Aim:

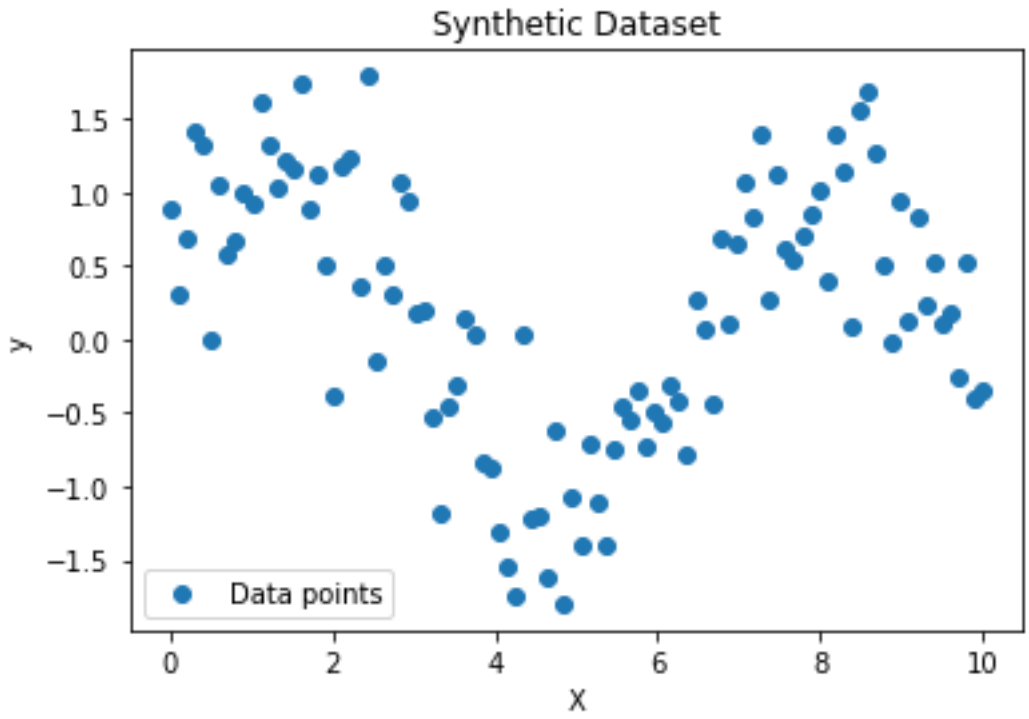
Understand and analyse the concept of Regression algorithm techniques.

Program:

Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

In [1]:

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


# Generate synthetic data
np.random.seed(0)
X = np.linspace(0, 10, 100)
y = np.sin(X) + np.random.normal(scale=0.5, size=X.shape)


plt.scatter(X, y, label='Data points')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Synthetic Dataset')
plt.legend()
plt.show()
```

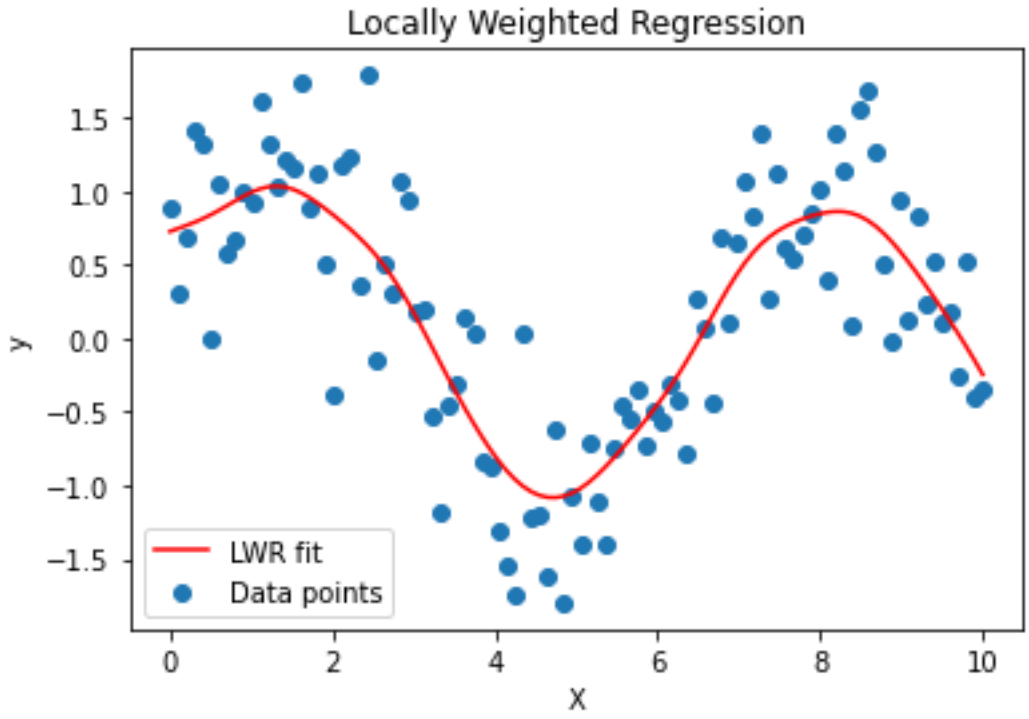


In [3]:

```
def locally_weighted_regression(X, y, x_query, tau):
    """
    Perform Locally Weighted Regression.

    Parameters:
    - X: array-like, shape (n_samples,)
    - y: array-like, shape (n_samples,)
    - x_query: scalar or array-like, the point(s) at which to evaluate the regression
    - tau: bandwidth parameter (controls the locality of the weighting)

    Returns:
    - y_pred: array-like, shape (n_samples,)
    """
    m = X.shape[0]
    y_pred = np.zeros_like(x_query)
```

```
    for i, x_q in enumerate(x_query):
        # Calculate weights
        weights = np.exp(- (X - x_q) ** 2 / (2 * tau ** 2))

        # Form the weighted design matrix
        W = np.diag(weights)

        # Fit the linear model
        X_ = np.vstack([np.ones(m), X]).T
        theta = np.linalg.inv(X_.T @ W @ X_) @ X_.T @ W @ y

        # Predict the value at x_query
        y_pred[i] = np.array([1, x_q]) @ theta

    return y_pred
In [4]:
# Parameters
tau = 0.5  # Bandwidth parameter

# Predict values
x_query = np.linspace(0, 10, 100)
y_pred = locally_weighted_regression(X, y, x_query, tau)

# Plot results
plt.scatter(X, y, label='Data points')
plt.plot(x_query, y_pred, color='red', label='LWR fit')
```

```
plt.xlabel('X')
plt.ylabel('y')
plt.title('Locally Weighted Regression')
plt.legend()
plt.show()
```



In [ ]: