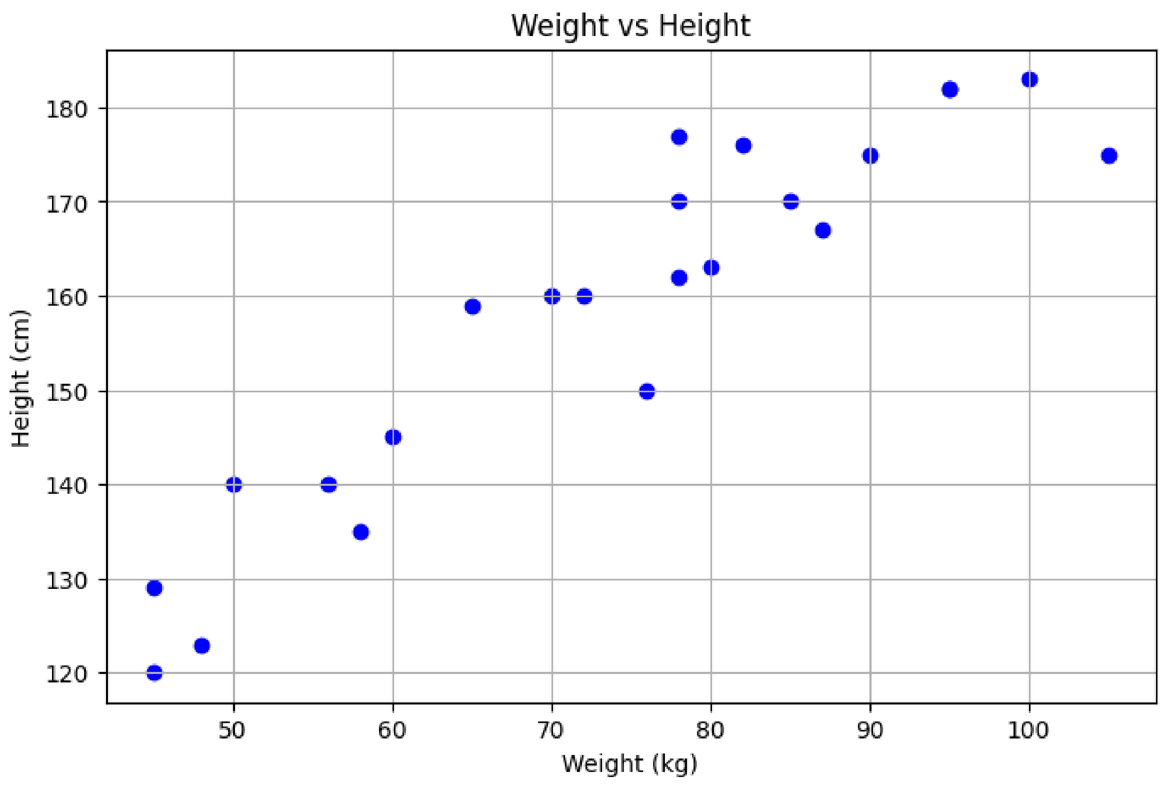
1. Simple Linear Regression Python script using weight as the independent variable (X) and height as the dependent variable (Y) with 30 sample rows with Standardization included

In [11]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# 2. Create a DataFrame
df=pd.read_csv('height-weight.csv')
print("Sample data:\n", df.head(), "\n")

# 3. Plot the raw data
plt.figure(figsize=(8, 5))
plt.scatter(df['Weight'], df['Height'], color='blue')
plt.title("Weight vs Height")
plt.xlabel("Weight (kg)")
plt.ylabel("Height (cm)")
plt.grid(True)
plt.show()

# 4. Define features (X) and target (y)
X = df[['Weight']]  # Independent variable must be 2D
y = df['Height']    # Dependent variable

# 5. Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

# 6. Standardize the features using StandardScaler
scaler = StandardScaler()

# Fit on training data and transform both train and test sets
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Print mean and std used for scaling
print("Scaler Mean:", scaler.mean_)
print("Scaler Scale (Std Dev):", scaler.scale_)

# 7. Create and train the Linear Regression model using scaled data
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# 8. Print learned parameters
print("\nIntercept:", model.intercept_)         # This is the bias term
print("Coefficient (Slope):", model.coef_[0])   # Weight given to the sta

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

# 10. Evaluate the model
```

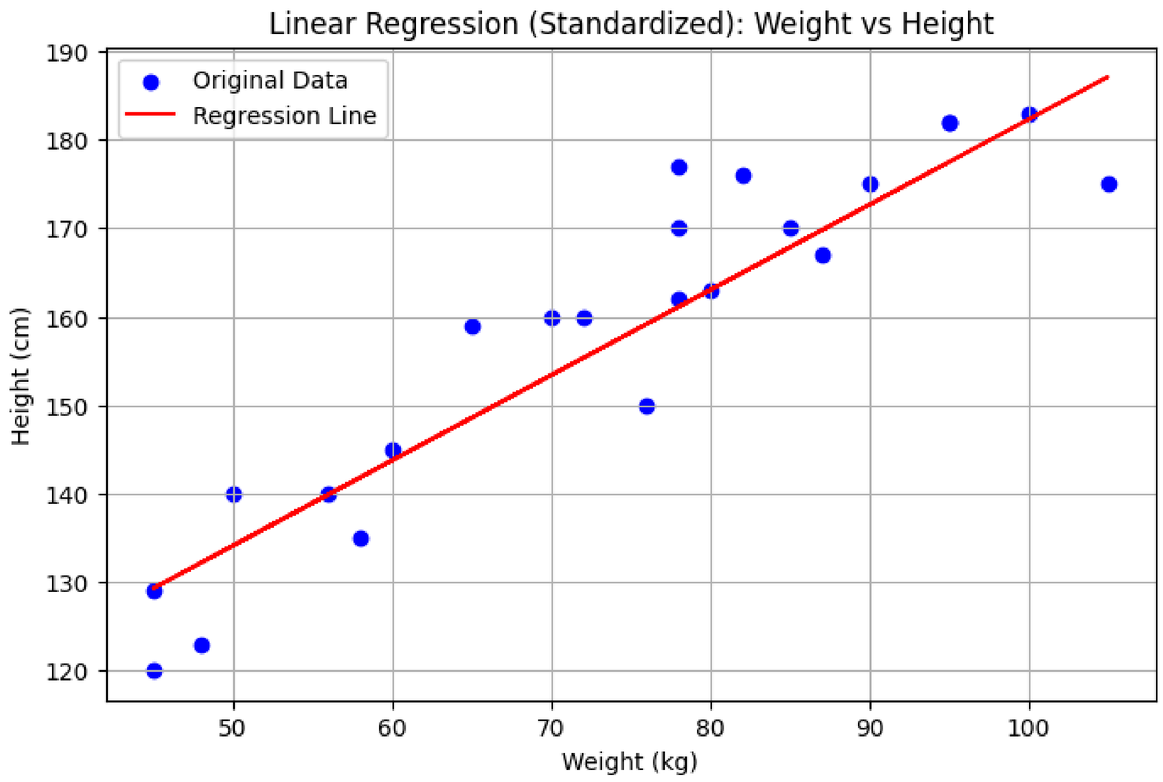
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"\nMean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {np.sqrt(mse):.2f}")
print(f"R² Score: {r2:.2f}")
print(f"Model Accuracy (as R² %): {r2 * 100:.2f}%")


# 11. Compare actual vs predicted values
comparison = pd.DataFrame({
    'Actual Height': y_test.values,
    'Predicted Height': y_pred.round(2)
})
print("\nActual vs Predicted:\n", comparison)

# 12. Plot regression line on the standardized full dataset
# First, standardize full X using the scaler
X_scaled_full = scaler.transform(X)

# Predict using the full dataset (scaled)
y_pred_full = model.predict(X_scaled_full)

# Plotting
plt.figure(figsize=(8, 5))
plt.scatter(X, y, color='blue', label='Original Data')
plt.plot(X, y_pred_full, color='red', label='Regression Line')
plt.title("Linear Regression (Standardized): Weight vs Height")
plt.xlabel("Weight (kg)")
plt.ylabel("Height (cm)")
plt.legend()
plt.grid(True)
plt.show()

# 13. Predicting on New Data (New Weights)
new_weights = pd.DataFrame({'Weight': [55, 75, 90]})  # Replace with actu

# Standardize using the previously fit scaler
new_weights_scaled = scaler.transform(new_weights)

# Predict height
new_heights_pred = model.predict(new_weights_scaled)

# Display results
new_data_results = pd.DataFrame({
    'Weight (kg)': new_weights['Weight'],
    'Predicted Height (cm)': new_heights_pred.round(2)
})
print("\nPredictions for New Records:\n", new_data_results)

#14. Plot New Predictions
print("\nPlotting new Predictions")
plt.figure(figsize=(8, 5))
plt.scatter(X, y, color='blue', label='Training Data')
plt.plot(X, y_pred_full, color='red', label='Regression Line')
plt.scatter(new_weights, new_heights_pred, color='green', marker='x', s=1
plt.title("New Predictions on Regression Line")
plt.xlabel("Weight (kg)")
plt.ylabel("Height (cm)")
plt.legend()
plt.grid(True)
```

```python
plt.show()

# Residual Plot (To Visually Check Errors)
#If the residuals are randomly scattered around 0, your model fits well.
residuals = y_test - y_pred
plt.figure(figsize=(8, 5))
plt.scatter(y_pred, residuals, color='purple')
plt.axhline(y=0, color='black', linestyle='--')
plt.title("Residual Plot")
plt.xlabel("Predicted Height")
plt.ylabel("Residuals (Actual - Predicted)")
plt.grid(True)
plt.show()
```

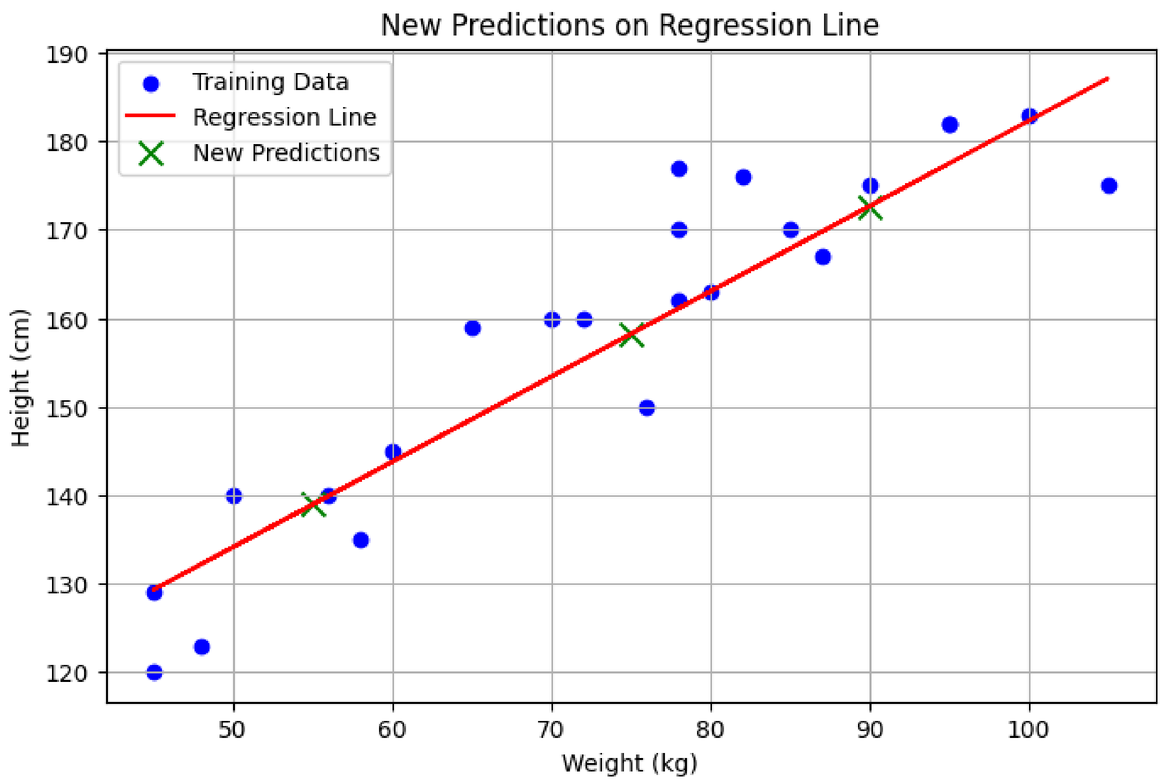
```
Sample data:
   Weight  Height
0      45     120
1      58     135
2      48     123
3      60     145
4      70     160
```

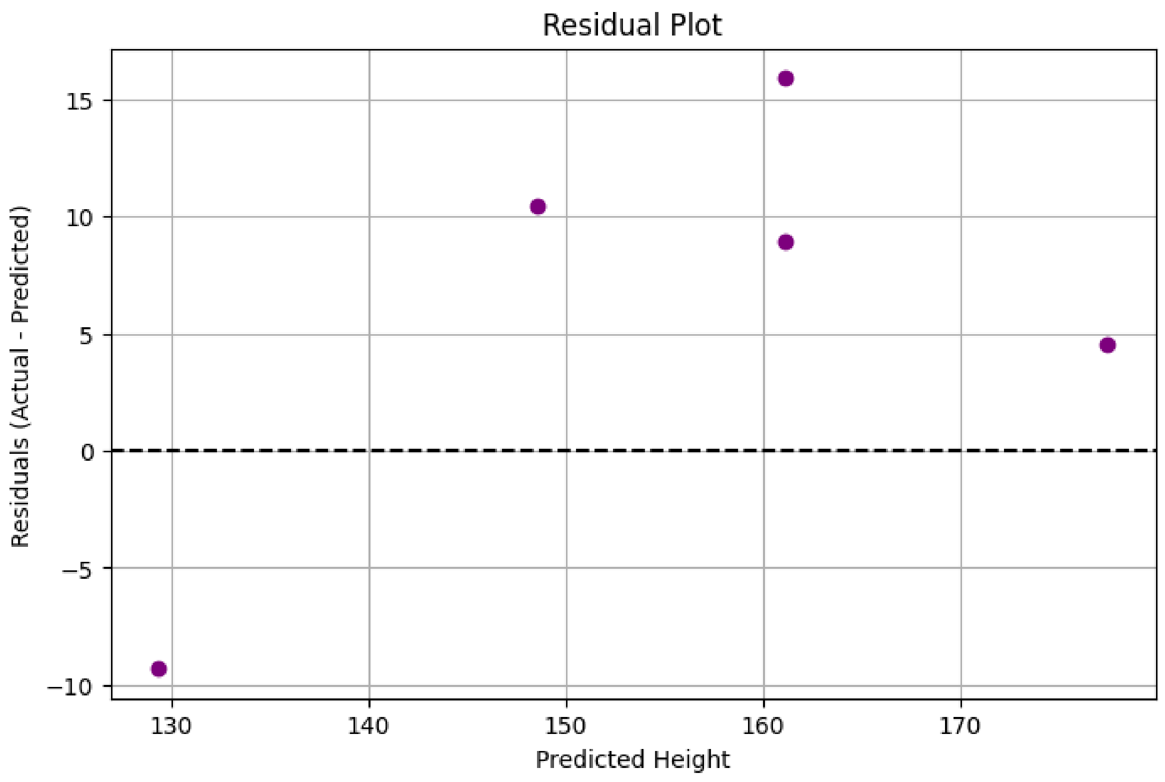


```
Scaler Mean: [74.27777778]
Scaler Scale (Std Dev): [17.68805484]

Intercept: 157.5
Coefficient (Slope): 17.034408719095538

Mean Squared Error (MSE): 109.78
Root Mean Squared Error (RMSE): 10.48
R² Score: 0.78
Model Accuracy (as R² %): 77.70%

Actual vs Predicted:
    Actual Height  Predicted Height
0             177            161.08
1             170            161.08
2             120            129.30
3             182            177.46
4             159            148.57
```



```
Predictions for New Records:
    Weight (kg)  Predicted Height (cm)
0            55                 138.93
1            75                 158.20
2            90                 172.64

Plotting new Predictions
```





```
In [12]: # Cross-validation helps ensure that your model performs well on differen

         from sklearn.model_selection import cross_val_score

         # Perform 5-fold cross-validation on the scaled data
         cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring

         # Convert the negative MSE scores to positive
         cv_scores = -cv_scores

         # Compute the average of the cross-validation scores
         cv_mean = np.mean(cv_scores)
```

```
cv_std = np.std(cv_scores)

# Print the cross-validation result
print(f"\nCross-validation MSE (Mean): {cv_mean:.2f}")
print(f"Cross-validation MSE (Std Dev): {cv_std:.2f}")
print(f"Cross-validation RMSE (Mean): {np.sqrt(cv_mean):.2f}")


# The Adjusted R² score adjusts the R² score for the number of predictors

# For a simple linear regression model (with one feature like weight), th

# Adjusted R² formula: 1 - [(1 - R²) * (n - 1) / (n - p - 1)]
# Where:
# n = number of data points (size of the dataset)
# p = number of predictors (features, 1 in this case for weight)

n = len(X_train)  # Number of data points in training set
p = X_train.shape[1]  # Number of predictors (features)

adj_r2 = 1 - ((1 - r2) * (n - 1)) / (n - p - 1)
print(f"\nAdjusted R²: {adj_r2:.2f}")
```

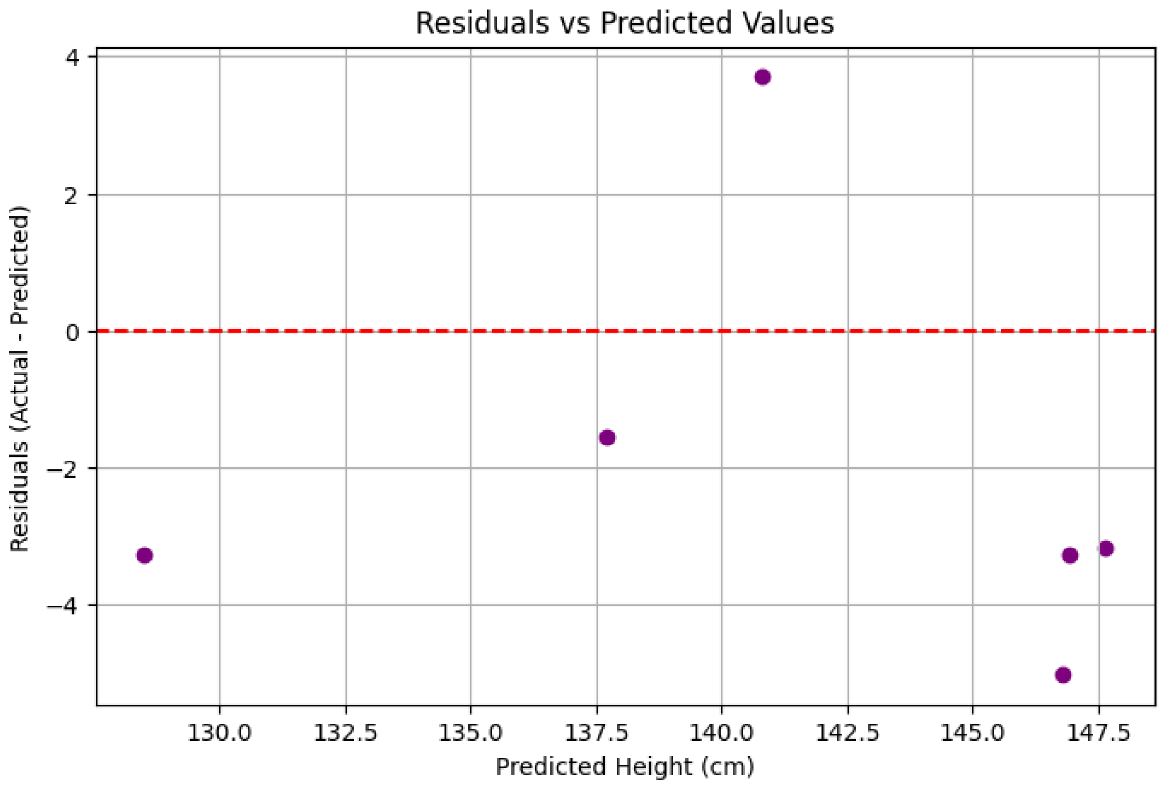
```
Cross-validation MSE (Mean): 50.23
Cross-validation MSE (Std Dev): 25.78
Cross-validation RMSE (Mean): 7.09

Adjusted R²: 0.76
```

2. Multiple Linear Regression example in Python using 2 features — Weight(kg) and Age — to predict Height(cm), with standardization, evaluation, and inline comments for every major step:

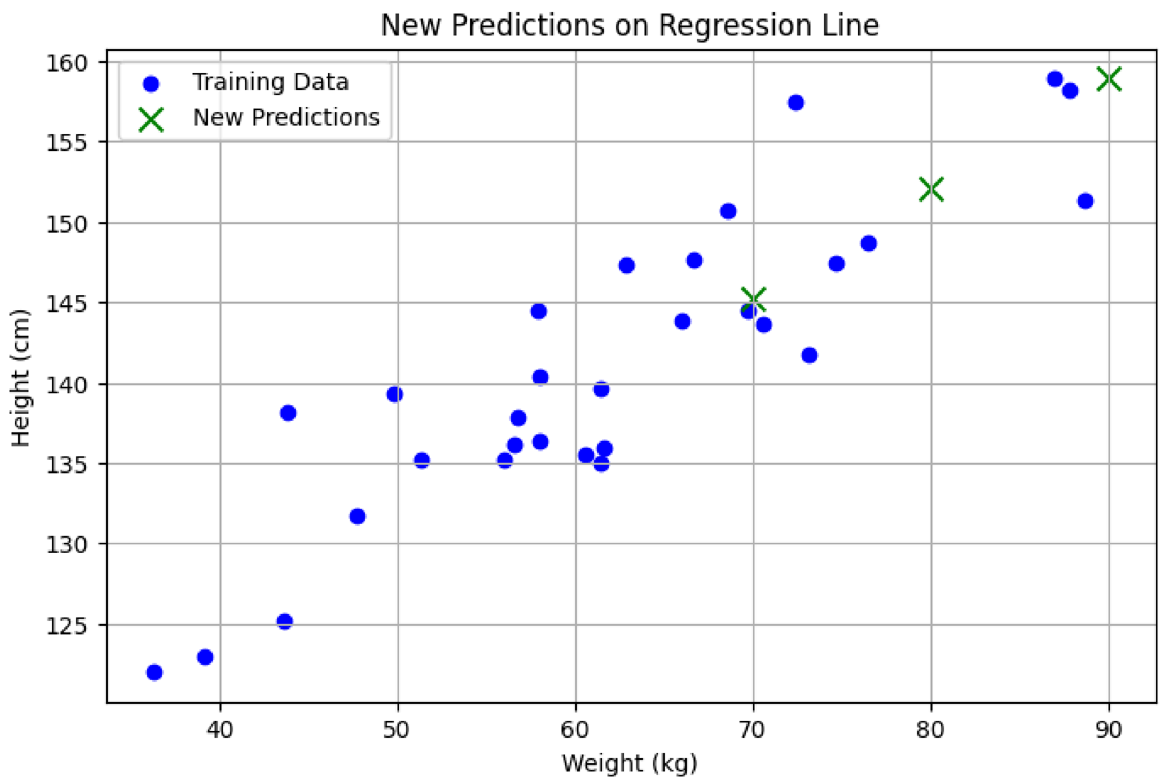
In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

# 1. Generate synthetic data
np.random.seed(42)
weights = np.random.normal(loc=65, scale=15, size=30).round(2)   # Weight
ages = np.random.randint(18, 60, size=30)                          # Age i
# Height depends on weight and age (with some noise)
heights = (weights * 0.7 + ages * 0.2 + np.random.normal(0, 4, 30)).round

# 2. Create a DataFrame
df = pd.DataFrame({
    'Weight(kg)': weights,
    'Age': ages,
    'Height(cm)': heights
})

print("Sample data:\n", df.head(), "\n")
```

```python
# 3. Define features (X) and target (y)
X = df[['Weight(kg)', 'Age']]  # Two features
y = df['Height(cm)']

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

# 5. Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("Feature means after standardization (should be ~0):", X_train_scal
print("Feature std devs (should be ~1):", X_train_scaled.std(axis=0), "\n

# 6. Train the Linear Regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# 7. Output coefficients
print("Intercept (bias term):", model.intercept_)
print("Coefficients (slopes):")
for feature, coef in zip(X.columns, model.coef_):
    print(f"  {feature}: {coef:.4f}")

# 8. Predict using test data
y_pred = model.predict(X_test_scaled)

# 9. Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"\nMean Squared Error: {mse:.2f}")
print(f"R² Score: {r2:.2f}")

# 10. Cross-validation
cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring
cv_scores = -cv_scores  # Convert negative MSE to positive
cv_mean = np.mean(cv_scores)
cv_std = np.std(cv_scores)
print(f"\nCross-validation MSE (Mean): {cv_mean:.2f}")
print(f"Cross-validation RMSE (Mean): {np.sqrt(cv_mean):.2f}")

# 11. Compare actual vs predicted
results = pd.DataFrame({
    'Weight': X_test['Weight(kg)'].values,
    'Age': X_test['Age'].values,
    'Actual Height': y_test.values,
    'Predicted Height': y_pred.round(2)
})

print("\nActual vs Predicted:\n", results)

# 12. Residual plot (for visualizing model fit)
residuals = y_test - y_pred

plt.figure(figsize=(8, 5))
plt.scatter(y_pred, residuals, color='purple')
plt.axhline(y=0, color='red', linestyle='--')
plt.title("Residuals vs Predicted Values")
plt.xlabel("Predicted Height (cm)")
plt.ylabel("Residuals (Actual - Predicted)")
plt.grid(True)
plt.show()

# 13. Test for new records (Example: new weights and ages)
new_data = pd.DataFrame({
    'Weight(kg)': [70, 80, 90],  # New data points
    'Age': [25, 30, 35]
})

# Standardize the new data
new_data_scaled = scaler.transform(new_data)

# Predict heights for new records
new_predictions = model.predict(new_data_scaled)

# Display new predictions
new_results = pd.DataFrame({
    'Weight(kg)': new_data['Weight(kg)'],
    'Age': new_data['Age'],
    'Predicted Height(cm)': new_predictions.round(2)
})

print("\nPredictions for New Records:\n", new_results)

# 14. Plotting new predictions along with original data and regression li
plt.figure(figsize=(8, 5))
plt.scatter(X['Weight(kg)'], y, color='blue', label='Training Data')
plt.scatter(new_data['Weight(kg)'], new_predictions, color='green', marke
plt.title("New Predictions on Regression Line")
plt.xlabel("Weight (kg)")
plt.ylabel("Height (cm)")
plt.legend()
plt.grid(True)
plt.show()
```

```
Sample data:
    Weight(kg)  Age  Height(cm)
0        72.45   37      157.48
1        62.93   45      147.33
2        74.72   24      147.49
3        87.85   25      158.17
4        61.49   52      139.63

Feature means after standardization (should be ~0): [-2.59052039e-16 -2.22
044605e-16]
Feature std devs (should be ~1): [1. 1.]

Intercept (bias term): 141.54625000000001
Coefficients (slopes):
  Weight(kg): 8.8222
  Age: 1.1855

Mean Squared Error: 12.10
R² Score: 0.75

Cross-validation MSE (Mean): 21.47
Cross-validation RMSE (Mean): 4.63

Actual vs Predicted:
    Weight  Age  Actual Height  Predicted Height
0    70.64   38         143.67            146.93
1    56.57   35         136.15            137.70
2    43.63   25         125.22            128.48
3    69.71   51         144.50            147.66
4    57.96   57         144.52            140.82
5    73.14   21         141.77            146.79
```



```
Predictions for New Records:
    Weight(kg)  Age  Predicted Height(cm)
0           70   25                145.21
1           80   30                152.06
2           90   35                158.91
```



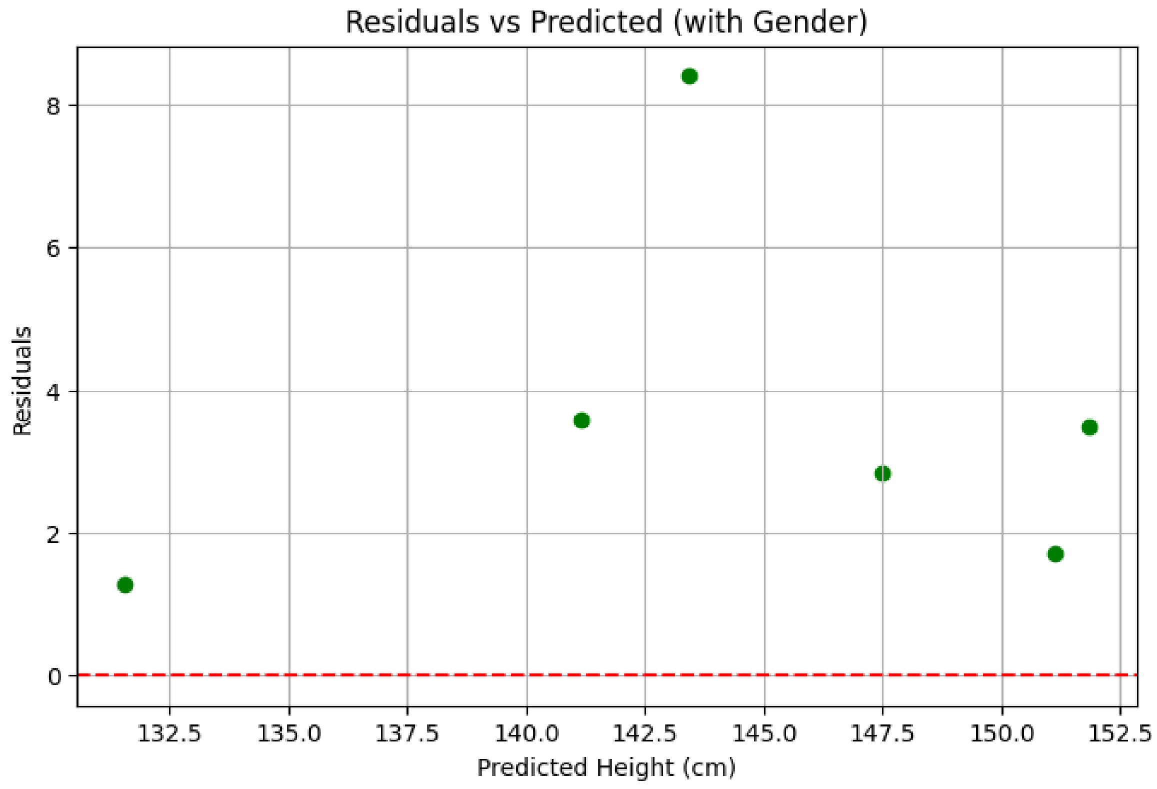
3. Multiple Linear Regression example to include a categorical variable — for instance, Gender — and walk through how to encode it and use it in the regression model.

In [14]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

# 1. Generate synthetic data
np.random.seed(42)
weights = np.random.normal(loc=65, scale=15, size=30).round(2)
ages = np.random.randint(18, 60, size=30)
genders = np.random.choice(['Male', 'Female'], size=30)

# Simulate height: depend on weight, age, and gender (with noise)
# Let's assume 'Male' adds about 5 cm more on average
gender_bias = np.where(genders == 'Male', 5, 0)
heights = (weights * 0.7 + ages * 0.2 + gender_bias + np.random.normal(0,

# 2. Create DataFrame
df = pd.DataFrame({
    'Weight(kg)': weights,
    'Age': ages,
    'Gender': genders,
    'Height(cm)': heights
})

print("Sample data with Gender:\n", df.head(), "\n")
```

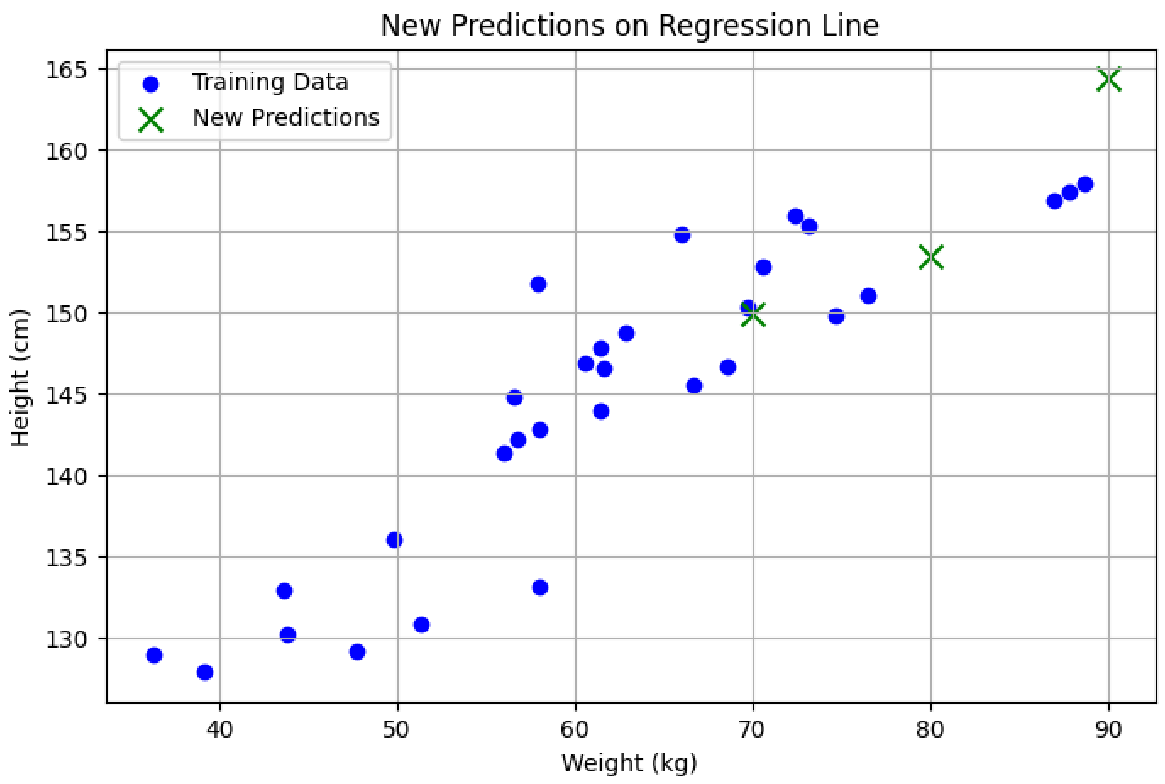
```
# 3. One-hot encode the categorical variable 'Gender'
df_encoded = pd.get_dummies(df, columns=['Gender'], drop_first=True)
# 'Gender_Male' will be 1 if Male, 0 if Female

# 4. Separate features and target
X = df_encoded.drop(columns=['Height(cm)'])
y = df_encoded['Height(cm)']

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

# 6. Standardize only numeric columns (not dummy variables)
numeric_features = ['Weight(kg)', 'Age']
scaler = StandardScaler()

# Fit on training numeric columns
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[numeric_features] = scaler.fit_transform(X_train[numeric_f
X_test_scaled[numeric_features] = scaler.transform(X_test[numeric_feature

# 7. Train the regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# 8. Output coefficients
print("Intercept (bias):", model.intercept_)
print("Coefficients:")
for feature, coef in zip(X.columns, model.coef_):
    print(f"  {feature}: {coef:.4f}")

# 9. Predict and evaluate
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"\nMean Squared Error: {mse:.2f}")
print(f"R² Score: {r2:.2f}")

# 10. Cross-validation
cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring
cv_scores = -cv_scores  # Convert negative MSE to positive
cv_mean = np.mean(cv_scores)
cv_std = np.std(cv_scores)
print(f"\nCross-validation MSE (Mean): {cv_mean:.2f}")
print(f"Cross-validation RMSE (Mean): {np.sqrt(cv_mean):.2f}")

# 11. Compare actual vs predicted
results = pd.DataFrame({
    'Weight': X_test['Weight(kg)'].values,
    'Age': X_test['Age'].values,
    'Gender_Male': X_test['Gender_Male'].values,
    'Actual Height': y_test.values,
    'Predicted Height': y_pred.round(2)
})

print("\nActual vs Predicted with Gender:\n", results)

# 12. Residual plot
```

```
residuals = y_test - y_pred
plt.figure(figsize=(8, 5))
plt.scatter(y_pred, residuals, color='green')
plt.axhline(y=0, color='red', linestyle='--')
plt.title("Residuals vs Predicted (with Gender)")
plt.xlabel("Predicted Height (cm)")
plt.ylabel("Residuals")
plt.grid(True)
plt.show()

# 13. Test for new records (Example: new weights, ages, and genders)
new_data = pd.DataFrame({
    'Weight(kg)': [70, 80, 90],  # New data points
    'Age': [25, 30, 35],
    'Gender': ['Male', 'Female', 'Male']
})

# One-hot encode new gender data
new_data_encoded = pd.get_dummies(new_data, columns=['Gender'], drop_firs

# Standardize the new data (same scaler)
new_data_scaled = new_data_encoded.copy()
new_data_scaled[numeric_features] = scaler.transform(new_data_encoded[num

# Predict heights for new records
new_predictions = model.predict(new_data_scaled)

# Display new predictions
new_results = pd.DataFrame({
    'Weight(kg)': new_data['Weight(kg)'],
    'Age': new_data['Age'],
    'Gender_Male': new_data_encoded['Gender_Male'],
    'Predicted Height(cm)': new_predictions.round(2)
})

print("\nPredictions for New Records:\n", new_results)

# 14. Plotting new predictions along with original data and regression li
plt.figure(figsize=(8, 5))
plt.scatter(X['Weight(kg)'], y, color='blue', label='Training Data')
plt.scatter(new_data['Weight(kg)'], new_predictions, color='green', marke
plt.title("New Predictions on Regression Line")
plt.xlabel("Weight (kg)")
plt.ylabel("Height (cm)")
plt.legend()
plt.grid(True)
plt.show()
```

```
Sample data with Gender:
    Weight(kg)  Age  Gender  Height(cm)
0        72.45   37    Male      155.91
1        62.93   45    Male      148.75
2        74.72   24  Female      149.75
3        87.85   25  Female      157.46
4        61.49   52    Male      147.77

Intercept (bias): 141.506117273088
Coefficients:
  Weight(kg): 9.6636
  Age: 0.6915
  Gender_Male: 3.7302

Mean Squared Error: 18.03
R² Score: 0.68

Cross-validation MSE (Mean): 18.19
Cross-validation RMSE (Mean): 4.27

Actual vs Predicted with Gender:
    Weight  Age  Gender_Male  Actual Height  Predicted Height
0   70.64   38         True         152.81            151.11
1   56.57   35         True         144.74            141.15
2   43.63   25         True         132.86            131.57
3   69.71   51        False         150.33            147.50
4   57.96   57         True         151.82            143.42
5   73.14   21         True         155.33            151.84
```



```
Predictions for New Records:
    Weight(kg)  Age  Gender_Male  Predicted Height(cm)
0           70   25         True                149.89
1           80   30        False                153.41
2           90   35         True                164.38
```



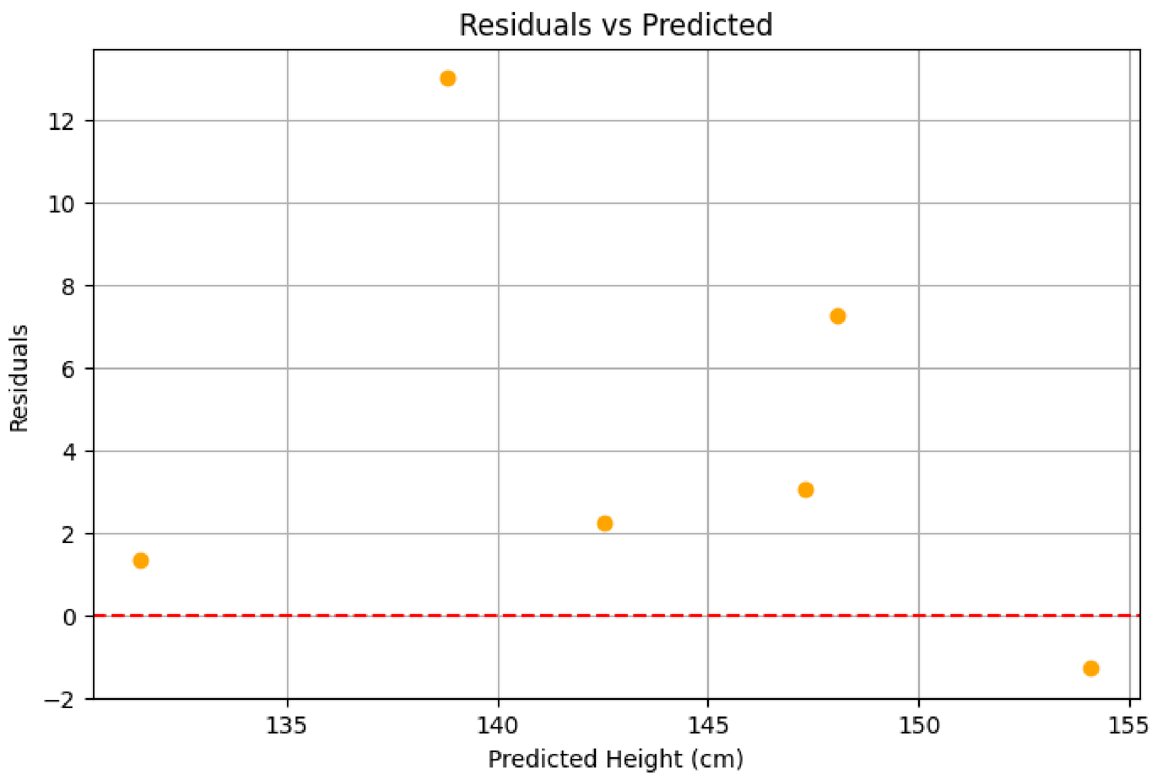
4. Polynomial Regression:

- ✅ Generates synthetic data with gender, weight, and age
- ✅ Applies polynomial feature transformation
- ✅ Splits and standardizes data
- ✅ Trains a model and evaluates it
- ✅ Predicts height for 10 new samples at the end

In [15]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# 1. Generate synthetic data
np.random.seed(42)
weights = np.random.normal(loc=65, scale=15, size=30).round(2)
ages = np.random.randint(18, 60, size=30)
genders = np.random.choice(['Male', 'Female'], size=30)

# Add gender influence: +5 cm for male
gender_bias = np.where(genders == 'Male', 5, 0)
heights = (weights * 0.7 + ages * 0.2 + gender_bias + np.random.normal(0,

# 2. Create DataFrame
df = pd.DataFrame({
    'Weight(kg)': weights,
    'Age': ages,
    'Gender': genders,
    'Height(cm)': heights
})

print("Sample data:\n", df.head(), "\n")

# 3. One-hot encode Gender
df_encoded = pd.get_dummies(df, columns=['Gender'], drop_first=True)

# 4. Feature-target split
X = df_encoded.drop(columns=['Height(cm)'])
y = df_encoded['Height(cm)']

# 5. Polynomial Feature Transformation (degree 2)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)

# 6. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=

# 7. Standardize
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 8. Train the model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# 9. Coefficients
print("\nIntercept:", model.intercept_)
print("Coefficients:")
for name, coef in zip(poly.get_feature_names_out(X.columns), model.coef_)
    print(f"  {name}: {coef:.4f}")

# 10. Predict and evaluate
y_pred = model.predict(X_test_scaled)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"\nMean Squared Error: {mse:.2f}")
print(f"R² Score: {r2:.2f}")

# 11. Actual vs Predicted
X_test_df = pd.DataFrame(X_test, columns=poly.get_feature_names_out(X.col
results = pd.DataFrame({
    'Actual Height': y_test.values,
    'Predicted Height': y_pred.round(2)
})
print("\nActual vs Predicted:\n", results)

# 12. Residual plot
residuals = y_test - y_pred
plt.figure(figsize=(8, 5))
plt.scatter(y_pred, residuals, color='orange')
plt.axhline(y=0, color='red', linestyle='--')
plt.title("Residuals vs Predicted")
plt.xlabel("Predicted Height (cm)")
plt.ylabel("Residuals")
plt.grid(True)
plt.show()
```

```python
# 13. Predicting multiple new records
new_samples = pd.DataFrame({
    'Weight(kg)': [72, 60, 68, 55, 80, 77, 62, 59, 70, 65],
    'Age':        [25, 32, 40, 22, 28, 35, 30, 27, 45, 38],
    'Gender_Male': [1, 0, 1, 0, 1, 1, 0, 0, 1, 0]
})

# Transform and predict
new_samples_poly = poly.transform(new_samples)
new_samples_scaled = scaler.transform(new_samples_poly)
new_preds = model.predict(new_samples_scaled)

# Combine input and prediction
new_results = new_samples.copy()
new_results['Predicted Height (cm)'] = new_preds.round(2)

print("\nPredicted Heights for New Samples:\n", new_results)
```

```
Sample data:
    Weight(kg)  Age  Gender  Height(cm)
0        72.45   37    Male      155.91
1        62.93   45    Male      148.75
2        74.72   24  Female      149.75
3        87.85   25  Female      157.46
4        61.49   52    Male      147.77


Intercept: 143.8375
Coefficients:
  Weight(kg): -2.4436
  Age: 7.1496
  Gender_Male: -5.8697
  Weight(kg)^2: 4.4067
  Weight(kg) Age: 7.6409
  Weight(kg) Gender_Male: 8.8809
  Age^2: -15.3730
  Age Gender_Male: 3.1451
  Gender_Male^2: -5.8697

Mean Squared Error: 40.00
R² Score: 0.29

Actual vs Predicted:
    Actual Height  Predicted Height
0          152.81            154.10
1          144.74            142.52
2          132.86            131.51
3          150.33            147.29
4          151.82            138.80
5          155.33            148.06
```



```
Predicted Heights for New Samples:
    Weight(kg)  Age  Gender_Male  Predicted Height (cm)
0           72   25            1                 150.02
1           60   32            0                 145.13
2           68   40            1                 152.13
3           55   22            0                 140.78
4           80   28            1                 157.90
5           77   35            1                 158.69
6           62   30            0                 145.66
7           59   27            0                 143.79
8           70   45            1                 153.90
9           65   38            0                 147.66
```