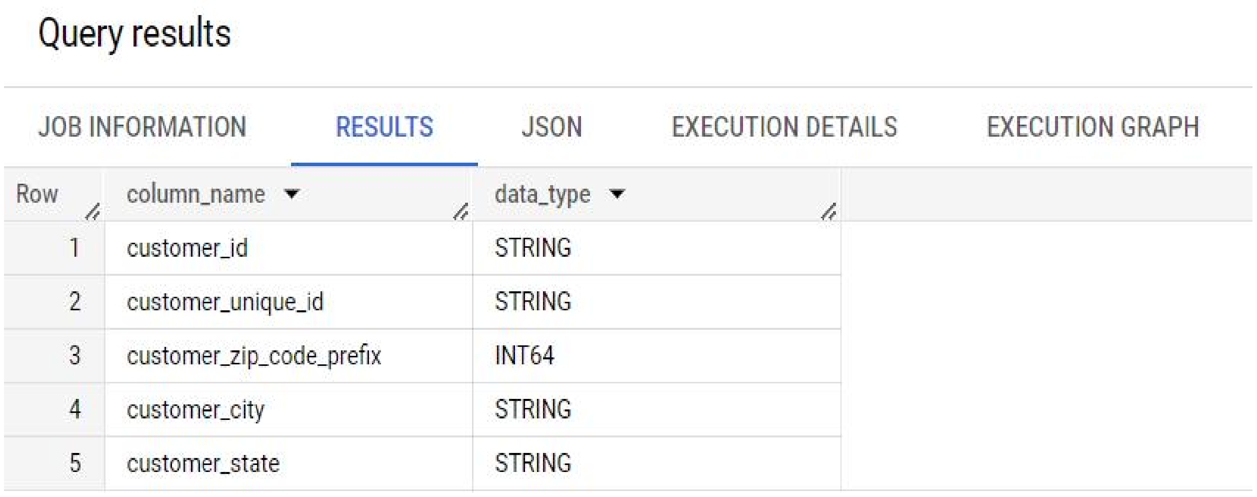
SQL PROJECT – TARGET

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.***

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



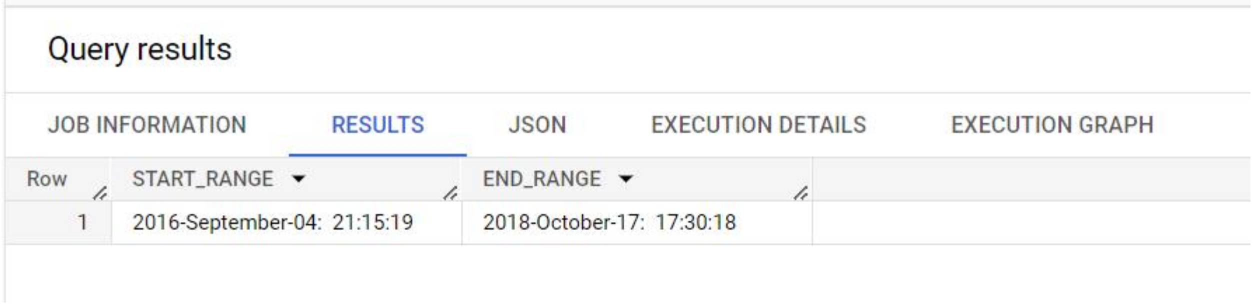
Query results

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

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.***

```
SELECT FORMAT_DATE("%Y-%B-%d:  %T", MIN(order_purchase_timestamp))
START_RANGE,
FORMAT_DATE("%Y-%B-%d:  %T", MAX(order_purchase_timestamp)) END_RANGE
FROM `target_sql.orders`
```

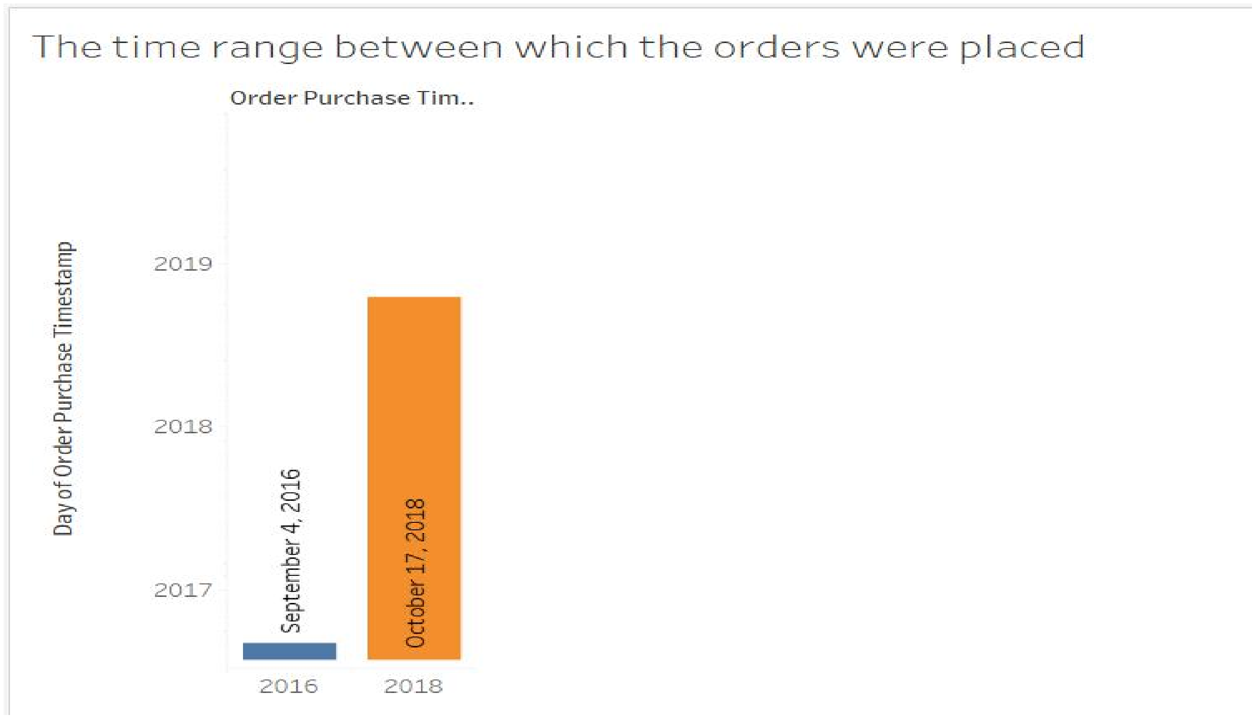


Query results

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

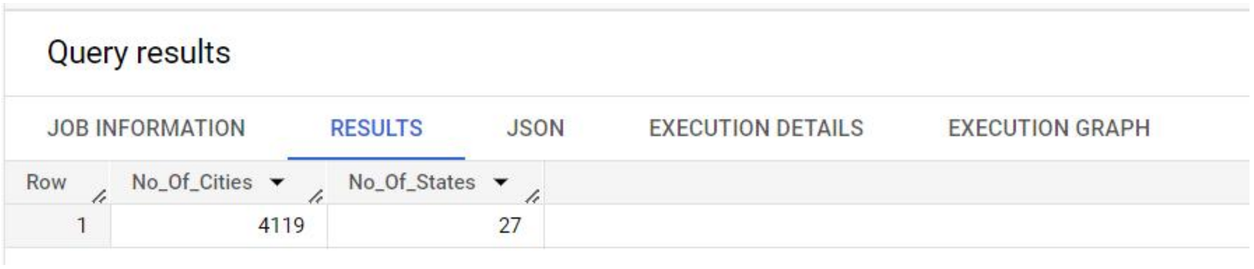
Row	START_RANGE	END_RANGE
1	2016-September-04: 21:15:19	2018-October-17: 17:30:18

Visualisation



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

```
SELECT
COUNT(DISTINCT customer_city) AS No_Of_Cities,COUNT(DISTINCT customer_state)
AS No_Of_States
FROM `target_sql.customers` c INNER JOIN `target_sql.orders` o
ON c.customer_id=o.customer_id
```



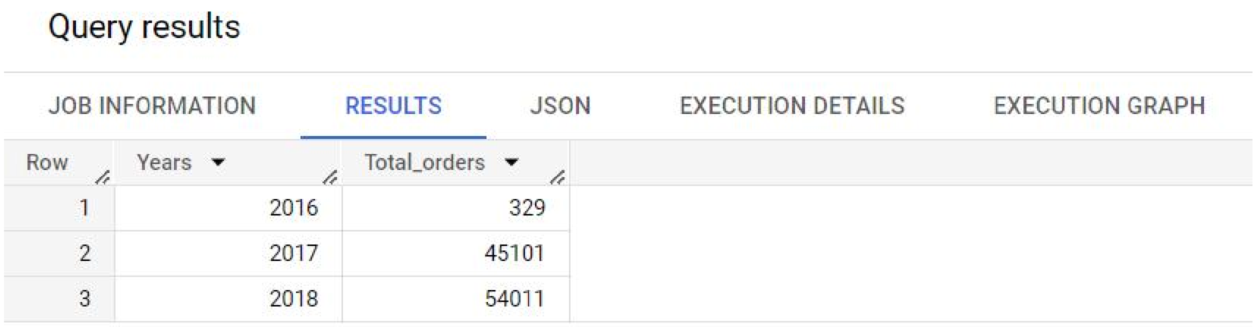
Query results

Row	No_Of_Cities	No_Of_States
1	4119	27

2) In-depth Exploration:

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

```
SELECT EXTRACT(year FROM order_purchase_timestamp) Years,
COUNT(order_id) Total_orders FROM `target_sql.orders`
GROUP BY 1
ORDER BY 1
```

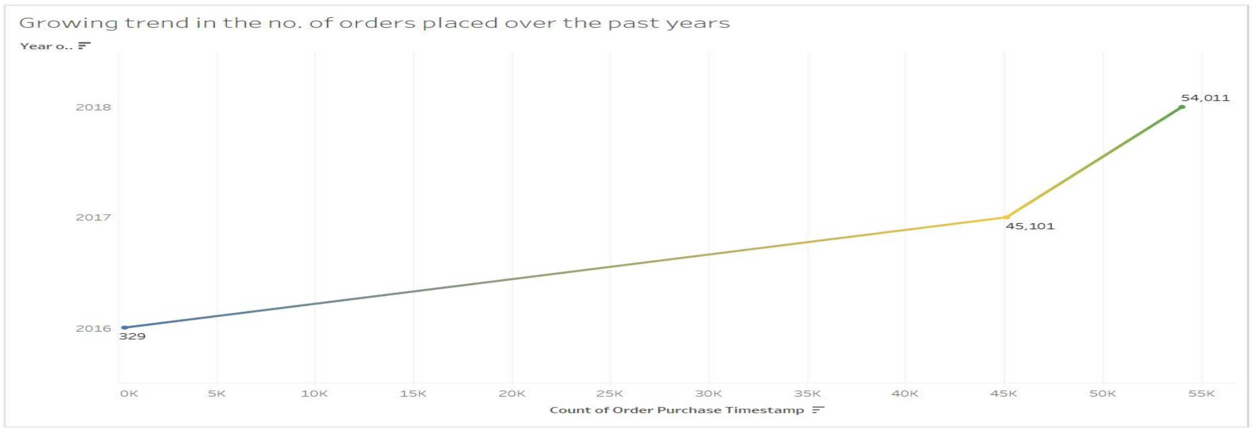


Query results

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

Row	Years	Total_orders
1	2016	329
2	2017	45101
3	2018	54011

Visualisation

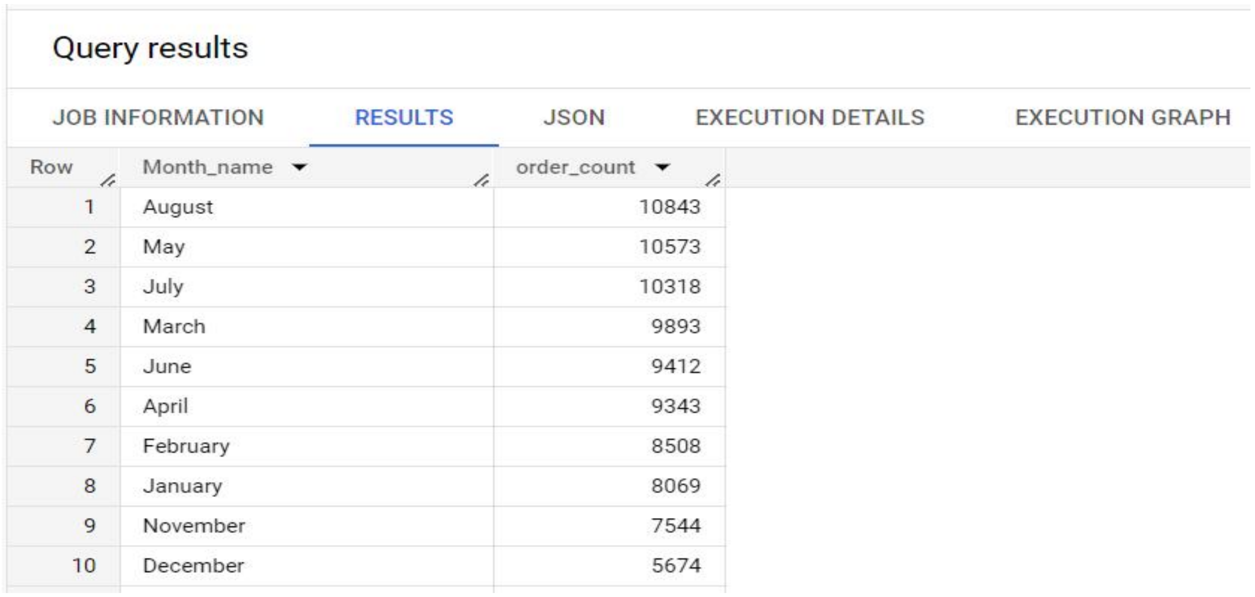


Insights:

- Yes, there is a growing trend in the number of orders. The growth percentage between 2016 to 2017 is 13608.51% where as in 2018 the percentage is only 19.76%.

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

```
SELECT DISTINCT(FORMAT_DATE("%B",order_purchase_timestamp))
AS Month_name,
COUNT(order_id) AS order_count
FROM `target_sql.orders`
GROUP BY Month_name
ORDER BY order_count DESC
```

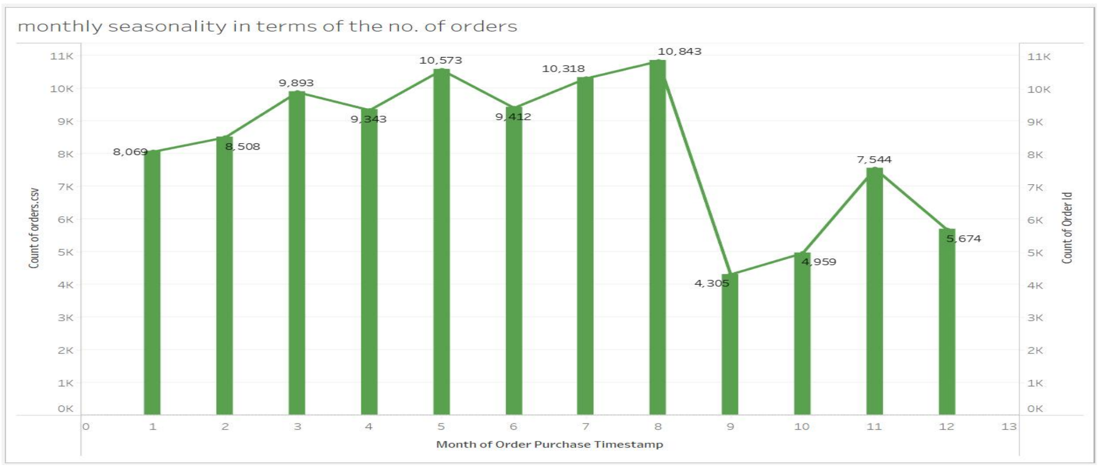


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	Month_name	order_count
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

visualisation



Insights:

- There is a high sales in the month of August, in December month half of the sales are reduced and September month has the least sales.
- So altogether the last quarter of the year has low sales value.

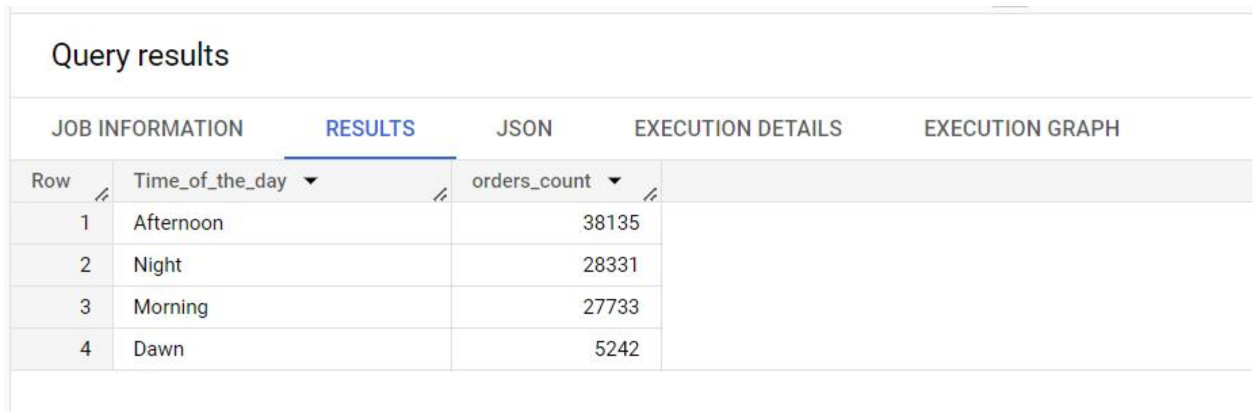
Recommendation:

- The sale is found to be less during spring season in Brazil. So spring offers like buy 1 get 1 can be introduced to clear out the old stocks.
- During May, June, July and August high sales are happening. So it is recommended to focus on inventory stock.

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***

```
SELECT Time_of_the_day, COUNT(Time_of_the_day) AS orders_count
FROM
(SELECT order_purchase_timestamp,
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'
    ELSE 'Night' END AS Time_of_the_day
FROM `target_sql.orders`) A
GROUP BY Time_of_the_day
ORDER BY orders_count DESC
```



Row	Time_of_the_day	orders_count
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Insights:

- Customers are actively purchasing during Afternoon and Night.
- During dawn people make less no. of. Orders.

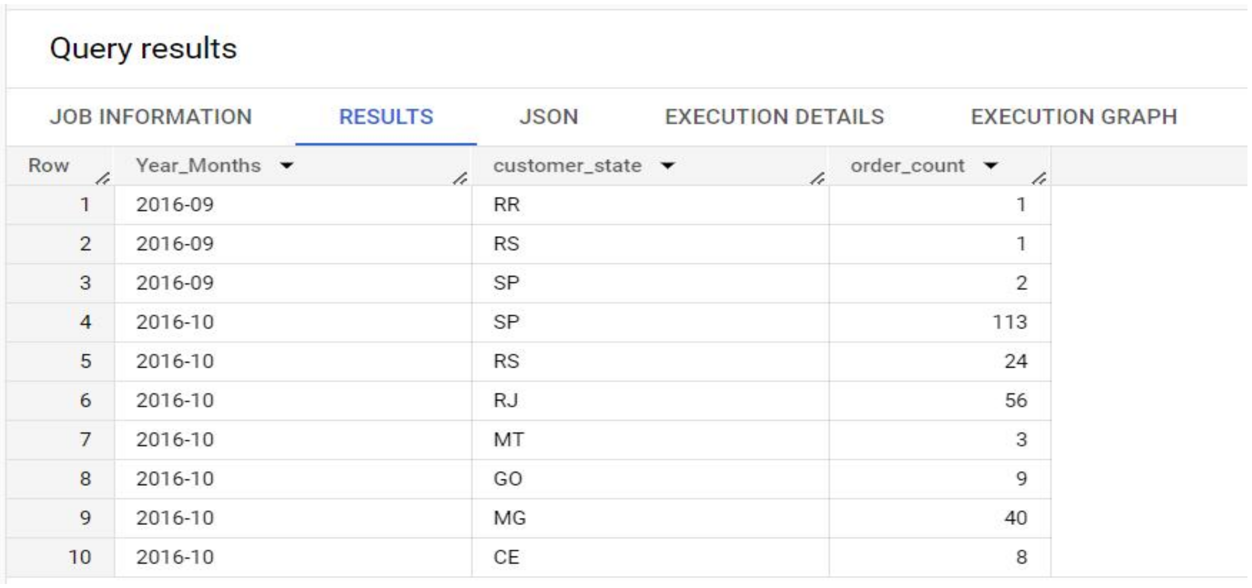
Recommendation:

- The business must ensure that there are enough sales people during Afternoon and Night in order to help the customers to have better shopping experience.
- In order to increase the sales in dawn, the business can give extra discounts during dawn.

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

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

```
SELECT
FORMAT_DATE('%Y-%m',o.order_purchase_timestamp) AS Year_Months,
c.customer_state, COUNT(o.order_id) AS order_count
FROM `target_sql.orders` o INNER JOIN `target_sql.customers` c
ON o.customer_id = c.customer_id
GROUP BY Year_Months, c.customer_state
ORDER BY Year_Months
```



Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

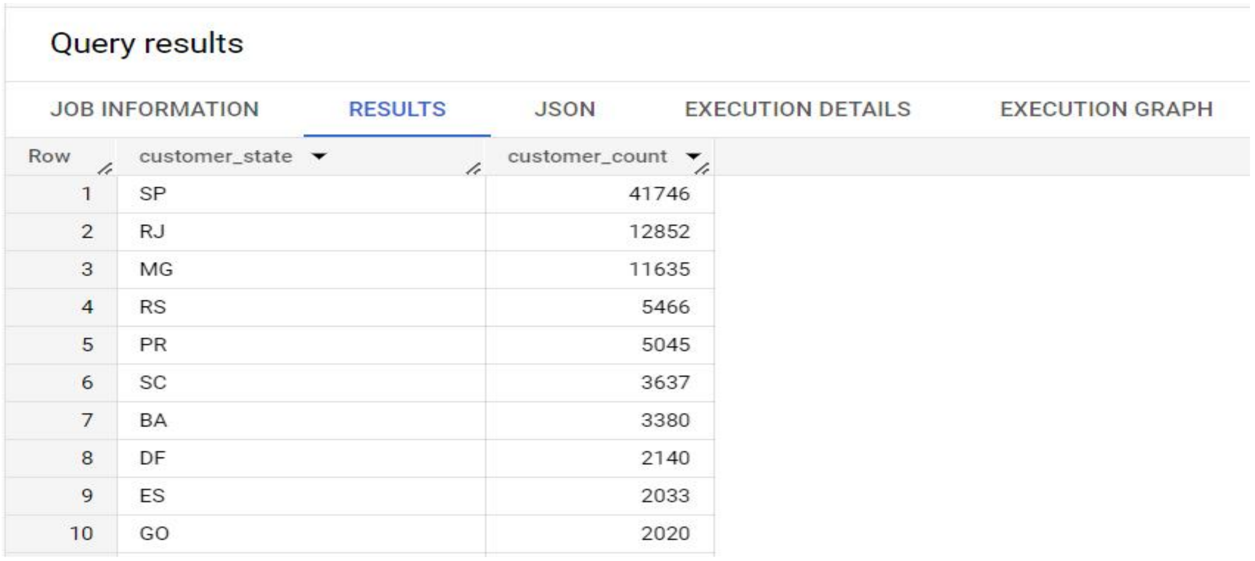
Row	Year_Months	customer_state	order_count
1	2016-09	RR	1
2	2016-09	RS	1
3	2016-09	SP	2
4	2016-10	SP	113
5	2016-10	RS	24
6	2016-10	RJ	56
7	2016-10	MT	3
8	2016-10	GO	9
9	2016-10	MG	40
10	2016-10	CE	8

Insights:

- From the above analysis we can understand that state SP is the no. 1 state to make more orders.
- RR state has the least no of orders in all the years.

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

```
SELECT  customer_state, COUNT(customer_id) customer_count
FROM `target_sql.customers`
GROUP BY customer_state
ORDER BY customer_count DESC
```

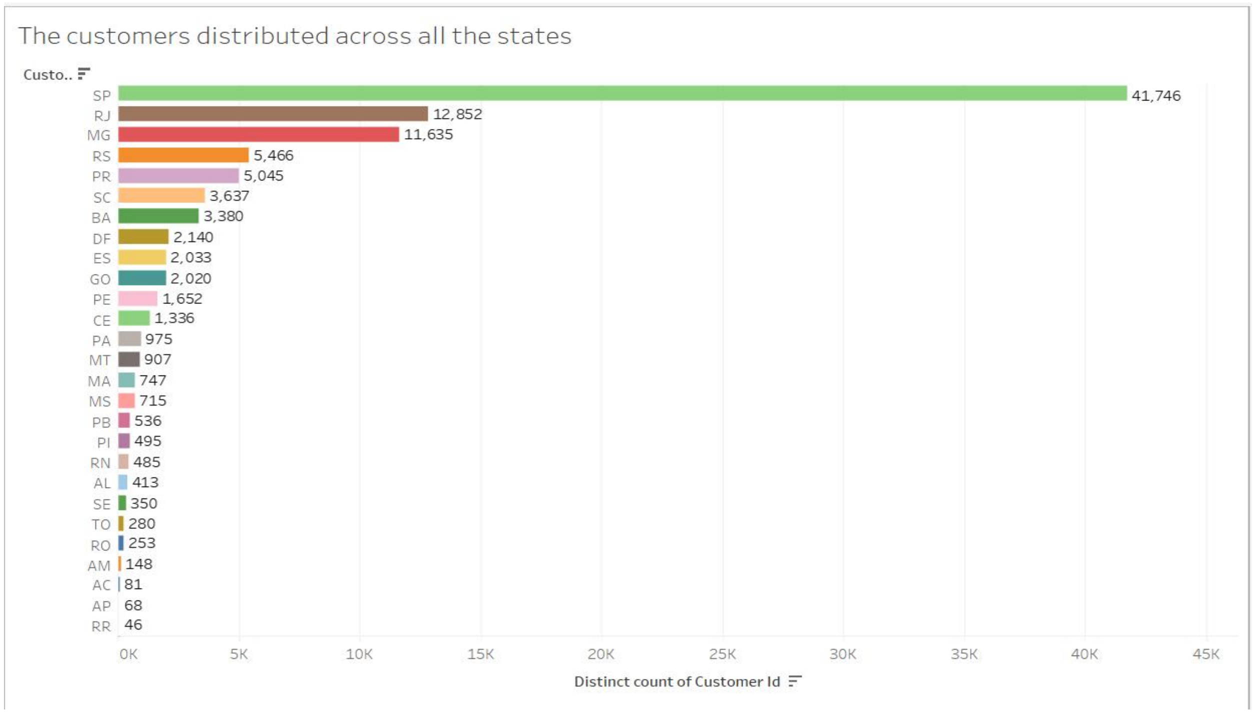


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	customer_state	customer_count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020

visualisation



Insights:

- Highest no. of. Customers are from state SP.
- Least no. of. Customers are from state RR.

Recommendation:

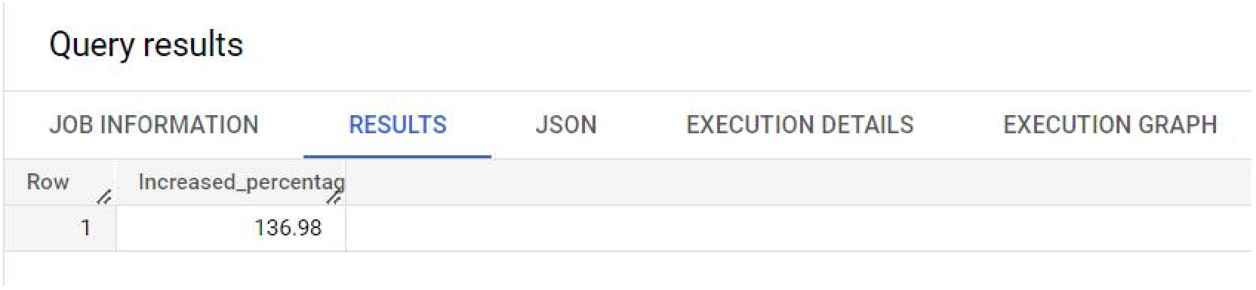
- The business has to be on focus to keep satisfying the customers by providing more customer support like instant replacements, refunds etc.
- To cover other states, the business has to focus on marketing on television and social media platforms.
- Getting feedback and suggestions from customers also will help to improve the business in those areas.

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.

```
CREATE VIEW target_sql.percentage_increase  AS
 (WITH cte  AS
(SELECT  EXTRACT(year FROM o.order_purchase_timestamp)  AS       Year_month,
SUM(p.payment_value)  AS sales_17
      FROM `target_sql.orders` o INNER JOIN `target_sql.payments` p
      ON o.order_id = p.order_id
      WHERE EXTRACT(MONTH FROM order_purchase_timestamp)
      BETWEEN 1 AND 8 AND
   EXTRACT(YEAR  FROM order_purchase_timestamp) IN (2017,2018)
      GROUP BY 1
      ORDER BY 1)

  SELECT Year_month, sales_17,
        LEAD(sales_17) OVER(ORDER BY sales_17)  AS sales_18,
        FROM cte
   ORDER BY 1);
SELECT ROUND((((sales_18-sales_17)/sales_17)*100),2) AS Increased_percentage
FROM `target_sql.percentage_increase`
LIMIT 1
```



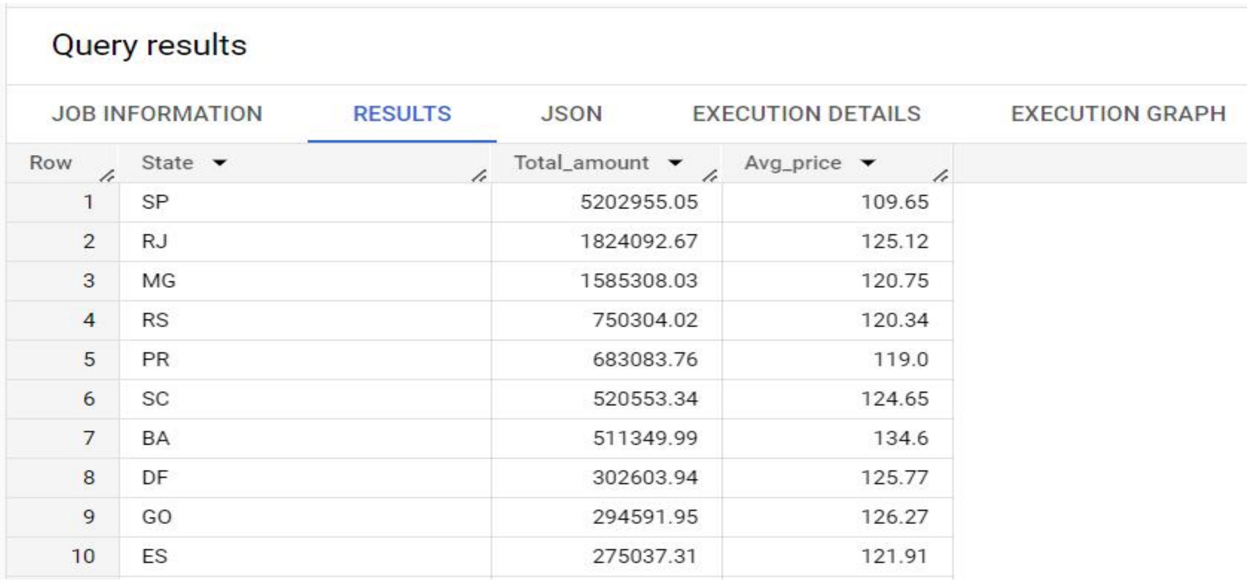
Row	Increased_percentag
1	136.98

Insights:

- The increase percentage is 136.98 from 2017 (Jan – Aug) to 2018 (Jan – Aug)

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

```
SELECT c.customer_state  AS State,
ROUND(SUM(i.price),2)  AS Total_amount, ROUND(AVG(i.price),2) AS Avg_price
FROM `target_sql.orders` o INNER JOIN `target_sql.order_items` i
ON o.order_id = i.order_id
INNER JOIN  `target_sql.customers` c
ON o.customer_id = c.customer_id
GROUP BY 1
ORDER BY 2 DESC
```



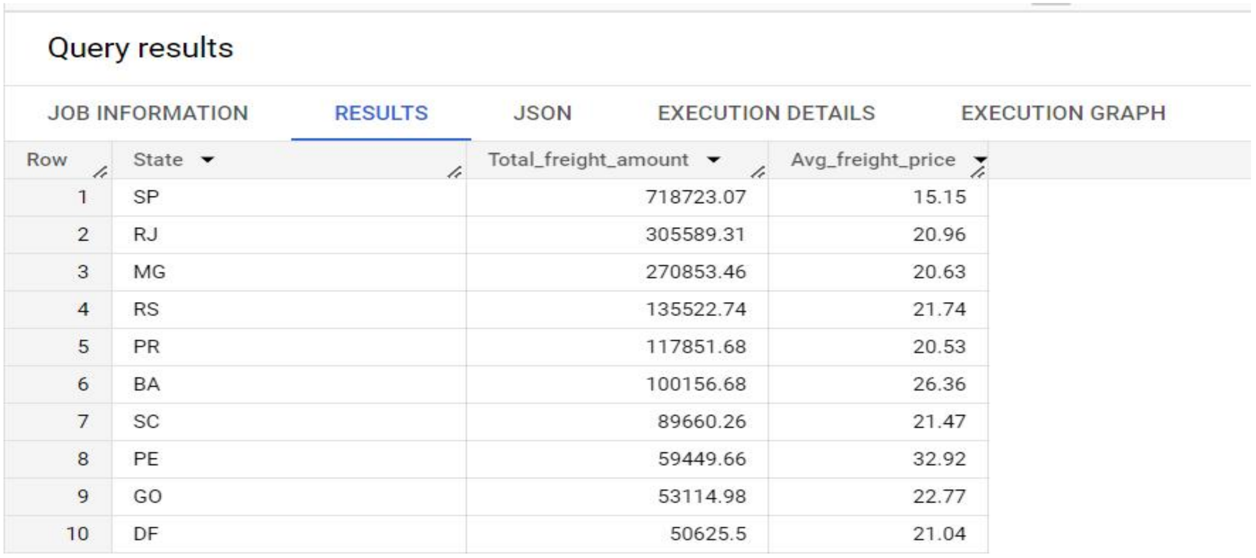
Row	State	Total_amount	Avg_price
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91

Insights:

- SP state has the least avg.price 109.65
- State PB has the highest avg.price 191.48

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

```
SELECT c.customer_state AS State,
ROUND(SUM(i.freight_value),2) AS Total_freight_amount, ROUND(AVG(i.freight_value),2)
AS Avg_freight_price
FROM `target_sql.orders` o INNER JOIN `target_sql.order_items` i
ON o.order_id = i.order_id
INNER JOIN  `target_sql.customers` c
ON o.customer_id = c.customer_id
GROUP BY 1
ORDER BY 2 DESC
```


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

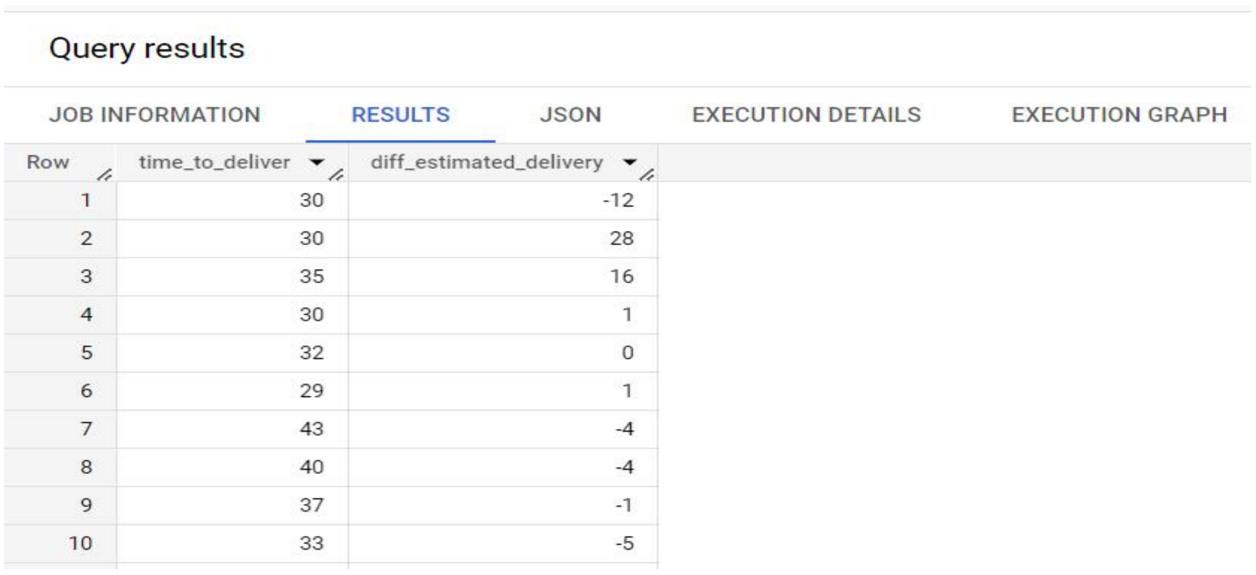
Row	State	Total_freight_amount	Avg_freight_price
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04

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:

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

```
SELECT  DATE_DIFF
( order_delivered_customer_date,order_purchase_timestamp, DAY)  time_to_deliver,
DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date, DAY)
diff_estimated_delivery
FROM `target_sql.orders`
```


Query results

Row	time_to_deliver	diff_estimated_delivery
1	30	-12
2	30	28
3	35	16
4	30	1
5	32	0
6	29	1
7	43	-4
8	40	-4
9	37	-1
10	33	-5

Recommendation:

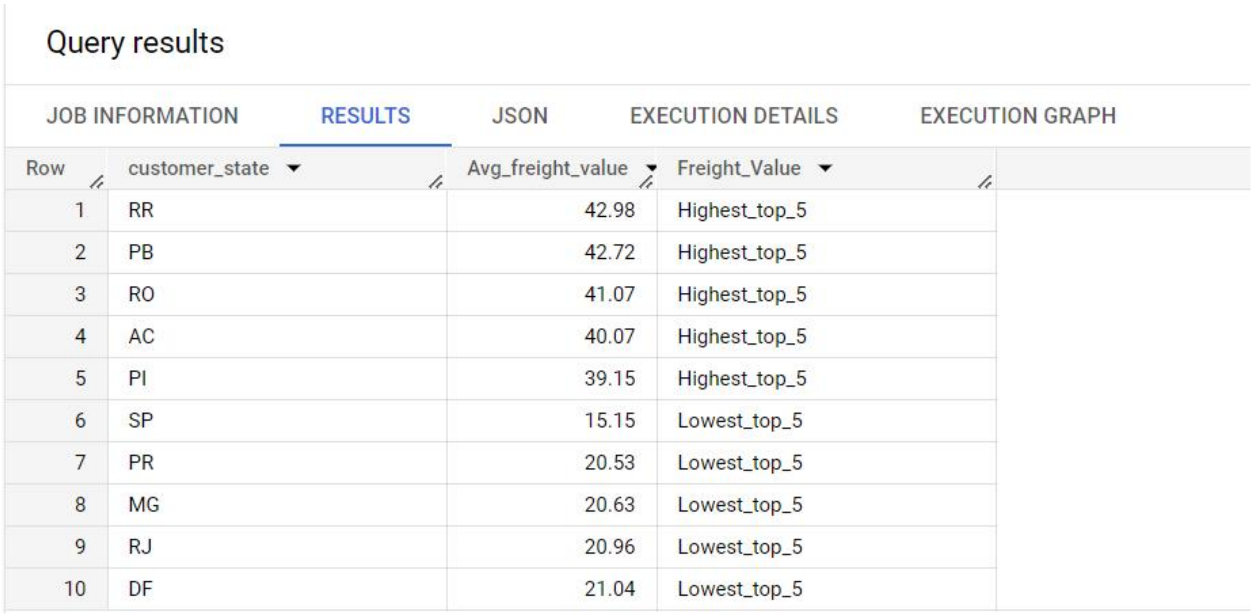
- The business should improve the service quality provided by the delivery partners.
- It is important to replace the slow delivery with better service for the customers' satisfaction.

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

```
 WITH TOP_CTE  AS

          (SELECT customer_state, ROUND(AVG(freight_value),2)  AS Avg_freight_value
 FROM `target_sql.orders`o JOIN `target_sql.order_items`oi
ON o.order_id = oi.order_id
 JOIN `target_sql.customers`c
ON o.customer_id = c.customer_id
GROUP BY customer_state)

(SELECT customer_state, TOP_CTE.Avg_freight_value, 'Highest_top_5' as Freight_Value
FROM TOP_CTE
ORDER BY 2 DESC
 LIMIT 5)
UNION ALL
(SELECT customer_state, TOP_CTE.Avg_freight_value, 'Lowest_top_5' as Freight_Value
FROM TOP_CTE
ORDER BY 2 ASC
LIMIT 5)
```


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	customer_state	Avg_freight_value	Freight_Value
1	RR	42.98	Highest_top_5
2	PB	42.72	Highest_top_5
3	RO	41.07	Highest_top_5
4	AC	40.07	Highest_top_5
5	PI	39.15	Highest_top_5
6	SP	15.15	Lowest_top_5
7	PR	20.53	Lowest_top_5
8	MG	20.63	Lowest_top_5
9	RJ	20.96	Lowest_top_5
10	DF	21.04	Lowest_top_5

Insights:

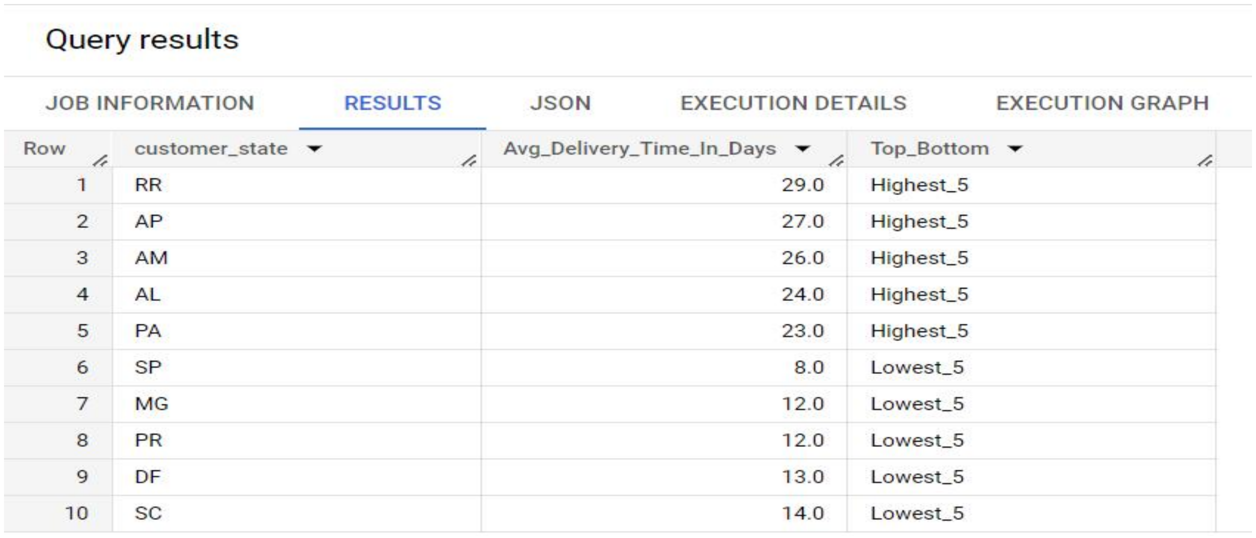
- RR state pays the highest freight value
- SP state pays the least freight value

Recommendation:

- From the above analysis, it is recommended that the business can extend or limit their delivery areas based on the insights.

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

```
WITH CTE AS
(SELECT c.customer_state, ROUND(AVG(date_diff(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY))) AS Avg_Delivery_Time_In_Days
FROM `target_sql.orders` o INNER JOIN `target_sql.customers` c
ON o.customer_id = c.customer_id
GROUP BY 1)
(SELECT customer_state, CTE.Avg_Delivery_Time_In_Days, 'Highest_5' AS Top_Bottom
FROM CTE
ORDER BY 2 DESC
LIMIT 5)
UNION ALL
(SELECT customer_state, CTE.Avg_Delivery_Time_In_Days, 'Lowest_5' AS Top_Bottom
FROM CTE
ORDER BY 2 ASC
LIMIT 5)
```


Query results

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

Row	customer_state	Avg_Delivery_Time_In_Days	Top_Bottom
1	RR	29.0	Highest_5
2	AP	27.0	Highest_5
3	AM	26.0	Highest_5
4	AL	24.0	Highest_5
5	PA	23.0	Highest_5
6	SP	8.0	Lowest_5
7	MG	12.0	Lowest_5
8	PR	12.0	Lowest_5
9	DF	13.0	Lowest_5
10	SC	14.0	Lowest_5

Insights:

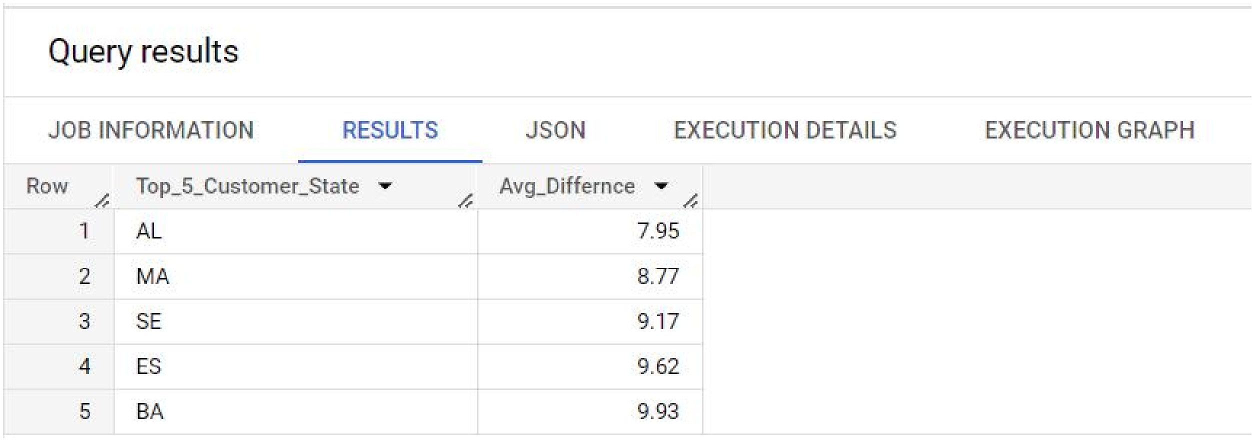
- The delivery duration is more in RR state.

Recommendation:

- It's recommended to be cautious on the estimated delivery date
- It will not affect the business in missing out any customers.

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.***

```
SELECT customer_state  AS Top_5_Customer_State,
ROUND(AVG(Dif_Estimated_Delivery),2)  Avg_Differnce
FROM
(SELECT c.customer_state,
DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY)
Dif_Estimated_Delivery
FROM `target_sql.orders` o INNER JOIN `target_sql.customers` c
ON o.customer_id = c.customer_id) a
GROUP BY 1
ORDER BY 2 ASC
LIMIT 5
```


Query results

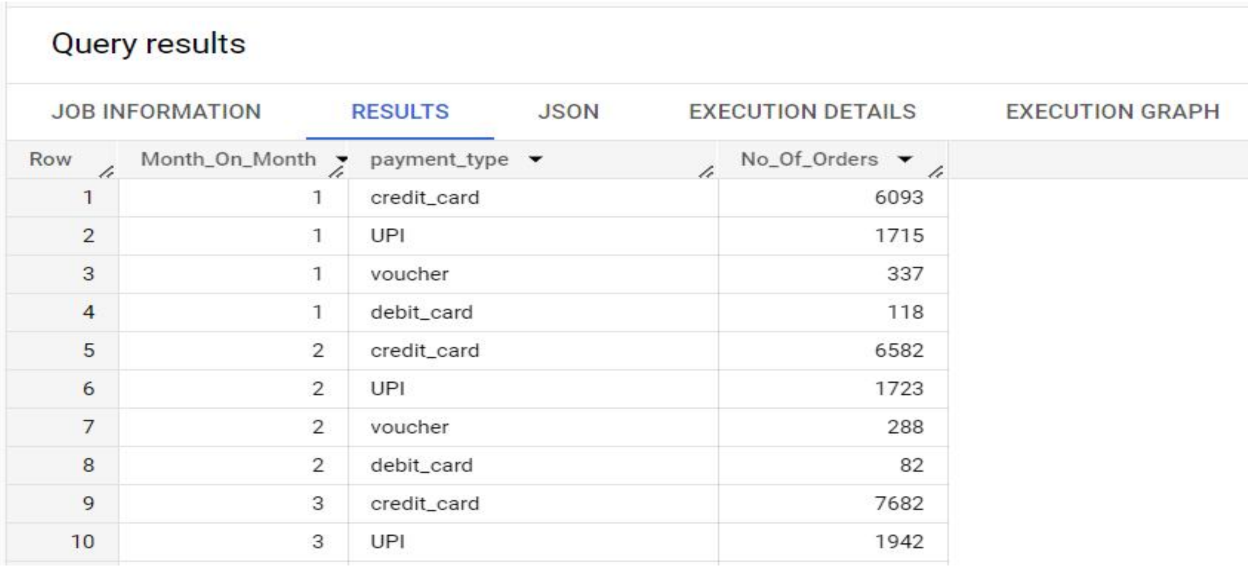
JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	Top_5_Customer_State	Avg_Differnce
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

6) Analysis based on the payments:

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

```
SELECT EXTRACT(MONTH  FROM  o.order_purchase_timestamp)  AS Month_On_Month,
p.payment_type, COUNT(DISTINCT o.order_id) AS No_Of_Orders
FROM `target_sql.payments` p INNER JOIN `target_sql.orders` o
ON p.order_id = o.order_id
GROUP BY 1,2
ORDER BY 1 ASC, 3 DESC
```


Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	Month_On_Month	payment_type	No_Of_Orders
1	1	credit_card	6093
2	1	UPI	1715
3	1	voucher	337
4	1	debit_card	118
5	2	credit_card	6582
6	2	UPI	1723
7	2	voucher	288
8	2	debit_card	82
9	3	credit_card	7682
10	3	UPI	1942

Insights:

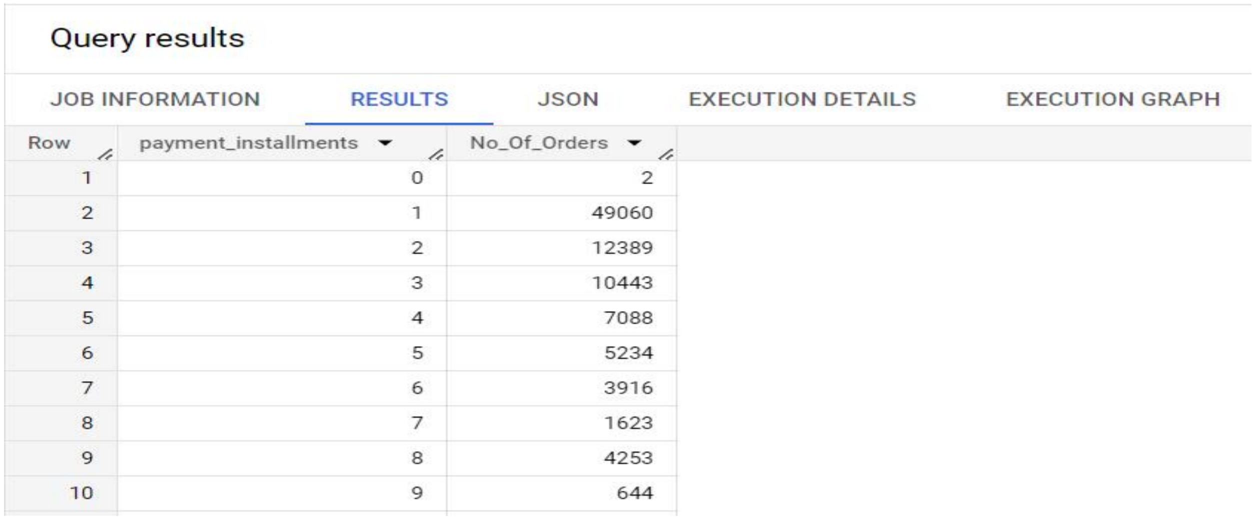
- Most of the customers are making their payments using credit cards.

Recommendation:

- So the credit card and UPI users can be given cash backs, vouchers and extra discounts

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

```
SELECT  payment_installments, COUNT(DISTINCT order_id)  AS No_Of_Orders
FROM `target_sql.payments`
GROUP BY 1
ORDER BY 1
```



Row	payment_installments	No_Of_Orders
1	0	2
2	1	49060
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:

- As the no.of.installments increases the no.of.orders reduces.

Recommendation:

- They can introduce no cost EMI system for certain duration.
- It will increase more customers.