TARGET BUSINESS CASE STUDY

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

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

Approach:

i) select column_name and data_type from the information_schema.
Syntax-
(databaseName.datasetName.Information_Schema.Columns)
ii) For table customers

Query:

```
SELECT
    COLUMN_NAME,
    DATA_TYPE
FROM
`scaler-dsml-sql-444401.Target_SQL_Buisness_Case.INFORMATION_SCHEMA.COLUMNS`
WHERE TABLE_NAME = 'customers';
```

Output:

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

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

Approach:

i) We define a CTE to calculate the date and time for first and last order placed.
ii) combine date and time for first and last order together using CONCAT to calculate the time range.

Query:

```
WITH first_and_last_order AS
(
SELECT
MIN(order_purchase_timestamp) AS first_order,
MAX(order_purchase_timestamp) AS recent_order
from `Target_SQL_Buisness_Case.orders`
)
SELECT
CONCAT(first_order,' - ',recent_order) AS `Time Range between first and last
    order`
FROM first_and_last_order;
```

Output:

Row	Time Range between first and last order
1	2016-09-04 21:15:19+00 - 2018-10-17 17:30:18+00

Insights:

Customer Engagement: Understanding the earliest and latest orders helps gauge customer activity trends over the given period.

Operational Impact: The start and end dates of orders can indicate how well the business handled demand and fulfillment over time.

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

Assumption:

We assume that the time period for which we have to count is between 2016 and 2018.

Approach:

i) First join the customer table with order table with customer id. As information about cities and states is located in the customer table and information about orders is present in the orders table.
ii) Count the number of cities and number of states of customers who ordered between 2016 and 2018.

Query:

```
SELECT
COUNT(DISTINCT c.customer_city) as city_count,
COUNT(DISTINCT c.customer_state) AS state_count
FROM `Target_SQL_Buisness_Case.customers` c
INNER JOIN `Target_SQL_Buisness_Case.orders` o
ON c.customer_id = o.customer_id
WHERE EXTRACT(YEAR FROM o.order_purchase_timestamp) BETWEEN 2016 AND 2018;
```

Output:

Row	city_count	state_count
1	4119	27

Insights:

Geographic Distribution: Counting the cities and states reveals the geographic spread of customers during the 2017-2018 period.

Market Concentration: Some cities and states may show higher order volumes, indicating a concentration of demand in specific regions.

Actions:

Target High-Volume Areas: Focus marketing and product availability in cities and states with higher order counts to maximize reach.

Expand in Low-Volume Areas: Develop strategies (e.g., promotions, delivery incentives) to attract customers from cities and states with fewer orders.

Optimize Logistics: Improve delivery infrastructure in high-order cities/states to reduce fulfillment times and costs.

Monitor Market Trends: Track changes in geographic distribution over time to identify emerging markets or areas needing attention.

2. In-depth Exploration:

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

Approach:

i) Define a CTE to count the number of orders yearly.
ii) Define another CTE (percentage) to calculate the previous year orders and percentage of difference of orders placed this year over previous years.
iii) By comparing the orders of this year and previous year we calculate the trend whether it is increasing or decreasing using '*case when*'
iv) order the result set by order_year in ascending order.

Query:

```
WITH CTE AS
(
  SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp ) AS order_year,
  COUNT(order_id) AS no_of_orders
  from `Target_SQL_Buisness_Case.orders`
  GROUP BY order_year
),
percentage AS
(
select
order_year,
no_of_orders,
LAG(no_of_orders) OVER(ORDER BY order_year) AS prev_year_orders,
CASE
WHEN
LAG(no_of_orders) OVER(ORDER BY order_year) IS NULL
THEN NULL
ELSE
ROUND((no_of_orders - LAG(no_of_orders) OVER(ORDER BY
    order_year))/LAG(no_of_orders) OVER(ORDER BY order_year)*100,2)
END AS year_by_year_growth_percentage
from CTE
```

```
ORDER BY order_year
)
SELECT *,
CASE
WHEN (no_of_orders - prev_year_orders) > 0 THEN 'increasing'
WHEN (no_of_orders - prev_year_orders) < 0 THEN 'decreasing'
WHEN prev_year_orders IS NULL THEN 'N/A'
ELSE 'no change'
END AS trend
FROM percentage
order by order_year;
```

Output:

Row	order_year	no_of_orders	prev_year_orders	year_by_year_growth	trend
1	2016	329	*null*	*null*	N/A
2	2017	45101	329	13608.51	increasing
3	2018	54011	45101	19.76	increasing

Insights:

Growth Indication: A growing trend in the number of orders reflects increasing customer demand, brand recognition, and market reach.

Business Expansion: Continuous growth in order volume may indicate successful marketing strategies, product launches, or a broader customer base.

Actions:

Enhance Marketing: Continue investing in successful marketing strategies and explore new channels to maintain growth momentum.

Customer Retention: Implement loyalty programs or offers to retain existing customers and further increase order volume.

Forecast Demand: Use trends to predict future demand and adjust inventory and staffing levels proactively.

Monitor Growth Consistency: Ensure that growth is sustained across regions and periods, and address any potential slowdowns promptly.

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

Approach:

i) First Extract the month part from order_purchase_timestamp) and alias it as order_month.

```
FORMAT_DATE('%B', DATE(order_purchase_timestamp)) AS order_month,
```

ii) Then count the total number of orders (order_id) in each month using and alias it as total_orders.

iii) Group the result by the order_month.

iv) orders the result by order_month in ascending order.

Query:

```
SELECT
FORMAT_DATE('%B', DATE(order_purchase_timestamp)) AS order_month,
COUNT(order_id) AS total_orders
FROM
`Target_SQL_Buisness_Case.orders`
GROUP BY
order_month
ORDER BY
total_orders desc;
```

Output:

Row	order_month	total_orders
1	August	10843
2	May	10573
3	July	10318
4	March	9893
5	June	9412
6	April	9343
7	February	8508
8	January	8069
9	November	7544
10	December	5674
11	October	4959
12	September	4305

Insights:

Seasonal Demand: Monthly order patterns help identify periods of high demand (e.g., holidays, festivals) and slower months.

Customer Behavior: Fluctuations in order volume across months highlight customer preferences and potential factors like weather, events, or economic cycles.

Revenue Forecasting: Monthly seasonality insights can aid in forecasting revenue, inventory needs, and logistics planning.

Actions:

Plan Promotions: Run targeted marketing campaigns during peak months to capitalize on increased demand.

Prepare for Low-Season: Use insights to plan for slow months by adjusting inventory, pricing, or offering discounts to boost sales.

Optimize Inventory: Align stock levels with seasonal demand to prevent stockouts during high-demand months.

Enhance Logistics: Adjust logistics and fulfillment strategies based on expected order volume during peak months.

Track Trends: Continuously monitor and adjust for any shifts in seasonal patterns to stay responsive to customer behavior.

3. 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

Approach:

i) Firstly we define a CTE as order_time to calculate the time frame at which Brazilian customers mostly place their orders using a *'case when'* statement using the order table.
ii) Then count the number of orders placed for each time frame.
iii) Order the result set in descending order to get the highest orders on top.
iv) show only the first row using *limit* clause to mention the time frame in which maximum number of orders are placed.

Query:

```
With order_time AS
(
  SELECT
  CASE
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >=0 AND EXTRACT(HOUR FROM
    order_purchase_timestamp)<=6 THEN 'Dawn'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 7 AND EXTRACT(HOUR FROM
    order_purchase_timestamp) <=12 THEN 'Mornings'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 13 AND EXTRACT(HOUR FROM
    order_purchase_timestamp) <= 18 THEN 'Afternoon'
  ELSE 'NIGHT'
  END AS Order_Time
  FROM `Target_SQL_Buisness_Case.orders`
)
SELECT
Order_Time,
COUNT(*) AS no_of_orders,
FROM order_time
GROUP BY Order_Time
```

```
ORDER BY no_of_orders  desc
LIMIT 1;
```

Output:

Row	Order_Time.Order_Time	no_of_orders
1	Afternoon	38135

Insights:

Peak Ordering Times: Identifying the most common time of day for placing orders helps understand customer behavior and peak engagement periods.

Customer Availability: The peak ordering times reveal when customers are most active, either due to work schedules, free time, or convenience.

Order Fulfillment Timing: Knowing peak times allows better scheduling of warehouse operations and delivery logistics.

Actions:

Targeted Marketing: Run time-based promotions during peak ordering hours to maximize conversions.

Optimize Support Hours: Ensure customer service and support are available during peak times for better engagement.

Efficient Order Processing: Align fulfillment and dispatch teams with peak order times to speed up processing.

Dynamic Pricing: Consider adjusting shipping or pricing strategies based on time of day to encourage orders during off-peak times.

Engage Customers: Use insights to send reminders or special offers at peak times when customers are most likely to engage.

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

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

Approach:

i) First join the table *customers* with *orders* as the customers table contains the information about state and orders table contains information about orders.
ii) Extract month from the order_purchase_timestamp to look into the order month.
iii)count the total orders placed for each month in each state..
iv) order the result set by state and month in ascending order.

Query:

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

Output:

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

Insights:

Seasonality Trends: Identifying month-on-month order patterns reveals demand fluctuations and seasonal trends in each state.

State-Specific Demand: Certain states may show higher order volumes in specific months, indicating local events or seasonality.

Growth Opportunities: States with consistently low order volumes across months may require targeted strategies to boost demand.

Actions:

Seasonal Promotions: Run state-specific promotions or discounts during peak demand months to maximize revenue.

Boost Off-Season Sales: Implement marketing campaigns to drive sales in low-demand months for each state.

Inventory Planning: Align inventory and logistics planning with demand patterns to avoid stockouts or excess inventory.

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

Approach:

i) Choose a '*customers*' table.
ii) Count the customer_unique_id for each state and alias it as total_customers.
iii) Order the result set by total_customers in descending order to get the highest number of customers on top.

Query:

```
SELECT
customer_state,
COUNT(customer_unique_id) AS total_customers
FROM
`Target_SQL_Buisness_Case.customers`
GROUP BY
customer_state
ORDER BY
total_customers DESC;
```

Output:

Row	customer_state	total_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:

Customer Concentration: States with the highest number of customers are key markets and represent the largest revenue potential.

Untapped Markets: States with lower customer distribution may need targeted efforts to improve market penetration.

Regional Trends: Analyzing customer distribution helps identify geographic patterns and potential for regional expansion.

Actions:

Focus on High-Density States: Invest in marketing, logistics, and customer engagement in states with the largest customer base.

Expand in Low-Density States: Launch campaigns or promotions in states with fewer customers to grow presence.

Optimize Resources: Allocate resources (warehouses, delivery routes) based on customer density to improve efficiency.

Marketing Strategies: Design state-specific marketing strategies based on customer distribution and preferences.

Monitor Growth: Regularly track changes in distribution to identify emerging markets or declining regions.

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

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.

Approach:

i) Define a CTE to calculate the total cost for each year (2017 to 2018) including months between Jan to Aug only.
ii) Select the previous year cost to compare it with this year's cost.
iii) Use case when statement to calculate the percentage increase in total cost from previous year.
If previous year's cost is null then set percentage increase in cost to null else calculate it with the formula:
100 * (present_year_cost - previous_year_cost)/previous_year_cost
iv) order the result set by year in ascending order.

Query:

```
WITH total_payment AS
(
  SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
  ROUND(SUM(p.payment_value),2) AS total_cost
  FROM `Target_SQL_Buisness_Case.orders` o
  JOIN `Target_SQL_Buisness_Case.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 year
  order by year
)
SELECT *,
LAG(total_cost)OVER(ORDER BY year) AS prev_year_cost,
CASE
WHEN LAG(total_cost)OVER(ORDER BY year) IS NULL THEN NULL
ELSE ROUND((total_cost - LAG(total_cost)OVER(ORDER BY
    year))/LAG(total_cost)OVER(ORDER BY year)*100,2)
```

```
END AS percentage_increase_in_cost
FROM total_payment
ORDER BY year;
```

Output:

Row	year	total_cost	prev_year_cost	percentage_increase
1	2017	3669022.119999...	*null*	*null*
2	2018	8694733.839999...	3669022.119999...	136.98

Insights:

Year-on-Year Growth: The percentage increase in order costs between 2017 and 2018 (Jan to Aug) reflects business growth and customer demand during this period.

Revenue Trends: A significant increase may indicate strong performance.

Seasonality Check: Comparing Jan–Aug ensures insights are aligned with seasonal patterns rather than full-year trends.

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

Approach:

i) Join the tables customers, orders and payments.

ii) Calculate the total and average value of order price for each state.

iii) Order the result set by state in ascending order.

Query:

```
SELECT c.customer_state,
ROUND(SUM(oi.price),2) AS total_price,
ROUND(SUM(oi.price)/COUNT(DISTINCT o.order_id),2) AS avg_price
FROM `Target_SQL_Buisness_Case.customers` c
JOIN `Target_SQL_Buisness_Case.orders` o
ON c.customer_id = o.customer_id
JOIN `Target_SQL_Buisness_Case.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
order by total_price desc,avg_price desc;
```

Output:

Row	customer_state	total_price	avg_price
1	SP	5202955.05	125.75
2	RJ	1824092.67	142.93
3	MG	1585308.03	137.33
4	RS	750304.02	138.13
5	PR	683083.76	136.67
6	SC	520553.34	144.12
7	BA	511349.99	152.28
8	DF	302603.94	142.4
9	GO	294591.95	146.78
10	ES	275037.31	135.82

Insights:

Revenue Contribution: The total order price per state highlights which regions generate the most revenue.

Customer Spending: The average order price per state indicates customer purchasing behavior and affordability trends in each region.

High-Value Markets: States with high total and average order values represent key markets with greater revenue potential.

Low-Value States: Regions with low average or total order values may need targeted marketing or product adjustments to boost sales.

Actions:

Targeted Marketing Campaigns: Develop marketing campaigns focused on states with higher average order values to maximize return on investment.

Inventory Management: Ensure availability of high-demand products in regions identified as key markets.

Warehouse setup and dispatch team: Set up more warehouses with more dispatch team in states where the number of orders are more for timely deliveries

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

Approach:

i) Join the tables customers, orders and order_items on customer_id and order_id respectively.

ii) Calculate the total and average freight value for each state.

iii) Order the result set by state in ascending order.

Query:

```
SELECT c.customer_state,
ROUND(SUM(oi.freight_value),2) AS total_freight,
ROUND(SUM(oi.freight_value)/COUNT(DISTINCT o.order_id),2) AS avg_freight
FROM `Target_SQL_Buisness_Case.customers` c
JOIN `Target_SQL_Buisness_Case.orders` o
ON c.customer_id = o.customer_id
JOIN `Target_SQL_Buisness_Case.order_items` oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state,
order by c.customer_state;
```

Output:

Row	customer_state	total_freight	avg_freight
1	AC	3686.75	45.52
2	AL	15914.59	38.72
3	AM	5478.89	37.27
4	AP	2788.5	41.01
5	BA	100156.68	29.83
6	CE	48351.59	36.44
7	DF	50625.5	23.82
8	ES	49764.6	24.58
9	GO	53114.98	26.46
10	MA	31523.77	42.6

Insights:

Cost Analysis: Understanding the total and average freight costs by state can help identify regions where shipping expenses are higher.

State-Level Efficiency: High average freight cost may indicate logistical challenges like remote locations or poor infrastructure.

Regional Profitability: States with higher total freight costs may reduce overall profitability.

Actions:

Pricing Strategy: By analyzing freight costs, companies can adjust their pricing strategies particularly in states with higher average freight costs.

Market Expansion: States showing lower freight costs could be potential targets for market expansion or increased marketing efforts, as lower shipping expenses can enhance profitability in those regions.

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

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.

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

i. time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
ii. diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

Approach:

i)From table orders, select order_id and calculate the difference between order_delivered_customer_date and order_purchase_timestamp to calculate the number of days taken to deliver each order.
ii)Calculate the difference between order_delivered_customer_date and order_estimated_delivery_date to calculate the difference between estimated and actual delivery date.

iii)Filter the result for order_delivered_customer_date is not null for already delivered order.

Query:

```
SELECT
order_id,
TIMESTAMP_DIFF(order_delivered_customer_date,  order_purchase_timestamp,  day)
AS delivery_time,
TIMESTAMP_DIFF(order_delivered_customer_date,   order_estimated_delivery_date,
day) AS diff_estimated_delivery
FROM `Target_SQL_Buisness_Case.orders`
WHERE order_delivered_customer_date IS NOT NULL;
```

Output:

Row	order_id	delivery_time	diff_estimated_delive
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:

Delivery Speed: There are certain orders for which delivery speed is much slower.

Estimated Accuracy: The difference (`diff_estimated_delivery`) highlights gaps between promised and actual delivery times.

Customer Satisfaction: Faster deliveries or delays directly impact customer experience.

Actions:

Optimize Logistics: Reduce delays by improving supply chain and delivery processes.

Improve Accuracy: Refine delivery time estimates to align with actual performance.

Monitor Trends: Identify patterns in delayed or early deliveries to address systemic issues.

Communicate Updates: Notify customers of delays proactively to maintain trust.

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

Approach:

i) Create a CTE for calculating the average freight value for each state. (state_freight)
ii) In the second CTE we order the result set of the first CTE in descending order to show the state with the highest freight value on top. (highest)
Provide the row number to each row using window function.

iii) In the third CTE we order the result set of the first CTE in ascending order to show the state with lowest freight value on top. (lowest)
Provide the row number to each row using window function.
iv) Now inner join the highest and lowest CTEs on the basis of row number.
v)select the state with their average freight value from both the CTEs.

Query:

```
-- calculating average freight value for each state
WITH state_freight AS
(
SELECT
s.seller_state AS state,
ROUND(AVG(oi.freight_value),2) AS freight_value
FROM `Target_SQL_Buisness_Case.sellers` s
JOIN `Target_SQL_Buisness_Case.order_items` oi
ON s.seller_id = oi.seller_id
GROUP BY s.seller_state
),
-- states with highest freight value
Highest AS
(SELECT
state,
```

```
freight_value,
row_number()over(ORDER BY freight_value DESC) as h_row_num
FROM state_freight
ORDER BY freight_value desc
),
-- states with lowest freight value
Lowest AS
(
SELECT
state,
freight_value,
row_number()over(ORDER BY freight_value ASC) as l_row_num
FROM state_freight
ORDER BY freight_value ASC
)
-- selecting top 5 states with highest and lowest freight value
SELECT
h.state AS state_highest_freight_value,
h.freight_value AS avg_freight_vlaue,
l.state AS state_lowest_freight_value,
l.freight_value AS avg_freight
FROM
Highest h JOIN Lowest l
ON h.h_row_num = l.l_row_num
LIMIT 5;
```

Output:

Row	state_highest_freight_value	avg_freight_vlaue	state_lowest_freight_value	avg_freight
1	RO	50.91	SP	18.45
2	CE	46.38	PA	19.39
3	PB	39.19	RJ	19.47
4	PI	36.94	DF	20.57
5	AC	32.84	PR	22.72

Insights:

High Freight Value States: States with the highest average freight value indicate regions with higher transportation costs, due to distance, accessibility, or infrastructure challenges.

Low Freight Value States: States with the lowest average freight value indicate cost-efficient delivery zones, due to close reach to warehouses or better infrastructure.

Profitability Analysis: High freight costs may reduce profitability in certain regions.

Customer Experience: High freight costs may result in longer delivery times or additional charges for customers, potentially affecting customer satisfaction in those regions.

Actions:

Warehouse Placement: Locate warehouses closer to high freight value states to reduce transportation costs.

Delivery Partnerships: Partner with local logistics providers to reduce costs in challenging regions.

Pricing Strategy: Implement minimum order requirements in high-cost regions.

Improve Infrastructure: Collaborate with logistics providers to improve delivery efficiency in costly states.

Customer Communication: Transparently communicate delivery charges in high freight regions to manage customer expectations.

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

Approach:

i) Create a CTE for calculating the average delivery time for each state. (statewise_average_delivery_time)
ii) In the second CTE we order the result set of the first CTE in descending order to show the state with highest delivery time on top. (highest)
Provide the row number to each row using window function.

iii) In the third CTE we order the result set of the first CTE in ascending order to show the state with lowest delivery time on top. (lowest)
Provide the row number to each row using window function.
iv) Now inner join the highest and lowest CTEs on the basis of row number.
v)select the state with their average delivery time from both the CTEs.

Query:

```
-- calculating average delivery time for each state
WITH statewise_average_delivery_time AS
(
SELECT
```

```sql
c.customer_state AS state,
CAST(CEIL(AVG(DATE_DIFF(o.order_delivered_customer_date,
    o.order_purchase_timestamp, DAY)))AS INT) AS avg_delivery_time_in_days
FROM `Target_SQL_Buisness_Case.customers` c
JOIN `Target_SQL_Buisness_Case.orders` o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
order by avg_delivery_time_in_days
),
-- states with highest delivery time
highest AS
(
  SELECT
  state,
  avg_delivery_time_in_days,
  ROW_NUMBER()OVER(order by avg_delivery_time_in_days DESC ) row_num_h
  FROM statewise_average_delivery_time
  ORDER BY avg_delivery_time_in_days DESC
),
-- states with lowest delivery time
lowest AS
(
  SELECT
  state,
  avg_delivery_time_in_days,
  ROW_NUMBER()OVER(order by avg_delivery_time_in_days ASC) row_num_l
  FROM statewise_average_delivery_time
  order by avg_delivery_time_in_days ASC
)
-- selecting top 5 states with the highest & lowest average delivery time
SELECT
h.state AS `highest Delivery Time State`,
h.avg_delivery_time_in_days AS `Delivery Time In Days`,
l.state AS `Lowest Delivery Time State`,
l.avg_delivery_time_in_days AS `Delivery Time In Days`
FROM highest h
JOIN lowest l
ON h.row_num_h = l.row_num_l
LIMIT 5;
```

Output:

Row	highest Delivery Time State	Delivery Time In Days	Lowest Delivery Time State	Delivery Time In Days_1
1	RR	29	SP	9
2	AP	27	MG	12
3	AM	26	PR	12
4	AL	25	DF	13
5	PA	24	RS	15

Insights:

High Delivery Time States: States with highest average delivery time may face some logistic issues. It could negatively impact customer satisfaction in these states.

Low Delivery Time States: States with lowest average delivery time highlight good delivery processes.

Actions:

Optimize Routes: Improve transportation routes and delivery planning in high delivery time states.

Add Warehouses: Set up regional warehouses closer to high delivery time areas to speed up shipments.

Leverage Technology: Use predictive analytics to anticipate delays and proactively manage them.

Improve Communication: Notify customers in high delivery time states of expected delays to manage expectations.

Monitor Performance: Continuously track delivery times to ensure improvements in problematic regions.

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.

Approach:

i) We can calculate the difference between the actual delivered date and estimated delivery date.

ii) we can calculate the average of actual delivery time taken for each state.(avg_actual_delivery)

iii) Actual delivery date should not be null (means order is delivered)

iv) filter the result set where average < 0. It means the order is delivered faster than the estimated time.

v) order the result set by average of actual delivery and show only top 5 states.

Query:

```
SELECT
c.customer_state,
ROUND(AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,o.order_estimated_del
    ivery_date,day)),2) AS avg_actual_delivery
FROM `Target_SQL_Buisness_Case.orders` o
JOIN `Target_SQL_Buisness_Case.customers` c
ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date IS NOT NULL
GROUP BY c.customer_state
HAVING avg_actual_delivery < 0
ORDER BY avg_actual_delivery
limit 5;
```

Output:

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

Insights:

Efficient States: The top 5 states with faster-than-estimated deliveries indicate highly efficient logistics operations.

Customer Satisfaction: Delivering before the estimated date enhances customer satisfaction and boosts brand loyalty.

Actions:

Leverage Technology: Implementing real-time tracking systems and route optimization software can further reduce delivery times and improve customer satisfaction.

Improve Communication: Communicate customers about delays for better customer experience.

Monitor Performance: Continuously track delivery times to ensure improvements in problematic regions.

6. Analysis based on the payments:

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

Approach:

i) Join the tables orders and payments on order_id.
ii) Extract the year and month from order_purchase_timestamp.
iii) Count the number of orders for each payment type for each month of each year.
iv) Order the result set by year, month and number of orders.

Query:

```
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 no_of_orders
FROM `Target_SQL_Buisness_Case.orders` o
JOIN `Target_SQL_Buisness_Case.payments` p
ON o.order_id = p.order_id
GROUP BY year,month, payment_type
order by year , month, no_of_orders desc;
```

Output:

Row	year	month	payment_type	no_of_orders
1	2016	9	credit_card	3
2	2016	10	credit_card	253
3	2016	10	UPI	63
4	2016	10	voucher	11
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	582
8	2017	1	UPI	197
9	2017	1	voucher	33
10	2017	1	debit_card	9

Insights:

Credit Card Popularity: Credit cards are consistently the most used payment method, especially in January 2017 (583 orders).

Digital Payments Adoption: UPI shows gradual adoption, highlighting the growing trend of digital payment methods.(63 in October 2016 to 197 in January 2017

Low Usage of Vouchers and Debit Cards: Vouchers and debit cards have minimal adoption.

Monthly Seasonal Trends: Orders spike in certain months (e.g., October), indicating opportunities for targeted marketing during festive or sales periods.

Actions:

Target Seasonal Campaigns: Align promotions or discounts with payment modes popular during specific months.

Encourage Underused Payment Modes: Offer incentives for less-used payment methods (voucher, debit card) to expand their adoption.

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

Approach:

i) Count the Distinct orders for each payment installments from the payments table.
ii) Filter the result set where at least one installment is paid. It means we have to filter the results where the total value paid by the customer is greater than zero.
Where payment_value > 0
iii) Order the result set by payment installments.

Query:

```
SELECT payment_installments,
COUNT(distinct order_id) AS total_orders
FROM `Target_SQL_Buisness_Case.payments`
WHERE payment_value > 0
GROUP BY payment_installments
order by payment_installments;
```

Output:

Row	payment_installment	total_orders
1	0	2
2	1	49057
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644

Insights:

Customer Payment Behavior: Most customers (49,057 orders) prefer to pay in a single installment, indicating a preference for full payment upfront.

Declining Trend: As the number of installments increases, the total orders decrease significantly, It shows limited interest in longer installment plans.

Actions:

Simplify Installment Plans: Focus on offering short-duration installment plans (e.g., 2-4 months) as they are relatively more popular.

Marketing for High Installments: Promote longer installment options to specific customer segments.