Exercise 1

1. Load the dataset into Python using pandas:

In [10]:
```
import pandas as pd
data = pd.read_csv('healthcare-dataset-stroke-data 3.csv')
data.head()
```

Out[10]:

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
0	9046	67.0	0	1	228.69	36.6	1
1	51676	61.0	0	0	202.21	NaN	1
2	31112	80.0	0	1	105.92	32.5	1
3	60182	49.0	0	0	171.23	34.4	1
4	1665	79.0	1	0	174.12	24.0	1

2. Calculate descriptive statistics:

In [11]:
```
features = ['age', 'avg_glucose_level', 'bmi']# Calculate descriptive statistics for age, avg_glucose_level, and bmi

mean_values = data[features].mean()
median_values = data[features].median()
mode_values = data[features].mode().iloc[0]  # The first mode value
std_values = data[features].std()
variance_values = data[features].var()

print("Mean:\n", mean_values)
print("\nMedian:\n", median_values)
print("\nMode:\n", mode_values)
print("\nStandard Deviation:\n", std_values)
print("\nVariance:\n", variance_values)
```

```
Mean:
 age                   43.226614
avg_glucose_level    106.147677
bmi                   28.893237
dtype: float64

Median:
 age                  45.000
avg_glucose_level    91.885
bmi                  28.100
dtype: float64

Mode:
 age                  78.00
avg_glucose_level    93.88
bmi                  28.70
Name: 0, dtype: float64

Standard Deviation:
 age                  22.612647
avg_glucose_level    45.283560
bmi                   7.854067
dtype: float64

Variance:
 age                   511.331792
avg_glucose_level    2050.600820
bmi                    61.686364
dtype: float64
```

3. Conduct a hypothesis test:

In [12]:
```
from scipy import stats
chosen_value = 120
t_stat, p_value = stats.ttest_1samp(data['avg_glucose_level'].dropna(), chosen_value)

print(f"T-statistic: {t_stat}, P-value: {p_value}")
if p_value < 0.05:
    print("Reject the null hypothesis: The mean avg_glucose_level is significantly different from 120 mg/dL.")
else:
    print("Fail to reject the null hypothesis: No significant difference from 120 mg/dL.")
```

```
T-statistic: -21.867165560192404, P-value: 2.10116765029635e-101
Reject the null hypothesis: The mean avg_glucose_level is significantly different from 120 mg/dL.
```

4. Compute a 95% confidence interval for the mean of a selected feature:

In [13]:
```
import numpy as np

mean_bmi = data['bmi'].mean()
std_error = stats.sem(data['bmi'].dropna())

confidence_interval = stats.t.interval(0.95, len(data['bmi'].dropna())-1, loc=mean_bmi, scale=std_error)

print(f"95% confidence interval for BMI: {confidence_interval}")
```

```
95% confidence interval for BMI: (np.float64(28.6734745690652), np.float64(29.112999254524134))
```

Exercise 2

1. Linear Regression Analysis on the Stroke Prediction Dataset

In [17]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
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 r2_score

# Load dataset
data = pd.read_csv('healthcare-dataset-stroke-data 3.csv')

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

Out[17]:

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
0	9046	67.0	0	1	228.69	36.6	1
1	51676	61.0	0	0	202.21	NaN	1
2	31112	80.0	0	1	105.92	32.5	1
3	60182	49.0	0	0	171.23	34.4	1
4	1665	79.0	1	0	174.12	24.0	1

2. Preprocessing the data:

In [18]:
```
data = data[['age', 'avg_glucose_level', 'bmi']]
data = data.dropna()
data.describe()
```

Out[18]:

	age	avg_glucose_level	bmi
count	4909.000000	4909.000000	4909.000000
mean	42.865374	105.305150	28.893237
std	22.555115	44.424341	7.854067
min	0.080000	55.120000	10.300000
25%	25.000000	77.070000	23.500000
50%	44.000000	91.680000	28.100000
75%	60.000000	113.570000	33.100000
max	82.000000	271.740000	97.600000

3. Perform Linear Regression (BMI vs. Glucose Level):

In [19]:
```
X_bmi = data['bmi']
y_glucose = data['avg_glucose_level']
X_bmi = sm.add_constant(X_bmi)
model_bmi_glucose = sm.OLS(y_glucose, X_bmi).fit()
print(model_bmi_glucose.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:      avg_glucose_level   R-squared:                       0.031
Model:                            OLS   Adj. R-squared:                  0.031
Method:                 Least Squares   F-statistic:                     155.9
Date:                Thu, 05 Sep 2024   Prob (F-statistic):           2.98e-35
Time:                        17:28:07   Log-Likelihood:                -25512.
No. Observations:                4909   AIC:                         5.103e+04
Df Residuals:                    4907   BIC:                         5.104e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         76.6234      2.380     32.193      0.000      71.957      81.290
bmi            0.9927      0.079     12.488      0.000       0.837       1.149
==============================================================================
Omnibus:                     1201.714   Durbin-Watson:                   1.935
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2384.482
Skew:                           1.485   Prob(JB):                         0.00
Kurtosis:                       4.685   Cond. No.                         114.
==============================================================================

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

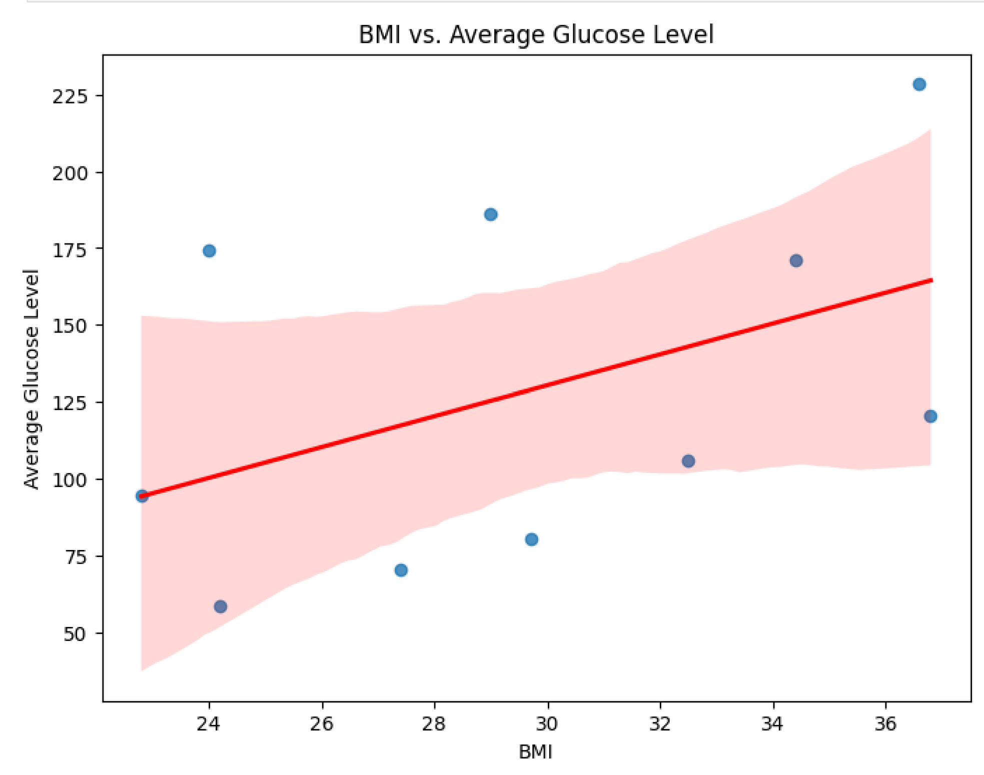
4. Perform Linear Regression (Age vs. BMI):

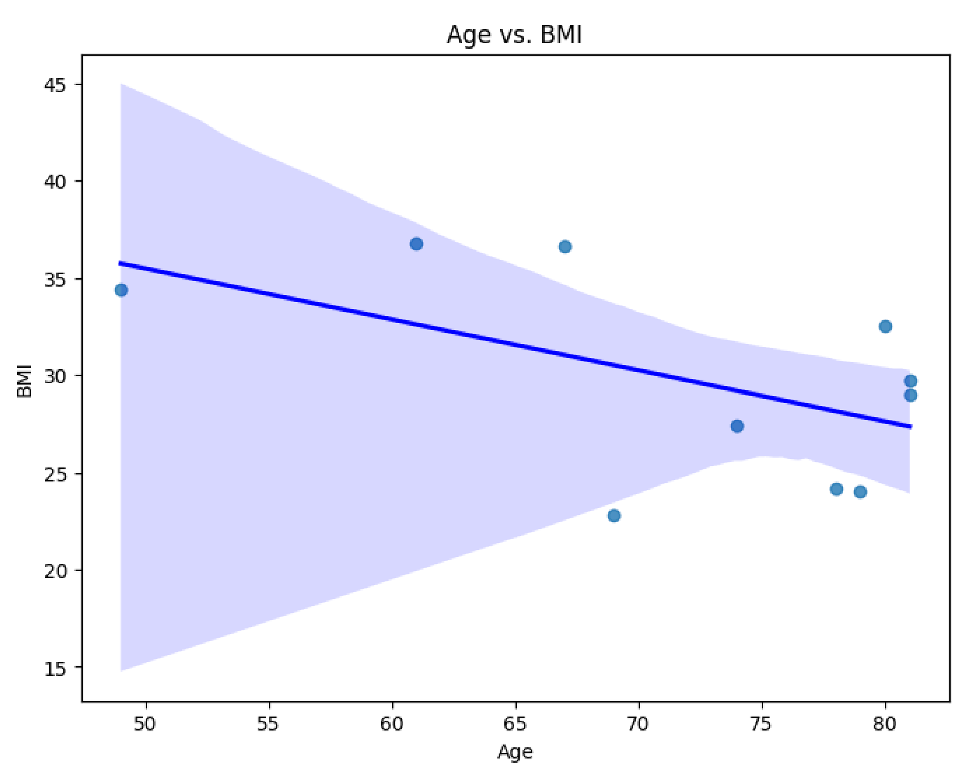
In [20]:
```
X_age = data['age']
y_bmi = data['bmi']
X_age = sm.add_constant(X_age)
model_age_bmi = sm.OLS(y_bmi, X_age).fit()
print(model_age_bmi.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    bmi   R-squared:                       0.111
Model:                            OLS   Adj. R-squared:                  0.111
Method:                 Least Squares   F-statistic:                     613.6
Date:                Thu, 05 Sep 2024   Prob (F-statistic):          9.52e-128
Time:                        17:29:30   Log-Likelihood:                -16793.
No. Observations:                4909   AIC:                         3.359e+04
Df Residuals:                    4907   BIC:                         3.360e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         23.9168      0.227    105.360      0.000      23.472      24.362
age            0.1161      0.005     24.772      0.000       0.107       0.125
==============================================================================
Omnibus:                     1519.890   Durbin-Watson:                   2.016
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6828.081
Skew:                           1.441   Prob(JB):                         0.00
Kurtosis:                       8.008   Cond. No.                         104.
==============================================================================

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

5. Visualizing the relationships:

In [26]:
```
df_1=data.head(10)
plt.figure(figsize=(8, 6))
sns.regplot(x='bmi', y='avg_glucose_level', data=df_1, line_kws={"color":"red"})
plt.title('BMI vs. Average Glucose Level')
plt.xlabel('BMI')
plt.ylabel('Average Glucose Level')
plt.show()

plt.figure(figsize=(8, 6))
sns.regplot(x='age', y='bmi', data=df_1, line_kws={"color":"blue"})
plt.title('Age vs. BMI')
plt.xlabel('Age')
plt.ylabel('BMI')
plt.show()
```






```
In [ ]:
```