Olympics Database Report

Final Assignment

Jaeden Mah

20717432

Thursday 10am

Introduction

This project involved working with a MySQL database to store data from 2024 Olympics. Data to be managed includes athletes/teams, countries, medals, coaches and the disciplines. These data were individually entered into separate tables in order to be later accessed in different ways based on the needs of the user. The user accesses the database via a python based program with a basic menu function. From the python program, the user can select options that call queries and pre defined procedures.

The tasks in the assignment were carefully planned starting from deriving entities/relationships and drawing out the schema to creating the structure of the database followed by creating complex functionality within the database and ending with a connection to a python based program.

Design of Database

For my Olympics2024 database design, I selected the entities and relationships I felt were most appropriate:

Athletes

Athletes are the main entity of the Olympics so creating a table just for this entity will allow me to store important details such as their id, name, gender and their total medals won. Their ID is important as it identifies each athlete from each other

Teams

Many of the events are team based so teams as an entity will allow the user to search for their details such as their ID and number of athletes in the team. The team ID distinguishes the teams from each other.

Discipline

After athletes, disciplines are the next most important entity as without disciplines, the athletes have nothing to compete in and wouldn't be able to exist. Disciplines store the name, gender of the discipline and venue its held in. Both the name and gender identify the discipline.

Medals

Medals are awarded to athletes who win however a medal cannot be distinguished on its own and needs an athlete ID to identify it. The medals table can store both the type of medal won and its corresponding code.

Countries

This is vital for the database as athletes and teams represent a country they are competing for. The country table can store a country name and its corresponding three letter code with its name identifying the country.

Coaches

A coach guides the teams in a specific discipline and must store the name, ID and the gender of the coach. Here the ID will distinguish the coaches from each other.

IndividualRepresent

An athlete must represent a country they are competing for. Many athletes can compete for the same country, but each athlete must only represent one country. An athlete doesn't always have to represent a country at all but a country must have athletes representing it or else it cant participate.

Won

A medal must be won by an athlete. Each medal can only be awarded to one athlete but an athlete can win many medals over the duration of the Olympics. Not every athlete will win a medal and all medals won must have an athlete who won them.

TeamRepresent

A team must represent a country they are competing for. Many teams can compete for the same country, but each team can only represent one country. A team doesn't always have to represent a country at all but a country must have teams representing it or else it cant participate.

CoachRepresent

A coach must represent a country as well which will be the same country as the team he is coaching represents. Multiple coaches can represent the same country as they will be in different disciplines, but each coach can only represent one country. A coach can exist without a country must have coaches to participate.

Coaches

Every coach is assigned to a team and every team has only one coach. A coach must have a team in order to participate, however a team doesn't always need a coach.

CompetingIn

This states the relationship between a team and the discipline that they are competing in. A team can competing in only one discipline but each discipline can have multiple teams competing in it from different countries. A team cannot exist without competing in a discipline and a discipline cannot exist unless it has a team participating in it in 2024.

Entities	Primary Attributes	Attributes
Athletes	ID	athleteName, gender, totalMedals
Teams	ID	numAthletes
Discipline	DisciplineName, gender	Venue
Medals (weak)	Code	MedalType
Countries	CountryName	code
Coach	ID	coachName gender,

Relationship	Entities	Cardinality	Participation
IndividualRepresent	Athletes-Countries	Many-One	Partial-Total (country cant participate unless it has athletes
Won (weak)	Athletes-Medals	One-Many	Partial-Total
TeamRepresent	Team-Countries	Many-One	Partial-Total
CoachRepresent	Coaches-Countries	Many-One	Partial-Total
Coaches	Coaches-Teams	One-One	Total-Partial (coach needs a team in order to be in Olympics)
CompetingIn	Teams-Discipline	One-Many	Total-total

Schema

Athletes(ID, athleteName, totalMedals, gender, countryName)

FK fk_Athletes_countryName REF Countries(countryName)

Teams(ID, numAthletes, countryName, disciplineName, disciplineGender)

FK fk_Teams_countryName REF Countries(countryName)

FK fk_Teams_discipline REF Disciplines(disciplineName)

FK fk_Teams_discipline REF Discipline(disciplineGender)

Disciplines(disciplineeName, gender, venue)

Medals(Code, athleteID, medalType)

FK fk_Medals_athleteID REF Athletes(ID)

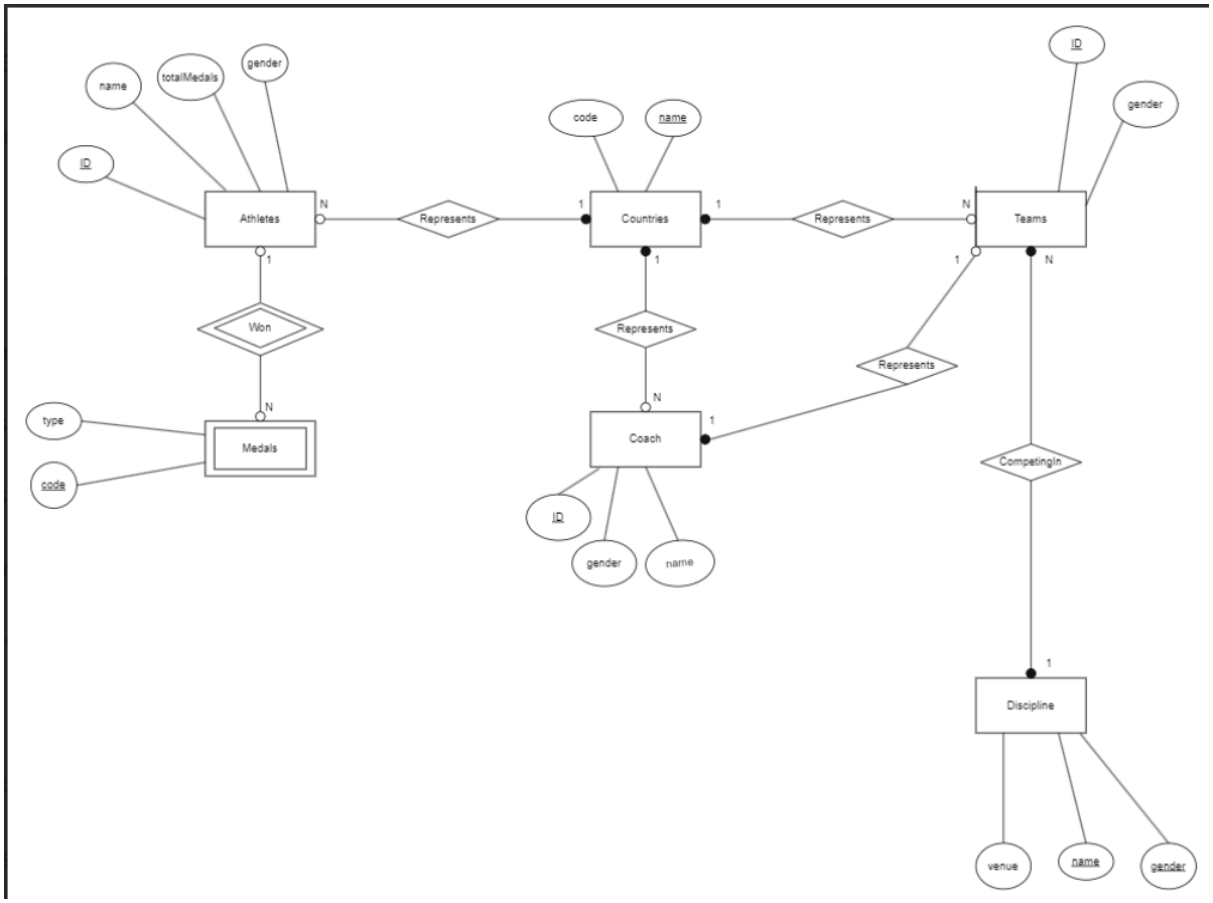
Countries(countryName, code)

Coach(ID, coachName, gender, countryName, teamID)

FK fk_Coach_countryName REF Countries(countryName)

FK fk_Coach_teamID REF Teams(ID)

ER Diagram


name
totalMedals
gender
ID
Athletes
N
Represents
1
code
name
Countries
1
Represents
N
ID
gender
Teams
1
N
Won
1
N
Represents
1
N
Represents
type
code
Medals
Coach
1
ID
gender
name
CompetingIn
1
Discipline
venue
name
gender

Athletes

Attribute	Data type	Description/Business Rule
ID	CHAR(7) Primary Key	Must be exactly 7 characters, and unique.
athleteName	VARCHAR(35) NOT NULL	Every individual must have a name.
totalMedals	INT	Number of medals won
gender	ENUM('M', 'F')	M for male and F for female
countryName	VARCHAR(30)	Country athlete is competing for

Teams

Attribute	Data type	Description
ID	VARCHAR(25) Primary Key	Must be less than 25 characters
numAthletes	Int CHECK (numAthletes > 0)	Number of athletes in the team. Must be more than 0
CountryName	VARCHAR(30) NOT NULL	Country the team represents
DisciplineName	VARCHAR(30) NOT NULL	Discipline the team is competing in
DisciplineGender	ENUM('M', 'F') NOT NULL	Events are either for men or women participants

Disciplines

Attribute	Data type	Description
disciplineName	VARCHAR(30) Primary Key	Discipline name must be less than 30 characters
Gender	ENUM('M', 'F') Primary Key	Events are either for men or women participants
Venue	VARCHAR(50) NOT NULL	Venue the event is held in

Medals

Attribute	Data type	Description
code	INT Primary Key	Either 1, 2 or 3 corresponding to gold, silver or bronze respectively
athleteID	CHAR(7) Primary Key	Id of the athlete who won the medal
medalType	ENUM('Gold Medal', 'Silver Medal', 'Bronze Medal') NOT NULL	"Gold, "Silver", "Bronze"

Countries

Attribute	Data type	Description
countryName	VARCHAR(30) Primary Key	Name of the country
code	CHAR(3) NOT NULL	Country name abbreviated as 3 character string

Coach

Attribute	Data type	Description
ID	CHAR(7) Primary Key	Id of the coach must be 7 characters
coachName	VARCHAR(35) NOT NULL	Name of the coach
Gender	ENUM('M', 'F')	M for male and F for female
CountryName	VARCHAR(30) NOT NULL	Country the coach represents
teamID	VARCHAR(25) NOT NULL	id of the team the coach trains

Assumptions

- An athlete cannot change the country they represent during the Olympics games
- A team can only be formed from athletes from the same country
- A coach can only coach one team
- An athlete may not represent a country

Implementing Database and Adding Sample Data

Entities such as Athletes, Teams, Countries, Disciplines, Coaches and Medals were defined each with primary keys to distinguish them from others. Medals was designed as a weak entity as it depends on Athletes.

The relationships derived were implemented to represent the connections between the entities. On top of that, cardinality and participation constraints were added to show the nature of the instances of the entity's participation with each other.

The database was normalised to 3NF to ensure there was minimal redundancy and anomalies. For example, instead of including the country in most entities, a separate table for Countries was added and linked with the other entities by countryID. This is a good use of foreign keys which is used to make sure relationships between entities were consistently linking the correct instances.

```
# Create Countries table
CREATE TABLE Countries(
    countryName VARCHAR(30) PRIMARY KEY,
    code CHAR(3) NOT NULL
);


# Create Athletes table
CREATE TABLE Athletes(
    ID CHAR(7) PRIMARY KEY,
    athleteName VARCHAR(35) NOT NULL,
    gender ENUM('M', 'F'),
    totalMedals INT,
    countryName VARCHAR(30),
    CONSTRAINT fk_Athletes_countryName
        FOREIGN KEY (countryName)
        REFERENCES Countries(countryName)
);


# Create Disciplines table
CREATE TABLE Disciplines(
    disciplineName VARCHAR(30),
    gender ENUM('M', 'F'),
    venue VARCHAR(50) NOT NULL,
    PRIMARY KEY (disciplineName, gender)
);


# Create Teams table
CREATE TABLE Teams(
    ID VARCHAR(25) PRIMARY KEY,
    numAthletes INT CHECK (numAthletes > 0),
    countryName VARCHAR(30) NOT NULL,
    disciplineName VARCHAR(30) NOT NULL,
    disciplineGender ENUM('M', 'F'),
    CONSTRAINT fk_Teams_countryName
        FOREIGN KEY (countryName)
        REFERENCES Countries(countryName),
    CONSTRAINT fk_Teams_discipline
        FOREIGN KEY (disciplineName, disciplineGender)
        REFERENCES Disciplines(disciplineName, gender)
);
```

```
# Create Medals table
CREATE TABLE Medals(
    code INT,
    athleteID CHAR(7),
    medalType ENUM('Gold Medal', 'Silver Medal', 'Bronze Medal') NOT NULL,
    PRIMARY KEY (code, athleteID),
    CONSTRAINT fk_Medals_athleteID
        FOREIGN KEY (athleteID)
        REFERENCES Athletes(ID)
);


# Create Coach table
CREATE TABLE Coach(
    ID CHAR(7) PRIMARY KEY,
    coachName VARCHAR(35) NOT NULL,
    gender ENUM('M', 'F'),
    countryName VARCHAR(30) NOT NULL,
    teamID VARCHAR(25) NOT NULL,
    CONSTRAINT fk_Coach_countryName
        FOREIGN KEY (countryName)
        REFERENCES Countries(countryName),
    CONSTRAINT fk_Coach_teamID
        FOREIGN KEY (teamID)
        REFERENCES Teams(ID)
);
```

The Olympics was obtained by downloading csv files from data *Paris 2024 Olympic Summer Games [1]* and building the columns for each of my entities table in Excel. After this step, I created a java program to read the csv file and format it where it returns each row of the csv file as a SQL insert statement and appends each statement to a .sql file for a specific entity.

A function in my java program to convert csv file with Athlete data from excel into SQL insert Athlete statements

```
public static String athletes(String data){
  String insertStatement;

  String[] medalAttributes = data.split(",");

  if((medalAttributes[2].trim().replace("'", "''")).equals("Male")){
    medalAttributes[2] = "M";
  }
  if((medalAttributes[2].trim().replace("'", "''")).equals("Female")){
    medalAttributes[2] = "F";
  }

  insertStatement = "INSERT INTO Athletes VALUES('" + medalAttributes[0].trim().replace("'", "''") + "', '" + medalAttributes[1].trim().replace("'", "''") + "', '" + me

  return insertStatement;

}
```

The csv file being processed

	A	B	C	D	E
1	code	name	gender	country	
2	1532872	ALEKSANY	Male	Armenia	
3	1532873	AMOYAN I	Male	Armenia	
4	1532874	GALSTYAN	Male	Armenia	
5	1532944	HARUTYUI	Male	Armenia	
6	1532945	TEVANYAN	Male	Armenia	
7	1532951	ARENAS Lo	Female	Colombia	
8	1533112	McKENZIE	Male	Jamaica	
9	1533136	BASS BITT/	Female	Gambia	
10	1533176	CAMARA E	Male	Gambia	
11	1533188	RUEDA SA	Female	Mexico	
12	1533189	TAPIA VID/	Female	Mexico	
13	1533190	GRAJALES	Male	Mexico	
14	1533208	MAAROUF	Male	Comoros	
15	1533209	SAADI Mae	Female	Comoros	
16	1533230	DIOSDADO	Female	Mexico	
17	1533231	JIMENEZ J	Female	Mexico	
18	1533232	SOBRINO	Female	Mexico	
19	1533234	ALFEREZ R	Female	Mexico	
20	1533235	ARELLANC	Female	Mexico	
21	1533237	TOSCANO	Female	Mexico	
22	1533239	GONZALEZ	Female	Mexico	
23	1533240	RODRIGUE	Female	Mexico	
24	1533243	INZUNZA (	Female	Mexico	
25	1533245	SEHEN Saj	Male	Iraq	

The resulting insert statements aftter csv file is processed

```
INSERT INTO Athletes VALUES('1532872', 'ALEKSANYAN Artur', 'M', NULL, 'Armenia');
INSERT INTO Athletes VALUES('1532873', 'AMOYAN Malkhas', 'M', NULL, 'Armenia');
INSERT INTO Athletes VALUES('1532874', 'GALSTYAN Slavik', 'M', NULL, 'Armenia');
INSERT INTO Athletes VALUES('1532944', 'HARUTYUNYAN Arsen', 'M', NULL, 'Armenia');
INSERT INTO Athletes VALUES('1532945', 'TEVANYAN Vazgen', 'M', NULL, 'Armenia');
INSERT INTO Athletes VALUES('1532951', 'ARENAS Lorena', 'F', NULL, 'Colombia');
INSERT INTO Athletes VALUES('1533112', 'McKENZIE Ashley', 'M', NULL, 'Jamaica');
INSERT INTO Athletes VALUES('1533136', 'BASS BITTAYE Gina Mariam', 'F', NULL, 'Gamb
INSERT INTO Athletes VALUES('1533176', 'CAMARA Ebrahima', 'M', NULL, 'Gambia');
INSERT INTO Athletes VALUES('1533188', 'RUEDA SANTOS Lizeth', 'F', NULL, 'Mexico');
INSERT INTO Athletes VALUES('1533189', 'TAPIA VIDAL Rosa Maria', 'F', NULL, 'Mexic
INSERT INTO Athletes VALUES('1533190', 'GRAJALES Crisanto', 'M', NULL, 'Mexico');
INSERT INTO Athletes VALUES('1533208', 'MAAROUFOU Hachim', 'M', NULL, 'Comoros');
INSERT INTO Athletes VALUES('1533209', 'SAADI Maesha', 'F', NULL, 'Comoros');
```

.sql file that will import all the files containing the insert statements for each table

```
# Create new database
CREATE DATABASE Olympics2024_20717432
USE Olympics2024_20717432

# Call file that creates the tables
SOURCE CreateTables.sql;

# Call the files that fill the tables
SOURCE CreateTables.sql;
SOURCE insertCountries.sql;
SOURCE insertAthletes.sql;
SOURCE insertDisciplines.sql;
SOURCE insertTeams.sql;
SOURCE insertMedals.sql;
SOURCE insertCoaches.sql;

# Add other sql files
SOURCE procedures.sql;
SOURCE triggers.sql;
```

Result of calling the above file shown below where the insert statements are executed

```
mysql> source InitialiseDatabase.sql;
Query OK, 1 row affected (0.02 sec)

Database changed
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.05 sec)
```

The resulting tables created along with views

```
mysql> show tables;
+---------------------------------+
| Tables_in_Olympics2024_20717432 |
+---------------------------------+
| AthleteMedal                    |
| Athletes                        |
| Coach                           |
| Countries                       |
| Disciplines                     |
| Medals                          |
| TeamDisciplineCoachInfo         |
| Teams                           |
| TeamsInBadminton                |
| TeamsWithCountries              |
+---------------------------------+
10 rows in set (0.00 sec)
```

Once these files are imported, the constraints enforced earlier on in the creation of the tables will ensure any lines with unusual data will be left out preventing "invalid" data from being entered into the database (note not all the data from the 2024 Olympics will be entered into this database). Examples of constraints being enforced when inserting out are shown below.

Example 1

Gender cannot be anything other than M or F

```
# Create Athletes table
CREATE TABLE Athletes(
    ID CHAR(7) PRIMARY KEY,
    athleteName VARCHAR(35) NOT NULL,
    gender ENUM('M', 'F'),
    totalMedals INT,
    countryName VARCHAR(30),
    CONSTRAINT fk_Athletes_countryName
        FOREIGN KEY (countryName)
        REFERENCES Countries(countryName)
```

Rejects this statement as G is used instead of M or F

```
mysql> insert INSERT INTO Athletes VALUES('1532872', 'ALEKSANYAN Artur', 'G', NULL,
'Armenia');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corr
esponds to your MySQL server version for the right syntax to use near 'INSERT INTO A
thletes VALUES('1532872', 'ALEKSANYAN Artur', 'G', NULL, 'Armenia')' at line 1
```

Example 2

Number of athletes (numAthletes) must be more than 0

```
# Create Teams table
CREATE TABLE Teams(
    ID VARCHAR(25) PRIMARY KEY,
    numAthletes INT CHECK (numAthletes > 0),
    countryName VARCHAR(30) NOT NULL,
    disciplineName VARCHAR(30) NOT NULL,
    disciplineGender ENUM('M', 'F'),
    CONSTRAINT fk_Teams_countryName
        FOREIGN KEY (countryName)
        REFERENCES Countries(countryName),
```

Rejects this statement as number of athletes entered is 0

```
mysql> INSERT INTO Teams VALUES('ARCMTEAM3---CHN01', '0', 'China', 'Archery', 'M');
ERROR 3819 (HY000): Check constraint 'Teams_chk_1' is violated.
mysql>
```

Example 3

Medal type must be either Gold, Bronze or Silver

```
# Create Medals table
CREATE TABLE Medals(
    code INT,
    athleteID CHAR(7),
    medalType ENUM('Gold Medal', 'Silver Medal', 'Bronze Medal') NOT NULL,
    PRIMARY KEY (code, athleteID),
    CONSTRAINT fk_Medals_athleteID
        FOREIGN KEY (athleteID)
        REFERENCES Athletes(ID)
```

Rejects this statement as medal entered is not of Gold, Silver or Bronze type

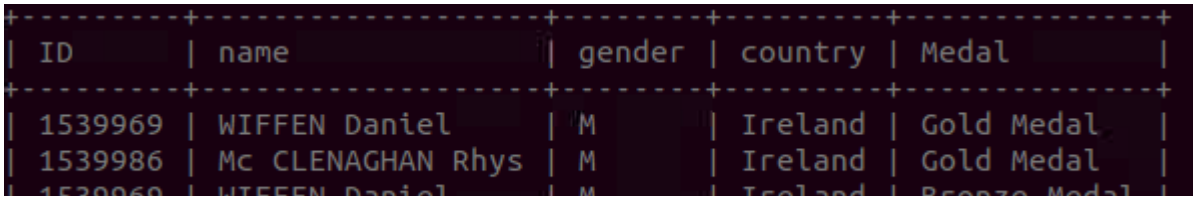
```
mysql> INSERT INTO Medals VALUES('1', '1903136', 'Blue Medal');
ERROR 1265 (01000): Data truncated for column 'medalType' at row 1
mysql>
```

Implementing Queries

Query 1

This query allows the user to find the information of an Irish competing athlete that has won at least 1 medal. These results are useful for finding what athletes from a specific country have won medals.

```
# Query for obtaining the individual athletes who won medals and are from Ireland using INNER
JOIN
SELECT ID, athleteName AS "name", gender, countryName AS "country", medalType AS "Medal"
FROM Athletes INNER JOIN Medals
ON ID = athleteID
WHERE countryName = "Ireland";
```

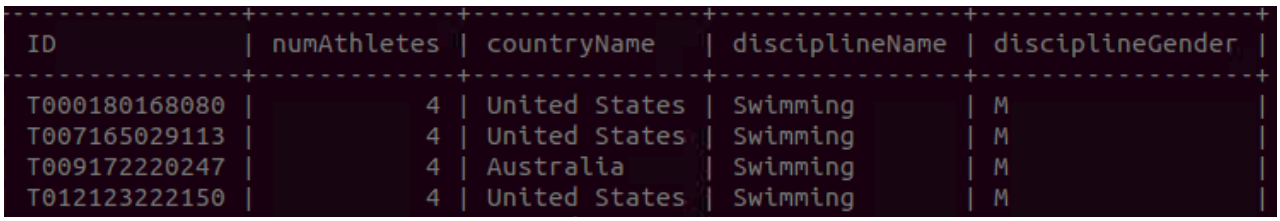


ID	name	gender	country	Medal
1539969	WIFFEN Daniel	M	Ireland	Gold Medal
1539986	Mc CLENAGHAN Rhys	M	Ireland	Gold Medal

Query 2

This query allows the user to find all the teams who compete in a specific discipline and their team name starts with a specific letter

```
# Query to find all the teams who compete in Swimming and ID start with "T"
SELECT*
FROM Teams
WHERE disciplineName = "Swimming" and ID LIKE 'T%';
```

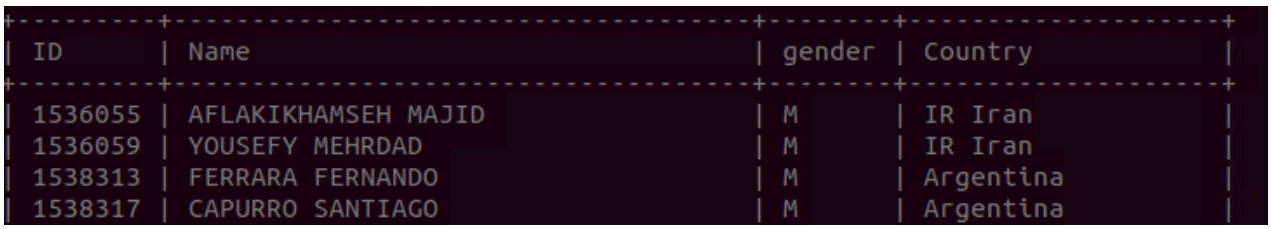


ID	numAthletes	countryName	disciplineName	disciplineGender
T000180168080	4	United States	Swimming	M
T007165029113	4	United States	Swimming	M
T009172220247	4	Australia	Swimming	M
T012123222150	4	United States	Swimming	M

Query 3

Query to allow the user to change the name of male coachs to all uppercase letter in case there is a change of format required

```
# Query to manipulate the name string of a Coach that is male to uppercase letters
SELECT ID, UPPER(coachName) AS "Name", gender, countryName AS "Country"
FROM Coach
WHERE gender = "M";
```



ID	Name	gender	Country
1536055	AFLAKIKHAMSEH MAJID	M	IR Iran
1536059	YOUSEFY MEHRDAD	M	IR Iran
1538313	FERRARA FERNANDO	M	Argentina
1538317	CAPURRO SANTIAGO	M	Argentina

Query 4

Query to allow the user to find female teams with more than 10 athletes

```
# Query to display teams with more than 10 athletes and are are female team
SELECT*
FROM Teams
WHERE numAthletes > 10 and disciplineGender = "F";
```

ID	numAthletes	countryName	disciplineName	disciplin
BKBWTEAM5---AUS01	12	Australia	Basketball	F
BKBWTEAM5---BEL01	12	Belgium	Basketball	F
BKBWTEAM5---CAN01	12	Canada	Basketball	F
BKBWTEAM5---CHN01	12	China	Basketball	F

Query 5

I couldn't find a legitimate use of concatenation for my database so I concatenated name and gender just to demonstrate I can concatenate in general,

```
# Query concatenating full name and gender(I have all names in database as full name so instead
i will concatenate gender and name)
SELECT ID, CONCAT(athleteName, ' ', gender) AS "Name and Gender", countryName AS country
FROM Athletes;
```

ID	Name and Gender	country
1532872	ALEKSANYAN Artur M	Armenia
1532873	AMOYAN Malkhas M	Armenia
1532874	GALSTYAN Slavik M	Armenia

Query 6

A query to get a list of athletes from specified countries

```
# Query to get athletes from countries that are from a specified list of countries
SELECT*
FROM Athletes
WHERE countryName IN ('Ethiopia', 'Argentina', 'Mexico');
```

ID	athleteName	gender	totalMedals	countryNam
1533188	RUEDA SANTOS Lizeth	F	NULL	Mexico
1533189	TAPIA VIDAL Rosa Maria	F	NULL	Mexico
1533190	GRAJALES Crisanto	M	NULL	Mexico
1533230	DIOSDADO Nuria	F	NULL	Mexico

Query 7

A query to allow the user to find the number of athletes representing a country

```
# Query to find the number of athletes in each country using Count() and ORDER BY and
GROUP BY
SELECT countryName, COUNT(ID) as athlete_count
FROM Athletes
GROUP BY countryName
ORDER BY athlete_count DESC;
```

countryName	athlete_count
Argentina	84
Mexico	84
India	42
Ireland	26
IR Iran	25
Ethiopia	20
Romania	19
Colombia	15

Query 8

A query to allow the user to find any athletes who have won more medals than the average amount of medals won by an athlete

```
# Query to find any athlete who won more medals than the average number of medals won
by individuals
SELECT A.athleteName, COUNT(M.medalType) AS total_medals
FROM Athletes A
JOIN Medals M ON A.ID = M.athleteID
GROUP BY A.athleteName
HAVING total_medals > (
    SELECT AVG(total_medals)
    FROM (SELECT COUNT(*) AS total_medals
          FROM Medals
          GROUP BY athleteID) AS subquery
);
```

athleteName	total_medals
ALFRED Julien	2
WIFFEN Daniel	2

Query 9

A query to allow the user to display every single athlete and also any medals they won if applicable

```
# Query to display all athletes and if applicable, if they won a medal
SELECT A.ID, A.athleteName AS "name", A.gender, A.countryName as "Country",
M.medalType AS "Medal"
FROM Athletes A LEFT JOIN Medals M ON A.ID = M.athleteID;
```

```
    -> FROM Athletes A LEFT JOIN Medals M ON A.ID = M.athleteID;
+---------+--------------------------------------+--------+----------------+-------
| ID      | name                                 | gender | Country        | Medal
+---------+--------------------------------------+--------+----------------+-------
| 1532872 | ALEKSANYAN Artur                     | M      | Armenia        | Silver
| 1532873 | AMOYAN Malkhas                       | M      | Armenia        | Bronze
| 1532874 | GALSTYAN Slavik                      | M      | Armenia        | NULL
| 1532944 | HARUTYUNYAN Arsen                    | M      | Armenia        | NULL
| 1532945 | TEVANYAN Vazgen                      | M      | Armenia        | NULL
| 1532951 | ARENAS Lorena                        | F      | Colombia       | NULL
```

Query 10

A query to allow the user to view what disciplines each coach is training their teams for

```
# Query to see what disciplines the coaches are coaching using 2 JOIN statements
SELECT C.coachName, C.gender AS "Coach Gender", D.disciplineName AS "Discipline",
D.gender AS "Discipline Gender"
FROM Coach C
JOIN Teams T ON C.teamID = T.ID
JOIN Disciplines D ON T.disciplineName = D.disciplineName AND T.disciplineGender =
D.gender;
```

```
+--------------------------------------+--------------+----------------+------------
| coachName                            | Coach Gender | Discipline     | Discipline
+--------------------------------------+--------------+----------------+------------
| PEDRERO Ofelia                       | F            | Archery        | M
| AFLAKIKHAMSEH Majid                  | M            | Archery        | M
| YOUSEFY Mehrdad                      | M            | Archery        | M
| MADDAH Minoo                         | F            | Archery        | M
| LOFTUS Adriana                       | F            | Archery        | M
```

Implementing Advanced Features

View 1

Creates a view consisting of team and their corresponding country information

```
# Shows view for teams and their country information
CREATE VIEW TeamsWithCountries AS
SELECT
    T.ID AS TeamID,
    T.numAthletes AS NumberOfAthletes,
    C.countryName AS Country,
    C.code AS CountryCode
FROM
    Teams T
JOIN
    Countries C ON T.countryName = C.countryName
ORDER BY
    C.countryName, T.ID;


# View for teams in Badminton
CREATE VIEW TeamsInBadminton AS
SELECT
    t.ID AS TeamID,
    t.numAthletes AS NumberOfAthletes,
    t.countryName AS CountryName,
    d.disciplineName AS DisciplineName,
    d.gender AS DisciplineGender
FROM
    Teams t
JOIN
    Disciplines d ON t.disciplineName = d.disciplineName AND t.disciplineGender =
d.gender
WHERE
    d.disciplineName = 'Badminton';
```

```
mysql> select* from TeamsWithCountries;
+--------------------+------------------+----------------------+-------------+
| TeamID             | NumberOfAthletes | Country              | CountryCode |
+--------------------+------------------+----------------------+-------------+
| CSPMC2-500M-AIN01  |                2 | AIN                  | AIN         |
| TENMDOUBLES-AIN01  |                2 | AIN                  | AIN         |
| TENWDOUBLES-AIN01  |                2 | AIN                  | AIN         |
| TENWDOUBLES-AIN02  |                2 | AIN                  | AIN         |
| FENWTEAMSABRALG01  |                4 | Algeria              | ALG         |
| FBLMTEAM11--ARG01  |               18 | Argentina            | ARG         |
| HBLMTEAM7---ARG01  |               15 | Argentina            | ARG         |
| HOCMTEAM11--ARG01  |               18 | Argentina            | ARG         |
```

View 2

Creates a view consisting of all the teams that compete in badminton

```
# View for teams in Badminton
CREATE VIEW TeamsInBadminton AS
SELECT
    t.ID AS TeamID,
    t.numAthletes AS NumberOfAthletes,
    t.countryName AS CountryName,
    d.disciplineName AS DisciplineName,
    d.gender AS DisciplineGender
FROM
    Teams t
JOIN
    Disciplines d ON t.disciplineName = d.disciplineName AND t.disciplineGender =
d.gender
WHERE
    d.disciplineName = 'Badminton';
```

```
mysql> select* from TeamsInBadminton;
+-------------------+------------------+----------------+----------------+---------+
| TeamID            | NumberOfAthletes | CountryName    | DisciplineName | Discipli
+-------------------+------------------+----------------+----------------+---------+
| BDMMDOUBLES-CAN01 |                2 | Canada         | Badminton      | M
| BDMMDOUBLES-CHN01 |                2 | China          | Badminton      | M
| BDMMDOUBLES-CHN02 |                2 | China          | Badminton      | M
| BDMMDOUBLES-CZE01 |                2 | Czechia        | Badminton      | M
| BDMMDOUBLES-DEN01 |                2 | Denmark        | Badminton      | M
| BDMMDOUBLES-FRA01 |                2 | France         | Badminton      | M
| BDMMDOUBLES-FRA02 |                2 | France         | Badminton      | M
```

View 3

Creates a view consisting of all the coaches and teams they coach

```
# view for Coaches and their teams from their specific country
CREATE VIEW TeamDisciplineCoachInfo AS
SELECT
    t.ID AS TeamID,
    t.numAthletes AS NumberOfAthletes,
    t.countryName AS CountryName,
    d.disciplineName AS DisciplineName,
    d.gender AS DisciplineGender,
    c.coachName AS CoachName,
    c.gender AS CoachGender
FROM
    Teams t
JOIN
    Disciplines d ON t.disciplineName = d.disciplineName AND t.disciplineGender =
d.gender
JOIN
    Coach c ON t.ID = c.teamID;
```

```
+-----------------+------------------+--------------+----------------+------------------+-----------------------------+--------------+
| TeamID          | NumberOfAthletes | CountryName  | DisciplineName | DisciplineGender | CoachName                   | CoachGender |
+-----------------+------------------+--------------+----------------+------------------+-----------------------------+--------------+
| ARCMTEAM3---CHN01 |                3 | China        | Archery        | M                | PEDRERO Ofelia              | F           |
| ARCMTEAM3---FRA01 |                3 | France       | Archery        | M                | AFLAKIKHAMSEH Majid         | M           |
| ARCMTEAM3---GBR01 |                3 | Great Britain | Archery        | M                | YOUSEFY Mehrdad             | M           |
| ARCMTEAM3---IND01 |                3 | India        | Archery        | M                | MADDAH Minoo                | F           |
| ARCMTEAM3---ITA01 |                3 | Italy        | Archery        | M                | LOFTUS Adriana              | F           |
| ARCMTEAM3---JPN01 |                3 | Japan        | Archery        | M                | FERRARA Fernando            | M           |
```

View 4

Creates a view consisting of all the athletes and any medal they won if applicable

```
# view to show all the athletes information and medals they won if applicable
CREATE VIEW AthleteMedal AS
SELECT
    a.athleteName AS Athlete_Name,
    a.gender AS Gender,
    a.countryName AS Country,
    a.totalMedals AS Total_Medals,
    GROUP_CONCAT(m.medalType ORDER BY m.medalType) AS Medals_Won
FROM
    Athletes a
LEFT JOIN
    Medals m ON a.ID = m.athleteID
GROUP BY
    a.ID, a.athleteName, a.gender, a.countryName, a.totalMedals;
```

```
mysql> select* from AthleteMedal
    -> ;
+-----------------------------------+--------+---------------+-------------+-------------------------+
| Athlete_Name                      | Gender | Country       | Total_Medals | Medals_Won              |
+-----------------------------------+--------+---------------+-------------+-------------------------+
| ALEKSANYAN Artur                  | M      | Armenia       |         NULL | Silver Medal            |
| AMOYAN Malkhas                    | M      | Armenia       |         NULL | Bronze Medal            |
| GALSTYAN Slavik                   | M      | Armenia       |         NULL | NULL                    |
| HARUTYUNYAN Arsen                 | M      | Armenia       |         NULL | NULL                    |
| TEVANYAN Vazgen                   | M      | Armenia       |         NULL | NULL                    |
```

Trigger 1

Whenever a new medal is added to the Medals table, the corresponding athlete in the Athletes table should have its total_medals column incremented by 1 or set to 1 if it was originally 0 or NULL

```
CREATE TRIGGER afterInsertMedalUpdateAthlete
AFTER INSERT ON Medals
FOR EACH ROW
BEGIN
    DECLARE current_total INT;

    # Gets the current total medals for the athlete
    SELECT totalMedals INTO current_total FROM Athletes WHERE ID = NEW.athleteID;

    # if the current total is NULL then set it to 1 otherwise, increment it
    IF current_total IS NULL THEN
        UPDATE Athletes
        SET totalMedals = 1
        WHERE ID = NEW.athleteID;
    ELSE
        UPDATE Athletes
        SET totalMedals = current_total + 1
        WHERE ID = NEW.athleteID;
    END IF;
END;
```

Before a medal is added

```
mysql> select* from Athletes where ID = "1533243";
+---------+----------------+--------+-------------+-------------+
| ID      | athleteName    | gender | totalMedals | countryName |
+---------+----------------+--------+-------------+-------------+
| 1533243 | INZUNZA Glenda | F      |        NULL | Mexico      |
+---------+----------------+--------+-------------+-------------+
1 row in set (0.00 sec)
```

After a medal is added, the total_medals is updated

```
mysql> INSERT INTO Medals VALUES('1', '1533243', 'Gold Medal');
Query OK, 1 row affected (0.01 sec)

mysql> select* from Athletes where ID = "1533243";
+---------+----------------+--------+-------------+-------------+
| ID      | athleteName    | gender | totalMedals | countryName |
+---------+----------------+--------+-------------+-------------+
| 1533243 | INZUNZA Glenda | F      |           1 | Mexico      |
+---------+----------------+--------+-------------+-------------+
```

Trigger 2

Whenever a medal is removed from the Medals table, the corresponding athlete in the Athletes table should have its total_medals column decremented by 1

```
CREATE TRIGGER afterDeleteMedalUpdateAthlete
AFTER Delete ON Medals
FOR EACH ROW
BEGIN
    DECLARE current_total INT;

    # Gets the current total medals for the athlete
    SELECT totalMedals INTO current_total FROM Athletes WHERE ID = OLD.athleteID;

    UPDATE Athletes
    SET totalMedals = current_total - 1
    WHERE ID = OLD.athleteID;
END;

//

DELIMITER ;
```

Before a medal is removed

```
mysql> select* from Athletes where ID = "1533243";
+---------+----------------+--------+-------------+-------------+
| ID      | athleteName    | gender | totalMedals | countryName |
+---------+----------------+--------+-------------+-------------+
| 1533243 | INZUNZA Glenda | F      |           1 | Mexico      |
+---------+----------------+--------+-------------+-------------+
1 row in set (0.00 sec)
```

After a medal is removed, the total_medals is updated

```
mysql> delete from Medals where athleteID = "1533243";
Query OK, 1 row affected (0.01 sec)

mysql> select* from Athletes where ID = "1533243";
+---------+----------------+--------+-------------+-------------+
| ID      | athleteName    | gender | totalMedals | countryName |
+---------+----------------+--------+-------------+-------------+
| 1533243 | INZUNZA Glenda | F      |           0 | Mexico      |
+---------+----------------+--------+-------------+-------------+
1 row in set (0.00 sec)
```

Procedure 1

This procedure demonstrates the usage of a local variable and IN parameters. Whenever a new athlete is inserted into the database, it should automatically be assigned the next available ID which is calculated by incrementing by 1 from the largest ID

```
# Procedure that has local variable that increments
DELIMITER //

DROP PROCEDURE IF EXISTS insertNewAthlete;

CREATE PROCEDURE insertNewAthlete(
    IN newAthleteName VARCHAR(35),
    IN newGender ENUM('M', 'F'),
    IN newCountryName VARCHAR(30)
)
COMMENT 'Inserts new athletes into the Athletes table.'
BEGIN
    DECLARE nextNewID CHAR(7); # next available athlete number
    SELECT MAX(ID)+1 FROM Athletes INTO nextNewID; # new id is obtained by finding
the largest athlete id in the database and incrementing it by 1

    # insert new athlete
    INSERT INTO Athletes (ID, athleteName, gender, countryName)
    VALUES (nextNewID, newAthleteName, newGender, newCountryName);
END //

DELIMITER ;
```

Shows the largest ID in the Athletes table before new athlete inserted

```
mysql> SELECT MAX(ID) FROM Athletes;
+---------+
| MAX(ID) |
+---------+
| 1541276 |
+---------+
1 row in set (0.00 sec)
```

Next ID available for new athlete is 1541276

```
mysql> call insertNewAthlete("Bob", "M", "Australia");
Query OK, 1 row affected (0.01 sec)

mysql> select* from Athletes where athleteName = "Bob";
+---------+-------------+--------+-------------+-------------+
| ID      | athleteName | gender | totalMedals | countryName |
+---------+-------------+--------+-------------+-------------+
| 1541277 | Bob         | M      |        NULL | Australia   |
+---------+-------------+--------+-------------+-------------+
1 row in set (0.00 sec)
```

Procedure 2

This procedure gives the user the ability to change the name and gender of a coach whether its during or after the Olympics

```
# Basic Procedure
DELIMITER //

DROP PROCEDURE IF EXISTS modifyCoach;
CREATE PROCEDURE modifyCoach(
    IN coachID CHAR(7),
    IN newCoachName VARCHAR(35),
    IN newGender ENUM('M', 'F')
)
COMMENT 'Sets new name or gender if applicable of a specific Coach'
BEGIN
    UPDATE Coach
    SET coachName = newCoachName,
        gender = newGender
    WHERE ID = coachID;
END //

DELIMITER ;
```

Shows information of athlete 1533246 before the name and gender change

```
mysql>
mysql> select* from Coach where ID = "1533246";
+---------+----------------+--------+-------------+--------------------+
| ID      | coachName      | gender | countryName | teamID             |
+---------+----------------+--------+-------------+--------------------+
| 1533246 | PEDRERO Ofelia | F      | Mexico      | ARCMTEAM3---CHN01 |
+---------+----------------+--------+-------------+--------------------+
1 row in set (0.00 sec)
```

Information after the change

```
mysql> call modifyCoach("1533246", "Tom Marks", "M");
Query OK, 1 row affected (0.01 sec)

mysql> select* from Coach where ID = "1533246";
+---------+-----------+--------+-------------+--------------------+
| ID      | coachName | gender | countryName | teamID             |
+---------+-----------+--------+-------------+--------------------+
| 1533246 | Tom Marks | M      | Mexico      | ARCMTEAM3---CHN01 |
+---------+-----------+--------+-------------+--------------------+
1 row in set (0.00 sec)
```

Procedure 3

This procedure gives the user the ability to count all the medals for each athlete and update this info by going through all the medals in Medals table and counting the number of medal that are won by the same person according to their ID. It will then update total_medals in Athletes table for the athlete with the corresponding ID. It will do this for all the athletes in the Athletes table. If the Athlete didn't win a medal then their total_medals will be set to 0 from NULL

```
CREATE PROCEDURE UpdateTotalMedals()
COMMENT 'Updates total medals in Athletes table after counting from the Medals table'
BEGIN

    DECLARE done INT DEFAULT 0;
    DECLARE athleteID CHAR(7);
    DECLARE medalCount INT;

    # Declare a cursor to get distinct athlete IDs from the Medals table
    DECLARE medalCursor CURSOR FOR
    SELECT athleteID, COUNT(*) FROM Medals
    GROUP BY athleteID;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN medalCursor;

    # Go through each athlete and update their total medals
    countMedals: LOOP

        FETCH medalCursor INTO athleteID, medalCount;

        IF done = 1 THEN
            LEAVE countMedals;
        END IF;

        # Update the totalMedals for the current athlete
        UPDATE Athletes
        SET totalMedals = medalCount
        WHERE ID = athleteID;

    END LOOP;
    CLOSE medalCursor;

    UPDATE Athletes
    SET totalMedals = 0
    WHERE totalMedals IS NULL OR ID NOT IN (SELECT DISTINCT athleteID FROM Medals);

END//
```

Before procedure is called all the totalMedals are set to NULL by default

```
mysql> select* from Athletes;
+---------+-----------------------------------+--------+-------------+---------------+
| ID      | athleteName                       | gender | totalMedals | countryName   |
+---------+-----------------------------------+--------+-------------+---------------+
| 1532872 | ALEKSANYAN Artur                  | M      |        NULL | Armenia       |
| 1532873 | AMOYAN Malkhas                    | M      |        NULL | Armenia       |
| 1532874 | GALSTYAN Slavik                   | M      |        NULL | Armenia       |
| 1532944 | HARUTYUNYAN Arsen                 | M      |        NULL | Armenia       |
| 1532945 | TEVANYAN Vazgen                   | M      |        NULL | Armenia       |
| 1532951 | ARENAS Lorena                     | F      |        NULL | Colombia      |
| 1533112 | MCKENZIE Ashley                   | M      |        NULL | Jamaica       |
```

After the procedure calculates the number of medals for each athlete

```
mysql> call UpdateTotalMedals();
Query OK, 392 rows affected (0.03 sec)

mysql> select* from Athletes;
+---------+-----------------------------------+--------+-------------+---------------+
| ID      | athleteName                       | gender | totalMedals | countryName   |
+---------+-----------------------------------+--------+-------------+---------------+
| 1532872 | ALEKSANYAN Artur                  | M      |           0 | Armenia       |
| 1532873 | AMOYAN Malkhas                    | M      |           0 | Armenia       |
| 1532874 | GALSTYAN Slavik                   | M      |           0 | Armenia       |
| 1532944 | HARUTYUNYAN Arsen                 | M      |           0 | Armenia       |
| 1532945 | TEVANYAN Vazgen                   | M      |           0 | Armenia       |
| 1533112 | MCKENZIE Ashley                   | M      |           0 | Jamaica       |
```

Triggers

These triggers have been implemented to improve the speed of retrieving data. These indexes have been created as these columns are frequently used for data retrieval.

```
# Index for athletes table
CREATE INDEX AthletesIdx ON Athletes(countryName);

# Index for teams table
CREATE INDEX TeamsIdx ON Teams(countryName);

# Index for Teams table
CREATE INDEX DisciplinesIdx ON Disciplines(disciplineName);

# Index for Medals table
CREATE INDEX MedalsIdx ON Medals(athlete_ID);

# Index for Countries table
CREATE INDEX CountriesIdx ON Countries(code);

# Index for Coach table
CREATE INDEX CoachIdx ON Coach(teamID);
```

Database Connectivity with Python

Database connectivity for Olympics2024 is handled using the mysql.connector library in python. First a connection is established by asking the user for the database username and password before passing those input values into the code.

```
def main():
    # Get database credentials from user
    user = input("Enter your database username: ")
    password = input("Enter your database password: ")

    # Make a connection to the database
    conn = mysql.connector.connect(
        user=user,
        password=password,
        host='127.0.0.1',
```

The program displays a menu providing the user with 6 functionalities

```
Menu

1. Find team in certain discipline and team start with specific letter
2. Add an athlete
3. Delete an athlete
4. Update athlete gender
5. Display a table
6. Modify coach
7. Exit
Please select an option (1-6):
```

1. Finding teams based on discipline and ID prefix (SELECT query)

```
# Function to find a team according to teamsa first letter of the name and their discipline
def find_team(cursor, conn):
    # Get discipline and letter
    discipline_name = input("Enter the discipline name (e.g., Swimming): ")
    id_prefix = input("Enter the ID prefix for the team (e.g., T): ")

    # Query to find all the teams who compete in a specified discipline and ID start with a specified letter
    query = "SELECT * FROM Teams WHERE disciplineName = %s AND ID LIKE %s;"

    cursor.execute(query, (discipline_name, f"{id_prefix}%"))
    results = cursor.fetchall()

    # Print the results
    print(f"\nResults for discipline '{discipline_name}' with ID prefix '{id_prefix}':")
    for row in results:
        print(row)
```

```
Enter the discipline name (e.g., Swimming): Swimming
Enter the ID prefix for the team (e.g., T): T

Results for discipline 'Swimming' with ID prefix 'T':
('T000180168080', 4, 'United States', 'Swimming', 'M')
('T007165029113', 4, 'United States', 'Swimming', 'M')
('T009172220247', 4, 'Australia', 'Swimming', 'M')
('T012123222150', 4, 'United States', 'Swimming', 'M')
```

2. Adding a new athlete to the Athletes table (INSERT query)

```python
# Function to add a new athlete from table Athletes
def add_athlete(cursor, conn):
    # Get athletes info
    athlete_id = input("Enter the athlete's ID (7 characters): ")
    athlete_name = input("Enter the athlete's name: ")
    athlete_gender = input("Enter the athlete's gender as F or M: ")
    athlete_total_medals = input("Enter the athlete's total number of medals: ")
    athlete_country = input("Enter the athlete's country: ")

    # Insert statement and formating of data to insert
    insert_stmt = "INSERT INTO Athletes (ID, athleteName, gender, totalMedals, countryName) VALUES (%s, %s, %s, %s, %s)"
    data = (athlete_id, athlete_name, athlete_gender, athlete_total_medals, athlete_country)

    cursor.execute(insert_stmt, data)
    conn.commit()

    print(f"Athlete with ID {athlete_id} has been added.")
```

Adding a new athlete via option 2

```
Please select an option (1-6): 2
You selected Option 2.
Enter the athlete's ID (7 characters): 0000000
Enter the athlete's name: George Bush
Enter the athlete's gender as F or M: M
Enter the athlete's total number of medals: 3
Enter the athlete's country: Australia
Athlete with ID 0000000 has been added.
```

The new athlete shown in the database Athletes table

```
mysql> select* from Athletes where athleteName = "George Bush";
+---------+-------------+--------+-------------+-------------+
| ID      | athleteName | gender | totalMedals | countryName |
+---------+-------------+--------+-------------+-------------+
| 0000000 | George Bush | M      |           3 | Australia   |
+---------+-------------+--------+-------------+-------------+
1 row in set (0.00 sec)
```

3. Deleting medals of an athlete from the Medals table (DELETE query)

```python
# Function to delete a medal from the Medals table
def delete_medal(cursor, conn):
    # Get medal athleteID
    athlete_id = input("Enter the athlete's ID (7 characters): ")

    # Delete statement
    delete_stmt = "DELETE FROM Medals WHERE athleteID = %s"

    cursor.execute(delete_stmt, (athlete_id,))
    conn.commit()

    print(f"Medal with ID {athlete_id} has been removed.")
```

Delete a medalist from the Medals table

```
Please select an option (1-6): 3
You selected Option 3.
Enter the athlete's ID (7 characters): 1536045
Medal with ID 1536045 has been removed.
```

Table before and after function is called (becomes empty set)

```
mysql> select* from Medals where athleteID = "1536045";
+------+-----------+------------+
| code | athleteID | medalType  |
+------+-----------+------------+
|    1 | 1536045   | Gold Medal |
+------+-----------+------------+
1 row in set (0.00 sec)

mysql> select* from Medals where athleteID = "1536045";
Empty set (0.01 sec)
```

4. Updating an athlete's gender (UPDATE query)

```python
# Function to update an athletes gender
def update_athlete_gender(cursor, conn):
    # Get athlete new gender and ID
    athlete_id = input("Enter the athlete's ID (7 characters): ")
    athlete_gender = input("Enter new gender as F or M: ")

    # Update statement and formating of data
    update_stmt = "UPDATE Athletes SET gender = %s WHERE ID = %s"
    data = (athlete_gender, athlete_id)

    cursor.execute(update_stmt, data)
    conn.commit()

    print(f"Athlete with ID {athlete_id} has been updated.")
```

Updating the gender of an athlete

```
Please select an option (1-6): 4
You selected Option 4.
Enter the athlete's ID (7 characters): 0000000
Enter new gender as F or M: F
Athlete with ID 0000000 has been updated.
```

Athletes gender before and after update function is called

```
mysql> select* from Athletes where athleteName = "George Bush";
+---------+-------------+--------+-------------+-------------+
| ID      | athleteName | gender | totalMedals | countryName |
+---------+-------------+--------+-------------+-------------+
| 0000000 | George Bush | M      |           3 | Australia   |
+---------+-------------+--------+-------------+-------------+
1 row in set (0.00 sec)

mysql> select* from Athletes where athleteName = "George Bush";
+---------+-------------+--------+-------------+-------------+
| ID      | athleteName | gender | totalMedals | countryName |
+---------+-------------+--------+-------------+-------------+
| 0000000 | George Bush | F      |           3 | Australia   |
+---------+-------------+--------+-------------+-------------+
1 row in set (0.00 sec)
```

5. Display a specified table (SELECT query)

```python
# Function to display any table specified by the user
def disp_tables(cursor, conn):
    # List of valid table names
    valid_tables = ["Athletes", "Countries", "Medals", "Coach", "Teams", "Disciplines"]

    # Get table name from user
    table_name = input("Choose a table to display \n 1. Athletes\n 2. Countries \n 3. Medals \n 4. Coaches \n 5. Teams \n 6. Disciplines: ")

    # Validate the input
    while table_name not in valid_tables:
        print("Invalid input. Please choose a valid table name.")
        table_name = input("Choose a table to display \n 1. Athletes\n 2. Countries \n 3. Medals \n 4. Coach \n 5. Teams \n 6. Disciplines: ")

    query = f"SELECT * FROM {table_name}"

    cursor.execute(query)
    results = cursor.fetchall()

    # Print the results
    print(f"\nResults for '{table_name}':")
    for row in results:
        print(row)
```

Shows all the information in Teams table

6. Modifying a coach's details using a stored procedure (CALL statement)

```python
# Function to update coaches name and gender if applicable
def modify_coach(cursor, conn):
    # Get coach info from user
    coach_id = input("Enter the coach's ID (7 characters): ")
    new_coach_name = input("Enter the coach's new name: ")
    new_gender = input("Enter the new gender as F or M: ")

    # Call the stored procedure from mysql
    cursor.callproc('modifyCoach', (coach_id, new_coach_name, new_gender))

    conn.commit()

    print(f"Coach with ID {coach_id} has been updated.")
```

Updating the name and gender of a coach

```
Please select an option (1-6): 6
Enter the coach's ID (7 characters): 1533246
Enter the coach's new name: Sally Marks
Enter the new gender as F or M: F
Coach with ID 1533246 has been updated.
```

Details of the coach before and after the procedure is called

```
mysql> select* from Coach where ID = "1533246";
+---------+-----------+--------+-------------+-------------------+
| ID      | coachName | gender | countryName | teamID            |
+---------+-----------+--------+-------------+-------------------+
| 1533246 | Tom Marks | M      | Mexico      | ARCMTEAM3---CHN01 |
+---------+-----------+--------+-------------+-------------------+
1 row in set (0.00 sec)

mysql> select* from Coach where ID = "1533246";
+---------+-------------+--------+-------------+-------------------+
| ID      | coachName   | gender | countryName | teamID            |
+---------+-------------+--------+-------------+-------------------+
| 1533246 | Sally Marks | F      | Mexico      | ARCMTEAM3---CHN01 |
+---------+-------------+--------+-------------+-------------------+
1 row in set (0.00 sec)
```

Reflection

Overall, the assignment was enjoyable to work on. I implemented a design that models the core entities of the Olympics and made meaningful relationships between these entities. These relationships were then enforced using constraints and foreign keys to maintain data integrity. Connecting a java/python program to a database in SQL wasn't as difficult as I thought it would be and I'm glad I can now apply this knowledge to projects I can start on the summer holidays that will involve real life databases.

Challenges I faced involved trying to normalise the database to at least 3NF. It required lots of time and breaking down entities ensuring attributes were placed in the correct tables.

Areas this database could improve in would be:

- Somehow incorporating a way of knowing what team athletes are in
- A much more interactive and diverse Olympics menu
- Implementation of integrity constraints
- Adding more types of relationships such as many to many
- Adding a tertiary relationship to simplify the database

Though the database is functional and effective, future adjustments could allow it to address more specific tasks and include other entities such as technical officials and schedules.

References

[1] *Paris 2024 Olympic Summer Games*. (2024, August 27). Kaggle. https://www.kaggle.com/datasets/piterfm/paris-2024-olympic-summer-games