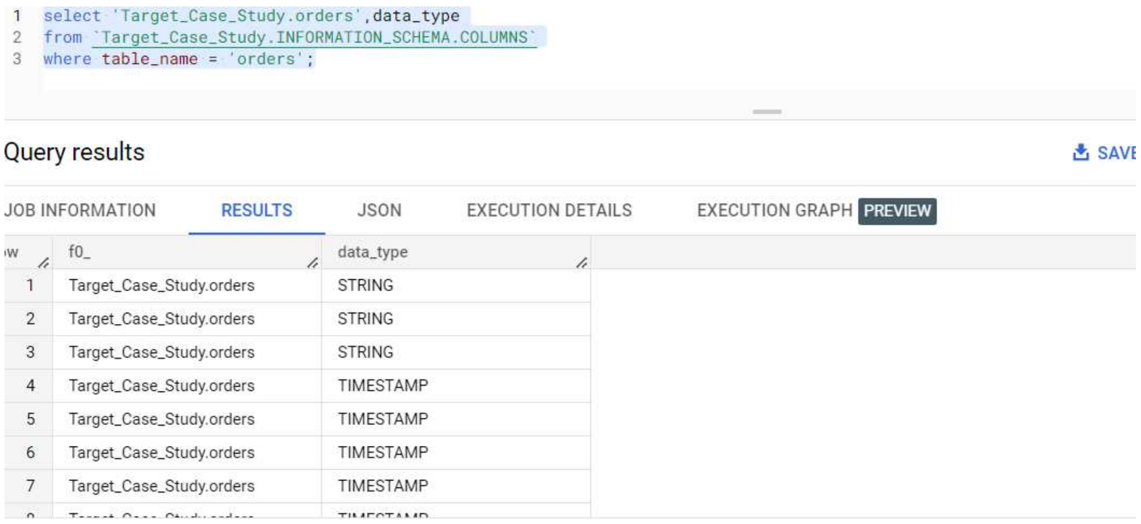
1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset
    1. Data type of columns in a table
    2. Time period for which the data is given
    3. Cities and States of customers ordered during the given period

1.a.)

```
select 'Target_Case_Study.orders',data_type
from `Target_Case_Study.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'orders';
```



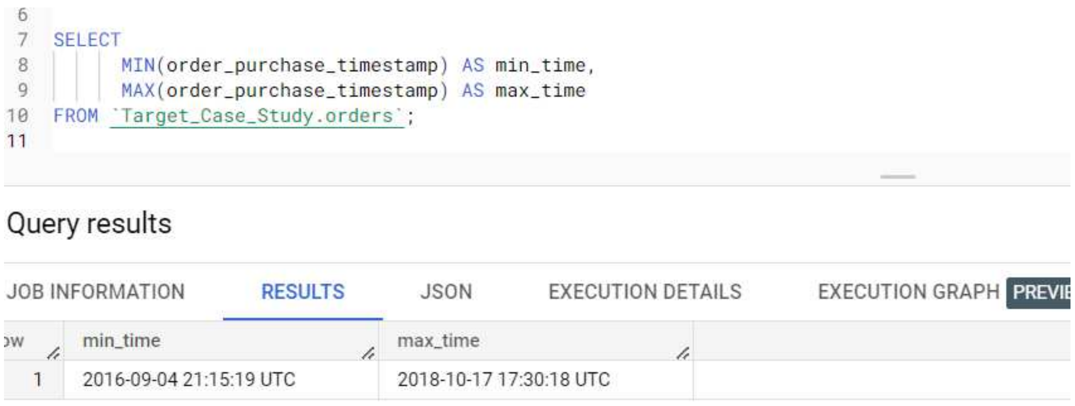
```
select 'Target_Case_Study.orders',data_type
from `Target_Case_Study.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'orders';
```

Query results

w	f0_	data_type
1	Target_Case_Study.orders	STRING
2	Target_Case_Study.orders	STRING
3	Target_Case_Study.orders	STRING
4	Target_Case_Study.orders	TIMESTAMP
5	Target_Case_Study.orders	TIMESTAMP
6	Target_Case_Study.orders	TIMESTAMP
7	Target_Case_Study.orders	TIMESTAMP

1b.)

```
SELECT
      MIN(order_purchase_timestamp) AS min_time,
      MAX(order_purchase_timestamp) AS max_time
FROM `Target_Case_Study.orders`;
```



```
SELECT
      MIN(order_purchase_timestamp) AS min_time,
      MAX(order_purchase_timestamp) AS max_time
FROM `Target_Case_Study.orders`;
```

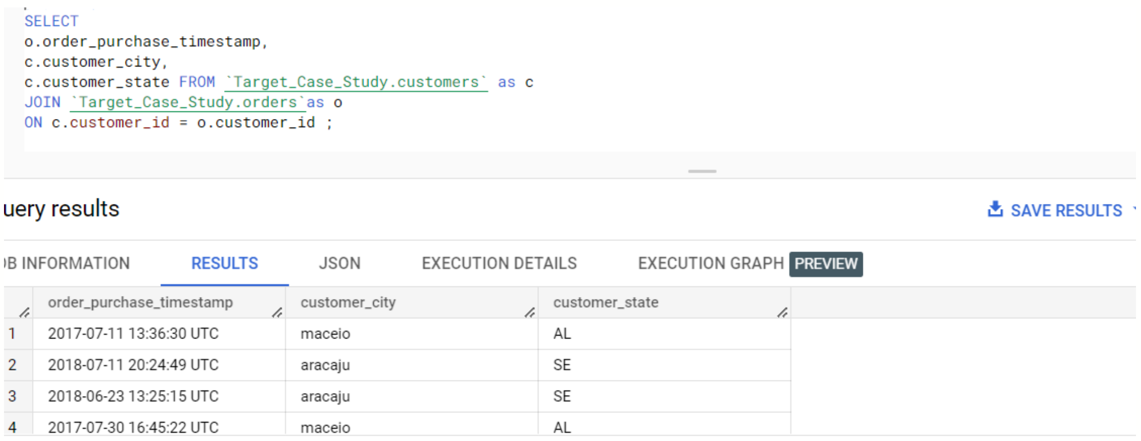
Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIE

ɔw	min_time	max_time
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

1.c.)

```
SELECT
o.order_purchase_timestamp,
c.customer_city,
c.customer_state FROM `Target_Case_Study.customers` as c
JOIN `Target_Case_Study.orders`as o
ON c.customer_id = o.customer_id ;
```



```
SELECT
o.order_purchase_timestamp,
c.customer_city,
c.customer_state FROM `Target_Case_Study.customers` as c
JOIN `Target_Case_Study.orders`as o
ON c.customer_id = o.customer_id ;
```

uery results

SAVE RESULTS

ɔB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUTION GRAPH PREVIEW

	order_purchase_timestamp	customer_city	customer_state
1	2017-07-11 13:36:30 UTC	maceio	AL
2	2018-07-11 20:24:49 UTC	aracaju	SE
3	2018-06-23 13:25:15 UTC	aracaju	SE
4	2017-07-30 16:45:22 UTC	maceio	AL

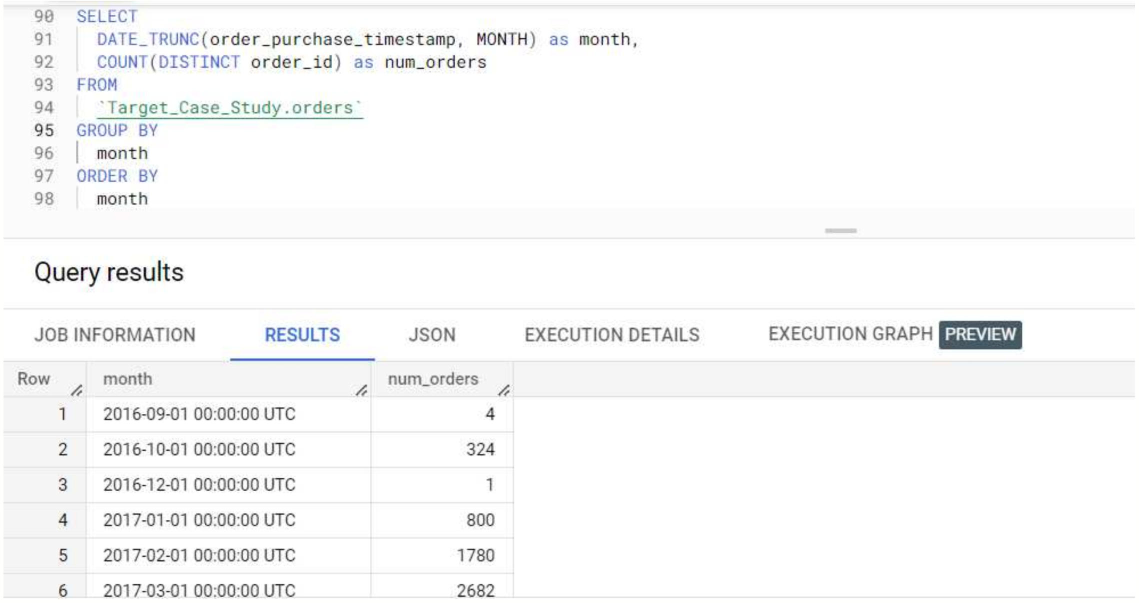
2. In-depth Exploration:

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

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

2.1)

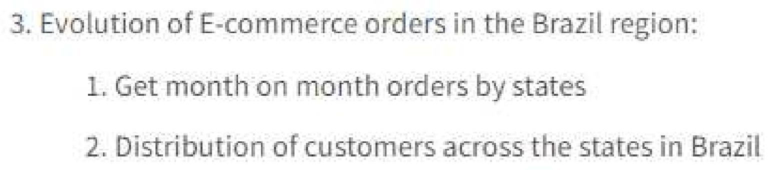
```
SELECT
  DATE_TRUNC(order_purchase_timestamp, MONTH) as month,
  COUNT(DISTINCT order_id) as num_orders
FROM
  `Target_Case_Study.orders`
GROUP BY
  month
ORDER BY
  Month
```



```
SELECT
  DATE_TRUNC(order_purchase_timestamp, MONTH) as month,
  COUNT(DISTINCT order_id) as num_orders
FROM
  `Target_Case_Study.orders`
GROUP BY
  month
ORDER BY
  month
```

Query results

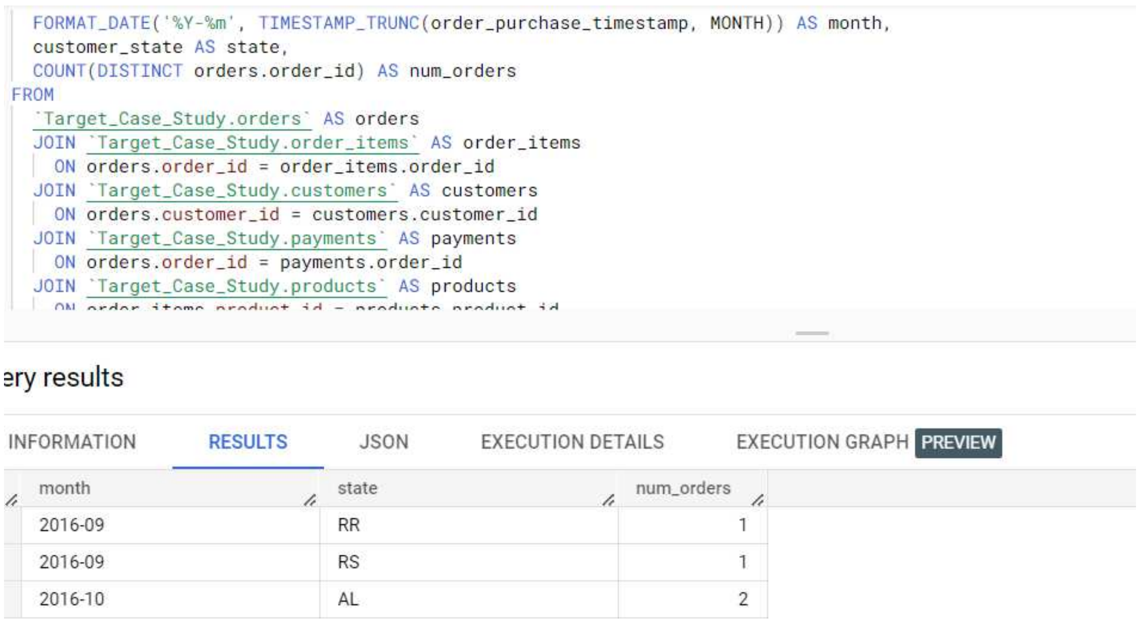
Row	month	num_orders
1	2016-09-01 00:00:00 UTC	4
2	2016-10-01 00:00:00 UTC	324
3	2016-12-01 00:00:00 UTC	1
4	2017-01-01 00:00:00 UTC	800
5	2017-02-01 00:00:00 UTC	1780
6	2017-03-01 00:00:00 UTC	2682



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

3.1)

```
SELECT
  FORMAT_DATE('%Y-%m', TIMESTAMP_TRUNC(order_purchase_timestamp, MONTH)) AS month,
  customer_state AS state,
  COUNT(DISTINCT orders.order_id) AS num_orders
FROM
  `Target_Case_Study.orders` AS orders
  JOIN `Target_case_Study.order_items` AS order_items
    ON orders.order_id = order_items.order_id
  JOIN `Target_Case_Study.customers` AS customers
    ON orders.customer_id = customers.customer_id
  JOIN `Target_Case_Study.payments` AS payments
    ON orders.order_id = payments.order_id
  JOIN `Target_Case_Study.products` AS products
    ON order_items.product_id = products.product_id
  JOIN `Target_Case_Study.sellers` AS sellers
    ON order_items.seller_id = sellers.seller_id
  JOIN `Target_Case_Study.geolocation` AS geolocation
    ON sellers.seller_zip_code_prefix = geolocation.geolocation_zip_code_prefix
GROUP BY
  month,state
ORDER BY
  month,state
```

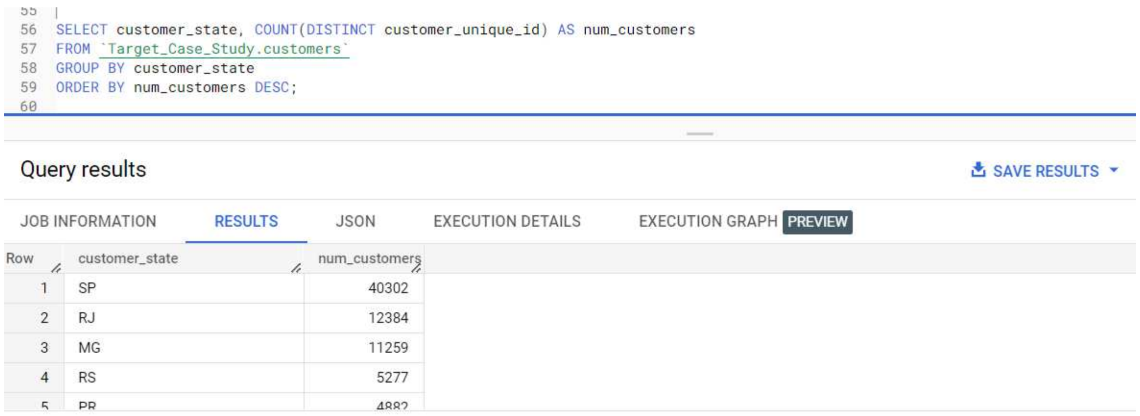


```
  FORMAT_DATE('%Y-%m', TIMESTAMP_TRUNC(order_purchase_timestamp, MONTH)) AS month,
  customer_state AS state,
  COUNT(DISTINCT orders.order_id) AS num_orders
FROM
  `Target_Case_Study.orders` AS orders
  JOIN `Target_Case_Study.order_items` AS order_items
    ON orders.order_id = order_items.order_id
  JOIN `Target_Case_Study.customers` AS customers
    ON orders.customer_id = customers.customer_id
  JOIN `Target_Case_Study.payments` AS payments
    ON orders.order_id = payments.order_id
  JOIN `Target_Case_Study.products` AS products
    ON order_items.product_id = products.product_id
```

month	state	num_orders
2016-09	RR	1
2016-09	RS	1
2016-10	AL	2

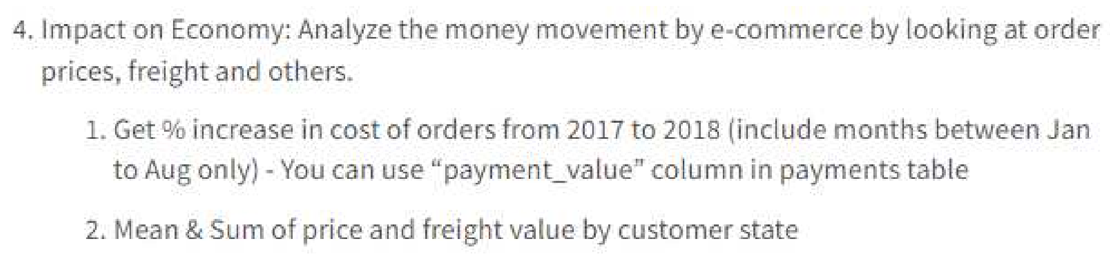
3.2)

SELECT customer_state, COUNT(DISTINCT customer_unique_id) AS num_customers
FROM `Target_Case_Study.customers`
GROUP BY customer_state
ORDER BY num_customers DESC;



```
SELECT customer_state, COUNT(DISTINCT customer_unique_id) AS num_customers
FROM `Target_Case_Study.customers`
GROUP BY customer_state
ORDER BY num_customers DESC;
```

Row	customer_state	num_customers
1	SP	40302
2	RJ	12384
3	MG	11259
4	RS	5277
5	PR	4882



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

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

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

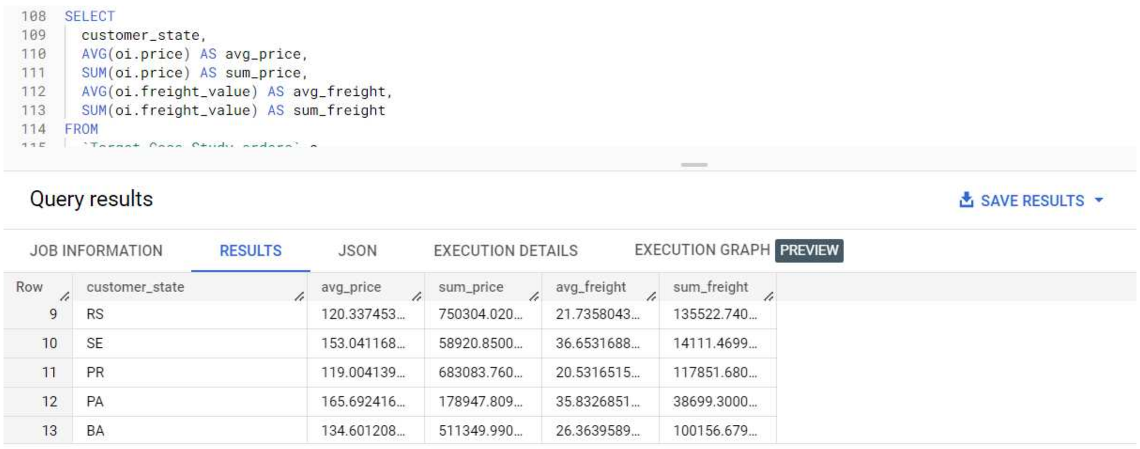
4.1)

```
SELECT
    (SUM(CASE WHEN YEAR(order_purchase_timestamp) = 2018 THEN payment_value ELSE 0 END) / S
UM(CASE WHEN YEAR(order_purchase_timestamp) = 2017 THEN payment_value ELSE 0 END) - 1) * 10
0 AS percentage_increase
FROM
`Target_Case_Study.payments`
WHERE
MONTH(order_purchase_timestamp) BETWEEN 1 AND 8 AND YEAR(order_purchase_timestamp) IN (2017
,2018);
```

4.2)

```
SELECT
  customer_state,
  AVG(oi.price) AS avg_price,
  SUM(oi.price) AS sum_price,
  AVG(oi.freight_value) AS avg_freight,
  SUM(oi.freight_value) AS sum_freight
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
```



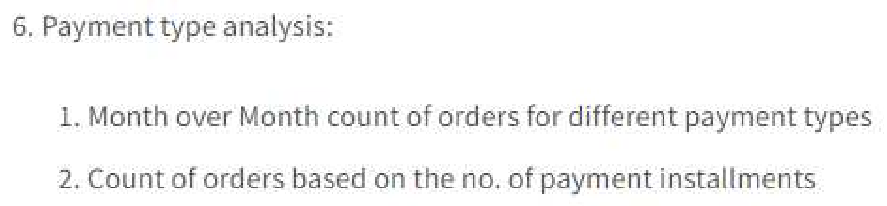
```
SELECT
  customer_state,
  AVG(oi.price) AS avg_price,
  SUM(oi.price) AS sum_price,
  AVG(oi.freight_value) AS avg_freight,
  SUM(oi.freight_value) AS sum_freight
FROM
```

Query results

SAVE RESULTS

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

Row	customer_state	avg_price	sum_price	avg_freight	sum_freight
9	RS	120.337453...	750304.020...	21.7358043...	135522.740...
10	SE	153.041168...	58920.8500...	36.6531688...	14111.4699...
11	PR	119.004139...	683083.760...	20.5316515...	117851.680...
12	PA	165.692416...	178947.809...	35.8326851...	38699.3000...
13	BA	134.601208...	511349.990...	26.3639589...	100156.679...

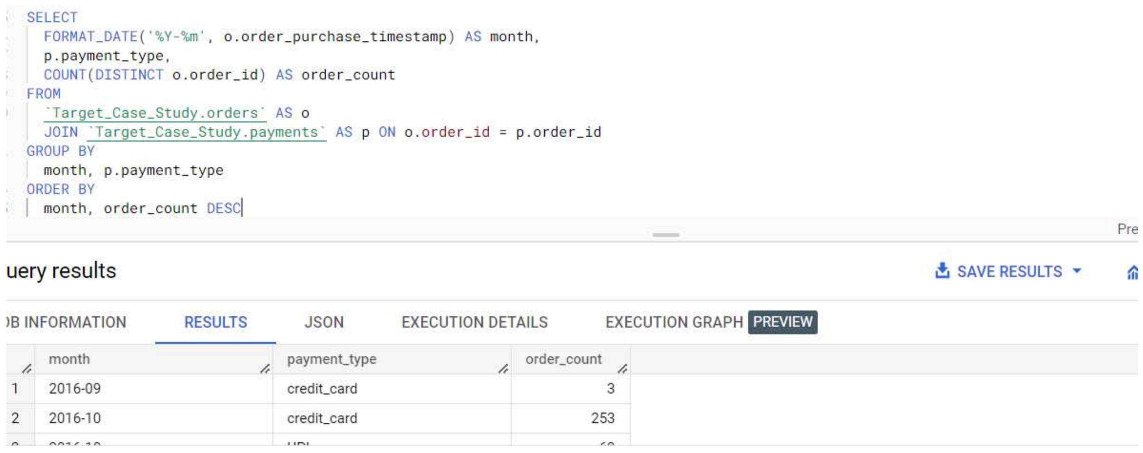


6. Payment type analysis:

1. Month over Month count of orders for different payment types
2. Count of orders based on the no. of payment installments

6.1)

```
SELECT
  FORMAT_DATE('%Y-%m', o.order_purchase_timestamp) AS month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS order_count
FROM
  `Target_Case_Study.orders` AS o
  JOIN `Target_Case_Study.payments` AS p ON o.order_id = p.order_id
GROUP BY
  month, p.payment_type
ORDER BY
  month, order_count DESC
```

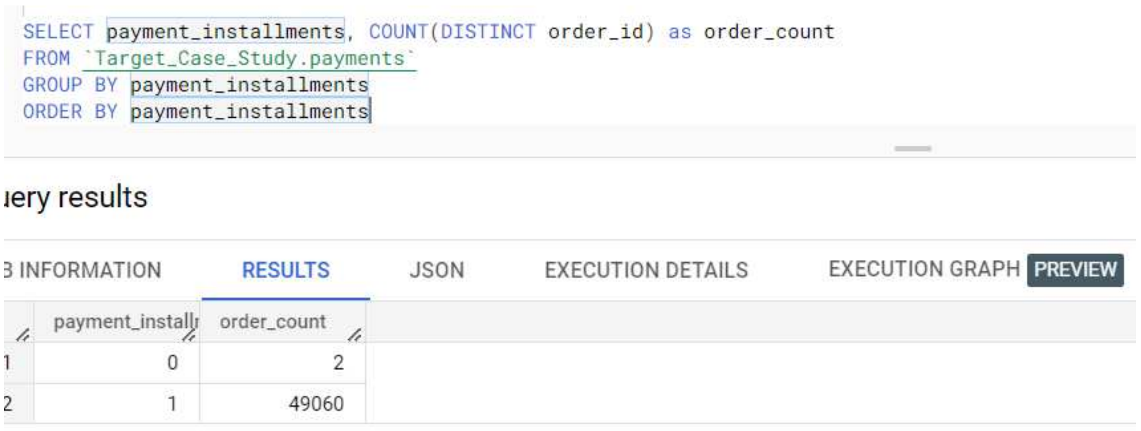


```
SELECT
  FORMAT_DATE('%Y-%m', o.order_purchase_timestamp) AS month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS order_count
FROM
  `Target_Case_Study.orders` AS o
  JOIN `Target_Case_Study.payments` AS p ON o.order_id = p.order_id
GROUP BY
  month, p.payment_type
ORDER BY
  month, order_count DESC
```

	month	payment_type	order_count
1	2016-09	credit_card	3
2	2016-10	credit_card	253

6.2)

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



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

ıery results

	payment_installı	order_count
1	0	2
2	1	49060