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.

Query:

```
select table_name,column_name,data_type from scaler-dsml-sql-399507.target_sql.INFORMATION_SCHEMA.COLUMNS where table_name ='customers'
```

Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON | EXECUTION DETAILS

Row	table_name	column_name	data_type
1	customers	customer_id	STRING
2	customers	customer_unique_id	STRING
3	customers	customer_zip_code_prefix	INT64
4	customers	customer_city	STRING
5	customers	customer_state	STRING

Insight: The Customer table contains information related to the Target's customers. Key information like Customer ID and customer unique ID which are helpful in retrieving the data respective to the customers.

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

Query:

```
SELECT min(order_purchase_timestamp) start_order, max(order_purchase_timestamp) last_order FROM target_sql.orders
```

Screenshot:

JOB INFORMATION | RESULTS | CHART PREVIEW | JSON

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

Insight: Orders table contains data related to all the customer respective to date, timestamp, order status, estimated delivery date, etc.,

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

Query:

```
SELECT count(distinct c.customer_city)
city_count,count(distinct c.customer_state) state_count  from
target_sql.customers c
inner join target_sql.orders o
on c.customer_id = o.customer_id
where order_purchase_timestamp between (SELECT
min(order_purchase_timestamp) FROM target_sql.orders ) and
(SELECT max(order_purchase_timestamp) FROM target_sql.orders
);
```

Screenshot:

Query results

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	city_count	state_count
1	4119	27

Insight: There are totally 8011 cities in the geolocation table, however, customers from 4000+ cities only have done purchases during this period.

---

2. **In-depth Exploration:**

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

Query:

```
select distinct EXTRACT(YEAR FROM order_purchase_timestamp)
yr,EXTRACT(MONTH FROM order_purchase_timestamp) month,
count(*) over(partition by EXTRACT(YEAR FROM
order_purchase_timestamp),EXTRACT(MONTH FROM
order_purchase_timestamp) ) no_of_orders_per_month,
from target_sql.orders
order by 1,2
```

Screenshot:

yr	month	no_of_orders_per_month
2016	9	4
2016	10	324
2016	12	1
2017	1	800
2017	2	1780
2017	3	2682
2017	4	2404
2017	5	3700
2017	6	3245
2017	7	4026
2017	8	4331
2017	9	4285
2017	10	4631
2017	11	7544
2017	12	5673
2018	1	7269

Insight: Based on the data provided, the number of orders increased over the years, In the year 2016 the count is very low (per the data provided), whereas we see a significant raise in the years 2017 and 2018.

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

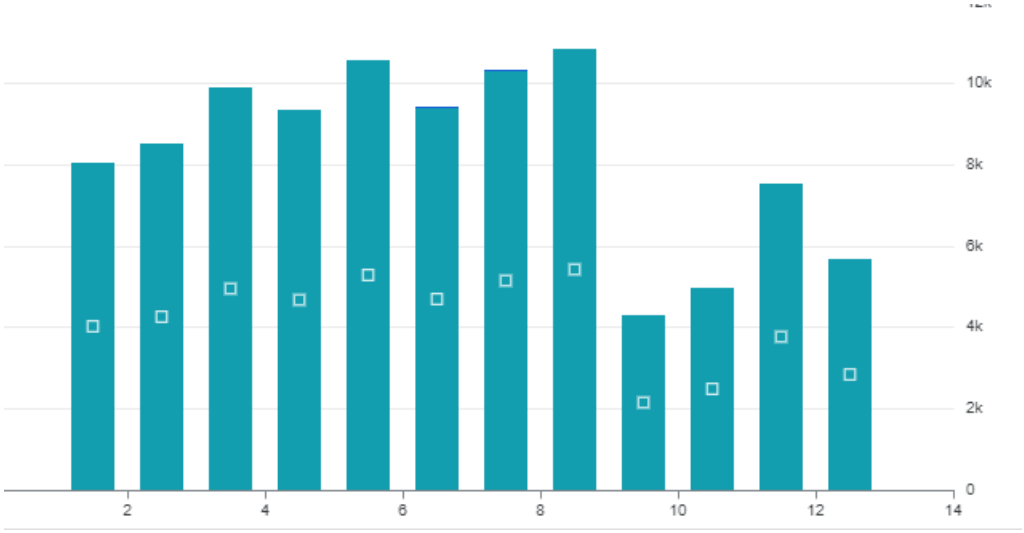
Query:

```
select distinct EXTRACT(MONTH FROM order_purchase_timestamp)
mnth,
count(*) over(partition by EXTRACT(MONTH FROM
order_purchase_timestamp)) orders_per_month
from target_sql.orders
where order_purchase_timestamp between (select
min(order_purchase_timestamp) from target_sql.orders)
and (select max(order_purchase_timestamp) from
target_sql.orders)
order by 1
```

Result:

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

Insight: When we consider the orders per month, irrespective of years, the number of orders from Jan to Aug are at a similar level, whereas the sales in the months of September, October, November, and December is very low.



3. 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 distinct time_of_day, sum(hourly_count) over(partition
by time_of_day) orders_time_frame from (
SELECT distinct EXTRACT(HOUR FROM order_purchase_timestamp)
hr,
count(*) over(partition by EXTRACT(HOUR FROM
order_purchase_timestamp)) hourly_count,
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) between 0 and
6 then 'Dawn 0-6 hrs'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) between 7 and
12 then 'Mornings 7-12 hrs'
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) between 13
and 18 then 'Afternoon 13-18 hrs'
Else 'Night 19-23 hrs'
END as time_of_day
 from target_sql.orders
where order_purchase_timestamp between '2016-01-01' and '2018-
12-31'
)
```

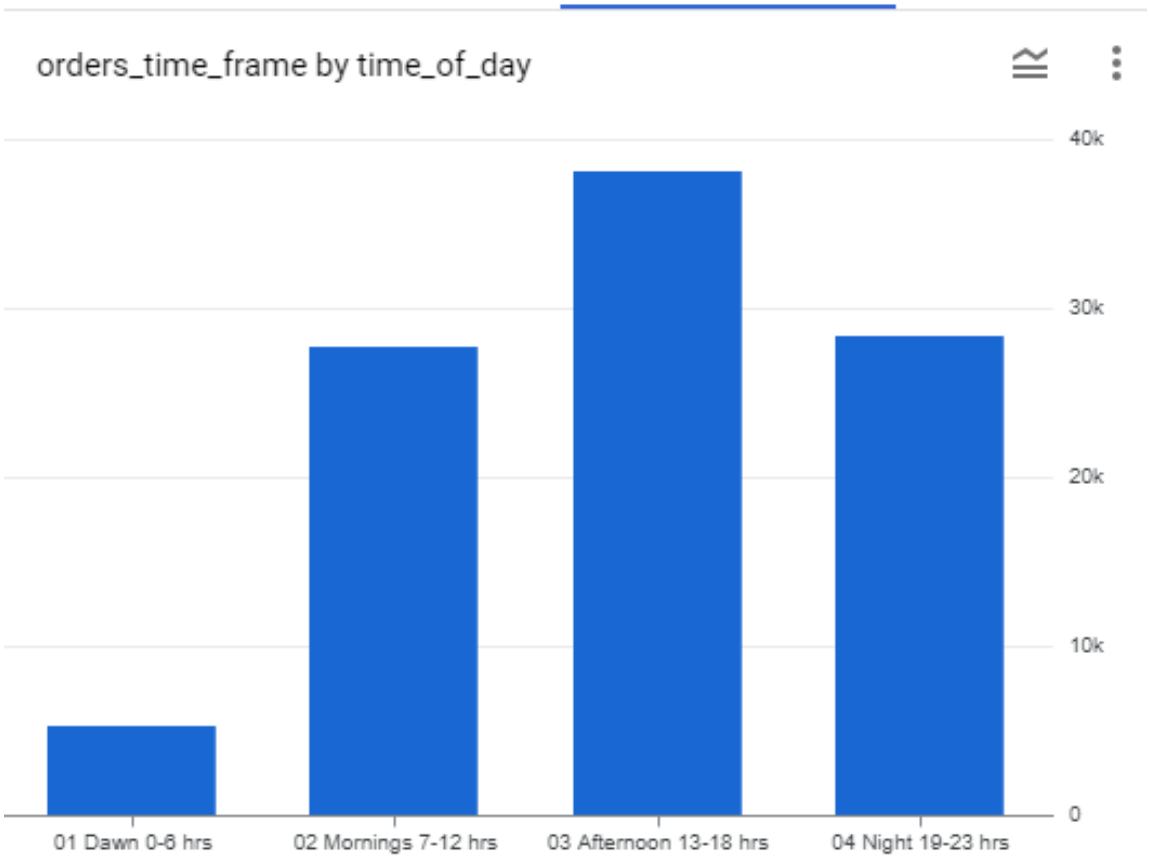
Result:

JOB INFORMATION | RESULTS | CHART | PREVIEW

Row	time_of_day	orders_time_frame
1	Night 19-23 hrs	28331
2	Dawn 0-6 hrs	5242
3	Afternoon 13-18 hrs	38135
4	Mornings 7-12 hrs	27733

Insight:

Based on the data, the highest order placing is happening in the Afternoons, 2nd during the nights, followed by Mornings, and Dawn respectively.



3. **Evolution of E-commerce orders in the Brazil region:**
    1. Get the month on month no. of orders placed in each state.

Query:

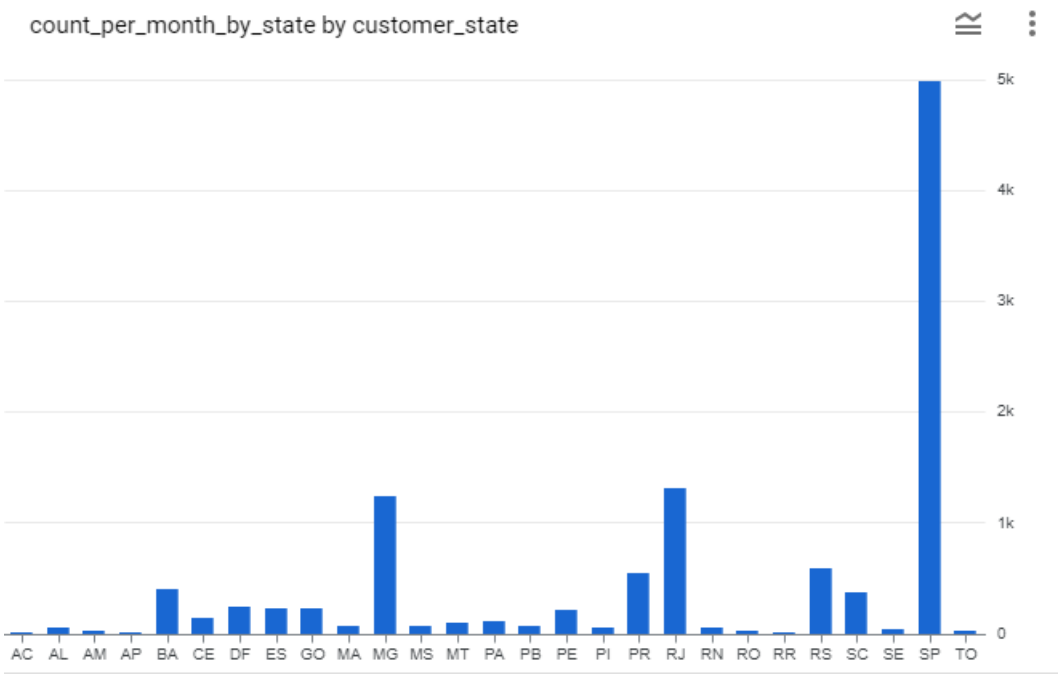
```
select distinct c.customer_state ,
extract(month from order_purchase_timestamp) as Month,
count(*) over(partition by c.customer_state,extract(month from
order_purchase_timestamp)) count_per_month_by_state
from target_sql.orders o
inner join target_sql.customers c
on o.customer_id = c.customer_id
order by c.customer_state, Month
```

Result:

Row	customer_state	Month	count_per_month_by_state
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
15	AL	3	40

Insight:

The State SP has the highest number of orders (41746) during the tenure, followed by RJ and MG touching 12K, 11K respectively. The states RS and PR are at 5K, the states SC and BA are at 5K. Other states are ranging from 3K through too few hundred.



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

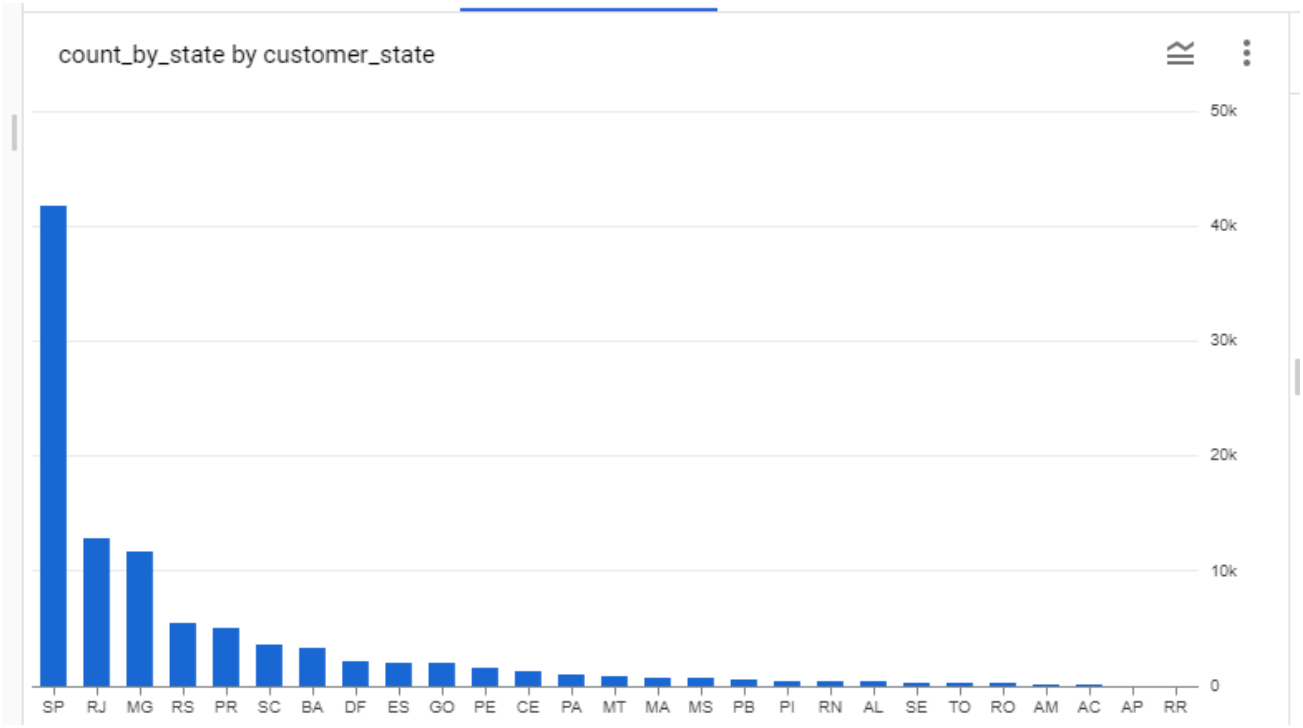
Query:

```
select distinct c.customer_state , count(*) over(partition by
c.customer_state) count_by_state  from target_sql.orders o
inner join target_sql.customers c
on o.customer_id = c.customer_id
order by count_by_state desc,c.customer_state
```

Result:

Row	customer_state	count_by_state
1	SP	41746
2	RJ	12852
3	MG	11635
4	RS	5466
5	PR	5045
6	SC	3637

Insight: The State SP has the highest number of orders (41746) during the tenure, followed by RJ and MG touching 12K, 11K respectively. The states RS and PR are at 5K, the states SC and BA are at 5K. Other states are ranging from 3K through few hundreds.


count_by_state by customer_state
50k
40k
30k
20k
10k
0
SP
RJ
MG
RS
PR
SC
BA
DF
ES
GO
PE
CE
PA
MT
MA
MS
PB
PI
RN
AL
SE
TO
RO
AM
AC
AP
RR

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.

Query:

```
select year, tmp.sum_current_year, following_year,
round((((following_year- tmp.sum_current_year)
/tmp.sum_current_year)*100),2) percentage_increase
from(
select *,
lead(sum_current_year) over(order by year) following_year
from (select distinct
extract(year from order_purchase_timestamp) year,
Sum(payment_value) over(partition by extract(year from
order_purchase_timestamp)) sum_current_year
from target_sql.payments p
inner join target_sql.orders o
on p.order_id = o.order_id
where order_purchase_timestamp between '2017-01-01' and '2018-
08-31 23:59:59.999'
and extract(month from order_purchase_timestamp) in
(1,2,3,4,5,6,7,8)
) tmp
) tmp
where following_year is not null
order by 1
```

Result:

Row	year	sum_current_year	following_year	percentage_increase
1	2017	3669022.12	8694733.84	136.98

Insight: there is a rise of ~137% of payment value from 2017 to 2018.

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

Query :

```
select distinct c.customer_state,round(sum(price)
over(partition by c.customer_state),2) Total_Value,
round(avg(price) over(partition by c.customer_state),2)
Average_Value
FROM
target_sql.order_items p
inner join target_sql.orders s
on p.order_id = s.order_id
inner join target_sql.customers c
on s.customer_id = c.customer_id
order by 2 desc
```

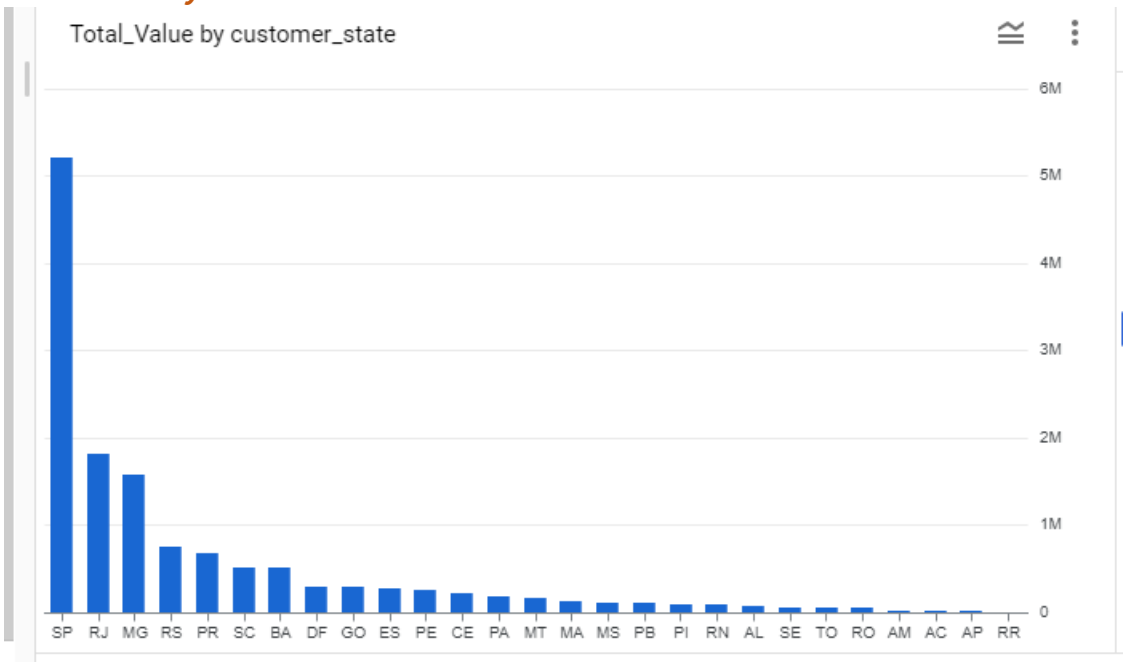
Screenshot:

Row	customer_state	Total_Value	Average_Value
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27

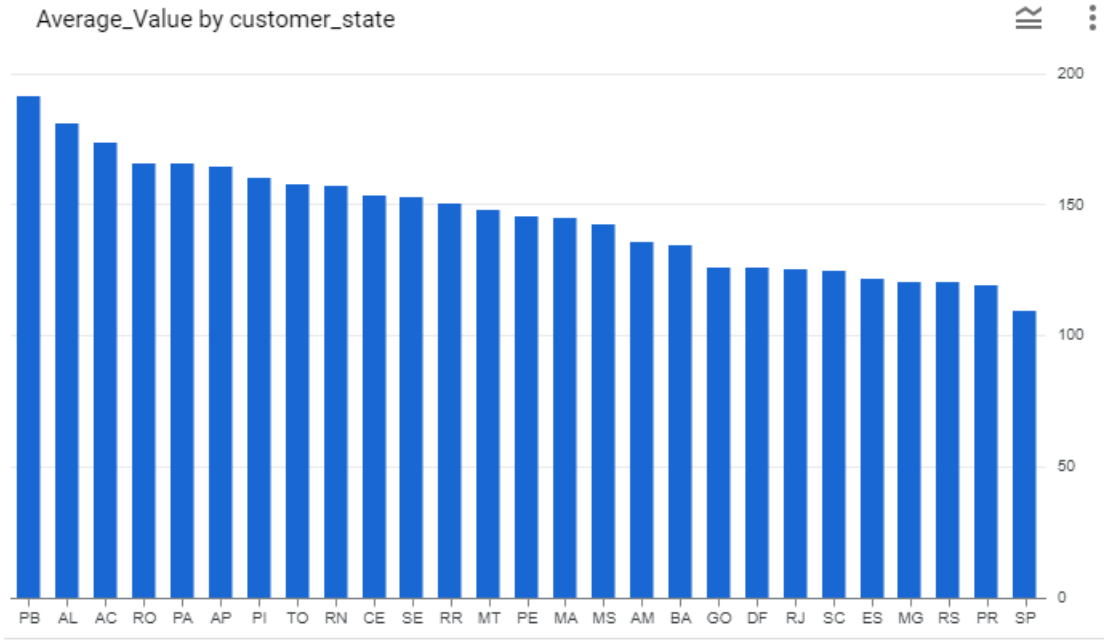
Insight:

The state of SP has the highest total value, whereas the state of PB has the highest average.

Total value by state:



Average value by state:



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

Query:

```
select distinct c.customer_state,round(sum(freight_value)
over(partition by c.customer_state),2) Total_Value,
round(avg(freight_value) over(partition by
c.customer_state),2) Average_Value
FROM
target_sql.order_items p
inner join target_sql.orders s
on p.order_id = s.order_id
inner join target_sql.customers c
on s.customer_id = c.customer_id
order by 2 desc
```

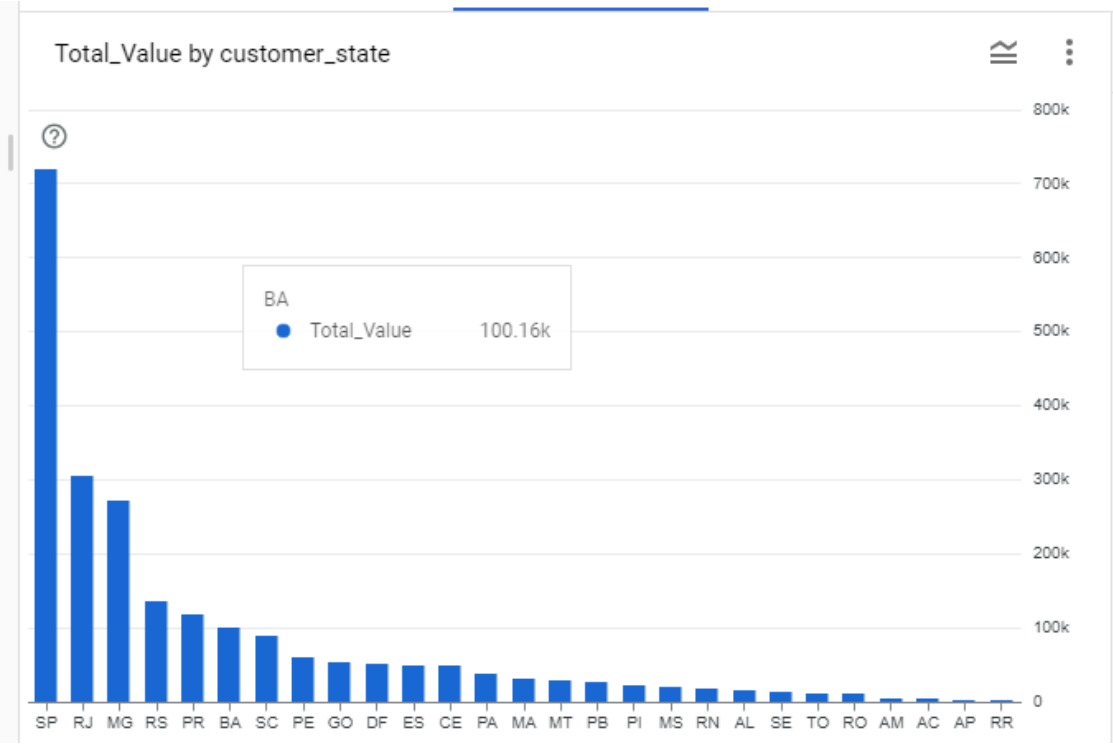
Results:

Row	customer_state	Total_Value	Average_Value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04
11	ES	49764.6	22.06
12	CE	48351.59	32.71

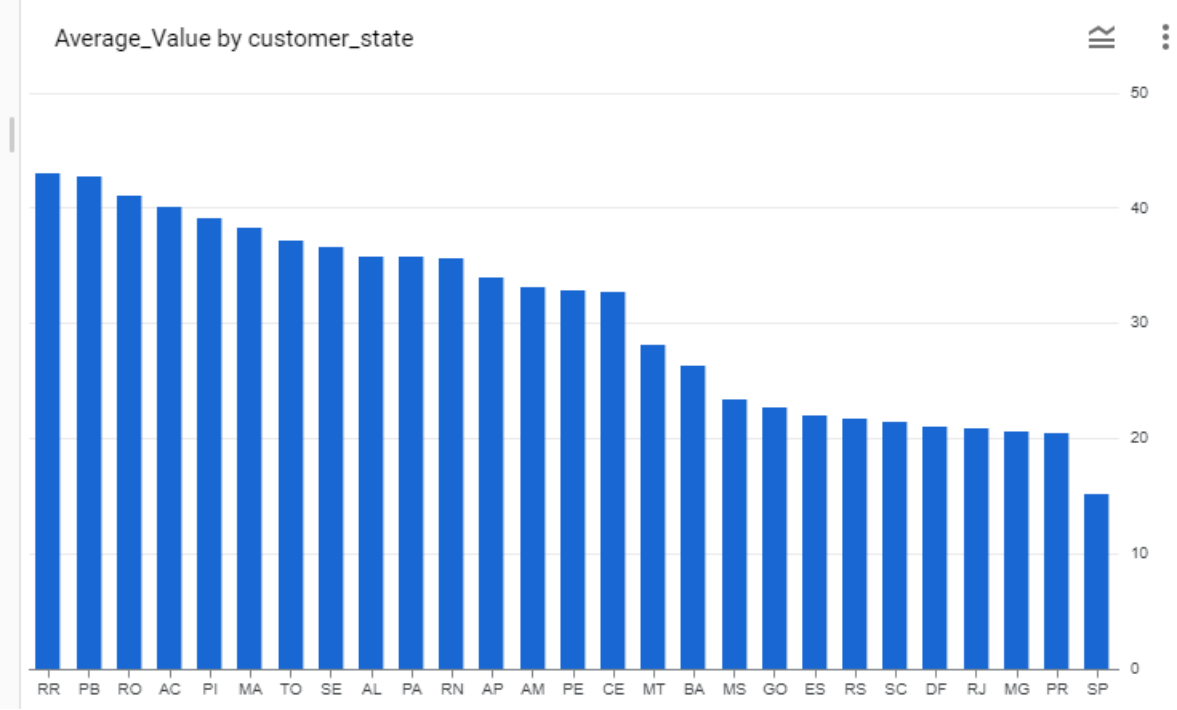
Insight:

The state of SP has the highest total value, whereas the state of PR has the highest average.

Total value by state:



Average value by state:



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

Query:

```
select order_id,
date_diff(extract(date from order_delivered_customer_date),
extract(date from order_purchase_timestamp),Day)
time_to_deliver,
date_diff(extract(date from
order_estimated_delivery_date),extract(date from
order_delivered_customer_date), Day) diff_estimated_delivery
from target_sql.orders
where order_delivered_customer_date is not null
and lower(order_status) in ('delivered')
order by order_id, customer_id
```

Screenshot:

Row	order_id	time_to_deliver	diff_estimated_delive
1	00010242fe8c5a6d1ba2dd792...	7	9
2	00018f77f2f0320c557190d7a1...	16	3
3	000229ec398224ef6ca0657da...	8	14
4	00024acbcdf0a6daa1e931b03...	6	6
5	00042b26cf59d7ce69dfabb4e...	25	16
6	00048cc3ae777c65dbb7d2a06...	7	15
7	00054e8431b9d7675808bcb8...	8	17
8	000576fe39319847cbb9d288c...	5	16
9	0005a1a1728c9d785b8e2b08...	10	0
10	0005f50442cb953dcd1d21e1f...	2	19

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

Highest Average Freight Value:

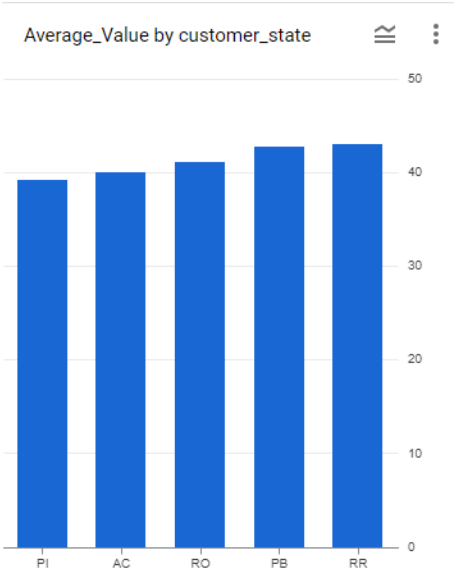
Query:

```
select * from(
select distinct c.customer_state,
round(avg(freight_value) over(partition by
c.customer_state),2) Average_Value,
FROM
target_sql.order_items p
inner join target_sql.orders s
on p.order_id = s.order_id
inner join target_sql.customers c
on s.customer_id = c.customer_id
order by 2 desc
limit 5
) order by 2
```

Result:

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

Insight: The state of RR has the highest average freight value followed by PB, RO, AC, and PI respectively.



Lowest Average Freight Value:

Query:

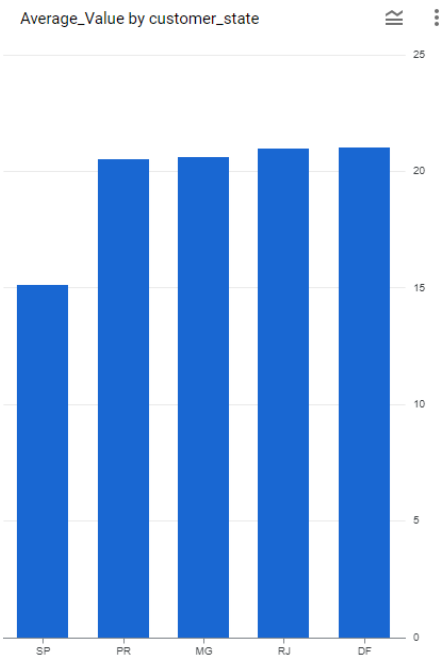
```
select distinct c.customer_state,
round(avg(freight_value) over(partition by
c.customer_state),2) Average_Value,
FROM
target_sql.order_items p
inner join target_sql.orders s
on p.order_id = s.order_id
inner join target_sql.customers c
on s.customer_id = c.customer_id
order by 2
limit 5
```

Result :

Row	customer_state	Average_Value
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04

customer_state

Insight: The state of SP has the lowest average freight value followed by PR,MG,RJ, and DF respectively.



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

Highest Average delivery time:

Query:

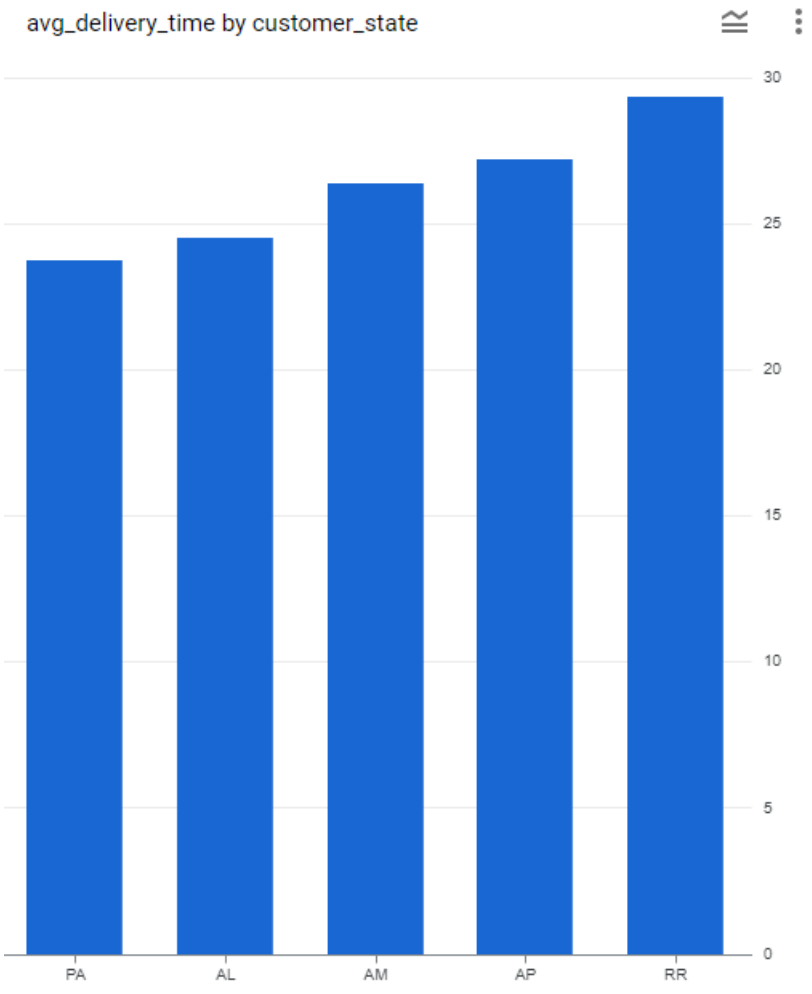
```
select * from (
with my_cte as (
select
order_id,o.customer_id,order_purchase_timestamp,c.customer_sta
te,
date_diff(extract(date from order_delivered_customer_date),
extract(date from order_purchase_timestamp),Day)
time_to_deliver,
#date_diff(extract(date from
order_estimated_delivery_date),extract(date from
order_delivered_customer_date), Day) diff_estimated_delivery
from target_sql.orders o
inner join target_sql.customers c
on o.customer_id = c.customer_id
where lower(order_status) in ('delivered')
) select distinct customer_state, round(avg(time_to_deliver)
over(partition by customer_state),2) avg_delivery_time from
my_cte
order by avg_delivery_time desc limit 5
) order by avg_delivery_time
```

Results :

customer_state	avg_delivery_time
PA	23.73
AL	24.5
AM	26.36
AP	27.18
RR	29.34

Insight :

The State of RR has the highest average delivery time followed by AP, AM, AL, and PA respectively.



Lowest Average delivery time:

Query:

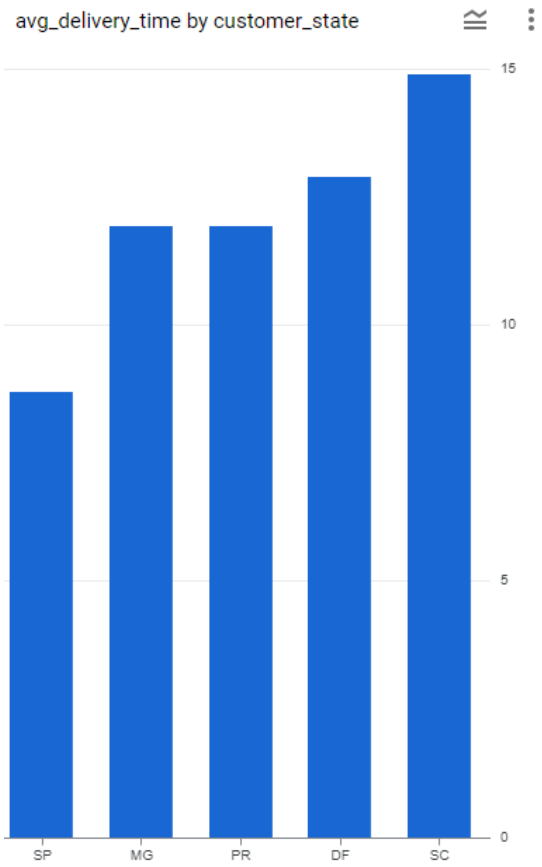
```
with my_cte as (
select
order_id,o.customer_id,order_purchase_timestamp,c.customer_sta
te,
date_diff(extract(date from order_delivered_customer_date),
extract(date from order_purchase_timestamp),Day)
time_to_deliver,
#date_diff(extract(date from
order_estimated_delivery_date),extract(date from
order_delivered_customer_date), Day) diff_estimated_delivery
from target_sql.orders o
inner join target_sql.customers c
on o.customer_id = c.customer_id
where lower(order_status) in ('delivered')
) select distinct customer_state, round(avg(time_to_deliver)
over(partition by customer_state),2) avg_delivery_time from
my_cte
order by avg_delivery_time limit 5
```

Result :

Row	customer_state	avg_delivery_time
1	SP	8.7
2	MG	11.94
3	PR	11.94
4	DF	12.9
5	SC	14.9

Insight :

The State of SP has the lowest average delivery time, followed by the states of MG, PR, DF, and SC respectively.



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.

Query:

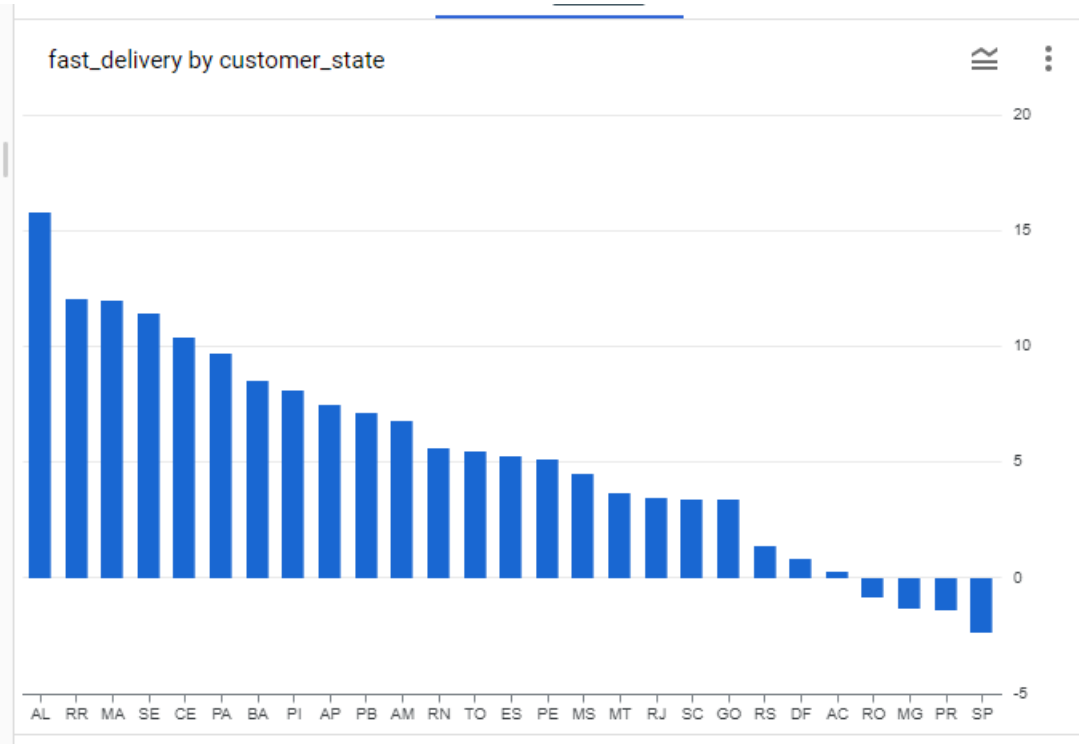
```
select customer_state, Round((avg_delivery_time-
avg_est_delivery_time),2) fast_delivery
from (
with my_cte as (
select
order_id,o.customer_id,order_purchase_timestamp,c.customer_sta
te,
date_diff(extract(date from order_delivered_customer_date),
extract(date from order_purchase_timestamp),Day)
time_to_deliver,
date_diff(extract(date from
order_estimated_delivery_date),extract(date from
order_delivered_customer_date), Day) diff_estimated_delivery
from target_sql.orders o
inner join target_sql.customers c
on o.customer_id = c.customer_id
where lower(order_status) in ('delivered')
```

```
)select distinct customer_state,
round(avg(time_to_deliver) over(partition by
customer_state),2) avg_delivery_time,
round(avg(diff_estimated_delivery) over(partition by
customer_state),2) avg_est_delivery_time
from my_cte
)
order by fast_delivery desc limit 5
```

Results:

Row	customer_state	fast_delivery
1	AL	15.79
2	RR	12.05
3	MA	11.94
4	SE	11.44
5	CE	10.4

Insight: The states of AL, RR, MA, SE, and CE have the fastest delivery rate among all the states.



6. **Analysis based on the payments:**
    1. Find the month on month no. of orders placed using different payment types.

Query:

```
with my_cte as (
select p.payment_type,
o.order_purchase_timestamp,extract(month from
o.order_purchase_timestamp) order_month from
target_sql.payments p
inner join target_sql.orders o
on p.order_id = o.order_id
) select order_month,payment_type,count(*) no_of_orders
 from my_cte
group by order_month,payment_type
order by order_month,payment_type
```

Screenshot:

Row	order_month	payment_type	no_of_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
11	3	debit_card	109
12	3	voucher	591
13	4	UPI	1783
14	4	credit_card	7301
15	4	debit_card	124

Insight:

Credit Card transactions have been the go to type of payment methods for the customers over the years, followed by UPI and then the Vouchers.

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

Query:

```
select count(*) no_of_orders from target_sql.payments
where payment_installments>1
and payment_sequential >= 1
```

Results:

no_of_orders
51338

Insight:

In this scenario, payment_installments column provides us with the number of instalments used for a particular purchase and the payment_sequential column provides us with the information of the number of instalments paid so far.

Hence, payment_installments value should be higher than 1 and payment_sequential should be at least 1, 1 means at least one EMI is paid for that particular order.