

Window Functions

DATA MANIPULATION IN SQL



Mona Khalil

Data Scientist, Greenhouse Software

Working with aggregate values

- Requires you to use `GROUP BY` with **all** non-aggregate columns

```
SELECT
  country_id,
  season,
  date,
  AVG(home_goal) AS avg_home
FROM match
GROUP BY country_id;
```

```
ERROR: column "match.season" must appear in the GROUP BY
clause or be used in an aggregate function
```

Introducing window functions!

- Perform calculations on an already generated result set (a *window*)
- Aggregate calculations
 - Similar to subqueries in `SELECT`
 - Running totals, rankings, moving averages

What's a window function?

- *How many goals were scored in each match in 2011/2012, and how did that compare to the average?*

```
SELECT
  date,
  (home_goal + away_goal) AS goals,
  (SELECT AVG(home_goal + away_goal)
   FROM match
   WHERE season = '2011/2012') AS overall_avg
FROM match
WHERE season = '2011/2012';
```

date	goals	overall_avg
2011-07-29	3	2.71646
2011-07-30	2	2.71646
2011-07-30	4	2.71646
2011-07-30	1	2.71646

What's a window function?

- *How many goals were scored in each match in 2011/2012, and how did that compare to the average?*

```
SELECT
  date,
  (home_goal + away_goal) AS goals,
  AVG(home_goal + away_goal) OVER() AS overall_avg
FROM match
WHERE season = '2011/2012';
```

date	goals	overall_avg
2011-07-29	3	2.71646
2011-07-30	2	2.71646
2011-07-30	4	2.71646
2011-07-30	1	2.71646

Generate a RANK

- *What is the rank of matches based on number of goals scored?*

```
SELECT
    date,
    (home_goal + away_goal) AS goals
FROM match
WHERE season = '2011/2012';
```

date	goals
2011-07-29	3
2011-07-30	2
2011-07-30	4
2011-07-30	1

Generate a RANK

- *What is the rank of matches based on number of goals scored?*

```
SELECT
    date,
    (home_goal + away_goal) AS goals,
    RANK() OVER(ORDER BY home_goal + away_goal) AS goals_rank
FROM match
WHERE season = '2011/2012';
```

date	goals	goals_rank	
-----	-----	-----	
2012-04-28	0	1	
2011-12-26	0	1	
2011-09-10	0	1	
2011-08-27	0	1	

Generate a RANK

- *What is the rank of matches based on number of goals scored?*

```
SELECT
    date,
    (home_goal + away_goal) AS goals,
    RANK() OVER(ORDER BY home_goal + away_goal DESC) AS goals_rank
FROM match
WHERE season = '2011/2012';
```

date	goals	goals_rank	
-----	-----	-----	
2011-11-06	10	1	
2011-08-28	10	1	
2012-05-12	9	3	
2012-02-12	9	3	

Key differences

- Processed *after* every part of query except `ORDER BY`
 - Uses information in result set rather than database
- Available in PostgreSQL, Oracle, MySQL, SQL Server...
 - ...but NOT SQLite

Let's practice!

DATA MANIPULATION IN SQL

Window Partitions

DATA MANIPULATION IN SQL



Mona Khalil

Data Scientist, Greenhouse Software

OVER and PARTITION BY

- Calculate separate values for different categories
- Calculate *different* calculations in the same column

```
AVG(home_goal) OVER(PARTITION BY season)
```

Partition your data

- *How many goals were scored in each match, and how did that compare to the overall average?*

```
SELECT
  date,
  (home_goal + away_goal) AS goals,
  AVG(home_goal + away_goal) OVER() AS overall_avg
FROM match;
```

date	goals	overall_avg
2011-12-17	3	2.73210
2012-05-01	2	2.73210
2012-11-27	4	2.73210
2013-04-20	1	2.73210
2013-11-09	5	2.73210

Partition your data

- *How many goals were scored in each match, and how did that compare to the season's average?*

```
SELECT
  date,
  (home_goal + away_goal) AS goals,
  AVG(home_goal + away_goal) OVER(PARTITION BY season) AS season_avg
FROM match;
```

date	goals	season_avg
2011-12-17	3	2.71646
2012-05-01	2	2.71646
2012-11-27	4	2.77270
2013-04-20	1	2.77270
2013-11-09	5	2.76682

PARTITION by Multiple Columns

```
SELECT
  c.name,
  m.season,
  (home_goal + away_goal) AS goals,
  AVG(home_goal + away_goal)
    OVER(PARTITION BY m.season, c.name) AS season_ctry_avg
FROM country AS c
LEFT JOIN match AS m
ON c.id = m.country_id
```

name	season	goals	season_ctry_avg
Belgium	2011/2012	1	2.88
Netherlands	2014/2015	1	3.08
Belgium	2011/2012	1	2.88
Spain	2014/2015	2	2.66

PARTITION BY considerations

- Can partition data by 1 or more columns
- Can partition aggregate calculations, ranks, etc

Let's practice!

DATA MANIPULATION IN SQL

Sliding windows

DATA MANIPULATION IN SQL



Mona Khalil

Data Scientist, Greenhouse Software

Sliding windows

- Perform calculations relative to the current row
- Can be used to calculate running totals, sums, averages, etc
- Can be partitioned by one or more columns

Sliding window keywords

ROWS BETWEEN <start> **AND** <finish>

PRECEDING

FOLLOWING

UNBOUNDED PRECEDING

UNBOUNDED FOLLOWING

CURRENT ROW

Sliding window example

```
-- Manchester City Home Games
SELECT
  date,
  home_goal,
  away_goal,
  SUM(home_goal)
    OVER(ORDER BY date ROWS BETWEEN
          UNBOUNDED PRECEDING AND CURRENT ROW) AS running_total
FROM match
WHERE hometeam_id = 8456 AND season = '2011/2012';
```

date	home_goal	away_goal	running_total	
-----	-----	-----	-----	
2011-08-15	4	0	4	
2011-09-10	3	0	7	
2011-09-24	2	0	9	
2011-10-15	4	1	13	

Sliding window frame

```
-- Manchester City Home Games
SELECT date,
       home_goal,
       away_goal,
       SUM(home_goal)
         OVER(ORDER BY date
              ROWS BETWEEN 1 PRECEDING
              AND CURRENT ROW) AS last2
FROM match
WHERE hometeam_id = 8456
      AND season = '2011/2012';
```

date	home_goal	away_goal	last2
2011-08-15	4	0	4
2011-09-10	3	0	7
2011-09-24	2	0	5
2011-10-15	4	1	6

Let's practice!

DATA MANIPULATION IN SQL

Bringing it all Together

DATA MANIPULATION IN SQL



Mona Khalil

Data Scientist, Greenhouse Software

What you've learned so far

- `CASE` statements
- Simple subqueries
- Nested and correlated subqueries
- Common table expressions
- Window functions

Let's do a case study!

Who defeated Manchester United in the 2013/2014 season?



Steps to construct the query

- Get team names with CTEs
- Get match outcome with `CASE` statements
- Determine how badly they lost with a window function



Getting the database for yourself

Full European Soccer Database

Let's practice!

DATA MANIPULATION IN SQL