Data Exploration & Visualization Unit - 4

Dr. SELVA KUMAR S
ASSISTANT PROFESSOR
DEPT. OF CSE

Agenda

- Univariate analysis
- Bivariate analysis
- Multivariate analysis
- Time Series Analysis (TSA)
- Fundamentals of TSA - characteristics of TSA
- Time based indexing
- Visualizing time series
- Grouping and resampling time series data.

Types of Analysis

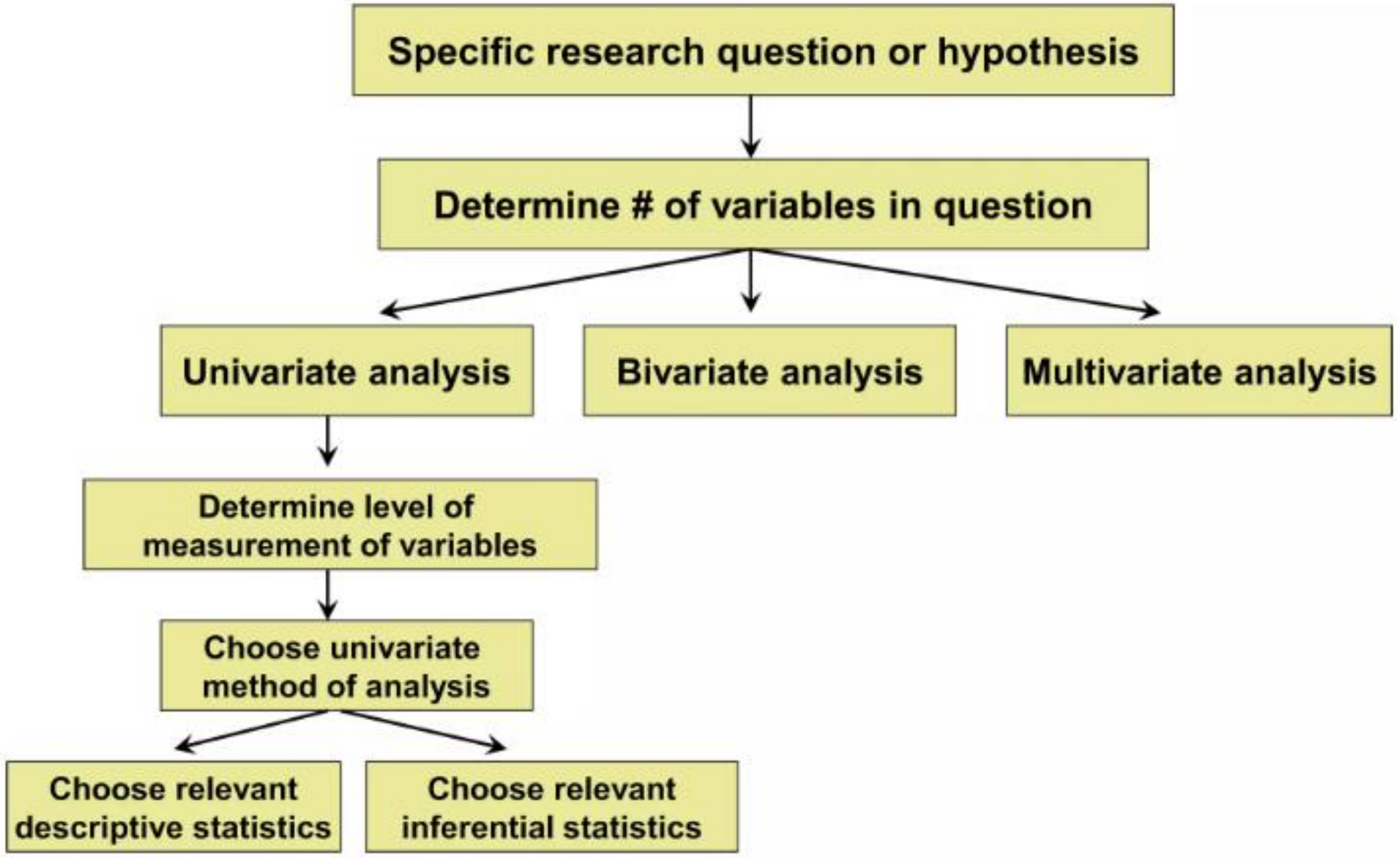
- Univariate Analysis

  -Mainly for Description
- Bivariate Analysis

  -Determining the empirical relationship between the two variables.
- Multivariate Analysis

  -Determining the empirical relationship among multiple variables.

Choosing the Statistical Technique


Specific research question or hypothesis
Determine # of variables in question
Univariate analysis
Bivariate analysis
Multivariate analysis
Determine level of measurement of variables
Choose univariate method of analysis
Choose relevant descriptive statistics
Choose relevant inferential statistics

Univariate analysis

- Analysis on a single type of dataset is called **univariate analysis.**
- Simplest form of analyzing data: Data has only one type of variable.
- The main purpose of univariate analysis is to take data, summarize that data, and find patterns among the values.
- Several techniques that describe the patterns found in univariate data
  - Central tendency: the mean, mode, and median.
  - Dispersion: the range, variance, maximum and minimum quartiles
  - Interquartile range and standard deviation

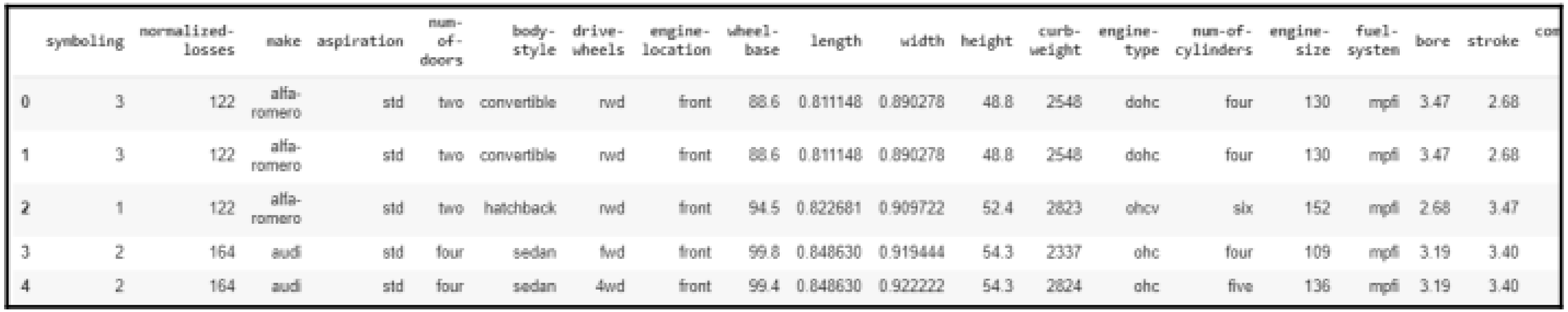
Example-1

```
#import libraries
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
# loading dataset as Pandas dataframe
df = pd.read_csv("data.csv")
df.head()
```

```
#calculate mean, median and mode of dat set height
mean = df["height"].mean()
median =df["height"].median()
mode = df["height"].mode()
print(mean , median, mode)
```

The output of those descriptive functions is as follows:

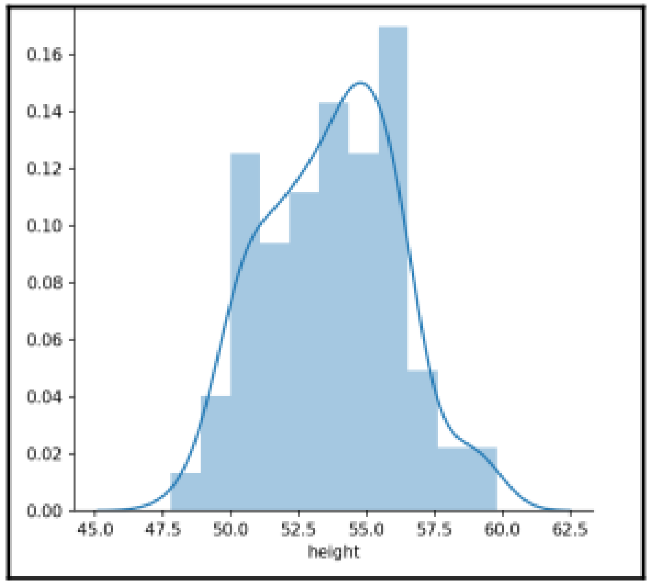
```
53.766666666666715 54.1 0 50.8
dtype: float64
```



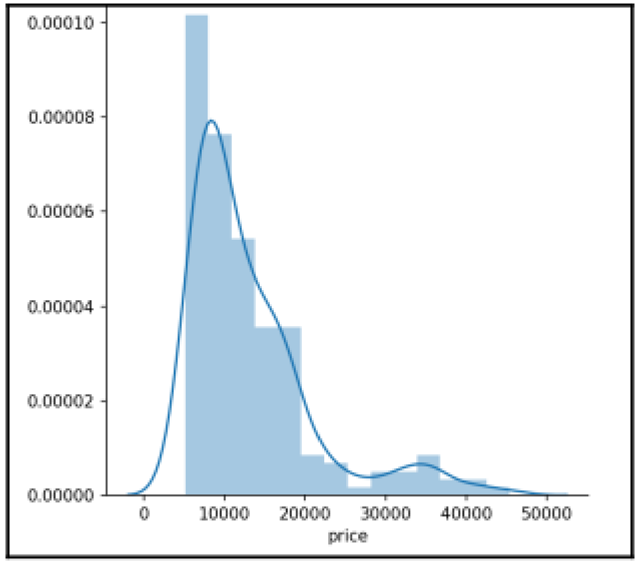
	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	engine-type	num-of-cylinders	engine-size	fuel-system	bore	stroke	com
0	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
1	3	122	alfa-romero	std	two	convertible	rwd	front	88.6	0.811148	0.890278	48.8	2548	dohc	four	130	mpfi	3.47	2.68	
2	1	122	alfa-romero	std	two	hatchback	rwd	front	94.5	0.822681	0.909722	52.4	2823	ohcv	six	152	mpfi	2.68	3.47	
3	2	164	audi	std	four	sedan	fwd	front	99.8	0.848630	0.919444	54.3	2337	ohc	four	109	mpfi	3.19	3.40	
4	2	164	audi	std	four	sedan	4wd	front	99.4	0.848630	0.922222	54.3	2824	ohc	five	136	mpfi	3.19	3.40	

Example-1

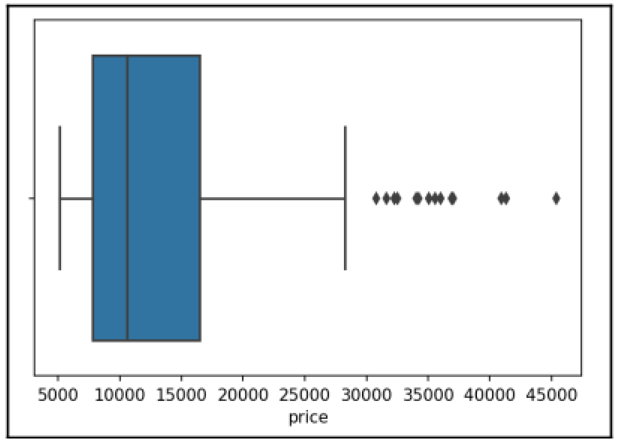
- #distribution plot
- sns.FacetGrid(df,size=5).map(sns.distplot,"height").add_legend()
- We can observe that the maximum height of maximum cars ranges from 53 to 57.



- #distribution plot
- sns.FacetGrid(df,size=5).map(sns.distplot,"price").add_legend()
- We can say that the price ranges from 5,000 to 45,000

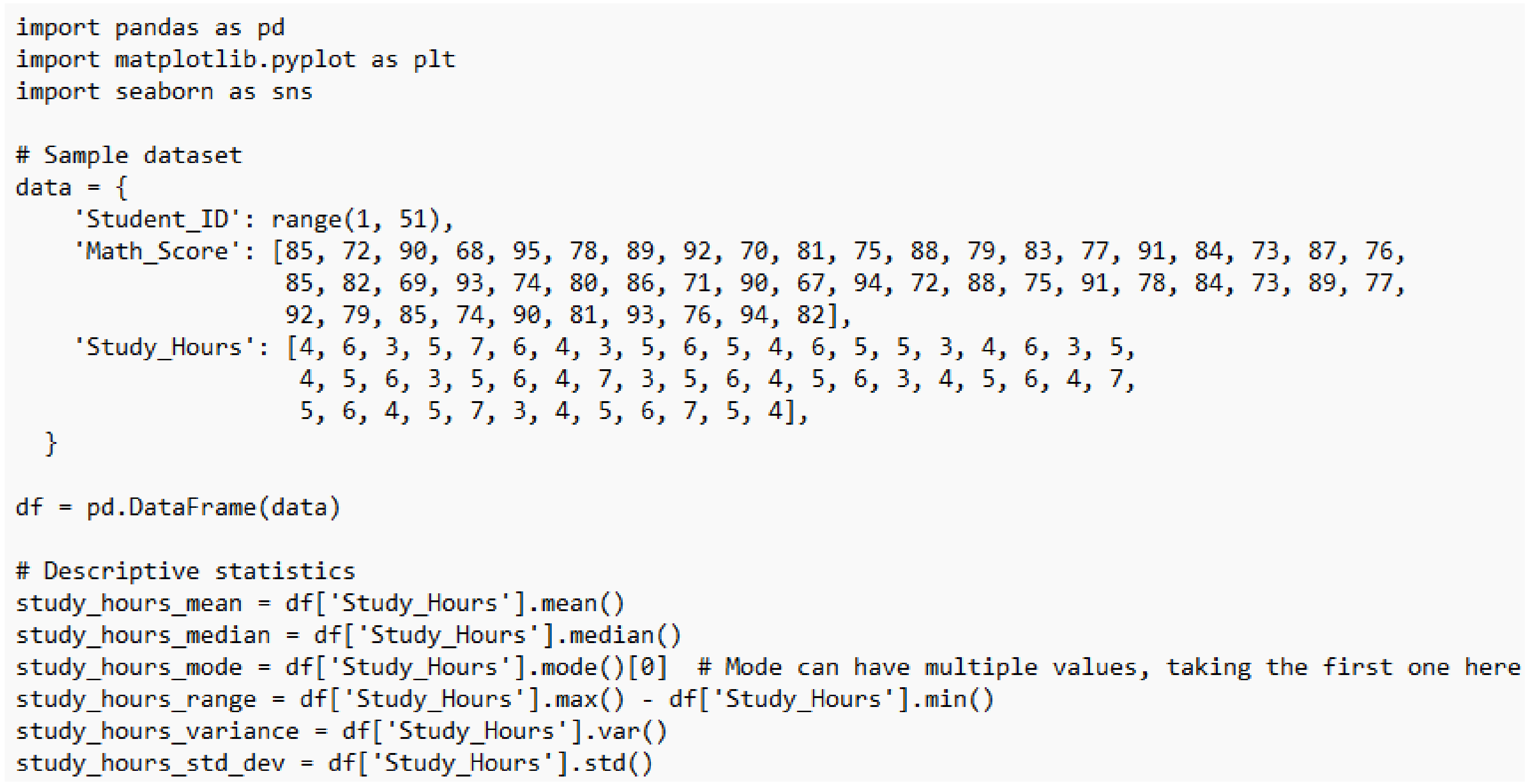


- #boxplot for price of cars
- sns.boxplot(x="price",data=df)
- plt.show()



Example-2

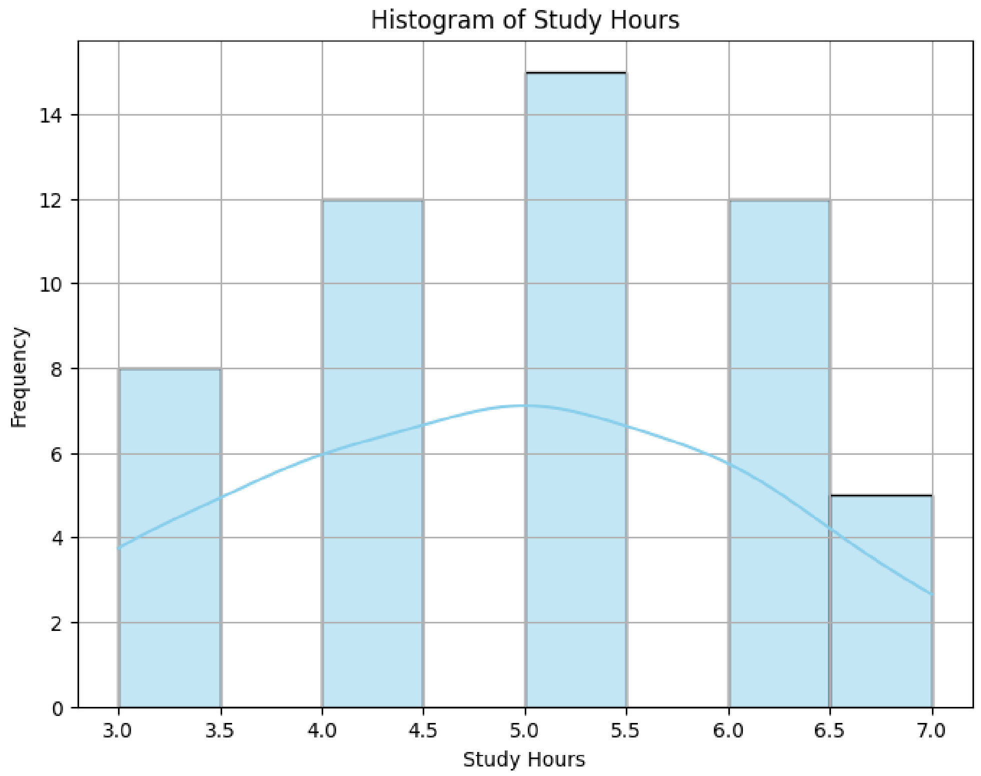
- Let's consider a univariate analysis using a dataset that contains multiple variables. For instance, suppose we have a dataset that includes information about students' math scores, study hours, and their ages.
- We'll perform univariate analysis on one variable—'Study Hours.’



```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Sample dataset
data = {
    'Student_ID': range(1, 51),
    'Math_Score': [85, 72, 90, 68, 95, 78, 89, 92, 70, 81, 75, 88, 79, 83, 77, 91, 84, 73, 87, 76,
                   85, 82, 69, 93, 74, 80, 86, 71, 90, 67, 94, 72, 88, 75, 91, 78, 84, 73, 89, 77,
                   92, 79, 85, 74, 90, 81, 93, 76, 94, 82],
    'Study_Hours': [4, 6, 3, 5, 7, 6, 4, 3, 5, 6, 5, 4, 6, 5, 5, 3, 4, 6, 3, 5,
                    4, 5, 6, 3, 5, 6, 4, 7, 3, 5, 6, 4, 5, 6, 3, 4, 5, 6, 4, 7,
                    5, 6, 4, 5, 7, 3, 4, 5, 6, 7, 5, 4],
  }

df = pd.DataFrame(data)

# Descriptive statistics
study_hours_mean = df['Study_Hours'].mean()
study_hours_median = df['Study_Hours'].median()
study_hours_mode = df['Study_Hours'].mode()[0]  # Mode can have multiple values, taking the first one here
study_hours_range = df['Study_Hours'].max() - df['Study_Hours'].min()
study_hours_variance = df['Study_Hours'].var()
study_hours_std_dev = df['Study_Hours'].std()
```

Example-2

```
print("Mean Study Hours:", study_hours_mean)
print("Median Study Hours:", study_hours_median)
print("Mode of Study Hours:", study_hours_mode)
print("Range of Study Hours:", study_hours_range)
print("Variance of Study Hours:", study_hours_variance)
print("Standard Deviation of Study Hours:", study_hours_std_dev)

# Visualization - Histogram
plt.figure(figsize=(8, 6))
sns.histplot(df['Study_Hours'], bins=8, kde=True, color='skyblue')
plt.title('Histogram of Study Hours')
plt.xlabel('Study Hours')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



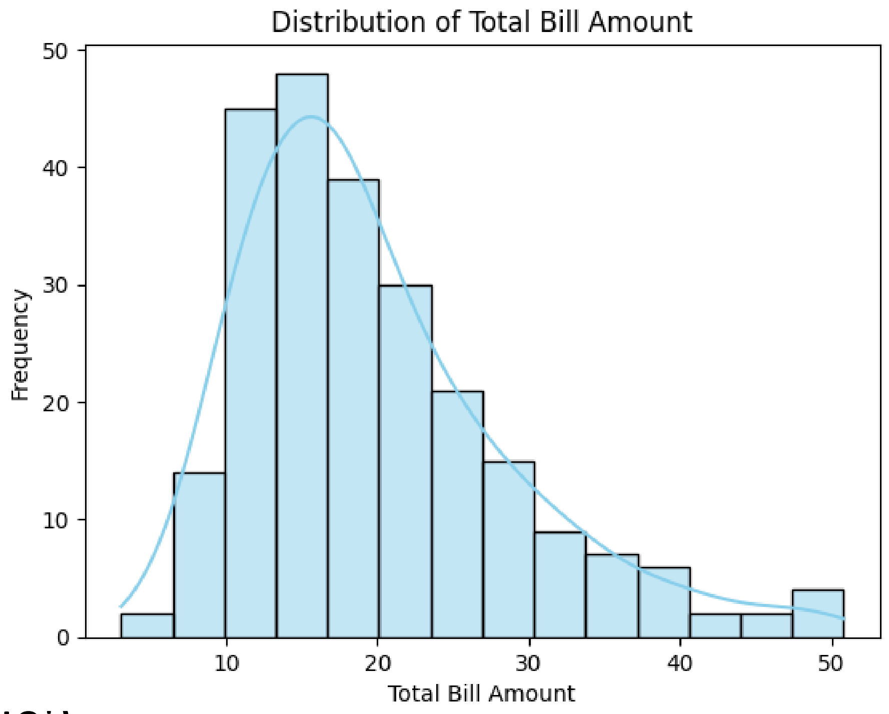
```
Mean Study Hours: 4.884615384615385
Median Study Hours: 5.0
Mode of Study Hours: 5
Range of Study Hours: 4
Variance of Study Hours: 1.4766214177978885
Standard Deviation of Study Hours: 1.2151631239458711
```

Example 3

- import seaborn as sns
- import matplotlib.pyplot as plt

- # Load a sample dataset from Seaborn
- tips_data = sns.load_dataset("tips")

- # Univariate analysis using a histogram
- sns.histplot(tips_data["total_bill"], kde=True, color='skyblue')
- plt.title('Distribution of Total Bill Amount')
- plt.xlabel('Total Bill Amount')
- plt.ylabel('Frequency')
- plt.show()

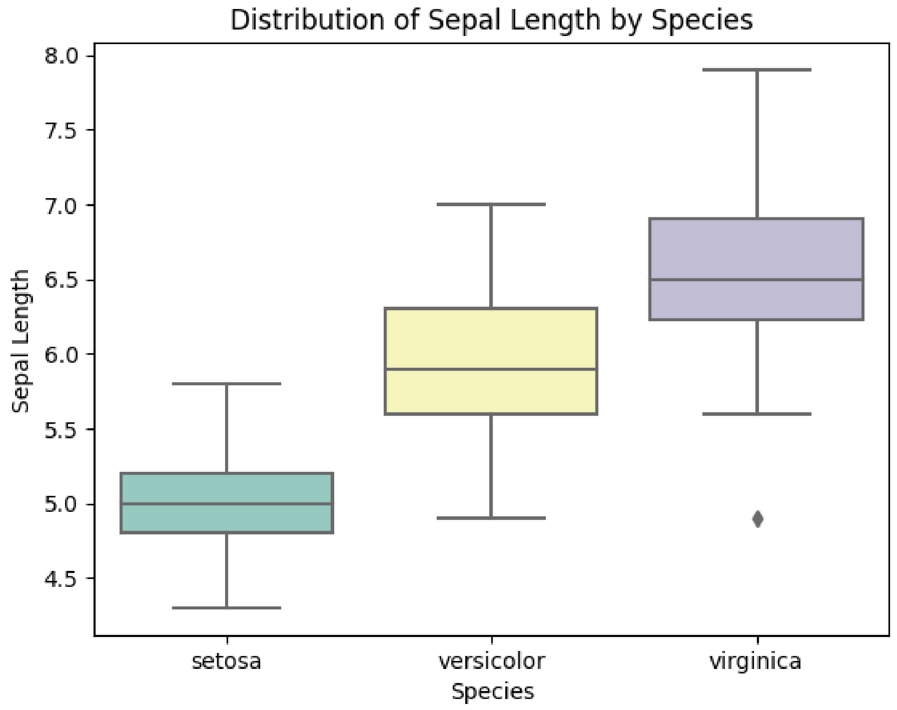


Example 4

- import seaborn as sns
- import matplotlib.pyplot as plt

- # Load a sample dataset from Seaborn
- iris_data = sns.load_dataset("iris")

- # Univariate analysis using a box plot
- sns.boxplot(x=iris_data["species"], y=iris_data["sepal_length"], palette="Set3")
- plt.title('Distribution of Sepal Length by Species')
- plt.xlabel('Species')
- plt.ylabel('Sepal Length')
- plt.show()



Exercise-1

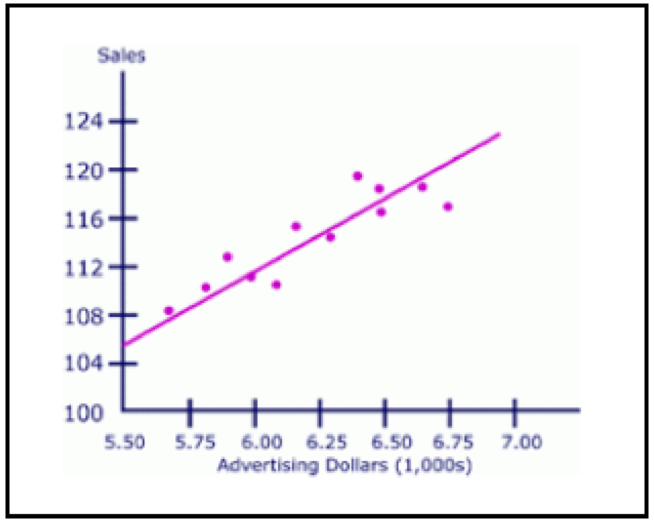
- Scenario: You work as a data analyst for an e-commerce company. The marketing team is interested in understanding the distribution of purchase amounts made by customers to tailor promotional strategies effectively.
- Question: Using univariate analysis, explore and describe the distribution of purchase amounts made by customers in the dataset.

Exercise-2

- You've been hired as a data analyst for a healthcare organization. The management wants insights into the distribution of patient ages to better understand the demographics they serve.

Bivariate analysis

- **Bivariate Analysis** involves the analysis of exactly two variables.
- It is used to find out whether there is a relationship between two different variables.
- Generally, bivariate analysis helps us to predict a value for one variable (that is, a dependent variable) if we are aware of the value of the independent variable.

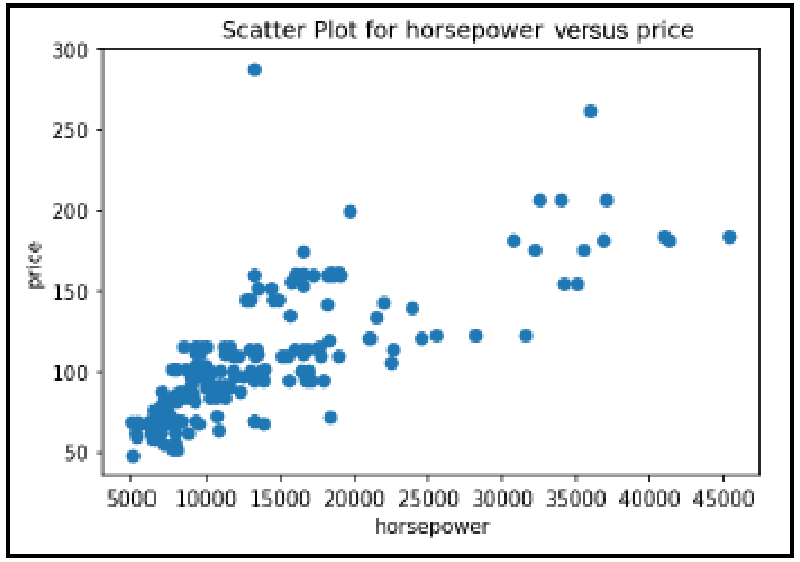


Importance

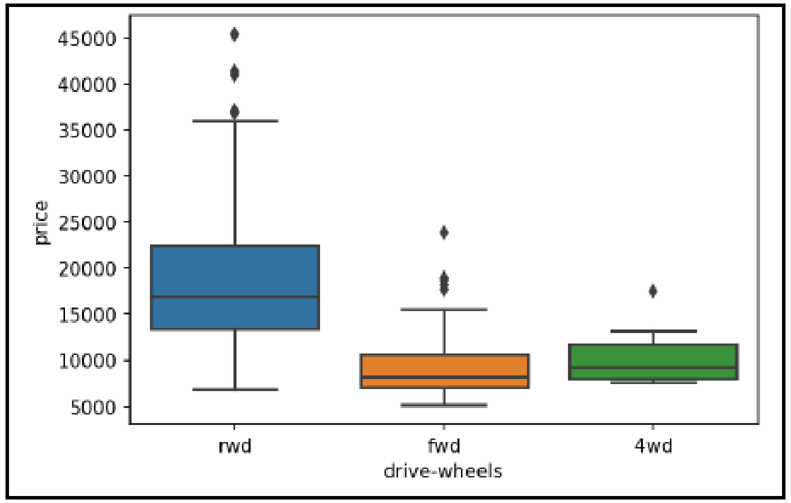
- Bivariate analysis helps identify trends and patterns.
- Bivariate analysis helps identify cause and effect relationships.
- It helps researchers make predictions.
- It helps inform decision-making: Business, public policy, and healthcare decision-making can benefit from bivariate analysis.

Example -5

- # plot the relationship between “horsepower” and ”price”
- plt.scatter(df["price"], df["horsepower"])
- plt.title("Scatter Plot for horsepower vs price")
- plt.xlabel("horsepower")
- plt.ylabel("price")

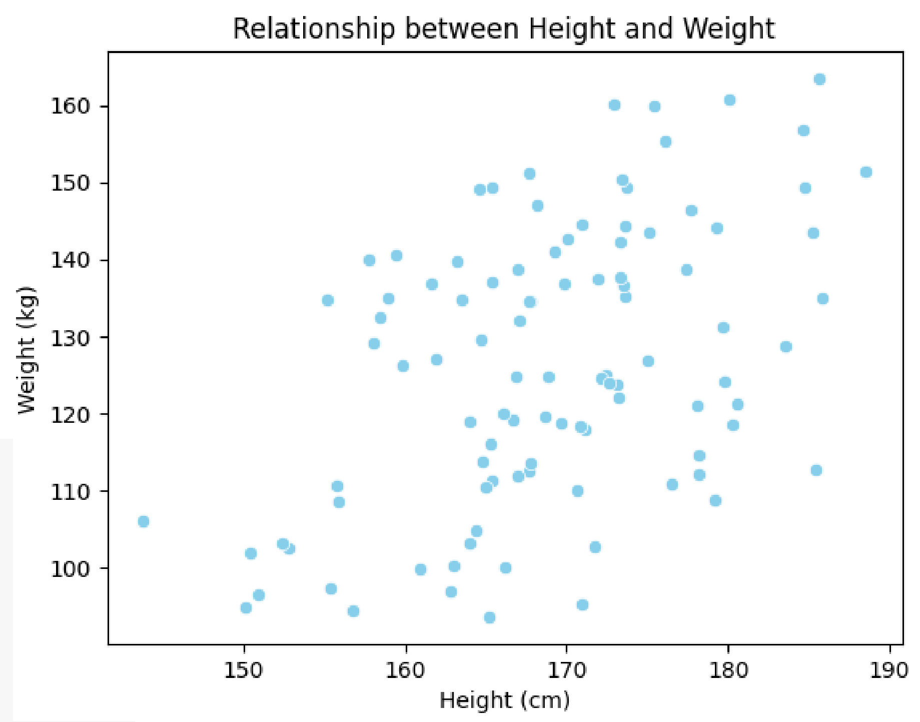


- #boxplot to visualize the distribution of "price" with types of
- "drive-wheels"
- sns.boxplot(x="drive-wheels", y="price",data=df)



Example - 6

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

# Generating random data for 'height' and 'weight'
np.random.seed(42)
height = np.random.normal(170, 10, 100)  # Mean: 170, Standard Deviation: 10
weight = height * np.random.uniform(0.6, 0.9, 100) + np.random.normal(0, 5, 100)

# Creating a DataFrame
data = pd.DataFrame({'Height': height, 'Weight': weight})

# Bivariate analysis using a scatter plot
sns.scatterplot(x='Height', y='Weight', data=data, color='skyblue')
plt.title('Relationship between Height and Weight')
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.show()
```



Example-7

- import pandas as pd
- import numpy as np
- from scipy.stats import pearsonr

- # Generating hypothetical data
- np.random.seed(42)
- age = np.random.randint(20, 80, 100) # Simulating ages between 20 and 80 years
- blood_pressure = age * 0.5 + np.random.normal(0, 10, 100) # Simulating blood pressure

- # Creating a DataFrame
- health_data = pd.DataFrame({'Age': age, 'Blood Pressure': blood_pressure})

- # Calculating Pearson's correlation coefficient
- correlation, p_value = pearsonr(health_data['Age'], health_data['Blood Pressure'])
- print(f"Pearson's correlation coefficient: {correlation:.2f}")
- print(f"P-value: {p_value:.4f}")

```
Pearson's correlation coefficient: 0.71
P-value: 0.0000
```

Exercise-3

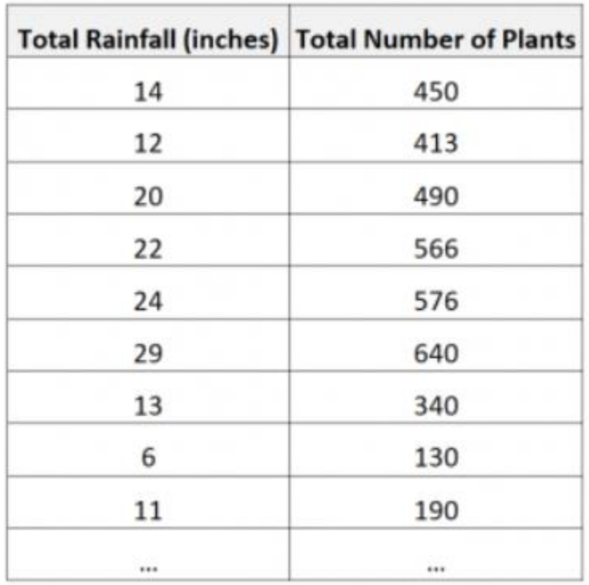
- Scenario: You're working as a data analyst for a retail company that wants to understand the relationship between the amount spent by customers and the time they spend on the company's website. This information will aid in optimizing the website experience and marketing strategies.
- Question: Using a bivariate analysis technique, explore the relationship between the time spent on the website (in minutes) and the amount spent by customers (in dollars) to derive insights for the retail company.

Exercise-4

- Suppose you're analyzing the relationship between advertising expenditure and product sales for a range of products in a retail store. Describe how a bivariate analysis can help uncover insights into the effectiveness of advertising campaigns and their impact on product sales. Provide an approach and potential findings that could emerge from such an analysis.

Exercise-5

- Suppose you're analyzing the relationship between data on total rainfall and total number of plants in different regions given below. Describe how a bivariate analysis can help uncover insights



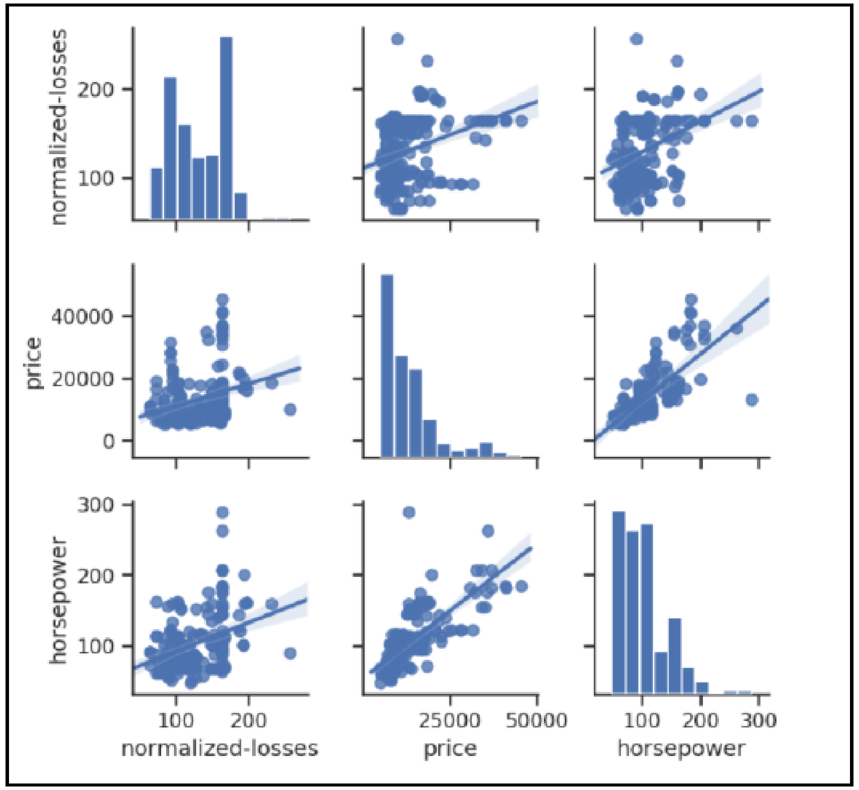
Total Rainfall (inches)	Total Number of Plants
14	450
12	413
20	490
22	566
24	576
29	640
13	340
6	130
11	190
...	...

Multivariate Analysis

- Multivariate analysis is the analysis of three or more variables.
- This allows us to look at correlations and attempt to make predictions for future behavior more accurately.
- It's a set of techniques that helps us make sense of complex information by examining multiple variables at once.

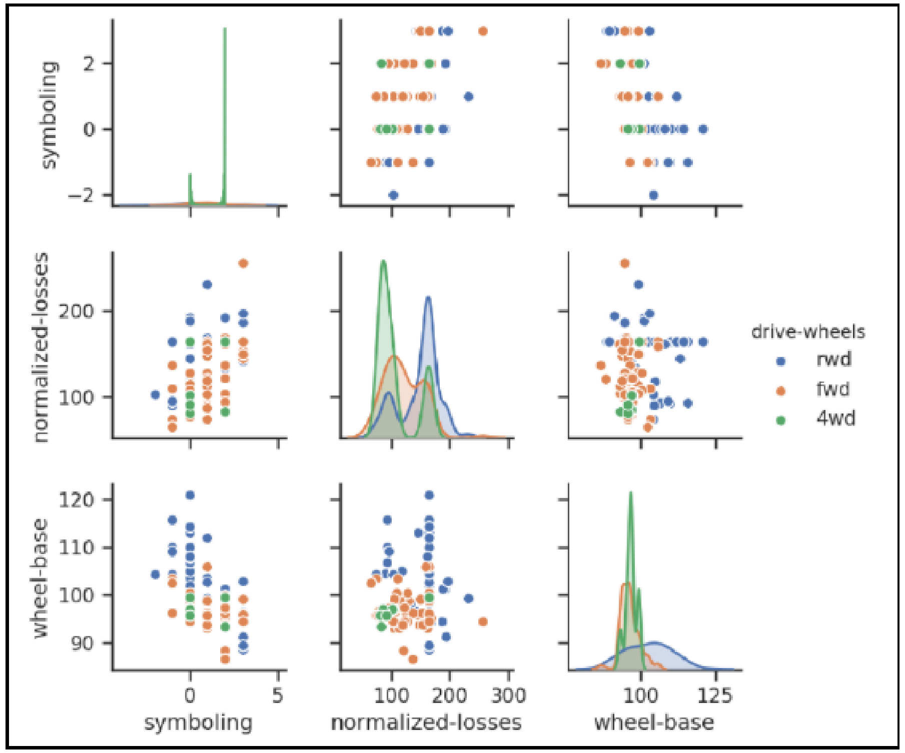
Example-8

- One common way of plotting multivariate data is to make a matrix scatter plot, known as a pair plot.
- A matrix plot or pair plot shows each pair of variables plotted against each other.
- # pair plot with plot type regression
- sns.pairplot(df,vars = ['normalized-losses', 'price','horsepower'],
- kind="reg")
- plt.show()



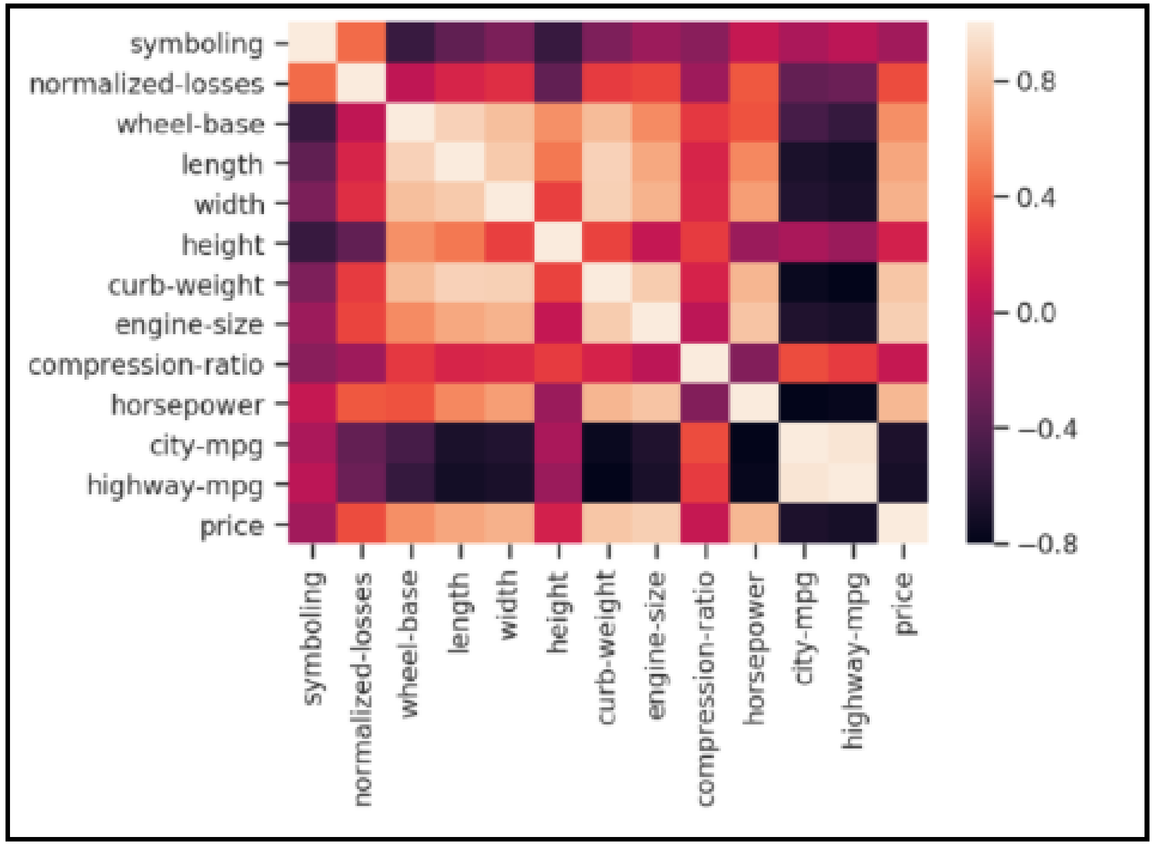
Example-9

- #pair plot (matrix scatterplot) of few columns
- sns.set(style="ticks", color_codes=True)
- sns.pairplot(df,vars = ['symboling', 'normalized-losses','wheelbase'],
- hue="drive-wheels")
- plt.show()



Example-10

- sns.heatmap(correlation,xticklabels=correlation.columns, yticklabels=correlation.columns)



Multivariate data analysis technique

- There are many different techniques for multivariate analysis, and they can be divided into two categories:
- Dependence techniques : Dependence methods are used when one or some of the variables are dependent on others.
- Interdependence techniques :Interdependence methods are used to understand the structural makeup and underlying patterns within a dataset.

Multivariate Analysis Techniques

- Multiple linear regression
- Multiple logistic regression
- Multivariate analysis of variance (MANOVA)
- Factor analysis
- Cluster analysis

Multiple Linear Regression

- Multiple linear regression is a dependence method.
- It looks at the relationship between one dependent variable and two or more independent variables.
- A multiple regression model will tell you the extent to which each independent variable has a linear relationship with the dependent variable.
- Example: crop growth is dependent variable and independent variables could be rainfall, temperature, amount of sunlight, and amount of fertilizer added to the soil.

Multiple Logistic Regression

- Logistic regression analysis is used to calculate (and predict) the probability of a binary event occurring.
- Based on a set of independent variables, logistic regression can predict how likely it is that a certain scenario will arise.
- It is also used for classification.
- Example:

1. filters used to classify email as “spam” or “not spam.”

2. Insurance : Claim or Not claim bases on independent variable age, health condition, occupation etc.

Multivariate Analysis of Variance (MANOVA)

- Multivariate analysis of variance (MANOVA) is used to measure the effect of multiple independent variables on two or more dependent variables.
- The independent variables are categorical.
- Example:
- Categorical independent variables could be:
- Engine type, categorized as E1, E2, or E3
- Material used for the rocket exterior, categorized as M1, M2, or M3
- Type of fuel used to power the rocket, categorized as F1, F2, or F3
- Your metric dependent variables are speed in kilometers per hour. This should help you to find the optimal design solution for your rocket.

Factor Analysis

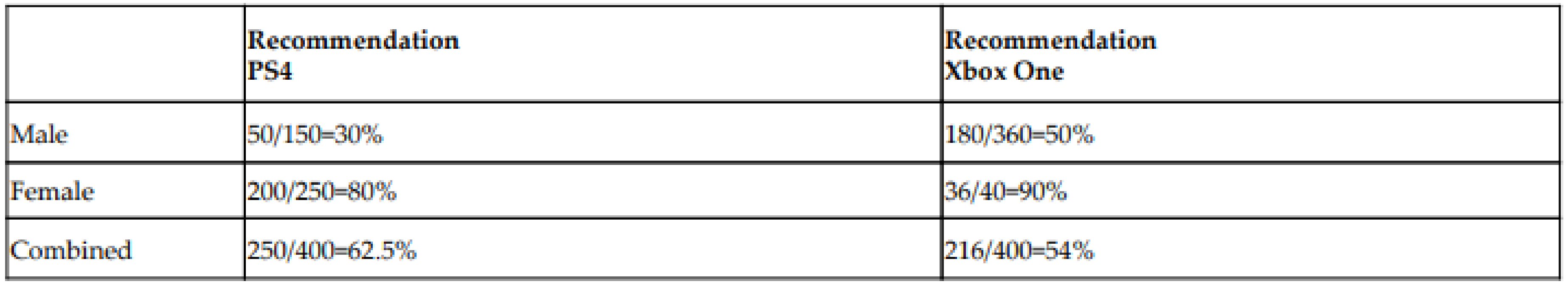
- Factor analysis is an interdependence technique.
- It seeks to reduce the number of variables in a dataset.
- If you have too many variables, it can be difficult to find patterns and lead to overfitting.
- Factor analysis works by detecting sets of variables which correlate highly with each other.
- These variables may then be condensed into a single variable.
- Example: You might also have data on how happy they were with customer service, how much they like a certain product, and how likely they are to recommend the product to a friend. Each of these variables could be grouped into the single factor “customer satisfaction”

Cluster Analysis

- Another interdependence technique, cluster analysis is used to group similar items within a dataset into clusters.
- This is measured in terms of intracluster and intercluster distance.
- Cluster analysis helps you to understand how data in your sample is distributed, and to find patterns.
- Example :As a **healthcare analyst**, you might use cluster analysis to explore whether certain lifestyle factors or geographical locations are associated with higher or lower cases of certain illnesses.

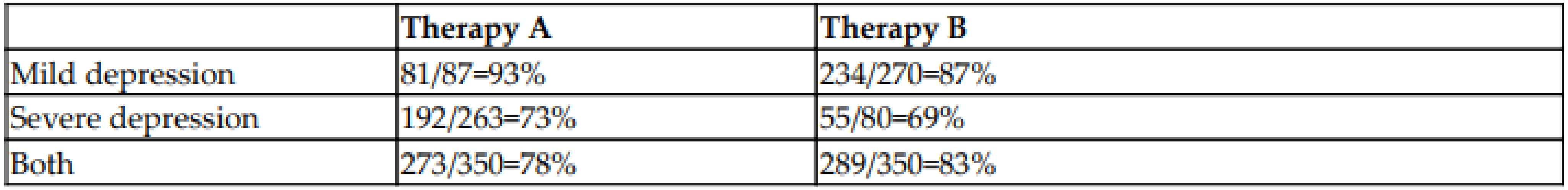
Outlining Simpson's paradox

- The decisions differ when we segregate the data into groups and apply statistical measures, or when we aggregate it together and then apply statistical measures.
- This kind of anomalous behavior in the results of the same dataset is generally called Simpson's paradox



	Recommendation PS4	**Recommendation Xbox One**
Male	50/150=30%	180/360=50%
Female	200/250=80%	36/40=90%
Combined	250/400=62.5%	216/400=54%

Simpson's paradox



	Therapy A	Therapy B
Mild depression	81/87=93%	234/270=87%
Severe depression	192/263=73%	55/80=69%
Both	273/350=78%	289/350=83%

- Confounding variables are something we don't see in the data table but they can be determined by background analysis of the data.
- Data is never purely objective and neither is the final plot. Therefore, we must consider whether we are getting the whole story when dealing with a set of data.

Correlation does not imply causation

- **Correlation** reveals how strongly a pair of variables are related to each other and change together.
- **Causation** explains that any change in the value of one variable will cause a difference in the amount of another variable.
- In this case, one variable makes the other variable happen. This phenomenon is known as **cause and effect**.
- We should not form our conclusions too quickly based on correlation. It is essential to invest some time in finding the underlying factors of the data in order to understand any critical, hidden factors.

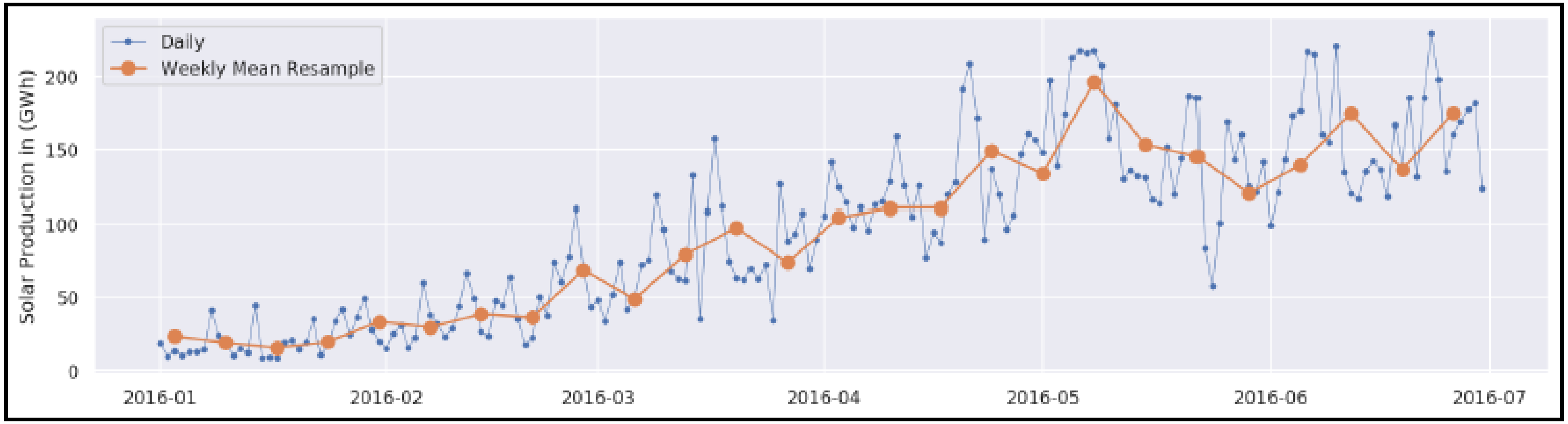
Time Series Analysis

- An ordered sequence of timestamp values at equally spaced intervals is referred to as a *time series*.
- Time series data is in the form of a sequence of quantitative observations about a system or process and is made at successive points in time.
- Analysis of a time series is used in many applications such as sales forecasting, utility studies, budget analysis, economic forecasting, inventory studies, and so on.
- Two important key phrases here—
- **a collection of observations** and
- **sequentially in time**

Understanding Time Series Dataset

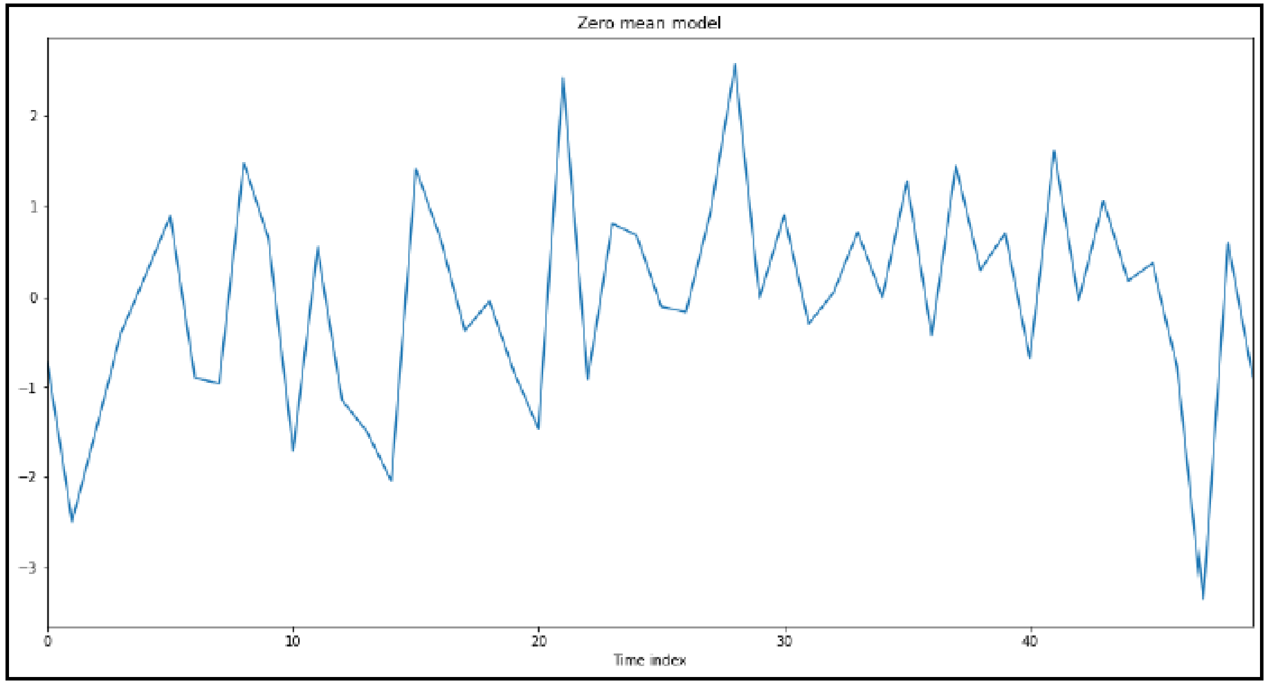
- A time series is a collection of observations made sequentially in time.
- Two important key phrases : **a collection of observations** and

  Sequentially in time.



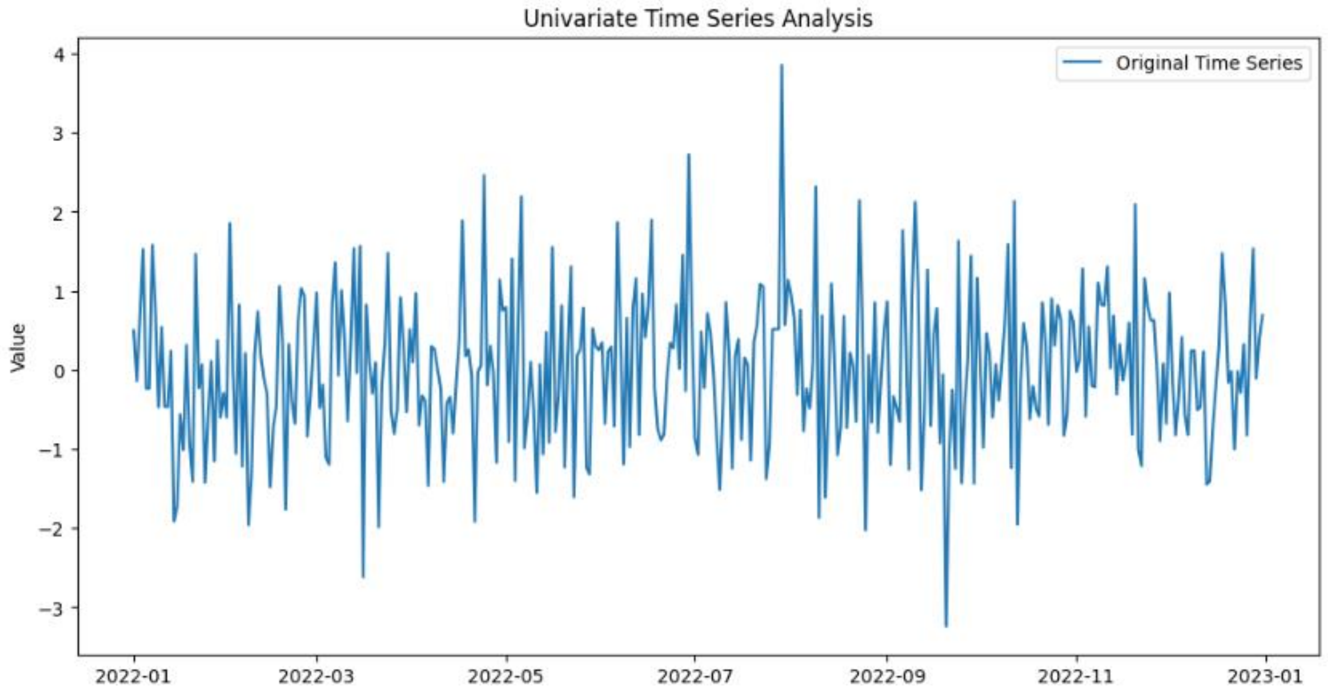
Example-1

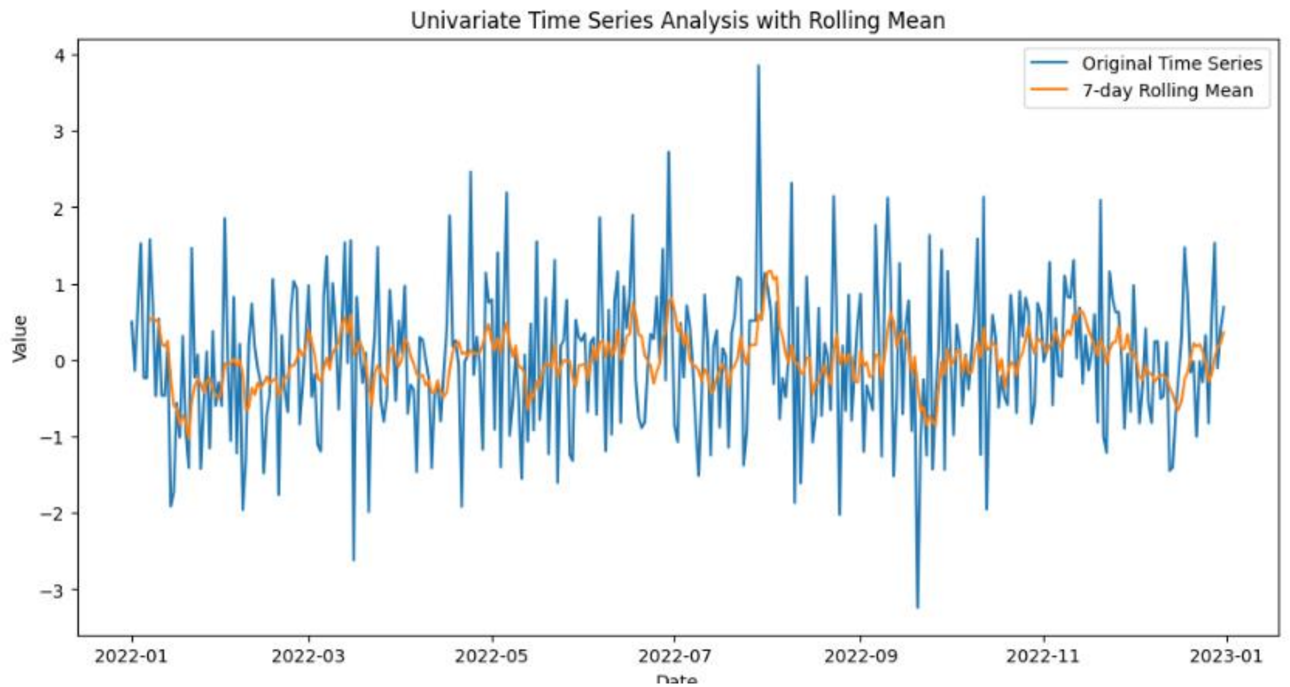
- plt.figure(figsize=(16, 8))
- g = sns.lineplot(data=zero_mean_series)
- g.set_title('Zero mean model')
- g.set_xlabel('Time index')
- plt.show()



Example-2

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Generate a simple time series dataset
np.random.seed(42)
date_rng = pd.date_range(start='2022-01-01', end='2022-12-31', freq='D')
data = np.random.randn(len(date_rng))  # Random values for simplicity
df = pd.DataFrame(data, columns=['Value'], index=date_rng)

# Plot the original time series
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Value'], label='Original Time Series')
plt.title('Univariate Time Series Analysis')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()

# Basic statistics of the time series
print("Basic Statistics:")
print(df.describe())

# Calculate and plot the rolling mean (e.g., 7-day rolling mean)
window_size = 7
df['RollingMean'] = df['Value'].rolling(window=window_size).mean()

plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Value'], label='Original Time Series')
plt.plot(df.index, df['RollingMean'], label=f'{window_size}-day Rolling Mean')
plt.title('Univariate Time Series Analysis with Rolling Mean')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```



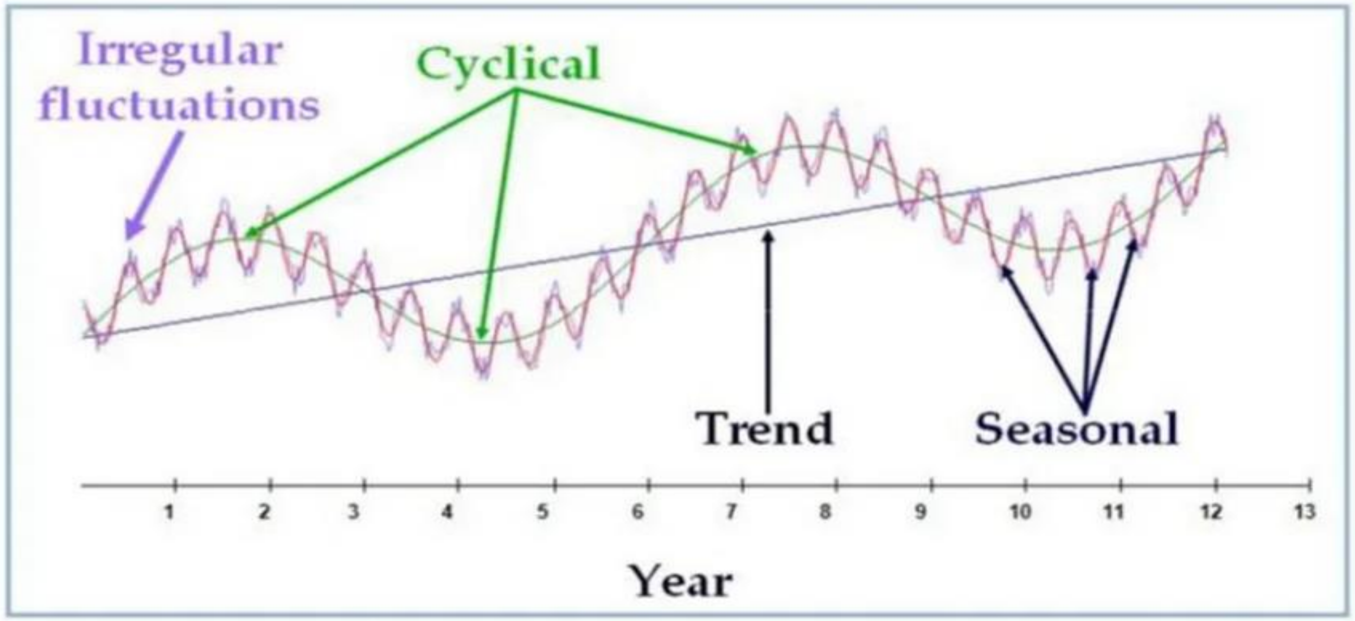


Univariate Time series

- When we capture a sequence of observations for the same variable over a particular duration of time.
- In general, in a univariate time series, the observations are taken over regular time periods.
- Example: change in temperature over time throughout a day.

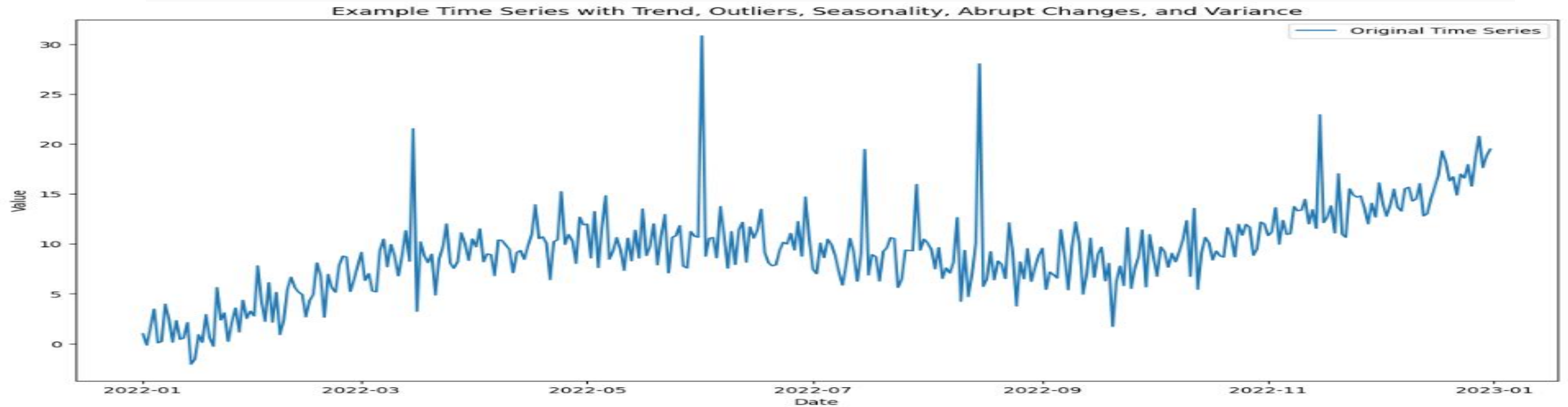
Characteristics of Time series data

- When looking at time series data, it is essential to see if there is any **trend**.
- Time series data may contain a notable amount of **outliers**.
- Some data in time series tends to repeat over a certain interval in some patterns : **seasonality**.
- Sometimes, there is an uneven change in time series data : **abrupt changes**.
- Some series tend to follow **constant variance** over time.



Example-3

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Generate a time series with trend, outliers, seasonality, abrupt changes, and constant variance
np.random.seed(42)
date_rng = pd.date_range(start='2022-01-01', end='2022-12-31', freq='D')
trend = 0.05 * np.arange(len(date_rng))
seasonality = 5 * np.sin(2 * np.pi * np.arange(len(date_rng)) / 365)
abrupt_changes = np.where(date_rng.isin(['2022-03-15', '2022-07-15', '2022-11-15']), 10, 0)
outliers = np.where(date_rng.isin(['2022-06-01', '2022-08-15']), 20, 0)
variance = 2 * np.random.randn(len(date_rng))

time_series = trend + seasonality + abrupt_changes + outliers + variance

# Create a DataFrame
df = pd.DataFrame(time_series, columns=['Value'], index=date_rng)

# Plot the time series
plt.figure(figsize=(14, 8))
plt.plot(df.index, df['Value'], label='Original Time Series')
plt.title('Example Time Series with Trend, Outliers, Seasonality, Abrupt Changes, and Variance')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```



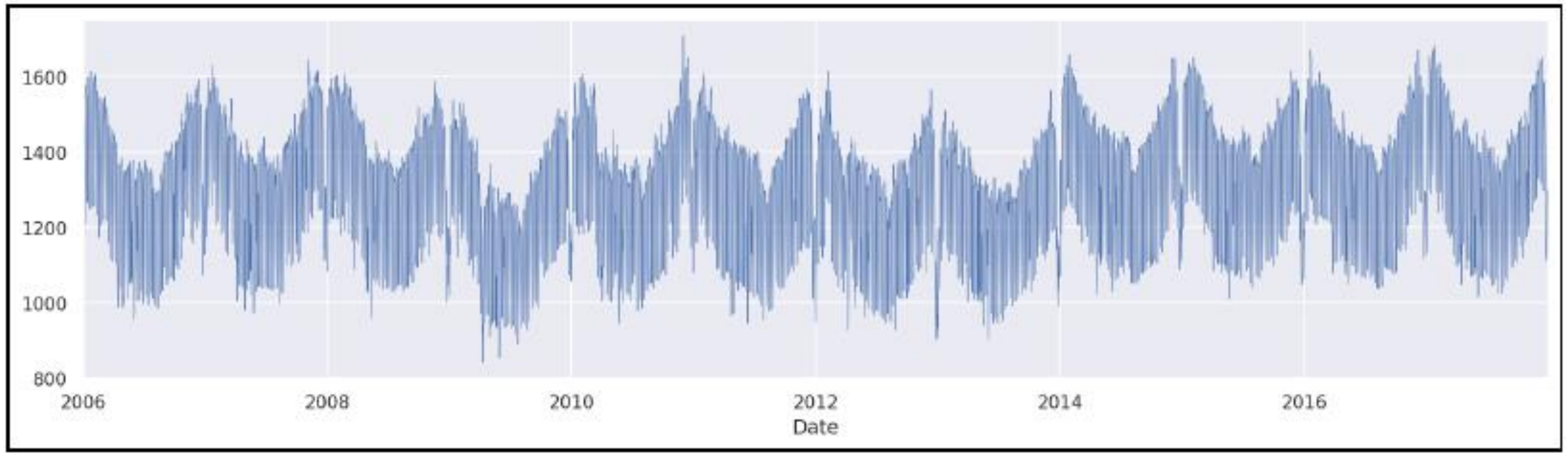
Exercise-1

- Consider a dataset containing monthly sales data for a retail store over a period of two years. The dataset has two columns: "Date" and "Sales." Your task is to perform time series analysis on this data.

1. Load the dataset and convert the "Date" column to the datetime format.
2. Set the "Date" column as the index of the DataFrame.
3. Plot the time series to visualize the sales pattern over the two-year period.
4. Calculate and plot the 7-day rolling mean of sales to identify trends.
5. Determine whether there are any seasonal patterns in the data.

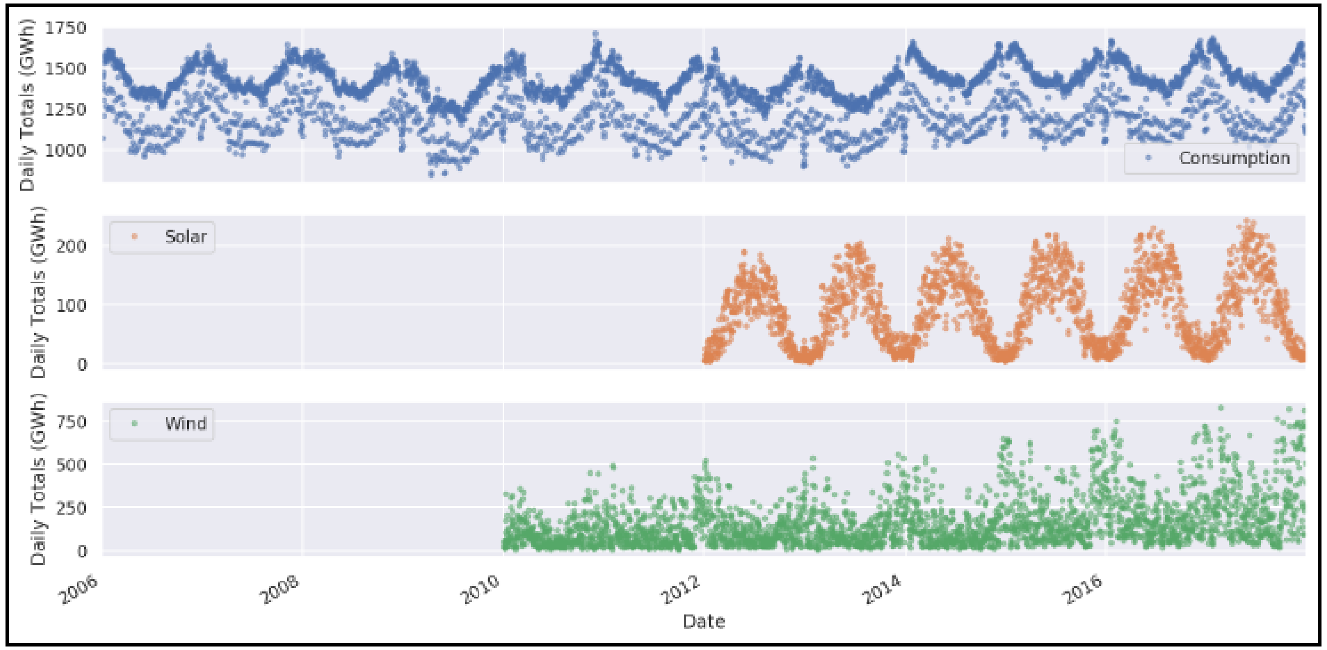
Visualizing Time series

- import matplotlib.pyplot as plt
- import seaborn as sns
- sns.set(rc={'figure.figsize':(11, 4)})
- plt.rcParams['figure.figsize'] = (8,5)
- plt.rcParams['figure.dpi'] = 150
- df_power['Consumption'].plot(linewidth=0.5)



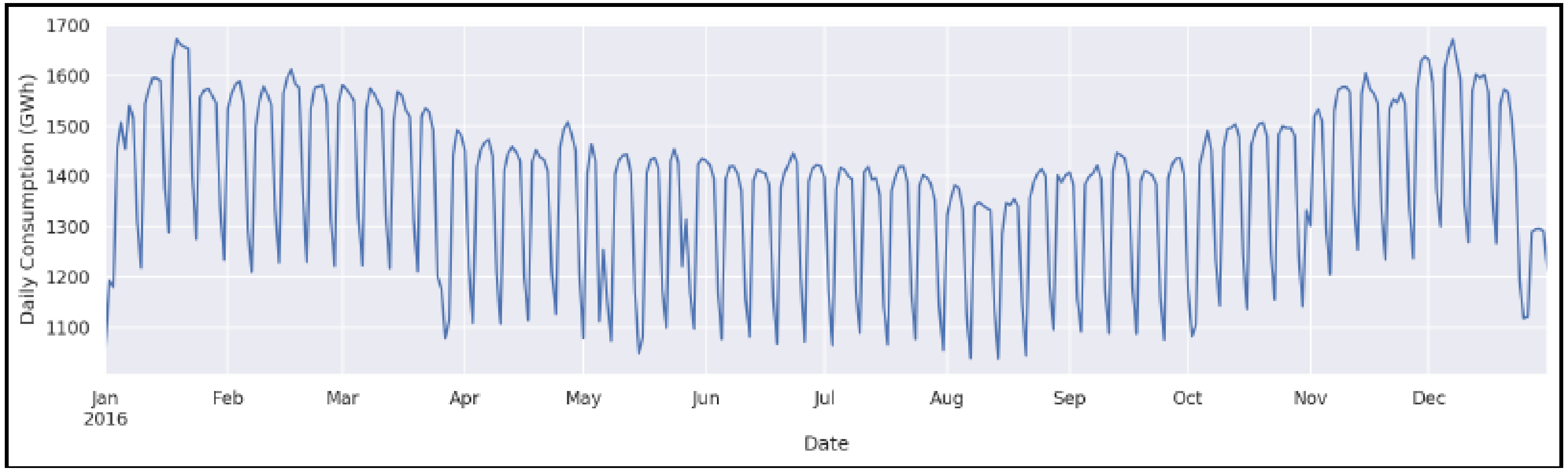
Visualizing Time series

- cols_to_plot = ['Consumption', 'Solar', 'Wind']
- axes = df_power[cols_to_plot].plot(marker='.', alpha=0.5,
- linestyle='None',figsize=(14, 6), subplots=True)
- for ax in axes:
- ax.set_ylabel('Daily Totals (GWh)')



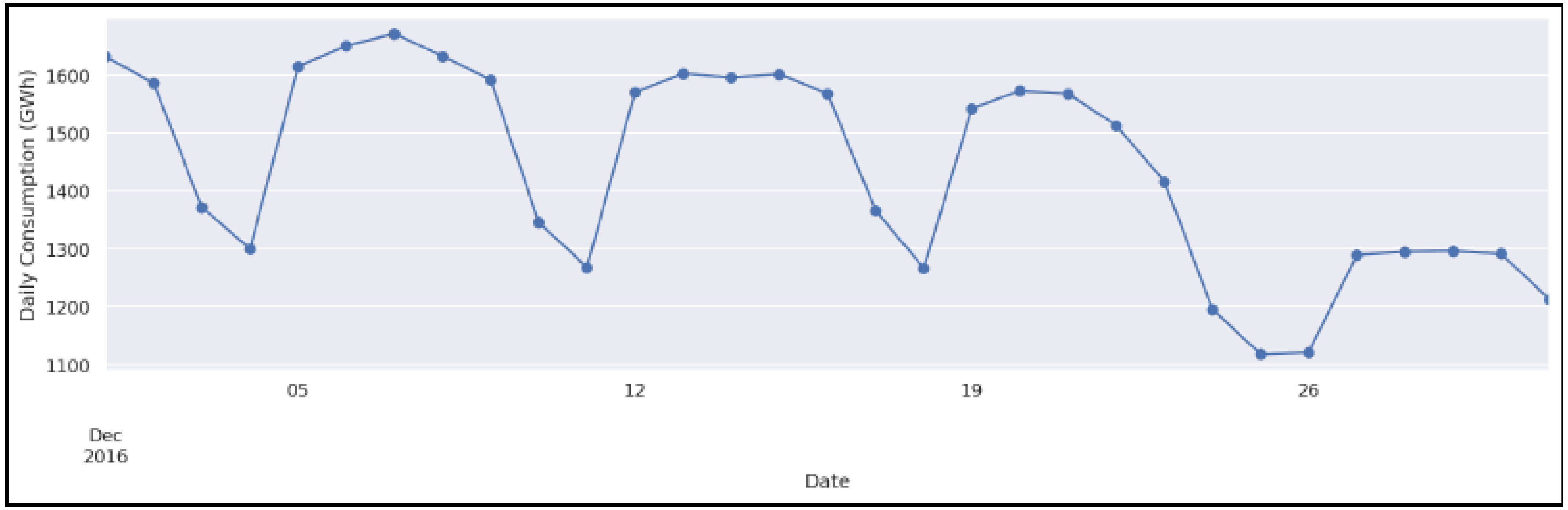
Visualizing Time series

- ax = df_power.loc['2016', 'Consumption'].plot()
- ax.set_ylabel('Daily Consumption (GWh)');



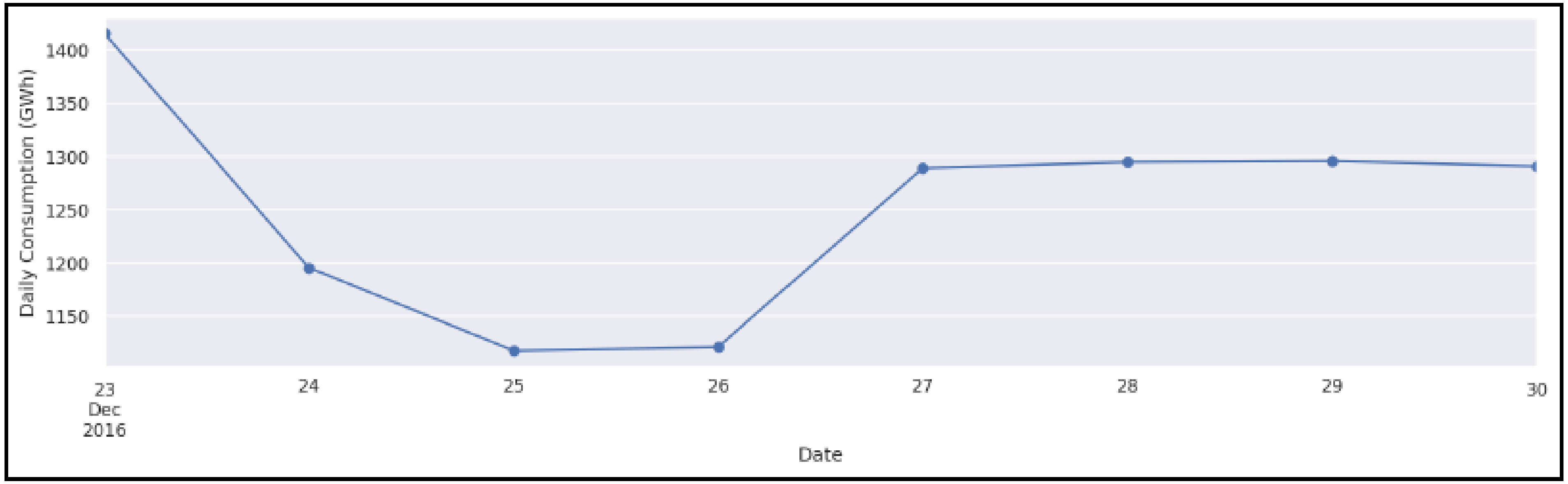
Visualizing Time series

- ax = df_power.loc['2016-12', 'Consumption'].plot(marker='o', linestyle='-')
- ax.set_ylabel('Daily Consumption (GWh)');



Visualizing Time series

- ax = df_power.loc['2016-12-23':'2016-12-30',
- 'Consumption'].plot(marker='o', linestyle='-')
- ax.set_ylabel('Daily Consumption (GWh)');

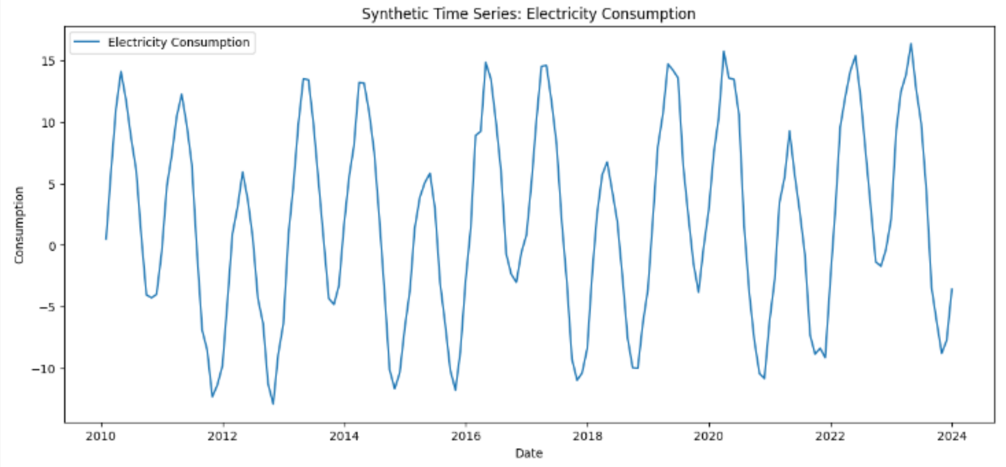


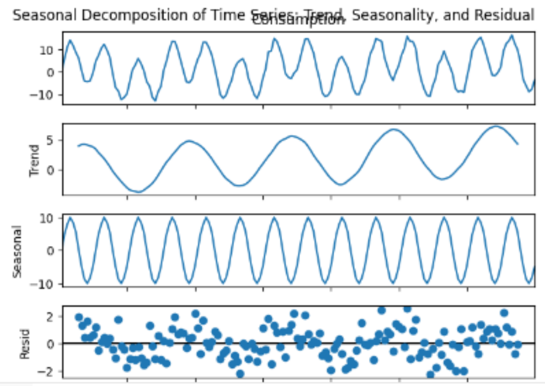
Example-3

- You are provided with a time series dataset containing monthly electricity consumption data for a residential area over several years. Your task is to perform an analysis to identify and interpret the trend, seasonal pattern, and potential cyclical behavior in the data.

- Load the dataset and convert the "Date" column to the datetime format.
- Set the "Date" column as the index of the DataFrame.
- Plot the time series to visualize the overall pattern.
- Use the seasonal_decompose function from the statsmodels library to decompose the time series into trend, seasonal, and residual components.
- Interpret and explain the identified trend, seasonal pattern, and any potential cyclical behavior based on the decomposition results.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

# Generate a synthetic time series dataset with trend, seasonality, and cyclical pattern
np.random.seed(42)
date_rng = pd.date_range(start='2010-01-01', end='2023-12-31', freq='M')
trend = 0.02 * np.arange(len(date_rng))
seasonality = 10 * np.sin(2 * np.pi * np.arange(len(date_rng)) / 12)
cyclical = 5 * np.sin(2 * np.pi * np.arange(len(date_rng)) / 36)
noise = np.random.randn(len(date_rng))
electricity_consumption = trend + seasonality + cyclical + noise

# Create a DataFrame
df = pd.DataFrame(electricity_consumption, columns=['Consumption'], index=date_rng)

# Plot the time series
plt.figure(figsize=(14, 6))
plt.plot(df.index, df['Consumption'], label='Electricity Consumption')
plt.title('Synthetic Time Series: Electricity Consumption')
plt.xlabel('Date')
plt.ylabel('Consumption')
plt.legend()
plt.show()

# Decompose the time series into trend, seasonality, and residual
result = seasonal_decompose(df['Consumption'], model='additive', period=12)

# Plot the decomposition
result.plot()
plt.suptitle('Seasonal Decomposition of Time Series: Trend, Seasonality, and Residual')
plt.show()

# Interpretation
print("\nInterpretation:")
print("1. Trend:")
```

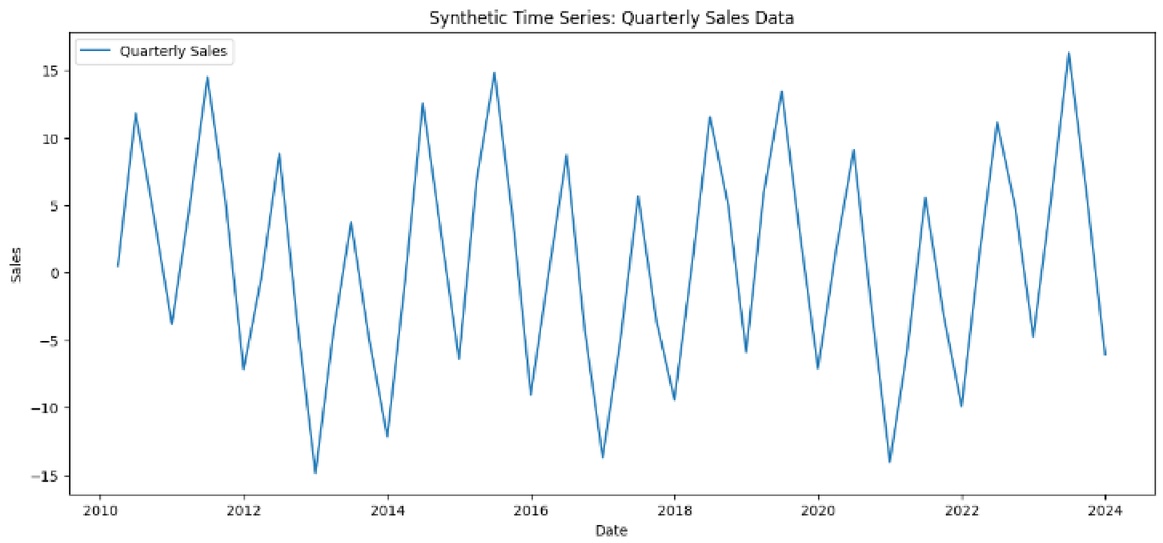


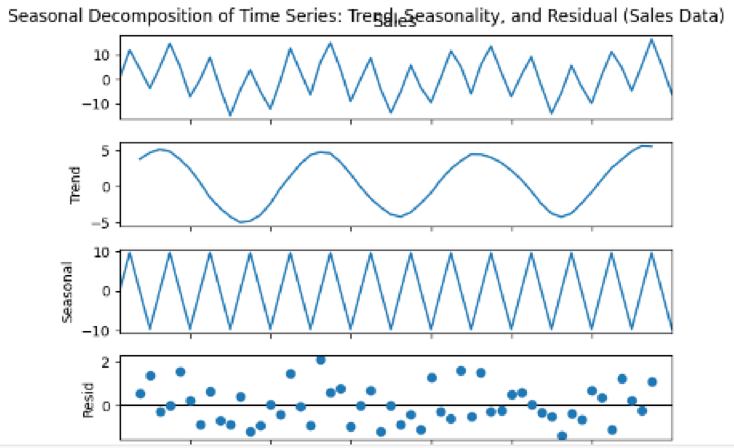


Exercise-2

- You have been given a dataset representing quarterly sales data for a retail business over the past decade. Your objective is to analyze the time series data and identify the trend, seasonal pattern, and any potential cyclical behavior.

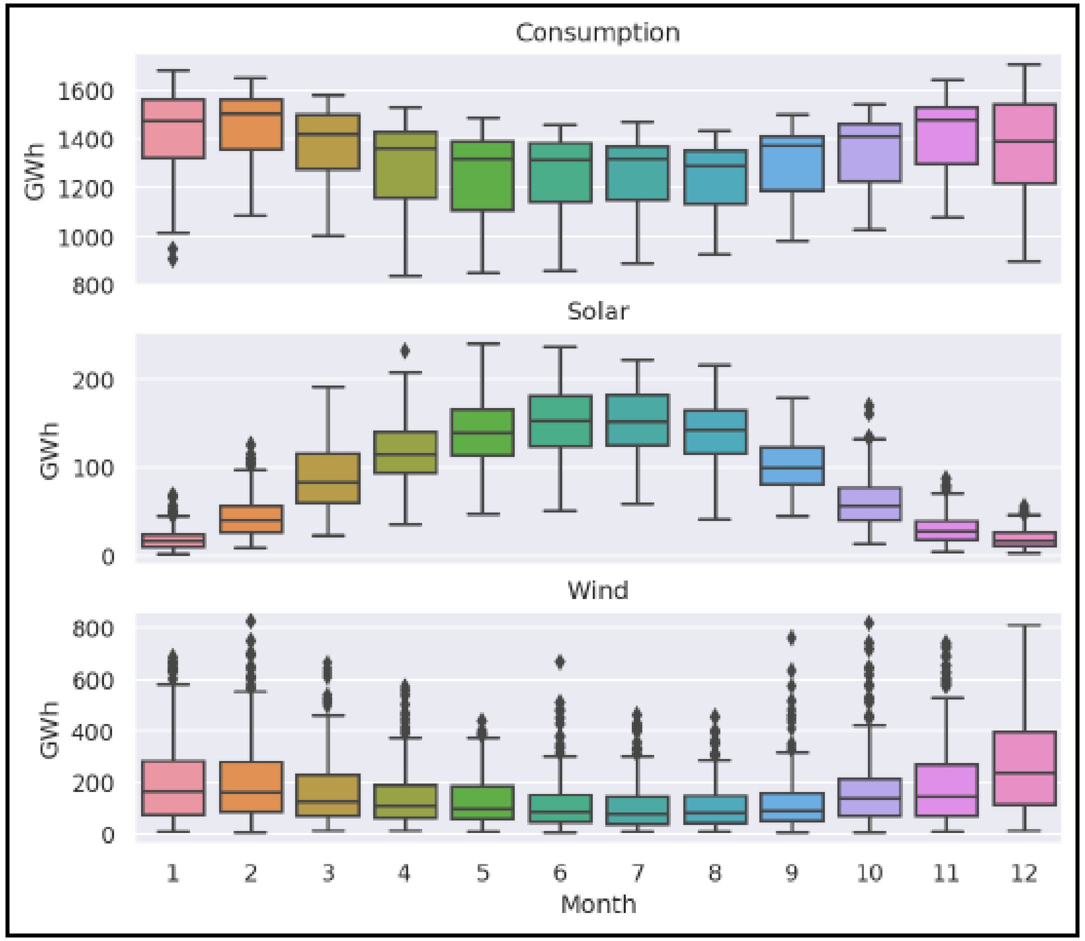
- Load the dataset and convert the "Date" column to the datetime format.
- Set the "Date" column as the index of the DataFrame.
- Plot the time series to visualize the overall pattern in the sales data.
- Apply a suitable method (e.g., moving averages or seasonal decomposition) to identify and separate the trend, seasonal, and residual components.
- Interpret the identified trend, seasonal pattern, and any observed cyclical behavior.





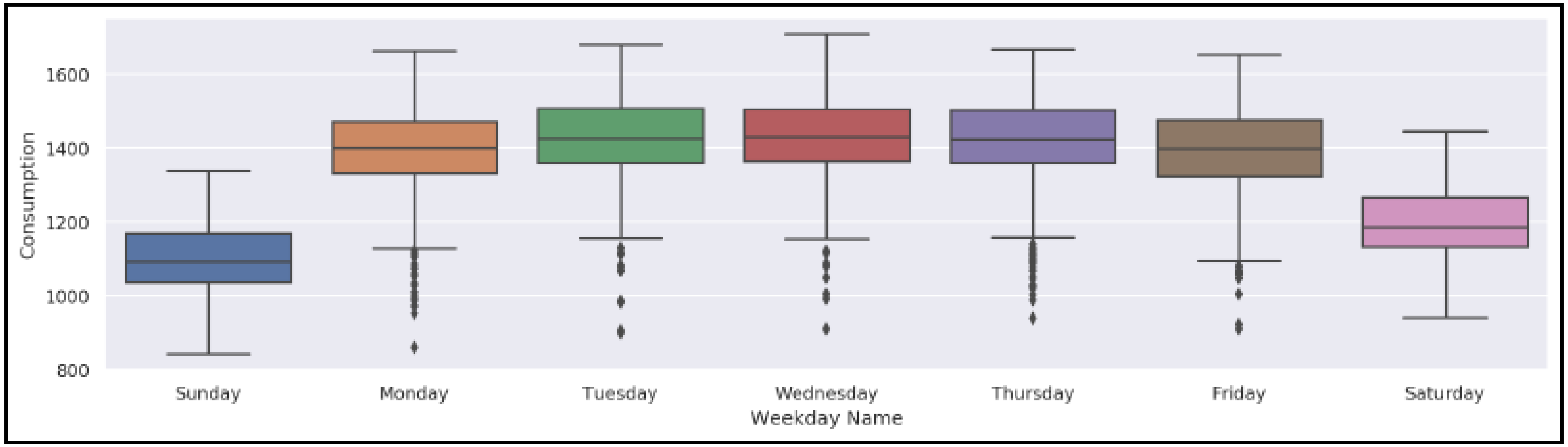
Grouping Time series data

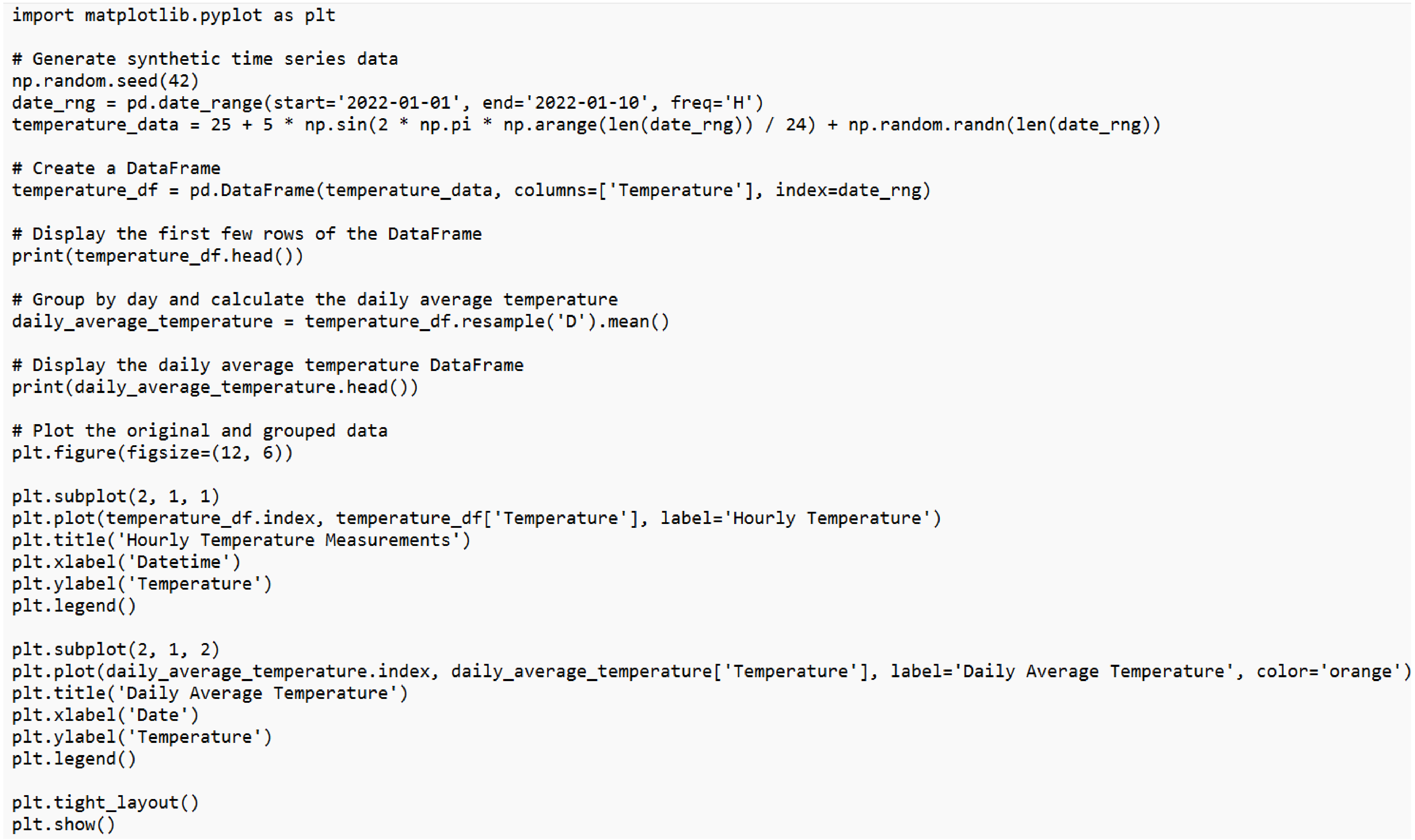
- Group the data by months
- fig, axes = plt.subplots(3, 1, figsize=(8, 7), sharex=True)
- for name, ax in zip(['Consumption', 'Solar', 'Wind'], axes):
- sns.boxplot(data=df_power, x='Month', y=name, ax=ax)
- ax.set_ylabel('GWh')
- ax.set_title(name)
- if ax != axes[-1]:
- ax.set_xlabel('')



Grouping Time series data

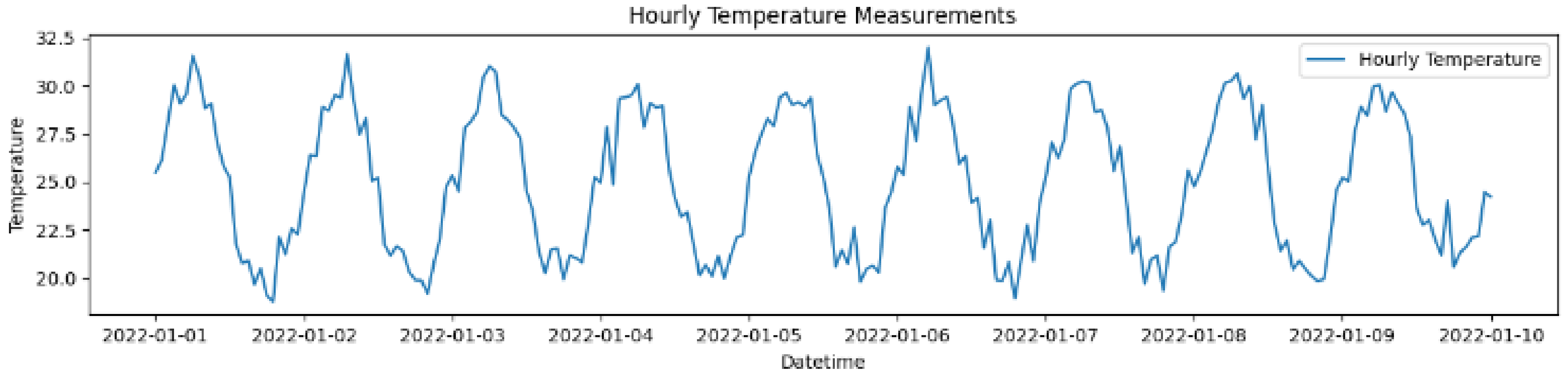
- Group the consumption of electricity by the day of the week
- sns.boxplot(data=df_power, x='Weekday Name', y='Consumption');

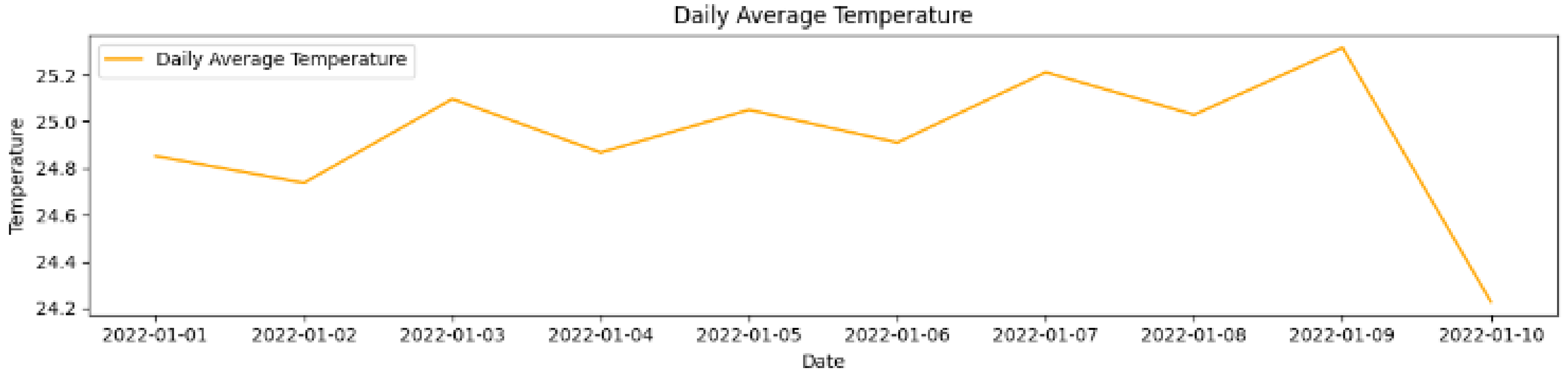




```
import matplotlib.pyplot as plt

# Generate synthetic time series data
np.random.seed(42)
date_rng = pd.date_range(start='2022-01-01', end='2022-01-10', freq='H')
temperature_data = 25 + 5 * np.sin(2 * np.pi * np.arange(len(date_rng)) / 24) + np.random.randn(len(date_rng))

# Create a DataFrame
temperature_df = pd.DataFrame(temperature_data, columns=['Temperature'], index=date_rng)

# Display the first few rows of the DataFrame
print(temperature_df.head())

# Group by day and calculate the daily average temperature
daily_average_temperature = temperature_df.resample('D').mean()

# Display the daily average temperature DataFrame
print(daily_average_temperature.head())

# Plot the original and grouped data
plt.figure(figsize=(12, 6))

plt.subplot(2, 1, 1)
plt.plot(temperature_df.index, temperature_df['Temperature'], label='Hourly Temperature')
plt.title('Hourly Temperature Measurements')
plt.xlabel('Datetime')
plt.ylabel('Temperature')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(daily_average_temperature.index, daily_average_temperature['Temperature'], label='Daily Average Temperature', color='orange')
plt.title('Daily Average Temperature')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.legend()

plt.tight_layout()
plt.show()
```

```
                     Temperature
2022-01-01 00:00:00    25.496714
2022-01-01 01:00:00    26.155831
2022-01-01 02:00:00    28.147689
2022-01-01 03:00:00    30.058564
2022-01-01 04:00:00    29.095974
            Temperature
2022-01-01    24.852362
2022-01-02    24.737044
2022-01-03    25.096827
2022-01-04    24.867419
2022-01-05    25.050503
```

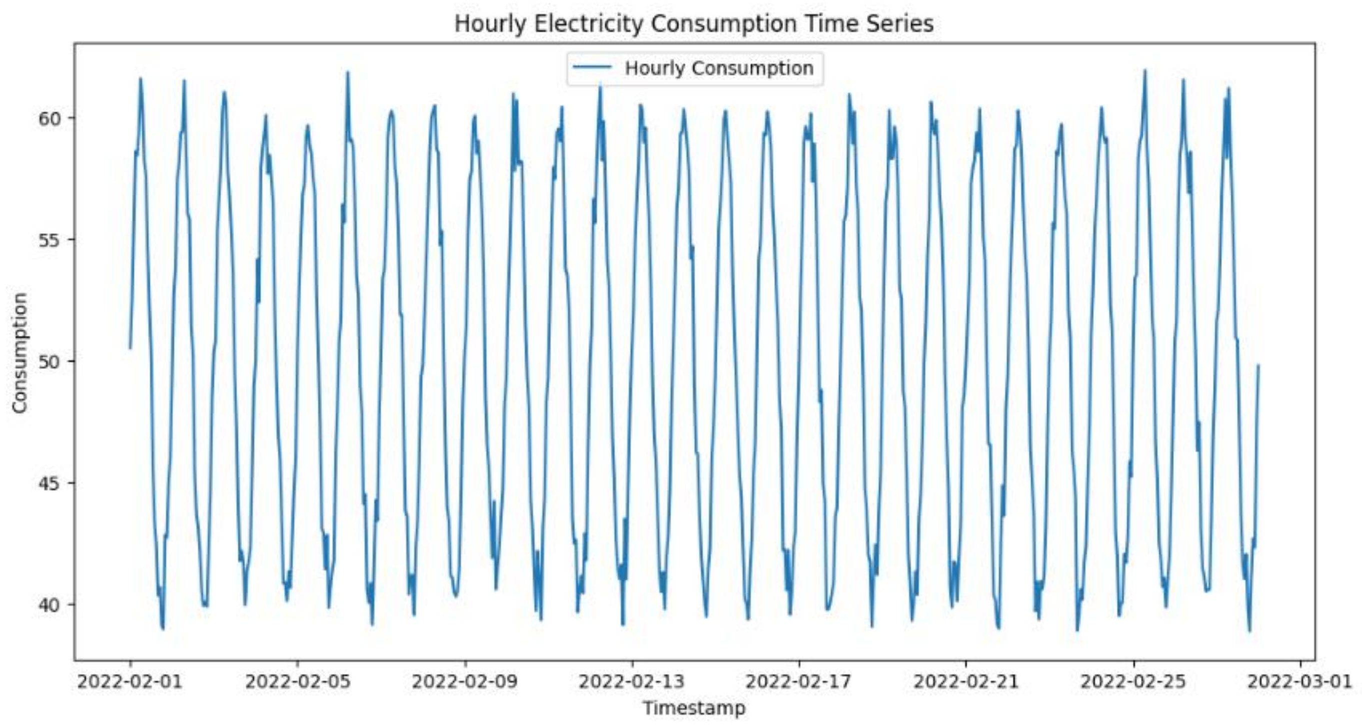


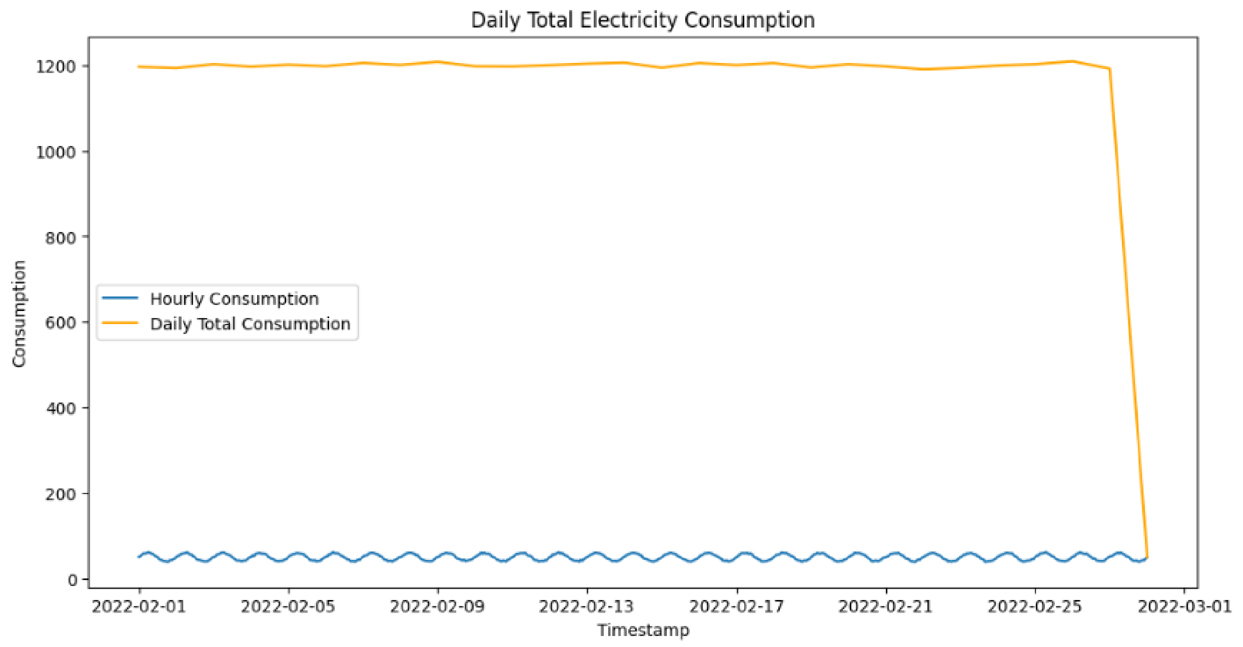


Exercise-5

- You are given a time series dataset containing hourly electricity consumption data for a residential area over a month. Your goal is to perform time series analysis and create a grouped summary to better understand consumption patterns.

- Load the dataset and convert the "Timestamp" column to the datetime format.
- Set the "Timestamp" column as the index of the DataFrame.
- Plot the original time series to visualize the hourly electricity consumption pattern.
- Group the data by day and calculate the daily total electricity consumption.
- Plot the original time series along with the daily total electricity consumption to observe any patterns or trends.


Hourly Electricity Consumption Time Series
Hourly Consumption
Consumption
60
55
50
45
40
2022-02-01
2022-02-05
2022-02-09
2022-02-13
2022-02-17
2022-02-21
2022-02-25
2022-03-01
Timestamp


Daily Total Electricity Consumption
Hourly Consumption
Daily Total Consumption
Consumption
1200
1000
800
600
400
200
0
2022-02-01
2022-02-05
2022-02-09
2022-02-13
2022-02-17
2022-02-21
2022-02-25
2022-03-01
Timestamp