**ISM 6218 - Section 003**

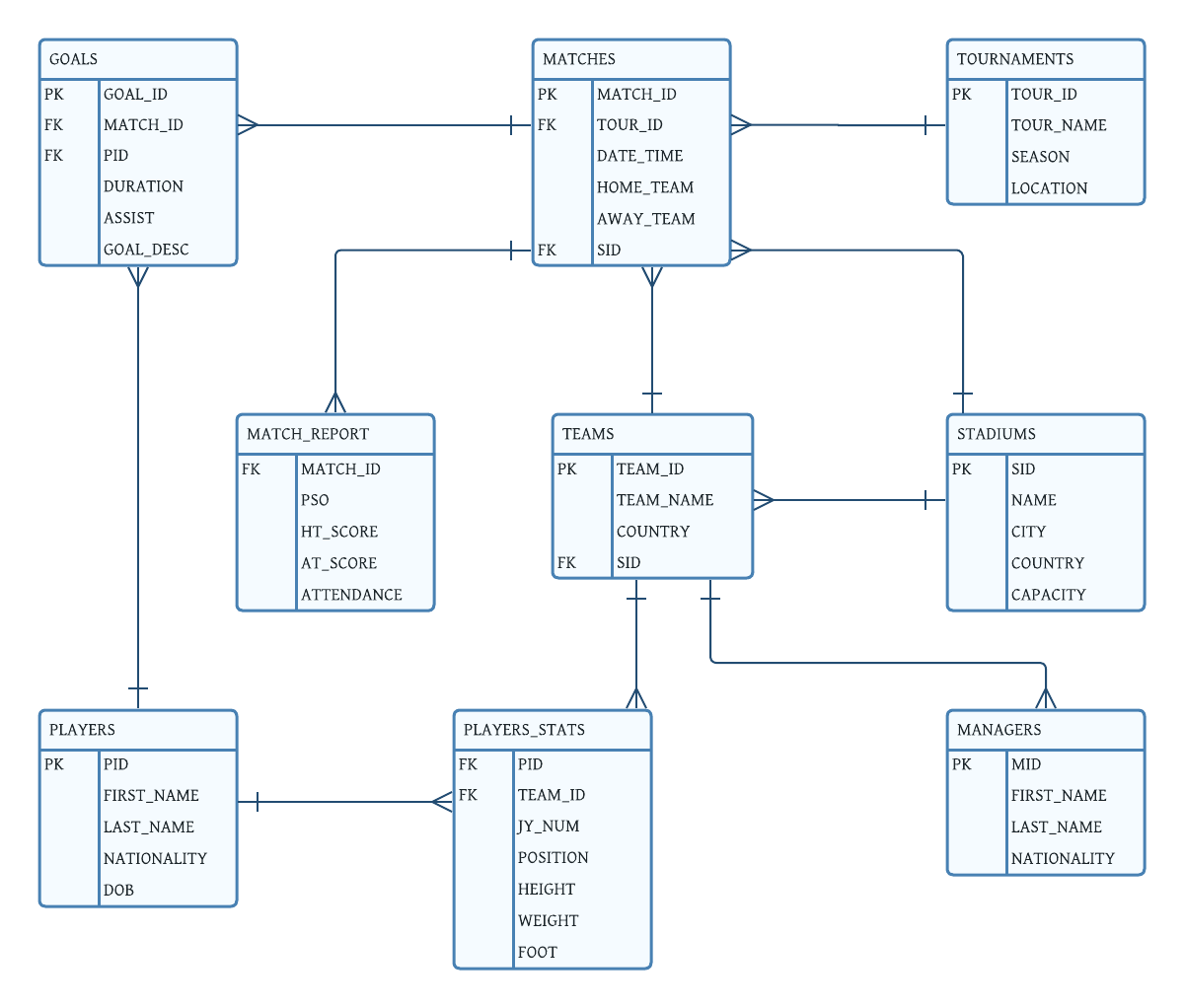
**Advanced Database Management**

**Final Project**

**Dataset**UEFA Champions League (UCL) is one of the biggest football competitions conducted by the Union of European Football Association. Started in 1955, UCL is one of the most viewed and anticipated football tournaments in the world. This dataset contains 7 tables that include teams, players details, tournaments, team managers, matches, goals, and stadiums. The dataset was sourced from Kaggle, use the following link to access the data: <https://www.kaggle.com/datasets/cbxkgl/uefa-champions-league-2016-2022-data/data>**.**

1. **Entity Relationship Design**

A logical database ER design, using normalization to control redundancy and integrity constraints for data quality.

****

1. **Query Writing**

SQL queries, transactions, and database programming for stored procedures. There are 3 types of queries which are point, range and scan queries with a total of 13 queries.

1. **Find the most common player positions:**

SELECT position, COUNT(pid) as total

FROM player\_stats

GROUP BY position

HAVING COUNT(pid) > 1

ORDER BY total DESC;

**Result:**



1. **How many goals did Sergio Ramos score?**

SELECT count(\*)

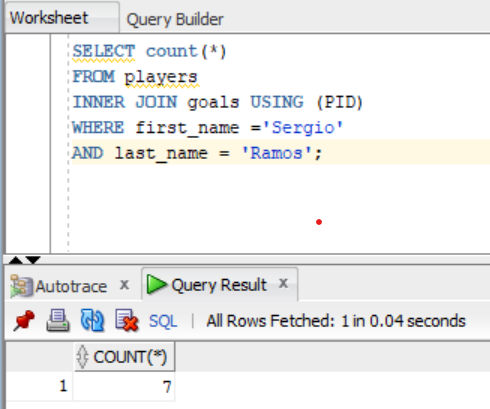
FROM players

INNER JOIN goals USING (PID)

WHERE first\_name ='Sergio'

AND last\_name = 'Ramos';

**Result:**



1. **Which stadium had the most goals scored by the home team?**

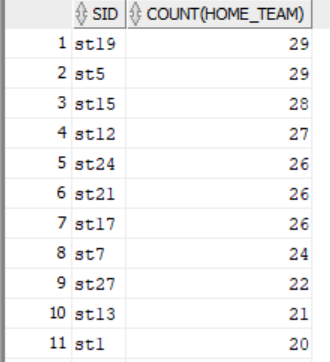
SELECT sid, COUNT(home\_team)

FROM matches

GROUP BY sid

ORDER BY COUNT(home\_team) DESC;

**Result:**



1. **Which teams have the tallest players in them? Get a list of all the teams and use descending order**

SELECT team\_name, country, ROUND(AVG(height)), ROUND(AVG(weight))

FROM player\_stats

INNER JOIN players USING(pid)

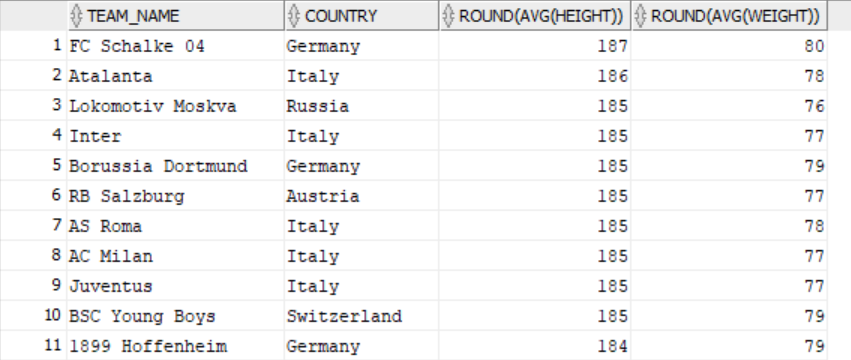
INNER JOIN teams USING(team\_id)

WHERE height IS NOT NULL

GROUP BY team\_name, country

ORDER BY ROUND(AVG(height)) DESC;

**Result:**



1. **All the matches that were played in stadiums with seating capacity greater than 60,000.**

SELECT city, name, AVG(capacity)

FROM matches

INNER JOIN stadiums USING (SID)

WHERE capacity > 60000

GROUP BY city, name

ORDER BY AVG(capacity) desc;

**Result:**



1. **How many goals did the home team and away team scored at each match? (we are in favor of home team, hence order by home team score)**

SELECT match\_id, home\_team, avg(ht\_score) as home\_team\_score, away\_team, avg(at\_score) as away\_team\_score

FROM matches

INNER JOIN match\_report using (match\_ID)

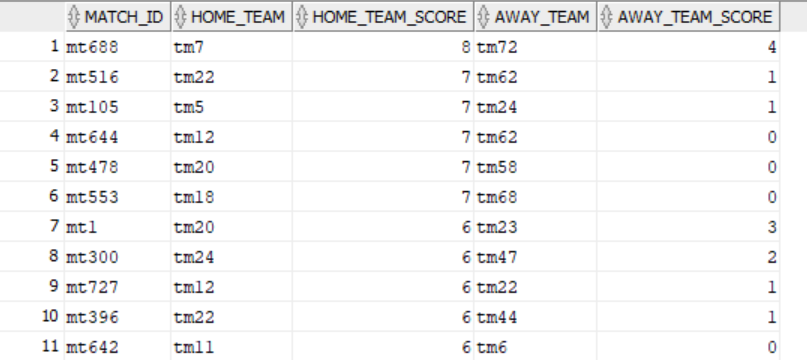
INNER JOIN teams using (sid)

INNER JOIN goals using (match\_ID)

GROUP BY match\_id, home\_team, away\_team

ORDER BY avg(ht\_score) desc, avg(at\_score) desc;

**Result:**



1. **The top goal scorers with number of goals scored:**

SELECT p.first\_name, p.last\_name, COUNT(g.goal\_id) as goals\_scored

FROM players p

JOIN goals g ON g.pid = p.pid

GROUP BY p.first\_name, p.last\_name

ORDER BY goals\_scored DESC;

**Result:**



1. **Players who were born before 1973**

SELECT first\_name, last\_name, dob

FROM players

WHERE dob BETWEEN '01-Jan-53' AND '01-Jan-73'

ORDER BY dob;

**Result:**

****

SELECT first\_name, last\_name

FROM player\_stats

inner join players2 using(pid)

WHERE position = 'Goalkeeper'

order by last\_name ;

**Result:**

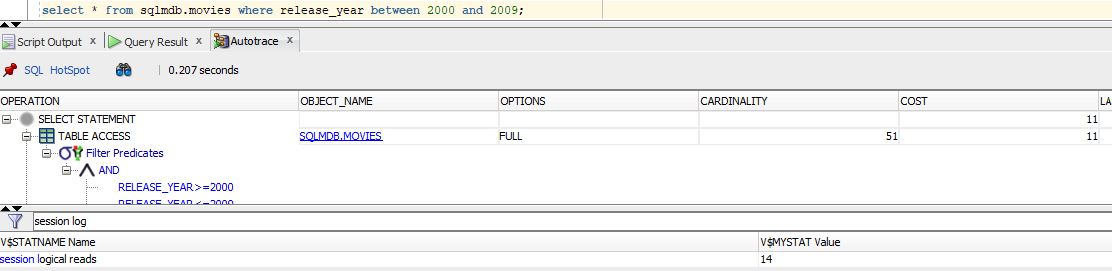


1. **Performance Tuning**

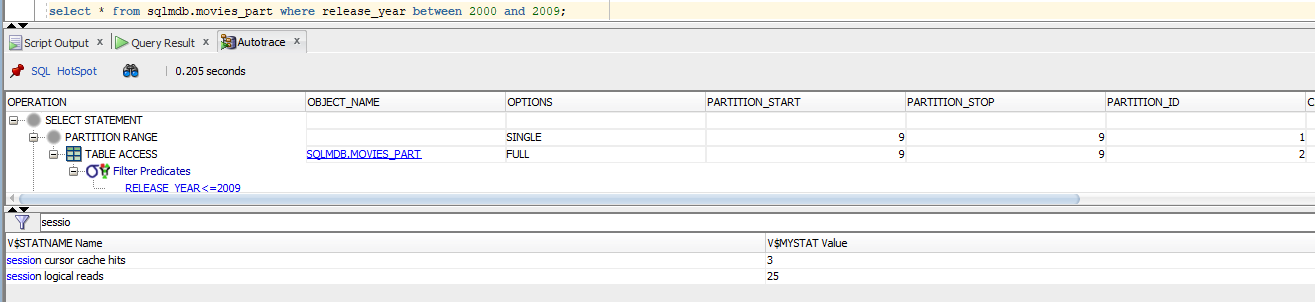
Every query in part 2 is shown in the screenshot “before/after” with their autotrace and session logical reads to demonstrate the effectiveness of our performance tuning. It is important to note that not every query would need a particular performance tuning (index,partitioning, optimizer modes, and parallel execution) as it may potentially increase the execution time. Here are some reasons why we decided to exclude a specific tuning in queries:

Indexing: indexing helps in targeted or pointed queries that would look for a specific value in a large dataset. If the dataset is small or queries do not benefit significantly from indexing, the overhead of maintaining indexes might not be justified.

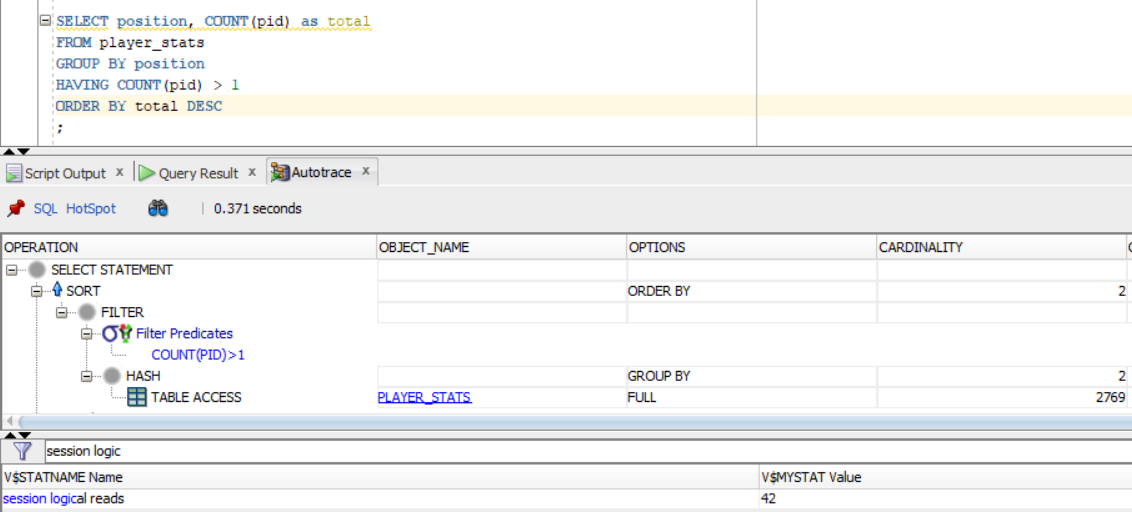
Partitioning: partitioning is usually used in doing any aggregate functions over groups of data that would have a specific condition. It could possibly decrease the performance even more. It is better to use group by in several cases rather than adding a partitioning if the changes in execution time is minimal due to the storage of partitioning. Moreover, if the data happen to overcome some adjustments, the partitioning would not be as effective or work anymore. It is important to note that some queries could benefit a lot from partitioning, where on the other hand, other queries will not. One of the examples on this is from our movies database, where the following query (example) did not benefit from the query (queries are shown above the autotrace):



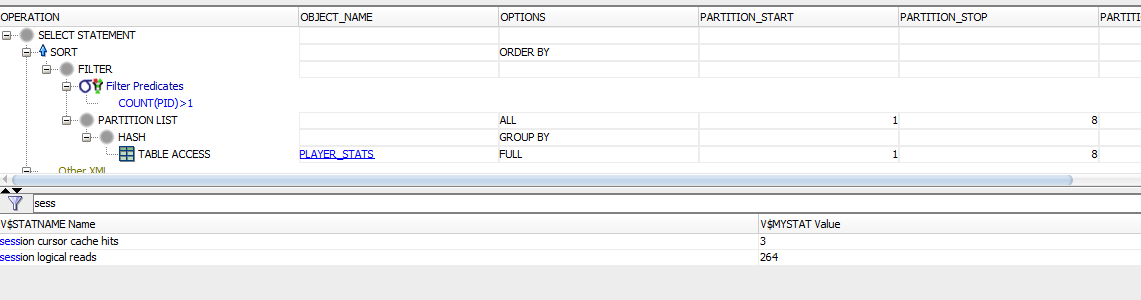
With partitioning:



Before performance tuning:



After performance tuning:



Explanation:

This is not a targeted query (this is a scan query) that prevents us from benefiting from the index. However, we did create the index on position in case if the query will be modified in the future and would use a where clause with position = ‘any value’:

CREATE INDEX player\_stats\_position ON player\_stats(position);

In the execution, the query used list partitioning. We used the following code to create a partitioned table:

CREATE TABLE player\_stats(

pid VARCHAR(10),

team\_id VARCHAR(10),

jy\_num INT,

position VARCHAR(20),

height INT,

weight INT,

foot CHAR CHECK (foot = 'R' OR foot = 'L')

)

PARTITION BY LIST (position) (

PARTITION Midfielder VALUES ('Midfielder'),

PARTITION Defender VALUES ('Defender'),

PARTITION Forward VALUES ('Forward'),

PARTITION Goalkeeper VALUES ('Goalkeeper'),

PARTITION Republic VALUES ('Republic'),

PARTITION Goalkeeping VALUES ('Goalkeeping'),

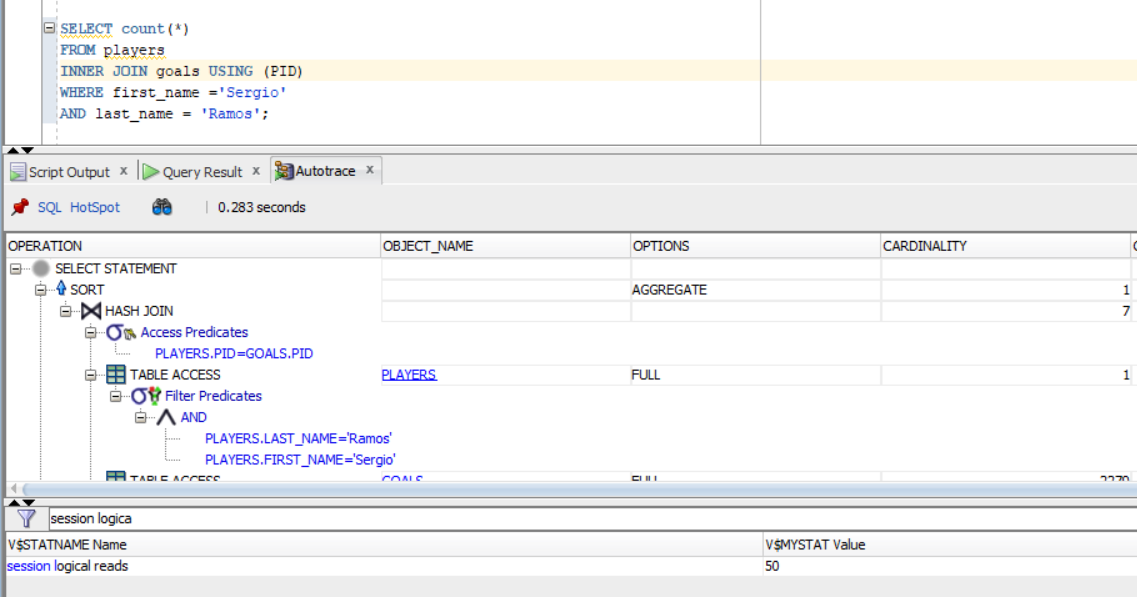
PARTITION Athletic VALUES ('Athletic'),

PARTITION Other VALUES (DEFAULT)

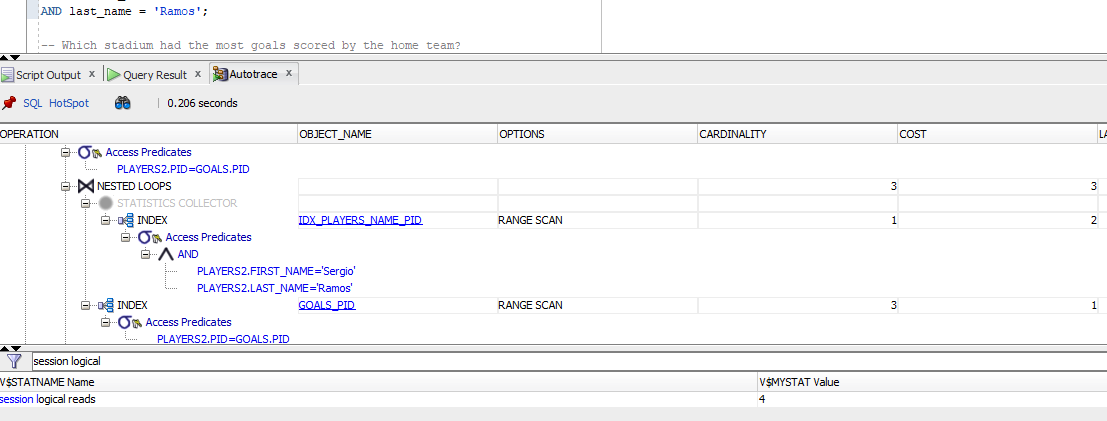
);

The partitioning in this following query did not help to reduce or optimize the query. This is because the list partitioning is less effective than range partitions and it can be more difficult to optimize for queries that use multiple predicates. Because the database has to check each partition to see if it contains the data that is needed. However, this is a good example to show that not every partitioning is a good fit for every query and it will not optimize every query

Before performance tuning:



After performance tuning:



Explanation:

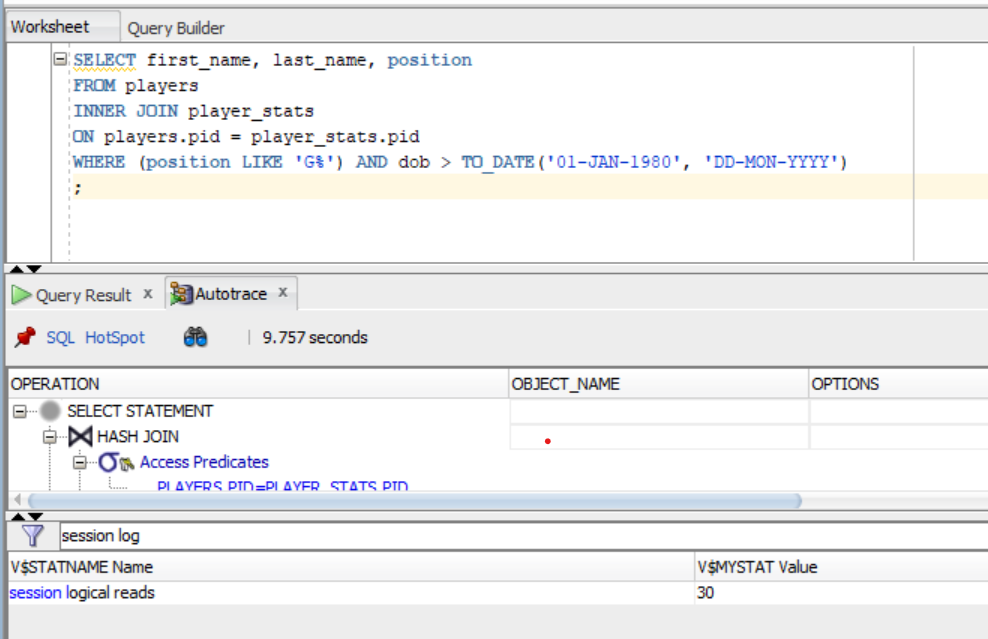
We created 2 indexes for the query. As you see in the screenshot above, before indexing we had 50 in the SLR and adding 2 indexes helped us to reduce it to 8:

CREATE INDEX idx\_players\_name\_pid ON players(first\_name, last\_name, pid);

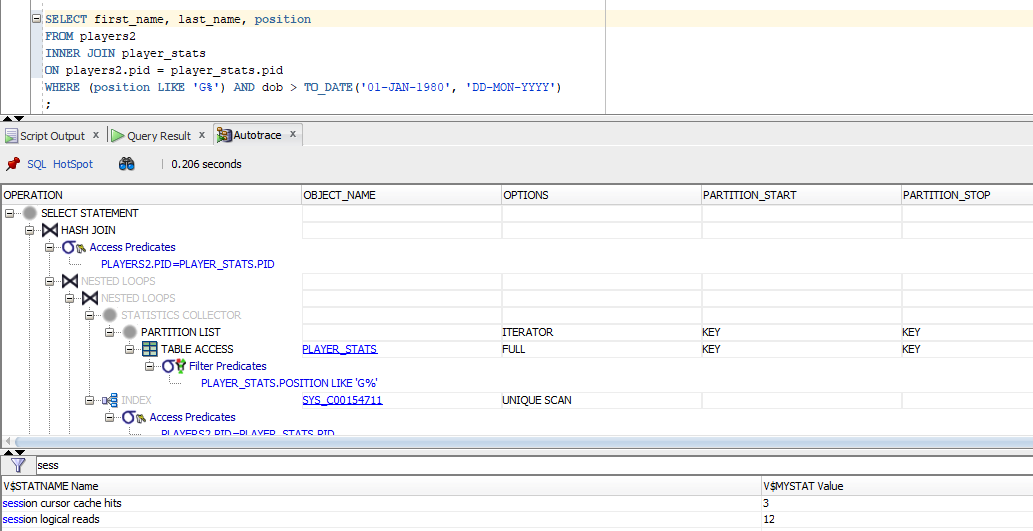
CREATE INDEX goals\_pid ON goals(pid);

Statistical collector of partitioning was also used here, which reduced the session logical reads to 4.

1. Before performance tuning:



After performance tuning:

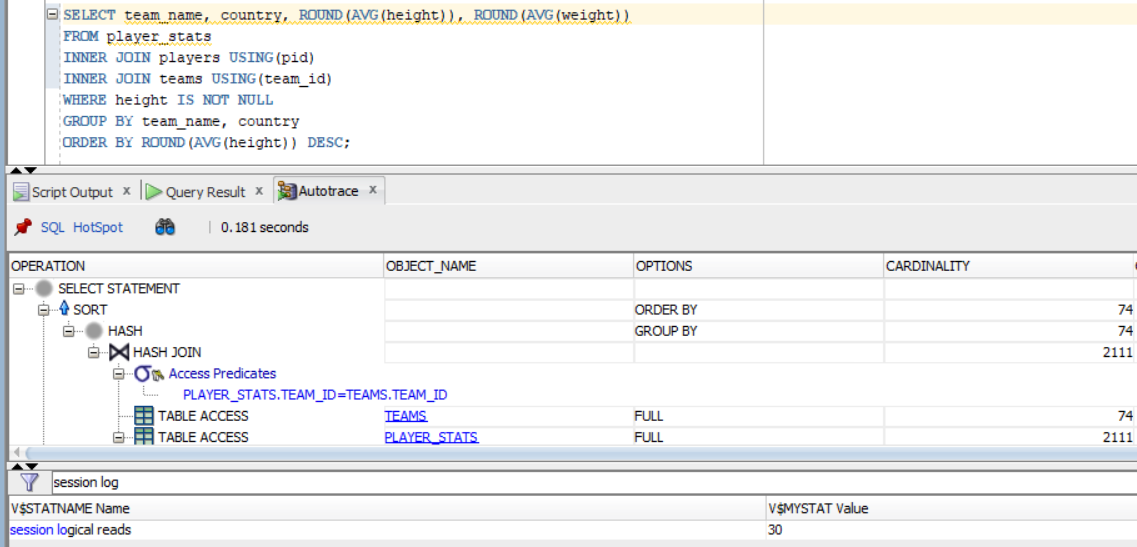


Explanation:

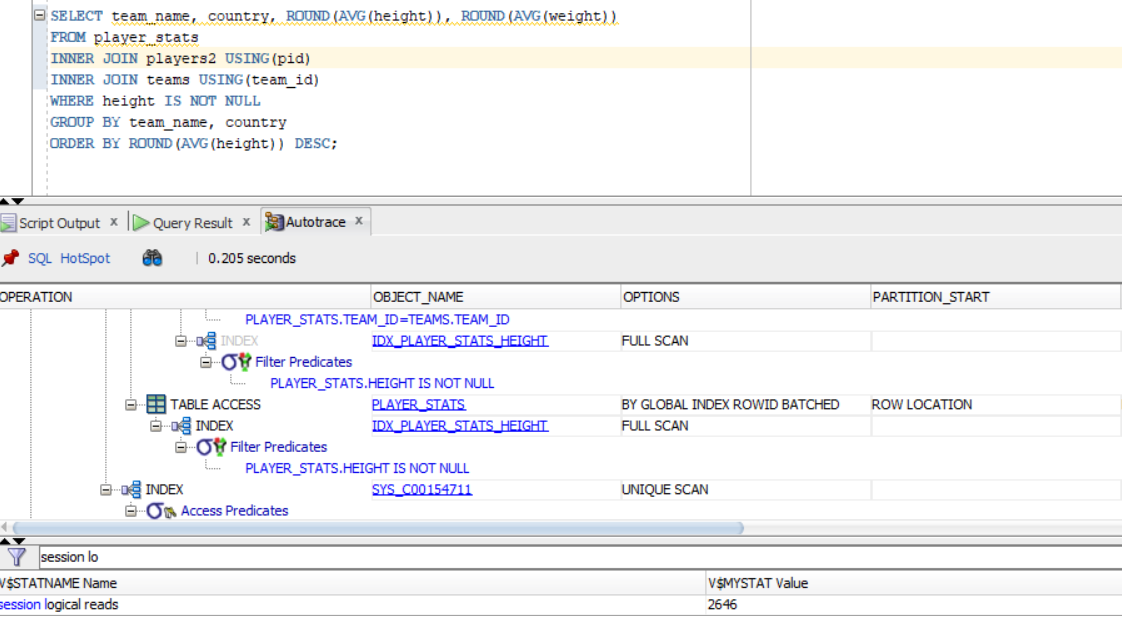
No index needed in this query since this is a specifically targeted query. We could potentially add an index on home\_team if the data size was larger, however, for our dataset it would be too much.

They are using a partitioning list here where they grouped positions into small subgroups. This partitioning strategy is coupled with parallel processing, where each list or category is handled independently by its own processing unit, known as an iterator.

Before performance tuning:



After performance tuning:



Explanation:

This is another example of the query that is not benefiting from the performance tuning. You can see all the indexes that were accessed on the screenshot above, and the partitioning in both tables of player\_stats and players2 are overwhelming the system. To get the most benefit from partitioning and indexing, we need to analyze what are those performance tuning factors doing to the tables and which queries can benefit the most from our suggestions. We wanted to show you that some queries that are taking advantage of those IP (indexing and partitioning) can reduce the execution time and session logical reads, but writing any queries without thinking about how it turns out can lead to a longer processing time.

The following is the partition plan created for the players2 table (that has the original table called players)

CREATE TABLE players2 (

pid VARCHAR(10) PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

nationality VARCHAR(50),

dob DATE

)

PARTITION BY RANGE (DOB)

(

PARTITION year1970s VALUES LESS THAN (TO\_DATE('01-JAN-70', 'DD-MON-YY')),

PARTITION year1980s VALUES LESS THAN (TO\_DATE('01-JAN-80', 'DD-MON-YY')),

PARTITION year1985 VALUES LESS THAN (TO\_DATE('01-JAN-85', 'DD-MON-YY')),

PARTITION year1990 VALUES LESS THAN (TO\_DATE('01-JAN-90', 'DD-MON-YY')),

PARTITION year1995 VALUES LESS THAN (TO\_DATE('01-JAN-95', 'DD-MON-YY')),

PARTITION year2000 VALUES LESS THAN (TO\_DATE('01-JAN-00', 'DD-MON-YY')),

PARTITION year2010s VALUES LESS THAN (MAXVALUE)

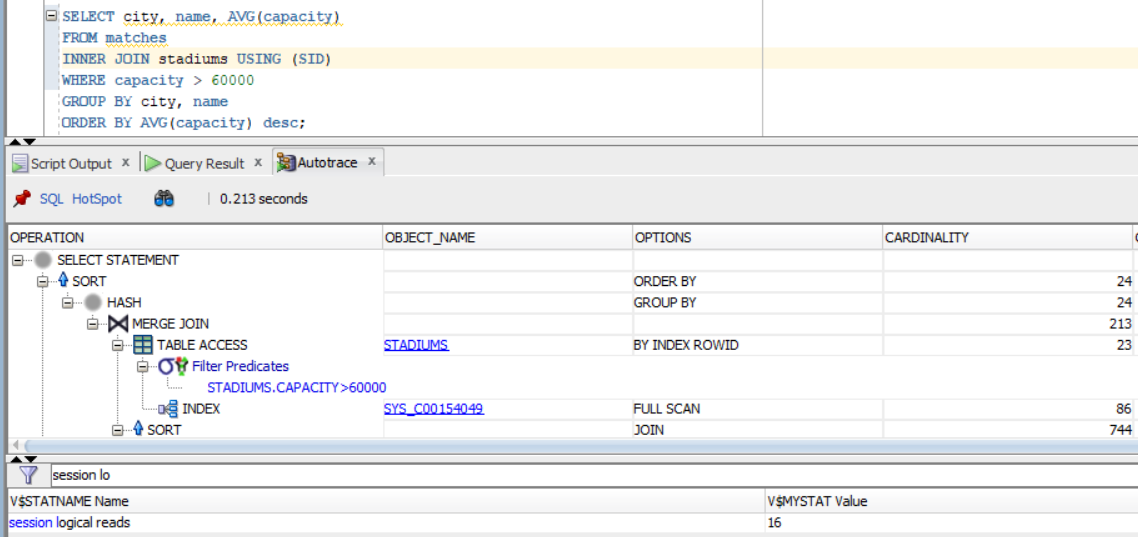
)

);

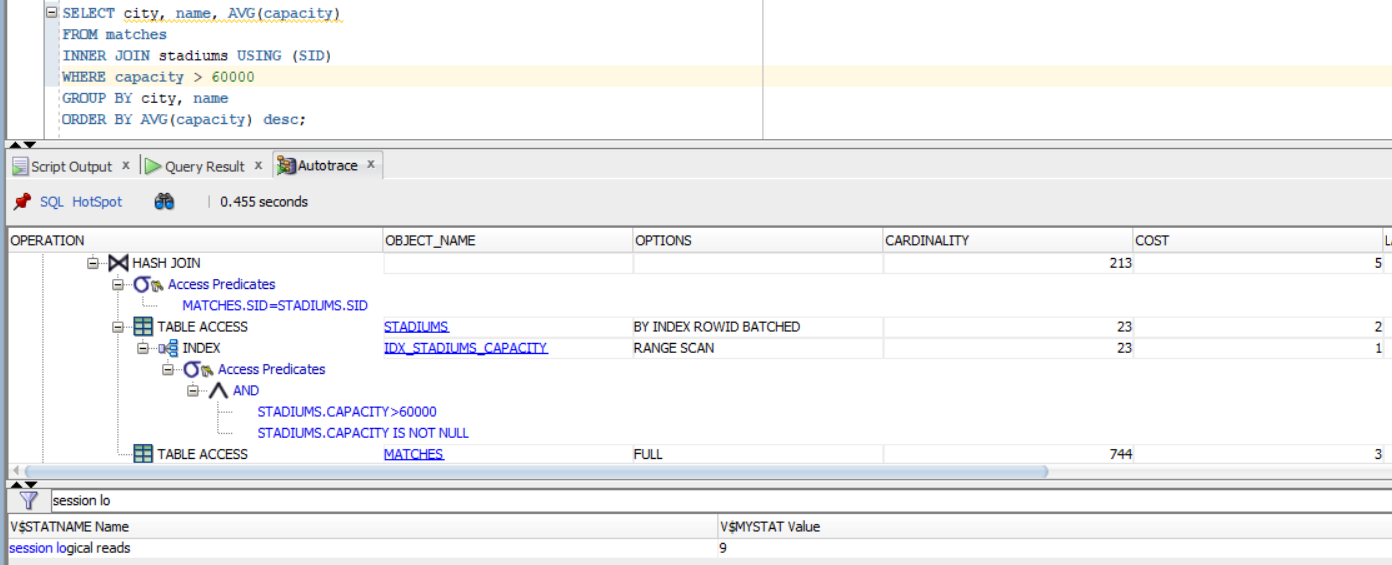
Insert into players2

Select \* From players;

Before performance tuning:



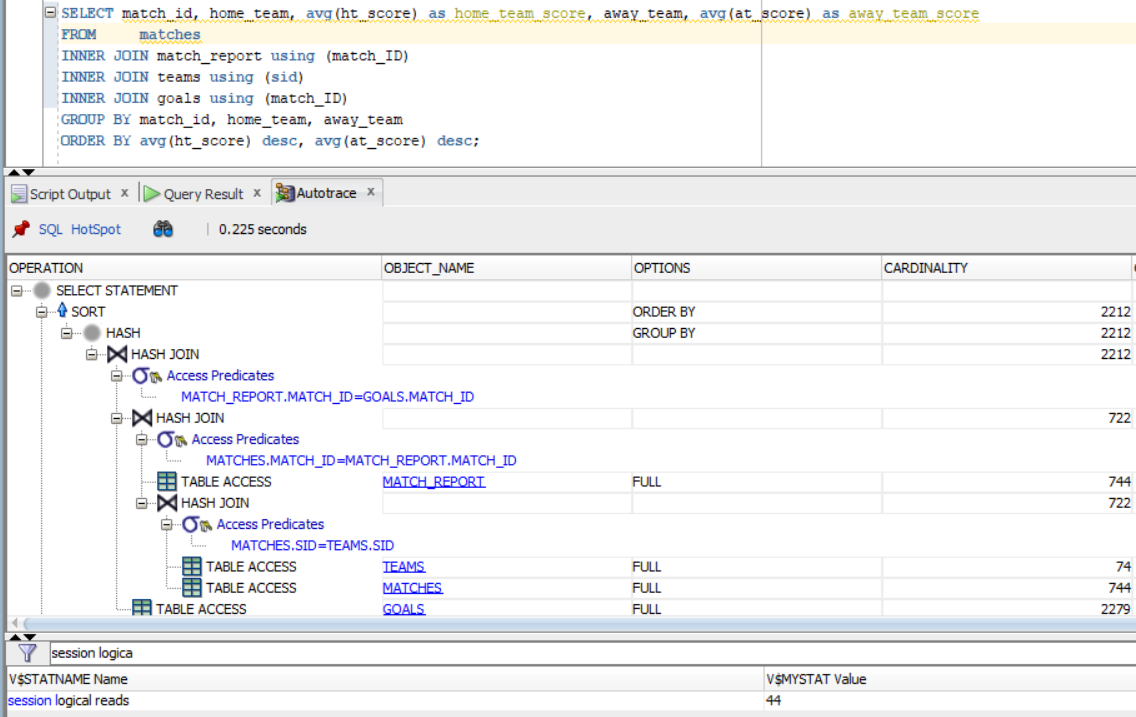
After performance tuning:



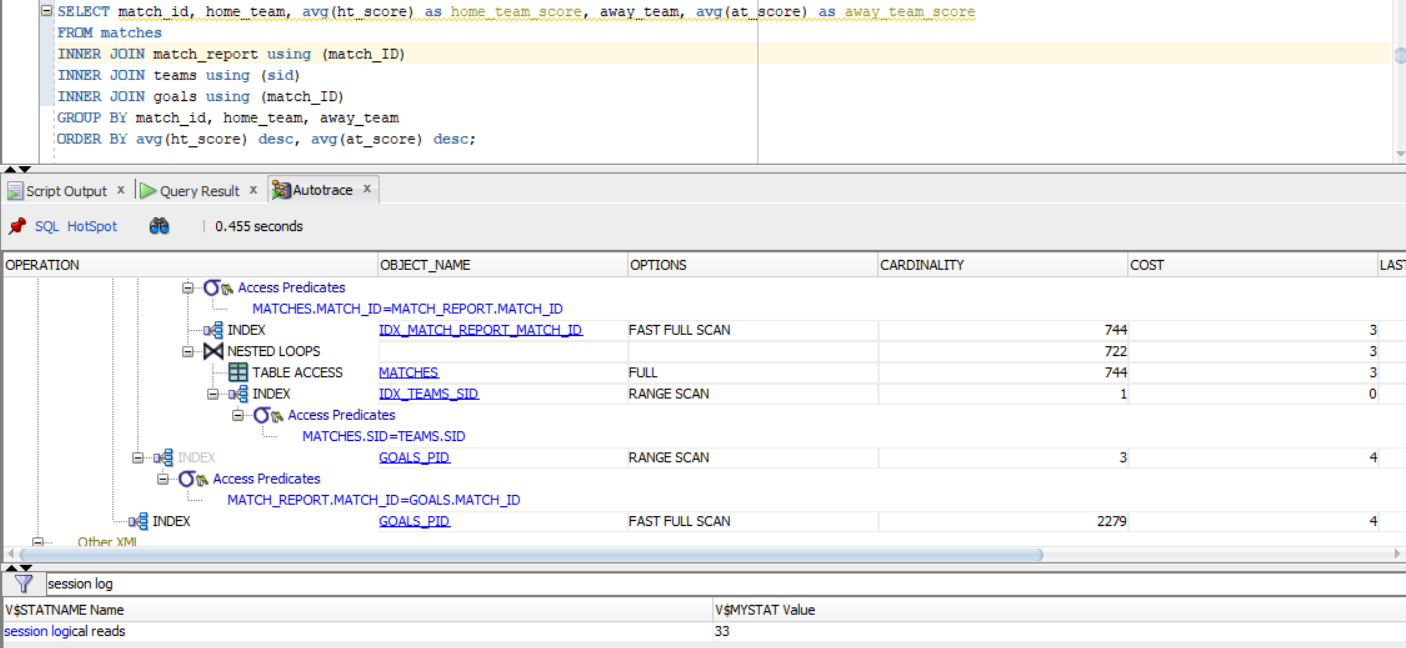
Explanation:

We created an index on the city in the stadiums table that is looking in the descending order that is a function-based normal type of index. We also have an index that is looking through each stadium and capacity combined together, that looks at the capacity first and then scans the stadium. We did not include any partitioning here because it could take more time in executing as we saw in the previous example. After assessing the capabilities of this query, we chose not to include any other performance tuning.

Before performance tuning:



After performance tuning:



Explanation:

We created a few indexes that was used here to optimize the retrieval of this query. We were able to reduce the session logical reads by 11. The following indexes were created to accomplish it:

CREATE INDEX idx\_matches\_match\_id ON matches(home\_team, away\_team,match\_id);

CREATE INDEX match\_report\_id ON match\_report(match\_id);

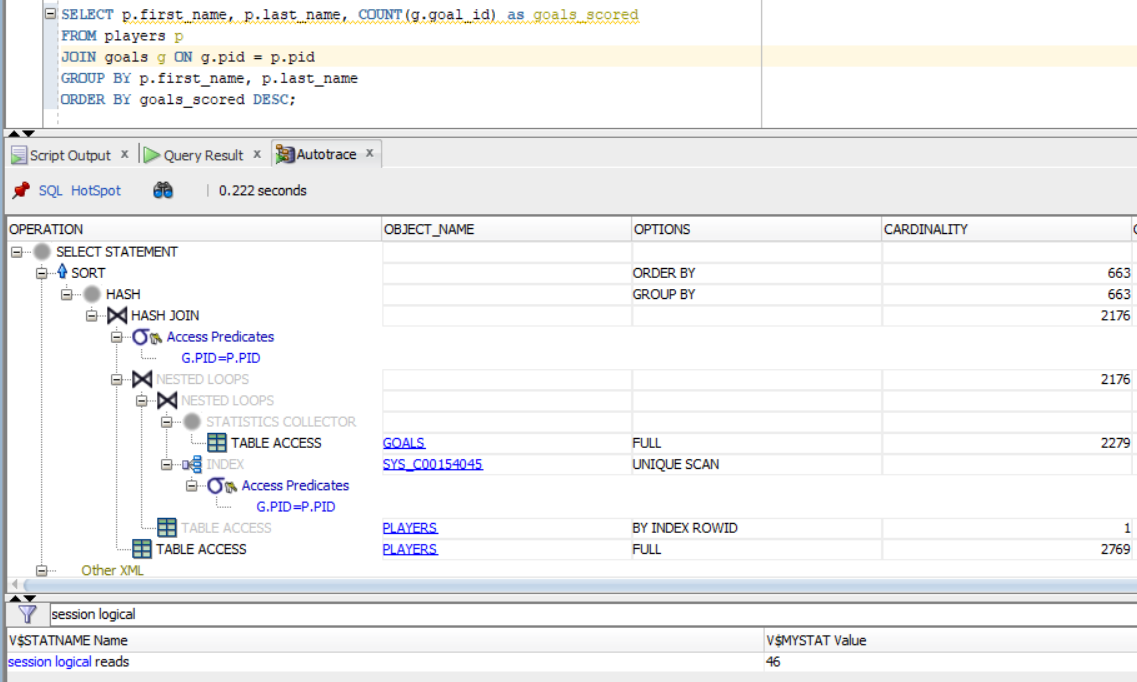
CREATE INDEX idx\_match\_report\_match\_id ON match\_report(ht\_score, at\_score,match\_id);

CREATE INDEX idx\_teams\_team ON teams(team\_name);

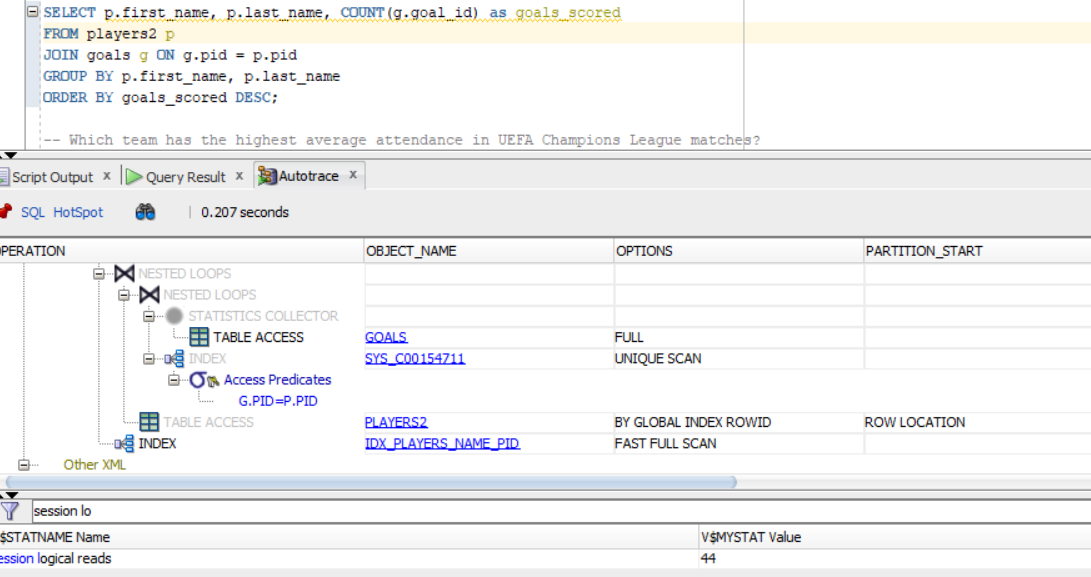
CREATE INDEX goals\_pid ON goals(match\_id, goal\_id);

No partitioning is needed in this query. By looking at autotrace, we can also see that the cardinality and cost decreased too, and more options were opted for fast full scan rather than regular full scan

Before performance tuning:



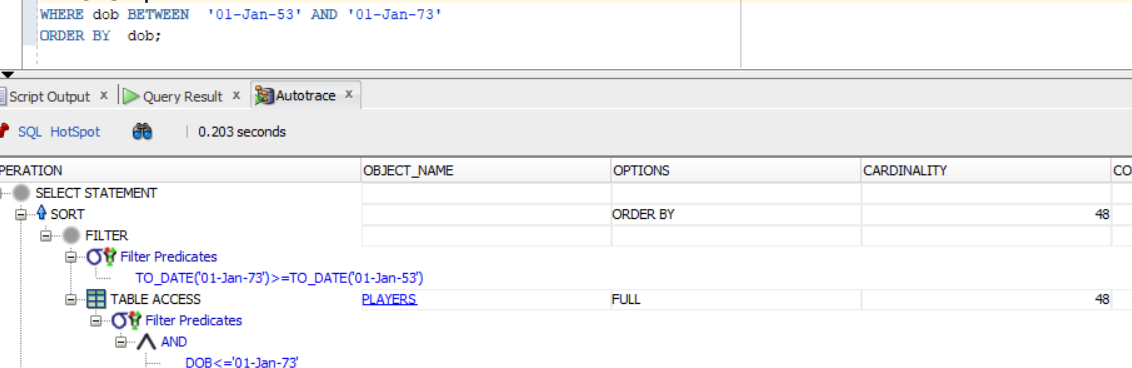
After performance tuning:



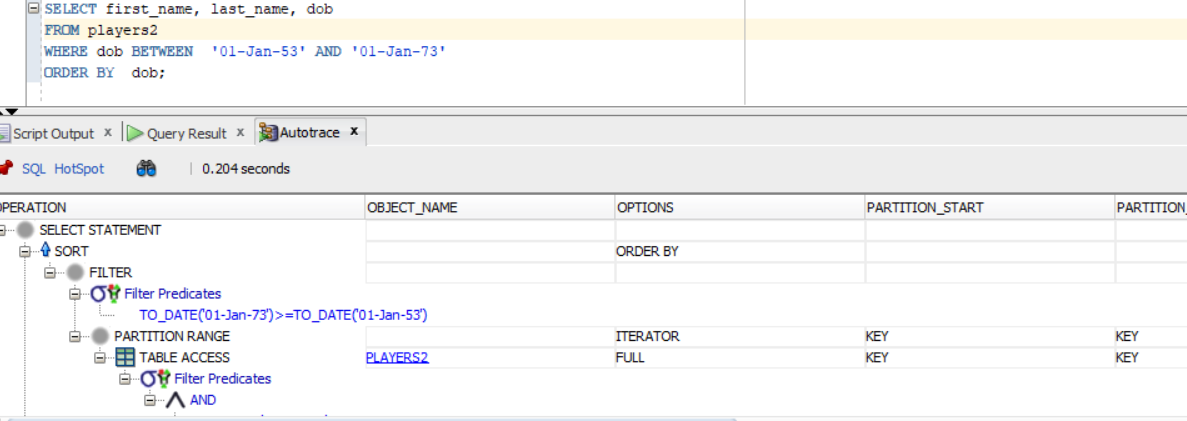
Explanation:

We created an index IDX\_PLAYERS\_NAME\_PID that searches through player’s first\_name, last\_name, and pid. There could be some other performance tuning implemented to receive a lower session logical reads, however, overturning it can cause more issues with the storage. After endless tries on improving the performance for this query, we opt to keep it like this to minimize storage cost

Before performance tuning:



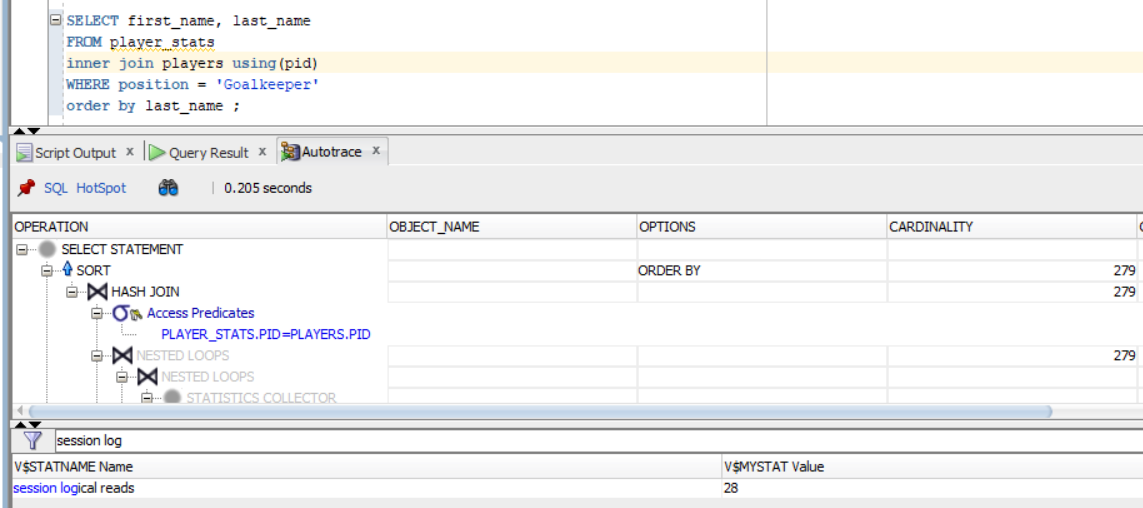
After performance tuning:



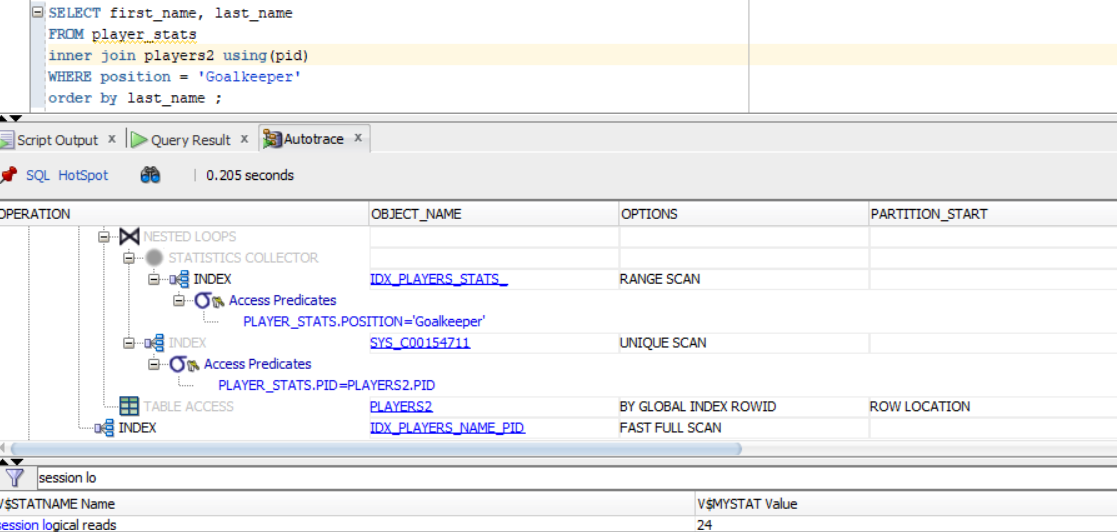
Explanation:

As you can see, in the second example they are using the partitioning range that we created for easier retrieval on the date of birth. There is no need to use an index since the query is not targeted enough. We did not copy the creation of partitioning again since we showed the plan in earlier examples

Before performance tuning:



After performance tuning:



Explanation:

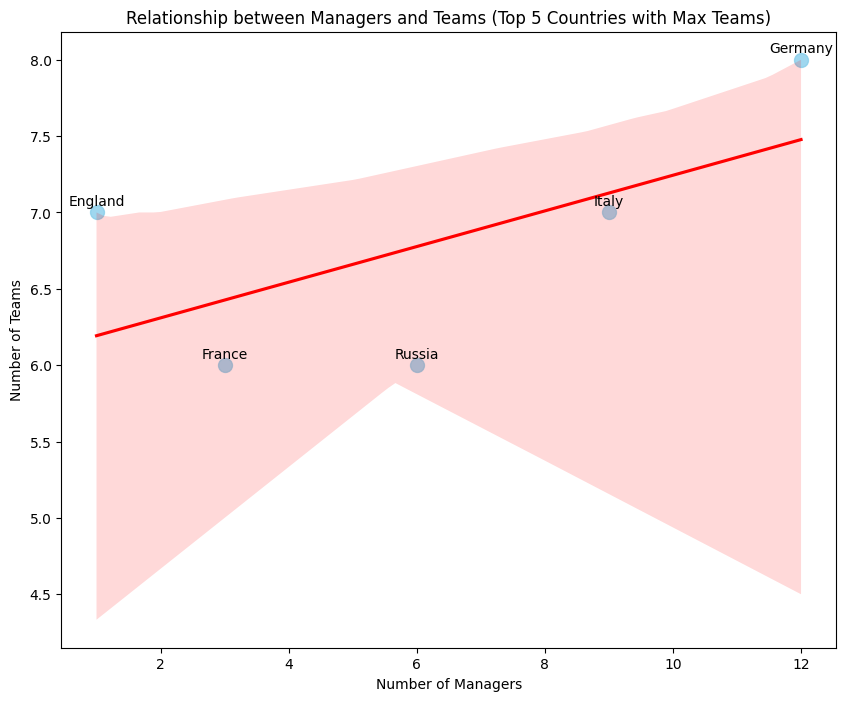
Here, we can see that the SLR decreased by 4 after implementing a few indexes. We have an index on player\_stats table that searches based on position and position+pid (2 indexes in general). And one more index on players table that looks based on the first and last name. The partitioning is also being used for the position table as it is the list partitioning.

1. **Other Topics**

Data Visualization

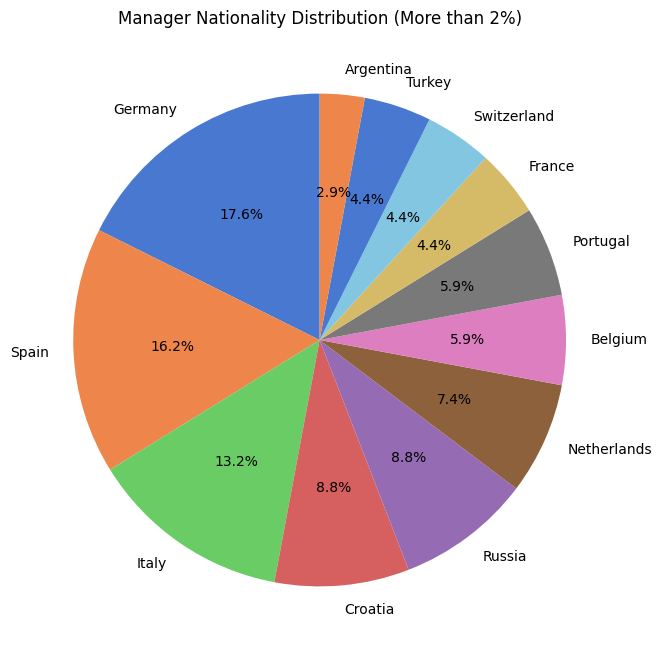
Among the given concentration choices, we chose to emphasize data visualization for this part of the assignment. This decision was made by weighing our interest in exploring diverse project aspects and the flexibility granted in the directions. Opting for data visualization aligns with the program's analytics theme, providing an opportunity to enhance our project with visual representations using tools like Python. .The following are some data visualization insights from the UEFA Champions League dataset:

### Relationship between Managers and Teams (Top 5 countries with max teams)



**Interpretation:** The scatterplot shows a positive correlation between the number of managers and the number of teams in the top 5 countries with maximum teams. This means that *as the number of managers increases, the number of teams also tends to increase*.  
**Business Value:** UEFA could invest in developing managers to train the teams in countries with a high potential for growth.

### Manager Nationality Distribution



**Interpretation**: Majority of managers come from a small subset of Western European countries like Spain, England, Italy, Germany and France. There is limited representation from Eastern Europe.

**Business value**: There is a significant representation of managers from Western European countries, with 6 of the top 10 nationalities being from this region. UEFA could focus on expanding its reach into Eastern Europe, where there is a growing interest in football.

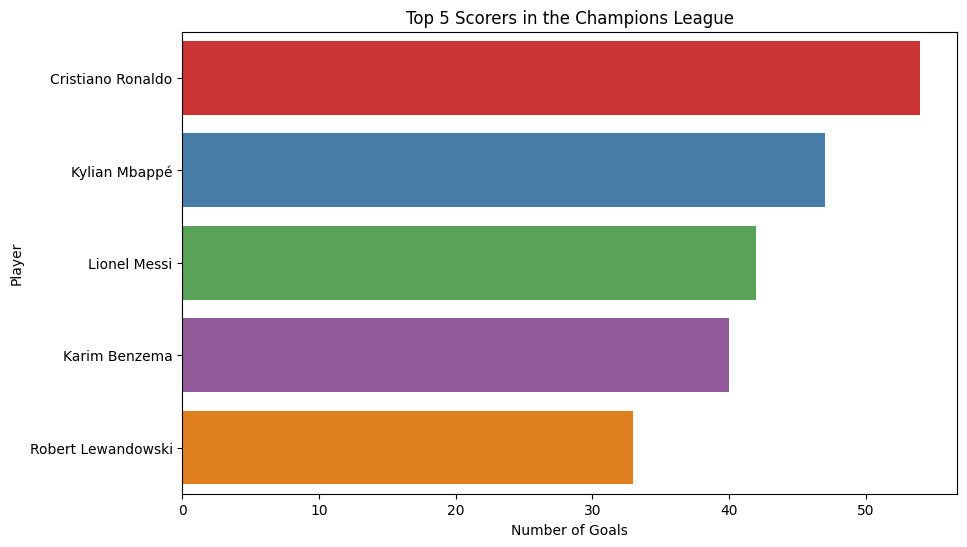
### Number of teams from each country

**Interpretation:**

The top 5 countries such as Germany, Italy, England, France and Russia account for almost 50% of all teams, revealing the concentration of football clubs in a few nations.

**Business Value:** Outside the top 15 countries, most nations have only 1-2 teams, representing an opportunity for grassroots growth.

### Top 5 scorers in the Champions League



**Interpretation:** Strikers like Lewandowski and Ronaldo are the top scorers, highlighting the impact of specialized forward positions. Left-footed players like Messi and Salah feature prominently, suggesting left-foot dominance in scoring goals.

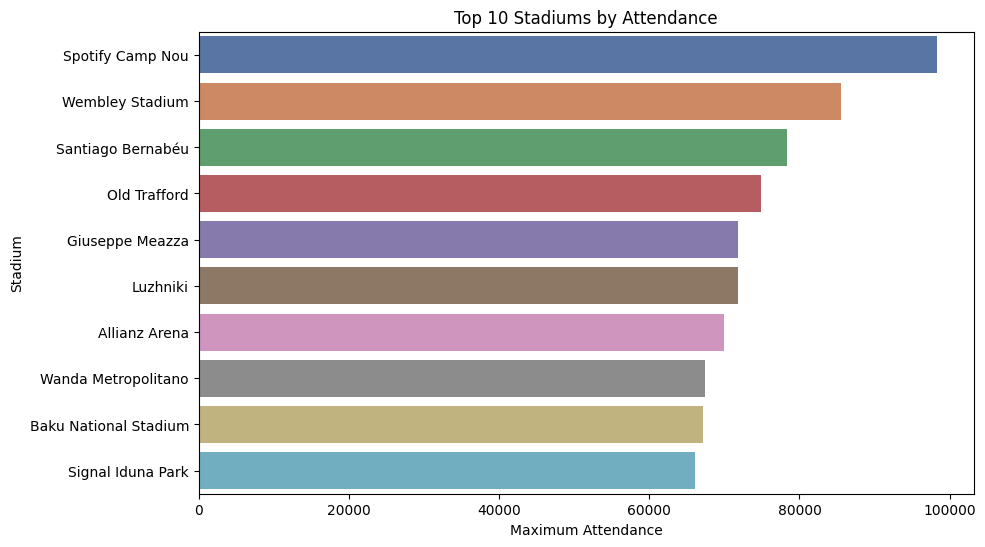
**Business Value**: Training regimes could focus more on developing ambidextrous scoring skills.

### Average Height of All Players: 182.39 cm

**Interpretation:** The average height of 182 cm is on the taller side for general populations but typical for football professionals competing aerially.

**Business Value:** Height could be used by scouts as a simple predictive filter while identifying youth talent. Also, tall players standing out physically may make them more marketable individuals. For example, Cristiano Ronaldo’s height is 187 cm which is considerably higher than the average height.

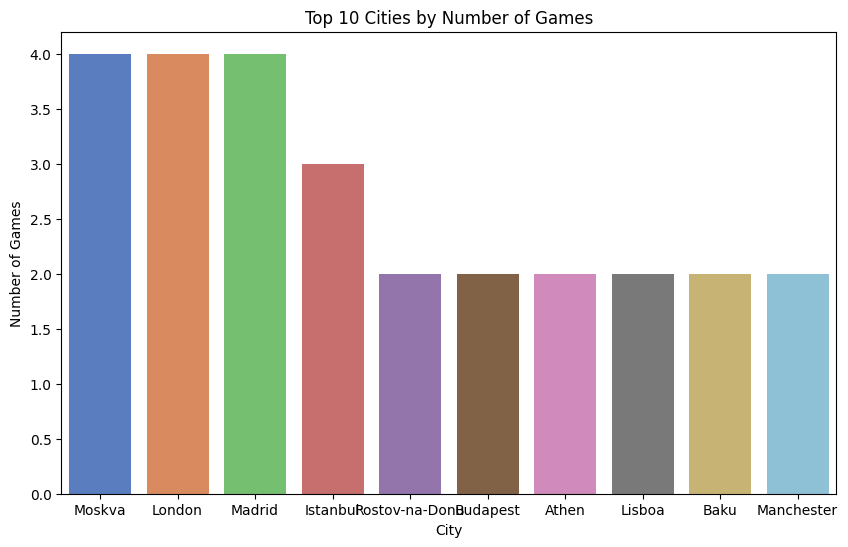
### Top 10 stadiums by attendance



**Interpretation:** The top 10 stadiums by attendance are all located in Europe.

**Business Value**: The top 10 stadiums by attendance are all high-profile stadiums with a large audience reach. Businesses could sponsor these stadiums to gain exposure to a large number of potential customers.

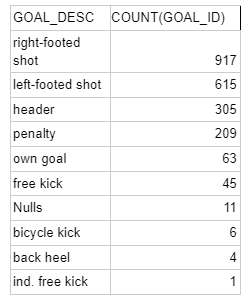
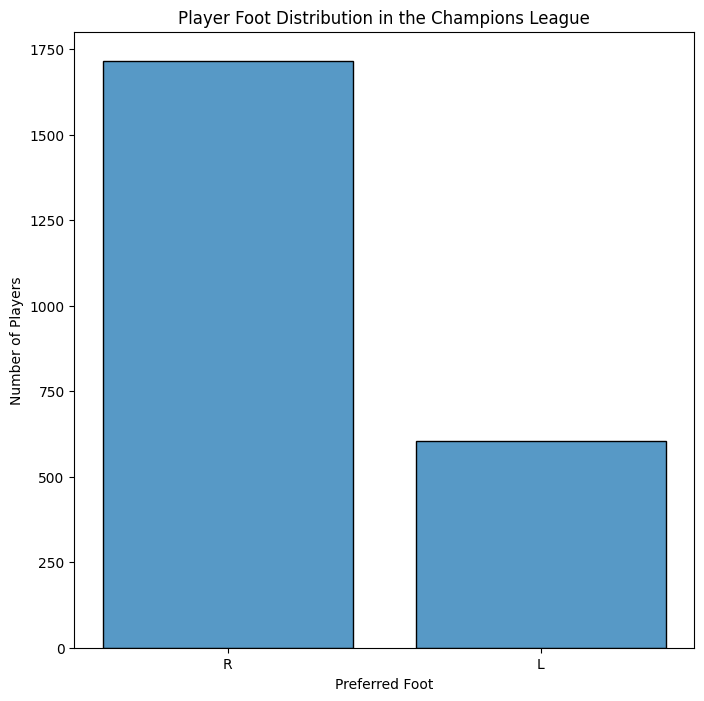
1. Top 10 cities by number of games



**Interpretation:** The top 10 cities by number of games are all located in Europe.

**Business Value:** The top 10 cities by number of games are all high-profile cities with a large audience reach. Businesses could sponsor these cities to gain exposure to a large number of potential customers.

1. Foot distributions and impact on the game



| **PID** | **FIRST\_NAME** | **LAST\_NAME** | **FOOT** | **GOALCOUNT** | **LEFTFOOTGOALS** | **RIGHTFOOTGOALS** | **OTHERFOOTGOALS** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ply398 | Robert | Lewandowski | R | 54 | 9 | 19 | 26 |
| ply699 | Cristiano | Ronaldo | Null | 47 | 5 | 22 | 19 |
| ply740 | Lionel | Messi | L | 42 | 26 | 4 | 12 |
| ply831 | Karim | Benzema | R | 40 | 3 | 21 | 16 |
| ply590 | Mo | Salah | L | 33 | 21 | 7 | 5 |

**Interpretation:** : There are 1715 right footed players (73.92%) and 605 left footed players (26.08%). However, the following table demonstrates the goals count for soccer players with different preferences for using the body parts for scoring goals which indicates that the goals are not scored in accordance with their preferred foot. Managers could invest in training the players to become ambidextrous and leverage their potential to its full capacity.

**Business value:** Businesses could develop products and services that are specifically designed for right-footed or left-footed players. For example, a shoe company could develop a line of soccer shoes that are specifically designed for right-footed or left-footed players.

Data Mining

We are also interested in writing complex queries that help us gain valuable information and retrieve business insights from the database. We could leverage this information for conducting further research.

1. **Which team has the highest average attendance in UEFA Champions League matches?**

SELECT t.team\_name, ROUND(AVG(mr.attendance),1) AS avg\_attendance

FROM teams t

JOIN matches m ON t.team\_id = m.home\_team OR t.team\_id = m.away\_team

JOIN tournaments t ON m.tour\_id = t.tour\_id

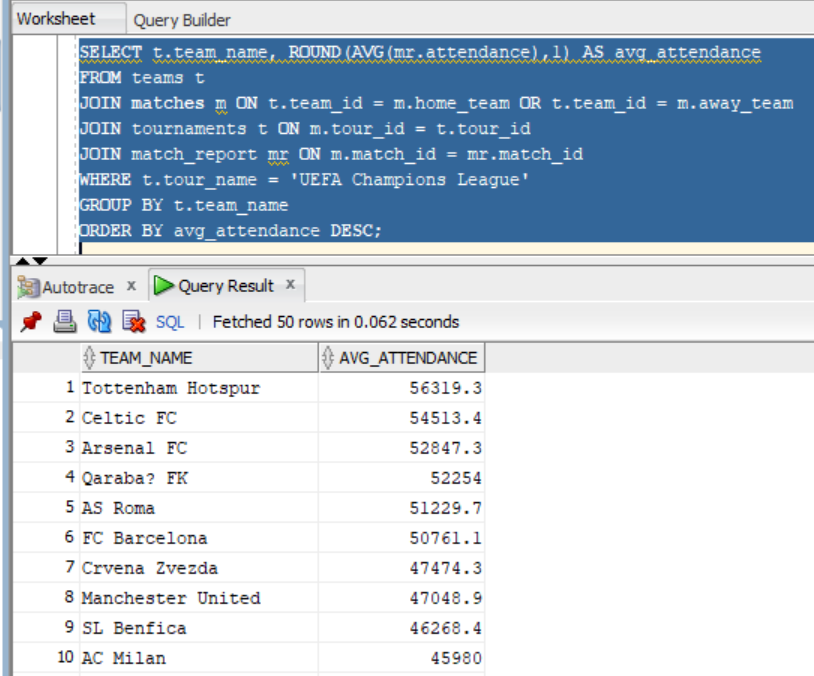
JOIN match\_report mr ON m.match\_id = mr.match\_id

WHERE t.tour\_name = 'UEFA Champions League'

GROUP BY t.team\_name

ORDER BY avg\_attendance DESC;

Result:



1. **Which manager has the highest win percentage in UEFA Champions League matches?**

SELECT m.last\_name, m.first\_name, ROUND(AVG(CASE WHEN mr.ht\_score > mr.at\_score THEN 1 ELSE 0 END),4) AS win\_pct

FROM managers m

JOIN teams t ON m.team\_id = t.team\_id

JOIN matches m ON t.team\_id = m.home\_team OR t.team\_id = m.away\_team

JOIN tournaments t ON m.tour\_id = t.tour\_id

JOIN match\_report mr ON m.match\_id = mr.match\_id

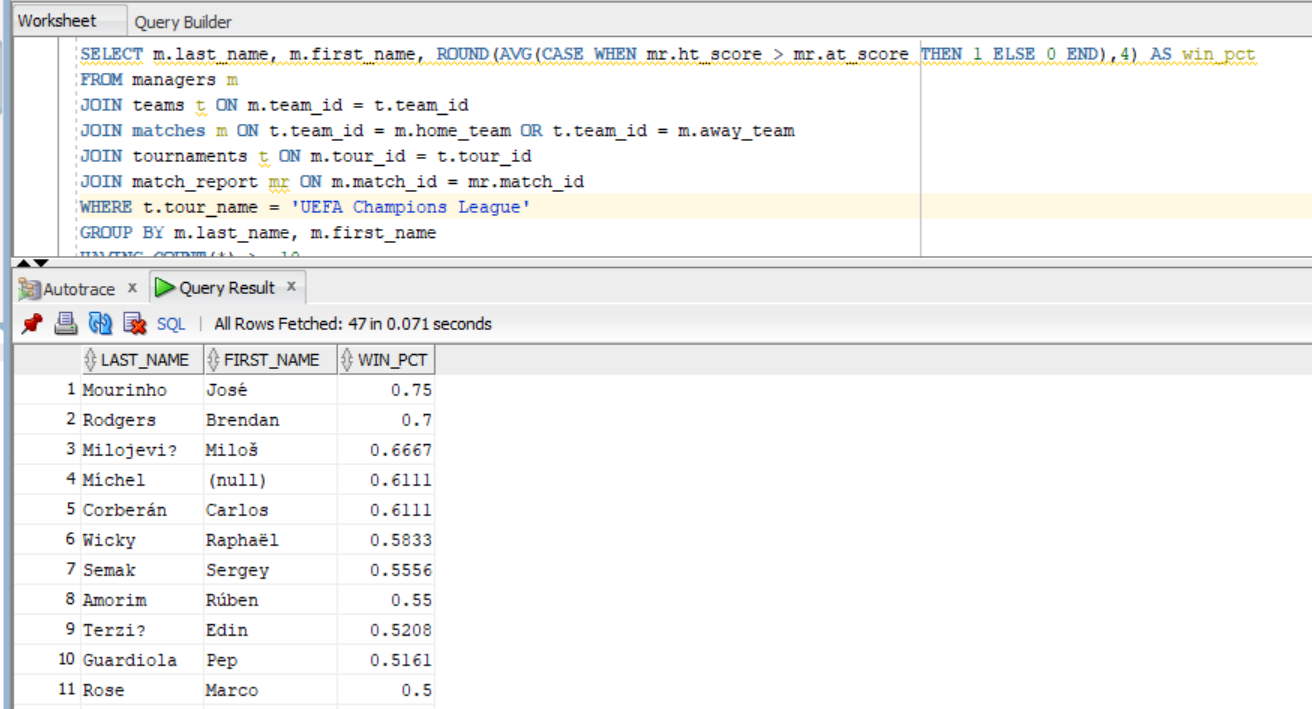
WHERE t.tour\_name = 'UEFA Champions League'

GROUP BY m.last\_name, m.first\_name

HAVING COUNT(\*) >= 10

ORDER BY win\_pct DESC;

Result:



1. **Which player has the highest number of assists in UEFA Champions League matches?**

SELECT p.first\_name, p.last\_name, COUNT(\*) AS assists

FROM players p

JOIN goals g ON p.pid = g.assist

JOIN matches m ON g.match\_id = m.match\_id

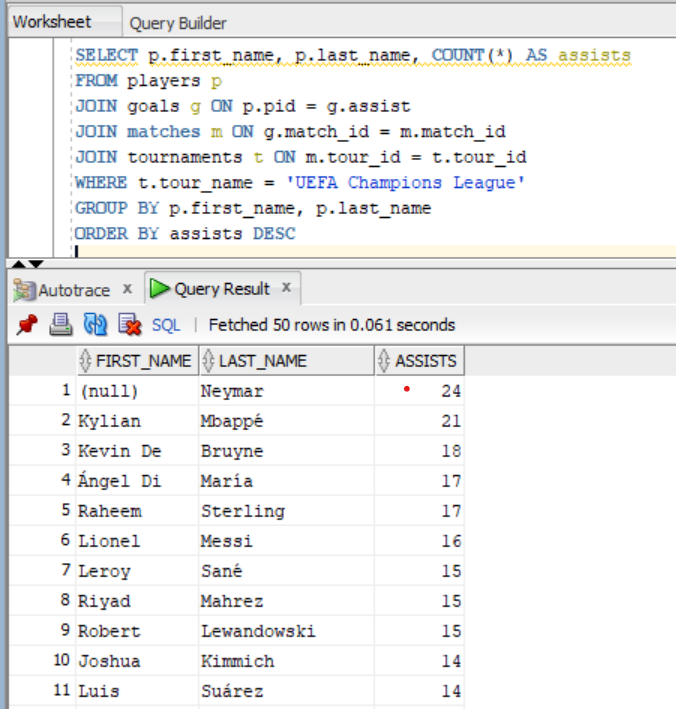
JOIN tournaments t ON m.tour\_id = t.tour\_id

WHERE t.tour\_name = 'UEFA Champions League'

GROUP BY p.first\_name, p.last\_name

ORDER BY assists DESC

Result:



1. **Which stadium has hosted the most UEFA Champions League finals?**

SELECT s.name, COUNT(\*) AS finals\_hosted

FROM stadiums s

JOIN matches m ON s.sid = m.sid

JOIN tournaments t ON m.tour\_id = t.tour\_id

JOIN match\_report mr ON m.match\_id = mr.match\_id

WHERE t.tour\_name = 'UEFA Champions League'

AND EXISTS (

SELECT 1

FROM matches m2

JOIN tournaments t2 ON m2.tour\_id = t2.tour\_id

WHERE m2.tour\_id = m.tour\_id

AND m2.match\_id != m.match\_id

AND t2.tour\_name = 'UEFA Champions League'

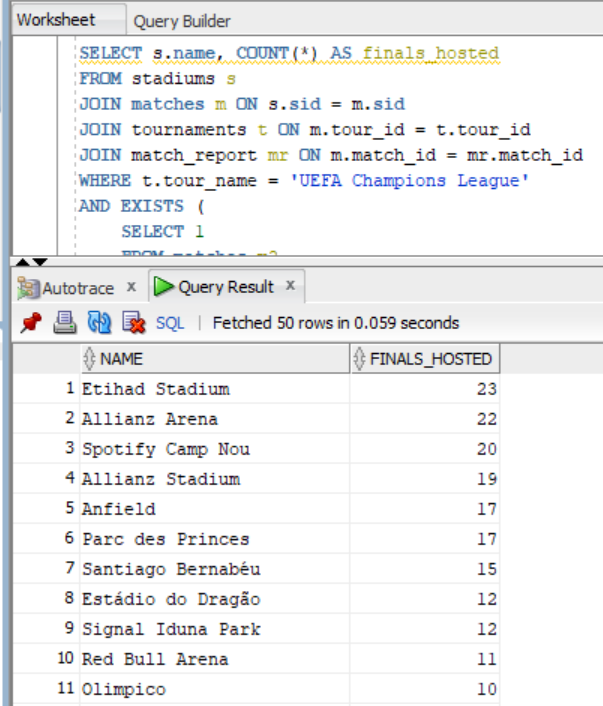
AND mr.ht\_score > mr.at\_score

)

GROUP BY s.name

ORDER BY finals\_hosted DESC;

**Result**:



1. **Which position has the highest average goal-scoring rate in UEFA Champions League matches?**

SELECT ps.position, ROUND(AVG(g.duration),1) AS avg\_goal\_scoring\_rate

FROM player\_stats ps

JOIN players p ON ps.pid = p.pid

JOIN goals g ON p.pid = g.pid

JOIN matches m ON g.match\_id = m.match\_id

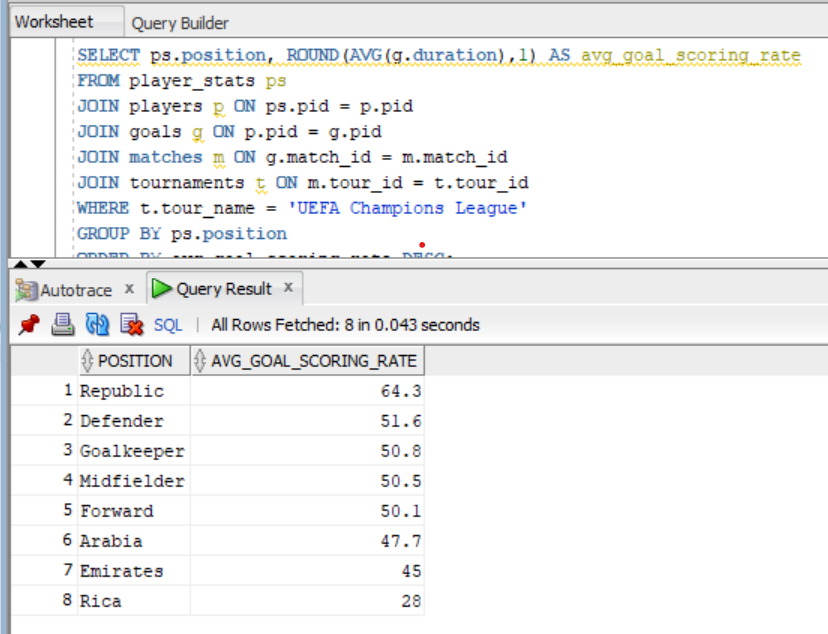
JOIN tournaments t ON m.tour\_id = t.tour\_id

WHERE t.tour\_name = 'UEFA Champions League'

GROUP BY ps.position

ORDER BY avg\_goal\_scoring\_rate DESC;

**Result:**



1. **Which team has the highest average number of goals conceded per match in UEFA Champions League matches?**

SELECT t.team\_name, ROUND(AVG(mr.at\_score),2) AS avg\_goals\_conceded

FROM teams t

JOIN matches m ON t.team\_id = m.home\_team OR t.team\_id = m.away\_team

JOIN tournaments t ON m.tour\_id = t.tour\_id

JOIN match\_report mr ON m.match\_id = mr.match\_id

WHERE t.tour\_name = 'UEFA Champions League'

GROUP BY t.team\_name

ORDER BY avg\_goals\_conceded DESC;

**Result**:

