In [1]:
```
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]:
```
# 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]:
```
#Performing Inferential Statistics

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]:
```
#Confidence Intervals

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_err

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

```
95% Confidence Interval for BMI: (np.float64(-0.004439332370169141), np.float64(0.00
44393323701686915))
```

In [5]:
```
#Regression Analysis

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:                Sun, 08 Sep 2024   Prob (F-statistic):           3.47e-42
Time:                        13:24:21   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 spe
cified.
```

Analyzing a Health-Related Dataset Load the Dataset First, we will load the Heart Disease dataset using pandas.

In [19]:
```python
import pandas as pd

# Load the dataset
data = pd.read_csv(r"C:\Users\pavan\Downloads\heart_statlog_cleveland_hungary_final
# Display the first few rows of the dataset
data.head()
```

Out[19]:

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope
0	40	1	2	140	289	0	0	172	0	0.0	1
1	49	0	3	160	180	0	0	156	0	1.0	2
2	37	1	2	130	283	0	1	98	0	0.0	1
3	48	0	4	138	214	0	0	108	1	1.5	2
4	54	1	3	150	195	0	0	122	0	0.0	1

Calculate Descriptive Statistics Next, we will calculate the mean, median, mode, standard deviation, and variance for relevant features.

In [20]:
```python
# Calculate mean, median, mode, standard deviation, and variance
descriptive_stats = {
    'Mean': data.mean(),
    'Median': data.median(),
```

```
    'Mode': data.mode().iloc[0],
    'Standard Deviation': data.std(),
    'Variance': data.var()
}

descriptive_stats_df = pd.DataFrame(descriptive_stats)
print(descriptive_stats_df)
```

```
                           Mean  Median   Mode  Standard Deviation  \
age                   53.720168    54.0   54.0            9.358203
sex                    0.763866     1.0    1.0            0.424884
chest pain type        3.232773     4.0    4.0            0.935480
resting bp s         132.153782   130.0  120.0           18.368823
cholesterol          210.363866   229.0    0.0          101.420489
fasting blood sugar    0.213445     0.0    0.0            0.409912
resting ecg            0.698319     0.0    0.0            0.870359
max heart rate       139.732773   140.5  150.0           25.517636
exercise angina        0.387395     0.0    0.0            0.487360
oldpeak                0.922773     0.6    0.0            1.086337
ST slope               1.624370     2.0    2.0            0.610459
target                 0.528571     1.0    1.0            0.499393

                         Variance
age                     87.575960
sex                      0.180527
chest pain type          0.875124
resting bp s           337.413674
cholesterol          10286.115598
fasting blood sugar      0.168028
resting ecg              0.757525
max heart rate         651.149724
exercise angina          0.237520
oldpeak                  1.180129
ST slope                 0.372660
target                   0.249393
```

Conduct a Hypothesis Test Let's conduct a hypothesis test to determine if the average cholesterol level (chol) is significantly different from a chosen value (e.g., 200).

In [23]:

```
from scipy import stats

# Set the chosen value
chosen_value = 200

# Perform a one-sample t-test
t_statistic, p_value = stats.ttest_1samp(data['cholesterol'], chosen_value)

# Display the results
print(f"T-statistic: {t_statistic}, P-value: {p_value}")
```

```
T-statistic: 3.5250846500630435, P-value: 0.00043944322756286143
```

Compute a 95% Confidence Interval We will compute a 95% confidence interval for the mean of the cholesterol levels.

In [26]:

```
import numpy as np

# Calculate the mean and standard error
mean_chol = data['cholesterol'].mean()
std_error = stats.sem(data['cholesterol'])
```

```
# Calculate the confidence interval
confidence_level = 0.95
degrees_freedom = len(data['cholesterol']) - 1
confidence_interval = stats.t.interval(confidence_level, degrees_freedom, mean_chol

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

```
95% Confidence Interval for Cholesterol: (np.float64(204.59563478483102), np.float64
(216.13209630760593))
```

In [ ]:
```
Exploring Regression Analysis on the Heart Disease Dataset
Perform Linear Regression Analysis
We will analyze how age impacts cholesterol levels.
```

Data Preprocessing We will check for missing values and ensure that the relevant features are in the correct format.

In [27]:
```
# Check for missing values
print(data.isnull().sum())

# Display the data types
print(data.dtypes)
```

```
age                    0
sex                    0
chest pain type        0
resting bp s           0
cholesterol            0
fasting blood sugar    0
resting ecg            0
max heart rate         0
exercise angina        0
oldpeak                0
ST slope               0
target                 0
dtype: int64
age                      int64
sex                      int64
chest pain type          int64
resting bp s             int64
cholesterol              int64
fasting blood sugar      int64
resting ecg              int64
max heart rate           int64
exercise angina          int64
oldpeak                float64
ST slope                 int64
target                   int64
dtype: object
```

Linear Regression Analysis Let's analyze how age impacts chol (cholesterol levels). We'll perform a linear regression analysis.

In [29]:
```
import statsmodels.api as sm

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

# Add a constant to the model (intercept)
```

```
X = sm.add_constant(X)

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

# Display the model summary
model_summary = model.summary()
print(model_summary)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            cholesterol   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     2.571
Date:                Sun, 08 Sep 2024   Prob (F-statistic):              0.109
Time:                        13:56:40   Log-Likelihood:                -7183.7
No. Observations:                1190   AIC:                         1.437e+04
Df Residuals:                    1188   BIC:                         1.438e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        237.4200     17.127     13.862      0.000     203.818     271.022
age           -0.5036      0.314     -1.604      0.109      -1.120       0.113
==============================================================================
Omnibus:                      109.698   Durbin-Watson:                   0.789
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              144.205
Skew:                          -0.755   Prob(JB):                     4.86e-32
Kurtosis:                       3.792   Cond. No.                         318.
==============================================================================

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

Model Summary Interpretation Coefficients: The coefficient for age indicates how much cholesterol levels change with a one-year increase in age. P-values: A p-value < 0.05 suggests that the relationship is statistically significant. R-squared: This value indicates the proportion of the variance in cholesterol levels that can be explained by age. Visualizations We'll create visualizations to illustrate the relationship between age and cholesterol levels along with the regression line.

In [31]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style
sns.set(style="whitegrid")

# Create a scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='cholesterol', data=data, color='blue', label='Data Poin

# Plot the regression line
plt.plot(data['age'], model.predict(X), color='red', label='Regression Line')

# Add titles and labels
plt.title('Age vs Cholesterol Levels with Regression Line')
plt.xlabel('Age')
plt.ylabel('Cholesterol Levels')
```

```
plt.legend()
plt.show()
```