Rakamin Academy X KALBE Nutritionals

ARIMA TIME SERIES SALES FORECASTING AND CUSTOMER SEGMENTATION USING KMEANS CLUSTERING

Data Scientist Intern

Presented by

Cyntia Angelica

ABOUT ME

CYNTIA ANGELICA

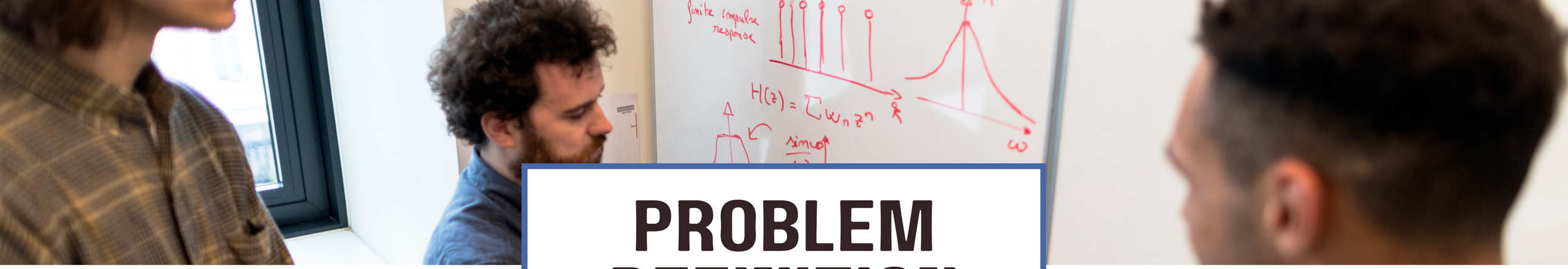
Computer Science and Statistics Student
@BINUS University

Highly motivated and enthusiastic data science enthusiast with a passion for solving complex problems using data-driven approaches. Seeking an opportunity to further enhance skills in data science and machine learning while contributing to meaningful projects in a dynamic and collaborative environment.

Proficient in Python, R, SQL, Java, and C equipped with certifications from obtained from several courses and have also worked as a data science tutor for high school student. Great in teamworking and time management with experience of managing public relation division in campus organization.

https://www.linkedin.com/in/cyntia-angelica-6b5439217/

CASE STUDY



PROBLEM DEFINITION

In this rapidly evolving digital age, data has a central role in driving smart and strategic business decisions. One area where data science has proven great value is in predicting sales and identifying potential customer segments. With the ability to glean deep insights from multiple data sources, companies can take evidence-based steps to optimize operations, increase efficiency, and improve user experience.

This project aims to combine the power of data science in predicting future sales trends and implementing better customer segmentation analysis. By utilizing the latest data analysis techniques, advanced prediction models, and innovative segmentation algorithms, this project will assist companies in taking strategic measures based on deep insights into customer behavior and market trends.

DATA COLLECTION

Primary Key

Foreign Key

ProductID	Unique number for every product
Product Name	Product's name
Price	Product's price

'product' table
Rows: 10

CustomerID	Unique number for every customer
Age	Customer's age
Gender	Customer's gender. 0 represents female, 1 represents male.
Marital Status	Marital status of customer (Married/single)
Income	Customer's income in million Rupiah

'customer' table
Rows: 447

StoreID	Unique number for every store
StoreName	StoreName
GroupStore	Store's group
Type	Modern Trade or General Trade
Latitude	Latitude code
Longtitude	Longtitude code

store table
Rows: 14

TransactionID	Unique code for every transaction
CustomerID	Unique number for every customer
Date	Date when the transaction took place
ProductID	Unique number for every product
Price	Product's price per item
Qty	Number of items purchased by the customer
TotalAmount	Price * Qty
StoreID	Store's unique number

transaction table
Rows: 5020

EDA WITH SQL

WHAT IS THE AVERAGE AGE OF CUSTOMERS BASED ON THEIR MARITAL STATUS?

Query

```sql
SELECT `Marital Status`, AVG(Age) AS 'Average Age'
FROM customer
GROUP BY `Marital Status`;
```

Marital Status	Average Age
	31.333333333333332
Married	43.03823529411765
Single	29.384615384615383

Average age of customers who don’t have marital status = 31.33 y. o.

Married customers' average age = 43.04 y. o.

Single customers' average age = 29.38 y. o.

WHAT IS THE AVERAGE AGE OF CUSTOMERS BASED ON THEIR GENDERS?

Query

```sql
SELECT `Gender`, AVG(Age) AS 'Average Age'
FROM customer
GROUP BY `Gender`;
```

Gender	Average Age
0	40.32644628099174
1	39.141463414634146

Average age of female customers = 40.32 y. o.

Average age of male customers = 39.14 y. o.

DETERMINE THE NAME OF THE STORE WITH MOST QUANTITIES SOLD!

Query

```sql
SELECT s.StoreName, SUM(t.Qty) AS TotalQuantity
FROM transaction t
JOIN store s ON t.StoreID = s.StoreID
GROUP BY s.StoreName
ORDER BY TotalQuantity DESC
LIMIT 1;
```

StoreName	TotalQuantity
Lingga	2777

Store that sold the most items is Lingga with 2777 items sold.

DETERMINE THE BEST SELLING ITEM BASED ON THE GENERATED REVENUE!

Query

```sql
SELECT p.`Product Name`, SUM(t.TotalAmount) AS TotalAmount
FROM transaction t
JOIN product p ON t.ProductID = p.ProductID
GROUP BY p.`Product Name`
ORDER BY TotalAmount DESC
LIMIT 1;
```

Product Name	TotalAmount
Cheese Stick	27615000

The best-selling item is ‘Cheese Stick’ which generates revenue of Rp27,615,000.

DATA VISUALIZATION

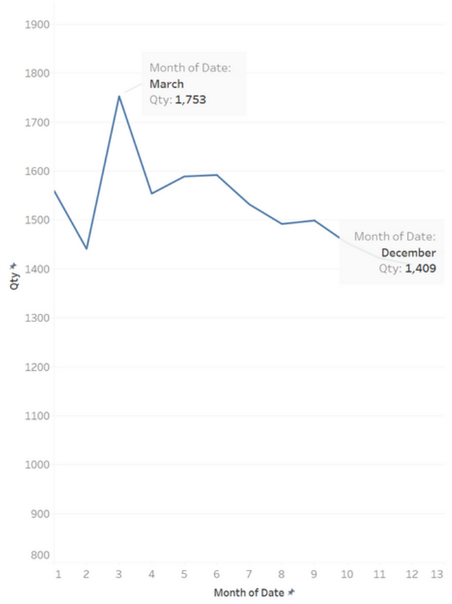
with Tableau

TABLEAU DASHBOARD

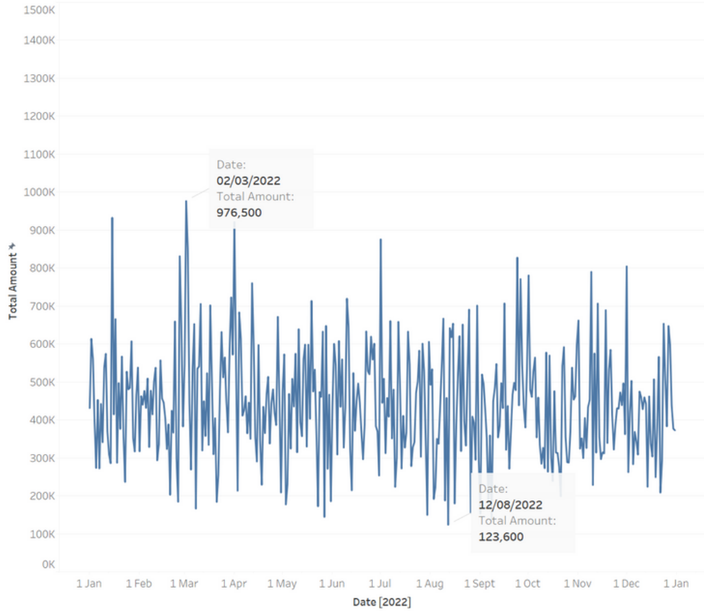
Overview of 2022 Performance

Here we analyze the sales performance based on number of items sold and revenue generated.

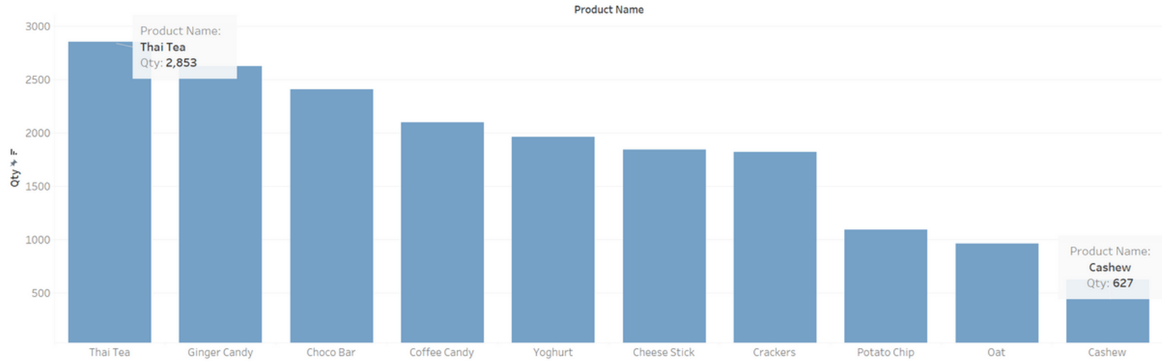
Monthly Sales Based on Quantities Sold (2022)



Daily Revenue (2022)



Quantities Sold per Product (2022)



Revenue Generated by Each Store (2022)

TABLEAU WORKSHEET

Visualization of Monthly Sales Based on Items Sold in 2022

From the visualization, we can see that the quantity of goods sold in each month fluctuates but tends to decrease. The quantity of items sold peaked in March 2022 with 1753 items sold. After March 2022, the number of items sold decreased until December 2022, where December 2022 was also the month with the lowest number of sales with 1409 items sold.

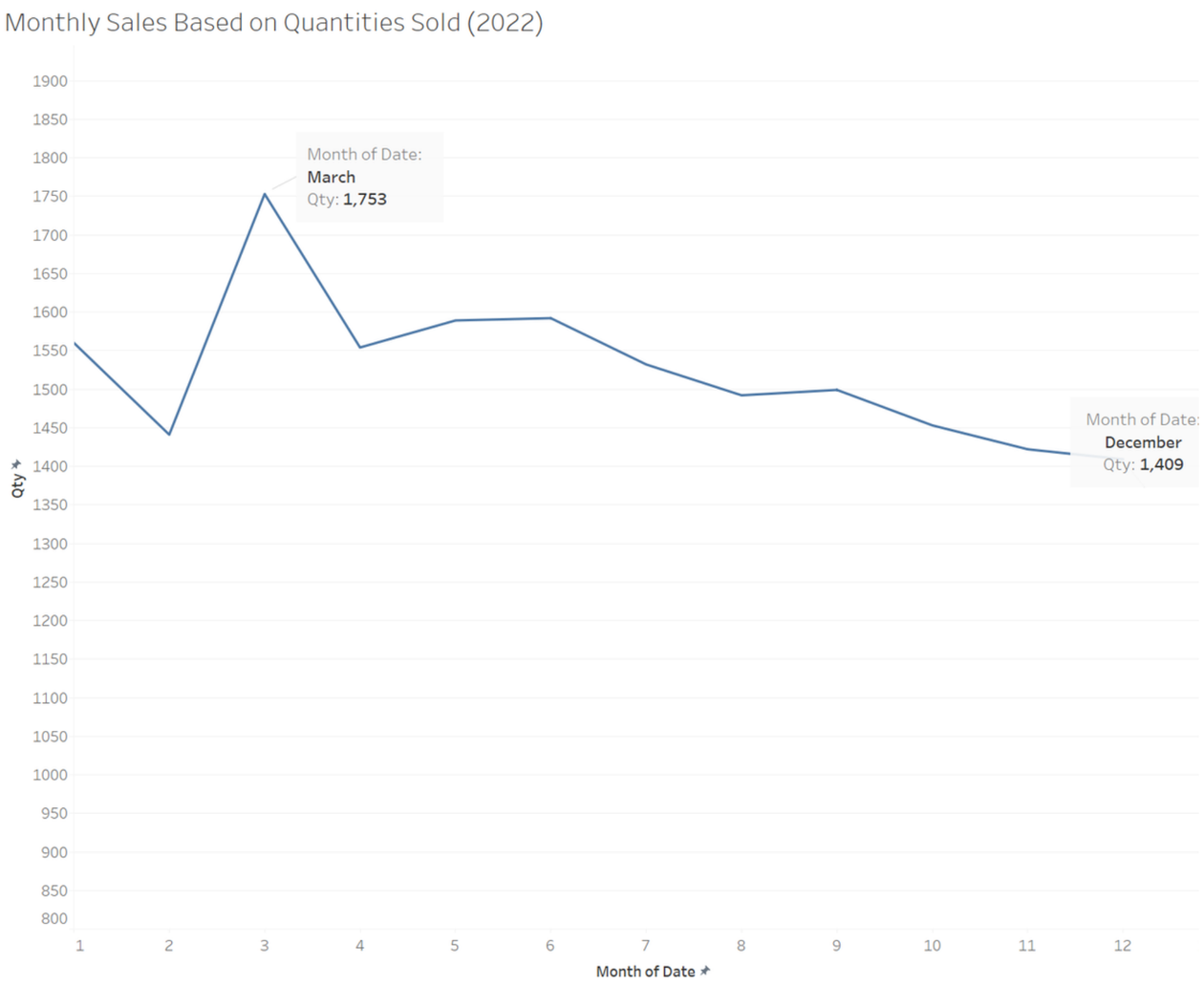


TABLEAU WORKSHEET

Visualization of Daily Revenue in 2022

From the visualization, we can see that the amount of TotalAmount earned per day fluctuates greatly. TotalAmount peaked on March 2 2022 with a revenue of Rp976,500. While the date with the lowest revenue is August 12 2022 with a revenue of Rp123,600.

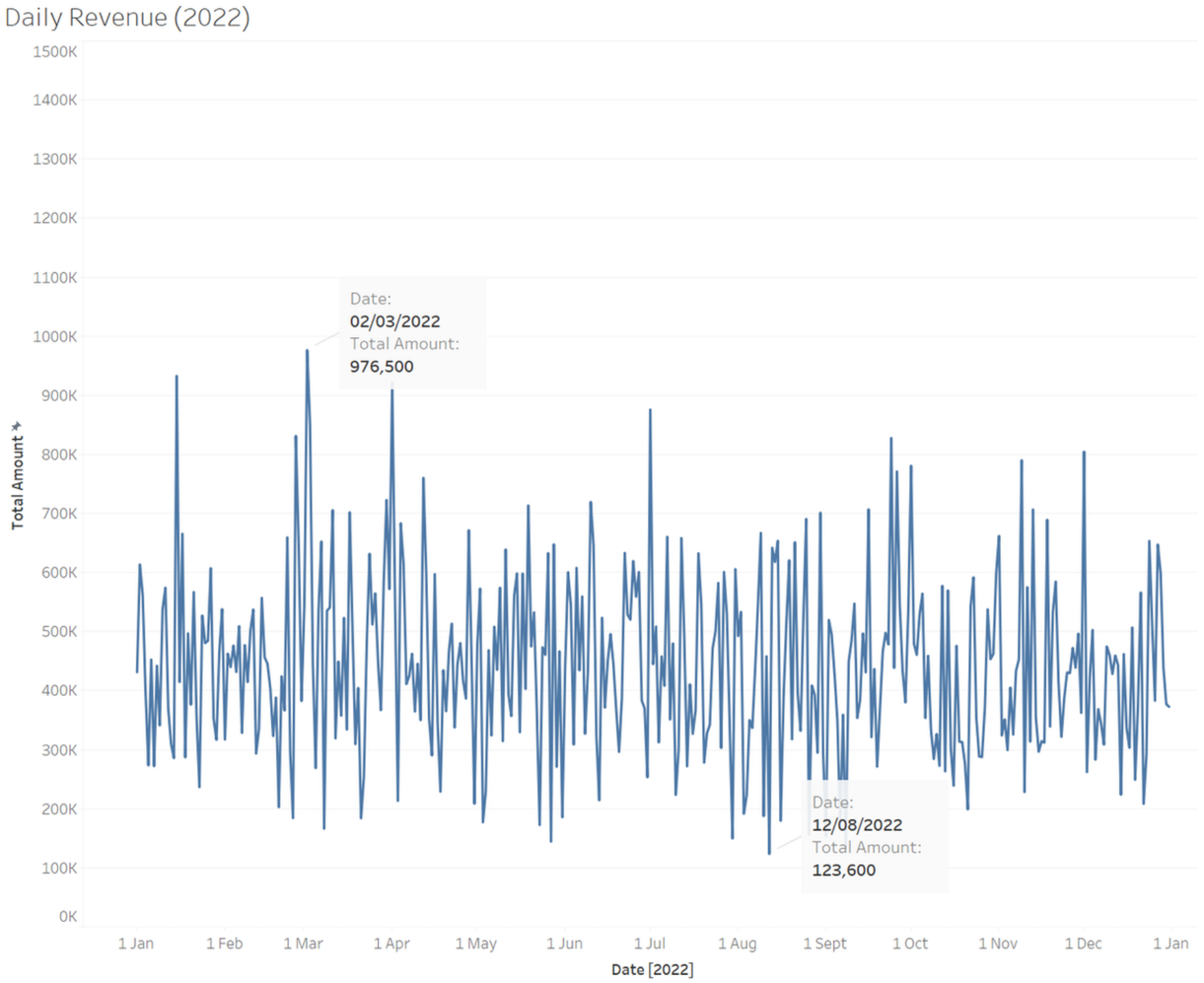


TABLEAU WORKSHEET

Visualization of the Sales of Each Product

From the visualization, we can see that the best-selling product is ‘Thai Tea’ with a total of 2853 items sold. While the item with the least sales is Cashew with 627 items sold.

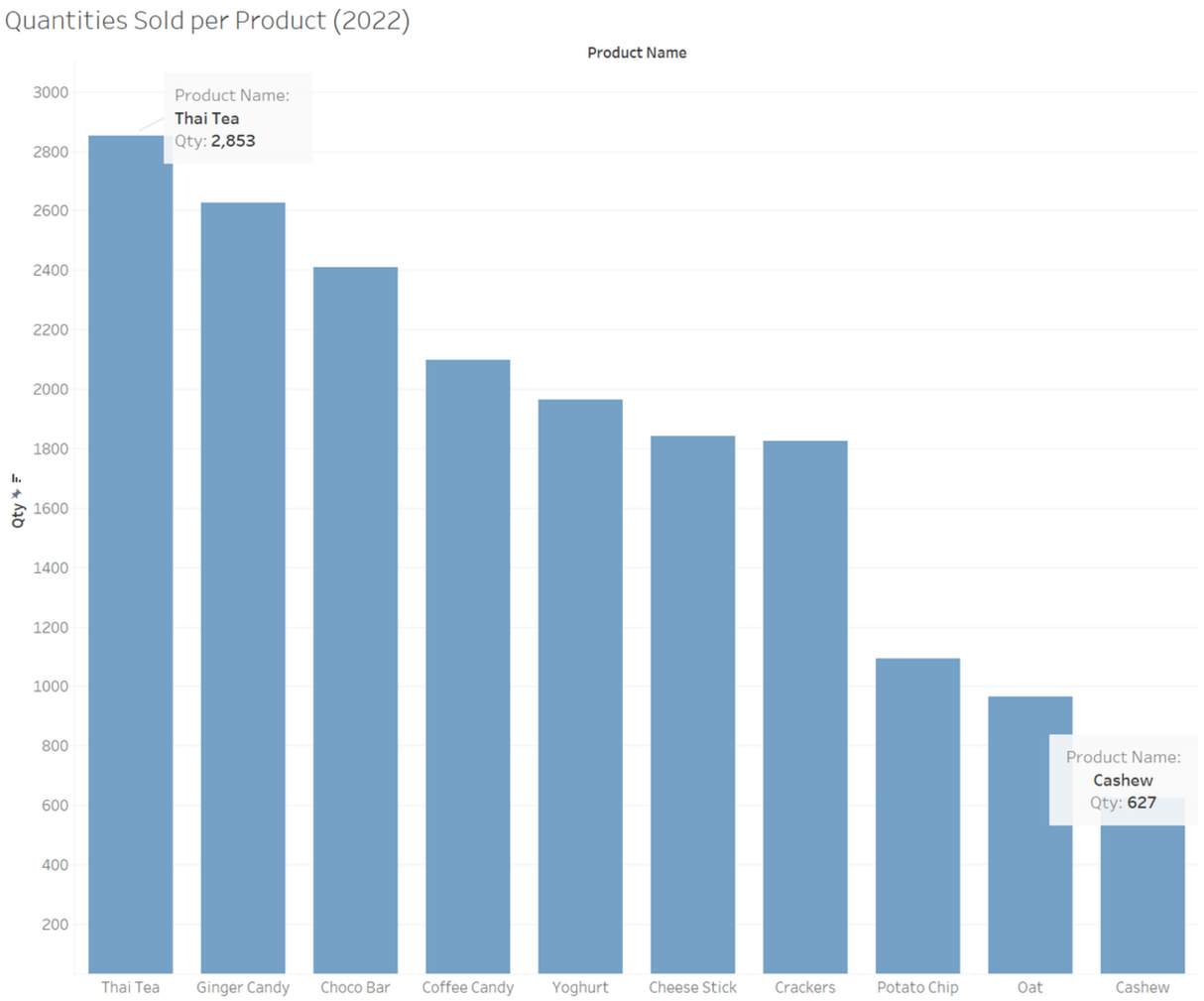
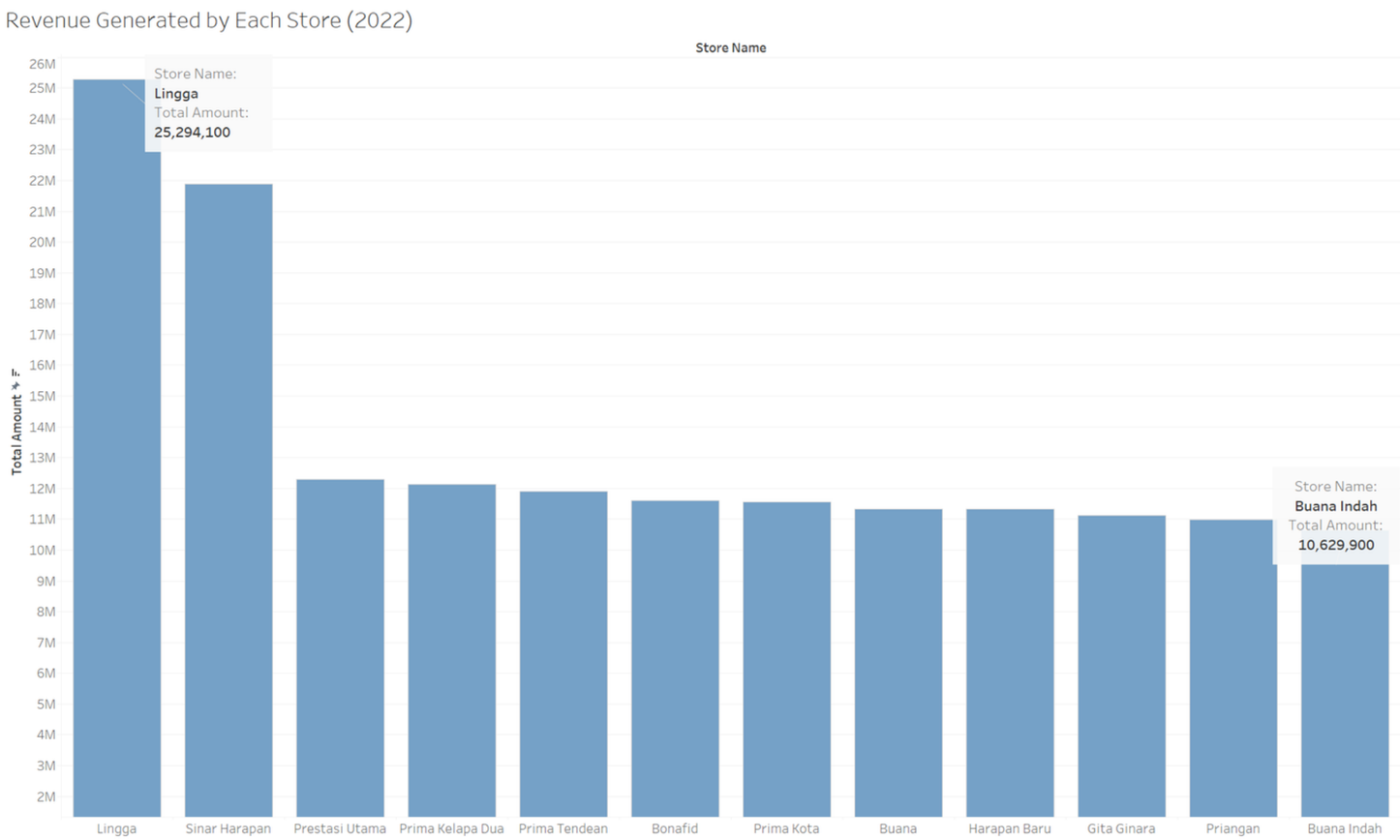


TABLEAU WORKSHEET



Visualization of Generated Revenue by Store

From the visualization, we can see that the store with the highest revenue is Lingga with a total revenue of Rp25,294,100. While the store with the lowest revenue is Buana Indah with a total revenue of Rp10,629,900.

ARIMA TIME SERIES SALES FORECASTING

DATA CLEANING

Missing Values

There are 3 missing values in 'customer' table. Thus, we need to fill it with fillna(method = 'ffill').

```
print(customer.isna().sum())
```

```
CustomerID        0
Age               0
Gender            0
Marital Status    0
Income            0
dtype: int64
```

Wrong Data Type

In 'customer', 'store', and 'transaction' table, there are some variables that doesn't match the data type it's supposed to be. Example, 'income' variable is type object instead of float. Thus we need to change this using astype() function.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 447 entries, 0 to 446
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   CustomerID      447 non-null    int64
 1   Age             447 non-null    int64
 2   Gender          447 non-null    int64
 3   Marital Status  447 non-null    object
 4   Income          447 non-null    object
dtypes: int64(3), object(2)
memory usage: 17.6+ KB
None
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   StoreID     14 non-null     int64
 1   StoreName   14 non-null     object
 2   GroupStore  14 non-null     object
 3   Type        14 non-null     object
 4   Latitude    14 non-null     object
 5   Longitude   14 non-null     object
dtypes: int64(1), object(5)
memory usage: 800.0+ bytes
None
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5020 entries, 0 to 5019
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   TransactionID   5020 non-null   object
 1   CustomerID      5020 non-null   int64
 2   Date            5020 non-null   object
 3   ProductID       5020 non-null   object
 4   Price           5020 non-null   int64
 5   Qty             5020 non-null   int64
 6   TotalAmount     5020 non-null   int64
 7   StoreID         5020 non-null   int64
dtypes: int64(5), object(3)
memory usage: 313.9+ KB
None
```

```
customer['Income'] = customer['Income'].str.replace(',', '.')
customer['Income'] = customer['Income'].astype(float)
store['Longitude'] = store['Longitude'].str.replace(',', '.')
store['Longitude'] = store['Longitude'].astype(float)
store['Latitude'] = store['Latitude'].str.replace(',', '.')
store['Latitude'] = store['Latitude'].astype(float)
transaction['Date'] = pd.to_datetime(transaction['Date'], format='%d/%m/%Y')
```

DATA PREPROCESSING

Merging Datasets

```
data_time_series = pd.merge(left = transaction,
                  right = customer,
                  left_on = 'CustomerID',
                  right_on = 'CustomerID',
                  how = 'left')
```

```
data_time_series = pd.merge(left = data_time_series,
                  right = product,
                  left_on = ['ProductID', 'Price'],
                  right_on = ['ProductID', 'Price'],
                  how = 'left')
```

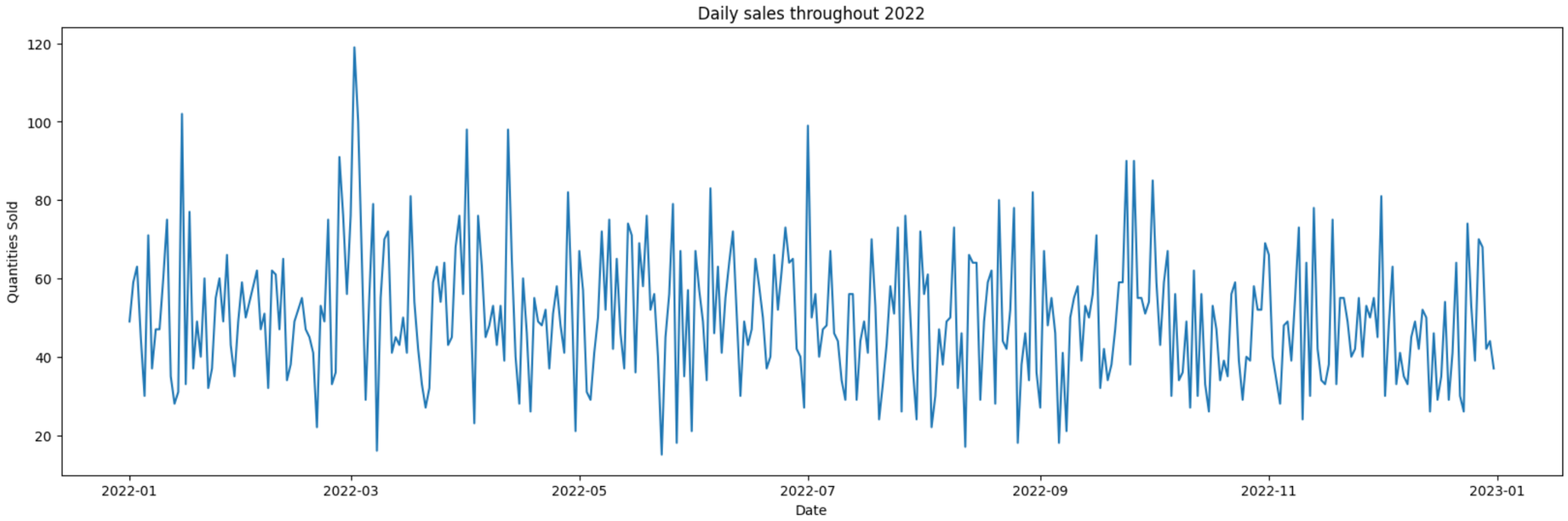
```
data_time_series = pd.merge(left = data_time_series,
            right = store,
            left_on = 'StoreID',
            right_on = 'StoreID',
            how = 'left')
```

Aggregating Datasets

```
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
data_qty = df.groupby('Date').agg({'Qty':'sum'})
```

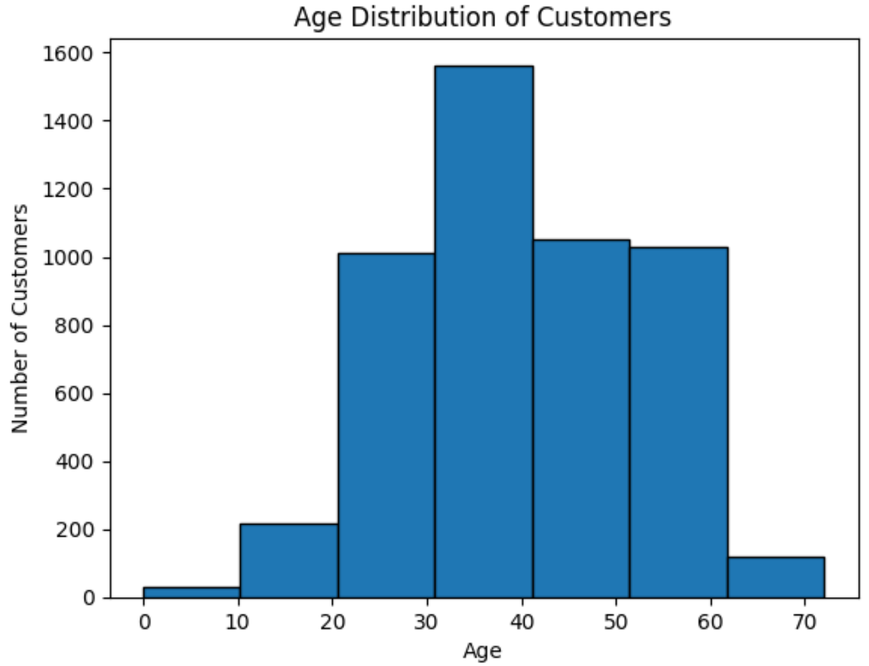
To be able to analyze the data even further, we need to merge the datasets given to us according to the foreign keys present in the dataset. Furthermore, we need to aggregate the data based on the date the transaction took place and sum the 'Qty'.

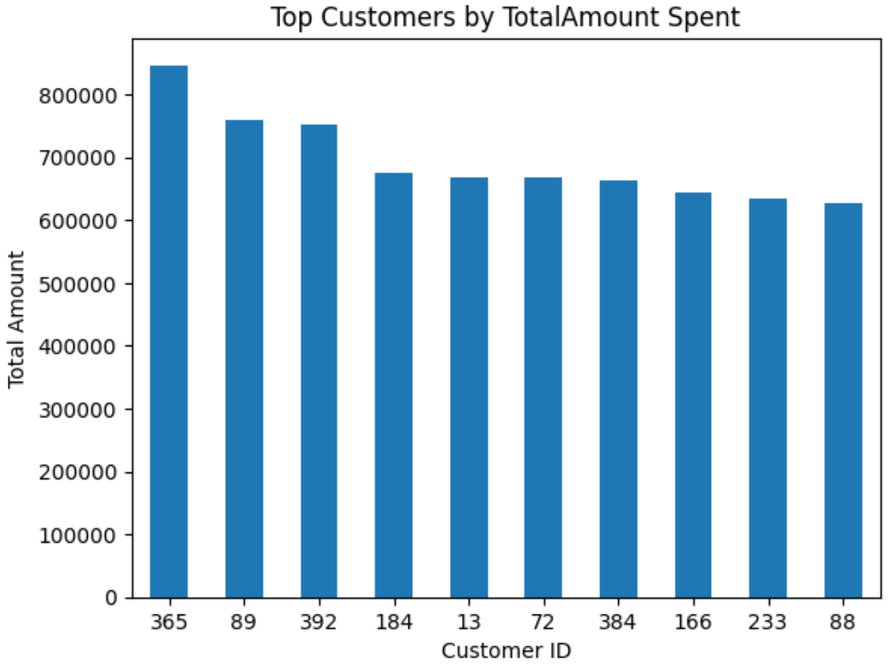
EXPLORATORY DATA ANALYSIS

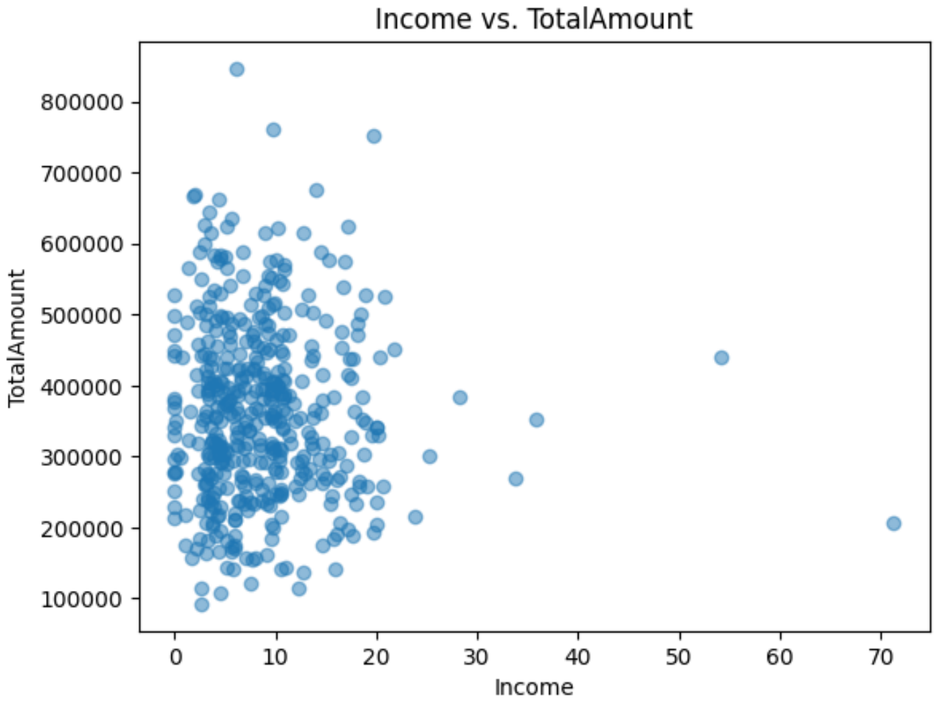


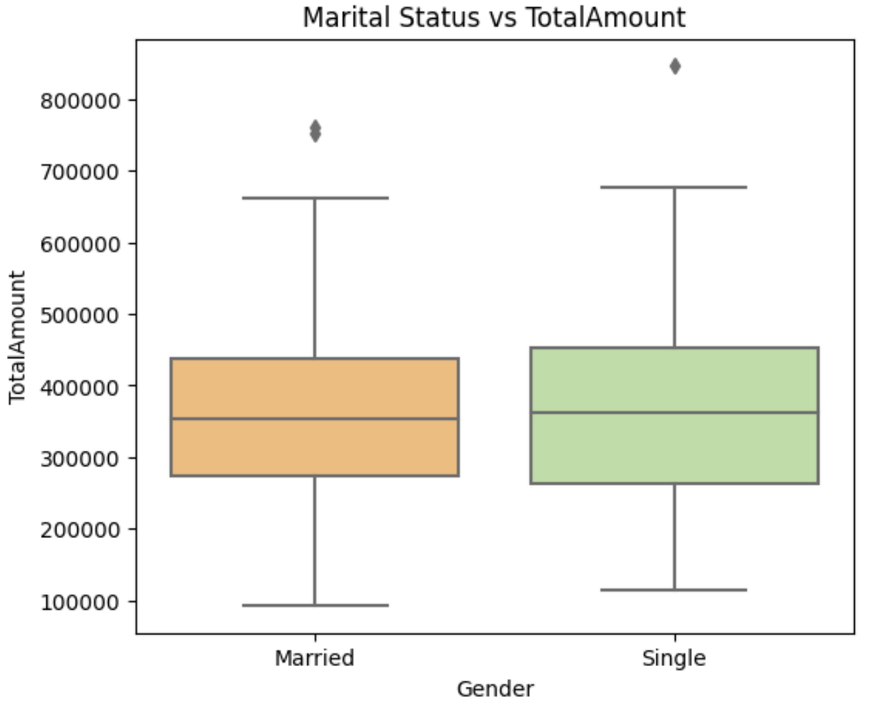
Visualization of daily sales based on quantities sold throughout 2022

EXPLORATORY DATA ANALYSIS

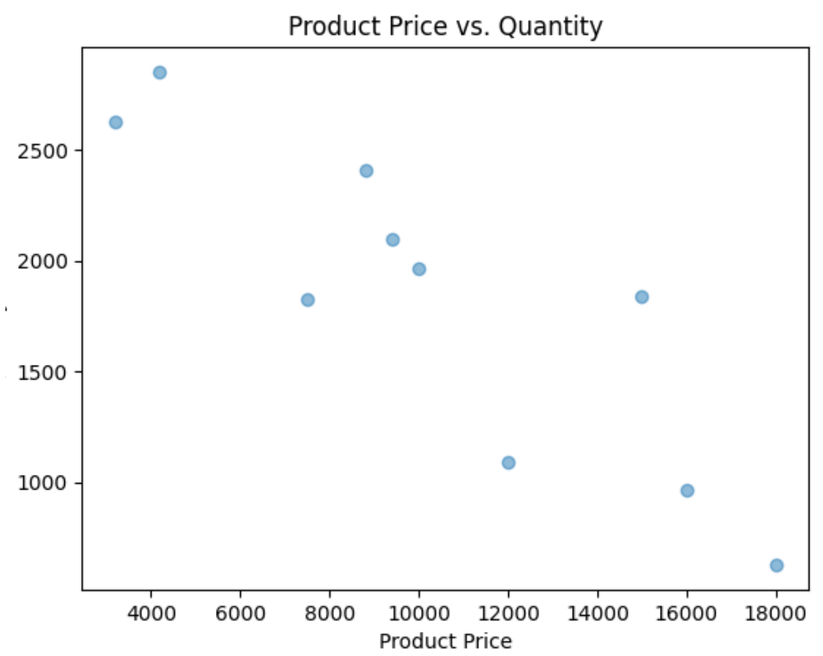


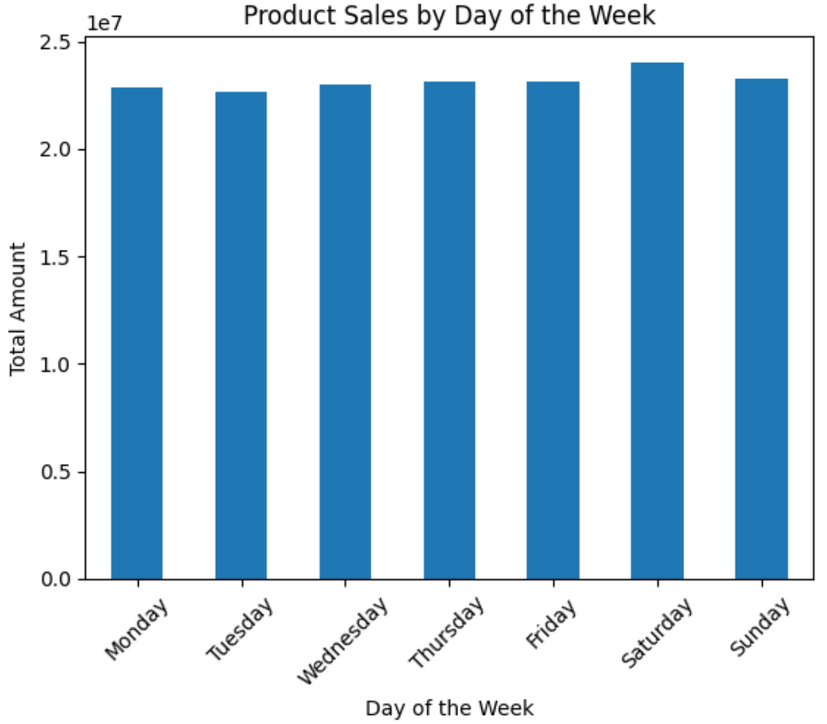




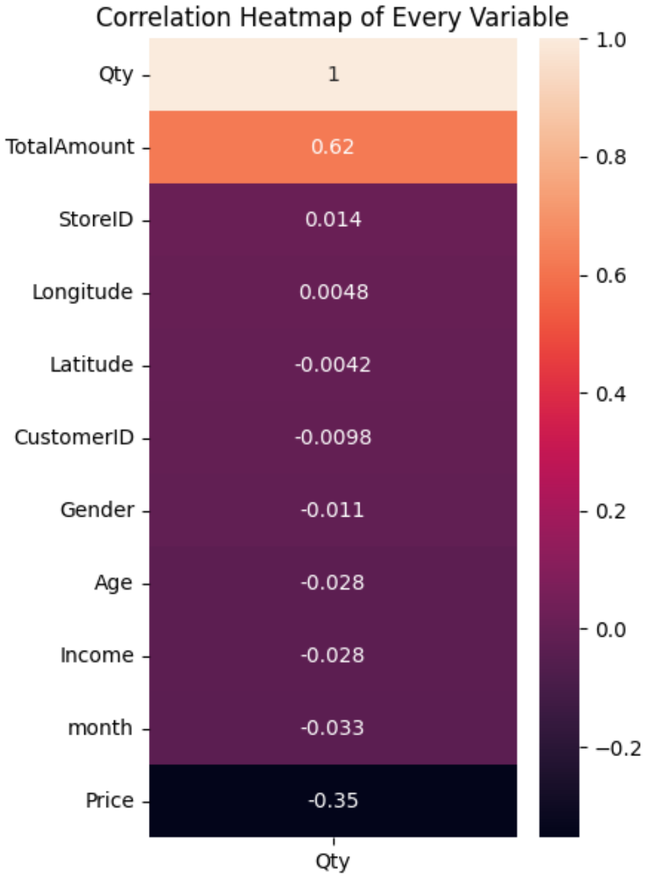


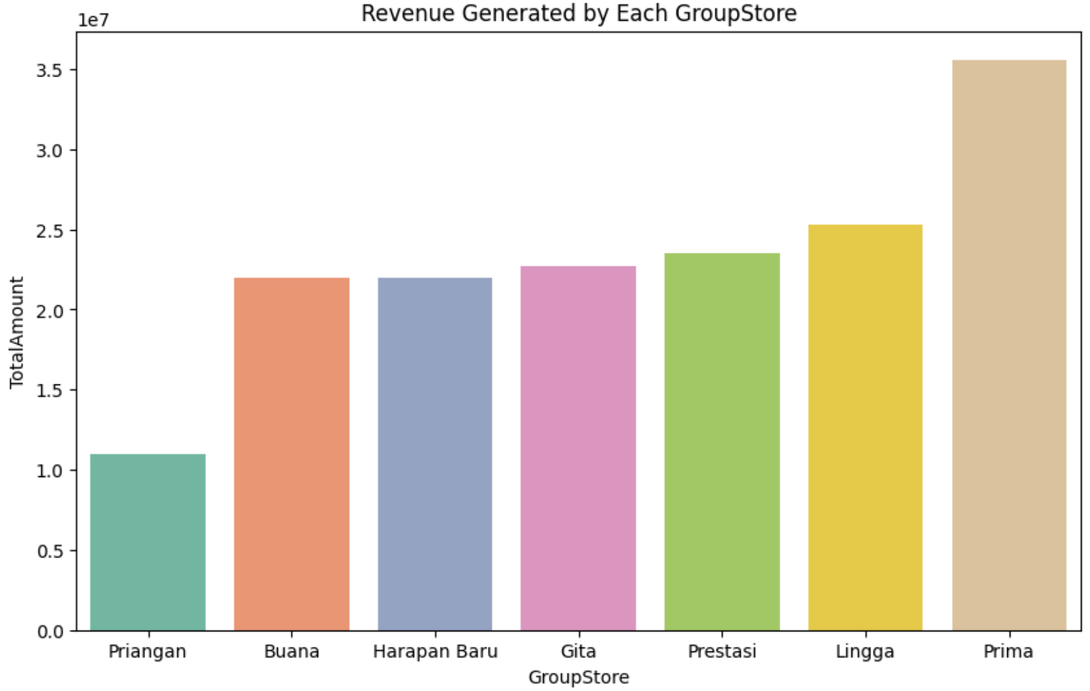
EXPLORATORY DATA ANALYSIS

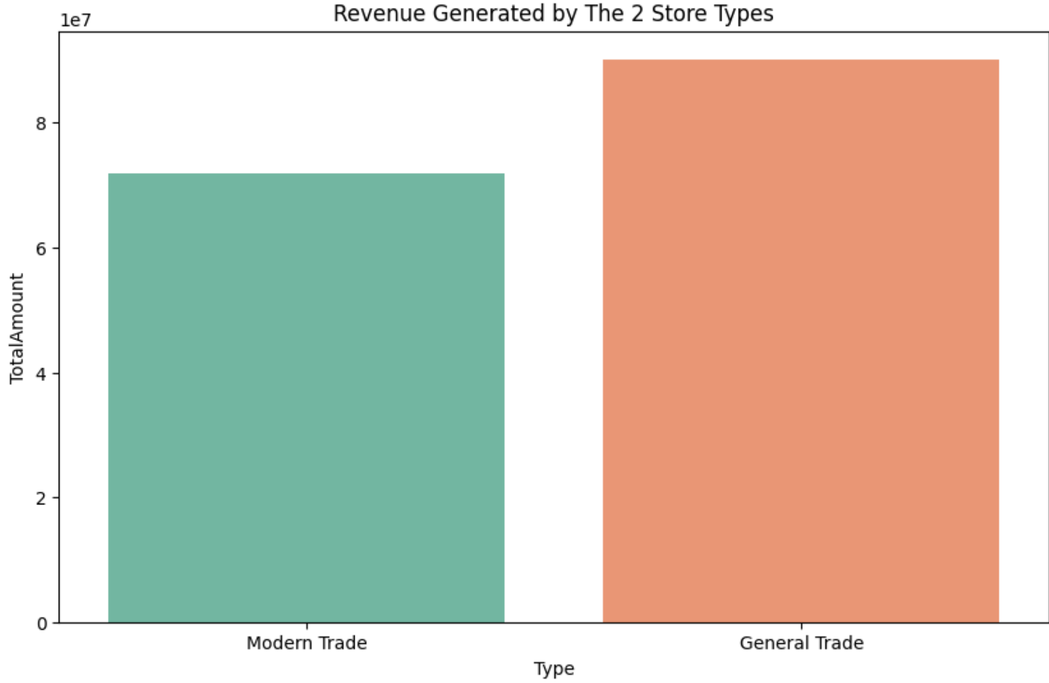




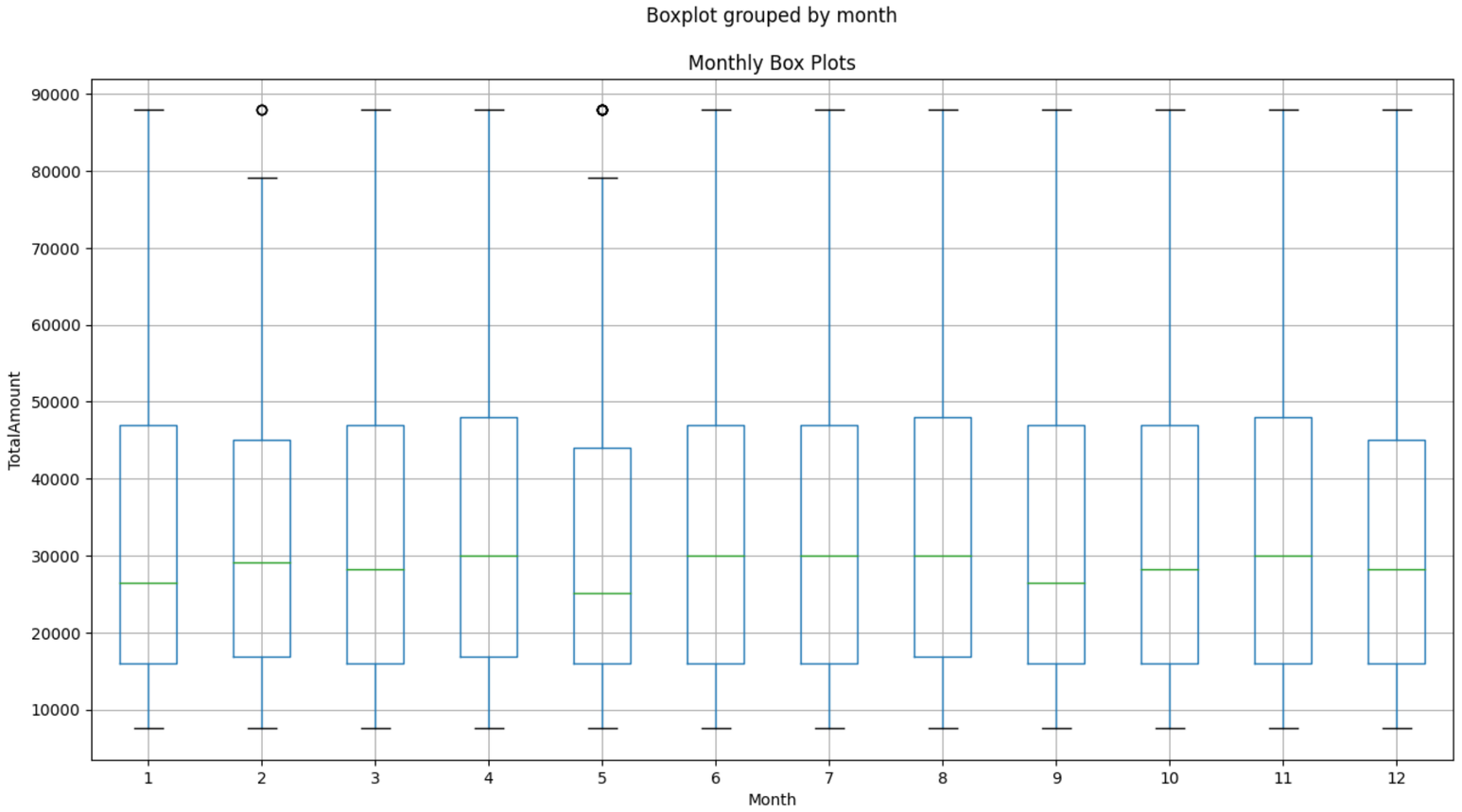
EXPLORATORY DATA ANALYSIS







EXPLORATORY DATA ANALYSIS



There is no significant difference between the revenue generated from each month.

STATIONARITY CHECK

```
1. ADF:  -19.448086319449082
2. P-Value:  0.0
3. Num of Lags:  0
4. Num of Observations Used for ADF Regression and Critical Values Calculation:  364
5. Critical Values:
        1% :  -3.4484434475193777
        5% :  -2.869513170510808
        10% :  -2.571017574266393
```

H0: The data is not stationary

H1: The data is stationary

We reject the null hypothesis if p-value < 0.05.

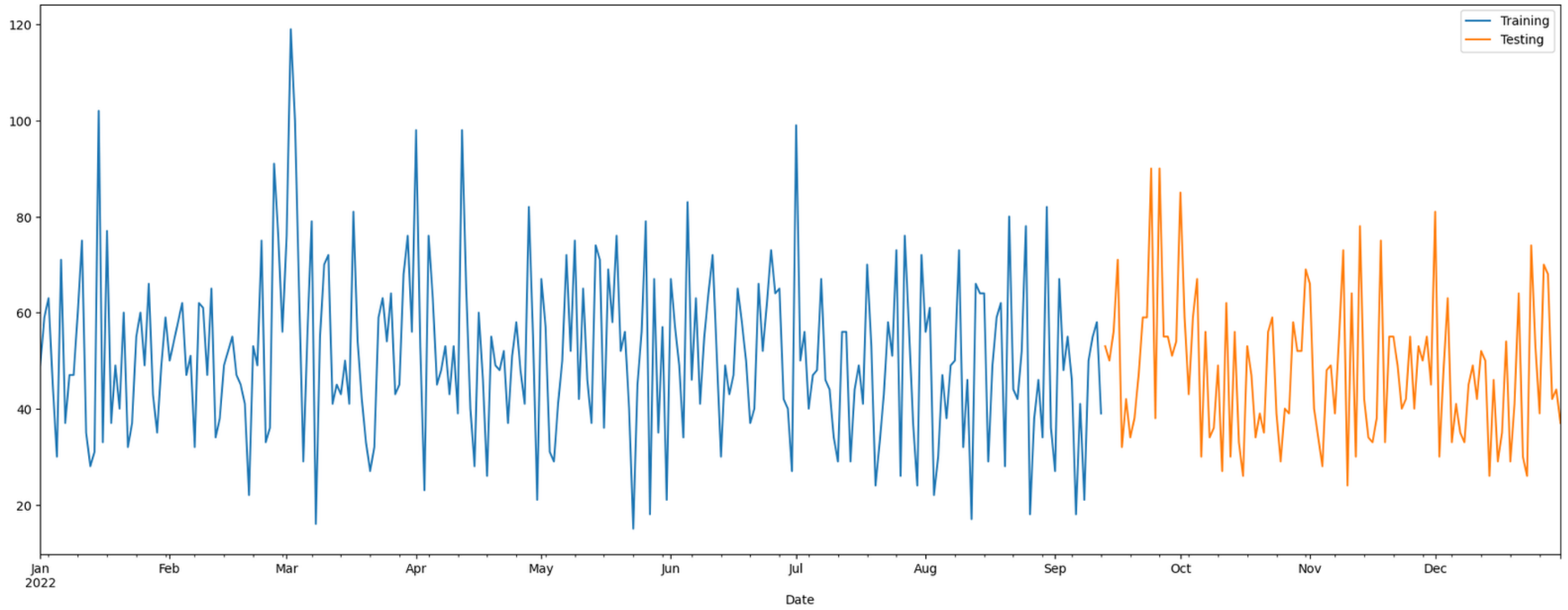
Because p-value = 0, p-value < 0.05. Therefore, **the data is stationary**.

MODELLING ARIMA

ARIMA stands for AutoRegressive Integrated Moving Average. It's a widely used time series forecasting and analysis method that combines autoregressive (AR) and moving average (MA) components with differencing to model and predict time series data. ARIMA is designed to handle various types of time series data, including those with trends, seasonality, and noise.

1. Splitting data into training and testing data
2. Finding the best parameter for the model and fitting the training data into the model
3. Testing the model on the testing data
4. Evaluating the model

SPLITTING DATA TO TRAINING AND TESTING DATA



Spllit data into training and testing data

Train: 255 rows

Test: 110 rows

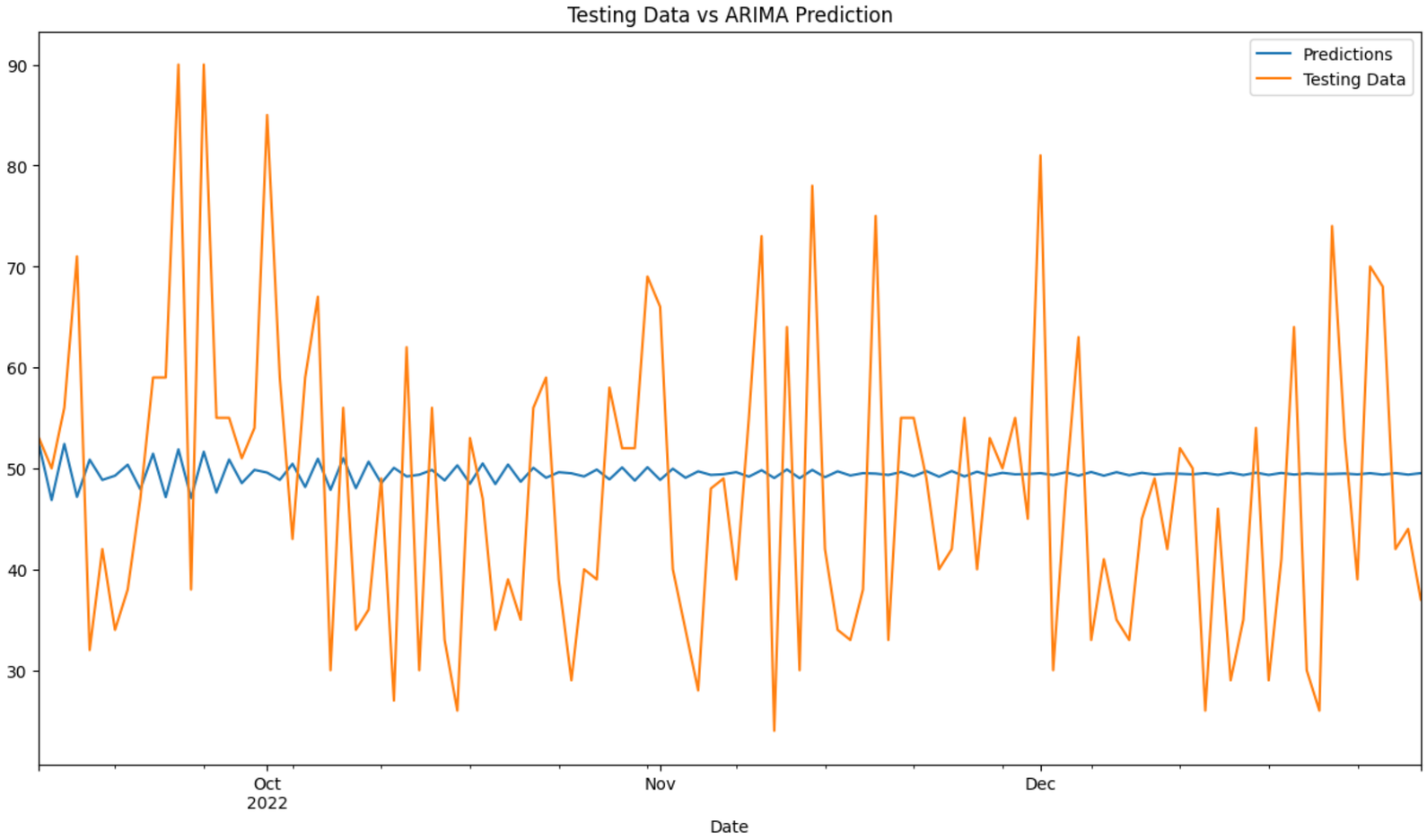
FINDING THE BEST PARAMETERS AND FIT TRAINING DATA

```
stepwise_model = auto_arima(train['Qty'], start_p = 1, start_q = 1, max_p = 5,
                            max_q = 5, m = 1, start_P = 0, seasonal = False,
                            d = 1, D = 1, trace = True, error_action = 'ignore',
                            supress_warnings = True, stepwise = True)
print(stepwise_model.summary())
```

```
Best model:  ARIMA(4,1,3)(0,0,0)[0]
Total fit time: 27.815 seconds
                                SARIMAX Results
==========================================================================================
Dep. Variable:                           y   No. Observations:                  255
Model:                 SARIMAX(4, 1, 3)   Log Likelihood               -1081.451
Date:                  Mon, 21 Aug 2023   AIC                           2178.903
Time:                          15:16:05   BIC                           2207.202
Sample:                      01-01-2022   HQIC                          2190.287
                           - 09-12-2022
Covariance Type:                    opg
==========================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
ar.L1         -1.8409      0.066    -27.766      0.000      -1.971      -1.711
ar.L2         -1.0066      0.113     -8.944      0.000      -1.227      -0.786
ar.L3         -0.1992      0.131     -1.520      0.128      -0.456       0.058
ar.L4         -0.1100      0.072     -1.528      0.127      -0.251       0.031
ma.L1          0.8733      0.047     18.592      0.000       0.781       0.965
ma.L2         -0.8844      0.035    -25.428      0.000      -0.953      -0.816
ma.L3         -0.9422      0.044    -21.191      0.000      -1.029      -0.855
sigma2       285.8242     22.208     12.870      0.000     242.297     329.351
==========================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                22.66
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               0.80   Skew:                             0.60
Prob(H) (two-sided):                  0.32   Kurtosis:                         3.84
==========================================================================================
```

By using auto_arima() with certain parameters, we have determined that the best parameters for the ARIMA model are where p = 4, d = 1, and q = 3 with AIC = 2178.903.

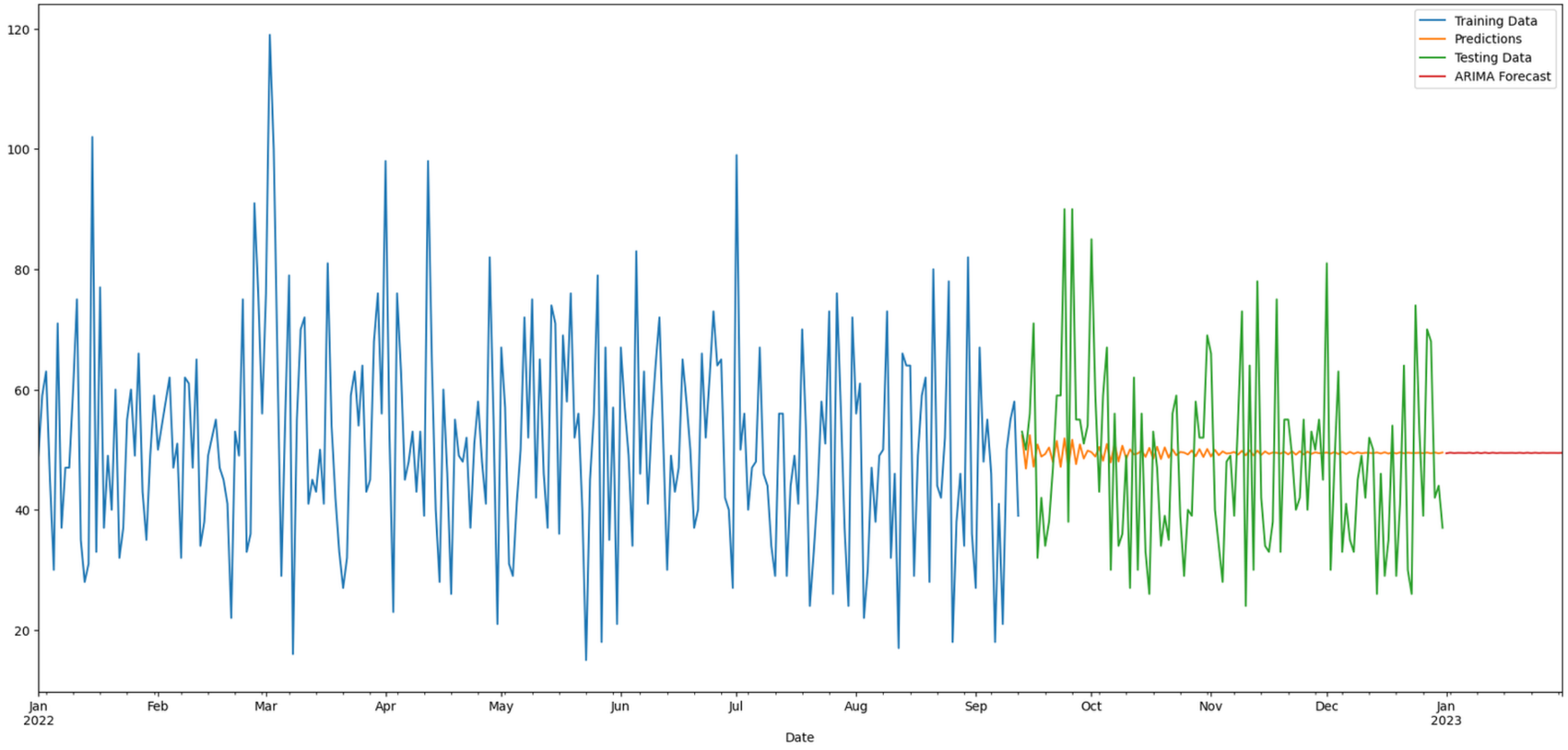
TESTING THE MODEL ON THE TESTING DATA



Testing and prediction comparison

The prediction plot doesn't show fluctuation, this happens because the historical data doesn't show strong seasonality and the forecasting model finds it difficult to predict the future data points. Therefore it simply take average of our previous values and predict as future.

TESTING THE MODEL ON THE TESTING DATA



Training, testing, prediction, and ARIMA Forecast on Future Dates

MODEL EVALUATION

MAPE: 28.63%
MAE: 12.05
MSE: 216.95
RMSE: 14.73
Explained Variance: 0.00

Error value and Explained Variance

CUSTOMER SEGMENTATION USING KMEANS CLUSTERING

DATA PREPROCESSING

Merging Datasets

```
data_clustering = pd.merge(left = transaction,
                  right = customer,
                  left_on = 'CustomerID',
                  right_on = 'CustomerID',
                  how = 'left')
```

```
data_clustering = pd.merge(left = data_clustering,
                  right = product,
                  left_on = ['ProductID', 'Price'],
                  right_on = ['ProductID', 'Price'],
                  how = 'left')
```

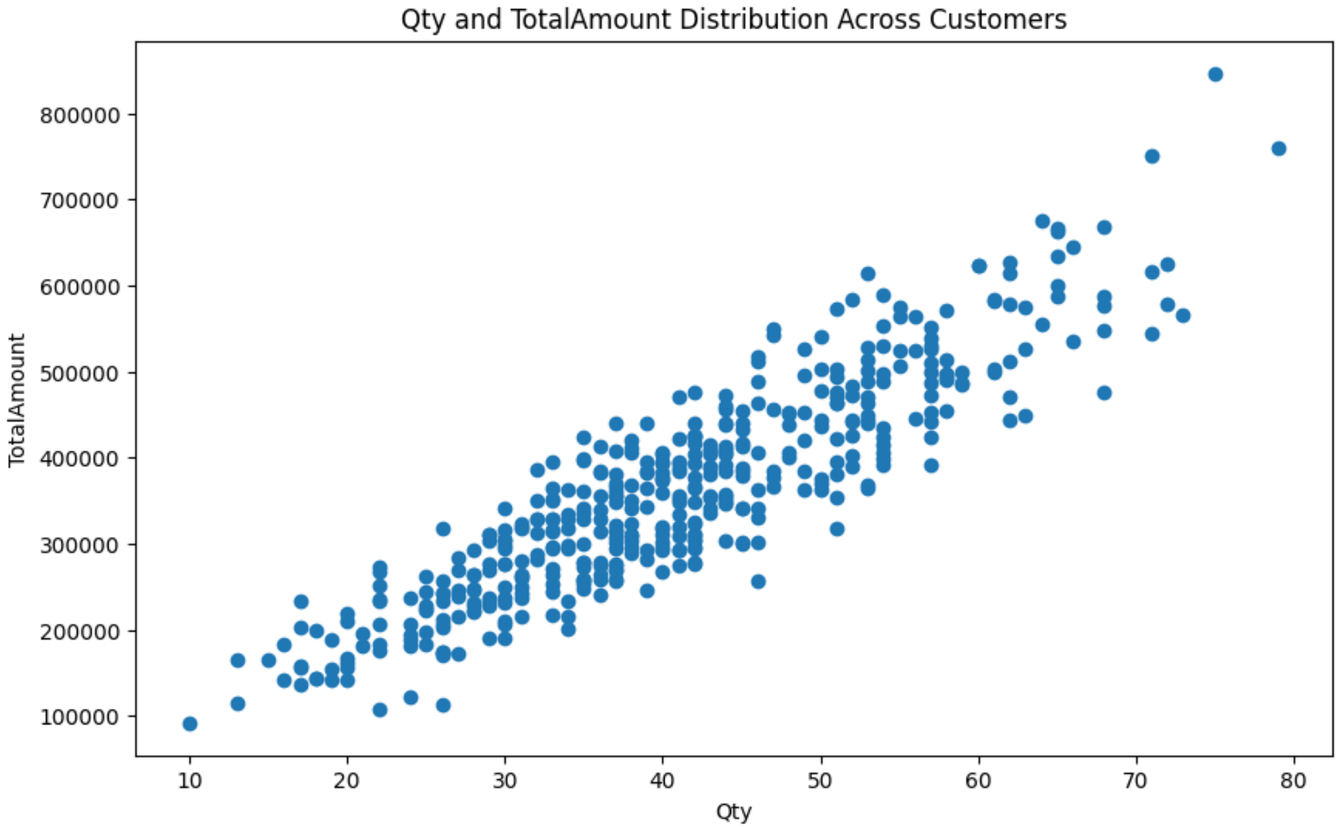
```
data_clustering = pd.merge(left = data_clustering,
                  right = store,
                  left_on = 'StoreID',
                  right_on = 'StoreID',
                  how = 'left')
```

Aggregating Datasets

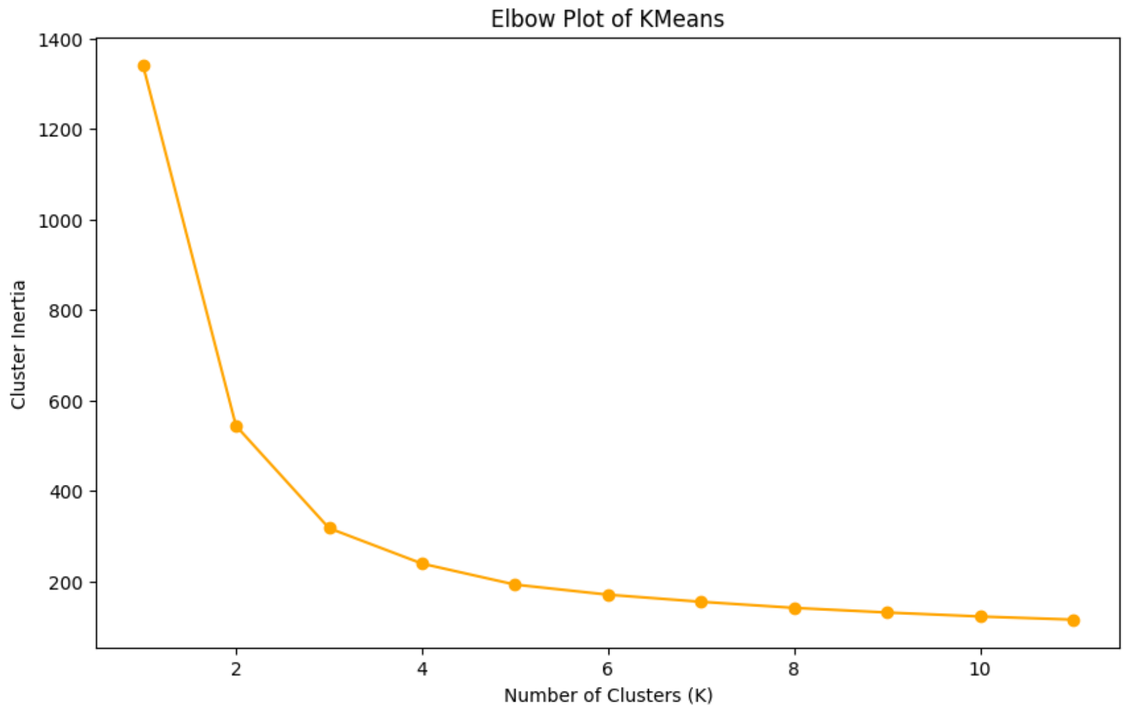
```
df = data_clustering.groupby('CustomerID').agg({'TransactionID':'count',
                                                'Qty': 'sum',
                                                'TotalAmount': 'sum'})
```

To be able to analyze the data even further, we need to merge the datasets given to us according to the foreign keys present in the dataset. Furthermore, we need to aggregate the data based on the date the transaction took place and sum the 'Qty'.

SCATTER PLOT



FINDING THE RIGHT K



From the elbow plot, we can conclude that K is between 4 and 5.
But this time we will use K = 4. Then we fit the model.

MODEL FITTING

```
kmeans_model = KMeans(n_clusters = 4)
kmeans_model.fit(scaled_data)
```

```
KMeans
KMeans(n_clusters=4)
```

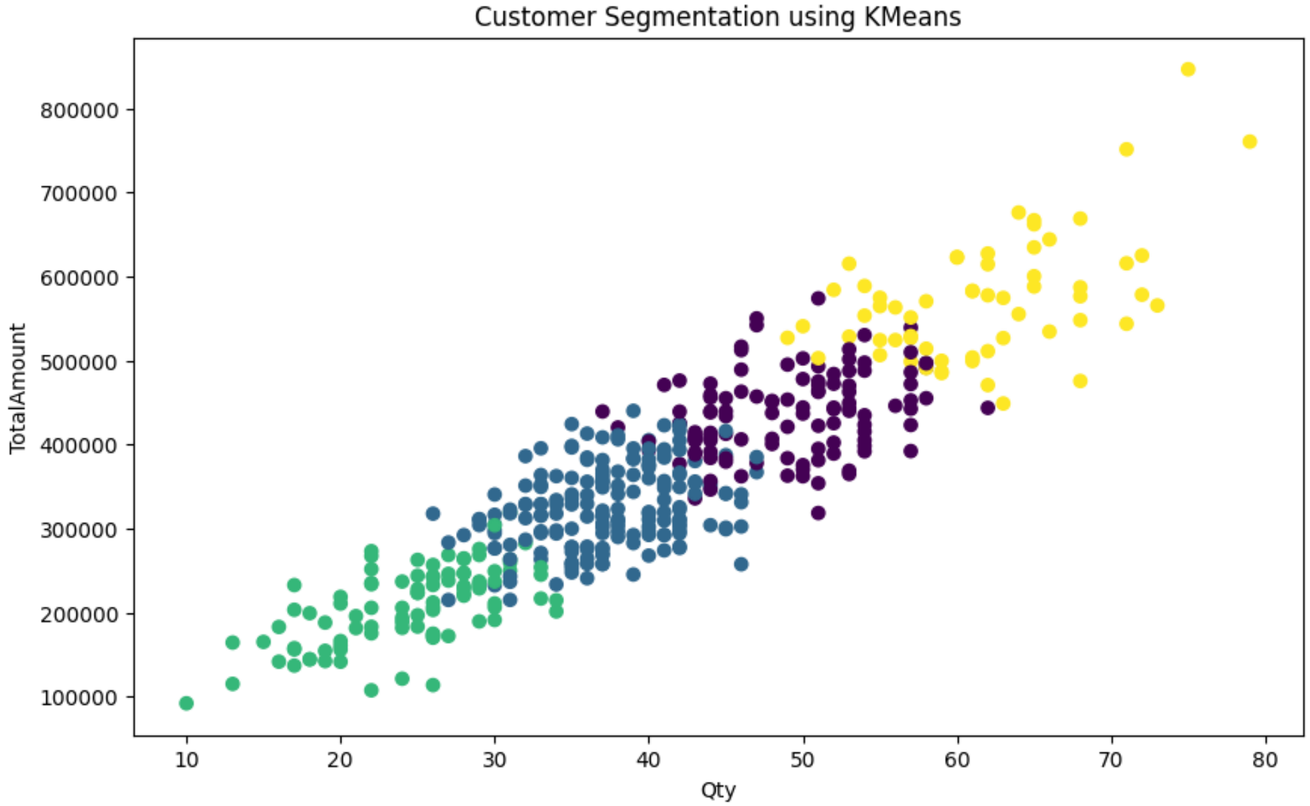
```
df['Cluster'] = kmeans_model.labels_
```

After fitting the model, we add 'Cluster' column to our dataset so that we can plot the different clusters of cutomers.

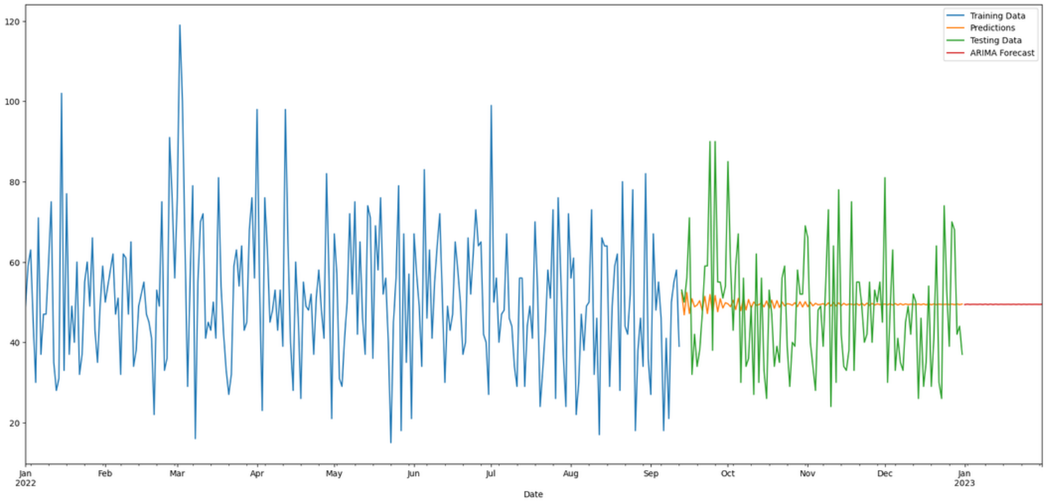
CUSTOMER SEGMENTATION PLOT

From the plot, we can conclude that there is 4 clusters of customers that have different types of buying pattern.

- Low-spending Customers
- Lower-middle-spending Customers
- Upper-middle-spending Customers
- High-spending Customers



SUMMARY REPORT

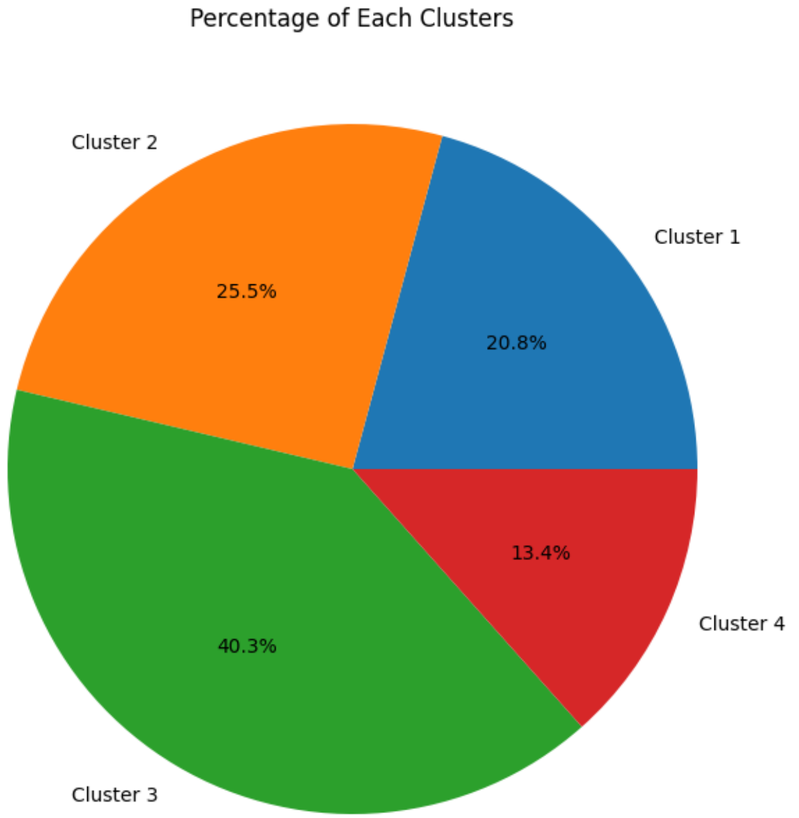


Time Series Forecasting

The historical data doesn't show strong seasonality and the forecasting model finds it difficult to predict the future data points. Thus, using a different model fitted for this kind of data might be a good idea.

Customer segmentation

Cluster 3 (upper-middle-spending) make up 40.3% of overall customers, followed by cluster 2 (lower-middle-spending) with 25.5%, cluster 1 with 20.8% (low-spending), and cluster 4 with just 13.4% (high-spending).



GITHUB LINK

https://github.com/Cyntiahp31/Kalbe-Data-Scientist-Intern-Project-Sales-Forecasting-and-Cluster-Segmentation/tree/main

PRESENTATION LINK

https://drive.google.com/file/d/1FWyAIbOHS0kQmROUrkZH5bT9xVIlKlXX/view?usp=sharing

THANK YOU

Rakamin Academy X KALBE Nutritionals