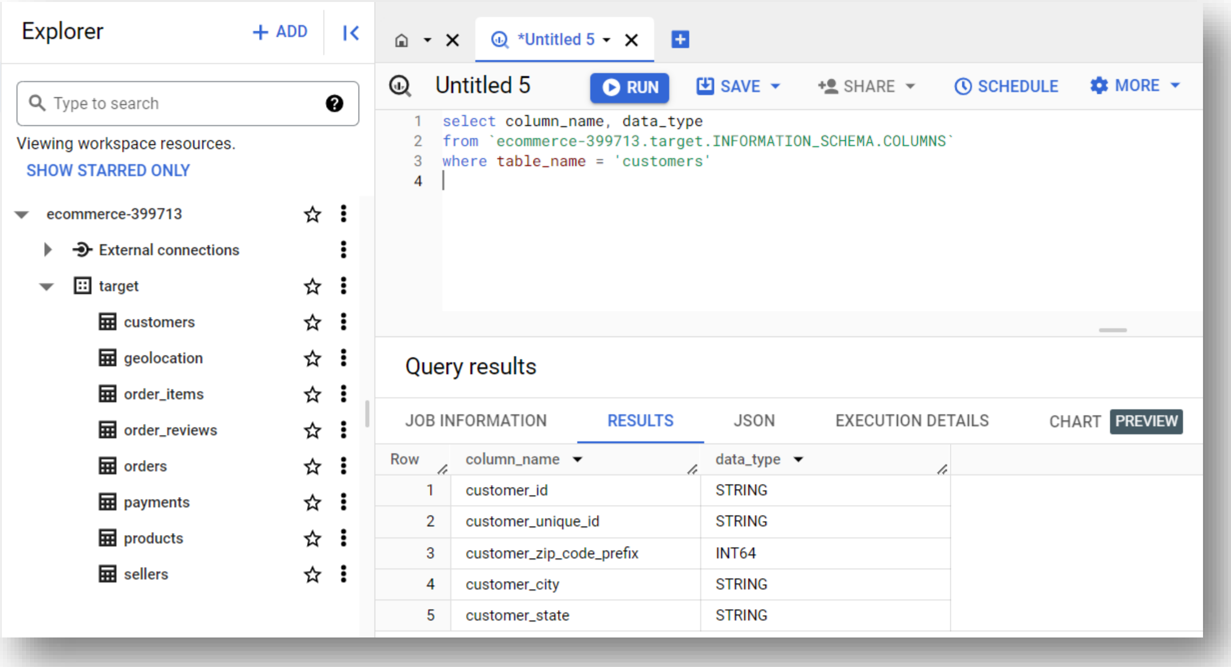
Business case: Target SQL

1.1 Data type of all columns in the “customers” table

Query:-

```
select column_name, data_type
from `ecommerce-399713.target.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
```

Output Screenshot:-



```
select column_name, data_type
from `ecommerce-399713.target.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
```

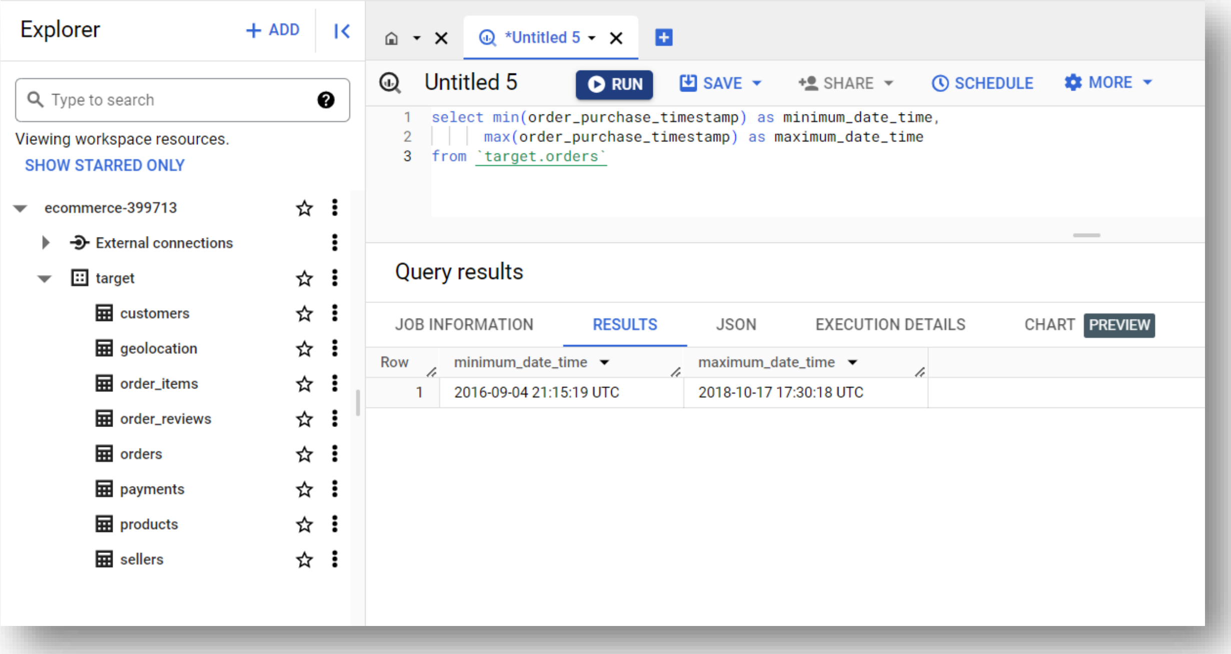
Row	column_name	data_type
1	customer_id	STRING
2	customer_unique_id	STRING
3	customer_zip_code_prefix	INT64
4	customer_city	STRING
5	customer_state	STRING

1.2 Get the time range between which the orders were placed

Query:-

```
select min(order_purchase_timestamp) as minimum_date_time,
max(order_purchase_timestamp) as maximum_date_time
from `target.orders`
```

Output Screenshot:-



```
select min(order_purchase_timestamp) as minimum_date_time,
       max(order_purchase_timestamp) as maximum_date_time
from `target.orders`
```

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

1.3 Count the Cities & States of customers who ordered during the given period

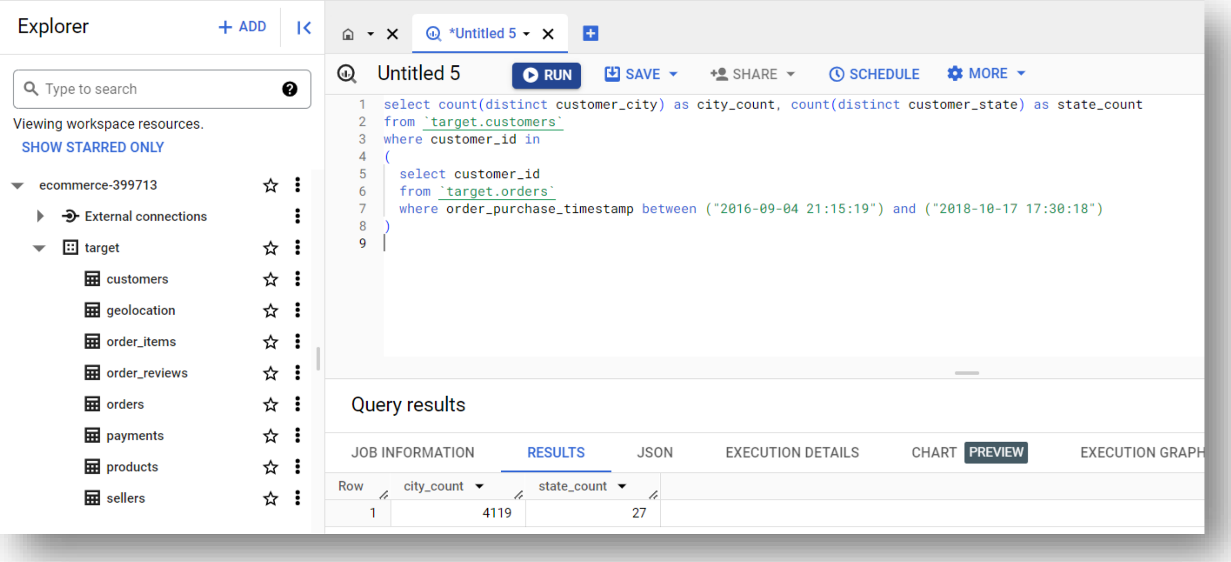
Query:-

```
select count(distinct customer_city) as city_count, count(distinct customer_state) as
state_count

from `target.customers`

where customer_id in
(
  select customer_id
  from `target.orders`
  where order_purchase_timestamp between ("2016-09-04 21:15:19") and ("2018-10-17
17:30:18")
)
```

Output Screenshot:-



```
select count(distinct customer_city) as city_count, count(distinct customer_state) as state_count
from `target.customers`
where customer_id in
(
  select customer_id
  from `target.orders`
  where order_purchase_timestamp between ("2016-09-04 21:15:19") and ("2018-10-17 17:30:18")
)
```

Row	city_count	state_count
1	4119	27

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

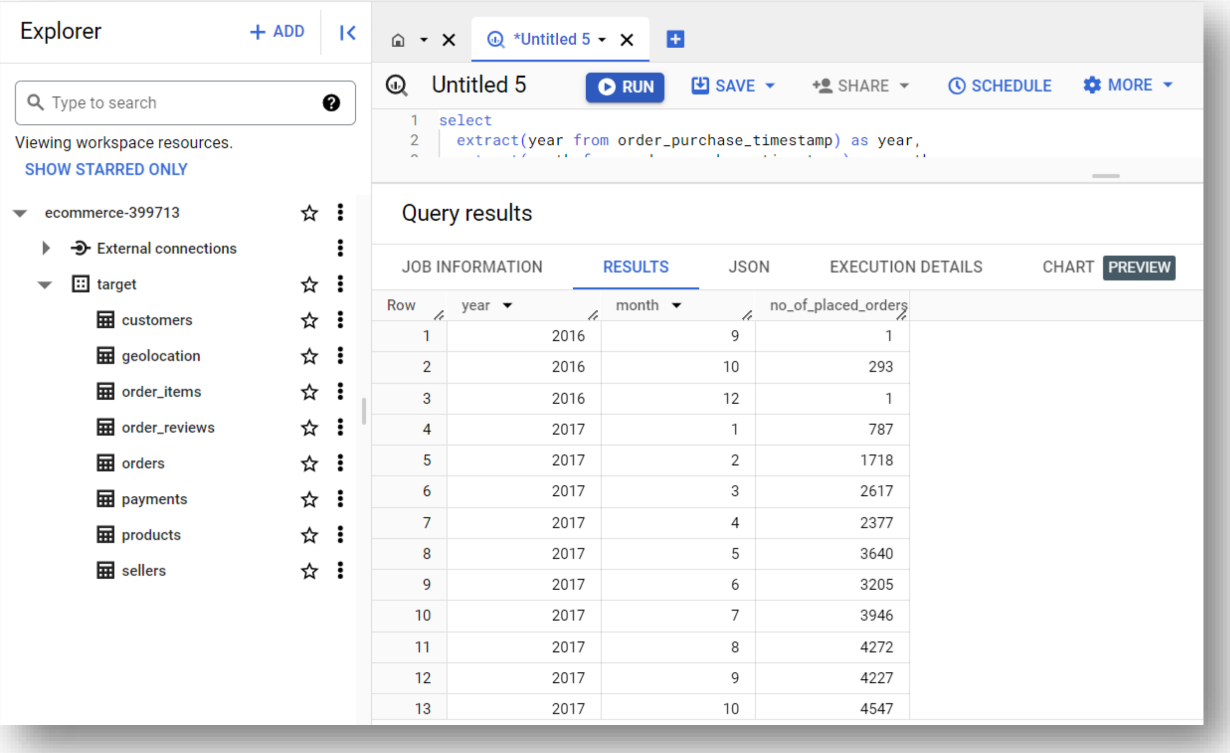
Assumption:-

- Considering, once the payment has been received for any purchase order that order is called as placed order. And not taking order status as cancelled and unavailable for placed orders.

Query:-

```
select
  extract(year from order_purchase_timestamp) as year,
  extract(month from order_purchase_timestamp) as month,
  count(distinct p.order_id) as no_of_placed_orders
from `target.payments` as p
inner join `target.orders` as o
on o.order_id=p.order_id
where order_status not in ("canceled", "unavailable")
group by year, month
order by year, month
```

Output Screenshot:-



```
select
  extract(year from order_purchase_timestamp) as year,
```

Query results

Row	year	month	no_of_placed_orders
1	2016	9	1
2	2016	10	293
3	2016	12	1
4	2017	1	787
5	2017	2	1718
6	2017	3	2617
7	2017	4	2377
8	2017	5	3640
9	2017	6	3205
10	2017	7	3946
11	2017	8	4272
12	2017	9	4227
13	2017	10	4547

Insights:-

Highest no. of placed orders in Nov, 2017

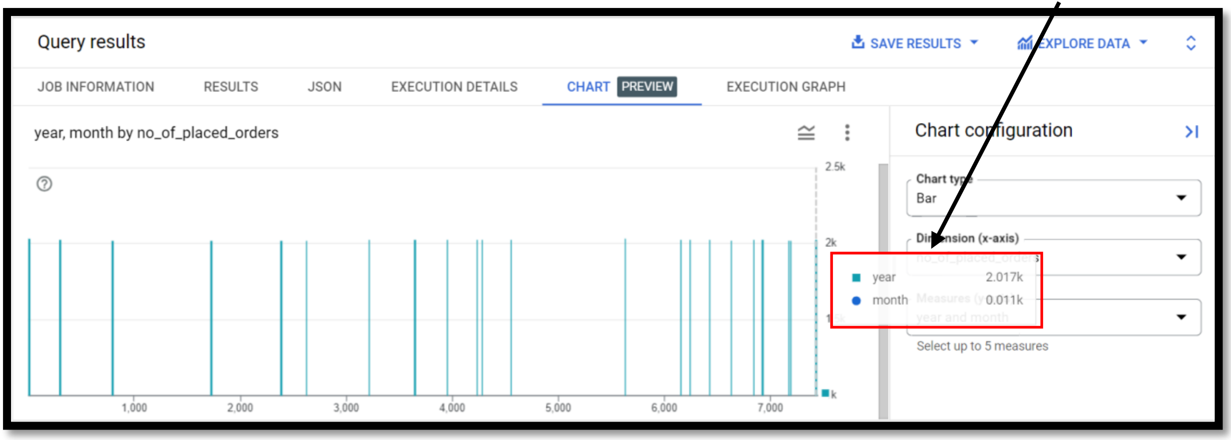


Chart type: Bar

Dimension (x-axis): no_of_placed_orders

Measures (y-axis): year and month

■ year	2.017k
● month	0.011k

There is a growing trend in the number of placed orders over the past years. People are choosing to buy more things online than before as online shopping is convenient from the time saving perspective as well.

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

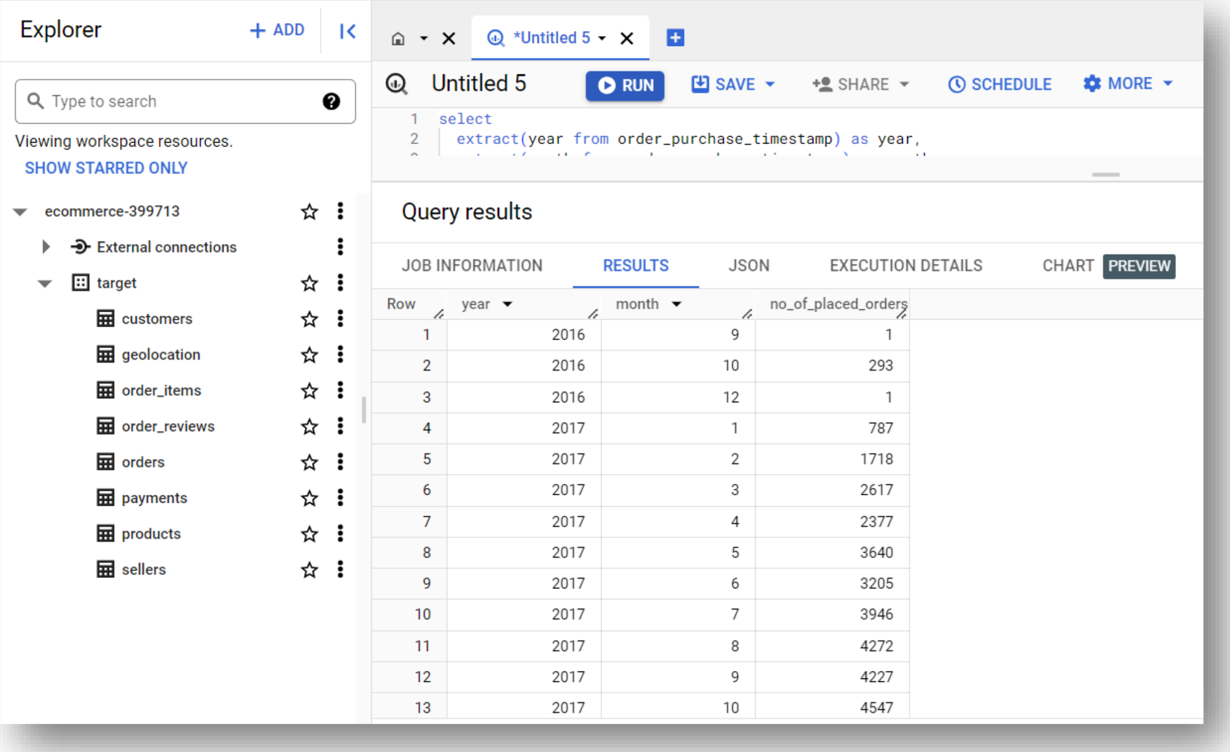
Assumption:-

- Considering, once the payment has been received for any purchase order that order is called as placed order. And not taking order status as cancelled and unavailable for placed orders.

Query:-

```
select
  extract(year from order_purchase_timestamp) as year,
  extract(month from order_purchase_timestamp) as month,
  count(distinct p.order_id) as no_of_placed_orders
from `target.payments` as p
inner join `target.orders` as o
on o.order_id=p.order_id
where order_status not in ("canceled", "unavailable")
group by year, month
order by year, month
```

Output Screenshot:-

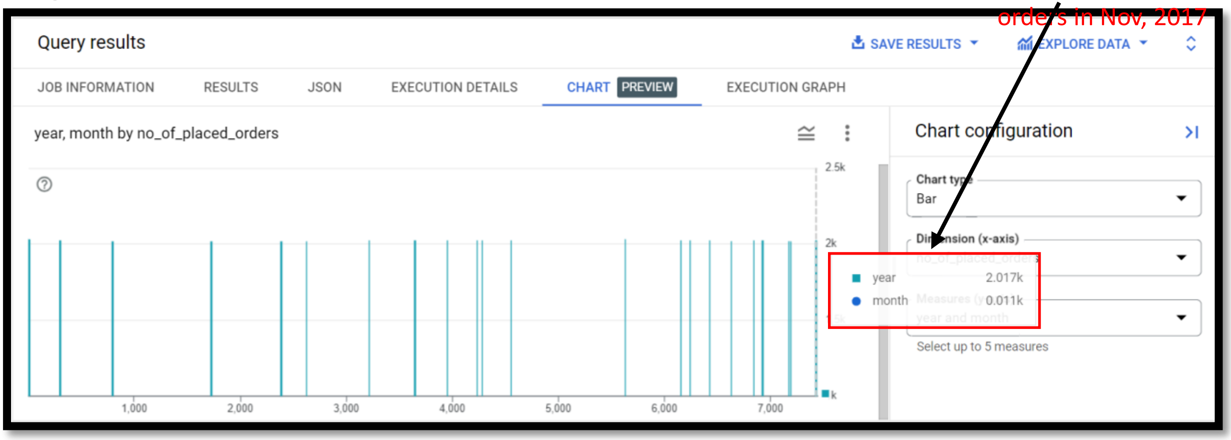


```
select
  extract(year from order_purchase_timestamp) as year,
```

Row	year	month	no_of_placed_orders
1	2016	9	1
2	2016	10	293
3	2016	12	1
4	2017	1	787
5	2017	2	1718
6	2017	3	2617
7	2017	4	2377
8	2017	5	3640
9	2017	6	3205
10	2017	7	3946
11	2017	8	4272
12	2017	9	4227
13	2017	10	4547

Insights:-

Highest no. of placed orders in Nov, 2017



Number of placed orders are varying as per the monthly seasonality. Through the output of the query, it is visible that in the year of 2016, October month has highest number of placed orders due to Halloween, while in the year of 2017, November month has the highest number of placed orders due to Black awareness day, Republic Proclamation Day, and upcoming new year celebration.

Recommendation:- The company should offer some discount and combos for Brazilian customers during the festival months also, in business “Anchored Price” concept can be used to increase the number of placed orders and sales.
(Anchored Price:- Price anchoring is a marketing strategy where a business establishes a visible starting price for a product but emphasizes its current discounted price. The initial price acts as a reference point or "anchor" against which the lower-priced option is contrasted, creating a perception of greater attractiveness for the discounted option.)

2.3 During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

- **0-6 hrs : Dawn**
- **7-12 hrs : Mornings**
- **13-18 hrs : Afternoon**
- **19-23 hrs : Night**

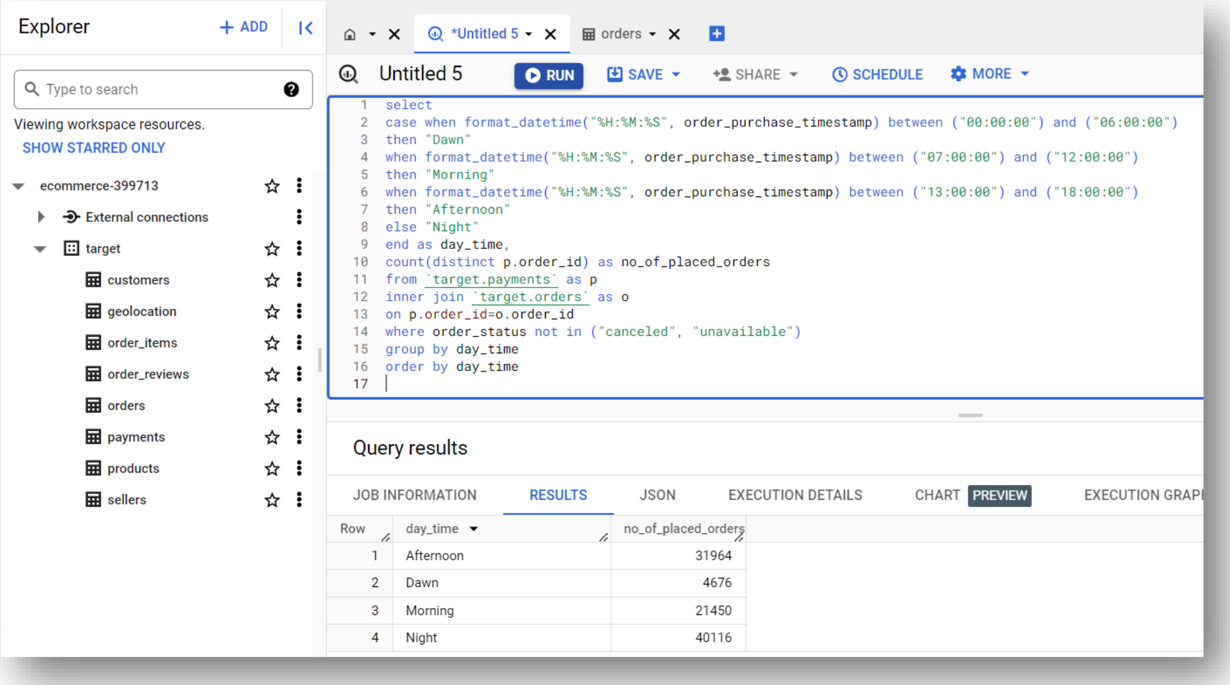
Assumption:-

- Considering, once the payment has been received for any purchase order that order is called as placed order. And not taking order status as cancelled and unavailable for placed orders.

Query:-

```
select
case when format_datetime("%H:%M:%S", order_purchase_timestamp) between ("00:00:00") and
("06:00:00")
then "Dawn"
when format_datetime("%H:%M:%S", order_purchase_timestamp) between ("07:00:00") and
("12:00:00")
then "Morning"
when format_datetime("%H:%M:%S", order_purchase_timestamp) between ("13:00:00") and
("18:00:00")
then "Afternoon"
else "Night"
end as day_time,
count(distinct p.order_id) as no_of_placed_orders
from `target.payments` as p
inner join `target.orders` as o
on p.order_id=o.order_id
where order_status not in ("canceled", "unavailable")
group by day_time
order by day_time
```

Output Screenshot:-



```sql
select
case when format_datetime("%H:%M:%S", order_purchase_timestamp) between ("00:00:00") and ("06:00:00")
then "Dawn"
when format_datetime("%H:%M:%S", order_purchase_timestamp) between ("07:00:00") and ("12:00:00")
then "Morning"
when format_datetime("%H:%M:%S", order_purchase_timestamp) between ("13:00:00") and ("18:00:00")
then "Afternoon"
else "Night"
end as day_time,
count(distinct p.order_id) as no_of_placed_orders
from `target.payments` as p
inner join `target.orders` as o
on p.order_id=o.order_id
where order_status not in ("canceled", "unavailable")
group by day_time
order by day_time
```

Row	day_time	no_of_placed_orders
1	Afternoon	31964
2	Dawn	4676
3	Morning	21450
4	Night	40116

Insights:-

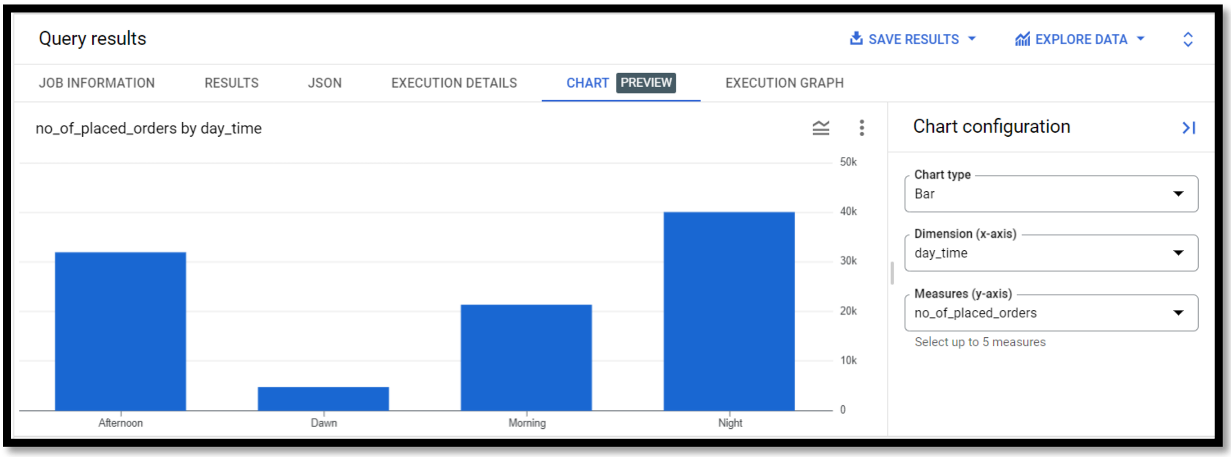


Chart type: Bar

Dimension (x-axis): day_time

Measures (y-axis): no_of_placed_orders

Through the query output, Brazilian customers mostly place their orders at night. This indicates that, customers are placing more orders after completing there day to day activities as per their convenience.

Recommendation:- With this information e-commerce company “Target” can make their marketing strategies to the specific time period that can maximize their reach to customers and sales.

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

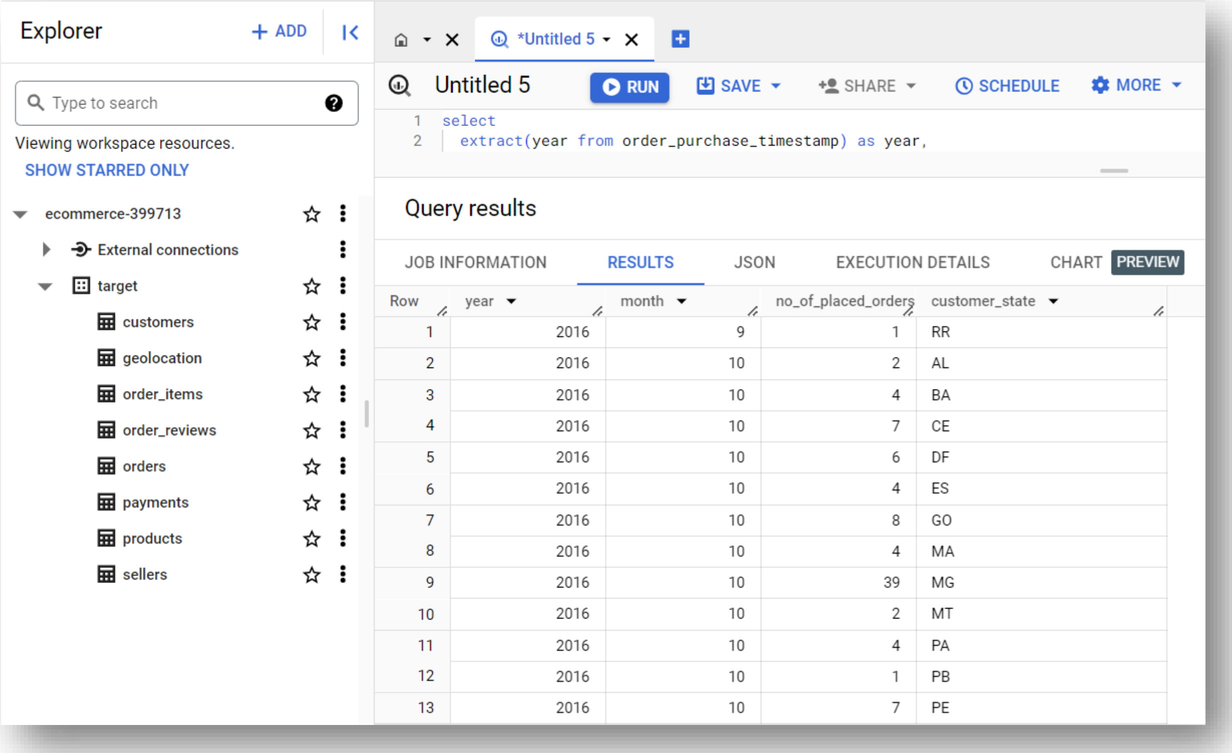
Assumption:-

- Considering, once the payment has been received for any purchase order that order is called as placed order. And not taking order status as cancelled and unavailable for placed orders.

Query:-

```
select
  extract(year from order_purchase_timestamp) as year,
  extract(month from order_purchase_timestamp) as month,
  count(distinct p.order_id) as no_of_placed_orders,
  customer_state
from `target.payments` as p
inner join `target.orders` as o
on p.order_id=o.order_id
inner join `target.customers` as c
on c.customer_id=o.customer_id
where order_status not in ("canceled", "unavailable")
group by year, month, customer_state
order by year, month, customer_state
```

Output Screenshot:-



```
select
  extract(year from order_purchase_timestamp) as year,
```

Row	year	month	no_of_placed_orders	customer_state
1	2016	9	1	RR
2	2016	10	2	AL
3	2016	10	4	BA
4	2016	10	7	CE
5	2016	10	6	DF
6	2016	10	4	ES
7	2016	10	8	GO
8	2016	10	4	MA
9	2016	10	39	MG
10	2016	10	2	MT
11	2016	10	4	PA
12	2016	10	1	PB
13	2016	10	7	PE

Insights:-

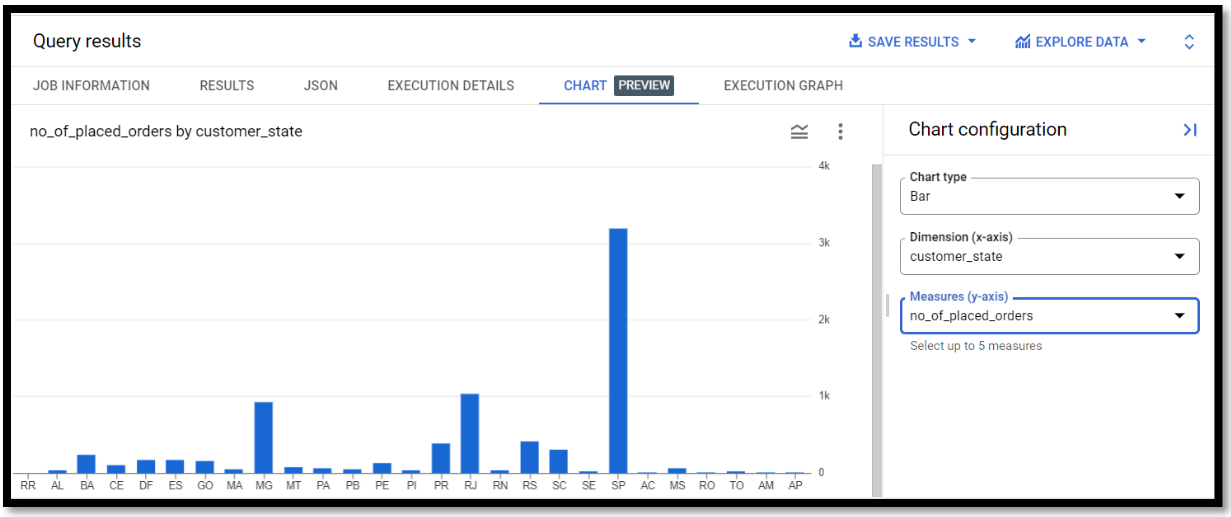


Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): no_of_placed_orders

Through the query output, São Paulo (SP) Brazilian state has the highest number of placed orders in October, 2016 and in November, 2017.

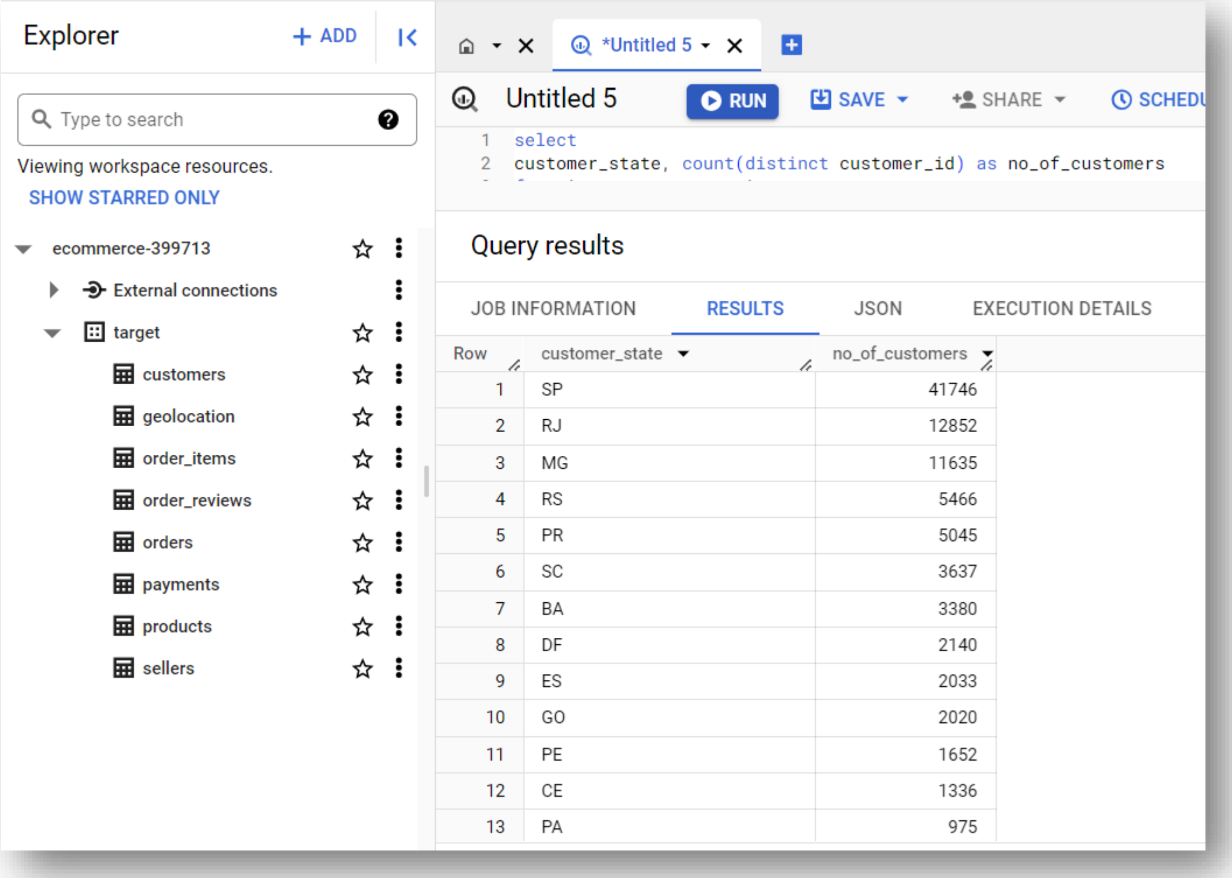
Recommendations:- Partnering with more vendors and sellers, customers can have variation in products and their price accordingly they can choose as per their budget which can lead to increment in order count and price as well.

3.2 How are the customers distributed across all the states?

Query:-

```
select
customer_state, count(distinct customer_id) as no_of_customers
from `target.customers`
group by customer_state
order by no_of_customers desc
```

Output Screenshot:-



```
select
customer_state, count(distinct customer_id) as no_of_customers
```

Row	customer_state	no_of_customers
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
13	PA	975

Insights:-

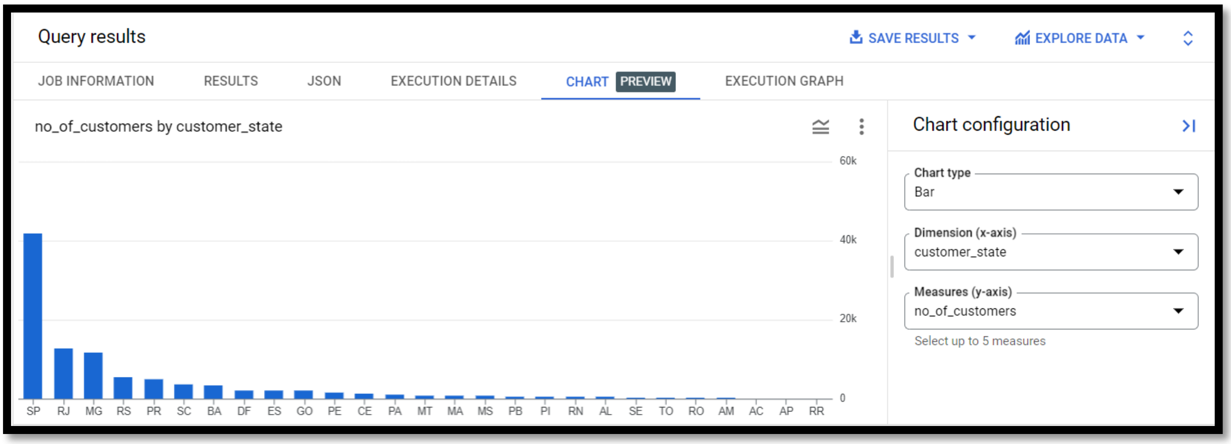


Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): no_of_customers

São Paulo (SP) Brazilian state has the highest number of “Target” customers. This analysis also indicates a good relation between number of customers in each state and number of orders.

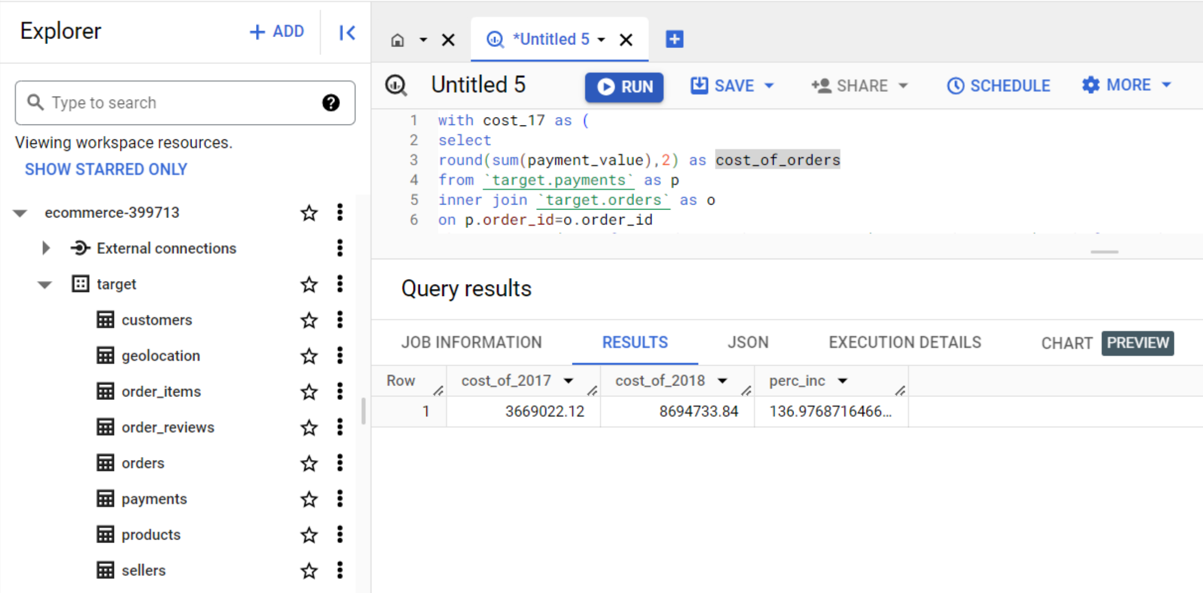
4.1 Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Query:-

```
with cost_17 as (
select
round(sum(payment_value),2) as cost_of_orders
from `target.payments` as p
inner join `target.orders` as o
on p.order_id=o.order_id
where extract(year from order_purchase_timestamp)=2017 and extract(month from
order_purchase_timestamp) between 1 and 8
), cost_18 as (
select
round(sum(payment_value),2) as cost_of_orders
from `target.payments` as p
inner join `target.orders` as o
on p.order_id=o.order_id
where extract(year from order_purchase_timestamp)=2018 and extract(month from
order_purchase_timestamp) between 1 and 8
)

select c17.cost_of_orders as cost_of_2017, c18.cost_of_orders as cost_of_2018,
 (((c18.cost_of_orders-c17.cost_of_orders)/c17.cost_of_orders)*100) as perc_inc
from cost_17 as c17, cost_18 as c18
```

Output Screenshot:-



```
with cost_17 as (
select
round(sum(payment_value),2) as cost_of_orders
from `target.payments` as p
inner join `target.orders` as o
on p.order_id=o.order_id
```

Row	cost_of_2017	cost_of_2018	perc_inc
1	3669022.12	8694733.84	136.9768716466...

Insights:- Overall percentage in cost of orders is increased by 136.98% from year 2017 to 2018, including the months from January to August only.

Recommendation:- After analysing this data, the company "Target" can expect more orders in upcoming years and can be ready for the marketing campaigns with catchy lines as per each age group, and using social media trending videos through which ads can be highlighted in different forms like reels over Instagram.

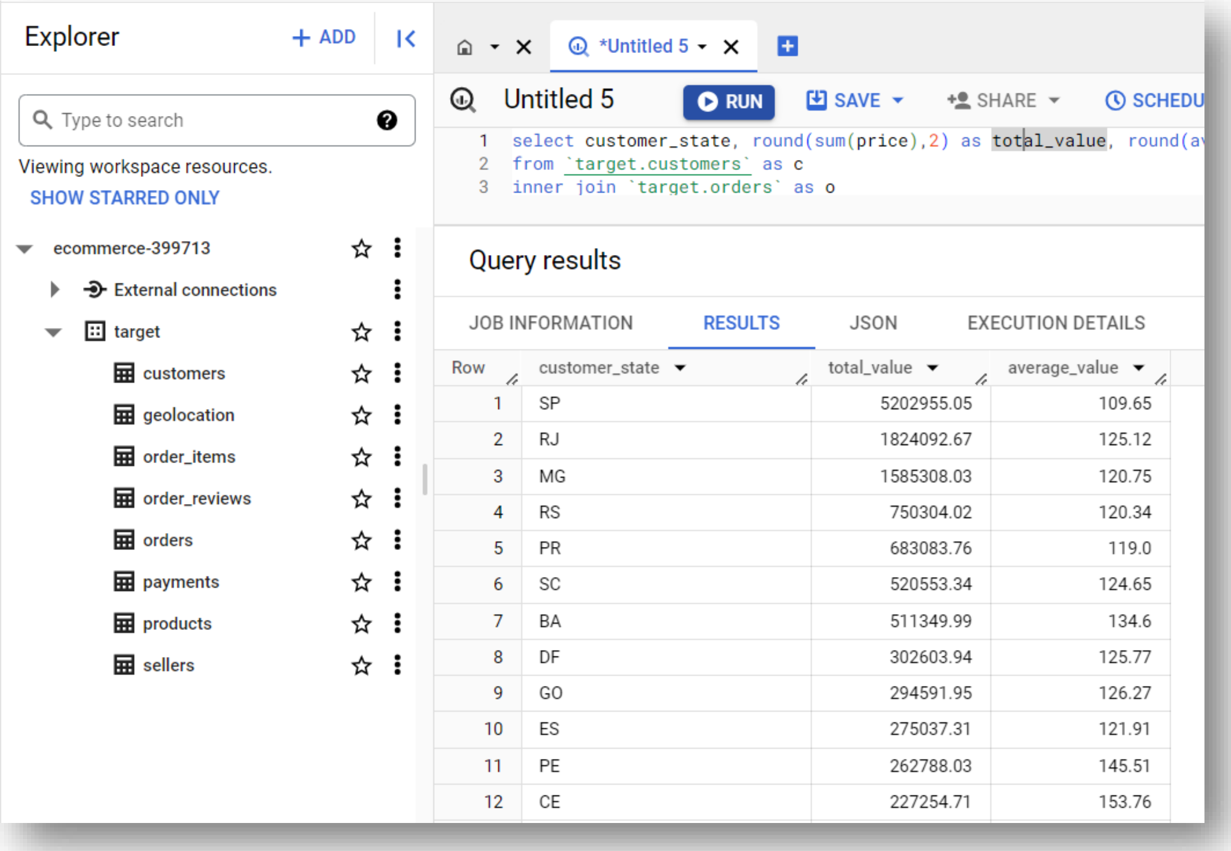
4.2 Calculate the Total & Average value of order price for each state.

Query:-

```
select customer_state, round(sum(price),2) as total_value, round(avg(price),2) as
average_value
from `target.customers` as c
```

```
inner join `target.orders` as o
on c.customer_id=o.customer_id
inner join `target.order_items` as oi
on o.order_id=oi.order_id
group by customer_state
order by total_value desc
```

Output Screenshot:-



```
select customer_state, round(sum(price),2) as total_value, round(a
from `target.customers` as c
inner join `target.orders` as o
```

Row	customer_state	total_value	average_value
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91
11	PE	262788.03	145.51
12	CE	227254.71	153.76

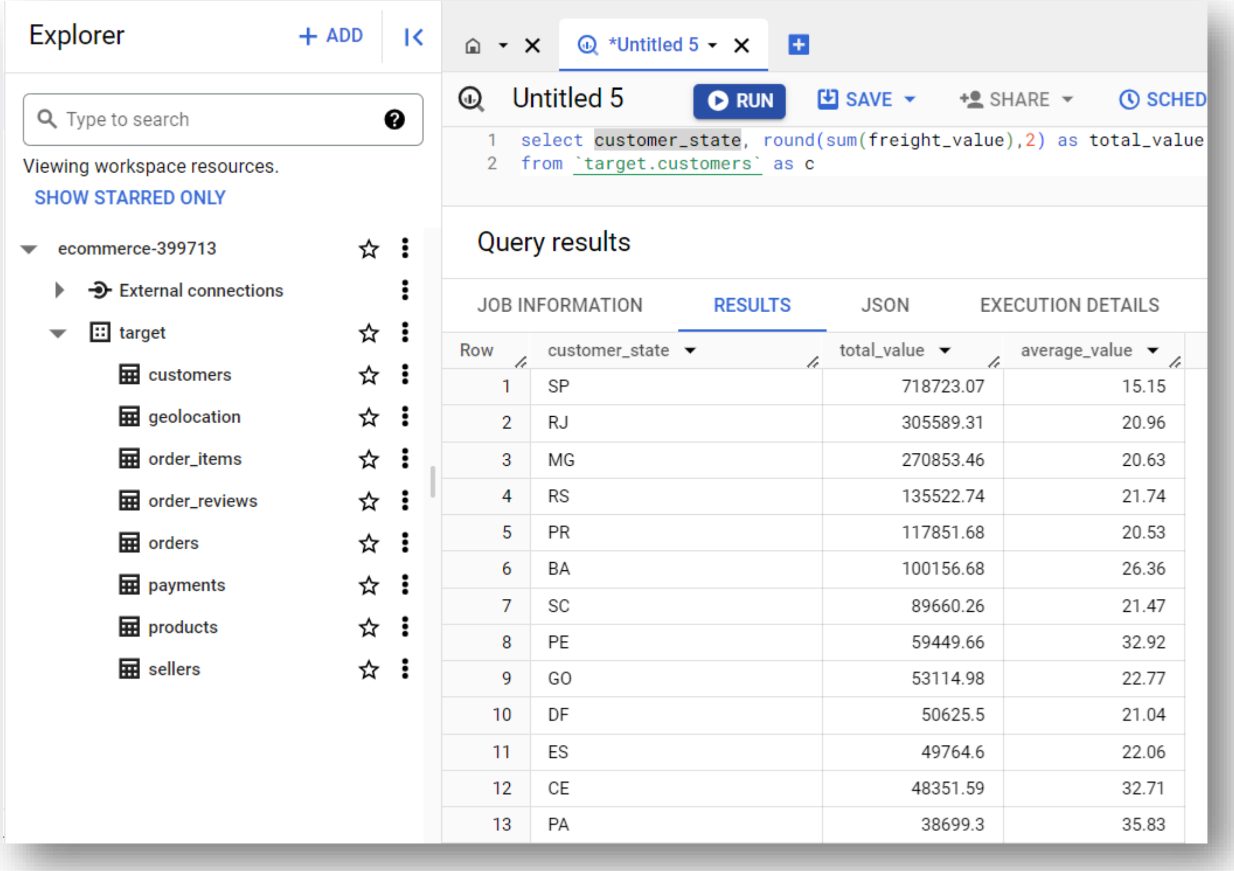
Insights:- São Paulo (SP) has the highest total value and lowest average value of order price while State Paraíba (PB) has the highest average value of order price among all states.

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

Query:-

```
select customer_state, round(sum(freight_value),2) as total_value,
round(avg(freight_value),2) as average_value
from `target.customers` as c
inner join `target.orders` as o
on c.customer_id=o.customer_id
inner join `target.order_items` as oi
on o.order_id=oi.order_id
group by customer_state
order by total_value desc
```

Output Screenshot:-



```
select customer_state, round(sum(freight_value),2) as total_value
from `target.customers` as c
```

Row	customer_state	total_value	average_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04
11	ES	49764.6	22.06
12	CE	48351.59	32.71
13	PA	38699.3	35.83

Insights:- São Paulo (SP) has the highest total value and lowest average value of order freight while Roraima (RR) has the highest average value of order freight among all states.

5.1 Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
Using the below formulas to calculate the delivery time and difference between the estimated and actual delivery date:-

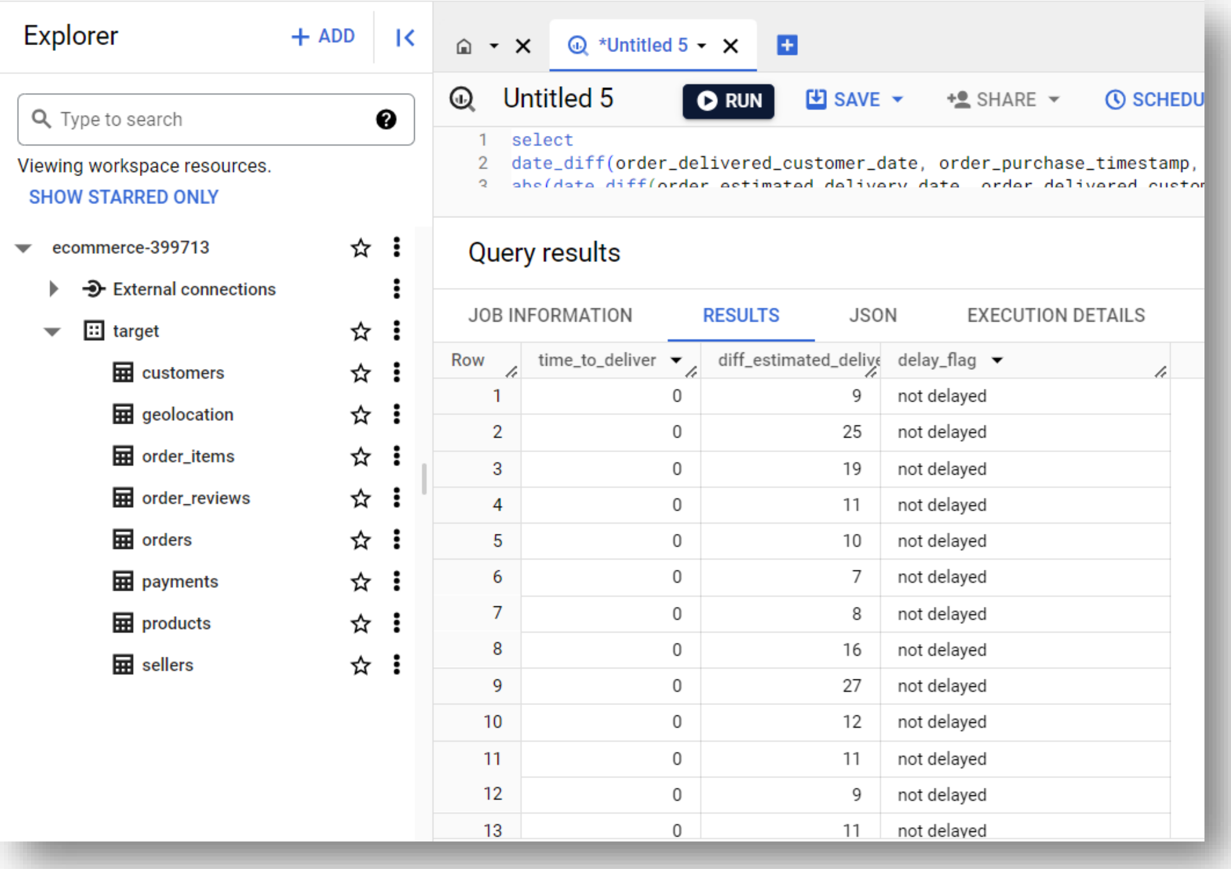
- **time_to_deliver = order_delivered_customer_date - order_purchase_timestamp**
- **diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date**

Query:-

```
select
date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as time_to_deliver,
abs(date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)) as
diff_estimated_delivery,
case when date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)<0
then "delayed"
else "not delayed"
end as delay_flag
from `target.orders`
```

```
where order_delivered_customer_date is not null and order_purchase_timestamp is not null
and order_estimated_delivery_date is not null
order by time_to_deliver
```

Output Screenshot:-



```
select
date_diff(order_delivered_customer_date, order_purchase_timestamp,
abs(date_diff(order_estimated_delivery_date, order_delivered_custom
```

Query results

Row	time_to_deliver	diff_estimated_delive	delay_flag
1	0	9	not delayed
2	0	25	not delayed
3	0	19	not delayed
4	0	11	not delayed
5	0	10	not delayed
6	0	7	not delayed
7	0	8	not delayed
8	0	16	not delayed
9	0	27	not delayed
10	0	12	not delayed
11	0	11	not delayed
12	0	9	not delayed
13	0	11	not delayed

Insights:- With the thorough analysis of delivery time, there are 6535 delayed deliveries and 89941 not delayed deliveries.

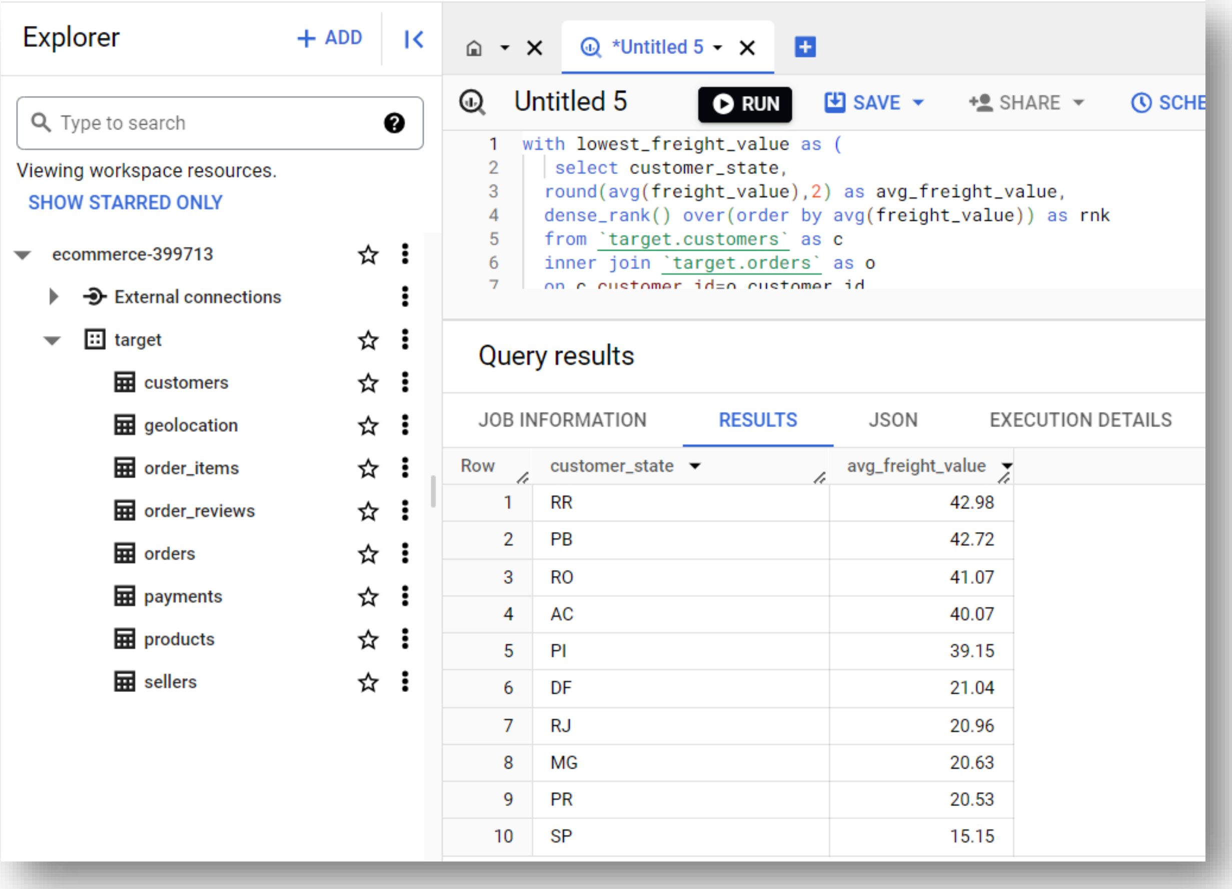
5.2 Find out the top 5 states with the highest & lowest average freight value.

Query:-

```
with lowest_freight_value as (
   select customer_state,
  round(avg(freight_value),2) as avg_freight_value,
  dense_rank() over(order by avg(freight_value)) as rnk
  from `target.customers` as c
  inner join `target.orders` as o
  on c.customer_id=o.customer_id
  inner join `target.order_items` as oi
  on o.order_id=oi.order_id
  group by customer_state
), highest_freight_value as (
  select customer_state,
  round(avg(freight_value),2) as avg_freight_value,
  dense_rank() over(order by avg(freight_value) desc) as rnk
  from `target.customers` as c
  inner join `target.orders` as o
  on c.customer_id=o.customer_id
  inner join `target.order_items` as oi
  on o.order_id=oi.order_id
  group by customer_state
)

select l.customer_state, l.avg_freight_value
from lowest_freight_value as l
where l.rnk between 1 and 5
union all
select h.customer_state, h.avg_freight_value
from highest_freight_value as h
where h.rnk between 1 and 5
order by avg_freight_value desc
```

Output Screenshot:-



```
with lowest_freight_value as (
  select customer_state,
  round(avg(freight_value),2) as avg_freight_value,
  dense_rank() over(order by avg(freight_value)) as rnk
  from `target.customers` as c
  inner join `target.orders` as o
  on c.customer_id=o.customer_id
```

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

Insights:- Roraima (RR) has the highest average freight value among all states

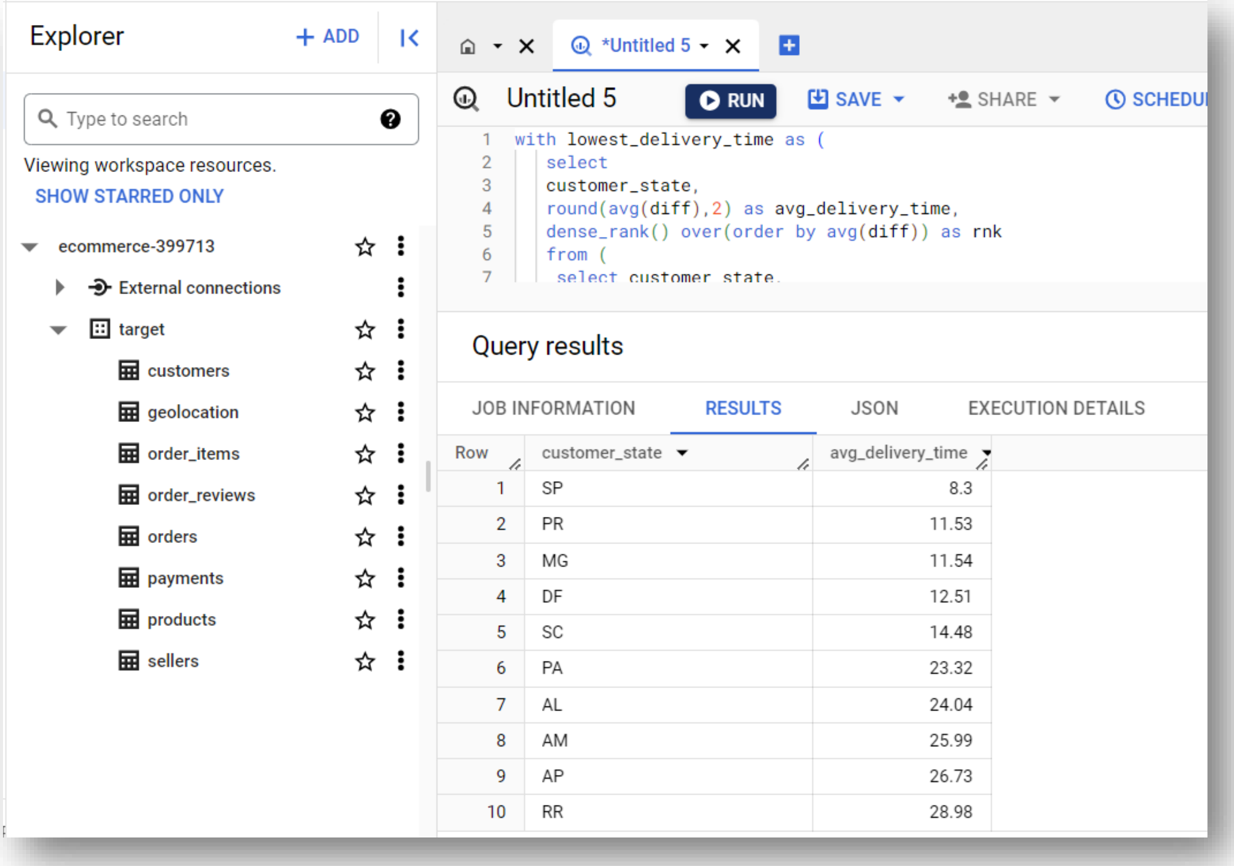
5.3 Find out the top 5 states with the highest & lowest average delivery time.

Query:-

```
with lowest_delivery_time as (
   select
   customer_state,
   round(avg(diff),2) as avg_delivery_time,
   dense_rank() over(order by avg(diff)) as rnk
   from (
    select customer_state,
    date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as diff
    from `target.customers` as c
    inner join `target.orders` as o
    on c.customer_id=o.customer_id
    where order_delivered_customer_date is not null and order_purchase_timestamp is not
null
   )
  group by customer_state
),
highest_delivery_time as (
  select
  customer_state,
  round(avg(diff),2) as avg_delivery_time,
  dense_rank() over(order by avg(diff)desc) as rnk
  from (
    select customer_state,
    date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as diff
    from `target.customers` as c
    inner join `target.orders` as o
    on c.customer_id=o.customer_id
    where order_delivered_customer_date is not null and order_purchase_timestamp is not
null
  )
  group by customer_state
)

select l.customer_state, l.avg_delivery_time
from lowest_delivery_time as l
where l.rnk between 1 and 5
union all
select h.customer_state, h.avg_delivery_time
from highest_delivery_time as h
where h.rnk between 1 and 5
order by avg_delivery_time
```

Output Screenshot:-



```
with lowest_delivery_time as (
  select
  customer_state,
  round(avg(diff),2) as avg_delivery_time,
  dense_rank() over(order by avg(diff)) as rnk
  from (
   select customer_state,
```

Row	customer_state	avg_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98

Insights:- São Paulo (SP) has the lowest average delivery time where the fastest delivery take place and Roraima (RR) state has the highest average delivery time which means that RR state has slowest delivery in Brazil.

Recommendation:- Logistics and shipping processes needs to be improve for customer satisfaction which includes the refinement of shipping routes and partnering with more courier services.

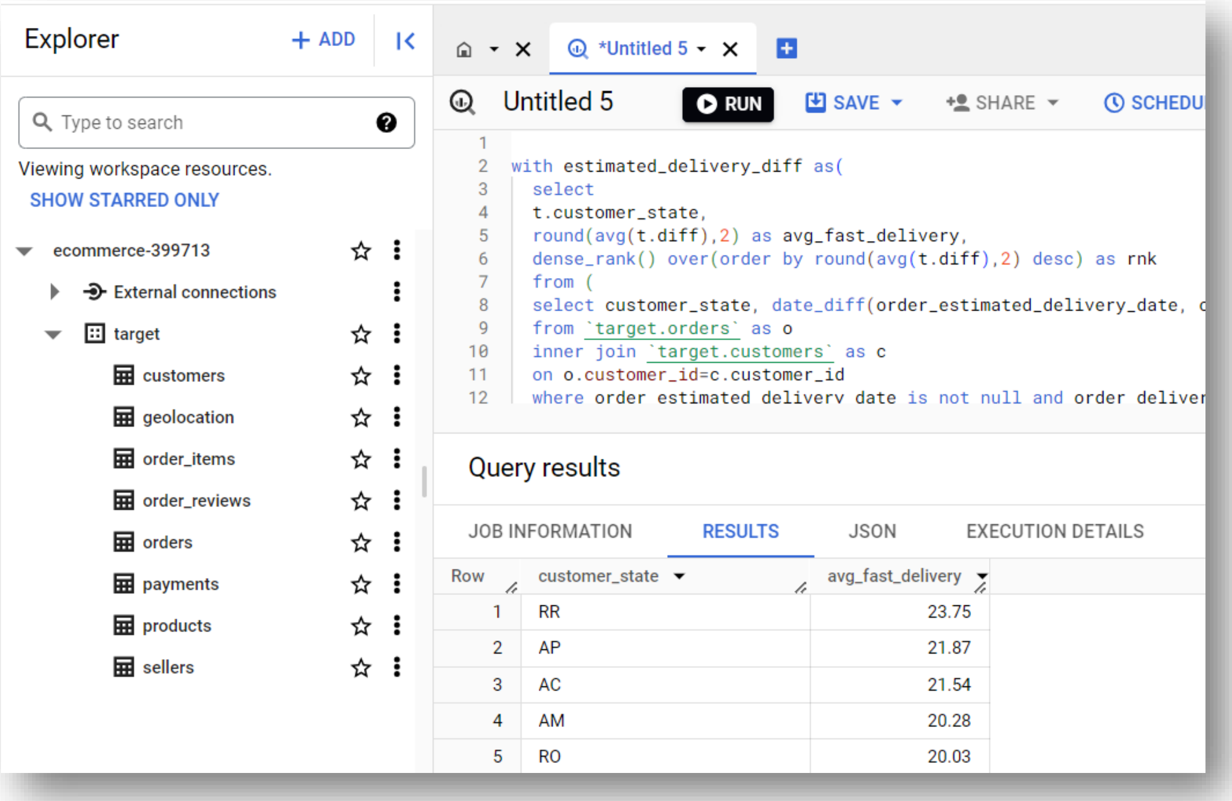
5.4 Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

Query:-

```
with estimated_delivery_diff as(
  select
  t.customer_state,
  round(avg(t.diff),2) as avg_fast_delivery,
  dense_rank() over(order by round(avg(t.diff),2) desc) as rnk
  from (
  select customer_state, date_diff(order_estimated_delivery_date,
order_delivered_customer_date, day) as diff
  from `target.orders` as o
  inner join `target.customers` as c
  on o.customer_id=c.customer_id
  where order_estimated_delivery_date is not null and order_delivered_customer_date is not
null and date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)>0
) as t
group by customer_state
)

select edd.customer_state, edd.avg_fast_delivery
from estimated_delivery_diff as edd
where rnk between 1 and 5
order by avg_fast_delivery desc
```

Output Screenshot:-



```

with estimated_delivery_diff as(
select
t.customer_state,
round(avg(t.diff),2) as avg_fast_delivery,
dense_rank() over(order by round(avg(t.diff),2) desc) as rnk
from (
select customer_state, date_diff(order_estimated_delivery_date, c
from `target.orders` as o
inner join `target.customers` as c
on o.customer_id=c.customer_id
where order estimated deliverv date is not null and order deliver
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	avg_fast_delivery
1	RR	23.75
2	AP	21.87
3	AC	21.54
4	AM	20.28
5	RO	20.03

Insights:-

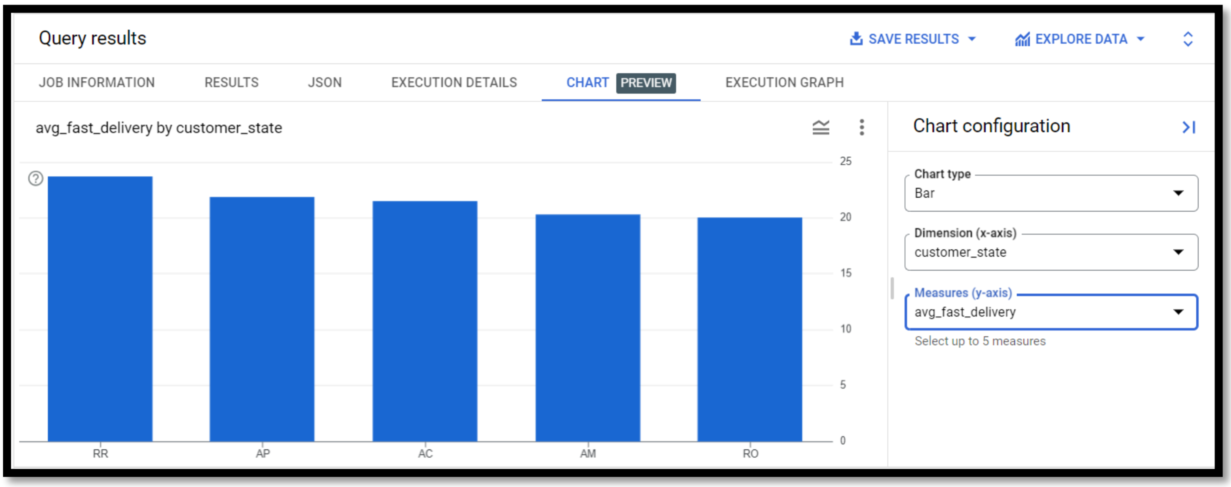


Chart type: Bar

Dimension (x-axis): customer_state

Measures (y-axis): avg_fast_delivery

Select up to 5 measures

Roraima (RR) state has the highest value of average of fast deliver that means, in RR deliveries are really fast as compared to the estimated date of delivery.

For the fastest delivery compared to estimated delivery date, calculated the difference of estimated date of delivery and the date when the product is delivered to customer the more the difference is the fast delivery is made.

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

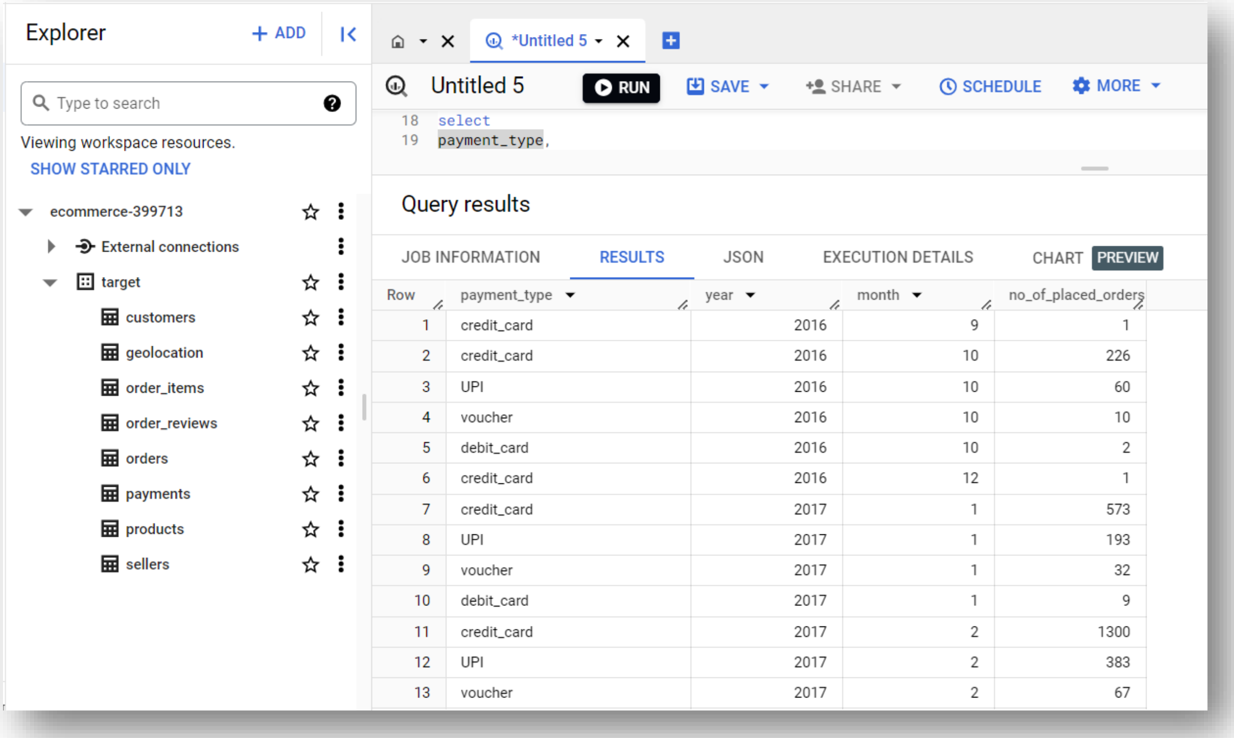
Assumption:-

- Considering, once the payment has been received for any purchase order that order is called as placed order. And not taking order status as cancelled and unavailable for placed orders.

Query:-

```
select
payment_type,
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
count(distinct p.order_id) as no_of_placed_orders
from `target.payments` as p
inner join `target.orders` as o
on p.order_id=o.order_id
where order_status not in ("canceled", "unavailable")
group by year, month, payment_type
order by  year, month, no_of_placed_orders desc
```

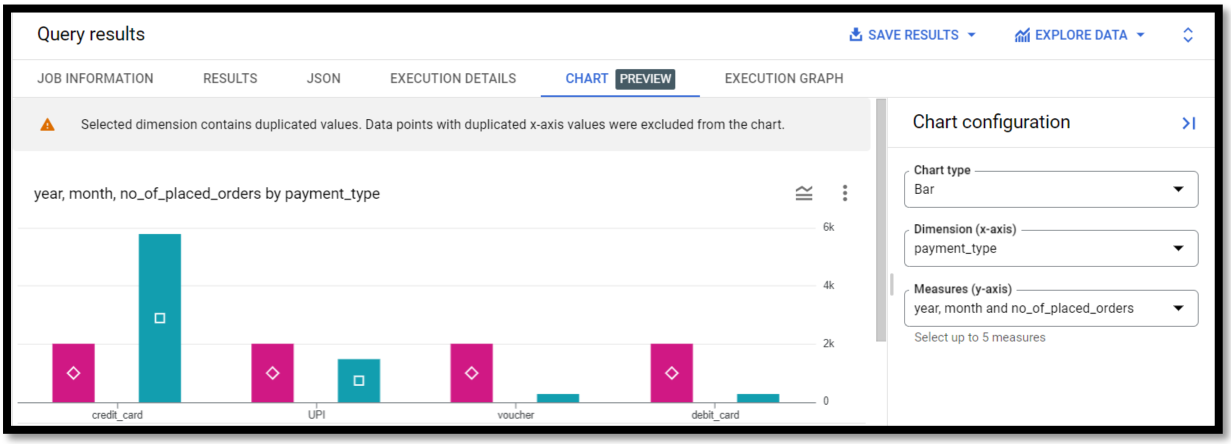
Output Screenshot:-



```
select
payment_type,
```

Row	payment_type	year	month	no_of_placed_orders
1	credit_card	2016	9	1
2	credit_card	2016	10	226
3	UPI	2016	10	60
4	voucher	2016	10	10
5	debit_card	2016	10	2
6	credit_card	2016	12	1
7	credit_card	2017	1	573
8	UPI	2017	1	193
9	voucher	2017	1	32
10	debit_card	2017	1	9
11	credit_card	2017	2	1300
12	UPI	2017	2	383
13	voucher	2017	2	67

Insights:-



Selected dimension contains duplicated values. Data points with duplicated x-axis values were excluded from the chart.

year, month, no_of_placed_orders by payment_type

Chart type: Bar

Dimension (x-axis): payment_type

Measures (y-axis): year, month and no_of_placed_orders

Select up to 5 measures

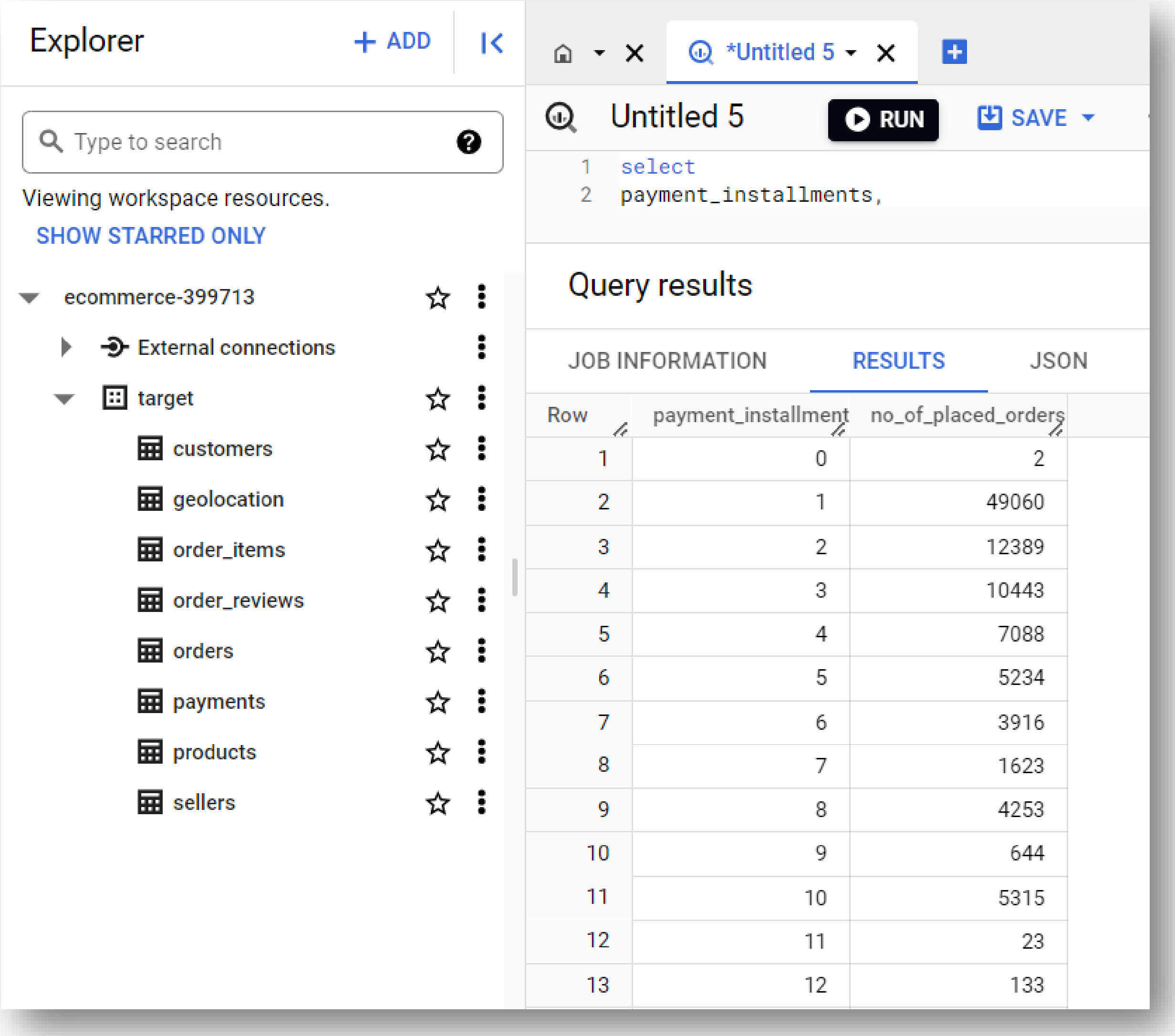
Credit card are mostly used to place the order due to the benefits such as; “buy now, pay later” with cashback and EMI offers.

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

Query:-

```
select
payment_installments,
count(distinct order_id) as no_of_placed_orders
from `target.payments`
group by payment_installments
order by payment_installments
```

Output Screenshot:-



Explorer

+ ADD

Type to search

Viewing workspace resources.

SHOW STARRED ONLY

- ecommerce-399713
  - External connections
  - target
    - customers
    - geolocation
    - order_items
    - order_reviews
    - orders
    - payments
    - products
    - sellers

*Untitled 5

Untitled 5

RUN

SAVE

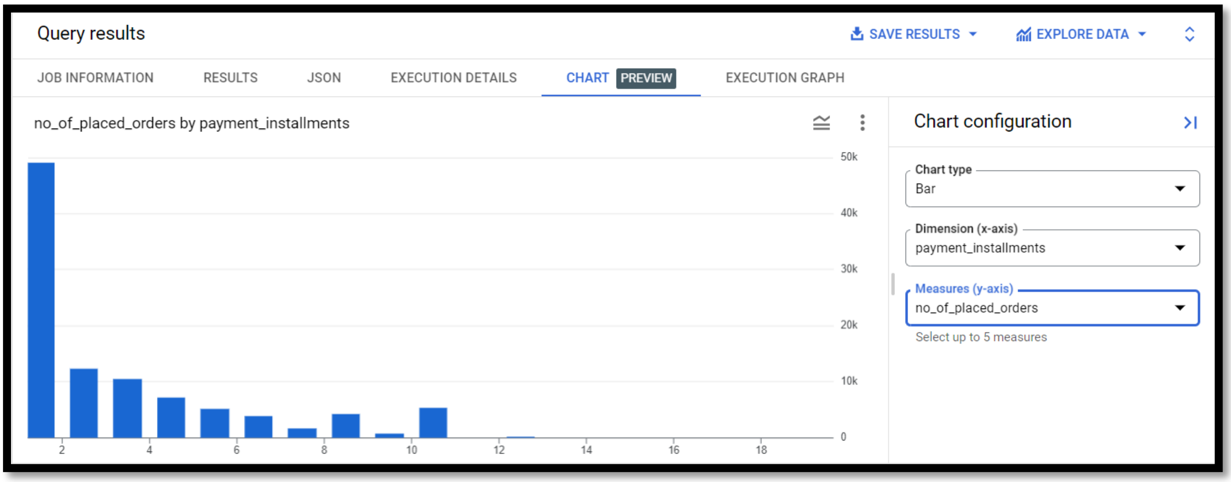
```
select
payment_installments,
```

Query results

JOB INFORMATION RESULTS JSON

Row	payment_installment	no_of_placed_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644
11	10	5315
12	11	23
13	12	133

Insights:-



Query results

SAVE RESULTS EXPLORE DATA

JOB INFORMATION RESULTS JSON EXECUTION DETAILS CHART PREVIEW EXECUTION GRAPH

no_of_placed_orders by payment_installments

Chart configuration

Chart type: Bar

Dimension (x-axis): payment_installments

Measures (y-axis): no_of_placed_orders

Select up to 5 measures

Most of the placed orders are associate with only one payment installments