Correlated Subqueries

INTERMEDIATE SQL

SQL

Mona Khalil
Data Scientist, Greenhouse Software

datacamp

Correlated subquery

- Uses values from the *outer* query to generate a result
- Re-run for every row generated in the final data set
- Used for advanced joining, filtering, and evaluating data

A simple example

- *Which match stages tend to have a higher than average number of goals scored?*

```sql
SELECT
    s.stage,
    ROUND(s.avg_goals,2) AS avg_goal,
    (SELECT AVG(home_goal + away_goal) FROM match
     WHERE season = '2012/2013') AS overall_avg
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
                     WHERE season = '2012/2013');
```

A simple example

- *Which match stages tend to have a higher than average number of goals scored?*

```sql
SELECT
    s.stage,
    ROUND(s.avg_goals,2) AS avg_goal,
    (SELECT AVG(home_goal + away_goal)
     FROM match
     WHERE season = '2012/2013') AS overall_avg
FROM (SELECT
        stage,
        AVG(home_goal + away_goal) AS avg_goals
      FROM match
      WHERE season = '2012/2013'
      GROUP BY stage) AS s -- Subquery in FROM
WHERE s.avg_goals > (SELECT AVG(home_goal + away_goal)
                     FROM match
                     WHERE season = '2012/2013'); -- Subquery in WHERE
```

A correlated example

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

A correlated example

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

Simple vs. correlated subqueries

Simple Subquery

- Can be run *independently* from the main query
- Evaluated once in the whole query

Correlated Subquery

- *Dependent* on the main query to execute
- Evaluated in loops
  - **Significantly slows down query runtime**

Correlated subqueries

- *What is the average number of goals scored in each country?*

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

```
| country     | avg_goals        |
|-------------|------------------|
| Belgium     | 2.89344262295082 |
| England     | 2.76776315789474 |
| France      | 2.51052631578947 |
| Germany     | 2.94607843137255 |
| Italy       | 2.63150867823765 |
| Netherlands | 3.14624183006536 |
| Poland      | 2.49375          |
| Portugal    | 2.63255360623782 |
| Scotland    | 2.74122807017544 |
| Spain       | 2.78223684210526 |
| Switzerland | 2.81054131054131 |
```

Correlated subqueries

- *What is the average number of goals scored in each country?*

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

```
| country     | avg_goals        |
|-------------|------------------|
| Belgium     | 2.89344262295082 |
| England     | 2.76776315789474 |
| France      | 2.51052631578947 |
| Germany     | 2.94607843137255 |
| Italy       | 2.63150867823765 |
| Netherlands | 3.14624183006536 |
| Poland      | 2.49375          |
| Portugal    | 2.63255360623782 |
| Scotland    | 2.74122807017544 |
| Spain       | 2.78223684210526 |
| Switzerland | 2.81054131054131 |
```

Let's practice!

INTERMEDIATE SQL

Nested Subqueries

INTERMEDIATE SQL

SQL

Mona Khalil
Data Scientist, Greenhouse Software

Nested subqueries??

- Subquery inside another subquery
- Perform multiple layers of transformation

A subquery...

- *How much did each country's average differ from the overall average?*

```
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 country;
```

A subquery...

```
| country     | avg_goals | avg_diff |
|-------------|-----------|----------|
| Belgium     | 2.8015    | 0.096    |
| England     | 2.7105    | 0.005    |
| France      | 2.4431    | -0.2624  |
| Germany     | 2.9016    | 0.196    |
| Italy       | 2.6168    | -0.0887  |
| Netherlands | 3.0809    | 0.3754   |
| Poland      | 2.425     | -0.2805  |
| Portugal    | 2.5346    | -0.1709  |
| Scotland    | 2.6338    | -0.0718  |
| Spain       | 2.7671    | 0.0616   |
| Switzerland | 2.9297    | 0.2241   |
```

...inside a subquery!

- *How does each month's total goals differ from the **average monthly total** of goals scored?*

```
SELECT
  EXTRACT(MONTH FROM date) AS month,
  SUM(m.home_goal + m.away_goal) AS total_goals,
  SUM(m.home_goal + m.away_goal) -
  (SELECT AVG(goals)
   FROM (SELECT
           EXTRACT(MONTH FROM date) AS month,
           SUM(home_goal + away_goal) AS goals
         FROM match
         GROUP BY month)) AS avg_diff
FROM match AS m
GROUP BY month;
```

Inner subquery

```
SELECT
  EXTRACT(MONTH from date) AS month,
  SUM(home_goal + away_goal) AS goals
FROM match
GROUP BY month;
```

```
| month | goals |
|-------|-------|
| 01    | 2988  |
| 02    | 3768  |
| 03    | 3936  |
| 04    | 4055  |
| 05    | 2719  |
| 06    | 84    |
| 07    | 366   |
```

Outer subquery

```sql
SELECT AVG(goals)
FROM (SELECT
        EXTRACT(MONTH from date) AS month,
        AVG(home_goal + away_goal) AS goals
FROM match
GROUP BY month) AS s;
```

```
2944.75
```

Final query

```
SELECT
  EXTRACT(MONTH FROM date) AS month,
  SUM(m.home_goal + m.away_goal) AS total_goals,
  SUM(m.home_goal + m.away_goal) -
  (SELECT AVG(goals)
   FROM (SELECT
           EXTRACT(MONTH FROM date) AS month,
           SUM(home_goal + away_goal) AS goals
         FROM match
         GROUP BY month) AS s) AS diff
FROM match AS m
GROUP BY month;
```

```
| month | goals | diff      |
|-------|-------|-----------|
| 01    | 5821  | -36.25    |
| 02    | 7448  | 1590.75   |
| 03    | 7298  | 1440.75   |
| 04    | 8145  | 2287.75   |
```

Correlated nested subqueries

- Nested subqueries can be correlated or uncorrelated
  - Or...a combination of the two
  - Can reference information from the *outer subquery* or *main query*

Correlated nested subqueries

- *What is the each country's average goals scored in the 2011/2012 season?*

```
SELECT
  c.name AS country,
  (SELECT AVG(home_goal + away_goal)
   FROM match AS m
   WHERE m.country_id = c.id
         AND id IN (
             SELECT id
             FROM match
             WHERE season = '2011/2012')) AS avg_goals
FROM country AS c
GROUP BY country;
```

Correlated nested subqueries

- *What is the each country's average goals scored in the 2011/2012 season?*

```
SELECT
  c.name AS country,
  (SELECT AVG(home_goal + away_goal)
   FROM match AS m
   WHERE m.country_id = c.id
         AND id IN (
             SELECT id -- Begin inner subquery
             FROM match
             WHERE season = '2011/2012')) AS avg_goals
FROM country AS c
GROUP BY country;
```

Correlated nested subquery

- *What is the each country's average goals scored in the 2011/2012 season?*

```
SELECT
  c.name AS country,
  (SELECT AVG(home_goal + away_goal)
   FROM match AS m
   WHERE m.country_id = c.id -- Correlates with main query
         AND id IN (
             SELECT id -- Begin inner subquery
             FROM match
             WHERE season = '2011/2012')) AS avg_goals
FROM country AS c
GROUP BY country;
```

Correlated nested subqueries

```
| country     | avg_goals          |
|-------------|--------------------|
| Belgium     | 2.87916666666667 |
| England     | 2.80526315789474 |
| France      | 2.51578947368421 |
| Germany     | 2.85947712418301 |
| Italy       | 2.58379888268156 |
| Netherlands | 3.25816993464052 |
| Poland      | 2.19583333333333 |
| Portugal    | 2.64166666666667 |
| Scotland    | 2.6359649122807  |
| Spain       | 2.76315789473684 |
| Switzerland | 2.62345679012346 |
```

Let's practice!

INTERMEDIATE SQL

datacamp

Common Table Expressions

INTERMEDIATE SQL

SQL

Mona Khalil
Data Scientist, Greenhouse Software

datacamp

When adding subqueries...

- Query complexity increases quickly!
  - Information can be difficult to keep track of

Solution: **Common Table Expressions!**

Common Table Expressions

Common Table Expressions (CTEs)

- Table *declared* before the main query
- *Named* and *referenced* later in `FROM` statement

Setting up CTEs

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

Take a subquery in FROM

```sql
SELECT
  c.name AS country,
  COUNT(s.id) AS matches
FROM country AS c
INNER JOIN (
  SELECT country_id, id
  FROM match
  WHERE (home_goal + away_goal) >= 10) AS s
ON c.id = s.country_id
GROUP BY country;
```

```
| country     | matches |
|-------------|---------|
| England     | 3       |
| Germany     | 1       |
| Netherlands | 1       |
| Spain       | 4       |
```

Place it at the beginning

```
(
  SELECT country_id, id
  FROM match
  WHERE (home_goal + away_goal) >= 10
)
```

Place it at the beginning

```
WITH s AS (
  SELECT country_id, id
  FROM match
  WHERE (home_goal + away_goal) >= 10
)
```

Show me the CTE

```
WITH s AS (
  SELECT country_id, id
  FROM match
  WHERE (home_goal + away_goal) >= 10
)
SELECT
  c.name AS country,
  COUNT(s.id) AS matches
FROM country AS c
INNER JOIN s
ON c.id = s.country_id
GROUP BY country;
```

```
| country     | matches |
|-------------|---------|
| England     | 3       |
| Germany     | 1       |
| Netherlands | 1       |
| Spain       | 4       |
```

Show me all the CTEs

```
WITH s1 AS (
  SELECT country_id, id
  FROM match
  WHERE (home_goal + away_goal) >= 10),
s2 AS (                                 -- New subquery
  SELECT country_id, id
  FROM match
  WHERE (home_goal + away_goal) <= 1
)
SELECT
  c.name AS country,
  COUNT(s1.id) AS high_scores,
  COUNT(s2.id) AS low_scores            -- New column
FROM country AS c
INNER JOIN s1
ON c.id = s1.country_id
INNER JOIN s2                           -- New join
ON c.id = s2.country_id
GROUP BY country;
```

Why use CTEs?

- Executed once
  - CTE is then stored in memory
  - Improves query performance
- Improving organization of queries
- Referencing other CTEs
- Referencing itself (`SELF JOIN`)

Let's Practice!

INTERMEDIATE SQL

Deciding on techniques to use

INTERMEDIATE SQL

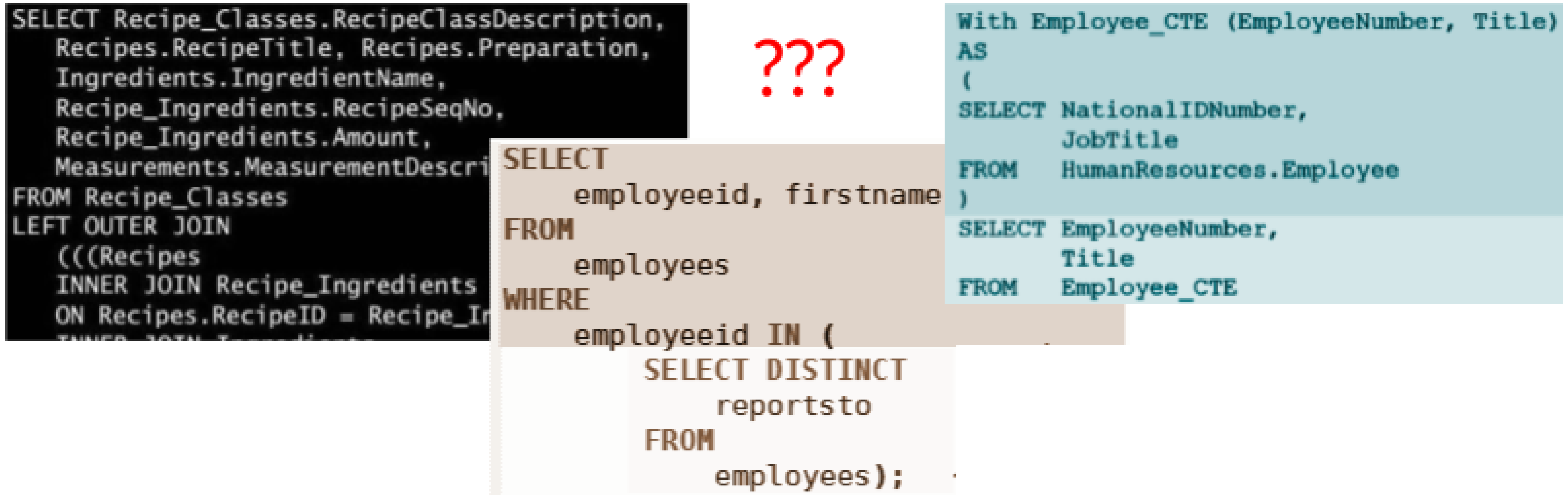
SQL

Mona Khalil
Data Scientist, Greenhouse Software

datacamp

Different names for the same thing?

- Considerable overlap...



```
SELECT Recipe_Classes.RecipeClassDescription,
   Recipes.RecipeTitle, Recipes.Preparation,
   Ingredients.IngredientName,
   Recipe_Ingredients.RecipeSeqNo,
   Recipe_Ingredients.Amount,
   Measurements.MeasurementDescri
FROM Recipe_Classes
LEFT OUTER JOIN
   (((Recipes
   INNER JOIN Recipe_Ingredients
   ON Recipes.RecipeID = Recipe_I
```

```
SELECT
    employeeid, firstname
FROM
    employees
WHERE
    employeeid IN (
        SELECT DISTINCT
            reportsto
        FROM
            employees);
```

```
With Employee_CTE (EmployeeNumber, Title)
AS
(
SELECT NationalIDNumber,
       JobTitle
FROM   HumanResources.Employee
)
SELECT EmployeeNumber,
       Title
FROM   Employee_CTE
```

- ...but **not** identical!

Differentiating Techniques

Joins

- Combine 2+ tables
  - Simple operations/aggregations

Correlated Subqueries

- Match subqueries & tables
  - Avoid limits of joins
  - **High processing time**

Multiple/Nested Subqueries

- Multi-step transformations
  - Improve accuracy and reproducibility

Common Table Expressions

- Organize subqueries sequentially
- Can reference other CTEs

So which do I use?

- Depends on your database/question
- The technique that best allows you to:
  - Use and reuse your queries
  - Generate clear and accurate results

Different use cases

Joins

- 2+ tables (*What is the total sales per employee?*)

Correlated Subqueries

- *Who does each employee report to in a company?*

Multiple/Nested Subqueries

- *What is the average deal size closed by each sales representative in the quarter?*

Common Table Expressions

- *How did the marketing, sales, growth, & engineering teams perform on key metrics?*

Let's Practice!

INTERMEDIATE SQL