GJU
الجامعة الألمانية الأردنية
German Jordanian University

Exploratory Data Analysis

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Introduction to Python for Data Science

- Python is a versatile programming language widely used in data science for its simplicity and extensive libraries.

- Importance for business students: Python allows analyzing and interpreting data, crucial for decision-making in various business domains.

Installing Python and Required Libraries

- Anaconda distribution is recommended for installing Python, as it comes with essential libraries pre-installed.

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Reading Data from Files (Pandas)

- Pandas is a powerful library for data manipulation and analysis, providing data structures and functions to work with structured data.
  - Loading data from files:
    - CSV: pd.read_csv('filename.csv')
    - Excel: pd.read_excel('filename.xlsx')
    - JSON: pd.read_json('filename.json')

GJU
الجامعة الألمانية الأردنية
German Jordanian University

data.csv

This dataset includes the following columns:

- Student_ID: Unique identifier for each student.
- Name: Name of the student in English alphabet.
- Gender: Gender of the student.
- Exam_Score: Exam score achieved by the student.
- Project_Grade: Grade obtained for a business project.
- Internship_Status: Indicates whether the student completed an internship (Completed/Not Completed).

Student_ID	Name	Gender	Exam_Score	Project_Grade	Internship_Status
1	Mohammed Ali	Male	85	90	Completed
2	Fatima Ahmed	Female	78	85	Not Completed
3	Abdul Rahman Khalid	Male	92	88	Completed
4	Noor Ali	Female	88	92	Completed
5	Lina Mohammed	Female	80	78	Not Completed
6	Ahmed Abdullah	Male	90	85	Completed
7	Sara Ali	Female	95	94	Completed
8	Rayan Youssef	Male	85	80	Not Completed
9	Maryam Hussein	Female	75	72	Not Completed
10	Amira Abdul Rahman	Female	87	90	Completed

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Read data from CSV file

```
import pandas as pd

# Load data from CSV file
data = pd.read_csv('data.csv')

# Display the first few rows of the dataframe
print(data.head())
```

```
In [2]: runfile('C:/Users/user/DataAnalyticsEx.py', wdir='C:/Users/user')
   Student_ID                Name  ... Project_Grade  Internship_Status
0           1        Mohammed Ali  ...            90          Completed
1           2        Fatima Ahmed  ...            85      Not Completed
2           3  Abdul Rahman Khalid  ...            88          Completed
3           4            Noor Ali  ...            92          Completed
4           5       Lina Mohammed  ...            78      Not Completed

[5 rows x 6 columns]
```

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Plotting in Python

Plotting in Python is a fundamental aspect of data analysis and visualization. Python offers several powerful libraries for creating various types of plots and charts, making it a preferred choice for data scientists, analysts, and researchers. In this introduction, we'll explore some of the key plotting libraries in Python and how they are used for data visualization.

1. Matplotlib: Matplotlib is one of the most widely used plotting libraries in Python. It provides a MATLAB-like interface for creating static, interactive, and publication-quality plots. Matplotlib offers extensive customization options for creating line plots, scatter plots, histograms, bar plots, box plots, and more.

2. Seaborn: Seaborn is built on top of Matplotlib and provides a high-level interface for creating attractive and informative statistical graphics. It simplifies the process of creating complex visualizations such as categorical plots, distribution plots, pair plots, and regression plots. Seaborn also offers built-in themes and color palettes for enhancing the appearance of plots.

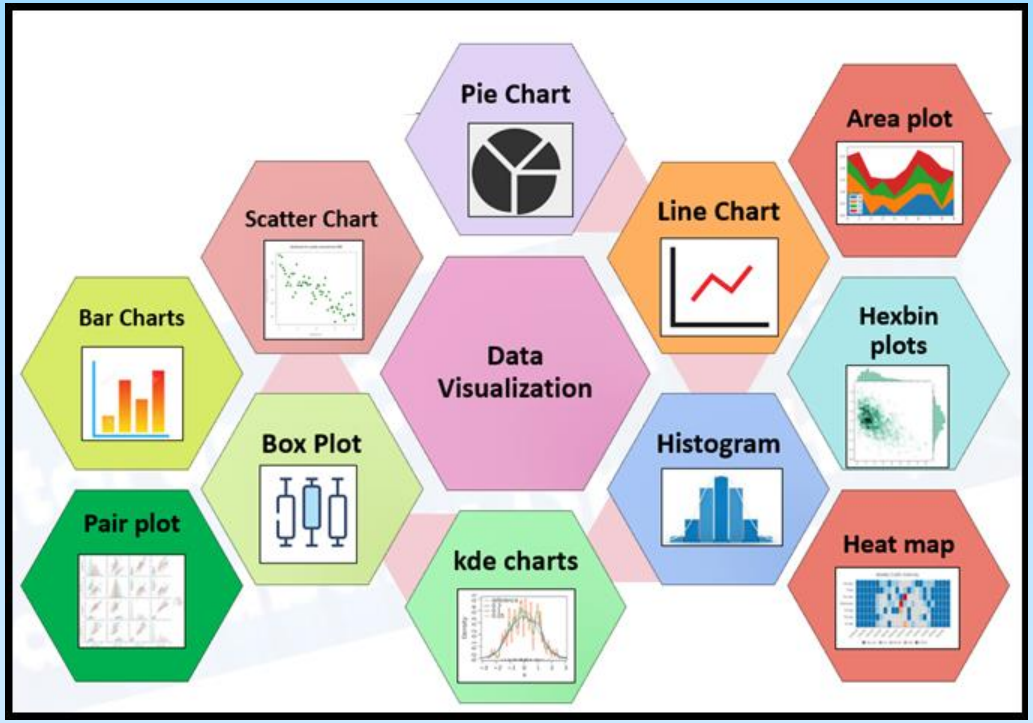
3. Pandas: Pandas, a popular data manipulation library in Python, also includes basic plotting functionality. It provides a convenient interface for creating plots directly from Pandas data structures such as Series and DataFrames. Pandas plotting functions are built on top of Matplotlib, making it easy to create simple visualizations with just a few lines of code.

4. Plotly: Plotly is a powerful library for creating interactive and web-based visualizations in Python. It supports a wide range of plot types, including scatter plots, line plots, bar plots, heatmap plots, and 3D plots. Plotly's interactive features allow users to explore and interact with data directly within the plot, making it suitable for creating dashboards and web applications.

Plotting in Python

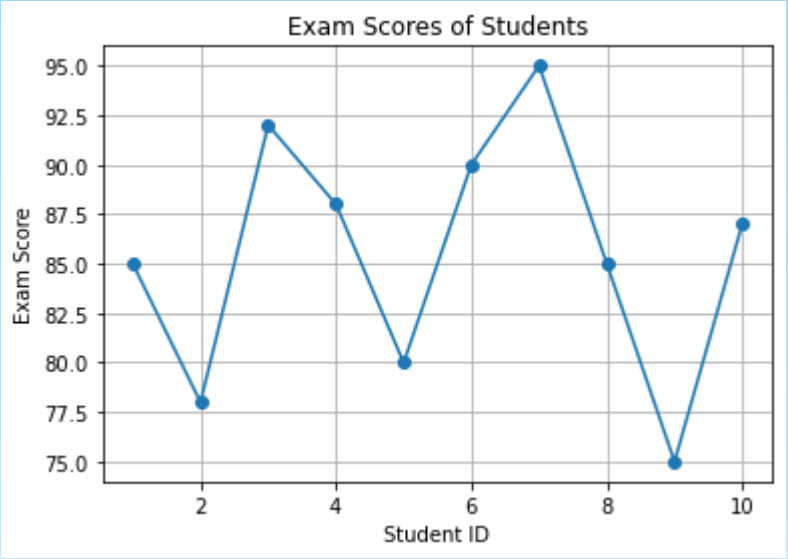
The following are some of the most commonly used plotting techniques in data analysis and visualization, and they can effectively represent various types of data and relationships.

1. Line Plot
2. Scatter Plot
3. Histogram
4. Bar Plot
5. Stacked Bar Plot
6. Box Plot
7. Pie Chart



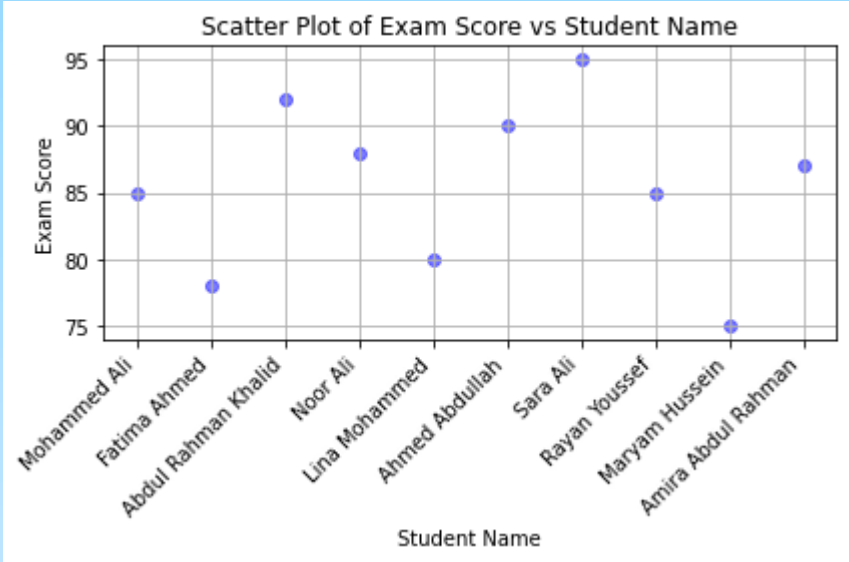
Line Plot


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

# Read data from CSV file
data = pd.read_csv('data.csv')

# Extracting data for the line plot
Student_ID = data['Student_ID']
Exam_Score = data['Exam_Score']

# Creating a line plot
plt.plot(Student_ID, Exam_Score, marker='o', linestyle='-')

# Adding labels and title
plt.xlabel('Student ID')
plt.ylabel('Exam Score')
plt.title('Exam Scores of Students')

# Displaying the plot
plt.grid(True)
plt.show()
```



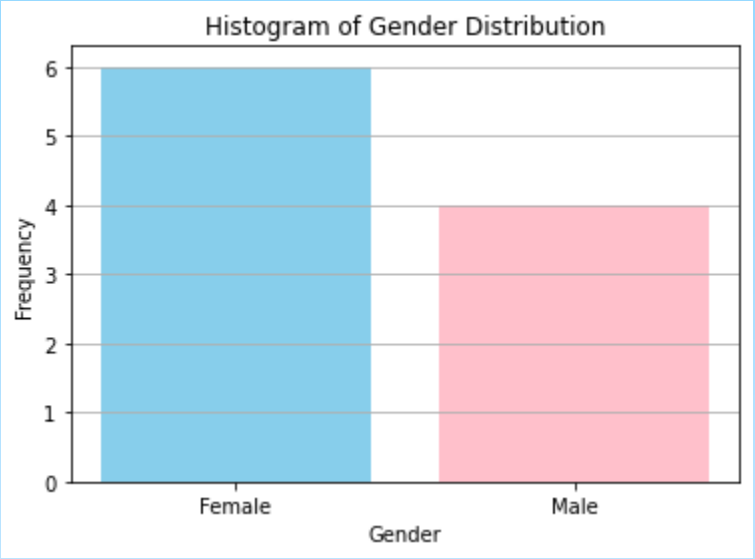
Scatter Plot


```
import pandas as pd
import matplotlib.pyplot as plt
# Read data from CSV file
data = pd.read_csv('data.csv')

# Extracting data for the scatter plot
Student_Name = data['Name']
Exam_Score = data['Exam_Score']


# Creating a scatter plot
plt.scatter(Student_Name, Exam_Score, color='blue', alpha=0.5)

# Adding student names as x-axis labels
plt.xticks(Student_Name, rotation=45, ha='right')

# Adding labels and title
plt.xlabel('Student Name')
plt.ylabel('Exam Score')
plt.title('Scatter Plot of Exam Score vs Student Name')
# Displaying the plot
plt.grid(True)
plt.tight_layout()
plt.show()
```



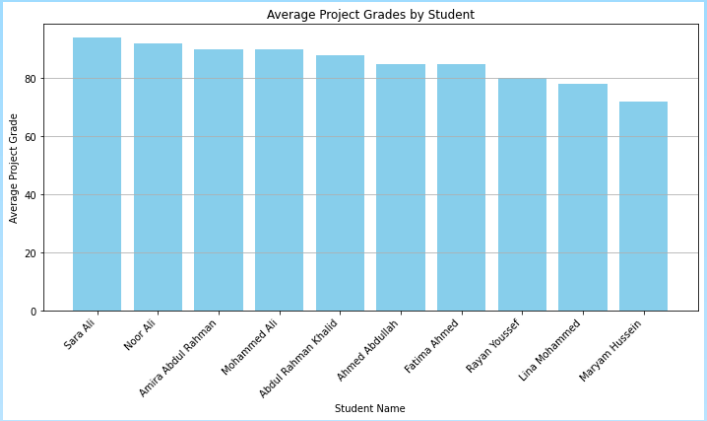
Histogram Plot


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

# Read data from CSV file
data = pd.read_csv('data.csv')

# Count the frequency of each gender category
gender_counts = data['Gender'].value_counts()

# Creating a histogram
plt.bar(gender_counts.index, gender_counts, color=['skyblue',
'pink'])

# Adding labels and title
plt.xlabel('Gender')
plt.ylabel('Frequency')
plt.title('Histogram of Gender Distribution')

# Displaying the plot
plt.grid(axis='y')
plt.show()
```



Bar Plot

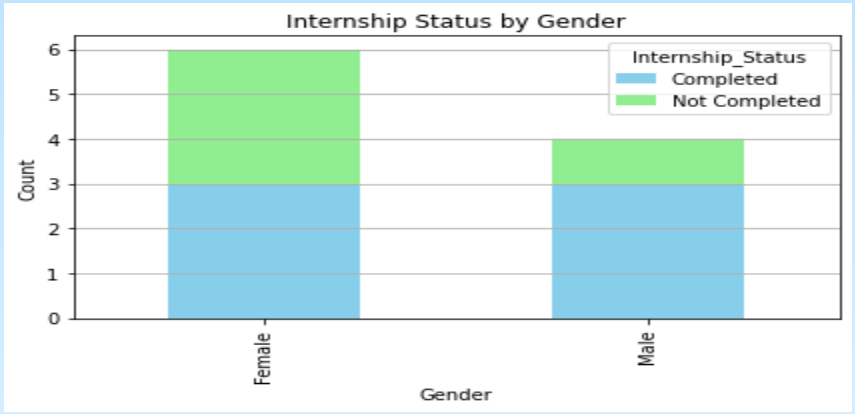

```
import pandas as pd
import matplotlib.pyplot as plt

# Read data from CSV file
data = pd.read_csv('data.csv')

Student_Name = data['Name']
Student_grades = data['Project_Grade']

# Creating a bar plot
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
plt.bar(Student_Name, Student_grades, color='skyblue')

# Adding labels and title
plt.xlabel('Student Name')
plt.ylabel('Project Grade')
plt.title('Project Grades by Student')

# Rotating x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

# Displaying the plot
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



GJU
الجامعة الألمانية الأردنية
German Jordanian University

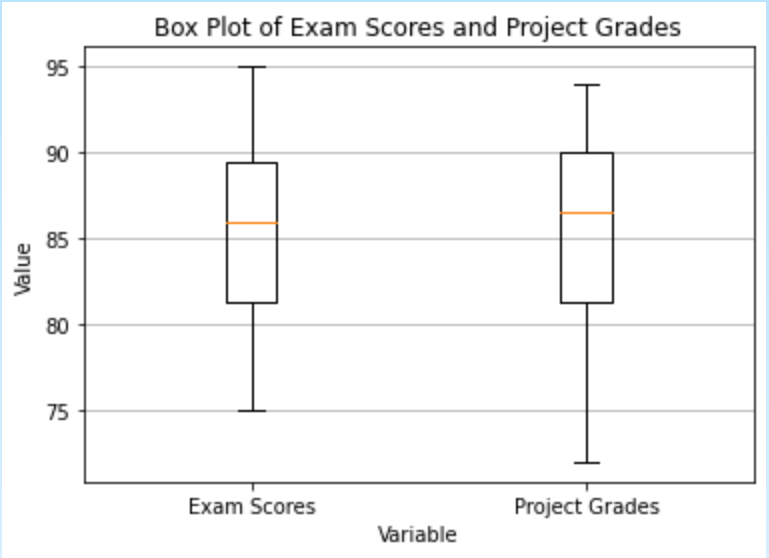
Stacked Bar Plot


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

# Read data from CSV file
data = pd.read_csv('data.csv')

# Group data by gender and internship status and calculate the counts
gender_internship_counts = data.groupby(['Gender', 'Internship_Status']).size().unstack()

# Creating a stacked bar plot
gender_internship_counts.plot(kind='bar', stacked=True, color=['skyblue', 'lightgreen'])

# Adding labels and title
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Internship Status by Gender')

# Displaying the plot
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



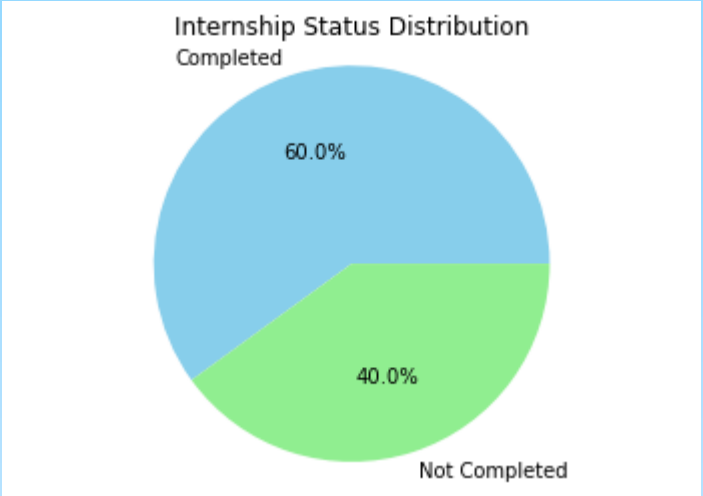
Box Plot


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

# Read data from CSV file
data = pd.read_csv('data.csv')

# Extracting data for box plot
exam_scores = data['Exam_Score']
project_grades = data['Project_Grade']

# Creating a box plot
plt.boxplot([exam_scores, project_grades], labels=['Exam Scores', 'Project Grades'])

# Adding labels and title
plt.xlabel('Variable')
plt.ylabel('Value')
plt.title('Box Plot of Exam Scores and Project Grades')

# Displaying the plot
plt.grid(axis='y')
plt.show()
```



GJU
الجامعة الألمانية الأردنية
German Jordanian University

Pie Chart




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

# Read data from CSV file
data = pd.read_csv('data.csv')

# Count the frequency of each internship status category
internship_counts = data['Internship_Status'].value_counts()

# Creating a pie chart
plt.pie(internship_counts, labels=internship_counts.index, autopct='%1.1f%%',
colors=['skyblue', 'lightgreen'])

# Adding title
plt.title('Internship Status Distribution')

# Displaying the plot
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```

Descriptive Statistics

Descriptive statistics are essential tools used to summarize and describe the main features of a dataset. They provide a concise summary that helps in understanding the characteristics of the data. Here, we'll introduce some common descriptive statistics measures: mean, median, mode, and variance.

1. Mean: The mean, also known as the average, is the sum of all values in a dataset divided by the total number of values. It represents the central tendency of the data. Mathematically, it can be expressed as:

$$\text{Mean} = \frac{\sum_{i=1}^{n} x_i}{n}$$

where (X_i) represents each individual value in the dataset, and (n) represents the total number of values.

2. Median: The median is the middle value of a dataset when it is ordered in ascending or descending order. It divides the dataset into two equal halves. If there is an even number of values, the median is the average of the two middle values.

3. Mode: The mode is the value that appears most frequently in a dataset. A dataset may have one mode (unimodal), two modes (bimodal), or more than two modes (multimodal), or it may have no mode if all values occur with the same frequency.

4. Variance: Variance measures the spread or dispersion of the data points around the mean. It is calculated by taking the average of the squared differences between each data point and the mean. Mathematically, it can be expressed as:

$$\text{Variance} = \frac{\sum_{i=1}^{n} (xi - \bar{X})}{n}$$

where $\bar{X}$ represents the mean of the dataset, (X_i) represents each individual value in the dataset, and (n) represents the total number of values.

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Descriptive Statistics In Python

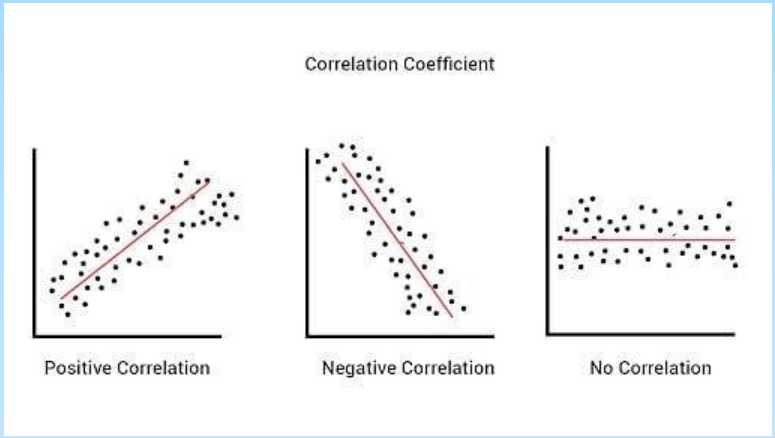

```
import pandas as pd
from scipy import stats

# Read data from CSV file
data = pd.read_csv('data.csv')

# Extract the column containing the data for which you want to calculate statistics
exam_scores = data['Exam_Score']
# Calculate mean
mean = exam_scores.mean()

# Calculate median
median = exam_scores.median()

# Calculate variance
variance = exam_scores.var()

# Calculate mode
# Mode returns a Series, so we use iloc[0] to get the first value
mode = exam_scores.mode().iloc[0]

print("Mean:", mean)
print("Median:", median)
print("Variance:", variance)
print("Mode:", mode)
```

```
Mean: 85.5
Median: 86.0
Variance: 39.833333333333336
Mode: 85
```

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Correlation Analysis

"In data analytics, correlation refers to the statistical relationship between two or more variables. It measures the degree to which the variables are associated or change together. Correlation helps in understanding the strength and direction of the relationship between variables."



In terms of market research this means that, correlation analysis is used to analyse quantitative data gathered from research methods such as surveys and polls, to identify whether there is any significant connections, patterns, or trends between the two.

GJU
الجامعة الألمانية الأردنية
German Jordanian University

Correlation Analysis


```
import pandas as pd
import scipy.stats
data = pd.DataFrame({'A':[1,2,3,4,5], 'B':[2,4,6,8,10]})
correlation_matrix = data.corr()
# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)

data = pd.DataFrame({'A':[1,2,3,4,5], 'B':[10,8,6,4,2]})
correlation_matrix = data.corr()
# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)

data = pd.DataFrame({'A':[1,2,3,4,5], 'B':[100,-80,16,0,2]})
correlation_matrix = data.corr()
# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)

data = pd.DataFrame({'A':[1,2,3,4,5], 'B':[100,200,300,200,100]})
correlation_matrix = data.corr()

# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
     A    B
A  1.0  1.0
B  1.0  1.0
```

```
Correlation Matrix:
     A    B
A  1.0 -1.0
B -1.0  1.0
```

```
Correlation Matrix:
          A         B
A  1.000000 -0.286693
B -0.286693  1.000000
```

```
Correlation Matrix:
     A    B
A  1.0  0.0
B  0.0  1.0
```