Business Case Study on "Target": SQL

Prepared by: **Anchal Raina**

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.

Ques 1. 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 eastern-dream-402409.target_SQL.INFORMATION_SCHEMA.COLUMNS
 where table_name = 'customers';
```

Output Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW | JSON

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: We can see most of the column names are of string data types and only customer_zip_code_prefix was stored in Integer datatype.

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_SQL.orders`
```

Output Screenshot:

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

Insights: First order was placed on 2016-09-04 at 21:15:19 UTC and last order was placed on 2018-10-17 at 17:30:18 UTC.

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

Query1: As it is asked to give the count of cities and states of customers who ordered during the 2016 to 2018 period.

```
Select count(distinct c.customer_city) as Total_cities,
       count(distinct c.customer_state) as Total_states,
from `target_SQL.orders` o
join `target_SQL.customers` c
on o.customer_id = c.customer_id
where o.order_purchase_timestamp between(select min(order_purchase_timestamp) as
first_order from `target_SQL.orders`) and (select
max(order_purchase_timestamp) as last_order from `target_SQL.orders`)
```

Output Screenshot:

Row	Total_cities	Total_states
1	4119	27

Query2: This query will show us the city and states and no of orders purchased respectively.

```
Select distinct c.customer_city, c.customer_state,
count(o.order_id) as order_count
from `target_SQL.orders` o
join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1, 2
order by 3 desc;
```

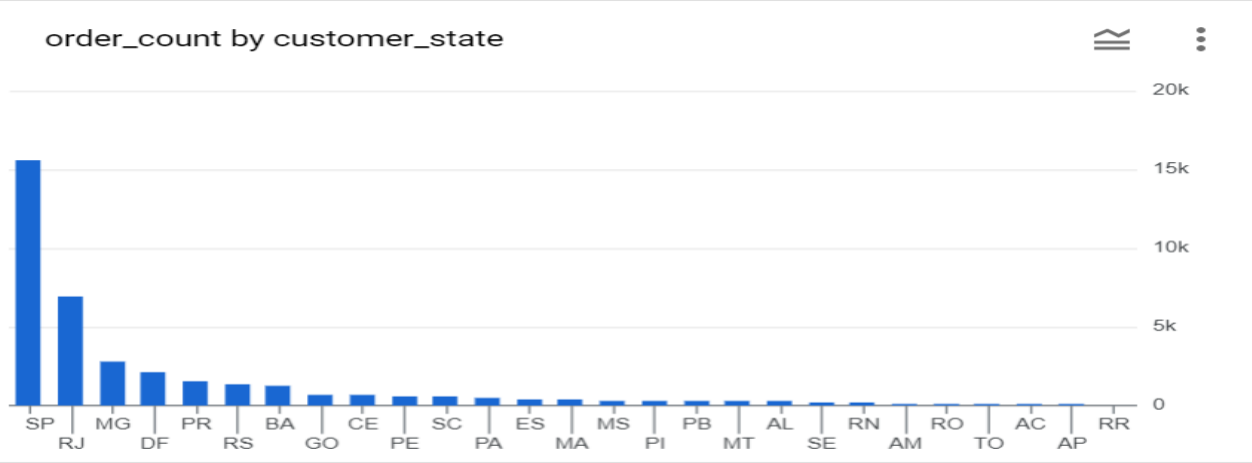
Output Screenshot:

Row	customer_city	customer_state	order_count
1	sao paulo	SP	15540
2	rio de janeiro	RJ	6882
3	belo horizonte	MG	2773
4	brasilia	DF	2131
5	curitiba	PR	1521
6	campinas	SP	1444
7	porto alegre	RS	1379
8	salvador	BA	1245
9	guarulhos	SP	1189
10	sao bernardo do campo	SP	938

Results per page: 50 1 – 50 of 4310

Insights: The above screenshots show only 10 rows however in the result I have got 4310 rows. We can see the cities and states along with their order counts. For example: The city 'Sao Paulo' and state 'SP' has an order count of 15540.

Chart:



Ques 2. In-depth Exploration:

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

Query:

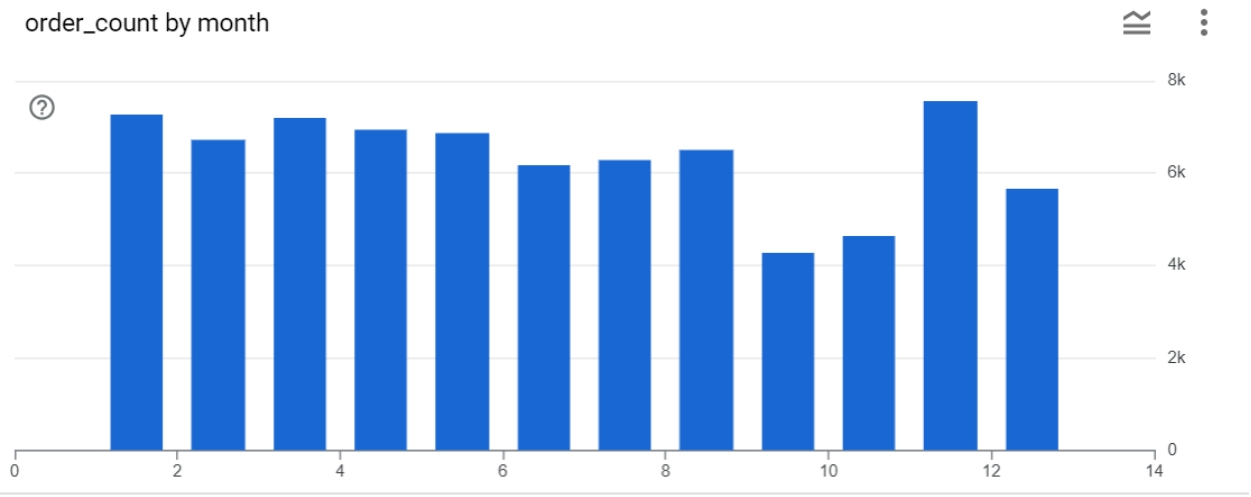
```
select extract(year from o.order_purchase_timestamp) as year,
    extract(month from o.order_purchase_timestamp) as month,
    count(distinct o.order_id) as order_count,
    round(sum(p.payment_value), 2) as revenue
from `target_SQL.orders` o
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
inner join `target_SQL.payments` p
on o.order_id = p.order_id
group by year, month
order by year, month
```

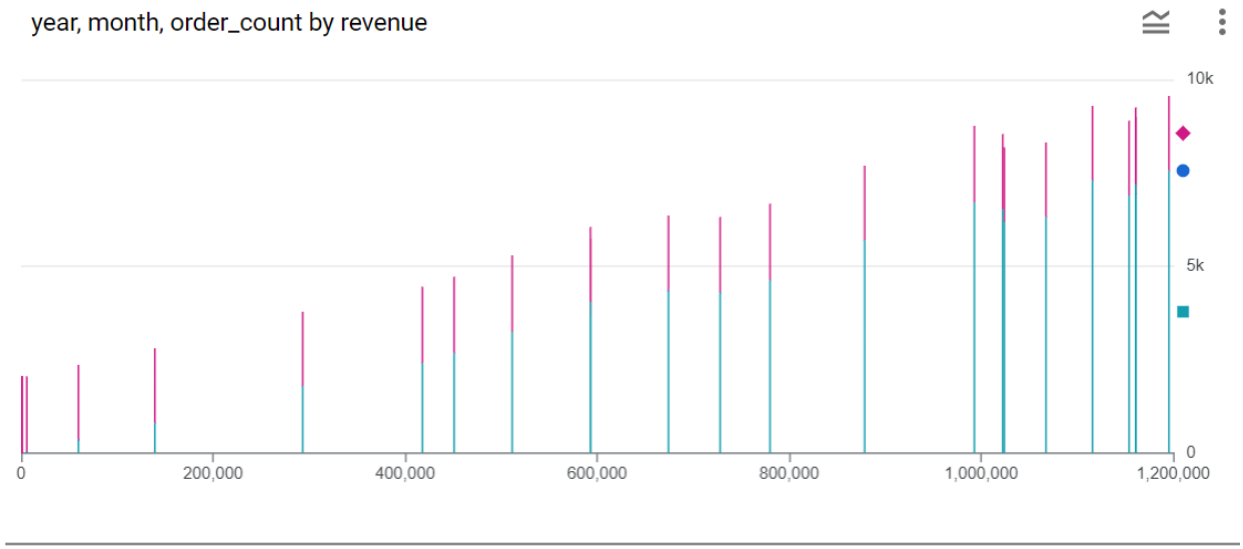
Output Screenshot:

Row	year	month	order_count	revenue
1	2016	9	3	252.24
2	2016	10	324	59090.48
3	2016	12	1	19.62
4	2017	1	800	138488.04
5	2017	2	1780	291908.01
6	2017	3	2682	449863.6
7	2017	4	2404	417788.03
8	2017	5	3700	592918.82
9	2017	6	3245	511276.38
10	2017	7	4026	592382.92

Insights: Based on the above data analysis of order count, it is observed that there is a growth in Brazil. The count of orders and the revenue has shown an overall upward trend, with some fluctuations. Refer to the output above and chart below:

Chart:





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

Query:

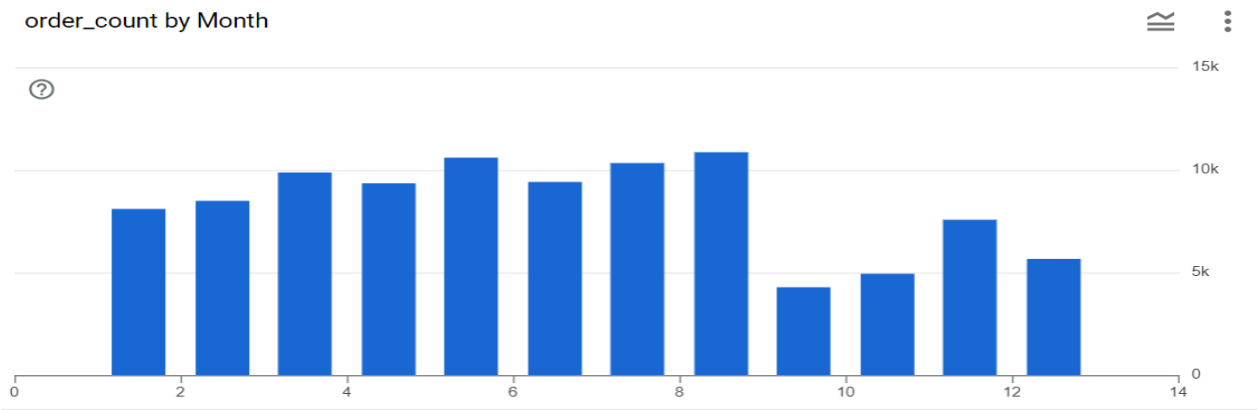
```
select extract(month from order_purchase_timestamp) as Month,
    count(distinct order_id) as order_count
from `target_SQL.orders`
group by month
order by month
```

Output Screenshot:

Row	Month	order_count
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

Insights: From the above data, we can see the order count along with respective months. The count of orders generally increases from March to August with fluctuations in between.

Chart:



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**

Query:

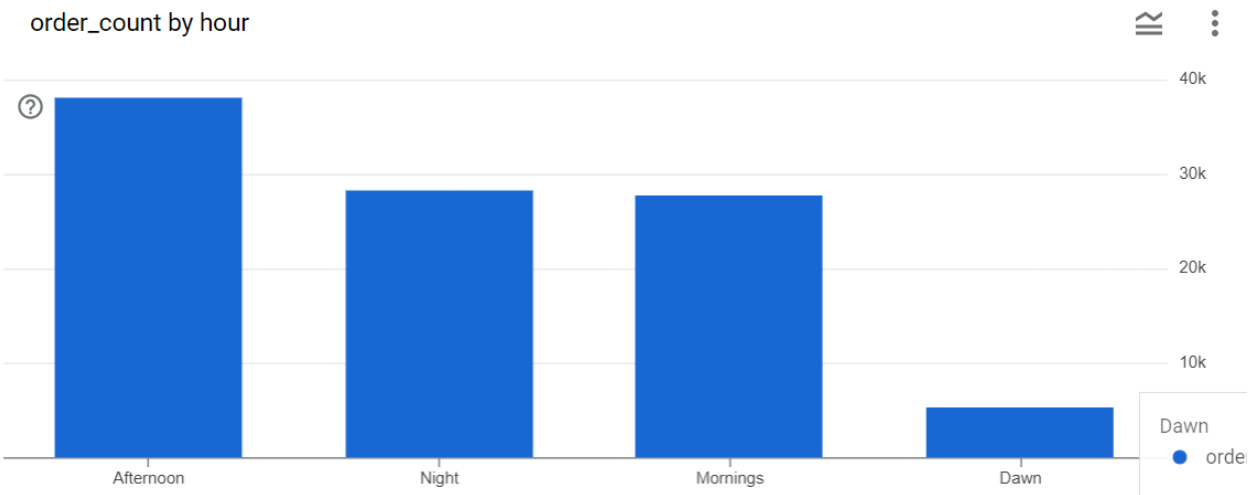
```
select case
    when extract(hour from o.order_purchase_timestamp) between 0 and 6 then 'Dawn'
    when extract(hour from o.order_purchase_timestamp) between 7 and 12 then 'Mornings'
    when extract(hour from o.order_purchase_timestamp) between 13 and 18 then 'Afternoon'
    when extract(hour from o.order_purchase_timestamp) between 19 and 23 then 'Night'
    end as hour,
count(o.order_id) as order_count
from `target_SQL.orders` o
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by hour
order by order_count desc;
```

Output Screenshot:

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

Insights: The SQL query was executed to categorize the order purchase timestamps into four periods: dawn, morning, afternoon, and night. Based on the analysis, we found that Brazilian customers tend to place most orders during the daytime, specifically in the afternoon and night.

Chart:



Ques 3. 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_state,
extract(month from o.order_purchase_timestamp) as Month,
count(o.order_id) as order_count
from `target_SQL.orders` o
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by c.customer_state, Month
order by c.customer_state, Month
```

Output Screenshot:

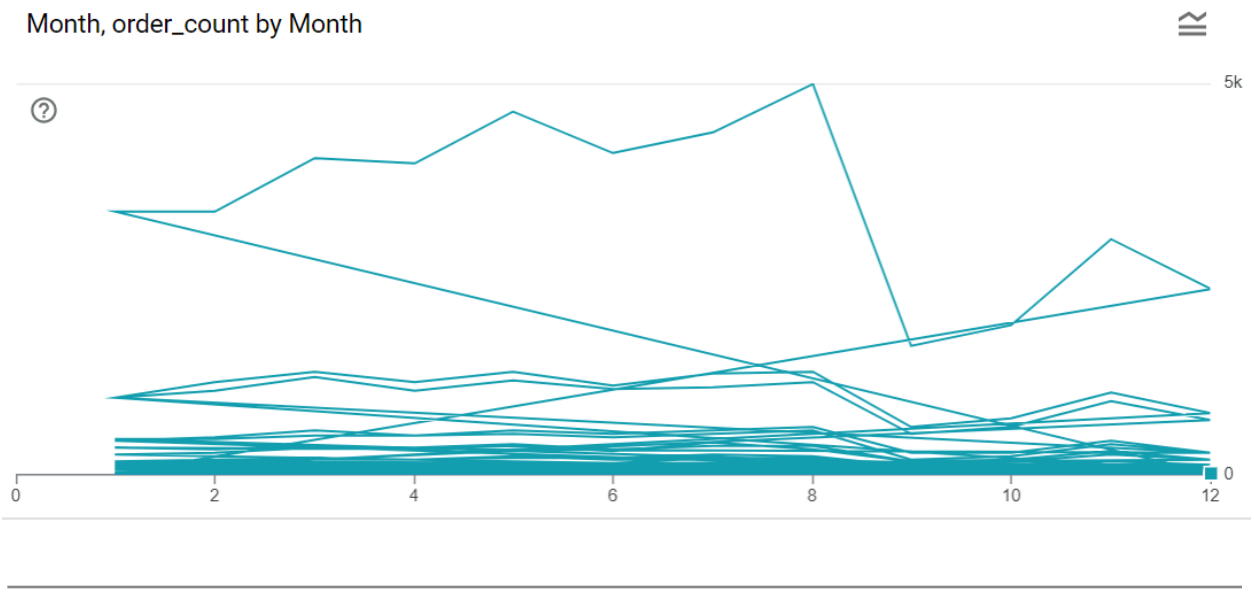
Row	customer_state	Month	order_count
1	AC	1	8
2	AC	2	6
3	AC	3	4
4	AC	4	9
5	AC	5	10
6	AC	6	7
7	AC	7	9
8	AC	8	7
9	AC	9	5
10	AC	10	6
11	AC	11	5
12	AC	12	5
13	AL	1	39
14	AL	2	39

Results per page: 50 1 – 50 of 322

Row	customer_state	Month	order_count
301	SP	3	4047
302	SP	4	3967
303	SP	5	4632
304	SP	6	4104
305	SP	7	4381
306	SP	8	4982
307	SP	9	1648
308	SP	10	1908
309	SP	11	3012
310	SP	12	2357

Insights: We analyzed the month-on-month order counts for each state. It is evident that São Paulo (SP) consistently has the highest number of orders in any given month, followed by Rio de Janeiro (RJ) and Minas Gerais (MG).

Chart:



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

Query:

```
select customer_state,
     count(customer_id) as no_of_customers
from  `target_SQL.customers`
group by 1
order by 2 desc
```

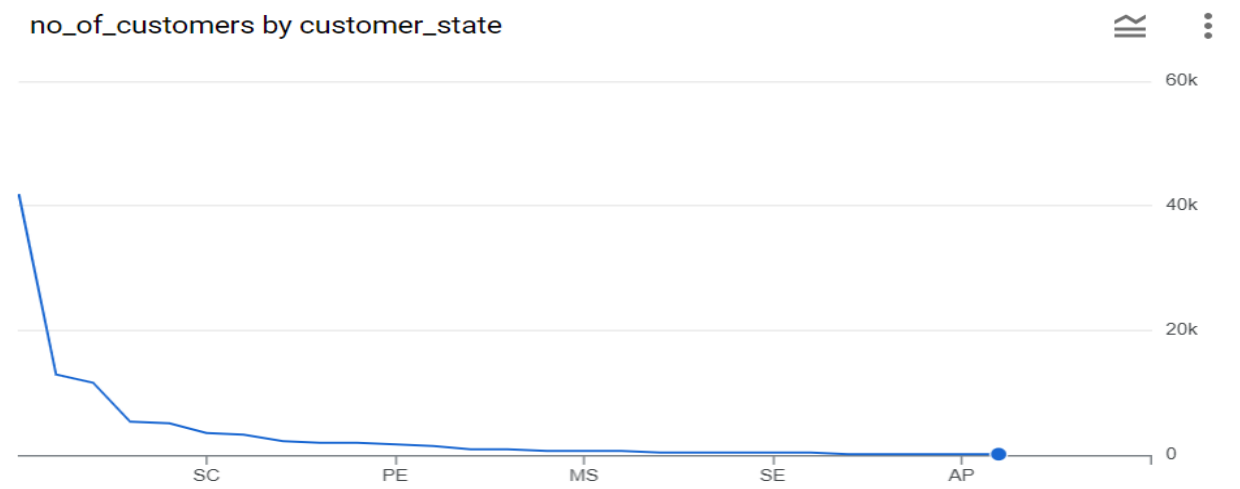
Output Screenshot:

Row	customer_state	no_of_customers
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

Results per page: 50 1 – 27 of 27

Insights: The data shows that Sao Paulo (SP) has the highest number of customers in Brazil.

Chart:



Ques 4. 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).*
Hint: You can use the "payment_value" column in the payments table to get the cost of orders.

Query:
To calculate the percentage increase:
First: work out the difference (increase) between the two numbers you are comparing. Then: divide the increase by the original number and multiply the answer by 100. % increase = Increase ÷ Original Number × 100.
Percent increase = [(new value - original value) / original value] x 100.

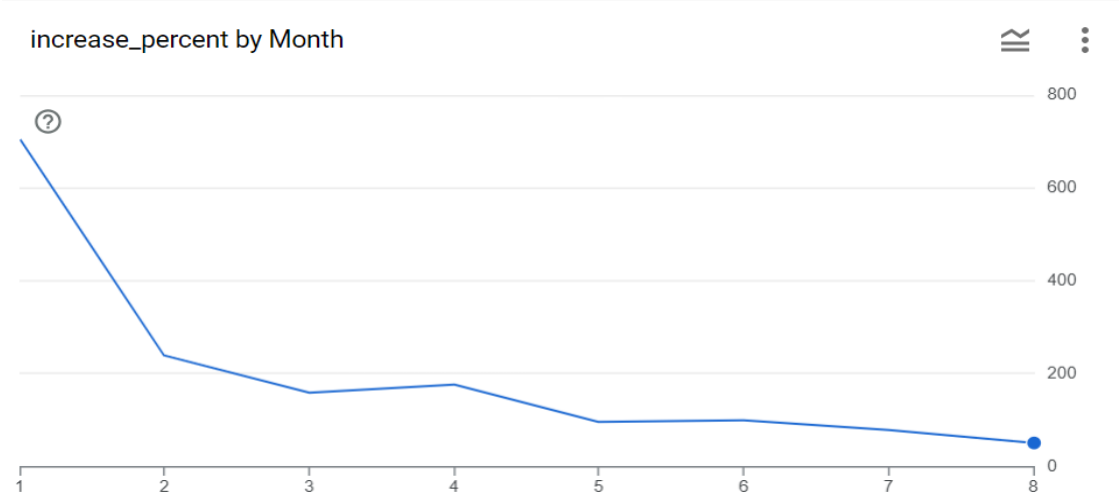
```
select extract(month from o.order_purchase_timestamp) as Month,
    ((sum(case
          when extract(year from o.order_purchase_timestamp) = 2018 and
          extract(month from o.order_purchase_timestamp) between 1 and 8
          then p.payment_value end)
    -
    sum(case
          when extract(year from o.order_purchase_timestamp) = 2017 and
          extract(month from o.order_purchase_timestamp) between 1 and 8
          then p.payment_value end))
    /
    sum(case
          when extract(year from o.order_purchase_timestamp) = 2017 and
          extract(month from o.order_purchase_timestamp) between 1 and 8
          then p.payment_value end)
          ) *100 as increase_percent
from `target_SQL.orders` o
inner join `target_SQL.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
order by 1
```

Output Screenshot:

Row	Month	increase_percent
1	1	705.1266954171...
2	2	239.9918145445...
3	3	157.7786066709...
4	4	177.8407701149...
5	5	94.62734375677...
6	6	100.2596912456...
7	7	80.04245463390...
8	8	51.60600520477...

Insights: We calculated percent increase from the year 2017 to 2018 including months from January to August and we have the highest percent increase from January month followed by February and April as shown in the output.

Chart:



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

Query:

```
select c.customer_state,
    round(sum(oi.price),2) as Total_price,
    round(avg(oi.price),2) as Average_price
from `target_SQL.orders` o
inner join `target_SQL.order_items` oi
on o.order_id = oi.order_id
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 1
```

Insights: The data shows Sao Paulo (SP) has the highest total price value (5202955.05); it surprisingly has the lowest average price value (109.65) among all states.

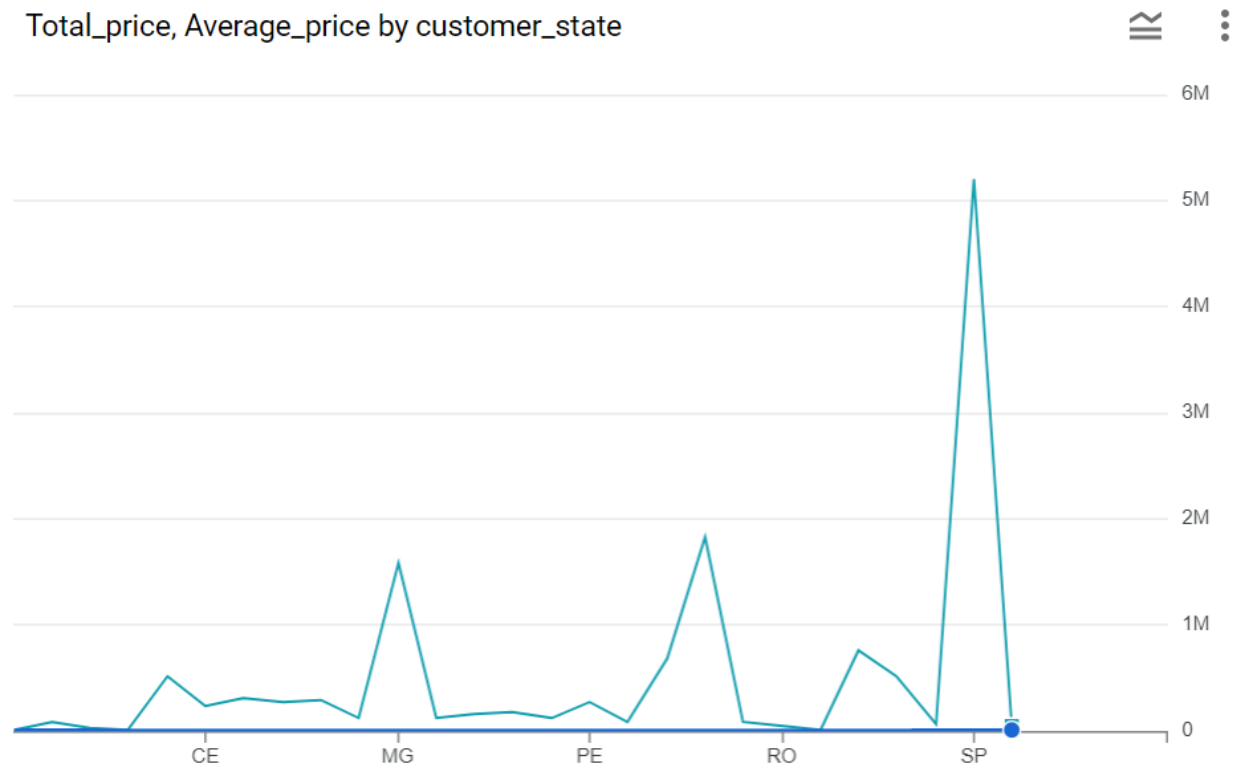
On the other hand, the state of Paraiba (PB) has the highest average price value (191.48).

Output Screenshot:

Row	customer_state	Total_price	Average_price
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2
11	MG	1585308.03	120.75
12	MS	116812.64	142.63
13	MT	156453.53	148.3
14	PA	178947.81	165.69

Results per page: 50 1 – 27 of 27

Chart:



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

Query:

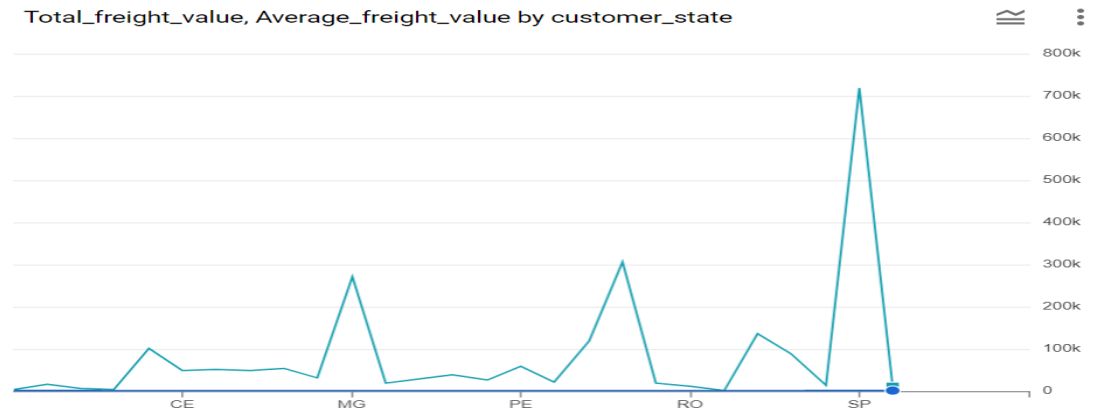
```
select c.customer_state,
    round(sum(oi.freight_value),2) as Total_freight_value,
    round(avg(oi.freight_value),2) as Average_freight_value
from `target_SQL.orders` o
inner join `target_SQL.order_items` oi
on o.order_id = oi.order_id
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 1
```

Output Screenshot:

Row	customer_state	Total_freight_value	Average_freight_valu
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26
11	MG	270853.46	20.63
12	MS	19144.03	23.37
13	MT	29715.43	28.17
14	PA	38699.3	35.83

Results per page: 50 1 – 27 of 27

Chart:



Insights: The data shows Sao Paulo (SP) has the highest total freight value (718723.07); it surprisingly has the lowest average freight value (15.15) among all states.
On the other hand, the state of Roraima (RR) has the highest average freight value (42.98).

Ques 5. 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 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

Query1: This query is a simplified version where we get the time to deliver and the difference between estimated and actual delivery data.

```
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_SQL.orders`
order by order_id
```

Query2: In addition to, As there are few Null values in the data so applied filter conditions to remove null values.

```
select order_id,
date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as
estimated_delivery_in_days,
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_SQL.orders`
where date_diff(order_estimated_delivery_date, order_delivered_customer_date, day) is not null
and date_diff(order_delivered_customer_date, order_purchase_timestamp, day) is not null
order by order_id
```

Output Screenshot:

Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08b...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18

Results per page: 50 1 – 50 of 99441

Insights: To understand the time duration between purchasing an order, its delivery, and the estimated delivery, we calculated the number of days.

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

```
with Final as(
(Select c.customer_state,
'highest avg state' as status,
avg(freight_value) as avg_freight_value
from `target_SQL.orders` o
inner join `target_SQL.order_items` oi
on o.order_id = oi.order_id
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 3 desc
limit 5)
union distinct
(Select c.customer_state,
'lowest avg state' as status,
avg(freight_value) as avg_freight_value
from `target_SQL.orders` o
```

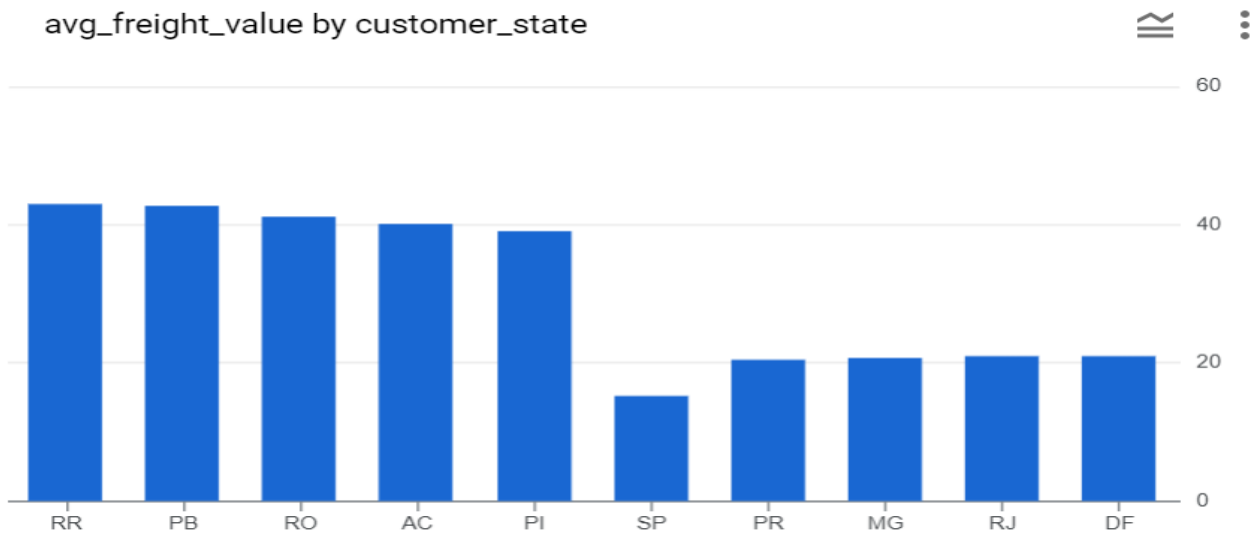
```
inner join `target_SQL.order_items` oi
on o.order_id = oi.order_id
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 3
limit 5))

Select *
from final
order by status
```

Output Screenshot:

Row	customer_state	status	avg_freight_value
1	RR	highest avg state	42.98442307692...
2	PB	highest avg state	42.72380398671...
3	RO	highest avg state	41.06971223021...
4	AC	highest avg state	40.07336956521...
5	PI	highest avg state	39.14797047970...
6	SP	lowest avg state	15.14727539041...
7	PR	lowest avg state	20.53165156794...
8	MG	lowest avg state	20.63016680630...
9	RJ	lowest avg state	20.96092393168...
10	DF	lowest avg state	21.04135494596...

Insights: As shown above, in the data analysis we can see the highest and lowest freight values with top 5 states.

Chart:



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

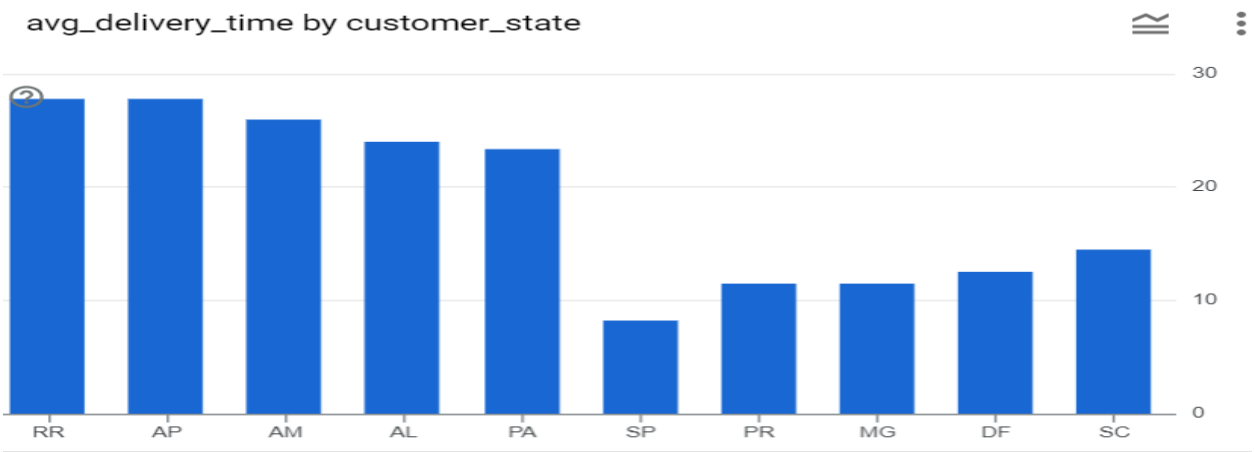
```
with Final as(
(Select c.customer_state,
'Highest avg delivery time' as status,
avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as
avg_delivery_time
from `target_SQL.orders` o
inner join `target_SQL.order_items` oi
on o.order_id = oi.order_id
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 3 desc
limit 5)

union distinct

(Select c.customer_state,
'Lowest avg delivery time' as status,
avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)) as
avg_delivery_time
from `target_SQL.orders` o
inner join `target_SQL.order_items` oi
on o.order_id = oi.order_id
inner join `target_SQL.customers` c
on o.customer_id = c.customer_id
group by 1
order by 3
limit 5))


Select *
from final
order by status
```

Output Screenshot:

Row	customer_state	status	avg_delivery_time
1	RR	Highest avg delivery time	27.82608695652...
2	AP	Highest avg delivery time	27.75308641975...
3	AM	Highest avg delivery time	25.96319018404...
4	AL	Highest avg delivery time	23.99297423887...
5	PA	Highest avg delivery time	23.30170777988...
6	SP	Lowest avg delivery time	8.259608552419...
7	PR	Lowest avg delivery time	11.48079306071...
8	MG	Lowest avg delivery time	11.51552218007...
9	DF	Lowest avg delivery time	12.50148619957...
10	SC	Lowest avg delivery time	14.52098584675...

Insights: As shown above, in the data analysis we can see the highest and lowest average time of delivery with top 5 states.

Chart:



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.

Query 1: Using date_diff

```
Select c.customer_state,
ROUND(avg(date_diff(o.order_estimated_delivery_date, o.order_delivered_customer_date, DAY)),2)
as avg_diff_estimated_delivery
from `target_SQL.orders` o
JOIN `target_SQL.customers` c
ON o.customer_id = c.customer_id
group by c.customer_state
order by 2
limit 5
```

OR

Query 2: Using timestamp_diff

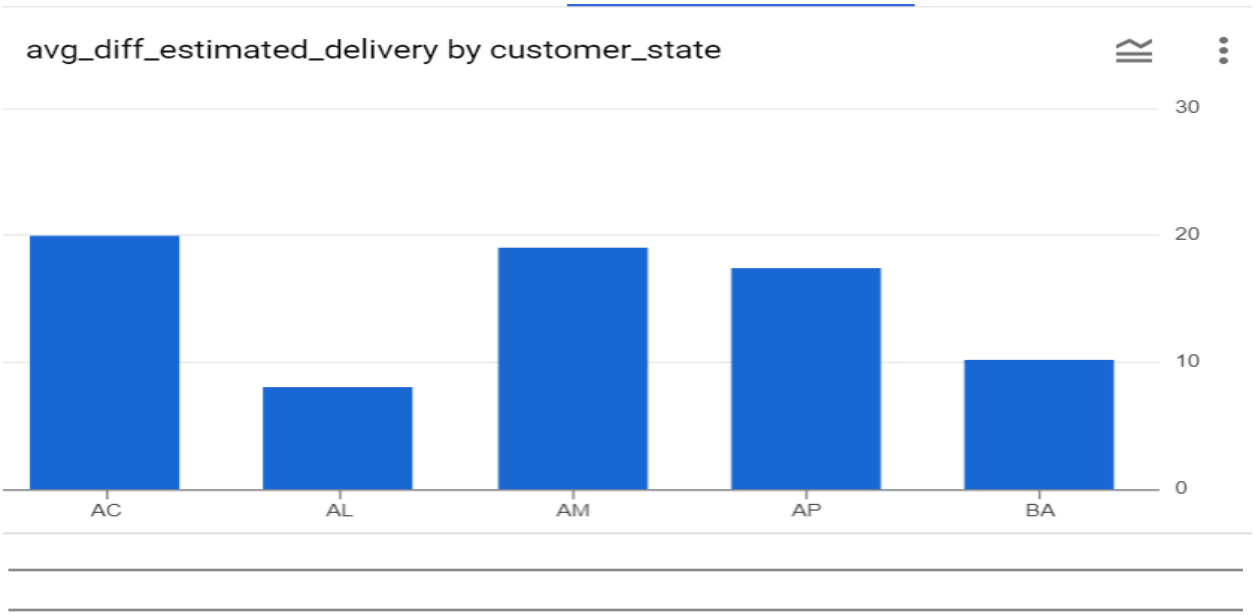
```
select customer_state, ROUND(avg(timestamp_diff(order_estimated_delivery_date,
order_delivered_customer_date, DAY)), 2) as avg_diff_estimated_delivery

from `target_SQL.orders` o
JOIN `target_SQL.customers` c
ON o.customer_id = c.customer_id
group by customer_state
order by 2
limit 5
```

Output Screenshot:

customer_state	avg_diff_estimated_delivery
AC	20.01
AL	7.98
AM	18.98
AP	17.44
BA	10.12

Insights: Here are the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

Chart:



Ques 6. 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 Month,
    count(distinct o.order_id) as no_of_orders
from `target_SQL.orders` o
inner join `target_SQL.payments` p
on o.order_id = p.order_id
group by 1,2
order by 1,2
```

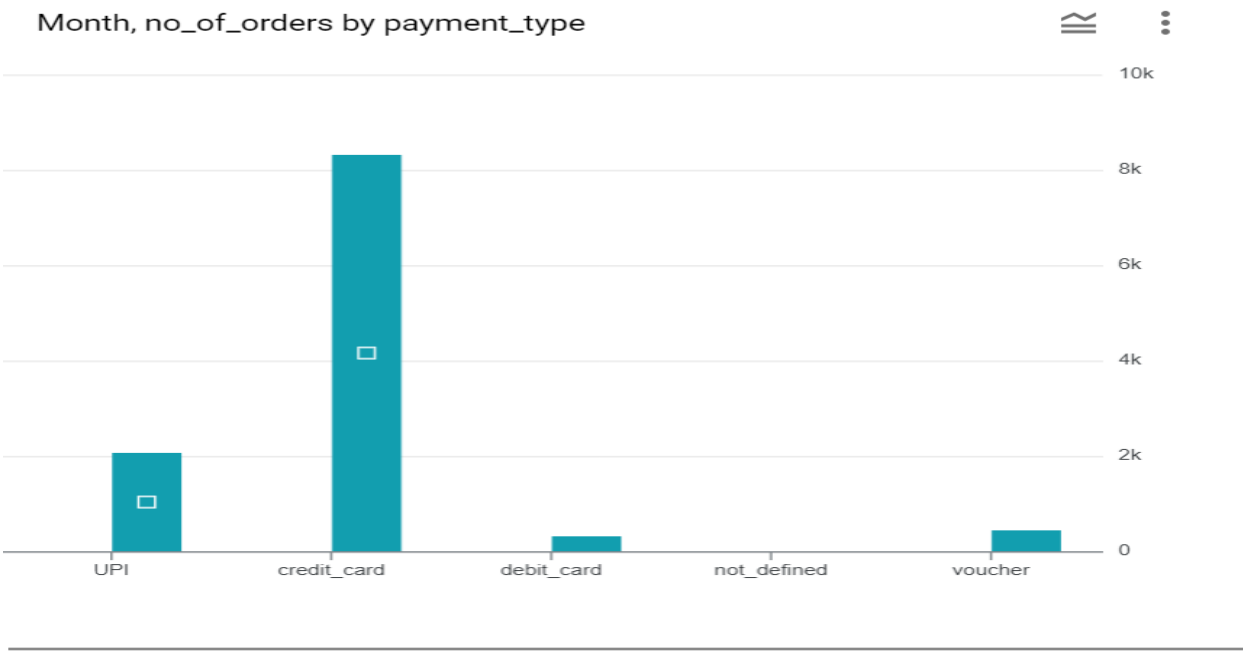
Output Screenshot:

Row	payment_type	Month	no_of_orders
1	UPI	1	1715
2	UPI	2	1723
3	UPI	3	1942
4	UPI	4	1783
5	UPI	5	2035
6	UPI	6	1807
7	UPI	7	2074
8	UPI	8	2077
9	UPI	9	903
10	UPI	10	1056
11	UPI	11	1509
12	UPI	12	1160
13	credit_card	1	6093
14	credit_card	2	6582

Results per page: 50 1 – 50 of 50

Insights: The analysis shows an overall uptrend from January to August and card transactions are the most popular payment method, followed by UPI. Debit card transactions are the least preferred option.

Chart:



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

Query:

```
select p.payment_installments,
count(distinct o.order_id) as no_of_orders
from `target_SQL.orders` o
inner join `target_SQL.payments` p
on o.order_id = p.order_id
where o.order_status != 'canceled' and payment_installments != 0
group by 1
order by 2 desc
```

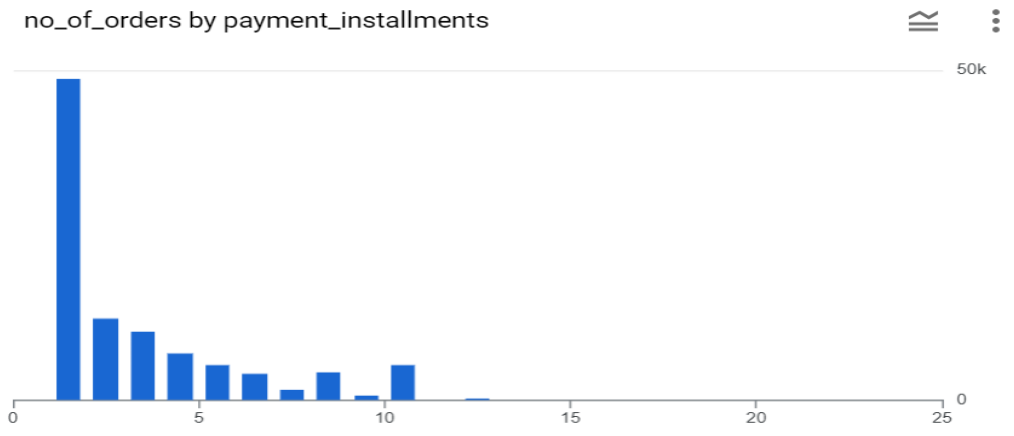
Output Screenshot:

Row	payment_installment	no_of_orders
1	1	48732
2	2	12329
3	3	10374
4	4	7046
5	10	5279
6	5	5204
7	8	4224
8	6	3894
9	7	1617
10	9	638

Results per page: 50 1 – 23 of 23

Insights: The data shows that the majority of orders have only one payment installment. The highest number of installments is 24, which is associated with 18 orders.

Chart:



Actionable Insights

- The data reveals that the state of SP has significantly more orders than the next five states combined. This indicates an opportunity for improvement in the other states. Focusing on these states can help increase the number of orders.
- With increased sales during festive periods, Seasonal variations in sales are observed. Businesses should plan their marketing and sales strategies and enhance customer satisfaction, resulting in overall sales growth.
- Improving delivery times in areas with longer delivery durations can have a positive impact on customer satisfaction and encourage repeat purchases.
- States like SP and RJ already have high order counts. To further boost sales and foster brand loyalty, it is recommended to focus on customer retention strategies, such as personalized marketing campaigns, loyalty programs, and exceptional customer service experiences.
- Analyzing customer demographics can provide valuable insights for tailoring products and marketing strategies to specific target audiences.
- The data indicates a decline in orders during September and October. Offering discounts or promotions during off-peak seasons can incentivize customers to make purchases during these periods

Conclusion

In conclusion, the analysis of e-commerce data in the Brazilian market provides valuable insights into customer buying patterns, sales trends, payment preferences, and delivery experiences. By understanding these patterns and trends, businesses can make informed decisions and implement strategies to optimize their operations and drive growth.

Thank You!