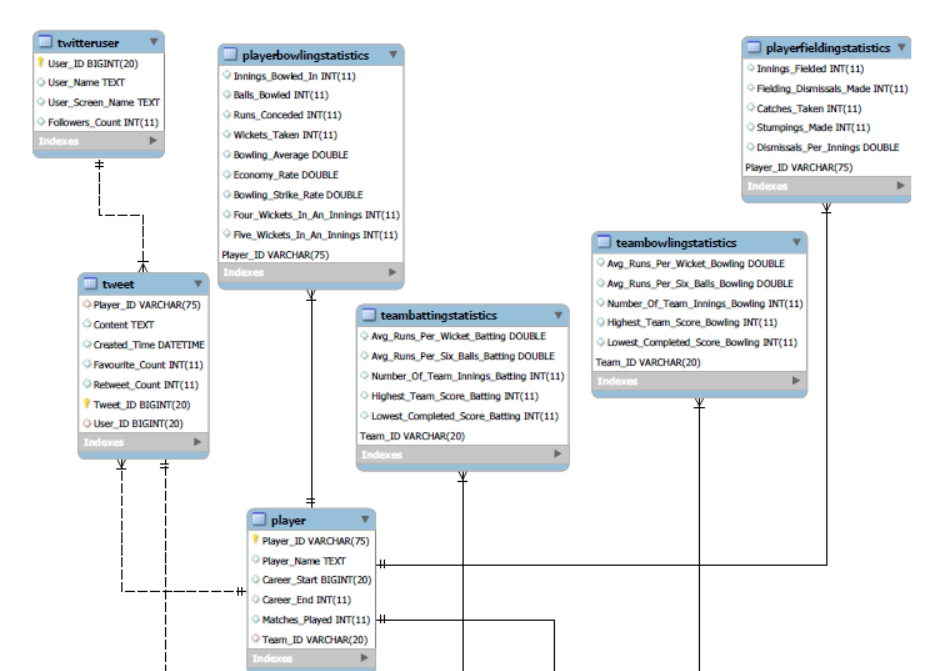
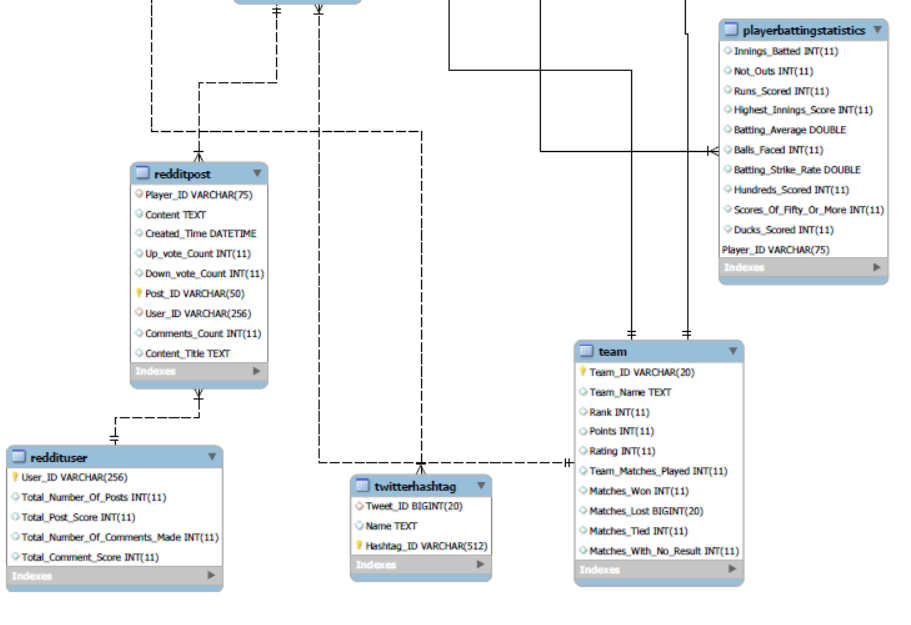
**CRICKET DATABASE – Normalization and Optimization**

**Abstract**

The purpose of the assignment is to normalize and optimize the cricket database, which was created as part of assignment 2. In addition, views for existing use cases, functions and stored procedures have also been added. As part of optimization, indexes have been added to display the increase in query performance.

**Physical model before normalization:**





**1\_NF:**

**Checks made:**

* Each table has a primary key: minimal set of attributes which can uniquely identify a Record
* The values in each column of a table are atomic (No multi-value attributes allowed)
* There are no repeating groups: two columns do not store similar information in the same table.

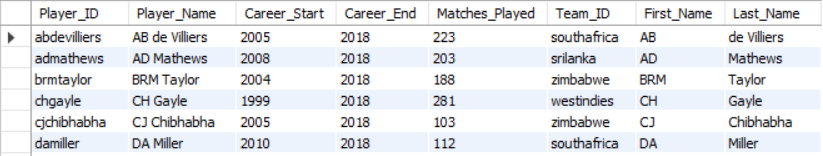
**Player table transformation**

|  |
| --- |
| Player |
| Player\_ID |
| Player Name |
| Career Start |
| Career End |
| Matches Played |
| Team\_ID |

|  |
| --- |
| Player |
| Player\_ID |
| Career Start |
| Career End |
| Matches Played |
| Team\_ID |
| First\_Name |
| Last\_Name |

The left table is the table before transformation and the right table is the table after transformation, where In the Player name is dropped and First and Last Names are added.

SQL showing creation of First\_Name ad Last\_Name



**2-NF**

**Checks made:**

• All requirements for 1st NF must be met.

• No partial dependencies.

• No calculated data.

**playerbattingstatistics table Transformation:**

|  |
| --- |
| Innings\_Batted |
| Not\_Outs |
| Runs\_Scored |
| Highest\_Innings\_Score |
| Batting\_Average |
| Balls\_Faced |
| Batting\_Strike\_Rate |
| Hundreds\_Scored |
| Scores\_Of\_Fifty\_Or\_More |
| Duck\_Outs |
| Player\_ID |

|  |
| --- |
| Innings\_Batted |
| Not\_Outs |
| Runs\_Scored |
| Highest\_Innings\_Score |
| Balls\_Faced |
| Hundreds\_Scored |
| Scores\_Of\_Fifty\_Or\_More |
| Duck\_Outs |
| Player\_ID |

**playerbowlingstatistics table Transformation:**

|  |
| --- |
| Innings\_Bowled |
| Balls\_Faced |
| Runs\_Conceded |
| Wickets\_Taken |
| Four\_Wickets\_In\_An\_Innings |
| Five\_Wickets\_In\_An\_Innings |
| Player\_ID |

|  |
| --- |
| Innings\_Bowled |
| Balls\_Faced |
| Runs\_Conceded |
| Wickets\_Taken |
| Bowling\_Average |
| Economy\_Rate |
| Bowling\_Strike\_rate |
| Four\_Wickets\_In\_An\_Innings |
| Five\_Wickets\_In\_An\_Innings |
| Player\_ID |

|  |
| --- |
| Innings\_Fielded |
| Catches\_Taken |
| Stumpings\_Made |
| Fieldings\_Dismissals\_Made |
| Dismissals\_Per\_Innings |
| Player\_ID |

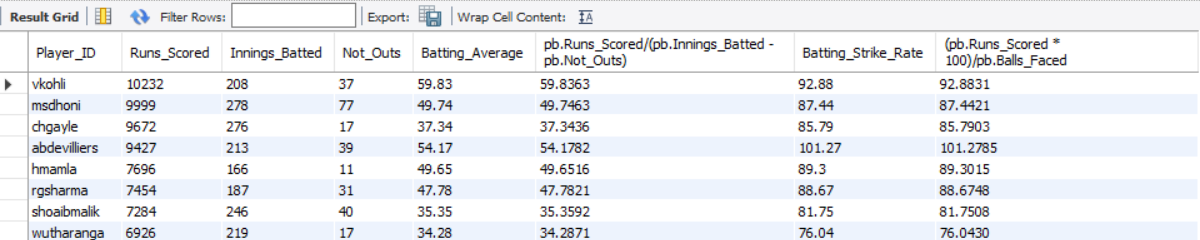
|  |
| --- |
| Innings\_Fielded |
| Catches\_Taken |
| Stumpings\_Made |
| Player\_ID |

**playerfieldingstatistics table Transformation:**

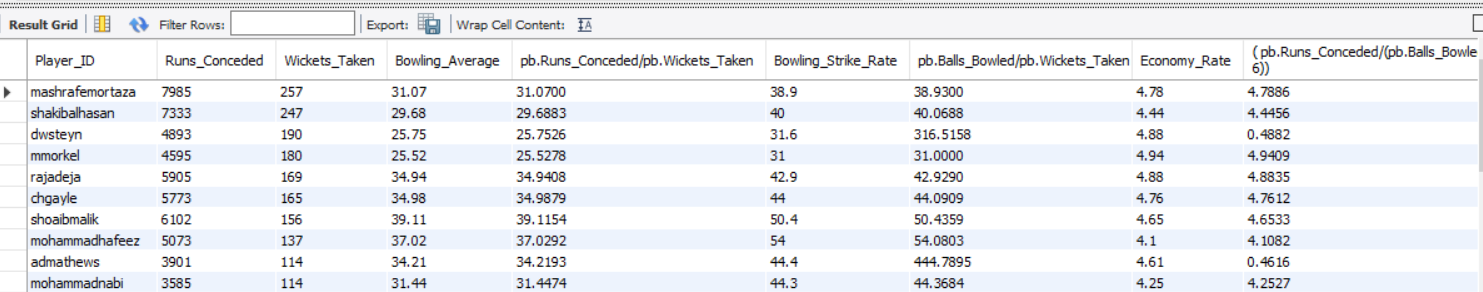
The transformed tables which are in the right side in all three tables

The below changes were made, as they were all calculated values.

1. In the Batting table we have deleted the batting average, batting strike rate columns.

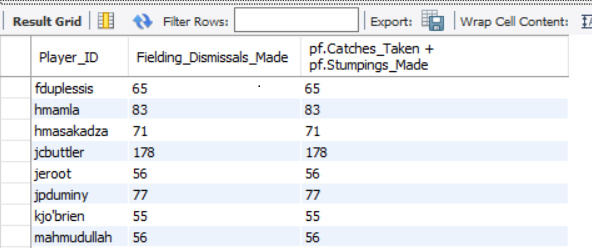


1. In the Bowling table we have deleted the bowling average, economy, bowling strike rate columns.

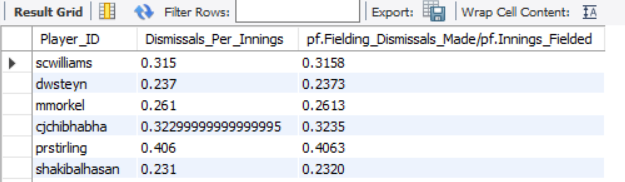


1. In the Fielding table we have deleted the Fieldings\_Dismissals\_Made , Dismissals\_Per\_Innings columns.

Fieldings\_Dismissals\_Made:



Dismissals\_Per\_Innings:



**3-NF:**

**Checks made:**

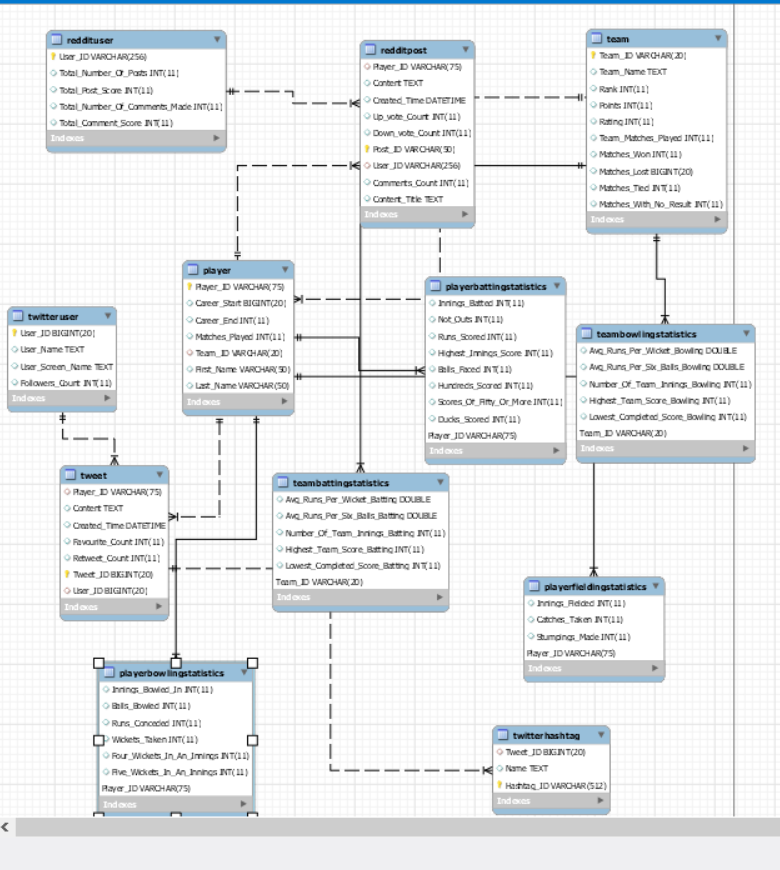
**•** All requirements for 2nd NF must be met.

• Eliminate fields that do not directly depend on the primary key; that is no transitive

dependencies.

No changes were required in the database, as it was already in 3-NF.

Physical model after change:



**#View of Use case (1)**

#Description: Displaying players with most twitter popularity count (Retweet count plus Favourite count)

#Precondition: Only Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: Find the tweets related to players and then calculate the popularity count (Retweet count plus Favourite count) of each tweet, thereby summing up the total for every player.

#Response: Return the list of players and their popularity count.

#Displaying the top 50 popular tweet ids, calculated in terms of sum of retweet count and favourite count

*create view player\_twitter\_popularity as (select p.First\_Name,p.Last\_Name, (SUM(t.Retweet\_Count) + SUM(t.Favourite\_Count)) as "Total\_Popularity\_Count"*

*from cricket\_socialmediaanalysis.player p*

*INNER join cricket\_socialmediaanalysis.tweet t*

*on p.Player\_ID = t.Player\_ID*

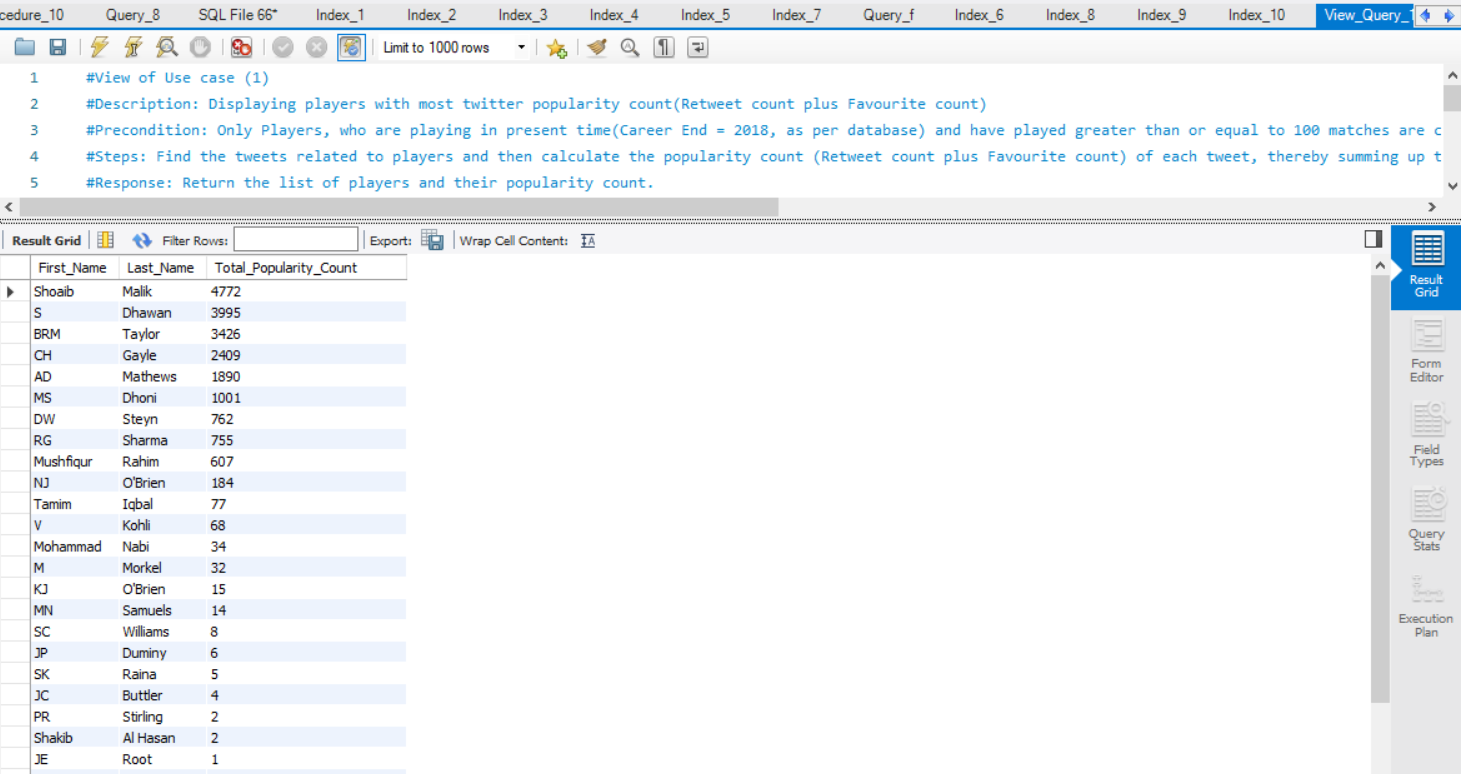
*group by p.First\_Name, p.Last\_Name*

*order by Total\_Popularity\_Count DESC);*

#Executing view:

*select \* from player\_twitter\_popularity;*

Result:



**#View of Use case (2):**

#Description: Displaying players with most reddit popularity count(Up vote count plus Comments count)

#Precondition: Only Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Also, posts under r/Cricket channel were alone considered for evaluation.

#Steps: Find the posts related to players and then calculate the popularity count (Up vote count plus Comment count) of each post, thereby summing up the total for every player.

#Response: Return the list of players and their popularity count.

*create view player\_reddit\_popularity as (select p.First\_Name,p.Last\_Name, (SUM(post.Up\_vote\_Count) + SUM(post.Comments\_Count)) as "Total\_Popularity\_Count"*

*from cricket\_socialmediaanalysis.player p*

*inner join cricket\_socialmediaanalysis.redditpost post*

*on p.Player\_ID = post.Player\_ID*

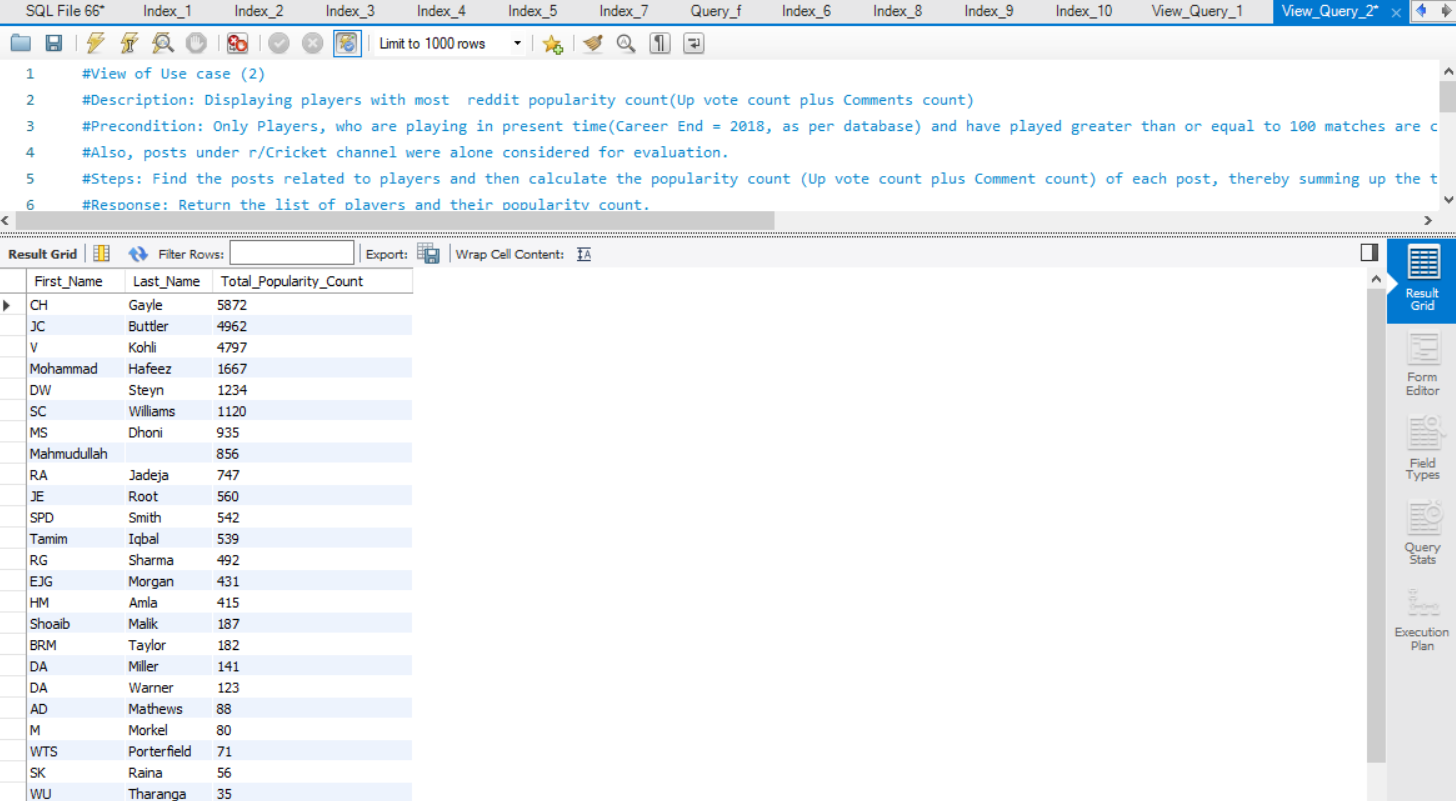
*group by p.First\_Name, p.Last\_Name*

*order by Total\_Popularity\_Count DESC);*

#Executing view:

*select \* from player\_reddit\_popularity;*

Result:



**#View of Use case (3)**

#Description: Displaying tweet count per team from the stored tweets information, thereby accessing team popularity in social media.

#Precondition: Only team with Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: Find the tweets related to players and then relate it to teams, thereby summing the tweets per team.

#Response: Return the list of teams with their related tweet count.

*create view team\_tweet\_count as (select te.Team\_Name,count(Tweet\_ID) as "Tweet\_Count"*

*from cricket\_socialmediaanalysis.tweet t*

*inner join cricket\_socialmediaanalysis.player p*

*on t.Player\_ID = p.Player\_ID*

*inner join cricket\_socialmediaanalysis.team te*

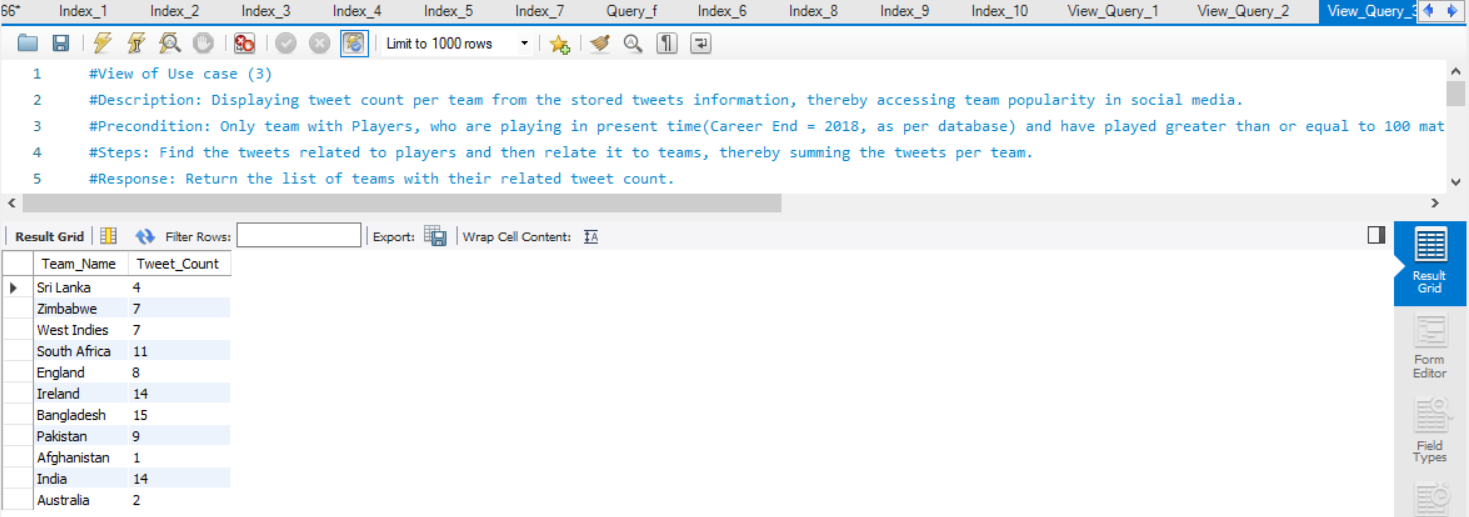
*on te.Team\_ID = p.Team\_ID*

*group by te.Team\_Name);*

#Executing view:

*select \* from team\_tweet\_count;*

Result:



**#View of Use case (4)**

#Description: Displaying number of reddit posts per day for a particular cricket team given as input(India in this case). The resulting date can be correlated with important cricketing events related to the respective team on that day, leading to increased social media presence.

#Precondition: Only Team with Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Also, posts under r/Cricket channel were alone considered for evaluation.

#Steps: Find the posts related to players, who belong to the team given in input and sum up the reddit posts per day, displaying the per day breakdown for the team

#Input: Any cricket team available under ‘team’ table.

#Response: Return the per day breakdown of number of reddit posts for the input team.

*create view reddit\_post\_count\_of\_india\_team\_per\_day as (select DATE(post.Created\_Time) as "Post\_Date", count(post.Post\_ID) 'Number\_Of\_Posts'*

*from cricket\_socialmediaanalysis.redditpost post*

*inner join cricket\_socialmediaanalysis.player p*

*on post.Player\_ID = p.Player\_ID*

*inner join cricket\_socialmediaanalysis.team te*

*on te.Team\_ID = p.Team\_ID*

*where te.Team\_Name = "India"*

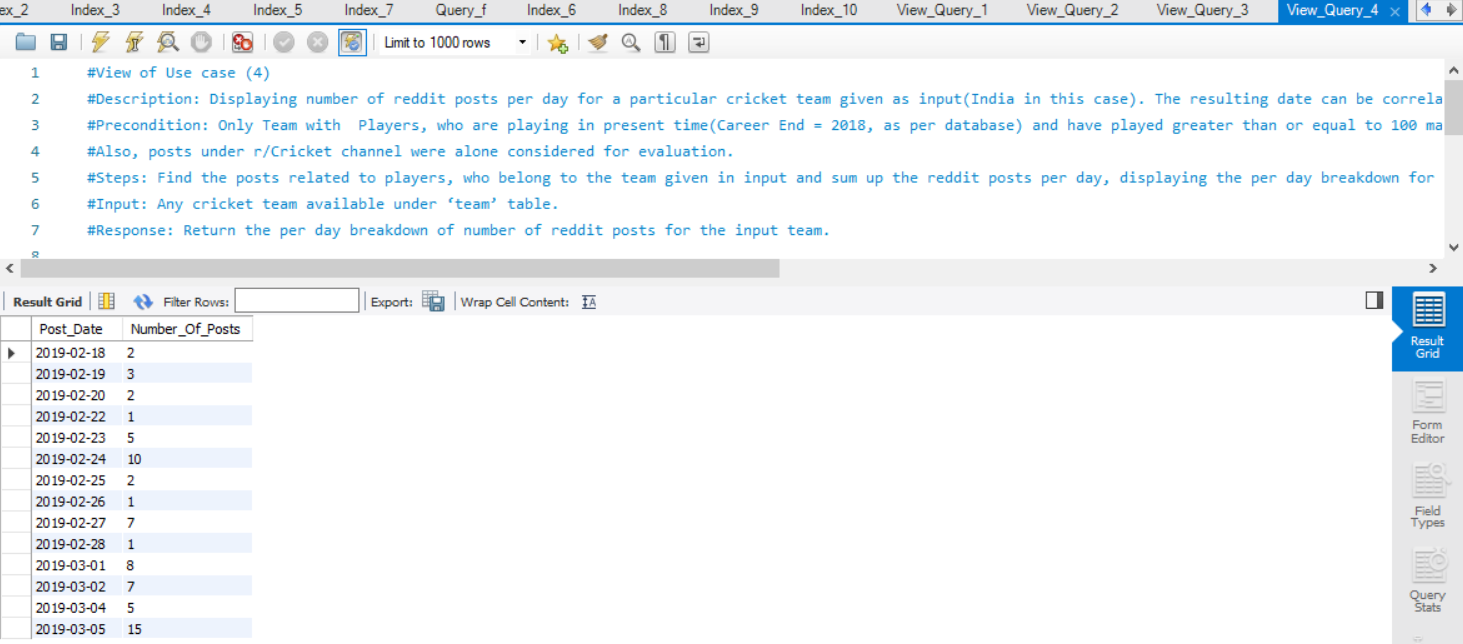
*group by Post\_Date*

*order by Post\_Date);*

#Executing view:

*select \* from reddit\_post\_count\_of\_india\_team\_per\_day;*

Result:



**#View of Use case (5)**

#Description: Displaying number of hashtags used in tweets against each player, thereby assessing the social media popularity of the player from the stored data.

#Precondition: Only Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: Find the hashtags related to tweets , which are again related to players, thereby summing the number of hashtags per player.

#Response: Return the list of players with their related hashtag count from the stored data..

*create view hashtag\_count\_attached\_to\_player as (select p.First\_Name, p.Last\_Name, count(ht.Name) as "Number\_Of\_Hashtags"*

*from cricket\_socialmediaanalysis.twitterhashtag ht*

*inner join cricket\_socialmediaanalysis.tweet t*

*on t.Tweet\_ID = ht.Tweet\_ID*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = t.Player\_ID*

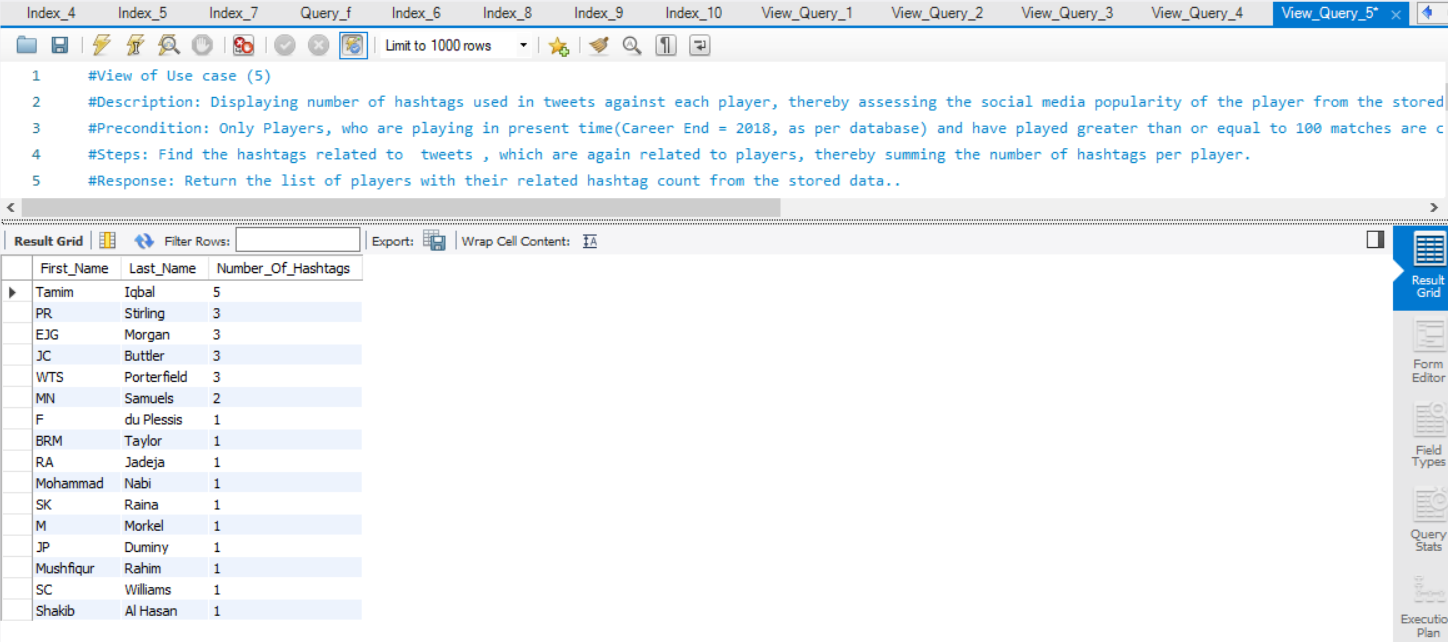
*group by p.First\_Name, p.Last\_Name*

*order by Number\_Of\_Hashtags DESC);*

#Executing view:

*select \* from hashtag\_count\_attached\_to\_player;*

Result:



**#View of Use case (6)**

#Description: Displaying the list of players and their respective teams, who have a batting average greater than 30 and bowling economy rate less than 5, so that they can be classified for all round performance and put up in all round rankings.

#Precondition: Only Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: Find the players who have batting average greater than 30 in a subquery. Then use the list to find the players who have bowling economy rate less than 5.0 in another subquery. Then use this restricted list to find which team the player belongs to.

#Response: Return the list of all round players and their team name.

*create view allround\_players as (select p.First\_Name,p.Last\_Name,t.Team\_Name*

*from cricket\_socialmediaanalysis.player p*

*inner join cricket\_socialmediaanalysis.team t*

*on p.Team\_ID = t.Team\_ID*

*where p.Player\_ID in*

*(select bowlst.Player\_ID*

*from cricket\_socialmediaanalysis.playerbowlingstatistics bowlst*

*where (bowlst.Runs\_Conceded/(bowlst.Balls\_Bowled / 6) <= 5.0) and*

*bowlst.Player\_ID in*

*(select batst.Player\_ID*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*where (batst.Runs\_Scored/(batst.Innings\_Batted - batst.Not\_Outs)> 30))));*

#Executing view:

*select \* from allround\_players;*

Result:



**#View of Use case (7)**

#Description: Displaying the list of players from all teams, who have scored more runs than all players of a particular team(Australia is used in this case).

#Precondition: Only Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: First total runs scored by Australian players is written with a subquery and is compared with total runs scored by all players of other teams using the ‘ALL’ clause.

#Response: Return the list of all players, who have score more runs than every other player in Australia.

*create view players\_with\_more\_runs\_than\_all\_australian\_players as (select p.First\_Name, p.Last\_Name,batst.Runs\_Scored, t.Team\_Name*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

*on p.Team\_ID = t.Team\_ID*

*where batst.Runs\_Scored > all*

*(select batst.Runs\_Scored*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

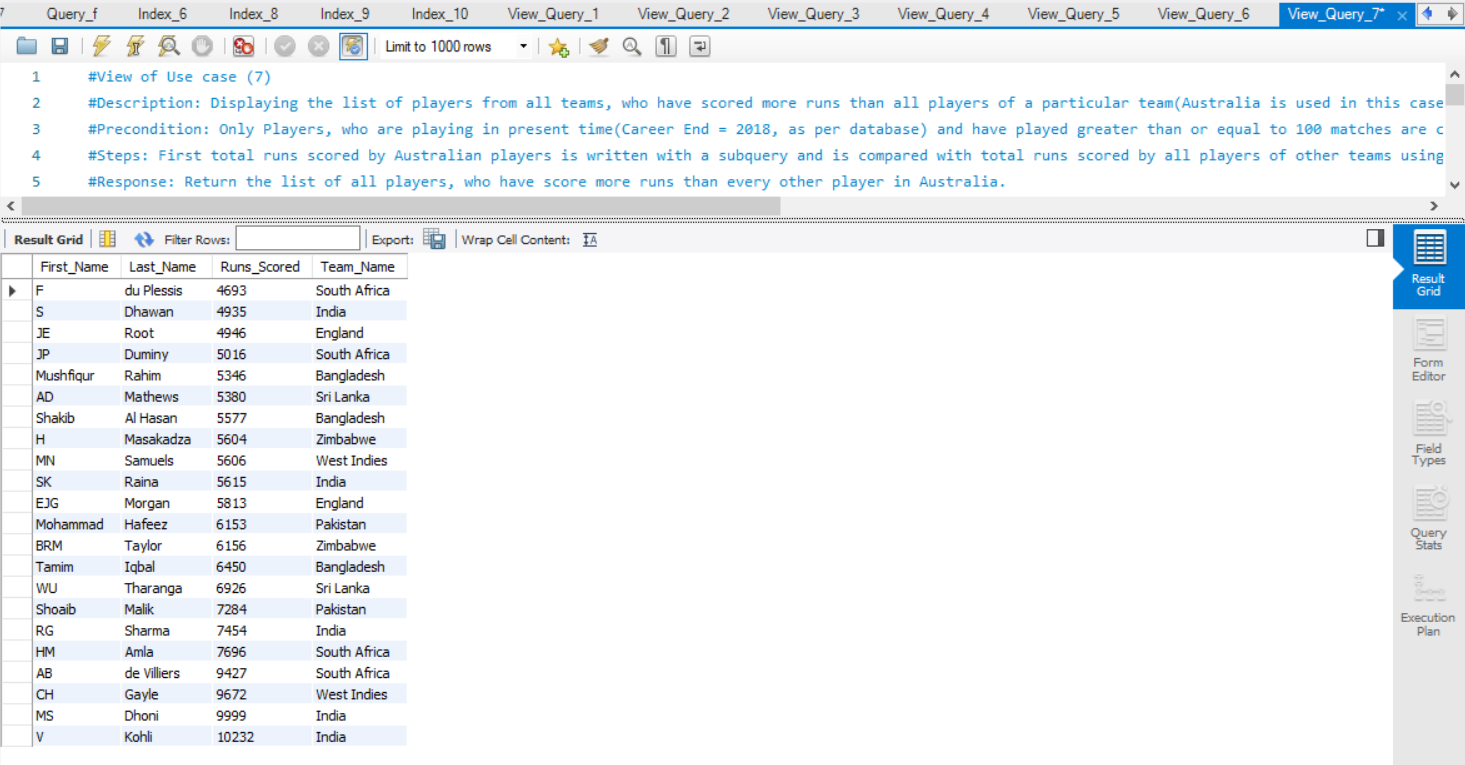
*on p.Team\_ID = t.Team\_ID*

*and t.Team\_Name = "Australia"));*

#Executing view:

*select \* from players\_with\_more\_runs\_than\_all\_australian\_players;*

Result:



**#View of Use case (8)**

#Description:

#Picking the teams with strong bowling figures by evaluating the team with good bowling average first and then considering the good bowlers within it with significant bowling average.

#Precondition: Only Players, who are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: First teams with good bowling average are listed though a subquery and is then used for comparison using ‘Exists’ to find the players within the team with good bowling average.

#Response: Return the list of all players, their respective bowling average and their team name.

*create view teams\_with\_strong\_bowlers as (select First\_Name,Last\_Name, bowst.Runs\_Conceded/bowst.Wickets\_Taken as "Player\_Bowling\_Average"*

*,t.Team\_Name*

*from cricket\_socialmediaanalysis.player p*

*inner join cricket\_socialmediaanalysis.team t*

*on p.Team\_ID = t.Team\_ID*

*inner join cricket\_socialmediaanalysis.playerbowlingstatistics bowst*

*on bowst.Player\_ID = p.Player\_ID*

*where (bowst.Runs\_Conceded/bowst.Wickets\_Taken <= 30.0) and*

*(bowst.Runs\_Conceded/bowst.Wickets\_Taken != 0) and*

*EXISTS*

*(select \**

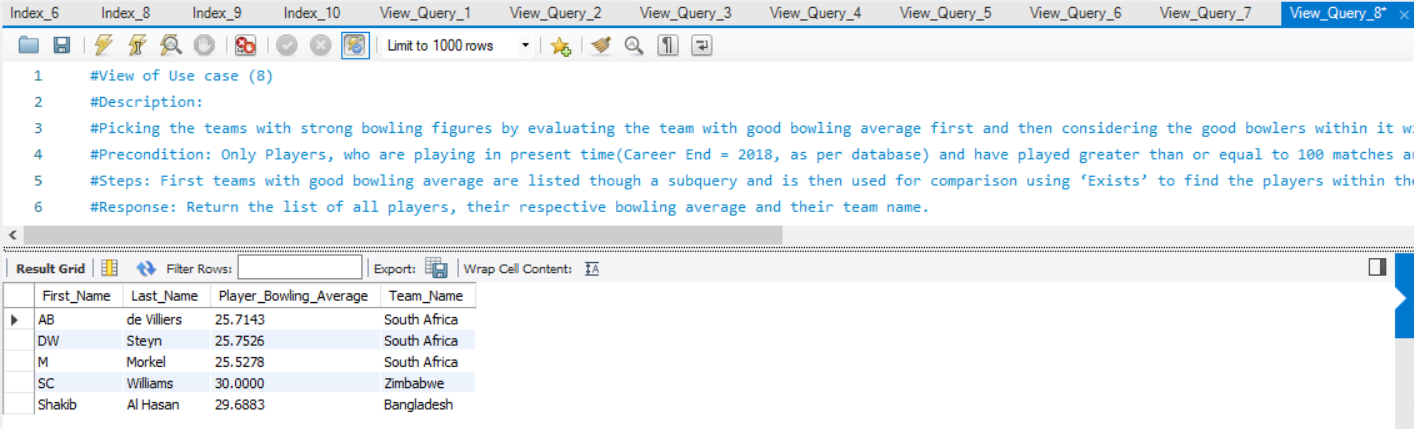
*from cricket\_socialmediaanalysis.teambowlingstatistics tbowst*

*where tbowst.Team\_ID = t.Team\_ID and tbowst.Avg\_Runs\_Per\_Wicket\_Bowling <= 40.0 ));*

#Executing view:

*select \* from teams\_with\_strong\_bowlers;*

Result:



**#View of Use case (9)**

#Description: Displaying the list of players and their respective teams, who had been in the team for more than 15 years for the teams to consider swapping chances for younger players.

#Precondition: Only Players, who have playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: Numerical operation(difference) is performed between ‘Career End’ and ‘Career Start; columns of Player table and is connected with team table to extract the team information

#Response: Return the list of all players, their tenure in the team and their team name.

*create view players\_with\_more\_than\_12\_year\_tenure as (select p.First\_Name,p.Last\_Name, (p.Career\_End - p. Career\_Start ) as "Tenure", t.Team\_Name*

*from cricket\_socialmediaanalysis.player p*

*inner join cricket\_socialmediaanalysis.team t*

*on p.Team\_ID = t.Team\_ID*

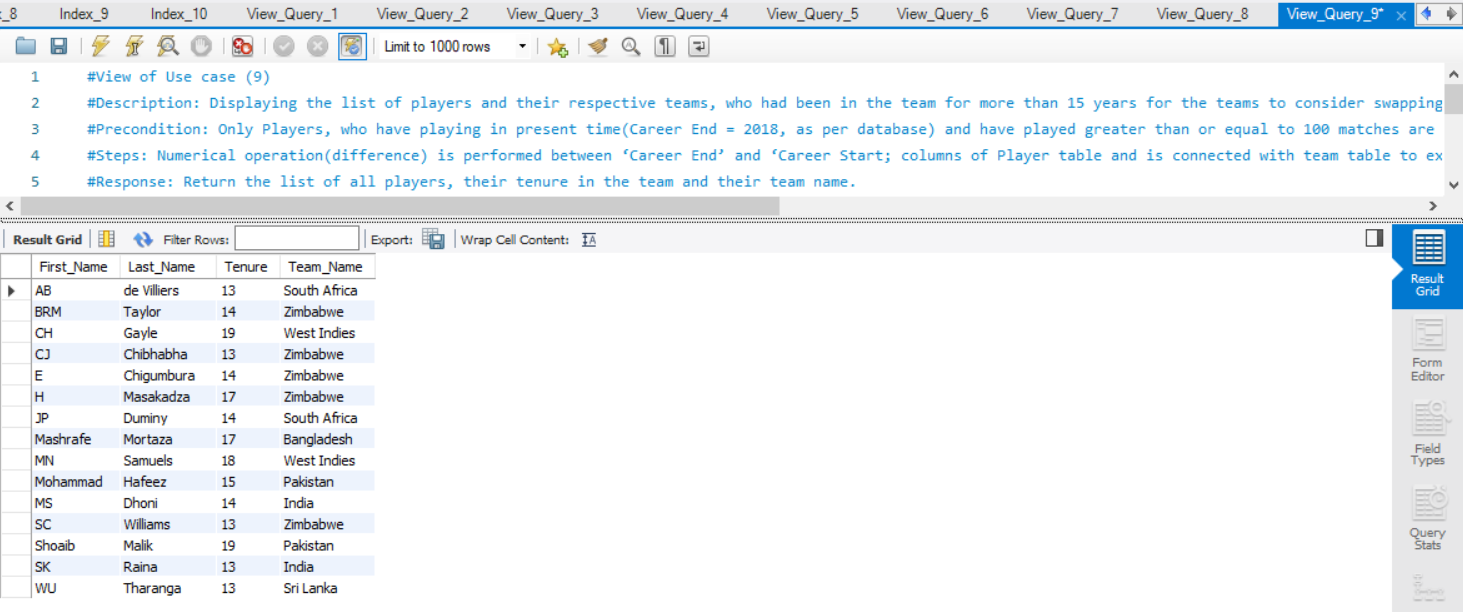
*where (p.Career\_End - p. Career\_Start ) > 12*

*order by p.First\_Name, p.Last\_Name);*

#Executing view:

*select \* from players\_with\_more\_than\_12\_year\_tenure;*

Result:



**#View of Use case (10)**

#Description: Listing the best fielder from every team with their field dismissals figure.

#Precondition: Only Players, who have are playing in present time(Career End = 2018, as per database) and have played greater than or equal to 100 matches are considered.

#Steps: ‘Max()’ aggregate function is applied on field dismissals value of players and the best player is from every country is picked up group by function.

#Response: Return the list of all players, their field dismissals figure and their team name.

*create view top\_fielders\_per\_team as (select p.First\_Name, p.Last\_Name, max(pfs.Catches\_Taken + pfs.Stumpings\_Made) as "Field\_Dismissals", t.Team\_Name*

*from cricket\_socialmediaanalysis.player p*

*inner join cricket\_socialmediaanalysis.playerfieldingstatistics pfs*

*on p.Player\_ID = pfs.Player\_ID*

*inner join cricket\_socialmediaanalysis.team t*

*on p.Team\_ID = t.Team\_ID*

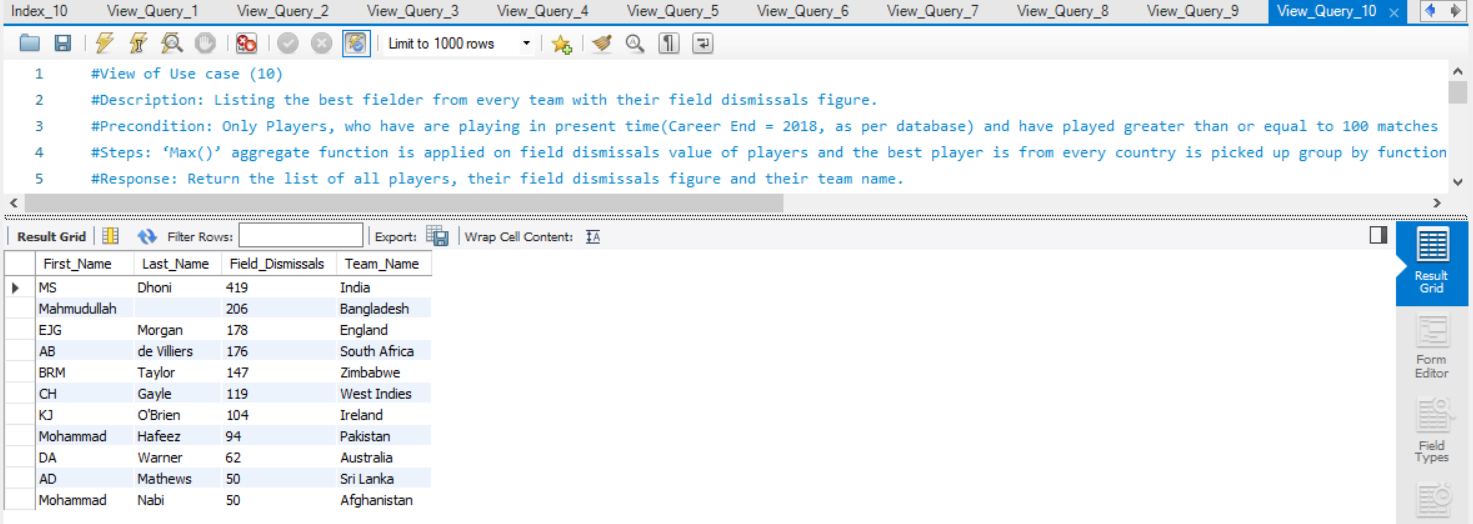
*group by t.Team\_Name*

*order by Field\_Dismissals DESC);*

#Executing view:

*select \* from top\_fielders\_per\_team;*

Result:



**FUNCTIONS**

**#Function (1) : number\_of\_tweets()**

#Description: Accepts username as input and returns the total number of tweets made by the user.

#Joins used: Tables 'tweet' and 'twitteruser' are joined usimg 'User\_ID'.

DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `number\_of\_tweets`(user\_name text) RETURNS int

READS SQL DATA

BEGIN

DECLARE number\_of\_tweets INT;

*SELECT count(t.Tweet\_ID) INTO number\_of\_tweets*

*FROM cricket\_socialmediaanalysis.tweet t, cricket\_socialmediaanalysis.twitteruser u*

*WHERE t.User\_ID = u.User\_ID and u.User\_Name = user\_name;*

*RETURN number\_of\_tweets;*

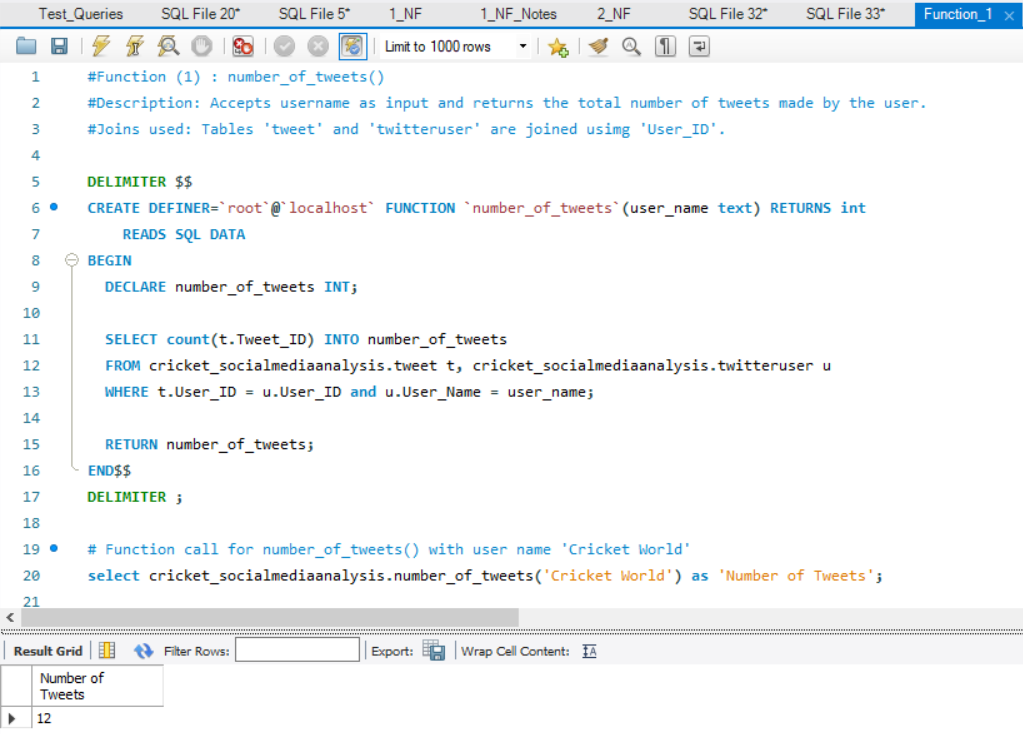
*END$$*

*DELIMITER ;*

# Function call for number\_of\_tweets() with user name 'Cricket World'

select cricket\_socialmediaanalysis.number\_of\_tweets('Cricket World') as 'Number of Tweets';

Result:



**#Function (2) : number\_of\_reddit\_posts()**

#Description: Accepts user\_id as input and returns the total number of reddit posts made by the user.

#Joins used: Table 'redditpost' is alone used.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` FUNCTION `number\_of\_reddit\_posts`(user\_id varchar(256)) RETURNS int*

READS SQL DATA

*BEGIN*

*DECLARE number\_of\_reddit\_posts INT;*

*SELECT count(p.Post\_ID) INTO number\_of\_reddit\_posts*

*FROM cricket\_socialmediaanalysis.redditpost p*

*WHERE p.User\_ID = user\_id;*

*RETURN number\_of\_reddit\_posts;*

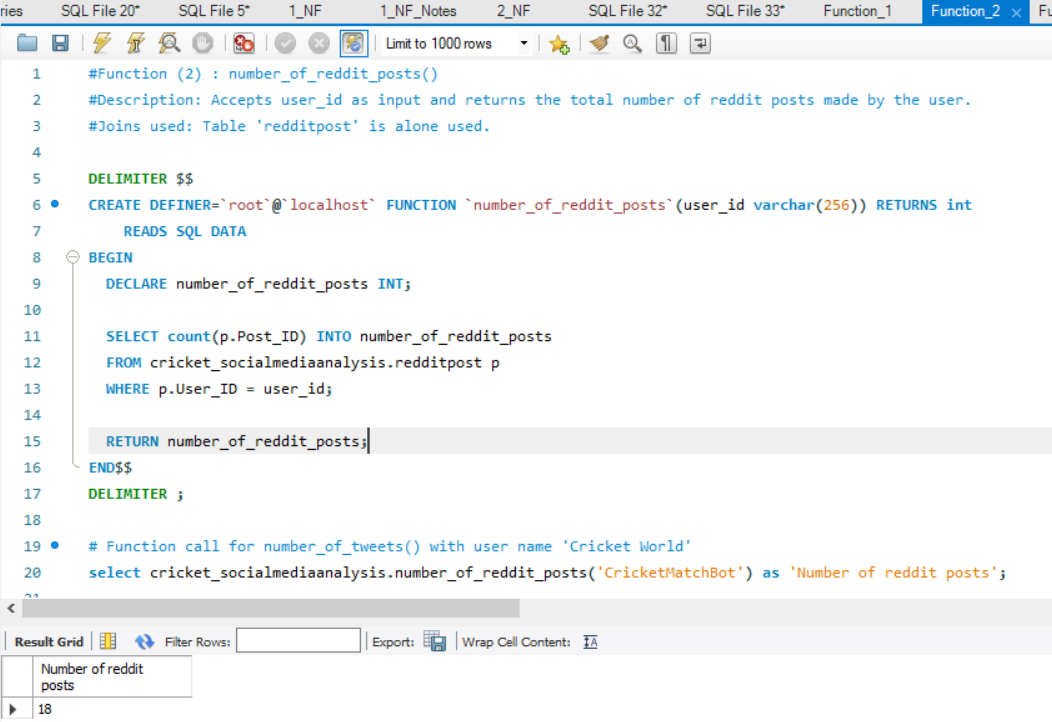
*END$$*

*DELIMITER ;*

# Function call for number\_of\_tweets() with user name 'Cricket World'

select cricket\_socialmediaanalysis.number\_of\_reddit\_posts('CricketMatchBot') as 'Number of reddit posts';

Result:



#Function (3) : players\_attached\_to\_a\_hashtag()

#Description: Accepts hashtag name as input and returns the concatenated list of player names(inner concatenation of

#first name and last name), about whom tweets were posted with the input hashtag.

#Joins used:

#1. Tables 'tweet' and 'twitterhashtag' are joined using 'Tweet\_ID'

#2. Tables 'tweet' and 'player' are joined using 'Player\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` FUNCTION `players\_attached\_to\_a\_hashtag`(hashtag\_name text) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE player\_names text;*

*select group\_concat( distinct (concat\_ws (' ', p.First\_Name, p.Last\_Name))) INTO player\_names*

*from cricket\_socialmediaanalysis.tweet t, cricket\_socialmediaanalysis.twitterhashtag h,*

*cricket\_socialmediaanalysis.player p*

*where t.Tweet\_ID = h.Tweet\_ID and t.Player\_ID = p.Player\_ID*

*and h.Name='Cricket';*

*RETURN player\_names;*

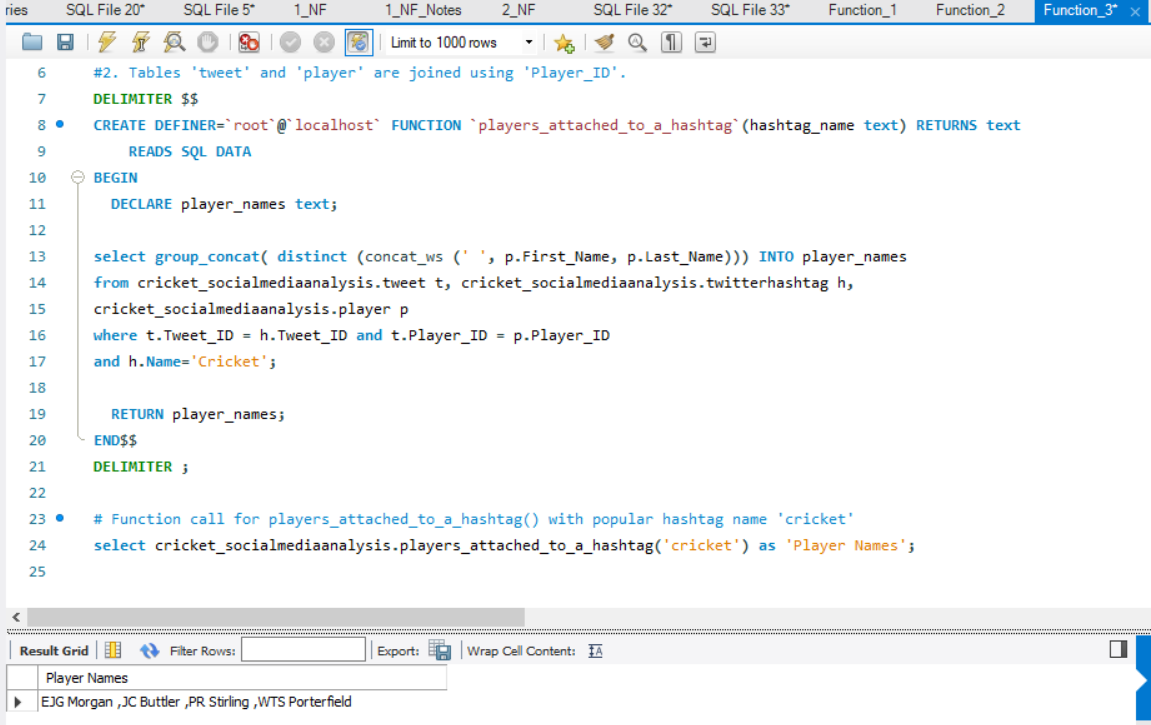
*END$$*

DELIMITER ;

# Function call for players\_attached\_to\_a\_hashtag() with popular hashtag name 'cricket'

select cricket\_socialmediaanalysis.players\_attached\_to\_a\_hashtag('cricket') as 'Player Names';

Result:



**#Function (4) : player\_names\_attached\_to\_reddit\_posts\_on\_selected\_day()**

#Description: Accepts date as input and returns the concatenated list of player names(inner concatenation of

#first name and last name), about whom reddit posts were posted on that particualr day.

#Joins used: Tables 'redditpost' and 'player' are joined using 'Player\_ID'.

DELIMITER $$

*CREATE DEFINER=`root`@`localhost` FUNCTION `player\_names\_attached\_to\_reddit\_posts\_on\_selected\_day`(dateofconcern datetime) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE player\_names text;*

*select group\_concat( distinct (concat\_ws (' ', p.First\_Name, p.Last\_Name))) INTO player\_names*

*from cricket\_socialmediaanalysis.redditpost r, cricket\_socialmediaanalysis.player p*

*where r.Player\_ID = p.Player\_ID and*

*r.Created\_Time between (dateofconcern) and timestamp (dateofconcern, '23:59:59');*

*RETURN player\_names;*

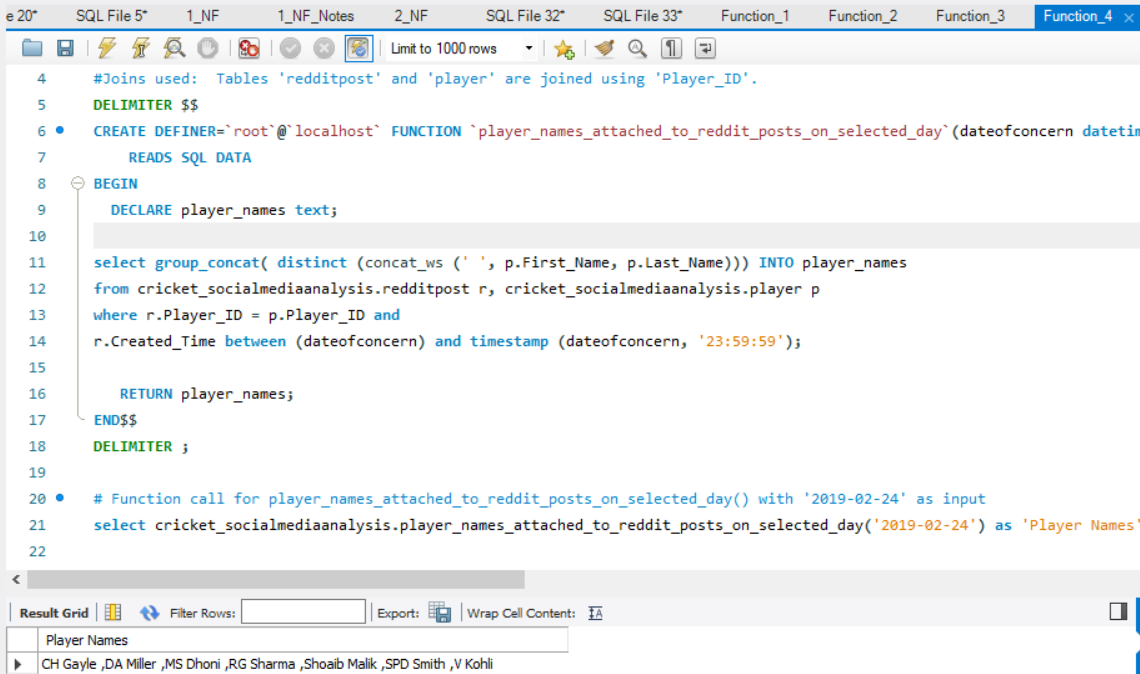
*END$$*

*DELIMITER ;*

# Function call for player\_names\_attached\_to\_reddit\_posts\_on\_selected\_day() with '2019-02-24' as input

select cricket\_socialmediaanalysis.player\_names\_attached\_to\_reddit\_posts\_on\_selected\_day('2019-02-24') as 'Player Names';

Result:



**#Function (5) : dates\_of\_team\_tweets()**

#Description: Accepts team name as input and returns the concatenated list of dates(extracting only date from datetime stamp),

#on which twwets were made about players from the input team.

#Joins used:

#1.Tables 'tweet' and 'player' are joined using 'Player\_ID'.

#2.Tables 'player' and 'team' are joined using 'Team\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` FUNCTION `dates\_of\_team\_tweets`(teamname text) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE dates text;*

*select group\_concat(DISTINCT(Date(t.Created\_Time))) INTO dates*

*from cricket\_socialmediaanalysis.tweet t, cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.team team*

*where t.Player\_ID = p.Player\_ID and team.Team\_ID = p.Team\_ID and team.Team\_Name like teamname;*

*RETURN dates;*

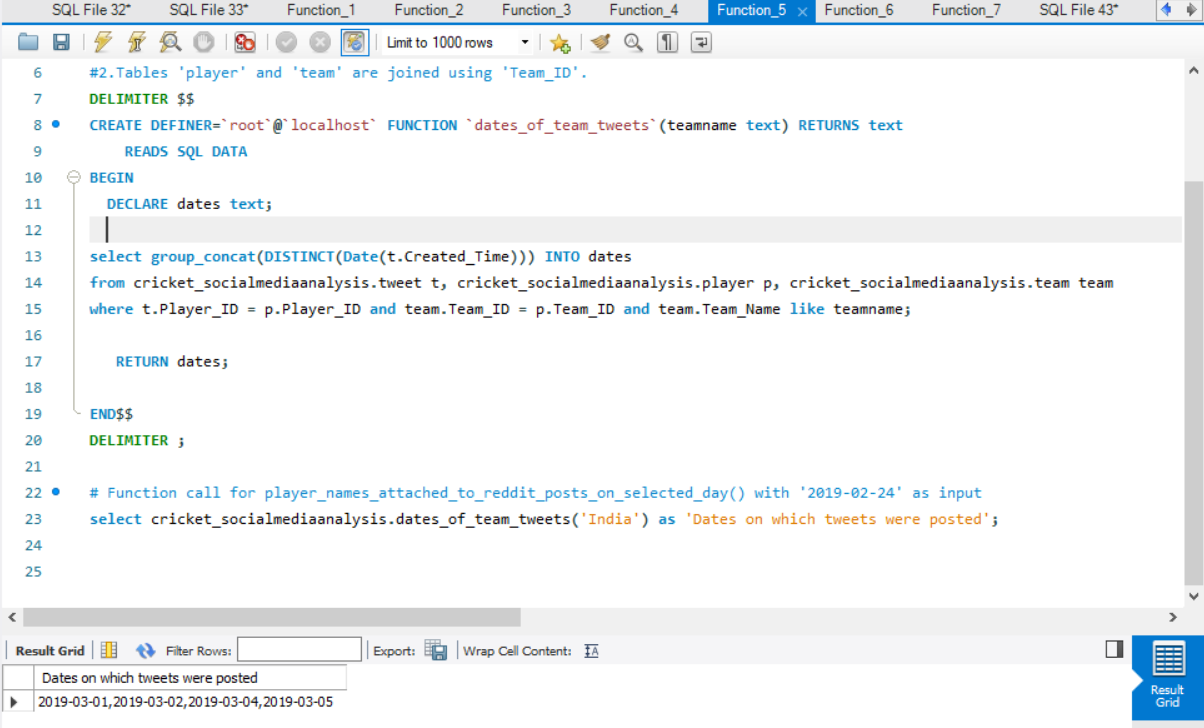
*END$$*

*DELIMITER ;*

# Function call for player\_names\_attached\_to\_reddit\_posts\_on\_selected\_day() with '2019-02-24' as input

select cricket\_socialmediaanalysis.dates\_of\_team\_tweets('India') as 'Dates on which tweets were posted';

Result:



**#Function (6) : players\_with\_hundrerds\_in\_selective\_innings()**

#Description: Accepts 'number\_of\_hundreds' and 'number\_of\_innings' as input parameters

#and returns the concatenated list of player names(inner concatenation of

#first name and last name), who had scored more than particular nunber of hundrerds in more than particualar number

#of matches given as input.

#Joins used: Tables 'player' and 'playerbattingstatistics' are joined using 'Player\_ID'.

DELIMITER $$

*CREATE DEFINER=`root`@`localhost` FUNCTION `players\_with\_hundrerds\_in\_selective\_innings`(number\_of\_hundreds int,*

*number\_of\_innings int) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE player\_names text;*

*select group\_concat( distinct (concat\_ws (' ', p.First\_Name, p.Last\_Name))) INTO player\_names*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerbattingstatistics pb*

*where p.Player\_ID = pb.Player\_ID and pb.Innings\_Batted >= number\_of\_innings and pb.Hundreds\_Scored >= number\_of\_hundreds;*

*RETURN player\_names;*

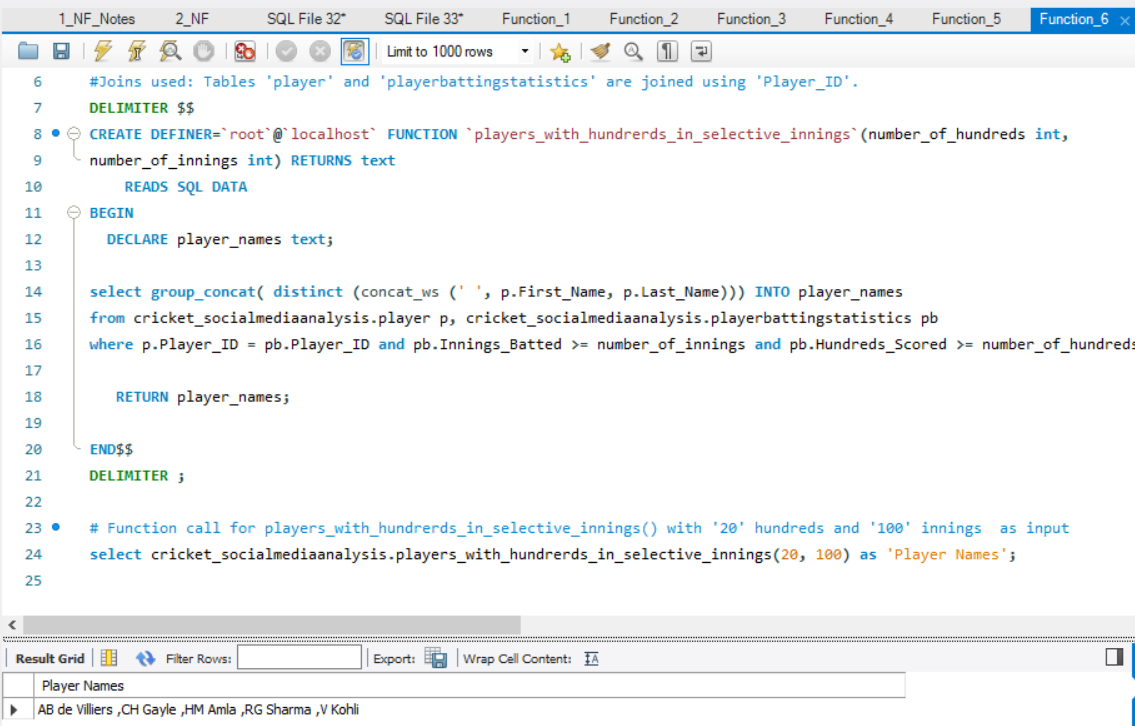
*END$$*

*DELIMITER ;*

# Function call for players\_with\_hundrerds\_in\_selective\_innings() with '20' hundreds and '100' innings as input

select cricket\_socialmediaanalysis.players\_with\_hundrerds\_in\_selective\_innings(20, 100) as 'Player Names';

Result:



**#Function (7) : players\_with\_four\_wicket\_records\_in\_selective\_innings()**

#Description: Accepts 'number\_of\_four\_wicket\_records' and 'number\_of\_innings' as input parameters

#and returns the concatenated list of player names(inner concatenation of

#first name and last name), who had obtained more than particular nunber of 'four-wicket-record'

#in more than particualar number of innings given as input.

#Joins used: Tables 'player' and 'playerbowlingstatistics' are joined using 'Player\_ID'.

DELIMITER $$

*CREATE DEFINER=`root`@`localhost` FUNCTION `players\_with\_four\_wicket\_records\_in\_selective\_innings`*

*(number\_of\_4\_wicket\_records int, number\_of\_innings int) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE player\_names text;*

*select group\_concat( distinct (concat\_ws (' ', p.First\_Name, p.Last\_Name))) INTO player\_names*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerbowlingstatistics pb*

*where p.Player\_ID = pb.Player\_ID and pb.Innings\_Bowled\_In >= number\_of\_innings*

*and pb.Four\_Wickets\_in\_An\_Innings >= number\_of\_4\_wicket\_records;*

*RETURN player\_names;*

*END$$*

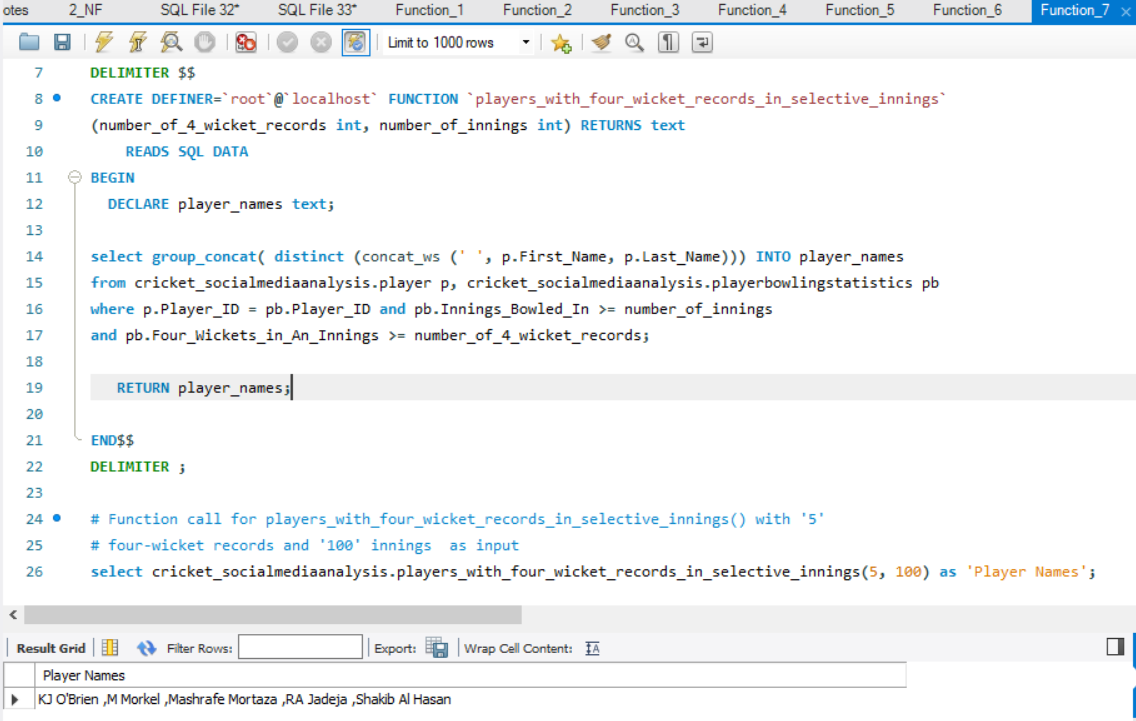
*DELIMITER ;*

# Function call for players\_with\_four\_wicket\_records\_in\_selective\_innings() with '5'

# four-wicket records and '100' innings as input

select cricket\_socialmediaanalysis.players\_with\_four\_wicket\_records\_in\_selective\_innings(5, 100) as 'Player Names';

Result:



**#Function (8) : team\_winning\_rate\_opinion()**

#Description: Accepts teamname as input and returns the team winning rate opinion based on the below logic

# 'Good' ---> If matches won is greater than matches lost

# 'Average' --> If matches won is equal to matches lost

# 'Poor' ---> If matches won is less than matches lost

#Joins used: None.

DELIMITER $$

*CREATE DEFINER=`root`@`localhost` FUNCTION `team\_winning\_rate\_opinion`(teamname text) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE team\_winning\_rate INT;*

*DECLARE team\_winning\_rate\_opinion text;*

*select (t.Matches\_Won - t.Matches\_Lost) INTO team\_winning\_rate*

*from cricket\_socialmediaanalysis.team t where t.Team\_Name like teamname;*

*IF team\_winning\_rate > 0 THEN*

*SET team\_winning\_rate\_opinion = 'Good';*

*ELSEIF team\_winning\_rate < 0 THEN*

*SET team\_winning\_rate\_opinion = 'Poor';*

*ELSE*

*SET team\_winning\_rate\_opinion = 'Average';*

*END IF;*

*RETURN team\_winning\_rate\_opinion;*

*END$$*

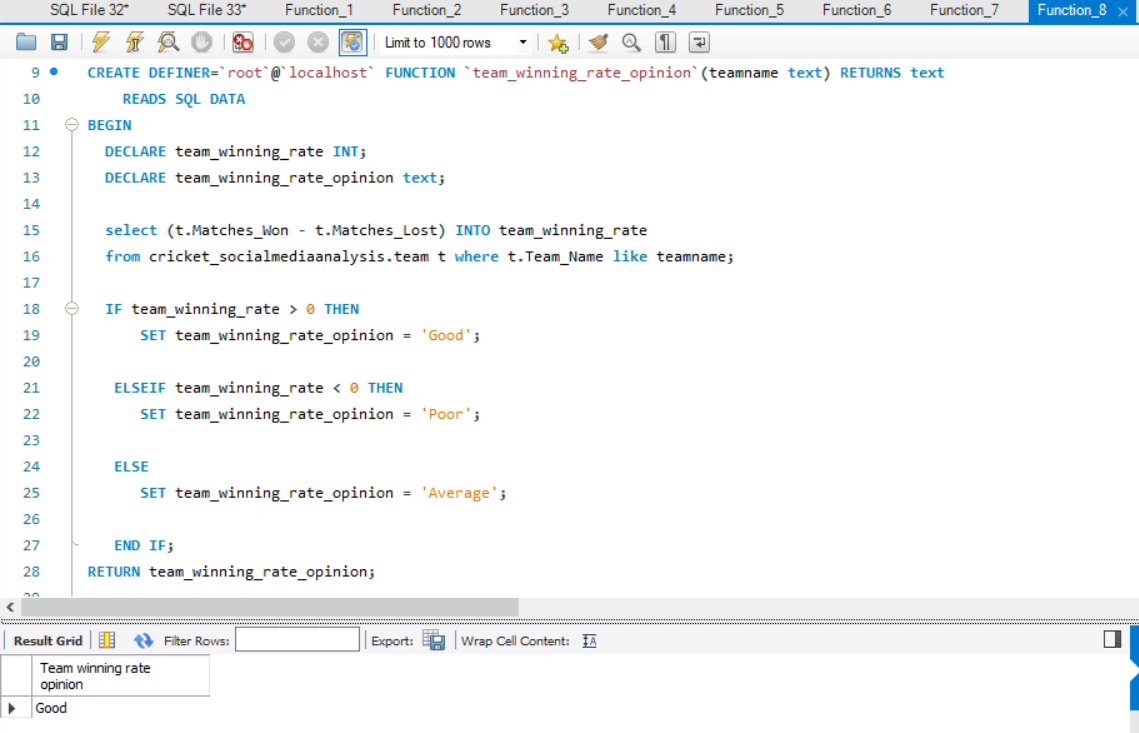
*DELIMITER ;*

# Function call for players\_with\_four\_wicket\_records\_in\_selective\_innings() with '5'

# four-wicket records and '100' innings as input

select cricket\_socialmediaanalysis.team\_winning\_rate\_opinion('New Zealand') as 'Team winning rate opinion';

Result:



**#Function (9) : allround\_players\_from\_team()**

#Description: Accepts 'teamname' as input parameter

#and returns the names of players belonging to that team, who have a batting average(computed through formula) more than 30 and

#bowling economy rate(computed through formula) less than equal to 5.0 and not equal to 0

#Joins used:

#1.Tables 'player' and 'playerbattingstatistics' are joined using 'Player\_ID'.

#2.Tables 'player' and 'playerbowlingstatistics' are joined using 'Player\_ID'.

#3.Tables 'player' and 'team' are joined using 'Team\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` FUNCTION `allround\_players\_from\_team`(teamname text) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE player\_names text;*

*select group\_concat( distinct (concat\_ws (' ', p.First\_Name, p.Last\_Name))) INTO player\_names*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.team t,*

*cricket\_socialmediaanalysis.playerbattingstatistics pb, cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*where p.Team\_ID = t.Team\_ID and p.Player\_ID = pb.Player\_ID and p.Player\_ID = pbow.Player\_ID and*

*pb.Runs\_Scored/(pb.Innings\_Batted - pb.Not\_Outs)> 30 and pbow.Runs\_Conceded/(pbow.Balls\_Bowled / 6) <= 5.0 and*

*pbow.Runs\_Conceded/(pbow.Balls\_Bowled / 6) != 0 and t.Team\_Name like teamname;*

*RETURN player\_names;*

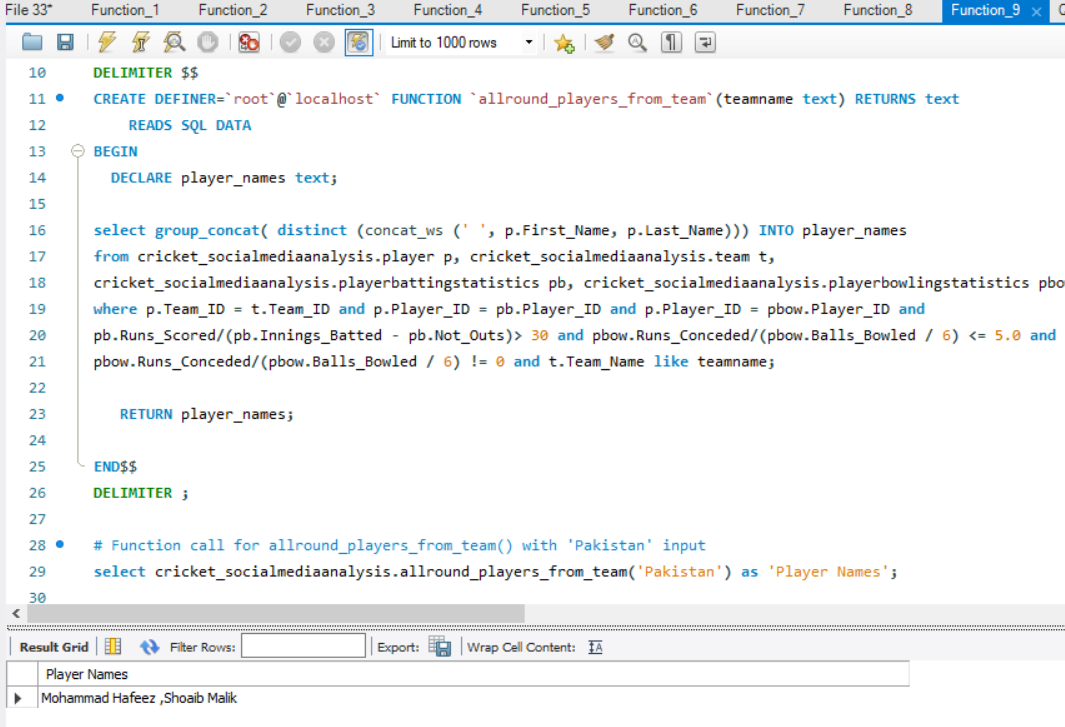
*END$$*

*DELIMITER ;*

# Function call for allround\_players\_from\_team() with 'Pakistan' input

select cricket\_socialmediaanalysis.allround\_players\_from\_team('Pakistan') as 'Player Names';

Result:



**#Function (10) : predict\_player\_type()**

#Description: Accepts firstname and lastname of the player as input and returns the player type based on the below logic

# 'Allrounder' ---> If batting average(computed by a formmula) is greater than 30 and bowling economy rate(computed by a formula)

# is less than or equal to 5.

# 'Bowler' --> If batting average(computed by a formmula) is less than 20 and bowling economy rate(computed by a formula)

# is lesser than or equal to 6 or not equal to 0

#'Batsman' --> Failure of above two cases

#Tables used:

#1.Tables 'player' and 'playerbattingstatistics' are joined using 'Player\_ID'.

#2.Tables 'player' and 'playerbowlingstatistics' are joined using 'Player\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` FUNCTION `predict\_player\_type`(firstname varchar(50), lastname varchar(50)) RETURNS text*

*READS SQL DATA*

*BEGIN*

*DECLARE batting\_average double;*

*DECLARE bowling\_economy\_rate double;*

*DECLARE player\_type text;*

*select (pb.Runs\_Scored/(pb.Innings\_Batted - pb.Not\_Outs)) , (pbow.Runs\_Conceded/(pbow.Balls\_Bowled / 6))*

*INTO batting\_average, bowling\_economy\_rate*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerbattingstatistics pb,*

*cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*where p.Player\_ID = pb.Player\_ID and p.Player\_ID = pbow.Player\_ID*

*and p.Last\_Name = lastname and p.First\_Name = firstname;*

*# select (pbow.Runs\_Conceded/(pbow.Balls\_Bowled / 6)) INTO bowling\_economy\_rate*

*#from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*#where p.Player\_ID = pbow.Player\_ID and p.Last\_Name like lastname and p.First\_Name like firstname;*

*IF batting\_average > 30.0 and bowling\_economy\_rate <= 5.0 THEN*

*SET player\_type = 'Allrounder';*

*ELSEIF batting\_average < 20.0 and (bowling\_economy\_rate <=6.0 or bowling\_economy\_rate != 0.0) THEN*

*SET player\_type = 'Bowler';*

*ELSEIF batting\_average > 20.0 THEN*

*SET player\_type = 'Batsman' ;*

*ELSE*

*SET player\_type = 'Invalid Input';*

*END IF;*

*RETURN player\_type;*

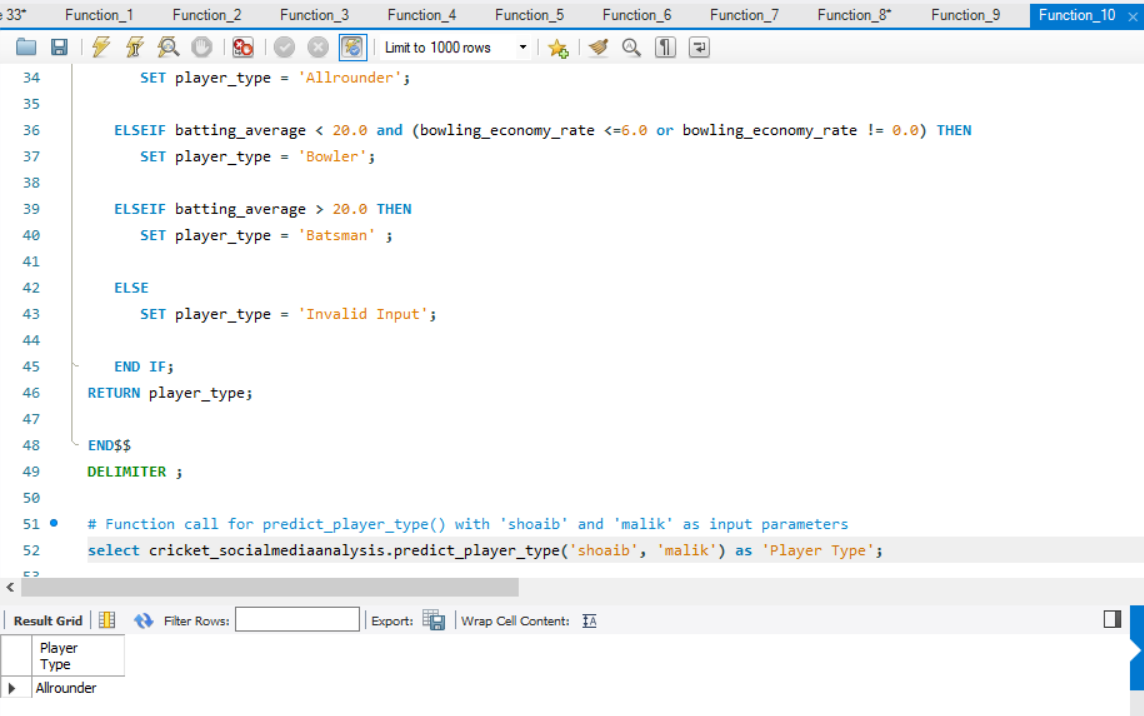
*END$$*

*DELIMITER ;*

# Function call for predict\_player\_type() with 'm' and 'morkel' as input parameters

select cricket\_socialmediaanalysis.predict\_player\_type('m', 'morkel') as 'Player Type';

Result:



**STORED PROCEDURES**

**#stored Procedure (1) : player\_specific\_tweets\_made\_on\_selected\_date()**

#Description: Consists of tweetdate as input parameter and returns the list of all tweets made on the

#input date, which is categorised with player name.

#Joins used: Tables 'player' and 'tweet' are joined using 'Player\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `player\_specific\_tweets\_made\_on\_selected\_date`(IN tweetdate date)*

*READS SQL DATA*

*BEGIN*

*SELECT (concat\_ws (' ', p.First\_Name, p.Last\_Name)) as 'Player Name', t.Content*

*FROM cricket\_socialmediaanalysis.tweet t, cricket\_socialmediaanalysis.player p*

*WHERE p.Player\_ID = t.Player\_ID*

*AND t.Created\_Time between (tweetdate) and timestamp (tweetdate, '23:59:59');*

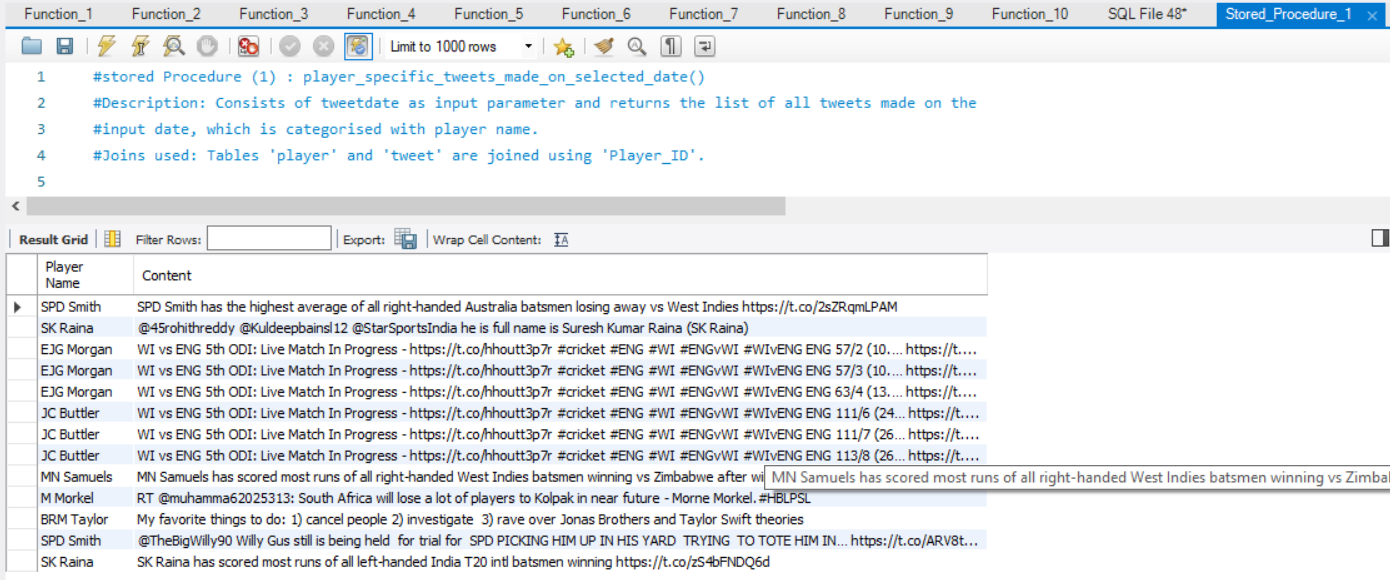
*END$$*

*DELIMITER ;*

# Procedure call for player\_specific\_tweets\_made\_on\_selected\_date() with '2019-03-02' as input parameter

Call player\_specific\_tweets\_made\_on\_selected\_date ('2019-03-02');

Result:



**#stored Procedure (2) : player\_specific\_reddit\_posts\_made\_on\_selected\_date()**

#Description: Consists of postdate as input parameter and returns the list of all reddit posts made on the

#input date, which is categorised with player name.

#Joins used: Tables 'player' and 'redditpost' are joined using 'Player\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `player\_specific\_reddit\_posts\_made\_on\_selected\_date`(IN postdate date)*

*READS SQL DATA*

*BEGIN*

*SELECT (concat\_ws (' ', p.First\_Name, p.Last\_Name)), post.Content\_Title*

*FROM cricket\_socialmediaanalysis.redditpost post, cricket\_socialmediaanalysis.player p*

*WHERE p.Player\_ID = post.Player\_ID*

*AND post.Created\_Time between (postdate) and timestamp (postdate, '23:59:59');*

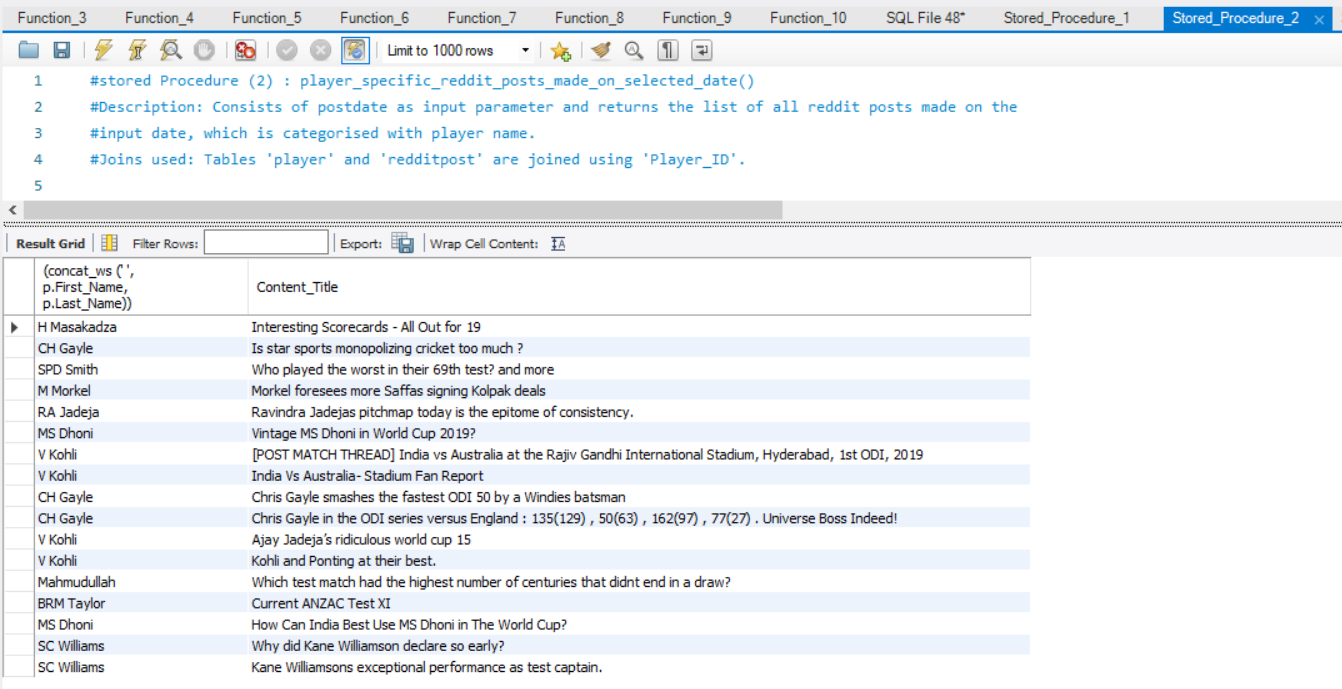
*END$$*

*DELIMITER ;*

# Procedure call for player\_specific\_reddit\_posts\_made\_on\_selected\_date() with '2019-03-02' as input parameter

Call player\_specific\_reddit\_posts\_made\_on\_selected\_date ('2019-03-02');

Result:



**#stored Procedure (3) : users\_with\_most\_retweeted\_tweets()**

#Description: Consists of number\_of\_users as input parameter and returns the list of twitters users(userID, username and user secreen name),

#where number of results is controlled by the input paramater

#Joins used: Tables 'tweet' and 'twitteruser' are joined using 'User\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `users\_with\_most\_retweeted\_tweets`(IN number\_of\_users INT)*

*READS SQL DATA*

*BEGIN*

*SELECT sum(t.Retweet\_Count) as 'Retweet Count', u.User\_ID, u.User\_Name,u.User\_Screen\_Name*

*from cricket\_socialmediaanalysis.tweet t, cricket\_socialmediaanalysis.twitteruser u*

*where t.User\_ID = u.User\_ID*

*group by u.User\_ID*

*order by sum(t.Retweet\_Count) DESC*

*LIMIT number\_of\_users;*

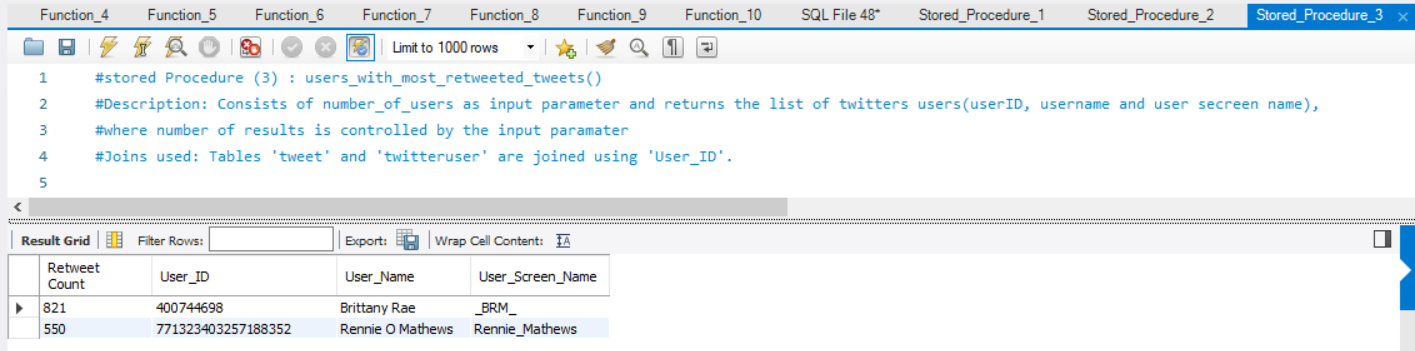
*END$$*

*DELIMITER ;*

# Procedure call for users\_with\_most\_retweeted\_tweets() with '2' as input parameter

Call users\_with\_most\_retweeted\_tweets (2);

Result:



**#stored Procedure (4) : users\_with\_most\_commented\_reddit\_posts()**

#Description: Consists of number\_of\_users as input parameter and returns the list of reddit users(userID),

#where number of results is controlled by the input paramater

#Joins used: Tables 'redditpost' and 'reddituser' are joined using 'User\_ID'.

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `users\_with\_most\_commented\_reddit\_posts`(IN number\_of\_users INT)*

*READS SQL DATA*

*BEGIN*

*select sum(p.Comments\_Count) as 'Total number of comments', u.User\_ID*

*from cricket\_socialmediaanalysis.redditpost p, cricket\_socialmediaanalysis.reddituser u*

*where p.User\_ID = u.User\_ID*

*group by u.User\_ID*

*order by sum(p.Comments\_Count) DESC*

*Limit number\_of\_users;*

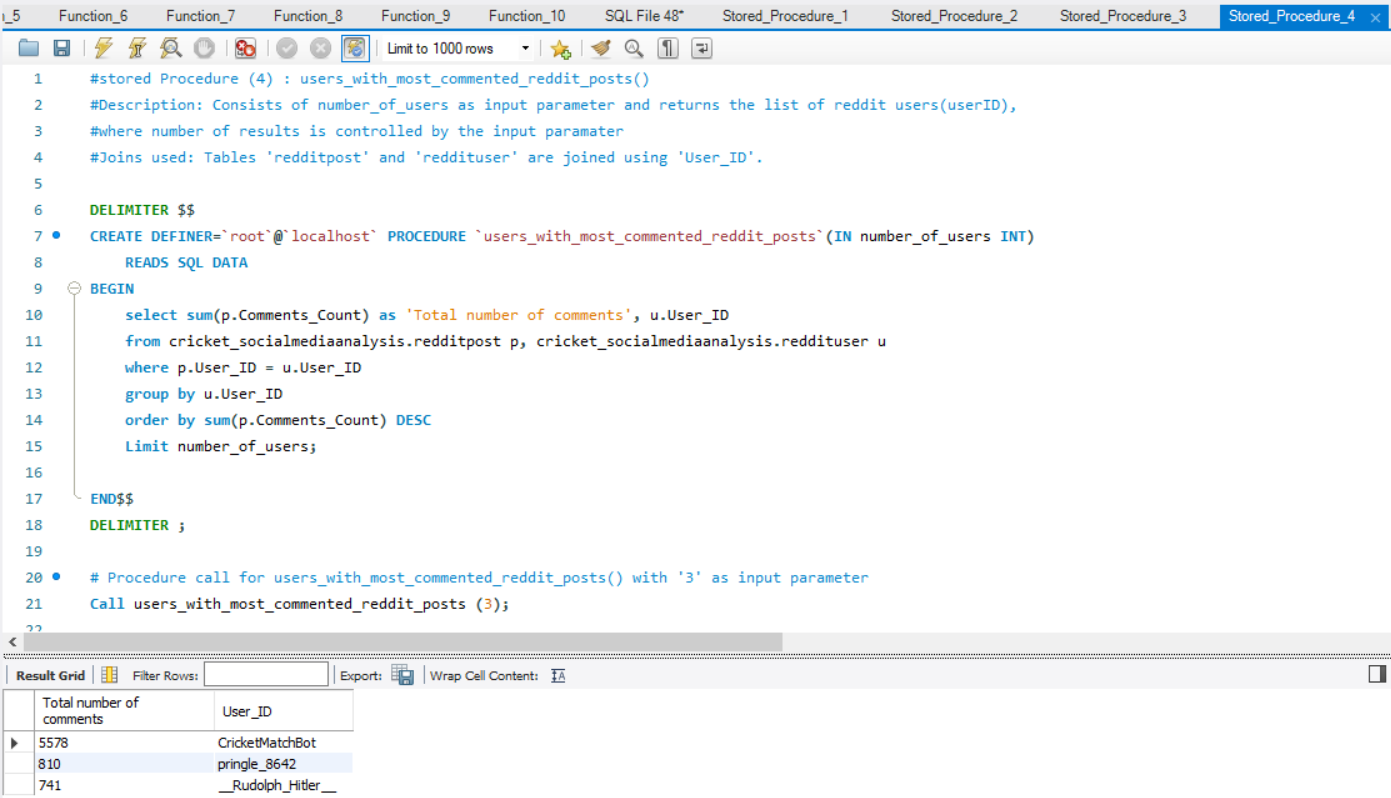
*END$$*

*DELIMITER ;*

# Procedure call for users\_with\_most\_commented\_reddit\_posts() with '3' as input parameter

Call users\_with\_most\_commented\_reddit\_posts (3);

Result:



**#stored Procedure (5) : team\_specific\_tweet\_count()**

#Description: Consists of teamname as input parameter and numberoftweets as output parameter.

#where number of tweets posted against the input team is calculated and passed to the output parameter.

#Joins used:

#1. Tables 'tweet' and 'player' are joined using 'Player\_ID'.

#2. Tables 'team' and 'player' are joined using 'Team\_ID'

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `team\_specific\_tweet\_count`(IN teamname text , OUT numberoftweets INT)*

*READS SQL DATA*

*BEGIN*

*select count(t.Tweet\_ID) into numberoftweets*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.team team, cricket\_socialmediaanalysis.tweet t*

*where t.Player\_ID = p.Player\_ID and p.Team\_ID = team.Team\_ID and team.Team\_Name like teamname;*

*END$$*

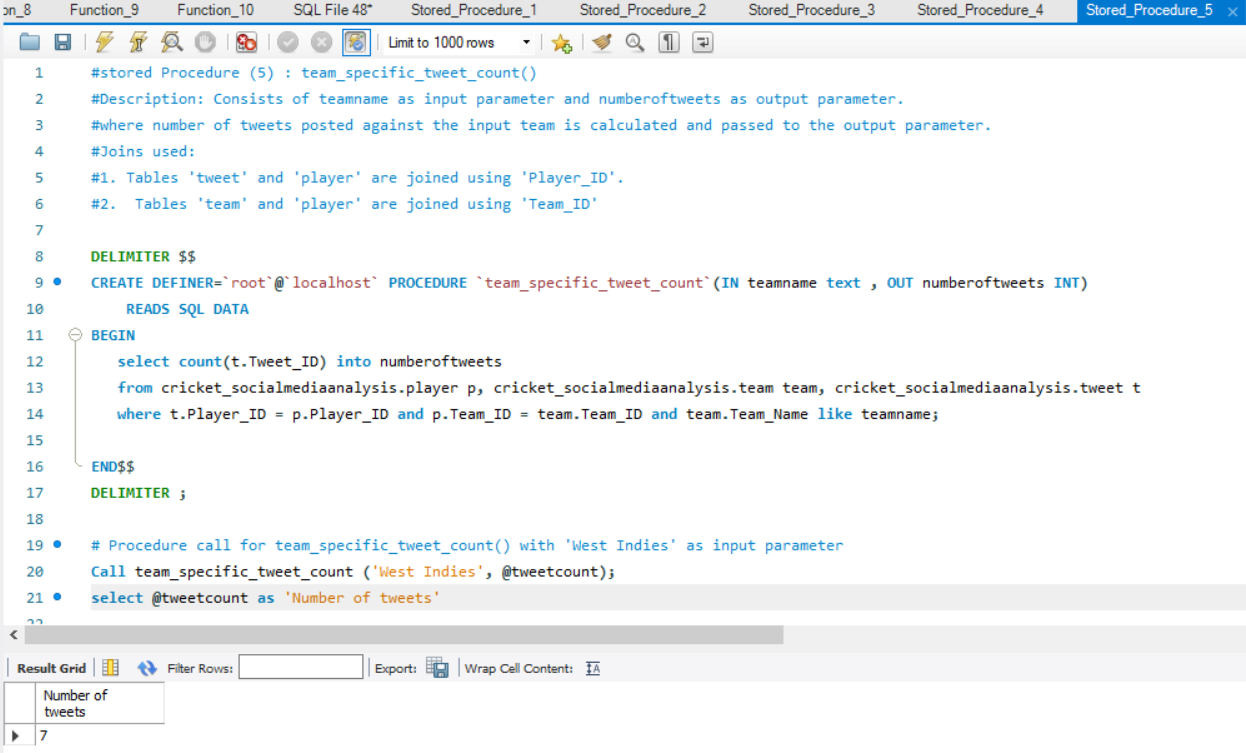
*DELIMITER ;*

# Procedure call for team\_specific\_tweet\_count() with 'West Indies' as input parameter

Call team\_specific\_tweet\_count ('West Indies', @tweetcount);

select @tweetcount as 'Number of tweets'

Result:



**#stored Procedure (6) : player\_not\_out\_percentage()**

#Description: Consists of teamname as input parameter and numberoftweets and returns the not out precentage of players in the

#team from highest to lowest

#Joins used:

#1. Tables 'team' and 'player' are joined using 'Team\_ID'

#2. Tables 'playerbattingstatistics' and 'player' are joined using 'Player\_ID'

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `player\_not\_out\_percentage`(IN teamname text)*

*READS SQL DATA*

*BEGIN*

*select concat\_ws (' ', p.First\_Name, p.Last\_Name) as 'Player Name', concat ((pb.Not\_Outs/pb.Innings\_Batted)\*100, '%') as 'Not Out Percentage'*

*from cricket\_socialmediaanalysis.playerbattingstatistics pb, cricket\_socialmediaanalysis.player p,*

*cricket\_socialmediaanalysis.team t*

*where pb.Player\_ID = p.Player\_ID and p.Team\_ID = t.Team\_ID and t.Team\_Name like teamname*

*order by ((pb.Not\_Outs/pb.Innings\_Batted)\*100) DESC;*

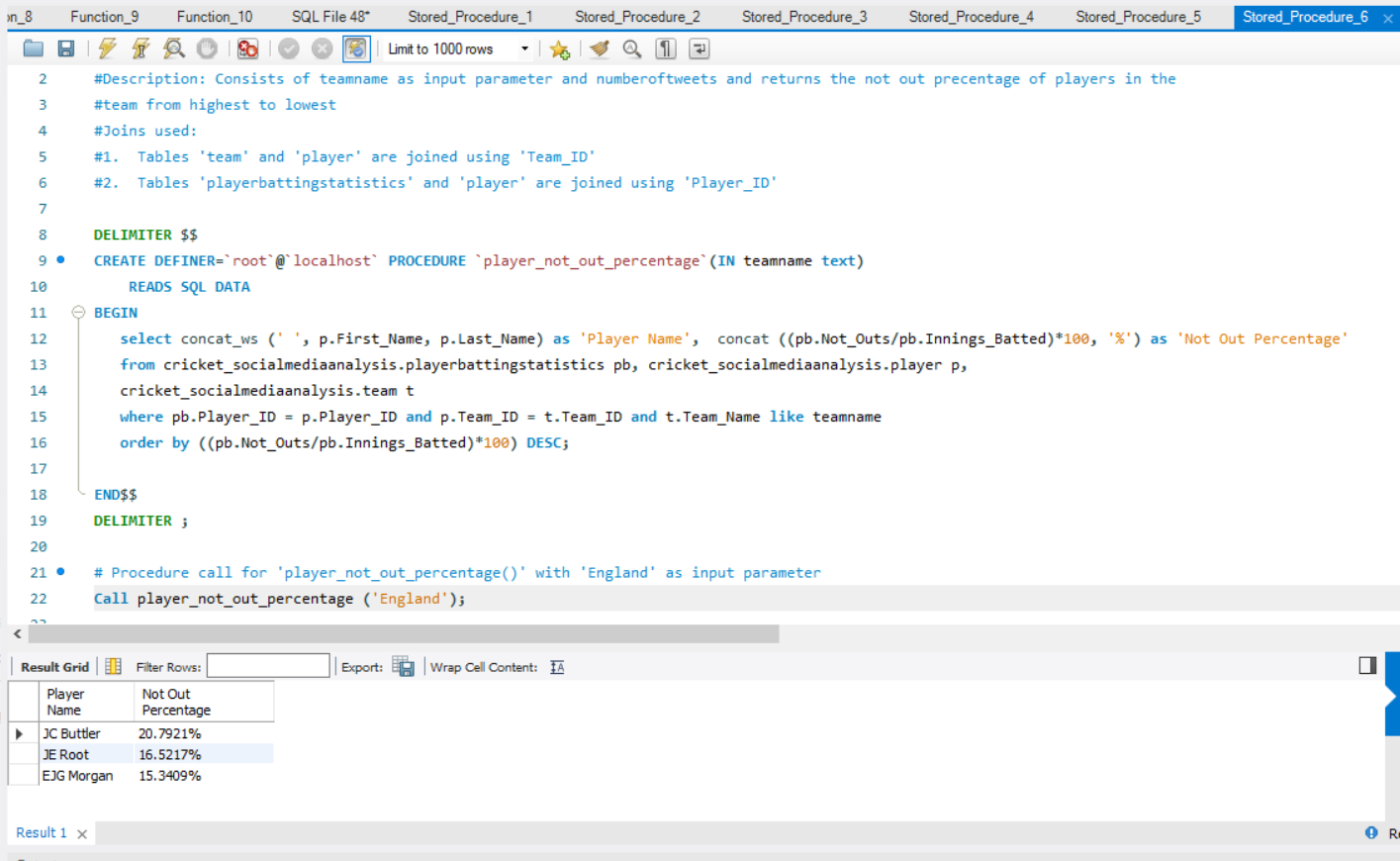
*END$$*

*DELIMITER ;*

# Procedure call for 'player\_not\_out\_percentage()' with 'England' as input parameter

Call player\_not\_out\_percentage ('England');

Result:



**#stored Procedure (7) : players\_with\_more\_runs\_than\_selected\_player()**

#Description: Consists of playername as input parameter and returns the list of all players from all teams

#with runs more than the runs scored by the player mentioned in the input parameter

#Joins used:

#1. Tables 'team' and 'player' are joined using 'Team\_ID'

#2. Tables 'playerbattingstatistics' and 'player' are joined using 'Player\_ID'

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `players\_with\_more\_runs\_than\_selected\_player`(IN playername text)*

*READS SQL DATA*

*BEGIN*

*select p.First\_Name, p.Last\_Name,batst.Runs\_Scored, t.Team\_Name*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

*on p.Team\_ID = t.Team\_ID*

*where batst.Runs\_Scored >*

*(select batst.Runs\_Scored*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*where p.Last\_Name = substring(playername from instr(playername, ' ') + 1)*

*and p.First\_Name = SUBSTRING\_INDEX(playername, ' ', 1));*

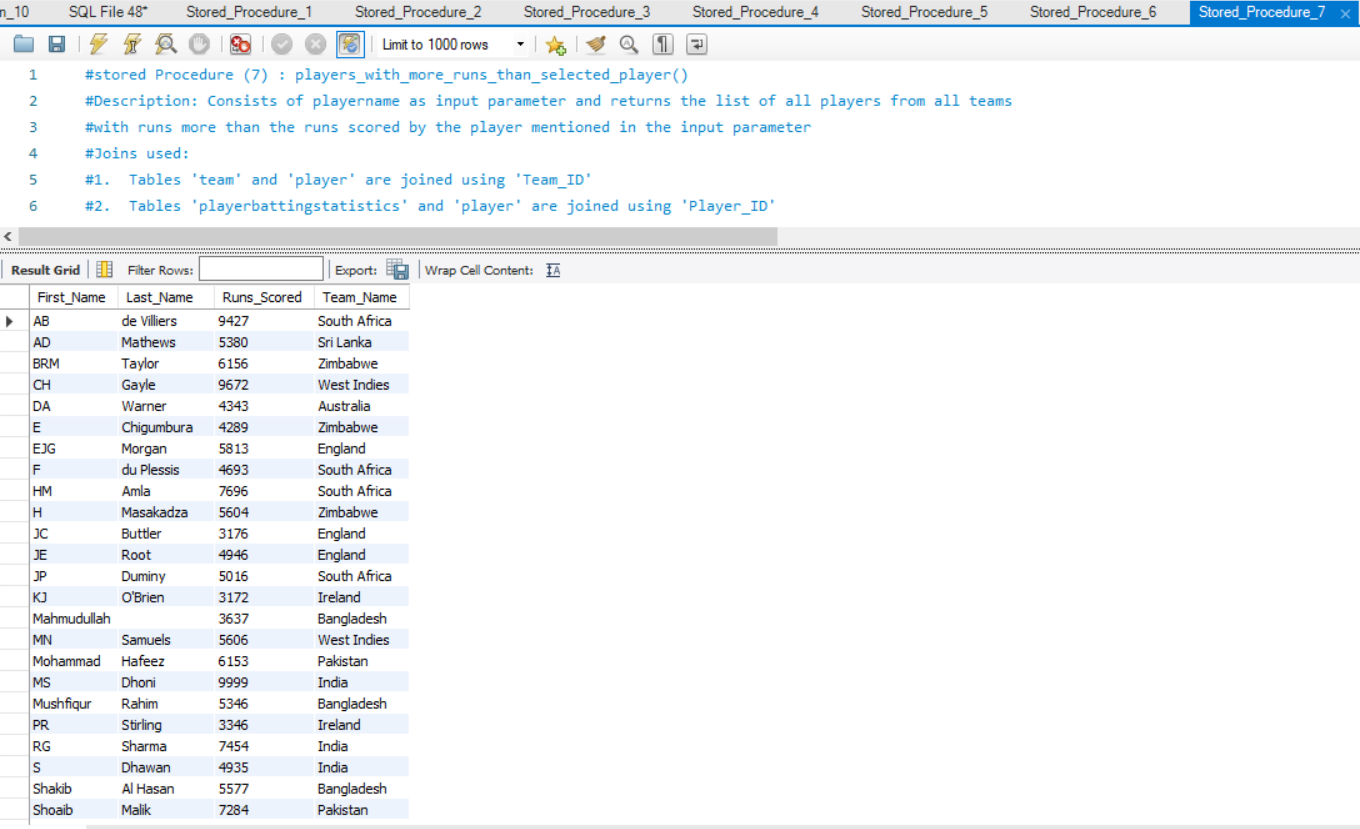
*END$$*

*DELIMITER ;*

# Procedure call for 'player\_not\_out\_percentage()' with 'DA Miller' as input parameter

Call players\_with\_more\_runs\_than\_selected\_player ('DA Miller');

Result:



**#stored Procedure (8) : players\_with\_more\_wicktets\_than\_all\_players\_of\_selected\_team()**

#Description: Consists of teamname as input parameter and returns the list of all players from all teams

#with wickets more than the wickets taken by the all players from team mentioned in the input parameter

#Joins used:

#1. Tables 'team' and 'player' are joined using 'Team\_ID'

#2. Tables 'playerbowlingstatistics' and 'player' are joined using 'Player\_ID'

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `players\_with\_more\_wicktets\_than\_all\_players\_of\_selected\_team`(IN teamname text)*

*READS SQL DATA*

*BEGIN*

*select p.First\_Name, p.Last\_Name,pbow.Wickets\_Taken, t.Team\_Name*

*from cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = pbow.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

*on p.Team\_ID = t.Team\_ID*

*where pbow.Wickets\_Taken > all*

*(select pbow.Wickets\_Taken*

*from cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = pbow.Player\_ID*

*inner join cricket\_socialmediaanalysis.team t*

*on p.Team\_ID = t.Team\_ID*

*where t.Team\_Name = teamname);*

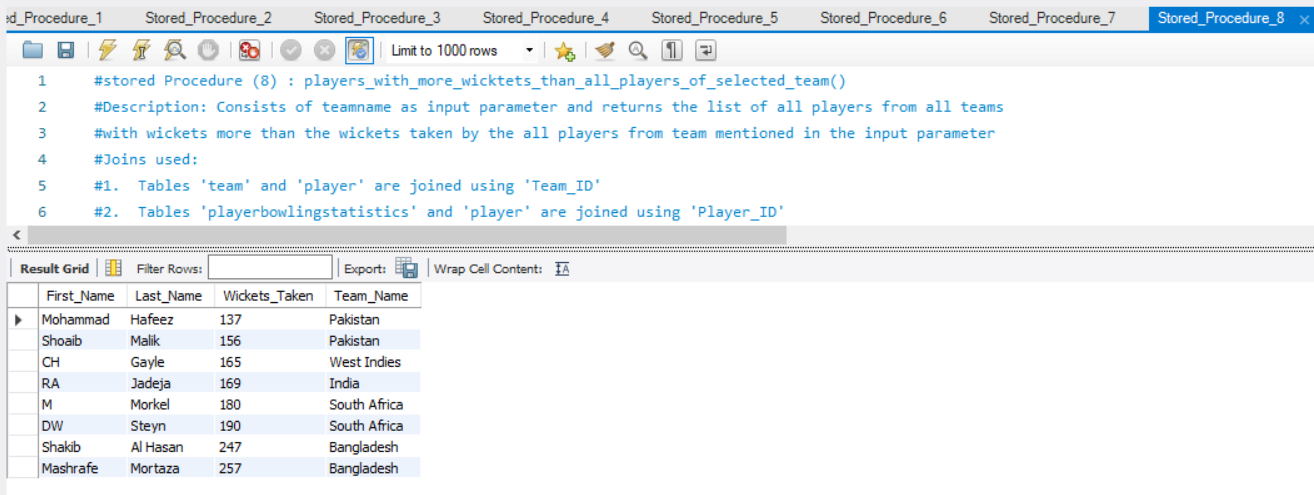
END$$

DELIMITER ;

# Procedure call for 'players\_with\_more\_wicktets\_than\_all\_players\_of\_selected\_team()' with 'Sri Lanka' as input parameter

Call players\_with\_more\_wicktets\_than\_all\_players\_of\_selected\_team ('Sri Lanka');

Result:



**#stored Procedure (9) : identify\_wicketkeppers\_from\_all\_teams()**

#Description: With no input parameter, this procedure identifies all the players who had been placed as

#'Wicketkeeprs' wothin their respective teams, based on 'Stumpings\_Made' column, as wicketkeepers are the only

#one players, who can make stumpings on field

#Joins used:

#1. Tables 'team' and 'player' are joined using 'Team\_ID'

#2. Tables 'playerfieldingstatistics' and 'player' are joined using 'Player\_ID'

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `identify\_wicketkeppers\_from\_all\_teams`()

READS SQL DATA

BEGIN

*select p.First\_Name, p.Last\_Name, t.Team\_Name*

*from cricket\_socialmediaanalysis.player p,cricket\_socialmediaanalysis.playerfieldingstatistics pfs,*

*cricket\_socialmediaanalysis.team t*

*where p.Player\_ID = pfs.Player\_ID and p.Team\_ID = t.Team\_ID and*

*pfs.Stumpings\_Made > 0;*

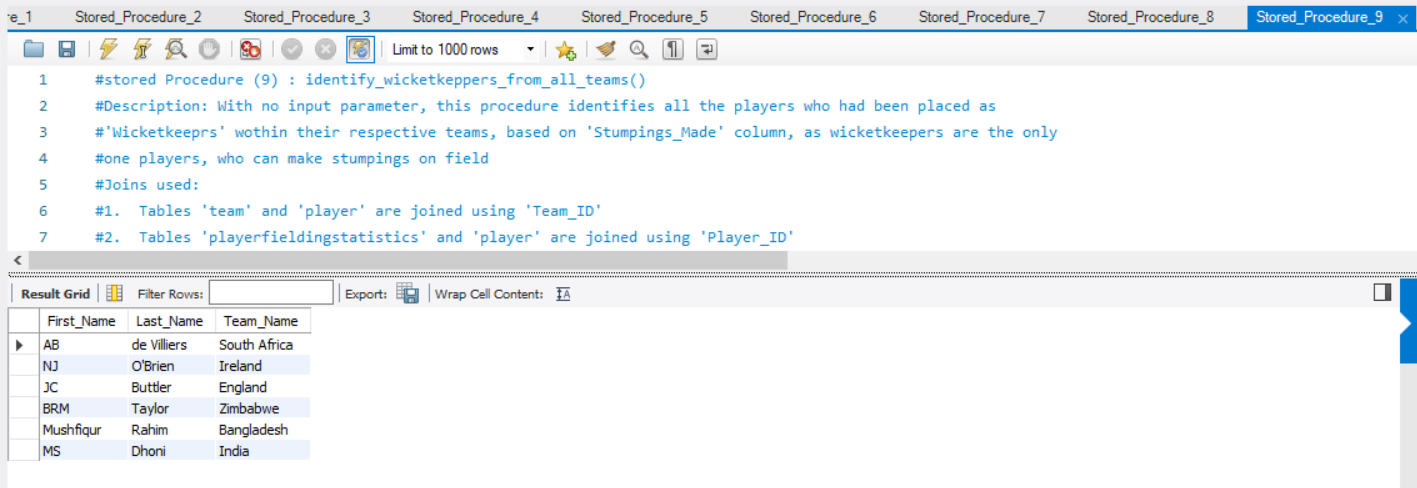
*END$$*

*DELIMITER ;*

# Procedure call for 'identify\_wicketkeppers\_from\_all\_teams()' with no input parameter

Call identify\_wicketkeppers\_from\_all\_teams ();

Result:



**#stored Procedure (10) : teams\_with\_selected\_average\_and\_winnablity\_in\_least\_matches()**

#Description: With 'battingstatslimit and 'bowlingstatslimit' as input parameters and

#'teamname' as output parameter, the procedure looks out for the team which has 'Avg\_Runs\_Per\_Wicket\_Batting' greater

#than or equal to the imput 'battingstatslimit' and 'Avg\_Runs\_Per\_Wicket\_Bowling' less than or equal to

#input 'bowlingstatslimit' in least number of matches and displays its winnability opinion predicted by logic written within

#function team\_winning\_rate\_opinion() (a function is called during procedure execution)

#Joins used:

#1. Tables 'teambattingstatistics' and 'team' are joined using 'Team\_ID'

#2. Tables 'teambowlingstatistics' and 'team' are joined using 'Team\_ID'

*DELIMITER $$*

*CREATE DEFINER=`root`@`localhost` PROCEDURE `teams\_with\_selected\_average\_and\_winnablity\_in\_least\_matches`(IN battingstatslimit float,*

*IN bowlingstatslimit float, OUT teamname text)*

*READS SQL DATA*

*BEGIN*

*select t.Team\_Name into teamname*

*from cricket\_socialmediaanalysis.team t, cricket\_socialmediaanalysis.teambattingstatistics tb,*

*cricket\_socialmediaanalysis.teambowlingstatistics tbo*

*where t.Team\_ID =tb.Team\_ID and t.Team\_ID = tbo.Team\_ID and tb.Avg\_Runs\_Per\_Wicket\_Batting >= battingstatslimit*

*and tbo.Avg\_Runs\_Per\_Wicket\_Bowling <= bowlingstatslimit*

*order by t.Team\_Matches\_Played*

*LIMIT 1;*

*END$$*

*DELIMITER ;*

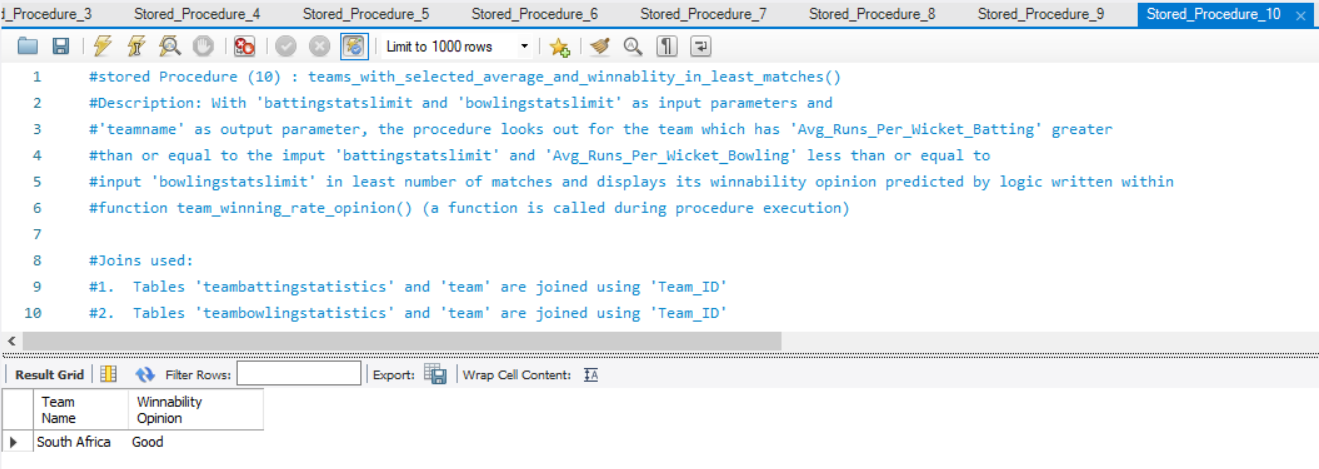
# Procedure call for 'teams\_with\_selected\_average\_and\_winnablity\_in\_least\_matches()' with 30 and 30 as input parameters with

#putput parameter 'teamname' given as input for function 'team\_winning\_rate\_opinion()'

Call cricket\_socialmediaanalysis.teams\_with\_selected\_average\_and\_winnablity\_in\_least\_matches (30,30, @teamname);

select @teamname as 'Team Name', cricket\_socialmediaanalysis.team\_winning\_rate\_opinion(@teamname) as 'Winnability Opinion' ;

Result:



**Index:**

**#Index (1) : comments\_count\_index**

#Description: Included index on 'Comments\_Count' field in redditpost table

#Displays the player names from a particular team, whose reddit posts in total had obtained more

#50 comments.

#Joins used:

#1. Tables player and redditpost are joined using Player\_ID.

#1. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select sum(rp.Comments\_Count) as 'Total Comments', p.First\_Name, p.Last\_Name*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.redditpost rp,*

*cricket\_socialmediaanalysis.team t*

*where p.Team\_ID = t.Team\_ID and rp.Player\_ID = p.Player\_ID and t.Team\_Name = 'India'*

*group by p.Player\_ID*

*having sum(rp.Comments\_Count) >=50;*

#Explain query

*explain select sum(rp.Comments\_Count) as 'Total Comments', p.First\_Name, p.Last\_Name*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.redditpost rp,*

*cricket\_socialmediaanalysis.team t*

*where p.Team\_ID = t.Team\_ID and rp.Player\_ID = p.Player\_ID and t.Team\_Name = 'India'*

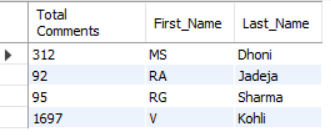
*group by p.Player\_ID*

*having sum(rp.Comments\_Count) >=50;*

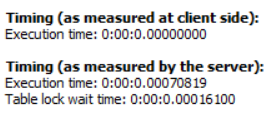
#Creation of index:

*CREATE INDEX comments\_count\_index ON cricket\_socialmediaanalysis.redditpost (Comments\_Count);*

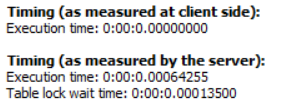
Result:



Execution time before index addition:



Execution time after index addition:



**#Index (2) : upvote\_count\_index**

#Description: Included index on 'Up\_vote\_Count' field in redditpost table

#Displays the player names from a selective teams, whose reddit posts in total had obtained more

#10 up votes.

#Joins used:

#1. Tables player and redditpost are joined using Player\_ID.

#1. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select sum(rp.Up\_vote\_Count) as 'Total Up votes', p.First\_Name, p.Last\_Name*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.redditpost rp,*

*cricket\_socialmediaanalysis.team t*

*where p.Team\_ID = t.Team\_ID and rp.Player\_ID = p.Player\_ID and t.Team\_Name in ('Pakistan', 'India', 'West Indies', 'Australia',*

*'Bangladesh')*

*group by p.Player\_ID*

*having sum(rp.Up\_vote\_Count) >= 10 ;*

#Explain query

*explain select sum(rp.Up\_vote\_Count) as 'Total Up votes', p.First\_Name, p.Last\_Name*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.redditpost rp,*

*cricket\_socialmediaanalysis.team t*

*where p.Team\_ID = t.Team\_ID and rp.Player\_ID = p.Player\_ID and t.Team\_Name in ('Pakistan', 'India', 'West Indies', 'Australia',*

*'Bangladesh')*

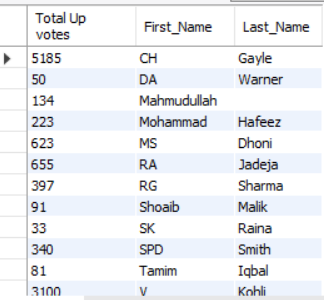
*group by p.Player\_ID*

*having sum(rp.Up\_vote\_Count) >= 10 ;*

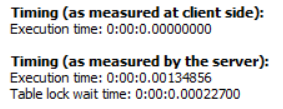
#Creation of index:

*CREATE INDEX upvote\_count\_index ON cricket\_socialmediaanalysis.redditpost (Up\_vote\_Count);*

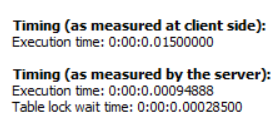
Result:



Execution time before index addition:



Execution time after index addition:



**#Index (3) : favourite\_count\_index**

#Description: Included index on 'Favourite\_Count' field in tweet table

#Displays the player names from selective teams, whose related tweets in total had obtained non zero favourite count.

#Joins used:

#1. Tables player and tweet are joined using Player\_ID.

#1. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select p.First\_Name, p.Last\_Name, team.Team\_Name,sum(t.Favourite\_Count) as 'Total Favourite Count'*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.tweet t,*

*cricket\_socialmediaanalysis.team team*

*where p.Team\_ID = team.Team\_ID and t.Player\_ID = p.Player\_ID and team.Team\_Name not in ('Pakistan', 'India', 'West Indies', 'Australia',*

*'Bangladesh')*

*group by p.Player\_ID*

*having sum(t.Favourite\_Count) != 0 ;*

#Explain query

*explain select p.First\_Name, p.Last\_Name, team.Team\_Name,sum(t.Favourite\_Count) as 'Total Favourite Count'*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.tweet t,*

*cricket\_socialmediaanalysis.team team*

*where p.Team\_ID = team.Team\_ID and t.Player\_ID = p.Player\_ID and team.Team\_Name not in ('Pakistan', 'India', 'West Indies', 'Australia',*

*'Bangladesh')*

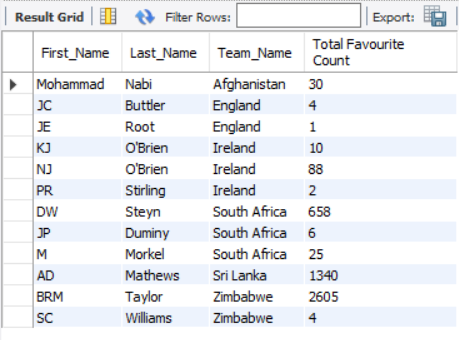
*group by p.Player\_ID*

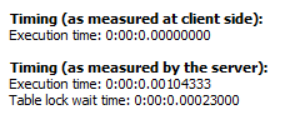
*having sum(t.Favourite\_Count) != 0 ;*

#Creation of index:

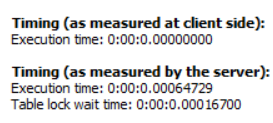
*CREATE INDEX favourite\_count\_index ON cricket\_socialmediaanalysis.tweet (Favourite\_Count);*

Result:



Execution time before index addition:  


Execution time after index addition:



**#Index (4) : retweet\_count\_index**

#Description: Included index on 'Retweet\_Count' field in tweet table

#Displays the player names from selective teams, whose related tweets in total had obtained non zero retweet count.

#Joins used:

#1. Tables player and tweet are joined using Player\_ID.

#1. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select p.First\_Name, p.Last\_Name, team.Team\_Name,sum(t.Retweet\_Count) as 'Total Retweet Count'*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.tweet t,*

*cricket\_socialmediaanalysis.team team*

*where p.Team\_ID = team.Team\_ID and t.Player\_ID = p.Player\_ID and team.Team\_Name not in ('Pakistan', 'India', 'West Indies', 'Australia',*

*'Bangladesh')*

*group by p.Player\_ID*

*having sum(t.Retweet\_Count) != 0;*

#Explain query

*explain select p.First\_Name, p.Last\_Name, team.Team\_Name,sum(t.Retweet\_Count) as 'Total Retweet Count'*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.tweet t,*

*cricket\_socialmediaanalysis.team team*

*where p.Team\_ID = team.Team\_ID and t.Player\_ID = p.Player\_ID and team.Team\_Name not in ('Pakistan', 'India', 'West Indies', 'Australia',*

*'Bangladesh')*

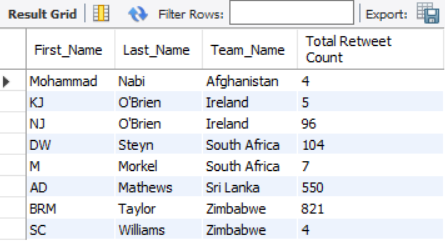
*group by p.Player\_ID*

*having sum(t.Retweet\_Count) != 0;*

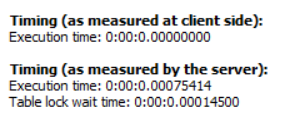
#Creation of index:

*CREATE INDEX retweet\_count\_index ON cricket\_socialmediaanalysis.tweet (Retweet\_Count);*

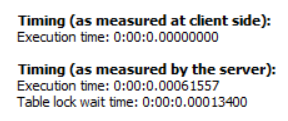
Result:



Execution before index addition:



Execution after index addition:



**#Index (5) : runs\_scored\_index**

#Description: Included index on 'Runs\_Scored' field in playerbattingstatistics table

#Displays the player names from all teams, who had scored more than all players of australia and also more than 2000.

#Joins used:

#1. Tables player and playerbattingstatistics are joined using Player\_ID.

#2. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select p.First\_Name, p.Last\_Name,batst.Runs\_Scored, t.Team\_Name*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

*on p.Team\_ID = t.Team\_ID*

*where batst.Runs\_Scored > all*

*(select batst.Runs\_Scored*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*inner join cricket\_socialmediaanalysis.team team*

*on p.Team\_ID = team.Team\_ID*

*where team.Team\_Name = 'Australia')*

*and batst.Runs\_Scored >= 2000 ;*

#Explain Query:

*explain select p.First\_Name, p.Last\_Name,batst.Runs\_Scored, t.Team\_Name*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

*on p.Team\_ID = t.Team\_ID*

*where batst.Runs\_Scored > all*

*(select batst.Runs\_Scored*

*from cricket\_socialmediaanalysis.playerbattingstatistics batst*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = batst.Player\_ID*

*inner join cricket\_socialmediaanalysis.team team*

*on p.Team\_ID = team.Team\_ID*

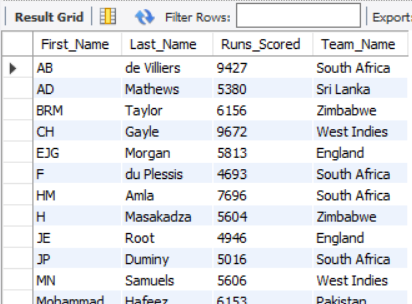
*where team.Team\_Name = 'Australia')*

*and batst.Runs\_Scored >= 2000 ;*

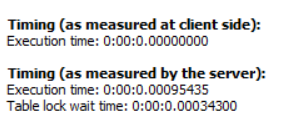
#Creation of index:

*CREATE INDEX runs\_scored\_index ON cricket\_socialmediaanalysis.playerbattingstatistics (Runs\_Scored);*

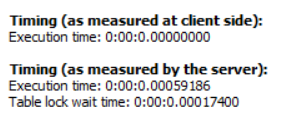
Result:



Execution time before index addition:



Execution time after index addition:



**#Index (6) : runs\_scored\_index**

#Description: Included index on 'Wickets\_Taken' field in playerbowlingstatistics table

#Displays the player names from all teams, who had taken more wicketsthan all players of australia and england

# and also more than 10.

#Joins used:

#1. Tables player and playerbowlingstatistics are joined using Player\_ID.

#2. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select p.First\_Name, p.Last\_Name,pbow.Wickets\_Taken, t.Team\_Name*

*from cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = pbow.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

*on p.Team\_ID = t.Team\_ID*

*where pbow.Wickets\_Taken > all*

*(select pbow.Wickets\_Taken*

*from cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = pbow.Player\_ID*

*inner join cricket\_socialmediaanalysis.team team*

*on p.Team\_ID = team.Team\_ID*

*where team.Team\_Name in ('Australia', 'England'))*

*and pbow.Wickets\_Taken >= 10;*

#Explain Query:

*explain select p.First\_Name, p.Last\_Name,pbow.Wickets\_Taken, t.Team\_Name*

*from cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = pbow.Player\_ID*

*inner join cricket\_socialmediaanalysis. team t*

*on p.Team\_ID = t.Team\_ID*

*where pbow.Wickets\_Taken > all*

*(select pbow.Wickets\_Taken*

*from cricket\_socialmediaanalysis.playerbowlingstatistics pbow*

*inner join cricket\_socialmediaanalysis.player p*

*on p.Player\_ID = pbow.Player\_ID*

*inner join cricket\_socialmediaanalysis.team team*

*on p.Team\_ID = team.Team\_ID*

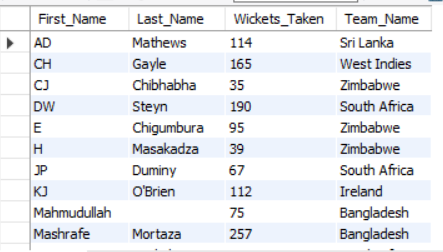
*where team.Team\_Name in ('Australia', 'England'))*

*and pbow.Wickets\_Taken >= 10;*

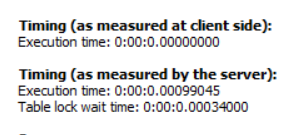
#Creation of index:

*CREATE INDEX wickets\_taken\_index ON cricket\_socialmediaanalysis.playerbowlingstatistics (Wickets\_Taken);*

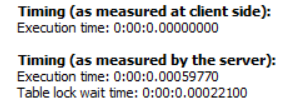
Result:



Execution time before index addition:



Execution time after index addition:



**#Index (7) : team\_matches\_played\_index**

#Description: Included index on 'Team\_Matches\_Played' field in team table

# Displays the names of all teams, which contain senior players and have played considerable number of matches.

#Joins used:

#1. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select t.Team\_Name*

*from cricket\_socialmediaanalysis.team t*

*where t.Team\_ID in*

*(select team.Team\_ID*

*from cricket\_socialmediaanalysis.team team, cricket\_socialmediaanalysis.player p*

*where team.Team\_ID = p.Team\_ID and*

*team.Team\_Matches\_Played - p.Matches\_Played <=700)*

*and t.Team\_Matches\_Played >=800;*

#Explain Query:

*explain select t.Team\_Name*

*from cricket\_socialmediaanalysis.team t*

*where t.Team\_ID in*

*(select team.Team\_ID*

*from cricket\_socialmediaanalysis.team team, cricket\_socialmediaanalysis.player p*

*where team.Team\_ID = p.Team\_ID and*

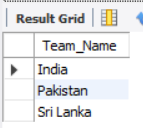
*team.Team\_Matches\_Played - p.Matches\_Played <=700)*

*and t.Team\_Matches\_Played >=800;*

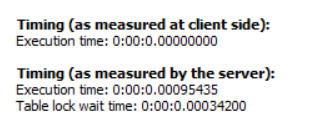
#Creation of index:

*CREATE INDEX team\_matches\_played\_index ON cricket\_socialmediaanalysis.team (Team\_Matches\_Played);*

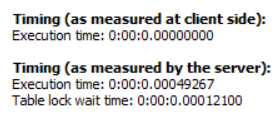
Result:



Execution time before index addition:



Execution time after index addition:



**#Index (8) : ducks\_scored\_index**

#Description: Included index on 'Ducks\_Scored' field in playerbattingstatistics table

#Displays the count of efficient batsmen within every team, who had scored very less numebr of ducks in their matches played.

#Joins used:

#1. Tables player and playerbattingstatistics are joined using Player\_ID.

#2. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select count(p.Player\_ID) as 'Count of Efficient Batsmen', t.Team\_Name*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.team t,*

*cricket\_socialmediaanalysis.playerbattingstatistics pb*

*where p.Team\_ID = t.Team\_ID and*

*p.Player\_ID = pb.Player\_ID and*

*p.player\_ID in (*

*select pbs.Player\_ID*

*from cricket\_socialmediaanalysis.playerbattingstatistics pbs*

*where pbs.Ducks\_Scored < 10)*

*and (pb.Innings\_Batted - pb.Ducks\_Scored )<=100*

*group by t.Team\_Name;*

#Explain Query:

*explain select count(p.Player\_ID) as 'Count of Efficient Batsmen', t.Team\_Name*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.team t,*

*cricket\_socialmediaanalysis.playerbattingstatistics pb*

*where p.Team\_ID = t.Team\_ID and*

*p.Player\_ID = pb.Player\_ID and*

*p.player\_ID in (*

*select pbs.Player\_ID*

*from cricket\_socialmediaanalysis.playerbattingstatistics pbs*

*where pbs.Ducks\_Scored < 10)*

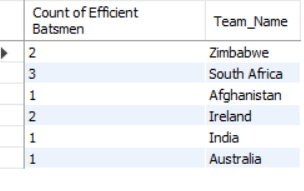
*and (pb.Innings\_Batted - pb.Ducks\_Scored )<=100*

*group by t.Team\_Name;*

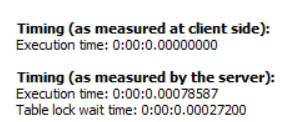
#Creation of index:

*CREATE INDEX ducks\_scored\_index ON cricket\_socialmediaanalysis.playerbattingstatistics (Ducks\_Scored);*

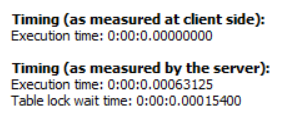
Result:



Execution time before index addition:



Execution time after index addition:



**#Index (9) : stumpings\_made\_index**

#Description: Included index on 'Stumpings\_Made' field in playerfieldingstatistics table

#Sub query picks up the list of wicketkeepers from all the teams, based on non-zero 'Stumpings\_Made' value.

#Main query picks up the wicketkeeper with more than 20 stumpings and displays the stumping per match percentage.

#Joins used:

#1. Tables player and playerfieldingstatistics are joined using Player\_ID.

#2. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select p.First\_Name,p.Last\_Name,t.Team\_Name, concat((pf.Stumpings\_Made/pf.Innings\_Fielded)\*100, '%') as 'Stumping per match Percentage'*

*from cricket\_socialmediaanalysis.playerfieldingstatistics pf, cricket\_socialmediaanalysis.player p,*

*cricket\_socialmediaanalysis.team t*

*where p.Player\_ID = pf.Player\_ID and p.Team\_ID = t.Team\_ID and pf.Stumpings\_Made > 20*

*and p.Player\_ID in (*

*select p.Player\_ID*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerfieldingstatistics pfs*

*where p.Player\_ID = pfs.Player\_ID and*

*pfs.Stumpings\_Made > 0);*

#Explain Query:

*explain select p.First\_Name,p.Last\_Name,t.Team\_Name, concat((pf.Stumpings\_Made/pf.Innings\_Fielded)\*100, '%') as 'Stumping per match Percentage'*

*from cricket\_socialmediaanalysis.playerfieldingstatistics pf, cricket\_socialmediaanalysis.player p,*

*cricket\_socialmediaanalysis.team t*

*where p.Player\_ID = pf.Player\_ID and p.Team\_ID = t.Team\_ID and pf.Stumpings\_Made > 20*

*and p.Player\_ID in (*

*select p.Player\_ID*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerfieldingstatistics pfs*

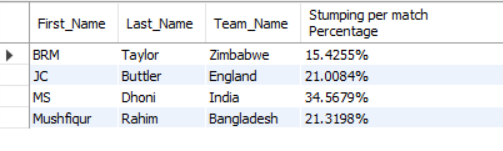
*where p.Player\_ID = pfs.Player\_ID and*

*pfs.Stumpings\_Made > 0);*

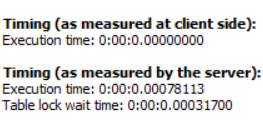
#Creation of index:

*CREATE INDEX stumpings\_made\_index ON cricket\_socialmediaanalysis.playerfieldingstatistics (Stumpings\_Made);*

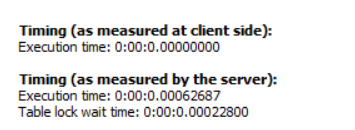
Result:



Execution time before index addition:



Execution time after index addition:



**#Index (10) : not\_outs\_index**

#Description: Included index on 'Not\_outs' field in playerbattingstatistics table

#Picks up the batsmen with a good nout out percentage, who can be considered for 'middle-order' batting position.

#First sets a limit of greater than or equal to 20 for 'Not\_outs' value and then computes the notout percentage and checks

#if the percentage is greater than or equa to 25. Pics up the batsmen, who satisfy this condition.

#Joins used:

#1. Tables player and playerbattingstatistics are joined using Player\_ID.

#2. Tables player and team are joined using Team\_ID.

#Query to check Before/After using Index:

*select concat\_ws (' ', p.First\_Name, p.Last\_Name) as 'Player Name' , t.Team\_Name, concat((pbs.Not\_outs/pbs.Innings\_Batted)\*100, '%') as 'Not-out Percentage'*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerbattingstatistics pbs,*

*cricket\_socialmediaanalysis.team t*

*where p.Player\_ID = pbs.Player\_ID*

*and pbs.Not\_Outs >=20*

*and (pbs.Not\_outs/pbs.Innings\_Batted)\*100 >=25*

*and t.Team\_ID = p.Team\_ID*

*order by p.First\_Name;*

#Explain Query:

*explain select concat\_ws (' ', p.First\_Name, p.Last\_Name) as 'Player Name' , t.Team\_Name, concat((pbs.Not\_outs/pbs.Innings\_Batted)\*100, '%') as 'Not-out Percentage'*

*from cricket\_socialmediaanalysis.player p, cricket\_socialmediaanalysis.playerbattingstatistics pbs,*

*cricket\_socialmediaanalysis.team t*

*where p.Player\_ID = pbs.Player\_ID*

*and pbs.Not\_Outs >=20*

*and (pbs.Not\_outs/pbs.Innings\_Batted)\*100 >=25*

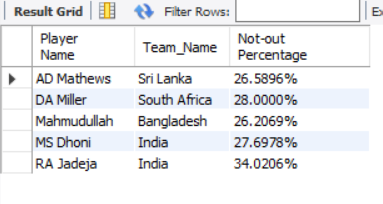
*and t.Team\_ID = p.Team\_ID*

*order by p.First\_Name;*

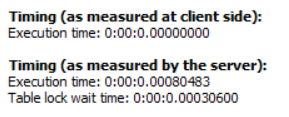
#Creation of index:

*CREATE INDEX not\_outs\_index ON cricket\_socialmediaanalysis.playerbattingstatistics (Not\_outs);*

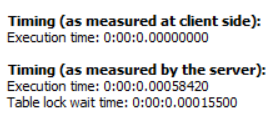
Result:



Execution time before index addition:



Execution time after index addition:



**Audit Validity-**

The whole dataset had more information than required for the proposed conceptual database model. To fit in to the database schema, the dataset was reformatted. Some of the bowling figures were in ‘-‘, which were replaced by 0. A few of the columns have been renamed for the purpose of making it simple and understandable.

**Audit Completeness**

The dataset is up to date, needs no more cleaning and matches the quality of the real world data. The data was cross verified with ESPN criconfo.

**Audit Consistence/Uniformity:**

The possible range of the dataset is covered from the new resultant data set. The data does not have any null values, limitations, negative values.

Also the values appear to remain constant with what is represented in ESPN cricinfo.

**Conclusion:**

In the assignment, all of the CSV tables were populated are populated with real-word data collected from three sources: Twitter API, Reddit API and kaggle. Normalization and optimization were effectively performed using the above mentioned methods.

Citations and References:

1. https://www.geeksforgeeks.org

2. https://github.com/nikbearbrown/INFO\_6210

3.google

4.kaggle