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

Answer: **Descriptive Statistics**

Definition: Descriptive statistics involve methods for **organizing, summarizing, and presenting data** in a meaningful way without making predictions or generalizations beyond the data. They help researchers understand the features of a dataset at a glance.

Common tools and measures:

- **Measures of central tendency:** mean, median, mode
- **Measures of dispersion:** range, variance, standard deviation
- **Data visualization:** graphs, charts, histograms, and frequency tables

Example: Suppose a teacher collects the marks of 50 students in a class. Descriptive statistics would compute the **average score** (mean), identify the **most common score** (mode), and summarize the **spread of scores** (standard deviation). A graph showing the distribution of marks would also be a form of descriptive analysis.

Inferential Statistics

Definition: Inferential statistics involve techniques for **making generalizations, predictions, or decisions about a population** based on a smaller sample. It uses probability theory to draw conclusions and estimate uncertainty.

Common methods:

- **Hypothesis testing:** t-tests, chi-square tests, ANOVA
- **Confidence intervals:** estimating population parameters from sample data
- **Regression analysis and correlations:** predicting relationships among variables

Example: Using the 50 students' marks, the teacher wants to **infer the average performance of all students in the school**, not just those in the class. Inferential statistics allow the teacher to **estimate the population mean** and decide, for instance, whether the class performs significantly better than students in other schools using hypothesis testing

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

Answer: Stratified random sampling divides a population into different groups based on certain characteristics, and a random sample is taken from each.
Simple random sampling is a statistical tool used to describe a very basic sample taken from a data population. This sample represents the equivalent of the entire population.

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

Answer:Mean, Median, and Mode are measures of the central tendency. These values are used to define the various parameters of the given data set. The measure of central tendency (Mean, Median, and Mode) gives useful insights about the data studied, these are used to study any type of data such as the average salary of employees in an organization, the median age of any class, the number of people who plays cricket in a sports club, etc.

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

Answer:**Skewness**
Skewness measures the asymmetry of a data distribution around its mean. It tells us whether the distribution is tilted to the left, right, or symmetrical.
Kurtosis
Kurtosis measures the "tailedness" or peakedness of a distribution. It tells us whether the data are heavy-tailed or light-tailed compared to a normal distribution.
Example:
If the income of a small town is positively skewed, most people earn a moderate income, but a few very high incomes stretch the distribution to the right, increasing the mean above the median.

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, 219, 19, 24, 24, 24, 26, 28] (Include your Python code and output in the code box below.)

Answer:This Python program demonstrates how to calculate the **mean**, **median**, and **mode** of a given list of numbers using both the built-in `statistics` module and manual calculations for clarity.

```
# Python program to compute mean, median, and mode of a given list

from statistics import mean, median, mode, StatisticsError

# Function to calculate mean, median, and mode
def compute_statistics(numbers):
    try:
        # Calculate mean
        avg = mean(numbers)

        # Calculate median
        med = median(numbers)

        # Calculate mode
        try:
            mod = mode(numbers)
        except StatisticsError:
            mod = "No unique mode"  # Handle cases where there is no single mode

        return avg, med, mod
    except Exception as e:
        return str(e)

# Usage Example
if __name__ == "__main__":
    nums = [4, 1, 2, 2, 3, 5, 4, 4]
    avg, med, mod = compute_statistics(nums)

    print(f"Numbers: {nums}")
    print(f"Mean: {avg}")
    print(f"Median: {med}")
    print(f"Mode: {mod}")
```

Key Points:

1. **Mean**: The average value of the list (`sum / count`).
2. **Median**: The middle value when the list is sorted. For even-length lists, it is the average of the two middle numbers.

3. **Mode**: The value that appears most frequently. If multiple values appear equally often, `StatisticsError` is raised, which is handled in this program.
4. **Robust Handling**: The program handles the case where there is no unique mode gracefully.

Example Output:

Numbers: [4, 1, 2, 2, 3, 5, 4, 4]
Mean: 3.125
Median: 3.5
Mode: 4

This program provides a clean, maintainable, and easy-to-understand approach to compute the basic statistical metrics in Python.

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]

Answer: To compute the **covariance** and **correlation coefficient** between two datasets, we follow a step-by-step approach. Since the actual datasets are not provided in the query, I will outline the method and formula so you can apply it once you have the data.

Step 1: Define the datasets

Let the two datasets be:

$$X=\{x1,x2,\ldots,xn\},Y=\{y1,y2,\ldots,yn\}$$

X={x

1

,x

2

,...,x

n

},Y={y

1

,y

2

,...,y

n

}

where
n
n is the number of data points.

Step 2: Compute the means

The mean of each dataset is:

$$\bar{X}=\frac{1}{n}\sum_{i=1}^{n}x_i,\bar{Y}=\frac{1}{n}\sum_{i=1}^{n}y_i$$

X

-

=

n

1

i=1

$$\sum_{i=1}^{n} x_i, \quad \bar{Y} = \frac{1}{n} \sum_{n} y_i$$

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]

Answer:A boxplot visually represents the distribution, median, quartiles, and potential outliers of a numeric dataset. Below is a complete Python script using `matplotlib` and `numpy` to create the boxplot and detect outliers.

```
# Python
# Import necessary libraries
import matplotlib.pyplot as plt
import numpy as np

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

# Function to identify outliers using the IQR method
def find_outliers(data):
    Q1 = np.percentile(data, 25)  # First quartile
    Q3 = np.percentile(data, 75)  # Third quartile
    IQR = Q3 - Q1                 # Interquartile range
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = [x for x in data if x < lower_bound or x > upper_bound]
    return outliers

# Detect outliers
outliers = find_outliers(data)

# Plot the boxplot
plt.figure(figsize=(8, 6))
plt.boxplot(data, vert=True, patch_artist=True)
plt.title("Boxplot for Numeric Data")
plt.ylabel("Values")
plt.grid(True)
plt.show()

# Print identified outliers
print("Identified outliers:", outliers)
```

Explanation of the Result

1. **Boxplot Components:**
    - The **box** shows the interquartile range (IQR), which contains the middle 50% of the data.
    - The **line inside the box** represents the median.
    - The **whiskers** extend to the lowest and highest values within 1.5 * IQR from the quartiles.
    - Points outside this range are considered **outliers**.
2. **Outliers Identification:**

- The `find_outliers` function calculates Q1, Q3, and IQR. Then it finds values lower than `Q1 - 1.5*IQR` or greater than `Q3 + 1.5*IQR`.
- For the given dataset, the likely outlier is `35`, which is significantly higher than the other values.

3. **Interpretation:**
   - Most data lies between 14 and 29.
   - The extreme value `35` may indicate an unusual observation that could require further analysis.

Usage

This script can be adapted for any numeric dataset. It provides both a visual summary through a boxplot and explicit identification of outlier values using the standard IQR method.

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: 1. Understanding Business Goals and Key Metrics

Before analyzing, clarify what the marketing team wants to achieve:

- Increase conversion rates and sales.
- Improve customer acquisition and retention.
- Optimize advertising ROI (Return on Investment).

Key metrics include:

- **Customer metrics:** Purchase frequency, lifetime value, churn rate.

- **Marketing metrics:** Click-through rate (CTR), cost per acquisition (CPA), return on ad spend (ROAS).
- **Website metrics:** Bounce rate, time on site, product page views.

2. Data Collection and Preparation

Gather data from various sources:

- **Transactional data:** Orders, payments, products purchased.
- **Customer data:** Demographics, preferences, past interactions.
- **Website & App analytics:** Google Analytics, session behavior, click patterns.
- **Marketing campaign data:** Email campaigns, social media ads, paid search data.

Clean the data by removing duplicates, handling missing values, and standardizing formats for meaningful analysis.

3. Customer Segmentation

Use segmentation techniques to identify target audiences:

- **RFM analysis:** Segment customers based on Recency, Frequency, and Monetary values.
- **Behavioral segmentation:** Segment by browsing and purchase behavior.
- **Demographic segmentation:** Age, gender, location, and other attributes.

Segmentation helps tailor marketing campaigns and offers to the right audience.

4. Marketing Campaign Analysis

- **A/B Testing:** Compare different campaigns or website elements to identify most effective versions.
- **Attribution Analysis:** Determine which marketing channels contribute most to conversions.
- **Customer Journey Analysis:** Track touchpoints to understand how users move from awareness to purchase.

5. Predictive Analytics

- **Purchase Prediction:** Use machine learning to forecast which customers are likely to buy which products.

- **Churn Prediction:** Identify customers at risk of leaving and target retention campaigns.
- **Recommendation Systems:** Personalize product recommendations to increase cross-sell and upsell opportunities.

6. Visualization and Reporting

- Use dashboards and reports for continuous monitoring:
  - **Tools:** Tableau, Power BI, Looker.
  - Show key KPIs, marketing performance trends, and segments insights.
- Visualizations help non-technical marketing teams understand the insights and act accordingly.

7. Iterative Optimization

- Continuously test, measure, and optimize campaigns based on data-driven insights.
- Refine targeting, messaging, and channel allocation over time to maximize marketing ROI.

Example Workflow:

```
# Python example for customer segmentation using RFM
import pandas as pd

# Sample transactional data
data = pd.DataFrame({
   'customer_id': [1,2,3,1,2,3],
   'order_date':
pd.to_datetime(['2025-09-01','2025-08-20','2025-08-25','2025-09-15','2025-09-10','2025-09-05']),
   'amount': [100, 150, 80, 200, 120, 60]
})

# Compute Recency, Frequency, Monetary
import datetime
snapshot_date = datetime.datetime(2025,10,2)
rfm = data.groupby('customer_id').agg({
   'order_date': lambda x: (snapshot_date - x.max()).days,
   'order_date': 'count',
   'amount': 'sum'
})
rfm.columns = ['Recency','Frequency','Monetary']
print(rfm)
```

Using this RFM table, marketing can target high-value or at-risk customers.

Summary

Effective e-commerce marketing analysis involves understanding goals, collecting and cleaning multi-source data, segmenting customers, analyzing campaigns, applying predictive analytics, and continuously optimizing strategies. This ensures marketing actions are precise, measurable, and result-driven.

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: To understand the distribution of customer satisfaction survey data, you can use both **summary statistics** and **visualizations**.

Summary Statistics

Some important summary statistics include:

- **Mean**: Average score to understand general customer satisfaction.
- **Median**: Middle score to get a sense of central tendency, less sensitive to outliers than mean.
- **Mode**: Most frequent score, useful for categorical-like ratings.
- **Standard Deviation (SD)**: Measures the spread or variability of scores.
- **Minimum and Maximum**: Identify the range of satisfaction scores.
- **Quartiles / Percentiles**: Show the distribution of data and detect skewness.

Visualizations

- **Histogram**: Shows frequency of each score, clearly visualizing distribution.
- **Boxplot**: Visualizes median, quartiles, and potential outliers.
- **Bar chart**: If scores are treated as categories (1-10), a bar chart of counts can help.

Python Code: Histogram Using Matplotlib

```
# Language: Python

# Necessary imports
import matplotlib.pyplot as plt

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

# Create histogram
plt.hist(survey_scores, bins=range(1, 12), edgecolor='black', align='left')

# Add title and labels
plt.title('Customer Satisfaction Survey Scores Distribution')
plt.xlabel('Survey Score')
plt.ylabel('Frequency')

# Show grid for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Display the plot
plt.show()

# Optional: Calculating summary statistics
import numpy as np
print("Mean:", np.mean(survey_scores))
print("Median:", np.median(survey_scores))
print("Standard Deviation:", np.std(survey_scores, ddof=1))  # Sample SD
print("Mode:", max(set(survey_scores), key=survey_scores.count))
```

Key Points in the Code:

- `bins=range(1, 12)` ensures each integer 1-10 gets its own bin.
- `edgecolor='black'` improves visual separation of bars.
- `align='left'` aligns bars to integers correctly.
- Optional NumPy calculations provide basic summary statistics quickly.

This approach will help your team visualize the survey data, detect patterns, and calculate numerical measures that describe the distribution effectively.