# TARGET SQL BUSINESS CASE

## PROJECT

## DESCRIPTION:

Target is a globally renowned brand and a prominent retailer in the United States. Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation, and an exceptional guest experience that no other retailer can deliver.

This Particular business case focuses on the operations of Target in Brazil and provides insightful information about 100,000 orders placed between 2016 and 2018. The dataset offers a comprehensive view of various dimensions including the order status, price, payment and freight performance, customer location, product attributes, and customer reviews.

By analysing this extensive dataset, it becomes possible to gain valuable insights into Target's operations in Brazil. The information can shed light on various aspects of the business, such as order processing, pricing strategies, payment and shipping efficiency, customer demographics, product characteristics, and customer satisfaction levels.

---

**Dataset**: https://drive.google.com/drive/folders/1TGEc66YKbD443nslRi1bWgVd238gJCnb

The data is available in 8 csv files:

1. customers.csv
2. sellers.csv
3. order_items.csv
4. geolocation.csv
5. payments.csv
6. reviews.csv
7. orders.csv
8. products.csv

---

The column description for these csv files is given below.

The **customers.csv** contain following features:

| Features | Description |
|---|---|
| customer_id | ID of the consumer who made the purchase |
| customer_unique_id | Unique ID of the consumer |
| customer_zip_code_prefix | Zip Code of consumer's location |
| customer_city | Name of the City from where order is made |
| customer_state | State Code from where order is made (Eg. são paulo - SP) |

The **sellers.csv** contains following features:

| Features | Description |
|---|---|
| **seller**_id | Unique ID of the seller registered |
| seller_zip_code_prefix | Zip Code of the seller's location |
| seller_city | Name of the City of the seller |
| seller_state | State Code (Eg. são paulo - SP) |

The **order_items.csv** contain following features:

| Features | Description |
|---|---|
| order_id | A Unique ID of order made by the consumers |
| order_item_id | A Unique ID given to each item ordered in the order |
| product_id | A Unique ID given to each product available on the site |
| seller_id | Unique ID of the seller registered in Target |
| shipping_limit_date | The date before which the ordered product must be shipped |
| price | Actual price of the products ordered |
| freight_value | Price rate at which a product is delivered from one point to another |

The **geolocations.csv** contain following features:

| Features | Description |
|---|---|
| geolocation_zip_code_prefix | First 5 digits of Zip Code |
| geolocation_lat | Latitude |
| geolocation_lng | Longitude |
| geolocation_city | City |
| geolocation_state | State |

The **payments.csv** contain following features:

| Features | Description |
|---|---|
| order_id | A Unique ID of order made by the consumers |
| payment_sequential | Sequences of the payments made in case of EMI |
| payment_type | Mode of payment used (Eg. Credit Card) |
| payment_installments | Number of installments in case of EMI purchase |
| payment_value | Total amount paid for the purchase order |

The **orders.csv** contain following features:

| Features | Description |
|---|---|
| order_id | A Unique ID of order made by the consumers |
| customer_id | ID of the consumer who made the purchase |
| order_status | Status of the order made i.e. delivered, shipped, etc. |
| order_purchase_timestamp | Timestamp of the purchase |
| order_delivered_carrier_date | Delivery date at which carrier made the delivery |
| order_delivered_customer_date | Date at which customer got the product |
| order_estimated_delivery_date | Estimated delivery date of the products |

The **reviews.csv** contain following features:

| Features | Description |
|---|---|
| review_id | ID of the review given on the product ordered by the order id |
| order_id | A Unique ID of order made by the consumers |
| review_score | Review score given by the customer for each order on a scale of 1-5 |
| review_comment_title | Title of the review |
| review_comment_message | Review comments posted by the consumer for each order |
| review_creation_date | Timestamp of the review when it is created |
| review_answer_timestamp | Timestamp of the review answered |

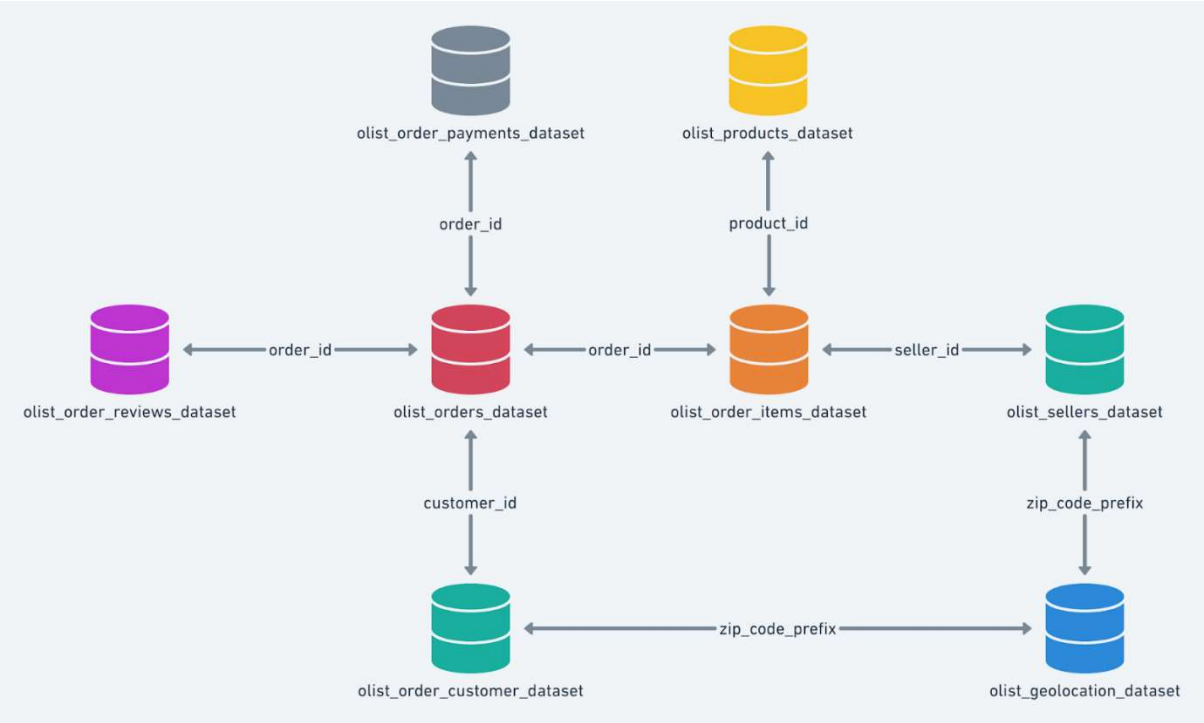
The **products.csv** contain following features:

| Features | Description |
|---|---|
| product_id | A Unique identifier for the proposed project. |
| product_category_name | Name of the product category |

| | |
|---|---|
| product_name_lenght | Length of the string which specifies the name given to the products ordered |
| product_description_lenght | Length of the description written for each product ordered on the site |
| product_photos_qty | Number of photos of each product ordered available on the shopping portal |
| product_weight_g | Weight of the products ordered in grams |
| product_length_cm | Length of the products ordered in centimeters |
| product_height_cm | Height of the products ordered in centimeters |
| product_width_cm | Width of the product ordered in centimeters |

---

**Dataset schema:**



**Problem Statement:**

**What does 'good' look like?**

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.

**Code Explanation-**

- MIN and Max Aggregate Functions used.
- Table Used = Orders
- Group by order Date (to check or count no. of orders placed per day)
- Extract is used to segregate Date and Time.

**Query:**

```
SELECT

table_name,

column_name,

data_type

FROM `Target_SQL_Business_Case`.INFORMATION_SCHEMA.COLUMNS

WHERE table_name ='customers'
```

**Result:**

| Row | table_name | column_name | data_type |
|---|---|---|---|
| 1 | customers | customer_id | STRING |
| 2 | customers | customer_unique_id | STRING |
| 3 | customers | customer_zip_code_prefix | INT64 |
| 4 | customers | customer_city | STRING |
| 5 | customers | customer_state | STRING |

**Insight:**

- We get data type of all the column in customers table.
- All column have String data type except column ‘customer_zip_code_prefix’ having INT data type.

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

FIRST APPROACH: -

**Query:**

```
SELECT
MIN(order_purchase_timestamp) as first_order,

MAX(order_purchase_timestamp) as last_order

FROM `Target_SQL_Business_Case.orders`
```

**Result:**

| Row | first_order | last_order |
|---|---|---|
| 1 | 2016-09-04 21:15:19 UTC | 2018-10-17 17:30:18 UTC |

**Insight:**

- First order was  placed on 4th September 2016.
- Last no. of order placed on 17th October 2018.

**OR**

SECOND APPROACH: -

**Query:**

```
SELECT
ss.order_date,

COUNT(order_id) as order_count_per_day,

MIN(ss.order_time) as order_start_time,

MAX(ss.order_time) as order_end_time

FROM (

SELECT

order_id,

customer_id,

EXTRACT(date from order_purchase_timestamp) as order_date,
```

```
        EXTRACT( TIME FROM order_purchase_timestamp ) as order_time
        FROM `Target_SQL_Business_Case.orders`
        ) as ss
        GROUP BY ss.order_date
        ORDER BY order_date
        LIMIT 10
        OFFSET 5
```

**Result:**

| Row | order_date | order_count_per_day | order_start_time | order_end_time |
|---|---|---|---|---|
| 1 | 2016-10-03 | 8 | 09:44:50 | 22:51:30 |
| 2 | 2016-10-04 | 63 | 09:06:10 | 23:59:01 |
| 3 | 2016-10-05 | 47 | 00:32:31 | 23:14:34 |
| 4 | 2016-10-06 | 51 | 00:06:17 | 23:49:18 |
| 5 | 2016-10-07 | 46 | 00:54:40 | 23:18:38 |
| 6 | 2016-10-08 | 42 | 01:28:14 | 23:46:06 |
| 7 | 2016-10-09 | 26 | 00:56:52 | 23:55:30 |
| 8 | 2016-10-10 | 39 | 00:01:50 | 18:09:39 |
| 9 | 2016-10-22 | 1 | 08:25:27 | 08:25:27 |
| 10 | 2016-12-23 | 1 | 23:16:47 | 23:16:47 |

**Insight:**

- Get count of orders placed per day to the orders time range.
- Give clear picture of growing no. of orders from first order and last order.

**ADDITIONAL QUESTION:**

- ❖ Which region is on trend?
- ❖ Which day/year has highest no. of orders placed?

3) Count the number of Cities and States in our dataset.

**Code Explanation-**

- DISTINCT is used to COUNT city and state.
- Table Used = geolocation and customers

**Query:**

```
SELECT
COUNT(distinct customer_city) AS city_count,
COUNT(distinct customer_state) AS state_count
FROM `Target_SQL_Business_Case.customers`
```

**Result:**

| Row | city_count | state_count |
|---|---|---|
| 1 | 4119 | 27 |

OR

**Query:**

```
SELECT
COUNT(distinct geolocation_city) AS city_count,
COUNT(distinct geolocation_state) AS state_count
FROM `Target_SQL_Business_Case.geolocation`
```

**Result:**

| Row | city_count | state_count |
|---|---|---|
| 1 | 8011 | 27 |

**OR**

**Query:**

```
SELECT

 COUNT(distinct seller_city) AS city_count,

 COUNT(distinct seller_state) AS state_count

 FROM `Target_SQL_Business_Case.sellers`
```

**Result:**

| Row | city_count | state_count |
|---|---|---|
| 1 | 611 | 23 |

**Insight:**

- In geolocation table, city count is resulted with the state count.
- In customer table, city count is resulted less in comparison to geolocation table city count, which shows that customers should grow in the cities.
- In seller table, city count is very less with the less state count comparison to geolocation table city count and state count, which shows that sellers should grow fast in the cities and states.

**ADDITIONAL QUESTION:**

❖ What is the count of active No. of customers in state and city?

## 2. In-depth Exploration:

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

**Code Explanation-**

- EXTRACT Year and Month.
- Group by Year and Month to get data growing trend in the no. of orders placed over the past years.
- Table Used = Orders

**Query:**

```
SELECT

EXTRACT(year from order_purchase_timestamp) as year_data,

EXTRACT(month from order_purchase_timestamp) as month_data,

count(*) as TOTAL_NO_OF_ORDERS_PLACED

FROM `Target_SQL_Business_Case.orders`

GROUP BY year_data,month_data

ORDER BY year_data,month_data
```

**Result:**

| Row | year_data | month_data | TOTAL_NO_OF_ORDERS_PLACED |
|---|---|---|---|
| 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 no sale in the month of November 2016.
- The total no of orders trends with the Year and Month.

**ADDITIONAL QUESTION:**

- ❖ Why is the sudden change in the trend after August month?

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

**Code Explanation-**

- EXTRACT Month and use FORMAT_DATETIME .
- Count aggregate function to count No. of order placed.
- Group by order month to get monthly seasonality in terms of the no. of orders being placed.
- Table Used = Orders

**Query:**

```
SELECT

order_month,

MONTH_NAME,

count(order_id) as No_Of_Order_Placed,

FROM(

SELECT

*,

FORMAT_DATETIME('%B', order_purchase_timestamp) AS MONTH_NAME,

Extract (month from  order_purchase_timestamp) as order_month

FROM `Target_SQL_Business_Case.orders`)

Group by order_month, MONTH_NAME

ORDER BY order_month
```

**Result:**

| Row | order_month | MONTH_NAME | No_Of_Order_Placed |
|---|---|---|---|
| 1 | 1 | January | 8069 |
| 2 | 2 | February | 8508 |
| 3 | 3 | March | 9893 |
| 4 | 4 | April | 9343 |
| 5 | 5 | May | 10573 |
| 6 | 6 | June | 9412 |
| 7 | 7 | July | 10318 |
| 8 | 8 | August | 10843 |
| 9 | 9 | September | 4305 |
| 10 | 10 | October | 4959 |
| 11 | 11 | November | 7544 |
| 12 | 12 | December | 5674 |

**Insight:**

- Max no. of order placed in between May to August.
- Number of orders placed is down falling after august.

3) During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
    a. 0-6 hrs : Dawn
    b. 7-12 hrs : Mornings
    c. 13-18 hrs : Afternoon
    d. 19-23 hrs : Night

**Code Explanation-**

- CTE method is used and named as day_data.
- Case statement is used and named as Order_time and simply Group by order_time.
- Table Used = Orders
- Use time range for both neglecting 1 hr between each interval and including each missing hour.

**Query:**

```
With A AS
(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 "Morning"
WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 13 AND
18 THEN "Afternoon"
WHEN EXTRACT(hour FROM order_purchase_timestamp) BETWEEN 19 AND
23 THEN "Night"
END AS Order_Time
FROM `Target_SQL_Business_Case.orders`)
SELECT
Order_Time,
COUNT(Order_Time) AS No_of_orders_placed
FROM A
GROUP BY Order_Time
ORDER BY No_of_orders_placed DESC
LIMIT 1
```

**Result:**

| Row | Order_Time | No_of_orders_placed |
|---|---|---|
| 1 | Afternoon | 38135 |

**Insight:**

- Afternoon order timing has the maximum number of orders placed.
- There should be some offer on order to be placed during Dawn timing in comparison to the other order timing which has the more of the orders placed.

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

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

**Code Explanation-**

- Use Join to join both tables to get the customers state and order place month.
- Group by customer state
- Table Used = Orders and Customers

**Query:**

```
SELECT

customer_state,

order_month,

count(order_id) as month_order_per_state

FROM

(

SELECT

t.order_id,

c.customer_state,

FORMAT_DATETIME('%B', t.order_purchase_timestamp) order_month

FROM `Target_SQL_Business_Case.orders` as t

join `Target_SQL_Business_Case.customers` as c

on t.customer_id = c.customer_id

)
```

```
GROUP by customer_state, order_month

order by order_month
```

**Result:**

| Row | customer_state | order_month | month_order_per_state |
|---|---|---|---|
| 1 | MT | April | 92 |
| 2 | SP | April | 3967 |
| 3 | RJ | April | 1172 |
| 4 | RS | April | 488 |
| 5 | BA | April | 318 |
| 6 | CE | April | 143 |
| 7 | MG | April | 1061 |
| 8 | PE | April | 154 |
| 9 | PA | April | 107 |
| 10 | PI | April | 50 |

**Insight:**

- SP state has max orders in each month.

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

**Code Explanation-**

- Count customer unique ID.
- Table Used = customers

**Query:**

```
SELECT

customer_state,

count(distinct customer_unique_id) as count_customer

FROM `Target_SQL_Business_Case.customers`

GROUP BY customer_state
```

```
ORDER BY customer_state DESC
```

**Result**:

| Row | customer_state | count_customer |
|---|---|---|
| 1 | TO | 273 |
| 2 | SP | 40302 |
| 3 | SE | 342 |
| 4 | SC | 3534 |
| 5 | RS | 5277 |
| 6 | RR | 45 |
| 7 | RO | 240 |
| 8 | RN | 474 |
| 9 | RJ | 12384 |
| 10 | PR | 4882 |

**Insight:**

- SP state has max customers.

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.

**Code Explanation-**

- EXTRACT and Sum are used.
- Group by Year and Month to get data growing trend in the no. of orders placed over the past years.
- Table Used = Orders and Payments

**Query:**

```
SELECT CONCAT(CAST(ROUND(((COST_OF_ORDERS_2018-
COST_OF_ORDERS_2017)/COST_OF_ORDERS_2017)*100,2) AS
STRING),'%') AS OVERALL_COST_INCREASE_PERCENTAGE
FROM(
SELECT SUM(p.payment_value) AS COST_OF_ORDERS_2018
FROM `Target_SQL_Business_Case.payments` AS p
JOIN `Target_SQL_Business_Case.orders` AS o
ON o.order_id = p.order_id
WHERE EXTRACT(year FROM order_purchase_timestamp) = 2018
AND
EXTRACT(month FROM order_purchase_timestamp) BETWEEN 1 AND 8),
(SELECT SUM(p1.payment_value) AS COST_OF_ORDERS_2017
FROM `Target_SQL_Business_Case.payments` AS p1
JOIN `Target_SQL_Business_Case.orders` AS o1
ON o1.order_id = p1.order_id
WHERE EXTRACT(year FROM order_purchase_timestamp) = 2017
AND
EXTRACT(month FROM order_purchase_timestamp) BETWEEN 1 AND 8)
```

**Result:**

| Row | OVERALL_COST_INCREASE_PERCENTAGE |
|---|---|
| 1 | 136.98% |

**Insight:**

- Overall Cost Increase Percentage from year 2017 to 2018 is 136.98%.

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

**Code Explanation-**

- SUM and AVG aggregate function is usedand then Round result by 2 decimal places.
- Join customers and orders table and then join payments table
- Table Used = Orders , customers and payments.
- Group by state.

**Query:**

```
SELECT c.customer_state AS STATE,

ROUND(SUM(p.payment_value),2) AS TOTAL_ORDER_PRICE,

ROUND(AVG(p.payment_value),2) AS AVERAGE_ORDER_PRICE

FROM `Target_SQL_Business_Case.customers` AS c

JOIN `Target_SQL_Business_Case.orders` as o

ON c.customer_id = o.customer_id

JOIN `Target_SQL_Business_Case.payments` as p

ON o.order_id = p.order_id

GROUP BY STATE

ORDER BY STATE
```

**Result:**

| Row | STATE | TOTAL_ORDER_PRICE | AVERAGE_ORDER_PRICE |
|---|---|---|---|
| 1 | AC | 19680.62 | 234.29 |
| 2 | AL | 96962.06 | 227.08 |
| 3 | AM | 27966.93 | 181.6 |
| 4 | AP | 16262.8 | 232.33 |
| 5 | BA | 616645.82 | 170.82 |
| 6 | CE | 279464.03 | 199.9 |
| 7 | DF | 355141.08 | 161.13 |
| 8 | ES | 325967.55 | 154.71 |
| 9 | GO | 350092.31 | 165.76 |
| 10 | MA | 152523.02 | 198.86 |

**Insight:**

- SP state ordered highest No. of orders with average order price 137.5.
- PB state has maximum average order price with 248.33

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

**Code Explanation-**

- SUM and AVG aggregate function are used and then round result by 2 decimal places.
- Join customers and orders table and then join items table.
- Table Used = Orders, customers, and items.
- Group by state.

**Query:**

```
SELECT c.customer_state AS STATE,

ROUND(SUM(oi.freight_value),2) AS TOTAL_ORDER_FREIGHT_VALUE,

ROUND(AVG(oi.freight_value),2) AS AVERAGE_ORDER_FREIGHT_VALUE,

FROM `Target_SQL_Business_Case.customers` AS c

JOIN `Target_SQL_Business_Case.orders` as o

ON c.customer_id = o.customer_id

JOIN `Target_SQL_Business_Case.order_items` as oi

ON o.order_id = oi.order_id

GROUP BY STATE

ORDER BY STATE
```

**Result:**

| Row | STATE | TOTAL_ORDER_FREIGHT_VALUE | AVERAGE_ORDER_FREIGHT_VALUE |
|---|---|---|---|
| 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 |

**Insight:**

- max Average value of order = RR and max value of order = 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

**Code Explanation-**

- DATE DIFF function used.
- Table Used = Orders
- Filtering condition when order status is "delivered"

- **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,
order_status,
order_purchase_timestamp,
order_estimated_delivery_date,
order_delivered_customer_date,
Actual_delivery_days,
estimated_delivery_days
FROM(
SELECT
*,
DATE_DIFF(order_delivered_customer_date,
order_purchase_timestamp, day) as Actual_delivery_days,
DATE_DIFF(order_estimated_delivery_date,
order_delivered_customer_date,day) as estimated_delivery_days
FROM
`Target_SQL_Business_Case.orders`) as t1
WHERE t1.order_status = "delivered" and Actual_delivery_days is
not null
ORDER BY Actual_delivery_days, estimated_delivery_days DESC
```

**Result:**

| Row | order_id | order_status | order_purchase_timestamp | order_estimated_delivery_date | order_delivered_customer_date | Actual_delivery_days | estimated_delivery_days |
|---|---|---|---|---|---|---|---|
| 1 | 8339b608be0d84fca9d8da68b... | delivered | 2018-06-26 20:48:33 UTC | 2018-07-25 00:00:00 UTC | 2018-06-27 17:31:53 UTC | 0 | 27 |
| 2 | bb5a519e352b45b714192a02f... | delivered | 2017-05-31 11:11:55 UTC | 2017-06-27 00:00:00 UTC | 2017-06-01 08:34:36 UTC | 0 | 25 |
| 3 | 434cecee7d1a65fc65358a632... | delivered | 2017-05-29 13:21:46 UTC | 2017-06-19 00:00:00 UTC | 2017-05-30 08:06:56 UTC | 0 | 19 |
| 4 | 38c1e3d4ed6a13cd0cf612d4c... | delivered | 2018-02-02 15:26:38 UTC | 2018-02-20 00:00:00 UTC | 2018-02-03 15:05:56 UTC | 0 | 16 |
| 5 | f349cdb62f69c3fae5c4d7d3f3... | delivered | 2018-06-28 14:34:48 UTC | 2018-07-12 00:00:00 UTC | 2018-06-29 14:12:18 UTC | 0 | 12 |
| 6 | d3ca7b82c922817b06e5ca211... | delivered | 2017-11-16 13:54:08 UTC | 2017-11-29 00:00:00 UTC | 2017-11-17 13:49:40 UTC | 0 | 11 |
| 7 | f3c6775ba3d2d9fe2826f93b71... | delivered | 2017-07-04 11:37:47 UTC | 2017-07-17 00:00:00 UTC | 2017-07-05 08:09:26 UTC | 0 | 11 |
| 8 | 21a8ffca665bc7a1087d31751... | delivered | 2017-05-31 12:00:35 UTC | 2017-06-13 00:00:00 UTC | 2017-06-01 10:28:24 UTC | 0 | 11 |
| 9 | 1d893dd7ca5f77ebf5f59f0d20... | delivered | 2017-06-19 08:19:45 UTC | 2017-06-30 00:00:00 UTC | 2017-06-19 21:07:52 UTC | 0 | 10 |
| 10 | e65f1eeee1f52024ad1dcd034... | delivered | 2018-05-18 15:03:19 UTC | 2018-05-29 00:00:00 UTC | 2018-05-19 12:28:30 UTC | 0 | 9 |

**Insight:**

- By doing **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date we get the difference (in days) between the estimated & actual delivery date of an order.

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

**Code Explanation-**

- Take round of average value of freight value.
- Join=Inner join and UNION ALL to add both sub query.
- Group by state.
- Table Used = Orders, customers, and items

**Query:**

```
SELECT * FROM

(SELECT c.customer_state AS STATE,

ROUND(AVG(oi.freight_value),2) AS AVERAGE_FREIGHT_VALUE,

'TOP5 FREIGHT VALUE'AS SORTED_BY

FROM `Target_SQL_Business_Case.customers` AS c

JOIN `Target_SQL_Business_Case.orders` as o

ON c.customer_id = o.customer_id

JOIN `Target_SQL_Business_Case.order_items` as oi

ON o.order_id = oi.order_id
```

```sql
GROUP BY STATE

ORDER BY AVERAGE_FREIGHT_VALUE DESC

LIMIT 5)

UNION ALL

(SELECT c.customer_state AS STATE,

ROUND(AVG(oi.freight_value),2) AS AVERAGE_FREIGHT_VALUE,

'BOTTOM5 FREIGHT VALUE'AS SORTED_BY

FROM `Target_SQL_Business_Case.customers` AS c

JOIN `Target_SQL_Business_Case.orders` as o

ON c.customer_id = o.customer_id

JOIN `Target_SQL_Business_Case.order_items` as oi

ON o.order_id = oi.order_id

GROUP BY STATE

ORDER BY AVERAGE_FREIGHT_VALUE ASC

LIMIT 5)

ORDER BY AVERAGE_FREIGHT_VALUE DESC
```

**Result:**

| Row | STATE | AVERAGE_FREIGHT | SORTED_BY |
|---|---|---|---|
| 1 | RR | 42.98 | TOP5 FREIGHT VALUE |
| 2 | PB | 42.72 | TOP5 FREIGHT VALUE |
| 3 | RO | 41.07 | TOP5 FREIGHT VALUE |
| 4 | AC | 40.07 | TOP5 FREIGHT VALUE |
| 5 | PI | 39.15 | TOP5 FREIGHT VALUE |
| 6 | DF | 21.04 | BOTTOM5 FREIGHT VALUE |
| 7 | RJ | 20.96 | BOTTOM5 FREIGHT VALUE |
| 8 | MG | 20.63 | BOTTOM5 FREIGHT VALUE |
| 9 | PR | 20.53 | BOTTOM5 FREIGHT VALUE |
| 10 | SP | 15.15 | BOTTOM5 FREIGHT VALUE |

**Insight:**

- The Top 5 and Bottom 5 states is separated by the column SORTED_BY and the highest & lowest average freight value is given in the column Average Freight.

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

**Code Explanation-**

- Take round of average value of Date difference to get actual delivery time.
- Join=Inner join and UNION ALL to add both sub query.
- Group by customer state.
- Table Used = Orders and customers.

**Query:**

```
(SELECT

c.customer_state,

ROUND(AVG(date_diff(order_delivered_customer_date,order_purchas

e_timestamp,day)),2) as actual_delivery_time,

"Top 5 State on Deliver Time " as SORTED_BY
```

```sql
FROM `Target_SQL_Business_Case.orders` as o

JOIN `Target_SQL_Business_Case.customers` as c

ON o.customer_id = c.customer_id

GROUP BY c.customer_state

ORDER BY actual_delivery_time DESC

LIMIT 5)

UNION ALL

(SELECT

c.customer_state,

ROUND(AVG(date_diff(order_delivered_customer_date,order_purchas

e_timestamp,day)),2) as actual_delivery_time,

"bottom 5 State on Deliver Time " as SORTED_BY

FROM `Target_SQL_Business_Case.orders` as o

JOIN `Target_SQL_Business_Case.customers` as c

ON o.customer_id = c.customer_id

GROUP BY c.customer_state

ORDER BY actual_delivery_time ASC

LIMIT 5)
```

**Result**:

| Row | customer_state | actual_delivery_time | SORTED_BY |
|---|---|---|---|
| 1 | RR | 28.98 | Top 5 State on Deliver Time |
| 2 | AP | 26.73 | Top 5 State on Deliver Time |
| 3 | AM | 25.99 | Top 5 State on Deliver Time |
| 4 | AL | 24.04 | Top 5 State on Deliver Time |
| 5 | PA | 23.32 | Top 5 State on Deliver Time |
| 6 | SP | 8.3 | bottom 5 State on Deliver Time |
| 7 | PR | 11.53 | bottom 5 State on Deliver Time |
| 8 | MG | 11.54 | bottom 5 State on Deliver Time |
| 9 | DF | 12.51 | bottom 5 State on Deliver Time |
| 10 | SC | 14.48 | bottom 5 State on Deliver Time |

**Insight:**

- The top 5 states with the highest & lowest average delivery time is given in the column customer state and actual delivery time which is separated by column SORTED_BY as it shows SP state has fastest delivery time and RR state has slowest delivery time.

4) Find out the top 5 states where the order delivery is 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.

**Code Explanation-**

- Round and AVG of date difference.
- Group by customer state
- Table Used = Orders and customer

- **time_to_deliver** = order_delivered_customer_date - order_purchase_timestamp
- **diff_estimated_delivery** = order_estimated_delivery_date - order_delivered_customer_date

**Query:**

```
SELECT
t1.customer_state,
ROUND((t1.Actual_delivery_days- t1.estimated_delivery_days),2)
as fastest_delivery
FROM
(SELECT
c.customer_state,
AVG(date_diff(order_delivered_customer_date,
order_purchase_timestamp, day)) as Actual_delivery_days,
AVG(date_diff(order_estimated_delivery_date,
order_delivered_customer_date,day)) as estimated_delivery_days
FROM `Target_SQL_Business_Case.orders` as o
JOIN `Target_SQL_Business_Case.customers` as c
ON c.customer_id = o.customer_id
WHERE o.order_status = "delivered"
GROUP BY c.customer_state
) as t1
WHERE ROUND((t1.Actual_delivery_days-
t1.estimated_delivery_days),2) >=0
ORDER BY fastest_delivery ASC
limit 5
```

**Result:**

| Row | customer_state | fastest_delivery |
|---|---|---|
| 1 | AC | 0.87 |
| 2 | DF | 1.39 |
| 3 | RS | 1.84 |
| 4 | SC | 3.87 |
| 5 | GO | 3.88 |

**Insight:**

- fastest order delivery of state AC that is 0.87.

## 6. Analysis based on the payments:

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

**Code Explanation-**

- Use CTE
- EXTRACT Year and Month.
- Join orders and payments table.
- Group by order Year, payment type and order month.
- Table Used = Orders and payments

**Query:**

```
WITH main_table as (

SELECT

o.order_id,

p.payment_type,

EXTRACT(YEAR FROM order_purchase_timestamp ) as order_year,

EXTRACT(MONTH FROM order_purchase_timestamp ) as order_month
```

```
FROM `Target_SQL_Business_Case.orders` as o
join`Target_SQL_Business_Case.payments` as p on o.order_id =
p.order_id)
SELECT
order_year,
order_month,
payment_type,
count (order_id) as  order_count
FROM main_table
GROUP BY
order_year,
order_month,
payment_type
Order BY
order_year,
order_month,
```

**Result:**

| Row | order_year | order_month | payment_type | order_count |
|---|---|---|---|---|
| 1 | 2016 | 9 | credit_card | 3 |
| 2 | 2016 | 10 | credit_card | 254 |
| 3 | 2016 | 10 | voucher | 23 |
| 4 | 2016 | 10 | debit_card | 2 |
| 5 | 2016 | 10 | UPI | 63 |
| 6 | 2016 | 12 | credit_card | 1 |
| 7 | 2017 | 1 | voucher | 61 |
| 8 | 2017 | 1 | UPI | 197 |
| 9 | 2017 | 1 | credit_card | 583 |
| 10 | 2017 | 1 | debit_card | 9 |

**Insight:**

- Maximum order purchased by credit card and min order purchased by debit card in each month.

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

**Code Explanation-**

- Group by payment_sequential,payment_installments
- Table Used = payments

**Query:**

```
SELECT

payment_sequential,

payment_installments,

count(order_id) as No_of_order_Placed

from `Target_SQL_Business_Case.payments`

WHERE payment_installments >= payment_sequential

GROUP BY payment_sequential,payment_installments

ORDER BY payment_sequential
```

**Result:**

| Row | payment_sequential | payment_installment | No_of_order_Placed |
|---|---|---|---|
| 1 | 1 | 1 | 48236 |
| 2 | 1 | 2 | 12360 |
| 3 | 1 | 3 | 10422 |
| 4 | 1 | 4 | 7066 |
| 5 | 1 | 5 | 5221 |
| 6 | 1 | 6 | 3904 |
| 7 | 1 | 7 | 1619 |
| 8 | 1 | 8 | 4242 |
| 9 | 1 | 9 | 644 |
| 10 | 1 | 10 | 5305 |

**Insight:**

- Maximum number of orders has been placed after paying the first installment.