Q)Implement a linear regression model using a library like seaborn in Python. Train the model on the training set and evaluate its performance on the testing set using metrics.

A)Seaborn is primarily a data visualization library, and it doesn't include functionality for building regression models. However, you can use other libraries like scikit-learn to implement a linear regression model. Here's a simple example using scikit-learn for linear regression and matplotlib for visualization:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

import seaborn as sns

sns.set(style="whitegrid")

# Generate some example data

np.random.seed(42)

X = 2 \* np.random.rand(100, 1)

y = 4 + 3 \* X + np.random.randn(100, 1)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train the linear regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions on the testing set

y\_pred = model.predict(X\_test)

# Evaluate the model performance

mse = metrics.mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = metrics.r2\_score(y\_test, y\_pred)

# Plot the regression line

plt.scatter(X\_test, y\_test, color='black', label='Actual Data')

plt.plot(X\_test, y\_pred, color='blue', linewidth=3, label='Regression Line')

plt.title('Linear Regression Model')

plt.xlabel('X')

plt.ylabel('y')

plt.legend()

plt.show()

# Display performance metrics

print(f'Mean Squared Error (MSE): {mse}')

print(f'Root Mean Squared Error (RMSE): {rmse}')

print(f'R-squared (R2): {r2}')