Bertelsmann Tech Scholarship - Data Track

5. In which month of which year did Walmart spend the most on gloss paper in terms of dollars?

```
SELECT DATE_TRUNC('month', o.occurred_at) ord_date,
SUM(o.gloss_amt_usd) tot_spent
FROM orders o
JOIN accounts a
ON a.id = o.account_id
WHERE a.name = 'Walmart'
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1;
```

28.29 Video: CASE Statements

- CASE state always goes in SELECT clause, include following components: WHEN, THEN, and END.
- ELSE = optional component to catch cases didn't meet any of the other previous CASE conditions.
- Can make any conditional state use any conditi operator (like WHERE) between WHEN and THEN.
- This includes stringing together multiple conditional states using AND and OR.
- Can include multiple WHEN states, as well as an ELSE state again, to deal with any unaddressed conditions.

Derive
TAKE DATA FROM EXISTING COLUMNS AND MODIFY THEM

"CASE" STATEMENT HANDLES "IF" "THEN" LOGIC

"CASE" STATEMENTS MUST END WITH THE WORD "END"

Else
CAPTURES VALUES NOT SPECIFIED IN "WHEN" AND "THEN" STATEMENTS

28.30 CASE-Aggregations using WHERE clause only get 1 set of data from the CASE at a time.

```
SELECT id,
       account_id,
       occurred_at,
       channel,
       CASE WHEN channel = 'facebook' OR channel = 'direct' THEN 'yes' ELSE 'no' END AS is_facebook
  FROM demo.web_events_full
 ORDER BY occurred_at
```

- Separating data into separate columns like this depending on what you want to do, but this separation might be easier to do in another language than w/ SQL.

```
SELECT account_id,
       occurred_at,
       total,
       CASE WHEN total > 500 THEN 'Over 500'
            WHEN total > 300 AND total <= 500 THEN '301 - 500'
            WHEN total > 100 AND total <= 300 THEN '101 - 300'
            ELSE '100 or under' END AS total_group
  FROM demo.orders
```

1. Write a query to display for each order, the account ID, total amount of the order, and the level of the order - 'Large' or 'Small' - depending on if the order is $3000 or more, or smaller than $3000.

```
SELECT account_id, total_amt_usd,
CASE WHEN total_amt_usd > 3000 THEN 'Large'
      ELSE 'Small' END AS total_amt_order
FROM orders;
```

(1)

2. Write a query to display the number of orders in each of three categories, based on the total number of items in each order. The three categories are: 'At Least 2000', 'Between 1000 and 2000' and 'Less than 1000'.

```
SELECT CASE WHEN total >= 2000 THEN 'At Least 2000'
WHEN total >= 1000 AND total < 2000 THEN 'Between 1000 and 2000'
ELSE 'Less than 1000' END AS order_category,
COUNT(*) AS order_count
FROM orders
GROUP BY 1;
```

(2)

3. We would like to understand 3 different levels of customers based on the amount associated with their purchases. The top level includes anyone with a Lifetime Value (total sales of all orders) greater than 200,000 usd. The second level is betw 200,000 and 100,000 usd. The lowest level is anyone under 100,000 usd. Provide a table that includes the level associated with each account. You should provide the account name, the total sales of all orders for the customer, and the level. Order with the top spending customers listed first.

```
SELECT a.name, SUM(total_amt_usd) total_spent,
   CASE WHEN SUM(total_amt_usd) > 200000 THEN 'top'
   WHEN  SUM(total_amt_usd) > 100000 THEN 'middle'
   ELSE 'low' END AS customer_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
GROUP BY a.name
ORDER BY 2 DESC;
```

(3)

4. We would now like to perform a similar calculation to the first, but we want to obtain the total amount spent by customers only in 2016 and 2017. Keep the same levels as in the previous question. Order with the top spending customers listed first.

```
SELECT a.name, SUM(total_amt_usd) total_spent,
   CASE WHEN SUM(total_amt_usd) > 200000 THEN 'top'
   WHEN  SUM(total_amt_usd) > 100000 THEN 'middle'
   ELSE 'low' END AS customer_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
WHERE occurred_at > '2015-12-31'
GROUP BY 1
ORDER BY 2 DESC;
```

(4)

5. We would like to identify top performing sales reps, which are sales reps associated with more than 200 orders. Create a table with the sales rep name, the total number of orders, and a column with top or not depending on if they have more than 200 orders. Place the top sales people first in your final table.

```
SELECT s.name, COUNT(*) num_ords,
   CASE WHEN COUNT(*) > 200 THEN 'top'
   ELSE 'not' END AS sales_rep_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.name
ORDER BY 2 DESC;
```

(5)

Bertelsmann Tech Scholarship - Data Track

6. The previous didn't account for the middle, nor the dollar amount associated with the sales. Management decides they want to see these characteristics represented as well. We would like to identify top performing sales reps, which are sales reps associated with more than 200 orders or more than 750000 in total sales. The middle group has any rep with more than 150 orders or 500000 in sales. Create a table with the sales rep name, the total number of orders, total sales across all orders, and a column with top, middle, or low depending on this criteria. Place the top sales people based on dollar amount of sales first in your final table. You might see a few upset sales people by this criteria!

```
SELECT s.name, COUNT(*), SUM(o.total_amt_usd) total_spent,
 CASE WHEN COUNT(*) > 200 OR SUM(o.total_amt_usd) > 750000 THEN 'top'
 WHEN COUNT(*) > 150 OR SUM(o.total_amt_usd) > 500000 THEN 'middle'
 ELSE 'low' END AS sales_rep_level
FROM orders o
JOIN accounts a
ON o.account_id = a.id
JOIN sales_reps s
ON s.id = a.sales_rep_id
GROUP BY s.name
ORDER BY 3 DESC;
```

You might see a few upset s

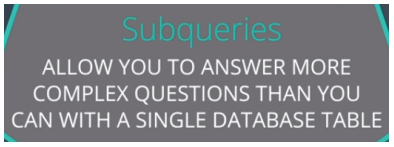
LESSON 29 SQL Subqueries & Temporary Table

29.01 Learn to query from results of another query.

This lesson will focus on three topics:

- Subqueries = inner/ nested queries
- Table Expressions
- Persistent Derived Tables

In this lesson, you will be learning to answer much more complex business questions using nested querying methods also known as subqueries.



Subqueries and table expressions are methods for being able to write a query that creates a table, and then write a query that interacts with this newly created table. Sometimes the question you are trying to answer doesn't have an answer when working directly with existing tables in database.

However, if we were able to create new tables from the existing tables, we know we could query these new tables to answer our question.

29.02 Subqueries = tool to perform operations in multiple steps

29.03 Quiz: Write Your First Subquery

- Inner q. runs 1st > then outer q. runs using result created by inner q.

```
SELECT DATE_TRUNC('day',occurred_at) AS day,
       channel,
       COUNT(*) as event_count
  FROM demo.web_events_full
 GROUP BY 1,2
 ORDER BY 1
```

29.03 Inner q. runs 1st

```
SELECT channel,
       AVG(event_count) AS avg_event_count
FROM
(SELECT DATE_TRUNC('day',occurred_at) AS day,
       channel,
       COUNT(*) as event_count
  FROM demo.web_events_full
 GROUP BY 1,2
 ) sub
  GROUP BY 1
  ORDER BY 2 DESC
```

29.03 then outer q. runs using result created by inner q.

29.05 Subquery Formatting > Well Formatted Query

```
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 day, channel, events
ORDER BY 2 DESC;
```

29.06 More On Subqueries

If only returning one value, you might use that value in a logical state like WHERE, HAVING, or even SELECT - the value could be nested within a CASE statement.

Not include alias in a subquery in a conditional state. Because subquery is treated as an individual value (or set of values in the IN case) rather than as a table.

Also the query compared a single value. If we returned an entire column IN would need to be used to perform a logical argument. If returning a table, we must use an ALIAS for table, & perform additional logic on the table.

```
SELECT *
  FROM demo.orders
  WHERE DATE_TRUNC('month',occurred_at) =
(SELECT DATE_TRUNC('month',MIN(occurred_at)) AS min_month
  FROM demo.orders )
 ORDER BY occurred_at
```

29.06 More On Subqueries

29.07 Quiz: More On Subqueries

Use **DATE_TRUNC** to pull `month` level information about the first order ever placed in the **orders** table.

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

Use the result of the previous query to find only the orders that took place in the same month and year as the first order, and then pull the average for each type of paper `qty` in this month.

```
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);
```

29.07 Quiz: More On Subqueries

Bertelsmann Tech Scholarship

29.10 Quiz: Subquery Mania

1. Provide the name of the sales_rep in each region with the largest amount of total_amt_usd sales.
2. For the region with the largest (sum) of sales total_amt_usd, how many total (count) orders were placed?
3. How many accounts had more total purchases than the account name which has bought the most standard_qty paper throughout their lifetime as a customer?

```
SELECT t3.rep_name, t3.region_name, t3.total_amt
FROM(SELECT region_name, MAX(total_amt) total_amt
     FROM(SELECT s.name rep_name, r.name region_name, SUM(o.total_amt_usd) total_amt
             FROM sales_reps s
             JOIN accounts a
             ON a.sales_rep_id = s.id
             JOIN orders o
             ON o.account_id = a.id
             JOIN region r
             ON r.id = s.region_id
             GROUP BY 1, 2) t1
     GROUP BY 1) t2
JOIN (SELECT s.name rep_name, r.name region_name, SUM(o.total_amt_usd) total_amt
     FROM sales_reps s
     JOIN accounts a
     ON a.sales_rep_id = s.id
     JOIN orders o
     ON o.account_id = a.id
     JOIN region r
     ON r.id = s.region_id
     GROUP BY 1,2
     ORDER BY 3 DESC) t3
ON t3.region_name = t2.region_name AND t3.total_amt = t2.total_amt;
```

1

29.12 WITH (CTE) state called a Common Table Expression or CTE this expressions serve exact same purpose as subqueries, as they tend to be cleaner for a future reader to follow the logic, make sure you have all the similarities between subqueries and these expressions.

```
WITH events AS (SELECT DATE_TRUNC('day',occurred_at) AS day,
       channel,
       COUNT(*) AS event_count
  FROM demo.web_events_full
 GROUP BY 1,2)

SELECT channel,
       AVG(event_count) AS avg_event_count
  FROM events
 GROUP BY 1
 ORDER BY 2 DESC
```

29.12 WITH (CTE)

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

SELECT channel, AVG(events) AS average_events
FROM events
GROUP BY channel
ORDER BY 2 DESC;
```

```
WITH table1 AS (
          SELECT *
          FROM web_events),

     table2 AS (
          SELECT *
          FROM accounts)
SELECT *
FROM table1
JOIN table2
ON table1.account_id = table2.id;
```

* When creating multiple tables using WITH, you add a comma after every table except the last table leading to your final query.
* The new table name is always aliased using table_name AS, which is followed by your query nested between parentheses.

29.14 Quiz: WITH

WITH statement performs the same task as a Subquery.

29.15 Solutions: WITH

- Subqueries and CTEs (WITH statements) > CTE is more readable, better performance.
- CTEs are more efficient, as the tables aren't recreated with each subquery portion. They are certainly more efficient from a computing perspective.

29.16 Video: Subquery Conclusion

This lesson was the first of the more advanced sequence in writing SQL. Arguably, the advanced features of Subqueries and CTEs are the most widely used in an analytics role within a company. Being able to break a problem down into the necessary tables and finding a solution using the resulting table is very useful in practice.

LESSON 30 SQL Data Cleaning

Cleaning data is an important part of the data analysis process. You will be learning how to perform data cleaning using SQL in this lesson.

30.01 Intro

- Will learn some techniques to > Clean and re-structure messy data. > Convert cols to different data types. >Tricks for manipulating NULLs.This will give you a robust toolkit to get fr raw data to clean data that's useful for analysis.

LENGTH
PULLS THE LENGTH OF A STRING

LEFT
PULL CHARACTERS FROM THE LEFT SIDE OF THE STRING AND PRESENT THEM AS A SEPARATE STRING

RIGHT
PULL FROM THE RIGHT SIDE OF THE STRING AND PRESENT AS A SEPARATE STRING

Bertelsmann Tech Scholarship

```
SELECT first_name,
       last_name,
       phone_number,
       LEFT(phone_number, 3) AS area_code,
       RIGHT(phone_number, 8) AS phone_number_only,
       RIGHT(phone_number, LENGTH(phone_number) - 4) AS phone_number_alt
  FROM demo.customer_data
```

30.02 Video: LEFT & RIGHT > Here we looked at 3 new fns: > LEFT - RIGHT - LENGTH

LEFT pulls a specified number of characters for each row in a specified col start at beginning/ from left > LEFT(phone_number, 3) // 310 > RIGHT pulls a specified number of characters for each row in a specified col starting at end/ from right > RIGHT(phone_number, 8) // 920-8993 > LENGTH provide number of characters for each row of a specified col. Use this to get len of each phone no as > LENGTH(phone_number) //

30.03 Quiz: LEFT & RIGHT

1. In the accounts table, there is a column holding the website for each company. The last three digits specify what type of web address they are using. A list of extensions (and pricing) is provided here. Pull these extensions and provide how many of each website type exist in the accounts table.

```
SELECT RIGHT(website, 3) AS domain, COUNT(*) num_companies
FROM accounts
GROUP BY 1
ORDER BY 2 DESC;
```

1

2. A debate about how much the name (or even 1st letter of a company name) matters. Use accounts table to pull 1st letter of each company name to see distribution of company names that begin w/ each letter (or number).

```
SELECT LEFT(UPPER(name), 1) AS first_letter, COUNT(*) num_companies
FROM accounts
GROUP BY 1
ORDER BY 2 DESC;
```

2

3. Use accounts table & a CASE state to create 2 groups: 1 group of company names that start w/ a number and a 2nd group of those company name that start with a letter. What proportion of company names start with a letter?

```
SELECT SUM(num) nums, SUM(letter) letters
FROM (SELECT name, CASE WHEN LEFT(UPPER(name), 1) IN ('0','1','2','3','4','5','6','7'
                        THEN 1 ELSE 0 END AS num,
             CASE WHEN LEFT(UPPER(name), 1) IN ('0','1','2','3','4','5','6','7','8','9')
                        THEN 0 ELSE 1 END AS letter
      FROM accounts) t1;
```

3

4. Consider vowels as a, e, i, o, and u. What proportion of company names start w/ a vowel, & what percent start w/ anything else?

```
SELECT SUM(vowels) vowels, SUM(other) other
FROM (SELECT name, CASE WHEN LEFT(UPPER(name), 1) IN ('A','E','I','O','U')
                        THEN 1 ELSE 0 END AS vowels,
             CASE WHEN LEFT(UPPER(name), 1) IN ('A','E','I','O','U')
                        THEN 0 ELSE 1 END AS other
      FROM accounts) t1;
```

4

30.5 Video: POSITION, STRPOS, & SUBSTR

- Learned > POSITION - STRPOS - LOWER - UPPER
- POSITION takes a character and a col, & provides the index where that character is for each row. Index' 1st position is 1 in SQL. Other language, indexing at 0. Index of a comma as POSITION(',' IN city_state).
- STRPOS provides same result as POSITION, but diff syntax for same results, STRPOS(city_state, ',').
- POSITION & STRPOS are case sensitive, so A diff > a.
- Therefore, to pull an index regardless of the case, use LOWER or UPPER to make all characters lower/ upper.

POSITION
PROVIDES THE POSITION OF A STRING COUNTING FROM THE LEFT

STRPOS
PROVIDES THE POSITION OF A STRING COUNTING FROM THE LEFT

POSITION AND STRING POSITION ARE CASE SENSITIVE

LOWER and UPPER
USED TO LOWERCASE OR CAPITALIZE ALL CHARACTERS OF A STRING

```
SELECT first_name,
       last_name,
       city_state,
       POSITION(',' IN city_state) AS comma_position,
       STRPOS(city_state,',') AS substr_comma_position,
       LOWER(city_state) AS lowercase,
       UPPER(city_state) AS uppercase,
       LEFT(city_state, POSITION(',' IN city_state) - 1) AS city
  FROM demo.customer_data
```

- 1

- 1 // no comma in city

30.06 POSITION, STRPOS, & SUBSTR - SAME DATA AS QUIZ 1

1. Use the accounts table to create first and last name columns that hold the first and last names for the primary_poc.

```
SELECT LEFT(primary_poc, STRPOS(primary_poc, ' ') -1 ) first_name,
RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name
FROM accounts;
```

1

2. Now see if you can do the same thing for every rep name in the sales_reps table. Again provide first and last name columns.

```
SELECT LEFT(name, STRPOS(name, ' ') -1 ) first_name,
       RIGHT(name, LENGTH(name) - STRPOS(name, ' ')) last_name
FROM sales_reps;
```

2

Bertelsmann Tech Scholarship

30.08 Video: CONCAT -

CONCAT

COMBINES VALUES FROM SEVERAL COLUMNS INTO ONE COLUMN

```
SELECT first_name,
       last_name,
       CONCAT(first_name,' ',last_name) AS full_name,
       first_name || ' ' || last_name AS full_name_alt
  FROM demo.customer_data
```

- Piping || // pipe character
- Will combine cols together across rows, 1st & last names stored in separate cols can combined together to create a full name: CONCAT(first_name, ' ', last_name) or with piping as first_name || ' ' || last_name.

30.09 Quiz: CONCAT

1. Each company in the accounts table wants to create an email address for each primary_poc. The email address should be the first name of the primary_poc . last name primary_poc @ company name .com.

```
WITH t1 AS (SELECT LEFT(primary_poc,    STRPOS(primary_poc, ' ') -1 ) first_name,
                   RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name, name
            FROM accounts)
SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@', name, '.com')
FROM t1;
```

2. You may have noticed that in the previous solution some of the company names include spaces, which will certainly not work in an email address. See if you can create an email address that will work by removing all of the spaces in the account name, but otherwise your solution should be just as in question 1. Some helpful documentation is here.

```
WITH t1 AS (SELECT LEFT(primary_poc,    STRPOS(primary_poc, ' ') -1 ) first_name,
                   RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name, name
            FROM accounts)
SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@', REPLACE(name, ' ', ''), '.com')
FROM t1;
```

3. We would also like to create an initial password, which they will change after their first log in. The first password will be the first letter of the primary_poc's first name (lowercase), then the last letter of their first name (lowercase), the first letter of their last name (lowercase), the last letter of their last name (lowercase), the number of letters in their first name, the number of letters in their last name, and then the name of the company they are working with, all capitalized with no spaces.

```
WITH t1 AS (SELECT LEFT(primary_poc,    STRPOS(primary_poc, ' ') -1 ) first_name,
                   RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name, name
            FROM accounts)
SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@', name, '.com'),
       LEFT(LOWER(first_name), 1) || RIGHT(LOWER(first_name), 1) || LEFT(LOWER(last_name), 1) ||
       RIGHT(LOWER(last_name), 1) || LENGTH(first_name) || LENGTH(last_name) || REPLACE(UPPER(name), ' ', '')
FROM t1;
```

30.11 Video: CAST

DATE_PART('month', TO_DATE(month, 'month'))

CAST

ALLOWS US TO CHANGE COLUMNS FROM ONE DATA TYPE TO ANOTHER

BOTH 'CAST' AND '::' ALLOW FOR THE CONVERTING OF ONE DATA TYPE TO ANOTHER

LEFT, RIGHT, OR SUBSTRING AUTOMATICALLY CAST DATA TO A STRING DATA TYPE

- additional functionality for work w/ dates > TO_DATE > CAST > Casting with `::`

DATE_PART('month', TO_DATE(month, 'month')) changed a month name into number associated with that particular month.

- change a string to a date using CAST. CAST change column types.

CAST(date_column AS DATE), or can use date_column::DATE

* LEFT, RIGHT, and TRIM are all used to select only certain elements of strings, but using them to select elements of a number or date will treat them as strings for the purpose of the function.

- TRIM used to remove characters from the beginning and end of a string, or unwanted spaces at the beginning or end of a row that often happen with data being moved from Excel or other storage systems.

30.12 Quiz: CAST

`4.` Write a query to change the date into the correct SQL date format.

You will need to use at least **SUBSTR** and **CONCAT** to perform this operation.

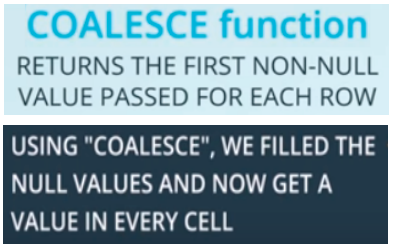
`5.` Once you have created a column in the correct format, use either `CAST` or `::` to convert this to a date.

```
SELECT date orig_date,
(SUBSTR(date, 7, 4) || '-' || LEFT(date, 2) || '-' || SUBSTR(date, 4, 2)) new_date
FROM sf_crime_data;
```

```
SELECT date orig_date,
(SUBSTR(date, 7, 4) || '-' || LEFT(date, 2) || '-' || SUBSTR(date, 4, 2))::DATE new_date
FROM sf_crime_data;
```

Bertelsmann Tech Scholarship

30.14 Video: COALESCE > returns the first non-NULL value passed for each row.
Hence why the video used no_poc if the value in the row was NULL.




```
SELECT *,
       COALESCE(primary_poc, 'no POC') AS primary_poc_modified
  FROM demo.accounts
 WHERE primary_poc IS NULL
```

website	lat	long	primary_poc	sales_rep_id	primary_poc_modified
www.intel.com	41.03153857	-74.66846407		321580	no POC
www.delta.com	40.75860903	-73.99067048		321510	no POC

```
SELECT COUNT(primary_poc) AS regular_count,
       COUNT(COALESCE(primary_poc, 'no POC')) AS modified_count
  FROM demo.accounts
```

	regular_count	modified_count
1	345	354

30.17 Video + Text: Recap
- Learn from what you have done in solving previous problems to solve new problem.
* SQL functions > https://www.w3schools.com/sql/
* Using SQL String Functions to Clean Data > https://mode.com/sql-tutorial/sql-string-functions-for-cleaning/

```
SELECT *,
       DATE_PART('month',TO_DATE(month, 'month')) AS clean_month,
       year || '-' || DATE_PART('month',TO_DATE(month, 'month')) || '-' || day AS concatenated_date,
       CAST(year || '-' || DATE_PART('month',TO_DATE(month, 'month')) || '-' || day AS date) AS formatted_date,
       (year || '-' || DATE_PART('month',TO_DATE(month, 'month')) || '-' || day)::date AS formatted_date_alt
  FROM demo.ad_clicks
```

FINISH SQL