Business Case: Target SQL

NOTE: All the Queries ran into the BIGQUERY.

1. 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.

SQL Code:

```
Select

       column_name,
       data_type,
       COLUMN_DEFAULT,
       IS_NULLABLE
from
       `Target_case_study.INFORMATION_SCHEMA.COLUMNS`
where
       table_name = "customers"
```

Query o/p:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	column_name	data_type	COLUMN_DEFAULT	IS_NULLABLE
1	customer_id	STRING	NULL	YES
2	customer_unique_id	STRING	NULL	YES
3	customer_zip_code_prefix	INT64	NULL	YES
4	customer_city	STRING	NULL	YES
5	customer_state	STRING	NULL	YES

Insides: “Provides an overview of the data type and content within each column, offering insights into the structure and characteristics of the dataset.”

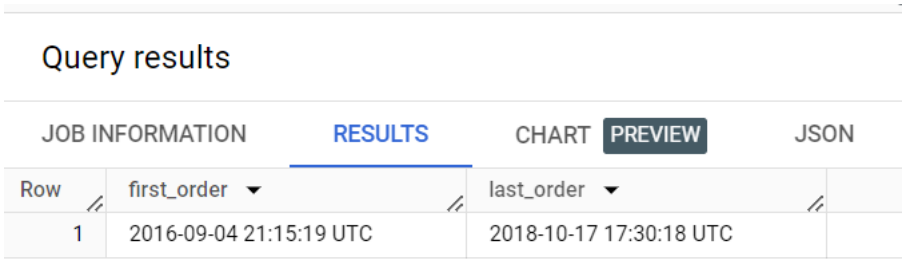
Recommendation:

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

Query:

```
SELECT
  min(order_purchase_timestamp) as first_order,
  max(order_purchase_timestamp) as last_order
 FROM
  `ultra-mason-406201.Target_case_study.orders`
```

Query Output:



Insides: The Given data set is from “2016 -09-04” to “2018-10-17”.

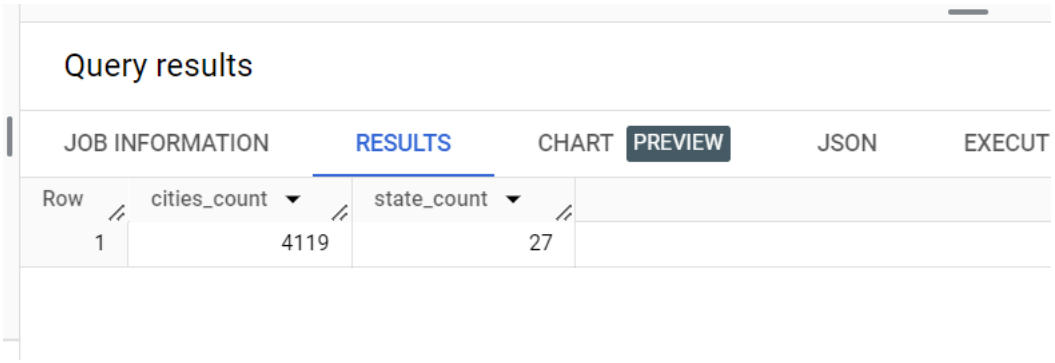
Recommendation:

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

Query:

```
SELECT
  count(DISTINCT c.customer_city) as cities_count,
  count(DISTINCT c.customer_state) as state_count
 FROM
  `ultra-mason-406201.Target_case_study.orders`  o
  inner join `Target_case_study.customers` c
  on o.customer_id = c.customer_id
```

Query Output:



Insides: “We have total count of state as 27 and cities count as 4119”

Recommendation: “Now that we have the count of cities and states, considering the possibility of any remaining cities provides an opportunity to contemplate expanding our business to those locations.”

2.In-depth Exploration:

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

Query:

```
with a as
(SELECT
  extract(year from order_purchase_timestamp) as count_orders,
 count(DISTINCT order_id) as count_a
 FROM
  `ultra-mason-406201.Target_case_study.orders`
 group by
  count_orders
order by
  count_orders
),
b as
(
  select
  count_orders,
  count_a,
  lag(count_a,1) over (order by count_orders desc) as future_values
from a
order by count_orders asc
)

select *,
  case
  when future_values is null or count_a < future_values then
"Order_Increased_from_last_year"
  ELSE "Order_has_not_increased_from_last_year"
  end
from b
```

Query O/p:

Query results

SAVE

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECU

Row	count_orders	count_a	future_values	f0_
1	2016	329	45101	Order_Increased_from_last_year
2	2017	45101	54011	Order_Increased_from_last_year
3	2018	54011	null	Order_Increased_from_last_year

Insides: Yes, there Is growing trend observed in the data.

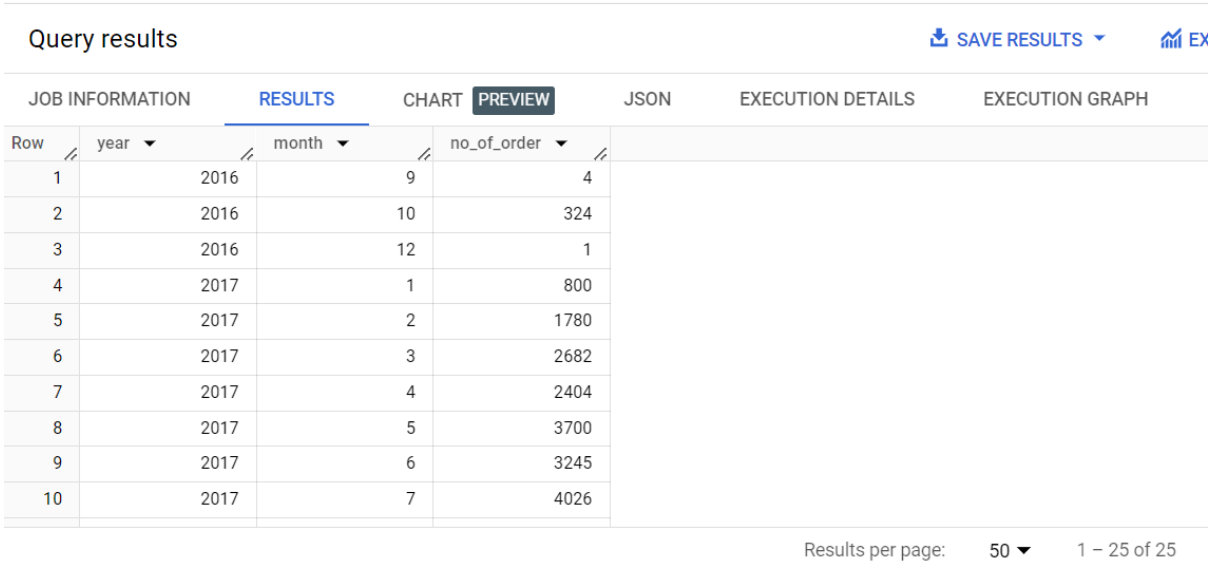
Recommendation: Seems like there is demand growing in the market, it is good time to thing on the discount and offer on product to be applied so that the grows is continues in the upcoming years as well.

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

Query:

```
 SELECT
  extract(year from order_purchase_timestamp) as year,
  extract(month from order_purchase_timestamp) as month,
 count(DISTINCT order_id) as no_of_order
 FROM
  `ultra-mason-406201.Target_case_study.orders`
 group by
  year,month
order by
   year,month
```

Query Output:



Row	year	month	no_of_order
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Insides: “There is a noticeable peak in activity during the 7th and 8th months for both 2017 and 2018.”

Recommendation:
"If there are any festivals during these months, we can consider bundling products with lower demand along with high-demand products to optimize sales and meet customer preferences."

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

i. 0-6 hrs : Dawn
ii. 7-12 hrs : Mornings
iii. 13-18 hrs : Afternoon
iv. 19-23 hrs : Night

Query:

```
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
"Mornings"
      WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
"Afternoon"
      ELSE "Night"
    END AS Interval_of_day,
 count(Distinct order_id) as count_of_ordered_placed_by_Brazil
 FROM
  `ultra-mason-406201.Target_case_study.orders`
where
  customer_id in (SELECT customer_id FROM `ultra-mason-
406201.Target_case_study.customers`)
 group by
  Interval_of_day
order by
  Interval_of_day
```

Query O/p:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUT

Row	Interval_of_day	count_of_ordered_placed_by_Brazil
1	Afternoon	38135
2	Dawn	5242
3	Mornings	27733
4	Night	28331

Insides: “Based on the analysis of order placement timestamps for Brazilian customers, it appears that the majority of orders are placed in the "Afternoon." This conclusion is drawn from categorizing order timestamps into different time periods, where the highest frequency falls within the afternoon hours. Therefore, it can be stated that, on average, Brazilian customers tend to place their orders in the afternoon”

Recommendation: “Observing that the highest number of orders occurs during the morning and afternoon periods suggests the idea of maintaining additional stock during these times to meet the increased demand.”

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

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

Query:

```
with a as
(SELECT
  c.customer_state,
  EXTRACT(YEAR FROM order_purchase_timestamp) as year,
  EXTRACT(month FROM order_purchase_timestamp) as month,
  count(Distinct o.order_id) as count_orders_state_wise
 FROM `ultra-mason-406201.Target_case_study.orders` o
 inner join `ultra-mason-406201.Target_case_study.customers` c using(customer_id)
group by
  c.customer_state,year,month
order by
  year,month,count_orders_state_wise desc
)
select * from a
```

Query o/p:

Query results

SAVE RESULTS EXP

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	year	month	count_orders_state_wise
1	SP	2016	9	2
2	RR	2016	9	1
3	RS	2016	9	1
4	SP	2016	10	113
5	RJ	2016	10	56
6	MG	2016	10	40
7	RS	2016	10	24
8	PR	2016	10	19
9	SC	2016	10	11
10	GO	2016	10	9

Results per page: 50 1 – 50 of 565

Insides:
"In our analysis, we have observed both the maximum and minimum values of orders placed in each month and year. This information proves invaluable in determining the highest and lowest order counts recorded for each state, providing valuable insights into the variation in order placement across different states during specific time periods."

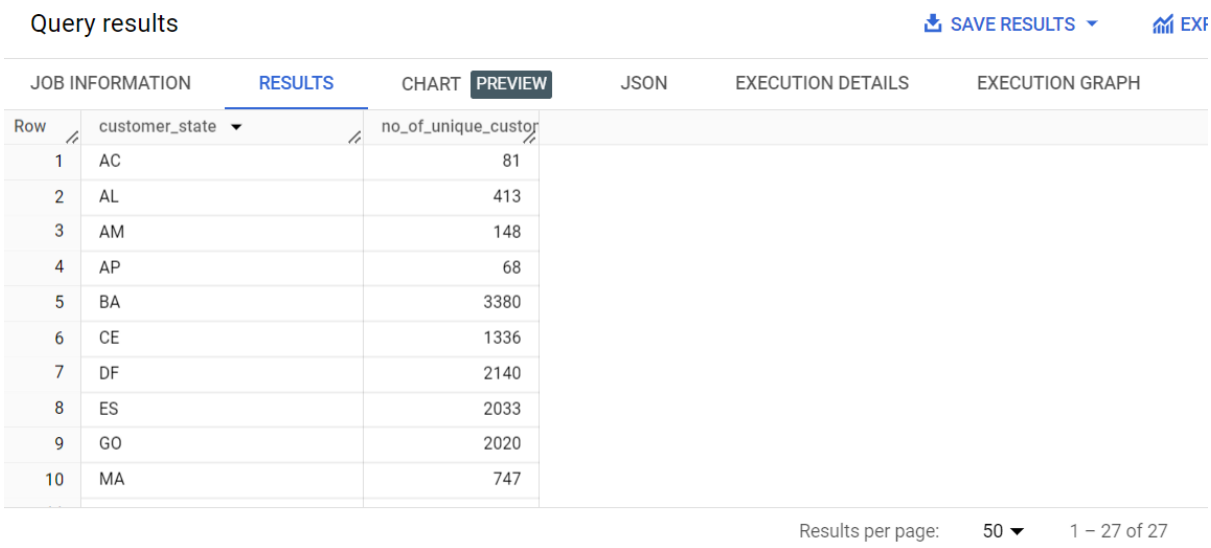
Recommendation:
"We should place a greater emphasis on locations with the minimum order placements. Understanding the reasons behind the lower order volumes in these areas can offer valuable insights into potential challenges or opportunities that may be influencing customer behaviour, enabling us to formulate targeted strategies for improvement."

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

Query:

```
SELECT
  customer_state,
  count(distinct customer_id) no_of_unique_customers
from
 `ultra-mason-406201.Target_case_study.customers`
  inner join `ultra-mason-406201.Target_case_study.orders`
  using(customer_id)
group by
  customer_state
order by
  customer_state
```

Query o/p:



Insides: "We have the count of customers in each state."

Recommendation:
"In areas where the customer count is lower, it is imperative to initiate a thorough analysis to uncover the reasons behind this phenomenon. Subsequently, we can implement necessary actions and strategies aimed at boosting customer engagement and increasing the customer count in those specific regions."

4.Impact on Economy: Analyse 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).
You can use the "payment_value" column in the payments table to get the cost of orders.

Query:

```
with A as
(SELECT
  a.order_id,
  a.order_purchase_timestamp,
  EXTRACT(YEAR FROM a.order_purchase_timestamp) AS year,
  FORMAT_DATE('%b', TIMESTAMP_TRUNC(a.order_purchase_timestamp, MONTH)) AS month,
  b.payment_value
from
 `ultra-mason-406201.Target_case_study.orders`  a
 inner join
 (SELECT Distinct
  order_id,
  round(sum(payment_value),2) as payment_value
  FROM `ultra-mason-406201.Target_case_study.payments`
group by
  order_id
) b
using (order_id)
),
B as
(
select * from A
where month in ("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug")
),
C as
(
  select
    year,
    sum(payment_value) as sum_payment_value
  from B
  group by
    year
  order by
    year
),
D as
(
select
  year,
  sum_payment_value,
  lead(sum_payment_value) over (order by year asc) as lead_sum_payment_value,
from C
  order by
    year
```

```
)
select
year,
  sum_payment_value,
  lead_sum_payment_value,
  (lead_sum_payment_value - sum_payment_value) / sum_payment_value *100 as
per_increase_in_the_cost_of_orders
from D
```

Query o/p:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

Row	year	sum_payment_value	lead_sum_payment_v	per_increase_in_the
1	2017	3669022.119999...	8694733.839999...	136.9768716466...
2	2018	8694733.839999...	null	null

Insides: "The cost of the order has experienced a substantial increase of **136.97%”**

Recommendation:

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

Query:

```
select Distinct
  c.customer_state,
  round(avg(oit.price),2) as average_price,
  round(sum(oit.price),2) as total_price
from
  `ultra-mason-406201.Target_case_study.order_items` oit
 inner join `ultra-mason-406201.Target_case_study.orders` o on oit.order_id =
o.order_id
 inner join `Target_case_study.customers` c on o.customer_id = c.customer_id
 group by
  c.customer_state
order by
  c.customer_state
```

Query o/p:

Query results

SAVE RESULTS | EXI

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	average_price	total_price
1	AC	173.73	15982.95
2	AL	180.89	80314.81
3	AM	135.5	22356.84
4	AP	164.32	13474.3
5	BA	134.6	511349.99
6	CE	153.76	227254.71
7	DF	125.77	302603.94
8	ES	121.91	275037.31
9	GO	126.27	294591.95
10	MA	145.2	119648.22

Results per page: 50 1 – 27 of 27

Insides: “We can obtain both the average and total prices, providing valuable metrics for further analysis in shaping our marketing strategy.”

Recommendation:

1. Implement a dynamic pricing strategy based on the average and total price data. This could involve adjusting prices based on demand, time of day, or other relevant factors.
2. Launch targeted promotional campaigns during periods of lower average order value to encourage higher spending. This could include discounts, bundles, or loyalty programs.

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

Query:

```
select Distinct
  c.customer_state,
  round(avg(oit.freight_value),2) as average_price,
  round(sum(oit.freight_value),2) as total_price
 FROM `ultra-mason-406201.Target_case_study.order_items` oit
 inner join `ultra-mason-406201.Target_case_study.orders` o on oit.order_id =
o.order_id
 inner join `Target_case_study.customers` c on o.customer_id = c.customer_id
 group by
  c.customer_state
order by
  c.customer_state
```

Query o/p:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	average_price	total_price
1	AC	40.07	3686.75
2	AL	35.84	15914.59
3	AM	33.21	5478.89
4	AP	34.01	2788.5
5	BA	26.36	100156.68
6	CE	32.71	48351.59
7	DF	21.04	50625.5
8	ES	22.06	49764.6
9	GO	22.77	53114.98
10	MA	38.26	31523.77

Results per page: 50 1 – 27 of 27

Insides: “We can obtain both the average and total freight, providing valuable metrics for further analysis in shaping our marketing strategy.”

Recommendation:

1. Identify states with high average freight costs and explore opportunities for cost optimization, such as negotiating better shipping rates or optimizing delivery routes.
2. Consider adjusting product prices or introducing targeted promotions in states with higher freight costs to maintain competitiveness while managing overall profitability.
3. Evaluate the efficiency of the supply chain in states with varying freight costs. Streamline logistics processes to reduce overall shipping expenses and enhance delivery timelines.

5.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.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

Query:

```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_deliver,
  date_diff(order_estimated_delivery_date,order_delivered_customer_date,DAY) as
diff_estimated_delivery
FROM
  `ultra-mason-406201.Target_case_study.orders`
where
  order_status = "delivered"
order by order_id
```

Query o/p:

Query results

SAVE RESULTS EX

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08b...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18

Results per page: 50 1 – 50 of 96478

Insides:
1.This query provides insights into the delivery process by presenting key parameters. It allows us to discern delivery dates, identify orders that have not been delivered, and pinpoint orders experiencing delayed deliveries.
2. diff_estimated_delivery **+ve sign indicates** that we have delivered the order **within expected time**.

Recommendation:
“The orders that have not been delivered represent a critical priority, denoted as the "**red alert**" status. In our query, these orders are highlighted and labelled as "order_not_yet_delivered," signalling the need for immediate attention and processing”

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

Query:

```
with
a as
(SELECT
  c.customer_state,
  round(avg(freight_value),2) AS Top5_avg_freight_values
FROM
  `ultra-mason-406201.Target_case_study.order_items` oit
 inner join `ultra-mason-406201.Target_case_study.orders` o on oit.order_id =
o.order_id
 inner join `Target_case_study.customers` c on o.customer_id = c.customer_id
GROUP BY
  c.customer_state
order by Top5_avg_freight_values desc
limit 5
),
b as
(
  select
  row_number() over (order by round(Top5_avg_freight_values,2) asc) as row_num,
  *
  from a
  order by
  Top5_avg_freight_values asc
),
c as
(
SELECT
  row_number() over (order by round(avg(freight_value),2) asc) as row_num,
  c.customer_state,
  round(avg(freight_value),2) AS bottom5_avg_freight_values
FROM
  `ultra-mason-406201.Target_case_study.order_items` oit
 inner join `ultra-mason-406201.Target_case_study.orders` o on oit.order_id =
o.order_id
 inner join `Target_case_study.customers` c on o.customer_id = c.customer_id
```

```
GROUP BY
  c.customer_state
order by bottom5_avg_freight_values asc
limit 5
),
d as
(
select
  b.row_num,
  b.customer_state,
  Top5_avg_freight_values,
  c.customer_state,
  bottom5_avg_freight_values
from
  b inner join c using(row_num)
)
select * from d
```

Query o/p:

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	row_num	customer_state	Top5_avg_freight_values	customer_state_1	bottom5_avg_freight_values
1	1	PI	39.15	SP	15.15
2	2	AC	40.07	PR	20.53
3	3	RO	41.07	MG	20.63
4	4	PB	42.72	RJ	20.96
5	5	RR	42.98	DF	21.04

Insides: “We have the top 5 states with the highest and lowest average Freight value”

Recommendation:

1. Communicate transparently with customers in states with higher freight costs. Consider implementing clear and informative shipping policies to manage customer expectations.
2. Optimize inventory management by considering the impact of freight costs

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

Query:

```
with a as
(SELECT
  c.customer_state,
  round(avg(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,Da
y)),2) AS top_5_avg_delivery_time_In_days
FROM
```

```
  `ultra-mason-406201.Target_case_study.orders` o
INNER JOIN
  `Target_case_study.customers` c
USING
  (customer_id)
GROUP BY
  c.customer_state
order by top_5_avg_delivery_time_In_days desc
limit 5
),
b as
(
  select
    row_number() over (order by top_5_avg_delivery_time_In_days asc) as row_num,
    *
  from a
),
c as
(
SELECT
  row_number() over (order by
round(avg(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,Day)
),2) asc) as row_num,
  c.customer_state,
  round(avg(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,Da
y)),2) AS Bottom_5_avg_delivery_time_In_days
FROM
  `ultra-mason-406201.Target_case_study.orders` o
INNER JOIN
  `Target_case_study.customers` c
USING
  (customer_id)
GROUP BY
  c.customer_state
order by Bottom_5_avg_delivery_time_In_days asc
limit 5
)
select
  b.row_num,
  b.customer_state,
  top_5_avg_delivery_time_In_days,
  c.customer_state,
  Bottom_5_avg_delivery_time_In_days
from
  b inner join c using(row_num)
```

Query o/p:

Query results

SAVE RESULTS | EXPLORE D

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	row_num	customer_state	top_5_avg_delivery_time_In_days	customer_state_1	Bottom_5_avg_delivery_time_In_days
1	1	PA	23.32	SP	8.3
2	2	AL	24.04	PR	11.53
3	3	AM	25.99	MG	11.54
4	4	AP	26.73	DF	12.51
5	5	RR	28.98	SC	14.48

Insides: “We have the top 5 states with the highest and lowest average delivery time”

Recommendation:

1. Investigate the reasons behind longer delivery times in these states. Optimize logistics and distribution channels to expedite the shipping process.
2. Consider establishing regional warehouses or distribution centers in these states to reduce transit times and improve overall delivery efficiency.

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.

Query:

```
SELECT
  c.customer_state,
  round(avg(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_da
te,DAY)),2) as diff_estimated_delivery
FROM
  `ultra-mason-406201.Target_case_study.orders` o
  inner join `Target_case_study.customers` c
  using (customer_id)
where
date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) is
not null
group by c.customer_state
order by diff_estimated_delivery desc
limit 5
```

Query o/p:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXE

Row	customer_state	diff_estimated_delivery
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

Insides:
1.Listed states are the most fast delivered states.
2. We are observing **positive values,** indicating that deliveries in this state are being accomplished in **less time** than the estimated delivery date.

Recommendation:
“Identifying states with slow delivery times enables us to focus on improving the efficiency of our delivery processes in those specific regions. This information allows us to pinpoint areas for optimization and enhance overall delivery performance”

6. Analysis based on the payments:

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

Query:

```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  EXTRACT(month FROM o.order_purchase_timestamp) AS month,
  p.payment_type,
  count(o.order_id) as order_count
FROM `ultra-mason-406201.Target_case_study.orders` o
  inner join `ultra-mason-406201.Target_case_study.payments` p using(order_id)
group by
  year,month,p.payment_type
order by
  year,month,p.payment_type
```

Query o/p:

Query results

SAVE RESULTS E

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	year	month	payment_type	order_count
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61

Results per page: 50 1 – 50 of 90

Insides: “The majority of orders, in terms of the highest transaction amounts, are observed to be processed using credit card payments.”

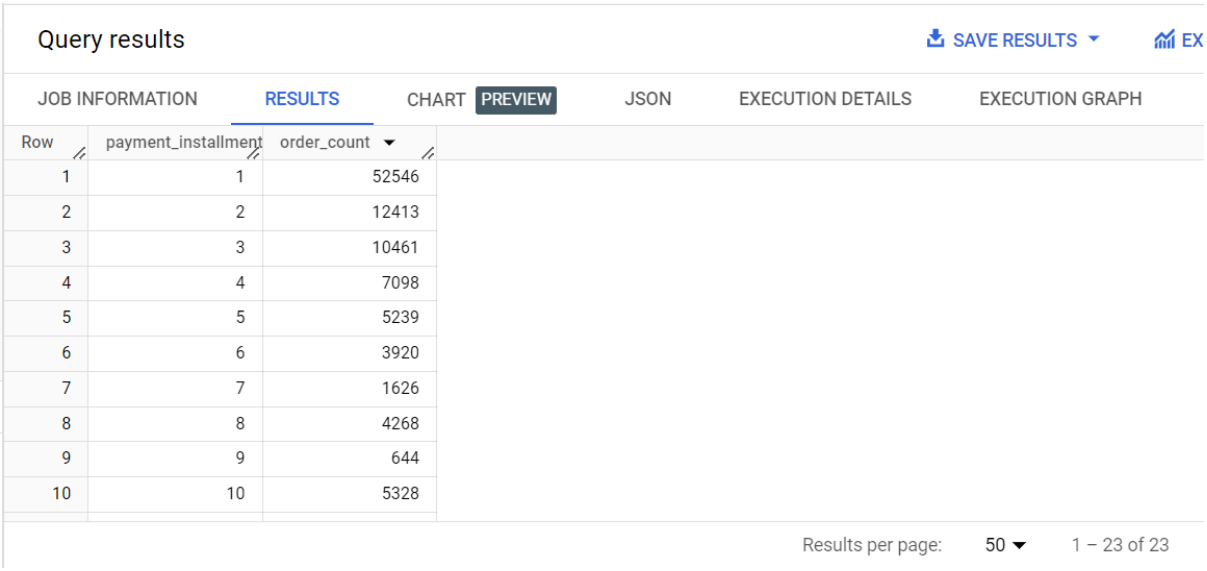
Recommendation:
1.Introducing a modest discount for transactions made through credit cards could potentially incentivize more users to opt for credit card payments.
2.To offset the potential discount loss, an alternative strategy could involve applying a nominal surcharge on credit card transactions while simultaneously offering a discount that effectively restores the product to its original price. This approach provides users with a perception of enjoying discounts while helping mitigate the impact on revenue.

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

Query:

```
SELECT
  p.payment_installments,
  COUNT(o.order_id) AS order_count
FROM
  `ultra-mason-406201.Target_case_study.orders` o
INNER JOIN
  `ultra-mason-406201.Target_case_study.payments` p USING(order_id)
where
  p.payment_installments >= 1
GROUP BY
  p.payment_installments
ORDER BY
  p.payment_installments;
```

Query o/p:



Insides: “We aim to understand the payment behaviour of our users by determining the number of individuals who opt for a single installment versus those who choose multiple installments.”

Recommendation:

1. Design targeted promotional campaigns or discounts for customers who opt for a higher number of payment installments. This could encourage more customers to choose installment options.
2. Implement loyalty programs that reward customers based on the number of payment installments chosen. This could foster customer loyalty and repeat business.