# Step 1: Import Libraries

import numpy as np

import pandas as pd

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 mean\_squared\_error, r2\_score

from sklearn.datasets import fetch\_california\_housing

# Step 2: Load Dataset

california = fetch\_california\_housing()

# Convert to DataFrame

df = pd.DataFrame(california.data, columns=california.feature\_names)

df['MedHouseVal'] = california.target # MedHouseVal is the target variable

# Step 3: Explore and Clean Data

print(df.head())

print(df.describe())

print(df.info())

# Check for missing values

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

# Step 4: Split Data into Training and Testing Sets

# Define features (X) and target (y)

X = df.drop('MedHouseVal', axis=1)

y = df['MedHouseVal']

# Split the data

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

# Step 5: Build and Train the Linear Regression Model

# Initialize the model

model = LinearRegression()

# Fit the model

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

# Step 6: Evaluate the Model

# Make predictions

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

# Calculate evaluation metrics

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

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

print(f"Mean Squared Error: {mse}")

print(f"R^2 Score: {r2}")

# Step 7: Make Predictions

# Make a prediction for a single example

example = X\_test.iloc[0].values.reshape(1, -1)

predicted\_value = model.predict(example)

print(f"Predicted value: {predicted\_value}")

print(f"Actual value: {y\_test.iloc[0]}")

# Optional: Visualize the relationship between actual and predicted values

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, alpha=0.3)

plt.xlabel("Actual Values")

plt.ylabel("Predicted Values")

plt.title("Actual vs Predicted Home Values")

plt.show()