Business Case: Target SQL

First of all, create a **dataset name Target_SQL** in **Big query**.

Upload all the 8 tables into the dataset Target_SQL and allow to auto detect from the schema during uploading.

Table Names are :-

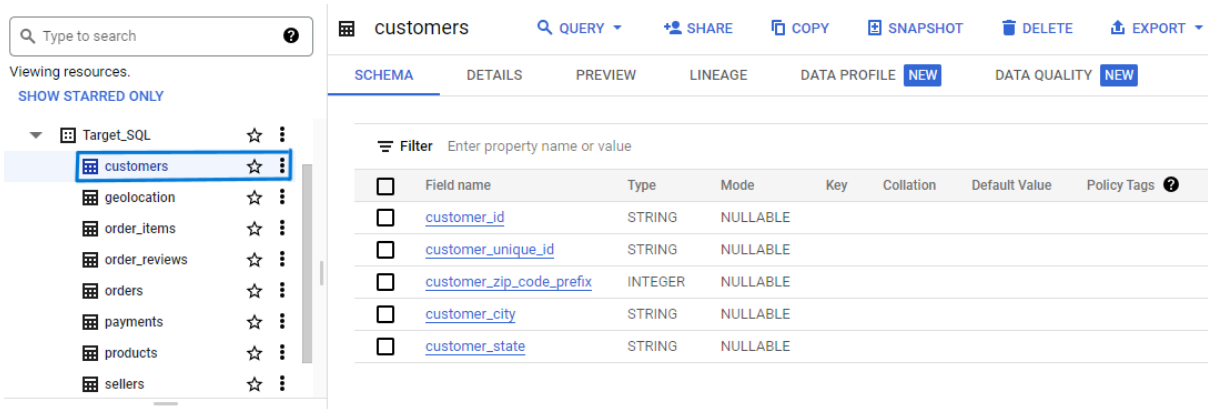
1. customers
2. geolocations
3. order_items
4. order_reviews
5. orders
6. payments
7. products
8. sellers

My method of doing every question is first to write the question, then write down its query/code of bigquery/SQL , the screenshot of my output getting after running the code and finally write my approach for getting the result.

Question :- 1

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

Click on the table name customers marked as blue box from the dataset Target_SQL & we will get the result



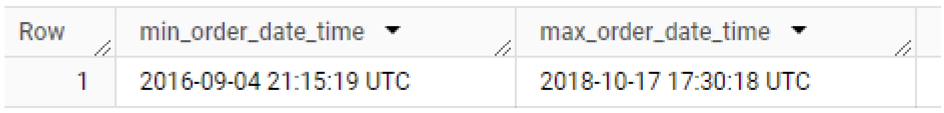
Field name	Type	Mode	Key	Collation	Default Value	Policy Tags
customer_id	STRING	NULLABLE				
customer_unique_id	STRING	NULLABLE				
customer_zip_code_prefix	INTEGER	NULLABLE				
customer_city	STRING	NULLABLE				
customer_state	STRING	NULLABLE				

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

Query: -

```
        select min(order_purchase_timestamp) as min_order_date_time,
        max(order_purchase_timestamp) as max_order_date_time

        from `Target_SQL.orders`;
```



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

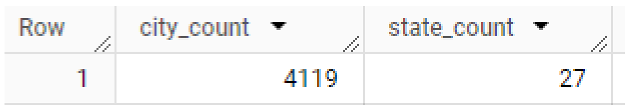
Range lie between minimum and maximum of the particular column.

So, the range of order_purchase lie between 2016-09-04 21:15:19 UTC and 2018-10-17 17:30:18 UTC.

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

Query :-

```
select count(distinct c.customer_city) as city_count,
  count(distinct c.customer_state) as state_count
from `Target_SQL.customers` c
inner join `Target_SQL.orders` o
on c.customer_id = o.customer_id
where extract(year from o.order_purchase_timestamp) between 2016 and 2018
```



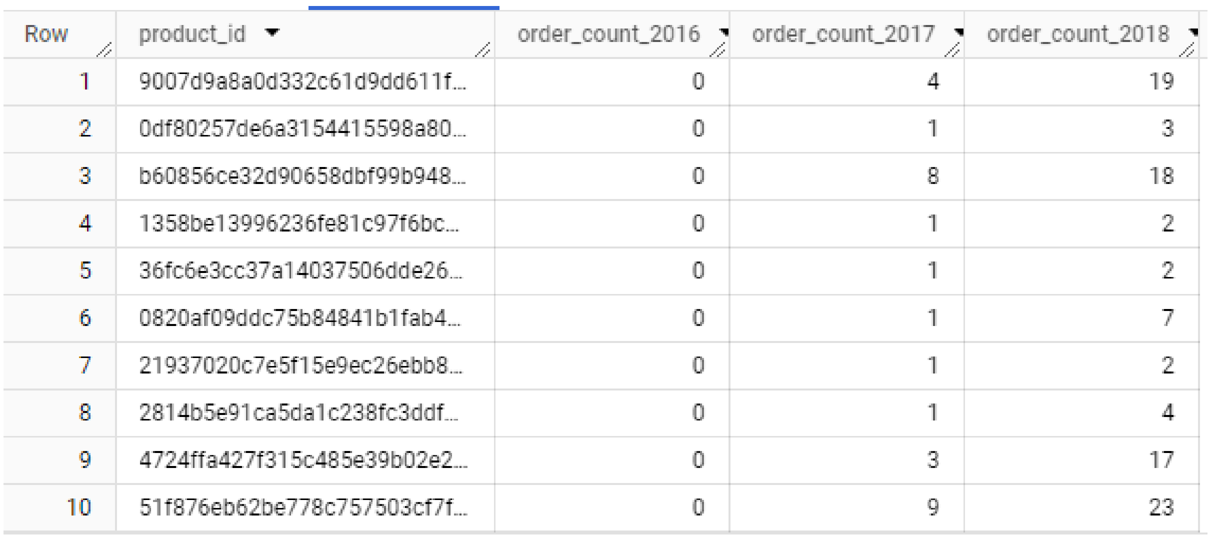
Row	city_count	state_count
1	4119	27

Totat count of city is 4119 & total count of state is 27.

Question :- 2

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

```
with A as (
  select oi.product_id,
    sum(if(extract(year from o.order_purchase_timestamp) = 2016, 1, 0)) as
order_count_2016,
    sum(if(extract(year from o.order_purchase_timestamp) = 2017, 1, 0)) as
order_count_2017,
    sum(if(extract(year from o.order_purchase_timestamp) = 2018, 1, 0)) as order_count_2018
  from `Target_SQL.order_items` oi
  inner join `Target_SQL.orders` o
  on oi.order_id = o.order_id
  group by product_id
)
select *
from A
where order_count_2016 < order_count_2017 and order_count_2017 < order_count_2018
```



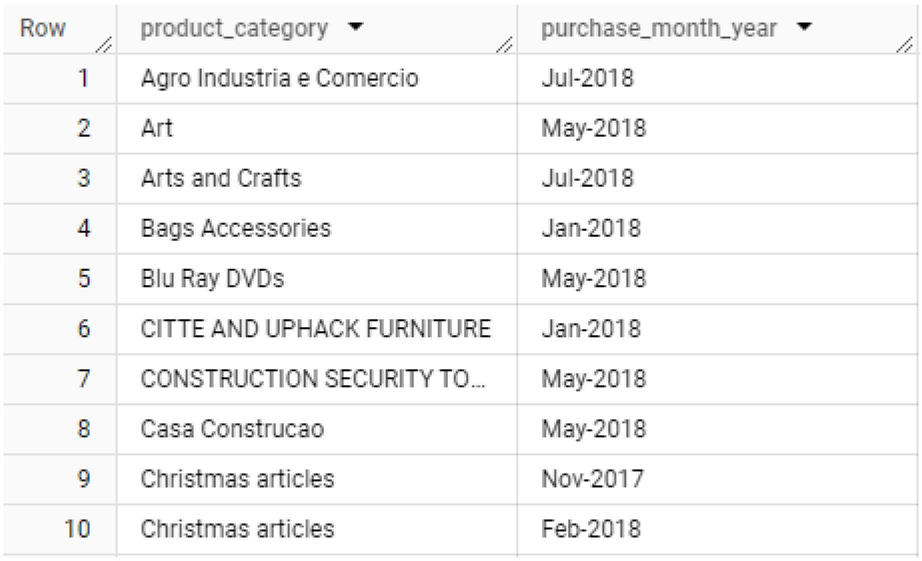
Row	product_id	order_count_2016	order_count_2017	order_count_2018
1	9007d9a8a0d332c61d9dd611f...	0	4	19
2	0df80257de6a3154415598a80...	0	1	3
3	b60856ce32d90658dbf99b948...	0	8	18
4	1358be13996236fe81c97f6bc...	0	1	2
5	36fc6e3cc37a14037506dde26...	0	1	2
6	0820af09ddc75b84841b1fab4...	0	1	7
7	21937020c7e5f15e9ec26ebb8...	0	1	2
8	2814b5e91ca5da1c238fc3ddf...	0	1	4
9	4724ffa427f315c485e39b02e2...	0	3	17
10	51f876eb62be778c757503cf7f...	0	9	23

Firstly, we join order_items and orders table on order_id. Then, group by with product_id column. Then, Create different columns or count of orders in 2016, 2017 & 2018. If count of orders are continuously increasing in year 2016,2017 & 2018, then that particular product are in trend.

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

Query:-

```
with A as (
  select o.order_id,
    format_date("%b-%Y", o.order_purchase_timestamp) as purchase_month_year,
    oi.product_id,
    p.product_category
  from `Target_SQL.orders` o
  inner join `Target_SQL.order_items` oi on o.order_id = oi.order_id
  inner join `Target_SQL.products` p on oi.product_id = p.product_id
  where p.product_category is not null
),
B as (
  select product_category,
    A.purchase_month_year,
    count(order_id) as no_of_orders,
    max(count(order_id)) over(partition by product_category) as max_no_of_orders
  from A
  group by product_category, A.purchase_month_year
  order by product_category
)
select product_category,
  purchase_month_year
from B
where no_of_orders = max_no_of_orders
order by product_category
```



Row	product_category	purchase_month_year
1	Agro Industria e Comercio	Jul-2018
2	Art	May-2018
3	Arts and Crafts	Jul-2018
4	Bags Accessories	Jan-2018
5	Blu Ray DVDs	May-2018
6	CITTE AND UPHACK FURNITURE	Jan-2018
7	CONSTRUCTION SECURITY TO...	May-2018
8	Casa Construcao	May-2018
9	Christmas articles	Nov-2017
10	Christmas articles	Feb-2018

Firstly, we join three tables orders & order_items table on order_id column and order_itmes & products table on product_id column. Then, extract the column order_id, extract month & year from order_purchase_timestamp column named as purchase_month_year, product_id & product_category column and named table as A with CTEs.

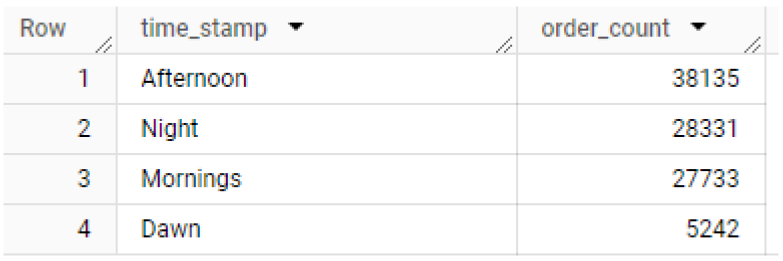
Then, group by table A with product_category and again with purchase_month_year. & extract columns product_category, purchase_month_year, count of order_id as no_of_orders & maximum of no_of_orders with each category. & named table as B CTEs. Then, extract the data where no_of_orders is equal to max_no_of_orders. & finally extract the column product_category & purchase_month_year from the output table.

<u>Note:-</u> I do the task with product_category. With the same way, we can also find out the seasonality of every product in which month, the product is selling the most. The month in which the no of orders are maximum is the seasonality of that product

3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    a. 0-6 hrs : Dawn
    b. 7-12 hrs : Mornings
    c. 13-18 hrs : Afternoon
    d. 19-23 hrs : Night

Code:-

```
with A as (
  select order_id,
  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_stamp
from `Target_SQL.orders`
)
select time_stamp,
  count(order_id) as order_count
from A
group by time_stamp
order by order_count desc
```



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

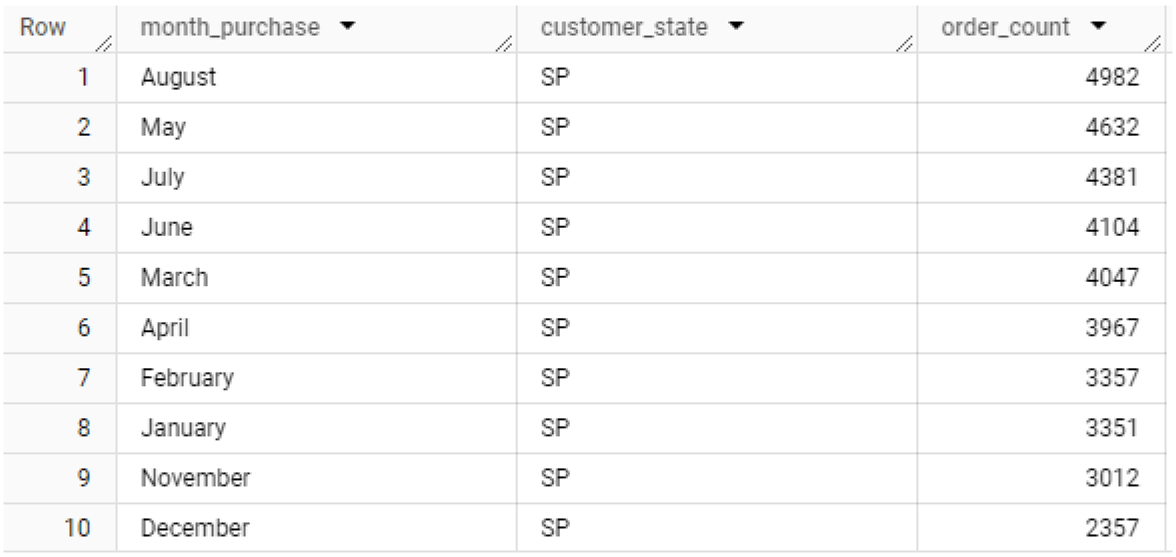
Firstly, we create a column time_stamp in which we categorised the data on the basis of hour from order_purchase_timestamp according to the questing. We count the orders on group by time_stamp and find out the time_stamp in which the orders are placed maximum.

The orders are placed maximum in **<u>Afternoon.</u>**

Question:- 3

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

```
select *
from
(select format_date("%B" , o.order_purchase_timestamp) as month_purchase,
  c.customer_state,
  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 format_date("%B" , o.order_purchase_timestamp), c.customer_state) t
order by order_count desc;
```



Row	month_purchase	customer_state	order_count
1	August	SP	4982
2	May	SP	4632
3	July	SP	4381
4	June	SP	4104
5	March	SP	4047
6	April	SP	3967
7	February	SP	3357
8	January	SP	3351
9	November	SP	3012
10	December	SP	2357

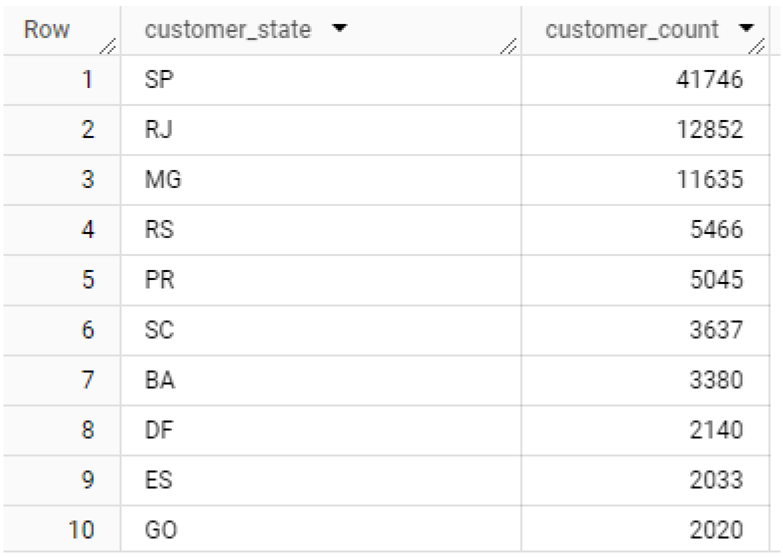
Firstly, join two tables orders & customers on customer_id. Then, extract the month name from the order_purchase_timestamp name as month_purchase. Then, group by the date on month_purhcase and again on customer_state. & give the output of month_purchase, customer_state & count of orders as order_count . order by the whole data on order_count.

Note:- I take the table in the inner query because while giving the output of query with only group by, we was getting output. But when we give output with order by order_count, it was giving error because of o.order_purchase_timestamp column. That's why I take the code as inner table.

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

Query:-

```
select customer_state,
  count(customer_id) as customer_count
from `Target_SQL.customers`
group by customer_state
order by customer_count desc
```



Row	customer_state	customer_count
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

Group by the customer table on custome_state & select the column customer_state and count of coustomer_id as customer_count or order by the data on customer_count desc & we get the output.

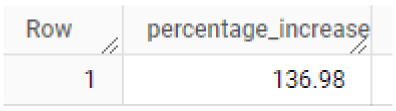
The state SP has largest customers (41746).

Question:- 4

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).

Query:-

```
with A as (
  select sum(if(extract(year from o.order_purchase_timestamp) = 2017, p.payment_value, 0))
as payment_2017,
    sum(if(extract(year from o.order_purchase_timestamp) = 2018, p.payment_value, 0)) as
payment_2018
  from `Target_SQL.orders` o
  inner join `Target_SQL.payments` p
  on o.order_id = p.order_id
  where extract(month from o.order_purchase_timestamp) between 01 and 08
)
select round((payment_2018 - payment_2017)*100/payment_2017, 2) as percentage_increase
from A
```



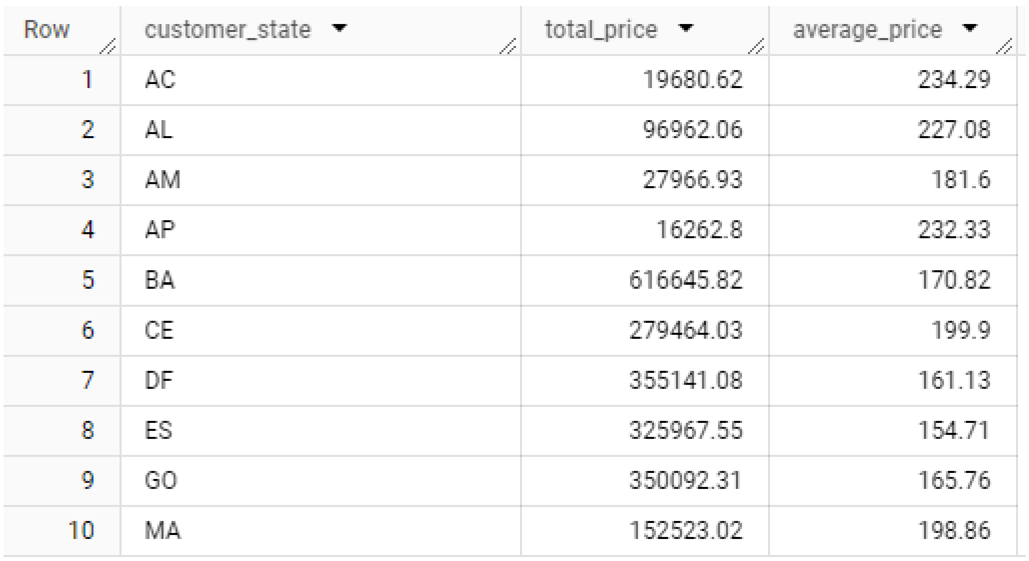
Row	percentage_increase
1	136.98

Firstly, we join two tables orders and payments on the basis of order_id. Then, filter out the data from January(01) to August(08) as per the requirement. Then, find out the sum of price of orders in 2017 and 2018 in 2 different columns. Then, find out the percentage increase of price in orders in 2017 to 2018 using percentage formula & finally round the output to 2 decimal places. & we get our output 136.98

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

Query:-

```sql
select c.customer_state,
  round(sum(p.payment_value), 2) as total_price,
  round(avg(p.payment_value), 2) as average_price
from `Target_SQL.customers` c
inner join `Target_SQL.orders` o on c.customer_id = o.customer_id
inner join `Target_SQL.payments` p on o.order_id = p.order_id
group by c.customer_state
order by customer_state
```



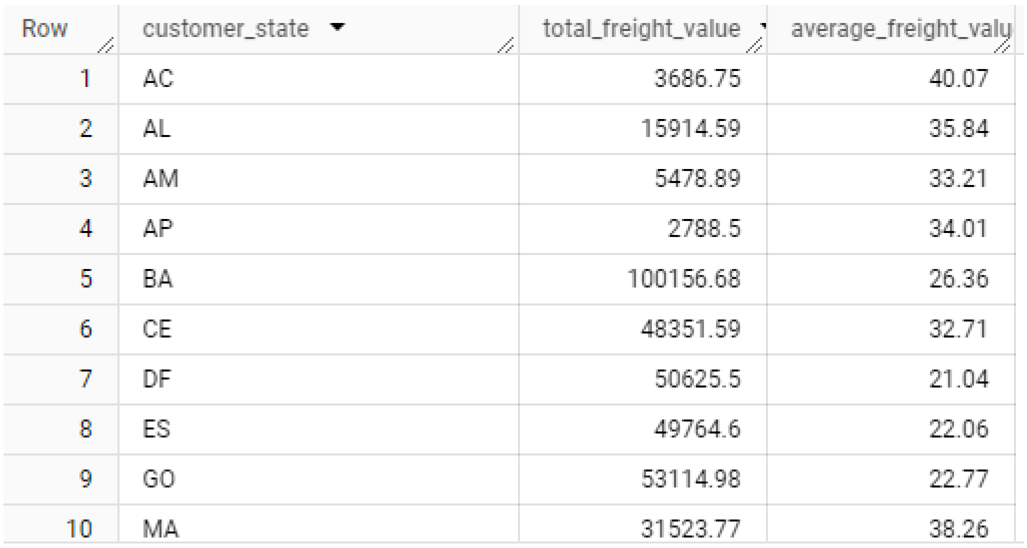
Row	customer_state	total_price	average_price
1	AC	19680.62	234.29
2	AL	96962.06	227.08
3	AM	27966.93	181.6
4	AP	16262.8	232.33
5	BA	616645.82	170.82
6	CE	279464.03	199.9
7	DF	355141.08	161.13
8	ES	325967.55	154.71
9	GO	350092.31	165.76
10	MA	152523.02	198.86

Firstly, we join three tables customers, orders & payments, customers & orders tables on customer_id column and orders and payments tables on order_id column. Then, group by the data on customer_state & find out the column customer_city, sum of payments as total payment done by the customers of the states and average of payments as average payment done by the customers of the states. & finally round the total_price and average_price to 2 decimal places.

3. 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.customers` c
inner join `Target_SQL.orders` o on c.customer_id = o.customer_id
inner join `Target_SQL.order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by customer_state
```



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

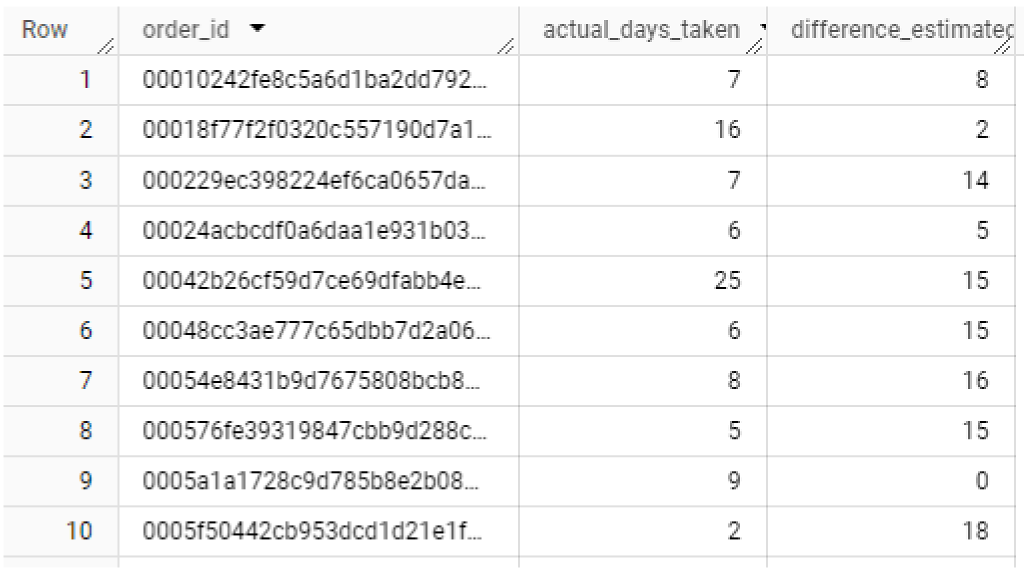
Firstly, we join three tables customers, orders & order_items, customers & orders tables on customer_id column and orders and order_itmes tables on order_id column. Then, group by the data on customer_state & find out the column customer_city, sum of freight payments as total_freight_payment of the states and average of freight payments as average_freight_payment of the states. & finally round the total_freight_price and average_freight_price to 2 decimal places.

Question:- 5

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.

Query:-

```
select order_id,
  date_diff(order_delivered_customer_date, order_purchase_timestamp, day)
actual_days_taken,
  date_diff(order_estimated_delivery_date, order_delivered_customer_date, day)
difference_estimated_actual
from `Target_SQL.orders`
where order_delivered_customer_date is not null
order by order_id
```



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

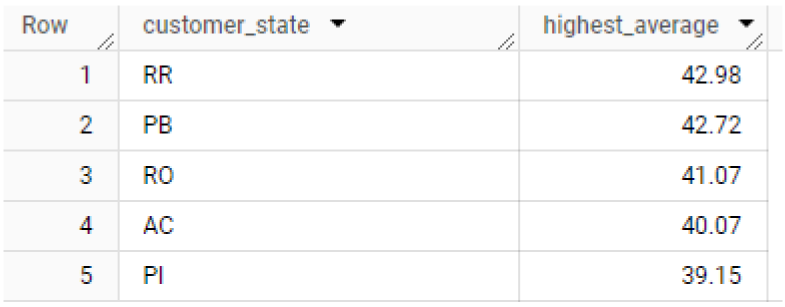
Firstly, filter the data of orders table and remove those rows that has order_delivered_customer_date column has null value. It may be possible that those orders have been cancelled by the customer due to any reason & we cannot find out the actual days count of delivery of those orders that doesn’t have their delivery date. Now, extract the days difference of order_delivered_customer_date and order_purchase_timestamp as actual_days_taken and the days difference of order_estimated_delivery_date and order_delivered_customer_date as difference_estimated_actual.

Finally, give the output as order_id, actual_days_taken & difference_estimated_actual and order it by order_id.

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

Query for top 5 states with the highest average freight value :-

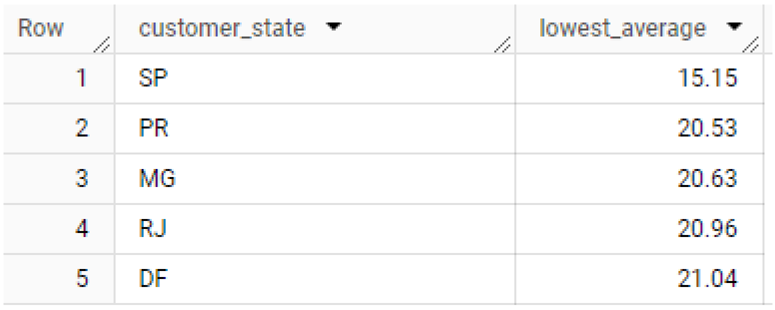
```
select c.customer_state,
  round(avg(oi.freight_value), 2) as highest_average
from `Target_SQL.customers` c
inner join `Target_SQL.orders` o on c.customer_id = o.customer_id
inner join `Target_SQL.order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by highest_average desc
limit 5
```



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

Query for lowest 5 states with the lowest average freight value :-

```
select c.customer_state,
  round(avg(oi.freight_value), 2) as lowest_average
from `Target_SQL.customers` c
inner join `Target_SQL.orders` o on c.customer_id = o.customer_id
inner join `Target_SQL.order_items` oi on o.order_id = oi.order_id
group by c.customer_state
order by lowest_average
limit 5
```



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

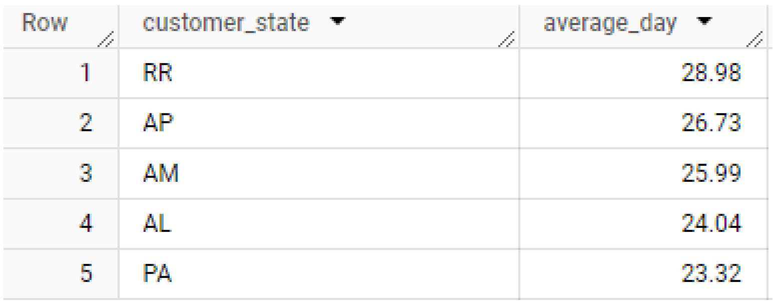
Firstly, we join three tables customers and orders table on customer_id column & orders and order_items table on order_id column. Then, group by the table on customer_state and select the columns customer_state and average of freight_value and round it to 2 decimal places.

Finally, order the output by the average of freight_value in ascending order to get lowest states & in descending order to get highest states. & put the limit 5 to get top 5 values.

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

Query for top 5 states with the highest average delivery time:-

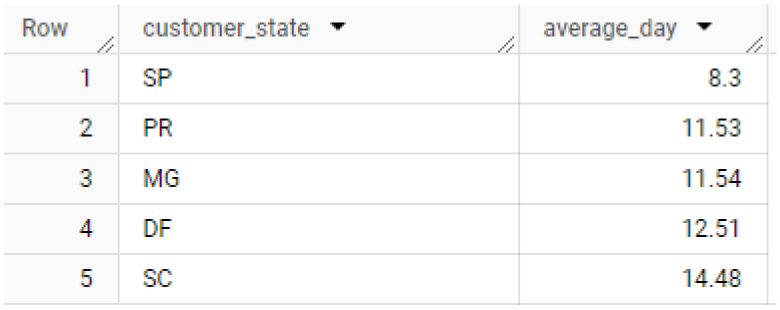
```
select c.customer_state,
  round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),
2) as average_day
from `Target_SQL.customers` c
inner join `Target_SQL.orders` o on c.customer_id = o.customer_id
where o.order_delivered_customer_date is not null
group by c.customer_state
order by average_day desc
limit 5
```



Row	customer_state	average_day
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Query for top 5 states with the lowest average delivery time:-

```
select c.customer_state,
  round(avg(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),
2) as average_day
from `Target_SQL.customers` c
inner join `Target_SQL.orders` o on c.customer_id = o.customer_id
where o.order_delivered_customer_date is not null
group by c.customer_state
order by average_day asc
limit 5
```



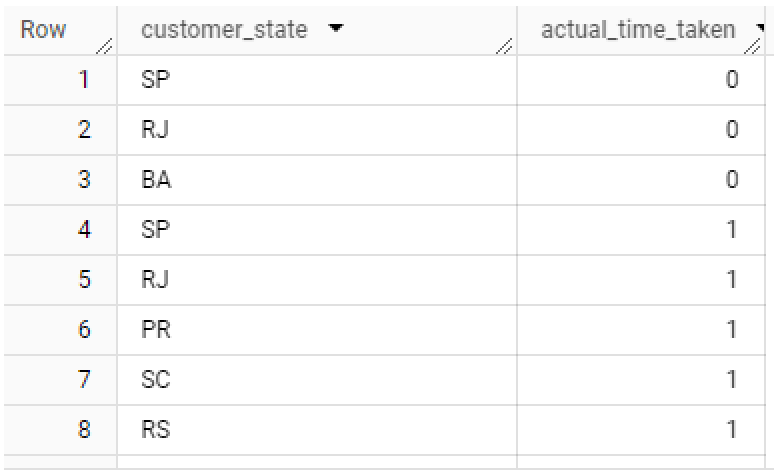
Row	customer_state	average_day
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48

Firstly, we join two tables customers and orders on customer_id column.& remove the rows that have null value in order_delivered_customer_date column.(The reason of this, I have already described in previous question). Then, group by the data on customer_state and find out the average of the days difference between order_delivered_customer_date and order_purchase_timestamp as average_day and round it to 2 decimal places. Order by the data in descending order for the highest average time delivery & in ascending order for the lowest average time delivery. & give limit 5 top get top 5 values in output.

4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.

Query:-

```
with A as (
  select customer_state,
    date_diff(order_delivered_customer_date, order_purchase_timestamp, day) as
actual_time_taken,
    date_diff(order_estimated_delivery_date, order_purchase_timestamp, day) as
estimated_time_taken
  from `Target_SQL.customers` c
  inner join `Target_SQL.orders` o on c.customer_id = o.customer_id
  where o.order_delivered_customer_date is not null
)
select distinct customer_state,
  A.actual_time_taken
from A
where A.actual_time_taken < A.estimated_time_taken
order by A.actual_time_taken
```



Row	customer_state	actual_time_taken
1	SP	0
2	RJ	0
3	BA	0
4	SP	1
5	RJ	1
6	PR	1
7	SC	1
8	RS	1

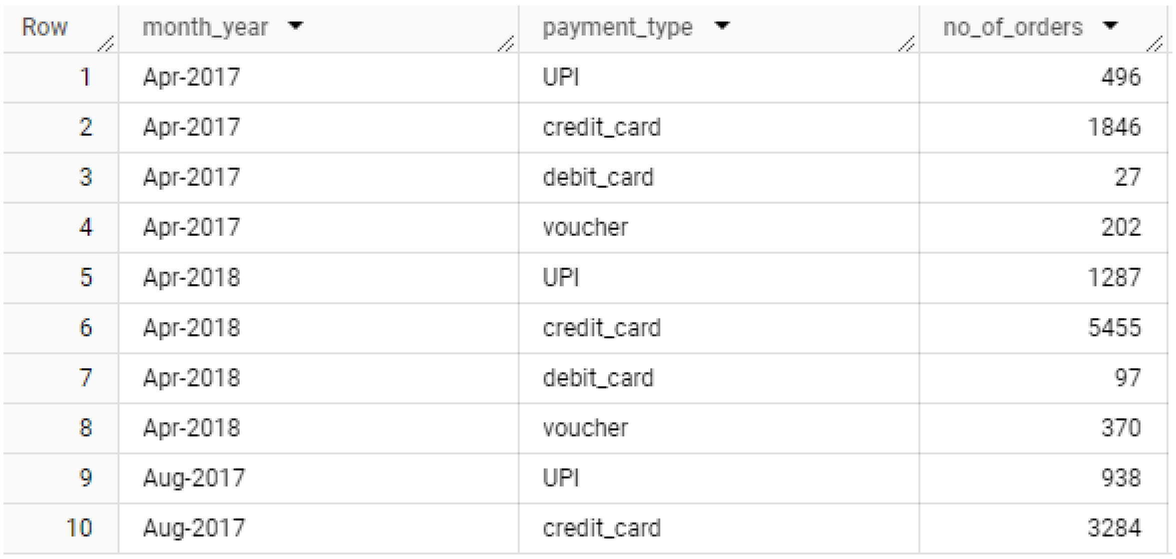
Firstly, we join customers and orders table on customer_id and filter the data by removing those rows that have null value in order_delivered_customer_date. Then, find the day difference of order_delivered_customer_date & order_purchase_timestamp as actual_time_taken and order_estimated_delivery_date & order_purchase_timestamp as estimated_time_taken.

Now, take this data as a table & filter data on actual_time_taken < estimated_time_taken & give customer_state & actual_time_taken as output & order by the data with actual_time_taken column so that we can find out the data of those states that taken minimum time in delivery.

Question:- 6

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

```
select * from
(select format_date("%b-%Y", order_purchase_timestamp) as month_year,
  p.payment_type,
  count(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 format_date("%b-%Y", order_purchase_timestamp), p.payment_type) t
order by month_year, payment_type
```



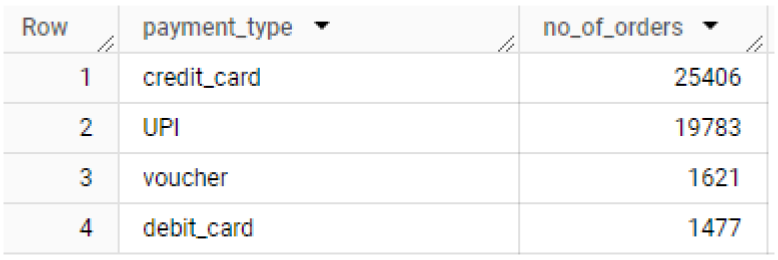
Row	month_year	payment_type	no_of_orders
1	Apr-2017	UPI	496
2	Apr-2017	credit_card	1846
3	Apr-2017	debit_card	27
4	Apr-2017	voucher	202
5	Apr-2018	UPI	1287
6	Apr-2018	credit_card	5455
7	Apr-2018	debit_card	97
8	Apr-2018	voucher	370
9	Aug-2017	UPI	938
10	Aug-2017	credit_card	3284

Firstly, join orders and payments table on order_id column, then extract the month-year from order_purchase_timestamp as month_year. Now, group by the data by month_year & payment_type. Then, give output columns of month_year, payment_type & count of order_id as no_of_orders & finally order by the data on month_year and payment_type.

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

Query:-

```sql
select payment_type,
  count(order_id) as no_of_orders
from `Target_SQL.payments`
where payment_sequential = payment_installments and payment_value != 0
group by payment_type
order by no_of_orders desc
```



Row	payment_type	no_of_orders
1	credit_card	25406
2	UPI	19783
3	voucher	1621
4	debit_card	1477

Take table payments. In this table, column payment_instalments is for the total number of instalments is created for the order and the column name payment_sequential is for that how many instalments the customer has deposited for that order. If we want to find out those orders whose all of the instalments have been paid then payment_instakments = payment_sequential and also filter out the data that have total payment values 0 . because those customers have not paid any amount on that orders.

Now, group by the data on payment_type and get count(order_id) as no_of_orders for finding total number of orders with different payment_type.

Finally order by the data with no_of_orders in descending order.