**CASE STUDY- TARGET SQL**

**- By Naman Agarwal**

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

1. **Data type of columns in a table – orders**

After analysis, the datatype of the columns in the table, orders are as follows-

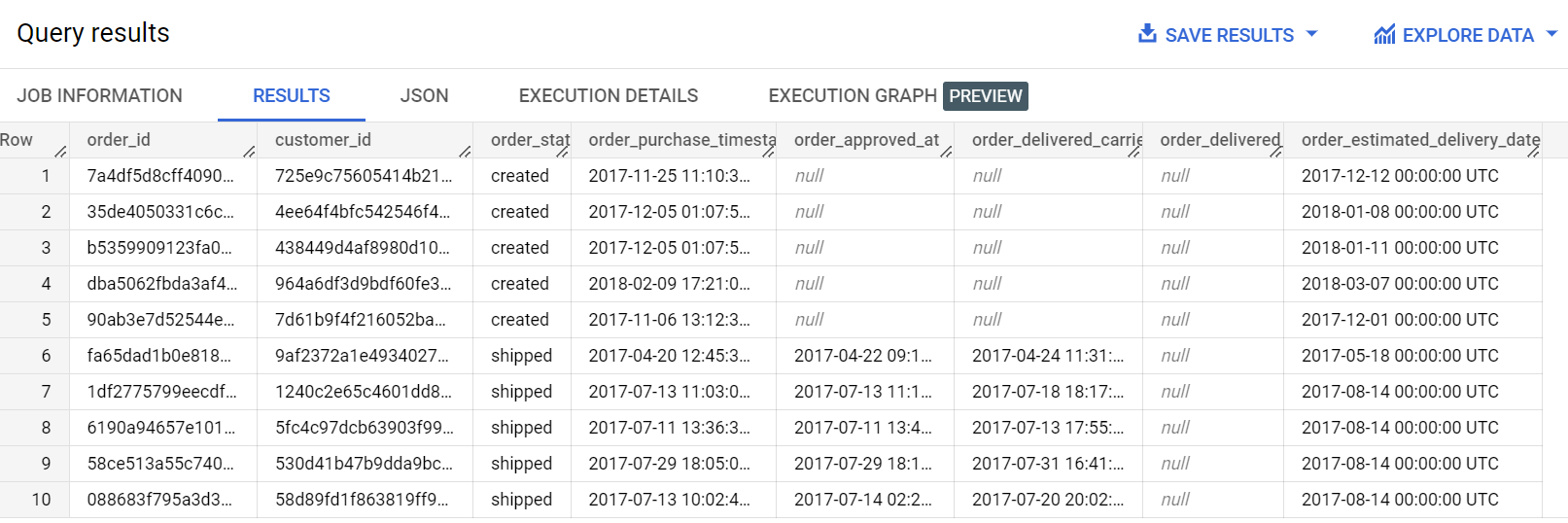
For order\_id, customer\_id, order\_status – the data type is VARCHAR string data type, means it can have variable length of characters.

For order\_purchase\_timestamp, order\_approved\_at, Level1order\_delivered\_carrier\_date, order\_delivered\_customer\_date, Level 1order\_estimated\_delivery\_date - data type is timestamp, which means it has date and time in the format YYYY-MM-DD hh:mm:ss .

SELECT \*

FROM `scaler-380812.target.orders`

LIMIT 10



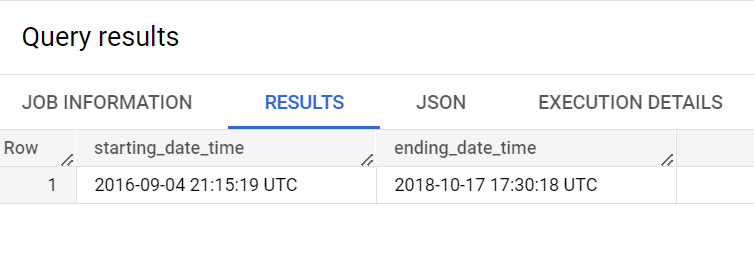
1. **Time period for which the data is given**

SELECT

  MIN(order\_purchase\_timestamp) AS starting\_date\_time,

  MAX(order\_purchase\_timestamp) AS ending\_date\_time

FROM `scaler-380812.target.orders`



By using min and max functions we can find the time period of the dataset given, it is giving first order date and last order date for which the dataset is given.

1. **Cities and States of customers ordered during the given period**

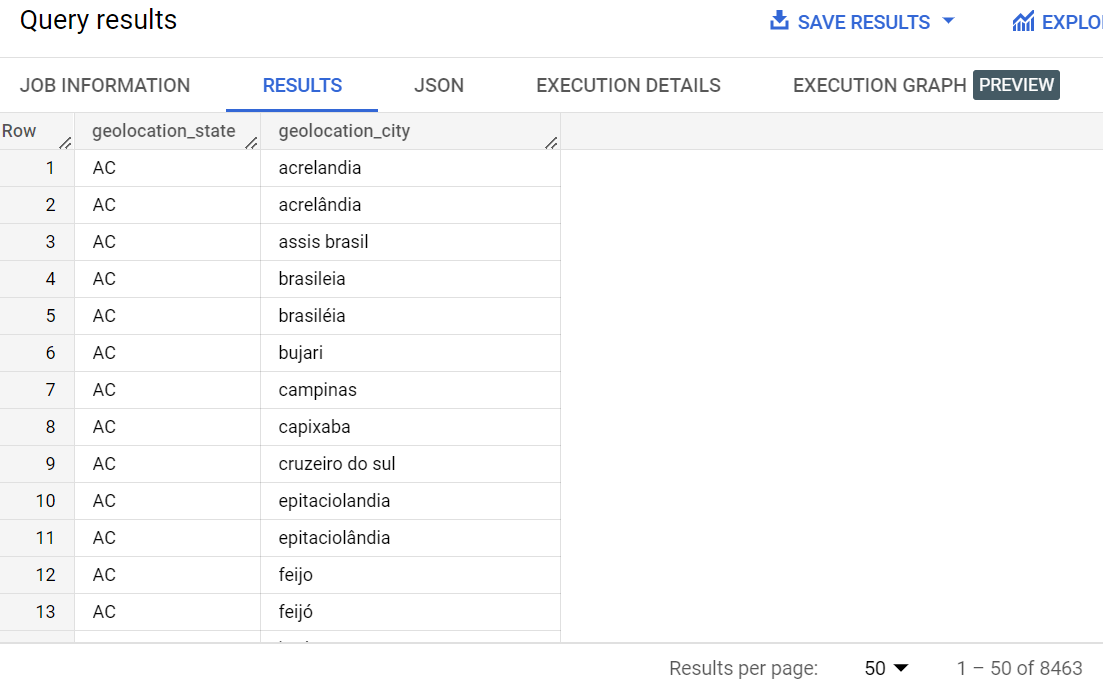
SELECT DISTINCT

  geolocation\_state,

  geolocation\_city

FROM `scaler-380812.target.geolocation`

ORDER BY geolocation\_state ASC, geolocation\_city ASC



By using distinct, we are finding unique city and state from where customers had ordered, so it is displaying output in such a way that for each state it is giving all the cities.

**2. In-depth Exploration:**

1. **Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?**

**For trend over years-**

SELECT

x.year,

x.total\_sales

FROM

(

SELECT

  EXTRACT(year FROM order\_purchase\_timestamp) AS year,

  ROUND(SUM(payment\_value),3) AS total\_sales

FROM `scaler-380812.target.orders` AS o

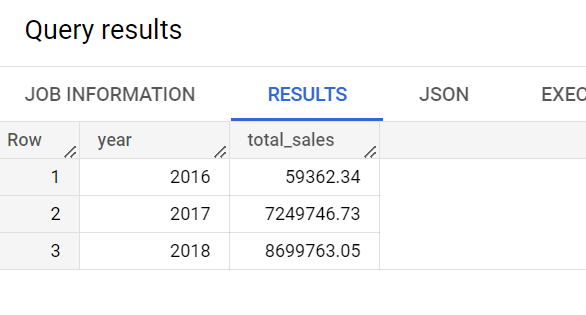
JOIN `scaler-380812.target.payments` AS p

ON o.order\_id = p.order\_id

GROUP BY EXTRACT(year FROM order\_purchase\_timestamp)

) AS x

ORDER BY x.year ASC



We are using group by clause on the year, extracted from the purchase date and using aggregation on payment value to get the total sales of the year.

From the above output it can be seen, there is an increasing trend in the sales, as sales increased from 2017 as compared to 2016 and sales also increased in 2018 as compared to 2017.

**For seasonality on months across years-**

SELECT

x.year,

x.month\_number,

x.total\_sales

FROM

(

SELECT

  EXTRACT(year FROM order\_purchase\_timestamp) AS year,

  EXTRACT(month FROM order\_purchase\_timestamp) AS month\_number,

  ROUND(SUM(payment\_value),3) AS total\_sales

FROM `scaler-380812.target.orders` AS o

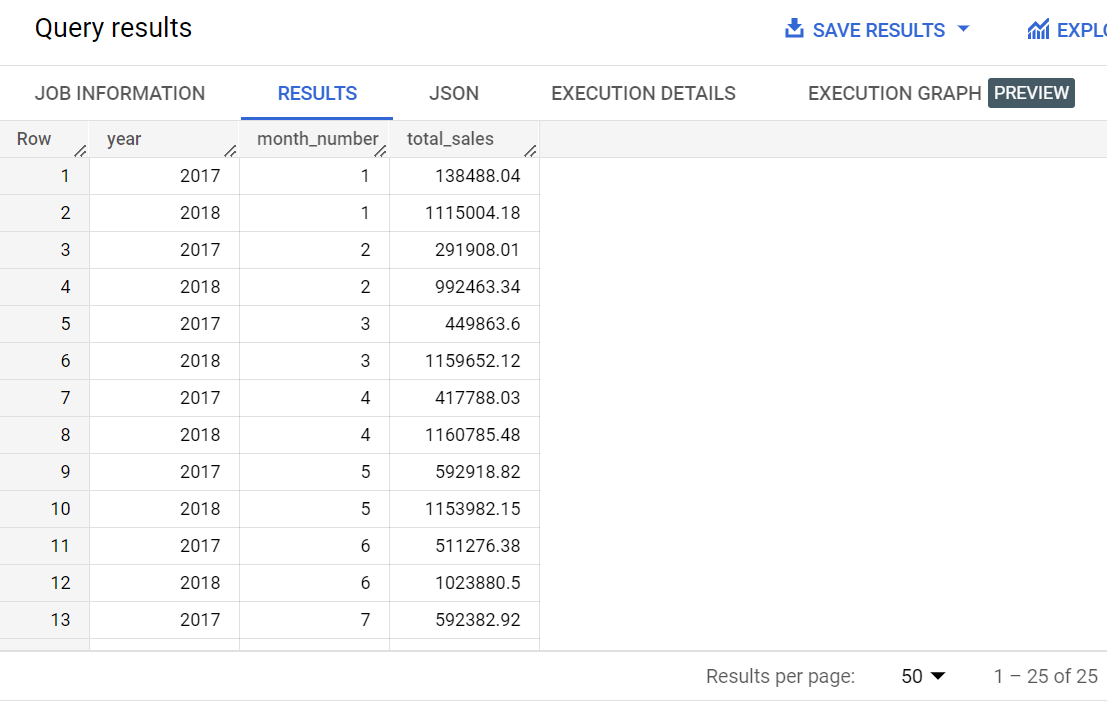
JOIN `scaler-380812.target.payments` AS p

ON o.order\_id = p.order\_id

GROUP BY EXTRACT(year FROM order\_purchase\_timestamp), EXTRACT(month FROM order\_purchase\_timestamp)

) AS x

ORDER BY  x.month\_number ASC , x.year ASC



As can be observed for the data, there is no specific seasonality in months over years where customer buying habits remains same. The sales does not have any specific trend in months over years.

So we can say customers buying habits are varying.

**2. What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?**

SELECT

  x.slots,

  COUNT(x.order\_id) AS no\_of\_orders

FROM

(

SELECT

CASE

WHEN EXTRACT(hour from order\_purchase\_timestamp) BETWEEN 0 AND 6

THEN "DAWN(0-6)"

WHEN EXTRACT(hour from order\_purchase\_timestamp) BETWEEN 7 AND 12

THEN "Mor(7-12)"

WHEN EXTRACT(hour from order\_purchase\_timestamp) BETWEEN 13 AND 18

THEN "Eve(13-18)"

WHEN EXTRACT(hour from order\_purchase\_timestamp) BETWEEN 19 AND 23

THEN "Nig(19-23)"

END AS slots,

order\_id,

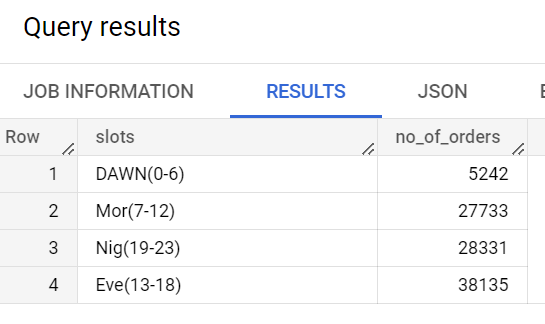
customer\_id

FROM `scaler-380812.target.orders`

) AS x

GROUP BY x.slots

ORDER BY no\_of\_orders



We are using bins to divide the time in 4 time slots and then using group by to group those slots as per count of orders.

As from the output, it can be observed, that in the evening (13 to 18 Hrs), most number of the orders were placed, hence we can say customers in Brazil prefer to buy in evening (13-18) Hrs.

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

**1.Get month on month orders by states**

SELECT

  g.geolocation\_state,

  o.year,

  o.month\_number,

  COUNT(o.order\_id) AS no\_of\_orders

FROM

(

SELECT

  order\_id,

  customer\_id,

  EXTRACT(month FROM order\_purchase\_timestamp) AS month\_number,

  EXTRACT(year FROM order\_purchase\_timestamp) AS year

FROM `scaler-380812.target.orders`

) AS o

JOIN `scaler-380812.target.customers` AS c

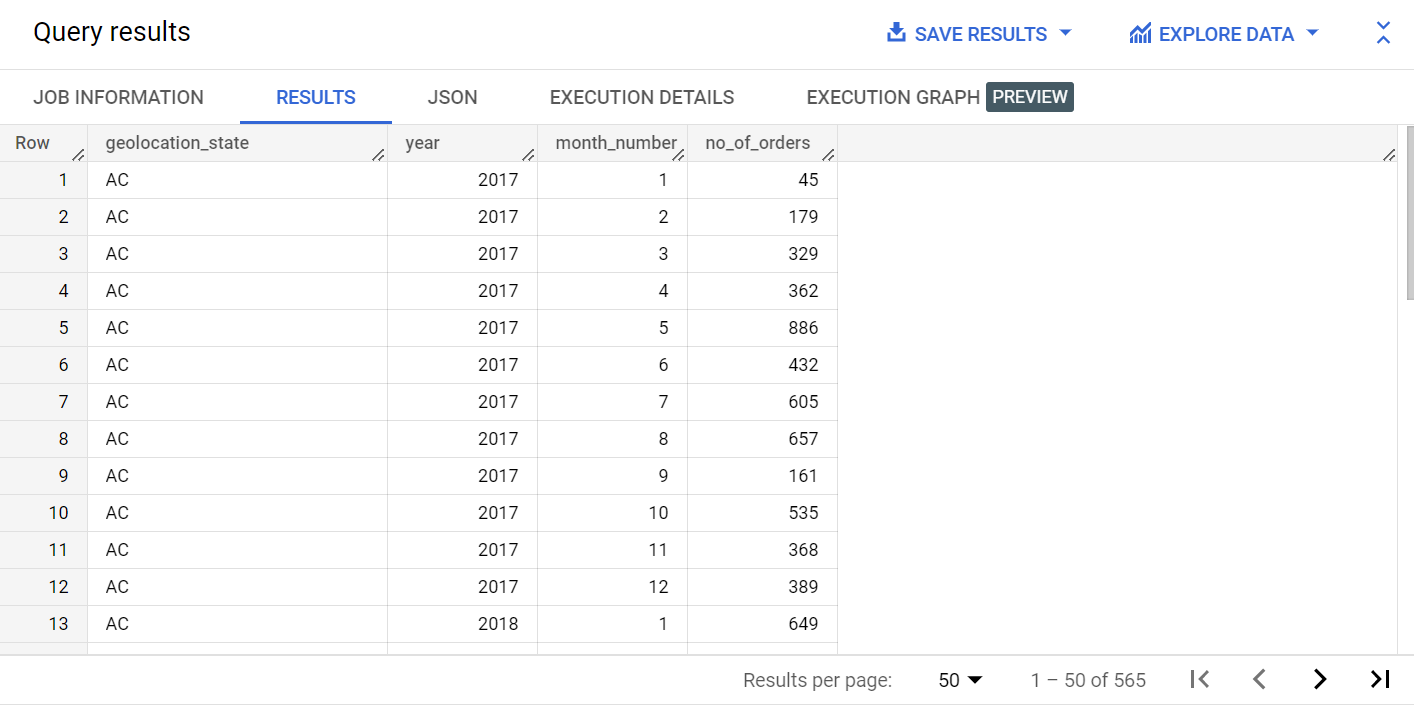
ON o.customer\_id = c.customer\_id

JOIN `scaler-380812.target.geolocation` AS g

ON c.customer\_zip\_code\_prefix = g.geolocation\_zip\_code\_prefix

GROUP BY  g.geolocation\_state , month\_number , year

ORDER BY  g.geolocation\_state ASC, year ASC, month\_number ASC



Here, we grouped the data on basis of state, month and year and getting aggregation on count of orders placed in each month, every year. So, it can be observed from above data the number of orders placed in each state in every month and in each year.

**2.Distribution of customers across the states in Brazil**

SELECT

  g.geolocation\_state,

  COUNT(DISTINCT c.customer\_id) AS no\_of\_customers

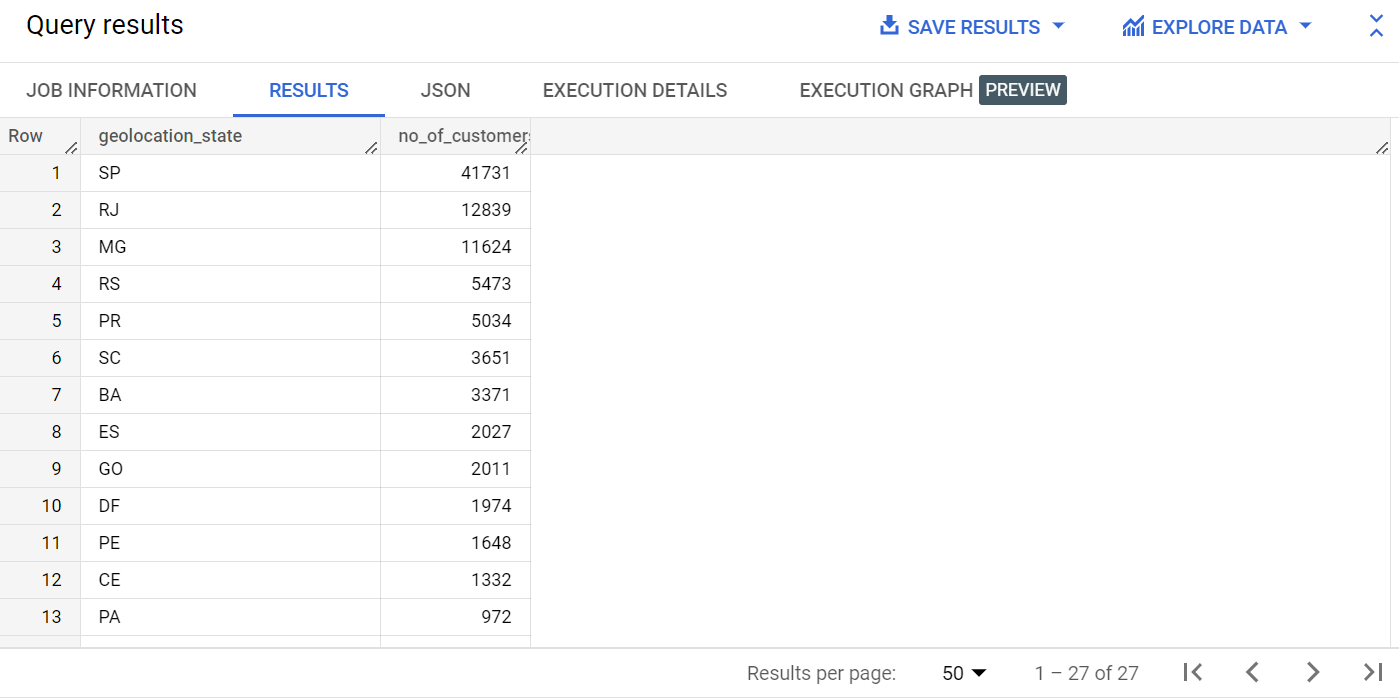
FROM `scaler-380812.target.customers` AS c

JOIN `scaler-380812.target.geolocation` AS g

ON c.customer\_zip\_code\_prefix = g.geolocation\_zip\_code\_prefix

GROUP BY  g.geolocation\_state

ORDER BY  no\_of\_customers DESC



We are grouping states and getting aggregation on count of distinct customer IDs to get count of unique customers in each state. From the above, it can be observed that state SP has highest number of customers

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

SELECT

  g.geolocation\_state,

  ROUND(SUM(oi.price),2) AS price\_of\_the\_product,

  ROUND(SUM(p.payment\_value),2) AS amount\_paid\_by\_customer,

  ROUND(SUM(oi.freight\_value),2) AS freight\_value

FROM `scaler-380812.target.order\_items` AS oi

JOIN  `scaler-380812.target.sellers` AS s

ON oi.seller\_id = s.seller\_id

JOIN `scaler-380812.target.geolocation` AS g

ON s.seller\_zip\_code\_prefix = g.geolocation\_zip\_code\_prefix

JOIN `scaler-380812.target.orders` AS o

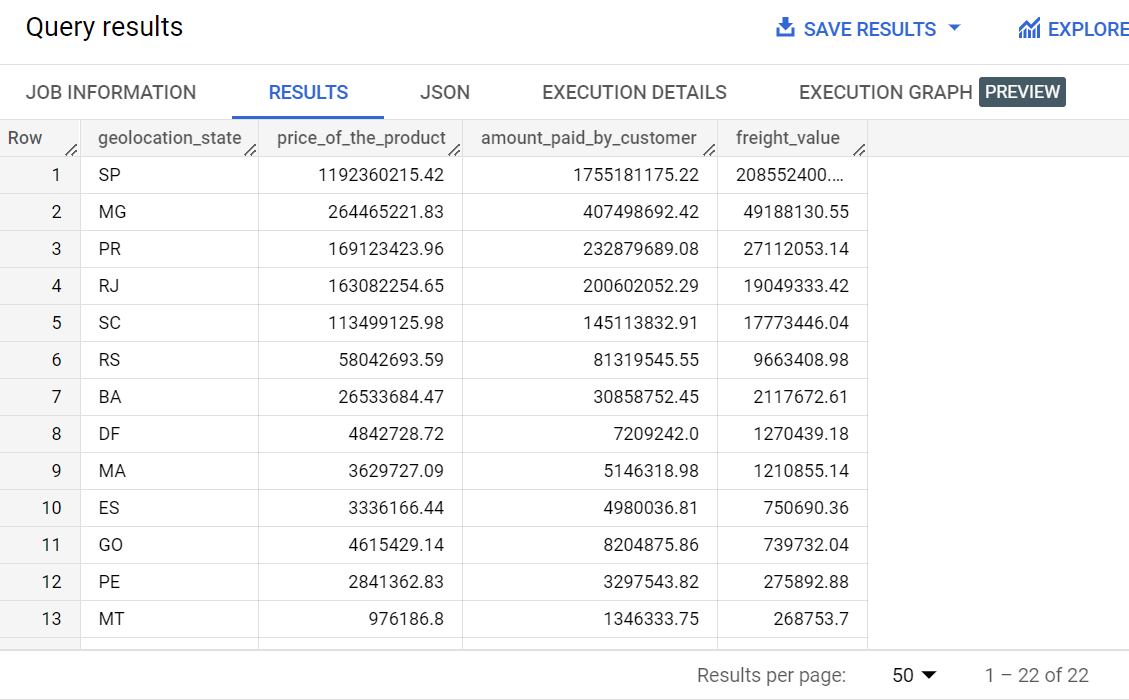
ON o.order\_id = oi.order\_id

JOIN `scaler-380812.target.payments` AS p

ON p.order\_id = o.order\_id

GROUP BY g.geolocation\_state

ORDER BY freight\_value DESC, price\_of\_the\_product DESC , amount\_paid\_by\_customer DESC



So, from the above, it can be seen the monetary value which comprises of the price of the product, amount paid by the customer and the freight value generated in each state is depicted.

Further-

SELECT

x.geolocation\_state,

ROUND((x.price\_of\_the\_product + x.amount\_paid\_by\_customer + x.freight\_value),2) AS total\_money\_transaction

FROM

(

SELECT

  g.geolocation\_state,

  ROUND(SUM(oi.price),2) AS price\_of\_the\_product,

  ROUND(SUM(p.payment\_value),2) AS amount\_paid\_by\_customer,

  ROUND(SUM(oi.freight\_value),2) AS freight\_value

FROM `scaler-380812.target.order\_items` AS oi

JOIN  `scaler-380812.target.sellers` AS s

ON oi.seller\_id = s.seller\_id

JOIN `scaler-380812.target.geolocation` AS g

ON s.seller\_zip\_code\_prefix = g.geolocation\_zip\_code\_prefix

JOIN `scaler-380812.target.orders` AS o

ON o.order\_id = oi.order\_id

JOIN `scaler-380812.target.payments` AS p

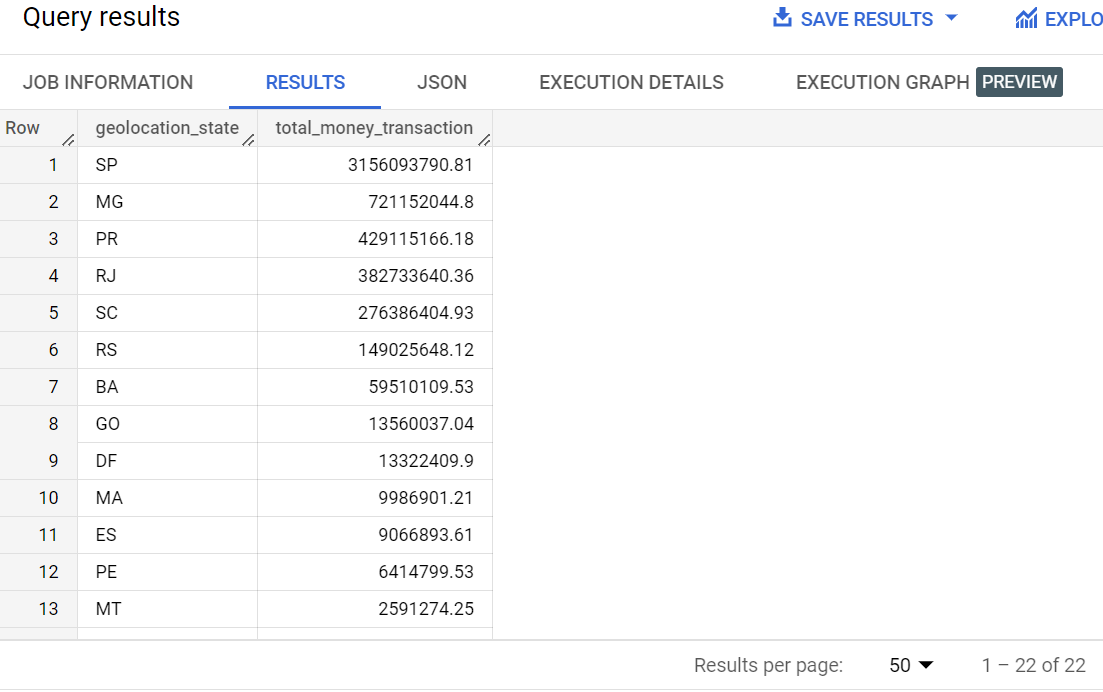
ON p.order\_id = o.order\_id

GROUP BY g.geolocation\_state

ORDER BY freight\_value DESC, price\_of\_the\_product DESC , amount\_paid\_by\_customer DESC

) AS x

ORDER BY total\_money\_transaction DESC



It can be observed,

State SP has the highest monetary transaction or highest monetary generation among all the states.

**1.Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment\_value” column in payments table**

WITH T1 AS

(

SELECT

EXTRACT(year FROM order\_purchase\_timestamp) AS year,

EXTRACT(month FROM order\_purchase\_timestamp) AS month,

order\_id

FROM `scaler-380812.target.orders`

WHERE EXTRACT(year FROM order\_purchase\_timestamp) IN (2017,2018) AND EXTRACT(month FROM order\_purchase\_timestamp) IN(1,2,3,4,5,6,7,8)

)

SELECT

y.year,

y.total\_sales,

y.prev\_year\_sales,

ROUND(((y.total\_sales - y.prev\_year\_sales)/y.prev\_year\_sales)\*100,2) AS percentage\_increase\_in\_sales

FROM

(

SELECT

x.year,

x.total\_sales,

LAG(x.total\_sales,1) OVER(ORDER BY x.year ASC) AS prev\_year\_sales

FROM

(

SELECT

T1.year,

ROUND(SUM(p.payment\_value),2) AS total\_sales

FROM T1

JOIN `scaler-380812.target.payments` AS p

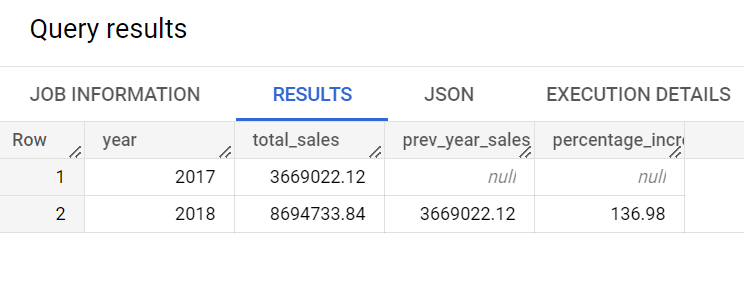
ON T1.order\_id = p.order\_id

GROUP BY T1.year

) AS x

) AS y

ORDER BY y.year ASC



AS from above data, it can be observed that there is growth in total sales in year 2018 as compared to year 2017 for the months from January to August. Now further to we had calculated the percentage increase in sales which is around 136.98 %.

**2.Mean & Sum of price and freight value by customer state.**

SELECT

  g.geolocation\_state,

  ROUND(SUM(oi.price),2) AS sum\_of\_price,

  ROUND(AVG(oi.price),2) AS mean\_of\_price,

  ROUND(SUM(oi.freight\_value),2) AS sum\_of\_freight\_value,

  ROUND(AVG(oi.freight\_value),2) AS mean\_of\_freight\_value

FROM `scaler-380812.target.geolocation` AS g

JOIN `scaler-380812.target.sellers` AS s

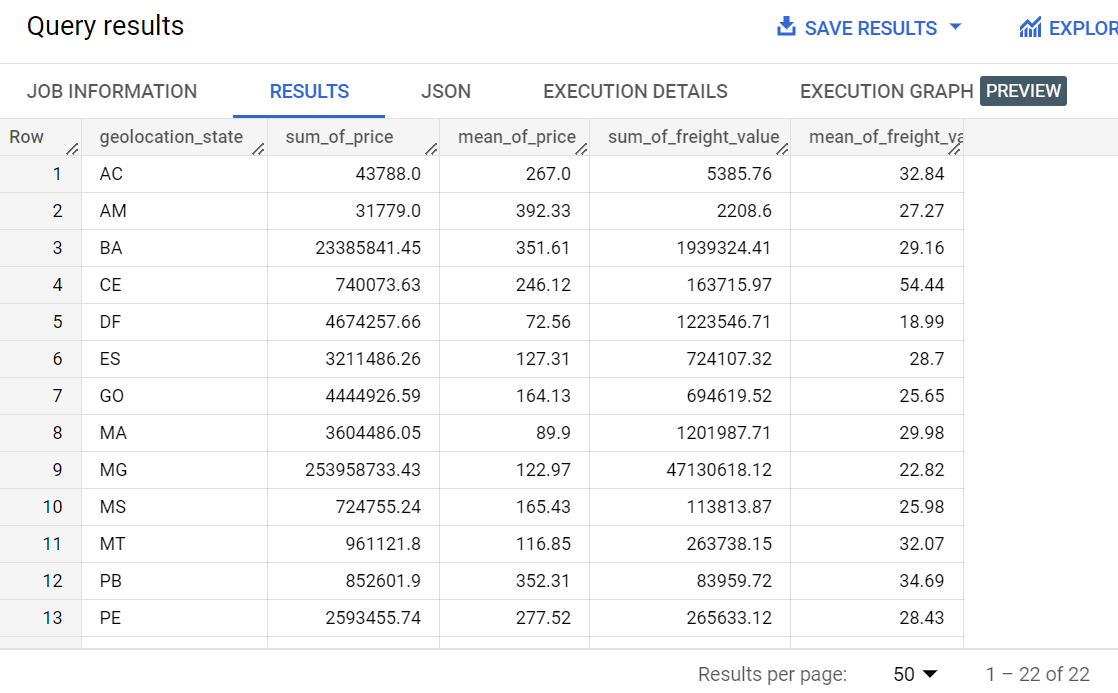
ON g.geolocation\_zip\_code\_prefix = s.seller\_zip\_code\_prefix

JOIN `scaler-380812.target.order\_items` AS oi

ON s.seller\_id = oi.seller\_id

GROUP BY g.geolocation\_state

ORDER BY g.geolocation\_state ASC



We are grouping the states and using aggregation function to calculate sum and mean of the price and freight value.

The above output is displaying the sum of price and average of the price for each state. It is also showing the total freight value and avg freight value for each state.

**5.Analysis on sales, freight and delivery time**

1. **Calculate days between purchasing, delivering and estimated delivery**

SELECT

x.order\_id,

x.order\_status,

DATE\_DIFF(x.delivered\_date, x.purchase\_date , day) AS day\_diff\_purchase\_delivered,

DATE\_DIFF(x.estimated\_delivery\_date, x.purchase\_date, day) AS day\_diff\_purchase\_est\_delivery

FROM

(

SELECT

order\_id,

order\_status,

EXTRACT(date FROM order\_purchase\_timestamp) AS purchase\_date,

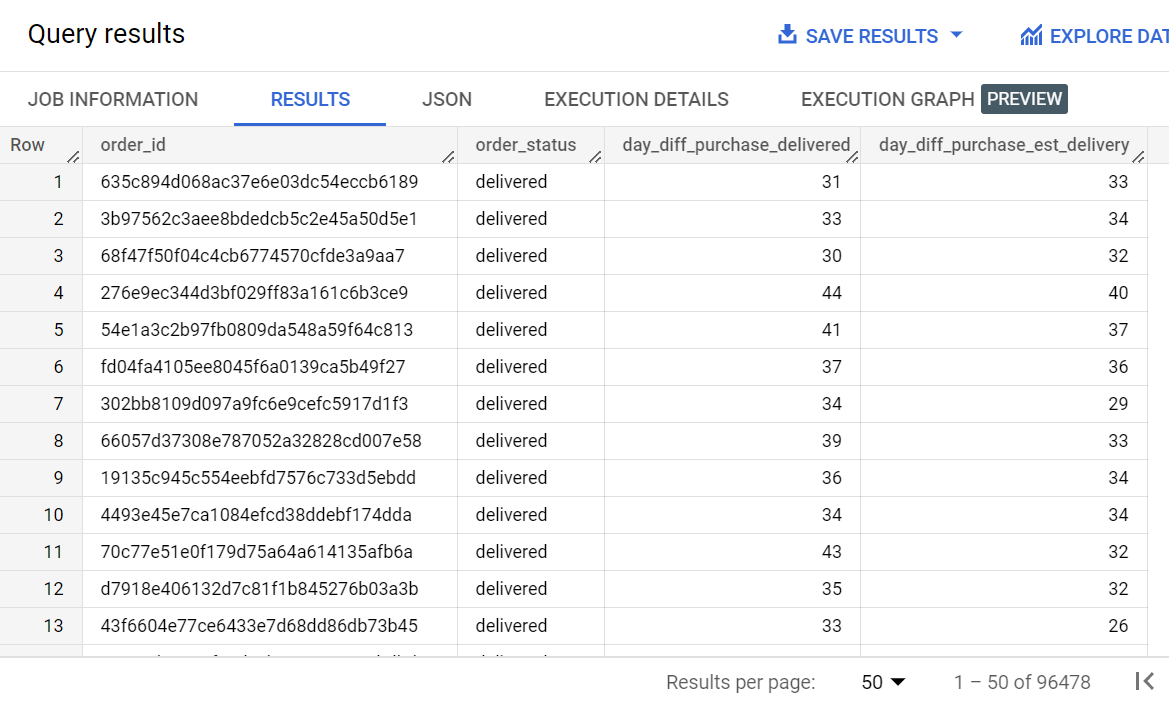
EXTRACT(date FROM order\_delivered\_customer\_date) AS delivered\_date,

EXTRACT(date FROM order\_estimated\_delivery\_date) AS estimated\_delivery\_date

FROM `scaler-380812.target.orders`

) AS x

WHERE x.order\_status = "delivered"



First, we are extracting date part from all the three date columns, then using datediff function, and calculating date difference in form of days between the dates. We are filtering the data to get order\_id for the delivered order only.

The above output displays the difference between delivered and purchase date and estimated delivery date and purchase date for each order.

**2.Find time\_to\_delivery & diff\_estimated\_delivery. Formula for the same given below:**

**time\_to\_delivery = order\_purchase\_timestamp-order\_delivered\_customer\_date**

**diff\_estimated\_delivery = order\_estimated\_delivery\_date-order\_delivered\_customer\_date**

SELECT

x.order\_id,

x.order\_status,

DATE\_DIFF(delivered\_date, purchase\_date , day) AS time\_to\_delivery ,

DATE\_DIFF(delivered\_date, estimated\_delivery\_date, day) AS diff\_estimated\_delivery

FROM

(

SELECT

order\_id,

order\_status,

EXTRACT(date FROM order\_purchase\_timestamp) AS purchase\_date,

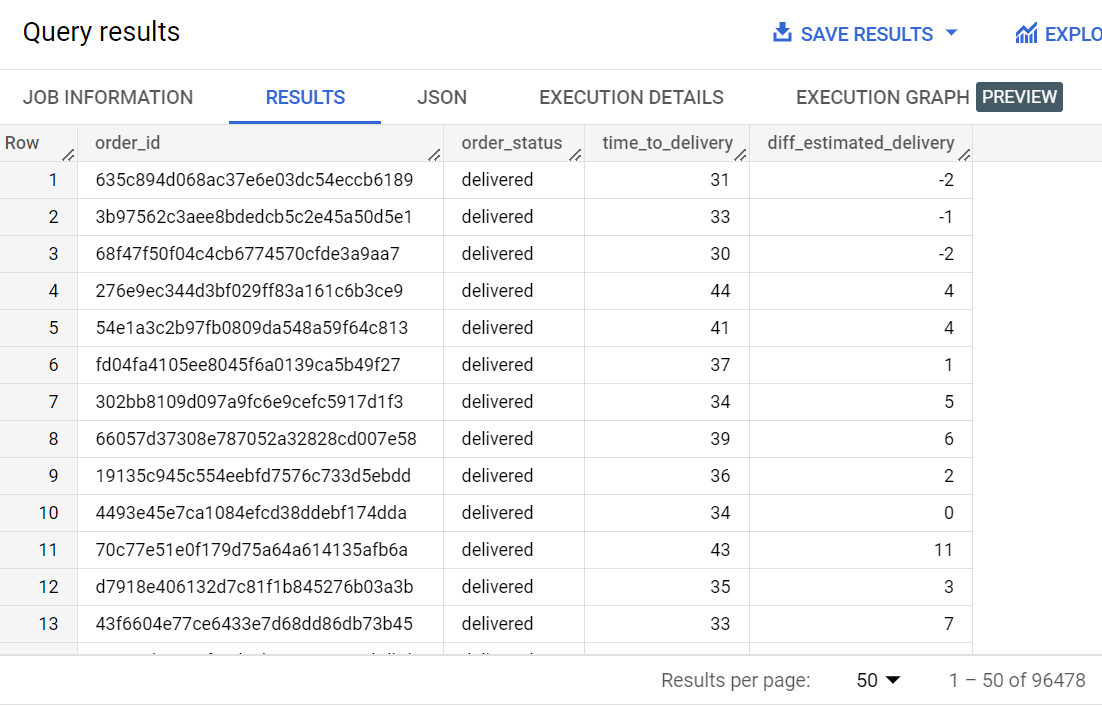
EXTRACT(date FROM order\_delivered\_customer\_date) AS delivered\_date,

EXTRACT(date FROM order\_estimated\_delivery\_date) AS estimated\_delivery\_date

FROM `scaler-380812.target.orders`

)AS x

WHERE x.order\_status = "delivered"



We are calculating time difference between purchase date and delivered date which is represented by time\_to\_delivery.

Then, we are also calculating diff\_estimated\_delivery, which is difference between estimated delivery date and delivered date.

negative diff\_estimated\_delivery means that the order is delivered before the estimated delivery date, while positive diff\_estimated\_delivery means order is delivered even after the estimated delivery date.

Further we are filtering the orders only for delivered orders, since we are interested in delivered orders.

The above output displays the time to deliver which is difference between delivered date and purchase date and estimated delivery which is the difference between delivered date and estimated delivery date for each order.

**3.Group data by state, take mean of freight\_value, time\_to\_delivery, diff\_estimated\_delivery**

WITH T1 AS

(

SELECT

order\_id,

order\_status,

DATE\_DIFF(delivered\_date, purchase\_date , day) AS time\_to\_delivery ,

DATE\_DIFF(delivered\_date, estimated\_delivery\_date, day) AS diff\_estimated\_delivery

FROM

(

SELECT

order\_id,

order\_status,

EXTRACT(date FROM order\_purchase\_timestamp) AS purchase\_date,

EXTRACT(date FROM order\_delivered\_customer\_date) AS delivered\_date,

EXTRACT(date FROM order\_estimated\_delivery\_date) AS estimated\_delivery\_date

FROM `scaler-380812.target.orders`

)

)

SELECT

g.geolocation\_state,

ROUND(AVG(oi.freight\_value),2) AS mean\_freight\_value,

ROUND(AVG(T1.time\_to\_delivery),2) AS mean\_time\_to\_delivery,

ROUND(AVG(T1.diff\_estimated\_delivery),2) AS mean\_diff\_estimated\_delivery

FROM `scaler-380812.target.geolocation` AS g

JOIN `scaler-380812.target.sellers` AS s

ON g.geolocation\_zip\_code\_prefix = s.seller\_zip\_code\_prefix

JOIN `scaler-380812.target.order\_items` AS oi

ON oi.seller\_id = s.seller\_id

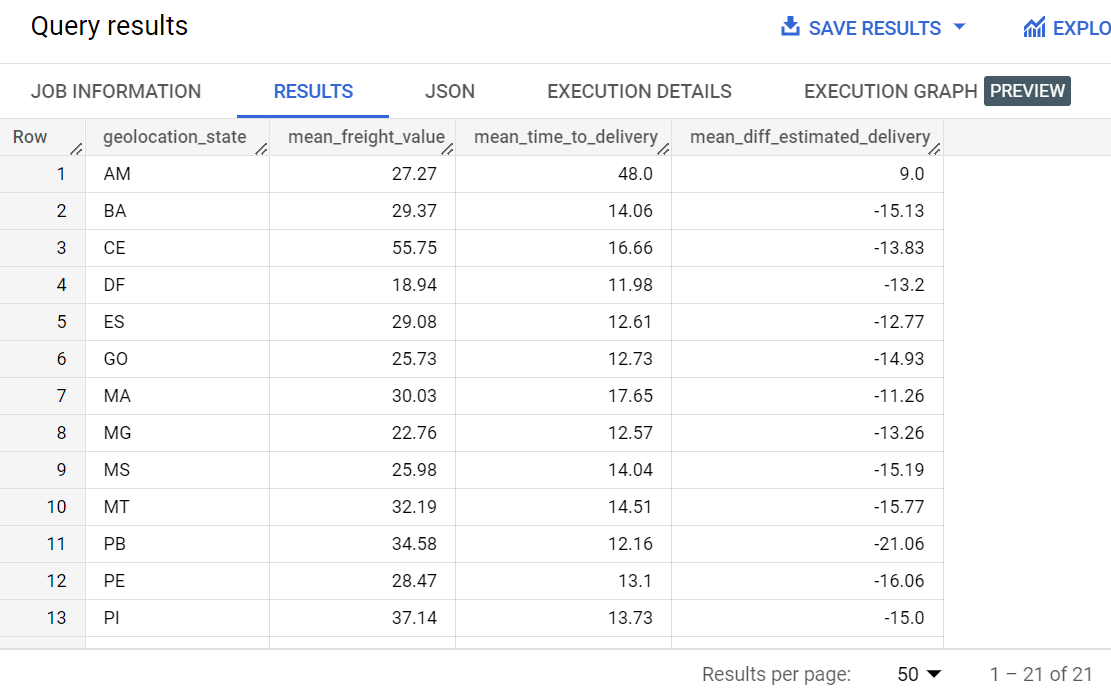
JOIN T1

ON T1.order\_id = oi.order\_id

WHERE T1.order\_status = "delivered"

GROUP BY g.geolocation\_state

ORDER BY g.geolocation\_state



We are first using CTE to get the time\_to\_delivery and diff\_estimated\_delivery

Then using joins to join all the tables to get the required output.

Finally we are using group by on states and using aggregation function AVG() to calculate mean of freight\_value, time\_to\_delivery, diff\_estimated\_delivery.

So for each state the average freight value, average time to deliver and average estimated delivery time is displayed.

Further we are filtering the data only for delivered orders since only for those orders above parameters are applicable.

Negative average estimated delivery means the order is delivered before the estimated delivery date and positive means order is delivered even after the estimated delivery date.

The above output displays the average freight value, average time to deliver and average estimated time to deliver for each state.

**4. Sort the data to get the following:**

1. **Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5**

SELECT

g.geolocation\_state,

ROUND(AVG(freight\_value),2) AS avg\_freight\_value

FROM `scaler-380812.target.geolocation` AS g

JOIN `scaler-380812.target.sellers` AS s

ON g.geolocation\_zip\_code\_prefix = s.seller\_zip\_code\_prefix

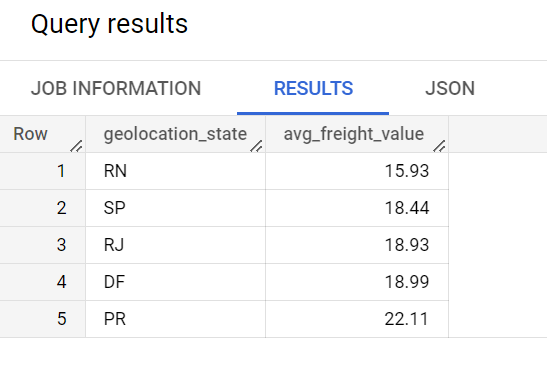
JOIN `scaler-380812.target.order\_items` AS oi

ON oi.seller\_id = s.seller\_id

GROUP BY g.geolocation\_state

ORDER BY avg\_freight\_value ASC

LIMIT 5



We are first joining tables then using group by to group the data on basis of state and using aggregating function AVG to get average freight value.

From the above data, it can be seen the state RN has the lowest average freight value, means in state RN customers had to pay least amount to get the order delivered at their location.

Further, it also shows the top 5 states where average freight value is least.

1. **Top 5 states with highest/lowest average time to delivery**

WITH T1 AS

(

SELECT

Order\_id,

order\_status,

DATE\_DIFF(delivered\_date, purchase\_date , day) AS time\_to\_delivery

FROM

(

SELECT

EXTRACT(date FROM order\_purchase\_timestamp) AS purchase\_date,

EXTRACT(date FROM order\_delivered\_customer\_date) AS delivered\_date,

order\_id,

order\_status

FROM `scaler-380812.target.orders`

)

)

SELECT

g.geolocation\_state,

ROUND(AVG(T1.time\_to\_delivery),2) AS avg\_time\_to\_deliver

FROM `scaler-380812.target.geolocation` AS g

JOIN `scaler-380812.target.sellers` AS s

ON g.geolocation\_zip\_code\_prefix = s.seller\_zip\_code\_prefix

JOIN `scaler-380812.target.order\_items` AS oi

ON oi.seller\_id = s.seller\_id

JOIN T1

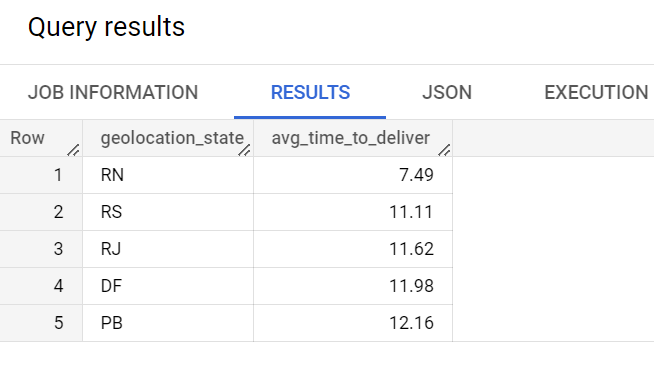
ON T1.order\_id = oi.order\_id

WHERE T1.order\_status = "delivered"

GROUP BY g.geolocation\_state

ORDER BY avg\_time\_to\_deliver ASC

LIMIT 5



First, we are using CTE to get time\_to\_deliver, then using joins to join the tables.

Then we are grouping the data by state and using AVG function to get average time to deliver the order.

From the above data, it can be seen that state RN has least average time to deliver the order. It also shows the top 5 states where the average time to deliver the order is least.

1. **Top 5 states where delivery is really fast/ not so fast compared to estimated date**

WITH T1 AS

(

SELECT

Order\_id,

DATE\_DIFF(delivered\_date, purchase\_date , day) AS time\_to\_deliver,

DATE\_DIFF(est\_delivery\_date , purchase\_date, day) AS est\_time\_to\_deliver

FROM

(

SELECT

EXTRACT(date FROM order\_purchase\_timestamp) AS purchase\_date,

EXTRACT(date FROM order\_delivered\_customer\_date) AS delivered\_date,

EXTRACT(date FROM order\_estimated\_delivery\_date) AS est\_delivery\_date,

order\_id

FROM `scaler-380812.target.orders`

)

)

SELECT

geolocation\_state,

ROUND(avg\_time\_to\_deliver,2) AS avg\_time\_to\_deliver,

ROUND(avg\_est\_time\_to\_deliver,2) AS avg\_est\_time\_to\_deliver

FROM

(

SELECT

g.geolocation\_state,

AVG(time\_to\_deliver) AS avg\_time\_to\_deliver,

AVG(est\_time\_to\_deliver) AS avg\_est\_time\_to\_deliver

FROM `scaler-380812.target.geolocation` AS g

JOIN `scaler-380812.target.sellers` AS s

ON g.geolocation\_zip\_code\_prefix = s.seller\_zip\_code\_prefix

JOIN `scaler-380812.target.order\_items` AS oi

ON oi.seller\_id = s.seller\_id

JOIN T1

ON T1.order\_id = oi.order\_id

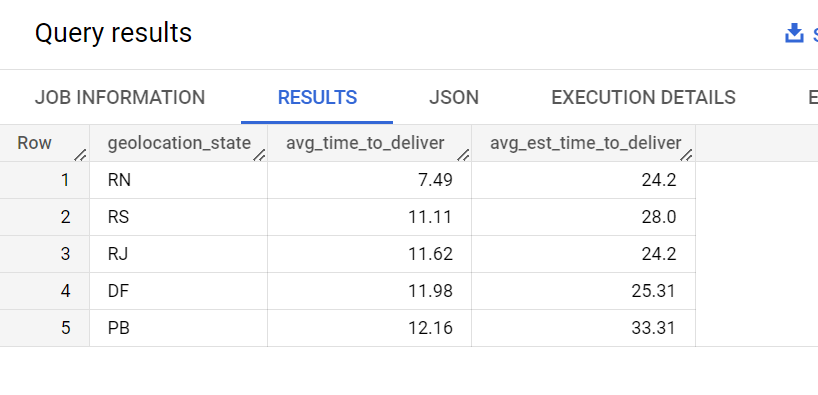
GROUP BY g.geolocation\_state

)

WHERE NOT avg\_time\_to\_deliver IS NULL

ORDER BY avg\_time\_to\_deliver ASC

LIMIT 5



We are using CTE first to extract date part from the columns and then calculating date difference so as to calculate the average time to deliver and average estimated time to deliver.

From the above result, this can be verified that average time to deliver in the mentioned states is much less than the estimated time to deliver. From all the states, state RN has least average time to deliver.

Above result shows the top 5 states where time to deliver the product is least.

**6. Payment type analysis:**

1. **Month over Month count of orders for different payment types**

SELECT

payment\_type,

year,

month\_number,

COUNT(o.order\_id) AS no\_of\_orders

FROM

  (

  SELECT

  EXTRACT(month FROM order\_purchase\_timestamp) AS month\_number,

  EXTRACT(year FROM order\_purchase\_timestamp) AS year,

  order\_id

  FROM `scaler-380812.target.orders`

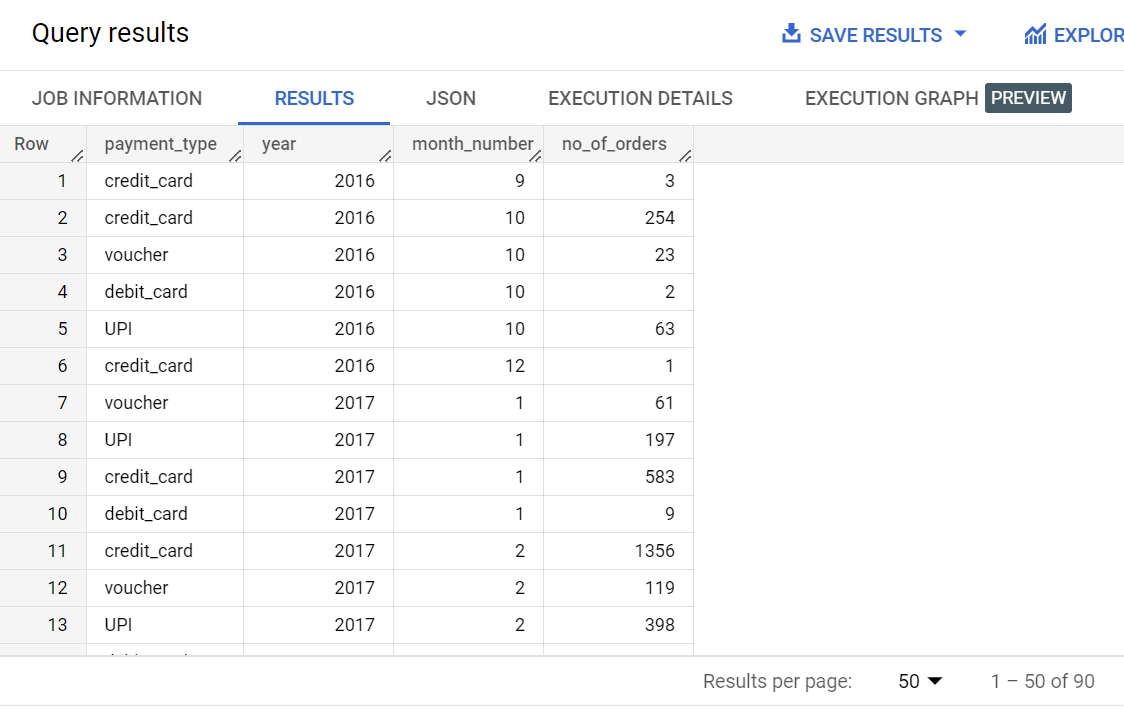
  ) AS o

JOIN `scaler-380812.target.payments` AS p

ON o.order\_id = p.order\_id

GROUP BY payment\_type, month\_number, year

ORDER BY year ASC,month\_number ASC



Here, the data displays the various mode of payment used by customers to buy the products. It also shows the number of orders placed in each month and in each year using different types of the payment methods.

Further,

SELECT

payment\_type,

COUNT(o.order\_id) AS no\_of\_orders

FROM

  (

  SELECT

  EXTRACT(month FROM order\_purchase\_timestamp) AS month\_number,

  EXTRACT(year FROM order\_purchase\_timestamp) AS year,

  order\_id

  FROM `scaler-380812.target.orders`

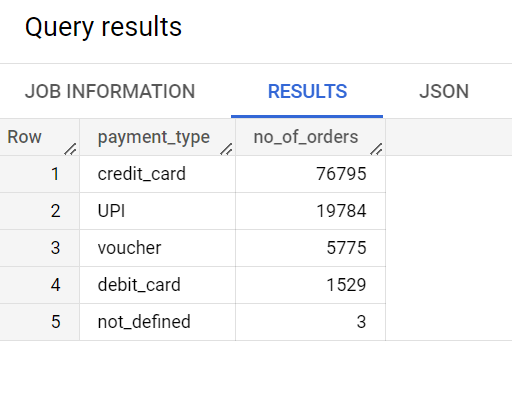
  ) AS o

JOIN `scaler-380812.target.payments` AS p

ON o.order\_id = p.order\_id

GROUP BY payment\_type

ORDER BY no\_of\_orders DESC



This output shows that most used payment mode is credit card by the customers to place an order. So, in additional we can provide some discount on the other payment modes so as to promote them if required.

1. **Count of orders based on the no. of payment installments**

SELECT

x.payment\_installments,

x.count\_of\_orders

FROM

(

SELECT

p.payment\_installments,

COUNT(o.order\_id) AS count\_of\_orders

FROM `scaler-380812.target.orders` AS o

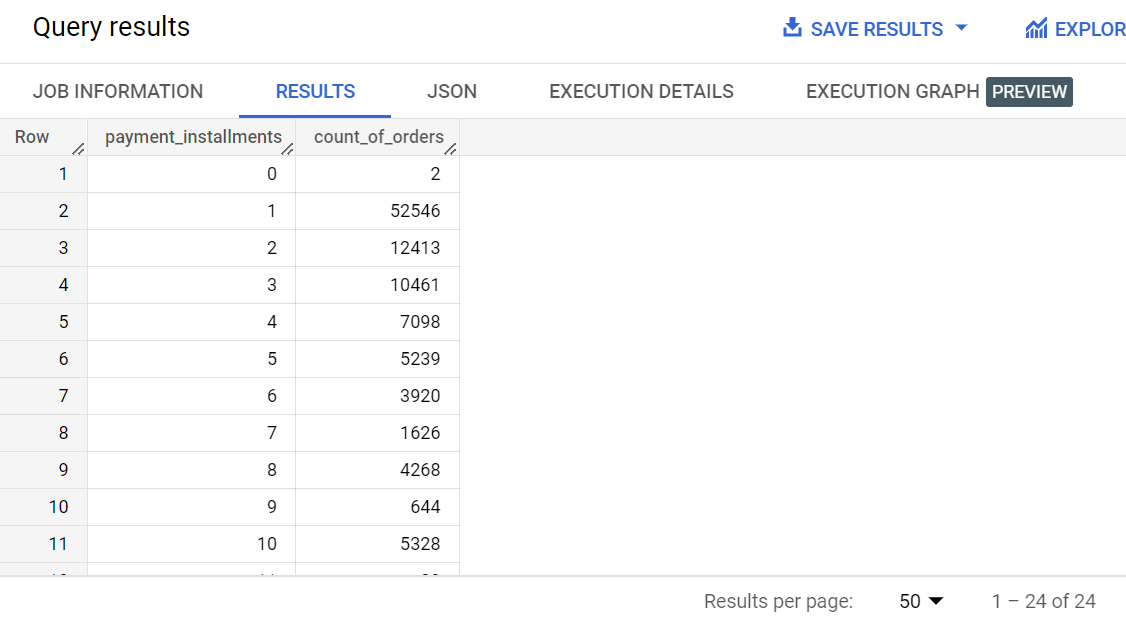
JOIN `scaler-380812.target.payments` p

ON o.order\_id = p.order\_id

GROUP BY p.payment\_installments

) AS x

ORDER BY x.payment\_installments ASC



This output shows the number of installments opted by the customers to pay for the product. And also showing the numbers of orders placed by customers using different number of installments.

Further-

SELECT

x.payment\_installments,

x.count\_of\_orders

FROM

(

SELECT

p.payment\_installments,

COUNT(o.order\_id) AS count\_of\_orders

FROM `scaler-380812.target.orders` AS o

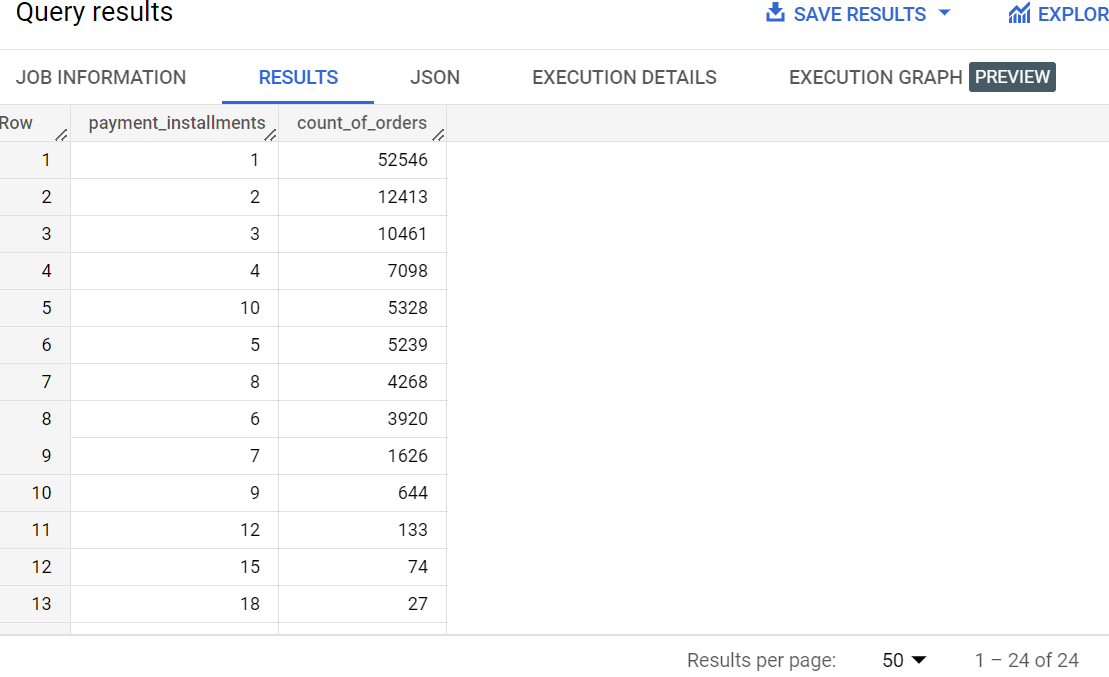
JOIN `scaler-380812.target.payments` p

ON o.order\_id = p.order\_id

GROUP BY p.payment\_installments

) AS x

ORDER BY x.count\_of\_orders DESC



This output shows the maximum number of orders are places using only 1 payment installment, which is good for business, in additional it can be seen even more than 10 payment installments are used by the customers to place an order, which means company has to wait for a long time to receive full payment.

We can sort this by providing some additional discounts or reducing the additional fee on the installments in order to promote less installments method.