DATABASE SYSTEMS

Final Assignment Report 2024

Olympics 2024 Database

Unit Code : ISYS5008

Name : Syed Muhammad Ahmed Zaidi

Student ID: 20972008

Lab Group : Thursday Sessions : 10 AM to 12 PM

Contents

Introduction

This project is a comprehensive guide towards the creation of a relational database from the beginning. It required us to dive into the data of a recent world event called Olympics 2024 in which we were expected to find data related to the topic from multiple sources and bring them together under a database by mapping relations with different selected entities and thoughtfully deciding its attributes and data types. The whole database had to be modelled with precise understanding of the knowledge gained throughout the unit to create a relational database which can allow querying interesting information. Furthermore, there was the implementation of advanced features like procedures, views and triggers to understand the potential of SQL in the world of Data. Finally, it was expected to connect to a programming language (python) to expand the horizon of case usage.

Design of the Database

Entities, Relationships and Assumptions

The dataset used in the project was available on Kaggle containing numerous entities of data related to Olympics 2024. Each entity had multiple attributes that were relevant to the topic and could be included in an extracted database being produced to complete the project requirements. However, after doing many brainstorming sessions and trial and error, below is a table showing the shortlisted entities and their attributes used to complete the project.

Table 1 : Entity sets, their keys and attributes

Entity Sets	Keys	Other Attributes
Athletes	athlete_id	ath_name, ath_gender, ath_dob, country_code
Coaches	coach_id, athlete_id	coach_name, coach_gender, coach_type,
Country	country_code	country_name
Discipline	discipline_code	discipline_name
Events	event_id	discipline_code, event_date, event_name, venue_id
Medals	medals_id	medal_code, medal_type, event_id, winner_id, winner_type
Team Participation	team_id, event_id	NA
Teams	team_id	country_code
Venues	venue_id	venue_name, venue_url
Athlete Participation	athlete_id, event_id	NA

The first entity shows one of the most important ones for the event called the athletes. The rationale behind selecting this entity is that it is linked to majority of other entities. This entity is essential to keep as it reflects which athlete played which event and hence won which medals etc. The attributes selected for these were generally the ones that distinguish one from the other. It included the primary key of athlete_id to give it a unique participant number that would differentiate it from all others. The rest of the attributes describe information about the person. For example, their name, gender, date of birth and country etc.

Second similar to it is the teams which represent athletes participating as a group in different events of Olympics. A team_id was given for their unique identity. Since Olympics is distinguishable through nationality, country code was used to describe which area are they presenting.

Another group of people included were the coaches. However, this was made simple by only containing original information of the coaches for athletes. This was created as weak entity having a weak relationship to as they can only be identified in presence of the athlete and not individually. Further more attributes were included to give information about them. For example, their name, gender, and event what type of coach they are.

Then general entities were added to make the model more complex and give more context to the information being provided. For example, Country was added to link it with athletes and teams to show what the nationality of each athlete is and also to show which country are they representing in different events. Similarly, disciplines table was created to show the different types of sports being played. This includes a special discipline code that was extracted from the original data to identify each sport uniquely. Venues was also created to see exactly where each of the games was played and also a url is attached to it as an attribute to find the recordings of the games played in that particular place.

Furthermore, another important entity that links majority of the above mentioned entities together is called Events. This contains a special ID to distinguish each event that happened in the whole of Olympics. It shows the date it took place and the place through the help of Discipline table. Further this is also the entity that has a many to many relationship with teams and athletes which is why special tables of team_particiapation and ath_partcipation were created to show solely which of them participated in which event.

Lastly, a medals table was created to give a conclusion to the information reflecting the results of each event. Medals were awarded in different events and hence this is linked by adding a winner_id that takes up the original ids from athlete's entity and the team's entity.

All above mentioned entities were picked from original data. Some containing around 200 rows while others touching above 9000. This whole database is based on original data with nothing fabricated. Some attributes like venue_id were generated as they were not available in the original data. However, every row of information reflects completely original data from Olympics 2024.

Below are three tables showing and overview of the types of relationships that were created between the selected entities and the constraints linked to them.

Table 2 : Relationship sets and Attributes

Relationship Sets	**Between which Entity Sets**	**Attributes of Relationship Set (if any)**
National of	Athletes & Country	N/A
Coached BY	Coaches & Athletes	N/A
Athlete Participation	Athletes & Events	N/A
Team Participation	Teams & Events	N/A
Represents	Teams & Country	N/A
Held At	Events & Venues	N/A
Includes	Events & Disciplines	N/A
Awarded	Medals & Events	N/A

Table 3 : Relationship sets and Cardinality Constraints

Relationship Set	**Between which Entities**	**Cardinality**
National of	Athletes, Country	One-Many (An athlete belongs to one country, but a country has many athletes)
Coached By	Coaches, Athletes	One-Many (A coach trains many athletes, but each athlete has one coach)
Athlete Participation	Athletes, Events	Many-Many (An athlete can participate in many events, and each event has many athletes)
Team Participation	Teams, Events	Many-Many (A team can participate in many events, and an event can have many teams)
Represents	Teams, Country	Many-One (A team represents one country, but a country can have many teams)
Held At	Events, Venues	One-Many (An event is held at one venue, and a venue holds many event)

Includes	Events, Disciplines	One-Many (An event can include one discipline, and a discipline can be in many events)
Awarded	Medals, Events	Many-One (A event has many medals awarded, one medal is given per event)

Table 3 : Relationship sets and Participation Constraints

Relationship Set	**Between which Entities**	**Participation**
National of	Athletes, Country	Athletes – Total, Country – Partial (Every athlete must belong to a country; A country may not have athletes)
Coached By	Coaches, Athletes	Athletes – Total, Coaches – Partial (Every athlete must have a coach; A coach may not coach any athlete)
Athlete Participation	Athletes, Events	Athletes – Partial, Events – Partial (Athletes may not participate in every event; Events may not include all athletes)
Team Participation	Teams, Events	Teams – Partial, Events – Partial (Teams may not participate in all events; Events may not include all teams)
Represents	Teams, Country	Teams – Total, Country – Partial (Every team must represent a country; Some countries may not have teams)
Held At	Events, Venues	Events – Total, Venues – Total (Every event must be held at a venue; Every venue hosts one event)
Includes	Events, Disciplines	Events – Partial, Disciplines – Partial (Not every event includes all disciplines; Not every discipline is in every event)
Awarded	Medals, Events	Medals – Partial, Events – Partial (Not every every event has a medal; Not every medal is awarded in each event)

The relationships were created keeping in mind the knowledge that was learnt over the unit in regards to the different types of relationships there can be between different entities. They were created in a way that it it reflects complexity however keeping it simple at the same time. From the above tables we can see that the first relationhip is between the athletes and the country they came from. This reflects that they are a national of that country. The cardinality constraint attached to this relationship is One to Many, since it was assumed that a athlete can be a national of one country only but since there are many atheletes having the same nationality. Similarly, since every country must belong to atleast one country, it was set as Total participation whereas it was assumed that there may be countries without athletes hence, partial participation was considered for the other side.

Next relationship is very much similar to the one just discussed. This is between teams and countries. To add verity, this was kept as that teams represent a country as a whole. Therefore it does have the same cardinality constraint of One to Many., the participation constraints is also teams being Total and country being Partial which presents the assumption that every team must represent a country however a country might not have any teams.

Two special relationships created to showcase diversity, were the relationship between events with teams and athletes. It’s a Many to Many relationship as it was assumed that there can be multiple teams and athletes playing in multiple events and at the same time an event can not be played by multiple teams and athletes. The participation constraint for these relationships was assumed to be partial at both sides as athletes may not participate in every event. And, at the same time events may not include all athletes.

Furthermore, to apply advanced relationship concepts, another entity ‘coaches’ reflects a weak entity which cannot be identified by their own characteristics. Hence, they must be attached with the identity of the athlete to make them distinguishable. This was created with the assumption of cardinality being One to Many as it was assumed that an athlete can only be coached by one person while a coach may have many athletes under them. The relationship constraint was set to be Coaches being Partial since, a coach may not be teaching any athlete at one point of time while the athlete must always have a coach.

The events were connected to veneus with the relationship of Held At. This shows where the event is taking place. The relationship is set to have a One to Many relationship with each event being played at only one Venue. However, there can be many events happening in a selected venue. The participation constrains was assumed to be total at both sides since, every event must be held at a venue from the list and also every venue hosts atleasts one event to be a part of Olympics 2024.

Similar to this relationship, another one was created between events and disciplines. This is another One to Many relationship reflecting the fact that an event (considering the event_id) will have one discipline being played. For example, if its mens hockey then theres a separate id for it. On the other hand, the relationship of discipline with events is many as a particular discipline can be in many events. The participation constraint was kept partial for both because not every event includes all disciplines, and at the same time, not every discipline is in every event.

The last relationship is created between the events and the medals. This reflects the result of each event and how each event awarded the medals. The relationship is assumed to be One to Many since an event has many medals awarded to different teams and athletes but only one medal is given per event to the winner. The participation constraint is assumed to be partial to partial since there are different events happening like knockouts etc so there can not be a medal for every event. And also because of knockout rounds, there cannot be a medal for every event.

ER Diagram

Below is a screenshot of the created ER Diagram through Lucid Chart. This can be seen and interacted by clicking the following link.

https://lucid.app/lucidchart/fa774a20-f573-40ee-a195-446670957728/edit?viewport_loc=-2018%2C-642%2C6832%2C2361%2C0eqBfotfVGlV&invitationId=inv_c872bffe-a066-4794-a601-92ec0dce650e

The ER Diagram was created in accordance with Chen's Notation. Suitable shapes were chosen to present entities and their attributes. Connections were created using lines with their endpoints presenting the partiality constraints. Notations of 1, N and M were used to show the cardinality of each relationship with the other. Furthermore, advanced relationships like weak relations were shown through appropriate double-lined shapes.

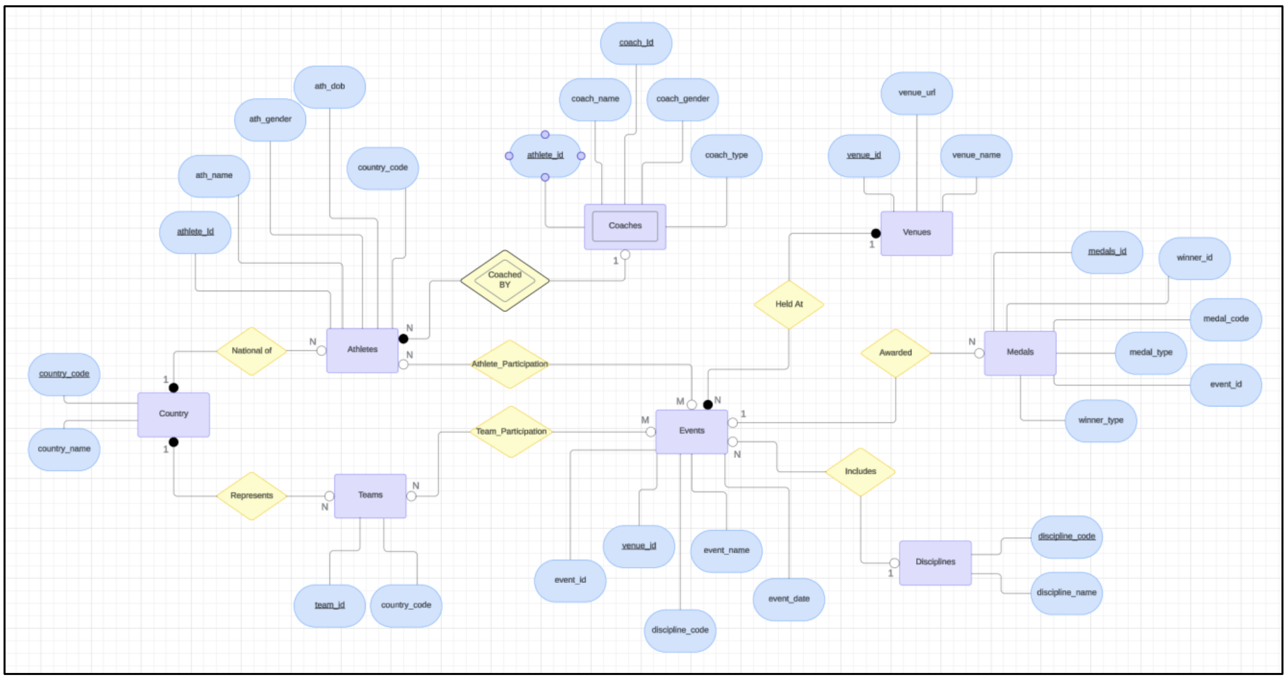


Figure 1 : ER Diagram Using Lucid Chart

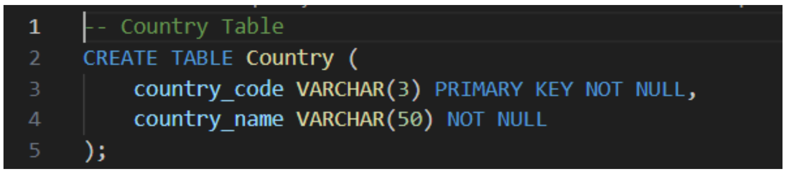
Relational Schema

Once the structure was set it through the ER diagram, the information was then converted into a relational schema. Below you can see how each entity is converted into a relational schema.

- Disciplines (discipline_code, discipline)
- Venues (venue_id, venue_name, venue_url)
- Events (event_id, discipline_code, event_date, event_name, venue_id)
- Teams (team_id, country_code)
- Team_Participation (team_id, event_id)
- Athletes (athlete_id, ath_name, ath_gender, ath_dob, country_code)
- Athlete_Participation (athlete_id, event_id)
- Coaches (coach_id, athlete_id, coach_name, coach_gender, coach_type)
- Medals (medals_id, medal_code, medal_type, event_id, winner_id, winner_type)

Tables, Data Types and Data Description

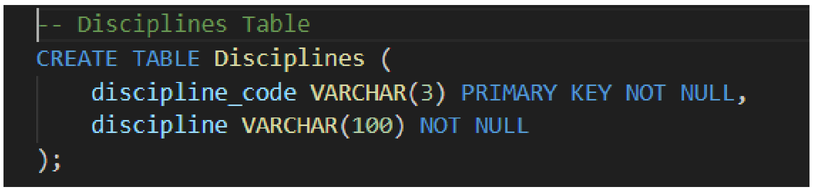
The data of each selected attribute was studied in depth to find the relevant data types. Below are screenshots of the code for each table created alongside the explanation of what kinds of data types are used for it.



```
-- Country Table
CREATE TABLE Country (
    country_code VARCHAR(3) PRIMARY KEY NOT NULL,
    country_name VARCHAR(50) NOT NULL
);
```

Figure 2 : Country

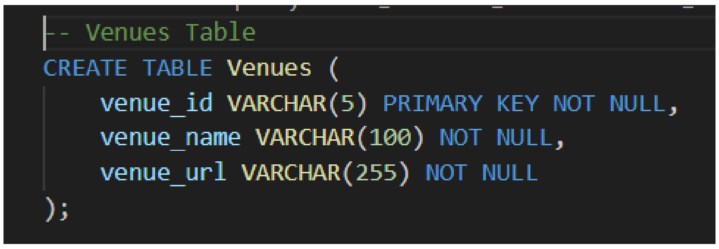
This being a comparatively easy table, the country entity was created with only two attributes, one of them reflecting the original codes from the dataset provided on Kaggle. The data type used for this is VARCHAR with a limit of 3 since all codes reflected a maximum of three alphabets. This was set to primary to make it unique since each country will only be taking up one space. Since this is a primary key it was set to NOT NULL so that it maintains the the integrity and does not cause errors later on. This is followed by country_name, which is the complete name of each country. The data type is set to be VARCHAR with a limit of 50 since there were some countries that had long names. In total, the data set contains 211 Rows of data with each representing the country that was used in the original dataset.



```sql
-- Disciplines Table
CREATE TABLE Disciplines (
    discipline_code VARCHAR(3) PRIMARY KEY NOT NULL,
    discipline VARCHAR(100) NOT NULL
);
```

Figure 3 : Disciplines

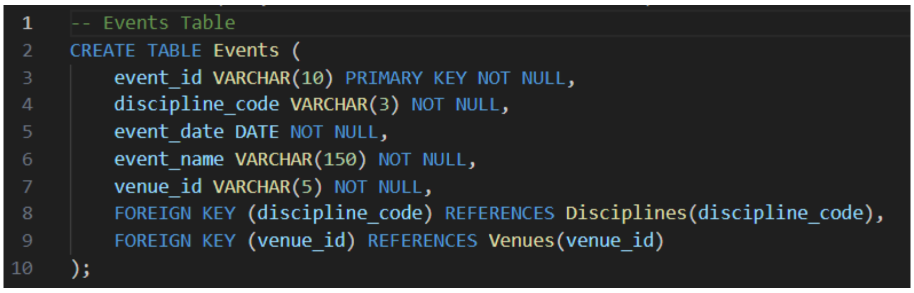
A very similar table with a different genre of information is Disciplines. This reflects the different types of sports that are being played. Each sport/ discipline is given a code which was extracted from the original data and was set in the same manner of VARCHAR (3). It was put as a primary key for keeping each of them unique. The name of the discipline was set to VARCHAR (100) to avoid errors if incase a discipline name is too long. Both attributes were set to NOT NULL so that the data must be filled up for each entry. The disciplines given in the data are all extracted from the original data, they are of the same name and type with no fabrication done. In total for this table there are 45 rows each reflecting a different discipline that was played in Olympics 2024.



```sql
-- Venues Table
CREATE TABLE Venues (
    venue_id VARCHAR(5) PRIMARY KEY NOT NULL,
    venue_name VARCHAR(100) NOT NULL,
    venue_url VARCHAR(255) NOT NULL
);
```

Figure 4 : Venues

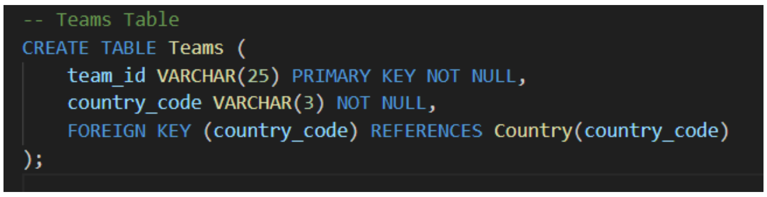
Following the same structure, the table of Venues includes 3 attributes. One being the primary key which is the Venue_id. This attribute was randomly generated since there was no venue_id given in any of the files. However, the venue_name and venue_url are real data that show the original name of the premises where certain disciplines were played along with the url used to watch them. The data type used for them is VARCHAR with varying limits on the characters depending on what they store. For example, it was kept the highest for url since, at times these can be very long. They are all set to NOT NULL to avoid anything being left without entry. In total in Olympics 2024, there were 35 Venues used and all 35 of them have been included in the dataset.



```sql
-- Events Table
CREATE TABLE Events (
    event_id VARCHAR(10) PRIMARY KEY NOT NULL,
    discipline_code VARCHAR(3) NOT NULL,
    event_date DATE NOT NULL,
    event_name VARCHAR(150) NOT NULL,
    venue_id VARCHAR(5) NOT NULL,
    FOREIGN KEY (discipline_code) REFERENCES Disciplines(discipline_code),
    FOREIGN KEY (venue_id) REFERENCES Venues(venue_id)
);
```

Figure 5 : Events

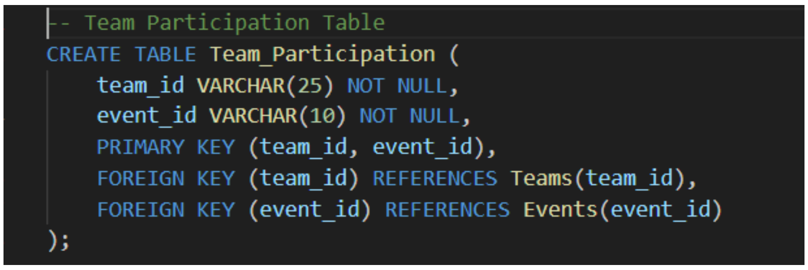
Moving towards more complex entites, the events table had multiple attributes and also had some foreign key references. The first one being the event_id giving each event a unique code. The datatype is set to be VARCHAR (10) reflecting the maximum limit of alphabets and numerics that were used in any code in the original dataset. This is followed by the discipline_code which is a FK referenced from the discplines table discussed above. Next some attributes were added to give information about the event. First being the event_date which is the original data for when that particular event took place in this year. The datatype is set as DATE to reflect the dates in the format that is accepted by MySQL. The name is stored in the event_name which is set to 150 since all names also have the discipline stated in the start. Lastly, the venue id is included as a foreign key to connect it with the venue's table. This data is spread over a total of 1035 rows reflecting almost 95% of the events that originally took place. Few of them had to be removed while doing the data cleaning process.



```sql
-- Teams Table
CREATE TABLE Teams (
    team_id VARCHAR(25) PRIMARY KEY NOT NULL,
    country_code VARCHAR(3) NOT NULL,
    FOREIGN KEY (country_code) REFERENCES Country(country_code)
);
```

Figure 6 Teams

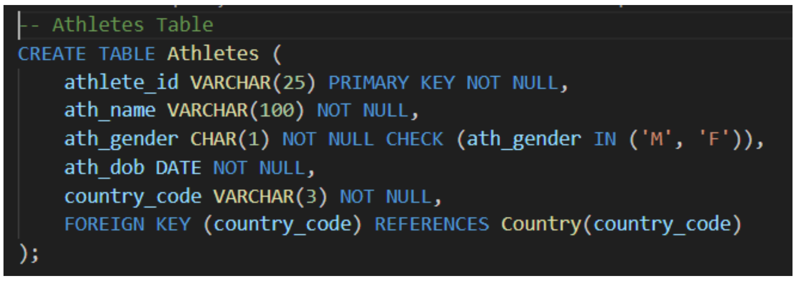
Next entity is teams which only includes information about the team_id which was fetched from the original data. The datatype set for this VARCHAR(25) because these are extracted from original data and some of them tend to be very long. The only other attribute added was the country that the team was representing. This was set as the foreign key that is being referenced from the Country table. In total, this table contains a total of 1340 Rows which are all from the original data. Without any fabrication, each row was added successfully and linked to other tables. All teams who won the medals are also included in this complete data file.



```sql
-- Team Participation Table
CREATE TABLE Team_Participation (
    team_id VARCHAR(25) NOT NULL,
    event_id VARCHAR(10) NOT NULL,
    PRIMARY KEY (team_id, event_id),
    FOREIGN KEY (team_id) REFERENCES Teams(team_id),
    FOREIGN KEY (event_id) REFERENCES Events(event_id)
);
```

Figure 7 : Team_Participation

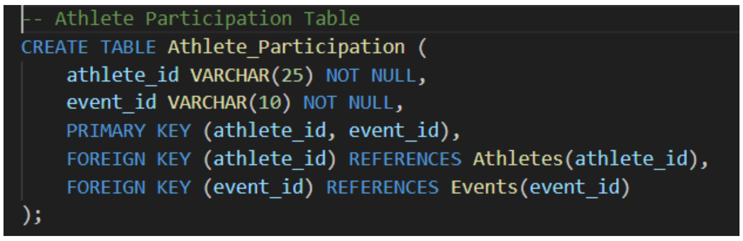
This entity is created as the relationship between Teams and Events was Many to Many and as per the accordance of Chen’s Notation, another table needs to be produced containing the primary ids of both other tables and also being referenced in the new one as forigen key. Therefore team_id and event_id were used as primary keys with the same data types to maintain referential integrity.



```sql
-- Athletes Table
CREATE TABLE Athletes (
    athlete_id VARCHAR(25) PRIMARY KEY NOT NULL,
    ath_name VARCHAR(100) NOT NULL,
    ath_gender CHAR(1) NOT NULL CHECK (ath_gender IN ('M', 'F')),
    ath_dob DATE NOT NULL,
    country_code VARCHAR(3) NOT NULL,
    FOREIGN KEY (country_code) REFERENCES Country(country_code)
);
```

Figure 8 : Athletes

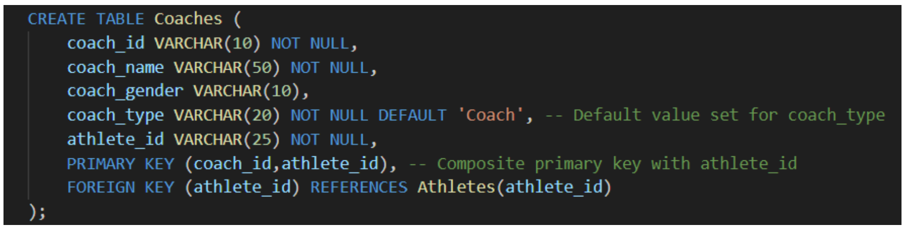
Athletes being one of the most significant entity was carefully drafted considering all the attributes that were available to enter. The first attribute is set to be athlete_id which has a VARCHAR of 25. The id itself is 7 digits long however, this was kept the same as the the team_id so that there is consistency when referencing both of them together as the winner_id in the medals table. Other information like ath_name ath_gender and ath_dob give further information about the athlete. The gender is specifically put on check so that it can only be M or F in alphabets for reducing complexity. The foreign key included in this one is the country code which is referenced from the Country table. This is by far the biggest table in all of them as it contains in total 10202 Rows. This is all original data which a few rows removed when doing data cleaning.



```sql
-- Athlete Participation Table
CREATE TABLE Athlete_Participation (
    athlete_id VARCHAR(25) NOT NULL,
    event_id VARCHAR(10) NOT NULL,
    PRIMARY KEY (athlete_id, event_id),
    FOREIGN KEY (athlete_id) REFERENCES Athletes(athlete_id),
    FOREIGN KEY (event_id) REFERENCES Events(event_id)
);
```

Figure 9 : Athlete_Participation

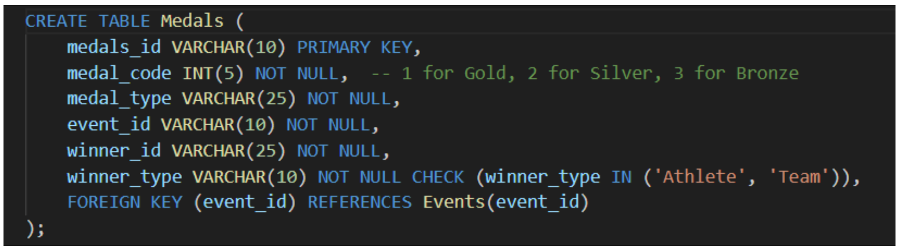
Similar to Team_Participation, Athlete_participation is also created due to Chen's notation requirement of creating another entity when there a many to many relationship. Beacause the relationship between athletes and events was M:N hence, this was created to store their ids as primary key and also reference them from their tables.



```sql
CREATE TABLE Coaches (
    coach_id VARCHAR(10) NOT NULL,
    coach_name VARCHAR(50) NOT NULL,
    coach_gender VARCHAR(10),
    coach_type VARCHAR(20) NOT NULL DEFAULT 'Coach', -- Default value set for coach_type
    athlete_id VARCHAR(25) NOT NULL,
    PRIMARY KEY (coach_id,athlete_id), -- Composite primary key with athlete_id
    FOREIGN KEY (athlete_id) REFERENCES Athletes(athlete_id)
);
```

Figure 10 : Coaches

Coaches table is showing a weak relationship between the athletes and the coaches. It contains the primary key of coach_id along with other attributes giving further information about who the coach is. The athlete_id is set as a composite primary key to give a better identity to every coach. There were multiple types of coach included in the raw data hence, Coach is set to default value if there is no other type of coach included.



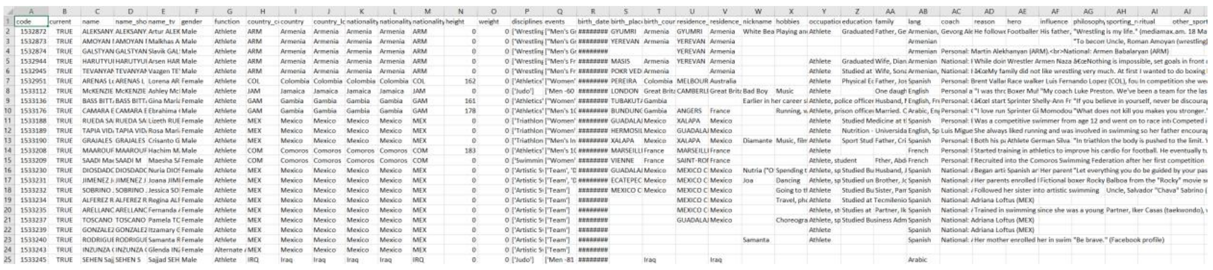
```sql
CREATE TABLE Medals (
    medals_id VARCHAR(10) PRIMARY KEY,
    medal_code INT(5) NOT NULL,  -- 1 for Gold, 2 for Silver, 3 for Bronze
    medal_type VARCHAR(25) NOT NULL,
    event_id VARCHAR(10) NOT NULL,
    winner_id VARCHAR(25) NOT NULL,
    winner_type VARCHAR(10) NOT NULL CHECK (winner_type IN ('Athlete', 'Team')),
    FOREIGN KEY (event_id) REFERENCES Events(event_id)
);
```

Figure 11 : Medals

Lastly, the medals table was created to give a sense of outcome to the overall structure. The first is the medals_id which was manually produced to give every medal a unique code. The medal code is a way to express whether it was gold, silver or bronze. Lastly, the medal type is the name written in complete letters which is why its set to VARCHAR(25). Then, it includes some foreign keys like the event_id to show a relationship between events to express which medal was given in which event. Lastly, the winner_id and type are a way to distinguish whether the winner is an athlete or a team. Both their ids can be used to find which medal they won.

Implemenation

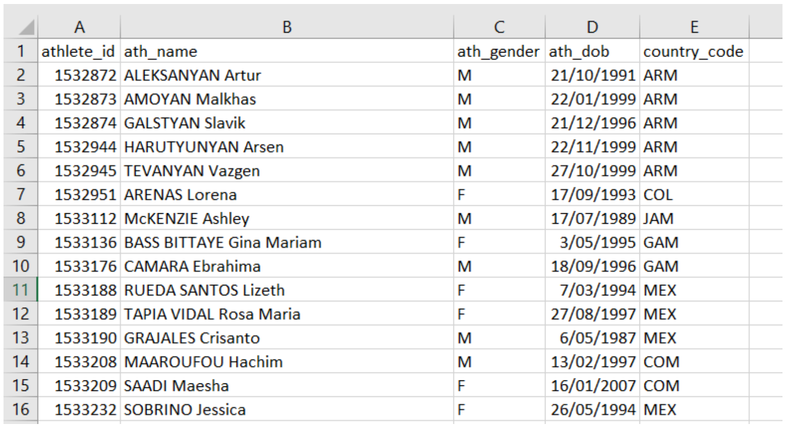
- The sole dataset used in this project is from Kaggle for Olympics 2024
- There were multiple csv files included in the zip file, however, only a relavant amount of them were selected to showcase the knowledge learnt in the unit and also to tell a complete story from the athletes/teams playing a sport till winning the medals.
- A thorough data cleaning session was done to clean each entity to keep only the relavant attributes. An example of before and after is given below for athletes data. Only few rows are reflected to show the change



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	code	current	name	name_sho	name_tv	gender	function	country_c	country	country_lc	nationality	nationality	nationality	height	weight	disciplines	events	birth_date
2	1532872	TRUE	ALEKSANY	ALEKSANY	Artur ALEK	Male	Athlete	ARM	Armenia	Armenia	Armenia	Armenia	ARM	0	0	['Wrestling	["Men's Gr	########
3	1532873	TRUE	AMOYAN I	AMOYAN I	Malkhas A	Male	Athlete	ARM	Armenia	Armenia	Armenia	Armenia	ARM	0	0	['Wrestling	["Men's Gr	########
4	1532874	TRUE	GALSTYAN	GALSTYAN	Slavik GAL	Male	Athlete	ARM	Armenia	Armenia	Armenia	Armenia	ARM	0	0	['Wrestling	["Men's Gr	########
5	1532944	TRUE	HARUTYUI	HARUTYUI	Arsen HAR	Male	Athlete	ARM	Armenia	Armenia	Armenia	Armenia	ARM	0	0	['Wrestling	["Men's Fr	########
6	1532945	TRUE	TEVANYAN	TEVANYAN	Vazgen TE	Male	Athlete	ARM	Armenia	Armenia	Armenia	Armenia	ARM	0	0	['Wrestling	["Men's Fr	########
7	1532951	TRUE	ARENAS Lc	ARENAS L	Lorena AR	Female	Athlete	COL	Colombia	Colombia	Colombia	Colombia	COL	162	0	['Athletics'	["Women'	########
8	1533112	TRUE	McKENZIE	McKENZIE	Ashley Mc	Male	Athlete	JAM	Jamaica	Jamaica	Jamaica	Jamaica	JAM	0	0	['Judo']	['Men -60	########
9	1533136	TRUE	BASS BITT	BASS BITT	Gina Mari	Female	Athlete	GAM	Gambia	Gambia	Gambia	Gambia	GAM	161	0	['Athletics'	["Women'	########
10	1533176	TRUE	CAMARA E	CAMARA E	Ebrahima (	Male	Athlete	GAM	Gambia	Gambia	Gambia	Gambia	GAM	178	0	['Athletics'	["Men's 1(	########
11	1533188	TRUE	RUEDA SA	RUEDA SA	Lizeth RUE	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Triathlon	["Women'	########
12	1533189	TRUE	TAPIA VID	TAPIA VID	Rosa Mari	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Triathlon	["Women'	########
13	1533190	TRUE	GRAJALES	GRAJALES	Crisanto G	Male	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Triathlon	["Men's In	########
14	1533208	TRUE	MAAROUF	MAAROUF	Hachim M.	Male	Athlete	COM	Comoros	Comoros	Comoros	Comoros	COM	183	0	['Athletics'	["Men's 1(	########
15	1533209	TRUE	SAADI Ma	SAADI M	Maesha S/	Female	Athlete	COM	Comoros	Comoros	Comoros	Comoros	COM	0	0	['Swimmin	["Women'	########
16	1533230	TRUE	DIOSDADC	DIOSDADC	Nuria DIOS	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team', 'D	########
17	1533231	TRUE	JIMENEZ J	JIMENEZ J	Joana JIMI	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team', 'D	########
18	1533232	TRUE	SOBRINO .	SOBRINO .	Jessica SO	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team']	########
19	1533234	TRUE	ALFEREZ R	ALFEREZ R	Regina ALF	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team']	########
20	1533235	TRUE	ARELLANC	ARELLANC	Fernanda /	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team']	########
21	1533237	TRUE	TOSCANO	TOSCANO	Pamela TC	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team']	########
22	1533239	TRUE	GONZALEZ	GONZALEZ	Itzamary E	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team']	########
23	1533240	TRUE	RODRIGUE	RODRIGUE	Samanta R	Female	Athlete	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team']	########
24	1533243	TRUE	INZUNZA (	INZUNZA (	Glenda IN	Female	Alternate /	MEX	Mexico	Mexico	Mexico	Mexico	MEX	0	0	['Artistic S	['Team']	########
25	1533245	TRUE	SEHEN Saj	SEHEN S	Sajjad SEH	Male	Athlete	IRQ	Iraq	Iraq	Iraq	Iraq	IRQ	0	0	['Judo']	['Men -81	########

	A	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ
1	code	birth_plac	birth_cour	residence_	residence_	nickname	hobbies	occupation	education	family	lang	coach	reason	hero	influence	philosophy	sporting_n	ritual	other_sport
2	1532872	GYUMRI	Armenia	GYUMRI	Armenia	White Bea	Playing an	Athlete	Graduated	Father, Ge	Armenian,	Gevorg Ale	He follows	Footballer	His father,	"Wrestling is my life." (mediamax.am. 18 Ma			
3	1532873	YEREVAN	Armenia	YEREVAN	Armenia						Armenian					"To becon	Uncle, Roman Amoyan (wrestling)		
4	1532874			YEREVAN	Armenia						Armenian	Personal: Martin Alekhanyan (ARM). National: Armen Babalaryan (ARM)							
5	1532944	MASIS	Armenia	YEREVAN	Armenia			Athlete	Graduated	Wife, Dian	Armenian	National: I	While doin	Wrestler Armen Naza		â€œNothing is impossible, set goals in front			
6	1532945	POKR VED	Armenia		Armenia			Athlete	Studied at	Wife, Sons	Armenian,	National: I	â€œMy family did not like wrestling very much. At first I wanted to do boxing						
7	1532951	PEREIRA	Colombia	MELBOUR	Australia			Athlete	Physical Ec	Father, Jos	Spanish	Personal: Brent Vallar		Race walker Luis Fernando Lopez (COL), fou		In competition she we			
8	1533112	LONDON	Great Brit	CAMBERLI	Great Brit	Bad Boy	Music	Athlete		One daugh	English	Personal a	"I was thrc	Boxer Mul	"My coach Luke Preston. We've been a team for the las				
9	1533136	TUBAKUT/	Gambia			Earlier in her career s		Athlete, police officer		Husband, I	English, Fr	Personal: (	â€œI start	Sprinter Shelly-Ann Fr		"If you believe in yourself, never be discoura			
10	1533176	BUNDUN	Gambia	ANGERS	France		Running, w	Athlete, prison officer		Married. C	Arabic, En	Personal: (	"I love run	Sprinter Gi	Momodou	"What does not kill you makes you stronger."			
11	1533188	GUADALAJ	Mexico	XALAPA	Mexico			Athlete	Studied Medicine at t		Spanish	Personal: I	Was a competitive swimmer from age 12 and went on to race int						Competed i
12	1533189	HERMOSIL	Mexico	GUADALAJ	Mexico			Athlete	Nutrition - Universida		English, Sp	Luis Migue	She always liked running and was involved in swimming so her father encoura						
13	1533190	XALAPA	Mexico	XALAPA	Mexico	Diamante	Music, film	Athlete	Sport Stud	Father, Cri	Spanish	Personal: I	Both his p	Athlete German Silva		"In triathlon the body is pushed to the limit. Y			
14	1533208	MARSEILL	France	MARSEILL	France			Athlete			French	Personal: I	Started training in athletics to improve his cardio for football. He eventually tu						
15	1533209	VIENNE	France	SAINT-RO	France			Athlete, student		Fther, Abd	French	Personal: I	Recruited into the Comoros Swimming Federation after her first competition						
16	1533230	GUADALAJ	Mexico	MEXICO C	Mexico	Nutria ("O	Spending t	Athlete, sp	Studied Bu	Husband, J	Spanish	National: /	Began arti	Spanish ar	Her parent	"Let everything you do be guided by your pas			
17	1533231	ECATEPEC	Mexico	MEXICO C	Mexico	Joa	Dancing	Athlete, sp	Studied un	Brother, Jc	Spanish	National: /	Her parents enrolled I	Fictional boxer Rocky Balboa from the "Rocky" movie s					
18	1533232	MEXICO C	Mexico	MEXICO C	Mexico		Going to tl	Athlete	Studied Bu	Sister, Pam	Spanish	National: /	Followed her sister into artistic swimming				Uncle, Salvador "Chava" Sabrino (		
19	1533234			MEXICO C	Mexico		Travel, phc	Athlete	Studied at Tecmilenio		Spanish	National: Adriana Loftus (MEX)							
20	1533235			MEXICO C	Mexico			Athlete, st	Studies at	Partner, Ik	Spanish	National: /	Trained in swimming since she was a young				Partner, Iker Casas (taekwondo),		
21	1533237			GUADALAJ	Mexico		Choreogra	Athlete, sp	Studied Business Adm		Spanish	National: Adriana Loftus (MEX)							
22	1533239							Athlete			Spanish	National: Adriana Loftus (MEX)							
23	1533240					Samanta		Athlete			Spanish	National: /	Her mother enrolled her in swim			"Be brave." (Facebook profile)			
24	1533243																		
25	1533245		Iraq		Iraq						Arabic								

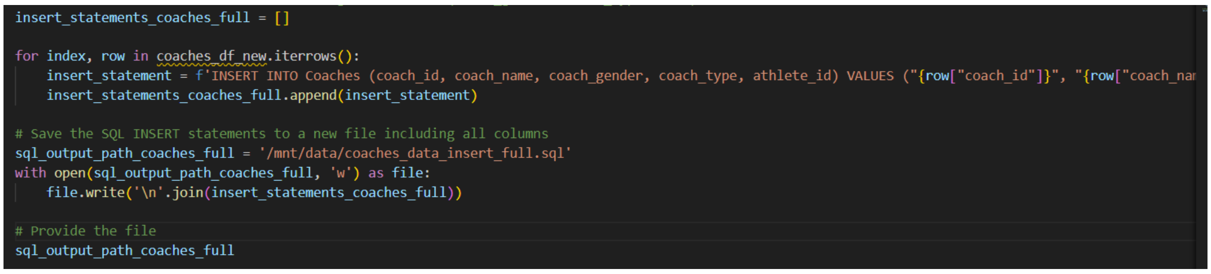
Figure 12 : Athletes data before Cleaning



	A	B	C	D	E
1	athlete_id	ath_name	ath_gender	ath_dob	country_code
2	1532872	ALEKSANYAN Artur	M	21/10/1991	ARM
3	1532873	AMOYAN Malkhas	M	22/01/1999	ARM
4	1532874	GALSTYAN Slavik	M	21/12/1996	ARM
5	1532944	HARUTYUNYAN Arsen	M	22/11/1999	ARM
6	1532945	TEVANYAN Vazgen	M	27/10/1999	ARM
7	1532951	ARENAS Lorena	F	17/09/1993	COL
8	1533112	McKENZIE Ashley	M	17/07/1989	JAM
9	1533136	BASS BITTAYE Gina Mariam	F	3/05/1995	GAM
10	1533176	CAMARA Ebrahima	M	18/09/1996	GAM
11	1533188	RUEDA SANTOS Lizeth	F	7/03/1994	MEX
12	1533189	TAPIA VIDAL Rosa Maria	F	27/08/1997	MEX
13	1533190	GRAJALES Crisanto	M	6/05/1987	MEX
14	1533208	MAAROUFOU Hachim	M	13/02/1997	COM
15	1533209	SAADI Maesha	F	16/01/2007	COM
16	1533232	SOBRINO Jessica	F	26/05/1994	MEX

Figure 13 : Athletes data after Cleaning

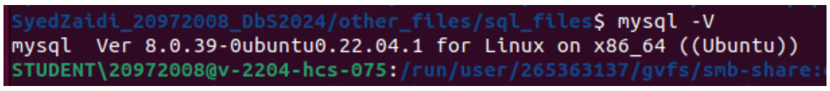
- In order to ensure referential integrity, the use of conditional formatting on excel was thoroughly done to see if there are any duplicates in the data, specially for entities like countries, venues and disciplines etc so that they can be individually linked to every athlete/team.
- Furthermore, wherever data was not available, Vlookup was used as an Excel formula to fetch data from different files of CSV. For the discipline code being fetched from the discipline table as the athletes table was only showing discipline_names.
- Once the data was completely cleaned and linked together, python was used to write scripts inorder to to add insert into commands as a sql file so that they can be sourced directly into mysql. A screenshot of the python script is shown below for coach data.



```
insert_statements_coaches_full = []

for index, row in coaches_df_new.iterrows():
    insert_statement = f'INSERT INTO Coaches (coach_id, coach_name, coach_gender, coach_type, athlete_id) VALUES ("{row["coach_id"]}", "{row["coach_nar
    insert_statements_coaches_full.append(insert_statement)

# Save the SQL INSERT statements to a new file including all columns
sql_output_path_coaches_full = '/mnt/data/coaches_data_insert_full.sql'
with open(sql_output_path_coaches_full, 'w') as file:
    file.write('\n'.join(insert_statements_coaches_full))

# Provide the file
sql_output_path_coaches_full
```

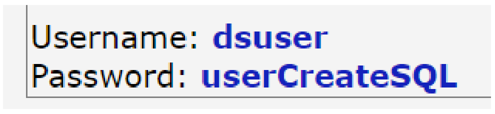
Figure 14 : Python scripts used to amend insert function for Sql files

- Once all insert files were created alongside the tables file, these were taken to VMware and used the following Linux and Mysql version to import it into the server mentioned on Blackboard.



```
SyedZaidi_20972008_DbS2024/other_files/sql_files$ mysql -V
mysql  Ver 8.0.39-0ubuntu0.22.04.1 for Linux on x86_64 ((Ubuntu))
STUDENT\20972008@v-2204-hcs-075:/run/user/265363137/gvfs/smb-share:
```

Figure 15 : MySQL version and Linux

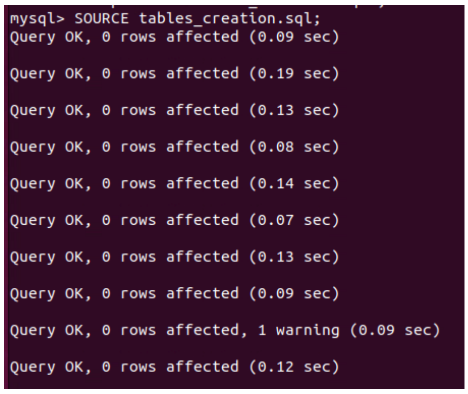


Username: **dsuser**
Password: **userCreateSQL**

Figure 16 : Credentials used for Server Login

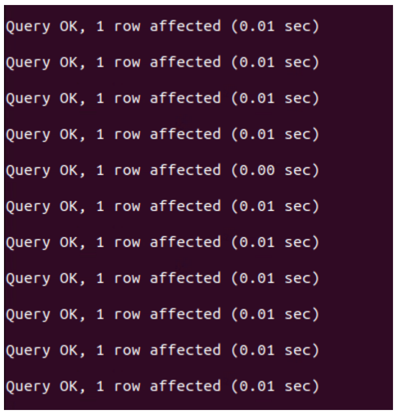
- The following commands were used to create a database under my name and then source all the tables and insert files into the server. (Explained in detail in the UserGuide)

SOURCE mydatabase.sql;
SOURCE tables_creation.sql;
SOURCE insert_all_data.sql;



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

Query OK, 0 rows affected (0.19 sec)

Query OK, 0 rows affected (0.13 sec)

Query OK, 0 rows affected (0.08 sec)

Query OK, 0 rows affected (0.14 sec)

Query OK, 0 rows affected (0.07 sec)

Query OK, 0 rows affected (0.13 sec)

Query OK, 0 rows affected (0.09 sec)

Query OK, 0 rows affected, 1 warning (0.09 sec)

Query OK, 0 rows affected (0.12 sec)
```

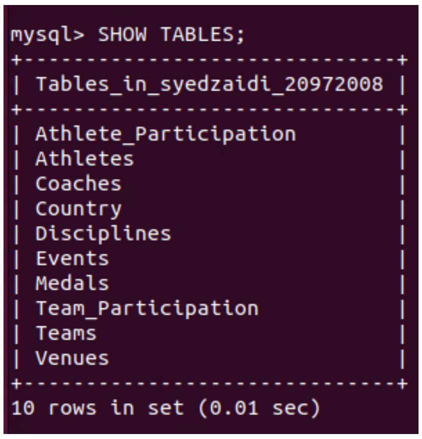
Figure 17 : Results after sourcing Tables;



```
Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)

Query OK, 1 row affected (0.01 sec)
```

Figure 18 : Results after Sourcing All inserts

- It takes a long time (Around 5 mins on VMware) to source thousands of rows, but once done the command of SHOW TABLES; shows the following results.



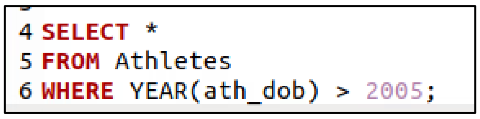
```
mysql> SHOW TABLES;
+-----------------------------+
| Tables_in_syedzaidi_20972008 |
+-----------------------------+
| Athlete_Participation        |
| Athletes                     |
| Coaches                      |
| Country                      |
| Disciplines                  |
| Events                       |
| Medals                       |
| Team_Participation           |
| Teams                        |
| Venues                       |
+-----------------------------+
10 rows in set (0.01 sec)
```

Figure 19 : Tables structure after sourcing

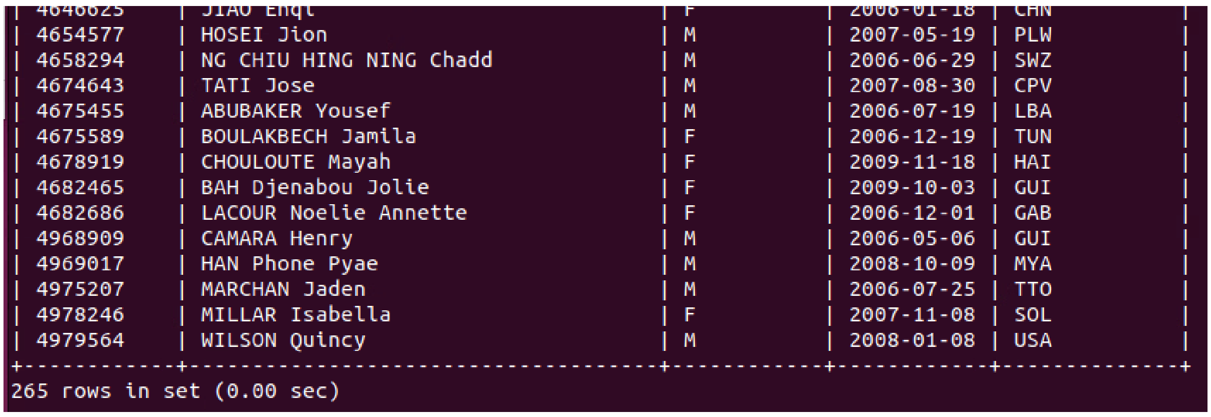
Use of the Database

Basic SQL Queries

Query 1 : Which athletes competed in the Olympics 2024 that were born after 2005?
The query intends to help in identifying younger generation of athletes that competed in the last Olympics to understand the trends that may follow up in the future with more younger athletes competiting internationally. The following query finds all the athletes who were born in after the year 2025.



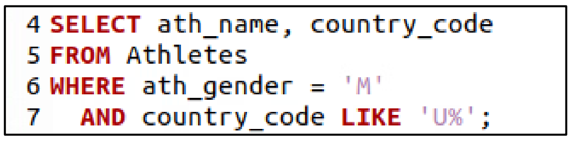
```sql
SELECT *
FROM Athletes
WHERE YEAR(ath_dob) > 2005;
```



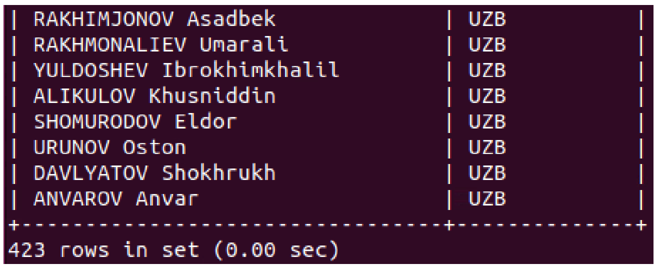
```
| 4646625     | JIAO Enqi                           | F          | 2006-01-18 | CHN          |
| 4654577     | HOSEI Jion                          | M          | 2007-05-19 | PLW          |
| 4658294     | NG CHIU HING NING Chadd             | M          | 2006-06-29 | SWZ          |
| 4674643     | TATI Jose                           | M          | 2007-08-30 | CPV          |
| 4675455     | ABUBAKER Yousef                     | M          | 2006-07-19 | LBA          |
| 4675589     | BOULAKBECH Jamila                   | F          | 2006-12-19 | TUN          |
| 4678919     | CHOULOUTE Mayah                     | F          | 2009-11-18 | HAI          |
| 4682465     | BAH Djenabou Jolie                  | F          | 2009-10-03 | GUI          |
| 4682686     | LACOUR Noelie Annette               | F          | 2006-12-01 | GAB          |
| 4968909     | CAMARA Henry                        | M          | 2006-05-06 | GUI          |
| 4969017     | HAN Phone Pyae                      | M          | 2008-10-09 | MYA          |
| 4975207     | MARCHAN Jaden                       | M          | 2006-07-25 | TTO          |
| 4978246     | MILLAR Isabella                     | F          | 2007-11-08 | SOL          |
| 4979564     | WILSON Quincy                       | M          | 2008-01-08 | USA          |
+-------------+-------------------------------------+------------+------------+--------------+
265 rows in set (0.00 sec)
```

The results show 265 rows, showing the information of the all these employees alongside their dob. This can help to track emerging talents and also understand which countries are focusing on sending younger athletes.

Query 2 : Which athletes were male from countries whose codes start with 'U'?
The query was used to test and extract the athletes that are national of countries with codes starting with U. It can be useful when finding relevant information about a country like Ukraine or Uzbekistan, etc.



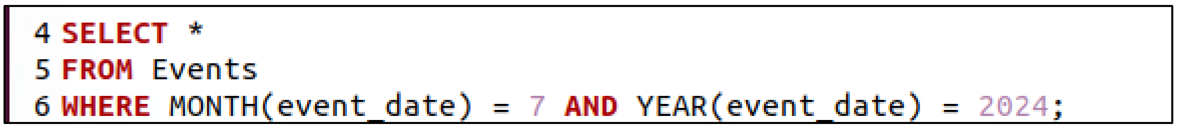
```sql
SELECT ath_name, country_code
FROM Athletes
WHERE ath_gender = 'M'
  AND country_code LIKE 'U%';
```



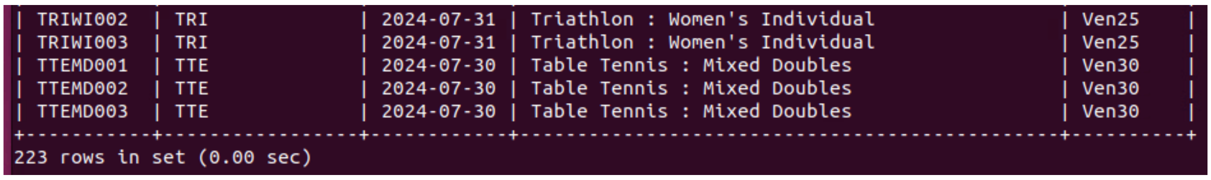
```
| RAKHIMJONOV Asadbek                | UZB          |
| RAKHMONALIEV Umarali               | UZB          |
| YULDOSHEV Ibrokhimkhalil           | UZB          |
| ALIKULOV Khusniddin                | UZB          |
| SHOMURODOV Eldor                   | UZB          |
| URUNOV Oston                       | UZB          |
| DAVLYATOV Shokhrukh                | UZB          |
| ANVAROV Anvar                      | UZB          |
+------------------------------------+--------------+
423 rows in set (0.00 sec)
```

The query returned 423 rows listing the male athletes that are representing countries with their code starting with ‘U’. This can be helpful when analysing information about demographics of particular country.

Query 3 : What are the events in Olympics that occurred in July 2024?
The query is a way to retrieve all the events that occurred in july by filtering the event date. This is vital information for carrying out tasks like scheduling, logistics planning and more.



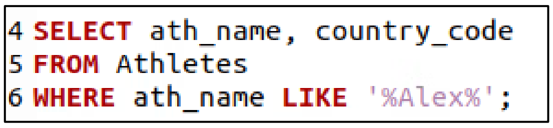
```sql
SELECT *
FROM Events
WHERE MONTH(event_date) = 7 AND YEAR(event_date) = 2024;
```



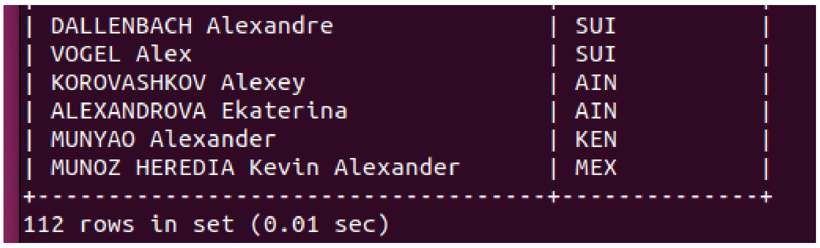
```
| TRIWI002  | TRI            | 2024-07-31 | Triathlon : Women's Individual                       | Ven25    |
| TRIWI003  | TRI            | 2024-07-31 | Triathlon : Women's Individual                       | Ven25    |
| TTEMD001  | TTE            | 2024-07-30 | Table Tennis : Mixed Doubles                         | Ven30    |
| TTEMD002  | TTE            | 2024-07-30 | Table Tennis : Mixed Doubles                         | Ven30    |
| TTEMD003  | TTE            | 2024-07-30 | Table Tennis : Mixed Doubles                         | Ven30    |
+-----------+----------------+------------+------------------------------------------------------+----------+
223 rows in set (0.00 sec)
```

The results returned 233 rows showing multiple events including table tennis, triathlon, swimming etc. The output helps understand the events and their distribution across all available venues.

Query 4 : Retrieve all athletes whose name contains the string ‘Alex’?
This question wanted to find out how many athletes are there that have Alex in their names. This is because it’s a very used name across multiple countries and finding such data can be usefull when carrying out name-based analysis or understanding cultural names of different countries.



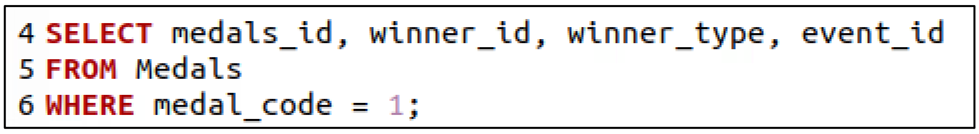
```sql
SELECT ath_name, country_code
FROM Athletes
WHERE ath_name LIKE '%Alex%';
```



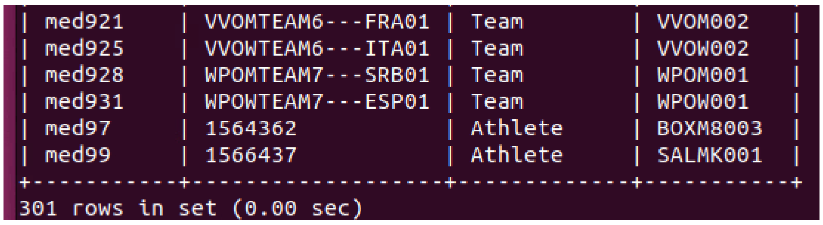
```
| DALLENBACH Alexandre              | SUI          |
| VOGEL Alex                        | SUI          |
| KOROVASHKOV Alexey                | AIN          |
| ALEXANDROVA Ekaterina             | AIN          |
| MUNYAO Alexander                  | KEN          |
| MUNOZ HEREDIA Kevin Alexander     | MEX          |
+-----------------------------------+--------------+
112 rows in set (0.01 sec)
```

Out of thousands of athletes around 1% of them have name of Alex within them. 112 rows show athletes with their full names and also provides information about their countries.

Query 5 : Give details of all gold medalists.
The questions wished to know information about the athletes and teams that were able to score the top results. This helps in identifying the top performers of each event. The following query does that and also gives further information about the details of the medal.



```sql
SELECT medals_id, winner_id, winner_type, event_id
FROM Medals
WHERE medal_code = 1;
```



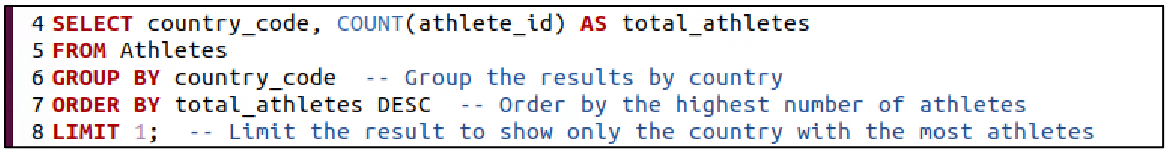
```
| med921     | VVOMTEAM6---FRA01 | Team         | VVOM002   |
| med925     | VVOWTEAM6---ITA01 | Team         | VVOW002   |
| med928     | WPOMTEAM7---SRB01 | Team         | WPOM001   |
| med931     | WPOWTEAM7---ESP01 | Team         | WPOW001   |
| med97      | 1564362           | Athlete      | BOXM8003  |
| med99      | 1566437           | Athlete      | SALMK001  |
+-----------+-------------------+-------------+-----------+
301 rows in set (0.00 sec)
```

The results show a total of 301 gold medals that were given out in different events. It shows their type if they are individual or team and also gives information about the event_id that they won in.

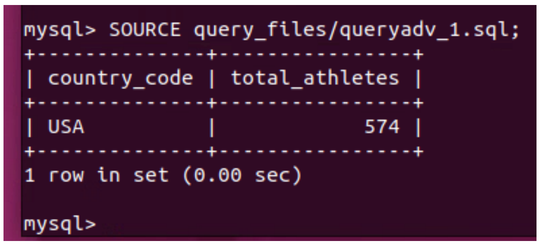
Advanced SQL Queries

Query 1 : Which country has the most athletes participating in 2024 Olympcis?

The questions wants to find only one country that has the most participants from. This can help compare with other years or countries to see if the trends have changed or to compare the number of participants.



```sql
SELECT country_code, COUNT(athlete_id) AS total_athletes
FROM Athletes
GROUP BY country_code  -- Group the results by country
ORDER BY total_athletes DESC  -- Order by the highest number of athletes
LIMIT 1;  -- Limit the result to show only the country with the most athletes
```

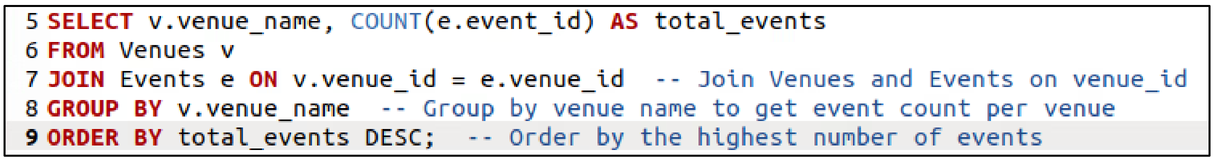


```
mysql> SOURCE query_files/queryadv_1.sql;
+--------------+----------------+
| country_code | total_athletes |
+--------------+----------------+
| USA          |            574 |
+--------------+----------------+
1 row in set (0.00 sec)

mysql>
```

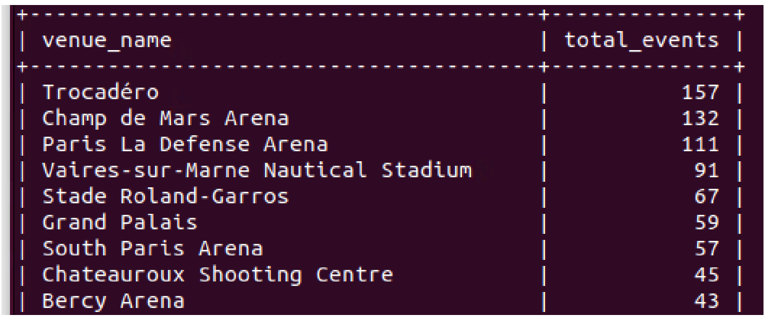
Since the limit was imposed to 1, the results only show the code of one country and the total count of athletes. USA being the one on top with 574 athletes competing in Olympics 2024.

Query 2 : Find the total number of events that took place in each venue

The question aims to find the quantiy of events that took place at each venue. This requires a join between the venue and events table to and use the aggregate function to find the count of events happening at a particular venue.



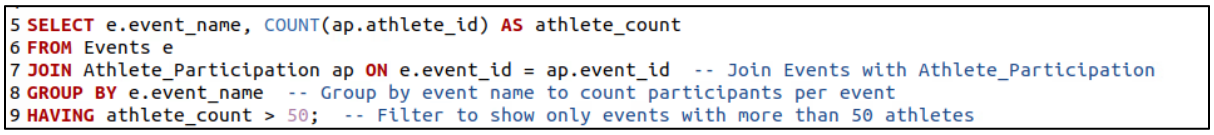
```sql
SELECT v.venue_name, COUNT(e.event_id) AS total_events
FROM Venues v
JOIN Events e ON v.venue_id = e.venue_id  -- Join Venues and Events on venue_id
GROUP BY v.venue_name  -- Group by venue name to get event count per venue
ORDER BY total_events DESC;  -- Order by the highest number of events
```



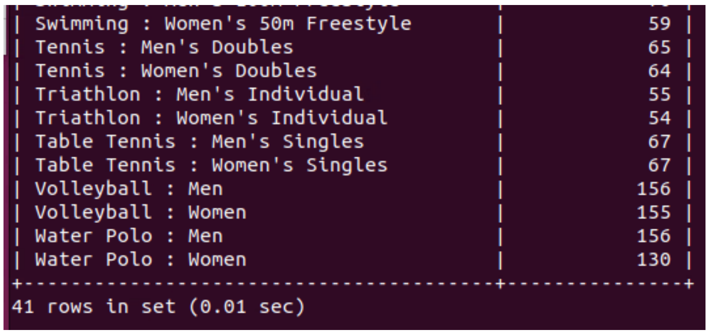
```
+--------------------------------------+---------------+
| venue_name                           | total_events  |
+--------------------------------------+---------------+
| Trocadéro                            |           157 |
| Champ de Mars Arena                  |           132 |
| Paris La Defense Arena               |           111 |
| Vaires-sur-Marne Nautical Stadium    |            91 |
| Stade Roland-Garros                  |            67 |
| Grand Palais                         |            59 |
| South Paris Arena                    |            57 |
| Chateauroux Shooting Centre          |            45 |
| Bercy Arena                          |            43 |
```

The cropped screenshot shows the top venues that had multiple events for them. The highest number of events took place in Trocadero with 157 games being played. Such outputs can assist in logistical planning and also understanding which venues can be further used in the future for high demand events.

Query 3 : Find the events with more than 50 Athletes participating
The question wants to find the count of events that have around 50 athletes participating in them. This should be done through the aggregate function of count using the Having function in after they have been grouped together according to the event name.



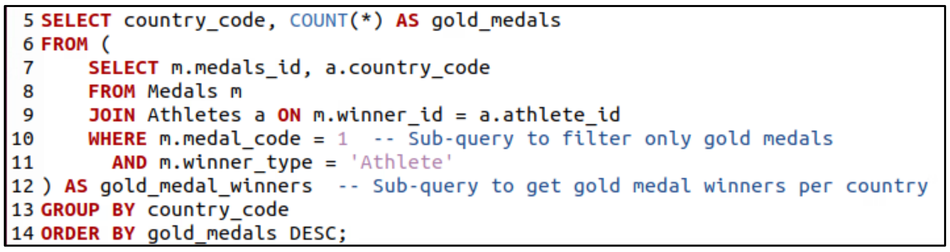
```sql
SELECT e.event_name, COUNT(ap.athlete_id) AS athlete_count
FROM Events e
JOIN Athlete_Participation ap ON e.event_id = ap.event_id  -- Join Events with Athlete_Participation
GROUP BY e.event_name  -- Group by event name to count participants per event
HAVING athlete_count > 50;  -- Filter to show only events with more than 50 athletes
```



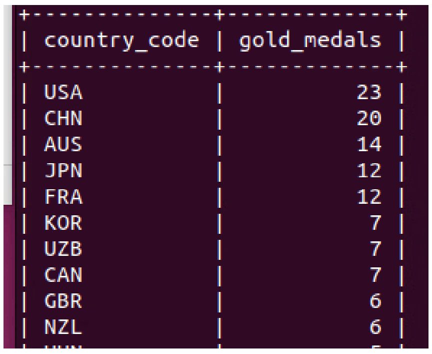
```
| Swimming : Women's 50m Freestyle     |            59 |
| Tennis : Men's Doubles               |            65 |
| Tennis : Women's Doubles             |            64 |
| Triathlon : Men's Individual         |            55 |
| Triathlon : Women's Individual       |            54 |
| Table Tennis : Men's Singles         |            67 |
| Table Tennis : Women's Singles       |            67 |
| Volleyball : Men                     |           156 |
| Volleyball : Women                   |           155 |
| Water Polo : Men                     |           156 |
| Water Polo : Women                   |           130 |
+--------------------------------------+---------------+
41 rows in set (0.01 sec)
```

A screenshot of the results shows around 41 different events that had more than 50 athletes playing. The majority of them are teams playing games like basketball and football, while others are those that are very famous, like swimming and table tennis. The outputs can be helpful for event managers as they can use the information to carry out sporting arrangements.

Query 4 : How many Gold medals have each country won?
This is an evolved version of the basic query understanding all athletes winning gold medals. This one requires to join the table of country along side do aggregate functions of counting the medals. Furthermore, it is required to do a sub query to find the narrowed medalists for country.



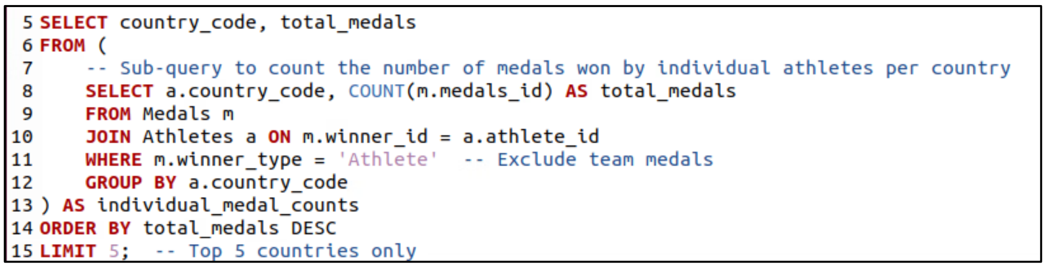
```sql
SELECT country_code, COUNT(*) AS gold_medals
FROM (
    SELECT m.medals_id, a.country_code
    FROM Medals m
    JOIN Athletes a ON m.winner_id = a.athlete_id
    WHERE m.medal_code = 1  -- Sub-query to filter only gold medals
      AND m.winner_type = 'Athlete'
) AS gold_medal_winners  -- Sub-query to get gold medal winners per country
GROUP BY country_code
ORDER BY gold_medals DESC;
```



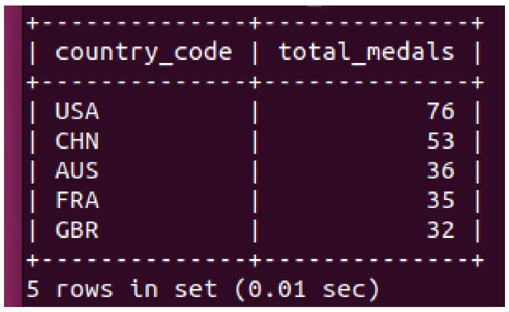
```
+--------------+-------------+
| country_code | gold_medals |
+--------------+-------------+
| USA          |          23 |
| CHN          |          20 |
| AUS          |          14 |
| JPN          |          12 |
| FRA          |          12 |
| KOR          |           7 |
| UZB          |           7 |
| CAN          |           7 |
| GBR          |           6 |
| NZL          |           6 |
```

The results show that USA is on the top with 23 Medals which is followed by China and Australia with 20 and 14 respectively. This output can be valuable when understanding the ranking of countries and knowing their success in the latest competition of Olympics.

Query 5 : Find the top 5 countries with the most medals, excluding team winners.
The query is something similar to what was done in query 4, except that in this, all medals are required, and also it is asked to exclude any medals that are won by teams. For this, once again an aggregate function of the count will be used. Also, subqueries will be to narrow down medal information with limtis on the number of countries being displayed.



```sql
SELECT country_code, total_medals
FROM (
    -- Sub-query to count the number of medals won by individual athletes per country
    SELECT a.country_code, COUNT(m.medals_id) AS total_medals
    FROM Medals m
    JOIN Athletes a ON m.winner_id = a.athlete_id
    WHERE m.winner_type = 'Athlete'  -- Exclude team medals
    GROUP BY a.country_code
) AS individual_medal_counts
ORDER BY total_medals DESC
LIMIT 5;  -- Top 5 countries only
```



```
+--------------+--------------+
| country_code | total_medals |
+--------------+--------------+
| USA          |           76 |
| CHN          |           53 |
| AUS          |           36 |
| FRA          |           35 |
| GBR          |           32 |
+--------------+--------------+
5 rows in set (0.01 sec)
```

From the results we can see that USA, China and Aus are still at the top. However, after that the results do change since it means Japan won more gold medals than all medals followed by Great Britain. Such analysis can once again help in comparing the performance of countries as a whole to understand the trends and predict future winners.

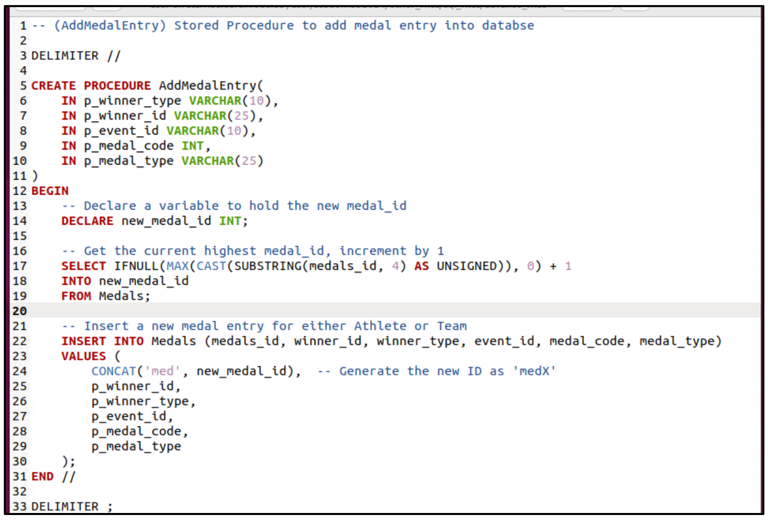
Advance Concepts

Procedure 1 – AddMedalEntry

Description : This is a stored procedure created to add a new medal entry into the database. It requires the use of the original winner_id which can either be a team or an individual athlete along side the event_id in which they won the medal. This way, a new medal could be entered to update the progress of Olympcis 2024.

Use: The use of this stored procedure ultimately simplifies and also makes it easy to another another entry into the database. Rather than updating the CSV file and then importing it all together, just by calling this procedure, we can update the database with the latest winner information. The medal_id is generated uniquely on the basis of incremental technique so that every time a new medal is added it gives it an incremented id.

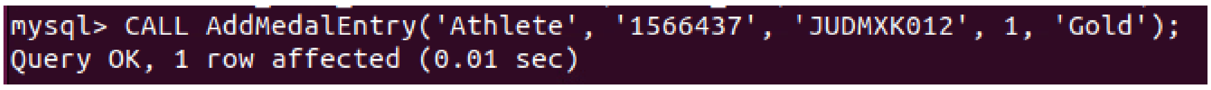
Evidence: First step is to source the procedure into the database by the following code (written in detail in UserGuide). The code is shown below for procedure_1.



```sql
-- (AddMedalEntry) Stored Procedure to add medal entry into databse

DELIMITER //

CREATE PROCEDURE AddMedalEntry(
    IN p_winner_type VARCHAR(10),
    IN p_winner_id VARCHAR(25),
    IN p_event_id VARCHAR(10),
    IN p_medal_code INT,
    IN p_medal_type VARCHAR(25)
)
BEGIN
    -- Declare a variable to hold the new medal_id
    DECLARE new_medal_id INT;

    -- Get the current highest medal_id, increment by 1
    SELECT IFNULL(MAX(CAST(SUBSTRING(medals_id, 4) AS UNSIGNED)), 0) + 1
    INTO new_medal_id
    FROM Medals;

    -- Insert a new medal entry for either Athlete or Team
    INSERT INTO Medals (medals_id, winner_id, winner_type, event_id, medal_code, medal_type)
    VALUES (
        CONCAT('med', new_medal_id),  -- Generate the new ID as 'medX'
        p_winner_id,
        p_winner_type,
        p_event_id,
        p_medal_code,
        p_medal_type
    );
END //

DELIMITER ;
```

Figure 20 : Code for Procedure 1

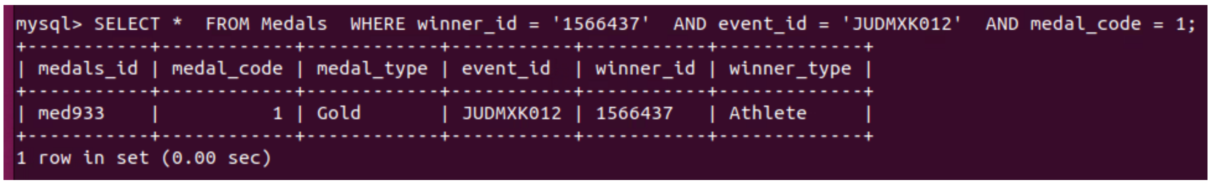
Then we call the function by adding in appropriate values to the fields. This should include the correct winner_id which can be athlete_id or team_id and also the correct event_id in which they won a medal. An example is shown below for the athelete BONTUS Valentin who won a Gold in Judo Men.



```
mysql> CALL AddMedalEntry('Athlete', '1566437', 'JUDMXK012', 1, 'Gold');
Query OK, 1 row affected (0.01 sec)
```

Figure 21 : Resuls of Procedure 1 being Called

We can run a select query to filter the relevant columns to confirm that the new medal has been inserted.



```
mysql> SELECT *  FROM Medals  WHERE winner_id = '1566437'  AND event_id = 'JUDMXK012'  AND medal_code = 1;
+-----------+------------+------------+-----------+-----------+-------------+
| medals_id | medal_code | medal_type | event_id  | winner_id | winner_type |
+-----------+------------+------------+-----------+-----------+-------------+
| med933    |          1 | Gold       | JUDMXK012 | 1566437   | Athlete     |
+-----------+------------+------------+-----------+-----------+-------------+
1 row in set (0.00 sec)
```

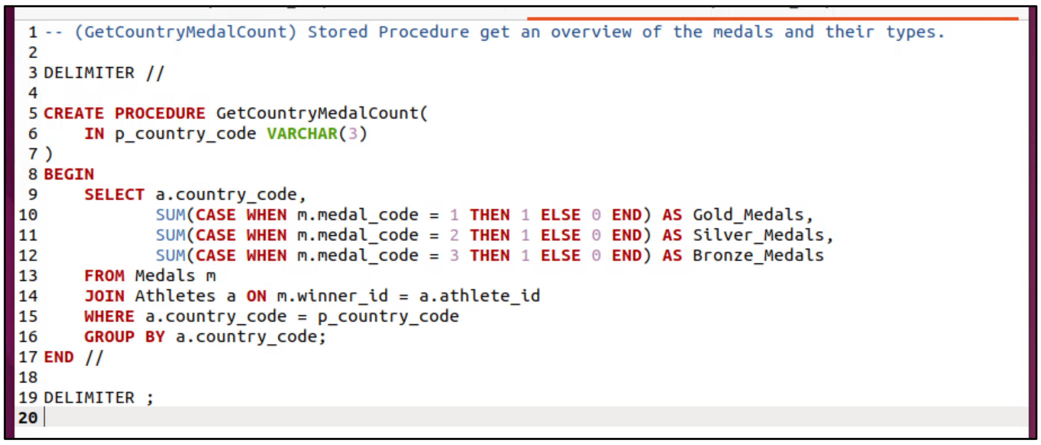
Figure 22 : Select Statement for Confirmation

Procedure 2 – GetCountryMedalCount

Description: This particular stored procedure is intended to retrieve a summary of the amount of medals a country has won and also the associated type for it. It returns an overview of the quantity of medals won in the Olympics 2024.

Use: The procedure simplifies the process of figuring out each type of medal that was won by the country. This makes it easy to understand how a particular country has performed over this Olympic without needing to write manual SQL scripts. This allows comparison between countries aur may even be used with historical data to see any trends in the country's overall performance.

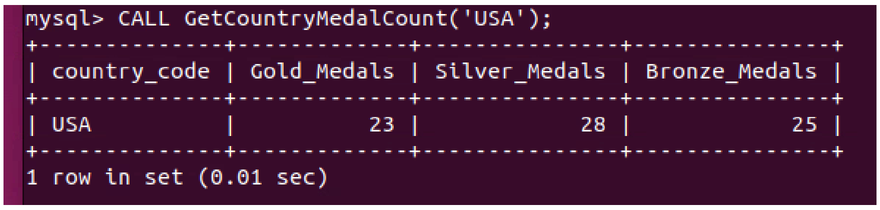
Evidence: First step is to use the provided code in the UserGuide to source the procedure into the database. The code is for the procedure is given below.



```
-- (GetCountryMedalCount) Stored Procedure get an overview of the medals and their types.

DELIMITER //

CREATE PROCEDURE GetCountryMedalCount(
    IN p_country_code VARCHAR(3)
)
BEGIN
    SELECT a.country_code,
           SUM(CASE WHEN m.medal_code = 1 THEN 1 ELSE 0 END) AS Gold_Medals,
           SUM(CASE WHEN m.medal_code = 2 THEN 1 ELSE 0 END) AS Silver_Medals,
           SUM(CASE WHEN m.medal_code = 3 THEN 1 ELSE 0 END) AS Bronze_Medals
    FROM Medals m
    JOIN Athletes a ON m.winner_id = a.athlete_id
    WHERE a.country_code = p_country_code
    GROUP BY a.country_code;
END //

DELIMITER ;
```

Figure 23 : Code for Procedure 2

Onces it has been sourced, it can be called using any country_code to find a summary of the results they have produced over the entire event. An example of United States of America (USA) is used to represent how the results look. This can be seen in the figure below.



```
mysql> CALL GetCountryMedalCount('USA');
+--------------+-------------+---------------+---------------+
| country_code | Gold_Medals | Silver_Medals | Bronze_Medals |
+--------------+-------------+---------------+---------------+
| USA          |          23 |            28 |            25 |
+--------------+-------------+---------------+---------------+
1 row in set (0.01 sec)
```

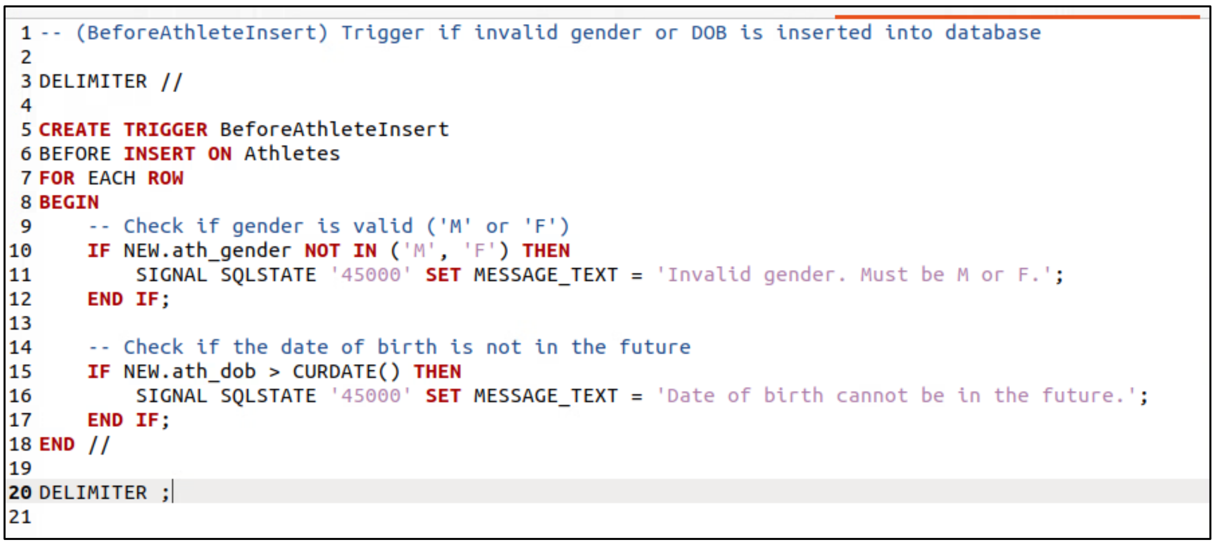
Figure 24 : Results of Procedure 2 being Called

Trigger 1 – BeforeAthleteInsert

Description: The intention of this trigger was to stop letting anyone insert information about the athlete in the wrong format. This is to validate certain fields before any records are entered in the athletes table. The focus is on the gender which can either be M for Males and F for Females. And the other focus is on the date of birth not being higher than the current date.

Use: The trigger ensures data integrity as be enforcing business rules at the database level, it prevents lots of future complications as there will not be any incorrect entries being made from the start. This allows the creation of clean and accurate data that has potential of being tampered by human errors when entering into database.

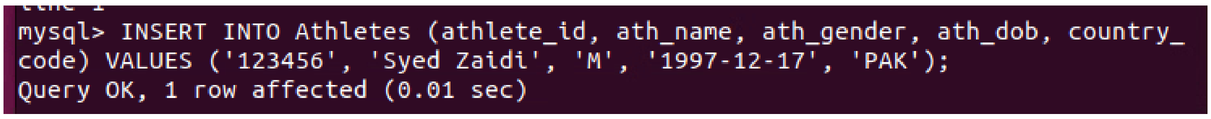
Evidence: First step is to source the trigger into the database which can be read in more detail on how to do it through the UserGuide. The following figure shows a picture of the code.



```sql
-- (BeforeAthleteInsert) Trigger if invalid gender or DOB is inserted into database

DELIMITER //

CREATE TRIGGER BeforeAthleteInsert
BEFORE INSERT ON Athletes
FOR EACH ROW
BEGIN
    -- Check if gender is valid ('M' or 'F')
    IF NEW.ath_gender NOT IN ('M', 'F') THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid gender. Must be M or F.';
    END IF;

    -- Check if the date of birth is not in the future
    IF NEW.ath_dob > CURDATE() THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Date of birth cannot be in the future.';
    END IF;
END //

DELIMITER ;
```

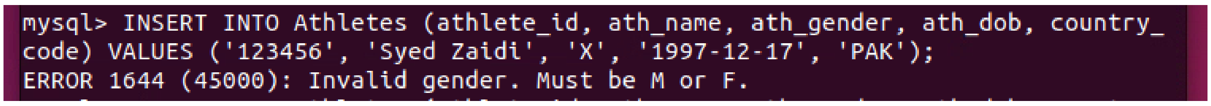
Figure 25 : Code for Trigger 1

Once it has been sourced, we can test it by adding different entries just to see if it works or not. It should let the write entry to pass but if there is any issues with gender or Dob it should give us the error.



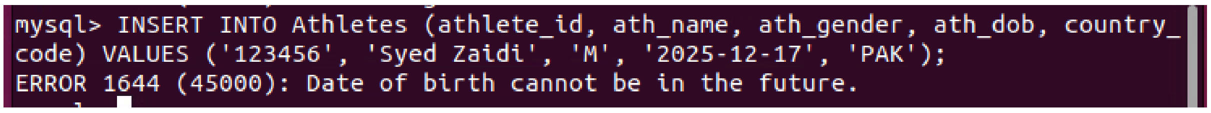
```
mysql> INSERT INTO Athletes (athlete_id, ath_name, ath_gender, ath_dob, country_
code) VALUES ('123456', 'Syed Zaidi', 'M', '1997-12-17', 'PAK');
Query OK, 1 row affected (0.01 sec)
```

Figure 26 : Inserting with Correct information



```
mysql> INSERT INTO Athletes (athlete_id, ath_name, ath_gender, ath_dob, country_
code) VALUES ('123456', 'Syed Zaidi', 'X', '1997-12-17', 'PAK');
ERROR 1644 (45000): Invalid gender. Must be M or F.
```

Figure 27 : Inserting with wrong Gender



```
mysql> INSERT INTO Athletes (athlete_id, ath_name, ath_gender, ath_dob, country_
code) VALUES ('123456', 'Syed Zaidi', 'M', '2025-12-17', 'PAK');
ERROR 1644 (45000): Date of birth cannot be in the future.
```

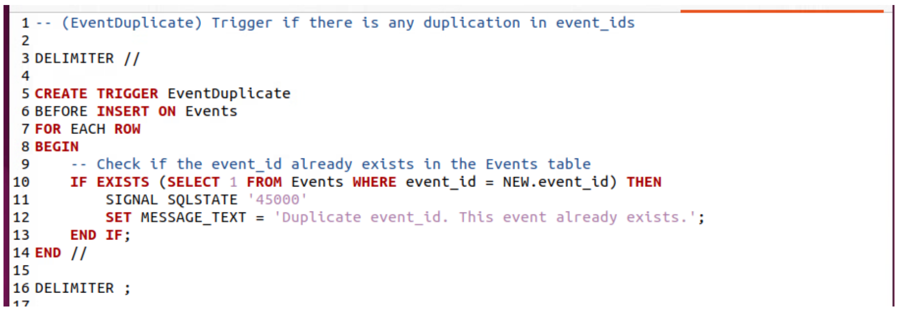
Figure 28 : Inserting with Wrong Date of Birth

Trigger 2 – EventDuplicate

Description: This trigger is way to systemise how data is being entered into the events table. From the dataset, it's understood that the event table is one of the most important ones, as it is connected to the majority of other entities. Hence, to be extra sure of its data validation, this trigger gives out an error if incase there is any insertion of a event_id duplicate.

Use: This trigger essentially maintains data integrity of the events table along side other tables connected to it by preventing any duplicates in the event_id attribute. This guarentees that all event_ids are unique and hence avoids any potential data conflicts in the future.

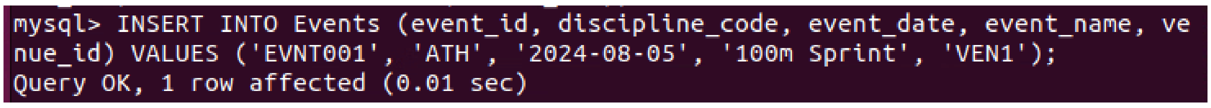
Evidence: First step again is to source the trigger by using the code written in the UserGuide. The code is as follows.



```
-- (EventDuplicate) Trigger if there is any duplication in event_ids

DELIMITER //

CREATE TRIGGER EventDuplicate
BEFORE INSERT ON Events
FOR EACH ROW
BEGIN
    -- Check if the event_id already exists in the Events table
    IF EXISTS (SELECT 1 FROM Events WHERE event_id = NEW.event_id) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Duplicate event_id. This event already exists.';
    END IF;
END //

DELIMITER ;
```

Figure 29 : Code for Trigger 2

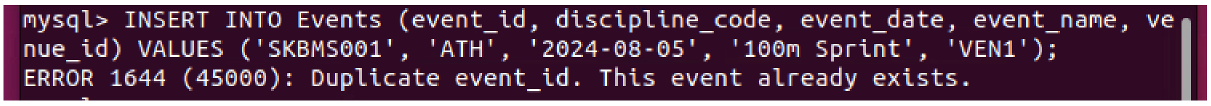
We can test this trigger by first inserting an event with no previous id available so that a new row can be created to represent that particular event.



```
mysql> INSERT INTO Events (event_id, discipline_code, event_date, event_name, ve
nue_id) VALUES ('EVNT001', 'ATH', '2024-08-05', '100m Sprint', 'VEN1');
Query OK, 1 row affected (0.01 sec)
```

Figure 30 : Inserting with a unique event_id

The row has been successfully entered as there is no other event_id with ‘EVNT001’. However, if we insert an id with a previous one available it should give an error. For example purposes, an id for Skateboarding for Men is already available with the id of “SKBMS001”. If we enter the same one the result are as follows.



```
mysql> INSERT INTO Events (event_id, discipline_code, event_date, event_name, ve
nue_id) VALUES ('SKBMS001', 'ATH', '2024-08-05', '100m Sprint', 'VEN1');
ERROR 1644 (45000): Duplicate event_id. This event already exists.
```

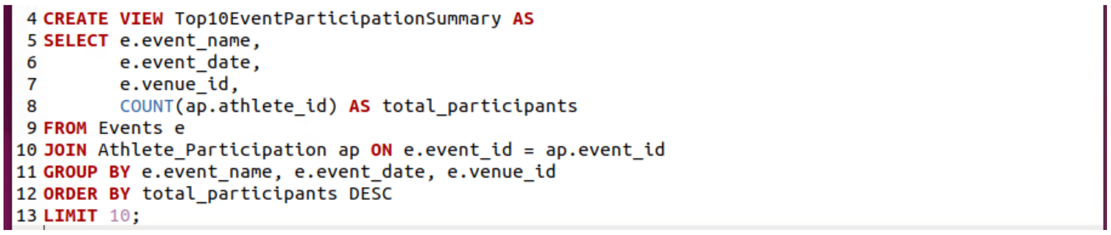
Figure 31 : Inserting with a duplicate event_id

Views 1 - Top10EventParticipationSummary

Description: The view is created as additional advance function to retrieve that data related to the top 10 events with the highest number of athletes playing.

Use: This is a quick an easy way to understand the sports with highest amount of participation and to quickly identify which one is the most popular one among athletes.

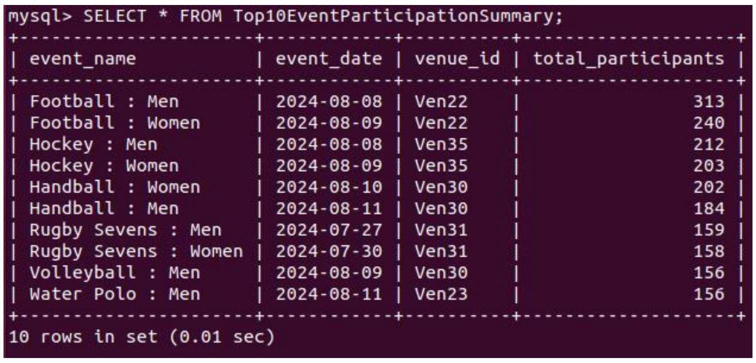
Evidence: First step is to source the view through the code description given in UserGuide. The screenshot of the code is below



```sql
CREATE VIEW Top10EventParticipationSummary AS
SELECT e.event_name,
       e.event_date,
       e.venue_id,
       COUNT(ap.athlete_id) AS total_participants
FROM Events e
JOIN Athlete_Participation ap ON e.event_id = ap.event_id
GROUP BY e.event_name, e.event_date, e.venue_id
ORDER BY total_participants DESC
LIMIT 10;
```

Figure 32 : Code for View 1

Now once we call this view through a select statements the results would show something like this

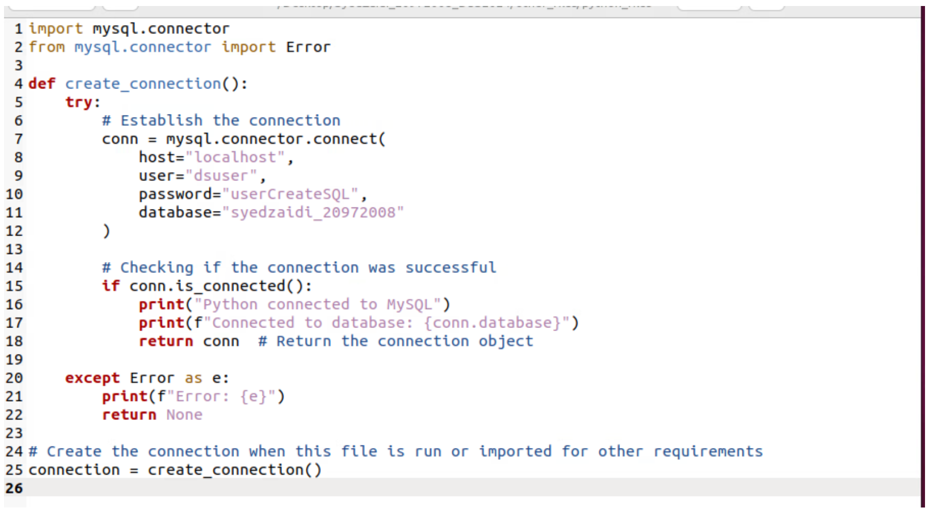


```
mysql> SELECT * FROM Top10EventParticipationSummary;
+----------------------+------------+----------+--------------------+
| event_name           | event_date | venue_id | total_participants |
+----------------------+------------+----------+--------------------+
| Football : Men       | 2024-08-08 | Ven22    |                313 |
| Football : Women     | 2024-08-09 | Ven22    |                240 |
| Hockey : Men         | 2024-08-08 | Ven35    |                212 |
| Hockey : Women       | 2024-08-09 | Ven35    |                203 |
| Handball : Women     | 2024-08-10 | Ven30    |                202 |
| Handball : Men       | 2024-08-11 | Ven30    |                184 |
| Rugby Sevens : Men   | 2024-07-27 | Ven31    |                159 |
| Rugby Sevens : Women | 2024-07-30 | Ven31    |                158 |
| Volleyball : Men     | 2024-08-09 | Ven30    |                156 |
| Water Polo : Men     | 2024-08-11 | Ven23    |                156 |
+----------------------+------------+----------+--------------------+
10 rows in set (0.01 sec)
```

Figure 33 : Results of View 1

Connection to Python

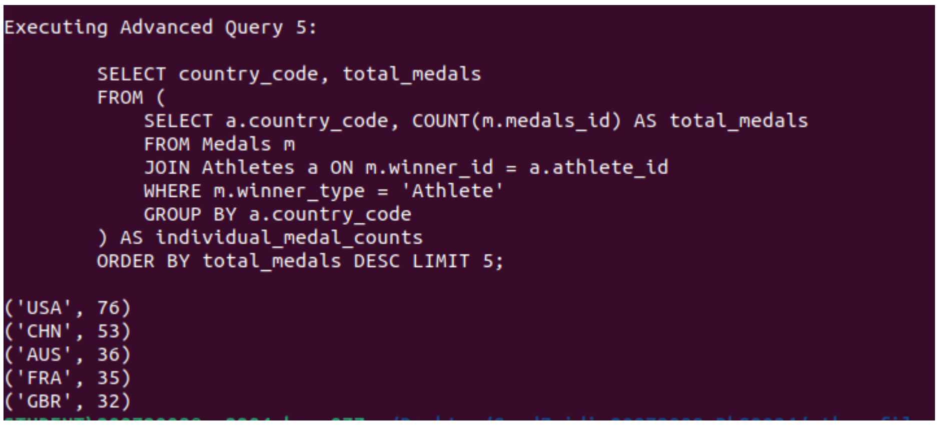
- Connecting the database to a programming language like python gives more control of how you can make changes to your data and retrieve results. In order to do so in the VMware environment, the first thing was to locate the right folder and there in terminal, install the mysql connector to python (discussed in detail in userguide).
- Here we run multiple files created in the "python_files" folder to fullfill the requirements of the project.
- First being the one to connect with sql data base. A screenshot of the python script is attached below.



```python
import mysql.connector
from mysql.connector import Error

def create_connection():
    try:
        # Establish the connection
        conn = mysql.connector.connect(
            host="localhost",
            user="dsuser",
            password="userCreateSQL",
            database="syedzaidi_20972008"
        )

        # Checking if the connection was successful
        if conn.is_connected():
            print("Python connected to MySQL")
            print(f"Connected to database: {conn.database}")
            return conn  # Return the connection object

    except Error as e:
        print(f"Error: {e}")
        return None

# Create the connection when this file is run or imported for other requirements
connection = create_connection()
```

Figure 34 : Python script to connect to MySQL

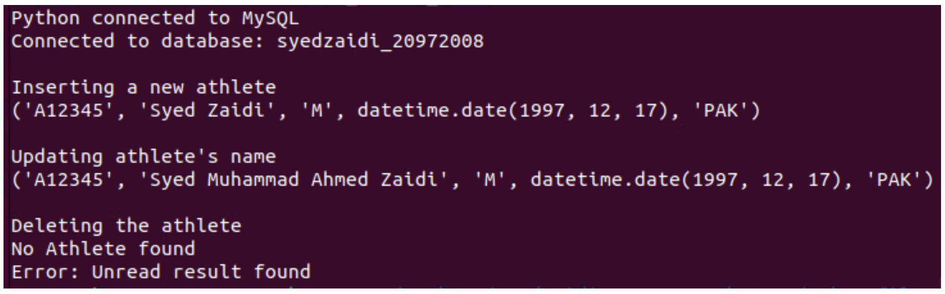
- Once this is done, the requirement was to use the defined queries, this is thoroughly explained on how to do it in order to run them from python eviorment in the UserGuide. A screenshot of the results is shown below of the last advanced query results in python.



```
Executing Advanced Query 5:

        SELECT country_code, total_medals
        FROM (
            SELECT a.country_code, COUNT(m.medals_id) AS total_medals
            FROM Medals m
            JOIN Athletes a ON m.winner_id = a.athlete_id
            WHERE m.winner_type = 'Athlete'
            GROUP BY a.country_code
        ) AS individual_medal_counts
        ORDER BY total_medals DESC LIMIT 5;

('USA', 76)
('CHN', 53)
('AUS', 36)
('FRA', 35)
('GBR', 32)
```

Figure 35 : Results of Advance query 5 through Python

- Once all queries are executed, the last requirement was to insert, update and delete the data from the sql database using python. This is done by creating an athlete of my name and credentials, updating the name to full name and then deleting it. A screenshot of the results is attached below. The code can be seen in the "insert_update_delete.py" file.



```
Python connected to MySQL
Connected to database: syedzaidi_20972008

Inserting a new athlete
('A12345', 'Syed Zaidi', 'M', datetime.date(1997, 12, 17), 'PAK')

Updating athlete's name
('A12345', 'Syed Muhammad Ahmed Zaidi', 'M', datetime.date(1997, 12, 17), 'PAK')

Deleting the athlete
No Athlete found
Error: Unread result found
```

Figure 36 : Results of Insert, update, Delete using Python

Discussion and Reflection

In summary, the project was a comprehensive dive into how a database is created, queried and connected to other programming languages. It was required to use a real-life event, Olympics 2024 to find relevant original data and create a relational database from scratch. It was essential to think and brainstorm the entities that were needed and consider all the attributes available inorder to select only the ones that would fullfill the requirements within the time allocated. Once the database was all set, the second phase needed us to answer questions by using the query language learnt over the phase of this unit. From using basic concepts to advanced, information was retrieved to give precise answers to already defined questions. Lastly, advanced features like stored procedures, views and triggers were used to explore further use of SQL and how more integrity can be added to the databse design. Finally, it was required to establish a connection with python to do all these tasks through a programming language.

Challenges

I personally came across multiple challenges over the completion of this project. These are shared as below

- The biggest and the most time consuming challenge was the selection of the right entities and attributes. The data is widely available on multiple platforms plus its huge. Majority of it is not cleaned and selecting the right information was a key challenge faced. It was overcome by starting small from 2-3 entities and slowly increasing the size.
- Another challenge was the creation of foreign keys. This took a a lot of time to execute since the data had so many duplicates and some were not in the right data format which is why importing in to MySql always caused errors. The challenge was

overcome by using Excel with formulas like Vlookup and conditional formatting to remove discrepancies and keep only the right rows.

- The challenge to do the work in VMware, specially with the Linux server was difficult. This is because, at times it was very slow and there was a period in which it remained crashed as well. Hence, patience was required to see everything go through with slow place.
- The last challenge was time management. The project is a comprehensive one and needs to be done one step at a time. This requires time and dedication to remain consistent. The challenge was overcome by dividing it into smaller milestones and focusing on each at a time.

Limitations

- The project does not provide complete information about the Olympics 2024. Since some rows were duplicated and were not cleaned so had to be deleted.
- Although some automation was achieved with procedures and triggers, it could be made more dynamic and fully automated for further ease.
- The project was required to do on Linux environment, which may not be considered as the highly used platform in the industry.
- Python scripts and other SQL statements were created and executed in a terminal. These can be considered slow compared to the IDEs now available.

Improvements

- I would personally want to improve the project by creating it more complex by adding more entities and attributes.
- There could be the use of more widely used platforms to practice this, for example, using Windows as an operating system and using IDEs for Python scripts.
- I would like to improve further on the advanced concepts to see how other stored procedures and triggers can be added to the design to add more integrity.
- The data can be improved further by cross checking it with other sources to see if the retrieved information through the defined queries is correct.

References

https://www.kaggle.com/datasets/piterfm/paris-2024-olympic-summer-games

https://support.microsoft.com/en-au/office/vlookup-function-0bbc8083-26fe-4963-8ab8-93a18ad188a1

https://www.ablebits.com/office-addins-blog/excel-conditional-formatting/

https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html

https://stackoverflow.com/questions/372885/how-do-i-connect-to-a-mysql-database-in-python

https://www.geeksforgeeks.org/what-is-stored-procedures-in-sql/