**Name:** **Sadaf Iqbal**  
**Department:** **BS Computer Science (BSCS)**  
**University Roll No.:** **23-ARID-740**  
**Course:** **Python Programming**  
**Semester:** **5th – Evening**  
**Submitted To:** **[M.Azhar ]**  
**Date:** **[12/08/2025 ]**

**---------- Statistic Analysis ------------**

**create file/array of Data for different Domain like computing, medical , social sciences etc . Generate sample Data using AI OR DOWNLOAD ANY SAMPLE DATASET.**

**and perform Statictic Analysis .**

**Steps to perform:**

**code in python**

**1. load Data Set of different Domain.**

**2. Statistic Analysis techniques/ formulas**

**3. peform calculation Function (using build-in/custom function)**

**4. Show result in Graph & Table format**

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

# 1. Generate Sample Datasets from Different Domains

def generate\_sample\_data():

    # Computing Domain (e.g., algorithm performance)

    computing\_data = pd.DataFrame({

        'Algorithm': ['A', 'B', 'C', 'D', 'E'],

        'Execution Time (ms)': np.random.normal(200, 50, 5),

        'Memory Usage (MB)': np.random.normal(120, 15, 5)

    })

    # Medical Domain (e.g., patient data)

    medical\_data = pd.DataFrame({

        'Patient ID': range(1, 11),

        'Age': np.random.randint(20, 80, 10),

        'Blood Pressure': np.random.randint(110, 180, 10),

        'Cholesterol': np.random.randint(150, 250, 10)

    })

    # Social Science Domain (e.g., survey)

    social\_data = pd.DataFrame({

        'Respondent ID': range(1, 11),

        'Happiness Score': np.random.uniform(1, 10, 10),

        'Income (k)': np.random.randint(20, 100, 10),

        'Education Level': np.random.choice(['High School', 'Bachelor', 'Master', 'PhD'], 10)

    })

    return computing\_data, medical\_data, social\_data

# 2. Define Statistical Analysis Function

def perform\_statistics(df, numeric\_columns):

    stats\_dict = {}

    for col in numeric\_columns:

        mode\_val = stats.mode(df[col], keepdims=True).mode[0]  # Fixed for SciPy >= 1.9

        stats\_dict[col] = {

            'Mean': np.mean(df[col]),

            'Median': np.median(df[col]),

            'Mode': mode\_val,

            'Standard Deviation': np.std(df[col]),

            'Variance': np.var(df[col]),

            'Min': np.min(df[col]),

            'Max': np.max(df[col]),

        }

    return pd.DataFrame(stats\_dict)

# 3. Plotting Functions

def plot\_histograms(df, numeric\_columns, title):

    for col in numeric\_columns:

        plt.figure()

        sns.histplot(df[col], kde=True)

        plt.title(f"{title}: Histogram of {col}")

        plt.xlabel(col)

        plt.ylabel('Frequency')

        plt.grid(True)

        plt.show()

def plot\_correlation(df, numeric\_columns, title):

    plt.figure(figsize=(6, 4))

    sns.heatmap(df[numeric\_columns].corr(), annot=True, cmap='coolwarm')

    plt.title(f"{title}: Correlation Matrix")

    plt.show()

# 4. Main Program

def main():

    computing\_data, medical\_data, social\_data = generate\_sample\_data()

    print("---------- Computing Domain ----------")

    comp\_stats = perform\_statistics(computing\_data, ['Execution Time (ms)', 'Memory Usage (MB)'])

    print(comp\_stats)

    plot\_histograms(computing\_data, ['Execution Time (ms)', 'Memory Usage (MB)'], "Computing")

    plot\_correlation(computing\_data, ['Execution Time (ms)', 'Memory Usage (MB)'], "Computing")

    print("\n---------- Medical Domain ----------")

    med\_stats = perform\_statistics(medical\_data, ['Age', 'Blood Pressure', 'Cholesterol'])

    print(med\_stats)

    plot\_histograms(medical\_data, ['Age', 'Blood Pressure', 'Cholesterol'], "Medical")

    plot\_correlation(medical\_data, ['Age', 'Blood Pressure', 'Cholesterol'], "Medical")

    print("\n---------- Social Sciences Domain ----------")

    social\_numeric = ['Happiness Score', 'Income (k)']

    social\_stats = perform\_statistics(social\_data, social\_numeric)

    print(social\_stats)

    plot\_histograms(social\_data, social\_numeric, "Social Sciences")

    plot\_correlation(social\_data, social\_numeric, "Social Sciences")

# ✅ Correct way to run the script

if \_\_name\_\_ == "\_\_main\_\_":

    main()

output:

























