In [ ]:

```
#Import
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.diagnostic import het_breuschpagan
import statsmodels.api as sm
from scipy import stats
```

In [ ]:

```
# Functions to check dataset
def test_def(y,x):

    X_with_constant = sm.add_constant(x)

    # Linearity (Residuals vs. Predicted plot)
    model = sm.OLS(y, X_with_constant)
    results = model.fit()
    residuals = results.resid
    predicted = results.fittedvalues

    plt.scatter(predicted, residuals)
    plt.xlabel("Predicted Values")
    plt.ylabel("Residuals")
    plt.title("Residuals vs. Predicted")
    plt.show()

    # Heteroscedasticity (Breusch-Pagan test)
    _, p_value, _, _ = het_breuschpagan(residuals, X_with_constant)
    print("Heteroscedasticity (Breusch-Pagan test) - p-value:", p_value)
```

In [ ]:

```
#Import Data
## Change excel document and sheet, dependent on what regression to run
data = pd.read_excel('Dataset_GRA19703.xlsx', sheet_name = 'Return t - CEO t - NO')
data = pd.DataFrame(data)

print(data.head(8))
print(data.info())
```

In [ ]:

```
# Control Variable: Time Dummies (Dummy variables for each year)
time_dummies = pd.get_dummies(data['Year'], prefix = 'Year', drop_first=True)
data = pd.concat([data, time_dummies], axis=1)

# Control Variable: Industry Dummies (GICS)
industry_dummies = pd.get_dummies(data["IND"], prefix = 'IND', drop_first=True)
data = pd.concat([data, industry_dummies], axis=1)

print(data.head(8))
```

In [ ]:

```
# Plotting the return values to check for outliers
plt.figure(figsize=(10, 6))  # Set the size of the plot
plt.plot(data['ROA'], marker='o', linestyle='-', color='b')  # Change return measure
plt.xlabel('Company Name')
plt.ylabel('ROA') # Change return measure
plt.title('ROA for each Company') # Change return measure
plt.xticks(range(len(data)), data['Company Name'], rotation=90)
plt.tight_layout()  # Ensure labels are not cut off
plt.show()
```

In [ ]:

```
#Correlation Matrix
data[['ROE','ROA','ROS','Leverage', 'ln_firm_age',
      'ln_firm_size','ln_total_assets','ln_sales',
      'ln_BS', 'ln_VP','ln_total_pay', 'BS_TP_%',
      'VP_TP_%','OB_TP_%']].corr()
```

In [ ]:

```
## Create Q-Q plot

# Observed data
observed_data = data['ln_total_pay'] # Change variable

# Generate a Q-Q plot
stats.probplot(observed_data, plot=plt)

# Add labels and title
plt.xlabel('Expected Normal Value')
plt.ylabel('Observed Value')
plt.title('Normal Q-Q plot for Total Pay') # Change variable name

# Display the plot
plt.show()
```

In [ ]:

```
## OLS Regression

# x- and y-variable
y = data['ROA'] # Change dependent vaiable
x = data[['ln_total_pay','ln_firm_age', 'ln_firm_size', 'Leverage']+
         list(time_dummies.columns) + list(industry_dummies.columns)] # Change independen
x = sm.add_constant(x)

## Test Linearity and Heteroscedasticity
test_def(y,x)

#OLS Regression continued

model_1 = sm.OLS(y,x)
regr_result_model_1 = model_1.fit()

print(regr_result_model_1.summary())
```

In [ ]:

```
## Perform regression with robust standard errors
robust_results = regr_result_model_1.get_robustcov_results(cov_type='HC0')

# Print the results summary
print(robust_results.summary())
```