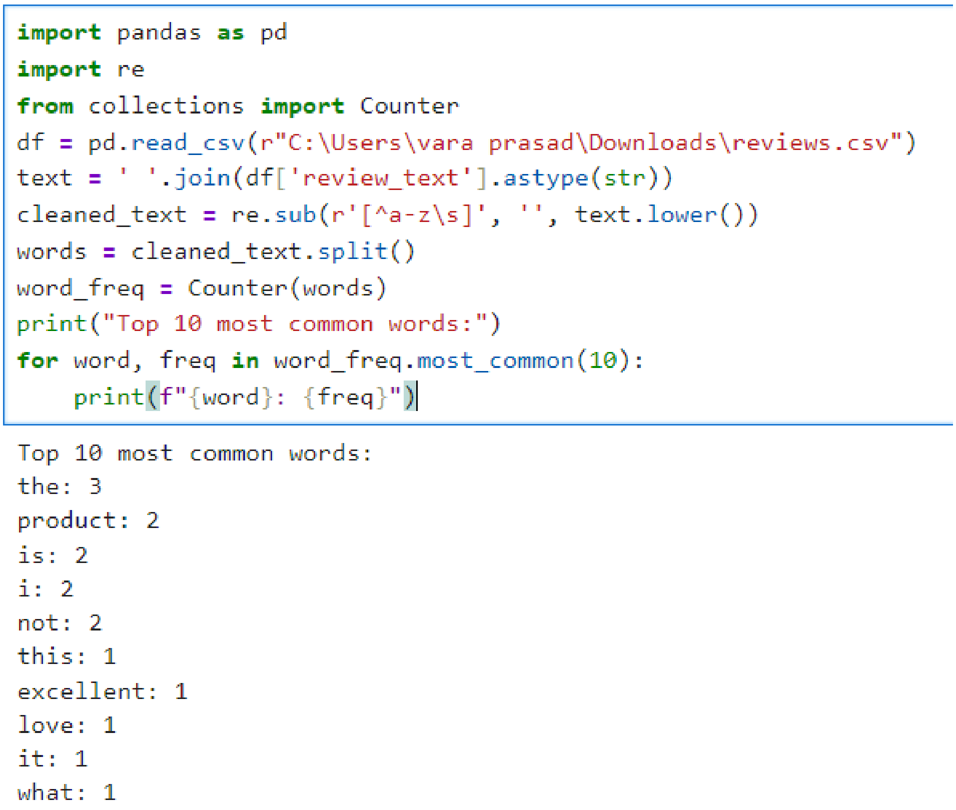
16. Scenario: You are working on a project that involves analyzing customer reviews for a product. You have a dataset containing customer reviews, and your task is to develop a Python program that calculates the frequency distribution of words in the reviews.

Question: Develop a Python program to calculate the frequency distribution of words in the customer reviews dataset?

Code:

```
import pandas as pd

import re

from collections import Counter

df = pd.read_csv(r"C:\Users\vara prasad\Downloads\reviews.csv")

text = ' '.join(df['review_text'].astype(str))

cleaned_text = re.sub(r'[^a-z\s]', '', text.lower())

words = cleaned_text.split()

word_freq = Counter(words)

print("Top 10 most common words:")

for word, freq in word_freq.most_common(10):

    print(f"{word}: {freq}")
```

output:



```python
import pandas as pd
import re
from collections import Counter
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\reviews.csv")
text = ' '.join(df['review_text'].astype(str))
cleaned_text = re.sub(r'[^a-z\s]', '', text.lower())
words = cleaned_text.split()
word_freq = Counter(words)
print("Top 10 most common words:")
for word, freq in word_freq.most_common(10):
    print(f"{word}: {freq}")
```

```
Top 10 most common words:
the: 3
product: 2
is: 2
i: 2
not: 2
this: 1
excellent: 1
love: 1
it: 1
what: 1
```

Dataset:

review_id	review_text			
1	This product is excellent! I love it.			
2	Not what I expected. The quality could be better.			
3	Amazing value for the price. Highly recommend!			
4	The product is okay, but shipping was delayed.			
5	Terrible experience. Will not buy again!			

17. Scenario: You are a data analyst working for a marketing research company. Your team has collected a large dataset containing customer feedback from various social media platforms. The dataset consists of thousands of text entries, and your task is to develop a Python program to analyze the frequency distribution of words in this dataset. Your program should be able to perform the following tasks:

 Load the dataset from a CSV file (data.csv) containing a single column named "feedback" with each row representing a customer comment.

 Preprocess the text data by removing punctuation, converting all text to lowercase, and eliminating any stop words (common words like "the," "and," "is," etc. that don't carry significant meaning).

 Calculate the frequency distribution of words in the preprocessed dataset.

 Display the top N most frequent words and their corresponding frequencies, where N is provided as user input.

 Plot a bar graph to visualize the top N most frequent words and their frequencies.

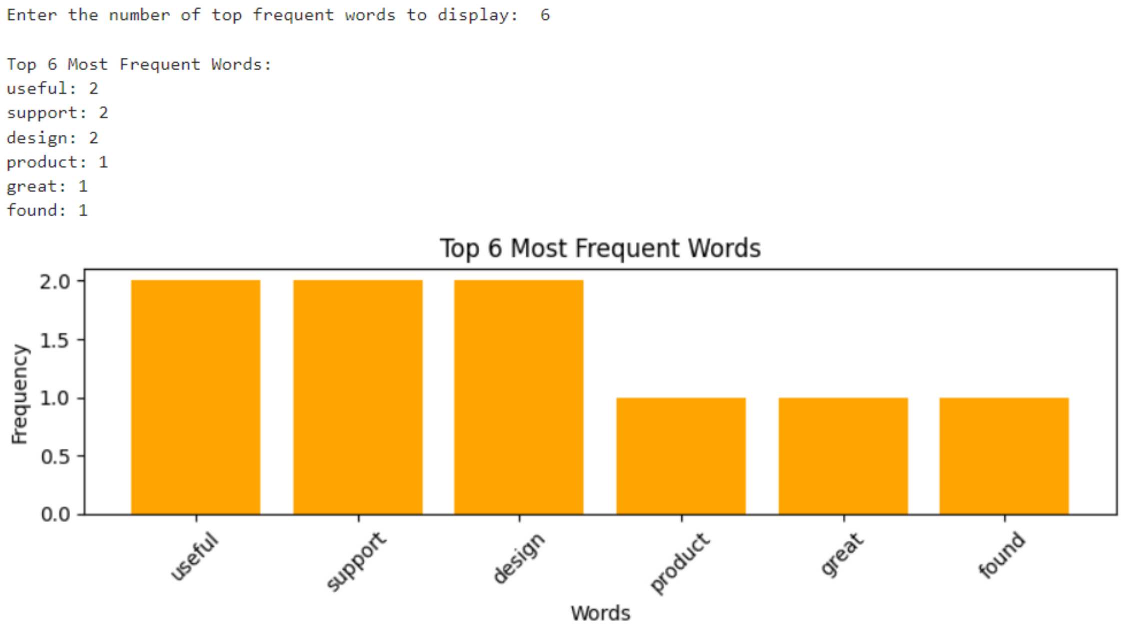
Question: Create a Python program that fulfills these requirements and helps your team gain insights from the customer feedback data.

Code:

```
import pandas as pd
import string
import matplotlib.pyplot as plt
from collections import Counter
import re
import os
stop_words = {
    'the', 'is', 'and', 'in', 'to', 'with', 'a', 'for', 'of', 'on', 'it', 'this',
    'that', 'i', 'was', 'but', 'be', 'have', 'not', 'are', 'as', 'very', 'so', 'from'
}
file_path = os.path.join(r"C:\Users\vara prasad\Downloads\data (1).csv")
def load_data(filepath):
    try:
        df = pd.read_csv(filepath)
        return df['feedback'].dropna().astype(str)
    except Exception as e:
        print("Error loading file:", e)
        return pd.Series()
def preprocess(texts):
    words = []
    for text in texts:
        text = text.lower()
        text = re.sub(f"[{string.punctuation}]", "", text)
        tokens = text.split()
        tokens = [word for word in tokens if word not in stop_words]
        words.extend(tokens)
    return words
def plot_words(word_freq, n):
common = word_freq.most_common(n)
words, counts = zip(*common)
plt.figure(figsize=(8,3 ))
plt.bar(words, counts, color='orange')
plt.title(f"Top {n} Most Frequent Words")
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
def main():
feedback_data = load_data(file_path)
if feedback_data.empty:
print("No feedback data found.")
return
words = preprocess(feedback_data)
word_freq = Counter(words)
try:
n = int(input("Enter the number of top frequent words to display: "))
except ValueError:
print("Please enter a valid number.")
return
```

```
print(f"\nTop {n} Most Frequent Words:")
for word, count in word_freq.most_common(n):
print(f"{word}: {count}")
plot_words(word_freq, n)
if __name__ == "__main__":
main()
```

output:



```
Enter the number of top frequent words to display:  6

Top 6 Most Frequent Words:
useful: 2
support: 2
design: 2
product: 1
great: 1
found: 1
```

Dataset:

feedback

I love this product! Itâ€™s amazing and easy to use.

Terrible service. I will not buy from here again.

Good value for money. Highly recommended!

The product is okay, but delivery was slow.

Excellent customer support. Very helpful and friendly.

Not satisfied with the quality. Could be better.

This is the best purchase I have made this year.

Poor packaging. The item was damaged on arrival.

Great experience overall. Will shop
again!

Average product. Nothing special, but not bad
either.

18. Suppose a hospital tested the age and body fat data for 18 randomly selected adults with the following result.

Question:

Calculate the mean, median and standard deviation of age and %fat using Pandas.

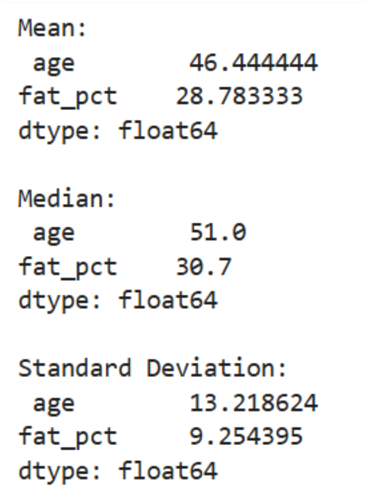
1. Draw the boxplots for age and %fat.
2. Draw a scatter plot and a q-q plot based on these two variables

Code:

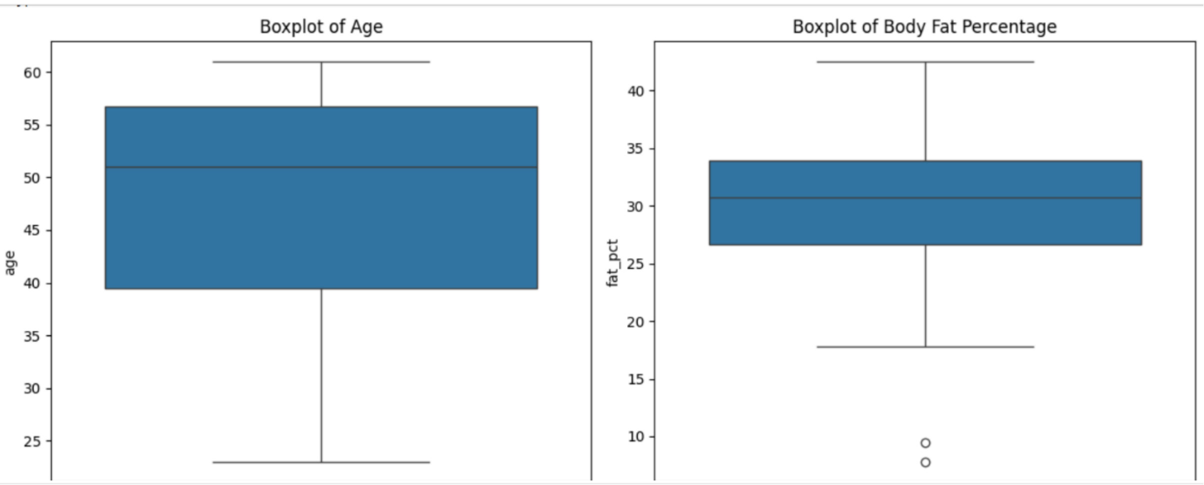
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
csv_path = r"C:\Users\vara prasad\Downloads\age_fat_percentage.csv"
df = pd.read_csv(csv_path)
print("Mean:\n", df.mean())
print("\nMedian:\n", df.median())
print("\nStandard Deviation:\n", df.std())
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Boxplots
sns.boxplot(y=df["age"], ax=axes[0, 0])
axes[0, 0].set_title("Boxplot of Age")

sns.boxplot(y=df["fat_pct"], ax=axes[0, 1])
axes[0, 1].set_title("Boxplot of Body Fat Percentage")
```

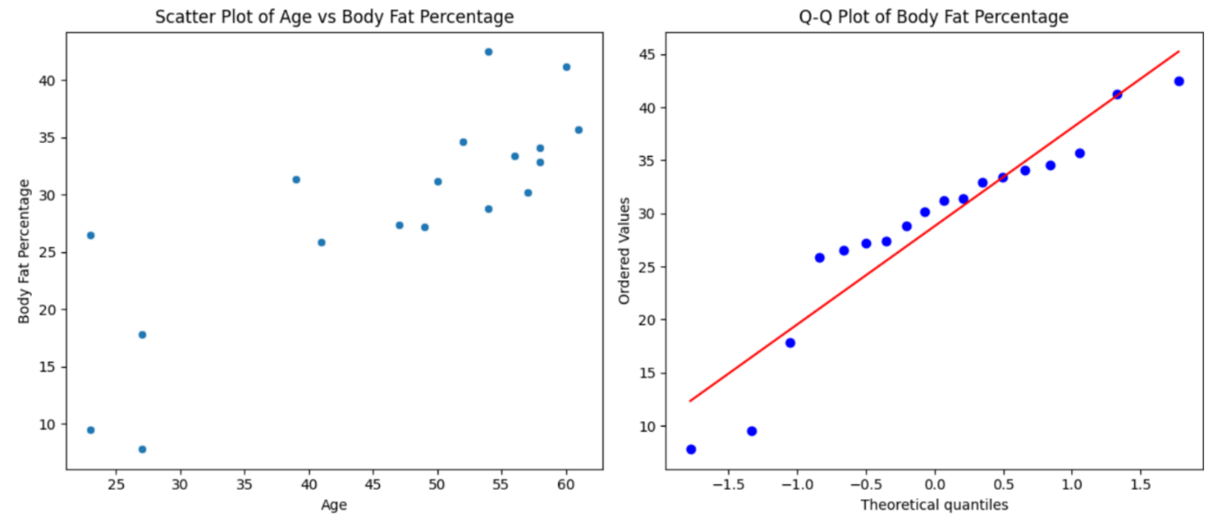
```
# Scatter plot
sns.scatterplot(x=df["age"], y=df["fat_pct"], ax=axes[1, 0])
axes[1, 0].set_title("Scatter Plot of Age vs Body Fat Percentage")
axes[1, 0].set_xlabel("Age")
axes[1, 0].set_ylabel("Body Fat Percentage")

# Q-Q Plot
stats.probplot(df["fat_pct"], dist="norm", plot=axes[1, 1])
axes[1, 1].set_title("Q-Q Plot of Body Fat Percentage")

plt.tight_layout()
plt.show()
```

output:



```
Mean:
 age         46.444444
fat_pct     28.783333
dtype: float64

Median:
 age         51.0
fat_pct     30.7
dtype: float64

Standard Deviation:
 age         13.218624
fat_pct      9.254395
dtype: float64
```





Dataset:

age	fat+A1:B20_pct
23	9.5
23	26.5
27	7.8

27	17.8
39	31.4
41	25.9
47	27.4
49	27.2
50	31.2
52	34.6
54	42.5
54	28.8
56	33.4
57	30.2
58	34.1
58	32.9
60	41.2
61	35.7

19. Scenario:

You are a medical researcher investigating the effectiveness of a new drug in reducing blood pressure. You conduct a clinical trial with a sample of 50 patients who were randomly assigned to receive either the new drug or a placebo. After measuring their blood pressure levels at the end of the trial, you obtain the data for both groups. Now, you want to determine the confidence intervals for the mean reduction in blood pressure for both the drug and placebo groups.

Question:

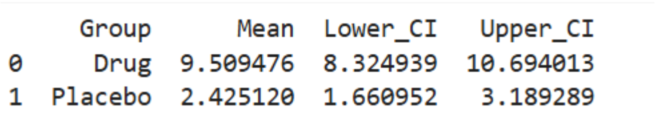
"What is the 95% confidence interval for the mean reduction in blood pressure for patients who received the new drug? Also, what is the 95% confidence interval for the mean reduction in blood pressure for patients who received the placebo?

Code:

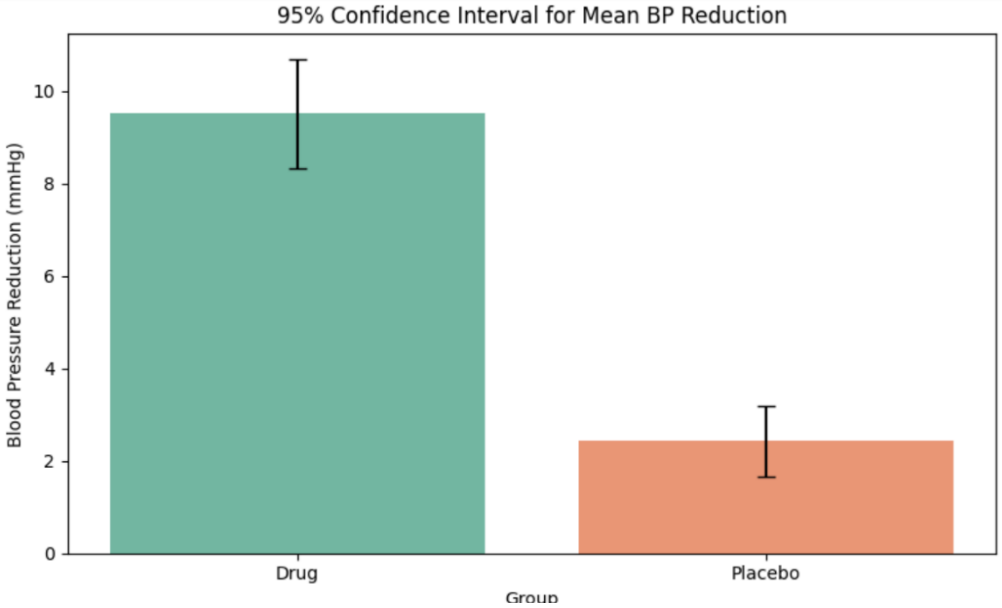
```
import pandas as pd

import numpy as np
```

```
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\blood_pressure_trial.csv")
def confidence_interval(data):
    mean = np.mean(data)
    sem = stats.sem(data)
    margin = sem * stats.t.ppf((1 + 0.95) / 2, len(data) - 1)
    return mean, margin
summary = df.groupby('Group')['BP_Reduction'].apply(confidence_interval).reset_index()
summary[['Mean', 'Margin']] = pd.DataFrame(summary['BP_Reduction'].tolist(),
index=summary.index)
# Plot
plt.figure(figsize=(8, 5))
sns.barplot(data=summary, x='Group', y='Mean', hue='Group', legend=False,
            palette="Set2", capsize=0.1, errorbar=None)
plt.errorbar(x=range(len(summary)), y=summary['Mean'],
             yerr=summary['Margin'], fmt='none', c='black', capsize=5)

plt.title("95% Confidence Interval for Mean BP Reduction")
plt.ylabel("Blood Pressure Reduction (mmHg)")
plt.xlabel("Group")
plt.xticks(ticks=range(len(summary)), labels=summary['Group'])  # Match bar positions
plt.tight_layout()
plt.show()
```

output:



```
     Group      Mean  Lower_CI   Upper_CI
0     Drug  9.509476  8.324939  10.694013
1  Placebo  2.425120  1.660952   3.189289
```



Dataset:

Patient_ID	Group	BP_Reduction
P1	Drug	11.49014
P2	Drug	9.585207
P3	Drug	11.94307
P4	Drug	14.56909
P5	Drug	9.29754
P6	Drug	9.297589
P7	Drug	14.73764
P8	Drug	12.3023
P9	Drug	8.591577
P10	Drug	11.62768
P11	Drug	8.609747
P12	Drug	8.602811
P13	Drug	10.72589
P14	Drug	4.260159
P15	Drug	4.825247
P16	Drug	8.313137
P17	Drug	6.961507
P18	Drug	10.94274

P19	Drug	7.275928
P20	Drug	5.763089
P21	Drug	14.39695
P22	Drug	9.322671
P23	Drug	10.20258
P24	Drug	5.725755
P25	Drug	8.366852
P26	Placebo	3.221845
P27	Placebo	0.698013
P28	Placebo	3.751396
P29	Placebo	1.798723
P30	Placebo	2.416613
P31	Placebo	1.796587
P32	Placebo	6.704556
P33	Placebo	2.973006
P34	Placebo	0.884578
P35	Placebo	4.64509
P36	Placebo	0.558313
P37	Placebo	3.417727
P38	Placebo	-0.91934
P39	Placebo	0.343628
P40	Placebo	3.393722
P41	Placebo	4.476933
P42	Placebo	3.342737
P43	Placebo	2.768703
P44	Placebo	2.397793
P45	Placebo	0.042956
P46	Placebo	1.560312
P47	Placebo	2.078722
P48	Placebo	5.114244
P49	Placebo	3.687237

P50	Placebo	-0.52608

20. Scenario:

You are a data scientist working for an e-commerce company. The marketing team has conducted an A/B test to evaluate the effectiveness of two different website designs (A and B) in terms of conversion rate. They randomly divided the website visitors into two groups, with one group experiencing design A and the other experiencing design B. After a week of data collection, you now have the conversion rate data for both groups. You want to determine whether there is a statistically significant difference in the mean conversion rates between the two website designs.

Question:

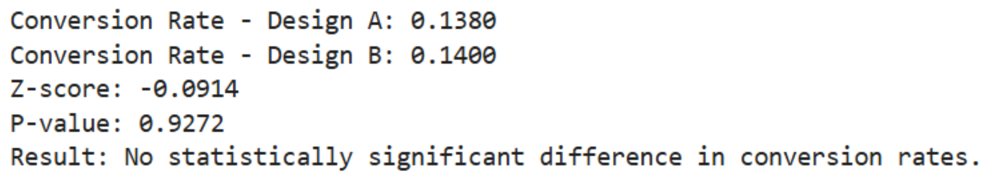
"Based on the data collected from the A/B test, is there a statistically significant difference in the mean conversion rates between website design A and website design B?"

code:

```
import pandas as pd
import numpy as np
from scipy import stats
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\ab_test_conversion_data.csv")
group_A = df[df["Design"] == "A"]["Converted"]
group_B = df[df["Design"] == "B"]["Converted"]
conv_rate_A = group_A.mean()
conv_rate_B = group_B.mean()
print(f"Conversion Rate - Design A: {conv_rate_A:.4f}")
print(f"Conversion Rate - Design B: {conv_rate_B:.4f}")
success_a = group_A.sum()
success_b = group_B.sum()
n_a = group_A.count()
n_b = group_B.count()
p_pool = (success_a + success_b) / (n_a + n_b)
```

```
se = np.sqrt(p_pool * (1 - p_pool) * (1/n_a + 1/n_b))

z_score = (conv_rate_A - conv_rate_B) / se

p_value = 2 * (1 - stats.norm.cdf(abs(z_score)))

print(f"Z-score: {z_score:.4f}")

print(f"P-value: {p_value:.4f}")

alpha = 0.05

if p_value < alpha:

   print("Result: Statistically significant difference in conversion rates.")

else:

   print("Result: No statistically significant difference in conversion rates.")
```

output:



```
Conversion Rate - Design A: 0.1380
Conversion Rate - Design B: 0.1400
Z-score: -0.0914
P-value: 0.9272
Result: No statistically significant difference in conversion rates.
```

Dataset:

Visitor_ID	Design	Converted
V15	A	0
V16	A	0
V17	A	0
V18	A	0
V19	A	0
V20	A	0
V21	A	0
V22	A	0
V23	A	0
V24	A	0
V25	A	0
V26	A	0
V27	A	0
V28	A	0

21.Scenario:

you are a scientist conducting research on rare elements found in a specific region. Your goal is to estimate the average concentration of a rare element in the region using a random sample of measurements. You will use the NumPy library to perform point estimation and calculate confidence intervals for the population mean.The rare element concentration data is stored in a CSV file named "rare_elements.csv," where each row contains a single measurement of the concentration.

Question:

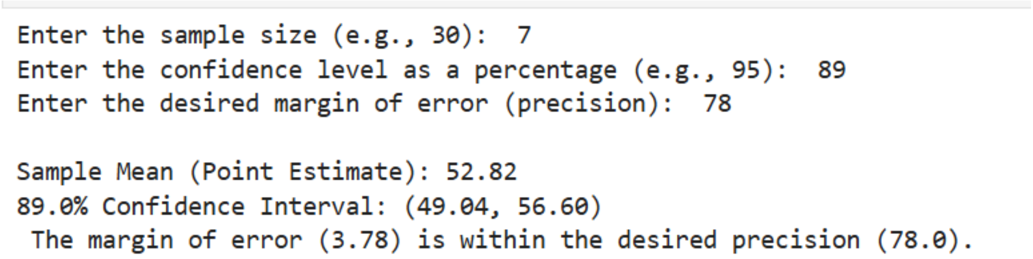
write a Python program that allows the user to input the sample size, confidence level, and desired level of precision.

Code:

```
import pandas as pd
import numpy as np
from scipy import stats
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\rare_elements.csv")
sample_size = int(input("Enter the sample size (e.g., 30): "))
confidence_level = float(input("Enter the confidence level as a percentage (e.g., 95): "))
precision = float(input("Enter the desired margin of error (precision): "))
sample = df["Concentration"].sample(n=sample_size, random_state=1)
sample_mean = np.mean(sample)
std_err = stats.sem(sample)
alpha = 1 - (confidence_level / 100)
margin = std_err * stats.t.ppf(1 - alpha/2, df=sample_size - 1)
lower_bound = sample_mean - margin
upper_bound = sample_mean + margin
print(f"\nSample Mean (Point Estimate): {sample_mean:.2f}")
print(f"{confidence_level:.1f}% Confidence Interval: ({lower_bound:.2f}, {upper_bound:.2f})")
if margin <= precision:
   print(f" The margin of error ({margin:.2f}) is within the desired precision ({precision}).")
else:
```

print(f" The margin of error ({margin:.2f}) exceeds the desired precision ({precision}). Consider increasing sample size.")

output:



```
Enter the sample size (e.g., 30):  7
Enter the confidence level as a percentage (e.g., 95):  89
Enter the desired margin of error (precision):  78

Sample Mean (Point Estimate): 52.82
89.0% Confidence Interval: (49.04, 56.60)
 The margin of error (3.78) is within the desired precision (78.0).
```

Dataset:

Concentration

58.12173

46.94122

47.35914

44.63516

54.32704

38.49231

58.72406

46.19397

22.Scenario:

Imagine you are an analyst for a popular online shopping website. Your task is to analyze customer reviews and provide insights on the average rating and customer satisfaction level for a specific product category.

Question:

You will use the pandas library to calculate confidence intervals to estimate the true population mean rating.

You have been provided with a CSV file named "customer_reviews.csv," which contains customer ratings for products in the chosen category.

Code:

import pandas **as** pd

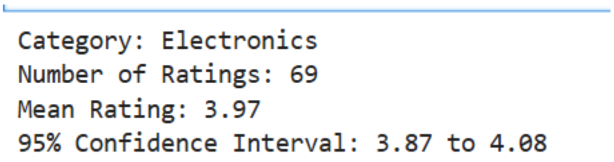
import numpy **as** np

from scipy **import** stats

df = pd.read_csv(r"C:\Users\vara prasad\Downloads\customer_reviews (1).csv")

```
category = 'Electronics'

category_data = df[df['category'] == category]

mean_rating = category_data['rating'].mean()

std_dev = category_data['rating'].std()

n = category_data['rating'].count()

confidence_level = 0.95

alpha = 1 - confidence_level

t_score = stats.t.ppf(1 - alpha/2, df=n-1)

margin_of_error = t_score * (std_dev / np.sqrt(n))

confidence_interval = (mean_rating - margin_of_error, mean_rating + margin_of_error)

print(f"Category: {category}")

print(f"Number of Ratings: {n}")

print(f"Mean Rating: {mean_rating:.2f}")

print(f"95% Confidence Interval: {confidence_interval[0]:.2f} to {confidence_interval[1]:.2f}")
```

output:



```
Category: Electronics
Number of Ratings: 69
Mean Rating: 3.97
95% Confidence Interval: 3.87 to 4.08
```

Dataset:

product_id	category	rating	review
C345	Electronics	3.6	Very satisfied
A123	Home	3.9	Loved it!
C345	Electronics	3.8	Loved it!
C345	Home	4	Not what I expected
A123	Home	5.2	Not what I expected
A123	Beauty	3.1	Very satisfied
C345	Home	4.3	Excellent quality

B234	Beauty	3.2	Great product!

23.Scenario:

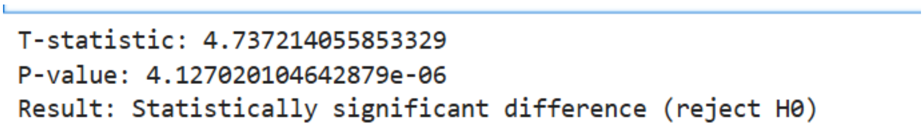
You are a researcher working in a medical lab, investigating the effectiveness of a new treatment for a specific disease. You have collected data from a clinical trial with two groups: a control group receiving a placebo, and a treatment group receiving the new drug.Your goal is to analyze the data using hypothesis testing and calculate the p-value to determine if the new treatment has a statistically significant effect compared to the placebo. You will use the matplotlib library to visualize the data and the p-value.

code:

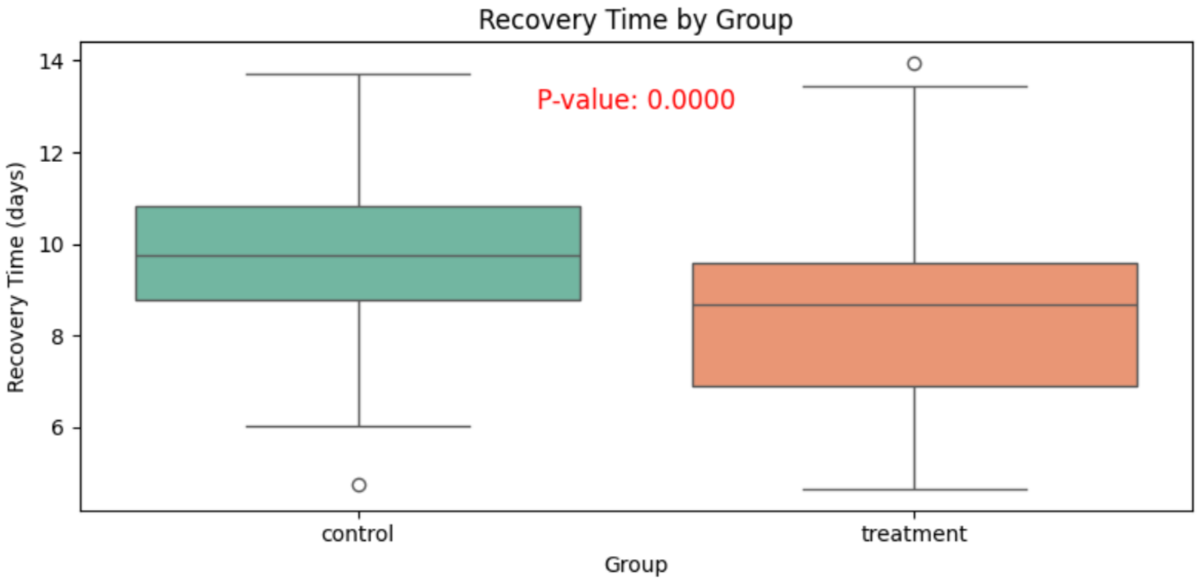
```
import pandas as pd

import scipy.stats as stats

import matplotlib.pyplot as plt

import seaborn as sns


# Load the dataset

df = pd.read_csv(r"C:\Users\vara prasad\Downloads\clinical_trial_data.csv")


# Separate data into control and treatment groups

control = df[df['group'] == 'control']['recovery_time']

treatment = df[df['group'] == 'treatment']['recovery_time']


# Perform an independent t-test

t_stat, p_value = stats.ttest_ind(control, treatment)


# Print hypothesis testing results

print("T-statistic:", t_stat)

print("P-value:", p_value)


# Interpret the result

alpha = 0.05
```

```
if p_value < alpha:
    print("Result: Statistically significant difference (reject H0)")
else:
    print("Result: Not statistically significant (fail to reject H0)")

# Visualization (Future-proofed for Seaborn v0.14.0+)
plt.figure(figsize=(10, 6))
sns.boxplot(x='group', y='recovery_time', hue='group', data=df, palette='Set2', legend=False)
plt.title('Recovery Time by Group')
plt.xlabel('Group')
plt.ylabel('Recovery Time (days)')
plt.text(0.5, max(df['recovery_time']) - 1, f'P-value: {p_value:.4f}',
        ha='center', fontsize=12, color='red')
plt.tight_layout()
plt.show()
```

output:



```
T-statistic: 4.737214055853329
P-value: 4.127020104642879e-06
Result: Statistically significant difference (reject H0)
```



Dataset:

group	recovery_time
control	10.99343
control	9.723471
control	11.29538
control	13.04606
control	9.531693
control	9.531726
control	13.15843
control	11.53487
control	9.061051
control	11.08512
control	9.073165
control	9.06854
control	10.48392
control	6.17344
control	6.550164
control	8.875425
control	7.974338

24.Question: K-Nearest Neighbors (KNN) Classifier

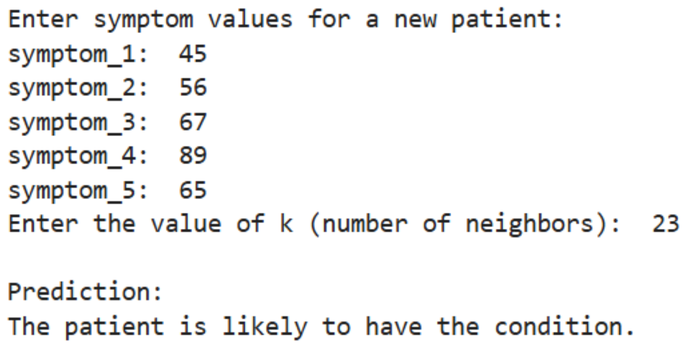
You are working on a classification problem to predict whether a patient has a certain medical condition or not based on their symptoms. You have collected a dataset of patients with labeled data (0 for no condition, 1 for the condition) and various symptom features.

Write a Python program that allows the user to input the features of a new patient and the value of k (number of neighbors). The program should use the KNN classifier from the scikit-learn library to predict whether the patient has the medical condition or not based on the input features.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler

df = pd.read_csv(r"C:\Users\vara prasad\Downloads\medical_condition_data.csv")

X = df.drop('condition', axis=1)

y = df['condition']

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

print("Enter symptom values for a new patient:")

input_features = []

for col in X.columns:

val = float(input(f"{col}: "))

input_features.append(val)

k = int(input("Enter the value of k (number of neighbors): "))

input_df = pd.DataFrame([input_features], columns=X.columns)

input_scaled = scaler.transform(input_df)

knn = KNeighborsClassifier(n_neighbors=k)

knn.fit(X_train, y_train)

prediction = knn.predict(input_scaled)

print("\nPrediction:")

if prediction[0] == 1:

print("The patient is likely to have the condition.")

else:

print("The patient is unlikely to have the condition.")
```

output:



```
Enter symptom values for a new patient:
symptom_1:  45
symptom_2:  56
symptom_3:  67
symptom_4:  89
symptom_5:  65
Enter the value of k (number of neighbors):  23

Prediction:
The patient is likely to have the condition.
```

Dataset:

symptom_1	symptom_2	symptom_3	symptom_4	symptom_5	condition
0.62181	-1.65195	-1.57022	-1.44223	2.085088	0
-0.70434	-1.79983	-1.40846	-0.96551	1.906484	0
-0.0953	0.481188	0.279022	-0.93401	1.924062	1
-0.24124	-0.72872	0.352055	-0.4672	0.131468	1
-1.2803	0.708512	0.872457	0.887858	-0.69771	0
0.05963	1.768863	-0.64694	-0.78038	1.541053	1
1.665474	-1.2281	1.01437	-1.56762	1.658235	0
1.073632	-0.82413	-1.02652	-1.44093	1.199015	1
-1.22213	-1.19404	0.712998	-1.04597	1.235794	0

25.Question 2: Decision Tree for Iris Flower Classification

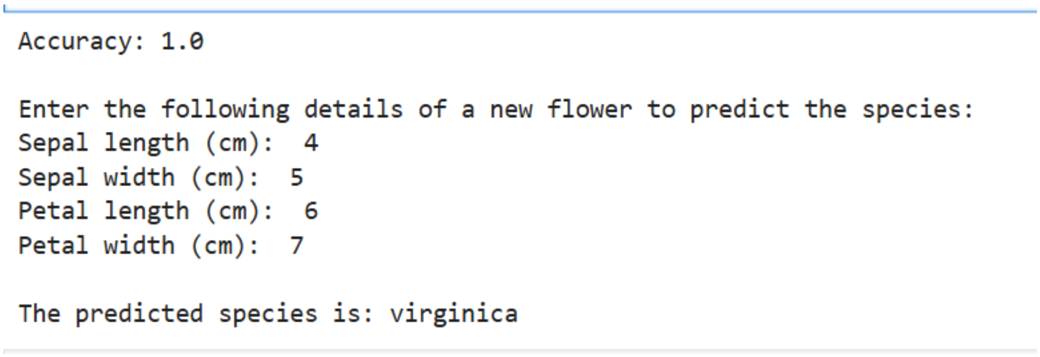
You are analyzing the famous Iris flower dataset to classify iris flowers into three species based on their sepal and petal dimensions. You want to use a Decision Tree classifier to accomplish this task. Write a Python program that loads the Iris dataset from scikit-learn, and allows the user to input the sepal length, sepal width, petal length, and petal width of a new flower. The program should then use the Decision Tree classifier to predict the species of the new flower.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df['species'] = iris.target
iris_df['species'] = iris_df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})
dataset_path = r"C:\Users\vara prasad\Downloads\iris_dataset.csv"
iris_df.to_csv(dataset_path, index=False)
df = pd.read_csv(dataset_path)
X = df.drop('species', axis=1)
y = df['species']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(f"Accuracy: {metrics.accuracy_score(y_test, y_pred)}")
print("\nEnter the following details of a new flower to predict the species:")
sepal_length = float(input("Sepal length (cm): "))
sepal_width = float(input("Sepal width (cm): "))
petal_length = float(input("Petal length (cm): "))
petal_width = float(input("Petal width (cm): "))
new_data = pd.DataFrame([[sepal_length, sepal_width, petal_length, petal_width]],
                    columns=["sepal length (cm)", "sepal width (cm)", "petal length (cm)", "petal width (cm)"])
prediction = model.predict(new_data)
species = prediction[0]
print(f"\nThe predicted species is: {species}")
```

output:



```
Accuracy: 1.0

Enter the following details of a new flower to predict the species:
Sepal length (cm):  4
Sepal width (cm):  5
Petal length (cm):  6
Petal width (cm):  7

The predicted species is: virginica
```

Dataset:

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
6.9	3.1	5.4	2.1	virginica
6.7	3.1	5.6	2.4	virginica
6.9	3.1	5.1	2.3	virginica
5.8	2.7	5.1	1.9	virginica

6.8	3.2	5.9	2.3	virginica
6.7	3.3	5.7	2.5	virginica
6.7	3	5.2	2.3	virginica
6.3	2.5	5	1.9	virginica
6.5	3	5.2	2	virginica
6.2	3.4	5.4	2.3	virginica
5.9	3	5.1	1.8	virginica

26.Question 3: Linear Regression for Housing Price Prediction

You are a real estate analyst trying to predict housing prices based on various features of the houses, such as area, number of bedrooms, and location. You have collected a dataset of houses with their respective prices.

Write a Python program that allows the user to input the features (area, number of bedrooms, etc.) of a new house. The program should use linear regression from scikit-learn to predict the price of the new house based on the input features.

Code :

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
# Load data from Excel
data = pd.read_excel(r"C:\Users\vara prasad\Downloads\26_excel.csv")
# Clean column names safely
data.columns = [str(col).strip() for col in data.columns]
# Check required columns
required_columns = ['Area (sqft)', 'Bedrooms', 'Location', 'Price ($)']
for col in required_columns:
    if col not in data.columns:
        raise ValueError(f"Missing required column: {col}")

# Features and Target

X = data[['Area (sqft)', 'Bedrooms', 'Location']]

y = data['Price ($)']

# Preprocessing for categorical data

preprocessor = ColumnTransformer(

    transformers=[('cat', OneHotEncoder(), ['Location'])],

    remainder='passthrough'

)


# Create pipeline

model = Pipeline([

    ('preprocessor', preprocessor),

    ('regressor', LinearRegression())

])

# Train model

model.fit(X, y)

# Get user input for prediction

area = float(input("Enter area (in sqft): "))

bedrooms = int(input("Enter number of bedrooms: "))

location = input("Enter location (e.g., Downtown, Suburb, Rural): ")

# Create input for prediction

new_house = pd.DataFrame([[area, bedrooms, location]], columns=['Area (sqft)', 'Bedrooms',
'Location'])

# Predict

predicted_price = model.predict(new_house)[0]

print(f"\nPredicted House Price: ${predicted_price:,.2f}")
```

output :

```
Enter area (in sqft):  56

Enter number of bedrooms:  5

Enter location (e.g., Downtown, Suburb, Rural):  Downtown
```

```
Predicted House Price: $153,646.84
```

Dataset :

Area (sqft)	**Bedrooms**	**Location**	**Price ($)**
1400	3	Downtown	300000
1600	3	Suburb	280000
1700	4	Suburb	310000
1875	3	Downtown	340000
1100	2	Rural	190000
1550	3	Suburb	295000
2350	4	Downtown	405000
2450	4	Downtown	430000
1425	3	Rural	220000
1700	3	Suburb	310000

27.Question: Logistic Regression for Customer Churn Prediction

You are working for a telecommunications company, and you want to predict whether a customer will churn (leave the company) based on their usage patterns and demographic data. You have collected a dataset of past customers with their churn status (0 for not churned, 1 for churned) and various features.

Write a Python program that allows the user to input the features (e.g., usage minutes, contract duration) of a new customer. The program should use logistic regression from scikit-learn to predict whether the new customer will churn or not based on the input features.

Code :

```
#27
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Step 1: Load data from Excel
data = pd.read_excel(r"C:\Users\vara prasad\Downloads\27_excel.xlsx)
# Step 2: Strip column names
data.columns = [str(col).strip() for col in data.columns]
# Step 3: Features and Target
X = data[['Usage (mins)', 'Contract (months)', 'Age']]
y = data['Churn']
# Step 4: Preprocess and Split
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Step 5: Train model
model = LogisticRegression()
model.fit(X_scaled, y)
# Step 6: Get user input
usage = float(input("Enter usage in minutes: "))
contract = int(input("Enter contract duration in months: "))
age = int(input("Enter age of customer: "))

# Step 7: Predict
user_input = pd.DataFrame([[usage, contract, age]], columns=['Usage (mins)', 'Contract (months)', 'Age'])
user_input_scaled = scaler.transform(user_input)
prediction = model.predict(user_input_scaled)[0]
prob = model.predict_proba(user_input_scaled)[0][1]
# Step 8: Output result
status = "will churn" if prediction == 1 else "will not churn"
print(f"\nPrediction: The customer {status} (Probability: {prob:.2f})")
```

output :

```
Enter usage in minutes:  53
Enter contract duration in months:  5
Enter age of customer:  20
Prediction: The customer will churn (Probability: 0.93)
```

Dataset :

Usage (mins)	Contract (months)	Age	Churn
300	12	25	0
500	24	45	0
200	6	22	1
450	18	35	0
150	3	30	1
400	12	40	0
100	1	28	1
380	10	33	0
220	4	27	1
350	12	50	0

28.Question: K-Means Clustering for Customer Segmentation

You are working for an e-commerce company and want to segment your customers into distinct groups based on their purchasing behavior. You have collected a dataset of customer data with various shopping-related features.

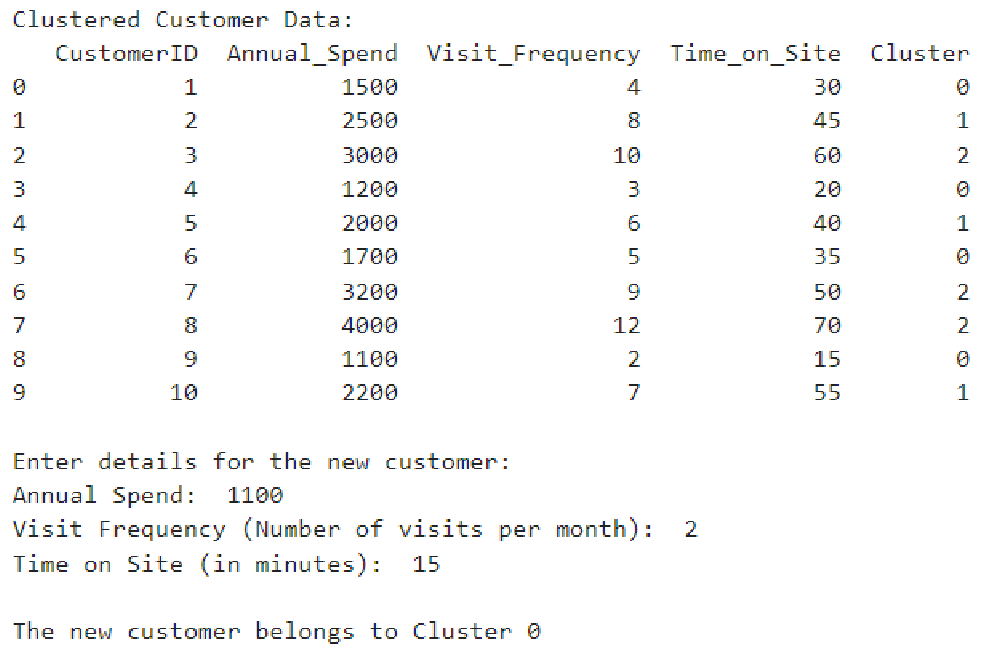
Write a Python program that allows the user to input the shopping-related features of a new customer. The program should use K-Means clustering from scikit-learn to assign the new customer to one of the existing segments based on the input features.

Code;

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
data_path = r"C:\Users\vara prasad\Downloads\customer_data.csv" # Update this path to where you saved the CSV file
df = pd.read_csv(data_path)
```

```
if 'CustomerID' not in df.columns:
df['CustomerID'] = range(1, len(df) + 1)
X = df[['Annual_Spend', 'Visit_Frequency', 'Time_on_Site']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
df['Cluster'] = kmeans.labels_
print("Clustered Customer Data:")
print(df)
def classify_new_customer(annual_spend, visit_freq, time_on_site):
new_customer = pd.DataFrame([[annual_spend, visit_freq, time_on_site]], columns=['Annual_Spend', 'Visit_Frequency', 'Time_on_Site'])
new_customer_scaled = scaler.transform(new_customer)
cluster = kmeans.predict(new_customer_scaled)
return cluster[0]
print("\nEnter details for the new customer:")
annual_spend = float(input("Annual Spend: "))
visit_freq = int(input("Visit Frequency (Number of visits per month): "))
time_on_site = float(input("Time on Site (in minutes): "))
new_customer_cluster = classify_new_customer(annual_spend, visit_freq, time_on_site)
print(f"\nThe new customer belongs to Cluster {new_customer_cluster}")
```

output:



```
Clustered Customer Data:
   CustomerID  Annual_Spend  Visit_Frequency  Time_on_Site  Cluster
0           1          1500                4            30        0
1           2          2500                8            45        1
2           3          3000               10            60        2
3           4          1200                3            20        0
4           5          2000                6            40        1
5           6          1700                5            35        0
6           7          3200                9            50        2
7           8          4000               12            70        2
8           9          1100                2            15        0
9          10          2200                7            55        1

Enter details for the new customer:
Annual Spend:  1100
Visit Frequency (Number of visits per month):  2
Time on Site (in minutes):  15

The new customer belongs to Cluster 0
```

Dataset:

CustomerID	Annual_Spend	Visit_Frequency	Time_on_Site
1	1500	4	30
2	2500	8	45
3	3000	10	60
4	1200	3	20
5	2000	6	40
6	1700	5	35
7	3200	9	50
8	4000	12	70
9	1100	2	15
10	2200	7	55

29.Question: Evaluation Metrics for Model Performance

You have trained a machine learning model on a dataset, and now you want to evaluate its performance using various metrics.

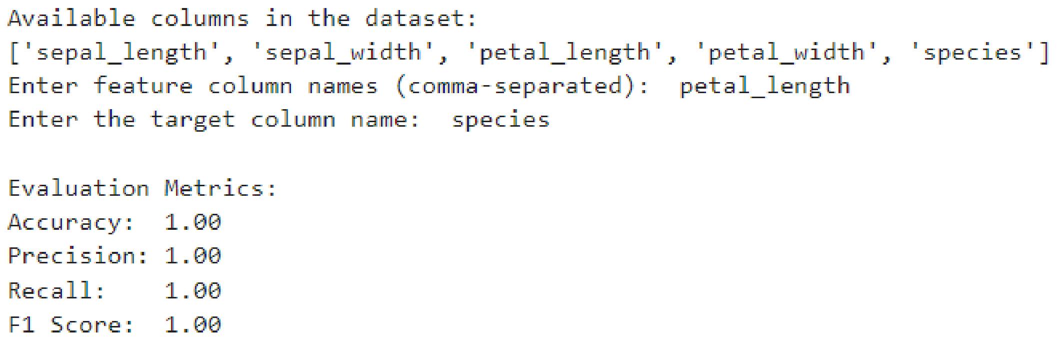
Write a Python program that loads a dataset and trained model from scikit-learn. The program should ask the user to input the names of the features and the target variable they want to use for evaluation. The program should then calculate and display common evaluation metrics such as accuracy, precision, recall, and F1-score for the model's predictions on the test data.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Load dataset
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\iris_sample.csv")

# Show available columns
print("Available columns in the dataset:")
print(df.columns.tolist())

# Get user input
features = input("Enter feature column names (comma-separated): ").split(',')
target = input("Enter the target column name: ")

# Strip extra spaces
features = [col.strip() for col in features]
target = target.strip()

# Encode target if categorical
if df[target].dtype == 'object':
    le = LabelEncoder()
    df[target] = le.fit_transform(df[target])

# Split into features (X) and target (y)
X = df[features]
y = df[target]
```

```
# Split into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# Train model

model = RandomForestClassifier(random_state=42)

model.fit(X_train, y_train)


# Make predictions

y_pred = model.predict(X_test)


# Calculate evaluation metrics

print("\nEvaluation Metrics:")

print(f"Accuracy:  {accuracy_score(y_test, y_pred):.2f}")

print(f"Precision: {precision_score(y_test, y_pred, average='weighted'):.2f}")

print(f"Recall:    {recall_score(y_test, y_pred, average='weighted'):.2f}")

print(f"F1 Score:  {f1_score(y_test, y_pred, average='weighted'):.2f}")
```

output:



```
Available columns in the dataset:
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
Enter feature column names (comma-separated):  petal_length
Enter the target column name:  species

Evaluation Metrics:
Accuracy:  1.00
Precision: 1.00
Recall:    1.00
F1 Score:  1.00
```

Dataset:

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
6.2	2.8	4.8	1.8	virginica
5.9	3	5.1	1.8	virginica

5.5	2.3	4	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5	3.4	1.5	0.2	setosa
6.7	3.1	4.7	1.5	versicolor
6.3	3.3	6	2.5	virginica
5.6	2.8	4.9	2	virginica

30.Question: Classification and Regression Trees (CART) for Car Price Prediction

You are working for a car dealership, and you want to predict the price of used cars based on various features such as the car's mileage, age, brand, and engine type. You have collected a dataset of used cars with their respective prices.

Write a Python program that loads the car dataset and allows the user to input the features of a new car they want to sell. The program should use the Classification and Regression Trees (CART) algorithm from scikit-learn to predict the price of the new car based on the input features.

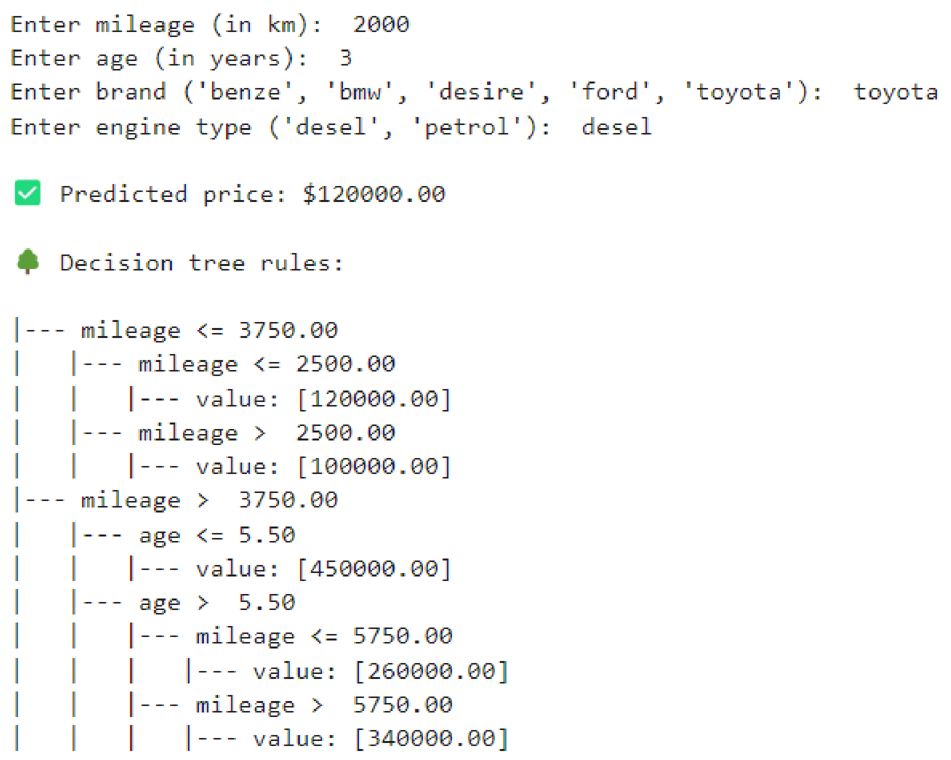
The CART algorithm will create a tree-based model that will split the data into subsets based on the chosen features and their values, leading to a decision path that eventually predicts the price of the car. The program should output the predicted price and display the decision path (the sequence of conditions leading to the prediction) for the new car.

Code:

```
import pandas as pd
from sklearn.tree import DecisionTreeRegressor, export_text
from sklearn.preprocessing import LabelEncoder
import os
import sys

# === File path ===
file_path = r"C:\Users\vara prasad\Downloads\car_data.csv.xlsx"

# === Load dataset ===
if not os.path.exists(file_path):
    print(f"❌ Error: File not found at {file_path}")
    sys.exit()

try:
    df = pd.read_excel(file_path)
except Exception as e:
    print(f"❌ Error reading Excel file: {e}")
    sys.exit()

# === Validate columns ===
required_cols = ['mileage', 'age', 'brand', 'engine', 'price']
if not all(col in df.columns for col in required_cols):
    print(f"❌ Error: Dataset must contain columns: {required_cols}")
    sys.exit()

# === Encode categorical columns ===
label_encoders = {}
for col in ['brand', 'engine']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# === Features and target ===
X = df[['mileage', 'age', 'brand', 'engine']]
y = df['price']

# === Train CART model ===
model = DecisionTreeRegressor(random_state=42)
model.fit(X, y)
```

```
# === User input ===
def get_user_input():
    mileage = float(input("Enter mileage (in km): "))
    age = int(input("Enter age (in years): "))

    brand_options = tuple(label_encoders['brand'].classes_)
    brand = input(f"Enter brand {brand_options}: ")
    while brand not in brand_options:
        brand = input(f"❌ Invalid brand. Enter again {brand_options}: ")

    engine_options = tuple(label_encoders['Engine'].classes_)
    engine = input(f"Enter engine type {engine_options}: ")
    while engine not in engine_options:
        engine = input(f"❌ Invalid engine type. Enter again {engine_options}: ")

    brand_encoded = label_encoders['Brand'].transform([brand])[0]
    engine_encoded = label_encoders['Engine'].transform([engine])[0]

    return pd.DataFrame([[mileage, age, brand_encoded, engine_encoded]],
                        columns=['mileage', 'age', 'brand', 'engine'])

# === Predict and explain ===
user_features = get_user_input()
predicted_price = model.predict(user_features)[0]
print(f"\n✅ Predicted price: ${predicted_price:.2f}")

# === Display decision path ===
tree_rules = export_text(model, feature_names=['mileage', 'age', 'brand', 'engine'])
print("\n🌳 Decision tree rules:\n")
print(tree_rules)
```

output:



```
Enter mileage (in km):  2000
Enter age (in years):  3
Enter brand ('benze', 'bmw', 'desire', 'ford', 'toyota'):  toyota
Enter engine type ('desel', 'petrol'):  desel

✅ Predicted price: $120000.00

🌳 Decision tree rules:

|--- mileage <= 3750.00
|   |--- mileage <= 2500.00
|   |   |--- value: [120000.00]
|   |--- mileage >  2500.00
|   |   |--- value: [100000.00]
|--- mileage >  3750.00
|   |--- age <= 5.50
|   |   |--- value: [450000.00]
|   |--- age >  5.50
|   |   |--- mileage <= 5750.00
|   |   |   |--- value: [260000.00]
|   |   |--- mileage >  5750.00
|   |   |   |--- value: [340000.00]
```

Dataset:

mileage	age	brand	engine	price
2000	3	toyota	petrol	120000
3000	8	ford	petrol	100000
4500	7	bmw	petrol	260000
7000	6	benze	desel	340000
6000	5	desire	desel	450000

31. Scenario: You work as a data scientist for an e-commerce company that sells a wide range of products online. The company collects vast amounts of data about its customers, including their purchase history, browsing behavior, demographics, and more. The marketing team wants to understand their customer base better and improve their targeted marketing strategies. They have asked you to perform customer segmentation using clustering techniques to identify distinct groups of customers with similar characteristics.

Question: Your task is to use Python and clustering algorithms to segment the customers into different groups based on their behavior and characteristics. The marketing team will use these segments to tailor their marketing campaigns and promotions effectively.

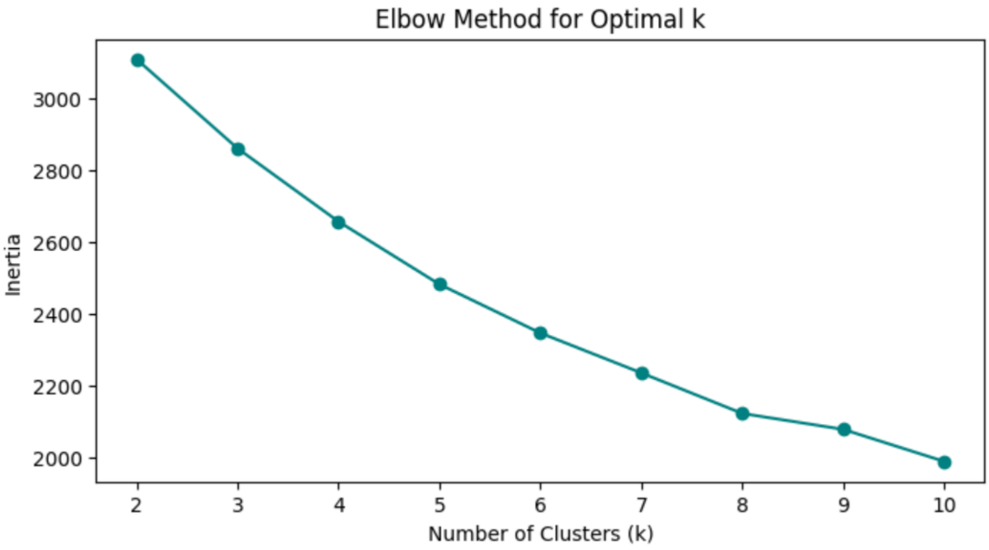
Code:

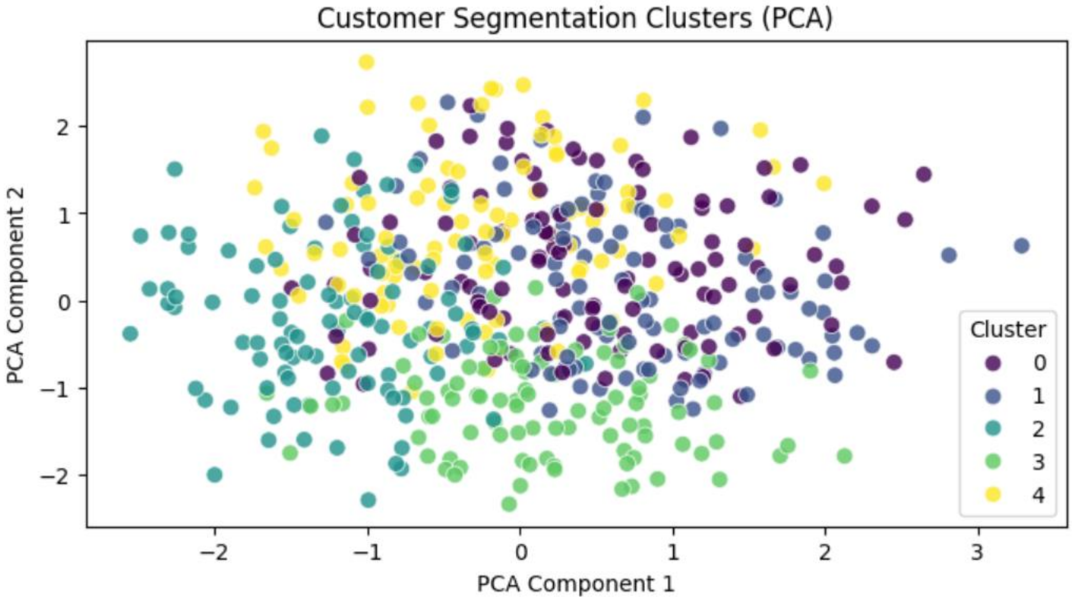
import pandas as pd

```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

file_path =r"C:\Users\jampa\Downloads\customer_segmentation_data.csv"

df = pd.read_csv(file_path)

features = ['Age', 'Annual_Income', 'Spending_Score', 'Online_Sessions_per_Month',
'Average_Order_Value', 'Total_Purchases', 'Customer_Lifetime_Value']

X = df[features]

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

inertia = []

cluster_range = range(2, 11)

for k in cluster_range:

    kmeans = KMeans(n_clusters=k, random_state=42)

    kmeans.fit(X_scaled)

    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8,4))

plt.plot(cluster_range, inertia, marker='o', linestyle='-', color='teal')

plt.title('Elbow Method for Optimal k')

plt.xlabel('Number of Clusters (k)')

plt.ylabel('Inertia')

plt.xticks(cluster_range)

plt.show()

optimal_k = 5

kmeans = KMeans(n_clusters=optimal_k, random_state=42)

kmeans.fit(X_scaled)

df['Cluster'] = kmeans.labels_

pca = PCA(n_components=2)
```

```
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,4))

sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=df['Cluster'], palette='viridis', s=60, alpha=0.8)

plt.title('Customer Segmentation Clusters (PCA)')

plt.xlabel('PCA Component 1')

plt.ylabel('PCA Component 2')

plt.legend(title='Cluster')

plt.show()

df.to_csv('segmented_customers.csv', index=False)

print('Customer segmentation complete. Results saved to segmented_customers.csv.')
```

output:





```
Customer segmentation complete. Results saved to segmented_customers.csv.
```

Dataset:

CustomerID	Age
1	56
2	69
3	46
4	32
5	60
6	25
7	38
8	56
9	36
10	40
11	28
12	28
13	41
14	53
15	57
16	41
17	20

32. Scenario: You work as a data scientist for a real estate company. The company has collected data on various houses, including features such as the size of the house, number of bedrooms, location, and other relevant attributes. The marketing team wants to build a predictive model to estimate the price of houses based on their features. They believe that linear regression modeling can be an effective approach for this task.

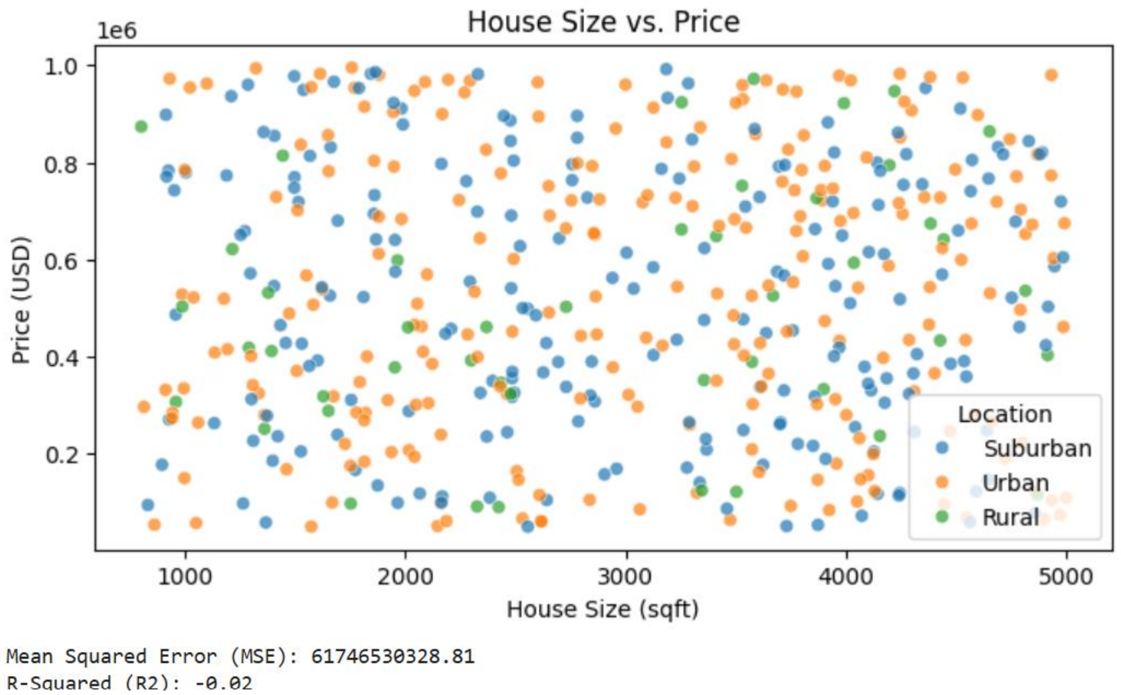
Question:Your task is write a Python program to perform bivariate analysis and build a linear regression model to predict house prices based on a selected feature (e.g., house size) from the dataset. Additionally, you need to evaluate the model's performance to ensure its accuracy and reliability.

Code:

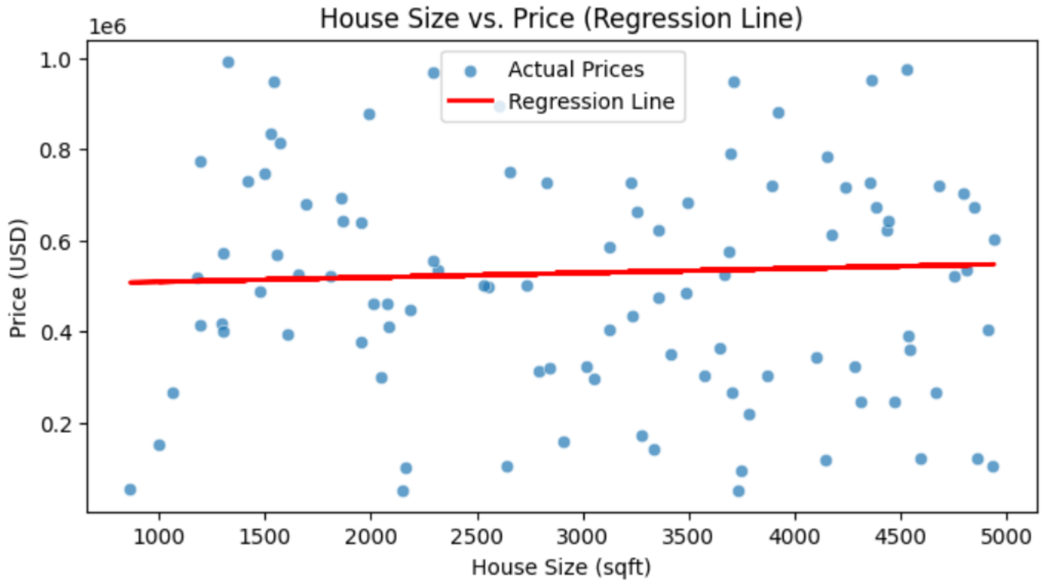
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the house price dataset
file_path =r"C:\Users\jampa\Downloads\house_price_data.csv"
df = pd.read_csv(file_path)

# Perform bivariate analysis (House Size vs. Price)
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='House_Size_sqft', y='Price', hue='Location', alpha=0.7)
plt.title('House Size vs. Price')
plt.xlabel('House Size (sqft)')
plt.ylabel('Price (USD)')
plt.show()

# Prepare data for linear regression
```

```
X = df[['House_Size_sqft']]
y = df['Price']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-Squared (R2): {r2:.2f}")

# Plot the regression line
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_test['House_Size_sqft'], y=y_test, alpha=0.7, label='Actual Prices')
plt.plot(X_test['House_Size_sqft'], y_pred, color='red', linewidth=2, label='Regression Line')
plt.title('House Size vs. Price (Regression Line)')
plt.xlabel('House Size (sqft)')
plt.ylabel('Price (USD)')
plt.legend()
plt.show()
```

output;



```
Mean Squared Error (MSE): 61746530328.81
R-Squared (R2): -0.02
```



Dataset:

HouseID	House_Size_sqft	Bedrooms	Bathrooms	Location	Lot_Size_acres	Year_Built	Price
1	1660	1	4	Suburban	2.32	1958	525593.1
2	4572	3	4	Suburban	1.4	1999	804789.8
3	3892	4	2	Urban	2.12	1955	721732.2
4	1266	2	4	Suburban	1.35	2009	97714.71
5	4244	4	4	Suburban	1.6	1993	119256.7
6	3971	4	2	Urban	0.31	1960	432729.6

7	3719	5	3	Suburban	1.91	2015	330526
8	930	2	4	Suburban	0.41	1996	270765.1
9	2485	4	4	Suburban	2.08	2022	316954.3
10	1569	4	4	Suburban	1.98	1952	813308.6
11	3191	2	1	Suburban	1.8	1979	932766.7
12	3233	2	2	Suburban	0.19	1970	434847.6
13	1984	4	1	Suburban	0.83	1977	910805.5
14	4185	2	2	Suburban	0.73	2011	355420.9
15	4917	4	3	Suburban	0.96	1982	502615.1
16	3704	4	3	Suburban	0.31	1984	264727.6
17	1274	5	4	Suburban	2.35	2001	658452.2
18	1882	1	2	Urban	1.43	1983	980032.1
19	3358	4	3	Suburban	0.83	1990	623318.4
20	2847	3	2	Suburban	1.05	2000	389923.4
21	3547	1	3	Urban	1.17	1996	665426.6
22	1775	1	1	Suburban	1.54	2007	166774.6
23	2606	1	3	Urban	1.34	2006	894226.1
24	989	5	1	Urban	2.31	1972	527929.8
25	3534	4	3	Suburban	1.29	1997	476882.3
26	3805	5	4	Urban	2.48	1971	606571.6
27	2699	4	1	Suburban	2.14	1966	643544.7
28	2067	5	3	Suburban	0.6	2018	118187
29	2328	5	4	Suburban	2.33	1960	698486.4
30	4002	3	1	Urban	0.38	1964	279835.1
31	4356	5	4	Urban	2.06	1984	728255
32	4690	2	2	Suburban	1.01	1982	831408.1
33	1446	3	2	Rural	2.21	1965	813760.6
34	3688	5	1	Suburban	2.18	1983	574875.9
35	3235	1	3	Urban	2.03	2015	544161.4
36	1400	2	1	Suburban	2	1989	185732.2
37	3163	2	4	Suburban	0.83	1977	786578.8

38	2861	2	3	Suburban	0.29	1996	307838.9
39	1041	3	2	Urban	1.07	1988	521860.7
40	2841	5	3	Suburban	0.52	2019	320060.4
41	3624	5	2	Suburban	1.77	2022	177136.9

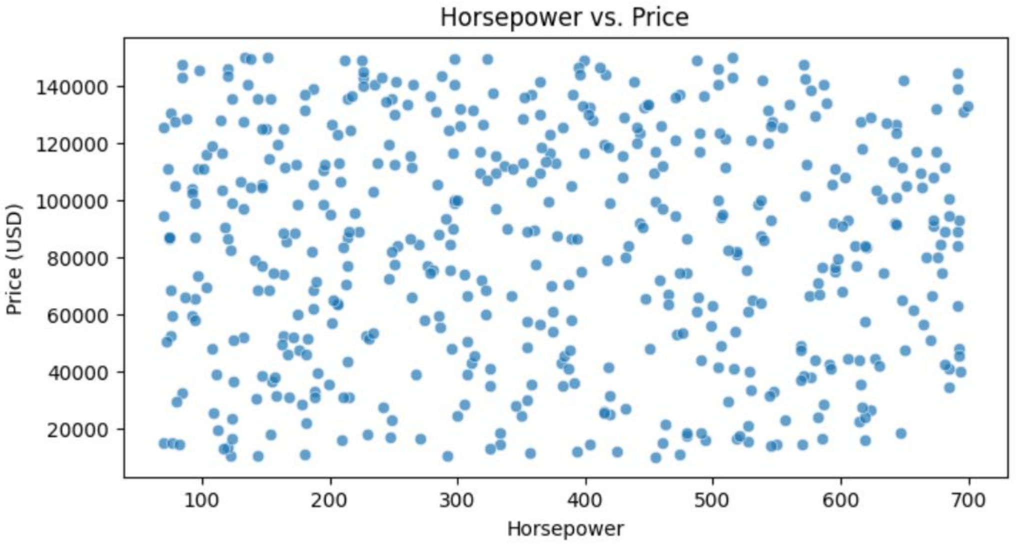
33. Scenario: You work as a data scientist for an automobile company that sells various car models. The company has collected data on different car attributes, such as engine size, horsepower, fuel efficiency, and more, along with their corresponding prices. The marketing team wants to build a predictive model to estimate the price of cars based on their features.

Question: Your task is write a Python program that perform linear regression modeling to predict car prices based on a selected set of features from the dataset. Additionally, you need to evaluate the model's performance and provide insights to the marketing team to understand the most influential factors affecting car prices.

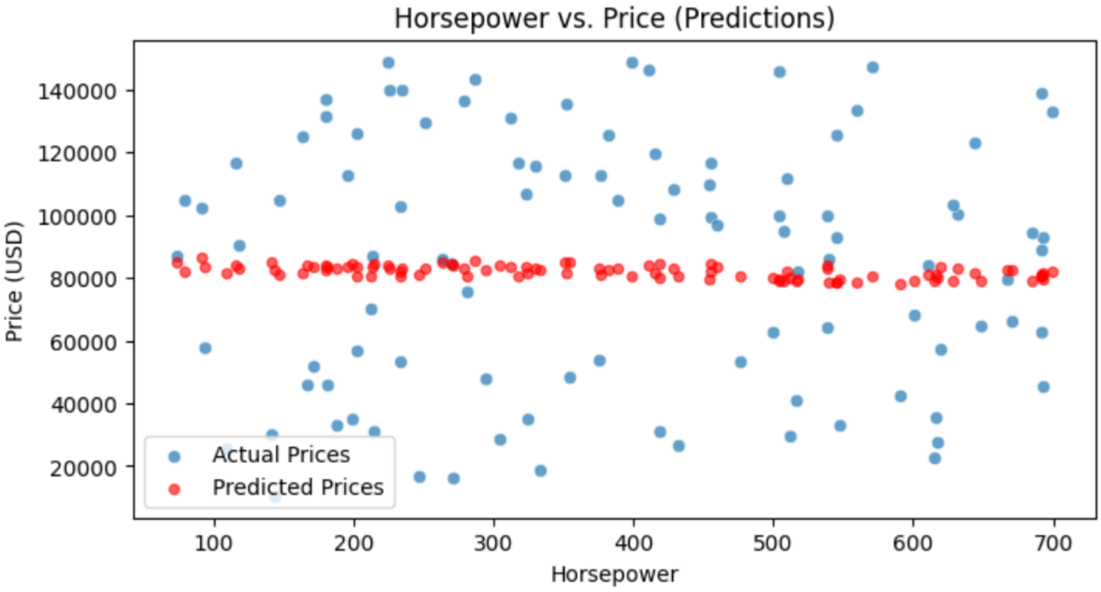
Code:

```
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.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the car price dataset
file_path =r"C:\Users\jampa\Downloads\car_price_data.csv"
df = pd.read_csv(file_path)

# Perform bivariate analysis (Horsepower vs. Price)
plt.figure(figsize=(8,4))
sns.scatterplot(data=df, x='Horsepower', y='Price_USD', alpha=0.7)
plt.title('Horsepower vs. Price')
plt.xlabel('Horsepower')
plt.ylabel('Price (USD)')
plt.show()

# Prepare data for linear regression
features = ['Engine_Size_L', 'Horsepower', 'Fuel_Efficiency_MPG', 'Weight_lbs', 'Cylinders', 'Year']
X = df[features]
y = df['Price_USD']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-Squared (R2): {r2:.2f}")

# Plot the regression line for a single feature (e.g., Horsepower)
plt.figure(figsize=(8,4))
sns.scatterplot(x=X_test['Horsepower'], y=y_test, alpha=0.7, label='Actual Prices')
plt.scatter(X_test['Horsepower'], y_pred, color='red', s=20, alpha=0.6, label='Predicted Prices')
plt.title('Horsepower vs. Price (Predictions)')
plt.xlabel('Horsepower')
plt.ylabel('Price (USD)')

plt.legend()

plt.show()
```

output:



```
Mean Squared Error (MSE): 1548971798.45
R-Squared (R2): -0.01
```



Dataset:

CarID	Engine_Size_L	Horsepower	Fuel_Efficiency_MPG	Weight_lbs	Cylinders	Year	Price_USD
1	3.1	571	49	3058	8	2017	147215.3
2	6.2	214	41.2	2197	4	2017	76902.98
3	5	270	14.6	4532	4	1997	84516.05

4	4.3	530	32.6	5828	4	1998	33567.17
5	1.9	627	49.4	3082	4	2020	44567.04
6	1.9	616	28.8	4884	12	2019	127396.9
7	1.3	322	17.3	3184	6	2009	59865.99
8	5.8	459	29.4	3008	10	2000	125754.1
9	4.3	663	30.5	4738	6	2009	109261.9
10	4.9	325	39.7	3229	6	2015	34803.93

.34 Scenario: Suppose you are working as a data scientist for a medical research organization. Your team has collected data on patients with a certain medical condition and their treatment outcomes. The dataset includes various features such as age, gender, blood pressure, cholesterol levels, and whether the patient responded positively ("Good") or negatively ("Bad") to the treatment. The organization wants to use this model to identify potential candidates who are likely to respond positively to the treatment and improve their medical approach.

Question: Your task is to build a classification model using the KNN algorithm to predict the treatment outcome ("Good" or "Bad") for new patients based on their features. Evaluate the model's performance using accuracy, precision, recall, and F1-score.Make predictions on the test set and display the results.

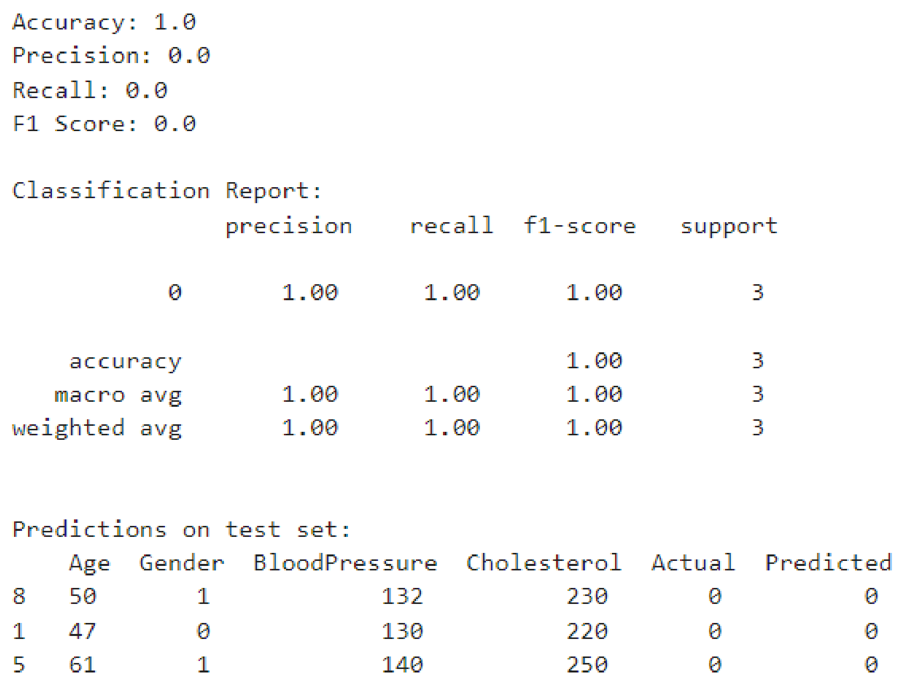
Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

# Load the data
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\patient_data.csv")

# Encode categorical variables
```

```
le_gender = LabelEncoder()
df['Gender'] = le_gender.fit_transform(df['Gender']) # Male=1, Female=0

# Features and target
X = df[['Age', 'Gender', 'BloodPressure', 'Cholesterol']]
y = df['Outcome']
y = LabelEncoder().fit_transform(y) # Good=1, Bad=0

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Train the model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)

# Predict and evaluate
y_pred = knn.predict(X_test_scaled)

# Metrics
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Optional: Show predictions
results = X_test.copy()
```

```
results['Actual'] = y_test

results['Predicted'] = y_pred

print("\nPredictions on test set:\n", results)
```

Output:



```
Accuracy: 1.0
Precision: 0.0
Recall: 0.0
F1 Score: 0.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00         3

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3


Predictions on test set:
    Age  Gender  BloodPressure  Cholesterol  Actual  Predicted
8   50       1            132          230       0          0
1   47       0            130          220       0          0
5   61       1            140          250       0          0
```

Dataset:

Age	Gender	BloodPressure	Cholesterol	Outcome
25	Male	120	180	Good
47	Female	130	220	Bad
52	Female	135	210	Bad
36	Male	128	190	Good
29	Female	118	170	Good

61	Male	140	250	Bad
45	Male	138	240	Bad
38	Female	125	200	Good
50	Male	132	230	Bad
60	Female	145	260	Bad

35. Scenario: You work as a data scientist for a retail company that operates multiple stores. The company is interested in segmenting its customers based on their purchasing behavior to better understand their preferences and tailor marketing strategies accordingly. To achieve this, your team has collected transaction data from different stores, which includes customer IDs, the total amount spent in each transaction, and the frequency of visits.

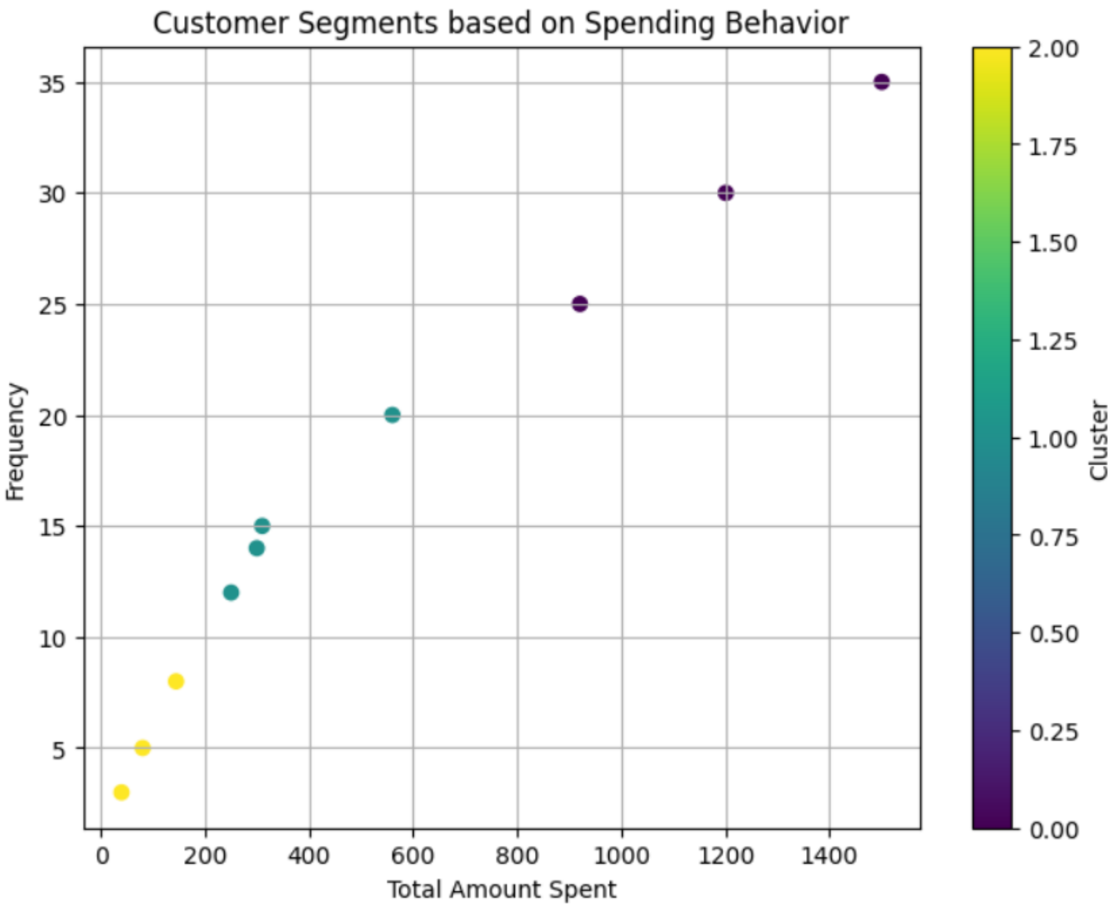
Question: Your task is to build a clustering model using the K-Means algorithm to group customers into distinct segments based on their spending patterns.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Step 1: Load the data
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\customer_transactions.csv")

# Step 2: Preprocess the data
features = df[['TotalAmountSpent', 'Frequency']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Step 3: Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)
```

```
# Step 4: Visualize the clusters

plt.figure(figsize=(8,6))

plt.scatter(df['TotalAmountSpent'], df['Frequency'], c=df['Cluster'], cmap='viridis')

plt.xlabel('Total Amount Spent')

plt.ylabel('Frequency')

plt.title('Customer Segments based on Spending Behavior')

plt.colorbar(label='Cluster')

plt.grid(True)

plt.show()


# Optional: Save the clustered data

df.to_csv("clustered_customers.csv", index=False)
```

output:



Dataset:

CustomerID	TotalAmountSpent	Frequency
1001	250.75	12
1002	145	8
1003	560.4	20
1004	1200.6	30
1005	80.9	5
1006	300	14
1007	920.5	25
1008	40.25	3
1009	1500	35
1010	310.4	15

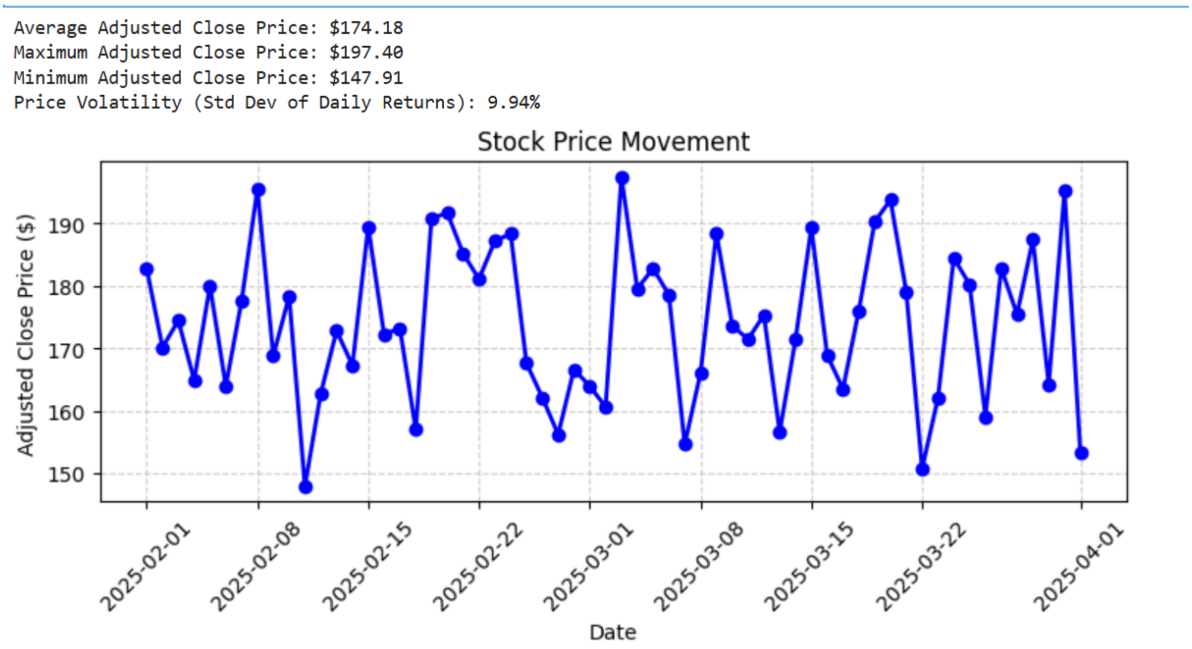
36. Scenario: You are a data analyst working for a finance company. Your team is interested in analyzing the variability of stock prices for a particular company over a certain period. The company's stock data includes the closing prices for each trading day of the specified period.

Question: Your task is to build a Python program that reads the stock data from a CSV file, calculates the variability of stock prices, and provides insights into the stock's price movements.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the stock data from the CSV file
file_path =r"C:\Users\jampa\Downloads\stock_price_data.csv"
df = pd.read_csv(file_path, parse_dates=['Date'])

def analyze_stock_variability(df):
    # Calculate daily returns (percent change)
    df['Daily Return (%)'] = df['Adj Close'].pct_change() * 100
    # Calculate basic statistics
    avg_price = df['Adj Close'].mean()
    max_price = df['Adj Close'].max()
    min_price = df['Adj Close'].min()
    volatility = df['Daily Return (%)'].std()

    print(f"Average Adjusted Close Price: ${avg_price:.2f}")
    print(f"Maximum Adjusted Close Price: ${max_price:.2f}")
    print(f"Minimum Adjusted Close Price: ${min_price:.2f}")
    print(f"Price Volatility (Std Dev of Daily Returns): {volatility:.2f}%")

    # Plot the adjusted close prices
    plt.figure(figsize=(9,3))
    plt.plot(df['Date'], df['Adj Close'], marker='o', color='blue', linewidth=2)
    plt.title('Stock Price Movement')
    plt.xlabel('Date')
    plt.ylabel('Adjusted Close Price ($)')
    plt.xticks(rotation=45)
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.show()

# Run the analysis
analyze_stock_variability(df)
```

output:



```
Average Adjusted Close Price: $174.18
Maximum Adjusted Close Price: $197.40
Minimum Adjusted Close Price: $147.91
Price Volatility (Std Dev of Daily Returns): 9.94%
```

Dataset:

Date	Open	High	Low	Close	Adj Close	Volume
2/1/2025	183.23	183.6	181.06	183.16	182.79	3903847
2/2/2025	171.23	171.26	170.46	171.11	170.12	1831623
2/3/2025	177.79	182.6	173.7	174.96	174.42	2444228
2/4/2025	167.16	171.91	165.05	165.11	164.88	3157227
2/5/2025	179.58	182.54	178.45	181.21	180	1254419
2/6/2025	164.15	165.74	161.65	164.57	163.94	4557241
2/7/2025	173.85	178.4	173.44	177.91	177.59	2112570
2/8/2025	194.29	198.31	193.34	195.87	195.54	2370028
2/9/2025	168.68	169.64	167.36	169.37	168.76	4204595
2/10/2025	179.52	181.67	175.04	180	178.3	2755150
2/11/2025	150.95	154.17	148.32	148.92	147.91	3232499
2/12/2025	164.73	164.89	163.88	163.98	162.75	4638015

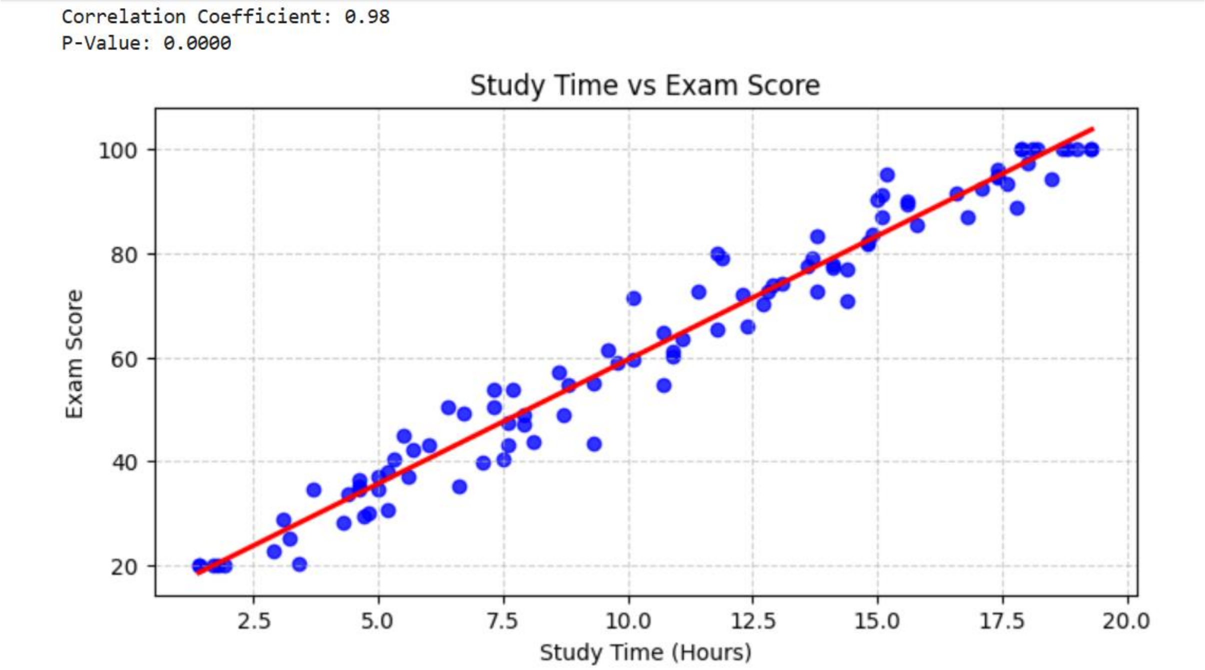
37. Scenario: You are a data scientist working for an educational institution, and you want to explore the correlation between students' study time and their exam scores. You have collected data from a group of students, noting their study time in hours and their corresponding scores in an exam.

Question: Identify any potential correlation between study time and exam scores and explore various plotting functions to visualize this relationship effectively.

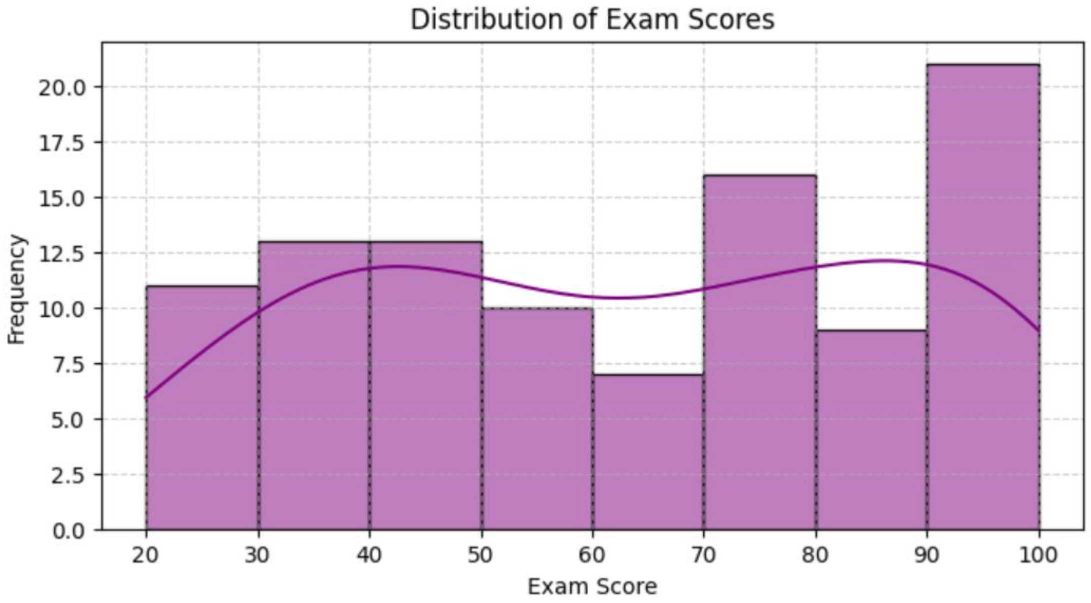
Code:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.stats import pearsonr


# Load the student study and score data

file_path =r"C:\Users\jampa\Downloads\student_study_scores.csv"

df = pd.read_csv(file_path)


def analyze_study_score_correlation(df):

    # Calculate correlation coefficient

    correlation, p_value = pearsonr(df['Study_Time_Hours'], df['Exam_Score'])

    print(f"Correlation Coefficient: {correlation:.2f}")

    print(f"P-Value: {p_value:.4f}")


    # Scatter plot with regression line

    plt.figure(figsize=(8,4))

    sns.regplot(x='Study_Time_Hours', y='Exam_Score', data=df, ci=None, scatter_kws={'color': 'blue'},
line_kws={'color': 'red'})

    plt.title('Study Time vs Exam Score')

    plt.xlabel('Study Time (Hours)')

    plt.ylabel('Exam Score')

    plt.grid(True, linestyle='--', alpha=0.6)

    plt.show()


    # Distribution plot

    plt.figure(figsize=(8,4))

    sns.histplot(df['Exam_Score'], kde=True, color='purple')
    plt.title('Distribution of Exam Scores')

    plt.xlabel('Exam Score')

    plt.ylabel('Frequency')

    plt.grid(True, linestyle='--', alpha=0.6)

    plt.show()


# Run the analysis

analyze_study_score_correlation(df)
```

output:



```
Correlation Coefficient: 0.98
P-Value: 0.0000
```



Dataset:

Student_ID	Study_Time_Hours	Exam_Score
S001	8.6	57.2
S002	19.3	100
S003	5.2	30.7
S004	16.6	91.3
S005	3.2	25.3
S006	12.9	73.9
S007	15.6	90
S008	17.1	92.4
S009	18.8	100
S010	6	43

38. Scenario: You work for a weather data analysis company, and your team is responsible for developing a program to calculate and analyze variability in temperature data for different cities. Question: Write a python program will take in a dataset containing daily temperature readings for each city over a year and perform the following tasks:

1. Calculate the mean temperature for each city.

2. Calculate the standard deviation of temperature for each city.

3. Determine the city with the highest temperature range (difference between the highest and lowest temperatures).

4. Find the city with the most consistent temperature (the lowest standard deviation).

Code:

```
import pandas as pd
# Load the dataset
df = pd.read_csv(r"C:\Users\jampa\Downloads\city_temperature_data.csv")
# Group by city and calculate required statistics
summary = df.groupby('City').agg(
Mean_Temperature=('Temperature', 'mean'),
Std_Deviation=('Temperature', 'std'),
Max_Temperature=('Temperature', 'max'),
Min_Temperature=('Temperature', 'min')
)
# Calculate the temperature range
summary['Temperature_Range'] = summary['Max_Temperature'] - summary['Min_Temperature']
# Find the city with the highest temperature range
highest_range_city = summary['Temperature_Range'].idxmax()
# Find the city with the most consistent temperature (lowest std deviation)
most_consistent_city = summary['Std_Deviation'].idxmin()
# Display results
print("=== Temperature Summary by City ===")
print(summary)
print("\nCity with the Highest Temperature Range:", highest_range_city)
print("City with the Most Consistent Temperature:", most_consistent_city)
```

Output:

Dataset:

```
Date City Temperature
1/1/2024 New York 16.96714
1/2/2024 New York 10.61736
1/3/2024 New York 18.47689
```

1/4/2024 New York 27.2303

1/5/2024 New York 9.658466

1/6/2024 New York 9.65863

1/7/2024 New York 27.79213

1/8/2024 New York 19.67435

1/9/2024 New York 7.305256

1/10/2024 New York 17.4256

1/11/2024 New York 7.365823

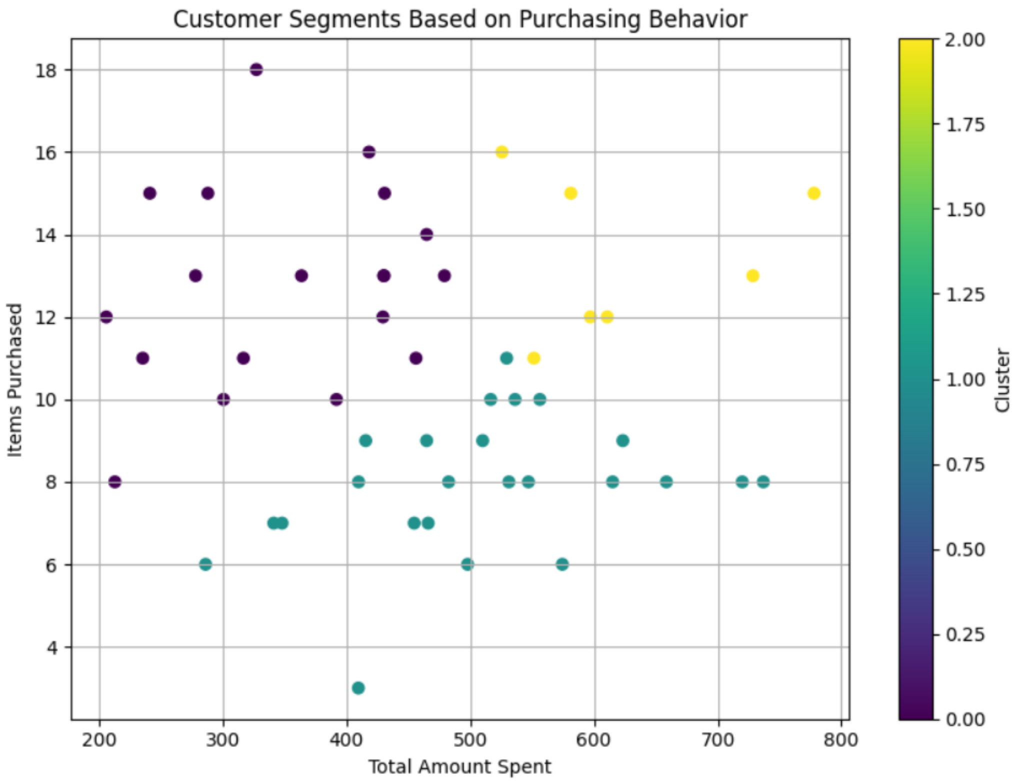
1/12/2024 New York 7.342702

39. Scenario: You work as a data scientist for a marketing agency, and one of your clients is a large e-commerce company. The company wants to understand the purchasing behavior of its customers and segment them into different groups based on their buying patterns. The e-commerce company has provided you with transaction data, including customer IDs, the total amount spent in each transaction, and the number of items purchased.

Question: Build a clustering model using the K-Means algorithm to group customers based on their spending and purchase behavior and visualize the clusters using scatter plots or other appropriate visualizations to gain insights into customer distribution and distinguish different segments.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Generate synthetic data
np.random.seed(42)
num_customers = 50
data = {
"CustomerID": [f"CUST{i:03d}" for i in range(1, num_customers + 1)],
"TotalAmountSpent": np.random.normal(500, 150, num_customers).round(2),
"ItemsPurchased": np.random.poisson(10, num_customers)
}


df = pd.DataFrame(data)
df.to_csv("ecommerce_transactions.csv", index=False) # Save to CSV


# Load the data
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\ecommerce_transactions (1).csv")


# Prepare features
X = df[['TotalAmountSpent', 'ItemsPurchased']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)


# Visualization
plt.figure(figsize=(8, 6))
scatter = plt.scatter(df['TotalAmountSpent'], df['ItemsPurchased'], c=df['Cluster'], cmap='viridis')
plt.xlabel("Total Amount Spent")
plt.ylabel("Items Purchased")
plt.title("Customer Segments Based on Purchasing Behavior")
plt.colorbar(scatter, label='Cluster')
plt.grid(True)
plt.tight_layout()
plt.show()
```

output:



Dataset:

CustomerID	TotalAmountSpent	ItemsPurchased
CUST001	574.51	6
CUST002	479.26	13
CUST003	597.15	12
CUST004	728.45	13
CUST005	464.88	9
CUST006	464.88	14
CUST007	736.88	8
CUST008	615.12	8
CUST009	429.58	12
CUST010	581.38	15
CUST011	430.49	13
CUST012	430.14	13
CUST013	536.29	10
CUST014	213.01	8
CUST015	241.26	15

CUST047	430.9	15
CUST048	658.57	8
CUST049	551.54	11
CUST050	235.54	11

40. Scenario: You are a data analyst working for a sports analytics company. The company has collected data on various soccer players, including their names, ages, positions, number of goals scored, and weekly salaries. Create dataset on your own and store in a CSV file.

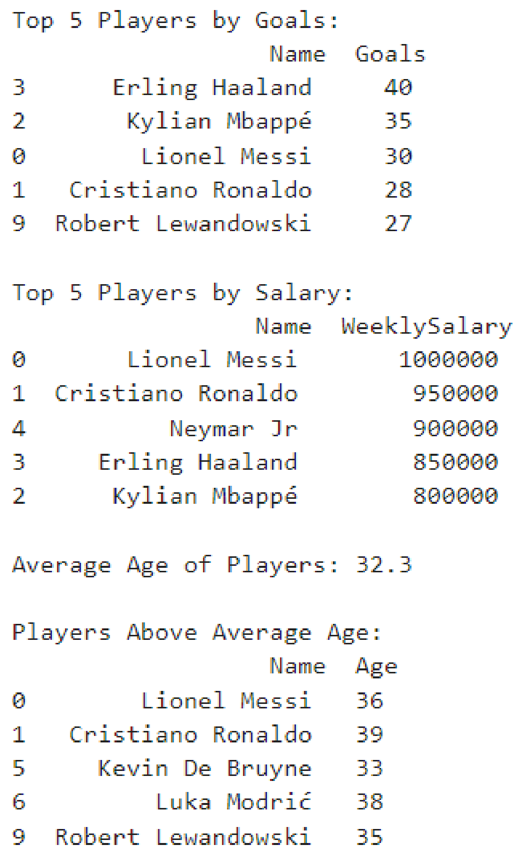
Question: Develop a Python program to read the data from the CSV file into a pandas data frame, to find the top 5 players with the highest number of goals scored and the top 5 players with the highest salaries. Also calculate the average age of players and display the names of players who are above the average age and visualize the distribution of players based on their positions using a bar chart.

Code:

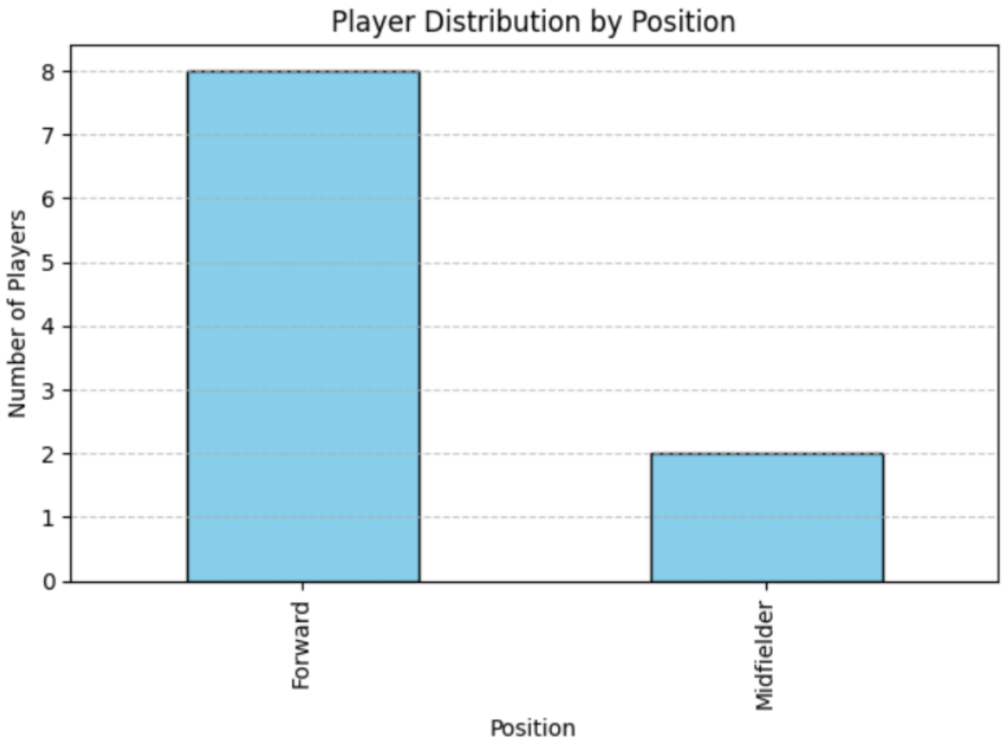
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
df = pd.read_csv(r"C:\Users\vara prasad\Downloads\soccer_players.csv")

# Top 5 players with the highest number of goals
top_goals = df.sort_values(by="Goals", ascending=False).head(5)
print("Top 5 Players by Goals:\n", top_goals[['Name', 'Goals']], "\n")

# Top 5 players with the highest weekly salaries
top_salary = df.sort_values(by="WeeklySalary", ascending=False).head(5)
print("Top 5 Players by Salary:\n", top_salary[['Name', 'WeeklySalary']], "\n")

# Average age calculation
avg_age = df['Age'].mean()
```

```
print("Average Age of Players:", avg_age)


# Players above average age

above_avg_age = df[df['Age'] > avg_age]

print("\nPlayers Above Average Age:\n", above_avg_age[['Name', 'Age']], "\n")


# Visualize player distribution by position

position_counts = df['Position'].value_counts()

position_counts.plot(kind='bar', color='skyblue', edgecolor='black')

plt.title("Player Distribution by Position")

plt.xlabel("Position")

plt.ylabel("Number of Players")

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()

plt.show()
```

output:



```
Top 5 Players by Goals:
                 Name  Goals
3      Erling Haaland     40
2       Kylian Mbappé     35
0        Lionel Messi     30
1   Cristiano Ronaldo     28
9  Robert Lewandowski     27

Top 5 Players by Salary:
                Name  WeeklySalary
0       Lionel Messi       1000000
1  Cristiano Ronaldo        950000
4          Neymar Jr        900000
3     Erling Haaland        850000
2      Kylian Mbappé        800000

Average Age of Players: 32.3

Players Above Average Age:
                 Name  Age
0        Lionel Messi   36
1   Cristiano Ronaldo   39
5     Kevin De Bruyne   33
6         Luka Modrić   38
9  Robert Lewandowski   35
```



Dataset:

Name	Age	Position	Goals	WeeklySalary	
Lionel Messi	36	Forward	30	1000000	
Cristiano Ronaldo	39	Forward	28	950000	
Kylian MbappÃ©	25	Forward	35	800000	
Erling Haaland	24	Forward	40	850000	
Neymar Jr	32	Forward	22	900000	
Kevin De Bruyne	33	Midfielder	10	400000	
Luka ModriÄ‡	38	Midfielder	5	350000	
Mohamed Salah	31	Forward	26	500000	
Harry Kane	30	Forward	24	600000	
Robert Lewandowski	35	Forward	27	700000	

41. Sales and Profit Analysis: a) Load the “sales_data.csv” file into a Pandas data frame, which contains columns “Date,” “Product,” “Quantity Sold,” and “Unit Price”

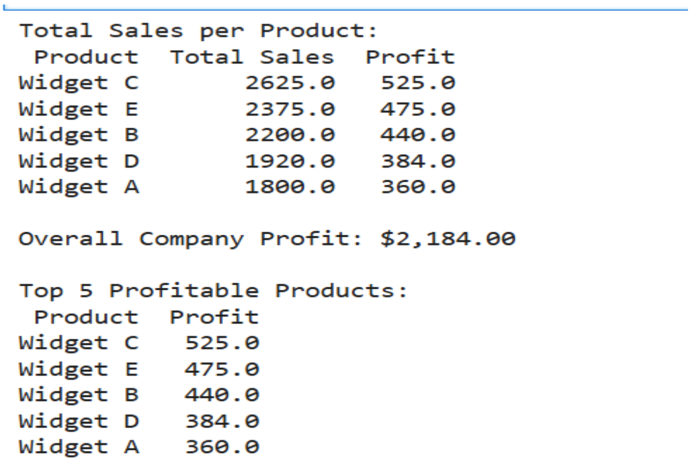
b) Create a new column named “Total Sales” that calculates the total sales for each transaction (Quantity Sold * Unit Price).

c) Calculate the total sales for each product and the overall profit, considering a 20% profit margin on each product. Display the top 5 most profitable products.

CODE:

```
import pandas as pd

df = pd.read_csv(r"C:\Users\jampa\Downloads\sales_data.csv")

df['Total Sales'] = df['Quantity Sold'] * df['Unit Price']

product_sales = df.groupby('Product').agg({'Total Sales': 'sum'}).reset_index()

product_sales['Profit'] = product_sales['Total Sales'] * 0.20

overall_profit = product_sales['Profit'].sum()

top_products = product_sales.nlargest(5, 'Profit')

print("Total Sales per Product:")

print(product_sales.sort_values('Total Sales', ascending=False).to_string(index=False))

print(f"\nOverall Company Profit: ${overall_profit:,.2f}")

print("\nTop 5 Profitable Products:")

print(top_products[['Product', 'Profit']].to_string(index=False))
```

OUTPUT:



```
Total Sales per Product:
 Product  Total Sales  Profit
Widget C       2625.0   525.0
Widget E       2375.0   475.0
Widget B       2200.0   440.0
Widget D       1920.0   384.0
Widget A       1800.0   360.0

Overall Company Profit: $2,184.00

Top 5 Profitable Products:
 Product  Profit
Widget C   525.0
Widget E   475.0
Widget B   440.0
Widget D   384.0
Widget A   360.0
```

Dataset:

Date	Product	Quantity Sold	Unit Price
1/1/2025	Widget A	100	10

1/2/2025	Widget B	50	20
1/3/2025	Widget C	75	15
1/4/2025	Widget A	80	10
1/5/2025	Widget B	60	20
1/6/2025	Widget D	90	12
1/7/2025	Widget E	40	25
1/8/2025	Widget C	100	15
1/9/2025	Widget D	70	12
	Widget		

42. Customer Segmentation: a) Load “customer_data.” file into a Pandas data frame, which contains “Customer ID,” ”Age,” “Gender,” and “Total Spending.”

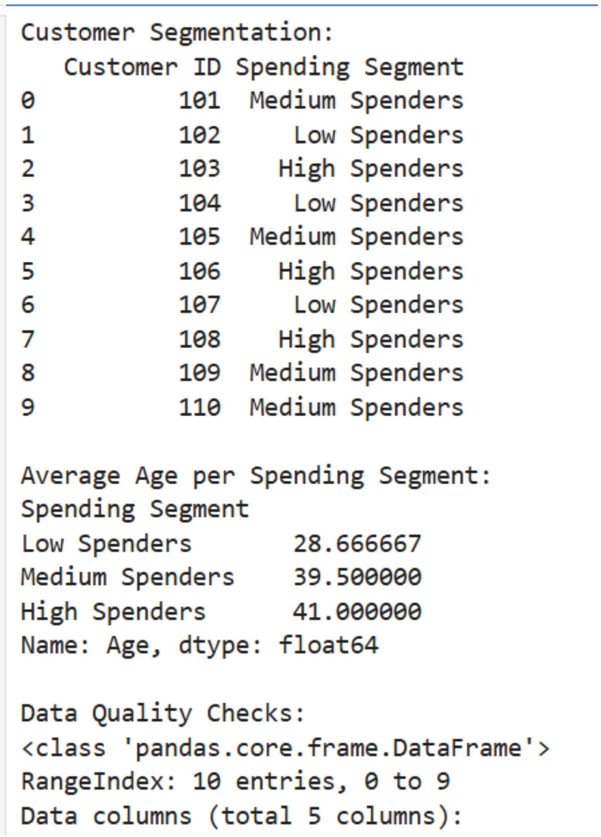
b) Segment customers into three groups based on their total spending: “High Spenders,” ”Medium Spenders,” and “Low Spenders.” Assign these segments to a new column in the data frame.

c) Calculate the average age of customers in each spending segment.

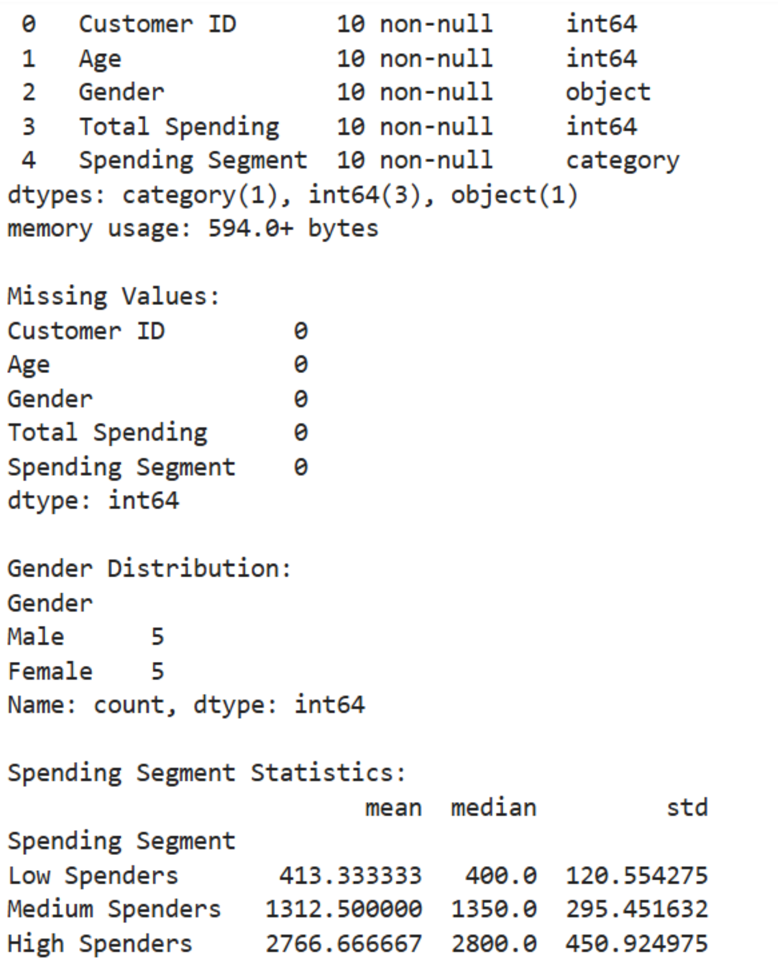
CODE:

```
import pandas as pd

df = pd.read_csv(r"C:\Users\jampa\OneDrive\文档\customer_data.csv")

quantiles = df['Total Spending'].quantile([0.33, 0.67])

df['Spending Segment'] = pd.cut(df['Total Spending'], bins=[-1, quantiles[0.33], quantiles[0.67], float('inf')],

                          labels=['Low Spenders', 'Medium Spenders', 'High Spenders'])

avg_age = df.groupby('Spending Segment')['Age'].mean()

print("Customer Segmentation:")

print(df[['Customer ID', 'Spending Segment']])

print("\nAverage Age per Spending Segment:")

print(avg_age)
print("\nData Quality Checks:")

df.info()

print("\nMissing Values:")

print(df.isna().sum())

print("\nGender Distribution:")

print(df['Gender'].value_counts())

print("\nSpending Segment Statistics:")

print(df.groupby('Spending Segment', observed=True)['Total Spending'].agg(['mean', 'median', 'std']))
```

OUTPUT:



```
Customer Segmentation:
   Customer ID Spending Segment
0          101  Medium Spenders
1          102     Low Spenders
2          103    High Spenders
3          104     Low Spenders
4          105  Medium Spenders
5          106    High Spenders
6          107     Low Spenders
7          108    High Spenders
8          109  Medium Spenders
9          110  Medium Spenders

Average Age per Spending Segment:
Spending Segment
Low Spenders       28.666667
Medium Spenders    39.500000
High Spenders      41.000000
Name: Age, dtype: float64

Data Quality Checks:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
```



```
 0   Customer ID       10 non-null     int64
 1   Age               10 non-null     int64
 2   Gender            10 non-null     object
 3   Total Spending    10 non-null     int64
 4   Spending Segment  10 non-null     category
dtypes: category(1), int64(3), object(1)
memory usage: 594.0+ bytes

Missing Values:
Customer ID         0
Age                 0
Gender              0
Total Spending      0
Spending Segment    0
dtype: int64

Gender Distribution:
Gender
Male      5
Female    5
Name: count, dtype: int64

Spending Segment Statistics:
                         mean  median         std
Spending Segment
Low Spenders       413.333333   400.0  120.554275
Medium Spenders   1312.500000  1350.0  295.451632
High Spenders     2766.666667  2800.0  450.924975
```

Dataset:

Customer ID	Age	Gender	Total Spending
101	25	Male	1200
102	34	Female	540
103	45	Male	3200
104	23	Female	300
105	52	Male	1500
106	37	Female	2300
107	29	Female	400
108	41	Male	2800
109	33	Male	950
110	48	Female	1600

43. Data Cleaning and Transformation

Task:

- a) Load the employee_data.csv file into a Pandas DataFrame. The file contains the columns:

Employee ID, Full Name, Department, and Salary.

- b) Convert the Salary column to a numeric data type.

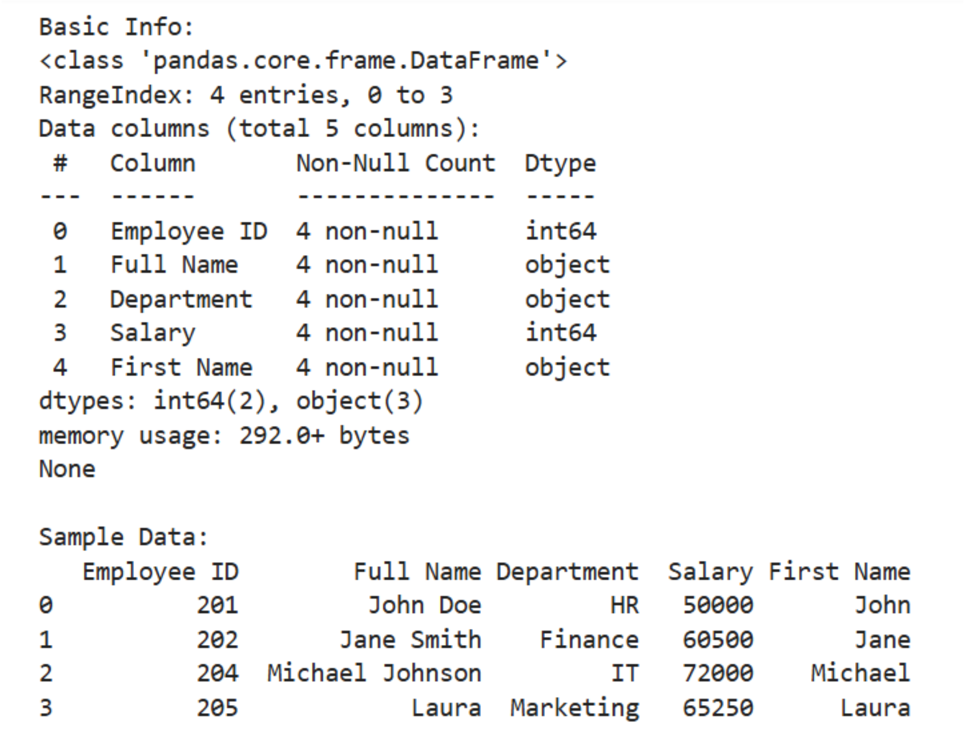
- c) Remove any rows where the Department column has missing values.

- d) Create a new column named First Name that extracts the first name from the Full Name column.

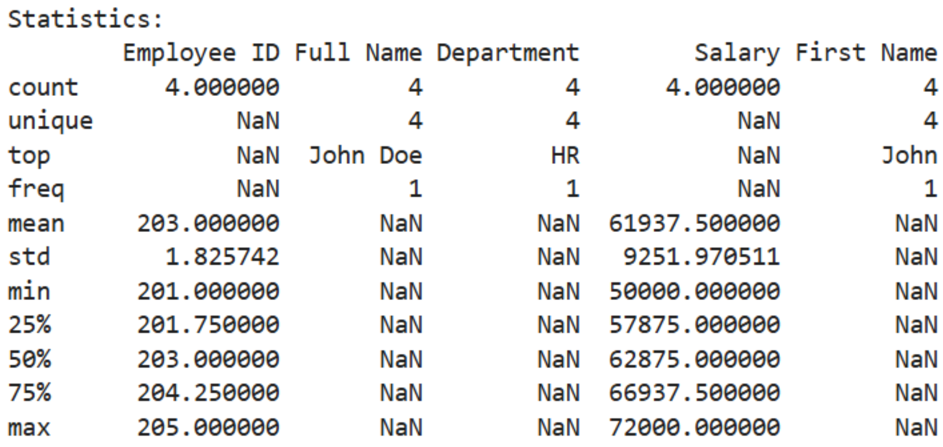
CODE:

```
import pandas as pd

employee_df = pd.read_csv("employee_data.csv")

employee_df['Salary'] = pd.to_numeric(employee_df['Salary'].replace('[\$,]', '', regex=True), errors='coerce')

employee_df = employee_df.dropna(subset=["Department"])

employee_df['First Name'] = employee_df['Full Name'].str.split().str[0]

print(employee_df.head())
```

OUTPUT:



```
Basic Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Employee ID  4 non-null      int64 
 1   Full Name    4 non-null      object
 2   Department   4 non-null      object
 3   Salary       4 non-null      int64 
 4   First Name   4 non-null      object
dtypes: int64(2), object(3)
memory usage: 292.0+ bytes
None

Sample Data:
   Employee ID        Full Name Department  Salary First Name
0          201         John Doe         HR   50000       John
1          202       Jane Smith    Finance   60500       Jane
2          204  Michael Johnson         IT   72000    Michael
3          205            Laura  Marketing   65250      Laura
```



```
Statistics:
        Employee ID Full Name Department        Salary First Name
count      4.000000         4          4      4.000000          4
unique          NaN         4          4           NaN          4
top             NaN  John Doe         HR           NaN       John
freq            NaN         1          1           NaN          1
mean     203.000000       NaN        NaN  61937.500000        NaN
std        1.825742       NaN        NaN   9251.970511        NaN
min      201.000000       NaN        NaN  50000.000000        NaN
25%      201.750000       NaN        NaN  57875.000000        NaN
50%      203.000000       NaN        NaN  62875.000000        NaN
75%      204.250000       NaN        NaN  66937.500000        NaN
max      205.000000       NaN        NaN  72000.000000        NaN
```

Dataset:

Employee ID	Full Name	Department	Salary
201	John Doe	HR	$50,000
202	Jane Smith	Finance	$60,500
203	Emily Davis		$55,000
204	Michael Johnson	IT	$72,000
205	Laura	Marketing	$65,250

44. Time Series Analysis

Task:

- a) Load the temperature_data.csv file into a Pandas DataFrame. The file contains the columns:

Date and Temperature (Celsius).

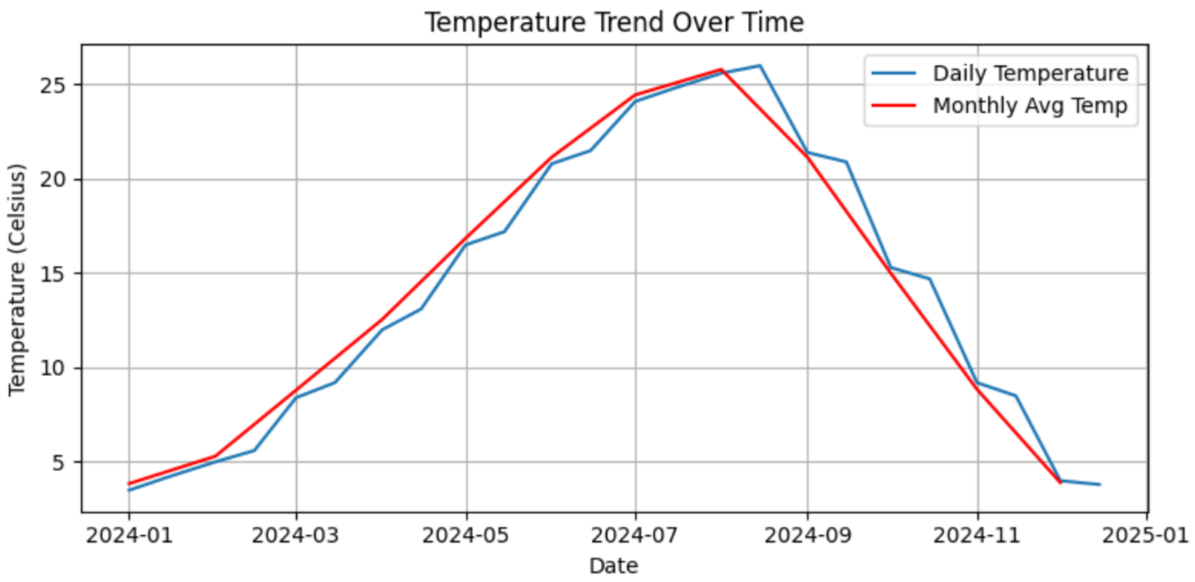
- b) Convert the Date column to the Pandas datetime data type.
- c) Calculate the average temperature for each month and display the results in chronological order.
- d) Plot a line chart to visualize the temperature trend over time.

CODE:

```
import pandas as pd

import matplotlib.pyplot as plt

temp_df = pd.read_csv("temperature_data.csv")
```

```
temp_df['Date'] = pd.to_datetime(temp_df['Date'])

temp_df['Month'] = temp_df['Date'].dt.to_period('M')

monthly_avg_temp = temp_df.groupby('Month')['Temperature (Celsius)'].mean().reset_index()

monthly_avg_temp['Month'] = monthly_avg_temp['Month'].dt.to_timestamp()

plt.figure(figsize=(10, 5))

plt.plot(temp_df['Date'], temp_df['Temperature (Celsius)'])

plt.plot(monthly_avg_temp['Month'], monthly_avg_temp['Temperature (Celsius)'], color='red')

plt.title('Temperature Trend Over Time')

plt.xlabel('Date')

plt.ylabel('Temperature (Celsius)')

plt.grid(True)

plt.tight_layout()

plt.show()
```

OUTPUT:



Dataset:

Date	Temperature (Celsius)
1/1/2024	3.5
1/15/2024	4.2
2/1/2024	5

2/15/2024	5.6
3/1/2024	8.4
3/15/2024	9.2
4/1/2024	12
4/15/2024	13.1
5/1/2024	16.5
5/15/2024	17.2
6/1/2024	20.8
6/15/2024	21.5
7/1/2024	24.1
7/15/2024	24.8
8/1/2024	25.6
8/15/2024	26
9/1/2024	21.4
9/15/2024	20.9
10/1/2024	15.3
10/15/202	14.7
11/1/2024	9.2
11/15/202	8.5
12/1/2024	4
12/15/202	3.8

45. Joining DataFrames

Task:

- a) Load the orders_data.csv file into a Pandas DataFrame. This file contains:

Order ID, Customer ID, and Order Date.

- b) Load the customer_info.csv file into another Pandas DataFrame. This file contains:

Customer ID, Name, Email, and Phone Number.

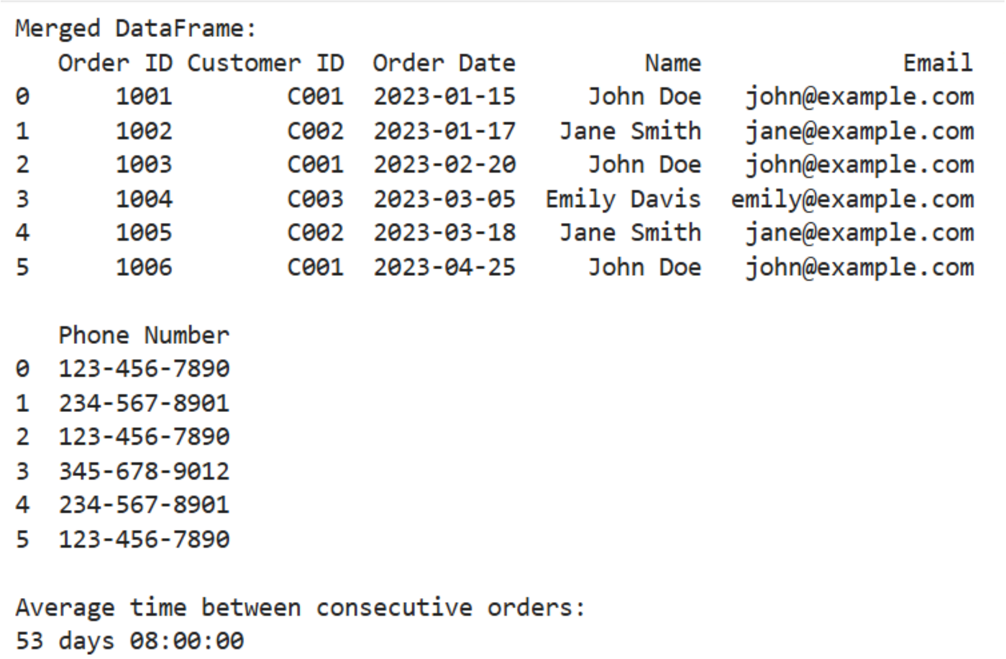
- c) Merge the two DataFrames on the Customer ID column to create a single DataFrame that includes both order and customer details.

- d) Calculate the average time it takes for a customer to place another order after their first one (i.e., time between consecutive orders).

CODE:

```
import pandas as pd

orders_df = pd.read_csv("order_data.csv")

customers_df = pd.read_csv("customer_info.csv")

merged_df = pd.merge(orders_df, customers_df, on="Customer ID", how="inner")

print(merged_df)

merged_df['Order Date'] = pd.to_datetime(merged_df['Order Date'])

merged_df = merged_df.sort_values(by=['Customer ID', 'Order Date'])

merged_df['Time Diff'] = merged_df.groupby('Customer ID')['Order Date'].diff()

avg_time_diff = merged_df['Time Diff'].dropna().mean()

print("\nAverage time between consecutive orders:")

print(avg_time_diff)
```

OUTPUT:



```
Merged DataFrame:
   Order ID Customer ID  Order Date         Name              Email
0      1001        C001  2023-01-15     John Doe   john@example.com
1      1002        C002  2023-01-17   Jane Smith   jane@example.com
2      1003        C001  2023-02-20     John Doe   john@example.com
3      1004        C003  2023-03-05  Emily Davis  emily@example.com
4      1005        C002  2023-03-18   Jane Smith   jane@example.com
5      1006        C001  2023-04-25     John Doe   john@example.com

   Phone Number
0  123-456-7890
1  234-567-8901
2  123-456-7890
3  345-678-9012
4  234-567-8901
5  123-456-7890

Average time between consecutive orders:
53 days 08:00:00
```

Dataset:

Customer data:

Customer ID	Name	Email	Phone Number
C001	John Doe	john@example.com	123-456-7890
C002	Jane Smith	jane@example.com	234-567-8901
C003	Emily Davis	emily@example.com	345-678-9012

Order details;

Order ID	Customer ID	Order Date
1001	C001	1/15/2023
1002	C002	1/17/2023
1003	C001	2/20/2023
1004	C003	3/5/2023
1005	C002	3/18/2023
1006	C001	4/25/2023