2.9.x Final Assignment SQL

*Covered topics: Databases & SQL*

### Assignment Instructions

You will be working with the **European Soccer Database**, a collection of four individual CSV files that you will find inside the 2.9.x (data) European Soccer Database.zip compressed folder, containing:

* leagues.csv
* team.csv
* player.csv
* match.csv

Make a copy of this Google Doc and, for each of the tasks that you’ll find in the next page:

* Paste the SQL query that generates the solution right below the question;
* Write the answer to the question (when possible) in the following table.

|  |  |
| --- | --- |
| Question # | Answer |
| 1 | Not Required |
| 2 | Link to lucidchart: <https://lucid.app/lucidchart/87013f61-7640-4cba-a3dd-de62ae6e0aa6/edit?invitationId=inv_233defeb-d0f5-43a1-a68d-aeebafbc4deb#> |
| 3 | 2868 days |
| 4 | 2009/2010 England Premier League |
| 5 | Number of Seasons: 8.  During the 2013/2014 Season the Belgium Jupiler League has only disputed 12 matches |
| 6 | 10197 rows |
| 7 | 863 players |
| 8 | FC Barcelona. 112 goals scored |
| 9 | Real Madrid CF |
| 10 | 45 |

Data Analysis with SQL

Using the abovementioned database, complete the following tasks:

1. Create a new data set called “Final\_Exercise” in Google BigQuery and load each csv file as a separate table.
2. Using <https://lucid.app/>, create a schema that represents the relationship between all the tables:
   1. For each table, write to the left of the variable's name if it is a primary key (PK), a foreign key (FK) or just a simple variable (leave blank).
   2. For each table, write its shape (write the number of rows and columns near the table name).
   3. With a line, link the tables to each other through their keys (when possible).
3. How many days have passed from the oldest **Match** to the most recent one (dataset time interval)?

SELECT date\_diff(max(date), min(date), day) as Days\_From\_First\_Match\_To\_Last

FROM `angular-pursuit-349916.Final\_Exercise.Match`

1. Produce a table which, for each Season and **League** Name, shows the following statistics about the home goals scored:
   1. min
   2. average
   3. mid-range
   4. max
   5. sum

*Hint: there is no function for the mid-range, research it and calculate it.*Which combination of Season-League has the highest number of goals?

SELECT

season as Season,

name as League,

min(home\_team\_goal) as Min\_Goals,

round(avg(home\_team\_goal),1) as Average\_Goals,

round((max(home\_team\_goal)+min(home\_team\_goal))/2,1) as Mid\_range,

max(home\_team\_goal) as Max\_Goals,

sum(home\_team\_goal) as Goals\_Sum

FROM `angular-pursuit-349916.Final\_Exercise.Match` Match

LEFT JOIN `angular-pursuit-349916.Final\_Exercise.Leagues` Leagues

on Leagues.id=Match.league\_id

group by season,Leagues.name

order by Goals\_Sum desc

1. Find out how many unique seasons there are in the **Match** table.   
   Then write a query that shows, for each Season, the number of matches played by each League. Do you notice anything out of the ordinary?

SELECT

count(distinct season) as Number\_of\_Seasons

FROM `angular-pursuit-349916.Final\_Exercise.Match`

--------

SELECT season as Season,

name as League,

count(\*) as Matches\_Played

FROM `angular-pursuit-349916.Final\_Exercise.Match` Match

LEFT JOIN `angular-pursuit-349916.Final\_Exercise.Leagues` Leagues

on Leagues.id=Match.league\_id

group by season, name

order by Matches\_Played

1. Using Players as the starting point, create a new table (PlayerBMI) and add:
   1. a new variable that represents the players’ weight in kg (divide the mass value by 2.205) and call it kg\_weight;
   2. a variable that represents the height in metres (divide the cm value by 100) and call it m\_height;
   3. a variable that shows the body mass index (BMI) of the player;  
      *Hint: research how to calculate the formula of the BMI*
   4. Filter the table to show only the players with an optimal BMI (from 18.5 to 24.9).

CREATE TABLE `angular-pursuit-349916.Final\_Exercise.PlayerBMI` AS

SELECT \*,

weight/2.205 AS kg\_weight,

height/100 AS m\_height,

(weight/2.205)/((height/100)\*(height/100)) AS BMI

FROM `angular-pursuit-349916.Final\_Exercise.Player`

WHERE (weight/2.205)/((height/100)\*(height/100))>18.5 AND (weight/2.205)/((height/100)\*(height/100))<24.9

How many rows does this table have?

1. How many players do not have an optimal BMI?

11060-10197=863

1. Which **Team** has scored the highest total number of goals (home + away) during the most recent available season? How many goals has it scored?

Select team\_long\_name as Team,sum(Home\_goals) as Goals\_Scored

from

(select \*

from(SELECT

home\_team\_api\_id, sum(home\_team\_goal) as Home\_goals

FROM `angular-pursuit-349916.Final\_Exercise.Match`

where season="2015/2016"

group by home\_team\_api\_id

order by home\_team\_api\_id desc)

union all

select \*

from(SELECT

away\_team\_api\_id, sum(away\_team\_goal) as Away\_goals

FROM `angular-pursuit-349916.Final\_Exercise.Match`

where season="2015/2016"

group by away\_team\_api\_id

order by away\_team\_api\_id desc)) as a

left join `angular-pursuit-349916.Final\_Exercise.Team ` as b

on a.home\_team\_api\_id=b.team\_api\_id

group by home\_team\_api\_id, team\_long\_name

order by Goals\_Scored desc

1. Create a query that, for each season, shows the name of the team that ranks first in terms of total goals scored (the output table should have as many rows as the number of seasons).   
   Which team was the one that ranked first in most of the seasons?

select season as Season,team\_long\_name as Team,goals\_scored as Goals

from

(WITH summary AS (

SELECT p.team\_long\_name,

p.season,

p.goals\_scored,

ROW\_NUMBER() OVER(PARTITION BY p.season

ORDER BY p.goals\_scored DESC) AS rank

FROM

(select season, team\_long\_name,goals\_scored

from

(select season,team\_long\_name,goals\_scored

from

(select season,team\_long\_name,sum(goals) as goals\_scored

from

(SELECT season,home\_team\_api\_id,home\_team\_goal as goals FROM `angular-pursuit-349916.Final\_Exercise.Match`

union all

select season,away\_team\_api\_id,away\_team\_goal FROM `angular-pursuit-349916.Final\_Exercise.Match`) as M

left join `angular-pursuit-349916.Final\_Exercise.Team ` as T

on M.home\_team\_api\_id=T.team\_api\_id

group by season, team\_long\_name)

order by season, goals\_scored desc)

order by season, goals\_scored desc) p)

select \*

from summary

where rank=1)

order by season

1. From the query above create a new table (TopScorer) containing the top 10 teams in terms of total goals scored (*hint: add the team id as well*).   
   Then write a query that shows all the possible “pair combinations” between those 10 teams. How many “pair combinations” did it generate?

create table `angular-pursuit-349916.Final\_Exercise.Top\_Scorer` as select id,Team

from

(Select home\_team\_api\_id as id, team\_long\_name as Team,sum(Home\_goals) as Goals\_Scored

from

(select \*

from(SELECT

home\_team\_api\_id, sum(home\_team\_goal) as Home\_goals

FROM `angular-pursuit-349916.Final\_Exercise.Match`

group by home\_team\_api\_id

order by home\_team\_api\_id desc)

union all

select \*

from(SELECT

away\_team\_api\_id, sum(away\_team\_goal) as Away\_goals

FROM `angular-pursuit-349916.Final\_Exercise.Match`

group by away\_team\_api\_id

order by away\_team\_api\_id desc)) as a

left join `angular-pursuit-349916.Final\_Exercise.Team ` as b

on a.home\_team\_api\_id=b.team\_api\_id

group by home\_team\_api\_id, team\_long\_name

order by Goals\_Scored desc

limit 10)

--------

select e1.Team, e2.Team

from `angular-pursuit-349916.Final\_Exercise.Top\_Scorer` e1

inner join `angular-pursuit-349916.Final\_Exercise.Top\_Scorer` e2 on e1.Team < e2.Team