DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

|  |  |
| --- | --- |
| PES1201802827 | Sooryanath.I.T |

|  |
| --- |
| The IPL\_TOURNAMENT data model is represented in its highest abstract form (ERD) and then converted to a Relational Database using ER mapping technique.Implemented using PostgreSQL .This data model deals with transactions related to the teams participating, players playing under those teams,the schedule of matches ,injury record of players and the wholesome set of Sponsors associated with a team.DDL queries have been implemented to ensure all 3 integrity constraints.  Few advanced queries involving quantifiers and DQL,DML etc have been executed along with a set of correlated sub-queries and aggregate queries, concluded with the outer join queries.A constraint for trigger is identified .A trigger and a procedure call have been implemented in union to prohibit violation of constraints.The output of queries have been shown as screen snips.    The DB allows the user to schedule the matches\game**s** between participating teams at ease as there aretriggers which can warn /raise exception when a violation occurs.The team-management of various teams can obtain details of their team-player's injury record, schedule with ease , hence the designed DB supportsvarious level of views.Few future developments like inclusion of graphical visualization of order able data can be incorporated into the system. |

**[Introduction](#_gwqiqapyxr0x)** 2

**[Data Model](#_p451stoz4u81)** 3

**[FD and Normalization](#_wc0v2611q54c)** 4

**[DDL](#_pn441yprtk2f)** 8

**[Triggers](#_a9bswny9ffj)** 10

**[SQL Queries](#_exvna2br67rz)** 12

**[Conclusion](#_k6bbxckcf51j)** 16

# 

# Introduction

1. **Mini world Description**

The mini world requirements for a simple database to be created for the Indian Premier League (IPL)

· The IPL has many teams participating each with **unique name**

· Each team has a name, represents a state , a head coach, a captain, and a set of players.There maybe more than one team representing the same state for a different city.Every team belongs to a Pool ‘A’ or ‘B’ and has won no.of.titles.

· Each player **belongs to only one** team.

· Each player has a name,a position(batsman,Bowler,All-rounder), **franchise\_player ID**, a skill level.

.A player may have a log of injury records describing the injury(hamstring ,tear etc)

And date of injury.

· A **player also captains a side**/team [Ignore Vice captain ]

· A game is **played between two teams** (where one team hosts at home and the other is a visitor) and has a date (such as May 11th, 1999) and a venue (stadium). A team can play only one game per day. A team may be a host or a visitor.

·Every **Team has one or more** Sponsor (Some Business Unicorns/corporate) ,but a sponsor can invest funds only in one team for certain period of years.

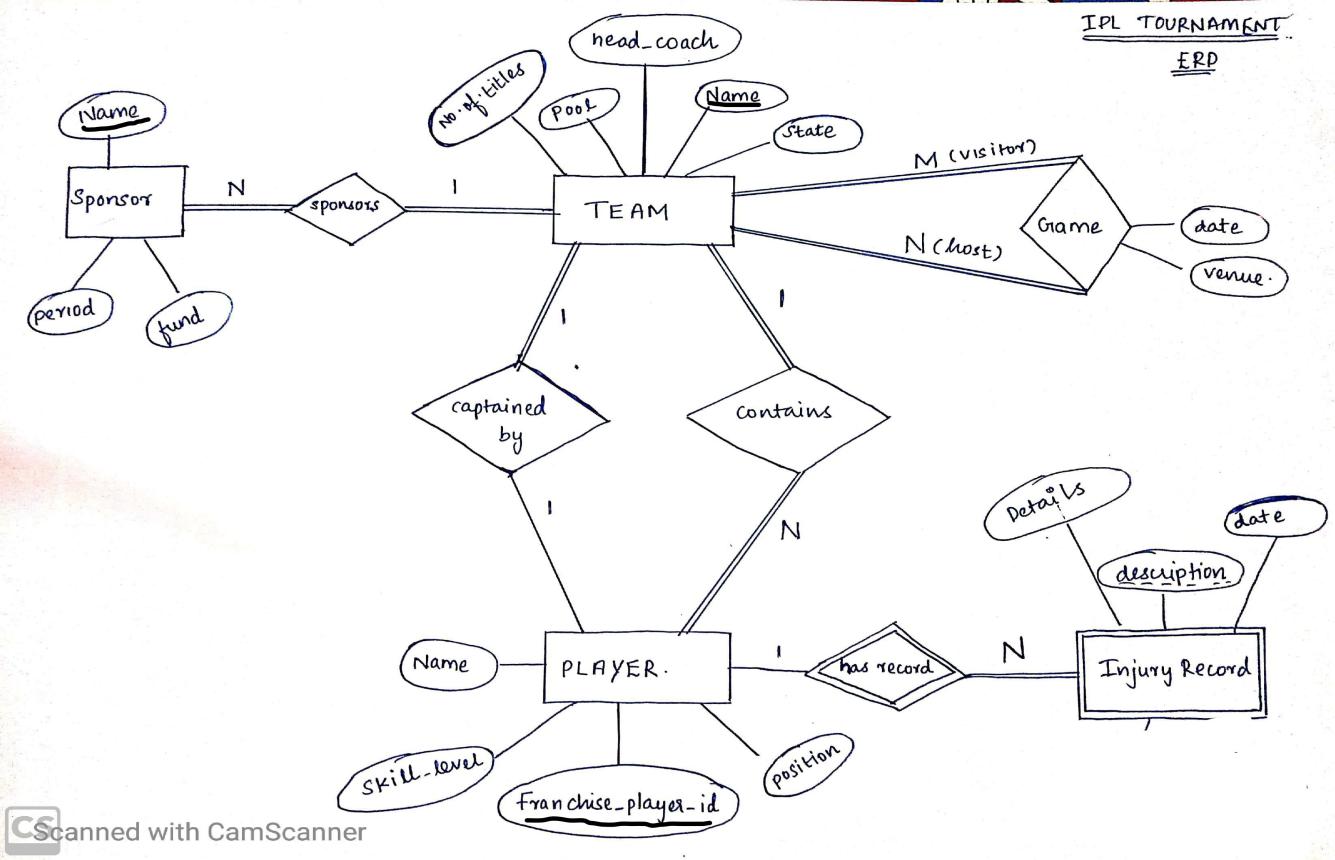
1. **Transactions of The System**

Atomicity property is ensured such that an operation is fully executed else withdrawn.A player can have an injury record only if he is a part of the players’ relation else the execution is called off.Consistency is ensured as no database operation leaves the database in an incomplete phase .All operations can be executed in parallel threads without interference hence reinforcing isolation property.Any commits made are long lasting and durable.

# 

# Data Model

ER Diagram for the specified Mini world of interest.



Relational Schema

Team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Team\_Name | State | Head\_coach | No.of.titles | Captain | Pool |

Players

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Franchise\_Player\_id | Skill | Position | Name | Team\_name |

Injury record

|  |  |  |  |
| --- | --- | --- | --- |
| Player\_id | Description | Date | details |

Sponsors

|  |  |  |  |
| --- | --- | --- | --- |
| Corporate\_name | Team\_name | Funds | Sponsor\_period |

Game

|  |  |  |  |
| --- | --- | --- | --- |
| Host | Visitor | Venue | Date |

Discussion on Choice of keys

Importance

**Primary keys enforce entity integrity by uniquely identifying entity instances. Where as foreign keys enforce referential integrity constraints.**

1. ’**Team**’ relation/table lists all the competing teams in the tournament , given in the mini world description that every team has an unique name ,we declare the Team\_nameattribute of the team as the primary key of identification,hence we may consider **‘head\_coach’ as a candidate key**. Example :Super Kings,Royal Challengers etc.No teams can share the same name.Out of available candidate keys , we choose ‘team\_name’ as primary key.

**Possible super keys** : {name,state},**primary key**=’team\_name’, **foreign\_key**=’Captain’

1. ’**Players**’ relation lists the details like expertise/position/role , skill level of a player with each player assigned with an ID,the team\_name may not be unique ( as N players may belong to same team),player name doesn’t qualify as a key (2 or more players can possess the same name).Hence we declare the **Franchise**\_**Player\_id** as the identifying key attribute.

**Possible super keys** : {Franchise\_player\_id,name}.**Primary key**=’ Franchise\_Player\_id’, **Foreign Key**=’Team’

1. ’**Injury\_record**’ relation lists the log of injuries (like hamstring ,meniscus tear) associated with a player in a season, which lets the team management to take a decision based on one’s fitness record.A combination of player\_id and description (partial key) forms the primary key.

**Foreign\_key**=’player\_id’ ,**primary composite key**={player\_id,description}

1. **’Sponsor’** relation lists the name of the fund investing companies and to which team they are sponsoring.Since a sponsor can invest fund only on one company **‘**corporate\_name’ is the primary key in accordance with the relation.

**Primary key**=’corporate\_name’ ,**foreign\_key**=’team\_name’

1. **’Game’** is the relation/table which involves the entity ‘Team’ in 2 different roles.One as the host and the other as the visiting team. Every team has the privilege of hosting the game.Since a venue can host only one game per day , we create a composite key **‘date,venue’.**

**Primary key=’{team(host),team(visitor)}**’

# FD and Normalization

A --> B is read as **A functionally determines B** , whenever two tuples have the same value for A, they *must have* the same value for B. If K is a key of a Relation S, then K functionally determines all attributes in S (because K is unique ).

* Functional dependencies of the Relations in the IPL\_Tournament are listed below

1. **Team\_Name** →{State,Captain,head\_coach,no.of.titles,Pool} in Relation ‘Team’.
2. **Franchise\_player\_id** →{Name,skill\_level,position,team\_name} is one among the functional dependencies for Relation ‘Player’.
3. **Corporate\_name** →{team\_name,fund,period} , since every corporate can sponsor only one team, this is a functional dependency in Relation ‘Sponsors’.
4. **{host,visitor}** →{venue,date} is a functional dependency in Relation ‘Game’.
5. **{Player\_id,description}** → {date,details} is a functional dependency in Relation ‘Injury\_record’.

**Normal Forms**

The relational schema is obtained after a following the steps of ERD to relational schema mapping.

**First Normal Form (1NF ):**

All relational schemas are in 1NF as the entries in tuples (attributes of an instance ) **are atomic and indivisible**, there are no group of values for an attribute.

**Second Normal Form (2NF):**

Relation ‘Game’ and ‘Injury\_record’ possess a composite primary key.

1. {host,visitor} →{venue,date}. Removal of host ,makes the functional dependency invalid , the similar case with removal of visitor from the key.Hence it is a **full functional dependency.**Hence it is in 2NF.
2. {Player\_id,description} → {details,date}.Removal of player\_id makes the functional dependency invalid.Similarly by the removal of description it becomes impossible to track down right injury record.

Hence the above 2 Relations are undoubtedly in 2ND NF.

**Example of possible Violations of 2NF:**

1. **Addition of a column ‘injury\_id**’ (which yields unique values only) in the Relation ‘Injury\_record’ and making it the partial key may violate the 2NF.

{player\_id,injury\_id} -> {description,date,details} will be the FD.Where as just

{injury\_id} ->{description,date,details} alone holds good as well.Which doesn’t make the FD a full functional dependency.

**Third Normal Form (3NF) :**

3NF is violated when **there is transitive dependency involving a non prime attribute** in R. 3NF is obeyed when there is dependency X-> Y and Y -> Z where in Y is a candidate key.

Example: ‘head\_coach’ is a candidate key in Relation ‘Team’

‘team\_name’ -> ‘head\_coach’ , ‘head\_coach’ -> ‘no.of.titles’ works fine,because ‘head\_coach’ is a candidate key.

**Example of possible Violations of 3NF:**

1. **Addition of the column ‘head\_coach’** to the relation ‘Players’,once we add the a functional dependency ‘**franchise\_player\_id’ -> ‘head\_coach**’ exists ,and ‘head\_coach’ can uniquely determine the ‘team\_name’ **but ‘head\_coach’ is not a candidate key** ,as more than one player may be mentored by a same coach.

This is of the form X->Y and Y-> Z where in **Y is a non-prime attribute** / not a candidate key.

**Normal Form of Relations:**

With above proofs and explanation , it is evident that all relations are in 3NF.Which implicitly means they qualify the rules of 1NF and 2NF.

# DDL

Below set of SQL statements are the DDL statements ,**DDL commands are used to create and modify the structure of a database , enforce relationship integrity constraints and modify db objects.**

create database IPL\_TOURNAMENT;

**CREATE TABLE** TEAM

( TEAM\_NAME VARCHAR(50) **PRIMARY KEY**,

STATE VARCHAR(50) NOT NULL,

HEAD\_COACH VARCHAR(50) UNIQUE NOT NULL,

NO\_OF\_TITLES INTEGER NOT NULL,

CAPTAIN CHAR(10) UNIQUE NOT NULL,

POOL CHAR(1)NOT NULL,

**CHECK** (NO\_OF\_TITLES >= 0 AND NO\_OF\_TITLES < 13),

**CHECK** ( POOL IN ('A','B'))

);

**ALTER TABLE** TEAM

**ADD CONSTRAINT** "CAPTAIN\_ID\_FKEY" FOREIGN KEY (CAPTAIN) **REFERENCES** PLAYER(FRANCHISE\_PLAYER\_ID);

**CREATE** TABLE PLAYER

(

FRANCHISE\_PLAYER\_ID CHAR(10) **PRIMARY KEY**,

SKILL INTEGER ,

POSITION VARCHAR(15) NOT NULL,

NAME VARCHAR(50) NOT NULL,

TEAM VARCHAR(50) ,

**CHECK** (SKILL > 0 AND SKILL < 11),

**CHECK** (POSITION IN ('BOWLER','BATSMAN','ALL-ROUNDER','WICKET-KEEPER'))

);

**ALTER** TABLE PLAYER

**ADD CONSTRAINT** "TEAM\_NAME\_FKEY" **FOREIGN KEY** (TEAM) **REFERENCES** TEAM(TEAM\_NAME) **ON DELETE CASCADE**;

**CREATE TABLE** INJURY\_RECORD

(

PLAYER\_ID CHAR(10),

DESCRIPTION VARCHAR(50) NOT NULL,

DATE\_OF\_INJURY DATE NOT NULL,

DETAILS TEXT,

**PRIMARY KEY(PLAYER\_ID,DESCRIPTION)**

);

**ALTER TABLE** INJURY\_RECORD

**ADD CONSTRAINT** "INJURED\_player\_id" FOREIGN KEY (PLAYER\_ID) **REFERENCES** PLAYER(FRANCHISE\_PLAYER\_ID);

**CREATE TABLE** SPONSOR

(

CORPORATE\_NAME VARCHAR(50) **PRIMARY KEY**,

FUNDS INTEGER NOT NULL,

TEAM VARCHAR(50) ,

TIME\_PERIOD\_YEARS INTEGER NOT NULL

);

**ALTER TABLE** SPONSOR

**ADD CONSTRAINT** 'SPONSORING\_TEAM' FOREIGN KEY (TEAM) **REFERENCES** TEAM(TEAM\_NAME);

**CREATE TABLE** GAME

(

HOST VARCHAR(50) REFERENCES TEAM(TEAM\_NAME),

VISITOR VARCHAR(50) REFERENCES TEAM(TEAM\_NAME),

VENUE VARCHAR(50) NOT NULL,

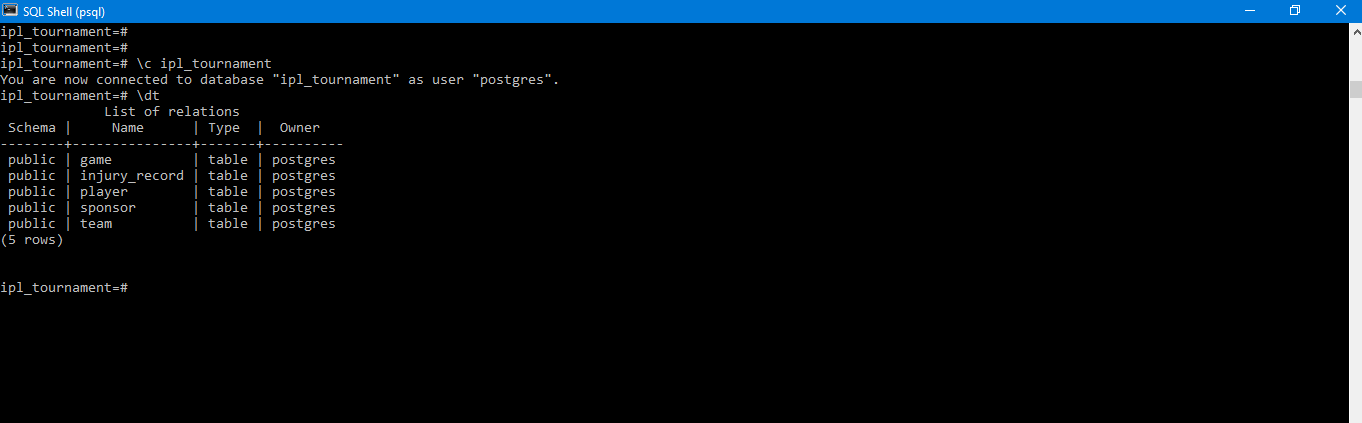
DATE\_OF\_GAME DATE NOT NULL,

**PRIMARY KEY(HOST,VISITOR)**

);

ALTER TABLE GAME ADD CONSTRAINT "HOST\_VISITOR\_CHECK" CHECK (HOST != VISITOR );

**List Of Relations created in Database ‘IPL\_TOURNAMENT’ successfully.**



# Triggers

Trigger is a function invoked automatically whenever an event e.g., insert, update, or delete occurs with respect to the table of interest.

**CONSTRAINT:** Ensure that a team doesn’t play more than one match/game on the same day.

* This constraint is to be checked **BEFORE** a record is **INSERTED**.
* The above constraint needs to be handled before INSERT and cannot be handled using a normal **CHECK** constraint.
* A procedure is written/defined to check if the teams(host ,visitor) in the current record to be inserted ,have a game to play already on the same date as in the new row/record.
* The Procedure is named as **SCHEDULE\_CHECK( ), SQL code goes below**.

**CREATE OR REPLACE FUNCTION schedule\_check(**)

RETURNS trigger AS

$BODY$

BEGIN

**IF EXISTS** (SELECT \* FROM GAME WHERE DATE\_OF\_GAME=NEW.DATE\_OF\_GAME AND (HOST=NEW.HOST OR VISITOR=NEW.VISITOR OR HOST=NEW.VISITOR OR VISITOR=NEW.HOST))

**THEN**

**RAISE EXCEPTION** 'NO TEAM CAN PLAY 2 GAMES ON THE SAME DAY : ';

**END IF**;

**RETURN NEW;**

END;

$BODY$

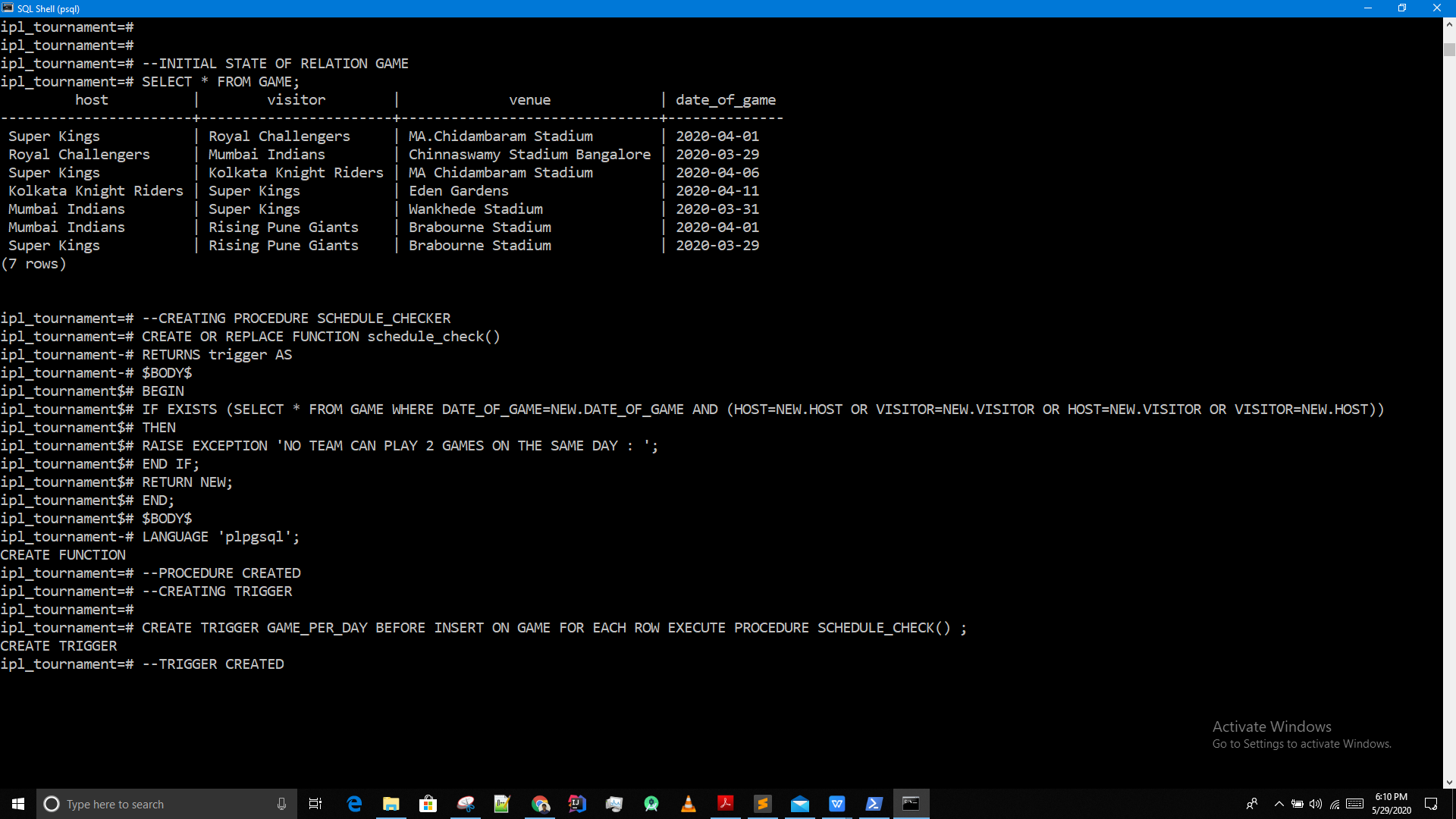
**LANGUAGE 'plpgsql'**;

* The procedure checks in the GAME relation if there exist a schedule for the teams in the insert statement, which corresponds to the same date as in the record to be inserted.
* Now a **trigger GAME\_PER\_DAY** is created on table GAME which calls the procedure SCHEDULE\_CHECK().

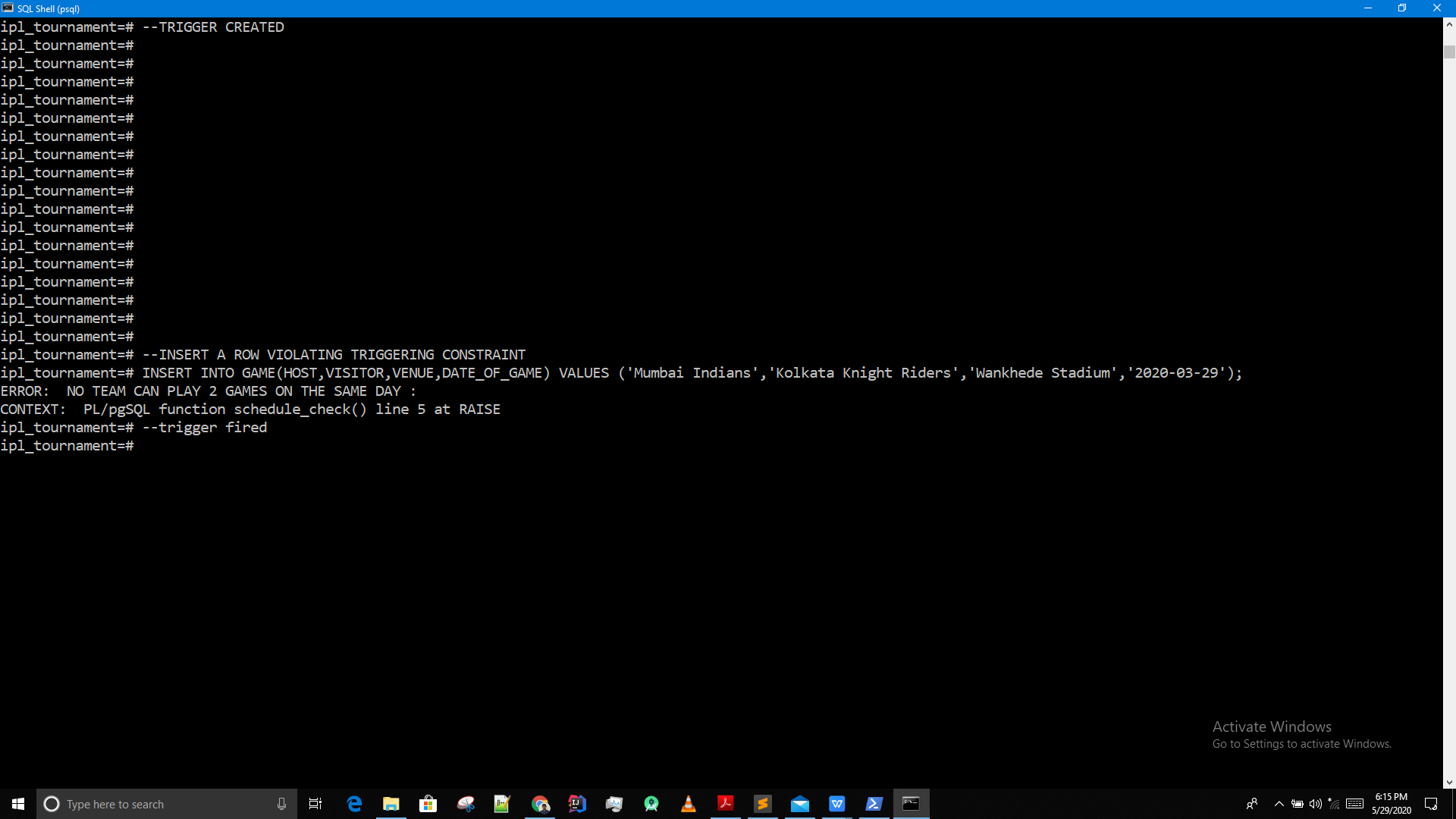
**CREATE TRIGGER** GAME\_PER\_DAY **BEFORE INSERT** ON GAME **FOR EACH ROW** **EXECUTE PROCEDURE SCHEDULE\_CHECK()** ;

* This **Trigger gets fired whenever a new record** is to be inserted into GAME relation.**Throws Exception when constraint is violated** , else record is inserted.

OUTPUT OF TRIGGER IS SHOWN BELOW.



**TRIGGER AND PROCEDURES CREATED.**



**TRIGGER RAISES EXCEPTION WHEN ROW VIOLATING CONSTRAINT IS TO BE ADDED**

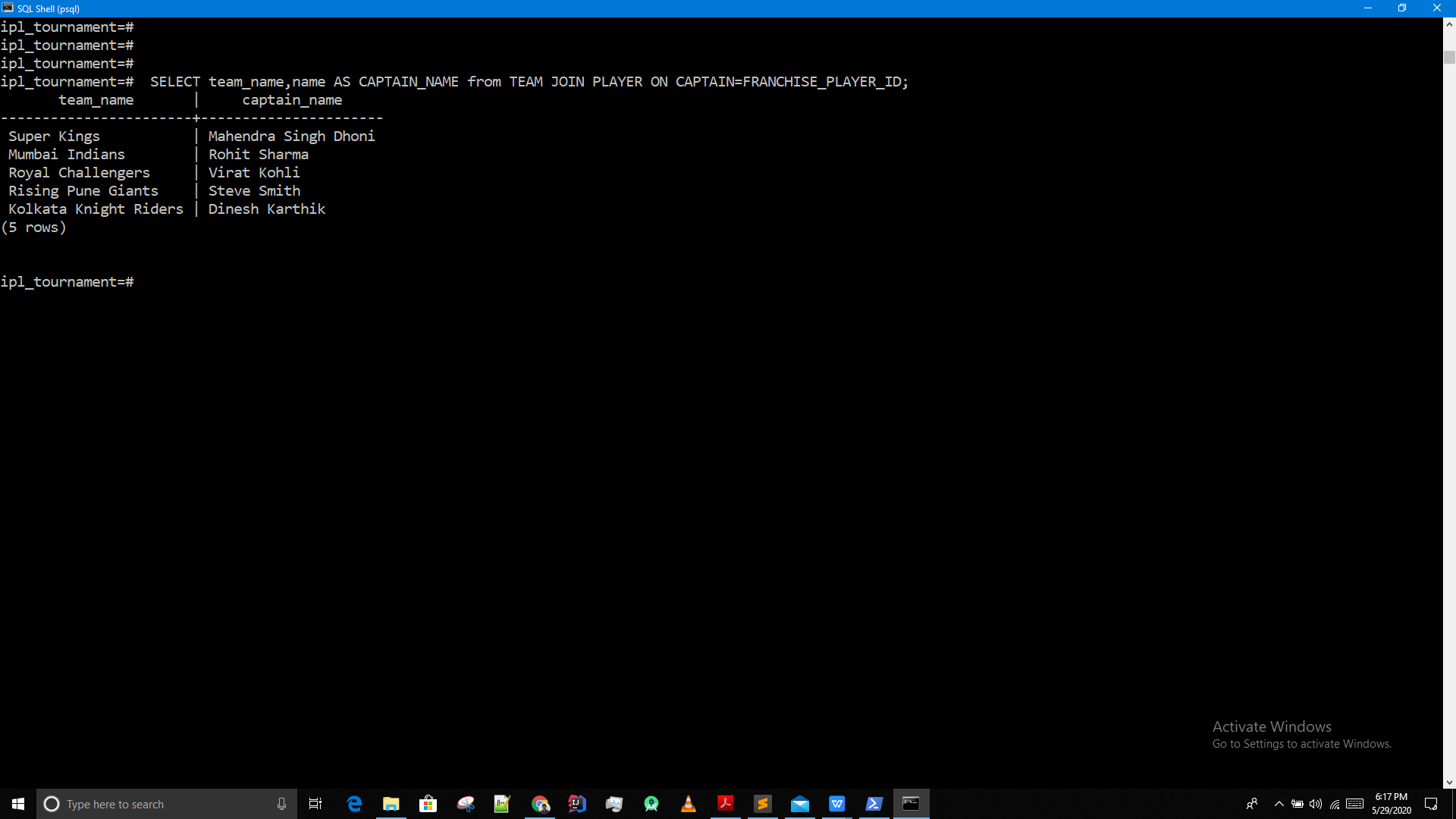
# SQL Queries

**I.JOIN ON CONDITION (INNER JOIN)** query

**English statement**: List the skipper name for every corresponding team

**Query\_1 : SELECT** team\_name,name AS CAPTAIN\_NAME **from** TEAM **JOIN** PLAYER **ON**  CAPTAIN=FRANCHISE\_PLAYER\_ID;

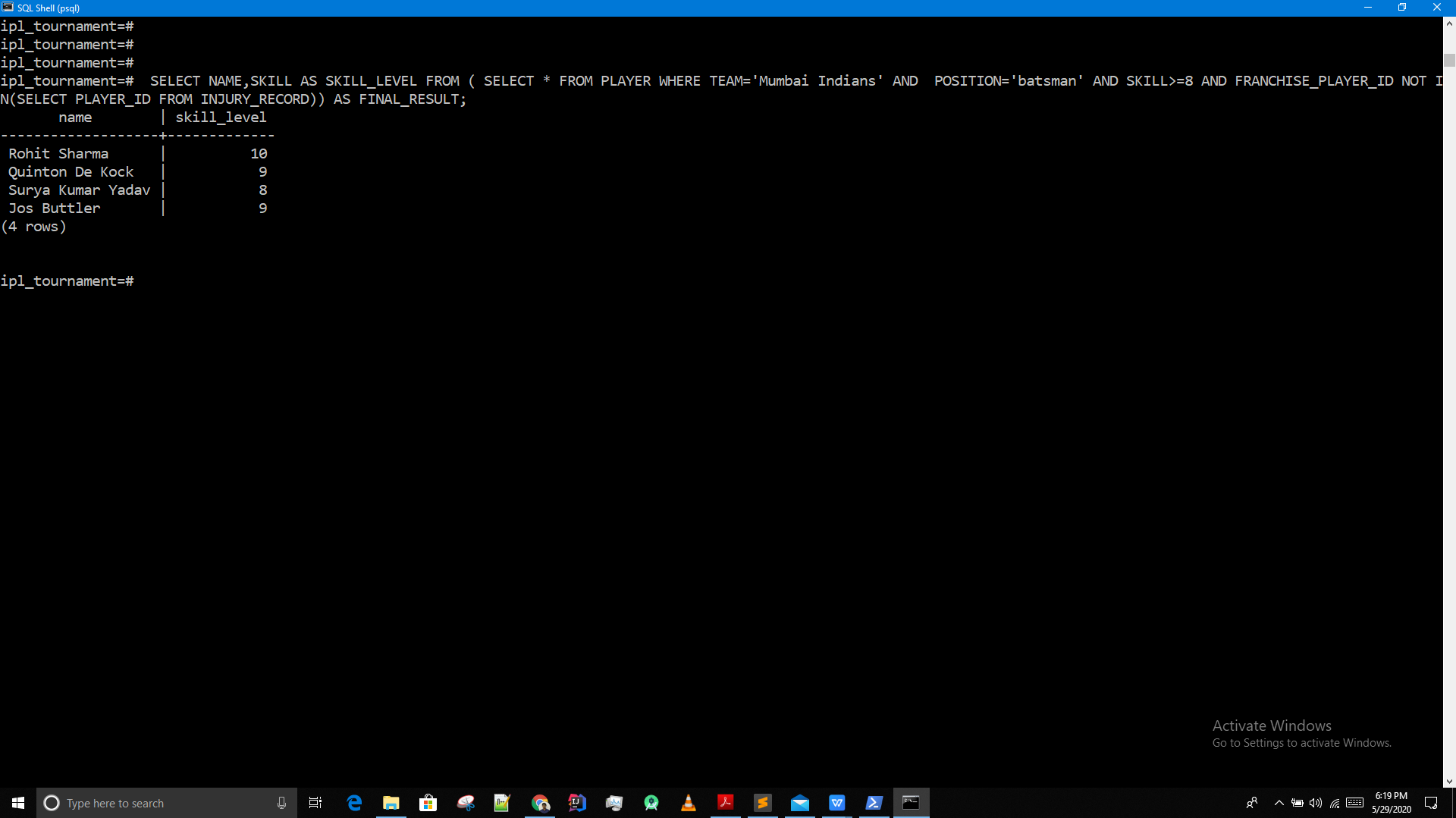
Output



**II.NESTED SUB QUERIES**

1. **English statement :** Retrieve only the name and skill level of all the batsmen with skill level >=8 ,playing for the team Mumbai Indians with no injury records in this season**.**

**Query\_2 : SELECT** NAME,SKILL AS SKILL\_LEVEL FROM ( SELECT \* FROM PLAYER WHERE TEAM='Mumbai Indians' AND POSITION='batsman' AND SKILL>=8 AND FRANCHISE\_PLAYER\_ID NOT IN(SELECT PLAYER\_ID FROM INJURY\_RECORD)) **AS FINAL\_RESULT**;



**II.English statement :** List the venue and the opponents of the team 'Super Kings' on all occasions when their team player was injured in this season.

**Query\_3** : **CREATE TABLE NEW** AS **SELECT** HOST,VISITOR,VENUE FROM GAME **WHERE DATE\_OF\_GAME IN** (SELECT DATE\_OF\_INJURY FROM PLAYER JOIN INJURY\_RECORD ON TEAM='Super Kings' AND FRANCHISE\_PLAYER\_ID=PLAYER\_ID) AND (HOST='Super Kings' OR VISITOR='Super Kings') ;

--temporary table is created on fly--

**SELECT** VENUE,

**CASE**

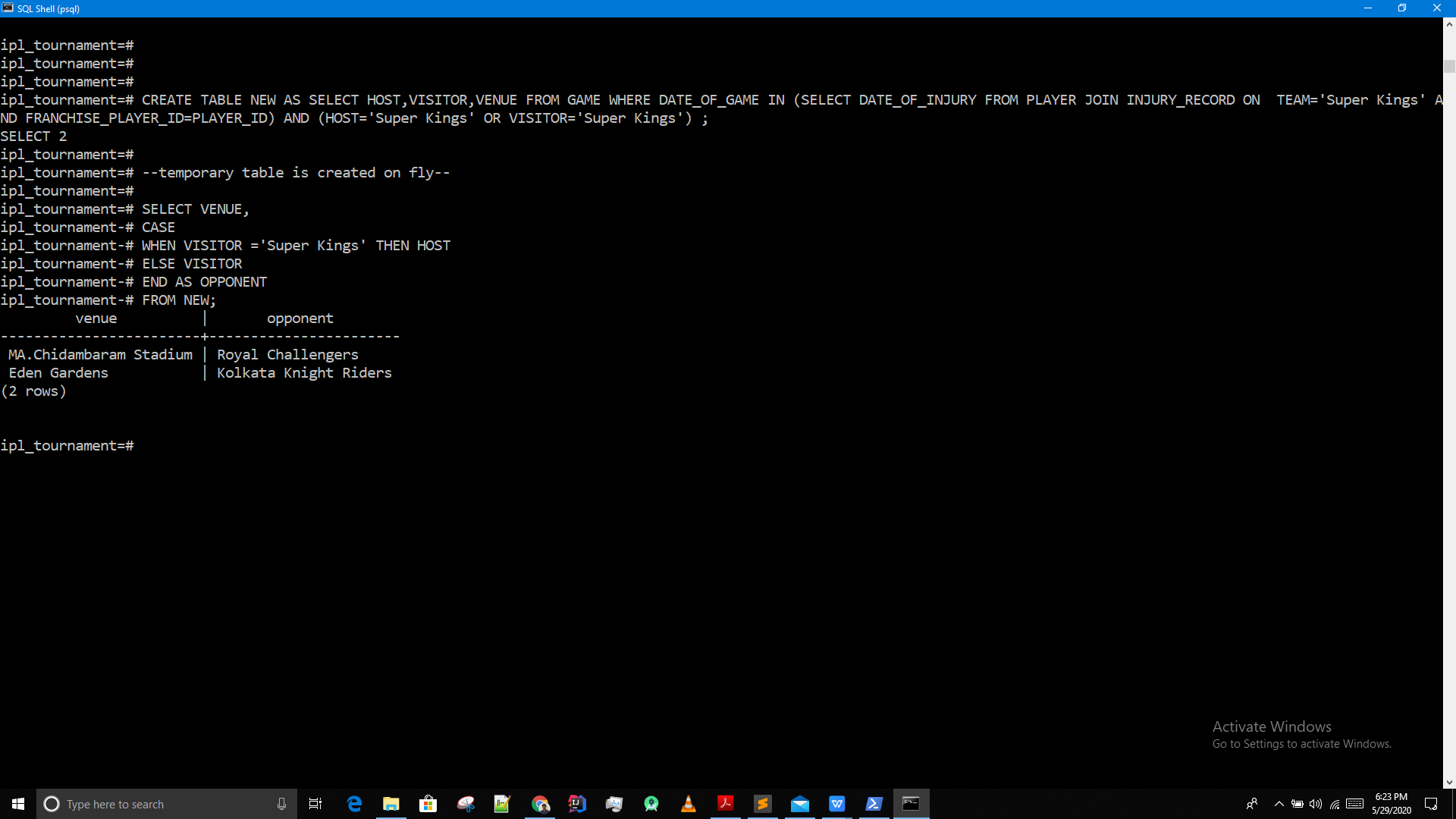
**WHEN** VISITOR ='Super Kings' THEN HOST

**ELSE** VISITOR

**END AS** OPPONENT

FROM NEW;

OUTPUT



**III.Aggregate Queries ( Sum ,count ,group by )**

**English statement :** What is the total no.of.corporate sponsoring the team 'Super Kings'?

**Query\_4 :SELECT COUNT(CORPORATE\_NAME)** as total\_sponsors FROM SPONSOR **WHERE TEAM='Super Kings';**

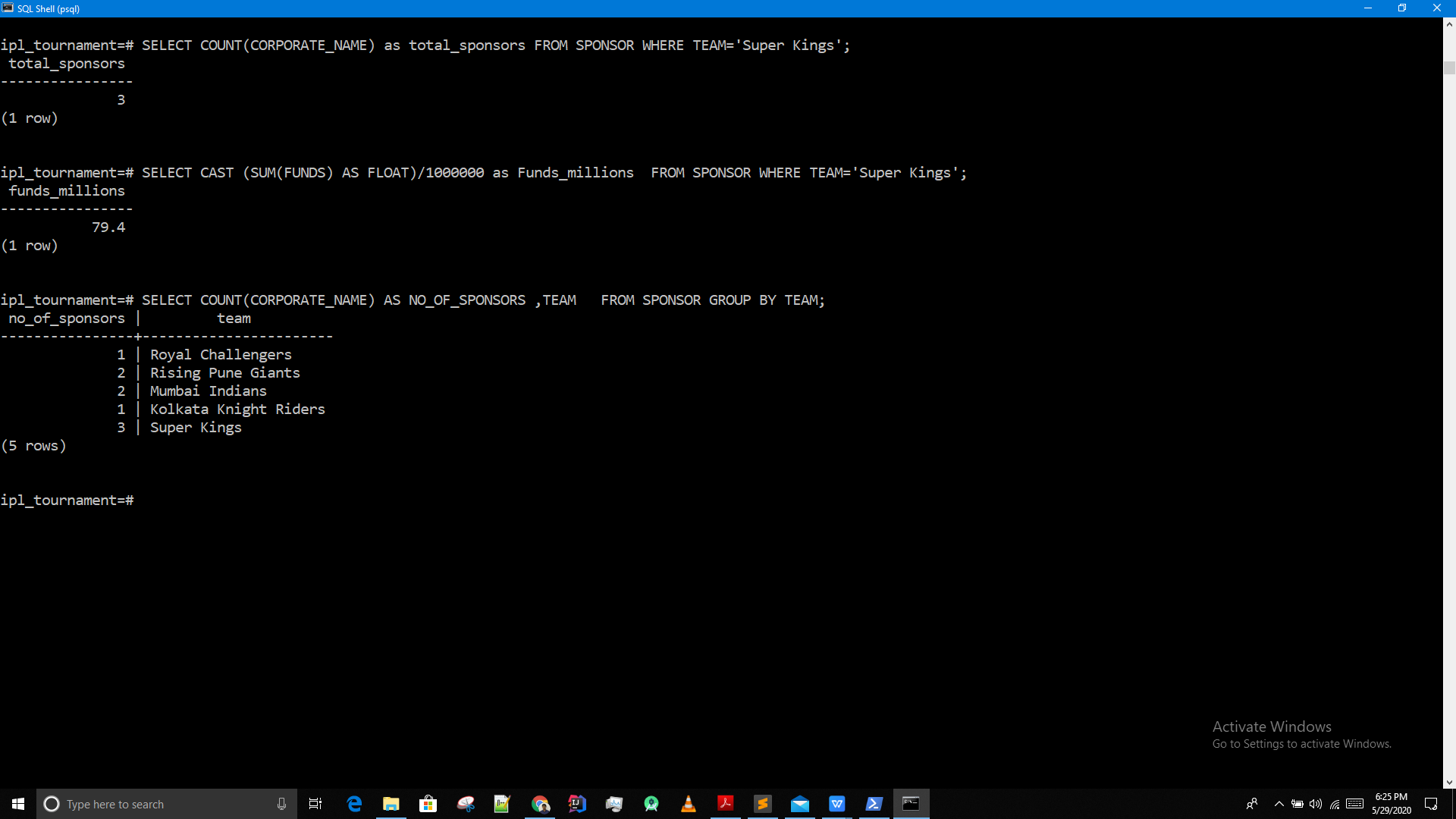
**English statement :** Retrieve the total amount invested(Millions) on 'Super Kings' franchise.

**Query\_5 : SELECT CAST (SUM(FUNDS) AS FLOAT)/1000000** as Funds\_millions FROM SPONSOR WHERE TEAM='Super Kings';

**English statement** : List the total no.of.sponsors per team along with the team\_name

**Query\_6 : SELECT COUNT(CORPORATE\_NAME)** AS NO\_OF\_SPONSORS **,**TEAMFROM SPONSOR **GROUP BY TEAM;**

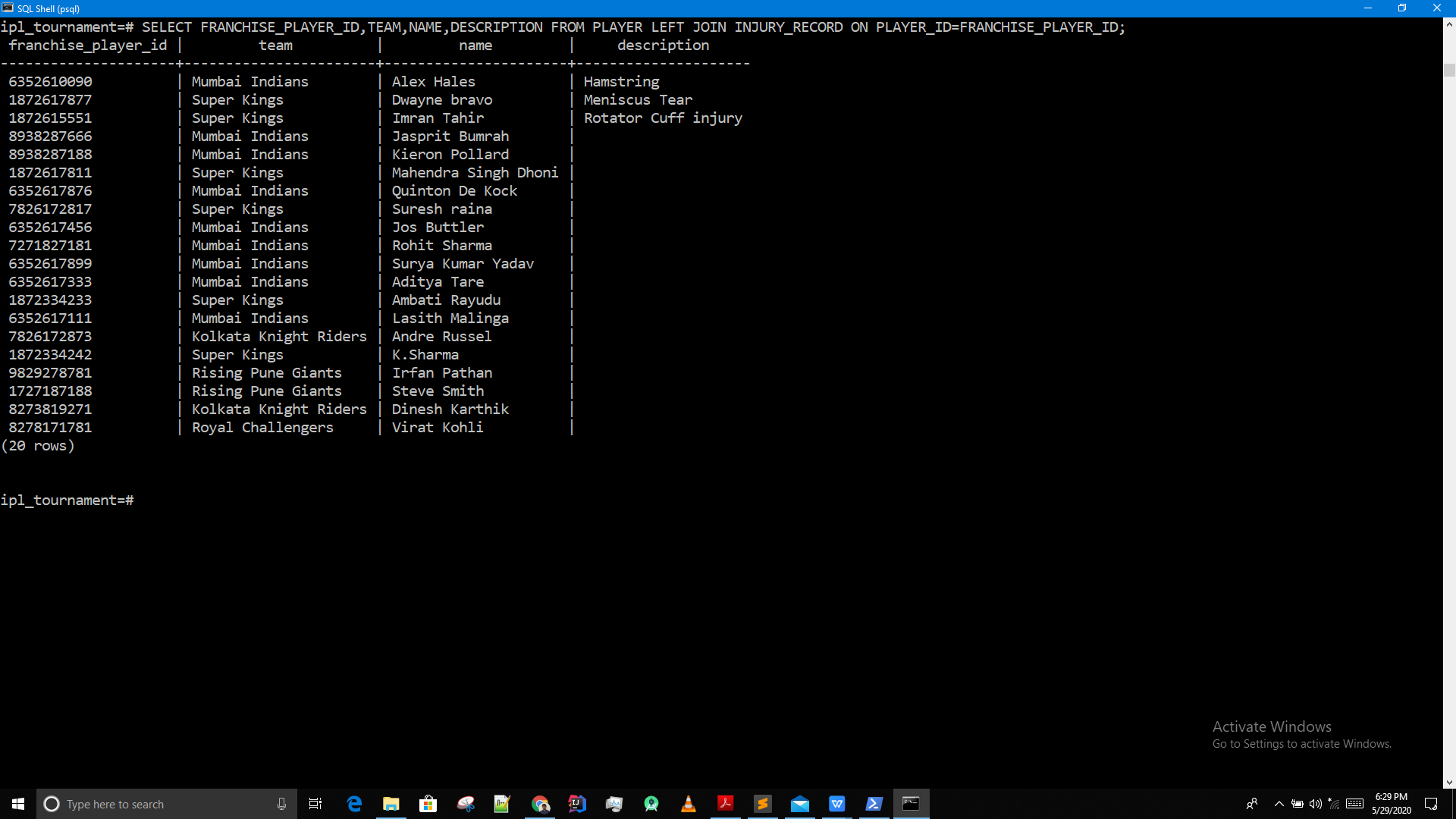
**OUTPUT OF ALL AGGREGATE QUERIES**

****

**IV. Outer Join Queries**

**English statement :** Perform a left outer join on players table and injury\_record of players based on their player\_id and list the name,id,team\_name and injury description.

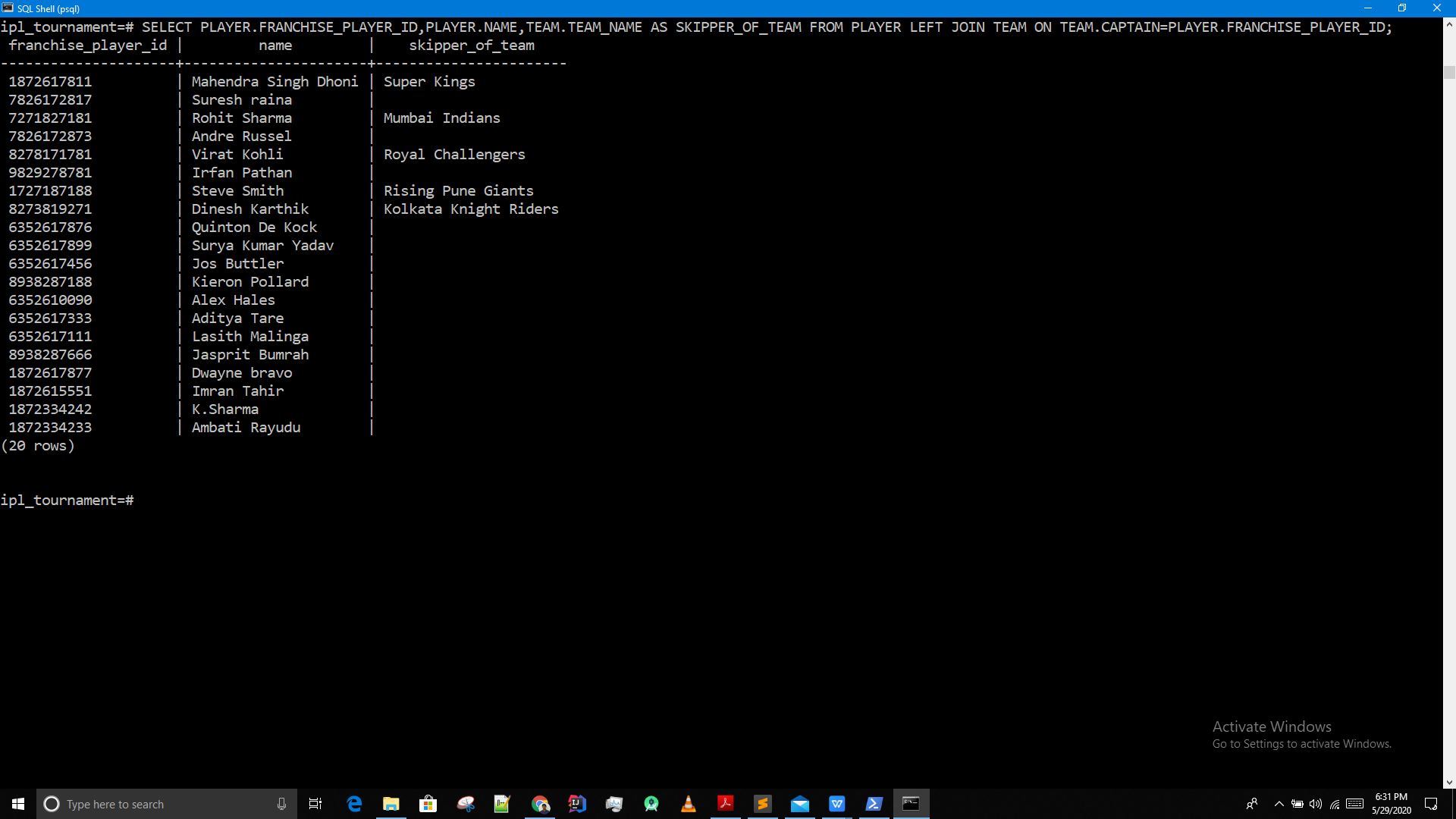
**Query\_7 : SELECT** FRANCHISE\_PLAYER\_ID,TEAM,NAME,DESCRIPTION **FROM** PLAYER **LEFT JOIN** INJURY\_RECORD **ON PLAYER\_ID=FRANCHISE\_PLAYER\_ID;**



**The EMPTY FIELDS signify that not all columns on the Left side of join find a match with the relation on right side**

**English statement :** Perform an outer join to list the players' name ,team\_name for which they are the captain**.**

**Query\_8: SELECT** PLAYER.FRANCHISE\_PLAYER\_ID,PLAYER.NAME,TEAM.TEAM\_NAME AS SKIPPER\_OF\_TEAM FROM PLAYER **LEFT JOIN TEAM ON** **TEAM.CAPTAIN=PLAYER.FRANCHISE\_PLAYER\_ID**;



**V.Correlated Sub Queries :** A correlated sub-query executes once for each candidate row considered by the outer query.

**English statement =** Select the player with highest skill level in every Participating team along with his skill\_level.

**Query\_9: SELECT** NAME,SKILL,TEAM FROM **PLAYER AS P** WHERE SKILL **= ( SELECT MAX(SKILL)** FROM **PLAYER D WHERE D.TEAM=P.TEAM );**

# q88

# Conclusion

**Capabilities of IPL\_TOURNAMENT DB system**

i.The designed IPL database **ensures that data integrity is preserved** across channels/relations with viable and necessary check constraints and Triggers.

ii.While staying **consistent and scalable**, this IPL\_tournament DB lets multiple people view and work with data simultaneously.

iii.The relations have been decomposed into the NORMAL FORMS hence **minimizing any redundancy** in the database tables.

iv.The DB **allows the user to schedule the matches\games** between participating teams at ease as there are **triggers which can warn /raise exception when a violation occurs**.

v.The team-management of various teams can obtain details of their team-player's injury record, schedule with ease. Hence the designed DB **supports multiple views.**

**Limitations**

1. The Data types used are of fixed length in few cases or too big ,with a deeper investigation about the types of data to be recorded,maybe we can minimize the size of data type used and leverage between content and space.

**Future Developments**

1. **Inclusion of data visualization** such as displaying No.of..titles secured by a team through a graph
2. **Real time analysis of team’s performance** using periodically updated data.

Extending the **DB to a CLOUD BASED DB** distribution provide access in terms of services to the client.