```
import numpy as np
import matplotlib.pyplot as plt

# Radial Kernel Bias Function
def radial_kernel(x0, X, tau):
    return np.exp(np.sum((X - x0) ** 2, axis=1) / (-2 * tau * tau))

# Local Regression Function
def local_regression(x0, X, Y, tau):
    # add bias term
    x0 = np.r_[1, x0]  # Add one to avoid the loss in information
    X = np.c_[np.ones(len(X)), X]  # Add bias to X

    # fit model: normal equations with kernel
    xw = X.T * radial_kernel(x0, X, tau)  # XTranspose * W
    beta = np.linalg.pinv(xw @ X) @ xw @ Y  # @ Matrix Multiplication or Dot Product

    # predict value
    return x0 @ beta  # @ Matrix Multiplication or Dot Product for prediction

n = 1000

# Generate dataset
X = np.linspace(-3, 3, num=n)
print("The Data Set (10 Samples) X:\n", X[1:10])

Y = np.log(np.abs(X ** 2 - 1) + 0.5)
print("The Fitting Curve Data Set (10 Samples) Y:\n", Y[1:10])

# Jitter X
X += np.random.normal(scale=0.1, size=n)
print("Jitter (10 Samples) X:\n", X[1:10])

domain = np.linspace(-3, 3, num=300)
print("Xo Domain Space (10 Samples):\n", domain[1:10])

def plot_lwr(tau):
    # Prediction through regression
    predictions = [local_regression(x0, X, Y, tau) for x0 in domain]
    plt.scatter(X, Y, color='blue', alpha=0.3, s=20)
    plt.plot(domain, predictions, color='red', linewidth=3)
    plt.show()

# Plotting the curves with different tau
plot_lwr(10.0)
plot_lwr(1.0)
plot_lwr(0.1)
plot_lwr(0.01)
```