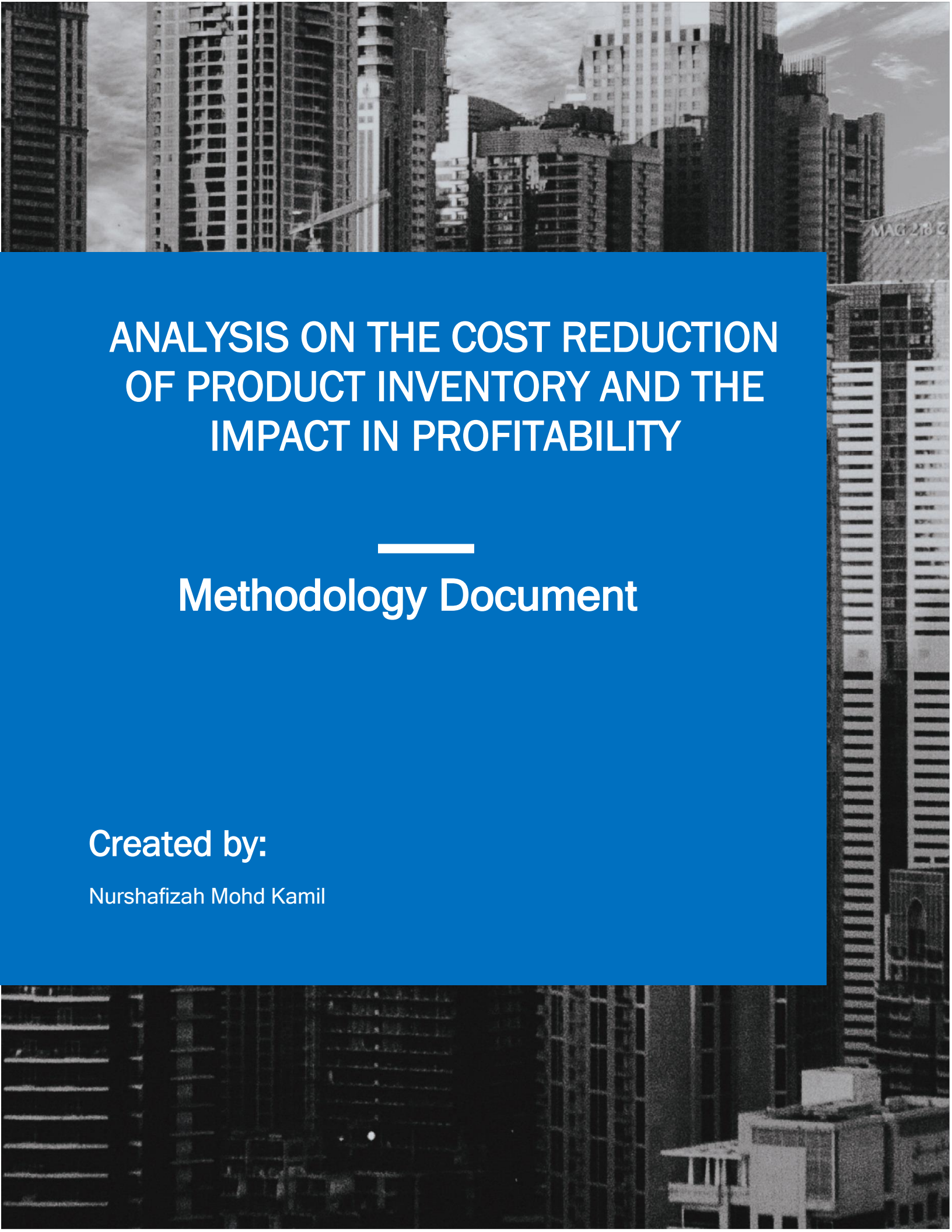


ANALYSIS ON THE COST REDUCTION OF PRODUCT INVENTORY AND THE IMPACT IN PROFITABILITY

Methodology Document

Created by:

Nurshafizah Mohd Kamil

ANALYSIS ON THE COST REDUCTION OF PRODUCT INVENTORY BY USING MARKET BASKET AND PARETO ANALYSIS

(METHODOLOGY DOCUMENTS)

PROBLEM STATEMENTS:

OList is an e-commerce company that has faced some losses recently and they want to manage their inventory very well so as to reduce any unnecessary costs.

With the emergence of the e-commerce in the industry, it is very important for them to plan their inventory accordingly without storing tons of products.

OBJECTIVE:

- ✓ To gather some important insights for the company to reduce their inventory cost by focusing on the Market Basket Analysis
- ✓ To analyse and improve our understanding on customer preferences and purchase behaviour.
- ✓ To provide recommendations to the company on future inventory level.

olist
empowering commerce

ORDERS

DATA ANALYSIS/ DATA WRANGLING:

A. Analysing the attributes

The process of analyzing the data had been carried out in Jupyter Notebook.

```
# Identifying the rows and columns available

orders.shape
```

```
(99441, 7)
```

```
# Check the column-wise info and the data type of the columns

orders.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 7 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  datetime64[ns]
 4   order_approved_at              99281 non-null  datetime64[ns]
 5   order_delivered_timestamp      96476 non-null  datetime64[ns]
 6   order_estimated_delivery_date  99441 non-null  datetime64[ns]
dtypes: datetime64[ns](4), object(3)
memory usage: 5.3+ MB
```

Orders sheet consists of 99441 rows and 7columns.

<u>Null Values</u>

Out of 7 columns in the Orders sheet, order_delivered_timestamp and order_approved_at consists of 2965 (2.98%) and 160 (0.16%) missing values which will be imputed in the next step.

```
# Checking Null values of the Orders sheet

orders.isnull().sum()
```

```
order_id                           0
customer_id                        0
order_status                       0
order_purchase_timestamp           0
order_approved_at                160
order_delivered_timestamp       2965
order_estimated_delivery_date      0
dtype: int64
```

olist
empowering commerce

Selecting only the 'Delivered' order status

```
# Analysing the order_status

orders.order_status.value_counts()
```

```
delivered      96478
shipped         1107
canceled         625
unavailable      609
invoiced         314
processing       301
created            5
approved           2
Name: order_status, dtype: int64
```

We will be selecting only the 'Delivered' status for our analysis, since that it consists almost 97% if the entire order status.

```
# Analysis with the order_status 'delivered' since that 97% of our data consists of only Delivered products

orders = orders[orders.order_status == "delivered"]
orders.shape
```

```
(96478, 7)
```

B. Handling the Missing Values

```
# Finding the median values of 'order_approved_at - order_purchase_timestamp'
# With an assumptions of there will be time differences when customer purchasing the order and seller accepting the order

order_time_differences = (orders['order_approved_at'] - orders['order_purchase_timestamp']).median()

# Replacing the null values by adding the order_time_differences values with the order_approved_at column

orders.order_approved_at.fillna((orders.order_purchase_timestamp + order_time_differences), inplace=True)
```

The imputation of these null values based on the median values of the differences, are made because of the assumptions that there are time differences between the time when the customer placed the order and the order get approved.

As well as times differences when the order was approved and delivered to the customers. It is impossible to ignore the differences and replace with any other values besides median.

```
# Finding the median values of 'order_delivered_timestamp - order_approved_at'
# With an assumptions of there will be time differences when seller approved the order and the delivered times

order_delivered_differences = (orders['order_delivered_timestamp'] - orders['order_approved_at']).median()

# Replacing the null values by adding the order_time_differences values with the order_approved_at column

orders['order_delivered_timestamp'].fillna(orders['order_approved_at'] + order_delivered_differences,inplace=True)
```

olist
empowering commerce

```
# Checking the null values in order df

orders.isnull().sum()
```

```
order_id                          0
customer_id                       0
order_status                      0
order_purchase_timestamp          0
order_approved_at                 0
order_delivered_timestamp         0
order_estimated_delivery_date     0
dtype: int64
```

```
orders.duplicated().sum()
```

```
0
```

There are **NO null values and NO duplicates values** exists in the orders column, thus let's process with the next data.

ORDER ITEM

A. Analysing the Attributes

```
# Identifying the total number rows and columns

order_item.shape
```

```
(112650, 6)
```

Order Item column consists of 112650 rows and 6 columns.

```
# Checking the null values in order_item

order_item.isnull().sum()
```

```
order_id           0
order_item_id      0
product_id         0
seller_id          0
price              0
shipping_charges   0
dtype: int64
```

There are **No Missing Values, and No Duplicates Values** in the dataset, thus let's proceed further.

```
# Checking Duplicates value in order_items

order_item.duplicated().sum()
```

```
0
```

```
order_item.order_id.isnull().sum()
```

```
0
```

olist
empowering commerce

B. Analysing the Numerical features in the datasets

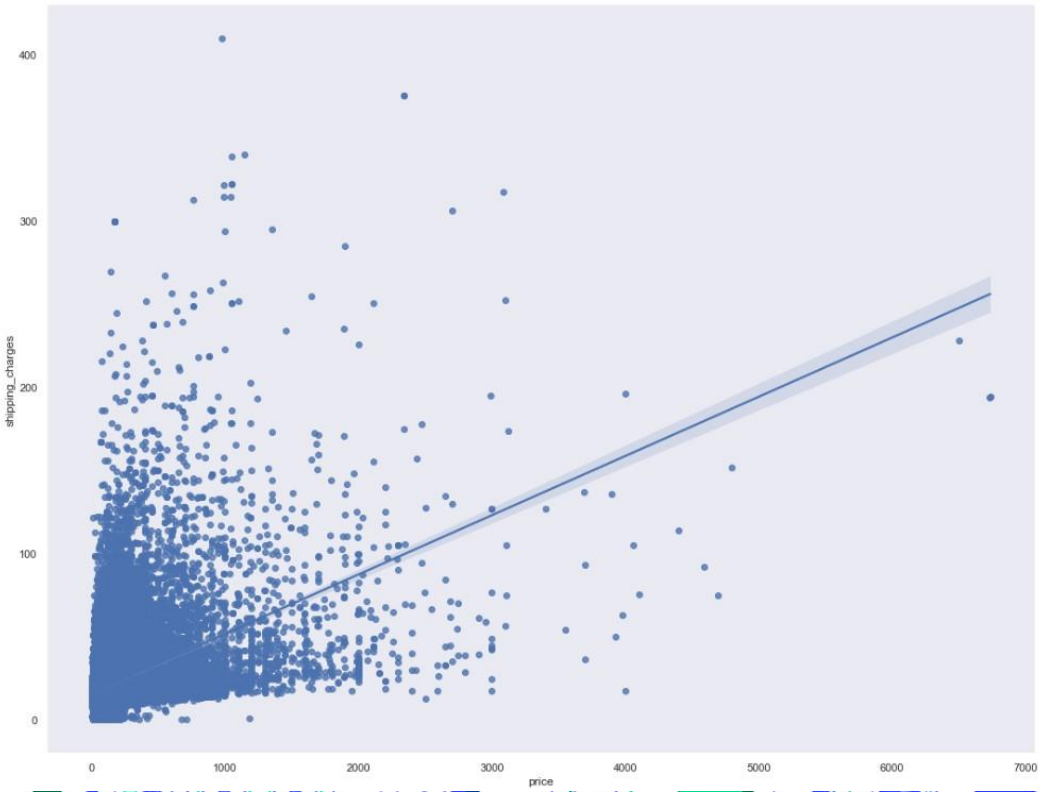
```
# Checking the numerical columns in order_item

order_item.describe().T
```

	count	mean	std	min	25%	50%	75%	max
order_item_id	112650.0	1.197834	0.705124	1.00	1.00	1.00	1.00	21.00
price	112650.0	120.653739	183.633928	0.85	39.90	74.99	134.90	6735.00
shipping_charges	112650.0	19.990320	15.806405	0.00	13.08	16.26	21.15	409.68

From the above data, we can infer that:

1. The mean for the price is 120.65, with the minimum value of 0.85 and maximum values at 6735.00. Obviously, there is an outliers present in the column, which will be treated in the next step.
2. For the shipping charges column, the mean value is at 19.99, minimum value with 0 charges and the maximum values at 409.68.

```
# Analysing the correlation between price and shipping charges column

plt.figure(figsize=(18,14))
sns.set(style='dark')
ax = sns.regplot(x="price", y="shipping_charges", data=order_item, color='b')
plt.show()
```

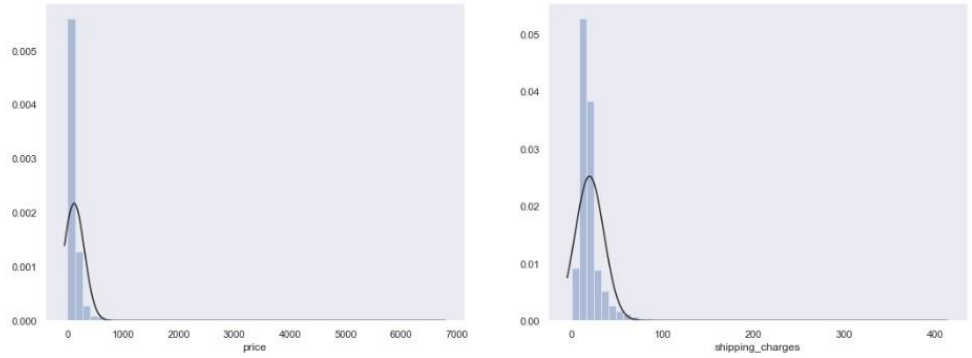


Analyzing the correlation between 'price' and 'shipping charges'

From the regplot, it clearly indicates that price and shipping charges are positively correlated

olist
empowering commerce

```
# Analysing the distribution plot for 'price' and 'shipping charges'

from scipy.stats import norm

plt.figure(figsize=(18,14))
sns.set_style("dark")
plt.subplot(2,2,1)
sns.distplot(order_item.price,fit=norm, kde=False)
plt.subplot(2,2,2)
sns.distplot(order_item.shipping_charges, fit=norm, kde=False)

plt.show()
```



The distplot for price and shipping charges are very important to determine the skewness of the data. Both of the variables are rightly skewed.

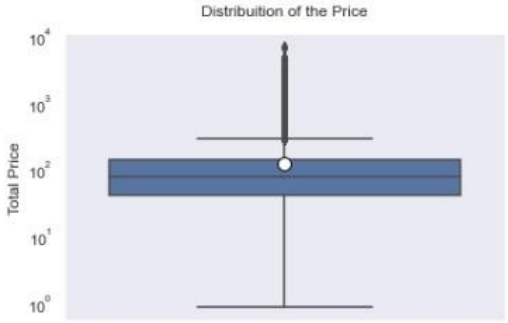
C. Handling the outliers

i. Price

Boxplot has shown that there are an outliers present in the Price column, an equal distribution between the 1st quartile and the 3rd quartile. Mean value lies at 120.65.

We have decided to impute the value > 3000 to median value.

```
# Analysing the outliers for the Price variables

sns.boxplot(data= order_item, y='price', showmeans=True,meanprops={"marker":"o",
                       "markerfacecolor":"white",
                       "markeredgecolor":"black",
                      "markersize":"10"})
plt.yscale('log')
plt.title('Distribuition of the Price')
plt.ylabel('Total Price')
plt.show()
print('Mean value :', order_item.price.mean())
```



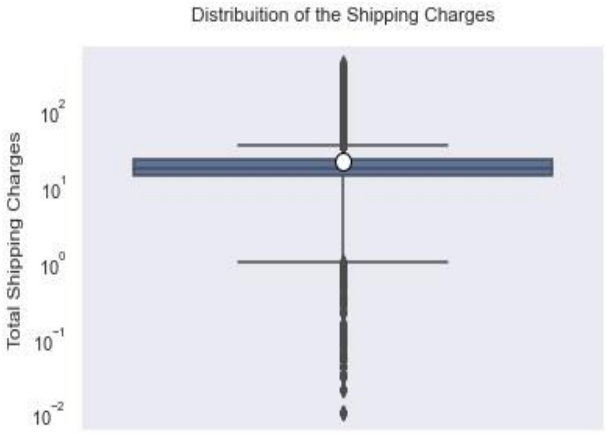

```
Mean value : 120.65373901477311
```

olist
empowering commerce

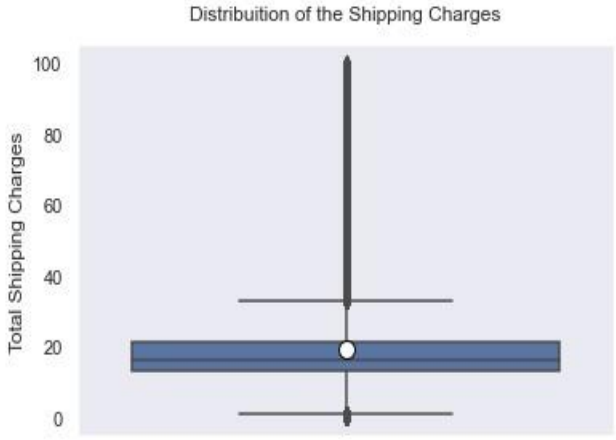
ii. Shipping Charges

We are imputing the shipping charges for the value > 100.

Before Imputation



After Imputation




```
# Analysing the distribution for the Shipping charges variables

sns.boxplot(data= order_item, y='shipping_charges', showmeans=True,meanprops={"marker":"o",
                       "markerfacecolor":"white",
                       "markeredgecolor":"black",
                      "markersize":"10"})
plt.yscale('log')
plt.title('Distribuition of the Shipping Charges')
plt.ylabel('Total Shipping Charges')
plt.show()
print('Mean value :', order_item.shipping_charges.mean())
```

olist
empowering commerce

CUSTOMERS

A. Analysing the Attributes

```
# Checking the rows and column for the Customers data

customers.shape
```

```
(99441, 4)
```

Customer's dataset consists of 99441 rows with 4 columns.

```
# Checking the Null values in the column

customers.isnull().sum()
```

```
customer_id                 0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
```

There are NO NULL VALUES presents in this dataset.

```
# Checking the Duplicated values in the column

customers.duplicated().sum()
```

```
3089
```

```
# Dropping the duplicates value in customer_id by keeping only distinct customer_id

customers.drop_duplicates(subset='customer_id', keep = 'first', inplace = True)
```

However, there are total of 3089 duplicates values present in this dataset. Thus, we dropped this duplicates values for our further analysis.

```
customers.duplicated().sum()
```

```
0
```

```
customers.isnull().sum()
```

```
customer_id                 0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
```

Final Data with NO missing and duplicates values.

olist
empowering commerce

PAYMENTS

A. Analysing the Attributes

```
# Checking the number of rows and columns in the dataframe

payments.shape
```

```
(103886, 5)
```

```
# Checking the Null values in the data

payments.isnull().sum()
```

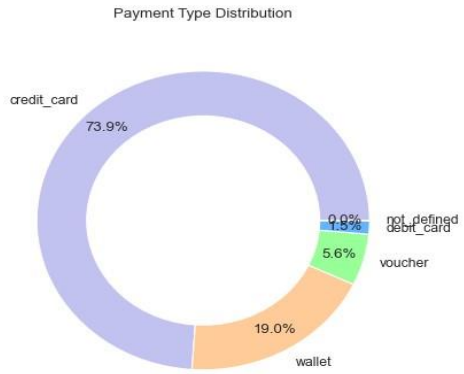
```
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64
```

```
# Proceed with the duplicated values

payments.duplicated().sum()
```

```
0
```

The Payments datasets consists of 103886 rows and 5 columns. There are No Missing values and No duplicates values presents.

B. Visualizing the Payment Type




```
# Visualising the distribution for the Payment Type columns

from matplotlib.pyplot import figure
figure(figsize=(8, 6), dpi=80)

#sns.color_palette("tab10")
temp_series = payments['payment_type'].value_counts()
labels = (np.array(temp_series.index))
#explode = (0.05,0.1,0.1,0.1)
colors = ['#c2c2f0', '#ffcc99', '#99ff99', '#66b3ff']

plt.pie(temp_series, labels = labels,  autopct="%1.1f%%", colors=colors, pctdistance=0.85)
plt.title('Payment Type Distribution')

centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)


plt.show()
```

From the donut chart, we can infer that:

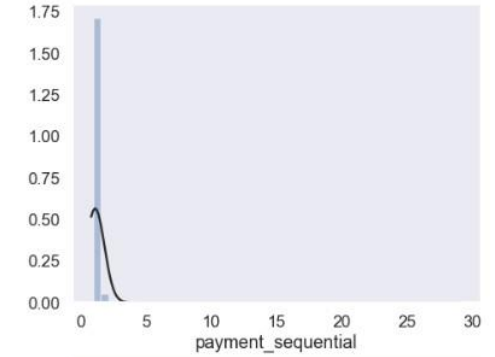
- Most of the orders done by customers are through the Credit Card which is 73.9%. Followed closely by Wallet and Voucher with 19% and 5.56%.
- There are some medium of payment which is not defined, for this we will drop the payment type as it will not affect our analysis later on.

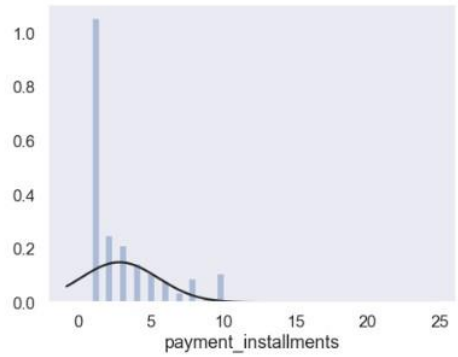
```
#Dropping the rows with not_defined which won't have a significant impact on the dataset.

i = payments[payments['payment_type']=='not_defined'].index
payments.drop(i, axis=0, inplace=True)
```

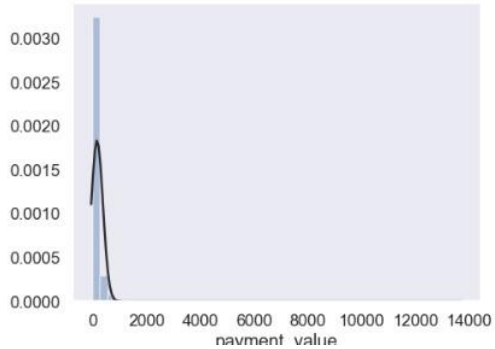
olist
empowering commerce

C. Analysing the Numerical Features in the dataset

```
# Analysing the distribution plot for payment_sequential, payment_installment and payment_values

from scipy.stats import norm

plt.figure(figsize=(16,12))
sns.set_style("dark")
plt.subplot(2,2,1)
sns.distplot(payments.payment_sequential,fit=norm, kde=False)
plt.subplot(2,2,2)
sns.distplot(payments.payment_installments, fit=norm, kde=False)
plt.subplot(2,2,3)
sns.distplot(payments.payment_value, fit=norm, kde=False)

plt.show()
```







All of the 3 graphs showed that the data are rightly skewed and there are outlier present in the dataset.

Final data for Payments do not have any missing values as well as duplicates values.

```
# Checking the final null values for payments data

payments.isnull().sum()
```

```
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64
```

```
# Checking the final duplicates value if its exist in the payments data

payments.duplicated().sum()
```

```
0
```

olist
empowering commerce

PRODUCTS

A. Analysing the Attributes

```
# Identifying the rows and columns available

products.shape
```

```
(32951, 6)
```

There are 32951 rows in the dataset and 6 columns.

```
# Checking Null values of the Products data

products.isnull().sum().sort_values(ascending=False)
```

```
product_category_name    170
product_weight_g           2
product_length_cm          2
product_height_cm          2
product_width_cm           2
product_id                 0
dtype: int64
```

Out of 6 columns, 5 columns having missing values.

B. Handling the Missing Values

```
# Finding the mode value for the product_category_name

products.product_category_name.mode()[0]
```

```
'toys'
```

```
# Imputing the rest of the null values in the column with the common value (mode)

products.product_category_name.fillna(products.product_category_name.mode()[0], inplace=True)
```

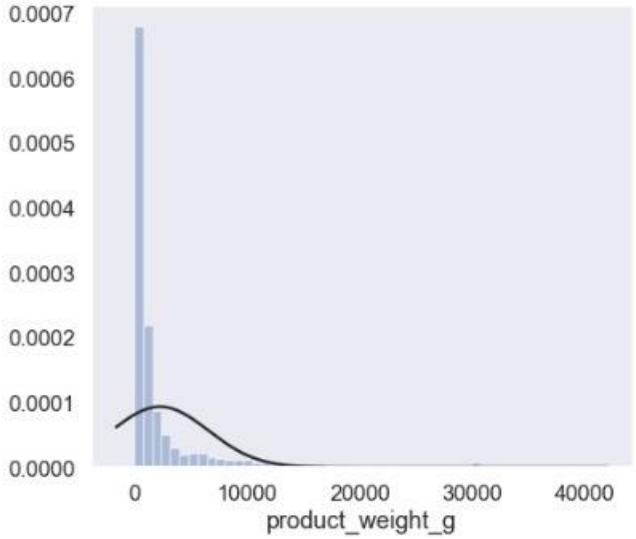
For the missing values in the product category name, we are imputing the missing values with the common value, which is mode. In this case, the missing values are replaced with 'Toys'.

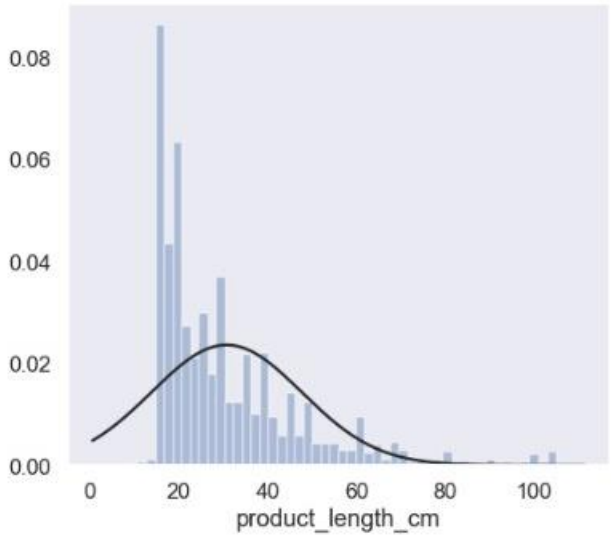
```
# Imputing the rest of the null values in the column with the median due to the skewness of the data

products.product_weight_g.fillna(products.product_weight_g.median(), inplace=True)
products.product_length_cm.fillna(products.product_length_cm.median(), inplace=True)
products.product_height_cm.fillna(products.product_height_cm.median(), inplace=True)
products.product_width_cm.fillna(products.product_width_cm.median(), inplace=True)
```

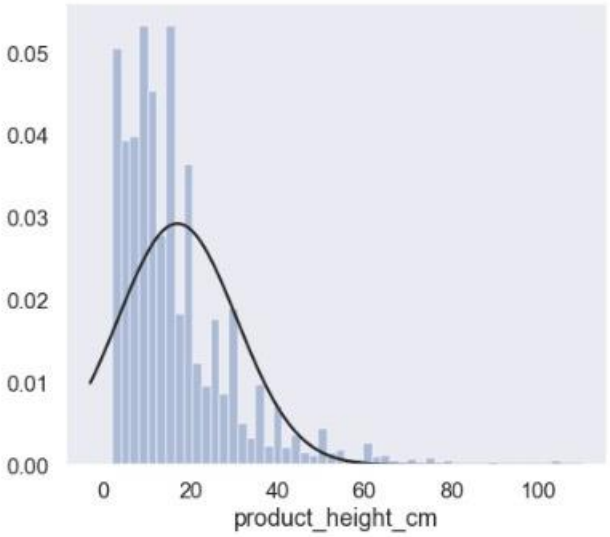
For the remaining missing values in the numerical columns, we are imputing those values by replacing it with the median values.

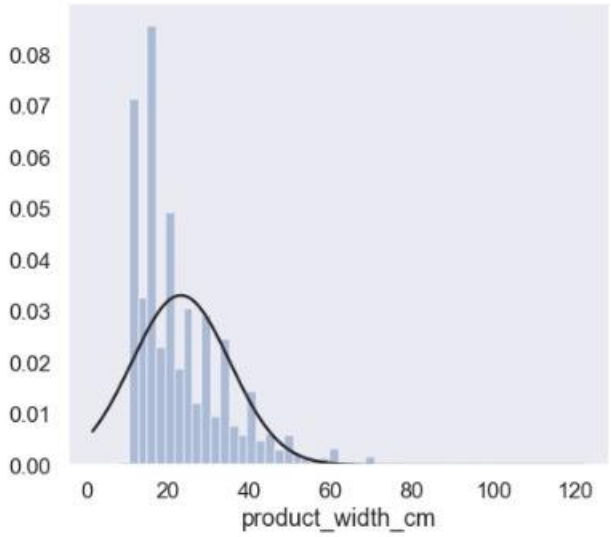
olist
empowering commerce

C. Analysing the Numerical Features



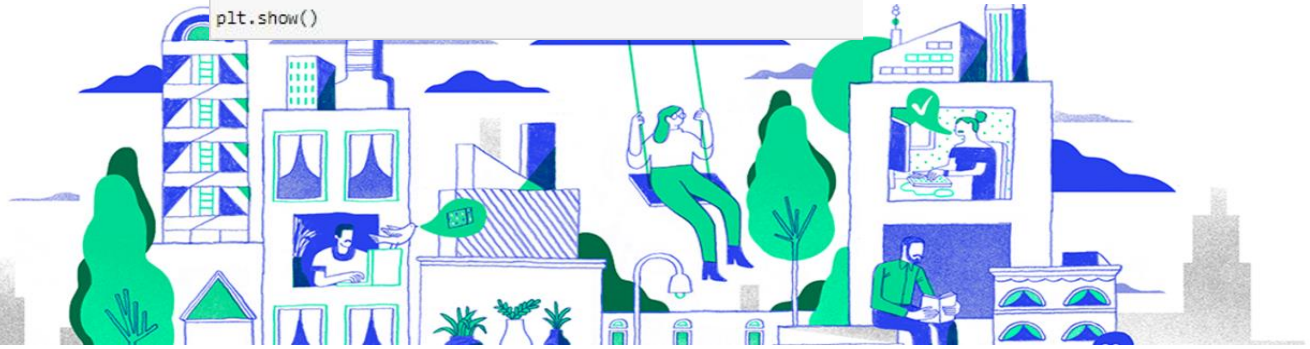






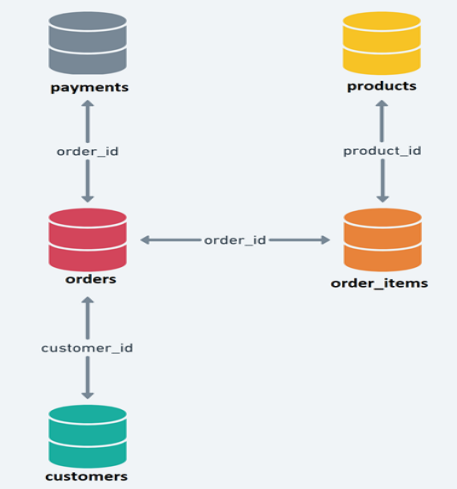

```
# Plotting the distribution of the variables by using distplot

plt.figure(figsize=(16,14))
sns.set_style('dark')
plt.subplot(2,2,1)
sns.distplot(products.product_weight_g,fit=norm, kde=False)
plt.subplot(2,2,2)
sns.distplot(products.product_length_cm,fit=norm, kde=False)
plt.subplot(2,2,3)
sns.distplot(products.product_height_cm,fit=norm, kde=False)
plt.subplot(2,2,4)
sns.distplot(products.product_width_cm,fit=norm, kde=False)

plt.show()
```

All of the numerical columns are rightly skewed.



olist
empowering commerce

MERGING THE DATASETS



This Retail dataset consists of 5 main sheets which includes orders, order_item, products, payments and customers.

All of the sheets are connected with the common value such as Order ID, product ID and customer ID.

Merging of the datasets are based on the following codes:

```
# Merging the column of orders and order_item

orditem_ord= pd.merge(orders, order_item, how='inner', on='order_id')
orditem_ord.head()
```

```
# Merging the column of orders and customers

ord_cust= pd.merge(orditem_ord,customers, how='inner', on='customer_id')
ord_cust.head()
```

```
# Merging the column of ord_cust and payments

ordcust_pay= pd.merge(ord_cust, payments, how='inner', on='order_id')
ordcust_pay.head()
```

```
# Merging the column of order_item and products

retail_final= pd.merge(ordcust_pay, products, how='inner', on='product_id')
retail_final.head()
```

The final datasets after merging consists of 115035 rows and 24 columns. There are NO MISSING VALUES and NO DUPLICATES VALUES present in the dataset.

```
# Identifying the rows and columns in the dataset

retail_final.shape
```

```
(115035, 24)
```

```
# Checking the duplicates values

retail_final.duplicated().sum()
```

```
0
```

```
# Checking if the null values exists in the dataset

retail_final.isnull().sum()
```

```
order_id                        0
customer_id                     0
order_status                    0
order_purchase_timestamp        0
order_approved_at               0
order_delivered_timestamp       0
order_estimated_delivery_date   0
order_item_id                   0
product_id                      0
seller_id                       0
price                           0
shipping_charges                0
customer_zip_code_prefix        0
customer_city                   0
customer_state                  0
payment_sequential              0
payment_type                    0
payment_installments            0
payment_value                   0
product_category_name           0
product_weight_g                0
product_length_cm               0
product_height_cm               0
product_width_cm                0
dtype: int64
```

MARKET BASKET ANALYSIS

We have created a separate dataframe for our Market Basket Analysis as follows:

```
# Using selected columns for the analysis

mb = retail_final[['product_category_name','order_id', 'order_item_id']]

# Checking the Top 5 rows
mb.head()
```

	product_category_name	order_id	order_item_id
0	housewares	e481f51cbdc54678b7cc49136f2d6af7	1
1	housewares	e481f51cbdc54678b7cc49136f2d6af7	1
2	housewares	e481f51cbdc54678b7cc49136f2d6af7	1
3	housewares	128e10d95713541c87cd1a2e48201934	1
4	housewares	0e7e841ddf8f8f2de2bad69267ecfbcf	1

```
# Finalizing the rows and columns

mb.shape
```

```
(115035, 3)
```

i. Only the selected features such as order id, order item id and product category name are selected for this analysis.

ii. The importance of this analysis is to determine on which items frequently bought together, hence will help Olist to come out with a future planning for their inventory cost.

iii. Upon completing the Market Basket dataset, it has been finalized that this dataset consists of 115035 rows and 3 columns.

olist
empowering commerce

EXPORTING THE DATASET TO EXCEL

Once the data is cleaned, we have exported the dataset to excel (below code) before proceed further with our visualization in Tableau.

The insights we have gathered from our Tableau visualization as well as the recommendations, which can be implemented, have been compiled in a file called Executive Summary.

Exporting the cleaned dataset

```
import xlsxwriter

# Exporting the cleaned retail_datasets
writer = pd.ExcelWriter('Market_Retail_Analysis(Final) - Nurshafizah Mohd Kamil.xlsx', engine='xlsxwriter')
orders.to_excel(writer, sheet_name='orders', index = False, na_rep=0)
order_item.to_excel(writer, sheet_name='order_items', index = False, na_rep=0)

customers.to_excel(writer, sheet_name='customers', index = False, na_rep=0)
payments.to_excel(writer, sheet_name='payments', index = False, na_rep=0)
products.to_excel(writer, sheet_name='products', index = False, na_rep=0)
writer.save()
print("Cleaned Data Sets exported successfully.")
```

```
Cleaned Data Sets exported successfully.
```

Exporting the Market Basket dataset

```
import xlsxwriter

# Exporting the cleaned retail_datasets
writer = pd.ExcelWriter('Market Basket - Nurshafizah Mohd Kamil.xlsx', engine='xlsxwriter')
mb.to_excel(writer, sheet_name='mb', index = False)

writer.save()
print("Cleaned Data Sets exported successfully.")
```

```
Cleaned Data Sets exported successfully.
```

olist
empowering commerce