WOMEN WHO CODE®
/data-science

SQL Coding Series

WEBINAR SERIES

SESSION 5
Advanced

| FRIDAY, FEB 10

| 10 PM ET

Sneha Saxena

Data Scientist @ Grubhub

womenwhocode.com/datascience/events

ADVANCED SESSION

AGENDA

1. Subqueries
2. Window Functions
3. Common Table Expressions (CTEs)

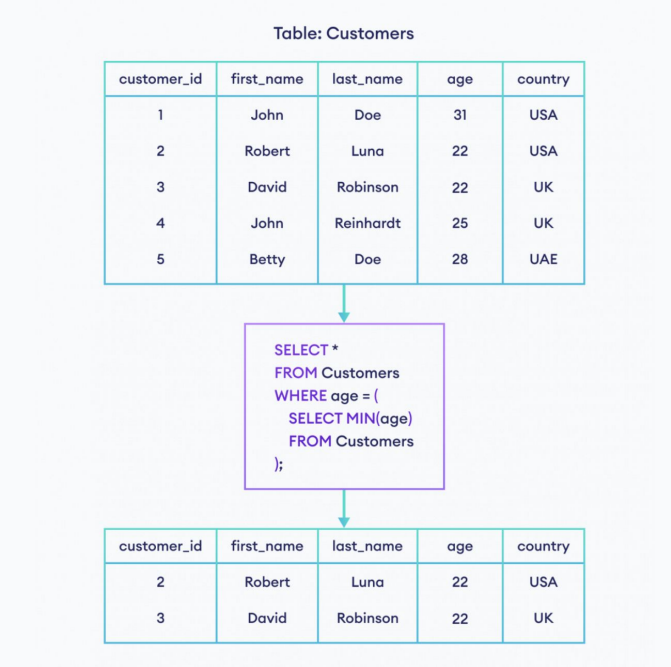
SUBQUERIES

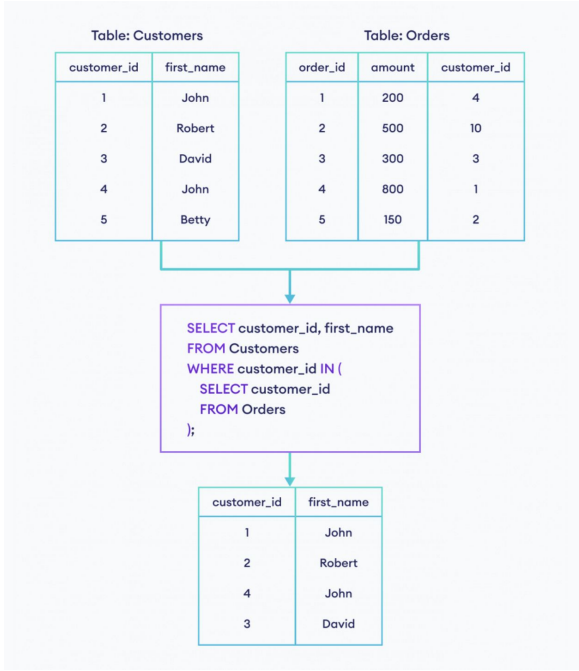
A subquery is a nested or inner query that is present inside another SQL query

Use Cases

- Using subqueries to aggregate in multiple stages
- Subqueries in conditional logic
- Joining subqueries

SUBQUERIES - Examples





SUBQUERY STRUCTURE

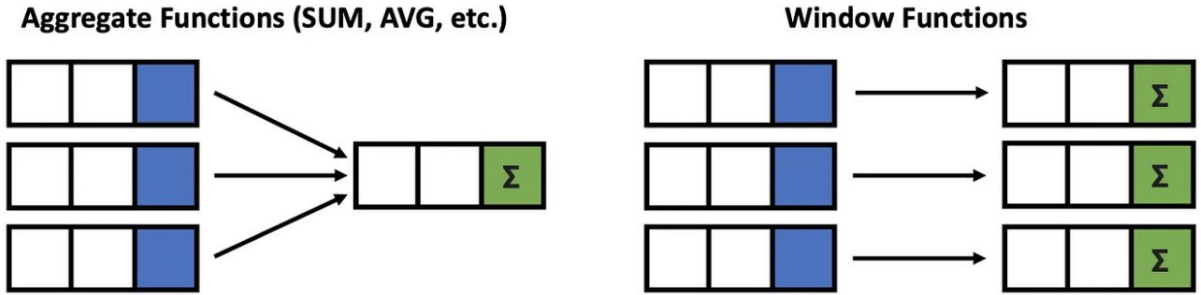
```
SELECT sub.*
  FROM (
       <<results from inner query go here>>
       ) sub
 WHERE sub.resolution = 'NONE'
```

SUBQUERIES vs JOINS

- In several scenarios Joins are optimized and more efficient to run than subqueries
- Subqueries can help make complex code more readable and easy to debug
- Multiple joins can sometimes make queries heavy and can be hard to read

WINDOW FUNCTIONS

> *"A* window function *performs a calculation across a set of table rows that are somehow related to the current row…Behind the scenes, the window function is able to access more than just the current row of the query result."*



Window Function: SYNTAX

```
SELECT <column_1>, <column_2>,
        <window_function> (expression) OVER
        (PARTITION BY <partition_list> ORDER BY <order_list> ROWS frame_clause)
FROM <table_name>
```

Example

JOB_TITLE	SALARY
ANALYST	3100
ANALYST	2900
ANALYST	3250
SALES	1700
SALES	2500
SALES	4100
SALES	1600
SALES	2200
ENGINEER	3500
ENGINEER	3100
ENGINEER	4100

GROUP BY

JOB_TITLE	AVG_SALARY
ANALYST	3083.33333
ENGINEER	3566.66667
SALES	2420

Window Function

JOB_TITLE	SALARY	AVG_SALARY
ANALYST	3100	3083.333333
ANALYST	2900	3083.333333
ANALYST	3250	3083.333333
SALES	1700	2420
SALES	2500	2420
SALES	4100	2420
SALES	1600	2420
SALES	2200	2420
ENGINEER	3500	3566.666667
ENGINEER	3100	3566.666667
ENGINEER	4100	3566.666667

Commonly used WINDOW FUNCTIONS

- ➔ SUM, COUNT, AVG
- ➔ ROW_NUMBER
- ➔ RANK, DENSE_RANK
- ➔ LAG, LEAD
- ➔ NTILE

CTEs

Common table expressions (CTEs), also known as WITH clauses, are used to create named subqueries that can be referenced in the main query. CTEs are not saved for future use and **can be referenced only within the query** where they are defined. The basic syntax is:

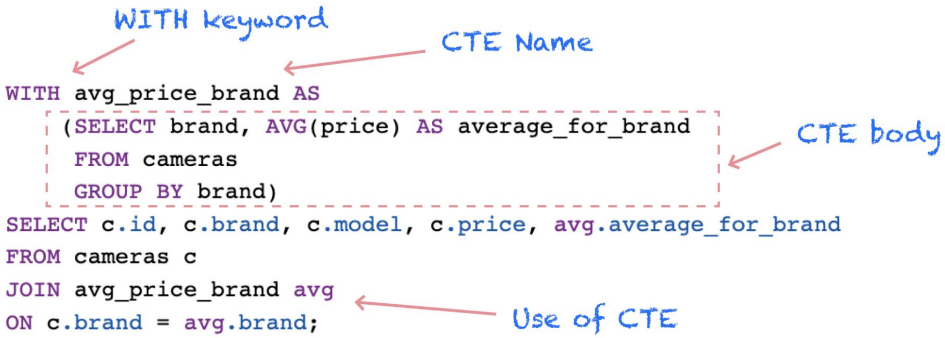
Code

```
WITH cte_name AS
(SELECT … cte body ... )
SELECT … main query  ...
```

Example

cameras

id	brand	model	megapixels	price
11	Canon	EOS 6D Mark II	26.2	1399.00
12	Canon	EOS Rebel T7	24.1	449.00
13	Canon	PowerShot ELPH 320 HS	16.1	249.95
14	Canon	PowerShot ELPH 500 HS	12.1	99.97
15	Kodak	PIXPRO Astro Zoom AZ652-BK	20	254.90
16	Kodak	PIXPRO Astro Zoom AZ252-RD	16	122.55
17	Kodak	EasyShare Z950	12	199.00



Why use CTEs?

While common table expressions operate similarly to subqueries, they have several benefits including:

- The ability to reference the same temporary result set repeatedly across the query
- Improved readability for collaboration and debugging
- Better visibility into commonly used result sets that are good candidates to become permanent tables/views

Join us on Slack to ask questions and keep the discussion going!

Use the channel:

#sql-coding-series

We would love to know your experience attending this series.

Please fill the feedback form shared in the chat box.
It will help us improve our content and delivery for future sessions.

Code with SQL

https://www.sql-practice.com/

Thank you!

WOMEN WHO CODE®
/data-science