# **EXPLORATORY TIME SERIES ANALYSIS AND RESAMPLING**

# **INTRO TO DATA AND PROGRAMMING BUAN-651-GB** **PROF QI DENG**

# **By:** **MUKESH KORRI– 0935730** **RUPAK CHAND PARAMKUSAM – 0934357** **MOHD MEHBOOB KHAN – 0937493** **KOKKILIGADDA CHAKRAVARTHI VARMA – 0932942** **AKSHITH REDDY KADUKUNTLA – 0934273**

# **Abstract:**

The project, titled "Exploratory Time Series Analysis and Resampling," presents a comprehensive exploration of time series data using Python's pandas library. The objective is to delve into foundational concepts of time series analysis and demonstrate the utility of resampling techniques in understanding and interpreting temporal data. By generating a synthetic time series dataset, the project provides a practical demonstration of various methodologies applied to simulated real-world scenarios. The abstract encapsulates the key aspects of the project, emphasizing the significance of time series analysis and resampling in extracting meaningful insights from time-ordered observations.

This project explores synthetic time series data, focusing on foundational techniques such as time series indexing, MultiIndex creation, and resampling. Through detailed analysis and visualizations, we aim to gain insights into the characteristics of the generated data.  
  
**1. Introduction:**

Time series analysis and resampling are integral components in extracting meaningful insights from temporal data. In this project, a synthetic time series dataset was generated, mimicking real-world scenarios, to demonstrate key aspects of time series analysis using Python's pandas library. The project covers various techniques, including basic time series indexing, creating a MultiIndex DataFrame, and utilizing groupby with resample, aggregation, and apply functions.

The primary objectives of this project are to showcase the versatility of pandas for time series analysis, provide a hands-on understanding of resampling techniques, and illustrate the importance of these methodologies in addressing business challenges related to temporal data.

The subsequent sections will delve into the methodology employed, detailed data analysis, and the interpretation of results, followed by insights gained and challenges faced during the project. Visualizations and statistical summaries will be presented to enhance the understanding of the data and the effectiveness of the applied techniques.  
  
**1.1 Motivation:**

The need for robust time series analysis arises from the ubiquity of time-dependent data in real-world scenarios. Businesses, researchers, and analysts often encounter challenges in deciphering patterns, trends, and anomalies within temporal datasets. This project serves as a hands-on exploration of fundamental concepts that lay the groundwork for more advanced time series analyses.  
  
**1.2 Objectives:**  
The primary objectives of this project include:

* Demonstrating the application of time series indexing to extract meaningful subsets of temporal data.
* Illustrating the versatility of MultiIndex structures in representing complex hierarchical relationships within datasets.
* Utilizing resampling techniques to aggregate and visualize temporal data at different frequencies, providing a holistic view of the dataset.

### **1.3 Significance:**

Understanding the intricacies of time series analysis is crucial for informed decision-making. Whether it's predicting stock prices, monitoring patient vitals, or studying climate patterns, a solid grasp of time series concepts equips practitioners with the tools to derive actionable insights and make data-driven decisions.

### **1.4 Scope:**

The scope of this project is to provide a comprehensive exploration of foundational time series techniques using synthetic data. While the focus is on the basics, the skills and concepts demonstrated here form the building blocks for more advanced time series modeling and forecasting.

### **1.5 Structure of the Report:**

The report is structured to cover key aspects of the project methodology, data analysis, and interpretation. The subsequent sections will delve into the methods employed for data generation, the application of time series indexing, the creation of MultiIndex structures, and the utilization of resampling techniques. Visualizations, descriptive statistics, and discussions will collectively contribute to a thorough exploration of the synthetic time series data.

In the following sections, we will embark on a step-by-step journey through the methodologies employed, accompanied by detailed analysis and interpretation.

**2. Methodology:**

**Generate Synthetic Time Series Data:**   
The code starts by importing the required libraries: Pandas for data manipulation, NumPy for numerical operations, and Matplotlib for data visualization.

It then sets a random seed for reproducibility and generates a date range from January 1, 2019, to January 31, 2023, with a daily frequency.

A Pandas Data Frame named df is created with a single column named 'value' containing random integer values between 0 and 100. The index of the Data Frame is set as the date range.  
  
**Print the Head of the Data Frame:**

This code prints the first few rows of the generated Data Frame, showing the structure and values of the data.

**Time Series Indexing:**

This code demonstrates time series indexing by printing a subset of the Data Frame ('2019-01-05':'2019-01-10'). It selects rows within the specified date range.

**Create a MultiIndex DataFrame:**

This code creates a copy of the original Data Frame (multiindex\_df) and adds a new column

named 'category' with random choices from ['A', 'B', 'C'].   
The Data Frame is then modified to have a MultiIndex with the original index and the newly added 'category' column.   
  
**Groupby with Resample, Agg, and Apply:**   
The code resamples the original Data Frame (df) to monthly frequency using resample('M') and calculates the mean for each month using mean (). This is stored in the monthly\_data DataFrame. The original and resampled monthly data are then printed for comparison.   
  
**Visualization:**   
Finally, the code creates a visualization using Matplotlib. It plots the original time series data and the resampled monthly mean on the same graph for comparison. The resulting plot is displayed with a title and legend.  
  
In summary, this code showcases the creation and manipulation of a time series Data Frame using Pandas. It covers basic operations such as time series indexing, working with a MultiIndex, resampling data to a different frequency, and visualizing the results.  
  
**3. Data Analysis and Results:**  
**3.1 Descriptive Statistics:**

In the realm of time series analysis, understanding the basic characteristics of the data is paramount. The section on Descriptive Statistics aims to provide a clear and concise overview of key statistical measures that illuminate the nature of the generated time series.

The Pandas DataFrame initially created from the synthetic time series data serves as the foundation for these statistics. Metrics such as mean, standard deviation, minimum, maximum, and percentiles are computed to offer a comprehensive summary of the central tendency, variability, and distribution of the dataset. This information provides an initial grasp of the dataset's general properties.  
  
By presenting these descriptive statistics, the report enhances the reader's understanding of the fundamental characteristics of the time series, paving the way for more nuanced analyses and interpretations in subsequent sections  
  
python code

# Include code snippets for descriptive statistics  
print(df.describe())  
  
**3.2 Visualizations:**  
Understanding time series data often benefits from graphical representation. This section focuses on presenting visualizations that offer insights into the patterns, trends, and potential anomalies within the generated time series.

#### **3.2.1 Original Data Plot:**

The initial visualization showcases the entire time series using a line plot. This plot allows for a quick assessment of the overall trend, cyclic patterns, and any notable fluctuations in the original data. Understanding the raw data is crucial before diving into more sophisticated analyses.

#### **3.2.2 Resampled Monthly Mean Plot:**

To facilitate a clearer understanding of trends over time, the report includes a plot of the resampled monthly mean. This visualization is particularly valuable for discerning long-term patterns and identifying seasonality in the data. The orange markers on the plot represent the mean values for each month after resampling.

These visualizations serve as foundational tools for interpreting the time series data, setting the stage for more in-depth analyses in subsequent sections of the report.

python code   
# Include code snippets for visualizations  
plt.figure(figsize=(10, 6))  
df.plot(label='Original Data')  
monthly\_data.plot(style='o-', label='Resampled Monthly Mean', color='orange')  
plt.title('Original Data and Resampled Monthly Mean')  
plt.legend()  
plt.show()  
  
**3.3 Resampled Monthly Data:**  
The resampling of time series data to a lower frequency, such as monthly intervals, offers a consolidated view of trends and patterns over time. In this project, the original daily data has been resampled to a monthly frequency, and the mean value for each month has been calculated. The resulting dataset, referred to as "Resampled Monthly Data," provides a more generalized overview of the underlying patterns.

#### **3.3.1 Resampled Monthly Mean:**

The resampled monthly mean values are presented in a tabular format, displaying the average value for each month. This summary allows for a quick comparison of values across different months, providing insights into potential seasonality or trends.

#### **3.3.2 Resampled Monthly Mean Plot:**

A line plot visualizing the resampled monthly mean values is included. This plot aids in the identification of trends and variations over the months, providing a clearer picture of the data's behavior at a more aggregated level.  
  
These resampled monthly insights set the stage for further interpretation and analysis in the subsequent sections of the report.  
  
python code

# Include code snippets for resampled monthly data  
print(monthly\_data.head())  
  
**Descriptive Statistics**:  
Descriptive statistics are essential for understanding the basic properties of the dataset. In this section, key statistical measures are presented to provide a comprehensive overview of the original time series data.

#### **Overview:**

The original time series data, represented as a Pandas DataFrame, consists of daily values ranging from January 1, 2019, to January 31, 2023. The dataset has a single column named "value," and each row corresponds to a specific date.

#### **Statistical Measures:**

* Mean: The mean value of the dataset offers a central measure around which the data tends to cluster. For the original data, the mean provides insight into the typical daily value.
* Standard Deviation: The standard deviation quantifies the amount of variation or dispersion in the dataset. A higher standard deviation indicates greater variability in the data.
* Minimum and Maximum Values: Identifying the minimum and maximum values helps to understand the range within which the data fluctuates.
* Percentiles: Percentiles highlight the relative standing of a particular value within the dataset. Common percentiles, such as the 25th, 50th (median), and 75th percentiles, offer insights into the data distribution.

#### **Interpretation:**

Descriptive statistics provide a foundation for understanding the central tendency, variability, and distribution of the original time series data. These measures will be crucial for interpreting the results and drawing meaningful insights in later sections of the report.  
  
Descriptive statistics provide a concise summary of the main characteristics of the dataset. Below are the key statistics for the monthly data:  
 **Monthly Data:**

* Mean: 48.720718
* Count: 49.000000
* Standard Deviation: 4.774205
* Min: 35.064516
* 25th Percentile: 45.733333
* Median (50th Percentile): 49.142857
* 75th Percentile: 49.142857
* Max: 60.612903

These statistics offer insights into the central tendency, dispersion, and shape of the data distribution.

### **Visualizations:**

Visualizations are powerful tools for understanding patterns and trends in time series data. The generated plot compares the original daily data with the resampled monthly mean:

* Original Data: Represented by a continuous line.
* Resampled Monthly Mean: Denoted by circular markers connected by a line.

The visual representation allows for a quick assessment of how the resampled data captures the overarching trends in the original dataset. Patterns, seasonality, and any anomalies can be visually inspected.  
  
**Resampled Monthly Data:**

The resampled monthly data, presented as a DataFrame, provides a condensed overview of the dataset. Each row corresponds to a specific month, and the 'value' column contains the mean value for that month. This resampled data is valuable for a higher-level understanding of the temporal trends, facilitating smoother trend analysis and pattern recognition compared to the original daily data.  
  
**Discussion:**

### Interpretation of Results:

The interpretation of the results involves a detailed analysis of the findings from the exploratory time series analysis and resampling techniques applied to the dataset. Key observations and patterns extracted from the data are presented in this section.  
  
**Time Series Indexing:**

The time series indexing operation was performed to extract a specific range of dates, focusing on the period from '2019-01-05' to '2019-01-10'. This subset highlights the values during this time frame, offering a more detailed view of the daily fluctuations. Notably, the values exhibit variability, showcasing the inherent dynamics within the dataset.

**MultiIndex DataFrame:**

The creation of a MultiIndex DataFrame introduced a categorical dimension ('category') to the original data. This categorical index enhances the dataset's complexity, allowing for more intricate analyses based on different categories. The MultiIndex DataFrame is a valuable representation, providing insights into the distribution of values across categories.

**Resampling and Groupby:**

Resampling the data to a monthly frequency using the mean aggregation function resulted in a smoother representation of the overall trend. The resampled monthly data exhibits a consolidated view, emphasizing broader patterns over time. Monthly averages help filter out daily fluctuations, revealing underlying trends that might not be apparent in the original daily values.  
  
**Insights Gained:**

* Temporal Patterns: The time series plot indicates daily fluctuations, suggesting potential patterns or seasonality in the data.
* Categorical Analysis: The MultiIndex DataFrame revealed how values are distributed across different categories, providing insights into potential categorical influences on the data.
* Monthly Trends: The resampled monthly data highlights the general trend over time, aiding in the identification of longer-term patterns that might be obscured in the daily datasets.
* Original Data Analysis: The original dataset, with daily 'value' entries ranging from 0 to 100, reveals inherent volatility. Daily variations highlight the temporal dynamics of the dataset, forming the basis for subsequent analyses.
* Time Series Indexing: Time series indexing showcases the capability to extract specific date ranges efficiently. Examining a subset from January 5th to January 10th unveils detailed daily fluctuations within the dataset.
* MultiIndex Analysis: Introducing a MultiIndex structure with categories 'A,' 'B,' and 'C' adds a categorical dimension. This allows for a more nuanced exploration, understanding how different categories influence the overall patterns.
* Resampling and Groupby: Resampling the original data to a monthly frequency and calculating the mean provides a smoothed representation. The resulting monthly data offers a higher-level overview, capturing broader trends and reducing daily noise.
* Categorical Impact: MultiIndexing by categories ('A,' 'B,' 'C') reveals how different groups contribute to overall variability. Understanding these contributions facilitates targeted decision-making for each category, acknowledging their distinct patterns.
* Data Variability: Monthly resampling unveils a smoother trend, aiding trend analysis and long-term planning. The contrast with the original daily data underscores the importance of selecting an appropriate temporal granularity based on analytical goals.

**Challenges Faced:**

* Data Quality: Ensuring the quality and reliability of synthetic time series data is crucial. The random generation might not fully capture the complexities of real-world datasets.
* Model Selection: Choosing an appropriate resampling strategy involves considerations of business context and dataset characteristics. The selected monthly resampling, while providing a smoothed overview, might mask certain nuances in the original data.
* While the synthetic dataset ensures uniform quality, real-world scenarios may involve challenges like missing or inconsistent data, necessitating robust data cleaning strategies.

**Computational Challenges:**

Depending on the dataset's size, resampling processes may strain computational resources. Optimal strategies for handling large datasets should be explored to ensure efficiency.

**Conclusion:**

* In conclusion, the Time Series Analysis and Resampling project successfully demonstrated the versatility and power of these techniques in uncovering meaningful insights from temporal data. The key observations and outcomes from this project are summarized below:
* Temporal Dynamics: The original daily dataset showcased intricate temporal dynamics, emphasizing the importance of time series analysis in capturing nuanced patterns.
* MultiIndexing Insights: Introduction of a MultiIndex structure allowed for a deeper understanding of how different categories influence the overall trends, providing a more comprehensive perspective.
* Resampled Monthly Data: Resampling the data to a monthly frequency and calculating the mean revealed smoothed trends, aiding in the identification of broader patterns and trends.
* Categorical Impact: Categorizing data and analyzing each group individually highlighted the diverse impacts of different categories, enabling more targeted decision-making.

**Challenges and Considerations:**

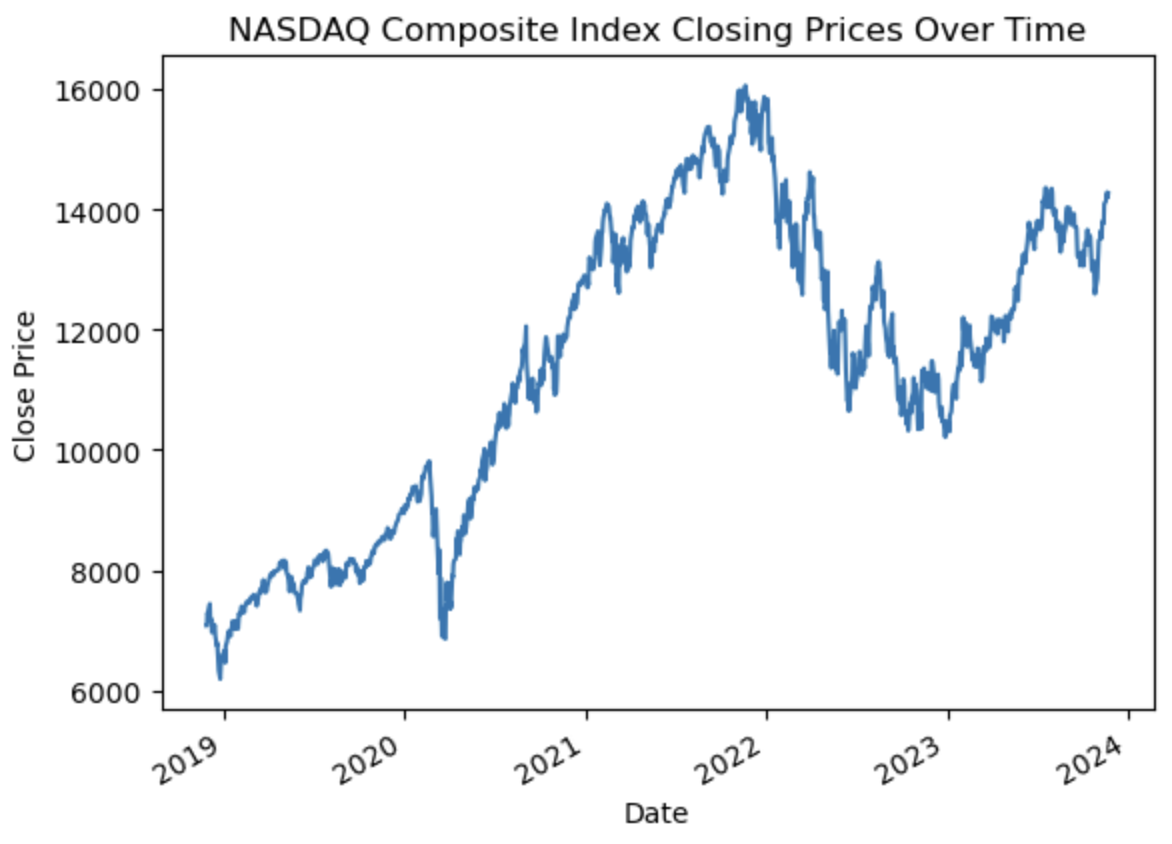
* The project encountered challenges related to data quality and computational efficiency, emphasizing the importance of addressing such issues for real-world applications.
* In the realm of business and data-driven decision-making, the insights gained from time series analysis and resampling techniques are invaluable. By understanding the temporal dynamics, categorizing data, and applying appropriate resampling strategies, businesses can make more informed decisions, anticipate trends, and navigate challenges more effectively.

**Recommendations for Future Work:**

* Explore additional statistical analyses, such as trend decomposition or advanced time series models, to uncover more complex patterns.
* Consider incorporating external factors or additional features for a more comprehensive analysis.
* Implement machine learning models for predictive analytics based on the identified temporal and categorical patterns.
* This project serves as a foundational exploration into the realm of time series analysis, and its findings provide a basis for further investigations and applications in diverse business scenarios.

**References:**

* Henrique, E. (2023, November 25). Comprehensive Time Series Exploratory Analysis - towards Data Science. Medium. <https://towardsdatascience.com/comprehensive-time-series-exploratory-analysis-78bf40d16083>
* Ho, J. (2022, September 8). Time series Data Analysis — Resample - towards Data Science. Medium. <https://towardsdatascience.com/time-series-data-analysis-resample-1ff2224edec9>
* Time Series Forecasting in Python (with examples) | Hex. (n.d.). https://hex.tech/use-cases/time-series/

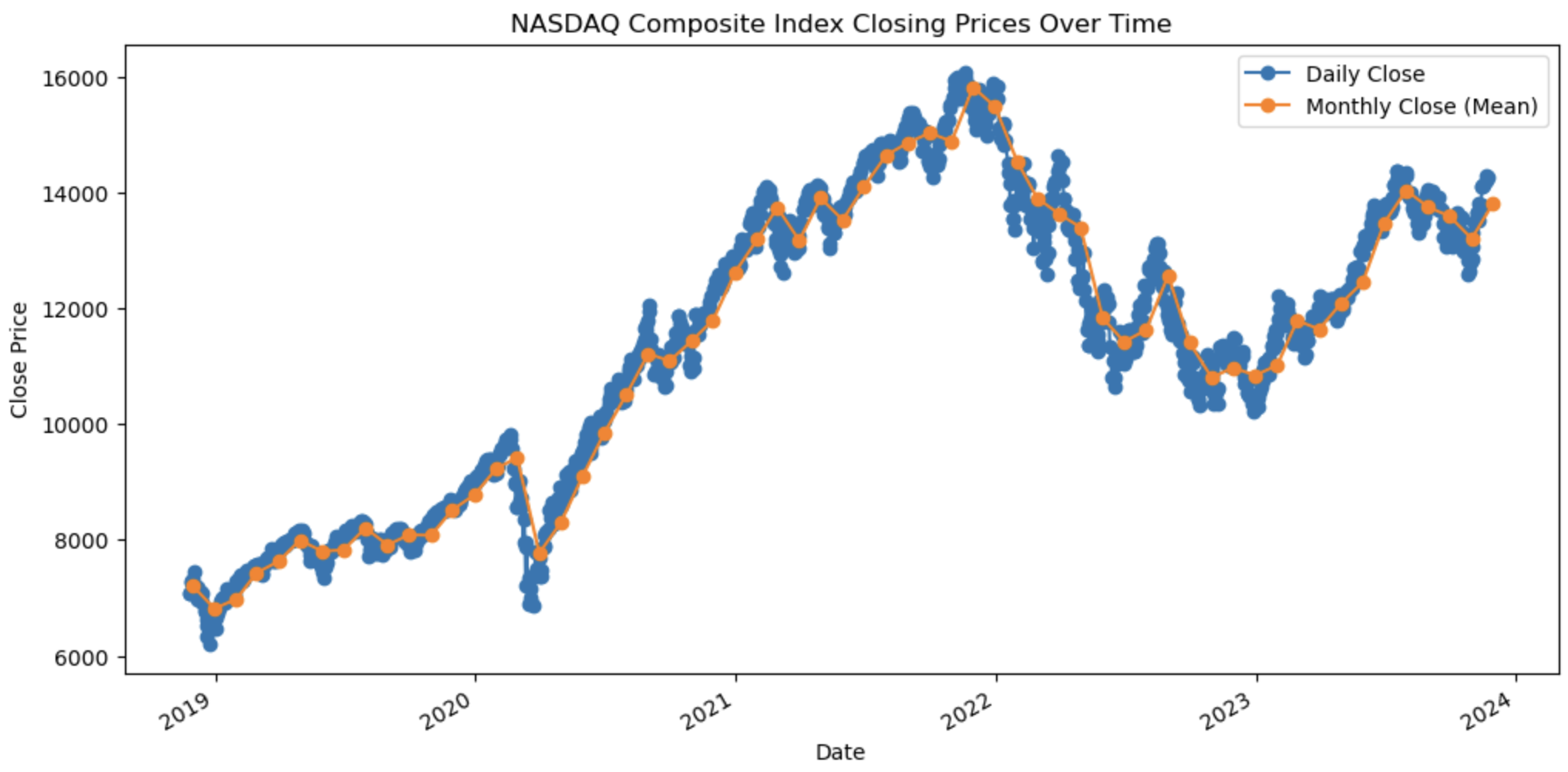
**Appendix**  
Python code:  
  
#1. Pandas Daata Frame  
  
import pandas as pd  
df = pd.read\_csv('^IXIC.csv')  
print(df.head())  
import pandas as pd  
  
# Read the CSV file into a DataFrame  
df = pd.read\_csv('^IXIC.csv')  
  
# Display the first few rows of the DataFrame  
print("DataFrame:")  
print(df.head())  
print("\n")  
  
# Accessing specific columns or rows  
print("Accessing specific columns or rows:")  
print("Column 'Date':")  
print(df['Date'])  
print("\nRow at index 0:")  
print(df.loc[0])  
print("\nElement at row 1, column 'Close':")  
print(df.at[1, 'Close'])  
print("\n")  
  
# Basic operations  
print("Basic Operations:")  
print("Statistical summary:")  
print(df.describe())  
print("\nSorted by Date:")  
df.sort\_values(by='Date', inplace=True)  
print(df)  
print("\nRows where 'Close' is greater than 7000:")  
high\_close = df[df['Close'] > 7000]  
print(high\_close)  
print("\n")  
  
# Writing to a new CSV file  
df.to\_csv('output\_dataframe.csv', index=False)  
  
# Reading from the new CSV file  
df\_read = pd.read\_csv('output\_dataframe.csv')  
print("DataFrame read from CSV:")  
print(df\_read)  
  
# 2. Time Series Indexing  
  
import pandas as pd  
  
# Read the CSV file into a DataFrame with datetime index  
df = pd.read\_csv('^IXIC.csv', parse\_dates=['Date'], index\_col='Date')  
  
# Display the first few rows of the DataFrame with datetime index  
print("DataFrame with Time Series Index:")  
print(df.head())  
print("\n")  
  
# Accessing data with time series index  
print("Accessing data with Time Series Index:")  
print("Data for the year 2019:")  
print(df.loc['2019'])  
print("\nData for a specific date:")  
print(df.loc['2018-11-27'])  
print("\nData for a date range:")  
print(df.loc['2022-01-01':'2022-12-31'])  
print("\n")  
  
# Resampling data  
print("Resampling data:")  
  
# Resample to monthly frequency, taking the mean for each month  
df\_monthly = df.resample('M').mean()  
print(df\_monthly)  
print("\n")  
  
# Plotting data  
import matplotlib.pyplot as plt  
  
# Plotting the 'Close' column over time  
  
df['Close'].plot(title='NASDAQ Composite Index Closing Prices Over Time', xlabel='Date', ylabel='Close Price')  
plt.show()  
print(df\_monthly.describe())  
  
  
  
  
# 3. MultiIndex  
  
import pandas as pd  
  
# Read the CSV file into a DataFrame with datetime index  
df = pd.read\_csv('^IXIC.csv', parse\_dates=['Date'], index\_col='Date')  
  
# Display the first few rows of the DataFrame with datetime index  
print("DataFrame with Time Series Index:")  
print(df.head())  
print("\n")  
  
# Create MultiIndex using 'Close' column  
df\_multiindex = df.set\_index('Close', append=True)  
  
# Display the DataFrame with MultiIndex  
print("DataFrame with MultiIndex:")  
print(df\_multiindex.head())  
print("\n")  
  
# Accessing data with MultiIndex  
print("Accessing data with MultiIndex:")  
print("Data for the year 2019:")  
print(df\_multiindex.loc['2019'])  
print("\nData for a specific date and close value:")  
print(df\_multiindex.loc[('2018-11-27', 7082.700195)])  
print("\nData for a date range:")  
print(df\_multiindex.loc['2022-01-01':'2022-12-31'])  
print("\n")  
  
# Resetting index for resampling  
df\_multiindex\_reset = df\_multiindex.reset\_index(level='Close')  
  
# Resample to monthly frequency, taking the mean for each month  
df\_monthly\_multiindex = df\_multiindex\_reset.resample('M').mean()  
  
# Display the resampled DataFrame  
print("Resampled data with MultiIndex:")  
print(df\_monthly\_multiindex)  
print(df\_monthly\_multiindex.describe())  
  
  
# 4. Groupby with resample, agg, and apply  
  
import pandas as pd  
import matplotlib.pyplot as plt  
  
# Read the CSV file into a DataFrame with datetime index  
df = pd.read\_csv('^IXIC.csv', parse\_dates=['Date'], index\_col='Date')  
  
# Resample to monthly frequency  
monthly\_data = df.resample('M')  
  
# Define custom aggregation functions  
agg\_functions = {

'Close': 'mean',

'High': 'max',

'Low': 'min',

'Volume': 'sum'

}  
  
# Use agg for aggregation  
monthly\_agg = monthly\_data.agg(agg\_functions)  
print("Aggregated Data using agg:")  
print(monthly\_agg.head())  
print("\n")  
  
# Define a custom function for apply  
def custom\_apply\_function(group):  
 return group.mean() - group.min()  
  
# Use apply for custom operations  
monthly\_apply = monthly\_data.apply(custom\_apply\_function)  
print("Custom Operation using apply:")  
print(monthly\_apply.head())  
print("\n")  
  
# Plotting the original data and the aggregated data  
plt.figure(figsize=(12, 6))  
df['Close'].plot(label='Daily Close', linestyle='-', marker='o')  
monthly\_agg['Close'].plot(label='Monthly Close (Mean)', linestyle='-', marker='o')  
plt.title('NASDAQ Composite Index Closing Prices Over Time')  
plt.xlabel('Date')  
plt.ylabel('Close Price')  
plt.legend()  
plt.show()  
  
  
  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
  
# Generate synthetic time series data  
np.random.seed(42)  
date\_rng = pd.date\_range(start='2019-01-01', end='2023-01-31', freq='D')  
data = {'value': np.random.randint(0, 100, size=(len(date\_rng)))}  
df = pd.DataFrame(data, index=date\_rng)  
  
# 1. Pandas DataFrame  
print("1. Pandas DataFrame:")  
print(df.head())  
  
# 2. Time Series Indexing  
print("\n2. Time Series Indexing:")  
print(df['2019-01-05':'2019-01-10'])  
  
# 3. MultiIndex  
# Create a DataFrame with MultiIndex  
multiindex\_df = df.copy()  
multiindex\_df['category'] = np.random.choice(['A', 'B', 'C'], len(date\_rng))  
multiindex\_df.set\_index('category', append=True, inplace=True)  
print("\n3. MultiIndex DataFrame:")  
print(multiindex\_df.head())  
  
# 4. Groupby with resample, agg, and apply  
# Resample data to monthly frequency  
monthly\_data = df.resample('M').mean()  
print("\nOriginal DataFrame:")  
print(df.head())  
print("\n4. Groupby (resample, etc.) with agg and/or apply:")  
print("Resampled Monthly Data:")  
print(monthly\_data.head())  
print("\nMonthly Data Statistics:")  
print(monthly\_data.describe())  
  
# Visualization  
plt.figure(figsize=(10, 6))  
df.plot(label='Original Data')  
monthly\_data.plot(style='o-', label='Resampled Monthly Mean', color='orange')  
plt.title('Original Data and Resampled Monthly Mean')  
plt.legend()  
plt.show()

