# STRUCTURE & CHARACTERISTICS OF THE DATASET

1. Get number of rows in the data

- Customers

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.customers`
```

| Row | No_of_rows |
|---|---|
| 1 | 99441 |

- Geolocation

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.geolocation`
```

| Row | No_of_rows |
|---|---|
| 1 | 1000163 |

- Order Items

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.order_items`
```

| Row | No_of_rows |
|---|---|
| 1 | 112650 |

- Order reviews

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.order_reviews`
```

| Row | No_of_rows |
|---|---|
| 1 | 99224 |

- Orders

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.orders`
```

| Row | No_of_rows |
|---|---|
| 1 | 99441 |

- Payments

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.payments`
```

| Row | No_of_rows |
|---|---|
| 1 | 103886 |

- Products

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.products`
```

| Row | No_of_rows |
|---|---|
| 1 | 32951 |

- Sellers

```
SELECT COUNT(*) No_of_rows
FROM `sql-scaler.TargetSQL.sellers`
```

| Row | No_of_rows |
|---|---|
| 1 | 3095 |

2. Number of null or missing values in a column

- Customers

```
SELECT
  SUM(CASE WHEN customer_id IS NULL THEN 1 ELSE 0 END) customer_id,
  SUM(CASE WHEN customer_unique_id IS NULL THEN 1 ELSE 0 END)customer_unique_id,
  SUM(CASE WHEN customer_zip_code_prefix IS NULL THEN 1 ELSE 0 END)customer_zip_code_prefix,
  SUM(CASE WHEN customer_city IS NULL THEN 1 ELSE 0 END)customer_city,
  SUM(CASE WHEN customer_state IS NULL THEN 1 ELSE 0 END)customer_state
FROM `sql-scaler.TargetSQL.customers`
```

```
NOTE: USE count(1) from table where column is null instaed ( faster execution) when doing for single column
```

| Row | customer_id | customer_unique_id | customer_zip_code_prefix | customer_city | customer_state |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |

- Geolocation

```
SELECT
  sum(case when geolocation_zip_code_prefix is null then 1 else 0 end) geolocation_zip_code_pr
efix,
  sum(case when geolocation_lat is null then 1 else 0 end) geolocation_lat,
  sum(case when geolocation_lng is null then 1 else 0 end) geolocation_lng,
  sum(case when geolocation_city is null then 1 else 0 end) geolocation_city,
  sum(case when geolocation_state is null then 1 else 0 end) geolocation_state
FROM `sql-scaler.TargetSQL.geolocation`
```

| Row | geolocation_zip_code_prefix | geolocation_lat | geolocation_lng | geolocation_city | geolocation_state |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |

- Order Items

```
SELECT
  sum(case when order_id is null then 1 else 0 end) order_id,
  sum(case when order_item_id is null then 1 else 0 end) order_item_id,
  sum(case when product_id is null then 1 else 0 end) product_id,
  sum(case when seller_id is null then 1 else 0 end) seller_id,
  sum(case when shipping_limit_date is null then 1 else 0 end) shipping_limit_date,
  sum(case when price is null then 1 else 0 end) price,
  sum(case when freight_value is null then 1 else 0 end) freight_value
FROM `sql-scaler.TargetSQL.order_items`
```

| Row | order_id | order_item_id | product_id | seller_id | shipping_limit_date | price | freight_value |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Order reviews

```
SELECT
  sum(case when review_id is null then 1 else 0 end) review_id,
  sum(case when order_id is null then 1 else 0 end) order_id,
  sum(case when review_score is null then 1 else 0 end) review_score,
  sum(case when review_comment_title is null then 1 else 0 end) review_comment_title,
  sum(case when review_creation_date is null then 1 else 0 end) review_creation_date,
  sum(case when review_answer_timestamp is null then 1 else 0 end) review_answer_timestamp
FROM `sql-scaler.TargetSQL.order_reviews`
```

| Row | review_id | order_id | review_score | review_comment_title | review_creation_date | review_answer_timestamp |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 87675 | 0 | 0 |

- Orders

```
SELECT
  sum(case when customer_id is null then 1 else 0 end) customer_id,
  sum(case when order_id is null then 1 else 0 end) order_id,
  sum(case when order_status is null then 1 else 0 end) order_status,
  sum(case when order_purchase_timestamp is null then 1 else 0 end) order_purchase_timestamp,
  sum(case when order_approved_at is null then 1 else 0 end) order_approved_at,
  sum(case when order_delivered_carrier_date is null then 1 else 0 end) order_delivered_carrie
r_date,
  sum(case when order_delivered_customer_date is null then 1 else 0 end) order_delivered_custo
mer_date,
  sum(case when order_estimated_delivery_date is null then 1 else 0 end) order_estimated_deliv
ery_date
FROM `sql-scaler.TargetSQL.orders`
```

| customer_id | order_id | order_status | order_purchase_timestamp | order_approved_at | order_delivered_carrier_date | order_delivered_customer_date | order_estimated_delivery_date |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 160 | 1783 | 2965 | 0 |

- Payments

```
SELECT
  sum(case when order_id is null then 1 else 0 end) order_id,
  sum(case when payment_sequential is null then 1 else 0 end) payment_sequential,
  sum(case when payment_type is null then 1 else 0 end) payment_type,
  sum(case when payment_installments is null then 1 else 0 end) payment_installments,
  sum(case when payment_value is null then 1 else 0 end) payment_value
FROM `sql-scaler.TargetSQL.payments`
```

| Row | order_id | payment_sequential | payment_type | payment_installments | payment_value |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |

- Products

```
SELECT
  sum(case when product_id is null then 1 else 0 end) product_id,
  sum(case when product_category is null then 1 else 0 end) product_category,
  sum(case when product_name_length is null then 1 else 0 end) product_name_length,
  sum(case when product_description_length is null then 1 else 0 end) product_description_leng
th,
```

```
  sum(case when product_photos_qty is null then 1 else 0 end) product_photos_qty,
  sum(case when product_weight_g is null then 1 else 0 end) product_weight_g,
  sum(case when product_length_cm is null then 1 else 0 end) product_length_cm,
  sum(case when product_height_cm is null then 1 else 0 end) product_height_cm,
  sum(case when product_width_cm is null then 1 else 0 end) product_width_cm,
FROM `sql-scaler.TargetSQL.products`
```

| product_id | product_category | product_name_length | product_description_length | product_photos_qty | product_weight_g | product_length_cm | product_height_cm | product_width_cm |
|---|---|---|---|---|---|---|---|---|
| 0 | 610 | 610 | 610 | 610 | 2 | 2 | 2 | 2 |

- Sellers

```
SELECT
  sum(case when seller_id is null then 1 else 0 end) seller_id,
  sum(case when seller_zip_code_prefix is null then 1 else 0 end) seller_zip_code_prefix,
  sum(case when seller_city is null then 1 else 0 end) seller_city,
  sum(case when seller_state is null then 1 else 0 end) seller_state,
FROM `sql-scaler.TargetSQL.sellers`
```

| Row | seller_id | seller_zip_code_prefix | seller_city | seller_state |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |

3. Data type of columns in a table

- Customers

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where
table_name = 'customers'
```

| 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 |

- Geolocation

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where
table_name = 'geolocation'
```

| Row | column_name | DATA_TYPE |
|---|---|---|
| 1 | geolocation_zip_code_prefix | INT64 |
| 2 | geolocation_lat | FLOAT64 |
| 3 | geolocation_lng | FLOAT64 |
| 4 | geolocation_city | STRING |
| 5 | geolocation_state | STRING |

- Order Items

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where
table_name = 'order_items'
```

| Row | column_name | DATA_TYPE |
|---|---|---|
| 1 | order_id | STRING |
| 2 | order_item_id | INT64 |
| 3 | product_id | STRING |
| 4 | seller_id | STRING |
| 5 | shipping_limit_date | TIMESTAMP |
| 6 | price | FLOAT64 |
| 7 | freight_value | FLOAT64 |

- Order reviews

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where table_name = 'order_reviews'
```

| Row | column_name | DATA_TYPE |
|---|---|---|
| 1 | review_id | STRING |
| 2 | order_id | STRING |
| 3 | review_score | INT64 |
| 4 | review_comment_title | STRING |
| 5 | review_creation_date | TIMESTAMP |
| 6 | review_answer_timestamp | TIMESTAMP |

- Orders

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where table_name = 'orders'
```

| Row | column_name | DATA_TYPE |
|---|---|---|
| 1 | order_id | STRING |
| 2 | customer_id | STRING |
| 3 | order_status | STRING |
| 4 | order_purchase_timestamp | TIMESTAMP |
| 5 | order_approved_at | TIMESTAMP |
| 6 | order_delivered_carrier_date | TIMESTAMP |
| 7 | order_delivered_customer_date | TIMESTAMP |
| 8 | order_estimated_delivery_date | TIMESTAMP |

- Payments

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where table_name = 'payments'
```

| Row | column_name | DATA_TYPE |
|---|---|---|
| 1 | order_id | STRING |
| 2 | payment_sequential | INT64 |
| 3 | payment_type | STRING |
| 4 | payment_installments | INT64 |
| 5 | payment_value | FLOAT64 |

- Products

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where
table_name = 'products'
```

| Row | column_name | DATA_TYPE |
|---|---|---|
| 1 | product_id | STRING |
| 2 | product_category | STRING |
| 3 | product_name_length | INT64 |
| 4 | product_description_length | INT64 |
| 5 | product_photos_qty | INT64 |
| 6 | product_weight_g | INT64 |
| 7 | product_length_cm | INT64 |
| 8 | product_height_cm | INT64 |
| 9 | product_width_cm | INT64 |

- Sellers

```
SELECT column_name , DATA_TYPE from TargetSQL.INFORMATION_SCHEMA.COLUMNS where
table_name = 'sellers'
```

| Row | column_name | DATA_TYPE |
|---|---|---|
| 1 | seller_id | STRING |
| 2 | seller_zip_code_prefix | INT64 |
| 3 | seller_city | STRING |
| 4 | seller_state | STRING |

4. Time period for which the data is given

```
SELECT
  min(DATE(order_purchase_timestamp)) as first_purchase,
  max(DATE(order_purchase_timestamp)) as last_purchase
FROM `sql-scaler.TargetSQL.orders`
```

| Row | first_purchase | last_purchase |
|---|---|---|
| 1 | 2016-09-04 | 2018-10-17 |

5. Number of cities in our dataset

```
SELECT
  COUNT(DISTINCT customer_city) No_of_cities
FROM `sql-scaler.TargetSQL.customers`
```

| Row | No_of_cities |
|---|---|
| 1 | 4119 |

6. Number of states in our dataset

```
SELECT
  COUNT(DISTINCT customer_state) No_of_states
FROM `sql-scaler.TargetSQL.customers`
```

| Row | No_of_states |
|---|---|
| 1 | 27 |

## IN-DEPTH EXPLORATION:

1. **How many orders do we have for each order status?**

```
SELECT
  order_status,
  COUNT(order_id) No_of_orders
FROM `sql-scaler.TargetSQL.orders`
GROUP BY order_status
```

| Row | order_status | No_of_orders |
|---|---|---|
| 1 | created | 5 |
| 2 | shipped | 1107 |
| 3 | approved | 2 |
| 4 | canceled | 625 |
| 5 | invoiced | 314 |
| 6 | delivered | 96478 |
| 7 | processing | 301 |
| 8 | unavailable | 609 |

2. **Is there a growing trend on e-commerce in Brazil? How can we describe a complete scenario?**

```
SELECT
  EXTRACT(YEAR from DATE(order_purchase_timestamp)) Year,

  COUNT(order_id) orders
FROM `sql-scaler.TargetSQL.orders`
GROUP BY 1
```

```
ORDER BY 1
```

| Row | Year | orders |
|---|---|---|
| 1 | 2016 | 329 |
| 2 | 2017 | 45101 |
| 3 | 2018 | 54011 |

```
SELECT
  EXTRACT(YEAR from DATE(order_purchase_timestamp)) Year,
  EXTRACT(MONTH from DATE(order_purchase_timestamp)) Month,
  COUNT(order_id) orders
FROM `sql-scaler.TargetSQL.orders`
GROUP BY 1,2
ORDER BY 1,2
```

| Year | Month | orders |
|---|---|---|
| 2017 | 1 | 800 |
| 2017 | 2 | 1780 |
| 2017 | 3 | 2682 |
| 2017 | 4 | 2404 |
| 2017 | 5 | 3700 |
| 2017 | 6 | 3245 |
| 2017 | 7 | 4026 |
| 2017 | 8 | 4331 |
| 2017 | 9 | 4285 |
| 2017 | 10 | 4631 |

```
SELECT
  EXTRACT(YEAR from DATE(order_purchase_timestamp)) Year,
  product_category,
  count(product_category) Most_bought
FROM `sql-scaler.TargetSQL.orders` o
JOIN `sql-scaler.TargetSQL.order_items` ot
  ON o.order_id = ot.order_id
JOIN `sql-scaler.TargetSQL.products` p
  ON ot.product_id = p.product_id
GROUP BY 1,2
ORDER BY 1, 3 DESC
```

| Year | product_category | Most_bought | Year | product_category | Most_bought |
|---|---|---|---|---|---|
| 2016 | Furniture Decoration | 69 | 2017 | bed table bath | 5223 |
| 2016 | HEALTH BEAUTY | 51 | 2017 | Furniture Decoration | 4147 |
| 2016 | perfumery | 33 | 2017 | sport leisure | 4095 |

| Year | product_category | Most_bought |
|---|---|---|
| 2018 | HEALTH BEAUTY | 5951 |
| 2018 | bed table bath | 5884 |
| 2018 | computer accessories | 4708 |

We can observe a substantial increase in the number of order per year and also per month of every year except for June and December for 2016.

Number of orders per month in 2018 are less than the previous months of 2017 but total orders are greater.

Therefore we can conclude that there is a growing trend on e-commerce.

When observing top 3 most bought product_categories every year we see that there is a trend of buying from Furniture Decoations, Health Beauty and bed table bath categories as compared to other categories.

3. **On what day of week brazilians customers tend to do online purchasing?**

```
SELECT
  EXTRACT(DAYOFWEEK from DATE(order_purchase_timestamp)) Day_of_week,
  COUNT(order_id) orders
FROM `sql-scaler.TargetSQL.orders`
GROUP BY 1
ORDER BY 2 DESC
```

| Row | Day_of_week | orders |
|---|---|---|
| 1 | 2 | 16196 |
| 2 | 3 | 15963 |
| 3 | 4 | 15552 |
| 4 | 5 | 14761 |
| 5 | 6 | 14122 |
| 6 | 1 | 11960 |
| 7 | 7 | 10887 |

The Brazilian customers tend to do most online shopping on Monday (1), Tuesday (2) and Wednesday (3) and least on Saturday (7).

The sales on weekends, Saturday and Sundays are the least.

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

```
SELECT
```

```
  CASE
    WHEN TIME(order_purchase_timestamp) BETWEEN "00:00:00" AND "06:00:00"
    THEN "DAWN"
    WHEN TIME(order_purchase_timestamp) BETWEEN "06:00:00" AND "12:00:00"
    THEN "MORNING"
    WHEN TIME(order_purchase_timestamp) BETWEEN "12:00:00" AND "18:00:00"
    THEN "AFTERNOON"
    ELSE "NIGHT"
  END AS Time_of_day,
  COUNT(order_id) orders
FROM `sql-scaler.TargetSQL.orders`
GROUP BY 1
ORDER BY 2 DESC
```

| Row | Time_of_day | orders |
|---|---|---|
| 1 | AFTERNOON | 38365 |
| 2 | NIGHT | 34096 |
| 3 | MORNING | 22240 |
| 4 | DAWN | 4740 |

Customers tend to buy most during Afternoon followed by Night. They buy least during the Dawn, which is expected.

5. **Feature Extraction: Through order_purchase_timestamp in “orders” dataset extract**
    1. **order_purchase_year**
    2. **order_purchase_month**
    3. **order_purchase_date**
    4. **order_purchase_day**
    5. **order_purchase_dayofweek**
    6. **order_purchase_dayofweek_name**
    7. **order_purchase_hour**
    8. **order_purchase_time_day**

```
SELECT
  EXTRACT(YEAR from order_purchase_timestamp) year,
  EXTRACT(MONTH from order_purchase_timestamp) month,
  EXTRACT(DATE from order_purchase_timestamp) date,
  EXTRACT(DAY from order_purchase_timestamp) day,
  EXTRACT(DAYOFWEEK from order_purchase_timestamp) dayofweek,
  FORMAT_DATE("%A", order_purchase_timestamp) dayofweek_name,
  EXTRACT(HOUR from order_purchase_timestamp) hour,
  EXTRACT(TIME from order_purchase_timestamp) time_day
FROM `sql-scaler.TargetSQL.orders`
LIMIT 10
```

| Row | year | month | date | day | dayofweek | dayofweek_name | hour | time_day |
|---|---|---|---|---|---|---|---|---|
| 1 | 2017 | 11 | 2017-11-25 | 25 | 7 | Saturday | 11 | 11:10:33 |
| 2 | 2017 | 12 | 2017-12-05 | 5 | 3 | Tuesday | 1 | 01:07:58 |
| 3 | 2017 | 12 | 2017-12-05 | 5 | 3 | Tuesday | 1 | 01:07:52 |
| 4 | 2018 | 2 | 2018-02-09 | 9 | 6 | Friday | 17 | 17:21:04 |
| 5 | 2017 | 11 | 2017-11-06 | 6 | 2 | Monday | 13 | 13:12:34 |
| 6 | 2017 | 4 | 2017-04-20 | 20 | 5 | Thursday | 12 | 12:45:34 |
| 7 | 2017 | 7 | 2017-07-13 | 13 | 5 | Thursday | 11 | 11:03:05 |
| 8 | 2017 | 7 | 2017-07-11 | 11 | 3 | Tuesday | 13 | 13:36:30 |
| 9 | 2017 | 7 | 2017-07-29 | 29 | 7 | Saturday | 18 | 18:05:07 |
| 10 | 2017 | 7 | 2017-07-13 | 13 | 5 | Thursday | 10 | 10:02:47 |

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

### 1. Get month on month orders by region

```
SELECT
  customer_city,
  EXTRACT(YEAR FROM order_purchase_timestamp) Year,
  EXTRACT(MONTH FROM order_purchase_timestamp) Month,
  COUNT(o.order_id) Total_orders
FROM `sql-scaler.TargetSQL.customers` c
JOIN `sql-scaler.TargetSQL.orders` o
  ON c.customer_id = o.customer_id
GROUP BY 1,2,3
ORDER BY 2,3, 4 DESC
```

| Row | customer_state | Year | Month | Total_orders |
|---|---|---|---|---|
| 1 | SP | 2016 | 9 | 2 |
| 2 | RR | 2016 | 9 | 1 |
| 3 | RS | 2016 | 9 | 1 |
| 4 | SP | 2016 | 10 | 113 |
| 5 | RJ | 2016 | 10 | 56 |
| 6 | MG | 2016 | 10 | 40 |
| 7 | RS | 2016 | 10 | 24 |
| 8 | PR | 2016 | 10 | 19 |
| 9 | SC | 2016 | 10 | 11 |
| 10 | GO | 2016 | 10 | 9 |

### 2. Total of customer orders by state

```
SELECT
  customer_state,
  COUNT(o.order_id) Total_orders
FROM `sql-scaler.TargetSQL.customers` c
```

```
JOIN `sql-scaler.TargetSQL.orders` o
  ON c.customer_id = o.customer_id
GROUP BY 1
ORDER BY 2 DESC
```

| Row | customer_state | Total_orders |
|---|---|---|
| 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 |

## 3. Top 10 brazilian cities most no. of orders

```
SELECT
  customer_city,
  COUNT(o.order_id) Total_orders
FROM `sql-scaler.TargetSQL.customers` c
JOIN `sql-scaler.TargetSQL.orders` o
  ON c.customer_id = o.customer_id
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
```

| Row | customer_city | Total_orders |
|---|---|---|
| 1 | sao paulo | 15540 |
| 2 | rio de janeiro | 6882 |
| 3 | belo horizonte | 2773 |
| 4 | brasilia | 2131 |
| 5 | curitiba | 1521 |
| 6 | campinas | 1444 |
| 7 | porto alegre | 1379 |
| 8 | salvador | 1245 |
| 9 | guarulhos | 1189 |
| 10 | sao bernardo do campo | 938 |

## 4. How are customers distributed in Brazil

```
SELECT
  customer_state,
  COUNT(DISTINCT customer_unique_id) Customers     #edited here
FROM `sql-scaler.TargetSQL.customers` c
```

```
GROUP BY 1
ORDER BY 2 DESC
```

| Row | customer_state | Customers |
|---|---|---|
| 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 |

We observe a very heavy majority (41746) customers stay in state with code SP.

Next state with most customers are from RJ followed by MG

5. **City wise number of unique customers**

```
SELECT
  customer_city,
  COUNT(DISTINCT customer_unique_id) Customers
FROM `sql-scaler.TargetSQL.customers` c
GROUP BY 1
ORDER BY 2 DESC
```

| Row | customer_city | Customers |
|---|---|---|
| 1 | sao paulo | 15540 |
| 2 | rio de janeiro | 6882 |
| 3 | belo horizonte | 2773 |
| 4 | brasilia | 2131 |
| 5 | curitiba | 1521 |
| 6 | campinas | 1444 |
| 7 | porto alegre | 1379 |
| 8 | salvador | 1245 |
| 9 | guarulhos | 1189 |
| 10 | sao bernardo do campo | 938 |

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

Step 1: Using CTE

1. “order_items” + “order” joined on order id where order_purchase timestamp is already divided into month & year
2. Group data by year and month, aggregation count(order_id), sum(price), sum(freight_value)
3. Create new columns:

    1. price_per_order = sum(price) / count(order_id)
    2. freight_per_order= sum(freight_value) / count(order_id)

```
WITH orderdata AS(
  SELECT
    EXTRACT(YEAR FROM order_purchase_timestamp) Year,
    EXTRACT(MONTH FROM order_purchase_timestamp) Month,
    count(o.order_id) Total_orders,
    sum(price) Total_price,
    sum(freight_value) Total_freight_value,
    sum(price) / count(o.order_id) price_per_ordersum,
    sum(freight_value) / count(o.order_id) freight_per_order
  FROM `sql-scaler.TargetSQL.order_items` ot
  JOIN `sql-scaler.TargetSQL.orders` o
    ON ot.order_id = o.order_id
  GROUP BY 1,2
  ORDER BY 1,2 , 3 DESC, 4 DESC, 5 DESC

)
```

Step 2:

1. **Total amount sold in 2017 between Jan to August**

```
SELECT
  SUM(Total_price) AS JAN_AUG_2017
FROM orderdata
WHERE orderdata.Year = 2017 AND (orderdata.Month BETWEEN 1 AND 8)
```

| Row | JAN_AUG_2017 |
|---|---|
| 1 | 3113000.3199994233 |

2. **Total amount sold in 2018 between Jan to august**

```
SELECT
  SUM(Total_price) AS JAN_AUG_2018
FROM orderdata
WHERE orderdata.Year = 2018 AND (orderdata.Month BETWEEN 1 AND 8)
```

| Row | JAN_AUG_2018 |
|---|---|
| 1 | 7385905.8000043072 |

3. **% increase from 2017 to 2018**

```sql
SELECT
  x.Year,
  x.Total,
  (x.Total - x.Prev_year_total) * 100 / x.Total Percent_Inc
FROM
  (SELECT
    Year,
    sum(Total_price) Total,
    LAG(sum(Total_price), 1) OVER (order by Year) as Prev_year_total
  FROM orderdata
  WHERE Year IN (2017 , 2018)
  GROUP BY 1
  ORDER BY 1 DESC) as x
```

| Row | Year | Total | Percent_Inc |
|---|---|---|---|
| 1 | 2018 | 7386050.8000076534 | 16.656314088733904 |
| 2 | 2017 | 6155806.9800049355 | *null* |

Step 3: Join (orders+order_items) table from previous step with “customers” table on Customer_id and find:

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

```sql
SELECT
  c.customer_state,
  sum(price) Total_price,
  sum(freight_value) Total_freight_value,
  AVG(price) mean_price,
  AVG(freight_value) mean_freight
FROM `sql-scaler.TargetSQL.order_items` ot
JOIN `sql-scaler.TargetSQL.orders` o
  ON ot.order_id = o.order_id
JOIN `sql-scaler.TargetSQL.customers` c
  ON o.customer_id = c.customer_id
GROUP BY 1
ORDER BY 1
```

| Row | customer_state | Total_price | Total_freight_value | mean_price | mean_freight |
|---|---|---|---|---|---|
| 1 | AC | 15982.949999999984 | 3686.7500000000014 | 173.72771739130431 | 40.073369565217362 |
| 2 | AL | 80314.809999999576 | 15914.589999999989 | 180.88921171171162 | 35.843671171171167 |
| 3 | AM | 22356.840000000029 | 5478.8900000000012 | 135.49599999999998 | 33.205393939393922 |
| 4 | AP | 13474.299999999988 | 2788.5000000000009 | 164.32073170731709 | 34.006097560975626 |
| 5 | BA | 511349.99000002112 | 100156.67999999922 | 134.60120821268725 | 26.36395893656228 |
| 6 | CE | 227254.70999999647 | 48351.589999999982 | 153.75826116373477 | 32.714201623816017 |
| 7 | DF | 302603.93999999622 | 50625.499999999418 | 125.77054862842866 | 21.041354945968422 |
| 8 | ES | 275037.30999999505 | 49764.599999999722 | 121.91370124113459 | 22.058776595744732 |
| 9 | GO | 294591.94999999512 | 53114.979999999705 | 126.27173167595375 | 22.766815259322772 |
| 10 | MA | 119648.21999999964 | 31523.77000000004 | 145.20415048543708 | 38.257002427184474 |

# Analysis on sales, freight and delivery time

## 1. Calculating days between purchasing, delivering and estimated delivery

```
SELECT
  DATE(order_purchase_timestamp) Purchase_date,
  DATE(order_delivered_customer_date) Delivery_date,
  DATE(order_estimated_delivery_date) Est_del_date,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY) days_btw_deli_purchase,
  DATE_DIFF(order_estimated_delivery_date, order_purchase_timestamp,DAY) days_btw_estdeli_purchase,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date, DAY) days_btw_deli_estdeli
FROM `sql-scaler.TargetSQL.orders`
WHERE order_delivered_customer_date is NOT NULL
```

| Row | Purchase_date | Delivery_date | Est_del_date | days_btw_deli_purchase | days_btw_estdeli_purchase | days_btw_deli_estdeli |
|---|---|---|---|---|---|---|
| 1 | 2016-10-07 | 2016-10-14 | 2016-11-29 | 7 | 52 | 45 |
| 2 | 2016-10-09 | 2016-11-09 | 2016-12-08 | 30 | 59 | 28 |
| 3 | 2016-10-09 | 2016-10-16 | 2016-11-30 | 7 | 51 | 44 |
| 4 | 2016-10-08 | 2016-10-19 | 2016-11-30 | 10 | 52 | 41 |
| 5 | 2016-10-03 | 2016-11-08 | 2016-11-25 | 35 | 52 | 16 |
| 6 | 2017-03-17 | 2017-04-07 | 2017-05-18 | 20 | 61 | 40 |
| 7 | 2017-03-20 | 2017-03-30 | 2017-05-18 | 10 | 58 | 48 |
| 8 | 2017-03-21 | 2017-04-18 | 2017-05-18 | 28 | 57 | 29 |
| 9 | 2018-08-20 | 2018-08-29 | 2018-10-04 | 9 | 44 | 35 |
| 10 | 2018-08-12 | 2018-08-23 | 2018-10-04 | 10 | 52 | 41 |

## 2. Create columns:

a. **time_to_delivery = order_purchase_timestamp-order_delivered_customer_date**

b. **diff_estimated_delivery = order_estimated_delivery_date-order_delivered_customer_date**

```
SELECT
  DATE(order_purchase_timestamp) Purchase_date,
  DATE(order_delivered_customer_date) Delivery_date,
  DATE(order_estimated_delivery_date) Est_del_date,
  DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY) time_to_delivery,
  DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY) diff_estimated_delivery
FROM `sql-scaler.TargetSQL.orders`
WHERE order_delivered_customer_date is NOT NULL
```

| Row | Purchase_date | Delivery_date | Est_del_date | time_to_delivery | diff_estimated_delivery |
|---|---|---|---|---|---|
| 1 | 2016-10-07 | 2016-10-14 | 2016-11-29 | 7 | 45 |
| 2 | 2018-02-19 | 2018-03-21 | 2018-03-09 | 30 | -12 |
| 3 | 2016-10-09 | 2016-10-16 | 2016-11-30 | 7 | 44 |
| 4 | 2016-10-08 | 2016-10-19 | 2016-11-30 | 10 | 41 |
| 5 | 2017-05-10 | 2017-05-23 | 2017-05-18 | 12 | -5 |
| 6 | 2017-04-08 | 2017-05-22 | 2017-05-18 | 43 | -4 |
| 7 | 2017-04-11 | 2017-04-18 | 2017-05-18 | 6 | 29 |
| 8 | 2017-03-17 | 2017-04-07 | 2017-05-18 | 20 | 40 |
| 9 | 2017-04-11 | 2017-05-22 | 2017-05-18 | 40 | -4 |
| 10 | 2017-03-20 | 2017-03-30 | 2017-05-18 | 10 | 48 |

2. **Grouping data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery**
3. **Sort the data to get the following:**
    a. **Top 5 states with highest/lowest average freight value**

### HIGHEST AVERAGE

```
SELECT
  c.customer_state,
  AVG(freight_value) mean_freight,
  AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY)) mean_time_to_delivery,
  AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY)) mean_diff_estimate
d_delivery
FROM `sql-scaler.TargetSQL.order_items` ot
JOIN `sql-scaler.TargetSQL.orders` o
  ON ot.order_id = o.order_id
JOIN `sql-scaler.TargetSQL.customers` c
  ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date is NOT NULL
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5
```

| Row | customer_state | mean_freight | mean_time_to_delivery | mean_diff_estimated_delivery |
|---|---|---|---|---|
| 1 | PB | 43.091689419795252 | 20.119453924914676 | 12.150170648464l6 |
| 2 | RR | 43.088043478260865 | 27.826086956521738 | 17.434782608695652 |
| 3 | RO | 41.330549450549434 | 19.282051282051292 | 19.080586080586084 |
| 4 | AC | 40.047912087912081 | 20.329670329670336 | 20.010989010989018 |
| 5 | PI | 39.115086042064924 | 18.931166347992352 | 10.682600382409184 |

### LOWEST AVERAGE

```
SELECT
  c.customer_state,
  AVG(freight_value) mean_freight,
  AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY)) mean_time_to_delivery,
```

```
  AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY)) mean_diff_estimate
d_delivery
FROM `sql-scaler.TargetSQL.order_items` ot
JOIN `sql-scaler.TargetSQL.orders` o
  ON ot.order_id = o.order_id
JOIN `sql-scaler.TargetSQL.customers` c
  ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date is NOT NULL
GROUP BY 1
ORDER BY 2
LIMIT 5
```

| Row | customer_state | mean_freight | mean_time_to_delivery | mean_diff_estimated_delivery |
|---|---|---|---|---|
| 1 | SP | 15.114994078763218 | 8.25960855241909 | 10.26559438451439 |
| 2 | PR | 20.471816250663817 | 11.480793060718735 | 12.533899805275263 |
| 3 | MG | 20.6258372687155 | 11.515522180072811 | 12.397151041263502 |
| 4 | RJ | 20.909784391347358 | 14.689382157500321 | 11.14449314293797 |
| 5 | DF | 21.072161358811066 | 12.501486199575384 | 11.274734607218704 |

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

#### HIGHEST TIME TO DELIVERY

```
SELECT
  c.customer_state,
  AVG(freight_value) mean_freight,
  AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY)) mean_time_to_delivery,
  AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY)) mean_diff_estimate
d_delivery
FROM `sql-scaler.TargetSQL.order_items` ot
JOIN `sql-scaler.TargetSQL.orders` o
  ON ot.order_id = o.order_id
JOIN `sql-scaler.TargetSQL.customers` c
  ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date is NOT NULL
GROUP BY 1
ORDER BY 3 DESC
LIMIT 5
```

| Row | customer_state | mean_freight | mean_time_to_delivery | mean_diff_estimated_delivery |
|---|---|---|---|---|
| 1 | RR | 43.088043478260865 | 27.826086956521738 | 17.434782608695652 |
| 2 | AP | 34.1604938271605 | 27.753086419753075 | 17.444444444444443 |
| 3 | AM | 33.310613496932525 | 25.963190184049076 | 18.975460122699381 |
| 4 | AL | 35.870655737704922 | 23.992974238875881 | 7.9765807962529349 |
| 5 | PA | 35.629013282732458 | 23.301707779886126 | 13.37476280834913 |

#### LOWEST TIME TO DELIVERY

```
SELECT
```

```
  c.customer_state,
  AVG(freight_value) mean_freight,
  AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY)) mean_time_to_delivery,
  AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY)) mean_diff_estimate
d_delivery
FROM `sql-scaler.TargetSQL.order_items` ot
JOIN `sql-scaler.TargetSQL.orders` o
  ON ot.order_id = o.order_id
JOIN `sql-scaler.TargetSQL.customers` c
  ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date is NOT NULL
GROUP BY 1
ORDER BY 3
LIMIT 5
```

| Row | customer_state | mean_freight | mean_time_to_delivery | mean_diff_estimated_delivery |
|---|---|---|---|---|
| 1 | SP | 15.114994078763255 | 8.25960855241901 | 10.265594384514429 |
| 2 | PR | 20.47181625066376 | 11.480793060718726 | 12.533899805275258 |
| 3 | MG | 20.625837268715657 | 11.515522180072761 | 12.39715104126349 |
| 4 | DF | 21.07216135881108 | 12.501486199575361 | 11.274734607218669 |
| 5 | SC | 21.506627623230823 | 14.520985846754524 | 10.668862859931693 |

### c. Top 5 states where delivery is really fast/ not so fast compared to estimated date

#### FAST

```
SELECT
  c.customer_state,
  AVG(freight_value) mean_freight,
  AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY)) mean_time_to_delivery,
  AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY)) mean_diff_estimate
d_delivery
FROM `sql-scaler.TargetSQL.order_items` ot
JOIN `sql-scaler.TargetSQL.orders` o
  ON ot.order_id = o.order_id
JOIN `sql-scaler.TargetSQL.customers` c
  ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date is NOT NULL
GROUP BY 1
ORDER BY 4
LIMIT 5
```

| Row | customer_state | mean_freight | mean_time_to_delivery | mean_diff_estimated_delivery |
|---|---|---|---|---|
| 1 | AL | 35.870655737704922 | 23.992974238875881 | 7.9765807962529349 |
| 2 | MA | 38.492712500000032 | 21.203750000000017 | 9.1099999999999923 |
| 3 | SE | 36.573173333333358 | 20.978666666666651 | 9.1653333333333276 |
| 4 | ES | 22.02897977528092 | 15.192808988764023 | 9.7685393258427116 |
| 5 | BA | 26.487556339940287 | 18.774640238935675 | 10.119467825142538 |

#### NOT SO FAST

```
SELECT
  c.customer_state,
  AVG(freight_value) mean_freight,
  AVG(DATE_DIFF(order_delivered_customer_date, order_purchase_timestamp,DAY)) mean_time_to_delivery,
  AVG(DATE_DIFF(order_estimated_delivery_date, order_delivered_customer_date,DAY)) mean_diff_estimate
d_delivery
FROM `sql-scaler.TargetSQL.order_items` ot
JOIN `sql-scaler.TargetSQL.orders` o
  ON ot.order_id = o.order_id
JOIN `sql-scaler.TargetSQL.customers` c
  ON o.customer_id = c.customer_id
WHERE order_delivered_customer_date is NOT NULL
GROUP BY 1
ORDER BY 4 DESC
LIMIT 5
```

| Row | customer_state | mean_freight | mean_time_to_delivery | mean_diff_estimated_delivery |
|---|---|---|---|---|
| 1 | AC | 40.047912087912081 | 20.329670329670336 | 20.010989010989018 |
| 2 | RO | 41.330549450549434 | 19.282051282051292 | 19.080586080586084 |
| 3 | AM | 33.310613496932525 | 25.963190184049076 | 18.975460122699381 |
| 4 | AP | 34.1604938271605 | 27.753086419753075 | 17.444444444444443 |
| 5 | RR | 43.088043478260865 | 27.826086956521738 | 17.434782608695652 |

## Payment type analysis: Join “payments” dataset with the existing data on order_id

### a. Count of orders for different payment types

```
SELECT
  payment_type,
  COUNT(*) No_of_orders
FROM `sql-scaler.TargetSQL.orders` o
JOIN `sql-scaler.TargetSQL.payments` p
  ON p.order_id = o.order_id
GROUP BY 1
ORDER BY 2 DESC
```

| Row | payment_type | No_of_orders |
|---|---|---|
| 1 | credit_card | 76795 |
| 2 | UPI | 19784 |
| 3 | voucher | 5775 |
| 4 | debit_card | 1529 |
| 5 | not_defined | 3 |

### b. Distribution of payment installments and count of orders

```
SELECT
  payment_installments,
  COUNT(*) No_of_orders
```

```
FROM `sql-scaler.TargetSQL.orders` o
JOIN `sql-scaler.TargetSQL.payments` p
  ON p.order_id = o.order_id
GROUP BY 1
ORDER BY 2 DESC
```

| Row | payment_installments | No_of_orders |
|---|---|---|
| 1 | 1 | 52546 |
| 2 | 2 | 12413 |
| 3 | 3 | 10461 |
| 4 | 4 | 7098 |
| 5 | 10 | 5328 |
| 6 | 5 | 5239 |
| 7 | 8 | 4268 |
| 8 | 6 | 3920 |
| 9 | 7 | 1626 |
| 10 | 9 | 644 |

| Row | payment_installments | No_of_orders |
|---|---|---|
| 15 | 24 | 18 |
| 16 | 20 | 17 |
| 17 | 13 | 16 |
| 18 | 14 | 15 |
| 19 | 17 | 8 |
| 20 | 16 | 5 |
| 21 | 21 | 3 |
| 22 | 0 | 2 |
| 23 | 23 | 1 |
| 24 | 22 | 1 |

### c. Count of orders for different payment types Month over Month

```
SELECT
  EXTRACT(YEAR FROM order_purchase_timestamp) YEAR,
  EXTRACT(MONTH FROM order_purchase_timestamp) MONTH,
  payment_type,
  COUNT(*) No_of_orders
FROM `sql-scaler.TargetSQL.orders` o
JOIN `sql-scaler.TargetSQL.payments` p
  ON p.order_id = o.order_id
GROUP BY 1,2,3
ORDER BY 1,2,4 DESC
```

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

## Actionable Insights

- Though the total orders in 2018 are more than 2017, there is a decrease in monthly no. of orders in 2018 against the steady increase observed in the 2017.
- For 2017 and 2018 we observe after top 7 categories, by sales, the sale is very less for all other categories.
  Also some categories are redundant with respect to some other, broader categories for example:  Furniture- Room furniture, Art-Art and craft etc.
- The sales on weekends is least compared to the week days.
- There is a sharp gap between sales for dawn-morning and afternoon-night time.
- A heavy majority of orders are from SP state. There is a higly significant difference in orders from other states compared to SP.
- Similar scenario exists for cities with Sao Paulo have very high orders in comparison to other cities.
- Though most orders being from SP the highest mean price of order is of PB.
- Highest mean freight is of RR despite having very low orders and total price along with the longest mean time to delivery. These factors might be a reason for low sales in this state.
- SP has lowest mean freight value along with fastest average time to delivery , which evidently is a factor for SP having highest sales.
- Customers heavily prefer credit cards for payments compared to other modes.
- A very high percentage of orders have 1 payment installment and only 2 orders have zero installments implying customers are more inclined towards paying for orders in 1 installment

## Recommendations

- Find the factors responsible low monthly orders in 2018.
- Redefine the categories with sub categories for better categorization and tracking.
- Providing special discounts or offers specifically targeting the low selling categories.
- Any new launches / schemes / programs must be organized in afternoon/ night due to higher activity during this period.
- Sale days / offers may start at dawn to increases orders during that time.
- A deeper study is required for the factors that are resulting in very high sales in specific state/city and very low in others. After that only any valid recommendations can be made.
- A strategy needs to be developed to reduce time to delivery and the freight values, to attract more customers.
- Due to higher preference to credit cards, some loyalty programs may be planned in collaborations with card companies to expand the customer base.