Assignment -2 Key

1. Given a dataset containing information about sales transactions, write Python code using Pandas to perform the following tasks:

a. Load the dataset into a Pandas DataFrame.

b. Display the first 5 rows of the DataFrame.

c. Calculate the total sales amount.

Given a dataset containing information about sales transactions, write Python code using Pandas to perform the following tasks:

1) Load the dataset into a Pandas DataFrame.

Python

Copy code

```
import pandas as pd

# Assuming the dataset is named 'sales_data.csv'
sales_df = pd.read_csv('sales_data.csv')
```

2) Display the first 5 rows of the DataFrame.

python

Copy code

```
print(sales_df.head())
```

3) Calculate the total sales amount.

```
python
Copy code
total_sales = sales_df['Sales_Amount'].sum()
print("Total sales amount:", total_sales)
```

2. Explain the anatomy of a Matplotlib plot and discuss its key components. Then, demonstrate how to customize the following aspects of a Matplotlib plot using Python code:

a) Change the color and style of the plot lines.

b) Add a title, labels to the x-axis and y-axis, and a legend to the plot.

a) Change the color and style of the plot lines.

```
Python
Copy code
import matplotlib.pyplot as plt

# Example data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Plot with blue dashed lines
plt.plot(x, y, color='blue', linestyle='--')
plt.show()
```

b) Add a title, labels to the x-axis and y-axis, and a legend to the plot.

Python

Copy code

```
plt.plot(x, y, label='Example Data')
plt.title('Example Plot')
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.legend()
plt.show()
```

3. Using a dataset of your choice (e.g., iris dataset), perform the following tasks:

a) Load the dataset into a Pandas DataFrame.

b) Explore the dataset to understand its structure and contents.

a) Load the dataset into a Pandas DataFrame.

Python

Copy code

```
from sklearn.datasets import load_iris

iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
```

b) Explore the dataset to understand its structure and contents.

Python

Copy code

```
print(iris_df.head())
print(iris_df.info())
print(iris_df.describe())
```

4. Write Python code to create a complex visualization using Matplotlib, incorporating the following elements:

a. Multiple subplots are arranged in a grid layout.

b. Different types of plots (e.g., line plot, bar plot, scatter plot) within each subplot.

a) Multiple subplots arranged in a grid layout.

Python

Copy code

```
fig, axs = plt.subplots(2, 2)  # 2x2 grid layout

# Example data
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [3, 6, 9, 12, 15]

# Plotting on each subplot
axs[0, 0].plot(x, y1)
axs[0, 1].bar(x, y2)
axs[1, 0].scatter(x, y1)
axs[1, 1].plot(x, y2)
```

```
plt.show()
```

b) Different types of plots (e.g., line plot, bar plot, scatter plot) within each subplot. (Already demonstrated above)

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

# Load the Iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target

# Create subplots with different types of plots
fig, axs = plt.subplots(2, 2, figsize=(12, 10))

# Plot 1: Line plot of sepal length
axs[0, 0].plot(iris_df['sepal length (cm)'], color='blue')
axs[0, 0].set_title('Sepal Length')
axs[0, 0].set_xlabel('Index')
axs[0, 0].set_ylabel('Sepal Length (cm)')

# Plot 2: Bar plot of sepal width
axs[0, 1].bar(iris_df.index, iris_df['sepal width (cm)'], color='green')
axs[0, 1].set_title('Sepal Width')
axs[0, 1].set_xlabel('Index')
axs[0, 1].set_ylabel('Sepal Width (cm)')
```

```
# Plot 3: Scatter plot of petal length vs. petal width
colors = ['red', 'green', 'blue']
for i, species in enumerate(iris.target_names):
species_df = iris_df[iris_df['species'] == i]
axs[1, 0].scatter(species_df['petal length (cm)'], species_df['petal width (cm)'], color=colors[i], label=species)
axs[1, 0].set_title('Petal Length vs. Petal Width')
axs[1, 0].set_xlabel('Petal Length (cm)')
axs[1, 0].set_ylabel('Petal Width (cm)')
axs[1, 0].legend()

# Plot 4: Histogram of sepal length by species
for i, species in enumerate(iris.target_names):
species_df = iris_df[iris_df['species'] == i]
axs[1, 1].hist(species_df['sepal length (cm)'], bins=10, alpha=0.5, label=species)
axs[1, 1].set_title('Sepal Length Distribution by Species')
axs[1, 1].set_xlabel('Sepal Length (cm)')
axs[1, 1].set_ylabel('Frequency')
axs[1, 1].legend()

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

5. Provide several visualizations generated from real-world datasets (e.g., housing prices over time, stock market trends, weather patterns), each created using Pandas and Matplotlib. For each visualization:

1. **Describe the dataset and the variables being visualized.**

2. Present the Python code used to generate the visualization.

a) Describe the dataset and the variables being visualized:

- The Iris dataset is a classic dataset in the field of machine learning and statistics. It contains information about 150 iris flowers from three different species: Setosa, Versicolor, and Virginica. There are four features recorded for each flower: sepal length, sepal width, petal length, and petal width.

b) Present the Python code used to generate the visualization:

python

Copy code

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt

# Load the Iris dataset
from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Visualize sepal length distribution
plt.figure(figsize=(8, 6))
plt.hist(iris_df['sepal length (cm)'], bins=20, color='skyblue')
plt.title('Distribution of Sepal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')
```

```
plt.grid(True)
plt.show()

# Visualize petal width vs petal length
plt.figure(figsize=(8, 6))
plt.scatter(iris_df['petal length (cm)'], iris_df['petal width (cm)'], c=iris.target, cmap='viridis')
plt.title('Petal Width vs Petal Length')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.colorbar(label='Species')
plt.grid(True)
plt.show()
```