Summer Olympics 2024 – Swimming Sports Database Research

Izhary Pauline Rodriguez Fortun

Introduction

This project aims to organize and manage specific data related to various water sports events in the Olympics Summer 2024. The database captures essential information about athletes, teams and disciplines. It also tracks medals awarded to teams and their respective achievements.

I created a relational model with entities Athlete, Team, Event, Venue, Country, Discipline and Medal to model the reality of the Olympics Summer 2024 competition. The sports covered in this database are swimming, artistic swimming, diving, water polo and surfing. The database uses foreign keys to maintain relationships between tables to ensure the data remains consistent.Stored procedures were utilized to retrieve how many medals an athlete has won as well as medal counts with a specific criteria. Triggers were used to automate updates and ensure data accuracy.

Python scripts were used to populate the data, utilizing pandas– A Python Data Analysis Library and facilitating CRUD operations (Create, Read, Update, Delete) on existing data in the database. The database supports various SQL queries to enable users to filter results based on disciplines and date ranges for medal achievements for the purpose of analyzing performance in the competitions.

Database Design

Entities

The focus of the database design is to model the Paris 2024 Olympic Summer Games for watersports as accurately as possible, specifically covering events like swimming, artistic swimming, diving, water polo and surfing.

The entities represent key components of the Olympic games, corresponding to real-world objects or participants in the Olympics. Athlete are central to the Olympic Games, this entity represents the individual competitors in various water sports disciplines. Details provided of the athlete include name, gender, country, discipline(s) and whether they belong to a team. This helps track the performance of individual athletes and their associated teams. Team is essential to team-based sports like water polo and artistic swimming. It includes attributes such as team name, gender, country and discipline(s) the team is competing in. Some athletes participate on their own so their Team_Code can be NULL. Event represents the specific water sport competitions that the athletes or teams participate in. Each Event has a unique Event_ID as its primary key for identification. It has attributes such as its tag, sport and venue to link the events

to locations and the Discipline_Name. Each Venue can host one or more sports, by storing the venue name, sports played and the dates during its use, we can track where and when competitions are held. Medal tracks the awards given to athletes or teams. It has medal type (i.e. gold, silver, bronze), date achieved, athlete code and event tag to help record the outcome of an event. This allows people to see who won what in which events. As every athlete or team represents a country, Country has Country_Code and Country_Name so we can easily filter athletes or teams by their nationality. Discipline is the type of sport athletes will be competing in, it will have a Discipline_Code as the primary key and a Discipline_Name.

Assumptions

1. Some events are individual such as diving and surfing while other events involve teams (e.g. water polo, artistic swimming). Therefore, athletes may or may not have a Team_Code depending on the type of event.
2. Each event is unique and can be identified by its Event_ID. The tag is based on the sport and event type (e.g. marathon-swimming) and is not used as the key as there could be multiples of an event with the same tag.
3. Every medal must be awarded to a team. Medals are associated with an event and a date they are achieved, but they can be rewarded to a team. Individual athletes and their medal wins are excluded from this database.
4. Venues may be used for multiple sports. For example, the same aquatics center could host both water polo and artistic swimming events. The assumption is that a venue can host several events but each event is associated with one venue at a time.
5. Every athlete and team must represent a country due to the nature of the Olympics. No individual or team can compete without representing a country.
6. An athlete or team can participate in one or more than one discipline, but every discipline must have at least two athletes or teams competing in it.
7. The project focuses on the main watersports categories including artistic swimming, swimming, marathon swimming, diving, water polo and surfing. Canoe sports have been intentionally excluded to narrow the scope because they have a distinct set of rules, athletes and venues.

Entity Tables and Relationships

Entity	Key(s)	Attributes
Athlete	Athlete_Code (PK), Country_Code (FK to Country), Team_Code(FK to Team, NULL if Individual)	First_Name, Last_Name, Gender, Discipline

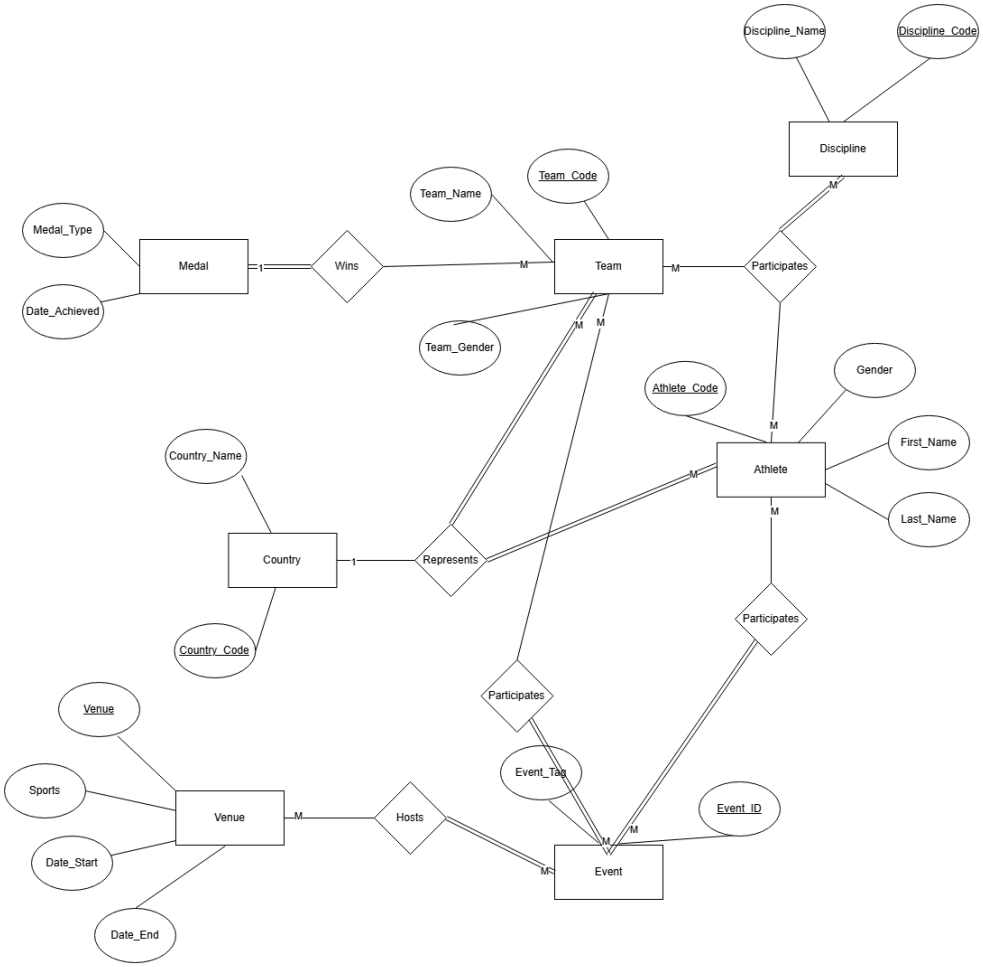
Team	Team_Code (PK), Country_Code (FK to Country)	Team_Name, Team_Gender, Discipline, Event_Tag
Event	Event_ID, Venue (FK to Venue)	Event_Tag, Event_Name, Discipline_Name
Venue	Venue (PK)	Sports (list), Date_Start, Date_End
Medal	Athlete_Code (FK to Athlete), Event_ID (FK to Event)	Medal_Type, Date_Achieved
Country	Country_Code (PK)	Country_Name
Discipline	Discipline_Code (PK)	Discipline_Name

Relationship	Between Sets	Other Attributes
Participates	Athletes & Event	Athlete_Code, Event_ID
Participates	Team & Event	Team_Code
Hosts	Venue & Event	Venue, Event_ID
Wins	Team & Medal	Team_Code, Event_ID, Medal_Type
Represents	Country & Athlete	Country_Code, Athlete_Code
Represents	Country & Team	Country_Code, Team_Code
Participates	Athletes & Discipline	Athlete_Code, Discipline_Code
Participates	Team & Discipline	Team_Code, Discipline_Code

Relationship	Cardinality Constraint
Participates	Athlete to Event: Many-to-many
Participates	Team to Event: Many-to-many
Hosts	Event to Venue: Many-to-many
Wins	Team to Medal: One-to-many
Represents	Country to Athlete: One-to-many

Represents	Country to Team: One-to-many
Participates	Athlete to Discipline: Many-to-many
Participates	Team to Discipline: Many-to-many

Relationship	Participation Constraint
Participates	Athlete to Event: Partial, Total
Participates	Team to Event: Partial, Total
Hosts	Event to Venue: Total, Partial
Wins	Team to Medal: Partial, Total
Represents	Country to Athlete: Partial, Total
Represents	Country to Team: Partial, Total
Participates	Athlete to Discipline: Partial, Total
Participates	Team to Discipline: Partial, Total



The relationships model how entities interact with one another in the Olympics. Athletes and Teams participate in Events. Since one athlete or team can participate in many events, and many athletes or teams can participate in one event. Venues can host multiple events but each event can only be held in one venue at a time so there is a one to many relationship. Teams win a Medal in an Event, it has a one to many relationship as one athlete or team can win many medals. Each Athlete or Team is associated with one country but a country can represent multiple athletes or teams hence why their relationship is one to many. One Athlete or Team can participate in multiple disciplines and one discipline can have multiple teams.

The constraints reflect the reality of the Olympics, where not all athletes or teams that participate in all events and now all venues are used for all sports. Not every athlete or team must participate in every discipline but every discipline must have at least one athlete competing in it. It also shows that medals are only awarded to teams and every participant must belong to a country.

ER to Relational Schema

Athlete(Athlete_Code (PK), First_Name, Last_Name, Gender, Country_Code (FK), Team_Code (FK), Discipline_Code (FK))

Athlete_Code (PK)	First_Name	Last_Name	Gender	Country_Code (FK to Country)	Team_Code (FK to Team, NULL if individual)	Discipline_Code (FK to Discipline)

Team(Team_Code (PK), Team_Name, Team_Gender, Event_Tag, Country_Code (FK), Discipline_Code (FK))

Team_Code (PK)	Team_Name	Team_Gender	Event_Tag	Country_Code (FK to Country)	Discipline_Code (FK to Discipline)

Event(Event_ID (PK), Event_Tag, Event_Name, Discipline_Code (FK), Venue (FK))

Event_ID (PK)	Event_Tag	Event_Name	Discipline_Code (FK to Discipline)	Venue (FK to Venue)

Venue(Venue (PK), Sports, Date_Start, Date_End)

Venue (PK)	Sports	Date_Start	Date_End

Refined to

Venue(Venue (PK), Date_Start, Date_End)

Venue (PK)	Date_Start	Date_End

Created Venue_Sport

Venue_Sport(Venue (FK), Discipline_Code (FK))

Venue (FK)	Discipline_Code (FK to Discipline)

Medal(Athlete_Code (FK), Event_ID (PK), Medal_Type, Date_Achieved)

Athlete_Code (FK to	Event_ID (PK) (FK to	Medal_Type	Date_Achieved

Athlete)	Event)		

Country(Country_Code (PK), Country_Name)

Country_Code (PK)	Country_Name

Discipline(Discipline_Code (PK), Discipline_Name)

Discipline_Code (PK)	Discipline_Name

Initially, all the tables are in 1NF in which all the primary keys (PK) and foreign keys (FK) constraints are indicated. Partial dependency is when a non-key attribute depends on only part of a composite key. In 2NF, all partial dependencies must be removed. In 3NF, all transitive dependencies – where non-key attributes depend on other non-key attributes, must be removed. The current table has a list of sports, but storing lists is not normalised so a separate table is created: Venue_Sport containing attributes: Venue which is the primary key of Venue and Discipline_Code which is the primary key of Discipline. Because all of the tables have no partial or transitive dependencies, there is no need for further refinement.

Data Description

Athlete

Attribute	Data Type	Description	Constraints
Athlete_Code	VARCHAR(50)	Unique identifier for each athlete	PK, NOT NULL, AUTO_INCREMENT
First_Name	VARCHAR(50)	First name of the athlete	NOT NULL
Last_Name	VARCHAR(50)	Last name of the athlete	NOT NULL
Gender	VARCHAR(50)	Gender of the athlete	NOT NULL
Country_Code	CHAR(3)	Country code where the athlete is from	FK to Country, NOT NULL
Team_Code	INT	Code of the team the athlete belongs to (NULL if individual)	FK to Team, NULLABLE
Discipline_Code	CHAR(5)	Code for the discipline the athlete competes in	FK to Discipline, NOT NULL

Team

Attribute	Data Type	Description	Constraints
Team_Code	VARCHAR(50)	Unique identifier for each team	PK, NOT NULL, AUTO_INCREMENT
Team_Name	VARCHAR(50)	Name of the team	NOT NULL
Team_Gender	CHAR(1)	Gender of the team (M/F/Mixed)	NOT NULL, CHECK (Team_Gender = 'M' or 'F' or 'X' or 'O')
Country_Code	CHAR(3)	Country code for the team	FK to Country, NOT NULL
Discipline_Code	CHAR(5)	Discipline code for the sport the team competes in	FK to Discipline, NOT NULL
Event_Tag	VARCHAR(50)	Tag associated with the event the team competes in	FK to Event, NOT NULL

Event

Attribute	Data Type	Description	Constraints
Event_ID	VARCHAR(50)	Unique identifier for each event	PK, NOT NULL
Event_Tag	VARCHAR(50)	Event identifier	NOT NULL
Event_Name	VARCHAR(100)	Name of the event (e.g., men's team, mixed team)	NOT NULL
Discipline_Code	CHAR(5)	Code for the discipline of the event	FK to Discipline, NOT NULL
Venue	VARCHAR(100)	Name of the venue where the event takes place	FK to Venue, NOT NULL

Venue

Attribute	Data Type	Description	Constraints
Venue	VARCHAR(50)	Unique name of the venue	PK, NOT NULL

Date_Start	DATE	Start date of the venue usage	NOT NULL
Date_End	DATE	End date of the venue usage	NOT NULL

Medal

Attribute	Data Type	Description	Constraints
Athlete_Code	VARCHAR(50)	Code of the athlete who won the medal	FK to Athlete, NOT NULL
Event_ID	VARCHAR(50)	Event where the medal was won	FK to Event, NOT NULL
Medal_Type	VARCHAR(10)	Type of medal (Gold, Silver, Bronze)	NOT NULL, CHECK (Medal_Type IN ('Gold', 'Silver', 'Bronze'))
Date_Achieved	DATE	Date when the medal was achieved	NOT NULL

Country

Attribute	Data Type	Description	Constraints
Country_Code	CHAR(3)	ISO 3166-1 Alpha-3 code of the country	PK, NOT NULL
Country_Name	VARCHAR(100)	Name of the country	NOT NULL

Discipline

Attribute	Data Type	Description	Constraints
Discipline_Code	VARCHAR(50)	Code for the discipline (e.g., DIV for diving)	PK, NOT NULL
Discipline_Name	VARCHAR(50)	Full name of the discipline (e.g., Diving)	NOT NULL

Venue_Sport

Attribute	Data Type	Description	Constraints
Venue	VARCHAR(100)	Name of the venue	FK to Venue, NOT

			NULL
Discipline_Code	VARCHAR(50)	Code for the discipline held at the venue	FK to Discipline, NOT NULL

Business Rules and Assumptions

1. Athletes may compete in only one discipline but can be part of a team or complete as individuals.
2. Athletes are categorized by their gender, usually male, female or other.
3. Teams are categorized by their gender: male (M), female (F) or mixed (M) or other ('O')
4. An event is tied to a specific venue and represents a single occurrence such as "men's team" or "individual"
5. Medals are awarded to athletes for specific events and can only be Gold, Silver or Bronze.

Data Types and Other Actions

1. VARCHAR is used for fields like Discipline_Code, Team_Code etc. to allow for variable-length strings to accommodate various names and code lengths without wasting storage.
2. DATE and DATETIME is used for fields like Date_Achieved to store the date for contexts such as medal achievements and event scheduling.
3. CHECK is used for Medal_Type to enforce certain values because there are only three types of medals available to win.
4. ON DELETE CASCADE is an integrity constraint used to automatically delete records from child tables when the corresponding record in the parent table is deleted. For example in the Event table having foreign key relationships with the Medal table (i.e. Event_ID) and with the Team table (i.e.Team_Code)
5. ON DELETE SET NULL is another integrity constraint that automatically sets the foreign key field in the child table to NULL when the corresponding record in the parent table is deleted instead of deleting it completely.

Implementation

Database Implementation and Sample Data

I used MySQL in a WSL (Windows Subsystem for LINUX) environment to set up my database. To create the necessary tables for the Watersports Olympics 2024 database, I wrote a create_tables.sql script containing all the commands to define the schema. I ran this script using

SOURCE create_tables.sql. My sample data was sourced from a dataset on Kaggle: Paris 2024 Olympic Summer Games where I retrieved the following CSV files: athletes, medals, teams, events and venues. I created my own discipline.csv, country.csv and venue_sport.csv.
To filter and clean the data from CSV files, I used Pandas in Python. I filtered by categories I required from the dataset and excluded unnecessary columns. They were exported to new CSV files. Since the original dataset did not include a file for countries. For example,

```
import pandas as pd
import re

filtered_athletes_df = pd.read_csv('filtered_athletes.csv')
teams_df = pd.read_csv('teams.csv')
disciplines_df = pd.read_csv('disciplines.csv')

# Converts the athletes_codes column to string
teams_df['athletes_codes'] = teams_df['athletes_codes'].astype(str)

final_athletes_list = []

# Iterate through the filtered athletes
for index, athlete in filtered_athletes_df.iterrows():
    athlete_code = str(athlete['code'])
    # Matches team code to present athlete
    team_code_matches = teams_df[teams_df['athletes_codes'].str.contains(r'\b'
+ re.escape(athlete_code) + r'\b', na=False)]

    for _, team in team_code_matches.iterrows():
        discipline_name = athlete['disciplines']
        discipline_code_row = disciplines_df[disciplines_df['discipline_name']
== discipline_name]
        discipline_code = discipline_code_row['discipline_code'].values[0] if
not discipline_code_row.empty else None

        # Append the athlete's details!
        final_athletes_list.append({
            'code': athlete['code'],
            'first_Name': athlete['first_Name'],
            'last_Name': athlete['last_Name'],
            'gender': athlete['gender'],
            'country_code': athlete['country_code'],
            'team_code': team['code'],
            'discipline_code': discipline_code
        })

# Convert the list of dictionaries to a DataFrame and into a CSV
```

```
final_athletes_df = pd.DataFrame(final_athletes_list)
final_athletes_df.to_csv('final_filtered_athletes.csv', index=False)
```

I extracted unique country codes from the filtered athletes' data and created a new country.csv. This is because only the countries that participated in watersports were of interest.

```
import pandas as pd

# Reading from filtered athletes because we only require their countries
df = pd.read_csv('filtered_athletes.csv')

# Extract unique country codes
unique_country_codes = df['country_code'].unique()

# Save unique country codes
with open('unique_country_codes.txt', 'w') as f:
    for code in unique_country_codes:
        f.write(f"{code}\n")
```

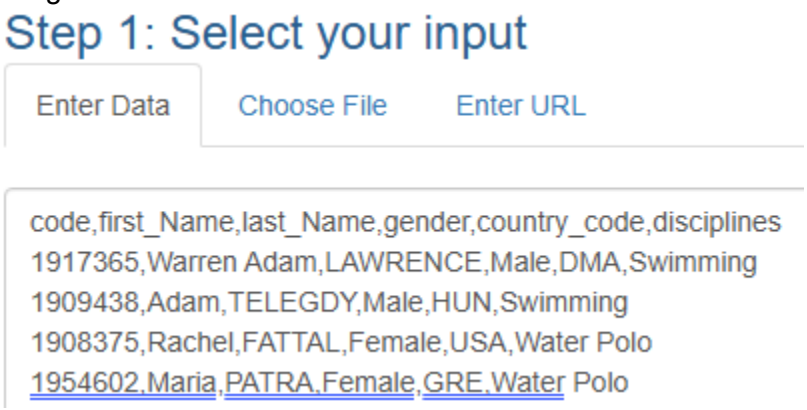
When creating a script to filter water sport Events, I added an auto incrementing Event_ID because the original dataset didn't include Event_IDs so I created them myself.

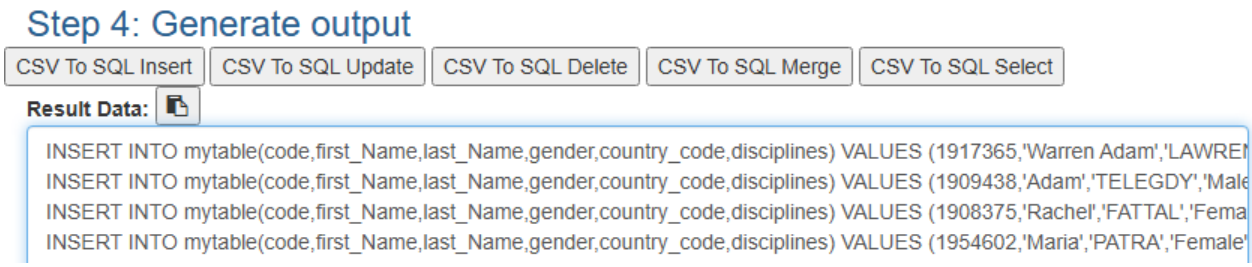
```
df_final['Event_ID'] = range(1, len(df_final) + 1)
```

I manually created the discipline and venue_sport CSV files as they didn't have a lot of entries so I was able to do them without the use of Python Scripts.

Inserting Sample Data

A free CSV to SQL converter was used to generate INSERT statements for each table. The website automatically reads the CSV file and separates them into columns by each comma and header name. These INSERT statements were saved into separate SQL files e.g., athlete_insert.sql and I inserted the data into the database by running each SQL file manually using the SOURCE command.





SQL Queries

Level 1 – Basic Queries

Query 1: Find all athletes competing in a discipline e.g. "water-polo" whose athlete code is greater than '1900000'.

```
SELECT Athlete_Code, First_Name, Last_Name, Discipline_Code
FROM Athlete
WHERE Discipline_Code = 'water-polo'
AND Athlete_Code > '1900000';
```

```
mysql> SELECT Athlete_Code, First_Name, Last_Name, Discipline_Code
    -> FROM Athlete
    -> WHERE Discipline_Code = 'water-polo'
    -> AND Athlete_Code > '1900000';
+--------------+-----------------+-------------------------+-----------------+
| Athlete_Code | First_Name      | Last_Name               | Discipline_Code |
+--------------+-----------------+-------------------------+-----------------+
| 1906412      | Emil            | BJORCH                  | water-polo      |
| 1906434      | Mehdi           | MARZOUKI                | water-polo      |
| 1906443      | Pierre-Frederic | VANPEPERSTRAETE         | water-polo      |
| 1908350      | Alex            | BOWEN                   | water-polo      |
| 1908359      | Drew            | HOLLAND                 | water-polo      |
| 1915769      | Luka            | BUKIC                   | water-polo      |
| 1925653      | Francesco       |  FULVIO                 | water-polo      |
| 1925665      | Nicholas        | PRESCIUTTI              | water-polo      |
| 1925667      | Alessandro      | VELOTTO                 | water-polo      |
| 1947504      | Blake           | EDWARDS                 | water-polo      |
| 1947531      | Milos           | MAKSIMOVIC              | water-polo      |
| 1954614      | Stylianos       | ARGYROPOULOS KANAKAKIS  | water-polo      |
| 1954662      | Nikolaos        | GKILLAS                 | water-polo      |
| 1954879      | Martin          | FAMERA                  | water-polo      |
| 1956426      | Milos           | CUK                     | water-polo      |
| 1956434      | Radoslav        | FILIPOVIC               | water-polo      |
| 1956444      | Dusan           | MANDIC                  | water-polo      |
+--------------+-----------------+-------------------------+-----------------+
```

Query 2: Retrieve all countries where the name starts with a certain string e.g. "United".

```
SELECT Country_Code, Country_Name
FROM Country
WHERE Country_Name LIKE 'United%';
```

```
mysql> SELECT Country_Code, Country_Name
    -> FROM Country
    -> WHERE Country_Name LIKE 'United%';
+--------------+----------------+
| Country_Code | Country_Name   |
+--------------+----------------+
| GBR          | United Kingdom |
| USA          | United States  |
+--------------+----------------+
```

Query 3: Show all medals awarded on a certain date e.g. July 27, 2024.

```sql
SELECT Medal_Type, Team_Code, Event_ID, Date_Achieved
FROM Medal
WHERE Date_Achieved = '2024-07-27';
```

```
mysql> SELECT Medal_Type, Team_Code, Event_ID, Date_Achieved
    -> FROM Medal
    -> WHERE Date_Achieved = '2024-07-27';
+--------------+-------------------+----------+---------------+
| Medal_Type   | Team_Code         | Event_ID | Date_Achieved |
+--------------+-------------------+----------+---------------+
| Silver Medal | SWMM4X100MFRAUS01 | 46       | 2024-07-27    |
| Bronze Medal | SWMM4X100MFRITA01 | 46       | 2024-07-27    |
| Gold Medal   | SWMM4X100MFRUSA01 | 46       | 2024-07-27    |
+--------------+-------------------+----------+---------------+
```

Query 4: Calculate the number of days an event is scheduled at a certain venue e.g. "Pont Alexandre III".

```sql
SELECT Venue,
       DATEDIFF(Date_End, Date_Start) AS Event_Duration
FROM Venue
WHERE Venue = 'Pont Alexandre III';
```

```
mysql> SELECT Venue,
    ->        DATEDIFF(Date_End, Date_Start) AS Event_Duration
    -> FROM Venue
    -> WHERE Venue = 'Pont Alexandre III';
+--------------------+----------------+
| Venue              | Event_Duration |
+--------------------+----------------+
| Pont Alexandre III |             10 |
+--------------------+----------------+
1 row in set (0.03 sec)
```

Query 5: Find all athletes from a specific country (e.g. USA):

```sql
SELECT First_Name, Last_Name, Gender
FROM Athlete
WHERE Country_Code = 'USA';
```

```
mysql> SELECT First_Name, Last_Name, Gender
    -> FROM Athlete
    -> WHERE Country_Code = 'USA';
+------------+-----------+--------+
| First_Name | Last_Name | Gender |
+------------+-----------+--------+
| Alex       | BOWEN     | Male   |
| Drew       | HOLLAND   | Male   |
| Nic        | FINK      | Male   |
| Calista    | LIU       | Female |
+------------+-----------+--------+
4 rows in set (0.01 sec)
```

Level 2 – Advanced Queries

Query 1: Show the total number of medals awarded in each discipline.

```sql
SELECT d.Discipline_Name, COUNT(m.Medal_Type) AS Medal_Count
FROM Medal m
JOIN Event e ON m.Event_ID = e.Event_ID
JOIN Discipline d ON e.Discipline_Code = d.Discipline_Code
GROUP BY d.Discipline_Name
ORDER BY Medal_Count DESC;
```

```
mysql> SELECT d.Discipline_Name, COUNT(m.Medal_Type) AS Medal_Count
    -> FROM Medal m
    -> JOIN Event e ON m.Event_ID = e.Event_ID
    -> JOIN Discipline d ON e.Discipline_Code = d.Discipline_Code
    -> GROUP BY d.Discipline_Name
    -> ORDER BY Medal_Count DESC;
+------------------+-------------+
| Discipline_Name  | Medal_Count |
+------------------+-------------+
| Swimming         |          12 |
| Diving           |           6 |
| Water Polo       |           6 |
| Artistic Swimming |           3 |
| Surfing          |           3 |
+------------------+-------------+
5 rows in set (0.01 sec)
```

Query 2: Find the top 3 teams that have won the most gold medals.

```sql
SELECT t.Team_Name, COUNT(m.Medal_Type) AS Gold_Medals
FROM Medal m
JOIN Team t ON m.Team_Code = t.Team_Code
WHERE m.Medal_Type = 'Gold Medal'
GROUP BY t.Team_Name
ORDER BY Gold_Medals DESC
LIMIT 3;
```

```
mysql> SELECT t.Team_Name, COUNT(m.Medal_Type) AS Gold_Medals
    -> FROM Medal m
    -> JOIN Team t ON m.Team_Code = t.Team_Code
    -> WHERE m.Medal_Type = 'Gold Medal'
    -> GROUP BY t.Team_Name
    -> ORDER BY Gold_Medals DESC
    -> LIMIT 3;
+---------------------------+-------------+
| Team_Name                 | Gold_Medals |
+---------------------------+-------------+
| People's Republic of China |           4 |
| Serbia                    |           3 |
| United States of America  |           2 |
+---------------------------+-------------+
```

Query 3: Find the total number of medals for each team in the a discipline e.g. Water Polo

```sql
SELECT t.Team_Name, COUNT(m.Medal_Type) AS Total_Medals
FROM Medal m
JOIN Team t ON m.Team_Code = t.Team_Code
JOIN Discipline d ON t.Discipline_Code = d.Discipline_Code
WHERE d.Discipline_Name = 'Water Polo'
```

```
GROUP BY t.Team_Name
ORDER BY Total_Medals DESC;
```

```
mysql> SELECT t.Team_Name, COUNT(m.Medal_Type) AS Total_Medals
    -> FROM Medal m
    -> JOIN Team t ON m.Team_Code = t.Team_Code
    -> JOIN Discipline d ON t.Discipline_Code = d.Discipline_Code
    -> WHERE d.Discipline_Name = 'Water Polo'
    -> GROUP BY t.Team_Name
    -> ORDER BY Total_Medals DESC;
+-------------------------+--------------+
| Team_Name               | Total_Medals |
+-------------------------+--------------+
| Croatia                 |            3 |
| Serbia                  |            3 |
| United States of America |            3 |
+-------------------------+--------------+
3 rows in set (0.00 sec)
```

Query 4: List all teams and their total medal count

```
SELECT t.Team_Name, COUNT(m.Medal_Type) AS Medal_Count
FROM Team t
LEFT JOIN Medal m ON t.Team_Code = m.Team_Code
GROUP BY t.Team_Name
ORDER BY Medal_Count DESC;
```

```
mysql> SELECT t.Team_Name, COUNT(m.Medal_Type) AS Medal_Count
    -> FROM Team t
    -> LEFT JOIN Medal m ON t.Team_Code = m.Team_Code
    -> GROUP BY t.Team_Name
    -> ORDER BY Medal_Count DESC;
+---------------------------+-------------+
| Team_Name                 | Medal_Count |
+---------------------------+-------------+
| United States of America  |           8 |
| People's Republic of China |           5 |
| Serbia                    |           3 |
| Croatia                   |           3 |
| Great Britain             |           3 |
| Australia                 |           3 |
| Canada                    |           1 |
| Italy                     |           1 |
| Spain                     |           1 |
| Mexico                    |           1 |
| France                    |           1 |
| Montenegro                |           0 |
| South Africa              |           0 |
| Lithuania                 |           0 |
| Switzerland               |           0 |
| Netherlands               |           0 |
| Republic of Korea         |           0 |
| Ireland                   |           0 |
| Romania                   |           0 |
| Austria                   |           0 |
| Sweden                    |           0 |
| Poland                    |           0 |
| Israel                    |           0 |
| Hungary                   |           0 |
| Greece                    |           0 |
| Brazil                    |           0 |
| Japan                     |           0 |
| Egypt                     |           0 |
| Ukraine                   |           0 |
| Germany                   |           0 |
+---------------------------+-------------+
30 rows in set (0.01 sec)
```

Query 5: Get the average number of athletes per team in each discipline

```sql
SELECT Discipline_Name, AVG(athlete_count) AS Avg_Athletes_Per_Team
FROM (
    SELECT d.Discipline_Name, COUNT(a.Athlete_Code) AS athlete_count
    FROM Team t
    JOIN Athlete a ON t.Team_Code = a.Team_Code
    JOIN Discipline d ON t.Discipline_Code = d.Discipline_Code
    GROUP BY t.Team_Code, d.Discipline_Name
) AS Athlete_Team
GROUP BY Discipline_Name
ORDER BY Avg_Athletes_Per_Team DESC;
```

```
mysql> SELECT Discipline_Name, AVG(athlete_count) AS Avg_Athletes_Per_Team
    -> FROM (
    ->     SELECT d.Discipline_Name, COUNT(a.Athlete_Code) AS athlete_count
    ->     FROM Team t
    ->     JOIN Athlete a ON t.Team_Code = a.Team_Code
    ->     JOIN Discipline d ON t.Discipline_Code = d.Discipline_Code
    ->     GROUP BY t.Team_Code, d.Discipline_Name
    -> ) AS Athlete_Team
    -> GROUP BY Discipline_Name
    -> ORDER BY Avg_Athletes_Per_Team DESC;
+-------------------+-----------------------+
| Discipline_Name   | Avg_Athletes_Per_Team |
+-------------------+-----------------------+
| Artistic Swimming |                2.3333 |
| Water Polo        |                2.0909 |
| Diving            |                1.6667 |
| Swimming          |                1.1200 |
+-------------------+-----------------------+
```

Query 6: Medals achieved within a date range

```sql
SELECT t.Team_Name, m.Medal_Type, m.Date_Achieved
FROM Medal m
JOIN Team t ON m.Team_Code = t.Team_Code
WHERE m.Date_Achieved BETWEEN '2024-07-27' AND '2024-08-10'
ORDER BY m.Date_Achieved ASC;
```

```
mysql> SELECT t.Team_Name, m.Medal_Type, m.Date_Achieved
    -> FROM Medal m
    -> JOIN Team t ON m.Team_Code = t.Team_Code
    -> WHERE m.Date_Achieved BETWEEN '2024-07-27' AND '2024-08-10'
    -> ORDER BY m.Date_Achieved ASC;
+--------------------------+-------------+---------------+
| Team_Name                | Medal_Type  | Date_Achieved |
+--------------------------+-------------+---------------+
| United States of America | Gold Medal  | 2024-07-27    |
| Italy                    | Bronze Medal | 2024-07-27    |
| Australia                | Silver Medal | 2024-07-27    |
| People's Republic of China | Gold Medal  | 2024-07-29    |
| Great Britain            | Silver Medal | 2024-07-29    |
| Canada                   | Bronze Medal | 2024-07-29    |
| United States of America | Silver Medal | 2024-07-30    |
| Great Britain            | Gold Medal  | 2024-07-30    |
| Australia                | Bronze Medal | 2024-07-30    |
| Mexico                   | Silver Medal | 2024-08-02    |
| Great Britain            | Bronze Medal | 2024-08-02    |
| People's Republic of China | Gold Medal  | 2024-08-02    |
| United States of America | Gold Medal  | 2024-08-03    |
| People's Republic of China | Silver Medal | 2024-08-03    |
| Australia                | Bronze Medal | 2024-08-03    |
| United States of America | Silver Medal | 2024-08-04    |
| France                   | Bronze Medal | 2024-08-04    |
| People's Republic of China | Gold Medal  | 2024-08-04    |
| United States of America | Silver Medal | 2024-08-07    |
| Spain                    | Bronze Medal | 2024-08-07    |
| People's Republic of China | Gold Medal  | 2024-08-07    |
+--------------------------+-------------+---------------+
21 rows in set (0.01 sec)
```

Query 7: retrieves the highest medal count for each team in a given time period e.g. '2024-07-27' AND '2024-08-10'

```sql
SELECT t.Team_Name, MAX(Medal_Count) AS Highest_Medals
FROM (
SELECT t.Team_Code, Count(m.Medal_Type) AS Medal_Count
FROM Medal m
      JOIN Team t ON m.Team_Code = t.Team_Code
WHERE m.Date_Achieved BETWEEN '2024-07-27' AND '2024-08-10'
GROUP BY t.Team_Code
) AS SubQuery
JOIN Team t ON SubQuery.Team_Code = t.Team_Code
Group BY t.Team_Name;
```

```
mysql> SELECT t.Team_Name, MAX(Medal_Count) AS Highest_Medals
    -> FROM (
    ->     SELECT t.Team_Code, COUNT(m.Medal_Type) AS Medal_Count
    ->     FROM Medal m
    ->     JOIN Team t ON m.Team_Code = t.Team_Code
    ->     WHERE m.Date_Achieved BETWEEN '2024-07-27' AND '2024-08-10'
    ->     GROUP BY t.Team_Code
    -> ) AS SubQuery
    -> JOIN Team t ON SubQuery.Team_Code = t.Team_Code
    -> GROUP BY t.Team_Name;
+--------------------------+----------------+
| Team_Name                | Highest_Medals |
+--------------------------+----------------+
| Canada                   |              1 |
| People's Republic of China |              1 |
| Great Britain            |              1 |
| Mexico                   |              1 |
| Spain                    |              1 |
| United States of America |              1 |
| Australia                |              1 |
| Italy                    |              1 |
| France                   |              1 |
+--------------------------+----------------+
9 rows in set (0.07 sec)
```

Stored Procedures and Triggers

Stored Procedure to Retrieve Athletes by Discipline

This stored procedure retrieves a list of athletes that participate in a specified discipline. It expects one parameter which is discipline VARCHAR(50) and then retrieves specific athletes that match the input parameter.

```sql
DELIMITER //

CREATE PROCEDURE GetAthletesByDiscipline(
    IN discipline VARCHAR(50)
)
```

```
BEGIN
    SELECT Athlete_Code, First_Name, Last_Name, Gender, Country_Code,
Team_Code
    FROM Athlete
    WHERE Discipline_Code = discipline;
END //

DELIMITER ;
```

```
mysql> SOURCE GetAthletesByDiscipline.sql
Query OK, 0 rows affected (0.03 sec)

mysql> CALL GetAthletesByDiscipline('swimming');
+-------------+------------+-------------+--------+--------------+------------------+
| Athlete_Code | First_Name | Last_Name   | Gender | Country_Code | Team_Code        |
+-------------+------------+-------------+--------+--------------+------------------+
| 1543303     | Max        |  CUSKER     | Male   | IRL          | SWMM4X100MMDIRL01 |
| 1565166     | Bernhard   | REITSHAMMER | Male   | AUT          | SWMM4X100MMDAUT01 |
| 1572556     | Bjoern     | SEELIGER    | Male   | SWE          | SWMM4X100MFRSWE01 |
| 1891648     | Naoki      | MIZUNUMA    | Male   | JPN          | SWMM4X100MMDJPN01 |
| 1891652     | Katsuhiro  | MATSUMOTO   | Male   | JPN          | SWMM4X100MMDJPN01 |
| 1897309     | Anastasia  | GORBENKO    | Female | ISR          | SWMX4X100MMDISR01 |
| 1901612     | Sunwoo     | HWANG       | Male   | KOR          | SWMM4X100MMDKOR01 |
| 1902960     | Adam       | JASZO       | Male   | HUN          | SWMM4X100MFRHUN01 |
| 1903540     | Alexander  | COHOON      | Male   | GBR          | SWMM4X100MFRGBR01 |
| 1903552     | James      | GUY         | Male   | GBR          | SWMM4X200MFRGBR01 |
| 1903557     | Anna       | HOPKIN      | Female | GBR          | SWMX4X100MMDGBR01 |
| 1909285     | Guillaume  | GUTH        | Male   | FRA          | SWMM4X100MFRFRA01 |
| 1909299     | Yohann     | BROUARD     | Male   | FRA          | SWMM4X100MMDFRA01 |
| 1909314     | Antoine    | VIQUERAT    | Male   | FRA          | SWMX4X100MMDFRA01 |
| 1909315     | W. Amazigh | YEBBA       | Male   | FRA          | SWMM4X100MFRFRA01 |
| 1925636     | Alessandro | MIRESSI     | Male   | ITA          | SWMM4X100MFRITA01 |
| 1925686     | Filippo    | MEGLI       | Male   | ITA          | SWMM4X200MFRITA01 |
| 1935901     | Nic        | FINK        | Male   | USA          | SWMM4X100MMDUSA01 |
| 1945149     | Changhao   | WANG        | Male   | CHN          | SWMM4X100MMDCHN01 |
| 1945166     | Xuwei      | PENG        | Female | CHN          | T172050237073     |
| 1946185     | Zac        | COOK        | Male   | AUS          | SWMX4X100MMDAUS01 |
| 1954343     | Guilherme  | SANTOS      | Male   | BRA          | SWMM4X100MFRBRA01 |
| 1954841     | Ferran     | JULIA TOUS  | Male   | ESP          | SWMM4X200MFRESP01 |
| 1967138     | Yuri       | KISIL       | Male   | CAN          | SWMM4X100MFRCAN01 |
| 1967144     | Lorne      | WIGGINTON   | Male   | CAN          | SWMM4X200MFRCAN01 |
| 1967158     | Patrick    | HUSSEY      | Male   | CAN          | SWMM4X200MFRCAN01 |
| 1972527     | Tessa      | GIELE       | Female | NED          | SWMX4X100MMDNED01 |
| 1977717     | Nils       | LIESS       | Male   | SUI          | SWMM4X100MMDSUI01 |
+-------------+------------+-------------+--------+--------------+------------------+
28 rows in set (0.02 sec)
```

Stored Procedure to Retrieve the Number of Athletes for a Specific Discipline

This stored procedure retrieves the count of athletes that participate in a specified discipline. It expects one parameter which is Discipline_Code VARCHAR(50) and then retrieves specific athletes using COUNT to count the number of athletes that match the input parameter. JOIN is used to count the athletes.

```
DELIMITER //

CREATE PROCEDURE GetDisciplineDetails(IN p_Discipline_Code VARCHAR(50))
BEGIN
    SELECT
        d.Discipline_Name,
        COUNT(a.Athlete_Code) AS Athlete_Count
    FROM
        Discipline d
```

```
    LEFT JOIN
        Athlete a ON d.Discipline_Code = a.Discipline_Code
    WHERE
        d.Discipline_Code = p_Discipline_Code
    GROUP BY
        d.Discipline_Name;
END //

DELIMITER ;
```

```
mysql> CALL GetDisciplineDetails('artistic-swimming');
+-------------------+---------------+
| Discipline_Name   | Athlete_Count |
+-------------------+---------------+
| Artistic Swimming |            14 |
+-------------------+---------------+
1 row in set (0.06 sec)

Query OK, 0 rows affected (0.06 sec)
```

Trigger to Log Event Additions

This trigger logs the event details every time a new event is added to the Event table. AFTER INSERT ON is set to activate after an insert operation is performed. It will insert a new record of Event with its ID and Name into the Event_Log table .

```
DELIMITER //

CREATE TRIGGER LogEventInsertion
AFTER INSERT ON Event
FOR EACH ROW
BEGIN
    INSERT INTO Event_Log (Event_ID, Event_Name)
    VALUES (NEW.Event_ID, NEW.Event_Name);
END //

DELIMITER ;
```

```
mysql> CREATE TABLE Event_Log (
MENT PRIMARY KEY,
    Event_ID VARCHAR(50),
    Ev    ->     Log_ID INT AUTO_INCREMENT PRIMARY KEY,
    ->     Event_ID VARCHAR(50),
    ->     Event_Name VARCHAR(100),
    ->     Date_Added DATETIME DEFAULT CURRENT_TIMESTAMP
    -> );
Query OK, 0 rows affected (0.33 sec)

mysql> SOURCE LogEventInsertion.sql;
Query OK, 0 rows affected (0.09 sec)
```

```
mysql> INSERT INTO Event (Event_ID, Event_Tag, Event_Name, Discipline_Code, Venue) VALUES ('54'
, 'water-polo', 'Women-Two', 'water-polo', 'Paris La Defense Arena');
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM Event_Log;
+--------+----------+------------+---------------------+
| Log_ID | Event_ID | Event_Name | Date_Added          |
+--------+----------+------------+---------------------+
|      1 | 54       | Women-Two  | 2024-10-23 17:21:35 |
+--------+----------+------------+---------------------+
1 row in set (0.02 sec)
```

Trigger that Checks for Duplicate Event_IDs

This trigger ensures that duplicate Event_IDs cannot be inserted into the Event table. SELECT COUNT(*) INTO duplicate_count queries the Event table to count how many times the new Event_ID already exists in the table. This is used to maintain data integrity.

```
DELIMITER //

CREATE TRIGGER PreventDuplicateEventID
BEFORE INSERT ON Event
FOR EACH ROW
BEGIN
    DECLARE duplicate_count INT;

    SELECT COUNT(*)
    INTO duplicate_count
    FROM Event
    WHERE Event_ID = NEW.Event_ID;

    IF duplicate_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Duplicate Event_ID not allowed';
    END IF;
END //
```

```
DELIMITER ;
```

```
mysql> SOURCE PreventDuplicateEventID.sql;
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO Event (Event_ID, Event_Tag, Event_Name, Discipline_Code, Venue) VALUES ('54'
, 'water-polo', 'Women-Three', 'water-polo', 'Paris La Defense Arena');
ERROR 1644 (45000): Duplicate Event_Tag not allowed.
```

Python3 Scripts

CRUD Athletes

This script establishes a connection to MYSQL and provides basic Create, Read, Update, Delete operations on the Athlete table within the database. The main function will connect to the database using my credentials and then can call each function depending on which insert you keep in the code. Cursors are objects that allow you to retrieve a result set, in this case they were used to fetch the athletes' details.

Insert Athlete

```
if connection:
    # Insert an athlete
    insert_athlete(connection, '9999999', 'John', 'Doe', 'Male', 'USA', 'SWAWTEAM8---USA01', 'artistic-swimming')
```

```
mysql> SELECT * FROM Athlete WHERE Athlete_Code = '9999999';
+--------------+------------+-----------+--------+--------------+-------------------+-------------------+
| Athlete_Code | First_Name | Last_Name | Gender | Country_Code | Team_Code         | Discipline_Code   |
+--------------+------------+-----------+--------+--------------+-------------------+-------------------+
| 9999999      | John       | Doe       | Male   | USA          | SWAWTEAM8---USA01 | artistic-swimming |
+--------------+------------+-----------+--------+--------------+-------------------+-------------------+
1 row in set (0.00 sec)
```

Update Athlete

```
    # Update the athlete's team
    update_athlete(connection, '9999999', 'DIVM10MTEAM2UKR01')
```

```
mysql> SELECT * FROM Athlete WHERE Athlete_Code = '9999999';
+--------------+------------+-----------+--------+--------------+-------------------+-------------------+
| Athlete_Code | First_Name | Last_Name | Gender | Country_Code | Team_Code         | Discipline_Code   |
+--------------+------------+-----------+--------+--------------+-------------------+-------------------+
| 9999999      | John       | Doe       | Male   | USA          | DIVM10MTEAM2UKR01 | artistic-swimming |
+--------------+------------+-----------+--------+--------------+-------------------+-------------------+
1 row in set (0.00 sec)
```

Delete Athlete

```
# Delete the athlete
delete_athlete(connection, '9999999')
```

```
mysql> SELECT * FROM Athlete WHERE Athlete_Code = '9999999';
Empty set (0.01 sec)
```

Medal Queries

These scripts were defined from previous queries regarding total medals. The main function will connect to the database using my credentials and a simple menu allows users to enter the discipline or date range. Cursors are objects that allow you to retrieve a result set, in this example they were used to fetch discipline_name and date ranges.

Query 1: Find the total number of medals for each team in the a discipline e.g. Water Polo

```
 /mnt/c/Users/izhar/Downloads/Database/Database/sql_database/count_medals.py
Connection to MySQL database was successful.

Menu:
1. Find total medals for a discipline
2. Exit
Enter your choice: 1
Enter the name of the discipline: Swimming

Total Medals for Each Team in 'Swimming':
Team Name: United States of America, Total Medals: 4
Team Name: Australia, Total Medals: 3
Team Name: People's Republic of China, Total Medals: 2
Team Name: Italy, Total Medals: 1
Team Name: France, Total Medals: 1
Team Name: Great Britain, Total Medals: 1
```

Query 2: Medals Achieved Within a Date Range e.g. 2024-08-10 to 2024-09-10

```
izharyf@Izhary:/mnt/c/Users/izhar/Downloads/Database/Database/sql_database$ /bin/python3 /m
s/izhar/Downloads/Database/Database/sql_database/medals_date_range.py
Connection to MySQL database was successful.

Menu:
1. Find medals achieved within a date range
2. Exit
Enter your choice: 1
Enter the start date (YYYY-MM-DD): 2024-08-10
Enter the end date (YYYY-MM-DD): 2024-09-10

Medals Achieved Between '2024-08-10' and '2024-09-10':
Team Name: Croatia, Medal Type: Silver Medal, Date Achieved: 2024-08-11
Team Name: Croatia, Medal Type: Silver Medal, Date Achieved: 2024-08-11
Team Name: Croatia, Medal Type: Silver Medal, Date Achieved: 2024-08-11
Team Name: Serbia, Medal Type: Gold Medal, Date Achieved: 2024-08-11
Team Name: Serbia, Medal Type: Gold Medal, Date Achieved: 2024-08-11
Team Name: Serbia, Medal Type: Gold Medal, Date Achieved: 2024-08-11
Team Name: United States of America, Medal Type: Bronze Medal, Date Achieved: 2024-08-11
Team Name: United States of America, Medal Type: Bronze Medal, Date Achieved: 2024-08-11
Team Name: United States of America, Medal Type: Bronze Medal, Date Achieved: 2024-08-11
```

Discussion

Throughout the project, I was able to apply various database and programming concepts learned from previous practicals, including topics such as subqueries, constraints, stored procedures, triggers, and database connections via a programming language (Python). I created the required tables and relationships for the database, focusing on athletes, teams, medals, disciplines, and events. I also implemented constraints and triggers to ensure data integrity, such as preventing duplicate Event IDs and logging event insertions.I created Python scripts to interact with the database, such as inserting, updating, and deleting athlete records, as well as retrieving team medals for specific disciplines.

One of the main challenges was filtering the CSV files using Python. Differences in capitalization between column headers and data required a lot of manipulation, which led to unexpected errors. It was tedious to keep everything consistent. While I could create a Python script to retrieve medals won by teams, I struggled with implementing a similar script for individual athletes due to complexity in how the data was structured. Ultimately, I chose to exclude individual athletes' medal data. I initially tried to write a Python script to generate SQL INSERT statements for each table. However, I soon realized that free tools were available online. After spending time trying to get my script to work, I decided to switch to the online tool, which was much faster.

References

CSV To SQL Converter. (n.d.). Www.convertcsv.com. https://www.convertcsv.com/csv-to-sql.htm

Galli, K. (2018, October 26). *Complete Python Pandas Data Science Tutorial! (Reading CSV/Excel files, Sorting, Filtering, Groupby)*. Www.youtube.com. https://www.youtube.com/watch?v=vmEHCJofslg

Pandas. (2018). *Python Data Analysis Library*. Pydata.org. https://pandas.pydata.org/

tutorialspoint. (2024). *MySQL - SIGNAL Statement*. Tutorialspoint.com. https://www.tutorialspoint.com/mysql/mysql_signal_statement.htm

Vanyuk, P. (2024). *Paris 2024 Olympic Summer Games*. Kaggle.com. https://www.kaggle.com/datasets/piterfm/paris-2024-olympic-summer-games?resource=download&select=venues.csv