```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import seaborn as sns
#%%-------------------------Q1-----------------------
url = 'https://raw.githubusercontent.com/rjafari979/Information-Visualization-Data-
Analytics-Dataset-/main/stock%20prices.csv'
df = pd.read_csv(url)
missing_features = df.isna().sum()
if missing_features.sum() > 0:
    print("Yes, the dataset has missing values.")
    print("Total number of missing values after cleaning :",
df.isnull().sum().sum())
    print("Missing features and the number of missing entries:")
    print(missing_features[missing_features > 0])
else:
    print("No missing values.")
df.fillna(df.mean(numeric_only=True), inplace=True)
print("Replacing missing values with mean..")
missing_entries_after_cleaning = df.isna().sum()
print("Total number of missing values after cleaning :", df.isnull().sum().sum())
print("Testing missing features and the number of missing entries after cleaning:")
print(missing_entries_after_cleaning)
if missing_entries_after_cleaning.sum() == 0:
    print("\nAll missing values have been filled. The dataset is now clean.")
else:
    print("\nSome missing values still remain.")

#%%-----------------------Q2-----------------------
print("Total number of unique companies listed in the dataset are:",
df['symbol'].nunique())
print("List of unique companies:")
print(df['symbol'].unique())
qualitative_features = []
quantitative_features = []
for columns in df.columns:
    if df[columns].dtypes == 'object':
        qualitative_features.append(columns)
    else:
        quantitative_features.append(columns)
print("Qualitative Features: ",qualitative_features)
print("Quantitative Features: ",quantitative_features)
filtered_df = df[df['symbol'].isin(['GOOGL','AAPL'])]
print(filtered_df.head())
filtered_df.loc[:, 'date'] = pd.to_datetime(filtered_df['date'])
plt.figure(figsize=(12,8))
google_df= filtered_df[ filtered_df['symbol'] == 'GOOGL']
plt.plot(google_df['date'], google_df['close'], label='GOOGL', color='orange')
apple_df= filtered_df[ filtered_df['symbol'] == 'AAPL']
plt.plot(apple_df['date'], apple_df['close'], label='AAPL', color='blue')
plt.xlabel('date', fontsize=14)
plt.ylabel('USD($)', fontsize=14)
plt.title('Google and Apple Stock Closing Value Comparision', fontsize=16)
plt.xticks(rotation=45)
plt.gca().xaxis.set_major_locator(plt.MaxNLocator(6))
plt.legend()
plt.tight_layout()
plt.show()
```

```
#%%-----------------------Q3----------------------
aggregated_df=df.groupby('symbol').sum(numeric_only=True).reset_index()
original_df_row_count=len(df)
aggregated_df_row_count=len(aggregated_df)
print("Number of observations in cleaned dataset: ")
print(original_df_row_count)
print("Number of observations in aggregated dataset: ")
print(aggregated_df_row_count)
print("Displaying first 5 rows of aggregated dataset: ")
print(aggregated_df.head(5))

#%%--------------------Q4--------------------------
sliced_df = df[['symbol','close','volume']]
aggregated_sliced_df = sliced_df.groupby('symbol').agg({'close':
['mean','var'],'volume':['mean','var']}).reset_index()
aggregated_sliced_df.columns
=['symbol','close_mean','close_var','volume_mean','volume_var']
max_variance_index = np.argmax(aggregated_sliced_df['close_var'])
max_variance_symbol = aggregated_sliced_df.iloc[max_variance_index]['symbol']
max_variance_value = np.max(aggregated_sliced_df['close_var'])
print("Company that has maximum variance in the closing cost: ",
max_variance_symbol)
print(f"Variance of {max_variance_symbol} company is:
",max_variance_value.round(3))

#%%--------------------Q5----------------------------
df['date'] = pd.to_datetime(df['date'])
df_google_2015 = df[ (df['symbol'] == 'GOOGL') & (df['date'] > '2015-01-01')]
[['date','close']]
print("Google closing stock after 2015-01-01: ")
print(df_google_2015.head())

#%%--------------------Q6----------------------------
rolling_window_size=30
plt.figure(figsize=(12, 6))
df_rolling_mean = df_google_2015['close'].rolling(window=rolling_window_size,
center=True).mean()
plt.plot(df_google_2015.index, df_google_2015['close'], label='Close')
plt.plot(df_rolling_mean.index, df_rolling_mean,label='30-Day Rolling Mean',
color='orange')
plt.xlabel('date')
plt.ylabel('USD($)')
plt.title('Google closing stock price after jan 2015 versus Rolling window')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
roll_obs_missed = df_rolling_mean.isna().sum()
print(f"Number of observations missed with a rolling window of
{rolling_window_size} days: ", roll_obs_missed)

#%%-----------------------Q7--------------------
bin_labels=['very low','low','normal','high','very high']
df_google_2015['price_category'] = pd.cut(
    df_google_2015['close'],
    bins=5,
    labels=bin_labels
)
```

```
print("Created dataset with price_category:\n")
print(df_google_2015.to_string())
plt.figure(figsize=(12, 8))
custom_palette = {
    'very low': '#1f77b4',
    'low': '#ff7f0e',
    'normal': '#2ca02c',
    'high': '#d62728',
    'very high': '#9467bd'
}

sns.countplot(x='price_category',hue='price_category',
data=df_google_2015,palette=custom_palette,order=bin_labels)
plt.xlabel('Price Category')
plt.ylabel('Count')
plt.title('Equal width discretization')
plt.tight_layout()
plt.show()

#%%----------------------Q8-------------------
plt.figure(figsize=(12, 8))
plt.hist(df_google_2015['close'],bins=5,color='skyblue',edgecolor='black')
plt.xlabel('Close Value')
plt.ylabel('Frequency')
plt.title('Histogram of Close price')
plt.tight_layout()
plt.show()

#%%-----------------------Q9-----------------------
bin_labels=['very low','low','normal','high','very high']
df_google_2015['price_category'] = pd.qcut(
    df_google_2015['close'],
    q=5,
    labels=bin_labels
)
print("Created dataset with price_category:\n")
print(df_google_2015.to_string())
plt.figure(figsize=(12, 8))
custom_palette = {
    'very low': '#1f77b4',
    'low': '#ff7f0e',
    'normal': '#2ca02c',
    'high': '#d62728',
    'very high': '#9467bd'
}
sns.countplot(x='price_category',hue='price_category',
data=df_google_2015,palette=custom_palette,order=bin_labels)
plt.xlabel('Price Category')
plt.ylabel('Count')
plt.title('Count plot of Equal Frequency Binning')
plt.tight_layout()
plt.show()

#%%-------------------------Q10-----------------
df_google_2015 = df[(df['symbol'] == 'GOOGL') & (df['date'] > '2015-01-01')]
df_numeric_google_2015 = df_google_2015.select_dtypes(include=['number'])
mean_values = df_numeric_google_2015.mean()
num_features = len(mean_values)
covariance_matrix = [[0] * num_features for _ in range(num_features)]
```

```
for i in range(num_features):
    for j in range(num_features):
        covariance_of_each = (((df_numeric_google_2015.iloc[:, i] -
mean_values.iloc[i]) * (df_numeric_google_2015.iloc[:, j] -
mean_values.iloc[j])).sum()
                              /(len(df_numeric_google_2015) - 1))
        covariance_matrix[i][j] = covariance_of_each
covariance_df = pd.DataFrame(covariance_matrix,
columns=df_numeric_google_2015.columns,index=df_numeric_google_2015.columns)
covariance_df= covariance_df.round(3)
print("Covariance Matrix:")
print(covariance_df)

#%%-------------------------Q11---------------
covariance_matrix = df_google_2015.cov(numeric_only=True)
covariance_matrix = covariance_matrix.round(3)
print("Covariance Matrix:")
print(covariance_matrix)
```