Answer created by Md. Iquball Hossain (NACTAR MLDS Batch-03, Roll: 12)

Machine Learning and Data Science
Module 03 Exam

Questions: Suppose you are working on a data science project and have been given a dataset in CSV format called "sales_data.csv". Your task is to analyze the data and extract meaningful insights using Python. Assume that you have already imported the necessary libraries and loaded the dataset into a Pandas DataFrame named "df".

Solution:

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("sales_data.csv")
```

I. Display the first 5 rows of the data frame.

```
print("I. Display the first 5 rows of the data frame:")
print(df.head())
```

II. Calculate the total number of rows in the DataFrame.

```
total_rows = len(df)
print("\nII. Total number of rows in the DataFrame:", total_rows)
```

III. Check if there are any missing values in the dataset.

```
missing_values = df.isnull().sum().any()
print("\nIII. Any missing values in the dataset:", missing_values)
```

IV. Calculate the average value of the "sales" column.

```
average_sales = df["sales"].mean()
print("\nIV. Average value of the 'sales' column:", average_sales)
```

V. Create a new column called "profit" that contains the difference between the "sales" and "expenses" columns.

```
df["profit"] = df["sales"] - df["expenses"]
```

VI. Sort the DataFrame in descending order based on the "profit" column.

```
df = df.sort_values(by="profit", ascending=False)
print("\nVI. DataFrame sorted in descending order based on the 'profit' column:")
```

VII. Save the sorted DataFrame to a new CSV file called "sorted_sales_data.csv".

```
df.to_csv("sorted_sales_data.csv", index=False)
```

VIII. Visualize the distribution of the "sales" column using a histogram plot.

```
plt.figure(figsize=(10, 6))
plt.hist(df["sales"], bins=20, color='blue', alpha=0.7)
plt.title("Histogram of Sales")
```

```
plt.xlabel("Sales")
plt.ylabel("Frequency")
plt.show()
```

IX. Create a scatter plot to explore the relationship between the "sales" and "expenses" columns.

```
plt.figure(figsize=(10, 6))
plt.scatter(df["sales"], df["expenses"], color='green', alpha=0.7)
plt.title("Scatter Plot: Sales vs Expenses")
plt.xlabel("Sales")
plt.ylabel("Expenses")
plt.show()
```

X. Calculate the correlation coefficient between the "sales" and "expenses" columns.

```
correlation_coefficient = df["sales"].corr(df["expenses"])
print("\nX. Correlation coefficient between 'sales' and 'expenses':", correlation_coefficient)
```

XI. Calculate the total sales for each month and display the result.

```
monthly_total_sales = df.groupby("month")["sales"].sum()
print("\nXI. Total sales for each month:")
print(monthly_total_sales)
```

XII. Find the maximum and minimum sales values for each year in the dataset.

```
yearly_max_sales = df.groupby("year")["sales"].max()
yearly_min_sales = df.groupby("year")["sales"].min()
print("\nXII. Maximum sales for each year:")
print(yearly_max_sales)
print("\n   Minimum sales for each year:")
print(yearly_min_sales)
```

XIII. Calculate the average sales for each category of products.

```
average_sales_by_category = df.groupby("category")["sales"].mean()
print("\nXIII. Average sales for each category of products:")
print(average_sales_by_category)
```

XIV. Group the data by the "region" column and calculate the total sales for each region.

```
total_sales_by_region = df.groupby("region")["sales"].sum()
print("\nXIV. Total sales for each region:")
print(total_sales_by_region)
```

XV. Calculate the sum of expenses for each quarter of the year.

```
expenses_by_quarter = df.groupby("quarter")["expenses"].sum()
print("\nXV. Sum of expenses for each quarter of the year:")
print(expenses_by_quarter)
```