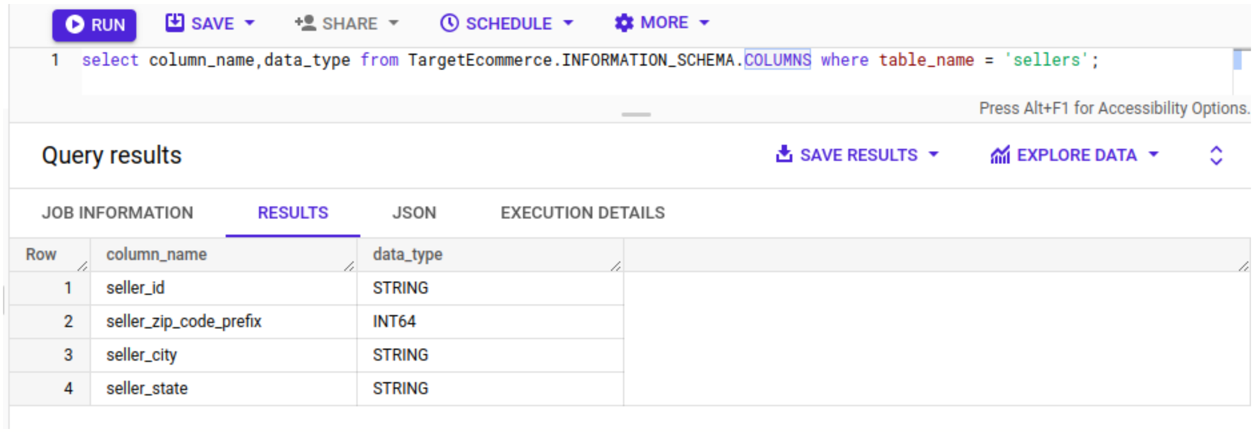
Sarwar Alam

1. **Initial Exploration of data(Checking data types of columns,time period of the data,and cities and states)**

Data types:

```
Table: sellers
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where
table_name = 'sellers';
```

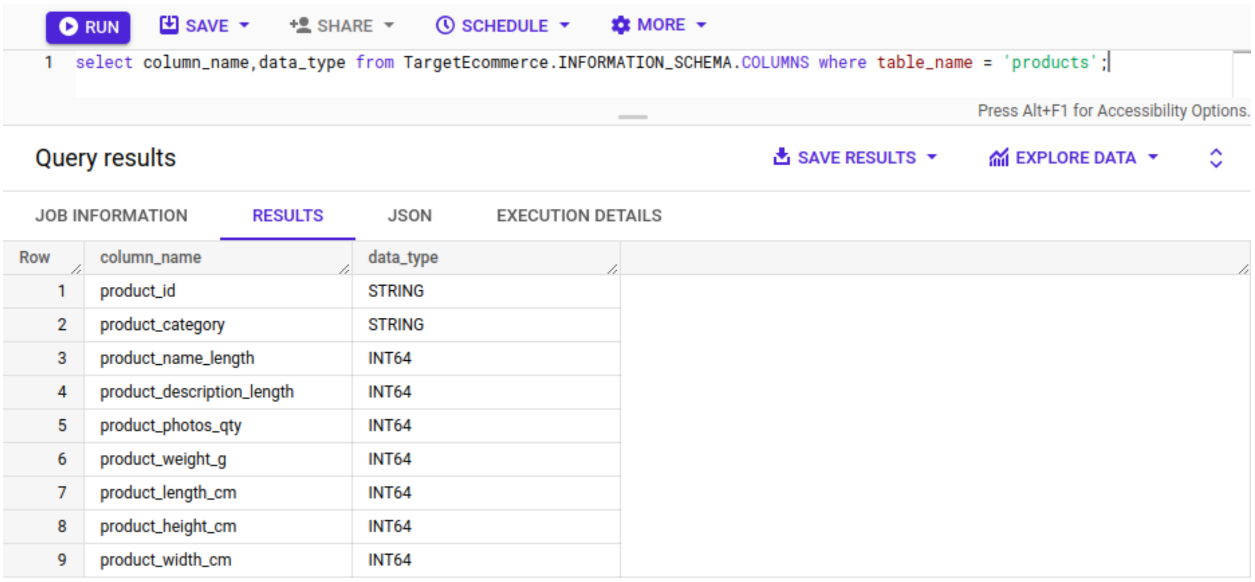


```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where table_name = 'sellers';
```

Row	column_name	data_type
1	seller_id	STRING
2	seller_zip_code_prefix	INT64
3	seller_city	STRING
4	seller_state	STRING

Table:products

```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where
table_name = 'products';
```

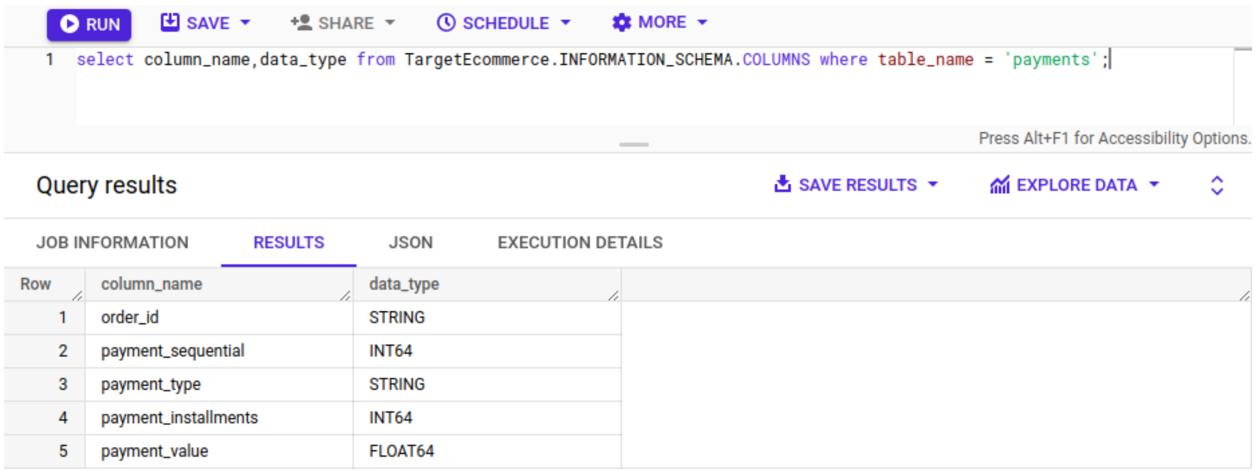


```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where table_name = 'products';
```

Row	column_name	data_type
1	product_id	STRING
2	product_category	STRING
3	product_name_length	INT64
4	product_description_length	INT64
5	product_photos_qty	INT64
6	product_weight_g	INT64
7	product_length_cm	INT64
8	product_height_cm	INT64
9	product_width_cm	INT64

Table:payments

```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where
table_name = 'payments';
```

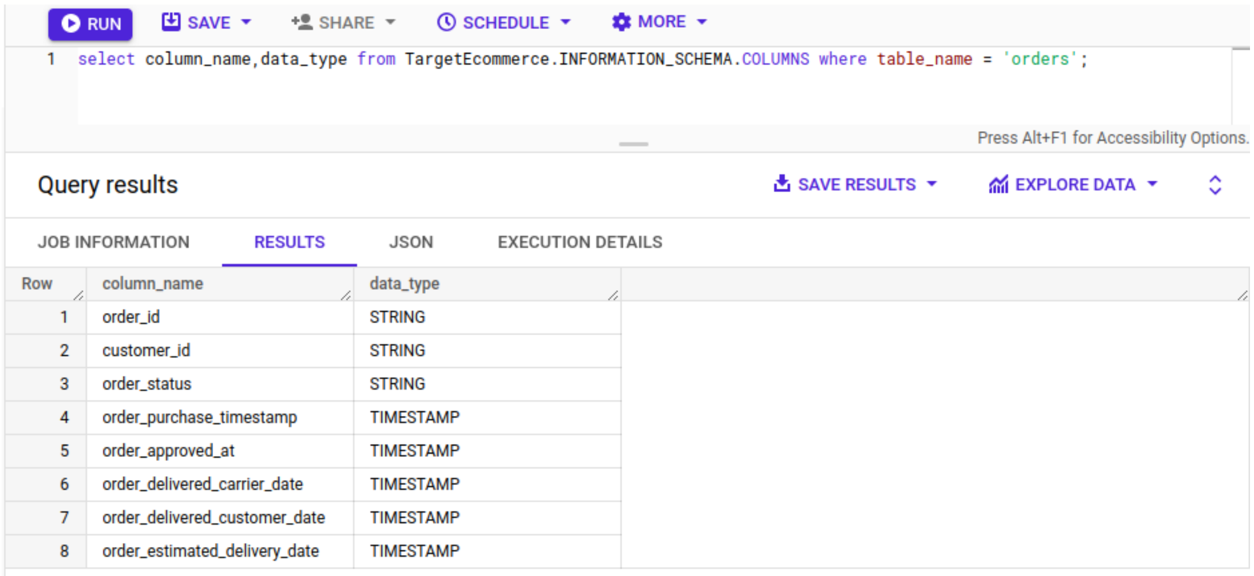


```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where table_name = 'payments';
```

Row	column_name	data_type
1	order_id	STRING
2	payment_sequential	INT64
3	payment_type	STRING
4	payment_installments	INT64
5	payment_value	FLOAT64

Table:orders

```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where
table_name = 'orders';
```

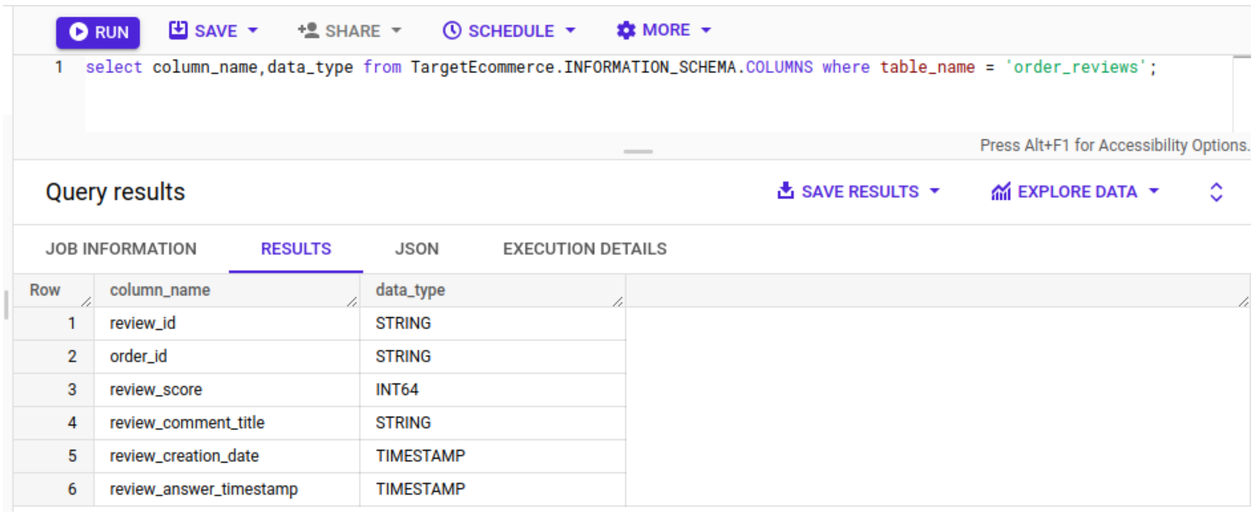


```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where table_name = 'orders';
```

Row	column_name	data_type
1	order_id	STRING
2	customer_id	STRING
3	order_status	STRING
4	order_purchase_timestamp	TIMESTAMP
5	order_approved_at	TIMESTAMP
6	order_delivered_carrier_date	TIMESTAMP
7	order_delivered_customer_date	TIMESTAMP
8	order_estimated_delivery_date	TIMESTAMP

Table:order_reviews

```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where
table_name = 'order_reviews';
```

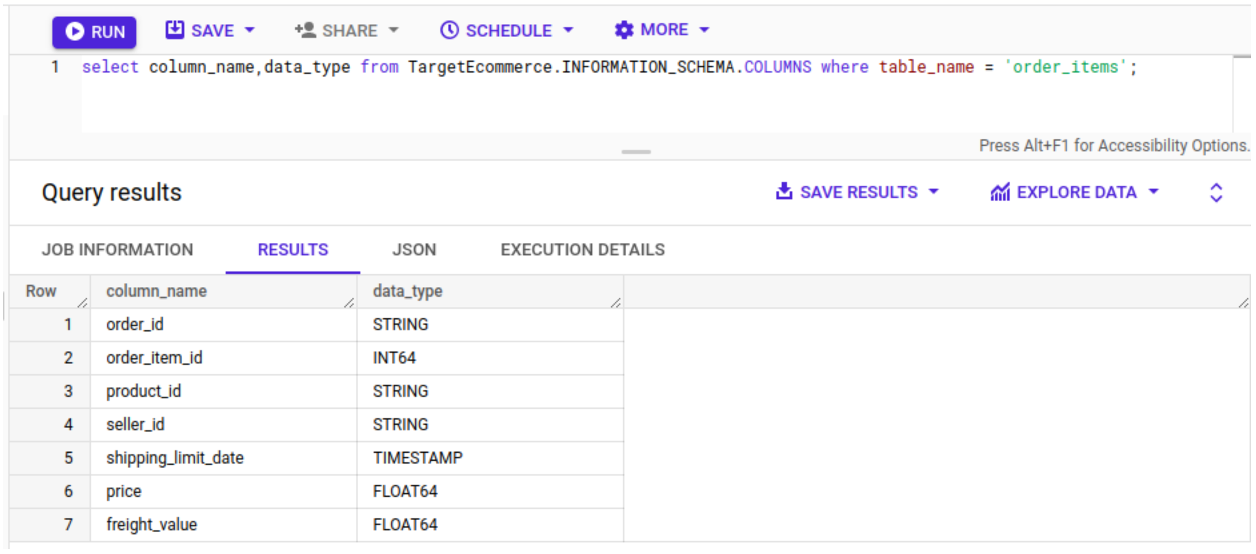


```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where table_name = 'order_reviews';
```

Row	column_name	data_type
1	review_id	STRING
2	order_id	STRING
3	review_score	INT64
4	review_comment_title	STRING
5	review_creation_date	TIMESTAMP
6	review_answer_timestamp	TIMESTAMP

Table:order_items

```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where
table_name = 'order_items';
```

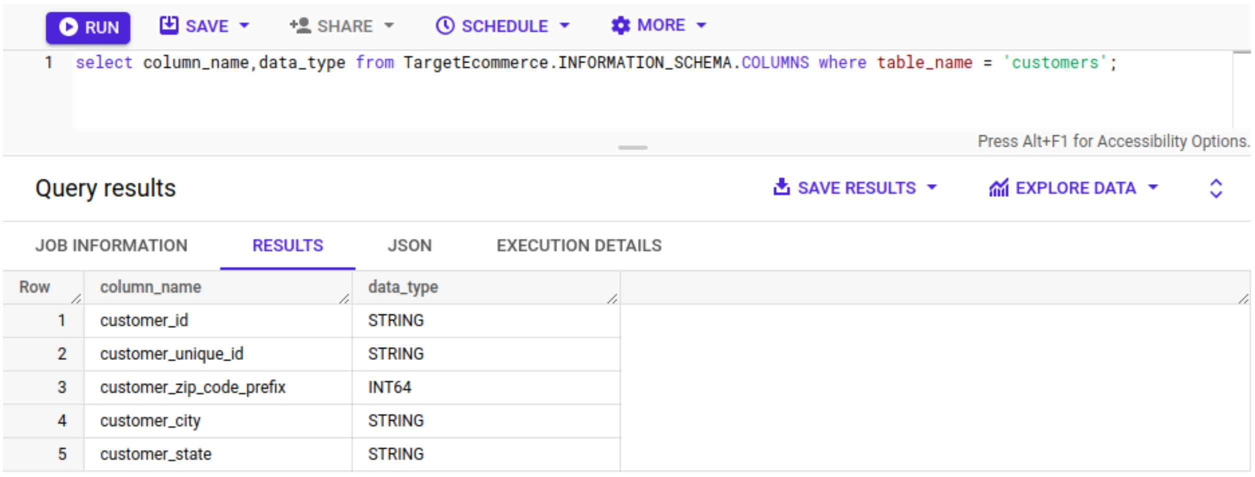


```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where table_name = 'order_items';
```

Row	column_name	data_type
1	order_id	STRING
2	order_item_id	INT64
3	product_id	STRING
4	seller_id	STRING
5	shipping_limit_date	TIMESTAMP
6	price	FLOAT64
7	freight_value	FLOAT64

Table:customers

```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where
table_name = 'customers';
```



```
select column_name,data_type from TargetEcommerce.INFORMATION_SCHEMA.COLUMNS where table_name = 'customers';
```

Query results

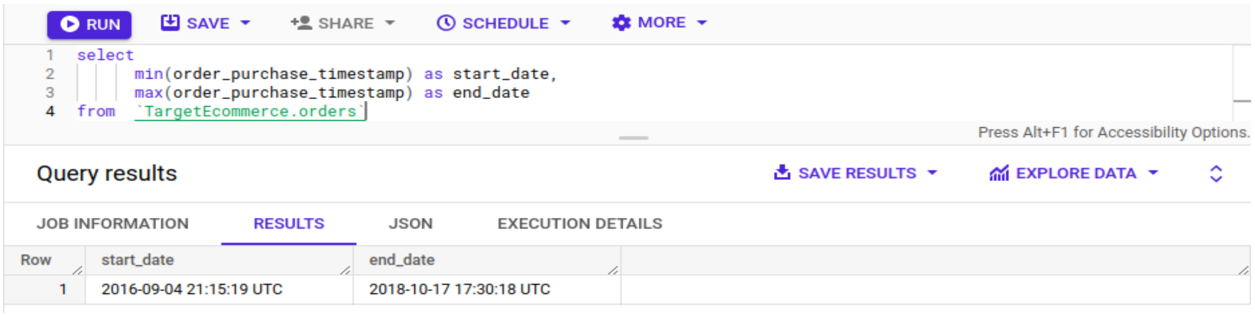
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

Time period for which the data is given:

We need to find when the first purchase and the last purchase in the given data took place.This would be the time period of the whole data.

```
 select

      min(order_purchase_timestamp) as start_date,

      max(order_purchase_timestamp) as end_date

from  `TargetEcommerce.orders`
```

Time period: From **2016-09-04 21:15:19 UTC** to **2018-10-17 17:30:18 UTC**



```
select
    min(order_purchase_timestamp) as start_date,
    max(order_purchase_timestamp) as end_date
from `TargetEcommerce.orders`
```

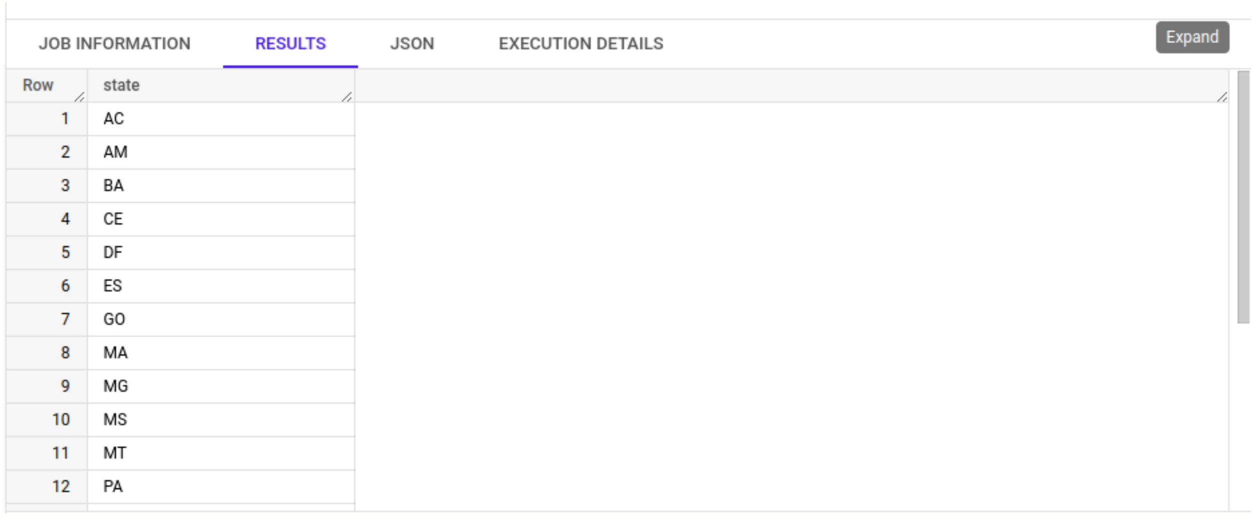
Query results

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

Cities and States covered in the dataset:

States:

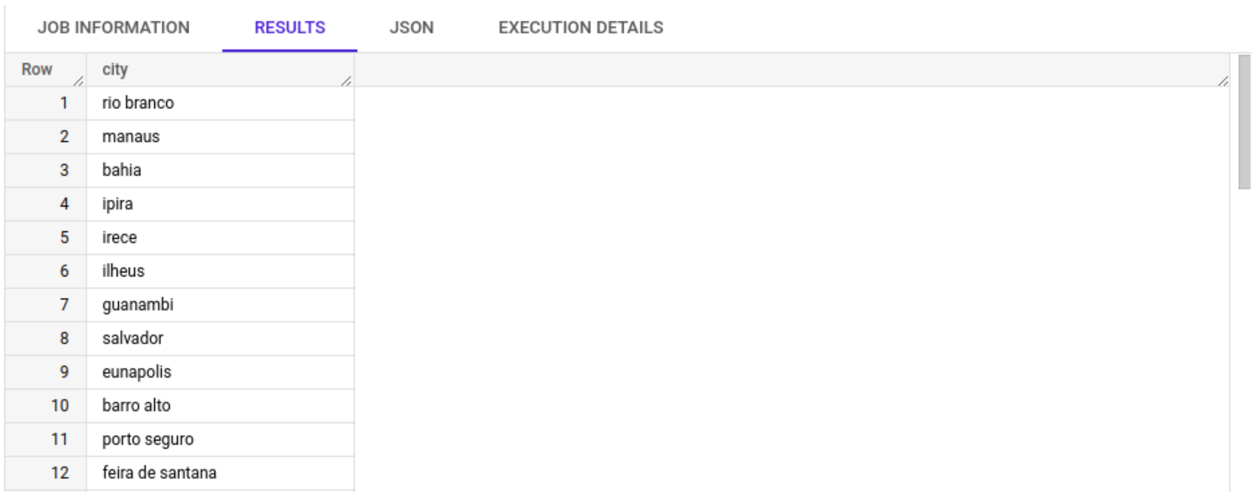
```
select
  distinct *
from (select
     customer_state as state
from  `TargetEcommerce.customers`
UNION ALL
select
     seller_state as state
from  `TargetEcommerce.sellers`)
```



Row	state
1	AC
2	AM
3	BA
4	CE
5	DF
6	ES
7	GO
8	MA
9	MG
10	MS
11	MT
12	PA

Cities:

```
select
  distinct *
from (select
     customer_city as city
from  `TargetEcommerce.customers`
UNION ALL
select
     seller_city as city
from  `TargetEcommerce.sellers`)
```



JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	city
1	rio branco
2	manaus
3	bahia
4	ipira
5	irece
6	ilheus
7	guanambi
8	salvador
9	eunapolis
10	barro alto
11	porto seguro
12	feira de santana

2.In-depth Exploration:

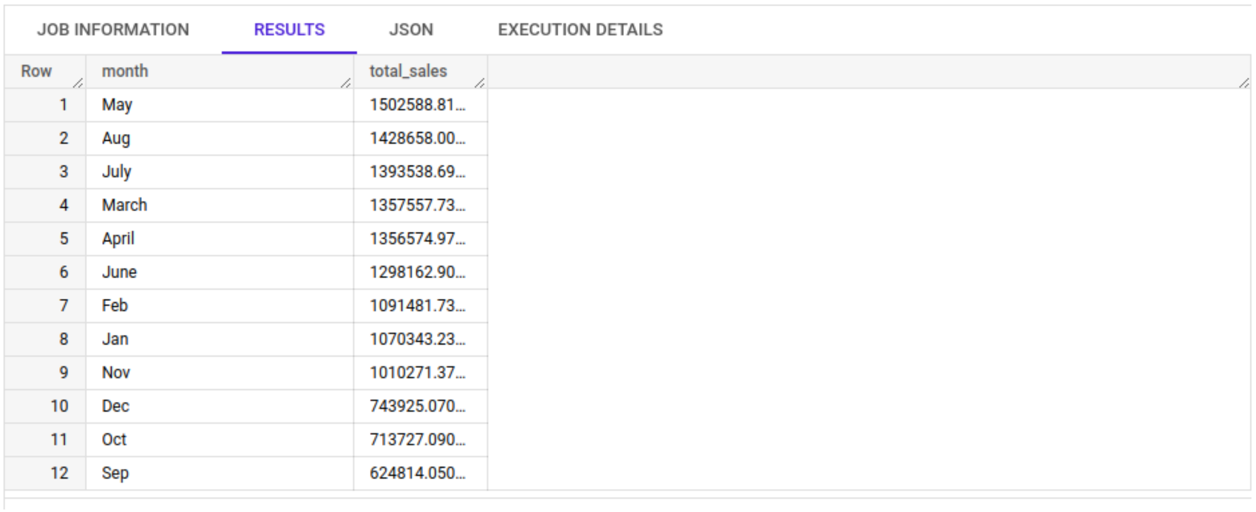
1. <u>**Trend on e-commerce in Brazil:**</u>

   a.To find trend ,we can have the total sales for each month(from year 2016 to 2018):

   Total purchase has been sorted in descending order.

```
with price_month AS (select
    (CASE
        WHEN extract(MONTH FROM o.order_purchase_timestamp)=1
        THEN 'Jan'
        WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

        THEN 'Feb'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

        THEN 'March'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

        THEN 'April'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

        THEN 'May'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

        THEN 'June'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

        THEN 'July'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

        THEN 'Aug'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

        THEN 'Sep'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

        THEN 'Oct'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

        THEN 'Nov'

        ELSE 'Dec'

     END)  month,

    oi.price
```

```
from `TargetEcommerce.orders` o left join `TargetEcommerce.order_items` oi

on o.order_id=oi.order_id),



total_purchases AS (select month,sum(price) as total_sales from price_month
group by month)

select * from total_purchases order by total_sales desc;
```


JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	month	total_sales
1	May	1502588.81...
2	Aug	1428658.00...
3	July	1393538.69...
4	March	1357557.73...
5	April	1356574.97...
6	June	1298162.90...
7	Feb	1091481.73...
8	Jan	1070343.23...
9	Nov	1010271.37...
10	Dec	743925.070...
11	Oct	713727.090...
12	Sep	624814.050...

Sales in May is more than double of sales in Sep

Surely there is a trend of purchases.There is a trend(upward) from January to March but suddenly in April the purchases go down.From April to May upward trend but in June the purchase go down.For some time there is a upward trend but there is also downward trend.Like a random walk.

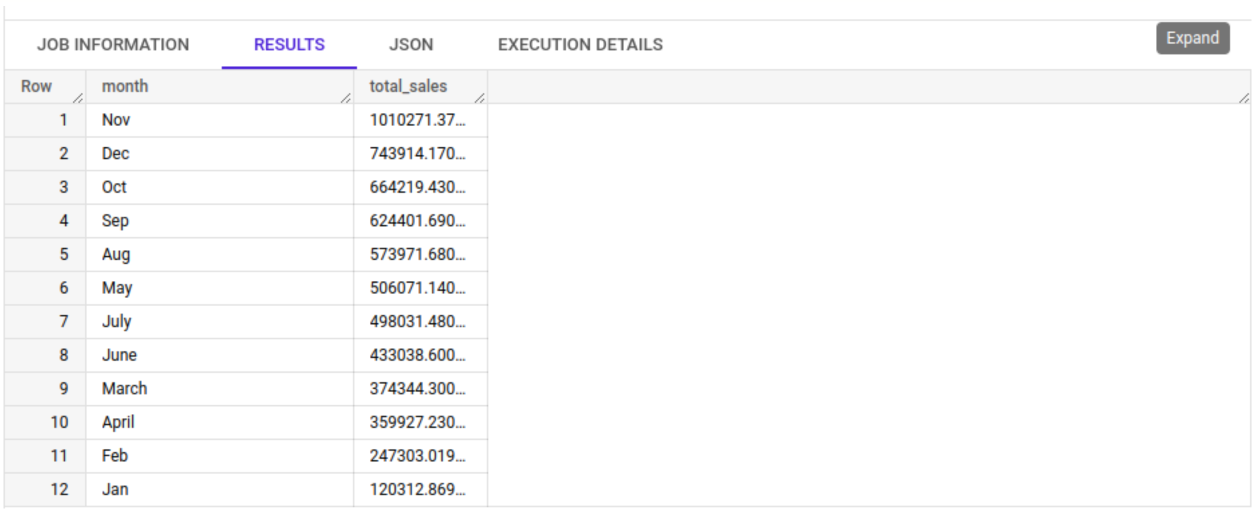
b.Also we will have a trend for each year(2017,2018) with respect to each month,the year 2016 has been excluded as the dataset has data only from Sep,Oct,Nov,and Dec.

Year 2017:Total purchase has been sorted in descending order

```
with price_month AS (select

    (CASE
```

```sql
WHEN extract(MONTH FROM o.order_purchase_timestamp)=1

THEN 'Jan'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

THEN 'Feb'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

THEN 'March'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

THEN 'April'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

THEN 'May'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

THEN 'June'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

THEN 'July'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

THEN 'Aug'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

THEN 'Sep'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

THEN 'Oct'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

THEN 'Nov'

ELSE 'Dec'
```

```
     END)  month,

     EXTRACT(YEAR FROM o.order_purchase_timestamp) as year,

    oi.price

from `TargetEcommerce.orders` o left join `TargetEcommerce.order_items` oi

on o.order_id=oi.order_id),



total_purchases AS (select month,sum(price) as total_sales from price_month
where year=2017 group by month)

select * from total_purchases order by total_sales desc;
```


JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | Expand

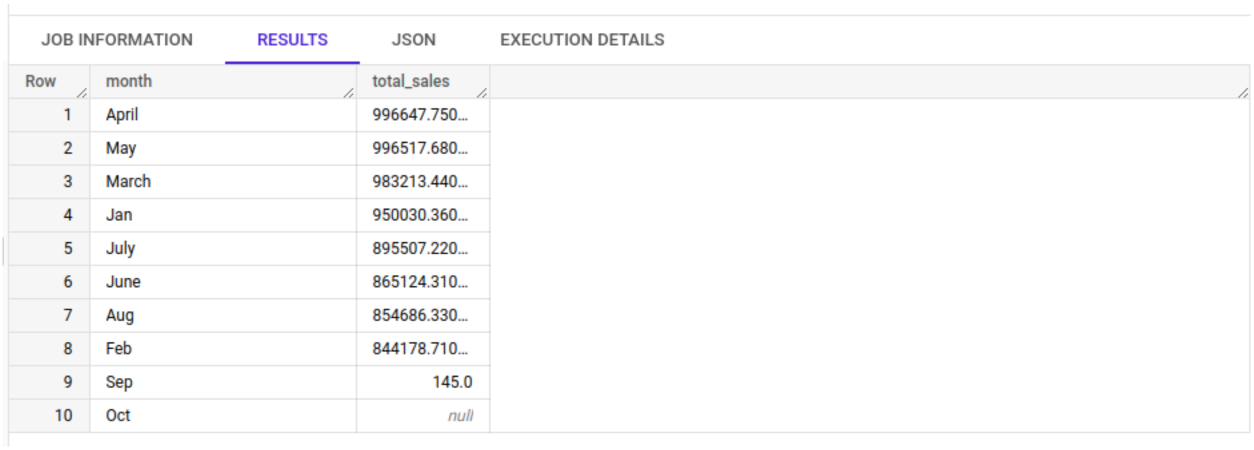
Row	month	total_sales
1	Nov	1010271.37...
2	Dec	743914.170...
3	Oct	664219.430...
4	Sep	624401.690...
5	Aug	573971.680...
6	May	506071.140...
7	July	498031.480...
8	June	433038.600...
9	March	374344.300...
10	April	359927.230...
11	Feb	247303.019...
12	Jan	120312.869...

In the year 2017,it's surprising that from January to Dec there is an upward trend.

Year 2018:Total purchase has been sorted in descending order

```
with price_month AS (select

    (CASE

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=1

        THEN 'Jan'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=2
        THEN 'Feb'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

        THEN 'March'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

        THEN 'April'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

        THEN 'May'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

        THEN 'June'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

        THEN 'July'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

        THEN 'Aug'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

        THEN 'Sep'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

        THEN 'Oct'

        WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

        THEN 'Nov'

        ELSE 'Dec'

     END)  month,

     EXTRACT(YEAR FROM o.order_purchase_timestamp) as year,

    oi.price
```

```
from `TargetEcommerce.orders` o left join `TargetEcommerce.order_items` oi

on o.order_id=oi.order_id),


total_purchases AS (select month,sum(price) as total_sales from price_month
where year=2018 group by month)

select * from total_purchases order by total_sales desc;
```



JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

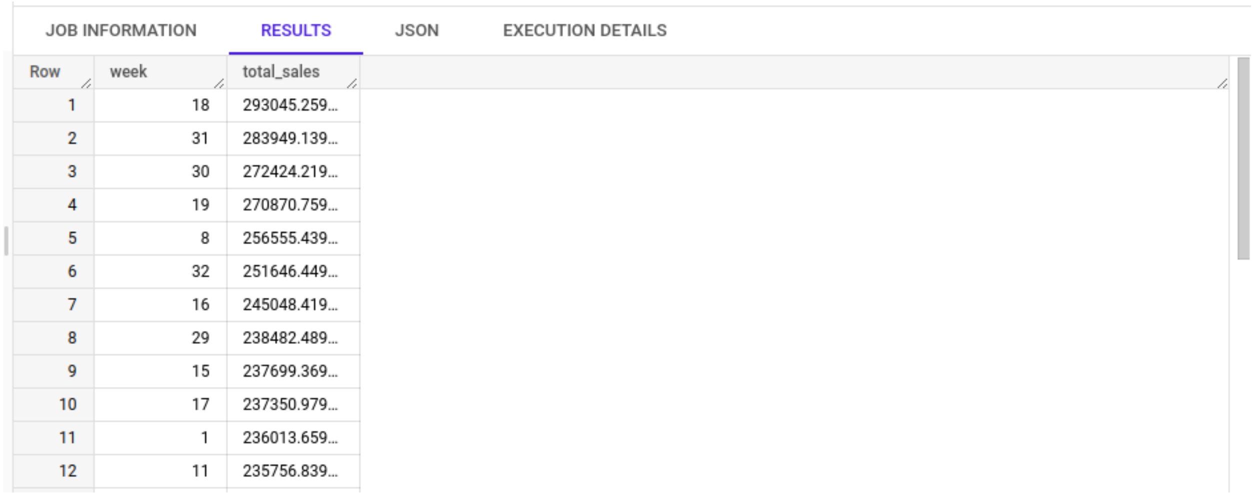
Row	month	total_sales
1	April	996647.750...
2	May	996517.680...
3	March	983213.440...
4	Jan	950030.360...
5	July	895507.220...
6	June	865124.310...
7	Aug	854686.330...
8	Feb	844178.710...
9	Sep	145.0
10	Oct	null

It's strange that there is no order in October 2018. September has the lowest total purchases.For other months the purchases are like random walks.

Week sales:2018

```
with price_week AS (select

    EXTRACT(WEEK FROM o.order_purchase_timestamp) as week,

     EXTRACT(YEAR FROM o.order_purchase_timestamp) as year,

    oi.price

from `TargetEcommerce.orders` o right join `TargetEcommerce.order_items` oi

on o.order_id=oi.order_id),


total_purchases AS (select week,sum(price) as total_sales from price_week where
year=2018 group by week)
```

```
select * from total_purchases order by total_sales desc;
```

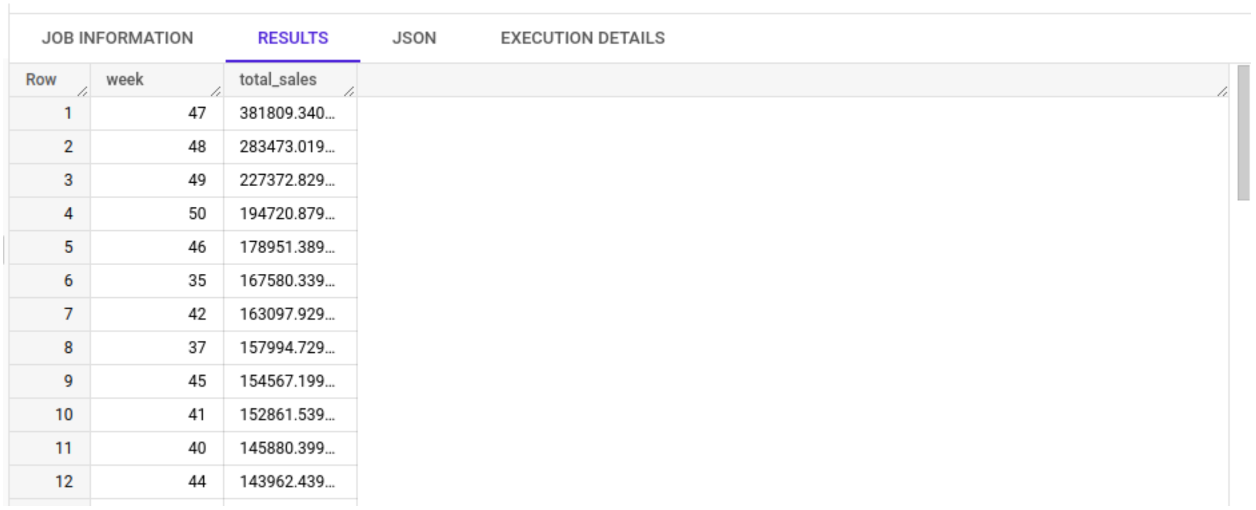


Row	week	total_sales
1	18	293045.259...
2	31	283949.139...
3	30	272424.219...
4	19	270870.759...
5	8	256555.439...
6	32	251646.449...
7	16	245048.419...
8	29	238482.489...
9	15	237699.369...
10	17	237350.979...
11	1	236013.659...
12	11	235756.839...

Highest week sale is in week 18 and lowest week sale is in week 36.

Week sales:2017

```
with price_week AS (select

    EXTRACT(WEEK FROM o.order_purchase_timestamp) as week,

     EXTRACT(YEAR FROM o.order_purchase_timestamp) as year,

    oi.price

from `TargetEcommerce.orders` o right join `TargetEcommerce.order_items` oi

on o.order_id=oi.order_id),



total_purchases AS (select week,sum(price) as total_sales from price_week where
year=2017 group by week)

select * from total_purchases order by total_sales desc;
```


JOB INFORMATION RESULTS JSON EXECUTION DETAILS

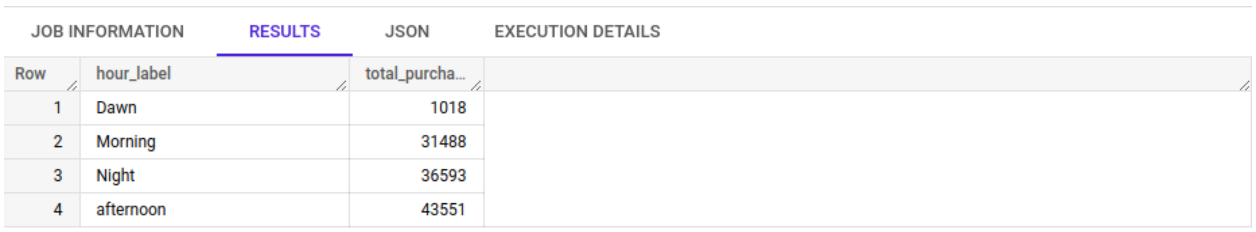
Row	week	total_sales
1	47	381809.340...
2	48	283473.019...
3	49	227372.829...
4	50	194720.879...
5	46	178951.389...
6	35	167580.339...
7	42	163097.929...
8	37	157994.729...
9	45	154567.199...
10	41	152861.539...
11	40	145880.399...
12	44	143962.439...

Highest week is in week 47 and lowest week sale is in week 3.

The year 2017 has a trend of purchases based on weeks.

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

```
with hour_count AS (select oi.product_id,EXTRACT(HOUR FROM o.order_purchase_timestamp)
as hour from `TargetEcommerce.orders` o join `TargetEcommerce.order_items` oi on
o.order_id=oi.order_id),


hour_named AS (select

product_id,

case

  when hour between 4 and 6

  THEN 'Dawn'

  when hour>6 and hour<=12

  then 'Morning'

  when hour>12 and hour<=18

  then 'afternoon'
  else 'Night'

END as hour_label

from hour_count),


total_count as (select hour_label,count(product_id) as total_purchase from hour_named
group by hour_label)


select * from total_count order by total_purchase;
```



JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	hour_label	total_purcha...
1	Dawn	1018
2	Morning	31488
3	Night	36593
4	afternoon	43551

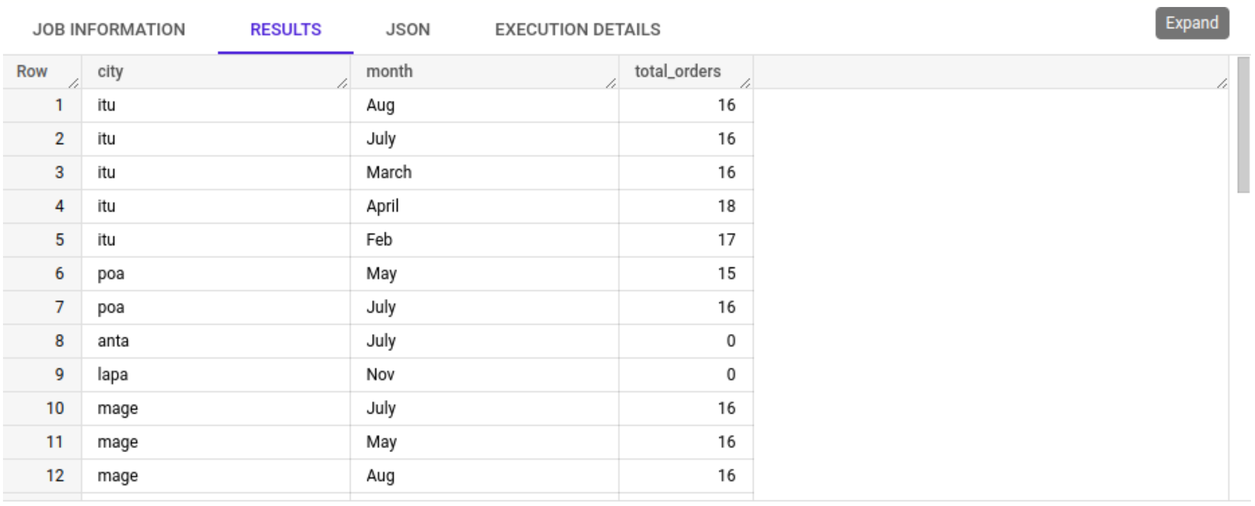
Brazilians prefer Afternoon as the ideal time to make a purchase ,Dawn is the least, followed by morning.

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

a. month on month orders by region, states:

```
-- to get the orders month by month for every city and states

-- we need to join tables customers,orders,and order_items

with city_orders AS (select

   c.customer_city,

   (CASE

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=1
```

```sql
    THEN 'Jan'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

    THEN 'Feb'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

    THEN 'March'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

    THEN 'April'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

    THEN 'May'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

    THEN 'June'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

    THEN 'July'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

    THEN 'Aug'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

    THEN 'Sep'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

    THEN 'Oct'

    WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

    THEN 'Nov'

    ELSE 'Dec'

END)  month,
    oi.order_item_id


from `TargetEcommerce.customers` c  left join `TargetEcommerce.orders` o

on o.customer_id=c.customer_id left join `TargetEcommerce.order_items` oi

on oi.order_id=o.order_id ),

total_orders_city AS (select customer_city as city,month,count(order_item_id) as
total_orders  from city_orders group by customer_city,month )

select * from total_orders_city;
```

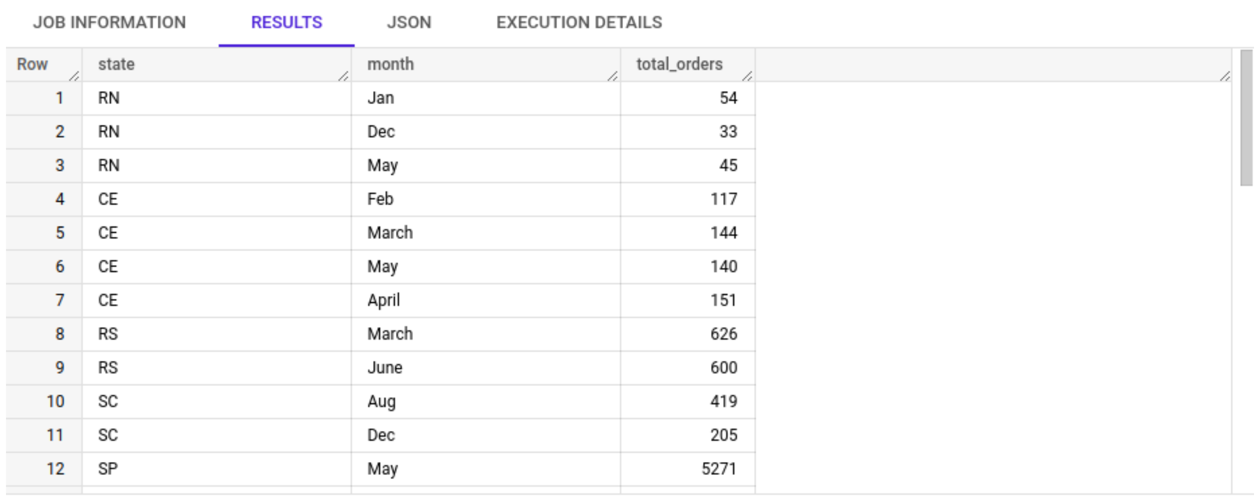


Row	city	month	total_orders
1	itu	Aug	16
2	itu	July	16
3	itu	March	16
4	itu	April	18
5	itu	Feb	17
6	poa	May	15
7	poa	July	16
8	anta	July	0
9	lapa	Nov	0
10	mage	July	16
11	mage	May	16
12	mage	Aug	16

City wise month on month total number of orders placed by customers.

```
-- to get the orders month by month for every city and states

-- we need to join tables customers,orders,and order_items

with state_orders AS (select

   c.customer_state,

   (CASE

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=1

       THEN 'Jan'
       WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

       THEN 'Feb'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

       THEN 'March'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

       THEN 'April'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

       THEN 'May'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

       THEN 'June'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

       THEN 'July'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

       THEN 'Aug'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

       THEN 'Sep'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

       THEN 'Oct'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

       THEN 'Nov'

       ELSE 'Dec'

    END)  month,

    oi.order_item_id
```

```
from `TargetEcommerce.customers` c  left join `TargetEcommerce.orders` o

on o.customer_id=c.customer_id left join `TargetEcommerce.order_items` oi

on oi.order_id=o.order_id ),

total_orders_state AS (select customer_state as state,month,count(order_item_id) as
total_orders  from state_orders group by customer_state,month )

select * from total_orders_state;
```



JOB INFORMATION RESULTS JSON EXECUTION DETAILS

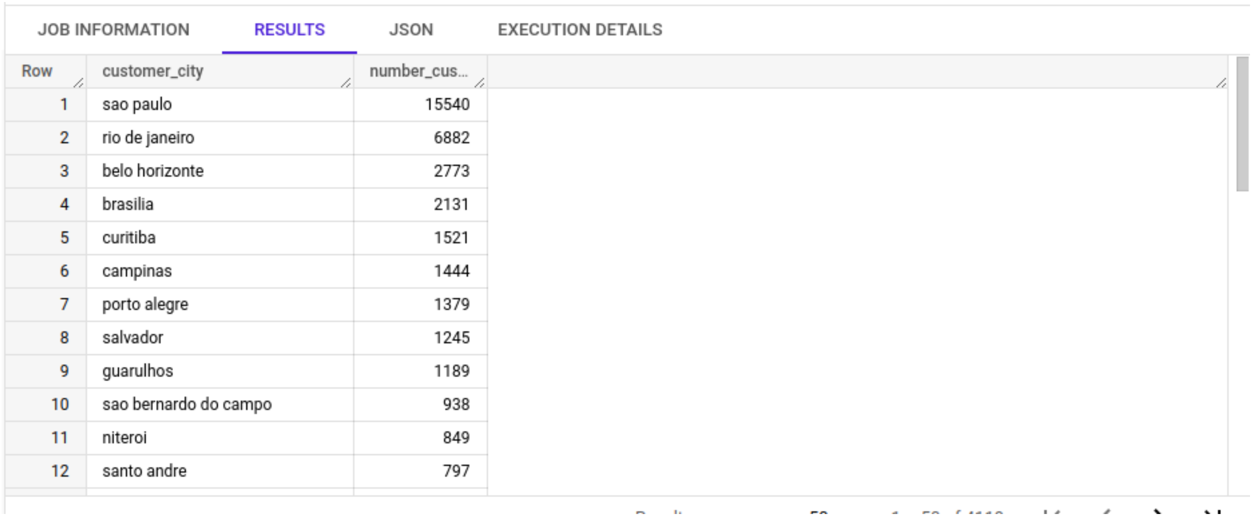
Row	state	month	total_orders
1	RN	Jan	54
2	RN	Dec	33
3	RN	May	45
4	CE	Feb	117
5	CE	March	144
6	CE	May	140
7	CE	April	151
8	RS	March	626
9	RS	June	600
10	SC	Aug	419
11	SC	Dec	205
12	SP	May	5271

State wise month on month total numbers placed by customers.

b.How are customers distributed in Brazil

City-wise:

```
select customer_city,count(*) as number_customers from
`TargetEcommerce.customers`group by customer_city order by count(*) desc;
```

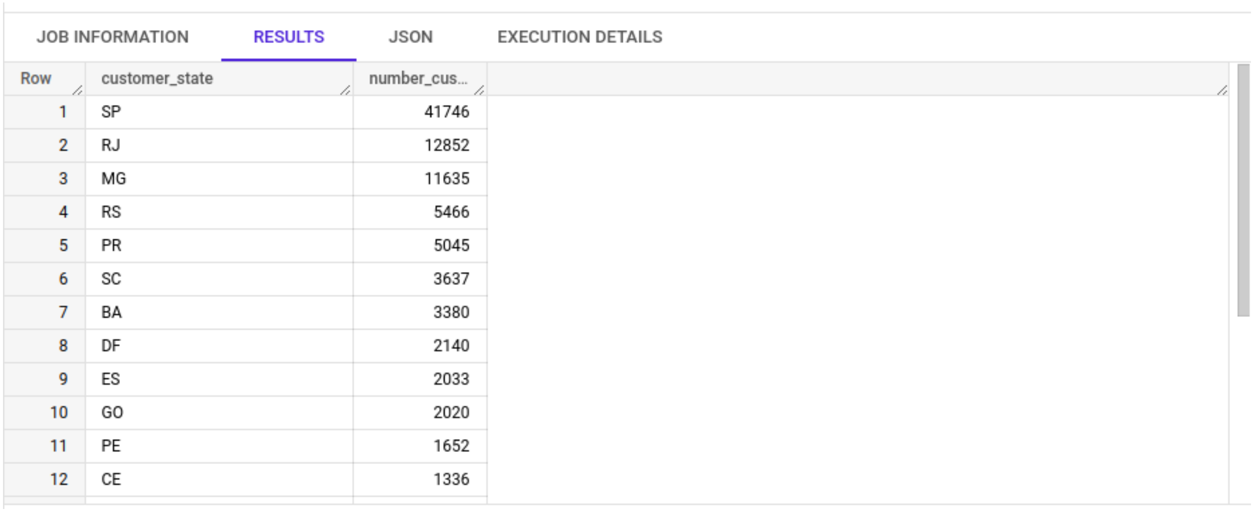

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_city	number_cus...
1	sao paulo	15540
2	rio de janeiro	6882
3	belo horizonte	2773
4	brasilia	2131
5	curitiba	1521
6	campinas	1444
7	porto alegre	1379
8	salvador	1245
9	guarulhos	1189
10	sao bernardo do campo	938
11	niteroi	849
12	santo andre	797

City-wise number of customers.Sao Paulo has the highest number of customers.

State-wise:

```
select customer_state,count(*) as number_customers from
`TargetEcommerce.customers`group by customer_state order by count(*) desc;
```


JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	customer_state	number_cus...
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

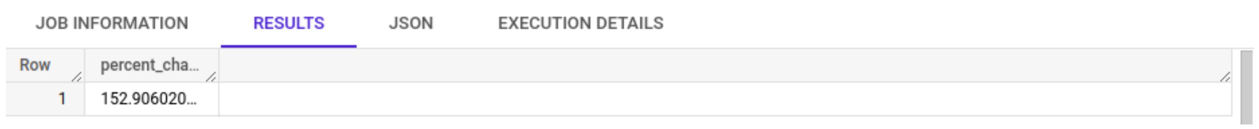
State of São Paulo is the state with the highest number of customers.

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

a.Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only)

Percentage change from 2017 to 2018 based on the freight values from month Jan to Aug

```
with price_percentage_change AS (select

  (CASE

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=1

      THEN 'Jan'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

      THEN 'Feb'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

      THEN 'March'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

      THEN 'April'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

      THEN 'May'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

      THEN 'June'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

      THEN 'July'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

      THEN 'Aug'
      WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

      THEN 'Sep'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

      THEN 'Oct'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

      THEN 'Nov'

      ELSE 'Dec'

   END)  month,

   oi.freight_value,

   oi.price,

   EXTRACT(YEAR FROM o.order_purchase_timestamp) as year

   from `TargetEcommerce.orders` o left join `TargetEcommerce.order_items` oi

   on o.order_id=oi.order_id),

   freight as (select year,sum(freight_value) as total_freight_value from
price_percentage_change where month in
('Jan','Feb','March','April','May','June','July','Aug') and year in (2017,2018) group
by year)


  select
100*(curr.total_freight_value-prev.total_freight_value)/prev.total_freight_value as
percent_change

  from freight as curr

  join freight as prev

  on curr.year=2018 and prev.year=2017;
```

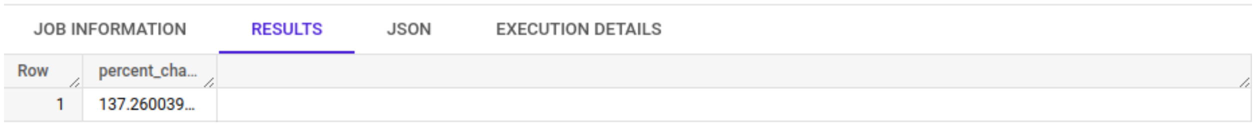

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	percent_cha...
1	152.906020...

There is 152% increase of freight value from 2017 to 2018

Percentage change from 2017 to 2018 based on the price from month Jan to Aug

```
with price_percentage_change AS (select

  (CASE

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=1

      THEN 'Jan'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

      THEN 'Feb'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

      THEN 'March'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

      THEN 'April'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

      THEN 'May'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

      THEN 'June'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

      THEN 'July'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

      THEN 'Aug'
      WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

      THEN 'Sep'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

      THEN 'Oct'

      WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

      THEN 'Nov'

      ELSE 'Dec'

   END)  month,

   oi.freight_value,

   oi.price,

   EXTRACT(YEAR FROM o.order_purchase_timestamp) as year

   from `TargetEcommerce.orders` o left join `TargetEcommerce.order_items` oi

   on o.order_id=oi.order_id),

   prices as (select year,sum(price) as total_price_value from price_percentage_change
where month in ('Jan','Feb','March','April','May','June','July','Aug') and year in
(2017,2018) group by year)


  select 100*(curr.total_price_value-prev.total_price_value)/prev.total_price_value as
percent_change

  from prices as curr

  join prices as prev

  on curr.year=2018 and prev.year=2017;
```

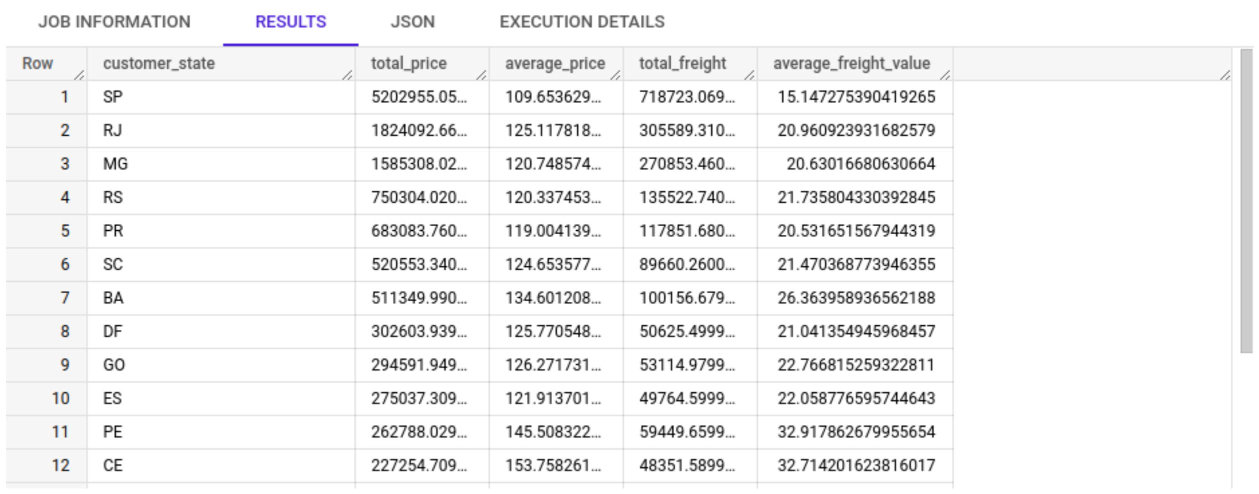

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	percent_cha...
1	137.260039...

There is a 137% increase in price of the products from 2017 to 2018 based on the months from Jan to Aug .

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

```
with state_order as (select c.customer_state,

oi.price,

oi.freight_value

from `TargetEcommerce.customers` c left join `TargetEcommerce.orders`o

on c.customer_id=o.customer_id

left join `TargetEcommerce.order_items` oi

on oi.order_id=o.order_id)



select customer_state,sum(price) as total_price,avg(price) as
average_price,sum(freight_value) as total_freight,

avg(freight_value) as average_freight_value from state_order group by customer_state
order by sum(price) desc,sum(freight_value) desc;
```



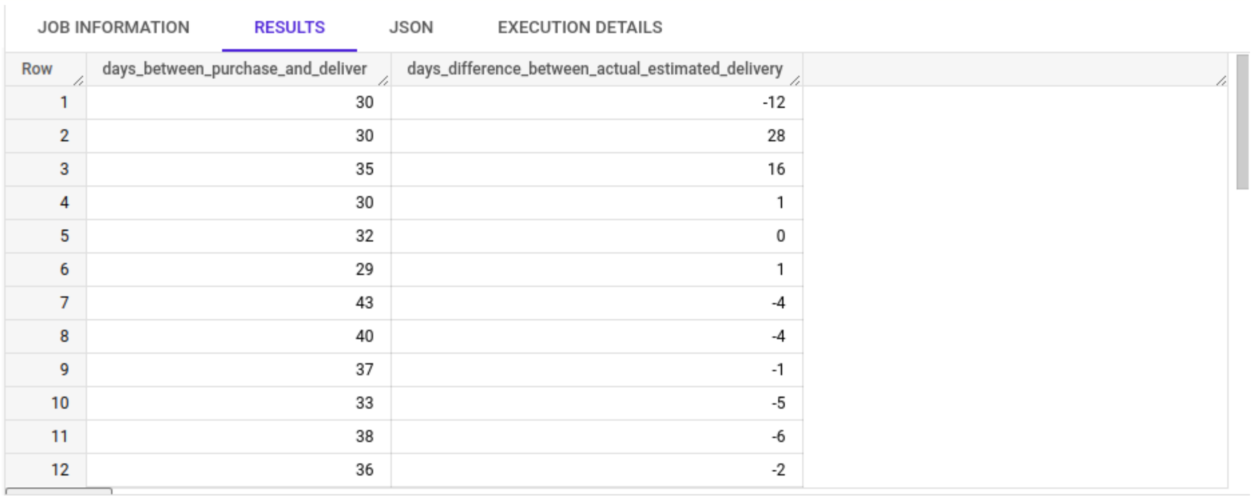
Row	customer_state	total_price	average_price	total_freight	average_freight_value
1	SP	5202955.05...	109.653629...	718723.069...	15.147275390419265
2	RJ	1824092.66...	125.117818...	305589.310...	20.960923931682579
3	MG	1585308.02...	120.748574...	270853.460...	20.63016680630664
4	RS	750304.020...	120.337453...	135522.740...	21.735804330392845
5	PR	683083.760...	119.004139...	117851.680...	20.531651567944319
6	SC	520553.340...	124.653577...	89660.2600...	21.470368773946355
7	BA	511349.990...	134.601208...	100156.679...	26.363958936562188
8	DF	302603.939...	125.770548...	50625.4999...	21.041354945968457
9	GO	294591.949...	126.271731...	53114.9799...	22.766815259322811
10	ES	275037.309...	121.913701...	49764.5999...	22.058776595744643
11	PE	262788.029...	145.508322...	59449.6599...	32.917862679955654
12	CE	227254.709...	153.758261...	48351.5899...	32.714201623816017

The state of Sao Paulo has the highest total price whereas the state of Roraima has the lowest total price.

5.Analysis on sales, freight and delivery time

a & b.Days between purchasing, delivering and estimated delivery.

```
-- days between purchasing, delivering and estimated delivery

with delivery_days AS (select

-- delivery time between purchase and delivery date

    DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY) as
time_to_deliver,

-- difference between actual delivery date and estimated delivery date

  DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,DAY) as
diff_estimated_delivery



from `TargetEcommerce.orders`)

select time_to_deliver AS days_between_purchase_and_deliver,

 diff_estimated_delivery AS days_difference_between_actual_estimated_delivery
 from delivery_days;
```



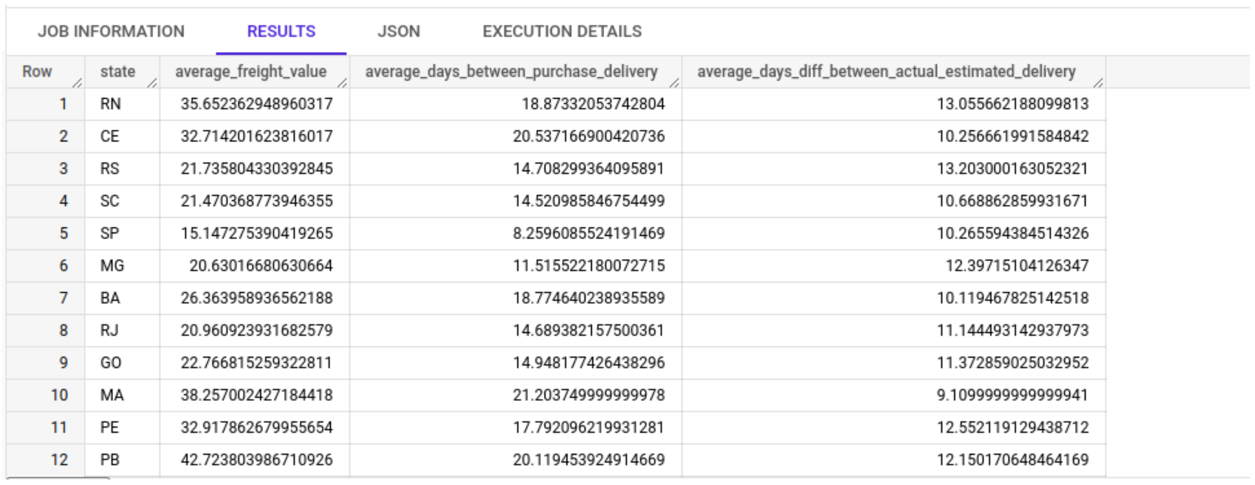
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	days_between_purchase_and_deliver	days_difference_between_actual_estimated_delivery
1	30	-12
2	30	28
3	35	16
4	30	1
5	32	0
6	29	1
7	43	-4
8	40	-4
9	37	-1
10	33	-5
11	38	-6
12	36	-2

There are some values in the third column that are negative which indicates that the actual delivery took x number of more days than the estimated delivery time.

c. Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

```
with price_days_freight AS (select

 c.customer_state,

 DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY) as
time_to_deliver,

  DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) as
diff_estimated_delivery,

  oi.freight_value

 from `TargetEcommerce.customers` c left join `TargetEcommerce.orders` o

 on c.customer_id=o.customer_id left join `TargetEcommerce.order_items` oi

 on oi.order_id=o.order_id

)

select customer_state as state,avg(freight_value) as
average_freight_value,avg(time_to_deliver) as average_days_between_purchase_delivery,
avg(diff_estimated_delivery) as average_days_diff_between_actual_estimated_delivery

from price_days_freight group by customer_state;
```



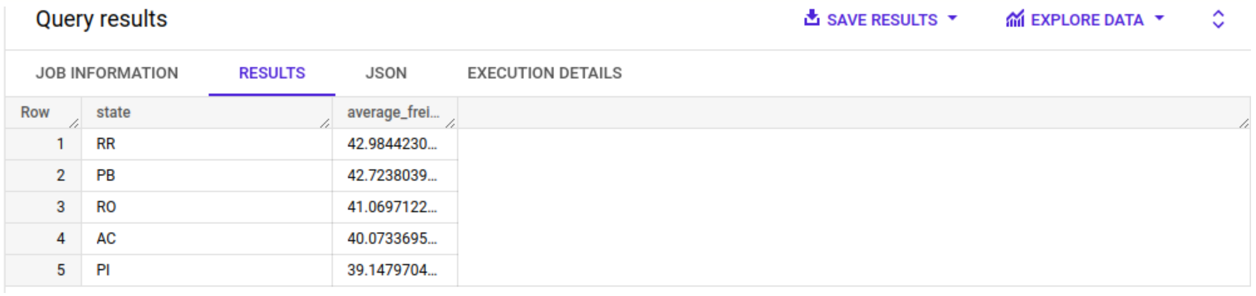
JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	state	average_freight_value	average_days_between_purchase_delivery	average_days_diff_between_actual_estimated_delivery
1	RN	35.652362948960317	18.87332053742804	13.055662188099813
2	CE	32.714201623816017	20.537166900420736	10.256661991584842
3	RS	21.735804330392845	14.708299364095891	13.203000163052321
4	SC	21.470368773946355	14.520985846754499	10.668862859931671
5	SP	15.147275390419265	8.2596085524191469	10.265594384514326
6	MG	20.63016680630664	11.515522180072715	12.39715104126347
7	BA	26.363958936562188	18.774640238935589	10.119467825142518
8	RJ	20.960923931682579	14.689382157500361	11.144493142937973
9	GO	22.766815259322811	14.948177426438296	11.372859025032952
10	MA	38.257002427184418	21.203749999999978	9.1099999999999941
11	PE	32.917862679955654	17.792096219931281	12.552119129438712
12	PB	42.723803986710926	20.119453924914669	12.150170648464169

```
The state of RN took on average 18 days to delivery a product to the customer
```

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

```
with price_days_freight AS (select

c.customer_state,

DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY) as
time_to_deliver,

 DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) as
diff_estimated_delivery,

 oi.freight_value

from `TargetEcommerce.customers` c left join `TargetEcommerce.orders` o

on c.customer_id=o.customer_id left join `TargetEcommerce.order_items` oi

on oi.order_id=o.order_id

),

states_with_avg_freight as (select customer_state as state,avg(freight_value) as
average_freight_value,avg(time_to_deliver) as average_days_between_purchase_delivery,
avg(diff_estimated_delivery) as average_days_diff_between_actual_estimated_delivery

from price_days_freight group by customer_state)

-- top 5 states with heighest average freight value

select state,average_freight_value from states_with_avg_freight order by
average_freight_value desc limit 5;
```



Query results

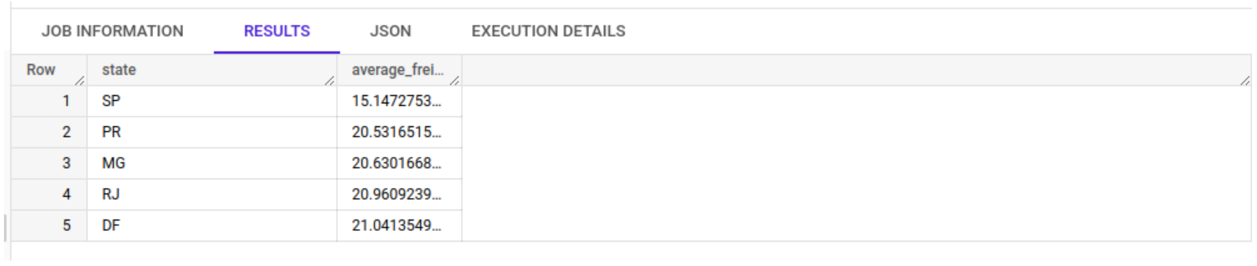
SAVE RESULTS | EXPLORE DATA

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	state	average_frei...
1	RR	42.9844230...
2	PB	42.7238039...
3	RO	41.0697122...
4	AC	40.0733695...
5	PI	39.1479704...

Top 5 states with highest average freight value

```
with price_days_freight AS (select

c.customer_state,

DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY) as
time_to_deliver,

 DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) as
diff_estimated_delivery,

 oi.freight_value

from `TargetEcommerce.customers` c left join `TargetEcommerce.orders` o

on c.customer_id=o.customer_id left join `TargetEcommerce.order_items` oi

on oi.order_id=o.order_id

),
states_with_avg_freight as (select customer_state as state,avg(freight_value) as
average_freight_value,avg(time_to_deliver) as average_days_between_purchase_delivery,

avg(diff_estimated_delivery) as average_days_diff_between_actual_estimated_delivery

from price_days_freight group by customer_state)

-- top 5 states with heighest average freight value

select state,average_freight_value from states_with_avg_freight order by
average_freight_value asc limit 5;
```

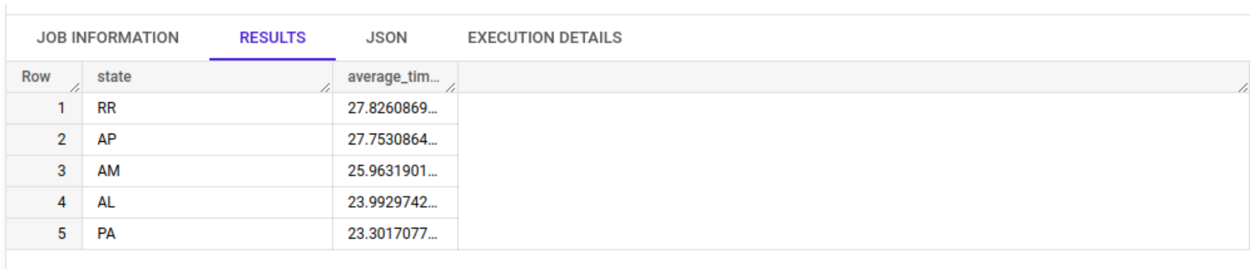


Row	state	average_frei...
1	SP	15.1472753...
2	PR	20.5316515...
3	MG	20.6301668...
4	RJ	20.9609239...
5	DF	21.0413549...

Top 5 states with lowest average freight value

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

```
with price_days_freight AS (select

c.customer_state,

DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY) as
time_to_deliver,

 DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) as
diff_estimated_delivery,

 oi.freight_value

from `TargetEcommerce.customers` c left join `TargetEcommerce.orders` o

on c.customer_id=o.customer_id left join `TargetEcommerce.order_items` oi

on oi.order_id=o.order_id
),

states_with_avg_freight as (select customer_state as state,avg(freight_value) as
average_freight_value,avg(time_to_deliver) as average_days_between_purchase_delivery,

avg(diff_estimated_delivery) as average_days_diff_between_actual_estimated_delivery

from price_days_freight group by customer_state)

-- top 5 states with heighest average time to delivery

select state,average_days_between_purchase_delivery as average_time_to_delivery from
states_with_avg_freight order by average_days_between_purchase_delivery  desc limit 5;
```

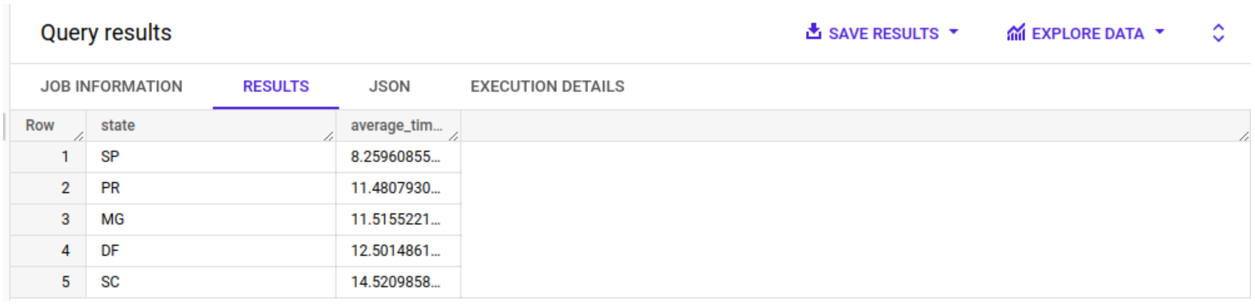


JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	state	average_tim...
1	RR	27.8260869...
2	AP	27.7530864...
3	AM	25.9631901...
4	AL	23.9929742...
5	PA	23.3017077...

Top 5 states with highest average time to delivery.

```
with price_days_freight AS (select

c.customer_state,

DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY) as
time_to_deliver,

 DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) as
diff_estimated_delivery,

 oi.freight_value

from `TargetEcommerce.customers` c left join `TargetEcommerce.orders` o

on c.customer_id=o.customer_id left join `TargetEcommerce.order_items` oi

on oi.order_id=o.order_id

),
states_with_avg_freight as (select customer_state as state,avg(freight_value) as
average_freight_value,avg(time_to_deliver) as average_days_between_purchase_delivery,

avg(diff_estimated_delivery) as average_days_diff_between_actual_estimated_delivery

from price_days_freight group by customer_state)

-- top 5 states with heighest average time to delivery

select state,average_days_between_purchase_delivery as average_time_to_delivery from
states_with_avg_freight order by average_days_between_purchase_delivery  asc limit 5;
```

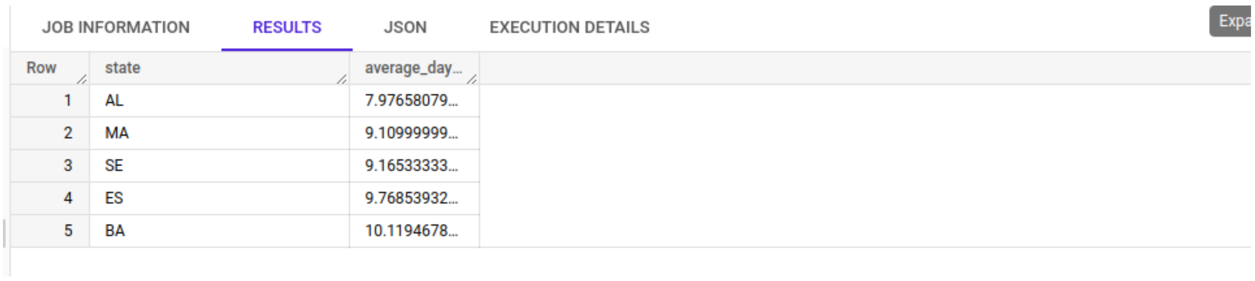


Row	state	average_tim...
1	SP	8.25960855...
2	PR	11.4807930...
3	MG	11.5155221...
4	DF	12.5014861...
5	SC	14.5209858...

Top 5 states with lowest average time to delivery.

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

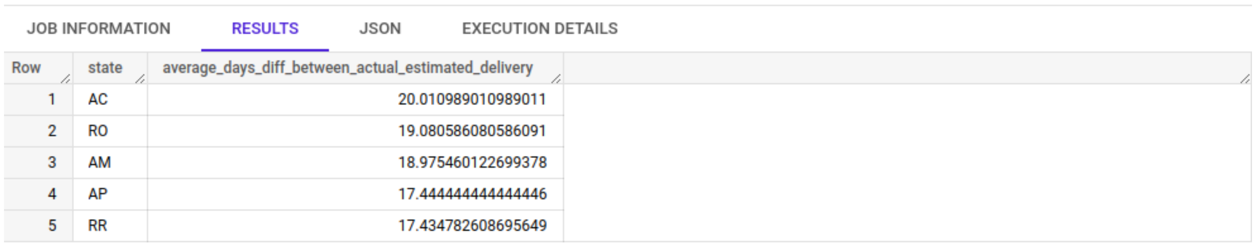
```
with price_days_freight AS (select

 c.customer_state,

 DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY) as
time_to_deliver,

  DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) as
diff_estimated_delivery,

  oi.freight_value

 from `TargetEcommerce.customers` c left join `TargetEcommerce.orders` o

 on c.customer_id=o.customer_id left join `TargetEcommerce.order_items` oi
 on oi.order_id=o.order_id

),

states_with_avg_freight as (select customer_state as state,avg(freight_value) as
average_freight_value,avg(time_to_deliver) as average_days_between_purchase_delivery,

avg(diff_estimated_delivery) as average_days_diff_between_actual_estimated_delivery

from price_days_freight group by customer_state)


-- top 5 states where delivery is really fast compared to estimated date

-- to get this we need to sort the
"average_days_diff_between_actual_estimated_delivery" in ascending order

select state,average_days_diff_between_actual_estimated_delivery from
states_with_avg_freight order by average_days_diff_between_actual_estimated_delivery
asc limit 5;
```



Row	state	average_day...
1	AL	7.97658079...
2	MA	9.10999999...
3	SE	9.16533333...
4	ES	9.76853932...
5	BA	10.1194678...

Top 5 states where delivery really fast compared to estimated date.For example the state AL is the state with fastest delivery.

```
with price_days_freight AS (select

 c.customer_state,

 DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,DAY) as
time_to_deliver,

  DATE_DIFF(o.order_estimated_delivery_date,o.order_delivered_customer_date,DAY) as
diff_estimated_delivery,
  oi.freight_value

 from `TargetEcommerce.customers` c left join `TargetEcommerce.orders` o

 on c.customer_id=o.customer_id left join `TargetEcommerce.order_items` oi

 on oi.order_id=o.order_id

),

states_with_avg_freight as (select customer_state as state,avg(freight_value) as
average_freight_value,avg(time_to_deliver) as average_days_between_purchase_delivery,

avg(diff_estimated_delivery) as average_days_diff_between_actual_estimated_delivery

from price_days_freight group by customer_state)



-- top 5 states where delivery is really slow compared to estimated date

-- to get this we need to sort the
"average_days_diff_between_actual_estimated_delivery" in ascending order

select state,average_days_diff_between_actual_estimated_delivery from
states_with_avg_freight order by average_days_diff_between_actual_estimated_delivery
desc limit 5;
```



Row	state	average_days_diff_between_actual_estimated_delivery
1	AC	20.010989010989011
2	RO	19.080586080586091
3	AM	18.975460122699378
4	AP	17.444444444444446
5	RR	17.434782608695649

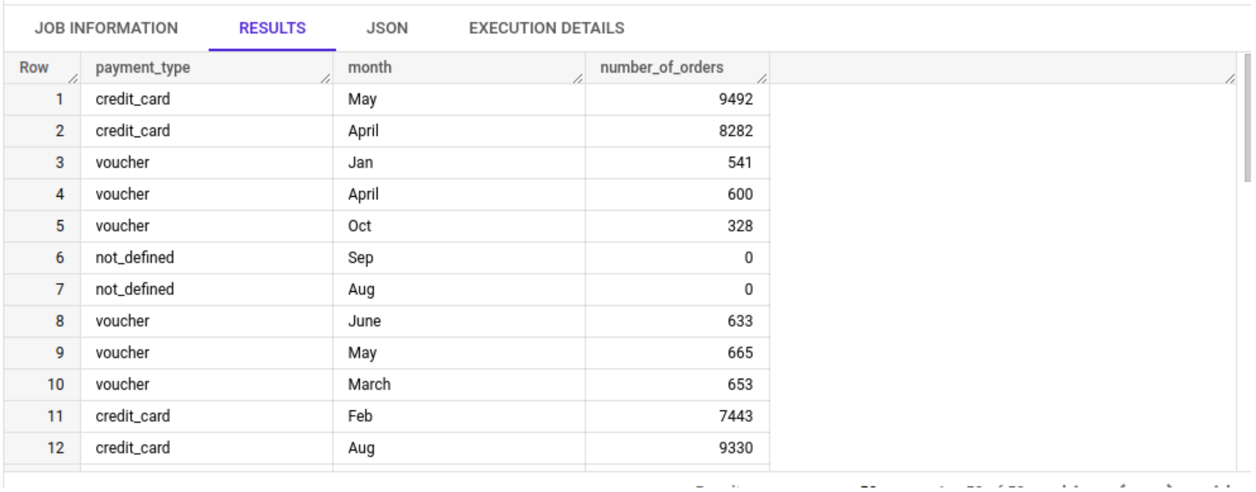
Top 5 states where delivery is really slow compared to estimated date.For example the state AC is the slowest state in terms of delivery.

6.Payment type analysis:

a.Month over Month count of orders for different payment types

```sql
with month_orders AS (select

   (CASE

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=1

       THEN 'Jan'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

       THEN 'Feb'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

       THEN 'March'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

       THEN 'April'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

       THEN 'May'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

       THEN 'June'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

       THEN 'July'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

       THEN 'Aug'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

       THEN 'Sep'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

       THEN 'Oct'

       WHEN extract(MONTH FROM o.order_purchase_timestamp)=11
```

```
        THEN 'Nov'

        ELSE 'Dec'

     END)  month,

   p.payment_type,

   oi.order_item_id

   from `TargetEcommerce.orders` o join `TargetEcommerce.payments` p

   on o.order_id=p.order_id left join `TargetEcommerce.order_items` oi

   on o.order_id=oi.order_id)


   select payment_type,month,count(order_item_id) as number_of_orders from
month_orders group by payment_type,month;
```



Row	payment_type	month	number_of_orders
1	credit_card	May	9492
2	credit_card	April	8282
3	voucher	Jan	541
4	voucher	April	600
5	voucher	Oct	328
6	not_defined	Sep	0
7	not_defined	Aug	0
8	voucher	June	633
9	voucher	May	665
10	voucher	March	653
11	credit_card	Feb	7443
12	credit_card	Aug	9330

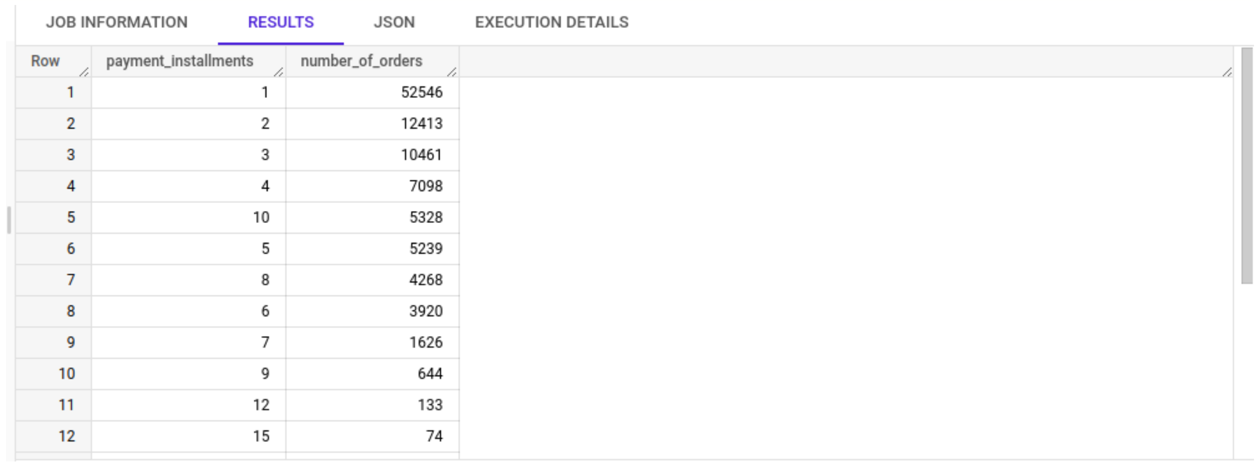
Month over month number of orders for each different payment type.

b.Distribution of payment installments and count of orders

```
with month_orders AS (select

   (CASE
```

```sql
WHEN extract(MONTH FROM o.order_purchase_timestamp)=1

THEN 'Jan'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=2

THEN 'Feb'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=3

THEN 'March'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=4

THEN 'April'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=5

THEN 'May'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=6

THEN 'June'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=7

THEN 'July'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=8

THEN 'Aug'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=9

THEN 'Sep'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=10

THEN 'Oct'

WHEN extract(MONTH FROM o.order_purchase_timestamp)=11

THEN 'Nov'

ELSE 'Dec'
```

```
 END)  month,

p.payment_installments

from `TargetEcommerce.orders` o join `TargetEcommerce.payments` p

on o.order_id=p.order_id)



select payment_installments,count(*) as number_of_orders from month_orders group by
payment_installments order by count(*) desc;
```



JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	payment_installments	number_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133
12	15	74

1 installment has highest number of orders