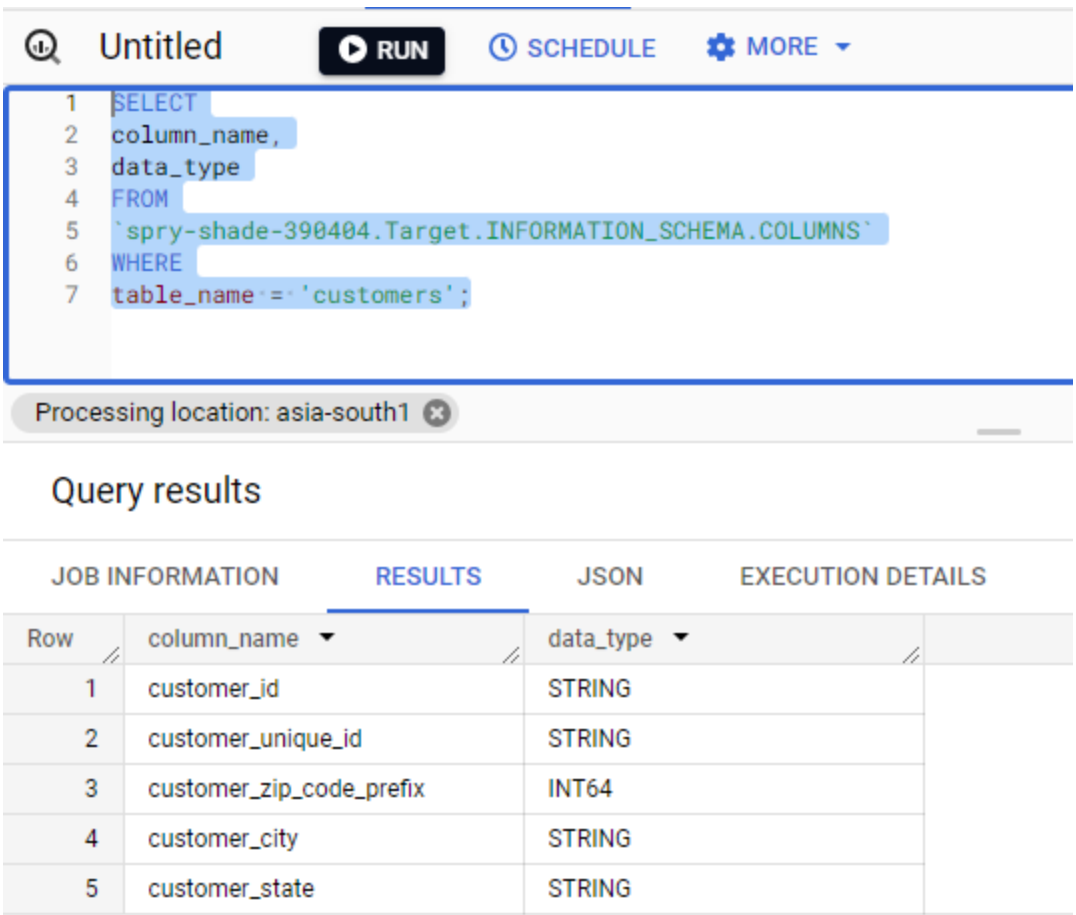
1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
1: Data type of all columns in the "customers" table.

Query:

```
SELECT
column_name,
data_type
FROM
`spry-shade-390404.Target.INFORMATION_SCHEMA.COLUMNS`
WHERE
table_name = 'customers';
```

SS:



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

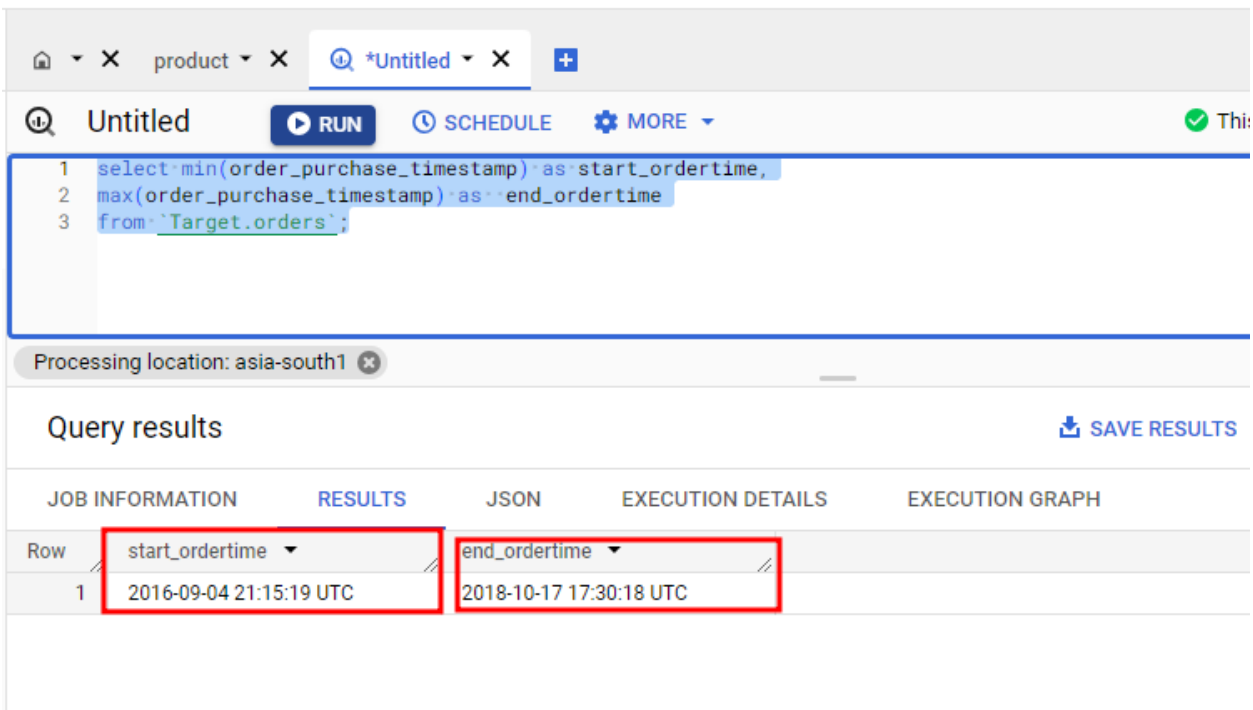
INSIGHT: It helps us to understand the structure and characteristics of the data type and columns stored in the provided "customers" table.

RECOMMENDATION: Based on the above query it is helpful to validate the data type in the particular column. For example numeric data should be as **INTERGER/FLOAT** and text as **VARCHAR/STRING**.

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

```
select min(order_purchase_timestamp) as start_ordertime,
max(order_purchase_timestamp) as  end_ordertime
from `Target.orders`;
```

SS:



INSIGHT: This SQL query retrieves the start time & end time where the order placed in the provided time period by using minimum and maximum values of the "order_purchase_timestamp" column from the "orders" table.

RECOMMENDATION: Time-series analysis can help identify seasonal fluctuations, peak periods, or any significant changes in customer behavior. Also helps

marketing strategies, and overall business planning.

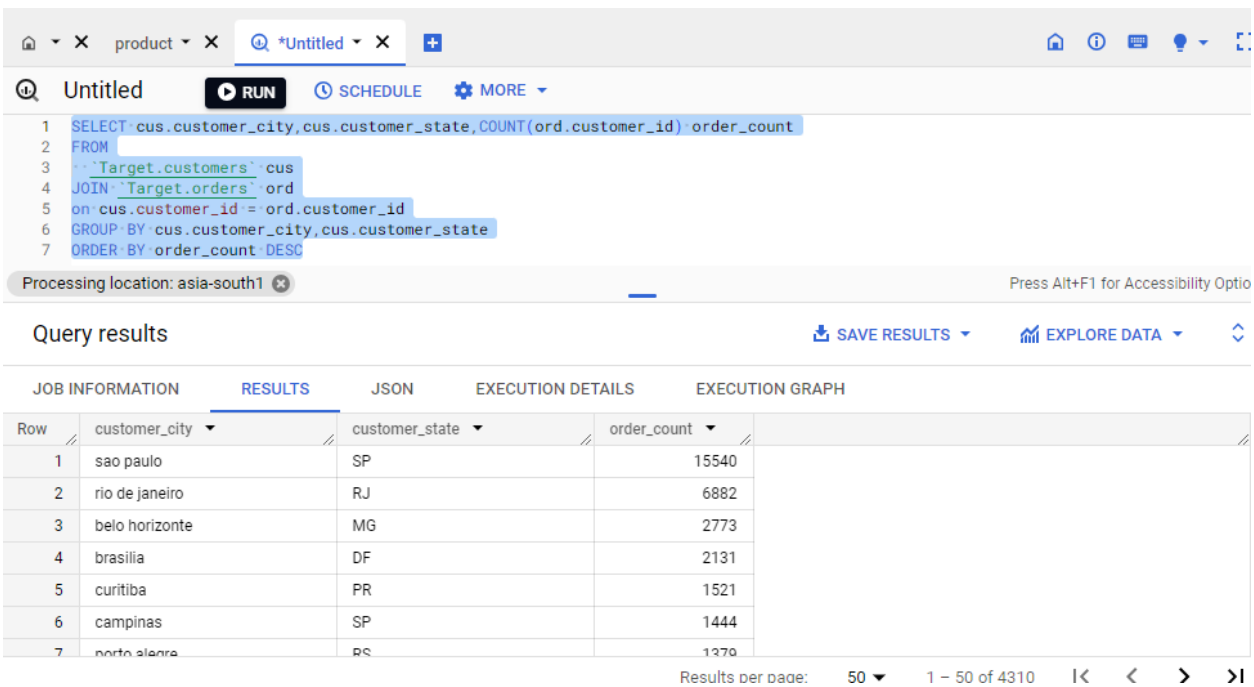
ASSUMPTION: Assumption made here is that the SQL query provided is correctly the max.min order purchase time stamp**.**

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

Query:

```
SELECT cus.customer_city,cus.customer_state,COUNT(ord.customer_id) order_count
FROM
  `Target.customers` cus
JOIN `Target.orders` ord
on cus.customer_id = ord.customer_id
GROUP BY cus.customer_city,cus.customer_state
ORDER BY order_count DESC
```

SS:



Row	customer_city	customer_state	order_count
1	sao paulo	SP	15540
2	rio de janeiro	RJ	6882
3	belo horizonte	MG	2773
4	brasilia	DF	2131
5	curitiba	PR	1521
6	campinas	SP	1444
7	porto alegre	RS	1379

INSIGHT: It helps us to understand the distribution of count of orders across different cities and states were done in the provided given period.

RECOMMENDATION: This analysis can provide the valuable insights like where the most of orders comes from particular cities or states in the regions, by using this we can target accordingly in future plans or strategies and we can easily understand where the company has a significant customer base and sales are most concentrated.

ASSUMPTION: Assumption made here is that the state **SP** have received more counts of orders from the customers and followed by the states **RJ,MG,DF**

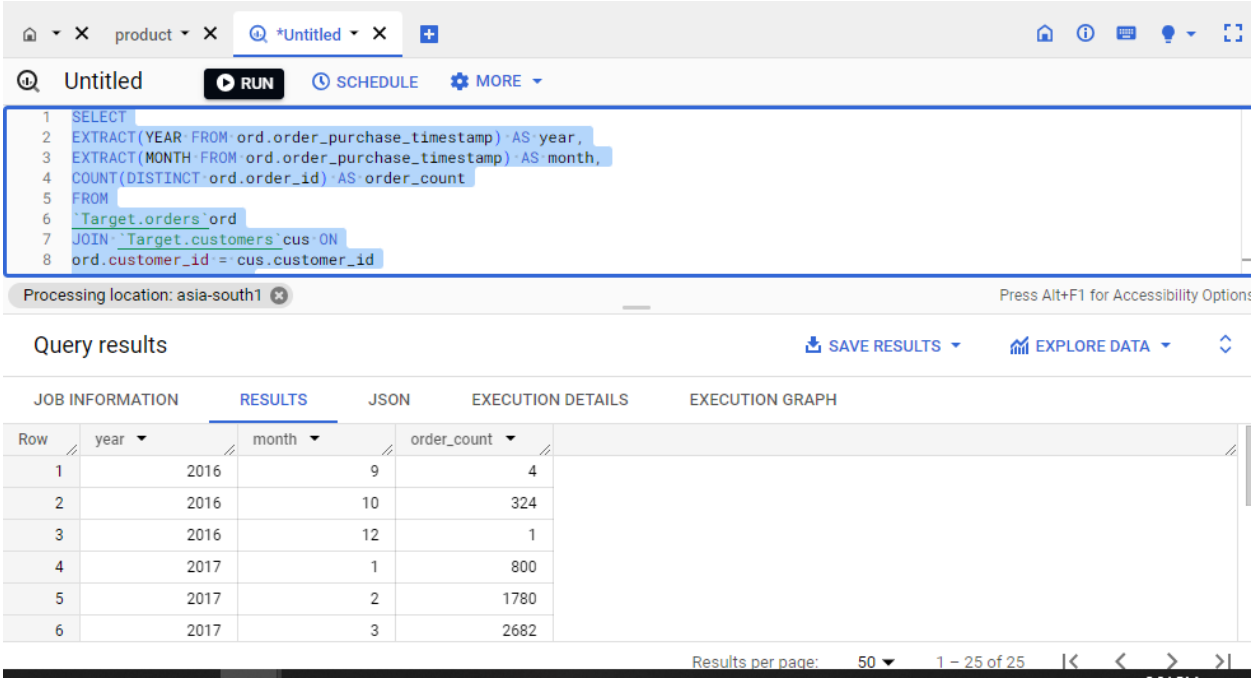
2: In-depth Exploration:

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

Query:

```
SELECT
EXTRACT(YEAR FROM ord.order_purchase_timestamp) AS year,
EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS month,
COUNT(DISTINCT ord.order_id) AS order_count
FROM
`Target.orders`ord
JOIN `Target.customers`cus ON
ord.customer_id = cus.customer_id
GROUP BY year,month
ORDER BY year,month
```

SS:



INSIGHT: This SQL query helps us to understand the trends of orders placed in each month and year, by using we can observe the increasing trends in the year over year.

RECOMMENDATION: Identifying a growing trend that indicate the company's increasing popularity and potential for future growth, and good signs for us.

ASSUMPTION: Assumption made here is that based on the output there is increase on trending growth based on 2016/2017. So, it leads to give us a conclusion that we created a good customer base

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

Query:

```
SELECT
extract(month from order_purchase_timestamp) as Month,
COUNT(order_id) as Total_Orders_in_Month
FROM `Target.orders`
GROUP BY Month
order by Month;
```

SS:

product *Untitled

Untitled RUN SCHEDULE MORE

```sql
SELECT
extract(month from order_purchase_timestamp) as Month,
COUNT(order_id) as Total_Orders_in_Month
FROM `Target.orders`
GROUP BY Month
order by Month;
```

Processing location: asia-south1

Query results

JOB INFORMATION RESULTS JSON EXECUTION DETAILS

Row	Month	Total_Orders_in_Mon
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318

INSIGHT: It helps us to calculates the count of orders placed in each month, allowing us to observe the potential of each and every month and seasonality in the number of orders in the respective months

RECOMMENDATION: It to visualize the data using seasonality in the number of orders over the months and we can conclude like in which month we receive more number of orders by using the we can target that specific month with huge budget for marketing, it give us a good result.

ASSUMPTION: Assumption made here is during that month **May, July, August** month the orders are getting increased due to some seasonality change on **BRAZIL**

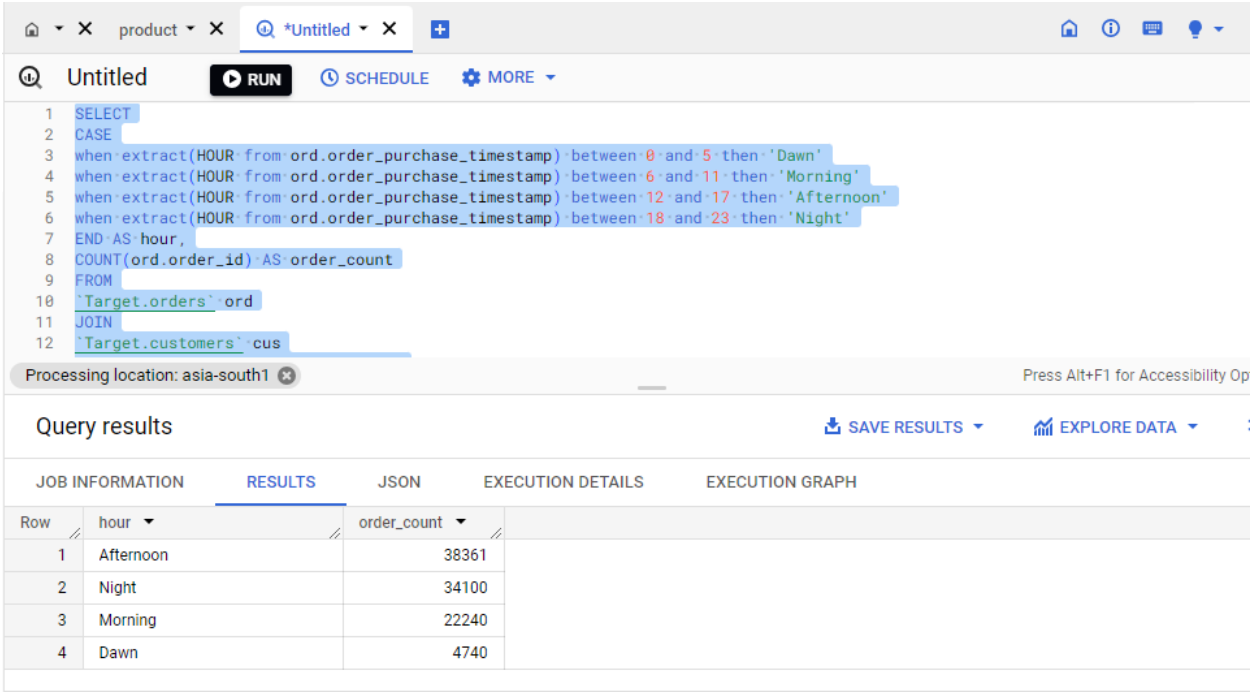
3.During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

1. **0-6 hrs : Dawn**
2. **7-12 hrs : Mornings**
3. **13-18 hrs : Afternoon**
4. **19-23 hrs : Night**

Query:

```
SELECT
CASE
when extract(HOUR from ord.order_purchase_timestamp) between 0 and 5 then
'Dawn'
when extract(HOUR from ord.order_purchase_timestamp) between 6 and 11
then 'Morning'
when extract(HOUR from ord.order_purchase_timestamp) between 12 and 17
then 'Afternoon'
when extract(HOUR from ord.order_purchase_timestamp) between 18 and 23
then 'Night'
END AS hour,
COUNT(ord.order_id) AS order_count
FROM
`Target.orders` ord
JOIN
`Target.customers` cus
ON ord.customer_id =cus.customer_id
GROUP BY
  hour
ORDER BY
  order_count DESC;
```

SS:



INSIGHT: In this SQL query we categorized the orders placed by Brazilian customers on the time of day they were made their respective orders in Dawn, Morning, Afternoon, or Night

RECOMMENDATION: This analysis can provide valuable insights into the preferred time periods for placing orders by Brazilian customers. So it will helps us to find the peak time periods and we can target those peak times with special offers, discounts and increase 'n' number of orders.

ASSUMPTION: Assumption made here is that during afternoon and night the BRAZILAIN people placing more orders

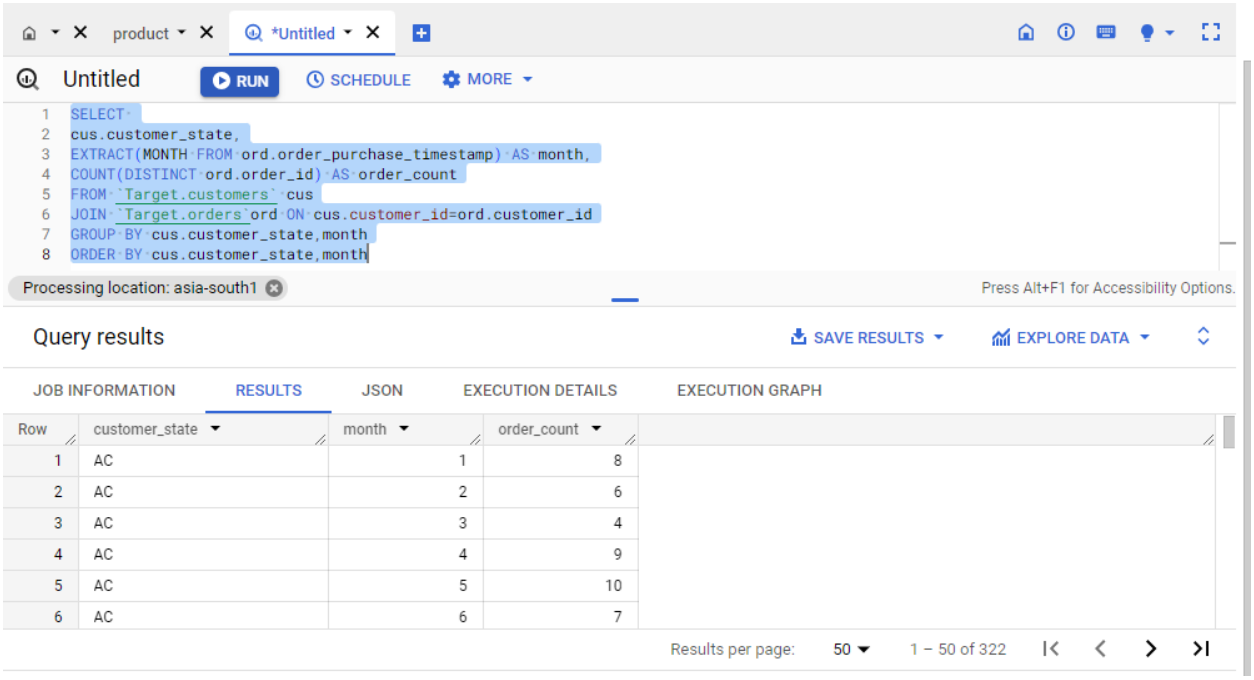
3.Evolution of E-commerce orders in the Brazil region:

1.Get the month on month no. of orders placed in each state.
Query:

```
SELECT
cus.customer_state,
EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS month,
COUNT(DISTINCT ord.order_id) AS order_count
```

```
FROM `Target.customers` cus
JOIN `Target.orders`ord ON cus.customer_id=ord.customer_id
GROUP BY cus.customer_state,month
ORDER BY cus.customer_state,month
```

SS:



INSIGHT: This SQL query calculates the count of orders placed in each state of the Brazil customers on each & every month.

RECOMMENDATION: This one gives us the insights called which month & from which state we were into the growth phase or down phase of our orders and then we can easily either increase the market budgets for the down fall one or completely exclude those.

ASSUMPTION: It helps us to understand when and where the orders made will help to identify the market trend to take decision in marketing strategies.

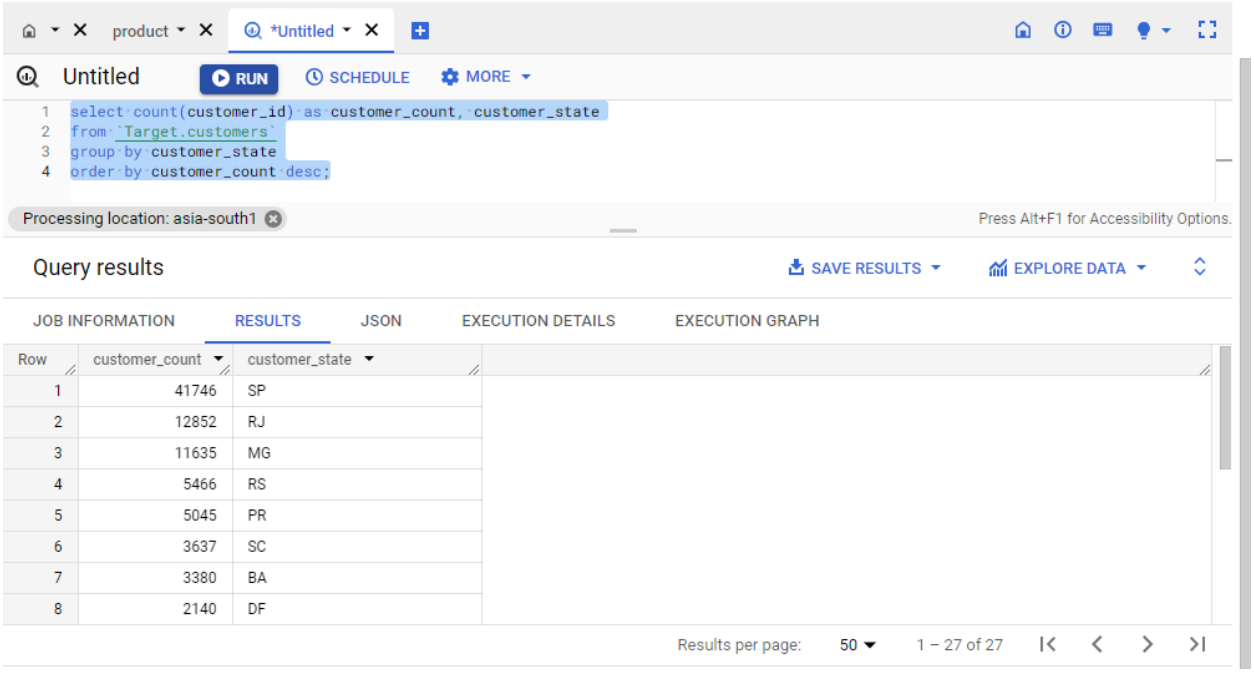
2.How are the customers distributed across all the states?

Query:

```
select count(customer_id) as customer_count, customer_state
from `Target.customers`
```

```
group by customer_state
order by customer_count desc;
```

SS:



Row	customer_count	customer_state
1	41746	SP
2	12852	RJ
3	11635	MG
4	5466	RS
5	5045	PR
6	3637	SC
7	3380	BA
8	2140	DF

INSIGHT: It helps us to identify the count of customers distributed in each states

RECOMMENDATION: This one made the marketing budgets and targeting strategies very easily like where we focus more or where we focus less because of how customers were distributed geographically and identify states with the highest customer strength.

ASSUMPTION: It clearly gives that **SP, RJ and MG** have the most number of customers.

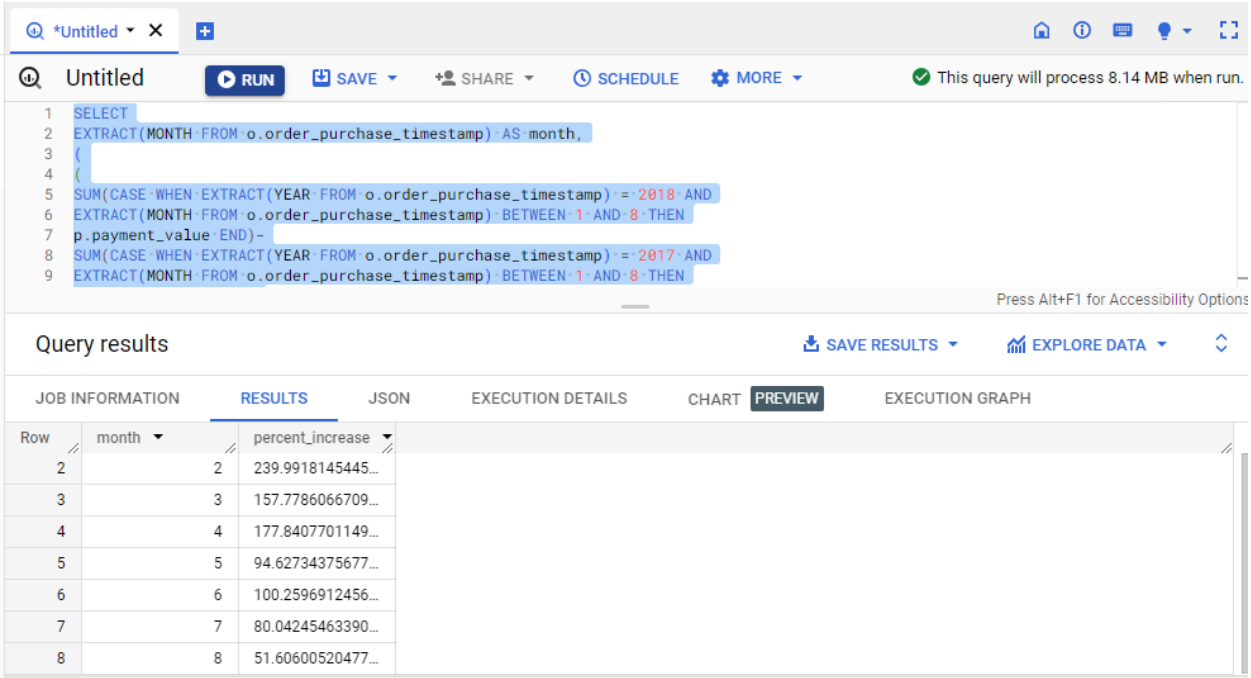
4. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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.

Query:

```
SELECT
EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
(
(
SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
AND
EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
p.payment_value END)-
SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
AND
EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
p.payment_value END)
)/
SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
AND
EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
p.payment_value END)
)*100 AS percent_increase
FROM
`Target.orders` o
JOIN
`Target.payments` p ON o.order_id = p.order_id
WHERE
EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018) AND
EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY 1
ORDER BY 1;
```

SS:



INSIGHT: This SQL query calculates the percentage of increase in the cost of orders from the year 2017 to 2018, and mainly for the months from January to August.

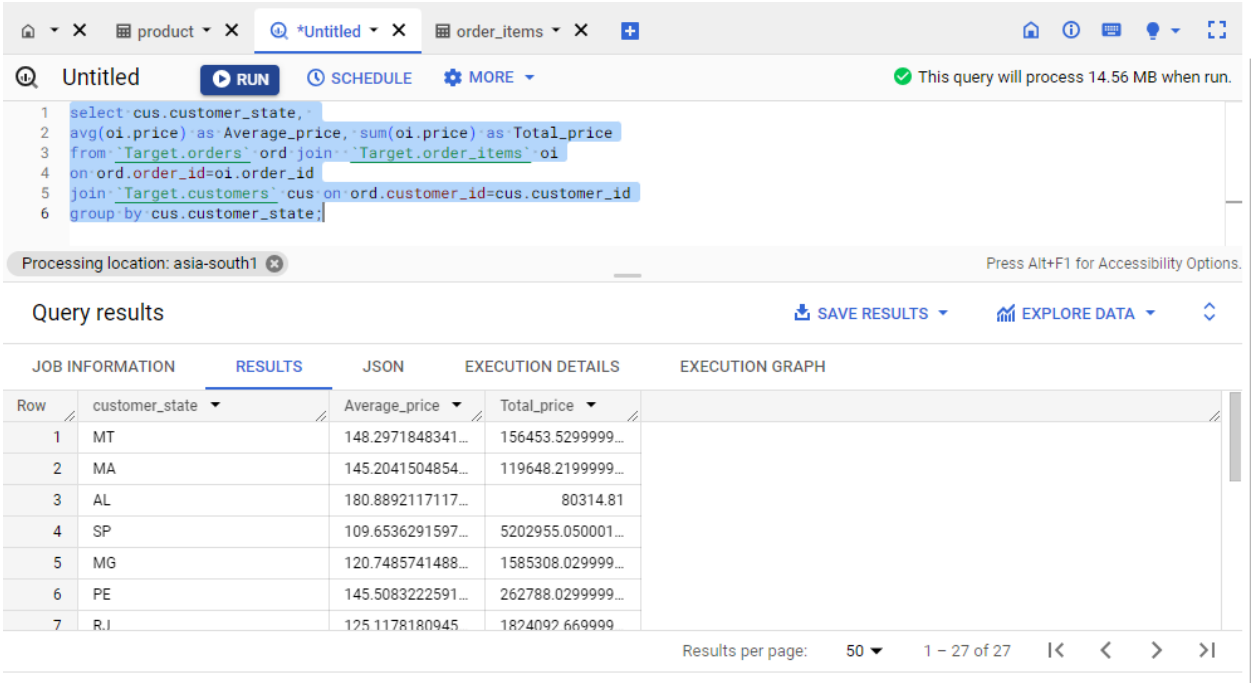
RECOMMENDATION: This recommendation helps us understand the growth rate of order costs during this specific period.

ASSUMPTION: Assumption made here is that compare between 2017 and 2018 there is increase in percentage of cost of orders

2.Calculate the Total & Average value of order price for each state.
Query:

```
select cus.customer_state,
avg(oi.price) as Average_price, sum(oi.price) as Total_price
from `Target.orders` ord join  `Target.order_items` oi
on ord.order_id=oi.order_id
join `Target.customers` cus on ord.customer_id=cus.customer_id
group by cus.customer_state;
```

SS:



INSIGHT: It gives the total and average order prices for each state.

RECOMMENDATION: It easily gives the total order prices and average order prices of thr each and every state and provides a breakdown of the financial performance in each state, offering an understanding of the total revenue generated and the average value of orders placed by customers in different states.

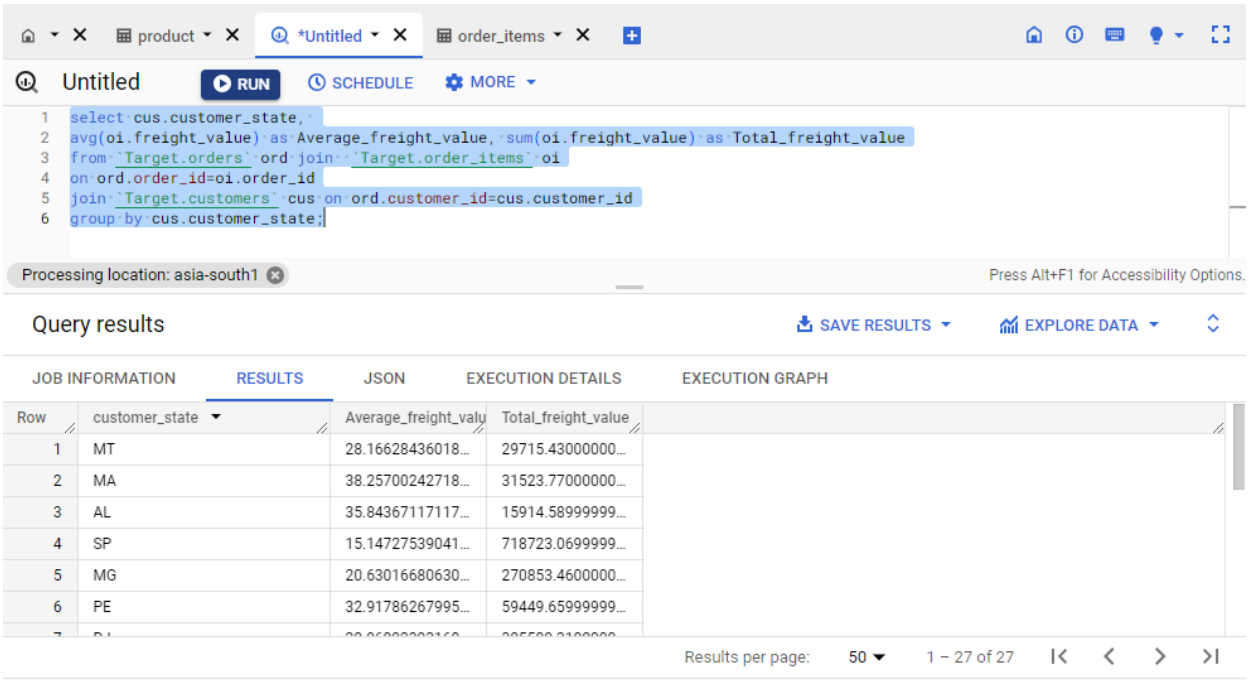
ASSUMPTION: Assumption made here is that we can identify the average customer order in the particular state and if there is any lower state, we can come with some business strategies.

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

Query:

```
select cus.customer_state,
avg(oi.freight_value) as Average_freight_value, sum(oi.freight_value) as
Total_freight_value
from `Target.orders` ord join  `Target.order_items` oi
on ord.order_id=oi.order_id
join `Target.customers` cus on ord.customer_id=cus.customer_id
group by cus.customer_state;
```

SS:



INSIGHT: The SQL query calculates the total and average order freight for each state.

RECOMMENDATION: It helps us understand the growth rate of order freight during this specific period.

ASSUMPTION: Assumption made here is that we can identify the average customer freight in the particular state and if there is any lower state, we can come with some business strategies.

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

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.

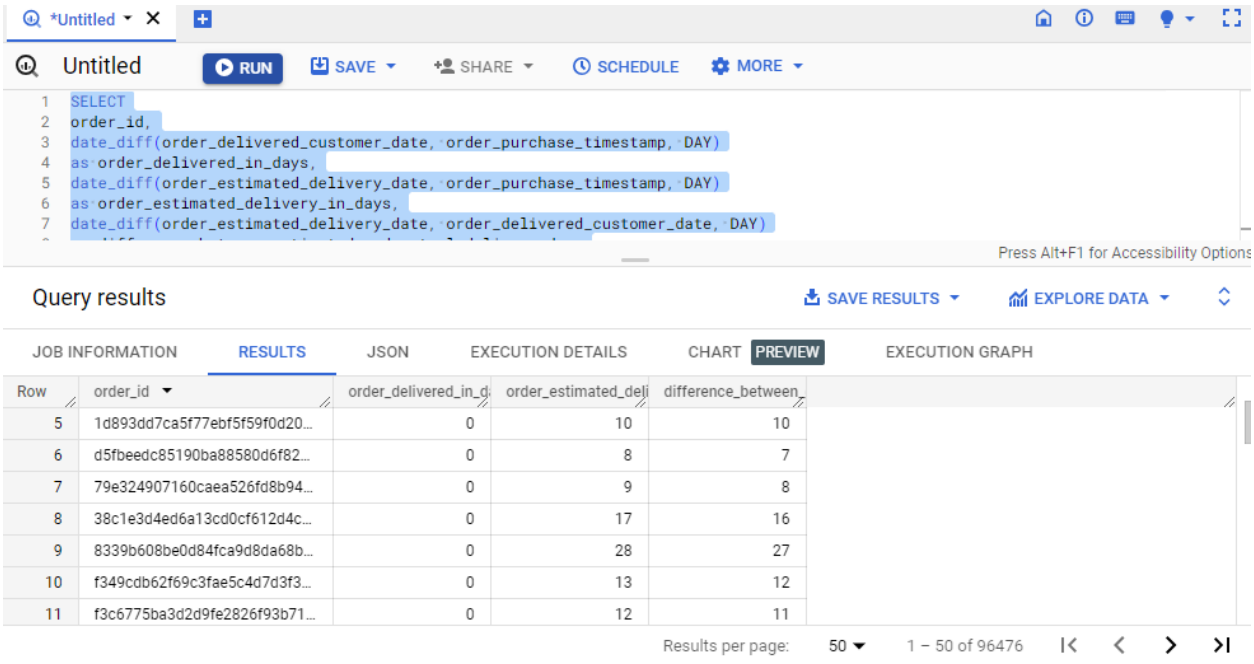
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

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

Query:

```
SELECT
order_id,
date_diff(order_delivered_customer_date, order_purchase_timestamp, DAY)
as order_delivered_in_days,
date_diff(order_estimated_delivery_date, order_purchase_timestamp, DAY)
as order_estimated_delivery_in_days,
date_diff(order_estimated_delivery_date, order_delivered_customer_date, DAY)
as difference_between_estimated_and_actual_delivery_days
FROM
`Target.orders`
WHERE
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) is
NOT NULL
ORDER BY
order_delivered_in_days;
```

SS:



Row	order_id	order_delivered_in_d	order_estimated_deli	difference_between_
5	1d893dd7ca5f77ebf5f59f0d20...	0	10	10
6	d5fbeedc85190ba88580d6f82...	0	8	7
7	79e324907160caea526fd8b94...	0	9	8
8	38c1e3d4ed6a13cd0cf612d4c...	0	17	16
9	8339b608be0d84fca9d8da68b...	0	28	27
10	f349cdb62f69c3fae5c4d7d3f3...	0	13	12
11	f3c6775ba3d2d9fe2826f93b71...	0	12	11

INSIGHT: The provided SQL query calculates the delivery time for each and every order, that is the number of days taken to deliver the order from the purchased date

RECOMMENDATION: It will gives us the delivery time of the order, so in future we can improve delivery time on the cities/states where we lag. So it gives the good impact on our customers.

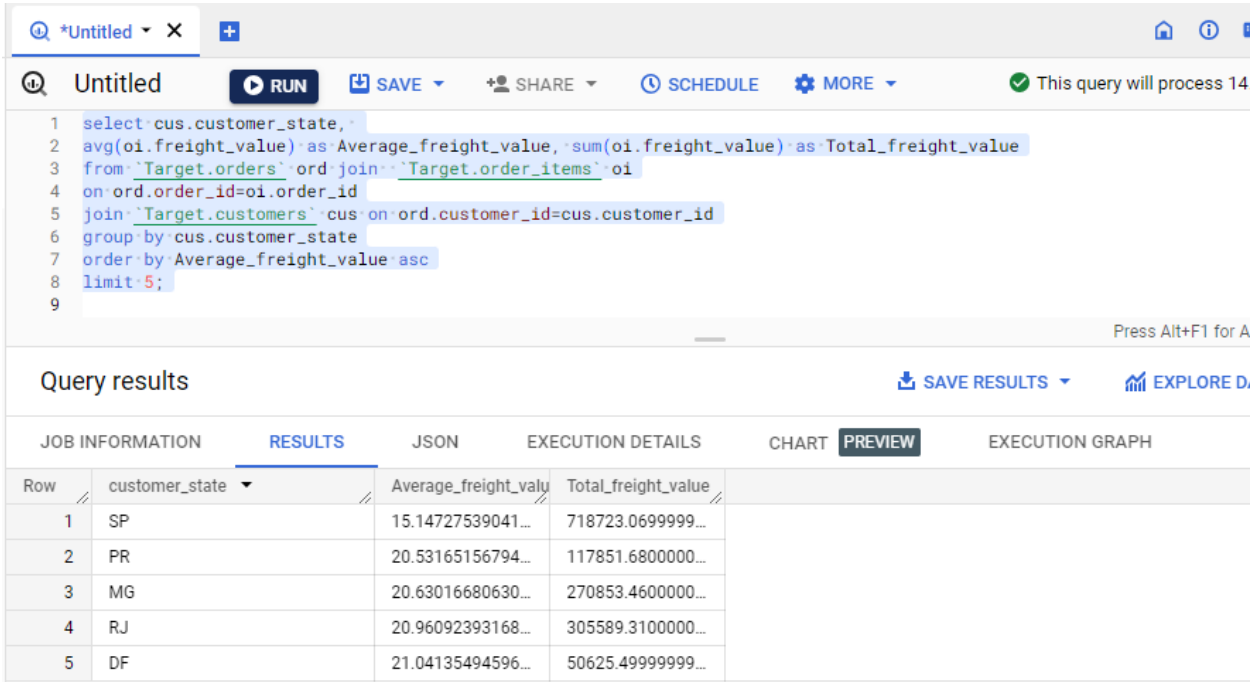
ASSUMPTION: Assumption made here is that we can identify products that are delivered to the customer on time or whether we need to improve the standard deliver to make customer happy.

2.Find out the top 5 states with the highest & lowest average freight value.
Lowest 5:
Query:

```
select cus.customer_state,
avg(oi.freight_value) as Average_freight_value, sum(oi.freight_value) as
Total_freight_value
from `Target.orders` ord join  `Target.order_items` oi
on ord.order_id=oi.order_id
join `Target.customers` cus on ord.customer_id=cus.customer_id
```

```
group by cus.customer_state
order by Average_freight_value asc
limit 5;
```

SS:



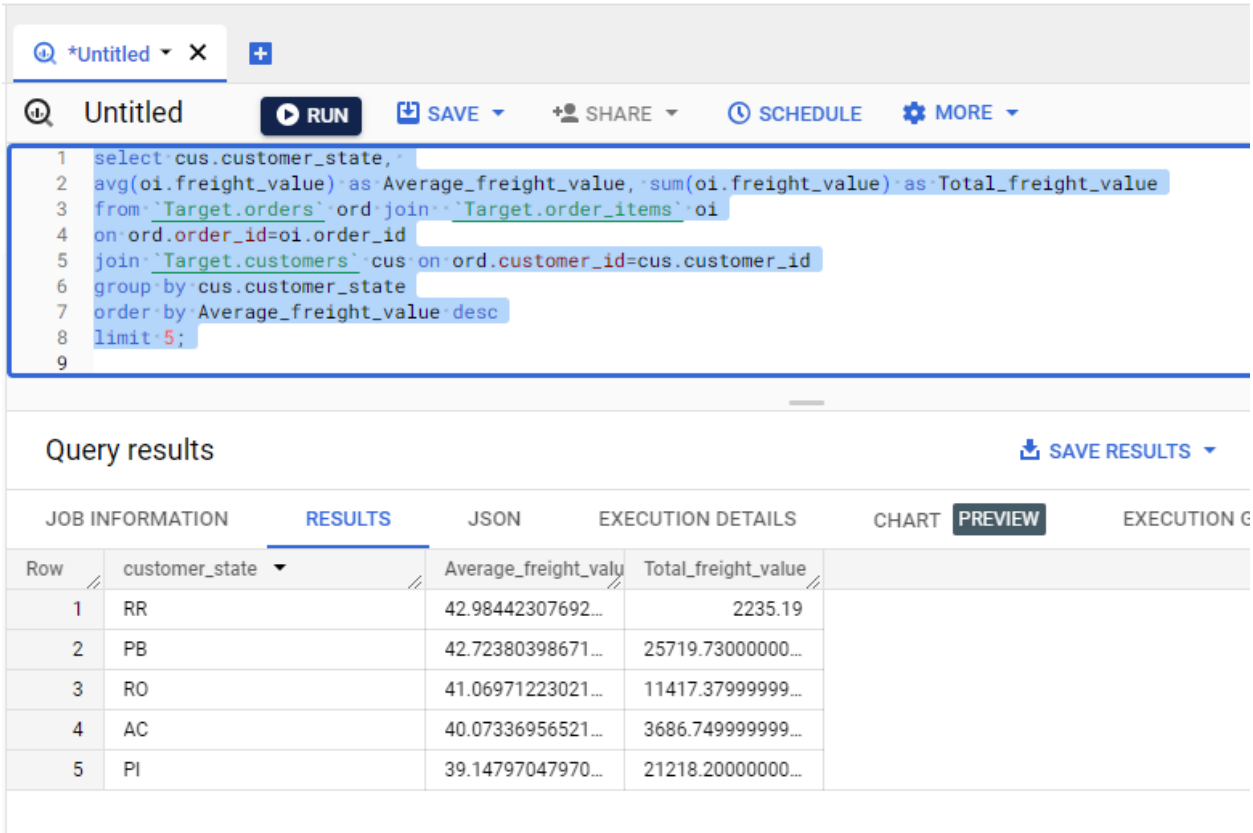
Row	customer_state	Average_freight_valu	Total_freight_value
1	SP	15.14727539041...	718723.0699999...
2	PR	20.53165156794...	117851.6800000...
3	MG	20.63016680630...	270853.4600000...
4	RJ	20.96092393168...	305589.3100000...
5	DF	21.04135494596...	50625.49999999...

Highest 5:

Query:

```
select cus.customer_state,
avg(oi.freight_value) as Average_freight_value, sum(oi.freight_value) as
Total_freight_value
from `Target.orders` ord join  `Target.order_items` oi
on ord.order_id=oi.order_id
join `Target.customers` cus on ord.customer_id=cus.customer_id
group by cus.customer_state
order by Average_freight_value desc
limit 5;
```

SS:



INSIGHT: The query calculates the average and total freight values for each state and presents the top 5 states with the highest and lowest average freight values.

RECOMMENDATION: In order to this we need to focus more on the lowest average freight values of the states to increase our customer base.

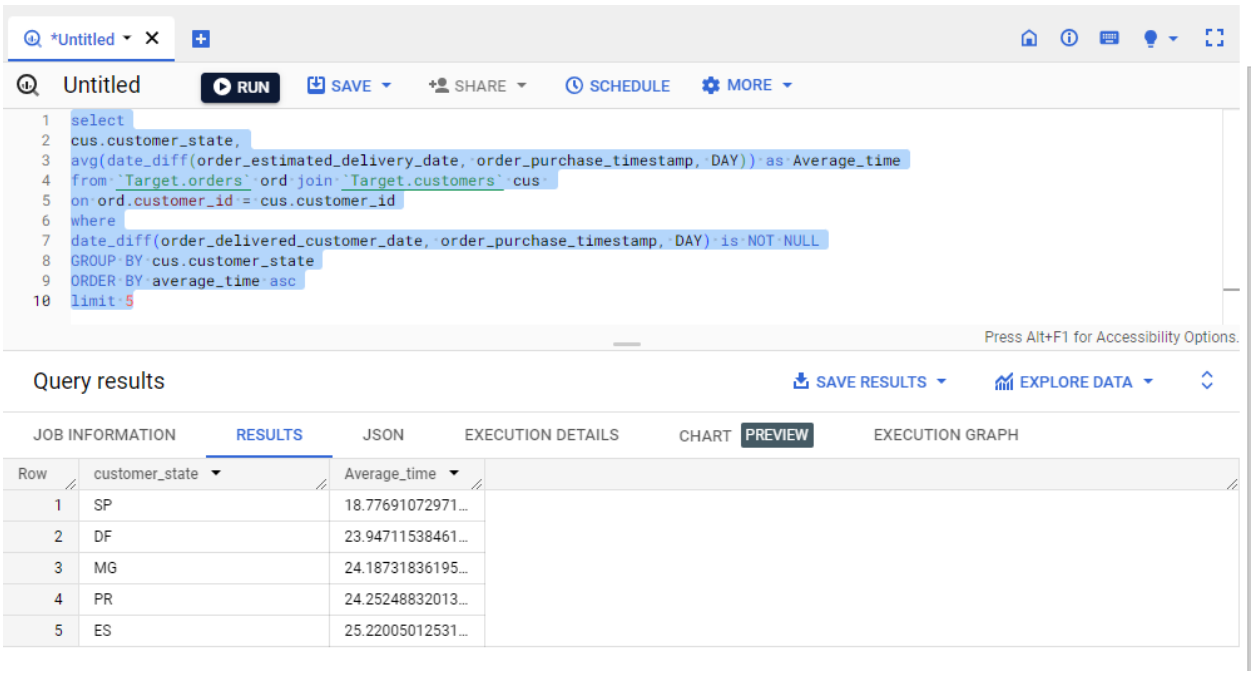
ASSUMPTION: Assumption made here is that we can identify the highest freight on the each state to decision in business level.

3.Find out the top 5 states with the highest & lowest average delivery time.
Lowest:
Query:

```
select
cus.customer_state,
avg(date_diff(order_estimated_delivery_date, order_purchase_timestamp, DAY))
as Average_time
from `Target.orders` ord join `Target.customers` cus
```

```
on ord.customer_id = cus.customer_id
where
date_diff(order_delivered_customer_date, order_purchase_timestamp, DAY) is
NOT NULL
GROUP BY cus.customer_state
ORDER BY average_time asc
limit 5
```

SS:



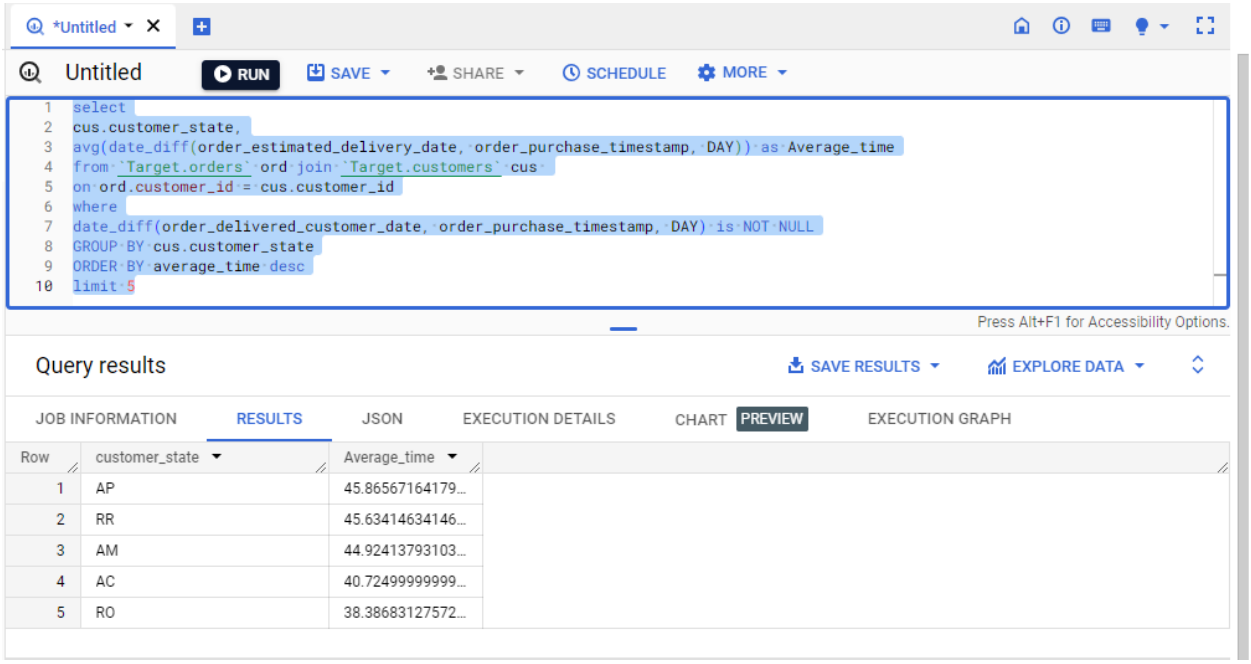
Highest:

Query:

```
select
cus.customer_state,
avg(date_diff(order_estimated_delivery_date, order_purchase_timestamp, DAY))
as Average_time
from `Target.orders` ord join `Target.customers` cus
on ord.customer_id = cus.customer_id
where
date_diff(order_delivered_customer_date, order_purchase_timestamp, DAY) is
NOT NULL
GROUP BY cus.customer_state
ORDER BY average_time desc
```

limit 5

SS:



INSIGHT: It calculates the average and total freight values for each state and presents the top 5 states with the highest and lowest average delivery time.

RECOMMENDATION: It helps us to understand on which state our order delivery time takes more days to deliver to the customers, take some steps to reduce the delivery time

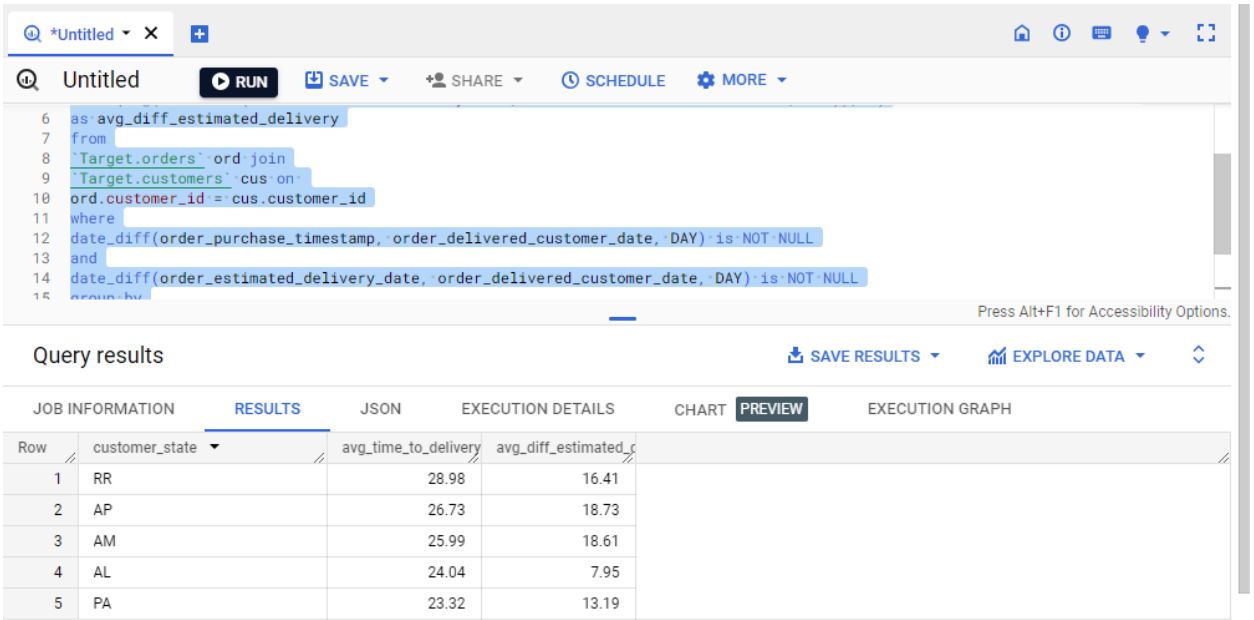
ASSUMPTION: Assumption made here is that we can identify the quickest delivery time on the each state to take decision in business level to improve the same.

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.
Query:

select

```
cus.customer_state,
round(avg(date_diff(order_delivered_customer_date,
order_purchase_timestamp, DAY)), 2)
as avg_time_to_delivery,
round(avg(date_diff(order_estimated_delivery_date,
order_delivered_customer_date, DAY)), 2)
as avg_diff_estimated_delivery
from
`Target.orders` ord join
`Target.customers` cus on
ord.customer_id = cus.customer_id
where
date_diff(order_purchase_timestamp, order_delivered_customer_date, DAY) is
NOT NULL
and
date_diff(order_estimated_delivery_date, order_delivered_customer_date, DAY)
is NOT NULL
group by
cus.customer_state
order by
avg_time_to_delivery desc, avg_diff_estimated_delivery desc
limit 5
```

SS:



INSIGHT: The provided SQL query calculates the average time taken for order delivery and the average difference between the actual and estimated delivery time for each state

RECOMMENDATION: The results identify the top 5 states where the order delivery is notably faster than the estimated delivery date. So we need to deliver them still more fast will give the big impact on our customers and also increase the customer base.

ASSUMPTION: Assumption made here is that we can identify conducting customer satisfaction surveys in these states can provide insights into the reasons behind the positive delivery experience and help in identifying areas for further improvement.

6.Analysis based on the payments:

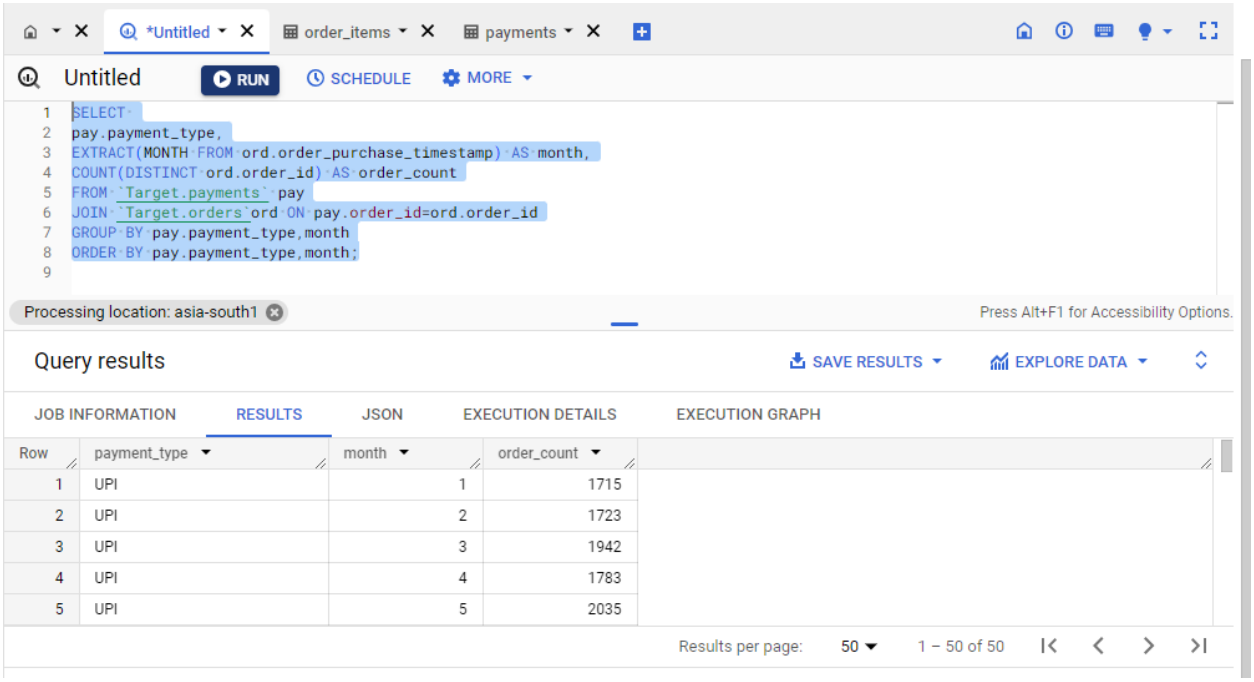
1.Find the month on month no. of orders placed using different payment types.

Query:

```
SELECT
pay.payment_type,
EXTRACT(MONTH FROM ord.order_purchase_timestamp) AS month,
COUNT(DISTINCT ord.order_id) AS order_count
```

```
FROM `Target.payments` pay
JOIN `Target.orders`ord ON pay.order_id=ord.order_id
GROUP BY pay.payment_type,month
ORDER BY pay.payment_type,month;
```

SS:



INSIGHT: The provided SQL query analyzes the number of orders placed using different payment types on a month over month basis.

RECOMMENDATION: By using this information we need to optimize the payment processing methods and also give some offer promotions or incentives for using certain payment types, and improve overall payment options to meet customer preferences.

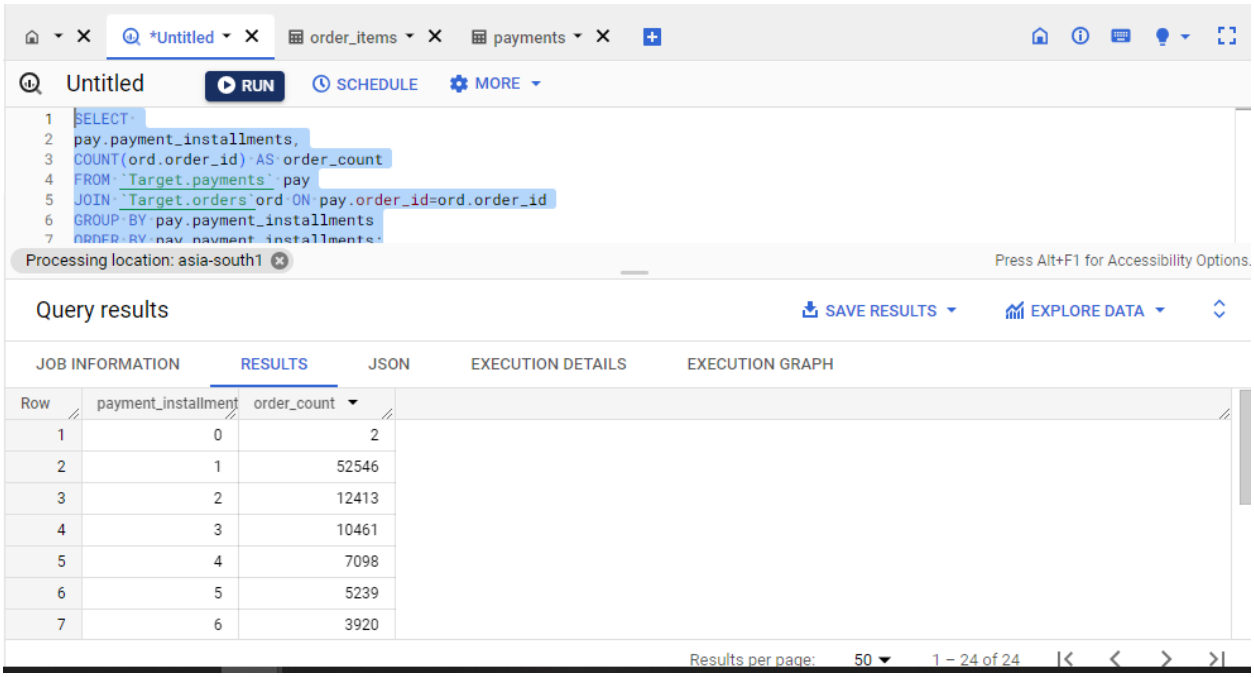
ASSUMPTION: Assumption made here is that we can identify the customer with the payment mode normally used to purchase the products, mostly UPI payments are followed.

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

Query:

```sql
SELECT
pay.payment_installments,
COUNT(ord.order_id) AS order_count
FROM `Target.payments` pay
JOIN `Target.orders`ord ON pay.order_id=ord.order_id
GROUP BY pay.payment_installments
ORDER BY pay.payment_installments;
```

SS:



INSIGHT: The provided SQL query analyzes the number of orders placed on the basis of the payment installments method that have been paid

RECOMMENDATION: We can use this information to made their payment options, also provide more flexible installment plans, and better accommodate customer preferences.

ASSUMPTION: Assumption made here is that we can identify the customer are mostly preferred with installments, so that we can provide offers based on the scenario.