PROJECT: RETAIL BUSINESS CASE STUDY

Q1. **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.

Filter Enter property name or value

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

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

```
1 SELECT
2     MIN(order_purchase_timestamp) AS First,
3     MAX(order_purchase_timestamp) AS Last,
4     date_diff(MAX(order_purchase_timestamp), MIN(order_purchase_timestamp), day) as Time_range
5 FROM `Target.orders`
```

Query results

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

Row	First	Last	Time_range
1	2016-09-04 21:15:19 UTC	2018-10-17 17:30:18 UTC	772

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

```
SELECT COUNT(DISTINCT geolocation_city) as city_count,
       COUNT(DISTINCT geolocation_state) as state_count
FROM `Target.geolocation`
WHERE geolocation_zip_code_prefix IN (SELECT DISTINCT customer_zip_code_prefix
                                      FROM `Target.customers`
                                      WHERE customer_id IN (SELECT DISTINCT customer_id
                                                            FROM `Target.orders`))
```

Query results

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

Row	city_count	state_count
1	5812	27

Q2. **In-depth Exploration:**

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

```
SELECT EXTRACT(YEAR from(order_purchase_timestamp)) as year,
       COUNT(order_id) order_count
FROM `Target.orders`
GROUP BY year
ORDER BY year
```

Query results

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

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

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

   The query below is showing the number of orders for each month of each year:

```
SELECT EXTRACT(YEAR from(order_purchase_timestamp)) as year,
       EXTRACT(MONTH from(order_purchase_timestamp)) as month,
       COUNT(order_id) order_count
FROM `Target.orders`
GROUP BY year,month
ORDER BY year,month
```

Query results

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

Row	year	month	order_count
1	2016	9	4
2	2016	10	324
3	2016	12	1
4	2017	1	800
5	2017	2	1780
6	2017	3	2682
7	2017	4	2404
8	2017	5	3700
9	2017	6	3245
10	2017	7	4026

Below is a visualization of the above data:

We can also run this query based on orders grouped by month only. That is, orders for each month over the years:

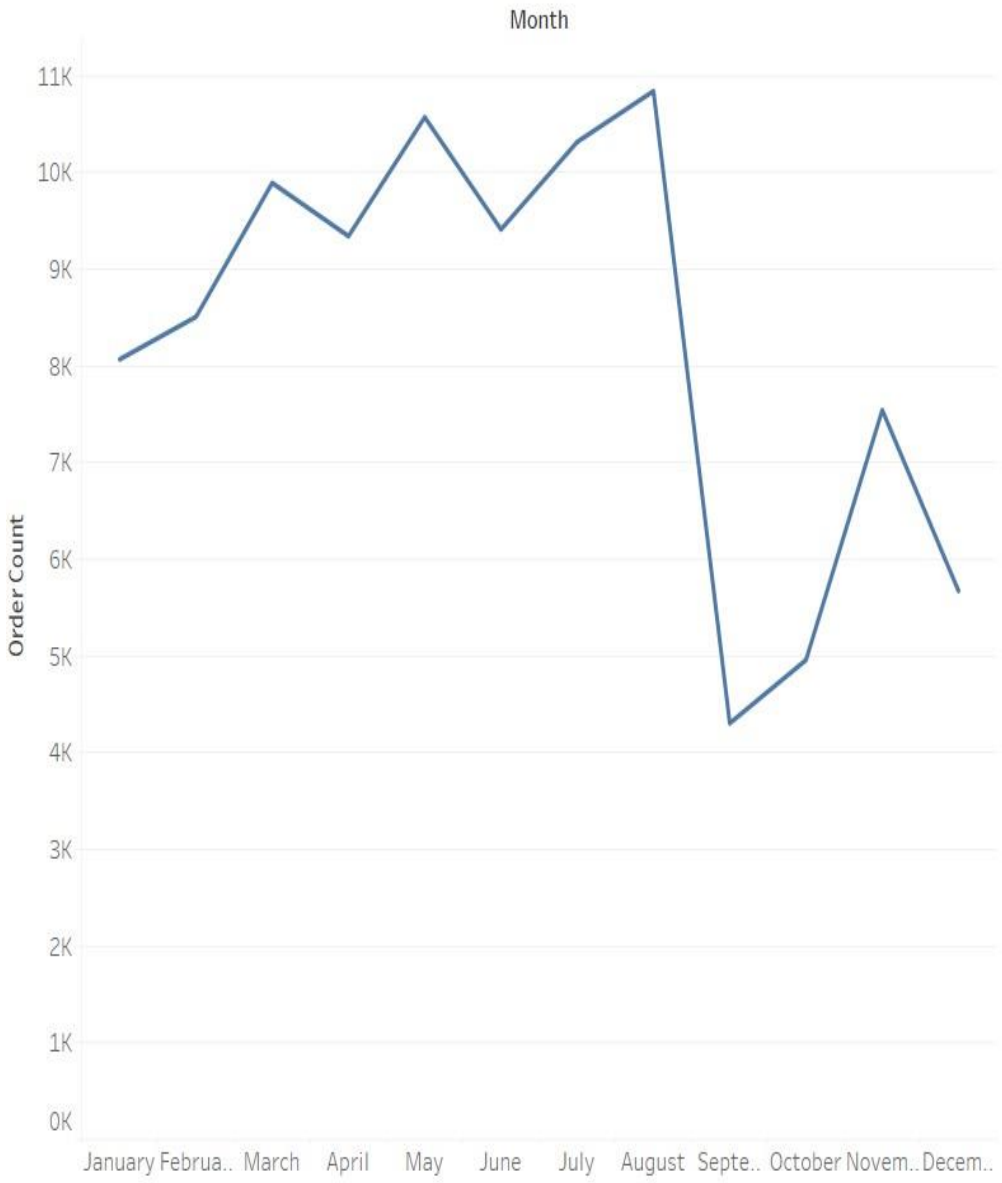
```
SELECT
        EXTRACT(MONTH from(order_purchase_timestamp)) as month,
        COUNT(order_id) order_count
FROM `Target.orders`
GROUP BY month
ORDER BY month
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXEC

Row	month	order_count
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
10	10	4959

Below is a visualization of the above data:



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

```sql
SELECT
    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"
    ELSE "Night"
    END as time_period,
    ROUND((COUNT(order_id)/(SELECT COUNT(order_id) from `Target.orders`))*100,2) as order_count_percentage
FROM `Target.orders`
GROUP BY time_period
```

Query results

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

Row	time_period	order_count_percent
1	Mornings	27.89
2	Dawn	5.27
3	Afternoon	38.35
4	Night	28.49

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

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

```
SELECT
      g.geolocation_state,
      EXTRACT(YEAR FROM o.order_purchase_timestamp) AS year,
      EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
      COUNT(o.order_id) AS Order_count
FROM `Target.orders` o JOIN `Target.customers` c ON o.customer_id=c.customer_id JOIN `Target.geolocation` g ON g.geolocation_zip_code_prefix=c.customer_zip_code_prefix
GROUP BY g.geolocation_state, year,month
ORDER BY g.geolocation_state, year,month
```

Press Alt+F1 for Acces

Query results

SAVE RESULTS ▾ EXPLORE DATA

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

Row	geolocation_state	year	month	Order_count
1	AC	2017	1	45
2	AC	2017	2	179
3	AC	2017	3	329
4	AC	2017	4	362
5	AC	2017	5	886
6	AC	2017	6	432
7	AC	2017	7	605
8	AC	2017	8	657
9	AC	2017	9	161

Since the question asks for monthly analysis with respect to states, we can also analyze like this:

```
SELECT
        g.geolocation_state,
        EXTRACT(MONTH FROM o.order_purchase_timestamp) AS month,
        COUNT(o.order_id)
FROM `Target.orders` o JOIN `Target.customers` c ON o.customer_id=c.customer_id JOIN `Target.geolocation` g ON g.geolocation_zip_code_prefix=c.
customer_zip_code_prefix

GROUP BY g.geolocation_state,month
ORDER BY g.geolocation_state,month
```

Press Alt+F1 for Access

Query results

SAVE RESULTS ▾ EXPLORE DATA

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

Row	geolocation_state	month	f0_
1	AC	1	694
2	AC	2	515
3	AC	3	516
4	AC	4	789
5	AC	5	1161
6	AC	6	563
7	AC	7	937
8	AC	8	1060
9	AC	9	161

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

```sql
SELECT
    customer_state,
    COUNT(customer_unique_id) AS customer_count,
    ROUND((COUNT(customer_unique_id)/(SELECT COUNT(customer_unique_id) FROM `Target.customers`))*100,2) AS customer_dist
FROM `Target.customers`

GROUP BY customer_state
ORDER BY customer_count DESC
```

Query results

SAVE RESULTS

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

Row	customer_state	customer_count	customer_dist
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

NOTE: *In the above results the third column,* ***customer_dist****, signifies the distribution based on the percentage of users.*

Q4. **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 the 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.

```sql
SELECT ROUND((f.Total_2018-f.Total_2017) /f.Total_2017 *100,2) AS Percentage_increase
FROM
(SELECT
        SUM(CASE WHEN o.year=2017 THEN payment_value END) AS Total_2017,
        SUM(CASE WHEN o.year=2018 THEN payment_value END) AS Total_2018
  FROM `Target.payments` as q JOIN
(SELECT order_id,
        EXTRACT(month FROM order_purchase_timestamp) AS month,
        EXTRACT(year FROM order_purchase_timestamp) AS year
        FROM `Target.orders`) AS  o
ON q.order_id=o.order_id WHERE o.month BETWEEN 1 AND 8) AS f
```

Query results

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

Row	Percentage_increase
1	136.98

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

```
SELECT c.customer_state,
       ROUND(SUM(p.payment_value),2) as Total_price,
       ROUND(AVG(p.payment_value),2) as Avg_price
FROM `Target.payments` p JOIN `Target.orders` o ON p.order_id=o.order_id JOIN `Target.customers` c ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY Total_price DESC
```

Press

Query results

SAVE RESULTS

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

Row	customer_state	Total_price	Avg_price
1	SP	5998226.96	137.5
2	RJ	2144379.69	158.53
3	MG	1872257.26	154.71
4	RS	890898.54	157.18
5	PR	811156.38	154.15
6	SC	623086.43	165.98
7	BA	616645.82	170.82
8	DF	355141.08	161.13
9	GO	350092.31	165.76
10	ES	325967.55	154.71

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

```
SELECT c.customer_state,
       ROUND(SUM(p.freight_value),2) as Total_freight_cost,
       ROUND(AVG(p.freight_value),2) as Avg_freight_cost
FROM `Target.order_items` p JOIN `Target.orders` o ON p.order_id=o.order_id JOIN `Target.customers` c ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY Avg_freight_cost DESC
```

Press Alt+F1

Query results

SAVE RESULTS

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

Row	customer_state	Total_freight_cost	Avg_freight_cost
1	RR	2235.19	42.98
2	PB	25719.73	42.72
3	RO	11417.38	41.07
4	AC	3686.75	40.07
5	PI	21218.2	39.15
6	MA	31523.77	38.26
7	TO	11732.68	37.25
8	SE	14111.47	36.65
9	AL	15914.59	35.84
10	PA	38699.3	35.83

Q5. **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.

```
SELECT order_id,
       date_diff(order_delivered_customer_date, order_purchase_timestamp, day) AS time_to_deliver,
       date_diff(order_estimated_delivery_date,order_delivered_customer_date, day) AS diff_estimated_delivery
FROM `Target.orders`
```

Query results

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

Row	order_id	time_to_deliver	diff_estimated_delivery
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

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

Highest freight value ->

```
SELECT c.customer_state,
       ROUND(AVG(oi.freight_value),2) AS Avg_freight_value
FROM `Target.order_items` oi JOIN `Target.orders` o ON oi.order_id=o.order_id JOIN `Target.customers` c ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY Avg_freight_value DESC
```

Press Alt+F1 fo

Query results

SAVE RESULTS EXPLORE

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

Row	customer_state	Avg_freight_value
1	RR	42.98
2	PB	42.72
3	RO	41.07
4	AC	40.07
5	PI	39.15
6	MA	38.26
7	TO	37.25
8	SE	36.65
9	AL	35.84
10	PA	35.83

Results per page: 50 1 – 27 of 27

Lowest freight value ->

```
SELECT c.customer_state,
       ROUND(AVG(oi.freight_value),2) AS Avg_freight_value
FROM `Target.order_items` oi JOIN `Target.orders` o ON oi.order_id=o.order_id JOIN `Target.customers` c ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY Avg_freight_value DESC
```

Press Alt+F1

Query results

SAVE RESULTS EXPLO

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

Row	customer_state	Avg_freight_value
18	MS	23.37
19	GO	22.77
20	ES	22.06
21	RS	21.74
22	SC	21.47
23	DF	21.04
24	RJ	20.96
25	MG	20.63
26	PR	20.53
27	SP	15.15

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

   Top 5 states with highest delivery time ->

```sql
SELECT c.customer_state,
       ROUND(AVG(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2) AS Avg_delivery_time
FROM `Target.orders` o JOIN `Target.customers` c ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY Avg_delivery_time DESC
```

Query results

SAVE RESULTS

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

Row	customer_state	Avg_delivery_time
1	RR	28.98
2	AP	26.73
3	AM	25.99
4	AL	24.04
5	PA	23.32

Top 5 states with the lowest delivery time:

```
SELECT c.customer_state,
       ROUND(AVG(date_diff(o.order_delivered_customer_date, o.order_purchase_timestamp, day)),2) AS Avg_delivery_time
FROM `Target.orders` o JOIN `Target.customers` c ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY Avg_delivery_time DESC
```

Query results

SAVE RESULT

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

Row	customer_state	Avg_delivery_time
18	ES	15.33
19	MS	15.19
20	GO	15.15
21	RJ	14.85
22	RS	14.82
23	SC	14.48
24	DF	12.51
25	MG	11.54
26	PR	11.53
27	SP	8.3

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

```
SELECT c.customer_state,
       ROUND(AVG(date_diff(o.order_estimated_delivery_date,o.order_delivered_customer_date, day)),2) AS Avg_delivery_time

FROM `Target.orders` o JOIN `Target.customers` c ON o.customer_id=c.customer_id
GROUP BY c.customer_state
ORDER BY Avg_delivery_time DESC
```

Query results

SAVE RESULTS

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

Row	customer_state	Avg_delivery_time
1	AC	19.76
2	RO	19.13
3	AP	18.73
4	AM	18.61
5	RR	16.41

Q6. **Analysis based on the payments**

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

```
SELECT EXTRACT(YEAR from o.order_purchase_timestamp) AS year,
       EXTRACT(MONTH from o.order_purchase_timestamp) AS month,
       p.payment_type,
       COUNT(o.order_id) AS Count_orders
FROM `Target.orders` o JOIN `Target.payments` p ON o.order_id=p.order_id
GROUP BY year,month,p.payment_type
ORDER BY year, month,p.payment_type
```

Query results

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

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

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

```sql
SELECT
      p.payment_installments,
      COUNT(o.order_id) AS Count_orders
FROM `Target.orders` o JOIN `Target.payments` p ON o.order_id=p.order_id
GROUP BY 1
HAVING p.payment_installments>0
ORDER BY 1
```

Query results

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXECUT

Row	payment_installment	Count_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

Q6. **Actionable Insights & Recommendations**

1. We can deduct that the highest number of customers ordered during the afternoon time followed by night time. The lowest number of orders were during dawn time. RETAIL can give discounts or offers during the afternoon and night time to attract more customers during these periods of the day.
2. By analysis we see that while state RR has the highest average freight cost it has the lowest total order value, and contrary to this SA has the highest order value while having the lowest average freight cost. We can draw a conclusion that lower freight cost leads to a higher number of orders due to a reduction in the cost of an item.
3. State of SP has the highest number of customers (nearly 42%). The second state on the list has only one-third (nearly 12%) of SP. Steps should be taken to increase the user base in other cities.
4. One of the ways to do the above is to decrease the delivery time to these locations. For e.g., the state of RR has the highest delivery time but has the lowest customer base as well as the lowest order value. A lot of other states also fall under the same observation.
5. Lower number of installments has the highest number of orders. Steps could be taken to make these more attractive to customers by providing offers.
6. Most orders were done using credit cards followed by UPI. Discounts and offers could be provided on these methods to increase the user share and make it more attractive for the present users. Also, depending on strategy, if we want customers to use UPI more, we can provide various offers for specifically that.
7. Orders are increasing over the years in terms of both numbers well as total value. That signifies that market reach is increasing over the years and hence, marketing and other policies used so far have been effective.