# Faculty of Computers and Artificial Intelligence.

**Forecasting and Predictive Analytics**

**Project Assignment**

**"Retail Sales Forecasting and Customer Segmentation"**

**2023 / 2024**

**Supervisor: Dr . Ghada Tolan**

**TA. Ahmed Sayed Fouad**

## 1. Data Exploration and Preprocessing:

### A) Data Cleaning

- **Missing Value:**

  - **Preserving Categorical Information For Product Id:**
    - The (**Product id )** is a categorical variable, will calculate mode formula, helps preserve the categorical nature of the variable. The mode represents the most frequent category and can be a reasonable estimate for missing values, so based on that take Mode for **missing values** in column (**Product id)**.

  - **Using FORECAST.LINEAR To Forecast Quantity Sold:**
    - in the missing value context implies that i are assuming a linear trend in the **quantity sold** over time for the relationship between datetime and quantity sold follows a roughly linear trend, meaning that the quantity sold increases or decreases steadily over time.

- **The Outliers for Data:**
  - This formula calculates the z-score for each data point. The z-score measures how many standard deviations a data point is from the mean. If the absolute z-score is less than 2, the data point is considered not to be an outlier based on chosen threshold. We can apply this formula to each cell in column and filter or highlight the values where the formula evaluates.

- **Remove duplicates:**
  - There aren't any duplicating values.

- **correct inconsistencies in the dataset:**
  - we checked by standardizing text data, fixing date and standardizing categorical data.

## • B) Exploratory Data Analysis (EDA)

| Statistic descriptive | total_purchase_amount | age | sales_price |
|---|---|---|---|
| | | | |
| Mean | 3370.26 | 40.33 | 69.19 |
| Standard Error | 50.69 | 0.42 | 2.37 |
| Median | 2356.00 | 36.00 | 44.00 |
| Mode | 5544.00 | 23.00 | 7.00 |
| Standard Deviation | 1672.87 | 13.74 | 78.06 |
| Sample Variance | 2798489.36 | 188.80 | 6093.27 |
| Kurtosis | -1.58 | -1.19 | 4.13 |
| Skewness | 0.24 | 0.41 | 2.22 |
| Range | 4412.00 | 42.00 | 316.00 |
| Minimum | 1132.00 | 23.00 | 7.00 |
| Maximum | 5544.00 | 65.00 | 323.00 |
| Sum | 3670211.00 | 43917.00 | 75347.00 |
| Count | 1089.00 | 1089.00 | 1089.00 |

- This provides a comprehensive overview of three variables:

  - total_purchase_amount , age, and sales_price. These descriptive statistics aim to offer insights into the central tendency, dispersion, and shape of the distributions for each variable.

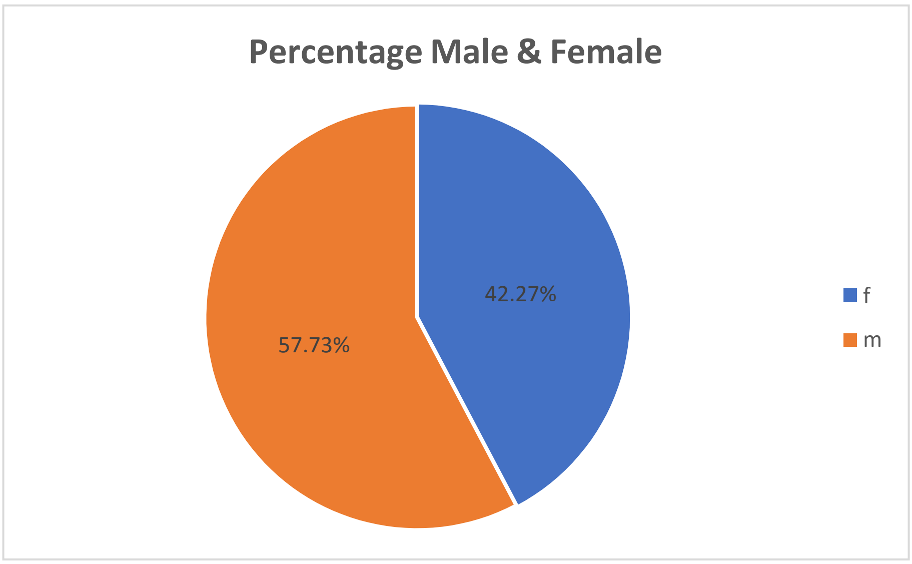
  - The total purchase amount has a mean of 3370.26, indicating the average value of purchases. The distribution is slightly positively skewed (skewness = 0.24) and exhibits a moderate kurtosis (-1.58).

  - The average age is 40.33, and the distribution is slightly positively skewed (skewness = 0.41). The kurtosis value of -1.19 suggests a relatively flat distribution.

  - The mean sales price is 69.19, and the distribution is positively skewed (skewness = 2.22) with a high kurtosis (4.13), indicating heavy tails.

## C) Data Visualization

### 1. Age and Gender Analysis:



- The dataset used for this analysis includes information on both age and gender distribution. The total population considered is 100%, with specific emphasis on various age and gender categories:
    - Males aged 20-30: 36%.
    - Males aged 30-40: 46%.
    - Males aged 40-60: 18%.
    - Females aged 20-30: 21%.
    - Females aged 30-50: 22%.
    - Females aged 50-60: 57%.

    - **Key Findings:**

    - Male Age Distribution:

      Males aged 20-30 constitute 36% of the total population.

      Males aged 30-40 represent the largest share at 46%, indicating a higher concentration in this age bracket.

      Males aged 40-60 account for 18% of the population.

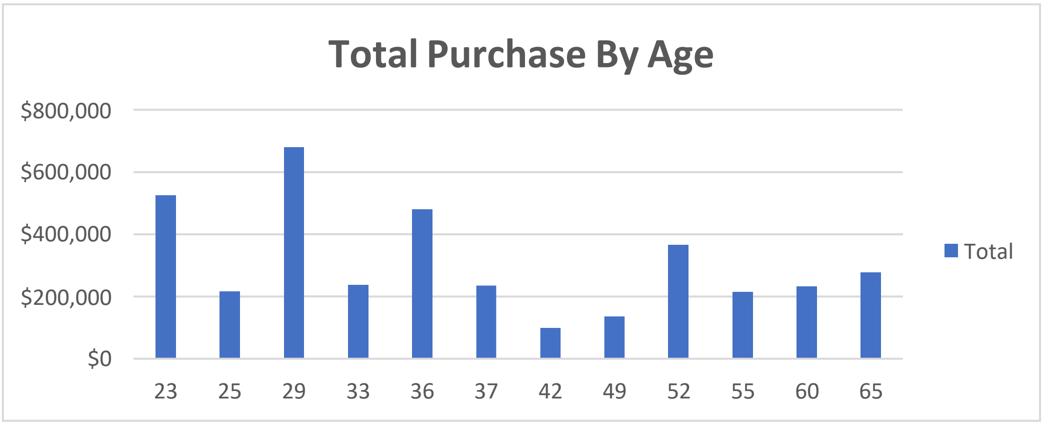
- Female Age Distribution:

  Females aged 20-30 make up 21% of the total population.

  Females aged 30-50 represent 22% of the population.

  Females aged 50-60 have the highest percentage at 57%, indicating a notable concentration in the 50-60 age range.

## 2. Sales by Age Group:

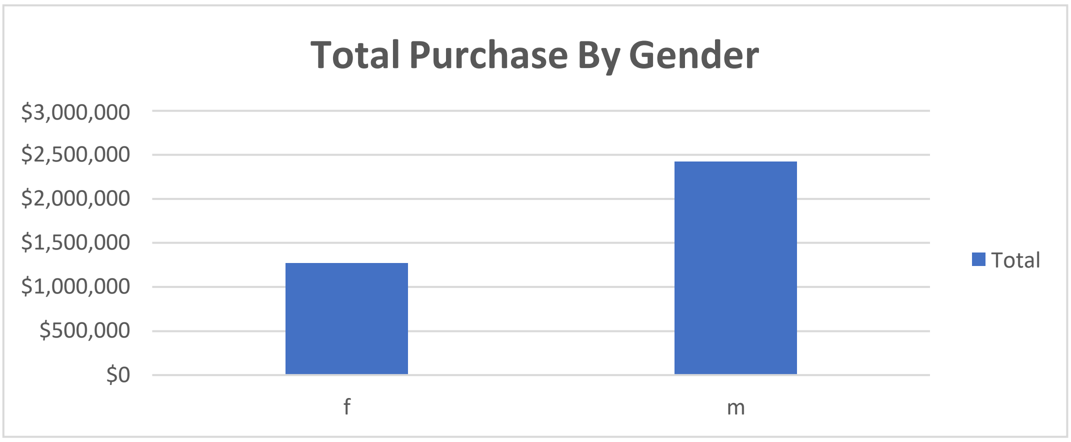


1. **Age Interval: 20-29**
   - Customers in the age range of 20 to 29 have accumulated a total purchase amount of $1,419,814.
2. **Age Interval: 30-39**
   - The age range of 30 to 39 exhibits a total purchase amount of $716,310.
3. **Age Interval: 40-49**
   - Customers aged between 40 and 49 have contributed $364,914 in total purchases.
4. **Age Interval: 50-59**
   - The age group of 50 to 59 shows a total purchase amount of $600,501.
5. **Age Interval: 60-69**
   - Customers in the age range of 60 to 69 have accumulated a total purchase amount of $509,730.

Conclusion:

Analyzing the distribution of total purchase amounts within specific age intervals provides a more nuanced understanding of customer spending patterns. This information can be valuable for targeted marketing strategies, allowing businesses to tailor their approach based on the preferences and behaviors of customers in different age brackets. The highest total purchase amounts are observed in the 20-29 age interval, indicating a potentially lucrative market segment that warrants focused attention in marketing and sales efforts.

## 3. Sales by Gender:

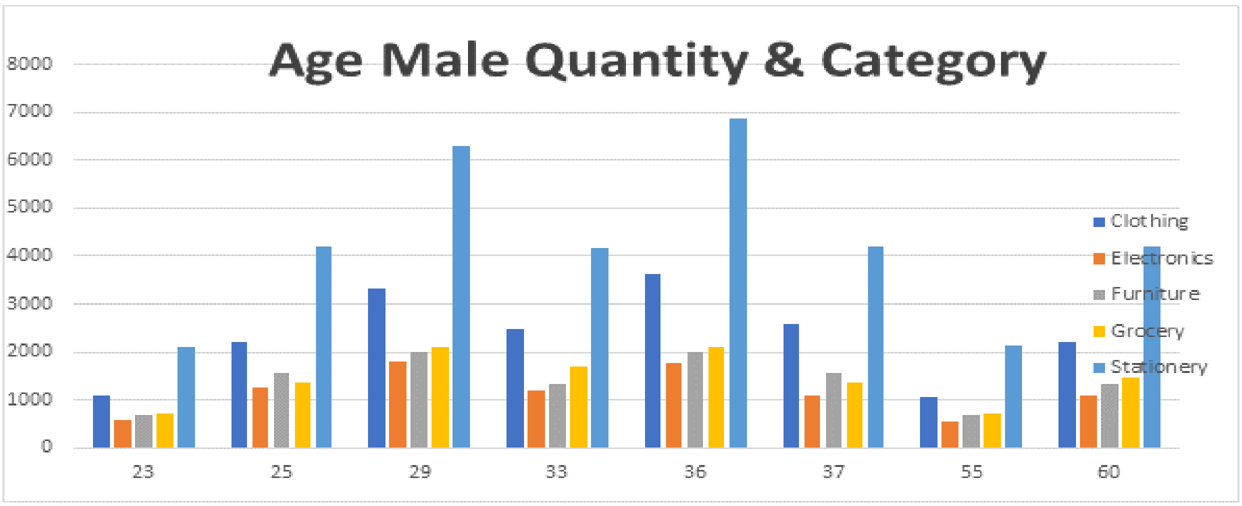


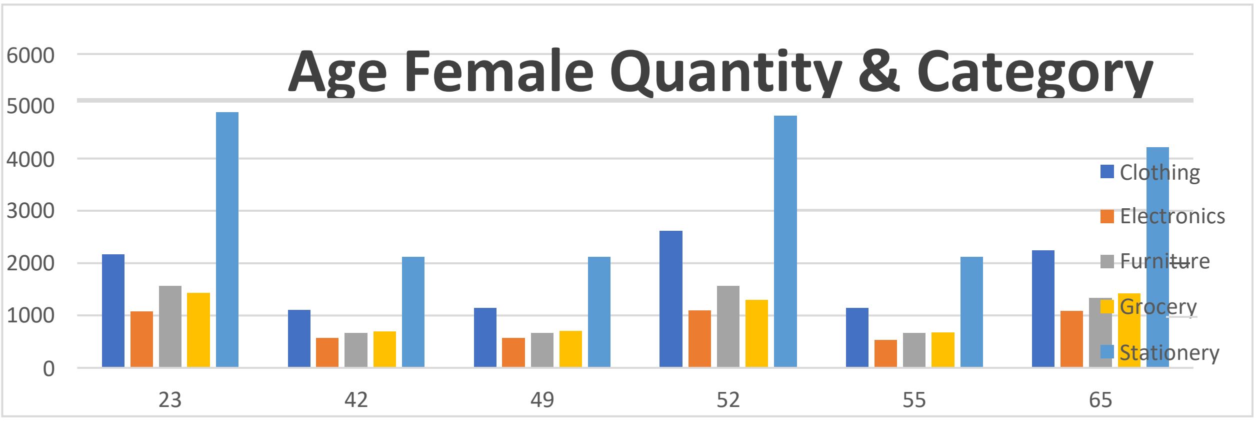
1. **Gender: Female (f)**
    - Female customers have accumulated a total purchase amount of $1,273,474.
2. **Gender: Male (m)**
    - Male customers exhibit a higher total purchase amount, contributing $2,422,227.

Conclusion:

Analyzing total purchase amounts by gender provides valuable insights into the spending patterns of male and female customers. The data suggests that male customers, with a total purchase amount of $2,422,227, have a higher overall spending compared to female customers, who have a total purchase amount of $1,273,474. This information can guide marketing and sales strategies to tailor approaches based on the preferences and behaviors of each gender, ensuring targeted and effective outreach.

## 4. Gender and Category Analysis





- Overall Sales Distribution:

  total quantity sold is higher for females (48,320) compared to males (84,732), indicating a substantial contribution from female customers across all product categories.

- Clothing Category:

  Females consistently show higher quantities sold in the Clothing category across different age groups.

  Males contribute significantly to the total quantity sold in Clothing but are consistently surpassed by females.

- Electronics Category:

  Males dominate the quantity sold in the Electronics category, with a grand total of 9,297, compared to 4,933 for females.

  The Electronics category is more popular among male customers.

- Furniture Category:

  The Furniture category sees a relatively balanced distribution between males and females.

  No significant dominance by either gender is observed in this category.

- Grocery Category:

  Females contribute more to the quantity sold in the Grocery category, with a total of 6,212 compared to 11,522 for males.

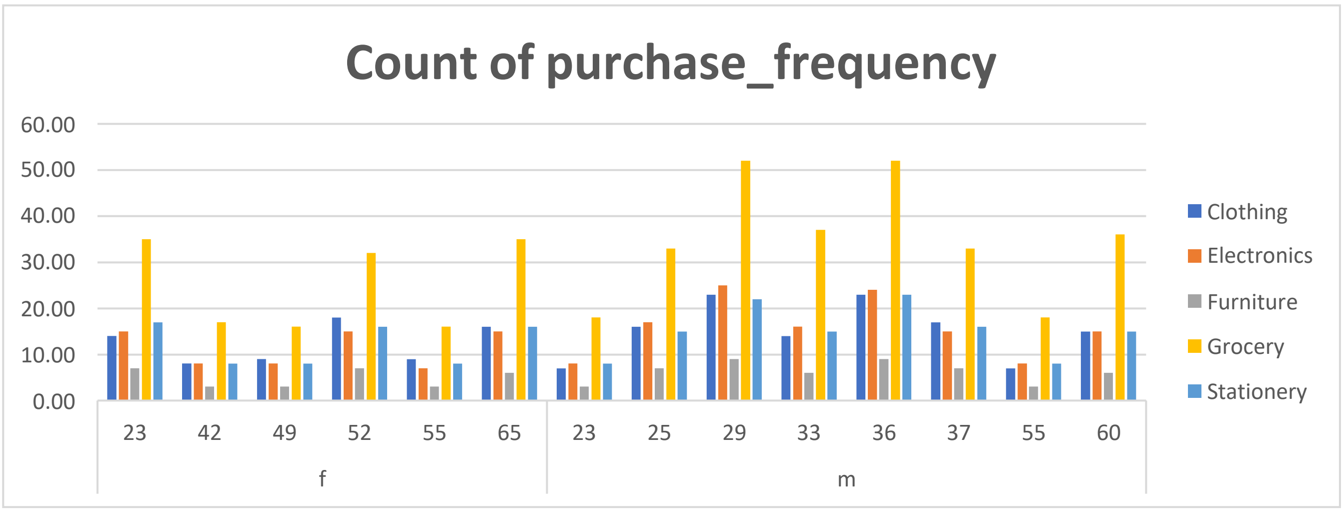
  The Grocery category is more popular among female customers.

- Stationery Category:

  Females consistently lead in the Stationery category across different age groups.

  Males contribute significantly but are consistently surpassed by females.

## 5. Age and Category Analysis



- Clothing Category

  Age 23 and 29: Both males and females at ages 23 and 29 demonstrate higher purchase frequencies in the Clothing category.

  Age 36: Males at age 36 also show increased interest in Clothing, indicating potential fashion-conscious customers in this age bracket.

- Electronics Category

  Age 29 and 36: Both males and females at ages 29 and 36 exhibit higher purchase frequencies in Electronics.

- Furniture Category

  Age 29 and 36: Both males and females at ages 29 and 36 display heightened interest in the Furniture category.

- Grocery Category

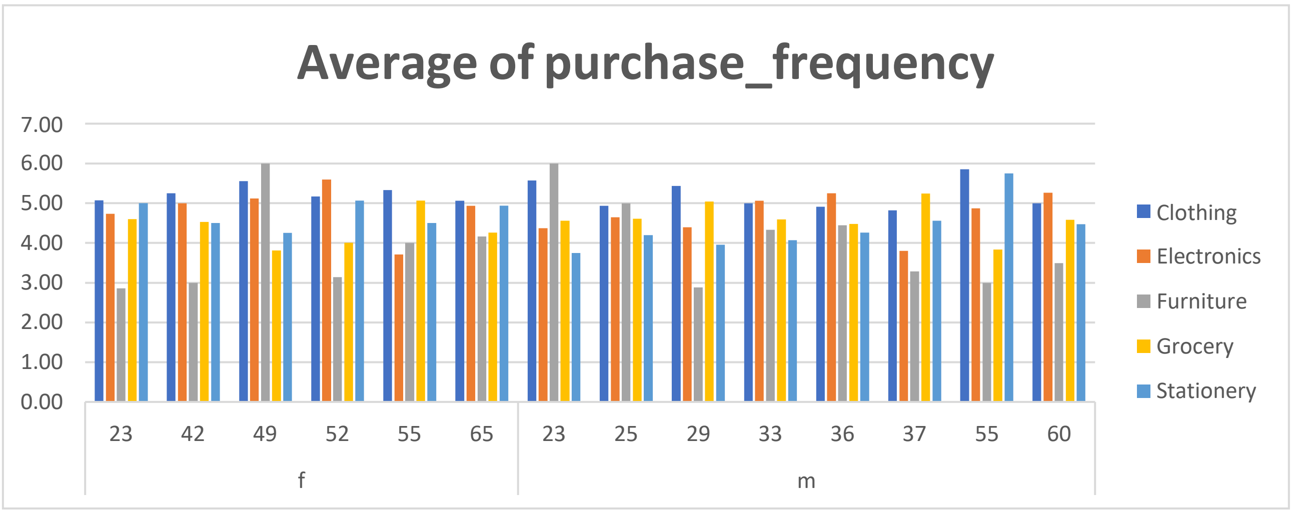
  Age 29 and 36: Both males and females at ages 29 and 36 demonstrate increased purchase frequencies in the Grocery category.

- Stationery Category

  Age 23, 29, and 60: Females at ages 23 and 60, and males at age 29, show elevated interest in the Stationery category.

## 6. Purchase Behavior

**- Average purchase frequency by age and gender:**



- Age 23 and 65:

  Females at age 23 and 65 display higher average purchase frequencies, indicating distinct shopping behaviours.

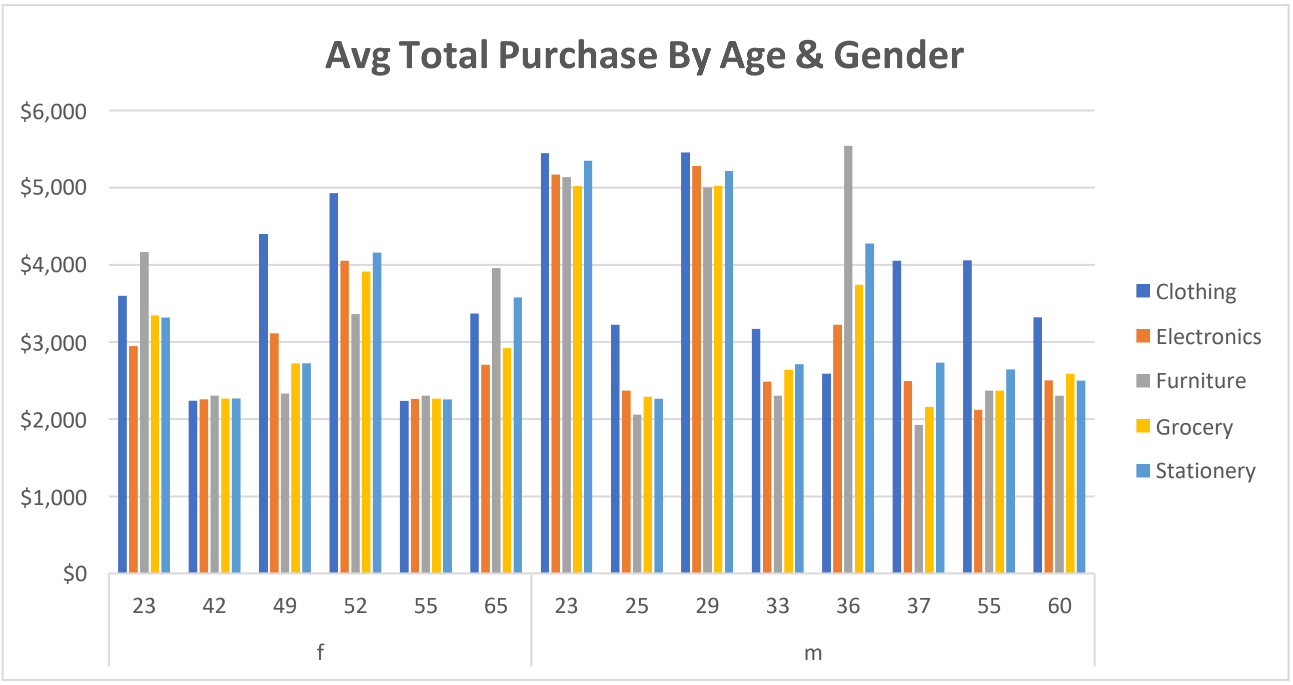
  Females consistently show higher average purchase frequencies in the Stationery category.

- Age 29 and 36:

  Males at ages 29 and 36 exhibit elevated average purchase frequencies, suggesting potential high-value customer segments during these life stages.

  Males consistently demonstrate higher average purchase frequencies in Electronics and Furniture categories.

**- Average total purchase amount by age and gender:**



- **Clothing:**

  The age group of 23 appears to have the highest spending on clothing with an average expenditure of $3,600. This is followed by the age group of 29 with an average expenditure of $5,456.

The age group of 42 has the lowest spending on clothing with an average expenditure of $2,239.

- **Electronics:**

The age group of 29 shows the highest spending on electronics with an average expenditure of $5,282. This is followed closely by the age group of 23 with an average expenditure of $5,169.

The age group of 60 has the lowest spending on electronics with an average expenditure of $2,503.

- **Furniture:**

The age group of 36 has the highest spending on furniture with an average expenditure of $5,544. This is followed by the age group of 23 with an average expenditure of $4,167.

The age group of 37 has the lowest spending on furniture with an average expenditure of $1,929.

- **Grocery:**

The age group of 65 has the highest spending on grocery items with an average expenditure of $3,958. This is followed by the age group of 52 with an average expenditure of $3,915.

The age group of 37 has the lowest spending on grocery items with an average expenditure of $2,161.

- **Stationery:**

The age group of 23 has the highest spending on stationery with an average expenditure of $5,350. This is followed by the age group of 29 with an average expenditure of $5,216.

The age group of 55 has the lowest spending on stationery with an average expenditure of $2,256.

- **Conclusion:**

The analysis of consumer spending patterns across different age groups reveals interesting insights. Certain age groups tend to spend more in specific categories compared to others. For example, the age group of 23 shows higher spending on clothing, furniture, electronics, and stationery, while the age group of 65 spends more on grocery items. These findings can be valuable for businesses and marketers in understanding consumer behavior and tailoring their strategies to target specific age groups effectively.

# 2. Time Series Forecasting:

## A) Time Series Analysis

Trends, Patterns, and Variations in Total Purchase Amounts:

- Monthly Variations:

  March 2020: Significant increases in Clothing, Electronics, and Grocery purchases.

  May 2021: Considerable spikes in Electronics and Grocery purchases.

- Quarterly Trends:

  Qtr2 2020: Noticeable dip in total purchase amounts, especially in Electronics and Furniture.

  Qtr3 2021: Grocery purchases peak across all three months.

- Seasonal Trends:

  Grocery Category: Shows seasonal peaks in March and May across multiple years, suggesting potential seasonal demand.

- Category-wise Performance:

  Electronics: Consistently contributes a substantial portion to the total purchase amount.

  Furniture and Stationery: Show relatively lower figures compared to other categories.

- Yearly Overview:

  2020 to 2022: Overall increasing trend in total purchase amounts across all categories, reaching a significant total of $3,695,701 by the end of 2022.

- Quarterly Total Purchase Amounts:

  Qtr4 2020: A notable increase in total purchase amounts, particularly in Grocery.

  Qtr2 2021: An overall dip in total purchase amounts, with a recovery in subsequent quarters.

- Monthly Patterns:

  Consistent Peaks: Some months consistently exhibit higher purchase amounts (e.g., March and May in various years).

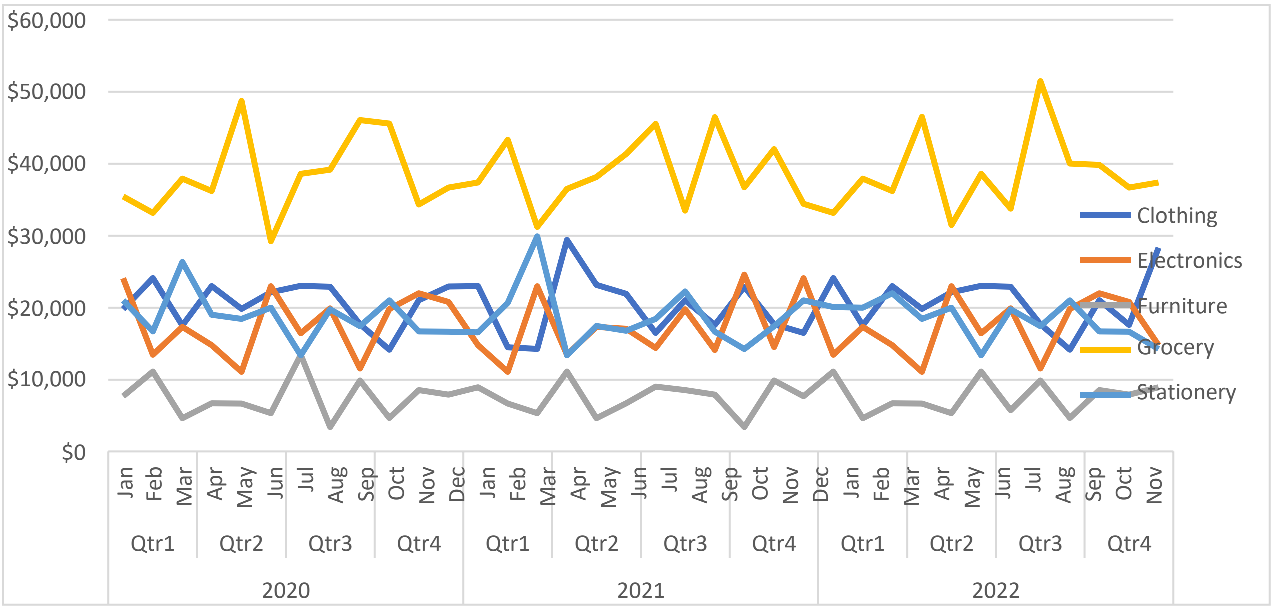
- Total Purchase Amount Distribution:

  Overall Scale: The grand total of $3,695,701 provides an understanding of the cumulative purchase amounts across all categories for the entire period.

- Potential Areas for Further Analysis:

  External Factors: Investigate external factors influencing fluctuations in purchase amounts.

  Promotions and Events: Explore the impact of promotions or events on monthly and quarterly variations.



## B) Quantitative Forecasting

- Monthly and Quarterly Overview:

  2020 Q1: Overall steady performance across categories, with varying quantities sold.

  2020 Q2: Consistent quantities sold, with a slight increase in June.

  2020 Q3: A spike in July, particularly in Clothing and Stationery.

  2020 Q4: December sees a notable increase in quantities sold across all categories.

- Yearly Trends:

  2020: Relatively stable throughout the year, with fluctuations in specific months.

  2021: Steady performance with occasional spikes, such as in January and July.

  2022: The quantity_sold shows a consistent increase, with June and December experiencing higher sales.

- Category-wise Performance:

  Clothing: Maintains a consistently high quantity sold, especially in peak months.

  Electronics: Steady performance, with occasional spikes in June and December.

  Furniture: Shows stability, with slight fluctuations in quantity_sold.

  Grocery: Steady sales throughout the year, with occasional increases.

  Stationery: Consistent performance, with noticeable spikes in certain months.

- Quarterly Patterns:

  Qtr2 2020: Stable quantities sold across all categories.

  Qtr3 2021: July experiences a significant increase in quantities sold across all categories.

  Qtr4 2022: Consistent growth in quantity_sold, particularly in December.

- Monthly Patterns:

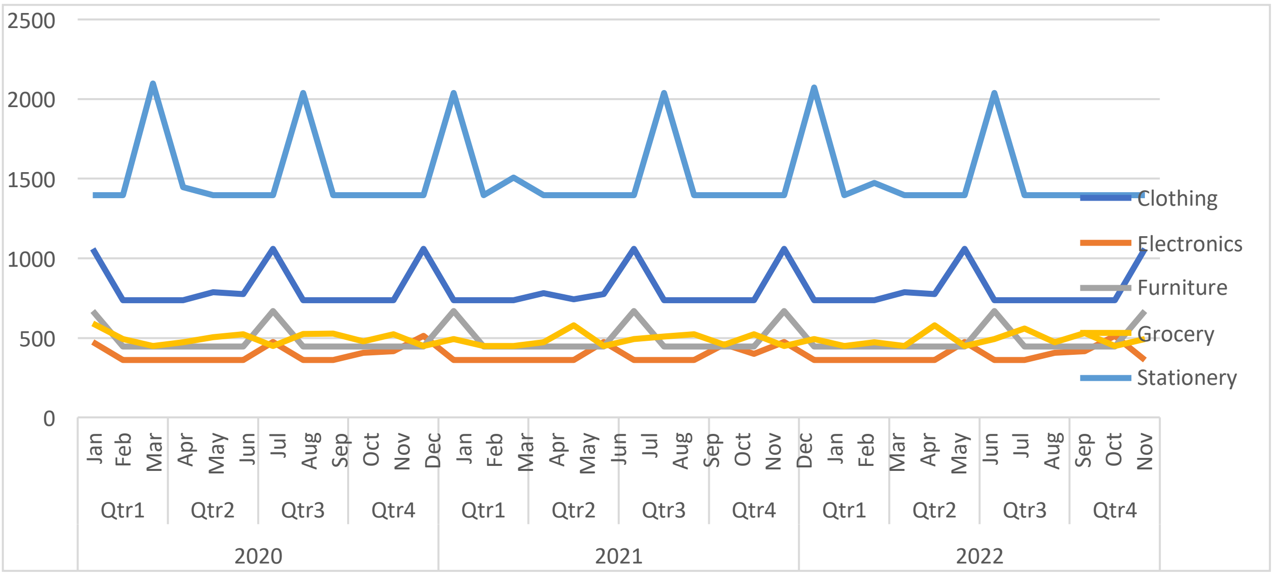
  December: Consistently high quantities sold across all years, indicating potential holiday-related demand.

  July: Frequently sees spikes in quantity_sold, especially in 2020 and 2021.

- Grand Total Analysis:

  The grand total quantity_sold for the entire period is 133,052, showcasing an overall positive trend in sales.

  Total quantity_sold for the entire period is 133,052 units. There is a trend of fluctuation in the quantity_sold across quarters, with variations observed in each year.



- **Quarterly Performance Analysis:**

  2020:

  Q1 (11714): A strong start to the year with high quantity_sold.

  Q2 (10462): A slight dip compared to Q1 but maintaining a substantial level of sales.

  Q3 (11624): A notable increase, reaching levels comparable to Q1.

  Q4 (10846): A slight decrease, concluding the year with a stable performance.

  2021:

  Q1 (11189): Similar to Q4 2020, maintaining a consistent level of sales.

  Q2 (10523): A moderate decrease from Q1.

  Q3 (11534): A significant increase, surpassing Q2 levels.

  Q4 (11047): A slight dip compared to Q3, ending the year on a stable note.

  2022:

  Q1 (10988): A relatively stable start to the year, maintaining a similar level as Q4 2021.

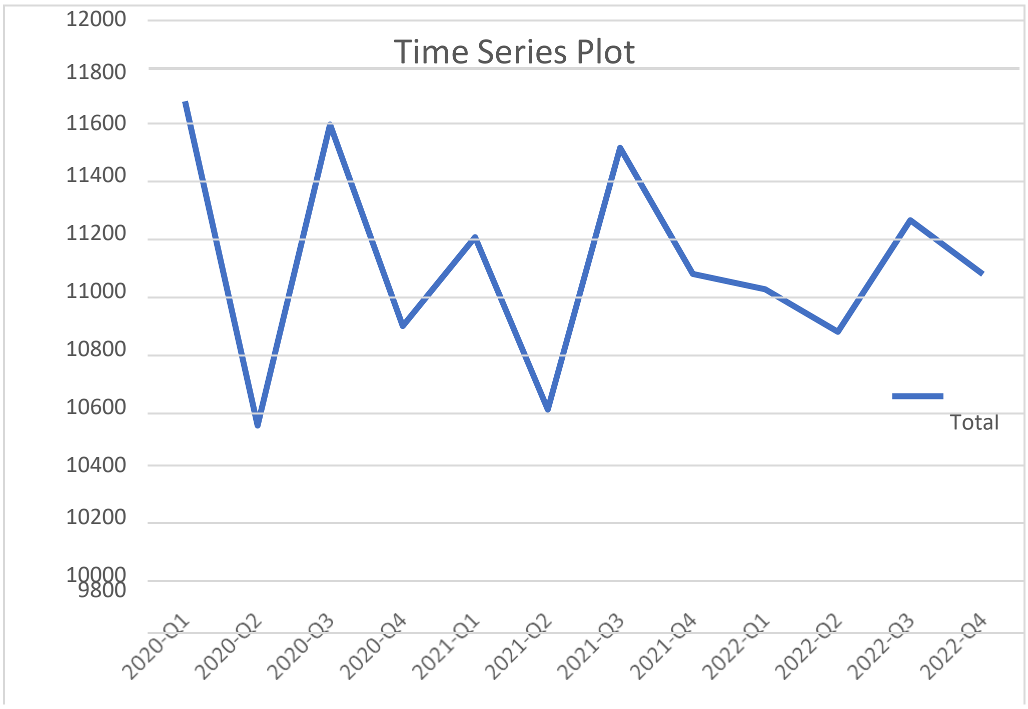
Q2 (10824): A slight decrease from Q1 but still within a consistent range.

Q3 (11255): An increase, surpassing Q2 levels.

Q4 (11046): Maintaining a stable performance, similar to Q2.

- **Yearly Performance Analysis:**

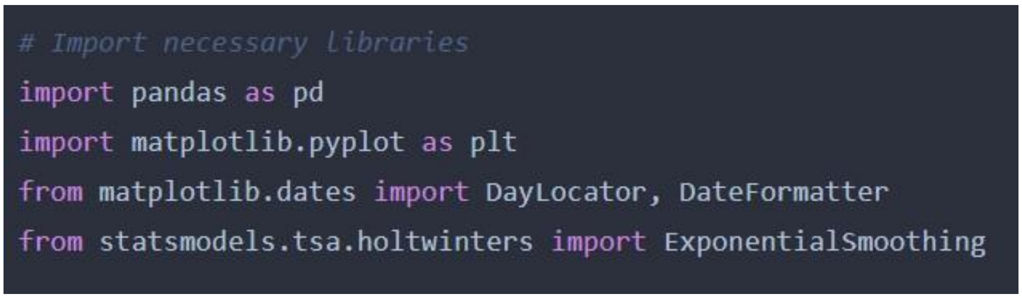
  2020 to 2022: While there are quarterly variations, the overall quantity_sold has remained within a relatively stable range, showcasing a balanced performance over the three years.



.

- **Quantitative Forecasting:**

  - Forecasting For Sales Price:



```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DayLocator, DateFormatter
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

  - This section imports the required libraries for data analysis, visualization, and time series forecasting using Holt-Winters Exponential Smoothing.

```python
# Read data from an Excel file into a Pandas DataFrame
df = pd.read_excel('Kako.xlsx', sheet_name='Last_update')

# Assuming your data has a 'date' column and a 'sales_price' column
df['date'] = pd.to_datetime(df['date'])
df = df.set_index('date')
```

- The code reads the data from an Excel file named 'Kako.xlsx' and selects the sheet named 'Last_update'. The data is stored in a Pandas DataFrame named df.
- Here, the 'date' column is converted to datetime format, and then the DataFrame's index is set to the 'date' column. This prepares the data for time series analysis.

```python
# Apply Exponential Smoothing
model = ExponentialSmoothing(df[' sales_price'], seasonal='add', seasonal_periods=30)
fit_model = model.fit()
```

- This section applies the Holt-Winters Exponential Smoothing model to the historical sales data. It uses a seasonal additive approach with a seasonal period of set to 30, as I mentioned that the Sales Price changes every 30 days, The model is then fitted to the historical data.

```python
# Predict future sales for the next 30 days
forecast_days = int(input("Enter the number of forecast days: "))
forecast = fit_model.forecast(steps=forecast_days)
```

- The code prompts the user to input the number of days for which they want to forecast sales.
- The model is used to predict future sales for the number of days specified by the user.

```python
# Convert the forecasted values to integers
forecast_int = forecast.astype(int)
```

- Convert the forecasted values to integers.

```python
# Plot the historical sales data and the forecasted values
plt.figure(figsize=(15, 7))
plt.plot(pd.date_range(start=df.index[-1] + pd.DateOffset(days=1), periods=forecast_days, freq='D'),
         forecast, label='Forecasted Sales', linestyle='solid', color='red')
plt.title(f'Forecasted Sales for the Next {forecast_days} Days')  # Corrected title formatting
plt.xlabel('Date')
plt.ylabel('Sales Price')
plt.legend()
```

- This part of the code creates a new plot displaying both the historical sales data and the forecasted values for the specified number of days. It adds labels and a legend for clarity.

```python
# Display forecasted values on the graph
for date, value in zip(pd.date_range(start=df.index[-1] + pd.DateOffset(days=1),
                                     periods=forecast_days, freq='D'), forecast):
    plt.text(date, value, f'{value:.2f}', color='blue')
```

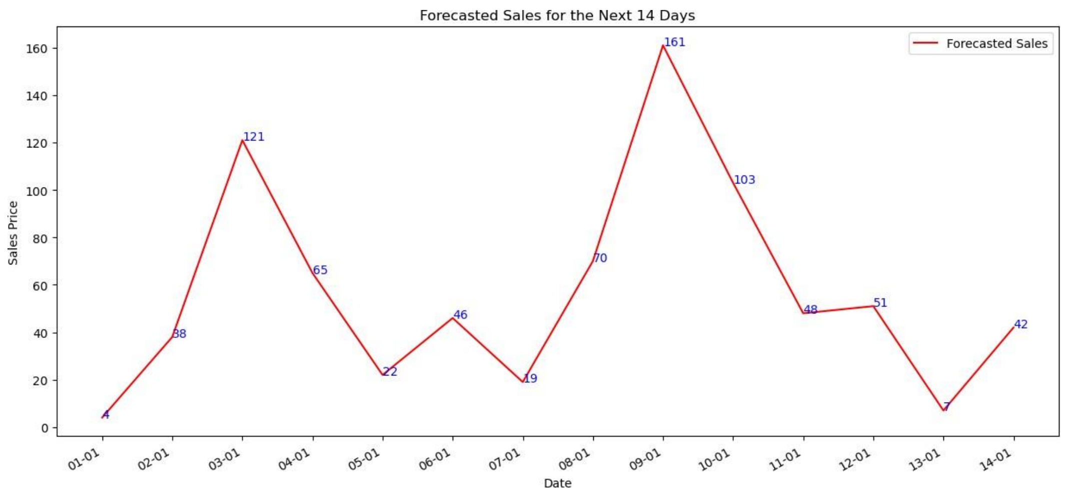
- Text annotations are added to the graph to display the forecasted values on their respective dates.

```python
plt.gca().xaxis.set_major_locator(DayLocator())
plt.gca().xaxis.set_major_formatter(DateFormatter('%d-%m'))
plt.gcf().autofmt_xdate()
```

- These lines set the major locator and formatter for the x-axis to improve the date formatting and readability of the plot.

```python
plt.show()
print(forecast)
```

- This line displays the final plot to the user.
- The code prints the forecasted values for the specified number of days to the console for reference.

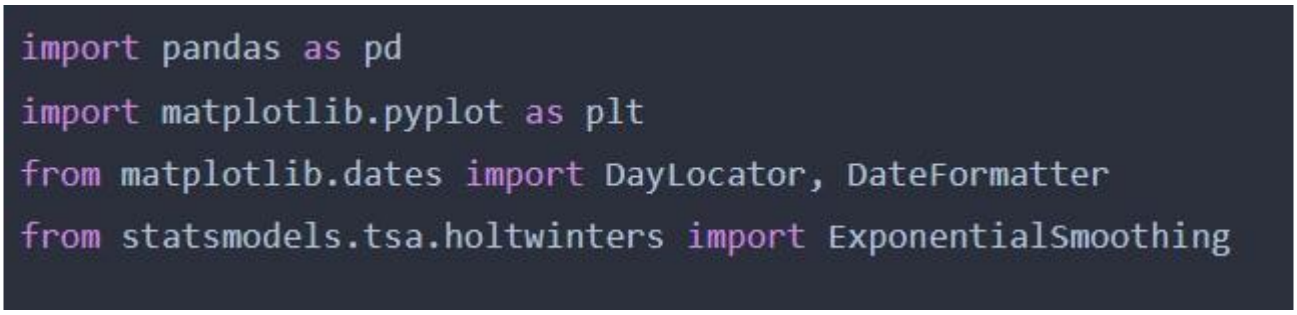


- **Here, forecasted for 14 days.**

- **Output:**

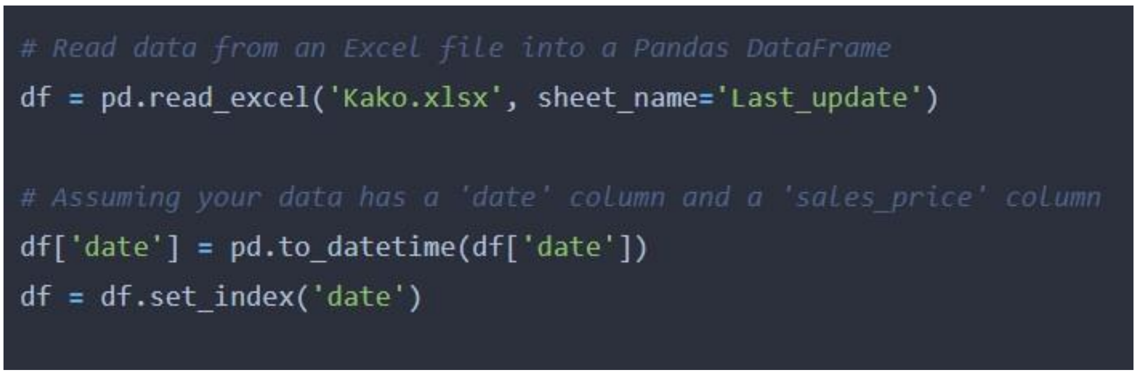
```
2023-01-01    4      2023-01-02    38
2023-01-03   121     2023-01-04    65
2023-01-05    22     2023-01-06    46
2023-01-07    19     2023-01-08    70
2023-01-09   161     2023-01-10   103
2023-01-11    48     2023-01-12    51
2023-01-13    7      2023-01-14    42
```

- **Forecasting For Quantity Sold:**



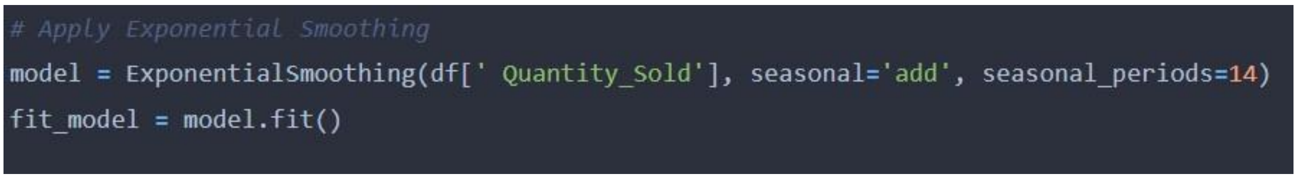
```python
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.dates import DayLocator, DateFormatter
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

- Import necessary libraries: pandas for data manipulation, matplotlib for plotting, and ExponentialSmoothing from statsmodels for Holt-Winters exponential smoothing.



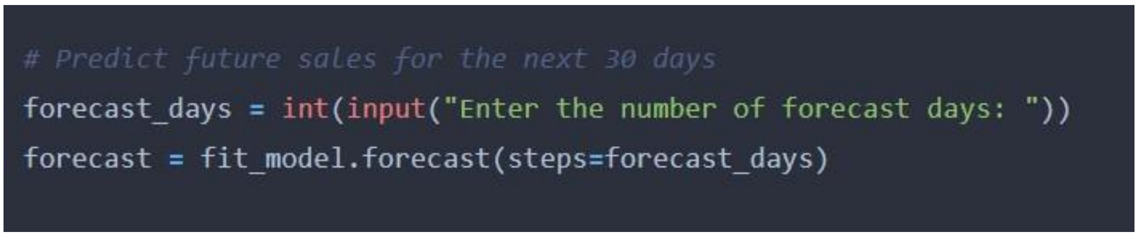
```python
# Read data from an Excel file into a Pandas DataFrame
df = pd.read_excel('Kako.xlsx', sheet_name='Last_update')

# Assuming your data has a 'date' column and a 'sales_price' column
df['date'] = pd.to_datetime(df['date'])
df = df.set_index('date')
```

- The code reads the data from an Excel file named 'Kako.xlsx' and selects the sheet named 'Last_update'. The data is stored in a Pandas DataFrame named df.
- Here, the 'date' column is converted to datetime format, and then the DataFrame's index is set to the 'date' column. This prepares the data for time series analysis.



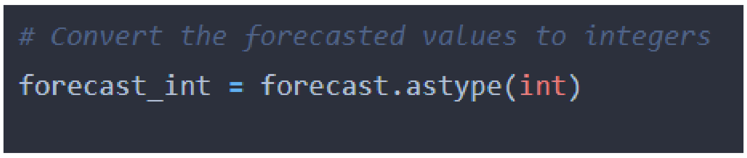
```python
# Apply Exponential Smoothing
model = ExponentialSmoothing(df[' Quantity_Sold'], seasonal='add', seasonal_periods=14)
fit_model = model.fit()
```

- Create an exponential smoothing model using the Holt-Winters method with an additive seasonality. The seasonal periods is set to 14, as I mentioned that the quantity sold changes every 14 days. The model is then fitted to the historical data.



```python
# Predict future sales for the next 30 days
forecast_days = int(input("Enter the number of forecast days: "))
forecast = fit_model.forecast(steps=forecast_days)
```

- Take user input for the number of days to forecast.
- Use the fitted model to forecast the quantity sold for the specified number of days.



```python
# Convert the forecasted values to integers
forecast_int = forecast.astype(int)
```

- Convert the forecasted values to integers.

```python
# Plot the historical sales data and the forecasted values
plt.figure(figsize=(15, 7))
plt.plot(pd.date_range(start=df.index[-1] + pd.DateOffset(days=1), periods=forecast_days, freq='D'),
         forecast_int, label='Forecasted Quantity Sold', linestyle='solid', color='red')

plt.title(f'Forecasted Quantity Sold for the Next {forecast_days} Days')
plt.xlabel('Date')
plt.ylabel('Quantity Sold')
plt.legend()
```

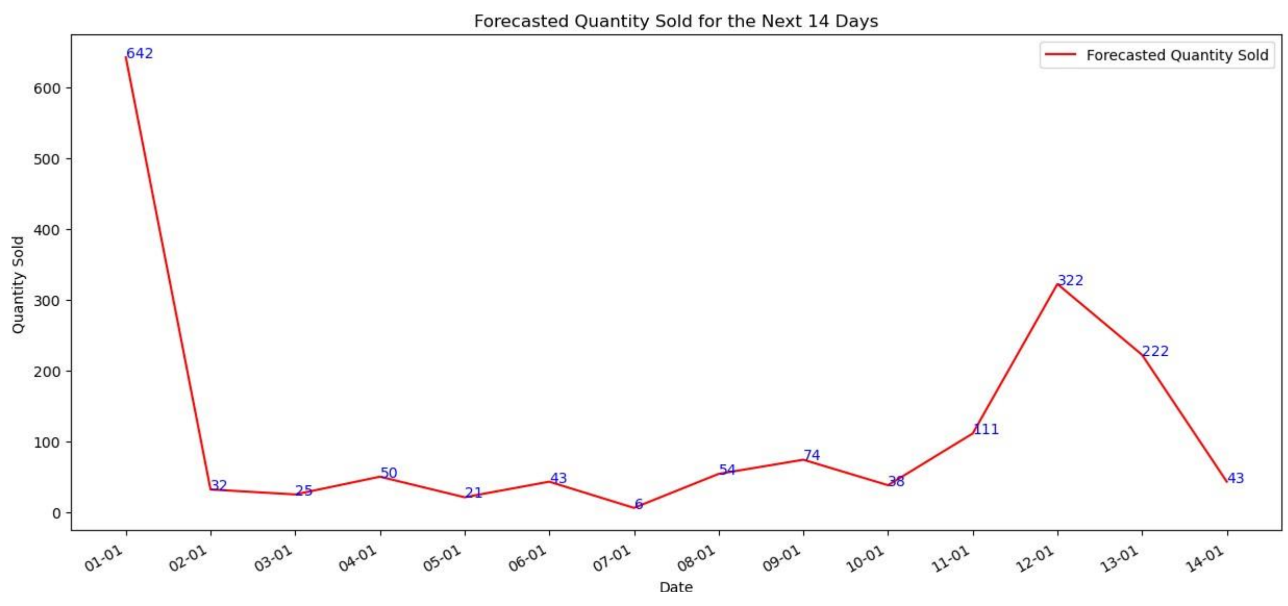
- Create a plot with historical data and forecasted values. The x-axis represents the date, and the y-axis represents the forecasted quantity sold.
- Set the title and labels for the plot.

```python
# Display forecasted values on the graph
for date, value in zip(pd.date_range(start=df.index[-1] + pd.DateOffset(days=1),
                                     periods=forecast_days, freq='D'), forecast_int):
    plt.text(date, value, str(value), color='blue')
```

- Display the forecasted values on the graph using blue text.

```python
plt.gca().xaxis.set_major_locator(DayLocator())
plt.gca().xaxis.set_major_formatter(DateFormatter('%d-%m'))
plt.gcf().autofmt_xdate()

plt.show()
print(forecast_int)
```

- Adjust the formatting of the x-axis to show the day and month, and rotate the dates for better visibility.
- Display the plot and print the forecasted values.

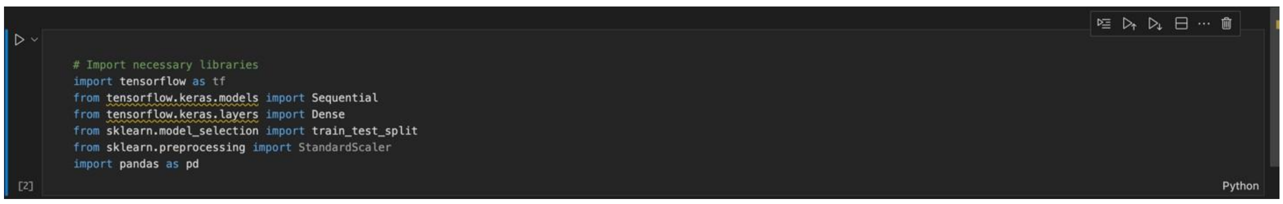


- **Here, forecasted for 14 days.**

- **Output:**

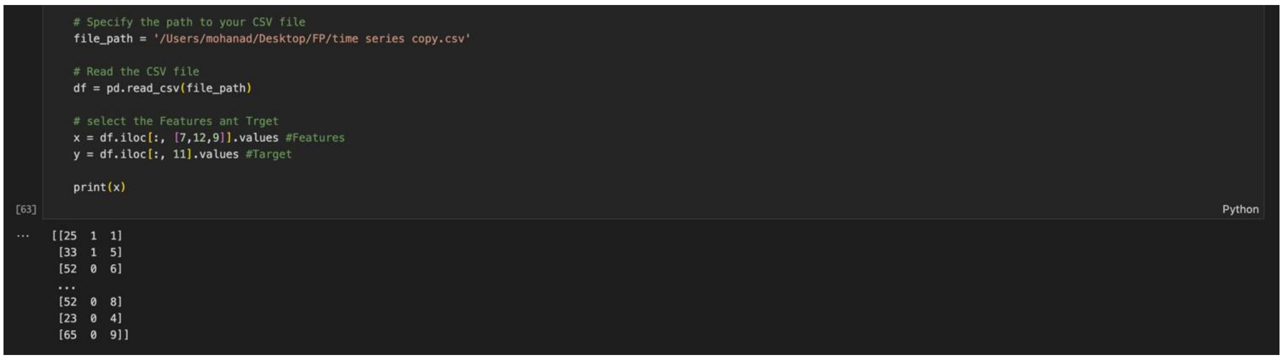
| | | | |
|---|---|---|---|
| 2023-01-01 | 642 | 2023-01-02 | 32 |
| 2023-01-03 | 25 | 2023-01-04 | 50 |
| 2023-01-05 | 21 | 2023-01-06 | 43 |
| 2023-01-07 | 6 | 2023-01-08 | 54 |
| 2023-01-09 | 74 | 2023-01-10 | 38 |
| 2023-01-11 | 111 | 2023-01-12 | 322 |
| 2023-01-13 | 222 | 2023-01-14 | 43 |

## 3. Customer Segmentation:



```python
# Import necessary libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

**-This cell contains important libraries for code.**

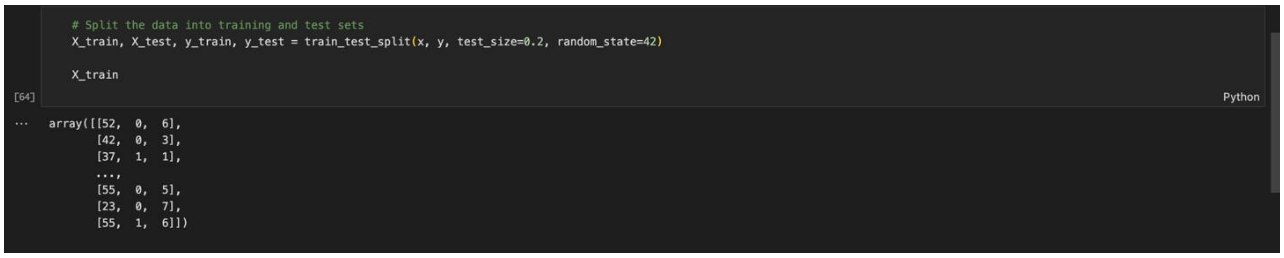


```
# Specify the path to your CSV file
file_path = '/Users/mohanad/Desktop/FP/time series copy.csv'

# Read the CSV file
df = pd.read_csv(file_path)

# select the Features ant Trget
x = df.iloc[:, [7,12,9]].values #Features
y = df.iloc[:, 11].values #Target

print(x)
```

[63] Python

```
[[25  1  1]
 [33  1  5]
 [52  0  6]
 ...
 [52  0  8]
 [23  0  4]
 [65  0  9]]
```

**-Here we identify the data set and use the (age, gender, purchase frequency) as a Features and (Category) as Target.**

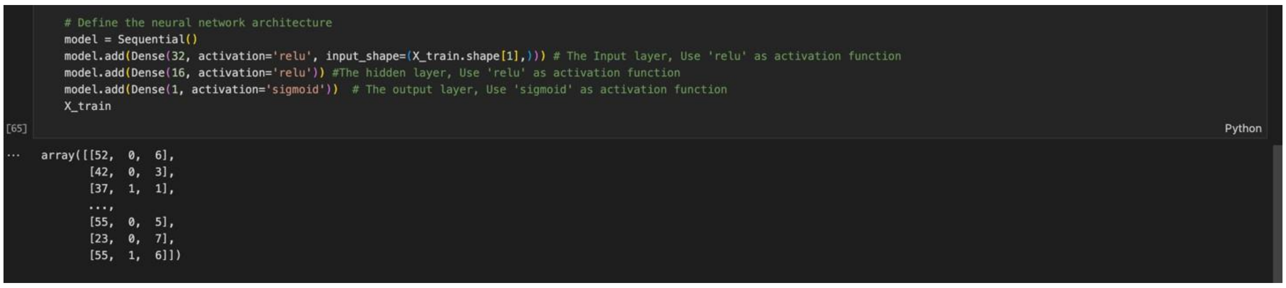


```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

X_train
```

[64] Python

```
array([[52,  0,  6],
       [42,  0,  3],
       [37,  1,  1],
       ...,
       [55,  0,  5],
       [23,  0,  7],
       [55,  1,  6]])
```

**-We split the data into train and test sets.**

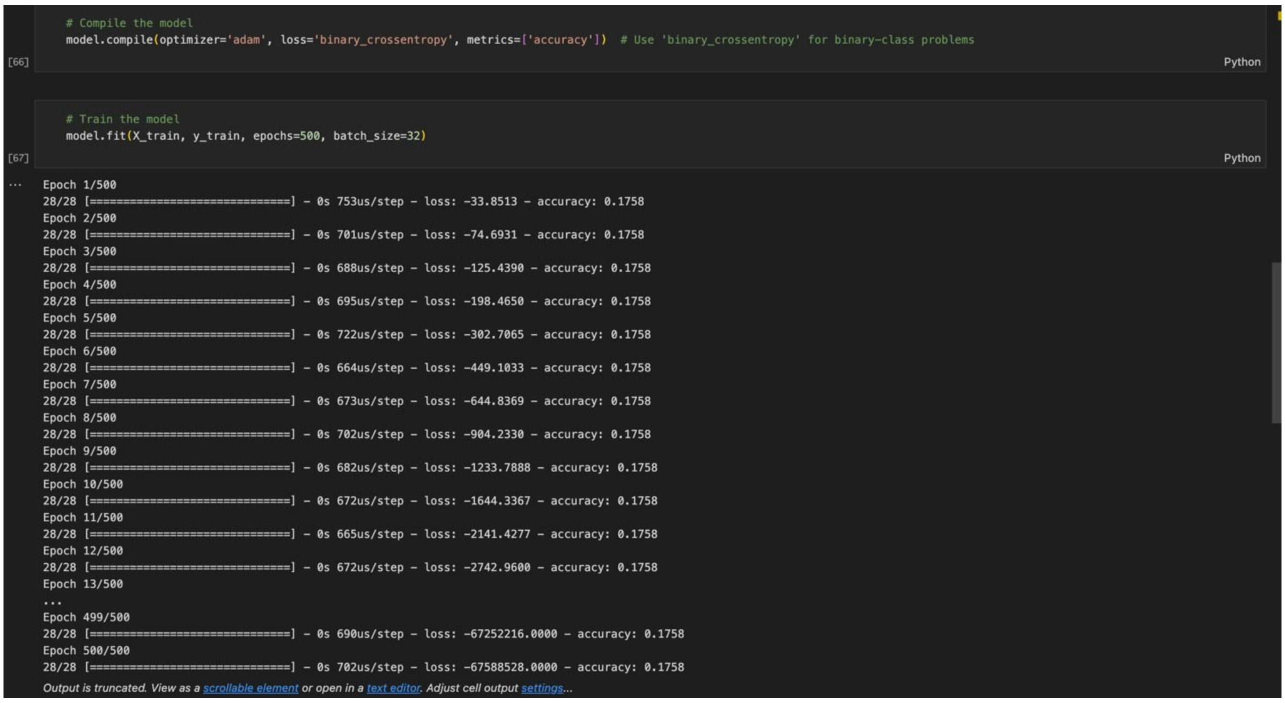


```
# Define the neural network architecture
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(X_train.shape[1],))) # The Input layer, Use 'relu' as activation function
model.add(Dense(16, activation='relu')) #The hidden layer, Use 'relu' as activation function
model.add(Dense(1, activation='sigmoid'))  # The output layer, Use 'sigmoid' as activation function
X_train
```

[65] Python

```
array([[52,  0,  6],
       [42,  0,  3],
       [37,  1,  1],
       ...,
       [55,  0,  5],
       [23,  0,  7],
       [55,  1,  6]])
```

**-We build the Neural Network model with Input layer, using 'relu' as activation function,hidden layer, UsIng 'relu' as activation function and utput layer, Using 'sigmoid' as activation function.**

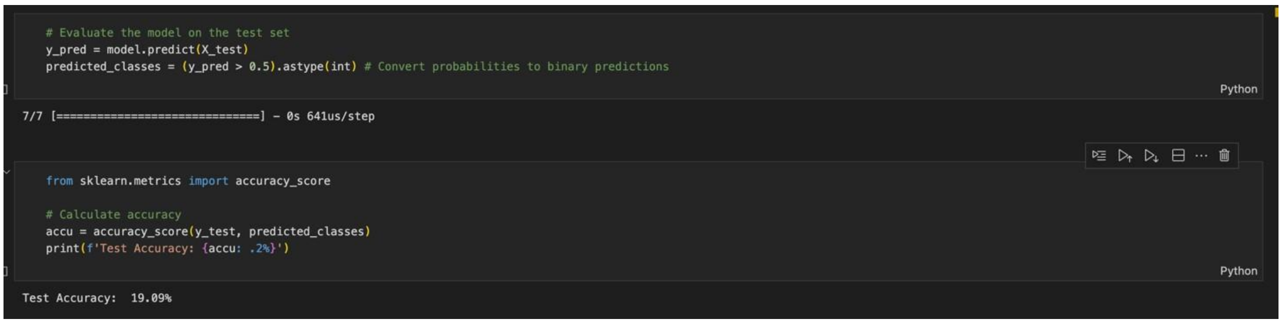


```
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])  # Use 'binary_crossentropy' for binary-class problems
```

```
# Train the model
model.fit(X_train, y_train, epochs=500, batch_size=32)
```

```
Epoch 1/500
28/28 [==============================] - 0s 753us/step - loss: -33.8513 - accuracy: 0.1758
Epoch 2/500
28/28 [==============================] - 0s 701us/step - loss: -74.6931 - accuracy: 0.1758
Epoch 3/500
28/28 [==============================] - 0s 688us/step - loss: -125.4390 - accuracy: 0.1758
Epoch 4/500
28/28 [==============================] - 0s 695us/step - loss: -198.4650 - accuracy: 0.1758
Epoch 5/500
28/28 [==============================] - 0s 722us/step - loss: -302.7065 - accuracy: 0.1758
Epoch 6/500
28/28 [==============================] - 0s 664us/step - loss: -449.1033 - accuracy: 0.1758
Epoch 7/500
28/28 [==============================] - 0s 673us/step - loss: -644.8369 - accuracy: 0.1758
Epoch 8/500
28/28 [==============================] - 0s 702us/step - loss: -904.2330 - accuracy: 0.1758
Epoch 9/500
28/28 [==============================] - 0s 682us/step - loss: -1233.7888 - accuracy: 0.1758
Epoch 10/500
28/28 [==============================] - 0s 672us/step - loss: -1644.3367 - accuracy: 0.1758
Epoch 11/500
28/28 [==============================] - 0s 665us/step - loss: -2141.4277 - accuracy: 0.1758
Epoch 12/500
28/28 [==============================] - 0s 672us/step - loss: -2742.9600 - accuracy: 0.1758
Epoch 13/500
...
Epoch 499/500
28/28 [==============================] - 0s 690us/step - loss: -67252216.0000 - accuracy: 0.1758
Epoch 500/500
28/28 [==============================] - 0s 702us/step - loss: -67588528.0000 - accuracy: 0.1758
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

**-We start to compile the model using ‘adam’ as optimizer and 'binary_crossentropy’ As loss function Then we trained the model with 500 epochs and 32 batch size.**



```
# Evaluate the model on the test set
y_pred = model.predict(X_test)
predicted_classes = (y_pred > 0.5).astype(int) # Convert probabilities to binary predictions
```

```
7/7 [==============================] - 0s 641us/step
```

```
from sklearn.metrics import accuracy_score

# Calculate accuracy
accu = accuracy_score(y_test, predicted_classes)
print(f'Test Accuracy: {accu: .2%}')
```

```
Test Accuracy:  19.09%
```

**-We evaluate the model on test set and Convert probabilities to binary predictions, Then we calculate the accuracy of the model which turned out 19% and with more epochs and hidden layers we can improve it.**