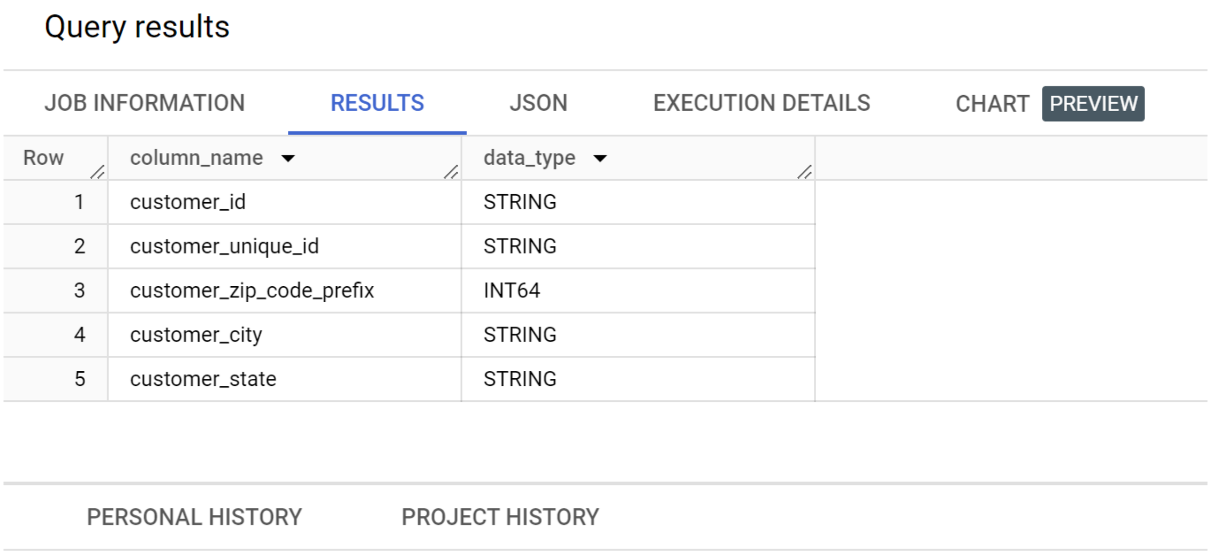
TARGET BUSINESS CASE STUDY

I. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

A. Data type of all columns in the "customers" table.

Ans-
```
select column_name, data_type
  from target.INFORMATION_SCHEMA.COLUMNS
  where table_name = "customers";
```



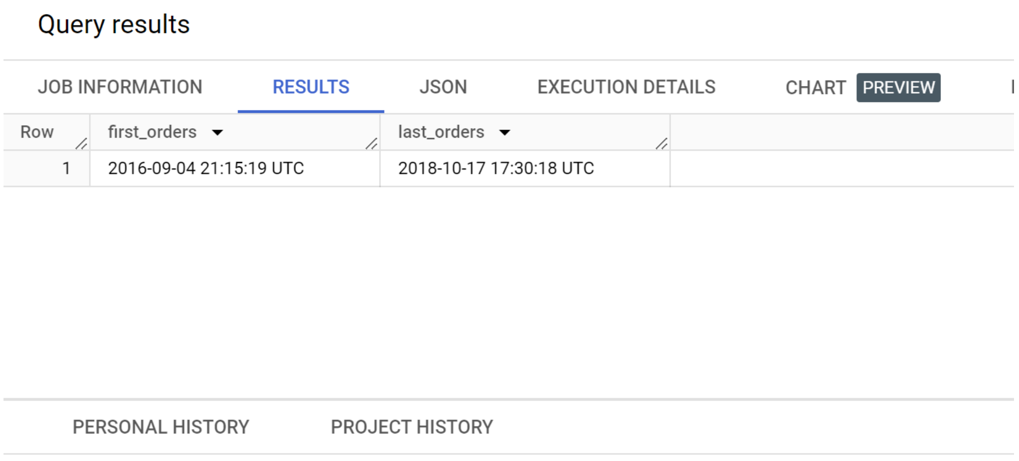
Query results

Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

Insights- Here, we got the datatype of different columns of customer table.

B. Get the time range between which the orders were placed.

Ans-
```
select min(order_purchase_timestamp) as first_orders,
  max(order_purchase_timestamp) as last_orders
  from target.orders;
```



Query results

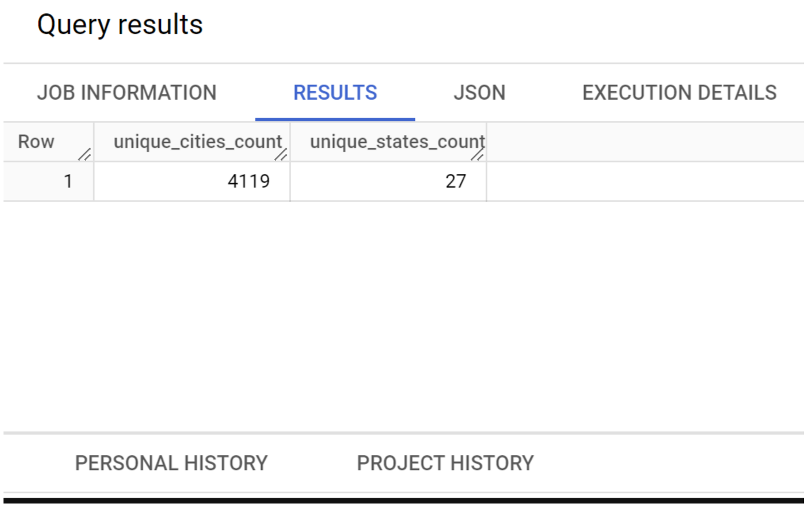
Row	first_orders	last_orders
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

Insight- we see that the time period of the date is from 2016-09-04 21:15:19 UTC to 2018-10-17 17:30:18 UTC. Around 2 years

C. Count the Cities & States of customers who ordered during the given period.

Ans-

```
SELECT
COUNT(DISTINCT c.customer_city) AS unique_cities,
COUNT(DISTINCT c.customer_state) AS unique_states
FROM
target.customers c
INNER JOIN
target.orders o ON c.customer_id = o.customer_id
WHERE
o.order_purchase_timestamp BETWEEN '2016-09-04 21:15:19' AND '2018-10-
17 17:30:18';
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	unique_cities_count	unique_states_count
1	4119	27

PERSONAL HISTORY | PROJECT HISTORY

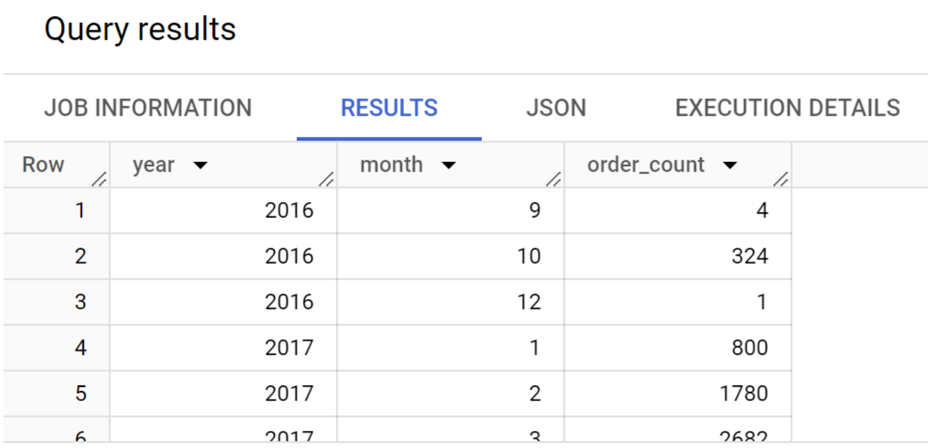
Insight – There were around 4119 unique cities from 27 states where customers had placed their order.

II. In-depth Exploration:

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

Ans-

```
SELECT
EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
COUNT(DISTINCT o.order_id) AS order_count
FROM
`target.orders` o
JOIN
`target.customers` c
ON
o.customer_id = c.customer_id
GROUP BY
year, month
ORDER BY
year, month;
```


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

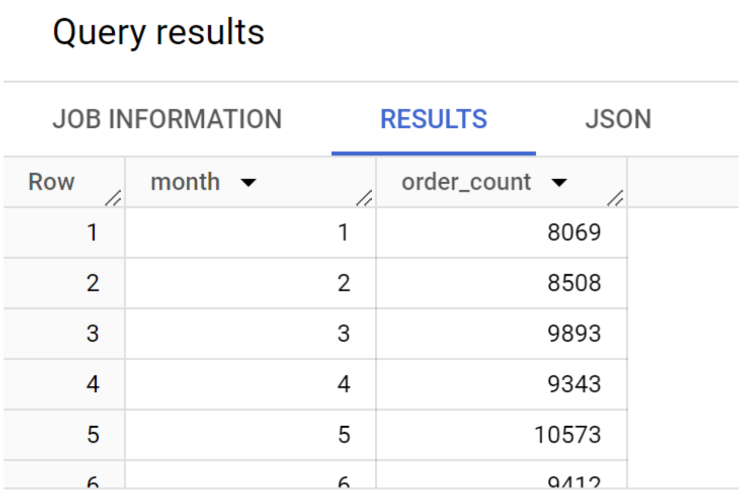
Row	year	month	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682

Insight :- We see orders were less in 2016. However, from 2017 the number of orders has increased rapidly. In Nov 2017 it was highest , sales peak and they fluctuate in 2018 come back down by year end of 2018.

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

Ans-

```
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
COUNT(DISTINCT order_id) AS order_count
FROM
 `target.orders`
GROUP BY
 month
ORDER BY
month;
```


Query results

JOB INFORMATION | RESULTS | JSON

Row	month	order_count
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412

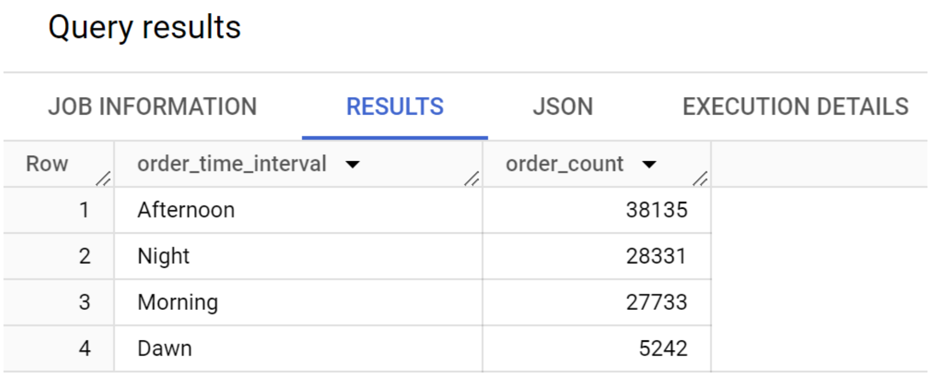
Insight:- we see that sales peak in the mid-year period during the months of May, July and August..

C. 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

Ans:-

```
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'
ELSE 'Unknown'
END AS order_time_interval,
COUNT(*) AS order_count
FROM
`target.orders`
GROUP BY
order_time_interval
ORDER BY
order_count DESC;
```



Query results

Row	order_time_interval	order_count
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

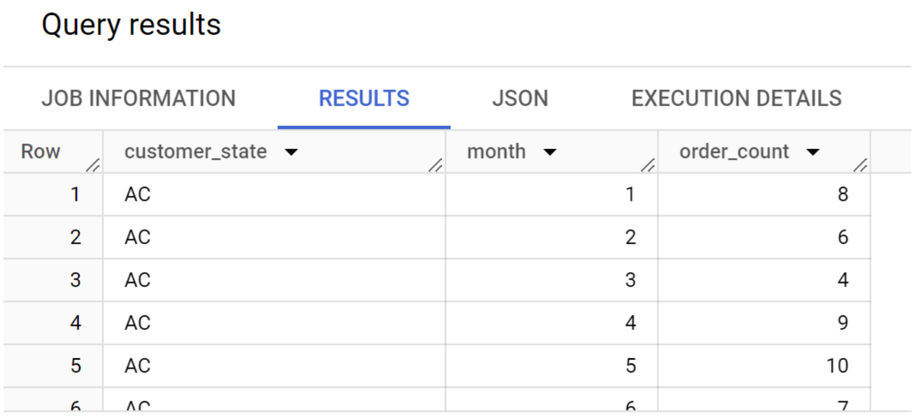
Insight:- this query will give the most popular time interval (Dawn, Morning, Afternoon, or Night) during which Brazilian customers mostly place their orders.it shows most order were placed during afternoon.

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

A. Get the month on month no. of orders placed in each state.

Ans:-

```
SELECT
c.customer_state,
EXTRACT(month FROM o.order_purchase_timestamp) AS month,
COUNT(o.order_id) AS order_count
FROM
target.orders o
JOIN
target.customers c
ON
o.customer_id = c.customer_id
GROUP BY
c.customer_state, month
ORDER BY
c.customer_state, month;
```



Query results

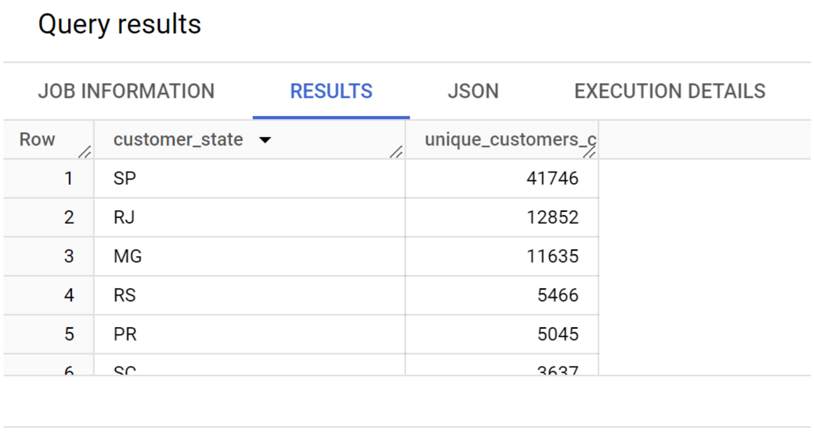
Row	customer_state	month	order_count
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7

Insight:- the query shows the result categorized in way where it shows count of order place for each state in each month from 1-12 then again shows the data for next state again with each month and so on.

B. How are the customers distributed across all the states?

Ans:-

```
SELECT
customer_state,
COUNT(DISTINCT customer_id) AS unique_customers_count
FROM
`target.customers`
GROUP BY
 customer_state
ORDER BY
 unique_customers_count DESC;
```



Query results

Row	customer_state	unique_customers_c
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637

Insight:- query shows the distribution of unique customers across all the states by grouping states column in customers table and counting the customer id.

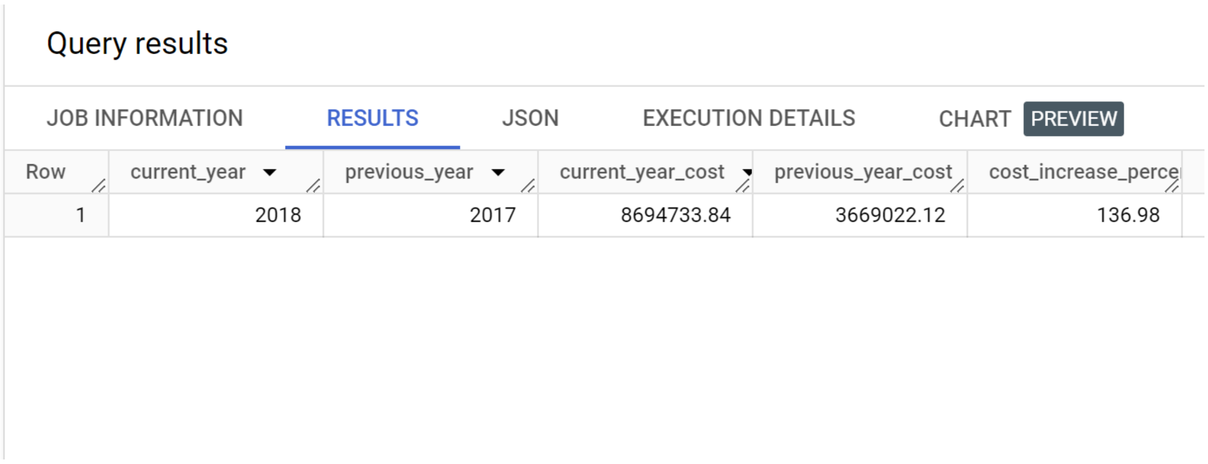
IV. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

A. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Ans:-

```
WITH OrderCosts AS (
SELECT
 EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
 EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
 round(SUM(p.payment_value),2) AS total_cost
 FROM
 `target.orders` o
JOIN
 `target.payments` 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
GROUP BY
 order_year, order_month
)

SELECT
2018 AS current_year,
2017 AS previous_year,
SUM(CASE WHEN order_year = 2018 THEN total_cost ELSE 0 END) AS
current_year_cost,
SUM(CASE WHEN order_year = 2017 THEN total_cost ELSE 0 END) AS
previous_year_cost,
round((SUM(CASE WHEN order_year = 2018 THEN total_cost ELSE 0 END) -
SUM(CASE WHEN order_year = 2017 THEN total_cost ELSE 0 END)) /
SUM(CASE WHEN order_year = 2017 THEN total_cost ELSE 0 END) * 100,2) AS
cost_increase_percentage
FROM
OrderCosts
WHERE
order_year IN (2017, 2018);
```


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW

Row	current_year	previous_year	current_year_cost	previous_year_cost	cost_increase_perce
1	2018	2017	8694733.84	3669022.12	136.98

Insight:- With conditional aggregation to sum the payment values for given years and month. The other query calculates the percentage increase in costs by comparing the total costs of orders in 2018 to those in 2017 which is 136.98%.

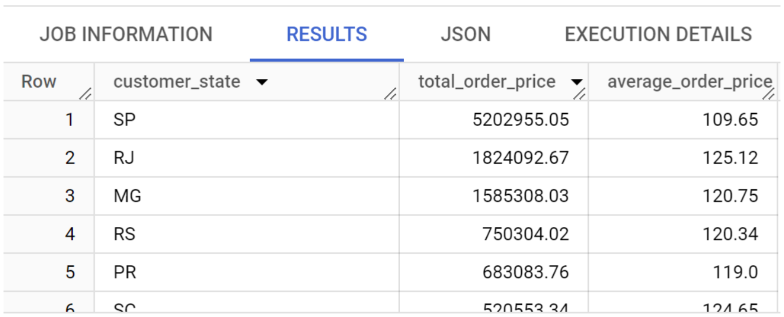
B. Calculate the Total & Average value of order price for each state.

Ans:-

```
SELECT
c.customer_state,
round(SUM(oi.price),2) AS total_order_price,
round(AVG(oi.price),2) AS average_order_price
FROM
`target.customers` c
```

```
JOIN
`target.orders` o ON c.customer_id = o.customer_id
JOIN
`target.order-items` oi ON o.order_id = oi.order_id
GROUP BY
 c.customer_state
ORDER BY
total_order_price desc,average_order_price desc;
```

Query results



JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

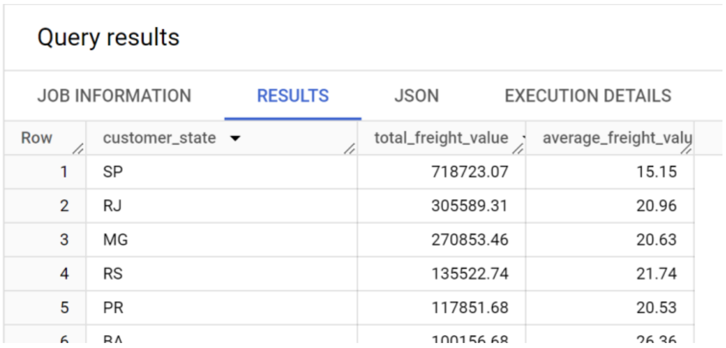
Row	customer_state	total_order_price	average_order_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65

Insight:- JOIN clauses is used to join the Customers, Orders, and Order_Items tables to get the necessary data. Then SUM() function is used to calculates the total order price for each state .The AVG() function calculates the average order price for each state. Then The GROUP BY clause is used to groups the results by customer_state we found Sao paolo has highest total order price and lowest average price.

C. Calculate the Total & Average value of order freight for each state.

Ans:-

```
SELECT
c.customer_state,
round(SUM(oi.freight_value),2) AS total_freight_value,
round(AVG(oi.freight_value),2) AS average_freight_value
FROM
`target.customers` c
JOIN
`target.orders` o ON c.customer_id = o.customer_id
JOIN
`target.order-items` oi ON o.order_id = oi.order_id
GROUP BY
c.customer_state
ORDER BY
total_freight_value desc,average_freight_value desc;
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_freight_value	average_freight_valu
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36

Insight:- JOIN clauses is used to join the Customers, Orders, and Order_Items tables to get the necessary data. Then SUM() function is used to calculates the total freight value for each state .The

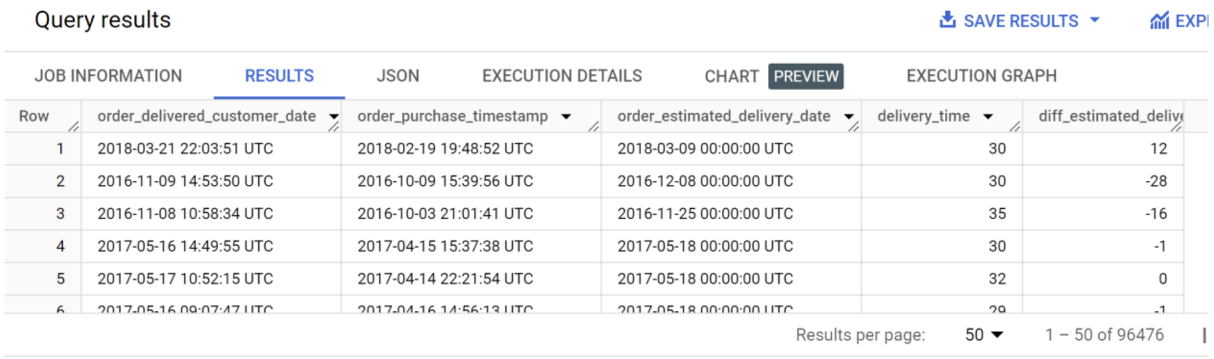
AVG() function calculates the average freight value for each state. Then The GROUP BY clause is used to groups the results by customer_state we found sao paolo has highest total freight value and least average freight value.

V. Analysis based on sales, freight and delivery time.

A. 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

Ans:-

```
SELECT
o.order_delivered_customer_date,o.order_purchase_timestamp,o.order_estimated
_delivery_date,
DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)
AS delivery_time,
DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date,
DAY) AS diff_estimated_delivery
FROM
`target.orders` o
WHERE
o.order_delivered_customer_date IS NOT NULL;
```



Row	order_delivered_customer_date	order_purchase_timestamp	order_estimated_delivery_date	delivery_time	diff_estimated_delive
1	2018-03-21 22:03:51 UTC	2018-02-19 19:48:52 UTC	2018-03-09 00:00:00 UTC	30	12
2	2016-11-09 14:53:50 UTC	2016-10-09 15:39:56 UTC	2016-12-08 00:00:00 UTC	30	-28
3	2016-11-08 10:58:34 UTC	2016-10-03 21:01:41 UTC	2016-11-25 00:00:00 UTC	35	-16
4	2017-05-16 14:49:55 UTC	2017-04-15 15:37:38 UTC	2017-05-18 00:00:00 UTC	30	-1
5	2017-05-17 10:52:15 UTC	2017-04-14 22:21:54 UTC	2017-05-18 00:00:00 UTC	32	0
6	2017-05-16 09:07:47 UTC	2017-04-16 14:56:13 UTC	2017-05-18 00:00:00 UTC	29	-1

Results per page: 50   1 – 50 of 96476

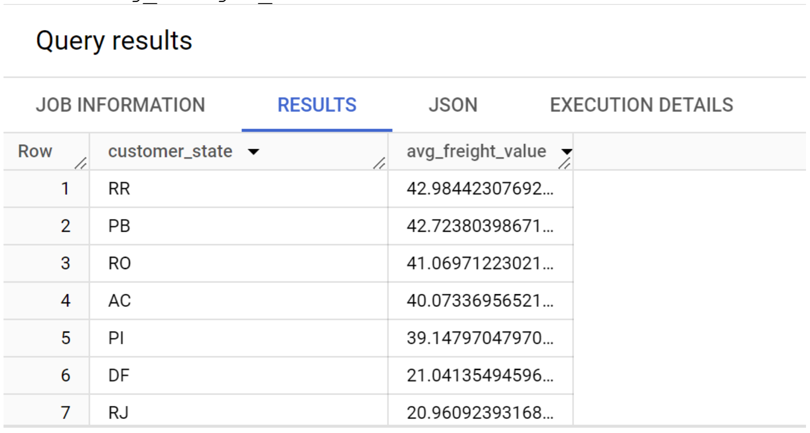
Insight:- this query will give the `delivered_customer_date, order_purchase_timestamp`, `order_estimated_delivery_date` ,delivery_time (in days), and diff_estimated_actual_delivery (in days) for each order, representing the delivery time and the difference between the estimated and actual delivery dates in a single query.

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

Ans:-

```
SELECT
customer_state,
avg_freight_value,
FROM (
SELECT
  c.customer_state,
  AVG(oi.freight_value) AS avg_freight_value,
  ROW_NUMBER() OVER (ORDER BY AVG(oi.freight_value) DESC) AS top_rank,
  ROW_NUMBER() OVER (ORDER BY AVG(oi.freight_value) ASC) AS bottom_rank
FROM
  `target.customers` c
JOIN
  `target.orders` o ON c.customer_id = o.customer_id
JOIN
  `target.order-items` oi ON o.order_id = oi.order_id
GROUP BY
```

```
  c.customer_state
)
WHERE
top_rank <= 5 OR bottom_rank <= 5
ORDER BY
avg_freight_value DESC;
```


Query results

Row	customer_state	avg_freight_value
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...
6	DF	21.04135494596...
7	RJ	20.96092393168...

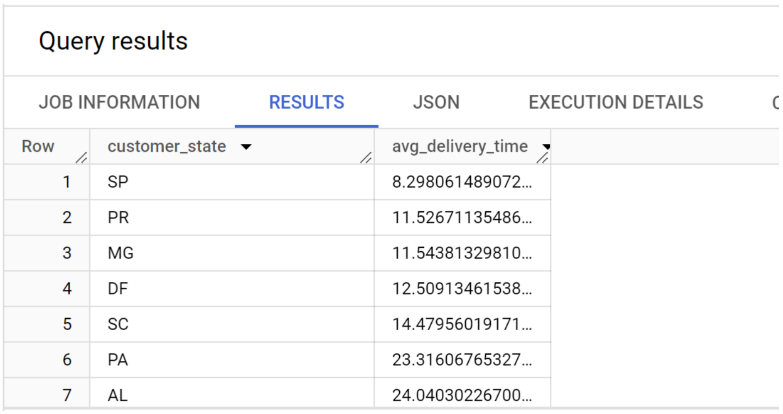
Insight:- here it calculates the average freight value for each state and assigns row numbers to the results based on both ascending and descending order of average freight values. Then it filters the top 5 and bottom 5 states based on the row numbers.

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

Ans:-

```
WITH StateDeliveryAverage AS (
SELECT
  c.customer_state,
  AVG(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,DA
Y)) AS avg_delivery_time
FROM
  `target.customers` c
JOIN
  `target.orders` o ON c.customer_id = o.customer_id
 WHERE
  o.order_delivered_customer_date IS NOT NULL
GROUP BY
  c.customer_state
)

SELECT
customer_state,
avg_delivery_time
FROM (
SELECT
  customer_state,
  avg_delivery_time,
  ROW_NUMBER() OVER (ORDER BY avg_delivery_time ASC) AS lowest_rank,
  ROW_NUMBER() OVER (ORDER BY avg_delivery_time DESC) AS highest_rank
FROM
  StateDeliveryAverage
) AS ranked_data
WHERE
lowest_rank <= 5 OR highest_rank <= 5
ORDER BY avg_delivery_time ASC;
```


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

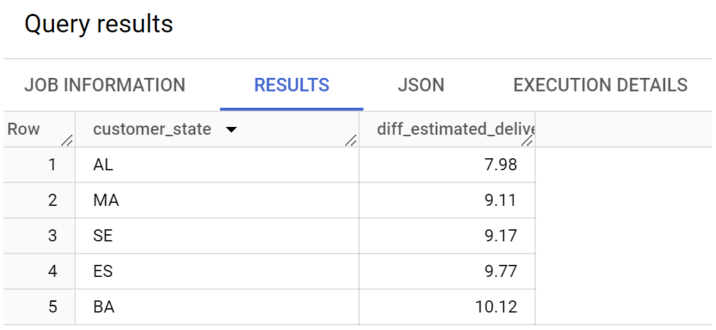
Row	customer_state	avg_delivery_time
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...
6	PA	23.31606765327...
7	AL	24.04030226700...

Insight:- Top 5 row show the top 5 state along with the lowest average delivery time and bottom 5 row shows the top 5 states with the highest average delivery times arranged in increasing order.

D. 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.

Ans:-

```
SELECT
c.customer_state,
round(AVG(datetime_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date,day)),2) as diff_estimated_delivery
FROM
`target.orders` o
JOIN
`target.order-items` i ON o.order_id = i.order_id
JOIN
`target.customers` c ON o.customer_id = c.customer_id
WHERE
o.order_status = 'delivered'
GROUP BY
c.customer_state
ORDER BY
diff_estimated_delivery
limit 5;
```


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	diff_estimated_delive
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

Insight:- calculatIng the difference in days between order_delivered_customer_date and order_estimated_delivery_date, and the results are ordered in ascending order of this difference filtering where the status is delivered This will give the top 5 states where the order delivery is fastest compared to the estimated date of delivery.

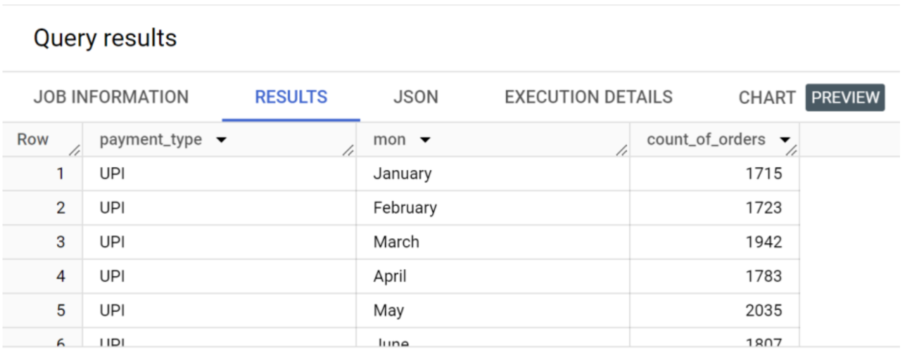
VI. Analysis based on the payments:

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

Ans:-
```
WITH sub AS (

SELECT *,

FORMAT_DATETIME("%B", DATETIME (order_purchase_timestamp)) AS mon,
EXTRACT (MONTH FROM order_purchase_timestamp) AS mon_no
FROM `target.orders`
)
SELECT payment_type, mon, COUNT (DISTINCT sub.order_id) AS
count_of_orders
FROM sub
JOIN `target.payments` p
ON sub.order_id = p.order_id
GROUP BY payment_type, mon, mon_no
ORDER BY payment_type,mon_no;
```

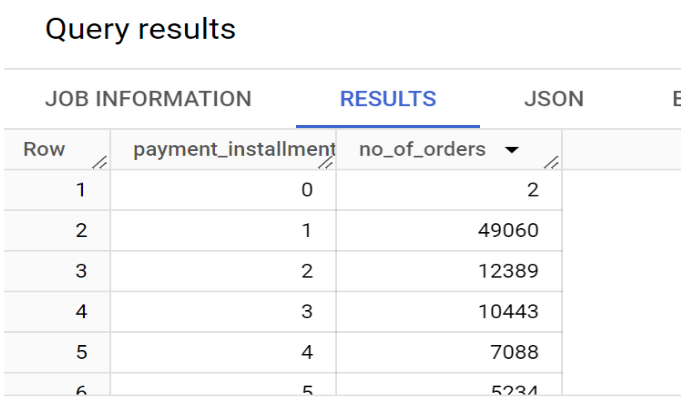

Query results

Row	payment_type	mon	count_of_orders
1	UPI	January	1715
2	UPI	February	1723
3	UPI	March	1942
4	UPI	April	1783
5	UPI	May	2035
6	UPI	June	1807

Insight:- joining the orders table with the payments table on the order_id to link payment information to each order. The result set is grouped by the order month and payment type, allowing us to count the number of orders placed using each payment method in each month. We see that no. of orders steadily increase month over month for all payment types up until august and then it drastically falls. Credit card payments are the highest.

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

Ans:-
```
SELECT payment_installments, COUNT (DISTINCT order_id) AS no_of_orders
FROM `target.payments`
GROUP BY payment_installments
ORDER BY payment_installments;
```


Query results

JOB INFORMATION RESULTS JSON E

Row	payment_installment	no_of_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234

Insight:- the query groups the payments by the number of instalments and then counts the number of distinct orders within each group and will show how many orders are associated with each number of payment instalments and We can observe the number of one-time purchases is highest.