TARGET – SQL PROJECT

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

1. Data type of columns in a table.
2. Time period for which the data is given.
3. Cities and States of customers ordered during the given period.

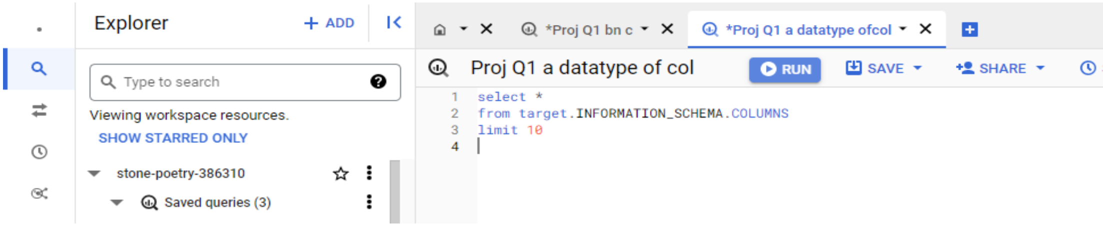
Solution:

1. i. Data type of columns in a table:

SQL QUERY:

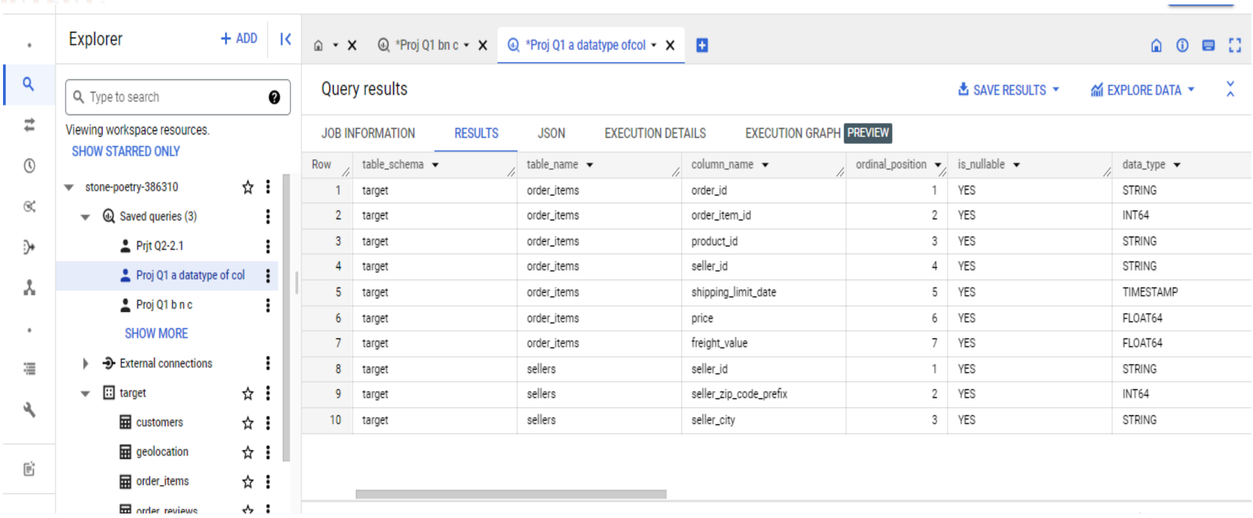
```
select *
from target.INFORMATION_SCHEMA.COLUMNS
limit 10
```

QUERY SCREENSHOT:



```
select *
from target.INFORMATION_SCHEMA.COLUMNS
limit 10
```

OUTPUT:



Row	table_schema	table_name	column_name	ordinal_position	is_nullable	data_type
1	target	order_items	order_id	1	YES	STRING
2	target	order_items	order_item_id	2	YES	INT64
3	target	order_items	product_id	3	YES	STRING
4	target	order_items	seller_id	4	YES	STRING
5	target	order_items	shipping_limit_date	5	YES	TIMESTAMP
6	target	order_items	price	6	YES	FLOAT64
7	target	order_items	freight_value	7	YES	FLOAT64
8	target	sellers	seller_id	1	YES	STRING
9	target	sellers	seller_zip_code_prefix	2	YES	INT64
10	target	sellers	seller_city	3	YES	STRING

ANALYSIS:

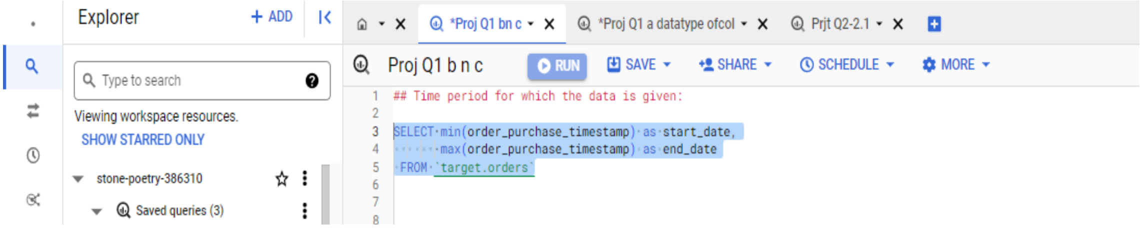
- ✓ Using ‘Information_schema’, we can get the data type of all the columns from the tables of the dateset.
- ✓ With this information, we know what type of data (i.e int, float, string, time) exists in the given dataset.

1.ii Time period for which the data is given:

SQL QUERY:

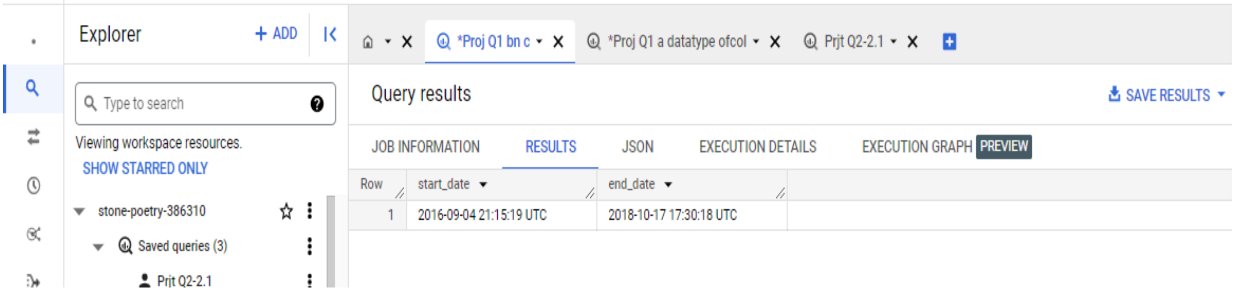
```
SELECT min(order_purchase_timestamp) as start_date,
       max(order_purchase_timestamp) as end_date
 FROM `target.orders`
```

QUERY SCREENSHOT:



```
## Time period for which the data is given:

SELECT min(order_purchase_timestamp) as start_date,
      max(order_purchase_timestamp) as end_date
 FROM `target.orders`
```

OUTPUT:



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

ANALYSIS:

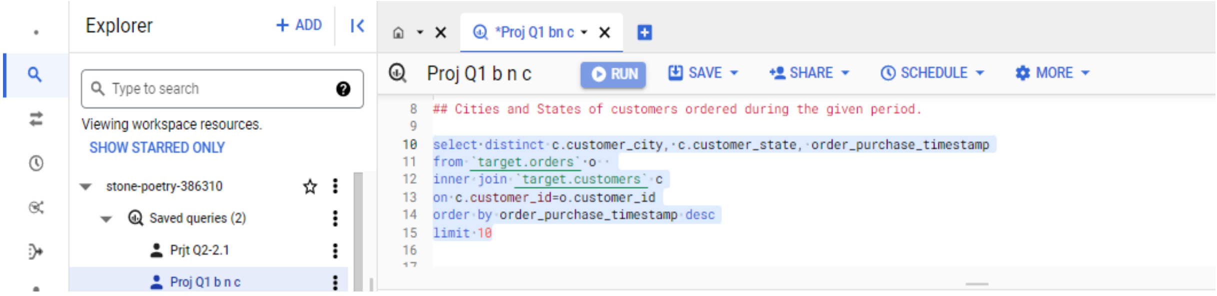
- ✓ The provided data is from **2016 September – 2018 October.** Reconfirming the same using the above query.

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

SQL QUERY:

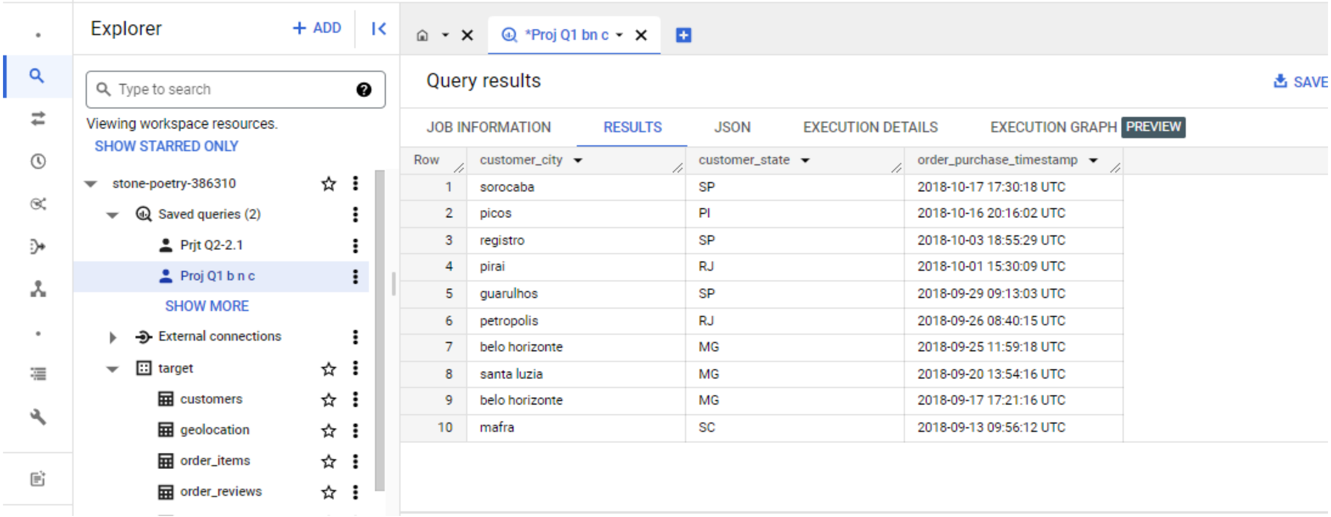
```
select distinct c.customer_city, c.customer_state, order_purchase_timestamp
from `target.orders` o
inner join `target.customers` c
on c.customer_id=o.customer_id
order by order_purchase_timestamp desc
limit 10
```

QUERY SCREENSHOT:



```
## Cities and States of customers ordered during the given period.

select distinct c.customer_city, c.customer_state, order_purchase_timestamp
from `target.orders` o
inner join `target.customers` c
on c.customer_id=o.customer_id
order by order_purchase_timestamp desc
limit 10

```

OUTPUT:



Row	customer_city	customer_state	order_purchase_timestamp
1	sorocaba	SP	2018-10-17 17:30:18 UTC
2	picos	PI	2018-10-16 20:16:02 UTC
3	registro	SP	2018-10-03 18:55:29 UTC
4	pirai	RJ	2018-10-01 15:30:09 UTC
5	guarulhos	SP	2018-09-29 09:13:03 UTC
6	petropolis	RJ	2018-09-26 08:40:15 UTC
7	belo horizonte	MG	2018-09-25 11:59:18 UTC
8	santa luzia	MG	2018-09-20 13:54:16 UTC
9	belo horizonte	MG	2018-09-17 17:21:16 UTC
10	mafra	SC	2018-09-13 09:56:12 UTC

ANALYSIS:

- ✓ With the above query,we get to know the list of customers city and state, based on the order_purchase_timestamp.

===

Q.2. In-depth Exploration:

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

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

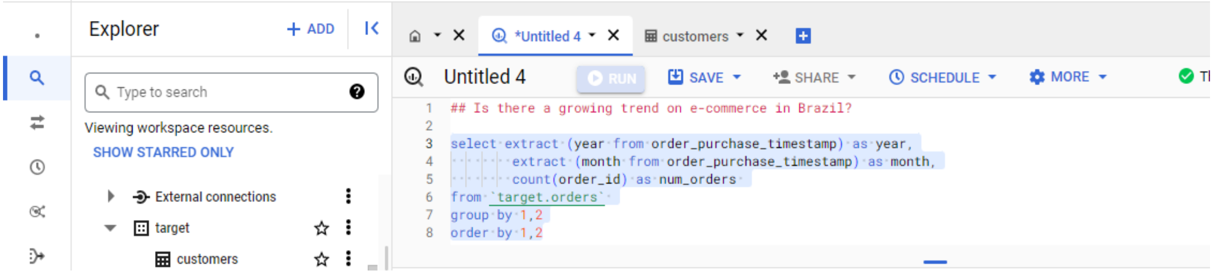
Solution:

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

SQL QUERY:

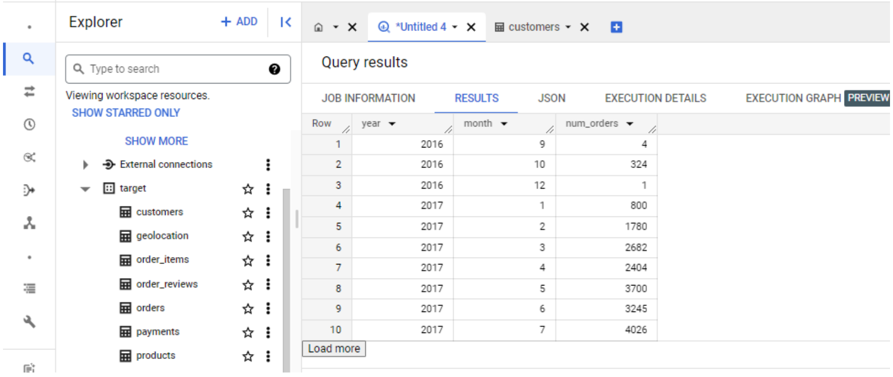
```
select extract (year from order_purchase_timestamp) as year,
       extract (month from order_purchase_timestamp) as month,
       count(order_id) as num_orders
from `target.orders`
group by 1,2
order by 1,2
```

QUERY SCREENSHOT:



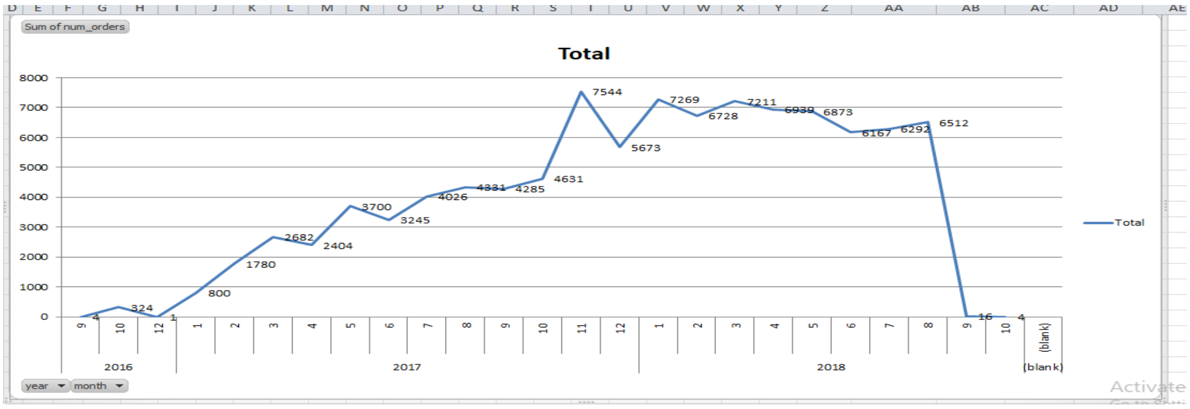
```
## Is there a growing trend on e-commerce in Brazil?

select extract (year from order_purchase_timestamp) as year,
      extract (month from order_purchase_timestamp) as month,
      count(order_id) as num_orders
from `target.orders`
group by 1,2
order by 1,2
```

OUTPUT:



Row	year	month	num_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

CHART:



ANALYSIS:

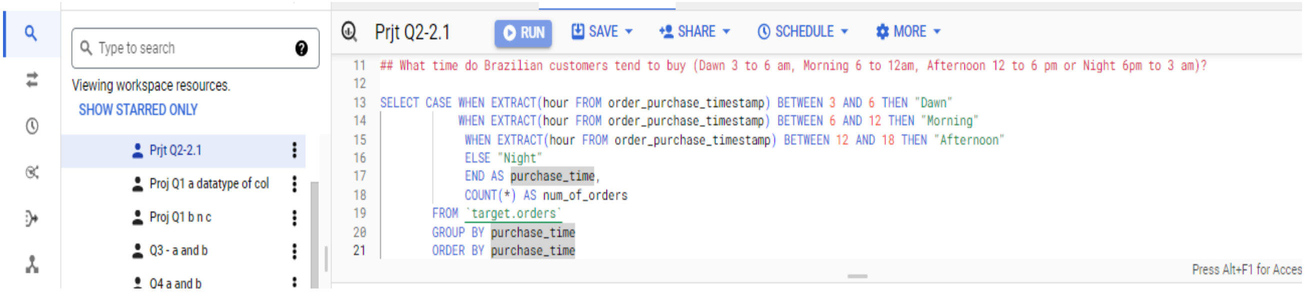
- ✓ There is an increasing trend in the e-commerce business in Brazil since 2016 as we can see an upward trend in the orders.
- ✓ It was at its highest in November, 2017 and thereafter had started to decline gradually.
- ✓ By the end of 2018, it has reached the same point where it was in the year 2016 reflecting a kind of cyclical trend.

2.2 What time do Brazilian customers tend to buy (Dawn 3 to 6 am, Morning 6 to 12am, Afternoon 12 to 6 pm or Night 6pm to 3 am)?

SQL QUERY:

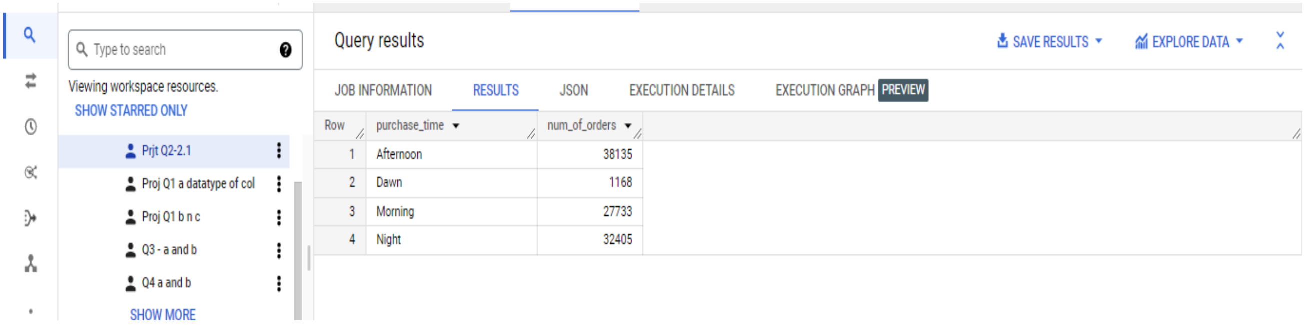
```
SELECT CASE WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 3 AND 6 THEN "Dawn"
            WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 6 AND 12 THEN "Morning"
             WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 12 AND 18 THEN "Afternoon"
             ELSE "Night"
             END AS purchase_time,
             COUNT(*) AS num_of_orders
        FROM `target.orders`
        GROUP BY purchase_time
        ORDER BY purchase_time
```

QUERY SCREENSHOT:



```
## What time do Brazilian customers tend to buy (Dawn 3 to 6 am, Morning 6 to 12am, Afternoon 12 to 6 pm or Night 6pm to 3 am)?

SELECT CASE WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 3 AND 6 THEN "Dawn"
            WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 6 AND 12 THEN "Morning"
             WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 12 AND 18 THEN "Afternoon"
             ELSE "Night"
             END AS purchase_time,
             COUNT(*) AS num_of_orders
        FROM `target.orders`
        GROUP BY purchase_time
        ORDER BY purchase_time
```

OUTPUT:



Row	purchase_time	num_of_orders
1	Afternoon	38135
2	Dawn	1168
3	Morning	27733
4	Night	32405

ANALYSIS:

- ✓ With the above query, we can interpret that most of the orders are place in the afternoon (i.e. 12 pm to 6pm) and then night time i.e. 6pm to 3 am.
- ✓ Remaining orders are placed during morning time i.e. from 6am to 12pm.
- ✓ Very few orders are placed during 'Dawn' time.

==

Q.3. Evolution of E-commerce orders in the Brazil region:
i. Get month on month orders by states.
ii. Distribution of customers across the states in Brazil.

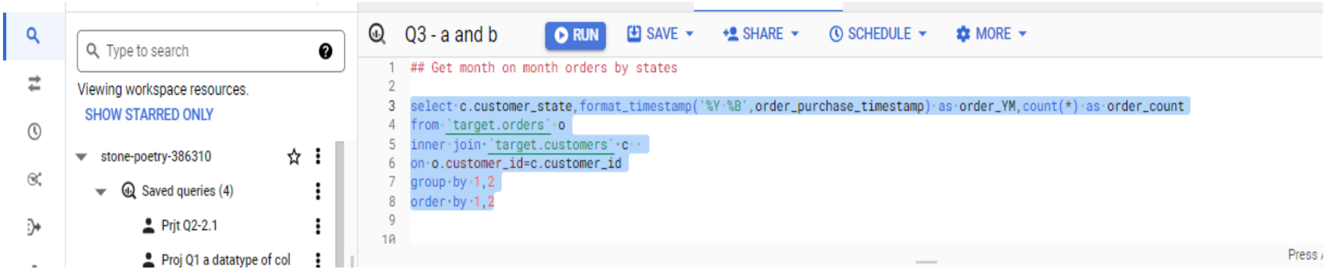
Solution:

3. i Get month on month orders by states.

QUERY:

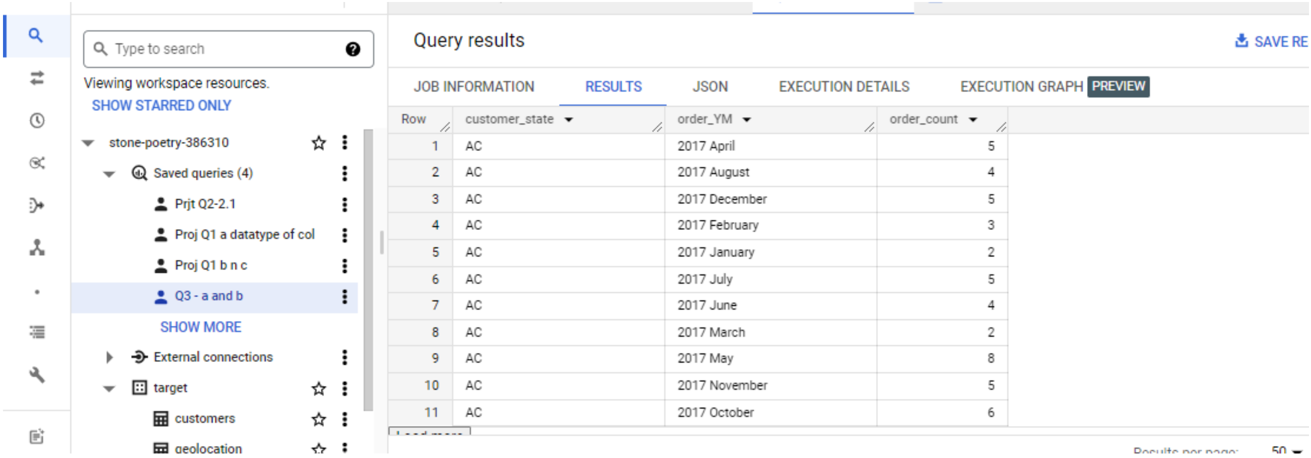
```
select c.customer_state,format_timestamp('%Y %B',order_purchase_timestamp) as order_YM,count(*) as order_count
from `target.orders` o
inner join `target.customers` c
on o.customer_id=c.customer_id
group by 1,2
order by 1,2
```

QUERY SCREENSHOT:



```
## Get month on month orders by states

select c.customer_state,format_timestamp('%Y %B',order_purchase_timestamp) as order_YM,count(*) as order_count
from `target.orders` o
inner join `target.customers` c
on o.customer_id=c.customer_id
group by 1,2
order by 1,2
```

OUTPUT:



Row	customer_state	order_YM	order_count
1	AC	2017 April	5
2	AC	2017 August	4
3	AC	2017 December	5
4	AC	2017 February	3
5	AC	2017 January	2
6	AC	2017 July	5
7	AC	2017 June	4
8	AC	2017 March	2
9	AC	2017 May	8
10	AC	2017 November	5
11	AC	2017 October	6

ANALYSIS:

- ✓ From the above output, we can analyse number of orders from each state for each month.
- ✓ Based on this we can try to understand which state has more orders, which has less order and in which particular month.
- ✓ We can focus on the marketing and selling strategies for states from where we get less number of orders.
- ✓ Also we can check if the customers are following any ordering patterns during any of the months.

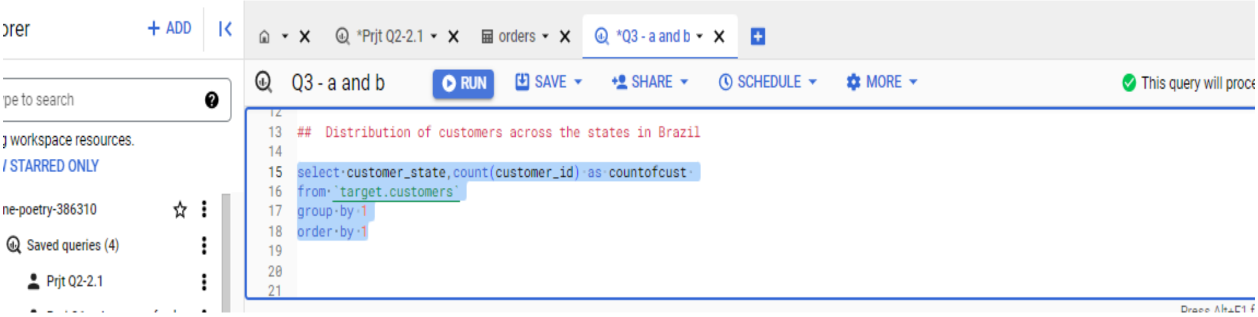
---

3. ii Distribution of customers across the states in Brazil.

SQL QUERY:

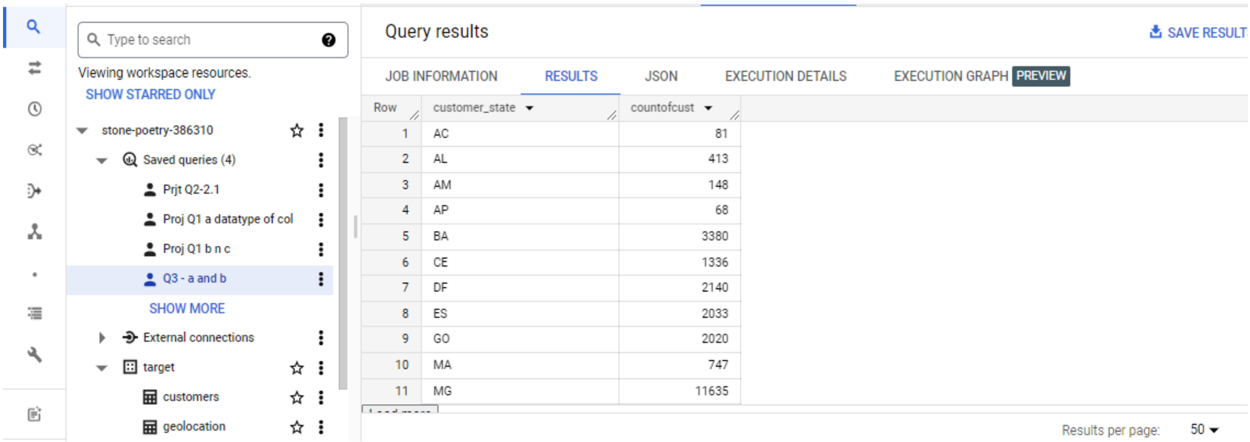
```
select customer_state,count(customer_id) as countofcust
from `target.customers`
group by 1
order by 1
```

QUERY SCREENSHOT:



```
##  Distribution of customers across the states in Brazil

select customer_state,count(customer_id) as countofcust
from `target.customers`
group by 1
order by 1
```

OUTPUT:



Row	customer_state	countofcust
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635

ANALYSIS:

- ✓ This query helps us in understanding which state has the highest/lowest number of customers.
- ✓ Allows us to focus on increasing the demand by understanding as to which product is in more demand in that particular region.

==

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

i. **Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table.**

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

Solution:

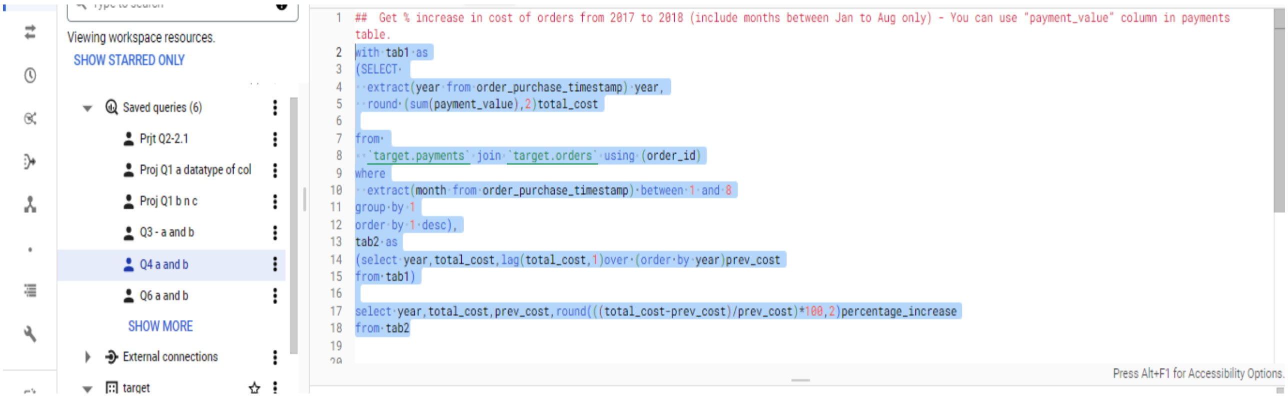
4. i. Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table.

SQL QUERY:

```
with tab1 as
(SELECT
  extract(year from order_purchase_timestamp) year,
  round (sum(payment_value),2)total_cost

from
  `target.payments` join `target.orders` using (order_id)
where
  extract(month from order_purchase_timestamp) between 1 and 8
group by 1
order by 1 desc),
tab2 as
(select year,total_cost,lag(total_cost,1)over (order by year)prev_cost
from tab1)
```

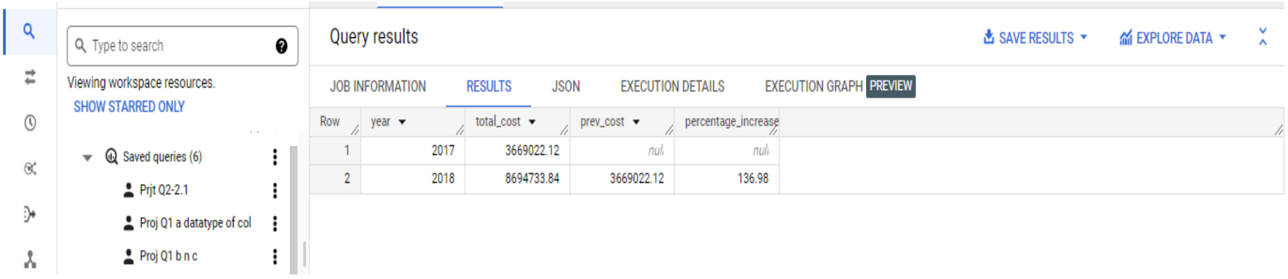
```
select year,total_cost,prev_cost,round(((total_cost-prev_cost)/prev_cost)*100,2)percentage_increase
from tab2
```

QUERY SCREENSHOT:



```
##  Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use "payment_value" column in payments table.
with tab1 as
(SELECT
  extract(year from order_purchase_timestamp) year,
  round (sum(payment_value),2)total_cost

from
  `target.payments` join `target.orders` using (order_id)
where
  extract(month from order_purchase_timestamp) between 1 and 8
group by 1
order by 1 desc),
tab2 as
(select year,total_cost,lag(total_cost,1)over (order by year)prev_cost
from tab1)

select year,total_cost,prev_cost,round(((total_cost-prev_cost)/prev_cost)*100,2)percentage_increase
from tab2
```

OUTPUT:



Row	year	total_cost	prev_cost	percentage_increase
1	2017	3669022.12	null	null
2	2018	8694733.84	3669022.12	136.98

ANALYSIS:

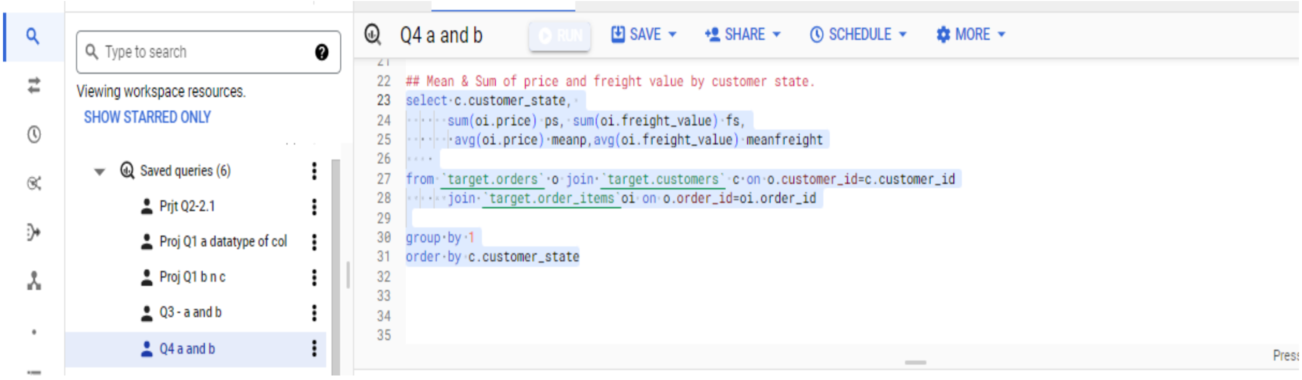
- ✓ Since 2016 data is not given, therefore the percentage increase in cost for year 2017 is null.
- ✓ There is an increase of 136.98% in the year 2018 in the cost from the previous year (2017).
- ✓ This helps us in understanding where the cost is being incurred and have the measures in place to control the same.

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

SQL QUERY:

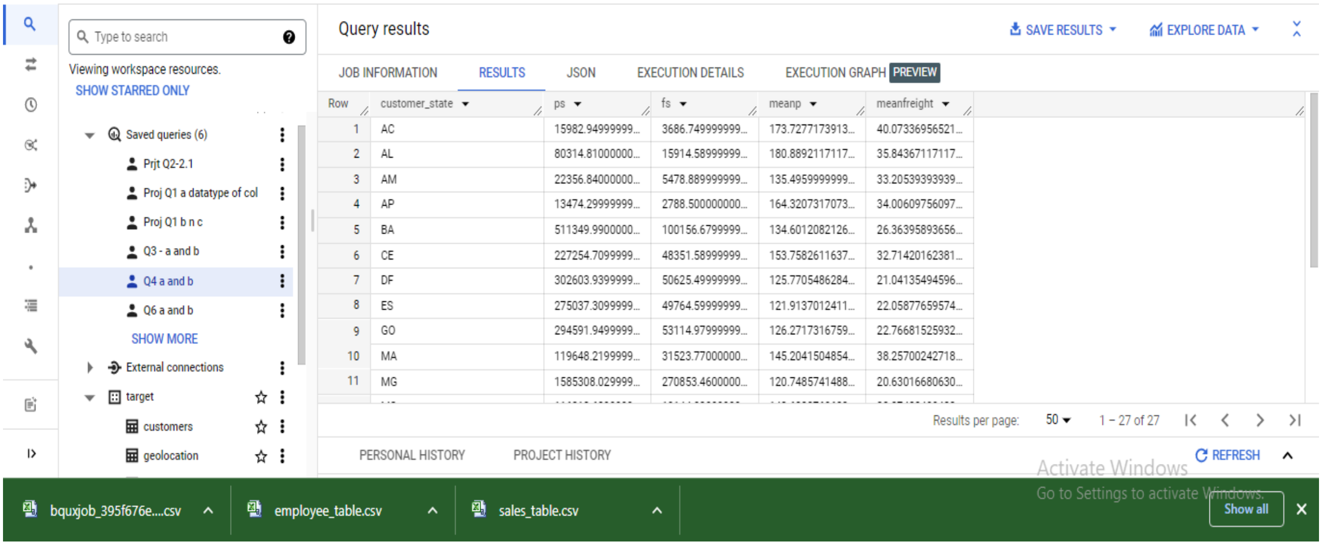
```
select c.customer_state,
      sum(oi.price) ps, sum(oi.freight_value) fs,
       avg(oi.price) meanp,avg(oi.freight_value) meanfreight

from `target.orders` o join `target.customers` c on o.customer_id=c.customer_id
      join `target.order_items`oi on o.order_id=oi.order_id
group by 1
order by c.customer_state
```

QUERY SCREENSHOT:



```
## Mean & Sum of price and freight value by customer state.
select c.customer_state, 
     sum(oi.price) ps, sum(oi.freight_value) fs,
      avg(oi.price) meanp,avg(oi.freight_value) meanfreight
    
from `target.orders` o join `target.customers` c on o.customer_id=c.customer_id
      join `target.order_items`oi on o.order_id=oi.order_id

group by 1
order by c.customer_state
```

OUTPUT:



Row	customer_state	ps	fs	meanp	meanfreight
1	AC	15982.94999999...	3686.749999999...	173.7277173913...	40.07336956521...
2	AL	80314.81000000...	15914.58999999...	180.8892117117...	35.84367117117...
3	AM	22356.84000000...	5478.889999999...	135.4959999999...	33.20539393939...
4	AP	13474.29999999...	2788.500000000...	164.3207317073...	34.00609756097...
5	BA	511349.9900000...	100156.6799999...	134.6012082126...	26.36395893656...
6	CE	227254.7099999...	48351.58999999...	153.7582611637...	32.71420162381...
7	DF	302603.9399999...	50625.49999999...	125.7705486284...	21.04135494596...
8	ES	275037.3099999...	49764.59999999...	121.9137012411...	22.05877659574...
9	GO	294591.9499999...	53114.97999999...	126.2717316759...	22.76681525932...
10	MA	119648.2199999...	31523.77000000...	145.2041504854...	38.25700242718...
11	MG	1585308.029999...	270853.4600000...	120.7485741488...	20.63016680630...

ANALYSIS:

- ✓ From the above output we get to know the sum of price as well as sum of freight according to the state.
- ✓ We also get to know the mean (avg) of price as well as freight.
- ✓ This helps us in understanding which state is incurring more freight charges and any another alternate mode of transport that can be used to control the same.
- ✓ This also helps us in understanding the actual price of the product in each state and understand the variation for the same.

==

Q.5. Analysis on sales, freight and delivery time:

1. **Calculate days between purchasing, delivering and estimated delivery.**
2. **Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:**
   - **time_to_delivery = order_delivered_customer_date-order_purchase_timestamp**
   - **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**
3. **Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery.**
4. **Sort the data to get the following:**
   - **Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5**
   - **Top 5 states with highest/lowest average time to delivery**
   - **Top 5 states where delivery is really fast/ not so fast compared to estimated date.**

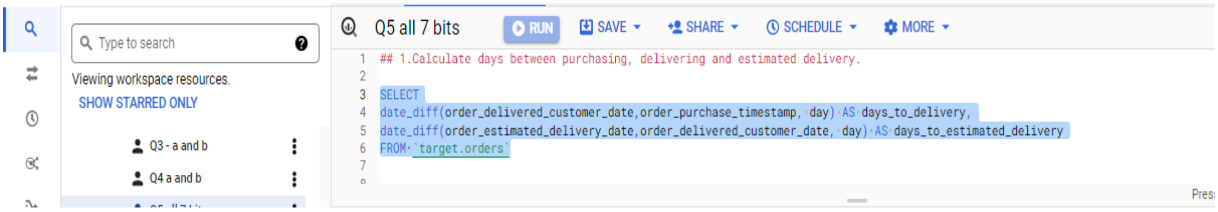
Solution:

5.1 Calculate days between purchasing, delivering and estimated delivery.

SQL QUERY:

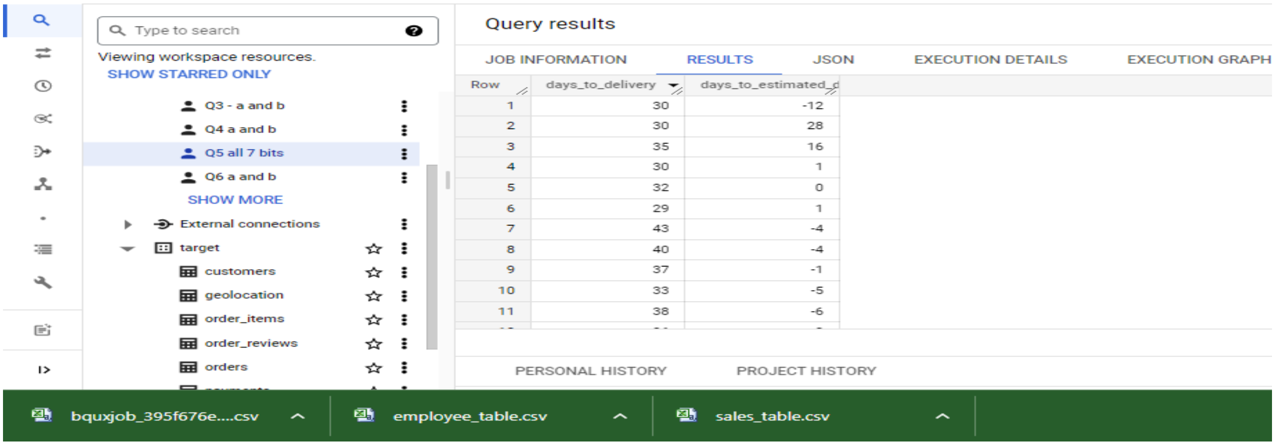
```
SELECT
date_diff(order_delivered_customer_date,order_purchase_timestamp, day) AS days_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) AS days_to_estimated_delivery
FROM `target.orders`
```

QUERY SCREENSHOT:



```
## 1.Calculate days between purchasing, delivering and estimated delivery.

SELECT
date_diff(order_delivered_customer_date,order_purchase_timestamp, day) AS days_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) AS days_to_estimated_delivery
FROM `target.orders`
```

OUTPUT:



Query results

Row	days_to_delivery	days_to_estimated_d
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

ANALYSIS:

- ✓ The above output helps us in understanding the actual number of days for delivery and the estimated days for delivery.
- ✓ This helps us to identify any delay that is occuring which might lead to customer dissatisfaction.
- ✓ One can take necessary measures to deliver the products on time while reducing the number of days to deliver.

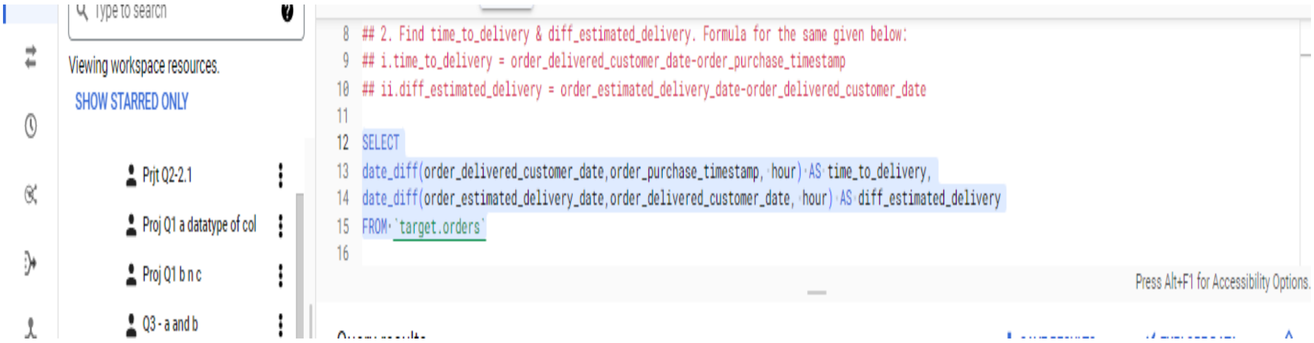
5.2 Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- **time_to_delivery = order_delivered_customer_date-order_purchase_timestamp**
- **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**

SQL QUERY:

```
SELECT
date_diff(order_delivered_customer_date,order_purchase_timestamp, hour) AS time_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, hour) AS diff_estimated_delivery
FROM `target.orders`
```

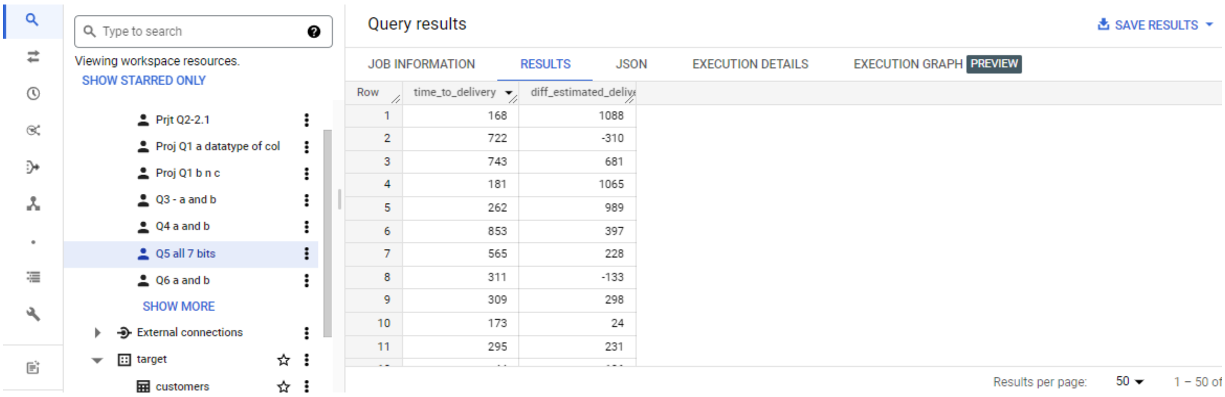
QUERY SCREENSHOT:



Type to search
Viewing workspace resources.
SHOW STARRED ONLY
Prjt Q2-2.1
Proj Q1 a datatype of col
Proj Q1 b n c
Q3 - a and b
Press Alt+F1 for Accessibility Options.

```
## 2. Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:
## i.time_to_delivery = order_delivered_customer_date-order_purchase_timestamp
## ii.diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

SELECT
date_diff(order_delivered_customer_date,order_purchase_timestamp, hour) AS time_to_delivery,
date_diff(order_estimated_delivery_date,order_delivered_customer_date, hour) AS diff_estimated_delivery
FROM `target.orders`
```

OUTPUT:



Row	time_to_delivery	diff_estimated_delive
1	168	1088
2	722	-310
3	743	681
4	181	1065
5	262	989
6	853	397
7	565	228
8	311	-133
9	309	298
10	173	24
11	295	231

ANALYSIS:

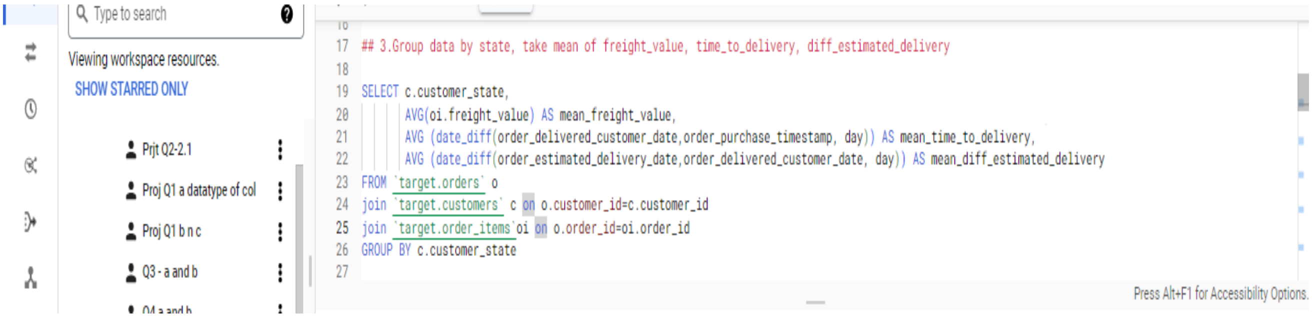
- ✓ The above output helps us in understanding the number of hours for delivery and the estimated hours for delivery.
- ✓ This helps us to identify any delay that is occurring which might lead to customer dissatisfaction.
- ✓ One can take necessary measures to deliver the products within few hours while gaining the customer satisfaction.

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

SQL QUERY:

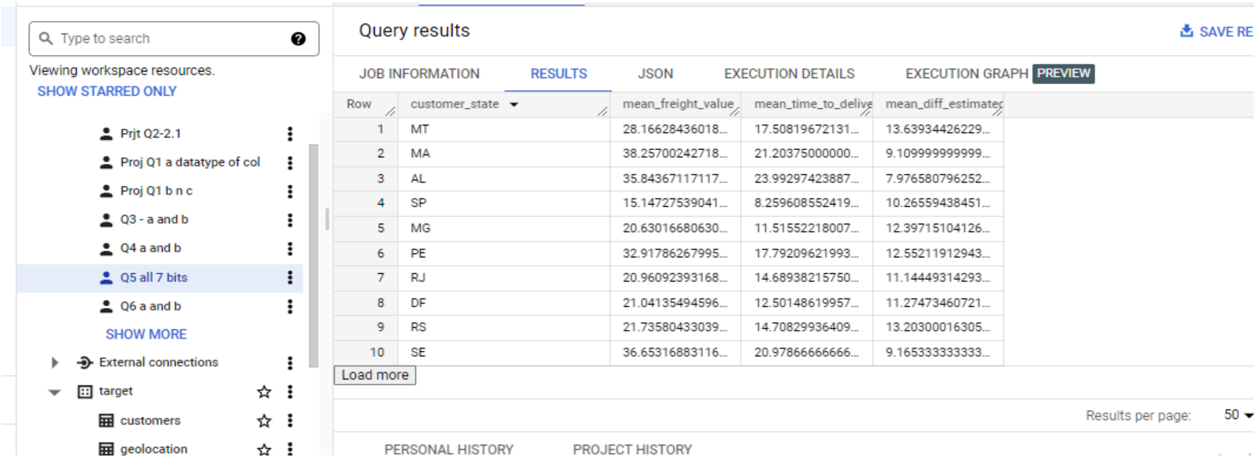
```
SELECT c.customer_state,
    AVG(oi.freight_value) AS mean_freight_value,
    AVG (date_diff(order_delivered_customer_date,order_purchase_timestamp, day)) AS mean_time_to_delivery,
    AVG (date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)) AS
mean_diff_estimated_delivery
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY c.customer_state
```

QUERY SCREENSHOT:


Type to search
Viewing workspace resources.
SHOW STARRED ONLY
Prjt Q2-2.1
Proj Q1 a datatype of col
Proj Q1 b n c
Q3 - a and b
Q4 a and b
Press Alt+F1 for Accessibility Options.

```
## 3.Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

SELECT c.customer_state,
      AVG(oi.freight_value) AS mean_freight_value,
      AVG (date_diff(order_delivered_customer_date,order_purchase_timestamp, day)) AS mean_time_to_delivery,
      AVG (date_diff(order_estimated_delivery_date,order_delivered_customer_date, day)) AS mean_diff_estimated_delivery
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY c.customer_state
```

OUTPUT:



Query results

Row	customer_state	mean_freight_value	mean_time_to_delive	mean_diff_estimatec
1	MT	28.16628436018...	17.50819672131...	13.63934426229...
2	MA	38.25700242718...	21.20375000000...	9.109999999999...
3	AL	35.84367117117...	23.99297423887...	7.976580796252...
4	SP	15.14727539041...	8.259608552419...	10.26559438451...
5	MG	20.63016680630...	11.51552218007...	12.39715104126...
6	PE	32.91786267995...	17.79209621993...	12.55211912943...
7	RJ	20.96092393168...	14.68938215750...	11.14449314293...
8	DF	21.04135494596...	12.50148619957...	11.27473460721...
9	RS	21.73580433039...	14.70829936409...	13.20300016305...
10	SE	36.65316883116...	20.97866666666...	9.165333333333...

ANALYSIS:

- ✓ The above output helps us to understand freight value, days to delivery and estimated number of days to deliver in different states.
- ✓ this helps us to understand which state is incurring more freight charges and is the state taking more number of days for delivery.
- ✓ This will help us in taking measures as to in which state we can make the delivery more fast by reducing the cost.

5.4 Sort the data to get the following:

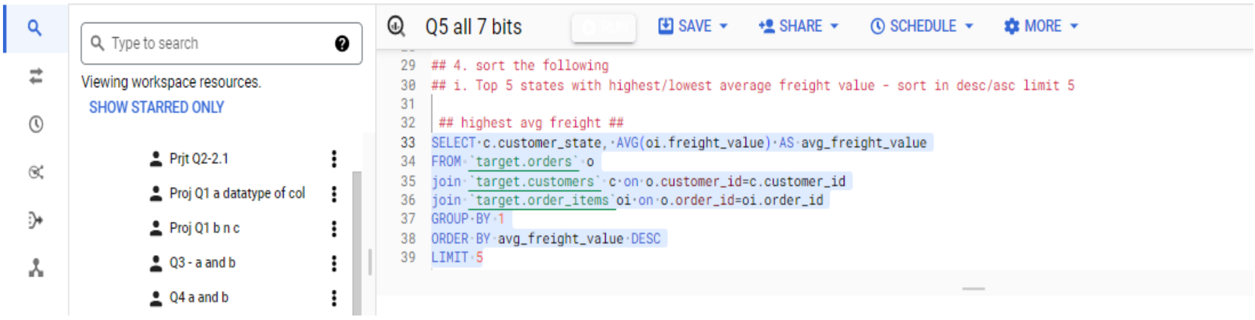
5.4.1 Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5.

SQL QUERY:

## highest avg freight ##

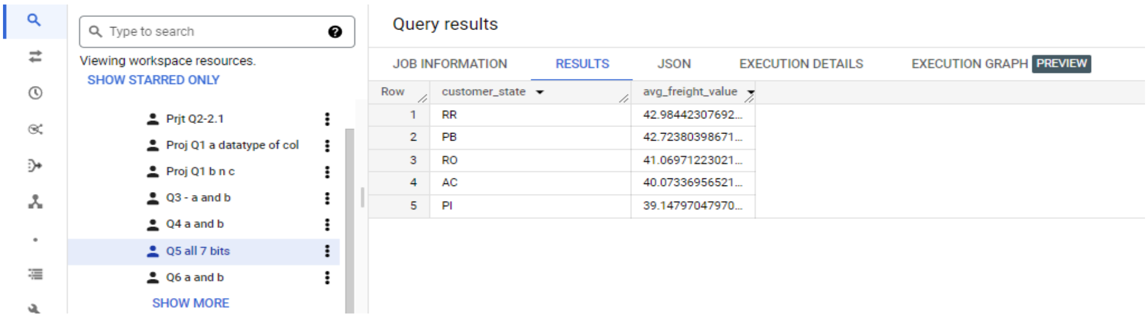
```
SELECT c.customer_state, AVG(oi.freight_value) AS avg_freight_value
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY 1
ORDER BY avg_freight_value DESC
LIMIT 5
```

QUERY SCREENSHOT:



```
## 4. sort the following
## i. Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

 ## highest avg freight ##
SELECT c.customer_state, AVG(oi.freight_value) AS avg_freight_value
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY 1
ORDER BY avg_freight_value DESC
LIMIT 5
```

OUTPUT:



Row	customer_state	avg_freight_value
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

ANALYSIS:

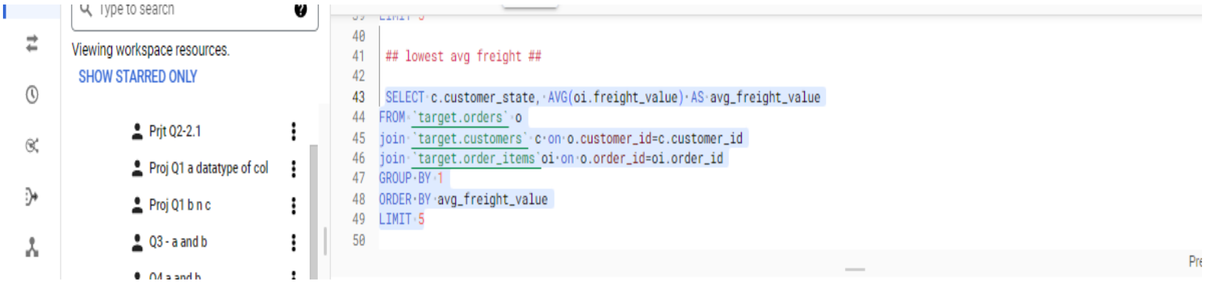
- ✓ With the above output we get the states with highest freight charges.
- ✓ With this, we can see if there is any other cost effective mode which helps in controlling the overall delivery cost.

SQL QUERY:

## lowest avg freight ##

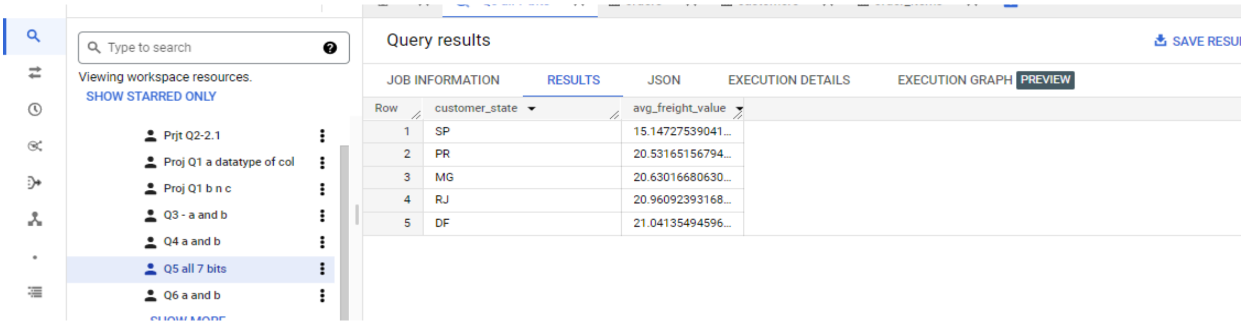
```
SELECT c.customer_state, AVG(oi.freight_value) AS avg_freight_value
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY 1
ORDER BY avg_freight_value
LIMIT 5
```

QUERY SCREENSHOT:



```
## lowest avg freight ##

SELECT c.customer_state, AVG(oi.freight_value) AS avg_freight_value
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY 1
ORDER BY avg_freight_value
LIMIT 5
```

OUTPUT:



Query results

Row	customer_state	avg_freight_value
1	SP	15.14727539041…
2	PR	20.53165156794…
3	MG	20.63016680630…
4	RJ	20.96092393168…
5	DF	21.04135494596…

ANALYSIS:

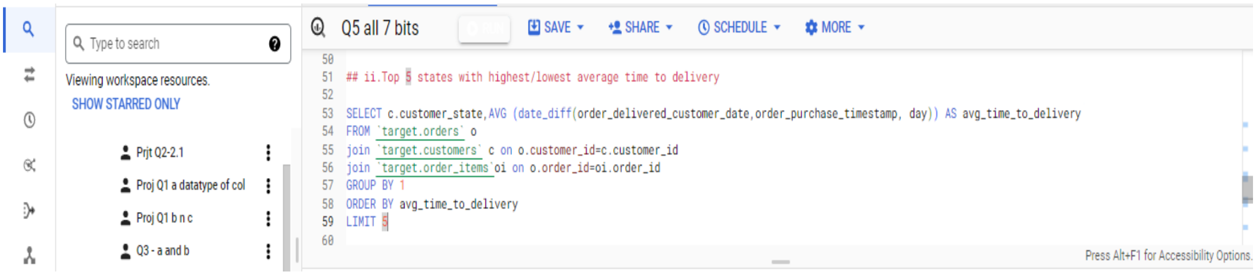
- ✓ With the above output we get the states with the lowest freight charges.
- ✓ For these states we can focus on other aspects of delivery like num of days, time of delivery and other to attain customer satisfaction.

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

SQL QUERY:

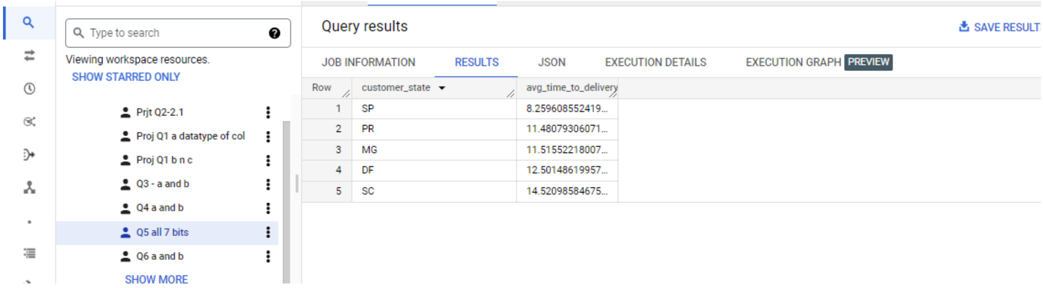
```
SELECT c.customer_state,AVG (date_diff(order_delivered_customer_date,order_purchase_timestamp, day)) AS
avg_time_to_delivery
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY 1
ORDER BY avg_time_to_delivery
LIMIT 5
```

QUERY SCREENSHOT:



```
## ii.Top 5 states with highest/lowest average time to delivery

SELECT c.customer_state,AVG (date_diff(order_delivered_customer_date,order_purchase_timestamp, day)) AS avg_time_to_delivery
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY 1
ORDER BY avg_time_to_delivery
LIMIT 5
```

OUTPUT:



Query results

Row	customer_state	avg_time_to_delivery
1	SP	8.259608552419...
2	PR	11.48079306071...
3	MG	11.51552218007...
4	DF	12.50148619957...
5	SC	14.52098584675...

ANALYSIS:

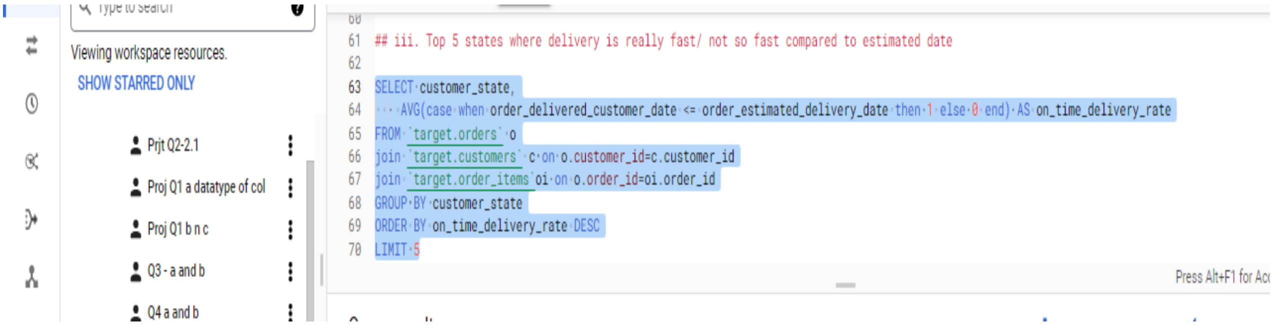
- ✓ This output helps us in understanding the states where the average time taken for delivery is the lowest.
- ✓ This helps to understand in which state the products are delivered in less time and gaining customer satisfaction.

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

SQL QUERY:

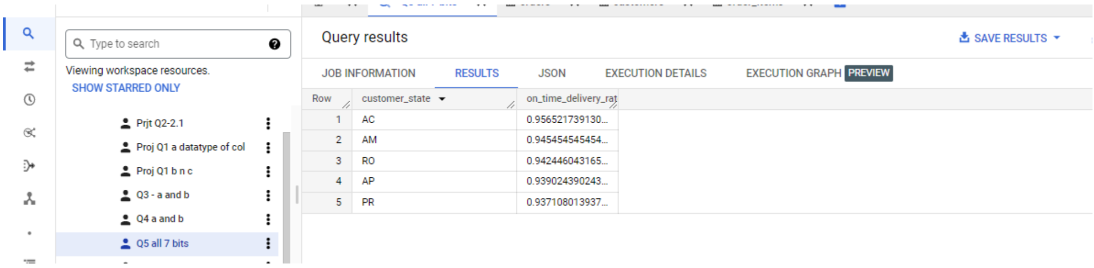
```
SELECT customer_state,
   AVG(case when order_delivered_customer_date <= order_estimated_delivery_date then 1 else 0 end) AS
on_time_delivery_rate
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY customer_state
ORDER BY on_time_delivery_rate DESC
LIMIT 5
```

QUERY SCREENSHOT:



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

SELECT customer_state,
   AVG(case when order_delivered_customer_date <= order_estimated_delivery_date then 1 else 0 end) AS on_time_delivery_rate
FROM `target.orders` o
join `target.customers` c on o.customer_id=c.customer_id
join `target.order_items`oi on o.order_id=oi.order_id
GROUP BY customer_state
ORDER BY on_time_delivery_rate DESC
LIMIT 5
```

OUTPUT:



Query results

Row	customer_state	on_time_delivery_rat
1	AC	0.956521739130...
2	AM	0.945454545454...
3	RO	0.942446043165...
4	AP	0.939024390243...
5	PR	0.937108013937...

ANALYSIS:

- ✓ This hels us in understanding in which state the orders are being delivered on time and how quickly there are being delivered.

Q.6. Payment type analysis:

i. Month over Month count of orders for different payment types.
ii. Count of orders based on the no. of payment instalments.

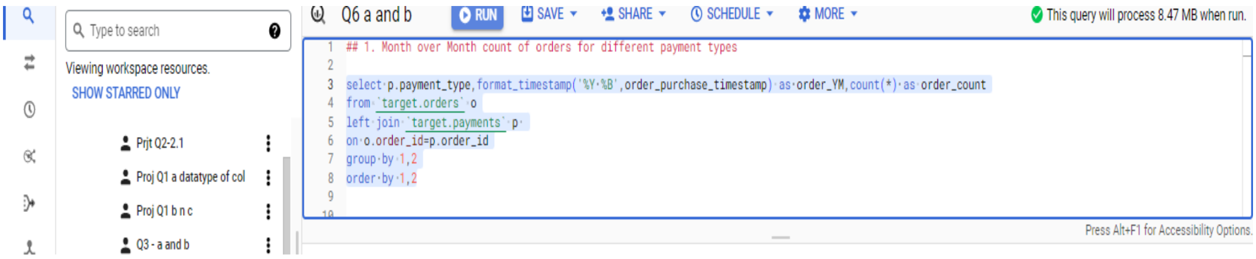
Solution:

6.1 Month over Month count of orders for different payment types

SQL QUERY:

```
select p.payment_type,format_timestamp('%Y %B',order_purchase_timestamp) as order_YM,count(*) as order_count
from `target.orders` o
left join `target.payments` p
on o.order_id=p.order_id
group by 1,2
order by 1,2
```

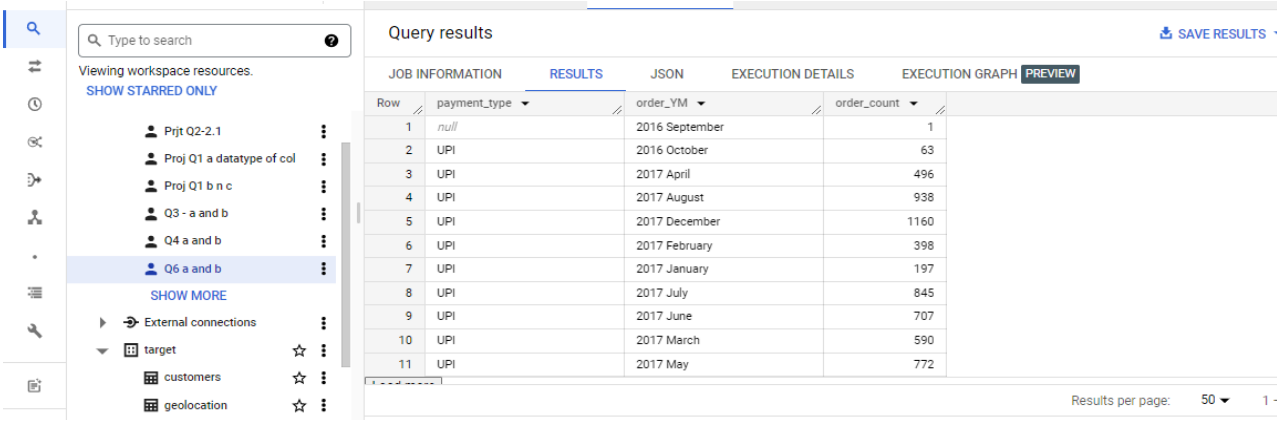
QUERY SCREENSHOT:



This query will process 8.47 MB when run.

```
## 1. Month over Month count of orders for different payment types

select p.payment_type,format_timestamp('%Y %B',order_purchase_timestamp) as order_YM,count(*) as order_count
from `target.orders` o
left join `target.payments` p
on o.order_id=p.order_id
group by 1,2
order by 1,2
```

OUTPUT:



Row	payment_type	order_YM	order_count
1	null	2016 September	1
2	UPI	2016 October	63
3	UPI	2017 April	496
4	UPI	2017 August	938
5	UPI	2017 December	1160
6	UPI	2017 February	398
7	UPI	2017 January	197
8	UPI	2017 July	845
9	UPI	2017 June	707
10	UPI	2017 March	590
11	UPI	2017 May	772

ANALYSIS:

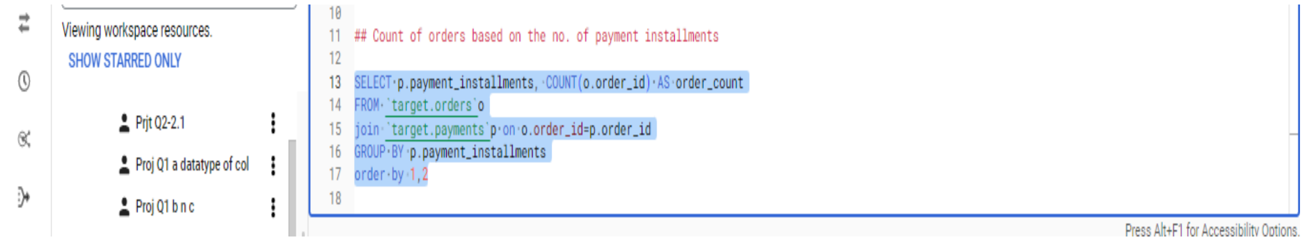
- ✓ The above output gives us an overview about the payment type for orders on the basis of the months.

6.2 Count of orders based on the no. of payment instalments.

SQL QUERY:

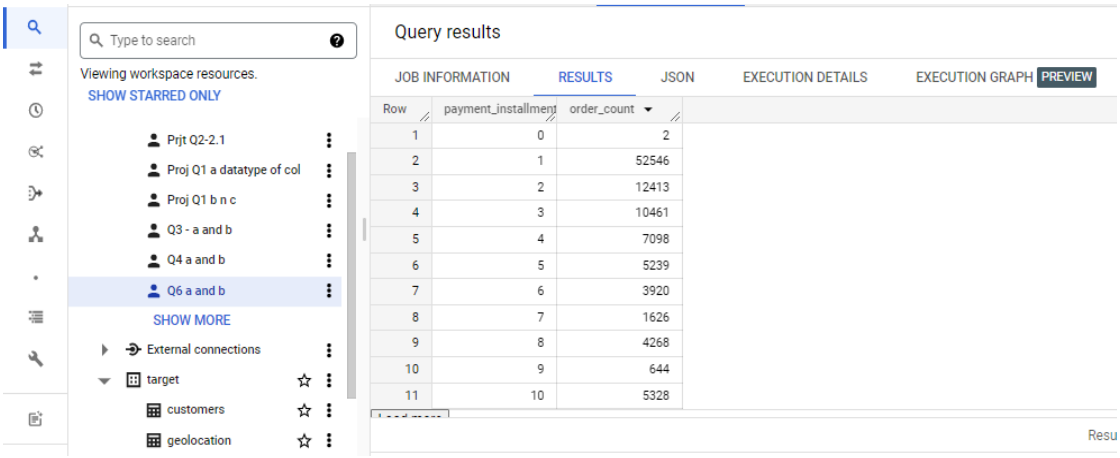
```
SELECT p.payment_installments, COUNT(o.order_id) AS order_count
FROM `target.orders`o
join `target.payments`p on o.order_id=p.order_id
GROUP BY p.payment_installments
order by 1,2
```

QUERY SCREENSHOT:


Viewing workspace resources.
SHOW STARRED ONLY
Prjt Q2-2.1
Proj Q1 a datatype of col
Proj Q1 b n c
Press Alt+F1 for Accessibility Options.

```
## Count of orders based on the no. of payment installments

SELECT p.payment_installments, COUNT(o.order_id) AS order_count
FROM `target.orders`o
join `target.payments`p on o.order_id=p.order_id
GROUP BY p.payment_installments
order by 1,2
```

OUTPUT:



Query results

Row	payment_installment	order_count
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644
11	10	5328

ANALYSIS:

- ✓ The above output gives us an understanding about the number of installemts by each order.
- ✓ This helps us in keeping the track of the payment yet to be received by the order and also if its being paid on time.
- ✓ If any default in payment of installmets also that can be tracked.

END