Business Case: Target SQ

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

1. Data type of all columns in the "customers" table.

Ans: -

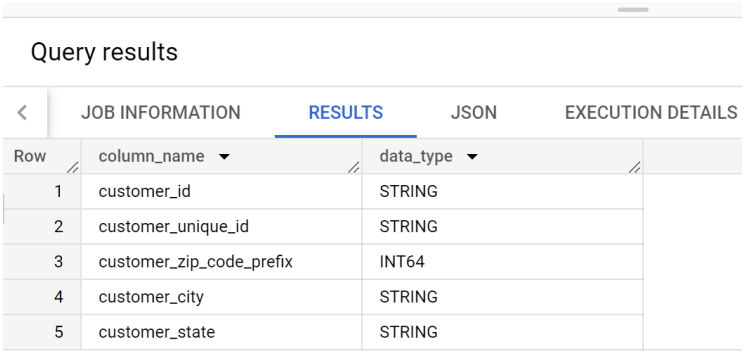
```
SELECT

column_name,
data_type
FROM case_study.INFORMATION_SCHEMA.COLUMNS
where
table_name = 'customers';
```

Insights: -

1. Customer table consist of 5 columns and data types.
2. The column names and their data types are:

column_name	data_type
customer_id	STRING
customer_unique_id	STRING
customer_zip_code_prefix	INT64
customer_city	STRING
customer_state	STRING

Output: -



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

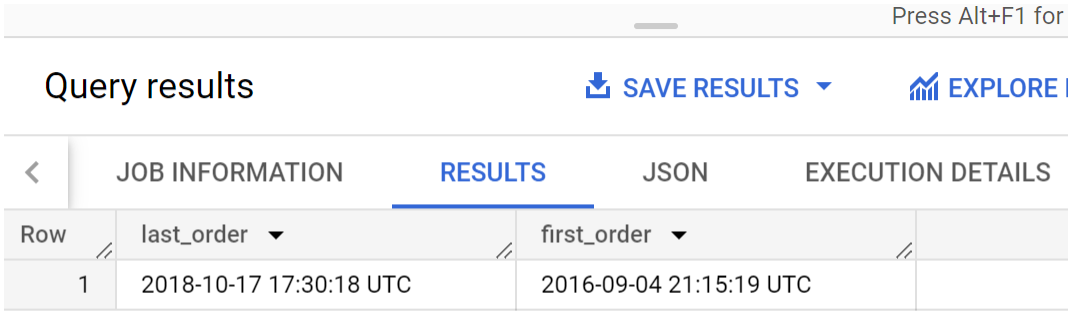
Ans: -

```
SELECT

max(order_purchase_timestamp) as last_order,
min(order_purchase_timestamp) as first_order
FROM case_study.orders;
```

Insights: -

1. The time range is displayed as “last date” and “first date”.
2. The data type used in displaying time is “timestamp”.

Output: -



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

Ans: -

```
select
count(distinct c.customer_state) as no_of_states,
count(distinct c.customer_city) as no_of_cities
from
case_study.customers c
join case_study.orders o on c.customer_id = o.customer_id
where
o.order_purchase_timestamp between '2016-09-04' and '2018-10-17'
;
```

Insights: -

1. The given time period in which orders were made is taken as “2016-09-04” to “2018-10-17”.
2. The no of cities in the given the given time period is 4119.
3. The no of state in the given time period is 27.

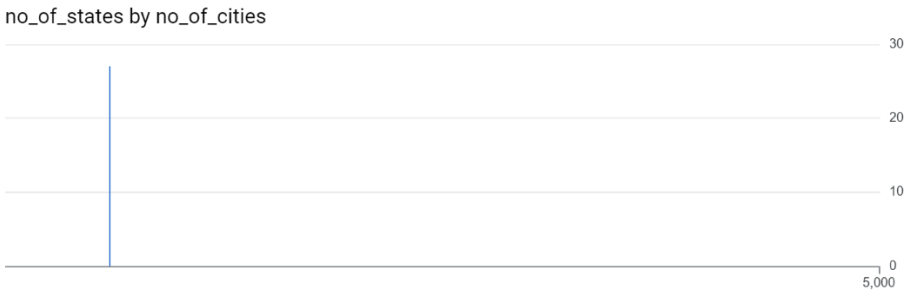
Output: -

Query results

JOB INFORMATION | RESULTS | JSON

Row	no_of_states	no_of_cities
1	27	4119

Chart: -



2.In-depth Exploration:

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

Ans: -

```
WITH growthTrend AS (
    SELECT
        extract(year from order_purchase_timestamp) AS year,
        COUNT(*) AS orderCount
    FROM
        case_study.orders
    GROUP BY
        1
)
SELECT
    year,
    orderCount,
    LAG(orderCount) OVER (ORDER BY year) AS last_year_order,
    round(((orderCount - LAG(orderCount) OVER (ORDER BY year)) /
LAG(orderCount) OVER (ORDER BY year)) * 100,2) AS growth_percent
FROM
    growthTrend
    order by 1;
```

Insights: -

1. The orders are made in year 2016, 2017 & 2018.
2. There is a drastic increase in number of orders placed from 2016 to 2017.

3. From 2017 there is a slight increase in number of orders.

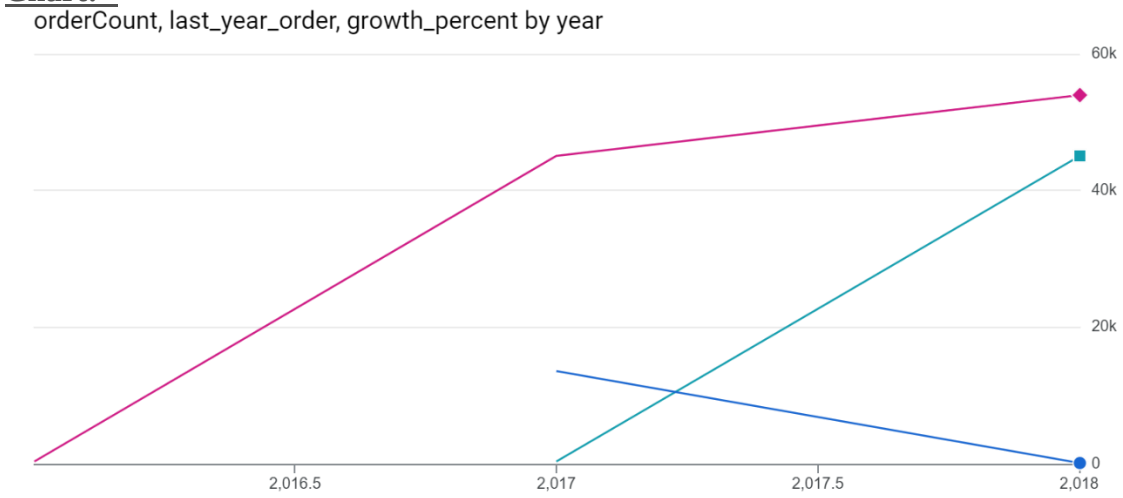
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHART

Row	year	orderCount	last_year_order	growth_percent
1	2016	329	null	null
2	2017	45101	329	13608.51
3	2018	54011	45101	19.76

Chart: -



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

Ans:-

```
WITH growthTrend AS (
    SELECT
        extract(month from order_purchase_timestamp) AS month,
        COUNT(*) AS orderCount
    FROM
        case_study.orders
    GROUP BY
        1
```

```
)
SELECT
    month,
    orderCount,
    LAG(orderCount) OVER (ORDER BY month) AS last_month_order,
    round(((orderCount - LAG(orderCount) OVER (ORDER BY month)) /
LAG(orderCount) OVER (ORDER BY month)) * 100,2) AS growth_percent
FROM
    growthTrend
    order by 1;
```

Insights:

1. There is a decrease in no of orders placed from
    - March-April
    - May-June
    - August- September
    - November-December
2. Apart from above there is a continuous increase in no of orders placed.

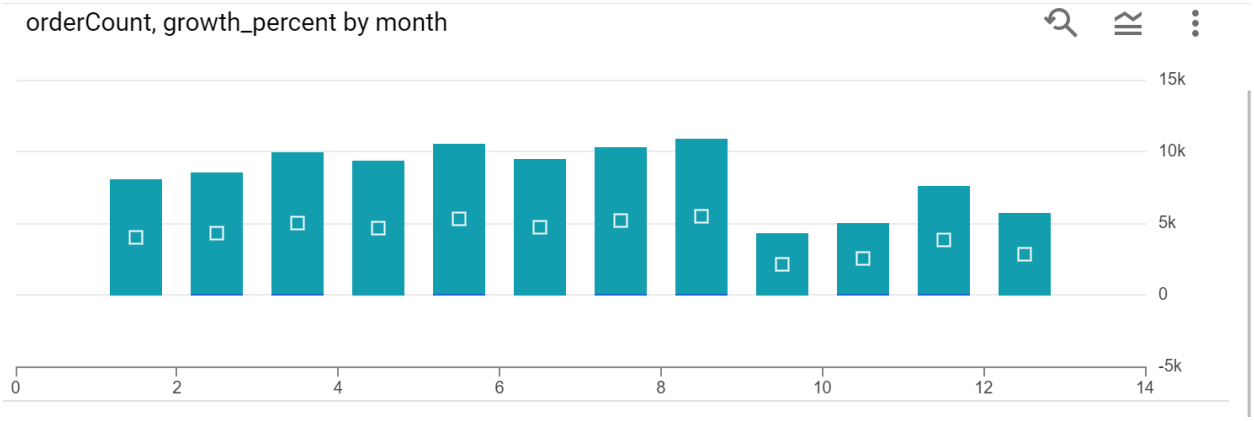
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHA

Row	month	orderCount	last_month_order	growth_percent
1	1	8069	null	null
2	2	8508	8069	5.44
3	3	9893	8508	16.28
4	4	9343	9893	-5.56
5	5	10573	9343	13.16
6	6	9412	10573	-10.98
7	7	10318	9412	9.63
8	8	10843	10318	5.09
9	9	4305	10843	-60.3
10	10	4959	4305	15.19
11	11	7544	4959	52.13
12	12	5674	7544	-24.79

Chart:-



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

Ans: -

```
SELECT
case
   when times>=0 and times<=6 then 'Dawn'
   when times>=7 and times<=12 then 'Morning'
   when times>=13 and times<=18 then 'Afternoon'
   else
   'Night'
   end as Time_of_day,

count(*) as order_count
FROM
(select
extract(hour from order_purchase_timestamp) as times
from case_study.orders) t
group by 1;
```

Insights: -

1. Brazilians place their orders mostly in “Afternoon”.
2. Least number of orders are placed during “Dawn” Time of the Day

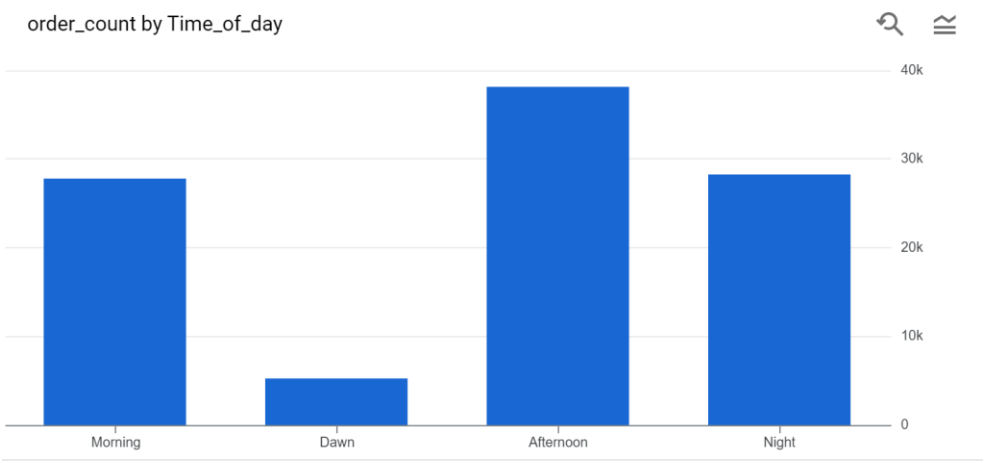
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Time_of_day	order_count
1	Morning	27733
2	Dawn	5242
3	Afternoon	38135
4	Night	28331

Charts: -



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

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

Ans: -

```
SELECT
distinct extract(month from o.order_purchase_timestamp) as date,
count(*) as order_count,
c.customer_state
FROM case_study.orders o
inner join case_study.customers c on o.customer_id = c.customer_id
group by 1,3
order by 1
;
```

Insights: -

1. In month of January, the maximum orders were placed in state of “SP”.
2. Only 2 orders were placed in state “RR” in January.

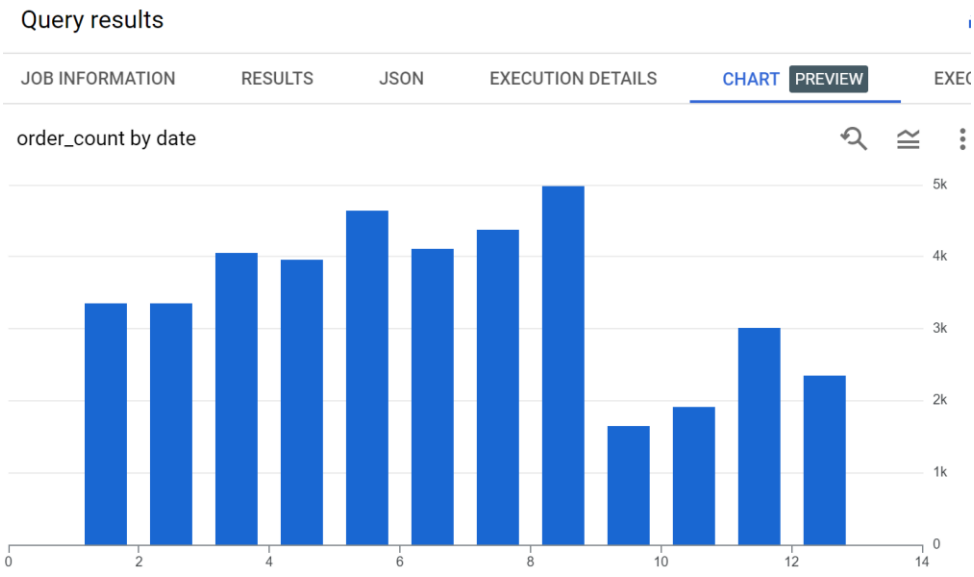
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	date	order_count	customer_state
1	1	990	RJ
2	1	3351	SP
3	1	151	DF
4	1	427	RS
5	1	99	CE
6	1	113	PE
7	1	443	PR
8	1	264	BA
9	1	971	MG
10	1	51	RN
11	1	82	PA
12	1	66	MA

Chart: -



2 How are the customers distributed across all the states?

Ans: -

```
select
customer_state,
count(*)
from case_study.customers
group by 1
```

```
order by 1;
```

Insights: -

1. Least number of customers are in state “RR”.
2. Maximum number of customers are in state “SP”.

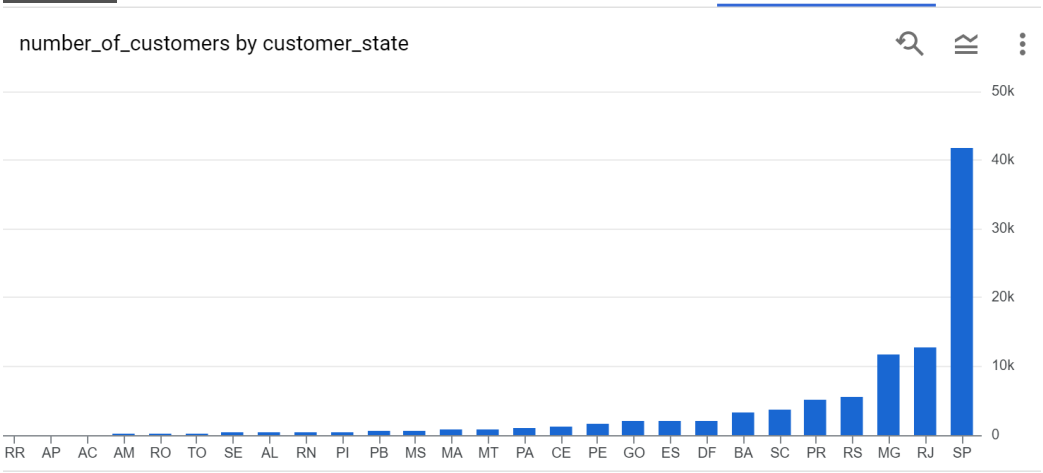
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECU

Row	customer_state	number_of_custome
1	RR	46
2	AP	68
3	AC	81
4	AM	148
5	RO	253
6	TO	280
7	SE	350
8	AL	413
9	RN	485
10	PI	495
11	PB	536

Chart: -



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

1. 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.

Ans: -

```
with yearlyCostIncrease as
  ( select
    extract(year from o.order_purchase_timestamp) as year,
    round(sum(p.payment_value), 2) as cost_of_orders
    from case_study.orders o
    join case_study.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 1
    )

  select
  year,
  cost_of_orders,
  LAG(cost_of_orders) OVER (ORDER BY year) AS last_year_order,
  round(((cost_of_orders - LAG(cost_of_orders) OVER (ORDER BY year)) /
LAG(cost_of_orders) OVER (ORDER BY year)) * 100,2) AS percent_increase
  from
  yearlyCostIncrease
  order by 1
;
```

Insights: -

1. The sum of cost of all the orders are taken.
2. There is a percent increase of 136.98 from 2017 to 2018 in cost of orders.
3. Data of January to August are taken.

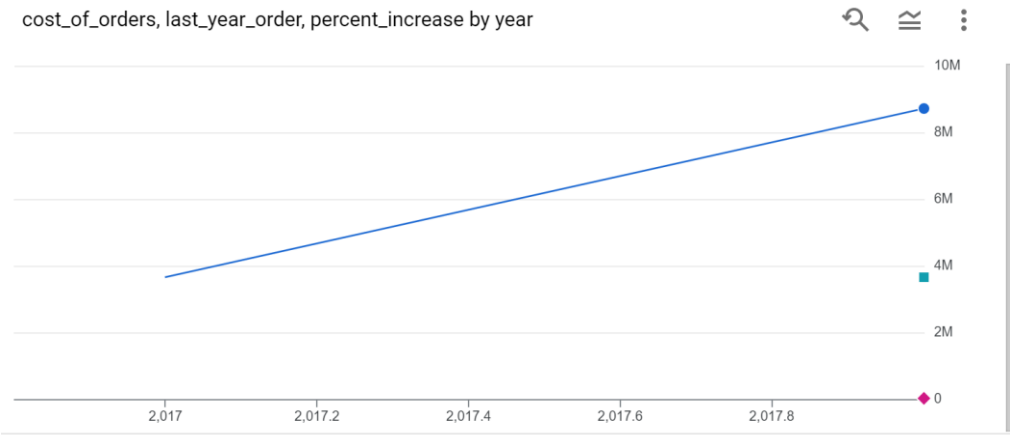
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | CHAI

Row	year	cost_of_orders	last_year_order	percent_increase
1	2017	3669022.12	null	null
2	2018	8694733.84	3669022.12	136.98

Chart: -



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

Ans: -

```
select
c.customer_state,
round(sum(payment_value), 2) as total_order_price,
round(avg(payment_value), 2) as avg_order_price
from case_study.customers c
join case_study.orders o on c.customer_id = o.customer_id
join case_study.payments p on o.order_id = p.order_id
group by 1
order by 2;
```

Insights: -

1. The highest total and average value of order price is in state “SP”.
2. The lowest total and average value of order price is in state “RR”.

Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_order_price	avg_order_price
1	RR	10064.62	218.8
2	AP	16262.8	232.33
3	AC	19680.62	234.29
4	AM	27966.93	181.6
5	RO	60866.2	233.2
6	TO	61485.33	204.27
7	SE	75246.25	208.44
8	AL	96962.06	227.08
9	RN	102718.13	196.78
10	PI	108523.97	207.11

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

Ans: -

```
select
c.customer_state,
round(sum(freight_value), 2) as total_freight_order,
round(avg(freight_value), 2) as avg_freight_order
from case_study.customers c
join case_study.orders o on c.customer_id = o.customer_id
join case_study.order_items ot on o.order_id = ot.order_id
group by 1
order by 2;
```

Insights: -

1. The highest total and average value of freight order is in state “SP”.
2. The lowest total and average value of freight order is in state “RR”.

Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	customer_state	total_freight_order	avg_freight_order
1	RR	2235.19	42.98
2	AP	2788.5	34.01
3	AC	3686.75	40.07
4	AM	5478.89	33.21
5	RO	11417.38	41.07
6	TO	11732.68	37.25
7	SE	14111.47	36.65
8	AL	15914.59	35.84
9	RN	18860.1	35.65
10	MS	19144.03	23.37

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

1. 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 delivery 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

Ans: -

```
select
DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, DAY) as
delivery_time,
DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) as
diff_estimated_delivery
from case_study.orders;
```

Insights: -

1. Most of the orders got delivered before the estimated delivery time.

Output: -

Query results

JOB INFORMATION | RESULTS | JSON

Row	delivery_time	diff_estimated_deliv
1	30	-12
2	30	28
3	35	16
4	30	1
5	32	0
6	29	1
7	43	-4
8	40	-4
9	37	-1
10	33	-5

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

Ans: -

```
(select
    c.customer_state,
    round(avg(ot.freight_value),2) as highest_freight,

from case_study.order_items ot
inner join case_study.orders o on ot.order_id = o.order_id
inner join case_study.customers c on o.customer_id = c.customer_id
group by 1
```

```sql
order by 2 desc
limit 5)
UNION ALL
(select
    c.customer_state,
    round(avg(ot.freight_value),2) as low_freight
from case_study.order_items ot
inner join case_study.orders o on ot.order_id = o.order_id
inner join case_study.customers c on o.customer_id = c.customer_id
group by 1
order by 2
limit 5);
```

Insight: -

1. The first 5 output are highest 5 average freight values.
2. The last 5 outputs are lowest 5 average freight values.
3. Average freight value is highest in state “RR”.
4. Average freight value is lowest in state “SP”.

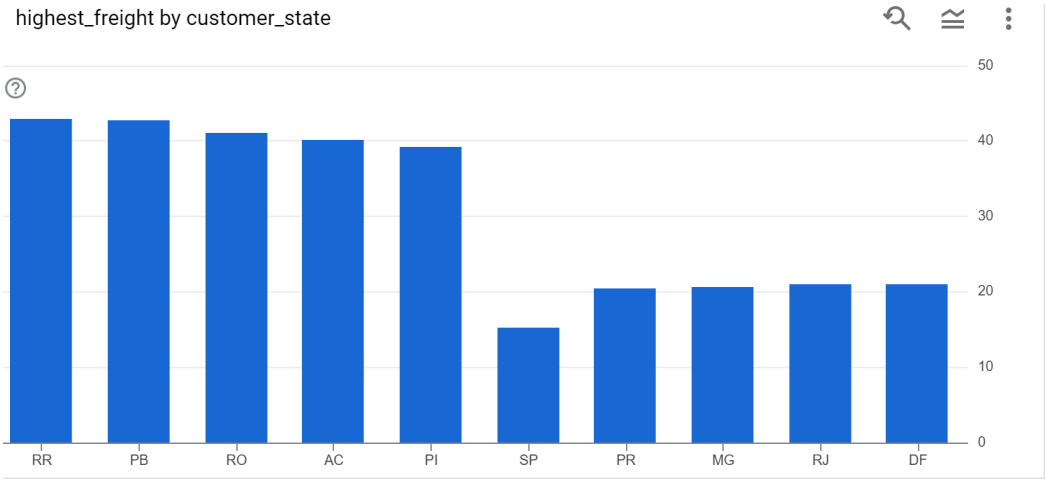
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTIC

Row	customer_state	highest_freight
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15
6	SP	15.15
7	PR	20.53
8	MG	20.63
9	RJ	20.96
10	DF	21.04

Chart: -



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

Ans: -

```
(select
    c.customer_state,
round(avg(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,
DAY)),2) as avg_delivery_time,

from case_study.orders o
inner join case_study.customers c on o.customer_id = c.customer_id
group by 1
order by 2 desc
limit 5)
UNION ALL
(select
    c.customer_state,
    round(avg(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2) as lowest_avg_delivery_time
from case_study.orders o
inner join case_study.customers c on o.customer_id = c.customer_id
group by 1
order by 2
limit 5);
```

Insights: -

1. The first 5 output are highest 5 average delivery time
2. The last 5 outputs are lowest 5 average delivery time
3. Average delivery time is highest in state “RR”.
4. Average delivery time is lowest in state “SP”.

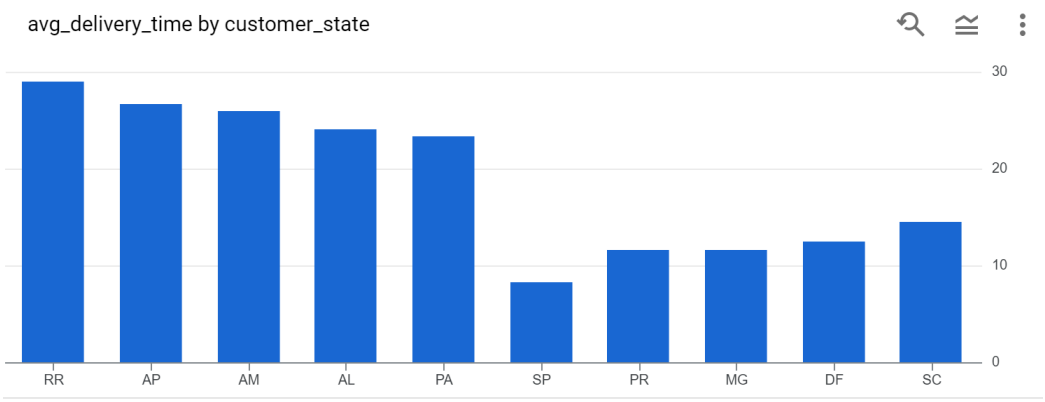
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION D

Row	customer_state	avg_delivery_time
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

Chart: -



4. 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.

Ans: -

```
with fastDelivery as
        (select
            c.customer_state as state,
            round(avg(DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, DAY)),2) as avg_delivery_time

        from case_study.orders o
```

```
        inner join case_study.customers c on o.customer_id = c.customer_id
        group by 1)

select
state,
avg_delivery_time
from fastDelivery
order by 2
limit 5;
```

Insights: -

1. Average delivery time of the states are gotten in order the to identify the 5 states having fastest delivery time.
2. The state having the fastest delivery time is “SP”.

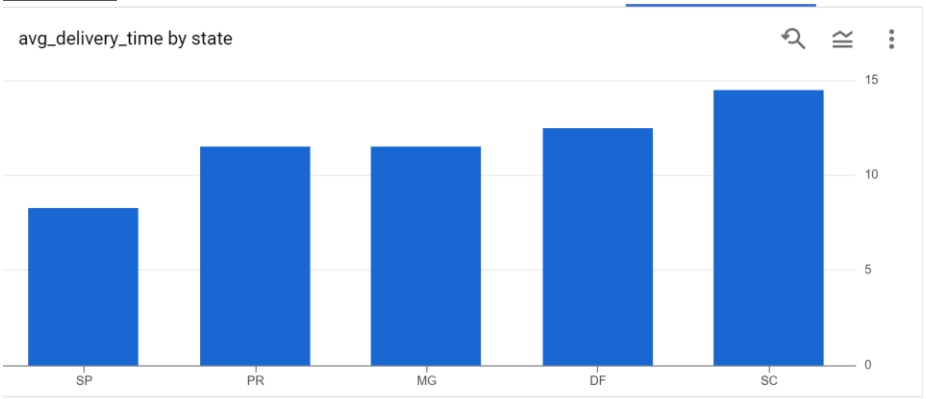
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION

Row	state	avg_delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Chart: -



6.Analysis based on the payments:

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

Ans: -

```
select
```

```sql
extract(month from o.order_purchase_timestamp) as month_no,
p.payment_type,
count(*) as no_of_orders
from
case_study.payments p
inner join case_study.orders o on p.order_id = o.order_id
group by 1,2
order by 1;
```

Insight: -

1. In month of January maximum orders were placed using credit card.
2. Least of orders were placed using debit card.
3. Overall customers like to use credit card the most and debit card the least.

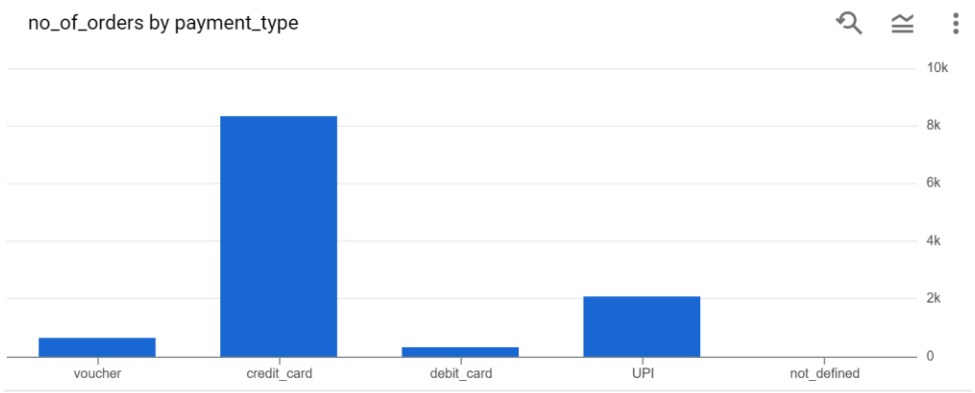
Output: -

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	month_no	payment_type	no_of_orders
1	1	voucher	477
2	1	credit_card	6103
3	1	debit_card	118
4	1	UPI	1715
5	2	credit_card	6609
6	2	voucher	424
7	2	UPI	1723
8	2	debit_card	82
9	3	voucher	591
10	3	credit_card	7707

Chart: -



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

Ans: -

```
select
count(*) as done_pay_installment
from
case_study.payments
where
payment_installments != 0;
```

Insight: -

1. Total of 103884 orders were placed on the basis of payment installment that have been paid.

Output: -

Query results

JOB INFORMATION | RESULTS

Row	done_pay_installment
1	103884

----------END----------