Business Case: Target SQL
PAGE DISTIBUTION

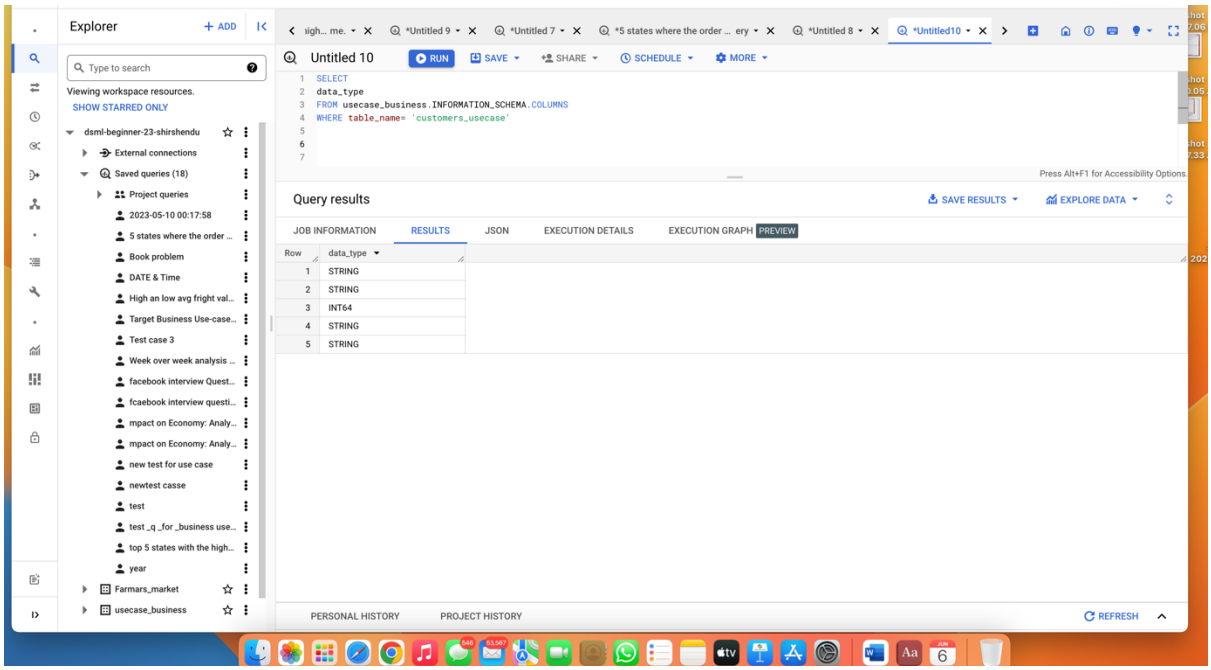
1) Solutions and solution approach
2) Actionable Insights & Recommendations

1.1: Data type of all columns in the "customers" table.

Code :

```
SELECT
data_type
FROM usecase_business.INFORMATION_SCHEMA.COLUMNS
WHERE table_name= 'customers_usecase'
```

Screenshot :



Explanation: From this INFORMATION_SCHEMA.COLUMNS query we can identify the data type of customer table for all the columns.

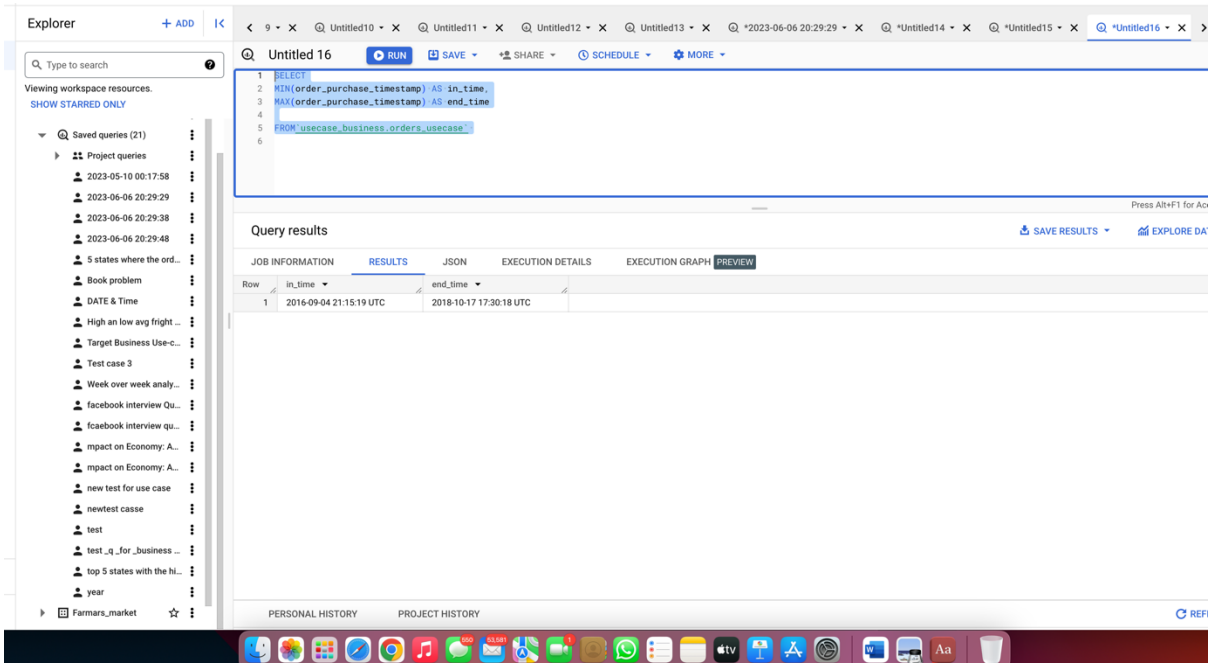
Insights: datatype of each column in customer table .can help us to proceed further dig down the analysis

1.2: Get the time range between which the orders were placed.

Answer-
Code - USING BIG – Query

```
SELECT
MIN(order_purchase_timestamp) AS in_time,
MAX(order_purchase_timestamp) AS end_time

FROM`usecase_business.orders_usecase`
```

Screenshot:



Explanation: We are just using MIN and MAX function to retrieve the time range between which the orders were placed

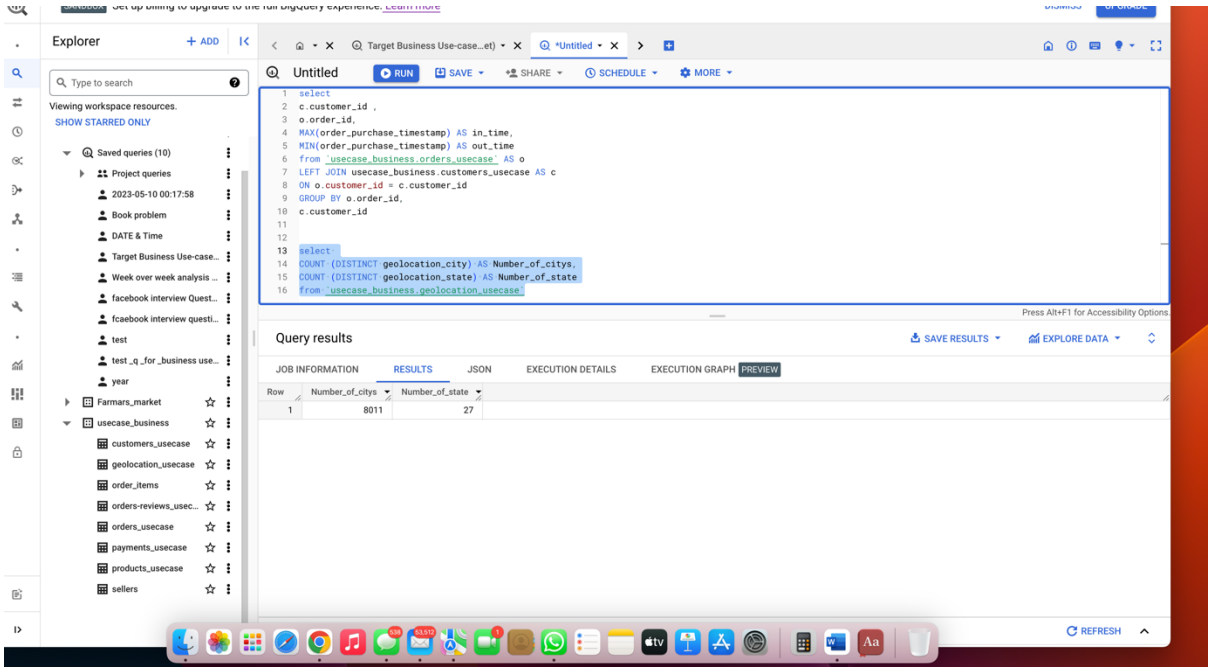
Insight - Output is providing us an insight about period at which orders were placed by users .

1.3 : Count the number of Cities and States in our dataset

Code - USING BIG – Query

```
select
COUNT (DISTINCT geolocation_city) AS Number_of_citys,
COUNT (DISTINCT geolocation_state) AS Number_of_state
from `usecase_business.geolocation_usecase`
```

Screenshot:



Explanation : Using count function from the geolocation table we can fetch out the details of number of Cities and States in our dataset

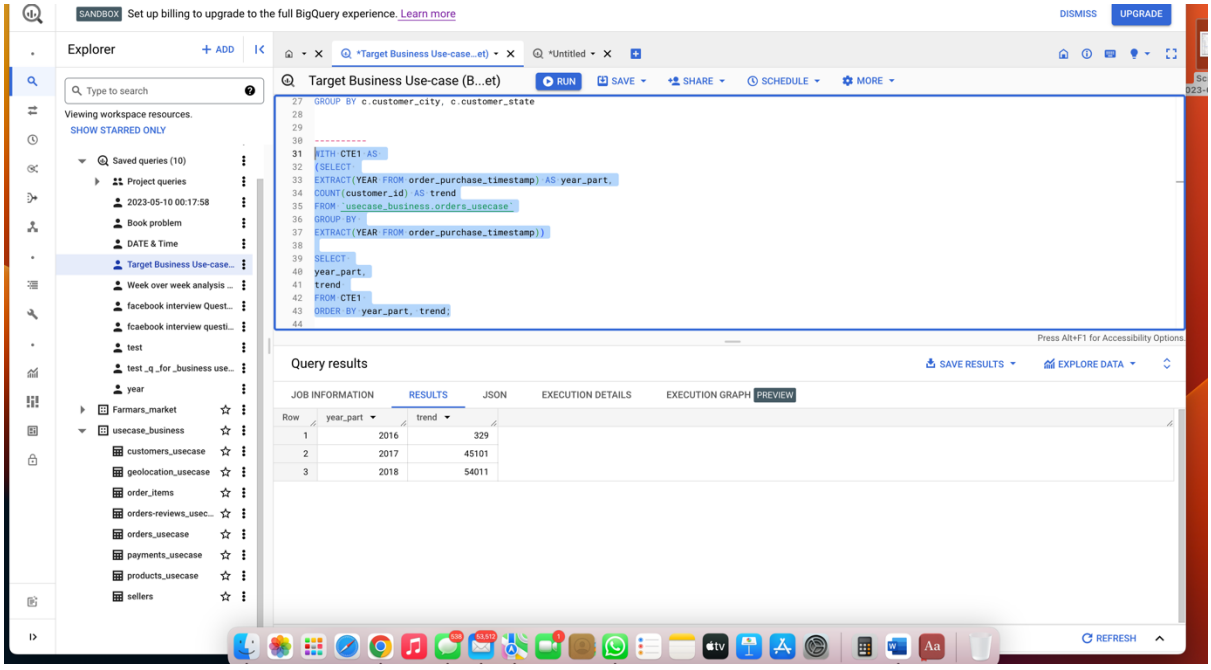
Insights : Pulling out a total number would be great for in depth exploration . if we Look at the total number of state and city that would be great to go further exploration and it will be easy to analyse data further

2.1 Is there a growing trend in the no. of orders placed over the past years?

Code:

```
WITH CTE1 AS
(SELECT
EXTRACT(YEAR FROM order_purchase_timestamp) AS year_part,
COUNT(order_id) AS trend
FROM `usecase_business.orders_usecase`
GROUP BY
EXTRACT(YEAR FROM order_purchase_timestamp))

SELECT
year_part,
trend
FROM CTE1
ORDER BY year_part, trend;
```

Screenshot:



Explanation : Using Extract year function we are pulling out the years and then grouping them to get individual count of customer As trend who were in orders table .

If the count of customer As trend is greater than the previous year data we can clearly identify the trend
Using Common table expression CTE to get a better visibility on the code

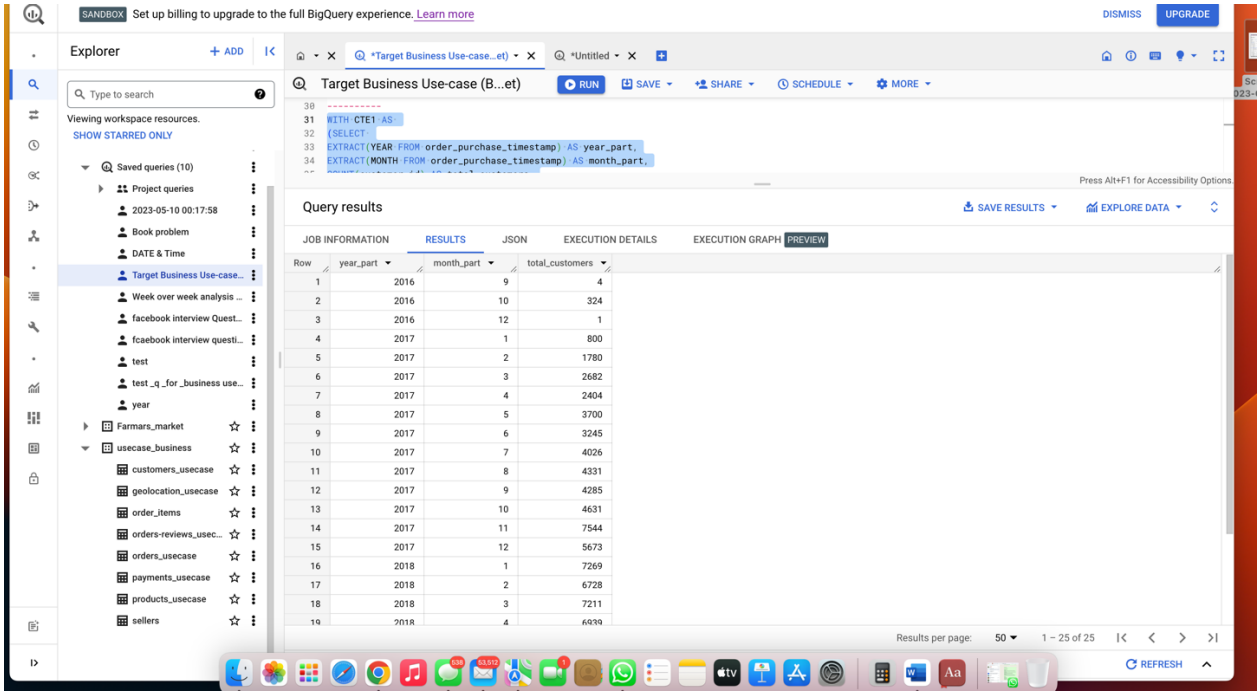
Insights: Analysing the trend in the number of orders placed over the years. Look for consistent growth or decline in order volume. Identify any significant spikes or drops in order count As we can see that there is a clear trend of increasing order every year

2.2 Can we see some kind of monthly seasonality in terms of the no. of orders being placed?

Code :

```
WITH CTE1 AS
(SELECT
EXTRACT(YEAR FROM order_purchase_timestamp) AS year_part,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month_part,
COUNT(customer_id) AS total_customers
FROM `usecase_business.orders_usecase`
GROUP BY
EXTRACT(YEAR FROM order_purchase_timestamp),
EXTRACT(MONTH FROM order_purchase_timestamp))

SELECT
year_part,
month_part,
total_customers
FROM CTE1
ORDER BY year_part, month_part, total_customers
```

Screenshot :
Next page



Explanation : Using Extract year and month function we are pulling out the years and month then grouping them to get individual count of customer As total customer per month data who were in orders table .

If the count of customer As total customers is greater than the previous month data we can clearly identify the z seasonality in terms of the no. of orders being placed.
Using Common table expression CTE to get a better visibility on the code

Insights & recommendation : Seasonal customer trends is basically Analyse the total number of customers for each year and month to identify seasonal patterns. Look for months or periods with higher customer counts and those with lower counts. This information can help you understand customer behaviour and plan marketing campaigns or promotions accordingly.

Q2.3 - During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- **0-6 hrs : Dawn**
- **7-12 hrs : Mornings**
- **13-18 hrs : Afternoon**

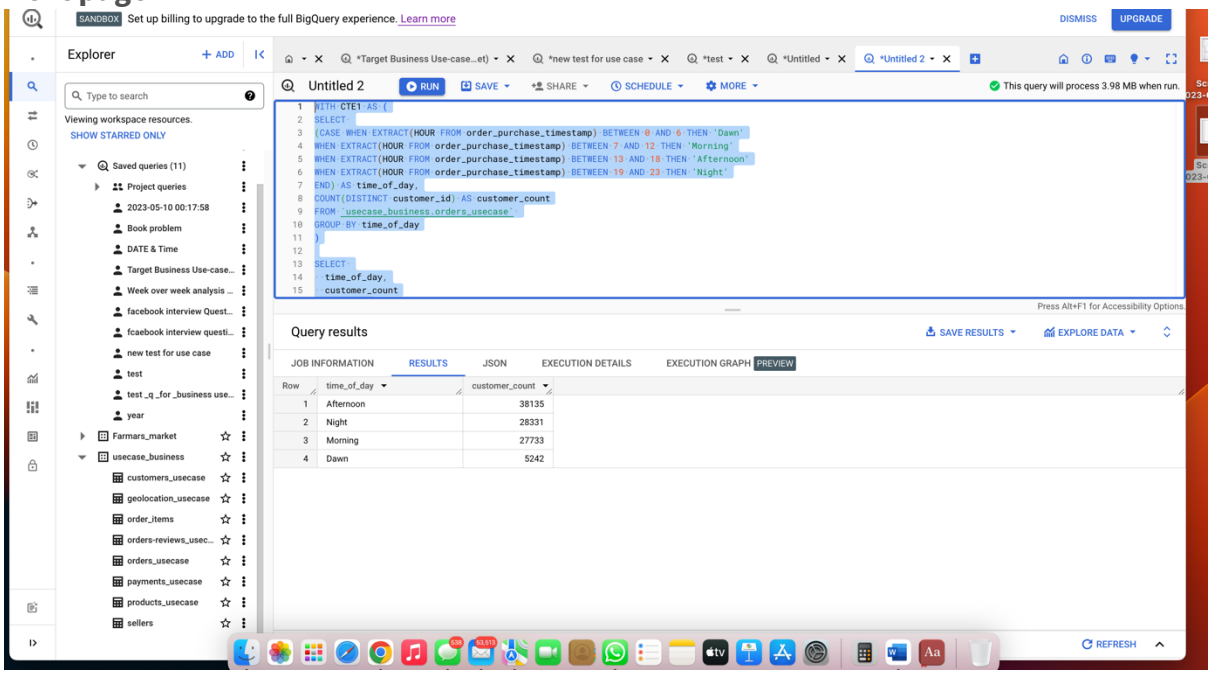
- **19-23 hrs : Night**

Code :

```
WITH CTE1 AS (
SELECT
(CASE WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
END) AS time_of_day,
COUNT(DISTINCT customer_id) AS customer_count
FROM `usecase_business.orders_usecase`
GROUP BY time_of_day
)

SELECT
  time_of_day,
  customer_count
```

Screenshot :

Next page



Explanation : Using Extract function we are pulling out the Hour details to get extract time if it is morning afternoon or night etc .

Then putting this on case statement and ending the case AS time _of the day

Now we are we are counting distinct customer ID from the orders table who placed the order

Then grouping the time of the day part them to get individual data where customers placed order
And finally using Order by in Dscending order to get during what time of the day, do the Brazilian customers mostly place their orders?

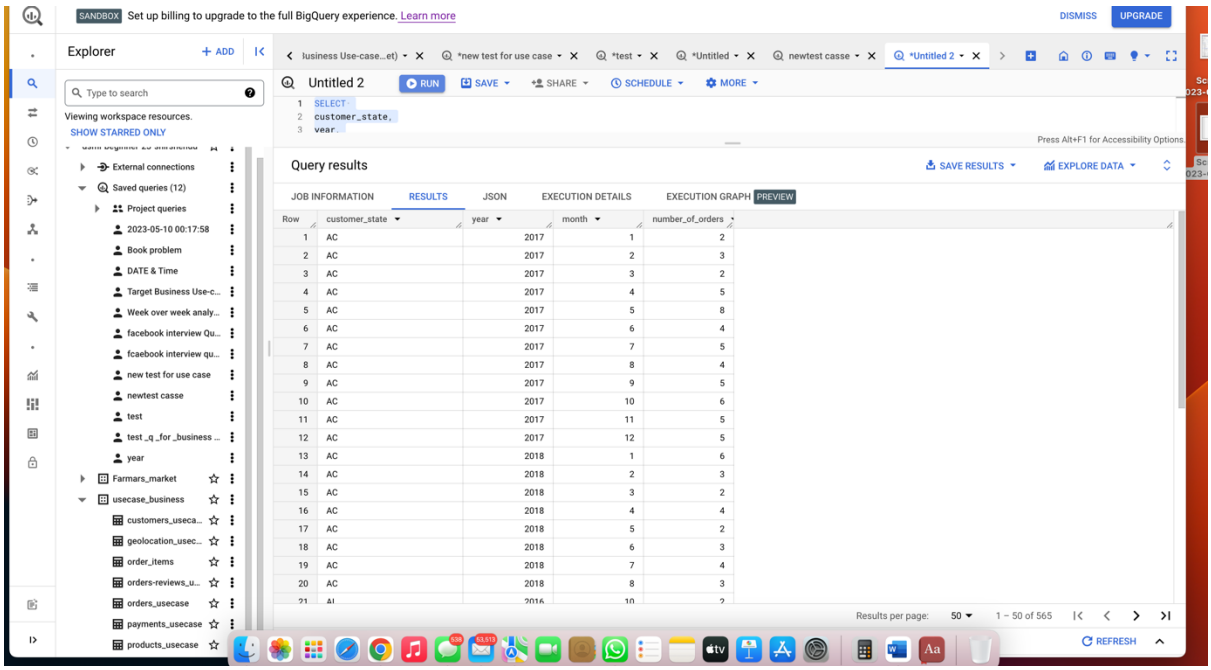
Insights & recommendation : analysing the customer count based on the time of day can reveal patterns in order placement behaviour. By categorizing orders into different time slots (Dawn, Morning, Afternoon, Night), you can identify when customers are most active in making purchases. For example in our case the customers mostly active at the Afternoon time and less active would be Dawn time of the day

Q-3- 1 Get the month on month no. of orders placed in each state.

Code :

```
SELECT
customer_state,
year,
month,
number_of_orders
FROM (SELECT
c.customer_state,
EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
COUNT(o.customer_id) AS number_of_orders
FROM `usecase_business.customers_usecase` AS c
LEFT JOIN`usecase_business.orders_usecase` AS o
ON c.customer_id = o.customer_id
GROUP BY
c.customer_state,
EXTRACT(YEAR FROM order_purchase_timestamp),
EXTRACT(MONTH FROM order_purchase_timestamp))
ORDER BY
customer_state,
year,
month,
number_of_orders
```

Screenshot :



Explanation :

We are extracting month on subsequently year wise data for each customer state

In order to do that first we are joining 2 tables :

1) **Customers and Orders**
2) **From customer table we can get the state**
3) **From orders table we can get the customers who ordered**
4) **Now counting the customers who was there in the order table**
5) **GROUPING the data for each state and month**
6) **Using subquery only pulling out the relevant details**
7) **We are getting each year each state individual month by month orders data**

Insights & recommendation : By analysing the number of orders placed month-on-month for each state, you can identify the order trends and patterns specific to each region. This information can help you understand the demand fluctuations in different states and make data-driven decisions accordingly.
Example : if we look at this data clearly we can see how each year corresponded to month by month data is signifying if there is any growth for a particular state or not

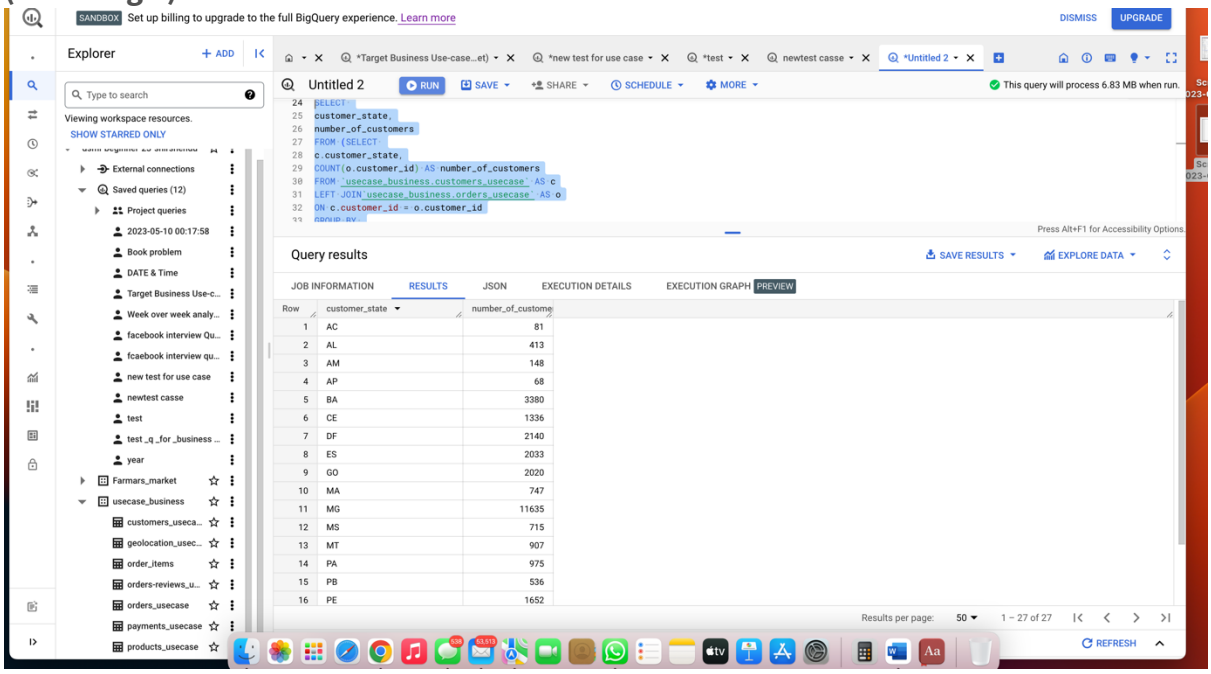
Q3-2 How are the customers distributed across all the states?

Code :

```sql
SELECT
customer_state,
number_of_customers
FROM (SELECT
c.customer_state,
COUNT(o.customer_id) AS number_of_customers
FROM `usecase_business.customers_usecase` AS c
LEFT JOIN`usecase_business.orders_usecase` AS o
ON c.customer_id = o.customer_id
GROUP BY
c.customer_state )
ORDER BY
customer_state,
number_of_customers
```

Screenshot :

(Next Page)



Explanation. :

1) **JOINING Customers and Orders using LEFT join**
2) **From customer table we can get the state**
3) **From orders table we can get the customers who ordered**
4) **Now counting the customers who was there in the order table**
5) **GROUPING the data for each state**
6) **Using subquery only pulling out the relevant details**
7) **We are getting each state customer count who has placed order**

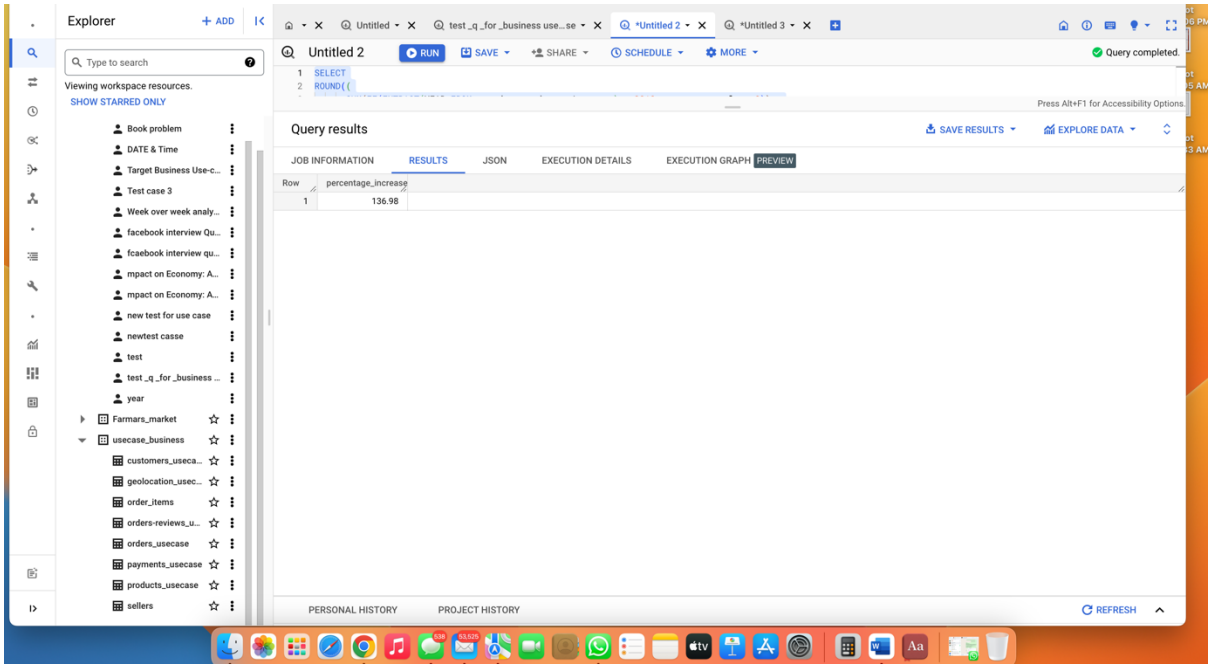
Insights : The number of customers in each state, you can identify states with higher customer concentrations. This information can help you allocate marketing resources and prioritize customer engagement activities in states where you have a larger customer base, and also it will help you grow your business

4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
You can use the "payment value" column in the payments table to get the cost of orders.

Code :

```
SELECT
ROUND((
      SUM(IF(EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018, p.payment_value,
0)) -
      SUM(IF(EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017, p.payment_value,
0))
    ) / SUM(IF(EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017,
p.payment_value, 0)) * 100, 2) AS percentage_increase
  FROM
    `usecase_business.orders_usecase` AS o
  LEFT JOIN `usecase_business.payments_usecase` AS p
    ON o.order_id = p.order_id
  WHERE
    EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018)
    AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
```

Screenshot :



Explanation :
Cost of orders: The query utilizes the "payment value" column in the payments table to calculate the cost of orders.
Time range: The calculation is specific to the years 2017 and 2018, considering only the months between January and August. This allows for a focused analysis within a specific period.
Percentage increase: The query calculates the percentage increase in the cost of orders from 2017 to 2018. It compares the sum of payment values for 2018 orders with the sum of payment values for 2017 orders and calculates the percentage change.
Rounded result: The final percentage increase value is rounded to two decimal places for a more readable and concise presentation.

Insights : analyse the corresponding impact on profitability due to the increase in order costs. Evaluate whether the revenue generated from the higher payment values justifies the associated expenses As we can clearly see there is a percentage increase from our data

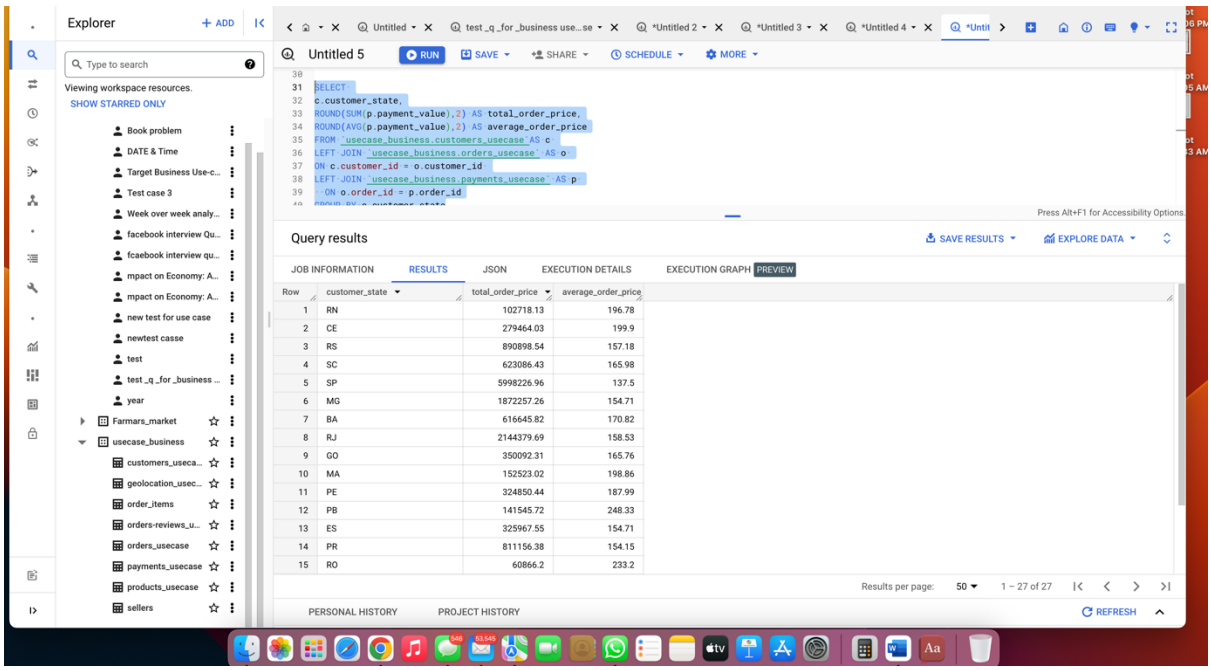
Q4.2- Calculate the Total & Average value of order price for each state.

Code :

```
SELECT
c.customer_state,
ROUND(SUM(p.payment_value),2) AS total_order_price,
ROUND(AVG(p.payment_value),2) AS average_order_price
FROM `usecase_business.customers_usecase`AS c
LEFT JOIN `usecase_business.orders_usecase` AS o
ON c.customer_id = o.customer_id
```

```
LEFT JOIN `usecase_business.payments_usecase` AS p
  ON o.order_id = p.order_id
GROUP BY c.customer_state
```

Screenshot :



Explanation. :

1) **customers_usecase, orders_usecase, and payments_usecase tables to retrieve the necessary information. It calculates the total order price by summing up the payment values**
2) **rounds the result to two decimal places using the ROUND function for our better visibility**
3) **In the same way it is calculates the average order price by averaging the payment values and also rounds the result to two decimal places.**
4) **The GROUP BY clause groups the results by the customer state**

Insights :

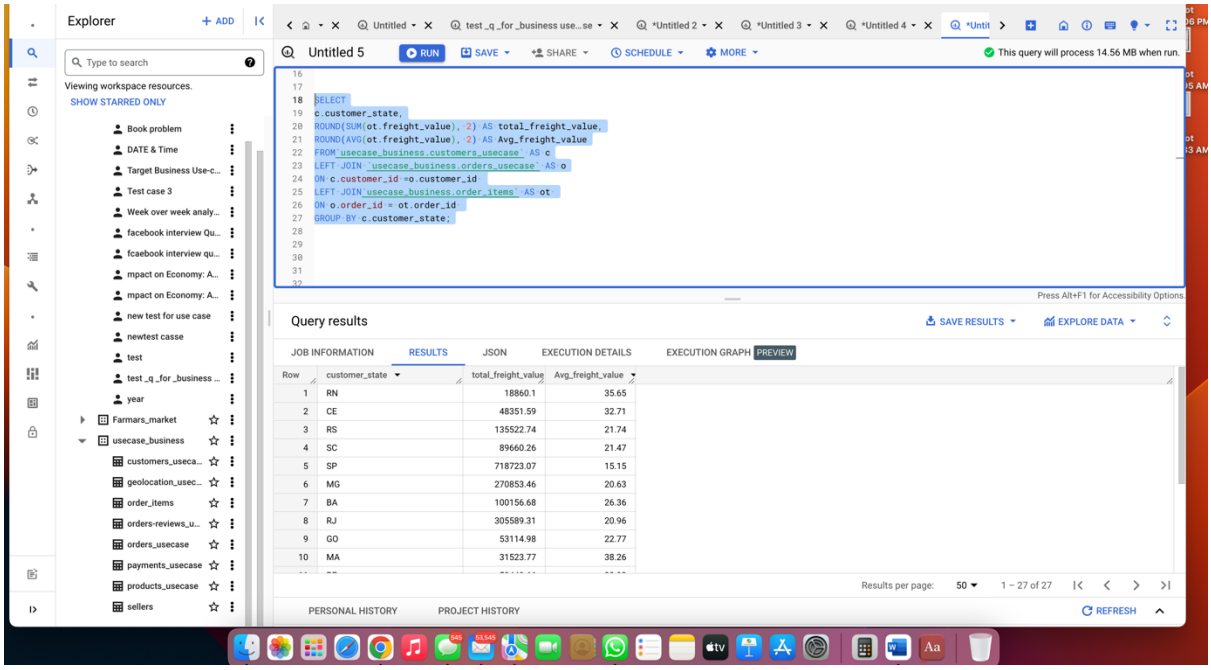
This query provides insights into the total and average order prices for each state. Analyse states with low total order prices we can Identify states where the total order price is relatively lower than others. Investigate the reasons behind this lower average and then accordingly put some marketing campaign to increase the sal

Q4.3 - Calculate the Total & Average value of order freight for each state.

Code

```sql
SELECT
c.customer_state,
ROUND(SUM(ot.freight_value), 2) AS total_freight_value,
ROUND(AVG(ot.freight_value), 2) AS Avg_freight_value
FROM`usecase_business.customers_usecase` AS c
LEFT JOIN `usecase_business.orders_usecase` AS o
ON c.customer_id =o.customer_id
LEFT JOIN`usecase_business.order_items` AS ot
ON o.order_id = ot.order_id
GROUP BY c.customer_state;
```

Screenshot :



Explanation :

1) **JOINING customers_usecase table with the orders_usecase table on the customer_id column and then joins the resulting table with the order_items**
2) **The SUM function is used to calculate the total freight value, and the AVG function is used to calculate the average freight value.**
3) **Then results are rounded to two decimal places**
4) **The GROUP BY clause groups the results by customer_state.**

Insights : Identify states where the total freight value is relatively lower than others accordingly we can start identifying what needs to be our next step to increase that amount

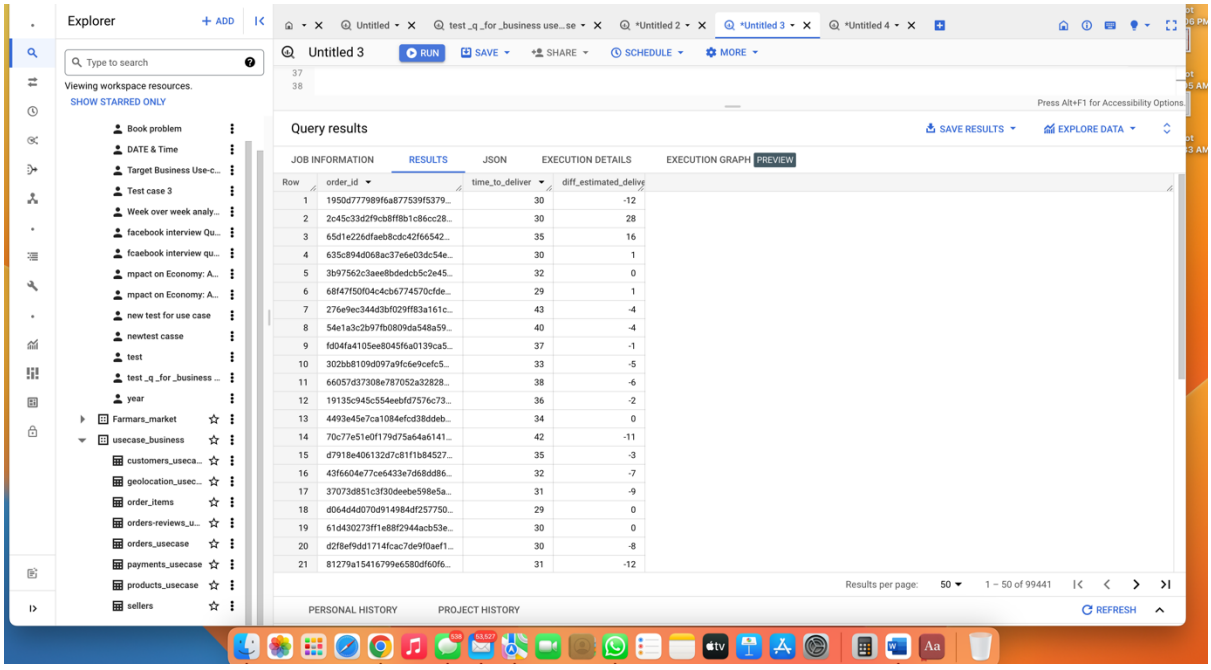
Q5.1 –

Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
Do this in a single query.

Code :

```
select
order_id,
timestamp_diff(order_delivered_customer_date , order_purchase_timestamp, DAY) AS
time_to_deliver,
timestamp_diff(order_estimated_delivery_date ,order_delivered_customer_date,DAY) AS
diff_estimated_delivery
from `usecase_business.orders_usecase`
```

Screenshot :



Explanation :

1) **The query retrieves the order_id and calculates the time_to_deliver by subtracting the order_purchase_timestamp from the order_delivered_customer_date and specifying the unit as DAY.**
2) **In the same way it calculates the diff_estimated_delivery by subtracting the order_delivered_customer_date from the order_estimated_delivery_date and specifying the unit as DAY**
3) **And we are getting our result**

Insights : By analysing the delivery time, businesses can identify any delays in the delivery process. This can help optimize logistics and improve overall efficiency of the customer service process

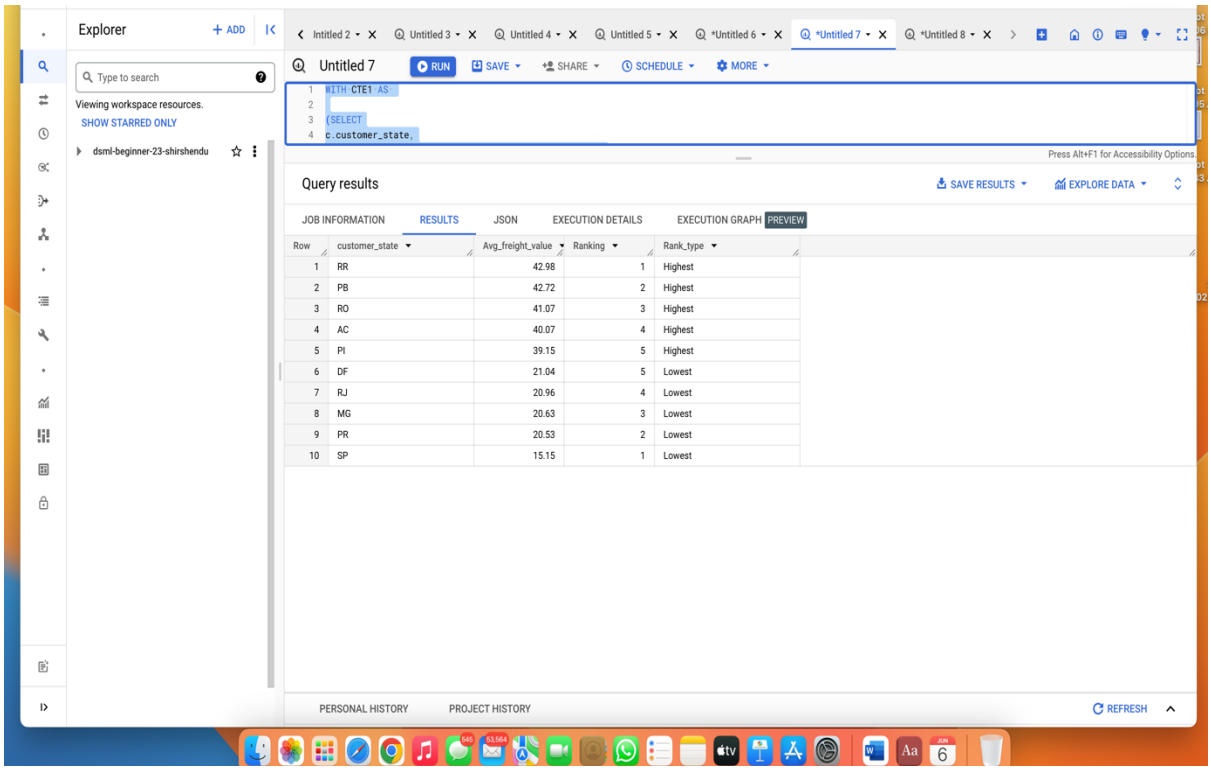
5.2 Find out the top 5 states with the highest & lowest average freight value.

Code :

```
WITH CTE1 AS

(SELECT
c.customer_state,
ROUND(AVG(ot.freight_value), 2) AS Avg_freight_value,
FROM`usecase_business.customers_usecase` AS c
LEFT JOIN `usecase_business.orders_usecase` AS o
ON c.customer_id =o.customer_id
LEFT JOIN`usecase_business.order_items` AS ot
ON o.order_id = ot.order_id
GROUP BY c.customer_state),CTE2 AS
(
SELECT
*,
 DENSE_RANK() OVER (ORDER BY Avg_freight_value DESC) AS rank_highest,
 DENSE_RANK() OVER (ORDER BY Avg_freight_value ASC) AS rank_lowest
FROM
CTE1)

SELECT
  customer_state,
  Avg_freight_value,
  rank_highest AS Ranking,
  'Highest' AS Rank_type
FROM
  CTE2
WHERE
  rank_highest <= 5

UNION ALL

SELECT
  customer_state,
  Avg_freight_value,
  rank_lowest AS Ranking,
    'Lowest' AS Rank_type
FROM
  CTE2
```

```
WHERE
  rank_lowest <= 5

ORDER BY Avg_freight_value DESC, ranking;
```

Screenshot :



Explanation :

1) **(CTE1 and CTE2) using the WITH clause. CTE1 calculates the average freight value for each customer state by joining the customers_usecase, orders_usecase, and order_items tables. The result is grouped by the customer state, and the average freight value is rounded to 2 decimal places.**

2) **CTE2 builds on CTE1 and adds two additional columns rank_highest and rank_lowest. These columns are calculated using the RANK() function and assign a ranking to each state based on the average freight value. The highest average freight values receive lower ranks, while the lowest average freight values receive higher ranks.**

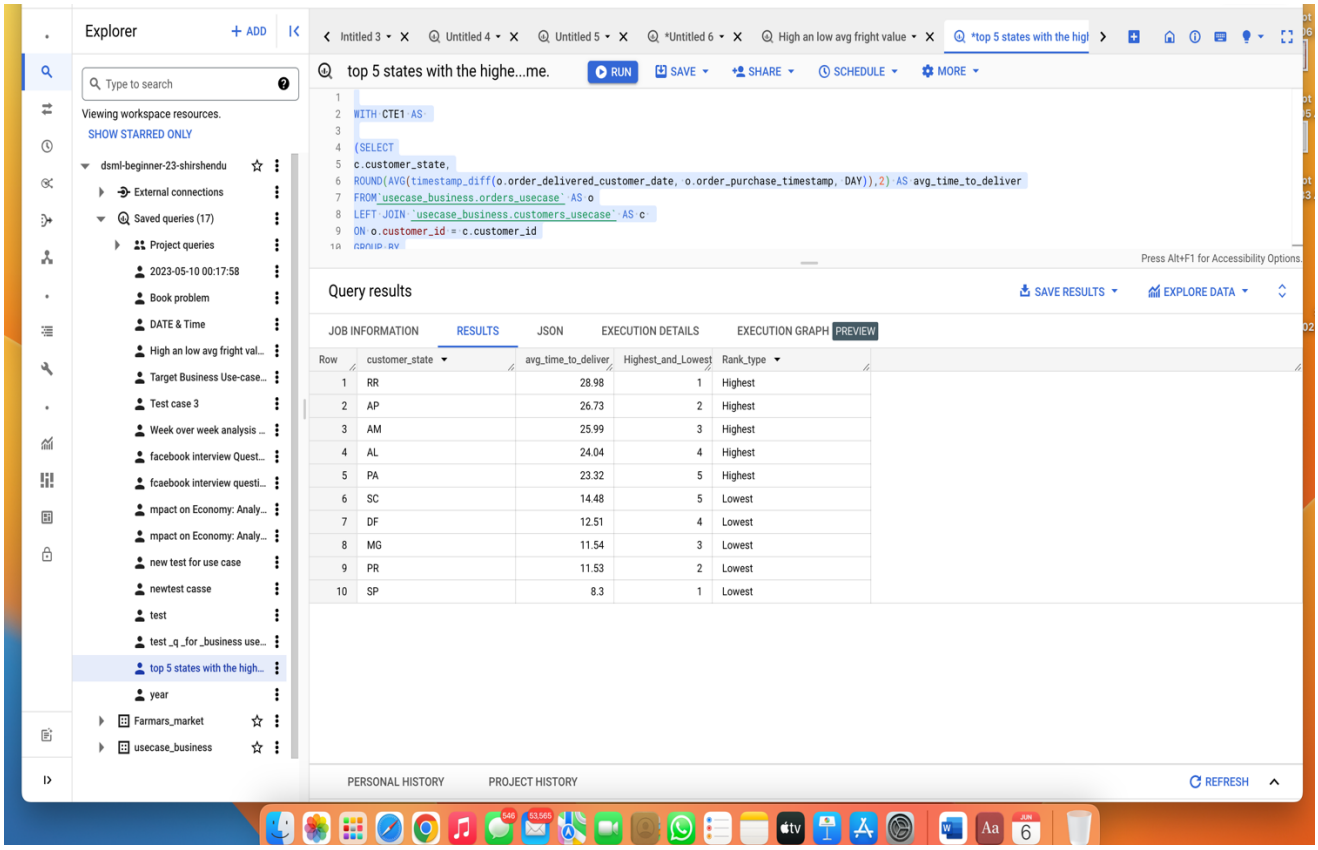
Insights : Highest Average Freight Value: The top 5 states with the highest average freight Lowest Average Freight Value: The top 5 states with the lowest average freight value are also presented,
where customers generally have lower shipping costs. These states may benefit .
Through this information we can improve the channelisation of the transport and make better customer experience next time

5.3– Find out the top 5 states with the highest & lowest average delivery time.

CODE:

```
WITH CTE1 AS

(SELECT
c.customer_state,
ROUND(AVG(timestamp_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) AS avg_time_to_deliver
FROM`usecase_business.orders_usecase` AS o
LEFT JOIN `usecase_business.customers_usecase` AS c
ON o.customer_id = c.customer_id
GROUP BY
c.customer_state), CTE2 AS

(
SELECT
*,
DENSE_RANK() OVER ( ORDER BY avg_time_to_deliver DESC ) AS  Highest_avarage,
DENSE_RANK() OVER ( ORDER BY avg_time_to_deliver ASC ) AS  Lowest_avarage
FROM CTE1
)

SELECT
customer_state,
CTE2.avg_time_to_deliver,
Highest_avarage AS Highest_and_Lowest_avg,
'Highest' AS Rank_type
FROM CTE2
WHERE
Highest_avarage<= 5

UNION ALL

SELECT
customer_state,
CTE2.avg_time_to_deliver,
Lowest_avarage AS Highest_and_Lowest_avg,
'Lowest' AS Rank_type
FROM CTE2
WHERE
Lowest_avarage<= 5
ORDER BY avg_time_to_deliver DESC ,
Highest_and_Lowest_avg,
Rank_type
```

Screenshot:



Explanation :

1) **CTE1: This CTE calculates the average delivery time in days for each state. It uses the orders_usecase table to calculate the time difference between the order purchase date and the delivery date. The average delivery time is rounded to two decimal places.**
2) **CTE2: This CTE assigns rankings to the average delivery time values. It uses the DENSE_RANK() function to assign ranks based on the average delivery time, both in ascending and descending order.**

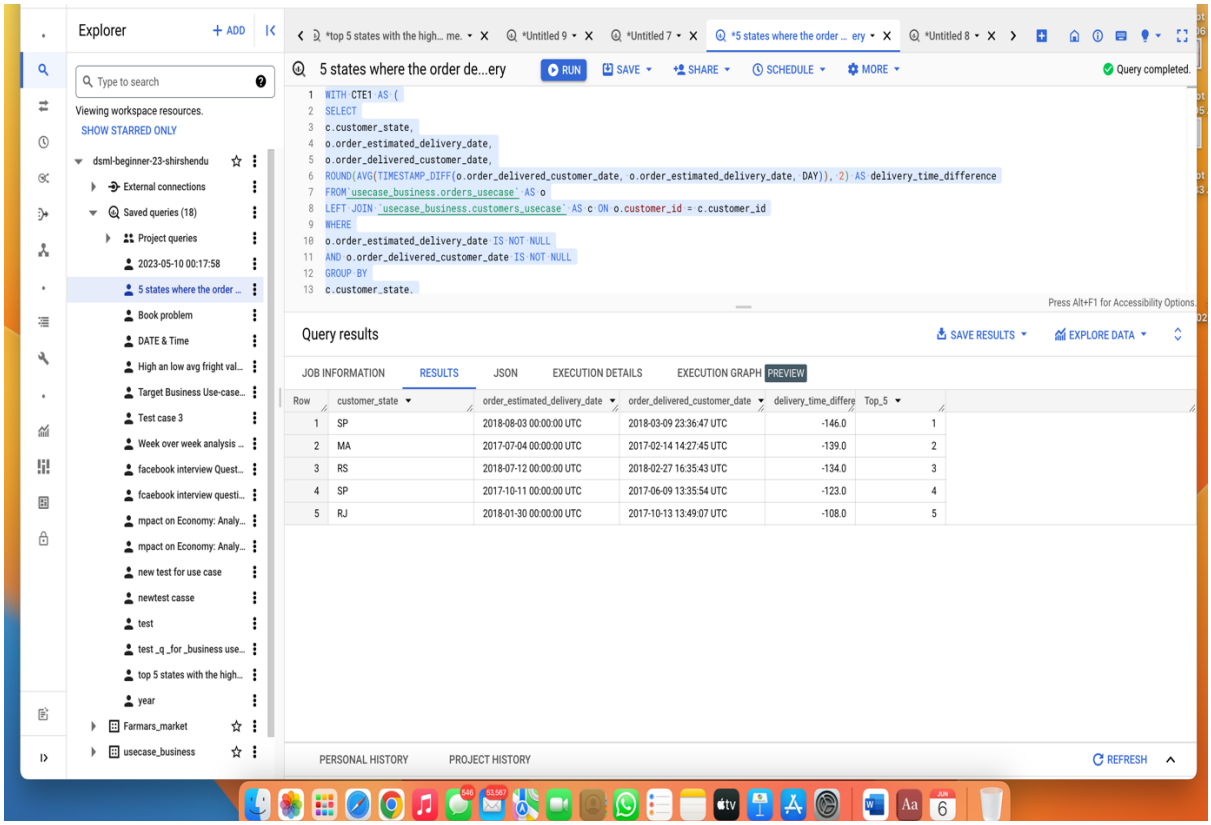
Insights : insights into the top 5 states with the highest and lowest average delivery times. Businesses can use this information to identify regions where delivery times are longer or shorter on average. This can help in optimizing logistics operation

5.4-Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

CODE :

```
WITH CTE1 AS (
SELECT
c.customer_state,
o.order_estimated_delivery_date,
o.order_delivered_customer_date,
ROUND(AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)), 2) AS delivery_time_difference
FROM`usecase_business.orders_usecase` AS o
LEFT JOIN `usecase_business.customers_usecase` AS c ON o.customer_id =
c.customer_id
WHERE
o.order_estimated_delivery_date IS NOT NULL
AND o.order_delivered_customer_date IS NOT NULL
GROUP BY
c.customer_state,
o.order_estimated_delivery_date,
o.order_delivered_customer_date
),
CTE2 AS (
SELECT
 *,
DENSE_RANK() OVER (ORDER BY delivery_time_difference ASC) AS raank
FROM
CTE1
)
SELECT
customer_state,
order_estimated_delivery_date,
order_delivered_customer_date,
delivery_time_difference,
raank AS Top_5
FROM
CTE2
WHERE
raank <= 5
ORDER BY
delivery_time_difference;
```

Screenshot :



Explanation :

Common Table Expressions (CTEs):

1) **CTE1: This CTE calculates the delivery time difference in days between the estimated delivery date and the actual delivered date for each state. It uses the orders_usecase table to calculate the time difference and rounds it to two decimal places.**
2) **The CTE filters out records where either the estimated delivery date or the actual delivered date is null.**
3) **CTE2: This CTE assigns rankings based on the delivery time difference values. It uses the DENSE_RANK() function to assign ranks in ascending order.**
4) **Note : Formula we have used :**

   difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

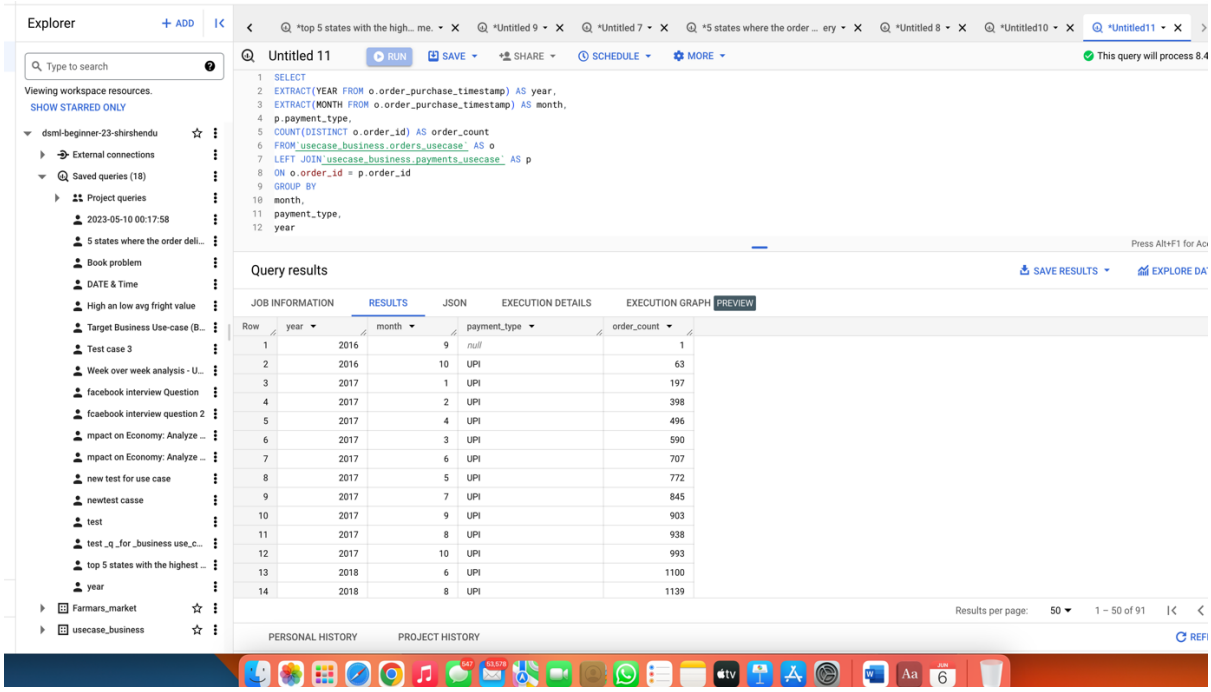
Insight : From this insights into the top 5 states where the order delivery is exceptionally fast compared to the estimated date of delivery. This information can help businesses identify regions where their logistics operations are performing efficiently and delivering orders ahead of schedu

6.1 Find the month on month no. of orders placed using different payment types.

CODE:

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
p.payment_type,
COUNT(DISTINCT o.order_id) AS order_count
FROM`usecase_business.orders_usecase` AS o
LEFT JOIN`usecase_business.payments_usecase` AS p
ON o.order_id = p.order_id
GROUP BY
month,
payment_type,
year
ORDER BY
payment_type,
order_count,
year
```

Screenshot :



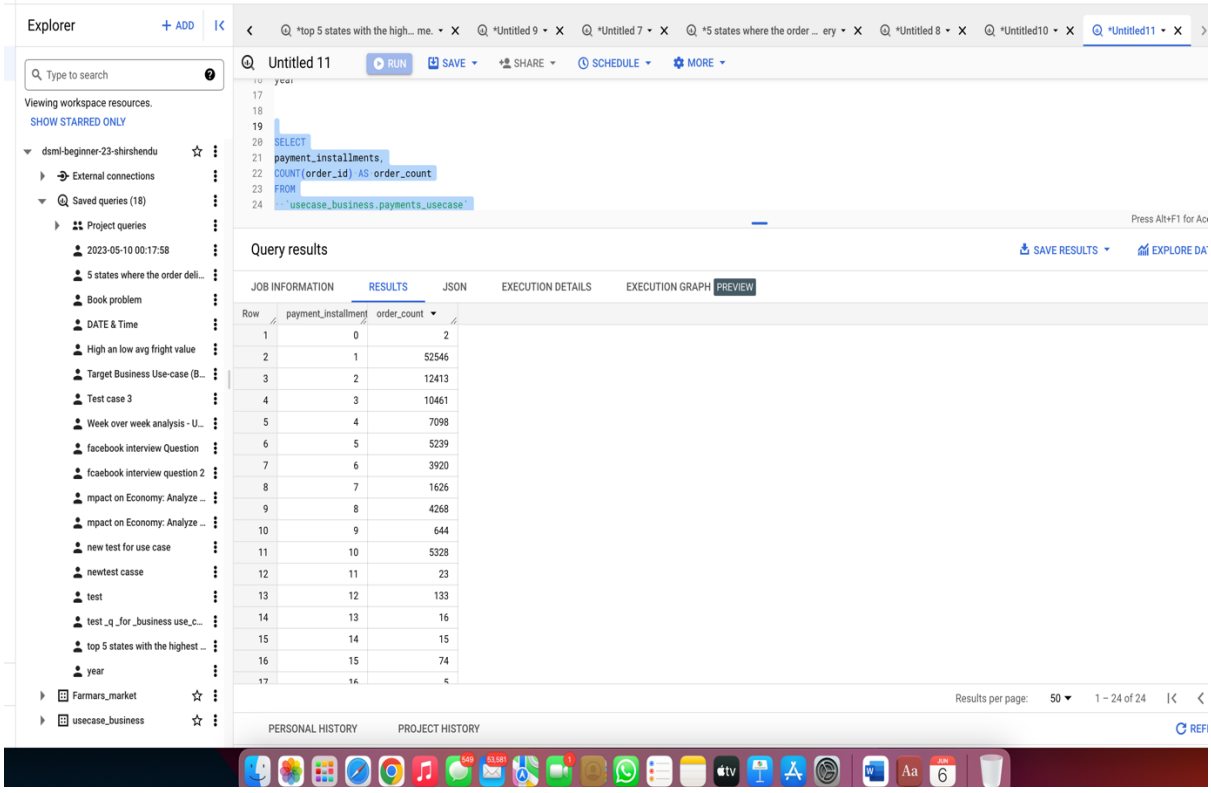
Insights : The month-to-month variation in the number of orders placed using different payment types. By grouping the data based on payment types, you can identify trends and patterns and then understand the payment behaviour of the customer

6.2 Find the no. of orders placed on the basis of the payment instalments that have been paid.

Code :

```
SELECT
payment_installments,
COUNT(order_id) AS order_count
FROM
  `usecase_business.payments_usecase`
GROUP BY
  payment_installments
```

Screenshot:



Explanation :

The main query selects the payment installments and counts the number of order IDs.It retrieves data from the payments_usecase table.
The result is grouped by payment installments to calculate the count of order IDs for each installment count.
Insights : The distribution of orders based on the number of payment instalments that have been paid. This information can be useful for understanding customer payment behaviour and preferences