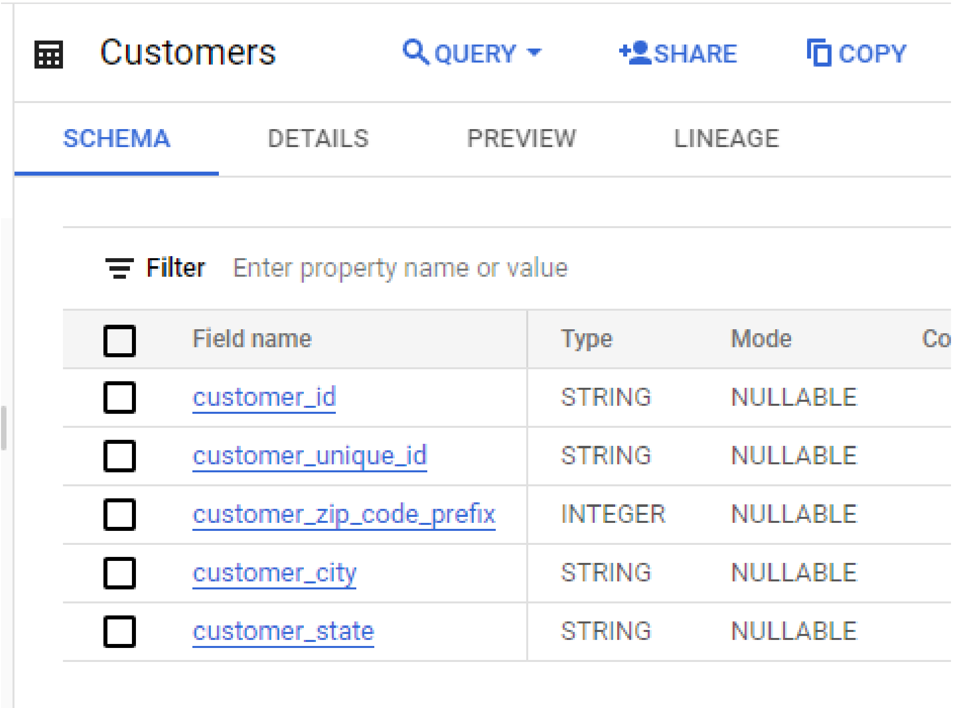
PROJECT_TARGET_SQL_01

- ***By Nishtha Nagar***

1. (Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset);

#(1. Data type of columns in a table);

```
select*
from `target_retail_dataset.customers`;
```



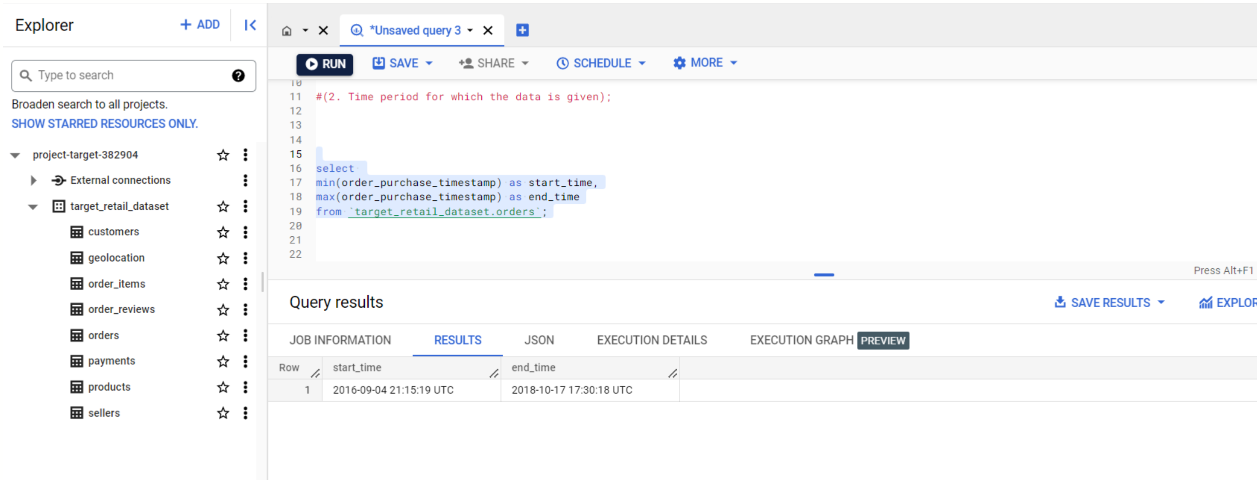
Field name	Type	Mode	Co
customer_id	STRING	NULLABLE	
customer_unique_id	STRING	NULLABLE	
customer_zip_code_prefix	INTEGER	NULLABLE	
customer_city	STRING	NULLABLE	
customer_state	STRING	NULLABLE	

#(2. Time period for which the data is given);

Methodology: Use of min and max function to show the earliest and latest purchase timestamps in the orders table, which can be used to determine the range of time for which the orders data is available.

Query Code:

```
select
min(order_purchase_timestamp) as start_time,
max(order_purchase_timestamp) as end_time
from `target_retail_dataset.orders`;
```



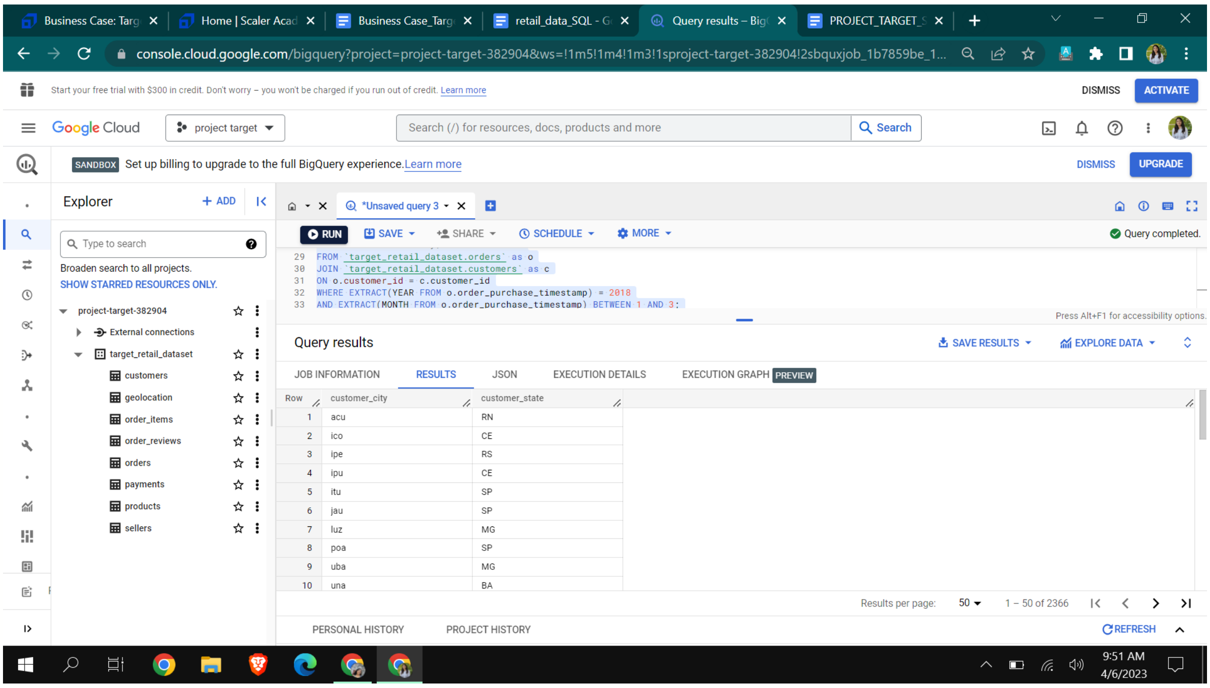
Insights:
Start_time: 2016-09-04 21:15:19 UTC
End_time: 2018-10-17 17:30:18 UTC

#(3. Cities and States of customers ordered during the given period);

Methodology: The JOIN clause is used to join the two tables based on the customer_id column, which is present in both tables.
The DISTINCT keyword is used to ensure that the resulting rows are unique.
Thus, the query retrieves a list of unique customer locations for orders placed during the first quarter of 2018 from January to March.

Query Code:

```
SELECT
DISTINCT customer_city, customer_state
FROM `target_retail_dataset.orders` as o
JOIN `target_retail_dataset.customers` as c
ON o.customer_id = c.customer_id
WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 3;
```



---

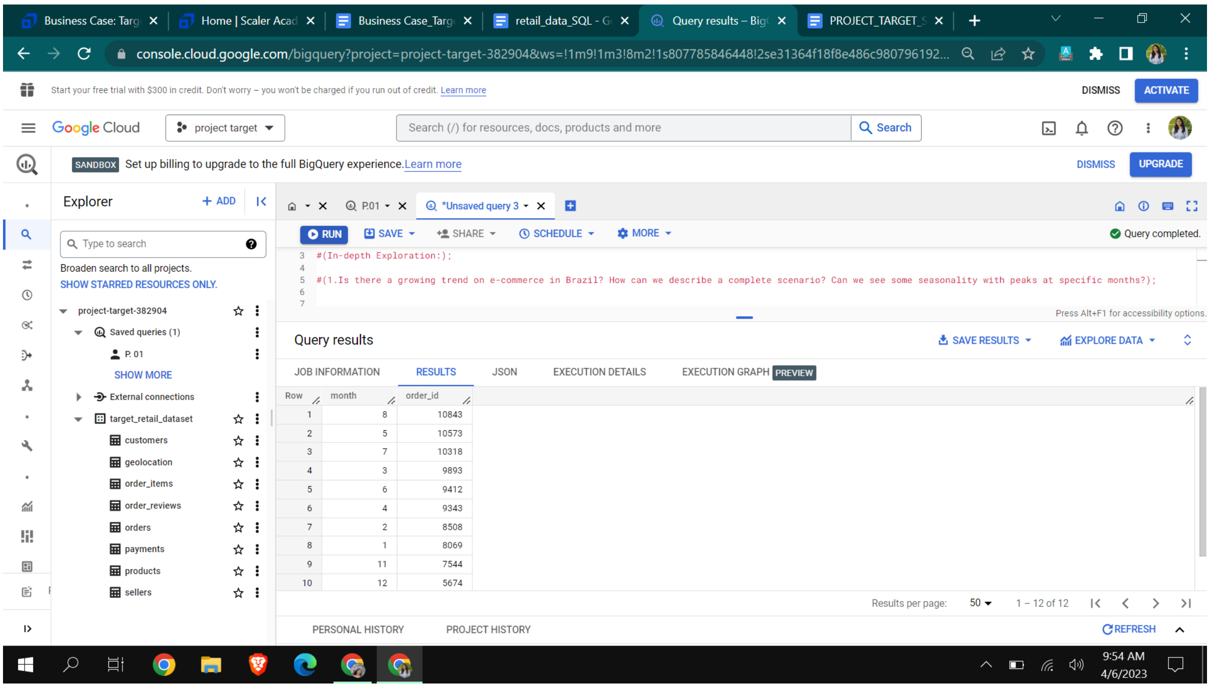
2. (In-depth Exploration:);

#(1.Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?);

Methodology: Selecting the month and the count of order_id from the table and grouping the data by month. The resulting output will show the number of orders made each month, sorted in descending order by the count of order_id.

Query Code:

```
SELECT
 extract(month from order_purchase_timestamp) as month,
 count(order_id) as order_id
from `target_retail_dataset.orders`
group by
 month
order by
 order_id desc;
```



Insights:

The Number of orders is highest in the months of August and May.

Assumptions:

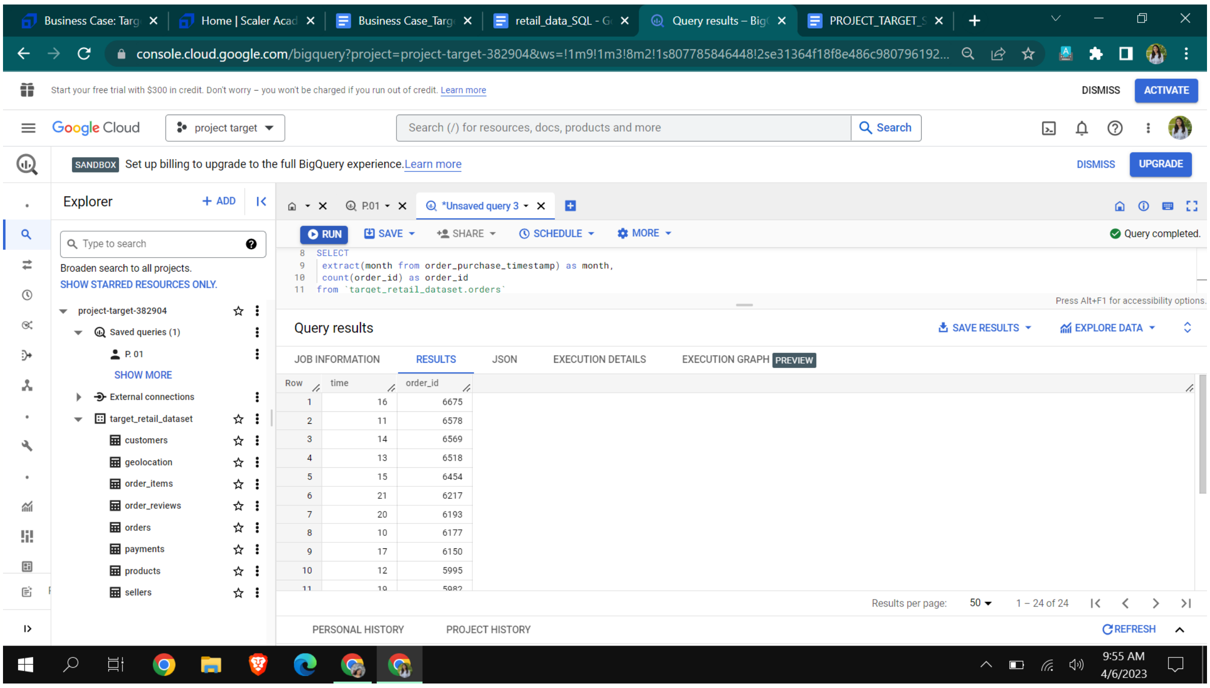
The reason can be anything like, national holiday, or end of winter sales.

#(2.What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night?);

Methodology: To analyse the purchasing behaviour of Brazilian customers in terms of the time of day they tend to buy products, we will group the orders by the hour of the day they were placed and count the number of orders in each group.

Query Code:

```
SELECT
 extract(hour from order_purchase_timestamp) as time,
 count(order_id) as order_id
from `target_retail_dataset.orders`
group by
 time
order by
 order_id desc;
```



Insights:
Brazilians tend to buy in the afternoon and morning time.

Recommendations: Doesn't seem any scope of recommendations.

---

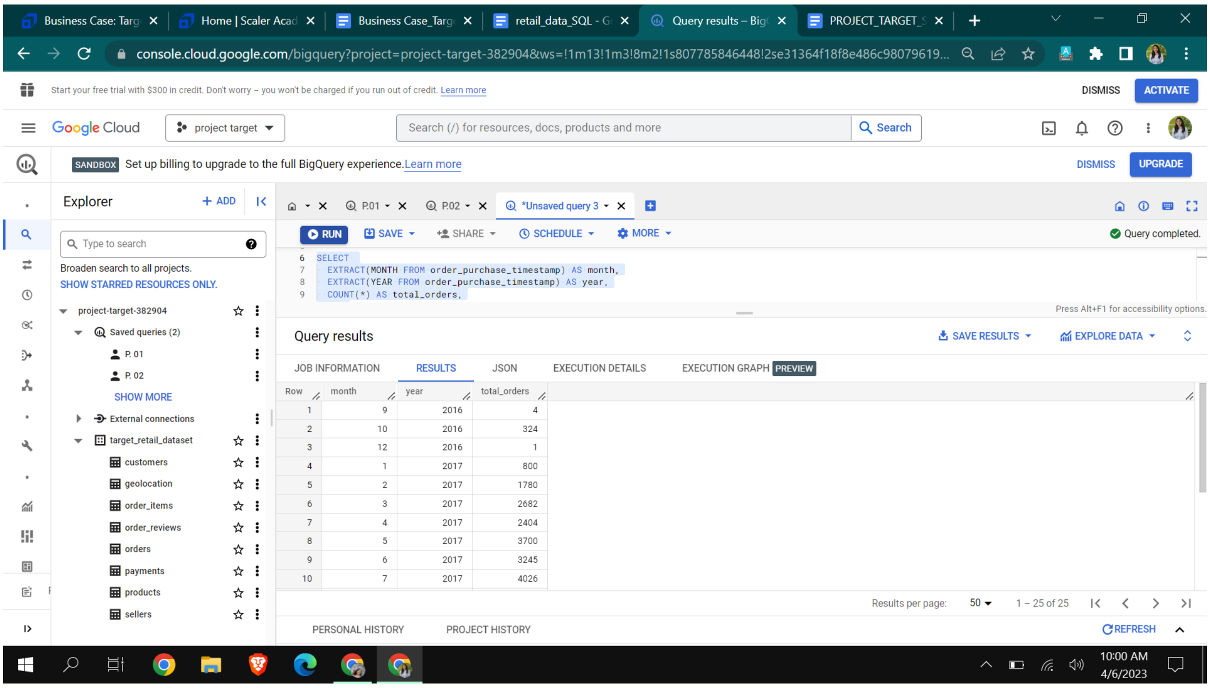
3. Evolution of E-commerce orders in the Brazil region:

#1.Get month on month orders by states

Methodology:
To solve this problem, we will extract the month and year from the order_purchase_timestamp column using the EXTRACT function, and then count the number of orders placed in each month using the COUNT function. We will also use GROUP BY clause to group the data by year and month so that the results are aggregated by month. Finally, we will sort the results in ascending order by year and month using the ORDER BY clause.

Query Code:
```
SELECT
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  COUNT(*) AS total_orders,
FROM
  `target_retail_dataset.orders`
```

```sql
GROUP BY
  year,
  month
ORDER BY
  year,
  month;
```



Insights: No aggregated insight can be provided.

Recommendations: Doesn't seem any scope of recommendations.

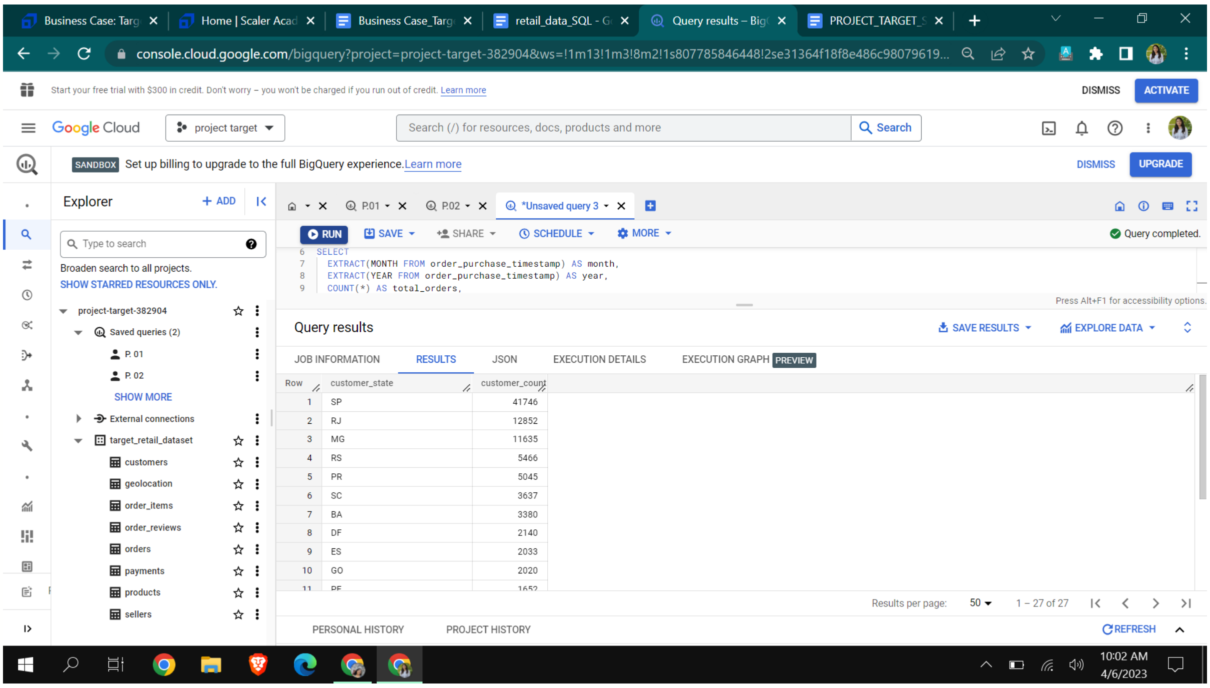
02. Distribution of customers across the states in Brazil

Methodology:

To find the number of customers for each state in Brazil, we will use the columns "customer_state" and "customer_id" from the "customers" table and groups the data by "customer_state". Then, we will use the "COUNT" function to count the number of distinct "customer_id" values for each state.

Query Code:

```sql
SELECT customer_state,
 COUNT(DISTINCT customer_id) as customer_count
FROM `target_retail_dataset.customers`
GROUP BY customer_state
ORDER BY customer_count DESC;
```



Insights:
SP (Sao Paulo) state of brazil has the highest count of customers, while the RR (Roraima) state of Brazil has the lowest count of customers.

Recommendations: Doesn't seem any scope of recommendations.

---

4. Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight and others.

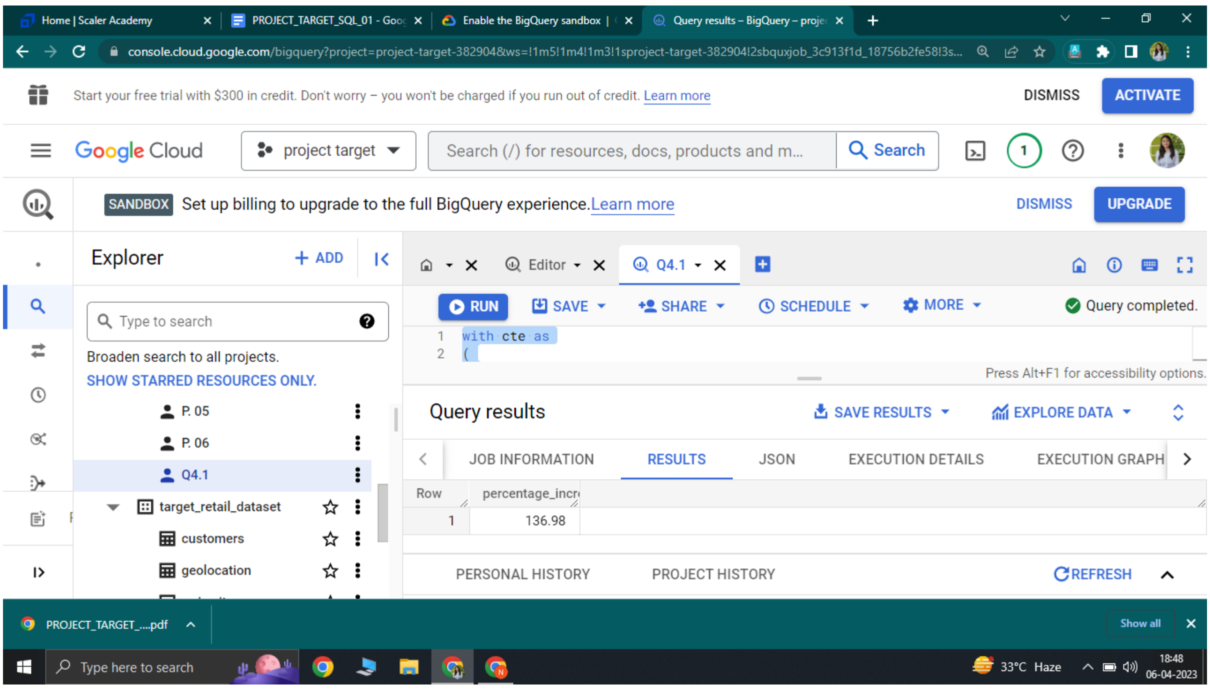
1. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use the "payment_value" column in the payments table.

Methodology:
We will first join the "payments" table with the "orders" table using the order_id column. We will then extract the year and month from the order_purchase_timestamp column and filter for only the months between January and August in 2017 and 2018. Next, we will group the results by year and calculate the sum of the payment values for each year.

To calculate the percentage increase in cost, we will use the lead function to compare the payment values between 2017 and 2018. Finally, we will calculate the percentage increase by dividing the difference in payment values by the previous year's payment value and multiplying by 100.

Query Code:

```
with cte as
(
select
((x.payment_value -
lead(x.payment_value) over( order by year desc))/lead(x.payment_value) over( order
by year
desc)) * 100 as percentage_increase
from
( select sum(p.payment_value) payment_value,
extract(year from o.order_purchase_timestamp) as year
from
`target_retail_dataset.payments`as p
join `target_retail_dataset.orders`as o
on p.order_id = o.order_id
where extract(year from o.order_purchase_timestamp) in (2017,2018)
and extract(month from o.order_purchase_timestamp) between 01 and 08
group by extract(year from o.order_purchase_timestamp)) x
)
select round(percentage_increase,2) as percentage_increase
from cte
where cte.percentage_increase is not null
```



Insights: 136.98%

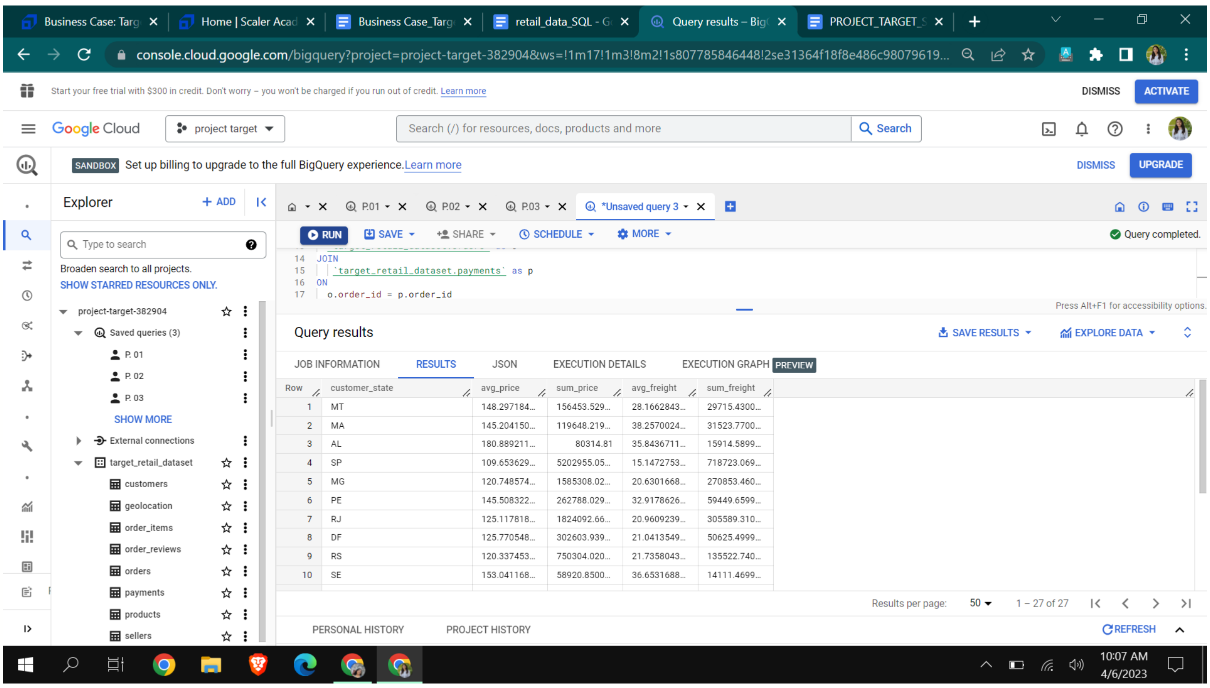
Recommendations: adjusting prices or promotions, investing in marketing campaigns to increase sales, and analysing the data further to identify any patterns or trends that could inform future business strategies.

2. Mean & Sum of price and freight value by customer state

Methodology:
In this method, we will retrieve the average and sum of the price and freight value for each customer state by joining three tables - "orders", "order_items" and "customers". We will use JOIN statements to combine the three tables based on their common IDs. It then groups the resulting data by customer state using the GROUP BY clause.

Query Code:

```
SELECT
customer_state,
AVG(price) AS avg_price,
SUM(price) AS sum_price,
AVG(freight_value) AS avg_freight,
SUM(freight_value) AS sum_freight
FROM `target_retail_dataset.orders` as o
JOIN `target_retail_dataset.order_items` as oi
ON o.order_id = oi.order_id
JOIN `target_retail_dataset.customers` as c
ON o.customer_id = c.customer_id
GROUP BY customer_state;
```

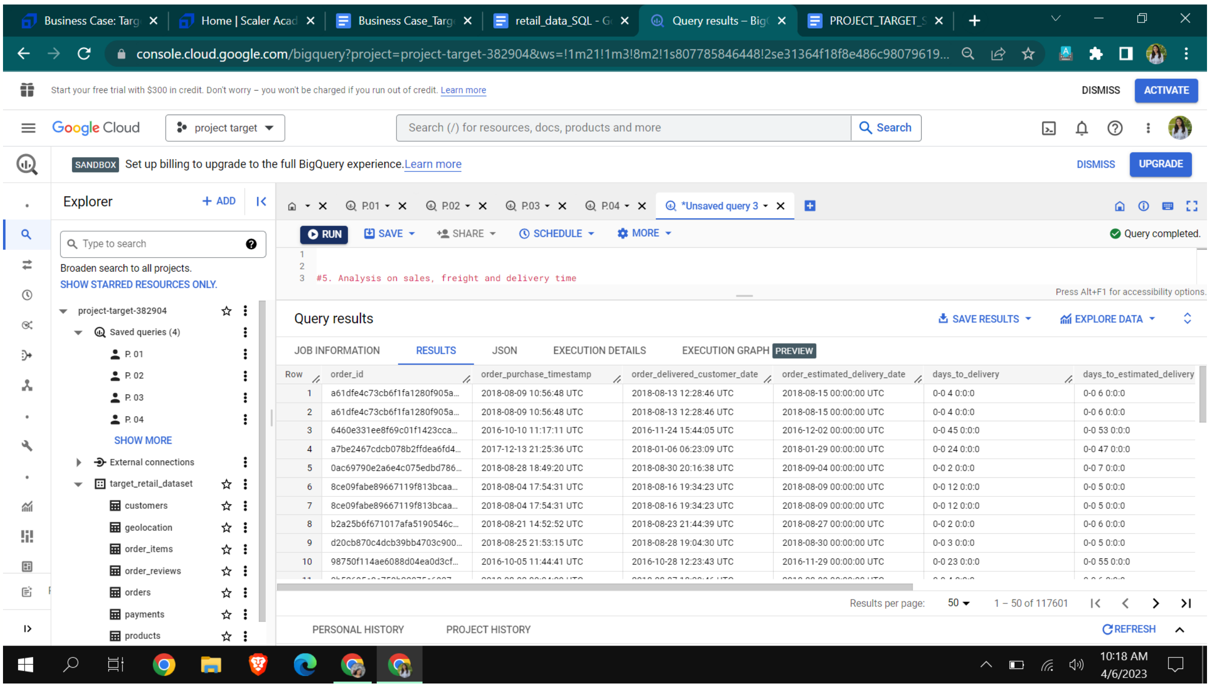


#5. Analysis on sales, freight and delivery time

1.Calculate days between purchasing, delivering and estimated delivery

Methodology:
Joining the 'orders', 'order_items', 'payments', 'customers', and 'sellers' tables to get relevant information. We need to calculate the number of days between the order purchase timestamp and the actual delivery date and estimated delivery date using the EXTRACT and DATE functions. It also selects the payment value and customer location information such as the city and state.

Query Code:
```
SELECT
  o.order_id,
  o.order_purchase_timestamp,
  o.order_delivered_customer_date,
  o.order_estimated_delivery_date,
  EXTRACT(DATE FROM o.order_delivered_customer_date) - EXTRACT(DATE FROM
o.order_purchase_timestamp) AS days_to_delivery,
  EXTRACT(DATE FROM o.order_estimated_delivery_date) - EXTRACT(DATE FROM
o.order_purchase_timestamp) AS days_to_estimated_delivery,
  p.payment_value,
  c.customer_city,
  c.customer_state,
FROM
   `target_retail_dataset.orders` as o
JOIN
   `target_retail_dataset.order_items` as oi
  ON o.order_id = oi.order_id
JOIN
   `target_retail_dataset.payments` as p
  ON o.order_id = p.order_id
JOIN
   `target_retail_dataset.customers` as c
  ON o.customer_id = c.customer_id
JOIN
   `target_retail_dataset.sellers` s
  ON oi.seller_id = s.seller_id;
```



Recommendations:

The results of this query can be used to analyse the delivery performance of the company and identify areas where improvements can be made to optimise delivery times and reduce customer complaints.

#(2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

#time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

#diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date)

Methodology:

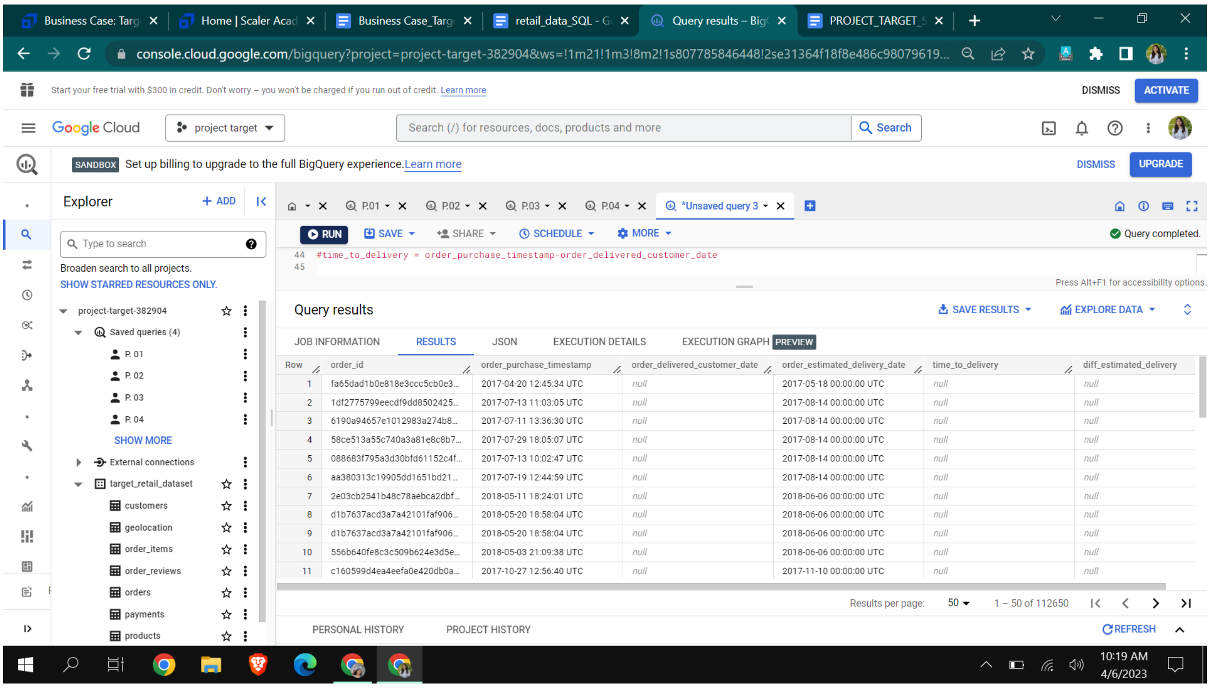
To calculate the "time_to_delivery" variable, we need to subtract the "order_delivered_customer_date" from the "order_purchase_timestamp" for each order. This calculation gives the time difference between when the order was placed and when it was delivered to the customer.

And, to calculate the "diff_estimated_delivery" variable, we need to subtract the "order_delivered_customer_date" from the "order_estimated_delivery_date" for each order. This calculation gives the difference between the estimated delivery date and the actual delivery date for each order.

We will then use JOIN statements to combine the data from the three tables: "orders", "order_items", and "products". It joins these tables on their common columns "order_id" and "product_id" to retrieve data related to each order.

Query Code:

```
SELECT
  o.order_id,
  o.order_purchase_timestamp,
  o.order_delivered_customer_date,
  o.order_estimated_delivery_date,
  o.order_delivered_customer_date - o.order_purchase_timestamp AS time_to_delivery,
  o.order_estimated_delivery_date - o.order_delivered_customer_date AS
diff_estimated_delivery
FROM
   `target_retail_dataset.orders` as o
JOIN
  `target_retail_dataset.order_items` as oi
  ON o.order_id = oi.order_id
JOIN
  `target_retail_dataset.products` as p
  ON oi.product_id = p.product_id;
```



Insights:

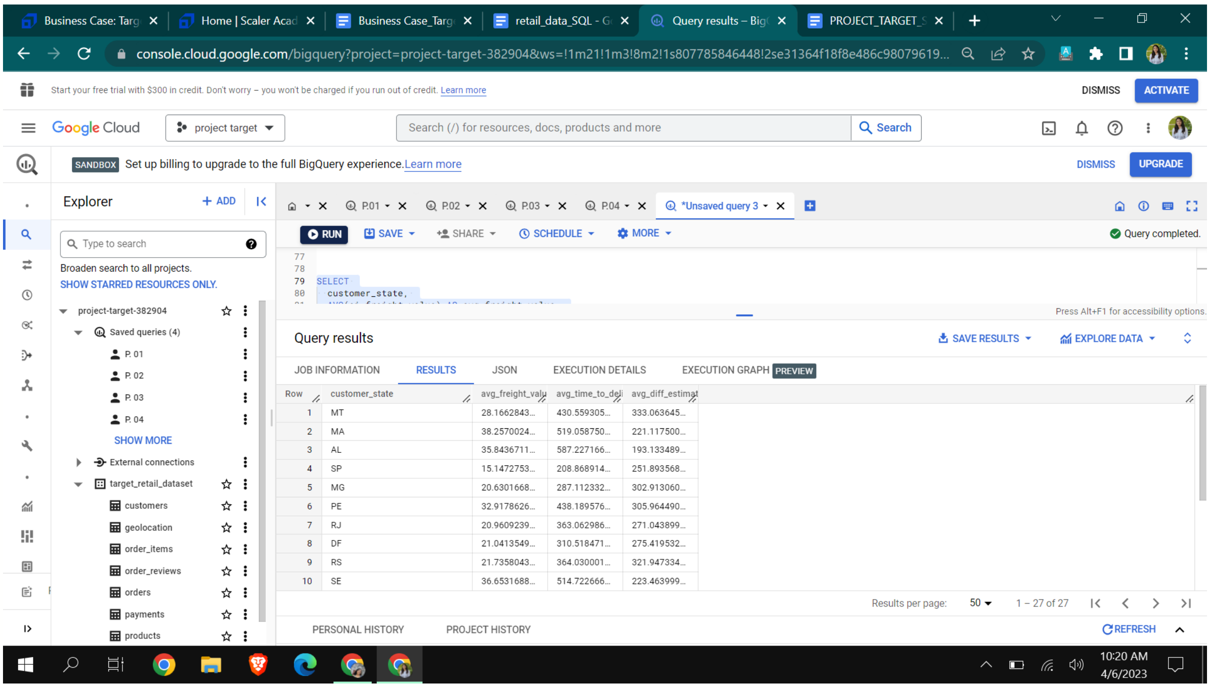
This query provides valuable insights into the delivery performance of a retail company.

Recommendations:
We can evaluate the accuracy of the estimated delivery dates and identify areas where the company needs to improve its forecasting and planning processes.

3. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Methodology:
To solve this, we will use a JOIN operation to combine data from three different tables, namely orders, order_items, and customers. We will then group the data by customer_state, and the average values for freight_value, time_to_delivery, and will calculate the diff_estimated_delivery using the AVG() function. The freight_value represents the shipping cost of the order, the time_to_delivery represents the time taken for the order to be delivered to the customer, and the diff_estimated_delivery represents the difference between the estimated delivery date and the actual delivery date of the order.

Query Code:
```
SELECT
  customer_state,
  AVG(oi.freight_value) AS avg_freight_value,
  AVG(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
HOUR)) AS avg_time_to_delivery,
  AVG(TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
HOUR)) AS avg_diff_estimated_delivery
FROM
  `target_retail_dataset.orders` as o
JOIN
  `target_retail_dataset.order_items` as oi
  ON o.order_id = oi.order_id
JOIN
  `target_retail_dataset.customers` as c
  ON o.customer_id = c.customer_id
GROUP BY
  customer_state;
```



Insights

The query provides valuable insights into the average shipping cost and delivery time for each state. By grouping the data by state, the query allows us to identify which states have the highest and lowest shipping costs and delivery times. This information can be used to optimise logistics and shipping strategies, especially for states with high shipping costs and long delivery times.

Recommendations:

Based on the insights provided by the query, the company can take several actions to optimise its logistics and shipping strategies. Additionally, the company can leverage this information to tailor its marketing and sales efforts to customers in different states, by highlighting faster delivery times or lower shipping costs for specific states.

5.4. Sort the data to get the following:

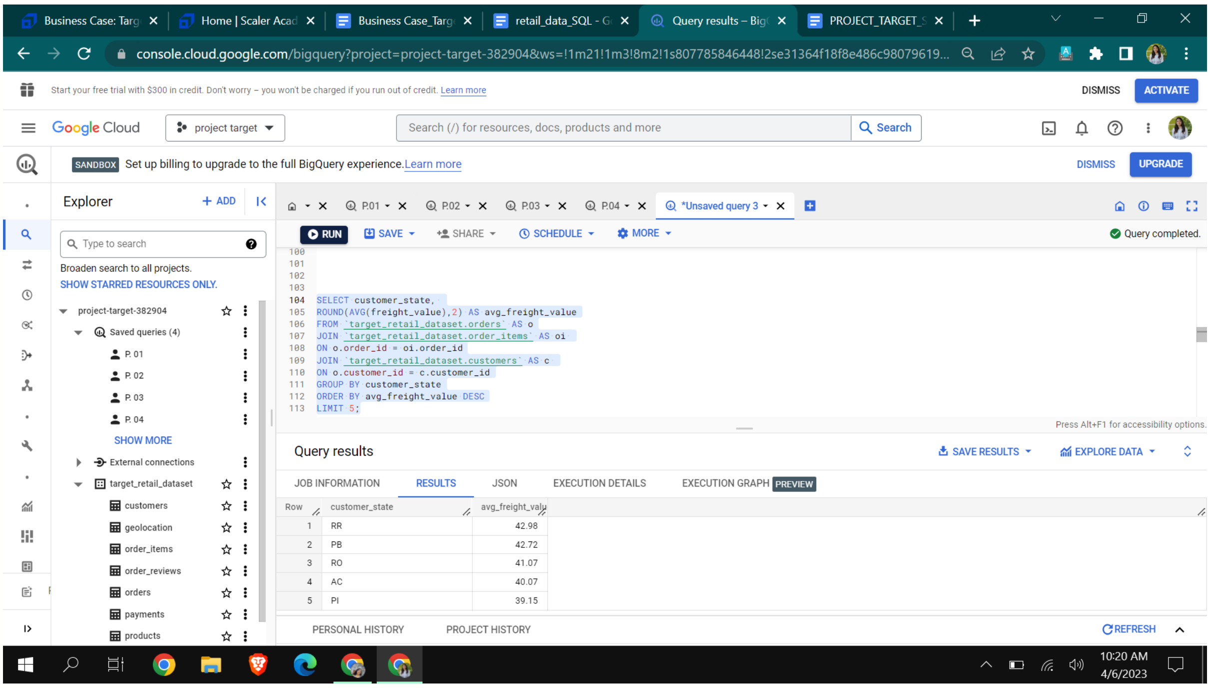
4.1 Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

Methodology:

We will join three tables in the dataset - "orders," "order_items," and "customers" - on their respective primary and foreign keys. And then group the results by customer_state and calculate the average freight_value for each state. Finally, we will sort results in descending order by average freight_value and limited to the top 5 states.

Query Code:

```sql
SELECT customer_state,
ROUND(AVG(freight_value),2) AS avg_freight_value
FROM `target_retail_dataset.orders` AS o
JOIN `target_retail_dataset.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `target_retail_dataset.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY avg_freight_value DESC
LIMIT 5;
```



Insights

Top 5 States are: RR, PB, RO, AC, PI

Recommendations

- For the states with the highest average freight value, the company could explore options like negotiating better shipping rates with logistics providers, optimizing their shipping routes, and encouraging customers to opt for slower delivery options to reduce costs.
- For the states with the lowest average freight value, the company could focus on increasing their sales in these states by running targeted promotions and advertisements to attract more customers.
- The company could also look at the product mix being sold in each state to identify any trends or patterns that could be contributing to higher or

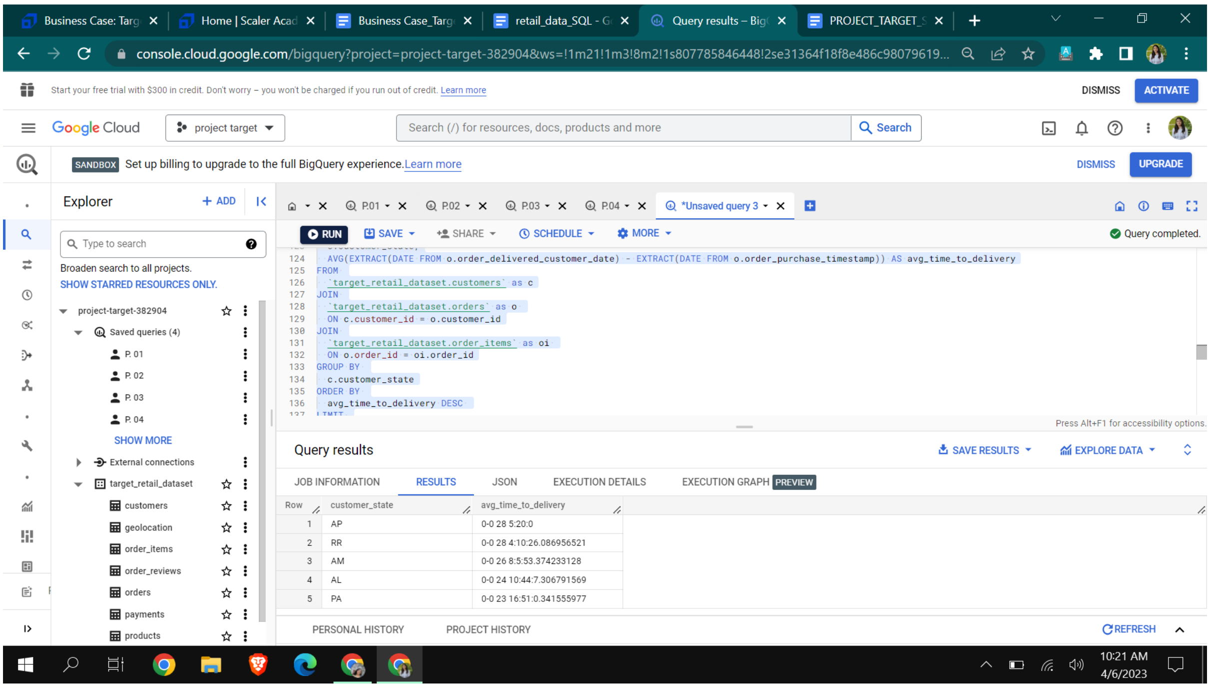
lower shipping costs. This could help them optimize their inventory and shipping operations to reduce costs and improve overall profitability.

5.4. Sort the data to get the following:
Top 5 states with highest/lowest average time to delivery

Methodology:
Retrieving the data from three tables: customers, orders, and order_items, using joins. Then calculating the average time to delivery for each customer state by subtracting the purchase timestamp from the delivered customer date, and then taking the average. Finally, sorting the data in descending order of average time to delivery, and the top 5 states with the highest average time to delivery are returned.

Query Code:
```
SELECT
  c.customer_state,
  AVG(EXTRACT(DATE FROM o.order_delivered_customer_date) - EXTRACT(DATE FROM
o.order_purchase_timestamp)) AS avg_time_to_delivery
FROM
  `target_retail_dataset.customers` as c
JOIN
  `target_retail_dataset.orders` as o
  ON c.customer_id = o.customer_id
JOIN
  `target_retail_dataset.order_items` as oi
  ON o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  avg_time_to_delivery DESC
LIMIT
  5;
```

-- Top 5 states with highest average time to delivery



Insights: Top 5 States: AP, RR, AM, AL, PA

Recommendations:

- Identify the reasons for longer delivery times in the states with the highest average time to delivery. This could involve analyzing the delivery network, identifying bottlenecks, and optimizing delivery routes to reduce transit times.
- Implement strategies to improve delivery times in the states with the highest average time to delivery. This could involve increasing the number of delivery personnel or using technology such as automated delivery vehicles to streamline the process.
- Monitor delivery times across all states and take action to address any issues promptly. This could involve setting targets for delivery times, tracking progress, and taking corrective action if necessary.

--- Top 5 states with lowest average time to delivery

Methodology:

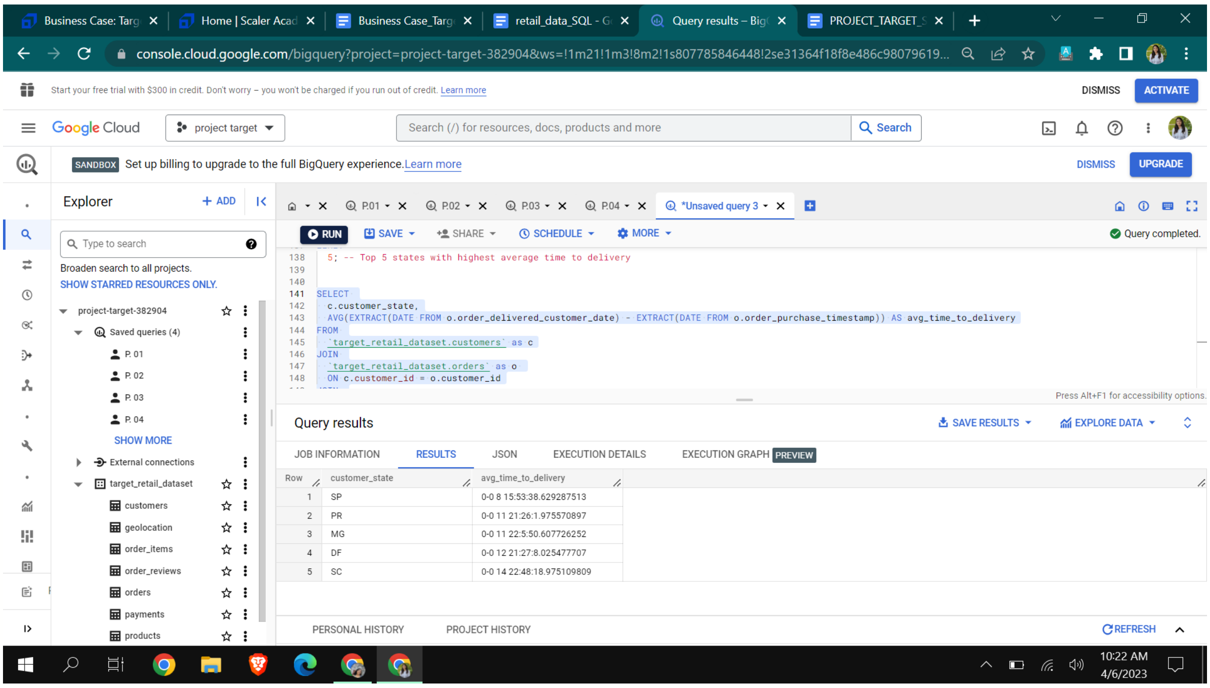
We will join the same tables using the customer_id and order_id columns to establish relationships between them. We will then calculate the average time to delivery for each state by subtracting the purchase timestamp from the delivery timestamp and averaging the results. Finally, we will sort the results in ascending order by the average time to delivery and limited to the top five states.

Query Code:

```
SELECT
  c.customer_state,
```

```
  AVG(EXTRACT(DATE FROM o.order_delivered_customer_date) - EXTRACT(DATE FROM
o.order_purchase_timestamp)) AS avg_time_to_delivery
FROM
  `target_retail_dataset.customers` as c
JOIN
  `target_retail_dataset.orders` as o
  ON c.customer_id = o.customer_id
JOIN
  `target_retail_dataset.order_items` as oi
  ON o.order_id = oi.order_id
GROUP BY
  c.customer_state
ORDER BY
  avg_time_to_delivery ASC
LIMIT
  5;

-- Top 5 states with lowest average time to delivery
```



Insights: Top 5 States: SP, PR, MG, DF, SC

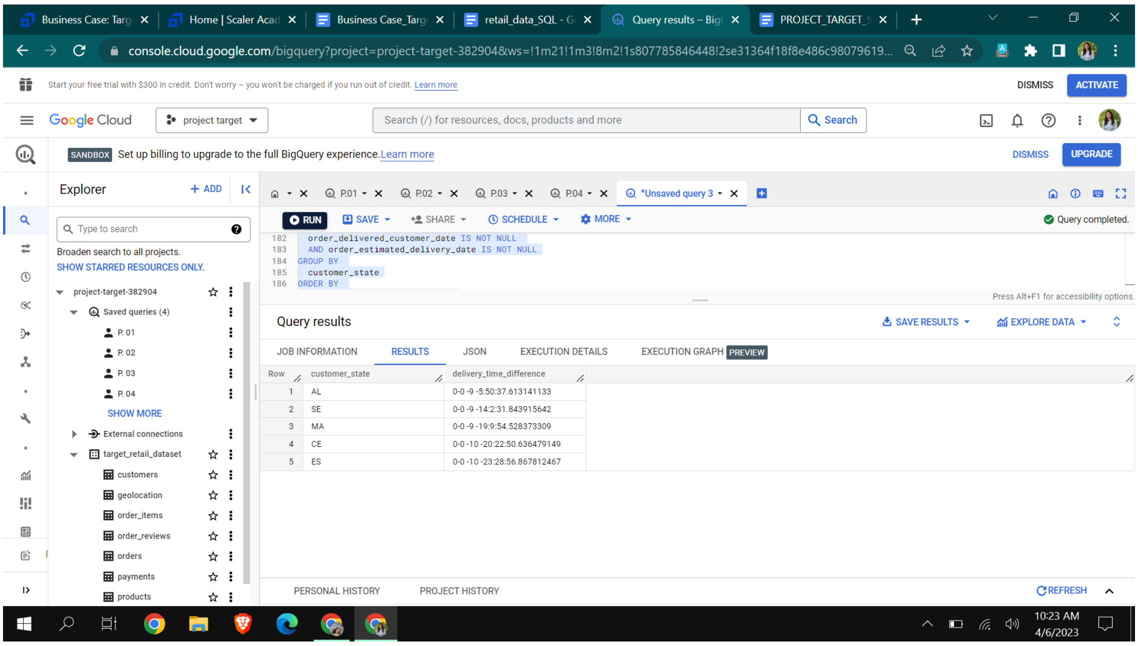
Recommendations: Focus on improving delivery times in the states with higher average delivery times. We can potentially figure out areas for improvement in other regions.

5.4. Sort the data to get the following:

4.3 Top 5 states where delivery is really fast/ not so fast compared to estimated date

Methodology:
We will extract data from the orders, order_items, sellers, customers, and geolocation tables of the target_retail_dataset database. It joins these tables to get the customer state and the difference between the actual delivery date and the estimated delivery date. The query then groups the data by customer state and sorts it by delivery time difference in descending order. Finally, it limits the results to the top 5 states.

Query Code:
```
SELECT
  customer_state,
  AVG(EXTRACT(DATE FROM order_delivered_customer_date) - EXTRACT(DATE FROM
order_estimated_delivery_date)) AS delivery_time_difference
FROM
  `target_retail_dataset.orders` AS o
  JOIN `target_retail_dataset.order_items` AS oi
  ON o.order_id = oi.order_id
  JOIN `target_retail_dataset.sellers` AS s
  ON oi.seller_id = s.seller_id
  JOIN `target_retail_dataset.customers` AS c
  ON o.customer_id = c.customer_id
  JOIN `target_retail_dataset.geolocation` AS g
  ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
WHERE
  order_delivered_customer_date IS NOT NULL
  AND order_estimated_delivery_date IS NOT NULL
GROUP BY
  customer_state
ORDER BY
  delivery_time_difference DESC
LIMIT
  5;
```



Insights: Top 5 States are:AL, SE, MA, CE, ES

Recommendations: We can identify the reasons for the delay and work on addressing them to improve customer satisfaction. On the other hand, the states where delivery is faster than the estimated delivery date can be analysed to identify the reasons for the faster delivery and replicate them in other states.

#6. Payment type analysis:

1. Month over Month count of orders for different payment types

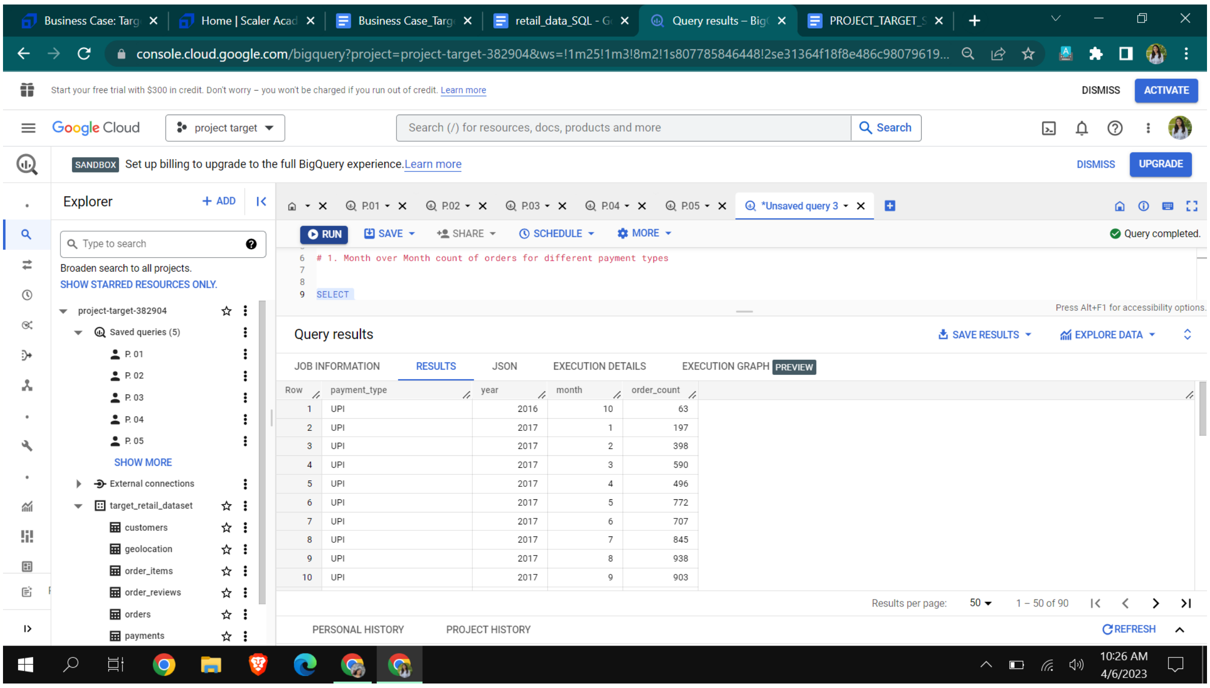
Methodology:

Joining the orders and payments tables on the order ID and grouping the results by payment type, year, and month. Then, ordering the results by payment type, year, and month.

Query Code:

```
SELECT
  payment_type,
  EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
  COUNT(DISTINCT o.order_id) AS order_count
FROM
  `target_retail_dataset.orders` as o
  INNER JOIN `target_retail_dataset.payments` as p
  ON o.order_id = p.order_id
GROUP BY
```

```sql
  payment_type,
  year,
  month
ORDER BY
  payment_type,
  year,
  month;
```



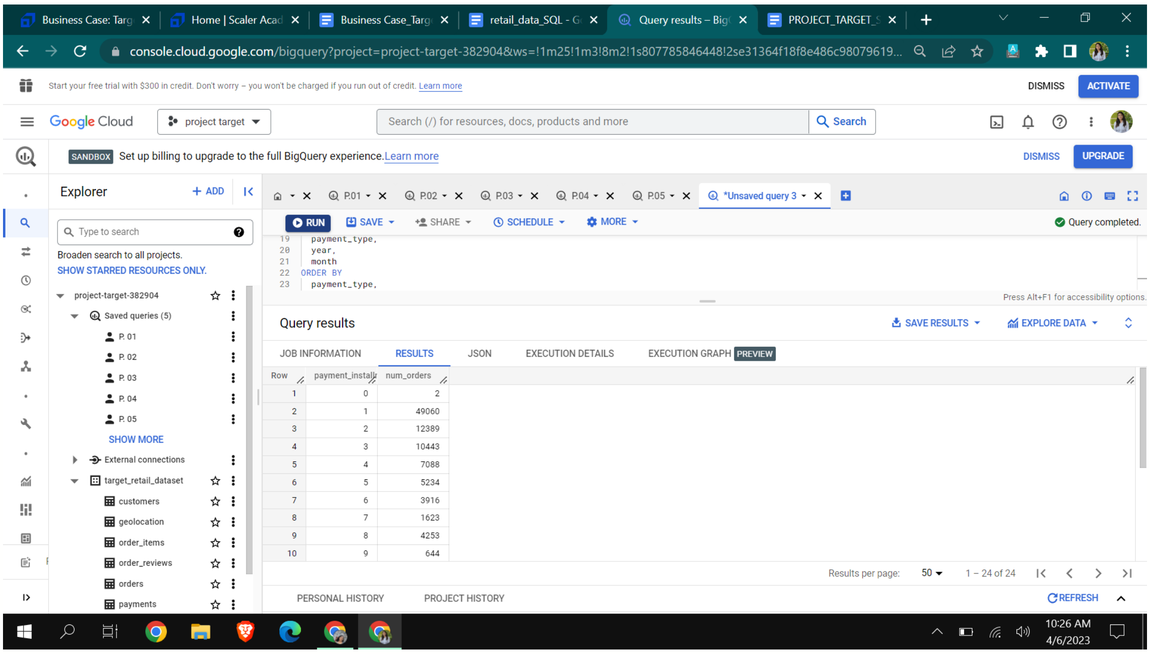
Recommendations: Optimise payment processing and improve the customer experience.

2.Count of orders based on the no. of payment instalments

Methodology: We need to analyse the frequency of orders based on the number of payment instalments. By counting the number of unique order IDs for each distinct value of payment_installments, the query will be able to show how many orders were made using a particular number of payment instalments.

Query Code:

```sql
SELECT payment_installments,
COUNT(DISTINCT order_id) as num_orders
FROM `target_retail_dataset.payments`
GROUP BY payment_installments;
```



Explorer
project-target-382904
Saved queries (5)
P.01
P.02
P.03
P.04
P.05
SHOW MORE
External connections
target_retail_dataset
customers
geolocation
order_items
order_reviews
orders
payments
RUN
SAVE
SHARE
SCHEDULE
MORE
Query completed.
payment_type,
year,
month
ORDER BY
payment_type,
Query results
JOB INFORMATION
RESULTS
JSON
EXECUTION DETAILS
EXECUTION GRAPH
PREVIEW
Row
payment_install
num_orders
1 0 2
2 1 49060
3 2 12389
4 3 10443
5 4 7088
6 5 5234
7 6 3916
8 7 1623
9 8 4253
10 9 644
Results per page: 50
1 – 24 of 24
PERSONAL HISTORY
PROJECT HISTORY
REFRESH
10:26 AM
4/6/2023