FINAL ASSESSMENT

~

DATABASE SYSTEMS

Sophia Matassa

20864846

Monday: 12pm-2pm

INTRODUCTION

Created data frames using the pandas library in python which I used for data cleaning and manipulation before putting the data into the MYSQL databases and their corresponding tables.

Database included data about the Olympics. Queries and advanced features were then used to display useful information that a person would be interested in knowing.

DESIGN

Entity Set	**Key**	**Other Attributes**
Country	countryCode	country
Athlete	athleteCode	name, gender
Discipline	sportCode	discipline
Venue	venueCode	venue
Event	event	eventType
Team	teamCode	noAthletes

Relationship Set	**Between Which Entities**	**Cardinality**
represent	Country, Athlete, Team	One-Many (Athletes or teams can only represent one country at a time; A country can have many athletes or teams)
part of	Athlete, Team	Many-Many (Athletes can be part of many teams as can teams have many athletes)
participate	Athlete, Team, Event, Discipline	Many-Many (Athletes or teams can participate in several events; Events have several athletes or teams participating in them) – discipline is required for the event to be uniquely identified
results	Athlete, Team, Event, Discipline	Many-Many (Athletes or teams can win several events; An event can have several athletes or teams winning a medal) – discipline is required for the event to be uniquely identified
element of	Discipline, Event	One-Many (An event can be an element of one discipline; Many different events can be an element of a discipline)
location	Event, Venue	Many-Many

		(An event can have different phases at different venues; A venue can have different events occurring at it)

Relationship Set	**Between Which Entities**	**Participation**
represent	Country, Athlete, Team	Country – partial, Athlete – total, Team-total (An athlete/team in the Olympics must represent a country to be an athlete in the Olympics; A country can exist without any athletes/teams.
part of	Athlete, Team	Athlete – partial, Team – total (Athletes are able to compete without a team; A team cannot exist without athletes to be part of it)
participate	Athlete, Team, Event, Discipline	Athlete – total, Team – total, Event – partial, Discipline – partial (Athletes/Teams need to compete in at least one event to participate in the Olympics; Events and disciplines can exist without athletes or teams.
results	Athlete, Team, Event, Discipline	Athlete – partial, Team – partial, Event – partial, Discipline – partial (Athletes/Teams do not need to receive a medal to be in the Olympics; Events and disciplines can exist without medals being receive by athletes or teams.
element of	Discipline, Event	Discipline – partial, Event – total (A discipline exists even without events; An event can't exist without being part of a discipline)
location	Event, Venue	Event – total, Venue – partial (For an event to occur it must occur in a venue of some sort; A venue can exist without an event occurring in it)

RELATIONAL SCHEMA

Country(<u>countryCode</u>, country)
Athlete(<u>athleteCode</u>, name, gender, countryCode)
    FK countryCode REF Country(countryCode)
Team(<u>teamCode</u>, noAthletes, countryCode)
    FK countryCode REF Country(countryCode)
Venue(<u>venueCode</u>, venue)
Discipline(sportsCode, <u>discipline</u>)
Event(<u>event</u>, eventType, venueCode, <u>discipline</u>)
    FK venueCode REF Venue(venueCode),

FK discipline REF Discipline(discipline)

PartOf(athleteCode, teamCode)

FK athleteCode REF Team(teamCode)

Results(medal, athleteCode, teamCode, event, discipline)

FK athleteCode REF Athlete(athleteCode),
FK teamCode REF Team(teamCode),
FK event REF Event(event),
FK discipline REF Discipline(discipline)

Participate(athleteCode, teamCode, event, discipline),

FK athleteCode REF Athlete(athleteCode),
FK teamCode REF Team(teamCode),
FK event REF Event(event),
FK discipline REF Discipline(discipline)

Location(phase, startTime, startDate, endTime, endDate, event, discipline, venueCode)

FK event REF Event(event),
FK discipline REF Discipline(discipline),
FK venueCode REF Venue(venueCode)

Country

Field	Type	Null	Key	Default	Extra	Description
countryCode	VARCHAR(3)	NO	PRI			code that identifies the country
country	VARCHAR(50)	NO				name of the country

Athlete

Field	Type	Null	Key	Default	Extra	Description
athleteCode	VARCHAR(7)	NO	PRI			code that identifies the athlete
name	VARCHAR(50)	NO				name of the athlete
gender	CHAR(8)	YES		NULL		gender of the athlete
countryCode	VARCHAR(3)	NO			FK	

Team

Field	Type	Null	Key	Default	Extra	Description
teamCode	VARCHAR(15)	NO	PRI			code that identifies the team
noAthletes	SMALL INT	YES		0		number of athletes in the team
countryCode	VARCHAR(3)	NO			FK	

Venue

Field	Type	Null	Key	Default	Extra	Description
venueCode	VARCHAR(3)	NO	PRI			code that identifies the venue
venue	VARCHAR(50)	YES		NULL		name of the venue

Discipline

Field	Type	Null	Key	Default	Extra	Description
sportCode	VARCHAR(3)	NO				code that identifies the discipline
discipline	VARCHAR(20)	YES	PRI	NULL		name of the discipline

Event

Field	Type	Null	Key	Default	Extra	Description
event	VARCHAR(50)	NO	CAND			name of the event
eventType	CHAR(8)	YES		NULL		the type of event occurring
discipline	VARCHAR(50)	NO	CAND		FK	

Results

Field	Type	Null	Key	Default	Extra	Description
medal	SMALLINT	NO			0>medal>=3	medal number (1
athleteCode	VARCHAR(7)	YES		NULL	FK	is first, etc)
teamCode	CHAR(16)	YES		NULL	FK	
event	VARCHAR(50)	NO			FK	
discipline	VARCHAR(50)	NO			FK	
UNIQUE(athlete_code, event, discipline) UNIQUE(team_code, event, discipline)						

Location

Field	Type	Null	Key	Default	Extra	Description
phase	VARCHAR(50)	NO	PRI			phase of a specific event
startTime	TIME	NO	PRI			time at which the phase is starting
startDate	DATE	NO	PRI			date at which the phase is starting
endTime	TIME	YES		NULL		time at which the phase is ending
endDate	DATE	YES		NULL		date at which the phase is ending
venueCode	VARCHAR(3)	NO			FK	
event	VARCHAR(50)	NO	PRI		FK	
discipline	VARCHAR(50)	NO	PRI		FK	

Participate

Field	Type	Null	Key	Default	Extra
athleteCode	VARCHAR(7)	YES		NULL	FK
teamCode	VARCHAR(15)	YES		NULL	FK
event	VARCHAR(50)	NO			FK
discipline	VARCHAR(50)	NO			FK
UNIQUE(athlete_code, event, discipline) UNIQUE(team_code, event, discipline)					

Part of

Field	Type	Null	Key	Default	Extra
athleteCode	VARCHAR(7)	NO			FK
teamCode	VARCHAR(15)	NO	PRI		FK

Explanation of Selected Entities

From the data given to us, the entities that were chosen to provide the most useful information in a relatively large database that maintains 3NF form. Country, Athlete, Team, Event, Discipline and Venue were chosen to be the entities, with Event being a weak entity. Athlete was chosen due to its integral nature of tying all other information together. Country, Event and discipline were chosen with the same reasoning. Event is a weak entity due to it relying on the key discipline from the Discipline table to be uniquely identified.

Team and venue were chosen to add information that isn't integral but adds an extra layer of information that would be generally expected. Team entity allows for easy access to information about team sports which are very important in the Olympics. Venue entity allows for access to information about where a sport occurs which is

useful in the event that a spectator would like to watch the event (In the scenario that this database is live during the Olympics).

Explanation of Selected Relationships

The relationships that were chosen are as follows: Represent, PartOf, Participate, Results, ElementOf and Location. Tables were required to be made for PartOf, Participate, Results and Location due to their nature to ensure that all databased maintained 3NF form.

The Team table does not store the athleteCodes. This choice was made to reduce redundancy in the information in other tables and leave the identification of which teams the athletes are a part of to the relationship table PartOf. This creates a dependency of athletes that are part of teams to go through the PartOf table before accessing other information such as the events they're part of or the medal they received.

Participate stores the athletes and teams that participated in specific events. No primary keys are in this relationship; unique keys are instead used to supplement primary keys. This is due to the athlete and team keys being in separate columns and so either can be null so did not allow for implementation of primary keys. As mentioned before, if an athlete competes in a team they're id will not be present. However, if they compete in an individual event and team event, their code will only be present for the individual event.

Results stores the medal that has been awarded. Again, no primary key was used in this table, only unique keys. This was due to the presence of ties with medals as the original plan was to have the medal, discipline and event as the primary key. I felt like I was losing important information by removing ties in the data so I went ahead with using unique keys instead. This involves the teamCode or athleteCode, depending on which is not null.

Location stores additional information not present in event or venue. This is the phase of the event that is occurring and the time and date at which the phase is occurring. This was not stored in the Event table due to the constraints of the medal data being present.

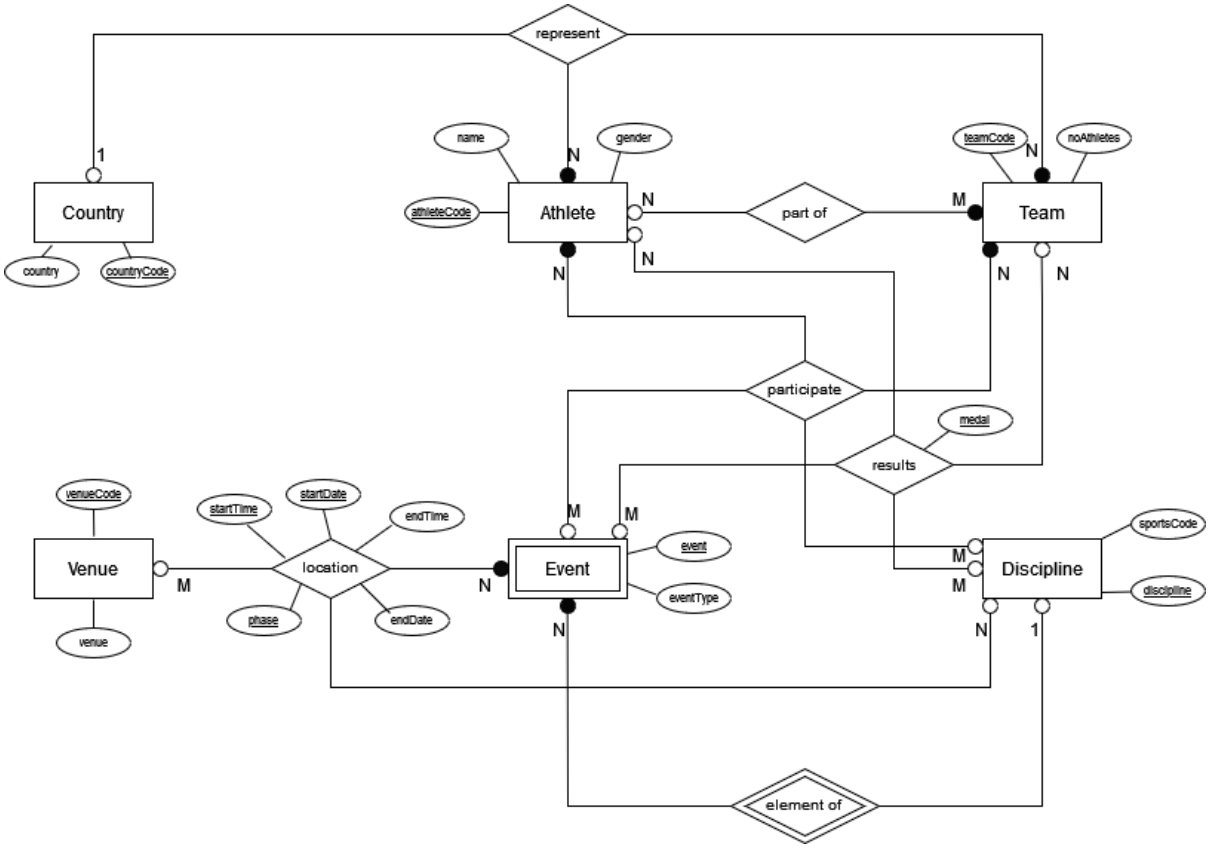
Explanation of Data types

VARCHAR(50) was used for discipline, event and phase to ensure the descriptions would fit and wasn't ridiculously big so as to lose valuable memory and make the program slower.

SMALLINT was also used with the intent on saving memory as compared to INT which would use a lot more.

VARCHAR(3) was used for all the codes that were 3 of length. VARCHAR(7) for athleteCodes and VARCHAR(15) for teamCodes. This ensured data integrity of these important values from the unique sizes. It also minimizes the amount of storage used.

DATE and TIME data types were used for the dates and times to allow for easy use of this information in queries.



Assumptions Made During Design

- In the database provided an athlete is not part of the same team twice.
- athletes and teams that are not current are not important information
- alternate athlete information was also considered not important

IMPLEMENTATION OF DATABASE

Pandas data frame was used to allow for cleaning of data. This involved removing teams and athletes that were not current, and athletes that were alternate athletes is this was deemed to not be important information in the database.

Data was taken from this url:

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

The following sections of code demonstrate how the csv was read into the database. A brief overview is provided as there is a lot of code involved in this part.

The first section shows the method used to read the specific table from the data file.

```
def create_table(table_name, columns):
    """ reads csv file into a dataframe and selects specific
columns """
    all_data = pd.read_csv(f"./data/{table_name}")
    return all_data[columns]
```

The next section demonstrates how the data was cleaned. Raw_athletes contains raw data of the csv athlete file. The method that is used from dataframe_methods is included below, however there are more used in the cleaning of other dataframes.

```
raw_athletes = dm.create_table("athletes.csv",
df_cols['athletes'])
cou1 = raw_athletes[['country_code','country']]
Country = dm.remove_duplicates(cou1)
```

```
def remove_duplicates(df):
    new_df = df.drop_duplicates()
    return new_df
```

Country would then be stored in frames[] with the rest of the dataframes. The last section shows how the insert queries were carried out with the frames list.

```
for i in range(len(queries)):
            sm.database_insert(sql_cur,frames[i], queries[i])
```

PART 3 – DESIGN AND IMPLEMENTATION OF QUERIES

Query 1:

Show all the events that occurred in the morning of the first week of the Olympics (24th of July to 28th of July).

```
SELECT phase, discipline
FROM Location
WHERE endDate < '2024-07-28' AND endTime < '12:00:00';
```

- If I was only free during the morning to watch the Olympics of the first week, I would like to know what events I can view.

endTime	endDate	phase	discipline
10:30:00	2024-07-25	Women's Preliminary Round Group A	Handball
09:30:00	2024-07-26	10m Air Rifle Mixed Team Training	Shooting
11:15:00	2024-07-26	10m Air Pistol Men's Training	Shooting
09:10:00	2024-07-27	Mixed Doubles Group Play Stage - Grp B	Badminton
09:10:00	2024-07-27	Mixed Doubles Group Play Stage - Grp D	Badminton
09:10:00	2024-07-27	Women's Singles Group Play Stage - Grp N	Badminton
10:30:00	2024-07-27	Men's Preliminary Round Group A	Handball
10:07:00	2024-07-27	Men's Single Sculls Heats	Rowing
09:30:00	2024-07-27	10m Air Rifle Mixed Team Qualification	Shooting
10:30:00	2024-07-27	Men's Preliminary Round - Pool C	Volleyball
10:00:00	2024-07-27	Men's Doubles Group Play Stage - Grp A	Badminton
10:00:00	2024-07-27	Mixed Doubles Group Play Stage - Grp D	Badminton
10:00:00	2024-07-27	Women's Singles Group Play Stage - Grp I	Badminton
10:25:00	2024-07-27	Women's Épée Individual Table of 64	Fencing
11:30:00	2024-07-27	Men's Pool A	Hockey
10:06:00	2024-07-27	Men -60 kg Elimination Round of 32	Judo
10:06:00	2024-07-27	Women -48 kg Elimination Round of 32	Judo
10:12:00	2024-07-27	Men -60 kg Elimination Round of 32	Judo
10:12:00	2024-07-27	Women -48 kg Elimination Round of 32	Judo
10:50:00	2024-07-27	Men's Doubles Group Play Stage - Grp A	Badminton
10:50:00	2024-07-27	Mixed Doubles Group Play Stage - Grp B	Badminton
10:50:00	2024-07-27	Women's Doubles Group Play Stage - Grp A	Badminton
10:18:00	2024-07-27	Men -60 kg Elimination Round of 32	Judo
10:18:00	2024-07-27	Women -48 kg Elimination Round of 32	Judo
11:19:00	2024-07-27	Women's Single Sculls Heats	Rowing
10:24:00	2024-07-27	Men -60 kg Elimination Round of 32	Judo
10:24:00	2024-07-27	Women -48 kg Elimination Round of 32	Judo
10:30:00	2024-07-27	Men -60 kg Elimination Round of 32	Judo
10:30:00	2024-07-27	Women -48 kg Elimination Round of 32	Judo
10:50:00	2024-07-27	Men's Sabre Individual Table of 64	Fencing

Query 2:

Select all the athletes that have the last name Brown.

```
SELECT *
FROM Athlete
WHERE SUBSTRING_INDEX(name, ' ', 1) = 'Brown';
```

- Interested to know how many people in the Olympics would share this last name. Expected more.

athleteCode	name	gender	countryCode
1538830	BROWN Janeek	Female	JAM
1544123	BROWN Sky	Female	GBR
1563933	BROWN Peter	Male	IRL
1574112	BROWN Ronald	Male	RSA
1896125	BROWN Charles	Male	GBR
1897751	BROWN Nina	Female	NZL
1913343	BROWN Lynnzee	Female	HAI
1918759	BROWN Georgia-Rose	Female	NZL
1940173	BROWN Grace	Female	AUS
1940450	BROWN Jayden	Male	AUS
1945376	BROWN Max	Male	NZL
1945994	BROWN Rhiannan	Female	AUS
1957818	BROWN Patrick James	Male	GBR
1958472	BROWN Lorenzo	Male	ESP
1960588	BROWN Brittany	Female	USA
1960646	BROWN Joseph	Male	USA
1960650	BROWN Kaylyn	Female	USA
1966448	BROWN Kaiya	Female	SAM
1974072	BROWN Aaron	Male	CAN

Query 3:

Show all the ties that occurred.

```
SELECT r1.medal, r1.athleteCode, r1.discipline, r1.event
FROM Results r1
JOIN (
     SELECT medal, discipline, event
     FROM Results
     GROUP BY medal, discipline, event
     HAVING COUNT(athleteCode)>1
) r2
ON r1.medal=r2.medal AND r1.discipline=r2.discipline AND
r1.event=r2.event;
```

- Interesting to know how many duplicate medals needed to be awarded due to ties occurring.

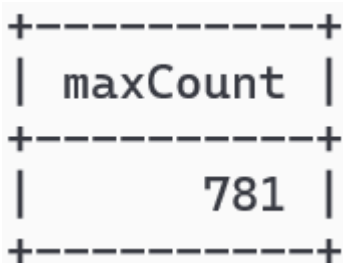
medal	athleteCode	discipline	event
2	1935901	Swimming	Men's 100m Breaststroke
2	1956469	Swimming	Men's 100m Breaststroke
3	1565754	Judo	Men -100 kg
3	1892957	Judo	Men -100 kg
3	1893274	Taekwondo	Men -58kg
3	1947736	Taekwondo	Men -58kg
3	1563544	Judo	Men -60 kg
3	1896752	Judo	Men -60 kg
3	1561138	Judo	Men -66 kg
3	1935410	Judo	Men -66 kg
3	1910367	Taekwondo	Men -68kg
3	1955815	Taekwondo	Men -68kg
3	1563437	Judo	Men -73 kg
3	1896740	Judo	Men -73 kg
3	1914832	Taekwondo	Men -80kg
3	1929685	Taekwondo	Men -80kg
3	1570561	Judo	Men -81 kg
3	1928079	Judo	Men -81 kg
3	1891318	Judo	Men -90 kg
3	1961318	Judo	Men -90 kg
3	1566931	Judo	Men +100 kg
3	1567512	Judo	Men +100 kg
3	1548401	Taekwondo	Men +80kg
3	1897088	Taekwondo	Men +80kg
3	1879120	Boxing	Men's +92kg
3	1895903	Boxing	Men's +92kg
3	1892464	Boxing	Men's 51kg
3	1908817	Boxing	Men's 51kg
3	1555185	Boxing	Men's 57kg
3	1940136	Boxing	Men's 57kg

Query 4:

What is the most times a venue was used?

```
SELECT COUNT(venueCode) AS maxCount
FROM Location
GROUP BY venueCode
ORDER BY maxCount DESC
LIMIT 1;
```

- Interesting statistics number that was important to know.



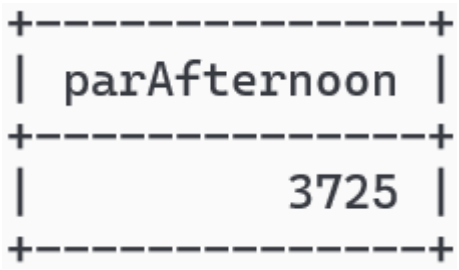
maxCount
781

Query 5:

How many athletes participated in the Olympics in the afternoons?

```
SELECT COUNT(DISTINCT ath.athleteCode) AS parAfternoon
FROM Athlete ath
JOIN Participate par
     ON ath.athleteCode = par.athleteCode
JOIN Event eve
     ON par.event=eve.event AND par.discipline=eve.discipline
JOIN Location loc
     ON eve.event=loc.event AND eve.discipline=loc.discipline
WHERE loc.startTime > '12:00:00' AND loc.endTime <
'18:00:00';
```

- Important to have an estimate of how many athletes participated in the afternoons. If I'm at the Olympics, nice number to know if I'll hang around in the afternoon.



parAfternoon
3725

Query 6:

Show all medals won by all countries and display in descending order.

```
SELECT COALESCE(ath.countryCode, tea.countryCode) AS
countryCode,
     COUNT(CASE WHEN res.medal='1' THEN 1 END) AS gold,
     COUNT(CASE WHEN res.medal='2' THEN 1 END) AS silver,
     COUNT(CASE WHEN res.medal='3' THEN 1 END) AS bronze,
FROM Results res
LEFT JOIN Athlete ath ON ath.athleteCode=res.athleteCode
LEFT JOIN Team tea ON tea.teamCode=res.teamCode
GROUP BY countryCode
ORDER BY gold DESC, silver DESC, bronze DESC;
```

- Important to see the results of the Olympics.

countryCode	gold	silver	bronze
USA	40	44	42
CHN	40	27	24
JPN	20	12	13
AUS	18	19	16
FRA	16	26	22
NED	15	7	12
GBR	14	22	29
KOR	13	9	10
ITA	12	13	15
GER	12	13	8
NZL	10	7	3
CAN	9	7	11
UZB	8	2	3
HUN	6	7	6
ESP	5	4	9
SWE	4	4	3
KEN	4	2	5
NOR	4	1	3
IRL	4	0	3
BRA	3	7	10
IRI	3	6	3
UKR	3	5	4
ROU	3	4	2
GEO	3	3	1
BEL	3	1	6
BUL	3	1	3
SRB	3	1	1
CZE	3	0	2
DEN	2	2	5
AZE	2	2	3

PART 4 - DESIGN AND IMPLEMENTATION OF ADVANCED FEATURES

File for calling advanced queries:

```
-- counts all the distinct phases of events in the olympics
there are.
SOURCE procedure1.sql;
CALL count_phases(@pcount);
SELECT @pcount;

-- user provides a discipline, event and an athleteCode, and
the program tells
     -- the user if the athlete is in the event and
discipline provided.
SOURCE procedure2.sql;
CALL discipline_medals('1946887', 'Badminton', "Women's
Singles", @result);
SELECT @result;

-- creates a view of the location and start time of all team
sports
SOURCE view1.sql;
SELECT * FROM team_loc;

-- creates a view of the start time of the first event in
each discipline
SOURCE view2.sql;
SELECT * FROM start_discipline;
```

Procedure 1:

Counts all the distinct phases of events in the Olympics there are.

```
DELIMITER //
DROP PROCEDURE IF EXISTS count_phases;
CREATE PROCEDURE count_phases(
    OUT pcount INT
)
    BEGIN
        DECLARE done INT DEFAULT 0;
        DECLARE eve CHAR(50);
        DECLARE disc CHAR(50);
        DECLARE cur CURSOR FOR SELECT event, discipline FROM
Event;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
        SET pcount = 0;
        OPEN cur;
        read_loop: LOOP
            FETCH cur INTO eve, disc;
            IF done THEN
                LEAVE read_loop;
            END IF;
            SELECT COUNT(DISTINCT phase)
            INTO @curCount
            FROM Location
            WHERE event = eve AND discipline = disc;
         SET pcount = pcount + @curCount;
every loop
        END LOOP;
        CLOSE cur;
    END //
DELIMITER ;
```

```
mysql> SOURCE advanced.sql;
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.03 sec)

+---------+
| @pcount |
+---------+
|    1243 |
+---------+
1 row in set (0.00 sec)
```

Procedure 2:

User provides a discipline, event and an athleteCode, and the program tells the user if the athlete is in the event and discipline provided.

```
DELIMITER //

DROP PROCEDURE IF EXISTS discipline_medals;
CREATE PROCEDURE discipline_medals(
    IN acode VARCHAR(17),
    IN disc VARCHAR(50),
    IN eve VARCHAR(50),
    OUT participation CHAR(8)
)
    BEGIN
        DECLARE curCount INT DEFAULT 0;
        DECLARE pcount INT DEFAULT 0;

        SELECT COUNT(*)
        INTO curCount
        FROM Athlete ath
        JOIN Participate par
        ON ath.athleteCode=par.athleteCode
        WHERE par.discipline=disc AND par.event=eve AND
par.athleteCode=acode;
        SET pcount = pcount + curCount;

        SELECT COUNT(*)
        INTO curCount
        FROM Athlete ath
        JOIN PartOf pao
            ON ath.athleteCode=pao.athleteCode
        JOIN Participate par
            ON pao.teamCode=par.teamCode
        WHERE par.discipline=disc AND par.event=eve AND
par.athleteCode=acode;

        SET pcount = pcount + curCount;

        IF pcount > 0 THEN
            SET participation = 'yes';
        ELSE
            SET participation = 'no';
     END IF;
    END //
DELIMITER ;
```

```
Query OK, 0 rows affected (0.02 sec)

Query OK, 0 rows affected (0.05 sec)

Query OK, 1 row affected (0.00 sec)

+---------+
| @result |
+---------+
| yes     |
+---------+
1 row in set (0.00 sec)
```

View 1:

Creates a view of the location and start time of all team sports.

```
DROP VIEW IF EXISTS team_loc;
CREATE VIEW team_loc AS
    SELECT DISTINCT
loc.discipline,loc.phase,loc.startTime,loc.startDate
    FROM Team tea
    JOIN Participate par
        ON tea.teamCode = par.teamCode
    JOIN Event eve
        ON par.event=eve.event AND
par.discipline=eve.discipline
    JOIN Location loc
        ON loc.event=eve.event AND
loc.discipline=eve.discipline

    WHERE par.event=loc.event AND
par.discipline=loc.discipline;
```

discipline	phase	startTime	startDate	venueCode
3x3 Basketball	Men's Pool Round	18:35:00	2024-07-30	LC1
3x3 Basketball	Men's Pool Round	19:05:00	2024-07-30	LC1
3x3 Basketball	Men's Pool Round	22:05:00	2024-07-30	LC1
3x3 Basketball	Men's Pool Round	22:35:00	2024-07-30	LC1
3x3 Basketball	Men's Pool Round	18:35:00	2024-07-31	LC1
3x3 Basketball	Men's Pool Round	19:05:00	2024-07-31	LC1
3x3 Basketball	Men's Pool Round	22:05:00	2024-07-31	LC1
3x3 Basketball	Men's Pool Round	22:35:00	2024-07-31	LC1
3x3 Basketball	Men's Pool Round	10:05:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	10:35:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	13:35:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	14:05:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	19:05:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	19:35:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	22:35:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	23:35:00	2024-08-01	LC1
3x3 Basketball	Men's Pool Round	10:05:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	10:35:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	13:35:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	14:05:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	18:35:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	19:05:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	22:05:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	22:35:00	2024-08-02	LC1
3x3 Basketball	Men's Pool Round	17:30:00	2024-08-04	LC1
3x3 Basketball	Men's Pool Round	18:00:00	2024-08-04	LC1
3x3 Basketball	Men's Pool Round	18:35:00	2024-08-04	LC1
3x3 Basketball	Men's Pool Round	19:05:00	2024-08-04	LC1
3x3 Basketball	Men's Play-in Games	21:30:00	2024-08-04	LC1
3x3 Basketball	Men's Play-in Games	22:05:00	2024-08-04	LC1
3x3 Basketball	Men's Semifinals	18:00:00	2024-08-05	LC1
3x3 Basketball	Men's Semifinals	19:00:00	2024-08-05	LC1
3x3 Basketball	Men's Finals	21:30:00	2024-08-05	LC1
3x3 Basketball	Men's Finals	22:30:00	2024-08-05	LC1
3x3 Basketball	Women's Pool Round	17:30:00	2024-07-30	LC1
3x3 Basketball	Women's Pool Round	18:00:00	2024-07-30	LC1
3x3 Basketball	Women's Pool Round	21:00:00	2024-07-30	LC1
3x3 Basketball	Women's Pool Round	21:30:00	2024-07-30	LC1
3x3 Basketball	Women's Pool Round	17:30:00	2024-07-31	LC1
3x3 Basketball	Women's Pool Round	18:00:00	2024-07-31	LC1
3x3 Basketball	Women's Pool Round	21:00:00	2024-07-31	LC1
3x3 Basketball	Women's Pool Round	21:30:00	2024-07-31	LC1
3x3 Basketball	Women's Pool Round	09:00:00	2024-08-01	LC1
3x3 Basketball	Women's Pool Round	09:30:00	2024-08-01	LC1

View 2:

Creates a view of the start time of first event in each discipline

```
DROP VIEW IF EXISTS start_discipline;

CREATE VIEW start_discipline AS
    SELECT loc.discipline, loc.phase, loc.startDate,
loc.startTime
    FROM Location loc
    INNER JOIN (
        SELECT discipline, MIN(CONCAT(startDate, ' ',
startTime)) as eventTime
          FROM Location
     GROUP BY discipline
     ) earliestEvent
    ON loc.discipline = earliestEvent.discipline
    AND CONCAT(loc.startDate, ' ', loc.startTime) =
earliestEvent.eventTime
    ORDER BY loc.startDate ASC, loc.startTime ASC;
```

discipline	phase	startDate	startTime
Football	Men's Group B	2024-07-24	15:00:00
Football	Men's Group C	2024-07-24	15:00:00
Rugby Sevens	Men's Pool B	2024-07-24	15:30:00
Handball	Women's Preliminary Round Group A	2024-07-25	09:00:00
Archery	Women's Individual Ranking Round	2024-07-25	09:30:00
Shooting	10m Air Rifle Mixed Team Training	2024-07-26	09:00:00
Badminton	Mixed Doubles Group Play Stage - Grp B	2024-07-27	08:30:00
Badminton	Mixed Doubles Group Play Stage - Grp D	2024-07-27	08:30:00
Badminton	Women's Singles Group Play Stage - Grp N	2024-07-27	08:30:00
Rowing	Men's Single Sculls Heats	2024-07-27	09:00:00
Volleyball	Men's Preliminary Round - Pool C	2024-07-27	09:00:00
Equestrian	Eventing Individual Dressage	2024-07-27	09:30:00
Equestrian	Eventing Team Dressage	2024-07-27	09:30:00
Fencing	Women's Épée Individual Table of 64	2024-07-27	10:00:00
Hockey	Men's Pool A	2024-07-27	10:00:00
Judo	Men -60 kg Elimination Round of 32	2024-07-27	10:00:00
Judo	Women -48 kg Elimination Round of 32	2024-07-27	10:00:00
Artistic Gymnastics	Men's Qualification	2024-07-27	11:00:00
Basketball	Men's Group Phase - Group A	2024-07-27	11:00:00
Diving	Women's Synchronised 3m Springboard	2024-07-27	11:00:00
Swimming	Women's 100m Butterfly - Heats	2024-07-27	11:00:00
Tennis	Men's Singles First Round	2024-07-27	12:00:00
Tennis	Women's Doubles First Round	2024-07-27	12:00:00
Tennis	Women's Singles First Round	2024-07-27	12:00:00
Beach Volleyball	Men's Preliminary - Pool D - Matches	2024-07-27	14:00:00
Water Polo	Women's Preliminary Round - Group A	2024-07-27	14:00:00
Cycling Road	Women's Individual Time Trial	2024-07-27	14:30:00
Canoe Slalom	Men's Canoe Single Heats 1st and 2nd Run	2024-07-27	15:00:00
Table Tennis	Men's Singles Preliminary Round	2024-07-27	15:00:00
Table Tennis	Women's Singles Preliminary Round	2024-07-27	15:00:00
Boxing	Women's 54kg - Prelims - Round of 32	2024-07-27	15:30:00
Surfing	Men's Round 1	2024-07-27	19:00:00
Skateboarding	Women's Street Prelims	2024-07-28	12:00:00
Sailing	Women's Skiff Opening Series	2024-07-28	12:35:00
Cycling Mountain Bike	Women's Cross-country	2024-07-28	14:10:00
Cycling BMX Freestyle	Women's Park Qualification	2024-07-30	13:25:01
3x3 Basketball	Women's Pool Round	2024-07-30	17:30:00
Triathlon	Women's Individual	2024-07-31	08:00:00
Athletics	Men's 20km Race Walk Final	2024-08-01	08:00:00
Golf	Men's Individual Stroke Play	2024-08-01	09:00:00
Cycling BMX Racing	Men, Quarterfinals	2024-08-01	20:00:00
Trampoline Gymnastics	Women's Qualification	2024-08-02	12:00:00
Sport Climbing	Men's Boulder & Lead Semifinals	2024-08-05	10:00:00
Wrestling	MGR 130kg 1/8 Finals	2024-08-05	15:00:00
Wrestling	MGR 60kg Qualifications	2024-08-05	15:00:00
Wrestling	WFS 68kg 1/8 Finals	2024-08-05	15:00:00
Cycling Track	Women's Team Sprint - Qualifying	2024-08-05	17:00:00

Database Connectivity and Python Implementations

As shown below, python is connected to MYSQL through caching password connection and this connection is used to insert all the data frames into the MYSQL databases.

main.py:

```
username = input("Enter your username: ")
password = input("Enter your password: ")
sql_database = sm.connect_sql(username, password)
```

sql_methods.py

```
def connect_sql(username, password):
    db = mysql.connector.connect(host='localhost',
user=username, password=password)
    return db
```

A couple methods utilised this connection. The first one was for specifically for inserting data into the databases.

```
def database_insert(sql_cur, dataframe, query):
    print(f"Executing {query}")
    for i, row in dataframe.iterrows():
        sql_cur.execute(query, tuple(row))
```

Another method I would like to point out reads all queries that were necessary as part of this assessment and is able to handle reading files through recursion.

This method also prints tables in the same format as MYSQL tables allowing for a similar experience to just using MYSQL.

```
def sql_command(cur, command):
    command = command.strip()
    case1 = ["CREATE", "UPDATE", "INSERT", "DROP", "USE",
"ALTER","DELETE"]
    case2 = ["SELECT", "DESCRIBE"]
    case3 = ["SOURCE", "./"]
    if command:
        if command.upper().split(" ")[0] in case1:
            print(f"Executing: {command}")
            cur.execute(command)
        elif command.upper().split(" ")[0] in case2:
            print(f"Executing: {command}")
            cur.execute(command)
            result = cur.fetchall()
            col = [desc[0] for desc in cur.description]
            print(tabulate(result, headers=col,
tablefmt="pretty"))
        elif command.upper().split(" ")[0] in case3:
            source_file = command.split()[1]
            print(f"Executing SOURCE file: {source_file}")
            read_sql_file(cur, source_file)
        else:
            print(f"unknown command: {command.split()[0]}")

def read_sql_file(cur, file):
    """ contains logic to process any sql statements """
    with open(f"./sql_files/{file}", 'r') as file:
        file = file.read()
        print(file)
    commands = file.split(";")
    for command in commands:
        sql_command(cur, command)
```

These are several examples of this method being put to use to demonstrate how powerful it is. I use this method to create all the tables, add foreign keys in where necessary, delete values where necessary. I also created a menu where they are able to enter queries.

This is the demo file used to experiment with the connector in real-time.

```
INSERT INTO Athlete VALUES ('5000000', 'SMITH John', 'Male',
'AUS');
SELECT * FROM Athlete WHERE athleteCode='5000000';

UPDATE Athlete SET name='SMALL John' WHERE
athleteCode='5000000';
SELECT * FROM Athlete WHERE athleteCode='5000000';

DELETE FROM Athlete WHERE athleteCode='5000000';
SELECT * FROM Athlete WHERE athleteCode='5000000';
```

And this is it being used in the menu selection.

```
Would you like to:
(1) read a sql file
(2) create a query?
(0) exit
1
Enter the filename: demo.sql
INSERT INTO Athlete VALUES ('5000000', 'SMITH John', 'Male', 'AUS');
SELECT * FROM Athlete WHERE athleteCode='5000000';

UPDATE Athlete SET name='SMALL John' WHERE athleteCode='5000000';
SELECT * FROM Athlete WHERE athleteCode='5000000';

DELETE FROM Athlete WHERE athleteCode='5000000';
SELECT * FROM Athlete WHERE athleteCode='5000000';

Executing: INSERT INTO Athlete VALUES ('5000000', 'SMITH John', 'Male', 'AUS')
Executing: SELECT * FROM Athlete WHERE athleteCode='5000000'
+-------------+------------+--------+-------------+
| athleteCode |    name    | gender | countryCode |
+-------------+------------+--------+-------------+
|   5000000   | SMITH John |  Male  |     AUS     |
+-------------+------------+--------+-------------+
Executing: UPDATE Athlete SET name='SMALL John' WHERE athleteCode='5000000'
Executing: SELECT * FROM Athlete WHERE athleteCode='5000000'
+-------------+------------+--------+-------------+
| athleteCode |    name    | gender | countryCode |
+-------------+------------+--------+-------------+
|   5000000   | SMALL John |  Male  |     AUS     |
+-------------+------------+--------+-------------+
Executing: DELETE FROM Athlete WHERE athleteCode='5000000'
Executing: SELECT * FROM Athlete WHERE athleteCode='5000000'
+-------------+------+--------+-------------+
| athleteCode | name | gender | countryCode |
+-------------+------+--------+-------------+
+-------------+------+--------+-------------+
```

DISCUSSION

Achieved

Created a fully functional database that contains useful information about the Olympics through the use of MYSQL connector and pandas.

Challenges and Limitations

The implementation of the database that I presented was too complicated and could have easily been simplified in the preparation phase. However, it did provide an opportunity to learn the ins and outs of pandas data frames which I would not have learnt otherwise. There were some strange errors that occurred when converting the data frames to databases. This raised the question of if it was worth going through the effort of creating them in the first place. If I were to repeat this assessment, I would try to find a simpler way to bring in the information. A solution I thought of was by making temporary tables in MYSQL that are dropped after creating the tables that queries will be used on. This likely would have allowed me to spend less time on this assessment as a large amount of time was spent learning how to use pandas.

Improvements

A more in-depth menu selection could have been programmed, however due to the amount of time that was spent as previously mentioned, this was not possible.

Relationship tables could have been planned better as well. When implementing the ternary relationships, I didn't realise this would affect the implementation of primary keys. While the use of unique keys supplements this, more preparation would have helped me avoid this.