A SQL PROJECT ON BRAZIL'S TARGET STORE DATASET

PROJECT NUMBER 1

BAHADURSHA A V L SAINADH
SCALER-JULY 2022 BEGINNER BATCH

SUBMITTED ON: 18TH JANUARY, 2023
LAST DATE TO SUBMIT: 18TH JANUARY, 2023
SCALER DATA SCIENCE, ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
SCALER NEOVARSITY / WOOLF UNIVERSITY

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

In Scaler Data Science, Artificial Intelligence and Machine Learning course, first homework project assigned on TARGET store dataset. It is preferred to be solved by using Google's BigQuery Sandbox SQL platform. Last date to submit the project is 18th January, 2023.

1.1 INTRODUCTION TO TARGET CORPORATION

Target Corporation is a multi-national company headquartered in Minneapolis, Minnesota, USA. It is founded by George Dayton, Douglas Dayton & John Geisse. Previously Target Corporation is known as Dayton Corporation which was founded on June 24, 1902. It is a general merchandise retailer with 1,948 stores in US, 51 supply chain facilities in the US, 1 global capabilities centre in Bengaluru, India and nearly 20 sourcing offices globally.

Target Tag line is "Expect More, Pay Less". Target makes itself a preferred shopping destination by offering outstanding value, inspiration, innovation and an exceptional guest experience that no other retailer can deliver.

1.2 BUSINESS CASE DETAILS

This business case has information of 100k orders from 2016 to 2018 made at Target in Brazil. Its features allow viewing an order from multiple dimensions: from order status, price, payment and freight performance to customer location, product attributes and finally reviews written by customers. Data is available in 8 Tables in .csv (Comma delimited) format. Tables are named as following:

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

1.2.1 Features of customers.csv

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 the location of the consumer.
customer_city	Name of the City from where order is made.
customer_state	State Code from where order is made (Ex- Sao Paulo-SP)

1.2.2 Features of geolocations.csv

Features	Description
geolocation_zip_code_prefix	First 5 digits of zip code
geolocation_lat	Latitude
geolocation_lng	Longitude
geolocation_city	City name
geolocation_state	State

1.2.3 Features of orders.csv

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.

1.2.4 Features of order_items.csv

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	A unique Id of the seller registered in Target.
shipping_limit_date	The date before which shipping of the ordered product must be completed.
price	Actual price of the products ordered.
freight_value	Price rate at which a product is delivered from one point to another.

1.2.5 Features of order_reviews.csv

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 the 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.

1.2.6 Features of payments.csv

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. (Ex-Credit Card)
payment_installments	Number of installments in case of EMI purchase.
payment_value	Total amount paid for the purchase order.

1.2.7 Features of products.csv

Features	Description
product_id	A unique identifier for the proposed project.
product_category_name	Name of the product category
product_name_length	length of the string which specifies the name given to the products ordered.
product_description_length	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 centimetres.
product_height_cm	Height of the products ordered in centimetres.
product_width_cm	width of the product ordered in centimetres.

1.2.8 Features of sellers.csv

Features	Description
seller_id	Unique Id of the seller registered
seller_zip_code_prefix	Zip Code of the location of the seller.
seller_city	Name of the City of the seller.
seller_state	State Code (Ex- Sao Paulo-SP)

1.3 ENTITY RELATIONSHIP DIAGRAM

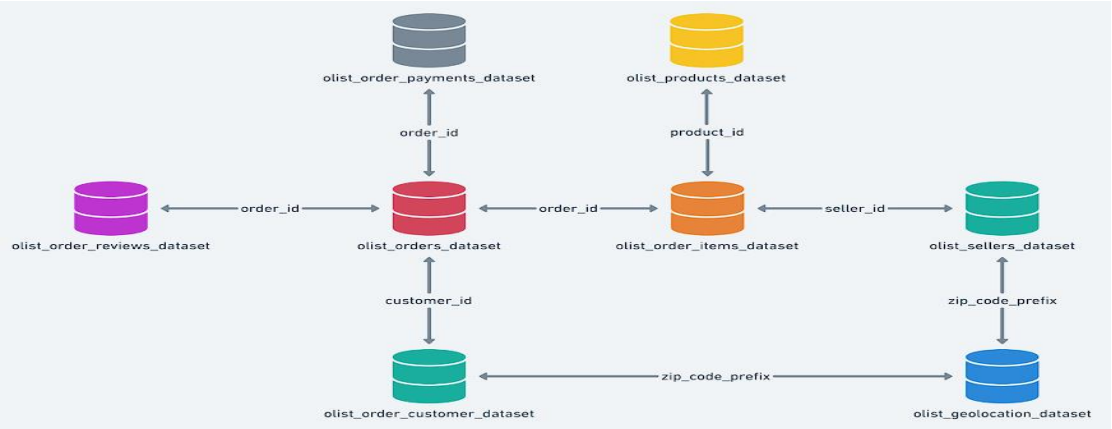


Figure 1-1 High level overview of relationship between datasets

CHAPTER 2: IMPORTING THE DATASET AND PRELIMINARY EXPLORATORY ANALYSIS

2.1 IMPORTING THE DATASET INTO GOOGLE'S BIGQUERY SANDBOX

Firstly, Brazil's Target Store Data set is provided by Scaler in 8 Tables in .csv format through the Google Drive link. Download them to local storage. After signing in Google's BigQuery Sandbox SQL platform using a Gmail account, create a new project named as "scaler-ds-ai-ml-projects". Create a new data set/schema under "scaler-ds-ai-ml-projects" named as "TARGET_BRAZIL_DATA". One after another, create 8 tables under "target_brazil_data" schema by uploading 8 .csv format files (comma delimited) provided by Scaler. Use auto-detect schema option while creating tables.

2.1.1 Why BigQuery Sandbox?

BigQuery Sandbox is a SQL platform having following characteristics –

1. Can be easily accessed through internet from anywhere using a Gmail account and using the BigQuery Sandbox link.
2. Free to use without providing any credit card information
3. Can load Big Data in short amount of time (MySQL does not support Big Data, takes lot of time while importing the Big Data)
4. Sandbox gives 10GB of active storage and 1TB of processed query data per month
5. All datasets have default table expiration time. (About 60 days)
6. Free license does not support streaming data, data manipulation language and data transfer services.

2.2 DATATYPES OF EVERY COLUMN IN ALL TABLES

Query:

```
SELECT
  TABLE_CATALOG,
  TABLE_SCHEMA,
  TABLE_NAME,
  COLUMN_NAME,
  IS_NULLABLE,
  DATA_TYPE
FROM
  TARGET_BRAZIL_DATA.INFORMATION_SCHEMA.COLUMNS
```

Result:

Table 2.1 Information_schema.columns

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	IS_NULLABLE	DATA_TYPE
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_REVIEWS	review_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_REVIEWS	order_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_REVIEWS	review_score	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_REVIEWS	review_comment_title	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_REVIEWS	review_creation_date	YES	TIMESTAMP
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_REVIEWS	review_answer_timestamp	YES	TIMESTAMP

scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_category	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_name_length	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_description_length	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_photos_qty	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_weight_g	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_length_cm	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_height_cm	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PRODUCTS	product_width_cm	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	SELLERS	seller_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	SELLERS	seller_zip_code_prefix	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	SELLERS	seller_city	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	SELLERS	seller_state	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	GEOLOCATION	geolocation_zip_code_prefix	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	GEOLOCATION	geolocation_lat	YES	FLOAT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	GEOLOCATION	geolocation_lng	YES	FLOAT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	GEOLOCATION	geolocation_city	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	GEOLOCATION	geolocation_state	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	CUSTOMERS	customer_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	CUSTOMERS	customer_unique_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	CUSTOMERS	customer_zip_code_prefix	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	CUSTOMERS	customer_city	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	CUSTOMERS	customer_state	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PAYMENTS	order_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PAYMENTS	payment_sequential	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PAYMENTS	payment_type	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PAYMENTS	payment_installments	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	PAYMENTS	payment_value	YES	FLOAT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	order_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	customer_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	order_status	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	order_purchase_timestamp	YES	TIMESTAMP
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	order_approved_at	YES	TIMESTAMP
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	order_delivered_carrier_date	YES	TIMESTAMP
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	order_delivered_customer_date	YES	TIMESTAMP
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDERS	order_estimated_delivery_date	YES	TIMESTAMP
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_ITEMS	order_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_ITEMS	order_item_id	YES	INT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_ITEMS	product_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_ITEMS	seller_id	YES	STRING
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_ITEMS	shipping_limit_date	YES	TIMESTAMP
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_ITEMS	price	YES	FLOAT64
scaler-ds-ai-ml-projects	TARGET_BRAZIL_DATA	ORDER_ITEMS	freight_value	YES	FLOAT64

Insights or Observations:

1. Only 4 variants of data types are used.
2. All date related attributes are in TIMESTAMP format (YYYY:MM:DD HH:MM:SS.SSSSSS UTC) (UTC – Universal Time Coordinated or Greenwich Meridian Time).
3. All attributes having numbers without decimals are INT64 format.
4. All attributes having numbers with decimals are FLOAT64 format.
5. All other attributes are in STRING format.

2.3 FINDING PRIMARY KEYS OF EACH AND EVERY TABLE

Query:

```
# Number of customer id's in customers table
SELECT
  COUNT(c.customer_id)
FROM
  `TARGET_BRAZIL_DATA.CUSTOMERS` as c;

# Number of Distinct customer_id's in customer table
SELECT
  COUNT(DISTINCT c.customer_id)
FROM
  `TARGET_BRAZIL_DATA.CUSTOMERS` as c;

# How many null values are present in customer_id column
SELECT
  COUNT(c.customer_id)
FROM
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
WHERE
  c.customer_id IS NULL;
```

Result:

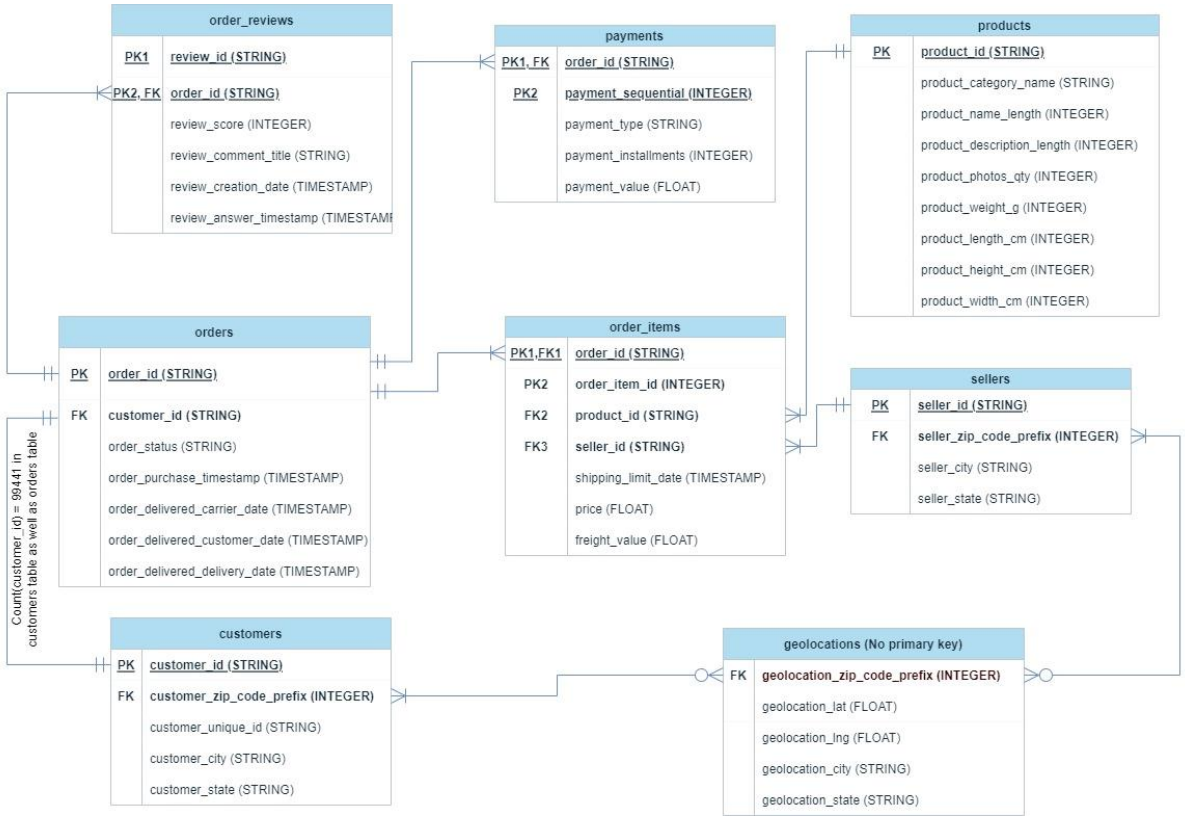
Number of customer_id's in customers table = 99441
Number of Distinct customer_id's in customer table = 99441
Number of Null values in customer_id columns = 0

Insights or Observations:

1. As number of customer_id's and number of distinct customer_id's are equal (unique) and there are no null values in the column, customer_id act as primary key for customers table.
2. Similarly, same procedure can be applied on each and every table to find primary keys.
3. Geolocation table is an exception for which there is no primary key. A surrogate key should be generated in geolocation table.

Table 2.2 Primary keys of each and every table

Tables	Primary Keys
customers	customer_id
geolocations	No primary key
orders	order_id
order_items	CONCAT(order_id, order_item_id)
order_reviews	CONCAT(review_id, order_id)
payments	CONCAT(order_id, payment_sequential)
products	product_id
sellers	seller id



2.4 TIME PERIOD OF GIVEN DATA

Query:

```
SELECT
  MIN(order_purchase_timestamp) AS Date_of_first_order_purchased,
  MAX(order_purchase_timestamp) AS Date_of_last_order_purchased,
  MAX(order_delivered_customer_date) AS Date_of_last_order_delivered,
  DATE_DIFF(MAX(order_purchase_timestamp),MIN(order_purchase_timestamp),DAY) AS Tim
e_period_of_given_data
FROM
  `TARGET_BRAZIL_DATA.ORDERS`
```

Result :

Date_of_first_order_purchase	Date_of_last_order_purchase	Date_of_last_order_delivered	Time_period_of_given_data
2016-09-04 21:15:19.000000 UTC	2018-10-17 17:30:18.000000 UTC	2018-10-17 13:22:46.000000 UTC	772

Insights or Observations:

1. Date of first order purchased was 4th September, 2016 and Date of last order purchased was 17th October, 2018.
2. Date of last order purchased and date of last order delivered were same (17th October, 2018).
3. Time period of the given data is 772 days i.e., 2 Years 42 days.

2.5 DISTINCT CITIES AND STATES OF CUSTOMERS WHO ORDERED IN 772 DAYS' TIME PERIOD

Query:

```
  # Cities and States of customers ordered during the given period
SELECT
  DISTINCT c.customer_city,
  c.customer_state
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  o.customer_id = c.customer_id;

   # Number of distinct customer cities
SELECT
  COUNT(DISTINCT c.customer_city) AS COUNT_OF_DIFFERENT_CITIES
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  o.customer_id = c.customer_id;

  # Number of distinct customer states
SELECT
  COUNT(DISTINCT c.customer_state) AS COUNT_OF_DIFFERENT_STATES
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  o.customer_id = c.customer_id;
```

Result: (first 10 rows only)

customer_city	customer_state
rio de janeiro	RJ
sao leopoldo	RS
general salgado	SP
brasilia	DF
paranavai	PR
cuiaba	MT
sao luis	MA
maceio	AL
hortolandia	SP
varzea grande	MT

1. Number of distinct customer cities = 4119
2. Number of distinct customer states = 27

Insights or Observations:

1. Customers from 4119 cities and 27 states are ordered in Brazil Target stores from 2016 to 2018

CHAPTER 3: IN-DEPTH EXPLORATION

3.1 COMPLETE SCENARIO OF E-COMMERCE TREND (SALES OVER TIME)

3.1.1 Complete Scenario of E-commerce trend by comparing sales over month-on-month format

```
 # Create a temporary table using WITH, which changes purchase timestamp into Year
& month columns for SUCCESSFULLY DELIVERED order_ids from orders table.
WITH month_order_id AS
(
SELECT
  o.order_id as order_id,
  EXTRACT(MONTH FROM o.order_purchase_timestamp) as month,
  EXTRACT(YEAR FROM o.order_purchase_timestamp) as year
FROM `TARGET_BRAZIL_DATA.ORDERS` as o
WHERE o.order_status = "delivered"
ORDER BY month
),

# Create a temporary table using WITH which finds sales by grouping order_id in pay
ments table
Group_order_id_sales AS
(
  SELECT
    p.order_id,
    SUM(p.payment_value) as sales_by_order_id
  FROM `TARGET_BRAZIL_DATA.PAYMENTS` as p
  GROUP BY p.order_id
)

# Query for month on month vs sales
SELECT
  myo.month,
  myo.year,
  SUM(gois.sales_by_order_id) AS monthly_sales
FROM
  month_order_id as myo
LEFT JOIN
  Group_order_id_sales as gois ON gois.order_id = myo.order_id
GROUP BY
  myo.month,myo.year
HAVING monthly_sales IS NOT NULL
ORDER BY myo.month,myo.year ASC;
```

Result:

Table 3.1 E-commerce trend by comparing sales over month-on-month format

month	year	monthly_sales	month	year	monthly_sales
1	2017	127545.67	**7**	2017	566403.93
1	2018	1078606.86	**7**	2018	1027903.86
2	2017	271298.65	**8**	2017	646000.61

2	2018	966510.88	**8**	2018	985414.28
3	2017	414369.39	**9**	2017	701169.99
3	2018	1120678	**10**	2016	46566.71
4	2017	390952.18	**10**	2017	751140.27
4	2018	1132933.95	**11**	2017	1153528.05
5	2017	567066.73	**12**	2016	19.62
5	2018	1128836.69	**12**	2017	843199.17
6	2017	490225.6			
6	2018	1012090.68			

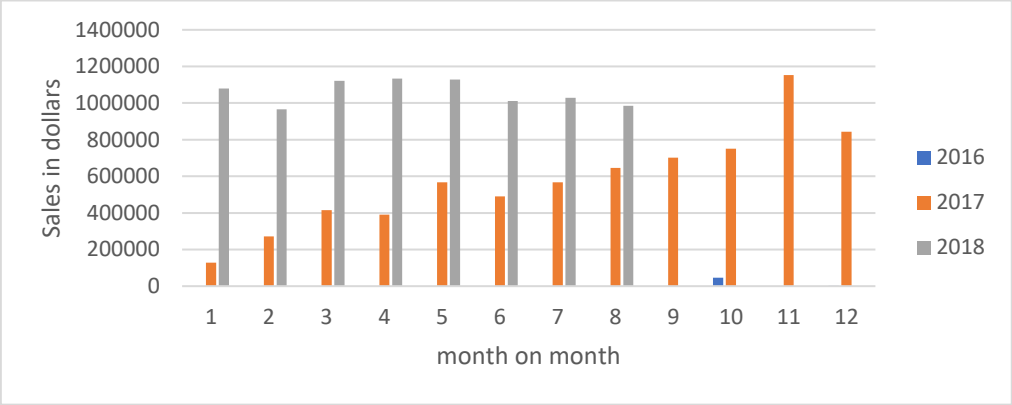


Figure 3-1 E-commerce trend by comparing sales over month-on-month format

Insights or Observations:

1. Significant number of successful sales were happened from January 2017 to August 2018. Out of this window of time, very negligible sales were happened.
2. In 2017, Sales was gradually increased from January to December. November 2017 is the month where the highest sales are occurred.
3. In 2018, Sales was maintained approximately similar to average value which is around 1000000 dollars.

3.1.2 Complete Scenario of E-commerce trend by comparing sales over month-year format

Query:

```
 # Create a temporary table using WITH which changes purchase timestamp into Year-month format for every order_id which is delivered successfully from orders table.
WITH
  month_year_order_id AS (
  SELECT
    o.order_id AS order_id,
    FORMAT_TIMESTAMP('%Y-%m',o.order_purchase_timestamp) AS month_year
  FROM
    `TARGET_BRAZIL_DATA.ORDERS` AS o
  WHERE
    o.order_status = "delivered"
  ORDER BY
    month_year ),

# Create a temporary table using WITH, which finds sales by grouping order_id in
```

```
payments table
  Group_order_id_sales AS (
  SELECT
    p.order_id,
    SUM(p.payment_value) AS sales_by_order_id
  FROM
    `TARGET_BRAZIL_DATA.PAYMENTS` AS p
  GROUP BY
    p.order_id )

# Query for month_year and monthly sales group by and order by month-year
SELECT
  myo.month_year,
  SUM(gois.sales_by_order_id) AS monthly_sales
FROM
  month_year_order_id AS myo
LEFT JOIN
  Group_order_id_sales AS gois
ON
  gois.order_id = myo.order_id
GROUP BY
  myo.month_year
HAVING
  monthly_sales IS NOT NULL
ORDER BY
  myo.month_year ASC;
```

Result:

Table 3.2 Monthly sales vs Month-year (considering only successfully delivered orders)

month_year	monthly_sales
2016-10	46566.71
2016-12	19.62
2017-01	127545.67
2017-02	271298.65
2017-03	414369.39
2017-04	390952.18
2017-05	567066.73
2017-06	490225.6
2017-07	566403.93
2017-08	646000.61
2017-09	701169.99

month_year	monthly_sales
2017-10	751140.27
2017-11	1153528.05
2017-12	843199.17
2018-01	1078606.86
2018-02	966510.88
2018-03	1120678
2018-04	1132933.95
2018-05	1128836.69
2018-06	1012090.68
2018-07	1027903.86
2018-08	985414.28

Figure 3-2 E-commerce trend by comparing sales over month-year format

Insights or Observations:

1. 2016 sales were interestingly very negligible compared to other. It represents something effected hugely on sales like prolonged recession or inflation or bankruptcy or political-military reforms
2. But 2017 was a year where sales were picked up gradually growth.

3.1.3 Complete Scenario of E-commerce trend by comparing sales over only month format and calculating the percentage difference by comparing with previous month

Query:

```
 # Create a temporary table using WITH which changes purchase timestamp into Only month format for every order_id which is delivered successfully from orders table.
WITH
  month_order_id AS (
  SELECT
    o.order_id AS order_id,
    EXTRACT(MONTH
    FROM
      o.order_purchase_timestamp) AS month
  FROM
    `TARGET_BRAZIL_DATA.ORDERS` AS o
  WHERE
    o.order_status = "delivered"
  ORDER BY
    month ),

# Create a temporary table using WITH, which finds sales by grouping order_id in payments table
  Group_order_id_sales AS (
  SELECT
    p.order_id,
    SUM(p.payment_value) AS sales_by_order_id
  FROM
    `TARGET_BRAZIL_DATA.PAYMENTS` AS p
  GROUP BY
    p.order_id )

# Query for month and monthly sales group by and order by month
SELECT
```

```
  myo.month,
  SUM(gois.sales_by_order_id) AS monthly_sales
FROM
  month_order_id AS myo
LEFT JOIN
  Group_order_id_sales AS gois
ON
  gois.order_id = myo.order_id
GROUP BY
  myo.month
HAVING
  monthly_sales IS NOT NULL
ORDER BY
  myo.month ASC;
```

Result:

Table 3.2 E-commerce trend by comparing sales over only month format and calculating the percentage difference by comparing with previous month

month	monthly_sales	percent difference (calculated using excel)
1	**1206152.53**	0
2	**1237809.53**	2.624626588
3	**1535047.39**	24.0132147
4	**1523886.13**	-0.727095468
5	**1695903.42**	11.28806717
6	**1502316.28**	-11.41498612
7	**1594307.79**	6.123311797
8	**1631414.89**	2.327474044
9	**701169.99**	-57.02074351
10	**797706.98**	13.76798656
11	**1153528.05**	44.60548534
12	**843218.79**	-26.90088551
Total sales	**15422461.77**	

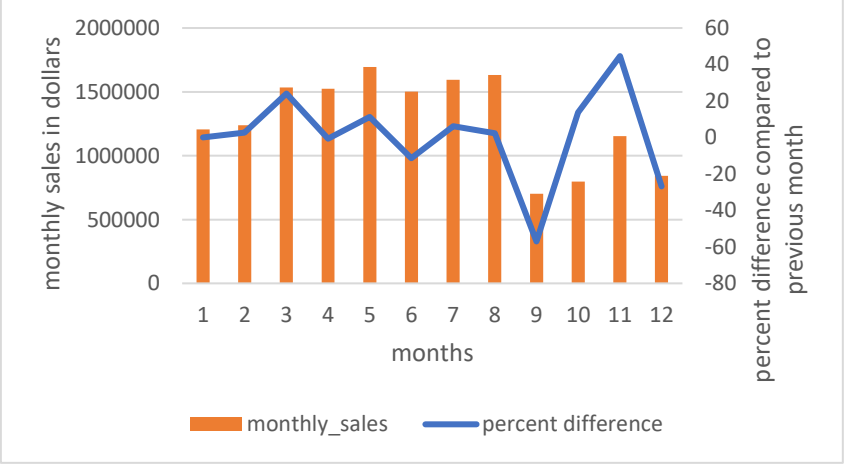


Figure 3-3 E-commerce trend by comparing sales over only month format and calculating the percentage difference by comparing with previous month

Insights or observations:

1. Among all months, September month is the month with lowest sales happened in all three years.
2. May month is the best month for sales cumulatively may be because of some festivals like CARNIVAL, Parintins Folklore fest, Festa Junaina etc.,

3.2 WHAT TIME DO BRAZILIAN CUSTOMERS TEND TO BUY?

Query:

```
   /* # Brasilia time = UTC - 3:00
# Dawn : 02:00:00 TO 06:59:59
# Morning : 07:00:00 TO 11:59:59
# Afternoon : 12:00:00 TO 16:59:59
# Evening : 17:00:00 TO 20:59:59
# Night : 21:00:00 TO 01:59:59
# Total day divided into 5 equal parts (approx.) */
SELECT
  part_of_the_day,
  COUNT(*) AS No_of_orders
FROM (
  SELECT
    purchase_time,
    CASE
      WHEN purchase_time BETWEEN "02:00:00" AND "06:59:59" THEN "Dawn"
      WHEN purchase_time BETWEEN "07:00:00" AND "11:59:59" THEN "Morning"
      WHEN purchase_time BETWEEN "12:00:00" AND "16:59:59" THEN "Afternoon"
      WHEN purchase_time BETWEEN "17:00:00" AND "20:59:59" THEN "Evening"
    ELSE
    "Night"
  END
    AS part_of_the_day
  FROM (
    SELECT
      EXTRACT(TIME
      FROM (TIMESTAMP_SUB(o.order_purchase_timestamp,INTERVAL 180 MINUTE))) AS purc
hase_time
    FROM
      `TARGET_BRAZIL_DATA.ORDERS` AS o) AS Brasilia_purchase_time)
GROUP BY
  part_of_the_day;

# WITHOUT CONVERTING THE TIMEZONES
SELECT
  part_of_the_day,
  COUNT(*) AS No_of_orders
FROM (
  SELECT
    purchase_time,
    CASE
      WHEN purchase_time BETWEEN "02:00:00" AND "06:59:59" THEN "Dawn"
      WHEN purchase_time BETWEEN "07:00:00" AND "11:59:59" THEN "Morning"
      WHEN purchase_time BETWEEN "12:00:00" AND "16:59:59" THEN "Afternoon"
      WHEN purchase_time BETWEEN "17:00:00" AND "20:59:59" THEN "Evening"
    ELSE
```

```
    "Night"
  END
    AS part_of_the_day
  FROM (
    SELECT
      EXTRACT(TIME
      FROM (o.order_purchase_timestamp)) AS purchase_time
    FROM
      `TARGET_BRAZIL_DATA.ORDERS` AS o) AS Brasilia_purchase_time)
GROUP BY
  part_of_the_day;
```

Result:

Table 3.3 Best Phase of the day for sales (changing the UTC timezone to Brasilia timezone)

part_of_the_day	No_of_orders
Dawn	9673
Morning	31837
Afternoon	31030
Evening	22349
Night	4552

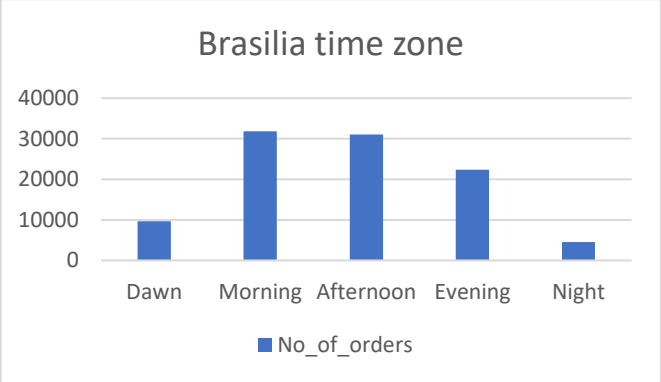
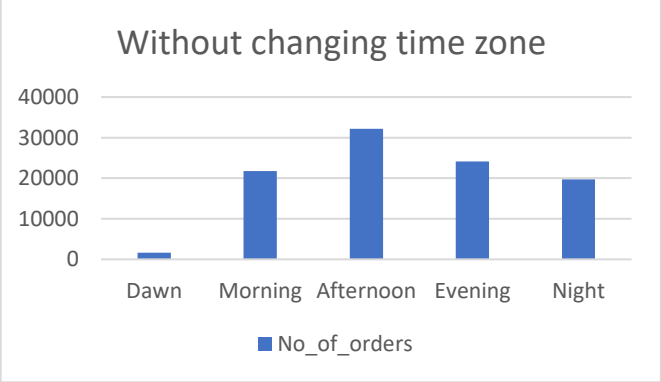


Table 3.4 Best phase of the day for sales (Without changing the timezones)

part_of_the_day	No_of_orders
Dawn	1678
Morning	21738
Afternoon	32211
Evening	24094
Night	19720



Insights or observations:

1. According Brasilia timezone , Morning hours from 7 AM to 11:59 AM was the best phase of the day where higher sales recorded.
2. According UTC timezone, Afternoon hours from 12 PM to 04:59 PM was the best phase of the day where higher sales recorded.
3. Dawn and Night hours were not good time for sales (It represents that some effective offers should be given to customers in these hours to increase the sales.)

CHAPTER 4: EVOLUTION OF E-COMMERCE IN BRAZIL REGION

4.1 E-COMMERCE TREND BY COMPARISON BETWEEN STATES AND TIME

4.1.1 Get month on month orders by states

```
# month on month orders by states
SELECT
  month,
  year,
  customer_state,
  COUNT(*) as No_of_orders
FROM (SELECT
  o.order_id,
  EXTRACT(MONTH from o.order_purchase_timestamp) as month,
  EXTRACT(YEAR from o.order_purchase_timestamp) as year,
  c.customer_state
FROM
  `TARGET_BRAZIL_DATA.ORDERS` as o
LEFT JOIN `TARGET_BRAZIL_DATA.CUSTOMERS` as c ON c.customer_id = o.customer_id)
GROUP BY month,year,customer_state
ORDER BY month,year,customer_state;
```

Result: (in pivoted format)

Table 4.1 month on month orders by states in pivoted format using excel

Row Labels	AC	AL	AM	AP	BA	CE	DF	ES	GO	MA	MG	MS	MT	PA	PB	PE	PI	PR	RJ	RN	RO	RR	RS	SC	SE	SP	TO
1		**8**	**39**	**12**	**11**	**264**	**99**	**151**	**159**	**164**	**66**	**971**	**71**	**96**	**82**	**33**	**113**	**55**	**443**	**990**	**51**	**23**	**2**	**427**	**345**	**24**	**3351**	**19**
2017		2	2			25	9	13	12	18	9	108	1	11	12	2	9	7	65	97	5	3		54	31	4	299	2
2018		6	37	12	11	239	90	138	147	146	57	863	70	85	70	31	104	48	378	893	46	20	2	373	314	20	3052	17
2		**6**	**39**	**16**	**4**	**273**	**101**	**196**	**186**	**176**	**67**	**1063**	**75**	**84**	**83**	**47**	**146**	**46**	**460**	**1176**	**31**	**25**	**7**	**473**	**316**	**27**	**3357**	**28**
2017		3	12	8	2	59	13	24	34	27	11	259	11	17	25	12	21	12	118	254	8	11	2	105	59	12	654	7
2018		3	27	8	2	214	88	172	152	149	56	804	64	67	58	35	125	34	342	922	23	14	5	368	257	15	2703	21
3		**4**	**40**	**14**	**8**	**340**	**126**	**207**	**182**	**199**	**77**	**1237**	**79**	**71**	**109**	**55**	**153**	**48**	**504**	**1302**	**52**	**29**	**8**	**569**	**362**	**43**	**4047**	**28**
2017		2	10	5	3	91	28	57	48	53	24	358	20	16	36	16	45	13	127	395	13	16	2	151	110	25	1010	8
2018		2	30	9	5	249	98	150	134	146	53	879	59	55	73	39	108	35	377	907	39	13	6	418	252	18	3037	20
4		**9**	**51**	**19**	**5**	**318**	**143**	**183**	**188**	**177**	**73**	**1061**	**58**	**92**	**107**	**51**	**154**	**50**	**500**	**1172**	**42**	**20**	**4**	**488**	**351**	**27**	**3967**	**33**
2017		5	23	13		93	43	35	46	41	27	275	15	27	36	20	40	13	114	338	10	9	2	139	105	13	908	14
2018		4	28	6	5	225	100	148	142	136	46	786	43	65	71	31	114	37	386	834	32	11	2	349	246	14	3059	19
5		**10**	**46**	**19**	**11**	**368**	**136**	**208**	**228**	**226**	**65**	**1190**	**74**	**104**	**75**	**47**	**174**	**56**	**524**	**1321**	**39**	**26**	**3**	**559**	**379**	**19**	**4632**	**34**
2017		8	27	10	5	127	62	64	94	87	33	428	29	37	35	18	68	25	213	488	17	9	2	208	152	11	1425	18
2018		2	19	9	6	241	74	144	134	139	32	762	45	67	40	29	106	31	311	833	22	17	1	351	227	8	3207	16
6		**7**	**34**	**8**	**4**	**307**	**121**	**220**	**204**	**184**	**59**	**1080**	**76**	**83**	**92**	**51**	**140**	**43**	**478**	**1128**	**49**	**22**	**8**	**526**	**321**	**37**	**4104**	**26**
2017		4	10	1	2	106	47	70	80	79	17	363	27	25	38	23	46	14	170	412	13	10	3	221	116	9	1331	8
2018		3	24	7	2	201	74	150	124	105	42	717	49	58	54	28	94	29	308	716	36	12	5	305	205	28	2773	18
7		**9**	**40**	**23**	**7**	**405**	**140**	**243**	**206**	**192**	**79**	**1111**	**74**	**85**	**96**	**79**	**210**	**52**	**523**	**1288**	**56**	**27**	**6**	**565**	**356**	**42**	**4381**	**23**
2017		5	17	5	1	155	53	77	83	77	39	453	25	38	39	27	73	20	203	571	27	11	1	249	158	14	1604	1
2018		4	23	18	6	250	87	166	123	115	40	658	49	47	57	52	137	32	320	717	29	16	5	316	198	28	2777	22
8		**7**	**34**	**9**	**5**	**323**	**130**	**232**	**200**	**213**	**70**	**1177**	**59**	**78**	**104**	**46**	**170**	**43**	**556**	**1307**	**40**	**23**		**599**	**365**	**43**	**4982**	**28**
2017		4	18	5	3	158	73	87	95	93	40	469	24	38	60	16	85	22	223	562	20	14		299	159	20	1729	15
2018		3	16	4	2	165	57	145	105	120	30	708	35	40	44	30	85	21	333	745	20	9		300	206	23	3253	13
9		**5**	**20**	**9**	**2**	**170**	**77**	**97**	**93**	**88**	**42**	**511**	**33**	**35**	**41**	**29**	**76**	**23**	**183**	**612**	**24**	**16**	**2**	**279**	**157**	**16**	**1648**	**17**
2016																						1	1				2	
2017		5	20	9	2	170	77	97	93	88	42	507	33	35	41	29	76	23	183	609	24	16	1	278	156	16	1638	17
2018												4								3					1		8	
10		**6**	**30**	**3**	**3**	**170**	**74**	**104**	**104**	**117**	**52**	**600**	**34**	**55**	**58**	**31**	**87**	**25**	**225**	**725**	**27**	**14**	**4**	**276**	**189**	**25**	**1908**	**13**
2016			2			4	8	6	4	9	4	40		3	4	1	7	1	19	56	4		1	24	11	3	113	
2017		6	28	3	3	166	66	98	100	108	48	560	34	52	54	30	80	23	206	668	23	14	3	252	178	22	1793	13
2018																		1		1							2	
11		**5**	**26**	**10**	**4**	**250**	**108**	**168**	**170**	**157**	**56**	**943**	**46**	**74**	**70**	**30**	**126**	**31**	**378**	**1048**	**44**	**17**	**2**	**422**	**303**	**27**	**3012**	**17**
2017		5	26	10	4	250	108	168	170	157	56	943	46	74	70	30	126	31	378	1048	44	17	2	422	303	27	3012	17
12		**5**	**14**	**6**	**4**	**192**	**81**	**131**	**113**	**127**	**41**	**691**	**36**	**50**	**58**	**37**	**103**	**23**	**271**	**783**	**30**	**11**		**283**	**193**	**20**	**2357**	**14**
2016																			1									
2017		5	14	6	4	192	81	131	113	127	41	691	36	50	58	37	103	23	270	783	30	11		283	193	20	2357	14

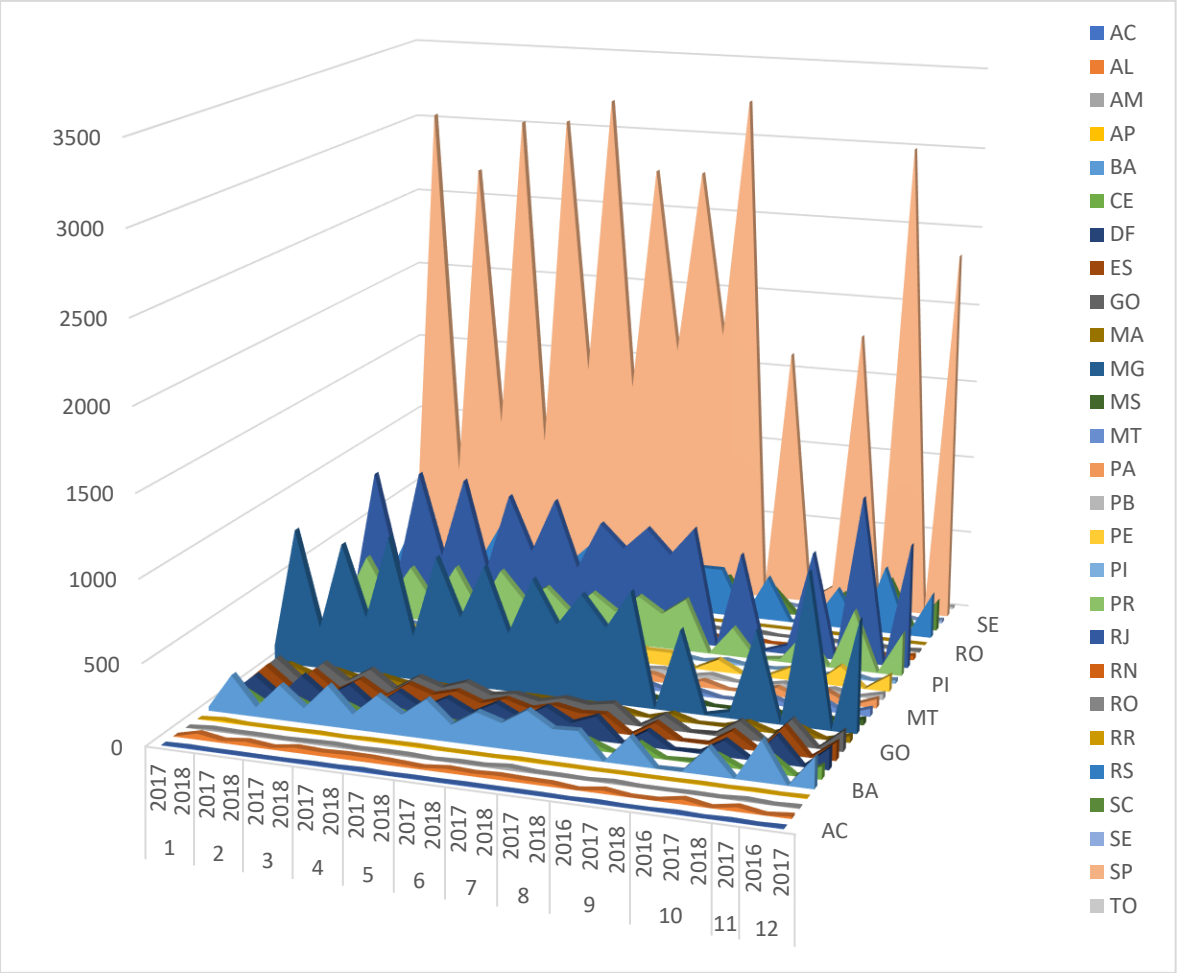


Figure 4-1 month on month orders by states - 3D Area chart

Insights or Observations:

1. State SP → Sao Paulo dominates other state in sales at every point of time
2. State RJ → Rio de janeiro was in second position and State MG → Minas Gerais was in third position.
3. Sharp decrease was observed from august to September at every state.

Query

```
SELECT
  month,
  customer_state,
  COUNT(*) AS No_of_orders
FROM (
  SELECT
    o.order_id,
    EXTRACT(MONTH
    FROM
      o.order_purchase_timestamp) AS month,
    c.customer_state
  FROM
    `TARGET_BRAZIL_DATA.ORDERS` AS o
  LEFT JOIN
    `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
```

```
  ON
    c.customer_id = o.customer_id)
GROUP BY
  month,
  customer_state
ORDER BY
  customer_state,
  month;
```

Result:

Table 4.2 monthly orders by states - pivoted chart using excel

Sum of No_of_orders	Column Labels												
Row Labels	1	2	3	4	5	6	7	8	9	10	11	12	Grand Tota
AC	8	6	4	9	10	7	9	7	5	6	5	5	81
AL	39	39	40	51	46	34	40	34	20	30	26	14	413
AM	12	16	14	19	19	8	23	9	9	3	10	6	148
AP	11	4	8	5	11	4	7	5	2	3	4	4	68
BA	264	273	340	318	368	307	405	323	170	170	250	192	3380
CE	99	101	126	143	136	121	140	130	77	74	108	81	1336
DF	151	196	207	183	208	220	243	232	97	104	168	131	2140
ES	159	186	182	188	228	204	206	200	93	104	170	113	2033
GO	164	176	199	177	226	184	192	213	88	117	157	127	2020
MA	66	67	77	73	65	59	79	70	42	52	56	41	747
MG	971	1063	1237	1061	1190	1080	1111	1177	511	600	943	691	11635
MS	71	75	79	58	74	76	74	59	33	34	46	36	715
MT	96	84	71	92	104	83	85	78	35	55	74	50	907
PA	82	83	109	107	75	92	96	104	41	58	70	58	975
PB	33	47	55	51	47	51	79	46	29	31	30	37	536
PE	113	146	153	154	174	140	210	170	76	87	126	103	1652
PI	55	46	48	50	56	43	52	43	23	25	31	23	495
PR	443	460	504	500	524	478	523	556	183	225	378	271	5045
RJ	990	1176	1302	1172	1321	1128	1288	1307	612	725	1048	783	12852
RN	51	31	52	42	39	49	56	40	24	27	44	30	485
RO	23	25	29	20	26	22	27	23	16	14	17	11	253
RR	2	7	8	4	3	8	6		2	4	2		46
RS	427	473	569	488	559	526	565	599	279	276	422	283	5466
SC	345	316	362	351	379	321	356	365	157	189	303	193	3637
SE	24	27	43	27	19	37	42	43	16	25	27	20	350
SP	3351	3357	4047	3967	4632	4104	4381	4982	1648	1908	3012	2357	41746
TO	19	28	28	33	34	26	23	28	17	13	17	14	280
Grand Total	8069	8508	9893	9343	10573	9412	10318	10843	4305	4959	7544	5674	99441

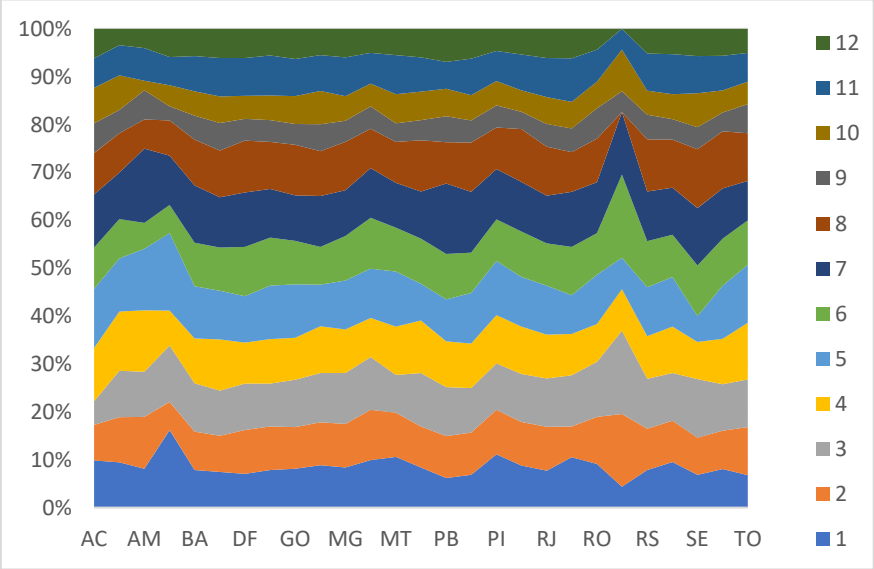


Figure 4-2 month on month orders by states - 100% Stacked area chart

Insights and observations:

1. 100% stacked area chart represents relative percentage differences. Combination of all area = 100 % orders = 99441 orders
2. Month 9 is lowest occupied area in stacked chart.
3. Month 8 is highest occupied area in stacked chart.

4.2 DISTRIBUTION OF CUSTOMERS ACROSS THE STATES IN BRAZIL:

Query:

```
  # Distribution OF customers across the states
SELECT
  c.customer_state,
  COUNT(*) AS No_of_customers
FROM
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state
```

Result:

Table 4.3 Distribution of customers across the states in Brazil

customer_state	No_of_customers	customer_state	No_of_customers	customer_state	No_of_customers
AC	81	MA	747	RJ	12852
AL	413	MG	11635	RN	485
AM	148	MS	715	RO	253
AP	68	MT	907	RR	46
BA	3380	PA	975	RS	5466
CE	1336	PB	536	SC	3637
DF	2140	PE	1652	SE	350
ES	2033	PI	495	SP	41746
GO	2020	PR	5045	TO	280

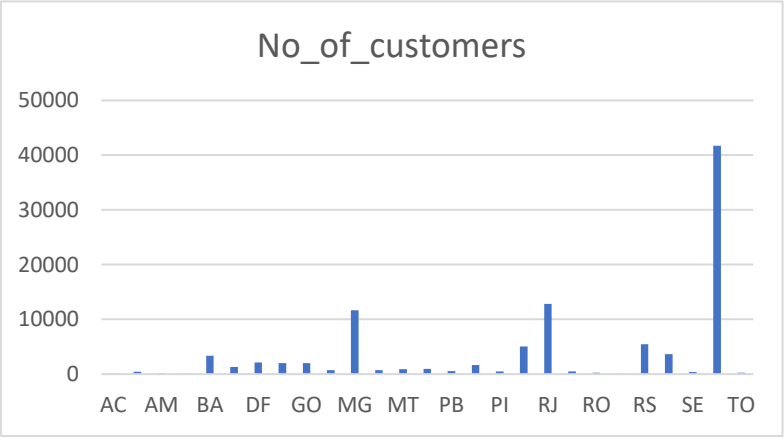


Figure 4-3 Distribution of customers across the states in Brazil

Insights or observations:

1. Number of customers are dominatingly high in SP→ Sao Paulo State. This results in higher sales in that state.

2. Next best states are MG and RJ, which were also plays major role in getting higher sales.

3. So Number of customers should be increased to get higher sales. To get more customers, card offers and membership offers should be provided.

CHAPTER 5: IMPACT ON ECONOMY: ANALYZE THE MONEY MOVEMENT BY E-COMMERCE BY LOOKING AT ORDER PRICES, FREIGHT AND OTHERS

5.1 % INCREASE IN COST OF ORDERS

5.1.1 Get % increase in cost of orders from 2017 to 2018 month over month (include months between Jan to Aug only) - You can use "payment_value" column in payments table

```
  /*# % increase OF cost OF orders FROM 2017 TO 2018 month over month considering F
ROM jan TO aug use payment value*/

# Sum of payment value group by order_id in payments table
WITH
  Total_cost_per_order_id AS (
  SELECT
    pa.order_id,
    SUM(pa.payment_value) AS Total_cost_per_order_id
  FROM
    `TARGET_BRAZIL_DATA.PAYMENTS` AS pa
  GROUP BY
    pa.order_id
  ORDER BY
    pa.order_id),

# Extract purchase year and purchase month from orders table where months are filte
red form Jan to august only
Year_month_order_id AS (
  SELECT
    o.order_id,
    EXTRACT(YEAR
    FROM
      o.order_purchase_timestamp) AS purchase_year,
    EXTRACT(MONTH
    FROM
      o.order_purchase_timestamp) AS purchase_month
  FROM
    `TARGET_BRAZIL_DATA.ORDERS` AS o
  WHERE
    EXTRACT(MONTH
    FROM
      o.order_purchase_timestamp) BETWEEN 01
    AND 08)

# Query the Year and month wise sales data
SELECT
  Y.purchase_month,
  Y.purchase_year,
  SUM(T.Total_cost_per_order_id) AS Total_sales_per_year
FROM
  Total_cost_per_order_id AS T
INNER JOIN
```

```
  Year_month_order_id AS Y
ON
  T.order_id = Y.order_id
WHERE
  Y.purchase_year = 2017
  OR Y.purchase_year = 2018
GROUP BY
  Y.purchase_month,Y.purchase_year
ORDER BY
  Y.purchase_month,Y.purchase_year
```

RESULT:

Table 5.1 % increase in cost of orders from 2017 to 2018 month over month

purchase_month	purchase_year	Total_sales_per_year
1	2017	138488.04
1	2018	1115004.18
2	2017	291908.01
2	2018	992463.34
3	2017	449863.6
3	2018	1159652.12
4	2017	417788.03
4	2018	1160785.48
5	2017	592918.82
5	2018	1153982.15
6	2017	511276.38
6	2018	1023880.5
7	2017	592382.92
7	2018	1066540.75
8	2017	674396.32
8	2018	1022425.32

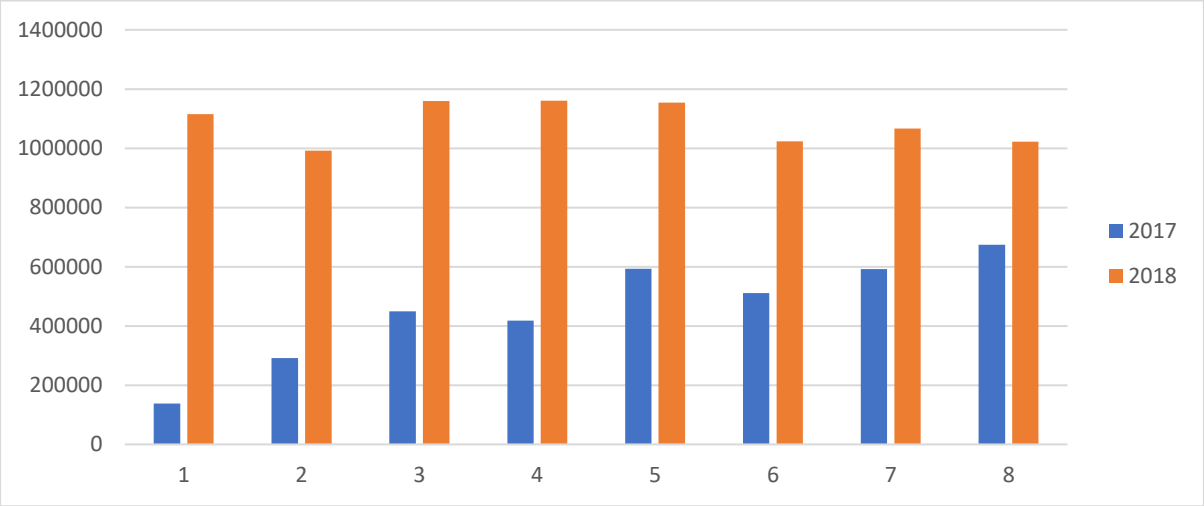


Figure 5-1 % increase in cost of orders from 2017 to 2018 month over month

5.1.2 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

Query:

```
  /*# % increase OF cost OF orders FROM 2017 TO 2018 considering FROM
jan TO aug use payment value*/

# Sum of payment value group by order_id in payments table
WITH
  Total_cost_per_order_id AS (
  SELECT
    pa.order_id,
    SUM(pa.payment_value) AS Total_cost_per_order_id
  FROM
    `TARGET_BRAZIL_DATA.PAYMENTS` AS pa
  GROUP BY
    pa.order_id
  ORDER BY
    pa.order_id),

# Extract purchase year from orders table where months are filtered form Jan to
august only
Year_order_id AS (
  SELECT
    o.order_id,
    EXTRACT(YEAR
    FROM
      o.order_purchase_timestamp) AS purchase_year
  FROM
    `TARGET_BRAZIL_DATA.ORDERS` AS o
  WHERE
    EXTRACT(MONTH
    FROM
      o.order_purchase_timestamp) BETWEEN 01
    AND 08)
```

```
# Query the Year wise sales data
SELECT
  Y.purchase_year,
  SUM(T.Total_cost_per_order_id) AS Total_sales_per_year
FROM
  Total_cost_per_order_id AS T
INNER JOIN
  Year_order_id AS Y
ON
  T.order_id = Y.order_id
WHERE
  Y.purchase_year = 2017
  OR Y.purchase_year = 2018
GROUP BY
  Y.purchase_year
```

Result:

Table 5.2 % increase in cost of orders from 2017 to 2018

purchase_year	Total_sales_per_year	% increase
2017	3669022	0
2018	8694734	136.9769

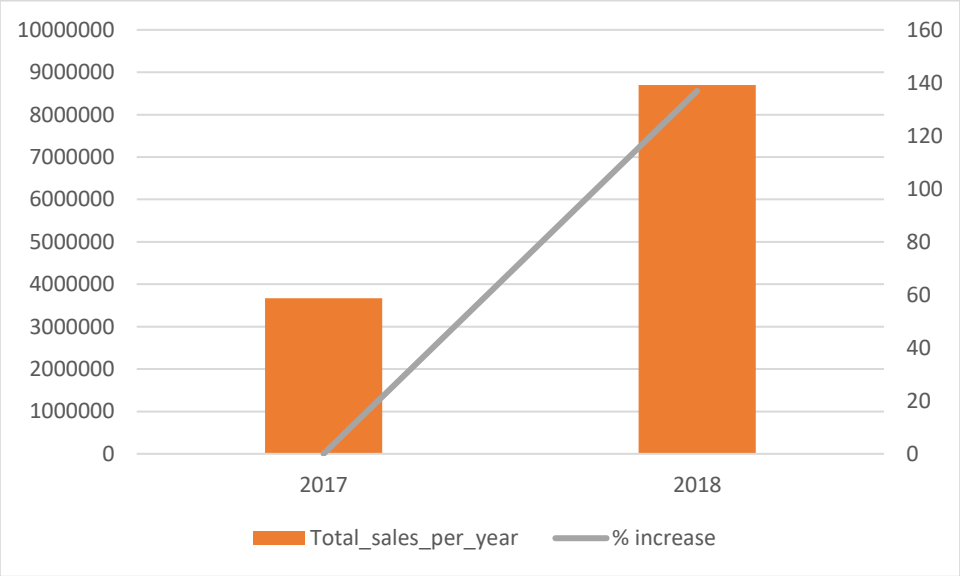


Figure 5-2 % increase in cost of orders from 2017 to 2018

Insights or observations:

1. From 2017 to 2018 , 1.3 times increase in sales was observed.

5.2 MEAN, SUM OF PRICE AND FREIGHT VALUE BY CUSTOMER STATES

Query:

```
SELECT
  c.customer_state,
  SUM(oi.price) AS sum_of_price,
  AVG(oi.price) AS mean_of_price,
  SUM(oi.freight_value) AS sum_of_freight_value,
  AVG(oi.freight_value) AS mean_of_freight_value
FROM
  `TARGET_BRAZIL_DATA.ORDER_ITEMS` AS oi
LEFT JOIN
  `TARGET_BRAZIL_DATA.ORDERS` AS o
ON
  oi.order_id = o.order_id
LEFT JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  o.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```

Result:

Table 5.3 Sum and mean of price and freight value for all states

customer_state	sum_of_price	mean_of_price	sum_of_freight_value	mean_of_freight_value
AC	15982.95	173.7277174	3686.75	40.07336957
AL	80314.81	180.8892117	15914.59	35.84367117
AM	22356.84	135.496	5478.89	33.20539394
AP	13474.3	164.3207317	2788.5	34.00609756
BA	511349.99	134.6012082	100156.68	26.36395894
CE	227254.71	153.7582612	48351.59	32.71420162
DF	302603.94	125.7705486	50625.5	21.04135495
ES	275037.31	121.9137012	49764.6	22.0587766
GO	294591.95	126.2717317	53114.98	22.76681526
MA	119648.22	145.2041505	31523.77	38.25700243
MG	1585308.03	120.7485741	270853.46	20.63016681
MS	116812.64	142.6283761	19144.03	23.374884
MT	156453.53	148.2971848	29715.43	28.16628436
PA	178947.81	165.6924167	38699.3	35.83268519
PB	115268.08	191.4752159	25719.73	42.72380399
PE	262788.03	145.5083223	59449.66	32.91786268
PI	86914.08	160.3580812	21218.2	39.14797048
PR	683083.76	119.0041394	117851.68	20.53165157

RJ	1824092.67	125.1178181	305589.31	20.96092393
RN	83034.98	156.9659357	18860.1	35.65236295
RO	46140.64	165.9735252	11417.38	41.06971223
RR	7829.43	150.5659615	2235.19	42.98442308
RS	750304.02	120.3374531	135522.74	21.73580433
SC	520553.34	124.6535776	89660.26	21.47036877
SE	58920.85	153.0411688	14111.47	36.65316883
SP	5202955.05	109.6536292	718723.07	15.14727539
TO	49621.74	157.5293333	11732.68	37.24660317

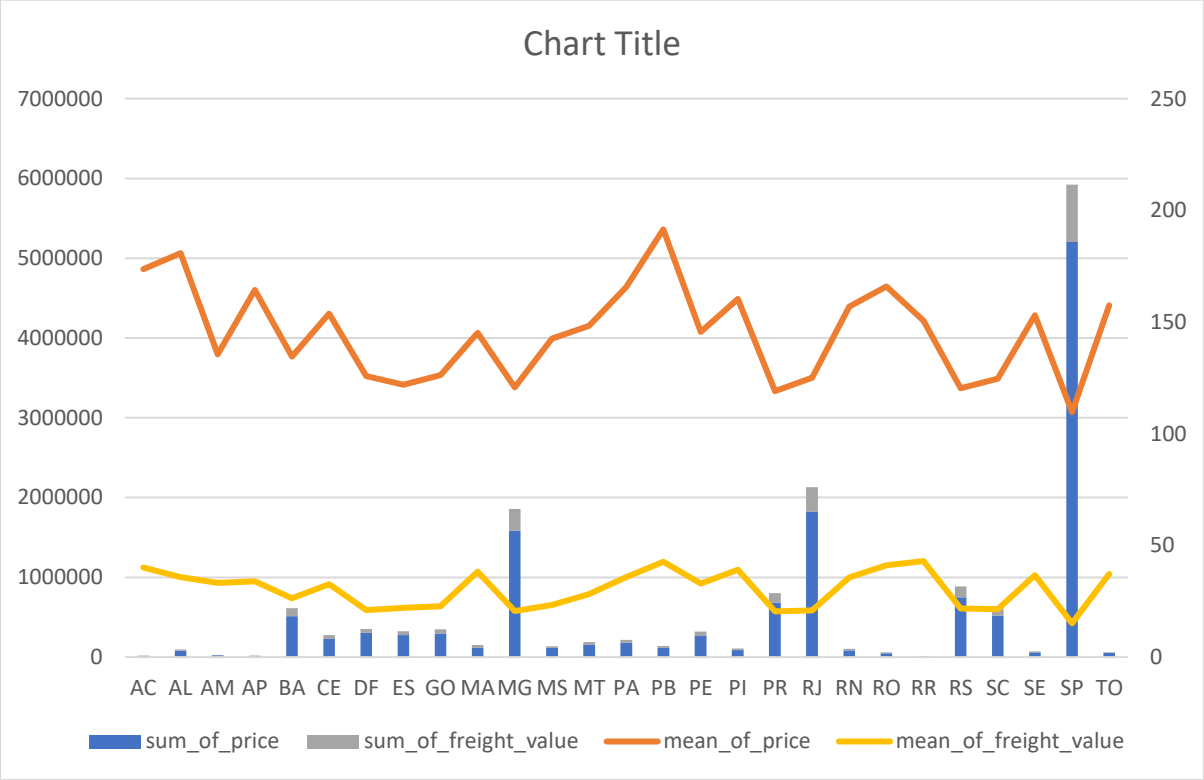


Figure 5-3 Sum and mean of price and freight value for all states

Insights or observations:

1. By observing the trends between mean of price and freight value, both of them are proportional to each other. Similar trend was occurred between them
2. Once again, Sao Paolo, Rio Di jenero and Minas Gerais states with excellent behaviour. The techniques used in these 3 states should be applied on other states too. At all these three states, mean line is dipped down compared to others implies that freight value and prices are significantly lesser compared to others.
3. Highest average of price is occurred at State PB → Paraiba state. So by decreasing the freight values and price values, we can expect increase in sales.

CHAPTER 6: ANALYSIS ON SALES, FREIGHT AND DELIVERY TIME

6.1 CALCULATE DAYS BETWEEN PURCHASING, DELIVERING AND ESTIMATED DELIVERY

Query:

```
  # days BETWEEN purchasing,delivery AND estimated delivery
SELECT
  o.order_id,
  DATE_DIFF(DATE(o.order_delivered_customer_date),DATE(o.order_purchase_timestamp),
DAY) AS Actual_delivery_days,
  DATE_DIFF(DATE(o.order_estimated_delivery_date),DATE(o.order_purchase_timestamp),
DAY) AS Estimated_delivery_days,
  DATE_DIFF(DATE(o.order_estimated_delivery_date),DATE(o.order_delivered_customer_d
ate),DAY) AS Days_between_actual_and_estimated
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
WHERE
  o.order_status = "delivered";
```

Result:

Table 6.1 Days between purchasing date, actual delivery date and estimated delivery date

order_id	Days_between_actual_and_estimated	Actual_delivery_days	Estimated_delivery_days
635c894d068ac37e6e03dc54eccb6189	2	31	33
3b97562c3aee8bdedcb5c2e45a50d5e1	1	33	34
68f47f50f04c4cb6774570cfde3a9aa7	2	30	32
276e9ec344d3bf029ff83a161c6b3ce9	-4	44	40
54e1a3c2b97fb0809da548a59f64c813	-4	41	37
fd04fa4105ee8045f6a0139ca5b49f27	-1	37	36
302bb8109d097a9fc6e9cefc5917d1f3	-5	34	29
66057d37308e787052a32828cd007e58	-6	39	33
19135c945c554eebfd7576c733d5ebdd	-2	36	34

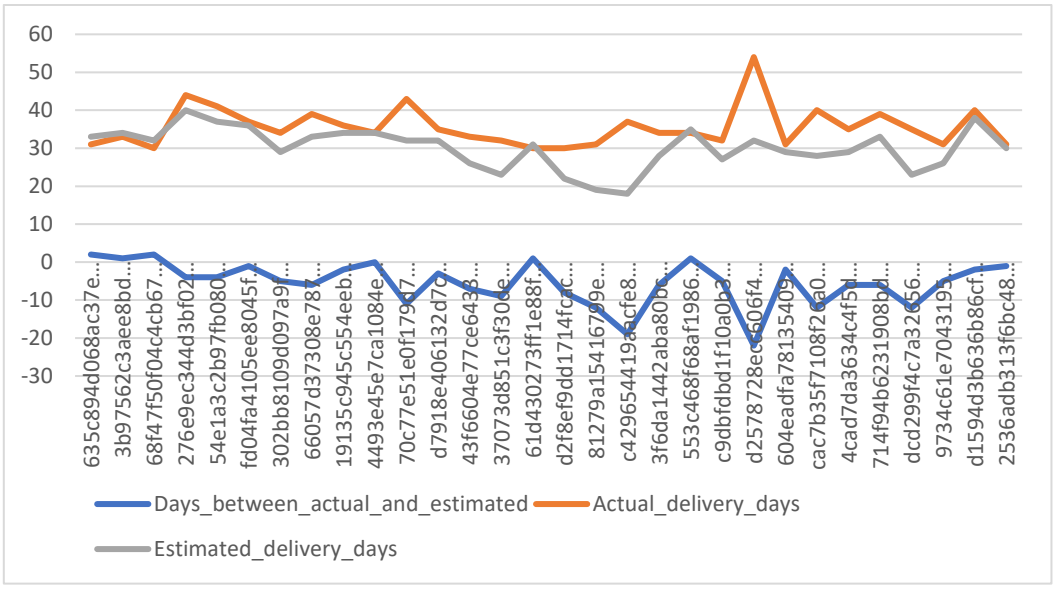


Figure 6-1 Days between purchasing date, actual delivery date and estimated delivery date

Insights or observations:

1. Actual delivery days on average is 12.87 days
2. Estimated delivery days on average is 24.37 days
3. Days between actual and estimated on average is 11.87 days
4. So Estimation algorithm should be updated as it is showing nearly 100% of error.

Query:

```
  # time BETWEEN purchasing,delivery AND estimated delivery
SELECT
  o.order_id,
  TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp),MIN
UTE) AS Actual_delivery_in_minutes,
  TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_purchase_timestamp),MIN
UTE) AS Estimated_delivery_in_minutes,
  TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_date
),MINUTE) AS Minutes_between_actual_and_estimated
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
WHERE
  o.order_status = "delivered";
```

Result:

Table 6.2 Time between purchasing, actual delivery and estimated delivery in minutes

order_id	Actual_delivery_in_minutes	Estimated_delivery_in_minutes	Minutes_between_actual_and_estimated
c158e9806f85a33877bdfd4f607b72e7	33903	47633	13729
b60b53ad0bb7dacacf2989fe27ad567a	18669	10676	-7992
c830f223aae08493ebecb52f29aa48ca	18577	36489	17912
a8aa2cd070eeac7e4368cae3d8222e2b	10419	11897	1477
813c55ce9b6baa8f879e064fbfbc9e07	17752	31618	13865
44558a1547e448b41c48c4087fe32ddd	2653	10272	7619
036b791897847cdb8e39df7943367474	9819	10585	765
1aba60c04110bdd421b250ea33187a31	31109	41919	10810
0312ecf90786def87f98aa19e0e0ce63	11227	10197	-1029

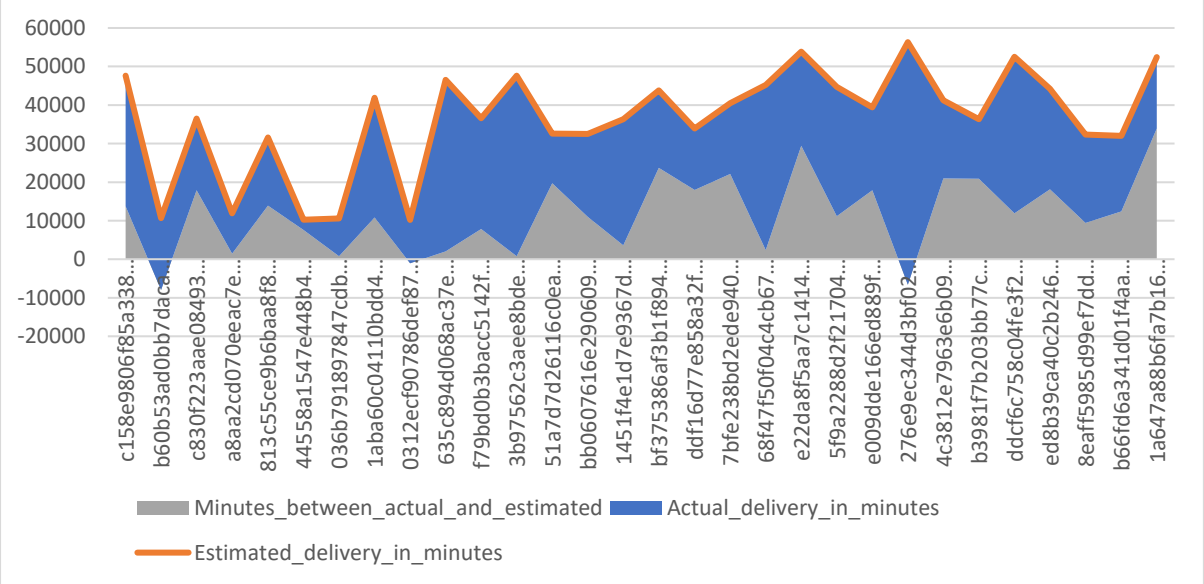


Figure 6-2 Time between purchasing, actual delivery and estimated delivery in minutes

Insights and observations:

1. Sum of actual delivery time and minutes between actual and estimate time = estimate delivery time. Similar behaviour observed in time units also.

6.2 GROUP DATA BY STATE, TAKE MEAN OF FREIGHT_VALUE, TIME_TO_DELIVERY, DIFF_ESTIMATED_DELIVERY

Query:

```
  /*# GROUP DATA BY state,take mean OF freight_value,time_to_delivery,diff_estimate
d_delivery*/
WITH
  total_freight_value_per_order AS (
  SELECT
    oi.order_id,
    SUM(oi.freight_value) AS total_freight_value
  FROM
    `TARGET_BRAZIL_DATA.ORDER_ITEMS` AS oi
  GROUP BY
    oi.order_id)
SELECT
  c.customer_state,
  AVG(total_freight_value) AS Mean_of_freight_value,
  AVG(TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp)
,MINUTE)) AS time_to_delivery,
  AVG(TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_
date),MINUTE)) AS diff_estimated_delivery
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  total_freight_value_per_order AS tfvpo
ON
  tfvpo.order_id = o.order_id
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  c.customer_state;
```

Result:

Table 6.3 Group data by state, take mean of freight_value, time_to_delivery, diff_estimate d_delivery

customer_state	time_to_delivery	diff_estimated_delivery	Mean_of_freight_value
AC	30290.9375	28910.4875	45.5154321
AL	35342.66751	11565.96222	38.72163017
AM	38052.95862	27146.22759	37.27136054
AP	39146	27445.25373	41.00735294
BA	27842.57463	14542.37193	29.82628946
CE	30623.36826	14558.13526	36.43676714
DF	18672.80337	16328.5625	23.82376471

ES	22736.11278	14107.53283	24.57511111
GO	22472.64333	16536.00255	26.46486298
MA	31064.59693	12798.23989	42.59968919
MG	17294.28789	18052.893	23.46270444
MS	22489.92011	14911.68474	27.00145275
MT	25999.81264	19706.82393	32.90745293
PA	34232.5148	19275.94397	39.89618557
PB	29414.03868	18142.95358	48.34535714
PE	26565.08977	18155.08663	36.07382282
PI	28017.72689	15306.26681	43.03894523
PR	17267.3833	18171.96019	23.57976791
RJ	22045.98616	15926.68032	23.94525231
RN	27760.06329	18665.44304	39.12883817
RO	27895.2428	27931.04938	46.22421053
RR	42317.5122	23895.87805	48.59108696
RS	22031.90232	19019.85853	24.94895803
SC	21540.89371	15561.30646	24.82288483
SE	30988.01493	13432.93433	40.90281159
SP	12615.85949	14948.5431	17.37095033
TO	25427.14234	16472.91606	42.05261649

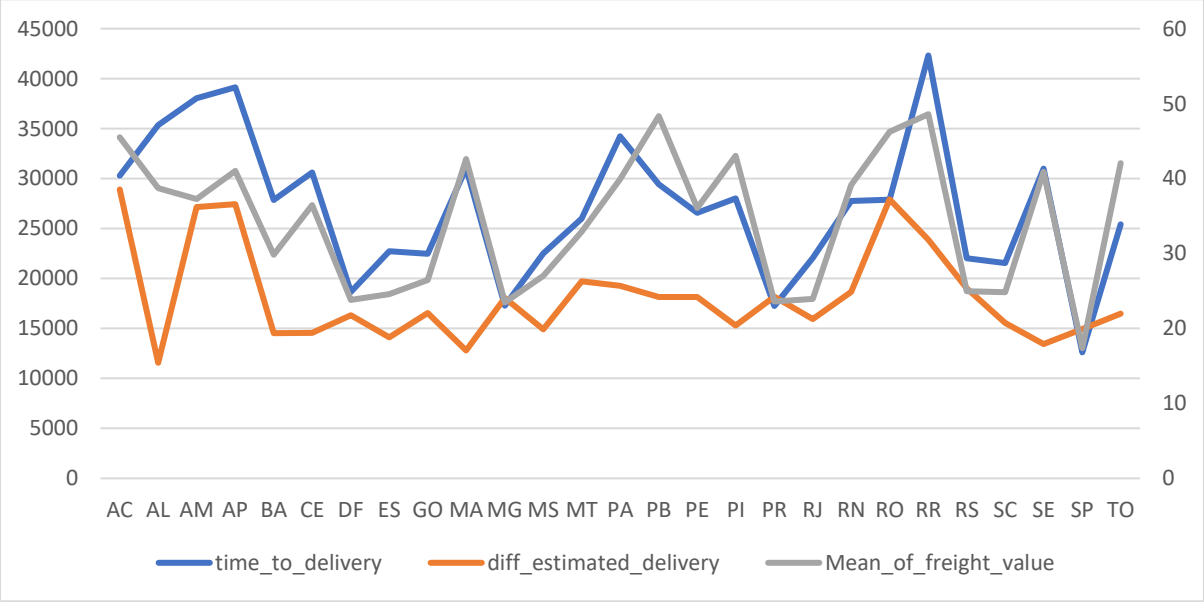


Figure 6-3 Group data by state, take mean of freight_value, time_to_delivery, diff_estimated_delivery

Insights or observations:

1. It looks like there is proportionaly behaviour between time required to delivery and average freight value.

6.3 SORT THE DATA TO GET THE FOLLOWING:

6.3.1 Top 5 states with lowest average freight value - sort in ASC limit 5

Query:

```
  /*# Group data by state, take mean of freight_value, time_to_delivery, diff_estim
ated_delivery */
WITH
  total_freight_value_per_order AS (
  SELECT
    oi.order_id,
    SUM(oi.freight_value) AS total_freight_value
  FROM
    `TARGET_BRAZIL_DATA.ORDER_ITEMS` AS oi
  GROUP BY
    oi.order_id)
SELECT
  c.customer_state,
  AVG(total_freight_value) AS Mean_of_freight_value,
  AVG(TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp)
,MINUTE)) AS time_to_delivery,
  AVG(TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_
date),MINUTE)) AS diff_estimated_delivery
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  total_freight_value_per_order AS tfvpo
ON
  tfvpo.order_id = o.order_id
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  Mean_of_freight_value ASC
LIMIT
  5
```

Result:

Table 6.4 Top 5 states with lowest average freight value - sort in ASC limit 5

customer_state	Mean_of_freight_value	time_to_delivery	diff_estimated_delivery
SP	17.37095033	12615.85949	14948.5431
MG	23.46270444	17294.28789	18052.893
PR	23.57976791	17267.3833	18171.96019
DF	23.82376471	18672.80337	16328.5625
RJ	23.94525231	22045.98616	15926.68032

Insights or observations:

1. Sao Paolo, Minas Gerais, Parana, Distrito Federal and Rio de Janeiro were the bottom 5 states according to mean freight value. That means these are the states where lowest freight value occurred.

6.3.2 Top 5 states with highest average freight value - sort in DESC limit 5

Query:

```
WITH
  total_freight_value_per_order AS (
  SELECT
    oi.order_id,
    SUM(oi.freight_value) AS total_freight_value
  FROM
    `TARGET_BRAZIL_DATA.ORDER_ITEMS` AS oi
  GROUP BY
    oi.order_id)
SELECT
  c.customer_state,
  AVG(total_freight_value) AS Mean_of_freight_value,
  AVG(TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp)
,MINUTE)) AS time_to_delivery,
  AVG(TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_
date),MINUTE)) AS diff_estimated_delivery
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  total_freight_value_per_order AS tfvpo
ON
  tfvpo.order_id = o.order_id
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  Mean_of_freight_value DESC
LIMIT
  5
```

Result:

Table 6.5 Top 5 states with highest average freight value - sort in DESC limit 5

customer_state	Mean_of_freight_value	time_to_delivery	diff_estimated_delivery
RR	48.59108696	42317.5122	23895.87805
PB	48.34535714	29414.03868	18142.95358
RO	46.22421053	27895.2428	27931.04938
AC	45.5154321	30290.9375	28910.4875
PI	43.03894523	28017.72689	15306.26681

Insights or observations:

1. Roraima, Paraiba, Rondonia, Acre and Piaui were the TOP 5 states where mean freight value was higher.
2. It is needed to decrease the freight value in these states.

6.3.3 Top 5 states with highest average time to delivery- sort in desc limit 5

Query:

```
WITH
  total_freight_value_per_order AS (
  SELECT
    oi.order_id,
    SUM(oi.freight_value) AS total_freight_value
  FROM
    `TARGET_BRAZIL_DATA.ORDER_ITEMS` AS oi
  GROUP BY
    oi.order_id)
SELECT
  c.customer_state,
  AVG(total_freight_value) AS Mean_of_freight_value,
  AVG(TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp)
,MINUTE)) AS time_to_delivery,
  AVG(TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_
date),MINUTE)) AS diff_estimated_delivery
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  total_freight_value_per_order AS tfvpo
ON
  tfvpo.order_id = o.order_id
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  time_to_delivery DESC
LIMIT
  5
```

Result:

Table 6.6 Top 5 states with highest average time to delivery- sort in desc limit 5

customer_state	Mean_of_freight_value	time_to_delivery	diff_estimated_delivery
RR	48.59108696	42317.5122	23895.87805
AP	41.00735294	39146	27445.25373
AM	37.27136054	38052.95862	27146.22759
AL	38.72163017	35342.66751	11565.96222
PA	39.89618557	34232.5148	19275.94397

Insights or observations:

1. Roraima, Amapa, Amazons, Alagoas and Para were the Top 5 states according to the average time required to deliver
2. In these state, It is necessary to increase the speed of delivery

6.3.4 Top 5 states with lowest average time to delivery - sort in asc limit 5

Query:

```
WITH total_freight_value_per_order AS (
SELECT
  oi.order_id,
  SUM(oi.freight_value) as total_freight_value
FROM
  `TARGET_BRAZIL_DATA.ORDER_ITEMS` as oi
GROUP BY oi.order_id)

SELECT
  c.customer_state,
  AVG(total_freight_value) as Mean_of_freight_value,
  AVG(TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp)
,MINUTE)) as time_to_delivery,
  AVG(TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_
date),MINUTE)) as diff_estimated_delivery
FROM `TARGET_BRAZIL_DATA.ORDERS` as o
INNER JOIN total_freight_value_per_order as tfvpo ON tfvpo.order_id = o.order_id
INNER JOIN `TARGET_BRAZIL_DATA.CUSTOMERS` as c ON c.customer_id = o.customer_id
GROUP BY c.customer_state
ORDER BY time_to_delivery ASC
LIMIT 5
```

Result:

Table 6.7 Top 5 states with lowest average time to delivery - sort in asc limit 5

customer_state	Mean_of_freight_value	time_to_delivery	diff_estimated_delivery
SP	17.37095033	12615.85949	14948.5431
PR	23.57976791	17267.3833	18171.96019
MG	23.46270444	17294.28789	18052.893
DF	23.82376471	18672.80337	16328.5625
SC	24.82288483	21540.89371	15561.30646

Insights or observations:

1. Sao Paulo, Parana, Minas Gerais, Distrito Federal and Santa Catarina were the bottom 5 states according to time required for delivery.
2. These are fairly developed according to transportation facilties.

6.3.5 Top 5 states where delivery is really not so fast compared to estimated date - sort in desc limit 5

Query:

```
WITH
  total_freight_value_per_order AS (
  SELECT
    oi.order_id,
    SUM(oi.freight_value) AS total_freight_value
  FROM
    `TARGET_BRAZIL_DATA.ORDER_ITEMS` AS oi
  GROUP BY
```

```
    oi.order_id)
SELECT
  c.customer_state,
  AVG(total_freight_value) AS Mean_of_freight_value,
  AVG(TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp)
,MINUTE)) AS time_to_delivery,
  AVG(TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_
date),MINUTE)) AS diff_estimated_delivery
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  total_freight_value_per_order AS tfvpo
ON
  tfvpo.order_id = o.order_id
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  diff_estimated_delivery DESC
LIMIT
  5
```

Result:

Table 6.8 Top 5 states where delivery is really not so fast compared to estimated date - sort in desc limit 5

customer_state	Mean_of_freight_value	time_to_delivery	diff_estimated_delivery
AC	45.5154321	30290.9375	28910.4875
RO	46.22421053	27895.2428	27931.04938
AP	41.00735294	39146	27445.25373
AM	37.27136054	38052.95862	27146.22759
RR	48.59108696	42317.5122	23895.87805

Insights or Observations:

1. Acre, Rondonia, Amapa, Amazons and Roraima are top 5 states where estimated delivery was having high offset.
2. Here Freight value should be controlled by increasing the transportation resources and bring the accuracy in delivery estimation with less freight price.

6.3.6 Top 5 states where delivery is really fast compared to estimated date - sort in asc limit 5

Query:

```
WITH

  total_freight_value_per_order AS (
  SELECT
    oi.order_id,
    SUM(oi.freight_value) AS total_freight_value
  FROM
```

```
    `TARGET_BRAZIL_DATA.ORDER_ITEMS` AS oi
  GROUP BY
    oi.order_id)
SELECT
  c.customer_state,
  AVG(total_freight_value) AS Mean_of_freight_value,
  AVG(TIMESTAMP_DIFF((o.order_delivered_customer_date),(o.order_purchase_timestamp)
,MINUTE)) AS time_to_delivery,
  AVG(TIMESTAMP_DIFF((o.order_estimated_delivery_date),(o.order_delivered_customer_
date),MINUTE)) AS diff_estimated_delivery
FROM
  `TARGET_BRAZIL_DATA.ORDERS` AS o
INNER JOIN
  total_freight_value_per_order AS tfvpo
ON
  tfvpo.order_id = o.order_id
INNER JOIN
  `TARGET_BRAZIL_DATA.CUSTOMERS` AS c
ON
  c.customer_id = o.customer_id
GROUP BY
  c.customer_state
ORDER BY
  diff_estimated_delivery ASC
LIMIT
  5
```

Result:

Table 6.9 Top 5 states where delivery is really fast compared to estimated date - sort in asc limit 5

customer_state	Mean_of_freight_value	time_to_delivery	diff_estimated_delivery
AL	38.72163017	35342.66751	11565.96222
MA	42.59968919	31064.59693	12798.23989
SE	40.90281159	30988.01493	13432.93433
ES	24.57511111	22736.11278	14107.53283
BA	29.82628946	27842.57463	14542.37193

Insights or observations:

1. Alagoas, Maranhao, Serigipe, Espirito santo and Bahia are bottom 5 states where estimated delivery is higly accurate comparatively.
2. But these states are not producing high sales. So sales promotion should be happened here.

CHAPTER 7: PAYMENT TYPE ANALYSIS

7.1 MONTH OVER MONTH COUNT OF ORDERS FOR DIFFERENT PAYMENT TYPES

7.1.1 Month over Month count of orders for different payment types

```
WITH
  month_year_order_id AS (
  SELECT
    o.order_id AS order_id,
      EXTRACT(YEAR
    FROM
      o.order_purchase_timestamp) AS purchase_year,
    EXTRACT(MONTH
    FROM
      o.order_purchase_timestamp) AS purchase_month
  FROM
    `TARGET_BRAZIL_DATA.ORDERS` AS o
  ORDER BY
    purchase_month, purchase_year )
SELECT
  COUNT(*) AS Number_of_orders,
  myoi.purchase_month,
  myoi.purchase_year,
  pa.payment_type
FROM
  `TARGET_BRAZIL_DATA.PAYMENTS` AS pa
INNER JOIN
  month_year_order_id AS myoi
ON
  myoi.order_id = pa.order_id
GROUP BY
  pa.payment_type,
  myoi.purchase_month,
  myoi.purchase_year
ORDER BY
  myoi.purchase_month,
  pa.payment_type,
  myoi.purchase_year
```

Table 7.1 Month over Month count of orders for different payment types (pivot table)

Sum of Number_of_orders	Column Labels					
Row Labels	credit_card	debit_card	not_defined	UPI	voucher	Grand Total
1	**6103**	**118**		**1715**	**477**	**8413**
2017	583	9		197	61	850
2018	5520	109		1518	416	7563
2	**6609**	**82**		**1723**	**424**	**8838**
2017	1356	13		398	119	1886
2018	5253	69		1325	305	6952
3	**7707**	**109**		**1942**	**591**	**10349**
2017	2016	31		590	200	2837
2018	5691	78		1352	391	7512
4	**7301**	**124**		**1783**	**572**	**9780**
2017	1846	27		496	202	2571
2018	5455	97		1287	370	7209
5	**8350**	**81**		**2035**	**613**	**11079**
2017	2853	30		772	289	3944
2018	5497	51		1263	324	7135
6	**7276**	**209**		**1807**	**563**	**9855**
2017	2463	27		707	239	3436
2018	4813	182		1100	324	6419
7	**7841**	**264**		**2074**	**645**	**10824**
2017	3086	22		845	364	4317
2018	4755	242		1229	281	6507
8	**8269**	**311**	**2**	**2077**	**589**	**11248**
2017	3284	34		938	294	4550
2018	4985	277	2	1139	295	6698
9	**3286**	**43**	**1**	**903**	**302**	**4535**
2016	3					3
2017	3283	43		903	287	4516
2018			1		15	16
10	**3778**	**54**		**1056**	**318**	**5206**
2016	254	2		63	23	342
2017	3524	52		993	291	4860
2018					4	4
11	**5897**	**70**		**1509**	**387**	**7863**
2017	5897	70		1509	387	7863
12	**4378**	**64**		**1160**	**294**	**5896**
2016	1					1
2017	4377	64		1160	294	5895
Grand Total	**76795**	**1529**	**3**	**19784**	**5775**	**103886**

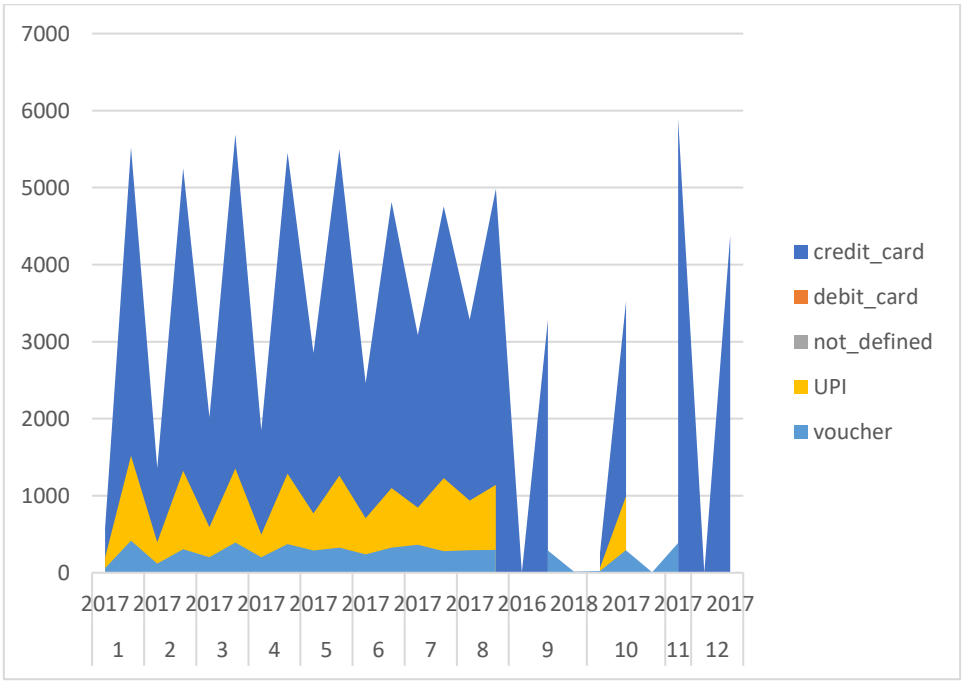


Figure 7-1 Month over Month count of orders for different payment types (area chart)

Insights or observations:

1. More number of payments happened by credit card in every month. Comparitively 2018 had more number of orders.
2. Continuity was missed from month 9 to 12

7.1.2 Month-year vs count OF orders FOR different payment types

Query:

```
WITH
  month_year_order_id AS (
  SELECT
    o.order_id AS order_id,
    FORMAT_TIMESTAMP('%Y-%m',o.order_purchase_timestamp) AS month_year
  FROM
    `TARGET_BRAZIL_DATA.ORDERS` AS o
  ORDER BY
    month_year )
SELECT
  COUNT(*) AS Number_of_orders,
  myoi.month_year,
  pa.payment_type
FROM
  `TARGET_BRAZIL_DATA.PAYMENTS` AS pa
INNER JOIN
  month_year_order_id AS myoi
ON
  myoi.order_id = pa.order_id
GROUP BY
  pa.payment_type,
  myoi.month_year
ORDER BY
  myoi.month_year,
  pa.payment_type
```

Result:

Table 7.2Month-year vs count OF orders FOR different payment types

month_year	credit_card	debit_card	not_defined	UPI	voucher
2016-09	3				
2016-10	254	2		63	23
2016-12	1				
2017-01	583	9		197	61
2017-02	1356	13		398	119
2017-03	2016	31		590	200
2017-04	1846	27		496	202
2017-05	2853	30		772	289
2017-06	2463	27		707	239
2017-07	3086	22		845	364
2017-08	3284	34		938	294
2017-09	3283	43		903	287
2017-10	3524	52		993	291
2017-11	5897	70		1509	387
2017-12	4377	64		1160	294
2018-01	5520	109		1518	416
2018-02	5253	69		1325	305
2018-03	5691	78		1352	391

2018-04	5455	97		1287	370
2018-05	5497	51		1263	324
2018-06	4813	182		1100	324
2018-07	4755	242		1229	281
2018-08	4985	277	2	1139	295
2018-09			1		15
2018-10					4

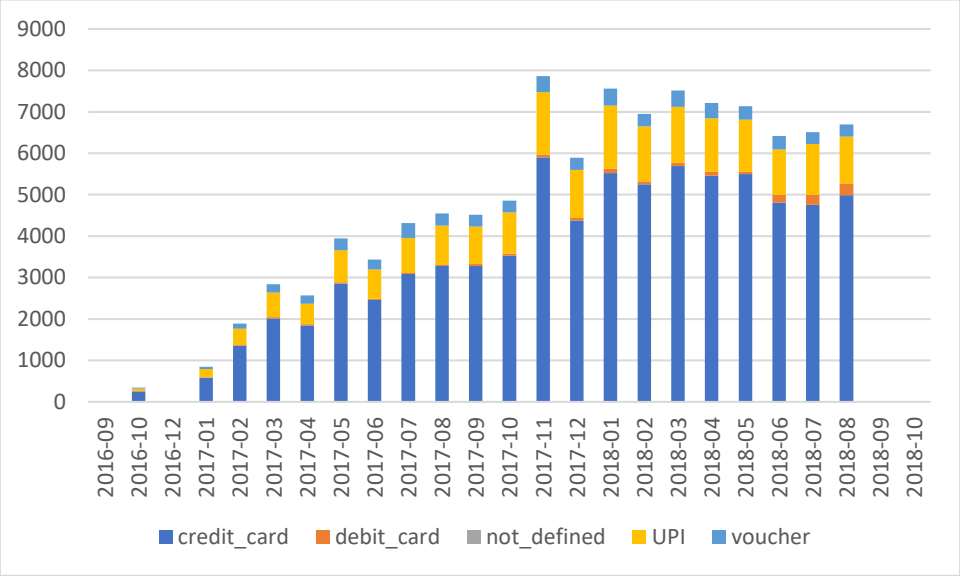


Figure 7-2 Month-year vs count OF orders FOR different payment types

Insights or observations:

3. More number of payments were happened by using Credit card and then UPI.
4. So to increase the sales rapidly, offeres should be posted for debit card and vouchers
5. Credit card and UPI customers can be attracted for higher cart amount with additional restrictions.
6. Trend on Credit card users and UPI users gradually increased from 2016 to 2018.

7.2 COUNT OF ORDERS BASED ON THE NO. OF PAYMENT INSTALL MENTS

Query:

```
  # Count OF orders based ON the no. OF payment installments
SELECT
  pa.payment_installments,
  COUNT(*) AS Number_of_orders
FROM
  `TARGET_BRAZIL_DATA.PAYMENTS` AS pa
GROUP BY
  pa.payment_installments
ORDER BY
  pa.payment_installments;
```

Result:

Table 7.3 Count OF orders based ON the no. OF payment installments

payment_installments	Number_of_orders
0	2
1	52546
2	12413
3	10461
4	7098
5	5239
6	3920
7	1626
8	4268
9	644
10	5328
11	23
12	133
13	16
14	15
15	74
16	5
17	8
18	27
20	17
21	3
22	1
23	1
24	18

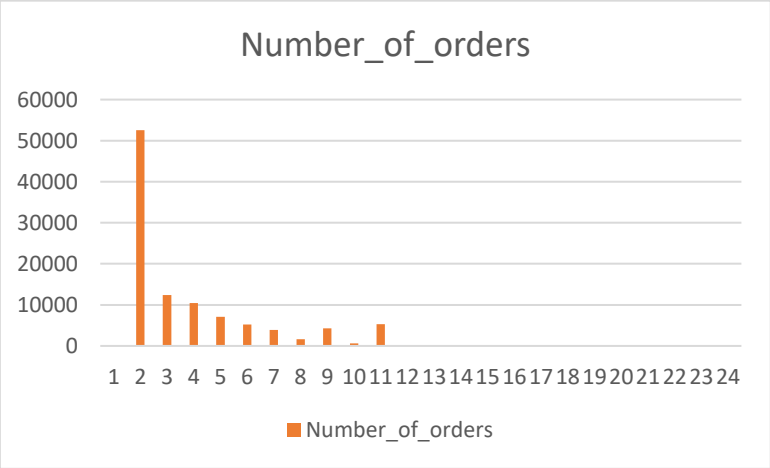


Figure 7-3 Count OF orders based ON the no. OF payment installments

Insights and Observations:

1. Number of orders were dominatingly more with two installments.

2. Very negligible number of orders were made with installments 1 and greater than 11.
3. It is may be due to high interest beyond 11 months.

CHAPTER 8: ACTIONABLE INSIGHTS

1. Time period of the given data is 2 Years 42 days only.
2. Total 4119 cities and 27 states customers involved in the sales of Target Store.
3. Last three months of 2016 data looks like the worst period of time for sales of Target store. Huge effected by Prolonged recession or inflation or political reforms.
4. Gradual growth in number of orders and sales observed in 2017.
5. Sales and number of orders growth maintained average behaviour in 2018.
6. Month wise – November 2017 had the highest sales.
7. Cumulatively September month has lowest sales (overall).
8. Best three phases of the day in Descending order are Afternoon → Morning → Evening
9. Sao Paolo, Rio De Janeiro and Minas Gerais dominates other states in number of orders and number of customers wise too.
10. From 2017 to 2018, 1.3 times growth was observed.
11. Estimation on delivery days should be improved. As actual and estimated delivery days difference in range of 12 days on average.
12. Sao Paolo was having lowest average freight value and also lowest average delivery time.
13. Roraima was having highest average freight value and also highest average delivery time.
14. Acre was having high offset in estimated delivery.
15. Alagoas was having low offset in estimated delivery.
16. Credit Card payments are in first position and UPI payments are in second postion.
17. More number of customers paid by using 2 installments only.

CHAPTER 9: RECOMMENDATIONS

1. Sales promotion campaign should be held in states like Roraima, Acre etc.,
2. Voucher offers should be provided to increase the sales through vouchers
3. Less EMI or No cost EMI should be provided to increase the number of installments.
4. Estimation strategy should be modified by using machine learning techniques like time series forecasting using ARIMA such that average offset in estimation should be below 3-4 days
5. Sales are highly happening in April to August time period because of CARNIVAL, Folklore fest etc., So inventory should be optimized well in this period. Big day sales can attract high number of sales.
6. Delivery time directly related to Freight value. So, Infrastructure investment is required in backlog states like Roraima, Acre etc.,
7. Sao Paolo, Rio De Janeiro and Minas Gerais are the three important states where Target should never loose grip on these three states.