TARGET CASE STUDY

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.

```
select
  column_name,
  data_type
from `target-case-study2.target_sql_2.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
```

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

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

```
select
  min(order_purchase_timestamp) as first_order,
  max(order_purchase_timestamp) as last_order
from `Target_sql.orders`
```

Query results

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

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

INSIGHTS:

The FIRST order was made in 2016, and the LAST order was placed in 2018.

C. Count the number of Cities and States in our dataset.

```
select
  count(distinct customer_city) as no_of_cities,
  count(distinct customer_state) as no_of_state
from `Target_sql.customers`
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	no_of_cities	no_of_state
1	4119	27

INSIGHTS:

No.of.cities - 4119

No.of.states - 27

II. In-depth Exploration:

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

```
select
  extract(month from order_purchase_timestamp) as month,
  extract(year from order_purchase_timestamp) as year,
  count(distinct order_id) as no_of_orders
from `Target_sql.orders`
group by month, year
order by year, month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

Insights: **The data indicates growing trend in the number of orders placed from 2017 to 2018. However, it's noteworthy that there is a drop in 2018. It's important to note that this drop is not significant, suggesting a generally positive trend in order placements over the years. Business can use this information to adapt strategies and maintain growth. While there is an overall increasing trend in the number of orders placed over the past years, it's important to note that there are notably low order volumes during the 9th and 10th months, suggesting potential seasonality or specific factors influencing customer behavior during those periods. Businesses should consider this when planning further investigation and targeted strategies.**

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

```
1. select
  *
from(
  select
  month,
  year,
  no_of_orders,
  dense_rank() over(partition by year order by no_of_orders desc) as highest_order
  from(
    select
    extract(month from order_purchase_timestamp) as month,
    extract(year from order_purchase_timestamp) as year,
    count(distinct order_id) as no_of_orders
    from `Target_sql.orders`
group by month, year)as t) as t2
where highest_order = 1
order by year, month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART

Row	month	year	no_of_orders	highest_order
1	10	2016	324	1
2	11	2017	7544	1
3	1	2018	7269	1

Insights: **Monthly data reveals seasonality in order placements, notably with increased orders occurring in the months of November, and January. This pattern shows the potential influence of external factors or holidays or festivals during these months. The insight suggests that if a sale or discount announcement coincides with these months, it could potentially lead to higher sales.**

I have also obtained the quarterly sales report:

```
select
   extract(quarter from order_purchase_timestamp) as quarter,
   count(distinct order_id) as highest_orders
from `Target_sql.orders`
group by quarter
order by quarter
```

Query results

JOB INFORMATION | RESULTS | JSON

Row	quarter	highest_orders
1	1	26470
2	2	29328
3	3	25466
4	4	18177

Insights: **The data not only shows monthly seasonality in order placements but also highlights an increase in orders during the second quarter.**

C. During what time of the day, do the Brazilian customers mostly place their orders?

```
select
   extract(hour from order_purchase_timestamp) as hour,
   count(distinct order_id) as no_of_orders
from `Target_sql.orders`
group by hour
```

```
order by hour
```

JOB INFORMATION | RESULTS | JSON

Row	hour	no_of_orders
1	0	2394
2	1	1170
3	2	510
4	3	272
5	4	206
6	5	188
7	6	502
8	7	1231
9	8	2967
10	9	4785

Insights: **Brazilian customers tend to place orders throughout the day, including mornings, afternoons, and evenings. The varied ordering behavior implies that businesses serving these customers need to stay operationally prepared throughout the day to adapt to changing demand levels effectively.**

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

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

```
select
   c.customer_state,
   extract(month from order_purchase_timestamp) as month,
   count(distinct order_id) as no_of_orders
from `Target_sql.orders` as o
join `Target_sql.customers` as c
on o.customer_id = c.customer_id
group by month, c.customer_state
order by month, c.customer_state
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	month	no_of_orders
1	AC	1	8
2	AL	1	39
3	AM	1	12
4	AP	1	11
5	BA	1	264
6	CE	1	99
7	DF	1	151
8	ES	1	159
9	GO	1	164
10	MA	1	66

Insights: These insights offer information on the number of orders placed by each state, segmented by month. This data can help business understand order patterns and alter their strategies to meet specific regional demands.

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

```
select
  customer_state,
  count(distinct customer_id) as no_of_customers
from `Target_sql.customers`
group by customer_state
order by no_of_customers desc
```

Row	customer_state	no_of_customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

Insights: The data reveals that the state SP has the highest number of customers. This information can be valuable for business, indicating a strong market presence in SP, which may create allocation to meet the high demand consistently. This data can be beneficial for business to focus their marketing and operational efforts on regions with a high order volumes. Although State SP has the highest number of customers, it's essential to divert attention to other

states. These states AC, AP , RR require more focus and effort to improve their performance, even though they might not have the highest total sales. SP leads with the highest number of customers, while State RR trails behind with the lowest customer count. This data underscores the need to allocate more resources and marketing efforts to States (AC, AP, RR) to boost its customer base.

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).

```
with cte as (select
   pa.payment_value,
   extract(month from order_purchase_timestamp) as month,
   extract(year from order_purchase_timestamp) as year
from `Target_sql.orders` as ot
join `Target_sql.payments` as pa
on ot.order_id = pa.order_id),

cte2 as (select
    month,
    year,
    round(sum(payment_value)) as cost_of_orders
from cte
where (month between 1 and 8) and (year between 2017 and 2018)
group by month, year
order by month, year),

cte3 as (select
  *,
  lag(cte2.cost_of_orders) over(order by cte2.cost_of_orders) as diff
from cte2)
select
   *,
   round(100*(cte3.cost_of_orders - cte3.diff)/ cte3.diff) as increase_percentage
```

```
from cte3

where cte3.diff is not null

order by month, year, increase_percentage desc
```

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

Row	month	year	cost_of_orders	diff	increase_percentage
1	1	2018	1115004.0	1066541.0	5.0
2	2	2017	291908.0	138488.0	111.0
3	2	2018	992463.0	674396.0	47.0
4	3	2017	449864.0	417788.0	8.0
5	3	2018	1159652.0	1153982.0	0.0
6	4	2017	417788.0	291908.0	43.0
7	4	2018	1160785.0	1159652.0	0.0
8	5	2017	592919.0	592383.0	0.0
9	5	2018	1153982.0	1115004.0	3.0
10	6	2017	511276.0	449864.0	14.0

Insights: The insight reveals a massive percentage increase in the cost of orders during February 2017. This suggests that while February needs attention, it's also crucial to focus on improving sales in other months to ensure a balanced and profitable business performance throughout the year.

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

```
select

  round(sum(ot.price)) as total_price,

  round(avg(ot.price)) as avg_price,

  c.customer_state

from `Target_sql.order_items` as ot

join `Target_sql.orders` as o

on ot.order_id = o.order_id

join `Target_sql.customers` as c

on o.customer_id = c.customer_id

group by c.customer_state

order by total_price desc, avg_price desc
```

JOB INFORMATION | RESULTS | JSON | EXECUTIOI

Row	total_price	avg_price	customer_state
1	5202955.0	110.0	SP
2	1824093.0	125.0	RJ
3	1585308.0	121.0	MG
4	750304.0	120.0	RS
5	683084.0	119.0	PR
6	520553.0	125.0	SC
7	511350.0	135.0	BA
8	302604.0	126.0	DF
9	294592.0	126.0	GO
10	275037.0	122.0	ES

Insights: State SP leads with the highest total order values, but there are other states (AP, AC, RR) with notably lower total order values. While State SP is performing strongly, it's crucial to focus efforts on these lower-performing states to improve their order values and achieve a more balanced revenue distribution across all regions.

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

```
select
  round(sum(ot.freight_value)) as total_freight,
  round(avg(ot.freight_value)) as avg_freight,
  c.customer_state
from `Target_sql.order_items` as ot
join `Target_sql.orders` as o
on ot.order_id = o.order_id
join `Target_sql.customers` as c
on o.customer_id = c.customer_id
group by c.customer_state
order by total_freight desc, avg_freight desc
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	total_freight	avg_freight	customer_state
1	718723.0	15.0	SP
2	305589.0	21.0	RJ
3	270853.0	21.0	MG
4	135523.0	22.0	RS
5	117852.0	21.0	PR
6	100157.0	26.0	BA
7	89660.0	21.0	SC
8	59450.0	33.0	PE
9	53115.0	23.0	GO
10	50625.0	21.0	DF

Insights: The insight highlights that the state SP has the highest total freight value of orders. State RR has the lowest freight value of orders.

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

```
select

order_id,
timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day) as time_to_deliver,
timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date, day) as diff_estimated_delivery
from `Target_sql.orders`
where order_delivered_customer_date is not null
```

Row	order_id	time_to_deliver	diff_estimated_deliv
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	33	5

Insights: The insight clarifies that the provided data represents the number of days taken to deliver each order from the purchase date, indicating the delivery time for each order. It also highlights that the data represents the difference between the estimated delivery date and the actual delivery date of an order, indicating any delays or variances in delivery times.

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

```
with cte as(
 select
  c.customer_state,
  round(avg(ot.freight_value), 2) as freight,
  dense_rank() OVER(order by avg(ot.freight_value)asc) as rnk
from `Target_sql.customers` as c
join `Target_sql.orders` as o
on c.customer_id = o.customer_id
join `Target_sql.order_items` as ot
on o.order_id = ot.order_id
group by c.customer_state
order by freight asc
limit 5),

cte1 as (
 select
  c.customer_state,
  round(avg(ot.freight_value), 2) as freight,
  dense_rank() OVER(order by avg(ot.freight_value)desc) as rnk
from `Target_sql.customers` as c
join `Target_sql.orders` as o
on c.customer_id = o.customer_id
join `Target_sql.order_items` as ot
on o.order_id = ot.order_id
group by c.customer_state
order by freight desc
limit 5)

select
   customer_state,
   freight
from cte
union all
select
   customer_state,
   freight
from cte1
```

```
ORDER BY freight
```

Row	customer_state	freight
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04
6	PI	39.15
7	AC	40.07
8	RO	41.07
9	PB	42.72
10	RR	42.98

Insights: The analysis has uncovered the top 5 shipments with the most and least expensive average freight costs. This information can guide us to manage costs better, find unusual costs, and see where we can save money on shipping in our logistics operations.

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

```
with cte as(
  select
    c.customer_state,
    round(avg(timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day)), 2) as avg_delivery,
    dense_rank() OVER(order by avg(timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day))) as rnk
from `Target_sql.customers` as c
join `Target_sql.orders` as o
on c.customer_id = o.customer_id
join `Target_sql.order_items` as ot
on o.order_id = ot.order_id
group by c.customer_state
order by avg_delivery
limit 5),

cte1 as (
  select
    c.customer_state,
    round(avg(timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day)), 2) as avg_delivery,
    dense_rank() OVER(order by avg(timestamp_diff(order_delivered_customer_date,order_purchase_timestamp, day))desc) as
rnk
from `Target_sql.customers` as c
join `Target_sql.orders` as o
on c.customer_id = o.customer_id
join `Target_sql.order_items` as ot
on o.order_id = ot.order_id
group by c.customer_state
order by avg_delivery desc
limit 5)
```

```
select
   customer_state,
   avg_delivery
from cte
union all
select
   customer_state,
   avg_delivery
from cte1
ORDER BY avg_delivery
```

Row	customer_state	avg_delivery
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52
6	PA	23.3
7	AL	23.99
8	AM	25.96
9	AP	27.75
10	RR	27.83

Insights: Based on the analysis, I have identified the top 5 orders with the longest and shortest average delivery times. This data offers visibility into delivery efficiency, enabling us to focus on areas where improvements can be made to enhance customer satisfaction and improve the delivery operations. This information provides insights into order fulfillment efficiency, helping the business pinpoint areas for improvement in the logistics and delivery processes.

D. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery

```
with cte as(
  select
   c.customer_state,
   round(avg(timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date, day)), 2) as est_delivery,
   dense_rank() OVER(order by avg(timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date, day))) as
rnk
from `Target_sql.customers` as c
join `Target_sql.orders` as o
on c.customer_id = o.customer_id
join `Target_sql.order_items` as ot
on o.order_id = ot.order_id
where order_status = 'delivered'
group by c.customer_state
```

```
order by est_delivery
limit 5),

cte1 as (
 select
  c.customer_state,
  round(avg(timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date, day)), 2) as est_delivery,
  dense_rank() OVER(order by avg(timestamp_diff(order_delivered_customer_date,order_estimated_delivery_date, day))desc)
as rnk
from `Target_sql.customers` as c
join `Target_sql.orders` as o
on c.customer_id = o.customer_id
join `Target_sql.order_items` as ot
on o.order_id = ot.order_id
where order_status = 'delivered'
group by c.customer_state
order by est_delivery desc
limit 5)

select
   customer_state,
   est_delivery
from cte
union all
select
   customer_state,
   est_delivery
from cte1
ORDER BY est_delivery asc
```

Row	customer_state	est_delivery
1	AC	-20.01
2	RO	-19.08
3	AM	-18.98
4	AP	-17.44
5	RR	-17.43
6	BA	-10.12
7	ES	-9.77
8	SE	-9.17
9	MA	-9.11
10	AL	-7.98

Insights: According to the analysis, I have pinpointed the top 5 states where order deliveries consistently exceed customer expectations by arriving significantly earlier than the estimated delivery dates. These regions are super efficient in deliveries and offering valuable insights into operational practices that contribute to exceeding customer expectations. These practices can serve

as benchmarks for enhancing delivery efficiency across our entire logistics network.

VI. Analysis based on the payments:

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

```
select
    distinct p.payment_type,
    count(o.order_id) as no_of_orders,
    extract(month from o.order_purchase_timestamp) as month,
   extract(year from o.order_purchase_timestamp) as year
from`Target_sql.payments` as p
join `Target_sql.orders` as o
on p.order_id = o.order_id
group by month, p.payment_type
order by month, no_of_orders desc
```

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

Row	payment_type	no_of_orders	month	year
1	credit_card	583	1	2017
2	UPI	197	1	2017
3	voucher	61	1	2017
4	debit_card	9	1	2017
5	credit_card	5520	1	2018
6	UPI	1518	1	2018
7	voucher	416	1	2018
8	debit_card	109	1	2018
9	credit_card	1356	2	2017
10	UPI	398	2	2017

Insights: The insight reveals that in each month, there is a higher number of orders placed using the credit card payment method. This information can help business understand customer payment preferences and make decisions regarding payment processing.

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

```
select
  payment_installments,
  count(order_id) as no_of_orders
from `Target_sql.payments`
```

```
where payment_installments >= 1

group by payment_installments
```

JOB INFORMATION | RESULTS | JSON

Row	payment_installment	no_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328

Insights: The insight indicates that a higher number of orders have paid on the first, second and third installments. This data can be helpful for tracking customer payment behaviors and updating payment plans or reminders to encourage timely payments.

I've also retrieved the orders that remain unpaid:

```
select

  payment_installments,

  count(order_id) as no_of_orders

from `Target_sql.payments`

where payment_installments = 0

group by payment_installments
```

JOB INFORMATION | RESULTS | JSON

Row	payment_installment	no_of_orders
1	0	2

Recommendation:

State SP is excelling in various aspects. However, it's essential to direct special attention to States AC, AP, and RR which exhibit lower performance levels. These can be achieved through the following strategies:

1. Pricing Strategy:

Adjust pricing models to maximize profitability when sales are declining.

2. Customer Segmentation and Targeting:

Customize marketing campaigns and product offerings to align with customer preferences in regions with lower sales.

3. Boosting Sales:

Examine what customers have purchased previously and how they shop. Find chances to offer them related or higher-value items to make each purchase more valuable.

4. Stock Management:

Ensure timely restocking of fast-moving items to avoid stockouts.

5. Market Expansion:

Develop strategies for entering new markets for potential collaboration or expanding product lines.

6. Customer Retention and Loyalty:

- Use customer data to predict customer churn and identify at-risk customers.
- Implement retention initiatives such as personalized offers and exceptional customer service.

7. E-commerce and Online Presence:

Leverage digital marketing strategies to reach a wider audience and drive online sales in regions with lower sales.

8. Tracking Performance:

- Set up dashboards using SQL for constantly watching how well things are going.
- Keep checking important performance signs and change plans as needed.