Target SQL Business Case

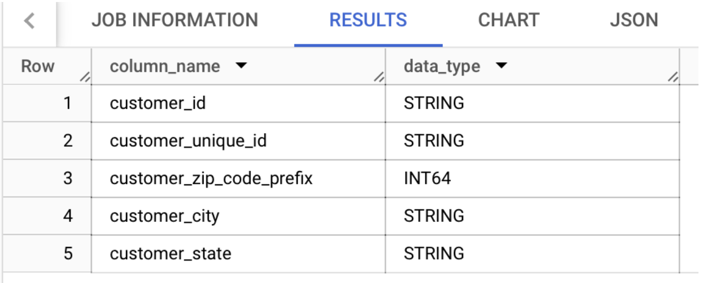
❖ Topic: SQL

❖ Jahnavi Yerroju

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.

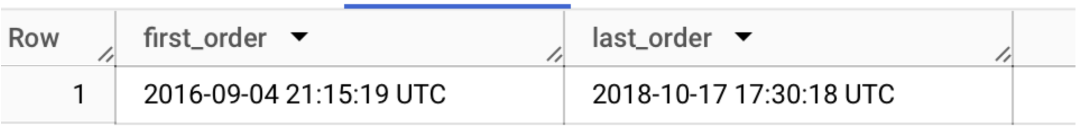
Query:
```
select column_name, data_type
      from target.INFORMATION_SCHEMA.COLUMNS
      where table_name = 'customers';
```



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

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

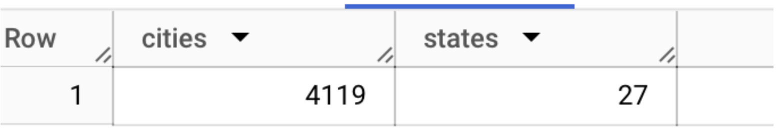
Query:
```
select min(order_purchase_timestamp) as first_order,
      max(order_purchase_timestamp) as last_order from `target.orders`;
```



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

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

Query:
```
select count (distinct c.customer_city) as cities,
      count (distinct c.customer_state) as states
      from `target.customers` c
      join `target.orders` o on c.customer_id = o.customer_id;
```



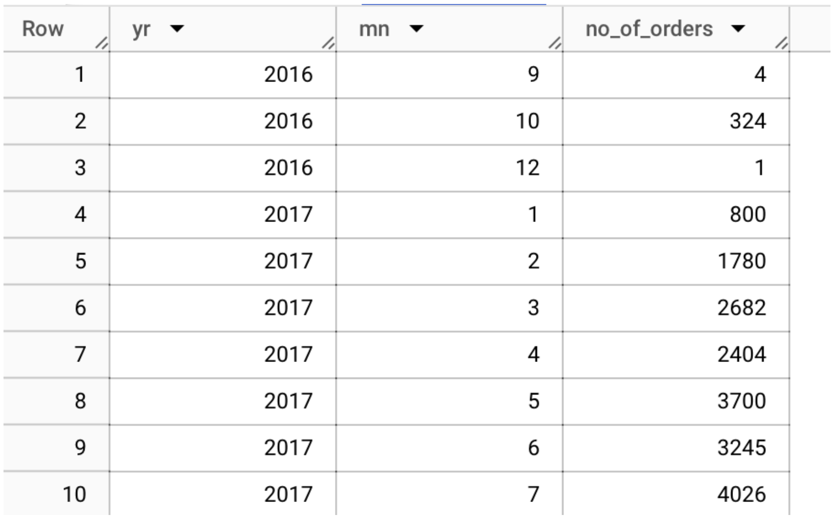
Row	cities	states
1	4119	27

II. In-depth Exploration:

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

- In the year 2017 growing trend is observed as the no.of orders increase month on month in most of the cases.

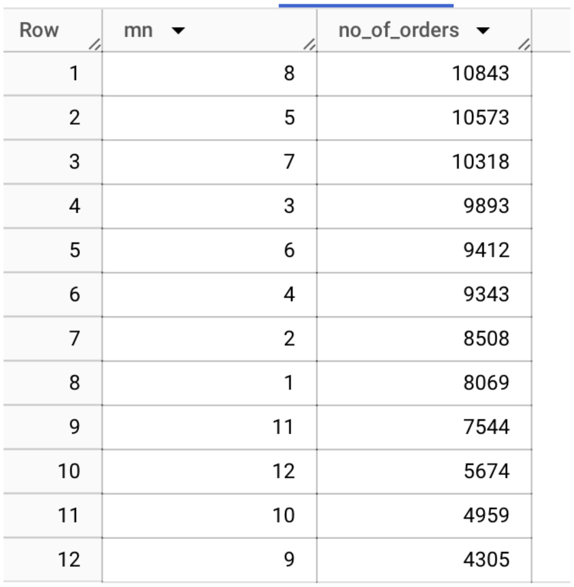
Query:
```
Select extract (year from order_purchase_timestamp) as yr,
extract (month from order_purchase_timestamp) as mn,
count(order_id) as no_of_orders
from `target.orders`
group by 1,2
order by 1,2
```



Row	yr	mn	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

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 mn,
count(order_id) as no_of_orders
from `target.orders`
group by 1
order by 2 desc
```

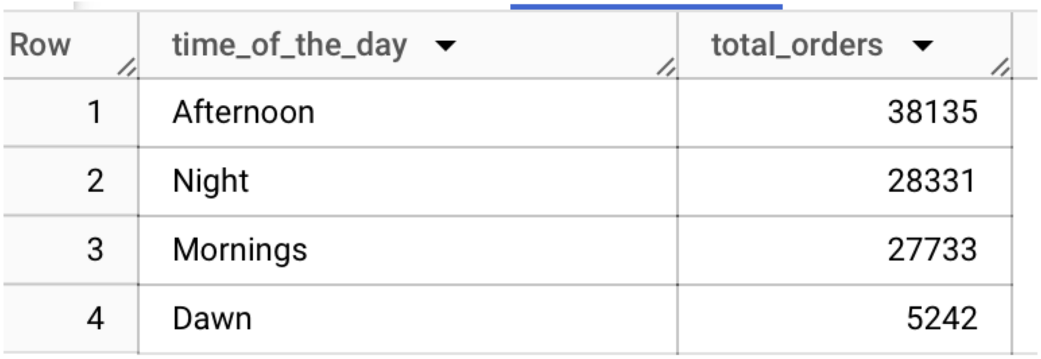


Row	mn	no_of_orders
1	8	10843
2	5	10573
3	7	10318
4	3	9893
5	6	9412
6	4	9343
7	2	8508
8	1	8069
9	11	7544
10	12	5674
11	10	4959
12	9	4305

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 order_purchase_timestamp)
               between 0 and 6 then'Dawn'
               when extract (hour from order_purchase_timestamp)
               between 7 and 12 then 'Mornings'
               when extract (hour from order_purchase_timestamp)
               between 13 and 18 then 'Afternoon'
               else 'Night'
       end as time_of_the_day,
       count (*) as total_orders
       from `target.orders`
       group by 1
       order by 2 desc
```

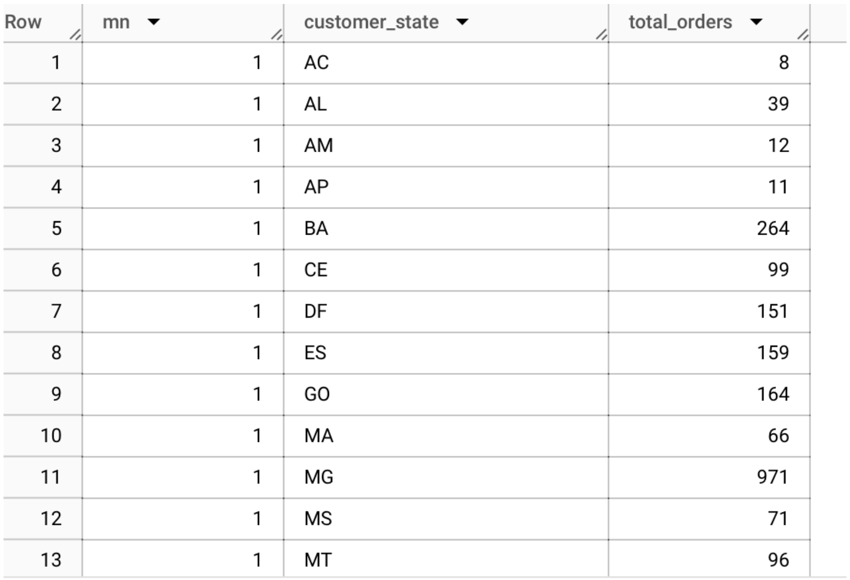


Row	time_of_the_day	total_orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

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

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

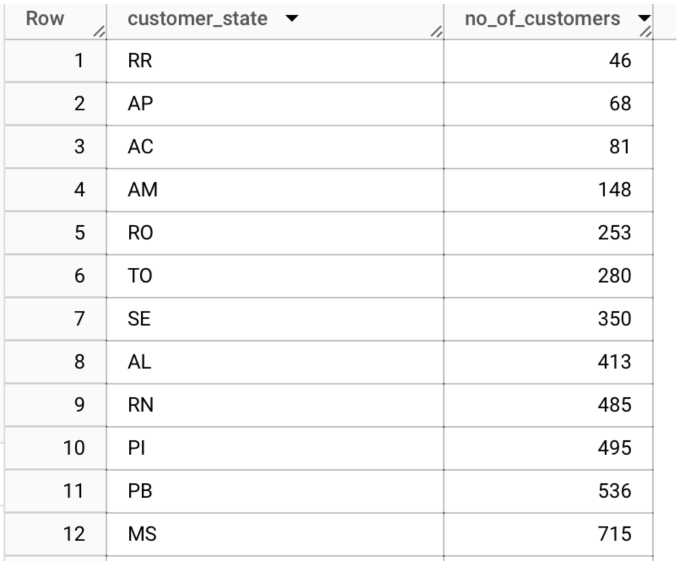
Query:
```
select
extract(month from o.order_purchase_timestamp)as mn,
c.customer_state,
count(*) as total_orders
from  `target.customers` c
join  `target.orders` o on o.customer_id = c.customer_id
group by
1,2
order by
 1,2;
```



Row	mn	customer_state	total_orders
1	1	AC	8
2	1	AL	39
3	1	AM	12
4	1	AP	11
5	1	BA	264
6	1	CE	99
7	1	DF	151
8	1	ES	159
9	1	GO	164
10	1	MA	66
11	1	MG	971
12	1	MS	71
13	1	MT	96

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

Query:
```
select customer_state, count(distinct customer_id) as no_of_customers
from `target.customers`
group by customer_state
order by 2;
```



Row	customer_state	no_of_customers
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
12	MS	715

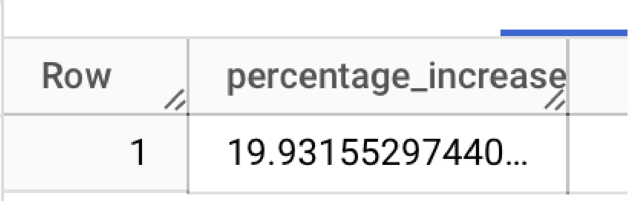
IV. Impact on Economy: Analyze the money movement by e-commerce by looking at

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).

- I'm not sure if I need to select the entire year of 2017 or just the months from January to August. So, I wrote query in both the cases

2017(1-12) – 2018(1-8)

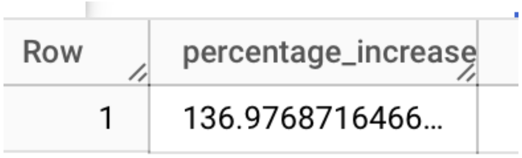
Query:
```
with final as (Select
        extract(year from o.order_purchase_timestamp) as year,

    sum(p.payment_value) as cost
        from `target.orders` o
        join `target.payments` p on o.order_id = p.order_id
        where
   (extract(year from o.order_purchase_timestamp) = 2017) or
   (extract(year from o.order_purchase_timestamp) = 2018 and
    extract(month from o.order_purchase_timestamp) between 1 and 8)
        group by 1)
        select
        (sum(case when year = 2018 then cost else 0 end) -
        sum(case when year = 2017 then cost else 0 end)) /
        (sum(case when year = 2017 then cost else 0 end)) * 100 as percentage_increase
        from final;
```



Row	percentage_increase
1	19.93155297440...

2017(1-8) – 2018(1-8)

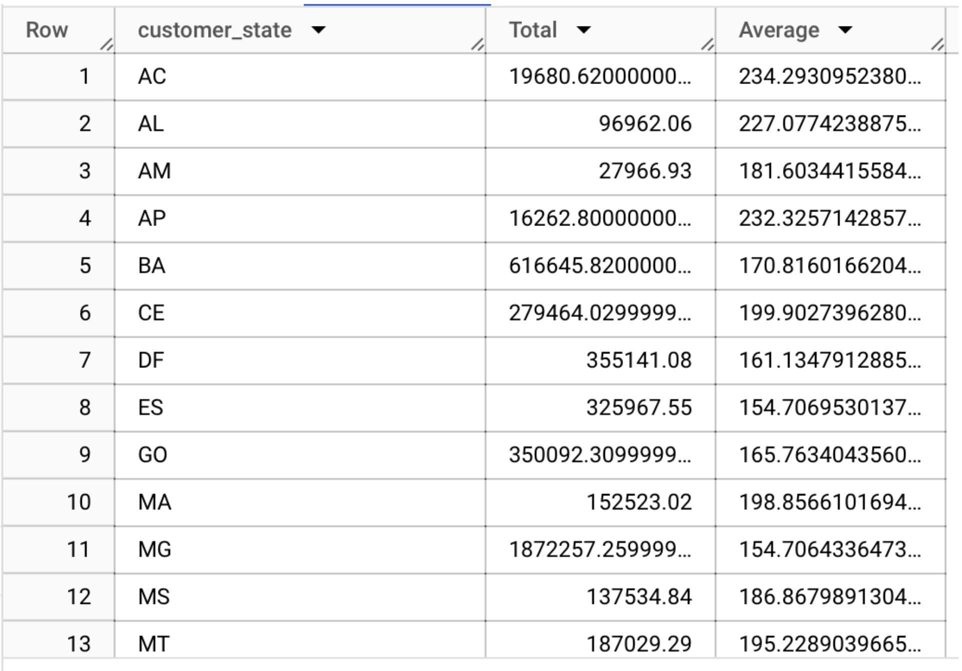
```
Query: with final as (
        select extract(year from o.order_purchase_timestamp) as year,
        sum(p.payment_value) as cost
        from `target.orders` o
        join `target.payments` p
        on p.order_id = o.order_id
        where extract(year from o.order_purchase_timestamp) in (2017,2018)
        and extract(month from o.order_purchase_timestamp) between 1 and 8
        group by 1
        )

        select
        (sum(case when year = 2018 then cost else 0 end) -
        sum(case when year = 2017 then cost else 0 end)) /
        (sum(case when year = 2017 then cost else 0 end)) * 100 as percentage_increase
        from final;
```



Row	percentage_increase
1	136.9768716466...

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

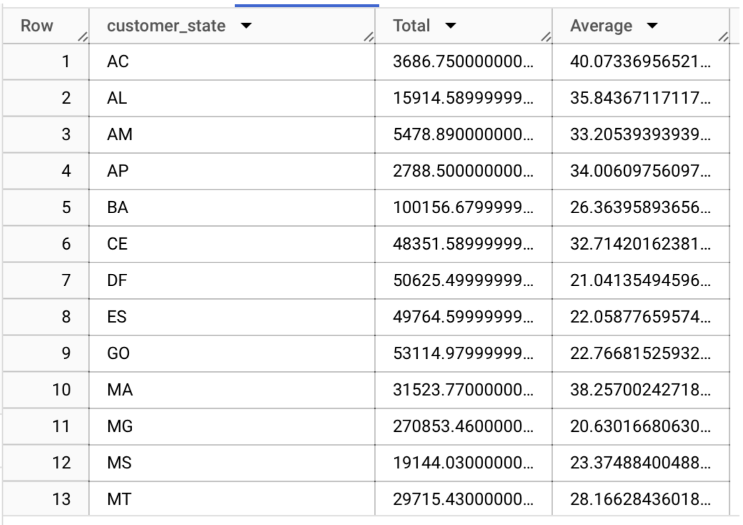
```
Query: select c.customer_state, sum(p.payment_value) as Total,
        avg(p.payment_value) as Average
        from `target.payments` p
        join `target.orders` o on p.order_id = o.order_id
        join `target.customers` c on c.customer_id = o.customer_id
        group by 1
        order by 1;
```



Row	customer_state	Total	Average
1	AC	19680.62000000...	234.2930952380...
2	AL	96962.06	227.0774238875...
3	AM	27966.93	181.6034415584...
4	AP	16262.80000000...	232.3257142857...
5	BA	616645.8200000...	170.8160166204...
6	CE	279464.0299999...	199.9027396280...
7	DF	355141.08	161.1347912885...
8	ES	325967.55	154.7069530137...
9	GO	350092.3099999...	165.7634043560...
10	MA	152523.02	198.8566101694...
11	MG	1872257.259999...	154.7064336473...
12	MS	137534.84	186.8679891304...
13	MT	187029.29	195.2289039665...

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

Query:
```
select c.customer_state, sum(o1.freight_value) as Total,
avg(o1.freight_value) as Average
from `target.order_items` o1
join `target.orders` o on o1.order_id = o.order_id
join `target.customers` c on c.customer_id = o.customer_id
group by 1
order by 1;
```



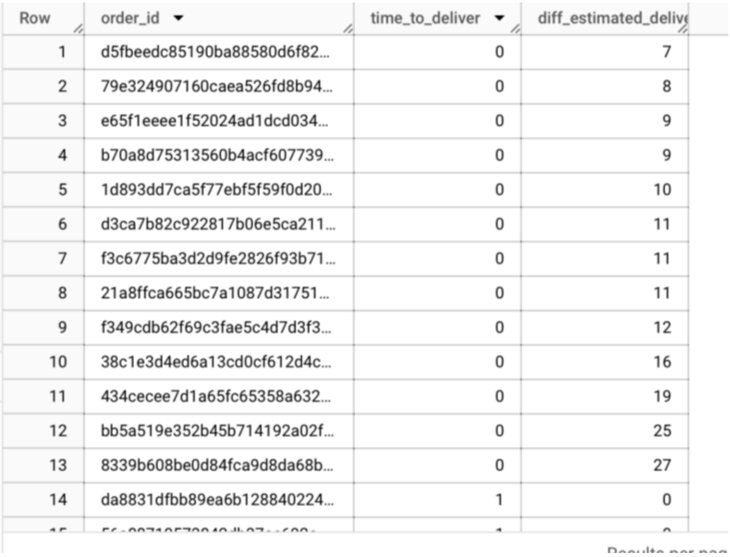
Row	customer_state	Total	Average
1	AC	3686.750000000...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.890000000...	33.20539393939...
4	AP	2788.500000000...	34.00609756097...
5	BA	100156.6799999...	26.36395893656...
6	CE	48351.58999999...	32.71420162381...
7	DF	50625.49999999...	21.04135494596...
8	ES	49764.59999999...	22.05877659574...
9	GO	53114.97999999...	22.76681525932...
10	MA	31523.77000000...	38.25700242718...
11	MG	270853.4600000...	20.63016680630...
12	MS	19144.03000000...	23.37488400488...
13	MT	29715.43000000...	28.16628436018...

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.

Query:
```
select o.order_id,date_diff(o.order_delivered_customer_date,
        o.order_purchase_timestamp, day) as time_to_deliver,
        date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day) as
        diff_estimated_delivery
        from `target.orders` o
        where o.order_status = 'delivered'
        and o.order_delivered_customer_date is not NULL
        and o.order_estimated_delivery_date is not NULL
        order by 2,3;
```

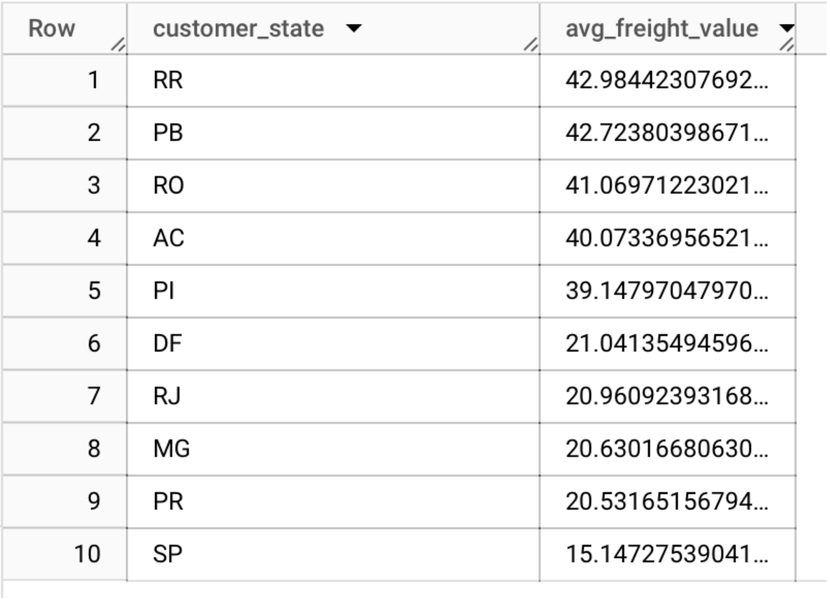


Row	order_id	time_to_deliver	diff_estimated_delive
1	d5fbeedc85190ba88580d6f82...	0	7
2	79e324907160caea526fd8b94...	0	8
3	e65f1eeee1f52024ad1dcd034...	0	9
4	b70a8d75313560b4acf607739...	0	9
5	1d893dd7ca5f77ebf5f59f0d20...	0	10
6	d3ca7b82c922817b06e5ca211...	0	11
7	f3c6775ba3d2d9fe2826f93b71...	0	11
8	21a8ffca665bc7a1087d31751...	0	11
9	f349cdb62f69c3fae5c4d7d3f3...	0	12
10	38c1e3d4ed6a13cd0cf612d4c...	0	16
11	434cecee7d1a65fc65358a632...	0	19
12	bb5a519e352b45b714192a02f...	0	25
13	8339b608be0d84fca9d8da68b...	0	27
14	da8831dfbb89ea6b128840224...	1	0
15	[illegible]	1	0

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

Query:
```
with final as (
        select c.customer_state,
        avg(o1.freight_value) as avg_freight_value
        from `target.order_items` o1
        join `target.orders` o on o1.order_id = o.order_id
        join `target.customers` c on c.customer_id = o.customer_id
        group by 1
        order by 2
        )

        (select customer_state,avg_freight_value
        from final
        order by avg_freight_value desc
        limit 5)
```

```
union all

(select customer_state,avg_freight_value
from final
order by avg_freight_value
limit 5)
order by avg_freight_value desc
```



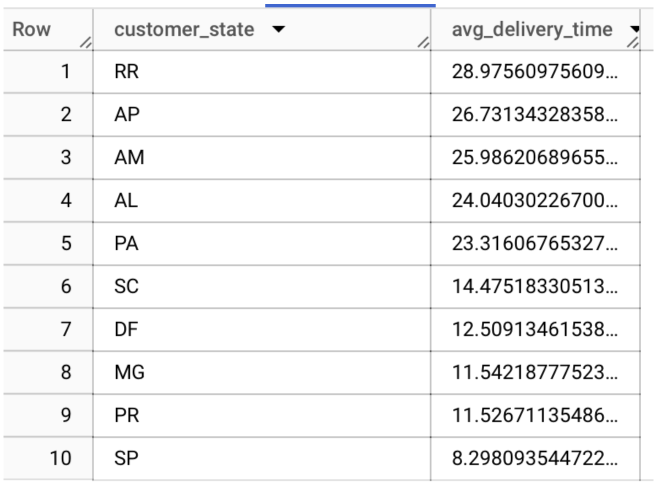
Row	customer_state	avg_freight_value
1	RR	42.98442307692…
2	PB	42.72380398671…
3	RO	41.06971223021…
4	AC	40.07336956521…
5	PI	39.14797047970…
6	DF	21.04135494596…
7	RJ	20.96092393168…
8	MG	20.63016680630…
9	PR	20.53165156794…
10	SP	15.14727539041…

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

Query:

```
with final as (
select  c.customer_state,
avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp, day)) as
avg_delivery_time
from `target.orders` o
join `target.customers` c on o.customer_id = c.customer_id
where o.order_status = 'delivered'
and o.order_delivered_customer_date is not null
group by c.customer_state
)

(select customer_state,avg_delivery_time
from final
order by avg_delivery_time desc
limit 5)

union all
```

```
(select customer_state,avg_delivery_time
from final
order by avg_delivery_time
limit 5)

order by avg_delivery_time desc;
```

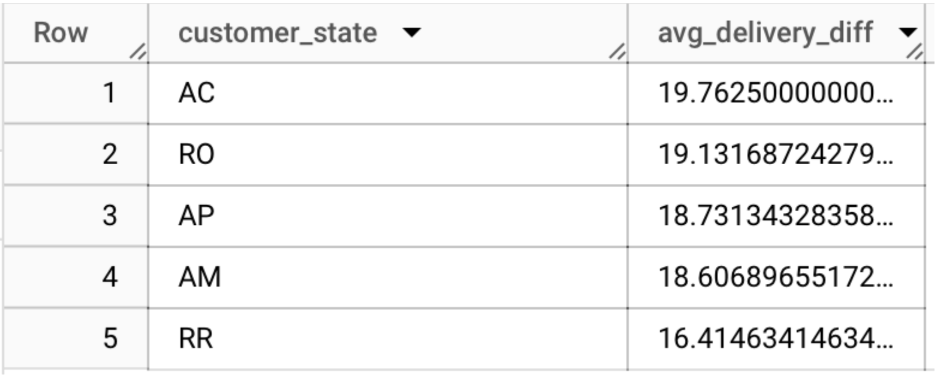


Row	customer_state	avg_delivery_time
1	RR	28.97560975609...
2	AP	26.73134328358...
3	AM	25.98620689655...
4	AL	24.04030226700...
5	PA	23.31606765327...
6	SC	14.47518330513...
7	DF	12.50913461538...
8	MG	11.54218777523...
9	PR	11.52671135486...
10	SP	8.298093544722...

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:
```
select c.customer_state,
avg(date_diff(o.order_estimated_delivery_date,
o.order_delivered_customer_date, day))
as avg_delivery_diff
from
`target.orders` o
join `target.customers` c ON o.customer_id = c.customer_id
where o.order_status = 'delivered'
group by 1
order by 2 desc
limit 5;
```

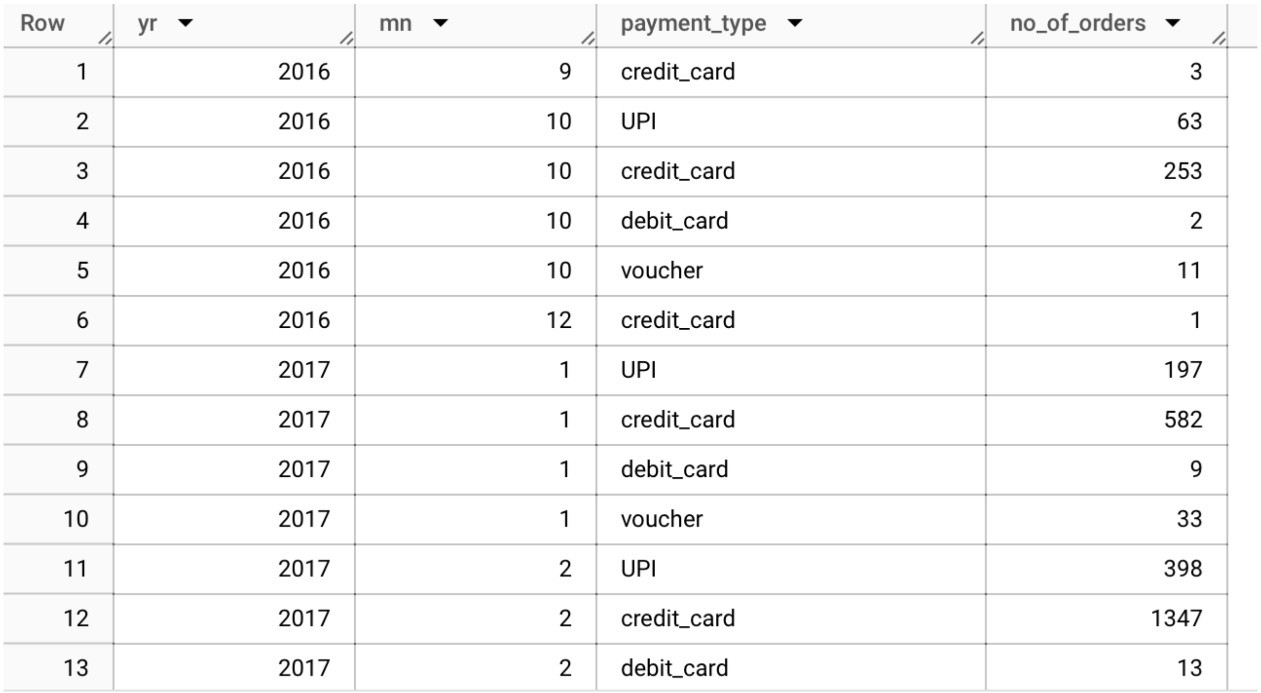


Row	customer_state	avg_delivery_diff
1	AC	19.76250000000...
2	RO	19.13168724279...
3	AP	18.73134328358...
4	AM	18.60689655172...
5	RR	16.41463414634...

VI. Analysis based on the payments:

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

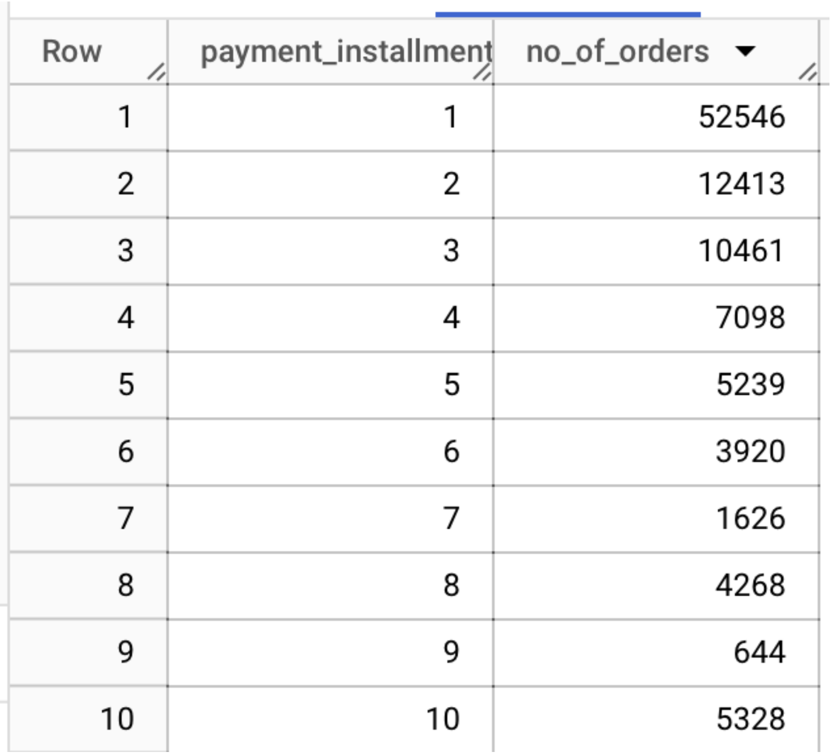
Query:
```
select extract(year from o.order_purchase_timestamp) as yr,
extract(month from o.order_purchase_timestamp) as mn,
p.payment_type,
count(distinct o.order_id) as no_of_orders
from `target.orders` o
join `target.payments` p on o.order_id = p.order_id
group by 1,2,3
order by 1,2,3;
```



Row	yr	mn	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
12	2017	2	credit_card	1347
13	2017	2	debit_card	13

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

```
Query: select payment_installments,
       count(order_id) as no_of_orders
       from `target.payments`
       where payment_installments>=1
       group by 1 order by 1, 2;
```



Row	payment_installment	no_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	5	5239
6	6	3920
7	7	1626
8	8	4268
9	9	644
10	10	5328