DATABASE SYSTEMS FINAL ASSESSMENT REPORT

SOHAIL BAKHSHI
ID: 20605126
LAB: 314:114 Thursday 12-2

Introduction

The goal of this project was to design and create a database based on either a cricket or a film festival scenario. As a result, I decided to go with the cricket scenario. Due to my selected scenario, I decided to use Indian Premier League data for my topic, and only used the first season's data since I thought dealing with thousands of records would be too time consuming. I had to go through various stages when creating the database before it was complete. These included designing the ER diagram, relational schemas, creating the database in MySQL, creating tables, loading the csv data into the tables, writing queries, writing procedures, writing triggers, and writing views as well as creating a python program that connects to my database which allows me to select, update, insert and delete tables.

Design of the Database

i) Explanation on why you have selected the entities, relationships, data types

Within my ER diagram there are 9 entities and 9 relationships.
Reason for entities:

- Match: This entity is required as a cricket database would require storing the match data such as match date, which teams are playing, winner and most valuable player/ man of the match (MVP).
- Player: A cricket database would require storing the players information such as their name, country, birthday and more.
- Venue: An event like the Indian premier league would require to be held at a venue. This entity is needed as a cricket database would require storing the venue information like the venue name, city, and country.
- Season: This entity is required as a cricket database would require storing each season's stats at the end of the season
- Team: This entity is required to identify the teams of a cricket database
- Batsman: The reason for this entity is to identify the batters and their stats during the season.
- BatsmanAverage: This entity was normalised from the batsman entity as the average was dependent on a few factors. It is required as during the season the players average would be constantly changing.
- Bowler: The reason for this entity is to identify the bowlers and their stats during the season.
- BowlerAverage: This entity was normalised from the bowler entity as the average is dependent on a few factors. It is required as during the season the players average would be constantly changing.

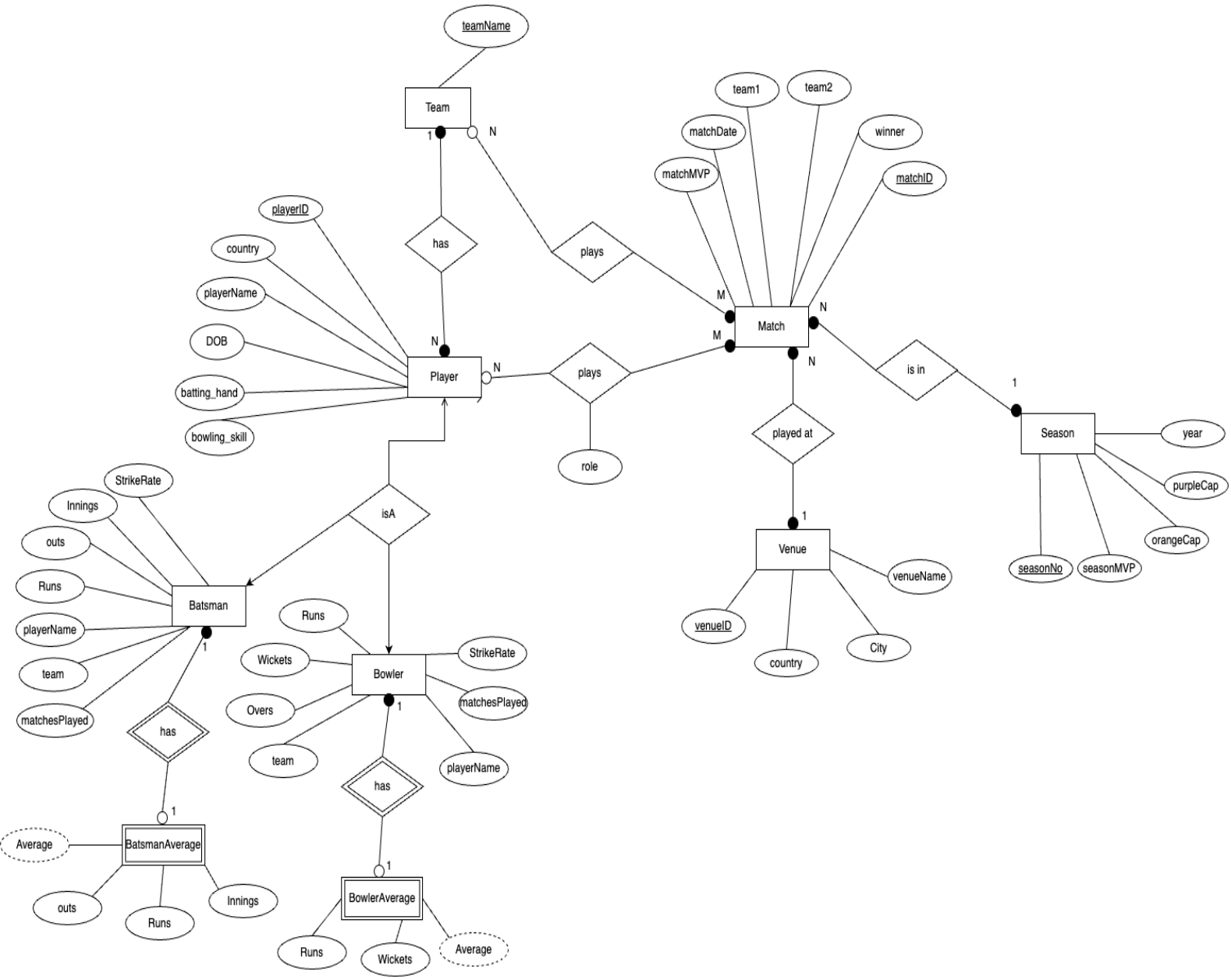
Reasons for relationships:

Relationship	Between which entities	Reason for relationship	Cardinality and reason	Participation and reason
Has	Team and player	A player has a team, and a team has a player	One-many because one team has many players, and many players have one team	Team -total: there has to be a team for there to be a player, player -total: there has to be a player in order for there to be a team
plays	Team and match	A team plays in a match	Many-many because many teams play many matches, and many matches are played by many teams	Team-partial: a team might not play in a match, Match-total: there needs to be a match for a team to play in
plays	Player and match	A player plays in a match	Many-many because many teams play many matches, and many matches are played by many teams	Player-partial: a player might not play in a match, Match-total: there needs to be a match for a player to play in
Is in	Match and season	Every sport has a season its matches are held in	Many-one because many matches are in one season and there is one season for many matches	Match- total: each match should be held within a certain season, season-total: each season must have matches
Played at	Match and venue	Each match is played at a particular venue location	Many-one because many matches are played at one venue and one venue has many matches	Match- total: a match needs to be held at a venue, Venue-total: a venue needs to have a match
isA	Player, batsman	A player is a batter	One-one because one player is a batter, and one batter is a player	Player-total, batsman -partial,
isA	Player ,bowler	A player is a bowler	One-one because one player is a bowler, and one bowler is a player	Player-total, bowler -partial
has	Bowler and bowlerAverage	A bowler has an average on based on wickets and runs	one-one because one bowler has one average, one average is given to one bowler	Bowler -total: there has to be a bowler for there to be an average, BowlerAverage –partial: it is not required for a bowler to have an average
has	Batsman and batsmanAverage	A batter has an average on based on runs /number of innings - outs	one-one because one batters has one average, one average is given to one batter	Batsman -total: there has to be a batter for there to be an average, BatsmanAverage – partial: it is not required for a batter to have an average

***explanation of datatypes in ii)

ii) ER diagram, Relational schema, data description

ER DIAGRAM



(Clearer version within the doc)

Relational Schemas

Team(teamName) * assuming there are no teams with the same name

Player(playerID, playerName, DOB, battingHand, bowlingSkill, country, teamName)
FK teamName REF Team(teamName)

Batsman(playerID, playerName,teamName, matchesPlayed, strikeRate,runs,innings,outs)
FK playerID REF Player(playerID)
FK teamName REF Team(teamName)

BatsmanAverage(playerID, runs,innings,outs,average)
FK playerID REF Player(playerID)

Bowler(playerID, playerName,teamName, matchesPlayed, strikeRate,wickets,Overs)
FK playerID REF Player(playerID)
FK teamName REF Team(teamName)

BowlerAverage(playerID, runs,wickets,average)
FK playerID REF Player(playerID)

Season(seasonNo, seasonMVP,orangeCap,purpleCap year)

Venue(venueID, venueName,city,country)

Match(matchID, matchDate, team1,team2,venueName, matchMVP, seasonNo, winner)
FK seasonNo REF Season(seasonNo)
FK venueName REF Venue(venueName)

Player_Plays(playerID ,matchID,role)
FK playerID REF Player (playerID)
FK matchID REF Match(matchID)

Team_Plays(teamName ,matchID)
FK teamName REF Player (teamName)
FK matchID REF Match(matchID)

Table descriptions

Team						
Description:	Stores the team data					
Attribute	Type	size	Reason for type	Null	Primary Key	description
teamName	varchar	36	Team names can be different lengths, so varchar is most suitable	No	Yes	The name of the team

Player						
Description:	Contains the data of the IPL players					
Attribute	Type	size	Reason for type	Null	Primary Key	description
playerID	INT	3	Player id could be anything from 1-999 so int is suitable	No	Yes	Unique player ID
playerName	varchar	36	Name could be of any length, so varchar is suitable	No	No	Player name
DOB	DATE		DATE is suitable for dates	Yes	No	Player's date of birth
battingHand	varchar	24	Batting hand description ranges in different length strings, so varchar is most suitable	Yes	No	Displays the batting hand of the player
bowlingSkill	varchar	36	Bowling skill description ranges in different length strings, so varchar is most suitable	Yes	No	Displays the bowling skill of the player
country	varchar	16	Countries names have different lengths, so varchar is most suitable	No	No	The name of the country
team	varchar	36	Team names can be different lengths, so varchar is most suitable	No	No	The players team

Season						
Description:	Contains the data of the IPL season					
Attribute	Type	size	Reason for type	Null	Primary Key	description
seasonNo	int	3	Most suitable type is an int as season number is a number	No	Yes	Season number
seasonMVP	varchar	16	Varchar is most suitable for name	Yes	No	Man of the series
orangeCap	varchar	16	Varchar is most suitable for name	Yes	No	batter who scored the most runs of the season
purpleCap	varchar	16	Varchar is most suitable for name	Yes	No	bowler with most dismissals
year	int	4	Int is suitable for year as it is just a number	Yes	No	Season year

cricketMatch						
Description:	Contains the match data					
Attribute	Type	size	Reason for type	Null	Primary Key	description
matchID	char	6	MatchID is one length so char or int is suitable	No	Yes	Match id
venueID	int	3	venueID could be any number from 1-999 so int is suitable	Yes	No	VenueID of the match
Team1	varchar	32	Team1 could be of any length, so varchar is suitable	Yes	No	Team 1
Team2	varchar	32	Team2 could be of any length, so varchar is suitable	Yes	No	Team 2
matchDate	DATE		DATE is suitable as matchDate is a date	Yes	No	Date of the match
seasonNo	INT	3	Most suitable type is an int as	Yes	No	Season number

			season number is a number			
winner	varchar	32	Winner could be of any length, so varchar is suitable	Yes	No	Winner of the match
matchMVP	int	4	The playerID is returned thus int is suitable	Yes	No	MVP of the match

Venue						
Description:	Contains the venue information where the matches were played at					
Attribute	Type	size	Reason for type	Null	Primary Key	description
venueID	int	3	Int is suitable as ID can range from 1-999	No	Yes	VenueID of the match
venueName	varchar	45	Venue name could be any length, so varchar is suitable	No	No	The name of the venue
cityName	varchar	16	City name could be of any length, so varchar is suitable	No	No	The city in which the venue is located
country	varchar	16	Country name could be of any length, so varchar is suitable	No	No	The country in which the venue is located

Player_Plays						
Description:	Table based on the player plays relationship					
Attribute	Type	size	Reason for type	Null	Primary Key	description
matchID	char	6	MatchID is one length so char or int is suitable	No	No	the id of the match
playerID	INT	3	Player id could be anything from 1-999 so int is suitable	No	No	Unique player ID
role	varchar	16	Role could be of any length, so varchar is suitable	Yes	No	Role of the player

Team_Plays						
Description:	Table based on the team plays relationship					
Attribute	Type	size	Reason for type	Null	Primary Key	description
matchID	char	6	MatchID is one length so char or int is suitable	No	No	Match id
teamName	varchar	36	Team name could be of any length, so varchar is suitable	No	No	The name of the team

Batsman						
Description:	Table for the batsman stats					
Attribute	Type	size	Reasons for type	Null	Primary Key	description
playerID	int	3	Player id could be anything from 1-999 so int is suitable	No	No	Identifies the player
playerName	varchar	36	Name could be of any length, so varchar is suitable	No	No	Player's name
team	varchar	36	Team name could be of any length, so varchar is suitable	No	No	Players team
matchesPlayed	int	2	Int is suitable as it is a number of matches played	Yes	No	The number of matches a player played
Innings	int	2	Int is suitable as it is count of how many number of innings	Yes	No	Number of Innings
NumOuts	int	2	Int is suitable as it is the number times a player got out	Yes	No	Number of times out
runs	int	3	Int is suitable as it is a count of the players runs	Yes	No	Number of runs
strikeRate	decimal	8,2	Decimal is suitable as it is a rate	Yes	No	Batters strike rate

Bowler						
Description:	Table for the bowler stats					
Attribute	Type	size	Reasons for type	Null	Primary Key	description
playerID	int	3	Player id could be anything from 1-999 so int is suitable	No	No	Identifies the player
playerName	varchar	36	Name could be of any length, so varchar is suitable	No	No	Player's name
team	varchar	36	Team name could be of any length, so varchar is suitable	No	No	Players team
matchesPlayed	int	2	Int is suitable as it is a number of matches played	Yes	No	The number of matches a player played
Innings	int	2	Int is suitable as it is count of how many number of innings	Yes	No	Number of Innings
runs	int	3	Int is suitable as it is a count of the players runs	Yes	No	Number of runs
wickets	Int	3	Int is suitable as it is a count for how many wickets taken	Yes	No	Number of wickets hit
strikeRate	decimal	8,2	Decimal is suitable as it is a rate	Yes	No	Bowlers strike rate

BatsmanAverage						
Description:	Displays the batsman average					
Attribute	Type	size	Reason for type	Null	Primary Key	description
playerID	int	3	Player id could be anything from 1-999 so int is suitable	No	No	Identifies the player
Innings	int	2	Int is suitable as it is count of how many number of innings	Yes	No	Number of Innings
NumOuts	int	2	Int is suitable as it is the number times a player got out	Yes	No	Number of times out

runs	int	3	Int is suitable as it is a count of the players runs	Yes	No	Number of runs
Average	decimal	8,2	Decimal is suitable for an average	Yes	No	Player's average

BowlersAverage						
Description:	Displays the batsman average					
Attribute	Type	size	Reason for type	Null	Primary Key	description
playerID	int	3	Player id could be anything from 1-999 so int is suitable	No	No	Identifies the player
runs	int	3	Int is suitable as it is a count of the players runs	Yes	No	Number of runs
wickets	Int	3	Int is suitable as it is a count for how many wickets taken	Yes	No	Number of wickets hit
Average	decimal	8,2	Decimal is suitable for an average	Yes	No	Player's average

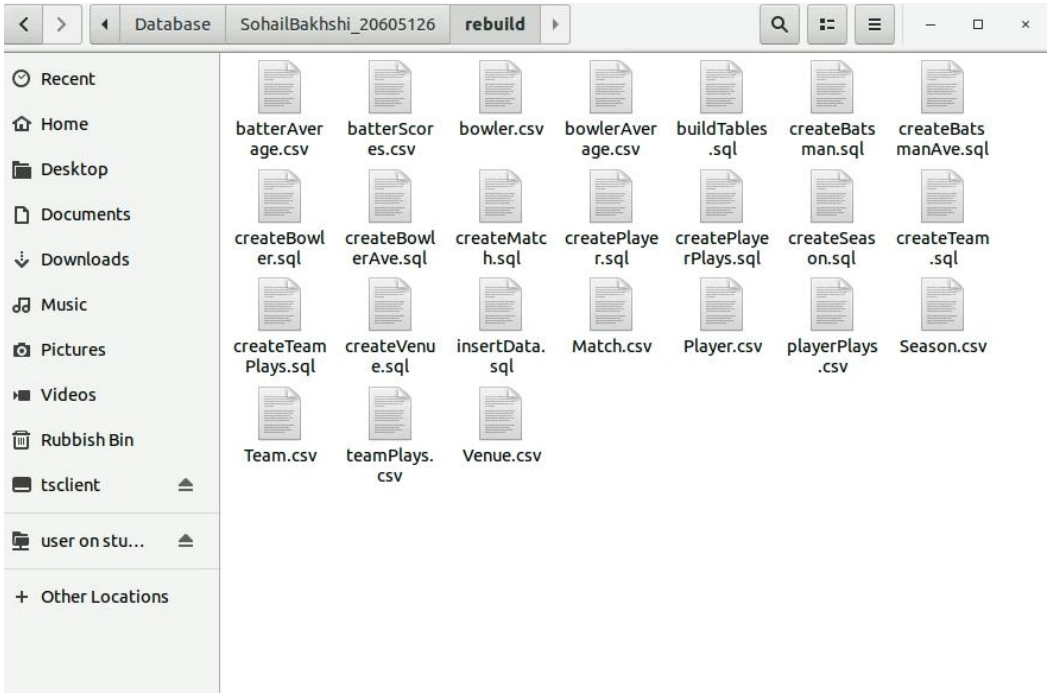
Business Rules	Description
BR1	Player should be over 18
BR2	Team should have at least 11 players in each match
BR3	Each player should have a role

Assumptions

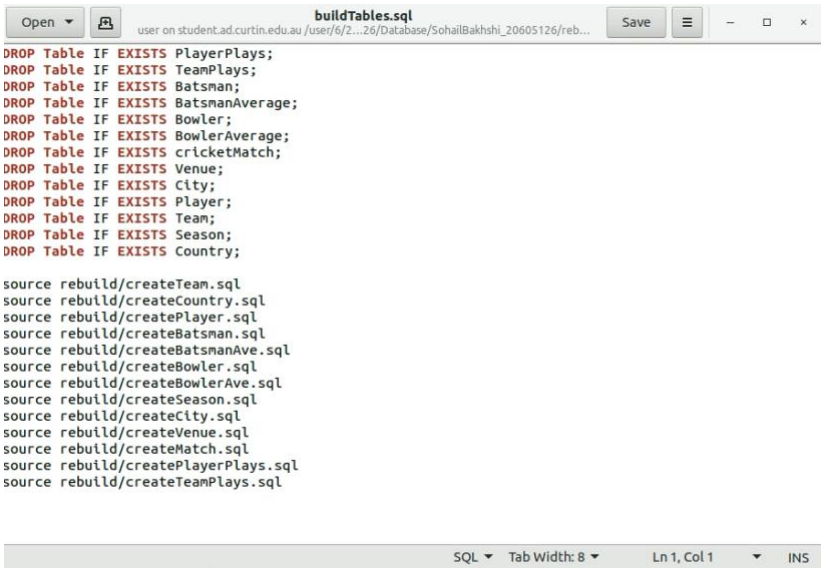
- No team has the same name
- Every player is at least 18 years old
- Primary keys and foreign keys should all be 'not null'
- Role is an attribute of the play's relationship
- All data is legit
- Every match has a winner

Implementation of database

I have implemented my cricket database based on IPL data which came from (Raghunath, 2017) and (Cricinfo, 2022). I began by gathering all the data I needed, and I came to a realisation that I need to alter a lot of the data in order for it to work with my relational schemas I have designed. Originally the csvs had a lot of data, so I had to cut the data down to one season. Within a file called rebuild_20605126.sql I have commands which create and use the database then I source buildTable.sql and insertData.sql which are files inside the rebuild folder.



Within the rebuild folder it contains all of the tables and the csvs that make up the database. I have one script that builds the tables and another script that inserts the data. The .csv files are my cricket data files I've obtained from (Raghunath, 2017) and (cricinfo, 2022) and the files that start with 'create' are files that create the tables.



The buildTables.sql script builds all the tables within the database as displayed above.

Open ▾ insertData.sql user on student.ad.curtin.edu.au /user/6/2...126/Database/SohailBakhshi_20605126/re... Save

```sql
LOAD DATA LOCAL
        INFILE 'rebuild/Team.csv'
        INTO TABLE Team
        FIELDS TERMINATED BY ',' ENCLOSED BY '"'
        LINES TERMINATED BY '\n'
        IGNORE 1 ROWS;


LOAD DATA LOCAL
        INFILE 'rebuild/Player.csv'
        INTO TABLE Player
        FIELDS TERMINATED BY ',' ENCLOSED BY '"'
        LINES TERMINATED BY '\n'
        IGNORE 1 ROWS;


LOAD DATA LOCAL
        INFILE 'rebuild/batterScores.csv'
        INTO TABLE Batsman
        FIELDS TERMINATED BY ',' ENCLOSED BY '"'
        LINES TERMINATED BY '\n'
        IGNORE 1 ROWS;

LOAD DATA LOCAL
        INFILE 'rebuild/batterAverage.csv'
        INTO TABLE BatsmanAverage
        FIELDS TERMINATED BY ',' ENCLOSED BY '"'
        LINES TERMINATED BY '\n'
        IGNORE 1 ROWS;
```

To load in the csv files, I have used 'LOAD DATA LOCAL' which inserts the data straight from the csvs on to the tables. This is accentuated by the example above.

An example of what the tables and csv data might look like is highlighted below.

Open ▾ createPlayer.sql user on student.ad.curtin.edu.au /user/6/...26/Database/SohailBakhshi_20605126/rebu... Save

```sql
#table for my player entity

CREATE TABLE Player(
        playerID INT(4) NOT NULL,
        playerName varchar(36),
        DOB DATE,
        battingHand varchar(24),
        bowlingSkill varchar(36),
        country varchar(16),
        team varchar(36),
        PRIMARY KEY(playerID),
        FOREIGN KEY(country) REFERENCES Country(countryName) ON DELETE RESTRICT,
        FOREIGN KEY(team) REFERENCES Team(teamName) ON DELETE CASCADE

);
```

Player.csv (Remote) - LibreOffice Calc

Player_Id	Player_Name	DOB	Batting_hand	Bowling_skill	Country_Name	Player_team
1	SC Ganguly	1972/07/08	Left-hand bat	Right-arm medium	India	Kolkata Knight Riders
2	BB McCullum	1981/09/27	Right-hand bat	Right-arm medium	New Zealand	Kolkata Knight Riders
3	RT Ponting	1974/12/19	Right-hand bat	Right-arm medium	Australia	Kolkata Knight Riders
4	DJ Hussey	1977/07/15	Right-hand bat	Right-arm offbreak	Australia	Kolkata Knight Riders
5	Mohammad Hafeez	1980/10/17	Right-hand bat	Right-arm offbreak	Pakistan	Kolkata Knight Riders
6	R Dravid	1973/01/11	Right-hand bat	Right-arm offbreak	India	Royal Challengers Bangalore
7	W Jaffer	1978/02/16	Right-hand bat	Right-arm offbreak	India	Royal Challengers Bangalore
8	V Kohli	1988/11/05	Right-hand bat	Right-arm medium	India	Royal Challengers Bangalore
9	JH Kallis	1975/10/16	Right-hand bat	Right-arm fast-medium	South Africa	Royal Challengers Bangalore
10	CL White	1983/08/18	Right-hand bat	Legbreak googly	Australia	Royal Challengers Bangalore
11	MV Boucher	1976/12/03	Right-hand bat	Right-arm medium	South Africa	Royal Challengers Bangalore
12	B Akhil	1977/10/07	Right-hand bat	Right-arm medium-fast	India	Royal Challengers Bangalore
13	AA Noffke	1977/04/30	Right-hand bat	Right-arm fast-medium	Australia	Royal Challengers Bangalore
14	P Kumar	1986/10/02	Right-hand bat	Right-arm medium	India	Royal Challengers Bangalore
15	Z Khan	1978/10/07	Right-hand bat	Left-arm fast-medium	India	Royal Challengers Bangalore
16	SB Joshi	1970/06/06	Left-hand bat	Slow left-arm orthodox	India	Royal Challengers Bangalore
17	PA Patel	1985/03/09	Left-hand bat	NULL	India	Chennai Super Kings
18	ML Hayden	1971/10/29	Left-hand bat	Right-arm medium	Australia	Chennai Super Kings
19	MEK Hussey	1975/05/27	Left-hand bat	Right-arm medium	Australia	Chennai Super Kings
20	MS Dhoni	1981/07/07	Right-hand bat	Right-arm medium	India	Chennai Super Kings
21	SK Raina	1986/11/27	Left-hand bat	Right-arm offbreak	India	Chennai Super Kings

USE OF DATABASE

i) Design and implementation of queries

Disclaimer the outputs don't show the full list of results for some as they wouldn't fit in one screenshot

<u>Basic queries</u>

```
Select playerID, playerName, floor(DATEDIFF(CURDATE(), DOB)/ 365.25) AS Age From Player where
floor(DATEDIFF(CURDATE(), DOB)/ 365.25) between 18 and 35 ;
```

The purpose for this query is because it enables me to determine the current age of the players, which will indicate if they are likely to still be playing today and have not yet retired.

20605126@vdi-1804-cs-027: .../Database/SohailBakhshi_20605126

```
mysql> Select playerID, playerName, floor(DATEDIFF(CURDATE(), DOB)/ 365.25) AS Age From Player w
here floor(DATEDIFF(CURDATE(), DOB)/ 365.25) between 18 and 35 ;
+----------+------------------+------+
| playerID | playerName       | Age  |
+----------+------------------+------+
|        8 | V Kohli          |   33 |
|       21 | SK Raina         |   35 |
|       24 | K Goel           |   35 |
|       30 | T Kohli          |   33 |
|       35 | RA Jadeja        |   33 |
|       47 | PR Shah          |   34 |
|       57 | RG Sharma        |   35 |
|       65 | S Sohal          |   34 |
|       67 | PP Chawla        |   33 |
|       79 | SS Tiwary        |   32 |
|       80 | DS Kulkarni      |   33 |
|       84 | I Sharma         |   34 |
|       85 | AM Rahane        |   34 |
|       91 | VY Mahesh        |   34 |
|       92 | TM Srivastava    |   32 |
|       96 | MK Pandey        |   33 |
|      112 | DB Ravi Teja     |   35 |
|      116 | PJ Sangwan       |   31 |
|      125 | S Anirudha       |   35 |
|      127 | CK Kapugedera    |   35 |
|      132 | SP Goswami       |   33 |
|      134 | U Kaul           |   34 |
|      140 | Iqbal Abdulla    |   32 |
|      142 | PM Sarvesh Kumar |   33 |
```

```
Select playerID, playerName,country from Player where team ='Chennai Super Kings' or team =
'Rajasthan Royals';
```

The reason for this query was to find out which players played in the final match

```
20605126@vdi-1804-cs-027: .../Database/SohailBakhshi_20605126
File  Edit  View  Search  Terminal  Help

mysql> Select playerID, playerName,country from Player where team ='Chennai Super Kings' or
 = 'Rajasthan Royals';
+----------+-----------------+--------------+
| playerID | playerName      | country      |
+----------+-----------------+--------------+
|       17 | PA Patel        | India        |
|       18 | ML Hayden       | Australia    |
|       19 | MEK Hussey      | Australia    |
|       20 | MS Dhoni        | India        |
|       21 | SK Raina        | India        |
|       22 | JDP Oram        | New Zealand  |
|       23 | S Badrinath     | India        |
|       30 | T Kohli         | India        |
|       31 | YK Pathan       | India        |
|       32 | SR Watson       | Australia    |
|       33 | M Kaif          | India        |
|       34 | DS Lehmann      | Australia    |
|       35 | RA Jadeja       | India        |
|       36 | M Rawat         | India        |
|       37 | D Salunkhe      | India        |
|       38 | SK Warne        | Australia    |
|       39 | SK Trivedi      | India        |
|       69 | Kamran Akmal    | Pakistan     |
|       74 | GC Smith        | South Africa |
|       75 | Pankaj Singh    | India        |
|      101 | SA Asnodkar     | India        |
```

```
select matchID,team1,team2, winner from cricketMatch;
```

The reason for this query was to display the winners of each match. This is important as it allows me to know who the better team was.

```
mysql> select matchID,team1,team2, winner from cricketMatch;
+---------+----------------------------+----------------------------+----------------------------+
| matchID | team1                      | team2                      | winner                     |
+---------+----------------------------+----------------------------+----------------------------+
| 335987  | Royal Challengers Bangalore | Kolkata Knight Riders      | Kolkata Knight Riders      |
| 335988  | Kings XI punjab            | Chennai Super Kings        | Chennai Super Kings        |
| 335989  | Delhi Daredevils           | Rajasthan Royals           | Delhi Daredevils           |
| 335990  | Mumbai Indians             | Royal Challengers Bangalore | Royal Challengers Bangalore |
| 335991  | Kolkata Knight Riders      | Deccan Chargers            | Kolkata Knight Riders      |
| 335992  | Rajasthan Royals           | Kings XI punjab            | Rajasthan Royals           |
| 335993  | Deccan Chargers            | Delhi Daredevils           | Delhi Daredevils           |
| 335994  | Chennai Super Kings        | Mumbai Indians             | Chennai Super Kings        |
| 335995  | Deccan Chargers            | Rajasthan Royals           | Rajasthan Royals           |
| 335996  | Kings XI punjab            | Mumbai Indians             | Kings XI punjab            |
| 335997  | Royal Challengers Bangalore | Rajasthan Royals           | Rajasthan Royals           |
| 335998  | Chennai Super Kings        | Kolkata Knight Riders      | Chennai Super Kings        |
| 335999  | Mumbai Indians             | Deccan Chargers            | Deccan Chargers            |
| 336000  | Kings XI punjab            | Delhi Daredevils           | Kings XI punjab            |
| 336001  | Royal Challengers Bangalore | Chennai Super Kings        | Chennai Super Kings        |
| 336002  | Kolkata Knight Riders      | Mumbai Indians             | Mumbai Indians             |
| 336003  | Delhi Daredevils           | Royal Challengers Bangalore | Delhi Daredevils           |
| 336004  | Deccan Chargers            | Kings XI punjab            | Kings XI punjab            |
| 336005  | Rajasthan Royals           | Kolkata Knight Riders      | Rajasthan Royals           |
| 336006  | Chennai Super Kings        | Delhi Daredevils           | Delhi Daredevils           |
| 336007  | Deccan Chargers            | Royal Challengers Bangalore | Royal Challengers Bangalore |
| 336008  | Kings XI punjab            | Kolkata Knight Riders      | Kings XI punjab            |
| 336009  | Mumbai Indians             | Delhi Daredevils           | Mumbai Indians             |
| 336010  | Rajasthan Royals           | Chennai Super Kings        | Rajasthan Royals           |
| 336011  | Royal Challengers Bangalore | Kings XI punjab            | Kings XI punjab            |
```

```
select playerName from Player where bowlingSkill is NOT NULL;
```

This query allows me to find players that have a fast-bowling skill and who would be a strong bowler for the team.

```
20605126@vdi-1804-cs-027: .../Database/SohailBakhshi_20605126
File  Edit  View  Search  Terminal  Help
mysql> select playerName,bowlingSkill from Player where bowlingSkill is NOT NULL and bowlingSkill LIKE '%fast%';
+-----------------+-----------------------+
| playerName      | bowlingSkill          |
+-----------------+-----------------------+
| JH Kallis       | Right-arm fast-medium |
| B Akhil         | Right-arm medium-fast |
| AA Noffke       | Right-arm fast-medium |
| Z Khan          | Left-arm fast-medium  |
| JDP Oram        | Right-arm fast-medium |
| IK Pathan       | Left-arm medium-fast  |
| SR Watson       | Right-arm fast-medium |
| SM Pollock      | Right-arm fast-medium |
| SB Bangar       | Right-arm medium-fast |
| WPUJC Vaas      | Left-arm fast-medium  |
| RP Singh        | Left-arm fast-medium  |
| B Lee           | Right-arm fast        |
| DJ Bravo        | Right-arm medium-fast |
| A Nehra         | Left-arm medium-fast  |
| Pankaj Singh    | Right-arm medium-fast |
| S Sreesanth     | Right-arm fast-medium |
| VRV Singh       | Right-arm medium-fast |
| AB Agarkar      | Right-arm fast        |
| I Sharma        | Right-arm fast-medium |
| R Bhatia        | Right-arm medium-fast |
| MF Maharoof     | Right-arm fast-medium |
| VY Mahesh       | Right-arm medium-fast |
| DW Steyn        | Right-arm fast        |
| DNT Zoysa       | Left-arm fast-medium  |
| D Kalyankrishna | Right-arm medium-fast |
| Sohail Tanvir   | Left-arm medium-fast  |
```

```
select DATEDIFF(MAX(matchDate),MIN(matchDate)) AS 'Season Runtime (Days)'  from cricketMatch;
```

This querys importance is due to it returning how long the season lasted for. This will give an estimate for how long each season will lasts for.

```
mysql> select DATEDIFF(MAX(matchDate),MIN(matchDate)) AS 'Season Runtime (Days)'  from cricketMatch;
+-----------------------+
| Season Runtime (Days) |
+-----------------------+
|                    44 |
+-----------------------+
```

Joins

```
select m.matchID,p.playerName AS 'Man Of The Match' from Player p inner join cricketMatch m on
m.matchMVP = p.playerID;
```

The reason I did this join query was to display the names of the man of the matches for each match as within the cricketMatch table the MVPs where displayed as the playerIDs not names

```
mysql> select m.matchID,p.playerName AS 'Man Of The Match' from Player p inner join cricketMatch m on m.matchMVP = p.playerID;
+---------+-----------------+
| matchID | Man Of The Match |
+---------+-----------------+
| 335987  | BB McCullum      |
| 335988  | MEK Hussey       |
| 335989  | MF Maharoof      |
| 335990  | MV Boucher       |
| 335991  | DJ Hussey        |
| 335992  | SR Watson        |
| 335993  | V Sehwag         |
| 335994  | ML Hayden        |
| 335995  | YK Pathan        |
| 335996  | KC Sangakkara    |
| 335997  | SR Watson        |
| 335998  | JDP Oram         |
| 335999  | AC Gilchrist     |
| 336000  | SM Katich        |
| 336001  | MS Dhoni         |
| 336002  | ST Jayasuriya    |
| 336003  | GD McGrath       |
| 336004  | SE Marsh         |
| 336005  | SA Asnodkar      |
| 336006  | V Sehwag         |
| 336007  | R Vinay Kumar    |
| 336008  | IK Pathan        |
| 336009  | SM Pollock       |
| 336010  | Sohail Tanvir    |
| 336011  | S Sreesanth      |
| 336012  | AC Gilchrist     |
| 336013  | A Nehra          |
| 336014  | MS Dhoni         |
| 336015  | SC Ganguly       |
| 336016  | YK Pathan        |
| 336017  | CRD Fernando     |
| 336018  | L Balaji         |
| 336019  | SC Ganguly       |
| 336020  | SR Watson        |
| 336021  | SE Marsh         |
| 336022  | Shoaib Akhtar    |
| 336023  | ST Jayasuriya    |
```

```
select m.matchID,v.venueName as Venue, country as Country,city AS City from Venue v inner join
cricketMatch m on m.venueID = v.venueID order by m.matchID LIMIT 5;
```

The reason I did this join query was to display the first 5 games and find out what venues they played at for the first 5 games.

```
mysql> select m.matchID,v.venueName as Venue, country as Country,city AS City from Venue v inn
by m.matchID LIMIT 5;
+---------+-----------------------------------+---------+------------+
| matchID | Venue                             | Country | City       |
+---------+-----------------------------------+---------+------------+
| 335987  | M Chinnaswamy Stadium             | India   | Bangalore  |
| 335988  | Punjab Cricket Association Stadium | India   | Chandigarh |
| 335989  | Feroz Shah Kotla                  | India   | Delhi      |
| 335990  | Wankhede Stadium                  | India   | Mumbai     |
| 335991  | Eden Gardens                      | India   | Kolkata    |
+---------+-----------------------------------+---------+------------+
5 rows in set (0.00 sec)
```

```
select playerName from Player p inner join PlayerPlays pp on p.playerID = pp.playerID where
pp.role ='Player' group by playerName;
```

The reason for this join query was to find the roles of each individual player to determine who is a player as the role could be different for each player.

```
mysql> select playerName from Player p inner join PlayerPlays pp on p.playerID = pp.playerID where pp.role ='Player' group by playerName;
+------------------+
| playerName       |
+------------------+
| A Chopra         |
| A Kumble         |
| A Mishra         |
| A Mukund         |
| A Nehra          |
| A Nel            |
| A Symonds        |
| AA Noffke        |
| AB Agarkar       |
| AB de Villiers   |
| AB Dinda         |
| Abdur Razzak     |
| AD Mascarenhas   |
| AM Nayar         |
| AM Rahane        |
| B Akhil          |
| B Chipli         |
| B Geeves         |
| B Lee            |
| BAW Mendis       |
| BB McCullum      |
| BJ Hodge         |
| CK Kapugedera    |
| CL White         |
| CRD Fernando     |
| D Kalyankrishna  |
| D Salunkhe       |
| DB Das           |
| DB Ravi Teja     |
| DJ Bravo         |
| DJ Hussey        |
| DJ Thornely      |
| DL Vettori       |
| DNT Zoysa        |
| DP Vijaykumar    |
| DPMD Jayawardene |
| DR Smith         |
| DS Kulkarni      |
```

```
select p1.playerID,p1.playerName,c.matchID AS 'MVP of match' from Player p1 left outer join
cricketMatch c on c.matchMVP = p1.playerID order by playerID;
```

The reason for this join query was to determine which match a player was man of the match in. If a player was not a man of a match in a match, then NULL should be displayed. If a player was man of the match then it will display the match ID they were man of the match of.

```
mysql> select p1.playerID,p1.playerName,c.matchID AS 'MVP of match' from Player p1 left outer join cricketMatch c on c.matchMVP = p1.playerID order by pla
yerID;
+----------+-----------------+--------------+
| playerID | playerName      | MVP of match |
+----------+-----------------+--------------+
|        1 | SC Ganguly      | 336019       |
|        1 | SC Ganguly      | 336015       |
|        2 | BB McCullum     | 335987       |
|        3 | RT Ponting      | NULL         |
|        4 | DJ Hussey       | 335991       |
|        5 | Mohammad Hafeez | NULL         |
|        6 | R Dravid        | NULL         |
|        7 | W Jaffer        | NULL         |
|        8 | V Kohli         | NULL         |
|        9 | JH Kallis       | NULL         |
|       10 | CL White        | NULL         |
|       11 | MV Boucher      | 335990       |
|       12 | B Akhil         | NULL         |
|       13 | AA Noffke       | NULL         |
|       14 | P Kumar         | 336039       |
|       15 | Z Khan          | NULL         |
|       16 | SB Joshi        | NULL         |
|       17 | PA Patel        | NULL         |
|       18 | ML Hayden       | 335994       |
|       19 | MEK Hussey      | 335988       |
|       20 | MS Dhoni        | 336014       |
|       20 | MS Dhoni        | 336001       |
|       21 | SK Raina        | 336042       |
|       22 | JDP Oram        | 335998       |
|       23 | S Badrinath     | NULL         |
|       24 | K Goel          | NULL         |
|       25 | JR Hopes        | NULL         |
|       26 | KC Sangakkara   | 335996       |
|       27 | Yuvraj Singh    | NULL         |
```

```
select s.orangeCap,b.matchesPlayed,b.innings,b.runs,b.strikeRate,s.purpleCap,b2.matchesPlayed,b2.innings,b2.runs,b2.strikeRate
from Batsman b inner join Season s on b.playerName =s.orangeCap inner join Bowler b2 on b2.playerName = s.purpleCap;
```

The reason for this query was to display the stats of the players that achieved the purple cap and orange cap of the season. This allows us to see what type of score they got in order to achieve this.

```
mysql> select s.orangeCap,b.matchesPlayed,b.innings,b.runs,b.strikeRate,s.purpleCap,b2.matchesPlayed,b2.innings,b2.runs,b2
.strikeRate from Batsman b inner join Season s on b.playerName =s.orangeCap inner join Bowler b2 on b2.playerName = s.purp
leCap;
+-----------+---------------+---------+------+------------+---------------+---------------+---------+------+------------+
| orangeCap | matchesPlayed | innings | runs | strikeRate | purpleCap     | matchesPlayed | innings | runs | strikeRate |
+-----------+---------------+---------+------+------------+---------------+---------------+---------+------+------------+
| SE Marsh  |            11 |      11 |  616 |     139.68 | Sohail Tanvir |            11 |      41 |  266 |      11.20 |
+-----------+---------------+---------+------+------------+---------------+---------------+---------+------+------------+
```

Sub-queries

```
select playerID,playerName from Player where playerID Not IN (select matchMVP from cricketMatch where playerID = matchMVP);
```

The reason for this subquery was to find out which players did not get man of the match in any games.

```
mysql> select playerID,playerName from Player where playerID Not IN (select matchMVP from cricketMatch where playerID = matchMVP)
;
+----------+------------------+
| playerID | playerName       |
+----------+------------------+
|        3 | RT Ponting       |
|        5 | Mohammad Hafeez  |
|        6 | R Dravid         |
|        7 | W Jaffer         |
|        8 | V Kohli          |
|        9 | JH Kallis        |
|       10 | CL White         |
|       12 | B Akhil          |
|       13 | AA Noffke        |
|       15 | Z Khan           |
|       16 | SB Joshi         |
|       17 | PA Patel         |
|       23 | S Badrinath      |
|       24 | K Goel           |
|       25 | JR Hopes         |
|       27 | Yuvraj Singh     |
|       30 | T Kohli          |
|       33 | M Kaif           |
|       34 | DS Lehmann       |
|       35 | RA Jadeja        |
|       36 | M Rawat          |
|       37 | D Salunkhe       |
|       38 | SK Warne         |
|       39 | SK Trivedi       |
|       40 | G Gambhir        |
|       42 | S Dhawan         |
|       43 | L Ronchi         |
```

```
select p1.playerName, floor(DATEDIFF(CURDATE(), DOB)/ 365.25) AS AGE from Player p1 where floor(DATEDIFF(CURDATE(), DOB)/
365.25) IN (select floor(DATEDIFF(CURDATE(), DOB)/ 365.25) from Player p2 where floor(DATEDIFF(CURDATE(), DOB)/ 365.25) > 35);
```

The reason for this subquery was to determine the players that have reached the retirement age. The result is important to know who is no longer playing cricket.

```
mysql>
mysql> select p1.playerName, floor(DATEDIFF(CURDATE(), DOB)/ 365.25) AS AGE from Player p1 where floor(DATEDIFF(CURDATE(), DOB)/
365.25) IN (select floor(DATEDIFF(CURDATE(), DOB)/ 365.25) from Player p2 where floor(DATEDIFF(CURDATE(), DOB)/ 365.25) > 35);
+------------------+------+
| playerName       | AGE  |
+------------------+------+
| SC Ganguly       |   50 |
| BB McCullum      |   41 |
| RT Ponting       |   47 |
| DJ Hussey        |   45 |
| Mohammad Hafeez  |   42 |
| R Dravid         |   49 |
| W Jaffer         |   44 |
| JH Kallis        |   47 |
| CL White         |   39 |
| MV Boucher       |   45 |
| B Akhil          |   45 |
| AA Noffke        |   45 |
| P Kumar          |   36 |
| Z Khan           |   44 |
| SB Joshi         |   52 |
| PA Patel         |   37 |
| ML Hayden        |   50 |
| MEK Hussey       |   47 |
| MS Dhoni         |   41 |
| JDP Oram         |   44 |
| S Badrinath      |   42 |
| JR Hopes         |   44 |
| KC Sangakkara    |   44 |
| Yuvraj Singh     |   40 |
| SM Katich        |   47 |
```

```
select team, count(team) as 'Number of players' from Player where team IN (select team from Player p2 where team = p2.team)
group by team;
```

The reason for this subquery was to determine how many players are in each team. This is important to know as it allows us to know which team is low on team members and whether or not having more members is beneficial.

```
mysql> select team, count(team) as 'Number of players' from Player where team IN (select team from Player p2 where team = p2.team
) group by team;
+----------------------------+-------------------+
| team                       | Number of players |
+----------------------------+-------------------+
| Chennai Super Kings        |                19 |
| Deccan Chargers            |                20 |
| Delhi Daredevils           |                18 |
| Kings XI Punjab            |                21 |
| Kolkata Knight Riders      |                20 |
| Mumbai Indians             |                24 |
| Rajasthan Royals           |                19 |
| Royal Challengers Bangalore |               22 |
+----------------------------+-------------------+
8 rows in set (0.00 sec)
```

```
select team AS Team,playerName AS Captain from Player p where playerID IN (select playerID from PlayerPlays pp where p.playerID
= pp.playerID and pp.role ='Captain');
```

The purpose of this subquery was to determine who the captains are for each team. This allows us to know who is charge of each team and whether or not they should continue.

```
mysql> select team AS Team,playerName AS Captain from Player p where playerID IN (select playerID from PlayerPlays pp where p.playerID = pp.playerID and pp.role ='Captain');
+----------------------------+----------------+
| Team                       | Captain        |
+----------------------------+----------------+
| Kolkata Knight Riders      | SC Ganguly     |
| Royal Challengers Bangalore | R Dravid       |
| Kings XI Punjab            | Yuvraj Singh   |
| Rajasthan Royals           | SK Warne       |
| Delhi Daredevils           | V Sehwag       |
| Mumbai Indians             | Harbhajan Singh |
| Deccan Chargers            | VVS Laxman     |
| Mumbai Indians             | SM Pollock     |
| Mumbai Indians             | SR Tendulkar   |
| Rajasthan Royals           | SR Watson      |
| Chennai Super Kings        | MS Dhoni       |
+----------------------------+----------------+
```

```sql
select b.playerName, b.strikeRate from Bowler b where b.strikeRate > ALL(select MAX(b2.strikeRate) from Bowler b2 group by
b.playerName having b.strikeRate< MAX(b2.strikeRate)-10);
```

The reason for this subquery was to determine which bowler has a strike rate of at least ten less than the max strike rate. The following outputted is important as it displays who has a good strike rate which could benefit the team.

```
mysql> select b.playerName, b.strikeRate from Bowler b where b.strikeRate > ALL(select MAX(b2.strikeRate) from Bowler b2 group by b.playerName
having b.strikeRate< MAX(b2.strikeRate)-10);
+----------------+------------+
| playerName     | strikeRate |
+----------------+------------+
| GD McGrath     |      27.00 |
| M Muralitharan |      31.60 |
| VRV Singh      |      26.10 |
| M Ntini        |      30.00 |
| A Kumble       |      32.80 |
| I Sharma       |      36.10 |
| R Vinay Kumar  |      27.60 |
| PJ Sangwan     |      28.80 |
+----------------+------------+
8 rows in set (0.00 sec)
```

ii) Design and implementation of advanced features

Procedures

Open ▾ | countMatchesPlayed.sql — user on student.ad.curtin.edu.au /user...abase/Soh... | Save

```sql
#procedure that allows you to get the number of matches a player has played
based on the playerID inputted


DROP PROCEDURE IF EXISTS matchesPlayed;
CREATE PROCEDURE matchesPlayed(
        IN ID INT(3),
        OUT ecount INT
)
COMMENT 'counts how many matches a player has played '
SELECT COUNT(playerID) from PlayerPlays where playerID=ID INTO ecount;
```

The purpose of the following procedure is to return the number of matches a player has played based on the player id inputted. The following examples output:

```
mysql> CALL matchesPlayed(150,@playCount);
Query OK, 1 row affected (0.00 sec)

mysql> select @playCount;
+------------+
| @playCount |
+------------+
|          2 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> CALL matchesPlayed(100,@playCount);
Query OK, 1 row affected (0.00 sec)

mysql> select @playCount;
+------------+
| @playCount |
+------------+
|         11 |
+------------+
```

```
mysql> CALL matchesPlayed(69,@playCount);
Query OK, 1 row affected (0.00 sec)

mysql> select @playCount;
+------------+
| @playCount |
+------------+
|          6 |
+------------+
1 row in set (0.00 sec)
```

countPlayers.sql
user on student.ad.curtin.edu.au /user/6/...Databa...

```sql
#counts the number of players based on the team inputted

DROP PROCEDURE IF EXISTS numPlayers;
CREATE PROCEDURE numPlayers(
        IN pTeam VARCHAR(32),
        OUT ecount INT
)
COMMENT 'count the num of players in team'
SELECT COUNT(*) FROM Player WHERE team = pTeam INTO ecount;
```

The following procedure returns the number of players a team has based on the team name inputted. For example:

```
mysql> CALL numPlayers("Mumbai Indians ",@playerCount);
Query OK, 1 row affected (0.00 sec)

mysql> select @playerCount;
+--------------+
| @playerCount |
+--------------+
|           24 |
+--------------+
1 row in set (0.00 sec)
```

```
mysql> CALL numPlayers("Kings XI Punjab",@playerCount);
Query OK, 1 row affected (0.00 sec)

mysql> select @playerCount;
+--------------+
| @playerCount |
+--------------+
|           21 |
+--------------+
1 row in set (0.00 sec)
```

insPlayer.sql
user on student.ad.curtin.edu.au /user/6/20.../Dat...

```sql
#procedure to insert player into player table

DROP PROCEDURE IF EXISTS insPlayer;

DELIMITER //

CREATE PROCEDURE insPlayer(
name varchar(36),
birthday DATE,
country varchar(24),
team varchar(36)
)

COMMENT 'insert into player table'
BEGIN
DECLARE nextid INT(3);
SELECT MAX(playerID)+1 From Player INTO nextid; #adds 1 to max playerID and
thats the new playerID

INSERT INTO
Player(playerID,playerName,DOB,country,team)
VALUES(nextid,name,birthday,country,team);

END//

DELIMITER ;
```

This procedure allows for player insertion. It allows for insertion of name, birthday, country, and team. When using the procedure, it will give the following output:

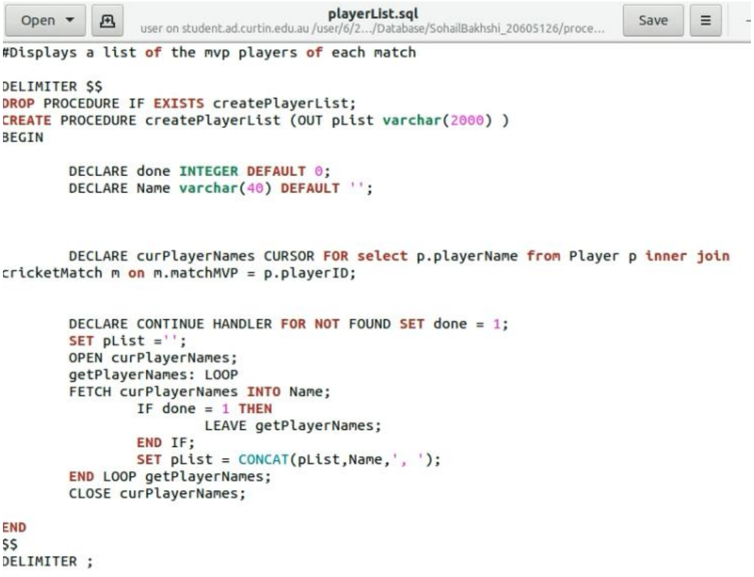
```
mysql> CALL insPlayer("S Bakhshi",'2002/10/04','Australia','Rajasthan Royals');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Player where playerName = 'S Bakhshi';
+----------+------------+------------+-------------+--------------+-----------+------------------+
| playerID | playerName | DOB        | battingHand | bowlingSkill | country   | team             |
+----------+------------+------------+-------------+--------------+-----------+------------------+
|      473 | S Bakhshi  | 2002-10-04 | NULL        | NULL         | Australia | Rajasthan Royals |
+----------+------------+------------+-------------+--------------+-----------+------------------+
1 row in set (0.00 sec)


mysql> CALL insPlayer("Shivam Valdi",'1999/11/01','India','Kings XI Punjab');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Player where playerName = 'Shivam Valdi';
+----------+--------------+------------+-------------+--------------+---------+-----------------+
| playerID | playerName   | DOB        | battingHand | bowlingSkill | country | team            |
+----------+--------------+------------+-------------+--------------+---------+-----------------+
|      474 | Shivam Valdi | 1999-11-01 | NULL        | NULL         | India   | Kings XI Punjab |
+----------+--------------+------------+-------------+--------------+---------+-----------------+
```

Open ▾ | **insVenue.sql** | user on student.ad.curtin.edu.au /user/6/20...6/Database/SohailBakhshi_20605126/pro... | Save

```
#procedure that allows you to insert into venue
#error catching in trigger

DROP PROCEDURE IF EXISTS insVenue;
DELIMITER //
CREATE PROCEDURE insVenue(
venue varchar(36),
city  varchar(36),
country varchar(36)

)

COMMENT 'insert into venue table'
BEGIN

DECLARE nextid INT(3);
SELECT MAX(venueID)+1 From Venue INTO nextid;

INSERT INTO
Venue Values(nextid,venue,city,country);



END //
DELIMITER ;
```

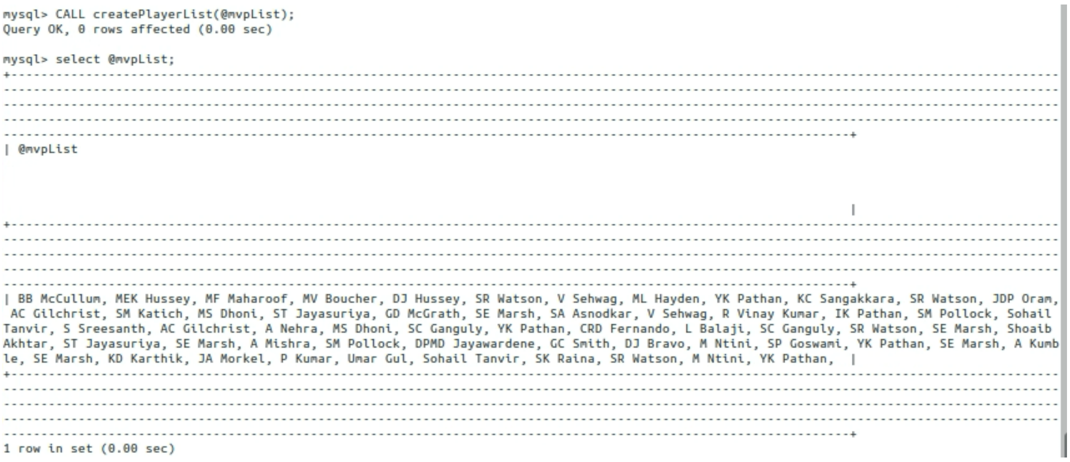
The purpose of this procedure is to insert a new venue location.

```
mysql> select * from Venue;
+---------+-----------------------------------+------------+---------+
| venueID | venueName                         | city       | country |
+---------+-----------------------------------+------------+---------+
|       1 | M Chinnaswamy Stadium             | Bangalore  | India   |
|       2 | Punjab Cricket Association Stadium | Chandigarh | India   |
|       3 | Feroz Shah Kotla                  | Delhi      | India   |
|       4 | Wankhede Stadium                  | Mumbai     | India   |
|       5 | Eden Gardens                      | Kolkata    | India   |
|       6 | Sawai Mansingh Stadium            | Jaipur     | India   |
|       7 | Rajiv Gandhi International Stadium | Hyderabad  | India   |
|       8 | MA Chidambaram Stadium            | Chennai    | India   |
|       9 | Dr DY Patil Sports Academy        | Mumbai     | India   |
+---------+-----------------------------------+------------+---------+
9 rows in set (0.00 sec)

mysql> CALL insVenue('Optus Stadium','Perth','Australia');
Query OK, 1 row affected (0.00 sec)

mysql> select * from Venue;
+---------+-----------------------------------+------------+-----------+
| venueID | venueName                         | city       | country   |
+---------+-----------------------------------+------------+-----------+
|       1 | M Chinnaswamy Stadium             | Bangalore  | India     |
|       2 | Punjab Cricket Association Stadium | Chandigarh | India     |
|       3 | Feroz Shah Kotla                  | Delhi      | India     |
|       4 | Wankhede Stadium                  | Mumbai     | India     |
|       5 | Eden Gardens                      | Kolkata    | India     |
|       6 | Sawai Mansingh Stadium            | Jaipur     | India     |
|       7 | Rajiv Gandhi International Stadium | Hyderabad  | India     |
|       8 | MA Chidambaram Stadium            | Chennai    | India     |
|       9 | Dr DY Patil Sports Academy        | Mumbai     | India     |
|      10 | Optus Stadium                     | Perth      | Australia |
+---------+-----------------------------------+------------+-----------+
```

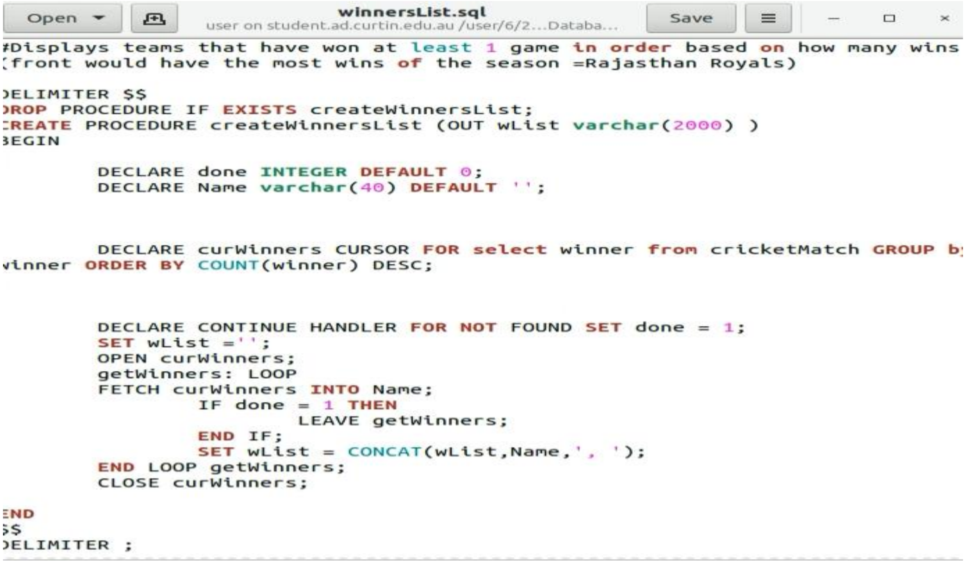


```
#Displays a list of the mvp players of each match

DELIMITER $$
DROP PROCEDURE IF EXISTS createPlayerList;
CREATE PROCEDURE createPlayerList (OUT pList varchar(2000) )
BEGIN

        DECLARE done INTEGER DEFAULT 0;
        DECLARE Name varchar(40) DEFAULT '';



        DECLARE curPlayerNames CURSOR FOR select p.playerName from Player p inner join
cricketMatch m on m.matchMVP = p.playerID;


        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
        SET pList ='';
        OPEN curPlayerNames;
        getPlayerNames: LOOP
        FETCH curPlayerNames INTO Name;
                IF done = 1 THEN
                        LEAVE getPlayerNames;
                END IF;
                SET pList = CONCAT(pList,Name,', ');
        END LOOP getPlayerNames;
        CLOSE curPlayerNames;

END
$$
DELIMITER ;
```

The purpose of this procedure is to get the names of the mvp players and put it into a list.



```
mysql> CALL createPlayerList(@mvpList);
Query OK, 0 rows affected (0.00 sec)

mysql> select @mvpList;
+--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------+
| @mvpList



                                                                                                                     |
+--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------+
| BB McCullum, MEK Hussey, MF Maharoof, MV Boucher, DJ Hussey, SR Watson, V Sehwag, ML Hayden, YK Pathan, KC Sangakkara, SR Watson, JDP Oram,
 AC Gilchrist, SM Katich, MS Dhoni, ST Jayasuriya, GD McGrath, SE Marsh, SA Asnodkar, V Sehwag, R Vinay Kumar, IK Pathan, SM Pollock, Sohail
Tanvir, S Sreesanth, AC Gilchrist, A Nehra, MS Dhoni, SC Ganguly, YK Pathan, CRD Fernando, L Balaji, SC Ganguly, SR Watson, SE Marsh, Shoaib
Akhtar, ST Jayasuriya, SE Marsh, A Mishra, SM Pollock, DPMD Jayawardene, GC Smith, DJ Bravo, M Ntini, SP Goswami, YK Pathan, SE Marsh, A Kumb
le, SE Marsh, KD Karthik, JA Morkel, P Kumar, Umar Gul, Sohail Tanvir, SK Raina, SR Watson, M Ntini, YK Pathan,  |
+--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```



```
#Displays teams that have won at least 1 game in order based on how many wins
(front would have the most wins of the season =Rajasthan Royals)

DELIMITER $$
DROP PROCEDURE IF EXISTS createWinnersList;
CREATE PROCEDURE createWinnersList (OUT wList varchar(2000) )
BEGIN

        DECLARE done INTEGER DEFAULT 0;
        DECLARE Name varchar(40) DEFAULT '';



        DECLARE curWinners CURSOR FOR select winner from cricketMatch GROUP by
winner ORDER BY COUNT(winner) DESC;


        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
        SET wList ='';
        OPEN curWinners;
        getWinners: LOOP
        FETCH curWinners INTO Name;
                IF done = 1 THEN
                        LEAVE getWinners;
                END IF;
                SET wList = CONCAT(wList,Name,', ');
        END LOOP getWinners;
        CLOSE curWinners;

END
$$
DELIMITER ;
```

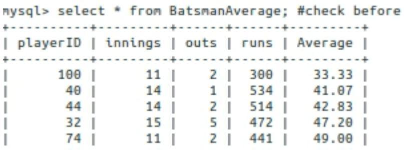
The purpose of this procedure was to produce a list on the teams that have won games and order them from most wins to least wins. The following output is produced when called.

```
mysql> CALL createWinnersList(@winnersList);
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @winnersList;
+------------------------------------------------------------------------------------------------------------------------------------------------+
| @winnersList                                                                                                                                   |
+------------------------------------------------------------------------------------------------------------------------------------------------+
| Rajasthan Royals, Kings XI punjab, Chennai Super Kings, Mumbai Indians, Delhi Daredevils, Kolkata Knight Riders, Royal Challengers Bangalore, Deccan Chargers,  |
+------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

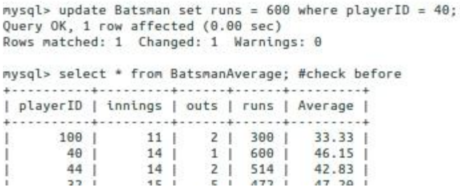
Triggers

```
#trigger that updates the batsman average if runs, innings or outs is changed,
updated

DELIMITER //
DROP TRIGGER IF EXISTS afterUpdatingBatsman;
CREATE TRIGGER afterUpdatingBatsman
        AFTER UPDATE ON Batsman
        FOR EACH ROW
BEGIN
        UPDATE BatsmanAverage SET Average =  NEW.runs/(NEW.innings-
NEW.NumOuts) where playerID = NEW.playerID;
        UPDATE BatsmanAverage SET runs =  NEW.runs where playerID =
NEW.playerID;
        UPDATE BatsmanAverage SET innings =  NEW.innings where playerID =
NEW.playerID;
        UPDATE BatsmanAverage SET outs =  NEW.NumOuts where playerID =
NEW.playerID;
END//



DELIMITER ;
```

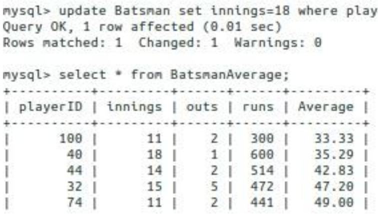
The purpose of this trigger is to update the average of the batter every time their score changes. For example, if the runs go up so will there average but if the number of innings goes up their average will go down if no additional runs are made.



```
mysql> select * from BatsmanAverage; #check before
+----------+---------+------+------+---------+
| playerID | innings | outs | runs | Average |
+----------+---------+------+------+---------+
|      100 |      11 |    2 |  300 |   33.33 |
|       40 |      14 |    1 |  534 |   41.07 |
|       44 |      14 |    2 |  514 |   42.83 |
|       32 |      15 |    5 |  472 |   47.20 |
|       74 |      11 |    2 |  441 |   49.00 |
```



```
mysql> update Batsman set runs = 600 where playerID = 40;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from BatsmanAverage; #check before
+----------+---------+------+------+---------+
| playerID | innings | outs | runs | Average |
+----------+---------+------+------+---------+
|      100 |      11 |    2 |  300 |   33.33 |
|       40 |      14 |    1 |  600 |   46.15 |
|       44 |      14 |    2 |  514 |   42.83 |
```



```
mysql> update Batsman set innings=18 where playerID = 40;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from BatsmanAverage;
+----------+---------+------+------+---------+
| playerID | innings | outs | runs | Average |
+----------+---------+------+------+---------+
|      100 |      11 |    2 |  300 |   33.33 |
|       40 |      18 |    1 |  600 |   35.29 |
|       44 |      14 |    2 |  514 |   42.83 |
|       32 |      15 |    5 |  472 |   47.20 |
|       74 |      11 |    2 |  441 |   49.00 |
```

Open afterUpdatingBowler.sql user on student.ad.curtin.edu.au /user...tabase/Soh... Save

```
#trigger that updates the bowler average if runs or wickets is changed/updatec
DELIMITER //
DROP TRIGGER IF EXISTS afterUpdatingBowler;
CREATE TRIGGER afterUpdatingBowler
        AFTER UPDATE ON Bowler
        FOR EACH ROW
BEGIN
        UPDATE BowlerAverage SET Average =  NEW.runs/NEW.Wickets where
playerID = NEW.playerID;
        UPDATE BowlerAverage SET runs =  NEW.runs where playerID =
NEW.playerID;
        UPDATE BowlerAverage SET wickets =  NEW.wickets where playerID =
NEW.playerID;

END//



DELIMITER ;
```

This procedure works similar to the batsman average trigger. However, if the runs go up and the wickets stay low then the average will be high. If the runs are low but the wickets are high, then the players average will be low.

```
mysql> select * from BowlerAverage; #check before
+----------+------+---------+---------+
| playerID | runs | wickets | Average |
+----------+------+---------+---------+
|      102 |  200 |      22 |    9.09 |
|       38 |  404 |      19 |   21.26 |
|       77 |  442 |      19 |   23.26 |
|       32 |  383 |      17 |   22.52 |
|       67 |  389 |      17 |   22.88 |
|      109 |  399 |      17 |   23.47 |
|      120 |  443 |      17 |   26.05 |
|       91 |  370 |      16 |   23.12 |
```

```
mysql> update Bowler set runs = 500 where playerID = 38;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from BowlerAverage; #check after
+----------+------+---------+---------+
| playerID | runs | wickets | Average |
+----------+------+---------+---------+
|      102 |  200 |      22 |    9.09 |
|       38 |  500 |      19 |   26.32 |
|       77 |  442 |      19 |   23.26 |
|       32 |  383 |      17 |   22.52 |
```

Open beforeInsPlayer.sql user on student.ad.curtin.edu.au /user/6...Database/SohailBakhshi_20605126

```
#trigger that displays an error if player is under 18
DELIMITER //
DROP TRIGGER IF EXISTS beforeInsPlayer;
CREATE TRIGGER beforeInsPlayer
        BEFORE INSERT ON Player
        FOR EACH ROW
        BEGIN
        DECLARE error VARCHAR(100);
        SET error = 'Player must at least 18';
        IF floor(DATEDIFF(CURDATE(), NEW.DOB)/ 365.25) <18 THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = error;
        END IF;

END//
DELIMITER ;
```

This trigger prevents adding players under the age of 18. If a player is under 18 and error will pop up saying player is under 18.

```
mysql> CALL insPlayer('S Bakhshi', '2007/10/04','Afghanistan','Chennai Super Kings'); #wont insert as player is under 18
ERROR 1644 (45000): Player must at least 18
```

Open ▾ | **beforeInsVenue.sql** user on student.ad.curtin.edu.au /user/6.../Database/SohailBakhshi_20605126/triggers | Save

```
#trigger that prevents inserting the same venue
DELIMITER //
DROP TRIGGER IF EXISTS beforeInsVenue;
CREATE TRIGGER beforeInsVenue
        BEFORE INSERT ON Venue
        FOR EACH ROW
        BEGIN
        DECLARE error VARCHAR(100);
        SET error = 'Venue is already in the table';
        IF NEW.venueName IN (select venueName from Venue) and NEW.city IN (select city from Venue)
and NEW.country IN (select country from Venue)  THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = error;
        END IF;

END//
DELIMITER ;
```

This trigger prevents from adding the same venue. If a venue is already in the table, it will display an error. This is important to prevent duplicates and confusion.

```
+---------+-----------------------------------+------------+-----------+
| venueID | venueName                         | city       | country   |
+---------+-----------------------------------+------------+-----------+
|       1 | M Chinnaswamy Stadium             | Bangalore  | India     |
|       2 | Punjab Cricket Association Stadium | Chandigarh | India     |
|       3 | Feroz Shah Kotla                  | Delhi      | India     |
|       4 | Wankhede Stadium                  | Mumbai     | India     |
|       5 | Eden Gardens                      | Kolkata    | India     |
|       6 | Sawai Mansingh Stadium            | Jaipur     | India     |
|       7 | Rajiv Gandhi International Stadium | Hyderabad  | India     |
|       8 | MA Chidambaram Stadium            | Chennai    | India     |
|       9 | Dr DY Patil Sports Academy        | Mumbai     | India     |
|      10 | Optus Stadium                     | Perth      | Australia |
+---------+-----------------------------------+------------+-----------+
10 rows in set (0.00 sec)

mysql> CALL insVenue('Optus Stadium','Perth','Australia'); #it should throw an error as it is already inserted previously
ERROR 1644 (45000): Venue is already in the table
```

VIEWS

```
CREATE VIEW Top5Bowlers AS SELECT playerID,playerName,Team from Bowler ORDER BY Wickets DESC LIMIT
5;
```

This view returns the top 5 bowlers based on the players stats. This is important for finding who the best five batters are of the season

```
mysql> CREATE VIEW Top5Bowlers AS SELECT playerID,playerName,Team from Bowler ORDER BY Wickets DESC LIMIT 5;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> select * from Top5Bowlers;
+----------+---------------+------------------+
| playerID | playerName    | Team             |
+----------+---------------+------------------+
|      102 | Sohail Tanvir | Rajasthan Royals |
|       38 | SK Warne      | Rajasthan Royals |
|       77 | S Sreesanth   | Kings XI Punjab  |
|       32 | SR Watson     | Rajasthan Royals |
|       67 | PP Chawla     | Kings XI Punjab  |
+----------+---------------+------------------+
5 rows in set (0.01 sec)
```

```sql
DROP view if exists Top5Batters;

CREATE VIEW Top5Batters AS SELECT playerID,playerName,team from Batsman ORDER BY runs DESC LIMIT 5;
```

This view returns the top 5 bowlers based on stats. This is important to figure out who the best five bowlers were of the season.

```
ysql> CREATE VIEW Top5Batters AS SELECT playerID,playerName,team from Batsman ORDER BY runs DESC LIMIT 5;
uery OK, 0 rows affected (0.00 sec)

ysql>
ysql> select * from Top5Batters;
+----------+---------------+------------------+
| playerID | playerName    | team             |
+----------+---------------+------------------+
|      100 | SE Marsh      | Kings XI Punjab  |
|       40 | G Gambhir     | Delhi Daredevils |
|       44 | ST Jayasuriya | Mumbai Indians   |
|       32 | SR Watson     | Rajasthan Royals |
|       74 | GC Smith      | Rajasthan Royals |
+----------+---------------+------------------+
 rows in set (0.01 sec)
```

```sql
DROP view if exists seasonWinner;

CREATE VIEW seasonWinner AS SELECT winner AS 'Season 1 Winner',COUNT(winner) AS 'Number of wins'
from cricketMatch GROUP by winner ORDER BY COUNT(winner) DESC LIMIT 1;
```

This view returns the winner of the season based on how many wins they got throughout the entire season.

```
mysql>
mysql> CREATE VIEW seasonWinner AS SELECT winner AS 'Season 1 Winner',COUNT(winner) AS 'Number of wins' from cricketMatch GROUP by winner ORDER BY COUNT(winner) DESC
IMIT 1;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> select * from seasonWinner;
+------------------+----------------+
| Season 1 Winner  | Number of wins |
+------------------+----------------+
| Rajasthan Royals |             13 |
+------------------+----------------+
```

iii) Database connectivity and python implementation

For the python implementation I have utilised the mysql.connector module which permitted me to connect to my database via python.

```
import mysql.connector
from mysql.connector import Error
#connecting to mysql database using python
conn = mysql.connector.connect(
  host="localhost",
  user="me",
  password="myUserPassword",
  database="IPLDatabase_20605126"
)
```

I have two python programs that have different functionalities. The first file is called part5Manual.py. This program already has prewritten queries which all you have to do is run the file and it will display a range of outputs on to the terminal screen. The queries include using procedures, testing triggers, select statements, updates, inserts, deletes, drop, and create.

```
#inserting a player using my procedure
try:

    cursor.callproc('insPlayer',['Sohail bakhshi','2002/10/04','Afghanistan','Chennai Super
Kings'])
    for result in cursor.stored_results():
        print(result.fetchall())

except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute stored procedure named insPlayer")
```

The example above inserts into the player table via my insPlayer procedure.

```
#using a select statement to select from player table
try:

    cursor.execute("SELECT playerID,playerName FROM Player")

    result = cursor.fetchall()
    print("\nDisplaying Players")
    for i in result:
        print(i)
except mysql.connector.Error as error:  #error handling
    print("Sorry failed to execute select statement")
```

The example above utilises a select statement to verify that the procedure worked and inserted into the table.

```
#using a select state to display the venue and city from venue table
try:

    cursor.execute("SELECT venueName,city FROM Venue")

    result = cursor.fetchall()
    print("\nDisplaying Venues")
    for i in result:
        print(i)
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute select statement")
```

The example above was used to display the venues table via a select statement.

```
#updating runs in bowler table
try:

    cursor.execute("UPDATE Bowler SET runs = 900 WHERE playerID =102 ")

    conn.commit()
    print(cursor.rowcount, "record(s) affected in Bowler")
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute update statement")
```

The example above updates the runs in the bowler table to check if trigger is working

```
#check if the trigger works and updates bowler average after updating the bowler table
try:

    cursor.execute("SELECT p.playerName,a.Average FROM BowlerAverage a inner join Player p on
p.playerID = a.playerID")

    result = cursor.fetchall()
    print("\nDisplayling BowlerAverage")
    for i in result:
        print(i)
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute select statement")
```

The example above is used to verify if the average changed after updating the bowler table

```
#checking if it updated by using a select statement
try:

    cursor.execute("SELECT playerID,runs FROM Bowler")

    result = cursor.fetchall()
    print("\nDisplaying Bowler")
    for i in result:
        print(i)
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute select statement")
```

The example above is used to verify if the table got updated in the bowler table

```
#inserting values into cricketMatch table
try:
    statement = "INSERT INTO cricketMatch (matchID,season,venueID) VALUES (%s,%s,%s)"
    val = ('336046',1,4)
    cursor.execute(statement,val)
    conn.commit()
    print(cursor.rowcount, "record inserted.")
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute insert statement {}".format(error))
```

The example above is inserting values into the match table

```
#checking if inserted via select statement
try:

    cursor.execute("SELECT * FROM cricketMatch WHERE matchID =336046")

    result = cursor.fetchall()
    print("\nDisplayling MATCH info")
    for i in result:
        print(i)
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute select statement")
```

The example above is checking if the values inserted previously got inserted

```
#deleting
try:
    cursor.execute("DELETE FROM cricketMatch where matchID=336046")
    print('\nDeleted row')

except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute query {}".format(error))
```

The example above is deleting the row previously inserted.

```
#dropping a table if exists
try:

    cursor.execute("DROP TABLE IF EXISTS newMatch ")
except mysql.connector.Error as error:  #error handling
    print("Sorry failed to execute query {}".format(error))
```

The example above is dropping the table if it exists

```
#creating a new table
try:

    cursor.execute("CREATE TABLE newMatch (matchID CHAR(6), Venue VARCHAR(36))")
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute query {}".format(error))
```

The example above is creating a new table

```
#inserting into the new table
try:
    statement = "INSERT INTO newMatch (matchID,Venue) VALUES (%s,%s)"
    val = ('123456','Wankhede Stadium ')
    cursor.execute(statement,val)
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute query {}".format(error))
```

The example above is inserting into the table I created

```
#checking if values got inserted
try:
    cursor.execute("SELECT * FROM newMatch")
    result = cursor.fetchall()
    print("\nDisplayling newMATCH ")
    for i in result:
        print(i)
except mysql.connector.Error as error: #error handling
    print("Sorry failed to execute query {}".format(error))
```

The example above is verifying if the values I inserted got inserted

The second file is an interactive menu which allows the user to interact with my database. This file is called part5Interactive.py. It contains several functions which include userInterface(), select(), insert(), delete(), createTable(), desc(), show(), menu(). These functions are named after what they should do. For example, select will select from database, insert will insert into a table in the database etc.

```
def menu():
    endLoop = False
    while endLoop == False:
        getInput = None
        if getInput != "x":    #while the input is not x return the menu
            getInput = userInterface()

        if getInput == "x":     #ends the loop
            endLoop = True

        if getInput == "1":
            insert()

        elif getInput == "2":
            select()

        elif getInput == "3":
            delete()

        elif getInput == "4":
            createTable()

        elif getInput == "5":

            update()

        elif getInput == "6":
            desc()

        elif getInput == "7":
            show()
```

The program essentially is just a loop that runs functions based on user input and exist the loop if the user enters x.

```
20605126@vdi-1804-cs-072:.../Database/SohailBakhshi_20605126$ python3 part5Interactive.py

Insert [1]
Select [2]
Delete [3]
Create [4]
Update [5]
Table Structure [6]
Display Tables [7]
Exit   [x]

Please Select an option: x
```

Discussion

Overall, I spent several days developing my database, and while it is likely not perfect, I tried to my best ability to make it perfect. Throughout this assignment, I put a significant amount of effort into constructing the ER diagram, which I had to alter numerous times since I believed it was inaccurate. The most difficult aspect of the task was getting started and then obtaining data that would work well with my design. Once I got the data I desired to utilise, I had to modify it slightly to match with my tables, which necessitated a long time because I didn't know much about how Excel operated and there was a significant amount of data I had to go through. After I got the data ready, I faced issues with inserting the data it into the tables as when using 'load data local' in MySQL it would mess up my tables. In order to combat this, I had to add a new blank column into each csv file which fixed my issue. I'm not too sure if the problem was due to the way I inserted or a MySQL glitch, but my solution was sufficient enough to let me move on to the next stage. One thing I wish I did differently was create less tables. I feel as though I might have over complicated things in this assignment which led me to work on it for longer than I should have. For example, I originally had an additional 3 entities which I removed because I felt as though they had no meaning behind its use. These entities included country, city, and umpire. Although I could have kept them in it was better off to focus on the more important things of a cricket database. When doing my advanced concepts, I may have gone overkill which caused me to spend longer than I should have although it wasn't necessary, I felt as though more is better as it shows that I know what I am doing. Ways I could have improved was by making the project less complicated on myself by keeping it simple with about 5-6 entities rather than 9. I could have also done less advanced concepts such as just doing procedures and views rather than procedures, triggers, and views. Furthermore, to conclude this assignment has demonstrated my ability to design a database and overall, I am happy with what I have accomplished.

References

Cricinfo. 2022. “Indian Premier League, 2007/08 Cricket Team Records & Stats | ESPNcricinfo.com.” ESPNcricinfo. 2022. https://stats.espncricinfo.com/ipl/engine/records/batting/most_runs_career.html?id=3519;type=tournament.

Cricinfo. 2022. “Indian Premier League, 2007/08 Cricket Team Records & Stats | ESPNcricinfo.com.” ESPNcricinfo. 2022. https://stats.espncricinfo.com/ipl/engine/records/bowling/most_wickets_career.html?id=3519;type=tournament.

Fernando, N. (2022) "Database Systems Lectures 1-10". Curtin University. Obtained via Blackboard.

Raghunath. 2017. “Indian Premier League(IPL Cricket) till 2016.” Data.world. data.world. May 22, 2017. https://data.world/raghu543/ipl-data-till-2016-set-of-csv-files.