SCALER - FEBRUARY 2023
BUSINESS CASE - TARGET

Ashwath J

Context -
Target is one of the world's most recognized brands and one of America's leading retailers. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

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

--1.1 Data type of columns in a table

There are 8 relational tables in the database.

1.Customers: customer_id is the primary key of the table

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

2.Orders: order_id is the primary key and customer_id is the foreign key.

Field name	Type	Mode
order_id	STRING	NULLABLE
customer_id	STRING	NULLABLE
order_status	STRING	NULLABLE
order_purchase_timestamp	TIMESTAMP	NULLABLE
order_approved_at	TIMESTAMP	NULLABLE
order_delivered_carrier_date	TIMESTAMP	NULLABLE
order_delivered_customer_date	TIMESTAMP	NULLABLE
order_estimated_delivery_date	TIMESTAMP	NULLABLE

3.Payments: order_id is the foreign key and the table has no primary key as such.

Field name	Type	Mode
order_id	STRING	NULLABLE
payment_sequential	INTEGER	NULLABLE
payment_type	STRING	NULLABLE
payment_installments	INTEGER	NULLABLE
payment_value	FLOAT	NULLABLE

4.Order_items: Order_item_id is the primary key and order_id,product_id,seller_id acts like a foreign key.

Field name	Type	Mode
order_id	STRING	NULLABLE
order_item_id	INTEGER	NULLABLE
product_id	STRING	NULLABLE
seller_id	STRING	NULLABLE
shipping_limit_date	TIMESTAMP	NULLABLE
price	FLOAT	NULLABLE
freight_value	FLOAT	NULLABLE

5.Geolocation:There is no primary key as such in this table,but a combination of geolocation_lat and geolocatin_lng act as primary key.

Field name	Type	Mode
geolocation_zip_code_prefix	INTEGER	NULLABLE
geolocation_lat	FLOAT	NULLABLE
geolocation_lng	FLOAT	NULLABLE
geolocation_city	STRING	NULLABLE
geolocation_state	STRING	NULLABLE

6.Order_reviews:Review_id is the primary key and order_id is the foreign key.

Field name	Type	Mode
review_id	STRING	NULLABLE
order_id	STRING	NULLABLE
review_score	INTEGER	NULLABLE
review_comment_title	STRING	NULLABLE
review_creation_date	TIMESTAMP	NULLABLE
review_answer_timestamp	TIMESTAMP	NULLABLE

7.Products:Product_id is the primary key

Field name	Type	Mode
product_id	STRING	NULLABLE
product_category	STRING	NULLABLE
product_name_length	INTEGER	NULLABLE
product_description_length	INTEGER	NULLABLE
product_photos_qty	INTEGER	NULLABLE
product_weight_g	INTEGER	NULLABLE
product_length_cm	INTEGER	NULLABLE
product_height_cm	INTEGER	NULLABLE
product_width_cm	INTEGER	NULLABLE

8.Sellers:Seller_id is the primary key.

Field name	Type	Mode
seller_id	STRING	NULLABLE
seller_zip_code_prefix	INTEGER	NULLABLE
seller_city	STRING	NULLABLE
seller_state	STRING	NULLABLE

SCALER - FEBRUARY 2023
BUSINESS CASE - TARGET

--1.2 Time period for which the data is given

```
SELECT MIN(EXTRACT(year from order_purchase_timestamp)) as starting_year,

MAX(EXTRACT(year from order_purchase_timestamp)) as ending_year

FROM `target_data.orders` as o
```

Output->

starting_year	ending_year
2016	2018

Insights - The data set contains information of **99441** orders placed over a time period of 2016 to 2018.But the data is inconsistent in the year 2016 and again the data is again inconsistent after August 2018.

--1.3 Cities and States of customers ordered during the given period

CITY QUERY -

```
SELECT DISTINCT customer_city FROM `target_data.orders` as o

left join `target_data.customers` as c

on o.customer_id = c.customer_id
```

Output->

customer_city
rio de janeiro
sao leopoldo
general salgado
brasilia
paranavai
cuiaba
sao luis
maceio
hortolandia
varzea grande

Insights - Customers have ordered from almost 4119 cities across Brazil.We can infer our supply chain to be strong.

Recommendation - There are 8011 cities in Brazil (data from geolocations) and we cover just 4119 cities.Our target for upcoming years is to increase our distribution channel.

STATE QUERY -

```
SELECT DISTINCT customer_state FROM `target_data.orders` as o

left join `target_data.customers` as c

on o.customer_id = c.customer_id
```

Output->

SCALER - FEBRUARY 2023

BUSINESS CASE - TARGET

customer_state
RJ
RS
SP
DF
PR
MT
MA
AL
MG
PE

Insights - The customers ordered are widespread in almost 27 states in Brazil.

Q2) In-depth Exploration:

--2.1 Is there a growing trend of e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?

Growth trends in e-commerce can be analyzed using monthly sales or monthly order counts.

1st Approach (order_count) -

```
SELECT order_year,order_month,count(order_id) as total_orders

from(

SELECT *,extract(year from order_purchase_timestamp) as order_year,

extract(month from order_purchase_timestamp) as order_month

FROM `target_data.orders`

) as tab

group  by order_year,order_month
```

```
order by order_year,order_month
```

2nd Approach(Sales) -

```
SELECT order_year,order_month,count(order_id) as
total_orders,ROUND(SUM(payment_value),2) as total_sales

from(

SELECT o.order_id,p.payment_value,extract(year from order_purchase_timestamp) as
order_year,

extract(month from order_purchase_timestamp) as order_month

FROM `target_data.orders` as o

left join `target_data.payments` as p

on p.order_id = o.order_id

) as tab

group  by order_year,order_month

order by order_year,order_month
```

Output->

order_year	order_month	total_orders	total_sales
2016	9	4	252.24
2016	10	342	59090.48
2016	12	1	19.62
2017	1	850	138488.04
2017	2	1886	291908.01
2017	3	2837	449863.6
2017	4	2571	417788.03
2017	5	3944	592918.82

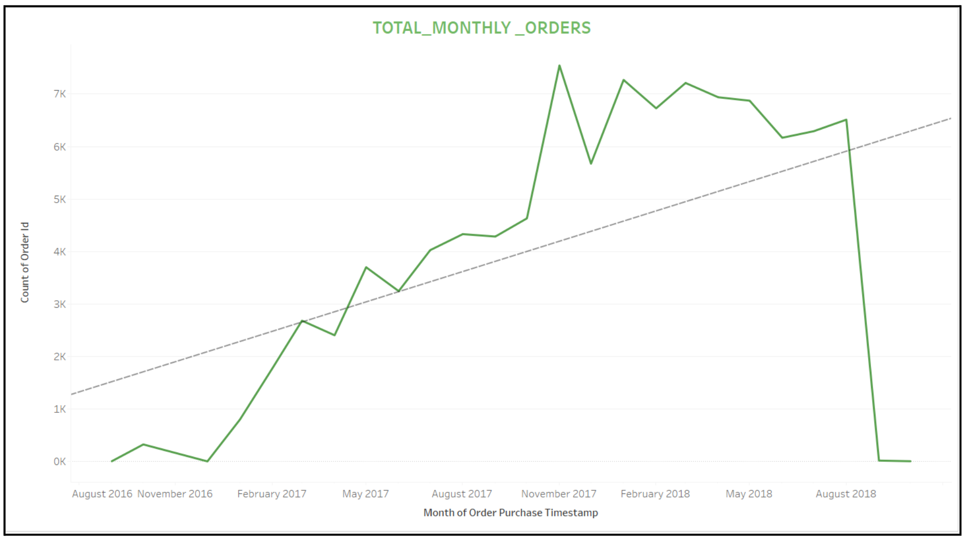
SCALER - FEBRUARY 2023

BUSINESS CASE - TARGET

Insights -

Let's not consider the data for 2016 since it's not consistent.

But in the year 2017 we can clearly see there is an increase in the total_orders exponentially..



Positive slope indicates the increase in the orders,indicating a growing trend in e-commerce in Brazil.

The growth is saturated after 2017 because in 2018 the growth is significantly less compared to 2017.

Recommendations - State wise there are some states which are not performing as well as compared to other states, So we can focus on these states to further increase sales and orders.

State AC has a 12 % decrease in the sales, so we can focus on these states.

SCALER - FEBRUARY 2023

BUSINESS CASE - TARGET

Row	customer_state	year_2017	year_2018	sum_2017	sum_2018	increase_in_sales
1	AC	2017	2018	7655.38000...	6683.10000...	-12.700610551011179
2	AL	2017	2018	29929.8399...	44537.4900...	48.80630835313525
3	SE	2017	2018	21441.9200...	34066.5299...	58.878169492284158
4	PA	2017	2018	66690.2399...	106809.299...	60.157318372223578
5	RO	2017	2018	14682.9000...	29382.4800...	100.11360153648123
6	TO	2017	2018	16088.1600...	32657.3899...	102.99021143499301
7	RJ	2017	2018	519162.490...	1074696.81...	107.0058682398241
8	AM	2017	2018	7173.98000...	14935.32	108.18736600882627
9	MT	2017	2018	44921.3599...	95208.8899...	111.94569799311509
10	RS	2017	2018	217385.980...	461414.54	112.25588697118339

Comment on Seasonality - There is no monthly or quarterly seasonality in the data.

There is some seasonality in seasons.In both the years the sale is high in summer.

Query -

```
SELECT year,seasons,sum(payment_value) as sales

from(

SELECT *

FROM(

SELECT *,extract(year from shipping_limit_date) as year,

extract(month from shipping_limit_date) as month,case when extract(month from shipping_limit_date) between 11
and 2 then 'winter'

when extract(month from shipping_limit_date) between 3 and 6 then 'summer'

when extract(month from shipping_limit_date) between 7 and 10 then 'rainy'

else 'autumn'

end as seasons

FROM `target_data.order_items` as oi

left join `target_data.products` as p

on p.product_id = oi.product_id

left join `target_data.payments` as py

on py.order_id = oi.order_id

where extract(year from shipping_limit_date) in (2017,2018) and
```

```
extract(month from shipping_limit_date) in (1,2,3,4,5,6,7,8)

) as tab

) as tab2

group by year,seasons

order by year,sales desc
```

Row	year	seasons	sales
1	2017	summer	2304415.20...
2	2017	rainy	1543822.29...
3	2017	autumn	466112.500...
4	2018	summer	5967018.32...
5	2018	rainy	2781964.92...
6	2018	autumn	2467001.81...

--2.2 What time do Brazilian customers tend to buy (Dawn, Morning, Afternoon or Night)?

```
select time_segments,COUNT(customer_id) AS total_customers

from(

select *,

CASE WHEN order_time between '6:00:00' and '11:59:00' THEN 'Morning'

     WHEN order_time between '00:00:00' and '5:59:00' THEN 'Dawn'

     WHEN order_time between '12:00:00' and '18:00:00' THEN 'Afternoon'

     ELSE 'Night'

END as time_segments

from(

select *,extract(time from order_purchase_timestamp) as order_time
```

```
from `target_data.orders`

) as tab

) as tab2

group by time_segments

order by total_customers desc
```

Output->

time_segments	total_customers
Afternoon	38365
Night	34216
Morning	22121
Dawn	4739

We could clearly see the customers tend to buy more in the Afternoon time segment between 12:00:00 and 18:00:00.

Insights - We can clearly see that people tend to shop during afternoon and night.

Recommendation - In order to increase our orders in Dawn we can plan some early morning flat sales.

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

--3.1 Get month on month orders by states

```
select *,ROUND(((tab3.total_monthly_orders / previous_monthly_orders)- 1)*100,2) as change_MOM

from(

select  *,lag(tab2.total_monthly_orders) over(partition by customer_state order by order_year, order_month) as previous_monthly_orders

from(

select customer_state,order_year,order_month,COUNT(order_id) as total_monthly_orders
```

```
from(

SELECT *,extract(year from order_purchase_timestamp) as order_year,

extract(month from order_purchase_timestamp) as order_month

FROM `target_data.orders` as o

left join `target_data.customers` as c

on o.customer_id = c.customer_id

) as tab

group by customer_state,order_year,order_month

) as tab2

order by customer_state,order_year,order_month

) as tab3
```

Output->

Month on month orders by state means the change in the orders for the current month from the previous month(in general a percentage of the change is calculated).

MOM_Change = ((current_month_value/previous_month_value) - 1) * 100 %

SCALER - FEBRUARY 2023
BUSINESS CASE - TARGET

Row	customer_state	order_year	order_month	total_monthly_orders	previous_monthly_orders	change_MOM
1	AC	2017	1	2	null	null
2	AC	2017	2	3	2	50.0
3	AC	2017	3	2	3	-33.33
4	AC	2017	4	5	2	150.0
5	AC	2017	5	8	5	60.0
6	AC	2017	6	4	8	-50.0
7	AC	2017	7	5	4	25.0
8	AC	2017	8	4	5	-20.0
9	AC	2017	9	5	4	25.0
10	AC	2017	10	6	5	20.0
11	AC	2017	11	5	6	-16.67
12	AC	2017	12	5	5	0.0
13	AC	2018	1	6	5	20.0
14	AC	2018	2	3	6	-50.0
15	AC	2018	3	2	3	-33.33
16	AC	2018	4	4	2	100.0

Interpretation of the change_MOM column - if the change_MOM is positive then there is an increase in the current total_monthly_orders by change_MOM percentage from the previous value.

Insights - We can see the orders increasing slowly from Jan to Dec 2017, but all of a sudden it fell in Feb 2018 and is gradually retracing its growth.

Recommendations - Root cause analysis can be done for states by asking their respective state managers to find the root cause.

-- 3.2 Distribution of customers across the states in Brazil

```
SELECT customer_state,COUNT(DISTINCT customer_unique_id) as total_customers

FROM `target_data.customers`

GROUP BY customer_state

order by total_customers desc
```

Output->

SCALER - FEBRUARY 2023

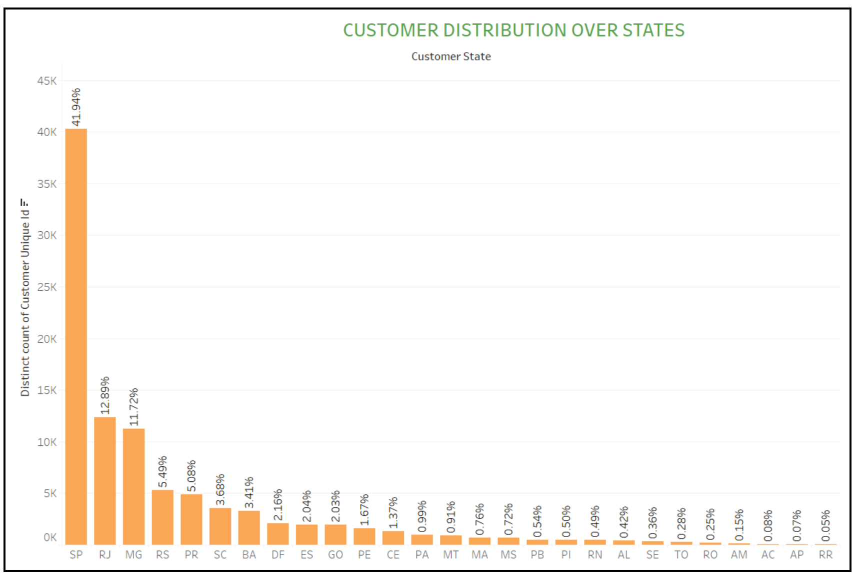
BUSINESS CASE - TARGET

Row	customer_state	total_customers
1	SP	40302
2	RJ	12384
3	MG	11259
4	RS	5277
5	PR	4882
6	SC	3534
7	BA	3277
8	DF	2075
9	ES	1964
10	GO	1952
11	PE	1609
12	CE	1313
13	PA	949
14	MT	876

Insights -

Most customers are from the state with code SP(almost 42% of overall customers).

Simple pareto holds true most of the customers 80% orders are coming from less than 20 % percent of the states.



Recommendations - Marketing strategies done in the top states can be benchmarked for the other states.

Customer experiences can be improved in less ordering states or some additional marketing like influencer marketing can be promoted.

SCALER - FEBRUARY 2023

BUSINESS CASE - TARGET

Q4)Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

--4.1 Get % increase in cost of orders from 2017 to 2018 (include months between Jan to Aug only) - You can use “payment_value” column in payments table

```
select *,ROUND(((Total_Sales - lag(Total_Sales) over(order by order_year)) / lag(Total_Sales) over(order by order_year))*100,2) as Percentage_Change

from(

select order_year,ROUND(SUM(payment_value),2) as Total_Sales

from(

SELECT *,extract(month from order_purchase_timestamp) as order_month,extract(year from order_purchase_timestamp) as order_year

FROM `target_data.payments` as p

left join `target_data.orders` as o

on p.order_id = o.order_id

) as tab

where (order_month between 1 and 8) and order_year in (2017,2018)

group by order_year

order by order_year

) as tab2

order by order_year
```

Output->

order_year	Total_Sales	Percentage_Change
2017	3669022.12	null
2018	8694733.84	136.98

SCALER - FEBRUARY 2023
BUSINESS CASE - TARGET

Insights -

The Total_Sales(order value) of 2018 is 136.98 % greater than Total_Sales 2017.

(months are limited between January and August).

Recommendations - Overall there is very good increase in the revenue,but metric like this would seem good on the highest level.

Further drill down of the metric into states or product categories would make us distinguish the best performers and worst performers.

--4.2 Mean & Sum of price and freight value by customer state

```
select customer_state,ROUND(sum(price),2) as SUM_Price,ROUND(sum(freight_value),2) as SUM_Freight,

ROUND(avg(price),2) as AVG_Price, ROUND(avg(freight_value),2) as AVG_Freight

from(

SELECT *,extract(month from order_purchase_timestamp) as m , extract(year from order_purchase_timestamp) as y

FROM `target_data.order_items` as oi

left join `target_data.orders` as o

on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab

group by customer_state

ORDER BY customer_state
```

Output->

Row	customer_state	SUM_Price	SUM_Freight	AVG_Price	AVG_Freight
1	AC	15982.95	3686.75	173.73	40.07
2	AL	80314.81	15914.59	180.89	35.84
3	AM	22356.84	5478.89	135.5	33.21
4	AP	13474.3	2788.5	164.32	34.01
5	BA	511349.99	100156.68	134.6	26.36
6	CE	227254.71	48351.59	153.76	32.71
7	DF	302603.94	50625.5	125.77	21.04
8	ES	275037.31	49764.6	121.91	22.06
9	GO	294591.95	53114.98	126.27	22.77
10	MA	119648.22	31523.77	145.2	38.26

Insights -

Row	customer_state	SUM_Price	SUM_Freight	AVG_Price	AVG_Freight
1	RR	7829.43	2235.19	150.57	42.98

The state RR has the lowest sales compared to other states, this is due to the exceptionally high **Freight cost**, it's almost 30% of the average order value.

Recommendations - Logistics partners can be changed for this state or some other efficient ways to improve supply chain can be introduced to reduce the freight cost.

Q5) Analysis on sales, freight and delivery time

--5.1 Calculate days between purchasing, delivering and estimated delivery

```
SELECT *,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
delivery_days,

date_diff(order_estimated_delivery_date,order_purchase_timestamp,day) as
est_delivery_days

FROM `target_data.orders`
```

Days between purchasing and delivery are calculated and named as delivery_days.

Days between purchasing and estimated delivery date is calculated and named as est_delivery_days.

Output->

order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	delivery_days	est_delivery_day
delivered	2017-03-17 15:56:47 UTC	2017-03-17 15:56:47 UTC	2017-04-04 10:53:37 UTC	2017-04-07 13:14:56 UTC	2017-05-18 00:00:00 UTC	20	61
delivered	2017-03-20 11:01:17 UTC	2017-03-20 11:01:17 UTC	2017-03-22 08:50:27 UTC	2017-03-30 14:04:04 UTC	2017-05-18 00:00:00 UTC	10	58
delivered	2017-03-21 13:38:25 UTC	2017-03-21 13:38:25 UTC	2017-04-04 16:30:16 UTC	2017-04-18 13:52:43 UTC	2017-05-18 00:00:00 UTC	28	57
delivered	2018-08-20 15:56:23 UTC	2018-08-22 03:55:07 UTC	2018-08-22 14:28:00 UTC	2018-08-29 22:52:40 UTC	2018-10-04 00:00:00 UTC	9	44
delivered	2018-08-12 18:14:29 UTC	2018-08-12 18:25:16 UTC	2018-08-14 13:41:00 UTC	2018-08-23 02:08:44 UTC	2018-10-04 00:00:00 UTC	10	52
delivered	2018-08-16 07:55:32 UTC	2018-08-16 08:15:14 UTC	2018-08-17 10:40:00 UTC	2018-08-23 00:09:45 UTC	2018-10-04 00:00:00 UTC	6	48
delivered	2018-08-22 22:39:54 UTC	2018-08-23 02:45:13 UTC	2018-08-23 12:28:00 UTC	2018-08-29 19:11:48 UTC	2018-10-04 00:00:00 UTC	6	42
delivered	2018-08-20 17:04:34 UTC	2018-08-20 17:29:29 UTC	2018-08-21 15:33:00 UTC	2018-08-29 16:41:59 UTC	2018-10-04 00:00:00 UTC	8	44
delivered	2018-08-09 19:17:50 UTC	2018-08-09 19:30:11 UTC	2018-08-10 12:34:00 UTC	2018-08-22 18:04:27 UTC	2018-10-04 00:00:00 UTC	12	55
delivered	2018-08-13 12:12:46 UTC	2018-08-13 12:25:21 UTC	2018-08-17 13:29:00 UTC	2018-08-29 20:58:39 UTC	2018-10-04 00:00:00 UTC	16	51
delivered	2018-08-20 17:09:20 UTC	2018-08-20 17:29:41 UTC	2018-08-22 11:29:00 UTC	2018-08-30 19:03:50 UTC	2018-10-04 00:00:00 UTC	10	44
delivered	2018-08-21 14:33:37 UTC	2018-08-21 14:50:15 UTC	2018-08-21 15:12:00 UTC	2018-08-27 21:18:38 UTC	2018-10-04 00:00:00 UTC	6	43
delivered	2018-08-23 20:56:41 UTC	2018-08-25 02:35:16 UTC	2018-08-27 14:21:00 UTC	2018-08-30 20:38:29 UTC	2018-10-04 00:00:00 UTC	6	41

Insights - As far as customer experience metric like NPS/CSAT is concerned delivery_days <= est_delivery_days, because no customers would like late delivery.Experience of late deliveries also churn customers from platform.

Out of 96478 delivered orders 7308 orders were delivered late (or after est_delivery_date). Approximately 7.57 %.

Recommendations - An apology message can be forwarded to the customers.

Customers who are D60 churned after their bad experience can be given some offer coupons.

--5.2 Find time_to_delivery & diff_estimated_delivery. Formula for the same given below:

- time_to_delivery = order_purchase_timestamp-order_delivered_customer_date
- diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date

```
SELECT *,

date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,
```

```
date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery

FROM `target_data.orders`
```

Output ->

Row	order_status	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	time_to_delivery	diff_estimated_
1	canceled	2018-02-19 19:48:52 UTC	2018-02-19 20:56:05 UTC	2018-02-20 19:57:13 UTC	2018-03-21 22:03:51 UTC	2018-03-09 00:00:00 UTC	30	-12
2	canceled	2016-10-09 15:39:56 UTC	2016-10-10 10:40:49 UTC	2016-10-14 10:40:50 UTC	2016-11-09 14:53:50 UTC	2016-12-08 00:00:00 UTC	30	28
3	canceled	2016-10-03 21:01:41 UTC	2016-10-04 10:18:57 UTC	2016-10-25 12:14:28 UTC	2016-11-08 10:58:34 UTC	2016-11-25 00:00:00 UTC	35	16
4	delivered	2017-04-15 15:37:38 UTC	2017-04-15 15:45:14 UTC	2017-04-27 16:06:59 UTC	2017-05-16 14:49:55 UTC	2017-05-18 00:00:00 UTC	30	1
5	delivered	2017-04-14 22:21:54 UTC	2017-04-15 22:30:19 UTC	2017-04-17 09:08:52 UTC	2017-05-17 10:52:15 UTC	2017-05-18 00:00:00 UTC	32	0
6	delivered	2017-04-16 14:56:13 UTC	2017-04-16 15:05:14 UTC	2017-04-17 10:44:19 UTC	2017-05-16 09:07:47 UTC	2017-05-18 00:00:00 UTC	29	1
7	delivered	2017-04-08 21:20:24 UTC	2017-04-08 21:30:16 UTC	2017-04-25 10:53:00 UTC	2017-05-22 14:11:31 UTC	2017-05-18 00:00:00 UTC	43	-4
8	delivered	2017-04-11 19:49:45 UTC	2017-04-11 20:02:27 UTC	2017-04-12 14:47:39 UTC	2017-05-22 16:18:42 UTC	2017-05-18 00:00:00 UTC	40	-4
9	delivered	2017-04-12 12:17:08 UTC	2017-04-13 12:22:08 UTC	2017-04-19 14:19:04 UTC	2017-05-19 13:44:52 UTC	2017-05-18 00:00:00 UTC	37	-1
10	delivered	2017-04-19 22:52:59 UTC	2017-04-19 23:05:12 UTC	2017-04-26 09:43:45 UTC	2017-05-23 14:19:48 UTC	2017-05-18 00:00:00 UTC	33	-5

Insights -

Row	customer_state	avg_diff_est_delivery
1	AC	19.762500000000006
2	RO	19.13168724279836
3	AP	18.731343283582088
4	AM	18.60689655172413
5	RR	16.414634146341463
6	MT	13.431151241534996
7	PA	13.190274841437633

The greater the avg_diff_est_delivery the better is the state performing in delivering orders.We can clearly see states AC and RO are fast in delivering orders compared to other states.

If diff_estimated_delivery is negative/low then the delivery has happened after the order_estimated_delivery_date, which impacts our customer experience metric.

Recommendation - Incentives can be given to workers of best performing states, which motivates other states to strive hard.

--5.3 Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery.

```
SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time

FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o

left join `target_data.order_items` as oi

on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab group by customer_state

Output->
```

Row	customer_state	mean_freight_value	mean_delivery_time	mean_diff_time
1	RJ	20.96	14.69	11.14
2	RS	21.74	14.71	13.2
3	SP	15.15	8.26	10.27
4	DF	21.04	12.5	11.27
5	PR	20.53	11.48	12.53
6	MT	28.17	17.51	13.64
7	MA	38.26	21.2	9.11
8	AL	35.84	23.99	7.98
9	MG	20.63	11.52	12.4
10	PE	32.92	17.79	12.55
11	SE	36.65	20.98	9.17
12	PA	35.83	23.3	13.37

--5.4 Sort the data to get the following:

```
SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time

FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o

left join `target_data.order_items` as oi

on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab group by customer_state

order by customer_state,mean_freight_value,mean_delivery_time,mean_diff_time
```

Output ->

Row	customer_state	mean_freight_va	mean_delivery_ti	mean_diff_time
1	AC	40.07	20.33	20.01
2	AL	35.84	23.99	7.98
3	AM	33.21	25.96	18.98
4	AP	34.01	27.75	17.44
5	BA	26.36	18.77	10.12
6	CE	32.71	20.54	10.26
7	DF	21.04	12.5	11.27
8	ES	22.06	15.19	9.77
9	GO	22.77	14.95	11.37
10	MA	38.26	21.2	9.11

As per the question given the result set is sorted on each column.(ascending)

--5.5 Top 5 states with highest/lowest average freight value - sort in desc/asc limit 5

TOP 5 WITH HIGHEST FREIGHT VALUE

```
SELECT customer_state,mean_freight_value

FROM(

SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time

FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o
```

```
left join `target_data.order_items` as oi

on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab

group by customer_state

) as state_wise

ORDER BY mean_freight_value DESC

LIMIT 5
```

Output->

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

Insights - These are the top 5 states with highest freight values, freight value on an item is a non value price for customers(therefore higher the price lower is the sales).

Recommendations - Some orders could be joined on pincode basis to avoid extra freight charges.

Ex. Let's say OD1 and OD2 are ordered from the same pincode in that case the both orders can be packed together to reduce the freight charges.

Looking for better prices in the market and making them as there logistics partner for these states.

TOP 5 WITH LOWEST FREIGHT VALUE

```
SELECT customer_state,mean_freight_value

FROM(

SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time

FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o

left join `target_data.order_items` as oi

on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab

group by customer_state

) as state_wise

ORDER BY mean_freight_value

LIMIT 5

Output->
```

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

Insights - These 5 states have minimum freight charges which is good in terms of customers perspective.

Freight value is a non value adding price to the customers purchasing from Target,

Minimizing this cost will increase in our sales metric.

--5.6 Top 5 states with highest/lowest average time to delivery

TOP 5 STATES WITH HIGHEST AVERAGE TIME TO DELIVERY

```
SELECT customer_state,mean_delivery_time

FROM(

SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time

FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o

left join `target_data.order_items` as oi
```

```
on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab

group by customer_state

) as state_wise

ORDER BY mean_delivery_time desc

LIMIT 5
```

Output->

Row	customer_state	mean_delivery_time
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

* time is in days

Insights - These are the top 5 states with highest mean_delivery_time, we cannot directly infer the speed or performance from this time. Since distance from the warehouse plays an important role in mean_delivery time.

BOTTOM 5 STATES WITH LOWEST AVERAGE DELIVERY TIME

```
SELECT customer_state,mean_delivery_time

FROM(

SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time
```

```
FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o

left join `target_data.order_items` as oi

on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab

group by customer_state

) as state_wise

ORDER BY mean_delivery_time

LIMIT 5
```

Output->

Row	customer_state	mean_delivery_t
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

* time is in days

SCALER - FEBRUARY 2023

BUSINESS CASE - TARGET

--5.7 Top 5 states where delivery is really fast/ not so fast compared to estimated date

TOP 5 STATES WHERE DELIVERY IS FAST

```
SELECT customer_state,mean_diff_time

FROM(

SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time

FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o

left join `target_data.order_items` as oi

on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab

group by customer_state

) as diff_time

ORDER BY mean_diff_time desc

LIMIT 5

Output->
```

SCALER - FEBRUARY 2023

BUSINESS CASE - TARGET

Row	customer_state	mean_diff_time
1	AC	20.01
2	RO	19.08
3	AM	18.98
4	AP	17.44
5	RR	17.43

* time is in days

Insights - Delivery is really fast when the diff_estimated_delivery is large.(estimated days - delivered days)

Delivery speed cannot be benchmarked with the days of delivery since logistics plays a vital role so speed has to be benchmarked with the diff_estimated_delivery, larger the difference faster is the delivery smaller the difference slower is the delivery

TOP 5 STATES WHERE DELIVERY IS SLOW

```
SELECT customer_state,mean_diff_time

FROM(

SELECT customer_state,ROUND(AVG(freight_value),2) as
mean_freight_value,ROUND(AVG(time_to_delivery),2) as mean_delivery_time,

ROUND(AVG(diff_estimated_delivery),2) as mean_diff_time

FROM(

SELECT o.*,date_diff(order_delivered_customer_date,order_purchase_timestamp,day) as
time_to_delivery,

date_diff(order_estimated_delivery_date,order_delivered_customer_date,day) as
diff_estimated_delivery,

oi.freight_value,c.customer_state

FROM `target_data.orders` as o

left join `target_data.order_items` as oi
```

```
on oi.order_id = o.order_id

left join `target_data.customers` as c

on c.customer_id = o.customer_id

) as tab

group by customer_state

) as diff_time

ORDER BY mean_diff_time

LIMIT 5
```

Output->

Row	customer_state	mean_diff_time
1	AL	7.98
2	MA	9.11
3	SE	9.17
4	ES	9.77
5	BA	10.12

* time is in days

Q6) Payment type analysis:

--6.1 Month over Month count of orders for different payment types

```
SELECT *,ROUND((monthly_orders / previous_month_orders - 1) * 100,2) as MOM_Orders

FROM(

SELECT payment_type,order_year,order_month,COUNT(order_id) as
monthly_orders,lag(COUNT(order_id)) over(partition by payment_type,order_year order by
order_month) as previous_month_orders

FROM(
```

SCALER - FEBRUARY 2023

BUSINESS CASE - TARGET

```
SELECT p.*,o.order_purchase_timestamp,extract(year from o.order_purchase_timestamp) as
order_year,

extract(month from o.order_purchase_timestamp) as order_month

FROM `target_data.payments` as p

LEFT JOIN `target_data.orders` as o

on p.order_id = o.order_id

) as tab

GROUP BY payment_type,order_year,order_month

order by payment_type,order_year,order_month

) as tab2
```

Output->

Row	payment_type	order_year	order_month	monthly_orders	previous_month	MOM_Orders
1	UPI	2016	10	63	*null*	*null*
2	UPI	2017	1	197	*null*	*null*
3	UPI	2017	2	398	197	102.03
4	UPI	2017	3	590	398	48.24
5	UPI	2017	4	496	590	-15.93
6	UPI	2017	5	772	496	55.65
7	UPI	2017	6	707	772	-8.42
8	UPI	2017	7	845	707	19.52
9	UPI	2017	8	938	845	11.01
10	UPI	2017	9	903	938	-3.73

MOM_Orders is a percentage of difference of current month and previous month orders for a particular payment_type.

Insights - We can clearly observe people in Brazil use credit cards while shopping.

payment_type	total_orders
credit_card	76795
UPI	19784
voucher	5775
debit_card	1529
not_defined	3

Recommendations - We can target the credit_card users by bringing attractive offers for them.

--6.2 Count of orders based on the no. of payment installment

```
SELECT payment_installments,COUNT(order_id) as total_orders

FROM `target_data.payments`

GROUP BY payment_installments

ORDER BY total_orders desc
```

Output->

Row	payment_installments	total_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
12	15	74
13	18	27

Most of the customers prefer to pay in a single installment.

Percentage of orders:

```
SELECT *,ROUND((total_orders / SUM(total_orders) OVER())*100.0,2) as overall_total
```

```
FROM(

SELECT payment_installments,COUNT(order_id) as total_orders

FROM `target_data.payments`

GROUP BY payment_installments

) as tab

order by 2 desc
```

Output->

Row	payment_install	total_orders	order_percent
1	1	52546	50.58
2	2	12413	11.95
3	3	10461	10.07
4	4	7098	6.83
5	10	5328	5.13
6	5	5239	5.04
7	8	4268	4.11
8	6	3920	3.77
9	7	1626	1.57

More than 50% of the orders are paid in a single installment.

Insights - Most of the orders are paid on a single installments and we can see there is a steady decrease in the total_orders with increase in payment_installments.

Actionables - Some more additional emi options or no cost emi options can be introduced to facilitate some other customers to order more.

product_category	price	total_ordes
PCs	1098.34	203
HOUSE PASTALS OVEN AND C...	624.29	76
ELECTRICES 2	476.12	238
Agro Industria e Comercio	342.12	212

SCALER - FEBRUARY 2023
BUSINESS CASE - TARGET

We can clearly see that the costlier items are bought comparatively very less,the number of orders of such items can be improved by introducing no cost emi options or some discounted emi options.

THANK YOU

Done by,

Ashwath J