EAST WEST UNIVERSITY

Department of Computer Science & Engineering

Report for Lab-04 (Assignment):

Course Code: CSE303

Course Title: Statistics of Data Science

Section: 01

Submitted To:

Aminul Kader Bulbul
Lecturer
Department of Computer Science & Engineering

Submitted By:

Umme Mukaddisa

ID: 2022-3-60-317

To get access to the CSV file named **mnc_monthly_revenue** provided in the classroom, this part has been implemented initially:

```
[1]  import pandas as pd
     path="/content/drive/MyDrive/mnc_monthly_revenue.csv"
```

Question-01: (Determining rows & columns in the CSV file)

To read the given dataset & determine rows & columns numbers df.shape function has been implemented.

```
#Read the DataSet from CSV
df=pd.read_csv(path)

#Counting row & columns
rows,columns,=df.shape

print("Rows: ",rows)
print("Columns: ",columns)
```

```
Rows:  100
Columns:  32
```

Question-02: (Calculate the mean, median, standard deviation & variance for the dataset)

To calculate the mean, median, standard deviation & variance numerical_df.mean, numerical_df.median & numerical_df.mode has been implemented respectively.

```
# Select only columns for calculations
numerical_df = df.select_dtypes(include=['number'])

# Calculate mean for each column
mean_values = numerical_df.mean()

# Calculate median for each column
median_values = numerical_df.median()

# Calculate mode for each column
#For the multiple values of mode, we only took the firts one
mode_values = numerical_df.mode().iloc[0]

print("Mean values for each column: ")
display(mean_values)

print("\nMedian values for each column: ")
display(median_values)

print("\nMode values for each column: ")
display(mode_values)
```

Mean values for each column:

	0
Month_1	536.25
Month_2	519.94
Month_3	565.44
Month_4	531.89
Month_5	565.17

Median values for each column:

	0
Month_1	555.5
Month_2	507.5
Month_3	581.0
Month_4	554.5
Month_5	526.0
Month_6	640.0

Mode values for each column:

	0
Month_1	387.0
Month_2	239.0
Month_3	279.0
Month_4	171.0
Month_5	147.0
Month_6	300.0

Question-03: (Comparing mean values)

A module has been created initially, then the mean value calculated by the function created in the module & mean value calculated in pandas has been compared & shown.

Module with function:

```
def custom_mean(data):
  return sum(data) / len(data)
```

Comparison:

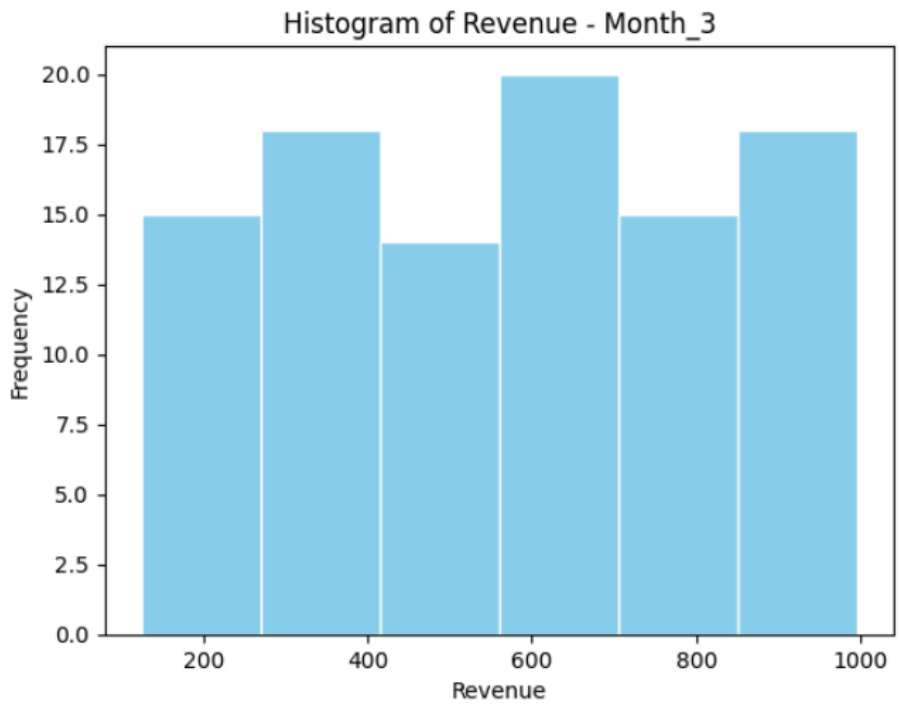
```
import lastattempt as cm

# Select the column
month= (input("Enter the month: "))
month_data = df[month].dropna().tolist()

# Using custom function
my_result = cm.custom_mean(month_data)

# Comparing with pandas
pandas_result = df[month].mean()

# Showing results
print(f"Custom Mean: {my_result}")
print(f"Pandas Mean: {pandas_result}")
```

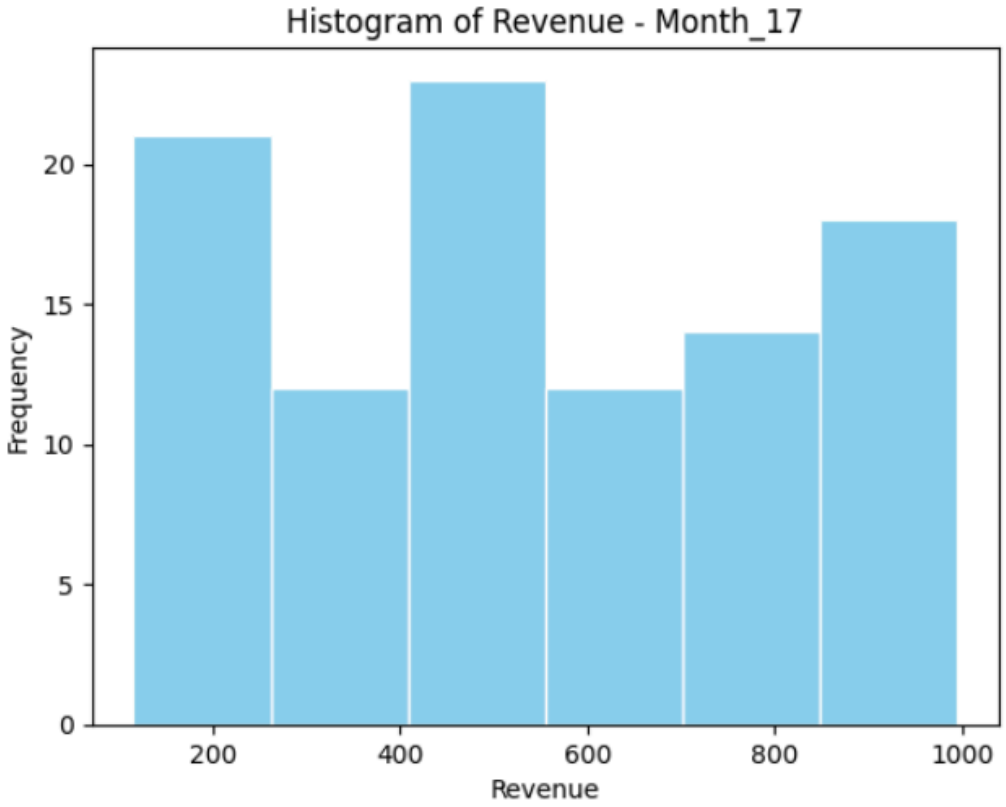
```
Enter the month: Month_5
Custom Mean: 565.17
Pandas Mean: 565.17
```

Question-04: (Plot histograms of revenue values)

For plot histogram matplotlib.pyplot has been used.

```
import matplotlib.pyplot as plt

month_3_data = df['Month_3'].dropna()
month_17_data = df['Month_17'].dropna()


# Month_3 histogram
plt.hist(df['Month_3'].dropna(), bins=6, color='skyblue', edgecolor='white')
plt.title("Histogram of Revenue - Month_3")
plt.xlabel('Revenue')
plt.ylabel('Frequency')
plt.show()


# Month_17 histogram
plt.hist(df['Month_17'].dropna(), bins=6, color='skyblue', edgecolor='white')
plt.title("Histogram of Revenue - Month_17")
plt.xlabel('Revenue')
plt.ylabel('Frequency')
plt.show()
```


Histogram of Revenue - Month_3
Frequency
Revenue
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
200
400
600
800
1000


Histogram of Revenue - Month_17
Frequency
Revenue
0
5
10
15
20
200
400
600
800
1000

Question-05: (High performing-1 & Low performing-0 brands)

To find out high & low performing brands in the dataset, the brands with revenue>$650 have been considered as high performing brands & others are low & set as 1 & 0 respectively in the class.

```
# average monthly revenue for each brand (row)
df['Average_Revenue'] = df.loc[:, 'Month_1':'Month_31'].mean(axis=1)


#  High-Performing (1)  Low-Performing (0)
df['Class'] = (df['Average_Revenue'] > 650).astype(int)


#percentage using class column
class_counts = df['Class'].value_counts(normalize=True) * 100

print("Percentage of High vs. Low Performing Brands:")
display(class_counts)
```

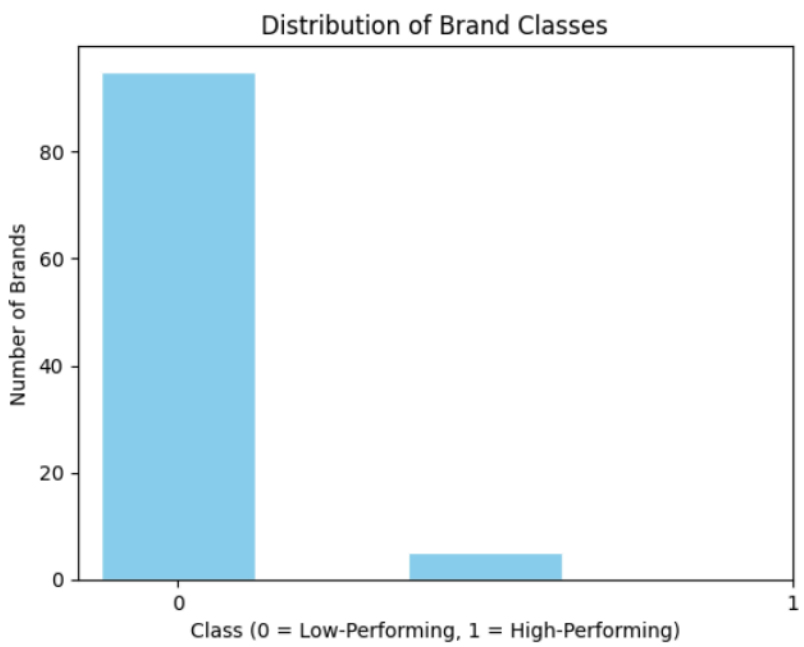
```
Percentage of High vs. Low Performing Brands:
```

Class	proportion
0	98.0
1	2.0

Question-06: (Histogram for high performing-1 & low performing-0 brands)

For plot histogram matplotlib.pyplot has been used implementing the previous class.

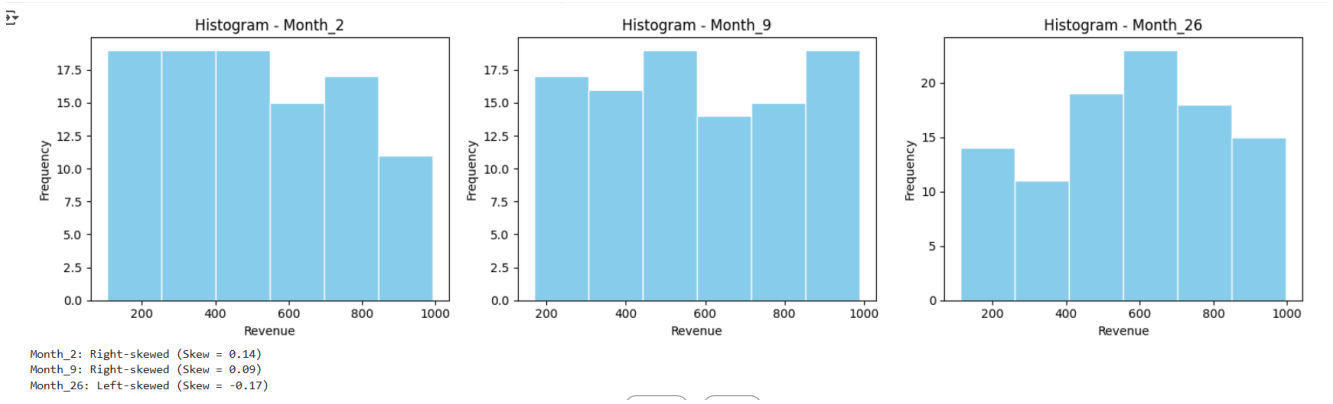
```
import matplotlib.pyplot as plt
# Plot histogram of the Class column
plt.hist(df['Class'], bins=2,color='skyblue', edgecolor='white', align='left', rwidth=0.5)
plt.title('Distribution of Brand Classes')
plt.xlabel('Class (0 = Low-Performing, 1 = High-Performing)')
plt.ylabel('Number of Brands')
plt.xticks([0, 1])  # To ensure x-axis has only 0 and 1
plt.show()
```



Question-07: (Plot histogram for selected months including skewness)

For plot histogram matplotlib.pyplot has been used & skew value>0 considered as right skewed, skew value<0 considered as left skewed & skew value=0 is symmetric.

```
import matplotlib.pyplot as plt

# Choose selected months
months = ['Month_2', 'Month_9', 'Month_26']

# Create subplots
plt.figure(figsize=(15, 4))

for i, month in enumerate(months):
    if month in df.columns:
        plt.subplot(1, 3, i+1)
        plt.hist(df[month].dropna(), bins=6, color='skyblue', edgecolor='white')
        plt.title(f'Histogram - {month}')
        plt.xlabel('Revenue')
        plt.ylabel('Frequency')

plt.tight_layout()
plt.show()


#Printing skewness
for month in months:
    if month in df.columns:
        skew_value = df[month].skew()
        if skew_value > 0:
            skew_type = 'Right-skewed'
        elif skew_value < 0:
            skew_type = 'Left-skewed'
        else:
            skew_type = 'Symmetric'
        print(f"{month}: {skew_type} (Skew = {round(skew_value, 2)})")
```



Question-08: (Correlation analysis to show strong positive correlation in revenue)

We considered corr_value>0.2 to show the strong positive correlation for the dataset.

```
# Select only monthly columns
monthly_revenue = df.loc[:, 'Month_1':'Month_31']

# Compute correlation matrix
correlation_matrix = monthly_revenue.corr()

# Find pairs with strong positive correlation
strong_pairs = []

for i in range(len(correlation_matrix.columns)):
    for j in range(i+1, len(correlation_matrix.columns)):
        month1 = correlation_matrix.columns[i]
        month2 = correlation_matrix.columns[j]
        corr_value = correlation_matrix.iloc[i, j]
        if corr_value > 0.2:
            strong_pairs.append((month1, month2, round(corr_value, 2)))

print("Strongly Positively Correlated Month Pairs: ")
for m1, m2, val in strong_pairs:
    print(f"{m1} and {m2} → Correlation: {val}")
```

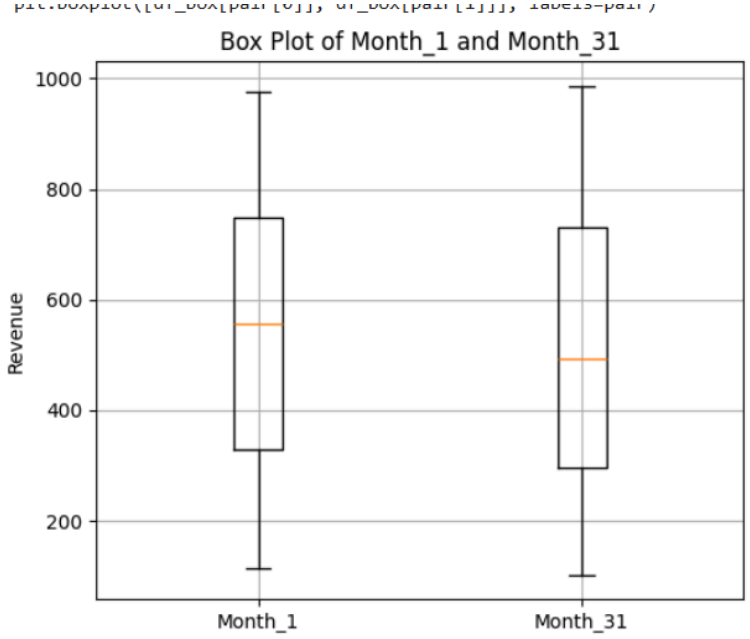
```
Strongly Positively Correlated Month Pairs:
Month_1 and Month_13 → Correlation: 0.28
Month_1 and Month_31 → Correlation: 0.29
Month_3 and Month_12 → Correlation: 0.26
Month_3 and Month_29 → Correlation: 0.23
Month_10 and Month_11 → Correlation: 0.21
Month_10 and Month_28 → Correlation: 0.21
Month_11 and Month_28 → Correlation: 0.24
Month_12 and Month_26 → Correlation: 0.21
Month_12 and Month_27 → Correlation: 0.27
Month_12 and Month_31 → Correlation: 0.27
Month_15 and Month_18 → Correlation: 0.28
Month_19 and Month_20 → Correlation: 0.27
Month_20 and Month_23 → Correlation: 0.24
Month_23 and Month_29 → Correlation: 0.26
```

Question-09: (BoxPlots for a few selected positively correlated month pairs)

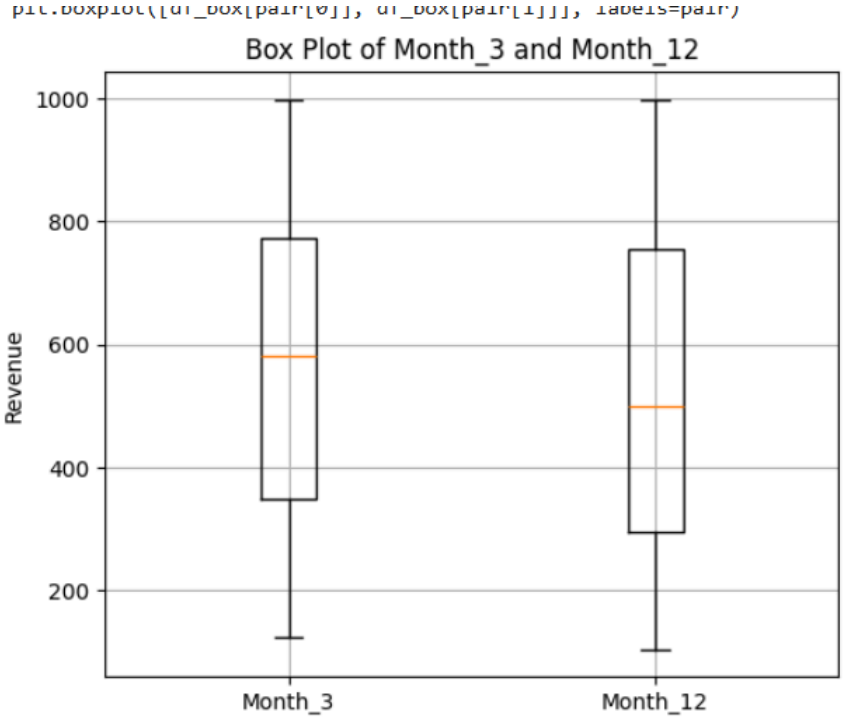
Using previous code we took 2 pairs of positively correlated months & then used matplotlib.pyplot for boxplot.

```python
import matplotlib.pyplot as plt

#correlated month pairs
month_pairs = [('Month_1', 'Month_31'), ('Month_3', 'Month_12')]


# Plotting each pair
for pair in month_pairs:
    plt.figure(figsize=(6, 5))
    df_box = df[list(pair)].dropna()  # select both columns and remove NaNs
    plt.boxplot([df_box[pair[0]], df_box[pair[1]]], labels=pair)
    plt.title(f'Box Plot of {pair[0]} and {pair[1]}')
    plt.ylabel('Revenue')
    plt.grid(True)
    plt.show()
```

First pair:



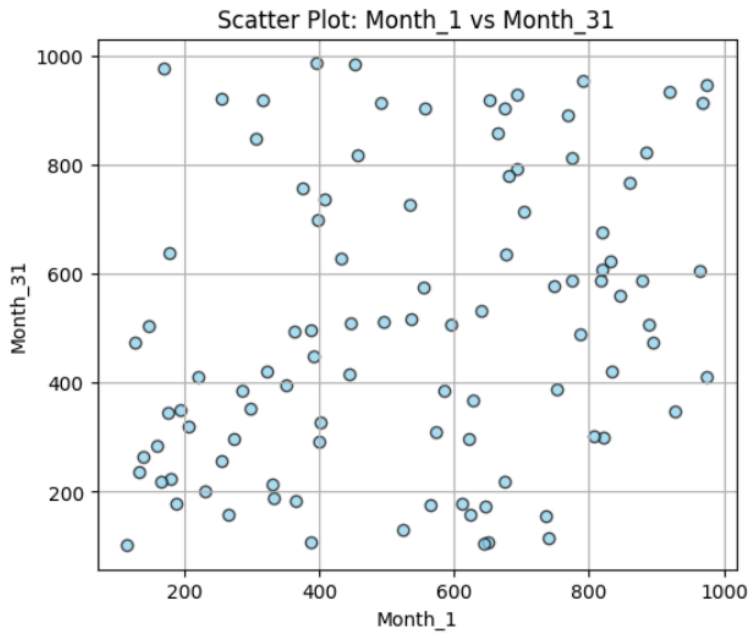
Second pair:

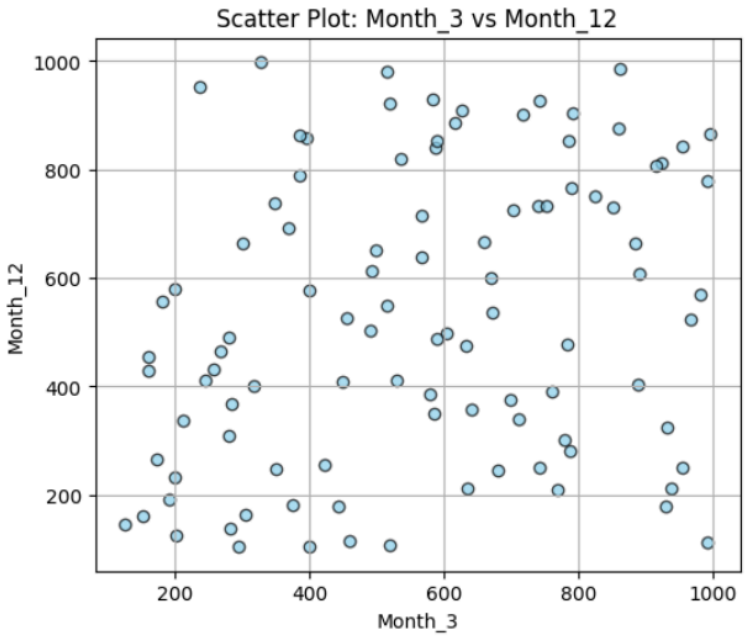


Question-10: (Scatter plot to show linear relationship)

The month pairs used before have been used here to show the scatter plots to get access to the linear relationship.

```
import matplotlib.pyplot as plt

#correlated month pairs
month_pairs = [('Month_1', 'Month_31'), ('Month_3', 'Month_12')]

# Loop over pairs & Scatter plots
for pair in month_pairs:
    plt.figure(figsize=(6, 5))
    plt.scatter(df[pair[0]], df[pair[1]], alpha=0.7, color='skyblue', edgecolor='black')
    plt.title(f'Scatter Plot: {pair[0]} vs {pair[1]}')
    plt.xlabel(pair[0])
    plt.ylabel(pair[1])
    plt.grid(True)
    plt.show()
```





Question-11: (Correlation analysis to show negative correlation in revenue)

We considered corr_value<-0.2 to show the negative correlation for the dataset.

```
monthly_data = df.loc[:, 'Month_1':'Month_31']
corr_matrix = monthly_data.corr()

#pairs with strong negative correlation
negative_corr_pairs = []

for i in range(len(corr_matrix.columns)):
    for j in range(i + 1, len(corr_matrix.columns)):
        month1 = corr_matrix.columns[i]
        month2 = corr_matrix.columns[j]
        corr_value = corr_matrix.iloc[i, j]
        if corr_value < -0.2:
            negative_corr_pairs.append((month1, month2, round(corr_value, 2)))

if negative_corr_pairs:
    print("Month pairs with strong negative correlation: ")
    for m1, m2, val in negative_corr_pairs:
        print(f"{m1} and {m2} → Correlation: {val}")
else:
    print("No strongly negatively correlated month pairs found.")
```

```
Month pairs with strong negative correlation:
Month_1 and Month_4 → Correlation: -0.22
Month_2 and Month_5 → Correlation: -0.22
Month_3 and Month_4 → Correlation: -0.2
Month_4 and Month_7 → Correlation: -0.23
Month_7 and Month_10 → Correlation: -0.24
Month_8 and Month_31 → Correlation: -0.2
Month_13 and Month_27 → Correlation: -0.23
Month_17 and Month_24 → Correlation: -0.34
Month_18 and Month_30 → Correlation: -0.22
Month_20 and Month_26 → Correlation: -0.21
Month_21 and Month_31 → Correlation: -0.23
```

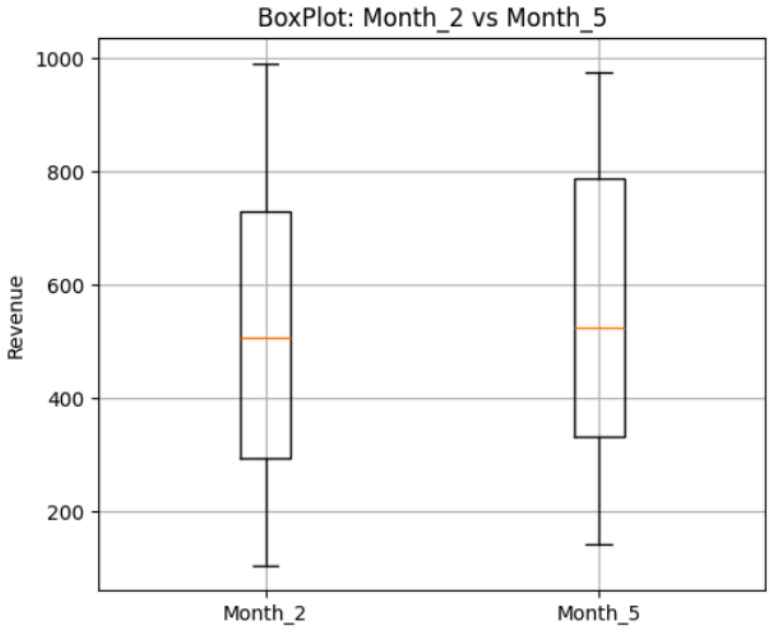
Question-12: (BoxPlots for negatively correlated month pairs)

Using previous code we took 2 pairs of negatively correlated months & then used matplotlib.pyplot for boxplot.

```
import matplotlib.pyplot as plt
#From code above
negatively_correlated_months = [('Month_2', 'Month_5'), ('Month_7', 'Month_10')]


for pair in negatively_correlated_months:
    month1, month2 = pair
    plt.figure(figsize=(6, 5))


    df_box = df[[month1, month2]].dropna()

    # Create boxplot
    plt.boxplot([df_box[month1], df_box[month2]], labels=[month1, month2])
    plt.title(f'BoxPlot: {month1} vs {month2}')
    plt.ylabel('Revenue')
    plt.grid(True)
    plt.show()
```

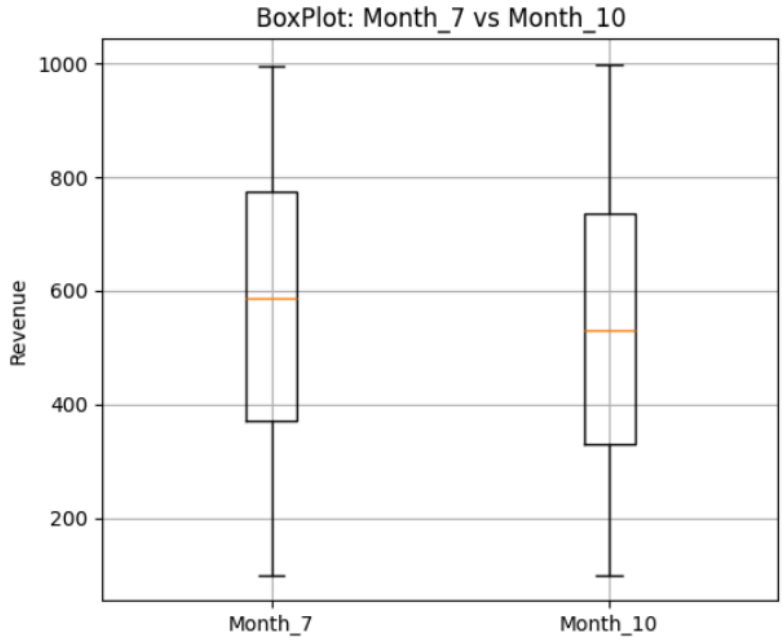
First pair:

```
/tmp/ipython-input-67-3559410049.py:14: MatplotlibDeprecationWarning: The 'lab
  plt.boxplot([df_box[month1], df_box[month2]], labels=[month1, month2])
```



/tmp/ipython-input-67-3559410049.py:14: MatplotlibDeprecationWarning: The 'lab

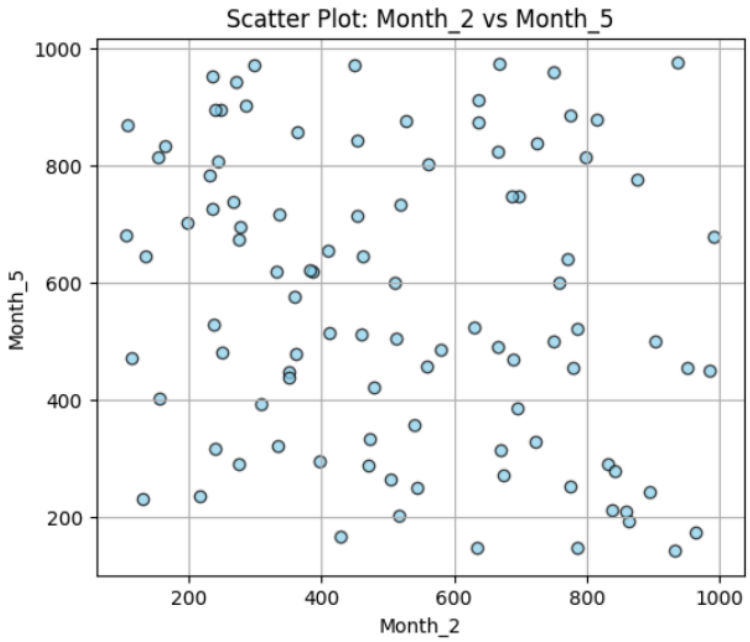
Second pair:

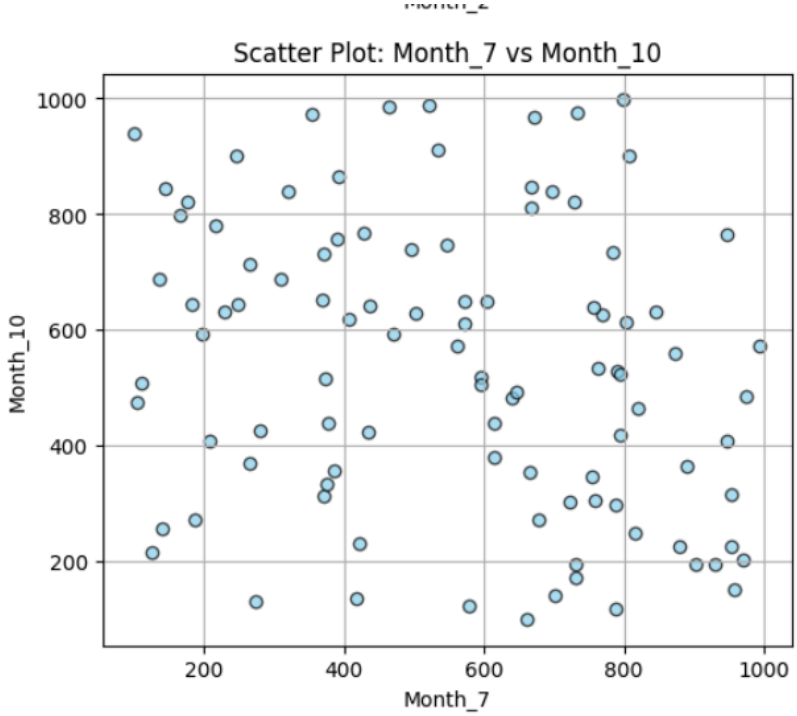


Question-13: (Scatterplot to visualize inverse relationship)

The month pairs used before have been used here to show the scatter plots to get access to the linear relationship.

```
import matplotlib.pyplot as plt
#From code above
negatively_correlated_months = [('Month_2', 'Month_5'), ('Month_7', 'Month_10')]

# Scatter plots
for month1, month2 in negatively_correlated_months:
    plt.figure(figsize=(6, 5))
    plt.scatter(df[month1], df[month2], alpha=0.7, color='skyblue', edgecolor='black')
    plt.title(f'Scatter Plot: {month1} vs {month2}')
    plt.xlabel(month1)
    plt.ylabel(month2)
    plt.grid(True)
    plt.show()
```




Scatter Plot: Month_7 vs Month_10
Month_10
Month_7
1000
800
600
400
200
200
400
600
800
1000