- Import the dataset and do the usual exploratory analysis steps like checking the structure & characteristics of the dataset:

INSIGHT: Data type of all columns in the "customers" table. All the columns are having string datatype while customer_zip_code is having integer data type.

Query:

```
select
column_name, data_type
FROM
Target_11102023.INFORMATION_SCHEMA.COLUMNS
where table_name = "Customers"
```

Output:

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

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

- Get the time range between which the orders were placed

INSIGHT: The time range between which the orders were placed can be obtained by taking minimum and maximum number of orders. The time range of orders placed are from Sep 2016 to Oct 2018.

Query

```
SELECT
MIN(order_purchase_timestamp) as first_order_date,
MAX(order_purchase_timestamp) as last_order_date
FROM scaler-dsml-sql-402216.Target_11102023.Orders
```

Output:

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

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

- Count the Cities and states of customers who ordered during the given period.

INSIGHT: As per my analysis, there are a total of 4119 cities and 27 states where customers are placing their orders. There is an increasing trend in the customer base over the Years. 2016(Total cities = 179, state count = 21), 2017(Total cities = 3290, state count = 27), 2018(Total cities = 3227, state count = 27)

Query:

```
SELECT
COUNT(distinct c.customer_city) as Total_cities,
COUNT(distinct c.customer_state) as Total_states
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c
JOIN scaler-dsml-sql-402216.Target_11102023.Orders o
USING(customer_id)
```

Output:

Query results | SAVE RESULTS | EXPLORE DATA

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

Row	Total_cities	Total_states
1	4119	27

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

INSIGHT: The data indicates a 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. Businesses 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 month, suggesting potential seasonality or specific factors influencing customer behavior during those periods. Businesses should consider this when planning further investigation and targeted strategies.

Query:

```
WITH A AS
```

```
(
SELECT order_purchase_timestamp, EXTRACT(YEAR FROM  o.order_purchase_timestamp ) AS YR,
EXTRACT(Month FROM o.order_purchase_timestamp)
AS MTH
FROM
scaler-dsml-sql-402216.Target_11102023.Orders o
ORDER BY order_purchase_timestamp
),
B AS (
SELECT
YR, MTH, COUNT(order_purchase_timestamp)AS order_count
FROM A
GROUP BY YR, MTH
ORDER BY YR, MTH)
SELECT CONCAT(YR,'-',MTH) AS date, order_count
from B
```

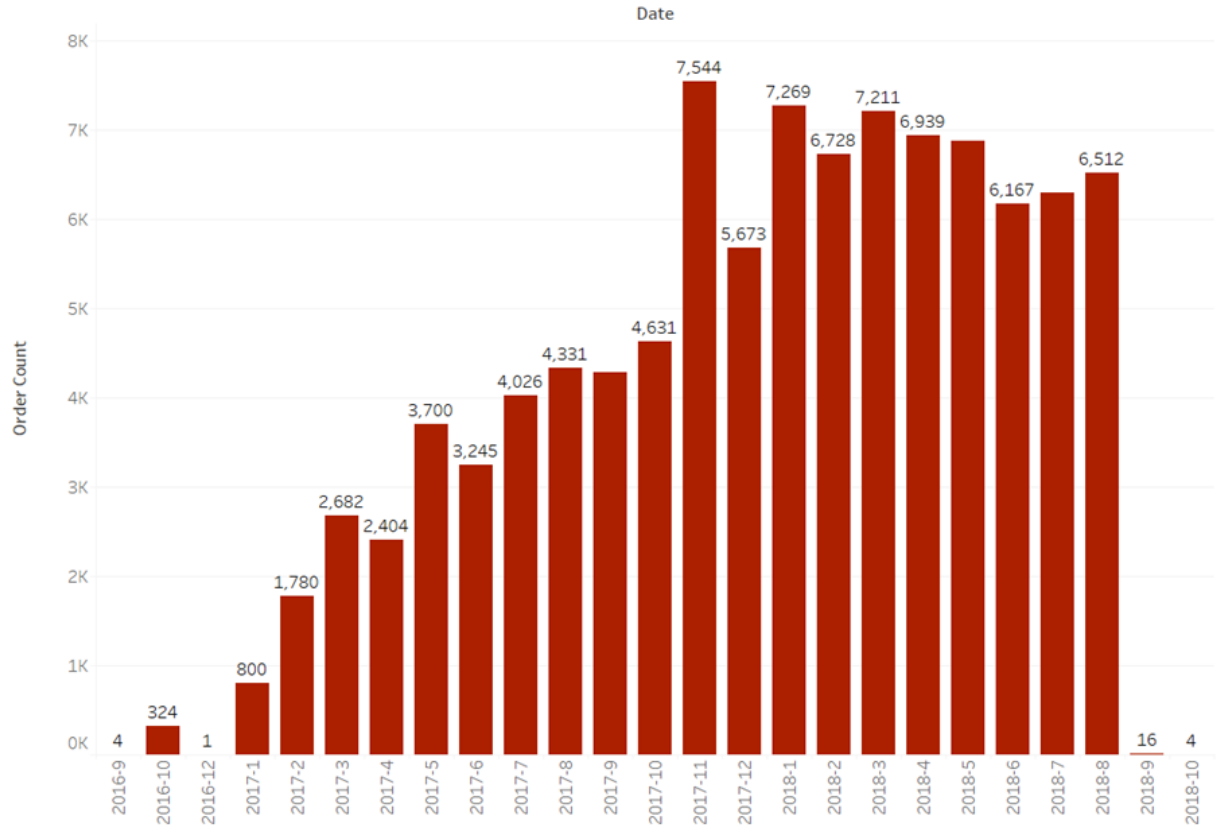
Output:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	date	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
7	2017-4	2404
8	2017-5	3700
9	2017-6	3245
10	2017-7	4026

Chart:



INSIGHT: We can see a growing trend in no. of orders placed year over year. The graph of the orders placed increased drastically from 2016 to 2017 and gradually during the year 2018.

RECOMMENDATION:

- **Every year, since the orders placed are higher, complimentary offers can be provided by clubbing the higher selling product with the lower selling product.**
- **The procurement of highest selling product can be increased.**
- **Clearance sale offers can also be implemented**

Query:

```
with Yearly_order_count as
(Select
Extract(year From order_purchase_timestamp) as Yr,
Count(order_id) as order_count
From
Target_11102023.Orders
Group by Yr
```

```
Order by Yr),
YOY as
(Select
Yr, order_count, lead(order_count) over (order by Yr) as nxt_yr_count
from Yearly_order_count
order by Yr)
Select Yr, order_count, nxt_yr_count, round(((nxt_yr_count - order_count)/order_count)*100,2) as
YOY_increase
from YOY
```

Output:

Row	Yr	order_count	nxt_yr_count	YOY_increase
1	2016	329	45101	13608.51
2	2017	45101	54011	19.76
3	2018	54011	*null*	*null*

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

INSIGHT:

We can see that there is a steep increase in the number of sales orders in November due to Black Friday week. We can also infer that after the month of November 2017, there have been ups and downs in the number of sales but they are around 6500 till September where orders decline.

RECOMMENDATION: The company can increase the production in December as most of the customers placed their orders during Black Friday week. Company can provide some lucrative offers in month of December which would lead to an increased sales and profit.

Query:

```
WITH A AS
(
SELECT order_purchase_timestamp, EXTRACT(YEAR FROM  o.order_purchase_timestamp ) AS YR,
EXTRACT(Month FROM o.order_purchase_timestamp)
AS MTH
FROM
scaler-dsml-sql-402216.Target_11102023.Orders o
ORDER BY order_purchase_timestamp
),
B AS (
```

```
SELECT
YR, MTH, COUNT(order_purchase_timestamp)AS order_count
FROM A
GROUP BY YR, MTH
ORDER BY YR, MTH),
C as
(SELECT CONCAT(YR,'-',MTH) AS dates, order_count
from B)
select *, lead(order_count) over (order by dates) as nxt_month_oc, round(((lead(order_count) over (order by dates) -
order_count)/order_count)*100,2) as percent_change
 from C
 order by dates
```

Row	dates	order_count	nxt_month_oc	percent_change
1	2016-10	324	1	-99.69
2	2016-12	1	4	300.0
3	2016-9	4	800	19900.0
4	2017-1	800	4631	478.88
5	2017-10	4631	7544	62.9
6	2017-11	7544	5673	-24.8
7	2017-12	5673	1780	-68.62
8	2017-2	1780	2682	50.67
9	2017-3	2682	2404	-10.37
10	2017-4	2404	3700	53.91
11	2017-5	3700	3245	-12.3
12	2017-6	3245	4026	24.07

I have also calculated the orders placed over the quarters of the year so that we can see the increasing number of orders in each quarter.

Query:

```
With A as
(select
order_id,
extract(Year from order_purchase_timestamp) as Yr,
extract(quarter from order_purchase_timestamp) as quarter,
FROM
scaler-dsml-sql-402216.Target_11102023.Orders)
Select
quarter, Yr, count(order_id) as order_count
From
A
group by quarter, Yr
```

order by Yr, quarter

Output:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JS

Row	quarter	Yr	order_count
1	3	2016	4
2	4	2016	325
3	1	2017	5262
4	2	2017	9349
5	3	2017	12642
6	4	2017	17848
7	1	2018	21208
8	2	2018	19979
9	3	2018	12820
10	4	2018	4

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

INSIGHT: On checking the shopping patterns of the customers in Brazil, I see that most of the customers shopped in Afternoon (38135) followed by Night (28331) and Morning (27733) and the least number of orders were placed at Dawn (5242).

RECOMMENDATION: Most of the customers are placing their orders in Afternoon so I would recommend company having additional vendors to accommodate/support the customers during Afternoon.

Query:

```
WITH A as
(SELECT
EXTRACT(Year FROM o.order_purchase_timestamp) AS Yrs,
```

```
CASE
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) >= 0 AND EXTRACT(HOUR FROM
o.order_purchase_timestamp) <= 6 THEN 'Dawn'
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) > 6 AND EXTRACT(HOUR FROM
o.order_purchase_timestamp) <= 12 THEN 'Morning'
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) > 12 AND EXTRACT(HOUR FROM
o.order_purchase_timestamp) <= 18 THEN 'Afternoon'
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) > 18 AND EXTRACT(HOUR FROM
o.order_purchase_timestamp) < 24 THEN 'Night'
END as totd
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN scaler-dsml-sql-402216.Target_11102023.Orders o
USING(customer_id)
ORDER BY o.order_purchase_timestamp)
SELECT totd, COUNT(totd) as purchase_count
FROM A
GROUP BY  totd
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	totd	purchase_count
1	Night	28331
2	Afternoon	38135
3	Morning	27733
4	Dawn	5242

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

INSIGHT- We can see the order count for every month for each state so that the company can understand the pattern to serve customers better by altering strategies at a regional level.

Query:

```
SELECT C.customer_state AS state, EXTRACT(MONTH FROM O.order_purchase_timestamp) AS MTH,
EXTRACT(YEAR FROM O.order_purchase_timestamp) AS YR, COUNT(distinct o.order_id) AS order_count
FROM
scaler-dsml-sql-402216.Target_11102023.Customers C JOIN scaler-dsml-sql-402216.Target_11102023.Orders O
USING(customer_id)
GROUP BY MTH,YR, state
ORDER BY YR, MTH
```

Output:

Query results

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

Row	state	MTH	YR	order_count
1	RR	9	2016	1
2	RS	9	2016	1
3	SP	9	2016	2
4	SP	10	2016	113
5	RS	10	2016	24
6	RJ	10	2016	56
7	MT	10	2016	3
8	GO	10	2016	9
9	MG	10	2016	40
10	CE	10	2016	8

- How are the customers distributed across all the states?

INSIGHT: I inferred that most of the customers are located in SP (41746) and RR had the least number of customers (46). The company can check if the demand and supply needs are met based on the customer distribution across all the states.

RECOMMENDATION: The company can verify the pattern of the low order count in particular states and work on developing strategies to increase business in those states as well.

Query:

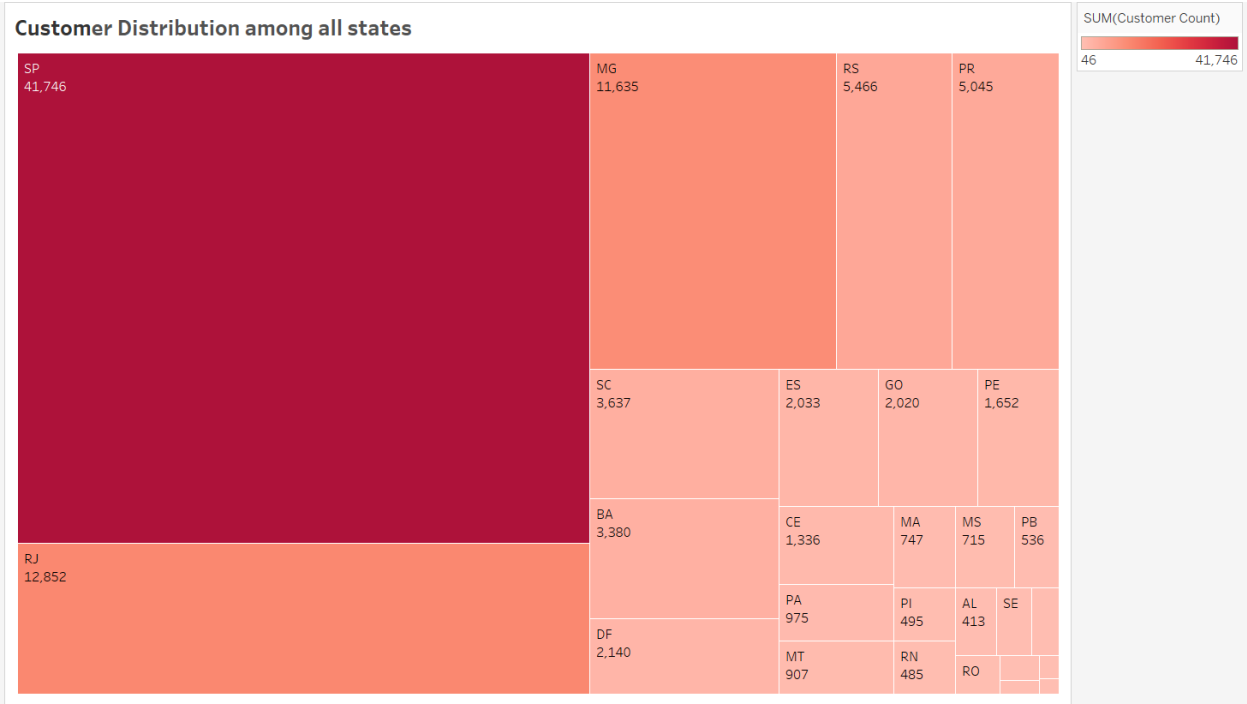
```
SELECT
customer_state, COUNT(customer_id) AS customer_count
FROM Target_11102023.Customers
GROUP BY customer_state
ORDER BY customer_count DESC
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	customer_state	customer_count
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



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

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

INSIGHT: I see a drastic increase in sales in 2018 than in 2017 from January to April, which stabilized after April. There is an average of 221% increase in revenue YoY.

Query:

```
with t1_2017 as
(SELECT
EXTRACT(MONTH from o.order_purchase_timestamp) AS mth, SUM(p.payment_value) as total_revenue_2017
FROM
scaler-dsml-sql-402216.Target_11102023.Orders o JOIN
scaler-dsml-sql-402216.Target_11102023.Payments p
USING(order_id)
where (EXTRACT(YEAR from o.order_purchase_timestamp) = 2017) and (EXTRACT(MONTH from
o.order_purchase_timestamp) between 1 and 8)
GROUP BY mth
ORDER BY mth),
t2_2018 as
(
  SELECT
EXTRACT(MONTH from o.order_purchase_timestamp) AS mth, SUM(p.payment_value) as total_revenue_2018
FROM
scaler-dsml-sql-402216.Target_11102023.Orders o JOIN
scaler-dsml-sql-402216.Target_11102023.Payments p
USING(order_id)
where (EXTRACT(YEAR from o.order_purchase_timestamp) = 2018) and (EXTRACT(MONTH from
o.order_purchase_timestamp) between 1 and 8)
GROUP BY mth
ORDER BY mth
)
select
*, round(((total_revenue_2018 - total_revenue_2017)/total_revenue_2017)*100,2) as percentage_change
FROM
t1_2017 join t2_2018
using(mth)
order by mth
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTI

Row	mth	total_revenue_2017	total_revenue_2018	percentage_change
1	1	138488.0399999...	1115004.180000...	705.13
2	2	291908.0099999...	992463.3400000...	239.99
3	3	449863.6000000...	1159652.119999...	157.78
4	4	417788.0300000...	1160785.480000...	177.84
5	5	592918.8200000...	1153982.150000...	94.63
6	6	511276.3800000...	1023880.499999...	100.26
7	7	592382.9200000...	1066540.750000...	80.04
8	8	674396.3200000...	1022425.319999...	51.61

- Calculate the Total and average value of the order price for each state

INSIGHT: I calculated the average value of the order price and the total value of the orders placed in each state. I observed that the SP is the highest revenue-generating state (total order value: 5998226.96 and Average order value of 137.5) and RR generated the least revenue (Total order value 10064.62 with an average of 218.8). Another insight would be that some high-value orders are being placed by certain customers which contribute to the overall value.

RECOMMENDATION: I would recommend the company to provide location-based offers such as fast delivery or in-store discounts.

Query:

```
SELECT c.customer_state,ROUND(SUM(p.payment_value),2) as total_revenue,
ROUND(AVG(p.payment_value),2) as average_order_value
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id)
JOIN
scaler-dsml-sql-402216.Target_11102023.Payments p
USING(order_id)
GROUP BY c.customer_state
order by total_revenue desc
```

Output:

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

Row	customer_state	total_revenue	average_order_value
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71

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

INSIGHT: I calculated the total and average value of order freight for each state. SP has the highest total freight value 753351.18 with an average of 15.2. RR has the lowest total freight value (total_fv) of 2235.19 with an average (avg_fv) of 42.98.

Query:

```
SELECT c.customer_state, ROUND(SUM(oi.freight_value),2) AS total_fv, ROUND(AVG(freight_value),2) AS
avg_fv
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id)
JOIN
scaler-dsml-sql-402216.Target_11102023.Payments p
USING(order_id)
JOIN scaler-dsml-sql-402216.Target_11102023.Order_items oi
USING(order_id)
group by c.customer_state
order by total_fv desc
```

Output:

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	customer_state	total_fv	avg_fv
1	SP	753351.18	15.2
2	RJ	323413.95	21.1
3	MG	281301.31	20.63
4	RS	141579.69	21.83
5	PR	122669.69	20.58
6	BA	106538.62	26.32
7	SC	92216.36	21.44
8	PE	61923.56	32.78
9	GO	55237.53	22.73
10	DF	52118.84	21.08

Analysis based on sales, freight, and delivery time.

- Find the no. of days taken to deliver each order from the order's purchase date as delivery time.

INSIGHT: The 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 difference between the estimated delivery date and the actual delivery date of an order,

showing delays or variances in delivery time.

RECOMMENDATION:

The velocity between the estimated & actual delivery date of an order can be increased by providing express delivery option available to the customers so that the waiting period is minimal.

Query:

```
SELECT
o.order_id, timestamp_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)
AS time_to_deliver, timestamp_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date, DAY) AS diff_estimated_delivery
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id)
```

```
where o.order_delivered_customer_date is not null
order by time_to_deliver desc
```

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON | EX

Row	order_id	time_to_deliver	diff_estimated_delivery
1	ca07593549f1816d26a572e06...	209	-181
2	1b3190b2dfa9d789e1f14c05b...	208	-188
3	440d0d17af552815d15a9e41a...	195	-165
4	0f4519c5f1c541ddec9f21b3bd...	194	-161
5	285ab9426d6982034523a855f...	194	-166
6	2fb597c2f772eca01b1f5c561b...	194	-155
7	47b40429ed8cce3aee9199792...	191	-175
8	2fe324febf907e3ea3f2aa9650...	189	-167
9	2d7561026d542c8dbd8f0daea...	188	-159
10	437222e3fd1b07396f1d9ba8c...	187	-144

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

INSIGHT: The first 5 states have the highest average freight value and the last 5 states have the minimum average freight value. This information can guide us to manage

costs better, find unusual costs, and figure out cost effective methods of shipping

in our logistics operations.

RECOMMENDATION:

- **The company can avoid shipping small quantities at short intervals in order to reduce freight costs.**
- **The company can plan on packaging freight more efficiently.**

Query:

```
WITH A AS
(SELECT c.customer_state, ROUND(SUM(oi.freight_value),2) AS total_fv, ROUND(AVG(freight_value),2) AS
```

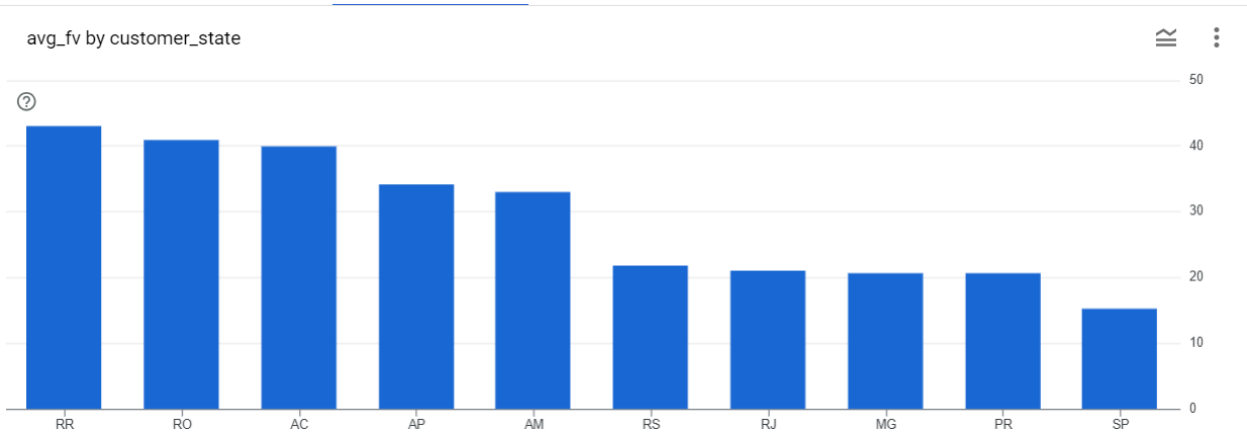
```
avg_fv
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id)
JOIN
scaler-dsml-sql-402216.Target_11102023.Payments p
USING(order_id)
JOIN scaler-dsml-sql-402216.Target_11102023.Order_items oi
USING(order_id)
group by c.customer_state
order by total_fv desc
LIMIT 5),
B AS
(
  SELECT c.customer_state, ROUND(SUM(oi.freight_value),2) AS total_fv, ROUND(AVG(freight_value),2) AS
avg_fv
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id)
JOIN
scaler-dsml-sql-402216.Target_11102023.Payments p
USING(order_id)
JOIN scaler-dsml-sql-402216.Target_11102023.Order_items oi
USING(order_id)
group by c.customer_state
order by total_fv
LIMIT 5
)
SELECT * FROM A
UNION ALL
SELECT * FROM B
ORDER BY avg_fv DESC
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	customer_state	avg_fv
1	PB	43.26
2	RR	42.98
3	RO	40.97
4	AC	40.02
5	TO	39.68
6	RJ	21.1
7	DF	21.08
8	MG	20.63
9	PR	20.58
10	SP	15.2



- Find out the top 5 states with the highest & lowest average delivery time
INSIGHT: This data offers visibility into delivery efficiency, enabling us to focus on areas where improvements can be made to enhance customer satisfaction and improve delivery operations. This information provides insights into order fulfillment efficiency, helping the business pinpoint areas for improvement in the logistics and delivery processes.

Query:

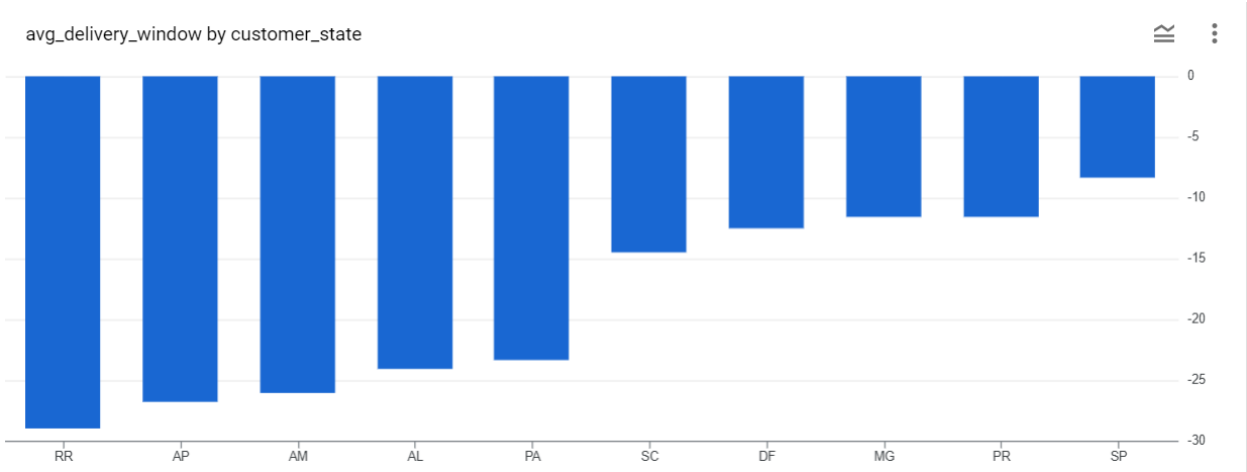
```
With A as
(SELECT c.customer_state, ROUND(avg(timestamp_diff( order_delivered_customer_date,
o.order_purchase_timestamp,  DAY)),2)
as avg_delivery_window, dense_rank() over (order by ROUND(avg(timestamp_diff(
order_delivered_customer_date,o.order_purchase_timestamp,DAY)),2)) as low_avg_del_rank
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id)
group by c.customer_state)
, B as
(
  SELECT c.customer_state, ROUND(avg(timestamp_diff(
order_delivered_customer_date,o.order_purchase_timestamp, DAY)),2)
as avg_delivery_window, dense_rank() over (order by ROUND(avg(timestamp_diff(
order_delivered_customer_date,o.order_purchase_timestamp, DAY)),2) desc) as high_avg_del_rank
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c JOIN
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id)
group by c.customer_state
)
Select customer_state,avg_delivery_window from A
where low_avg_del_rank between 1 and 5
union all
select customer_state,avg_delivery_window from B
where high_avg_del_rank between 1 and 5
order by avg_delivery_window
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	customer_state	avg_delivery_window
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98



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

INSIGHT: In the state SP, the order delivery is fast compared to the estimated date of delivery.

RECOMMENDATION:
The time taken to deliver the order can be reduced by choosing reliable vendors and rewarding incentives to the employees who does better service.

Query:

```
With A as
(Select
c.customer_state, timestamp_diff(o.order_purchase_timestamp, o.order_estimated_delivery_date, DAY) as
estimated_del_timeframe,
timestamp_diff(o.order_purchase_timestamp, o.order_delivered_carrier_date, DAY) as actual_del_timeframe
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c
join
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id))
Select
customer_state, round(avg(actual_del_timeframe) - avg(estimated_del_timeframe),2) as del_rate
FROM
A
group by customer_state
order by del_rate
limit 5
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	customer_state	del_rate
1	SP	16.13
2	DF	21.35
3	MG	21.47
4	PR	21.53
5	ES	22.38

Analysis based on the payments:

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

INSIGHT: I found the MoM data for the number of orders placed using various payment methods and turns out that the credit card is the most preferred method

of payment while the debit card was the least preferred.

RECOMMENDATION: The company can revisit the discounts/offers to encourage customers to use different types of payments as well. Reducing the efforts from the customer while completing an order will drastically improve the customer experience hence better sales.

Query:

```
With A as
(Select
c.customer_state, timestamp_diff(o.order_purchase_timestamp, o.order_estimated_delivery_date, DAY) as
estimated_del_timeframe,
timestamp_diff(o.order_purchase_timestamp, o.order_delivered_carrier_date, DAY) as actual_del_timeframe
FROM
scaler-dsml-sql-402216.Target_11102023.Customers c
join
scaler-dsml-sql-402216.Target_11102023.Orders o
using(customer_id))
Select
customer_state, round(avg(actual_del_timeframe) - avg(estimated_del_timeframe),2) as del_rate
FROM
A
group by customer_state
order by del_rate
limit 5
```

Output:

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

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

- Find the no. of orders placed based on the payment installments that have been paid.

INSIGHT: As per my analysis, most of the orders are being placed where customers prefer to pay the price within three installments.
RECOMMENDATION: The company can provide more offers to make payment for expensive products in EMI to increase their popularity.
Query:

```
With A as
(Select *,
extract(month from o.order_purchase_timestamp) as mth, extract(Year from
o.order_purchase_timestamp) as yr
from
Target_11102023.Orders o
join
scaler-dsml-sql-402216.Target_11102023.Payments p
using(order_id)
order by mth, yr)
SELECT mth, yr, payment_installments, count(order_id) as total_orders
FROM A
group by mth, yr, payment_installments
order by yr, mth
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUT

Row	mth	yr	payment_installment	total_orders
1	9	2016	1	1
2	9	2016	3	1
3	9	2016	2	1
4	10	2016	1	144
5	10	2016	10	42
6	10	2016	2	30
7	10	2016	3	43
8	10	2016	4	26
9	10	2016	5	20
10	10	2016	6	18

Other recommendations for the company:

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 the customers purchase history and offer them related items to increase the sales.

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 to monitor the progress in the company’s performance.