TARGET CASE STUDY

1. **Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset**
    1. **Data type of columns in a table**

SQL QUERY

```
SELECT column_name, data_type
FROM `Target_Case_Study.INFORMATION_SCHEMA.COLUMNS`
WHERE table_name = 'customers';
```

OUTPUT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETA

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

    2. **Time period for which the data is given**

SQL Query

```
SELECT
MIN(order_purchase_timestamp) Starting_date,
MAX(order_purchase_timestamp) End_date
FROM `Target_Case_Study.orders`;
```

OUTPUT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

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

3. Cities and States of customers ordered during the given period

SQL Query

```
SELECT x.customer_unique_id, x.customer_city, x.customer_state
FROM `Target_Case_Study.customers` x
JOIN `Target_Case_Study.orders` y
ON x.customer_id = y.customer_id
WHERE y.order_purchase_timestamp Between
(SELECT MIN(order_purchase_timestamp) Starting_date FROM `Target_Case_Study.orders`
) AND
(SELECT MAX(order_purchase_timestamp) End_date FROM `Target_Case_Study.orders`)
ORDER BY x.customer_unique_id
LIMIT 10;
```

OR

```
    c.customer_unique_id,
    c.customer_city,
    c.customer_state
    FROM `Target_Case_Study.customers` AS c
    INNER JOIN `Target_Case_Study.orders` AS o
    ON c.customer_id = o.customer_id
    WHERE o.order_purchase_timestamp BETWEEN "2016-09-04 00:00:00" AND "2018-10-
    17 00:00:00"
    ORDER BY x.customer_unique_id
    LIMIT 10;
```

OUTPUT

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

Row	customer_unique_id	customer_city	customer_state
1	0000366f3b9a7992bf8c76cfdf...	cajamar	SP
2	0000b849f77a49e4a4ce2b2a4...	osasco	SP
3	0000f46a3911fa3c080544448...	sao jose	SC
4	0000f6ccb0745a6a4b88665a1...	belem	PA
5	0004aac84e0df4da2b147fca7...	sorocaba	SP
6	0004bd2a26a76fe21f786e4fbd...	sao paulo	SP
7	00050ab1314c0e55a6ca13cf7...	campinas	SP
8	00053a61a98854899e70ed20...	curitiba	PR
9	0005e1862207bf6ccc02e4228...	teresopolis	RJ
10	0005ef4cd20d2893f0d9fbd94d...	sao luis	MA

2. **In-depth Exploration:**
    1. **Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario? Can we see some seasonality with peaks at specific months?**

SQL Query

```
SELECT month,
COUNT(order_id) as Orders_per_month
FROM
(SELECT *, EXTRACT(month FROM o.order_purchase_timestamp) as month
FROM `Target_Case_Study.customers` c
JOIN `Target_Case_Study.orders` o
ON c.customer_id = o.customer_id)
Group by month
Order by month
```

OUTPUT

JOB INFORMATION | RESULTS

Row	month	Orders_per_mon
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
11	11	7544
12	12	5674

Insight: -

- The number of orders made by customers tends to increase from the beginning of the year until the middle of the year (June), and then starts to decline towards the end of the year.
- August is the month with the highest number of orders (10,843), while September has the lowest number of orders (4,305).
- There seems to be a spike in the number of orders in May (10,573), which may be due to seasonal factors such as Mother's Day or the approach of the summer season.
- The data suggests that the business may need to focus on boosting sales during the months of September to November, as they have lower numbers of orders compared to other months. This could involve marketing campaigns or promotions targeted towards customers during these months.

- October and November also have relatively low numbers of orders compared to the other months.

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

SQL Query

```
SELECT
CASE
  WHEN Time BETWEEN '00:00:00' AND '06:00:00' then 'Dawn'
  WHEN Time BETWEEN '06:00:01' AND '12:00:00' then 'Morning'
  WHEN Time BETWEEN '12:00:01' AND '18:00:00' then 'Afternoon'
  ELSE 'Night'
END AS TIMES,
COUNT (order_id) as total_orders
FROM
(SELECT *,
EXTRACT(time from order_purchase_timestamp) as Time
FROM `Target_Case_Study.orders`)
Group by Times;
```

OUTPUT

JOB INFORMATION | RESULTS

Row	TIMES	total_orders
1	Morning	22240
2	Dawn	4740
3	Afternoon	38365
4	Night	34096

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

1. Get month on month orders by states

SQL QUERY

```
SELECT
EXTRACT(MONTH FROM order_purchase_timestamp) Month,
EXTRACT(YEAR FROM order_purchase_timestamp) Year,
c.customer_state,
COUNT(order_id) as Total_orders
FROM  `Target_Case_Study.orders` o
JOIN `Target_Case_Study.customers` c
ON o.customer_id = c.customer_id
GROUP BY Month, Year, customer_state
Order By Month, Year
LIMIT 10
```

OUTPUT

Row	Month	Year	customer_state	Total_orders
1	1	2017	BA	25
2	1	2017	MG	108
3	1	2017	ES	12
4	1	2017	MT	11
5	1	2017	SP	299
6	1	2017	PR	65
7	1	2017	PA	12
8	1	2017	MA	9
9	1	2017	RJ	97
10	1	2017	SC	31

2. Distribution of customers across the states in Brazil

SQL QUERY

```
SELECT customer_state,
COUNT(*) AS customer_count
FROM `Target_Case_Study.customers`
GROUP BY customer_state
ORDER BY customer_count DESC
LIMIT 10;
```

OUTPUT

Row	customer_state	customer_count
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

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

SQL Query

```
WITH DJ AS
(SELECT
  EXTRACT(YEAR FROM o.order_purchase_timestamp) as YEAR,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) as MONTH,
  p.payment_value
FROM `Target_Case_Study.orders`o
JOIN `Target_Case_Study.payments`p
ON o.order_id = p.order_id)

SELECT
  ROUND(total_payment_value_2017,2) Payment_value_2017,
  ROUND(total_payment_value_2018,2) Payment_value_2018,
  CONCAT (ROUND (SUM (((total_payment_value_2018- total_payment_value_2017) / total_pa
yment_value_2017)*100),2), "%") as cost_increase_percentage
FROM
(SELECT
  SUM (CASE
      WHEN YEAR = 2017 AND MONTH BETWEEN 1 AND 8 then payment_value
      ELSE 0
    END) AS total_payment_value_2017,
  SUM (CASE
      WHEN YEAR = 2018 AND MONTH BETWEEN 1 AND 8 then payment_value
      ELSE 0
    END) AS total_payment_value_2018
FROM DJ)
GROUP BY total_payment_value_2017, total_payment_value_2018;
```

OUTPUT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	Payment_value_2017	Payment_value_2018	cost_increase_percentage
1	3669022.12	8694733.84	136.98%

2. Mean & Sum of price and freight value by customer state

SQL Query

```
SELECT
    c.customer_state,
    ROUND (SUM(oi.price),2) Sum_Price,
    ROUND(SUM(oi.freight_value),2) Sum_freight_value,
    ROUND (AVG(oi.price),2) AVG_Price,
    ROUND (AVG(oi.freight_value),2)Avg_freight_value
FROM `Target_Case_Study.customers` c
JOIN `Target_Case_Study.orders` o
  ON c.customer_id =o.customer_id
JOIN `Target_Case_Study.order_items` oi
  ON o.order_id = oi.order_id
GROUP BY customer_state
ORDER BY customer_state
LIMIT 10;
```

OUTPUT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS | EXE

Row	customer_state	Sum_Price	Sum_freight_value	AVG_Price	Avg_freight_value
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

5. Analysis on sales, freight, and delivery time

1. Calculate days between purchasing, delivering and estimated delivery.

SQL Query

```
        SELECT
        Order_id,
        DATE_DIFF(order_delivered_carrier_date, order_purchase_timestamp, day) as de
        livery_day_diff,
        DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp, day) as E
        simated_day_diff,
        FROM `Target_Case_Study.orders`
        ORDER BY order_id
        LIMIT 10
```

OUTPUT

Row	Order_id	delivery_day_diff	Esimated_day_diff
1	00010242fe8c5a6d...	6	15
2	00018f77f2f0320c5...	8	18
3	000229ec398224ef...	1	21
4	00024acbcdf0a6da...	2	11
5	00042b26cf59d7ce...	11	40
6	00048cc3ae777c65...	1	21
7	00054e8431b9d767...	1	24
8	000576fe39319847...	1	20
9	0005a1a1728c9d78...	8	9
10	0005f50442cb953d...	1	20

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

SQL Query

```
SELECT
  order_id,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) ti
me_to_delivery,
  DATE_DIFF(order_estimated_delivery_date,order_delivered_customer_date,day)
 diff_estimated_delivery
  FROM `Target_Case_Study.orders`
  ORDER BY order_id
  LIMIT 10;
```

OUTPUT

JOB INFORMATION | RESULTS | JSON | EXECUTION DETAILS

Row	order_id	time_to_delivery	diff_estimated_delivery
1	00010242fe8c5a6d1...	7	8
2	00018f77f2f0320c55...	16	2
3	000229ec398224ef6...	7	13
4	00024acbcdf0a6daa...	6	5
5	00042b26cf59d7ce6...	25	15
6	00048cc3ae777c65d...	6	14
7	00054e8431b9d7675...	8	16
8	000576fe39319847c...	5	15
9	0005a1a1728c9d785...	9	0
10	0005f50442cb953dc...	2	18

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

SQL QUERY

```
SELECT
  c.customer_state AS state,
  AVG (oi.freight_value) AS avg_freight_value,
  AVG (DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp
, day)) AS time_to_delivery,
  AVG (DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer
_date, day)) AS diff_estimated_delivery
FROM `Target_Case_Study.customers` c
  JOIN `Target_Case_Study.orders` o ON c.customer_id = o.customer_id
  JOIN `Target_Case_Study.order_items` oi ON o.order_id = oi.order_id
GROUP BY c.customer_state
ORDER BY avg_freight_value DESC
LIMIT 10;
```

OUTPUT

Row	state	avg_freight_valu	time_to_delivery	diff_estimated_d
1	RR	42.9844230...	27.8260869...	17.4347826...
2	PB	42.7238039...	20.1194539...	12.1501706...
3	RO	41.0697122...	19.2820512...	19.0805860...
4	AC	40.0733695...	20.3296703...	20.0109890...
5	PI	39.1479704...	18.9311663...	10.6826003...
6	MA	38.2570024...	21.2037499...	9.10999999...
7	TO	37.2466031...	17.0032258...	11.4612903...
8	SE	36.6531688...	20.9786666...	9.16533333...
9	AL	35.8436711...	23.9929742...	7.97658079...
10	PA	35.8326851...	23.3017077...	13.3747628...

4. Sort the data to get the following:

```
WITH D AS (
SELECT
  c.customer_state AS state,
  AVG (oi.freight_value) AS avg_freight_value,
  AVG (DATE_DIFF(o.order_delivered_customer_date, o.order_purchase_timestamp, day))
 AS time_to_delivery,
  AVG (DATE_DIFF(o.order_estimated_delivery_date, o.order_delivered_customer_date,
day)) AS diff_estimated_delivery
FROM `Target_Case_Study.customers` c
  JOIN `Target_Case_Study.orders` o ON c.customer_id = o.customer_id
  JOIN `Target_Case_Study.order_items` oi ON o.order_id = oi.order_id
GROUP BY c.customer_state)
```

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

A. Top 5 states with highest average freight value - sort in desc limit 5

SQL QUERY

```
SELECT D.state,ROUND(D.avg_freight_value,2) as highest_f_v
FROM D
ORDER BY D.avg_freight_value desc
LIMIT 5;
```

NOTE:- please refer CTE for table D

OUTPUT

JOB INFORMATION | RESULTS

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

B. Top 5 states with Lowest average freight value - sort in asc limit 5

SQL Query

```
SELECT D.state,ROUND(D.avg_freight_value,2) as lowest_f_v
FROM D
ORDER BY D.avg_freight_value ASC
LIMIT 5;
```

OUTPUT

JOB INFORMATION | RESULTS

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

6. Top 5 states with highest/lowest average time to delivery

A. TOP 5 states with highest average time to delivery

SQL Query

```
SELECT D.state,ROUND(D.time_to_delivery,2) as highest_t_2_d
FROM D
ORDER BY D.time_to_delivery desc
LIMIT 5;
```

OUTPUT

JOB INFORMATION | RESULTS

Row	state	highest_t_2_d
1	RR	27.83
2	AP	27.75
3	AM	25.96
4	AL	23.99
5	PA	23.3

B. Top 5 states with lowest average time to delivery

SQL Query

```
SELECT D.state,ROUND(D.time_to_delivery,2) as lowest_t_2_d
FROM D
ORDER BY D.time_to_delivery ASC
LIMIT 5;
```

OUTPUT

JOB INFORMATION | RESULTS

Row	state	lowest_t_2_d
1	SP	8.26
2	PR	11.48
3	MG	11.52
4	DF	12.5
5	SC	14.52

7. Top 5 states where delivery is fast / not so fast compared to estimated date

SQL Query

```
SELECT D.state,ROUND(D.AVG_diff_estimated_delivery,2) as fast
FROM D
ORDER BY D.AVG_diff_estimated_delivery ASC
LIMIT 5;
```

OUTPUT

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

8. Top 5 states where delivery is not so fast compared to estimated date.

SQL Query

```
SELECT D.state,ROUND(D.AVG_diff_estimated_delivery,2) as not_so_fast
FROM D
ORDER BY D.AVG_diff_estimated_delivery DESC
LIMIT 5;
```

OUTPUT

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

5. **Payment type analysis:**

    1. **Month over Month count of orders for different payment types.**

SQL QUERY

```
SELECT
EXTRACT(MONTH FROM o.order_purchase_timestamp) as Month,
payment_type,
COUNT(p.order_id) as total_order,
FROM `Target_Case_Study.payments` p
JOIN `Target_Case_Study.orders` o ON p.order_id = o.order_id
GROUP BY 1,2
Order BY 1,2
LIMIT 10;
```

OUTPUT

JOB INFORMATION | RESULTS | JSON | E

Row	Month	payment_type	total_order
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

Insight:-

- There are three payment types represented in the table: UPI, credit card, debit card, and vouchers.
- Based on this information, we can see that credit card is the most popular payment type among the three, followed by UPI and debit card. The use of vouchers is relatively low in comparison.

2 Count of orders based on the no. of payment installments

SQL Query

```
SELECT
payment_installments,
COUNT(DISTINCT order_id) total_orders
FROM `Target_Case_Study.payments`
GROUP BY payment_installments
ORDER BY 1
LIMIT 10;
```

OUTPUT

JOB INFORMATION | RESULTS | JS

Row	payment_installments	total_orders
1	0	2
2	1	49060
3	2	12389
4	3	10443
5	4	7088
6	5	5234
7	6	3916
8	7	1623
9	8	4253
10	9	644

Insight- Based on this information, we can see that as the number of payment installments increases, the total amount of orders placed by the customer decreases. This could indicate that customers who choose to pay in installments may be more price-sensitive and may not be able to afford as many orders.