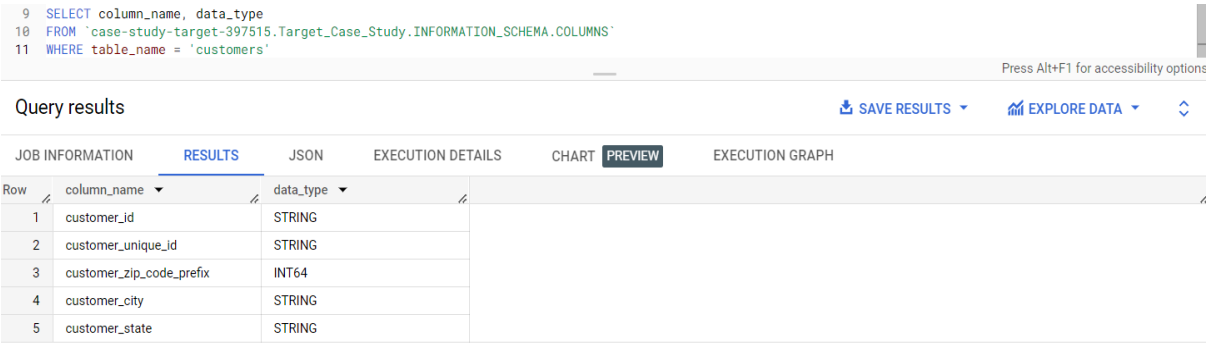
Target Case Study

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

1. Data type of all columns in the "customers" table.

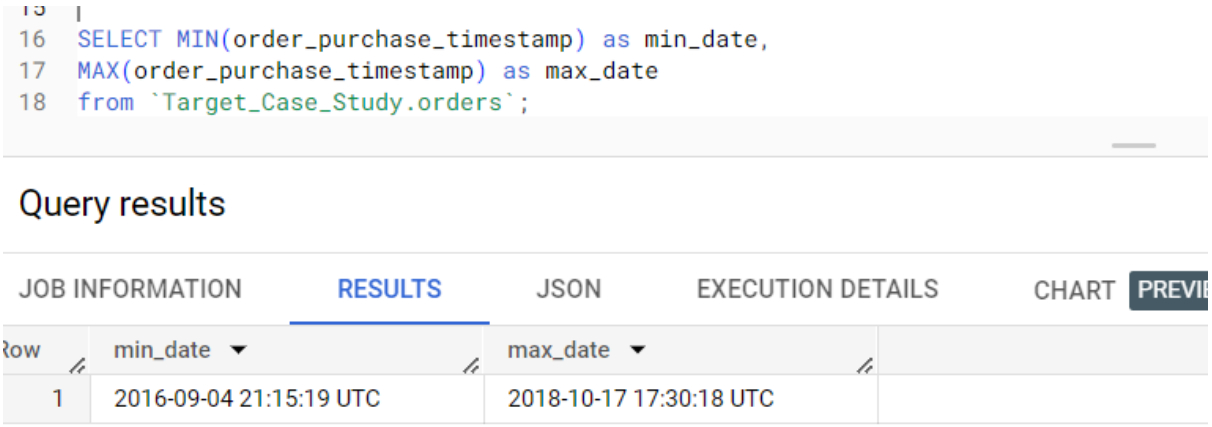
```
SELECT column_name, data_type
FROM `case-study-target-397515.Target_Case_Study.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
```



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

2. Get the time range between which the orders were placed.

```
SELECT MIN(order_purchase_timestamp) as min_date,
MAX(order_purchase_timestamp) as max_date
from `Target_Case_Study.orders`;
```



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

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

```
SELECT g.geolocation_city, g.geolocation_state, COUNT(c.customer_id) AS
customer_count
FROM `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
```

```
WHERE o.order_purchase_timestamp BETWEEN '2016-09-04 21:15:19 UTC' AND
'2018-10-17 17:30:18 UTC'
GROUP BY g.geolocation_city, g.geolocation_state
ORDER BY g.geolocation_state, g.geolocation_city;
```

```
SELECT g.geolocation_city, g.geolocation_state, COUNT(c.customer_id) AS customer_count
FROM `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
WHERE o.order_purchase_timestamp BETWEEN '2016-09-04 21:15:19 UTC' AND '2018-10-17 17:30:18 UTC'
GROUP BY g.geolocation_city, g.geolocation_state
ORDER BY g.geolocation_state, g.geolocation_city;
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRA

Row	geolocation_city	geolocation_state	customer_count
1	brasileia	AC	8
2	brasiléia	AC	5
3	cruzeiro do sul	AC	321
4	epitaciolandia	AC	6
5	epitaciolândia	AC	6
6	manoel urbano	AC	7
7	porto acre	AC	3

Results per pa

2. In-depth Exploration:

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

```
SELECT
 EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
 COUNT(DISTINCT order_id) AS order_count
FROM
 `Target_Case_Study.orders`
WHERE
 order_status = 'delivered'
 AND order_purchase_timestamp IS NOT NULL
GROUP BY
 year
ORDER BY
 year ASC
```

```
SELECT
 EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
 COUNT(DISTINCT order_id) AS order_count
FROM
 `Target_Case_Study.orders`
WHERE
 order_status = 'delivered'
 AND order_purchase_timestamp IS NOT NULL
GROUP BY
 year
ORDER BY
 year ASC
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year	order_count
1	2016	267
2	2017	43428
3	2018	52783

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

```
 SELECT
 EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
 COUNT(DISTINCT order_id) AS order_count
FROM
 `Target_Case_Study.orders`
WHERE
 order_status = 'delivered'
 AND order_purchase_timestamp IS NOT NULL
GROUP BY
 year
ORDER BY
 year ASC
```

```
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) AS month,
COUNT(DISTINCT order_id) AS order_count
FROM
`Target_Case_Study.orders`
WHERE
order_status = 'delivered'
AND order_purchase_timestamp IS NOT NULL
GROUP BY
month
ORDER BY
month ASC
```

JOB INFORMATION | RESULTS | JSON | EXECUTION DETA

Row	month	order_count
1	1	7819
2	2	8208
3	3	9549
4	4	9101
5	5	10295
6	6	9234
7	7	10031
8	8	10544
9	9	4151
10	10	4743
11	11	7289
12	12	5514

c. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
   i. 0-6 hrs : Dawn
   ii. 7-12 hrs : Mornings
   iii. 13-18 hrs : Afternoon
   iv. 19-23 hrs : Night

```
SELECT
   CASE
      WHEN EXTRACT(HOUR FROM CAST(order_purchase_timestamp AS TIMESTAMP) AT TIME
ZONE 'America/Sao_Paulo') >= 5 AND EXTRACT(HOUR FROM CAST(order_purchase_timestamp
AS TIMESTAMP) AT TIME ZONE 'America/Sao_Paulo') < 12 THEN 'Morning'
```

```
        WHEN EXTRACT(HOUR FROM CAST(order_purchase_timestamp AS TIMESTAMP) AT TIME
ZONE 'America/Sao_Paulo') >= 12 AND EXTRACT(HOUR FROM CAST(order_purchase_timestamp
AS TIMESTAMP) AT TIME ZONE 'America/Sao_Paulo') < 18 THEN 'Afternoon'
        WHEN EXTRACT(HOUR FROM CAST(order_purchase_timestamp AS TIMESTAMP) AT TIME
ZONE 'America/Sao_Paulo') >= 18 AND EXTRACT(HOUR FROM CAST(order_purchase_timestamp
AS TIMESTAMP) AT TIME ZONE 'America/Sao_Paulo') < 24 THEN 'Night'
        ELSE 'Dawn'
    END AS time_of_day,
    COUNT(DISTINCT orders.customer_id) AS num_customers
FROM
    Target_Case_Study.orders
JOIN
    Target_Case_Study.customers
ON
    orders.customer_id = customers.customer_id
WHERE
    customers.customer_state = 'BR'
    AND CAST(order_purchase_timestamp AS TIMESTAMP) AT TIME ZONE
'America/Sao_Paulo' BETWEEN '2016-09-04 21:15:19' AND '2018-10-17 17:30:18'
GROUP BY
    time_of_day
ORDER BY
    num_customers DESC;
```

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

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

```
SELECT DATE_TRUNC(o.order_purchase_timestamp, MONTH) as month_year,
c.customer_state, COUNT(DISTINCT o.order_id) as num_orders
FROM `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c
ON o.customer_id = c.customer_id
GROUP BY month_year, customer_state
ORDER BY month_year, customer_state
```

```google_sql
SELECT DATE_TRUNC(o.order_purchase_timestamp, MONTH) as month_year, c.customer_state, COUNT(DISTINCT o.order_id) as num_orders
FROM `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c
ON o.customer_id = c.customer_id
GROUP BY month_year, customer_state
ORDER BY month_year, customer_state
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	month_year	customer_state	num_orders
1	2016-09-01 00:00:00 UTC	RR	1
2	2016-09-01 00:00:00 UTC	RS	1
3	2016-09-01 00:00:00 UTC	SP	2
4	2016-10-01 00:00:00 UTC	AL	2
5	2016-10-01 00:00:00 UTC	BA	4
6	2016-10-01 00:00:00 UTC	CE	8
7	2016-10-01 00:00:00 UTC	DF	6
8	2016-10-01 00:00:00 UTC	ES	4
9	2016-10-01 00:00:00 UTC	GO	9

Query results

SAVE RESULTS EXPLORE DATA

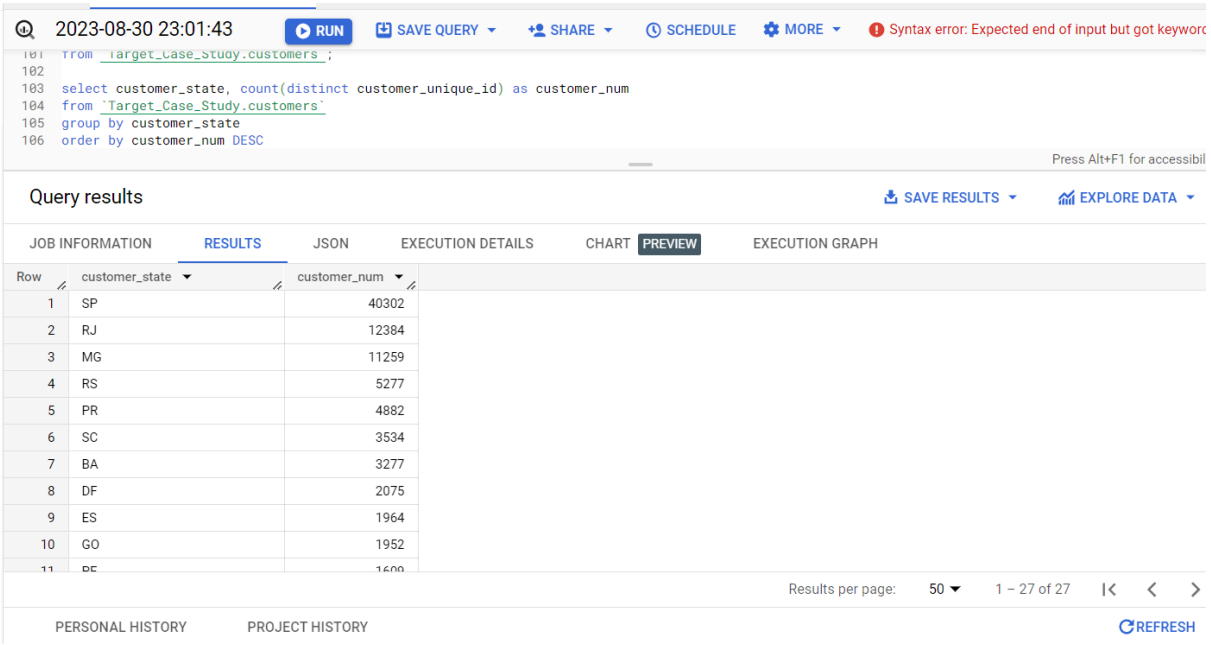
JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	month_year	customer_state	num_orders
1	2016-09-01 00:00:00 UTC	RR	1
2	2016-09-01 00:00:00 UTC	RS	1
3	2016-09-01 00:00:00 UTC	SP	2
4	2016-10-01 00:00:00 UTC	AL	2
5	2016-10-01 00:00:00 UTC	BA	4
6	2016-10-01 00:00:00 UTC	CE	8
7	2016-10-01 00:00:00 UTC	DF	6
8	2016-10-01 00:00:00 UTC	ES	4
9	2016-10-01 00:00:00 UTC	GO	9
10	2016-10-01 00:00:00 UTC	MA	4
11	2016-10-01 00:00:00 UTC	MG	40
12	2016-10-01 00:00:00 UTC	MT	3

Results per page: 50 1 – 50 of 565

b. How are the customers distributed across all the states?

```google_sql
select customer_state, count(distinct customer_unique_id) as customer_num
from `Target_Case_Study.customers`
group by customer_state
order by customer_num DESC
```



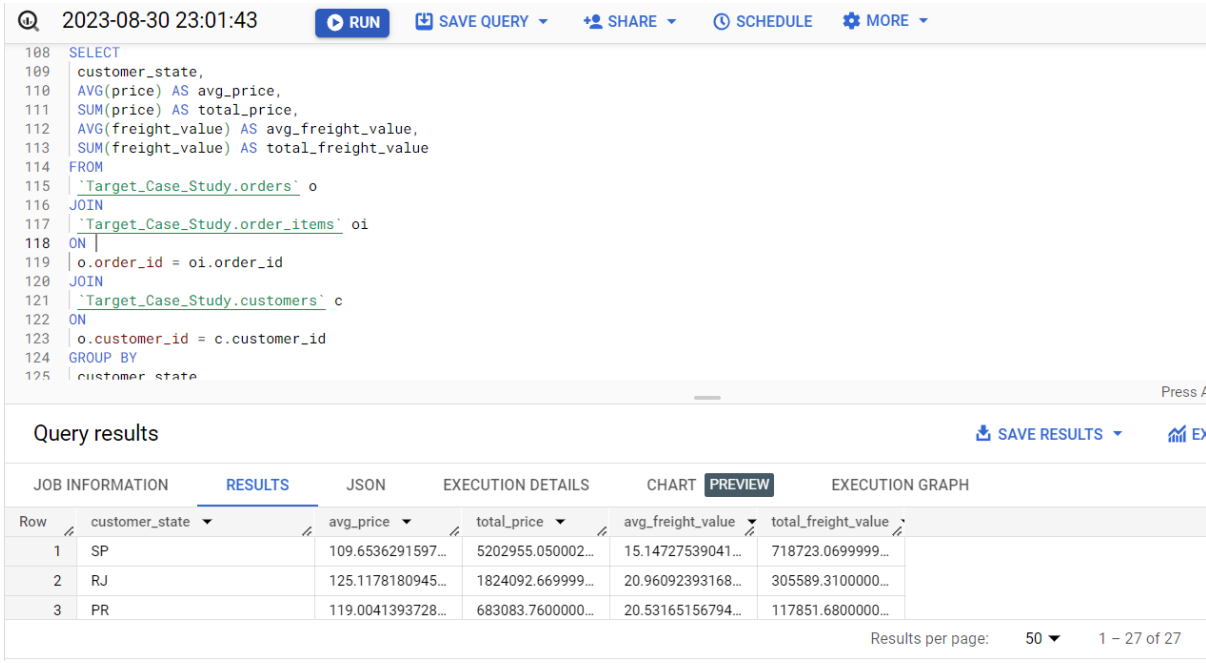
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

a. 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.

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

```
SELECT
 customer_state,
 AVG(price) AS avg_price,
 SUM(price) AS total_price,
 AVG(freight_value) AS avg_freight_value,
 SUM(freight_value) AS total_freight_value
FROM
 `Target_Case_Study.orders` o
JOIN
 `Target_Case_Study.order_items` oi
ON
 o.order_id = oi.order_id
JOIN
 `Target_Case_Study.customers` c
```

```
ON
 o.customer_id = c.customer_id
GROUP BY
 customer_state
```



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

```
SELECT
 customer_state,
 AVG(price) AS avg_price,
 SUM(price) AS total_price,
 AVG(freight_value) AS avg_freight_value,
 SUM(freight_value) AS total_freight_value
FROM
 `Target_Case_Study.orders` o
JOIN
 `Target_Case_Study.order_items` oi
ON
 o.order_id = oi.order_id
JOIN
 `Target_Case_Study.customers` c
ON
 o.customer_id = c.customer_id
GROUP BY
 customer_state
```

```
SELECT
 customer_state,
 AVG(price) AS avg_price,
 SUM(price) AS total_price,
 AVG(freight_value) AS avg_freight_value,
 SUM(freight_value) AS total_freight_value
FROM
 `Target_Case_Study.orders` o
JOIN
 `Target_Case_Study.order_items` oi
ON
 o.order_id = oi.order_id
JOIN
 `Target_Case_Study.customers` c
ON
 o.customer_id = c.customer_id
GROUP BY
```

Query results

SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	customer_state	avg_price	total_price	avg_freight_value	total_freight_value
1	SP	109.6536291597...	5202955.050002...	15.14727539041...	718723.0699999...
2	RJ	125.1178180945...	1824092.669999...	20.96092393168...	305589.3100000...
3	PR	119.0041393728...	683083.7600000...	20.53165156794...	117851.6800000...

Results per page: 50 1 – 27 of 27

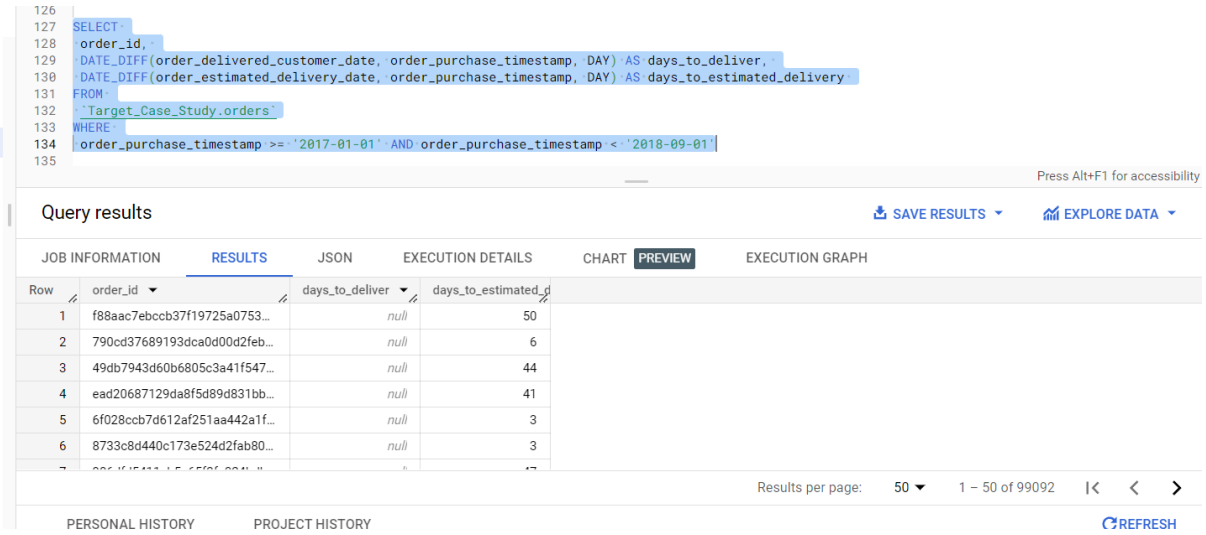
5. Analysis based on sales, freight and delivery time.

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.

```
SELECT
 order_id,
 DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS days_to_deliver,
 DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, DAY) AS days_to_estimated_delivery
FROM
 `Target_Case_Study.orders`
WHERE
 order_purchase_timestamp >= '2017-01-01' AND order_purchase_timestamp < '2018-09-01'
```



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

i. **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
ii. **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

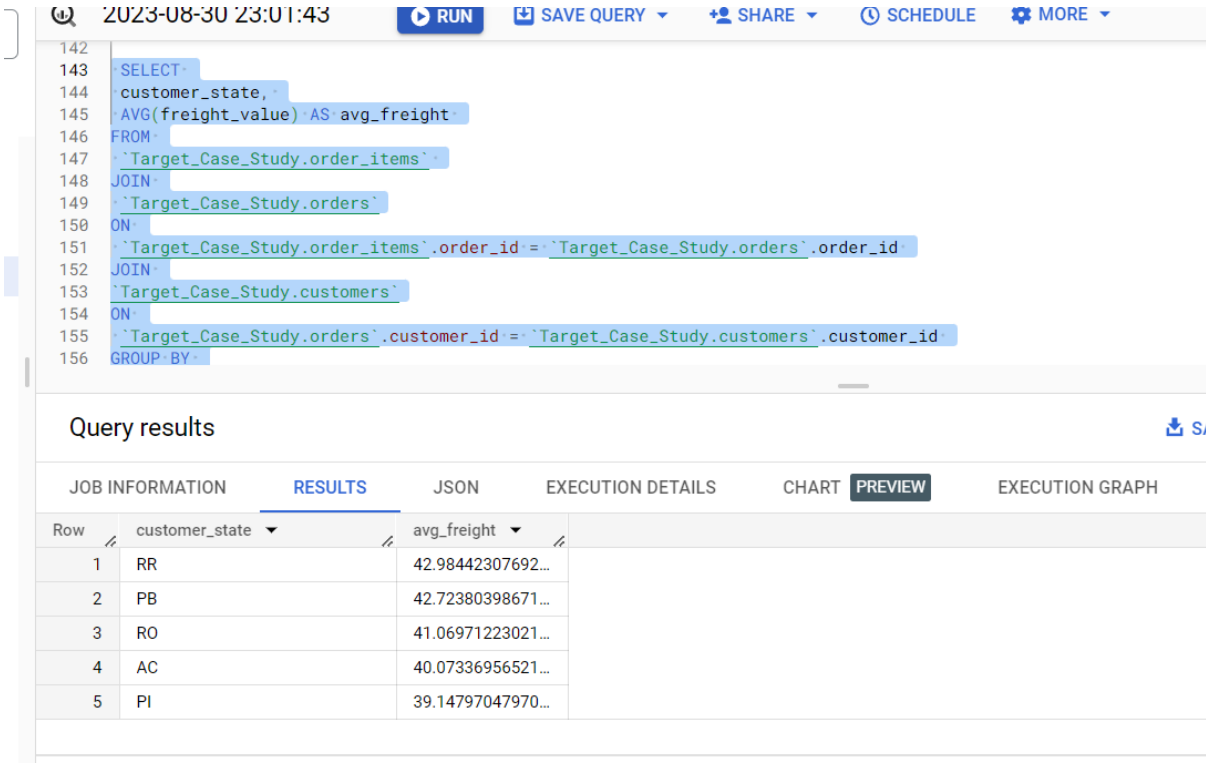
```
SELECT
 order_id,
 TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) AS
time_to_delivery,
 TIMESTAMP_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY)
AS diff_estimated_delivery
FROM
 `Target_Case_Study.orders`;
```

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

Five states with highest freight value

```
SELECT
customer_state,
AVG(freight_value) AS avg_freight
FROM
 `Target_Case_Study.order_items`
JOIN
 `Target_Case_Study.orders`
ON
 `Target_Case_Study.order_items`.order_id =
`Target_Case_Study.orders`.order_id
JOIN
`Target_Case_Study.customers`
ON
```

```
 `Target_Case_Study.orders`.customer_id =
`Target_Case_Study.customers`.customer_id
GROUP BY
 customer_state
ORDER BY
 avg_freight DESC
LIMIT
5
```

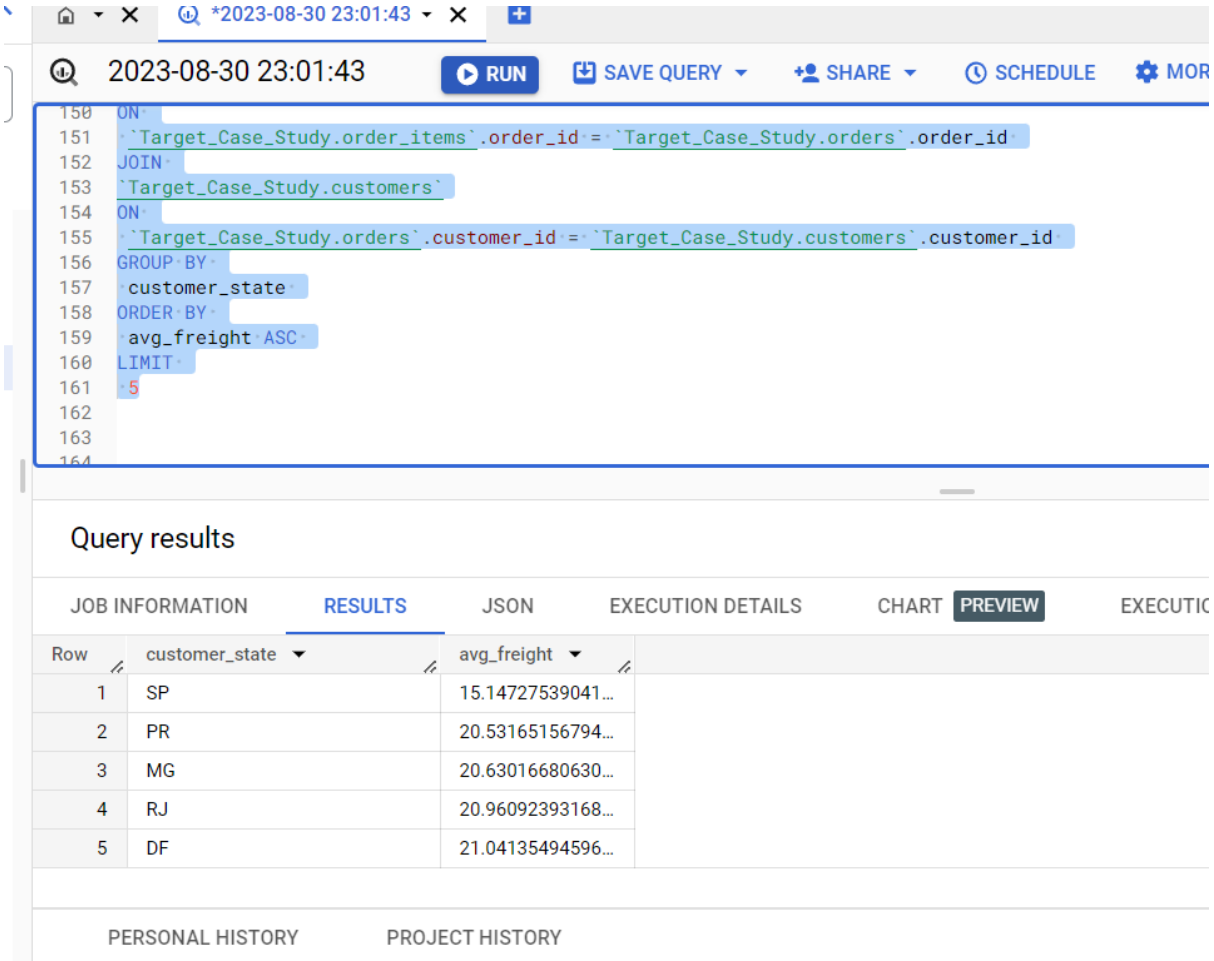


5

Five states with lowest freight value

```
SELECT
 customer_state,
 AVG(freight_value) AS avg_freight
FROM
 `Target_Case_Study.order_items`
JOIN
 `Target_Case_Study.orders`
ON
 `Target_Case_Study.order_items`.order_id =
`Target_Case_Study.orders`.order_id
JOIN
`Target_Case_Study.customers`
ON
```

```
 `Target_Case_Study.orders`.customer_id =
`Target_Case_Study.customers`.customer_id
GROUP BY
 customer_state
ORDER BY
 avg_freight ASC
LIMIT 5
```

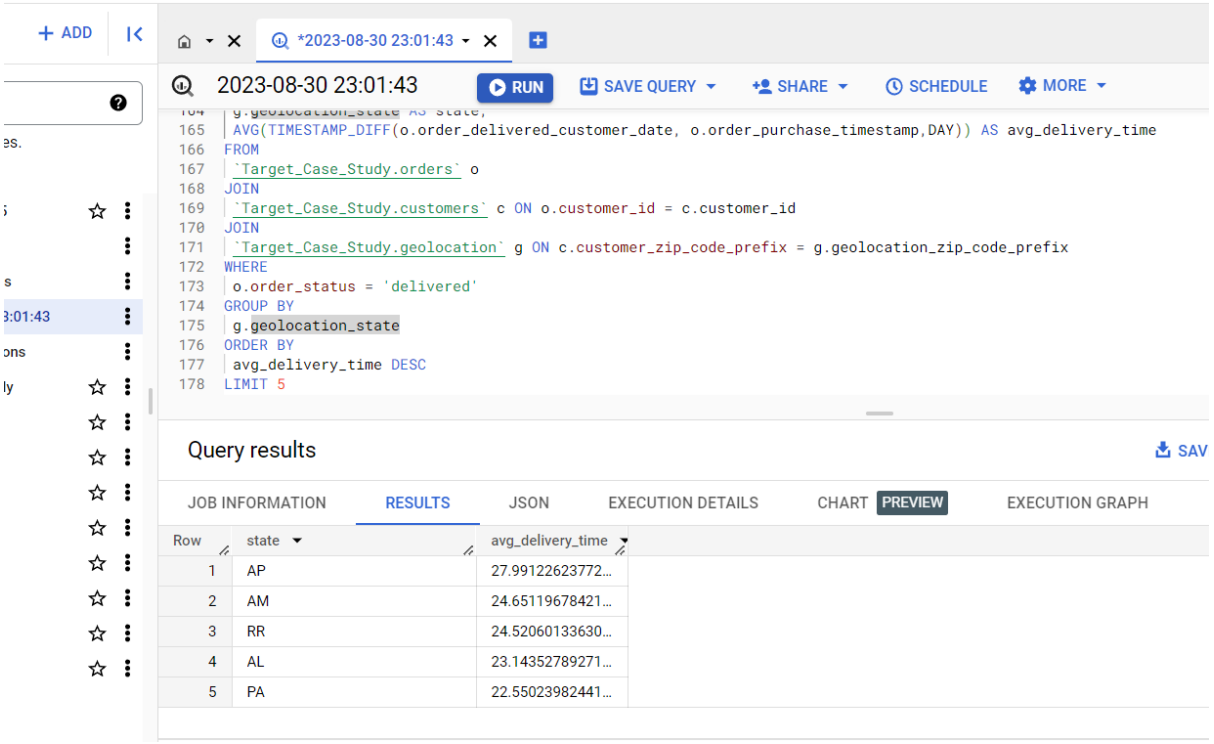


e. Find out the top 5 states with the highest & lowest average delivery time.

Highest

```
SELECT
 g.geolocation_state AS state,
 AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp,DAY)) AS avg_delivery_time
FROM
 `Target_Case_Study.orders` o
JOIN
 `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN
```

```
 `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
WHERE
 o.order_status = 'delivered'
GROUP BY
 g.geolocation_state
ORDER BY
 avg_delivery_time DESC
LIMIT 5
```



Lowest

```
SELECT
 g.geolocation_state AS state,
 AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_purchase_timestamp,DAY)) AS avg_delivery_time
FROM
 `Target_Case_Study.orders` o
JOIN
 `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN
 `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix =
g.geolocation_zip_code_prefix
WHERE
 o.order_status = 'delivered'
GROUP BY
```

```
 g.geolocation_state
ORDER BY
 avg_delivery_time ASC
LIMIT 5
```

2023-08-30 23:01:43 RUN SAVE QUERY SHARE SCHEDULE MORE

```
SELECT
 g.geolocation_state AS state,
 AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,DAY)) AS avg_delivery_time
FROM
 `Target_Case_Study.orders` o
JOIN
 `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
JOIN
 `Target_Case_Study.geolocation` g ON c.customer_zip_code_prefix = g.geolocation_zip_code_prefix
WHERE
 o.order_status = 'delivered'
GROUP BY
 g.geolocation_state
ORDER BY
```

Query results SAVE RESULTS

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	state	avg_delivery_time
1	SP	8.470555045096...
2	PR	11.03876404770...
3	MG	11.41821678372...
4	DF	12.49651789233...
5	SC	14.48408434580...

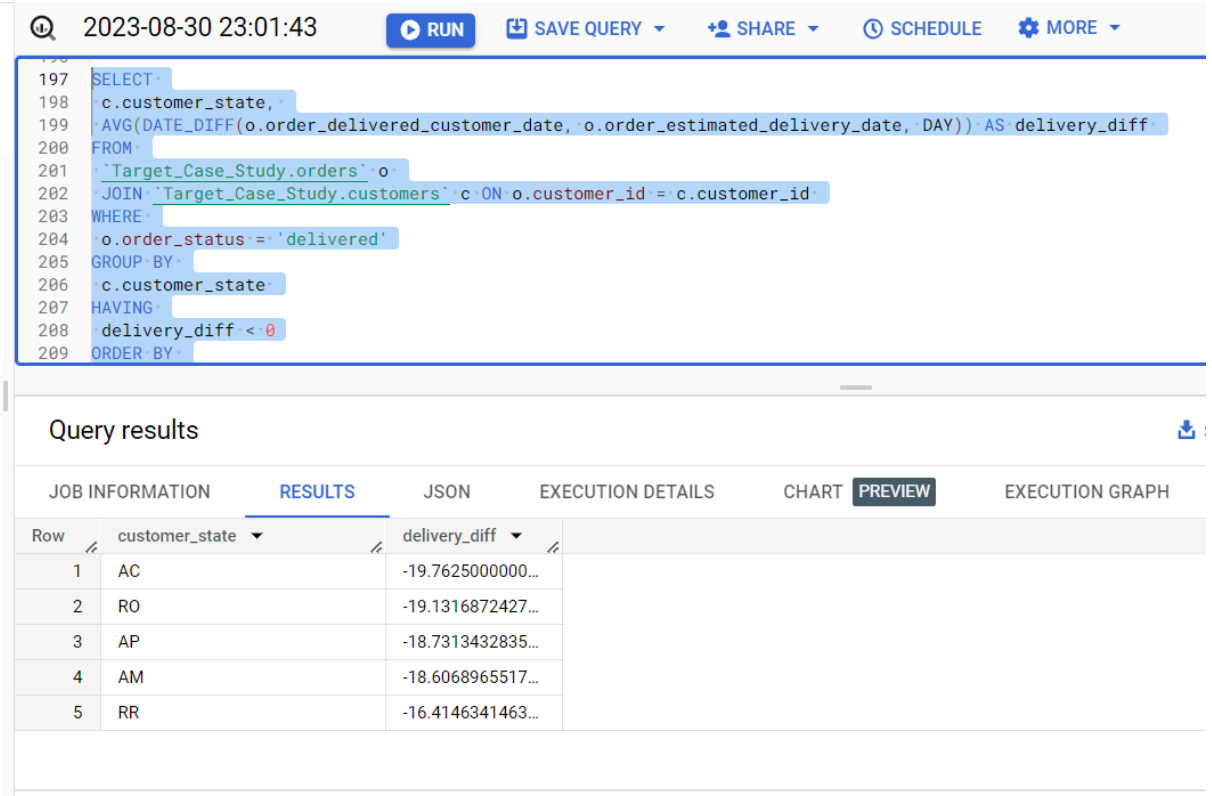
PERSONAL HISTORY PROJECT HISTORY

f. 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.

Top 5 states where the delivery was not so fast

```
SELECT
 c.customer_state,
 AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)) AS delivery_diff
FROM
 `Target_Case_Study.orders` o
 JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
WHERE
 o.order_status = 'delivered'
GROUP BY
 c.customer_state
HAVING
 delivery_diff < 0
```

```
ORDER BY
 delivery_diff ASC
LIMIT 5
```



Top 5 states where the delivery was fast

```
SELECT
 c.customer_state,
 AVG(DATE_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date, DAY)) AS delivery_diff
FROM
 `Target_Case_Study.orders` o
 JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
WHERE
 o.order_status = 'delivered'
GROUP BY
 c.customer_state
HAVING
 delivery_diff < 0
ORDER BY
 delivery_diff DESC
LIMIT
 5
```

2023-08-30 23:01:43 RUN SAVE QUERY SHARE SCHEDULE MORE

```google_sql
SELECT
 c.customer_state,
 AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS delivery_diff
FROM
 `Target_Case_Study.orders` o
 JOIN `Target_Case_Study.customers` c ON o.customer_id = c.customer_id
WHERE
 o.order_status = 'delivered'
GROUP BY
 c.customer_state
HAVING
 delivery_diff < 0
ORDER BY
```

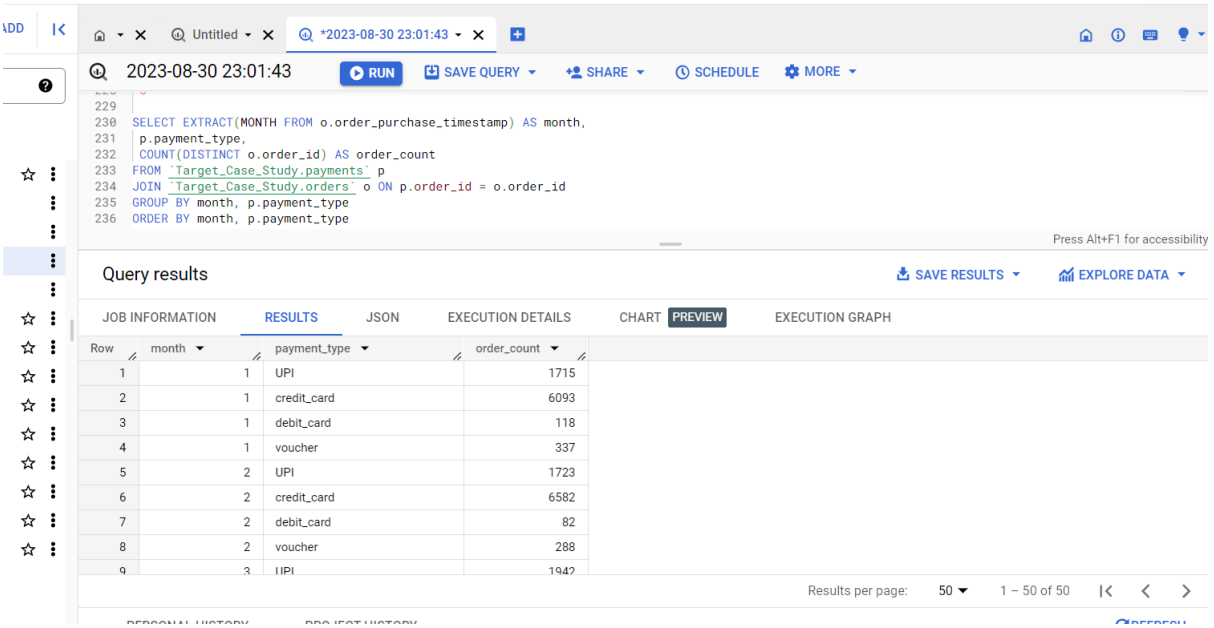
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART PREVIEW | EXECUTION GRAPH

Row	customer_state	delivery_diff
1	AL	-7.94710327455...
2	MA	-8.76847977684...
3	SE	-9.17313432835...
4	ES	-9.61854636591...
5	BA	-9.93488943488...

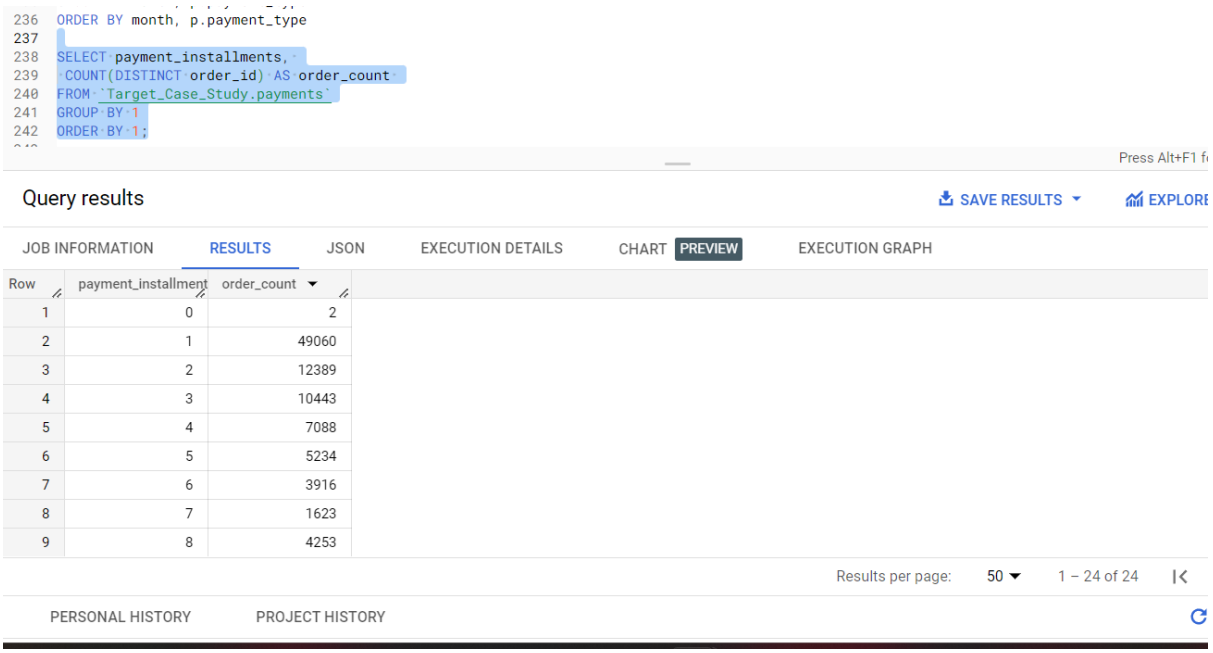
e. **Analysis based on the payments:**
   a. Find the month on month no. of orders placed using different payment types.

```google_sql
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
 p.payment_type,
 COUNT(DISTINCT o.order_id) AS order_count
FROM `Target_Case_Study.payments` p
JOIN `Target_Case_Study.orders` o ON p.order_id = o.order_id
GROUP BY month, p.payment_type
ORDER BY month, p.payment_type
```



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

```
SELECT payment_installments,
 COUNT(DISTINCT order_id) AS order_count
FROM `Target_Case_Study.payments`
GROUP BY 1
ORDER BY 1;
```



Actionable Insights and recommendations

The dataset contains the data between the time range of 2016-09-04 21:15

: 19 UTC to 2018-10-17 17:30:18 UTC.

Based on the geolocation city and customer count data the following three cities

Row	geolocation_city	customer_count
1	rio de janeiro	1913913
2	sao paulo	1164470
3	belo horizonte	737556

Have the highest customer foot fall so the marketing strategies can be focused on these three major cities.

As per the analysis 2016 has lower number of orders however there is tremendous growth observed in 2017 which is 43428 and also there is good improvement in 2018 as well and the highest orders were placed in the year 2018 which is 52783.

And, also there is seasonality in the sales observed in the months of August , May, July and March where in August the highest number of orders were placed and September month has the least number of orders . so the Inventory should be planned based on this outcome.

More focus on SP RJ MG states has contributed the most.

Target should focus on providing offers to the people who are making payments in 0,1,2,3 instalments has they make the majority of the transacϴons -

Offers and easy pay opϴons using credit cards and UPI will help the customers

Focus on the area where average delivery is higher than 20 days.

Focus on more products to the areas where there is a fastest delivery which will increase the sales.