https://classroom.udacity.com/courses/ud002-bert/lessons/b50a9cfd-566a-4b42-bf4f-70081b557c0b/concepts/b53dc474-19a9-4969-8fdf-5d1f164b18ff

**SQL SUBQUERIES & TABLES In 31**

## Solutions to Your First Subquery

1. First, we needed to group by the day and channel. Then ordering by the number of events (the third column) gave us a quick way to answer the first question.

```
SELECT DATE_TRUNC('day',occurred_at) AS day,
   channel, COUNT(*) as events
FROM web_events
GROUP BY 1,2
ORDER BY 3 DESC;
```

2. Here you can see that to get the entire table in question 1 back, we included an `*` in our SELECT statement. You will need to be sure to alias your table.

```
SELECT *
FROM (SELECT DATE_TRUNC('day',occurred_at) AS day,
            channel, COUNT(*) as events
     FROM web_events
     GROUP BY 1,2
     ORDER BY 3 DESC) sub;
```

3. Finally, here we are able to get a table that shows the average number of events a day for each channel.

```
SELECT channel, AVG(events) AS average_events
FROM (SELECT DATE_TRUNC('day',occurred_at) AS day,
             channel, COUNT(*) as events
      FROM web_events
      GROUP BY 1,2) sub
GROUP BY channel
ORDER BY 2 DESC;
```

## Subquery Formatting

**When writing Subqueries, it is easy for your query to look incredibly complex. In order to assist your reader, which is often just yourself at a future date, formatting SQL will help with understanding your code.**

**The important thing to remember when using subqueries is to provide some way for the reader to easily determine which parts of the query will be executed together. Most people do this by indenting the subquery in some way - you saw this with the solution blocks in the previous concept.**

**The examples in this class are indented quiet far—all the way to the parentheses. This isn't practical if you nest many subqueries, but in general, be thinking about how to write your queries in a readable way. Examples of the same query written multiple different ways is provided below. You will see that some are much easier to read than others.**

---

## Badly Formatted Queries

Though these poorly formatted examples will execute the same way as the well formatted examples, they just aren't very friendly for understanding what is happening!

Here is the first, where it is impossible to decipher what is going on:

```
SELECT * FROM (SELECT DATE_TRUNC('day',occurred_at) AS day, channel, COUNT(*) as events FROM web_events GROUP BY 1,2 ORDER BY 3 DESC) sub;
```

This second version, which includes some helpful line breaks, is easier to read than that previous version, but it is still not as easy to read as the queries in the Well Formatted Query section.

```
SELECT *
FROM (
SELECT DATE_TRUNC('day',occurred_at) AS day,
channel, COUNT(*) as events
FROM web_events
GROUP BY 1,2
ORDER BY 3 DESC) sub;
```

## Well Formatted Query

Now for a well formatted example, you can see the table we are pulling from much easier than in the previous queries.

```
SELECT *
FROM (SELECT DATE_TRUNC('day',occurred_at) AS day,
                channel, COUNT(*) as events
      FROM web_events
      GROUP BY 1,2
      ORDER BY 3 DESC) sub;
```

Additionally, if we have a GROUP BY, ORDER BY, WHERE, HAVING, or any other statement following our subquery, we would then indent it at the same level as our outer query.

The query below is similar to the above, but it is applying additional statements to the outer query, so you can see there are GROUP BY and ORDER BY statements used on the output are not tabbed. The inner query GROUP BY and ORDER BY statements are indented to match the inner table.

```
SELECT *
FROM (SELECT DATE_TRUNC('day',occurred_at) AS day,
```

```
                channel, COUNT(*) as events
      FROM web_events
      GROUP BY 1,2
      ORDER BY 3 DESC) sub
GROUP BY channel
ORDER BY 2 DESC;
```

**These final two queries are so much easier to read!**

## Queries Needed to Find the Solutions to the Previous Quiz

1. **Here is the necessary quiz to pull the first month/year combo from the orders table.**

```
SELECT DATE_TRUNC('month', MIN(occurred_at))
FROM orders;
```

2. **Then to pull the average for each, we could do this all in one query, but for readability, I provided two queries below to perform each separately.**

```
SELECT AVG(standard_qty) avg_std, AVG(gloss_qty) avg_gls,
AVG(poster_qty) avg_pst
FROM orders
WHERE DATE_TRUNC('month', occurred_at) =
     (SELECT DATE_TRUNC('month', MIN(occurred_at)) FROM orders);

SELECT SUM(total_amt_usd)
FROM orders
WHERE DATE_TRUNC('month', occurred_at) =
      (SELECT DATE_TRUNC('month', MIN(occurred_at)) FROM orders);
```

The **WITH** statement is often called a **Common Table Expression** or **CTE**. Though these expressions serve the exact same purpose as subqueries, they are more common in practice, as they tend to be cleaner for a future reader to follow the logic.
Had to read a lot about this WITH statement it seems to be tough sometimes