Final Project - Problem Statement PROJECT DONE BY AKSHAY ANAND.

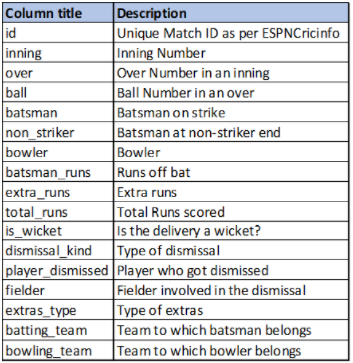
**Final Project - Sports Analytics using SQL**

In this project, you have to perform the job of a sports analyst. You are given two datasets related to IPL (Indian Premier League) cricket matches. One dataset contains ball-by-ball data and the other contains match-wise data. You have to import the datasets into an SQL database and perform the tasks given in this assignment to find important insights from this dataset.

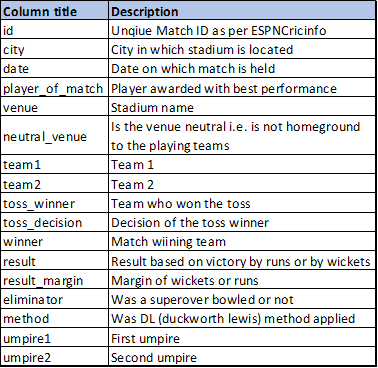
**About the Data**

Please download the datasets by clicking [here](https://trainings.internshala.com/uploads/sql-data-analytics/content_media/Data_For_Final_ProjectIPLMatches_IPLBall.zip) and have them ready before we get started.

The first CSV file is for ball-by-ball data and it has information of all the 193468 balls bowled between the years 2008 and 2020. It has 17 columns and below is the details of those 17 columns:



The second file contains match-wise data and has data of 816 IPL matches. This table has 17 columns and below is a short description of the columns in this table:



**Write queries for the following tasks:**

1. Create a table named ‘matches’ with appropriate data types for columns

CREATE TABLE matches (

match\_id int,

city varchar,

date date,

player\_of\_match varchar,

venue varchar,

neutral\_venue int,

team1 varchar,

team2 varchar,

toss\_winner varchar,

toss\_decision varchar,

winner varchar,

result\_mode varchar,

result\_margin int,

eliminator varchar,

method\_dl varchar,

umpire1 varchar,

umpire2 varchar

);

1. Create a table named ‘deliveries’ with appropriate data types for columns

CREATE TABLE deliveries (

match\_id int,

inning int,

over int,

ball int,

batsman varchar,

non\_striker varchar,

bowler varchar,

batsman\_runs int,

extra\_runs int,

total\_runs int,

wicket\_ball int,

dismissal\_kind varchar,

player\_dismissed varchar,

fielder varchar,

extras\_type varchar,

batting\_team varchar,

bowling\_team varchar

);

1. Import data from csv file ’IPL\_matches.csv’ attached in resources to the table ‘matches’ which was created in Q1

copy matches from 'C:\Program Files\PostgreSQL\14\data\IPL\IPL\_matches.csv' CSV header ;

1. Import data from csv file ’IPL\_Ball.csv’ attached in resources to the table ‘deliveries’ which was created in Q2

copy deliveries from 'C:\Program Files\PostgreSQL\14\data\IPL\IPL\_Ball.csv' CSV header ;

1. Select the top 20 rows of the *deliveries*table after ordering them by id, inning, over, ball in ascending order.

select \* from deliveries order by match\_id, inning,over,ball limit 20;

select \* from deliveries limit 20;

1. Select the top 20 rows of the *matches*table.

select \* from matches limit 20;

1. Fetch data of all the matches played on 2nd May 2013 from the *matches*table.

select \* from matches where date = '02-05-2013';

1. Fetch data of all the matches where the result mode is ‘runs’ and margin of victory is more than 100 runs.

select \* from matches where result\_mode = 'runs' and result\_margin > 100;

1. Fetch data of all the matches where the final scores of both teams tied and order it in descending order of the date.

select \* from matches where result\_mode ='tie' order by date desc;

1. Get the count of cities that have hosted an IPL match.

select count (distinct city) from matches;

1. Create table *deliveries\_v02*with all the columns of the table ‘*deliveries’*and an additional column *ball\_result*containing values *boundary*, *dot*or *other*depending on the *total\_run*(boundary for >= 4, dot for 0 and other for any other number)  
   (Hint 1 : CASE WHEN statement is used to get condition based results)  
   (Hint 2: To convert the output data of select statement into a table, you can use a subquery. Create table *table\_name*as *[entire select statement].*

create table deliveries\_v02 as select \*,

CASE WHEN total\_runs >= 4 THEN 'boundary'

WHEN total\_runs = 0 THEN 'dot'

ELSE 'other'

END as ball\_result

FROM deliveries;

1. Write a query to fetch the total number of boundaries and dot balls from the *deliveries\_v02*table.

select ball\_result, count (\*) from deliveries\_v02 group by ball\_result;

select ball\_result, count (ball\_result) from deliveries\_v02 where ball\_result='boundary' or ball\_result='dot' group by ball\_result;

1. Write a query to fetch the total number of boundaries scored by each team from the *deliveries\_v02*table and order it in descending order of the number of boundaries scored.

select batting\_team, count(\*) from deliveries\_v02 where ball\_result = 'boundary' group by batting\_team order by count desc;

1. Write a query to fetch the total number of dot balls bowled by each team and order it in descending order of the total number of dot balls bowled.

select bowling\_team, count(\*) from deliveries\_v02 where ball\_result = 'dot' group by bowling\_team order by count desc;

1. Write a query to fetch the total number of dismissals by dismissal kinds where dismissal kind is not NA

select dismissal\_kind, count (\*) from deliveries where dismissal\_kind <> 'NA' group by dismissal\_kind order by count desc;

1. Write a query to get the top 5 bowlers who conceded maximum extra runs from the *deliveries*table

select bowler, sum(extra\_runs) as total\_extra\_runs from deliveries group by bowler order by total\_extra\_runs desc limit 5;

1. Write a query to create a table named *deliveries\_v03*with all the columns of *deliveries\_v02*table and two additional column (named *venue*and *match\_date*) of *venue*and *date*from table *matches*

create table deliveries\_v03 AS SELECT a.\*, b.venue, b.match\_date from

deliveries\_v02 as a left join (select max(venue) as venue, max(date) as match\_date, match\_id from matches group by match\_id) as b on a.match\_id = b.match\_id;

1. Write a query to fetch the total runs scored for each venue and order it in the descending order of total runs scored.

select venue, sum(total\_runs) as runs from deliveries\_v03 group by venue order by runs desc;

1. Write a query to fetch the year-wise total runs scored at *Eden Gardens*and order it in the descending order of total runs scored.

select extract(year from match\_date) as IPL\_year, sum(total\_runs) as runs from deliveries\_v03 where venue = 'Eden Gardens' group by IPL\_year order by runs desc;

1. Get unique team1 names from the *matches*table, you will notice that there are two entries for*Rising Pune Supergiant* one with *Rising Pune Supergiant* and another one with *Rising Pune Supergiant****s***.  Your task is to create a *matches\_corrected*table with two additional columns *team1\_corr*and *team2\_corr*containing team names with replacing *Rising Pune Supergiant****s*** with *Rising Pune Supergiant*. Now analyse these newly created columns.

create table matches\_corrected as select \*, replace(team1, 'Rising Pune Supergiants', 'Rising Pune Supergiant') as team1\_corr , replace(team2, 'Rising Pune Supergiants', 'Rising Pune Supergiant') as team2\_corr from matches;

select distinct team1 from matches;select distinct team1\_corr from matches\_corrected;

1. Create a new table deliveries\_v04 with the first column as ball\_id containing information of match\_id, inning, over and ball separated by ‘-’ (For ex. 335982-1-0-1 match\_id-inning-over-ball) and rest of the columns same as deliveries\_v03)

create table deliveries\_v04 as select concat(match\_id,'-',inning,'-',over,'-',ball) as ball\_id, \* from deliveries\_v03;

create table deliveries\_v04 as select (match\_id||'-'||inning||'-'||over||'-'||ball) as ball\_id, \* from deliveries\_v03;

1. Compare the total count of rows and total count of distinct ball\_id in deliveries\_v04;

select count(\*) from deliveries\_v04; 🡨 COUNT OF ROWS

select count(distinct ball\_id) from deliveries\_v04; 🡨 COUNT OF DISTINCT BALL\_ID

1. SQL Row\_Number() function is used to sort and assign row numbers to data rows in the presence of multiple groups. For example, to identify the top 10 rows which have the highest order amount in each region, we can use row\_number to assign row numbers in each group (region) with any particular order (decreasing order of order amount) and then we can use this new column to apply filters. Using this knowledge, solve the following exercise. You can use hints to create an additional column of row number.  
   Create table deliveries\_v05 with all columns of deliveries\_v04 and an additional column for row number partition over ball\_id. (HINT : Syntax to add along with other columns,  row\_number() over (partition by ball\_id) as r\_num)

create table deliveries\_v05 as select \*, row\_number() over (partition by ball\_id) as r\_num from deliveries\_v04;

SELECT \* FROM deliveries\_v05;

create table deliveries\_v0511 as select \*, row\_number() over (partition by ball\_id ORDER BY ball\_id) as r\_num from deliveries\_v04;

SELECT \* FROM deliveries\_v0511 ;

create table deliveries\_v0512 as select \*, row\_number() over (partition by ball\_id ORDER BY ball\_id) as r\_num from deliveries\_v04 ORDER BY ball\_id;

SELECT \* FROM deliveries\_v0512 ;

1. Use the r\_num created in deliveries\_v05 to identify instances where ball\_id is repeating. (HINT : select \* from deliveries\_v05 WHERE r\_num=2;)

select count(\*) from deliveries\_v05;

select sum(r\_num) from deliveries\_v05;

SINCE SUM(r\_num) CAME OUT TO BE MORE THAN COUNT(\*) , IT MEANS SOME INSTANCES ARE REPEATING.

select count(\*) from deliveries\_v05 where r\_num=1;

select count(\*) from deliveries\_v05 where r\_num=2;

WE FIND 10 INSTANCES OF r\_num=2 AND THE REST OF r\_num=1.

select \* from deliveries\_v05 order by r\_num limit 20;

select \* from deliveries\_v05 WHERE r\_num=2;

select DISTINCT ball\_id from deliveries\_v05 WHERE r\_num=2;

SELECT \* FROM deliveries\_v0511 ;

SELECT \* FROM deliveries\_v0511 where r\_num=2;

select count(\*) from deliveries\_v0511;

select sum(r\_num) from deliveries\_v0511;

select count(\*) from deliveries\_v0511 where r\_num=1;

select count(\*) from deliveries\_v0511 where r\_num=2;

select \* from deliveries\_v0511 order by r\_num limit 20;

select \* from deliveries\_v0511 WHERE r\_num=2;

select DISTINCT ball\_id from deliveries\_v0511 WHERE r\_num=2;

SELECT \* FROM deliveries\_v0512 where r\_num=2;

select count(\*) from deliveries\_v0512;

select sum(r\_num) from deliveries\_v0512;

select count(\*) from deliveries\_v0512 where r\_num=1;

select count(\*) from deliveries\_v0512 where r\_num=2;

select \* from deliveries\_v0512 order by r\_num limit 20;

select \* from deliveries\_v0512 WHERE r\_num=2;

select DISTINCT ball\_id from deliveries\_v0512 WHERE r\_num=2;

1. Use subqueries to fetch data of all the ball\_id which are repeating. (HINT: SELECT \* FROM deliveries\_v05 WHERE ball\_id in (select BALL\_ID from deliveries\_v05 WHERE r\_num=2);

SELECT \* FROM deliveries\_v05 WHERE ball\_id in (select BALL\_ID from deliveries\_v05 WHERE r\_num=2);

SELECT ball\_id,r\_num FROM deliveries\_v0511 WHERE ball\_id in (select BALL\_ID from deliveries\_v0511 WHERE r\_num=2);

SELECT ball\_id,r\_num FROM deliveries\_v0512 WHERE ball\_id in (select BALL\_ID from deliveries\_v0512 WHERE r\_num=2);

REFERENCES🡪

<https://www.educba.com/partition-by-in-sql/>

<https://stackoverflow.com/questions/51761481/what-is-the-role-of-order-by-in-the-partition-by-function>

<https://www.sqltutorial.org/sql-window-functions/sql-partition-by/>