Business Case: Target SQL

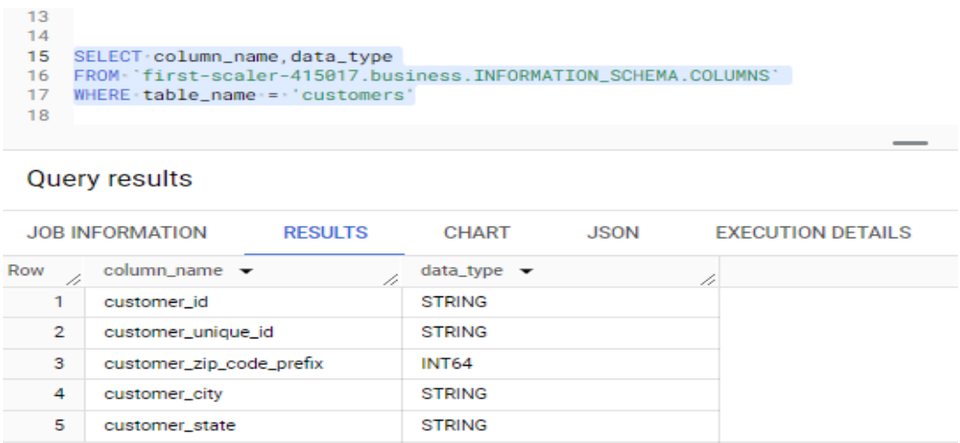
Q1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:

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

Query:

```
select column_name, data_type
from `first-scaler-415017.business.INFORMATION_SCHEMA.COLUMNS`
where table_name = 'customers'
```

Screenshot: -



```
SELECT column_name,data_type
FROM `first-scaler-415017.business.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers'
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

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: Here customer table is having 5 columns with 2 datatypes STRING and INT64

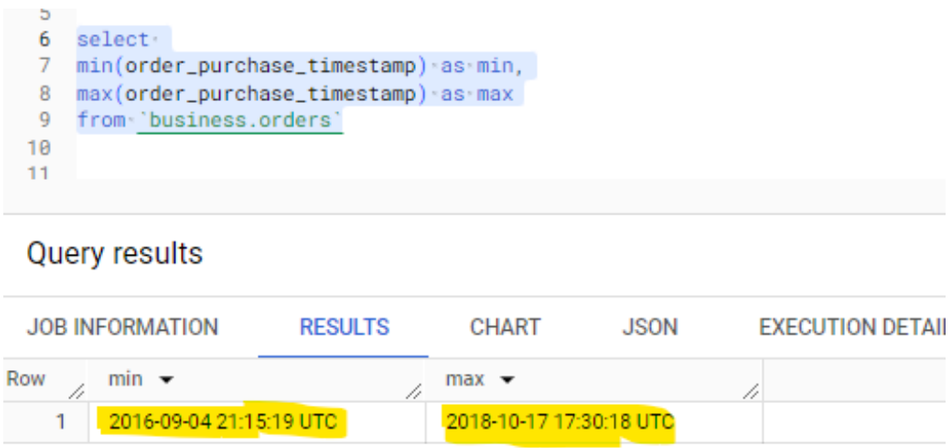
Recommendation: N/A

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

Query:

```
select min(order_purchase_timestamp) as min, max(order_purchase_timestamp) as max
from `business.orders`
```

Screenshot: -



```
select
min(order_purchase_timestamp) as min,
max(order_purchase_timestamp) as max
from `business.orders`
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAIL

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

Insights: - First order was made on 2016-09-04 and last order was made on 2018-10-17.

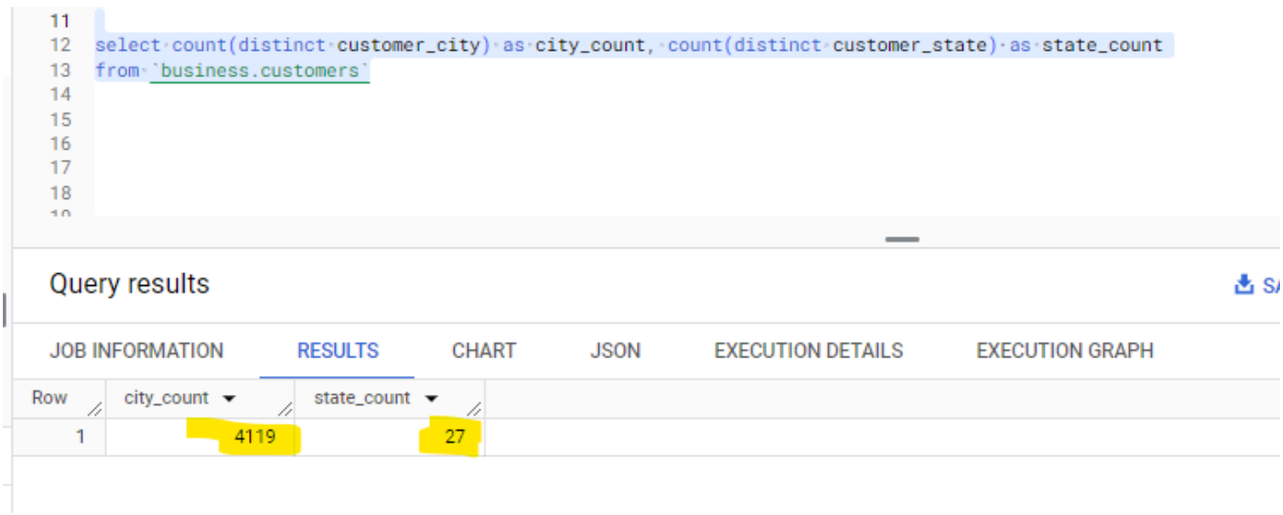
Recommendation: - N/A

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

Query: -

```
select count(distinct customer_city) as city_count, count(distinct customer_state) as state_count
from `business.customers`
```

Screenshot: -



Insights: - We have customers from 4119 cities and from 27 different states.

Suggestions: - Increase the customers by improving the overall performance. Also increase the network in various other states.

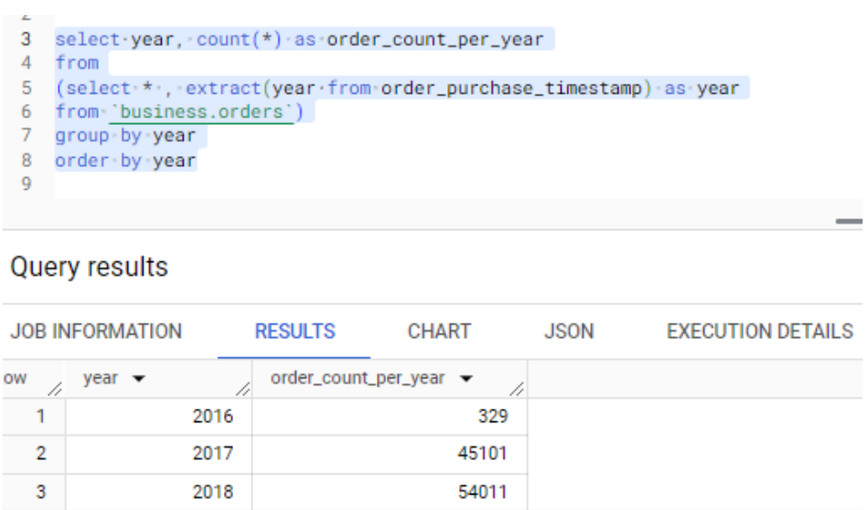
Q2. In-depth Exploration:

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

Query:

```
select year, count(*) as order_count_per_year
from
(select * , extract(year from order_purchase_timestamp) as year
from `business.orders`)
group by year
order by year
```

Screenshot:



Insights:

Yes, the number of orders were increased over the years

In 2016, we have 329 orders,

In 2017, we have 45,101 orders,

in 2018, we have 54,011 orders.

In 2016 we have only 329 orders because we have data from September 2016 to December 2016. i.e., In 2016 first order in the order table was on 2016-09-04 and last order was on 2016-12-23. Also, the orders in 2016 are very less, let us assume in 2017 the total orders are 45000 so on an average 3750 orders were placed every month. So, for 4 months total orders were 329 which is very less.

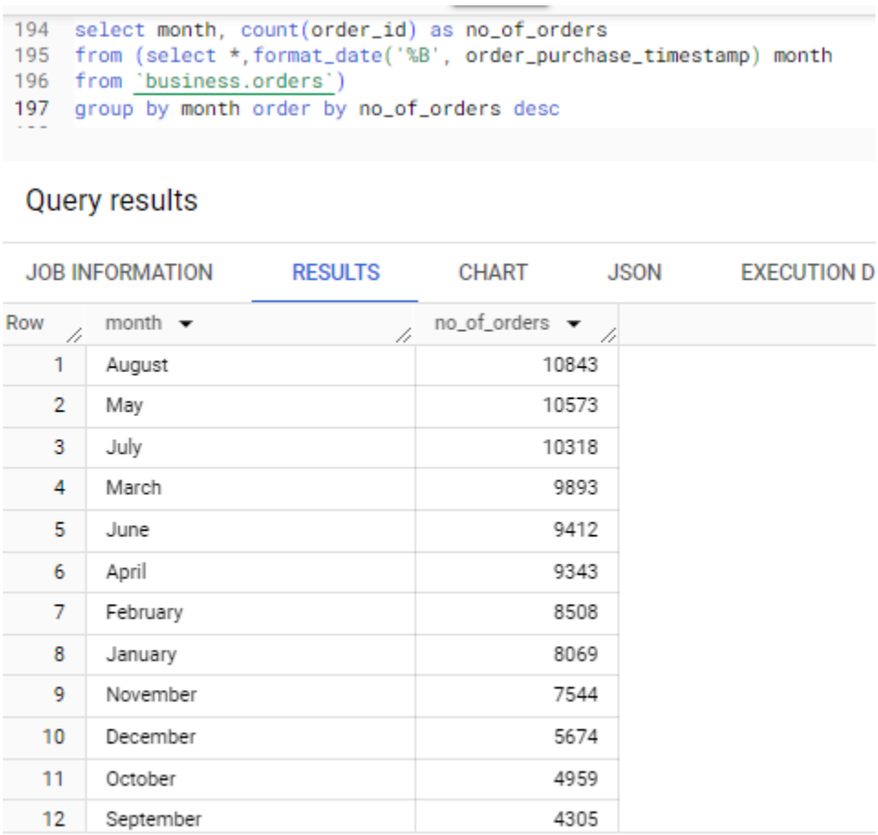
Suggestions: - We can increase the number of orders by improving the overall performance

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

Query:

```
select month, count(order_id) as no_of_orders
from (select *,format_date('%B', order_purchase_timestamp) month
from `business.orders`)
group by month
order by no_of_orders desc
```

Screenshot:



```
194  select month, count(order_id) as no_of_orders
195  from (select *,format_date('%B', order_purchase_timestamp) month
196  from `business.orders`)
197  group by month order by no_of_orders desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION D

Row	month	no_of_orders
1	August	10843
2	May	10573
3	July	10318
4	March	9893
5	June	9412
6	April	9343
7	February	8508
8	January	8069
9	November	7544
10	December	5674
11	October	4959
12	September	4305

Insights: Here we can say that the number of orders in the month of August, May and July are high. And the number of orders in the month of October and September are very less.

Suggestions:

- We can keep discounts in those months in the month of December, October and September so that the number of orders will increase.
- We need to keep some combo offers, like combining the items which are more ordered with less items ordered.
- We have to maintain more stock in the month of August, May and July.

3. **During what time of the day, do the Brazilian customers mostly place their orders?**

Query:

```
select
different_times, count(different_times) as count_different_times from
(select order_id,order_time,order_purchase_timestamp,
case
when order_time between '00:00:00' and '06:59:59'
then  'Dawn'
when order_time between '07:00:00' and '12:59:59'
then 'Mornings'
when order_time between '13:00:00' and '18:59:59'
then 'Afternoon'
else 'Night'
end as different_times
from(select extract(time from order_purchase_timestamp) as order_time,*
from `business.orders`) tbl1)tbl2
group by different_times
```

Screenshot:

```
30  select
31  different_times, count(different_times) as count_different_times from
32  (select order_id,order_time,order_purchase_timestamp,
33  case
34  when order_time between '00:00:00' and '06:59:59'
35  then  'Dawn'
36  when order_time between '07:00:00' and '12:59:59'
37  then 'Mornings'
38  when order_time between '13:00:00' and '18:59:59'
39  then 'Afternoon'
40  else 'Night'
41  end as different_times
42  from(select extract(time from order_purchase_timestamp) as order_time,*
43  from `business.orders`) tbl1)tbl2
44  group by different_times
45  order by count_different_times
46
```

Query results

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

Row	different_times	count_different_time
1	Dawn	5242
2	Mornings	27733
3	Night	28331
4	Afternoon	38135

Insights:

- Brazilian customers ordered the items during all the times in a day.
- In afternoon customers have ordered the most.
- Customers also ordered in morning and night but less than afternoon.
- The number of items ordered in dawn is less.

Suggestions:

- In afternoon since we have a greater number of orders, we can have more manpower working at this time and same goes to morning and night.
- In dawn since the orders are less, we can have some discount added in this timeline only so that the number of orders increases. Also, we can provide free delivery for those customers who order in dawn time.

Q3. Evolution of E-commerce orders in the Brazil region:

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

Query:

```
select customer_state, month, count(order_id) as no_of_orders
from (select order_id, customer_state,format_date('%B %Y',order_purchase_timestamp) as month
from `business.customers` c  join `business.orders` o
on c.customer_id = o.customer_id) tbl
```

group by customer_state,month
order by no_of_orders desc

Screenshot:

```
267 select customer_state, month, count(order_id) as no_of_orders
268 from (select order_id, customer_state,format_date('%B %Y',order_purchase_timestamp) as month
269 from `business.customers` c join `business.orders` o
270 on c.customer_id = o.customer_id) tbl
271 group by customer_state,month
272 order by month,no_of_orders desc
```

Query results

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

Row	customer_state	month	no_of_orders
1	SP	April 2017	908
2	RJ	April 2017	338
3	MG	April 2017	275
4	RS	April 2017	139
5	PR	April 2017	114
6	SC	April 2017	105
7	BA	April 2017	93
8	ES	April 2017	46
9	CE	April 2017	43
10	GO	April 2017	41

```
267 select customer_state, month, count(order_id) as no_of_orders
268 from (select order_id, customer_state,format_date('%B %Y',order_purchase_timestamp) as month
269 from `business.customers` c join `business.orders` o
270 on c.customer_id = o.customer_id) tbl
271 group by customer_state,month
272 order by no_of_orders, month
273
```

Query results

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

ow	customer_state	month	no_of_orders
1	PR	December 2016	1
2	MS	January 2017	1
3	AP	July 2017	1
4	TO	July 2017	1
5	RR	July 2017	1
6	AM	June 2017	1
7	RR	May 2018	1
8	PB	October 2016	1
9	PI	October 2016	1
10	RR	October 2016	1

```
select customer_state, month, count(order_id) as no_of_orders
from (select order_id, customer_state,format_date('%B %Y',order_purchase_timestamp) as month
from `business.customers` c join `business.orders` o
on c.customer_id = o.customer_id) tbl
group by customer_state,month
order by no_of_orders desc
```

Query results

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

Row	customer_state	month	no_of_orders
1	SP	August 2018	3253
2	SP	May 2018	3207
3	SP	April 2018	3059
4	SP	January 2018	3052
5	SP	March 2018	3037
6	SP	November 2017	3012
7	SP	July 2018	2777
8	SP	June 2018	2773
9	SP	February 2018	2703
10	SP	December 2017	2357

Insights:

- Number of orders were maximum in SP state.
- Number of orders in many states are less.

Suggestions:

- We have to maintain the customers in SP state.
- In all other states we have to increase our network by advertising.
- We have to get to know what items the customers look for in low count states.

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

Query:

```
select customer_state, count(customer_unique_id) as number_of_customers
from `business.customers` group by customer_state
order by number_of_customers desc
```

Screenshot:

```
select customer_state, count(customer_unique_id) as number_of_customers
from `business.customers` group by customer_state
order by number_of_customers desc
```

Query results

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

low	customer_state	number_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
11	PE	1652

```
#3.2
select customer_state, count(customer_unique_id) as number_of_customers
from `business.customers` group by customer_state
order by number_of_customers
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

	customer_state	number_of_custome
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

Insights:

- We have customers from all the states.
- Number of customers are more in SP, RJ, MG states.
- We have moderate number of customers in other states.
- We have less than 100 number of customers in states like RR, AP, AC.

Suggestions:

- We have to increase the number of customers in above mentioned states.
- We can provide sum additional benefits if one customer refers others from same state or different states.

Q4. Impact on Economy: Analyse 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 *, round(100 *((tbl2.cost_of_orders - lag(tbl2.cost_of_orders) over(order by tbl2.cost_of_orders ) )
/lag(tbl2.cost_of_orders) over(order by tbl2.cost_of_orders )),2) || ' %' as percentage_increase
from(select order_year,sum(payment_value) as cost_of_orders
from  (select *,
extract(month from t2.order_purchase_timestamp) as order_month,
extract(year from t2.order_purchase_timestamp) as order_year
from `business.payments` t1 join `business.orders` t2
on t1.order_id = t2.order_id) tbl1
where order_month between 1 and 8 and order_year between 2017 and 2018
group by order_year) tbl2
```

Screenshot:

```
select *,
round(100*((tbl2.cost_of_orders - lag(tbl2.cost_of_orders) over(order by tbl2.cost_of_orders ))
/lag(tbl2.cost_of_orders) over(order by tbl2.cost_of_orders )),2) || '%' as percentage_increase
from(select order_year,sum(payment_value) as cost_of_orders
from (select *,
extract(month from t2.order_purchase_timestamp) as order_month,
extract(year from t2.order_purchase_timestamp) as order_year
from `business.payments` t1 join `business.orders` t2
on t1.order_id = t2.order_id) tbl1
where order_month between 1 and 8 and order_year between 2017 and 2018
group by order_year) tbl2;
```

Query results

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

Row	order_year	cost_of_orders	percentage_increase
1	2017	3669022.120000...	*null*
2	2018	8694733.839999...	136.98 %

Insights:

- From 2017 to 2018 the number of orders increased by 136.98 % which is very good for the company.

Suggestions:

- The company has to maintain the number of customers and provide some additional benefits who refers somebody.
- We can provide some discounts for regular customers and good suggestions to order the items.
- We have to provide more benefits for new customers who joins.

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

Query:

```
select c.customer_state,
round(sum(payment_value),2) as total_value_of_order_price, round(avg(payment_value),2) as
average_value_of_order_price
from `business.payments` p join `business.orders` o on p.order_id = o.order_id
join `business.customers` c on o.customer_id = c.customer_id
group by c.customer_state order by total_value_of_order_price desc, average_value_of_order_price desc;
```

Screenshot:

```
select c.customer_state,
round(sum(payment_value),2) as total_value_of_order_price, round(avg(payment_value),2) as average_value_of_order_price
from `business.payments` p join `business.orders` o on p.order_id = o.order_id
join `business.customers` c on o.customer_id = c.customer_id
group by c.customer_state order by total_value_of_order_price desc, average_value_of_order_price desc;
```

Query results

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

Row	customer_state	total_value_of_order_price	average_value_of_order_price
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71
11	PE	324850.44	187.99

```
select c.customer_state,
round(sum(payment_value),2) as total_value_of_order_price, round(avg(payment_value),2) as average_value_of_order_price
from `business.payments` p join `business.orders` o on p.order_id = o.order_id
join `business.customers` c on o.customer_id = c.customer_id
group by c.customer_state order by total_value_of_order_price
```

Query results

SAVE R

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

Row	customer_state	total_value_of_order	average_value_of_or
1	RR	10064.62	218.8
2	AP	16262.8	232.33
3	AC	19680.62	234.29
4	AM	27966.93	181.6
5	RO	60866.2	233.2

```
select c.customer_state,
round(sum(payment_value),2) as total_value_of_order_price, round(avg(payment_value),2) as average_value_of_order_price
from `business.payments` p join `business.orders` o on p.order_id = o.order_id
join `business.customers` c on o.customer_id = c.customer_id
group by c.customer_state order by average_value_of_order_price
```

Query results

SAVE RE

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

Row	customer_state	total_value_of_order	average_value_of_or
1	SP	5998226.96	137.5

```
select c.customer_state,
round(sum(payment_value),2) as total_value_of_order_price, round(avg(payment_value),2) as average_value_of_order_price,
count(o.order_id) as order_count
from `business.payments` p join `business.orders` o on p.order_id = o.order_id
join `business.customers` c on o.customer_id = c.customer_id
group by c.customer_state order by total_value_of_order_price desc
```

uery results

SAVE RES

JB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS | EXECUTION GRAPH

Row	customer_state	total_value_of_order	average_value_of_or	order_count
1	SP	5998226.96	137.5	43622

Insights:

- In SP state the number of orders is maximum so the price of total number of orders are highest contributing to most of the company's economy.
- Followed by few other states (shown in screenshot).
- States with lowest total value are RR, AP, AC
- Average value in state SP is minimum because the number of orders is more.

Recommendation:

- There are almost 8 states which have less than 100000. So, Company has to focus on these states so that next year number of orders increase in these states.

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

Query:

```
select t2.seller_state,
round(sum(t1.freight_value),2) as total_freight_value, round(avg(t1.freight_value),2) as average_freight_value
from `business.order_items` t1 join `business.sellers` t2
on t1.seller_id = t2.seller_id
group by t2.seller_state order by total_freight_value desc, average_freight_value desc
```

Screenshot:

```
select t2.seller_state,
round(sum(t1.freight_value),2) as total_freight_value, round(avg(t1.freight_value),2) as average_freight_value
from `business.order_items` t1 join `business.sellers` t2
on t1.seller_id = t2.seller_id
group by t2.seller_state order by total_freight_value desc, average_freight_value desc
```

Query results

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

Row	seller_state	total_freight_value	average_freight_valu
1	SP	1482487.67	18.45
2	MG	212595.06	24.08
3	PR	197013.52	22.72
4	SC	106547.06	26.15
5	RJ	93829.9	19.47
6	RS	57243.09	26.03
7	BA	19700.68	30.64
8	DF	18494.06	20.57
9	GO	12565.5	24.16
10	PE	12392.46	27.66

```
select t2.seller_state,
round(sum(t1.freight_value),2) as total_freight_value, round(avg(t1.freight_value),2) as average_freight_value
from `business.order_items` t1 join `business.sellers` t2
on t1.seller_id = t2.seller_id
group by t2.seller_state order by total_freight_value
```

Query results

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

Row	seller_state	total_freight_value	average_freight_valu
1	AC	32.84	32.84
2	AM	81.8	27.27
3	PA	155.11	19.39
4	SE	318.49	31.85
5	PI	443.32	36.94
6	RO	712.78	50.91
7	MS	1198.96	23.98
8	RN	1304.11	23.29
9	PB	1489.15	39.19
10	CE	4359.83	46.38
11	MT	4631.73	31.94

Insights:

- SP state is having highest freight value because the number of orders in this state are more and so the average value is less.
- RJ, MG, RS, PR, BA, SC are having higher freight value.
- AC state is having the lowest freight value. If we observe total freight value and average freight value are same because there is only one seller in this state.

Suggestions:

- Company needs to increase the warehouses in all the places so that freight value will be less.

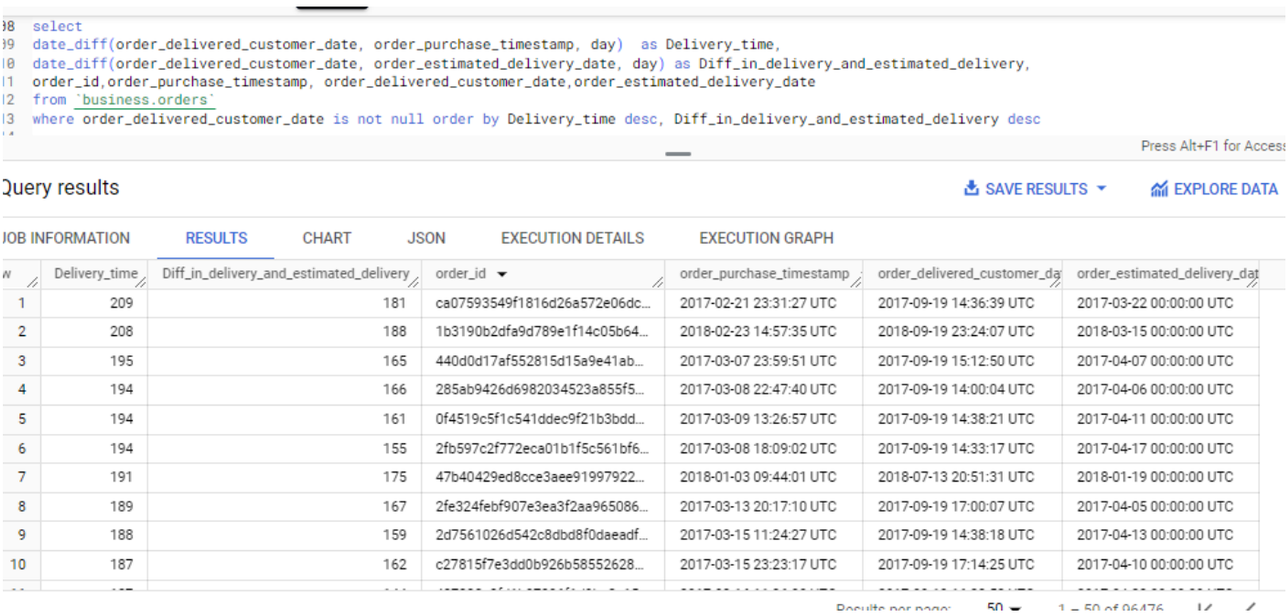
Q5: **Analysis based on sales, freight and delivery time.**

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.**

Query:

```
select
date_diff(order_delivered_customer_date, order_purchase_timestamp, day)  as Delivery_time,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as
Diff_in_delivery_and_estimated_delivery,
order_id,order_purchase_timestamp, order_delivered_customer_date,order_estimated_delivery_date
from `business.orders`
where order_delivered_customer_date is not null order by Delivery_time ,
Diff_in_delivery_and_estimated_delivery
```

Screenshot:



```
select
date_diff(order_delivered_customer_date, order_purchase_timestamp, day)  as Delivery_time,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as Diff_in_delivery_and_estimated_delivery,
order_id,order_purchase_timestamp, order_delivered_customer_date,order_estimated_delivery_date
from `business.orders`
where order_delivered_customer_date is not null order by Delivery_time desc, Diff_in_delivery_and_estimated_delivery desc
```

Press Alt+F1 for Acces

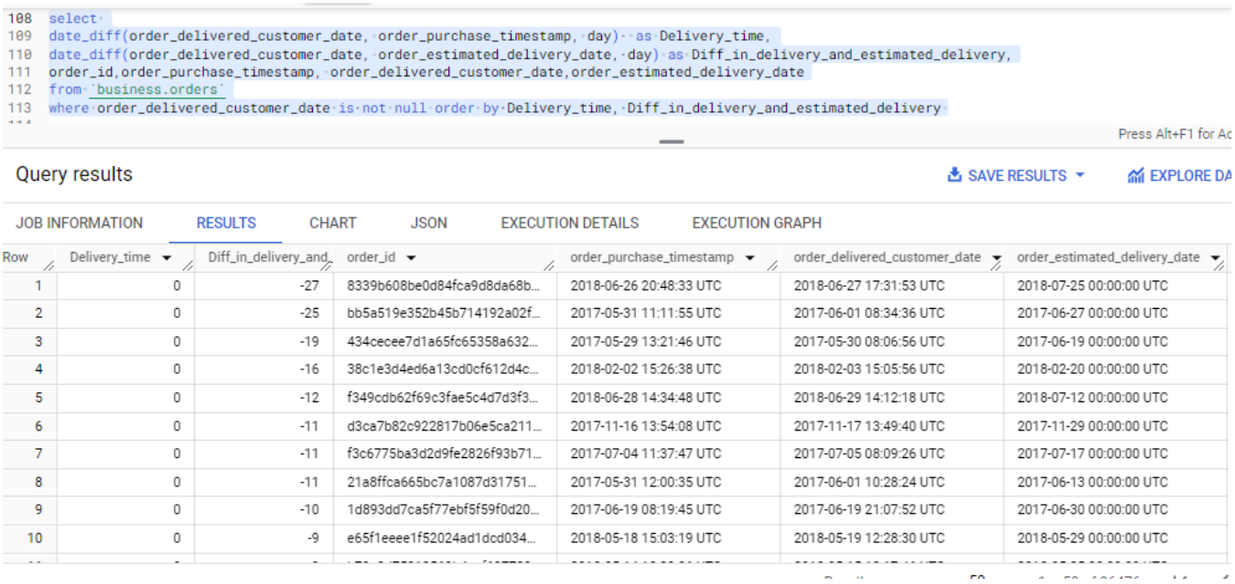
Query results

SAVE RESULTS | EXPLORE DATA

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

Row	Delivery_time	Diff_in_delivery_and_estimated_delivery	order_id	order_purchase_timestamp	order_delivered_customer_da	order_estimated_delivery_dat
1	209	181	ca07593549f1816d26a572e06dc...	2017-02-21 23:31:27 UTC	2017-09-19 14:36:39 UTC	2017-03-22 00:00:00 UTC
2	208	188	1b3190b2dfa9d789e1f14c05b64...	2018-02-23 14:57:35 UTC	2018-09-19 23:24:07 UTC	2018-03-15 00:00:00 UTC
3	195	165	440d0d17af552815d15a9e41ab...	2017-03-07 23:59:51 UTC	2017-09-19 15:12:50 UTC	2017-04-07 00:00:00 UTC
4	194	166	285ab9426d6982034523a855f5...	2017-03-08 22:47:40 UTC	2017-09-19 14:00:04 UTC	2017-04-06 00:00:00 UTC
5	194	161	0f4519c5f1c541ddec9f21b3bdd...	2017-03-09 13:26:57 UTC	2017-09-19 14:38:21 UTC	2017-04-11 00:00:00 UTC
6	194	155	2fb597c2f772eca01b1f5c561bf6...	2017-03-08 18:09:02 UTC	2017-09-19 14:33:17 UTC	2017-04-17 00:00:00 UTC
7	191	175	47b40429ed8cce3aee91997922...	2018-01-03 09:44:01 UTC	2018-07-13 20:51:31 UTC	2018-01-19 00:00:00 UTC
8	189	167	2fe324febf907e3ea3f2aa965086...	2017-03-13 20:17:10 UTC	2017-09-19 17:00:07 UTC	2017-04-05 00:00:00 UTC
9	188	159	2d7561026d542c8dbd8f0daeadf...	2017-03-15 11:24:27 UTC	2017-09-19 14:38:18 UTC	2017-04-13 00:00:00 UTC
10	187	162	c27815f7e3dd0b926b58552628...	2017-03-15 23:23:17 UTC	2017-09-19 17:14:25 UTC	2017-04-10 00:00:00 UTC

Results per page: 50 | 1 – 50 of 96476



```
select
date_diff(order_delivered_customer_date, order_purchase_timestamp, day)  as Delivery_time,
date_diff(order_delivered_customer_date, order_estimated_delivery_date, day) as Diff_in_delivery_and_estimated_delivery,
order_id,order_purchase_timestamp, order_delivered_customer_date,order_estimated_delivery_date
from `business.orders`
where order_delivered_customer_date is not null order by Delivery_time, Diff_in_delivery_and_estimated_delivery
```

Press Alt+F1 for Ac

Query results

SAVE RESULTS | EXPLORE DA

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

Row	Delivery_time	Diff_in_delivery_and_	order_id	order_purchase_timestamp	order_delivered_customer_date	order_estimated_delivery_date
1	0	-27	8339b608be0d84fca9d8da68b...	2018-06-26 20:48:33 UTC	2018-06-27 17:31:53 UTC	2018-07-25 00:00:00 UTC
2	0	-25	bb5a519e352b45b714192a02f...	2017-05-31 11:11:55 UTC	2017-06-01 08:34:36 UTC	2017-06-27 00:00:00 UTC
3	0	-19	434cecee7d1a65fc65358a632...	2017-05-29 13:21:46 UTC	2017-05-30 08:06:56 UTC	2017-06-19 00:00:00 UTC
4	0	-16	38c1e3d4ed6a13cd0cf612d4c...	2018-02-02 15:26:38 UTC	2018-02-03 15:05:56 UTC	2018-02-20 00:00:00 UTC
5	0	-12	f349cdb62f69c3fae5c4d7d3f3...	2018-06-28 14:34:48 UTC	2018-06-29 14:12:18 UTC	2018-07-12 00:00:00 UTC
6	0	-11	d3ca7b82c922817b06e5ca211...	2017-11-16 13:54:08 UTC	2017-11-17 13:49:40 UTC	2017-11-29 00:00:00 UTC
7	0	-11	f3c6775ba3d2d9fe2826f93b71...	2017-07-04 11:37:47 UTC	2017-07-05 08:09:26 UTC	2017-07-17 00:00:00 UTC
8	0	-11	21a8ffca665bc7a1087d31751...	2017-05-31 12:00:35 UTC	2017-06-01 10:28:24 UTC	2017-06-13 00:00:00 UTC
9	0	-10	1d893dd7ca5f77ebf5f59f0d20...	2017-06-19 08:19:45 UTC	2017-06-19 21:07:52 UTC	2017-06-30 00:00:00 UTC
10	0	-9	e65f1eeee1f52024ad1dcd034...	2018-05-18 15:03:19 UTC	2018-05-19 12:28:30 UTC	2018-05-29 00:00:00 UTC

Insights:

- In the output **Delivery time** is the difference in **order purchase date and order delivered date**.
- Here the first order (Screenshot 1) took **209 days** to deliver the order, but it was supposed to deliver in **181 days**. So, the company took much more time to deliver the order.
- In the second screenshot Delivery time is 0, it means the order was delivered in less than a day.
- Also, difference in **estimated delivery and actual delivery** is (-27). It means order was supposed to deliver within 27 days but the order has been delivered in less than a day. It means company took less time to deliver the order.

Suggestions:

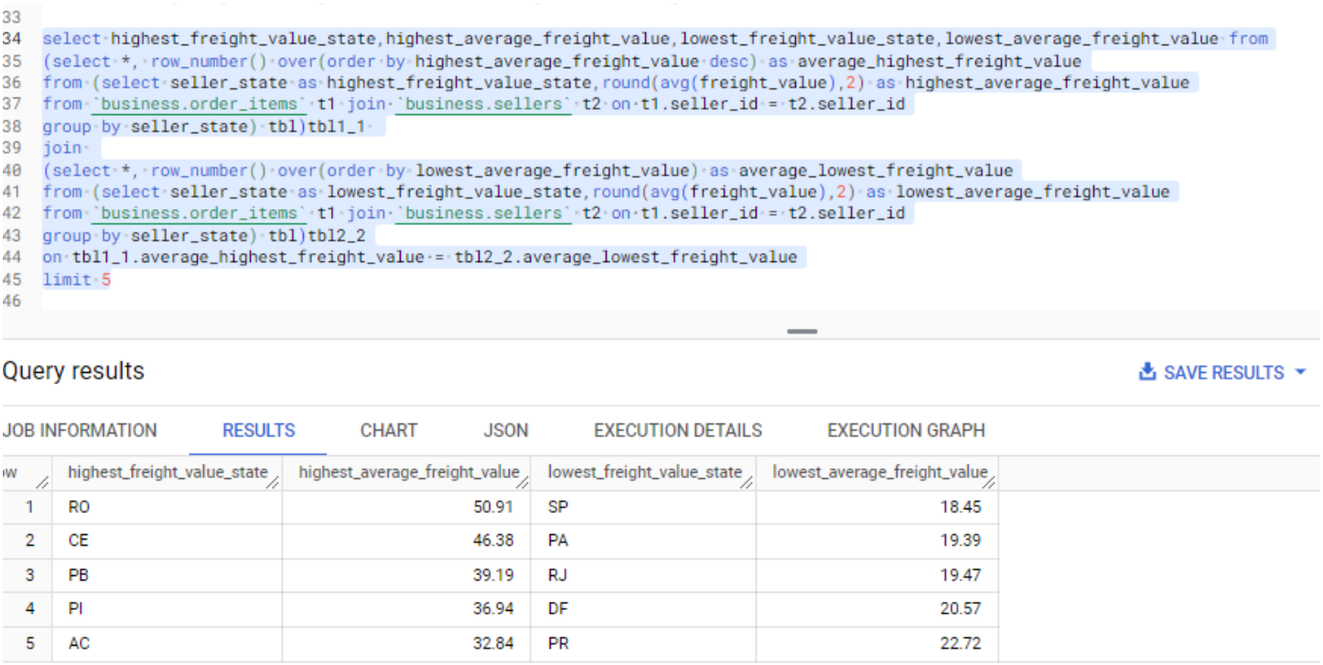
- In first screenshot company took more time to deliver the order. It is important to know why there was a delay in delivering the order.
- After knowing the reason for the delay, we have to overcome these problems.
- In the second screenshot company took less time to deliver the orders. Here we can reduce the estimated delivery time which is more.

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

Query:

```
select
highest_freight_value_state,highest_average_freight_value,lowest_freight_value_state,lowest_average_freight_value from
(select *, row_number() over(order by highest_average_freight_value desc) as average_highest_freight_value
from (select seller_state as highest_freight_value_state,round(avg(freight_value),2) as
highest_average_freight_value
from `business.order_items` t1 join `business.sellers` t2 on t1.seller_id = t2.seller_id
group by seller_state) tbl)tbl1_1
join
(select *, row_number() over(order by lowest_average_freight_value) as average_lowest_freight_value
from (select seller_state as lowest_freight_value_state,round(avg(freight_value),2) as
lowest_average_freight_value
from `business.order_items` t1 join `business.sellers` t2 on t1.seller_id = t2.seller_id
group by seller_state) tbl)tbl2_2
on tbl1_1.average_highest_freight_value = tbl2_2.average_lowest_freight_value
limit 5
```

Screenshot:



Query results

SAVE RESULTS

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

w	highest_freight_value_state	highest_average_freight_value	lowest_freight_value_state	lowest_average_freight_value
1	RO	50.91	SP	18.45
2	CE	46.38	PA	19.39
3	PB	39.19	RJ	19.47
4	PI	36.94	DF	20.57
5	AC	32.84	PR	22.72

Insights:

- RO, CE, PB, PI, AC these states have highest freight value.
- SP, PA, RJ, DF, PR have lowest freight value.

Suggestions:

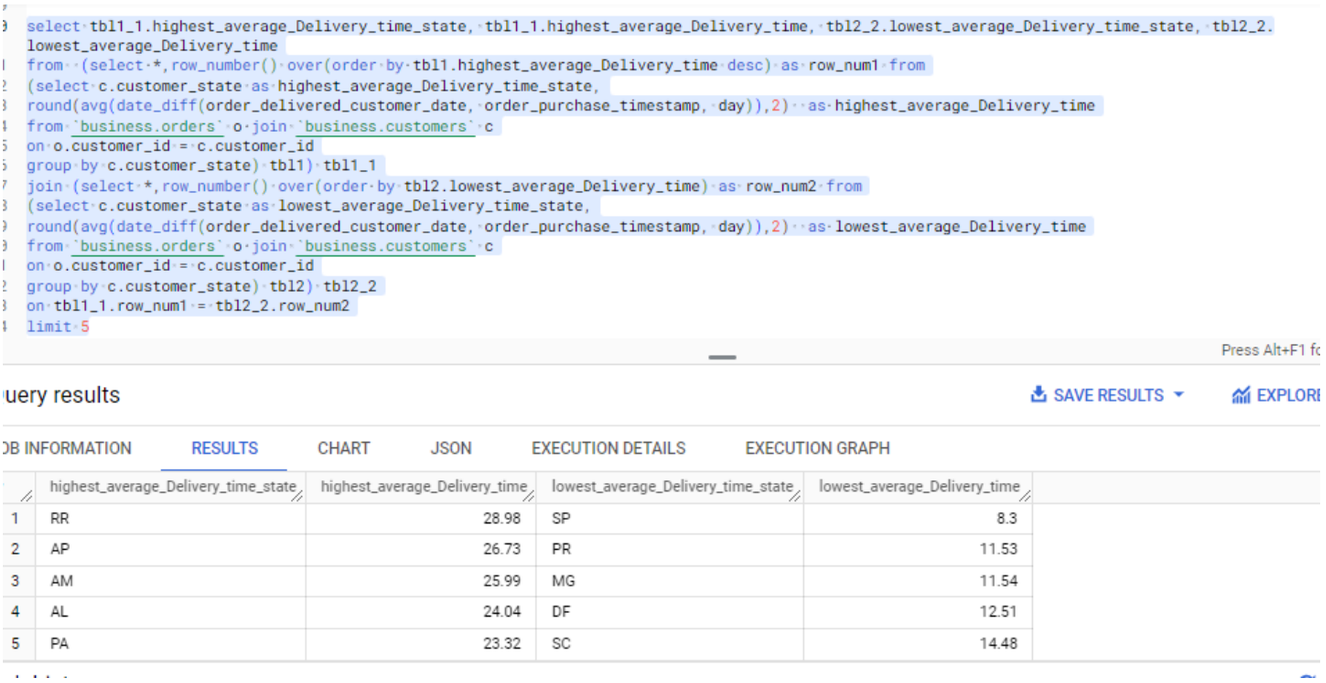
- Since we have freight value for each seller we have to increase the customers. The greater number of customer lesser the average freight value.

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

Query:

```
select tbl1_1.highest_average_Delivery_time_state, tbl1_1.highest_average_Delivery_time,
tbl2_2.lowest_average_Delivery_time_state, tbl2_2.lowest_average_Delivery_time
from  (select *,row_number() over(order by tbl1.highest_average_Delivery_time desc) as row_num1 from
(select c.customer_state as highest_average_Delivery_time_state,
round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)),2)  as
highest_average_Delivery_time
from `business.orders` o join `business.customers` c
on o.customer_id = c.customer_id
group by c.customer_state) tbl1) tbl1_1
join (select *,row_number() over(order by tbl2.lowest_average_Delivery_time) as row_num2 from
(select c.customer_state as lowest_average_Delivery_time_state,
round(avg(date_diff(order_delivered_customer_date, order_purchase_timestamp, day)),2)  as
lowest_average_Delivery_time
from `business.orders` o join `business.customers` c
on o.customer_id = c.customer_id
group by c.customer_state) tbl2) tbl2_2
on tbl1_1.row_num1 = tbl2_2.row_num2
limit 5
```

Screenshot:



	highest_average_Delivery_time_state	highest_average_Delivery_time	lowest_average_Delivery_time_state	lowest_average_Delivery_time
1	RR	28.98	SP	8.3
2	AP	26.73	PR	11.53
3	AM	25.99	MG	11.54
4	AL	24.04	DF	12.51
5	PA	23.32	SC	14.48

Insights:

- Result shows 5 highest average number of days to deliver the items and 5 lowest average number of days to deliver the items.

Suggestions:

- Based on the average number of days to deliver the items in particular state we can keep estimated time to deliver the items.

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

Query:

```
select customer_state, avg(tbl.fastest_delivery) as time_taken_to_deliver
from (select *,
date_diff(o.order_delivered_customer_date,o.order_estimated_delivery_date, day) as fastest_delivery
from `business.orders` o  join `business.customers` c
on o.customer_id = c.customer_id
where o.order_status = 'delivered'
order by fastest_delivery) tbl
group by customer_state
order by time_taken_to_deliver
limit 5
```

Screenshot:

```
select customer_state, avg(tbl.fastest_delivery) as time_taken_to_deliver
from
(select *,
date_diff(o.order_delivered_customer_date,o.order_estimated_delivery_date, day) as fastest_delivery
from `business.orders` o
join `business.customers` c
on o.customer_id = c.customer_id
where o.order_status = 'delivered'
order by fastest_delivery) tbl
group by customer_state
order by time_taken_to_deliver
limit 5
```

Query results

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

low	customer_state	time_taken_to_delive
1	AC	-19.7625000000...
2	RO	-19.1316872427...
3	AP	-18.7313432835...
4	AM	-18.6068965517...
5	RR	-16.4146341463...

Insights:

- Output shows top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
- Time taken value is in negative that means in this state average time taken to deliver the items is very fast.

Suggestions:

- While ordering customers check for estimated delivery date. So in these states where delivery is very fast as compared to estimated delivery date we can reduce the estimated delivery date so that customers will order more if they are getting the item soon.

Q6. Analysis based on the payments:

1. **Find the month_on_month no. of orders placed using different payment types.**

Query:

```
select tbl.order_month, count(order_id) as num_of_orders,payment_type
from (select o.*, FORMAT_DATETIME('%B %Y', o.order_purchase_timestamp) as
order_month,p.payment_type
from `business.orders` o join `business.payments` p
on o.order_id = p.order_id) tbl group by tbl.order_month,payment_type order by num_of_orders desc
```

Screenshot:

```
select tbl.order_month, count(order_id) as num_of_orders,payment_type
from (select o.*, FORMAT_DATETIME('%B %Y', o.order_purchase_timestamp) as order_month,p.payment_type
from `business.orders` o join `business.payments` p
on o.order_id = p.order_id) tbl group by tbl.order_month,payment_type order by num_of_orders desc
```

Query results

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

Row	order_month	num_of_orders	payment_type
1	November 2017	5897	credit_card
2	March 2018	5691	credit_card
3	January 2018	5520	credit_card
4	May 2018	5497	credit_card
5	April 2018	5455	credit_card
6	February 2018	5253	credit_card
7	August 2018	4985	credit_card
8	June 2018	4813	credit_card
9	July 2018	4755	credit_card
10	December 2017	4377	credit_card

Insights:

- Customers used all types of payment modes to order the item.
- Different payment modes were Credit card, debit card, UPI, Voucher.
- From the output we can see that most of the payments are made using credit card followed by UPI, voucher, debit card respectively.
- In November 2017 most of the payments are made using credit card.
- In 2018 from January to August many payments are made using credit card.
- Customers are not using debit cards most

Suggestions:

- Since customers are using credit card and UPI more, we can give some benefits or discounts on them.

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

Query:

```
select payment_installments, count(order_id) as no_of_orders
from `business.payments`
group by payment_installments
order by no_of_orders desc
```

Screenshot:

```
185
186 select payment_installments, count(order_id) as no_of_orders
187 from `business.payments`
188 group by payment_installments
189 having payment_installments >= 1
190 order by no_of_orders desc
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	payment_installment	no_of_orders
1	1	52546
2	2	12413
3	3	10461
4	4	7098
5	10	5328
6	5	5239
7	8	4268
8	6	3920
9	7	1626
10	9	644
11	12	133

```
260
261 select payment_installments, count(order_id) as no_of_orders
262 from `business.payments`
263 group by payment_installments
264 having payment_installments >= 1
265 order by no_of_orders
266
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DE

Row	payment_installment	no_of_orders
1	22	1
2	23	1
3	21	3
4	16	5
5	17	8
6	14	15
7	13	16
8	20	17
9	24	18
10	11	23

Insights:

- Most of the customers used 1 instalment followed by 2 and 3.
- Also, we can see some of the customers have done the payments in 10 instalments.
- Less number of people used higher number of instalments.

Suggestions:

- Since most of the customers are doing the payment in fewer instalments, we can give some benefits or offers or discounts.