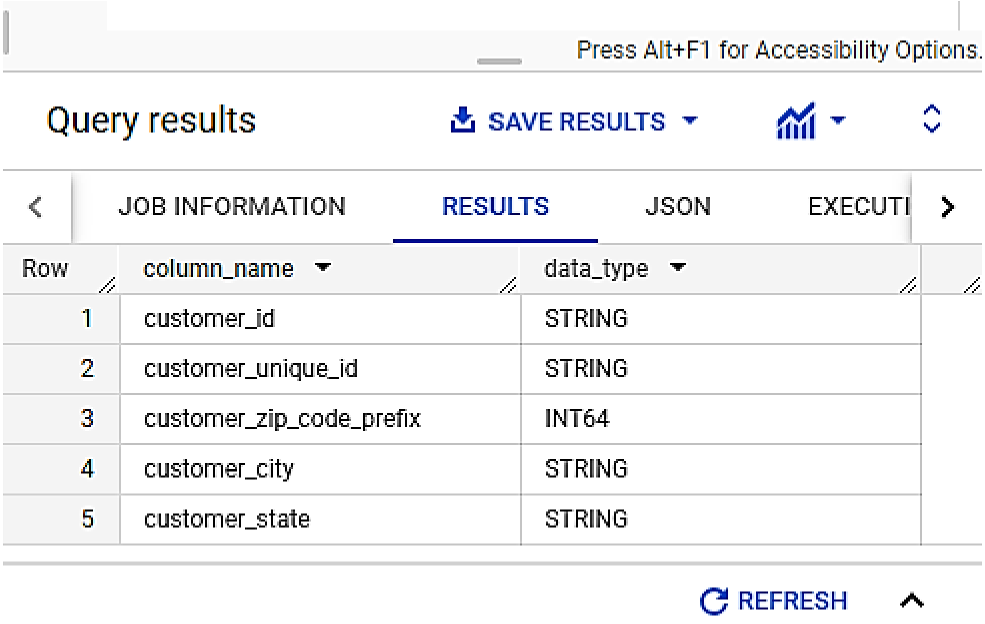
Business Case: Target SQL Solution

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

Query :-
```
SELECT column_name, data_type
FROM `linear-passage-386211.customers.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers table';
```



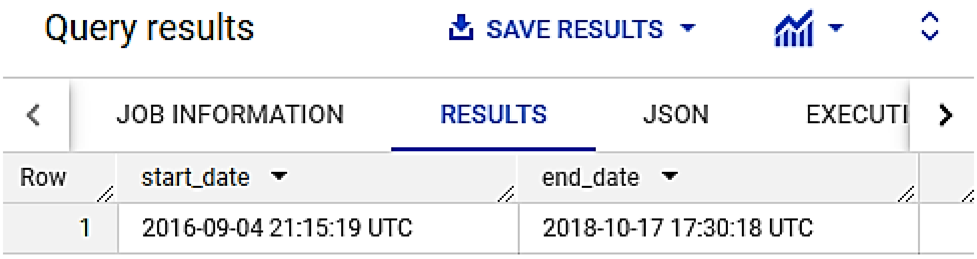
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

INSIGHTS:- The query retrieves the data types of all columns within the "customers" table, providing valuable information about the structure and format of the data in that table.

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

Query:-
```
SELECT MIN(order_purchase_timestamp) AS start_date,
       MAX (order_purchase_timestamp) AS end_date

       FROM `linear-passage-386211.orders.orders_ta
```



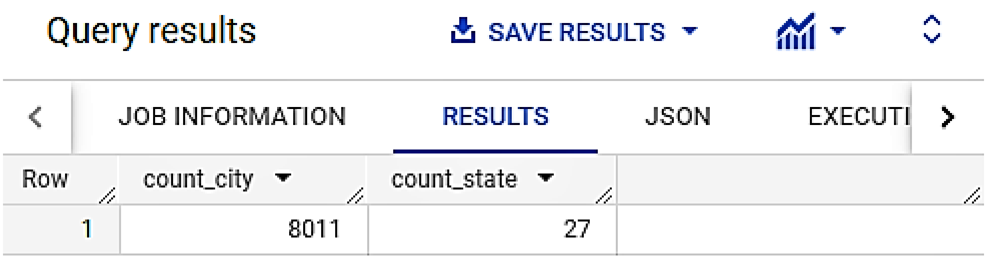
Row	start_date	end_date
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

INSIGHTS:- This query provides the time range during which orders were placed, allowing for a quick overview of the order placement timeline.

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

Query:-

```
SELECT COUNT(DISTINCT geolocation_city) AS count_city,
       COUNT (DISTINCT geolocation_state) AS count_state
       FROM `linear-passage-386211.geo_location.geo_location_table` ;
```



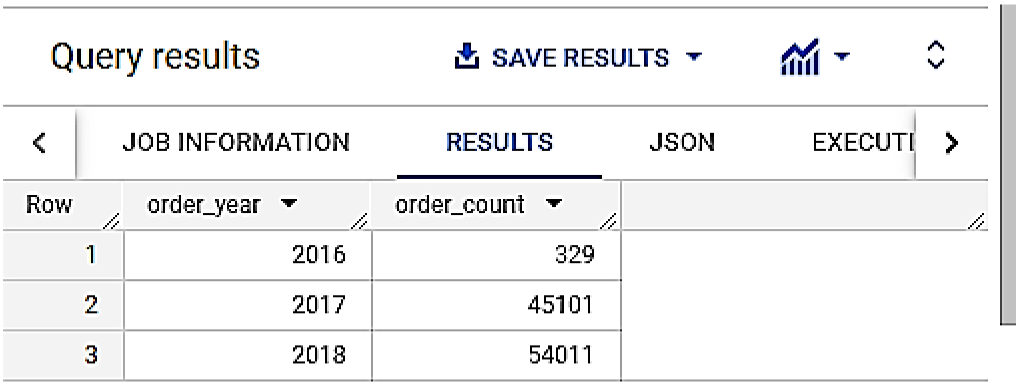
Row	count_city	count_state
1	8011	27

INSIGHTS:- This query provides the counts of unique cities and states where customers placed orders during the given period.

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

Query:-

```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        COUNT(order_id) AS order_count
        FROM `linear-passage-386211.orders.orders_table`
        GROUP BY order_year
        ORDER BY order_year ASC;
```

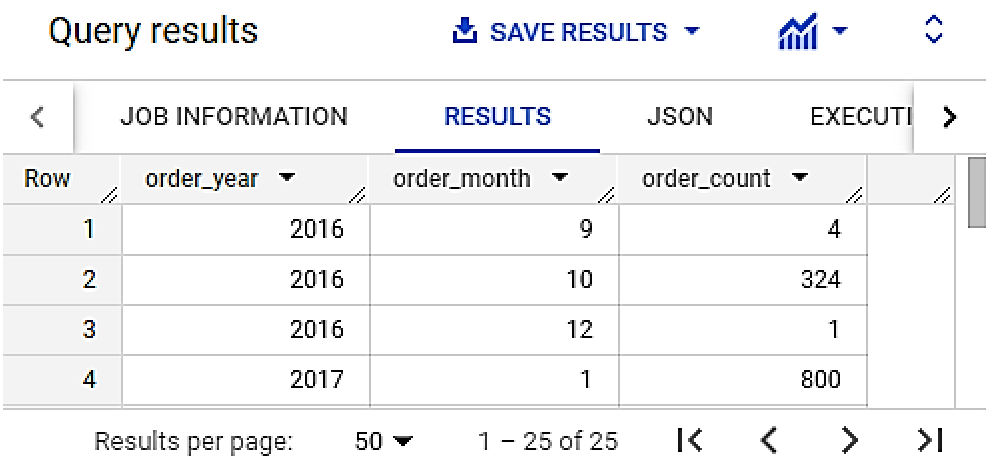


Row	order_year	order_count
1	2016	329
2	2017	45101
3	2018	54011

INSIGHTS:- This query provides a year-wise breakdown of order counts, enabling the observation of any trends or changes in order volume over the past few years.

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

Query:-
```
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
       EXTRACT (MONTH FROM order_purchase_timestamp) AS order_month,
       COUNT(order_id) AS order_count
       FROM `linear-passage-386211.orders.orders_table`
       GROUP BY order_year,order_month
       ORDER BY order_year, order_month ASC;
```



Row	order_year	order_month	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800

INSIGHTS:- This query reveals potential monthly seasonality patterns in order placement, offering a year-by-year breakdown of order counts, aiding in the analysis of order trends over time.

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

0-6 hrs : Dawn

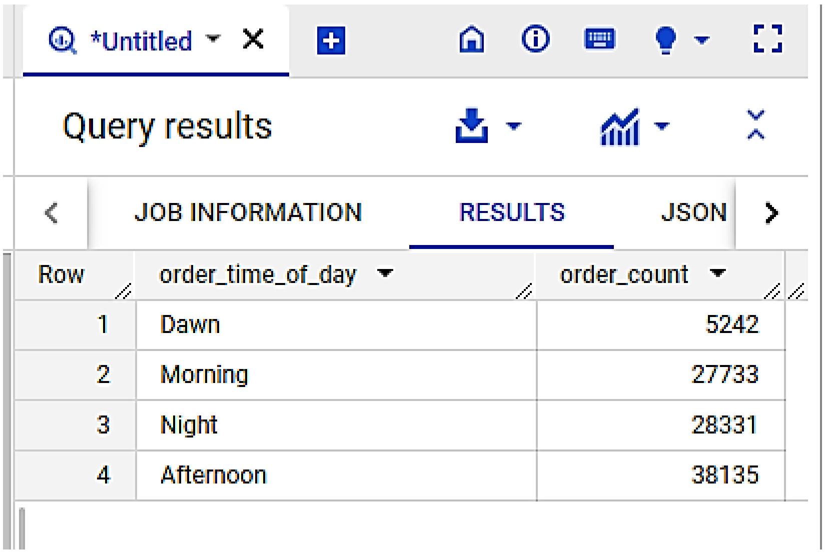
7-12 hrs : Mornings

13-18 hrs : Afternoon

19-23 hrs : Night

Query :–

```
SELECT
  CASE
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
  WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN
'Afternoon'
 WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
 END AS order_time_of_day,
 COUNT(order_id) AS order_count
 FROM `linear-passage-386211.orders.orders_table`
 GROUP BY order_time_of_day
 ORDER BY order_count ASC;
```



Row	order_time_of_day	order_count
1	Dawn	5242
2	Morning	27733
3	Night	28331
4	Afternoon	38135

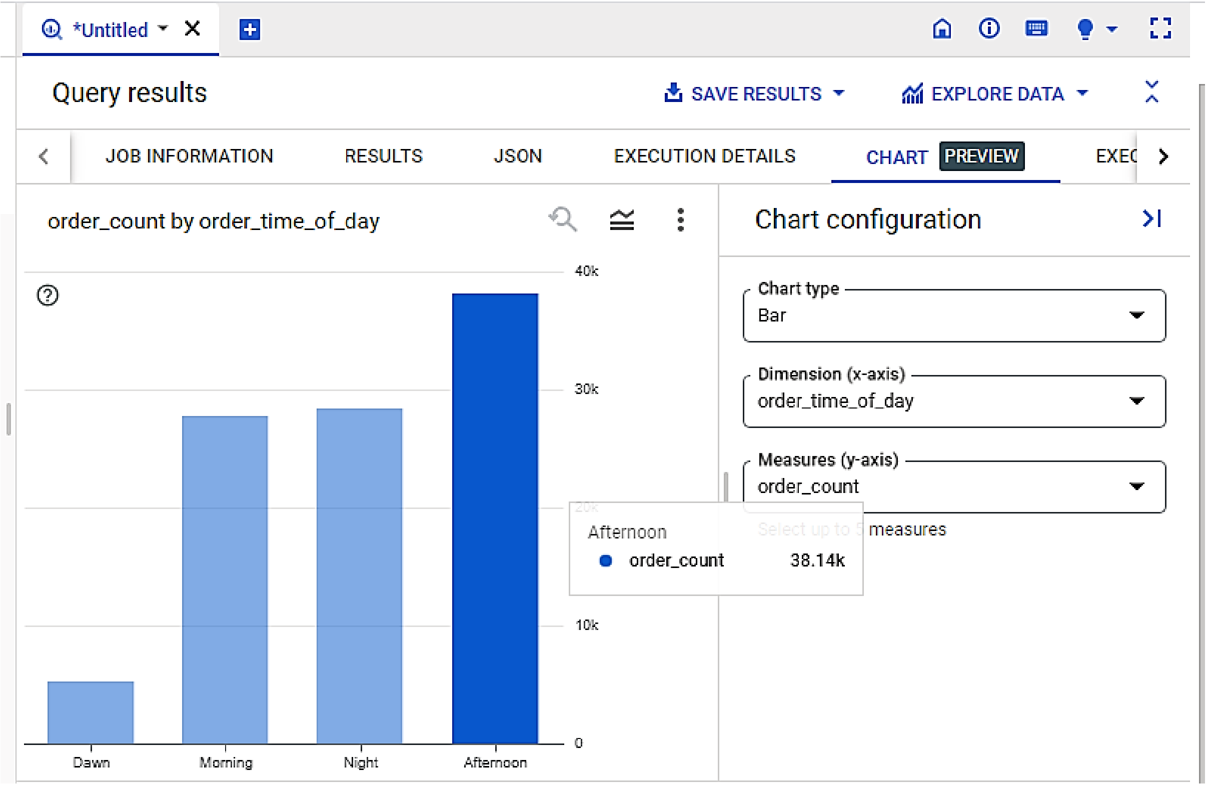


Chart type: Bar

Dimension (x-axis): order_time_of_day

Measures (y-axis): order_count

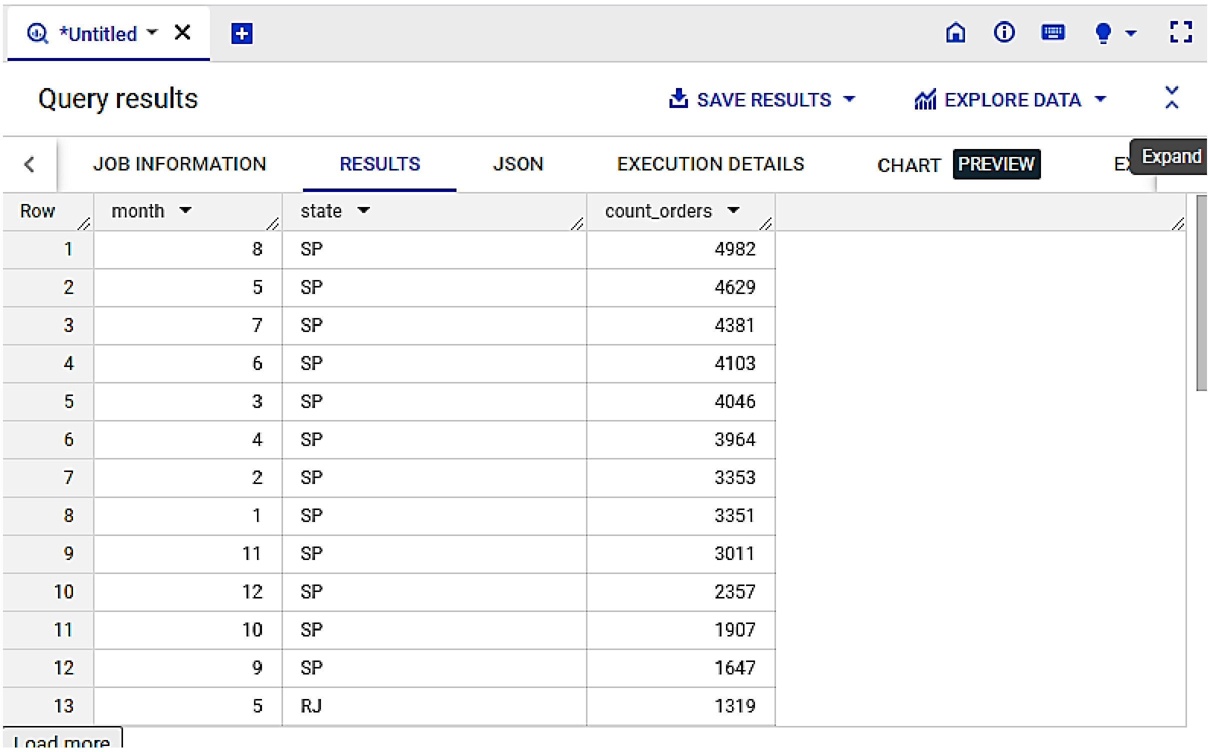
INSIGHTS :- This query segments Brazilian customer order placement times into four categories: Dawn, Morning, Afternoon, and Night, offering insights into their ordering habits throughout the day.

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

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

Query :- `SELECT t.month,t.state,t.count_orders`

```
FROM (SELECT
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
g.geolocation_state AS state,
COUNT(DISTINCT o.order_id) AS count_orders
FROM `linear-passage-386211.orders.orders_table` AS o
JOIN `linear-passage-386211.customers.customers table` AS c
ON c.customer_id = o.customer_id
JOIN `linear-passage-386211.geo_location.geo_location_table` AS g
ON g.geolocation_zip_code_prefix = c.customer_zip_code_prefix
GROUP BY month,state) AS t
ORDER BY t.count_orders DESC;
```



Row	month	state	count_orders
1	8	SP	4982
2	5	SP	4629
3	7	SP	4381
4	6	SP	4103
5	3	SP	4046
6	4	SP	3964
7	2	SP	3353
8	1	SP	3351
9	11	SP	3011
10	12	SP	2357
11	10	SP	1907
12	9	SP	1647
13	5	RJ	1319

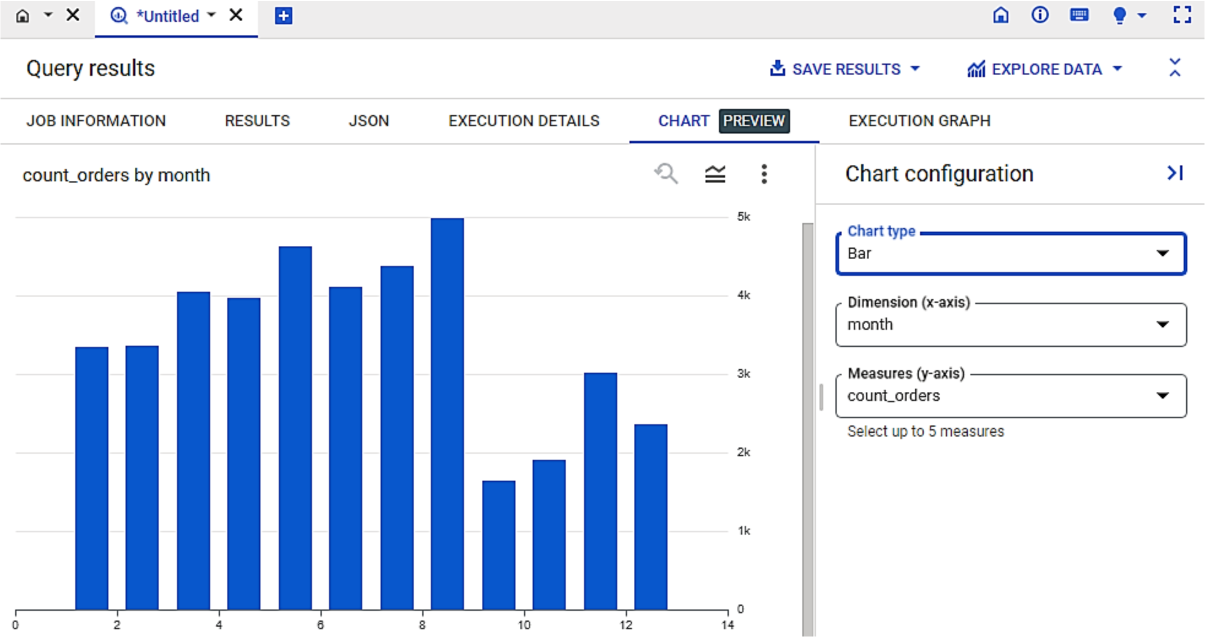


Chart type: Bar

Dimension (x-axis): month

Measures (y-axis): count_orders

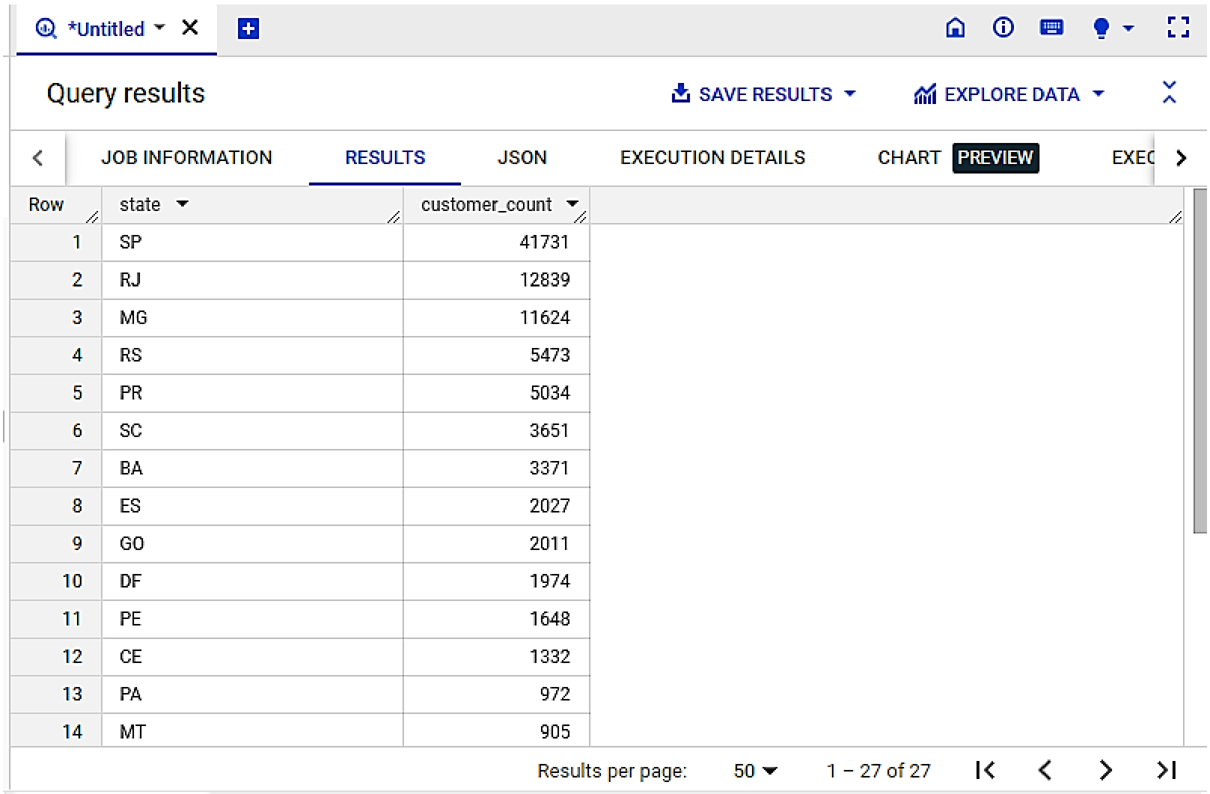
Select up to 5 measures

INSIGHTS :- This query, accompanied by a bar chart, illustrates the monthly order patterns across different states, enabling a visual comparison of order counts for deeper insights.

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

Query :- `SELECT g.geolocation_state AS state,`

```
COUNT(DISTINCT c.customer_id) AS customer_count
FROM`linear-passage-386211.customers.customers table` AS c
JOIN `linear-passage-386211.geo_location.geo_location_table` AS g
ON g.geolocation_zip_code_prefix = c.customer_zip_code_prefix
BY state
ORDER BY customer_count DESC;
```



Row	state	customer_count
1	SP	41731
2	RJ	12839
3	MG	11624
4	RS	5473
5	PR	5034
6	SC	3651
7	BA	3371
8	ES	2027
9	GO	2011
10	DF	1974
11	PE	1648
12	CE	1332
13	PA	972
14	MT	905

Results per page: 50 1 – 27 of 27

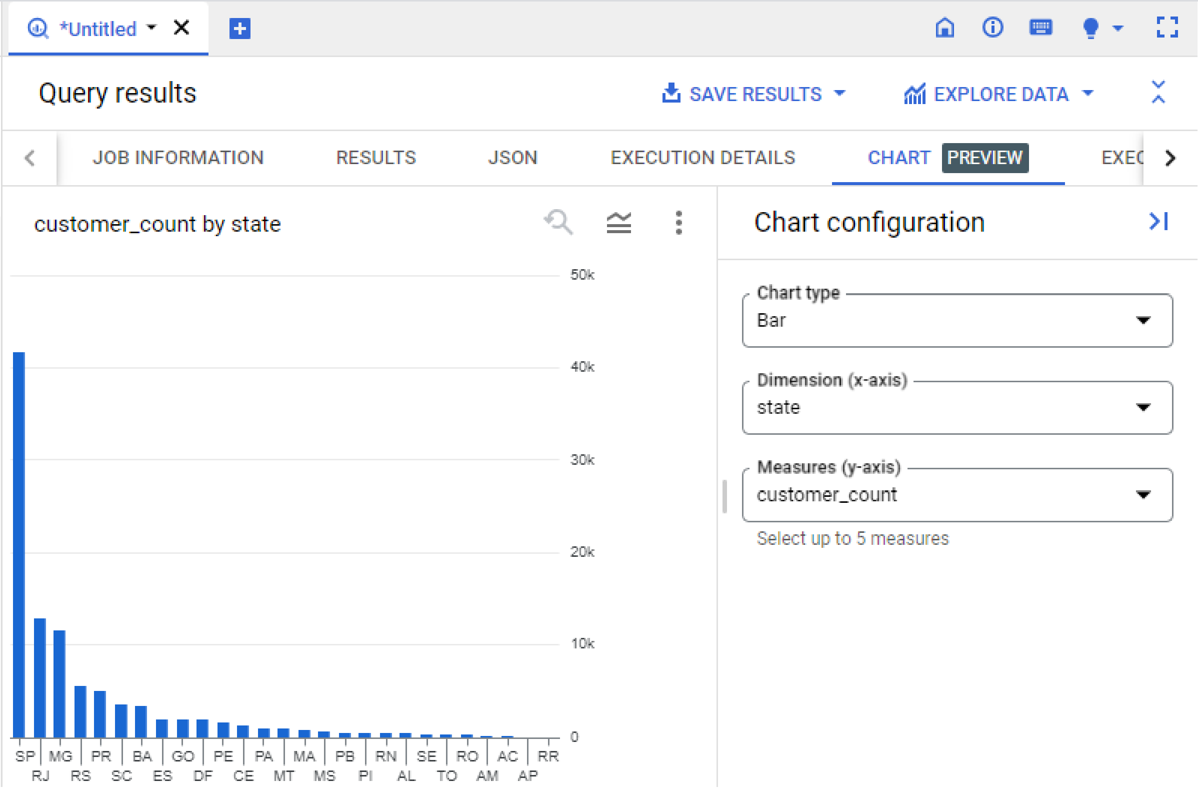


Chart configuration

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): customer_count

Select up to 5 measures

INSIGHTS :- This query, along with an accompanying bar graph, illustrates the distribution of customers across states, allowing easy comparison of customer counts in each state.

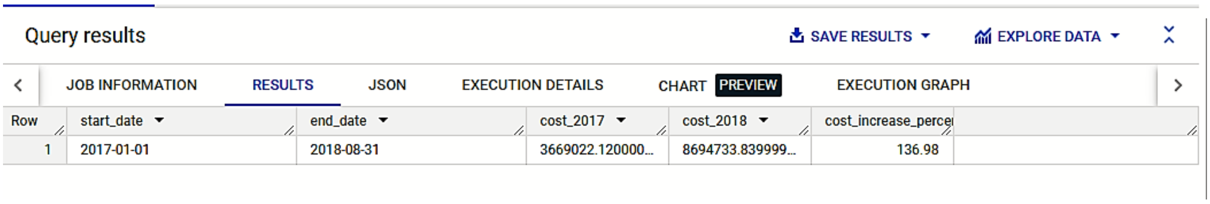
4.

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.

Query :-
```
SELECT

        '2017-01-01' AS start_date,
        '2018-08-31' AS end_date,
         SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 THEN
p.payment_value ELSE 0 END) AS cost_2017,
        SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 THEN
p.payment_value ELSE 0 END) AS cost_2018,
        IFNULL(ROUND(
        (SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 THEN
p.payment_value ELSE 0 END) -
        SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 THEN
p.payment_value ELSE 0 END)) /
        NULLIF(SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 THEN
p.payment_value ELSE 0 END), 0) * 100, 2), 0) AS cost_increase_percentage
        FROM`linear-passage-386211.orders.orders_table` AS o
        JOIN`linear-passage-386211.payments.payments_table` AS p
        ON o.order_id = p.order_id
        WHERE
        EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
        AND EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018);
```



Row	start_date	end_date	cost_2017	cost_2018	cost_increase_perce
1	2017-01-01	2018-08-31	3669022.120000...	8694733.839999...	136.98

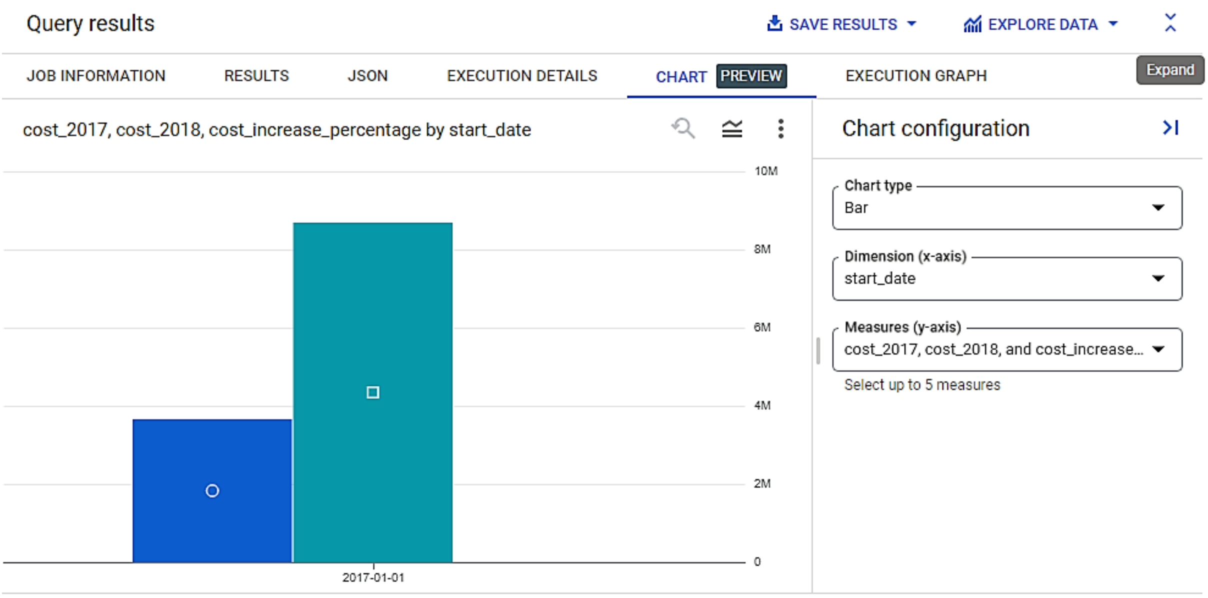


Chart type: Bar

Dimension (x-axis): start_date

Measures (y-axis): cost_2017, cost_2018, and cost_increase...

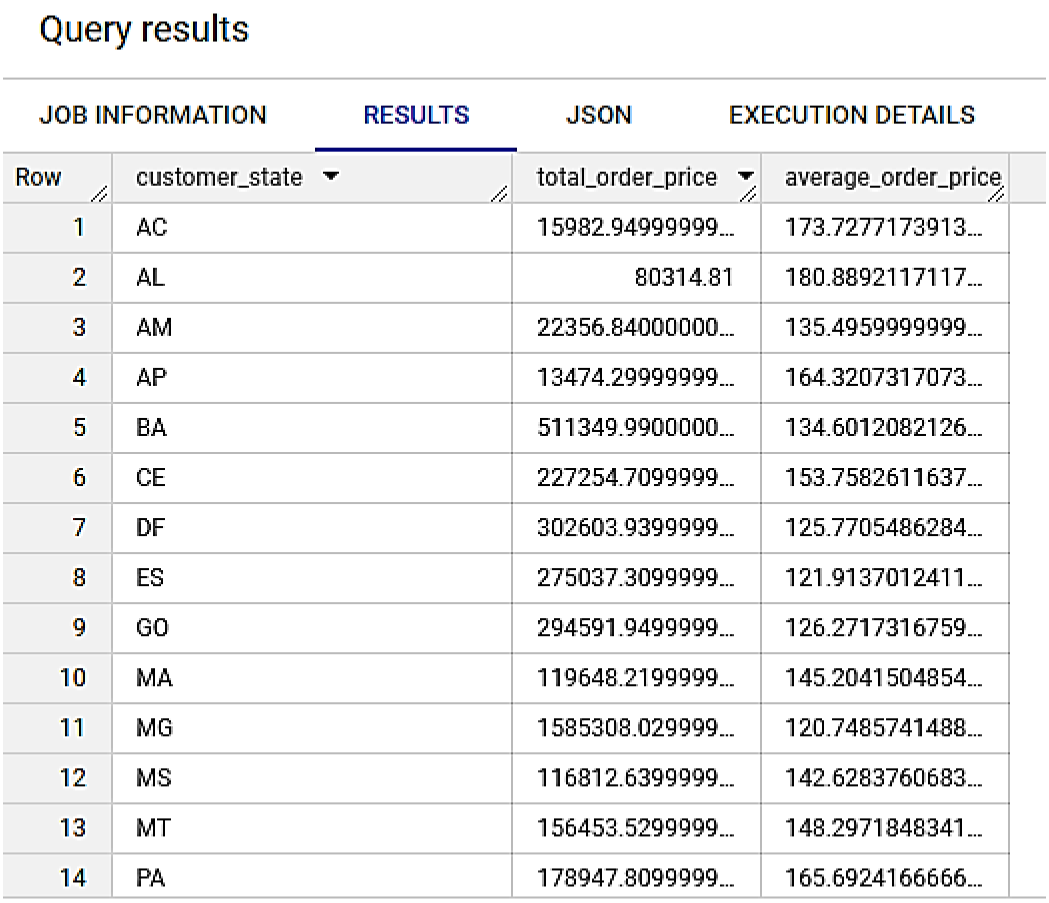
Select up to 5 measures

INSIGHTS :- This analysis, supported by a graph, reveals the percentage increase in order costs from January to August, comparing 2017 to 2018.

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

Query :- `SELECT c.customer_state,`

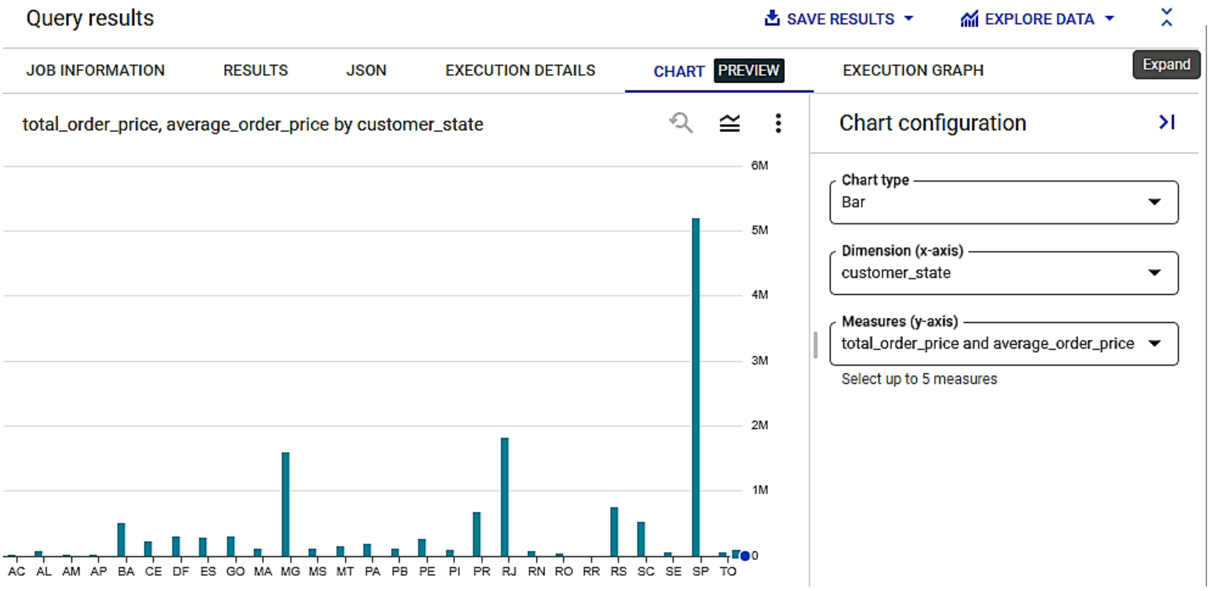
```
SUM(oi.price) AS total_order_price,
AVG(oi.price) AS average_order_price
FROM `linear-passage-386211.customers.customers table` AS c
JOIN `linear-passage-386211.orders.orders_table` AS o
ON c.customer_id = o.customer_id
JOIN `linear-passage-386211.order_item.order_item_table` AS oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```



Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_order_price	average_order_price
1	AC	15982.94999999...	173.7277173913...
2	AL	80314.81	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...
13	MT	156453.5299999...	148.2971848341...
14	PA	178947.8099999...	165.6924166666...



Query results

total_order_price, average_order_price by customer_state

Chart configuration

Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): total_order_price and average_order_price

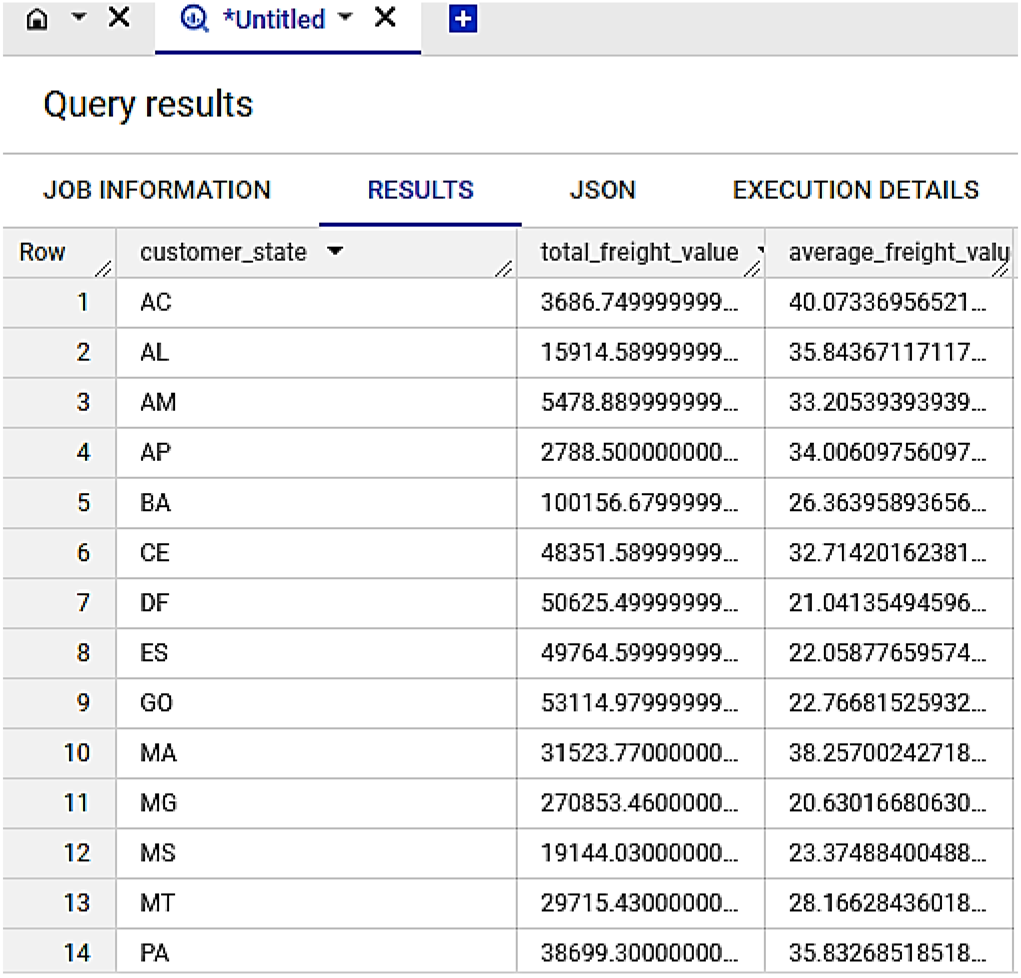
Select up to 5 measures

INSIGHTS - This query, along with a bar graph, helps analyze pricing trends across different states by calculating both total and average order prices for each state.

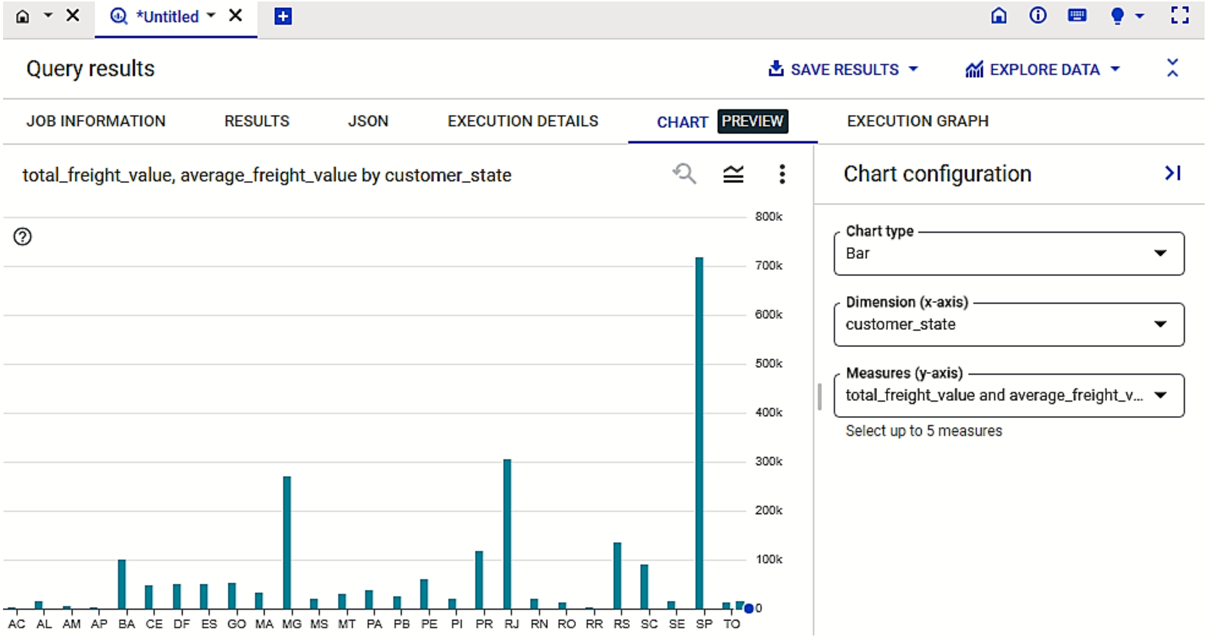
3. Calculate the Total & Average value of order freight for each state.

Query-
```
SELECT c.customer_state,

       SUM(oi.freight_value) AS total_freight_value,
       AVG(oi.freight_value) AS average_freight_value
       FROM `linear-passage-386211.customers.customers table` AS c
       JOIN `linear-passage-386211.orders.orders_table` AS o
       ON c.customer_id = o.customer_id
       JOIN `linear-passage-386211.order_item.order_item_table` AS oi
       ON o.order_id = oi.order_id
       GROUP BY c.customer_state
       ORDER BY c.customer_state;
```



Row	customer_state	total_freight_value	average_freight_valu
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...
12	MS	19144.03000000...	23.37488400488...
13	MT	29715.43000000...	28.16628436018...
14	PA	38699.30000000...	35.83268518518...



Query results

total_freight_value, average_freight_value by customer_state

Chart configuration

Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): total_freight_value and average_freight_v...

Select up to 5 measures

INSIGHTS:- This query, paired with a bar graph, calculates both the total and average order freight values for each state, allowing for a comprehensive analysis of freight expenditure patterns across regions.

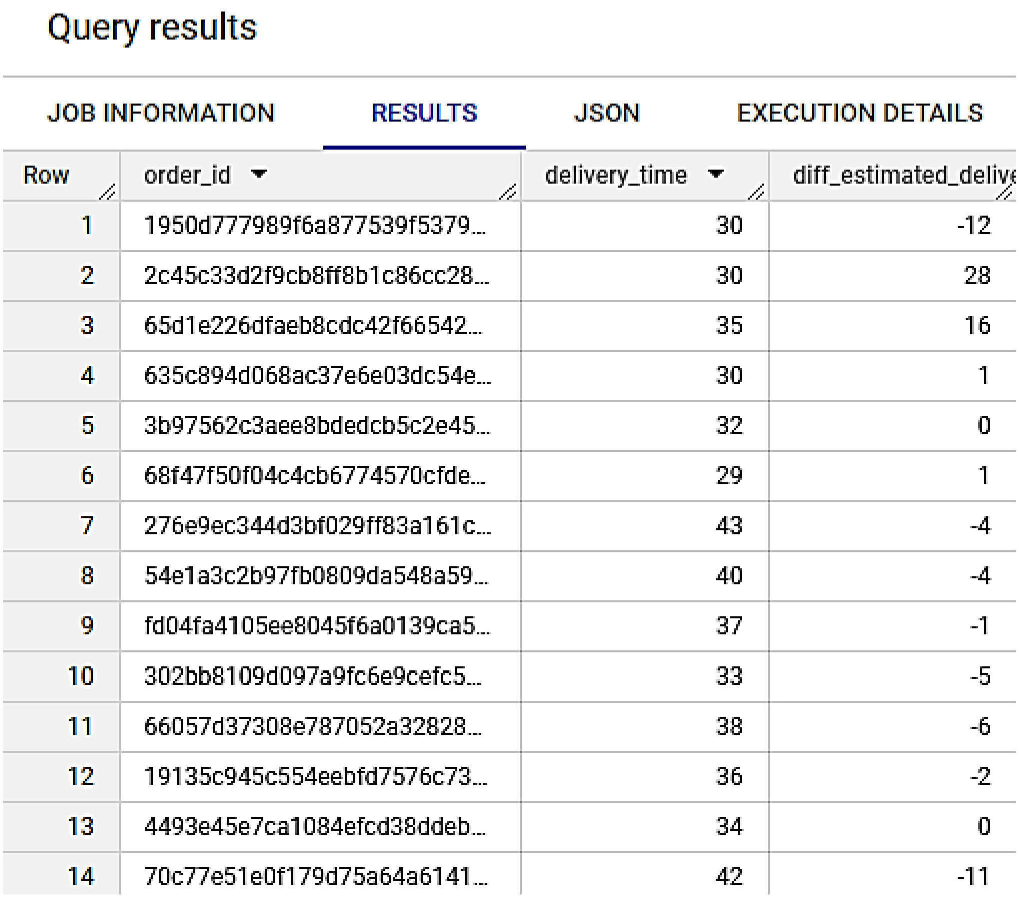
5 Analysis based on sales, freight and delivery time.

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.

Query:-

```
SELECT order_id,

       DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
delivery_time,
       DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) AS
diff_estimated_delivery
       FROM `linear-passage-386211.orders.orders_table`
```



Query results

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
11	66057d37308e787052a32828...	38	-6
12	19135c945c554eebfd7576c73...	36	-2
13	4493e45e7ca1084efcd38ddeb...	34	0
14	70c77e51e0f179d75a64a6141...	42	-11

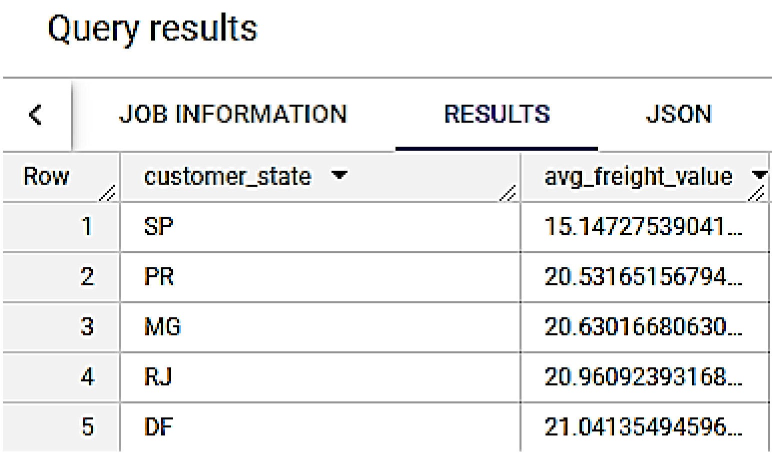
INSIGHTS:- This query calculates the delivery time for each order representing the number of days between the purchase date and delivery. It also computes the variance in days between the estimated and actual delivery dates for orders.

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

QUERY-

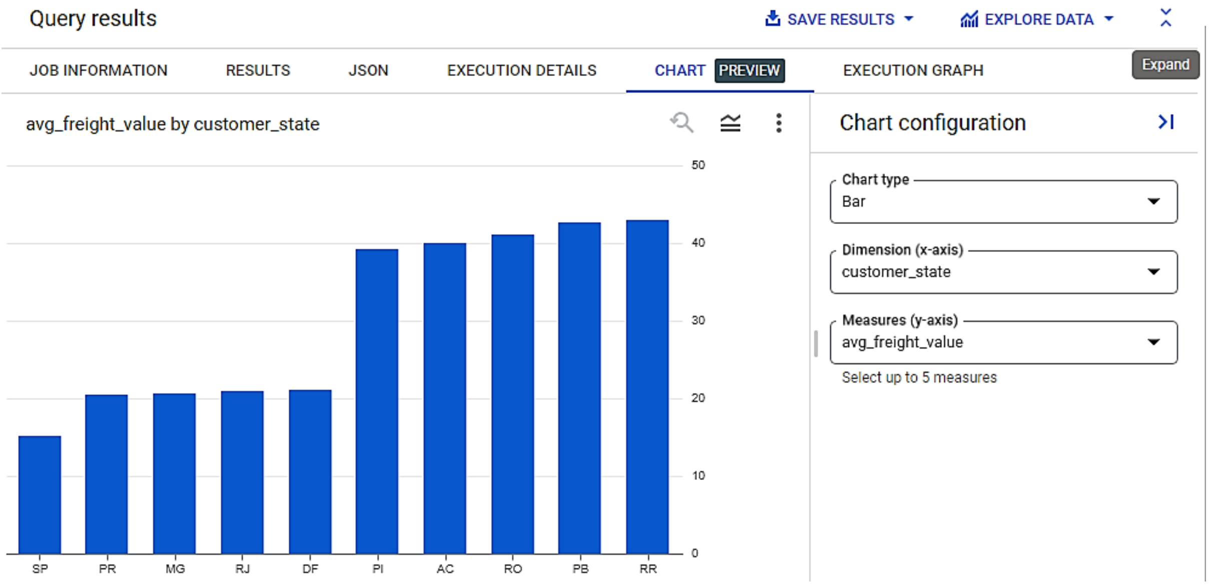
```
WITH cte AS (SELECT c.customer_state,

AVG(oi.freight_value) AS avg_freight_value
FROM `linear-passage-386211.orders.orders_table` AS o
JOIN`linear-passage-386211.customers.customers table` AS c
ON o.customer_id = c.customer_id
JOIN`linear-passage-386211.order_item.order_item_table` AS oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state)
SELECT customer_state,avg_freight_value
FROM (SELECT customer_state,avg_freight_value,
ROW_NUMBER() OVER (ORDER BY avg_freight_value DESC) AS high_rnk,
ROW_NUMBER() OVER (ORDER BY avg_freight_value ASC) AS low_rnk
FROM cte) AS ranked_data
WHERE high_rnk <= 5 OR low_rnk <= 5
ORDER BY low_rnk ASC, high_rnk ASC;
```



Query results

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...



Query results

Chart configuration

Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): avg_freight_value

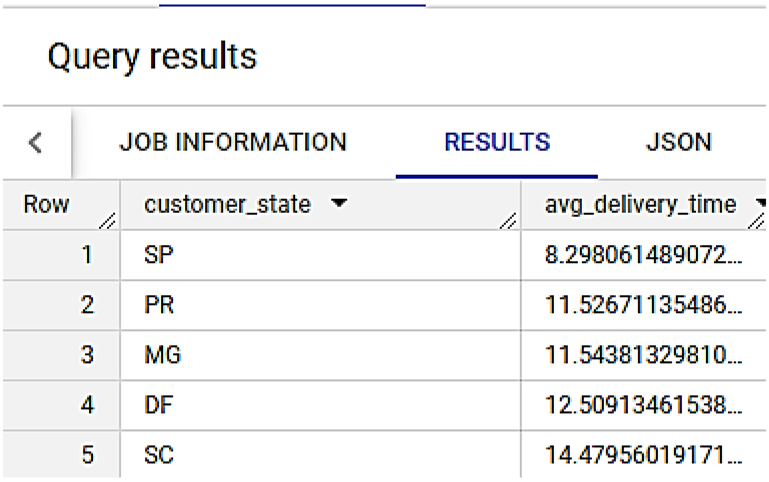
Select up to 5 measures

INSIGHTS - This query complemented by a bar graph, efficiently identifies the top 5 states with the highest and lowest average freight costs. It provides valuable insights into regional disparities in shipping expenses.

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

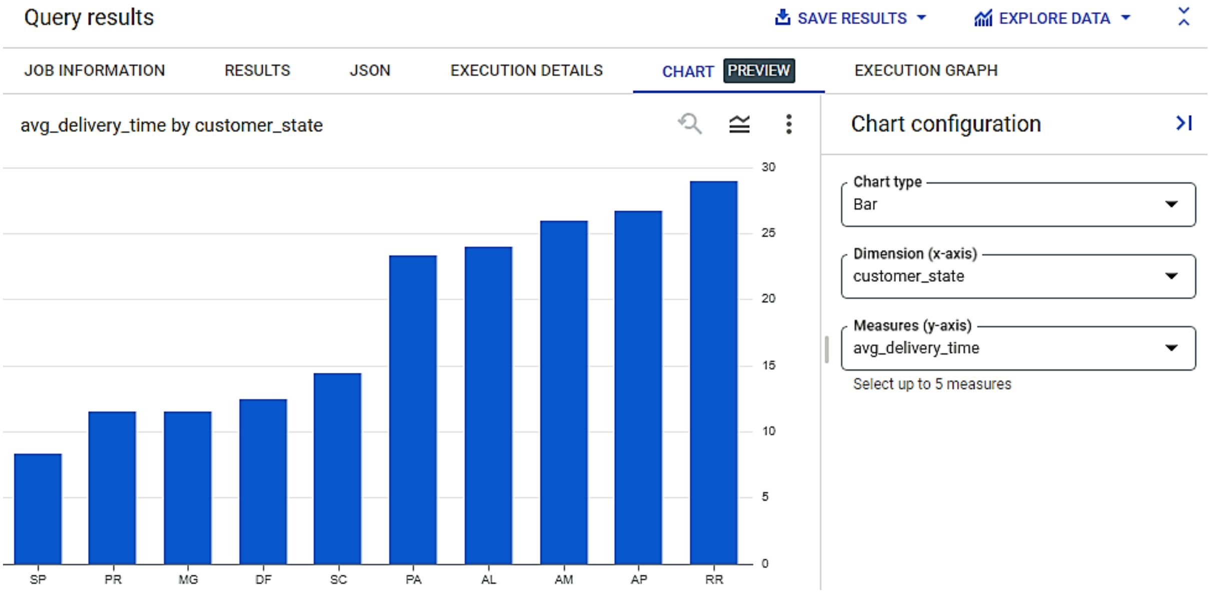
Query-
```
WITH cte AS (

       SELECT c.customer_state,
       AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, DAY)) AS
avg_delivery_time
       FROM
       `linear-passage-386211.orders.orders_table` AS o
       JOIN`linear-passage-386211.customers.customers table` AS c
       ON o.customer_id = c.customer_id
       GROUP BY c.customer_state)
       SELECT customer_state,avg_delivery_time
       FROM (SELECT customer_state,avg_delivery_time,
       ROW_NUMBER() OVER (ORDER BY avg_delivery_time DESC) AS high_rnk,
       ROW_NUMBER() OVER (ORDER BY avg_delivery_time ASC) AS low_rnk
       FROM cte) AS ranked_data
       WHERE high_rnk <= 5 OR low_rnk <= 5
       ORDER BY low_rnk ASC, high_rnk ASC;
```



Query results

Row	customer_state	avg_delivery_time
1	SP	8.298061489072...
2	PR	11.52671135486...
3	MG	11.54381329810...
4	DF	12.50913461538...
5	SC	14.47956019171...



Query results

Chart configuration

Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): avg_delivery_time

Select up to 5 measures

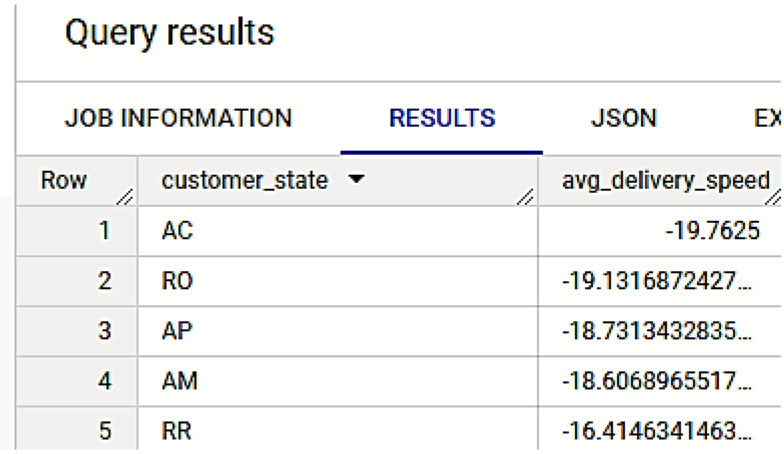
INSIGHTS - This query, along with a bar graph, effectively identifies and ranks the top 5 states with both the highest and lowest average delivery times. It offers valuable insights into regional variations in delivery speed, allowing for a clear visual comparison.

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 c.customer_state,

        AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date,
DAY)) AS avg_delivery_speed
       FROM `linear-passage-386211.orders.orders_table` AS o
       JOIN `linear-passage-386211.customers.customers table` AS c
       ON o.customer_id = c.customer_id
       GROUP BY c.customer_state)
       SELECT customer_state,avg_delivery_speed
       FROM (SELECT
       customer_state,avg_delivery_speed,
       ROW_NUMBER() OVER (ORDER BY avg_delivery_speed ASC) AS fast_rnk
       FROM
       cte) AS ranked_data
       WHERE fast_rnk <= 5
       ORDER BY fast_rnk ASC;
```



Row	customer_state	avg_delivery_speed
1	AC	-19.7625
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...

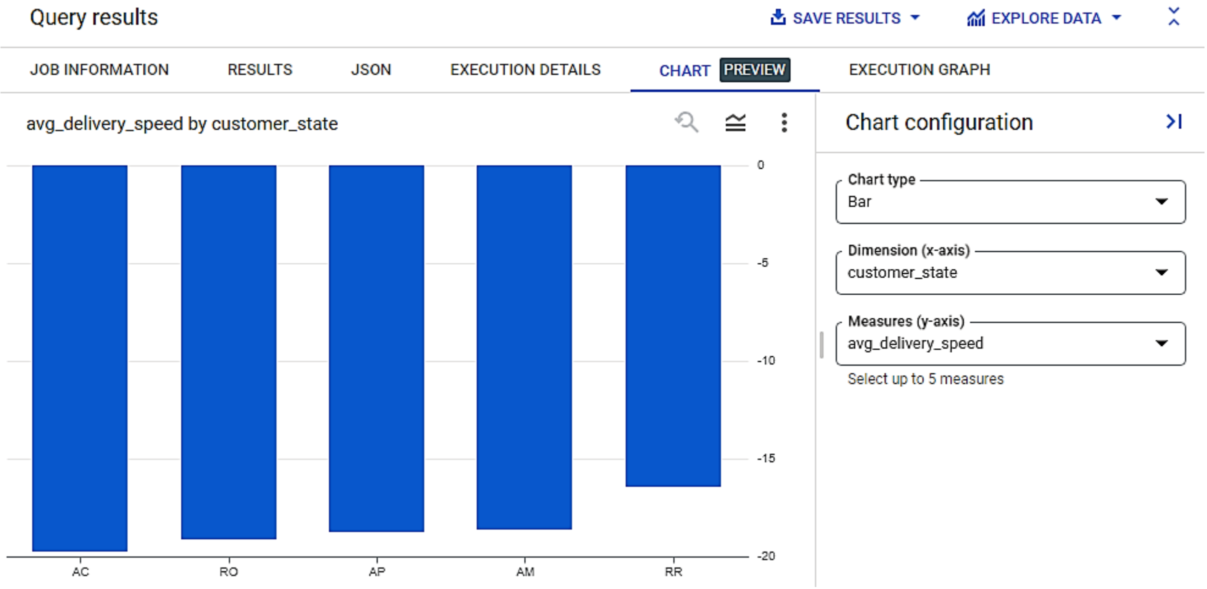


Chart type
Bar

Dimension (x-axis)
customer_state

Measures (y-axis)
avg_delivery_speed

Select up to 5 measures

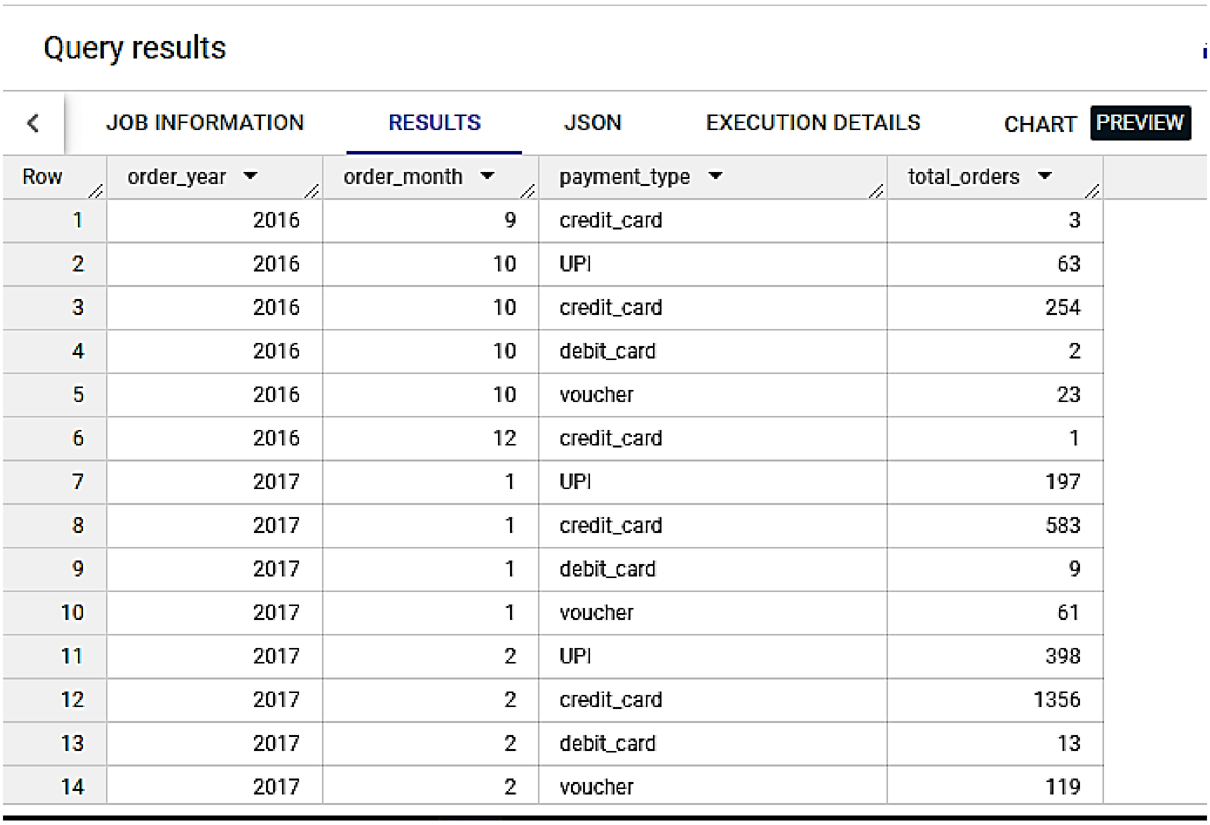
INSIGHTS - Using a bar graph, this query highlights the top 5 states with remarkably faster order deliveries than estimated. It's based on the difference between actual and estimated delivery times

6. Analysis based on the payments.

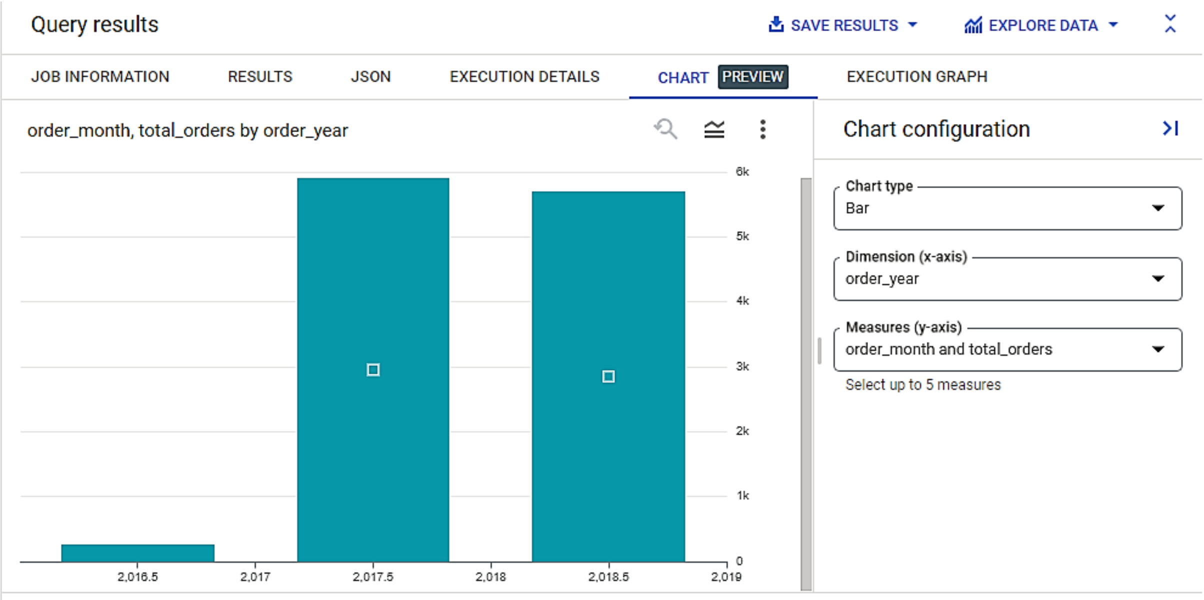
1. Find the month on month no. of orders placed using different payment types.

Query-
```
SELECT order_year,order_month,payment_type,

         SUM(order_count) AS total_orders
         FROM (SELECT
         EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
         EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
p.payment_type,
         COUNT(o.order_id) AS order_count
         FROM`linear-passage-386211.orders.orders_table` AS o
         JOIN`linear-passage-386211.payments.payments_table` AS p
         ON o.order_id = p.order_id
         GROUP BY order_year,order_month,payment_type) AS MonthlyOrders
         GROUP BY order_year,order_month,payment_type
         ORDER BY order_year,order_month,payment_type;
```



Row	order_year	order_month	payment_type	total_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	254
4	2016	10	debit_card	2
5	2016	10	voucher	23
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	583
9	2017	1	debit_card	9
10	2017	1	voucher	61
11	2017	2	UPI	398
12	2017	2	credit_card	1356
13	2017	2	debit_card	13
14	2017	2	voucher	119



order_month, total_orders by order_year

Chart type: Bar

Dimension (x-axis): order_year

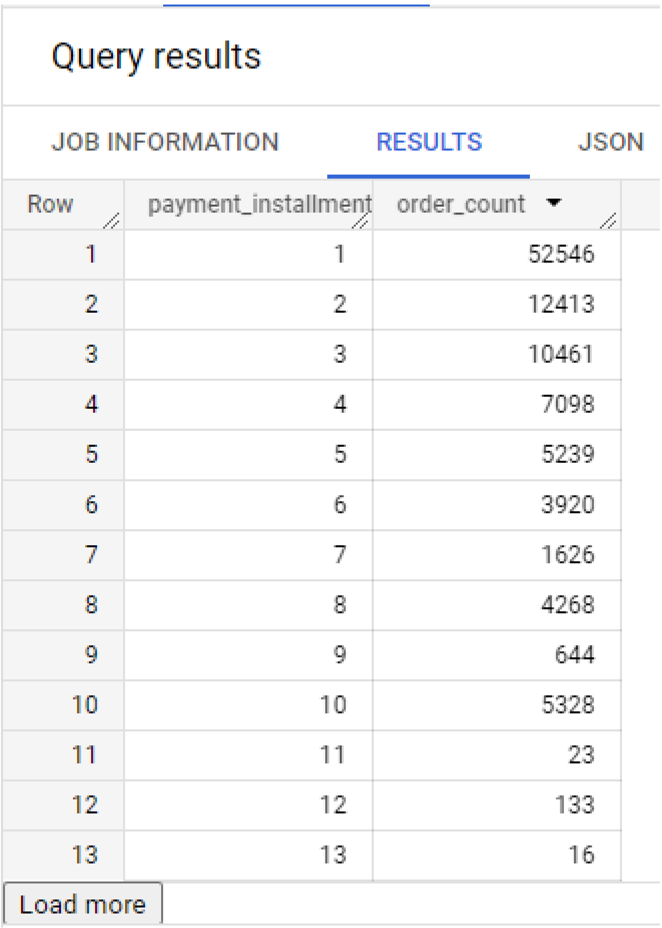
Measures (y-axis): order_month and total_orders

INSIGHTS - This query, along with a bar graph, presents the monthly breakdown of orders based on different payment types. It allows for a clear understanding of payment preferences over time

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 order_count
         FROM `linear-passage-386211.payments.payments_table`
         WHERE payment_installments > 0
         GROUP BY payment_installments
         ORDER BY payment_installments;
```



Row	payment_installment	order_count
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328
11	11	23
12	12	133
13	13	16

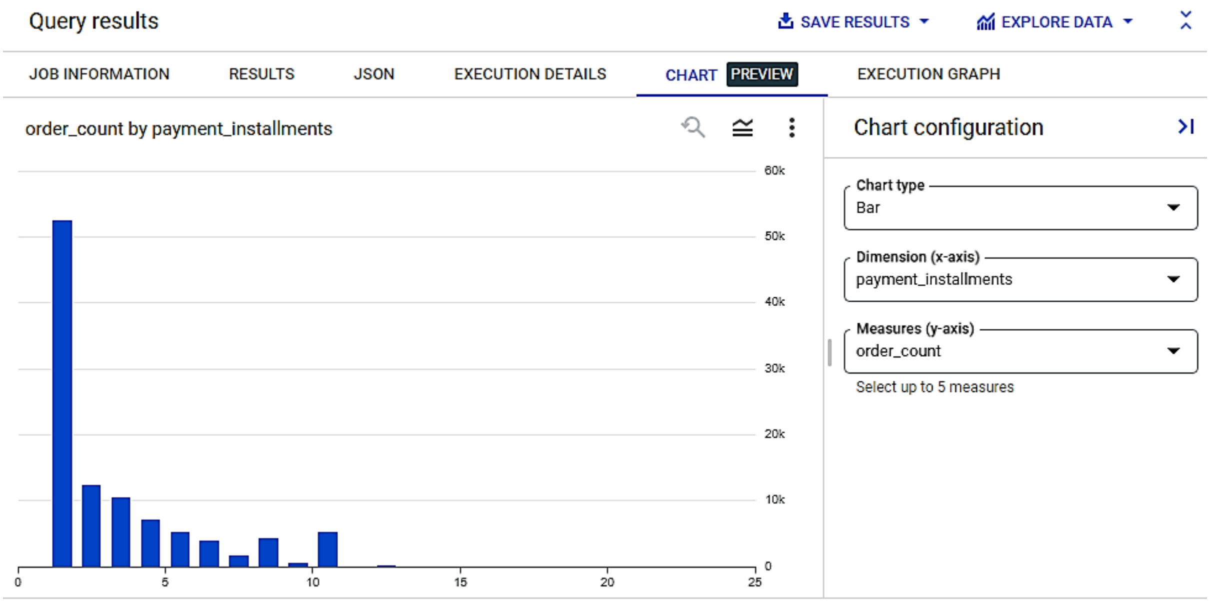


Chart type: Bar

Dimension (x-axis): payment_installments

Measures (y-axis): order_count

Select up to 5 measures

INSIGHTS- Using a bar graph, this query unveils how payment installment options impact order placement, offering insights into customer payment behavior.