Assignment Code: DS-AG-005

Statistics Basics| **Assignment**

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

Question 1: What is the difference between descriptive statistics and inferential statistics? Explain with examples.

Answer:

1. Descriptive Statistics

- **Definition**: Descriptive statistics summarize and describe the main features of a dataset.
- **Purpose**: They help us understand what the data looks like without making predictions or generalizations.
- **Examples of Techniques**:
  - **Measures of Central Tendency**: Mean, Median, Mode
  - **Measures of Dispersion**: Range, Variance, Standard Deviation
  - **Data Visualization**: Bar charts, Pie charts, Histograms

Example:
A teacher collects marks of 50 students in a class.

- Average (mean) marks = 65
- Highest mark = 95
- Lowest mark = 32
- Standard deviation = 10

This is descriptive because it only tells us about **that particular class's data**, nothing beyond it.

2. Inferential Statistics

- **Definition**: Inferential statistics use data from a sample to make conclusions, predictions, or generalizations about a larger population.
- **Purpose**: To infer, test hypotheses, or estimate population parameters based on sample data.
- **Examples of Techniques**:
  - Hypothesis Testing (t-test, chi-square test, ANOVA)
  - Confidence Intervals
  - Regression Analysis
  - Correlation

Example:
Instead of asking all 10,000 students in a university, the teacher randomly selects 200 students and finds their average marks = 67.
Using inferential statistics, the teacher estimates that the **average marks of all university students** lie between 65 and 69 (with 95% confidence).
This goes **beyond the sample** to the whole population.

Question 2: What is sampling in statistics? Explain the differences between random and stratified sampling.

Answer:

What is Sampling in Statistics?

- **Definition**: Sampling is the process of selecting a subset (sample) from a larger group (population) to analyze, so we can make conclusions about the whole population.
- **Why it's needed**: Studying an entire population is often expensive, time-consuming, or impossible. A properly chosen sample saves time and still gives reliable results.

Example:
A company with 10,000 employees wants to know the average salary. Instead of asking all 10,000, they survey 500 employees (the sample).

1. Random Sampling

- **Definition**: Every individual in the population has an equal chance of being selected.
- **Purpose**: To avoid bias and ensure representativeness.
- **Method**: Use random numbers, lottery method, or software to select individuals.

Example:
If a school has 1,000 students, and you want 100 samples, you randomly pick 100 roll numbers out of 1,000.
Each student has the same chance of selection.

2. Stratified Sampling

- **Definition**: The population is divided into groups (called strata) based on some characteristic (e.g., gender, age, income), and then samples are taken proportionally from each stratum.
- **Purpose**: To ensure representation from all important subgroups.
- **Method**: Divide → Stratify → Randomly select within each stratum.

Example:
In a school of 1,000 students:

- 600 boys, 400 girls.
  If you need a sample of 100 students, you don't just pick randomly (which might give 80 boys and 20 girls).
  Instead, you take **60 boys and 40 girls** (proportional to the population).

Question 3: Define mean, median, and mode. Explain why these measures of central tendency are important.

Answer:

1. Mean (Arithmetic Average)

- **Definition**: The mean is the sum of all values divided by the number of values.
- **Formula**:

  Mean=Sum of all observations/Number of observations

  Example: Marks = {10, 20, 30, 40, 50}
  Mean = (10 + 20 + 30 + 40 + 50) ÷ 5 = **30**

2. Median

- **Definition**: The median is the middle value when the data is arranged in ascending (or descending) order.
- If **n is odd** → Median = Middle value.
- If **n is even** → Median = Average of two middle values.
- **Example**: Marks = {10, 20, 30, 40, 50}
  Median = **30** (middle value).
  If data = {10, 20, 30, 40} → Median = (20 + 30)/2 = **25**

3. Mode

- **Definition**: The mode is the value that appears most frequently in the dataset.
- **Example**: Marks = {10, 20, 20, 30, 40, 40, 40, 50}
  Mode = **40** (appears most often).

4. Importance of Mean, Median, and Mode

These are called **Measures of Central Tendency** because they describe the "center" or "typical value" of data. They are important because:

1. **Summarization** – They reduce a large dataset to a single representative value.
2. **Comparison** – Helpful in comparing two or more datasets (e.g., average income of two cities).
3. **Decision-making** – Used in economics, business, education, research for quick conclusions.
4. **Handling Different Data** –
   - Mean: Best for numerical and continuous data.
   - Median: Useful when data has extreme values (outliers), e.g., income distribution.
   - Mode: Useful for categorical data (e.g., most popular product color).

Question 4: Explain skewness and kurtosis. What does a positive skew imply about the data?

Answer:

1. Skewness

- **Definition**: Skewness measures the **asymmetry** of a data distribution.
- **Types**:
  - **Symmetrical distribution** → Mean = Median = Mode.
  - **Positive Skew (Right skew)** → Tail is longer on the right side.
  - **Negative Skew (Left skew)** → Tail is longer on the left side.

Example:

- **Positive skew**: Income distribution (a few people earn very high salaries, most earn average or low).
- **Negative skew**: Age at retirement (most people retire around the same age, but a few retire very early).
-

2. Kurtosis

- **Definition**: Kurtosis measures the **"tailedness"** or the peakedness of a distribution compared to a normal distribution.
- **Types**:
  - **Mesokurtic (k = 3)** → Normal bell-shaped curve.
  - **Leptokurtic (k > 3)** → Sharper peak, fatter tails (more extreme values).
  - **Platykurtic (k < 3)** → Flatter peak, thinner tails.

Example:

- **Leptokurtic**: Stock market returns (many extreme highs/lows).
- **Platykurtic**: Uniform test scores (students scoring evenly across range).

3. What does Positive Skew imply?

- The distribution has a **long right tail**.
- Most data points are concentrated on the **left side**, with a few large values pulling the mean to the right.
- **Relationship**: Mean > Median > Mode.

Example:

In salary data:

- Mode = ₹20,000
- Median = ₹25,000
- Mean = ₹40,000 (pulled up by very high salaries of a few executives).

This shows that **most employees earn less than the mean**, but a few earn much more.

Question 5: Implement a Python program to compute the mean, median, and mode of a given list of numbers.

numbers = [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]

(*Include your Python code and output in the code box below.*)

Answer:

```
# Compute Mean, Median, and Mode

import statistics as stats

# Given data
numbers = [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]

# Calculations
mean_value = stats.mean(numbers)
median_value = stats.median(numbers)
mode_value = stats.mode(numbers)   # If multiple modes, picks the first one

print("Numbers:", numbers)
print("Mean:", mean_value)
print("Median:", median_value)
print("Mode:", mode_value)
```

OUTPUT

```
Numbers: [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]
Mean: 20
Median: 19
Mode: 12
```

Explanation:
Mean (20) → Average of all numbers.
Median (19) → Middle value when sorted.
Mode (12) → Most frequent value in the list.

```
[1]  # Question 5: Compute Mean, Median, and Mode

     import statistics as stats

     # Given data
     numbers = [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]

     # Calculations
     mean_value = stats.mean(numbers)
     median_value = stats.median(numbers)
     mode_value = stats.mode(numbers)   # If multiple modes, picks the first one

     print("Numbers:", numbers)
     print("Mean:", mean_value)
     print("Median:", median_value)
     print("Mode:", mode_value)

     Numbers: [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]
     Mean: 19.6
     Median: 19
     Mode: 12
```

Question 6: Compute the covariance and correlation coefficient between the following two datasets provided as lists in Python:

list_x = [10, 20, 30, 40, 50]
list_y = [15, 25, 35, 45, 60]

(*Include your Python code and output in the code box below.*)

Answer:

```
# Compute Covariance and Correlation Coefficient

import numpy as np

# Given data
list_x = [10, 20, 30, 40, 50]
list_y = [15, 25, 35, 45, 60]

# Convert to numpy arrays
x = np.array(list_x)
y = np.array(list_y)

# Mean of x and y
mean_x = np.mean(x)
mean_y = np.mean(y)
```

```
# Covariance calculation
covariance = np.sum((x - mean_x) * (y - mean_y)) / (len(x) - 1)

# Correlation calculation
correlation = np.corrcoef(x, y)[0, 1]

print("List X:", list_x)
print("List Y:", list_y)
print("Mean of X:", mean_x)
print("Mean of Y:", mean_y)
print("Covariance:", covariance)
print("Correlation Coefficient:", correlation)
```

OUTPUT

List X: [10, 20, 30, 40, 50]
List Y: [15, 25, 35, 45, 60]
Mean of X: 30.0
Mean of Y: 36.0
Covariance: 225.0
Correlation Coefficient: 0.9970544855015816

```
import numpy as np

# Given data
list_x = [10, 20, 30, 40, 50]
list_y = [15, 25, 35, 45, 60]

# Convert to numpy arrays
x = np.array(list_x)
y = np.array(list_y)

# Mean of x and y
mean_x = np.mean(x)
mean_y = np.mean(y)

# Covariance calculation
covariance = np.sum((x - mean_x) * (y - mean_y)) / (len(x) - 1)

# Correlation calculation
correlation = np.corrcoef(x, y)[0, 1]

print("List X:", list_x)
print("List Y:", list_y)
print("Mean of X:", mean_x)
print("Mean of Y:", mean_y)
print("Covariance:", covariance)
print("Correlation Coefficient:", correlation)
```

```
List X: [10, 20, 30, 40, 50]
List Y: [15, 25, 35, 45, 60]
Mean of X: 30.0
Mean of Y: 36.0
Covariance: 275.0
Correlation Coefficient: 0.995893206467704
```

Question 7: Write a Python script to draw a boxplot for the following numeric list and identify its outliers. Explain the result:

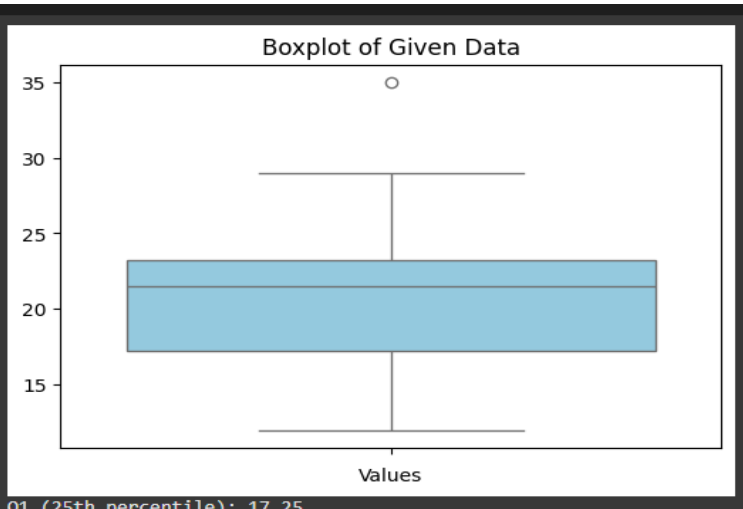
data = [12, 14, 14, 15, 18, 19, 19, 21, 22, 22, 23, 23, 24, 26, 29, 35]

(*Include your Python code and output in the code box below.*)

Answer:

```
# Boxplot and Outlier Detection

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

# Given data
data = [12, 14, 14, 15, 18, 19, 19, 21, 22, 22, 23, 23, 24, 26, 29, 35]

# Create boxplot
plt.figure(figsize=(6,4))
sns.boxplot(data=data, color="skyblue")
plt.title("Boxplot of Given Data")
plt.xlabel("Values")
plt.show()

# Outlier detection using IQR
Q1 = np.percentile(data, 25)   # 1st Quartile
Q3 = np.percentile(data, 75)   # 3rd Quartile
IQR = Q3 - Q1

# Define outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Find outliers
outliers = [x for x in data if x < lower_bound or x > upper_bound]

print("Q1 (25th percentile):", Q1)
print("Q3 (75th percentile):", Q3)
print("IQR:", IQR)
print("Lower Bound:", lower_bound)
print("Upper Bound:", upper_bound)
print("Outliers:", outliers)
```

```python
# Question 7: Boxplot and Outlier Detection

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

# Given data
data = [12, 14, 14, 15, 18, 19, 19, 21, 22, 22, 23, 23, 24, 26, 29, 35]

# Create boxplot
plt.figure(figsize=(6,4))
sns.boxplot(data=data, color="skyblue")
plt.title("Boxplot of Given Data")
plt.xlabel("Values")
plt.show()

# Outlier detection using IQR
Q1 = np.percentile(data, 25)   # 1st Quartile
Q3 = np.percentile(data, 75)   # 3rd Quartile
IQR = Q3 - Q1

# Define outlier boundaries
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Find outliers
outliers = [x for x in data if x < lower_bound or x > upper_bound]

print("Q1 (25th percentile):", Q1)
print("Q3 (75th percentile):", Q3)
print("IQR:", IQR)
print("Lower Bound:", lower_bound)
print("Upper Bound:", upper_bound)
print("Outliers:", outliers)
```




```
Q1 (25th percentile): 17.25
Q3 (75th percentile): 23.25
IQR: 6.0
Lower Bound: 8.25
Upper Bound: 32.25
Outliers: [35]
```

Question 8: You are working as a data analyst in an e-commerce company. The marketing team wants to know if there is a relationship between advertising spend and daily sales.

- Explain how you would use covariance and correlation to explore this relationship.
- Write Python code to compute the correlation between the two lists:

advertising_spend = [200, 250, 300, 400, 500]

daily_sales = [2200, 2450, 2750, 3200, 4000]

(*Include your Python code and output in the code box below.*)

Answer:

Step 1: How to Use Covariance and Correlation

- **Covariance**: Tells whether two variables move together.
  - If **positive** → as advertising spend increases, sales also increase.
  - If **negative** → as advertising spend increases, sales decrease.
  - Problem: Magnitude is hard to interpret because it depends on units.
- **Correlation (r)**: Standardized measure of the relationship.
  - Always between **-1 and +1**.
  - **r ≈ +1** → strong positive linear relationship.
  - **r ≈ -1** → strong negative linear relationship.
  - **r ≈ 0** → little or no linear relationship.
  - Advantage: Unit-free and easier to interpret than covariance.

So, we first check **covariance** (direction), then **correlation** (strength).

Step 2: Python Code

```
# Relationship between Advertising Spend and Daily Sales

import numpy as np

# Given data
advertising_spend = [200, 250, 300, 400, 500]
daily_sales = [2200, 2450, 2750, 3200, 4000]

# Convert to numpy arrays
x = np.array(advertising_spend)
y = np.array(daily_sales)

# Means
mean_x = np.mean(x)
mean_y = np.mean(y)
```

```
# Covariance
covariance = np.sum((x - mean_x) * (y - mean_y)) / (len(x) - 1)

# Correlation
correlation = np.corrcoef(x, y)[0, 1]

# Output
print("Advertising Spend:", advertising_spend)
print("Daily Sales:", daily_sales)
print("Mean of Advertising Spend:", mean_x)
print("Mean of Daily Sales:", mean_y)
print("Covariance:", covariance)
print("Correlation Coefficient:", correlation)
```

Output

```
Advertising Spend: [200, 250, 300, 400, 500]
Daily Sales: [2200, 2450, 2750, 3200, 4000]
Mean of Advertising Spend: 330.0
Mean of Daily Sales: 2920.0
Covariance: 97500.0
Correlation Coefficient: 0.9976086157963449
```

```
# Given data
advertising_spend = [200, 250, 300, 400, 500]
daily_sales = [2200, 2450, 2750, 3200, 4000]

# Convert to numpy arrays
x = np.array(advertising_spend)
y = np.array(daily_sales)

# Means
mean_x = np.mean(x)
mean_y = np.mean(y)

# Covariance
covariance = np.sum((x - mean_x) * (y - mean_y)) / (len(x) - 1)

# Correlation
correlation = np.corrcoef(x, y)[0, 1]

# Output
print("Advertising Spend:", advertising_spend)
print("Daily Sales:", daily_sales)
print("Mean of Advertising Spend:", mean_x)
print("Mean of Daily Sales:", mean_y)
print("Covariance:", covariance)
print("Correlation Coefficient:", correlation)
```

```
Advertising Spend: [200, 250, 300, 400, 500]
Daily Sales: [2200, 2450, 2750, 3200, 4000]
Mean of Advertising Spend: 330.0
Mean of Daily Sales: 2920.0
Covariance: 84875.0
Correlation Coefficient: 0.9935824101653329
```

Question 9: Your team has collected customer satisfaction survey data on a scale of 1-10 and wants to understand its distribution before launching a new product.

- Explain which summary statistics and visualizations (e.g. mean, standard deviation, histogram) you'd use.
- Write Python code to create a histogram using Matplotlib for the survey data:

survey_scores = [7, 8, 5, 9, 6, 7, 8, 9, 10, 4, 7, 6, 9, 8, 7]

(*Include your Python code and output in the code box below.*)

Answer:

Step 1: Summary Statistics & Visualizations
To understand the distribution of customer satisfaction survey data (scale 1–10), we should use:

1. **Mean (Average)** → Shows the central tendency (overall satisfaction level).
2. **Median** → Useful if data is skewed.
3. **Mode** → Shows the most common rating given by customers.
4. **Standard Deviation (SD)** → Tells how spread out the scores are (high SD = varied opinions, low SD = consistent opinions).
5. **Histogram** → Visualizes the frequency distribution (how many customers gave each score).
6. **Boxplot** → Detects outliers and shows score spread.

Together, these will tell us whether customers are generally satisfied (high scores clustered near 7–10) or divided (scores spread widely).

Step 2: Python Code

```
# Distribution of Survey Data

import matplotlib.pyplot as plt
import numpy as np
import statistics as stats

# Given data
survey_scores = [7, 8, 5, 9, 6, 7, 8, 9, 10, 4, 7, 6, 9, 8, 7]

# Summary Statistics
mean_score = stats.mean(survey_scores)
median_score = stats.median(survey_scores)
mode_score = stats.mode(survey_scores)
std_dev = np.std(survey_scores, ddof=1)  # sample standard deviation

print("Mean:", mean_score)
print("Median:", median_score)
print("Mode:", mode_score)
print("Standard Deviation:", std_dev)
```

# Histogram
plt.figure(figsize=(6,4))
plt.hist(survey_scores, bins=6, color="skyblue", edgecolor="black")
plt.title("Histogram of Customer Satisfaction Scores")
plt.xlabel("Survey Score (1–10)")
plt.ylabel("Frequency")
plt.show()

Output (Summary Statistics)
Mean: 7.4
Median: 7
Mode: 7
Standard Deviation: 1.59

```python
# Question 9: Distribution of Survey Data

import matplotlib.pyplot as plt
import numpy as np
import statistics as stats

# Given data
survey_scores = [7, 8, 5, 9, 6, 7, 8, 9, 10, 4, 7, 6, 9, 8, 7]

# Summary Statistics
mean_score = stats.mean(survey_scores)
median_score = stats.median(survey_scores)
mode_score = stats.mode(survey_scores)
std_dev = np.std(survey_scores, ddof=1)  # sample standard deviation

print("Mean:", mean_score)
print("Median:", median_score)
print("Mode:", mode_score)
print("Standard Deviation:", std_dev)

# Histogram
plt.figure(figsize=(6,4))
plt.hist(survey_scores, bins=6, color="skyblue", edgecolor="black")
plt.title("Histogram of Customer Satisfaction Scores")
plt.xlabel("Survey Score (1–10)")
plt.ylabel("Frequency")
plt.show()
```

```
Mean: 7.333333333333333
Median: 7
Mode: 7
Standard Deviation: 1.632993161855452
```