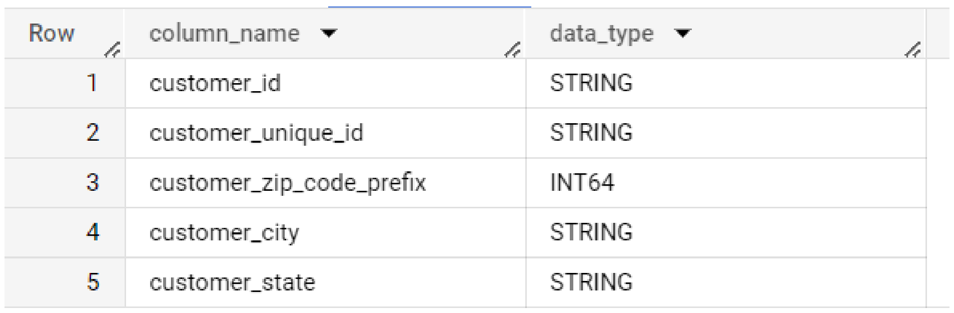
Business Case Study: Target SQL

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.

```
select column_name, data_type
 from `CaseStudy.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.

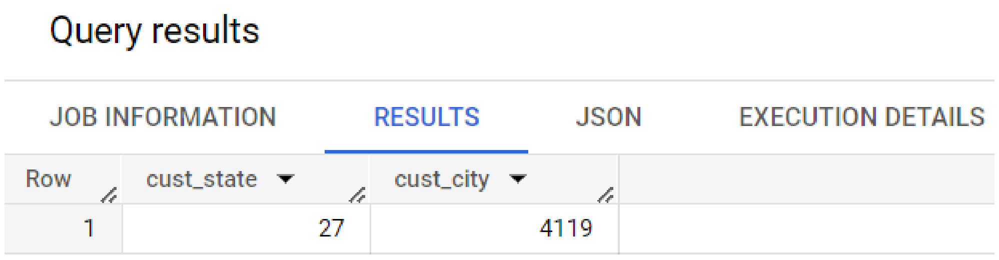
```
select min(order_purchase_timestamp) as first_time_stamp,

max(order_purchase_timestamp) as last_time_Stamp
from `CaseStudy.orders`;
```

So, here I used the min and max function to get the first and last date for which the order was placed.

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

```
select count(distinct customer_state) as cust_state ,
count(distinct customer_city) as cust_city
from `CaseStudy.orders` o inner join `CaseStudy.customers` c
on o.customer_id = c.customer_id;
```


Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	cust_state	cust_city
1	27	4119

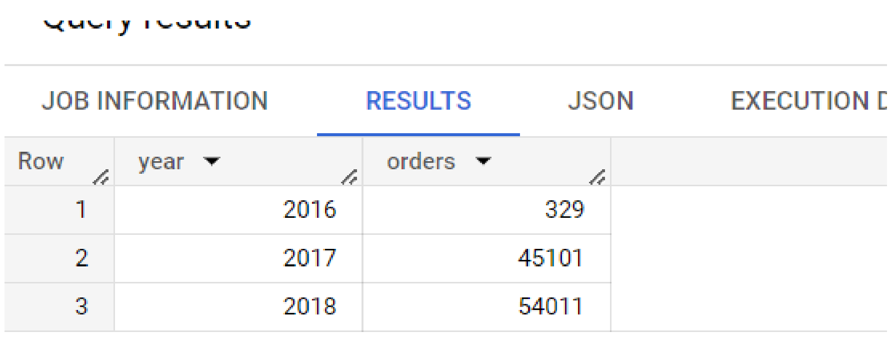
To find orders during the given period we have to join to tables and we don't want to use where clause so that's we use inner join. We use count function to count the number of state and city and also use distinct for state and city because we want unique states and cities. The customer_id is common column in both the table so we use that to join the both tables.

2. In-depth Exploration:

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

```
 select extract(year from order_purchase_timestamp) as year,

count(order_id) as orders
from `CaseStudy.orders`
group by 1
order by 1;
```



JOB INFORMATION | RESULTS | JSON | EXECUTION D

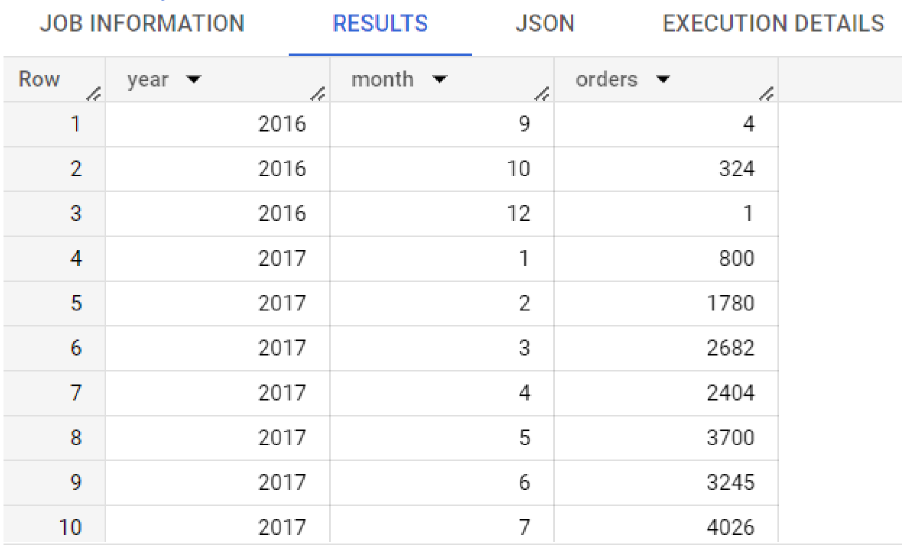
Row	year	orders
1	2016	329
2	2017	45101
3	2018	54011

we extract year timestamp and then group by the data. Then we count the order_id to get the no of orders placed during the year and also order by year in ascending to see data year by year.

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

```
select extract(year from order_purchase_timestamp) as year,

extract(month from order_purchase_timestamp) as month,
count(order_id) as orders
from `CaseStudy.orders`
group by 1,2
order by 1,2;
```



JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	year	month	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

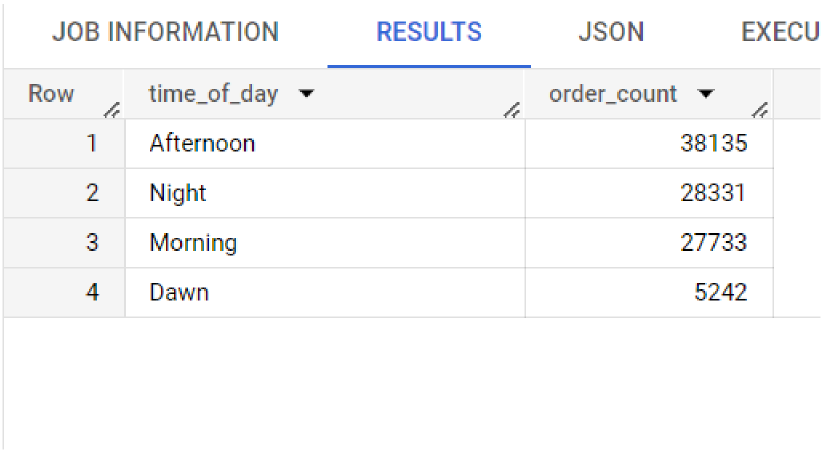
So, there is no seasonality in terms of data because the orders are increasing and decreasing every month. To check seasonality over the month, I used extract to extract year and month, then group by the data and also count the number of orders.

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'
    ELSE 'Unknown'
  END AS time_of_day,
  COUNT(*) AS order_count
FROM
  `CaseStudy.orders`
GROUP BY
  time_of_day
ORDER BY
  order_count DESC;
```



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

Afternoon (13 to 18 hours) has the highest number of orders. Firstly, I extract hour from timestamp and then divide the data according to question in 4 different categories. Then I group by the data according to categories and count the number of the orders. We use * in count because every order_id is corresponding to time_of_day so, that's why we used * in count.

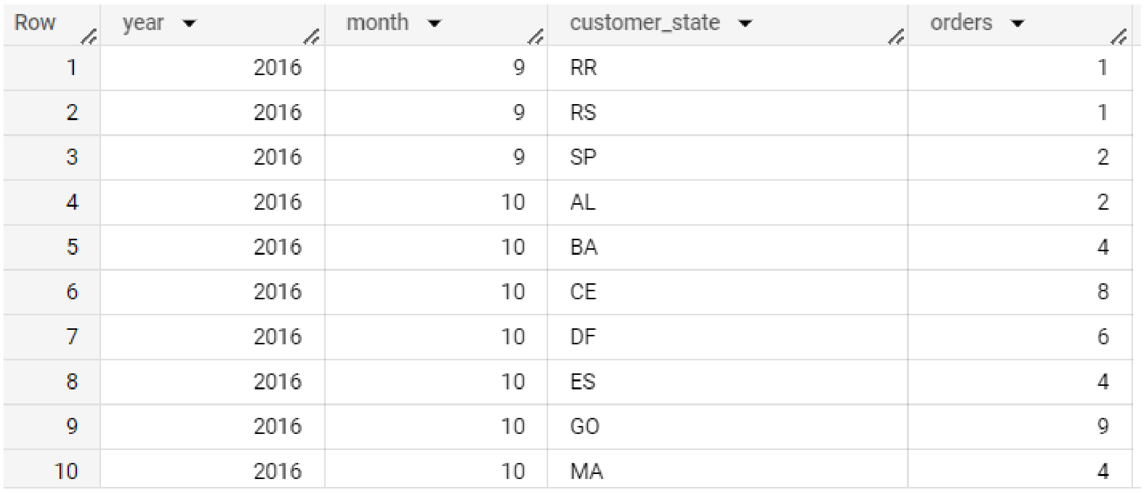
3. Evolution of E-commerce orders in the Brazil region:

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

```
 select extract(year from order_purchase_timestamp) as year,

extract(month from order_purchase_timestamp) as month,
customer_state,
count(order_id) as orders
from `CaseStudy.orders` o join
`CaseStudy.customers` c
on o.customer_id = c.customer_id
group by 1,2,3
```

```
order by 1,2,3;
```



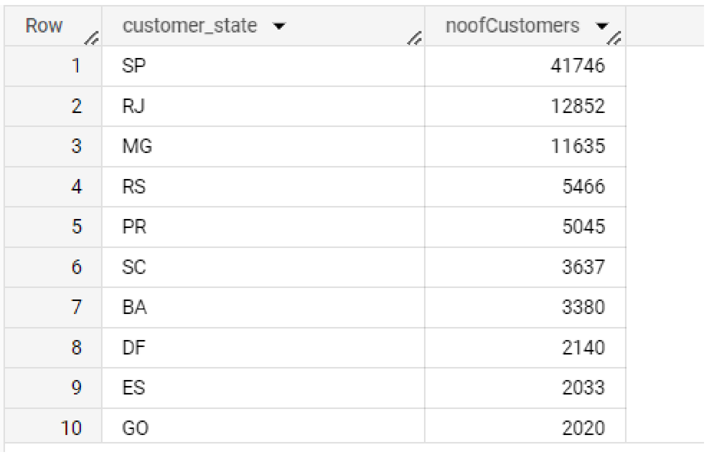
Row	year	month	customer_state	orders
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	AL	2
5	2016	10	BA	4
6	2016	10	CE	8
7	2016	10	DF	6
8	2016	10	ES	4
9	2016	10	GO	9
10	2016	10	MA	4

We inner join customers and orders table to get common rows between the two tables. We group by data by year, month and customer_state and use count function to count number of orders on each state month on month

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

```
select customer_state,

count(distinct customer_id) as noofCustomers
from `CaseStudy.customers`
group by customer_state
order by noofCustomers desc;
```



Row	customer_state	noofCustomers
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

I used group by function to group cutomer_state in customers table and then count number of distinct customer_id to get noofCustomers

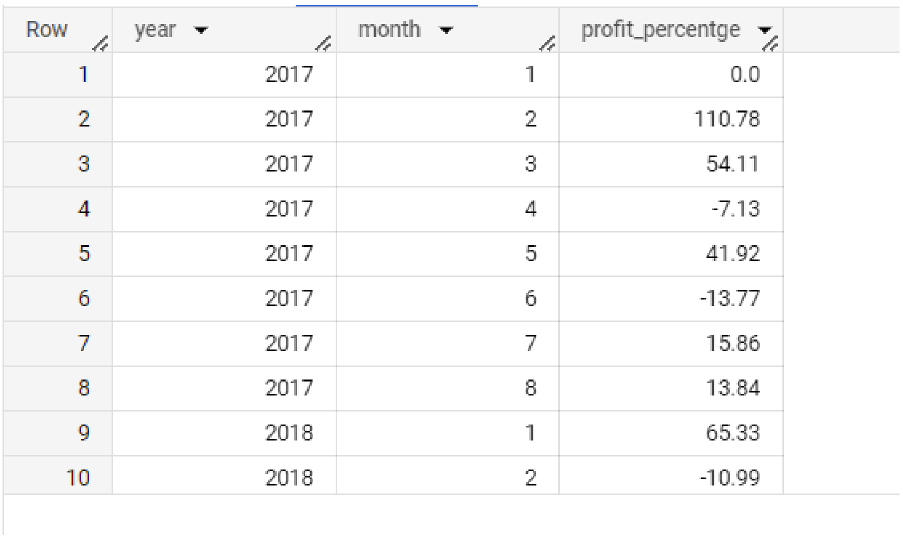
4. Impact on Economy: Analyse the money movement by e-commerce by looking at order prices, freight and others.

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.

```
select year, month,

ifnull(ROUND(((sum_of_payment - lag(sum_of_payment) over(order by
year,month))/lag(sum_of_payment) over(order by year,month))*100,2),0)
```

```
as profit_percentge
from
(
select
extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,
ROUND(sum(payment_value),1) as sum_of_payment
from `CaseStudy.orders` o join `CaseStudy.payments` p
on o.order_id = p.order_id
group by 1,2
)
where year between 2017 and 2018 and month between 1 and 8
order by 1,2;
```



Row	year	month	profit_percentge
1	2017	1	0.0
2	2017	2	110.78
3	2017	3	54.11
4	2017	4	-7.13
5	2017	5	41.92
6	2017	6	-13.77
7	2017	7	15.86
8	2017	8	13.84
9	2018	1	65.33
10	2018	2	-10.99

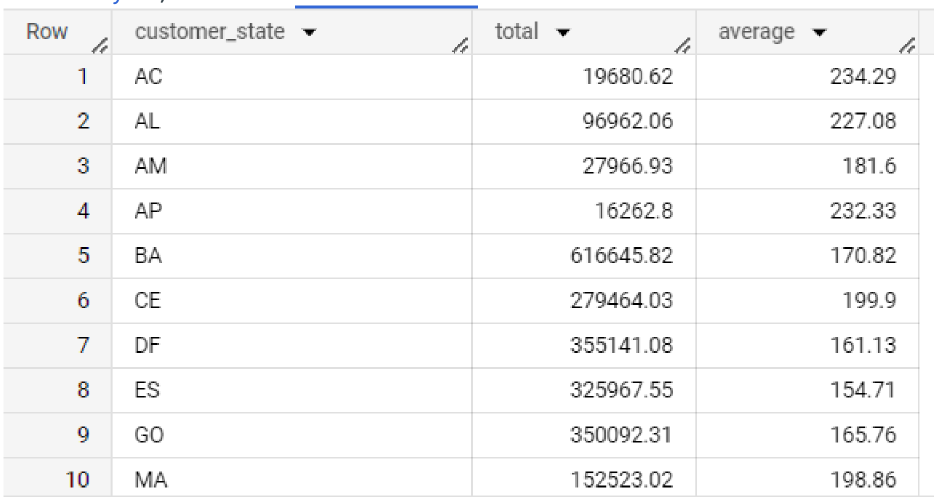
Here, I used two queries inner and outer. In inner query I inner join orders and payments table to all the common rows, then group by year and month. Use sum function to get sum of payment om that particular month and year and also round off the value
In the outer query I used where clause to get data between Jan and Aug for year 2017 and 2018. In the select statement we take year and month directly from table and we use window function lag to get value of previous month amount to get month on month increase or decrease in the payments. Also round off that value and use ifnull for values that are null it will replace it with 0.
To get % = ((New value -old value)/old value) *100)

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

```
 select distinct customer_state,

ROUND(sum(payment_value),2) as total,
ROUND(avg(payment_value),2) as average
from `CaseStudy.customers` c
join `CaseStudy.orders` o
on c.customer_id = o.customer_id
join `CaseStudy.payments` p
on o.order_id = p.order_id
group by 1
```

```
order by 1;
```



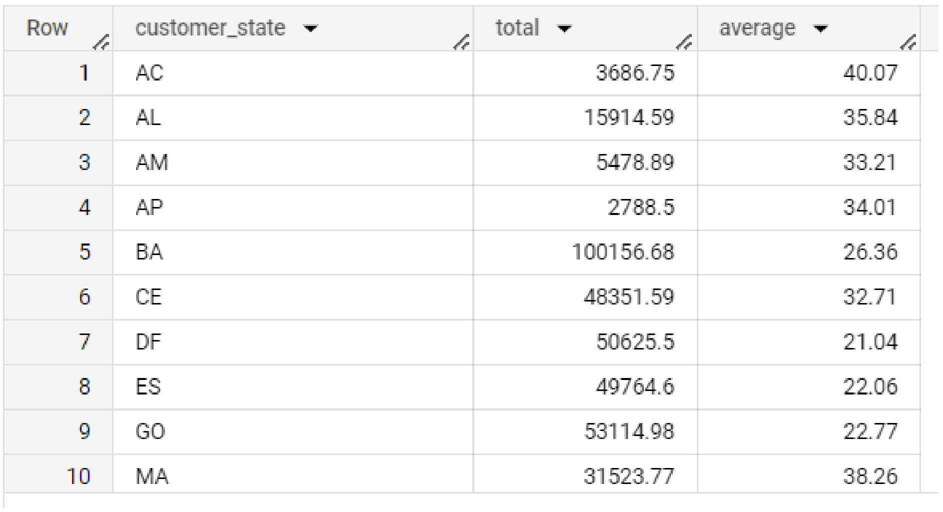
Row	customer_state	total	average
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86

Here, I use inner join to merge customers and orders and then join payments table because we cannot join the customers and payments table directly. Group by the data customer_state the use sum and avg function on payment_value to get total and average amount of payment for every state

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

```
select distinct customer_state,

ROUND(sum(freight_value),2) as total,
ROUND(avg(freight_value),2) as average
from `CaseStudy.customers` c
join `CaseStudy.orders` o
on c.customer_id = o.customer_id
join `CaseStudy.order_items` ot
on o.order_id = ot.order_id
group by 1
order by 1;
```



Row	customer_state	total	average
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

```
Here, I use inner join to merge customers and orders table and then join orderitems
table because we cannot join the customes and orderitems table directly.Group by
the data customer_state the use sum and avg function on freight_value to get total
and average amount of freight value for every state.
```

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.

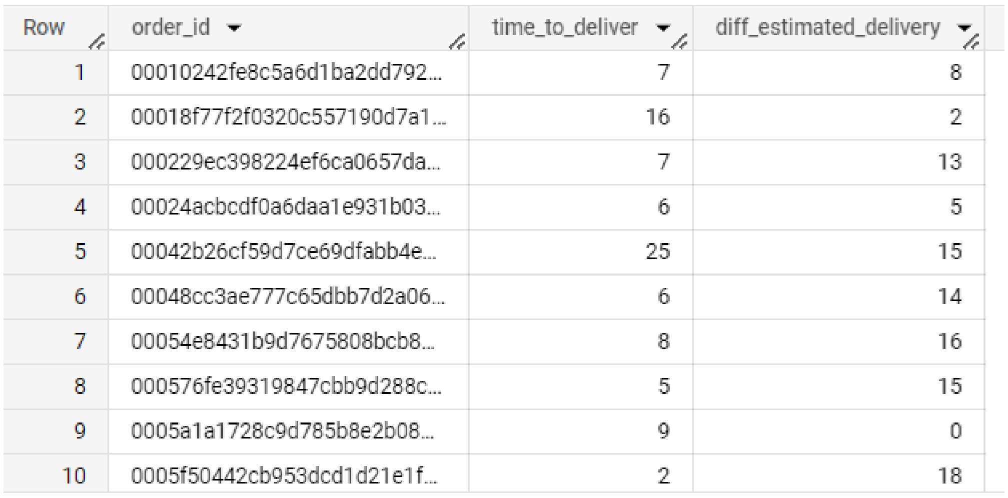
You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:

i. **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
ii. **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

```
select order_id,

time_to_deliver,diff_estimated_delivery
from(
  select
order_id,
timestamp_diff
(order_delivered_customer_date,order_purchase_timestamp, day)
as time_to_deliver,
timestamp_diff
(order_estimated_delivery_date,order_delivered_customer_date,
day) as diff_estimated_delivery ,
from `CaseStudy.orders`
where order_status = "delivered"

)
where diff_estimated_delivery is not null and time_to_deliver
is not null
order by 1,2,3;
```



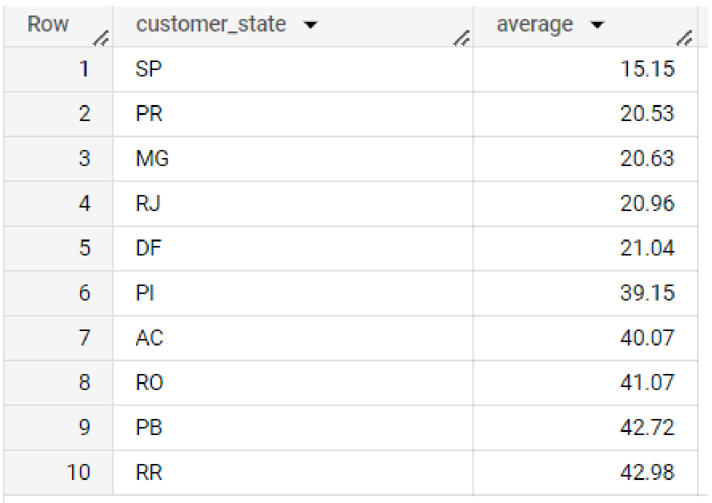
Row	order_id	time_to_deliver	diff_estimated_delivery
1	00010242fe8c5a6d1ba2dd792...	7	8
2	00018f77f2f0320c557190d7a1...	16	2
3	000229ec398224ef6ca0657da...	7	13
4	00024acbcdf0a6daa1e931b03...	6	5
5	00042b26cf59d7ce69dfabb4e...	25	15
6	00048cc3ae777c65dbb7d2a06...	6	14
7	00054e8431b9d7675808bcb8...	8	16
8	000576fe39319847cbb9d288c...	5	15
9	0005a1a1728c9d785b8e2b08...	9	0
10	0005f50442cb953dcd1d21e1f...	2	18

As in given question we used order delivered customer date, order purchase timestamp and order estimate delivery date to find time_to_deliver and estimated_delivery using timestamp_diff function. Also, filter out the data where order status is either delivered or canceled because in this case order is delivered and order delivered customer date is not null. Also, filtering out the data using where clause on the basis of order_status = “delivered”. Then use this as inner query for outer query in which we filter out the data if any of the 2 columns are not null for safety purpose and select all the column from inner query to display.

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

```
select customer_state,

t.average
from(
  select distinct customer_state,
ROUND(avg(freight_value),2) as average,
row_number() over(order by avg(freight_value)) as asc_count,
row_number() over(order by avg(freight_value) desc) as desc_count
from `CaseStudy.customers` c
join `CaseStudy.orders` o
on c.customer_id = o.customer_id
join `CaseStudy.order_items` ot
on o.order_id = ot.order_id
group by 1
) t
where asc_count between 1 and 5 or desc_count between 1 and 5
order by 2;
```

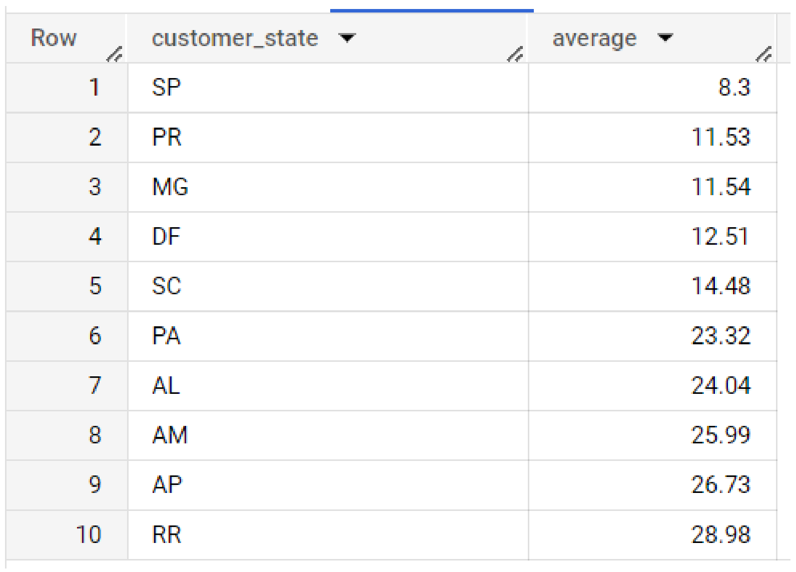


Row	customer_state	average
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04
6	PI	39.15
7	AC	40.07
8	RO	41.07
9	PB	42.72
10	RR	42.98

Here, we have two queries – one inner and one outer
In the inner query, I use inner join to merge customers and orders table and then join orderitems table because we cannot join the customes and orderitems table directly.Group by the data customer_state and use avg function on freight_value to get average amount of freight value for every state. Use window function row_number to lable all the states from ascending and descending order
In the outer query use where clause to get the value of top 5 lowest and highest state.

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

```
select customer_state, average
from(
  select distinct customer_state,
ROUND(avg(timestamp_diff(order_delivered_customer_date,order_purchase_
timestamp, day)),2) as average,
row_number() over(order by
ROUND(avg(timestamp_diff(order_delivered_customer_date,order_purchase_
timestamp, day)),2) ) as asc_count,
row_number() over(order by
ROUND(avg(timestamp_diff(order_delivered_customer_date,order_purchase_
timestamp, day)),2)  desc) as desc_count
from `CaseStudy.customers` c
join `CaseStudy.orders` o
on c.customer_id = o.customer_id
where order_status = "delivered"
group by 1
) t
where asc_count between 1 and 5 or desc_count between 1 and 5
order by 2;
```



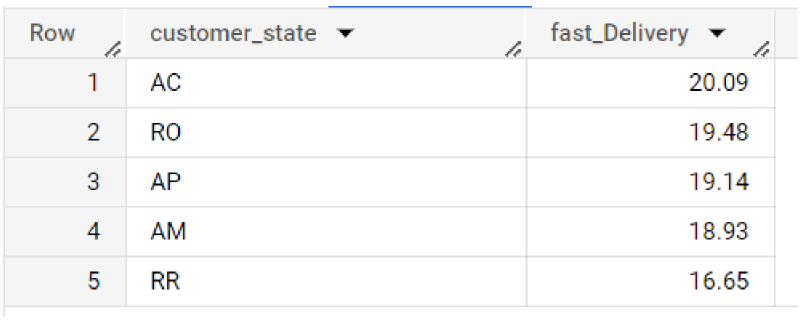
Row	customer_state	average
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98

Here, we have two queries – one inner and one outer
In the inner query, I use inner join to merge customers and orders table, then use group by customer_state and use avg function on timestamp_diff function which finds the diiference between order delivery customer date and order purchase timestamp to get average delivery time for every state. Use window function row_number to lable all the states from ascending and descending order. Also, filtering out the data using where clause on the basis of order_status = "delivered"
In the outer query use where clause to get the value of top 5 lowest and highest state and also display state name and average value in ascending order using select clause.

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.

```
select customer_state,

ROUND(Estimated_Avg - Acutual_Avg,2) as fast_Delivery
from(
  select distinct customer_state,
ROUND(avg(timestamp_diff(order_delivered_customer_date,order_purchase
_timestamp, day)),2) as Acutual_Avg,
ROUND(avg(timestamp_diff(order_estimated_delivery_date,order_purchase
_timestamp, day)),2) as Estimated_Avg
from `CaseStudy.customers` c
join `CaseStudy.orders` o
on c.customer_id = o.customer_id
where order_status = "delivered"
group by 1
) t
order by 2 desc
LIMIT 5;
```



Row	customer_state	fast_Delivery
1	AC	20.09
2	RO	19.48
3	AP	19.14
4	AM	18.93
5	RR	16.65

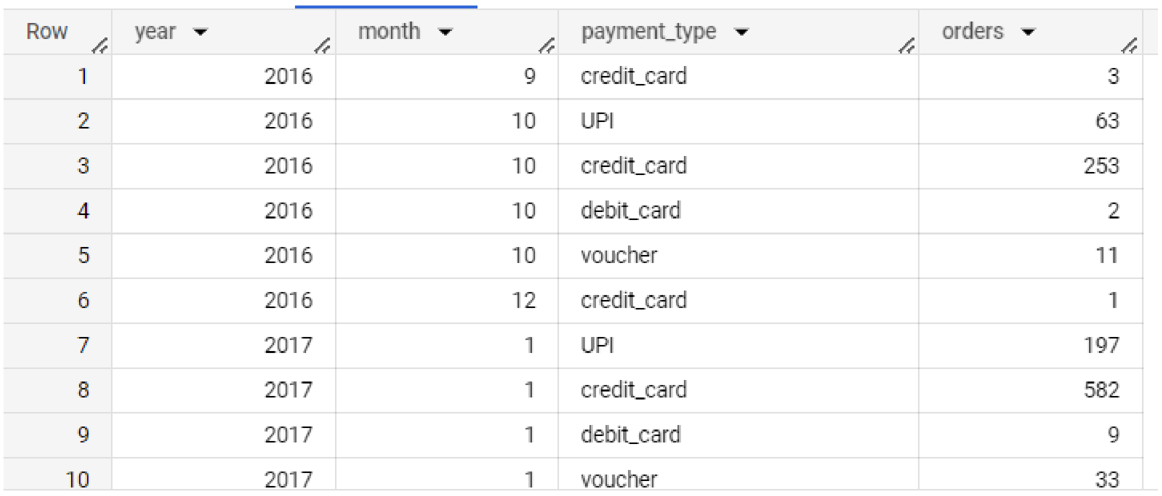
Here, we have two queries – one inner and one outer
In the inner query, I use inner join to merge customers and orders table, then use group by customer_state and use avg function on timestamp_diff function which finds the diiference between order delivery customer date and order purchase timestamp to get Actual_Avg for every state and similarly finds the Estinated Avg using order purchase timestamp and order estimated delivery time. Also, filtering out the data using where clause on the basis of order_status = "delivered"
In the order query use we subtract Estimated_Avg and Actual_Avg to get fast_delivery of every state, then order by fast_delivery and limit 5 to get top 5 state because if difference between estimated deliver time and actual delivery time is larger it means that product is delivered faster than the estimated delivery time

6. **Analysis based on the payments:**

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

```
select extract(year from order_purchase_timestamp) as year,

extract(month from order_purchase_timestamp) as month,
payment_type,
count(distinct o.order_id) as orders
from `CaseStudy.orders` o join
`CaseStudy.payments` p
on o.order_id = p.order_id
group by 1,2,3
order by 1,2,3;
```



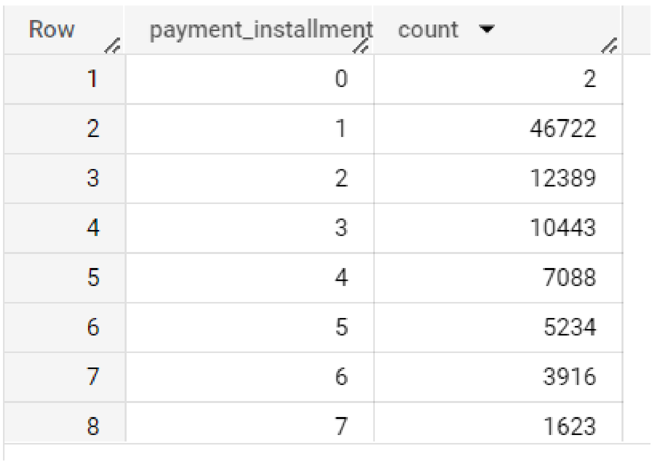
Row	year	month	payment_type	orders
1	2016	9	credit_card	3
2	2016	10	UPI	63
3	2016	10	credit_card	253
4	2016	10	debit_card	2
5	2016	10	voucher	11
6	2016	12	credit_card	1
7	2017	1	UPI	197
8	2017	1	credit_card	582
9	2017	1	debit_card	9
10	2017	1	voucher	33

I inner join orders and payments table to get all the common rows. Extract year and month from order_purchase_timestamp, then group by the data by year, month and the payment_methods column and count the number of orders to get all the orders number placed month on month using different payment method.

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

```
 Select payment_installments,

 count (distinct order_id) as count
 from `CaseStudy.payments`
where payment_type != "voucher"
 group by 1
 order by 1;
```



Row	payment_installment	count
1	0	2
2	1	46722
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623

I used inner join in orders and payments table to get all the common rows. Group by the data by payment_installements. Then count all the distinct order_id because there are multiple rows for similar order_id due to column payment_sequential and we need no of orders on the basis of payment_installement.

Actionable Insights

1. **Growing Trend in Orders:**
   a. The number of orders placed has been steadily increasing over the years. This indicates a positive trend in the e-commerce business.

   b. There is no specific monthly seasonality in the number of orders placed.

2. **Preferred Order Placing Time in Brazil:**
   a. Brazilian customers mostly place their orders in the afternoon (between 13 and 18 hours).

3. **Total & Average Values:**
   a. The average and total order value for each state can provide insights into the spending habits of customers in different regions. Marketing efforts can be tailored to target high-value customers in specific states.
   b. Analysing the average freight value for each state can help identify regions with higher shipping costs. This insight can be used to optimize logistics and shipping strategies to reduce expenses and improve customer satisfaction.
4. **Delivery Time Analysis:**
   a. Understanding the delivery time and the difference between estimated and actual delivery dates can help identify areas for improvement in the supply chain and fulfilment process.
   b. Identifying the top 5 states with the fastest delivery times compared to estimated delivery dates can help recognize regions where the logistics and delivery process is efficient. This can be used as a competitive advantage to attract more customers in those states.

5. **Analysis of Payments:**
   a. Analyzing the number of orders placed based on different payment types can provide insights into customer payment preferences. This information can guide decisions related to payment gateway optimization and offering additional payment options to cater to customer preferences.

Recommendations:

There are some recommendations on more questions that can be more beneficial for the company like:

a. Find the top 5 products that are been buyed in peak hours by the customers.

We already know the peak hour is afternoon, so we can also find the products that are selled in the market so that we can manage supply of those products and it help in managing inventory.

b. Find the quarter-on-quarter best product.

We can find the best product in the quarter due to which we can know about the product sale and manage the inventory of that product according to that.

c. Find the highest and lowest review order.

We can find highest and lowest reviews of the orders to get mistake that are done by company to improve them.

d. Find the product sale by category.

We can boost the categories which are not performing well by getting category details of products

e. Find the top buyer.

We can find top buyer to give them some extra benefits or coupon.