Data Science SLIPS SOLUTIONS

Q.2 A) Write a Python program to create a Pie plot to get the frequency of the three species of the Iris data (Use iris.csv)

```
import pandas as pd

import matplotlib.pyplot as plt

# Load the Iris dataset

# Ensure you have the iris.csv file in your current working directory

iris_data = pd.read_csv('iris.csv')

# Check the first few rows of the dataset (optional)

print(iris_data.head())


# Get the frequency of each species

species_counts = iris_data['species'].value_counts()


# Create a pie chart

plt.figure(figsize=(8, 6))

plt.pie(species_counts, labels=species_counts.index, autopct='%1.1f%%', startangle=140)

plt.title('Frequency of Iris Species')

plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

B) Write a Python program to view basic statistical details of the data.(Use wineequality-red.csv)

```
import pandas as pd
# Load the Wine Quality dataset
# Ensure you have the winequality-red.csv file in your current working directory
wine_data = pd.read_csv('wine quality-red.csv')
# Check the first few rows of the dataset (optional)
print("First few rows of the dataset:")
print(wine_data.head())
# Display basic statistical details
print("\nBasic Statistical Details:")
stats_summary = wine_data.describe()
print(stats_summary)
```

B) Write a Python program to view basic statistical details of the data.(Use wineequality-red.csv)

```
import pandas as pd
# Load the dataset
data = pd.read_csv('winequality-red.csv')

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

```
# Get basic statistical details of the dataset
statistical_details = data.describe()

# Display the statistical details
print("\nBasic Statistical Details:")
print(statistical_details)
```

Slip2

Q.2 A) Write a Python program for Handling Missing Value. Replace missing value of salary, age column with mean of that column.(Use Data.csv file).

```
import pandas as pd
# Load the dataset
# Ensure you have the Data.csv file in your current working directory
data = pd.read_csv('Data.csv')

# Display the first few rows of the dataset (optional)
print("Original Data:")
print(data.head())

# Check for missing values
print("\nMissing values before handling:")
print(data.isnull().sum())
```

```
# Replace missing values in 'salary' and 'age' columns with their mean
data['salary'].fillna(data['salary'].mean(), inplace=True)
data['age'].fillna(data['age'].mean(), inplace=True)

# Check for missing values after handling
print("\nMissing values after handling:")
print(data.isnull().sum())

# Display the updated DataFrame
print("\nUpdated Data:")
print(data.head())
```

Q.2 B) Write a Python program to generate a line plot of name Vs salary

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

# Load the dataset
# Ensure you have the data.csv file in your current working directory
data = pd.read_csv('data.csv')

# Display the first few rows of the dataset (optional)
print("Data Preview:")
print(data.head())

# Generate a line plot of name vs salary
plt.figure(figsize=(10, 6))
plt.plot(data['name'], data['salary'], marker='o', linestyle='-', color='b')
plt.title('Name vs Salary')
plt.xlabel('Name')
plt.ylabel('Salary')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.grid()
plt.tight_layout()  # Adjust layout to make room for rotated x-axis labels
plt.show()
```

Q.2 C) Download the heights and weights dataset and load the dataset from a given csv file into a dataframe. Print the first, last 10 rows and random 20 rows also display shape of the dataset.

```
import pandas as pd
# Load the heights and weights dataset
# Ensure you have the heights_and_weights.csv file in your current working directory
data = pd.read_csv('heights_and_weights.csv')

# Print the first 10 rows
print("First 10 rows:")
print(data.head(10))

# Print the last 10 rows
print("\nLast 10 rows:")
print(data.tail(10))

# Print 20 random rows
print("\nRandom 20 rows:")
print(data.sample(n=20))

# Display the shape of the dataset
print("\nShape of the dataset:")
print(data.shape)
```

Slip 3

Q.2 A) Write a Python program to create box plots to see how each feature i.e. Sepal Length, Sepal Width, Petal Length, Petal Width are distributed across the three species. (Use iris.csv dataset)

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

# Load the Iris dataset
# Make sure you have 'iris.csv' in the same directory or provide the full path
iris_data = pd.read_csv('iris.csv')

# Display the first few rows of the dataset to understand its structure
print(iris_data.head())

# Set the aesthetic style of the plots
sns.set(style="whitegrid")

# Create a box plot for each feature across the species
features = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
species = iris_data['Species'].unique()

for feature in features:
    plt.figure(fig size=(10, 6))
    sns.boxplot(x='Species', y=feature, data=iris_data)
    plt .title(f' Box plot of {feature} across Iris species')
    plt .xlabel('Species')
    plt.ylabel(feature)
    plt.show()
```

Q.2 B) Write a Python program to view basic statistical details of the data (Use Heights and Weights Dataset)

```
import pandas as pd
# Load the Heights and Weights dataset
# Ensure you have the heights_and_weights.csv file in your current working directory
data = pd.read_csv('heights_and_weights.csv')

# Display the first few rows of the dataset (optional)
print("Data Preview:")
print(data.head())

# Display basic statistical details
stats_summary = data.describe()
print("\nBasic Statistical Details:")
print(stats_summary)
```

Slip 4

Q.2 A) Generate a random array of 50 integers and display them using a line chart, scatter plot, histogram and box plot. Apply appropriate color, labels and styling options.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Generate a random array of 50 integers between 1 and 100
random_data = np.random.randint(1, 101, size=50)

# Set the style of seaborn
sns.set(style='whitegrid')

# Create a figure with subplots
plt.figure(figsize=(14, 10))

# Line Chart
plt.subplot(2, 2, 1)
plt.plot(random_data, color='blue', marker='o', linestyle='-', linewidth=2)
plt.title('Line Chart of Random Integers')
plt.xlabel('Index')
plt.ylabel('Value')
plt.grid()
```

```
# Scatter Plot
plt.subplot(2, 2, 2)
plt.scatter(range(len(random_data)), random_data, color='orange')
plt.title('Scatter Plot of Random Integers')
plt.xlabel('Index')
plt.ylabel('Value')
plt.grid()

# Histogram
plt.subplot(2, 2, 3)
plt.hist(random_data, bins=10, color='green', edgecolor='black')
plt.title('Histogram of Random Integers')
plt.xlabel('Value')
plt.ylabel('Frequency')

# Box Plot
plt.subplot(2, 2, 4)
sns.boxplot(data=random_data, color='purple')
plt.title('Box Plot of Random Integers')
plt.ylabel('Value')

# Adjust layout
plt.tight_layout()
plt.show()
```

Q.2 B) Write a Python program to print the shape, number of rows-columns, data types, feature names and the description of the data(Use User_Data.csv)

```
import pandas as pd

# Load the User Data dataset
# Make sure you have 'User_Data.csv' in the same directory or provide the full path
data = pd.read_csv('User_Data.csv')

# Print the shape of the DataFrame
print("Shape of the dataset (rows, columns):", data.shape)

# Print the number of rows and columns
num_rows, num_columns = data.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_columns)

# Print the data types of each feature
print("\nData types of each feature:")
print(data.dtypes)
```

```
# Print the feature names
print("\nFeature names:")
print(data.columns.tolist())

# Print the description of the data
print("\nDescription of the dataset:")
print(data.describe(include='all'))  # include='all' to get stats for categorical features as well
```

Slip 5

Q.2 A) Generate a random array of 50 integers and display them using a line chart, scatter plot, histogram and box plot. Apply appropriate color, labels and styling options.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set a seed for reproducibility
np.random.seed(0)

# Generate a random array of 50 integers between 1 and 100
data = np.random.randint(1, 101, size=50)

# Set the style for seaborn
sns.set(style="whitegrid")
```

```
# Create a figure with multiple subplots
fig, axs = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Random Integer Array Visualizations', fontsize=16)

# Line Chart
axs[0, 0].plot(data, color='blue', marker='o', linestyle='-', linewidth=2, markersize=5)
axs[0, 0].set_title('Line Chart')
axs[0, 0].set_xlabel('Index')
axs[0, 0].set_ylabel('Value')
axs[0, 0].grid(True)

# Scatter Plot
axs[0, 1].scatter(range(len(data)), data, color='orange', s=100, alpha=0.7)
axs[0, 1].set_title('Scatter Plot')
axs[0, 1].set_xlabel('Index')
axs[0, 1].set_ylabel('Value')
axs[0, 1].grid(True)

# Histogram
axs[1, 0].hist(data, bins=10, color='green', edgecolor='black', alpha=0.7)
axs[1, 0].set_title('Histogram')
axs[1, 0].set_xlabel('Value')
```

```
axs[1, 0].set_ylabel('Frequency')

# Box Plot
sns.boxplot(data=data, ax=axs[1, 1], color='purple')
axs[1, 1].set_title('Box Plot')
axs[1, 1].set_ylabel('Value')

# Adjust layout
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Leave space for the main title
plt.show()
```

Q.2 B) Write a Python program to print the shape, number of rows-columns, data types, feature names and the description of the data(Use User_Data.csv)

```
import pandas as pd
# Load the User Data dataset
# Make sure you have 'User_Data.csv' in the same directory or provide the full path
data = pd.read_csv('User_Data.csv')

# Print the shape of the DataFrame
print("Shape of the dataset (rows, columns):", data.shape)

# Print the number of rows and columns
num_rows, num_columns = data.shape
```

```
print("Number of rows:", num_rows)
print("Number of columns:", num_columns)

# Print the data types of each feature
print("\nData types of each feature:")
print(data.dtypes)

# Print the feature names
print("\nFeature names:")
print(data.columns.tolist())

# Print the description of the data
print("\nDescription of the dataset:")
print(data.describe(include='all'))  # include='all' to get stats for categorical features as well
```

Slip 6

Q.2 A) Write a Python program for Handling Missing Value. Replace missing value of salary, age column with mean of that column.(Use Data.csv file).

```
import pandas as pd
# Load the dataset
# Make sure you have 'Data.csv' in the same directory or provide the full path
data = pd.read_csv('Data.csv')

# Display the first few rows of the dataset to understand its structure
print("Original Data:")
print(data.head())

# Check for missing values
print("\nMissing values before handling:")
print(data.isnull().sum())

# Replace missing values in 'salary' and 'age' with their respective means
data['salary'].fillna(data['salary'].mean(), inplace=True)
data['age'].fillna(data['age'].mean(), inplace=True)

# Check for missing values again to confirm replacement
print("\nMissing values after handling:")
print(data.isnull().sum())

# Display the updated dataset
print("\nUpdated Data:")
print(data.head())
```

```
# Optionally, save the cleaned data to a new CSV file
data.to_csv('Cleaned_Data.csv', index=False)
```

Q.2 B) Write a Python program to generate a line plot of name Vs salary

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

# Load the cleaned dataset
data = pd.read_csv('Cleaned_Data.csv')

# Display the data (optional)
print("Cleaned Data:")
print(data)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(data['name'], data['salary'], marker='o', linestyle='-', color='b')

# Adding titles and labels
plt.title('Name vs Salary')
plt.xlabel('Name')
plt.ylabel('Salary')
plt.xticks(rotation=45)  # Rotate x labels for better readability
plt.grid()
```

```
# Show the plot
plt.tight_layout()
plt.show()
```

Q.2 C) Download the heights and weights dataset and load the dataset from a given csv file into a dataframe. Print the first, last 10 rows and random 20 rows also display shape of the dataset.

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

# Sample data: Create a DataFrame (or you can load from a CSV file)
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Salary': [50000, 60000, 55000, 70000, 65000]
}

df = pd.DataFrame(data)

# Display the DataFrame
print("Data:")
print(df)

# Generate a line plot
```

```
plt.figure(figsize=(10, 5))
plt.plot(df['Name'], df['Salary'], marker='o', linestyle='-', color='blue')
plt.title('Name vs Salary')
plt.xlabel('Name')
plt.ylabel('Salary')
plt.grid(True)

# Show the plot
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout
plt.show()
```

Slip 7

Q.2) Write a Python program to perform the following tasks :

a. Apply OneHot coding on Country column.

b. Apply Label encoding on purchased column (Data.csv have two categorical column the country column, and the purchased column).

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
# Load the dataset
# Make sure you have 'Data.csv' in the same directory or provide the full path
data = pd.read_csv('Data.csv')

# Display the original DataFrame
print("Original Data:")
print(data.head())

# a. Apply OneHot encoding on the Country column
data_onehot = pd.get_dummies(data, columns=['Country'], drop_first=True)

# Display DataFrame after OneHot encoding
print("\nData after OneHot encoding on Country column:")
print(data_onehot.head())

# b. Apply Label encoding on the Purchased column
label_encoder = LabelEncoder()
data_onehot['Purchased'] = label_encoder.fit_transform(data_onehot['Purchased'])

# Display DataFrame after Label encoding
print("\nData after Label encoding on Purchased column:")
print(data_onehot.head())
```

```
# Optionally, save the transformed data to a new CSV file
data_onehot.to_csv('Transformed_Data.csv', index=False)
```

Slip 8

Q.2) Write a program in python to perform following task : Standardizing Data (transform them into a standard Gaussian distribution with a mean of 0 and a standard deviation of 1) (Use winequality-red.csv)

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = pd.read_csv('winequality-red.csv')

# Display the initial data (optional)
print("Initial Data:")
print(data.head())

# Initialize the StandardScaler
scaler = StandardScaler()
```

```
# Standardize the features (excluding the target column if applicable)
# Assuming the last column is the target (quality), we standardize all but the last column
features = data.iloc[:, :-1]  # Select all columns except the last
standardized_features = scaler.fit_transform(features)

# Create a DataFrame for the standardized data
standardized_data = pd.DataFrame(standardized_features, columns=features.columns)

# Display the standardized data (optional)
print("\nStandardized Data:")
print(standardized_data.head())

# Optionally save the standardized data to a new CSV file
standardized_data.to_csv('standardized_winequality_red.csv', index=False)
```

Slip 9

Q.2 A) Generate a random array of 50 integers and display them using a line chart, scatter plot. Apply appropriate color, labels and styling options.

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

# Generate a random array of 50 integers between 1 and 100
```

```
random_integers = np.random.randint(1, 101, size=50)

# Create an array for the x-axis (indices)
x = np.arange(1, 51)

# Create a figure and axis
plt.figure(figsize=(12, 6))

# Line Chart
plt.subplot(1, 2, 1)
plt.plot(x, random_integers, marker='o', linestyle='-', color='b', markersize=5)
plt.title('Line Chart of Random Integers')
plt.xlabel('Index')
plt.ylabel('Random Integer Value')
plt.grid(True)

# Scatter Plot
plt.subplot(1, 2, 2)
plt.scatter(x, random_integers, color='r', s=50)
plt.title('Scatter Plot of Random Integers')
plt.xlabel('Index')
plt.ylabel('Random Integer Value')
plt.grid(True)
```

```
# Adjust layout and show the plots
plt.tight_layout()
plt.show()
```

Q.2 B) Create two lists, one representing subject names and the other representing marks obtained in those subjects. Display the data in a pie chart.

```
import matplotlib.pyplot as plt

# Define the subject names and their corresponding marks
subjects = ['Math', 'Science', 'English', 'History', 'Art']
marks = [85, 90, 78, 88, 92]

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(marks, labels=subjects, autopct='%1.1f%%', startangle=140,
colors=plt.cm.Paired.colors)

# Add a title
plt.title('Marks Distribution in Subjects')

# Show the pie chart
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular
plt.show()
```

Q.2 C) Write a program in python to perform following task (Use winequality-red.csv) Import Dataset and do the followings: a) Describing the dataset b) Shape of the dataset c) Display first 3 rows from dataset

```
import pandas as pd

# Load the dataset
data = pd.read_csv('winequality-red.csv')

# a) Describing the dataset
description = data.describe()
print("Description of the dataset:")
print(description)

# b) Shape of the dataset
shape = data.shape
print("\nShape of the dataset:")
print(shape)

# c) Display the first 3 rows from the dataset
first_three_rows = data.head(3)
print("\nFirst 3 rows of the dataset:")
print(first_three_rows)
```

Slip 10

Q.2 A) Write a python program to Display column-wise mean, and median for SOCR HeightWeight dataset.

```
import pandas as pd
# Load the SOCR HeightWeight dataset
data = pd.read_csv('HeightWeight.csv')

# Display the first few rows of the dataset (optional)
print("First few rows of the dataset:")
print(data.head())

# Calculate column-wise mean and median
mean_values = data.mean()
median_values = data.median()

# Display the results
print("\nColumn-wise Mean:")
print(mean_values)

print("\nColumn-wise Median:")
print(median_values)
```

Q.2 B) Write a python program to compute sum of Manhattan distance between all pairs of points.

Slip 11

Q.2 A) Write a Python program to create a Pie plot to get the frequency of the three species of the Iris data (Use iris.csv)

```
import pandas as pd

import matplotlib.pyplot as plt


# Load the Iris dataset

data = pd.read_csv('iris.csv')


# Display the first few rows of the dataset (optional)

print("First few rows of the dataset:")

print(data.head())


# Count the frequency of each species

species_counts = data['species'].value_counts()


# Create a pie plot

plt.figure(figsize=(8, 8))

plt.pie(species_counts, labels=species_counts.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors)
```

```
# Add a title
plt.title('Frequency of Iris Species')

# Show the pie chart
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular
plt.show()
```

Slip 12

Q.2 B) Write a Python program to create data frame containing column name, salary, department add 10 rows with some missing and duplicate values to the data frame. Also drop all null and empty values. Print the modified data frame.

```
import pandas as pd
import numpy as np

# Create a DataFrame with sample data
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Alice',
np.nan, 'Hank'],
    'Salary': [70000, 80000, np.nan, 60000, 90000, 50000, np.nan, 70000,
75000, 60000],
    'Department': ['HR', 'Finance', 'IT', 'IT', 'Finance', 'HR', 'IT', np.nan,
'Finance', 'HR']
}

df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Drop all null and empty values
df_cleaned = df.dropna()

# Print the modified DataFrame
print("\nModified DataFrame after dropping null values:")
print(df_cleaned)
```

Slip 13

Q.2 A) Write a Python program to create a graph to find relationship between the petal length and petal width.(Use iris.csv dataset)

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

# Load the Iris dataset
```

```
data = pd.read_csv('iris.csv')

# Display the first few rows of the dataset (optional)
print("First few rows of the dataset:")
print(data.head())

# Create a scatter plot for petal length vs petal width
plt.figure(figsize=(10, 6))
plt.scatter(data['petal_length'], data['petal_width'], c=data['species'].astype('category').cat.codes, cmap='viridis', alpha=0.7)

# Adding titles and labels
plt.title('Relationship between Petal Length and Petal Width')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')

# Adding a color bar to indicate species
plt.colorbar(ticks=[0, 1, 2], label='Species', format='%d')
plt.clim(-0.5, 2.5)  # Adjust the color limits

# Show the plot
plt.grid()
plt.show()
```

Q.2 B) Write a Python program to find the maximum and minimum value of a given flattened array.

```
import numpy as np


def find_max_min(arr):
    """
    Find the maximum and minimum values of a given flattened array.

    Parameters:
    arr (np.ndarray): Input array.

    Returns:
    tuple: Maximum and minimum values in the array.
    """
    # Ensure arr is a numpy array
    arr = np.asarray(arr)

    # Find maximum and minimum values
    max_value = np.max(arr)
    min_value = np.min(arr)

    return max_value, min_value

# Example usage
```

```
if __name__ == "__main__":
    # Create a flattened array
    flattened_array = np.array([3, 5, 1, 8, 2, 9, 4, 6, 7, 0])

    # Find maximum and minimum values
    max_val, min_val = find_max_min(flattened_array)

    print("Maximum value:", max_val)
    print("Minimum value:", min_val)
```

Slip 14

Q. 2 A) Write a Python NumPy program to compute the weighted average along the specified axis of a given flattened array.

```
import numpy as np
def weighted_average(arr, weights, axis=None):
    """
    Compute the weighted average along the specified axis of a given flattened array.

    Parameters:
    arr (np.ndarray): Input array.
    weights (np.ndarray): Weights for each value in arr.
    axis (int, optional): Axis along which the weighted average is computed.
```

```
                           If None, the weighted average is computed over the
flattened array.


    Returns:
    np.ndarray: Weighted average of the array along the specified axis.
    """
    # Ensure arr and weights are numpy arrays
    arr = np.asarray(arr)
    weights = np.asarray(weights)

    # Check if the shape of weights is compatible with arr
    if weights.shape != arr.shape:
        raise ValueError("Weights must have the same shape as arr.")

    # Compute the weighted average
    weighted_avg = np.sum(arr * weights, axis=axis) / np.sum(weights,
axis=axis)

    return weighted_avg

# Example usage
if __name__ == "__main__":
    # Flattened array
    arr = np.array([1, 2, 3, 4, 5])
    # Corresponding weights
    weights = np.array([0.1, 0.2, 0.3, 0.4, 0.5])

    # Compute weighted average
    result = weighted_average(arr, weights)

    print("Weighted average:", result)
```

Q. 2 B) Write a Python program to view basic statistical details of the data (Use advertising.csv)

```
import pandas as pd

# Load the advertising dataset
file_path = 'advertising.csv'  # Update this path as necessary
data = pd.read_csv(file_path)

# Display basic statistical details
def display_statistics(data):
    print("Basic Statistical Details:")
    print(data.describe())
    print("\nInformation about the DataFrame:")
    print(data.info())
    print("\nFirst 5 rows of the dataset:")
    print(data.head())
```

```
if __name__ == "__main__":
    display_statistics(data)
```

Slip 15

Q.2 A) Generate a random array of 50 integers and display them using a line chart, scatter plot, histogram and box plot. Apply appropriate color, labels and styling options.

```
import numpy as np

import matplotlib.pyplot as plt


# Generate a random array of 50 integers between 1 and 100

random_integers = np.random.randint(1, 101, size=50)


# Set up the plotting area

plt.figure(figsize=(14, 10))


# Line Chart

plt.subplot(2, 2, 1)

plt.plot(random_integers, color='blue', marker='o', linestyle='-', linewidth=2, markersize=5)

plt.title('Line Chart of Random Integers', fontsize=16)

plt.xlabel('Index', fontsize=14)

plt.ylabel('Value', fontsize=14)

plt.grid()
```

```
# Scatter Plot
plt.subplot(2, 2, 2)
plt.scatter(range(len(random_integers)), random_integers, color='orange', s=50)
plt.title('Scatter Plot of Random Integers', fontsize=16)
plt.xlabel('Index', fontsize=14)
plt.ylabel('Value', fontsize=14)
plt.grid()

# Histogram
plt.subplot(2, 2, 3)
plt.hist(random_integers, bins=10, color='green', alpha=0.7, edgecolor='black')
plt.title('Histogram of Random Integers', fontsize=16)
plt.xlabel('Value', fontsize=14)
plt.ylabel('Frequency', fontsize=14)

# Box Plot
plt.subplot(2, 2, 4)
plt.boxplot(random_integers, patch_artist=True, boxprops=dict(facecolor='lightblue', color='blue'),
            medianprops=dict(color='red'))
plt.title('Box Plot of Random Integers', fontsize=16)
plt.ylabel('Value', fontsize=14)
```

```
# Adjust layout
plt.tight_layout()
plt.show()
```

Q.2 B) Create two lists, one representing subject names and the other representing marks obtained in those subjects. Display the data in a pie chart.

```
import matplotlib.pyplot as plt

# Define the subject names and corresponding marks
subjects = ['Mathematics', 'Science', 'English', 'History', 'Art']
marks = [85, 92, 78, 88, 95]

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(marks, labels=subjects, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors)
plt.title('Marks Distribution by Subject', fontsize=16)
plt.axis('equal')  # Equal aspect ratio ensures that pie chart is a circle.

# Display the pie chart
plt.show()
```

Slip 16

Q.2 A) Write a python program to create two lists, one representing subject names and the other representing marks obtained in those subjects. Display the data in a pie chart and bar chart.

```
import matplotlib.pyplot as plt

# Define the subject names and corresponding marks
subjects = ['Mathematics', 'Science', 'English', 'History', 'Art']
marks = [85, 92, 78, 88, 95]

# Create a pie chart
plt.figure(figsize=(12, 6))

# Pie Chart
plt.subplot(1, 2, 1)  # 1 row, 2 columns, 1st subplot
plt.pie(marks, labels=subjects, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors)
plt.title('Marks Distribution by Subject', fontsize=16)
plt.axis('equal')  # Equal aspect ratio ensures that pie chart is a circle.

# Bar Chart
plt.subplot(1, 2, 2)  # 1 row, 2 columns, 2nd subplot
plt.bar(subjects, marks, color='skyblue')
plt.title('Marks Obtained in Subjects', fontsize=16)
plt.xlabel('Subjects', fontsize=14)
plt.ylabel('Marks', fontsize=14)
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability


# Adjust layout
plt.tight_layout()


# Display both charts
plt.show()
```

Q.2 B) Write a python program to create a data frame for students' information such as name, graduation percentage and age. Display average age of students, average of graduation percentage.

```
import pandas as pd


# Create a DataFrame with students' information
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
    'Graduation Percentage': [85, 92, 78, 88, 95],
    'Age': [20, 21, 19, 22, 20]
}


students_df = pd.DataFrame(data)


# Calculate average age and average graduation percentage
average_age = students_df['Age'].mean()
average_graduation_percentage = students_df['Graduation Percentage'].mean()
```

```
# Display the DataFrame and the averages
print("Students Information DataFrame:")
print(students_df)
print("\nAverage Age of Students:", average_age)
print("Average Graduation Percentage:", average_graduation_percentage)
```

Slip 17

Q.2 A) Write a Python program to draw scatter plots to compare two features of the iris dataset

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

# Load the iris dataset
iris = sns.load_dataset('iris')

# Display the first few rows of the dataset
print(iris.head())

# Set up the figure
plt.figure(figsize=(12, 6))

# Scatter plot comparing Sepal Length and Sepal Width
```

```
plt.subplot(1, 2, 1)
sns.scatterplot(data=iris, x='sepal_length', y='sepal_width', hue='species', style='species', s=100)
plt.title('Sepal Length vs Sepal Width', fontsize=16)
plt.xlabel('Sepal Length (cm)', fontsize=14)
plt.ylabel('Sepal Width (cm)', fontsize=14)

# Scatter plot comparing Petal Length and Petal Width
plt.subplot(1, 2, 2)
sns.scatterplot(data=iris, x='petal_length', y='petal_width', hue='species', style='species', s=100)
plt.title('Petal Length vs Petal Width', fontsize=16)
plt.xlabel('Petal Length (cm)', fontsize=14)
plt.ylabel('Petal Width (cm)', fontsize=14)

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

Q.2 B) Write a Python program to create a data frame containing columns name, age , salary, department . Add 10 rows to the data frame. View the data frame.

```
import pandas as pd
```

```
# Create a DataFrame with columns: name, age, salary, and department
data = {
    'Name': [
        'Alice', 'Bob', 'Charlie', 'David', 'Eva',
        'Frank', 'Grace', 'Hannah', 'Ian', 'Judy'
    ],
    'Age': [28, 34, 29, 42, 35, 30, 25, 32, 31, 38],
    'Salary': [
        70000, 80000, 75000, 120000, 95000,
        60000, 65000, 72000, 80000, 85000
    ],
    'Department': [
        'HR', 'IT', 'Finance', 'Marketing', 'Sales',
        'IT', 'Finance', 'HR', 'Marketing', 'Sales'
    ]
}

# Create the DataFrame
employees_df = pd.DataFrame(data)

# View the DataFrame
print("Employees DataFrame:")
print(employees_df)
```

Slip 18

Q.2 A) Write a Python program to create box plots to see how each feature i.e. Sepal Length, Sepal Width, Petal Length, Petal Width are distributed across the three species. (Use iris.csv dataset)

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

# Load the iris dataset from the CSV file
file_path = 'iris.csv'  # Update this path to where your iris.csv file is located
iris = pd.read_csv(file_path)

# Display the first few rows of the dataset
print(iris.head())

# Set up the plotting area
plt.figure(figsize=(12, 10))

# Box plot for Sepal Length
plt.subplot(2, 2, 1)
sns.boxplot(x='species', y='sepal_length', data=iris)
plt.title('Sepal Length Distribution by Species', fontsize=16)
```

```
# Box plot for Sepal Width
plt.subplot(2, 2, 2)
sns.boxplot(x='species', y='sepal_width', data=iris)
plt.title('Sepal Width Distribution by Species', fontsize=16)

# Box plot for Petal Length
plt.subplot(2, 2, 3)
sns.boxplot(x='species', y='petal_length', data=iris)
plt.title('Petal Length Distribution by Species', fontsize=16)

# Box plot for Petal Width
plt.subplot(2, 2, 4)
sns.boxplot(x='species', y='petal_width', data=iris)
plt.title('Petal Width Distribution by Species', fontsize=16)

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

Q.2 B) Use the heights and weights dataset and load the dataset from a given csv file into a dataframe. Print the first, last 5 rows and random 10 row

```
import pandas as pd
```

```
# Load the dataset from a CSV file
file_path = 'heights_weights.csv'  # Update this path to your CSV file location
data = pd.read_csv(file_path)

# Print the first 5 rows
print("First 5 rows:")
print(data.head())

# Print the last 5 rows
print("\nLast 5 rows:")
print(data.tail())

# Print 10 random rows
print("\nRandom 10 rows:")
print(data.sample(n=10))
```

Slip 19

Q.2) Write a Python program 1. To create a dataframe containing columns name, age and percentage. Add 10 rows to the dataframe. View the dataframe.

```
import pandas as pd

# Create a DataFrame with columns: name, age, and percentage
```

```
data = {
    'Name': [
        'Alice', 'Bob', 'Charlie', 'David', 'Eva',
        'Frank', 'Grace', 'Hannah', 'Ian', 'Judy'
    ],
    'Age': [20, 21, 19, 22, 20, 23, 24, 25, 21, 20],
    'Percentage': [85.5, 90.0, 78.5, 88.0, 95.0, 82.5, 76.0, 91.0, 89.5, 84.0]
}

# Create the DataFrame
students_df = pd.DataFrame(data)

# View the DataFrame
print("Students DataFrame:")
print(students_df)
```

2. To print the shape, number of rows-columns, data types, feature names and the description of the data

```
import pandas as pd

# Create a DataFrame with columns: name, age, and percentage
data = {
    'Name': [
        'Alice', 'Bob', 'Charlie', 'David', 'Eva',
        'Frank', 'Grace', 'Hannah', 'Ian', 'Judy'
    ],
    'Age': [20, 21, 19, 22, 20, 23, 24, 25, 21, 20],
    'Percentage': [85.5, 90.0, 78.5, 88.0, 95.0, 82.5, 76.0, 91.0, 89.5, 84.0]
}

# Create the DataFrame
students_df = pd.DataFrame(data)

# Print the shape of the DataFrame
print("Shape of DataFrame:", students_df.shape)

# Print the number of rows and columns
rows, columns = students_df.shape
print(f"Number of Rows: {rows}, Number of Columns: {columns}")

# Print the data types of each column
print("\nData Types:")
print(students_df.dtypes)

# Print the feature names
print("\nFeature Names:")
print(students_df.columns.tolist())
```

```
# Print the description of the data
print("\nDescription of the Data:")
print(students_df.describe())
```

3. To Add 5 rows with duplicate values and missing values. Add a column ‘remarks’ with empty values. Display the data.

```
import pandas as pd
import numpy as np

# Create a DataFrame with columns: name, age, and percentage
data = {
    'Name': [
        'Alice', 'Bob', 'Charlie', 'David', 'Eva',
        'Frank', 'Grace', 'Hannah', 'Ian', 'Judy'
    ],
    'Age': [20, 21, 19, 22, 20, 23, 24, 25, 21, 20],
    'Percentage': [85.5, 90.0, 78.5, 88.0, 95.0, 82.5, 76.0, 91.0, 89.5, 84.0]
}

# Create the DataFrame
students_df = pd.DataFrame(data)

# Add 5 rows with duplicate and missing values
duplicate_rows = pd.DataFrame({
    'Name': ['Alice', 'Bob', 'Charlie', np.nan, 'Eva'],
    'Age': [20, 21, 19, 22, np.nan],
    'Percentage': [85.5, 90.0, np.nan, 88.0, 95.0]
})

# Concatenate the original DataFrame with the new rows
students_df = pd.concat([students_df, duplicate_rows], ignore_index=True)

# Add a new column 'remarks' with empty values
students_df['Remarks'] = '   '

# Display the updated DataFrame
print("Updated Students DataFrame:")
print(students_df)
```

Slip 20

Q.2 A) Generate a random array of 50 integers and display them using a line chart, scatter plot, histogram and box plot. Apply appropriate color, labels and styling options.

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

# Generate a random array of 50 integers between 1 and 100
```

```
random_array = np.random.randint(1, 101, size=50)

# Set up the plotting area
plt.figure(figsize=(12, 10))

# Line Chart
plt.subplot(2, 2, 1)
plt.plot(random_array, color='blue', marker='o', linestyle='-')
plt.title('Line Chart of Random Integers', fontsize=16)
plt.xlabel('Index', fontsize=12)
plt.ylabel('Value', fontsize=12)
plt.grid(True)

# Scatter Plot
plt.subplot(2, 2, 2)
plt.scatter(range(len(random_array)), random_array, color='orange', s=100)
plt.title('Scatter Plot of Random Integers', fontsize=16)
plt.xlabel('Index', fontsize=12)
plt.ylabel('Value', fontsize=12)
plt.grid(True)

# Histogram
plt.subplot(2, 2, 3)
plt.hist(random_array, bins=10, color='green', alpha=0.7, edgecolor='black')
plt.title('Histogram of Random Integers', fontsize=16)
plt.xlabel('Value', fontsize=12)
plt.ylabel('Frequency', fontsize=12)

# Box Plot
plt.subplot(2, 2, 4)
plt.boxplot(random_array, patch_artist=True, boxprops=dict(facecolor='purple', color='black'))
plt.title('Box Plot of Random Integers', fontsize=16)
plt.ylabel('Value', fontsize=12)

# Adjust layout
plt.tight_layout()

# Show the plots
plt.show()
```

Q.2 B) Add two outliers to the above data and display the box plot.

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

```
# Generate a random array of 50 integers between 1 and 100
random_array = np.random.randint(1, 101, size=50)

# Add two outliers
outliers = [200, 250]  # Outlier values
data_with_outliers = np.concatenate((random_array, outliers))

# Set up the plotting area
plt.figure(figsize=(6, 6))

# Box Plot
plt.boxplot(data_with_outliers, patch_artist=True,
boxprops=dict(facecolor='purple', color='black'))
plt.title('Box Plot with Outliers', fontsize=16)
plt.ylabel('Value', fontsize=12)

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

Slip 21

Q.2 A) Import dataset “iris.csv”. Write a Python program to create a Bar plot to get the frequency of the three species of the Iris data.

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

# Load the iris dataset from a CSV file
file_path = 'iris.csv'  # Update this path to where your iris.csv file is located
iris = pd.read_csv(file_path)

# Count the frequency of each species
species_counts = iris['species'].value_counts()

# Create a bar plot for the frequency of the species
plt.figure(figsize=(8, 6))
sns.barplot(x=species_counts.index, y=species_counts.values, palette='viridis')
plt.title('Frequency of Iris Species', fontsize=16)
plt.xlabel('Species', fontsize=14)
plt.ylabel('Frequency', fontsize=14)

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

Q.2 B)Write a Python program to create a histogram of the three species of the Iris data.

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

# Load the iris dataset from a CSV file
file_path = 'iris.csv'  # Update this path to where your iris.csv file is located
iris = pd.read_csv(file_path)

# Set up the plotting area
plt.figure(figsize=(12, 6))

# Create a histogram for each feature colored by species
for feature in ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']:
    plt.subplot(2, 2, ['sepal_length', 'sepal_width', 'petal_length',
'petal_width'].index(feature) + 1)
    sns.histplot(data=iris, x=feature, hue='species', kde=True, bins=10,
palette='viridis', alpha=0.6)
    plt.title(f'Histogram of {feature.capitalize()}', fontsize=16)
    plt.xlabel(feature.capitalize(), fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.grid(True)

# Adjust layout
plt.tight_layout()
```

```
# Show the plots
plt.show()
```

Slip 22

Q.2) Dataset Name: winequality-red.csv Write a program in python to perform following tasks

a. Rescaling: Normalised the dataset using MinMaxScaler class

b. Standardizing Data (transform them into a standard Gaussian distribution with a mean of 0 and a standard deviation of 1)

c. Normalizing Data (rescale each observation to a length of 1 (a unit norm). For this, use the Normalizer class.)

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer

# Load the dataset
file_path = 'winequality-red.csv'  # Update this path to where your winequality-red.csv file is located
wine_data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Original Data:")
print(wine_data.head())
```

```
# a. Rescaling: Normalize the dataset using MinMaxScaler
min_max_scaler = MinMaxScaler()
wine_data_normalized = min_max_scaler.fit_transform(wine_data)

# Convert back to DataFrame
wine_data_normalized_df = pd.DataFrame(wine_data_normalized,
columns=wine_data.columns)
print("\nNormalized Data (MinMaxScaler):")
print(wine_data_normalized_df.head())

# b. Standardizing Data
standard_scaler = StandardScaler()
wine_data_standardized = standard_scaler.fit_transform(wine_data)

# Convert back to DataFrame
wine_data_standardized_df = pd.DataFrame(wine_data_standardized,
columns=wine_data.columns)
print("\nStandardized Data:")
print(wine_data_standardized_df.head())

# c. Normalizing Data
normalizer = Normalizer()
wine_data_normalized_unit = normalizer.fit_transform(wine_data)
```

```
# Convert back to DataFrame
wine_data_normalized_unit_df = pd.DataFrame(wine_data_normalized_unit, columns=wine_data.columns)
print("\nNormalized Data (Unit Norm):")
print(wine_data_normalized_unit_df.head())
```

Slip 23

Q.2) Dataset Name: winequality-red.csv Write a program in python to perform following task

a. Rescaling: Normalised the dataset using MinMaxScaler class

b. Standardizing Data (transform them into a standard Gaussian distribution with a mean of 0 and a standard deviation of 1)

c. Binarizing Data using we use the Binarizer class (Using a binary threshold, it is possible to transform our data by marking the values above it 1 and those equal to or below it, 0)

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Binarizer
# Load the dataset
file_path = 'winequality-red.csv'  # Update this path to where your winequality-red.csv file is located
wine_data = pd.read_csv(file_path)
# Display the first few rows of the dataset
print("Original Data:")
print(wine_data.head())
```

```
# a. Rescaling: Normalize the dataset using MinMaxScaler
min_max_scaler = MinMaxScaler()
wine_data_normalized = min_max_scaler.fit_transform(wine_data)
# Convert back to DataFrame
wine_data_normalized_df = pd.DataFrame(wine_data_normalized, columns=wine_data.columns)
print("\nNormalized Data (MinMaxScaler):")
print(wine_data_normalized_df.head())
# b. Standardizing Data
standard_scaler = StandardScaler()
wine_data_standardized = standard_scaler.fit_transform(wine_data)
# Convert back to DataFrame
wine_data_standardized_df = pd.DataFrame(wine_data_standardized, columns=wine_data.columns)
print("\nStandardized Data:")
print(wine_data_standardized_df.head())
# c. Binarizing Data
binarizer = Binarizer(threshold=0.5)  # Set the threshold for binarization
wine_data_binarized = binarizer.fit_transform(wine_data)
# Convert back to DataFrame
wine_data_binarized_df = pd.DataFrame(wine_data_binarized, columns=wine_data.columns)
print("\nBinarized Data:")
print(wine_data_binarized_df.head())
```

Slip 24

Q.2 A) Import dataset “iris.csv”. Write a Python program to create a Bar plot to get the frequency of the three species of the Iris data.

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

# Load the iris dataset from a CSV file
file_path = 'iris.csv'  # Update this path to where your iris.csv file is located
iris = pd.read_csv(file_path)

# Count the frequency of each species
species_counts = iris['species'].value_counts()

# Create a bar plot for the frequency of the species
plt.figure(figsize=(8, 6))
sns.barplot(x=species_counts.index, y=species_counts.values, palette='viridis')
plt.title('Frequency of Iris Species', fontsize=16)
plt.xlabel('Species', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
```

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

Q.2 B) Write a Python program to create a histogram of the three species of the Iris data.

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

# Load the Iris dataset (You can use Read csv command)
url = https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
iris_data = pd.read_csv(url, header=None, names=column_names)

# Set the aesthetic style of the plots
sns.set(style="whitegrid")

# Create a histogram
plt.figure(figsize=(12, 6))
sns.histplot(data=iris_data, x='sepal_length', hue='species', multiple='stack', bins=15, kde=False)
```

```
# Add titles and labels
plt.title('Histogram of Sepal Length by Iris Species')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')
plt.legend(title='Species')
plt.show()
```

Slip 26

Q.2 A) Generate a random array of 50 integers and display them using a line chart, scatter plot, histogram and box plot. Apply appropriate color, labels and styling options.

2.Create two lists, one representing subject names and the other representing marks obtained in those subjects. Display the data in bar chart.

```
import matplotlib.pyplot as plt

# Define the subject names and corresponding marks
subjects = ['Math', 'Science', 'English', 'History', 'Art']
marks = [85, 90, 75, 88, 92]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(subjects, marks, color='skyblue')

# Add titles and labels
plt.title('Marks Obtained in Subjects')
plt.xlabel('Subjects')
plt.ylabel('Marks')

# Display the bar chart
plt.ylim(0, 100)  # Set the y-axis limit
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

Slip 27

Q.2) Create a dataset data.csv having two categorical column (the country column, and the purchased column).

a. Apply OneHot coding on Country column.

b. Apply Label encoding on purchased column

```
import pandas as pd
# Sample data
data = {
    'Country': ['USA', 'Canada', 'USA', 'Canada', 'Mexico', 'USA', 'Mexico'],
    'Purchased': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes']
}
```

```
# Create DataFrame
df = pd.DataFrame(data)

# Save to CSV
df.to_csv('data.csv', index=False)
print("data.csv created with the following data:")
print(df)
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

# Load the dataset
df = pd.read_csv('data.csv')

# Display the original data
print("\nOriginal Data:")
print(df)

# One-Hot Encoding on 'Country' column
one_hot_encoder = OneHotEncoder(sparse=False)
country_encoded = one_hot_encoder.fit_transform(df[['Country']])

# Convert the result to a DataFrame and merge with the original DataFrame
```

```
country_df = pd.DataFrame(country_encoded,
columns=one_hot_encoder.get_feature_names_out(['Country']))

df = df.join(country_df)


# Label Encoding on 'Purchased' column

label_encoder = LabelEncoder()

df['Purchased'] = label_encoder.fit_transform(df['Purchased'])


# Display the transformed DataFrame

print("\nTransformed DataFrame:")

print(df)
```

Slip 28

Q.2) Write a Python program

1. To create a dataframe containing columns name, age and percentage. Add 10 rows to the dataframe. View the dataframe.

```
import pandas as pd


# Sample data

data = {

   'name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva',

           'Frank', 'Grace', 'Hannah', 'Ian', 'Julia'],
    'age': [23, 25, 22, 30, 27,
            24, 28, 21, 29, 26],
    'percentage': [85.5, 90.0, 78.5, 88.0, 92.5,
                   75.0, 80.0, 95.0, 89.0, 83.5]
}

# Create DataFrame
df = pd.DataFrame(data)

# View the DataFrame
print("DataFrame:")
print(df)
```

2. To print the shape, number of rows-columns, data types, feature names and the description of the data.

```
import pandas as pd

# Sample data
data = {
    'name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva',
             'Frank', 'Grace', 'Hannah', 'Ian', 'Julia'],
    'age': [23, 25, 22, 30, 27,
            24, 28, 21, 29, 26],
    'percentage': [85.5, 90.0, 78.5, 88.0, 92.5,
          75.0, 80.0, 95.0, 89.0, 83.5]
}

# Create DataFrame
df = pd.DataFrame(data)
# Print the shape of the DataFrame
print("Shape of DataFrame:", df.shape)

# Print the number of rows and columns
num_rows, num_cols = df.shape
print("Number of Rows:", num_rows)
print("Number of Columns:", num_cols)

# Print the data types of the DataFrame
print("\nData Types:")
print(df.dtypes)

# Print the feature names (column names)
print("\nFeature Names:")
print(df.columns.tolist())

# Print the description of the DataFrame
print("\nDescription of Data:")
print(df.describe(include='all'))  # include='all' to include all columns
```

3. To view basic statistical details of the data. 4. To Add 5 rows with duplicate values and missing values. Add a column ‘remarks’ with empty values. Display the data.

```
# Get basic statistical details

statistics = df.describe(include='all')  # include='all' includes all columns

print("Basic Statistical Details:")

print(statistics)
```

Slip 29

Q.2) Create a dataset data.csv having two categorical column (the country column, and the purchased column).

1. Apply OneHot coding on Country column.

2. Apply Label encoding on purchased column

```
import pandas as pd

# Sample data

data = {

    'Country': ['USA', 'Canada', 'USA', 'Canada', 'Mexico', 'USA', 'Mexico'],

    'Purchased': ['Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes']

}

# Create DataFrame

df = pd.DataFrame(data)
```

```
# Save to CSV
df.to_csv('data.csv', index=False)
print("data.csv created with the following data:")
print(df)

from sklearn.preprocessing import OneHotEncoder, LabelEncoder

# Load the dataset
df = pd.read_csv('data.csv')

# Display the original data
print("\nOriginal Data:")
print(df)

# One-Hot Encoding on 'Country' column
one_hot_encoder = OneHotEncoder(sparse=False)
country_encoded = one_hot_encoder.fit_transform(df[['Country']])

# Convert the result to a DataFrame and merge with the original DataFrame
country_df = pd.DataFrame(country_encoded, columns=one_hot_encoder.get_feature_names_out(['Country']))
df = df.join(country_df)
```

```
# Label Encoding on 'Purchased' column
label_encoder = LabelEncoder()
df['Purchased'] = label_encoder.fit_transform(df['Purchased'])

# Display the transformed DataFrame
print("\nTransformed DataFrame:")
print(df)
```