Practical No 6

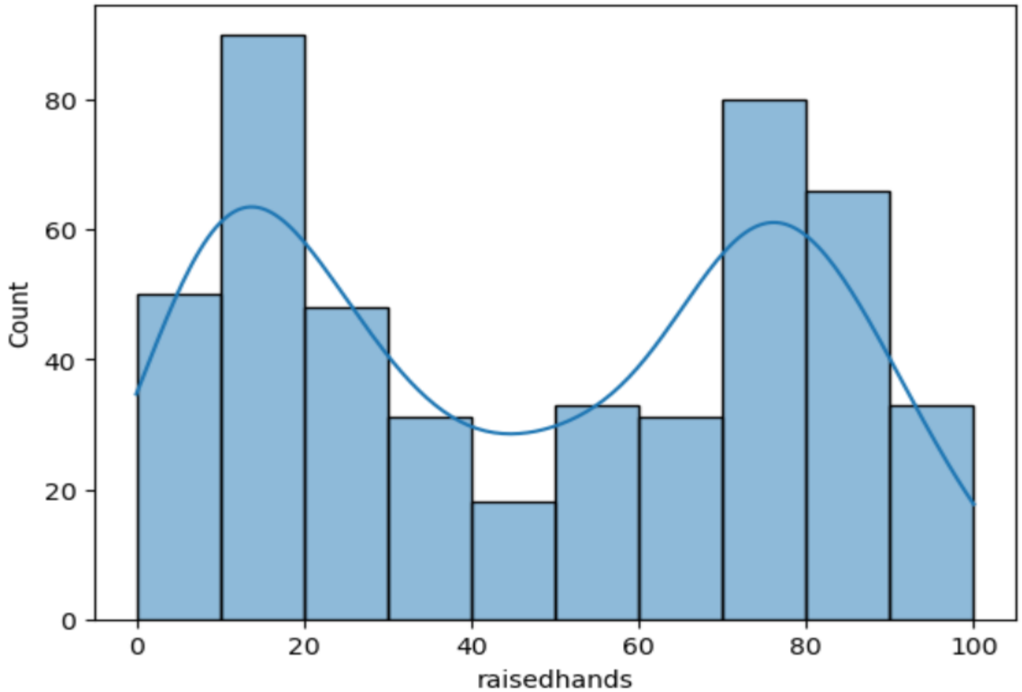
```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read_csv("xAPI-Edu-Data.csv")

print("\n=== First 5 rows of dataset ===")

print(df.head())

# Step 1: Scan Missing Values & Inconsistencies

print("\n=== Missing Values ===")

print(df.isnull().sum())

# Fill numeric missing with median

for col in df.select_dtypes(include=['int64','float64']).columns:

    df[col].fillna(df[col].median(), inplace=True)

# Fill categorical missing with mode

for col in df.select_dtypes(include='object').columns:

    df[col].fillna(df[col].mode()[0], inplace=True)

    df[col] = df[col].str.strip().str.lower()  # fix inconsistencies

print("\n=== After Missing Value Treatment ===")

print(df.isnull().sum())

# Step 2: Outlier Detection & Treatment (IQR Method)

numeric_cols = ['raisedhands', 'VisITedResources', 'AnnouncementsView', 'Discussion']

for col in numeric_cols:

    Q1 = df[col].quantile(0.25)

    Q3 = df[col].quantile(0.75)

    IQR = Q3 - Q1

    lower = Q1 - 1.5 * IQR

    upper = Q3 + 1.5 * IQR
```

```
 # Cap outliers (winsorization)
    df[col] = np.where(df[col] > upper, upper,
                       np.where(df[col] < lower, lower, df[col]))
print("\n=== Outliers handled using IQR capping ===")
# Step 3: Data Transformation
print("\n=== Distribution before transformation (raisedhands) ===")
sns.histplot(df['raisedhands'], kde=True)
plt.show()
df['raisedhands_log'] = np.log1p(df['raisedhands'])
print("\n=== Distribution after log transformation (raisedhands) ===")
sns.histplot(df['raisedhands_log'], kde=True)
plt.show()
print("\n=== Final Dataset Shape ===")
print(df.shape)
```

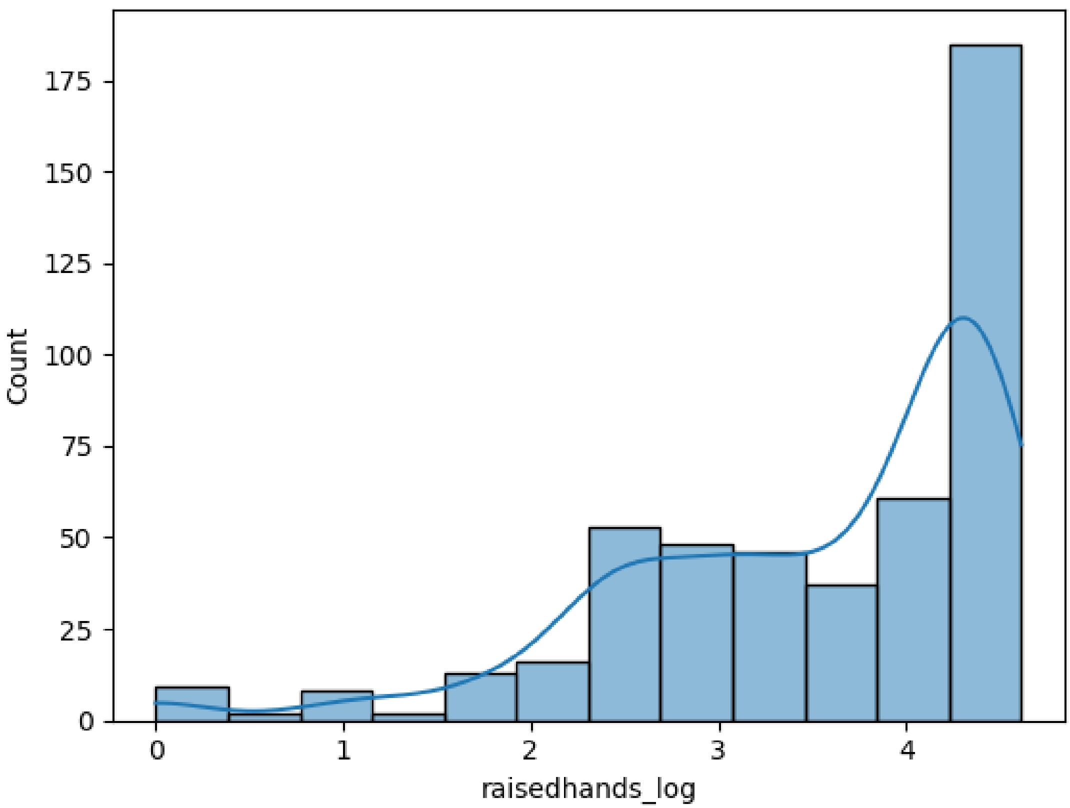
Output:-

```
=== First 5 rows of dataset ===
  gender NationalITy PlaceofBirth     StageID GradeID SectionID Topic  \
0      M          KW       KuwaIT  lowerlevel    G-04         A    IT
1      M          KW       KuwaIT  lowerlevel    G-04         A    IT
2      M          KW       KuwaIT  lowerlevel    G-04         A    IT
3      M          KW       KuwaIT  lowerlevel    G-04         A    IT
4      M          KW       KuwaIT  lowerlevel    G-04         A    IT

  Semester Relation  raisedhands  VisITedResources  AnnouncementsView  \
0        F   Father           15                16                  2
1        F   Father           20                20                  3
2        F   Father           10                 7                  0
3        F   Father           30                25                  5
4        F   Father           40                50                 12

   Discussion ParentAnsweringSurvey ParentschoolSatisfaction  \
0          20                   Yes                     Good
1          25                   Yes                     Good
2          30                    No                      Bad
3          35                    No                      Bad
4          50                    No                      Bad

  StudentAbsenceDays Class
0            Under-7     M
```

```
1            Under-7     M
2            Above-7     L
3            Above-7     L
4            Above-7     M

=== Missing Values ===
gender                      0
NationalITy                 0
PlaceofBirth                0
StageID                     0
GradeID                     0
SectionID                   0
Topic                       0
Semester                    0
Relation                    0
raisedhands                 0
VisITedResources            0
AnnouncementsView           0
Discussion                  0
ParentAnsweringSurvey       0
ParentschoolSatisfaction    0
StudentAbsenceDays          0
Class                       0
dtype: int64

=== After Missing Value Treatment ===
gender                      0
NationalITy                 0
PlaceofBirth                0
StageID                     0
GradeID                     0
SectionID                   0
Topic                       0
Semester                    0
Relation                    0
raisedhands                 0
VisITedResources            0
AnnouncementsView           0
Discussion                  0
ParentAnsweringSurvey       0
ParentschoolSatisfaction    0
StudentAbsenceDays          0
Class                       0
dtype: int64

=== Outliers handled using IQR capping ===

=== Distribution before transformation (raisedhands) ===
```



=== Distribution after log transformation (raisedhands) ===



=== Final Dataset Shape ===
(480, 18)

Practical No 7

```
import pandas as pd
df = pd.read_csv("Bigmart_Sales.csv")
print("=== First 5 rows ===")
print(df.head())
# 2. Pandas Indexing
print("\n=== Column Access ===")
print(df['Item_Identifier'].head())  # single column
print("\n=== Row Access with iloc (first 5 rows, first 3 cols) ===")
print(df.iloc[0:5, 0:3])
print("\n=== Row Access with loc (rows where Outlet_Type = 'Supermarket Type1') ===")
print(df.loc[df['Outlet_Type'] == 'Supermarket Type1'].head())
# 3. Handling Null Values
print("\n=== Null Values Count ===")
print(df.isnull().sum())
# Fill numeric null values with mean
for col in df.select_dtypes(include=['int64','float64']).columns:
    df[col].fillna(df[col].mean(), inplace=True)
# Fill categorical null values with mode
for col in df.select_dtypes(include='object').columns:
    df[col].fillna(df[col].mode()[0], inplace=True)
print("\n=== After Filling Null Values ===")
print(df.isnull().sum())
# Drop rows if still any null values remain
df.dropna(inplace=True)
# 4. Merging, Joining, Concatenating
df1 = df[['Item_Identifier', 'Item_Weight', 'Item_MRP']].head(10)
df2 = df[['Item_Identifier', 'Outlet_Identifier', 'Item_Outlet_Sales']].head(10)
```

```
# --- Merge (on common column Item_Identifier)

merged = pd.merge(df1, df2, on="Item_Identifier", how="inner")

print("\n=== Merged DataFrame ===")

print(merged)

# --- Join (need indexes)

df3 = df1.set_index("Item_Identifier")

df4 = df2.set_index("Item_Identifier")

joined = df3.join(df4, how="inner")

print("\n=== Joined DataFrame ===")

print(joined)

# --- Concatenate (stacking vertically)

concat_df = pd.concat([df1, df2], axis=0)

print("\n=== Concatenated DataFrame (vertical) ===")

print(concat_df.head(15))
```

Output:-

```
=== First 5 rows ===
  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
0           FDA15         9.30          Low Fat         0.016047   
1           DRC01         5.92          Regular         0.019278   
2           FDN15        17.50          Low Fat         0.016760   
3           FDX07        19.20          Regular         0.000000   
4           NCD19         8.93          Low Fat         0.000000   

              Item_Type  Item_MRP Outlet_Identifier  \
0                 Dairy  249.8092            OUT049   
1           Soft Drinks   48.2692            OUT018   
2                  Meat  141.6180            OUT049   
3  Fruits and Vegetables  182.0950            OUT010   
4             Household   53.8614            OUT013   

   Outlet_Establishment_Year Outlet_Size Outlet_Location_Type  \
0                       1999      Medium               Tier 1   
1                       2009      Medium               Tier 3   
2                       1999      Medium               Tier 1   
3                       1998         NaN               Tier 3   
4                       1987        High               Tier 3   
```

```
        Outlet_Type  Item_Outlet_Sales
0  Supermarket Type1          3735.1380
1  Supermarket Type2           443.4228
2  Supermarket Type1          2097.2700
3      Grocery Store           732.3800
4  Supermarket Type1           994.7052

=== Column Access ===
0    FDA15
1    DRC01
2    FDN15
3    FDX07
4    NCD19
Name: Item_Identifier, dtype: object

=== Row Access with iloc (first 5 rows, first 3 cols) ===
  Item_Identifier  Item_Weight Item_Fat_Content
0           FDA15         9.30          Low Fat
1           DRC01         5.92          Regular
2           FDN15        17.50          Low Fat
3           FDX07        19.20          Regular
4           NCD19         8.93          Low Fat

=== Row Access with loc (rows where Outlet_Type = 'Supermarket Type1') ===
  Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
0           FDA15         9.30          Low Fat         0.016047
2           FDN15        17.50          Low Fat         0.016760
4           NCD19         8.93          Low Fat         0.000000
6           FDO10        13.65          Regular         0.012741
8           FDH17        16.20          Regular         0.016687

      Item_Type  Item_MRP Outlet_Identifier  Outlet_Establishment_Year  \
0         Dairy  249.8092           OUT049                       1999
2          Meat  141.6180           OUT049                       1999
4     Household   53.8614           OUT013                       1987
6   Snack Foods   57.6588           OUT013                       1987
8  Frozen Foods   96.9726           OUT045                       2002

  Outlet_Size Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales
0      Medium               Tier 1  Supermarket Type1          3735.1380
2      Medium               Tier 1  Supermarket Type1          2097.2700
4        High               Tier 3  Supermarket Type1           994.7052
6        High               Tier 3  Supermarket Type1           343.5528
8         NaN               Tier 2  Supermarket Type1          1076.5986

=== Null Values Count ===
Item_Identifier                 0
Item_Weight                  1463
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  2410
```

```
Outlet_Location_Type           0
Outlet_Type                    0
Item_Outlet_Sales              0
dtype: int64

=== After Filling Null Values ===
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64

=== Merged DataFrame ===
  Item_Identifier  Item_Weight  Item_MRP Outlet_Identifier  Item_Outlet_Sales
0           FDA15     9.300000  249.8092            OUT049          3735.1380
1           DRC01     5.920000   48.2692            OUT018           443.4228
2           FDN15    17.500000  141.6180            OUT049          2097.2700
3           FDX07    19.200000  182.0950            OUT010           732.3800
4           NCD19     8.930000   53.8614            OUT013           994.7052
5           FDP36    10.395000   51.4008            OUT018           556.6088
6           FDO10    13.650000   57.6588            OUT013           343.5528
7           FDP10    12.857645  107.7622            OUT027          4022.7636
8           FDH17    16.200000   96.9726            OUT045          1076.5986
9           FDU28    19.200000  187.8214            OUT017          4710.5350

=== Joined DataFrame ===
                 Item_Weight  Item_MRP Outlet_Identifier  Item_Outlet_Sales
Item_Identifier
FDA15               9.300000  249.8092            OUT049          3735.1380
DRC01               5.920000   48.2692            OUT018           443.4228
FDN15              17.500000  141.6180            OUT049          2097.2700
FDX07              19.200000  182.0950            OUT010           732.3800
NCD19               8.930000   53.8614            OUT013           994.7052
FDP36              10.395000   51.4008            OUT018           556.6088
FDO10              13.650000   57.6588            OUT013           343.5528
FDP10              12.857645  107.7622            OUT027          4022.7636
FDH17              16.200000   96.9726            OUT045          1076.5986
FDU28              19.200000  187.8214            OUT017          4710.5350

=== Concatenated DataFrame (vertical) ===
  Item_Identifier  Item_Weight  Item_MRP Outlet_Identifier  Item_Outlet_Sales
0           FDA15     9.300000  249.8092               NaN                NaN
1           DRC01     5.920000   48.2692               NaN                NaN
2           FDN15    17.500000  141.6180               NaN                NaN
3           FDX07    19.200000  182.0950               NaN                NaN
4           NCD19     8.930000   53.8614               NaN                NaN
5           FDP36    10.395000   51.4008               NaN                NaN
```

Practical No 8

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder, KBinsDiscretizer
from sklearn.impute import SimpleImputer
# 1. Load Dataset
df = pd.read_csv("Social_Network_Ads.csv")
print("=== First 5 rows ===")
print(df.head())
# 2. Handle Missing Values (Imputation)
print("\n=== Missing Values Count ===")
print(df.isnull().sum())
# Impute numeric with mean
num_imputer = SimpleImputer(strategy="mean")
df[['Age', 'EstimatedSalary']] = num_imputer.fit_transform(df[['Age', 'EstimatedSalary']])
# Impute categorical with mode
cat_imputer = SimpleImputer(strategy="most_frequent")
df[['Gender']] = cat_imputer.fit_transform(df[['Gender']])
print("\n=== After Imputation ===")
print(df.isnull().sum())
# 3. Encoding (Convert Gender to numeric)
le = LabelEncoder()
df['Gender_encoded'] = le.fit_transform(df['Gender'])
print("\n=== Encoding Gender ===")
print(df[['Gender','Gender_encoded']].head())
# 4. Standardization (Z-score scaling)
scaler = StandardScaler()
df[['Age_std', 'Salary_std']] = scaler.fit_transform(df[['Age','EstimatedSalary']])
```

```
print("\n=== After Standardization ===")

print(df[['Age','Age_std','EstimatedSalary','Salary_std']].head())

# 5. Normalization (Min-Max scaling)

minmax = MinMaxScaler()

df[['Age_norm','Salary_norm']] = minmax.fit_transform(df[['Age','EstimatedSalary']])

print("\n=== After Normalization ===")

print(df[['Age','Age_norm','EstimatedSalary','Salary_norm']].head())

# 6. Discretization (Binning Age into categories)

discretizer = KBinsDiscretizer(n_bins=4, encode='ordinal', strategy='uniform')

df['Age_bin'] = discretizer.fit_transform(df[['Age']])

print("\n=== After Discretization (Age Bins) ===")

print(df[['Age','Age_bin']].head(10))

# Final Output

print("\n=== Final Processed Data (first 10 rows) ===")

print(df.head(10))
```

Output:-

```
=== First 5 rows ===
    User ID  Gender  Age  EstimatedSalary  Purchased
0  15624510    Male   19            19000          0
1  15810944    Male   35            20000          0
2  15668575  Female   26            43000          0
3  15603246  Female   27            57000          0
4  15804002    Male   19            76000          0

=== Missing Values Count ===
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64

=== After Imputation ===
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
```

```
dtype: int64

=== Encoding Gender ===
   Gender  Gender_encoded
0    Male               1
1    Male               1
2  Female               0
3  Female               0
4    Male               1

=== After Standardization ===
    Age   Age_std  EstimatedSalary  Salary_std
0  19.0 -1.781797          19000.0   -1.490046
1  35.0 -0.253587          20000.0   -1.460681
2  26.0 -1.113206          43000.0   -0.785290
3  27.0 -1.017692          57000.0   -0.374182
4  19.0 -1.781797          76000.0    0.183751

=== After Normalization ===
    Age  Age_norm  EstimatedSalary  Salary_norm
0  19.0  0.023810          19000.0     0.029630
1  35.0  0.404762          20000.0     0.037037
2  26.0  0.190476          43000.0     0.207407
3  27.0  0.214286          57000.0     0.311111
4  19.0  0.023810          76000.0     0.451852

=== After Discretization (Age Bins) ===
    Age  Age_bin
0  19.0      0.0
1  35.0      1.0
2  26.0      0.0
3  27.0      0.0
4  19.0      0.0
5  27.0      0.0
6  27.0      0.0
7  32.0      1.0
8  25.0      0.0
9  35.0      1.0


=== Final Processed Data (first 10 rows) ===


    User ID  Gender   Age  EstimatedSalary  Purchased  Gender_encoded  \
0  15624510    Male  19.0          19000.0          0               1
1  15810944    Male  35.0          20000.0          0               1
2  15668575  Female  26.0          43000.0          0               0
3  15603246  Female  27.0          57000.0          0               0
4  15804002    Male  19.0          76000.0          0               1
5  15728773    Male  27.0          58000.0          0               1
6  15598044  Female  27.0          84000.0          0               0
7  15694829  Female  32.0         150000.0          1               0
8  15600575    Male  25.0          33000.0          0               1
9  15727311  Female  35.0          65000.0          0               0
```

	Age_std	Salary_std	Age_norm	Salary_norm	Age_bin
0	-1.781797	-1.490046	0.023810	0.029630	0.0
1	-0.253587	-1.460681	0.404762	0.037037	1.0
2	-1.113206	-0.785290	0.190476	0.207407	0.0
3	-1.017692	-0.374182	0.214286	0.311111	0.0
4	-1.781797	0.183751	0.023810	0.451852	0.0
5	-1.017692	-0.344817	0.214286	0.318519	0.0
6	-1.017692	0.418669	0.214286	0.511111	0.0
7	-0.540127	2.356750	0.333333	1.000000	1.0
8	-1.208719	-1.078938	0.166667	0.133333	0.0
9	-0.253587	-0.139263	0.404762	0.370370	1.0

Practical No 9

```
import pandas as pd

import numpy as np

from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_regression, RFE

from sklearn.decomposition import PCA

from sklearn.linear_model import LinearRegression

# 1. Load Dataset

df = pd.read_csv("auto-mpg.csv")

print("=== First 5 rows ===")

print(df.head())

# 2. Handle Missing Values

df.replace('?', np.nan, inplace=True)

# Convert numeric columns to proper dtype

numeric_cols = ['mpg','cylinders','displacement','horsepower','weight','acceleration','model year','origin']

for col in numeric_cols:

    df[col] = pd.to_numeric(df[col], errors='coerce')

# Fill missing numeric values with mean

df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())

print("\n=== Missing values after imputation ===")

print(df.isnull().sum())

# 3. Separate Features and Target

X = df.drop(['mpg','car name'], axis=1)

y = df['mpg']

# Ensure all features are numeric

X = X.apply(pd.to_numeric)

# 4. Variance Threshold

var_thresh = VarianceThreshold(threshold=0.1)

X_var = var_thresh.fit_transform(X)
```

```
print("\n=== Variance Threshold Reduction ===")
print("Original features:", X.shape[1])
print("Reduced features:", X_var.shape[1])
# 5. Univariate Feature Selection (SelectKBest)
selector = SelectKBest(score_func=f_regression, k=5)
X_kbest = selector.fit_transform(X, y)
selected_cols = X.columns[selector.get_support()]
print("\n=== Univariate Feature Selection (Top 5) ===")
print(selected_cols)
# 6. Recursive Feature Elimination (RFE)
model = LinearRegression()
rfe = RFE(model, n_features_to_select=5)
X_rfe = rfe.fit_transform(X, y)
rfe_cols = X.columns[rfe.support_]
print("\n=== RFE Selected Features ===")
print(rfe_cols)
# 7. PCA (Principal Component Analysis)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
print("\n=== PCA Components (first 5 rows) ===")
print(X_pca[:5])
# 8. Correlation Analysis
# Only numeric columns for correlation
corr_matrix = df[numeric_cols].corr()
print("\n=== Correlation Matrix ===")
print(corr_matrix['mpg'].sort_values(ascending=False))
```

Output:-

```
=== First 5 rows ===

    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino

=== Missing values after imputation ===
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64

=== Variance Threshold Reduction ===
Original features: 7
Reduced features: 7
=== Univariate Feature Selection (Top 5) ===
Index(['cylinders', 'displacement', 'horsepower', 'weight', 'model year'], dtype='obj
ect')
=== RFE Selected Features ===
Index(['cylinders', 'horsepower', 'acceleration', 'model year', 'origin'], dtype='obj
ect')

=== PCA Components (first 5 rows) ===
[[543.65893755  51.0019395 ]
 [737.54386405  79.3495761 ]
 [478.18600654  75.61166505]
 [473.6069335   62.69978868]
 [488.87747985  55.94417034]]

=== Correlation Matrix ===
mpg             1.000000
model year      0.579267
origin          0.563450
acceleration    0.420289
horsepower     -0.771437
cylinders      -0.775396
displacement   -0.804203
weight         -0.831741
Name: mpg, dtype: float64
```