**UNIVARIATE ANALYSIS CODES  
  
  
1)BATTERY**

df = pd.DataFrame(data, columns=['Battery '])

# Frequency Distribution

frequency = df['Battery '].value\_counts()

# Proportions

proportions = df['Battery '].value\_counts(normalize=True) \* 100

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of Battery')

plt.xlabel('Battery ')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**2)CAMERA**

df = pd.DataFrame(data, columns=['camera'])

# Frequency Distribution

frequency = df['camera'].value\_counts()

# Proportions

proportions = df['camera'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency camera')

plt.xlabel('camera')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**3)DISPLAY**

df = pd.DataFrame(data, columns=['display'])

# Frequency Distribution

frequency = df['display'].value\_counts()

# Proportions

proportions = df['display'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of display')

plt.xlabel('display')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**4)MEMORY**

df = pd.DataFrame(data, columns=['memory'])

# Frequency Distribution

frequency = df['memory'].value\_counts()

# Proportions

proportions = df['memory'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of memory')

plt.xlabel('memory')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**5)NAME**

df = pd.DataFrame(data, columns=['name'])

# Frequency Distribution

frequency = df['name'].value\_counts()

# Proportions

proportions = df['name'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of name')

plt.xlabel('name')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**6)PROCESSOR**

df = pd.DataFrame(data, columns=['processor'])

# Frequency Distribution

frequency = df['processor'].value\_counts()

# Proportions

proportions = df['processor'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of processor')

plt.xlabel('processor')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**7)RATING**

df = pd.DataFrame(data, columns=['rating'])

# Frequency Distribution

frequency = df['rating'].value\_counts()

# Proportions

proportions = df['rating'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of rating')

plt.xlabel('rating')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**8)REVIEWS**

df = pd.DataFrame(data, columns=['reviews'])

# Frequency Distribution

frequency = df['reviews'].value\_counts()

# Proportions

proportions = df['reviews'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of reviews')

plt.xlabel('reviews')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**9) WARRANTY**

df = pd.DataFrame(data, columns=['warranty'])

# Frequency Distribution

frequency = df['warranty'].value\_counts()

# Proportions

proportions = df['warranty'].value\_counts(normalize=True) \* 100

# Display frequency and proportion tables

print("Frequency Distribution:\n", frequency)

print("\nProportion of Each Category:\n", proportions)

# Visualization

plt.figure(figsize=(10, 6))

frequency.plot(kind='bar', color='skyblue')

plt.title('Frequency Distribution of warranty')

plt.xlabel('warranty')

plt.ylabel('Frequency')

plt.xticks(rotation=90)

plt.show()

**10)PRICE**

df = pd.DataFrame(data, columns=['Price'])

# Descriptive Statistics

mean\_value = df['Price'].mean()

median\_value = df['Price'].median()

mode\_value = df['Price'].mode()[0] # Mode can have multiple values, so we take the first one

std\_dev = df['Price'].std()

min\_value = df['Price'].min()

max\_value = df['Price'].max()

quartiles = df['Price'].quantile([0.25, 0.5, 0.75])

# Display Descriptive Statistics

print("Mean:", mean\_value)

print("Median:", median\_value)

print("Mode:", mode\_value)

print("Standard Deviation:", std\_dev)

print("Minimum Value:", min\_value)

print("Maximum Value:", max\_value)

print("Quartiles:\n", quartiles)

# Visualizations

plt.figure(figsize=(12, 6))

# Histogram

plt.subplot(1, 2, 1)

plt.hist(df['Price'], bins=10, color='skyblue', edgecolor='black')

plt.title('Histogram of Prices')

plt.xlabel('Price')

plt.ylabel('Frequency')

# Box Plot

plt.subplot(1, 2, 2)

sns.boxplot(df['Price'], color='lightgreen')

plt.title('Box Plot of Prices')

plt.tight\_layout()

plt.show()