Target SQL

Problem Statement:

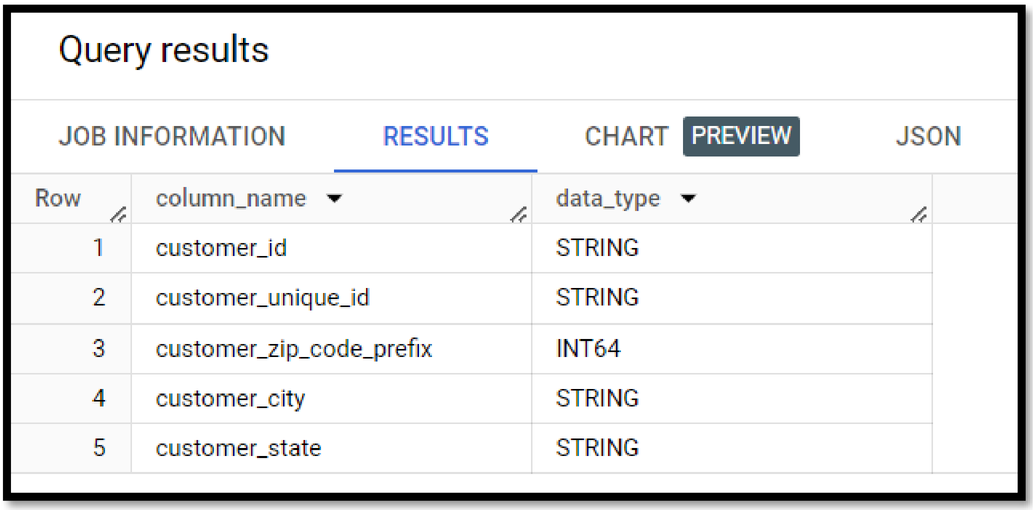
Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

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.

Answer: SQL Query

```
SELECT column_name, data_type

FROM
`target-sql-case-407005.target_sql.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```



Query results

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

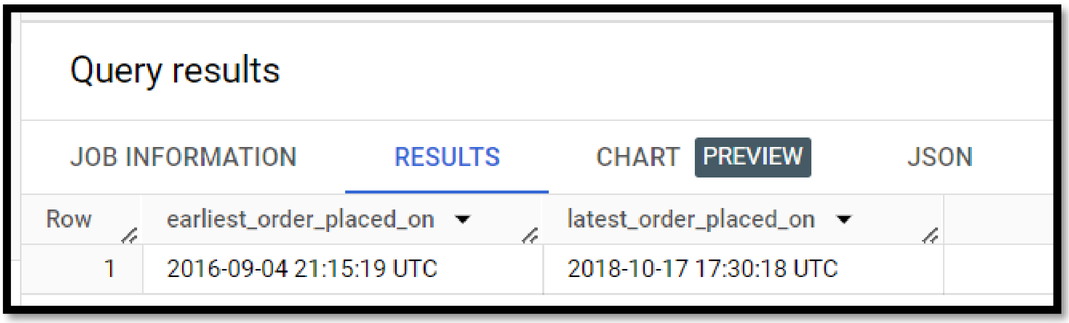
INSIGHTS:

It is observed that 4 out of 5 columns have STRING as their datatype and the remaining 1 column has INTEGER datatype

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

Answer: SQL Query

```
SELECT MIN(order_purchase_timestamp) AS earliest_order_placed_on,
      MAX(order_purchase_timestamp) AS latest_order_placed_on
FROM `target_sql.orders`
```



Query results

Row	earliest_order_placed_on	latest_order_placed_on
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC

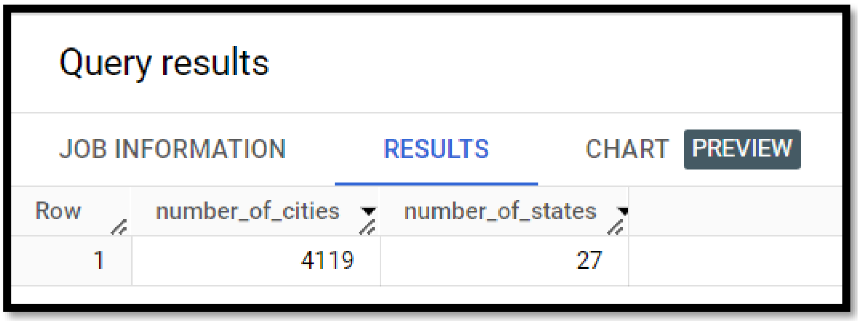
INSIGHTS:

It can be seen that the first order was placed on September 4, 2016 while the latest order was placed on October 17, 2018. Hence September 4, 2016 to October 17, 2018 is the time range within which orders were placed.

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

Answer: SQL Query

```
SELECT COUNT(DISTINCT c.customer_city) AS number_of_cities,
       COUNT(DISTINCT c.customer_state) AS number_of_states
FROM `target_sql.orders` o
JOIN `target_sql.customers` c
ON o.customer_id = c.customer_id
```



Row	number_of_cities	number_of_states
1	4119	27

INSIGHTS:

It can be observed that customers from 4119 different cities which are in 27 different states placed orders between the given time frame i.e. from September 4, 2016 to October 17, 2018.

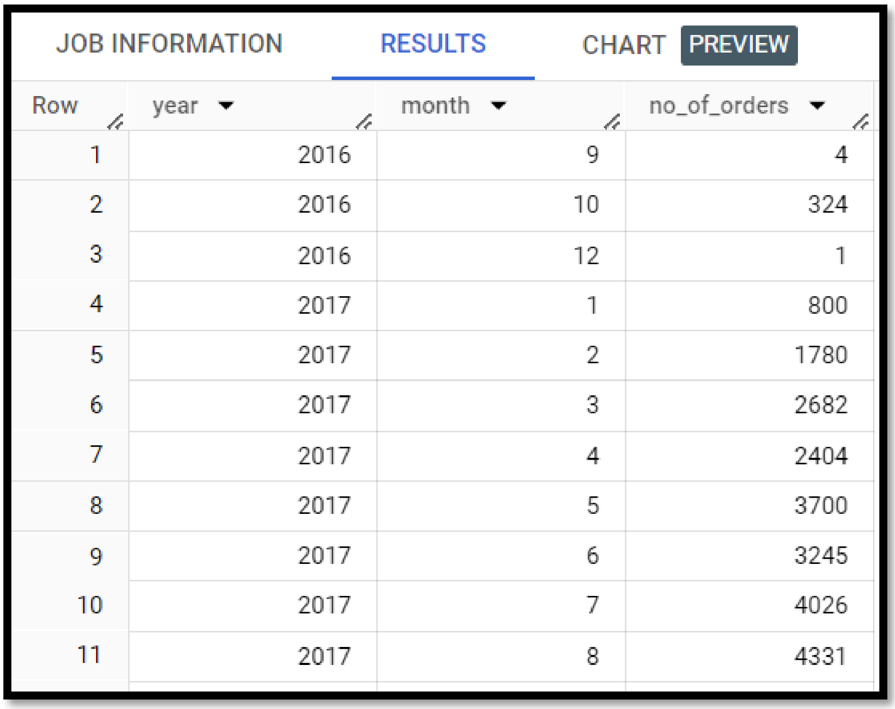
II. In-depth Exploration:

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

Answer: SQL Query

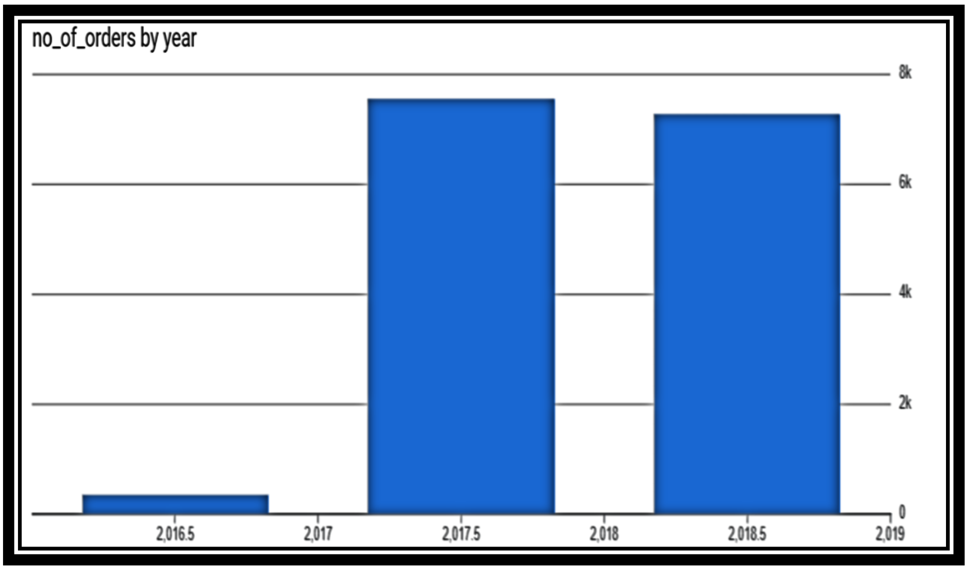
```
SELECT EXTRACT(year FROM order_purchase_timestamp) AS year,
      EXTRACT(month FROM order_purchase_timestamp) AS month,
      COUNT(1) AS no_of_orders
FROM `target_sql.orders`
GROUP BY year, month
ORDER BY year, month
```

Query Output



Row	year	month	no_of_orders
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026
11	2017	8	4331

INSIGHTS:



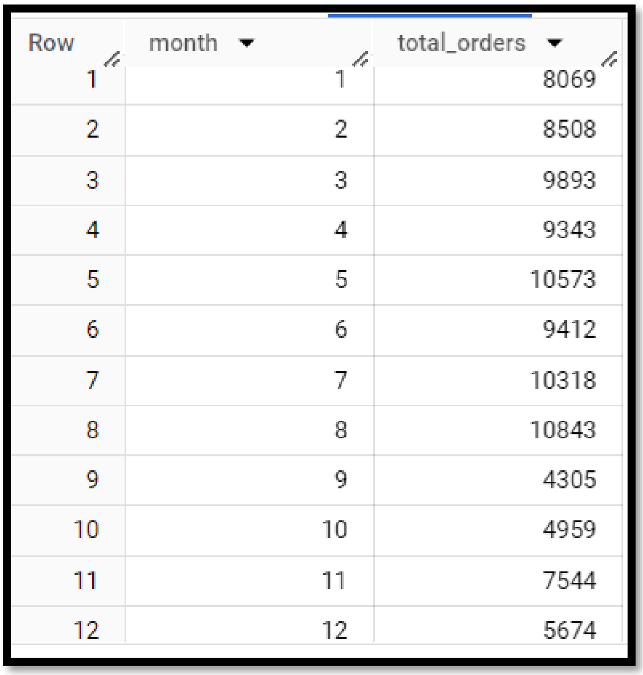
It can be observed that there is a growing trend in terms of number of orders placed. This trend can be observed from December 2016 to March 2018. The numbers of orders remain decrease hence with a slight increase noticed in August 2018. The number of orders slump to only 16 in September 2018 and further to 4 till October 17, 2018.

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

Answer: SQL Query

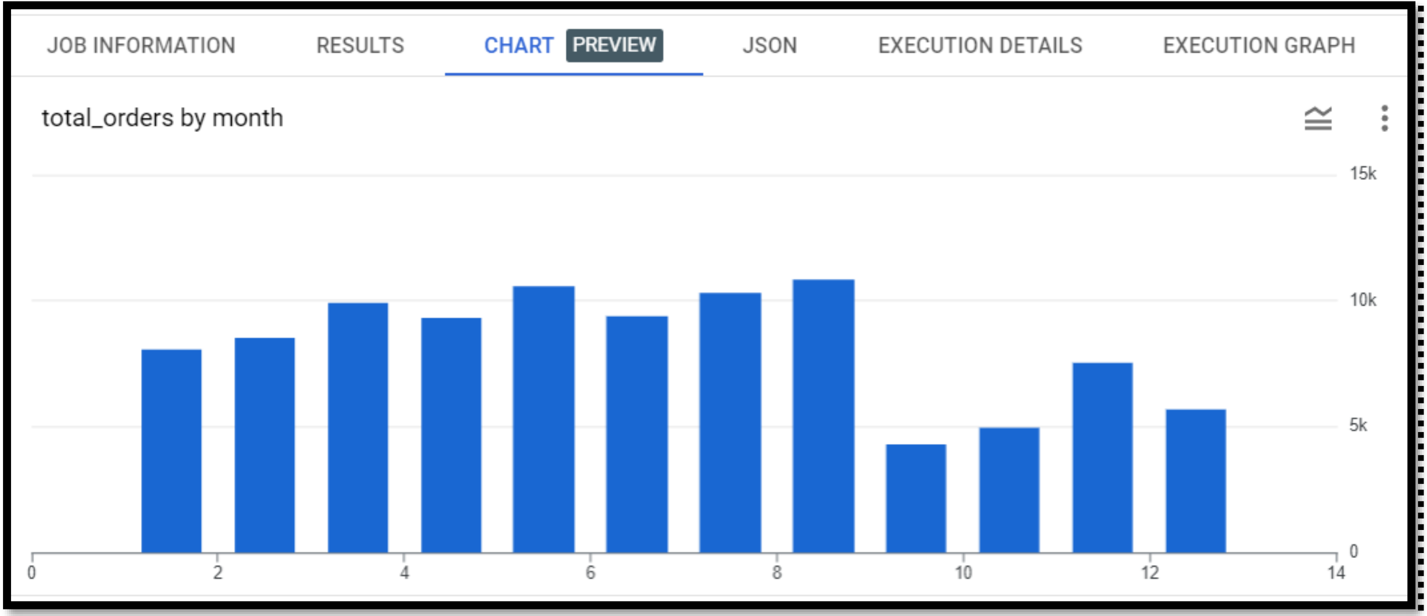
```
SELECT EXTRACT(month FROM order_purchase_timestamp) AS month,
       COUNT(order_id) AS total_orders
FROM `target_sql.orders`
GROUP BY month
ORDER BY month;
```

Query Output



Row	month	total_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305
10	10	4959
11	11	7544
12	12	5674

INSIGHTS:



x axis: months
y axis: no of orders

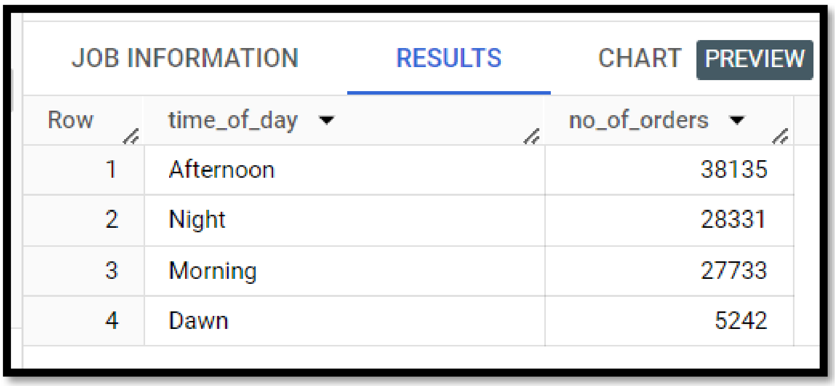
There is a seasonality in terms of number of orders being placed. The number of orders placed between March to August are significantly greater than the number of orders placed between September to February. The sales are quite low in the last 4 months of an year.

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

- **0-6 hrs : Dawn**
- **7-12 hrs : Mornings**
- **13-18 hrs : Afternoon**
- **19-23 hrs : Night**

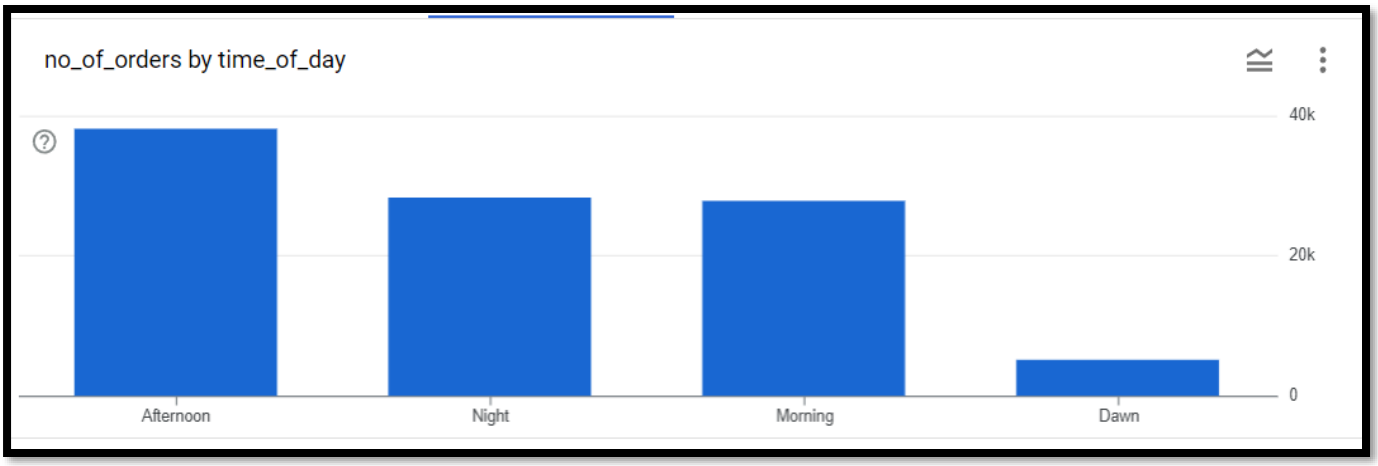
```
SELECT

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 'Morning'
  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_day,
COUNT(*) AS no_of_orders
FROM `target_sql.orders`
GROUP BY time_of_day
ORDER BY no_of_orders DESC;
```

Query Output



Row	time_of_day	no_of_orders
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

INSIGHTS:



It can be derived from the above chart that majority of orders are placed by customers during the afternoon, spending on ad and marketing campaigns can be done accordingly. The number of orders during night and morning are similar while the least number of orders are placed during first 6 hours of the day i.e. dawn in this case.

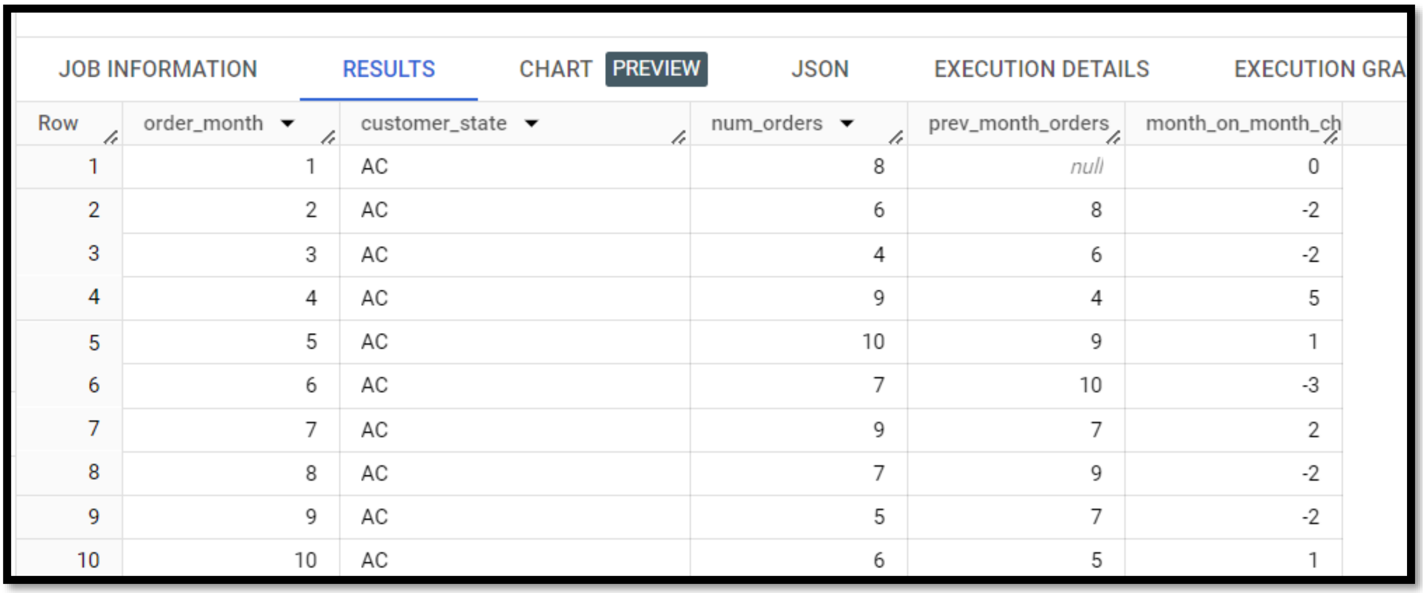
III. Evolution of E-commerce orders in the Brazil region:

A. Get the month on month no. of orders placed in each state.

Answer: SQL Query

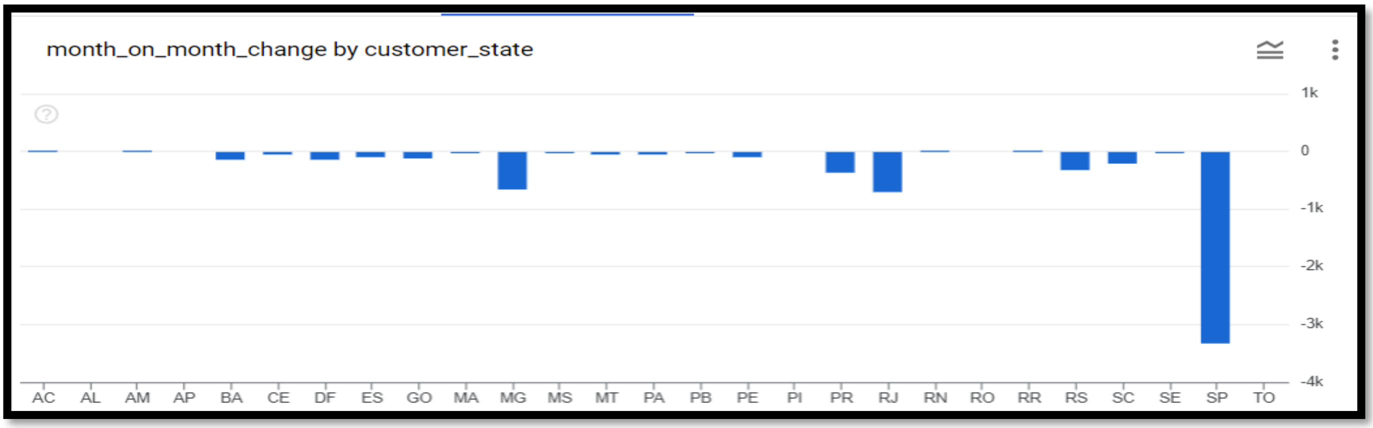
```
WITH MonthlyOrders AS (
    SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
        c.customer_state,
        COUNT(*) AS num_orders
    FROM `target_sql.orders` o
    JOIN `target_sql.customers` c
        ON o.customer_id = c.customer_id
    GROUP BY c.customer_state, order_month
)
SELECT order_month,
    customer_state,
    num_orders,
    LAG(num_orders) OVER (PARTITION BY customer_state ORDER BY order_month) AS
prev_month_orders,
    COALESCE(num_orders - LAG(num_orders) OVER (PARTITION BY customer_state
ORDER BY order_month), 0) AS month_on_month_change
FROM MonthlyOrders
ORDER BY customer_state, order_month
```

Query Output



Row	order_month	customer_state	num_orders	prev_month_orders	month_on_month_ch
1	1	AC	8	null	0
2	2	AC	6	8	-2
3	3	AC	4	6	-2
4	4	AC	9	4	5
5	5	AC	10	9	1
6	6	AC	7	10	-3
7	7	AC	9	7	2
8	8	AC	7	9	-2
9	9	AC	5	7	-2
10	10	AC	6	5	1

INSIGHTS:



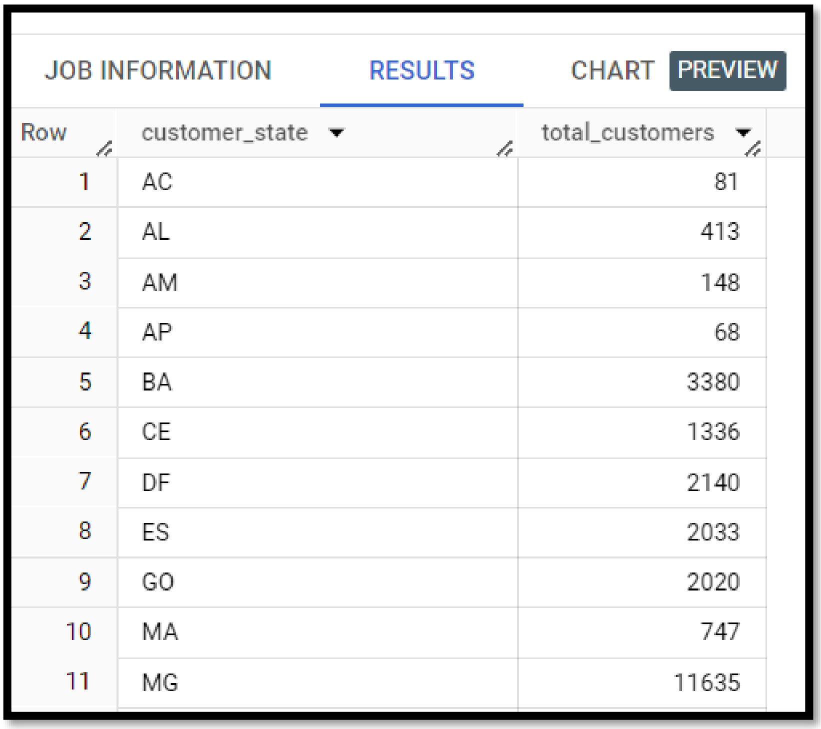
From the above chart it can be seen that see maximum month on month variation in SP state. The states MG, PR, RS, SC and RJ also have noticeable variation month wise while other states do not display such variation in terms of the number of orders month wise.

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

Answer: SQL Query

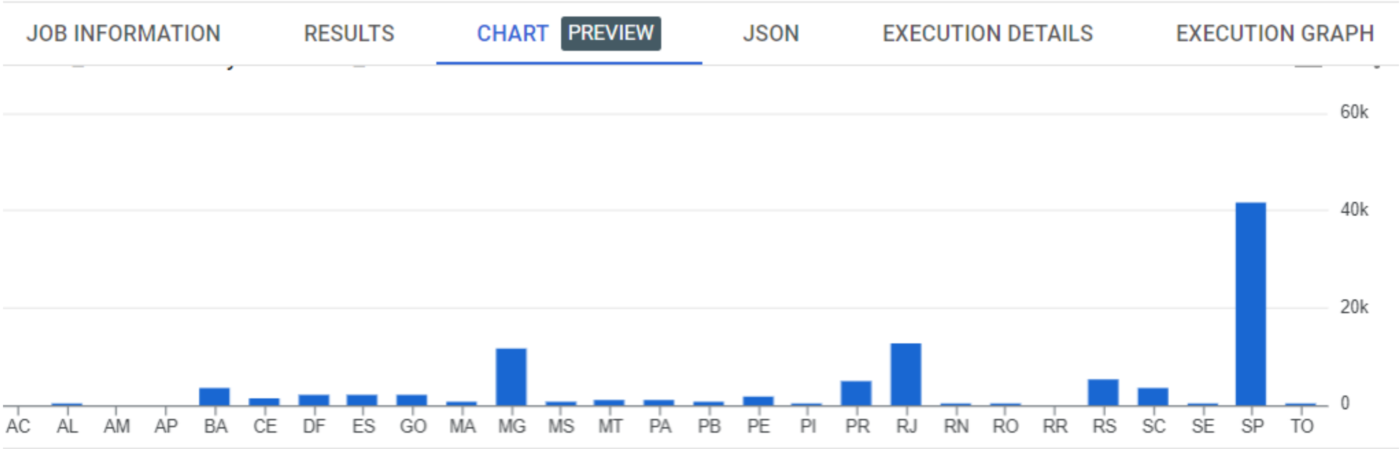
```
SELECT customer_state, COUNT(*) AS total_customers
FROM `target_sql.customers`
GROUP BY customer_state
ORDER BY customer_state;
```

Query Output



Row	customer_state	total_customers
1	AC	81
2	AL	413
3	AM	148
4	AP	68
5	BA	3380
6	CE	1336
7	DF	2140
8	ES	2033
9	GO	2020
10	MA	747
11	MG	11635

INSIGHTS:



x axis: states
y axis: no of customers

From the above result and chart, it can be seen that most of the customers reside in MG, PR, RS, SC and RJ. This can also explain the variation in month on month orders in the previous question which is witnessed in these states

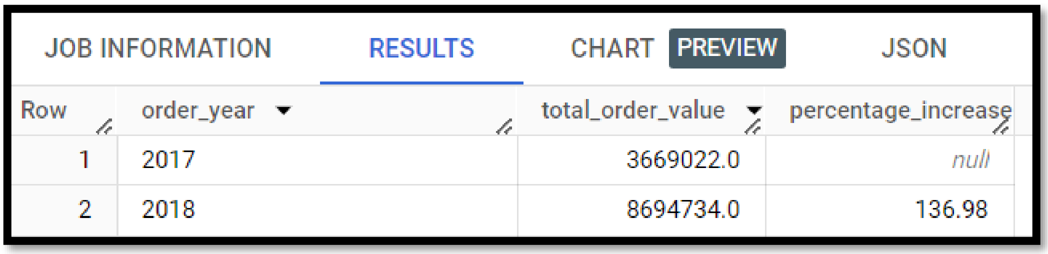
IV. Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

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

Answer: SQL Query

```sql
WITH payment_date AS (
  SELECT EXTRACT (DATE FROM o.order_purchase_timestamp) AS date_of_order,
    EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year_of_order,
    EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month_of_order,
    p.payment_value
  FROM `target_sql.orders` o
  JOIN `target_sql.payments` p
  ON p.order_id = o.order_id
),
order_value AS (
  SELECT
    FORMAT_DATE('%Y', date_of_order) AS order_year,
    ROUND(SUM(payment_value)) AS total_order_value
  FROM payment_date
  WHERE month_of_order BETWEEN 1 AND 8 AND year_of_order BETWEEN 2017 AND 2018
  GROUP BY 1
  ORDER BY 1
),
order_year_value_lag AS (
  SELECT *,
    LAG(total_order_value) OVER (ORDER BY order_year) AS prev_year_sale
  FROM order_value
  ORDER BY order_year
)
SELECT order_year,
  total_order_value,
  ROUND((((total_order_value - prev_year_sale) / prev_year_sale) * 100), 2) AS
percentage_increase
FROM order_year_value_lag
```

Query Output



JOB INFORMATION RESULTS CHART PREVIEW JSON

Row	order_year	total_order_value	percentage_increase
1	2017	3669022.0	*null*
2	2018	8694734.0	136.98

INSIGHTS:

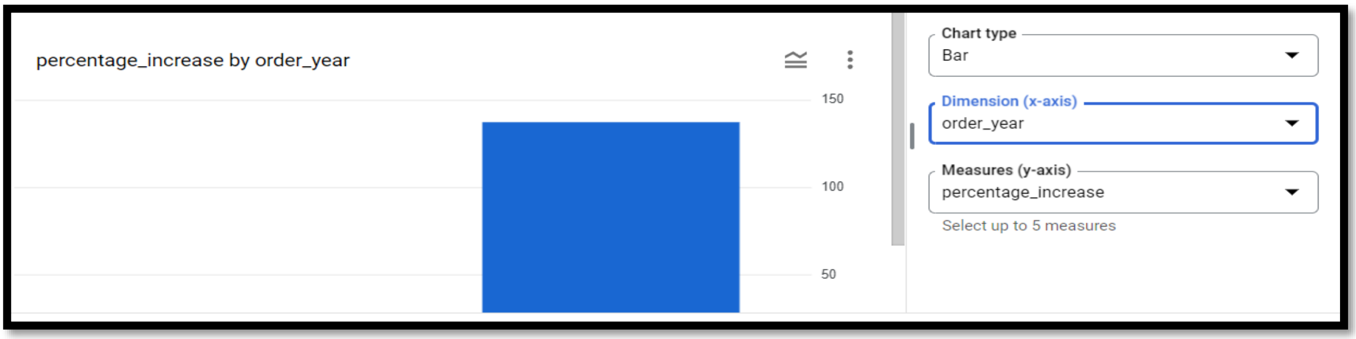


Chart type: Bar

Dimension (x-axis): order_year

Measures (y-axis): percentage_increase

Select up to 5 measures

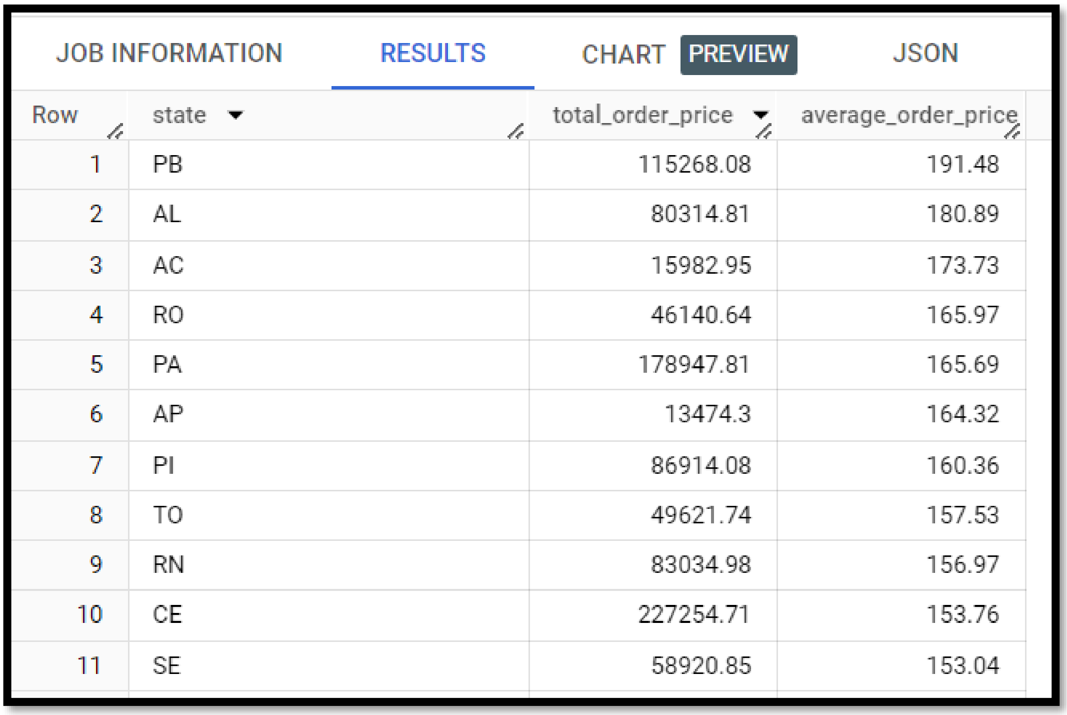
It can be observed that the number of orders placed between January to August in the year 2018 are 136% more than those placed between January to August in 2017. This indicates the increasing trend over the years between the months January to August. There is also a noticeable seasonality in terms of number of orders placed.

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

Answer: SQL Query

```
SELECT c.customer_state AS state,
    SUM(oi.price) AS total_order_price,
    AVG(oi.price) AS average_order_price
FROM `target_sql.order_items` oi
JOIN `target_sql.orders` o ON oi.order_id = o.order_id
JOIN `target_sql.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY average_order_price DESC
```

Query Output


JOB INFORMATION
RESULTS
CHART PREVIEW
JSON

Row	state	total_order_price	average_order_price
1	PB	115268.08	191.48
2	AL	80314.81	180.89
3	AC	15982.95	173.73
4	RO	46140.64	165.97
5	PA	178947.81	165.69
6	AP	13474.3	164.32
7	PI	86914.08	160.36
8	TO	49621.74	157.53
9	RN	83034.98	156.97
10	CE	227254.71	153.76
11	SE	58920.85	153.04

INSIGHTS:

Above is the list of states with total and average order price for each state ordered by average order price in desceding order

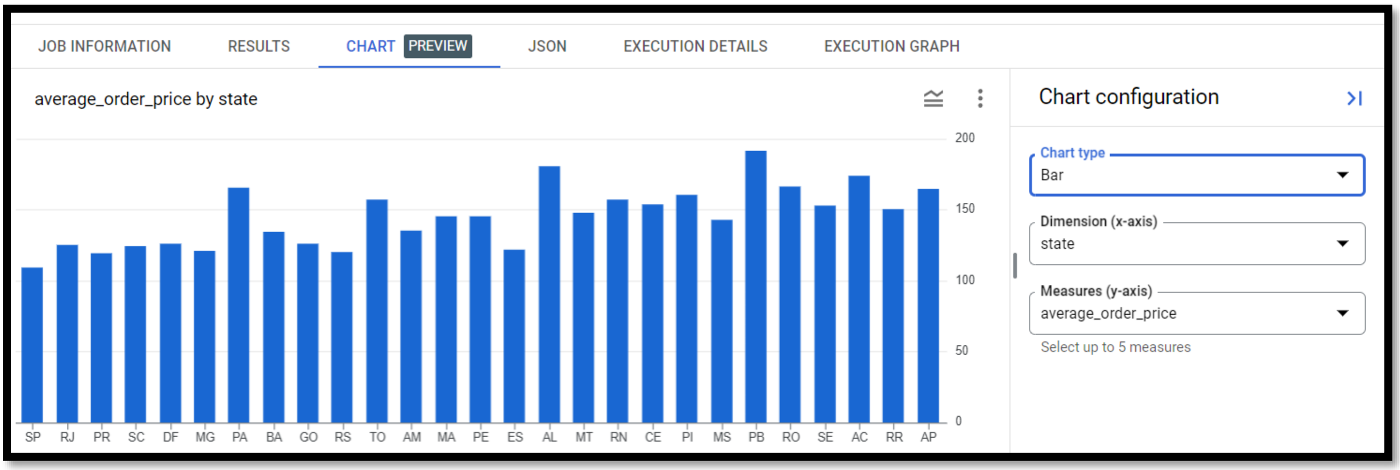


Chart configuration

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): average_order_price

Select up to 5 measures

It is observed that states PB, AL, AC, RO and PA have the highest average order price

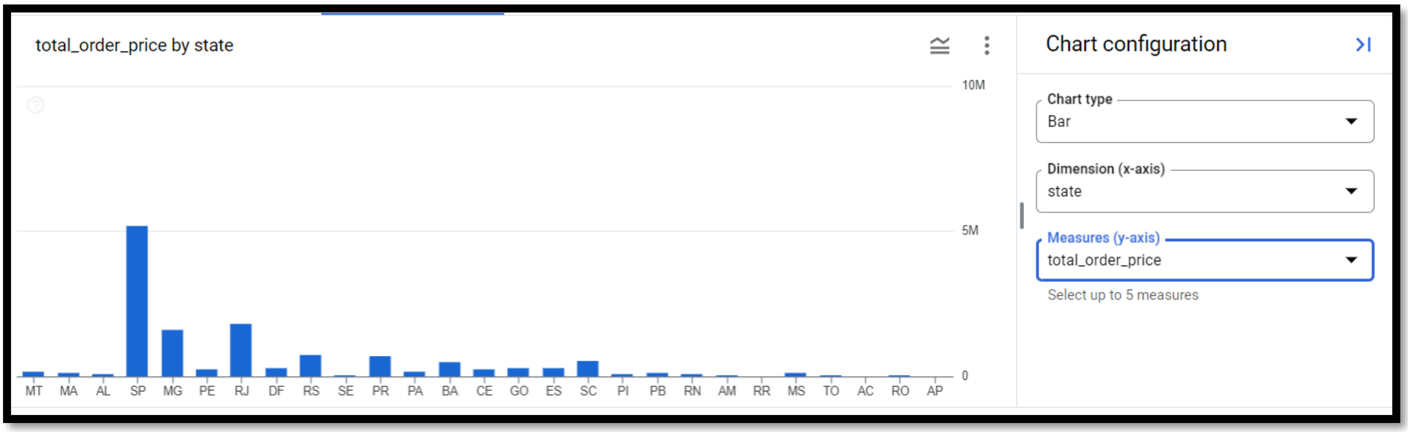


Chart configuration

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): total_order_price

Select up to 5 measures

States SP, RJ, MG, RS, PR are the top 5 states with highest total order price.

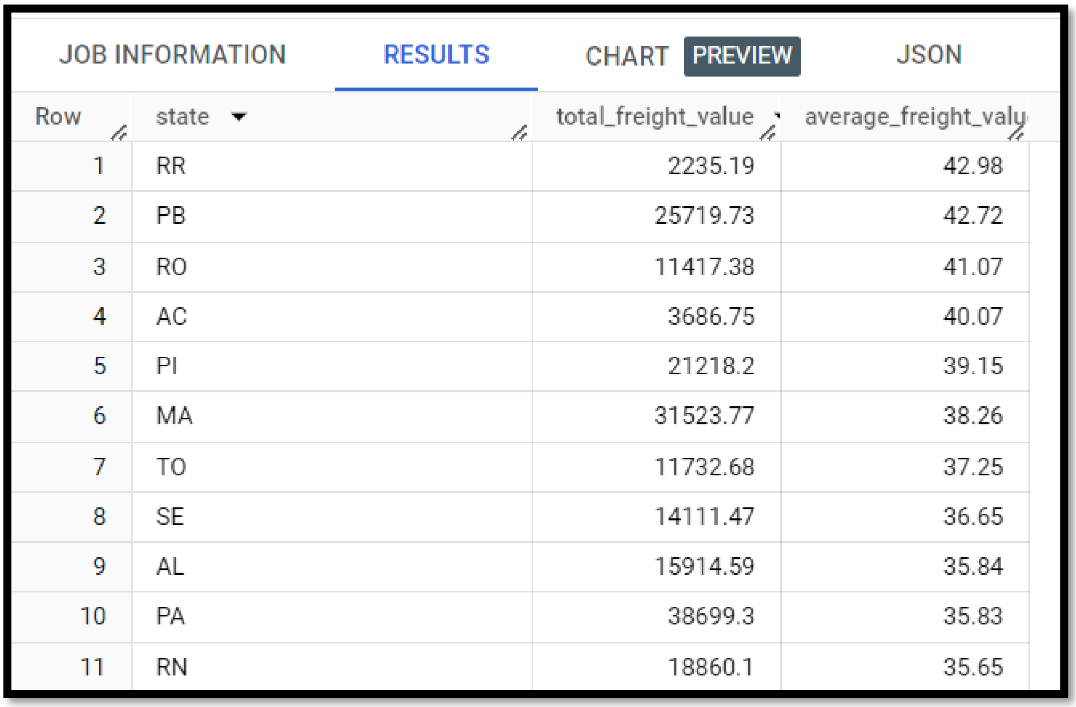
C. Calculate the Total & Average value of order freight for each state.

Answer: SQL Query

```
SELECT c.customer_state AS state,
    ROUND(SUM(oi.freight_value),2) AS total_freight_value,
    ROUND(AVG(oi.freight_value),2) AS average_freight_value
FROM `target_sql.order_items` oi
JOIN `target_sql.orders` o ON oi.order_id = o.order_id
JOIN `target_sql.customers` c ON o.customer_id = c.customer_id
GROUP BY c.customer_state
ORDER BY average_freight_value DESC
```

Query Output

Following is the list of states with total and average freight value ordered by average freight value



Row	state	total_freight_value	average_freight_valu
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
6	MA	31523.77	38.26
7	TO	11732.68	37.25
8	SE	14111.47	36.65
9	AL	15914.59	35.84
10	PA	38699.3	35.83
11	RN	18860.1	35.65

INSIGHTS:

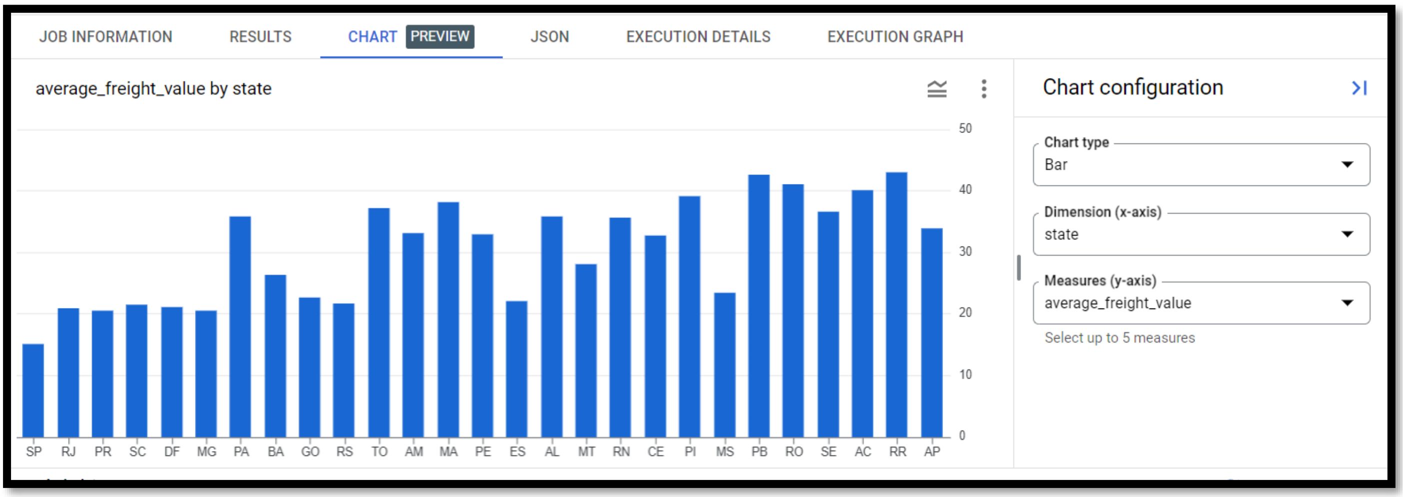


Chart configuration

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): average_freight_value

Select up to 5 measures

It is observed that states RR, PB, RO, AC and PI have highest average freight value.

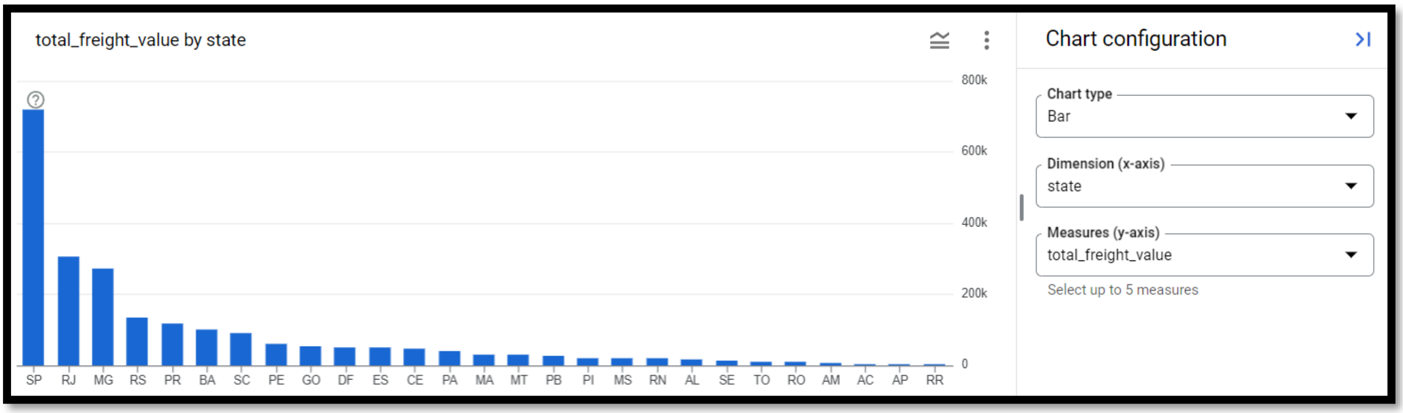


Chart configuration

Chart type: Bar

Dimension (x-axis): state

Measures (y-axis): total_freight_value

Select up to 5 measures

States SP, RJ, MG, RS and PR have highest total freight value.

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

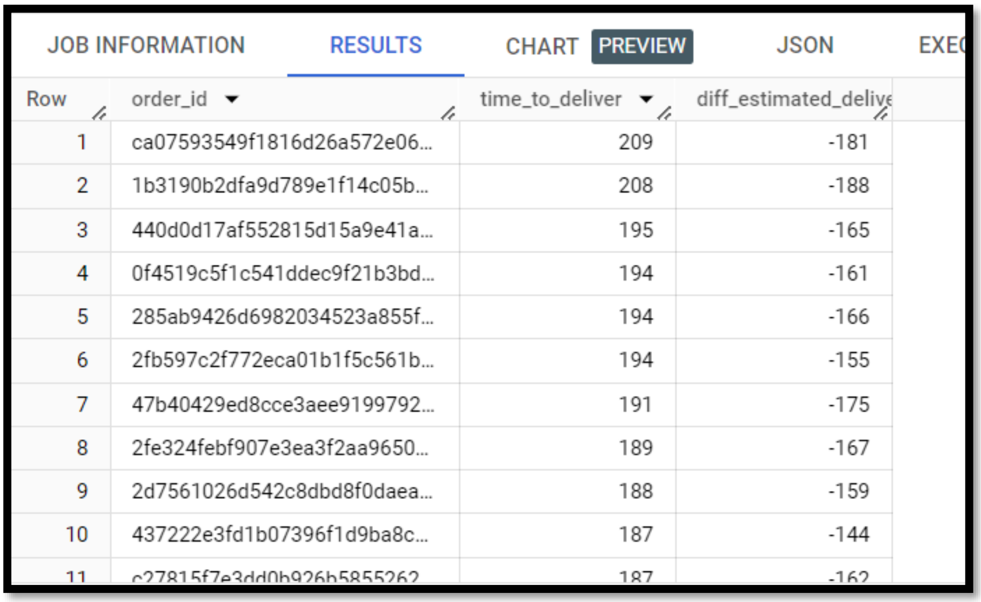
A. Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
Do this in a single query. You can calculate the delivery time and the difference between the estimated & actual d elivery 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**

Answer: SQL Query

```
SELECT order_id,
  IFNULL(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY), NULL) AS delivery_time,
  IFNULL(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,
DAY), NULL) AS diff_estimated_delivery
ORDER BY time_to_deliver DESC
FROM `target_sql.orders`
```

Query Output



Row	order_id	time_to_deliver	diff_estimated_delive
1	ca07593549f1816d26a572e06...	209	-181
2	1b3190b2dfa9d789e1f14c05b...	208	-188
3	440d0d17af552815d15a9e41a...	195	-165
4	0f4519c5f1c541ddec9f21b3bd...	194	-161
5	285ab9426d6982034523a855f...	194	-166
6	2fb597c2f772eca01b1f5c561b...	194	-155
7	47b40429ed8cce3aee9199792...	191	-175
8	2fe324febf907e3ea3f2aa9650...	189	-167
9	2d7561026d542c8dbd8f0daea...	188	-159
10	437222e3fd1b07396f1d9ba8c...	187	-144
11	c27815f7e3dd0b926b5855262...	187	-162

INSIGHTS:

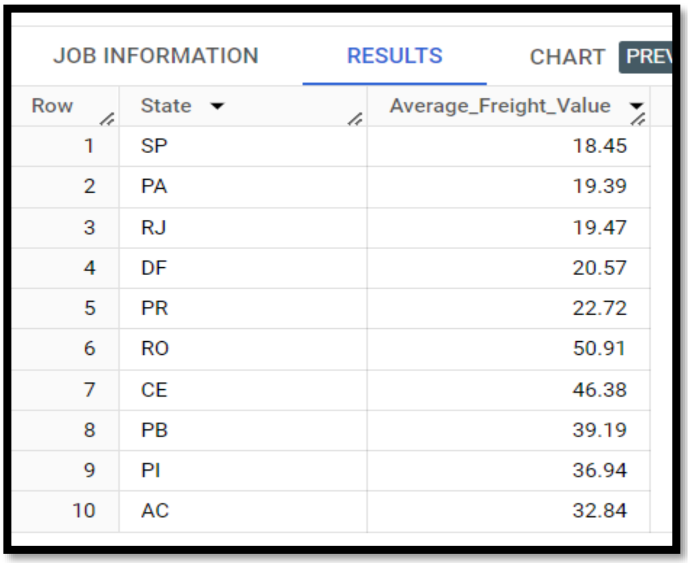
Above is the list of orders with time taken to deliver order and difference between actual and estimated date of order delivery in days. The output has been ordered by time taken to deliver order in descending order

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

Answer: SQL Query

```
WITH average_freight_per_state AS (
  SELECT s.seller_state AS State,
    ROUNd(AVG(oi.freight_value),2) AS Average_Freight_Value
  FROM `target_sql.order_items` oi
  JOIN `target_sql.sellers` s ON oi.seller_id = s.seller_id
  GROUP BY s.seller_state
)
SELECT State,
  Average_Freight_Value
FROM (
  SELECT State,
    Average_Freight_Value,
    ROW_NUMBER() OVER (ORDER BY Average_Freight_Value DESC) AS rank_high
  FROM average_freight_per_state
)
WHERE rank_high <= 5

UNION ALL

SELECT State,
  Average_Freight_Value
FROM (
  SELECT State,
    Average_Freight_Value,
    ROW_NUMBER() OVER (ORDER BY Average_Freight_Value ASC) AS rank_low
  FROM
    average_freight_per_state
)
WHERE rank_low <= 5;
```

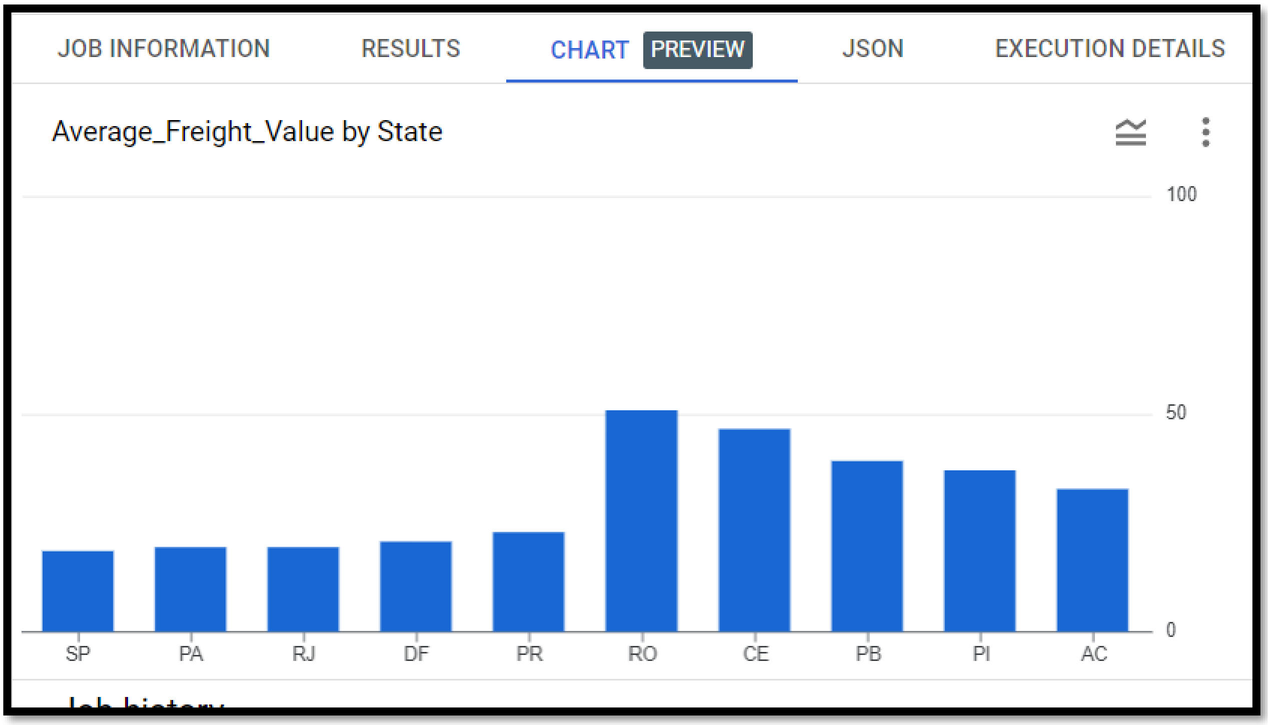
Query Output



Row	State	Average_Freight_Value
1	SP	18.45
2	PA	19.39
3	RJ	19.47
4	DF	20.57
5	PR	22.72
6	RO	50.91
7	CE	46.38
8	PB	39.19
9	PI	36.94
10	AC	32.84

INSIGHTS:

Here are the states with 5 states (top 5 in output) with lowest average freight values in ascending order and 5 states(bottom 5 in output) with highest average freight values in descending order



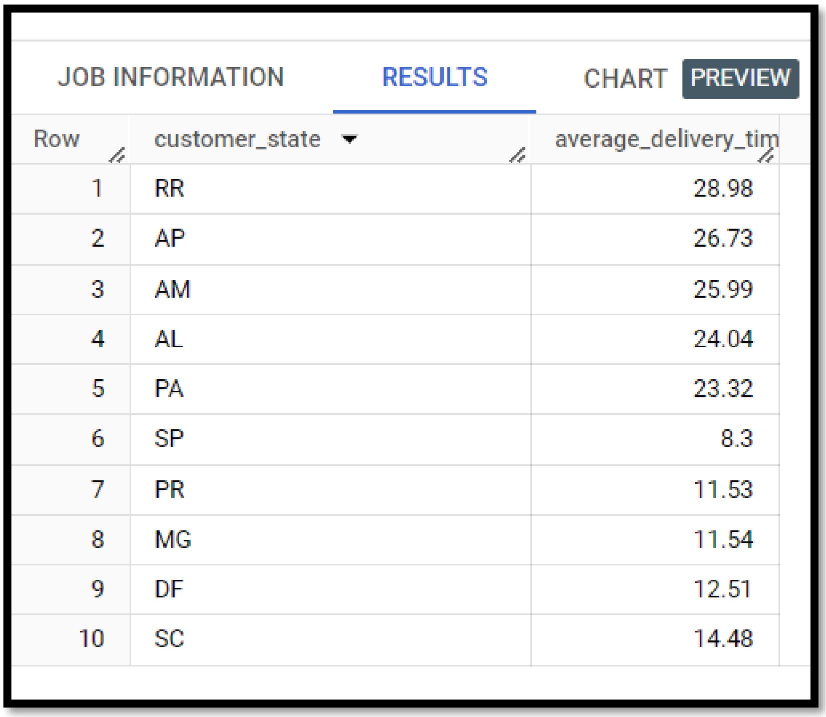
C. Find out the top 5 states with the highest & lowest average delivery time.

Answer: SQL Query

```
WITH DeliveryTimes AS (
  SELECT c.customer_state,
    TIMESTAMP_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp,
DAY) AS delivery_time
  FROM `target_sql.orders` o
  JOIN `target_sql.customers` c ON o.customer_id = c.customer_id
)

SELECT * FROM (
  SELECT customer_state,
    ROUND(AVG(delivery_time),2) AS average_delivery_time
  FROM DeliveryTimes
  GROUP BY customer_state
  ORDER BY average_delivery_time DESC
  LIMIT 5
) top_5_states

UNION ALL

SELECT * FROM (
  SELECT customer_state,
    ROUND(AVG(delivery_time),2) AS average_delivery_time
  FROM DeliveryTimes
  GROUP BY customer_state
  ORDER BY average_delivery_time ASC
```

```
    LIMIT 5
  ) bottom_5_states
```

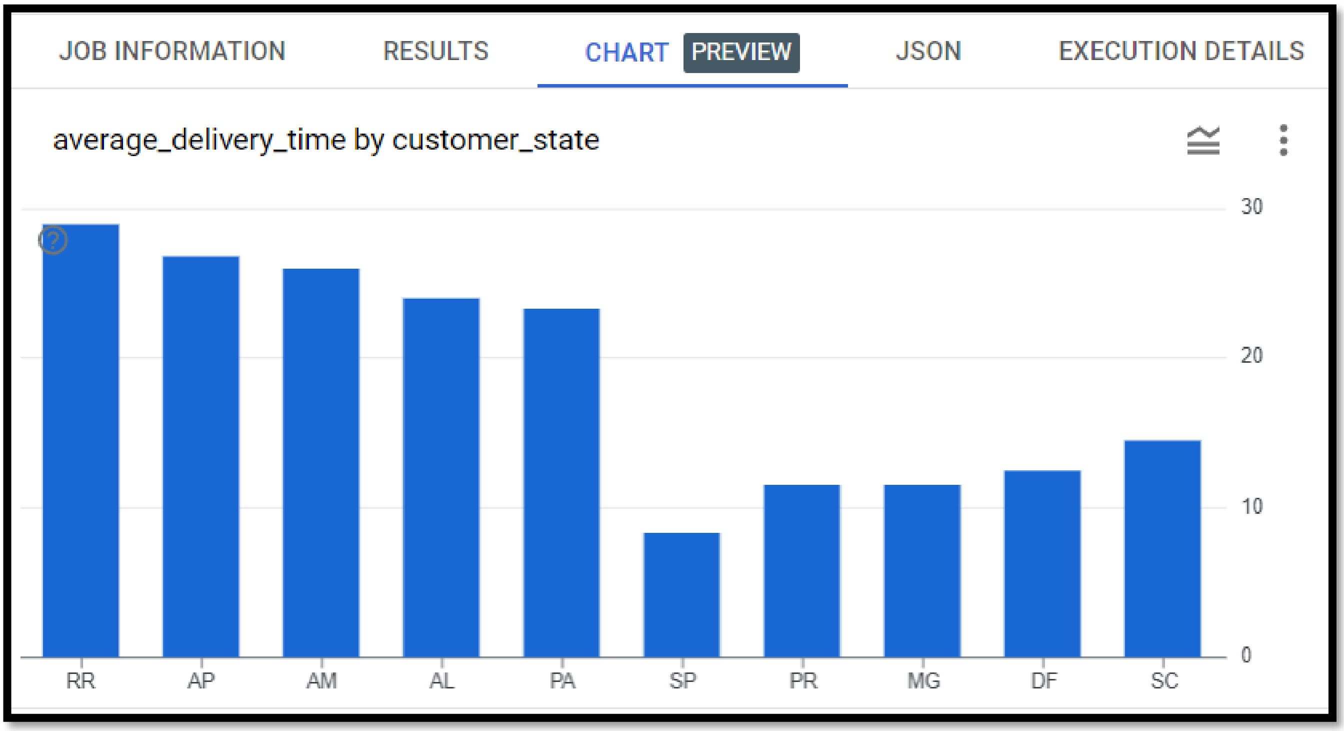
Query Output



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
6	SP	8.3
7	PR	11.53
8	MG	11.54
9	DF	12.51
10	SC	14.48

INSIGHTS:

Here are the states with 5 states(top 5 in output) with highest average delivery time in descending order and 5 states(bottom 5 in output) with lowest average delivery time in ascending order

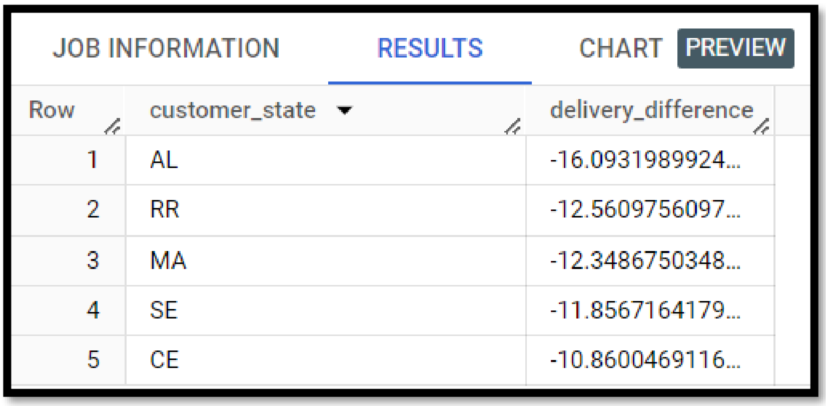


D. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery. You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

Answer: SQL Query

```
WITH OrderDeliveryTimes AS (
  SELECT c.customer_state,
    AVG(TIMESTAMP_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)) AS avg_days_purchase_to_delivery,
    AVG(TIMESTAMP_DIFF(order_estimated_delivery_date,
order_delivered_customer_date, DAY)) AS avg_days_delivery_to_estimated
  FROM `target_sql.orders` o
  JOIN `target_sql.customers` c
  ON o.customer_id = c.customer_id
  WHERE o.order_status = 'delivered'
  GROUP BY c.customer_state
)

SELECT customer_state,
  (avg_days_delivery_to_estimated - avg_days_purchase_to_delivery) AS
delivery_difference
FROM OrderDeliveryTimes
ORDER BY delivery_difference ASC
LIMIT 5;
```

Query Output



Row	customer_state	delivery_difference
1	AL	-16.0931989924...
2	RR	-12.5609756097...
3	MA	-12.3486750348...
4	SE	-11.8567164179...
5	CE	-10.8600469116...

INSIGHTS:

States AL, RR, MA, SE and CE are the top states where delivery is fast when compared with the estimated date of delivery

VI. Analysis based on the payments:

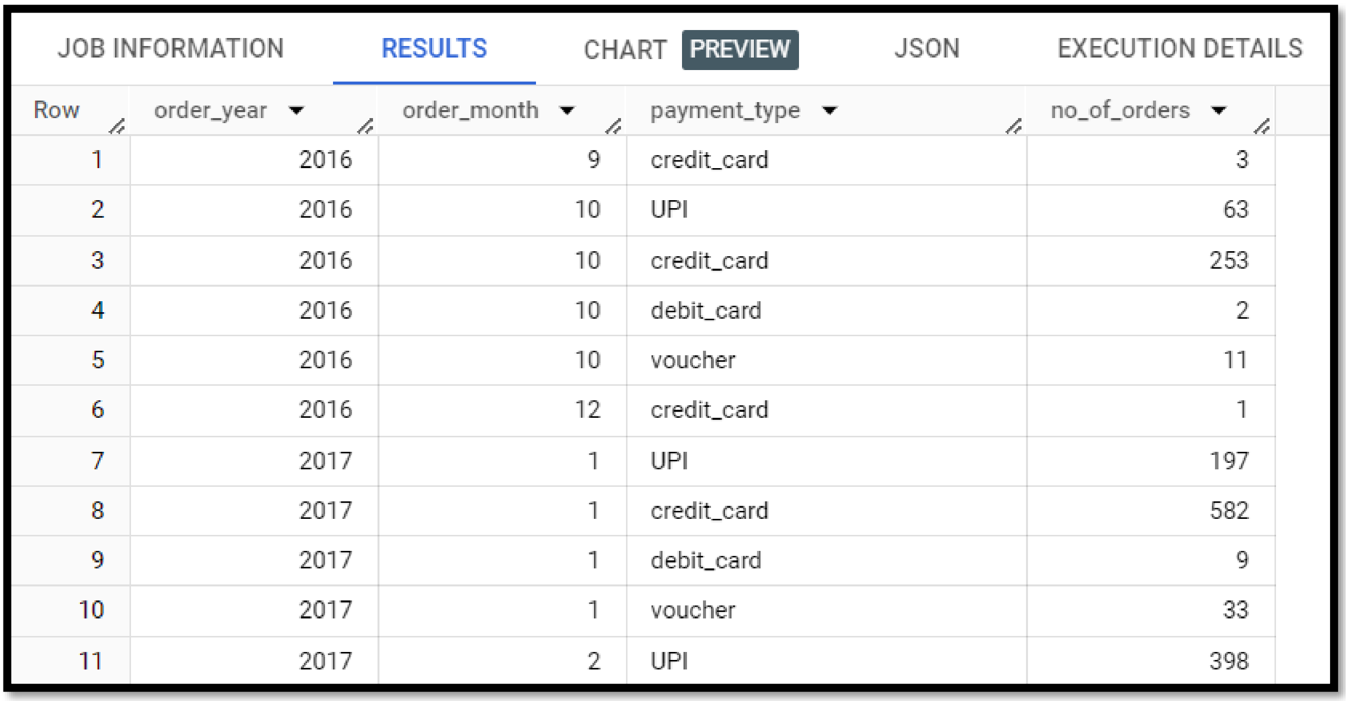
A. Find the month on month no. of orders placed using different payment types.

Answer: SQL Query

```
SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) AS order_year,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS num_orders
FROM `target_sql.payments` p
```

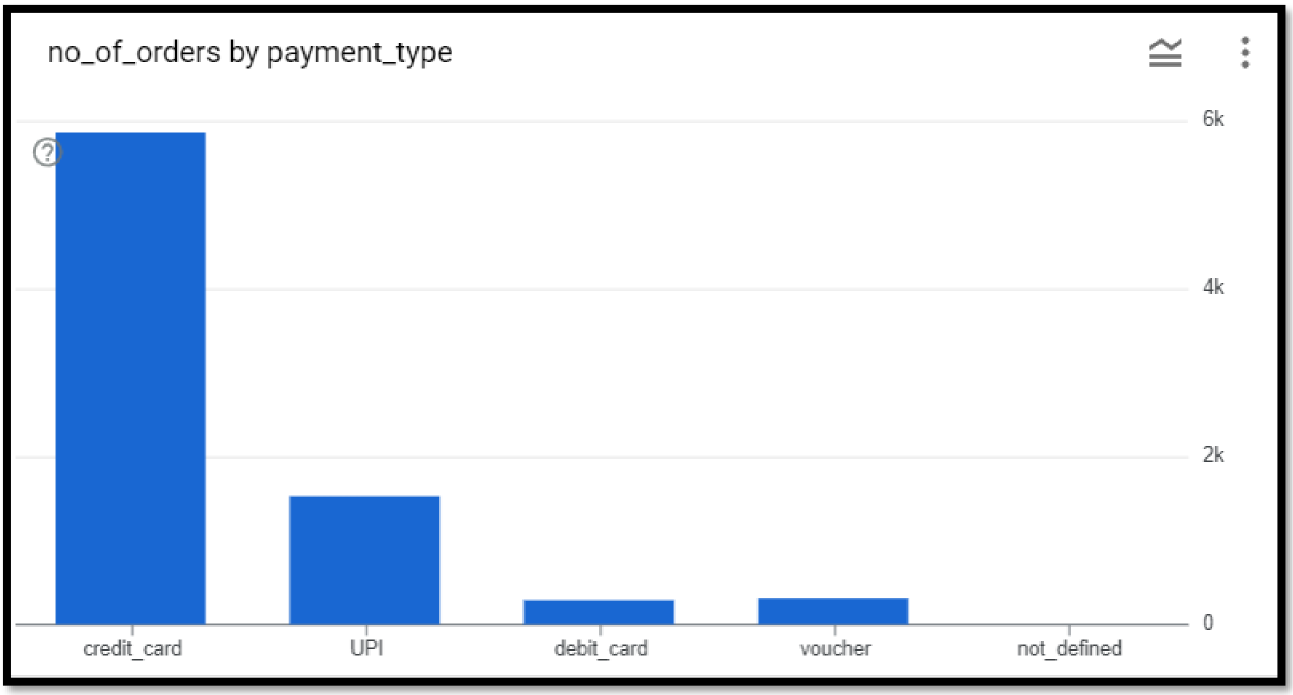
```
JOIN `target_sql.orders` o
ON p.order_id = o.order_id
GROUP BY order_year, order_month, p.payment_type
ORDER BY order_year, order_month, p.payment_type
```

Query Output



Row	order_year	order_month	payment_type	no_of_orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	253
4	2016	10	debit_card	2
5	2016	10	voucher	11
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	582
9	2017	1	debit_card	9
10	2017	1	voucher	33
11	2017	2	UPI	398

INSIGHTS:



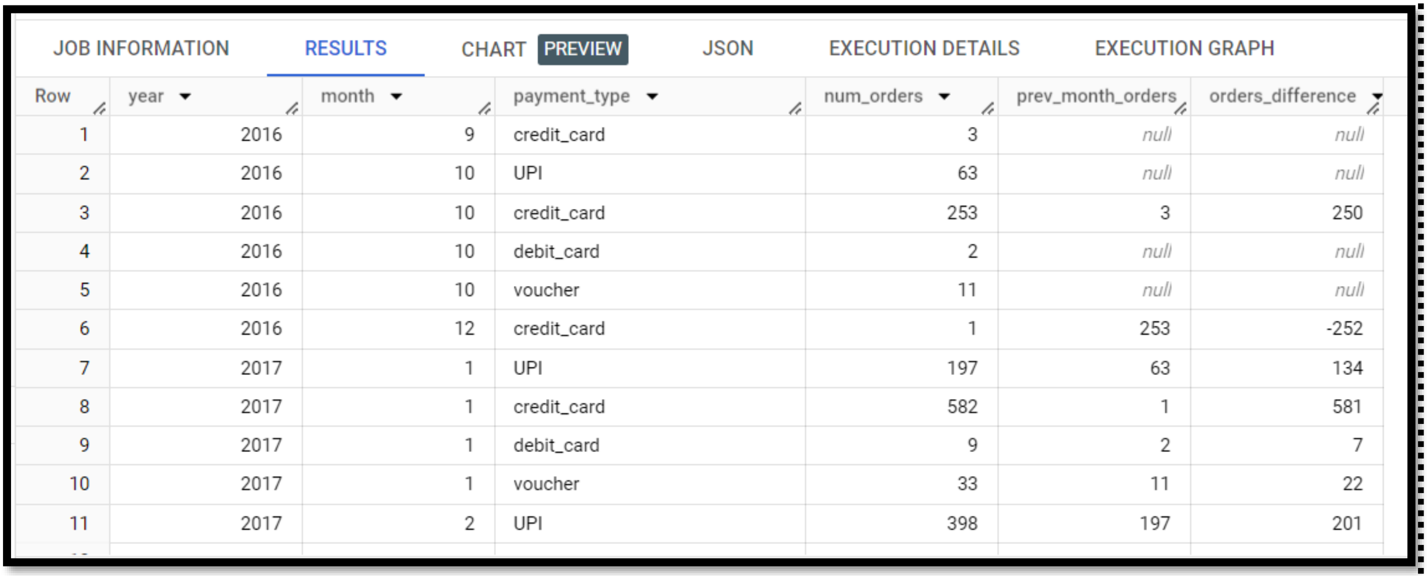
From the above result it can be observed that most orders have been placed using credit card, followed by UPI and voucher. Least no of orders purchased through debit card

2nd Approach(Question 6A)

Answer: SQL Query

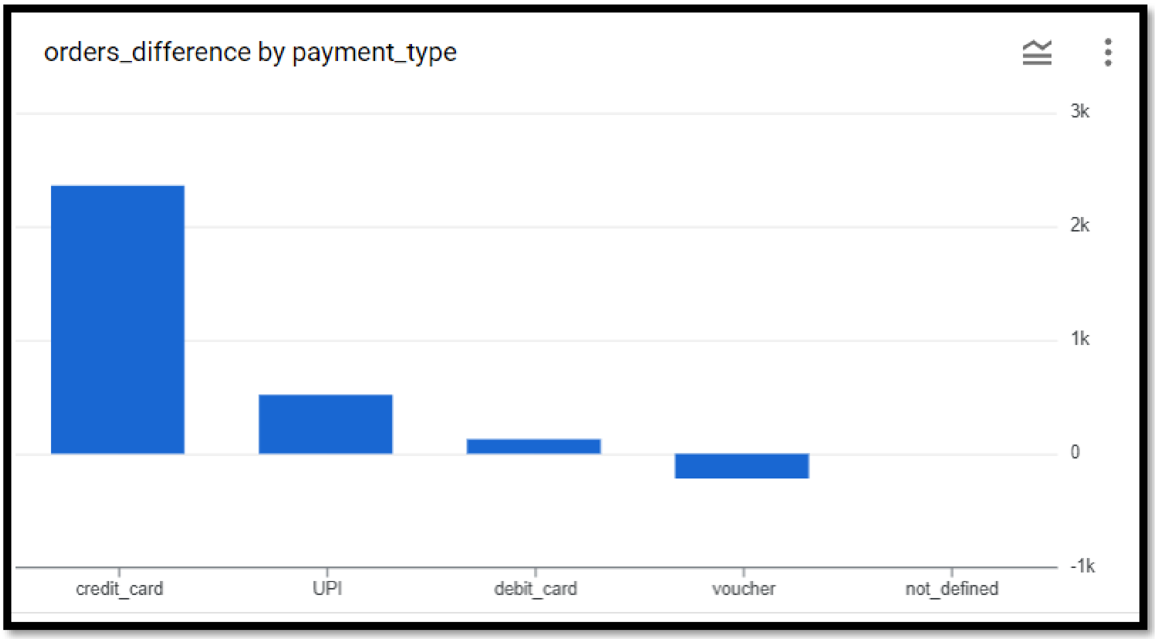
```
WITH MonthlyOrders AS (
  SELECT EXTRACT(YEAR FROM TIMESTAMP(order_purchase_timestamp)) AS year,
    EXTRACT(MONTH FROM TIMESTAMP(order_purchase_timestamp)) AS month,
    payment_type,
    COUNT(DISTINCT o.order_id) AS num_orders
  FROM `target_sql.orders` o
  JOIN `target_sql.payments` p ON o.order_id = p.order_id
  GROUP BY year, month,payment_type
)

SELECT year,
  month,
  payment_type,
  num_orders,
  LAG(num_orders) OVER (PARTITION BY payment_type ORDER BY year, month) AS
prev_month_orders,
  num_orders - LAG(num_orders) OVER (PARTITION BY payment_type ORDER BY year,
month) AS orders_difference
FROM MonthlyOrders
ORDER BY year, month, payment_type
```

Query Output



Row	year	month	payment_type	num_orders	prev_month_orders	orders_difference
1	2016	9	credit_card	3	null	null
2	2016	10	UPI	63	null	null
3	2016	10	credit_card	253	3	250
4	2016	10	debit_card	2	null	null
5	2016	10	voucher	11	null	null
6	2016	12	credit_card	1	253	-252
7	2017	1	UPI	197	63	134
8	2017	1	credit_card	582	1	581
9	2017	1	debit_card	9	2	7
10	2017	1	voucher	33	11	22
11	2017	2	UPI	398	197	201

INSIGHTS:



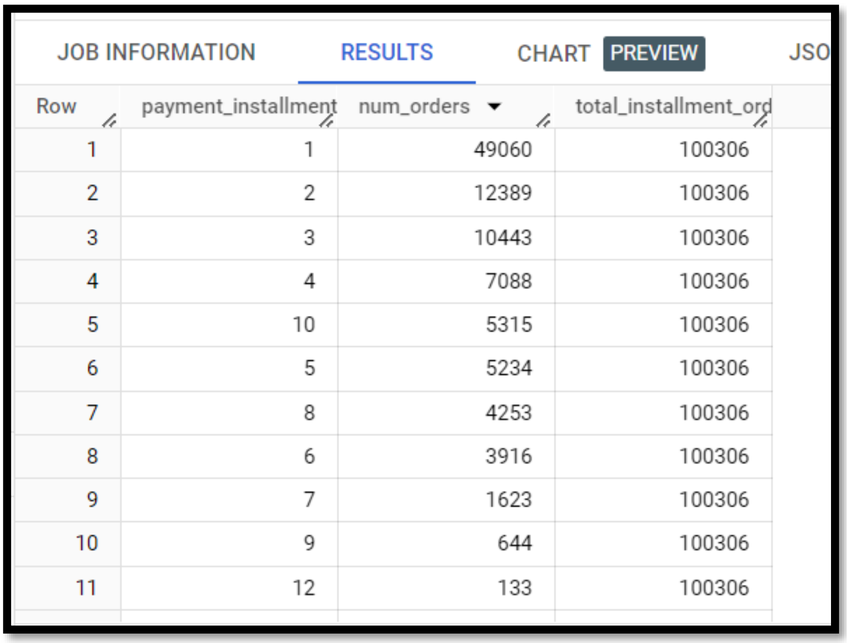
From the above chart we can see that there is a lot of variation in the transcations made using credit cards month on month which may be due to various credit card offers and the payment through vouchers has decreased.

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

Answer: SQL Query

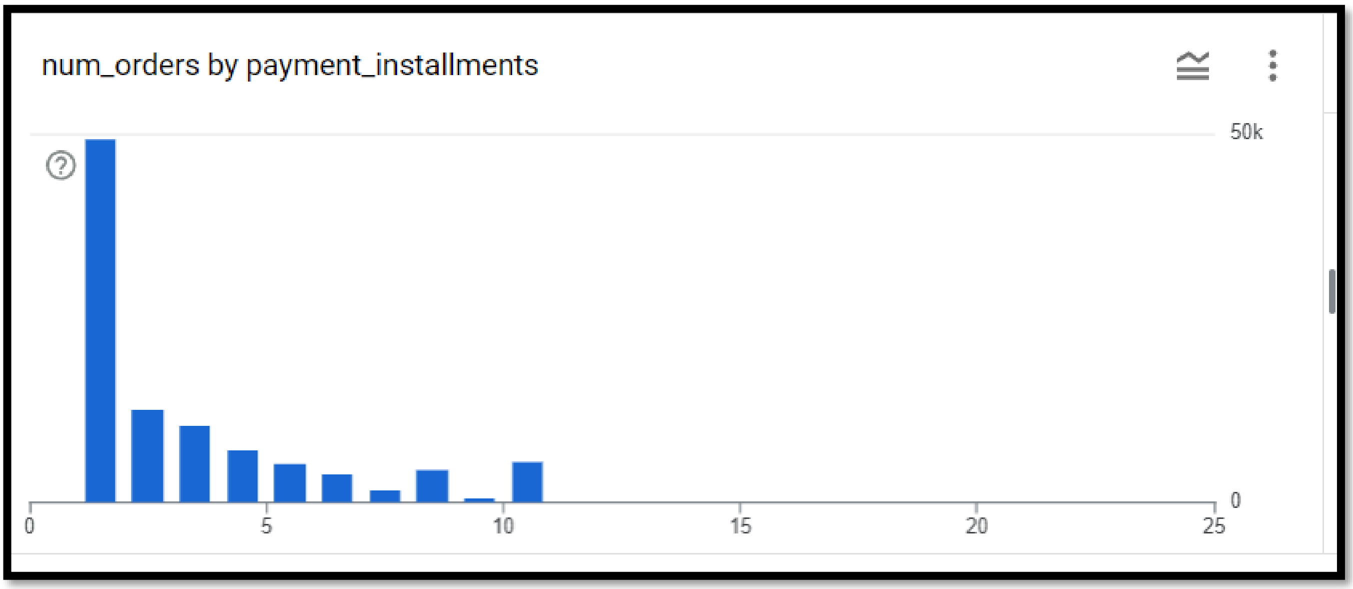
```
SELECT *,SUM(num_orders) OVER() AS total_installment_orders
FROM(
SELECT payment_installments, COUNT(DISTINCT order_id) AS num_orders
FROM `target_sql.payments`
GROUP BY payment_installments
HAVING payment_installments >= 1)
ORDER BY num_orders DESC
```

Query Output



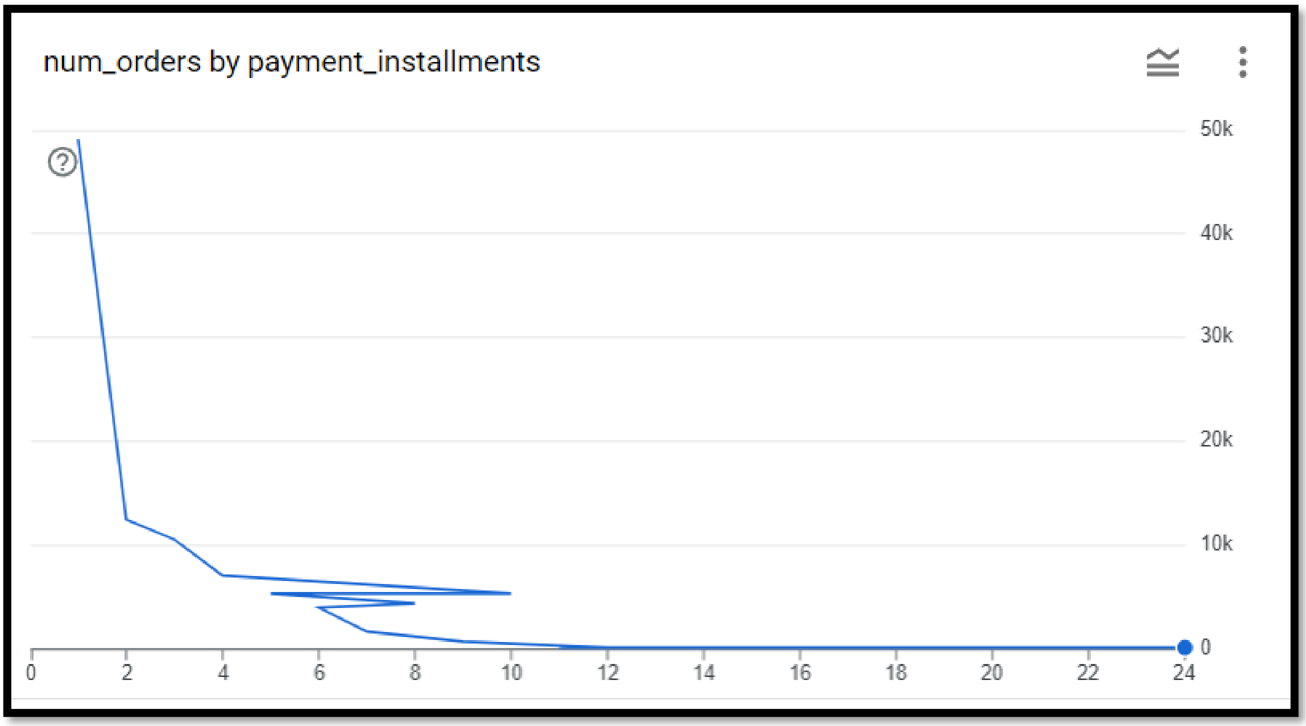
Row	payment_installment	num_orders	total_installment_ord
1	1	49060	100306
2	2	12389	100306
3	3	10443	100306
4	4	7088	100306
5	10	5315	100306
6	5	5234	100306
7	8	4253	100306
8	6	3916	100306
9	7	1623	100306
10	9	644	100306
11	12	133	100306

INSIGHTS:



X axis: No of instalments
y axis: No of orders

From the above chart and output, it can be seen that maximum orders (49060 orders) are paid in 1 installment only. Most of the payments are made in 1,2,3, and 4 installments which can also be observed from the below graph.



Thank you

****