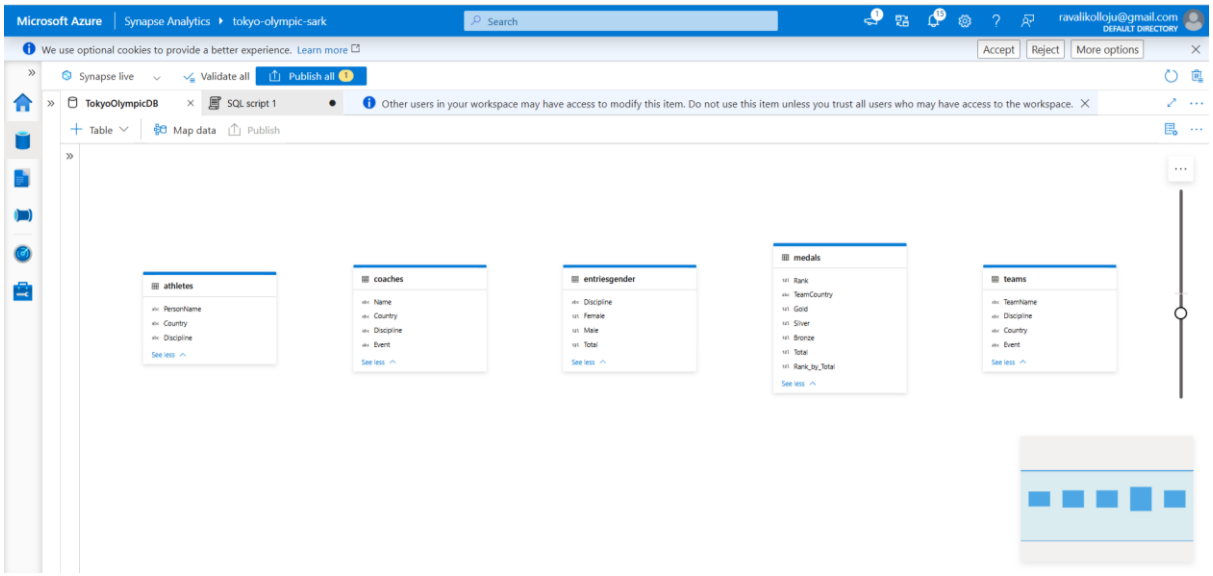
Publish the tables to the Synapse database.

10. Analyze Data using Azure Synapse Analytics:

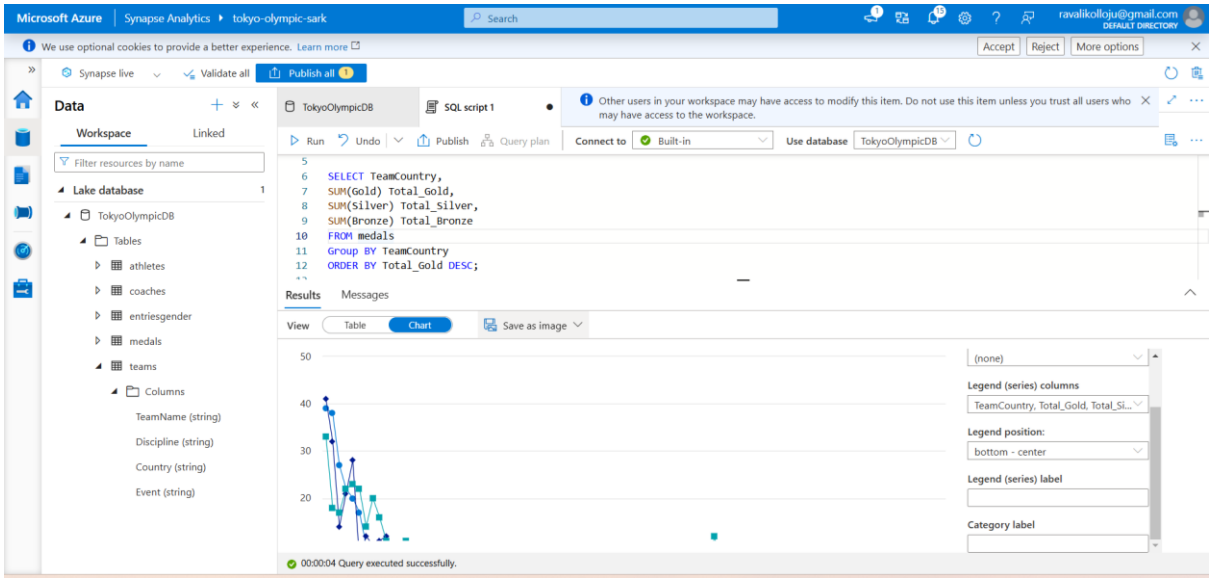
Use SQL queries to perform data analysis on the loaded tables.

Utilize built-in visualization tools in Synapse Studio to generate charts and graphs.



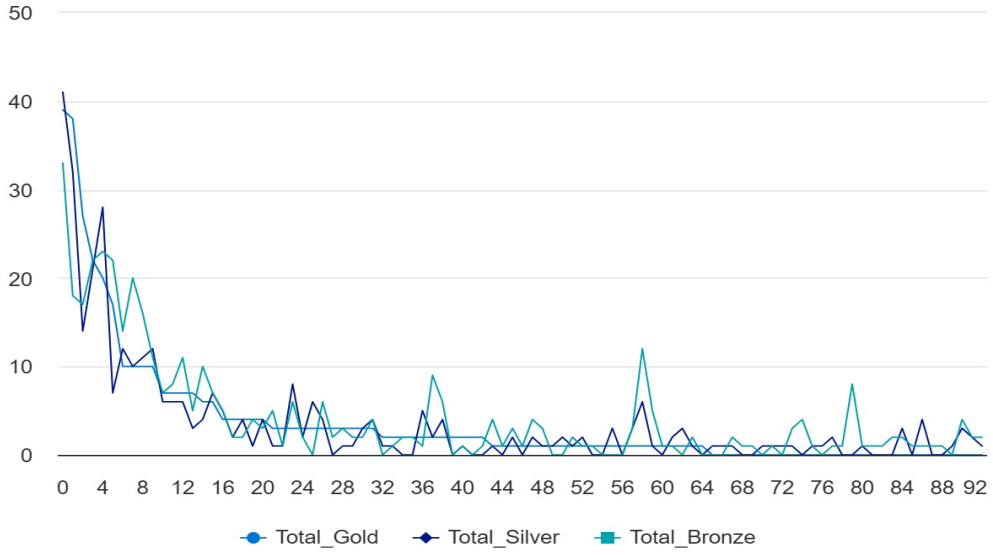


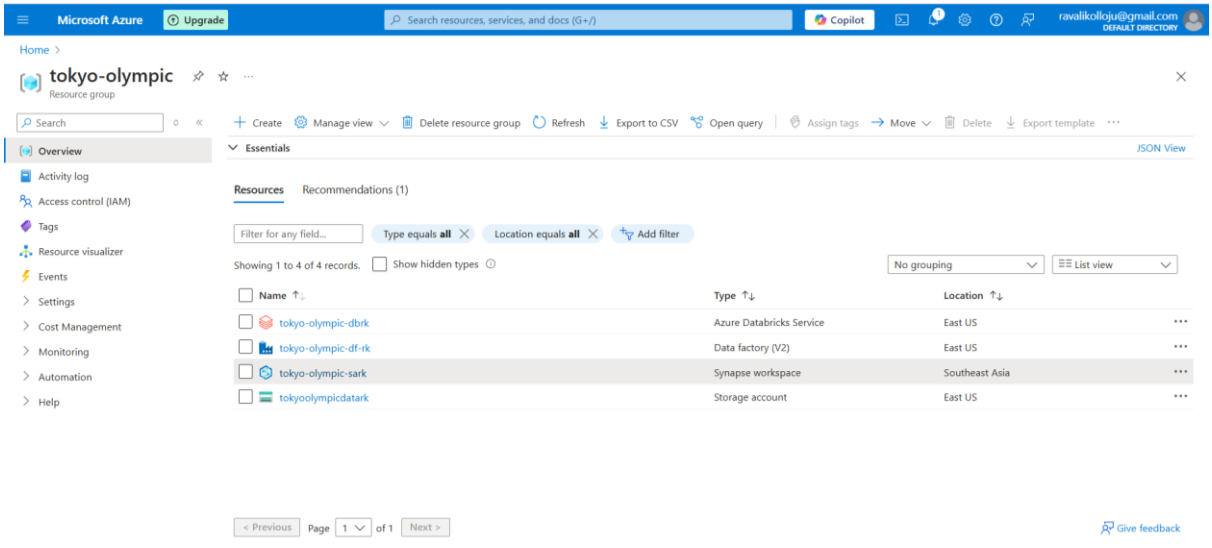
Azure Synapse SQL Query Results



Data Visualization

- **Azure Synapse Analytics** built-in visualizations were used to create graphs and charts based on the analysis.





Learning Outcomes

By completing this project, I gained hands-on experience with the following:

- **ETL Pipeline Design**: Understanding the core concepts of **Extract, Transform, Load** processes.
- **Data Factory Pipelines**: Gained proficiency in using **Azure Data Factory** to connect to data sources and load data to storage locations.
- **Data Storage Solutions**: Learned how to use **Azure Data Lake Storage Gen2** to manage raw and transformed data.
- **Data Transformation with PySpark**: Implemented data transformation tasks like schema enforcement, type conversions, and calculated fields using **Azure Databricks**.
- **SQL Analysis**: Analyzed the data using **SQL** queries in **Azure Synapse Analytics**.
- **Data Visualization**: Created effective data visualizations using **Synapse Analytics**, **Power BI**, and **Tableau** to present insights.

Conclusion

This project allowed me to explore and implement several critical aspects of **Azure Data Engineering**. By building an end-to-end pipeline, I gained hands-on experience with the essential tools and services used in data engineering workflows. I am now more confident in working with cloud-based data engineering solutions and transforming raw data into meaningful insights.