PROJECT 5 - TARGET

Context

Target is one of the world's most recognized brands and one of America's leading retailers. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

This business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allow viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers.

The **customers.csv** contain following features:

Features	Description
customer_id	Id of the consumer who made the purchase.
customer_unique_id	Unique Id of the consumer.
customer_zip_code_prefix	Zip Code of the location of the consumer.
customer_city	Name of the City from where order is made.
customer_state	State Code from where order is made(Ex- sao paulo-SP).

The **sellers.csv** contains following features:

Features	Description
seller_id	Unique Id of the seller registered
seller_zip_code_prefix	Zip Code of the location of the seller.
seller_city	Name of the City of the seller.
seller_state	State Code (Ex- sao paulo-SP)

The **order_items.csv** contain following features:

Features	Description
order_id	A unique id of order made by the consumers.
order_item_id	A Unique id given to each item ordered in the order.
product_id	A unique id given to each product available on the site.
seller_id	Unique Id of the seller registered in Target.
shipping_limit_date	The date before which shipping of the ordered product must be completed.
price	Actual price of the products ordered .
freight_value	Price rate at which a product is delivered from one point to another.

The **payments.csv** contain following features:

Features	Description
order_id	A unique id of order made by the consumers.
payment_sequential	sequences of the payments made in case of EMI.
payment_type	mode of payment used.(Ex-Credit Card)
payment_installments	number of installments in case of EMI purchase.
payment_value	Total amount paid for the purchase order.

The **orders.csv** contain following features:

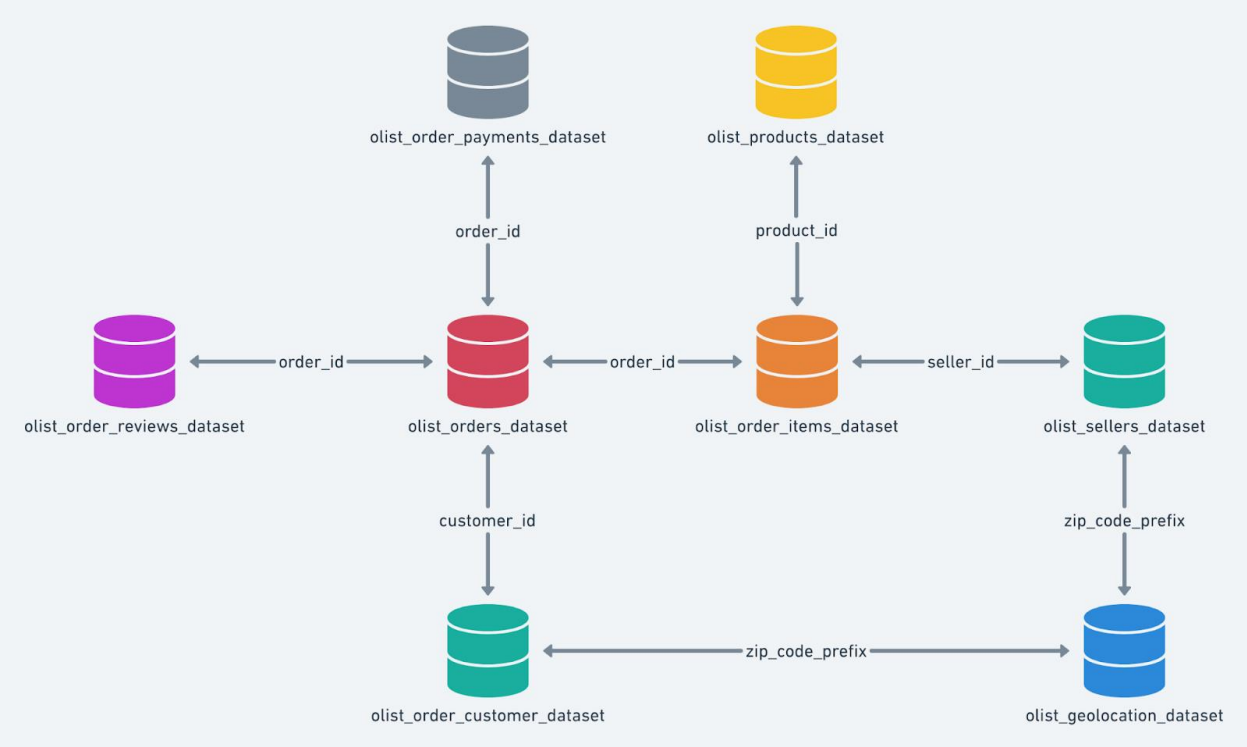
Features	Description
order_id	A unique id of order made by the consumers.
customer_id	Id of the consumer who made the purchase.
order_status	status of the order made i.e delivered, shipped etc.
order_purchase_timestamp	Timestamp of the purchase.
order_delivered_carrier_date	delivery date at which carrier made the delivery.
order_delivered_customer_date	date at which customer got the product.
order_estimated_delivery_date	estimated delivery date of the products.

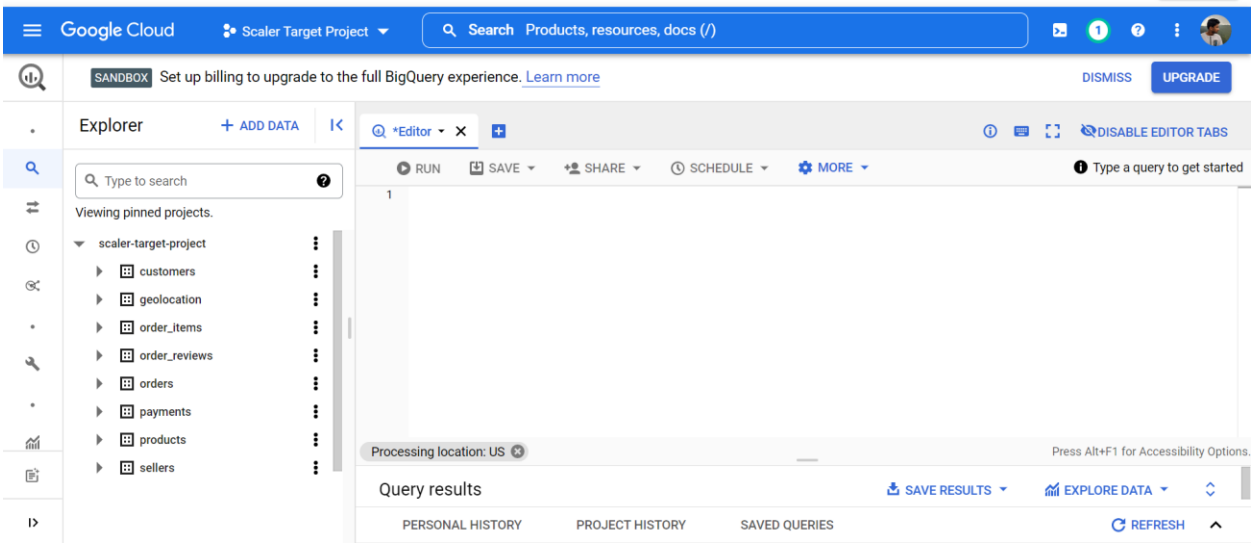
The **reviews.csv** contain following features:

Features	Description
review_id	Id of the review given on the product ordered by the order id.
order_id	A unique id of order made by the consumers.
review_score	review score given by the customer for each order on the scale of 1–5.
review_comment_title	Title of the review
review_comment_message	Review comments posted by the consumer for each order.
review_creation_date	Timestamp of the review when it is created.
review_answer_timestamp	Timestamp of the review answered.

The **products.csv** contain following features:

Features	**Description**
product_id	A unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_lenght	length of the string which specifies the name given to the products ordered.
product_description_lenght	length of the description written for each product ordered on the site.
product_photos_qty	Number of photos of each product ordered available on the shopping portal.
product_weight_g	Weight of the products ordered in grams.
product_length_cm	Length of the products ordered in centimeters.
product_height_cm	Height of the products ordered in centimeters.
product_width_cm	width of the product ordered in centimeters.



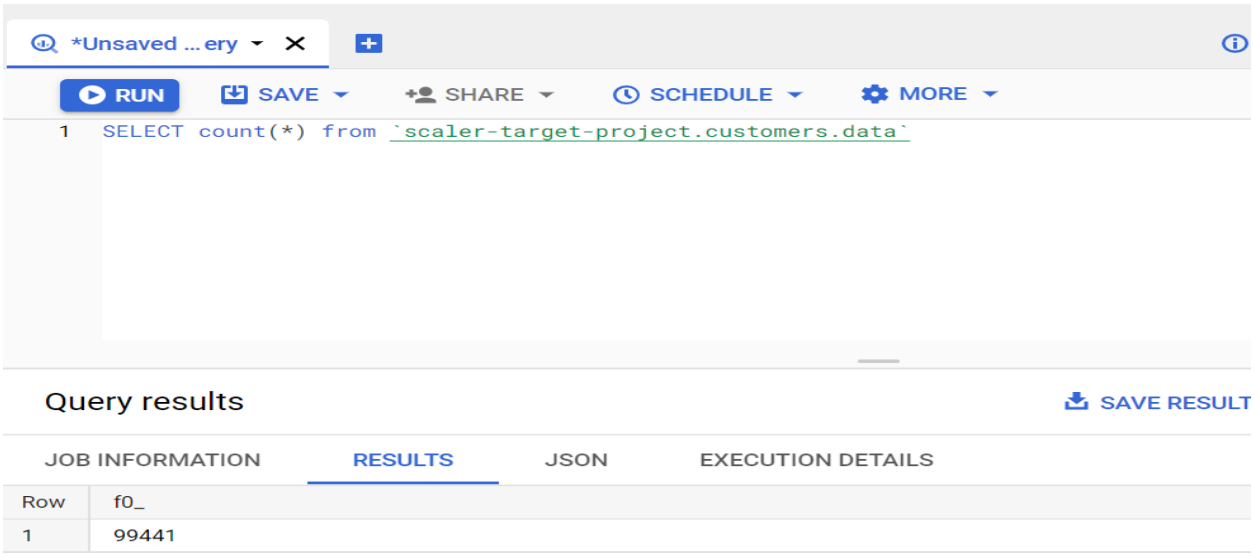


QUESTIONS:

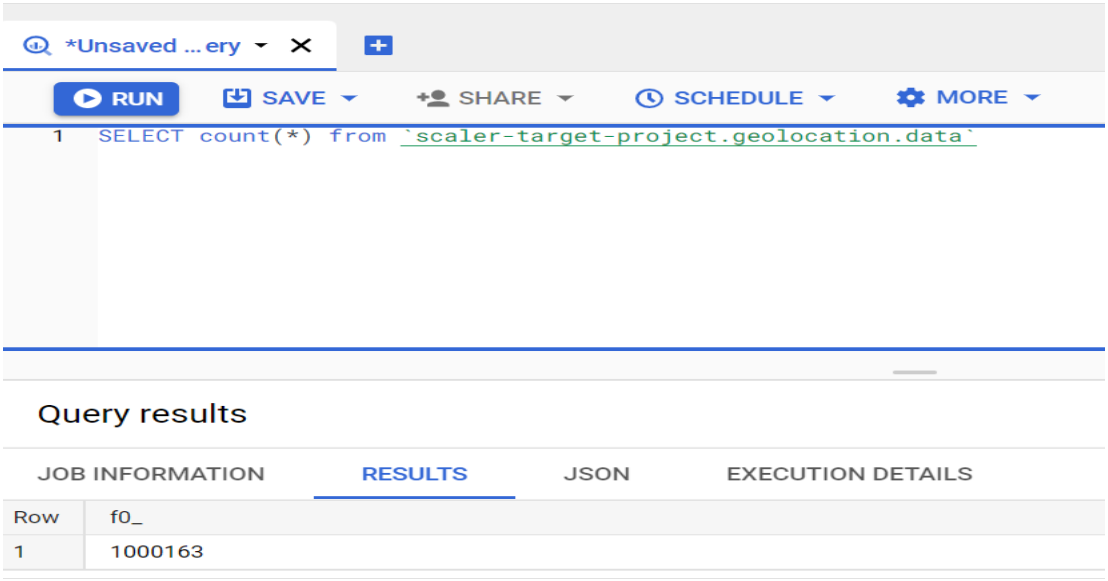
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

    1. Get number of rows in the data

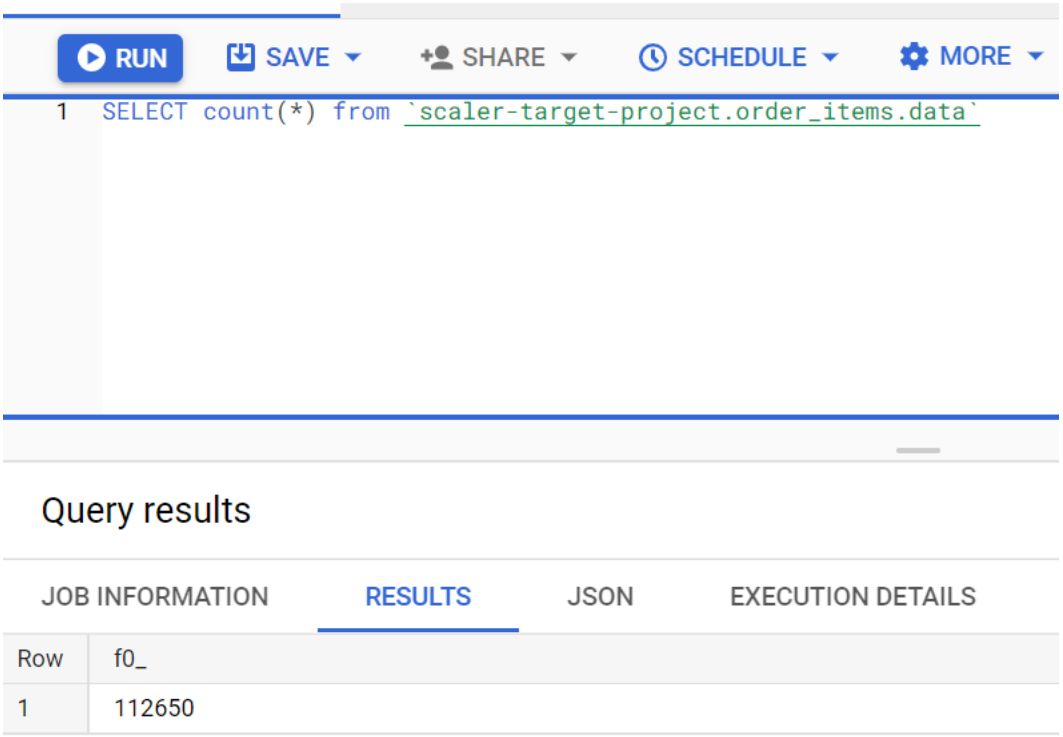
- **Customers Table** – 99441 Rows



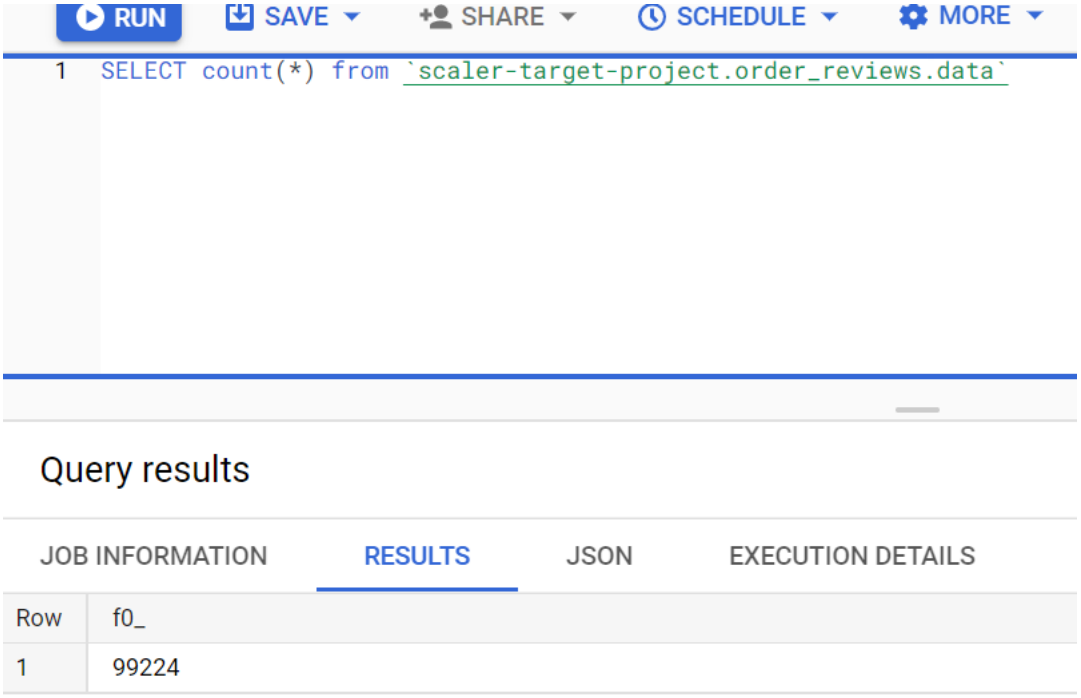
- **Geolocation Table** – 1000163 Rows



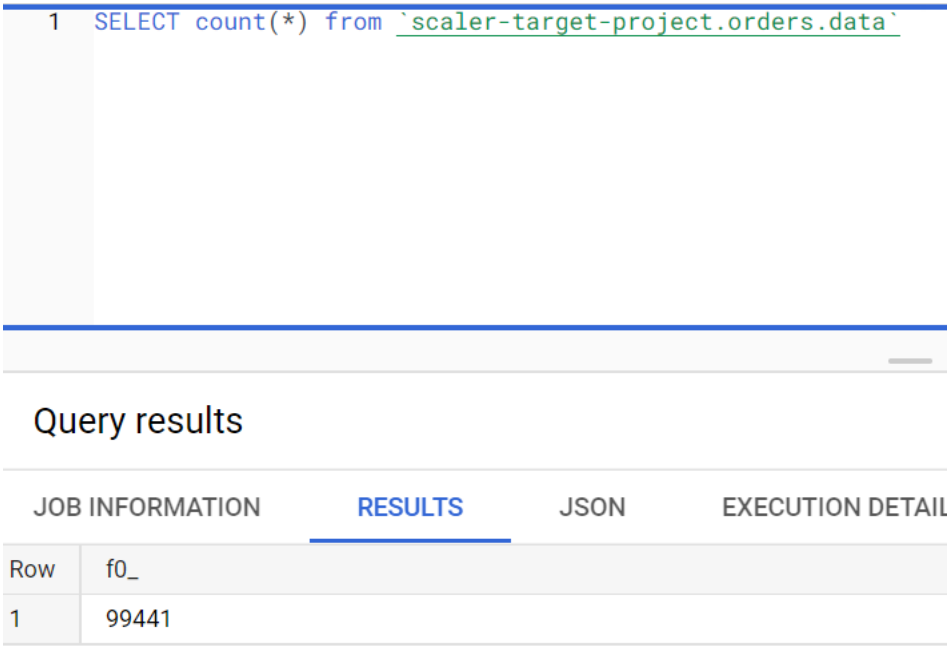
- **Order_Items Table** –112650 Rows



- **Order_Reviews Table** – 99224 Rows



- **Orders Table** – 99441 Rows



- **Payments Table** – 103886 Rows

```
1  SELECT count(*) from `scaler-target-project.payments.data`
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	f0_
1	103886

- **Products Table** – 32951 Rows

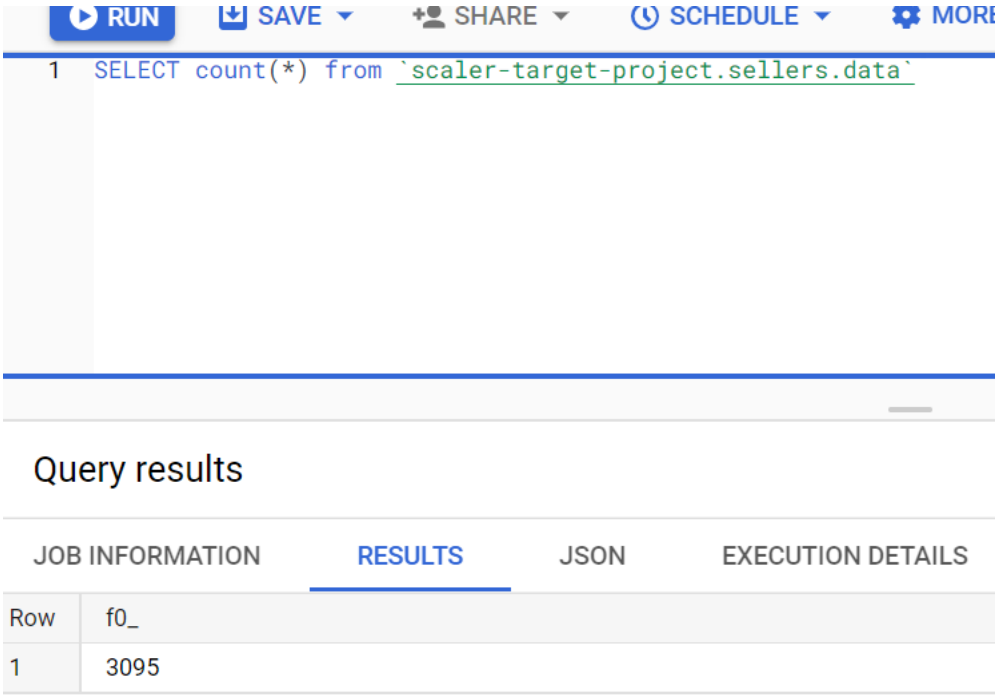
```
1  SELECT count(*) from `scaler-target-project.products.data`
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	f0_
1	32951

- **Sellers Table** – 3095 Rows



2. Number of null or missing values in a column

For Customers Table :

1) `SELECT count(*) from `scaler-target-project.customers.data` where customer_id IS NULL;`
   `0 Null Values`
2) `SELECT count(*) from `scaler-target-project.customers.data` where customer_unique_id IS NULL;` `0 Null Values`
3) `SELECT count(*) from `scaler-target-project.customers.data` where customer_zip_code_prefix IS NULL;` `0 Null Values`
4) `SELECT count(*) from `scaler-target-project.customers.data` where customer_city IS NULL;` `0 Null Values`
5) `SELECT count(*) from `scaler-target-project.customers.data` where customer_state IS NULL;` `0 Null Values`

For Sellers Table :

1) `SELECT count(*) from `scaler-target-project.sellers.data` where seller_id IS NULL;`
   `0 Null Values`
2) `SELECT count(*) from `scaler-target-project.sellers.data` where seller_zip_code_prefix IS NULL;` `0 Null Values`
3) `SELECT count(*) from `scaler-target-project.sellers.data` where seller_city IS NULL;`
   `0 Null Values`
4) `SELECT count(*) from `scaler-target-project.sellers.data` where seller_state IS NULL;`
   `0 Null Values`

For Order_Items Table :

1) SELECT count(*) from `scaler-target-project.order_items.data` where order_id IS NULL
   0 Null Values
2) SELECT count(*) from `scaler-target-
   project.order_items.data` where order_item_id IS NULL 0 Null Values
3) SELECT count(*) from `scaler-target-project.order_items.data` where product_id IS NULL
   0 Null Values
4) SELECT count(*) from `scaler-target-project.order_items.data` where seller_id IS NULL
   0 Null Values
5) SELECT count(*) from `scaler-target-
   project.order_items.data` where shipping_limit_date IS NULL 0 Null Values
6) SELECT count(*) from `scaler-target-project.order_items.data` where price IS NULL
   0 Null Values
7) SELECT count(*) from `scaler-target-
   project.order_items.data` where freight_value IS NULL 0 Null Values

For Payments Table :

1) SELECT count(*) from `scaler-target-project.payments.data` where order_id IS NULL;
   0 Null Values
2) SELECT count(*) from `scaler-target-
   project.payments.data` where payment_sequential IS NULL; 0 Null Values
3) SELECT count(*) from `scaler-target-
   project.payments.data` where payment_type IS NULL; 0 Null Values
4) SELECT count(*) from `scaler-target-
   project.payments.data` where payment_installments IS NULL; 0 Null Values
5) SELECT count(*) from `scaler-target-
   project.payments.data` where payment_type IS NULL; 0 Null Values

For Orders Table :

1) SELECT count(*) from `scaler-target-project.orders.data` where order_id IS NULL
   0 Null Values
2) SELECT count(*) from `scaler-target-project.orders.data` where customer_id IS NULL
   0 Null Values
3) SELECT count(*) from `scaler-target-project.orders.data` where order_status IS NULL
   0 Null Values
4) SELECT count(*) from `scaler-target-
   project.orders.data` where order_purchase_timestamp IS NULL 0 Null Values
5) SELECT count(*) from `scaler-target-
   project.orders.data` where order_delivered_carrier_date IS NULL 1783 Null Values
6) SELECT count(*) from `scaler-target-
   project.orders.data` where order_delivered_customer_date IS NULL 2965 Null Values
7) SELECT count(*) from `scaler-target-
   project.orders.data` where order_estimated_delivery_date IS NULL 0 Null Values

For Order_Reviews Table :

1) SELECT count(*) from `scaler-target-project.order_reviews.data` where review_id IS NULL; 0 Null Values
2) SELECT count(*) from `scaler-target-project.order_reviews.data` where order_id IS NULL; 0 Null Values
3) SELECT count(*) from `scaler-target-project.order_reviews.data` where review_score IS NULL; 0 Null Values
4) SELECT count(*) from `scaler-target-project.order_reviews.data` where review_comment_title IS NULL; 87675 Null Values
5) SELECT count(*) from `scaler-target-project.order_reviews.data` where review_creation_date IS NULL; 0 Null Values
6) SELECT count(*) from `scaler-target-project.order_reviews.data` where review_answer_timestamp IS NULL; 0 Null Values

For Products Table :

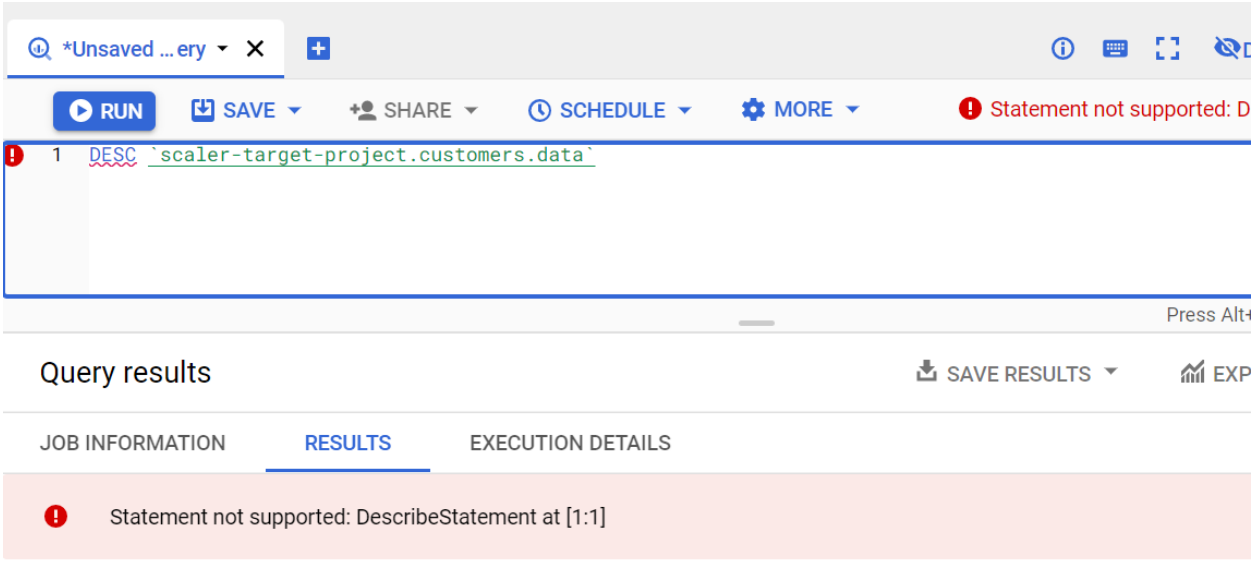
1) SELECT count(*) from `scaler-target-project.products.data` where product_id IS NULL 0 Null Values
2) SELECT count(*) from `scaler-target-project.products.data` where product_category IS NULL 610 0 Null Values
3) SELECT count(*) from `scaler-target-project.products.data` where product_name_length IS NULL 610 0 Null Values
4) SELECT count(*) from `scaler-target-project.products.data` where product_description_length IS NULL 610 0 Null Values
5) SELECT count(*) from `scaler-target-project.products.data` where product_photos_qty IS NULL 610 0 Null Values
6) SELECT count(*) from `scaler-target-project.products.data` where product_weight_g IS NULL 2 0 Null Values
7) SELECT count(*) from `scaler-target-project.products.data` where product_length_cm IS NULL 2 0 Null Values
8) SELECT count(*) from `scaler-target-project.products.data` where product_height_cm IS NULL 2 0 Null Values
9) SELECT count(*) from `scaler-target-project.products.data` where product_width_cm IS NULL 2 0 Null Values

For Geolocation Table :

1) SELECT count(*) from `scaler-target-project.geolocation.data` where geolocation_zip_code_prefix IS NULL 0 Null Values
2) SELECT count(*) from `scaler-target-project.geolocation.data` where geolocation_lat IS NULL 0 Null Values
3) SELECT count(*) from `scaler-target-project.geolocation.data` where geolocation_lng IS NULL 0 Null Values
4) SELECT count(*) from `scaler-target-project.geolocation.data` where geolocation_city IS NULL 0 Null Values
5) SELECT count(*) from `scaler-target-project.geolocation.data` where geolocation_state IS NULL 0 Null Values

3. Data type of columns in a table

We can see the data type of columns of a particular table using DESC Table_Name. This command is however not working in BigQuery. I have used Python to get the data types of the columns.




```
customers=pd.read_csv('customers.csv')
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   customer_id             99441 non-null  object
 1   customer_unique_id      99441 non-null  object
 2   customer_zip_code_prefix  99441 non-null  int64 
 3   customer_city           99441 non-null  object
 4   customer_state          99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
```

```
geolocation=pd.read_csv('geolocation.csv')
geolocation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000163 entries, 0 to 1000162
Data columns (total 5 columns):
 #   Column                        Non-Null Count    Dtype
---  ------                        --------------    -----
 0   geolocation_zip_code_prefix   1000163 non-null  int64
 1   geolocation_lat               1000163 non-null  float64
 2   geolocation_lng               1000163 non-null  float64
 3   geolocation_city              1000163 non-null  object
 4   geolocation_state             1000163 non-null  object
dtypes: float64(2), int64(1), object(2)
memory usage: 38.2+ MB
```

```
order_items=pd.read_csv('order_items.csv')
order_items.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   order_id             112650 non-null  object
 1   order_item_id        112650 non-null  int64
 2   product_id           112650 non-null  object
 3   seller_id            112650 non-null  object
 4   shipping_limit_date  112650 non-null  object
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
```

```
orders=pd.read_csv('orders.csv')
orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
```

```
payments=pd.read_csv('payments.csv')
payments.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103886 entries, 0 to 103885
Data columns (total 5 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   order_id              103886 non-null  object 
 1   payment_sequential    103886 non-null  int64  
 2   payment_type          103886 non-null  object 
 3   payment_installments  103886 non-null  int64  
 4   payment_value         103886 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 4.0+ MB
```

```
products=pd.read_csv('products.csv')
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   product_id                  32951 non-null  object 
 1   product category            32341 non-null  object 
 2   product_name_length         32341 non-null  float64
 3   product_description_length  32341 non-null  float64
 4   product_photos_qty          32341 non-null  float64
 5   product_weight_g            32949 non-null  float64
 6   product_length_cm           32949 non-null  float64
 7   product_height_cm           32949 non-null  float64
 8   product_width_cm            32949 non-null  float64
dtypes: float64(7), object(2)
memory usage: 2.3+ MB
```

```
sellers=pd.read_csv('sellers.csv')
sellers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3095 entries, 0 to 3094
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   seller_id               3095 non-null   object
 1   seller_zip_code_prefix  3095 non-null   int64 
 2   seller_city             3095 non-null   object
 3   seller_state            3095 non-null   object
dtypes: int64(1), object(3)
memory usage: 96.8+ KB
```

4. Get the time period for which the data is given

   The time period of the data can be found from the order_purchase_timestamp column from orders table. We can find the first and the last order_purchase_timestamp which would give us the time period of the given data.

RUN SAVE SHARE SCHEDULE MORE This

```sql
Select
  extract(date from min(order_purchase_timestamp)) as First_Date,
  extract(date from max(order_purchase_timestamp)) as Last_Date
from `scaler-target-project.orders.data`

```

Query results SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	First_Date	Last_Date
1	2016-09-04	2018-10-17

5. Number of cities in our dataset

No Of Customer_Cities in Customer Table – 4119

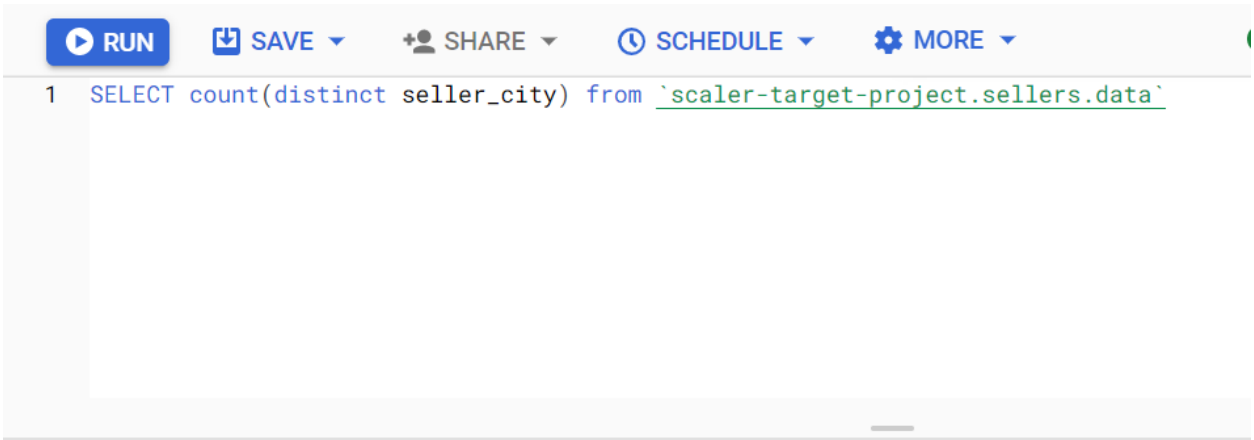
```sql
SELECT count(distinct customer_city) from `scaler-target-project.customers.data`

```

Query results SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	f0_
1	4119

No Of Seller_Cities in Seller Table – 611



Query results

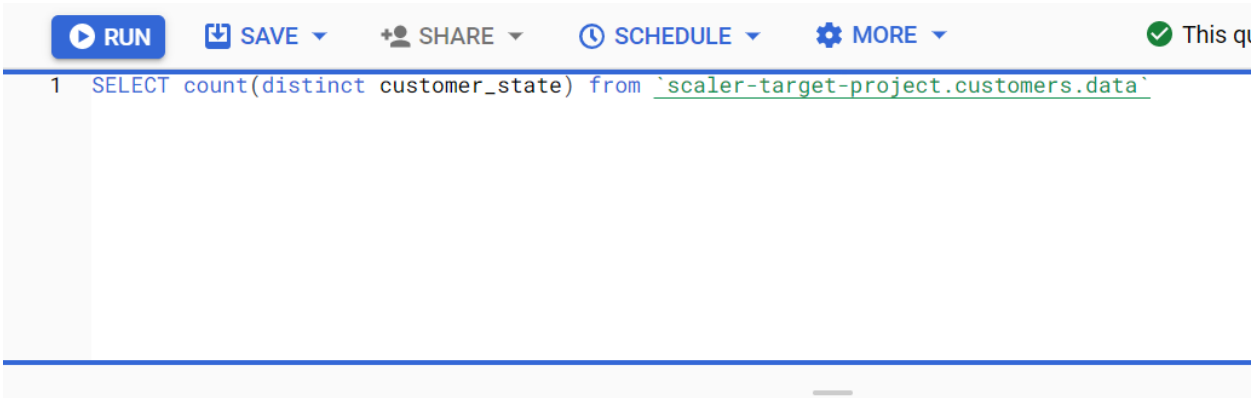
SAVE RESU

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	f0_
1	611

6. Number of states in our dataset

No Of Customer_States in Customer Table – 27



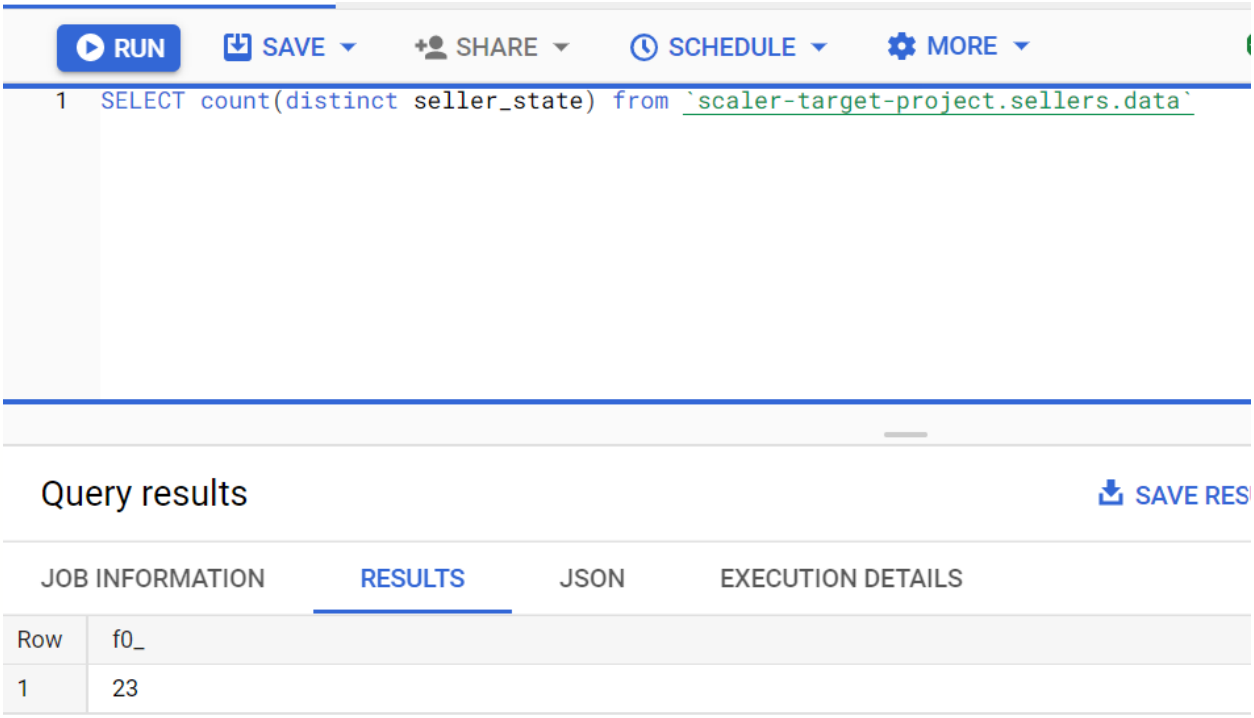
Query results

SAVE RESULTS

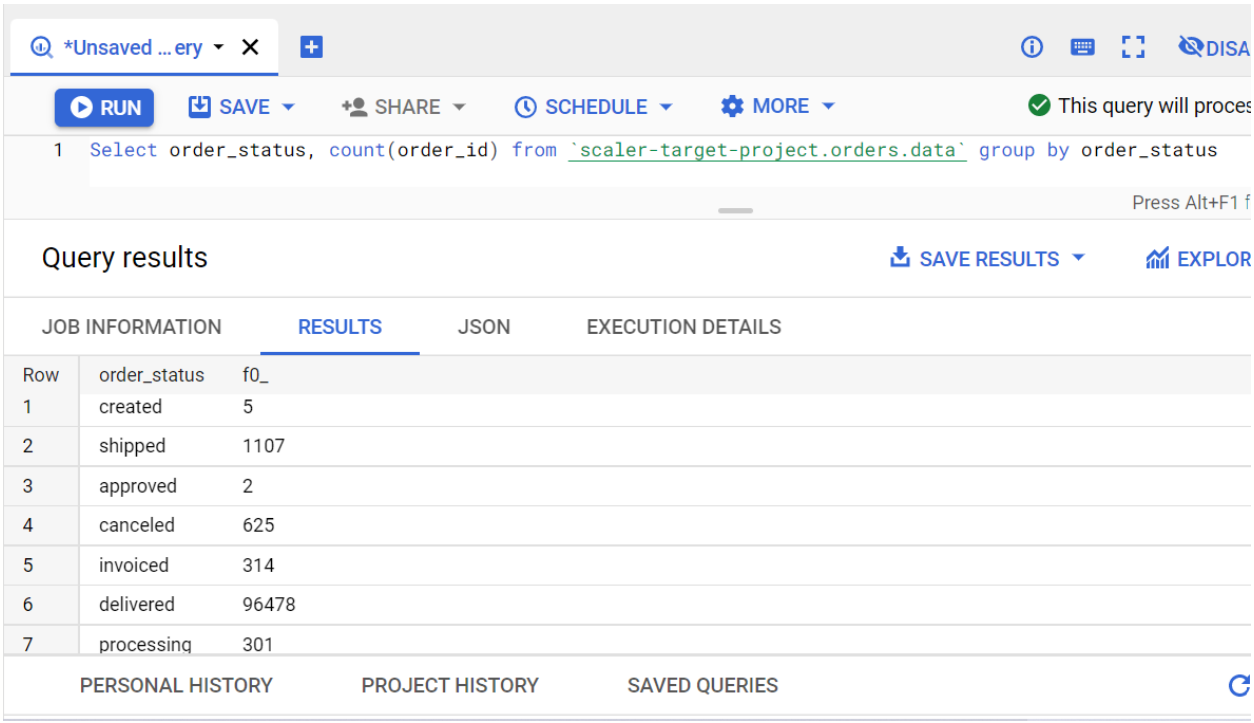
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	f0_
1	27

No Of Seller_States in Sellers Table - 23



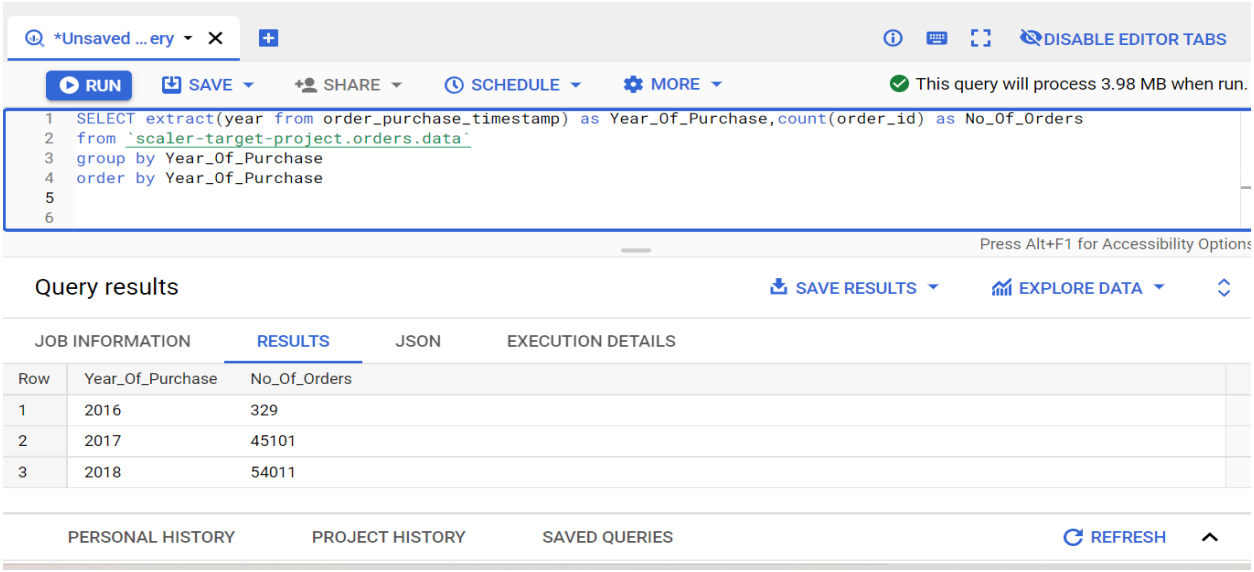
- ✓ In-depth Exploration:
    1. How many orders do we have for each order status?



1	created	5
2	shipped	1107
3	approved	2
4	canceled	625
5	invoiced	314
6	delivered	96478
7	processing	301
8	unavailable	609

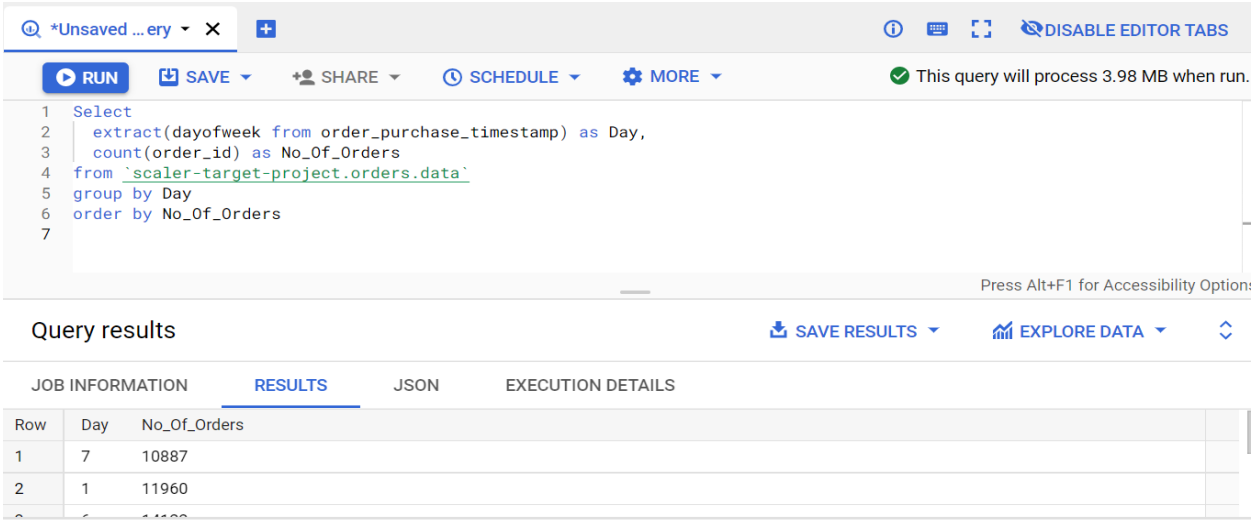
2. Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?

We can confirm from the results that there was a growing trend in Brazil as there is Year on Year increase in the number of orders.



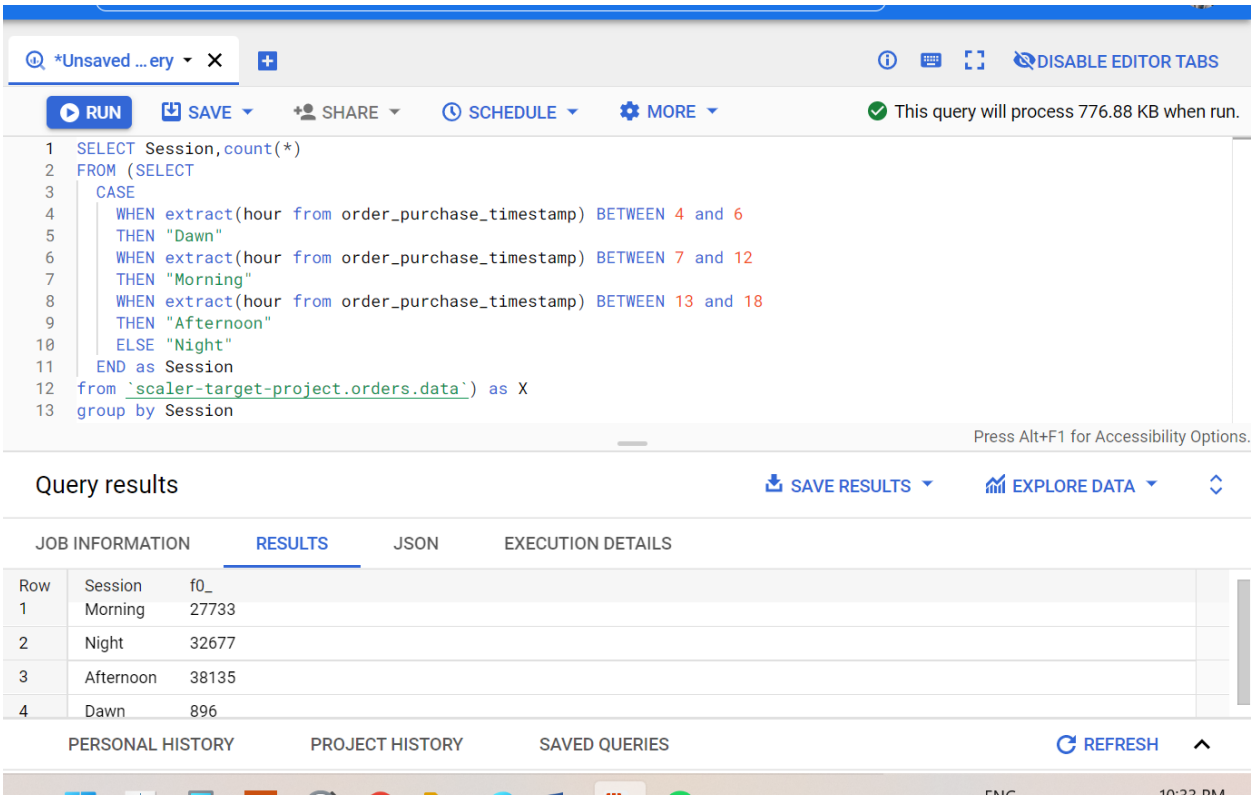
3. On what day of week brazilians customers tend to do online purchasing?

Assuming Day 1 to be Monday, maximum orders are placed on Day 3 which is Wednesday.



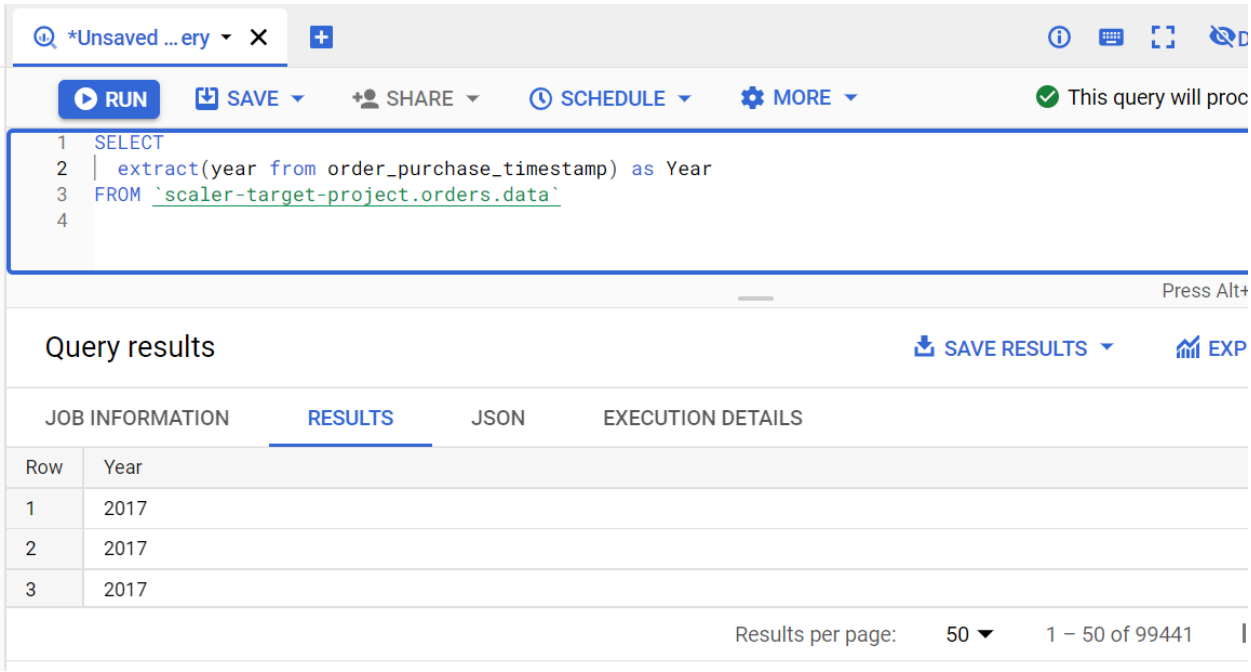
Row	Day	No_Of_Orders
1	7	10887
2	1	11960
3	6	14122
4	5	14761
5	4	15552
6	3	15963

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

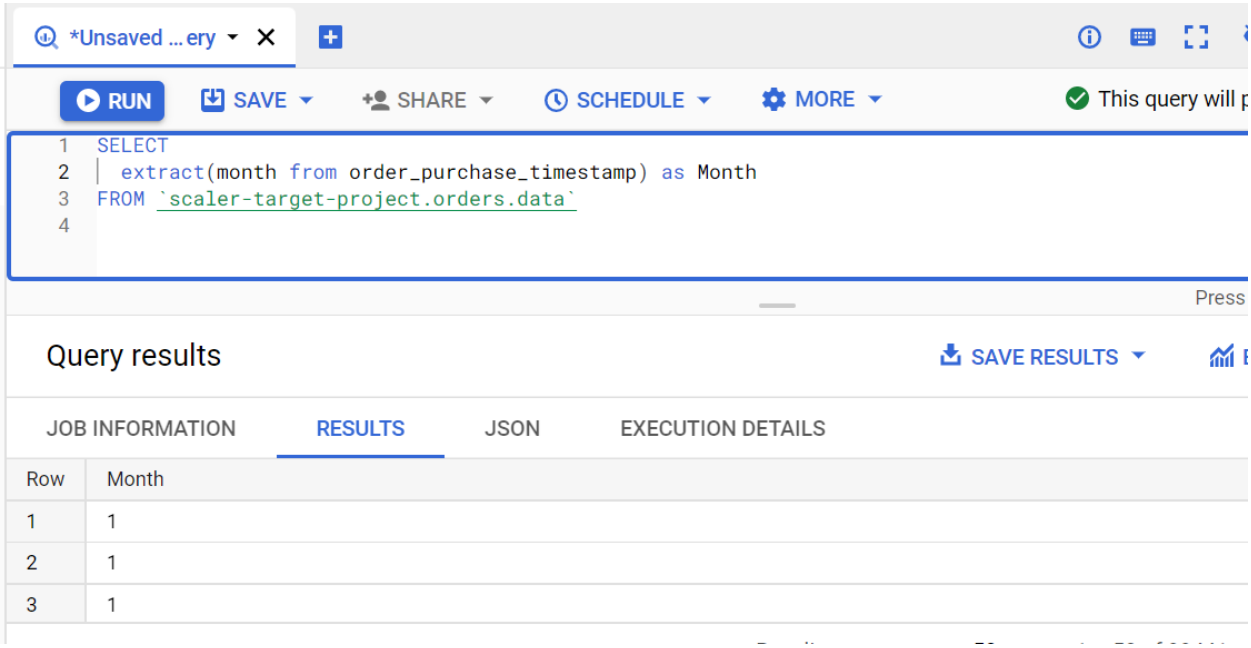


5. Feature Extraction: Through order_purchase_timestamp in “orders” dataset extract

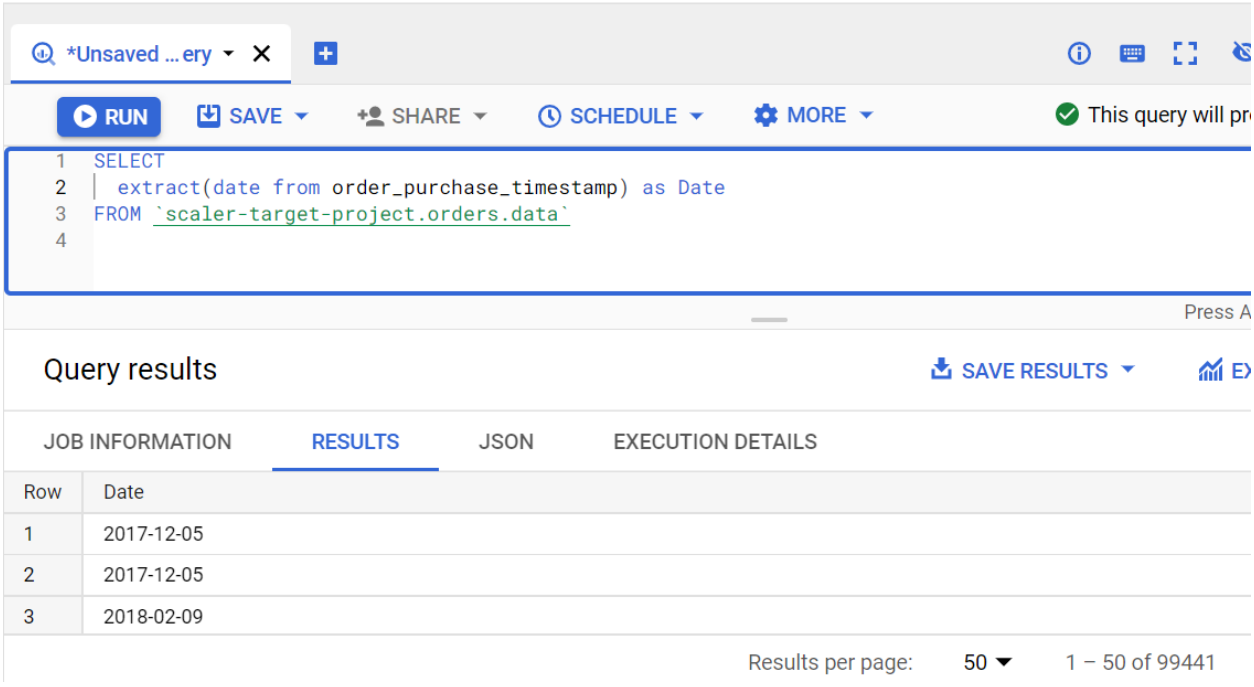
    1. order_purchase_year



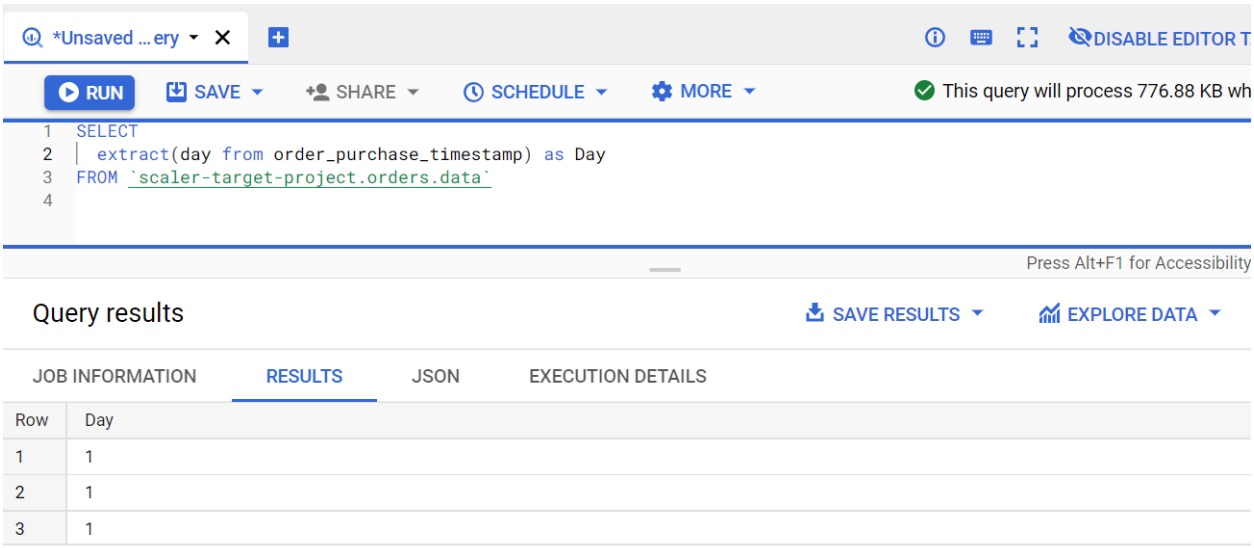
2. order_purchase_month



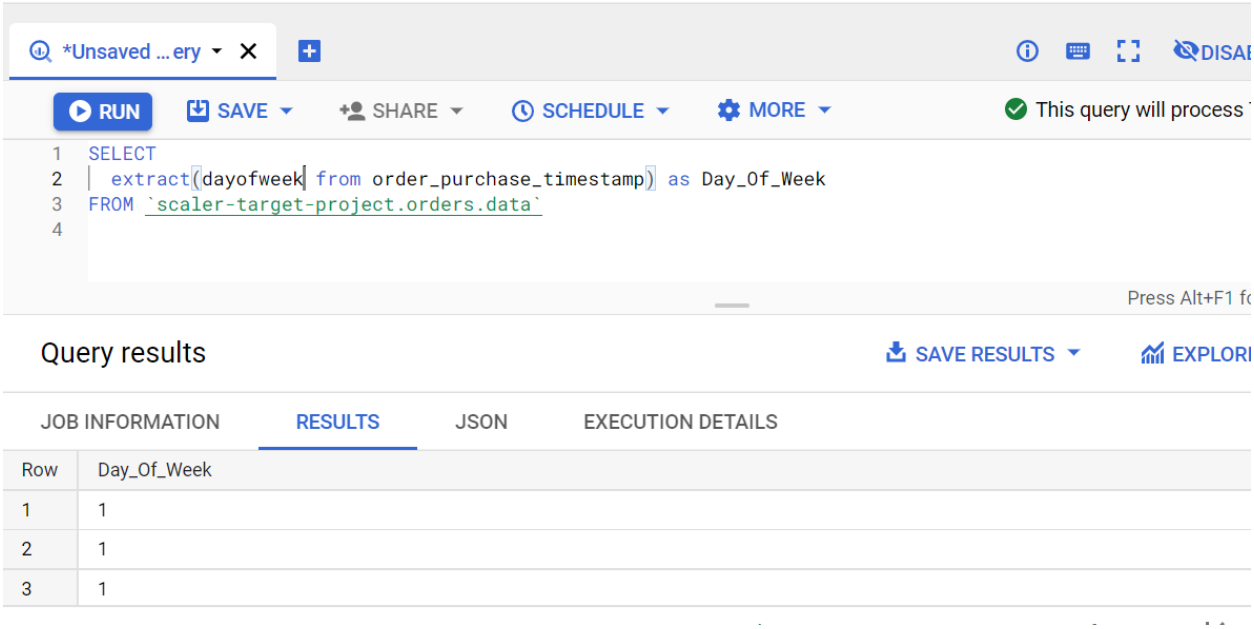
3. order_purchase_date



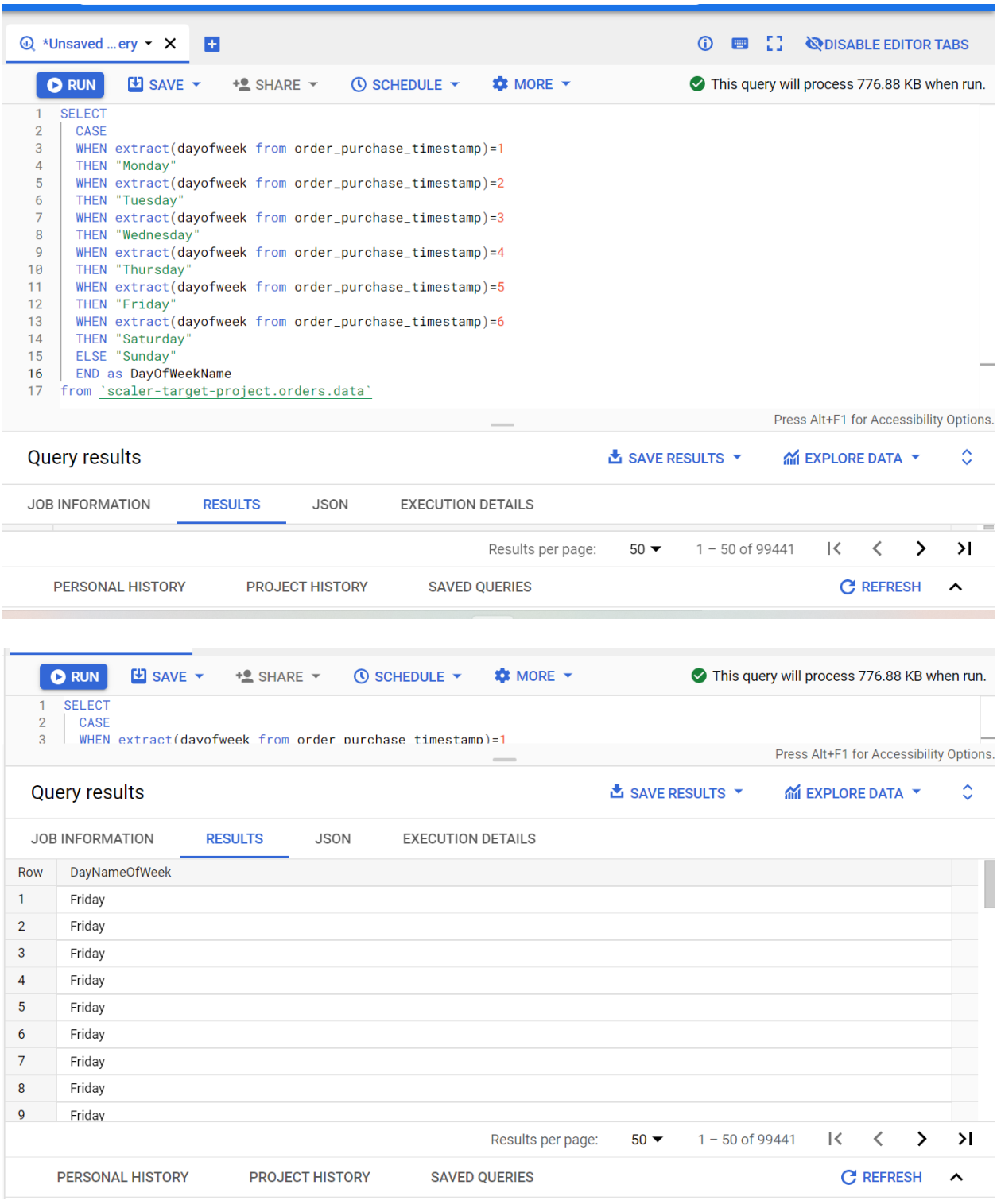
4. order_purchase_day



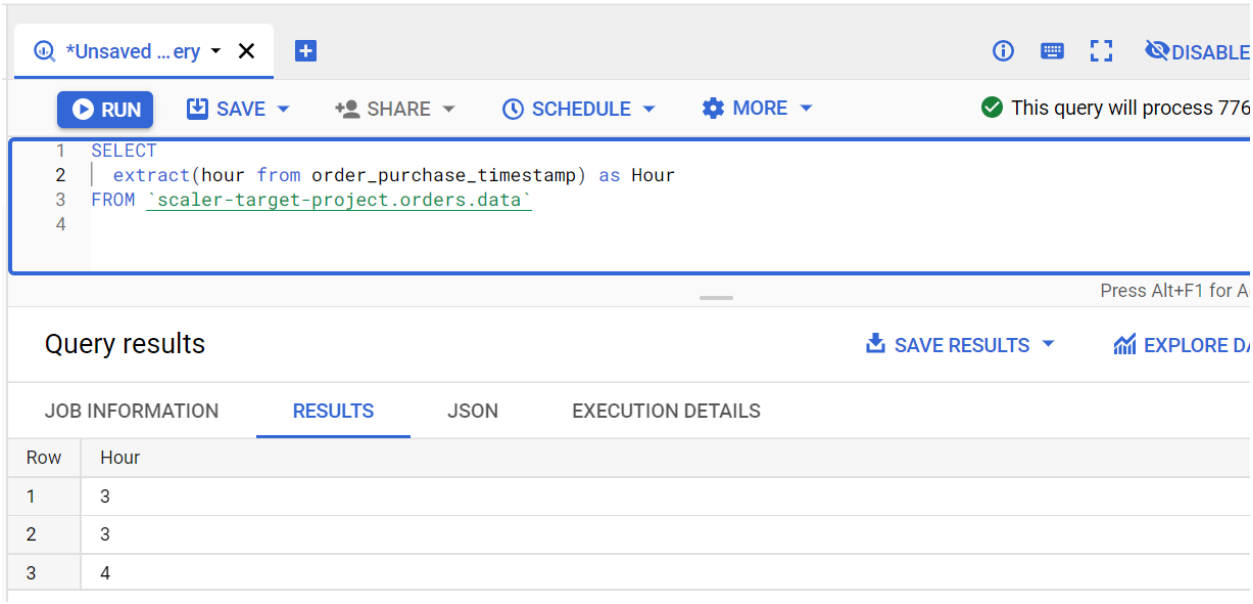
5. order_purchase_dayofweek



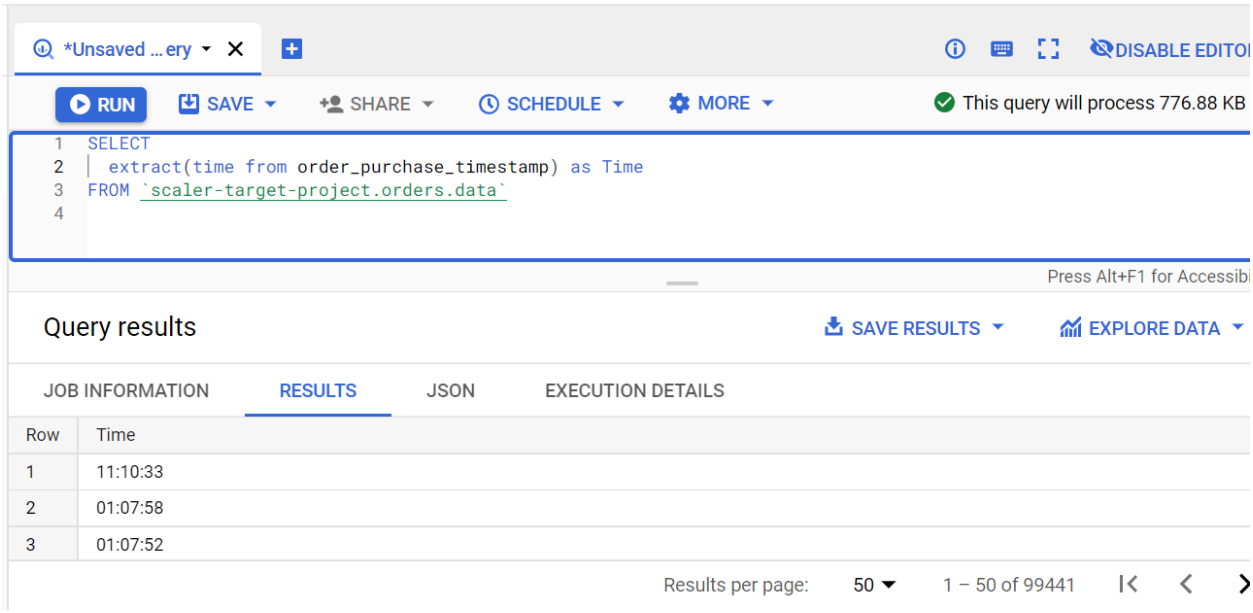
6. order_purchase_dayofweek_name



7. order_purchase_hour

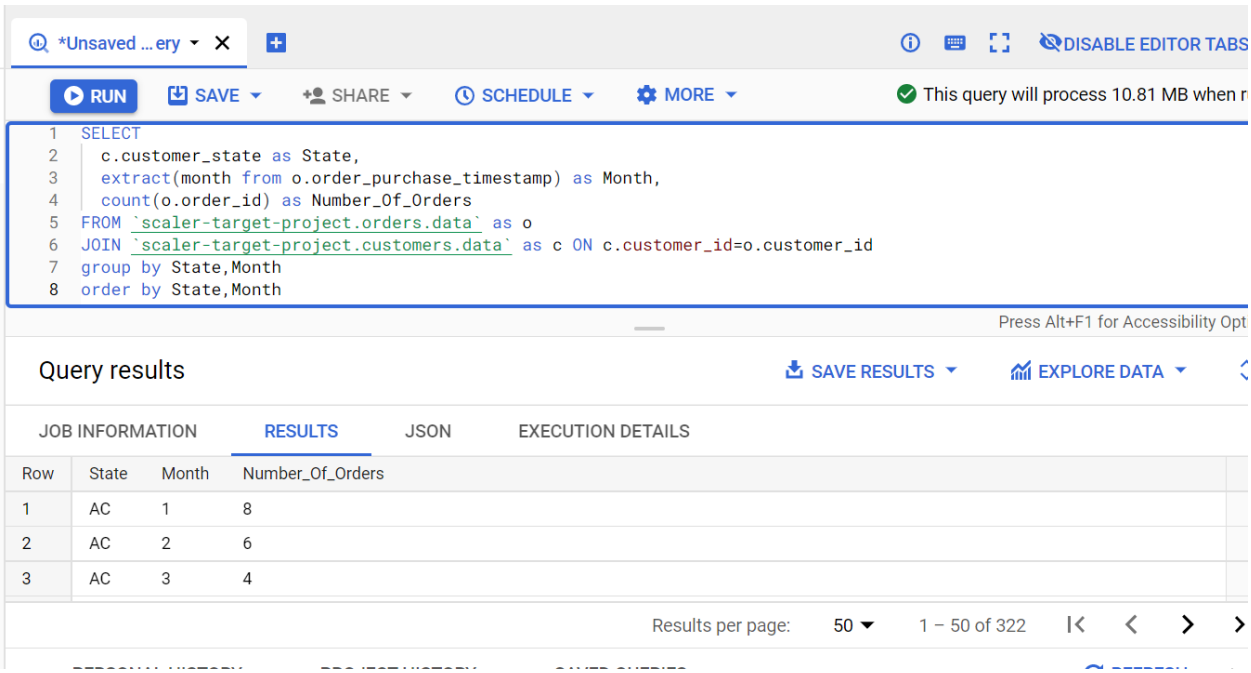


8. order_purchase_time_day

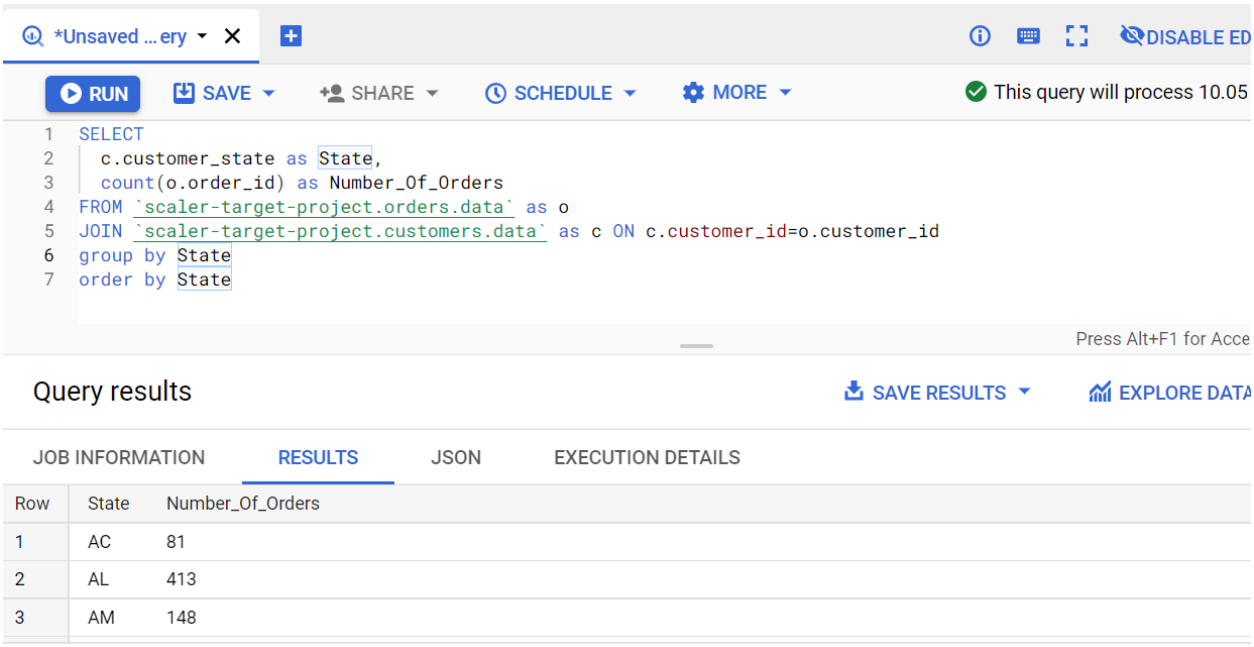


✓ Evolution of E-commerce orders in the Brazil region:

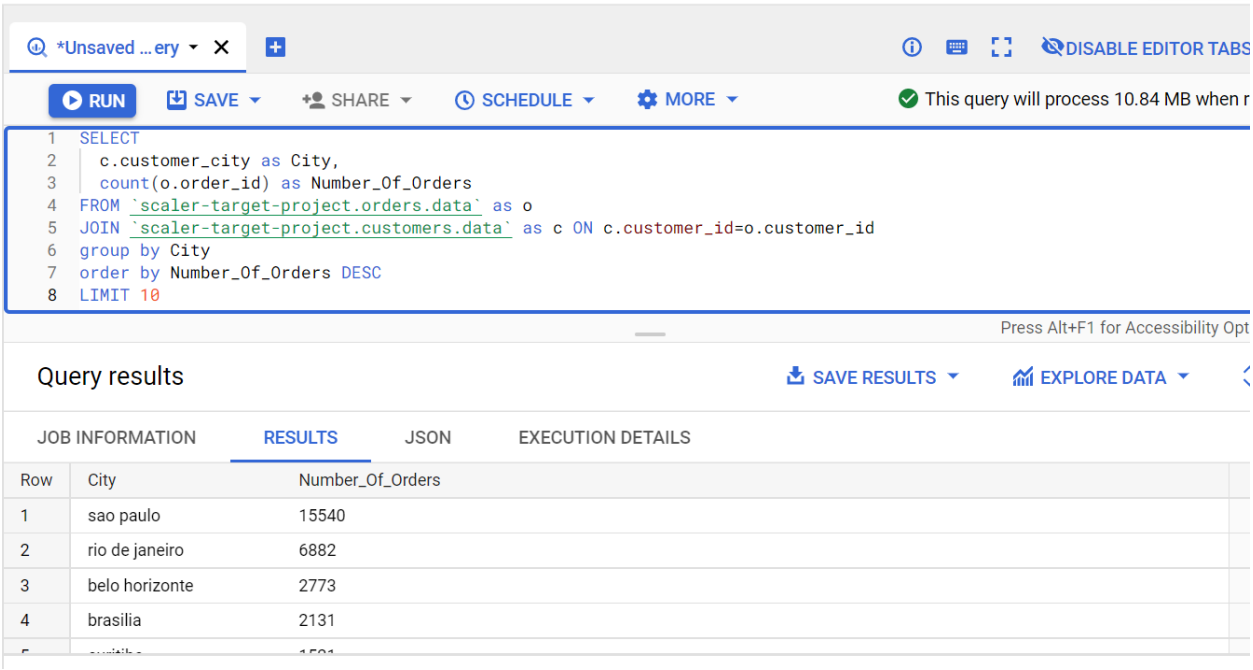
1. Get month on month orders by region



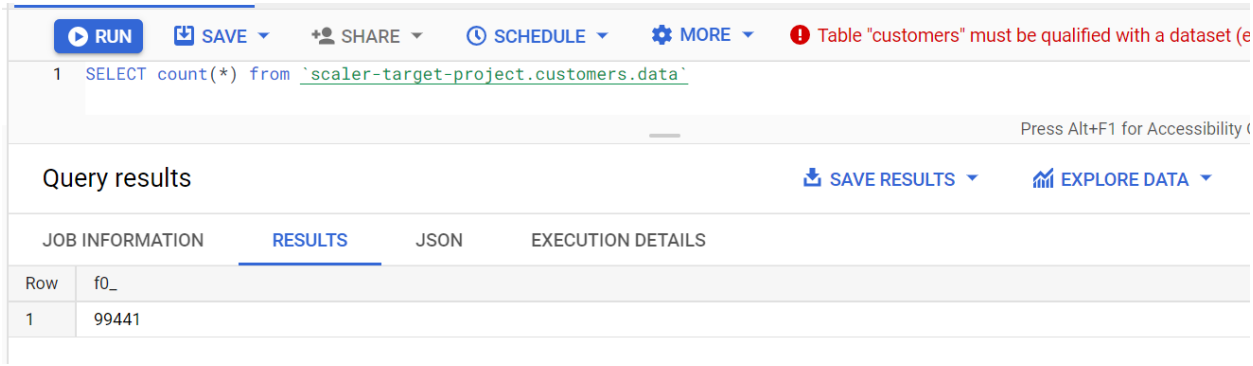
2. Total of customer orders by state

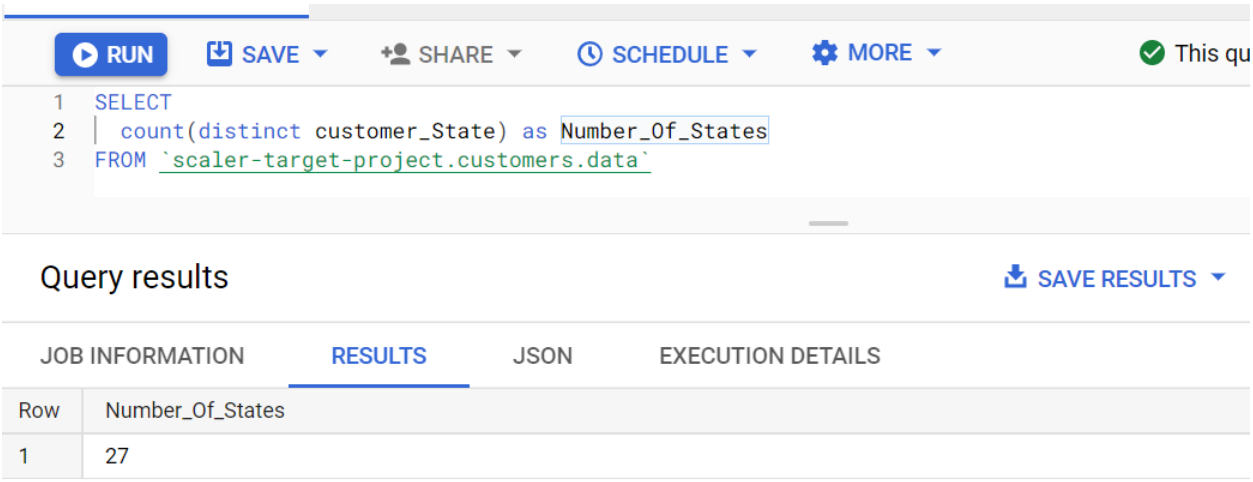


3. Top 10 brazilian cities most no. of orders



4. How are customers distributed in Brazil





RUN SAVE SHARE SCHEDULE MORE

```
SELECT
  count(distinct customer_City) as Number_Of_Cities
FROM `scaler-target-project.customers.data`
```

Query results

SAVE RESU

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Number_Of_Cities
1	4119

*Unsaved ... ery

DISABLE EDITOR TABS

RUN SAVE SHARE SCHEDULE MORE

This query will process 3.6 MB when run.

```
SELECT
  customer_state,
  count(customer_id) as Number_Of_Customers
FROM `scaler-target-project.customers.data`
group by customer_state
order by Number_Of_Customers desc
```

Press Alt+F1 for Accessibility Options.

Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	Number_Of_Customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380

Results per page: 50 1 – 27 of 27

RUN SAVE SHARE SCHEDULE MORE

```
1  SELECT
2    customer_city,
3    count(customer_id) as Number_Of_Customers
4  FROM `scaler-target-project.customers.data`
5  group by customer_city
6  order by Number_Of_Customers desc
```

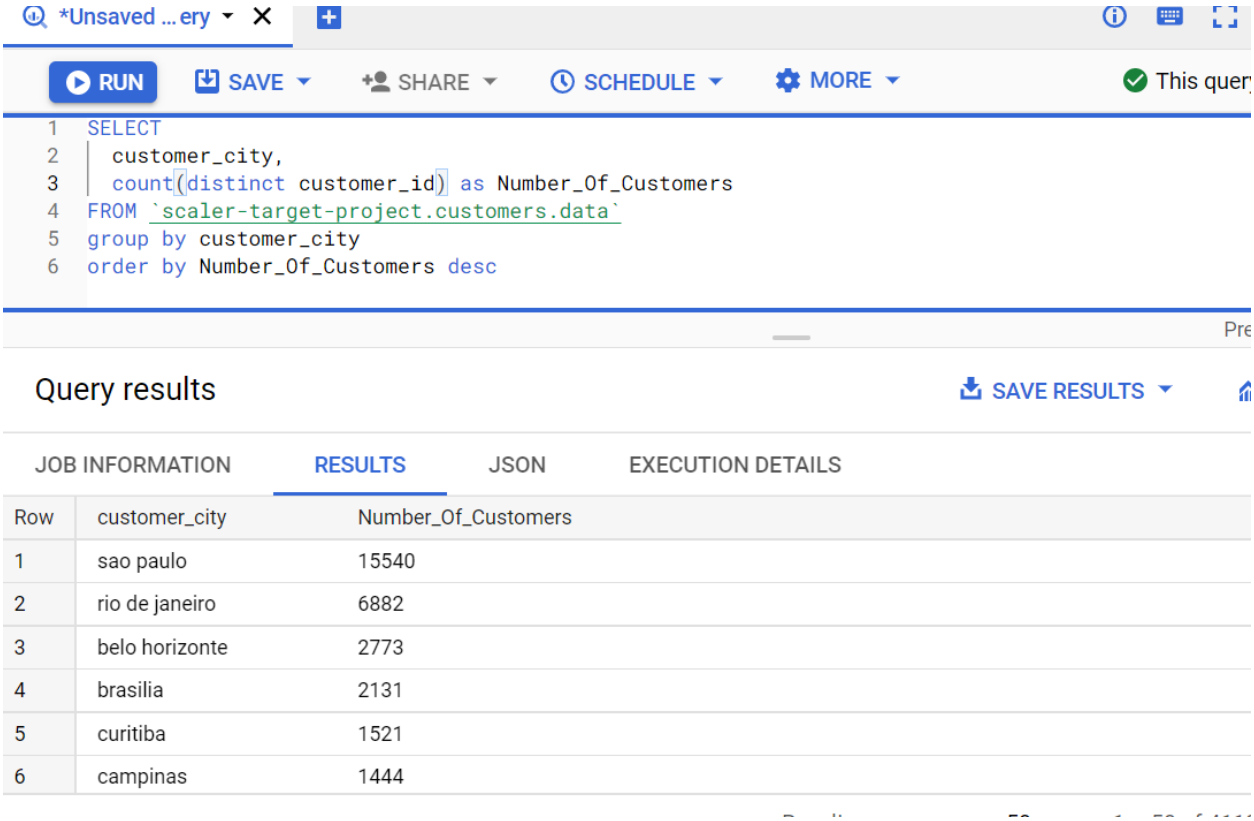
Query results

SAVE RESU

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_city	Number_Of_Customers
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444

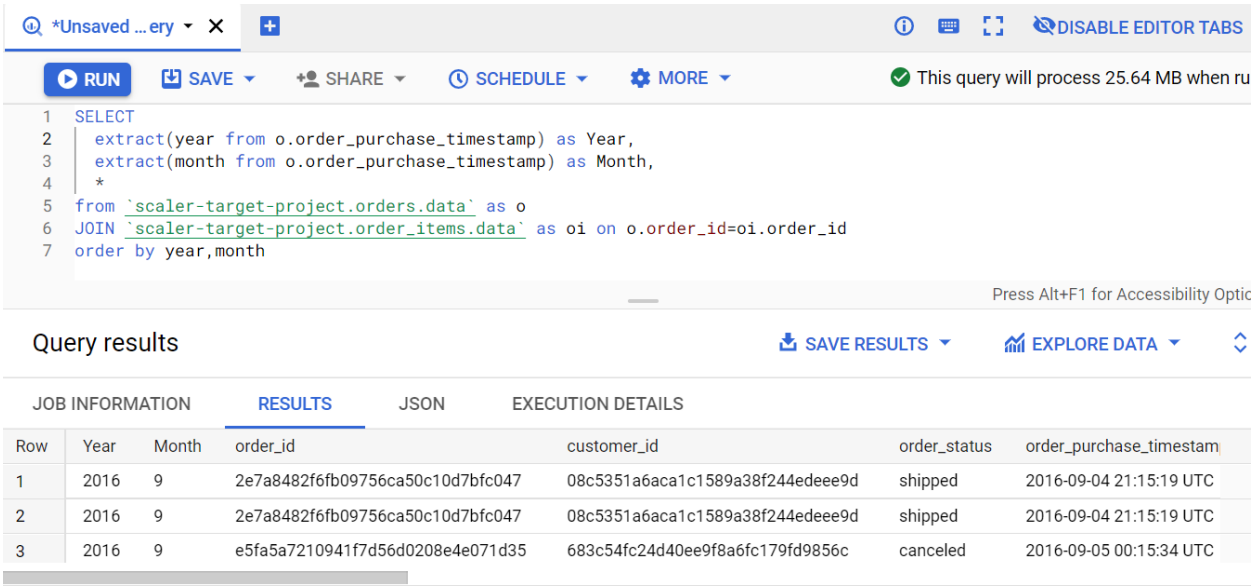
5. City wise number of unique customers



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

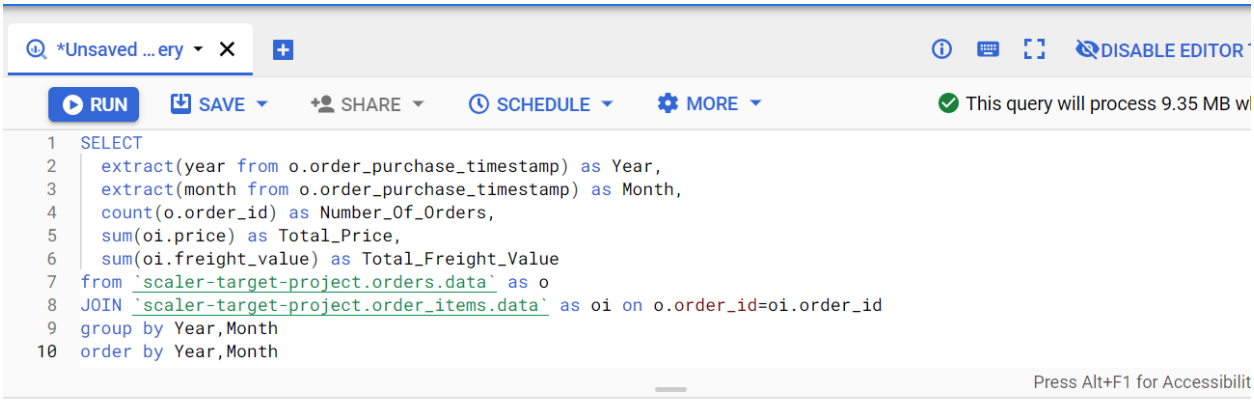
Step 1: Using CTE

1. “order_items” + “order” joined on order id where order_purchase timestamp is already divided into month & year



Row	Year	Month	order_id	customer_id	order_status	order_purchase_timestam
1	2016	9	2e7a8482f6fb09756ca50c10d7bfc047	08c5351a6aca1c1589a38f244edeee9d	shipped	2016-09-04 21:15:19 UTC
2	2016	9	2e7a8482f6fb09756ca50c10d7bfc047	08c5351a6aca1c1589a38f244edeee9d	shipped	2016-09-04 21:15:19 UTC
3	2016	9	e5fa5a7210941f7d56d0208e4e071d35	683c54fc24d40ee9f8a6fc179fd9856c	canceled	2016-09-05 00:15:34 UTC

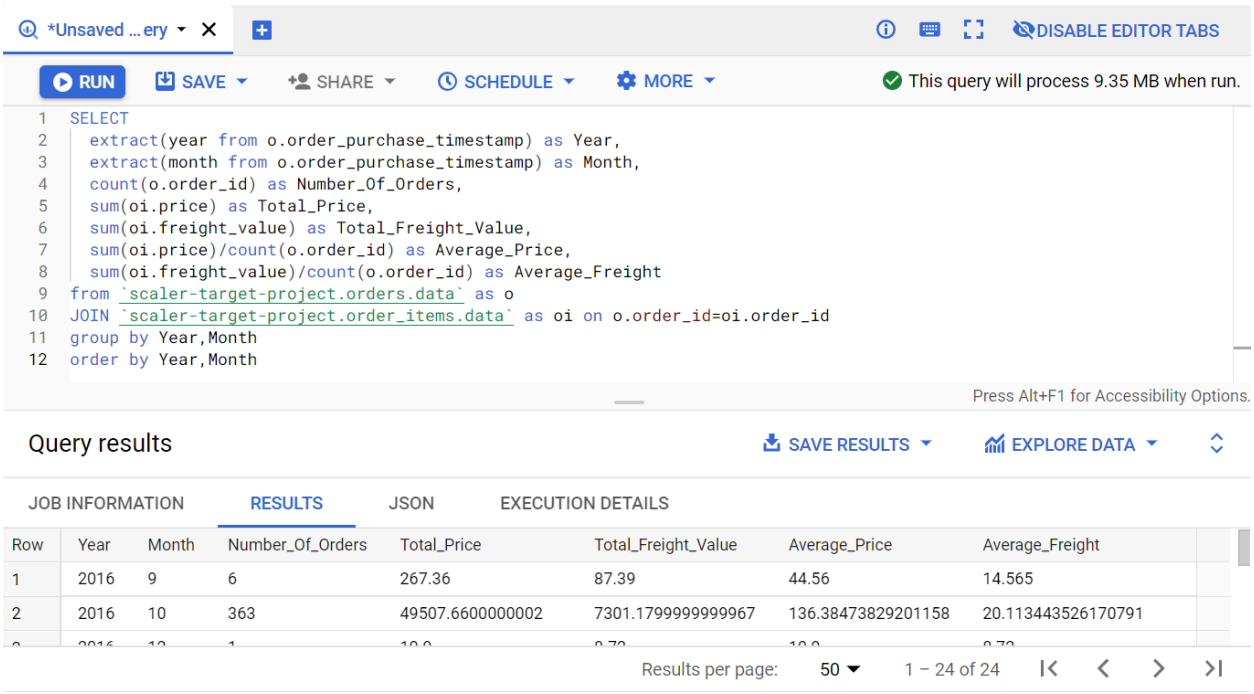
2. Group data by year and month, aggregation count(order_id), sum(price), sum(freight_value)



Row	Year	Month	Number_Of_Orders	Total_Price	Total_Freight_Value
1	2016	9	6	267.36	87.39
2	2016	10	363	49507.6600000002	7301.1799999999967
3	2016	12	1	10.9	8.72
4	2017	1	955	120312.86999999965	16875.61999999997

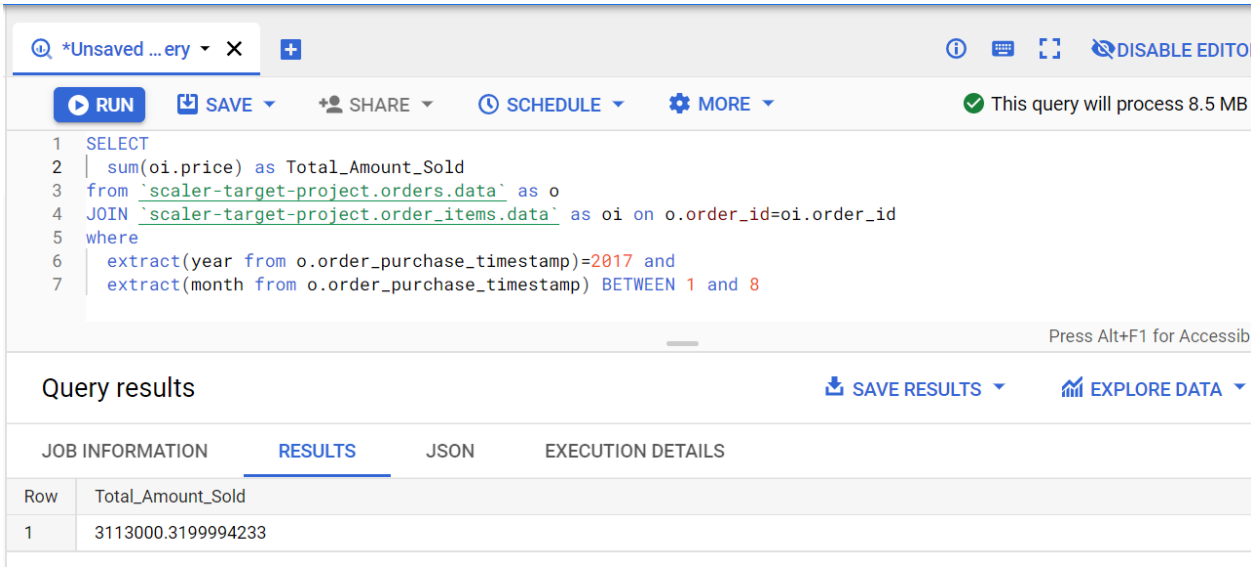
Results per page: 50 1 – 24 of 24

2. Create new columns:
price_per_order = sum(price) / count(order_id)
freight_per_order= sum(freight_value) / count(order_id)



Step 2: Answer the following questions:

1. Total amount sold in 2017 between Jan to August



2. Total amount sold in 2018 between Jan to august

RUN SAVE SHARE SCHEDULE MORE This query will process 8

```sql
SELECT
  sum(oi.price) as Total_Amount_Sold
from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.order_items.data` as oi on o.order_id=oi.order_id
where
  extract(year from o.order_purchase_timestamp)=2018 and
  extract(month from o.order_purchase_timestamp) BETWEEN 1 and 8
```

Press Alt+F1 for Ac

Query results SAVE RESULTS EXPLORE DA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Total_Amount_Sold
1	7385905.8000043072

3. % increase from 2017 to 2018

RUN SAVE SHARE SCHEDULE MORE This query will process 8

```sql
WITH Sales_2017 as (SELECT
  sum(oi.price) as Total_Amount_Sold from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.order_items.data` as oi on o.order_id=oi.order_id
where
  extract(year from o.order_purchase_timestamp)=2017 and
  extract(month from o.order_purchase_timestamp) BETWEEN 1 and 8),

Sales_2018 as (SELECT
  sum(oi.price) as Total_Amount_Sold from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.order_items.data` as oi on o.order_id=oi.order_id
where
  extract(year from o.order_purchase_timestamp)=2018 and
  extract(month from o.order_purchase_timestamp) BETWEEN 1 and 8)

SELECT ((Sales_2018.Total_Amount_Sold/Sales_2017.Total_Amount_Sold)-1)*100 as Percentage_Increase
from Sales_2018,Sales_2017

```

Press Alt+F1 for Ac

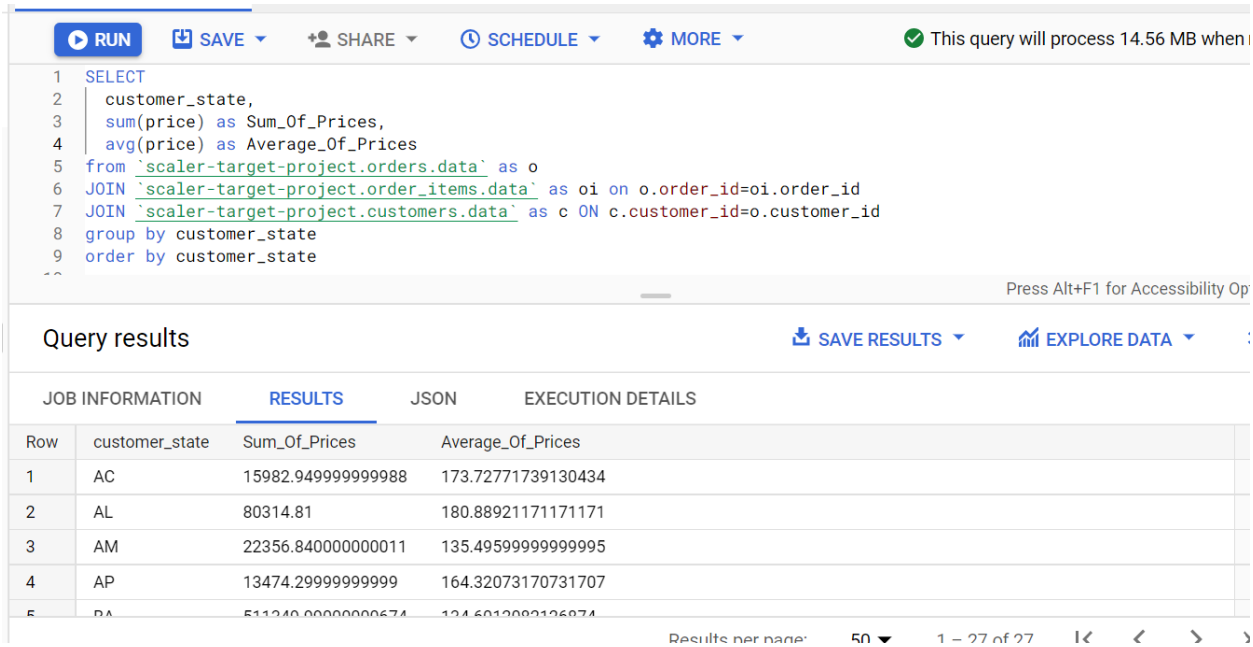
Query results SAVE RESULTS EXPLORE DA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

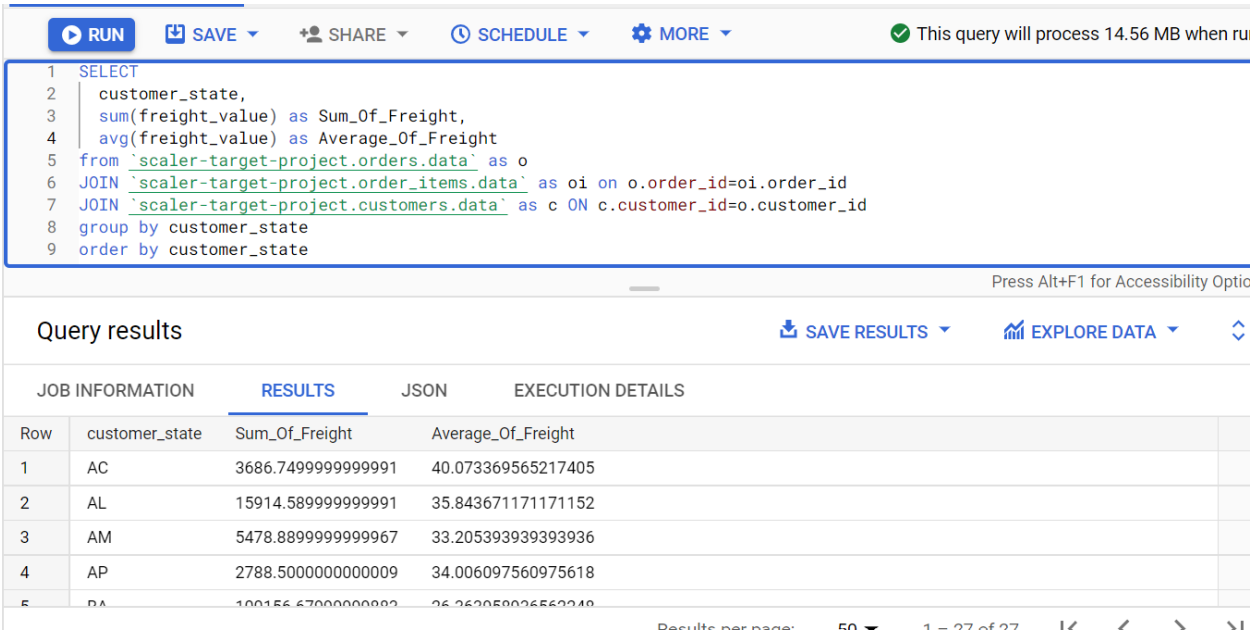
Row	Percentage_Increase
1	137.26003985791496

Step 3: Join (orders+order_items) table from previous step with “customers” table on Customer_id and find:

1. Mean & Sum of price by customer state

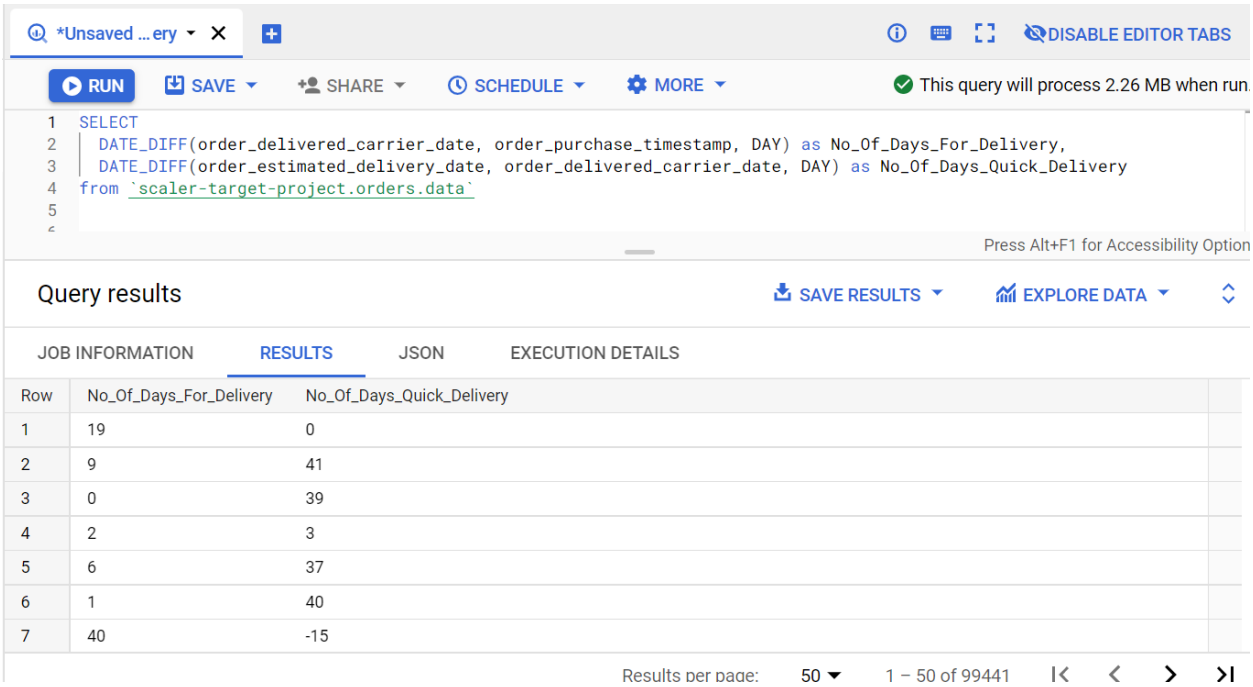


2. Mean & Sum of freight value by customer state



5. Analysis on sales, freight and delivery time

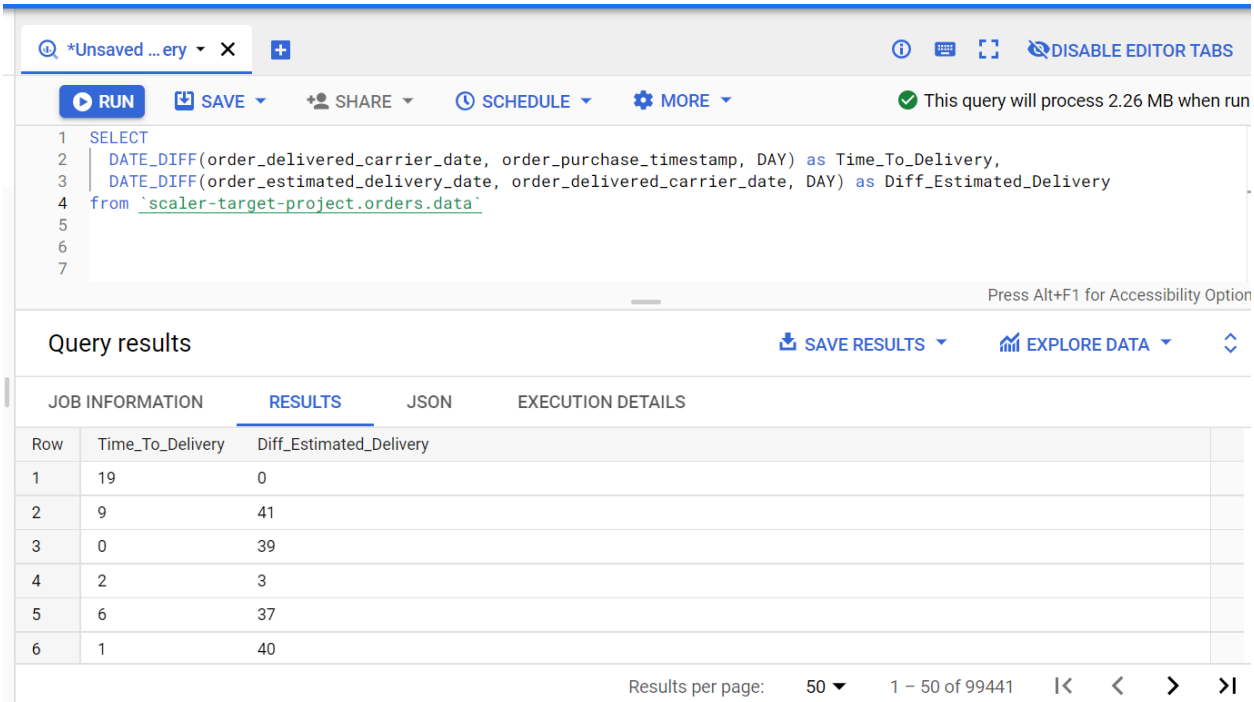
1. Calculating days between purchasing, delivering and estimated delivery



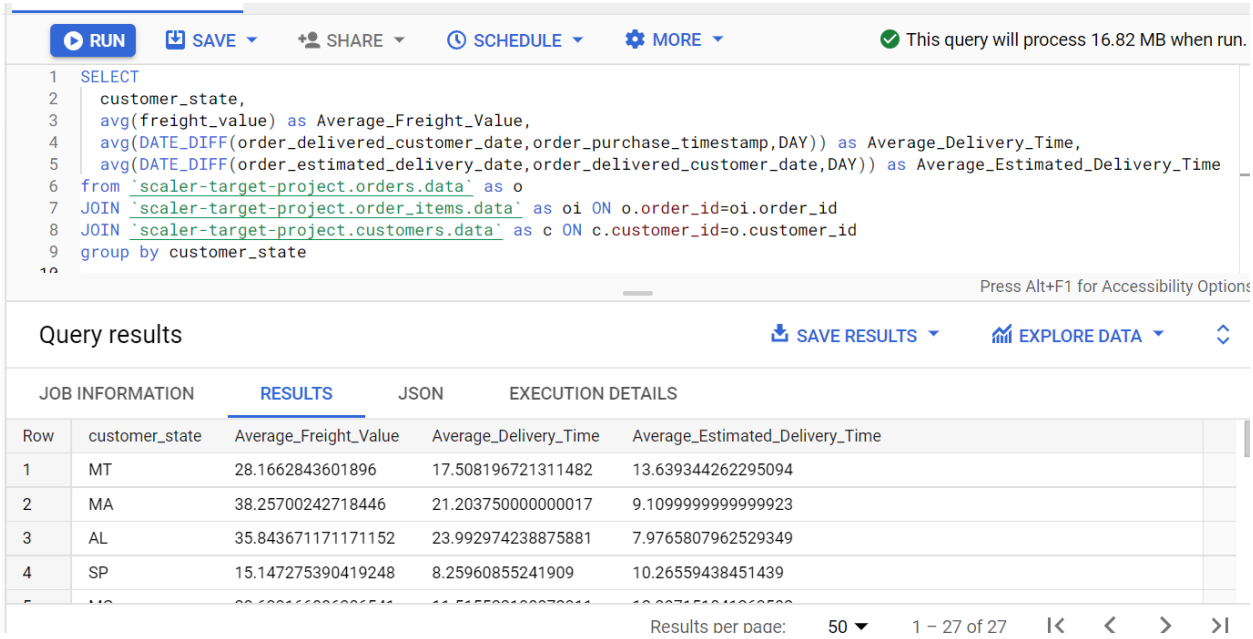
2. Create columns:

time_to_delivery = order_purchase_timestamp-order_delivered_customer_date

diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date



3. Grouping data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery



RUN SAVE SHARE SCHEDULE MORE This query will process 16.82 MB when run.

```
SELECT
  customer_state,
  avg(freight_value) as Average_Freight_Value,
  avg(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)) as Average_Delivery_Time,
  avg(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)) as Average_Estimated_Delivery_Time
from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.order_items.data` as oi ON o.order_id=oi.order_id
JOIN `scaler-target-project.customers.data` as c ON c.customer_id=o.customer_id
group by customer_state
```

Press Alt+F1 for Accessibility Options

Query results SAVE RESULTS EXPLORE DATA

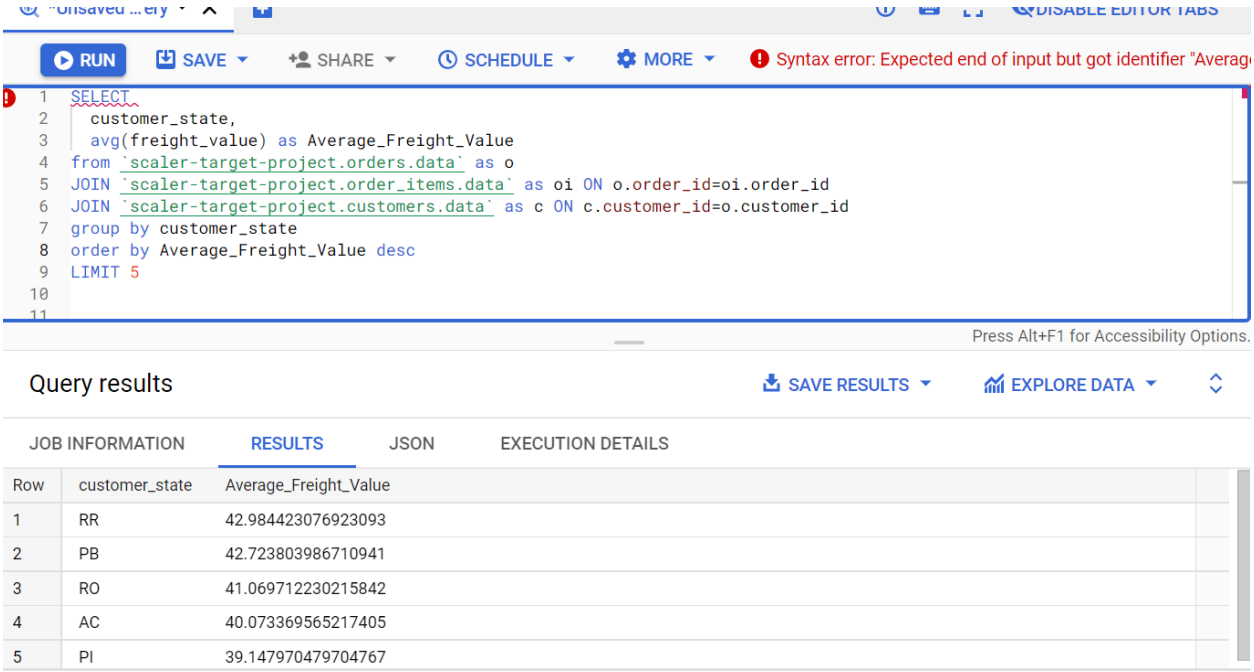
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	Average_Freight_Value	Average_Delivery_Time	Average_Estimated_Delivery_Time
1	MT	28.1662843601896	17.508196721311482	13.639344262295094
2	MA	38.25700242718446	21.203750000000017	9.1099999999999923
3	AL	35.843671171171152	23.992974238875881	7.9765807962529349
4	SP	15.147275390419248	8.25960855241909	10.26559438451439

Results per page: 50 1 – 27 of 27

4. Sort the data to get the following:

a. Top 5 states with highest/lowest average freight value



RUN SAVE SHARE SCHEDULE MORE Syntax error: Expected end of input but got identifier "Averag

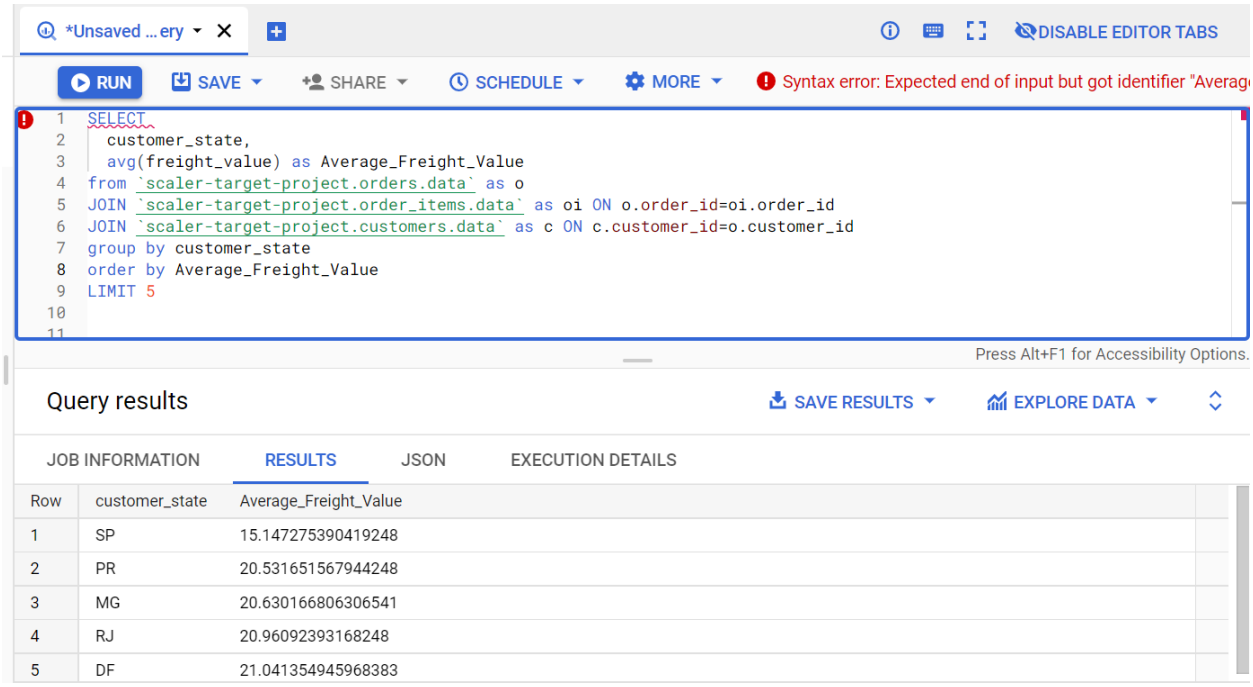
```
SELECT
  customer_state,
  avg(freight_value) as Average_Freight_Value
from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.order_items.data` as oi ON o.order_id=oi.order_id
JOIN `scaler-target-project.customers.data` as c ON c.customer_id=o.customer_id
group by customer_state
order by Average_Freight_Value desc
LIMIT 5
```

Press Alt+F1 for Accessibility Options.

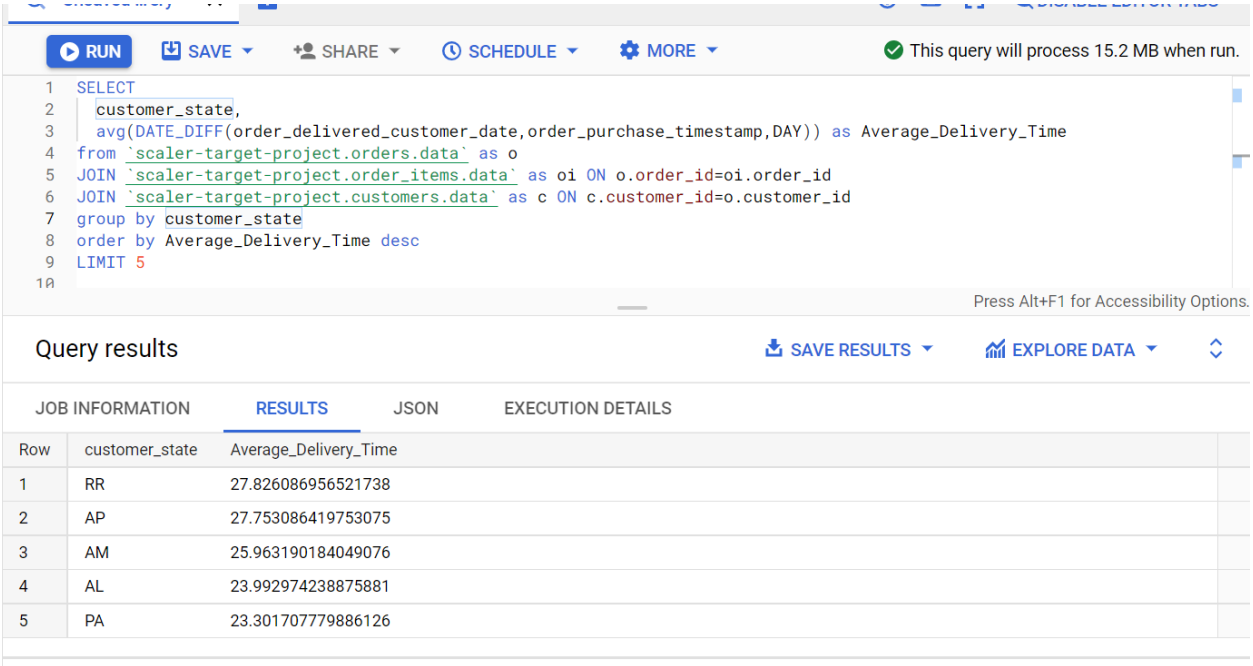
Query results SAVE RESULTS EXPLORE DATA

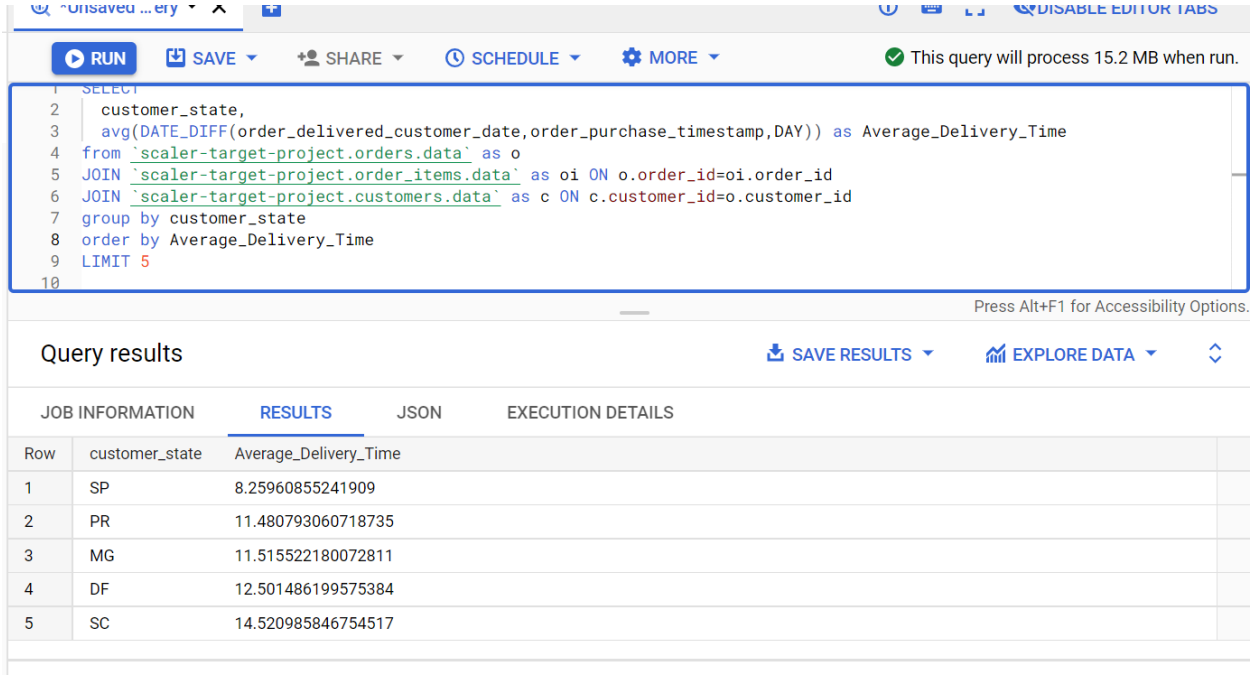
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	Average_Freight_Value
1	RR	42.984423076923093
2	PB	42.723803986710941
3	RO	41.069712230215842
4	AC	40.073369565217405
5	PI	39.147970479704767



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

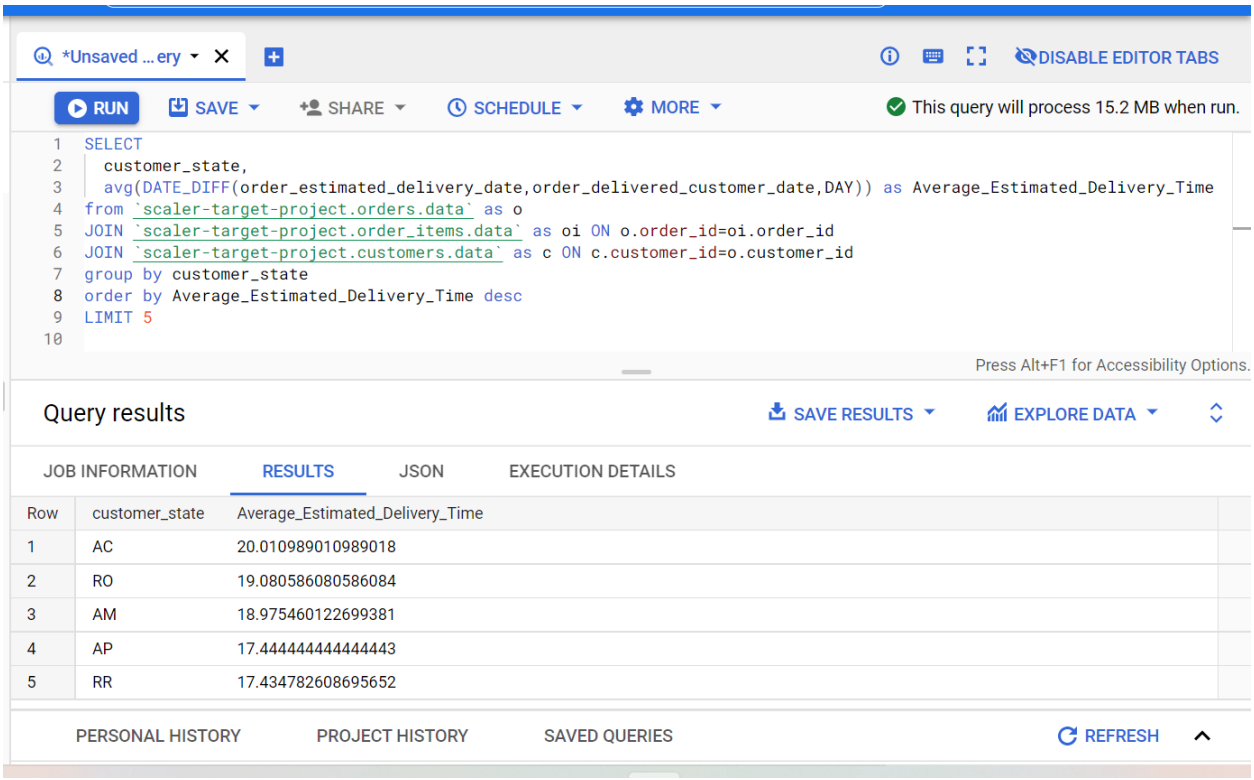




```
SELECT
  customer_state,
  avg(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY)) as Average_Delivery_Time
from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.order_items.data` as oi ON o.order_id=oi.order_id
JOIN `scaler-target-project.customers.data` as c ON c.customer_id=o.customer_id
group by customer_state
order by Average_Delivery_Time
LIMIT 5
```

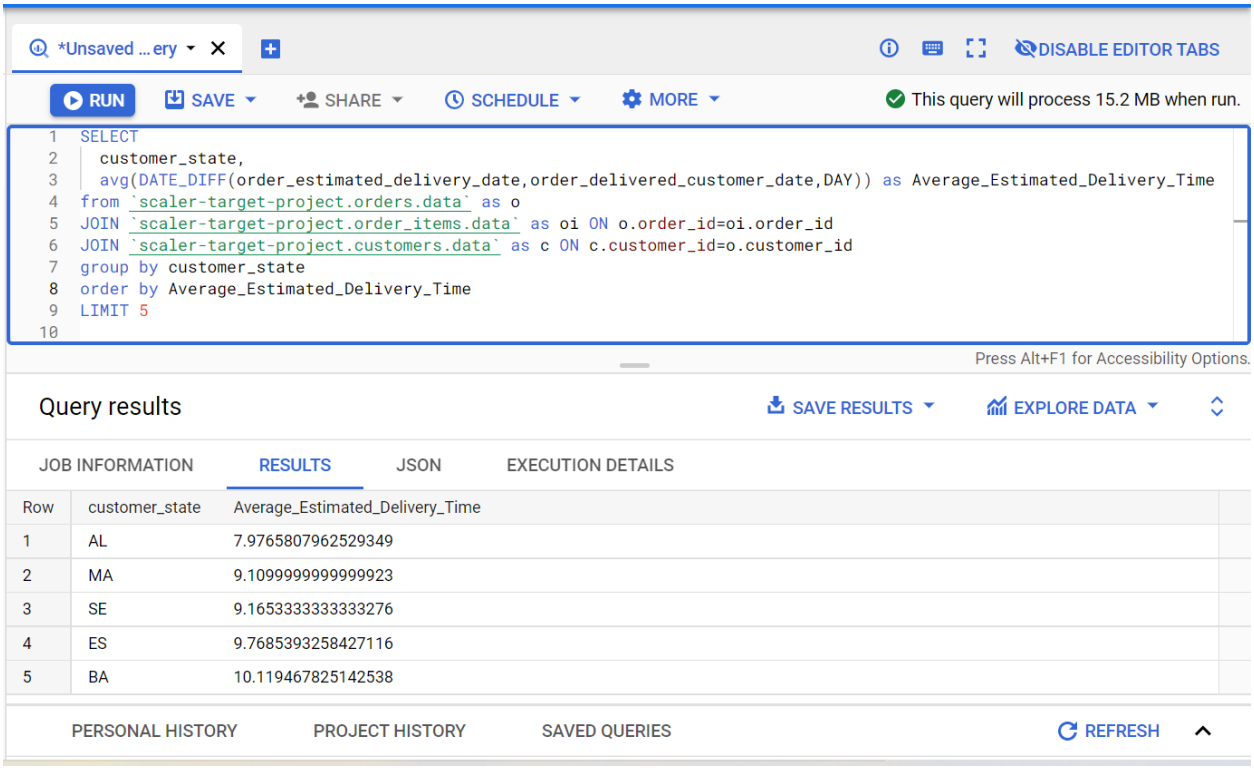
Row	customer_state	Average_Delivery_Time
1	SP	8.25960855241909
2	PR	11.480793060718735
3	MG	11.515522180072811
4	DF	12.501486199575384
5	SC	14.520985846754517

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



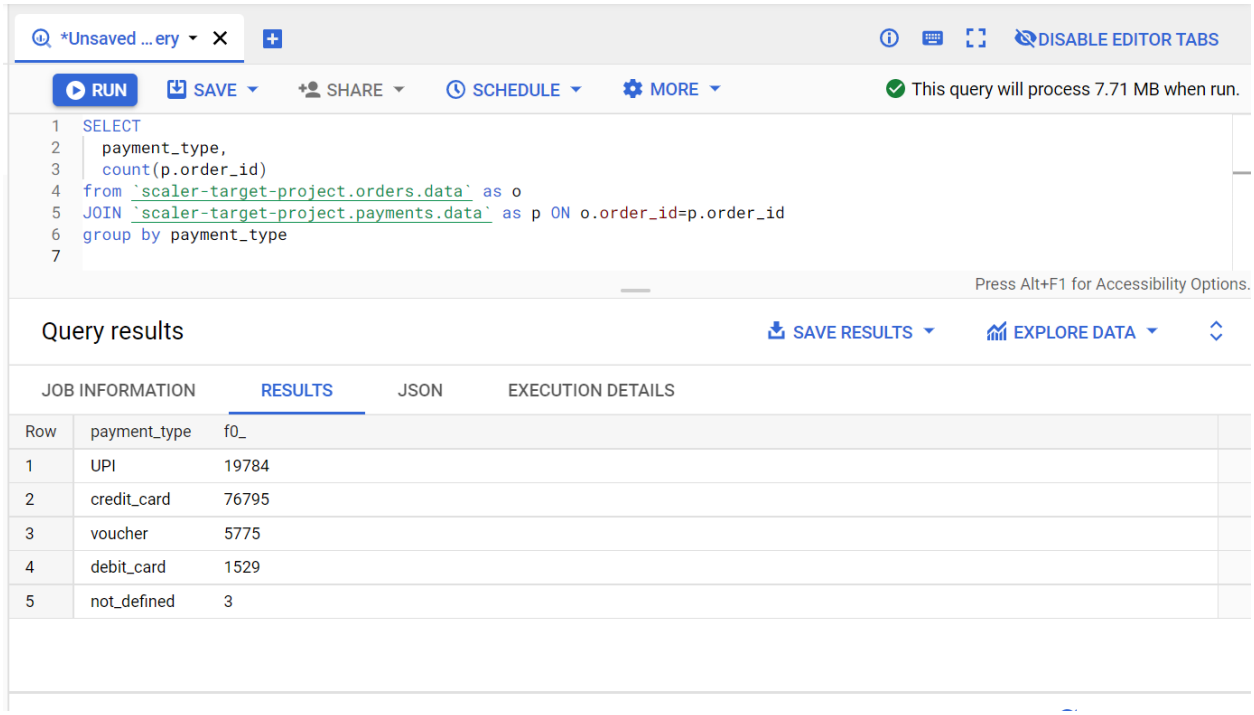
```
SELECT
  customer_state,
  avg(DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY)) as Average_Estimated_Delivery_Time
from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.order_items.data` as oi ON o.order_id=oi.order_id
JOIN `scaler-target-project.customers.data` as c ON c.customer_id=o.customer_id
group by customer_state
order by Average_Estimated_Delivery_Time desc
LIMIT 5
```

Row	customer_state	Average_Estimated_Delivery_Time
1	AC	20.010989010989018
2	RO	19.080586080586084
3	AM	18.975460122699381
4	AP	17.444444444444443
5	RR	17.434782608695652



5. Payment type analysis: Join “payments” dataset with the existing data on order_id

a. Count of orders for different payment types



b. Distribution of payment installments and count of orders

RUN SAVE SHARE SCHEDULE MORE This query will process 7.39 MB when run.

```
SELECT
  payment_installments,
  count(p.order_id)
from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.payments.data` as p ON o.order_id=p.order_id
group by payment_installments
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_installments	f0_
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239

Results per page: 50 1 – 24 of 24

c. Count of orders for different payment types Month over Month

RUN SAVE SHARE SCHEDULE MORE This query will process 8.47 MB when run.

```
SELECT
  payment_type,
  extract(month from order_purchase_timestamp) as Month,
  count(p.order_id) as No_Of_Orders
from `scaler-target-project.orders.data` as o
JOIN `scaler-target-project.payments.data` as p ON o.order_id=p.order_id
group by payment_type,month
order by payment_type,month
```

Press Alt+F1 for Accessibility Options.

Query results SAVE RESULTS EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	payment_type	Month	No_Of_Orders
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035

Results per page: 50 1 – 50 of 50

Actionable Insights:

- There are 4119 cities in 27 states to which orders are getting delivered.
- There are 611 cities in 23 states from which sellers sell their goods.
- 609 orders are unavailable.
- The numbers of orders are increasing year in year.
- Brazilians mostly buy on Saturday, Sunday and Monday.
- Brazilians mostly buy in afternoon and night.
- Most orders are placed from states “SP”, “RJ” and “MG”.
- Least orders are placed from “RR”, “AP” and “AC”.
- Most orders are placed from cities “sao Paulo”, “rio de janeiro” and “belo horizonte”
- Highest average Freight value is for states such as “RR”, “PB” and “RO”.
- Lowest average Freight value is for states such as “SP”, “PR” and “MG”.
- Highest average sales is for states such as “AC”, “PB” and “AL”.
- Lowest average sales is for states such as “SP”, “PR” and “RS”.
- Highest average delivery time is for states “RR”, “AP” and “AM”.
- Lowest average delivery time is for states “SP”, “PR” and “MG”.
- Most payments are done using Credit Card and UPI.
- Most people prefer no of installments of 1, 2 and 3.

Recommendations:

- ✓ There are total 23 seller states, but no of customer states are 27. More sellers can be identified in the other states so that the freight value can be decreased and also the delivery time.
- ✓ With the same logic as above, more sellers can be identified in various other small and big cities to reduce the freight value and also the delivery time as there are 4119 customer states and 611 seller states.
- ✓ Target can get more variety of goods and keep abundant stock of all goods since 609 orders were unavailable during the time period.
- ✓ Since Brazilians mostly buy on weekends and Monday, therefore Target can provide discounts or offers on weekdays to influence people to buy more on weekdays.
- ✓ Since Brazilians mostly buy in afternoon and night, therefore Target can provide discounts or offers in morning to influence people to buy more in mornings.
- ✓ More offers and discounts can be offered to people living in states like “RR”, “AP” and “AC” to influence them to buy more since these are the states from where lowest orders are placed.
- ✓ More sellers can be identified in state such as “RR”, “PB” and “RO” to decrease the freight value since these are the states with highest average freight value.
- ✓ More offers and discounts can be offered to people living in states “SP”, “PR” and “RS” because these are the states where the average sales is the lowest.
- ✓ Highest average delivery time is for states “RR”, “AP” and “AM”. More sellers can be identified in these states to bring down the delivery time, which will also help in reducing the freight value.
- ✓ Most payments are done using Credit Card. Target can offer some offers to influence more people to pay using UPI.