Target Business Use Case: Analysis Report

Author: Shibashis Bhattacharya.

Date: 12 Feb 2024.

Place: Kolkata, West Bengal, India.

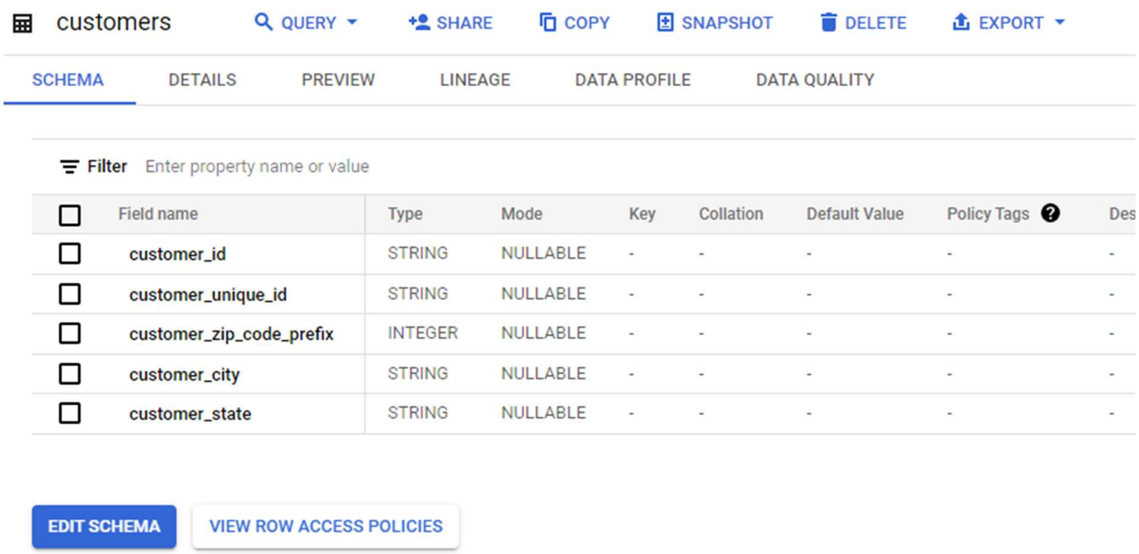
Q1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

1.1 – Data type of all columns in the "customers" table.

Analysis:

customers table is having five columns, out of which four columns (customer_id, customer_unique_id, customer_city, customer_state) are STRING type and one column (customer_zip_code_prefix) is of INTEGER type.

Big Query Screenshot:



□	Field name	Type	Mode	Key	Collation	Default Value	Policy Tags	Des
□	customer_id	STRING	NULLABLE	-	-	-	-	-
□	customer_unique_id	STRING	NULLABLE	-	-	-	-	-
□	customer_zip_code_prefix	INTEGER	NULLABLE	-	-	-	-	-
□	customer_city	STRING	NULLABLE	-	-	-	-	-
□	customer_state	STRING	NULLABLE	-	-	-	-	-

1.2 – Get the time range between which the orders were placed.

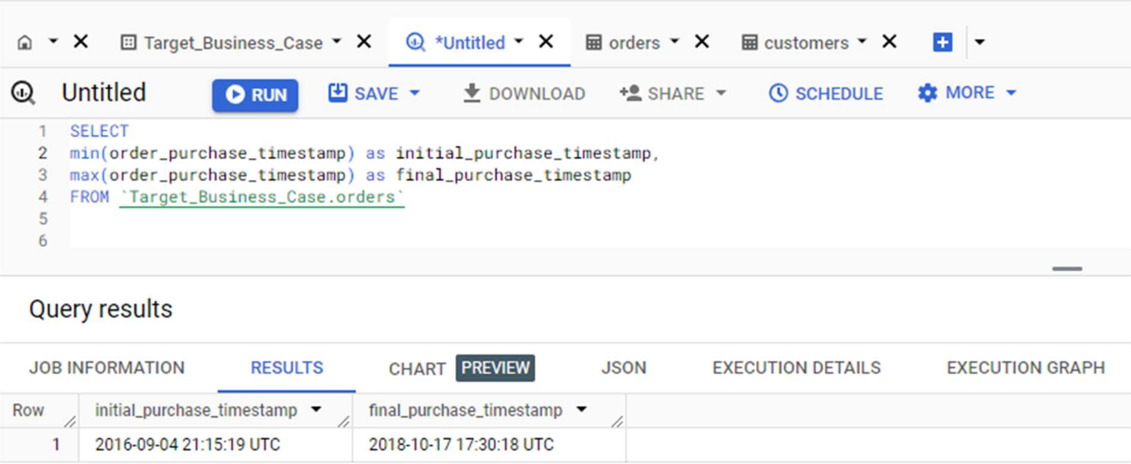
Analysis:

The orders table contains the order_purchase_timestamp column stating time of purchases. From this column we can derive the initial and the final timestamp of orders. Orders were placed between this time frame 2016-09-04 21:15:19 UTC and 2018-10-17 17:30:18 UTC.

Query:

```
SELECT
min(order_purchase_timestamp) as initial_purchase_timestamp,
max(order_purchase_timestamp) as final_purchase_timestamp
FROM `Target_Business_Case.orders`;
```

Big Query Screenshot:



```
SELECT
min(order_purchase_timestamp) as initial_purchase_timestamp,
max(order_purchase_timestamp) as final_purchase_timestamp
FROM `Target_Business_Case.orders`
```

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

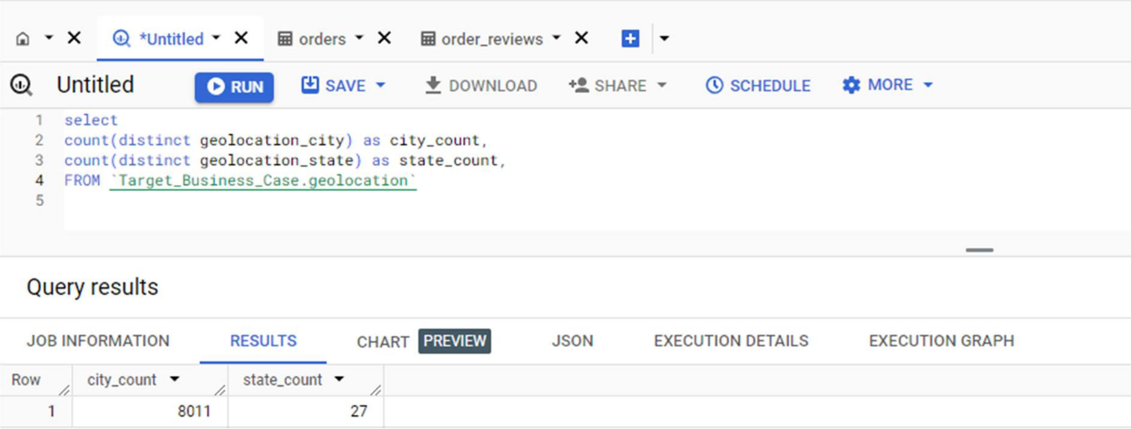
1.3 – Count the Cities & States of customers who ordered during the given period.

Analysis: There is total of 8011 unique cities and 27 unique states.

Query:

```
select
count(distinct geolocation_city) as city_count,
count(distinct geolocation_state) as state_count
FROM `Target_Business_Case.geolocation`;
```

Big Query Screenshot:



```
select
count(distinct geolocation_city) as city_count,
count(distinct geolocation_state) as state_count,
FROM `Target_Business_Case.geolocation`
```

Row	city_count	state_count
1	8011	27

Q2. In-depth Exploration:

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

Analysis:

We can clearly observe from the data that the orders have increased significantly from 2016 to 2017, it has kept on increasing but on a slower rate from 2017 to 2018.

In 2016, 329 orders were placed, in 2017, 45101 orders were placed, on 2018, 54011 orders were placed.

Further we dive in the month wise statistics for each year –

We understand that there was steady growth in orders placed on 10th month or last quarter of the 2016 year, followed by a lapse in the orders on 12th month of 2016.

Further the growth in a high pace is observed during the initial months or the first quarter of the 2017. This growth in the order continues throughout the second quarter (April, May, June months) as well in 2017.

3rd quarter of the 2017 year still got the increase in orders, there is a gradual increase in number during this period.

The mid of final quarter of the 2017 year saw orders reaching 7500+ units followed by a significant drop by 2000 unit on the December month of 2017.

During start of 2018 again, we observe a high rise in the orders placed crossing 7200+ units. This number attained on early 2018 January then slowly deteriorates over the months hitting 6100 around end of the second quarter that is June month. The initial couple of months for the third quarter of 2018 gets a gradual increase in order, though the orders further deteriorate to value as low as 16 on the September of 2018, further down to 4 orders in October of 2018.

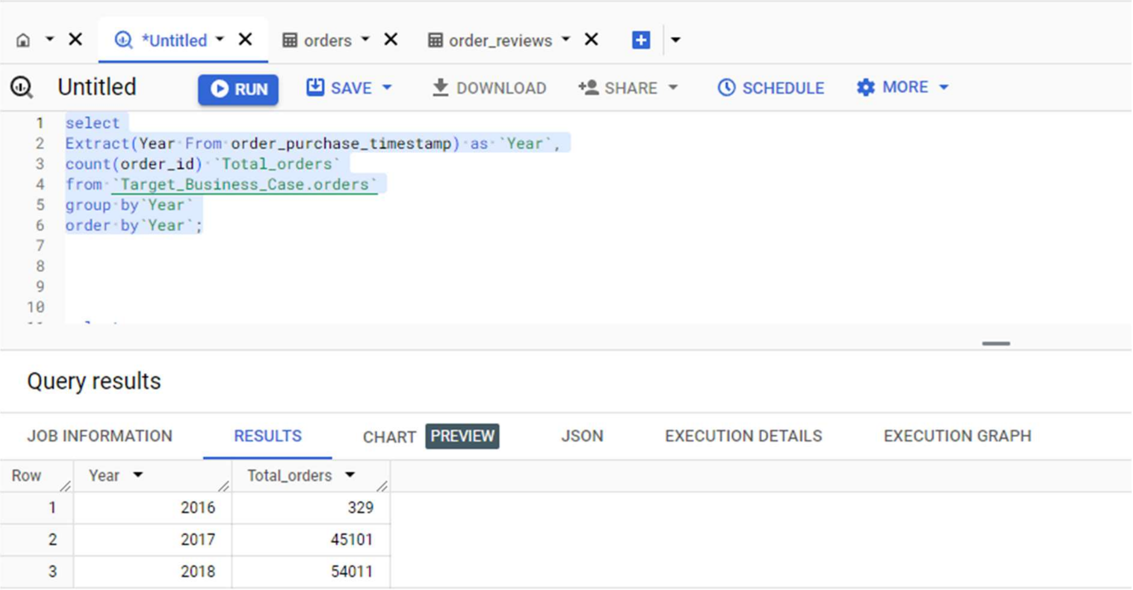
Insights/Recommendations:

During the mid-section of the years we observed fall in the order placed quantity, thus during these periods business need to come up with exiting offers to attract customers.

Query:

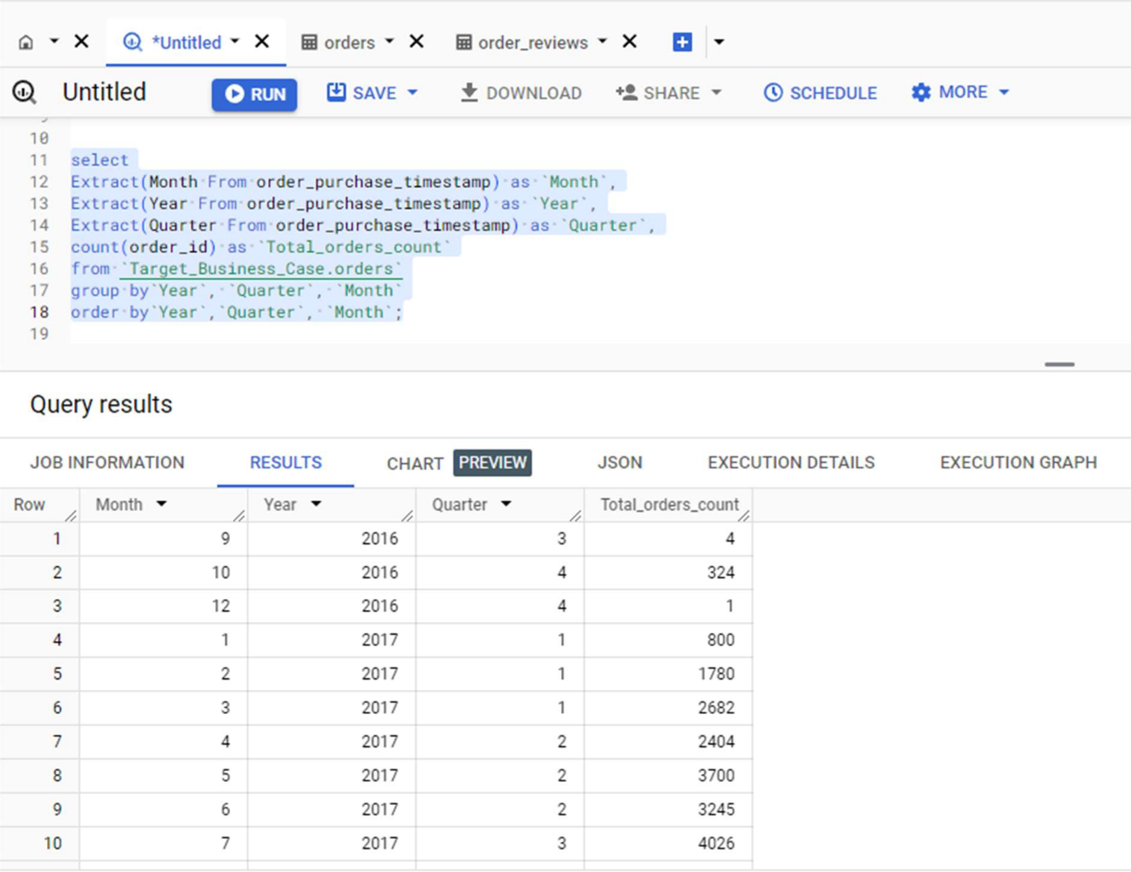
```
select
Extract(Year From order_purchase_timestamp) as `Year`,
count(order_id) `Total_orders`
from `Target_Business_Case.orders`
group by`Year`
order by`Year`;

select
Extract(Month From order_purchase_timestamp) as `Month`,
Extract(Year From order_purchase_timestamp) as `Year`,
Extract(Quarter From order_purchase_timestamp) as `Quarter`,
count(order_id) as `Total_orders_count`
from `Target_Business_Case.orders`
group by`Year`, `Quarter`, `Month`
order by`Year`,`Quarter`, `Month`;
```

Big Query Screenshot:



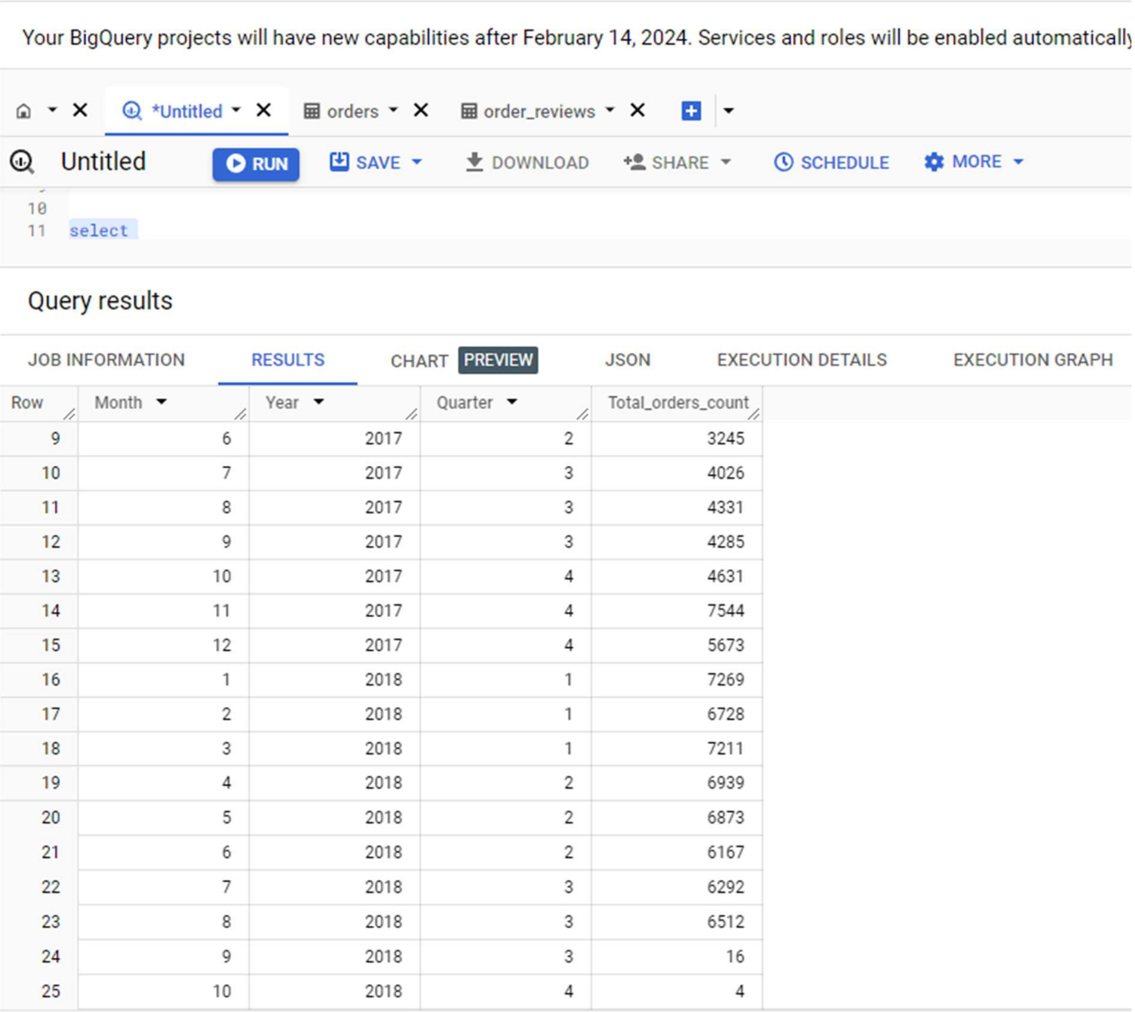
```
select
Extract(Year From order_purchase_timestamp) as `Year`,
count(order_id) `Total_orders`
from `Target_Business_Case.orders`
group by`Year`
order by`Year`;
```

Row	Year	Total_orders
1	2016	329
2	2017	45101
3	2018	54011



```sql
select
Extract(Month From order_purchase_timestamp) as `Month`,
Extract(Year From order_purchase_timestamp) as `Year`,
Extract(Quarter From order_purchase_timestamp) as `Quarter`,
count(order_id) as `Total_orders_count`
from `Target_Business_Case.orders`
group by`Year`, `Quarter`, `Month`
order by`Year`,`Quarter`, `Month`;
```

Row	Month	Year	Quarter	Total_orders_count
1	9	2016	3	4
2	10	2016	4	324
3	12	2016	4	1
4	1	2017	1	800
5	2	2017	1	1780
6	3	2017	1	2682
7	4	2017	2	2404
8	5	2017	2	3700
9	6	2017	2	3245
10	7	2017	3	4026



Your BigQuery projects will have new capabilities after February 14, 2024. Services and roles will be enabled automatically

```

select
```

Query results

Row	Month	Year	Quarter	Total_orders_count
9	6	2017	2	3245
10	7	2017	3	4026
11	8	2017	3	4331
12	9	2017	3	4285
13	10	2017	4	4631
14	11	2017	4	7544
15	12	2017	4	5673
16	1	2018	1	7269
17	2	2018	1	6728
18	3	2018	1	7211
19	4	2018	2	6939
20	5	2018	2	6873
21	6	2018	2	6167
22	7	2018	3	6292
23	8	2018	3	6512
24	9	2018	3	16
25	10	2018	4	4

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

Analysis:

When we arrange the data on monthly basis throughout the span from 2016 to 2018, we observe that order placed count peaks during the December of 2017 and start of the 2018 year. Throughout the initial quarter of the 2018 year, we observed a high quantity of order placed further the count deteriorates going forward on 2018.

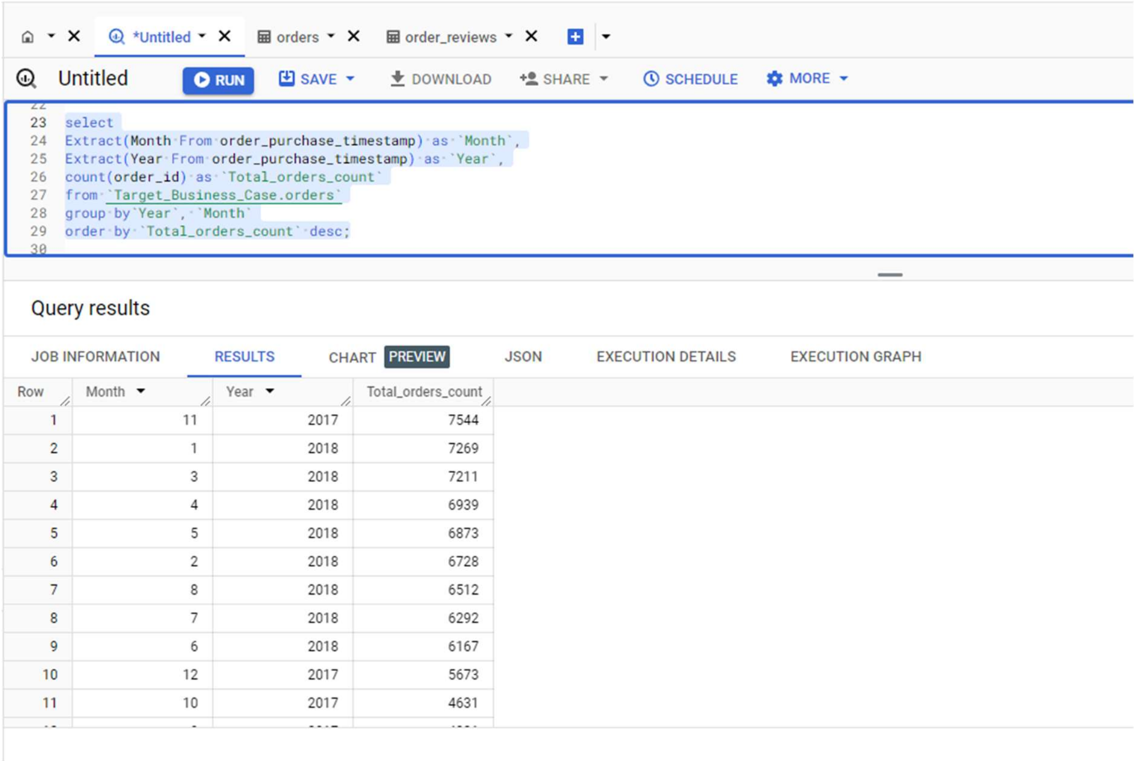
Insights/Recommendations:

Business should identify what is the exact cause of high order quantity during the vacation and festival season. Try to implement such nuances during the other parts of the year specially during the second quarters of the year to keep up the growth rate.

Query:

```
select
Extract(Month From order_purchase_timestamp) as `Month`,
Extract(Year From order_purchase_timestamp) as `Year`,
count(order_id) as `Total_orders_count`
from `Target_Business_Case.orders`
group by`Year`, `Month`
order by `Total_orders_count` desc;
```

Big Query Screenshot:



```
select
Extract(Month From order_purchase_timestamp) as `Month`,
Extract(Year From order_purchase_timestamp) as `Year`,
count(order_id) as `Total_orders_count`
from `Target_Business_Case.orders`
group by`Year`, `Month`
order by `Total_orders_count` desc;
```

Row	Month	Year	Total_orders_count
1	11	2017	7544
2	1	2018	7269
3	3	2018	7211
4	4	2018	6939
5	5	2018	6873
6	2	2018	6728
7	8	2018	6512
8	7	2018	6292
9	6	2018	6167
10	12	2017	5673
11	10	2017	4631

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

Analysis:

We know that the standard time in Brazil is 3 hours behind the UTC time, since the timestamps provided in the data are according to the UTC time, we will make the below assumptions:

Altering the timestamp value by reducing the hours by 3 hours.

On extracting the necessary set of data from the dataset we can identity that during the morning time, Brazilian customers mostly place their orders. Rest of the order traffic is high during Afternoon, followed by Night and then Dawn.

Mornings – 38291, Afternoon – 36986, Night – 14013, Dawn – 10151

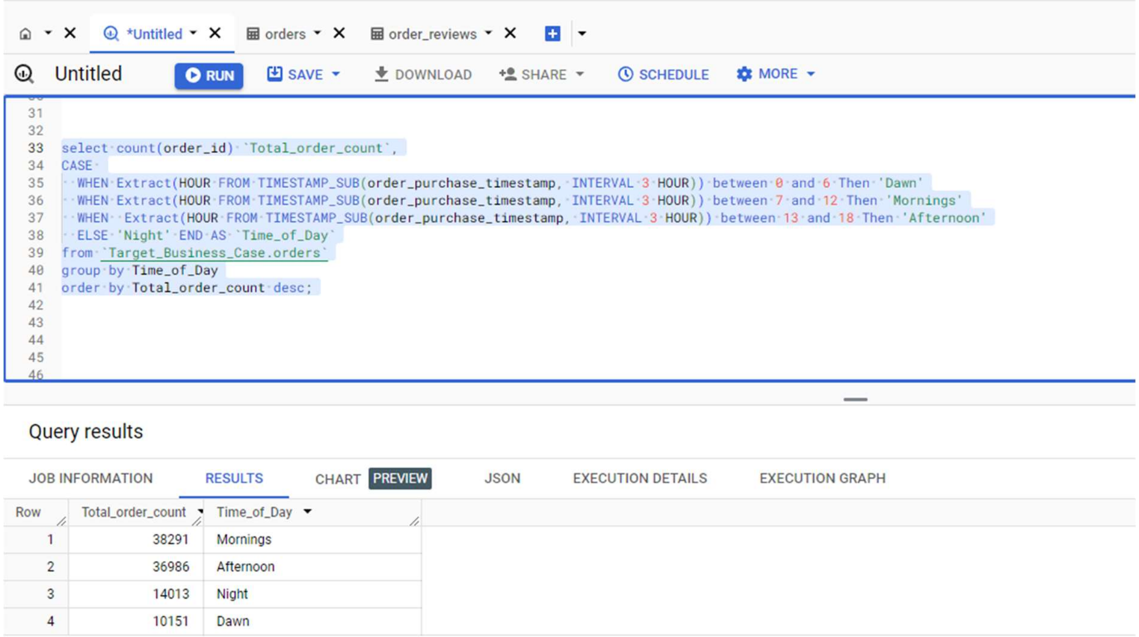
Insights/Recommendations:

Further we can draw from this outcome, that business can investigate time specific deal giving out to customers during the morning time which might be the cause of high order placed during that period of the day. Such kind of time specific offers or discounts in case provided during the other periods can provide opportunities for growth in quantity of order placed.

Query:

```
select count(order_id) `Total_order_count`,
CASE
  WHEN Extract(HOUR FROM TIMESTAMP_SUB(order_purchase_timestamp, INTERVAL 3 HOUR))
between 0 and 6 Then 'Dawn'
  WHEN Extract(HOUR FROM TIMESTAMP_SUB(order_purchase_timestamp, INTERVAL 3 HOUR))
between 7 and 12 Then 'Mornings'
  WHEN  Extract(HOUR FROM TIMESTAMP_SUB(order_purchase_timestamp, INTERVAL 3 HOUR))
between 13 and 18 Then 'Afternoon'
  ELSE 'Night' END AS `Time_of_Day`
from `Target_Business_Case.orders`
group by Time_of_Day
order by Total_order_count desc;
```

Big Query Screenshot:



```
select count(order_id) `Total_order_count`,
CASE
  WHEN Extract(HOUR FROM TIMESTAMP_SUB(order_purchase_timestamp, INTERVAL 3 HOUR)) between 0 and 6 Then 'Dawn'
  WHEN Extract(HOUR FROM TIMESTAMP_SUB(order_purchase_timestamp, INTERVAL 3 HOUR)) between 7 and 12 Then 'Mornings'
  WHEN  Extract(HOUR FROM TIMESTAMP_SUB(order_purchase_timestamp, INTERVAL 3 HOUR)) between 13 and 18 Then 'Afternoon'
  ELSE 'Night' END AS `Time_of_Day`
from `Target_Business_Case.orders`
group by Time_of_Day
order by Total_order_count desc;
```

Query results

Row	Total_order_count	Time_of_Day
1	38291	Mornings
2	36986	Afternoon
3	14013	Night
4	10151	Dawn

Q3. Evolution of E-commerce orders in the Brazil region:

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

Analysis:

From the month on month over each state's order we can clearly understand that the state of SP has placed the greatest number of orders throughout all the months.

The states of RR, AP, AC, AM are some of the states who has placed least number of orders throughout the months.

States like RJ and MG have placed more orders during the months of April to September thus the second and third quarter of the year.

Insights/Recommendations:

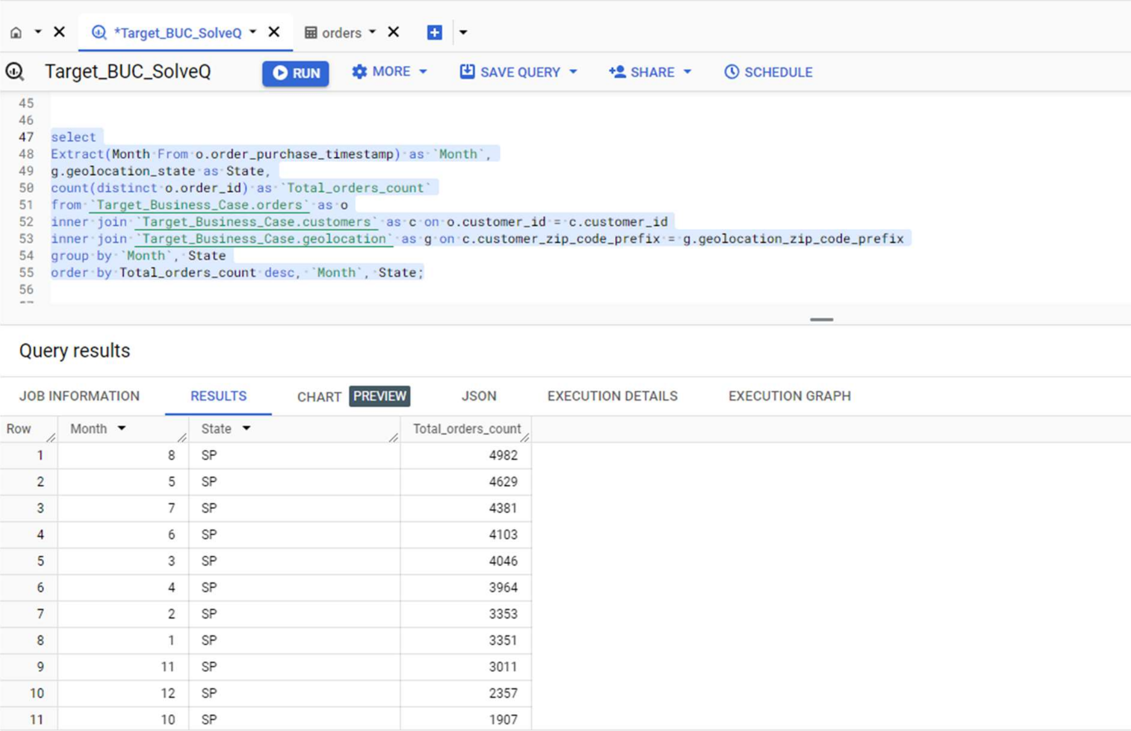
Business should take interest on checking the specific for the states appearing on top of the category to replicate the same for other states.

During second and third quarters of the year certain sale window should be going on in states with higher order quantity, which can be replicated on other states to improve their condition.

Query:

```sql
select
Extract(Month From o.order_purchase_timestamp) as `Month`,
g.geolocation_state as State,
count(distinct o.order_id) as `Total_orders_count`
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.customers` as c on o.customer_id = c.customer_id
inner join `Target_Business_Case.geolocation` as g on c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
group by `Month`, State
order by Total_orders_count desc, `Month`, State;
```

Big Query Screenshot:



```
select
Extract(Month From o.order_purchase_timestamp) as `Month`,
g.geolocation_state as State,
count(distinct o.order_id) as `Total_orders_count`
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.customers` as c on o.customer_id = c.customer_id
inner join `Target_Business_Case.geolocation` as g on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
group by `Month`, State
order by Total_orders_count desc, `Month`, State;
```

Query results

Row	Month	State	Total_orders_count
1	8	SP	4982
2	5	SP	4629
3	7	SP	4381
4	6	SP	4103
5	3	SP	4046
6	4	SP	3964
7	2	SP	3353
8	1	SP	3351
9	11	SP	3011
10	12	SP	2357
11	10	SP	1907

3.2 – How are the customers distributed across all the states?

Analysis:

The state of SP is having the greatest number of customers 41731, then the state of RJ with 12839 customers, followed by the state of MG with 11624 customers are the top three states based on unique customer count. Rest of the states are all having customers less than 5500.

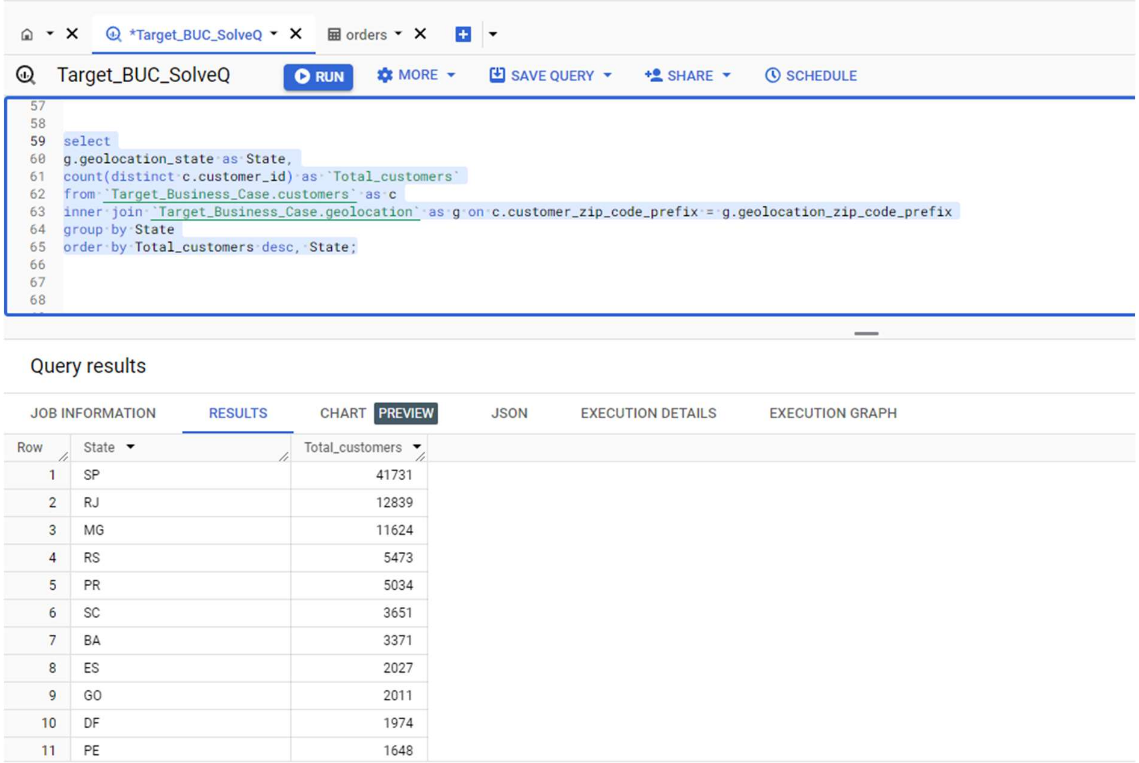
Insights/Recommendations:

The count of customers may vary depending on several factors such as population of the region, availability of internet, devices, economic conditions. Business should focus on various methods of marketing both in offline and online fashion to attract customers from all types.

Query:

```
select
g.geolocation_state as State,
count(distinct c.customer_id) as `Total_customers`
from `Target_Business_Case.customers` as c
inner join `Target_Business_Case.geolocation` as g on c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
group by State
order by Total_customers desc, State;
```

Big Query Screenshot:



```
select
g.geolocation_state as State,
count(distinct c.customer_id) as `Total_customers`
from `Target_Business_Case.customers` as c
inner join `Target_Business_Case.geolocation` as g on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
group by State
order by Total_customers desc, State;
```

Row	State	Total_customers
1	SP	41731
2	RJ	12839
3	MG	11624
4	RS	5473
5	PR	5034
6	SC	3651
7	BA	3371
8	ES	2027
9	GO	2011
10	DF	1974
11	PE	1648

Q4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

4.1 – Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only). You can use the "payment_value" column in the payments table to get the cost of orders.

Analysis:

On calculating the percentage increase based on the payment_value column from the payments table, we observed that there was a significant increase of 174.55 % in the order payment value between January and August of 2017 and 2018.

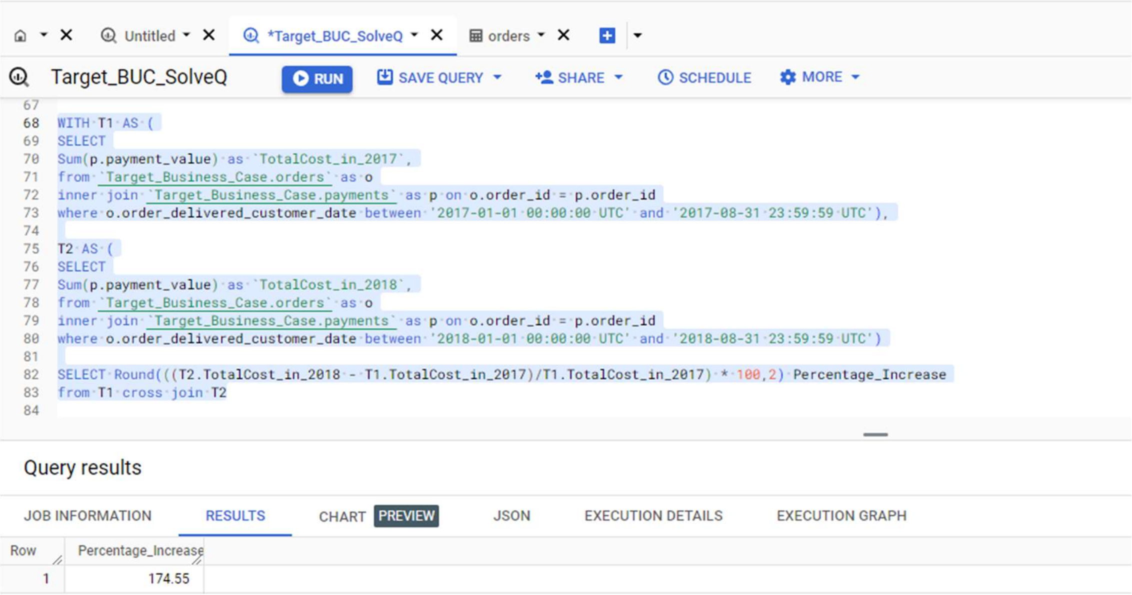
Insights/Recommendations:

An almost 200% increase in the valuation of the orders signify that business is making notable growth, ongoing processes and operations adhering to which is helping to sustain this kind of results should be documented thoroughly. Such documentations should provide support in future actions.

Query:

```
WITH T1 AS (
SELECT
Sum(p.payment_value) as `TotalCost_in_2017`,
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.payments` as p on o.order_id = p.order_id
where o.order_delivered_customer_date between '2017-01-01 00:00:00 UTC' and '2017-
08-31 23:59:59 UTC'),

T2 AS (
SELECT
Sum(p.payment_value) as `TotalCost_in_2018`,
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.payments` as p on o.order_id = p.order_id
where o.order_delivered_customer_date between '2018-01-01 00:00:00 UTC' and '2018-
08-31 23:59:59 UTC')

SELECT Round(((T2.TotalCost_in_2018 - T1.TotalCost_in_2017)/T1.TotalCost_in_2017) *
100,2) Percentage_Increase
from T1 cross join T2
```

Big Query Screenshot:



```
WITH T1 AS (
SELECT
Sum(p.payment_value) as `TotalCost_in_2017`,
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.payments` as p on o.order_id = p.order_id
where o.order_delivered_customer_date between '2017-01-01 00:00:00 UTC' and '2017-08-31 23:59:59 UTC'),

T2 AS (
SELECT
Sum(p.payment_value) as `TotalCost_in_2018`,
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.payments` as p on o.order_id = p.order_id
where o.order_delivered_customer_date between '2018-01-01 00:00:00 UTC' and '2018-08-31 23:59:59 UTC')

SELECT Round(((T2.TotalCost_in_2018 - T1.TotalCost_in_2017)/T1.TotalCost_in_2017) * 100,2) Percentage_Increase
from T1 cross join T2
```

Query results

Row	Percentage_Increase
1	174.55

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

Analysis:

On calculating the average price and total price and sorting the outcome with the average price for each state from high to low we observe that states like PB, BA, AM, RO, AC are the top five states having highest average price of orders.

While sorting the outcome with the total price for each state from high to low we observe states like SP, PR, MG, RJ, SC are the top five states having the highest total price, their average price is on the lower end of the spectrum, from this we can also derive that these states are experiencing high quantity of orders as well.

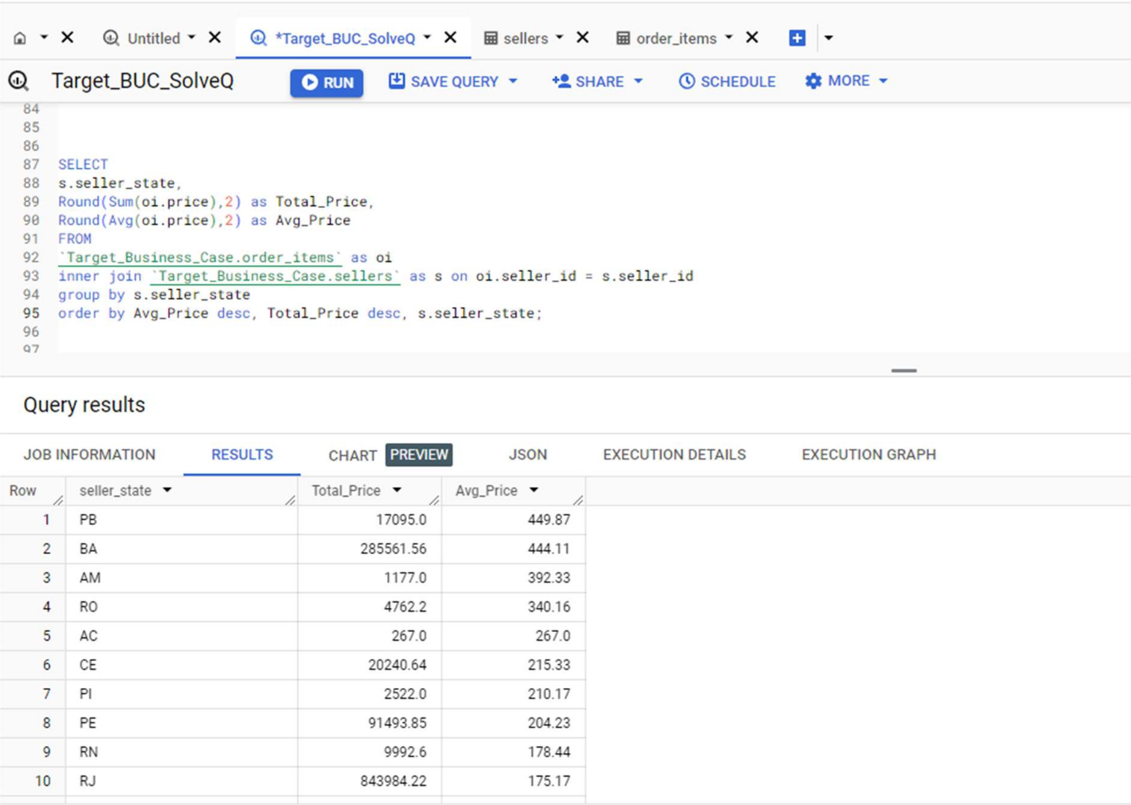
Insights/Recommendations:

Business should consider monitoring the leaderboards to keep up the growth. Should focus on the rest of the states for better marketing to make more profits.

Query:

```
SELECT
s.seller_state,
Round(Sum(oi.price),2) as Total_Price,
Round(Avg(oi.price),2) as Avg_Price
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Avg_Price desc, Total_Price desc, s.seller_state;
```

Big Query Screenshot:



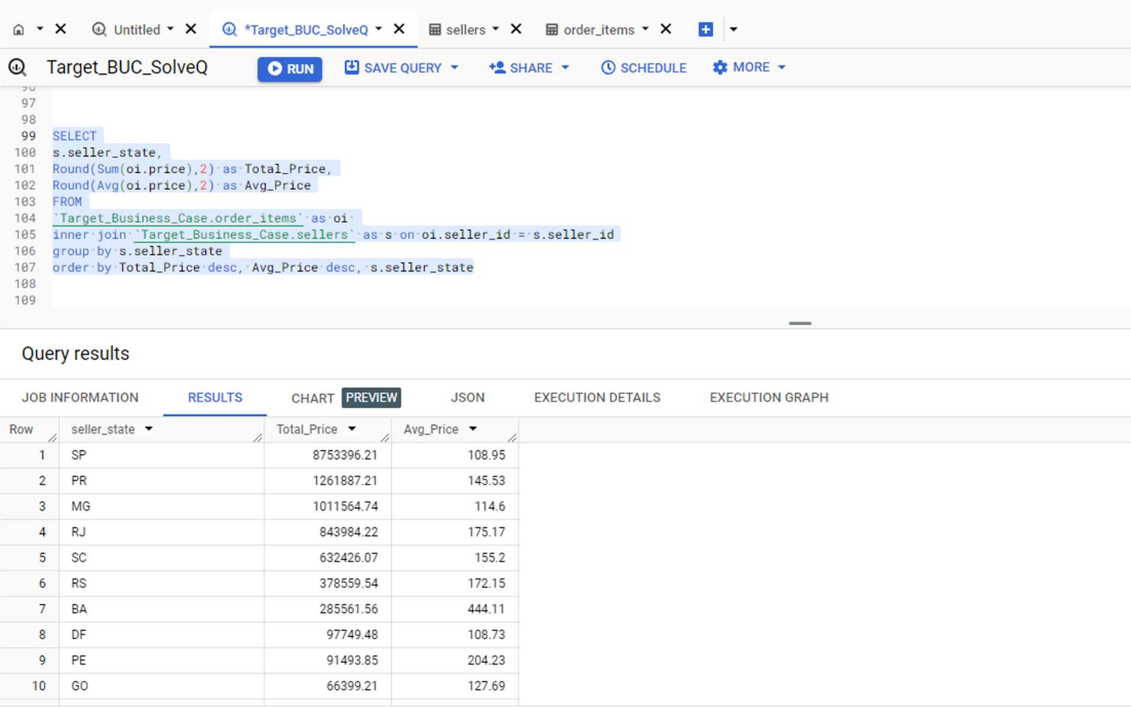
```
SELECT
s.seller_state,
Round(Sum(oi.price),2) as Total_Price,
Round(Avg(oi.price),2) as Avg_Price
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Avg_Price desc, Total_Price desc, s.seller_state;
```

Row	seller_state	Total_Price	Avg_Price
1	PB	17095.0	449.87
2	BA	285561.56	444.11
3	AM	1177.0	392.33
4	RO	4762.2	340.16
5	AC	267.0	267.0
6	CE	20240.64	215.33
7	PI	2522.0	210.17
8	PE	91493.85	204.23
9	RN	9992.6	178.44
10	RJ	843984.22	175.17

Query:

```
SELECT
s.seller_state,
Round(Sum(oi.price),2) as Total_Price,
Round(Avg(oi.price),2) as Avg_Price
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Total_Price desc, Avg_Price desc, s.seller_state
```

Big Query Screenshot:



```
SELECT
s.seller_state,
Round(Sum(oi.price),2) as Total_Price,
Round(Avg(oi.price),2) as Avg_Price
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Total_Price desc, Avg_Price desc, s.seller_state
```

Query results

Row	seller_state	Total_Price	Avg_Price
1	SP	8753396.21	108.95
2	PR	1261887.21	145.53
3	MG	1011564.74	114.6
4	RJ	843984.22	175.17
5	SC	632426.07	155.2
6	RS	378559.54	172.15
7	BA	285561.56	444.11
8	DF	97749.48	108.73
9	PE	91493.85	204.23
10	GO	66399.21	127.69

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

Analysis:

On calculating the average freight value and total freight value and sorting the outcome with the average freight value for each state from high to low we observe that states like RO, CE, PB, PI, AC are the top five states having highest average freight value of orders.

While sorting the outcome with the total freight value for each state from high to low we observe states like SP, MG, PR, SC, RJ are the top five states having the highest total freight value. Thus again signifying that the quantity of orders consumed by these states are among the highest.

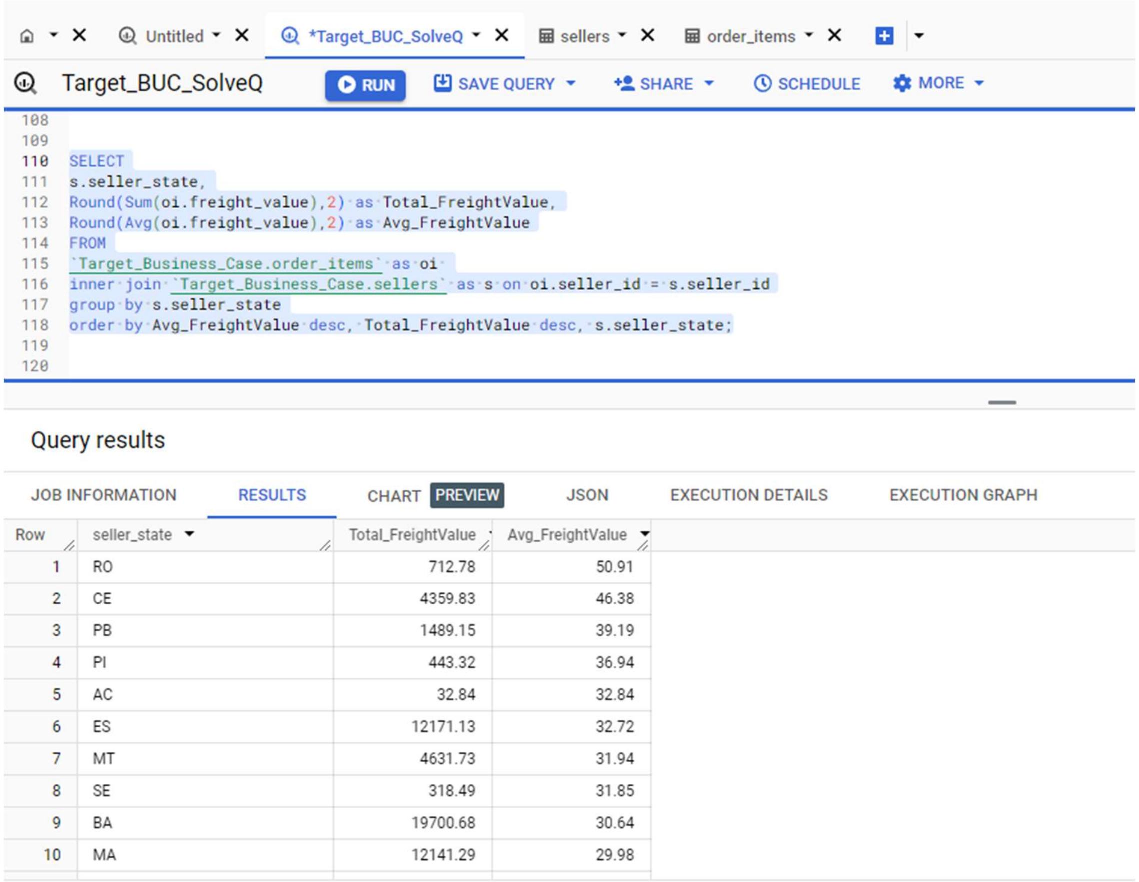
Insights/Recommendations:

Business should consider monitoring the leaderboards to keep up the stability of these states managing higher load of orders. Should focus on these states to undergo survey and understand if there is requirement in additional human resources. Further make notable observations to other states to facilitate operations involving higher quantity of order transactions.

Query:

```
SELECT
s.seller_state,
Round(Sum(oi.freight_value),2) as Total_FreightValue,
Round(Avg(oi.freight_value),2) as Avg_FreightValue
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Avg_FreightValue desc, Total_FreightValue desc, s.seller_state;
```

Big Query Screenshot:



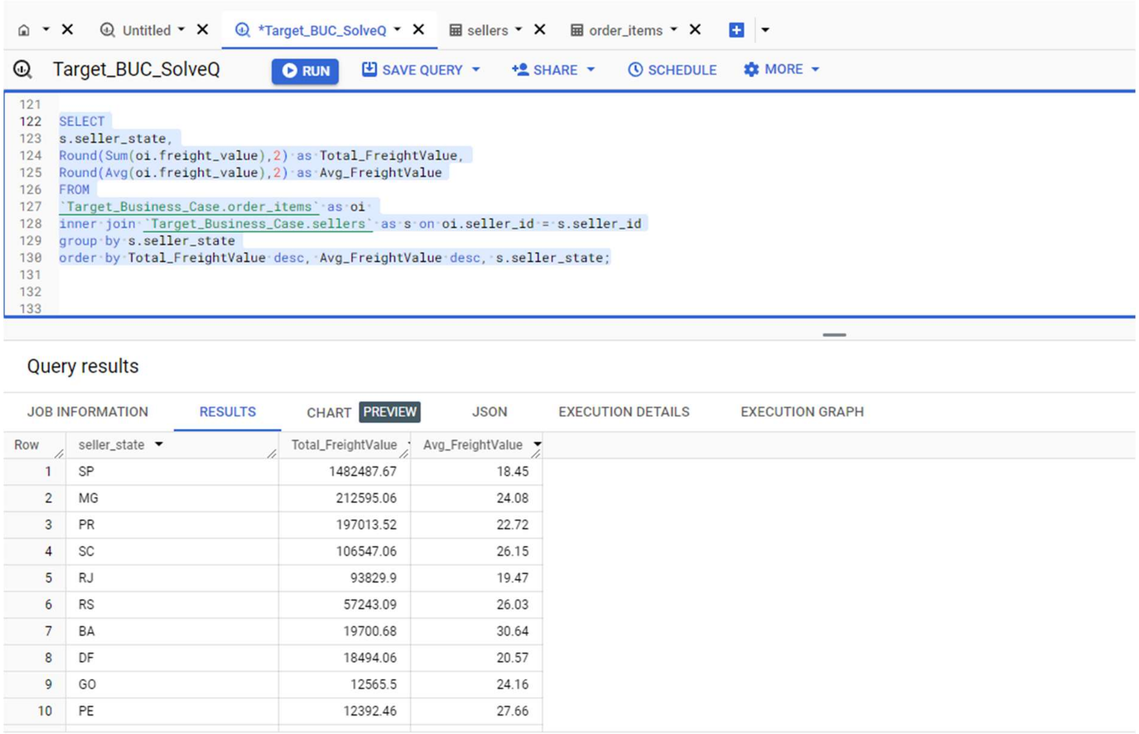
```
SELECT
s.seller_state,
Round(Sum(oi.freight_value),2) as Total_FreightValue,
Round(Avg(oi.freight_value),2) as Avg_FreightValue
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Avg_FreightValue desc, Total_FreightValue desc, s.seller_state;
```

Row	seller_state	Total_FreightValue	Avg_FreightValue
1	RO	712.78	50.91
2	CE	4359.83	46.38
3	PB	1489.15	39.19
4	PI	443.32	36.94
5	AC	32.84	32.84
6	ES	12171.13	32.72
7	MT	4631.73	31.94
8	SE	318.49	31.85
9	BA	19700.68	30.64
10	MA	12141.29	29.98

Query:

```
SELECT
s.seller_state,
Round(Sum(oi.freight_value),2) as Total_FreightValue,
Round(Avg(oi.freight_value),2) as Avg_FreightValue
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Total_FreightValue desc, Avg_FreightValue desc, s.seller_state;
```

Big Query Screenshot:



```
SELECT
s.seller_state,
Round(Sum(oi.freight_value),2) as Total_FreightValue,
Round(Avg(oi.freight_value),2) as Avg_FreightValue
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by s.seller_state
order by Total_FreightValue desc, Avg_FreightValue desc, s.seller_state;
```

Query results

Row	seller_state	Total_FreightValue	Avg_FreightValue
1	SP	1482487.67	18.45
2	MG	212595.06	24.08
3	PR	197013.52	22.72
4	SC	106547.06	26.15
5	RJ	93829.9	19.47
6	RS	57243.09	26.03
7	BA	19700.68	30.64
8	DF	18494.06	20.57
9	GO	12565.5	24.16
10	PE	12392.46	27.66

Q5: Analysis based on sales, freight and delivery time.

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.

Do this in a single query.

You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

time_to_deliver = order_delivered_customer_date - order_purchase_timestamp

diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

Analysis:

The calculations on time to deliver and difference estimated delivery shows us that maximum time to deliver some orders are even up to 209 day. We can also see from the data that the difference between the estimated date and actual date of delivery to customer has also deviate to 188 days at max.

Also, we can observe that multiple orders were delivered before the estimated date, even up to 27 days before the estimated date of delivery.

Certain orders were placed and delivered on the same day itself.

Insights/Recommendations:

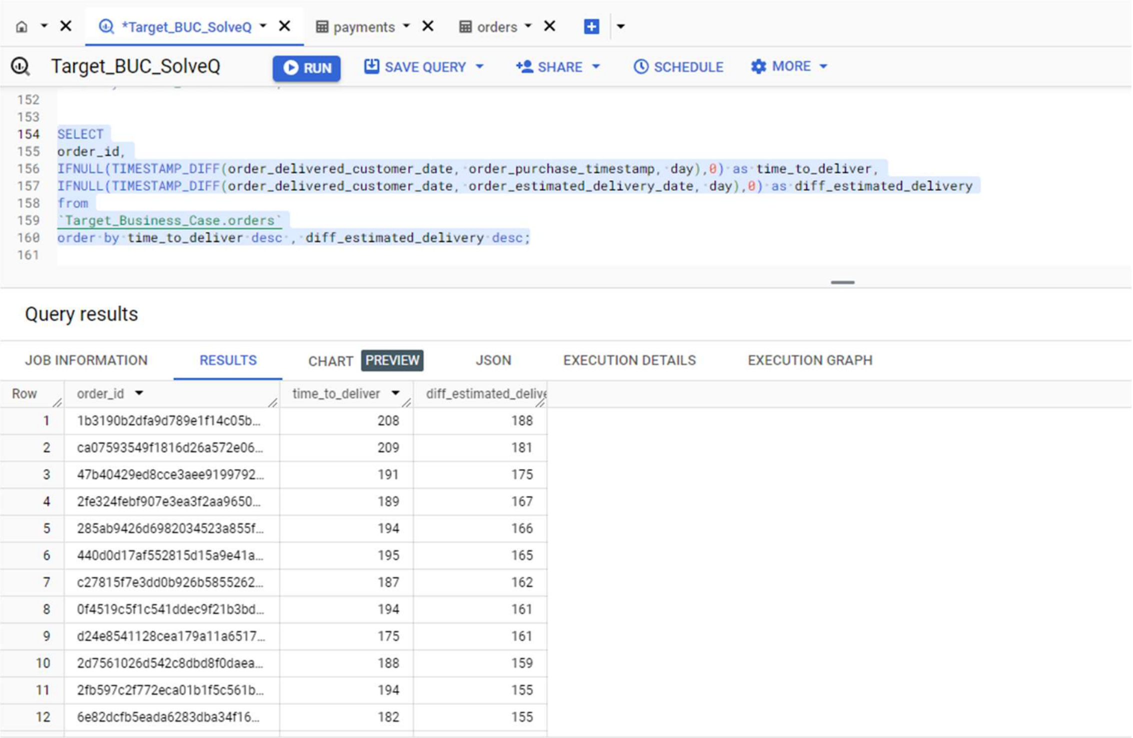
Numbers on the time taken to delivery order on certain cases can be observed on higher end towards 200, business can make actions to take collect feedback on such orders to know actual causes and make improvements on basis of that.

Query:

```
SELECT
order_id,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
day),0) as time_to_deliver,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date,
day),0) as diff_estimated_delivery
from
`Target_Business_Case.orders`
order by time_to_deliver desc , diff_estimated_delivery desc;
```

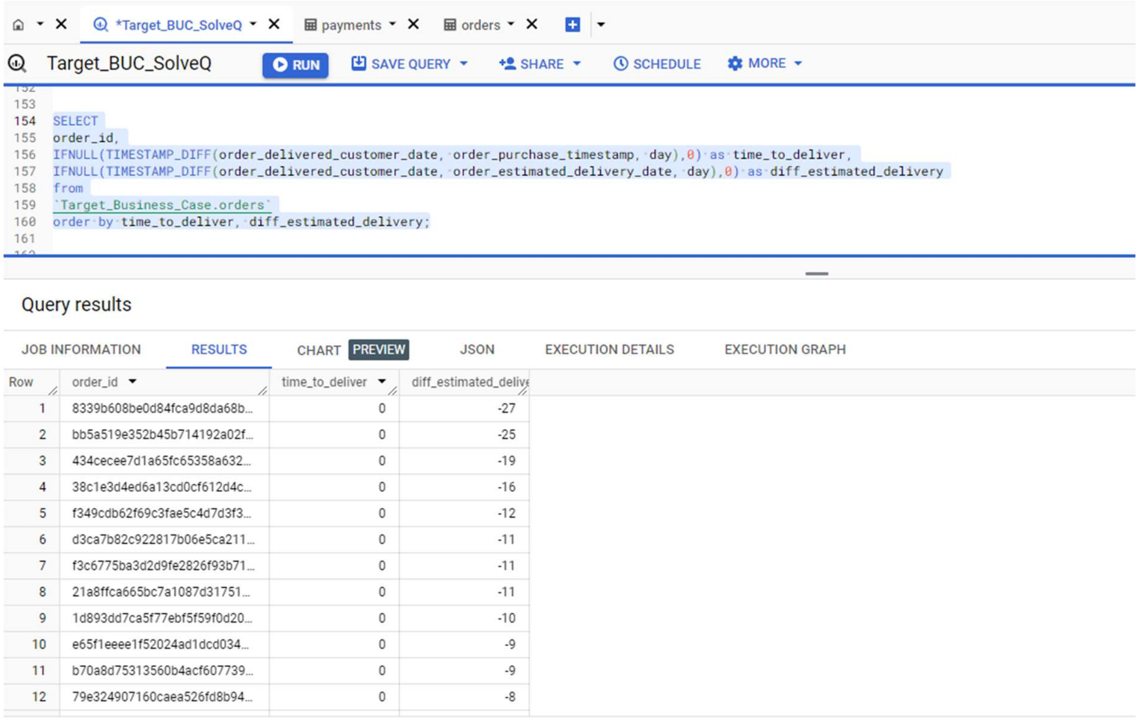
```
SELECT
order_id,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp,
day),0) as time_to_deliver,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date,
day),0) as diff_estimated_delivery
from
`Target_Business_Case.orders`
order by time_to_deliver, diff_estimated_delivery;
```

Big Query Screenshot:



```
SELECT
order_id,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day),0) as time_to_deliver,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0) as diff_estimated_delivery
from
`Target_Business_Case.orders`
order by time_to_deliver desc , diff_estimated_delivery desc;
```

Row	order_id	time_to_deliver	diff_estimated_delive
1	1b3190b2dfa9d789e1f14c05b...	208	188
2	ca07593549f1816d26a572e06...	209	181
3	47b40429ed8cce3aee9199792...	191	175
4	2fe324febf907e3ea3f2aa9650...	189	167
5	285ab9426d6982034523a855f...	194	166
6	440d0d17af552815d15a9e41a...	195	165
7	c27815f7e3dd0b926b5855262...	187	162
8	0f4519c5f1c541ddec9f21b3bd...	194	161
9	d24e8541128cea179a11a6517...	175	161
10	2d7561026d542c8dbd8f0daea...	188	159
11	2fb597c2f772eca01b1f5c561b...	194	155
12	6e82dcfb5eada6283dba34f16...	182	155



```
SELECT
order_id,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day),0) as time_to_deliver,
IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0) as diff_estimated_delivery
from
`Target_Business_Case.orders`
order by time_to_deliver, diff_estimated_delivery;
```

Row	order_id	time_to_deliver	diff_estimated_delive
1	8339b608be0d84fca9d8da68b...	0	-27
2	bb5a519e352b45b714192a02f...	0	-25
3	434cecee7d1a65fc65358a632...	0	-19
4	38c1e3d4ed6a13cd0cf612d4c...	0	-16
5	f349cdb62f69c3fae5c4d7d3f3...	0	-12
6	d3ca7b82c922817b06e5ca211...	0	-11
7	f3c6775ba3d2d9fe2826f93b71...	0	-11
8	21a8ffca665bc7a1087d31751...	0	-11
9	1d893dd7ca5f77ebf5f59f0d20...	0	-10
10	e65f1eeee1f52024ad1dcd034...	0	-9
11	b70a8d75313560b4acf607739...	0	-9
12	79e324907160caea526fd8b94...	0	-8

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

Analysis:

We can understand from the calculations that the states of RO, CE, PB, PI, AC are the among the top five highest average freight value. RO having the highest average freight value of 50.81.

While the states of SP, PA, RJ, DF, PR are the top five with lowest average freight value. SP having the lowest average freight value 18.45.

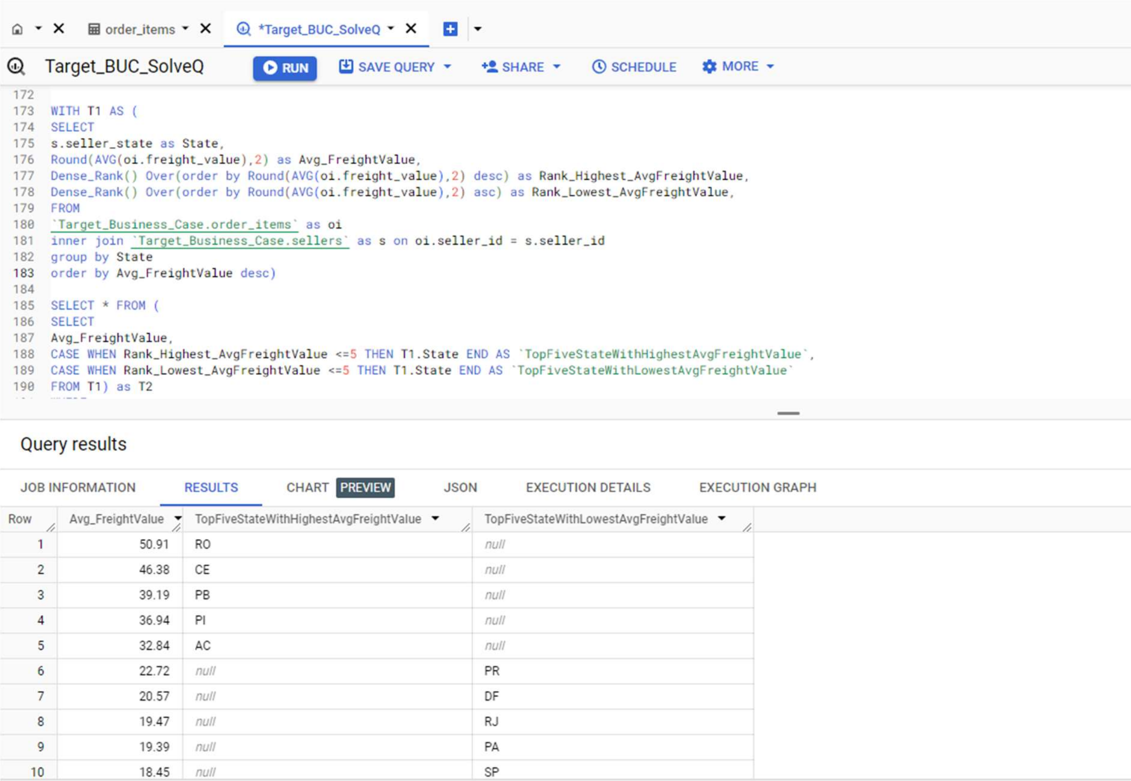
Insights/Recommendations:

Suggest business to perform inspections on states having lower average freight values to understand if they are experiencing any sort of challenges to handle orders or the general nature of the orders are less likely to have higher quantity from these regions. Further feedbacks should be taken into account from states falling on the higher end of the average freight values to understand their viewpoints and help them sustain the same.

Query:

```
WITH T1 AS (
SELECT
s.seller_state as State,
Round(AVG(oi.freight_value),2) as Avg_FreightValue,
Dense_Rank() Over(order by Round(AVG(oi.freight_value),2) desc) as
Rank_Highest_AvgFreightValue,
Dense_Rank() Over(order by Round(AVG(oi.freight_value),2) asc) as
Rank_Lowest_AvgFreightValue,
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by State
order by Avg_FreightValue desc)

SELECT * FROM (
SELECT
Avg_FreightValue,
CASE WHEN Rank_Highest_AvgFreightValue <=5 THEN T1.State END AS
`TopFiveStateWithHighestAvgFreightValue`,
CASE WHEN Rank_Lowest_AvgFreightValue <=5 THEN T1.State END AS
`TopFiveStateWithLowestAvgFreightValue`
FROM T1) as T2
WHERE
`TopFiveStateWithHighestAvgFreightValue` is not null or
`TopFiveStateWithLowestAvgFreightValue` is not null
```

Big Query Screenshot:



```sql

WITH T1 AS (
SELECT
s.seller_state as State,
Round(AVG(oi.freight_value),2) as Avg_FreightValue,
Dense_Rank() Over(order by Round(AVG(oi.freight_value),2) desc) as Rank_Highest_AvgFreightValue,
Dense_Rank() Over(order by Round(AVG(oi.freight_value),2) asc) as Rank_Lowest_AvgFreightValue,
FROM
`Target_Business_Case.order_items` as oi
inner join `Target_Business_Case.sellers` as s on oi.seller_id = s.seller_id
group by State
order by Avg_FreightValue desc)

SELECT * FROM (
SELECT
Avg_FreightValue,
CASE WHEN Rank_Highest_AvgFreightValue <=5 THEN T1.State END AS `TopFiveStateWithHighestAvgFreightValue`,
CASE WHEN Rank_Lowest_AvgFreightValue <=5 THEN T1.State END AS `TopFiveStateWithLowestAvgFreightValue`
FROM T1) as T2
```

Row	Avg_FreightValue	TopFiveStateWithHighestAvgFreightValue	TopFiveStateWithLowestAvgFreightValue
1	50.91	RO	null
2	46.38	CE	null
3	39.19	PB	null
4	36.94	PI	null
5	32.84	AC	null
6	22.72	null	PR
7	20.57	null	DF
8	19.47	null	RJ
9	19.39	null	PA
10	18.45	null	SP

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

Analysis:

The outcome of this calculation clearly shows that following states: SP, MG, PR, DF, RJ, RS, SC, GO, MS, ES are having the lowest average time of delivery. For the state of SP having the lowest average time of delivery of 8 days. The next lowest average time of delivery is 11 days which is seen for states of MG and PR. State of DF takes average time of 12 days to delivery orders. States like RJ, RS, SC takes on average 14 days and GO, MS, ES are taking 15 days.

On other end of the spectrum, we find AP, AM, AL, PA, SE, taking 27, 24, 23, 22 days respectively as average time to deliver orders and states of SE, RR taking 21 days to deliver orders are among the highest average time taking states to deliver orders to customers.

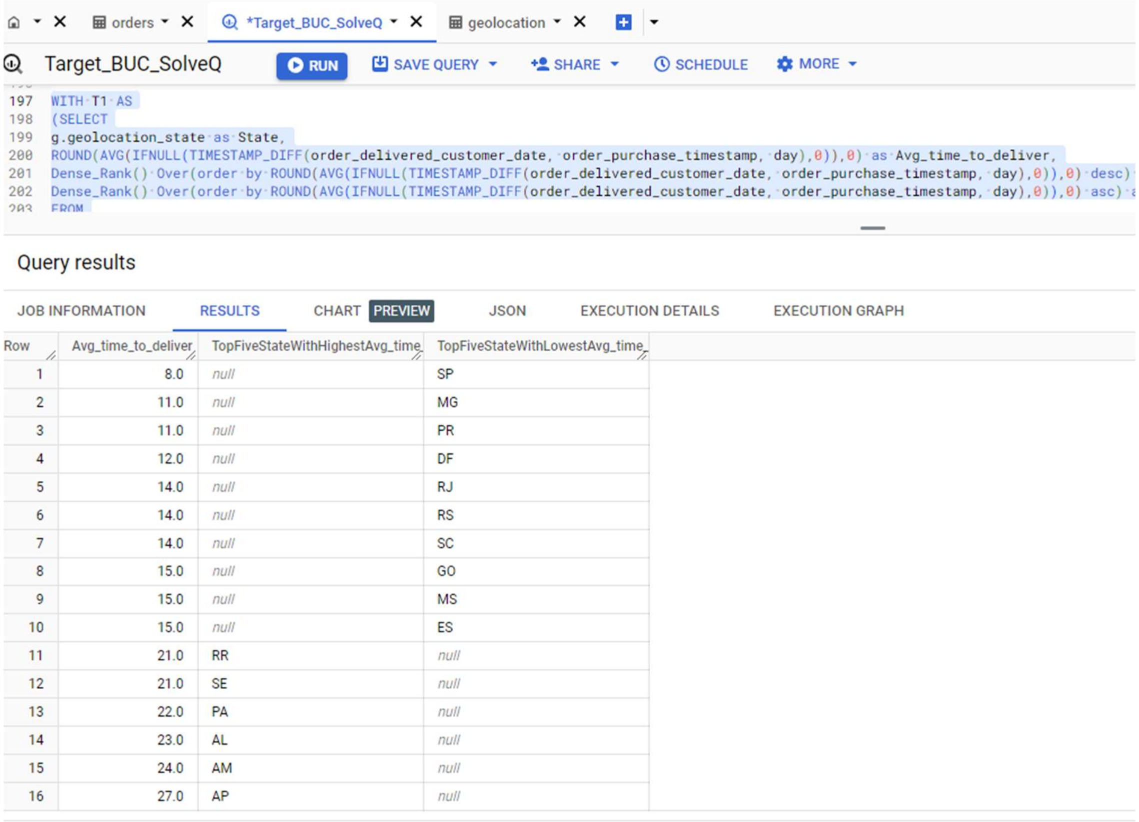
Insights/Recommendations:

For states having a delivery time average of above two weeks, for such cases business can implement feedback mechanisms from customers and delivery partners if there is some external or unpredictable conditions that are leading to the higher time of delivery of is it as per the estimated time of delivery.

Query:

```
WITH T1 AS
(SELECT
g.geolocation_state as State,
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, day),0)),0) as Avg_time_to_deliver,
Dense_Rank() Over(order by
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, day),0)),0) desc) as Rank_Highest_Avg_time_to_deliver,
Dense_Rank() Over(order by
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, day),0)),0) asc) as Rank_Lowest_Avg_time_to_deliver,
FROM
`Target_Business_Case.orders` as o inner join `Target_Business_Case.customers` as c
on o.customer_id = c.customer_id inner join `Target_Business_Case.geolocation` as g
on c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
group by State
order by Avg_time_to_deliver)

SELECT * FROM(
SELECT Avg_time_to_deliver,
CASE WHEN Rank_Highest_Avg_time_to_deliver <=5 THEN T1.State END AS
`TopFiveStateWithHighestAvg_time_to_deliver`,
CASE WHEN Rank_Lowest_Avg_time_to_deliver <=5 THEN T1.State END AS
`TopFiveStateWithLowestAvg_time_to_deliver`
FROM T1)
WHERE `TopFiveStateWithHighestAvg_time_to_deliver` is not null or
`TopFiveStateWithLowestAvg_time_to_deliver` is not null;
```

Big Query Screenshot:



```
WITH T1 AS
(SELECT
g.geolocation_state as State,
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day),0)),0) as Avg_time_to_deliver,
Dense_Rank() Over(order by ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day),0)),0) desc)
Dense_Rank() Over(order by ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day),0)),0) asc) a
FROM
```

Query results

Row	Avg_time_to_deliver	TopFiveStateWithHighestAvg_time	TopFiveStateWithLowestAvg_time
1	8.0	null	SP
2	11.0	null	MG
3	11.0	null	PR
4	12.0	null	DF
5	14.0	null	RJ
6	14.0	null	RS
7	14.0	null	SC
8	15.0	null	GO
9	15.0	null	MS
10	15.0	null	ES
11	21.0	RR	null
12	21.0	SE	null
13	22.0	PA	null
14	23.0	AL	null
15	24.0	AM	null
16	27.0	AP	null

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

You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Analysis:

Through this outcome of the calculation of average of the time difference between order delivery time and estimated delivery time, tells us that in all of the state we see that orders are reaching the customers before the estimated delivery time.

In states like AM, RR it is reaching to customers about 20 days before the estimated time, in states like RO it is reaching 19 days prior, in states like AC, AP it is reaching 18 days prior to the, in states like

PA, MT order reaches about 14 days prior and in states like PR, RN, PB, RS orders are likely to reach 13 days prior to the estimated delivery date. These states are having highest values of average difference of estimated to actual delivery time.

While when we observe the data to check the top five lowest values of average difference of estimated to actual delivery time, we find several states in them.

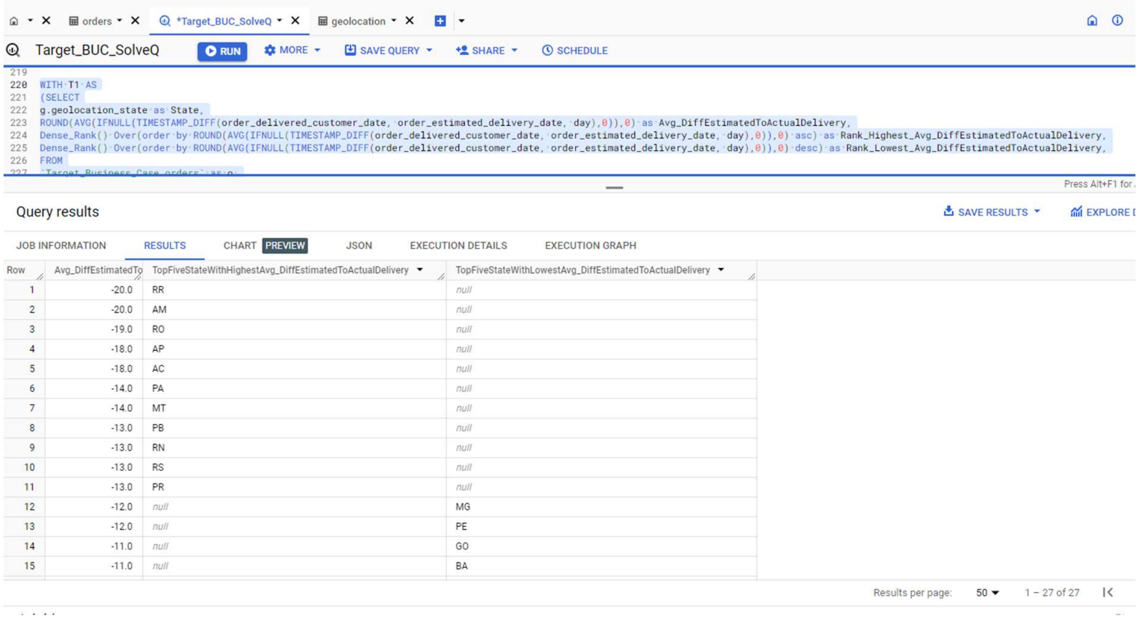
State of SE, AL are having the least value that is 8 days. MA gets orders 9 days prior to the estimated.

States such as ES, CE, SP, MS are getting orders 10 days prior to the estimated delivery time. There are total of seven states (TO, PI, DF, SC, RJ, BA, GO) who receives orders 11-day prior. States like PE and MG are getting orders 12 days prior.

Query:

```
WITH T1 AS
(SELECT
g.geolocation_state as State,
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, day),0)),0) as Avg_DiffEstimatedToActualDelivery,
Dense_Rank() Over(order by
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, day),0)),0) asc) as
Rank_Highest_Avg_DiffEstimatedToActualDelivery,
Dense_Rank() Over(order by
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date,
order_estimated_delivery_date, day),0)),0) desc) as
Rank_Lowest_Avg_DiffEstimatedToActualDelivery,
FROM
`Target_Business_Case.orders` as o
inner join `Target_Business_Case.customers` as c on o.customer_id = c.customer_id
inner join `Target_Business_Case.geolocation` as g on c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
WHERE o.order_status = 'delivered'
group by State
order by Avg_DiffEstimatedToActualDelivery)

SELECT * FROM(
SELECT Avg_DiffEstimatedToActualDelivery,
CASE WHEN Rank_Highest_Avg_DiffEstimatedToActualDelivery <=5 THEN T1.State END AS
`TopFiveStateWithHighestAvg_DiffEstimatedToActualDelivery`,
CASE WHEN Rank_Lowest_Avg_DiffEstimatedToActualDelivery <=5 THEN T1.State END AS
`TopFiveStateWithLowestAvg_DiffEstimatedToActualDelivery`
FROM T1) as T2
WHERE `TopFiveStateWithHighestAvg_DiffEstimatedToActualDelivery` is not null or
`TopFiveStateWithLowestAvg_DiffEstimatedToActualDelivery` is not null;
```

Big Query Screenshot:

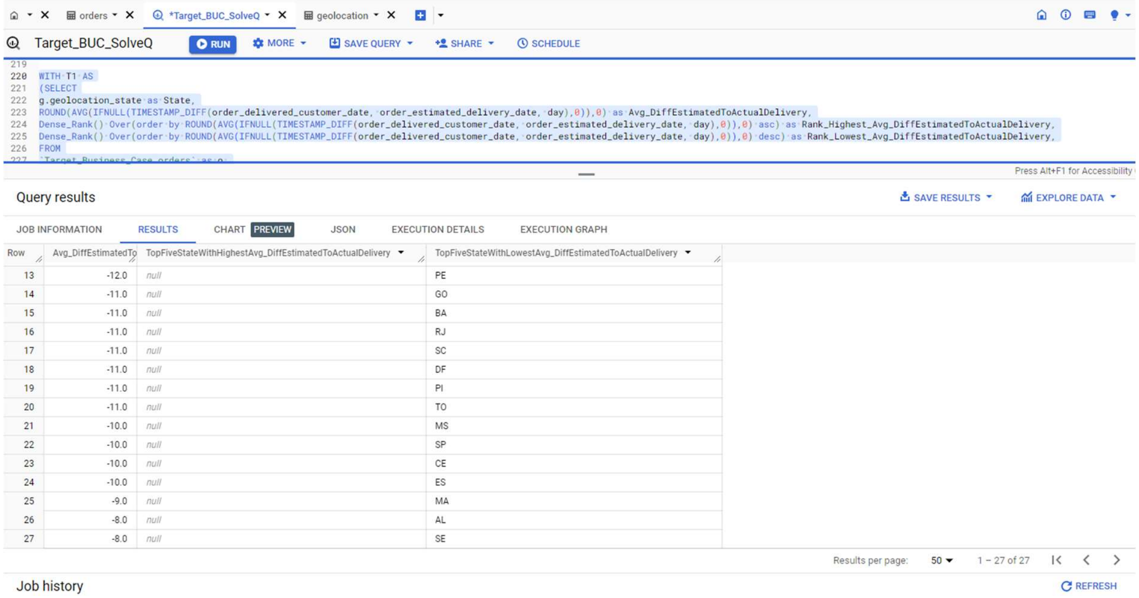


```
WITH T1 AS
(SELECT
g.geolocation_state as State,
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0)),0) as Avg_DiffEstimatedToActualDelivery,
Dense_Rank() Over(order by ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0)),0) asc) as Rank_Highest_Avg_DiffEstimatedToActualDelivery,
Dense_Rank() Over(order by ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0)),0) desc) as Rank_Lowest_Avg_DiffEstimatedToActualDelivery,
FROM
`Target_Business_Case.orders` as o
```

Query results

Row	Avg_DiffEstimatedTo	TopFiveStateWithHighestAvg_DiffEstimatedToActualDelivery	TopFiveStateWithLowestAvg_DiffEstimatedToActualDelivery
1	-20.0	RR	null
2	-20.0	AM	null
3	-19.0	RO	null
4	-18.0	AP	null
5	-18.0	AC	null
6	-14.0	PA	null
7	-14.0	MT	null
8	-13.0	PB	null
9	-13.0	RN	null
10	-13.0	RS	null
11	-13.0	PR	null
12	-12.0	null	MG
13	-12.0	null	PE
14	-11.0	null	GO
15	-11.0	null	BA

Results per page: 50 1 – 27 of 27



```
WITH T1 AS
(SELECT
g.geolocation_state as State,
ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0)),0) as Avg_DiffEstimatedToActualDelivery,
Dense_Rank() Over(order by ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0)),0) asc) as Rank_Highest_Avg_DiffEstimatedToActualDelivery,
Dense_Rank() Over(order by ROUND(AVG(IFNULL(TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date, day),0)),0) desc) as Rank_Lowest_Avg_DiffEstimatedToActualDelivery,
FROM
`Target_Business_Case.orders` as o
```

Query results

Row	Avg_DiffEstimatedTo	TopFiveStateWithHighestAvg_DiffEstimatedToActualDelivery	TopFiveStateWithLowestAvg_DiffEstimatedToActualDelivery
13	-12.0	null	PE
14	-11.0	null	GO
15	-11.0	null	BA
16	-11.0	null	RJ
17	-11.0	null	SC
18	-11.0	null	DF
19	-11.0	null	PI
20	-11.0	null	TO
21	-10.0	null	MS
22	-10.0	null	SP
23	-10.0	null	CE
24	-10.0	null	ES
25	-9.0	null	MA
26	-8.0	null	AL
27	-8.0	null	SE

Results per page: 50 1 – 27 of 27

Job history

REFRESH

Q6: Analysis based on the payments:

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

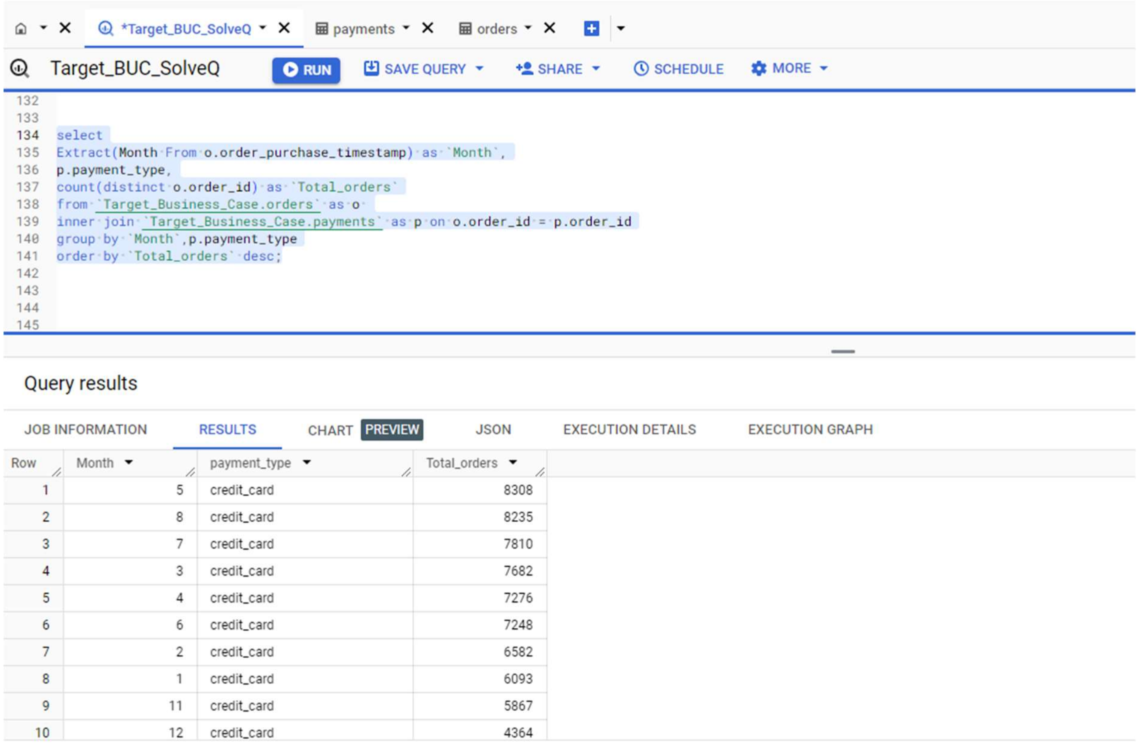
Analysis: On observing the total orders based on each payment type over the months, we can understand that the usage of credit card as a payment type is mostly preferred among customers. On every month from January to December the most popular payment type is credit card.

Further we can see that UPI and vouchers are next most preferred ways of payment for customers.

Query:

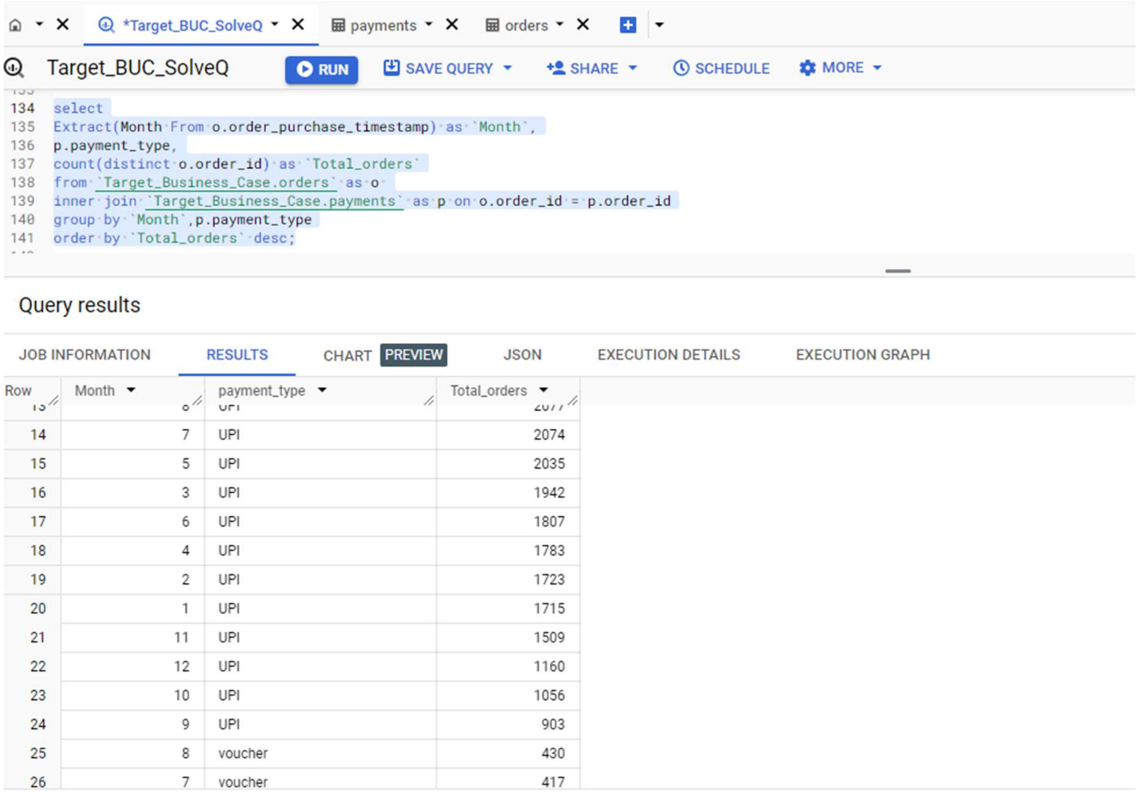
```
select
Extract(Month From o.order_purchase_timestamp) as `Month`,
p.payment_type,
count(distinct o.order_id) as `Total_orders`
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.payments` as p on o.order_id = p.order_id
group by `Month`,p.payment_type
order by `Total_orders` desc;
```

Big Query Screenshot:



```
select
Extract(Month From o.order_purchase_timestamp) as `Month`,
p.payment_type,
count(distinct o.order_id) as `Total_orders`
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.payments` as p on o.order_id = p.order_id
group by `Month`,p.payment_type
order by `Total_orders` desc;
```

Row	Month	payment_type	Total_orders
1	5	credit_card	8308
2	8	credit_card	8235
3	7	credit_card	7810
4	3	credit_card	7682
5	4	credit_card	7276
6	6	credit_card	7248
7	2	credit_card	6582
8	1	credit_card	6093
9	11	credit_card	5867
10	12	credit_card	4364



```
select
Extract(Month From o.order_purchase_timestamp) as `Month`,
p.payment_type,
count(distinct o.order_id) as `Total_orders`
from `Target_Business_Case.orders` as o
inner join `Target_Business_Case.payments` as p on o.order_id = p.order_id
group by `Month`,p.payment_type
order by `Total_orders` desc;
```

Query results

Row	Month	payment_type	Total_orders
13	8	UPI	2077
14	7	UPI	2074
15	5	UPI	2035
16	3	UPI	1942
17	6	UPI	1807
18	4	UPI	1783
19	2	UPI	1723
20	1	UPI	1715
21	11	UPI	1509
22	12	UPI	1160
23	10	UPI	1056
24	9	UPI	903
25	8	voucher	430
26	7	voucher	417

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

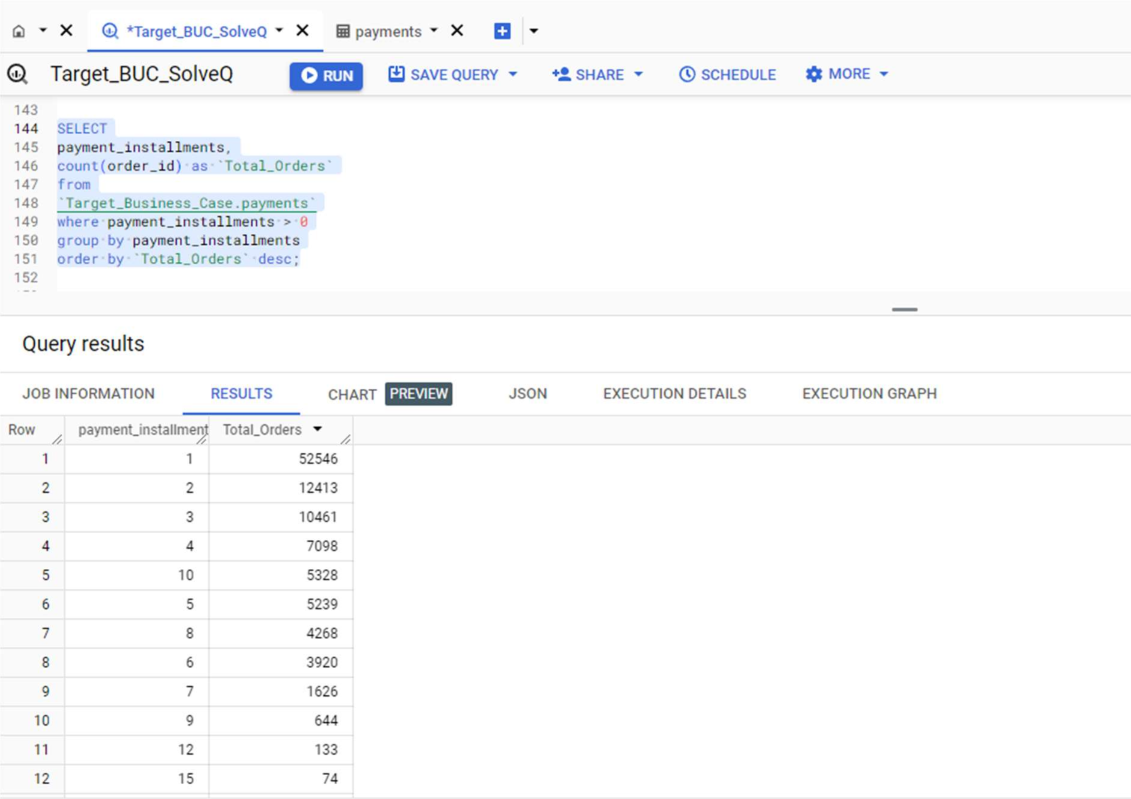
Analysis:

From this observation we can understand that most of the orders are on their first instalment. There are total of 24 instalments available for the customers to complete the payment towards the orders placed by them. Majority of the orders fall under the first four instalments category.

Query:

```
SELECT
payment_installments,
count(order_id) as `Total_Orders`
from
`Target_Business_Case.payments`
where payment_installments > 0
group by payment_installments
order by `Total_Orders` desc;
```

Big Query Screenshot:



```
SELECT
payment_installments,
count(order_id) as `Total_Orders`
from
`Target_Business_Case.payments`
where payment_installments > 0
group by payment_installments
order by `Total_Orders` desc;
```

Query results

Row	payment_installment	Total_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