Question 1: Develop a python program that loads the 'stock prices.csv' dataset from the course GitHub using the Pandas package. [6pts]

Code:

```
#-------------------------------------------------------------------------------------------------------------------------
#                                                Question 1
#-------------------------------------------------------------------------------------------------------------------------
df = pd.read_csv('https://raw.githubusercontent.com/rjafari979/Information-Visualization-Data-Analytics-Dataset-/main/stock%2

print(df.isna().sum())
df['open'].fillna(df['open'].mean(), inplace=True)
df['high'].fillna(df['high'].mean(), inplace=True)
df['low'].fillna(df['low'].mean(), inplace=True)

print("After filling the null values the total number of null values are:")
print(df.isna().sum())
print(df.isna().sum().sum())
if df.isna().sum().sum()==0:
    print("Missing value are fixed")
else:
    print("Missing values are not fixed")
```

a. Does the dataset contain missing entries? Display the missing features and the number of missing entries on the console.

=> Yes, the dataset contains 27 missing values.

b. Replace the missing values with 'mean'.

c. Run a test to check that all missing values are filled and cleaned. Show the result of the test on the console and explain how the dataset is cleaned. The rest of the questions must be answered using the cleaned dataset. Display on the console that missing values are fixed.

Output:

```
/Users/juileegayachari/opt/anaconda3/bin/python /Users/juileegayachari/PycharmProjects/pythonProject/venv/Lab2ML.py
symbol     0
date       0
open      11
high       8
low        8
close      0
volume     0
dtype: int64
After filling the null values the total number of null values are:
symbol    0
date      0
open      0
high      0
low       0
close     0
volume    0
dtype: int64
0
Missing value are fixed
```

Question 2: Using python, develop a program that answers the following questions: [9pts]

Code:

```
#                                                Question 2
#-------------------------------------------------------------------------------
print(df.head())
unique_companies = df['symbol'].unique()

num_unique_companies = len(unique_companies)
print(df['symbol'].unique())
print("Number of Unique companies ")
print(num_unique_companies)


newdf = ['AAPL','GOOGL']
filtered_df = df[df['symbol'].isin(newdf)]

plt.figure(figsize=(12, 8))

for company in newdf:
    company_data = filtered_df[filtered_df['symbol'] == company]
    plt.plot( *args: company_data['date'], company_data['close'], label=company)

plt.xlabel('Date')
plt.ylabel('USD ($)')
plt.title('APPLE AND GOOGLE closing value comparison')
 # Rotate x-axis labels for better visibility

plt.gca().xaxis.set_major_locator(plt.MaxNLocator(6))
plt.legend()
plt.show()

#-------------------------------------------------------------------------------
```

Output:

a. How many unique companies are listed in the dataset. List the unique name of companies on the console.

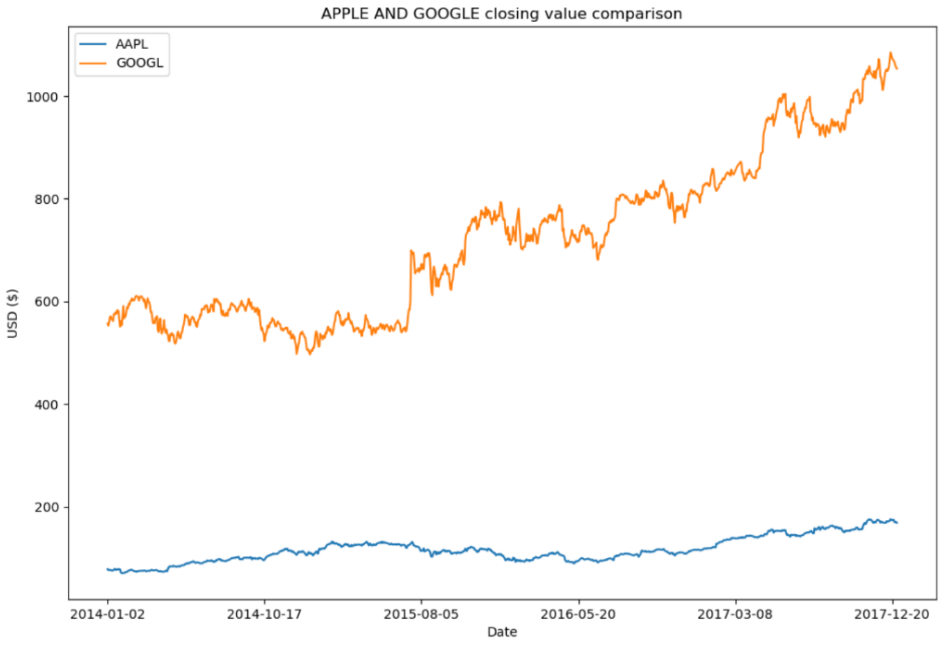
```
 'BHF' 'DWDP' 'APTV']
Number of Unique companies
505
The company with the maximum variance in closing cost is '    PCLN
```

```
['AAL' 'AAPL' 'AAP' 'ABBV' 'ABC' 'ABT' 'ACN' 'ADBE' 'ADI' 'ADM' 'ADP'
 'ADSK' 'ADS' 'AEE' 'AEP' 'AES' 'AET' 'AFL' 'AGN' 'AIG' 'AIV' 'AIZ' 'AJG'
 'AKAM' 'ALB' 'ALGN' 'ALK' 'ALLE' 'ALL' 'ALXN' 'AMAT' 'AMD' 'AME' 'AMGN'
 'AMG' 'AMP' 'AMT' 'AMZN' 'ANDV' 'ANSS' 'ANTM' 'AON' 'AOS' 'APA' 'APC'
 'APD' 'APH' 'ARE' 'ARNC' 'ATVI' 'AVB' 'AVGO' 'AVY' 'AWK' 'AXP' 'AYI'
 'AZO' 'A' 'BAC' 'BAX' 'BA' 'BBT' 'BBY' 'BDX' 'BEN' 'BF.B' 'BIIB' 'BK'
 'BLK' 'BLL' 'BMY' 'BRK.B' 'BSX' 'BWA' 'BXP' 'CAG' 'CAH' 'CAT' 'CA' 'CBG'
 'CBOE' 'CBS' 'CB' 'CCI' 'CCL' 'CDNS' 'CELG' 'CERN' 'CF' 'CHD' 'CHK'
 'CHRW' 'CHTR' 'CINF' 'CI' 'CLX' 'CL' 'CMA' 'CMCSA' 'CME' 'CMG' 'CMI'
 'CMS' 'CNC' 'CNP' 'COF' 'COG' 'COL' 'COO' 'COP' 'COST' 'COTY' 'CPB' 'CRM'
 'CSCO' 'CSX' 'CTAS' 'CTL' 'CTSH' 'CTXS' 'CVS' 'CVX' 'CXO' 'C' 'DAL' 'DE'
 'DFS' 'DGX' 'DG' 'DHI' 'DHR' 'DISCA' 'DISCK' 'DISH' 'DIS' 'DLR' 'DLTR'
 'DOV' 'DPS' 'DRE' 'DRI' 'DTE' 'DUK' 'DVA' 'DVN' 'D' 'EA' 'EBAY' 'ECL'
 'ED' 'EFX' 'EIX' 'EL' 'EMN' 'EMR' 'EOG' 'EQIX' 'EQR' 'EQT' 'ESRX' 'ESS'
 'ES' 'ETFC' 'ETN' 'ETR' 'EW' 'EXC' 'EXPD' 'EXPE' 'EXR' 'FAST' 'FBHS' 'FB'
 'FCX' 'FDX' 'FE' 'FFIV' 'FISV' 'FIS' 'FITB' 'FLIR' 'FLR' 'FLS' 'FL' 'FMC'
 'FOXA' 'FOX' 'FRT' 'FTI' 'F' 'GD' 'GE' 'GGP' 'GILD' 'GIS' 'GLW' 'GM'
 'GOOGL' 'GPC' 'GPN' 'GPS' 'GRMN' 'GS' 'GT' 'GWW' 'HAL' 'HAS' 'HBAN' 'HBI'
 'HCA' 'HCN' 'HCP' 'HD' 'HES' 'HIG' 'HII' 'HOG' 'HOLX' 'HON' 'HP' 'HRB'
 'HRL' 'HRS' 'HSIC' 'HST' 'HSY' 'HUM' 'IBM' 'ICE' 'IDXX' 'IFF' 'ILMN'
 'INCY' 'INTC' 'INTU' 'IPG' 'IP' 'IQV' 'IRM' 'IR' 'ISRG' 'ITW' 'IT' 'IVZ'
 'JBHT' 'JCI' 'JEC' 'JNJ' 'JNPR' 'JPM' 'JWN' 'KEY' 'KIM' 'KLAC' 'KMB'
 'KMI' 'KMX' 'KORS' 'KO' 'KR' 'KSS' 'KSU' 'K' 'LB' 'LEG' 'LEN' 'LH' 'LKQ'
 'LLL' 'LLY' 'LMT' 'LNC' 'LNT' 'LOW' 'LRCX' 'LUK' 'LUV' 'LYB' 'L' 'MAA'
 'MAC' 'MAR' 'MAS' 'MAT' 'MA' 'MCD' 'MCHP' 'MCK' 'MCO' 'MDLZ' 'MDT' 'MET'
 'MGM' 'MHK' 'MKC' 'MLM' 'MMC' 'MMM' 'MNST' 'MON' 'MOS' 'MO' 'MPC' 'MRK'
 'MRO' 'MSFT' 'MSI' 'MS' 'MTB' 'MTD' 'MU' 'MYL' 'M' 'NBL' 'NCLH' 'NDAQ'
 'NEE' 'NEM' 'NFLX' 'NFX' 'NI' 'NKE' 'NLSN' 'NOC' 'NOV' 'NRG' 'NSC' 'NTAP'
 'NTRS' 'NUE' 'NVDA' 'NWL' 'NWSA' 'NWS' 'OKE' 'OMC' 'ORCL' 'ORLY' 'OXY'
 'O' 'PAYX' 'PBCT' 'PCAR' 'PCG' 'PCLN' 'PDCO' 'PEG' 'PEP' 'PFE' 'PFG'
 'PGR' 'PG' 'PHM' 'PH' 'PKG' 'PKI' 'PLD' 'PM' 'PNC' 'PNR' 'PNW' 'PPG'
 'PPL' 'PRGO' 'PRU' 'PSA' 'PSX' 'PVH' 'PWR' 'PXD' 'PX' 'QCOM' 'RCL' 'REGN'
 'REG' 'RE' 'RF' 'RHI' 'RHT' 'RJF' 'RL' 'RMD' 'ROK' 'ROP' 'ROST' 'RRC'
```

b. Which of the predictors are quantitative and which are qualitative?

1. Quantitative predictors:
    - open, high, low, close, volume
2. Qualitative predictors:
    - symbol, date

c. Create a new Dataframe that only includes 'GOOGLE' and 'APPLE' stock with all the original predictors in place. Plot the closing stock value for google and apple in one graph as shown below. Note: Figure size = (12,8). Make sure the time x-axis (x-tick) is not crowded.



Question 3: Create a new Dataframe that is aggregated the 'symbol' with the summation operation. Compare the number of objects in the cleaned data set versus the aggregated data set. Display the first 5 rows of the aggregated dataset on the console

Code:

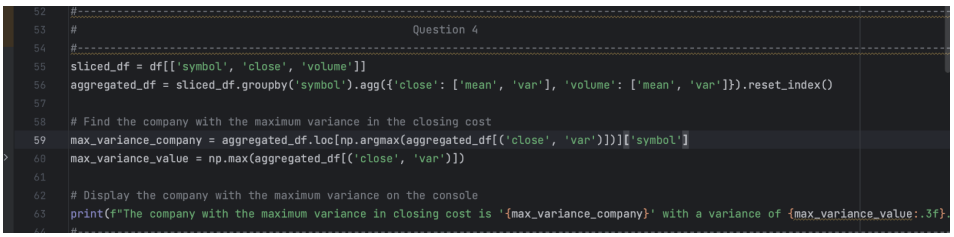
```
#-----------------------------------------------------------------------------------------------------------------------
#                                                  Question 3
#-----------------------------------------------------------------------------------------------------------------------



aggregated_df = df.groupby('symbol').sum()


print("First 5 rows of the aggregated dataset:")
print(aggregated_df.head())

original = len(df)
aggregated = len(aggregated_df)

print(f"Number of objects in the original dataset: {original}")
print(f"Number of objects in the aggregated dataset: {aggregated}")

#
```

Output:

```
First 5 rows of the aggregated dataset:
                open         high          low        close       volume
symbol
A        49429.1700   49831.7499   49021.6259   49441.4100   2207275088
AAL      42717.0438   43339.0077   42091.8799   42721.8746   9819781913
AAP     144207.5150  145729.8516  142654.6559  144189.4300   1145267855
AAPL    117629.0759  118571.2439  116667.0744  117657.7717  45485758169
ABBV     63739.8310   64407.9662   63090.3583   63781.6700   8467697834
Number of objects in the original dataset: 497472
Number of objects in the aggregated dataset: 505
```

Question 4: Create a new Dataframe that is sliced from the cleaned dataset with three features: 'symbol', 'close' and 'volume'. Then aggregate the sliced dataset over 'symbol' feature with the mean and variance operation. Find the company that has the maximum variance in the closing cost. Display a message on the console. Hint: Use the np.argmax and np.max.

Code:



```
#-------------------------------------------------------------------------------------------------------------------------
#                                                  Question 4
#-------------------------------------------------------------------------------------------------------------------------
sliced_df = df[['symbol', 'close', 'volume']]
aggregated_df = sliced_df.groupby('symbol').agg({'close': ['mean', 'var'], 'volume': ['mean', 'var']}).reset_index()

# Find the company with the maximum variance in the closing cost
max_variance_company = aggregated_df.loc[np.argmax(aggregated_df[('close', 'var')])]['symbol']
max_variance_value = np.max(aggregated_df[('close', 'var')])

# Display the company with the maximum variance on the console
print(f"The company with the maximum variance in closing cost is '{max_variance_company}' with a variance of {max_variance_value:.3f}.
```

Output:

```
The company with the maximum variance in closing cost is '    PCLN
Name: 360, dtype: object' with a variance of 69716.561.
```

Question 5: Create a new Dataframe that shows only the Google stock closing cost after 2015-01-01 only. Display the first 5 rows of the newly constructed dataset on the console. [9pts]

Code:

```
#-------------------------------------------------------------------------------
#                                          Question 5
#-------------------------------------------------------------------------------
df['date'] = pd.to_datetime(df['date'])
google_df = df[(df['symbol']=='GOOGL') & (df['date']>'2015-01-01')]
print("First 5 rows of the Google stock closing cost after January 1, 2015:")
print(google_df.head())
```

```
#-------------------------------------------------------------------------------
#                                          Question 5
#-------------------------------------------------------------------------------
df['date'] = pd.to_datetime(df['date'])
google_df = df[(df['symbol']=='GOOGL') & (df['date']>'2015-01-01')]
print("First 5 rows of the Google stock closing cost after January 1, 2015:")
print(google_df[['symbol','close']].head())


#-------------------------------------------------------------------------------
```

Output:

```
First 5 rows of the Google stock closing cost after January 1, 2015:
       symbol       date    open      high     low   close   volume
122595  GOOGL 2015-01-02  532.60  535.8000  527.88  529.55  1327870
123084  GOOGL 2015-01-05  527.15  527.9899  517.75  519.46  2059119
123573  GOOGL 2015-01-06  520.50  521.2100  505.55  506.64  2731813
124062  GOOGL 2015-01-07  510.95  511.4900  503.65  505.15  2345875
124551  GOOGL 2015-01-08  501.51  507.5000  495.02  506.91  3662224
```

```
        symbol   close
122595  GOOGL  529.55
123084  GOOGL  519.46
123573  GOOGL  506.64
124062  GOOGL  505.15
124551  GOOGL  506.91
```

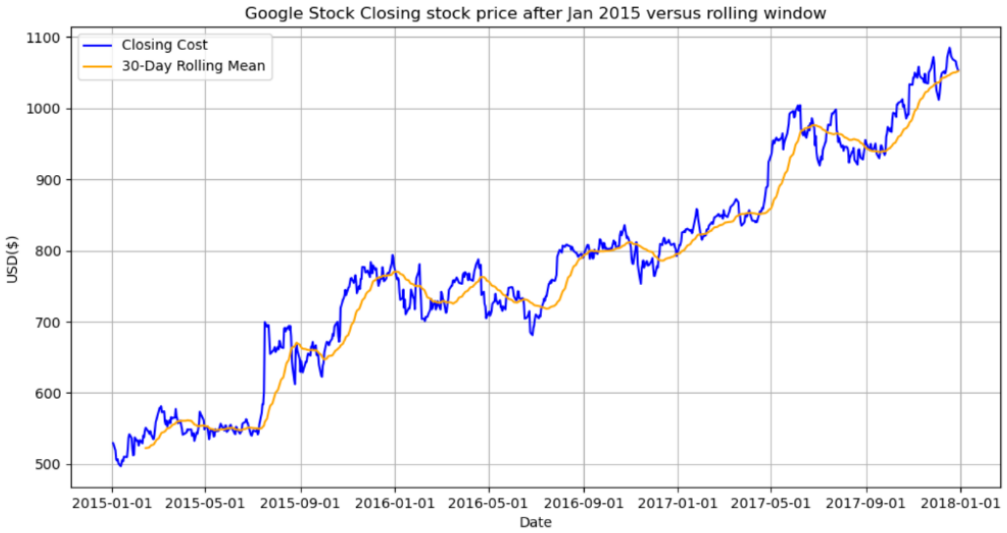
Question 6: Plot the closing cost in the previous question versus the time versus the rolling window [window size = 30 days]. How many observations will be missed when rolling window applies? What are the advantages and disadvantages of rolling windows? Note: The final plot should look like below. [10pts]

Code:

```
#-----------------------------------------------------------------------------------------------------------------
#                                                  Question 6
#-----------------------------------------------------------------------------------------------------------------

rolling_mean = google_df['close'].rolling(window=30).mean()

# Create a figure and axis for the plot
plt.figure(figsize=(12, 6))
# Plot the closing cost
plt.plot( *args: google_df['date'], google_df['close'], label='Closing Cost', color='blue')

# Plot the rolling mean
plt.plot( *args: google_df['date'], rolling_mean, label='30-Day Rolling Mean', color='orange')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m-%d"))
# Customize the plot
plt.xlabel('Date')
plt.ylabel('USD($)')
plt.title('Google Stock Closing stock price after Jan 2015 versus rolling window')
plt.legend()

# Show the plot
plt.grid(True)
plt.show()

# Calculate the number of observations missed due to rolling window
missed_observations = len(google_df) - len(rolling_mean.dropna())
print(f"Number of observations missed due to rolling window: {missed_observations}")


#-----------------------------------------------------------------------------------------------------------------
```

Output:

```
124351 GOOGL 2018-01-08 561.51 567.5000 493.82 566.91 3662224

Number of observations missed due to rolling window: 29

symbol     date     open     high      low    close  volume price_category
```



1. Advantages of rolling window:
    - In a rolling window we basically calculate the average of a subset of data points within a fixed sized window that moves over the dataset. It is usually used in time series analysis of data. It has many advantages.

- Firsty, it helps reduce the noise within the data. In time series analysis there can be short term fluctuations within the data which creates random noise. With the help of rolling window technique we can reduce this.
- Secondly, it can be useful for identifying long term trends in the data. They help in understanding important patterns within the data which can be useful for decision making.
- Rolling window can also be used for data compression to reduce the amount of data for visualisation and analysis.

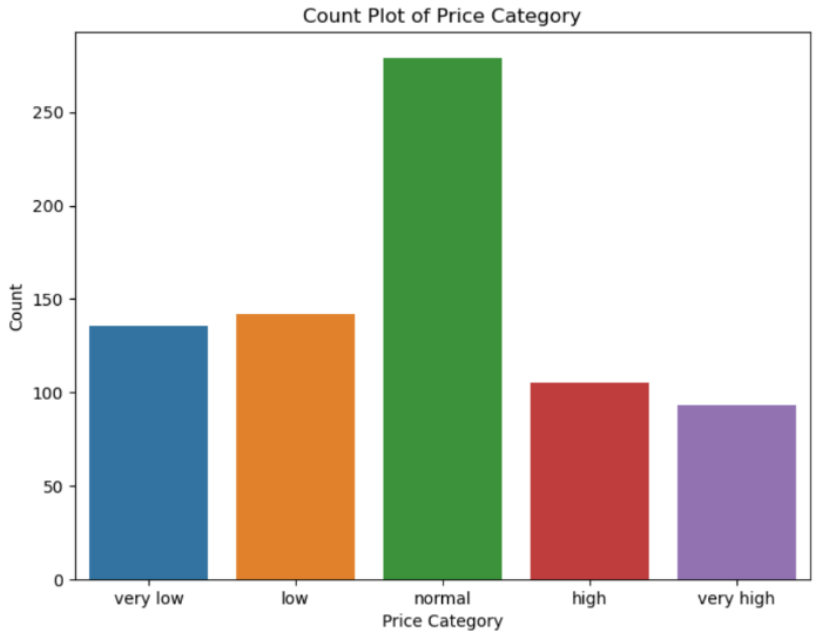
2. Disadvantages of Rolling Mean
    - The choice of window size is a very critical task and can impact the interpretation of data. Smaller window size captures short term variations and larger window size capture long term variations ignoring the short term trends within the data.
    - Another drawback of rolling windows is loss of data leading to less precision. While calculating the rolling mean we calculate the average of the data, by which fine grained details can be lost.

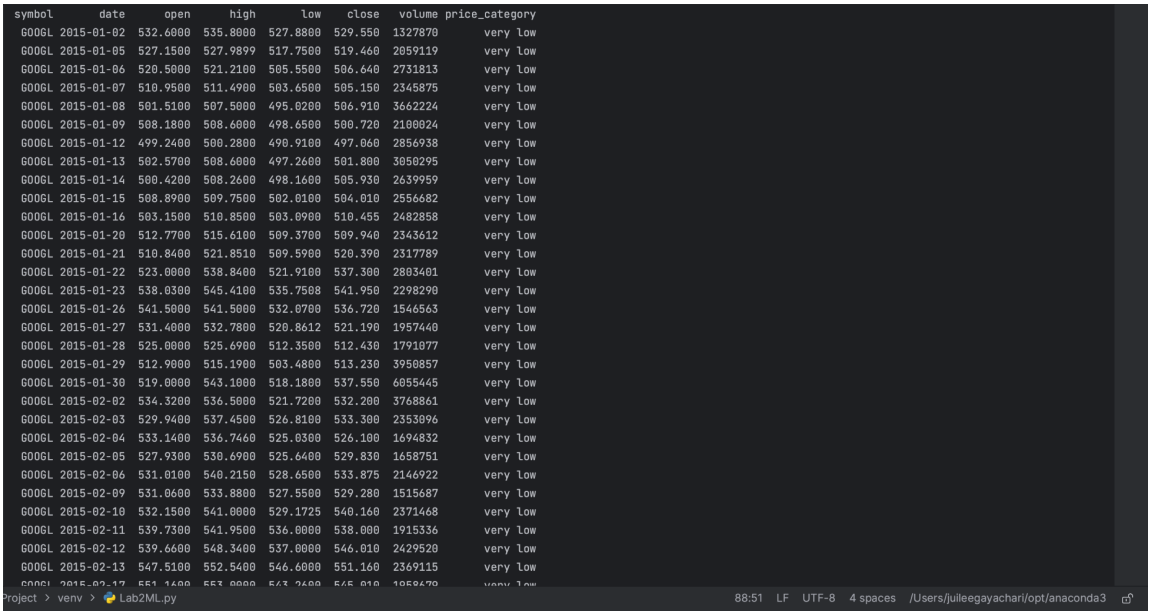
Question 7: Discretize the 'close' feature in the created dataset in question 5 into 5 equal-width bins: 'very low', 'low', 'normal', 'high', 'very high'. Name the new categorical feature as 'price_category'. Plot the count plot of the new added categorical feature. Display the created dataset on the console using the df.to_string() function. [6pts]

Code:

```
#-----------------------------------------------------------------------------------------------------------------
#                                                  Question 7
#-----------------------------------------------------------------------------------------------------------------
# Define bin labels
bin_labels = ['very low', 'low', 'normal', 'high', 'very high']

# Discretize the 'ClosingPrice' into 5 equal-width bins and create the 'price_category' column
google_df['price_category'] = pd.cut(google_df['close'], bins=5, labels=bin_labels)

# Plot a count plot of the 'price_category' feature
plt.figure(figsize=(8, 6))
sns.countplot(data=google_df, x='price_category', order=bin_labels)
plt.xlabel('Price Category')
plt.ylabel('Count')
plt.title('Count Plot of Price Category')
plt.show()

print(google_df.to_string(index=False))

#-----------------------------------------------------------------------------------------------------------------
```

Output:





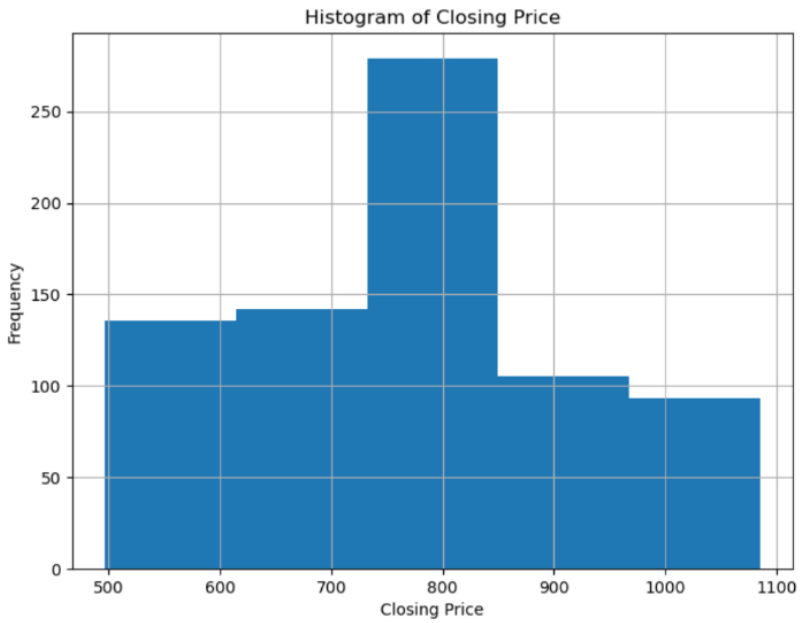
symbol	date	open	high	low	close	volume	price_category
GOOGL	2015-01-02	532.6000	535.8000	527.8800	529.550	1327870	very low
GOOGL	2015-01-05	527.1500	527.9899	517.7500	519.460	2059119	very low
GOOGL	2015-01-06	520.5000	521.2100	505.5500	506.640	2731813	very low
GOOGL	2015-01-07	510.9500	511.4900	503.6500	505.150	2345875	very low
GOOGL	2015-01-08	501.5100	507.5000	495.0200	506.910	3662224	very low
GOOGL	2015-01-09	508.1800	508.6000	498.6500	500.720	2100024	very low
GOOGL	2015-01-12	499.2400	500.2800	490.9100	497.060	2856938	very low
GOOGL	2015-01-13	502.5700	508.6000	497.2600	501.800	3050295	very low
GOOGL	2015-01-14	500.4200	508.2600	498.1600	505.930	2639959	very low
GOOGL	2015-01-15	508.8900	509.7500	502.0100	504.010	2556682	very low
GOOGL	2015-01-16	503.1500	510.8500	503.0900	510.455	2482858	very low
GOOGL	2015-01-20	512.7700	515.6100	509.3700	509.940	2343612	very low
GOOGL	2015-01-21	510.8400	521.8510	509.5900	520.390	2317789	very low
GOOGL	2015-01-22	523.0000	538.8400	521.9100	537.300	2803401	very low
GOOGL	2015-01-23	538.0300	545.4100	535.7508	541.950	2298290	very low
GOOGL	2015-01-26	541.5000	541.5000	532.0700	536.720	1546563	very low
GOOGL	2015-01-27	531.4000	532.7800	520.8612	521.190	1957440	very low
GOOGL	2015-01-28	525.0000	525.6900	512.3500	512.430	1791077	very low
GOOGL	2015-01-29	512.9000	515.1900	503.4800	513.230	3950857	very low
GOOGL	2015-01-30	519.0000	543.1000	518.1800	537.550	6055445	very low
GOOGL	2015-02-02	534.3200	536.5000	521.7200	532.200	3768861	very low
GOOGL	2015-02-03	529.9400	537.4500	526.8100	533.300	2353096	very low
GOOGL	2015-02-04	533.1400	536.7460	525.0300	526.100	1694832	very low
GOOGL	2015-02-05	527.9300	530.6900	525.6400	529.830	1658751	very low
GOOGL	2015-02-06	531.0100	540.2150	528.6500	533.875	2146922	very low
GOOGL	2015-02-09	531.0600	533.8800	527.5500	529.280	1515687	very low
GOOGL	2015-02-10	532.1500	541.0000	529.1725	540.160	2371468	very low
GOOGL	2015-02-11	539.7300	541.9500	536.0000	538.000	1915336	very low
GOOGL	2015-02-12	539.6600	548.3400	537.0000	546.010	2429520	very low
GOOGL	2015-02-13	547.5100	552.5400	546.6000	551.160	2369115	very low
GOOGL	2015-02-17	551.1400	553.0000	543.2600	545.010	1858479	very low

Question 8: Plot the histogram plot of the 'close' feature in the previous question (number of bins = 5) . Does the histogram plot make sense compared to the count plot in the previous question. Explain your answer. [7pts]

Code:

```
#-----------------------------------------------------------------------------------------------------------
#                                                  Question 8
#-----------------------------------------------------------------------------------------------------------
plt.figure(figsize=(8, 6))
plt.hist(google_df['close'], bins=5)
plt.xlabel('Closing Price')
plt.ylabel('Frequency')
plt.title('Histogram of Closing Price')
plt.grid(True)
plt.show()

#-----------------------------------------------------------------------------------------------------------
```

Output:



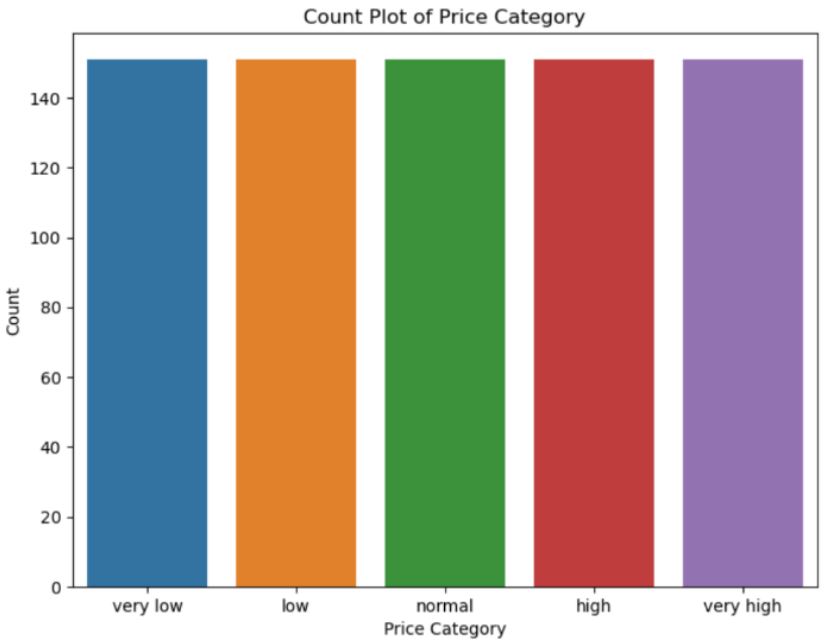
- The histogram plot visualises the distribution of close features by dividing it into 5 bins. Here the x-axis represents the closing prize. It is suitable for understanding the continuous distribution of close features. In histogram we also understand the range of width.
- On the other hand, the count plot gives us the frequency distribution of data. As in the above example we have categorised the features based upon their closing prizes into 5 distinct categories very low, low, normal, high, very high. We can properly visualise how the data is distributed across these categories.
- Both of the plots serve different analytical needs, The histogram plot provides a detailed view of the closing price distribution, while the count plot summarises the distribution based on predefined price categories. The visualisation and understanding of the data is easier for the count plot, as we can summarise the data in one glance.

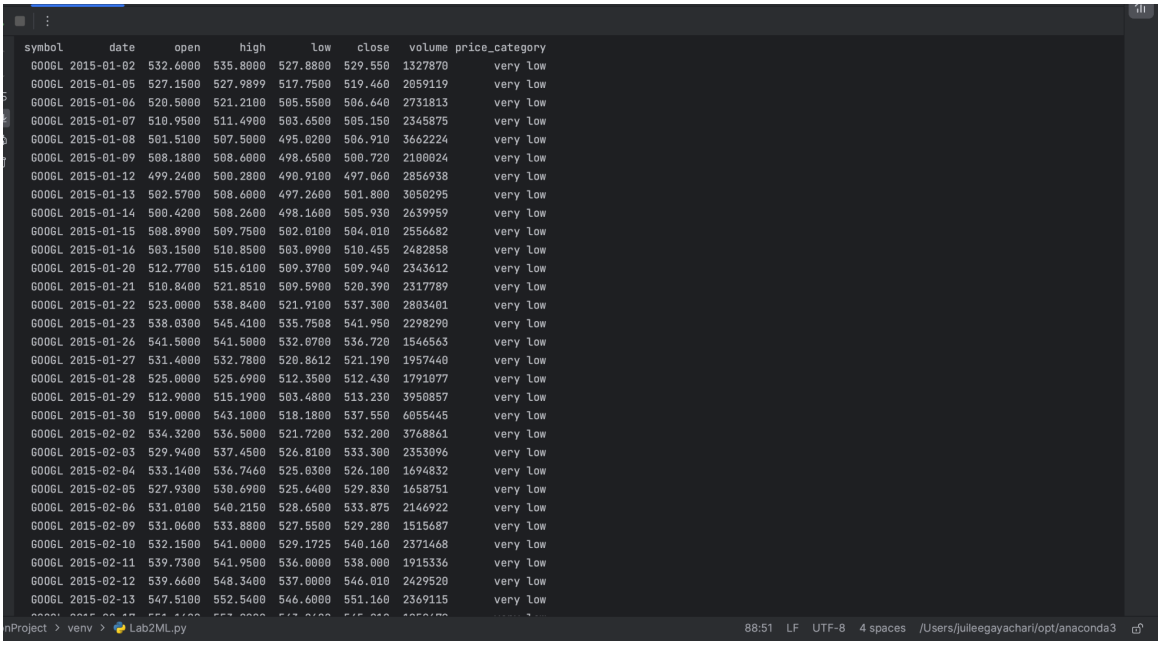
Question 9: Discretize the 'close' feature in the created dataset in question 5 into equal frequencies (q = 5). Bins: 'very low', 'low', 'normal', 'high', 'very high'. Name the new categorical feature as 'price_category'. Plot the count plot of the new added categorical feature. Display the created dataset on the console using the df.to_string() function. [6pts]

Code:

```
#-------------------------------------------------------------------------------------------------------------------
#                                                  Question 9
#-------------------------------------------------------------------------------------------------------------------
google_df['price_category'] = pd.qcut(google_df['close'], q=5, labels=['very low', 'low', 'normal', 'high', 'very high'])
plt.figure(figsize=(8,6))
sns.countplot(data_=_google_df, x='price_category', order=['very low', 'low', 'normal', 'high', 'very high'])
plt.xlabel('Price Category')
plt.ylabel('Count')
plt.title('Count Plot of Price Category')
plt.show()

print(google_df.to_string(index=False))
```

Output:





symbol	date	open	high	low	close	volume	price_category
GOOGL	2015-01-02	532.6000	535.8000	527.8800	529.550	1327870	very low
GOOGL	2015-01-05	527.1500	527.9899	517.7500	519.460	2059119	very low
GOOGL	2015-01-06	520.5000	521.2100	505.5500	506.640	2731813	very low
GOOGL	2015-01-07	510.9500	511.4900	503.6500	505.150	2345875	very low
GOOGL	2015-01-08	501.5100	507.5000	495.0200	506.910	3662224	very low
GOOGL	2015-01-09	508.1800	508.6000	498.6500	500.720	2100024	very low
GOOGL	2015-01-12	499.2400	500.2800	490.9100	497.060	2856938	very low
GOOGL	2015-01-13	502.5700	508.6000	497.2600	501.800	3050295	very low
GOOGL	2015-01-14	500.4200	508.2600	498.1600	505.930	2639959	very low
GOOGL	2015-01-15	508.8900	509.7500	502.0100	504.010	2556682	very low
GOOGL	2015-01-16	503.1500	510.8500	503.0900	510.455	2482858	very low
GOOGL	2015-01-20	512.7700	515.6100	509.3700	509.940	2343612	very low
GOOGL	2015-01-21	510.8400	521.8510	509.5900	520.390	2317789	very low
GOOGL	2015-01-22	523.0000	538.8400	521.9100	537.300	2803401	very low
GOOGL	2015-01-23	538.0300	545.4100	535.7508	541.950	2298290	very low
GOOGL	2015-01-26	541.5000	541.5000	532.0700	536.720	1546563	very low
GOOGL	2015-01-27	531.4000	532.7800	520.8612	521.190	1957440	very low
GOOGL	2015-01-28	525.0000	525.6900	512.3500	512.430	1791077	very low
GOOGL	2015-01-29	512.9000	515.1900	503.4800	513.230	3950857	very low
GOOGL	2015-01-30	519.0000	543.1000	518.1800	537.550	6055445	very low
GOOGL	2015-02-02	534.3200	536.5000	521.7200	532.200	3768861	very low
GOOGL	2015-02-03	529.9400	537.4500	526.8100	533.300	2353096	very low
GOOGL	2015-02-04	533.1400	536.7460	525.0300	526.100	1694832	very low
GOOGL	2015-02-05	527.9300	530.6900	525.6400	529.830	1658751	very low
GOOGL	2015-02-06	531.0100	540.2150	528.6500	533.875	2146922	very low
GOOGL	2015-02-09	531.0600	533.8800	527.5500	529.280	1515687	very low
GOOGL	2015-02-10	532.1500	541.0000	529.1725	540.160	2371468	very low
GOOGL	2015-02-11	539.7300	541.9500	536.0000	538.000	1915336	very low
GOOGL	2015-02-12	539.6600	548.3400	537.0000	546.010	2429520	very low
GOOGL	2015-02-13	547.5100	552.5400	546.6000	551.160	2369115	very low

Question 10: Using the definition of covariance matrix estimate the covariance matrix without using built-in python function (for the dataset in question 5 including all features (open, high, low, close and volume). For the covariance matrix estimation, you cannot use the built-in function like .cov(). The estimated covariance matrix needs to be constructed using the definition of covariance matrix. Display the covariance matrix contents on the console. [10pts]

Code:

```
#-----------------------------------------------------------------------------------------------------------------------
#                                                  Question 10
#-----------------------------------------------------------------------------------------------------------------------

numerical_columns = ['open', 'high', 'low', 'close', 'volume']
google_num = google_df[numerical_columns]
mean_vector = np.mean(google_num, axis=0)
centered_data = google_num - mean_vector
covariance_matrix = np.dot(centered_data.T, centered_data) / (len(google_num) - 1)
covariance_matrix_df = pd.DataFrame(covariance_matrix, columns=numerical_columns, index=numerical_columns)
print("Covariance Matrix for Numerical Variables:")
print(covariance_matrix_df.round(3))

#
```

Output:

```
Covariance Matrix for Numerical Variables:
                open          high           low         close        volume
open    2.196792e+04  2.200068e+04  2.189256e+04  2.195137e+04 -2.959788e+07
high    2.200068e+04  2.205908e+04  2.193635e+04  2.201092e+04 -2.796411e+07
low     2.189256e+04  2.193635e+04  2.185435e+04  2.191262e+04 -3.202995e+07
close   2.195137e+04  2.201092e+04  2.191262e+04  2.199438e+04 -3.054308e+07
volume -2.959788e+07 -2.796411e+07 -3.202995e+07 -3.054308e+07  1.004830e+12
```

Question 11: Estimate the covariance matrix for the previous question with the built-in python function. cov(). Display the result on the console. The answer to this question must be identical to the answer to the previous question. Write down your observations about the values on diagonal and off diagonal of the covariance matrix. What is the linear relationship between features in this dataset? [10pts]

Code:

```
157
158 #-----------------------------------------------------------------------------------------------------------------------
159 #                                                  Question 11
160 #-----------------------------------------------------------------------------------------------------------------------
161 cov_matrix = google_num.cov()
162 print("Covariance Matrix (Using built-in cov()):")
163 print(cov_matrix)
```

Output:

```
Covariance Matrix (Using built-in cov()):
                open          high           low         close        volume
open    2.196792e+04  2.200868e+04  2.189256e+04  2.195137e+04 -2.959788e+07
high    2.200868e+04  2.205908e+04  2.193635e+04  2.201093e+04 -2.796411e+07
low     2.189256e+04  2.193635e+04  2.185435e+04  2.191262e+04 -3.202995e+07
close   2.195137e+04  2.201093e+04  2.191262e+04  2.199438e+04 -3.054308e+07
volume -2.959788e+07 -2.796411e+07 -3.202995e+07 -3.054308e+07  1.004830e+12

Process finished with exit code 0
```

```
pythonProject › venv › Lab2ML.py   88:51  LF  UTF-8  4 spaces  /Users/juileegayachari/opt/anaconda3
```

- **Diagonal values:** The diagonal values represent the variances of individual features. The variance signifies the spread of data. A larger covariance indicates greater spread of data.
- **Off diagonal values:** The off diagonal values in the covariance matrix represent the covariance between pairs of features. Positive covariance states that there is a positive relationship between the two features. For example, close and open have a positive covariance relationship. Conversely, a negative covariance suggests that there is a negative linear relationship, where one feature tends to decrease as the other increases. A covariance value close to zero suggests little to no linear relationship.
- **Linear relationship between the features:** Based on covariance values we can identify the linear relationship between the values. As discussed in the above point. Positive Linear relationship indicates that, when one feature tends to increase the other one also simultaneously increases.Conversely, Negative Linear relationship suggests that when one feature tends to increase the other feature decreases.