Name: Sirugudu Tejeswara Rao
Mail: tejavishnu2000@gmail.com
Mobile No: 9110703897

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.

Query:

```
SELECT column_name, data_type
FROM target_e_commerce.INFORMATION_SCHEMA.COLUMNS
WHERE table_name='customers'
```

Output :

JOB INFORMATION | RESULTS | CHART | PREVIEW

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

```
Insights: Customers table contains 5 different columns with 2 different data types
STRING AND INT64 associated with it.
Recommendations: NA
```

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

Query:

```
SELECT MIN(order_purchase_timestamp) AS First_Order,
       MAX(order_purchase_timestamp) AS Last_Order
FROM target_e_commerce.orders
```

Output:

Query results | SAVE RESULTS | EXPLORE DATA

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

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

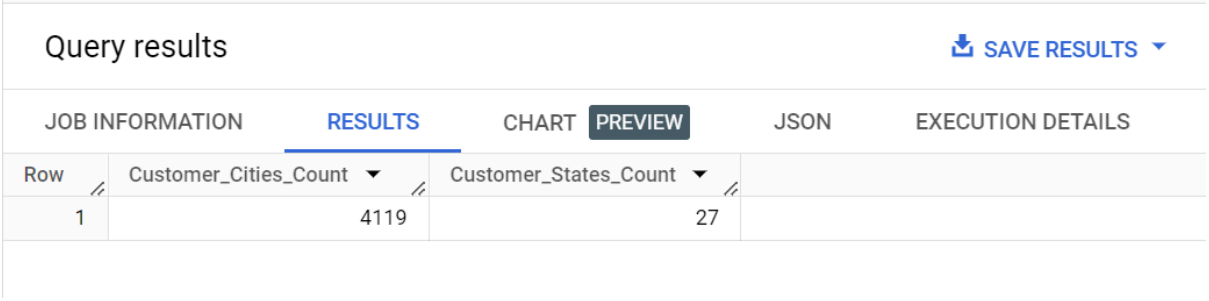
Insights: Target was placed first order on 04-09-2016 and last order on 17-10-2018.
Recommendations: NA

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

Query:

```
SELECT count(distinct customer_city) AS Customer_Cities_Count,
       count(distinct customer_state) AS Customer_States_Count
FROM `target_e_commerce.customers`
```

Output:



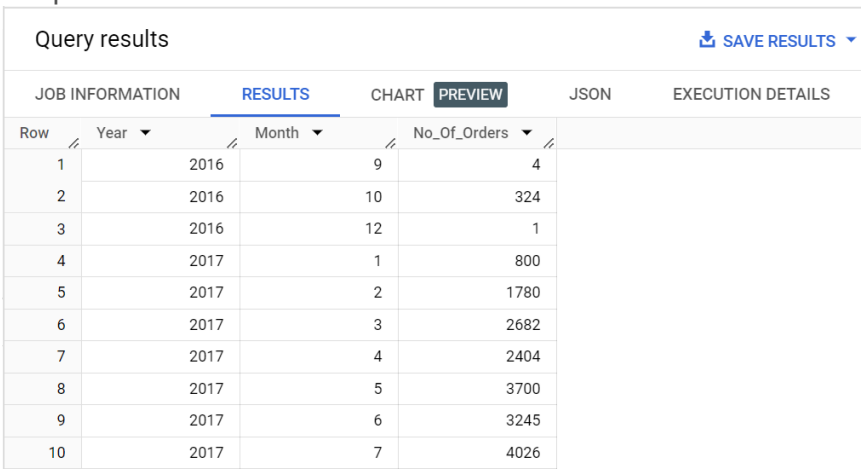
Insights: All customers belong to 27 different states and 4119 different cities.
Recommendations: NA

In-depth Exploration:

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

```
SELECT  EXTRACT(YEAR FROM order_purchase_timestamp) AS Year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
        count(order_id) AS No_Of_Orders
FROM `target_e_commerce.orders`
GROUP BY Year, Month
ORDER BY Year, Month
```

Output:



Row	Year	Month	No_Of_Orders
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

Insights:
High sales during festival seasons specially in January. Keep on increasing the order in 2017 from Jan to November and reaches all time high sales in January 2018 and keep maintain same level of orders till august 2018.

Low sales happened at starting and ending of orders placed especially in the year 2016 and September and October of 2018. All time less no.of orders placed in December 2016 which is 1.

Recommendations:
Keep maintain sufficient no.of employees to delivery the products on time. If required hire the no.of people to maintain the perfection and in time delivery.

Sale the all old products during the festive seasons by combining it with new one's such as combo offers, give more discount, Award winning prices ..etc.

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

Query:

```
SELECT EXTRACT(MONTH FROM order_purchase_timestamp) AS Month,
       count(order_id) AS No_Of_Orders
FROM `target_e_commerce.orders`
GROUP BY Month
ORDER BY Month
```

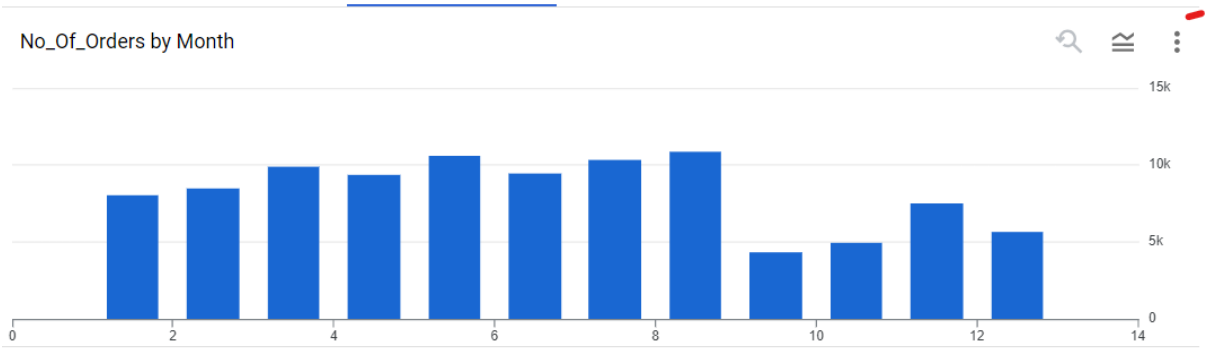
Output:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	Month	No_Of_Orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

Insights: We could see continuously sales increase for the months from Jan to May and same way increases from June to oct. Highest sales placed in the month of august and it seems some offers and near some festival in Brazil.



Recommendations: During festive seasons or event/occasional seasons need to maintain stock alive and hire more par time employees to do the work quickly especially in the months may, July and august cross 10k orders.

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

- 0-6 hrs : Dawn
- 7-12 hrs : Mornings
- 13-18 hrs : Afternoon
- 19-23 hrs : Night

Query:

```
Note: I wrote query with exactly 6:00:00 also included in Dawn, 12:00:00 included
in mornings, 18 : 00 : 00 included in afternoon.
```

```sql
WITH CTE AS
(
SELECT
CASE
WHEN (EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 and 5) OR (EXTRACT(HOUR
FROM order_purchase_timestamp) = 6 AND EXTRACT(MINUTE FROM
order_purchase_timestamp) = 0 AND EXTRACT(SECOND FROM order_purchase_timestamp) =
0)  THEN 'Dawn'
WHEN (EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 6 and 11) OR
(EXTRACT(HOUR FROM order_purchase_timestamp) = 12 AND EXTRACT(MINUTE FROM
order_purchase_timestamp) = 0 AND EXTRACT(SECOND FROM order_purchase_timestamp) =
0) THEN 'Mornings'
WHEN (EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 12 and 17) OR
(EXTRACT(HOUR FROM order_purchase_timestamp) = 18 AND EXTRACT(MINUTE FROM
order_purchase_timestamp) = 0 AND EXTRACT(SECOND FROM order_purchase_timestamp) =
0) THEN 'Afternoon'
WHEN (EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 18 and 23) THEN 'Night'
END AS Timing
FROM `target_e_commerce.orders`
)

SELECT Timing, count(*) AS Number
FROM CTE
GROUP BY Timing
ORDER BY Number
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	Timing	Number
1	Afternoon	38365
2	Night	34096
3	Mornings	22240
4	Dawn	4740

Insights: Brazilian customers mostly wants to place their orders at Afternoon time and then night time. Least orders they placed during Drawn time.

Recommendations: Prepare/Ready more orders during afternoon and night timings.

Evolution of E-commerce orders in the Brazil region:

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

Query:

```sql
SELECT customer_state, month, count(order_id)
FROM
(
SELECT c.customer_state, EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
o.order_id
FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o ON
c.customer_id = o.customer_id
)
GROUP BY customer_state,month
ORDER BY customer_state,month
```

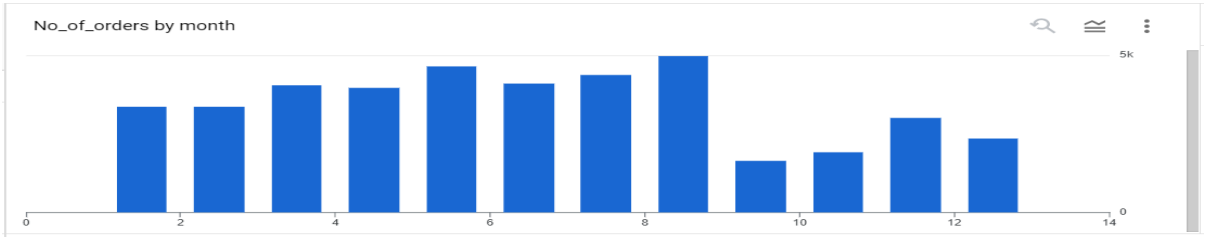
Output:

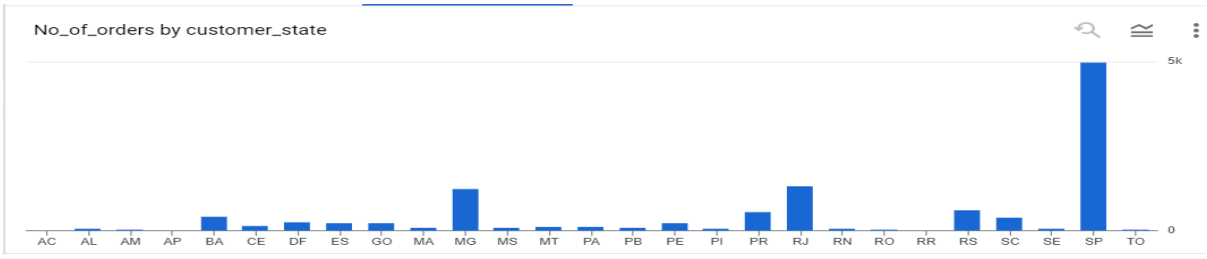
Query results

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

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
11	AC	11	5
12	AC	12	5

Insights: Individual insights for no.of orders month wise and state wise.
We can see SP state as huge no.of orders. And aug month has more more orders across all states.





Recommendations: Maintain and distribute the no.of people among the states and shuffle them month on month orders placed.

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

Query:

```sql
SELECT c.customer_state, count(distinct c.customer_id) AS No_of_customers

FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
GROUP BY customer_state
ORDER BY No_of_customers
```

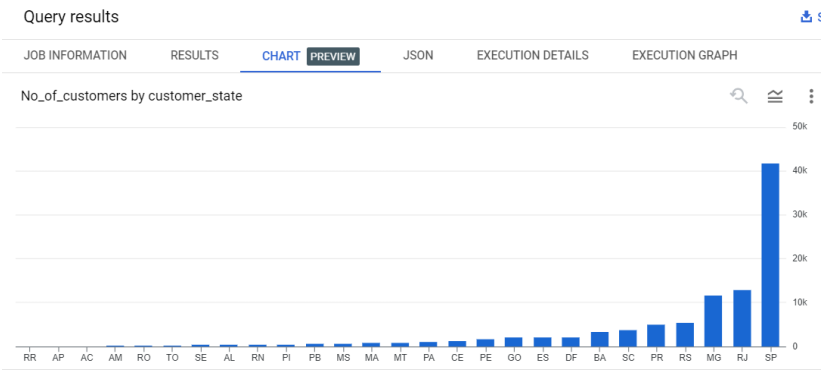
Output:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	customer_state	No_of_customers
1	RR	46
2	AP	68
3	AC	81
4	AM	148
5	RO	253
6	TO	280
7	SE	350
8	AL	413
9	RN	485
10	PI	495
11	PB	536

Insights: Maximum no.of customers are there in SP state and Minimum no.of customers RR state.



Recommendations: Better to expand stores at SP state and If possible try to arrange possible manufacturing unit there.

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.

Query: Month on month percentage increase from 2017 to 2018(Included only from Jan to Aug Only)

```
WITH CTE AS
(
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year, EXTRACT(MONTH FROM
order_purchase_timestamp) AS month,payment_value
FROM target_e_commerce.orders o JOIN target_e_commerce.payments p
ON o.order_id = p.order_id
),
CTE1 AS
(
  SELECT month, SUM(payment_value) AS Total_2017_cost
  FROM CTE
  WHERE year = 2017 AND (month BETWEEN 1 and 8)
  GROUP BY month
),
```

```
CTE2 AS
(
  SELECT month, SUM(payment_value) AS Total_2018_cost
  FROM CTE
  WHERE year = 2018 AND (month BETWEEN 1 and 8)
  GROUP BY month

)

SELECT *, (CTE2.Total_2018_cost - CTE1.Total_2017_cost) / CTE1.Total_2017_cost *
100 AS persentage_increase
FROM CTE1 JOIN CTE2 USING(month)
ORDER BY month
```

Output:

Query results

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

Row	month	Total_2017_cost	Total_2018_cost	persentage_increase
1	1	138488.0399999...	1115004.180000...	705.12669541716627
2	2	291908.0099999...	992463.3400000...	239.99181454458994
3	3	449863.6000000...	1159652.119999...	157.77860667099682
4	4	417788.0300000...	1160785.479999...	177.84077011492977
5	5	592918.8200000...	1153982.149999...	94.627343756771879
6	6	511276.3800000...	1023880.499999...	100.2596912456606
7	7	592382.9200000...	1066540.750000...	80.042454633903745
8	8	674396.3200000...	1022425.320000...	51.606005204773041

Query: Total percentage increase from 2017 to 2018(Included only from Jan to Aug Only)

```
WITH CTE AS
(
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year, EXTRACT(MONTH FROM
order_purchase_timestamp) AS month,payment_value
FROM target_e_commerce.orders o JOIN target_e_commerce.payments p
ON o.order_id = p.order_id
),
CTE1 AS
(
  SELECT month, SUM(payment_value) AS Total_2017_cost
  FROM CTE
  WHERE year = 2017 AND (month BETWEEN 1 and 8)
  GROUP BY month
),
CTE2 AS
(
  SELECT month, SUM(payment_value) AS Total_2018_cost
  FROM CTE
  WHERE year = 2018 AND (month BETWEEN 1 and 8)
  GROUP BY month

)

select (sum(Total_2018_cost)-sum(Total_2017_cost))/sum(Total_2017_cost)*100 AS
Total_persentage_increase FROM CTE1 JOIN CTE2 USING(month)
```

Output:

Query results

JOB INFORMATION | RESULTS

Row	Total_persentage_increase
1	136.97687164666004

Insights:
Observed that the January month has highest payment % increased from 2017 to 2018 and August month has less payments % increased from 2017 to 2018.

Recommendations: Make sure to maintain payment transactions properly in the month of January.

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

```
SELECT c.customer_state, SUM(price) AS Total_price,AVG(price) AS Avg_price
FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
JOIN target_e_commerce.order_items order_I
ON o.order_id = order_I.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state
```

Output:

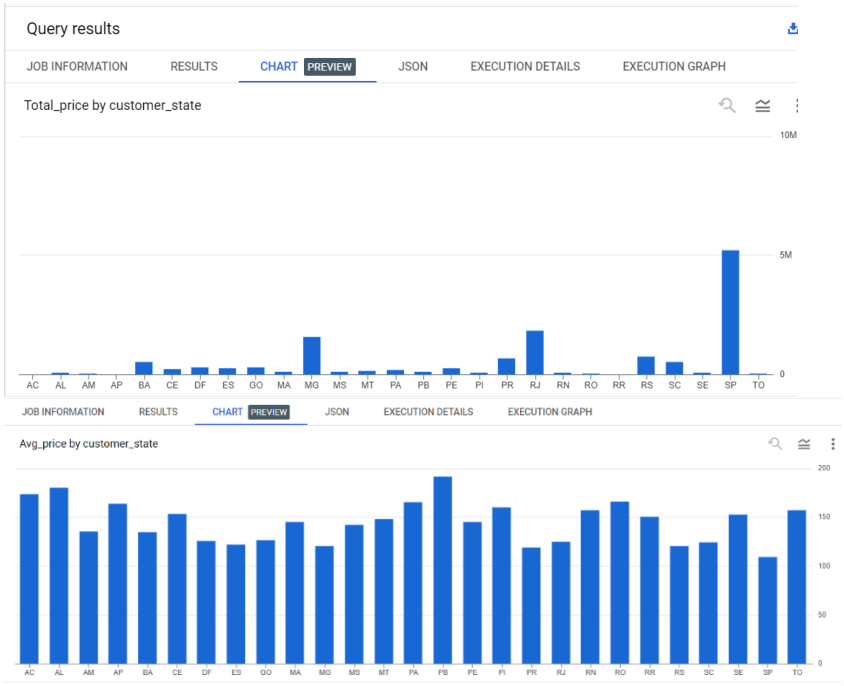
Query results

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

Row	customer_state	Total_price	Avg_price
1	AC	15982.94999999...	173.7277173913...
2	AL	80314.81000000...	180.8892117117...
3	AM	22356.84000000...	135.4959999999...
4	AP	13474.29999999...	164.3207317073...
5	BA	511349.9900000...	134.6012082126...
6	CE	227254.7099999...	153.7582611637...
7	DF	302603.9399999...	125.7705486284...
8	ES	275037.3099999...	121.9137012411...
9	GO	294591.9499999...	126.2717316759...
10	MA	119648.2199999...	145.2041504854...
11	MG	1585308.029999...	120.7485741488...
12	MS	116812.6399999...	142.6283760683...

Insights:

Total price is higher at the state ‘SP’. Low price is at ‘RR’ state, so more money spend to do the sales and get less income and profits some time may get losses.
Total average price is high at ‘PB state and so more income and profits generated from this state. Low average price at ‘SP’.
Even If the more order price got in SP state but average price is less which means more no.of less expensive products sold at this state. Same way more expensive products sold at ‘PB’ state.



Recommendations: Please categorise the products like more expensive, less expensive and average expensive products and distribute to maintain more selling product to corresponding states.

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

Query:

```
SELECT customer_state, SUM(freight_value) AS Total_freight_value,AVG(freight_value)
AS Avg_freight_value
FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
JOIN target_e_commerce.order_items order_I
ON o.order_id = order_I.order_id
GROUP BY c.customer_state
ORDER BY customer_state
```

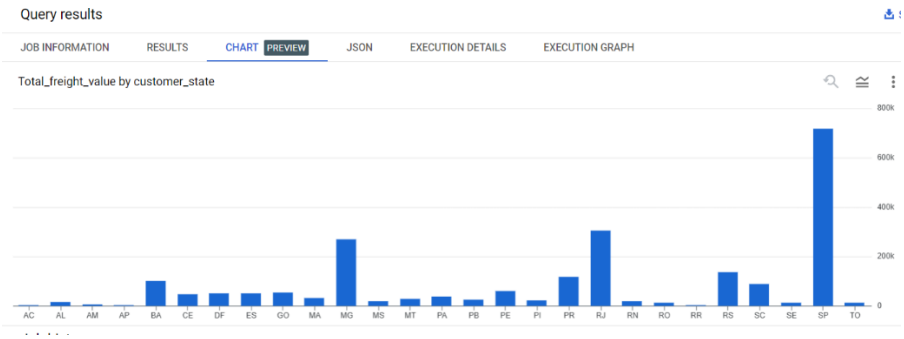
Output:

Query results

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

Row	customer_state	Total_freight_value	Avg_freight_value
1	AC	3686.749999999...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.889999999...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...
6	CE	48351.58999999...	32.71420162381...
7	DF	50625.49999999...	21.04135494596...
8	ES	49764.59999999...	22.05877659574...
9	GO	53114.97999999...	22.76681525932...
10	MA	31523.77000000...	38.25700242718...
11	MG	270853.4600000...	20.63016680630...

Insights:
Spent more total amount to deliver the product sales for ‘SP’ state and spent less delivery amount for ‘RR’ state.
Best average delivery amount spent on state ‘SP’ which is very low and worst average delivery amount spend on state RR which is very high.
So RR state is less total delivery amount and high cost to delivery products. SR is the best state for delivery the products easily.



Recommendations: Better to increase the sales on SR state by giving discounts, offers and some other gifts. But in RR state remove cheapest product orders and maintain consistency with limited one.

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:

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

Query:

```
SELECT order_id,
DATE_DIFF(EXTRACT(DATE FROM order_delivered_customer_date), EXTRACT(DATE FROM
order_purchase_timestamp), DAY) AS time_to_deliver,
DATE_DIFF(EXTRACT(DATE FROM order_delivered_customer_date), EXTRACT(DATE FROM
order_estimated_delivery_date), DAY) AS diff_estimated_delivery
FROM target_e_commerce.orders
ORDER BY order_id
```

Output:

Query results

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

Row	order_id	time_to_deliver	diff_estimated_delivery
1	1b3190b2dfa9d789e1f14c05b...	208	188
2	ca07593549f1816d26a572e06...	210	181
3	47b40429ed8cce3aee9199792...	191	175
4	2fe324febf907e3ea3f2aa9650...	190	167
5	285ab9426d6982034523a855f...	195	166
6	440d0d17af552815d15a9e41a...	196	165
7	c27815f7e3dd0b926b5855262...	188	162
8	0f4519c5f1c541ddec9f21b3bd...	194	161
9	d24e8541128cea179a11a6517...	175	161
10	2d7561026d542c8dbd8f0daea...	188	159
11	6e82dcfb5eada6283dba34f16...	183	155
12	2fb597c2f772eca01b1f5c561b...	195	155

Insights: Maximum days to deliver the products are 210 is very high and 181 days delay from estimate time. 188 days is the highest delayed delivery from estimated date. Customer is not satisfied with delayed deliveries.

Recommendations: Please make sure to reduce the delayed deliveries by increasing the people to delivery and in case any delay make sure to give proper justification to the customer. In case any damaged or insufficient stock to tell the customer will replace order and it takes some time to achieve customer satisfactions.

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

Query:
Top 5 states with highest average freight value

```
WITH CTE AS
(
SELECT customer_state, AVG(freight_value) AS Avg_freight_value
FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
JOIN target_e_commerce.order_items order_I
ON o.order_id = order_I.order_id
GROUP BY c.customer_state
```

```
)
#Top 5 states with highest average freight value
SELECT customer_state, Avg_freight_value
FROM CTE
ORDER BY Avg_freight_value desc
LIMIT 5;
```

Using Dense rank, Here we get top states with top 5 highest average freight value

```
WITH CTE AS
(
SELECT customer_state, AVG(freight_value) AS Avg_freight_value
FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
JOIN target_e_commerce.order_items order_I
ON o.order_id = order_I.order_id
GROUP BY c.customer_state
)

SELECT customer_state, Avg_freight_value
FROM
(
SELECT customer_state, Avg_freight_value,
DENSE_RANK() OVER(ORDER BY Avg_freight_value desc) AS rnk
FROM CTE
ORDER BY rnk
)
WHERE rnk<=5
```

Top 5 states with lowest average freight value

```
WITH CTE AS
(
SELECT customer_state, AVG(freight_value) AS Avg_freight_value
FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
JOIN target_e_commerce.order_items order_I
ON o.order_id = order_I.order_id
GROUP BY c.customer_state
)
#Top 5 states with highest average freight value
SELECT customer_state, Avg_freight_value
FROM CTE
ORDER BY Avg_freight_value asc
LIMIT 5;
```

Using Dense rank, Here we get top states with top 5 lowest average freight value

```
WITH CTE AS
(
SELECT customer_state, AVG(freight_value) AS Avg_freight_value
FROM target_e_commerce.customers c LEFT JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
JOIN target_e_commerce.order_items order_I
ON o.order_id = order_I.order_id
GROUP BY c.customer_state
)

SELECT customer_state, Avg_freight_value
FROM
(
```

```
SELECT customer_state, Avg_freight_value,
DENSE_RANK() OVER(ORDER BY Avg_freight_value asc) AS rnk
FROM CTE
ORDER BY rnk
)
WHERE rnk<=5
```

Note: Using limit and dense rank case gives same result as per our data. Which we could see the output.
Output:
Top 5 states with highest average freight value

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	customer_state	Avg_freight_value
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

Top 5 states with lowest average freight value

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

Row	customer_state	Avg_freight_value
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

Insights:
Top 5 states with highest average freight value are having the average freight value range between 39 and 43.
Top 5 states with lowest average freight value are having the average freight value range between 15 and 22.
Longest distance and no transport facility cities in that states may be reason for more average delivery charges. And some other factors also considered such as state tax and state government licences etc.

Recommendations: If possible, arrange own vehicles to increases transport facility which the states have highest average freight value. And mean while keep increase the sales in lowest average freight value states.

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

Query:
i)Top 5 states with highest average delivery time

```
WITH CTE AS
(
```

```sql
SELECT customer_state, AVG(DATE_DIFF(EXTRACT(DATE FROM
order_delivered_customer_date), EXTRACT(DATE FROM order_purchase_timestamp), DAY))
AS avg_time_to_deliver
FROM target_e_commerce.customers c JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)

#Top 5 states with highest average delivery time
SELECT customer_state, avg_time_to_deliver
FROM CTE
ORDER BY avg_time_to_deliver desc
LIMIT 5

Using Dense rank, Here we get top states with top 5 highest average delivery time
WITH CTE AS
(
SELECT customer_state, AVG(DATE_DIFF(EXTRACT(DATE FROM
order_delivered_customer_date), EXTRACT(DATE FROM order_purchase_timestamp), DAY))
AS avg_time_to_deliver
FROM target_e_commerce.customers c JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)

SELECT customer_state, avg_time_to_deliver
FROM
(
SELECT customer_state, avg_time_to_deliver,
DENSE_RANK() OVER(ORDER BY avg_time_to_deliver desc) AS rnk
FROM CTE
ORDER BY rnk
)
WHERE rnk<=5


ii)Top 5 states with lowest average delivery time
WITH CTE AS
(
SELECT customer_state, AVG(DATE_DIFF(EXTRACT(DATE FROM
order_delivered_customer_date), EXTRACT(DATE FROM order_purchase_timestamp), DAY))
AS avg_time_to_deliver
FROM target_e_commerce.customers c JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)

#Top 5 states with lowest average delivery time

SELECT customer_state, time_to_deliver
FROM CTE
ORDER BY time_to_deliver asc
LIMIT 5

Using Dense rank, Here we get top states with top 5 lowest average delivery time
WITH CTE AS
(
SELECT customer_state, AVG(DATE_DIFF(EXTRACT(DATE FROM
order_delivered_customer_date), EXTRACT(DATE FROM order_purchase_timestamp), DAY))
AS avg_time_to_deliver
```

```
FROM target_e_commerce.customers c JOIN target_e_commerce.orders o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
)

SELECT customer_state, avg_time_to_deliver
FROM
(
SELECT customer_state, avg_time_to_deliver,
DENSE_RANK() OVER(ORDER BY avg_time_to_deliver asc) AS rnk
FROM CTE
ORDER BY rnk
)
WHERE rnk<=5
```

Output:

Top 5 states with highest average delivery time

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	customer_state	avg_time_to_deliver
1	RR	29.34146341463...
2	AP	27.17910447761...
3	AM	26.35862068965...
4	AL	24.50125944584...
5	PA	23.72515856236...

Top 5 states with lowest average delivery time

Query results

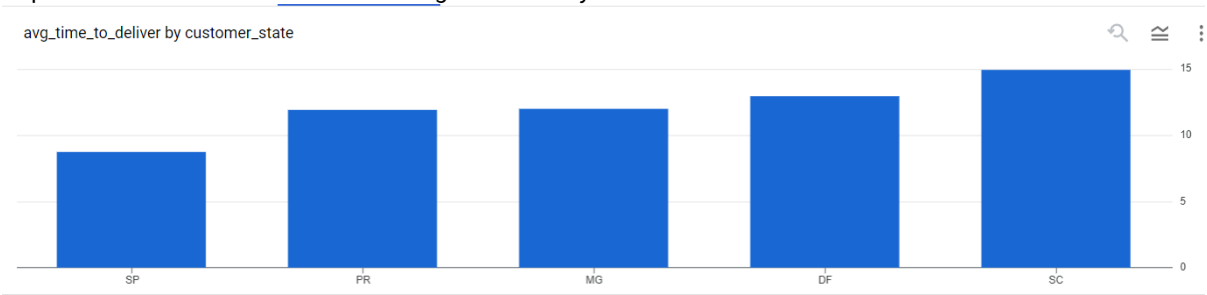
JOB INFORMATION | RESULTS | CHART PREVIEW

Row	customer_state	avg_time_to_deliver
1	SP	8.700530929744...
2	PR	11.93804590696...
3	MG	11.94654337296...
4	DF	12.89903846153...
5	SC	14.90752748801...

Insights:

Top 5 states with highest average delivery time.

Top 5 states with lowest average delivery time



Recommendations: Make sure to reduce delivery time by increasing employees and providing transport facility. Maintain on time delivery and get good appreciation and satisfaction from the customer.

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.
Query:

```
WITH CTE AS
(
SELECT
customer_state,
AVG(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)
*100/
DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp,DAY))/COUNT(o.orde
r_id) as fast_state_in_percentile

FROM target_e_commerce.orders o JOIN target_e_commerce.customers c
ON c.customer_id = o.customer_id
GROUP BY customer_state
)

SELECT customer_state, fast_state_in_percentile
FROM
(
SELECT customer_state, fast_state_in_percentile,
DENSE_RANK() OVER(ORDER BY fast_state_in_percentile desc) AS rnk
FROM CTE
)
WHERE rnk<=5
ORDER BY fast_state_in_percentile desc
```

Output:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	customer_state	fast_state_in_percen
1	SP	49.72756209989...
2	PR	48.32995278881...
3	MG	48.147770312307
4	AC	47.11527037034...
5	RO	46.93413438056...

Insights: Top 5 fastest order deliver states are SP,PR,MG,RO and AC.

Recommendations: Same as above – Try to focus more on these states and increase sales then get more profits.

Analysis based on the payments:

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

Query: Not included zero no.of records payment type records.

```
SELECT payment_type, EXTRACT(YEAR FROM order_purchase_timestamp) AS
year,EXTRACT(MONTH FROM order_purchase_timestamp) AS month, COUNT(o.order_id) as
No_of_orders
FROM target_e_commerce.orders o
JOIN target_e_commerce.payments p ON o.order_id = p.order_id
GROUP BY year,month, p.payment_type
ORDER BY p.payment_type,year, month
```

Output:

Query results

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

Row	payment_type	year	month	No_of_orders
1	UPI	2016	10	63
2	UPI	2017	1	197
3	UPI	2017	2	398
4	UPI	2017	3	590
5	UPI	2017	4	496
6	UPI	2017	5	772
7	UPI	2017	6	707
8	UPI	2017	7	845
9	UPI	2017	8	938
10	UPI	2017	9	903
11	UPI	2017	10	993
12	UPI	2017	11	1509

Included payment type with zero orders on that year and month records.

```
WITH CTE AS
(
SELECT payment_type, EXTRACT(YEAR FROM order_purchase_timestamp) AS
year,EXTRACT(MONTH FROM order_purchase_timestamp) AS month, COUNT(o.order_id) as
No_of_orders
FROM target_e_commerce.orders o
```

```
JOIN target_e_commerce.payments p ON o.order_id = p.order_id
GROUP BY year,month, p.payment_type
ORDER BY p.payment_type,year, month
),

CTE2 AS
(
  SELECT distinct year, month, payment_type
  FROM
  (
  (SELECT year,month, ROW_NUMBER() OVER() as rn FROM CTE GROUP BY year,month) AS t1
CROSS JOIN #selects 12 months and assign row number
  (SELECT payment_type, ROW_NUMBER() OVER() as rn FROM target_e_commerce.payments
GROUP BY payment_type ) AS t2 #selects all different payment types and assign row
number
  )
)

Select CTE2.payment_type,CTE2.year, CTE2.month, IFNULL(CTE.No_of_orders,0) from
CTE2 LEFT JOIN CTE on CTE.month=CTE2.month AND CTE.payment_type =CTE2.payment_type
ORDER BY CTE2.payment_type desc,CTE2.year, CTE2.month
```

Note: Both queries will give same insights, zero count doesn't affect anything only useful in show case.
Output: From our data only difference in "Not defined" payment type .

Query results

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

Row	payment_type	year	month	f0_
51	not_defined	2016	9	1
52	not_defined	2016	10	0
53	not_defined	2016	12	0
54	not_defined	2017	1	0
55	not_defined	2017	2	0
56	not_defined	2017	3	0
57	not_defined	2017	4	0
58	not_defined	2017	5	0
59	not_defined	2017	6	0
60	not_defined	2017	7	0
61	not_defined	2017	8	2
62	not_defined	2017	9	1

Insights:
We can see most of the payments done through credit_cards. It seems to be providing offers for credit cards. And less payments through debit card(If ignores not_defined case)

Recommendation: Not charge any amount for transactions and increase more offers on some payment types.

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

```
SELECT payment_installments,count(o.order_id) AS No_of_paid_orders
FROM target_e_commerce.orders o
JOIN target_e_commerce.payments p ON o.order_id = p.order_id
WHERE order_status <> 'canceled'
GROUP BY payment_installments
ORDER BY No_of_paid_orders desc
```

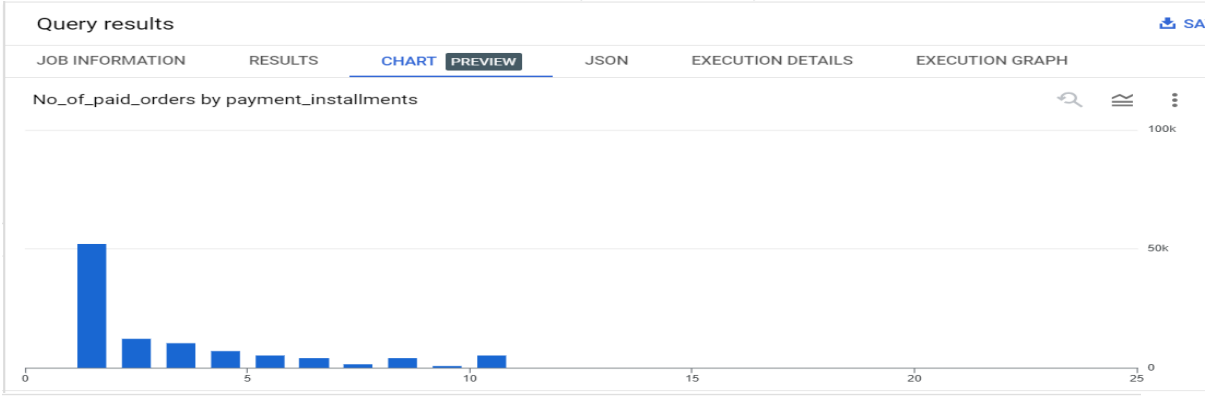
Output:

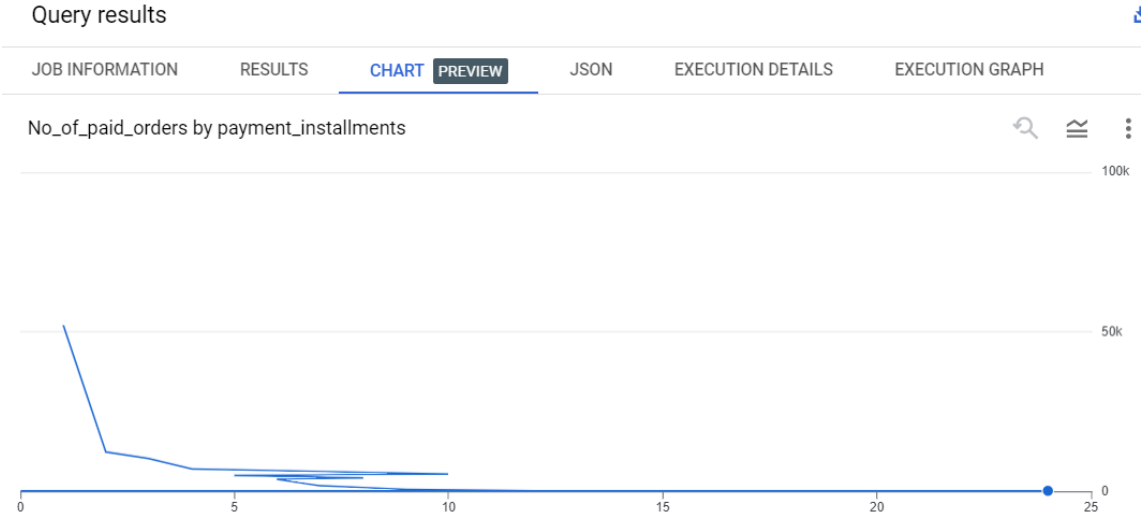
Query results

JOB INFORMATION | RESULTS | CHART PREVIEW

Row	payment_installment	No_of_paid_orders
1	1	52184
2	2	12353
3	3	10392
4	4	7056
5	10	5292
6	5	5209
7	8	4239
8	6	3898
9	7	1620
10	9	638
11	12	133
12	15	74

Insights: More orders payment done by single instalment and very few orders payment done by 23 and 24 instalments. From 11 instalment onwards no of orders are very less compared to previous ones that's why not clearly visible in bar below chart.





Recommendations: NA