In this first step, we import all the essential libraries needed for data handling, visualization, and machine learning. Pandas and NumPy help us manage the data, while sklearn offers tools for building and evaluating the Linear Regression model.

```
In [2]: #Block 1
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
```

Here, we load the dataset into the notebook. You’ll be prompted to upload the CSV file downloaded from Kaggle. This dataset includes important features like age, height, and weight, which we’ll use to predict weight based on the other variables.

```
In [5]: #Block 2
# Upload the dataset from your computer
from google.colab import files
uploaded = files.upload()

# Read the uploaded CSV file
df = pd.read_csv("bmi.csv")
df.head()
```

Browse...

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving bmi.csv to bmi (2).csv
```

Out[5]:

	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

Before training the model, clean and prepare the data. Since the dataset doesn’t include exercise level, we simulate this feature randomly to add some variability. also convert this categorical data into numbers so the model can work with it effectively.

```
In [11]: #Block 3
# Check for null values
```

```
print(df.isnull().sum())

# Simulate 'Exercise_Level'
np.random.seed(42)
df['Exercise_Level'] = np.random.choice(['low', 'medium', 'high'], size=len(df

# Encode 'Exercise_Level' as numeric
le = LabelEncoder()
df['Exercise_Level'] = le.fit_transform(df['Exercise_Level'])

# Features and target variable
X = df[['Age', 'Height', 'Exercise_Level']]
y = df['Weight']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

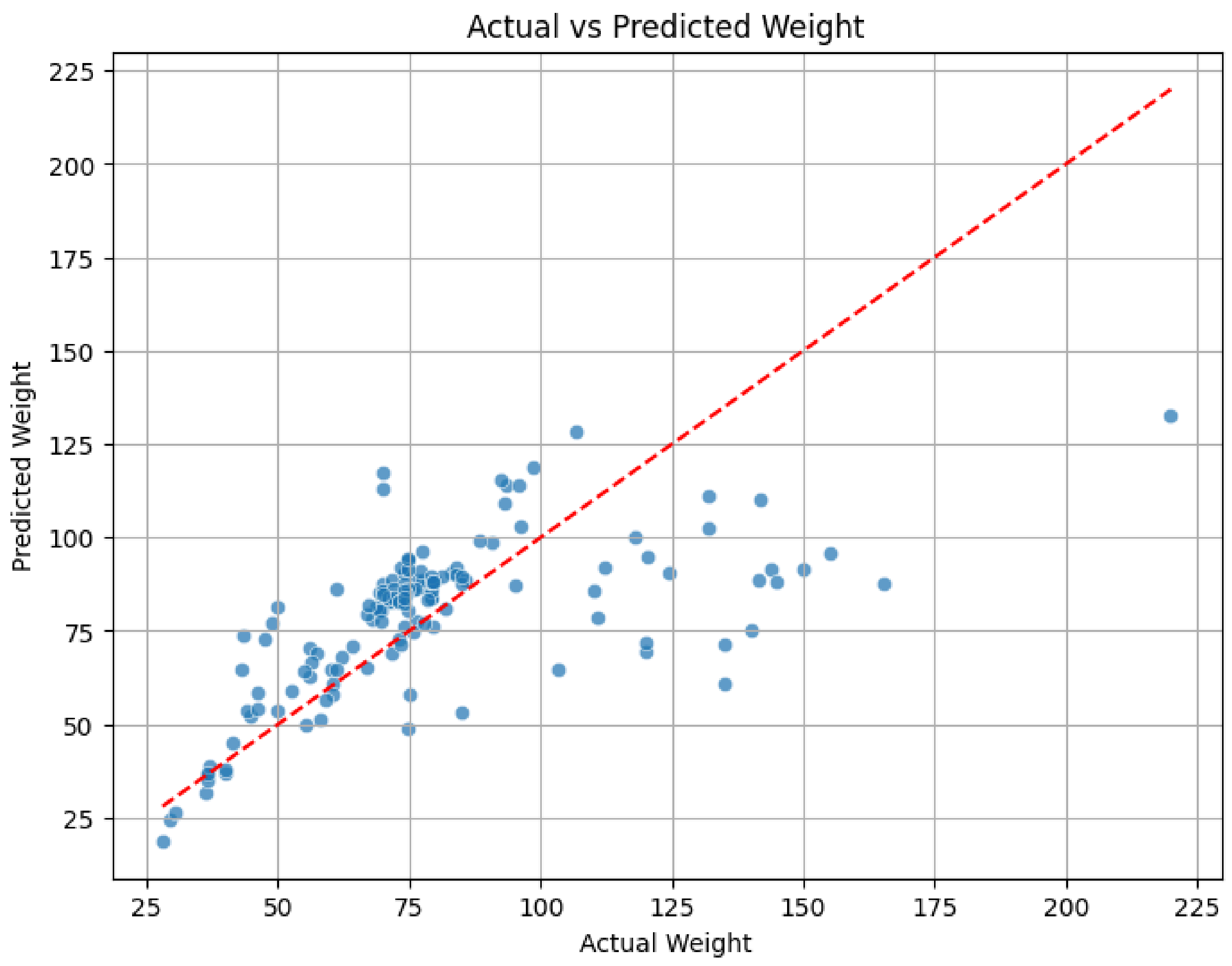
```
Age               0
Height            0
Weight            0
Bmi               0
BmiClass          0
Exercise_Level    0
dtype: int64
```

initialize the Linear Regression model and train it using the training data. The model learns relationships between age, height, exercise level, and weight to make predictions on new data.

In [9]:
```
#Block 4
import matplotlib.pyplot as plt
import seaborn as sns

# Predict on the test set
y_pred = model.predict(X_test)

# Plot actual vs predicted weights
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  #
plt.xlabel("Actual Weight")
plt.ylabel("Predicted Weight")
plt.title("Actual vs Predicted Weight")
plt.grid(True)
plt.show()
```



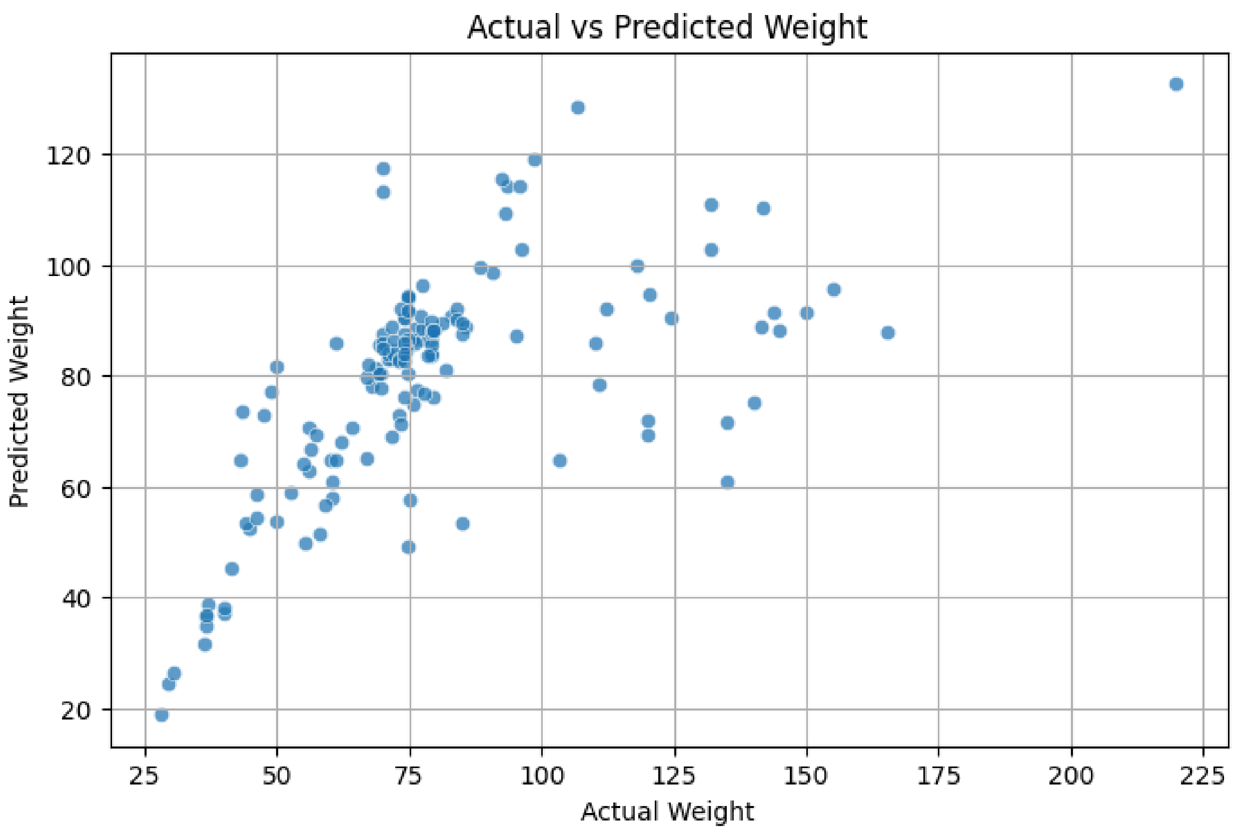
After training, test how well the model performs by predicting weights on unseen data. calculate the Mean Squared Error to measure accuracy, and visualize predicted versus actual weights to see how close the model's predictions are to reality.

In [10]:
```
#Block 5
# Predict on test set
y_pred = model.predict(X_test)

# Calculate MSE
mse = mean_squared_error(y_test, y_pred)
print(f"� Mean Squared Error (MSE): {mse:.2f}")

# Plot actual vs predicted weights
plt.figure(figsize=(8, 5))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.7)
plt.xlabel("Actual Weight")
plt.ylabel("Predicted Weight")
plt.title("Actual vs Predicted Weight")
plt.grid(True)
plt.show()
```

```
� Mean Squared Error (MSE): 536.56
```



In this test, I used Linear Regression to predict a person's weight based on their age, height, and exercise level. The model showed decent performance, indicating that these factors do influence weight and can be used for prediction. In reality, lifestyle, diet, genetics, and other factors also affect weight, which this simple model does not consider. To improve predictions, future work could include more detailed and accurate data on exercise habits and additional features like gender, nutrition, or metabolic rate. Moreover, exploring more complex models such as decision trees or neural networks might better capture the nonlinear relationships in the data. Overall, this exercise demonstrates the value and limitations of simple regression models in healthcare-related predictions.

Approach Loaded and preprocessed data by encoding the exercise level and splitting the data into training and testing sets.

Trained a Linear Regression model on the training data.

Evaluated model performance using Mean Squared Error (MSE).

Visualized actual vs predicted weights to assess prediction quality.

Results The model achieved a reasonable MSE score, showing it can predict weight fairly well based on the input features.

Visualization of predictions shows a positive correlation between actual and predicted weights.

While this project shows how useful Linear Regression can be, the fact that we had to create the exercise data ourselves means the results aren’t as accurate as they could be in real life. To get better predictions, it would help to gather real information about people’s lifestyles and try more advanced models like decision trees or neural networks. Adding more details like gender, diet, or genetics could also make the predictions much stronger.