SQL Functions and Filtering

Up until now, we have manipulated just numbers, what about strings?

Concatenating Strings: How to work with strings in SQL?

There are also **inline functions** that can be used to modify string values in SQL, as well.

Notice, our customer table has separate columns for each customer's first and last names.

Let's quickly look at the customer table.

```
SELECT *
FROM farmers_market.customer
LIMIT 5
```

Question: We want to merge each customer's name into a single column that contains the first name, then a space, and then the last name.

CONCAT()

- We can accomplish that by using the CONCAT() function.
- The list of string values you want to merge together are entered into the CONCAT() function as parameters.
- A space can be included by surrounding it with quotes.

```
SELECT
customer_id,
```

```
CONCAT(first_name, " ", last_name) AS full_name
FROM farmers_market.customer
LIMIT 5
```

Question: In a customer's full_name, we want the **first_name to be in upper case** followed by the last_name as it is.

UPPER() is a function that capitalizes string values.

- It's also possible to nest functions inside other functions, which the SQL interpreter executes **from the "inside" to the "outside."**
- We can enclose the UPPER() function inside the CONCAT() function to uppercase the full name.

```
SELECT
customer_id,
CONCAT(UPPER(first_name), " ", last_name) AS full_name
FROM farmers_market.customer
LIMIT 5
```

Similar to the UPPER() function, we also have a **LOWER()** function in case we want to turn some string value into lower case.

Question: What if we only want to display a subset of a string?

For that, we can use a function called **SUBSTR()**.

Syntax: SUBSTR(value, position, length)

- **position** tells us which character to start from.
- **length** tells us up to which character we should include.

Say we want to display only the first character of someone's last_name?

```
SELECT
customer_id,
CONCAT(UPPER(first_name), " ", SUBSTR(last_name, 1, 1) AS full_name
FROM farmers_market.customer
LIMIT 5
```

Question: We have a customer with first_name "**john**" and last_name "**doe**". How can we properly format his name by capitalizing the first letter of the first & last name while keeping all other letters in lowercase?

```
SELECT
CONCAT(UPPER(SUBSTR("john", 1, 1)),
LOWER(SUBSTR("john", 2)), " ",
UPPER(SUBSTR("doe", 1, 1)),
LOWER(SUBSTR("doe", 2))) AS full_name;
```

In BigQuery, we also have the **INITCAP()** function that we can use.

It takes a STRING and returns it with the first character in each word in uppercase and all other characters in lowercase.

Please keep in mind that INITCAP() is exclusive to BigQuery only. It is not available in MySQL.

```
SELECT
CONCAT(INITCAP("john")," ", INITCAP("doe")) AS full_name;
```

Question: Extract all the product names that are part of product category 1

Filtering data - The **WHERE** Clause

- The WHERE clause is the part of the SELECT statement in which you list conditions that are used to determine which rows in the table should be included in the results set.
- In other words, the WHERE clause is used for filtering of records.

```
SELECT
product_id, product_name, product_category_id
FROM farmers_market.product
WHERE product_category_id = 1
LIMIT 5
```

Similarly, what if we want all the product names that are part of any other product category except 1?

We can do this using the **!=** or **<>** (Not equal to) operator.

```
SELECT
product_id, product_name, product_category_id
FROM farmers_market.product
WHERE product_category_id <> 1
LIMIT 5
```

Order of Execution of a SQL query:

- **FROM** - The database gets the data from tables in FROM clause.
- **WHERE** - The data is filtered based on the conditions specified in the WHERE clause. Rows that do not meet the criteria are excluded.
- **SELECT** - It determines which columns to include in the final result set.

- **ORDER BY** - It allows you to sort the result set based on one or more columns, either in ascending or descending order.
- **OFFSET** - The specified number of rows are skipped from the beginning of the result set.
- **LIMIT** - After skipping the rows, the LIMIT clause is applied to restrict the number of rows returned.

Question: Print a report of everything the customer_id 4 has ever purchased at the market, sorted by date. Add total_amt column as well for each purchase.

```
SELECT
customer_id, market_date, quantity,
cost_to_customer_per_qty,
ROUND(quantity * cost_to_customer_per_qty, 2) AS total_amt
FROM `farmers_market.customer_purchases`
WHERE customer_id = 4
ORDER BY market_date ASC
```

Question: Get all the product info for products with id between 3 and 8 (not inclusive) and of products with id 10.

Filtering on multiple conditions - using operators - **“AND”**, **“OR”**, **“NOT”**

- Conditions with OR between them will jointly evaluate to TRUE, meaning the row will be returned, if any of the clauses are TRUE.
- Conditions with AND between them will only evaluate to TRUE in combination if all of the clauses evaluate to TRUE. Otherwise, the row will not be returned.
- **Remember that NOT flips the following boolean value to its opposite (TRUE becomes FALSE, and vice versa).**

According to the question, we need “>3 and <8” → 4, 5, 6, 7, 10
1. Product_id > 3
2. Product_id < 8
3. Product_id = 10

```
SELECT *
FROM `farmers_market.product`
WHERE (product_id > 3 AND product_id < 8) OR product_id =10
```

Question: Find the booth assignments for vendor_id 7 for all dates between April 3, 2019 and May 16, 2019, including the 2 dates.

```
SELECT *
FROM `farmers_market.vba`
WHERE vendor_id = 7 AND
(market_date >= "2019-04-03" AND market_date <= "2019-05-18")
```

Whenever you have such range based questions, you can use the **BETWEEN** keyword.

```
SELECT *
FROM `farmers_market.vba`
WHERE vendor_id = 7 AND
(market_date BETWEEN "2019-04-03" AND "2019-05-18")
```

NOTE:
Between is inclusive of the upper & lower limit of the range specified.

Question: Return a list of customers with the following last names: [Diaz, Edwards, Wilson]

To solve this question.. we could use a long list of OR comparisons.

```
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
 FROM farmers_market.customer
 WHERE
    customer_last_name = 'Diaz'
    OR customer_last_name = 'Edwards'
    OR customer_last_name = 'Wilson'
```

Alternative approach -

Whenever we have a list of values to filter from, we can use the **IN** keyword and provide a comma-separated list of values to compare against.

```
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
 FROM farmers_market.customer
 WHERE
    customer_last_name IN ('Diaz' , 'Edwards', 'Wilson')
```

In this case the **IN** keyword will return TRUE for any row with a customer_last_name that is in the provided list.

Question: What if we want a list of all customers except those with the last names: [Diaz, Edwards, Wilson]

In such a case, we can use the **NOT** operator along with the IN keyword to exclude the customers having last names as 'Diaz' , 'Edwards' or 'Wilson'.

```
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
 FROM farmers_market.customer
 WHERE
    customer_last_name NOT IN ('Diaz' , 'Edwards', 'Wilson')
```

Please note that the last names inside the IN clause are case sensitive.

So for words like diaz, EDWARDS, WiLSoN, the IN clause will return False.
If we want to include the last names despite their case, we can do...

```
SELECT
    customer_id,
    customer_first_name,
    customer_last_name
 FROM farmers_market.customer
 WHERE
    LOWER(customer_last_name) IN ('diaz' , 'edwards', 'wilson')
```

OR use `UPPER(customer_last_name) IN ('DIAZ' , 'EDWARDS', 'WILSON')`

Question: You want to get data about a customer you knew as Jerry but you are not sure if they are listed as Jeremy or Jeremiah or Jerry. Get all customers whose name starts with “jer”.

In SQL, you can search for **partially matched strings** using a **comparison operator** called **LIKE**, and **wildcard characters**, which serve as a placeholder for unknown characters in a string.

Wildcards :

- **%** - stand-in for 0 or more characters.
- **_** (underscore) - stand-in for one and only one character

```
SELECT *
FROM `farmers_market.customer`
WHERE lower(customer_first_name) LIKE "jer%"
```

Here are some examples showing different `LIKE` operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a__%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"
