SQL for Data Science Bootcamp

Final Project

Congratulations! You have completed this bootcamp and qualified to take this final project examination.

In this final project, you are asked to work mainly on two different datasets.

Method:

- You are going to provide your SQL queries in a written format in a docs like file for each question.
- You can use BigQuery to try your queries before summiting your answers.
- When you write or try your queries, you need project name, dataset name and table name. You can call you project name as what ever you would like to, but please create a dataset under your project, and give **final_project** name to this dataset. Then create your tables under it.

Question 1:

You have three tables:

Customers table:
https://drive.google.com/file/d/17oRPW3jA2mCsTOXKKt6Xx7wSgD78twq8/view?usp=sharing

Orders table:
https://drive.google.com/file/d/1TWBB9rqGRnWOJsDcwkUE2HkJNEafawQm/view?usp=sharing

Payments table
https://drive.google.com/file/d/1vakf_7F9cb7ZKOeCofyI370Paiswg3JX/view?usp=sharing

A. Which payment_type has the fastest average delivery day where delivery day can be found by using order_delivered_customer_date and order_purchase_timestamp columns? You should be careful about choosing these dates that shouldn't be null.
(Hint: The result should show only one payment_type with the average_delivery_day info)

B. What is the maximum payment_value of each state where order status is neither unavailable nor canceled? Additionally, the max_payment_revenue column should have 1 decimal point.
(Hint: The result table will only have state and max_payment_revenue columns)

C. In the payments table, how many distinct order_ids that exist more than once?
(Hint: Try window functions!)

D. For each row, find the payment_value of previous row and leading row in the partition of order_id and order of payment_sequential ascendingly.
(Hint: Again window functions!
The result table should have all columns in the payments table and two new columns: previous_payment_value and leading_payment_value.
Also, we don't want null values for these two new fields)

E. Get all the columns in the customers table and add three new columns:
   * order_id
   * order_purchase_date (Hint: convert the timestamp to the date value)
   * payment_value
   * payment_value_int (Hint: payment_value but in the integer format. You should use a function to convert it type)
   (In this join, we want all the rows from customers table, and if there is corresponding data in either orders or payments, we can have it. But we definitely want to have all info from customers table. Think about your join!)

F. Get all columns from the payments table excluding the payment_type column, and create payment_method column instead of that. This new column should have this format:

   If payment_type is “not_defined”, then payment_method will be 0
   If payment_type is “credit_card”, then payment_method will be 1
   If payment_type is “voucher”, then payment_method will be 2
   If payment_type is “debit_card”, then payment_method will be 3
   If payment_type is “boleto”, then payment_method will be 4

Question 2:

You have one table:

Netflix_movies: https://drive.google.com/file/d/1VRdGqa3KmgDj_BEX4_OBEtDa-JRmE1LH/view?usp=sharing

A. How many times each word is used in the table, that you get when you split the movie_name by each space?
   (Hint: The result table only have 2 columns: words_used_in_movie_names, number_of_occurence.
   Ex. You can imagine that asks for how many times you see Rome word in this table)

B. If a movie_name has “**Vol. 1**”, then change it to “Part 1”; if has “**Vol. 2**”, then change it to “Part. 2”. Else movie_name. After these changes, name show this column as “movie_name” instead of showing the original movie_name column.

C. In this table, we have some movie_names that occur multiple times. Create a new column in the result table, call this column as “**previous_year**” and show the previous year of each movie if they exist multiple times in that table. In the result table, there shouldn’t be any null previous_year values.