Target Case Study

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

- Data type of all columns in the customer's table.

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

Insights:

Apart from customer zip code prefix which is an Integer type all the other columns are string.

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

Code:

```
select max_timestamp, min_timestamp, date_diff(date(max_timestamp),
date(min_timestamp),day)
from(SELECT *,
max(order_purchase_timestamp) over() as max_timestamp,
min(order_purchase_timestamp) over() as min_timestamp
FROM `praxis-wall-390301.Target.orders`) a
```

Output:

Row	max_timestamp	min_timestamp	difference_in_days
1	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
2	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
3	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
4	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
5	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
6	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
7	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
8	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
9	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773
10	2018-10-17 17:30:18 UTC	2016-09-04 21:15:19 UTC	773

Insights:

The orders were placed between the year 2016 and 2018. The difference between the first order and the last order is 773 days.

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

Code:

```
select count(distinct customer_city) as Number_of_cities, count(distinct customer_state) as Number_of_states
from `Target.customers`;
```

Output:

Row	Number_of_cities	Number_of_states
1	4119	27

Insights:

The total number of cities and states of the customers who ordered during the given period are 4119 and 27 respectively.

2) In-depth Exploration

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

Code:

```
--    Is there a growing trend in the no. of orders placed over the past years?
select * from (select extract(YEAR from order_purchase_timestamp) as years, count(distinct order_id) as number_of_orders
from `Target.orders`
group by extract(YEAR from order_purchase_timestamp))a
order by years;
```

Output:

Row	years	number_of_orders
1	2016	329
2	2017	45101
3	2018	54011

Insights:

Yes, orders have increased over the past year from 2016 till 2018.

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

Code:

```
select * from (select extract(MONTH from order_purchase_timestamp) as Month, count(distinct order_id) as number_of_orders
from `Target.orders`
group by extract(MONTH from order_purchase_timestamp))a
order by Month;
```

Output:

Row	Month	number_of_orders
1	1	8069
2	2	8508
3	3	9893
4	4	9343
5	5	10573
6	6	9412
7	7	10318
8	8	10843
9	9	4305

Insights:

It looks like the maximum orders were placed in the month of August and minimum orders were placed in the month of September.

- During what time of the day, do the Brazilian customers mostly place their orders?

Code:

```
select Time_of_day , count(order_id) as number_of_orders from (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'
when extract(hour from order_purchase_timestamp) between 19 and 23 then 'Night'
else 'No_scope'
end as Time_of_day
from `Target.orders`) a
group by Time_of_Day
order by number_of_orders desc ;
```

Output:

Row	Time_of_day	number_of_orders
1	Afternoon	38135
2	Night	28331
3	Mornings	27733
4	Dawn	5242

Insights:
Brazilian customers place most of their orders in the afternoon time.
Some targeted ads can be promoted around this time in the regions where most orders are placed.

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

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

Code:

```
select
order_month,
customer_state,
count(order_id) as Number_of_orders
from
(select a.customer_state,
b.order_id,
extract(month from b.order_purchase_timestamp) as order_month
from `Target.customers` a
join `Target.orders` b
using (customer_id)) a
where customer_state = 'AC'
group by customer_state, order_month
order by  order_month
```

Output:

Row	order_month	customer_state	Number_of_orders
1	1	AC	8
2	2	AC	6
3	3	AC	4
4	4	AC	9
5	5	AC	10
6	6	AC	7
7	7	AC	9
8	8	AC	7
9	9	AC	5
10	10	AC	6

- How are the customers distributed across all the states?

Code:

```
select customer_state , count(distinct customer_id) as number_of_customers,
round((COUNT(DISTINCT customer_id) / SUM(COUNT(DISTINCT customer_id)) OVER ()) * 100,2) AS percentage_of_customers
from `Target.customers`
group by customer_state
order by number_of_customers desc;
```

Output:

Row	customer_state	number_of_customers	percentage_of_customers
1	SP	41746	41.98
2	RJ	12852	12.92
3	MG	11635	11.7
4	RS	5466	5.5
5	PR	5045	5.07
6	SC	3637	3.66
7	BA	3380	3.4
8	DF	2140	2.15
9	ES	2033	2.04
10	GO	2020	2.03

Insights:

The maximum number of customers are from SP state with 41746 and the minimum number of customers are from RR state with 46 customers.

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

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

Code:

```
with cte as(select Years, round(sum(b.payment_value),2) as cost_of_orders
from (select order_id, extract(MONTH from order_purchase_timestamp) as Months, extract(YEAR from order_purchase_timestamp) as Years
from `Target.orders`
where extract(MONTH from order_purchase_timestamp) between 1 and 8
and extract(YEAR from order_purchase_timestamp) in (2017 , 2018)) a
join
(select order_id, payment_value
from `Target.payments`) b
using(order_id)
group by Years
order by Years),

cte2 as (select Years as Year_2017,
cost_of_orders as cost_of_orders_2017,
lead(years) over(order by years) as Year_2018,
lead(cost_of_orders) over(order by years) as cost_of_orders_2018
from cte
limit 1)


select round((((cost_of_orders_2018 -  cost_of_orders_2017) / cost_of_orders_2017)* 100),2) as Percentage_increase
from cte2
```

Output:

Row	Percentage_increase
1	136.98

Insights:

The percentage increase in the cost of orders between 2017 and 2018 is 136.98%.

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

Code:

```
select a.customer_state , round(sum(c.price),2) as total_value, round(avg(c.price),2) as average_value
from `Target.customers` a
join `Target.orders` b
on a.customer_id = b.customer_id
join `Target.order_items` c
on b.order_id = c.order_id
group by a.customer_state
order by total_value desc;
```

Output:

Row	customer_state	total_value	average_value
1	SP	5202955.05	109.65
2	RJ	1824092.67	125.12
3	MG	1585308.03	120.75
4	RS	750304.02	120.34
5	PR	683083.76	119.0
6	SC	520553.34	124.65
7	BA	511349.99	134.6
8	DF	302603.94	125.77
9	GO	294591.95	126.27
10	ES	275037.31	121.91

Insights:

State SP has the total value of order price with 5202955 and RR state has the lease amount of total value with 7829.

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

Code:

```
select a.customer_state , round(sum(c.freight_value),2) as total_freight_value, round(avg(c.freight_value),2) as average_freight_value
from `Target.customers` a
join `Target.orders` b
on a.customer_id = b.customer_id
join `Target.order_items` c
on b.order_id = c.order_id
group by a.customer_state
order by total_freight_value desc;
```

Output:

Row	customer_state	total_freight_value	average_freight_value
1	SP	718723.07	15.15
2	RJ	305589.31	20.96
3	MG	270853.46	20.63
4	RS	135522.74	21.74
5	PR	117851.68	20.53
6	BA	100156.68	26.36
7	SC	89660.26	21.47
8	PE	59449.66	32.92
9	GO	53114.98	22.77
10	DF	50625.5	21.04

Insights:

SP State has the total freight value of 718723.07 and RR state has the least total freight value of 2235.19.

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

- 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.

Code:

```
select * from(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`) a
where time_to_deliver is not null
and diff_estimated_delivery is not null
order by time_to_deliver desc;
```

Output:

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

Insights:

The maximum time to deliver was 209 days and the maximum delay between the estimated delivery date and the final delivery day was 188 days.

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

Code:

```
with cte as (select *,
dense_rank() over(order by avg_freight_value asc) as avg_freight_rank_asc,
dense_rank() over(order by avg_freight_value desc) as avg_freight_rank_desc
 from (select distinct a.seller_state, round(avg(b.freight_value) over(partition by a.seller_state),2) as avg_freight_value
from `Target.sellers` a
join `Target.order_items` b
using(seller_id)) a)

select a.seller_state as bottom_five_states ,
a.avg_freight_value as bottom_5_avg_freight,
--a.avg_freight_rank_asc as bottom_5,
b.seller_state as top_five_states,
b.avg_freight_value as top_5_avg_freight,
--b.avg_freight_rank_desc
from cte a
join cte b
on a.avg_freight_rank_asc = b.avg_freight_rank_desc
where a.avg_freight_rank_asc <= 5
and b.avg_freight_rank_desc <=5
order by a.avg_freight_rank_asc
```

Output:

Row	bottom_five_states	bottom_5_avg_freight	top_five_states	top_5_avg_freight
1	SP	18.45	RO	50.91
2	PA	19.39	CE	46.38
3	RJ	19.47	PB	39.19
4	DF	20.57	PI	36.94
5	PR	22.72	AC	32.84

Insights:

Bottom 5 states with avg freight value are SP, PA, RJ, DF, FR.
Top 5 states with avg freight value are RO, CE, PB, PI, AC

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

Code:

```
with cte as(select *,
dense_rank() over(order by avg_time_to_deliver) as bottom_5,
dense_rank() over(order by avg_time_to_deliver desc) as Top_5
from (select  a.seller_state,
round(avg(DATETIME_DIFF(order_delivered_customer_date, order_purchase_timestamp, day)),2) as avg_time_to_deliver
from `Target.sellers` a
join `Target.order_items` b
on a.seller_id = b.seller_id
join `Target.orders` c
on b.order_id = c.order_id
group by a.seller_state)a
where avg_time_to_deliver is not null)

select a.seller_state as bottom_5_states, a.avg_time_to_deliver ,a.bottom_5 , b.seller_state as bottom_5_states , b.avg_time_to_deliver, b.Top_5
from cte a
join cte b
on a.bottom_5 = b.top_5
where a.bottom_5 <=5
and a.avg_time_to_deliver is not null
and b.avg_time_to_deliver is not null
order by bottom_5
```

Output:

Row	bottom_5_states	avg_time_to_deliver	bottom_5	bottom_5_states_1	avg_time_to_deliver	Top_5
1	RS	11.09	1	AM	47.33	1
2	RJ	11.55	2	CE	17.43	2
3	SP	11.81	3	MA	17.27	3
4	MS	11.9	4	RO	16.93	4
5	DF	12.09	5	MT	14.26	5

Insights:

States with the lowest avg delivery time are RS, RJ, SP, MS, DF.

States with the highest avg delivery time are AM, CE, MA, RO, MT

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

  Code:

```
select a.seller_state,
round(avg(date_diff(c.order_estimated_delivery_date ,c.order_delivered_customer_date, day)),2) as Top5_fastest_delivery_states
from `Target.sellers` a
join `Target.order_items` b
on a.seller_id = b.seller_id
join `Target.orders` c
on b.order_id = c.order_id
where c.order_delivered_customer_date is not null
group by a.seller_state
order by Top5_fastest_delivery_states desc
limit 5
```

  Output:

Row	seller_state	Top5_fastest_delivery_states
1	RO	23.5
2	PB	18.84
3	MS	16.46
4	SE	16.3
5	RS	15.37

Insights:

States RO, PB, MS, SE, RS have the fastest delivery time as compared to the estimated delivery time.

6) Analysis based on the payments:

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

Code:

```
select * from (select extract(MONTH from  a.order_purchase_timestamp) as Months, b.payment_type, count(a.order_id) as
Number_of_orders
from `Target.orders` a
join `Target.payments` b
using(order_id)
group by extract(MONTH from  a.order_purchase_timestamp), b.payment_type) a
order by Months, payment_type
```

Output:

Row	Months	payment_type	Number_of_orders
1	1	UPI	1715
2	1	credit_card	6103
3	1	debit_card	118
4	1	voucher	477
5	2	UPI	1723
6	2	credit_card	6609
7	2	debit_card	82
8	2	voucher	424
9	3	UPI	1942
10	3	credit_card	7707

Insights:

As per the output it is observer that most of the orders are made using credit across all the months and the lease number of orders are made using debit card. Maximum orders using credit were made in the month of May with 8350 total orders. As a data scientist, I would recommend promoting offers in the months like October and November where there are least purchases done to increase the total revenue.

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

Code:

```
select b.payment_installments ,count(a.order_id) as Number_of_orders
from `Target.orders` a
join `Target.payments` b
using(order_id)
WHERE b.payment_installments >= 1
and b.payment_value > 0
group by b.payment_installments
```

Output:

Row	payment_installment	Number_of_orders
1	1	52537
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:

Maximum Payments were made at installment 1 and the minimum payments were made at installment 22 and 23. Customers usually tend to play their initial installments and become defaulters later or maybe most of the customers don't opt for longer EMIs.