

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

Code:

```
8  #+-----#
9  #----- Question 1
10 #-----#
11 df = pd.read_csv('https://raw.githubusercontent.com/rjafar1979/Information-Visualization-Data-Analytics-Dataset-/main/stock%20prices.csv')
12
13 print(df.isna().sum())
14 df['open'].fillna(df['open'].mean(), inplace=True)
15 df['high'].fillna(df['high'].mean(), inplace=True)
16 df['low'].fillna(df['low'].mean(), inplace=True)
17
18 print("After filling the null values the total number of null values are:")
19 print(df.isna().sum())
20 print(df.isna().sum().sum())
21 if df.isna().sum().sum() == 0:
22     print("Missing value are fixed")
23 else:
24     print("Missing values are not fixed")
25
```

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:

```

23  # Question 2
24  #-----#
25  print(df.head())
26  unique_companies = df['symbol'].unique()
27
28  num_unique_companies = len(unique_companies)
29  print(df['symbol'].unique())
30  print("Number of Unique companies ")
31  print(num_unique_companies)
32
33
34  newdf = ['AAPL', 'GOOGL']
35  filtered_df = df[df['symbol'].isin(newdf)]
36
37  plt.figure(figsize=(12, 8))
38
39  for company in newdf:
40      company_data = filtered_df[filtered_df['symbol'] == company]
41      plt.plot(*args, company_data['date'], company_data['close'], label=company)
42
43  plt.xlabel('Date')
44  plt.ylabel('USD ($)')
45  plt.title('APPLE AND GOOGLE closing value comparison')
46  # Rotate x-axis labels for better visibility
47
48  plt.gca().xaxis.set_major_locator(plt.MaxNLocator(6))
49  plt.legend()
50  plt.show()
51
52  #-----#

```

Output:

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

```

'BHFT' 'DWDP' 'APTV'
Number of Unique companies
505
The company with the maximum variance in closing cost is 'PCIN'

```

```

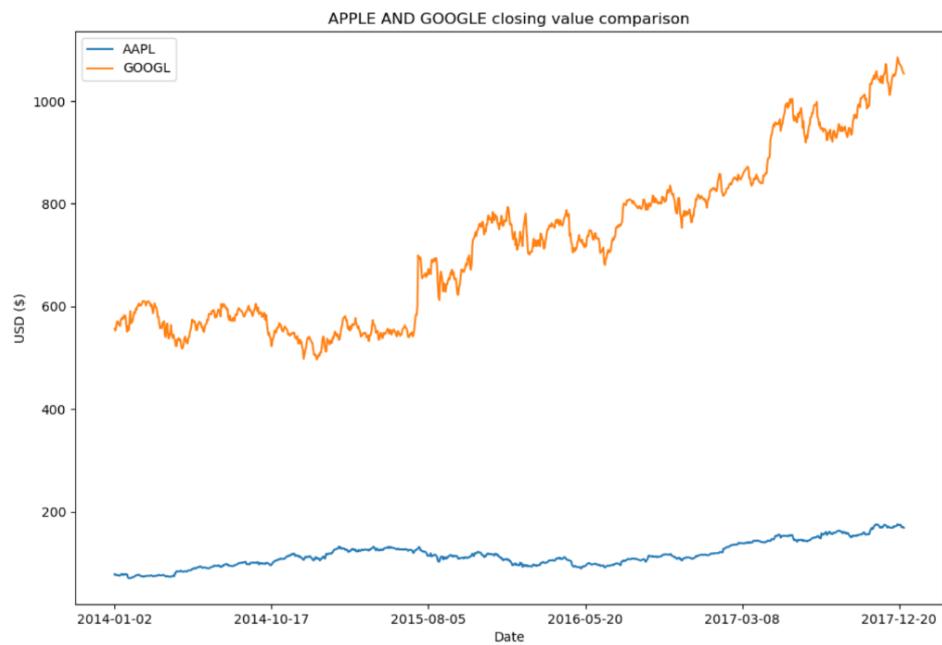
'AAL' 'AAPL' 'AAP' 'ABBV' 'ABC' 'ABT' 'ACN' 'ABE' 'ADI' 'ADM' 'ADP'
'ADSX' 'ADS' 'AEE' 'AEP' 'AES' 'AET' 'AFL' 'AGN' 'AIG' 'AIV' 'AIZ' 'AJG'
'AKAM' 'ALB' 'ALGN' 'ALK' 'ALLE' 'ALL' 'ALXN' 'AMAT' 'AMD' 'AME' 'AMGN'
'AMG' 'AMP' 'ANT' 'AMZN' 'ANDV' 'ANSS' 'ANTM' 'AON' 'AOS' 'APA' 'APC'
'APD' 'APH' 'ARE' 'ARNC' 'ATVI' 'AVB' 'AVGO' 'AVY' 'AWK' 'AXP' 'AYI'
'AZO' 'A' 'BAC' 'BAX' 'BA' 'BBT' 'BBY' 'BDX' 'BEN' 'BF.B' 'BIIB' 'BK'
'BLK' 'BLI' 'BHY' 'BRK.B' 'BSX' 'BWA' 'BXP' 'CAG' 'CAH' 'CAT' 'CA' 'CBG'
'COE' 'CBS' 'CB' 'CCL' 'CDNS' 'CELG' 'CERN' 'CE' 'CHO' 'CHK'
'CHRW' 'CHTR' 'CINF' 'CI' 'CLX' 'CL' 'CMA' 'CNCSA' 'CME' 'CNG' 'CNI'
'CNST' 'CNC' 'CNP' 'COP' 'COP' 'COL' 'COP' 'COP' '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' 'DWN' 'D' 'EA' 'EBAY' 'ECL'
'ED' 'EFX' 'EIX' 'EL' 'EMN' 'EMR' 'EOG' 'EOIX' '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' 'GPP' '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' 'LR' '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' 'MMMF' 'MNST' 'MON' 'MOS' 'MO' 'MPC' 'MRK'
'MRO' 'MSFT' 'MSI' 'MS' 'MTB' 'MTD' 'MU' 'MYL' 'M' 'NBL' 'NCLH' 'NDQ'
'NEE' 'NEM' 'NFLX' 'NFX' 'NT' 'NKE' 'NLSN' 'NOC' 'NOV' 'NRG' 'NSC' 'NTAP'
'NTRS' 'NUE' 'NVDA' 'NWNL' 'NWSA' 'NWS' 'OKE' 'OMC' 'ORCL' 'ORLY' 'OXY'
'O' 'PAYX' 'PBCT' 'PCAR' 'PCG' 'PCLN' 'PDCO' 'PEG' 'PEP' 'PFE' 'PFG'
'PGR' 'PG' 'PHM' 'PH' 'PKG' 'PKT' '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:

```
52  #-----#
53  #-----#                                     Question 3
54  #-----#
55
56
57  aggregated_df = df.groupby('symbol').sum()
58
59
60  print("First 5 rows of the aggregated dataset:")
61  print(aggregated_df.head())
62
63  original = len(df)
64  aggregated = len(aggregated_df)
65
66  print(f"Number of objects in the original dataset: {original}")
67  print(f"Number of objects in the aggregated dataset: {aggregated}")
68
69  #
```

Output:

```
First 5 rows of the aggregated dataset:
symbol      open      high      low     close      volume
A          49429.1700  49831.7499  49021.6259  49441.4100  2207275888
AAL        42717.0438  43339.0077  42891.8799  42721.8746  9819781913
AAP        144207.5150  145729.8516  142654.6559  144189.4500  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:

```
52: #-----#
53: #                                         Question 4
54: #
55: sliced_df = df[['symbol', 'close', 'volume']]
56: aggregated_df = sliced_df.groupby('symbol').agg({'close': ['mean', 'var'], 'volume': ['mean', 'var']}).reset_index()
57:
58: # Find the company with the maximum variance in the closing cost
59: max_variance_company = aggregated_df.loc[np.argmax(aggregated_df[['close', 'var']])]['symbol']
> 60: max_variance_value = np.max(aggregated_df[['close', 'var']])
61:
62: # Display the company with the maximum variance on the console
63: print(f"The company with the maximum variance in closing cost is '{max_variance_company}' with a variance of {max_variance_value:.3f}.
```

Output:

```
 0    1    2    3    4    5    6    7    8    9
The company with the maximum variance in closing cost is '  PCLN
Name: 360, dtype: object' with a variance of 69716.561.
First 5 rows of the Google stock closing cost after January 1, 2015.
```

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:

```
64  #-----  
65  #-----  
66  #-----  
67  df['date'] = pd.to_datetime(df['date'])  
68  google_df = df[(df['symbol']=='GOOGL') & (df['date']>='2015-01-01')]  
69  print("First 5 rows of the Google stock closing cost after January 1, 2015:")  
70  print(google_df.head())  
71  #-----  
72  #-----  
73  #-----  
74  #-----  
75  #-----  
76  #-----  
77  #-----  
78  #-----  
79  #-----  
80  #-----  
81  #-----  
82  #-----  
83  #-----  
84  #-----  
85  #-----  
86  #-----  
87  #-----  
88  #-----  
89  df['date'] = pd.to_datetime(df['date'])  
90  google_df = df[(df['symbol']=='GOOGL') & (df['date']>='2015-01-01')]  
91  print("First 5 rows of the Google stock closing cost after January 1, 2015:")  
92  print(google_df[['symbol','close']].head())  
93  #-----  
94  #-----  
95  #-----
```

Output:

```
▶  └── name: .006, dtype: object with a variance of 0.7710.301.  
  ↴  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  
  ↴  Number of observations missed due to rolling window: 20
```

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:

```

95
96 # Question 6
97 #-----
98
99 rolling_mean = google_df['close'].rolling(window=30).mean()
100
101 # Create a figure and axis for the plot
102 plt.figure(figsize=(12, 6))
103 # Plot the closing cost
104 plt.plot(*args: google_df['date'], google_df['close'], label='Closing Cost', color='blue')
105
106 # Plot the rolling mean
107 plt.plot(*args: google_df['date'], rolling_mean, label='30-Day Rolling Mean', color='orange')
108 plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m-%d"))
109 # Customize the plot
110 plt.xlabel('Date')
111 plt.ylabel('USD($)')
112 plt.title('Google Stock Closing stock price after Jan 2015 versus rolling window')
113 plt.legend()
114
115 # Show the plot
116 plt.grid(True)
117 plt.show()
118
119 # Calculate the number of observations missed due to rolling window
120 missed_observations = len(google_df) - len(rolling_mean.dropna())
121 print(f"Number of observations missed due to rolling window: {missed_observations}")
122
123 #-----
124

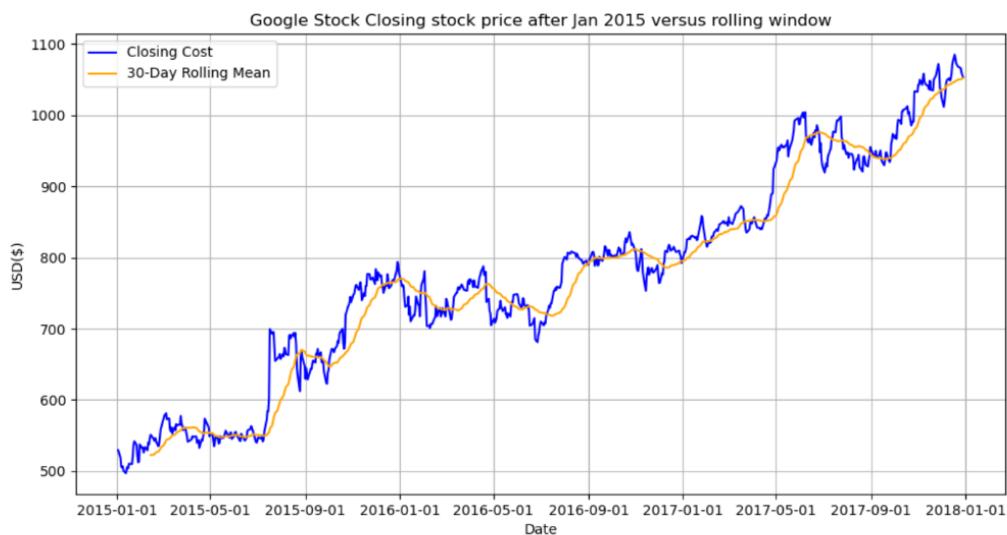
```

Output:

```

124351 GOOGL 2015-01-08 501.51 507.3000 493.02 500.71 3062224
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.

- Firstly, 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

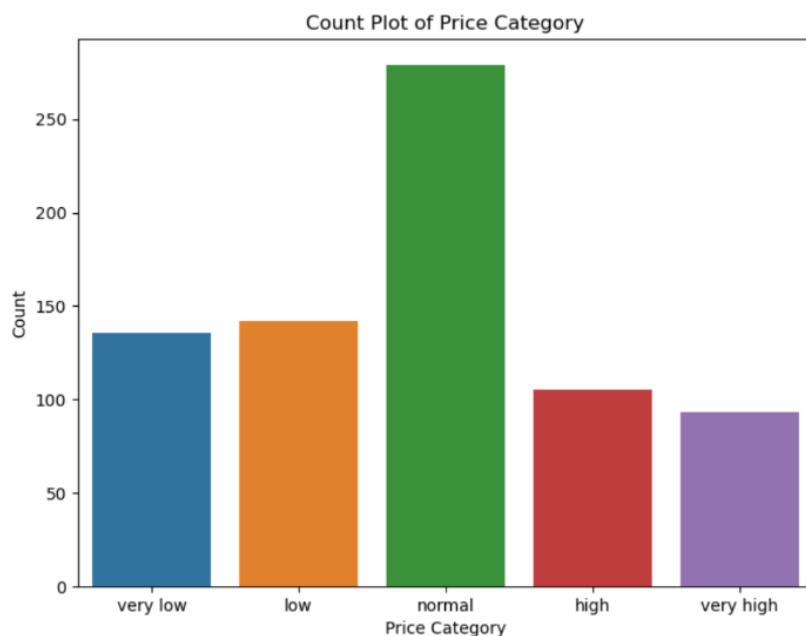
- The choice of window size is a very critical task and can impact the interpretation of data. Smaller window size captures short term variations and larger window size capture long term variations ignoring the short term trends within the data.
- Another drawback of rolling windows is loss of data leading 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:

```
102 #-----  
103 #-----  
104 #-----  
105 # Define bin labels  
106 bin_labels = ['very low', 'low', 'normal', 'high', 'very high']  
107  
108 # Discretize the 'ClosingPrice' into 5 equal-width bins and create the 'price_category' column  
109 google_df['price_category'] = pd.cut(google_df['close'], bins=5, labels=bin_labels)  
110  
111 # Plot a count plot of the 'price_category' feature  
112 plt.figure(figsize=(8, 6))  
113 sns.countplot(data=google_df, x='price_category', order=bin_labels)  
114 plt.xlabel('Price Category')  
115 plt.ylabel('Count')  
116 plt.title('Count Plot of Price Category')  
117 plt.show()  
118  
119 print(google_df.to_string(index=False))  
120  
121 #-----
```

Output:



symbol	date	open	high	low	close	volume	price_category
GOOGL	2015-01-02	532.4800	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	526.5000	521.2100	505.5500	506.440	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	2108024	very low
GOOGL	2015-01-12	499.2400	500.2800	498.9100	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	2462858	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	535.7500	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	1791677	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	536.7400	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	546.4000	551.160	2369115	very low
GOOGL	2015-02-17	551.1400	557.0000	542.2600	558.010	1058470	very low

Project > venv >  Lab2ML.py

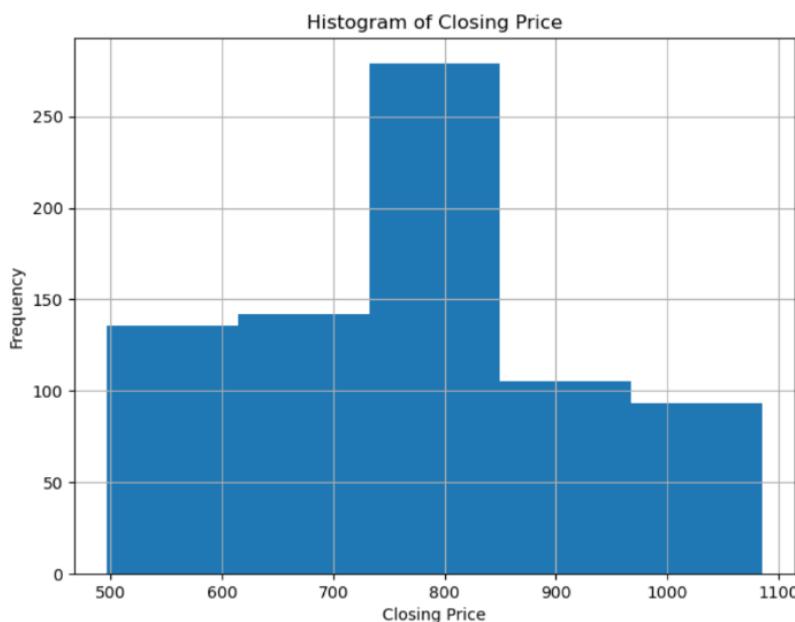
88:51 LF UTF-8 4 spaces /Users/julieegayachari/opt/anaconda3 

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:

```
123 #-----  
124 #-----  
125 #-----  
126 plt.figure(figsize=(8, 6))  
127 Question 8  
128 plt.hist(google_df['close'], bins=5)  
129 plt.xlabel('Closing Price')  
130 plt.ylabel('Frequency')  
131 plt.title('Histogram of Closing Price')  
132 plt.grid(True)  
133 plt.show()  
134 #-----
```

Output:



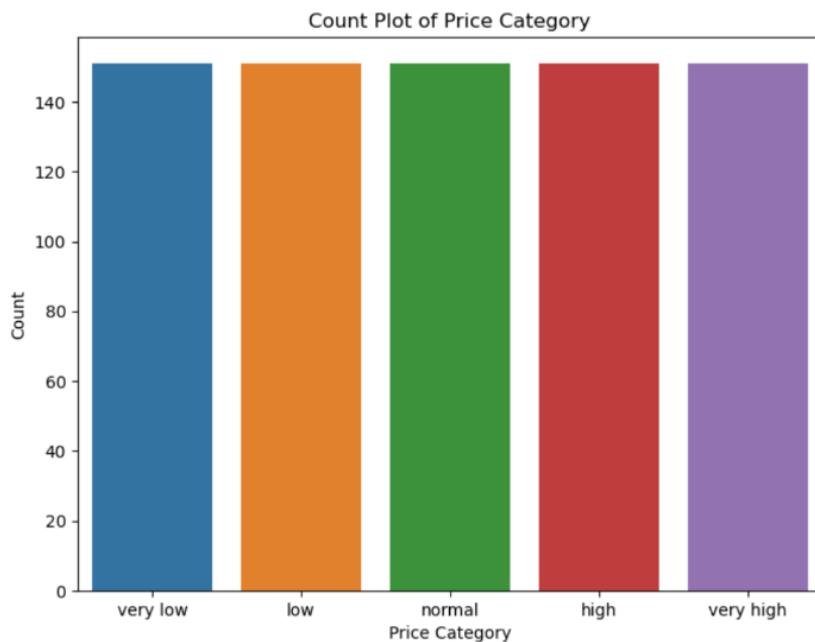
- The histogram plot visualises the distribution of close features by dividing it into 5 bins. Here the x-axis represents the closing prize. It is suitable for understanding the continuous distribution of close features. In histogram we also understand the range of width.
- On the other hand, the count 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:

```
132 #-----  
133 #-----  
134 #-----  
135 google_df['price_category'] = pd.qcut(google_df['close'], q=5, labels=['very low', 'low', 'normal', 'high', 'very high'])  
136 plt.figure(figsize=(8,6))  
137 sns.countplot(data=google_df, x='price_category', order=['very low', 'low', 'normal', 'high', 'very high'])  
138 plt.xlabel('Price Category')  
139 plt.ylabel('Count')  
140 plt.title('Count Plot of Price Category')  
141 plt.show()  
142  
143 print(google_df.to_string(index=False))  
144
```

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  495.8200  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  2883401  very low
GOOGL 2015-01-23  538.0300  545.4100  535.7500  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  2353996  very low
GOOGL 2015-02-04  533.1400  536.7400  525.8500  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  528.4500  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 2015-02-14  552.5400  557.6200  547.0000  555.010  2050170  very low
GOOGL 2015-02-15  557.6200  562.6000  551.1600  560.000  1950000  very low
GOOGL 2015-02-16  562.6000  567.6000  551.1600  565.000  1850000  very low
GOOGL 2015-02-17  567.6000  572.6000  551.1600  565.000  1750000  very low
GOOGL 2015-02-18  572.6000  577.6000  551.1600  565.000  1650000  very low
GOOGL 2015-02-19  577.6000  582.6000  551.1600  565.000  1550000  very low
GOOGL 2015-02-20  582.6000  587.6000  551.1600  565.000  1450000  very low
GOOGL 2015-02-21  587.6000  592.6000  551.1600  565.000  1350000  very low
GOOGL 2015-02-22  592.6000  597.6000  551.1600  565.000  1250000  very low
GOOGL 2015-02-23  597.6000  602.6000  551.1600  565.000  1150000  very low
GOOGL 2015-02-24  602.6000  607.6000  551.1600  565.000  1050000  very low
GOOGL 2015-02-25  607.6000  612.6000  551.1600  565.000  950000  very low
GOOGL 2015-02-26  612.6000  617.6000  551.1600  565.000  850000  very low
GOOGL 2015-02-27  617.6000  622.6000  551.1600  565.000  750000  very low
GOOGL 2015-02-28  622.6000  627.6000  551.1600  565.000  650000  very low
GOOGL 2015-03-01  627.6000  632.6000  551.1600  565.000  550000  very low
GOOGL 2015-03-02  632.6000  637.6000  551.1600  565.000  450000  very low
GOOGL 2015-03-03  637.6000  642.6000  551.1600  565.000  350000  very low
GOOGL 2015-03-04  642.6000  647.6000  551.1600  565.000  250000  very low
GOOGL 2015-03-05  647.6000  652.6000  551.1600  565.000  150000  very low
GOOGL 2015-03-06  652.6000  657.6000  551.1600  565.000  100000  very low
GOOGL 2015-03-07  657.6000  662.6000  551.1600  565.000  50000  very low
GOOGL 2015-03-08  662.6000  667.6000  551.1600  565.000  20000  very low
GOOGL 2015-03-09  667.6000  672.6000  551.1600  565.000  10000  very low
GOOGL 2015-03-10  672.6000  677.6000  551.1600  565.000  5000  very low
GOOGL 2015-03-11  677.6000  682.6000  551.1600  565.000  2000  very low
GOOGL 2015-03-12  682.6000  687.6000  551.1600  565.000  1000  very low
GOOGL 2015-03-13  687.6000  692.6000  551.1600  565.000  500  very low
GOOGL 2015-03-14  692.6000  697.6000  551.1600  565.000  200  very low
GOOGL 2015-03-15  697.6000  702.6000  551.1600  565.000  100  very low
GOOGL 2015-03-16  702.6000  707.6000  551.1600  565.000  50  very low
GOOGL 2015-03-17  707.6000  712.6000  551.1600  565.000  20  very low
GOOGL 2015-03-18  712.6000  717.6000  551.1600  565.000  10  very low
GOOGL 2015-03-19  717.6000  722.6000  551.1600  565.000  5  very low
GOOGL 2015-03-20  722.6000  727.6000  551.1600  565.000  2  very low
GOOGL 2015-03-21  727.6000  732.6000  551.1600  565.000  1  very low
GOOGL 2015-03-22  732.6000  737.6000  551.1600  565.000  0  very low
GOOGL 2015-03-23  737.6000  742.6000  551.1600  565.000  0  very low
GOOGL 2015-03-24  742.6000  747.6000  551.1600  565.000  0  very low
GOOGL 2015-03-25  747.6000  752.6000  551.1600  565.000  0  very low
GOOGL 2015-03-26  752.6000  757.6000  551.1600  565.000  0  very low
GOOGL 2015-03-27  757.6000  762.6000  551.1600  565.000  0  very low
GOOGL 2015-03-28  762.6000  767.6000  551.1600  565.000  0  very low
GOOGL 2015-03-29  767.6000  772.6000  551.1600  565.000  0  very low
GOOGL 2015-03-30  772.6000  777.6000  551.1600  565.000  0  very low
GOOGL 2015-03-31  777.6000  782.6000  551.1600  565.000  0  very low
GOOGL 2015-04-01  782.6000  787.6000  551.1600  565.000  0  very low
GOOGL 2015-04-02  787.6000  792.6000  551.1600  565.000  0  very low
GOOGL 2015-04-03  792.6000  797.6000  551.1600  565.000  0  very low
GOOGL 2015-04-04  797.6000  802.6000  551.1600  565.000  0  very low
GOOGL 2015-04-05  802.6000  807.6000  551.1600  565.000  0  very low
GOOGL 2015-04-06  807.6000  812.6000  551.1600  565.000  0  very low
GOOGL 2015-04-07  812.6000  817.6000  551.1600  565.000  0  very low
GOOGL 2015-04-08  817.6000  822.6000  551.1600  565.000  0  very low
GOOGL 2015-04-09  822.6000  827.6000  551.1600  565.000  0  very low
GOOGL 2015-04-10  827.6000  832.6000  551.1600  565.000  0  very low
GOOGL 2015-04-11  832.6000  837.6000  551.1600  565.000  0  very low
GOOGL 2015-04-12  837.6000  842.6000  551.1600  565.000  0  very low
GOOGL 2015-04-13  842.6000  847.6000  551.1600  565.000  0  very low
GOOGL 2015-04-14  847.6000  852.6000  551.1600  565.000  0  very low
GOOGL 2015-04-15  852.6000  857.6000  551.1600  565.000  0  very low
GOOGL 2015-04-16  857.6000  862.6000  551.1600  565.000  0  very low
GOOGL 2015-04-17  862.6000  867.6000  551.1600  565.000  0  very low
GOOGL 2015-04-18  867.6000  872.6000  551.1600  565.000  0  very low
GOOGL 2015-04-19  872.6000  877.6000  551.1600  565.000  0  very low
GOOGL 2015-04-20  877.6000  882.6000  551.1600  565.000  0  very low
GOOGL 2015-04-21  882.6000  887.6000  551.1600  565.000  0  very low
GOOGL 2015-04-22  887.6000  892.6000  551.1600  565.000  0  very low
GOOGL 2015-04-23  892.6000  897.6000  551.1600  565.000  0  very low
GOOGL 2015-04-24  897.6000  902.6000  551.1600  565.000  0  very low
GOOGL 2015-04-25  902.6000  907.6000  551.1600  565.000  0  very low
GOOGL 2015-04-26  907.6000  912.6000  551.1600  565.000  0  very low
GOOGL 2015-04-27  912.6000  917.6000  551.1600  565.000  0  very low
GOOGL 2015-04-28  917.6000  922.6000  551.1600  565.000  0  very low
GOOGL 2015-04-29  922.6000  927.6000  551.1600  565.000  0  very low
GOOGL 2015-04-30  927.6000  932.6000  551.1600  565.000  0  very low
GOOGL 2015-05-01  932.6000  937.6000  551.1600  565.000  0  very low
GOOGL 2015-05-02  937.6000  942.6000  551.1600  565.000  0  very low
GOOGL 2015-05-03  942.6000  947.6000  551.1600  565.000  0  very low
GOOGL 2015-05-04  947.6000  952.6000  551.1600  565.000  0  very low
GOOGL 2015-05-05  952.6000  957.6000  551.1600  565.000  0  very low
GOOGL 2015-05-06  957.6000  962.6000  551.1600  565.000  0  very low
GOOGL 2015-05-07  962.6000  967.6000  551.1600  565.000  0  very low
GOOGL 2015-05-08  967.6000  972.6000  551.1600  565.000  0  very low
GOOGL 2015-05-09  972.6000  977.6000  551.1600  565.000  0  very low
GOOGL 2015-05-10  977.6000  982.6000  551.1600  565.000  0  very low
GOOGL 2015-05-11  982.6000  987.6000  551.1600  565.000  0  very low
GOOGL 2015-05-12  987.6000  992.6000  551.1600  565.000  0  very low
GOOGL 2015-05-13  992.6000  997.6000  551.1600  565.000  0  very low
GOOGL 2015-05-14  997.6000  1002.6000  551.1600  565.000  0  very low
GOOGL 2015-05-15  1002.6000  1007.6000  551.1600  565.000  0  very low
GOOGL 2015-05-16  1007.6000  1012.6000  551.1600  565.000  0  very low
GOOGL 2015-05-17  1012.6000  1017.6000  551.1600  565.000  0  very low
GOOGL 2015-05-18  1017.6000  1022.6000  551.1600  565.000  0  very low
GOOGL 2015-05-19  1022.6000  1027.6000  551.1600  565.000  0  very low
GOOGL 2015-05-20  1027.6000  1032.6000  551.1600  565.000  0  very low
GOOGL 2015-05-21  1032.6000  1037.6000  551.1600  565.000  0  very low
GOOGL 2015-05-22  1037.6000  1042.6000  551.1600  565.000  0  very low
GOOGL 2015-05-23  1042.6000  1047.6000  551.1600  565.000  0  very low
GOOGL 2015-05-24  1047.6000  1052.6000  551.1600  565.000  0  very low
GOOGL 2015-05-25  1052.6000  1057.6000  551.1600  565.000  0  very low
GOOGL 2015-05-26  1057.6000  1062.6000  551.1600  565.000  0  very low
GOOGL 2015-05-27  1062.6000  1067.6000  551.1600  565.000  0  very low
GOOGL 2015-05-28  1067.6000  1072.6000  551.1600  565.000  0  very low
GOOGL 2015-05-29  1072.6000  1077.6000  551.1600  565.000  0  very low
GOOGL 2015-05-30  1077.6000  1082.6000  551.1600  565.000  0  very low
GOOGL 2015-05-31  1082.6000  1087.6000  551.1600  565.000  0  very low
GOOGL 2015-06-01  1087.6000  1092.6000  551.1600  565.000  0  very low
GOOGL 2015-06-02  1092.6000  1097.6000  551.1600  565.000  0  very low
GOOGL 2015-06-03  1097.6000  1102.6000  551.1600  565.000  0  very low
GOOGL 2015-06-04  1102.6000  1107.6000  551.1600  565.000  0  very low
GOOGL 2015-06-05  1107.6000  1112.6000  551.1600  565.000  0  very low
GOOGL 2015-06-06  1112.6000  1117.6000  551.1600  565.000  0  very low
GOOGL 2015-06-07  1117.6000  1122.6000  551.1600  565.000  0  very low
GOOGL 2015-06-08  1122.6000  1127.6000  551.1600  565.000  0  very low
GOOGL 2015-06-09  1127.6000  1132.6000  551.1600  565.000  0  very low
GOOGL 2015-06-10  1132.6000  1137.6000  551.1600  565.000  0  very low
GOOGL 2015-06-11  1137.6000  1142.6000  551.1600  565.000  0  very low
GOOGL 2015-06-12  1142.6000  1147.6000  551.1600  565.000  0  very low
GOOGL 2015-06-13  1147.6000  1152.6000  551.1600  565.000  0  very low
GOOGL 2015-06-14  1152.6000  1157.6000  551.1600  565.000  0  very low
GOOGL 2015-06-15  1157.6000  1162.6000  551.1600  565.000  0  very low
GOOGL 2015-06-16  1162.6000  1167.6000  551.1600  565.000  0  very low
GOOGL 2015-06-17  1167.6000  1172.6000  551.1600  565.000  0  very low
GOOGL 2015-06-18  1172.6000  1177.6000  551.1600  565.000  0  very low
GOOGL 2015-06-19  1177.6000  1182.6000  551.1600  565.000  0  very low
GOOGL 2015-06-20  1182.6000  1187.6000  551.1600  565.000  0  very low
GOOGL 2015-06-21  1187.6000  1192.6000  551.1600  565.000  0  very low
GOOGL 2015-06-22  1192.6000  1197.6000  551.1600  565.000  0  very low
GOOGL 2015-06-23  1197.6000  1202.6000  551.1600  565.000  0  very low
GOOGL 2015-06-24  1202.6000  1207.6000  551.1600  565.000  0  very low
GOOGL 2015-06-25  1207.6000  1212.6000  551.1600  565.000  0  very low
GOOGL 2015-06-26  1212.6000  1217.6000  551.1600  565.000  0  very low
GOOGL 2015-06-27  1217.6000  1222.6000  551.1600  565.000  0  very low
GOOGL 2015-06-28  1222.6000  1227.6000  551.1600  565.000  0  very low
GOOGL 2015-06-29  1227.6000  1232.6000  551.1600  565.000  0  very low
GOOGL 2015-06-30  1232.6000  1237.6000  551.1600  565.000  0  very low
GOOGL 2015-07-01  1237.6000  1242.6000  551.1600  565.000  0  very low
GOOGL 2015-07-02  1242.6000  1247.6000  551.1600  565.000  0  very low
GOOGL 2015-07-03  1247.6000  1252.6000  551.1600  565.000  0  very low
GOOGL 2015-07-04  1252.6000  1257.6000  551.1600  565.000  0  very low
GOOGL 2015-07-05  1257.6000  1262.6000  551.1600  565.000  0  very low
GOOGL 2015-07-06  1262.6000  1267.6000  551.1600  565.000  0  very low
GOOGL 2015-07-07  1267.6000  1272.6000  551.1600  565.000  0  very low
GOOGL 2015-07-08  1272.6000  1277.6000  551.1600  565.000  0  very low
GOOGL 2015-07-09  1277.6000  1282.6000  551.1600  565.000  0  very low
GOOGL 2015-07-10  1282.6000  1287.6000  551.1600  565.000  0  very low
GOOGL 2015-07-11  1287.6000  1292.6000  551.1600  565.000  0  very low
GOOGL 2015-07-12
```

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:

```
145  #----- Question 10
146  #-----
147  #-----
148
149  numerical_columns = ['open', 'high', 'low', 'close', 'volume']
150  google_num = google_df[numerical_columns]
151  mean_vector = np.mean(google_num, axis=0)
152  centered_data = google_num - mean_vector
153  covariance_matrix = np.dot(centered_data.T, centered_data) / (len(google_num) - 1)
154  covariance_matrix_df = pd.DataFrame(covariance_matrix, columns=numerical_columns, index=numerical_columns)
155  print("Covariance Matrix for Numerical Variables:")
156  print(covariance_matrix_df.round(3))
157
```

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  2.194388e+04 -3.854308e+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 #-----#
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  2.196792e+04  2.208068e+04  2.189256e+04  2.195137e+04 -2.959788e+07
high  2.208068e+04  2.205988e+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
88:51  LF  UTF-8  4 spaces  /Users/juliegayachari/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.