Analysing Brazil's Target Business Performance with SQL

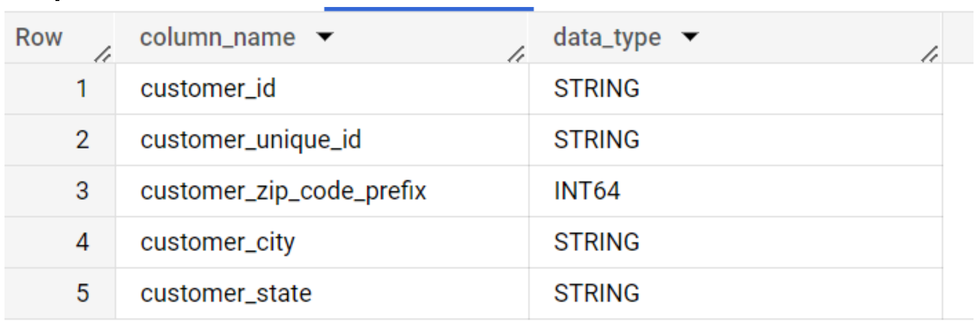
Created by:
Tanisha Singhal
Email: 22tanishasinghal@gmail.com
LinkedIn: linkedin.com/in/tanishasinghal
Github: github.com/tanishasinghal

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.INFORMATION_SCHEMA.COLUMNS
WHERE table_name = "customers"
```

Output:



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

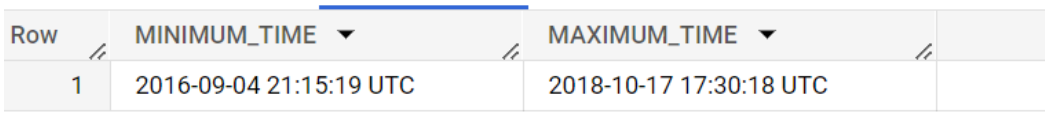
Insights:
This implies that
- customer_id, customer_unique_id, customer_city, and customer_state: are stored in STRING data type. Thus it can be of the type Char or Varchar.
- Customer_zip_code_prefix: is stored in INTEGER data type. This implies that it represents a whole number having a size of 64-bits.

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

Query:
```
SELECT
```

```
  MIN(order_purchase_timestamp) AS MINIMUM_TIME,
  MAX(order_purchase_timestamp) AS MAXIMUM_TIME
FROM `Target.orders`
```

Output:



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

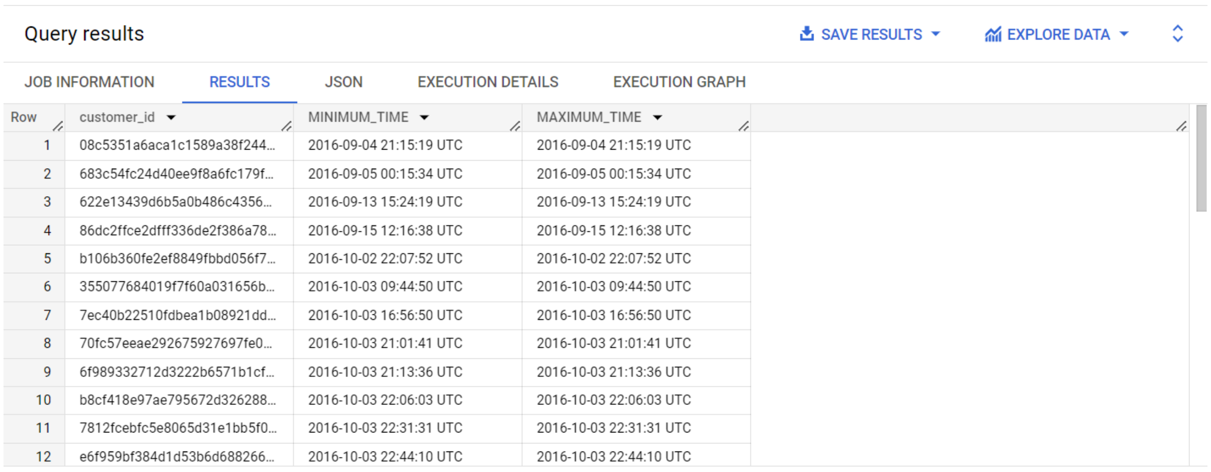
Insights:
The above output signifies the minimum and maximum time of the order placed.
Thus it gives us the information that the first order was placed on “2016-09-04 21:15:19 UTC” and the last order was placed on “2018-10-17 17:30:18 UTC”.

Extending this question further, we can calculate the time range for the orders that each customer has placed.

Query:

```
SELECT
  customer_id,
  MIN(order_purchase_timestamp) AS MINIMUM_TIME,
  MAX(order_purchase_timestamp) AS MAXIMUM_TIME
FROM `Target.orders`
GROUP BY customer_id
```

Output:



Row	customer_id	MINIMUM_TIME	MAXIMUM_TIME
1	08c5351a6aca1c1589a38f244…	2016-09-04 21:15:19 UTC	2016-09-04 21:15:19 UTC
2	683c54fc24d40ee9f8a6fc179f…	2016-09-05 00:15:34 UTC	2016-09-05 00:15:34 UTC
3	622e13439d6b5a0b486c4356…	2016-09-13 15:24:19 UTC	2016-09-13 15:24:19 UTC
4	86dc2ffce2dfff336de2f386a78…	2016-09-15 12:16:38 UTC	2016-09-15 12:16:38 UTC
5	b106b360fe2ef8849fbbd056f7…	2016-10-02 22:07:52 UTC	2016-10-02 22:07:52 UTC
6	355077684019f7f60a031656b…	2016-10-03 09:44:50 UTC	2016-10-03 09:44:50 UTC
7	7ec40b22510fdbea1b08921dd…	2016-10-03 16:56:50 UTC	2016-10-03 16:56:50 UTC
8	70fc57eeae292675927697fe0…	2016-10-03 21:01:41 UTC	2016-10-03 21:01:41 UTC
9	6f989332712d3222b6571b1cf…	2016-10-03 21:13:36 UTC	2016-10-03 21:13:36 UTC
10	b8cf418e97ae795672d326288…	2016-10-03 22:06:03 UTC	2016-10-03 22:06:03 UTC
11	7812fcebfc5e8065d31e1bb5f0…	2016-10-03 22:31:31 UTC	2016-10-03 22:31:31 UTC
12	e6f959bf384d1d53b6d688266…	2016-10-03 22:44:10 UTC	2016-10-03 22:44:10 UTC

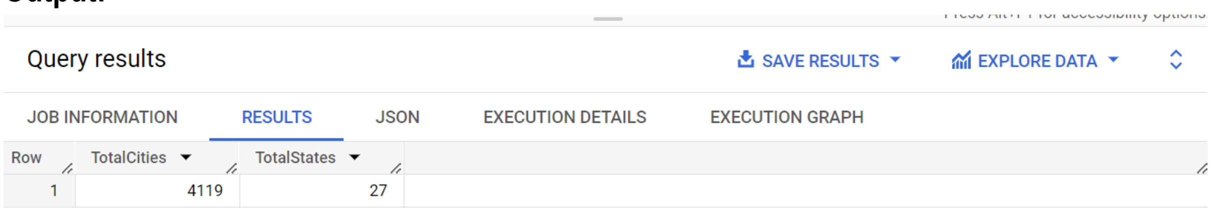
Insights:
The above data signifies the minimum and maximum time of the order being placed by each customer.
In other words, it gives us information about the first and the last order placed by each customer.

3. **Count the number of Cities and States in our dataset.**
   Interpretation: To count the total number of Cities and States.

Query:

```
SELECT
  COUNT(DISTINCT customer_city) AS TotalCities,
  COUNT(DISTINCT customer_state) AS TotalStates
FROM `Target.customers`
```

Output:



Row	TotalCities	TotalStates
1	4119	27

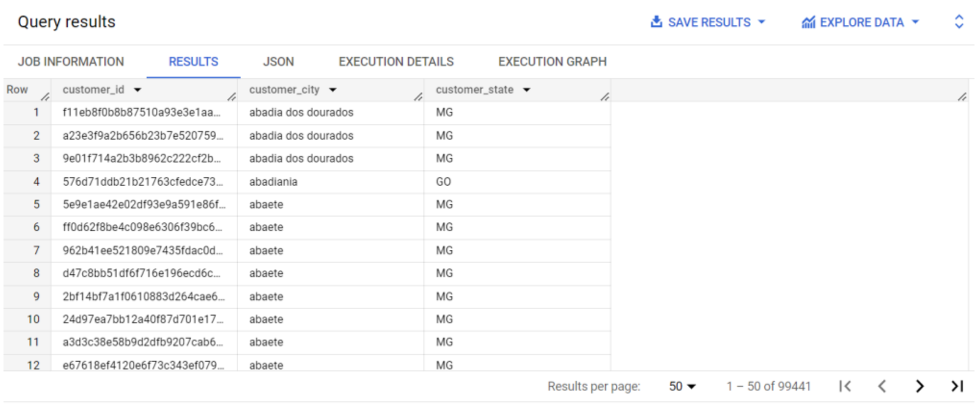
Insights:
This implies that in our dataset of Brazil, there are a total of 4119 Cities and 27 States.

Extending this question further, we can have the Cities and States of the customers who placed an order.

Query:

```
SELECT
  c.customer_id,
  c.customer_city,
  c.customer_state
FROM `Target.customers` as c
INNER JOIN `Target.orders` as o
ON c.customer_id = o.customer_id
ORDER BY c.customer_city, c.customer_state
```

Output:


Query results

Row	customer_id	customer_city	customer_state
1	f11eb8f0b8b87510a93e3e1aa...	abadia dos dourados	MG
2	a23e3f9a2b656b23b7e520759...	abadia dos dourados	MG
3	9e01f714a2b3b8962c222cf2b...	abadia dos dourados	MG
4	576d71ddb21b21763cfedce73...	abadiania	GO
5	5e9e1ae42e02df93e9a591e86f...	abaete	MG
6	ff0d62f8be4c098e6306f39bc6...	abaete	MG
7	962b41ee521809e7435fdac0d...	abaete	MG
8	d47c8bb51df6f716e196ecd6c...	abaete	MG
9	2bf14bf7a1f0610883d264cae6...	abaete	MG
10	24d97ea7bb12a40f87d701e17...	abaete	MG
11	a3d3c38e58b9d2dfb9207cab6...	abaete	MG
12	e67618ef4120e6f73c343ef079...	abaete	MG

Results per page: 50 1 – 50 of 99441

Insights:
The above output gives us the information of each customer and their respective City and State.

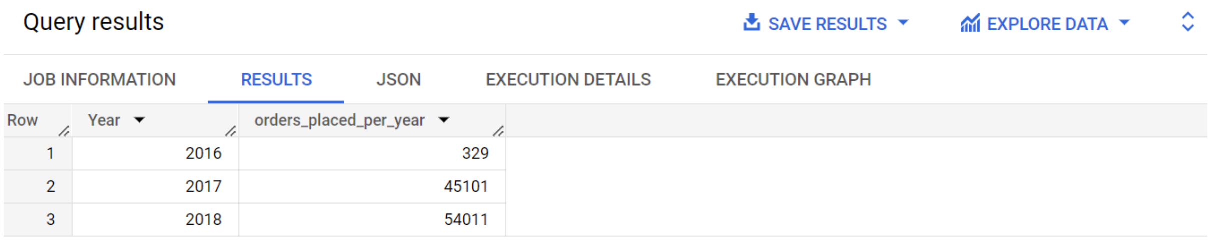
In-depth Exploration:

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,
  COUNT(*) as orders_placed_per_year
FROM `Target.orders`
GROUP BY Year
ORDER BY Year
```

Output:


Query results

Row	Year	orders_placed_per_year
1	2016	329
2	2017	45101
3	2018	54011

Insights:
Yes, there is a growing trend in the number of orders placed over the past years.

- The number of orders placed by the customers is increasing per year. It was 329 in the year 2016 → 45,101 in the year 2017 → 54,011 in the year 2018.
  Thus, the total orders placed were: 99441

- It should be noted that the orders placed in 2016 are significantly less than the following years because the orders placed in 2016 are calculated from the month of September.

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

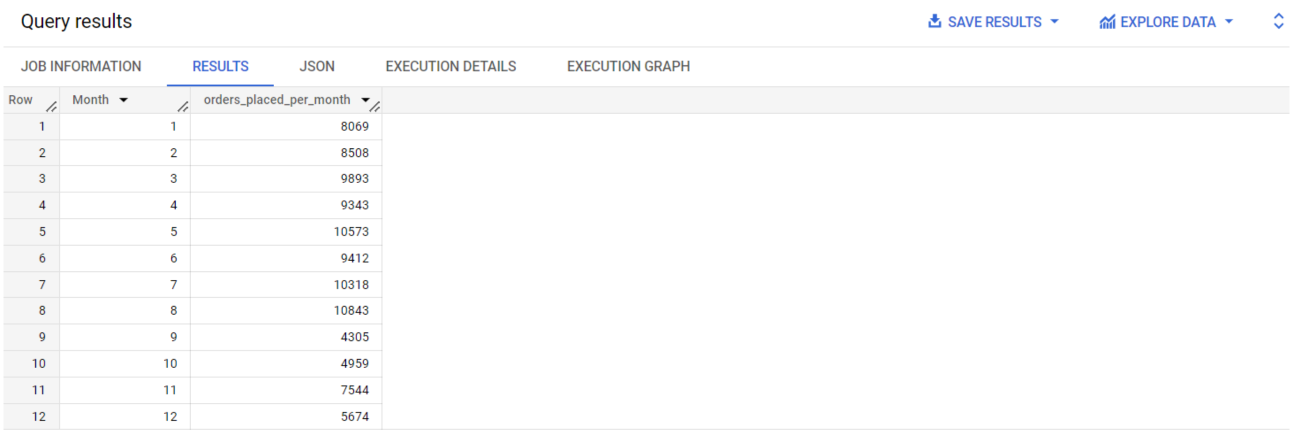
   Yes and we can approach this question in two ways.

   Approach: Calculating the number of orders being placed for months, irrespective of the year in our dataset.

   Query:

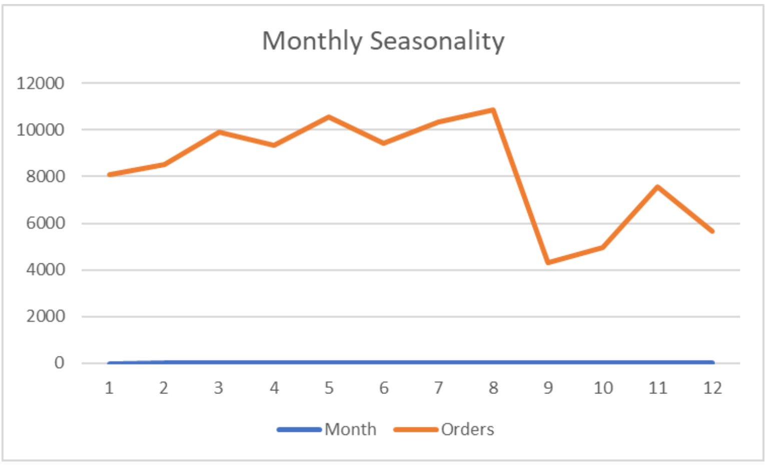
```
SELECT
  EXTRACT(MONTH from order_purchase_timestamp) AS Month,
  COUNT(*) as orders_placed_per_month
FROM `Target.orders`
GROUP BY Month
ORDER BY Month
```

   Output:



Row	Month	orders_placed_per_month
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:
Below is the graph drawn for the obtained data which represents the number of orders placed per month.



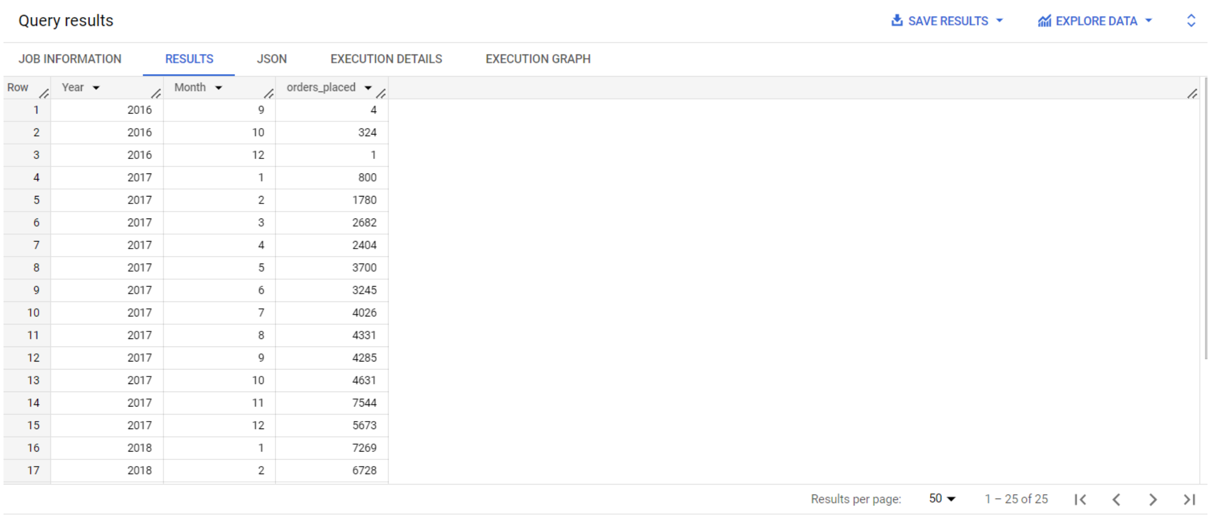
- It can be seen that comparatively a large number of orders are placed from the months January to August.
- The highest number of orders is placed in the month of August. It could be because July is generally the **coldest month** in Brazil. Thus, with the season changing from June to July to August, the number of purchases must have increased as the customers must have bought winter wear.
- With the **start of the new year**, the number of orders placed is increasing from December to January. Brazilian's celebrate their holidays with enthusiasm and the purchases of items like clothes, food and beverages, cleaning supplies, home decor, etc. increases simultaneously.
- It could also be due to the fact that **one of the batch of colleges in Brazil starts in the month of February to June**. Thus, students purchase their college supplies at reasonable prices from target.
- Between the end of February or the start of March, one of the **most famous holidays** of Brazil, the Carnival, is celebrated. Locals celebrate and make purchases. This could be one of the reasons for the increase in purchases in the months of February and March.
- The month of November can also be seen with a little increase in the number of orders. This could be due to the **holidays falling in this month like, Brazil's Republic day, All's Souls' day, and the most important black friday.**

Extending this question further, we can calculate the orders placed per month per year.

Query:

```
SELECT
  EXTRACT(Year from order_purchase_timestamp) AS Year,
  EXTRACT(MONTH from order_purchase_timestamp) AS Month,
  COUNT(*) as orders_placed
FROM `Target.orders`
GROUP BY Year, Month
ORDER BY Year, Month
```

Output:


Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS EXECUTION GRAPH

Row	Year	Month	orders_placed
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
11	2017	8	4331
12	2017	9	4285
13	2017	10	4631
14	2017	11	7544
15	2017	12	5673
16	2018	1	7269
17	2018	2	6728

Results per page: 50 1 – 25 of 25

Insights:

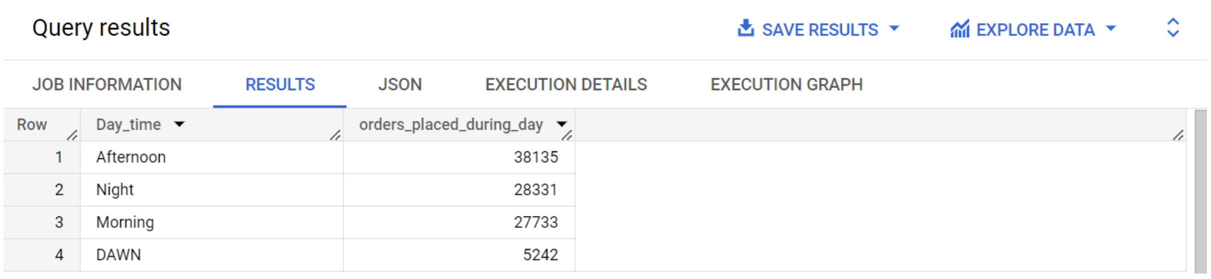
The above data set gives us the further breakdown of the number of orders placed in each month of the respective years.

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
  t2.Day_time,
  SUM(t2.orders_placed) as orders_placed_during_day
FROM (
  SELECT
    t.orders_placed,
    CASE
    WHEN t.hour_of_day BETWEEN 0 AND 6
    THEN "DAWN"
    WHEN t.hour_of_day BETWEEN 7 AND 12
    THEN "Morning"
    WHEN t.hour_of_day BETWEEN 13 AND 18
    THEN "Afternoon"
    WHEN t.hour_of_day BETWEEN 19 AND 23
    THEN "Night"
    END AS Day_time
  FROM (
      SELECT
        EXTRACT(Hour from order_purchase_timestamp) AS hour_of_day,
        COUNT(*) as orders_placed
      FROM `Target.orders`
      GROUP BY hour_of_day) as t)
as t2
GROUP BY t2.Day_time
ORDER BY orders_placed_during_day DESC
```

Output:


Query results

Row	Day_time	orders_placed_during_day
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	DAWN	5242

Insights:

This output gives the distribution of orders throughout the different times of the day.

- Customers in Brazil place most of their orders in the afternoon i.e. between 13-18 hrs. Thus, these are the most active hours in terms of the number of orders being placed.
- Considerable portion of customers also place their orders at night or late-evening hours i.e. between 19-23 hrs.
- This is followed by morning where a lowered number of orders is placed when compared to afternoon and night hours. Thus, some customers prefer to start their day by placing an order but not many.
- Lastly, there are customers who place their order at dawn where fewer customers engage in shopping. Also, this contains the time where most of the customers might be sleeping.

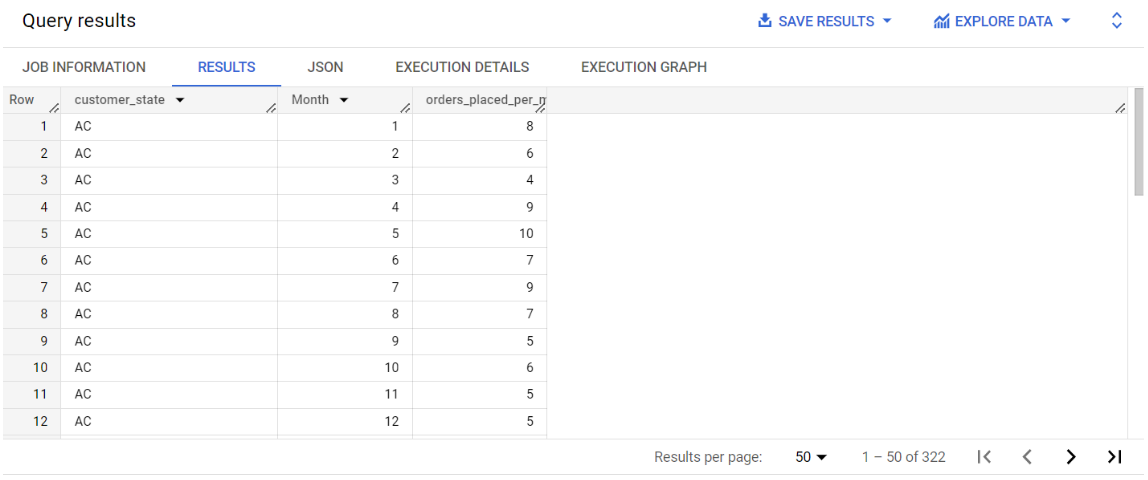
3. Evolution of E-commerce orders in the Brazil region:

1. **Get the month on month number of orders placed in each state.**

Query:
(Approach: calculating the number of orders placed in each state irrespective of the year)

```
SELECT
  c.customer_state,
  EXTRACT(MONTH from order_purchase_timestamp) AS Month,
  COUNT(*) as orders_placed_per_month
FROM `Target.orders` as o
INNER JOIN `Target.customers` as c
ON c.customer_id = o.customer_id
GROUP BY Month,c.customer_state
ORDER BY c.customer_state, Month
```

Output:


Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	Month	orders_placed_per_m
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

Results per page: 50 1 – 50 of 322

Insights:

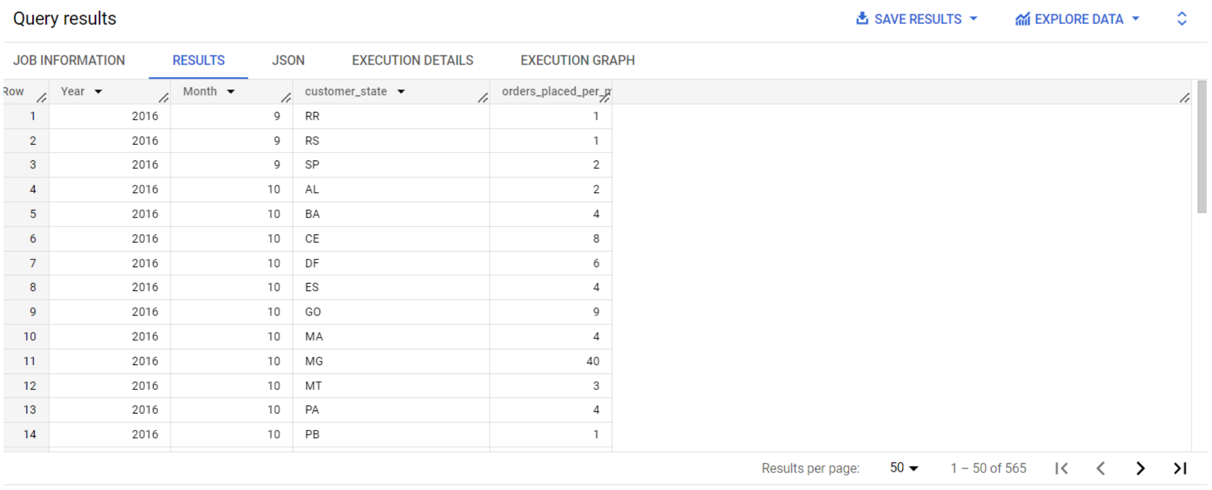
The above query gives us the seasonal demands in each region. Also, tracking the month on month orders can help us evaluate the customer's engagement and satisfaction. Growing or consistent number of orders placed will imply satisfied customers whereas, the declining number of orders placed suggests potential issues which should be evaluated and resolved with the help of suitable strategies. The trend here is also similar to what we observed in the monthly seasonality trend. Months like September, October, November, and December have fewer orders placed.

Extending this question further, we can calculate the orders placed in each state each year.

Query:

```
SELECT
  EXTRACT(YEAR from order_purchase_timestamp) AS Year,
  EXTRACT(MONTH from order_purchase_timestamp) AS Month,
  c.customer_state,
  COUNT(*) as orders_placed_per_month
FROM `Target.orders` as o
INNER JOIN `Target.customers` as c
ON c.customer_id = o.customer_id
GROUP BY Year, Month,c.customer_state
ORDER BY Year, Month, c.customer_state
```

Output:



Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Year	Month	customer_state	orders_placed_per_p
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4
9	2016	10	GO	9
10	2016	10	MA	4
11	2016	10	MG	40
12	2016	10	MT	3
13	2016	10	PA	4
14	2016	10	PB	1

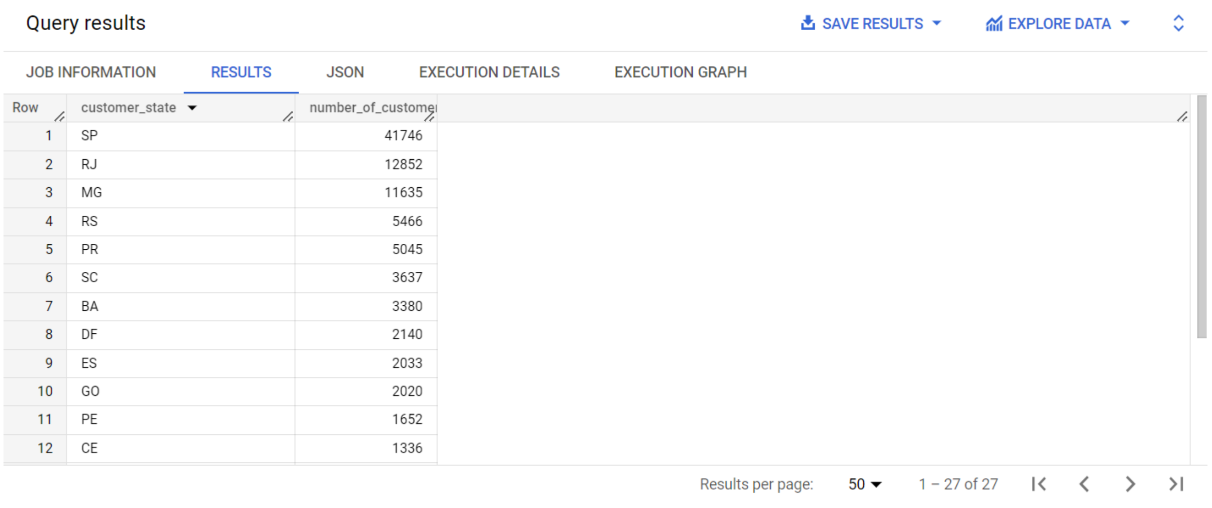
Results per page: 50 1 – 50 of 565

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

Query:

```
SELECT
  customer_state,
  COUNT(DISTINCT customer_id) as number_of_customers
FROM `Target.customers`
GROUP BY customer_state
ORDER BY number_of_customers DESC
```

Output:



Query results

SAVE RESULTS | EXPLORE DATA

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

Row	customer_state	number_of_custome
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
11	PE	1652
12	CE	1336

Results per page: 50 1 – 27 of 27

Insights:

- This output gives us very crucial information about the geographic reach of the company. It gives information about the distribution of customers across different states in Brazil. There is a significant variation in the number of customers from 41746 in the State SP to 46 customers in the State RR.
- Thus, while planning marketing campaigns, business growth, customer acquisition operations and other important strategies, this information will be kept in mind and the ways to increase the number of customers in the lower numbered states will be thought of.

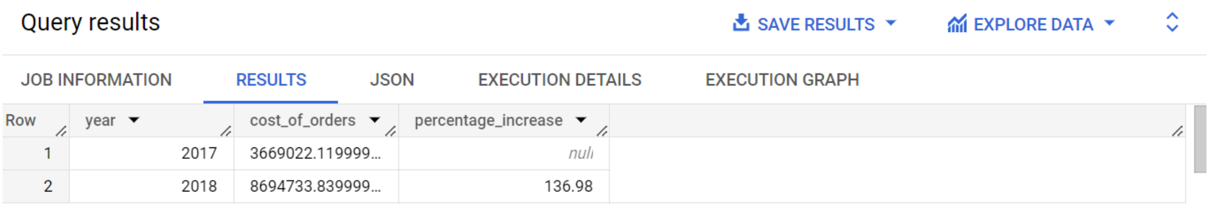
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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
  t.year,
  SUM(t.payment_value) as cost_of_orders,
  ROUND(((SUM(t.payment_value) - (LAG(SUM(t.payment_value)) OVER(ORDER BY
t.year)))/(LAG(SUM(t.payment_value)) OVER(ORDER BY t.year)) * 100),2) as
percentage_increase
FROM (
      SELECT
        EXTRACT(Year from o.order_purchase_timestamp) as year,
        EXTRACT(Month from o.order_purchase_timestamp) as month,
        p.payment_value
      FROM `Target.orders` as o
      INNER JOIN `Target.payments` as p
      ON o.order_id = p.order_id) t
WHERE (t.year=2017 OR t.year=2018) AND (t.month BETWEEN 1 and 8)
GROUP BY t.year
ORDER BY t.year
```

Output:

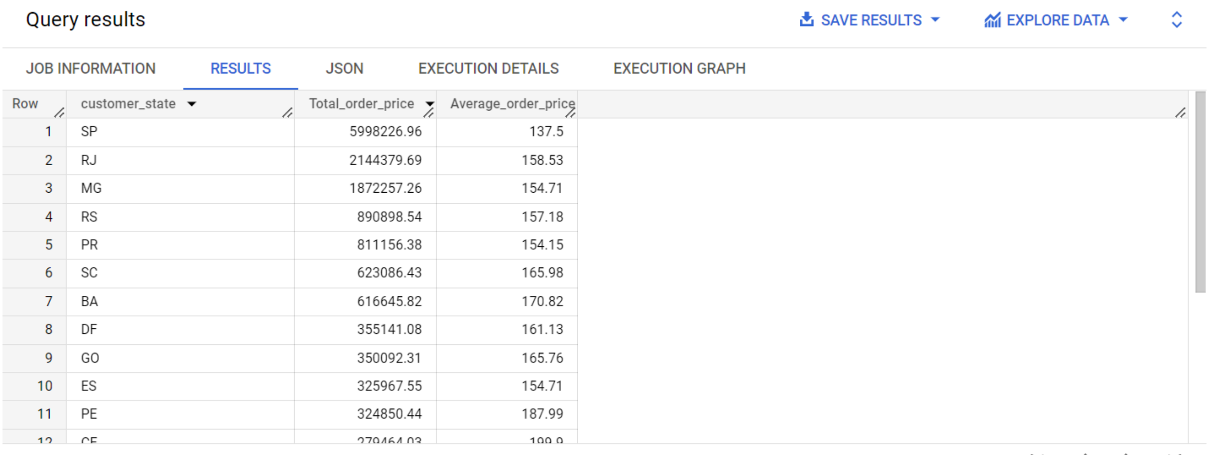

Query results

Row	year	cost_of_orders	percentage_increase
1	2017	3669022.119999...	null
2	2018	8694733.839999...	136.98

Insights:
The percentage increase in the cost of orders reflects the revenue growth of orders for the period 2017 to 2018.
There is an increase of 136.98% in cost of orders from the year 2017 to 2018. This could be due to various reasons including market demand, profitability, market competition, new additions in the product etc.
Important thing to consider here is the customers' perspective. It is very important to find out if the customers will find this increase reasonable and if they are willing to pay the increased price.

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

Query:
```
SELECT
  c.customer_state,
  ROUND(SUM(p.payment_value),2) as Total_order_price,
  ROUND(AVG(p.payment_value),2) as Average_order_price
FROM `Target.payments` as p
INNER JOIN `Target.orders` as o
ON p.order_id = o.order_id
INNER JOIN `Target.customers` as c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY  Total_order_price DESC,Average_order_price DESC
```

Output:

Query results

Row	customer_state	Total_order_price	Average_order_price
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71
11	PE	324850.44	187.99
12	CE	279464.03	199.9

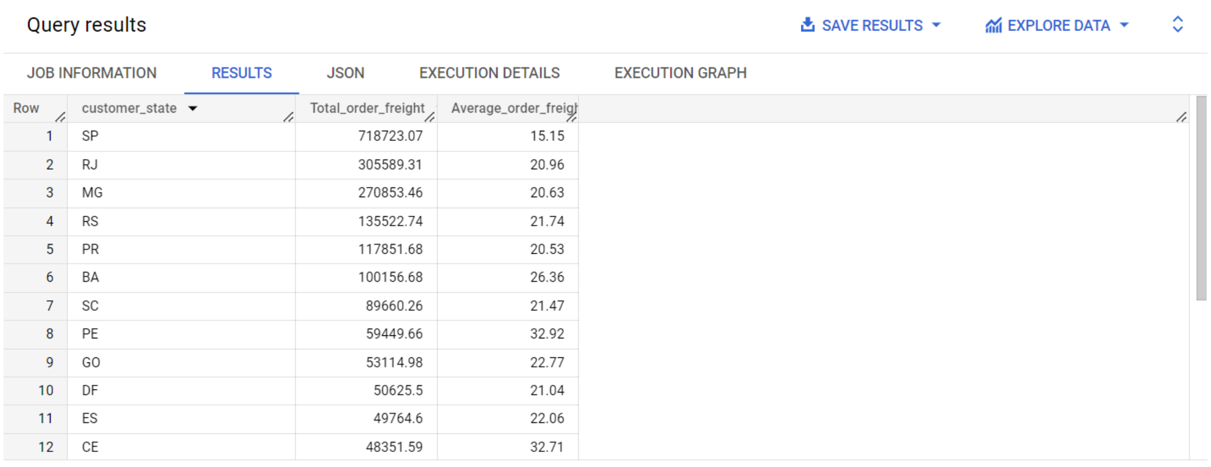
Insights:
Analysing the total and average value of order prices for each state will help in identifying the regions where the customers are willing to spend more or less on their orders placed. This will help in understanding the customers' behaviour. The above data is sorted in descending order for both total and average order price for each state.

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

Query:

```
SELECT
  c.customer_state,
  ROUND(SUM(oi.freight_value),2) as Total_order_freight,
  ROUND(AVG(oi.freight_value),2) as Average_order_freight
FROM  `Target.order_items` as oi
INNER JOIN `Target.orders` as o
ON oi.order_id = o.order_id
INNER JOIN `Target.customers` as c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY Total_order_freight DESC,Average_order_freight DESC
```

Output:



Query results

Row	customer_state	Total_order_freight	Average_order_freig
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
11	ES	49764.6	22.06
12	CE	48351.59	32.71

Insights:
The total and average freight value helps us in analysing the overall cost spent in shipping and delivering the order to its intended destination. If the freight cost is higher like for the state SP, some steps need to be implemented like revision or negotiation of prices with warehouse and shipping partners. The above data is sorted in descending order for both total and average order freight for each state.

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.

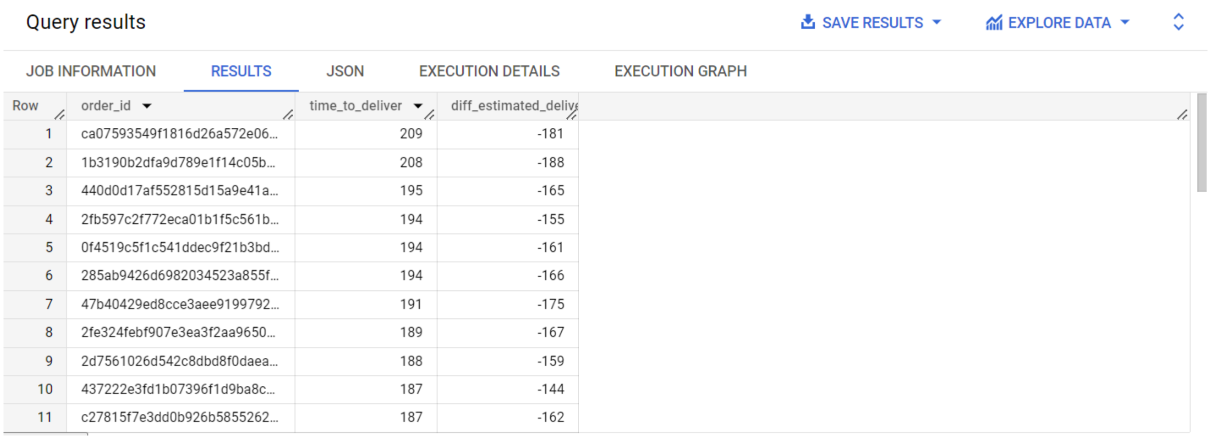
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

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

Query:

```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp, DAY) as
time_to_deliver,
  DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,
DAY) as diff_estimated_delivery
FROM  `Target.orders`
WHERE order_delivered_customer_date IS NOT NULL
ORDER BY time_to_deliver DESC, diff_estimated_delivery DESC
```

Output:



Row	order_id	time_to_deliver	diff_estimated_delive
1	ca07593549f1816d26a572e06...	209	-181
2	1b3190b2dfa9d789e1f14c05b...	208	-188
3	440d0d17af552815d15a9e41a...	195	-165
4	2fb597c2f772eca01b1f5c561b...	194	-155
5	0f4519c5f1c541ddec9f21b3bd...	194	-161
6	285ab9426d6982034523a855f...	194	-166
7	47b40429ed8cce3aee9199792...	191	-175
8	2fe324febf907e3ea3f2aa9650...	189	-167
9	2d7561026d542c8dbd8f0daea...	188	-159
10	437222e3fd1b07396f1d9ba8c...	187	-144
11	c27815f7e3dd0b926b5855262...	187	-162

Insights:

This query and data set can provide us with several crucial insights. We get to know about the number of days taken for each order to get delivered. This helps in analysing the delays in the delivery process. Customers expect and appreciate timely delivery and delays in delivery can lead to dissatisfaction.

Also, when the delivery date is compared with estimated delivery date, we can analyse if the order was delivered before or after the suggested estimated delivery. We have calculated:
diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date
which is coming out to be negative.

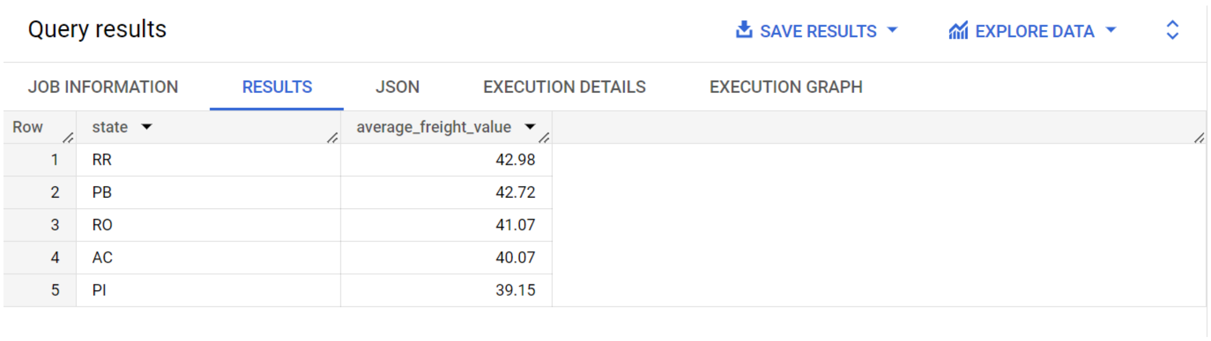
2. **Find out the top 5 states with the highest & lowest average freight value.**

Query:

Top 5 states with highest average freight value:

```
SELECT
    c.customer_state AS state,
    ROUND(AVG(oi.freight_value),2) AS average_freight_value
FROM `Target.orders` AS o
INNER JOIN `Target.order_items` AS oi
ON o.order_id = oi.order_id
INNER JOIN `Target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY average_freight_value DESC
LIMIT 5
```

Output:

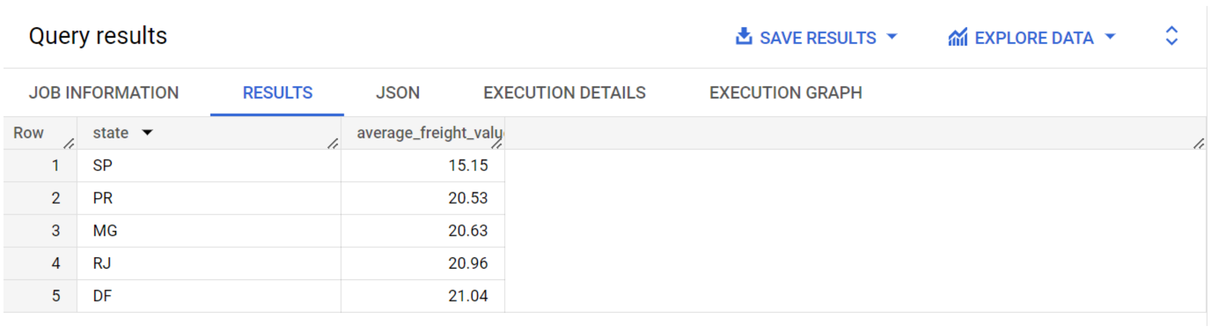


Row	state	average_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Top 5 states with lowest average freight value:

```
SELECT
    c.customer_state AS state,
    ROUND(AVG(oi.freight_value),2) AS average_freight_value
FROM `Target.orders` AS o
INNER JOIN `Target.order_items` AS oi
ON o.order_id = oi.order_id
INNER JOIN `Target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY average_freight_value ASC
LIMIT 5
```

Output:



Row	state	average_freight_valu
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

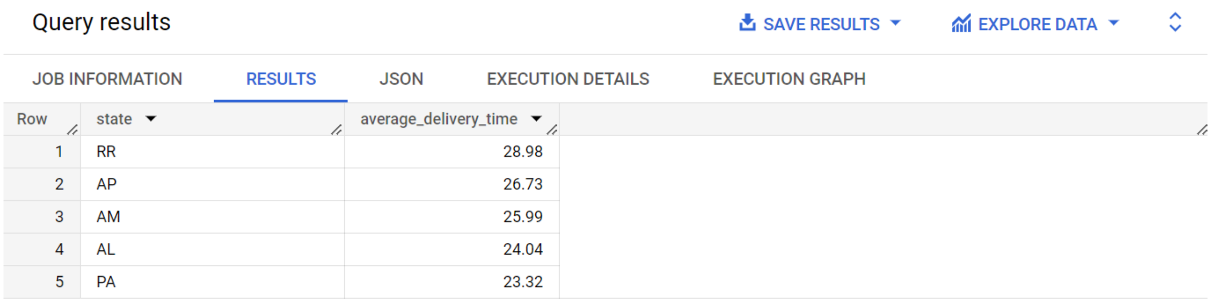
Insights:
From this analysis, we have the top 5 states with the highest and lowest average order freight value. We can study the shipping expenses and the strategies behind these. This will help in implementing the more advantageous cost distribution for the states with higher average order freight values.

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

Query:
Top 5 states with highest average delivery time:

```
SELECT
    c.customer_state AS state,
    ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) AS average_delivery_time
FROM `Target.orders` AS o
JOIN `Target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY average_delivery_time DESC
LIMIT 5
```

Output:

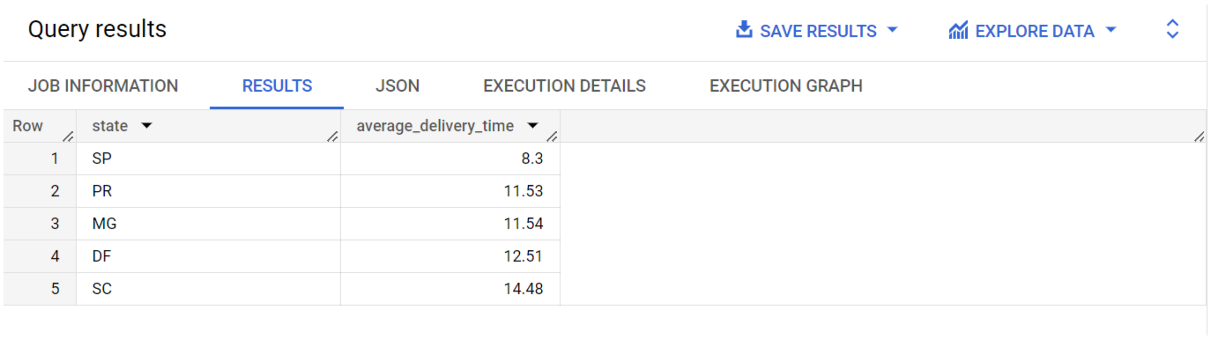


Row	state	average_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Query:
Top 5 states with lowest average delivery time:

```
SELECT
    c.customer_state AS state,
    ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp, DAY)),2) AS average_delivery_time
FROM `Target.orders` AS o
JOIN `Target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY state
ORDER BY average_delivery_time ASC
LIMIT 5
```

Output:



Row	state	average_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

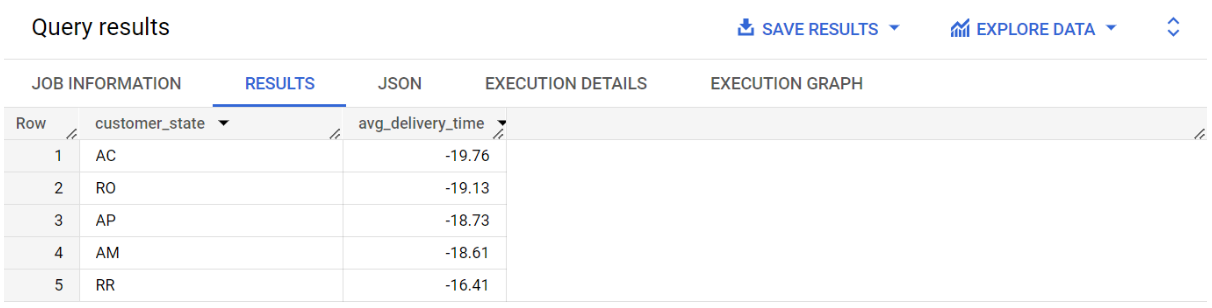
Insights:
From this analysis, we have the top 5 states with the highest and lowest average delivery time in days. As can be seen from the data, the least average delivery time is taken by the state SP and the highest average delivery time is taken by the state RR.

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:

```
SELECT
  c.customer_state,
  ROUND(AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)),2) AS avg_delivery_time
FROM `Target.orders` as o
JOIN `Target.customers` as c
ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY avg_delivery_time ASC
LIMIT 5
```

Output:



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

Insights:

We have calculated the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

The average delivery time is negative in the obtained data. A negative value indicates faster delivery. It is because in the query, it is calculated using the difference of delivery date and estimated delivery date.

This means that on average, the orders in those states are delivered earlier than the estimated delivery date.

6. Analysis based on the payments:

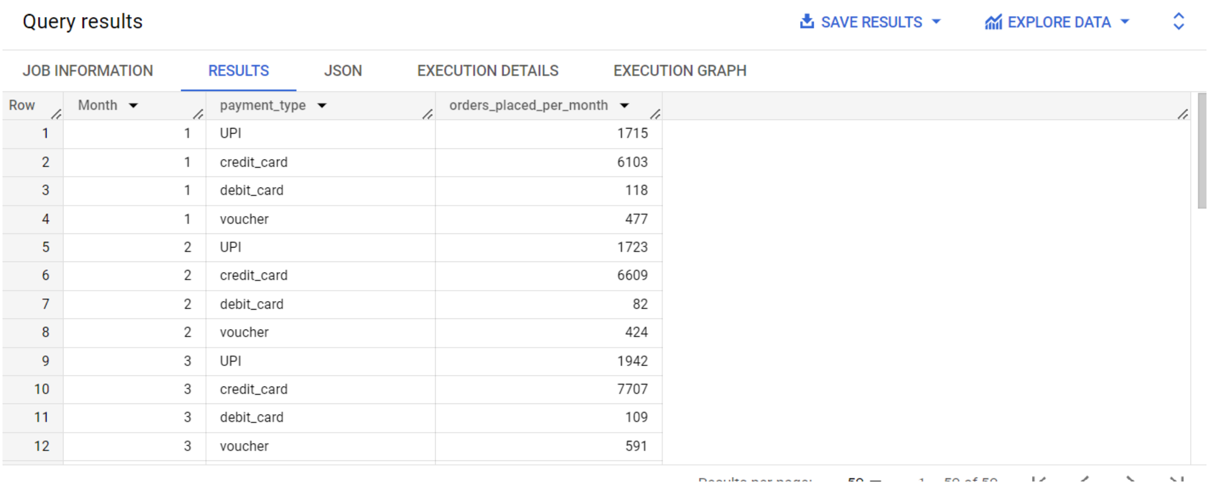
1. **Find the month on month no. of orders placed using different payment types.**

   Query:

   Approach 1: Calculating the month on month number of orders placed using different payment types irrespective of the year

```
SELECT
  EXTRACT(MONTH from order_purchase_timestamp) AS Month,
  p.payment_type,
  COUNT(*) as orders_placed_per_month
FROM `Target.payments` as p
INNER JOIN `Target.orders` as o
ON p.order_id = o.order_id
INNER JOIN `Target.customers` as c
ON o.customer_id = c.customer_id
GROUP BY Month,p.payment_type
ORDER BY Month, p.payment_type
```

   Output:

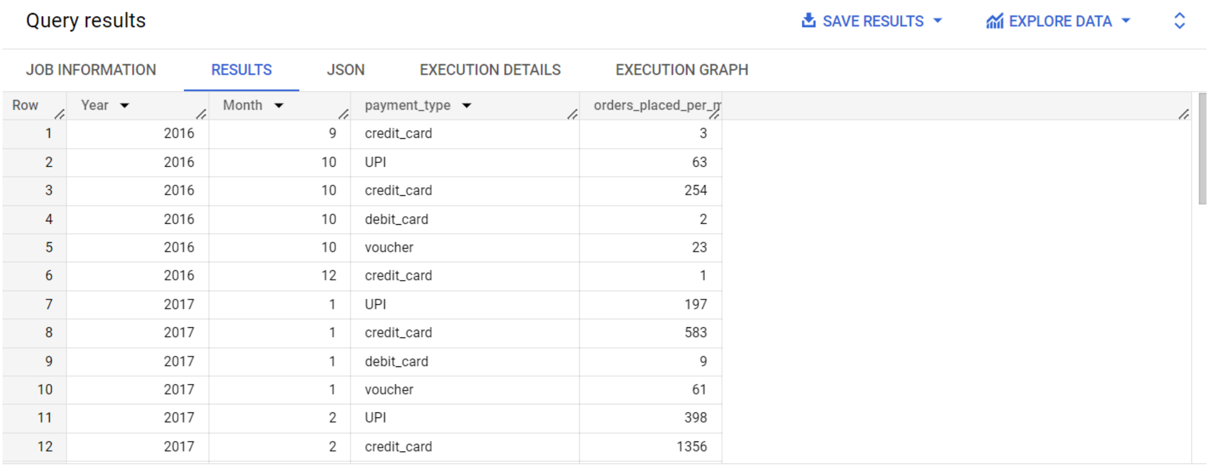


Row	Month	payment_type	orders_placed_per_month
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707
11	3	debit_card	109
12	3	voucher	591

Approach 2: Calculating the month on month number of orders placed using different payment types each year

```sql
SELECT
  EXTRACT(YEAR from order_purchase_timestamp) AS Year,
  EXTRACT(MONTH from order_purchase_timestamp) AS Month,
  p.payment_type,
  COUNT(*) as orders_placed_per_month
FROM `Target.payments` as p
INNER JOIN `Target.orders` as o
ON p.order_id = o.order_id
INNER JOIN `Target.customers` as c
ON o.customer_id = c.customer_id
GROUP BY Year,Month,p.payment_type
ORDER BY Year,Month, p.payment_type
```

Output



Row	Year	Month	payment_type	orders_placed_per_m
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

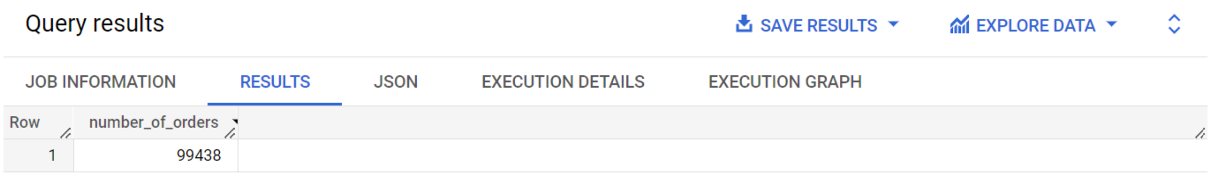
Insights:
Analysing the month on month number of orders placed using different payment modes like credit card, debit card, UPI, vouchers gives insights such as the payment type preferred by customers, payment trends, seasonal trend or partnerships.
From the obtained data, it can be seen that the number of transactions done using UPI is increasing significantly. This implies the preference of customers due to the ease in payment process.
It should also be noted that the transactions made from credit cards and vouchers are also increasing from month to month. This could be significantly due to the offers and discounts received in these modes of payments.

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

   Approach 1: Calculating the overall number or orders irrespective of the payment installments type

   Query:
```
SELECT
  count(DISTINCT order_id) as number_of_orders
FROM `Target.payments`
WHERE payment_installments >=1
```

   Output:



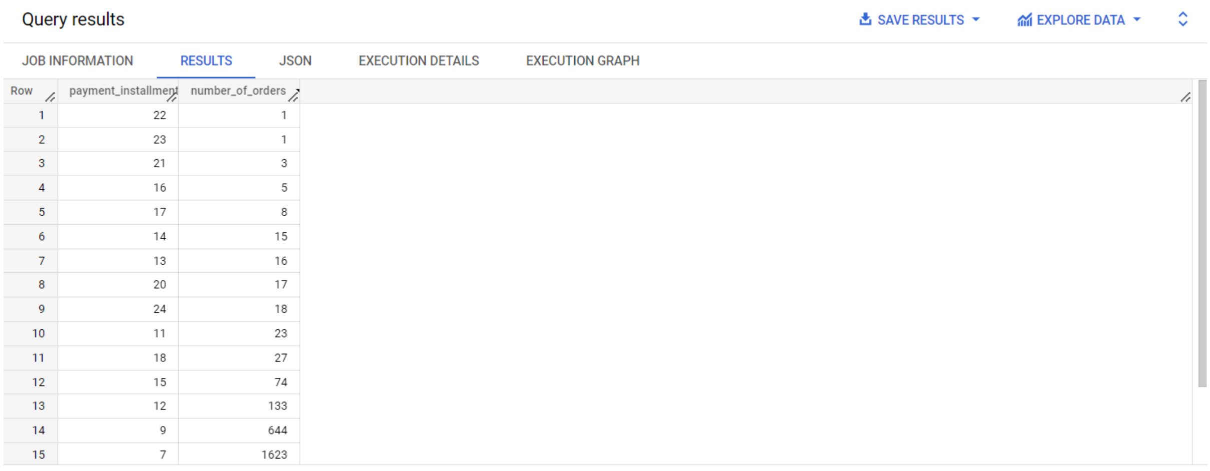
Row	number_of_orders
1	99438

Insights:
The above data gives us the number of orders that have been paid on the basis of payment installments. The output is coming out to be 99438.

Approach 2: Calculating the overall number or orders for each payment installments type

Query:
```
SELECT
  payment_installments,
  count(DISTINCT order_id) as number_of_orders
FROM `Target.payments`
WHERE payment_installments >=1
GROUP BY payment_installments
ORDER BY number_of_orders
```

Output:



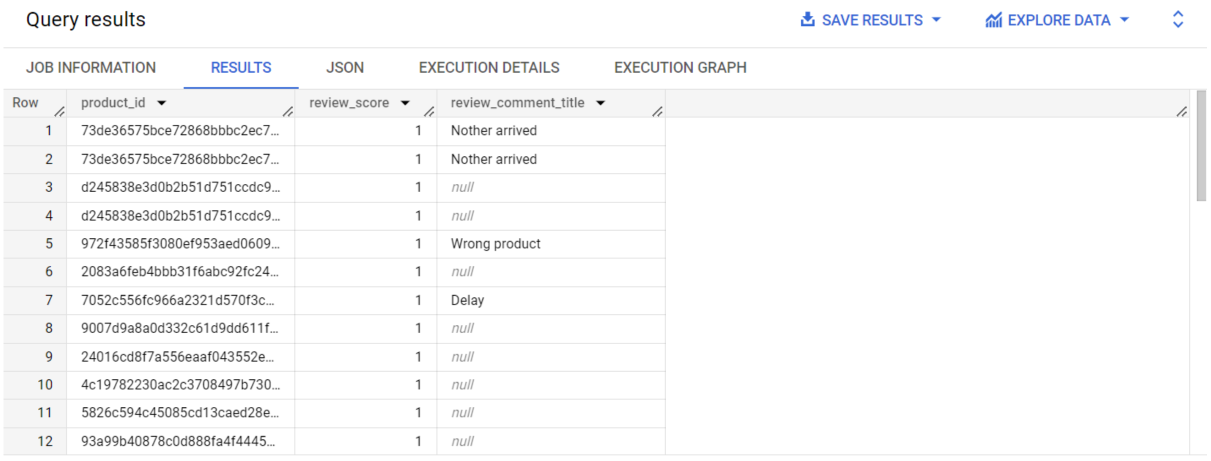
Row	payment_installmen	number_of_orders
1	22	1
2	23	1
3	21	3
4	16	5
5	17	8
6	14	15
7	13	16
8	20	17
9	24	18
10	11	23
11	18	27
12	15	74
13	12	133
14	9	644
15	7	1623

What is the review given by the customer for each product?

Query:

```
SELECT
  oi.product_id,
  o.review_score,
  o.review_comment_title
FROM `Target.order_reviews` as o
JOIN `Target.order_items` as oi
ON o.order_id = oi.order_id
ORDER BY o.review_score
```

Output:



Row	product_id	review_score	review_comment_title
1	73de36575bce72868bbbc2ec7...	1	Nother arrived
2	73de36575bce72868bbbc2ec7...	1	Nother arrived
3	d245838e3d0b2b51d751ccdc9...	1	null
4	d245838e3d0b2b51d751ccdc9...	1	null
5	972f43585f3080ef953aed0609...	1	Wrong product
6	2083a6feb4bbb31f6abc92fc24...	1	null
7	7052c556fc966a2321d570f3c...	1	Delay
8	9007d9a8a0d332c61d9dd611f...	1	null
9	24016cd8f7a556eaaf043552e...	1	null
10	4c19782230ac2c3708497b730...	1	null
11	5826c594c45085cd13caed28e...	1	null
12	93a99b40878c0d888fa4f4445...	1	null

Insights:

The above data represents the review score and review given by the respective customers for every product. Review score can be from 1 to 5, 1 being the lowest.

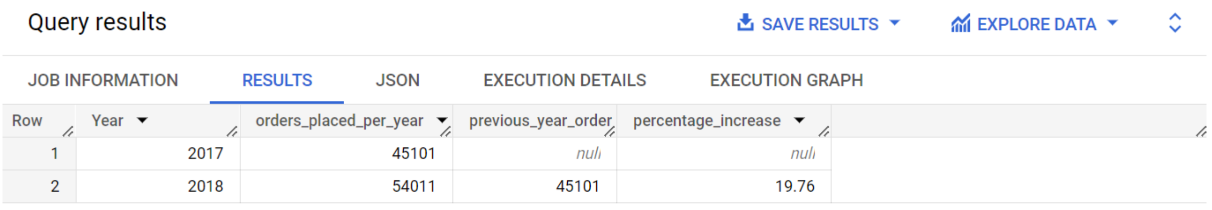
We can also calculate the percentage increase in the number of orders placed from the year 2017 to 2018.

Query:

```
SELECT
  t.Year,
  t.orders_placed_per_year,
  (LAG(t.orders_placed_per_year) OVER(ORDER BY t.Year)) as
previous_year_order,
  ROUND(((t.orders_placed_per_year - (LAG(t.orders_placed_per_year)
OVER(ORDER BY t.Year)))/ (LAG(t.orders_placed_per_year) OVER(ORDER BY
t.Year)) * 100),2) as percentage_increase
FROM (
    SELECT
      EXTRACT(YEAR from order_purchase_timestamp) AS Year,
      COUNT(*) as orders_placed_per_year
    FROM `Target.orders`
        GROUP BY Year

) t
WHERE t.Year = 2017 OR t.Year = 2018
ORDER BY t.Year
```

Output:



Query results

Row	Year	orders_placed_per_year	previous_year_order	percentage_increase
1	2017	45101	null	null
2	2018	54011	45101	19.76

Insights:

The above output gives the percentage increase per year. There is an increase in the number of orders by 19.76% from the year 2017 to the year 2018.