Submission by Kirty Chandra
Batch : MWF _Beginner_9pm

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 TargetSQL.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

Insight : Information Schema gives the metadata of the tables.

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

Query:

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

Output:

JOB INFORMATION | RESULTS | CHART | JSON

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

Insight :

Start Date of order : 2016-09-04 21:15:19 UTC
End Date of Order : 2018-10-17 17:30:18 UTC

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

Query :

```
select count(geolocation_city) as citycount, count(geolocation_state) as statecount from TargetSQL.geolocation
g inner join TargetSQL.customers c
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix;
```

Output :

JOB INFORMATION | RESULTS | CHART

Row	citycount	statecount
1	15083455	15083455

```
Insight: Count of cities : 15083455
         Count of states : 15083455
```

2. In-depth Exploration:

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

```
Query:
select extract(year from order_purchase_timestamp) as year,
count(order_id) as ordercount
from TargetSQL.orders
group by year
order by year;
```

Output :

JOB INFORMATION | RESULTS | CHART

Row	year	ordercount
1	2016	329
2	2017	45101
3	2018	54011

Insight: There is increase in sales each year which shows as growing trend.

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

Query:

```
select extract(year from order_purchase_timestamp) as year,
extract(month from order_purchase_timestamp) as month,count(order_id) as ordercount
from TargetSQL.orders
group by month,year
order by year,month;
```

Output:

JOB INFORMATION | RESULTS | CHART | JSON | EXECU

Row	year	month	ordercount
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 is a growing trend in orders but peaks seen during the month of Oct, Nov in each year

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

CASE
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
WHEN EXTRACT(HOUR FROM o.order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
END AS hour,
COUNT(o.order_id) AS order_count
FROM
TargetSQL.orders o
JOIN
```

```
TargetSQL.customers c
ON o.customer_id = c.customer_id
GROUP BY
hour
ORDER BY
order_count DESC;
```

Output:

Row	hour	order_count
1	Afternoon	38135
2	Night	28331
3	Morning	27733
4	Dawn	5242

Insight: Afternoon has maximum orders

3. **Evolution of E-commerce orders in the Brazil region:**
    a. Get the month on month no. of orders placed in each state.

Query :

```
select customer_state,extract(month from order_purchase_timestamp) as month,count(order_id) as ordercount
from TargetSQL.customers c
inner join TargetSQL.orders o
ON c.customer_id=o.customer_id
group by month,customer_state
order by month,customer_state;
```

Output:

Row	customer_state	month	ordercount
1	AC	1	8
2	AL	1	39
3	AM	1	12
4	AP	1	11
5	BA	1	264
6	CE	1	99
7	DF	1	151
8	ES	1	159
9	GO	1	164
10	MA	1	66

Insight : All states shows rise in orders in peak months of Oct, Nov

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

Query:

```
select g.geolocation_state,count(customer_id) as cust_count
from TargetSQL.customers c inner join TargetSQL.geolocation g
on c.customer_zip_code_prefix=g.geolocation_zip_code_prefix
group by g.geolocation_state
order by g.geolocation_state;
```

Output:

Row	geolocation_state	cust_count
1	AC	7688
2	AL	34861
3	AM	5587
4	AP	4912
5	BA	365875
6	CE	63507
7	DF	93309
8	ES	316654
9	GO	133146
10	MA	53383

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:

```
SELECT
  EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
  (
  (
  SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 AND
  EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
  p.payment_value END)
  -
  SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND
  EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
  p.payment_value END)
  )
  /
  SUM(CASE WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND
  EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN
  p.payment_value END)
  )*100 AS percent_increase
FROM
  TargetSQL.orders o
JOIN
  TargetSQL.payments p ON o.order_id = p.order_id
WHERE
  EXTRACT(YEAR FROM o.order_purchase_timestamp) IN (2017, 2018) AND
  EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8
GROUP BY 1
ORDER BY 1;
```

Output:

Row	month	percent_increase
1	1	705.1266954171...
2	2	239.9918145445...
3	3	157.7786066709...
4	4	177.8407701149...
5	5	94.62734375677...
6	6	100.2596912456...
7	7	80.04245463390...
8	8	51.60600520477...

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

Query:

```
select c.customer_state as State,round(sum(i.price),2) as Total,round(avg(i.price),2) as Average
from TargetSQL.order_items i
left join TargetSQL.orders o
on i.order_id=o.order_id
left join TargetSQL.customers c
on c.customer_id=o.customer_id
group by c.customer_state
order by c.customer_state;
```

Output:

Row	State	Total	Average
1	AC	15982.95	173.73
2	AL	80314.81	180.89
3	AM	22356.84	135.5
4	AP	13474.3	164.32
5	BA	511349.99	134.6
6	CE	227254.71	153.76
7	DF	302603.94	125.77
8	ES	275037.31	121.91
9	GO	294591.95	126.27
10	MA	119648.22	145.2

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

Query:

```
select c.customer_state as State,round(sum(i.freight_value),2) as Total_freight,round(avg(i.freight_value),2) as
Average_freight
from TargetSQL.order_items i
left join TargetSQL.orders o
on i.order_id=o.order_id
left join TargetSQL.customers c
on c.customer_id=o.customer_id
group by c.customer_state
order by c.customer_state;
```

Output:

Row	State	Total_freight	Average_freight
1	AC	3686.75	40.07
2	AL	15914.59	35.84
3	AM	5478.89	33.21
4	AP	2788.5	34.01
5	BA	100156.68	26.36
6	CE	48351.59	32.71
7	DF	50625.5	21.04
8	ES	49764.6	22.06
9	GO	53114.98	22.77
10	MA	31523.77	38.26

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 o.order_id,
date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp,day) as delivery_time,
date_diff(o.order_delivered_customer_date,order_estimated_delivery_date,day) as delivery_time_gap
from TargetSQL.orders o;
```

Output:

Row	order_id	delivery_time	delivery_time_gap
1	1950d777989f6a877539f5379...	30	12
2	2c45c33d2f9cb8ff8b1c86cc28...	30	-28
3	65d1e226dfaeb8cdc42f66542...	35	-16
4	635c894d068ac37e6e03dc54e...	30	-1
5	3b97562c3aee8bdedcb5c2e45...	32	0
6	68f47f50f04c4cb6774570cfde...	29	-1
7	276e9ec344d3bf029ff83a161c...	43	4
8	54e1a3c2b97fb0809da548a59...	40	4
9	fd04fa4105ee8045f6a0139ca5...	37	1
10	302bb8109d097a9fc6e9cefc5...	33	5

Insight : Many orders are delivered before estimated delivery time as well.

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

Query:

```
(select c.customer_state as State,round(avg(i.freight_value),2) as Top5_highest_lowest_Average_freight
from TargetSQL.order_items i
left join TargetSQL.orders o
on i.order_id=o.order_id
left join TargetSQL.customers c
on c.customer_id=o.customer_id
group by c.customer_state
order by Top5_highest_lowest_Average_freight desc
limit 5)
UNION ALL
(select c.customer_state as State,round(avg(i.freight_value),2) as Top5_highest_lowest_Average_freight
from TargetSQL.order_items i
left join TargetSQL.orders o
on i.order_id=o.order_id
left join TargetSQL.customers c
on c.customer_id=o.customer_id
group by c.customer_state
order by Top5_highest_lowest_Average_freight asc
limit 5)
order by Top5_highest_lowest_Average_freight;
```

Output:

Row	State	Top5_highest_lowest
1	SP	15.15
2	PR	20.53
3	MG	20.63
4	RJ	20.96
5	DF	21.04
6	PI	39.15
7	AC	40.07
8	RO	41.07
9	PB	42.72
10	RR	42.98

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

Query:

```
(select distinct c.customer_state as State,
round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)) over(partition by
c.customer_state),2) as delivery_time
from TargetSQL.orders o
inner join
TargetSQL.customers c
on c.customer_id=o.customer_id
order by delivery_time asc limit 5)
UNION ALL
(select distinct c.customer_state as State,
round(avg(date_diff(o.order_delivered_customer_date,o.order_purchase_timestamp,day)) over(partition by
c.customer_state),2) as delivery_time
from TargetSQL.orders o
inner join
TargetSQL.customers c
on c.customer_id=o.customer_id
order by delivery_time DESC limit 5)
order by delivery_time;
```

Output:

Row	State	delivery_time
1	SP	8.3
2	PR	11.53
3	MG	11.54
4	DF	12.51
5	SC	14.48
6	PA	23.32
7	AL	24.04
8	AM	25.99
9	AP	26.73
10	RR	28.98

Insight : State SP has lowest average delivery time and RR has highest average delivery time

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:

```
select distinct c.customer_state as State,
round(avg(date_diff(order_estimated_delivery_date,o.order_delivered_customer_date,day)) over(partition by
c.customer_state),2) as fastest_delivery
from TargetSQL.orders o
inner join
TargetSQL.customers c
on c.customer_id=o.customer_id
order by fastest_delivery ASC
limit 5;
```

Output:

Row	State	fastest_delivery
1	AL	7.95
2	MA	8.77
3	SE	9.17
4	ES	9.62
5	BA	9.93

Insight : State AL has fastest delivery time

6.Analysis based on the payments:

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

Query :

```
select extract(month from o.order_purchase_timestamp) as month,count(o.order_id)as order_count,
p.payment_type from TargetSQL.orders o
inner join TargetSQL.payments p
on o.order_id=p.order_id
group by month,p.payment_type
order by month,p.payment_type;
```

Output:

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

Insight : Across all months credit card has maximum number of order.

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

Query :

```
select p.payment_installments, count(o.order_id)as order_count,
from TargetSQL.orders o
inner join TargetSQL.payments p
on o.order_id=p.order_id
where p.payment_installments>=1
group by p.payment_installments
order by p.payment_installments;
```

Output:

Row	payment_installment	order_count
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

Insight : With installment value as 1 has the highest number of order count.