INTERMEDIATE SQL

Correlated Subqueries

Mona Khalil
Curriculum Lead, DataCamp

Correlated vs Simple Subquery

- Dependent on the main query to execute
- Evaluated in loops
- Slower query performance

Correlated Subqueries

```
SELECT
    s.stage,
    ROUND(s.avg_goals,2) AS avg_goals
FROM
    (SELECT stage, AVG(home_goal + away_goal) AS avg_goals
     FROM match
     WHERE season = '2012/2013'
     GROUP BY stage) AS s
WHERE s.avg_goals > (SELECT AVG(home_goal + away_goal)
                     FROM match AS m WHERE m.stage != s.stage);
```

stage	avg_goals
3	2.83
4	2.8
6	2.78
8	3.09
10	2.96

INTERMEDIATE SQL

Let's practice!

INTERMEDIATE SQL

Nested Subqueries

Mona Khalil
Curriculum Lead, DataCamp

A subquery inside a subquery

- Advanced filtering & structuring

```
SELECT
    c.name AS country,
    AVG(m.home_goal + m.away_goal) AS avg_goals,
    AVG(m.home_goal + m.away_goal) -
        (SELECT AVG(home_goal + away_goal) FROM match) AS avg_diff
FROM country AS c
LEFT JOIN match AS m
ON c.id = m.country_id
GROUP BY c.name;
```

A subquery inside a subquery

- Advanced filtering & structuring

```
SELECT
    AVG(m.home_goal + m.away_goal) AS avg_2013,
    (SELECT AVG(home_goal + away_goal)
         FROM match WHERE id IN
              (SELECT id FROM match
               WHERE season = '2013/2014'
                     AND EXTRACT(MONTH FROM date) = 11)) AS avg_2013_nov
FROM match AS m
WHERE m.season = '2013/2014';
```

avg_2013	avg_2013_nov
2.77	2.75

Correlated nested subqueries

- Nested subqueries can be correlated or uncorrelated

```
SELECT
    c.name AS country,
    AVG(m.home_goal + m.away_goal) AS avg_goals,
    (SELECT AVG(home_goal + away_goal)
         FROM match
         WHERE id IN
              (SELECT id FROM match
               WHERE season = '2013/2014'
                     AND EXTRACT(MONTH FROM date) = 11)
                AND country_id = m.country_id) AS avg_2013_nov
FROM country AS c
LEFT JOIN match AS m
ON c.id = m.country_id
GROUP BY c.name;
```

Correlated nested subqueries

country	avg_goals	avg_2013_nov
Belgium	2.8579	NULL
England	2.7342	2.6944
France	2.502	2.4054
Germany	2.9273	3.2059
Italy	2.6593	2.0938
Netherlands	3.1708	3.1613
Poland	2.4865	2.6
Portugal	2.5663	1.8889
Scotland	2.6721	2.9444
Spain	2.7599	3.1667
Switzerland	2.7407	3.6363

Some Considerations

- Query performance
- Organize your subqueries
- Leave comments in your query

```
/* Full line comment */
SELECT a, b -- mid-line comment
```

INTERMEDIATE SQL

Let's practice!

INTERMEDIATE SQL

Common Table Expressions

Mona Khalil
Curriculum Lead, DataCamp

Common Table Expressions

Common Table Expressions (CTEs)

- Table *declared* before the main query
- Named and referenced later in `FROM` statement
- Also referred to as factored subqueries

Common Table Expression Syntax

```
WITH cte AS (
    SELECT col1, col2
    FROM table)
SELECT
    AVG(col1) AS avg_col
FROM cte;
```

CTE Benefits

- Improves overall query performance
- Improves organization of queries

INTERMEDIATE SQL

Let's Practice!

INTERMEDIATE SQL

Working with Multiple CTEs

Mona Khalil
Curriculum Lead, DataCamp

One CTE...two CTEs...

```
WITH cte1 AS (
    SELECT col1, col2
    FROM table1),

cte2 AS (
    SELECT col1, col2
    FROM table2),

cte3 AS (
    SELECT col1, col2
    FROM table3)

SELECT
    AVG(col1) AS avg_col
FROM cte1, cte2, cte3
LEFT JOIN...;
```

Benefits of multiple CTEs

- Increased organization
- Increased join capabilities
- Solve otherwise impossible problems with recursive CTEs

INTERMEDIATE SQL

Let's Practice!