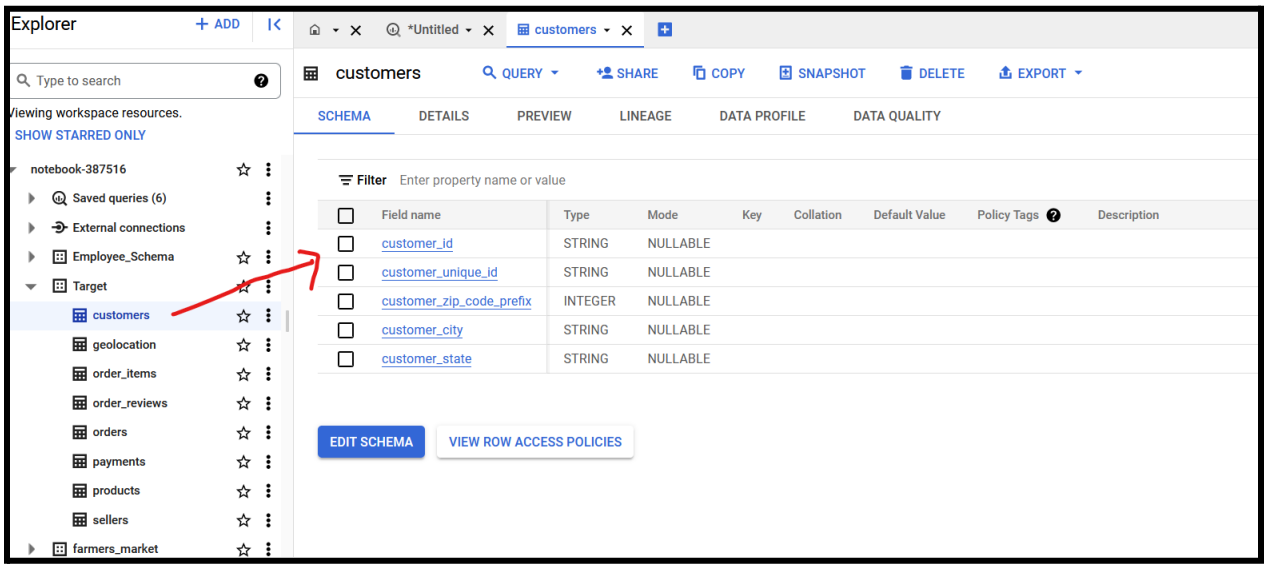
Q.1.a.answer

In order to check the data type of any table using BigQuery is, Double-click on the name of the table which is mentioned in the left side under the dataset ‘target’. And on the right side we can check data types of all columns.



Q.1.b.answer

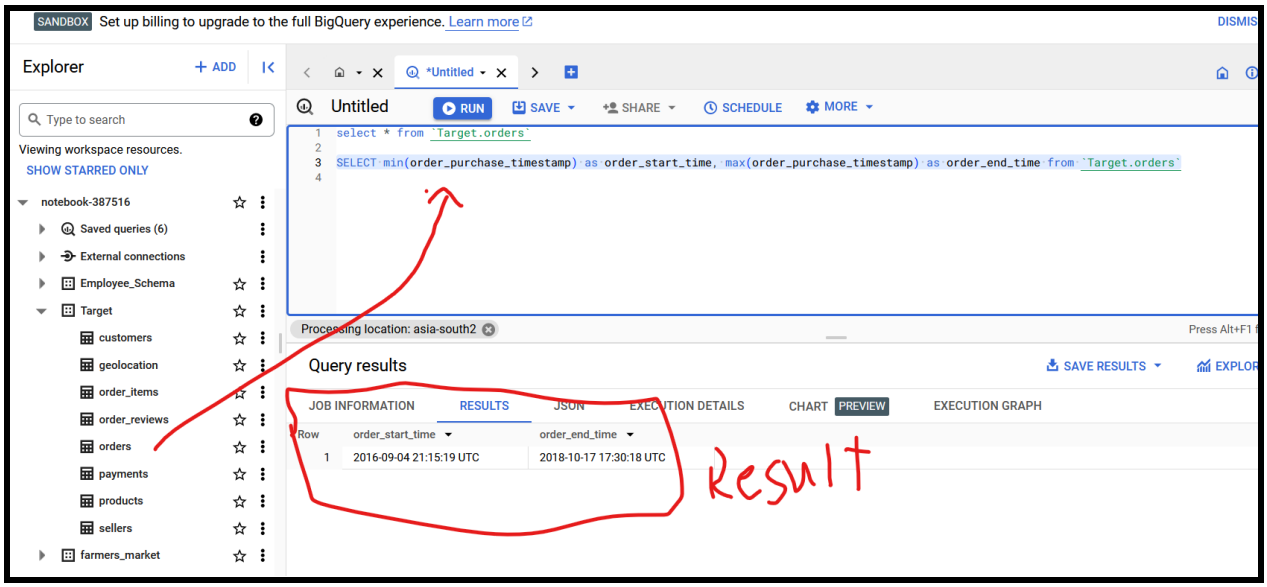
Query

SELECT min(order_purchase_timestamp) as order_start_time,
max(order_purchase_timestamp) as order_end_time from `Target.orders`

Explanation

To find the time range, we will look for the table name 'order' in which we already have 'order_purchase_timestamp' col - time range here means the day and time when the first order was placed and the last day when the order was placed. Using the Avg function Max and Min will give us the time range - the ideal option when the data type is in the format of date/time.

Screenshot of the result -



Q.1.c.answer

Query -

select count(distinct c.customer_city) as city, count(distinct c.customer_state) as state, count(distinct o.order_id) as total_order,

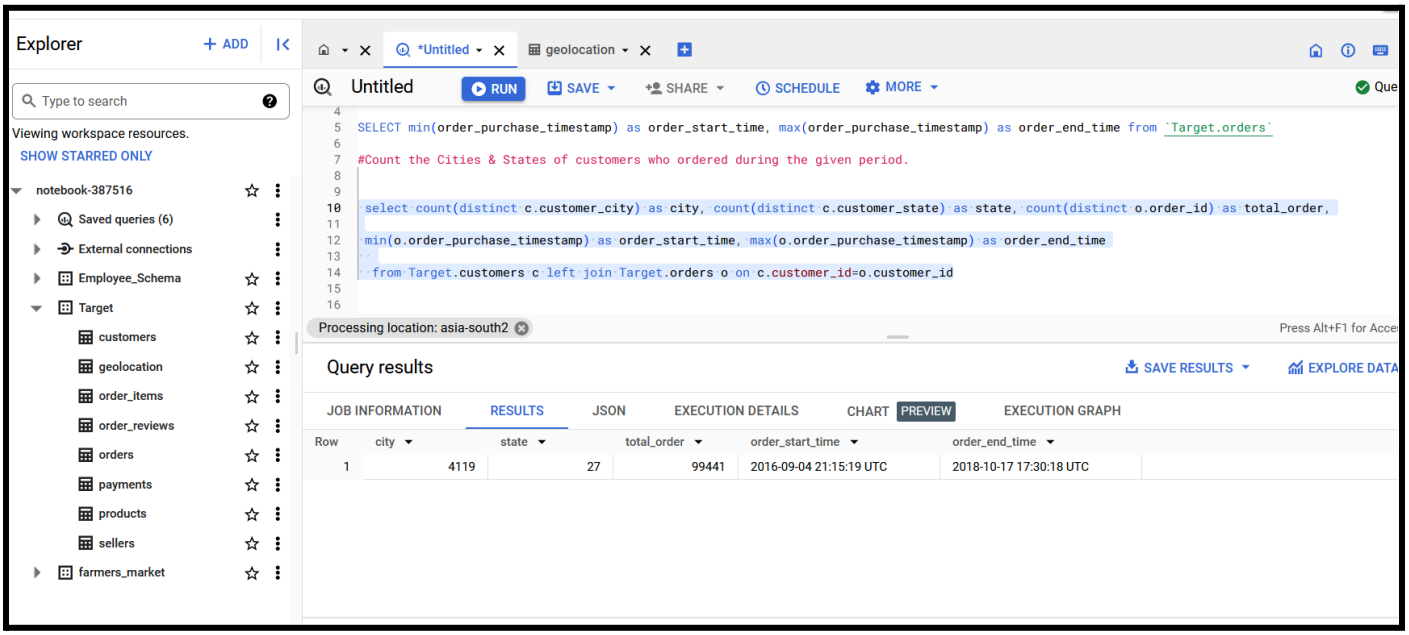
min(o.order_purchase_timestamp) as order_start_time, max(o.order_purchase_timestamp) as order_end_time

from Target.customers c left join Target.orders o on c.customer_id=o.customer_id

Explanation

Here, we have found out all orders placed from total cities and states, in the given time range, which we have solved in question 2 - in this query, we have used aggregate functions like count, min, max, and left join because we have to find out all orders placed by customers.

Screenshot of the result -



Q.2.a.answer

Query

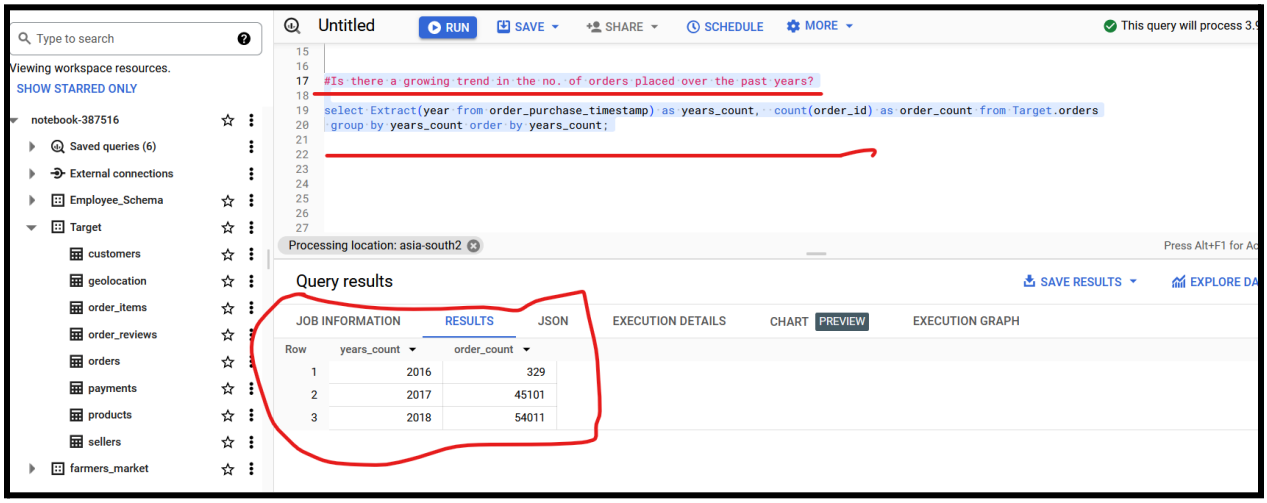
select Extract(year from order_purchase_timestamp) as years_count, count(order_id) as order_count from Target.orders

group by years_count order by years_count;

Explanation

The growing trend here means that with time, there is an increase in total orders. Now that we know that we have to show total orders, now we focus on time, which is in question is in years - means (2016, 2017, 2018,.....n).

Screenshot of results

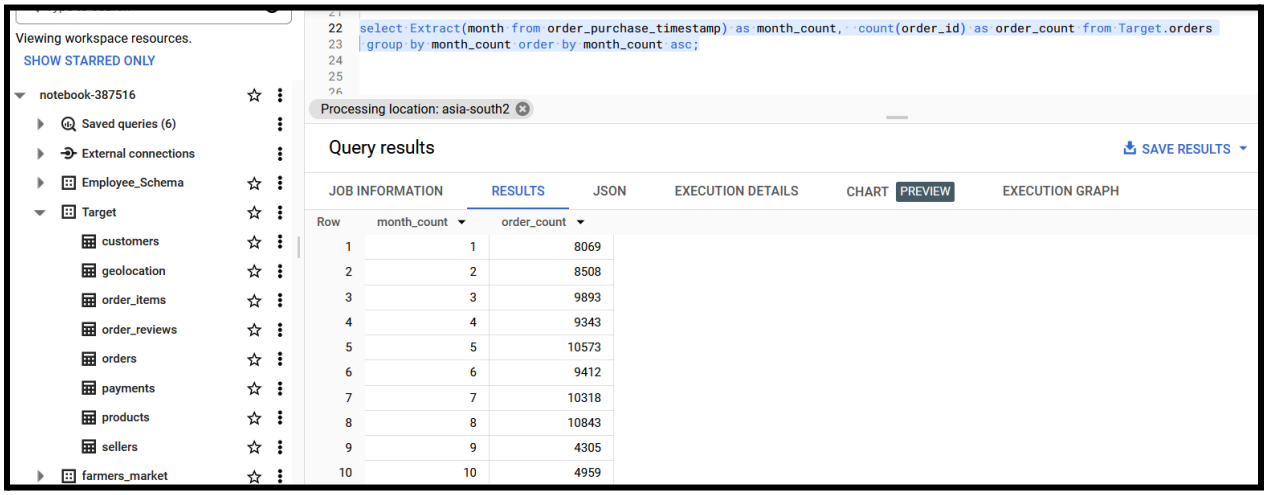


Q.2.b.answer

Query

select Extract(month from order_purchase_timestamp) as month_count, count(order_id) as order_count from Target.orders
 group by month_count order by month_count asc;

Screenshot



Q.2.c.Answer

SELECT order_purchase_timestamp,

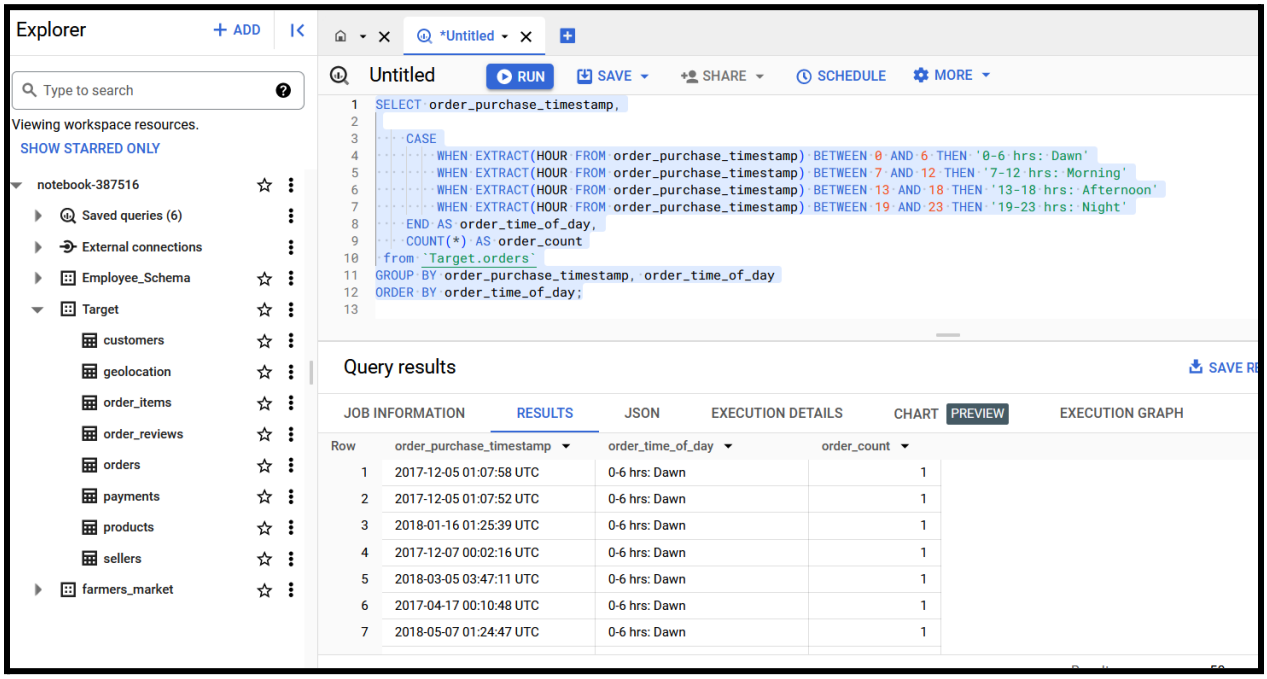
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN '0-6 hrs: Dawn'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN '7-12 hrs: Morning'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN '13-18 hrs: Afternoon'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN '19-23 hrs: Night'
END AS order_time_of_day,
COUNT(*) AS order_count
 from `Target.orders`
GROUP BY order_purchase_timestamp, order_time_of_day
ORDER BY order_time_of_day;

Explanation:

In the question it is clearly asked to segregate the data by range for that we can use the ‘Case’ condition which will segregate the data in 4 section ‘Dawn’, ‘Morning’, ‘Afternoon’ and ‘Night’ also, here we will count total orders placed by the customers for each section segregation this is

where group by roles come into the play. Also we could use the where clause with the country as ‘Brazil’ but we are assuming that the complete data is of “brazil” country.

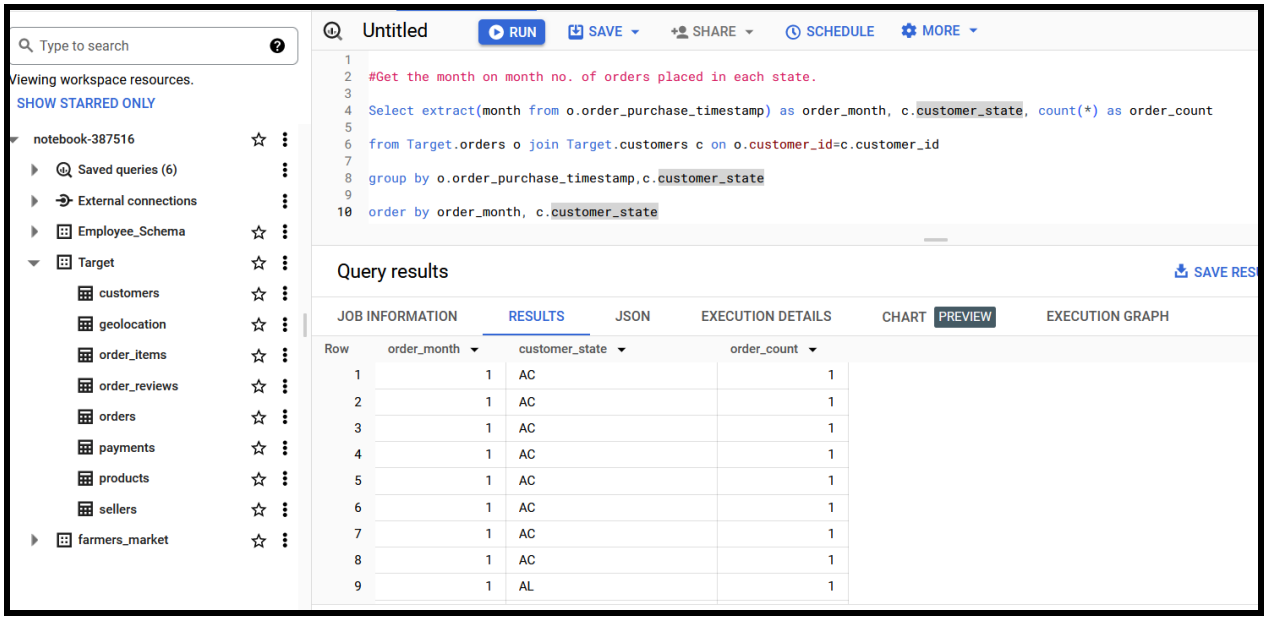
Screenshot:



Question.3.a
Query:
Select extract(month from o.order_purchase_timestamp) as order_month, c.customer_state, count(*) as order_count
from Target.orders o join Target.customers c on o.customer_id=c.customer_id
group by o.order_purchase_timestamp,c.customer_state
order by order_month, c.customer_state

Explanation:
As mentioned, we require month-by-month (1,2,3,4,....n) orders placed for each state. In this, we will use the customer and order table to collect states and total order count using left join. The key point here is to extract data by month, and in the "order_purchased_timestamp." The month is not available separately, so we will extract all months.

Screenshot



Question.3.b

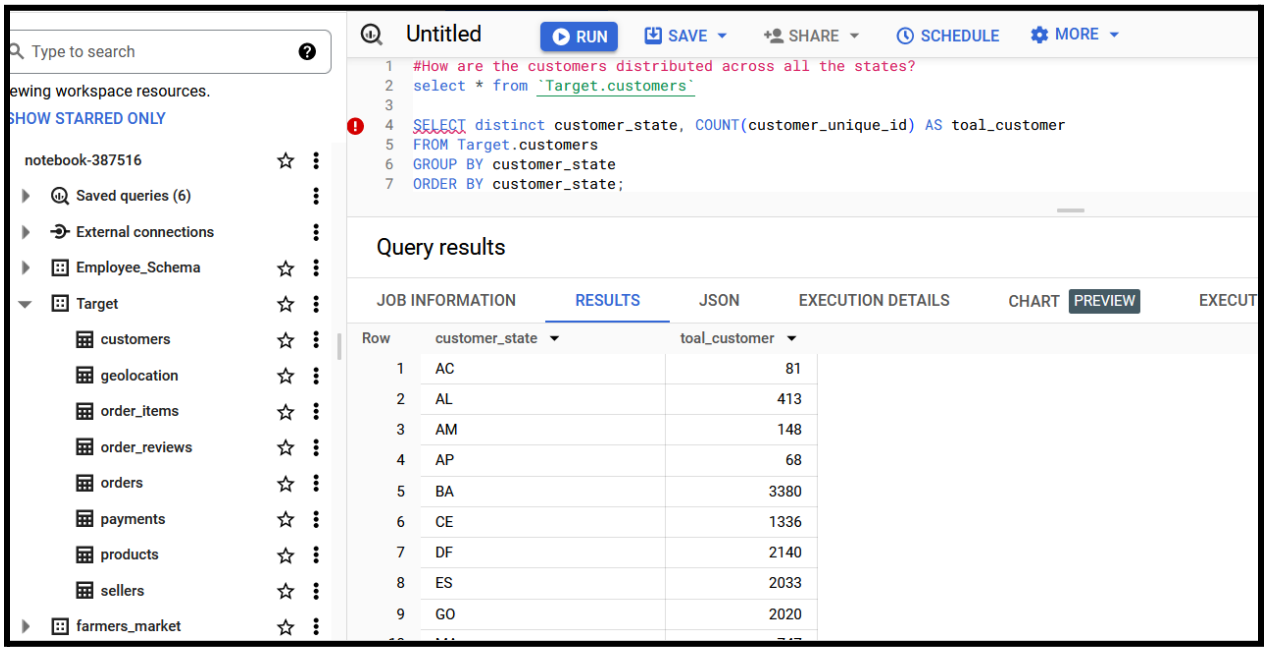
Query

SELECT distinct customer_state, COUNT(customer_unique_id) AS toal_customer
FROM Target.customers
GROUP BY customer_state
ORDER BY customer_state;

Explanation -

Customer distribution across all the states means here we have to find and count total unique customers for each state.

Screenshot



t:

Q.4.answer.a.

Query

```
With MonthlyPayments as (
select
o.order_id,
Extract(year from o.order_purchase_timestamp) as order_year,
Extract(Month from o.order_purchase_timestamp) as order_month,
Sum(P.payment_value) as total_payment
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
o.order_id,
extract (year from o.order_purchase_timestamp),
extract(month from o.order_purchase_timestamp)
)

SELECT
```

```
        (SUM(CASE WHEN order_year = 2018 THEN total_payment ELSE 0 END) -
SUM(CASE WHEN order_year = 2017 THEN total_payment ELSE 0 END))
        / SUM(CASE WHEN order_year = 2017 THEN total_payment ELSE 0 END) * 100 AS
percentage_increase
FROM
        MonthlyPayments
WHERE
        order_year IN (2017, 2018)
GROUP BY
        order_year;


        select * from `Target.orders`
```

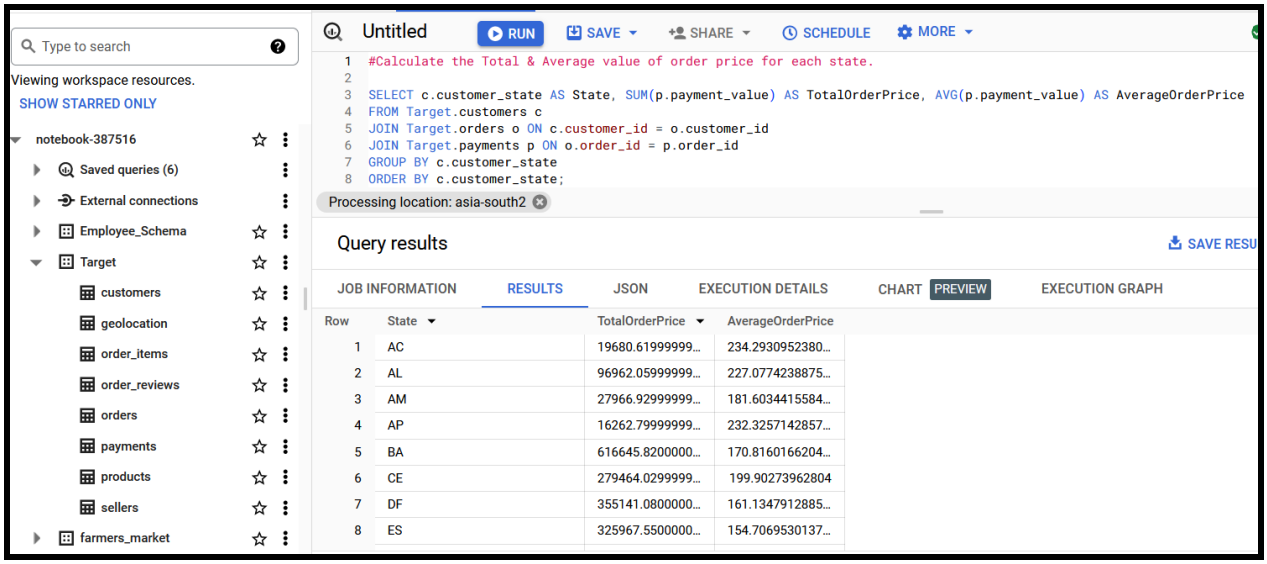
Explanation -
Somehow i managed to write a query by understanding the problem but still not sure -

Q.4.answer.b.
Query

```
SELECT c.customer_state AS State, SUM(p.payment_value) AS TotalOrderPrice,
AVG(p.payment_value) AS AverageOrderPrice
FROM Target.customers c
JOIN Target.orders o ON c.customer_id = o.customer_id
JOIN Target.payments p ON o.order_id = p.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state;
```

Explanation
In this problems there will be three tables will be engaged which are ‘Customers’ from which we will get each state, ‘Orders’ orders table will give us the order_id for which we will find out total sum and avg of the price from the table ‘Payment’

Screenshot



Q.4.Answer.c

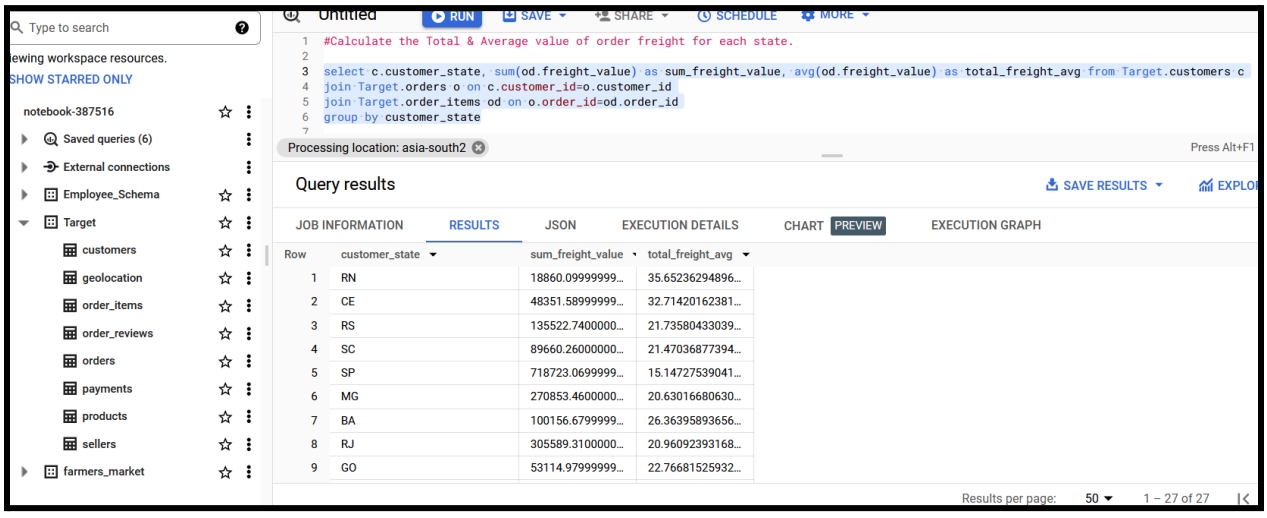
Query

```
select c.customer_state, sum(od.freight_value) as sum_freight_value, avg(od.freight_value) as total_freight_avg from Target.customers c
join Target.orders o on c.customer_id=o.customer_id
join Target.order_items od on o.order_id=od.order_id
group by customer_state
```

Explanation:

Just like the above question, in this problems also, we will use three tables ‘Customers’, ‘Order_items’ and ‘Orders’ - Frieght_value can be calculated using the table order_items although to connect with order_items we will use ‘order_id’ which is a common col for both tables “order_items’ and ‘Orders’ and result for each state ‘Customer table’ we will use group by.

Screenshot

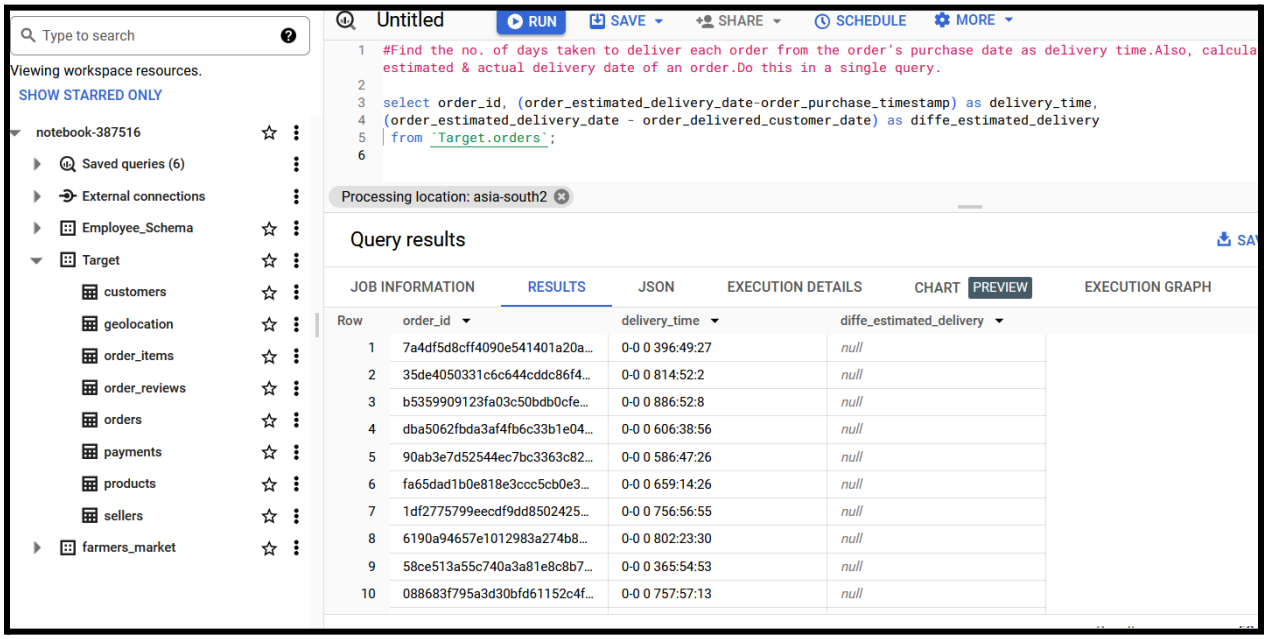


Q.5.answer.1.

Query:

```
select order_id, (order_estimated_delivery_date-order_purchase_timestamp) as
delivery_time,
(order_estimated_delivery_date - order_delivered_customer_date) as
diffe_estimated_delivery
 from `Target.orders`;
```

Screenshot



Q.5.Answer.b.

Query

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

UNION ALL
```

```sql
SELECT c.customer_state,AVG(oi.freight_value) AS avg_freight_value
FROM customers AS c
JOIN orders AS o
ON c.customer_id = o.customer_id
JOIN order_items AS oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY avg_freight_value ASC
LIMIT 5;
```

This query gave me error everytime -

Q.5.c

Query

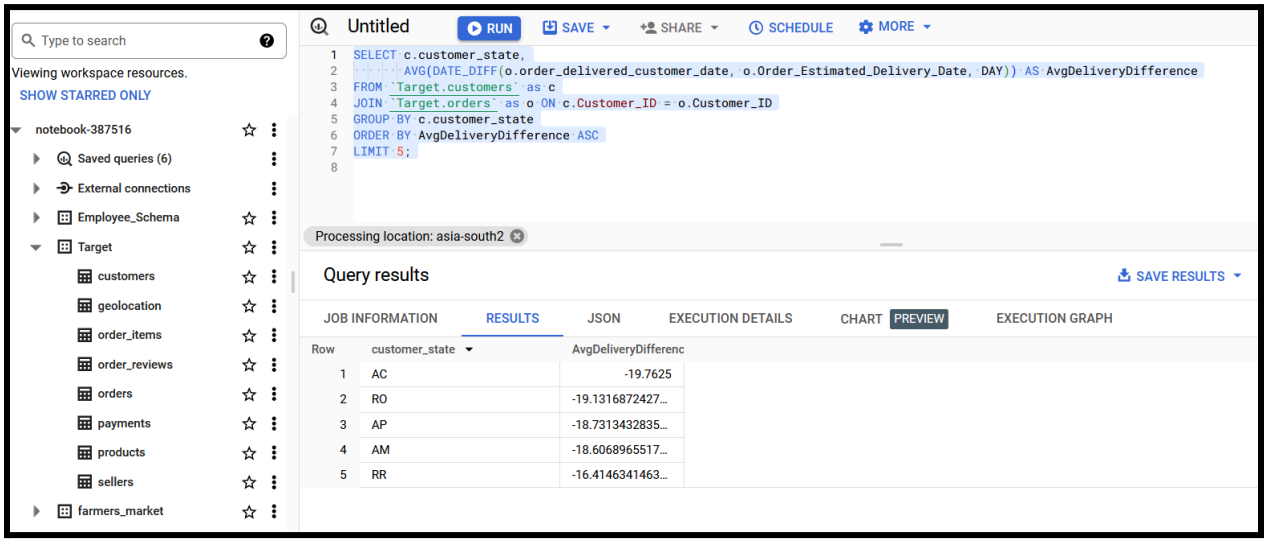
```sql
SELECT
    c.customer_state,
    AVG(DATEDIFF(order_delivered_customer_date, order_purchase_timestamp)) AS
avg_delivery_time
FROM
    customers AS c
JOIN
    orders AS o
ON
    c.customer_id = o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    avg_delivery_time DESC
LIMIT 5

UNION ALL

SELECT
    c.customer_state,
    AVG(DATEDIFF(order_delivered_customer_date, order_purchase_timestamp)) AS
avg_delivery_time
FROM
    customers AS c
JOIN
    orders AS o
ON
```

```
    c.customer_id = o.customer_id
GROUP BY
    c.customer_state
ORDER BY
    avg_delivery_time ASC
LIMIT 5;
```

Q.4.Answer.c.
Query

```
SELECT c.customer_state,
        AVG(DATE_DIFF(o.order_delivered_customer_date, o.Order_Estimated_Delivery_Date,
DAY)) AS AvgDeliveryDifference
FROM `Target.customers` as c
JOIN `Target.orders` as o ON c.Customer_ID = o.Customer_ID
GROUP BY c.customer_state
ORDER BY AvgDeliveryDifference ASC
LIMIT 5;
```

Screenshot



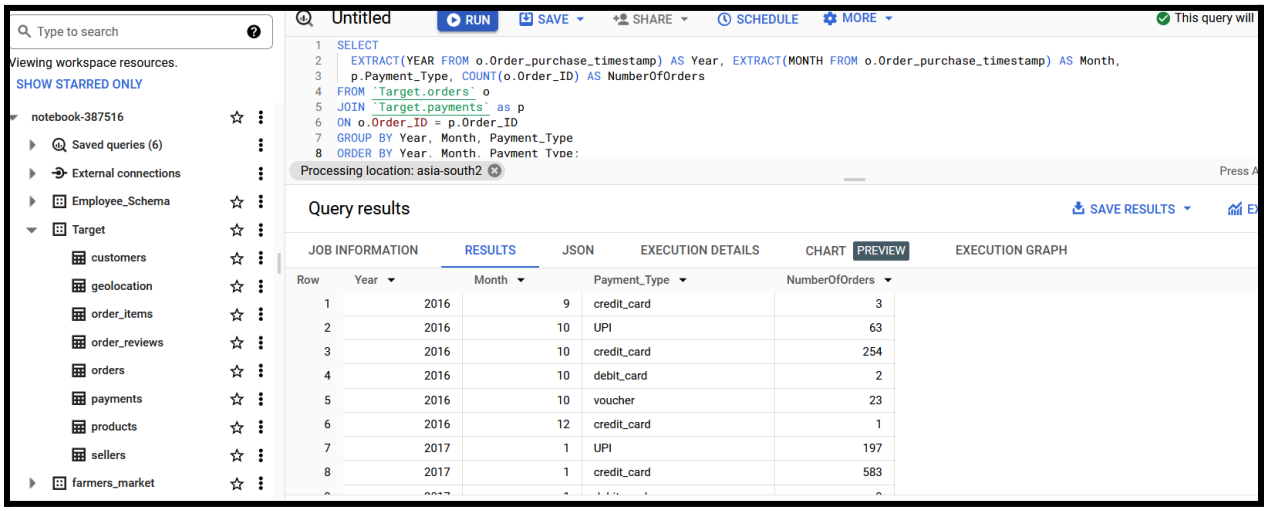
Q.5.Answer.1.
Query

```
SELECT
  EXTRACT(YEAR FROM o.Order_purchase_timestamp) AS Year, EXTRACT(MONTH FROM
o.Order_purchase_timestamp) AS Month,
  p.Payment_Type, COUNT(o.Order_ID) AS NumberOfOrders
FROM `Target.orders` o
JOIN `Target.payments` as p
```

ON o.Order_ID = p.Order_ID
GROUP BY Year, Month, Payment_Type
ORDER BY Year, Month, Payment_Type;

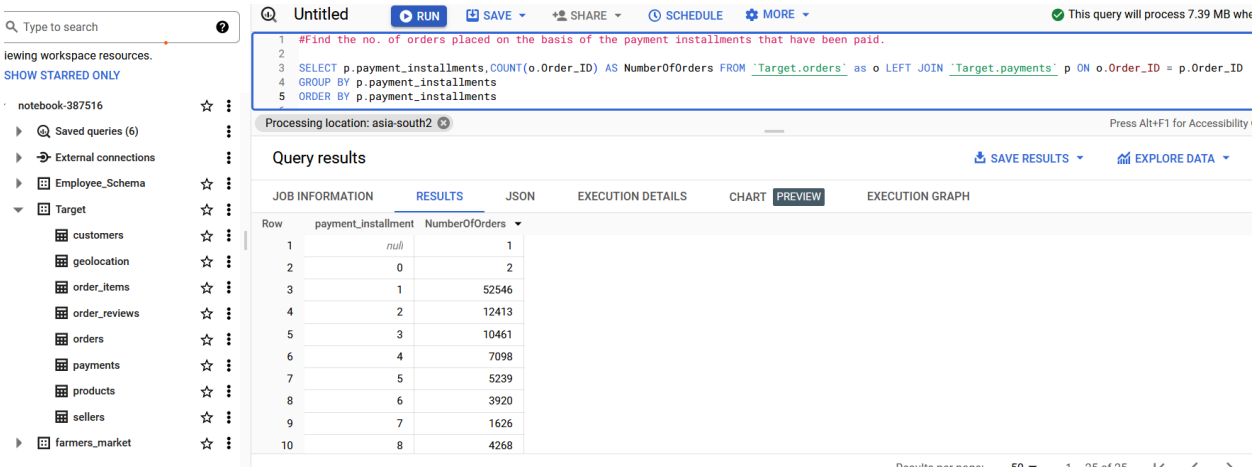
Explanation
As asked in the question, we will use 'extract' years and month from the table orders and in order to find payment type we will use 'Payment' table and use the inner join to connect both tables using 'order_id' because month we have to find on month no. of orders placed.

Screenshot



Question.5.Answer.2
Query
SELECT p.payment_installments,COUNT(o.Order_ID) AS NumberOfOrders FROM `Target.orders` as o LEFT JOIN `Target.payments` p ON o.Order_ID = p.Order_ID
GROUP BY p.payment_installments
ORDER BY p.payment_installments

Screenshot



Valuable Insights

- [ ] Here we have the range of all orders placed and purchased, this customized data can help 'Target' to understand the historical time of business operations. Also, this data will also help understanding in what hours the company should be proactive when it comes to showcase customized products to the users of brazil.

- [ ] From this result, we could do extensive research on how many users from all cities are purchasing and how the purchasing values of users could be enhanced.

- [ ] These timestamps can be used to calculate various performance metrics, such as the average order frequency or the total number of orders within the specified time frame.