SUB-SELECTS IN SQL

LEARNING OBJECTIVES

1. Ask two or more questions in a single SQL query.
2. Nest queries within different parts of **SELECT** statements.
3. Perform multi-step aggregations or filtering within one query.

INTRODUCTION: SUBSELECT

SUBSELECTS = SUBQUERIES

Until now, queries had no effect on each other....but what if we wanted to use the results from one query to inform another? *Subselects, also called subqueries, allow us to do just that.*

We can use subqueries to create nested queries that:

1. Run one query,
2. Get a result, and
3. Then "feed" that result immediately into another query.

Subqueries may be used within `WHERE`, `FROM`, or `SELECT` statements.

SUBSELECTS = SUBQUERIES

Subqueries also let us change the level of aggregation on the data and perform analysis on a larger scale.

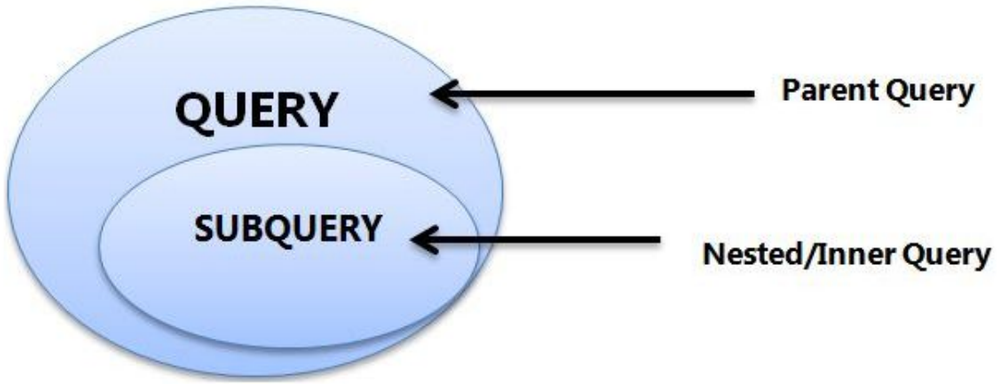
For example, sales in the Iowa Liquor Database are on a transactional basis, but we can reaggregate this to a monthly or weekly basis to look at trends at different levels.

DEMO:

SUBSELECT SYNTAX

SUBSELECTS: BASIC CONSTRUCTION GUIDELINES

- Subqueries are enclosed in parentheses.
- Subquery structures need complete query components for execution: they must have `SELECT`, `FROM`, and a specified criteria.
- Queries placed in subqueries will execute first (based on the inner loop).
- Assigning aliases is important for syntax and readability.



Source: graphic https://goo.gl/MsfKjc

SUBSELECTS: ASKING MULTIPLE QUESTIONS

Let's say you have two locations and you want to know, *on average,* how often each store is making more than $150,000.

In other words, you need to get the count for when recorded sales are more than $150,000 for each location and then create an average for that performance record.

In the subquery:

1. We need to identify the store with average sales of more than $150,000.
2. We need to compute the average sales for each store.

SUBSELECTS: SAMPLE TABLE

Let's use this sample data table to create a subquery.

The table is named "T1" (as an alias for: "Table 1") and contains the total sales by location for each day.

T1

DAYS	LOCATION	SALES
DAY 1	101	100,000
DAY 2	101	200,000
DAY 3	101	165,000
DAY 4	101	175,000
DAY 1	102	125,000
DAY 2	102	150,000
DAY 3	102	175,000
DAY 4	102	75,000

SUBSELECT SYNTAX WALK THROUGH

1. Review the subquery first:

```
SELECT location, AVG(Top_Stores)
FROM (
   SELECT location,
   COUNT(*) as Top_Stores
   FROM table_1
   WHERE sales >150000
   GROUP BY location) as T1
GROUP BY T1.Location;
```

2. Select all of the locations.
3. Do a **COUNT** of those locations and rename the result **Top_Stores**.
4. Filter the results for only stores with more than $150,000 in sales.
5. Because there is an aggregation (**COUNT**), there has to be a group based on what is being aggregated (**location**).
6. **T1** is an alias for the results. T1 serves as a temporary table, holding the results of this **SELECT** statement.
   a. You can rename the alias whatever you'd like.

SUBSELECT SYNTAX WALK THROUGH

Next, review the SELECT statement that will be reading results from T1:

```
SELECT location, AVG(Top_Stores)
FROM (
        SELECT location, COUNT(*)
         as Top_Stores FROM table_1
        WHERE sales >150000
        GROUP BY location) as T1
GROUP BY T1.Location;
```

- This selects locations and an average for **`Top_Stores`**, which was created in your inner query using the **`COUNT`** function.
- The **`FROM`** tells us where the information is coming from: the subquery.
- Because there is an aggregation (**`AVG`**), the locations need to be grouped.
- Notice in the **`GROUP BY`** that you have to reference the alias **`T1`**.

SUBSELECTS: QUERY TYPES

A subquery is a **query within a query** that allows for **multi-step operations**.

Subqueries are a common requirement when dealing with layered aggregations, filtering, and creating temporary tables with calculated or categorized columns.

There are **two types** of subqueries we'll be exploring today:

1. Type 1 allows us to feed the results of one query as additional data for the outer query.
2. Type 2 allows us to create a filter to test data against using the `IN` operator.

SUBSELECT SYNTAX: USING RESULTS

First, we will use the results of one query as a data source for the outer query by placing the subquery in the **FROM** clause. The syntax generally looks like this:

```
SELECT something FROM (SELECT something else FROM somewhere) AS temp;
```

We supply an entire query between parentheses where we'd normally have a table. The outer query then treats the results of the inner query as if they were another table.

Each independent **SELECT** query can be as complex as is needed. Give an alias name to the inner query with the **AS** keyword, even if you never refer to the inner query by this alias.

SUBSELECT SYNTAX: USING RESULTS

Let's walk through an example:

1. Classify the amount of mL per product into three categories: high_volume, med_volume, and low_volume.
2. Use this formula from the Products table: `bottle_size * pack * inner_pack`.

Use a subquery to *first calculate* the value "volume:"

```
SELECT item_description, bottle_size * pack * inner_pack AS
volume FROM products;
```

SUBSELECT SYNTAX: USING RESULTS

Then, classify it in the outer query:

```
SELECT *, CASE
    WHEN volume >= 50000 THEN 'high_volume'
    WHEN volume >= 7500 THEN 'med_volume'
    ELSE 'low_volume'
END as volume_category
FROM
    (SELECT item_description, bottle_size * pack * inner_pack AS
    volume
    FROM products) AS temp
ORDER BY 2 DESC;
```

SUBSELECT SYNTAX: CREATING FILTER

The second type of subquery allows you to filter the results of one query by the results of another using the `IN` operator. This type of query looks like:

```
SELECT *
FROM table_one
WHERE col_one IN (SELECT other_val FROM some_other_table);
```

- Here, we will be able to filter `table_one` where the values of `col_one` exist in the results of our inner subquery.
- In this use case, we do **not** need an alias on the subquery. We also can only call one column in the subquery.

SUBSELECT SYNTAX WALK THROUGH: CREATING FILTER

Let's look at an example:

1. How many sales occurred in counties with less than 100,000 people?
2. Write the query that selects the appropriate counties.

SUBSELECT SYNTAX WALK THROUGH: CREATING FILTER

Let's look at an example:

1. How many sales occurred in counties with less than 100,000 people?
2. Write the query that selects the appropriate counties.

```
SELECT county FROM counties WHERE population < 100000;
```

SUBSELECT SYNTAX WALK THROUGH: CREATING FILTER

Then, insert it into the outer query's **WHERE** clause:

```
SELECT COUNT(*)
FROM sales
WHERE county IN (SELECT county FROM counties WHERE
population < 100000);
```

Here, the inner query must **only** return one column for an **IN** filter. Consider that the inner query is *building an array of values* that are then substituted into the **WHERE** clause in the outer loop.

SUBSELECT SYNTAX: CREATING FILTER

Another advantage of this methodology is that the `NOT IN` function allows you to use a subquery to **exclude** data as well.

Going back to the previous query, we can find the number of sales in counties with populations more than 100,000 people in either of two ways.

1. Option 1: use a `NOT IN` statement
2. Option 2: change the logical operator from **less than** to **greater than**.

SUBSELECT SYNTAX: CREATING FILTER

Query 1:

```
SELECT COUNT(*)

FROM sales

WHERE county NOT
IN (SELECT county
FROM counties
WHERE population <
100000);
```

Query 2:

```
SELECT COUNT(*)

FROM sales

WHERE county IN
(SELECT county
FROM counties
WHERE population >
100000);
```

Run both of these queries and observe the results.

SUBSELECTS: RUNNING MULTIPLE QUERIES

You can also use the subqueries methodology to run multiple **SELECT** statements at once and get the results on the same screen.

For example, on the last slide, there were two **SELECT** statements that were run separately. We can wrap each **SELECT** statement in its own parentheses and they will be evaluated separately in a common results pane.

SUBSELECTS: RUNNING MULTIPLE QUERIES

```
select
(SELECT COUNT(*)
FROM sales
WHERE county NOT IN (SELECT county
FROM counties WHERE population <
100000)) as query_1_result

, (SELECT COUNT(*)
FROM sales
WHERE county IN (SELECT county FROM
counties WHERE population > 100000)
) as query_2_result
```

1. We have to start the query with a **SELECT** statement.
2. Notice that each query is wrapped in its own parentheses.
3. You'll need to separate each subquery with a comma.
4. **Alias** each query to keep track of the results.

GUIDED PRACTICE: SUBSELECTS

SUBSELECTS: MORE GUIDELINES

Be careful with the logic you design into your query with a `GROUP BY` clause. Be sure to match the location of the aggregation being performed.

- `CASE` statements can be used in *both types* of subqueries: the inner query and the outer query.
- One useful and common construction is to use the inner query to create a **flag** (a binary value) and then use the outer query to average that value.
- Let's build an example using a `CASE` classification and a binary flag.

SUBSELECT WITH CASE

Earlier, we classified counties by population as large, medium, and small. Now, let's say we want to use those categories to count how many counties fit into each group.

Let's look at our original classification query:

```
SELECT county,
population, CASE
   WHEN population >= 400000 THEN
   'large'
   WHEN population >= 100000 THEN
   'medium'
   ELSE 'small'
   END AS county_size
FROM counties;
```

SUBSELECT WITH CASE

This will serve as our inner query, with the outer query counting occurrences:

```
SELECT county_size, COUNT(*) AS number_counties
FROM (
    SELECT county, population,
        CASE
            WHEN population >= 400000 THEN 'large'
            WHEN population >= 100000 THEN 'medium'
            ELSE 'small'
        END AS county_size FROM counties) AS temp
GROUP BY county_size;
```

SUBSELECT USING A BINARY FLAG

Let's say you wanted to know, per store, what percentage of sales were more than $100.

First, create the inner query that classifies each sale as either more or less than $100:

```
SELECT store, total,
    CASE
        WHEN total > 100 THEN 1
        ELSE 0
    END AS over_100
FROM sales;
```

SUBSELECT USING A BINARY FLAG

Now, wrap that query in an outer query that averages the flag per store:

```
SELECT store, ROUND(AVG(over_100),4) AS
pct_over_100

FROM (
        SELECT store, total,
        CASE
            WHEN total > 100
        THEN 1
            ELSE 0
        END AS over_100
        FROM sales) AS temp
GROUP BY store
ORDER BY store;
```

INDEPENDENT PRACTICE: SUBSELECTS

ACTIVITY: SUBSELECTS

DIRECTIONS

The State of Iowa is getting a lot of press because of the counties that were designated for its alcoholism pilot program, and now the PR office has to respond to a couple of press inquiries.

As the analysts at Deloitte who made a recommendation for the program, we're now asked to respond to the following questions that came in via a press inquiry. Please use subqueries and CASE statements when possible.

1) Which counties have the highest sale of whisky or vodka products?
2) Which counties have the highest sale of items higher than 80 proof?
3) What percentage of sales per county are more than $100? What are the top five counties in this category?
4) What are the top five counties in terms of mL per capita?
5) Based on number of sales, what were the top five categories of liquor sold in the five most populous counties?

ACTIVITY: SUBSELECTS

SOLUTION

1) Which counties have the highest sale of whisky or vodka products?

ACTIVITY: SUBSELECTS

SOLUTION

1) Which counties have the highest sale of whisky or vodka products?

```
SELECT county, round(avg(whisk_or_vodka),4) AS pct_whisk_or_vodka
    FROM (
          SELECT county,
              CASE
              WHEN category_name iLIKE '%WHISK%' OR category_name
  LIKE '%VODKA%'
              THEN 1
              ELSE 0
               END AS whisk_or_vodka FROM sales) AS temp
    GROUP BY county
    ORDER BY pct_whisk_or_vodka DESC;
```

ACTIVITY: SUBSELECTS

SOLUTION

2) Which counties have the highest sale of items higher than 80 proof?

ACTIVITY: SUBSELECTS

SOLUTION

2) Which counties have the highest sale of items higher than 80 proof?

```
SELECT county, avg(over_80) AS pct_over_80
FROM (
    SELECT county,
        CASE
        WHEN CAST(proof AS numeric) >= 80 THEN 1 ELSE
        0
        END AS over_80
    FROM sales a
     INNER JOIN products b
     ON a.item = b.item_no) AS temp
GROUP BY county
ORDER BY pct_over_80 DESC;
```

ACTIVITY: SUBSELECTS

SOLUTION

3) What percentage of sales per county are more than $100? What are the top five counties in this category?

ACTIVITY: SUBSELECTS

SOLUTION

3) What percentage of sales per county are more than $100? What are the top five counties in this category?

```
SELECT county, round(avg(over_100),4) AS pct_over_100
FROM (
     SELECT county,
          CASE
                    WHEN total > '100' THEN 1
                    ELSE 0
                    END AS over_100
       FROM sales) AS temp
GROUP BY county
ORDER BY pct_over_100 DESC
LIMIT 5;
```

ACTIVITY: SUBSELECTS

SOLUTION

4) What are the top five counties in terms of mL per capita?

ACTIVITY: SUBSELECTS

SOLUTION

4) What are the top five counties in terms of mL per capita?

```
SELECT county, ROUND(CAST(mL_total AS numeric)/population,2) AS
    mL_per_cap
FROM counties
LEFT JOIN (
     SELECT county, SUM(pack * liter_size * bottle_qty) AS
    mL_total
     FROM sales
     GROUP BY county) AS temp
USING (county)
ORDER BY mL_per_cap DESC
LIMIT 5;
```

ACTIVITY: SUBSELECTS

SOLUTION

5) Based on number of sales, what were the top five categories of liquor sold in the five most populous counties?

ACTIVITY: SUBSELECTS

SOLUTION

5) Based on number of sales, what were the top five categories of liquor sold in the five most populous counties?

```
SELECT category_name, COUNT(*) AS count_of_sales
FROM sales
WHERE county IN (SELECT county FROM counties
           ORDER BY population DESC LIMIT 5)
GROUP BY category_name
ORDER BY count_of_sales DESC;
```

CONCLUSION

REVIEW: SUBSELECTS

In today's lesson, we learned:

1. When and why you would use a subselect in order to add an extra dimension to your ability to query data.
2. How to execute the correct syntax, including in subqueries within `FROM` and `WHERE` clauses.
3. How to answer business questions requiring multiple operations with your newfound SQL knowledge.

SUBSELECTS IN SQL

Q&A

RESOURCES

RESOURCES

- “Using Subqueries in the Select Statement” by essentialSQL.com: https://goo.gl/GsfwAb
- “Writing Subqueries in SQL” by Mode Analytics: https://community.modeanalytics.com/sql/tutorial/sql-subqueries/
- Beginner SQL Tutorial: http://beginner-sql-tutorial.com/sql-subquery.htm

CITATIONS

- Subquery graphic illustration from c-sharpcorner.com: https://goo.gl/MsfKjc.