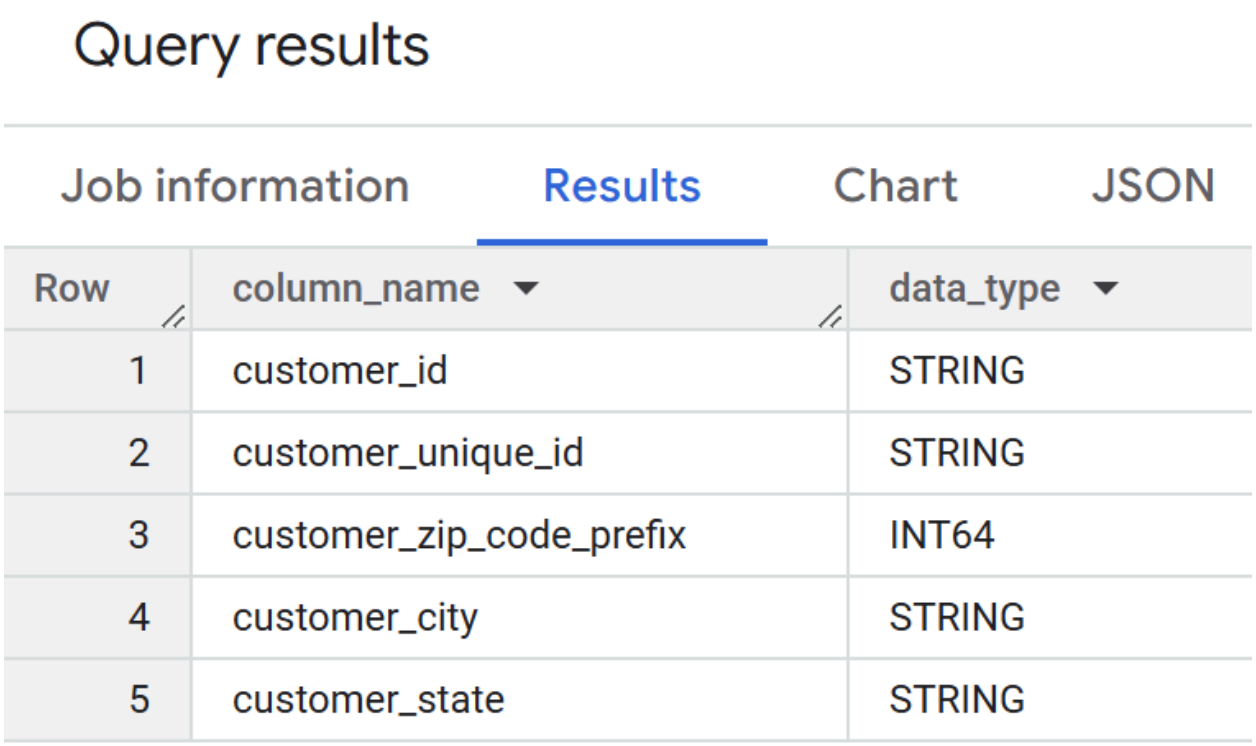
I. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset.

A. Data type of all columns in the “customers” table.
Query:

```
select column_name, data_type
from target.INFORMATION_SCHEMA.COLUMNS
where table_name='customers'
```

Output:



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

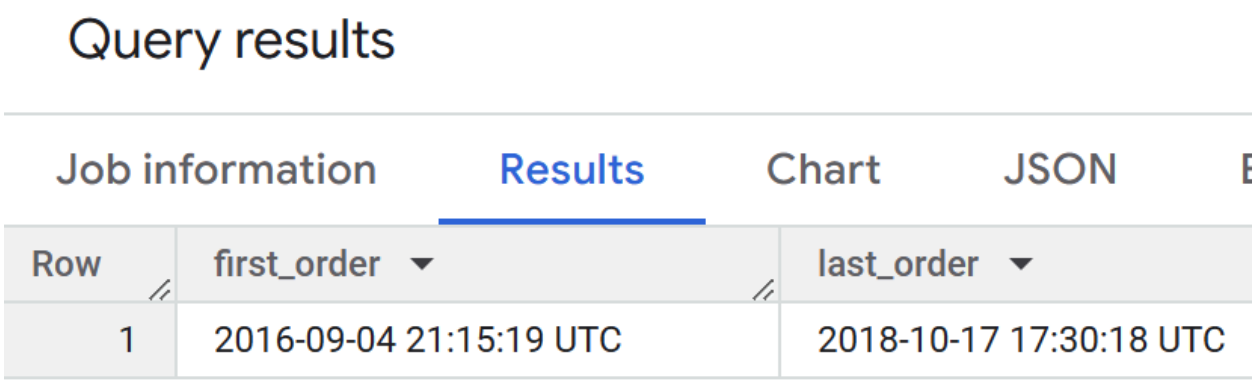
Insights:
Table contains 5 columns out of which primary id is of string type and except zip code which is int , all are string type

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

Query:

```
select min(order_purchase_timestamp) as first_order,
max(order_purchase_timestamp) as last_order
from `target.orders`
```

Output:



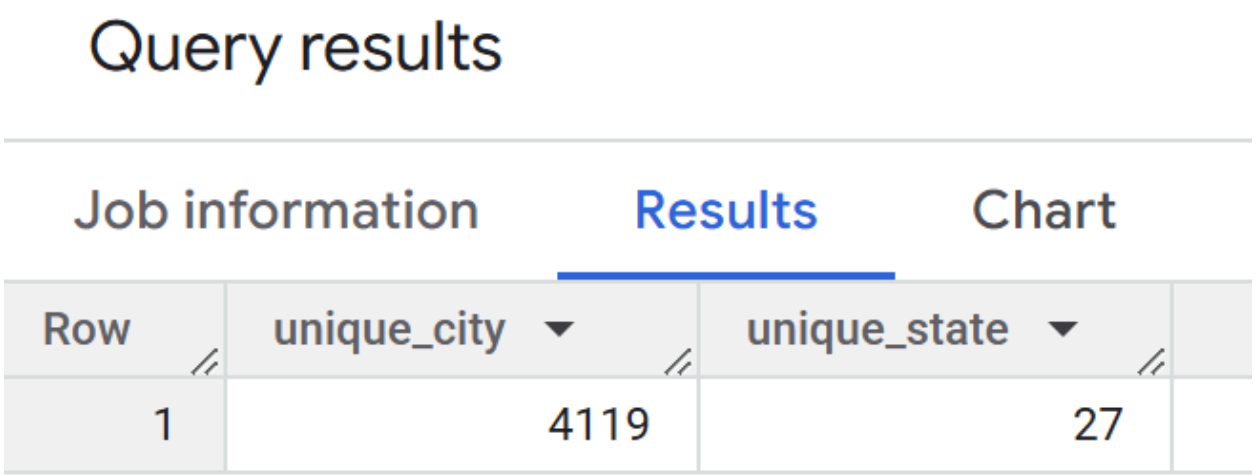
Insights:
First order is placed on 04th September 2016 at 9:15pm and
Lastorder is placed on 17th October 2018 at 5:30pm

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

Query:

```
select count(distinct customer_city) as unique_city,count(distinct customer_state) as
unique_state
from `target.customers`
```

Output:



Insight:
There are 4119 unique cities and 27 unique states

II. In-depth Exploration:

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

Query:

```
WITH MonthlyOrderCounts AS (
    SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
        COUNT(*) AS order_count
    FROM
        `target.orders`
    GROUP BY
        order_year, order_month
    ORDER BY
        order_year, order_month
)
SELECT
    order_year,
    order_month,
    order_count,
    LAG(order_count, 12, 0) OVER (ORDER BY order_year, order_month) AS
previous_year_count,
    CASE
        WHEN order_count > LAG(order_count, 12, 0) OVER (ORDER BY order_year,
order_month) THEN 'Increased'
        ELSE 'Not Increased'
    END AS yearly_increase
FROM
    MonthlyOrderCounts;
```

Output:

Query results

Job information | Results | Chart | JSON | Execution details | Execution graph

Row	order_year	order_month	order_count	previous_year_count	yearly_increase
1	2017	7	4026	0	Increased
2	2017	4	2404	0	Increased
3	2017	2	1780	0	Increased
4	2017	12	5673	1	Increased
5	2018	1	7269	800	Increased

Insights: Except in the months of september and october 2018, number of orders are gradually increasing

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

Query:

```
SELECT
        EXTRACT(YEAR FROM order_purchase_timestamp) AS order_year,
        EXTRACT(MONTH FROM order_purchase_timestamp) AS order_month,
        COUNT(*) AS order_count
    FROM
        `target.orders`
    GROUP BY
        order_year, order_month
    ORDER BY
        order_count desc
```

Output:

Query results

Job information | Results | Chart | JSON | Exe

Row	order_year	order_month	order_count
1	2017	11	7544
2	2018	1	7269
3	2018	3	7211
4	2018	4	6939
5	2018	5	6873

Insight:
Sales are peaks from period of November 2017 to August 2018

C. During what time of the day, do the Brazilian customers mostly place their Orders?

Query:

```
      SELECT
        case
          when EXTRACT(HOUR FROM order_purchase_timestamp) between 0 and 6 then 'Dawn'
          when EXTRACT(HOUR FROM order_purchase_timestamp) between 7 and 12 then
'Mornings'
          when EXTRACT(HOUR FROM order_purchase_timestamp) between 13 and 18 then
'Afternoon'
          when EXTRACT(HOUR FROM order_purchase_timestamp) between 19 and 23 then
'Night'
        end as Time_of_the_day,
        COUNT(order_id) AS order_count
    FROM
        `target.orders`
    GROUP BY
        Time_of_the_day
    ORDER BY
        order_count desc
```

Output:

Query results

Job information | Results | Chart | JSON

Row	Time_of_the_day	order_count
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

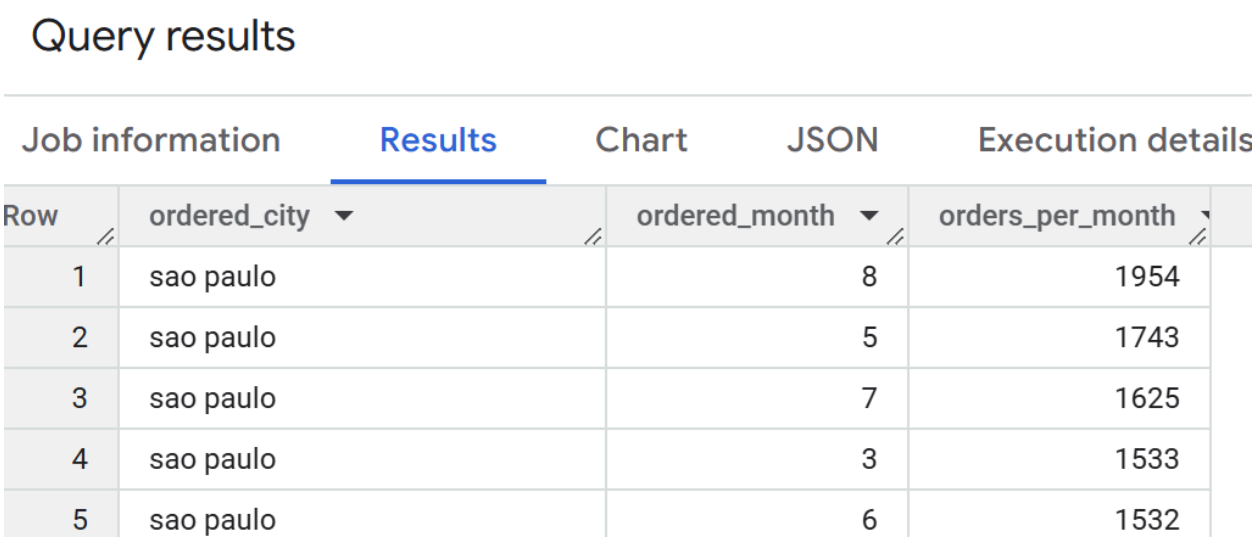
Insight:

Brazilian customers ordered more at afternoon i.e 38135 and less at dawn i.e 5242

III. Evolution of E-commerce orders in the Brazil region:
A. Get the month on month no. of orders placed in each state.
Query:

```
select c.customer_city as ordered_city,
  extract(month from o.order_purchase_timestamp) as ordered_month,
  count(o.order_id) as orders_per_month
from `target.customers` c join `target.orders` o on c.customer_id = o.customer_id
group by c.customer_city,ordered_month
order by orders_per_month desc
```

Output:



Row	ordered_city	ordered_month	orders_per_month
1	sao paulo	8	1954
2	sao paulo	5	1743
3	sao paulo	7	1625
4	sao paulo	3	1533
5	sao paulo	6	1532

Insight:
Sao paulo has highest orders in the month of august i.e 1954 and curitiba has lowest orders in month of January i.e 150

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

Query:

```
select customer_state, count(customer_unique_id) as Total_Customers
from `target.customers`
group by customer_state
order by Total_Customers desc
```

Output:

Job information | Results | Chart | JSON

Row	customer_state	Total_Customers
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045

Insight:
The State SP(Sao Paulo) has highest number of customers i.e 41746 and the state RR has lowest number of customers i.e 46

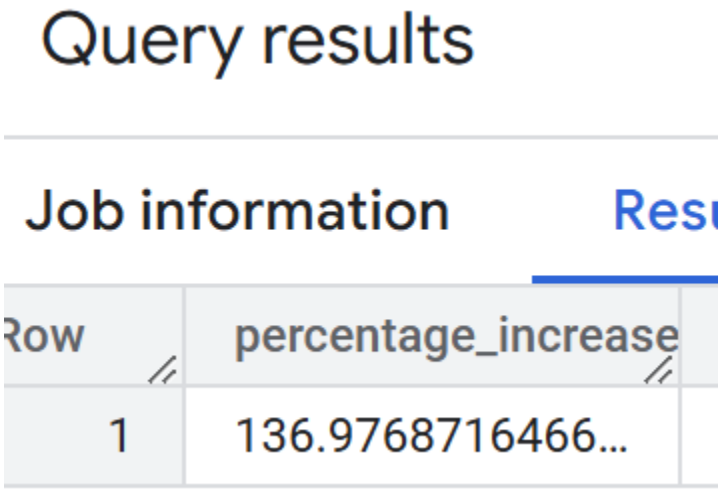
IV. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

A. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
Query:

```
WITH Cost2017 AS (
    SELECT
        SUM(p.payment_value) AS total_cost_2017
    FROM
        `target.orders` o join `target.payments` p on o.order_id=p.order_id
    WHERE
        EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017
        AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
),
```

```
Cost2018 AS (
    SELECT
        SUM(p.payment_value) AS total_cost_2018
    FROM
        `target.orders` o join `target.payments` p on o.order_id=p.order_id
    WHERE
        EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018
        AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
)
SELECT
    (
        (Cost2018.total_cost_2018 - Cost2017.total_cost_2017) /
Cost2017.total_cost_2017
    ) * 100 AS percentage_increase
FROM
    Cost2017, Cost2018;
```

Output:



Insight:
from 2017 to 2018 there is an increase of 137%

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

Query:

```
select c.customer_state,
  round(sum(p.payment_value),2) as total_price,
  round(avg(p.payment_value),2) as average_price
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 average_price desc
```

Output:

Query results

Job information | Results | Chart | JSON | Execution details

Row	customer_state	total_price	average_price
1	PB	141545.72	248.33
2	AC	19680.62	234.29
3	RO	60866.2	233.2
4	AP	16262.8	232.33
5	AL	96962.06	227.08

Insight:
There PB state has the highest average price i.e 248.33 and SP state has lowest average i.e 137.5

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

Query:

```
select c.customer_state,
  round(sum(i.freight_value),2) as total_freight_price,
  round(avg(i.freight_value),2) as average_freight_price
from `target.customers` c
join `target.orders` o on c.customer_id=o.customer_id
join `target.order_items` i on o.order_id=i.order_id
group by c.customer_state
order by average_freight_price desc
```

Output:

Query results

Job information | Results | Chart | JSON | Execution details

Row	customer_state	total_freight_price	average_freight_price
1	RR	2235.19	42.98
2	PB	25719.73	42.72
3	RO	11417.38	41.07
4	AC	3686.75	40.07
5	PI	21218.2	39.15

Insight:
Therefore RR state has highest average freight price i.e 42.98 and SP state has lowest freight price i.e 15.15

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

A. 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.

Query:

```
Select order_id,
  date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_deliver,
  date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery
from `target.orders`
where order_delivered_customer_date is not null and order_purchase_timestamp is not
null and order_estimated_delivery_date is not null
order by time_to_deliver desc
```

Output:

Query results

Job information | Results | Chart | JSON | Execution details

Row	order_id	time_to_deliver	diff_estimated_delive
1	ca07593549f1816d26a572e06...	209	-181
2	1b3190b2dfa9d789e1f14c05b...	208	-188
3	440d0d17af552815d15a9e41a...	195	-165
4	285ab9426d6982034523a855f...	194	-166
5	0f4519c5f1c541ddec9f21b3bd...	194	-161

Insight: the order with order_id = ca07593549f1816d26a572e06dc1eab6 has taken more time to deliver.

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

```
select customer_state,average_freight_price
from (
select c.customer_state,
  round(avg(i.freight_value),2) as average_freight_price,
  dense_rank() over(order by avg(i.freight_value)) as min_rnk,
  dense_rank() over(order by avg(i.freight_value) desc) as max_rnk
from `target.customers` c
join `target.orders` o on c.customer_id=o.customer_id
join `target.order_items` i on o.order_id=i.order_id
group by c.customer_state
) as s
where max_rnk<=5 or min_rnk<=5
order by average_freight_price desc
```

Output:

Query results

Job information | Results | Chart | JSON

Row	customer_state	average_freight_pric
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15

Insight:
State RR has the highest average freight price i.e 42.98 and State SP has lowest average freight price i,e 15.15

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

Query:

```
with delivery_time as(
  select c.customer_state,

round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)),2
) as average_delivery_time
  from `target.customers` c join `target.orders` o on c.customer_id = o.customer_id
  where o.order_delivered_customer_date is not null and o.order_purchase_timestamp is
not null
  group by c.customer_state
),
top_5 as (
  select customer_state,average_delivery_time
  from delivery_time
  order by average_delivery_time desc
```

```
  limit 5
),
bottom_5 as(
  select customer_state,average_delivery_time
  from delivery_time
  order by average_delivery_time
  limit 5
)
select customer_state,average_delivery_time
from top_5
UNION ALL
select customer_state,average_delivery_time
from bottom_5
order by average_delivery_time desc
```

Output:

Query results

Job information | Results | Chart | JSON

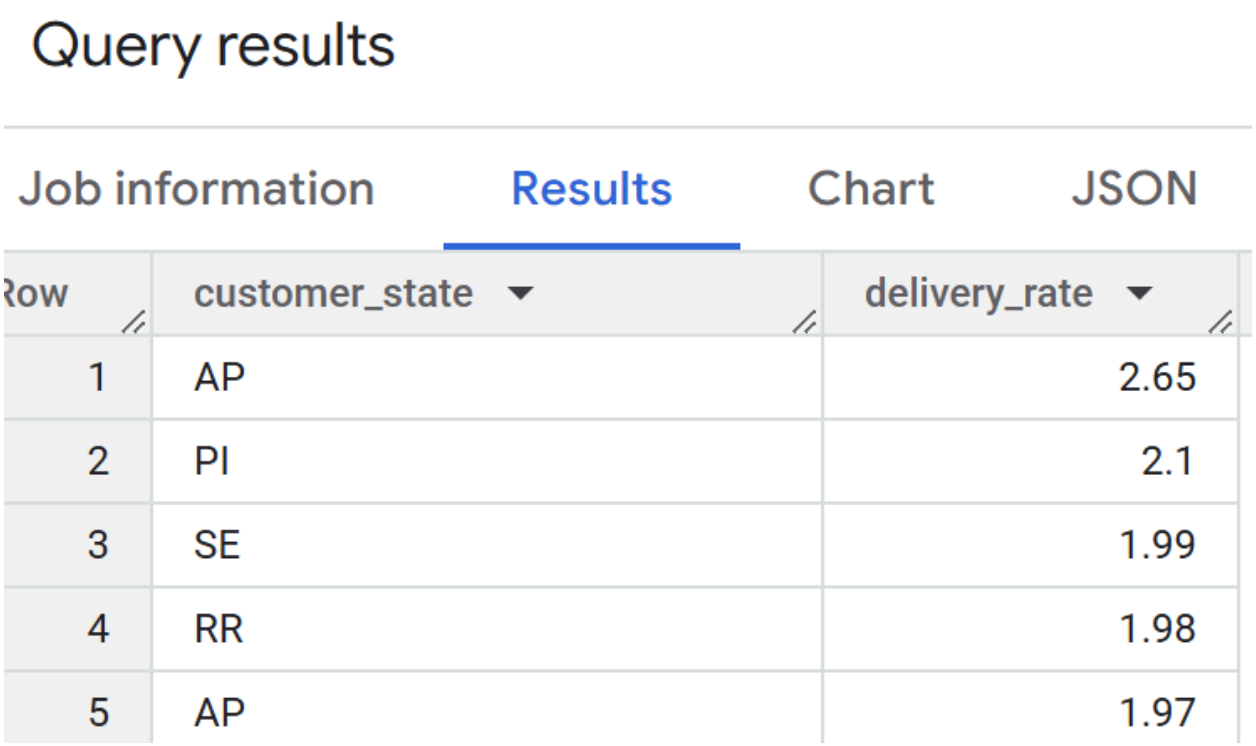
Row	customer_state	average_delivery_tim
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Insight:
RR state has highest average delivery i.e 28.98 means it takes more number of days ,therefore RR state is the slowest that takes 29 days to deliver the order and delivery time and the state SP has lowest average delivery i.e 8.3 means it less number of days to deliver , therefore SP state is the quickest that takes 8.3 days to deliver

D. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
Query:

```
with actual_delivery_time as(
  select c.customer_state,
    round(avg(extract(day from o.order_delivered_customer_date)),2) as
average_actual_time
  from `target.customers` c join `target.orders` o on c.customer_id = o.customer_id
  where o.order_delivered_customer_date is not null
  group by c.customer_state
),
estimated_delivery_time as(
  select c.customer_state,
    round(avg(extract(day from o.order_estimated_delivery_date)),2) as
average_estimated_time
  from `target.customers` c join `target.orders` o on c.customer_id = o.customer_id
  where o.order_estimated_delivery_date is not null
  group by c.customer_state
)
select a.customer_state,round((a.average_actual_time-e.average_estimated_time),3) as
delivery_rate
from estimated_delivery_time e,actual_delivery_time a
order by delivery_rate desc
limit 5
```

Output:



Row	customer_state	delivery_rate
1	AP	2.65
2	PI	2.1
3	SE	1.99
4	RR	1.98
5	AP	1.97

Insight:

AP state has highest delivery rate i.e 2.65 means it has delivered more faster than the expected delivery time

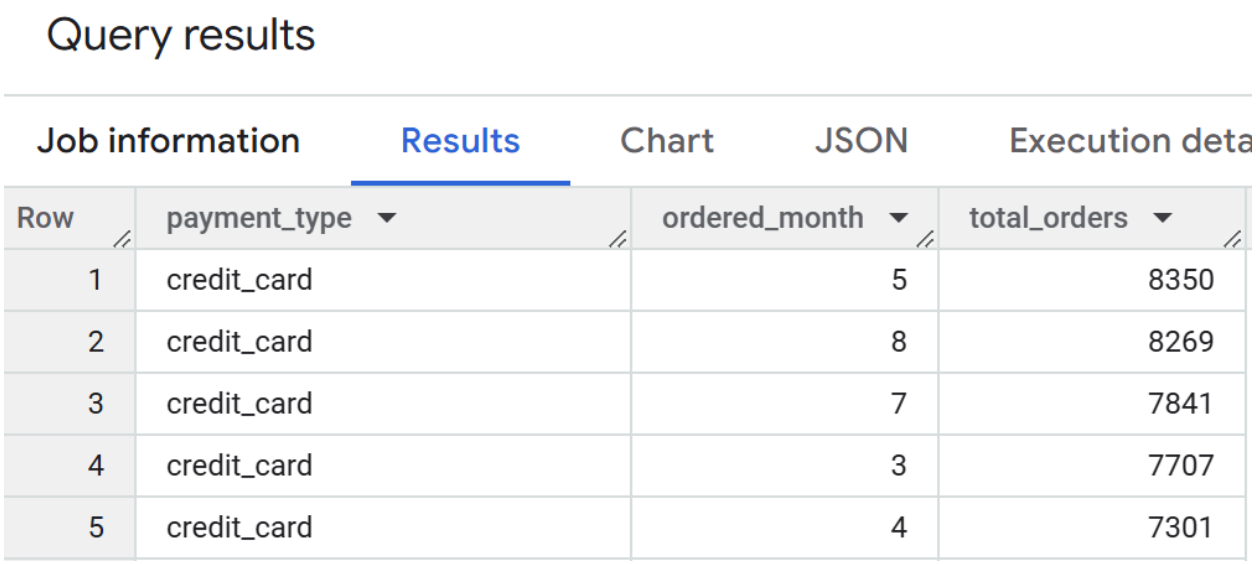
VI. Analysis based on the payments:

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

Query:

```
select p.payment_type,
  extract(month from o.order_purchase_timestamp) as ordered_month,
  count(o.order_id) as total_orders
from `target.orders` o join `target.payments` p on o.order_id=p.order_id
where p.payment_type != 'not_defined'
group by extract(month from o.order_purchase_timestamp),p.payment_type
order by total_orders desc
```

Output:



Row	payment_type	ordered_month	total_orders
1	credit_card	5	8350
2	credit_card	8	8269
3	credit_card	7	7841
4	credit_card	3	7707
5	credit_card	4	7301

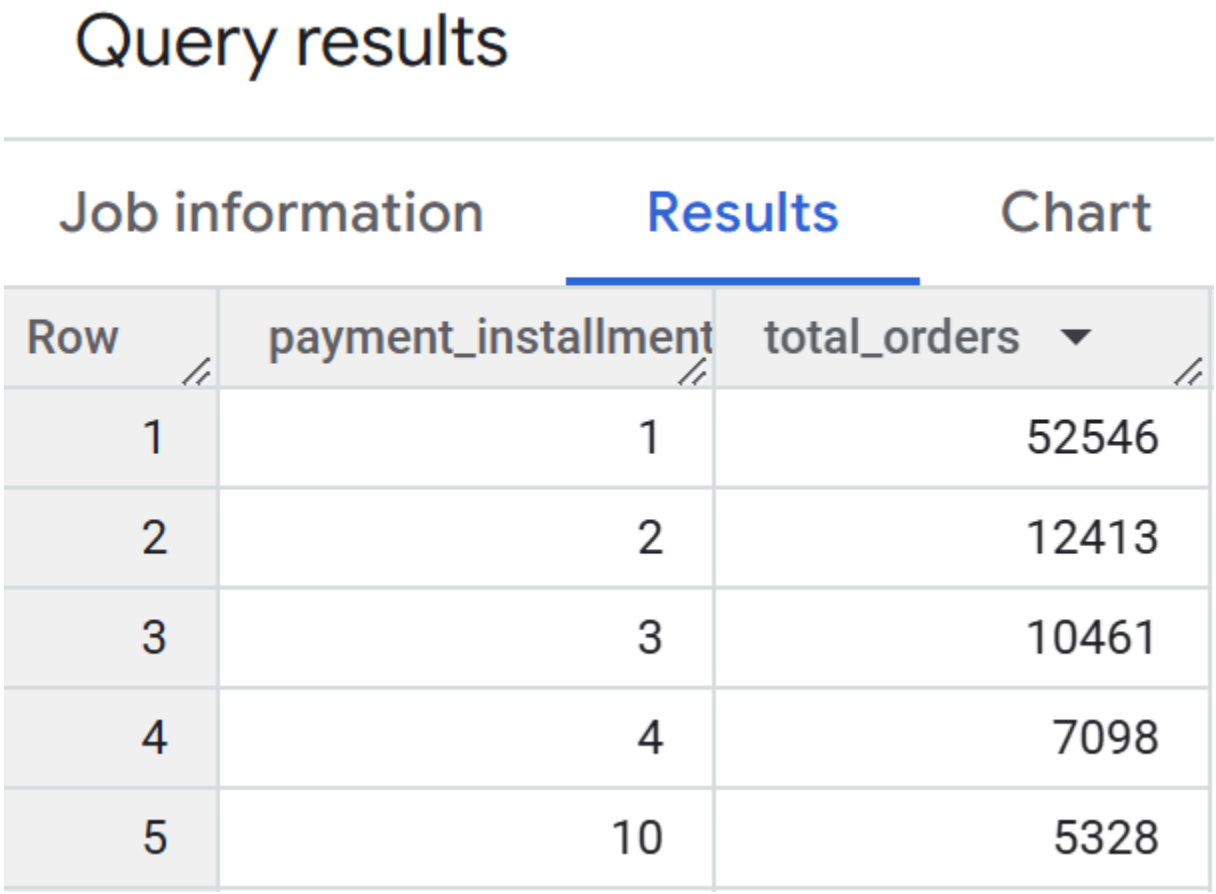
Insight:
In the month of may, using credit card highest number of orders are placed i.e 8350 and in the month of september, using debit card least number of orders are placed i.e 43

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

Query:

```
select p.payment_installments,
  count(o.order_id) as total_orders
from `target.orders` o join `target.payments` p on o.order_id=p.order_id
where p.payment_installments >=1
group by p.payment_installments
order by total_orders desc
```

Output:



Row	payment_installment	total_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328

Insight:
Hence, there are highest number of orders where Number of emi is 1and lowest number of orders where Number of emi are 23