# BUSINESS CASESTUDY-SCALER

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 target_SQL.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:

There are 5 columns in customers table:

1.customer_id,
2.customer_unique_id,
3.customer_zip_code_prefix,
4.customer_city,
5.customer_state

Out of all **5** only **customer_zip_code_prefix is INT64** and **others are STRING** datatype.

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

Query:

```
SELECT
    DATE(MIN(order_purchase_timestamp)) AS first_date,
    TIME(MIN(order_purchase_timestamp)) AS start_time,
    DATE(MAX(order_purchase_timestamp)) AS last_time,
    TIME(MAX(order_purchase_timestamp)) AS end_time
FROM `target_SQL.orders`;
```

Output:

`

| Row | first_date | start_time | last_time | end_time |
|---|---|---|---|---|
| 1 | 2016-09-04 | 21:15:19 | 2018-10-17 | 17:30:18 |

Insight:

Customers started ordering form **04th September 2016** at **09:15PM** and they were ordering till **17th October 2018** at **05:30PM** making it **2 years 1 month and 13 days** of ordering to be precise.

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

Query:

```
WITH given_period AS(
    SELECT
        MIN(order_purchase_timestamp) AS start_time,
        MAX(order_purchase_timestamp) AS end_time
    FROM `target_SQL.orders`
)
SELECT
    COUNT(DISTINCT c.customer_id)AS no_of_customers,
    COUNT(DISTINCT c.customer_city) AS no_of_cities,
    COUNT(DISTINCT c.customer_state) AS no_of_states
FROM `target_SQL.orders` o join `target_SQL.customers` c
ON o.customer_id=c.customer_id
WHERE o.order_purchase_timestamp BETWEEN (SELECT start_time
FROM given_period) AND (SELECT end_time FROM given_period);
```

Output:

| Row | no_of_customers | no_of_cities | no_of_states |
|---|---|---|---|
| 1 | 99441 | 4119 | 27 |

Insight:

The orders were done by **99441 customers** from **27 States** and **4119 cities**.

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 years,
  COUNT(DISTINCT order_id) AS no_of_orders
FROM `target_SQL.orders`
GROUP BY years
ORDER BY years;
```

    Output:

| Row | years | no_of_orders |
|---|---|---|
| 1 | 2016 | 329 |
| 2 | 2017 | 45101 |
| 3 | 2018 | 54011 |

    Insight:

- In year 2016 had just **329 orders**.
- In **year 2017** the orders increased to **45101** which is approximately **13609% more than** 2016 representing an **increase of more than 136 times**.
- IN **year 2018** the orders again increased to **54011** which is approximately **20% more than 2017** making it **1.2 times more than 2017** and **329 times more than 2016**.
- So overall the **number or orders have increased per year.**

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 months,
  COUNT(order_id) AS no_of_orders
FROM `target_SQL.orders`
GROUP BY months,year
ORDER BY year,months;
```

Output:

2016-

| Row | year | months | no_of_orders |
|---|---|---|---|
| 1 | 2016 | 9 | 4 |
| 2 | 2016 | 10 | 324 |
| 3 | 2016 | 12 | 1 |

2017-

| | | | |
|---|---|---|---|
| 4 | 2017 | 1 | 800 |
| 5 | 2017 | 2 | 1780 |
| 14 | 2017 | 11 | 7544 |
| 15 | 2017 | 12 | 5673 |

2018-

| | | | |
|---|---|---|---|
| 16 | 2018 | 1 | 7269 |
| 24 | 2018 | 9 | 16 |
| 25 | 2018 | 10 | 4 |

Insight:

- We can see that there is **inconsistency in the data**.
- Year 2016 has just 3 months September, October and December.
- Year 2017 has data for whole year.
- Year 2018 has data for just starting 10 months.
- For Year **2016 and 2017** there were **increase in order numbers** from **September to October** but for Year **2018** we can see **a dip**.
- For Year **2017 and 2018** there is a **dip in orders** from **March to April** and from **May to June** and **increase** in orders from **June to July.**
- But over all there is an increase in number of orders placed per year.

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

    Query:

```
SELECT
CASE
WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 0 AND
EXTRACT(HOUR FROM order_purchase_timestamp) <= 6
THEN 'Dawn'

WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 7 AND
EXTRACT(HOUR FROM order_purchase_timestamp) <= 12
THEN 'Morning'

WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 13 AND
EXTRACT(HOUR FROM order_purchase_timestamp) <= 18
THEN 'Afternoon'

WHEN EXTRACT(HOUR FROM order_purchase_timestamp) >= 19 AND
EXTRACT(HOUR FROM order_purchase_timestamp) <= 23
THEN 'Night'

END AS time_of_day,
COUNT(DISTINCT order_id) AS order_count
FROM `target_SQL.orders`
GROUP BY time_of_day
ORDER BY order_count DESC;
```

    Output:

| Row | time_of_day | order_count |
|---|---|---|
| 1 | Afternoon | 38135 |
| 2 | Night | 28331 |
| 3 | Morning | 27733 |
| 4 | Dawn | 5242 |

    Insight:

    - Brazilian customers tend to order the most during Afternoon i.e., from 01:00Pm to 06:00PM.
    - Brazilian customers order the least during Dawn i.e., between 12:00AM to 06:00AM.

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

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

Query:

```
SELECT
    c.customer_state as state,
    EXTRACT(month FROM o.order_purchase_timestamp) AS order_month,
    COUNT(o.order_id) as total_orders,
    SUM(CASE
            WHEN EXTRACT(YEAR FROM order_purchase_timestamp)=2016
            THEN 1
            ELSE 0
        END) AS orders_2016,
    SUM(CASE
            WHEN EXTRACT(YEAR FROM order_purchase_timestamp)=2017
            THEN 1
            ELSE 0
        END) AS orders_2017,
    SUM(CASE
            WHEN EXTRACT(YEAR FROM order_purchase_timestamp)=2018
            THEN 1
            ELSE 0
        END) AS orders_2018,
FROM `target_SQL.orders` AS o
JOIN `target_SQL.customers` AS c
ON c.customer_id =o.customer_id
GROUP BY order_month,customer_state
ORDER BY state,order_month;
```

Output:

| Row | state | order_month | total_orders | orders_2016 | orders_2017 | orders_2018 |
|---|---|---|---|---|---|---|
| 1 | AC | 1 | 8 | 0 | 2 | 6 |
| 2 | AC | 2 | 6 | 0 | 3 | 3 |
| 3 | AC | 3 | 4 | 0 | 2 | 2 |
| 4 | AC | 4 | 9 | 0 | 5 | 4 |
| 5 | AC | 5 | 10 | 0 | 8 | 2 |
| 6 | AC | 6 | 7 | 0 | 4 | 3 |
| 7 | AC | 7 | 9 | 0 | 5 | 4 |
| 8 | AC | 8 | 7 | 0 | 4 | 3 |
| 9 | AC | 9 | 5 | 0 | 5 | 0 |
| 10 | AC | 10 | 6 | 0 | 6 | 0 |

Insight:

- The **Maximum** number of **orders** were received by **state SP** in **August** month i.e. **4982**.
- The **Minimum** number of orders were received by state **RR** in **September, November and January** and **AP** in **October** month i.e. **2** respectively.

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

```
SELECT
  customer_state,
  COUNT(DISTINCT customer_id) AS no_of_customers
FROM `target_SQL.customers`
GROUP BY customer_state
ORDER BY no_of_customers DESC;
```

   Output:

| Row | customer_state | no_of_customers |
|---|---|---|
| 1 | SP | 41746 |
| 2 | RJ | 12852 |
| 3 | MG | 11635 |
| 4 | RS | 5466 |
| 5 | PR | 5045 |
| 23 | RO | 253 |
| 24 | AM | 148 |
| 25 | AC | 81 |
| 26 | AP | 68 |
| 27 | RR | 46 |

   Insight:

   - **Highest** number of customers are from **SP**.
   - **Lowest** number of customers are from **RR** .

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:

```
WITH yearly_payment_values AS(
  WITH Jan_to_Aug AS(
    SELECT
      order_id,
      EXTRACT(YEAR FROM order_purchase_timestamp) AS year,
      EXTRACT(MONTH FROM order_purchase_timestamp) AS months,
    FROM `target_SQL.orders`
    WHERE EXTRACT(YEAR FROM order_purchase_timestamp) IN (2017,2018)
    AND EXTRACT(MONTH FROM order_purchase_timestamp) BETWEEN 1 AND 8
)
  SELECT
    ja.year,
    COUNT(p.payment_value) AS yearly_payment_value
  FROM `target_SQL.payments` p JOIN Jan_to_Aug ja
  ON p.order_id=ja.order_id
  GROUP BY ja.year
)
SELECT DISTINCT ROUND(
  (((SELECT yearly_payment_value FROM yearly_payment_values WHERE
year=2018)-(SELECT yearly_payment_value FROM yearly_payment_values
WHERE year=2017))/(SELECT yearly_payment_value FROM
yearly_payment_values WHERE year=2017))*100,2) AS percentage_increase
FROM yearly_payment_values;
```

Output:

| Row | percentage_increase |
|---|---|
| 1 | 129.57 |

Insight:

- The cost of orders **increased by 129.57** approximately from year **2017 to 2018** for the months **January to August**.

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

Query:

```
WITH order_payment AS(
      SELECT
        o.order_id,
        o.customer_id,
        p.payment_value
      FROM `target_SQL.orders` o JOIN `target_SQL.payments` p
      ON o.order_id=p.order_id
)
SELECT
  c.customer_state,
  ROUND(SUM(op.payment_value),2) AS total_order_price,
  ROUND(AVG(op.payment_value),2) AS avg_order_price
FROM order_payment op JOIN `target_SQL.customers` c
ON op.customer_id=c.customer_id
GROUP BY customer_state
ORDER BY total_order_price DESC;
```

Output:

| Row | customer_state | total_order_price | avg_order_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 |

Insight:

- **Total order price** is **maximum** for state **SP** and **minimum** for state **RR**.
- **Average order price** is **maximum** for state **PB** and **minimum** for state **SP**.

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

Query:

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

Output:

| Row | customer_state | total_freight_value | average_freight_valu |
|---|---|---|---|
| 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 |

Insight:

- **Total freight value** is **maximum** for state **SP** and **minimum** for state **RR**.
- **Average freight value** is **maximum** for state **RR** and **minimum** for state **SP**.

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_estimated_delivery_date - order_delivered_customer_date

Query:

```
SELECT order_id,
       DATE_DIFF(order_delivered_carrier_date,order_purchase_timestamp
,DAY) AS time_to_deliver,
       DATE_DIFF(order_estimated_delivery_date,order_purchase_timestam
p,DAY) AS diff_estimated_delivery
FROM `target_SQL.orders`;
```

Output:

| Row | order_id | time_to_deliver | diff_estimated_delivery |
|---|---|---|---|
| 1 | f88aac7ebccb37f19725a0753... | 9 | 50 |
| 2 | 790cd37689193dca0d00d2feb... | 2 | 6 |
| 3 | 49db7943d60b6805c3a41f547... | 6 | 44 |
| 4 | 063b573b88fc80e516aba87df... | 22 | 54 |
| 5 | a68ce1686d536ca72bd2dadc4... | 33 | 56 |
| 6 | 45973912e490866800c0aea8f... | 18 | 54 |
| 7 | cda873529ca7ab71f677d5ec1... | 39 | 56 |
| 8 | ead20687129da8f5d89d831bb... | 1 | 41 |
| 9 | 6f028ccb7d612af251aa442a1f... | 1 | 3 |
| 10 | 8733c8d440c173e524d2fab80... | 0 | 3 |

Insight:

- With the output we can determine the difference in **estimated date to delivery** and **the actual time it took for the delivery**.
- Using this we can improve our customer satisfaction by decreasing delivery time.

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

Query:

```
WITH main_table AS(
  SELECT
    c.customer_state,
    ROUND(AVG(freight_value),2) AS average_freight_value
  FROM `target_SQL.order_items` oi JOIN `target_SQL.orders` o
  ON o.order_id=oi.order_id
  JOIN `target_SQL.customers` c
  ON o.customer_id=c.customer_id
  GROUP BY c.customer_state
),
top_5 AS(
  SELECT
    customer_state,
    main_table.average_freight_value
  FROM main_table
  ORDER BY average_freight_value DESC
  LIMIT 5
),
bottom_5 AS(
  SELECT
    customer_state,
    main_table.average_freight_value
  FROM main_table
  ORDER BY average_freight_value
  LIMIT 5
)
SELECT customer_state AS states,top_5.average_freight_value AS
avg_freight_value  FROM top_5
UNION ALL
SELECT customer_state,bottom_5.average_freight_value  FROM bottom_5;
```

Output:

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

Insight:

- **RR, PB, RO, AC** and **PI** are the **5 states** with **highest average freight value**.
- **SP, PR, MG, RJ** and **DF** are the **5 states** with **lowest average freight value.**

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

Query:

```
WITH avg_del_table AS(
WITH deliver_time AS(
SELECT
  customer_id,
  DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,DAY
) AS del_time
FROM `target_SQL.orders`
)
SELECT
  c.customer_state,
  ROUND(AVG(d.del_time),2) AS avg_deliver_time
FROM deliver_time d JOIN `target_SQL.customers` c
ON d.customer_id=c.customer_id
GROUP BY customer_state
),
top_5 AS(
  SELECT * FROM avg_del_table
  ORDER BY avg_del_table.avg_deliver_time DESC
  LIMIT 5
),
bottom_5 AS(
  SELECT * FROM avg_del_table
  ORDER BY avg_del_table.avg_deliver_time
  LIMIT 5
)
SELECT * FROM top_5
UNION ALL
SELECT * FROM bottom_5;
```

Output:

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

Insight:

- **RR, AP, AM, AL** and **PA** are the **5 states** with **highest average delivery time**.
- **SP, PR, MG, DF** and **SC** ate the **5 states** with **lowest 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:

```
WITH time AS(
  SELECT
    customer_id,
    DATE_DIFF(order_delivered_customer_date,order_purchase_timestamp,D
AY) AS deliver_time,
    DATE_DIFF(order_estimated_delivery_date,order_purchase_timestamp,D
AY) AS estimated_time
  FROM `target_SQL.orders`
)
SELECT
  c.customer_state,
  ROUND(AVG(t.estimated_time)-AVG(t.deliver_time),2) AS diff_time
FROM time t JOIN `target_SQL.customers` c
ON t.customer_id=c.customer_id
GROUP BY customer_state
ORDER BY diff_time DESC
LIMIT 5;
```

Output:

| Row | customer_state | diff_time |
|---|---|---|
| 1 | AC | 20.13 |
| 2 | RO | 19.49 |
| 3 | AP | 18.97 |
| 4 | AM | 18.77 |
| 5 | RR | 17.2 |

Insight:

- **AC, RO, AP, AM** and **RR** are the **top 5 states** where the **order delivery** is **really fast as compared to the estimated date** of delivery.

6. **Analysis based on the payments:**
    1. Find the month on month no. of orders placed using different payment types.

Query:

```
WITH order_t AS(
SELECT
  order_id,
  EXTRACT(MONTH FROM order_purchase_timestamp) AS months
FROM `target_SQL.orders`
)
SELECT
  p.payment_type,
  o.months,
  count(o.order_id) AS no_of_orders
FROM order_t o JOIN `target_SQL.payments` p
ON o.order_id=p.order_id
GROUP BY payment_type,months
ORDER BY payment_type,months;
```

Output:

| Row | payment_type | months | no_of_orders |
|---|---|---|---|
| 1 | UPI | 1 | 1715 |
| 2 | UPI | 2 | 1723 |
| 3 | UPI | 3 | 1942 |
| 4 | UPI | 4 | 1783 |
| 5 | UPI | 5 | 2035 |
| 6 | UPI | 6 | 1807 |
| 7 | UPI | 7 | 2074 |
| 8 | UPI | 8 | 2077 |
| 9 | UPI | 9 | 903 |
| 10 | UPI | 10 | 1056 |

Insight:

- The **payments** are done by **4 methods**: **UPI, Credit Card, Debit Card and Vouchers**.
- The **maximum no of orders** received on a particular month by a particular method are received on month **May** using **Credit Cards**.
- **Most** number of orders are done **using Credit Cards**.
- **Debit Cards** are the **least used** payment **method**.

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

Query:

```
SELECT
  payment_installments,
  COUNT(order_id) AS no_of_orders
FROM `target_SQL.payments`
GROUP BY payment_installments
ORDER BY payment_installments;
```

Output:

| Row | payment_installment | no_of_orders |
|---|---|---|
| 1 | 0 | 2 |
| 2 | 1 | 52546 |
| 3 | 2 | 12413 |
| 4 | 3 | 10461 |
| 5 | 4 | 7098 |
| 6 | 5 | 5239 |
| 7 | 6 | 3920 |
| 8 | 7 | 1626 |
| 9 | 8 | 4268 |
| 10 | 9 | 644 |

Insight:

- **Most payments** are done by using just **one instalment**.
- **Customers prefer** buying the product in just **one time payment**.