Chapter 5 : Subqueries

Aviral Janveja

1 Introduction

Subqueries are queries nested inside another query. Subqueries allow you to break down complex queries into smaller, more manageable steps. The following subquery filters the given data in two steps :

```
SELECT sub.*
FROM (
  SELECT *
  FROM tutorial.sf_crime_incidents_2014_01
  WHERE day_of_week = 'Friday'
 ) sub
WHERE sub.resolution = 'NONE'
```

Let us break down the above subquery step by step. Looking at the **inner query** first :

```
SELECT *
FROM tutorial.sf_crime_incidents_2014_01
WHERE day_of_week = 'Friday'
```

The database runs the inner query first, narrowing down incidents to those that happened on a Friday. The **outer query** takes this result and then filters it further to include only unresolved incidents. That is, where resolution = 'NONE' :

```
SELECT sub.*
FROM ( ... ) sub
WHERE sub.resolution = 'NONE'
```

Subqueries are required to have names, which are added after the parentheses, the same way you would add an alias to a normal table. In the above example, we have used the name **sub**.

The outer query uses **SELECT sub.*** to specify that it should select all columns from the result of the subquery called sub. This is equivalent to selecting each column individually, for example : sub.column1, sub.column2 and so on.

2 Using Subqueries to Aggregate

What if you wanted to know how many crime-incidents happen on average, on a Friday in January? There are two steps to this process: counting the number of incidents each day (**inner query**), then determining the monthly average (**outer query**). Let us write the inner query first :

```
SELECT day_of_week,
           date,
           COUNT(incidnt_num) AS incidents
      FROM tutorial.sf_crime_incidents_2014_01
     GROUP BY 1,2
```

Through the above inner query, we first count the number of incidents for each day of the week, Monday through Friday for all weeks included in the dataset. Next, we include the outer query which calculates the monthly average for each day of the week, such that you can compare what is the average number of crimes happening on a Friday in November versus Saturday in November or Friday in December and so on :

```
SELECT LEFT(sub.date, 2) AS cleaned_month,
   sub.day_of_week,
   AVG(sub.incidents) AS average_incidents
FROM (
    SELECT day_of_week,
           date,
           COUNT(incidnt_num) AS incidents
      FROM tutorial.sf_crime_incidents_2014_01
     GROUP BY 1,2
   ) sub
GROUP BY 1,2
ORDER BY 1,2
```

Note : In general, it is easier to write the inner query first, make sense of the result and then proceed to implementing the outer query.

3 Using Subqueries in Conditional Statements

You can use subqueries in conditional statements like WHERE and CASE. The following query for example, returns all of the entries from the earliest date in the dataset :

```
SELECT *
FROM tutorial.sf_crime_incidents_2014_01
WHERE Date = (SELECT MIN(date)
              FROM tutorial.sf_crime_incidents_2014_01
              )
```

Note that you should not include an alias when you write a subquery in a conditional statement. This is because the subquery is treated as an individual value or set of values, rather than as a table. For example, the result of the inner query above is only one cell, the minimum date value.

4 Using Subqueries with JOIN

Joining subqueries can be particularly useful when combined with aggregations. For example, the following query ranks all results according to how many incidents were reported in a given day. It does this by aggregating the total number of incidents each day in the inner query and then using those values to sort via the outer query :

```
SELECT incidents.*,
   sub.incidents AS incidents_that_day
FROM tutorial.sf_crime_incidents_2014_01 incidents
JOIN ( SELECT date,
       COUNT(incidnt_num) AS incidents
        FROM tutorial.sf_crime_incidents_2014_01
       GROUP BY 1
   ) sub
ON incidents.date = sub.date
ORDER BY sub.incidents DESC, time
```

5 Using Subqueries with UNION

It is possible that a dataset comes split into several parts, especially if it has passed through Excel at any point, since Excel can only handle about 1M rows per spreadsheet. However, we would like to perform operations on the entire dataset at once, rather than on the individual parts. You can do this by using a subquery alongside UNION :

```
SELECT COUNT(*) AS total_rows
FROM (
    SELECT *
      FROM tutorial.crunchbase_investments_part1

     UNION ALL

    SELECT *
      FROM tutorial.crunchbase_investments_part2
   ) sub
```