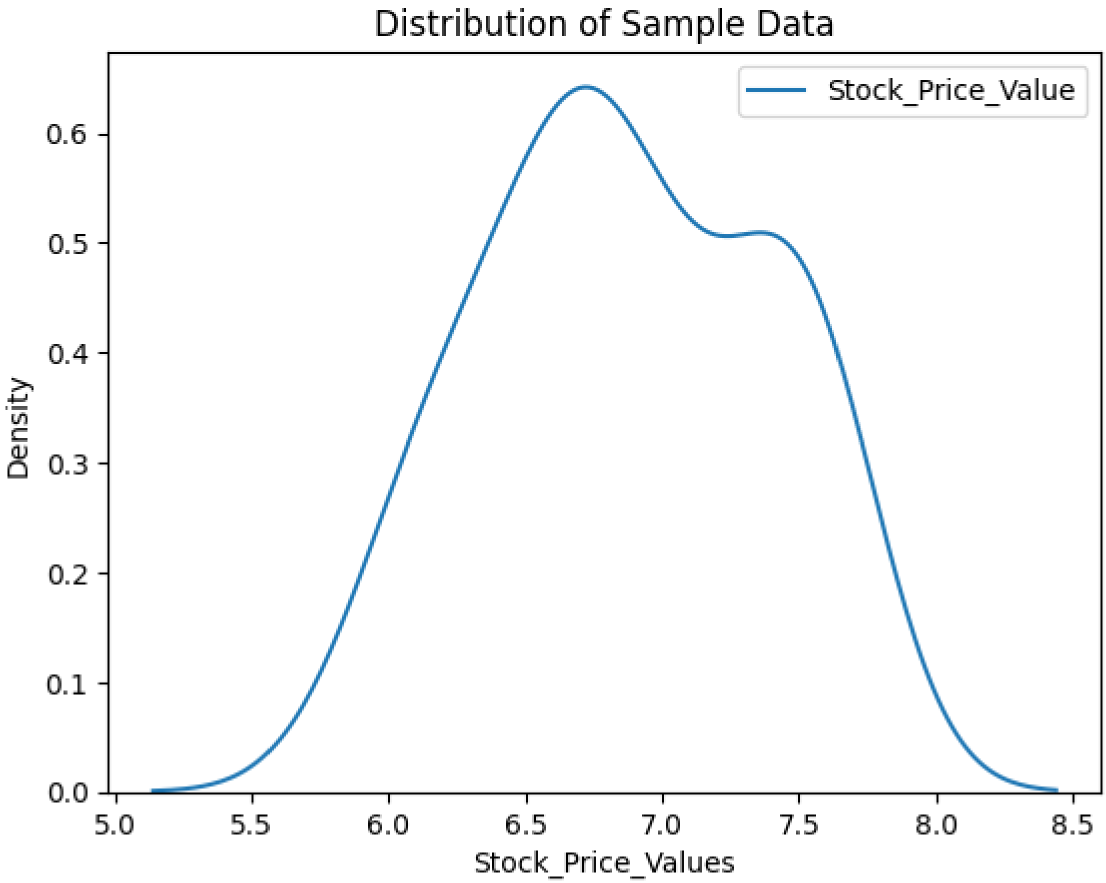
```
#Importing the Required Linraries for the Data Analysis and Visualisation
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import drive
from google.colab import files
import os
import zipfile


#Created Function process_uploaded_files to return 30 sample data randomly with Err
#process_uploaded_files create a new input file For function2 to detect the outlier

def process_uploaded_files():
  try:
    #Upload files to Colab
        uploaded = files.upload()
        # Create a temporary directory to hold the files
        directory_path = '/content/uploaded_files'
        if not os.path.exists(directory_path):
            os.makedirs(directory_path)
        # Save the uploaded files to the directory
        for filename in uploaded.keys():
            file_path = os.path.join(directory_path, filename)
            with open(file_path, 'wb') as f:
                f.write(uploaded[filename])
        # Validate the directory and files
        if not os.path.isdir(directory_path):
            raise FileNotFoundError(f"Directory '{directory_path}' does not exist."
        print(f"Directory '{directory_path}' exists.")

        # Get all files in the directory
        files_in_directory = [file for file in os.listdir(directory_path) if os.pat
        if not files_in_directory:
            raise FileNotFoundError("No files found in the specified directory.")
        print(f"{len(files_in_directory)} file(s) found in the directory.")

        # Process each file
        for file in files_in_directory:
            file_path = os.path.join(directory_path, file)
            # Check if file is not empty
            if os.path.getsize(file_path) == 0:
                raise ValueError(f"The file '{file_path}' is empty.")

            # Check if the file is in CSV format
            if not file_path.endswith('.csv'):
                raise ValueError(f"The file '{file_path}' is not in CSV format.")

            #Load the data
            #Create Headers
            headers = ["Stock_ID", "Timestamp", "Stock_Price_Value"]

            #Read the csv File
            df = pd.read_csv(file_path, names = headers, header = None)

            # Check if file has at least 30 data points
            if len(df) < 30:
                raise ValueError(f"The file '{file_path}' does not have the require

            print(f"File '{file_path}' passed all checks and is ready for processin

            #Randomly picking 30 values from given CSV, random state ensures the re
            random_rows = df.sample(n=30)

            #converting the output file into csv for the 2nd function to detect out
            input_2ndfunc = random_rows.to_csv("/content/Input_to_2ndFunc.csv", ind

            # Call detect_outliers function and pass the selected random_rows
            detect_outliers(random_rows)

  except Exception as e:
        print(f"Error: {e}")

  return input_2ndfunc


def detect_outliers(df):
  #Calculate Mean and Std Dev for Population (Complete Dataset)
  mu = df['Stock_Price_Value'].mean()
  sigma = df['Stock_Price_Value'].std()

  #Detecting outliers
  outliers_right = mu + (2*sigma)
  outliers_left = mu - (2*sigma)

  #Adding Actual Stock Price mean to the dataframe
  df2['actual_stock_price_mean'] = round(mu,2)

  #Calculate if the given value is falling outside the 2 Std Dev Range and mark it
  df2['Outliers_Found'] = np.where((df2['Stock_Price_Value'] > outliers_right) |

  #Calculate Mean for Sample (30 Data Points)
  mu1 = df2['Stock_Price_Value'].mean()

  #Adding Actual Stock Price mean to the dataframe
  df2['mean_of_30_data_points'] = round(mu1,2)

  '''As per the CLT(Central Limit Theory), the population and sample mean is tend
  as sample size increases'''

  #Calculate the PercentageofDeviation
  df2['%_ofDeviation'] = df2.apply(lambda x: round((((x['Stock_Price_Value'] - mu

  #Creating final output Values
  final_output = df2.to_csv("/content/Final_Output.csv", index=False)
  return final_output
```

```
process_uploaded_files()
```

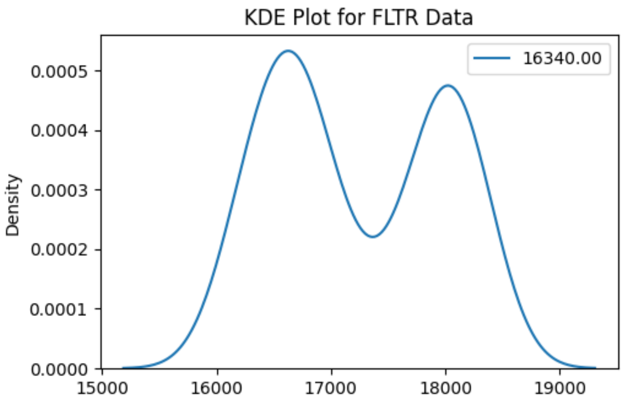
Show hidden output

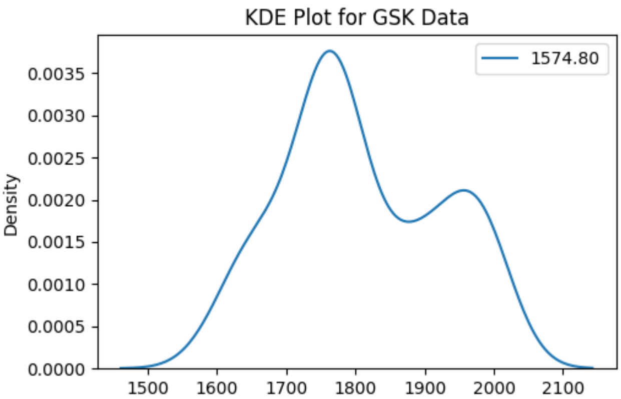
```
#Understand the distribution of Data for the sample taken
df2 = pd.read_csv("/content/Input_to_2ndFunc.csv")
#Draw a KDE Plot to Understand the Distribution
sns.kdeplot(data = df2)
plt.xlabel("Stock_Price_Values")
plt.title("Distribution of Sample Data")
plt.savefig("/content/kde_distribution_for_30 sample")
```

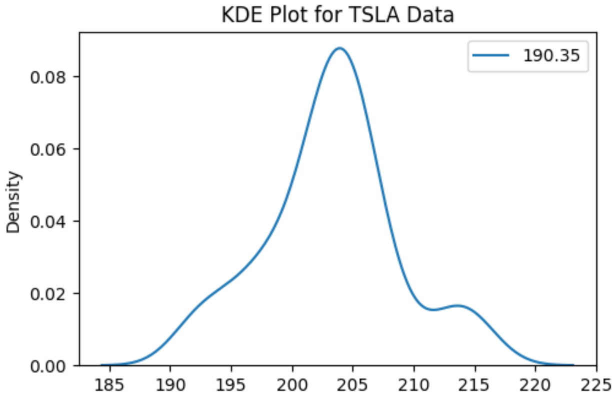


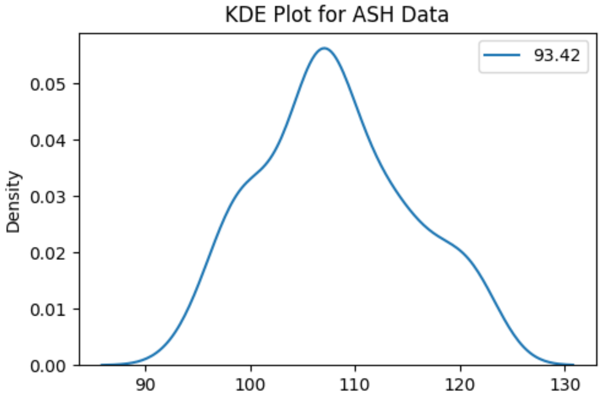
```
#Create Population Distribution plot for all Stock files

#Read Data
#Subplot1 - LSE --> FLTR,GSK
fltr = pd.read_csv("/content/FLTR LSE.csv")
gsk = pd.read_csv("/content/GSK LSE.csv")

#Subplot2 - NASDAQ --> TSLA
tsla = pd.read_csv("/content/TSLA.csv")

#Subplot3 - NYSE --> ASH, NMR
ash = pd.read_csv("/content/ASH.csv")
nmr = pd.read_csv("/content/NMR.csv")

#Create a 3*2 grid for subplots
fig, axes = plt.subplots(3,2,figsize=(12,12))
```

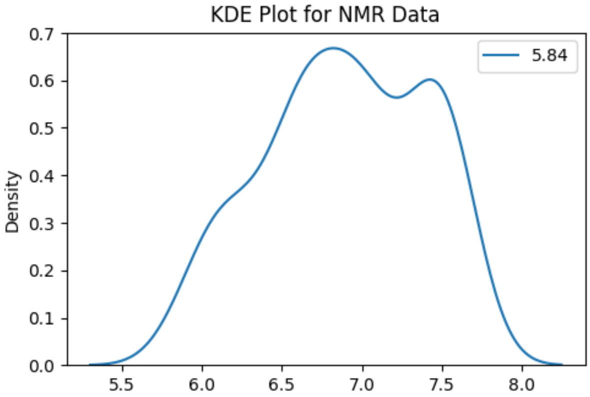
```
#1st Row, 1st Column - LSE : Plot KDEs for FLTR Data
sns.kdeplot(fltr, ax = axes[0,0])
axes[0,0].set_title("KDE Plot for FLTR Data") #Title for first subplot

#1st Row, 2nd Column - LSE : Plot KDEs for GSK Data
sns.kdeplot(gsk, ax = axes[0,1])
axes[0,1].set_title("KDE Plot for GSK Data") #Title for second subplot

#2nd Row, 1st Column - Nasdaq : Plot KDE for TSLA Data
sns.kdeplot(tsla, ax = axes[1,0])
axes[1,0].set_title("KDE Plot for TSLA Data") #Title for third subplot
axes[1,1].remove()

#3rd Row, 1st Column - NYSE : Plot KDE for ASH Data
sns.kdeplot(ash, ax = axes[2,0])
axes[2,0].set_title("KDE Plot for ASH Data") #Title for fourth subplot

#3rd Row, 2nd Column - NYSE : Plot KDE for NMR Data
sns.kdeplot(nmr, ax = axes[2,1])
axes[2,1].set_title("KDE Plot for NMR Data") #Title for fifth subplot

plt.savefig("/content/kde_distribution_for_population_data_all_stock_files")
```


KDE Plot for FLTR Data
16340.00
Density
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
15000
16000
17000
18000
19000


KDE Plot for GSK Data
1574.80
Density
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
1500
1600
1700
1800
1900
2000
2100


KDE Plot for TSLA Data
190.35
Density
0.08
0.06
0.04
0.02
0.00
185
190
195
200
205
210
215
220
225


KDE Plot for ASH Data
93.42
Density
0.05
0.04
0.03
0.02
0.01
0.00
90
100
110
120
130


KDE Plot for NMR Data
5.84
Density
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
5.5
6.0
6.5
7.0
7.5
8.0