Assignment No 2

Aim

Perform the following operations using R/Python on the data sets:

a) Compute and display summary statistics for each feature available in the dataset. (e.g. minimum value, maximum value, mean, range, standard deviation, variance and percentiles)

b) Illustrate the feature distributions using histogram.

c) Data cleaning, Data integration, Data transformation, Data model building (e.g.

Classification)

Objective

The objective of this assignment is to provide practical experience in performing summary statistics, visualizing feature distributions through histograms, applying data cleaning and transformation techniques, integrating datasets, and finally, constructing a basic classification model. This enhances both analytical and modeling skills using R or Python.

Theoretical

In the data science workflow, understanding the dataset is the first essential step. This is achieved through summary statistics and visualizations. Cleaning and transforming data ensures consistency and completeness. Data integration brings multiple sources into a cohesive structure. Finally, model building—especially classification—applies machine learning to predict categorical outcomes based on feature patterns.

Methods and Explanations of Operations

a) Compute and Display Summary Statistics

- - Summary statistics provide an overview of numerical columns in the dataset.
- - Includes: minimum, maximum, mean, median, range, standard deviation, variance, and percentiles (25th, 50th, 75th).
- - Why Important: Helps understand the distribution, spread, and central tendency of the data.

b) Illustrate Feature Distributions using Histogram

- - Histograms are graphical representations of the distribution of a numerical feature.
- - They divide data into bins and count how many data points fall into each bin.
- - Why Important: Helps to detect skewness, modality, and potential outliers in the data.

c) Data Cleaning, Integration, Transformation, and Model Building

- - Data Cleaning: Handling missing values, duplicates, inconsistencies, and errors.
- - Data Integration: Merging datasets from different sources into a unified dataset.
- - Data Transformation: Standardizing formats, normalizing numerical values, and encoding categorical variables.
- - Model Building (Classification): Applying supervised learning to predict categories (e.g. logistic regression, decision tree).
- - Why Important: These processes ensure high-quality input for modeling and enable meaningful predictions.

Outputs:

In [9]:

```python
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

In [8]:

```python
# Load dataset
file_path ="/content/Mall_Customers.csv"   # Change the path if needed
df = pd.read_csv(file_path)
```

In [10]:

```python
# Display first few rows
print("First 5 rows of dataset:\n", df.head())
```

```
First 5 rows of dataset:
    CustomerID   Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

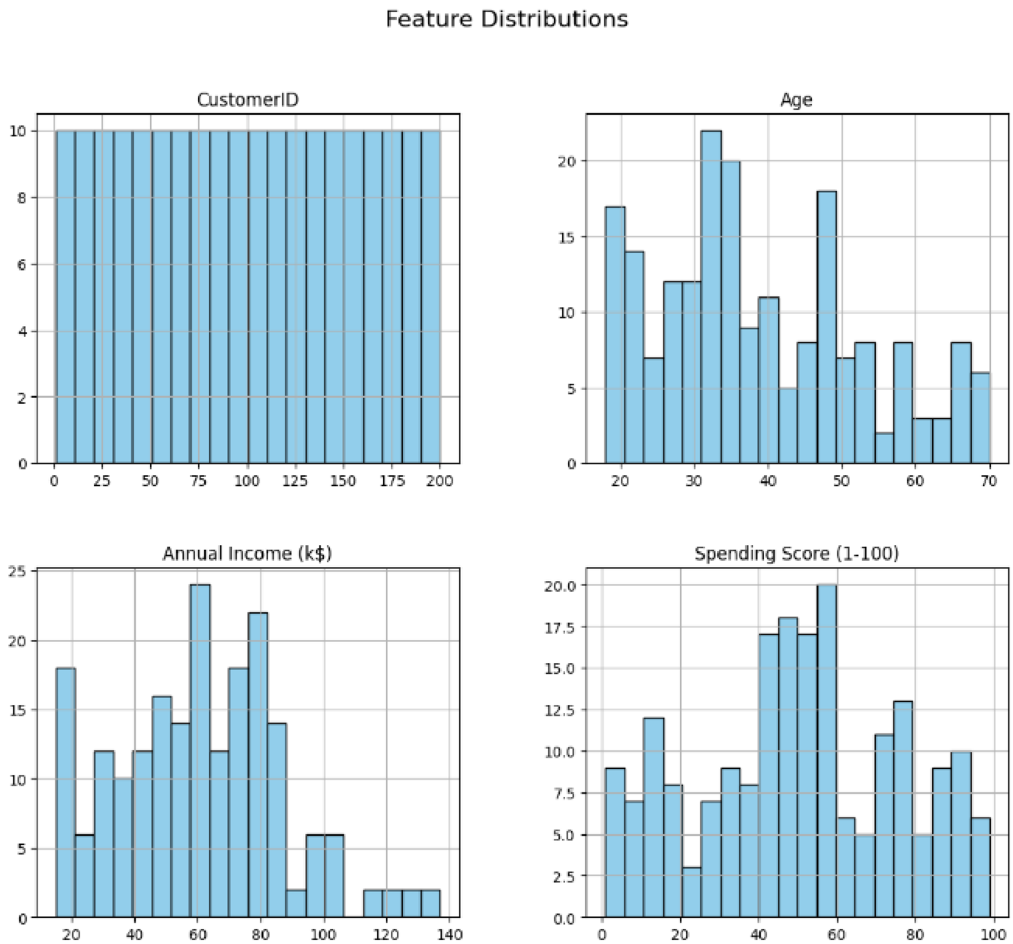
In [11]:

```python
print("\n 🔹 Summary Statistics:")
print(df.describe(include='all'))  # Includes mean, std, min, max, etc.

# Compute additional statistics manually
summary_stats = pd.DataFrame()
summary_stats["Min"] = df.min(numeric_only=True)
summary_stats["Max"] = df.max(numeric_only=True)
summary_stats["Mean"] = df.mean(numeric_only=True)
summary_stats["Range"] = df.max(numeric_only=True) - df.min(numeric_only=True)
summary_stats["Standard Deviation"] = df.std(numeric_only=True)
summary_stats["Variance"] = df.var(numeric_only=True)
summary_stats["25th Percentile"] = df.quantile(0.25, numeric_only=True)
summary_stats["50th Percentile (Median)"] = df.quantile(0.50, numeric_only=True)
summary_stats["75th Percentile"] = df.quantile(0.75, numeric_only=True)

print("\n 🔹 Detailed Summary Statistics:\n", summary_stats)
```

```
🔹 Summary Statistics:
        CustomerID   Genre         Age  Annual Income (k$)  \
count   200.000000     200  200.000000          200.000000
unique         NaN       2         NaN                 NaN
top            NaN  Female         NaN                 NaN
freq           NaN     112         NaN                 NaN
mean    100.500000     NaN   38.850000           60.560000
std      57.879185     NaN   13.969007           26.264721
min       1.000000     NaN   18.000000           15.000000
25%      50.750000     NaN   28.750000           41.500000
50%     100.500000     NaN   36.000000           61.500000
75%     150.250000     NaN   49.000000           78.000000
max     200.000000     NaN   70.000000          137.000000

        Spending Score (1-100)
count               200.000000
unique                     NaN
top                        NaN
freq                       NaN
mean                 50.200000
std                  25.823522
min                   1.000000
25%                  34.750000
50%                  50.000000
75%                  73.000000
max                  99.000000

🔹 Detailed Summary Statistics:
                         Min  Max    Mean  Range  Standard Deviation  \
CustomerID                1  200  100.50    199           57.879185
Age                      18   70   38.85     52           13.969007
Annual Income (k$)       15  137   60.56    122           26.264721
Spending Score (1-100)    1   99   50.20     98           25.823522

                          Variance  25th Percentile  \
CustomerID             3350.000000            50.75
Age                     195.133166            28.75
Annual Income (k$)      689.835578            41.50
Spending Score (1-100)  666.854271            34.75

                        50th Percentile (Median)  75th Percentile
CustomerID                                 100.5           150.25
Age                                         36.0            49.00
Annual Income (k$)                          61.5            78.00
Spending Score (1-100)                      50.0            73.00
```

In [12]:

```
# -------------------- (B) FEATURE DISTRIBUTIONS (HISTOGRAM) --------------------
df.hist(figsize=(12, 10), bins=20, color='skyblue', edgecolor='black')
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()
```



In [13]:

```
# -------------------- (C) DATA CLEANING --------------------
# Check for missing values
print("\n 🔹 Missing Values in Dataset:\n", df.isnull().sum())

# Fill missing values with column mean (for numerical) or mode (for categorical)
for column in df.columns:
    if df[column].dtype == "object":  # Categorical
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:  # Numerical
        df[column].fillna(df[column].mean(), inplace=True)
```

```
 🔹 Missing Values in Dataset:
 CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [15]:

```
### **Step 3: Data Cleaning & Transformation** ###
# Convert 'Genre' (Categorical) into Numeric
le = LabelEncoder()
df["Genre"] = le.fit_transform(df["Genre"])  # Male -> 1, Female -> 0

# Standardization of Numerical Data
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)  # Now df_scaled is defined
```

In [17]:

```
from sklearn.linear_model import LogisticRegression
```

In [18]:

```
### **Step 4: Data Model Building (Classification Example)** ###
# Define Features (X) and Target (Y)
X = df_scaled.drop(columns=["Spending Score (1-100)"])
y = (df_scaled["Spending Score (1-100)"] > df_scaled["Spending Score (1-100)"].median()).astype(int)  # Binary classif

# Split Data into Train & Test (80% Train, 20% Test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Logistic Regression Model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make Predictions
y_pred = model.predict(X_test)

# Evaluate Model
print("\nModel Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Model Accuracy: 0.725

Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.65      0.73        23
           1       0.64      0.82      0.72        17

    accuracy                           0.72        40
   macro avg       0.73      0.74      0.72        40
weighted avg       0.75      0.72      0.73        40
```

In [19]:

```
# Scatter Plot of Spending Score vs Income with Clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df["Annual Income (k$)"], y=df["Spending Score (1-100)"], hue=y, palette="coolwarm")
plt.title("Customer Segments Based on Spending & Income")
plt.show()
```

Conclusion

We have successfully performed descriptive statistics, histogram-based visualizations, data cleaning, integration, and transformation followed by basic classification modeling using R/Python. This assignment has provided a comprehensive understanding of data preparation and how it feeds into effective model building. Through this assignment, I learned how each preprocessing step influences the accuracy and reliability of data analysis and predictions.