**CASE STUDY: TARGET CORPORATION**

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

**1. Data type of all columns in t he "customers" table.**

# • Query

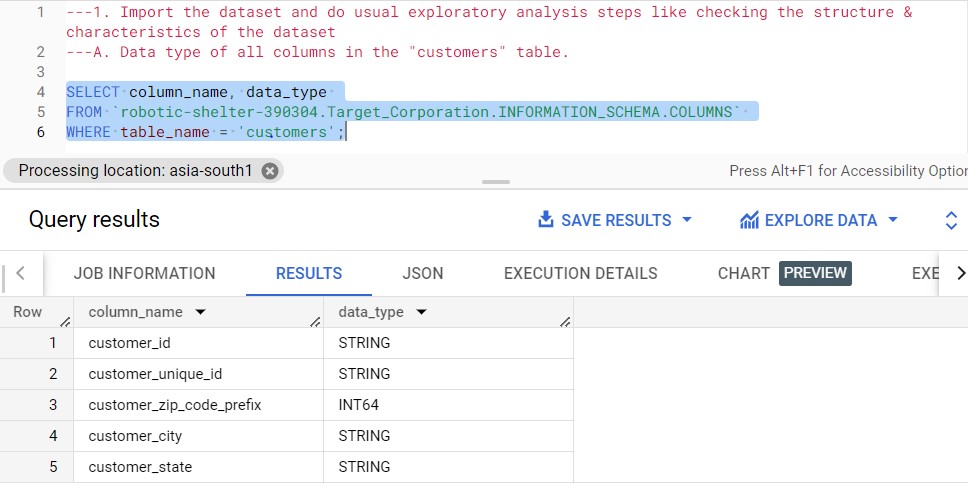
SELECT column\_name, data\_type

FROM `robotic-shelter-

390304.Target\_Corporation.INFORMATION\_SCHEMA.COLUMNS`

WHERE table\_name = 'customers';

# • Screenshot of Output



# • INSIGHTS

➢ By using this query, we can display the data type of each column present in the “customers” table.

# • Recommendations

➢ SQL data types can be broadly divided into the following categories.

1. Numeric data types such as: INT, TINYINT, BIGINT, FLOAT, REAL, etc.
2. Date and Time data types such as: DATE, TIME, DATETIME, etc.
3. Character and String data types such as: CHAR, VARCHAR, TEXT, etc.
4. Unicode character string data types such as: NCHAR, NVARCHAR, NTEXT, etc.
5. Binary data types such as: BINARY, VARBINARY, etc.
6. Miscellaneous data types - CLOB, BLOB, XML, CURSOR, TABLE, etc.

# • Assumptions

➢ By changing the table name in this query, we can display the data type of each column present in the other table also.

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

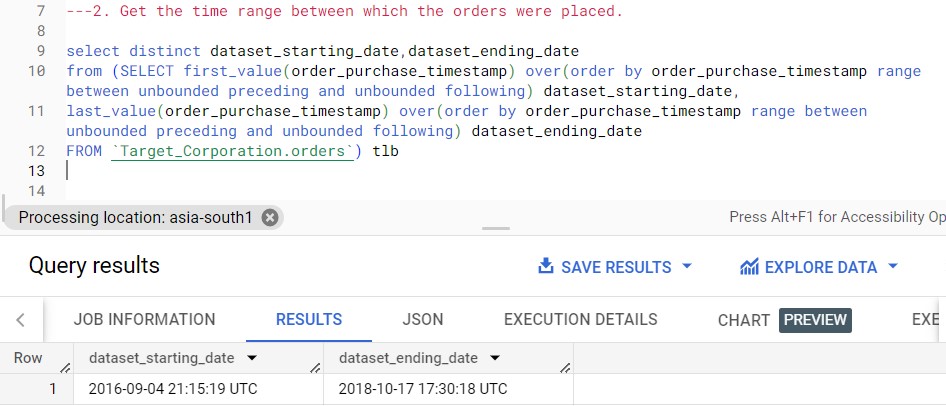
## • Query

select distinct dataset\_starting\_date,dataset\_ending\_date from (SELECT first\_value(order\_purchase\_timestamp) over(order by order\_purchase\_timestamp range between unbounded preceding and unbounded following) dataset\_starting\_date, last\_value(order\_purchase\_timestamp) over(order by order\_purchase\_timestamp range between unbounded preceding and

unbounded following) dataset\_ending\_date

FROM `Target\_Corporation.orders`) tlb

# • Screenshot of Output



## • Insights

➢ In this query, we can get the date & time when the first and last orders in our dataset were placed.

## • Recommendation

➢ In this query, we used first\_value and last\_value window function to get the first and last value of the order\_purchase\_timestamp.

1. order\_purchase\_timestamp: - Timestamp of the purchase.
2. first\_value(order\_purchase\_timestamp) over(order by order\_purchase\_timestamp range between unbounded preceding and unbounded following) dataset\_starting\_date: - this will provide first value in order\_purchase\_timestamp.
3. last\_value(order\_purchase\_timestamp) over(order by order\_purchase\_timestamp range between unbounded preceding and unbounded following) dataset\_ending\_date: - this will provide last value in order\_purchase\_timestamp.

## • Assumption

* if the dataset update with new values, the query will show updated date & time of last order placed.
* By making minor changes in query like table name and timestamp of other dataset, we can get the date & time when the first and last orders in another dataset also.

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

## • Query

select ct.city\_count,st.state\_count

from (select count(customer\_city) as city\_count, row\_number() over () as rows\_number from (Select customer\_city from `Target\_Corporation.customers` c left join `Target\_Corporation.orders` o

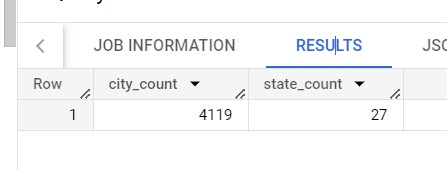
on c.customer\_id = o.customer\_id where o.customer\_id is not null group by customer\_city order by customer\_city)) ct

join (select count(customer\_state) as state\_count, row\_number() over

() as rows\_number from (Select customer\_state from `Target\_Corporation.customers` c left join `Target\_Corporation.orders` o

on c.customer\_id = o.customer\_id where o.customer\_id is not null group by customer\_state order by customer\_state)) st on ct.rows\_number = st.rows\_number;

# • Screenshot of Output



## • Insights

➢ We can count the number of unique cities and states present in our dataset.

## • Recommendation

* We join customer and order table by left join to find customers who ordered from Target Corporation. Then group by city then find the count of cities.
* In similar way, we join customer and order table by left join to find customers who ordered from Target Corporation. Then group by state then find the count of states.
* At last, we join the tables to display both the result as same table.

## • Assumption

➢ In this similar way we can find the count the Cities and States of the customers who ordered for other datasets.

**In-depth Exploration:**

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

with yearmonth as

(SELECT order\_id, customer\_id,

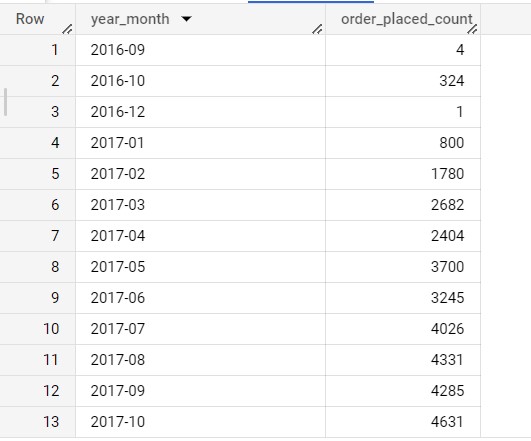
FORMAT\_TIMESTAMP("%Y-%m",order\_purchase\_timestamp) as year\_month

FROM `Target\_Corporation.orders`

order by order\_purchase\_timestamp)

select year\_month, count(\*) as order\_placed\_count from yearmonth group by year\_month order by year\_month

# • Screenshot of Output



## • Insights

➢ We can find out if no. of orders placed has increased gradually in each month, over the past years.

## • Recommendation

* FORMAT\_TIMESTAMP("%Y-%m", order\_purchase\_timestamp) as year\_month: - by using this query be extract Year and month part from purchase timestamp. Which comes out 2016-09 to 2018-10.
* Make this as Common Table Expression(cte).
* Then using this cte make group by year\_month, then count the order placed.

## • Assumption

* We can see in our data order gradually increases with year starts.
* We can see in our data order gradually decrease with year ends.

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

## • Query

with yearmonth as

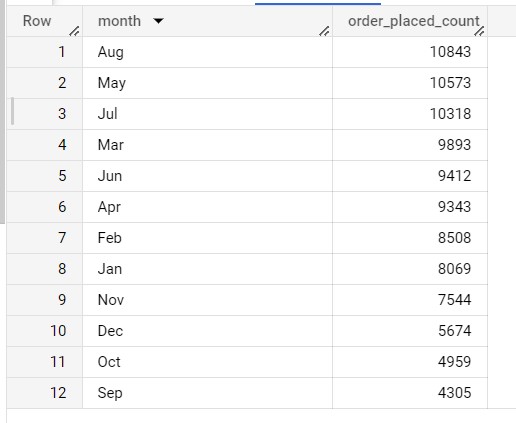
(SELECT order\_id, customer\_id,

FORMAT\_TIMESTAMP("%h",order\_purchase\_timestamp) as month FROM `Target\_Corporation.orders` order by order\_purchase\_timestamp)

select month, count(\*) as order\_placed\_count from yearmonth group by month

order by order\_placed\_count desc;

# • Screenshot of Output



## • Insights

➢ In the query, we can find out if the no. of orders placed are at peak during certain months.

## • Recommendation

* FORMAT\_TIMESTAMP("%h", order\_purchase\_timestamp) as month: - by using this query be extract month part from purchase timestamp.
* Make this as Common Table Expression(cte).
* Then using this cte make group by month, then count the order placed in that month over years.
* order by order\_placed\_count desc: - We order by order placed in descending order, to find the highest ordered month first then second and so on.

## • Assumption

* August, May and July have the highest ordered month.
* December, October and September have lowest ordered month.

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

## • Query

with hourday as

(SELECT customer\_id, order\_id, case when cast(hour as int) between 0 and 6 then 'Dawn'

when cast(hour as int) between 7 and 12 then 'Mornings'

when cast(hour as int) between 13 and 18

then 'Afternoon' else 'Night' end as Day\_timing

FROM (SELECT customer\_id, order\_id,

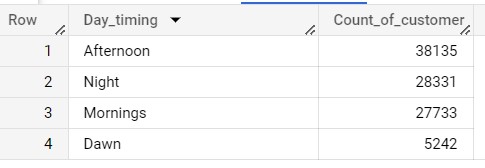
FORMAT\_TIMESTAMP("%H",order\_purchase\_timestamp) as hour

FROM `Target\_Corporation.orders`) tbl)

select Day\_timing, count(customer\_id) as Count\_of\_customer from hourday group by Day\_timing

order by Count\_of\_customer desc;

# • Screenshot of Output



## • Insights

* In this query, we can categorize the hours of a day into the given time brackets/ intervals and find out during which intervals the Brazilian customers usually order the most.
* Time intervals: - **1.** 0-6 hrs: Dawn
  1. 7-12 hrs: Mornings
  2. 13-18 hrs: Afternoon
  3. 19-23 hrs: Night

## • Recommendation

* FORMAT\_TIMESTAMP("%H",order\_purchase\_timestamp) as hour: - Using this query we extract hour from purchase timestamp and make that as subquery.
* Then using case when, we specify the Dawn, Mornings, Afternoon and Night, put it in column Day\_timing.
* Make is whole as cte.
* Then using cte we group by Day\_timing and count the order.

## • Assumption

* Most of the orders comes at Afternoon Day time.
* Least of the orders comes at Dawn Day time.

**Evolution of E-commerce orders in the Brazil region:**

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

## • Query

with city\_month as

(SELECT o.customer\_id, c.customer\_state,

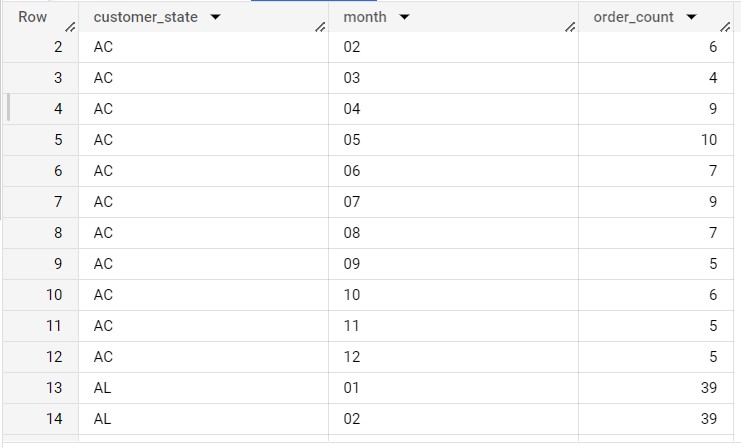
FORMAT\_TIMESTAMP("%m",o.order\_purchase\_timestamp) as month

from `Target\_Corporation.orders` o join `Target\_Corporation.customers` c on o.customer\_id = c.customer\_id)

select customer\_state, month, count(customer\_id) as order\_count from city\_month

group by customer\_state,month order by customer\_state,month

# • Screenshot of Output



## • Insights

➢ In this query, we can get the no. of orders placed in each state, in each month by our customers.

## • Recommendation

* FORMAT\_TIMESTAMP("%m", o.order\_purchase\_timestamp) as month: - Extract month from purchase timestamp.
* Join orders table and customer table.
* Make this as cte (Common table Expression).
* Use cte and group by customer\_state and month part.
* Then count customer\_id.
* Order by customer\_state and month.

## • Assumption

* We can observe in the screenshot: - state name, month and order count.
* State: - SP have highest no. of orders, then RJ State, then MG States and so on

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

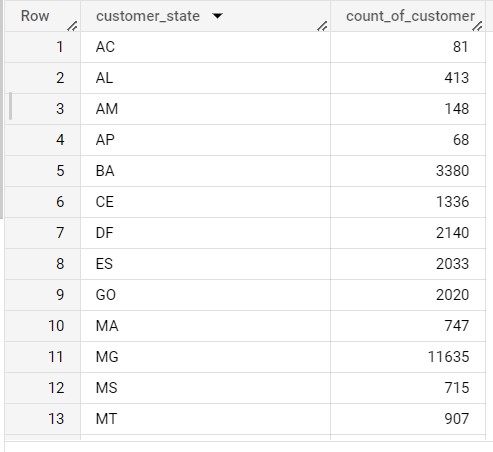
## • Query

SELECT customer\_state, count(distinct customer\_id) as count\_of\_customer

from `Target\_Corporation.customers`

group by customer\_state order by customer\_state

# • Screenshot of Output



## • Insights

➢ In this query, we can get the no. of unique customers present in each state.

## • Recommendation

* Group by customer\_state in table customers.
* Count distinct customers\_id.

## • Assumption

* SP state have highest no. of customer, then RJ have second highest customer and the MG state.
* **Number of customers is directly proposal to Number order.**

**Impact on Economy: Analyze the money movement by ecommerce 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).**

## • Query

with monthyear as

(select o.order\_id,FORMAT\_TIMESTAMP("%m",o.order\_purchase\_timestamp) as month,

FORMAT\_TIMESTAMP("%Y",o.order\_purchase\_timestamp) as year, payment\_value

from `Target\_Corporation.orders` o

join `Target\_Corporation.payments` p

on o.order\_id = p.order\_id)

select tlb1.month, tlb1.year, tlb1.total\_monthlycost2017, tlb2.year,

tlb2.total\_monthlycost2018, round((((tlb2.total\_monthlycost2018-

tlb1.total\_monthlycost2017)/tlb1.total\_monthlycost2017)\*100),2) as percentage\_increase,

from (select monthyear.month,monthyear.year, sum(payment\_value) as total\_monthlycost2017 from monthyear

where year = '2017' and cast(monthyear.month as int) between 01 and 08

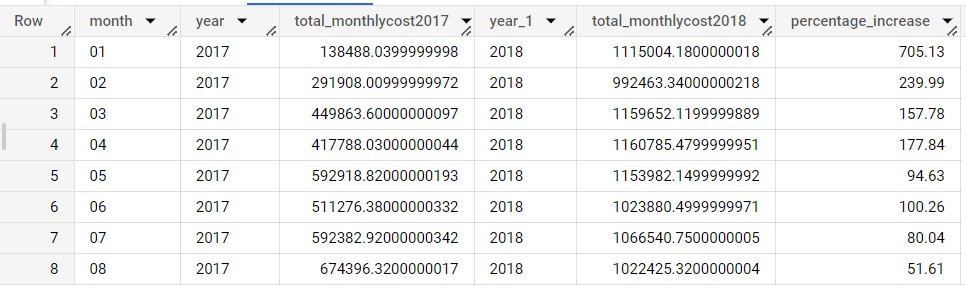
group by monthyear.month, monthyear.year order by monthyear.month) tlb1

join (select monthyear.month,monthyear.year, sum(payment\_value) as total\_monthlycost2018 from monthyear

where year = '2018' and cast(monthyear.month as int) between 01 and 08

group by monthyear.month, monthyear.year order by monthyear.month) tlb2 on tlb1.month = tlb2.month order by tlb1.month;

# • Screenshot of Output



• Insights

➢ Percentage\_ increase between months(January - August) over years.

## • Recommendation

* FORMAT\_TIMESTAMP("%m",o.order\_purchase\_timestamp) as month: - Extract month from purchase timestamp.
* FORMAT\_TIMESTAMP("%Y",o.order\_purchase\_timestamp) as year: - Extract year from purchase time stamp.
* Make this as cte.
* In one subquery find sum of months payment over year 2017.
* And in one subquery find sum of months payment over year 2018.
* Join both sub query.
* And find percentage increase over months using formula: - round((((tlb2.total\_monthlycost2018tlb1.total\_monthlycost2017)/tlb1.total\_monthlycost2017)\*100),2) as percentage\_increase.
* Round function helps to round off 2 digits after point(.).
* Between Month January to August.

## • Assumption

➢ Highest percentage increase over month is in January: - 705.13% ➢ Lowest percentage increase over month is in August: - 51.61% ➢ We can find percentage increase over years.

with cte as(

select FORMAT\_TIMESTAMP("%Y",o.order\_purchase\_timestamp) as year,

o.order\_id,FORMAT\_TIMESTAMP("%m",o.order\_purchase\_timestamp) as month, payment\_value,

from `Target\_Corporation.orders` o join `Target\_Corporation.payments` p

on o.order\_id = p.order\_id)

,cte2 as(

select cte.year, sum(payment\_value) total\_payment\_yearly from cte

where cast(cte.month as int) between 01 and 08 group by cte.year order by cte.year)

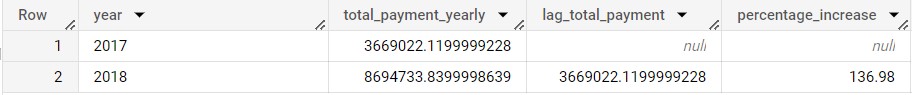
,cte3 as

(select \*, lag(cte2.total\_payment\_yearly) over(order by cte2.year) as lag\_total\_payment from cte2)

select \*,round((((cte3.total\_payment\_yearly-

cte3.lag\_total\_payment)/cte3.lag\_total\_payment)\*100),2) as percentage\_increase from cte3

order by cte3.year;



➢ Percentage Increase over Year 2017-2018 is 136.98%.

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

## • Query

SELECT distinct customer\_state,

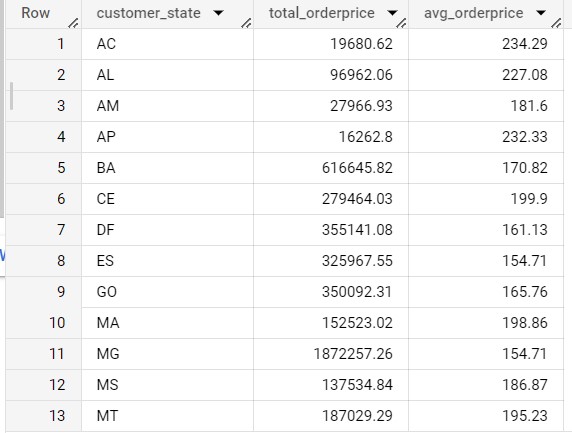
sum(payment\_value) over(partition by customer\_state order by

customer\_state) as total\_orderprice,

round(avg(payment\_value) over(partition by customer\_state order by customer\_state),2) as avg\_orderprice from `Target\_Corporation.customers` c join `Target\_Corporation.orders` o on c.customer\_id = o.customer\_id join `Target\_Corporation.payments` p

on o.order\_id = p.order\_id order by customer\_state;

# • Screenshot of Output



## • Insights

➢ In this query, we can fetch the total price and the average price of orders for each state.

## • Recommendation

* round(avg(payment\_value) over(partition by customer\_state order by customer\_state),2) as avg\_orderprice: - using this window function of average of payment\_value of states
* We join three Tables customers, orders and payments.
* Customer table helps in finding state.
* Order table helps in order details.
* Payments table helps in finding payment\_value.

## • Assumption

* PB State have highest average price of order: - 248.33
* Highest total\_orderprice SP State: - 5998226.96
* SP State have lowest average price of order: - 137.5 ➢ Lowest total\_orderprice AP State: - 16262.8
* We see SP State has high total sale but low Average

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

## • Query

SELECT distinct customer\_state,

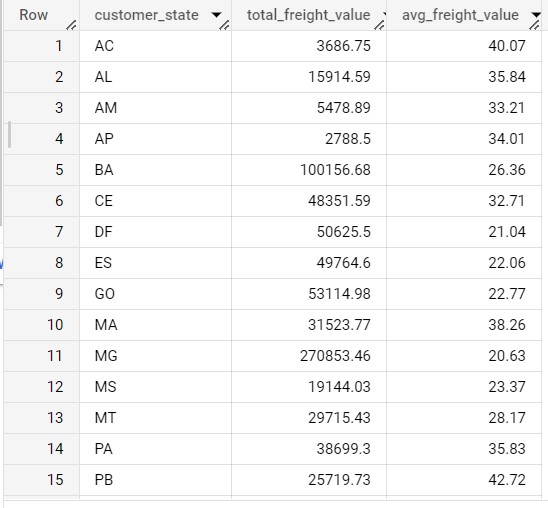
sum(freight\_value) over(partition by customer\_state order by customer\_state) as total\_freight\_value,

round(avg(freight\_value) over(partition by customer\_state order by customer\_state),2) as avg\_freight\_value from `Target\_Corporation.customers` c

join `Target\_Corporation.orders` o

on c.customer\_id = o.customer\_id join `Target\_Corporation.order\_items` oi on o.order\_id = oi.order\_id order by customer\_state;

# • Screenshot of Output



## • Insights

➢ In this Query, we can fetch the total freight value and the average freight value of orders for each state.

## • Recommendation

* round(avg(freight\_value) over(partition by customer\_state order by customer\_state),2) as avg\_freight\_value
* We join three Tables customers, orders and order\_item.
* Customer table helps in finding state.
* Order table helps in order details.
* Payments table helps in finding freight\_value.

## • Assumption

* RR State have highest average Freight of order: - 42.98
* Highest Total Freight value: - SP State 718723.07
* SP State have lowest average Freight of order: - 15.15 ➢ Lowest Total Freight value: - RR State 2235.19

**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.

## • Query

SELECT order\_id, customer\_id,order\_status, case

when order\_delivered\_customer\_date is null then 'Cancelled/Unavailable'

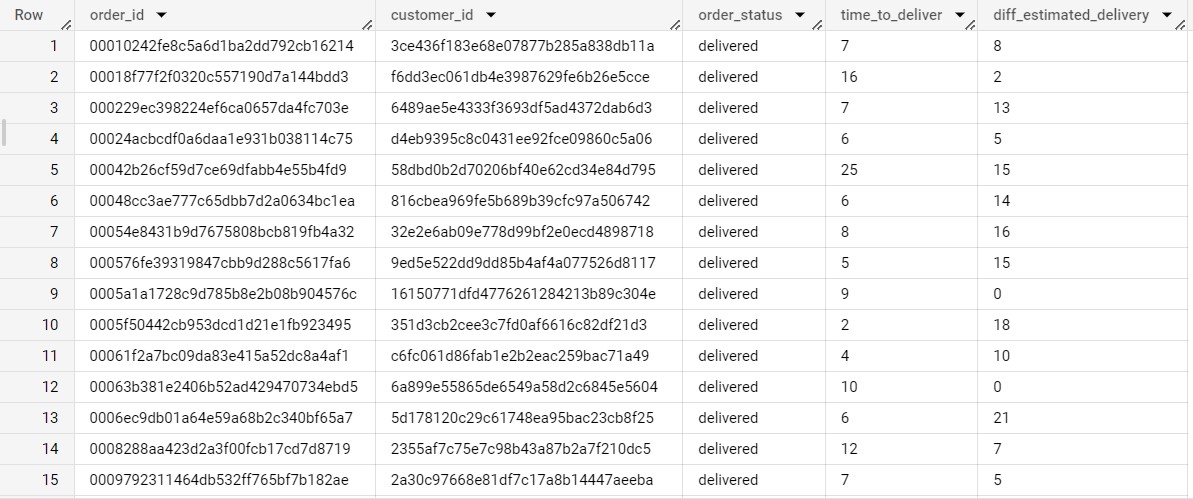
else cast(date\_diff(order\_delivered\_customer\_date, order\_purchase\_timestamp, day) as string)

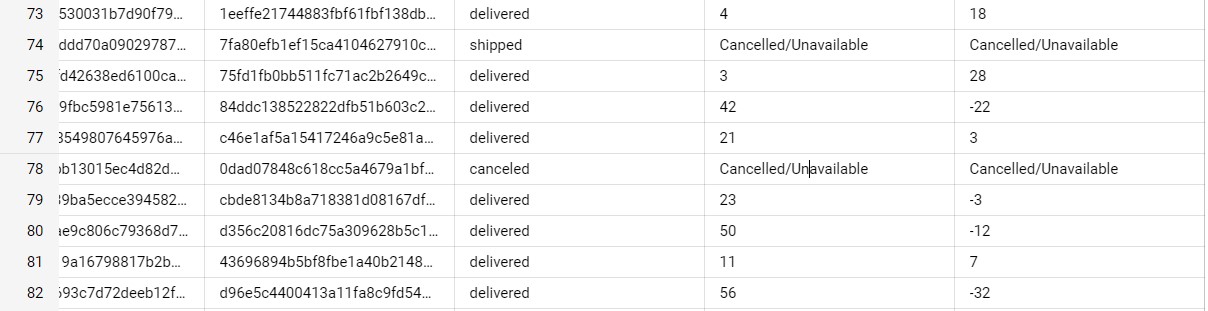
end as time\_to\_deliver, case

when order\_delivered\_customer\_date is null then 'Cancelled/Unavailable' else

cast(date\_diff(order\_estimated\_delivery\_date, order\_delivered\_customer\_date, day) as string) end as diff\_estimated\_delivery from `Target\_Corporation.orders` order by order\_id

# • Screenshot of Output





## • Insights

➢ In this Query, we 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

## • Recommendation

* date\_diff(order\_delivered\_customer\_date, order\_purchase\_timestamp, day):- this function help in find the difference in dates.
* Case when helps in finding cancelled and Unavailable.

## • Assumption

* time\_to\_deliver: - Differences in order\_delivered\_customer\_date and order\_purchase\_timestamp.
* diff\_estimated\_delivery: - Differences in order\_estimated\_delivery\_date and order\_delivered\_customer\_date
* In row 74 order is shipped but never reach the customer.
* In row 76,79,80 and 82 has negative values then it means order is deliver more days then estimated delivery time.

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

## • Query

(select customer\_state,avg\_freight\_value,'highest' as level from(select\*,dense\_rank() over(order by avg\_freight\_value desc) as ranking

from(SELECT distinct customer\_state,

round(avg(freight\_value) over(partition by customer\_state order by customer\_state),2) as avg\_freight\_value, from `Target\_Corporation.customers` c join `Target\_Corporation.orders` o

on c.customer\_id = o.customer\_id join `Target\_Corporation.order\_items` oi on o.order\_id = oi.order\_id order by avg\_freight\_value) a order by ranking) b where ranking <= 5)

UNION distinct

(select customer\_state,avg\_freight\_value,'lowest' as level from(select\*,dense\_rank() over(order by avg\_freight\_value) as ranking

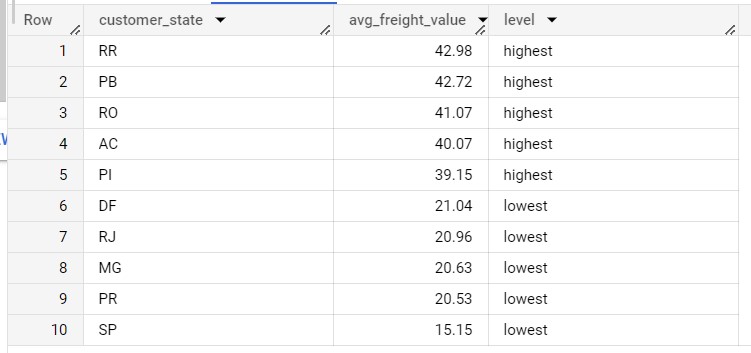
from(SELECT distinct customer\_state,

round(avg(freight\_value) over(partition by customer\_state order by customer\_state),2) as avg\_freight\_value, from `Target\_Corporation.customers` c join `Target\_Corporation.orders` o

on c.customer\_id = o.customer\_id join `Target\_Corporation.order\_items` oi on o.order\_id = oi.order\_id order by avg\_freight\_value) a order by ranking) b

where ranking <= 5) order by avg\_freight\_value desc;

# • Screenshot of Output



## • Insights

➢ In This Query, we can find the top 5 & the bottom 5 states arranged in increasing order of the average freight value.

## • Recommendation

* round(avg(freight\_value) over(partition by customer\_state order by customer\_state),2) as avg\_freight\_value: - find the average of freight\_value.
* Make this in subquery and the put dense\_rank() on subquery to find top 5 highest avg freight value. dense\_rank() over(order by avg\_freight\_value desc) as ranking.
* In Similar way we find top 5 lowest avg freight value. dense\_rank() over(order by avg\_freight\_value) as ranking.
* And at last union both of them.

## • Assumption

* 1-5 top 5 highest avenge freight value states.
* RR State have highest avenged freight value state. ➢ 6-10 top 5 lowest avenge freight value states
* SP State have lowest avenged freight value state.

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

## • Query

with cte as

(select distinct customer\_state, round(avg(c.time\_to\_deliver),2) as avg\_time\_to\_deliver

from (SELECT customer\_state, date\_diff(order\_delivered\_customer\_date, order\_purchase\_timestamp, day) as time\_to\_deliver from `Target\_Corporation.customers` c join `Target\_Corporation.orders` o

on c.customer\_id = o.customer\_id) c group by customer\_state)

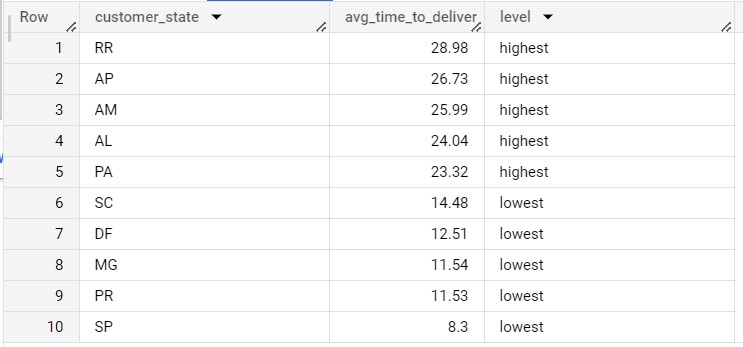
(select customer\_state, avg\_time\_to\_deliver,'highest' as level from(select customer\_state, cte.avg\_time\_to\_deliver,dense\_rank() over(order by cte.avg\_time\_to\_deliver desc) as ranking from cte) where ranking <= 5)

UNION ALL

(select customer\_state, avg\_time\_to\_deliver,'lowest' as level from(select customer\_state, cte.avg\_time\_to\_deliver,dense\_rank() over(order by cte.avg\_time\_to\_deliver) as ranking from cte) where ranking <= 5)

order by avg\_time\_to\_deliver desc

# • Screenshot of Output



## • Insights

➢ In this Query, we can find the top 5 & the bottom 5 states arranged in increasing order of the average delivery time.

## • Recommendation

* date\_diff(order\_delivered\_customer\_date, order\_purchase\_timestamp, day) as time\_to\_deliver: - date difference to find delivery time.
* Avg() on delivery time of states.
* Make this in cte and the put dense\_rank() on cte to find top 5 highest avg freight value. dense\_rank() over(order by cte.avg\_time\_to\_deliver desc) as ranking
* In Similar way we find top 5 lowest avg freight value. dense\_rank() over(order by cte.avg\_time\_to\_deliver) as ranking ➢ And at last union both of them.

## • Assumption

➢ 1-5 top 5 highest avenge time to deliver states. ➢ RR State have Highest delivery time average. ➢ 6-10 top 5 lowest avenge time to deliver states ➢ SP State have lowest average Delivery time.

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

## • Query

with cte5 as

(select customer\_state, round(avg(diff\_estimated\_delivery),2) as

avg\_diff\_estimated\_delivery

from (select customer\_state, date\_diff(order\_estimated\_delivery\_date, order\_delivered\_customer\_date, day) as diff\_estimated\_delivery

from `Target\_Corporation.customers` c join `Target\_Corporation.orders` o

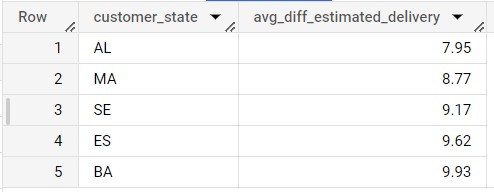
on c.customer\_id = o.customer\_id where order\_status = 'delivered') f group by customer\_state)

select customer\_state, avg\_diff\_estimated\_delivery,

from(select customer\_state, avg\_diff\_estimated\_delivery,dense\_rank() over(order by avg\_diff\_estimated\_delivery) as ranking from cte5) where ranking <= 5

order by avg\_diff\_estimated\_delivery;

# • Screenshot of Output



## • Insights

* In this Query, we find top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
* We can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

## • Recommendation

* date\_diff(order\_estimated\_delivery\_date, order\_delivered\_customer\_date, day) as diff\_estimated\_delivery: - Date Difference to find estimated date of delivery.
* Join two table customers and orders. ➢ Where clause to filter delivered ordered ➢ Make it as cte.
* Using cte us dense\_rank() we find the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

## • Assumption

➢ AL State fastest order delivery

**Analysis based on the payments:**

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

## • Query

with cte5 as

(SELECT FORMAT\_TIMESTAMP("%Y",o.order\_purchase\_timestamp) as year, FORMAT\_TIMESTAMP("%m",o.order\_purchase\_timestamp) as month,

p.payment\_type

from `Target\_Corporation.orders` o

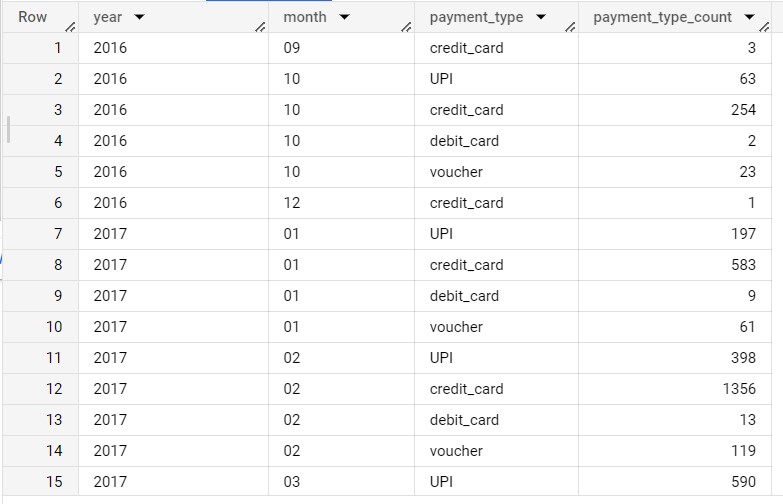
join `Target\_Corporation.payments` p

on o.order\_id = p.order\_id order by year, month,p.payment\_type)

select cte5.year,cte5.month,payment\_type, count(payment\_type) as payment\_type\_count from cte5

group by cte5.year, cte5.month, payment\_type order by year, month,payment\_type;

# • Screenshot of Output



## • Insights

➢ In this Query, we can count the no. of orders placed using different payment methods in each month over the past years.

## • Recommendation

* FORMAT\_TIMESTAMP("%Y",o.order\_purchase\_timestamp) as year, FORMAT\_TIMESTAMP("%m",o.order\_purchase\_timestamp) as month: - Extract year and month from purchase timestamp.
* Join tables orders and payments.
* We take out payment\_type from payments table.
* Make it as cte.
* Using cte we group by year, month and payment\_type.
* Count payment\_type.

## • Assumption

* Mostly people are using Credit Card to pay.
* UPI Second highest.
* Third is Vouches.

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

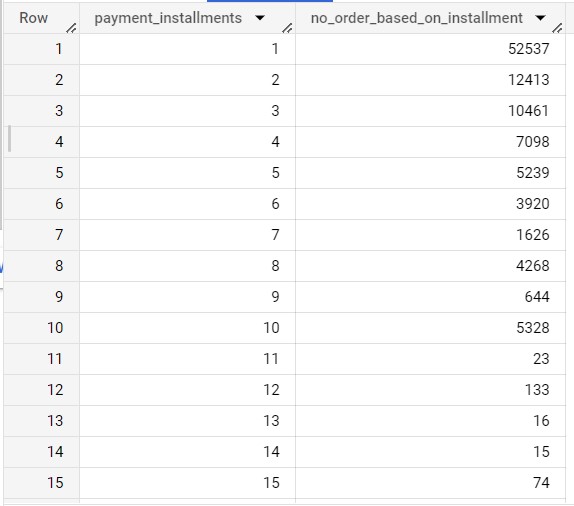
## • Query

select payment\_installments, count(order\_id) as no\_order\_based\_on\_installment from `Target\_Corporation.payments`

where payment\_installments >= 1 and payment\_value > 0

group by payment\_installments order by payment\_installments;

# • Screenshot of Output



## • Insights

➢ In this Query, we can count the no. of orders placed based on the no. of payment installments where at least one installment has been successfully paid.

## • Recommendation

* Using Table payment.
* where clause for filter payment\_installments >= 1 and payment\_value > 0 ➢ Group by payment installment.
* Count order id.

## • Assumption

➢ Most of the people paid in 1-2 instalments

**Submitted BY: - Smit Singh**

**Batch: - dsml-june-23-beginner-mor-mon-batch**