SQL Data Exploration

This database was used to explore different functional aspects of SQL from basic to advanced. This short project is to help anyone to setup their own local environment and practice querying.

Database used:

The database consists of different tables linked with a database schema. If you want to create a database from scratch, you can use these excel files and set it up.

Practice Querying

Basic

LIKE

1. '%C' and '%c' would give the same result as using LIKE is case-insensitive.
2. If the results should contain characters at a specific position, use _t%. For instance, this gives all the entries with t as the second character.

Use the accounts table to find

1. All the companies whose names start with 'C'.
2. All companies whose names contain the string 'one' somewhere in the name.
3. All companies whose names end with 's'.

Check queries Basic Like using this file queries_Like

IN

1. Use the accounts table to find the account name, primary_poc, and sales_rep_id for Walmart, Target, and Nordstrom.
2. Use the web_events table to find all information regarding individuals who were contacted via the channel of organic or adwords.

Check queries Basic IN using this file queries_IN

INTO

To save the outcome into a table - Ex: SELECT x, y, z INTO table FROM ...

Joins

Definitions

1. A **semi-join** returns the rows of the first table where it can find a match in the second table.

Trivia:

1. JOIN keywords works for INNER JOIN. It is the default join in SQL.

Questions

1. Provide a table for all web_events associated with account name of Walmart. There should be three columns. Be sure to include the primary_poc, time of the event, and the channel for each event. Additionally, you might choose to add a fourth column to assure only Walmart events were chosen.

2. Provide a table that provides the region for each sales_rep along with their associated accounts. Your final table should include three columns: the region name, the sales rep name, and the account name. Sort the accounts alphabetically (A-Z) according to account name.

3. Provide the name for each region for every order, as well as the account name and the unit price they paid (total_amt_usd/total) for the order. Your final table should have 3 columns: region name, account name, and unit price. A few accounts have 0 for total, so I divided by (total + 0.01) to assure not dividing by zero.

Check queries JOIN using this file queries_JOIN

Aggregations

DISTINCT

1. Use DISTINCT to test if there are any accounts associated with more than one region.
2. Have any sales reps worked on more than one account?

Check queries DISTINCT using this file queries_Agregations DISTINCT

MIN, MAX, AVG

let's talk about the MIN() function. It returns the smallest value in a set of values. The values can come from a numeric column or as the result of an expression returning a numeric value. (Note: The column can come from a table or a view.) It is a scalar function that returns one numeric value.

The syntax of the MIN() function is presented below : MIN(column_or_expression)

Like MIN(), MAX() is an aggregate function that returns a numeric value from a set. The difference is that it returns the largest (maximum) value. The values can come from a column or as the result of an expression that returns a numeric value or values. It is a scalar function that returns one value.

Here is the syntax of the MAX() function: MAX(column_or_expression)

The AVG() aggregate function returns the mean average of the numeric values in the specified column. For example, SELECT AVG(column_or_expression)

Check queries MIN, MAX, AVG using this file queries_Agregations MIN,MAX,AVG

HAVING

1. How many of the sales reps have more than 5 accounts that they manage?
2. How many accounts have more than 20 orders?
3. Which account has the most orders?
4. Which accounts spent more than 30,000 usd total across all orders?
5. Which accounts spent less than 1,000 usd total across all orders?
6. Which account has spent the most with us?
7. Which account has spent the least with us?
8. Which accounts used facebook as a channel to contact customers more than 6 times?
9. Which account used facebook most as a channel?
10. Which channel was most frequently used by most accounts?

Trivia:

1. ORDER BY can have the new variable name used
2. HAVING doesn't allow variable name rather it allows aggregation function with condition.
3. HAVING works only where there is an aggregation column (Ex: SUM(orders)...HAVING orders > 999) in the table/view built.

Check queries HAVING using this file queries_Agregations HAVING

GROUP BY

The Group By statement is used to group together any rows of a column with the same value stored in them, based on a function specified in the statement. Generally, these functions are one of the aggregate functions such as MAX() and SUM(). This statement is used with the SELECT command in SQL.
The SQL Group By statement uses the split-apply-combine strategy.

- Split: The different groups are split with their values.
- Apply: The aggregate function is applied to the values of these groups.
- Combine: The values are combined in a single row.

Check queries GROUP BY using this file queries_Agregations GROUP BY

DATE FUNCTION

The most difficult part when working with dates is to be sure that the format of the date you are trying to insert, matches the format of the date column in the database.

As long as your data contains only the date portion, your queries will work as expected. However, if a time portion is involved, it gets complicated.

Before talking about the complications of querying for dates, we will look at the most important built-in functions for working with dates.

Check queries DATE using this file queries DATE FUNCTION

CASE WHEN

CASE() function in MySQL is used to find a value by passing over conditions whenever any condition satisfies the given statement otherwise it returns the statement in an else part. However, when a condition is satisfied it stops reading further and returns the output.

Features:

- This function returns the statement in the else part if none of the stated conditions are true.

- This function returns NULL if none of the stated conditions are true as well as there is no else part also.
- This function comes under Advanced Functions.
- This function accepts two parameters namely conditions and results.

Check queries CASE WHEN using this file queries CASE WHEN

Data Cleansing

LEFT

LEFT() This function is used to take the character of a string with a certain length starting from the left. Suppose we want to display nis from the 3 left most characters

select LEFT(nis,3) as nis,name,address,gender,website from student

RIGHT() The RIGHT function is almost the same as the left function, this function takes the character of a string with a certain length starting from the right Suppose we want to display the nis of the 2 righ tmost characters

select RIGHT(nis23) as nis,name,address,gender, student website

COALESCE() function is used for returning the first non-null value in a list of expressions. If all the values in the list evaluate to NULL, then the COALESCE() function returns NULL.
The COALESCE() function accepts one parameter which is the list which can contain various values. The value returned by the MySQL COALESCE() function is the first non-null value in a list of expressions or NULL if all the values in a list are NULL.

Check queries LEFT, RIGHT, COALESCE using this file queries DATA CLEANSING

Window Functions

ROW_NUMBER & RANK

1. Ranking Total Paper Ordered by Account: Select the id, account_id, and total variable from the orders table, then create a column called total_rank that ranks this total amount of paper ordered (from highest to lowest) for each account using a partition. Your final table should have these four columns.

Check queries and ROW_NUMBER & RANK using this file windowfunctions-RANK

Advanced Joins & Performance Tuning

UNION

1. Write a query that uses UNION ALL on two instances (and selecting all columns) of the accounts table.
2. Add a WHERE clause to each of the tables that you unioned in the query above, filtering the first table where name equals Walmart and filtering the second table where name equals Disney.
3. Perform the union in your first query (under the Appending Data via UNION header) in a common table expression and name it double_accounts. Then do a COUNT the number of times a name appears in the double_accounts table. If you do this correctly, your query results should have a count of 2 for each name.

Check queries UNION using this file advanced_join(union)