Q1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset

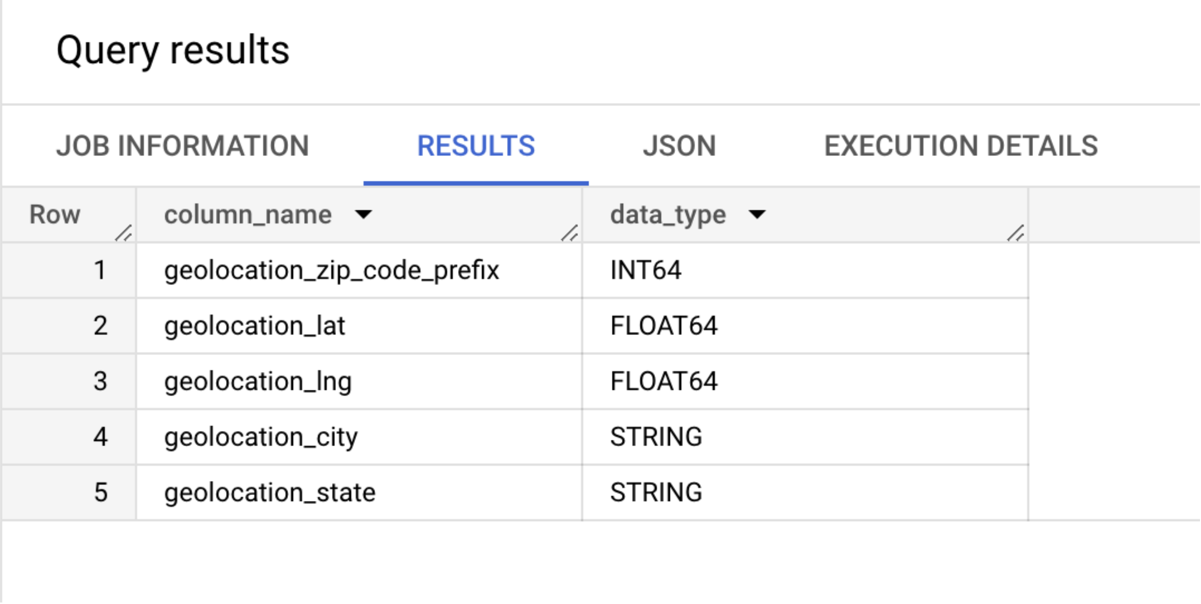
1. *Data type of columns in a table*

Ans:

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

(Inside table_name, we can write which table name data_type we want)

Output:-



Row	column_name	data_type
1	geolocation_zip_code_prefix	INT64
2	geolocation_lat	FLOAT64
3	geolocation_lng	FLOAT64
4	geolocation_city	STRING
5	geolocation_state	STRING

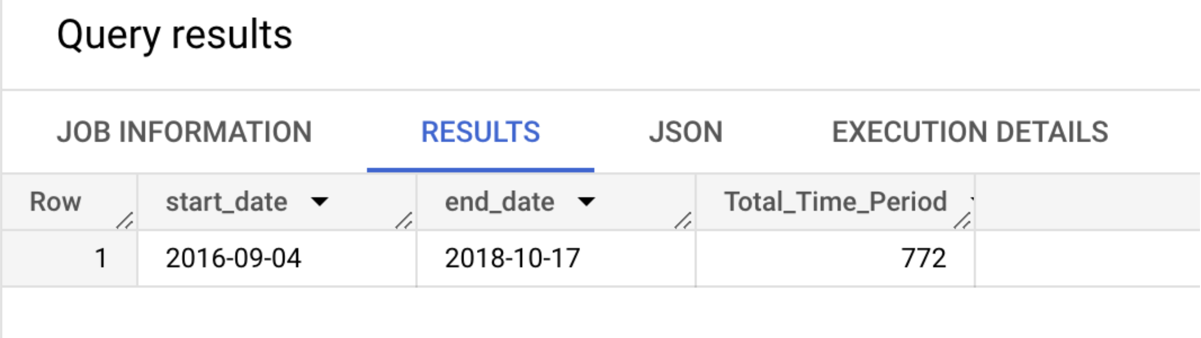
2. Time period for which the data is given

Ans:

```
SELECT
        Date(MIN(order_purchase_timestamp)) AS start_date,
        Date(MAX(order_purchase_timestamp)) AS end_date,
        Date_diff(MAX(order_purchase_timestamp),
        MIN(order_purchase_timestamp),DAY) AS Total_Time_Period
FROM
        `target-sql-case-study-387518.target.orders`
```

(Total time period using date_diff and start and end date using min and a max of the Purchase Order date i,e, 772 days)

Output:-



Row	start_date	end_date	Total_Time_Period
1	2016-09-04	2018-10-17	772

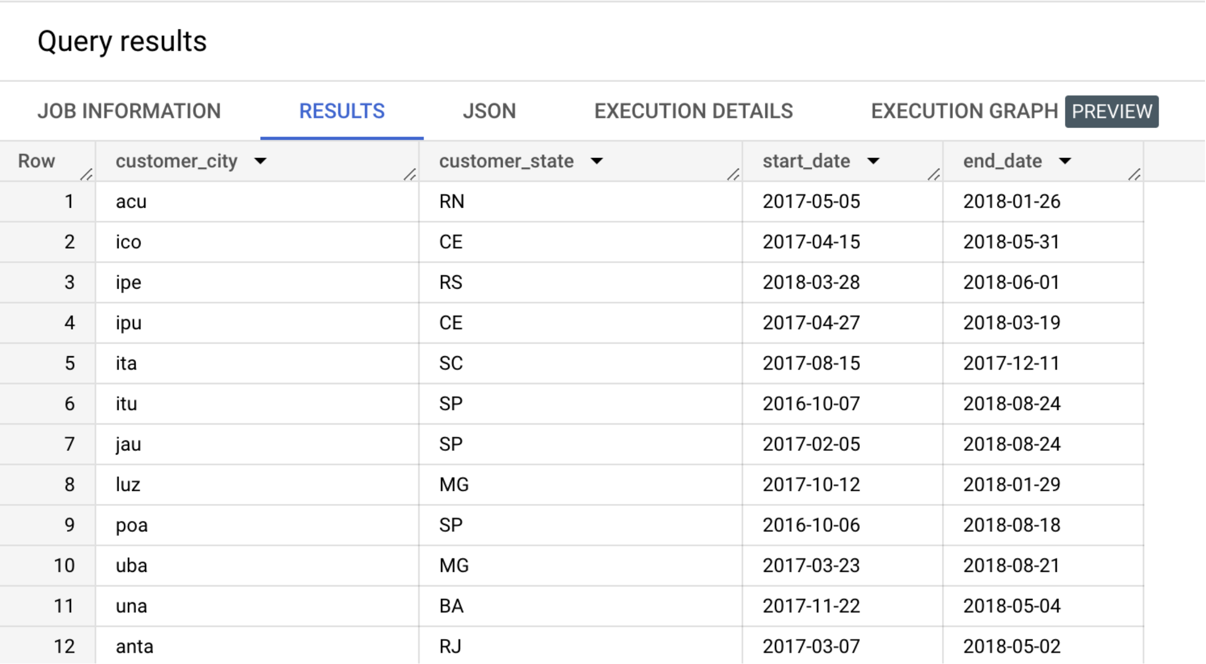
3. Cities and States of customers ordered during the given period

Ans:

```
SELECT
       c.customer_city, c.customer_state,
       Date(MIN(order_purchase_timestamp)) AS start_date,
       Date(MAX(order_purchase_timestamp)) AS end_date
FROM `target-sql-case-study-387518.target.customers` AS c
JOIN
       `target-sql-case-study-387518.target.orders` AS o
ON c.customer_id = o.customer_id
WHERE
       o.order_purchase_timestamp >= (SELECT MIN(order_purchase_timestamp)
       FROM `target-sql-case-study-387518.target.orders`)
       AND o.order_purchase_timestamp <= (SELECT
       MAX(order_purchase_timestamp) FROM
       `target-sql-case-study-387518.target.orders`)
GROUP BY c.customer_city, c.customer_state
```

(This SQL query will be the same if we don't use 'WHERE' and 'GROUP BY', I used it to group it by dates for specific locations)

Output:-



Row	customer_city	customer_state	start_date	end_date
1	acu	RN	2017-05-05	2018-01-26
2	ico	CE	2017-04-15	2018-05-31
3	ipe	RS	2018-03-28	2018-06-01
4	ipu	CE	2017-04-27	2018-03-19
5	ita	SC	2017-08-15	2017-12-11
6	itu	SP	2016-10-07	2018-08-24
7	jau	SP	2017-02-05	2018-08-24
8	luz	MG	2017-10-12	2018-01-29
9	poa	SP	2016-10-06	2018-08-18
10	uba	MG	2017-03-23	2018-08-21
11	una	BA	2017-11-22	2018-05-04
12	anta	RJ	2017-03-07	2018-05-02

Q2.In-depth Exploration:

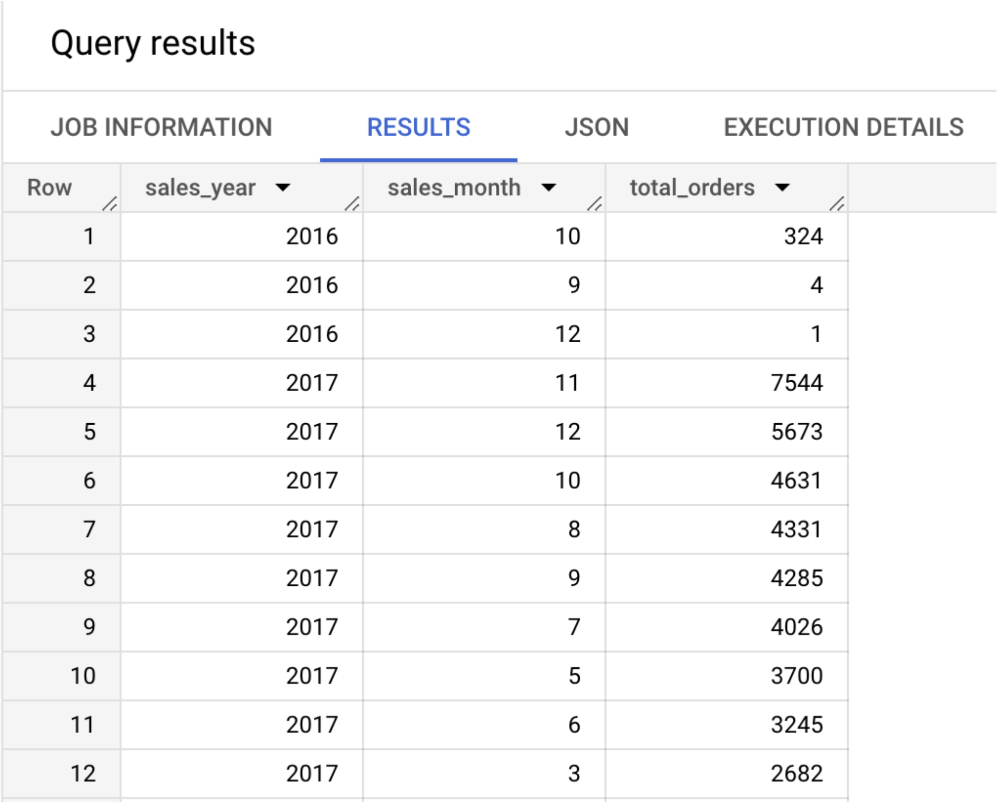
1. ***Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?***

Ans:

A) We'll Find Growing Trend of E-commerce in Brazil.

```
SELECT

        EXTRACT(YEAR FROM o.order_purchase_timestamp) AS sales_year,

        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS sales_month,

        COUNT(*) AS total_orders

FROM `target-sql-case-study-387518.target.orders` as o

GROUP BY sales_year, sales_month

ORDER BY sales_year, total_orders desc,sales_month
```

Output:-



Row	sales_year	sales_month	total_orders
1	2016	10	324
2	2016	9	4
3	2016	12	1
4	2017	11	7544
5	2017	12	5673
6	2017	10	4631
7	2017	8	4331
8	2017	9	4285
9	2017	7	4026
10	2017	5	3700
11	2017	6	3245
12	2017	3	2682

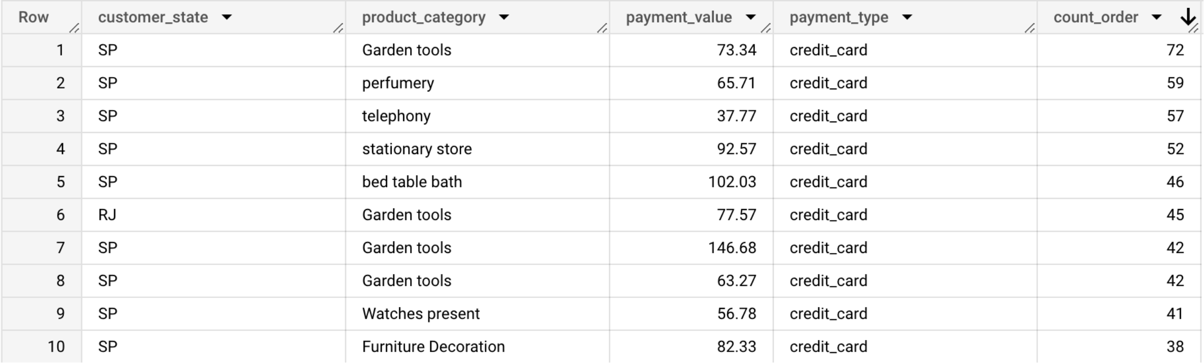
As we can at the output, we found the highest order per year per month, i.e.

- 2016- October (10th month) - *324 orders*
- 2017- November (11th month) - *7544 orders*
- 2018- January (1st Month) - *7269 orders*

B) Complete Scenario of E-commerce in Brazil

```
SELECT c.customer_state,
           p.product_category,
           pm.payment_value,
           pm.payment_type,
           count(o.order_id) as count_order
FROM `target-sql-case-study-387518.target.customers` c
JOIN `target-sql-case-study-387518.target.orders` o
ON c.customer_id = o.customer_id
JOIN `target-sql-case-study-387518.target.order_items` oi
ON o.order_id = oi.order_id
JOIN `target-sql-case-study-387518.target.products` p
ON oi.product_id = p.product_id
LEFT JOIN `target-sql-case-study-387518.target.payments` pm
ON o.order_id = pm.order_id
group by customer_state,product_category,payment_value,payment_type
```

Output:-



Row	customer_state	product_category	payment_value	payment_type	count_order
1	SP	Garden tools	73.34	credit_card	72
2	SP	perfumery	65.71	credit_card	59
3	SP	telephony	37.77	credit_card	57
4	SP	stationary store	92.57	credit_card	52
5	SP	bed table bath	102.03	credit_card	46
6	RJ	Garden tools	77.57	credit_card	45
7	SP	Garden tools	146.68	credit_card	42
8	SP	Garden tools	63.27	credit_card	42
9	SP	Watches present	56.78	credit_card	41
10	SP	Furniture Decoration	82.33	credit_card	38

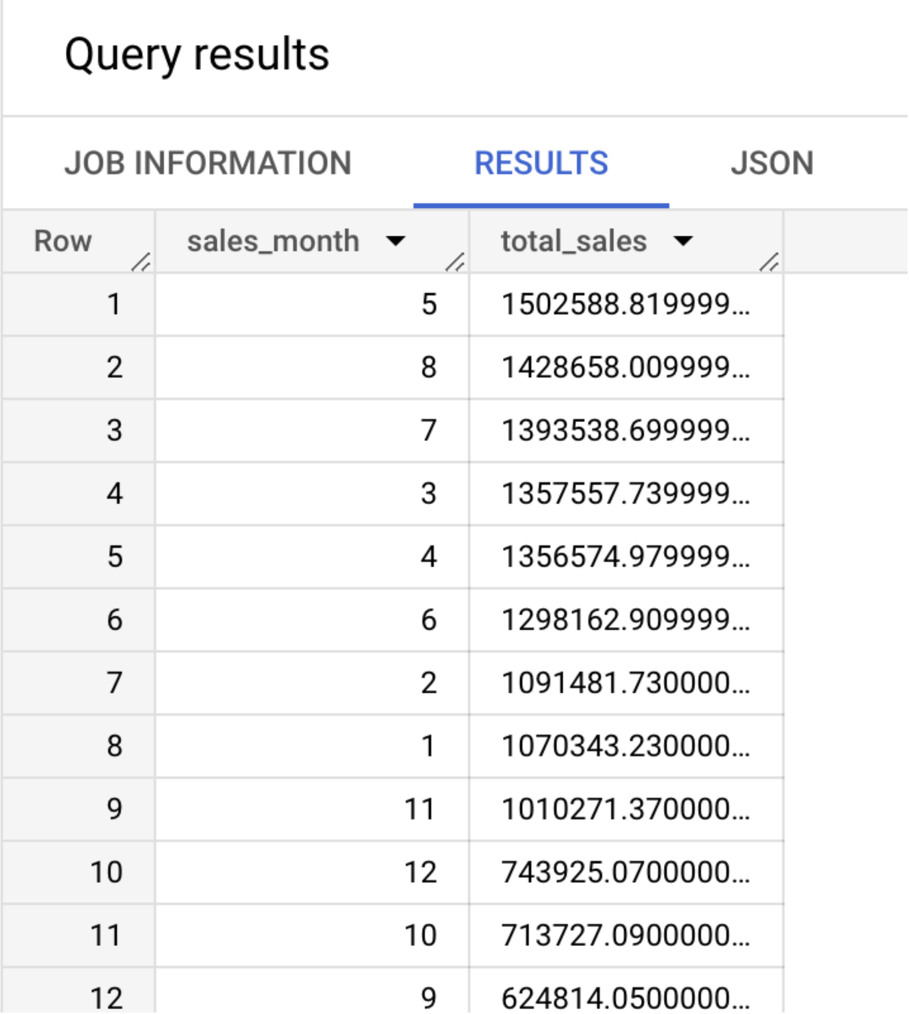
From the output, we got to know about scenarios (we can check each value using ‘order_by’):

- *‘fixed telephony’* has the highest payment_value
- *‘Perfumery, HEALTH BEAUTY, Garden tool’*, Had the lowest payment_value
- SP state had the highest orders from the category of ‘Garden tools’
- 'Maximum payment_type is *Credit Card*

C) Seasonality with Peaks at Specific Months:

```
SELECT
      EXTRACT(MONTH FROM o.order_purchase_timestamp) AS sales_month,
      SUM(oi.price) AS total_sales
FROM `target-sql-case-study-387518.target.orders` o
JOIN `target-sql-case-study-387518.target.order_items` oi
ON o.order_id = oi.order_id
GROUP BY sales_month
ORDER BY total_sales DESC
```

Output:-



Row	sales_month	total_sales
1	5	1502588.819999...
2	8	1428658.009999...
3	7	1393538.699999...
4	3	1357557.739999...
5	4	1356574.979999...
6	6	1298162.909999...
7	2	1091481.730000...
8	1	1070343.230000...
9	11	1010271.370000...
10	12	743925.0700000...
11	10	713727.0900000...
12	9	624814.0500000...

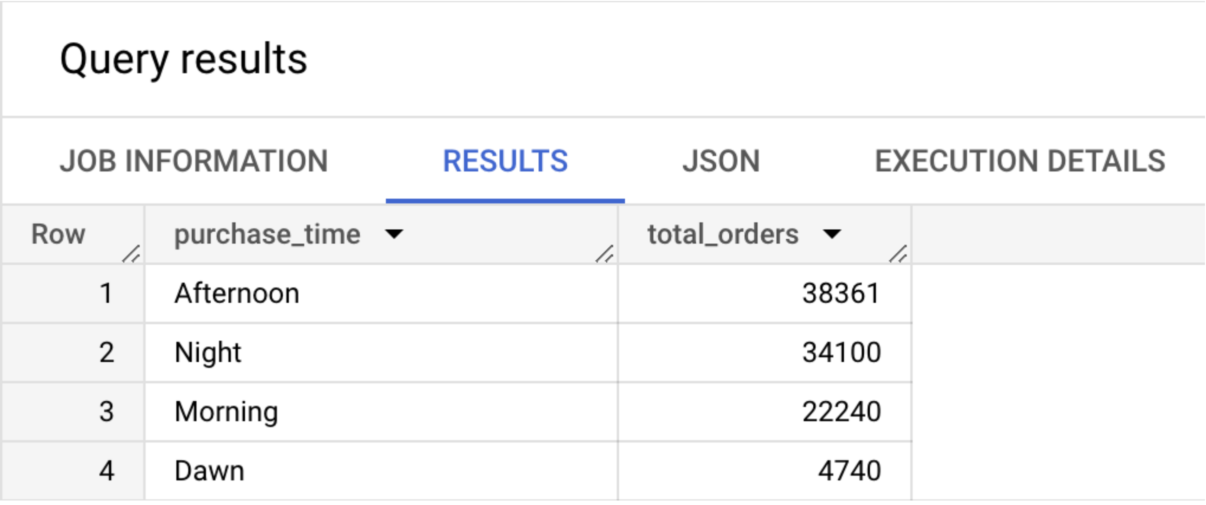
From the output we know which month (avg) had the best sales in all three years

- *‘Average -May(5th)’* month has highest sales
- If going from per month per year (using by month, by month) then *October to January is* Seasonality with peaks at these specific months

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

```
SELECT CASE
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 0 AND EXTRACT(HOUR FROM
order_purchase_timestamp) < 6 THEN 'Dawn'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 6 AND EXTRACT(HOUR FROM
order_purchase_timestamp) < 12 THEN 'Morning'
        WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 12 AND EXTRACT(HOUR
FROM order_purchase_timestamp) < 18 THEN 'Afternoon'
        ELSE 'Night'
      END AS purchase_time,
      COUNT(*) AS total_orders
FROM `target-sql-case-study-387518.target.orders`
GROUP BY purchase_time
ORDER BY total_orders DESC
```

Output:-



Query results

Row	purchase_time	total_orders
1	Afternoon	38361
2	Night	34100
3	Morning	22240
4	Dawn	4740

From the result we got to know the following:

- Brazilian customers tend to buy in '*Afternoon*'
- The lowest order comes at 'Dawn' time

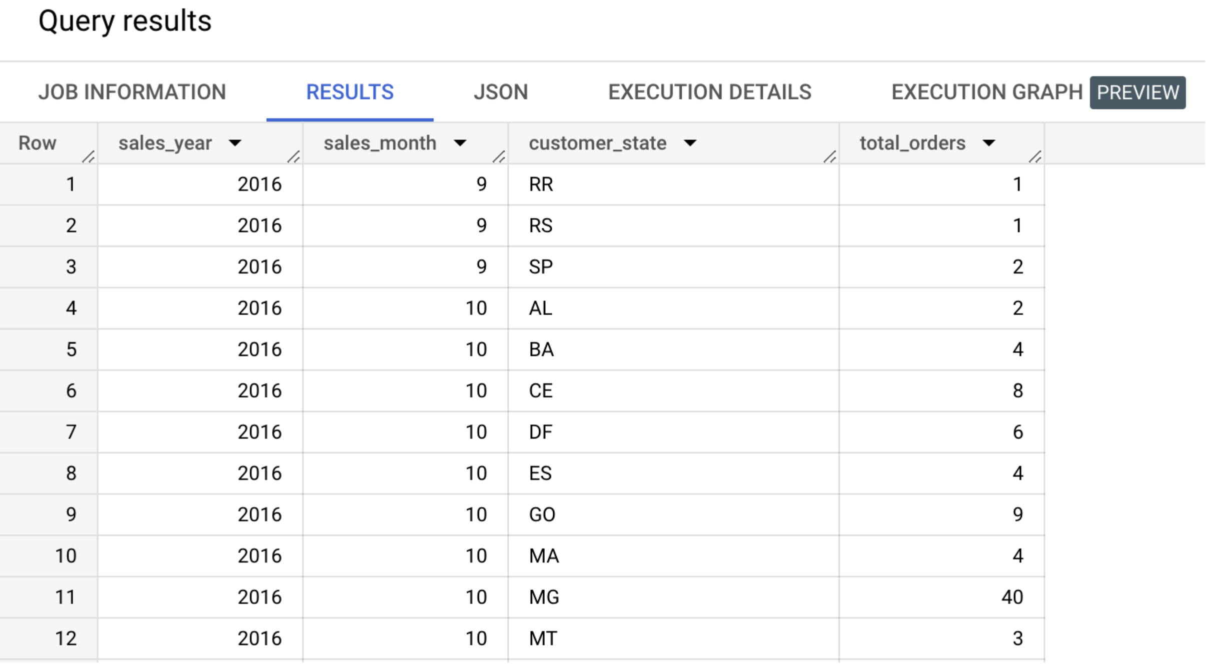
Q3. Evolution of E-commerce orders in the Brazil region:

1. ***Get month-on-month orders by states***

Ans:

```
SELECT
        EXTRACT(Year FROM o.order_purchase_timestamp) AS sales_year,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS
        sales_month,c.customer_state,
        COUNT(*) AS total_orders
FROM `target-sql-case-study-387518.target.orders` as o
Join `target-sql-case-study-387518.target.customers` as c
ON o.customer_id=c.customer_id
GROUP BY sales_year, sales_month, c.customer_state
ORDER BY sales_year, sales_month, c.customer_state
```

Output:-



Row	sales_year	sales_month	customer_state	total_orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4
9	2016	10	GO	9
10	2016	10	MA	4
11	2016	10	MG	40
12	2016	10	MT	3

From the result we got to know(using desc in order by):

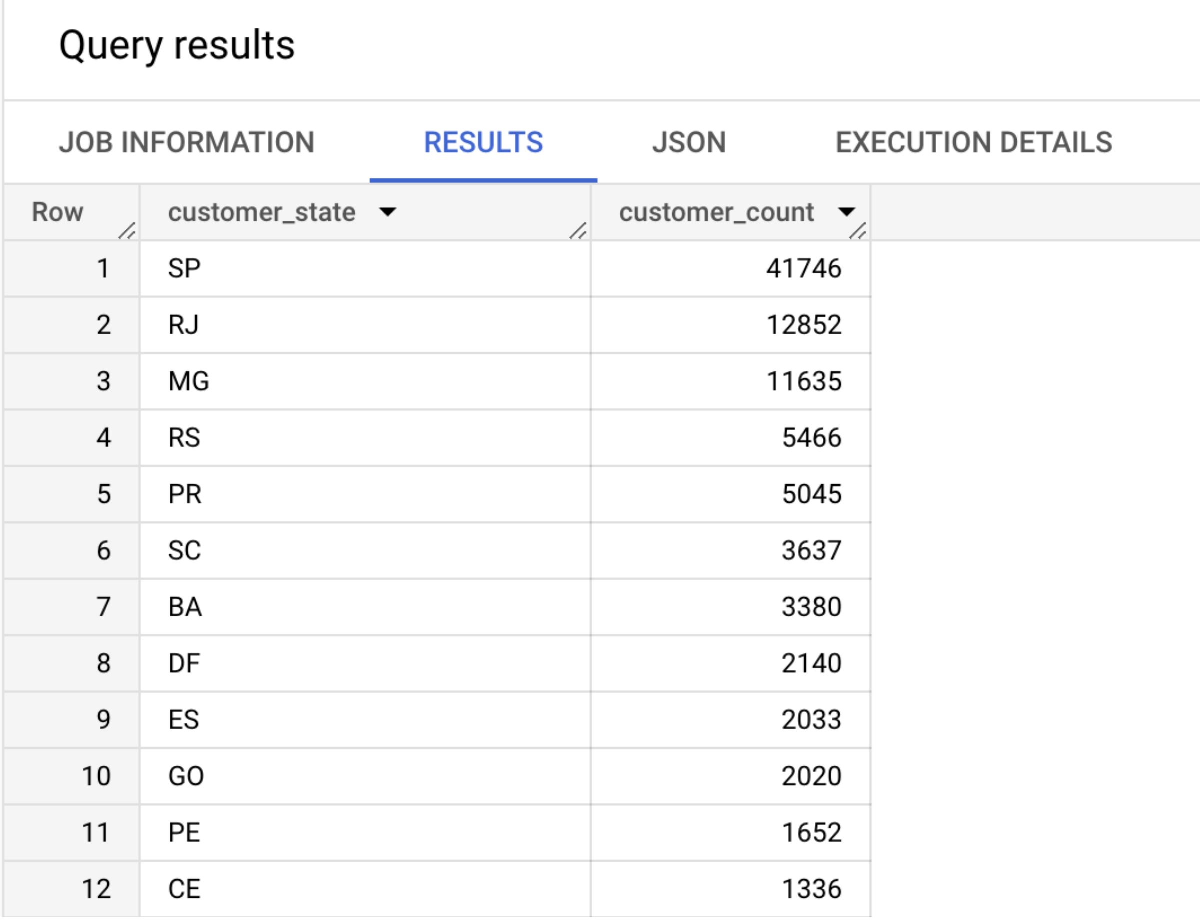
- The year 2018, 8th month, SP state, has the highest orders(3253 orders)

2. Distribution of customers across the states in Brazil

Ans

```
SELECT
        customer_state, COUNT(*) AS customer_count
FROM
        `target-sql-case-study-387518.target.customers`
GROUP BY customer_state
ORDER BY customer_count DESC
```

Output:-



Row	customer_state	customer_count
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637
7	BA	3380
8	DF	2140
9	ES	2033
10	GO	2020
11	PE	1652
12	CE	1336

From the result we got to know the following:

- SP State had the highest customer count

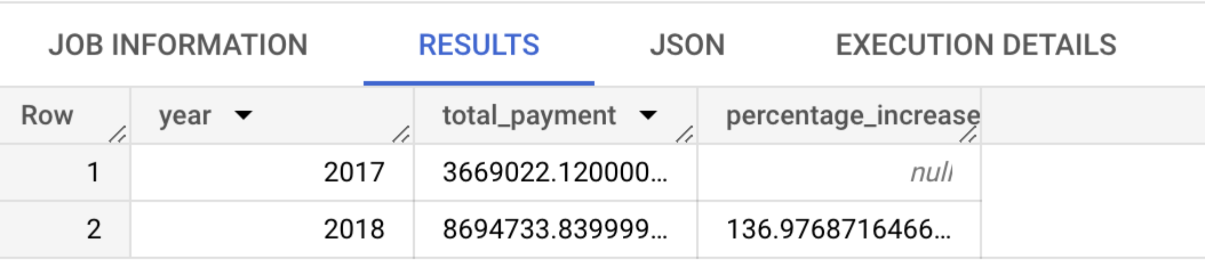
Q4. Impact on the Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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

Ans:

```
SELECT year,total_payment,

             (total_payment - LAG(total_payment) OVER (ORDER BY year)) /

             LAG(total_payment) OVER (ORDER BY year) * 100 AS percentage_increase

       FROM(

             SELECT

                    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,

                    SUM(p.payment_value) AS total_payment

             FROM `target-sql-case-study-387518.target.payments` p

             JOIN `target-sql-case-study-387518.target.orders` o

                    ON p.order_id = o.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 year

       ) order by year
```

Output:-



Row	year	total_payment	percentage_increase
1	2017	3669022.120000...	null
2	2018	8694733.839999...	136.9768716466...

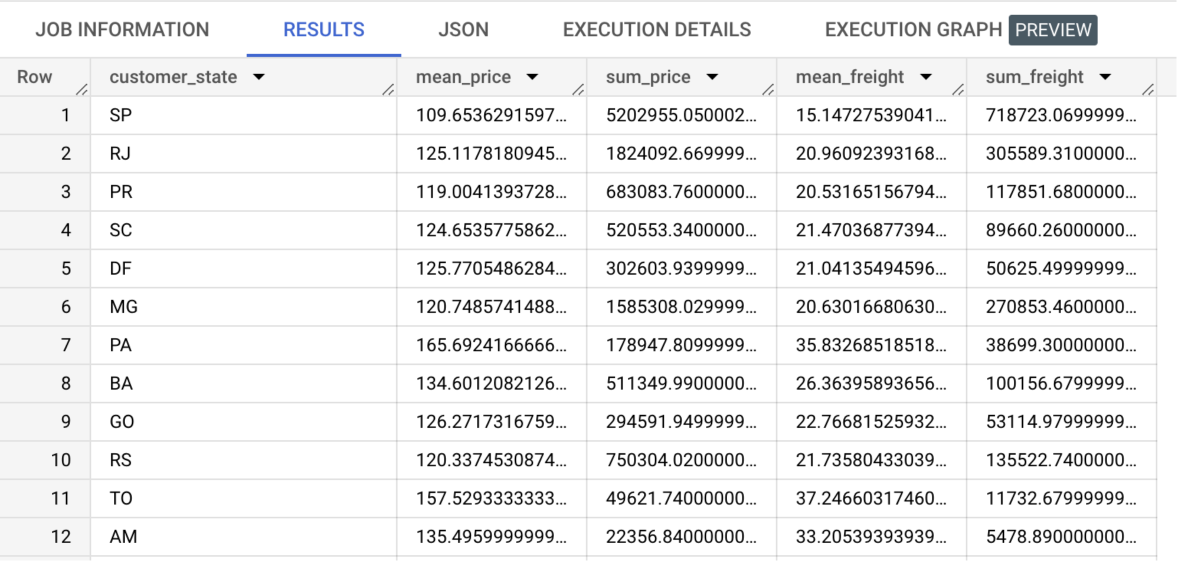
From the output, we know the following:

- *136.97% increase in the cost of orders from 2017 to 2018 (including months between Jan to Aug only)*
- We use LAG and SUBQUERY for the SQL query

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

ANS:
```
SELECT

        c.customer_state, avg(oi.price) as mean_price,

        sum(oi.price) as sum_price ,

        avg(oi.freight_value) as mean_freight,

        sum(oi.freight_value) as sum_freight

        from `target-sql-case-study-387518.target.orders` as o

        join `target-sql-case-study-387518.target.order_items` as oi

        on o.order_id=oi.order_id

        join `target-sql-case-study-387518.target.customers` c

        on  o.customer_id = c.customer_id

        group by customer_state
```

Output:-



Row	customer_state	mean_price	sum_price	mean_freight	sum_freight
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...
4	SC	124.6535775862...	520553.3400000...	21.47036877394...	89660.26000000...
5	DF	125.7705486284...	302603.9399999...	21.04135494596...	50625.49999999...
6	MG	120.7485741488...	1585308.029999...	20.63016680630...	270853.4600000...
7	PA	165.6924166666...	178947.8099999...	35.83268518518...	38699.30000000...
8	BA	134.6012082126...	511349.9900000...	26.36395893656...	100156.6799999...
9	GO	126.2717316759...	294591.9499999...	22.76681525932...	53114.97999999...
10	RS	120.3374530874...	750304.0200000...	21.73580433039...	135522.7400000...
11	TO	157.5293333333...	49621.74000000...	37.24660317460...	11732.67999999...
12	AM	135.4959999999...	22356.84000000...	33.20539393939...	5478.890000000...

From the output, we know the following:

- We added new columns using the aggregation functions AVG and SUM
- There is no same matching column for tables ‘customer’ and ‘order_items’, but there is one table between them that is ‘orders’ so we connect ‘order_items’ to ‘orders’ using *‘order_id’*, and the ‘orders’ to the ‘customer’ using *‘customer_id’*.

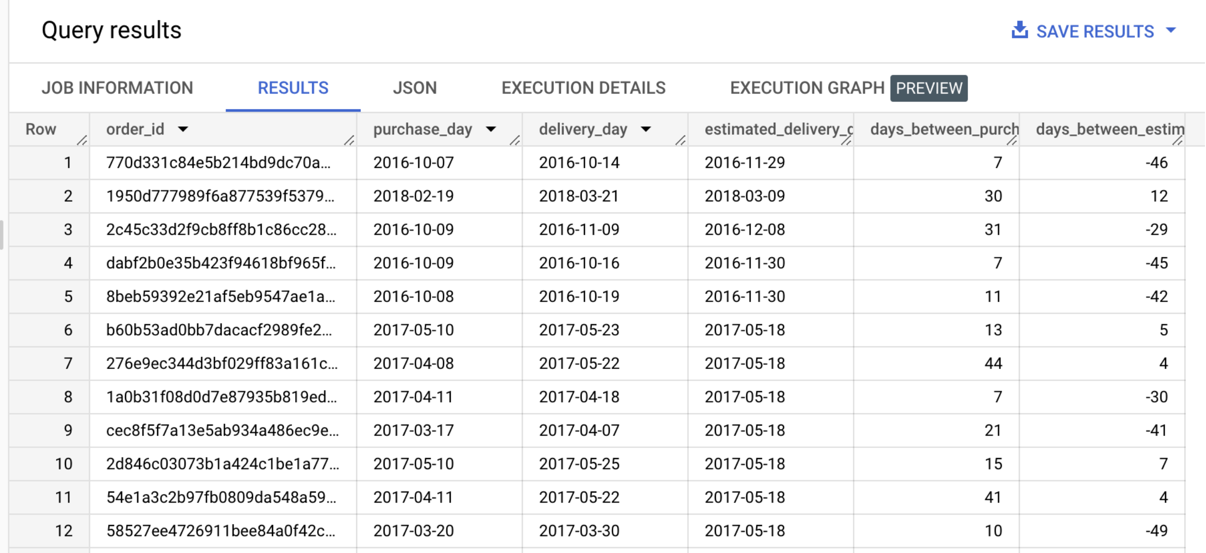
Q5. Analysis of sales, freight and delivery time

1. ***Calculate days between purchasing, delivering and estimated delivery***

Ans:

```
SELECT
    order_id,
    EXTRACT(DATE FROM order_purchase_timestamp) AS purchase_day,
    EXTRACT(DATE FROM order_delivered_customer_date) AS delivery_day,
    EXTRACT(DATE FROM order_estimated_delivery_date) AS
    estimated_delivery_day,
    date_diff(EXTRACT(DATE FROM
    order_delivered_customer_date),EXTRACT(DATE FROM
    order_purchase_timestamp), DAY) AS days_between_purchase_delivery,
Date_diff(EXTRACT(DATE FROM order_delivered_customer_date),EXTRACT(DATE FROM
order_estimated_delivery_date), DAY) AS days_between_estimated_delivery
FROM
    `target-sql-case-study-387518.target.orders`
```

Output:-



Row	order_id	purchase_day	delivery_day	estimated_delivery_c	days_between_purch	days_between_estim
1	770d331c84e5b214bd9dc70a...	2016-10-07	2016-10-14	2016-11-29	7	-46
2	1950d777989f6a877539f5379...	2018-02-19	2018-03-21	2018-03-09	30	12
3	2c45c33d2f9cb8ff8b1c86cc28...	2016-10-09	2016-11-09	2016-12-08	31	-29
4	dabf2b0e35b423f94618bf965f...	2016-10-09	2016-10-16	2016-11-30	7	-45
5	8beb59392e21af5eb9547ae1a...	2016-10-08	2016-10-19	2016-11-30	11	-42
6	b60b53ad0bb7dacacf2989fe2...	2017-05-10	2017-05-23	2017-05-18	13	5
7	276e9ec344d3bf029ff83a161c...	2017-04-08	2017-05-22	2017-05-18	44	4
8	1a0b31f08d0d7e87935b819ed...	2017-04-11	2017-04-18	2017-05-18	7	-30
9	cec8f5f7a13e5ab934a486ec9e...	2017-03-17	2017-04-07	2017-05-18	21	-41
10	2d846c03073b1a424c1be1a77...	2017-05-10	2017-05-25	2017-05-18	15	7
11	54e1a3c2b97fb0809da548a59...	2017-04-11	2017-05-22	2017-05-18	41	4
12	58527ee4726911bee84a0f42c...	2017-03-20	2017-03-30	2017-05-18	10	-49

From the output,

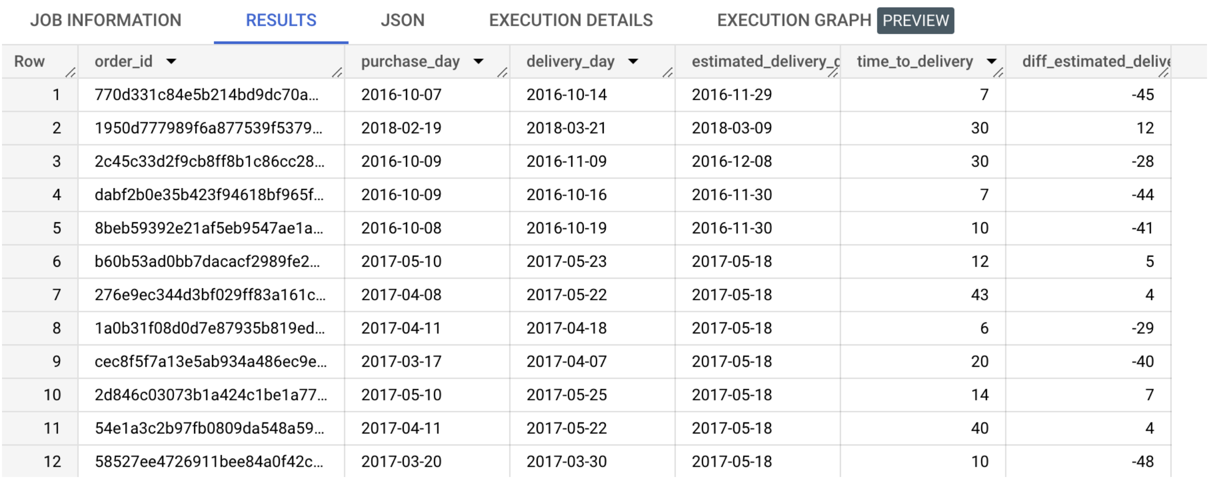
- **we know the days between purchase and delivery, and purchase and estimated delivery**
- **For example, 1st order_id was delivered before 46 days of estimated so it has ‘-46’**
- **We can also add days between purchase_day and estimate_delivery_day also, using the same ‘Date_Diff’**

2. ***Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:***
    a. ***time_to_delivery = order_delivered_customer_date-order_purchase_timestamp***
    b. ***diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date***

Ans:

```
SELECT
   order_id,
   EXTRACT(DATE FROM order_purchase_timestamp) AS purchase_day,
   EXTRACT(DATE FROM order_delivered_customer_date) AS delivery_day,
   EXTRACT(DATE FROM order_estimated_delivery_date) AS estimated_delivery_day,
   TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY)
    AS time_to_delivery,
   TIMESTAMP_DIFF(order_delivered_customer_date, order_estimated_delivery_date,DAY)
    AS diff_estimated_delivery
FROM
   `target-sql-case-study-387518.target.orders`
```

Output:-



Row	order_id	purchase_day	delivery_day	estimated_delivery_c	time_to_delivery	diff_estimated_delive
1	770d331c84e5b214bd9dc70a...	2016-10-07	2016-10-14	2016-11-29	7	-45
2	1950d777989f6a877539f5379...	2018-02-19	2018-03-21	2018-03-09	30	12
3	2c45c33d2f9cb8ff8b1c86cc28...	2016-10-09	2016-11-09	2016-12-08	30	-28
4	dabf2b0e35b423f94618bf965f...	2016-10-09	2016-10-16	2016-11-30	7	-44
5	8beb59392e21af5eb9547ae1a...	2016-10-08	2016-10-19	2016-11-30	10	-41
6	b60b53ad0bb7dacacf2989fe2...	2017-05-10	2017-05-23	2017-05-18	12	5
7	276e9ec344d3bf029ff83a161c...	2017-04-08	2017-05-22	2017-05-18	43	4
8	1a0b31f08d0d7e87935b819ed...	2017-04-11	2017-04-18	2017-05-18	6	-29
9	cec8f5f7a13e5ab934a486ec9e...	2017-03-17	2017-04-07	2017-05-18	20	-40
10	2d846c03073b1a424c1be1a77...	2017-05-10	2017-05-25	2017-05-18	14	7
11	54e1a3c2b97fb0809da548a59...	2017-04-11	2017-05-22	2017-05-18	40	4
12	58527ee4726911bee84a0f42c...	2017-03-20	2017-03-30	2017-05-18	10	-48

From the output,

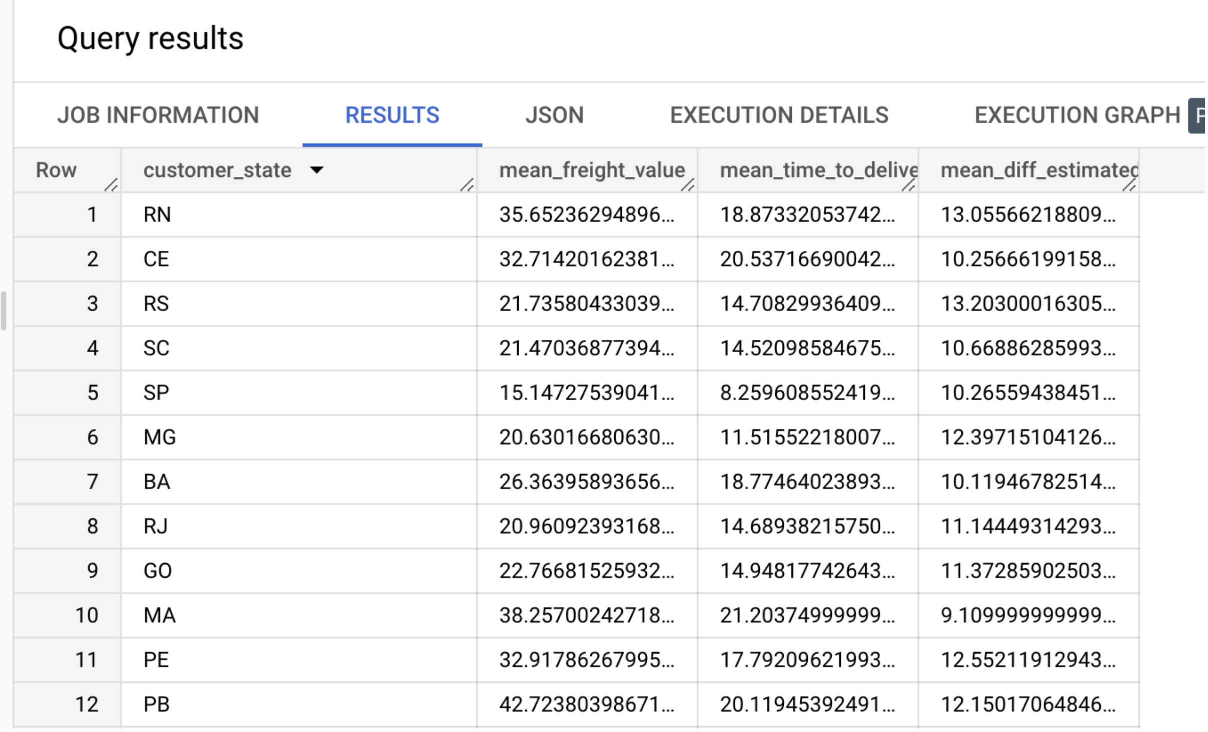
- **we know the days between purchase and delivery, and purchase and estimated delivery using the given formula**
- **Negative values means before delivery**

- **Example: First order delivered before 45 days of estimation_delivery so it is in negative '-45', (we can make all numbers positive using 'ABS' function)**

3. ***Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_deliver***

Ans:
```
SELECT
             c.customer_state,
             AVG(oi.freight_value) as mean_freight_value,
             AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
             order_purchase_timestamp, DAY)) AS mean_time_to_delivery,
             AVG(TIMESTAMP_DIFF(o.order_estimated_delivery_date,
             order_delivered_customer_date, DAY)) AS mean_diff_estimated_delivery
      FROM `target-sql-case-study-387518.target.orders` as o
      join `target-sql-case-study-387518.target.customers` as c
      on  o.customer_id=c.customer_id
      join `target-sql-case-study-387518.target.order_items` as oi
      on o.order_id=oi.order_id
      group by c.customer_state
```

Output:-



Row	customer_state	mean_freight_value	mean_time_to_delive	mean_diff_estimated
1	RN	35.65236294896...	18.87332053742...	13.05566218809...
2	CE	32.71420162381...	20.53716690042...	10.25666199158...
3	RS	21.73580433039...	14.70829936409...	13.20300016305...
4	SC	21.47036877394...	14.52098584675...	10.66886285993...
5	SP	15.14727539041...	8.259608552419...	10.26559438451...
6	MG	20.63016680630...	11.51552218007...	12.39715104126...
7	BA	26.36395893656...	18.77464023893...	10.11946782514...
8	RJ	20.96092393168...	14.68938215750...	11.14449314293...
9	GO	22.76681525932...	14.94817742643...	11.37285902503...
10	MA	38.25700242718...	21.20374999999...	9.109999999999...
11	PE	32.91786267995...	17.79209621993...	12.55211912943...
12	PB	42.72380398671...	20.11945392491...	12.15017064846...

From the output, we did:

- **We group data by state**
- **We did two join as there is no same matching column for tables ‘customer’ and ‘order_items’, but there is one table between them that is ‘orders’ so we connect ‘order_items’ to ‘orders’ using ‘order_id’, and the ‘orders’ to the ‘customer’ using ‘customer_id’.**

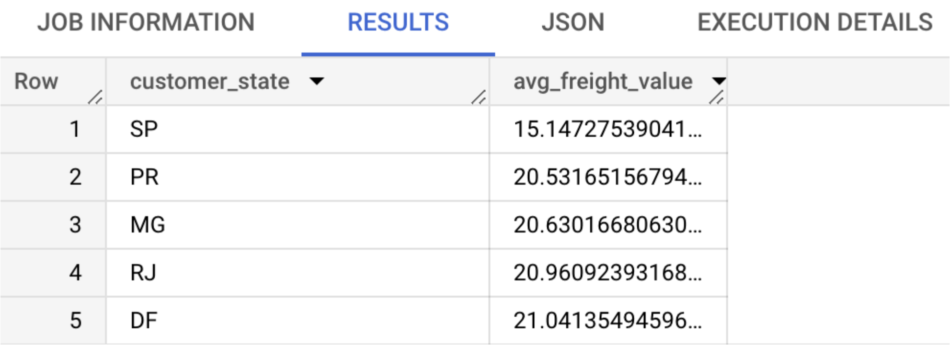
4. **Sort the data to get the following:**
    a. **Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5**

Ans:

- *Lowest Average Freight Value:*

```
SELECT distinct c.customer_state, avg(oi.freight_value) as avg_freight_value
from `target-sql-case-study-387518.target.order_items`as oi
join `target-sql-case-study-387518.target.orders` as o on
oi.order_id=o.order_id
join `target-sql-case-study-387518.target.customers` as c on o.customer_id =
c.customer_id
group by customer_state
order by avg_freight_value
limit 5
```

Output:-



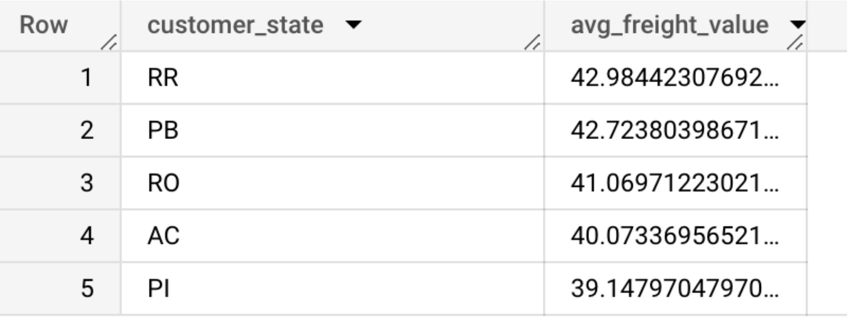
Row	customer_state	avg_freight_value
1	SP	15.14727539041...
2	PR	20.53165156794...
3	MG	20.63016680630...
4	RJ	20.96092393168...
5	DF	21.04135494596...

From the output, we got to know that ‘SP state’ has the lowest Average Freight Value among all-state

- *Highest Average Freight Value:*

```
SELECT distinct c.customer_state, avg(oi.freight_value) as avg_freight_value
from `target-sql-case-study-387518.target.order_items`as oi
join `target-sql-case-study-387518.target.orders` as o on
oi.order_id=o.order_id
join `target-sql-case-study-387518.target.customers` as c on o.customer_id =
c.customer_id
group by customer_state
order by avg_freight_value desc
limit 5
```

Output:



Row	customer_state	avg_freight_value
1	RR	42.98442307692...
2	PB	42.72380398671...
3	RO	41.06971223021...
4	AC	40.07336956521...
5	PI	39.14797047970...

From the output, we got to know that 'RR state' has the Highest Average Freight Value among all-state

b. Top 5 states with highest/lowest average time to delivery

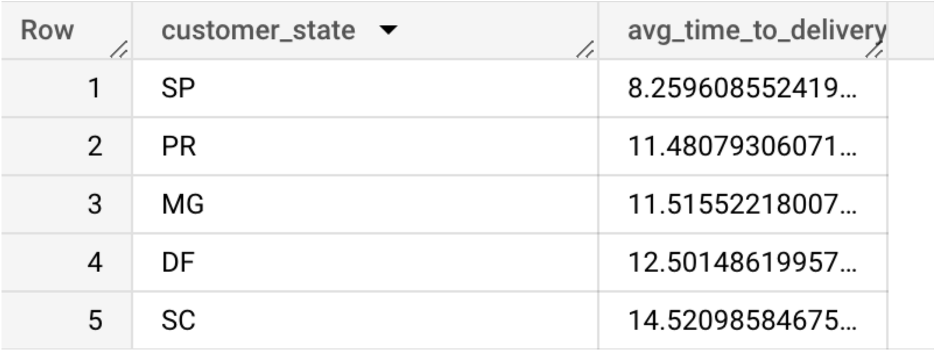
Ans:

- *Lowest Average time to delivery:*

```
SELECT distinct c.customer_state,
AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
order_purchase_timestamp, DAY)) AS avg_time_to_delivery
from `target-sql-case-study-387518.target.order_items`as oi
join `target-sql-case-study-387518.target.orders` as o
on oi.order_id=o.order_id
join `target-sql-case-study-387518.target.customers` as c
on o.customer_id = c.customer_id
```

```
group by customer_state
order by avg_time_to_delivery
limit 5
```

Output:



Row	customer_state	avg_time_to_delivery
1	SP	8.259608552419...
2	PR	11.48079306071...
3	MG	11.51552218007...
4	DF	12.50148619957...
5	SC	14.52098584675...

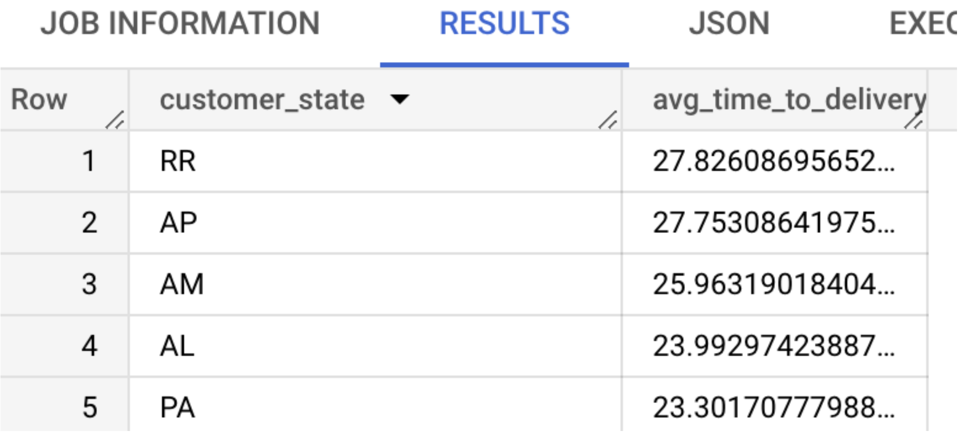
From the output,

- ***we got to know that SP state' has the Lowest Average time to deliver among all-state***
- ***With an average of 8.25 days***

- *Highest Average time to delivery:*

```
SELECT distinct c.customer_state,
AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
order_purchase_timestamp, DAY)) AS avg_time_to_delivery
from `target-sql-case-study-387518.target.order_items`as oi
join `target-sql-case-study-387518.target.orders` as o
on oi.order_id=o.order_id
join `target-sql-case-study-387518.target.customers` as c
on o.customer_id = c.customer_id
group by customer_state
order by avg_time_to_delivery desc
limit 5
```

Output:



Row	customer_state	avg_time_to_delivery
1	RR	27.82608695652...
2	AP	27.75308641975...
3	AM	25.96319018404...
4	AL	23.99297423887...
5	PA	23.30170777988...

From the output,

- **We got to know that the RR state' has the Highest Average time to deliver among all-state**
- **With an average 27.82 days**

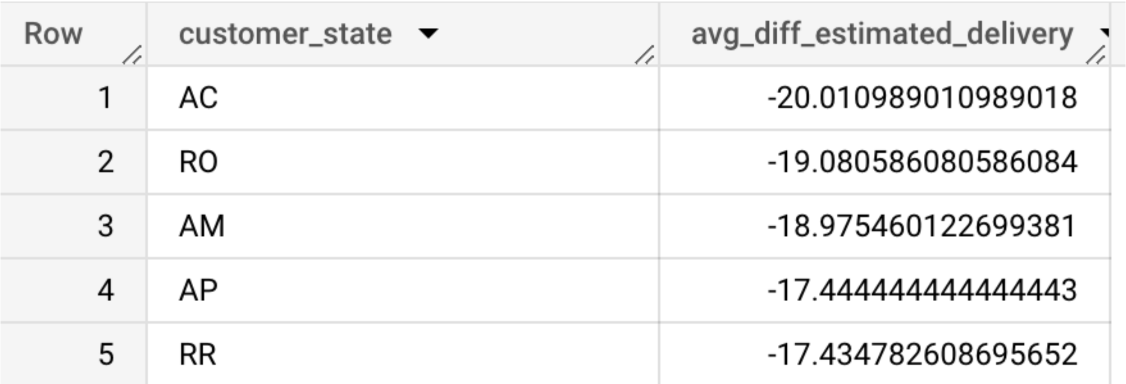
c. Top 5 states where delivery is really fast/ not so fast compared to the estimated date

Ans:

- *Fastest Delivery states compared to estimated date:*

```
SELECT distinct c.customer_state,
AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date,DAY)) AS avg_diff_estimated_delivery
from `target-sql-case-study-387518.target.order_items`as oi
join `target-sql-case-study-387518.target.orders` as o
on oi.order_id=o.order_id
join `target-sql-case-study-387518.target.customers` as c
on o.customer_id = c.customer_id
group by customer_state
ORDER BY diff_estimated_delivery
limit 5
```

Output:



Row	customer_state	avg_diff_estimated_delivery
1	AC	-20.010989010989018
2	RO	-19.080586080586084
3	AM	-18.975460122699381
4	AP	-17.444444444444443
5	RR	-17.434782608695652

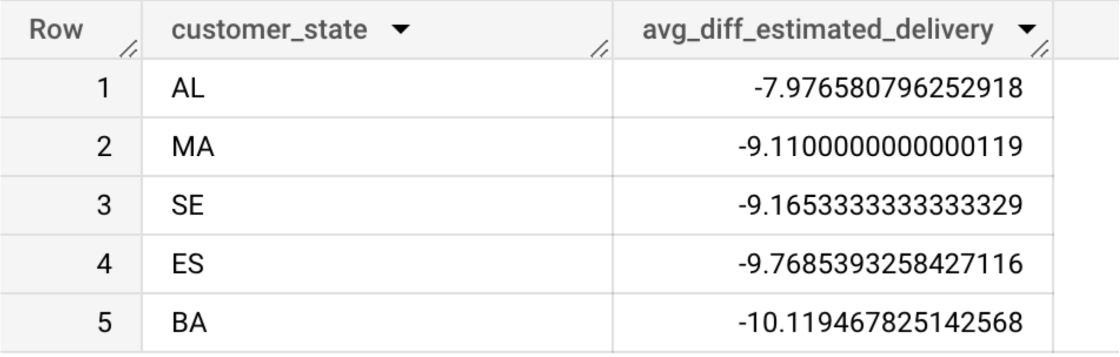
From the output,

- **We got to know that 'AC state' has the Fastest Delivery state compared to the estimated date among all-state with**
- **An average difference 20.01 days delivered before the estimate_delivery time**

- *Not so Fastest Delivery states compared to estimated date:*

```
SELECT distinct c.customer_state,
AVG(TIMESTAMP_DIFF(o.order_delivered_customer_date,
o.order_estimated_delivery_date,DAY)) AS avg_diff_estimated_delivery
from `target-sql-case-study-387518.target.order_items`as oi
join `target-sql-case-study-387518.target.orders` as o
on oi.order_id=o.order_id
join `target-sql-case-study-387518.target.customers` as c
on o.customer_id = c.customer_id
group by customer_state
ORDER BY avg_diff_estimated_delivery desc
limit 5
```

Output:



Row	customer_state	avg_diff_estimated_delivery
1	AL	-7.976580796252918
2	MA	-9.1100000000000119
3	SE	-9.1653333333333329
4	ES	-9.7685393258427116
5	BA	-10.119467825142568

From the output, we got to know

- **That 'AL state' has the Not-so fastest Delivery state compared to the estimated date among all-state**
- **With an average difference *7.976 days* delivered before the estimate_delivery time**

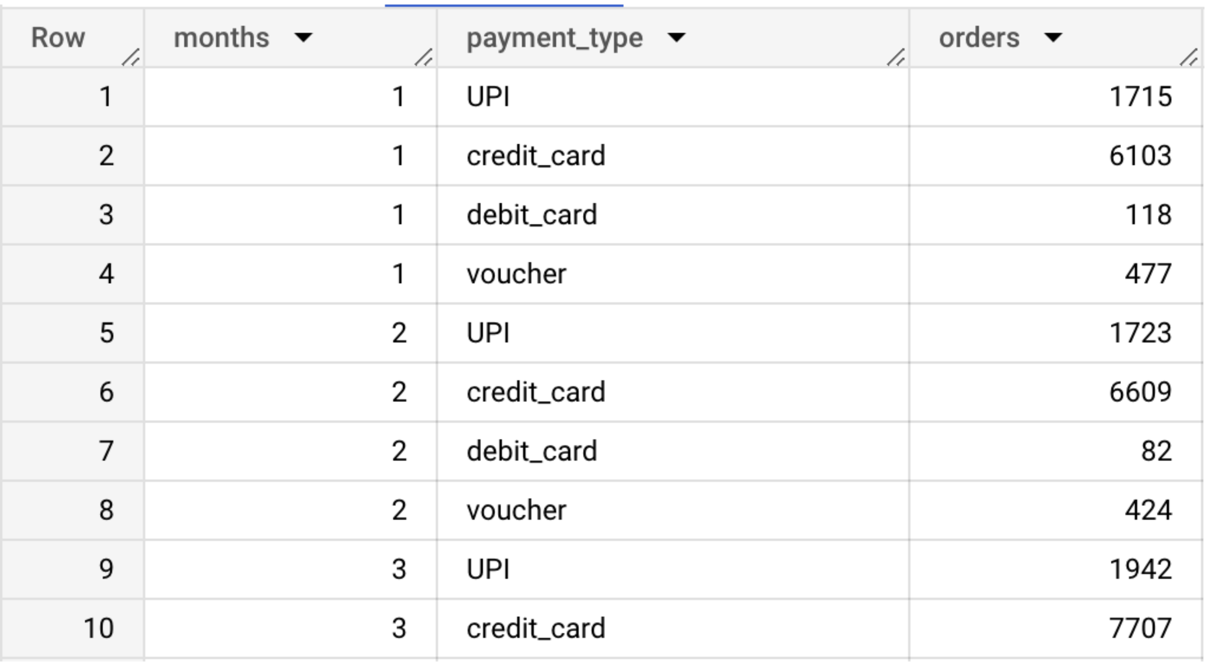
Q6. Payment type analysis:

1. **Month over Month count of orders for different payment types**

Ans:

```
SELECT

       extract(month from o.order_purchase_timestamp) as
       months,p.payment_type,count(o.order_id) as orders

from `target-sql-case-study-387518.target.orders` as o

join `target-sql-case-study-387518.target.payments` as p on o.order_id=p.order_id

group by months, payment_type

order by months,payment_type
```

Output:



Row	months	payment_type	orders
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707

From the output,

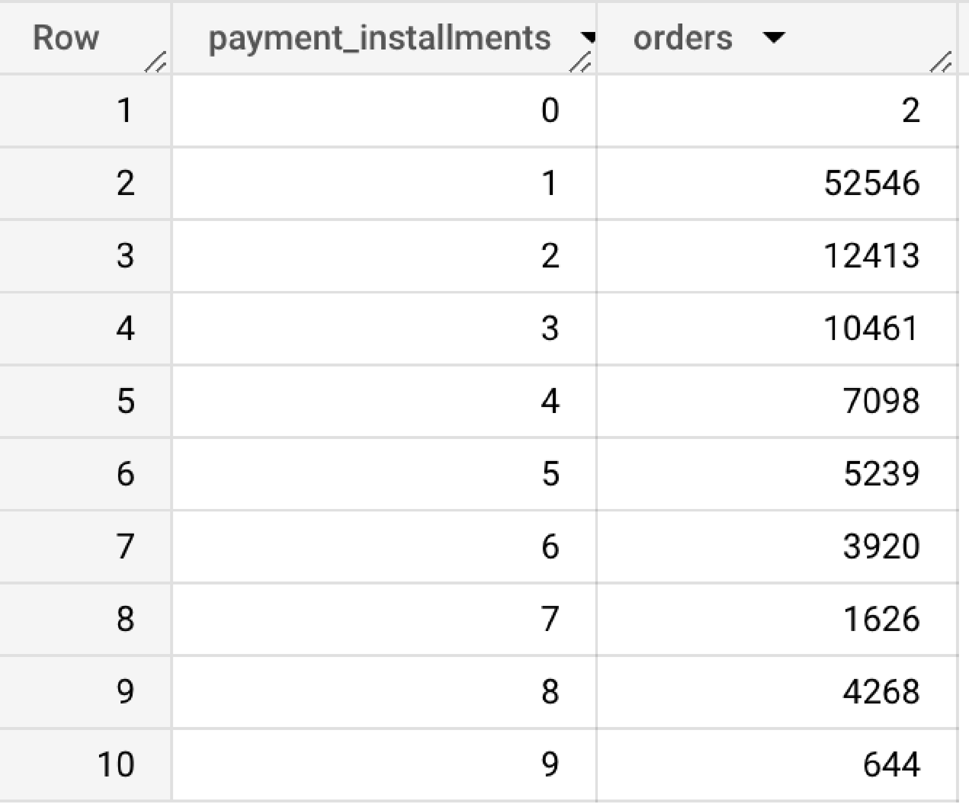
- **We got to know per month-on-month payment_type and how many orders we get it**
- **Credit card is the highest used payment method**

2. **Count of orders based on the no. of payment installments**

Ans:

```
SELECT p.payment_installments,count(o.order_id) as orders
from `target-sql-case-study-387518.target.orders` as o
join `target-sql-case-study-387518.target.payments` as p on o.order_id=p.order_id
group by payment_installments
order by payment_installments
```

Output:



Row	payment_installments	orders
1	0	2
2	1	52546
3	2	12413
4	3	10461
5	4	7098
6	5	5239
7	6	3920
8	7	1626
9	8	4268
10	9	644

We got to know the following:

- **zero or without payment_installments are 2, which is the lowest**
- **And 1 or 2-month payment_installments are the highest**

Summary, Insights and Recommendations:

1. **Data is from 4th September 2016 to 17th October 2018**

2. **October to January had a good number of sales, but May had the highest sales in total.**

3. **‘Fixed telephony’ has the highest payment value**

4. **‘SP state’ has the highest customer count (41746) and ‘Garden tool’ is the highest ordered product category in ‘SP state’. SP state has also the lowest freight value and lowest avg time to deliver orders**
   So, SP state is the gold mine for the company, we should increase marketing and other things there.

5. **RR state took the highest time to deliver compared to the estimate and also had the highest freight value. AL state is also near the estimated delivery**
   We should work with the shipping/delivery partner and resolve the issue

6. **Credit-card is highest used for payment and one or two-time payment installments had the highest orders in term of installment payment**
   So, we should provide some no-cost EMI offers for 1-3 intallment payments

1.