BUSINESS CASE – Target SQL

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

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

QUERY :

```
SELECT
column_name , data_type
FROM `dsml-jan-24.Target_SQL.INFORMATION_SCHEMA.COLUMNS`
WHERE Table_name="customers"
```

Output :

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECI

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

(OR)

customers | QUERY | SHARE | COPY | SNAPSHOT

SCHEMA | DETAILS | PREVIEW | LINEAGE | DATA PROFILE | DATA

Filter Enter property name or value

Field name	Type	Mode	Key	Collation
customer_id	STRING	NULLABLE	-	-
customer_unique_id	STRING	NULLABLE	-	-
customer_zip_code_prefix	INTEGER	NULLABLE	-	-
customer_city	STRING	NULLABLE	-	-
customer_state	STRING	NULLABLE	-	-

Insight :

Provides the datatype of each column in the customer table and specifies the columns available in the customer table.

Recommendation :

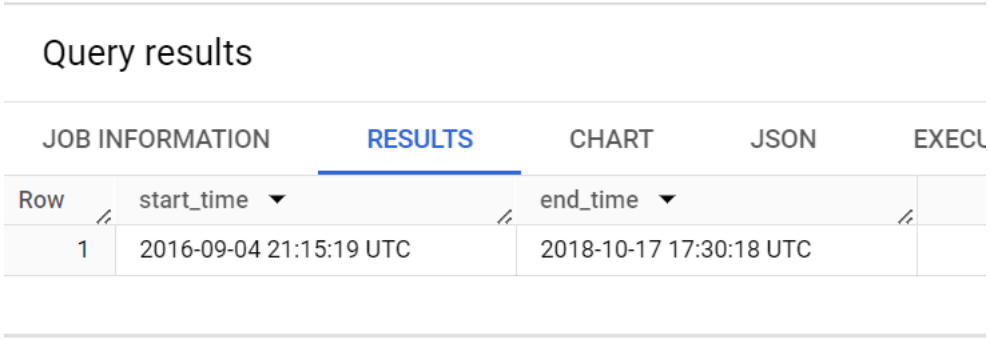
Customer and the few basic details of the customer can be provided to ease the job.

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

QUERY :

```
SELECT
MIN(order_purchase_timestamp) as start_time,
MAX(order_purchase_timestamp) as end_time
FROM `Target_SQL.orders`
```

OUTPUT:



Insight :

From here we get to know that the first order is placed on 2016-09-04 and the last order was on 2018-10-17 for the given set of data.

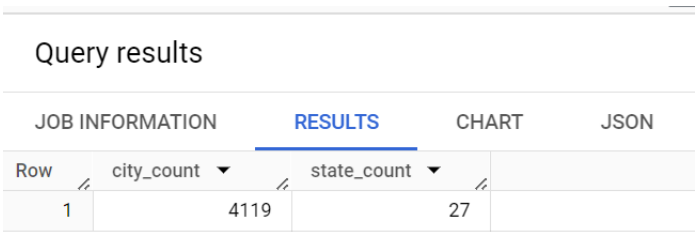
Recommendation :

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` AS c RIGHT JOIN `Target_SQL.orders` AS o
ON c.customer_id = o.customer_id
```

OUTPUT:



Insight :

Orders were placed from 4119 cities which come under 27 states which in turn specifies the orders are placed from all the states in Brazil.

Recommendation :

Slowly business can start expanding to nearby countries .

2. **In-depth Exploration:**

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

QUERY:

```
SELECT
x.Year,
COUNT(DISTINCT x.order_id) AS order_count
FROM (
SELECT
order_id ,
EXTRACT(Year from order_purchase_timestamp) as Year
FROM `Target_SQL.orders`)
AS x
GROUP BY x.Year
ORDER BY x.Year
```

OUTPUT:

Query results

JOB INFORMATION | RESULTS | CHART

Row	Year	order_count
1	2016	329
2	2017	45101
3	2018	54011

Insight :

We can see a drastic change in the number of orders placed for each year which specifies there is growing trend in each in terms of placing orders.

Recommendation :

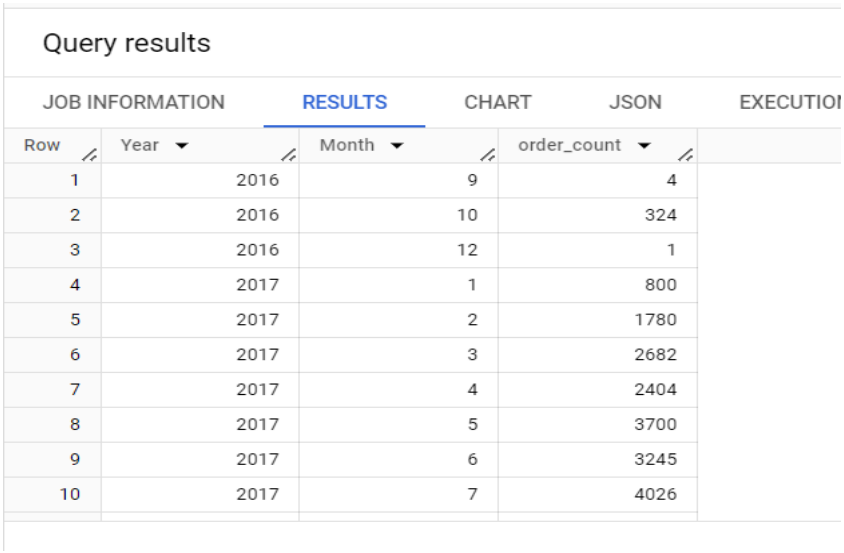
More offers and sales can be brought in to increase the orders getting placed.

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

QUERY:

```
SELECT
x.Year,x.Month,
COUNT(DISTINCT x.order_id) AS order_count
FROM (
SELECT
order_id ,
EXTRACT(Year from order_purchase_timestamp) as Year,
EXTRACT(Month from order_purchase_timestamp) as Month
FROM `Target_SQL.orders`)
AS x
GROUP BY x.Year , x.Month
ORDER BY x.Year , x.Month
```

OUTPUT:



Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTIOI

Row	Year	Month	order_count
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

Insight :

There a lot of changes in the number of orders getting placed for each month as the orders depend on the delas and offers provided. The orders are very high during the yearly offer times , whereas it is low in dry seasons where no offers or sales come.

Recommendation :

Minimal amount of sales or offers can be provided during the dry months like September , October to raise the orders.

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

- 19-23 hrs : Night

QUERY:

```
SELECT
COUNT(CASE
  WHEN FORMAT_DATETIME("%H",order_purchase_timestamp) BETWEEN "00" AND "06"
  THEN order_id END ) AS Dawn,
COUNT(CASE
  WHEN FORMAT_DATETIME("%H",order_purchase_timestamp) BETWEEN "07" AND "12"
  THEN order_id END ) AS Mornings,
COUNT(CASE
  WHEN FORMAT_DATETIME("%H",order_purchase_timestamp) BETWEEN "13" AND "18"
  THEN order_id END ) AS Afternoon,
COUNT(CASE
  WHEN FORMAT_DATETIME("%H",order_purchase_timestamp) BETWEEN "19" AND "24"
  THEN order_id END ) AS Night
FROM `Target_SQL.orders`;
```

OUTPUT:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	Dawn	Mornings	Afternoon	Night
1	5242	27733	38135	28331

Insight :

Most of the orders are getting placed in afternoon.

Recommendation :

More deals and offers can be provided during the Dawn , Mornings and Night time to attract the customers which in turn helps in ordering the product.

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

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

QUERY:

```
SELECT
x.Year,x.Month,c.customer_state,
IFNULL(COUNT(DISTINCT x.order_id),0) AS order_count
FROM (
SELECT
customer_id,
order_id ,
EXTRACT(Year from order_purchase_timestamp) as Year,
```

```
EXTRACT(Month from order_purchase_timestamp) as Month
FROM `Target_SQL.orders`)
AS x LEFT JOIN `Target_SQL.customers` as c
ON x.customer_id = c.customer_id
GROUP BY x.Year , x.Month,c.customer_state
ORDER BY x.Year , x.Month;
```

OUTPUT:

Query results

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

Row	Year	Month	customer_state	order_count
1	2016	9	RR	1
2	2016	9	RS	1
3	2016	9	SP	2
4	2016	10	SP	113
5	2016	10	RS	24
6	2016	10	RJ	56
7	2016	10	MT	3
8	2016	10	GO	9
9	2016	10	MG	40
10	2016	10	CE	8
11	2016	10	SC	11

Insight :

It clearly specifies that state SP always places the higher number of orders compared to other states every month.

Recommendation :

As per the requirement of products , they can be made in stock so that the product can be delivered easily .

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

QUERY:

```
SELECT
customer_state,
COUNT(DISTINCT customer_id) AS No_Of_Customers
FROM `Target_SQL.customers`
GROUP BY customer_state
ORDER BY No_Of_Customers DESC
```

OUTPUT:

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	No_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

Insight :

Most of the customers are from the State SP.

Recommendation :

In State SP more hubs , and delivery partners can be increased to have a good ordering experience which in turn motivates to order more.

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

1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only).
   You can use the "payment_value" column in the payments table to get the cost of orders.

QUERY:

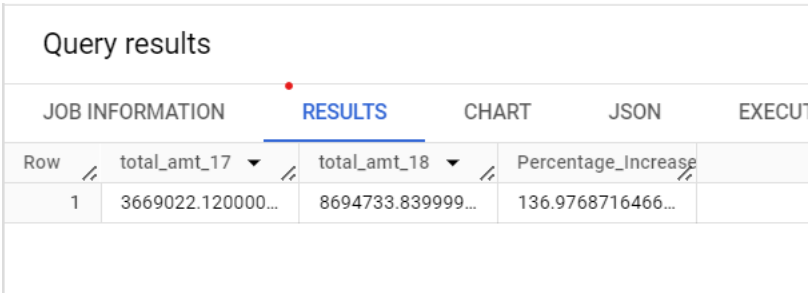
```
WITH x1 AS(
SELECT
EXTRACT(Year FROM o.order_purchase_timestamp ) as year,
EXTRACT(Month FROM o.order_purchase_timestamp ) as month,
SUM(p.payment_value) as amt
FROM `Target_SQL.orders` AS o JOIN `Target_SQL.payments` AS p
ON o.order_id = p.order_id
GROUP BY EXTRACT(Year FROM o.order_purchase_timestamp ),EXTRACT(Month FROM
o.order_purchase_timestamp )),

x2 AS(
SELECT
year , month,
SUM(CASE WHEN year=2017 THEN amt ELSE 0 END) AS amt_17,
SUM(CASE WHEN year=2018 THEN amt ELSE 0 END) AS amt_18
FROM x1
```

```
WHERE month BETWEEN 1 AND 8
GROUP BY year,month
ORDER BY year )

SELECT
SUM(amt_17) AS total_amt_17,
SUM(amt_18) AS total_amt_18,
((SUM(amt_18) - SUM(amt_17)) / SUM(amt_17)) *100 AS Percentage_Increase
FROM x2;
```

OUTPUT:



Insight :

The ordering percentage from 2017 to 2018 is at 136.98 % which indicates the number of orders has increased in good number from 2017 to 2018.

Recommendation :

More products , hubs , working staffs and more variety of products with increased sales and offers can be brought in to increase the order % year by year.

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

QUERY :

```
SELECT
c.customer_state ,
SUM(oi.price) AS Total_Price,
AVG(oi.price) AS Average_Price
FROM `Target_SQL.customers` AS c LEFT JOIN `Target_SQL.orders` AS o
ON c.customer_id = o.customer_id
LEFT JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state
```

OUTPUT:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETA

Row	customer_state	Total_Price	Average_Price
1	AC	15982.94999999...	173.7277173913...
2	AL	80314.81000000...	180.8892117117...
3	AM	22356.84000000...	135.4959999999...
4	AP	13474.29999999...	164.3207317073...
5	BA	511349.9900000...	134.6012082126...
6	CE	227254.7099999...	153.7582611637...
7	DF	302603.9399999...	125.7705486284...
8	ES	275037.3099999...	121.9137012411...
9	GO	294591.9499999...	126.2717316759...
10	MA	119648.2199999...	145.2041504854...
11	MG	1585308.029999...	120.7485741488...

Insight :

We get to know the total price of the orders and their average for each state , from which we can infer that State SP provides the highest order values.

Recommendation :

Can provide more offers and vouchers for other states to increase the orders.

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

QUERY:

```
SELECT
c.customer_state ,
SUM(oi.freight_value) AS Total_Freight_Price,
AVG(oi.freight_value) AS Average_Freight_Price
FROM `Target_SQL.customers` AS c LEFT JOIN `Target_SQL.orders` AS o
ON c.customer_id = o.customer_id
LEFT JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state
```

OUTPUT:

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAIL

Row	customer_state	Total_Freight_Price	Average_Freight_Pric
1	AC	3686.749999999...	40.07336956521...
2	AL	15914.58999999...	35.84367117117...
3	AM	5478.889999999...	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...

Insight :

State RR has highest average freight value which indicates that most of the products are getting transported and SP has least average value.

Recommendation :

Can create more hubs with more stocks of frequently brought items to avoid transportation of items.

5.Analysis based on sales, freight and delivery time.

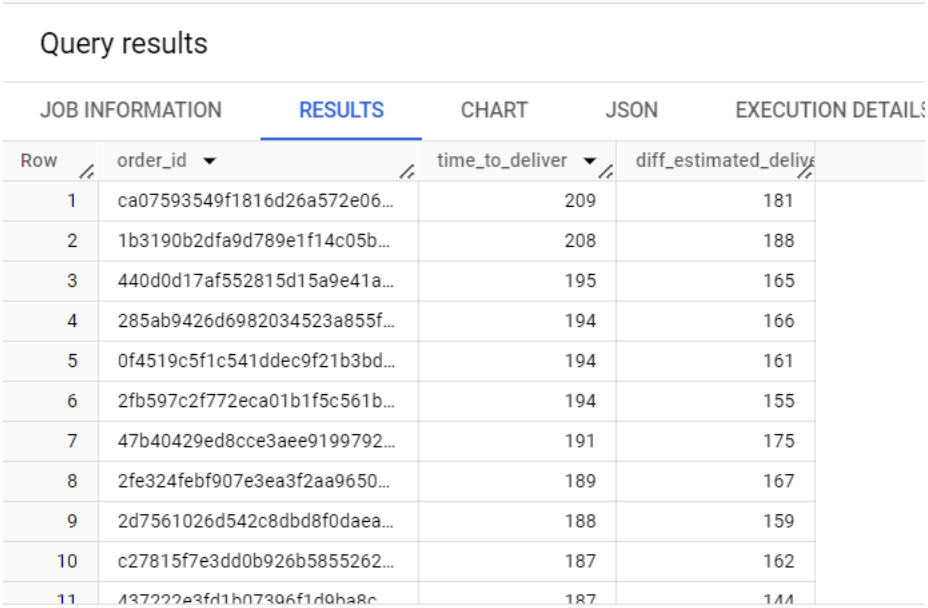
1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time.
   Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
   Do this in a single query.
   You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula:
   - **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
   - **diff_estimated_delivery** = order_delivered_customer_date - order_estimated_delivery_date

QUERY:

```
SELECT
order_id,
IFNULL(DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,Day),0) AS time_to_deliver,
```

```
IFNULL(DATE_DIFF(order_delivered_customer_date,order_estimated_delivery_date,Day),0) AS
diff_estimated_delivery
FROM `Target_SQL.orders`
ORDER BY time_to_deliver DESC ,diff_estimated_delivery DESC
```

OUTPUT:



Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	order_id	time_to_deliver	diff_estimated_delive
1	ca07593549f1816d26a572e06...	209	181
2	1b3190b2dfa9d789e1f14c05b...	208	188
3	440d0d17af552815d15a9e41a...	195	165
4	285ab9426d6982034523a855f...	194	166
5	0f4519c5f1c541ddec9f21b3bd...	194	161
6	2fb597c2f772eca01b1f5c561b...	194	155
7	47b40429ed8cce3aee9199792...	191	175
8	2fe324febf907e3ea3f2aa9650...	189	167
9	2d7561026d542c8dbd8f0daea...	188	159
10	c27815f7e3dd0b926b5855262...	187	162
11	437222e3fd1b07396f1d9ba8c...	187	144

Insight :

Provides the days taken to deliver a product in comparison with the estimated delivery date.

Recommendation :

There are states which delivered ahead of the estimated delivery date, where the stock and transportation needs to be improved.

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

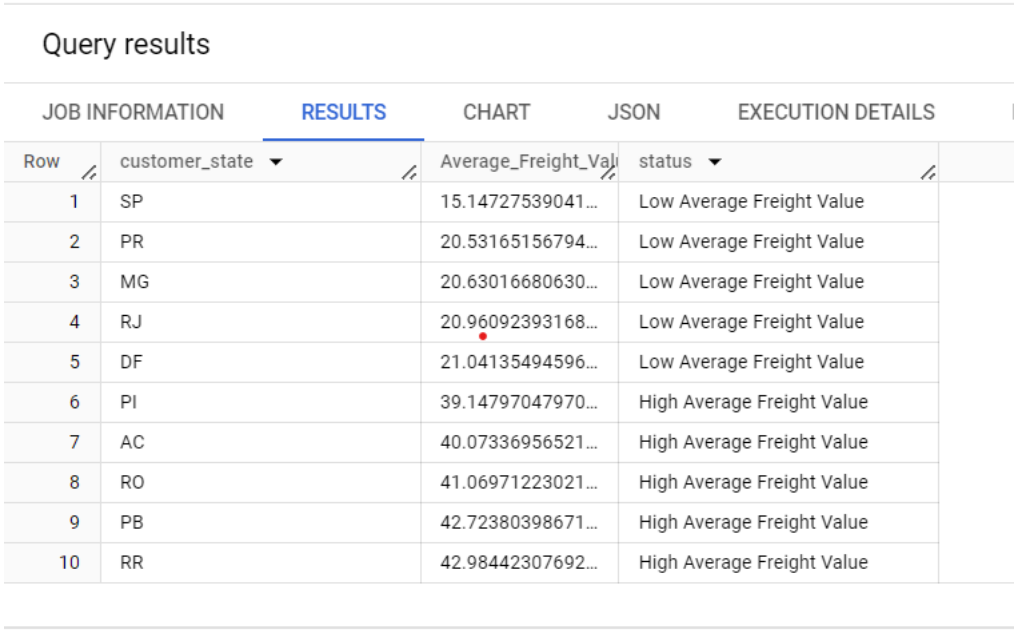
QUERY :

```
WITH avg_fre AS (
SELECT
c.customer_state ,
AVG(oi.freight_value) AS Average_Freight_Price
FROM `Target_SQL.customers` AS c JOIN `Target_SQL.orders` AS o
ON c.customer_id = o.customer_id
JOIN `Target_SQL.order_items` AS oi
ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY c.customer_state ),
ra_fre_low AS (
```

```
  SELECT customer_state,
  avg_fre.Average_Freight_Price,
  DENSE_RANK() OVER(ORDER BY Average_Freight_Price) AS ranking
  FROM avg_fre
  ORDER BY ranking
),
ra_fre_high AS (
  SELECT customer_state,
  avg_fre.Average_Freight_Price,
  DENSE_RANK() OVER(ORDER BY Average_Freight_Price DESC) AS ranking
  FROM avg_fre
  ORDER BY ranking
)

SELECT
customer_state,
ra_fre_low.Average_Freight_Price AS Average_Freight_Value,
"Low Average Freight Value" AS status
FROM ra_fre_low
WHERE ranking<= 5
union all
SELECT
customer_state,
ra_fre_high.Average_Freight_Price as Average_Freight_Value,
"High Average Freight Value" AS status
FROM ra_fre_high
WHERE ranking<= 5
order by Average_Freight_Value
```

OUTPUT :



Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	Average_Freight_Val	status
1	SP	15.14727539041...	Low Average Freight Value
2	PR	20.53165156794...	Low Average Freight Value
3	MG	20.63016680630...	Low Average Freight Value
4	RJ	20.96092393168...	Low Average Freight Value
5	DF	21.04135494596...	Low Average Freight Value
6	PI	39.14797047970...	High Average Freight Value
7	AC	40.07336956521...	High Average Freight Value
8	RO	41.06971223021...	High Average Freight Value
9	PB	42.72380398671...	High Average Freight Value
10	RR	42.98442307692...	High Average Freight Value

Insight :

State RR has more freight value where as SP has least freight value.

Recommendation :

The transportation facility and the hub in stock can be reduced to avoid the freight of goods.

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

QUERY :

```
WITH delivery AS (
SELECT
c.customer_state ,
AVG(IFNULL(DATE_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,Day),0)) AS
avg_time_to_deliver
FROM `Target_SQL.customers` AS c JOIN `Target_SQL.orders` AS o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state ),
delivery_ra_low AS (
  SELECT customer_state,
  delivery.avg_time_to_deliver,
  DENSE_RANK() OVER(ORDER BY delivery.avg_time_to_deliver) AS ranking
  FROM delivery
  ORDER BY ranking
) ,
delivery_ra_high AS (
  SELECT customer_state,
  delivery.avg_time_to_deliver,
  DENSE_RANK() OVER(ORDER BY delivery.avg_time_to_deliver DESC) AS ranking
  FROM delivery
  ORDER BY ranking
)

SELECT
customer_state,
delivery_ra_low.avg_time_to_deliver AS avg_delivery_time,
"Low Average Delivery Time" AS status
FROM delivery_ra_low
WHERE ranking<=5
union all
SELECT
customer_state,
delivery_ra_high.avg_time_to_deliver AS avg_delivery_time,
"High Average Delivery Time" AS status
FROM delivery_ra_high
WHERE ranking<=5
ORDER BY avg_delivery_time
```

OUTPUT :

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	customer_state	avg_delivery_time	status
1	SP	8.049393953911...	Low Average Delivery Time
2	PR	11.24796828543...	Low Average Delivery Time
3	MG	11.26600773528...	Low Average Delivery Time
4	DF	12.15841121495...	Low Average Delivery Time
5	SC	14.12125378058...	Low Average Delivery Time
6	PA	22.62256410256...	High Average Delivery Time
7	AL	23.10895883777...	High Average Delivery Time
8	AM	25.45945945945...	High Average Delivery Time
9	RR	25.82608695652...	High Average Delivery Time
10	AP	26.33823529411...	High Average Delivery Time

Insight :

State AP has taken more time to deliver a product where as SP has taken least time to deliver in comparison with the estimated delivery date.

Recommendation :

As per the stock available and the demand in the region the delivery time can be given as per that.

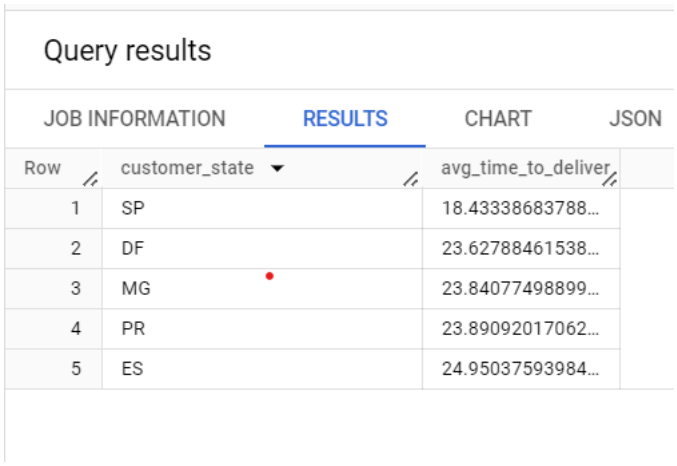
4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
   You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.

QUERY :

```
WITH delivery AS (
SELECT
c.customer_state ,
avg(DATETIME_DIFF(o.order_delivered_customer_date,o.order_purchase_timestamp,day)) -
avg(datetime_diff(o.order_delivered_customer_date,o.order_estimated_delivery_date,day)) AS
avg_time_to_deliver
FROM `Target_SQL.customers` AS c JOIN `Target_SQL.orders` AS o
ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY c.customer_state ),
delivery_ra AS (
  SELECT customer_state,
  delivery.avg_time_to_deliver,
  DENSE_RANK() OVER(ORDER BY delivery.avg_time_to_deliver) AS ranking
  FROM delivery
  ORDER BY ranking
```

```
)
SELECT
customer_state,
delivery_ra.avg_time_to_deliver
FROM delivery_ra
WHERE ranking<=5;
```

OUTPUT :



Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	customer_state	avg_time_to_deliver
1	SP	18.43338683788...
2	DF	23.62788461538...
3	MG	23.84077498899...
4	PR	23.89092017062...
5	ES	24.95037593984...

Insight :

In the state AC the orders are getting delivered fast as compared to the estimated delivery date.

Recommendation :

Can increase the stock and delivery agent functionality to increase the order delivery speed.

6.Analysis based on the payments:

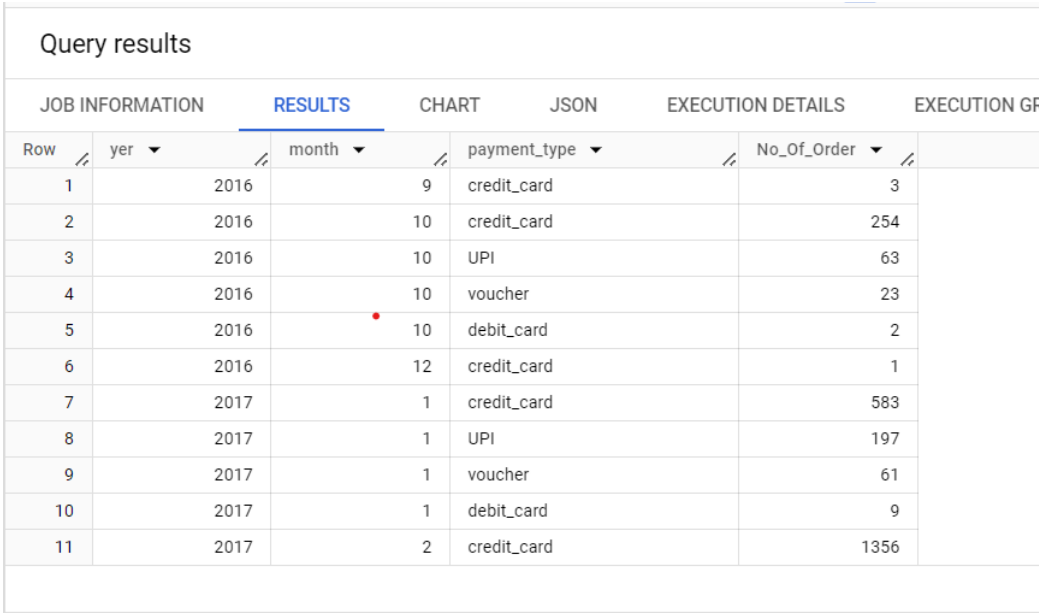
1. Find the month on month no. of orders placed using different payment types.

QUERY:

```
SELECT
x.yer,
x.month,
x.payment_type,
COUNT(x.order_id) AS No_Of_Order
FROM (
SELECT
o.order_id,
EXTRACT(Year From o.order_purchase_timestamp) AS yer,
EXTRACT(Month From o.order_purchase_timestamp) AS month,
p.payment_type
FROM `Target_SQL.orders` AS o JOIN `Target_SQL.payments` AS p
ON o.order_id = p.order_id ) AS x
```

```
GROUP BY x.yer , x.month, x.payment_type
ORDER BY x.yer,x.month
```

OUTPUT:



Query results

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

Row	yer	month	payment_type	No_Of_Order
1	2016	9	credit_card	3
2	2016	10	credit_card	254
3	2016	10	UPI	63
4	2016	10	voucher	23
5	2016	10	debit_card	2
6	2016	12	credit_card	1
7	2017	1	credit_card	583
8	2017	1	UPI	197
9	2017	1	voucher	61
10	2017	1	debit_card	9
11	2017	2	credit_card	1356

Insight :

Most of the orders are processed with credit card for every month.

Recommendation :

Can increase more offers for debit cards as well other payment methods to encourage the orders getting placed.

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

QUERY:

```
SELECT
payment_installments,
COUNT(DISTINCT order_id) AS No_Of_Order
FROM `Target_SQL.payments`
WHERE payment_installments > 0
GROUP BY payment_installments
ORDER BY payment_installments
```

OUTPUT:

Query results

JOB INFORMATION | RESULTS | CHART

Row	payment_installment	No_Of_Order
1	1	49060
2	2	12389
3	3	10443
4	4	7088
5	5	5234
6	6	3916
7	7	1623
8	8	4253
9	9	644
10	10	5315

Insight :

The data provides the information on the number of orders that got processed with each number of instalments. We get to know that most of the orders were placed with 1 instalment which indicates that the order is placed and paid at the same time instead of EMIs.

Recommendation :

More coupons and vouchers can be provided for customers with 1 instalment to encourage their shopping which in turn increases the company revenue.