In [1]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

In [2]:
```
my_data = pd.read_csv('bmi.csv')
my_data
```

Out[2]:

	Age	Height	Weight	Bmi	BmiClass
0	61	1.85	109.30	31.935720	Obese Class 1
1	60	1.71	79.02	27.023700	Overweight
2	60	1.55	74.70	31.092612	Obese Class 1
3	60	1.46	35.90	16.841809	Underweight
4	60	1.58	97.10	38.896010	Obese Class 2
...	...	...	...	...	...
736	34	1.86	95.70	27.662157	Overweight
737	44	1.91	106.90	29.302925	Overweight
738	25	1.82	88.40	26.687598	Overweight
739	35	1.88	98.50	27.868945	Overweight
740	45	1.93	109.90	29.504148	Overweight

741 rows × 5 columns

In [3]:
```
X = my_data.drop(columns=['Bmi'])
y = my_data['Bmi']
print(my_data.dtypes)
```

```
Age            int64
Height       float64
Weight       float64
Bmi          float64
BmiClass      object
dtype: object
```

In [4]:

```python
data_encoded = pd.get_dummies(my_data, columns=['BmiClass'])
print(data_encoded.head())
onehot_encoder = OneHotEncoder()
encoded_column = onehot_encoder.fit_transform(my_data['BmiClass'].values.r
encoded_df = pd.DataFrame(encoded_column, columns=onehot_encoder.get_featu
data_encoded = pd.concat([my_data.drop(columns=['BmiClass']), encoded_df],
print(data_encoded.head())
```

```
   Age  Height  Weight        Bmi  BmiClass_Normal Weight  \
0   61    1.85  109.30  31.935720                   False   
1   60    1.71   79.02  27.023700                   False   
2   60    1.55   74.70  31.092612                   False   
3   60    1.46   35.90  16.841809                   False   
4   60    1.58   97.10  38.896010                   False   

   BmiClass_Obese Class 1  BmiClass_Obese Class 2  BmiClass_Obese Class 3  
\
0                    True                   False                   False   
1                   False                   False                   False   
2                    True                   False                   False   
3                   False                   False                   False   
4                   False                    True                   False   

   BmiClass_Overweight  BmiClass_Underweight  
0                False                 False  
1                 True                 False  
2                False                 False  
3                False                  True  
4                False                 False  
   Age  Height  Weight        Bmi  BmiClass_Normal Weight  \
0   61    1.85  109.30  31.935720                     0.0   
1   60    1.71   79.02  27.023700                     0.0   
2   60    1.55   74.70  31.092612                     0.0   
3   60    1.46   35.90  16.841809                     0.0   
4   60    1.58   97.10  38.896010                     0.0   

   BmiClass_Obese Class 1  BmiClass_Obese Class 2  BmiClass_Obese Class 3  
\
0                     1.0                     0.0                     0.0   
1                     0.0                     0.0                     0.0   
2                     1.0                     0.0                     0.0   
3                     0.0                     0.0                     0.0   
4                     0.0                     1.0                     0.0   

   BmiClass_Overweight  BmiClass_Underweight  
0                  0.0                   0.0  
1                  1.0                   0.0  
2                  0.0                   0.0  
3                  0.0                   1.0  
4                  0.0                   0.0  
```

In [5]:
```
X = data_encoded.drop(columns=['Bmi'])
y = data_encoded['Bmi']
X
```

Out[5]:

	Age	Height	Weight	BmiClass_Normal Weight	BmiClass_Obese Class 1	BmiClass_Obese Class 2	BmiClass_ C
0	61	1.85	109.30	0.0	1.0	0.0	
1	60	1.71	79.02	0.0	0.0	0.0	
2	60	1.55	74.70	0.0	1.0	0.0	
3	60	1.46	35.90	0.0	0.0	0.0	
4	60	1.58	97.10	0.0	0.0	1.0	
...	...	...	...	...	...	...	
736	34	1.86	95.70	0.0	0.0	0.0	
737	44	1.91	106.90	0.0	0.0	0.0	
738	25	1.82	88.40	0.0	0.0	0.0	
739	35	1.88	98.50	0.0	0.0	0.0	
740	45	1.93	109.90	0.0	0.0	0.0	

741 rows × 9 columns

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r


model = LinearRegression()


model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 0.7690052384587545
```

In [7]:
```
r_squared = model.score(X_test, y_test)
print("R-squared:", r_squared)
```

```
R-squared: 0.9892416129298092
```

FINE TUNING

In [8]:
```
param_grid = {
    'copy_X': [True, False],
    'fit_intercept': [True, False],
    'n_jobs': [True, False],
    'positive':[True, False]
}
```

In [9]:
```
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

Out[9]:
```
GridSearchCV(cv=5, estimator=LinearRegression(),
             param_grid={'copy_X': [True, False],
                         'fit_intercept': [True, False],
                         'n_jobs': [True, False], 'positive': [True, Fals
e]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [10]:
```
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

In [11]:
```
best_model.fit(X_train, y_train)
y_pred_best = best_model.predict(X_test)
mse_best = mean_squared_error(y_test, y_pred_best)
print("Fine-Tuned Mean Squared Error:", mse_best)
```

```
Fine-Tuned Mean Squared Error: 0.7690052384586076
```

In [12]:

```
plt.scatter(y_test, y_pred_best)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Predictions vs Actuals")
plt.show()
```