What does 'good' look like?

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

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

Code:

```
Select column_name, data_type
from `target.INFORMATION_SCHEMA.COLUMNS`
where table_name='customers';
```

Output:

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:

The above query will help us to fetch the details of a table like what are the columns present and also to identify the data type associated with it.

Customer_id, typically a unique identifier that is assigned to each individual customer within the database. It basically help's us to keep track of customer information and order related.

Customer_unique_id refers to more unique or specific identifier that is assigned to a customer. This identifies the unique id from one customer to another in a more detailed format.

Whereas, the other columns which are customer_zip_code_prefix, customer_city and customer_state gives us the details of the customer address.

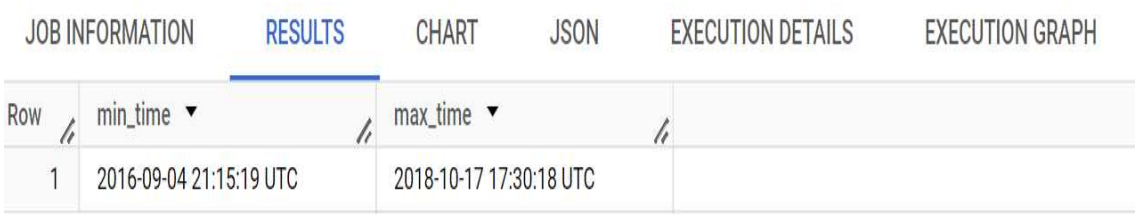
1.2 Get the time range between which the orders were placed

Code:

```
select min(order_purchase_timestamp) as min_time,
max(order_purchase_timestamp) as max_time
from `target.orders`;
```

```
select * from `target.orders`;

select min(order_purchase_timestamp) as min_time,
       max(order_purchase_timestamp) as max_time
from `target.orders`;
```

Output:



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

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

Insights:

In order to get the time range between the orders placed, min_time for the first ever order that has been placed and the max_time for the last order. Time period ranges approximately 2 years for the given data.

In other words, the time range can be calculated to identify the peak periods, optimize the seasonal management, design market strategies appropriately, promotions, analyze the customer patterns/preferences, trends, feedbacks, sales growth and customer reviews.

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

Code:

```
with locate as (
select geolocation_city as city,geolocation_state as state
from `target.geolocation`
union all
select customer_city as city, customer_state as state
from `target.customers`
union all
select seller_city as city,seller_state as state
from `target.sellers`)
select count (distinct city) as city,count(distinct state)
from locate ;
```

```sql
select * from `target.geolocation`;
select * from `target.customers`;
select * from `target.sellers`;

with locate as(
  select geolocation_city as city,geolocation_state as state
  from `target.geolocation`
  union all
  select customer_city as city, customer_state as state
  from `target.customers`
  union all
  select seller_city as city,seller_state as state
  from `target.sellers`
)
 select count(distinct city) as city,count(distinct state)
 from locate
```

Output:

JOB INFORMATION | RESULTS | CHART

Row	city	f0_
1	8126	27

Insights:

From this data, we can learn that Brazilian market has established a significant presence or strong foothold across various regions, widespread customer base spanning in the entire country

We can infer the count of unique number of cities and states in the geolocation has missed around (8126-8011) i.e. 115 that are mentioned in the whole dataset

The count of city and state offers the valuable information to identify potential opportunities customers based on locations and behavioral pattern, facilitate the decision-making process, conducting geographical analysis and strategic planning

2. In-depth Exploration

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

Code:

with years as (

select extract(year from order_purchase_timestamp) as year,

count(*) as order_no,

from `target.orders`

where order_status!='canceled'

group by 1

order by 1)

select *,

concat(round((order_no)/lagg)*100,1,'%') as growth_trend_years

from (

select *,

lag(order_no) over (order by year) as lagg

from years

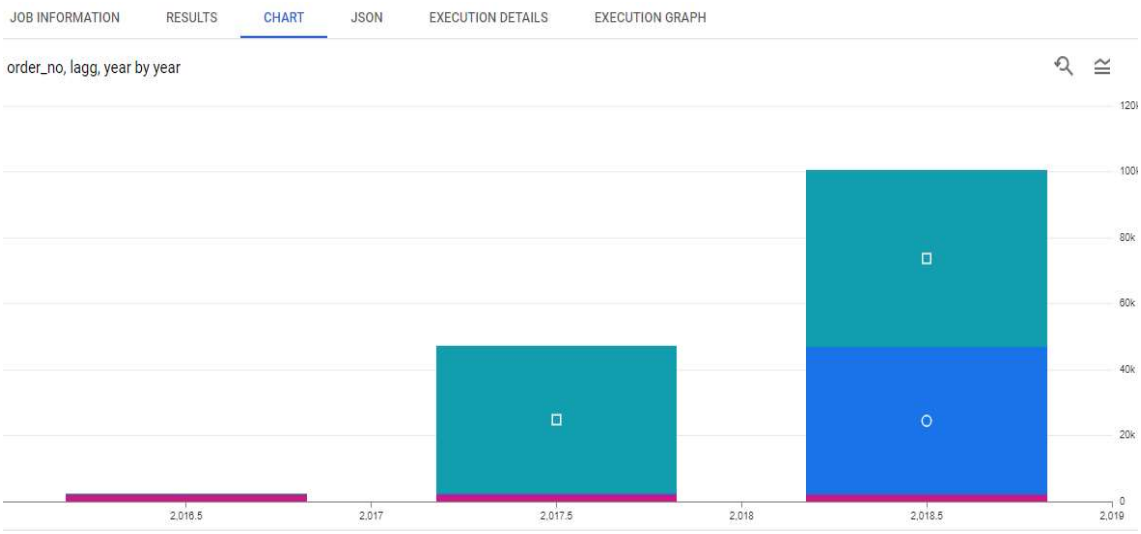
order by 1) ;

```
select * from `target.orders`;
#order_status:shipped,delivered,invoiced,unavailable,processing,created
with years as(
  select extract(year from order_purchase_timestamp) as year,
  count(*) as order_no,
  from `target.orders`
  where order_status!='canceled'
  group by 1
  order by 1
)
select *,
concat(round((order_no)/lagg)*100,1,'%') as growth_trend_years
from (
  select *,
  lag(order_no) over (order by year) as lagg
  from years
  order by 1
)
```

Output:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	year	order_no	lagg	growth_trend_years
1	2016	303	*null*	*null*
2	2017	44836	303	148001%
3	2018	53677	44836	1001%



Insights:

In this growth_trend_years column indicating the order growth rate compared to the previous year, it is evident that there was a substantial order of growth rate in 2018.

In general, it shows a consistent and significance upward trend in the number of orders placed. In Brazil region, base on revenue there is a leap trend is observed which indicates the positive business performance and increasing customer demand to gain revenue uptrend.

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

Code:

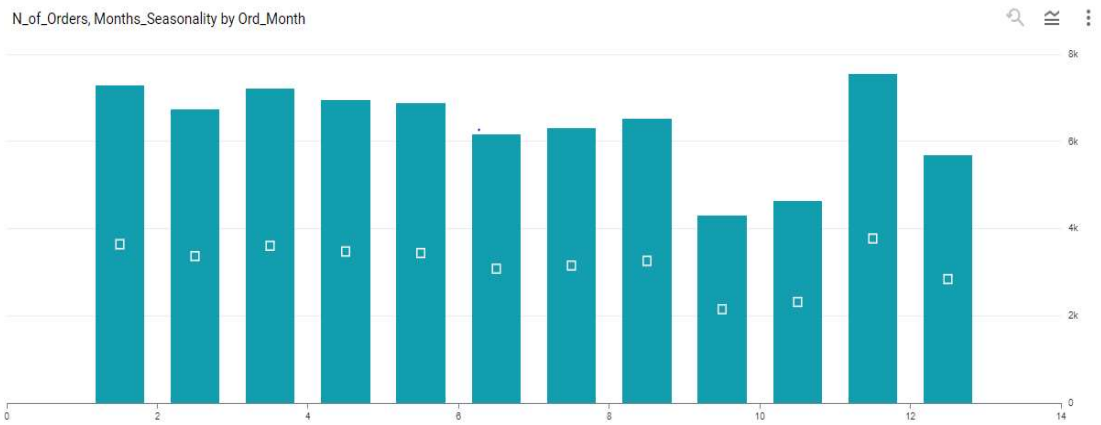
```
select Ord_year,

Ord_Month,N_of_Orders,

ntile (5) over (order by N_of_orders desc) as Months_Seasonality

from (

SELECT EXTRACT (YEAR FROM order_purchase_timestamp) AS Ord_Year,

EXTRACT (MONTH FROM order_purchase_timestamp) AS Ord_Month,

COUNT (order_id) AS N_of_Orders

FROM `target.orders`

GROUP BY Ord_year, Ord_month

ORDER BY Ord_year, Ord_month) sea

ORDER BY N_of_Orders desc, Ord_year asc;
```

```sql
select Ord_year,Ord_Month,N_of_Orders,
 ntile(5) over(order by N_of_orders desc) as Months_Seasonality
  from (SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS Ord_Year,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS Ord_Month,
  COUNT(order_id) AS N_of_Orders
   FROM  `target.orders`
   GROUP BY ord_year,ord_month
   ORDER BY ord_year,ord_month) sea
   ORDER BY N_of_Orders desc,Ord_year asc;
```

Output:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	Ord_year	Ord_Month	N_of_Orders	Months_Seasonality
1	2017	11	7544	1
2	2018	1	7269	1
3	2018	3	7211	1
4	2018	4	6939	1
5	2018	5	6873	1
6	2018	2	6728	2
7	2018	8	6512	2
8	2018	7	6292	2
9	2018	6	6167	2
10	2017	12	5673	2
11	2017	10	4631	3
12	2017	8	4331	3
13	2017	9	4285	3
14	2017	7	4026	3



Insights:

There is a clearly monthly seasonal pattern in the number of orders, with increases and decreases throughout the year.

This query wants to calculate the total number of orders for each month over the years. We can infer that Peak is at November 2017 shows the highest number of orders, while September 2016 has the lowest number of orders.

Comparing the number of orders across different years reveals growth and trough in number of orders. There is significant growth from 2016 to 2017, but a decrease in average orders in 2018 compared to the previous year.

2.3 During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

1. **.0-6 hrs: Dawn**
2. **7-12 hrs: Mornings**
3. **13-18 hrs: Afternoon**
4. **19-23 hrs: Night**

Code:

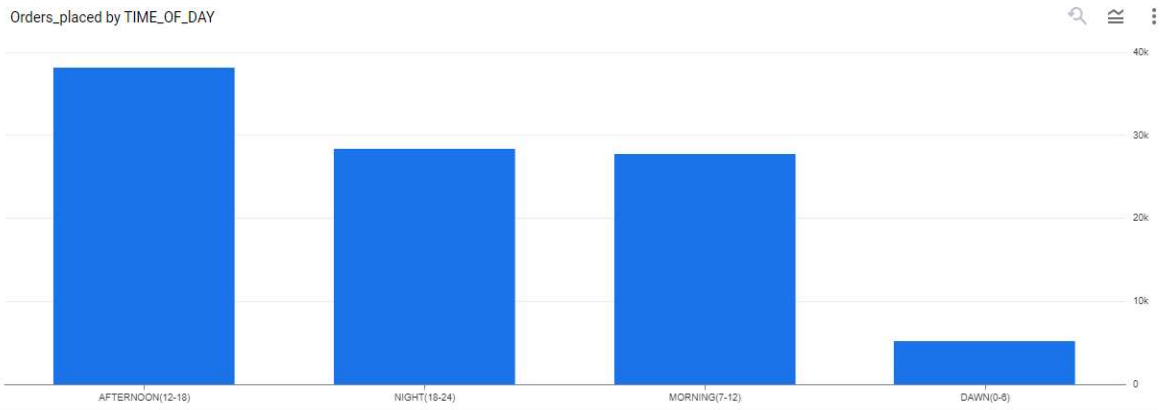
```
with T_DAY as
(
Select
Case
when extract(hour from order_purchase_timestamp)
between 0 and 6 then 'DAWN(0-6)'
when extract(hour from order_purchase_timestamp)
between 7 and 12 then 'MORNING(7-12)'
when extract(hour from order_purchase_timestamp)
between 13 and 18 then 'AFTERNOON(12-18)'
else 'NIGHT(18-24)'
end as TIME_OF_DAY,
count(*)as Orders_placed
from `target.orders`
group by TIME_OF_DAY
)
select * from T_DAY
order by 2 desc;
```

```sql
with T_DAY as
(
select
case
when extract(hour from order_purchase_timestamp)
between 0 and 6 then 'DAWN(0-6)'
when extract(hour from order_purchase_timestamp)
between 7 and 12 then 'MORNING(7-12)'
when extract(hour from order_purchase_timestamp)
between 13 and 18 then 'AFTERNOON(12-18)'
else 'NIGHT(18-24)'
end as TIME_OF_DAY,
count(*)as Orders_placed
from `target.orders`
group by TIME_OF_DAY
)
select * from T_DAY
order by 2 desc;
```

Output:

JOB INFORMATION | RESULTS | CHART | J

Row	TIME_OF_DAY	Orders_placed
1	AFTERNOON(12-18)	38135
2	NIGHT(18-24)	28331
3	MORNING(7-12)	27733
4	DAWN(0-6)	5242



Insights:

Based on the output, Brazilian customers tend to place the highest number of orders during the afternoon hours, followed by the night hours. Based on the output, Brazilian customers tend to place the highest number of orders during the afternoon hours, followed by the night hours.

This data provides us that customers are actively engaged in online shopping during their leisure time (afternoon, evening) with a significant number of orders placed in the morning as well

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

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

Code:

```
select yr_mnth, customer_state,
min(mon_mon_orders) over (partition by yr_mnth) as Min_orders_recd,
max(mon_mon_orders) over (partition by yr_mnth) as Max_orders_recd,
round(avg(mon_mon_orders) over (partition by customer_state order by yr_mnth),3) as
Avg_monthly_order,
mon_mon_orders as Total_ordersPerMonth,
sum(mon_mon_orders) over (partition by customer_state) as Monthly_state_order
from (select customer_state,
format_timestamp("%Y-%m",order_purchase_timestamp) as yr_mnth,
count (*) as mon_mon_orders
from `target.customers` c
join `target.orders` o
on c. customer_id= o.customer_id
where order_status not in ('canceled', 'unavailable')
group by customer_state, yr_mnth
order by customer_state, yr_mnth
)
group by mon_mon_orders, yr_mnth,customer_state
order by customer_state;
```

```
select yr_mnth,customer_state,
min(mon_mon_orders) over(partition by yr_mnth)as Min_orders_recd,
max(mon_mon_orders) over(partition by yr_mnth) as Max_orders_recd,
round(avg(mon_mon_orders) over(partition by customer_state order by yr_mnth),3)as Avg_monthly_order,
mon_mon_orders as Total_ordersPerMonth,
sum(mon_mon_orders) over(partition by customer_state)as Monthly_state_order
from
(
select customer_state,        *
format_timestamp("%Y-%m",order_purchase_timestamp) as yr_mnth,
count(*)as mon_mon_orders
from `target.customers` c
join `target.orders` *o
on c.customer_id=o.customer_id
where order_status not in ('canceled','unavailable')
group by customer_state,yr_mnth
order by customer_state,yr_mnth
)
group by mon_mon_orders,yr_mnth,customer_state
order by customer_state ;
```

Output:

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

Row	yr_mnth	customer_state	Min_orders_recd	Max_orders_recd	Avg_monthly_order	Total_ordersPerMont	Monthly_state_orde
1	2017-01	AC	1	294	2.0	2	81
2	2017-02	AC	2	630	2.5	3	81
3	2017-03	AC	2	991	2.333	2	81
4	2017-04	AC	2	895	3.0	5	81
5	2017-05	AC	2	1396	4.0	8	81
6	2017-06	AC	1	1308	4.0	4	81
7	2017-07	AC	1	1570	4.143	5	81
8	2017-08	AC	3	1698	4.125	4	81
9	2017-09	AC	1	1605	4.222	5	81
10	2017-10	AC	3	1754	4.4	6	81

Insights:

In this, can clearly observe month on month performance of each ,and also the minimum and maximum number of order values which can help management to set targets and plan operations accordingly.

This information allows prioritization of geolocations and targeted strategies to enhance orders in specific states. The state-wise average order counts highlights the orders with steady or increasing orders.

3.2 How are the customers distributed across all the states?

Code:

```
select customer_state, count(customer_unique_id) as Number_of_customers
from `target.customers`
group by customer_state
order by Number_of_customers desc
limit 10;
```

Output:

JOB INFORMATION | RESULTS | CHART | JS

Row	customer_state	Number_of_custome
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

Insights:

States like SP and RJ with a high count of customers indicate potential market hotspots.

States with a higher number of regular customers suggests reflects higher customer likability's or satisfaction.

State with lower customer counts such RR and AP with only 46 and 68 customers.

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

4.1. 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.

Code:

```
with year17 as
(select round (sum (p. payment_value)) as order_cost_2k17
from `target.orders` o
join `target.payments` p
on p. order_id=o. order_id
where
extract (year from o. order_purchase_timestamp) = 2017 and
extract (month from o. order_purchase_timestamp) between 1 and 8),
```

year18 as (

select round (sum (p. payment_value)) as order_cost_2k18

from `target.orders` o

join `target.payments` p

on p. order_id=o. order_id

where

extract (year from o. order_purchase_timestamp) = 2018 and

extract (month from o. order_purchase_timestamp) between 1 and 8)

select year17.order_cost_2k17, year18.order_cost_2k18,

concat(round(((year18.order_cost_2k18-
year17.order_cost_2k17)/year17.order_cost_2k17) *100),'%') as Percent_increase

from year17, year18;

```
with year17 as
(select round(sum(p.payment_value)) as order_cost_2k17
 from `target.orders` o
 join `target.payments` p
 on p.order_id=o.order_id
  where
   extract (year from o.order_purchase_timestamp)= 2017 and
   extract (month from o.order_purchase_timestamp) between 1 and 8 ),

year18 as (
  select round(sum(p.payment_value)) as order_cost_2k18
  from `target.orders` o
  join `target.payments` p
  on p.order_id=o.order_id
  where
   extract (year from o.order_purchase_timestamp)= 2018 and
   extract (month from o.order_purchase_timestamp) between 1 and 8 )

select year17.order_cost_2k17,year18.order_cost_2k18,
concat(round(((year18.order_cost_2k18-year17.order_cost_2k17)/year17.order_cost_2k17)*100),'%') as Percent_increase
from year17,year18;
```

Output:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

Row	order_cost_2k17	order_cost_2k18	Percent_increase
1	3669022.0	8694734.0	137%

Insights:

The data shows a positive trend of increasing order amounts from 2017 to 2018, a substantial decrease in cost of the products makes us understand the economic growth of the country indicating potential growth and success in the business across Brazilian states. The value of orders increased significantly and shows the potential business growth.

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

Code:

```
select customer_state,
concat (cast (sum (oi. price) as INT), ' Brazilian_reals') as Total_amount,
concat (cast (avg (oi. price) as INT), ' Brazilian_reals') as Avg_amt
from `target.customers` c
join `target.orders` o
on c. customer_id=o. customer_id
join `target.order_items` oi
on oi. order_id = o. order_id
group by customer_state
order by customer_state
limit 10;
```

```
select customer_state ,
concat(cast(sum(oi.price) as INT), ' Brazilian_reals') as Total_amount,
concat(cast(avg(oi.price) as INT), ' Brazilian_reals') as Avg_amt
from `target.customers` c
join `target.orders` o
on c.customer_id=o.customer_id
join`target.order_items`oi
on oi.order_id = o.order_id
group by customer_state
order by customer_state
limit 10;
```

Output:

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

Row	customer_state	Total_amount	Avg_amt
1	AC	15983 Brazilian_reals	174 Brazilian_reals
2	AL	80315 Brazilian_reals	181 Brazilian_reals
3	AM	22357 Brazilian_reals	135 Brazilian_reals
4	AP	13474 Brazilian_reals	164 Brazilian_reals
5	BA	511350 Brazilian_reals	135 Brazilian_reals
6	CE	227255 Brazilian_reals	154 Brazilian_reals
7	DF	302604 Brazilian_reals	126 Brazilian_reals
8	ES	275037 Brazilian_reals	122 Brazilian_reals
9	GO	294592 Brazilian_reals	126 Brazilian_reals
10	MA	119648 Brazilian_reals	145 Brazilian_reals

Insights:

The analysis revels significant variation in total and average order prices across different states. PB has got the highest average order amount with value 191 Brazilian_reals which shows the potential purchasing power. This potential can be used to pour in new products and services for the market. PB has got the lowest average order amount of 110 Brazilian_reals which could infer the low purchasing power.

This information enables targeted marketing campaigns and provides insights into regional customer behavior and preferences.

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

Code:

select c. customer_state,

cast (sum (oi. freight_value) as INT) as Total_freight_value,

cast (avg (oi. freight_value) as INT) as Avg_freight_value

from `target.customers` c

join `target.orders` o

on c. customer_id=o. customer_id

join `target.order_items` oi

on o. order_id=oi. order_id

group by c. customer_state

order by c. customer_state

limit 10;

```
select c.customer_state,
cast(sum(oi.freight_value) as INT) as Total_freight_value,
cast(avg(oi.freight_value) as INT) as Avg_freight_value
 from `target.customers` c
 join `target.orders` o
  on c.customer_id=o.customer_id
  join `target.order_items` oi
  on o.order_id=oi.order_id
  group by c.customer_state
  order by c.customer_state;
```

Output:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUT

Row	customer_state	Total_freight_value	Avg_freight_value
1	AC	3687	40
2	AL	15915	36
3	AM	5479	33
4	AP	2789	34
5	BA	100157	26
6	CE	48352	33
7	DF	50625	21
8	ES	49765	22
9	GO	53115	23
10	MA	31524	38

Insights:

The query showcases the cost of freight values for each state. States like SP and MG have higher total freight values, indicating higher overall shipping costs incurred. On the other hand, states like RR and AP have lower total freight values.

Average freight values reflect the average shipping cost per order, with states like SP and MG having lower average freight values, States like PB and PI have higher average freight values, indicating relatively higher shipping costs per order.

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

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.

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

5.1.1. time_to_deliver = order_delivered_customer_date - order_purchase_timestamp

5.1.2. diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

Code:

```
select order_id,
datetime_diff(order_delivered_customer_date, order_purchase_timestamp,day) as time_to_deliver,
datetime_diff (order_estimated_delivery_date, order_delivered_customer_date,day) as diff_estimated_delivery
from `target.orders`
order by order_id
limit 10;
```

```
select
order_id,
datetime_diff(order_delivered_customer_date,order_purchase_timestamp,day) as time_to_deliver,
datetime_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery
from `target.orders`
order by order_id
```

Output:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTI

Row	order_id	time_to_deliver	diff_estimated_delive
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

Insights:

From Query Analysis, the "time_to_deliver" column represents the number of days taken to deliver an order to the customer from the purchase date, while the "diff_estimated_delivery" column indicates the difference between the estimated delivery date and the actual delivery date.

By analyzing this data, we can identify orders that took longer to deliver and compare each delivery time with the average delivery timeline to assess delivery efficiency. Negative values in the "diff_estimated_delivery" column indicate delayed deliveries, while positive values indicate early deliveries.

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

Code:

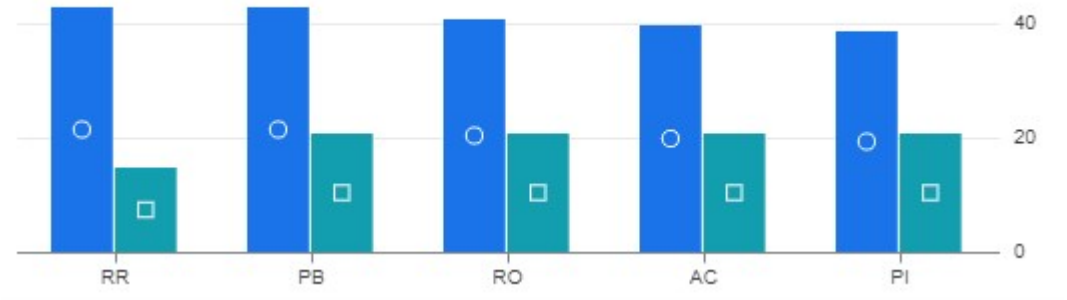
```
with r_data AS (
SELECT c. customer_state, ROUND (AVG (oi. freight_value)) AS avg_frieght_value,
row_number () over (ORDER BY AVG (oi. freight_value) desc) AS rank_high,
row_number () over (ORDER BY AVG (oi. freight_value) asc) AS rank_low
from `target.customers` c
join `target.orders` o
on c. customer_id=o. customer_id
join `target.order_items` oi
on o. order_id=oi. order_id
Group BY c. customer_state
)
select high. customer_state as high_FV_state,
high. Avg_frieght_value as high_AvgFV,
low. Customer_state as low_FV_state,
low. Avg_frieght_value as low_AvgFV
from r_data high
join r_data low
on high. rank_high=low. rank_low
where high. rank_high<=5
order by high. rank_high;
```

```sql
with r_data AS(
  SELECT c.customer_state,ROUND(AVG(oi.freight_value)) AS avg_frieght_value,
  row_number() over ( ORDER BY AVG(oi.freight_value) desc) AS rank_high,
  row_number() over ( ORDER BY AVG(oi.freight_value) asc) AS rank_low
  from `target.customers` c
  join `target.orders` o
  on c.customer_id=o.customer_id
  join `target.order_items` oi
  on o.order_id=oi.order_id
  Group BY c.customer_state
)
select high.customer_state as high_FV_state,
       high.avg_frieght_value as high_AvgFV,
       low.customer_state as low_FV_state,
       low.avg_frieght_value as low_AvgFV
from r_data high
join r_data low
on high.rank_high=low.rank_low
where high.rank_high<=5
order by high.rank_high;
```

Output:

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

Row	high_FV_state	high_AvgFV	low_FV_state	low_AvgFV
1	RR	43.0	SP	15.0
2	PB	43.0	PR	21.0
3	RO	41.0	MG	21.0
4	AC	40.0	RJ	21.0
5	PI	39.0	DF	21.0



Insights:

Our query displays the certain states that exhibit higher average freight values, indicating specific characteristics that result in increasing freight costs. Some states demonstrate lower average freight values, suggesting more favorable logistics infrastructure. With average freight costs, there is a notable difference in the average freight value among different states.

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

Code:

```
with A as(
select customer_id,
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as delivery_t
from `target.orders` ),
B AS(
select c.customer_state,round (AVG(A.delivery_t)) as Avg_deli_t,
ROW_NUMBER() OVER (ORDER BY AVG(A.delivery_t) DESC) AS Fast_deli_time,
ROW_NUMBER() OVER (ORDER BY AVG(A.delivery_t) ASC) AS Low_deli_time
from `target.customers` c
join A
on a.customer_id=c.customer_id
group by 1)
select fast.customer_state as fast_delivery_state,
fast.Avg_deli_t as fast_avg_delivery_t,
less.customer_state as least_delivery_state,
less.Avg_deli_t as least_avg_delivery_t
from B fast
join B less
on fast.Fast_deli_time=less.Low_deli_time
where fast.Fast_deli_time<=5
```

order by fast.Fast_deli_time ;

```
with A as(
  select customer_id,
date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as delivery_t
 from `target.orders`  ),
B  AS(
  select c.customer_state,
        round(AVG(A.delivery_t)) as Avg_deli_t,
        ROW_NUMBER() OVER (ORDER BY AVG(A.delivery_t) DESC) AS Fast_deli_time,
        ROW_NUMBER() OVER (ORDER BY AVG(A.delivery_t) ASC) AS Low_deli_time
  from `target.customers` c
  join A a
  on a.customer_id=c.customer_id
  group by 1
  )
select fast.customer_state as fast_delivery_state,
       fast.Avg_deli_t as fast_avg_delivery_t,
       less.customer_state  as least_delivery_state,
       less.Avg_deli_t  as least_avg_delivery_t
from B fast
join B less
on fast.Fast_deli_time=less.Low_deli_time
where fast.Fast_deli_time<=5
order by fast.Fast_deli_time ;
```

Output:

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

Row	fast_delivery_state	fast_avg_delivery_t	least_delivery_state	least_avg_delivery_t
1	RR	29.0	SP	8.0
2	AP	27.0	PR	12.0
3	AM	26.0	MG	12.0
4	AL	24.0	DF	13.0
5	PA	23.0	SC	14.0

Insights:

The states with high average delivery time are the ones which need attention with respect to resources and planning those states. If not improved can negatively impact the customer experience. The states with low average delivery time can be needed to devise new strategies to bring it down.

It is essential to identify and address the factors causing these prolonged delivery durations. Businesses should evaluate their logistics network, transportation routes, and last-mile delivery processes to optimize efficiency. Collaborating with local logistics providers or establishing strategic partnerships can also help improve delivery performance in these states. Implementing technology solutions like real-time tracking systems and efficient delivery scheduling tools can streamline the last-mile operations.

5.4. 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.

Code:

```
with A as (
select customer_id, order_id,
date_diff (order_estimated_delivery_date, order_delivered_customer_date,day) as delivery_t
from `target.orders`
where order_delivered_customer_date is not null or order_status='delivered'  )
select c.customer_state,
round(avg(a.delivery_t),1) as avg_delivery_time
from A a
join `target.customers` c
on c.customer_id=a.customer_id
group by 1
order by 2 desc
limit 5
```

```
with A as(
  select customer_id,order_id,
date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as delivery_t
 from `target.orders`
 where order_delivered_customer_date is not null or order_status='delivered'  )

 select c.customer_state,
        round(avg(a.delivery_t),1) as avg_delivery_time,
 from A a
 join `target.customers` c
 on c.customer_id=a.customer_id
 group by 1
 order by 2 desc
 limit 5
```

Output:

JOB INFORMATION | RESULTS | CHART | J

Row	customer_state	avg_delivery_time
1	AC	19.8
2	RO	19.1
3	AP	18.7
4	AM	18.6
5	RR	16.4

Insights:

The data displays the states with most efficient delivery time. This suggests efficient delivery processes, well-established transportation networks, optimized routing strategies, and proactive coordination with shipping carriers.

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

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

Code:

select extract (year from order_purchase_timestamp) as year,
extract (month from order_purchase_timestamp) as month,
p.payment_type,
count(p.order_id) as no_of_orders
from `target.orders` o
join `target.payments`p
on o.order_id=p.order_id
where payment_type is not null
group by 1,2,3
order by 1,2
limit 10;

```
select extract(year from order_purchase_timestamp) as year,
       extract(month from order_purchase_timestamp) as month,
       p.payment_type,
       count(p.order_id) as no_of_orders
from `target.orders` o
join `target.payments`p
on o.order_id=p.order_id
where payment_type is not null
group by 1,2,3
order by 1,2
```

Output:

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	year	month	payment_type	no_of_orders
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

Insights:

The query provides insights into customer payment preferences by analyzing the monthly distribution of payment types. By analyzing the monthly distribution of payment types, businesses can gain insights into customer payment preferences, identify trends in payment methods, this information can be valuable for optimizing the checkout experience, expanding payment options and improving overall customer satisfaction.

As observed, the most used methods include credit card, UPI, debit card and vouchers. Credit card emerges as the preferred payment method, highlighting its convenience, security, and widespread acceptance. Businesses should ensure seamless credit card payment processing and maintain strong partnerships with payment service providers. Analyzing payment trends helps optimize the checkout experience, expand payment options, and enhance overall customer satisfaction.

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

Code:

```
select payment_installments,

count( order_id) as No_of_Orders

from `target.payments`

where payment_installments>=1

group by 1

order by 1

limit 10
```

```
select payment_installments,
count( order_id) as No_of_Orders
from `target.payments`
where payment_installments>=1
group by 1
order by 1
limit 10
```

Output:

JOB INFORMATION | RESULTS | CH.

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

Insights:

As observed, the maximum number of orders belong to the first installment paid by the customers.

To leverage this pattern and drive sales growth, businesses should consider providing customers with flexible payment options and instalment plans. By offering the ability to divide payments into multiple instalments, businesses can accommodate customer preferences, enhance affordability, and potentially expand their customer base. Incorporating instalment plans into the payment options can be an effective strategy to boost sales and improve overall customer satisfaction.

Conclusion:

From the data analysis, target can improve its business by understanding customers and providing them with better offers, discounts, services and vouchers. This can be achieved by understanding the customer preferences, trends and then strategize to address them.