LAB SESSION 6

In [1]:
```python
import pandas as pd
from sklearn.datasets import load_diabetes

# Load the dataset
diabetes = load_diabetes()
df = pd.DataFrame(data=diabetes.data, columns=diabetes.feature_names)
df['target'] = diabetes.target

# Display the first few rows
print(df.head())
```

```
        age       sex       bmi        bp        s1        s2        s3  \
0  0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2  0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -0.032356
3 -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038
4  0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142

         s4        s5        s6  target
0 -0.002592  0.019907 -0.017646   151.0
1 -0.039493 -0.068332 -0.092204    75.0
2 -0.002592  0.002861 -0.025930   141.0
3  0.034309  0.022688 -0.009362   206.0
4 -0.002592 -0.031988 -0.046641   135.0
```

In [2]:
```python
# Calculate basic descriptive statistics
print("Mean:\n", df.mean())
print("\nMedian:\n", df.median())
print("\nMode:\n", df.mode().iloc[0])
print("\nStandard Deviation:\n", df.std())
print("\nVariance:\n", df.var())

# Additional descriptive statistics
print("\nRange:\n", df.max() - df.min())
print("\nSkewness:\n", df.skew())
print("\nKurtosis:\n", df.kurt())
```

```
Mean:
 age      -1.444295e-18
sex       2.543215e-18
bmi      -2.255925e-16
bp       -4.854086e-17
s1       -1.428596e-17
s2        3.898811e-17
s3       -6.028360e-18
s4       -1.788100e-17
s5        9.243486e-17
s6        1.351770e-17
target    1.521335e+02
dtype: float64

Median:
 age         0.005383
sex        -0.044642
bmi        -0.007284
bp         -0.005670
s1         -0.004321
s2         -0.003819
s3         -0.006584
s4         -0.002592
s5         -0.001947
s6         -0.001078
target    140.500000
dtype: float64

Mode:
 age        0.016281
sex       -0.044642
bmi       -0.030996
bp        -0.040099
s1        -0.037344
s2        -0.001001
s3        -0.013948
s4        -0.039493
s5        -0.018114
s6         0.003064
target    72.000000
Name: 0, dtype: float64

Standard Deviation:
 age        0.047619
sex        0.047619
bmi        0.047619
bp         0.047619
s1         0.047619
s2         0.047619
s3         0.047619
s4         0.047619
s5         0.047619
s6         0.047619
target    77.093005
dtype: float64

Variance:
 age          0.002268
sex          0.002268
bmi          0.002268
bp           0.002268
s1           0.002268
s2           0.002268
s3           0.002268
s4           0.002268
s5           0.002268
s6           0.002268
target    5943.331348
dtype: float64

Range:
 age         0.217952
sex         0.095322
bmi         0.260831
bp          0.244442
s1          0.280694
s2          0.314401
s3          0.283486
s4          0.261629
s5          0.259694
s6          0.273379
target    321.000000
dtype: float64

Skewness:
 age      -0.231382
sex       0.127385
bmi       0.598148
bp        0.290658
s1        0.378108
s2        0.436592
s3        0.799255
s4        0.735374
s5        0.291754
s6        0.207917
target    0.440563
dtype: float64

Kurtosis:
 age      -0.671224
sex      -1.992811
bmi       0.095094
bp       -0.532797
s1        0.232948
s2        0.601381
s3        0.981507
s4        0.444402
s5       -0.134367
s6        0.236917
target   -0.883057
dtype: float64
```

In [3]:
```python
from scipy import stats

# Example data: BMI values
bmi_values = df['bmi']

# Hypothetical population mean for BMI
population_mean = 0.05

# Perform one-sample t-test
t_stat, p_value = stats.ttest_1samp(bmi_values, population_mean)

print(f"T-Statistic: {t_stat}")
print(f"P-Value: {p_value}")
```

```
T-Statistic: -22.074985843710174
P-Value: 2.7634312235044638e-73
```

In [4]:
```python
import numpy as np
from scipy import stats

# Sample mean and standard error for BMI
sample_mean = np.mean(bmi_values)
standard_error = stats.sem(bmi_values)

# Compute 95% confidence interval for BMI
confidence_interval = stats.norm.interval(0.95, loc=sample_mean, scale=standard_error)

print(f"95% Confidence Interval for BMI: {confidence_interval}")
```

```
95% Confidence Interval for BMI: (-0.004439332370169141, 0.0044393323701686915)
```

In [5]:
```python
import statsmodels.api as sm

# Define independent variable (add constant for intercept)
X = sm.add_constant(df['bmi'])

# Define dependent variable
y = df['target']

# Fit linear regression model
model = sm.OLS(y, X).fit()

# Print model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 target   R-squared:                       0.344
Model:                            OLS   Adj. R-squared:                  0.342
Method:                 Least Squares   F-statistic:                     230.7
Date:                Thu, 05 Sep 2024   Prob (F-statistic):           3.47e-42
Time:                        11:25:18   Log-Likelihood:                -2454.0
No. Observations:                 442   AIC:                             4912.
Df Residuals:                     440   BIC:                             4920.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        152.1335      2.974     51.162      0.000     146.289     157.978
bmi          949.4353     62.515     15.187      0.000     826.570    1072.301
==============================================================================
Omnibus:                       11.674   Durbin-Watson:                   1.848
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                7.310
Skew:                           0.156   Prob(JB):                       0.0259
Kurtosis:                       2.453   Cond. No.                         21.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

HEART DISEASE DATA SET

EXERCISE 5.1

In [6]:
```python
import pandas as pd

# URL for the Heart Disease dataset
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data'
column_names = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target']
data = pd.read_csv(url, header=None, names=column_names, na_values='?')

# Display the first few rows of the dataset
print(data.head())
```

```
    age  sex   cp  trestbps   chol  fbs  restecg  thalach  exang  oldpeak  \
0  63.0  1.0  1.0     145.0  233.0  1.0      2.0    150.0    0.0      2.3
1  67.0  1.0  4.0     160.0  286.0  0.0      2.0    108.0    1.0      1.5
2  67.0  1.0  4.0     120.0  229.0  0.0      2.0    129.0    1.0      2.6
3  37.0  1.0  3.0     130.0  250.0  0.0      0.0    187.0    0.0      3.5
4  41.0  0.0  2.0     130.0  204.0  0.0      2.0    172.0    0.0      1.4

   slope   ca  thal  target
0    3.0  0.0   6.0       0
1    2.0  3.0   3.0       2
2    2.0  2.0   7.0       1
3    3.0  0.0   3.0       0
4    1.0  0.0   3.0       0
```

In [7]:
```python
import numpy as np

# Drop rows with missing values
data = data.dropna()

# Calculate descriptive statistics
mean = data[['age', 'trestbps', 'chol']].mean()
median = data[['age', 'trestbps', 'chol']].median()
mode = data[['age', 'trestbps', 'chol']].mode().iloc[0]
std_dev = data[['age', 'trestbps', 'chol']].std()
variance = data[['age', 'trestbps', 'chol']].var()

# Print the results
print("Mean:\n", mean)
print("Median:\n", median)
print("Mode:\n", mode)
print("Standard Deviation:\n", std_dev)
print("Variance:\n", variance)
```

```
Mean:
 age          54.542088
trestbps    131.693603
chol        247.350168
dtype: float64
Median:
 age          56.0
trestbps    130.0
chol        243.0
dtype: float64
Mode:
 age          58.0
trestbps    120.0
chol        197.0
Name: 0, dtype: float64
Standard Deviation:
 age          9.049736
trestbps    17.762806
chol        51.997583
dtype: float64
Variance:
 age           81.897716
trestbps     315.517290
chol        2703.748589
dtype: float64
```

In [8]:
```python
from scipy import stats

# Hypothesized value
hypothesized_mean = 200

# Perform a one-sample t-test
t_stat, p_value = stats.ttest_1samp(data['chol'], hypothesized_mean)

# Print the results
print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")

# Interpret the p-value
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The average cholesterol level is significantly different from 200 mg/dL.")
else:
    print("Fail to reject the null hypothesis: The average cholesterol level is not significantly different from 200 mg/dL.")
```

```
T-statistic: 15.69338391229138
P-value: 8.344474495694836e-41
Reject the null hypothesis: The average cholesterol level is significantly different from 200 mg/dL.
```

In [9]:
```python
# Mean and standard error
mean_age = data['age'].mean()
std_error_age = data['age'].sem()

# 95% Confidence Interval
confidence_interval_age = stats.t.interval(0.95, len(data['age']) - 1, loc=mean_age, scale=std_error_age)

# Print the results
print(f"95% Confidence Interval for Age mean: {confidence_interval_age}")
```

```
95% Confidence Interval for Age mean: (53.50864786485218, 55.57552721932291)
```

EXERCISE 5.2

In [10]:
```python
import statsmodels.api as sm

# Define the independent and dependent variables
X = data['age']  # Independent variable
y = data['chol']  # Dependent variable

# Add a constant to the independent variable (for the intercept)
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print the regression summary
print(model.summary())
```

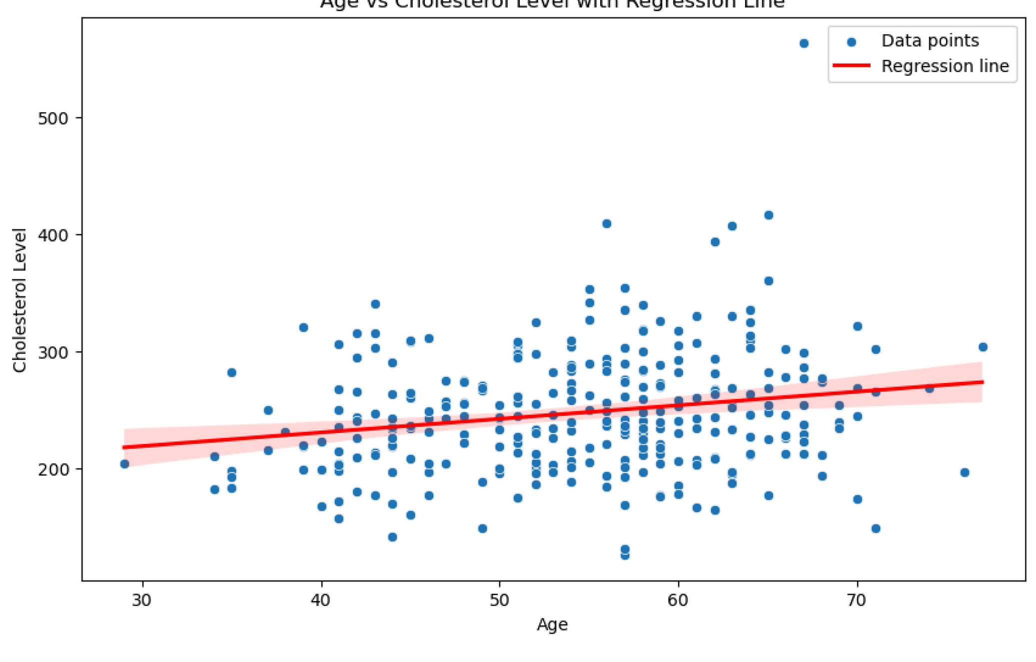
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   chol   R-squared:                       0.041
Model:                            OLS   Adj. R-squared:                  0.038
Method:                 Least Squares   F-statistic:                     12.63
Date:                Thu, 05 Sep 2024   Prob (F-statistic):           0.000441
Time:                        11:35:58   Log-Likelihood:                -1588.2
No. Observations:                 297   AIC:                             3180.
Df Residuals:                     295   BIC:                             3188.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        183.8446     18.111     10.151      0.000     148.202     219.488
age            1.1643      0.328      3.554      0.000       0.520       1.809
==============================================================================
Omnibus:                       70.967   Durbin-Watson:                   2.090
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              237.843
Skew:                           1.009   Prob(JB):                     2.25e-52
Kurtosis:                       6.892   Cond. No.                         338.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [11]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the data and the regression line
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='chol', data=data, label='Data points')
sns.regplot(x='age', y='chol', data=data, scatter=False, color='red', label='Regression line')

# Add labels and title
plt.xlabel('Age')
plt.ylabel('Cholesterol Level')
plt.title('Age vs Cholesterol Level with Regression Line')
plt.legend()
plt.show()
```



In [ ]: