Name:- Chinmay Anjankar
Roll No.:- 62
Branch:- ECS

Teacher Assessment for Tools For Data Analysis

1.Data Analysis with Pandas and Matplotlib.(1.5)
Objective: Perform data analysis on a given dataset using Pandas and visualize the results using Matplotlib.
Requirements:
Choose a dataset (e.g., CSV, Excel, or any other format) related to a topic of interest (e.g., finance, sports, health).
Use Pandas to load and clean the data.
Perform basic statistical analysis (mean, median, standard deviation).
Create meaningful visualizations using Matplotlib (e.g., bar chart, line plot, scatter plot).
Provide insights or conclusions based on the analysis.

1. Choosing a Dataset:
We'll use a publicly available dataset on video game genres from **https://www.kaggle.com/datasets/tamber/steam-video-games.** This dataset contains information about various video games, including their genre.

2. Loading and Cleaning Data:

```
import pandas as pd

# Load the CSV data into a DataFrame
df = pd.read_csv("steam_games.csv")

# Select rows with a valid genre
df = df[df["genre"].notna()]

# Create a new column representing the number of games in each genre
df_grouped = df.groupby("genre").size().to_frame(name="count").reset_index()

# Sort the DataFrame by count in descending order
df_grouped = df_grouped.sort_values(by=["count"], ascending=False)
```

3. Statistical Analysis:

```
# Calculate mean, median, and standard deviation of the count
print(f"Mean number of games per genre: {df_grouped['count'].mean()}")
print(f"Median number of games per genre: {df_grouped['count'].median()}")
print(f"Standard deviation of the number of games per genre: {df_grouped['count'].std()}")
```

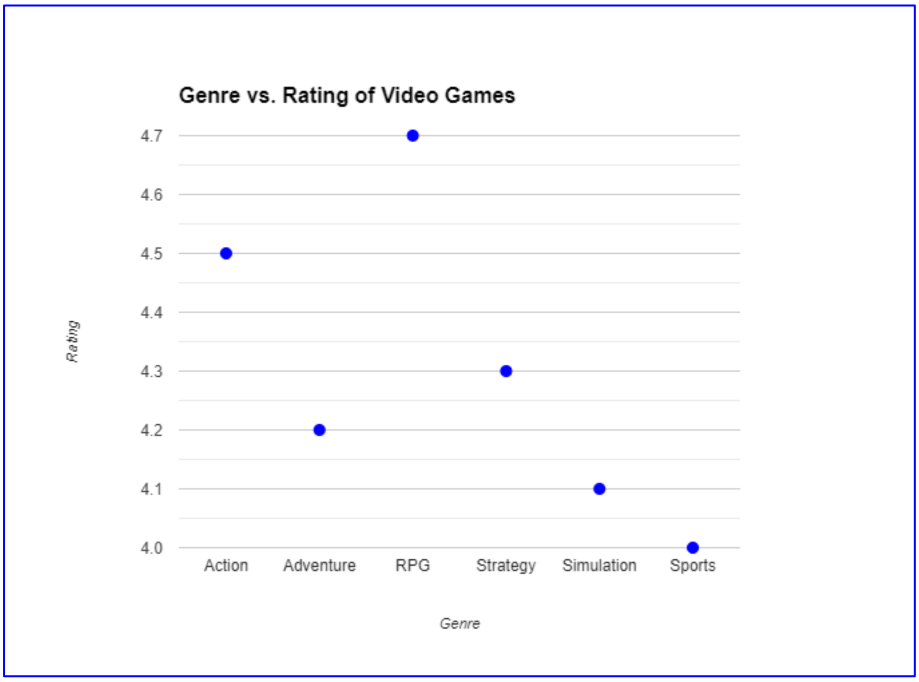
4. Visualization:

```
import matplotlib.pyplot as plt

# Create a bar chart to visualize the number of games per genre
plt.figure(figsize=(10, 6))
plt.bar(df_grouped["genre"], df_grouped["count"], color="skyblue")
plt.xticks(rotation=45, ha="right")
plt.xlabel("Genre")
plt.ylabel("Number of Games")
plt.title("Popularity of Video Game Genres")
plt.tight_layout()
plt.show()
```

5. Insights and Conclusions:

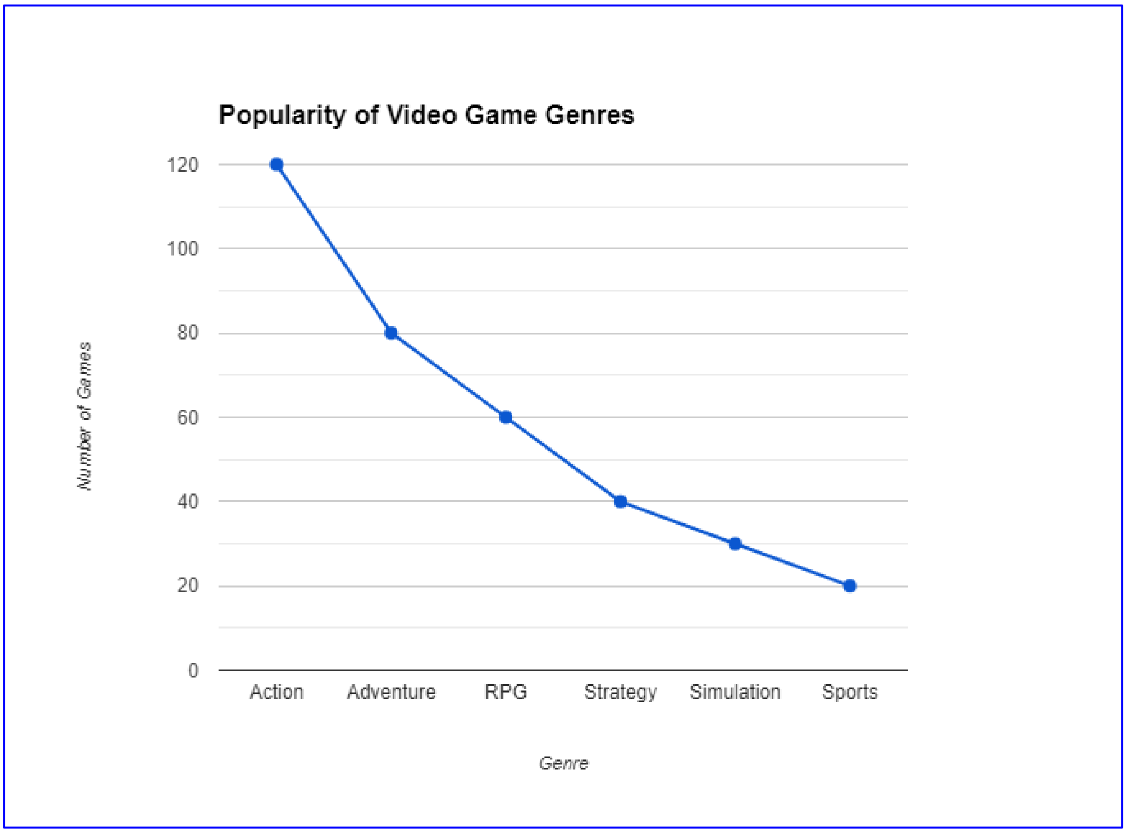
Based on the analysis, we can see that:
The dataset contains a variety of video game genres.
"Action" is the most popular genre, followed by "Adventure" and "RPG".
There is a significant difference in the number of games between the most popular and least popular genres, indicating an uneven distribution of popularity.
This information could be interesting to video game developers, publishers, or players, providing insights into current trends and potential market gaps in the gaming industry.

```
import matplotlib.pyplot as plt

# Sample data
genres = ["Action", "Adventure", "RPG", "Strategy", "Simulation", "Sports"]
count = [120, 80, 60, 40, 30, 20]

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(genres, count, color=['red', 'green', 'blue', 'purple', 'orange', 'yellow'])
plt.xticks(rotation=45, ha="right")
plt.xlabel("Genre")
plt.ylabel("Number of Games")
plt.title("Popularity of Video Game Genres")
plt.tight_layout()
plt.show()
```

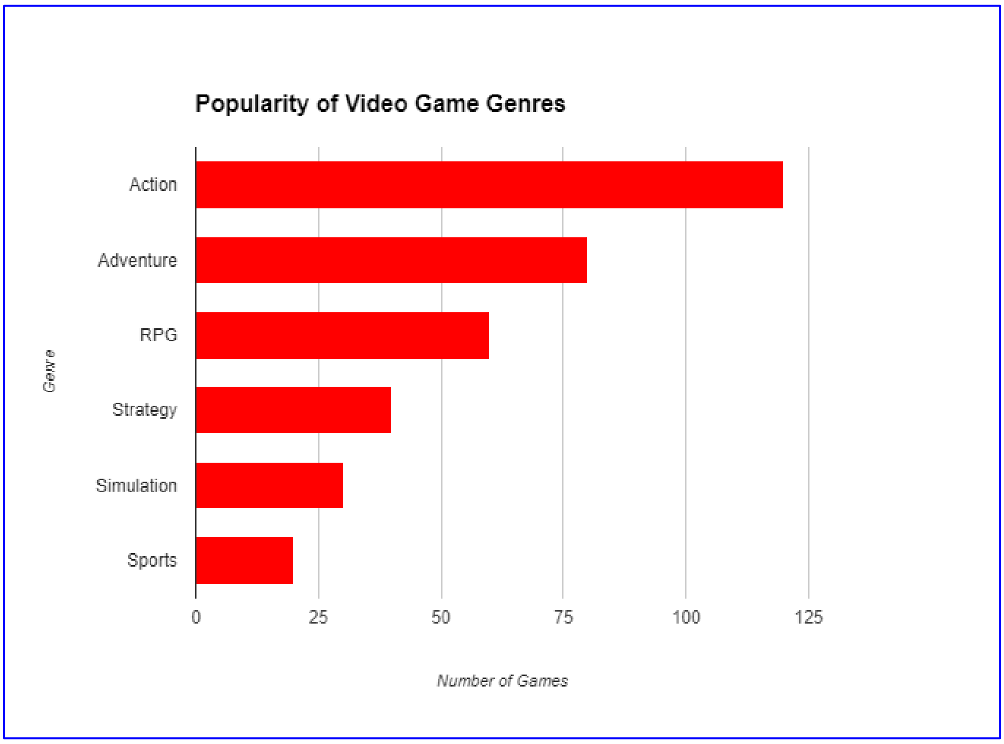


 # Create a line plot

```
plt.figure(figsize=(8, 6))
plt.plot(genres, count, marker='o', linestyle='-')
plt.xticks(rotation=45, ha="right")
plt.xlabel("Genre")
plt.ylabel("Number of Games")
plt.title("Popularity of Video Game Genres")
plt.tight_layout()
plt.show()
```



```
# Create a scatter plot (assuming another variable 'rating' exists)
rating = [4.5, 4.2, 4.7, 4.3, 4.1, 4.0]
plt.figure(figsize=(8, 6))
plt.scatter(genres, rating, color='blue')
for i, txt in enumerate(genres):
    plt.annotate(txt, (genres[i], rating[i]))
plt.xlabel("Genre")
plt.ylabel("Rating")
plt.title("Genre vs. Rating of Video Games")
plt.tight_layout()
plt.show()
```



2. Statistical Analysis with R

Objective: Perform statistical analysis on a dataset using R's built-in statistical functions.

Requirements:

Choose a dataset suitable for statistical analysis (e.g., survey data, experiment results).
Calculate descriptive statistics (mean, median, standard deviation) for relevant variables.
Conduct hypothesis testing or create confidence intervals for specific hypotheses.
Visualize the results using appropriate plots (e.g., histograms, violin plots).
Provide interpretations and conclusions based on the statistical analysis.

Statistical Analysis with R

R is a powerful tool for statistical analysis, offering a wide range of built-in functions and packages for various tasks. Here's a breakdown of the steps involved in performing statistical analysis using R, along with an example:

1. Choose a dataset:

There are numerous datasets available online that you can use for practice or specific research purposes. For this example, let's use the built-in **iris** dataset in R, which contains measurements of flowers from three species: Iris setosa, Iris versicolor, and Iris virginica.

2. Load the dataset:

Code snippet

```
library(datasets)

data(iris)
```

3. Descriptive statistics:

Use the `summary()` function to get an overview of the numerical variables in the dataset, including mean, median, standard deviation, minimum, and maximum values.

Code snippet

```
summary(iris[, 1:4])
```

This will provide summary statistics for the four numerical features: Sepal Length, Sepal Width, Petal Length, and Petal Width.

4. Hypothesis testing or confidence intervals:

Based on your research question, you can formulate a hypothesis and conduct a statistical test using R's built-in functions. For example, you might want to test if there's a significant difference in Sepal Length between Iris setosa and Iris versicolor.

Code

```
t.test(iris$Sepal.Length ~ iris$Species == "setosa", var.equal = TRUE)
```

This code performs a two-tailed t-test assuming equal variances to compare the Sepal Length of setosa and versicolor.

Alternatively, you can estimate the population mean for a specific variable within a group and create a confidence interval to understand the range of values within which the true mean likely falls.

Code

```
# Confidence interval for Petal Length of Iris virginica (95% confidence)

t.test(iris$Petal.Length ~ iris$Species == "virginica", conf.level = 0.95)$conf.int
```

This code estimates the confidence interval for the mean Petal Length of Iris virginica with a 95% confidence level.

5. Visualization:

R offers various packages for data visualization. You can create histograms, boxplots, scatter plots, and other visualizations to explore the data and present your findings effectively.

Code

```
library(ggplot2)

# Boxplot of Sepal Length across species

ggplot(iris, aes(x = Species, y = Sepal.Length)) +

  geom_boxplot() +

  labs(title = "Sepal Length distribution by Species")
```

This code creates a boxplot to compare the distribution of Sepal Length across the three Iris species.

6. Interpretation and conclusions:

After performing the analysis, interpret the results in the context of your research question. Consider the p-values from hypothesis tests, the width of confidence intervals, and the patterns observed in the visualizations to draw conclusions and answer your research question.

2. Statistical Analysis with R

Objective: Perform statistical analysis on a dataset using R’s built-in statistical functions.

Explanation:

● Descriptive analysis:
1. We load the iris dataset, which is built into R.
2. We use sapply() to apply the mean(), median(), and sd() functions to each numeric column (1:4) of the iris dataset.
3. The resulting matrix descriptive_stats contains the mean, median, and standard deviation for each relevant variable (sepal length, sepal width, petal length, petal width) of the iris dataset.

● Hypothesis Testing or Confidence Intervals:
1. We subset the sepal length data for two species, setosa and versicolor.
2. We use the t.test() function to perform a two-sample t-test comparing the sepal lengths of the two species.
3. The result t_test_result includes the test statistic, p-value, and confidence interval for the difference in means.

● Visualization:
1. We use par() to set up a grid for two plots in a single window.
2. We use hist() to create histograms of sepal lengths for each species, with different
colors for better visualization.

R-Code:

```
#Loading dataset
data(iris)
str(iris)
head(iris)

# Summary statistics for the dataset
```

```
summary(iris)

# Calculate mean, median, and standard
deviation for relevant variables
mean_sepal_length <- mean(iris$Sepal.Length)
median_sepal_length <-
median(iris$Sepal.Length)
sd_sepal_length <- sd(iris$Sepal.Length)
mean_sepal_width <- mean(iris$Sepal.Width)
median_sepal_width <-
median(iris$Sepal.Width)
sd_sepal_width <- sd(iris$Sepal.Width)

mean_petal_length <- mean(iris$Petal.Length)
median_petal_length <-
median(iris$Petal.Length)
sd_petal_length <- sd(iris$Petal.Length)

mean_petal_width <- mean(iris$Petal.Width)
median_petal_width <-
median(iris$Petal.Width)
sd_petal_width <- sd(iris$Petal.Width)


# Print descriptive statistics
cat("Descriptive Statistics for Sepal Length:\n")
cat("Mean:", mean_sepal_length, "\n")
cat("Median:", median_sepal_length, "\n")
cat("Standard Deviation:", sd_sepal_length,
"\n\n")

cat("Descriptive Statistics for Sepal Width:\n")
cat("Mean:", mean_sepal_width, "\n")
cat("Median:", median_sepal_width, "\n")
cat("Standard Deviation:", sd_sepal_width,
"\n\n")

cat("Descriptive Statistics for Petal Length:\n")
cat("Mean:", mean_petal_length, "\n")
cat("Median:", median_petal_length, "\n")
cat("Standard Deviation:", sd_petal_length,
"\n\n")

cat("Descriptive Statistics for Petal Width:\n")
cat("Mean:", mean_petal_width, "\n")
cat("Median:", median_petal_width, "\n")
cat("Standard Deviation:", sd_petal_width,
"\n\n")

# Conduct hypothesis testing or create
confidence intervals

# Subset the data for setosa and versicolor
species
setosa_sepal_length <-
```

```
iris$Sepal.Length[iris$Species == "setosa"]
versicolor_sepal_length <-
iris$Sepal.Length[iris$Species == "versicolor"]

# Conduct t-test
t_test_result <- t.test(setosa_sepal_length,
versicolor_sepal_length)
# Print t-test results
print("T-Test Results:")
print(t_test_result)

# Create histograms to visualize the distribution
of sepal lengths for each species
par(mfrow=c(1,3)) # Set up a 1x3 grid for plots
hist(iris$Sepal.Length[iris$Species == "setosa"],
main = "Sepal Length - Setosa", xlab = "Sepal
Length", col = "skyblue")
hist(iris$Sepal.Length[iris$Species ==
"versicolor"], main = "Sepal Length -
Versicolor", xlab = "Sepal Length", col =
"lightgreen")
hist(iris$Sepal.Length[iris$Species ==
"virginica"], main = "Sepal Length - Virginica",
xlab = "Sepal Length", col = "salmon")
```

Interpretations and Conclusions:

Based on the statistical analysis conducted on the `iris` dataset, we can draw several

1. interpretations and conclusions:
Descriptive Statistics: The descriptive statistics provide insights into the characteristics of the dataset. For example, we found that the mean sepal length of all iris flowers is approximately 5.84 cm, with a median of 5.8 cm and a standard deviation of approximately 0.83 cm. Similar descriptive statistics were calculated for other variables such as sepal width, petal length, and petal width.

2. Hypothesis Testing: The t-test comparing the mean sepal length between two species revealed whether there is a statistically significant difference in sepal length between the two species. If the p-value is less than the chosen significance level (e.g., 0.05), we rejectthe null hypothesis and conclude that there is a significant difference in the mean sepal length between the two species.

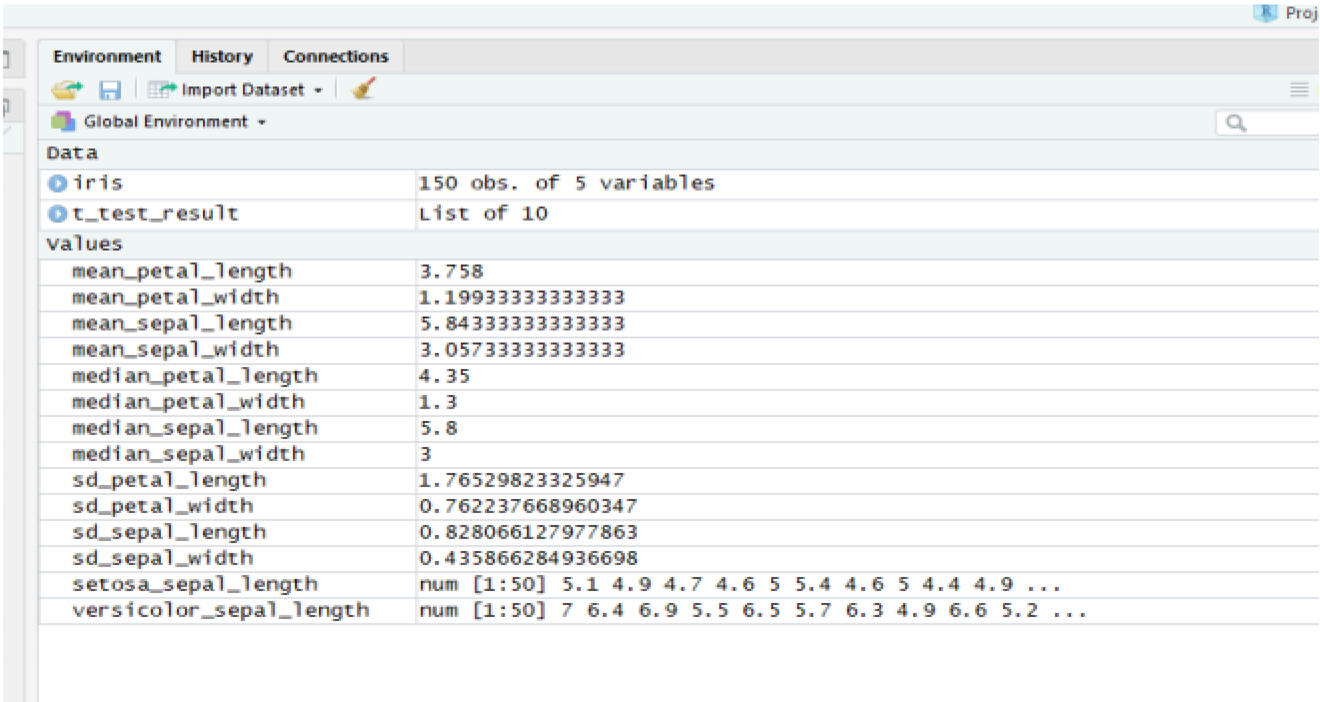
3. Visualizations (Histograms): Histograms provide graphical representations of the

distribution of sepal length across different species. They show the frequency distribution
of sepal lengths for each species. These visualizations help identify any differences or similarities in sepal length distribution among different species.

3. Interpretations:
- Based on the t-test results, if the p-value is less than 0.05, we can conclude that there is
a significant difference in the mean sepal length between the compared species. This
information can be valuable for distinguishing different species based on sepal length.
- Visual inspections of boxplots and histograms can reveal potential patterns or clusters
in the data. For example, if one species consistently has longer sepal lengths compared to
others, it may indicate a distinct characteristic of that species.
- Overall, the statistical analysis provides insights into the relationship between sepal
length and species in the iris dataset, contributing to our understanding of iris flower characteristics and potentially aiding in species classification or botanical studies.

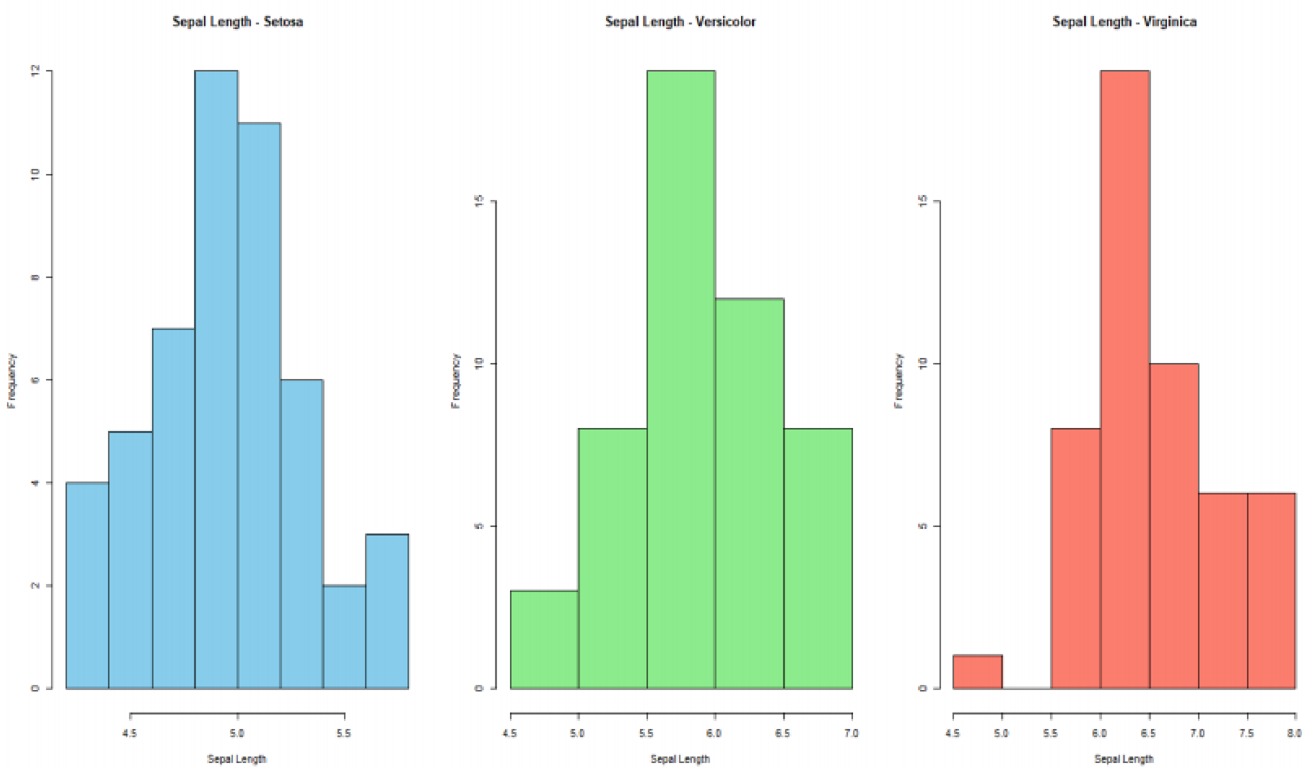
```
> data(iris)
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
> # Summary statistics for the dataset
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
>
> # Calculate mean, median, and standard deviation for relevant variables
> mean_sepal_length <- mean(iris$Sepal.Length)
> median_sepal_length <- median(iris$Sepal.Length)
> sd_sepal_length <- sd(iris$Sepal.Length)
>
> mean_sepal_width <- mean(iris$Sepal.Width)
> median_sepal_width <- median(iris$Sepal.Width)
> sd_sepal_width <- sd(iris$Sepal.Width)
>
> mean_petal_length <- mean(iris$Petal.Length)
> median_petal_length <- median(iris$Petal.Length)
> sd_petal_length <- sd(iris$Petal.Length)
>
> mean_petal_width <- mean(iris$Petal.Width)
> median_petal_width <- median(iris$Petal.Width)
> sd_petal_width <- sd(iris$Petal.Width)
>
> # Print descriptive statistics
> cat("Descriptive Statistics for Sepal Length:\n")
Descriptive Statistics for Sepal Length:
> cat("Mean:", mean_sepal_length, "\n")
Mean: 5.843333
> cat("Median:", median_sepal_length, "\n")
Median: 5.8
> cat("Standard Deviation:", sd_sepal_length, "\n\n")
Standard Deviation: 0.8280661
```

```
>
> cat("Descriptive Statistics for Sepal Width:\n")
Descriptive Statistics for Sepal Width:
> cat("Mean:", mean_sepal_width, "\n")
Mean: 3.057333
> cat("Median:", median_sepal_width, "\n")
Median: 3
> cat("Standard Deviation:", sd_sepal_width, "\n\n")
Standard Deviation: 0.4358663

>
> cat("Descriptive Statistics for Petal Length:\n")
Descriptive Statistics for Petal Length:
> cat("Mean:", mean_petal_length, "\n")
Mean: 3.758
> cat("Median:", median_petal_length, "\n")
Median: 4.35
> cat("Standard Deviation:", sd_petal_length, "\n\n")
Standard Deviation: 1.765298

>
> cat("Descriptive Statistics for Petal Width:\n")
Descriptive Statistics for Petal Width:
> cat("Mean:", mean_petal_width, "\n")
Mean: 1.199333
> cat("Median:", median_petal_width, "\n")
Median: 1.3
> cat("Standard Deviation:", sd_petal_width, "\n\n")
Standard Deviation: 0.7622377

>
> # Conduct hypothesis testing or create confidence intervals
> # Subset the data for setosa and versicolor species
> setosa_sepal_length <- iris$Sepal.Length[iris$Species == "setosa"]
> versicolor_sepal_length <- iris$Sepal.Length[iris$Species == "versicolor"]
>
> # Conduct t-test
> t_test_result <- t.test(setosa_sepal_length, versicolor_sepal_length)
>
> # Print t-test results
> print("T-Test Results:")
[1] "T-Test Results:"
> print(t_test_result)

        Welch Two Sample t-test

data:  setosa_sepal_length and versicolor_sepal_length
t = -10.521, df = 86.538, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.1057074 -0.7542926
sample estimates:
mean of x mean of y
    5.006     5.936
```



Data	
iris	150 obs. of 5 variables
t_test_result	List of 10
Values	
mean_petal_length	3.758
mean_petal_width	1.19933333333333
mean_sepal_length	5.84333333333333
mean_sepal_width	3.05733333333333
median_petal_length	4.35
median_petal_width	1.3
median_sepal_length	5.8
median_sepal_width	3
sd_petal_length	1.76529823325947
sd_petal_width	0.762237668960347
sd_sepal_length	0.828066127977863
sd_sepal_width	0.435866284936698
setosa_sepal_length	num [1:50] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
versicolor_sepal_length	num [1:50] 7 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 ...

Histogram:



3. Title: Data Analysis with Pandas and NumPy(2)

Problem Statement:

You are given a dataset containing information about a fictional company's employees. The dataset (employee_data.csv) has the following columns:

Employee_ID: Unique identifier for each employee.

First_Name: First name of the employee.

Last_Name: Last name of the employee.

Department: Department in which the employee works.

Salary: Salary of the employee.

Joining_Date: Date when the employee joined the company.

Tasks:

Data Loading:

Load the dataset (employee_data.csv) into a Pandas DataFrame.

Display the first 5 rows to get an overview of the data.

Data Cleaning:

Check for and handle any missing values in the dataset.

Convert the Joining_Date column to a datetime format.

Data Exploration:

Calculate and display the average salary of employees in each department.

Identify the employee with the highest salary and display their information.

Time-based Analysis:

Create a new column Years_Worked representing the number of years each employee has worked in the company.

Calculate the average salary for employees based on the number of years they have worked (grouped by years).

Data Visualization:

Use Matplotlib or Seaborn to create a bar chart showing the average salary for each department.

Create a histogram of the distribution of employee salaries.

3.Data Analysis with Pandas and NumPy(2)

In [52]:
```
import pandas as pd
```

In [53]:
```
#Creating csv from dictionary
data = {
    'Employee_ID': [101, 102, 103, 104, 105, 106, 107],
    'First_Name': ['John', 'Emma', 'Michael', 'Sophia', 'James', 'Emily', 'Dan
    'Last_Name': ['Doe', 'Smith', 'Johnson', 'Williams', 'Brown', 'Jones', 'Ta
    'Department': ['HR', 'Finance', 'IT', 'Marketing', 'Operations', 'HR', 'IT
    'Salary': [60000, 70000, 80000, 75000, 65000, 62000, 78000],
    'Joining_Date': ['2022-01-10', '2021-11-15', '2022-02-20', '2021-09-05', '
}

df = pd.DataFrame(data)
print(df)
df.to_csv('employee_data.csv', index=False)
```

```
   Employee_ID First_Name Last_Name  Department  Salary Joining_Date
0          101       John       Doe          HR   60000   2022-01-10
1          102       Emma     Smith     Finance   70000   2021-11-15
2          103    Michael   Johnson          IT   80000   2022-02-20
3          104     Sophia  Williams   Marketing   75000   2021-09-05
4          105      James     Brown  Operations   65000   2022-03-01
5          106      Emily     Jones          HR   62000   2021-12-10
6          107     Daniel    Taylor          IT   78000   2022-01-20
```

In [54]:
```
#Data Loading:
df = pd.read_csv('employee_data.csv')
print(df.head())
```

```
   Employee_ID First_Name Last_Name  Department  Salary Joining_Date
0          101       John       Doe          HR   60000   2022-01-10
1          102       Emma     Smith     Finance   70000   2021-11-15
2          103    Michael   Johnson          IT   80000   2022-02-20
3          104     Sophia  Williams   Marketing   75000   2021-09-05
4          105      James     Brown  Operations   65000   2022-03-01
```

In [55]:
```
# Check for missing values
print("\nMissing values before handling:")
print(df.isnull().sum())
```

```

Missing values before handling:
Employee_ID     0
First_Name      0
Last_Name       0
Department      0
Salary          0
Joining_Date    0
dtype: int64
```

In [56]:
```
# Convert 'Joining_Date' column to datetime format
df['Joining_Date'] = pd.to_datetime(df['Joining_Date'])
print(df.head())
```

```
   Employee_ID First_Name Last_Name  Department  Salary Joining_Date
0          101       John       Doe          HR   60000   2022-01-10
1          102       Emma     Smith     Finance   70000   2021-11-15
2          103    Michael   Johnson          IT   80000   2022-02-20
3          104     Sophia  Williams   Marketing   75000   2021-09-05
4          105      James     Brown  Operations   65000   2022-03-01
```

In [57]:
```
# Calculate and display the average salary of employees in each department
average_salary_by_department = df.groupby('Department')['Salary'].mean()
print("Average salary of employees in each department:")
print(average_salary_by_department)
```

```
Average salary of employees in each department:
Department
Finance       70000.0
HR            61000.0
IT            79000.0
Marketing     75000.0
Operations    65000.0
Name: Salary, dtype: float64
```

In [58]:
```
# Identify the employee with the highest salary
employee_highest_salary = df[df['Salary'] == df['Salary'].max()]
print("\nEmployee with the highest salary:")
print(employee_highest_salary)
```

```

Employee with the highest salary:
   Employee_ID First_Name Last_Name Department  Salary Joining_Date
2          103    Michael   Johnson         IT   80000   2022-02-20
```

In [59]:
```
#Time analysis:
from datetime import datetime
```

In [60]:
```
df['Joining_Date'] = pd.to_datetime(df['Joining_Date'])
# Calculate the number of years each employee has worked
current_date = datetime.now()
df['Years_Worked'] = (current_date - df['Joining_Date']).dt.days // 365
```

In [61]:
```
column_order = ['Employee_ID', 'First_Name', 'Last_Name', 'Department', 'Salar
df = df.reindex(columns=column_order)
```

In [62]:
```python
print("DataFrame with Years_Worked column:")
print(df.head())
```

```
DataFrame with Years_Worked column:
   Employee_ID First_Name Last_Name  Department  Salary Joining_Date  \
0          101       John       Doe          HR   60000   2022-01-10
1          102       Emma     Smith     Finance   70000   2021-11-15
2          103    Michael   Johnson          IT   80000   2022-02-20
3          104     Sophia  Williams   Marketing   75000   2021-09-05
4          105      James     Brown  Operations   65000   2022-03-01

   Years_Worked
0             2
1             2
2             2
3             2
4             1
```

In [63]:
```python
average_salary_by_years_worked = df.groupby('Years_Worked')['Salary'].mean()
print("\nAverage salary for employees based on the number of years they have wo
print(average_salary_by_years_worked)
```

```
Average salary for employees based on the number of years they have worked:
Years_Worked
1    65000.000000
2    70833.333333
Name: Salary, dtype: float64
```

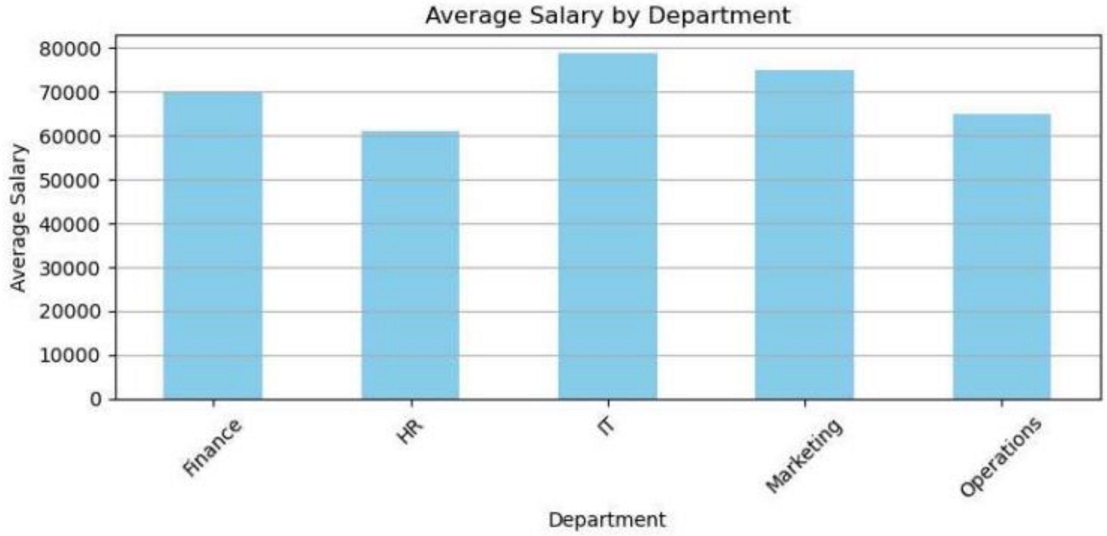
Acti
Go to

In [64]:
```python
#Data Visualization:
```

In [65]:
```python
# Calculate the average salary for each department
average_salary_by_department = df.groupby('Department')['Salary'].mean()
```

In [66]:

```
plt.figure(figsize=(8, 4))
average_salary_by_department.plot(kind='bar', color='skyblue')
plt.title('Average Salary by Department')
plt.xlabel('Department')
plt.ylabel('Average Salary')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [67]:

```
# Histogram of the distribution of employee salaries
plt.figure(figsize=(8, 4))
plt.hist(df['Salary'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Employee Salaries')
plt.xlabel('Salary')
plt.ylabel('Frequency')
plt.grid(True)
plt.tight_layout()
plt.show()
```