European Soccer Database

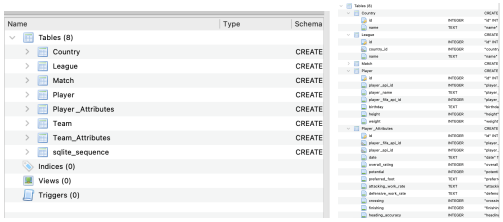
BY: Anik Shikarpuri

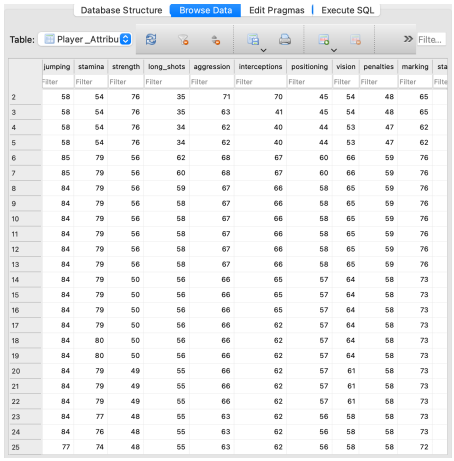
https://www.kaggle.com/datasets/hugomathien/soccer

The database I selected for this assignment is on European soccer. It has information about players, teams, matches, player attributes, and team attributes. This information is really interesting because we can dissect each player individually. And it is a great tool for teams looking for new players.

The dataset has very specific details about player attributes such as speed, dominant foot, shooting power, freekick rating, penalties scored, tackles won, player overall rating, and much more.There are eight different entries for this database. There is also data on scores for games between the European top teams from 2008 - 2020. The reason this information is relevant is because we can take what we know from the team's performance and find new players on the market that could improve their team as a whole. For example, in soccer you want the left wing position to usually be left footed so they can cross the ball in more easily. So using the database we can find a left winger who is left footed with a high player rating. Which will make it much easier for teams to find good replacements for their current players. The data was retrieved from FIFA (Federation Internationale de Football Association). Which is the organization that runs all these games and teams. So the data given is accurate because it was taken from the source itself.

When I was loading the data there were a lot of challenges I faced. First of which was finding which data I actually need. There was so much data given that it was hard to find what I needed. As you can see in the image below the database was divided into 8 different tables and each of those tables were then divided into several parts. I loaded the data by just looking going to the ‘browse data’ tab and that allowed me to get a look at what I want to use.



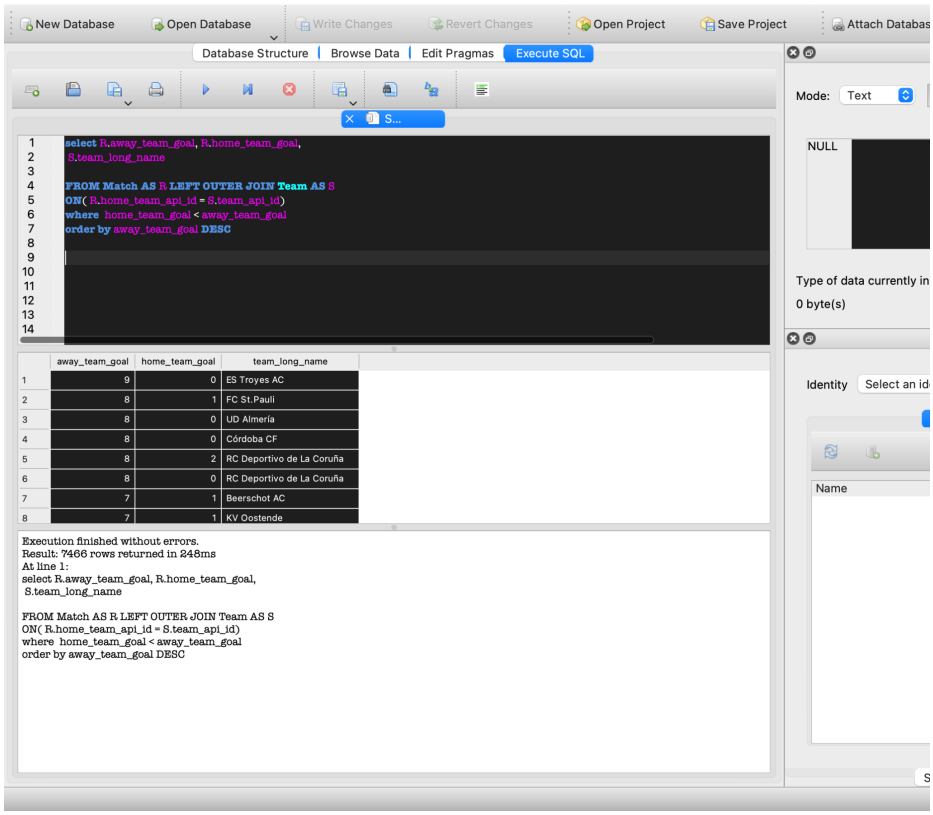


	jumping	stamina	strength	long_shots	aggression	interceptions	positioning	vision	penalties	marking	sta
2	58	54	76	35	71	70	45	54	48	65	
3	58	54	76	35	63	41	45	54	48	65	
4	58	54	76	34	62	40	44	53	47	62	
5	58	54	76	34	62	40	44	53	47	62	
6	85	79	56	62	68	67	60	66	59	76	
7	85	79	56	60	68	67	60	66	59	76	
8	84	79	56	59	67	66	58	65	59	76	
9	84	79	56	58	67	66	58	65	59	76	
10	84	79	56	58	67	66	58	65	59	76	
11	84	79	56	58	67	66	58	65	59	76	
12	84	79	56	58	67	66	58	65	59	76	
13	84	79	56	58	67	66	58	65	59	76	
14	84	79	50	56	66	65	57	64	58	73	
15	84	79	50	56	66	65	57	64	58	73	
16	84	79	50	56	66	65	57	64	58	73	
17	84	79	50	56	66	62	57	64	58	73	
18	84	80	50	56	66	62	57	64	58	73	
19	84	80	50	56	66	62	57	64	58	73	
20	84	79	49	55	66	62	57	61	58	73	
21	84	79	49	55	66	62	57	61	58	73	
22	84	79	49	55	66	62	57	61	58	73	
23	84	77	48	55	63	62	56	58	58	73	
24	84	76	48	55	63	62	56	58	58	73	
25	77	74	48	55	63	62	56	58	58	72	

The picture above shows the initial table for the Player Attributes table. Which was one of the tables I used for my queries.

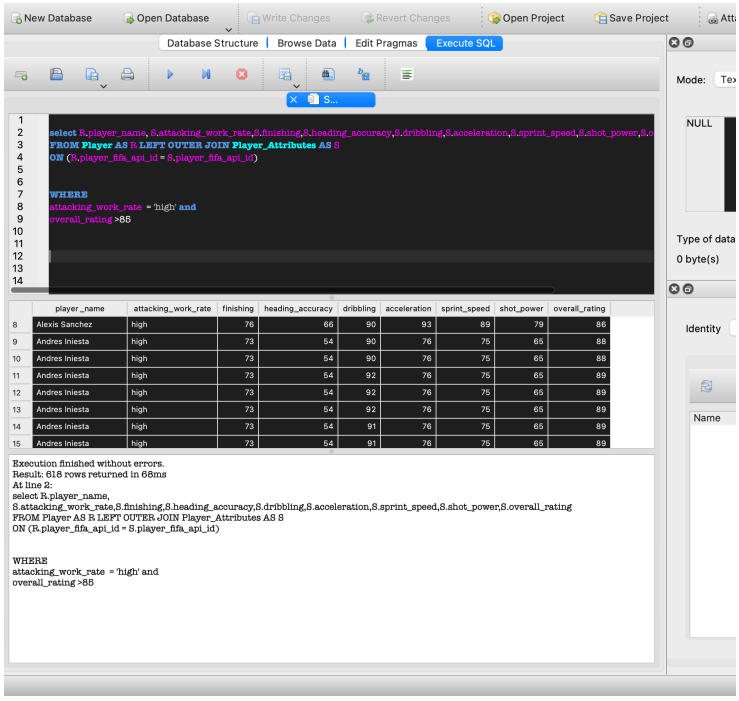
I created three queries to answer the question of which teams really need new players, and what players should they look for. I divided this into three parts.

1. Which teams have lost by the biggest goal differentials at home games?

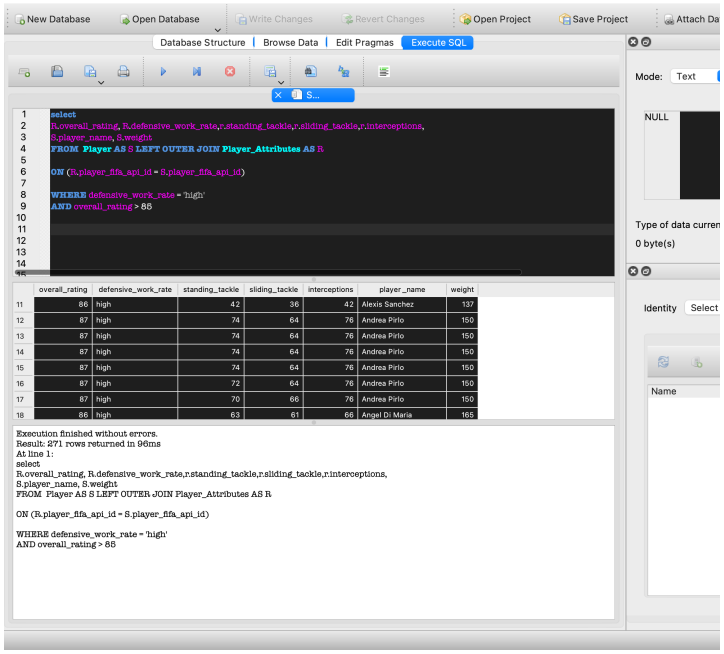


	away_team_goal	home_team_goal	team_long_name
1	9	0	ES Troyes AC
2	8	1	FC St.Pauli
3	8	0	UD Almería
4	8	0	Córdoba CF
5	8	2	RC Deportivo de La Coruña
6	8	0	RC Deportivo de La Coruña
7	7	1	Beerschot AC
8	7	1	KV Oostende

2. Which attackers would be best suited for these teams?



3. Which defensive players are best suited for these teams?



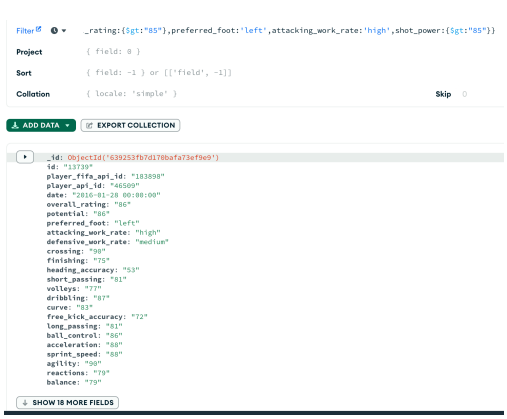
As you can see I loaded the data by taking different parts from different tables and creating a correlation between them. In the three tables I created, I connected them through their similar columns which allowed me to create a connection. One of the challenges I had was finding the correlation and how to connect the two tables together. And I also had an issue with my table

having repeating columns. To summarize the three tables, in table 1 I made a query that finds the teams with the biggest home game losses. I did this by finding the team api ID from the ‘player’ and ‘player attributes’ dataset. That way I can find the connection and then give the names of the team with the biggest home loss. For the second table I found a player with player ratings higher than 85. And gave all the attributes of an attacking player. So the teams can look for good attacking players with all the relevant attributes that could help them find the right player. I then did the same thing but for defensive players in table three. So this will allow teams to see which of them are performing the worst. And this will lead them to wanting new players which table 2 and 3 help them find.I was able to pull the same data from different tables. But I think no relation is better because it has lower data storage and has a higher scalability.

Part 2: NoSQL

So now we will take two of the questions I used for SQL, and try to find the same data in MongoDB. For MongoDB I will try to find solid defensive and offensive players that can make a significant impact to the team. In other words, which attackers would be best suited for these teams? And which defenders would be best suited for these teams? Like I stated previously, the NoSQL data tool I used was MongoDB. The difference is MongoDB uses python, therefore using a different language than SQL. It also uses a dynamic schema and is best suited for hierarchical data storage. However, it is not good for complex queries. I had a lot of issues when trying to load the data. The problem I had was trying to convert a .zip to a .csv. The data I used was downloaded as a .zip file, which needed to be converted to .csv. I tried different website converters but nothing was working. So, I opened the data in SQLite and then I exported the project as a .csv file so I could then open it in MongoDB. This solved the problem and allowed me to begin writing the queries in MongoDB. Below are my two queries:

1) Which attacking player will be best suited for a team?



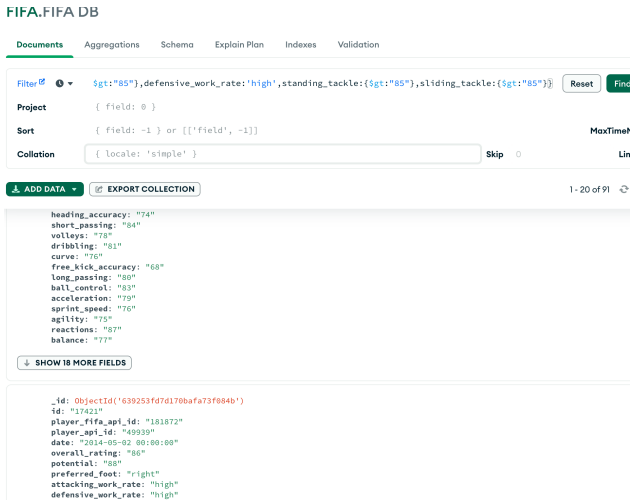
My Query:

```
{
    overall_rating: {
        $gt: "85"
    },
    preferred_foot: 'left',
    attacking_work_rate: 'high',
    shot_power: {
        $gt: "85"
    }
}
```

2) Which defensive players are best for the team?

My Query:

```
{
    overall_rating: {
        $gt: "85"
    },
    defensive_work_rate: 'high',
    standing_tackle: {
        $gt: "85"
    },
    sliding_tackle: {
        $gt: "85"
    }
}
```



So here we see that I was able to get the same data from MongoDB as SQLite. However, there are some issues. First of all it did find information I wanted, but it also gave all the information that falls under the data table with it. So it did give me the information I asked for but it also gave me data I didn't want. The other problem I had was that I couldn't use the 'match' table because when I tried adding the data it said it was "750,000" documents. Which meant it took way too long for it to import the dataset. So I chose not to use it. Lastly, the programming in itself is much more complicated for MongoDB than SQLite. After using both I would say SQLite would be preferable when trying to create complex queries. And the NoSQL data tool is more applicable when working with hierarchical data and if you want it to run well on the cloud. When working with large scale data the NoSQL tools are much better than SQL. This is because NoSQL was designed to be used across large diatrubuted systems due to the dynamic

schema. The other benefit being large amounts of data can be stored in NoSQL databases. So to conclude, NoSQL is much more suitable for large data sets but, when working with more complicated or challenging queries SQL is more suitable. After working on this project I would say if you are looking at data for personal use, SQL is much more user friendly and easier to find the data you are looking for.