In [ ]:

```
#PRACTICAL 5 – Simple Linear Regression & Assumptions
```

In [ ]:

```
#Name: Ritika R. Junekar
#Sub: PD
#Roll_No:29
```

In [1]:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression  # optional comparison
```

In [3]:

```
plt.style.use('ggplot')
```

In [5]:

```
# 1. Create Sample Data
np.random.seed(42)
X_data = np.random.rand(100) * 10
y_data = 5 + 1.5 * X_data + np.random.randn(100) * 2

df = pd.DataFrame({'Feature_X': X_data, 'Target_Y': y_data})
```

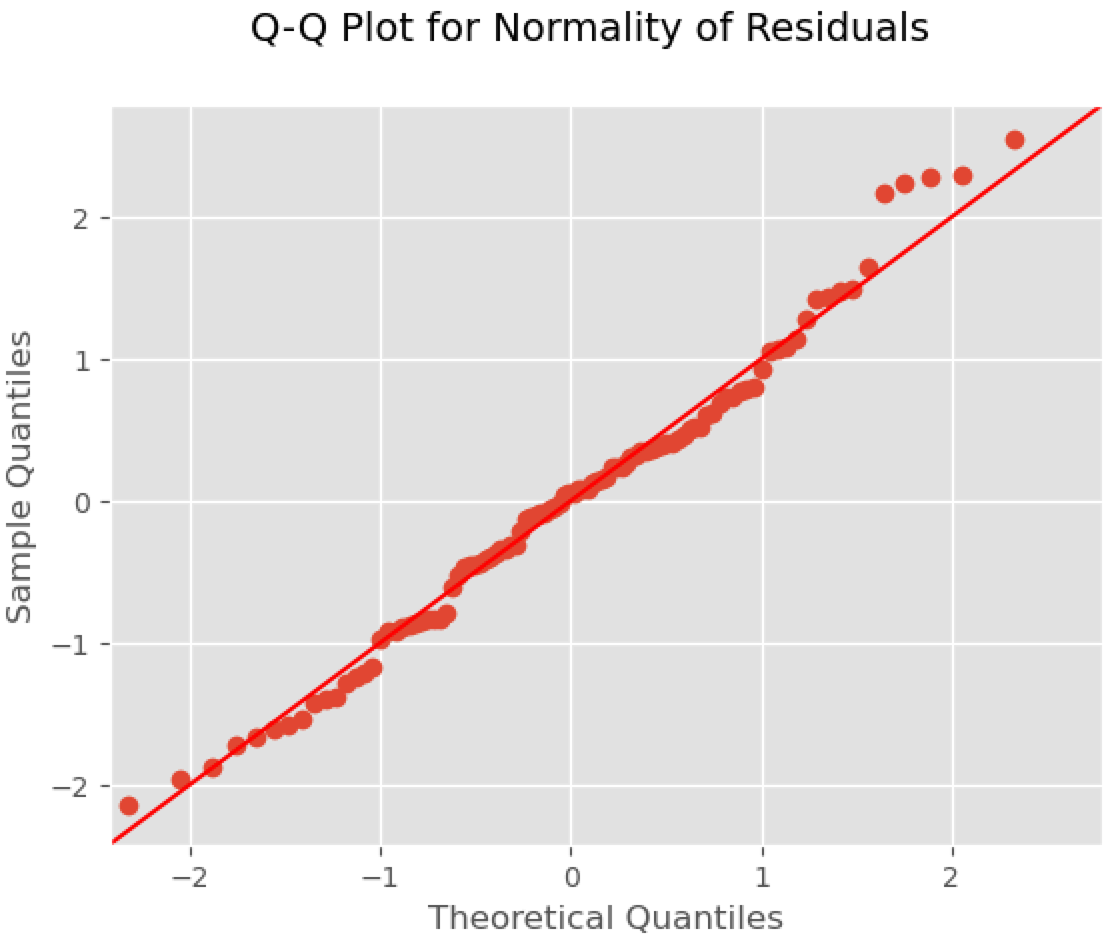
In [7]:

```
# --- statsmodels OLS ---
X_sm = sm.add_constant(df['Feature_X'])
model_sm = sm.OLS(df['Target_Y'], X_sm).fit()

print("--- Simple Linear Regression Model Summary (statsmodels) ---")
print(model_sm.summary())

residuals = model_sm.resid
fitted_values = model_sm.fittedvalues
```

```
--- Simple Linear Regression Model Summary (statsmodels) ---
                            OLS Regression Results                            
==============================================================================
Dep. Variable:               Target_Y   R-squared:                       0.843
Model:                            OLS   Adj. R-squared:                  0.842
Method:                 Least Squares   F-statistic:                     527.6
Date:                Tue, 09 Dec 2025   Prob (F-statistic):           3.10e-41
Time:                        23:12:54   Log-Likelihood:                -200.46
No. Observations:                 100   AIC:                             404.9
Df Residuals:                      98   BIC:                             410.1
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.4302      0.341     15.944      0.000       4.754       6.106
Feature_X      1.4080      0.061     22.970      0.000       1.286       1.530
==============================================================================
Omnibus:                        0.900   Durbin-Watson:                   2.285
Prob(Omnibus):                  0.638   Jarque-Bera (JB):                0.808
Skew:                           0.217   Prob(JB):                        0.668
Kurtosis:                       2.929   Cond. No.                         10.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
```

In [9]:

```
# a) Normality – Q-Q plot
fig_qq = sm.qqplot(residuals, line='45', fit=True)
fig_qq.suptitle("Q-Q Plot for Normality of Residuals", fontsize=14)
plt.show()
```



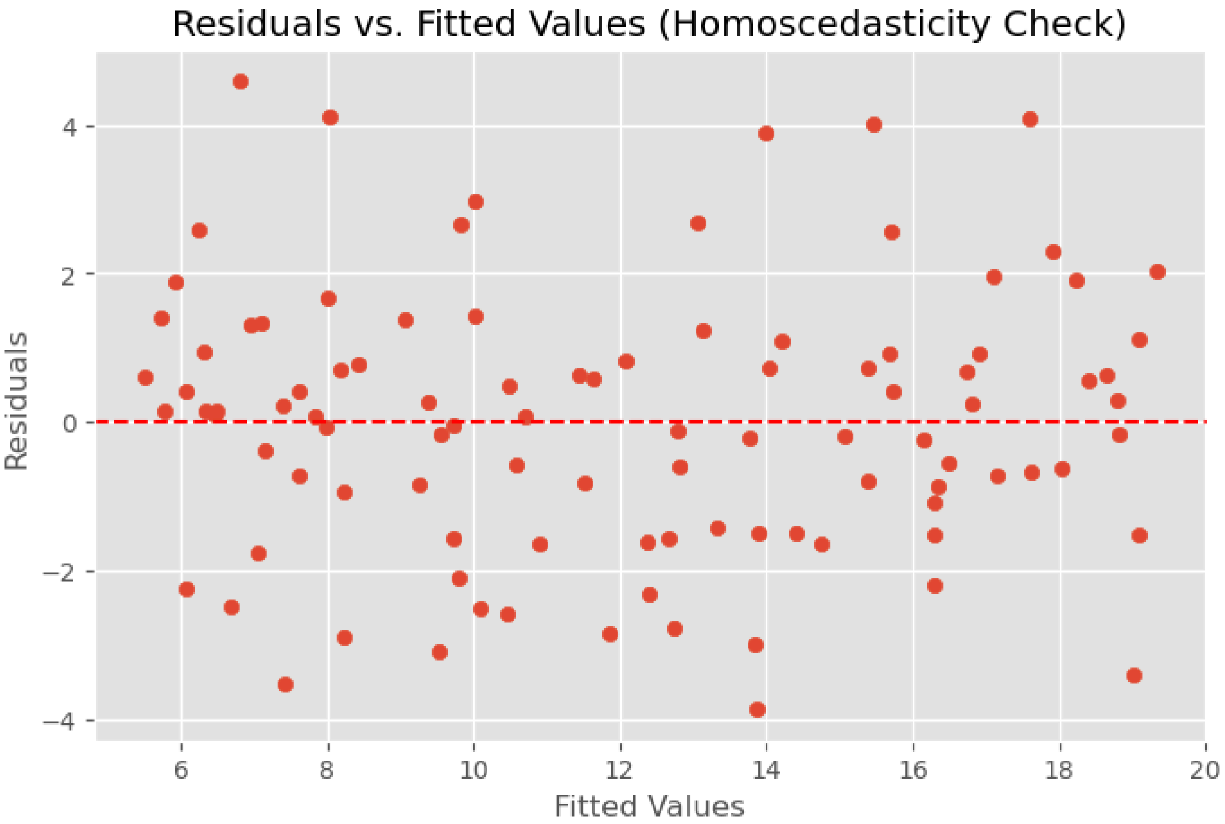
In [11]:

```
# Shapiro-Wilk test
shapiro_test = stats.shapiro(residuals)
print(f"\nShapiro-Wilk Test for Normality (p-value): {shapiro_test.pvalue:.4f}
```

```
Shapiro-Wilk Test for Normality (p-value): 0.2984
```

In [13]:

```
# b) Homoscedasticity — residuals vs fitted
plt.figure(figsize=(8, 5))
plt.scatter(fitted_values, residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title("Residuals vs. Fitted Values (Homoscedasticity Check)")
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.show()
```


Residuals vs. Fitted Values (Homoscedasticity Check)
Residuals
4
2
0
−2
−4
6
8
10
12
14
16
18
20
Fitted Values

In [ ]: