Portfolio Exercise 1

Introduction

This exercise has few Tasks, which you need to solve using python programming (jupyter notebook). The exercise tasks are based on the learning lecture/tutorial (1 to 3).

Scenario: Stocks and Trades

Dow Jones Industrial Average (DJIA) is a measure that tracks the performance of 30 large publicly traded companies in the United States. It is one of the oldest and most well-known stock market indexes in the world. The DJIA is calculated by adding up the stock prices of the 30 companies in the index and dividing the total by a number called the "Dow Divisor." This number is adjusted from time to time to account for changes in the index, such as when a company splits its stock or when a company is added or removed from the index. The Dow is often used as a way to measure the overall health of the stock market and the economy. When the Dow goes up, it usually means that the companies in the index are doing well and that investors are feeling confident about the future. On the other hand, when the Dow goes down, it can be a sign that the companies in the index are facing challenges or that investors are worried about the future. The data we have describes the DJIA each month between 1914 and 1968 (648 records). The data is structured as follows:

Date	Price
1914-12-01	55.00
1915-01-01	56.55
1915-02-01	56.00
….	….
1968-12-01	965.39

Table 1: The structure of the DJIA data.

For this exercise you must write the code needed to analyse this data, and then execute that code to return useful information. This information can then be interpreted to reveal insights.

Required Libraries and Sample Starting Code:

In the following program segment, a piece of code is given to get you started. It loads the libraries you'll need to complete the exercise. There are four that I've imported for you:

```
# These are the packages you'll need to undertake your analysis of the data described in the coursework document.

Import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
# Loads the DJIA data into a variable for you to use.
Djia_data = sns.load_dataset(‘dowjones’)

# This will print out the rough structure of the data
# we just loaded into the djia_data variable.
Print(djia_data)
```
…………………………………………………………..

Pandas: is a library for data manipulation and analysis in Python. It provides data structures for efficiently storing large amounts of data and tools for working with that data in a variety of ways, such as calculating statistics, filtering for specific rows or columns, and creating plots.

Matplotlib: is a library for creating static, animated, and interactive visualizations in Python. It is the most widely used data visualization library in the Python data science ecosystem and is a powerful tool for exploring and understanding data.

Seaborn: is a library for creating statistical plots in Python. It is built on top of the popular data visualization library Matplotlib but makes it easier to create a wider range of plots and to customize them in more sophisticated ways.

NumPy: is a library for working with large, multi-dimensional arrays and matrices of numerical data in Python. It provides tools for performing mathematical operations on these arrays, such as calculating statistics, finding patterns, and applying functions to each element in the array. NumPy is an essential library for many applications in data science and scientific computing, and is often used in conjunction with Pandas, Seaborn, and Matplotlib.

Note: The sample code creates a variable called `djia_data` which contains the DJIA data. The variable is of a special data type. It is a two-dimensional tabular data structure with rows and columns. It is similar to a spreadsheet or an SQL table. You can think of a DataFrame as a collection of rows, each of which is a collection of columns (We have already learned about DataFrame, and how to load data). The data is built in the seaborn library.

Task 1

Write the code needed to calculate and print out:

1. The mean of the DJIA.
2. The median of the DJIA.
3. The standard deviation of the DJIA.
4. The minimum value of the DJIA.
5. The maximum value of the DJIA.

This code should be written in notebook cell in an appropriate place.

Hint: This task can be easily solved using the built-in functions of the `dija_data` **DataFrame** variable, to calculate these values. With a little bit of research, and by looking at the data frame ***Application Programming Interface (API)***, you'll be able to complete this task. The data frame API can be found here:

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html#pandas.DataFrame

Task 2:

Write the code that will create a box plot for your data, using the boxplot function built into the 'DataFrame' variable `djia_data`. Ensure the plot has,

1. A meaningful title.
2. A meaningful x-axis label.

Notes: A box plot, also known as a box and whisker plot, is a graphical representation of the statistical distribution of a dataset. It is commonly used to visualize the spread and skewness of the data, and to identify potential outliers.
A box plot consists of a box, which represents the interquartile range (IQR) of the data, and a set of vertical lines called "whiskers" that extend from the box to the minimum and maximum values of the data. The box is divided into two parts by a horizontal line called the "median," which represents the middle value of the data.

Task 3

*Write the code that will create a violin plot for your data using the violin plot function built into **seaborn**.* Find the API details here (https://seaborn.pydata.org/generated/seaborn.violinplot.html) along with examples showing you how to apply it. Ensure the plot has,

1. A meaningful title.
2. Meaningful x-axis and y-axis labels.

Notes: A violin plot is a graphical representation of the distribution of a dataset. It is a way to visualize the spread and shape of the data, and to compare the distribution of the data across different categories or groups.
A violin plot looks like a "violin" shape, with a narrow centre and wider ends. The width of the violin at a given point is proportional to the number of observations at that point, so the plot gives a visual indication of the density of the data at different values. The plot also has horizontal lines called "whiskers" that extend from the violin shape to the minimum and maximum values of the data. Violin plots are similar to box plots, but they show more detailed information about the shape of the data. They can be useful for understanding how the data is distributed and for identifying patterns and trends in the data.

Task 4:

*Write the code that will create a line chart for your data using the plot function built into the **DataFrame** variable ```djia_data```. Ensure the plot has,*

1. A meaningful title.
2. Meaningful x-axis and y-axis labels.

Notes: A line chart is a graphical representation of a dataset that displays the data as a series of connected points, or dots, on a graph. The points are plotted along a horizontal (x) axis and a vertical (y) axis, and a line is drawn between the points to show the relationship between the data.

Line plots are useful for visualizing trends in data over time or across different categories. They can help you see how the data changes as the x-axis variable increases or decreases and can also show you if there are any patterns or trends in the data. Line plots are often used to compare the trends in two or more datasets, or to show the relationship between two or more variables. They are a simple and effective way to visualize data and can be useful for quickly understanding the trends and patterns in a dataset.

Task 5:

Write the code that will create a line of best fit chart for your data using the numpy ‘polyfit’ and ‘poly1d’ functions. There are a few steps to complete here:

1. Create data to represent the x-axis since you can’t fit a line of best fit directly to date time data. You can use code such as ‘x = list(range(0, 649))’ to create the x-axis data you need.
2. Calculate the coefficients using ‘polyfit’.
3. Create the line of best fit by passing the coefficients calculated above to poly1d.
4. Plot the line of best fit.

A little research will help you tackle this challenge. Be sure your plot has:

• *A meaningful title.*
• *Meaningful x-axis and y-axis labels.*

Notes: A line of best fit, is a straight line that is the best approximation of the data on a scatter plot. It is a way to visualize the relationship between two variables (in this case time and the DJIA), and to see how closely the data fits a linear pattern.

To create a line of best fit, you plot the data on a scatter plot and then draw a straight line that goes through as many points as possible. The line should be as close as possible to all of the points, without passing through any of the points. This line is the line of best fit.

The line of best fit can be used to make predictions about the value of one variable based on the value of the other variable. For example, if you have a scatter plot showing the relationship between the height and weight of a group of people, you could use the line of best fit to predict the weight of someone based on their height.

Evaluation of Exercise 1:

Evaluate all the tasks 1 to 5 solutions and consider the following questions to answer them in writing your evaluation:

1. What the data summary statistics are that you calculated and what they mean.
2. What the box plot is showing you.
3. What the violin plot is showing you.
4. What the line chart is showing you.
5. What the line of best fit is indicating.
6. What may have happened between months 180 – 220?