In [6]:

```
test_data_length = len(test_data)
print("Length of test_data:", test_data_length)
```

```
Length of test_data: 669
```

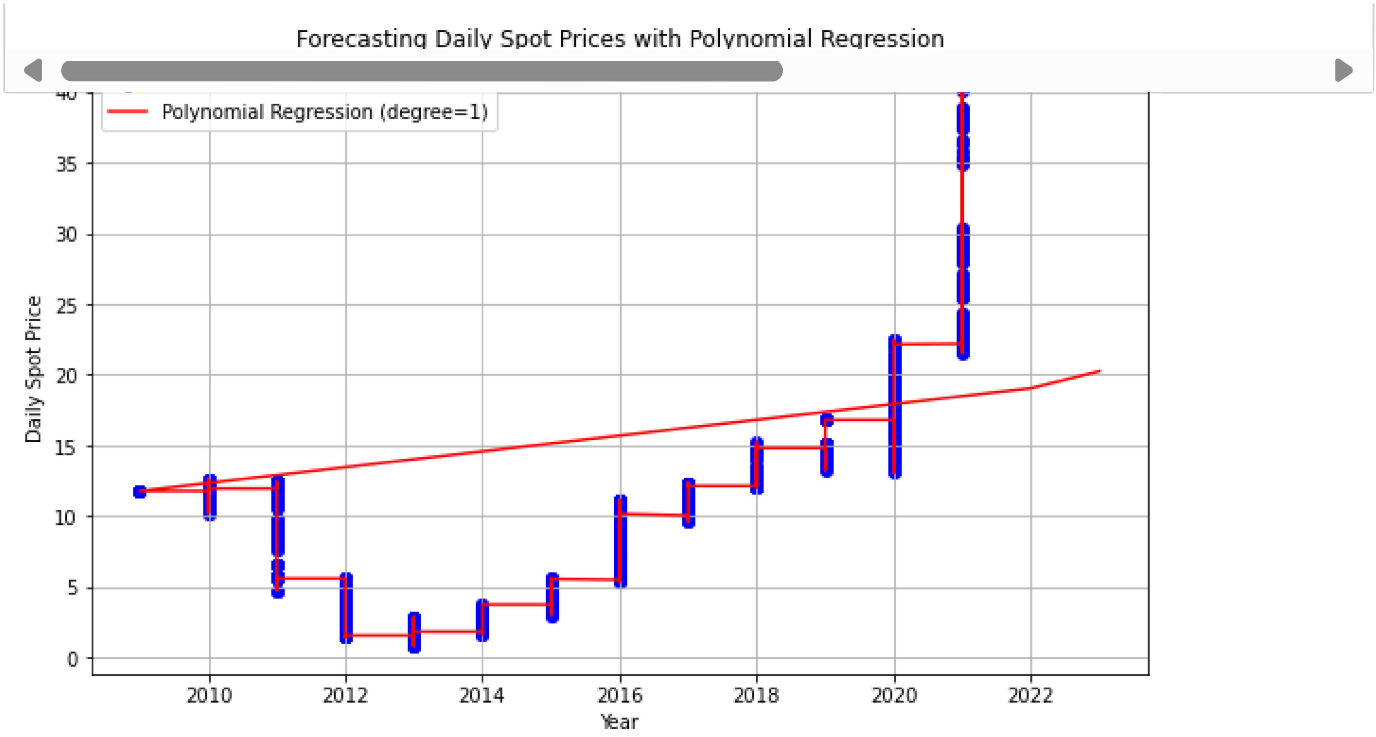
In [21]:

```
import pandas as pd


excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)
print(excel_data.head())
```

```
        Date  SpotPriceUSD
0 2021-12-31       40.2380
1 2021-12-30       40.2085
2 2021-12-29       40.2380
3 2021-12-28       40.3678
4 2021-12-27       40.3678
```

In [28]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# daily spot price data from Excel
excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)

# date and spot price columns
dates = excel_data['Date']
spot_prices = excel_data['SpotPriceUSD']

# Prepare the data for Polynomial Regression
years = dates.dt.year.values
values = spot_prices.values

# Reshaped data for modeling
years = years.reshape(-1, 1)
values = values.reshape(-1, 1)

#  polynomial features
degree = 1  # the degree of the polynomial
poly_features = PolynomialFeatures(degree=degree)
X_poly = poly_features.fit_transform(years)

# Fit for polynomial regression model
model = LinearRegression()
model.fit(X_poly, values)

# Forecast for years 2022 and 2023
forecast_years = np.array([2022, 2023]).reshape(-1, 1)
forecast_poly = poly_features.transform(forecast_years)
forecast_values = model.predict(forecast_poly)


plt.figure(figsize=(10, 6))
plt.scatter(years, values, color='blue', label='Actual Data')
plt.plot(np.concatenate((years, forecast_years)), np.concatenate((values, forecast_value
         color='red', label=f'Polynomial Regression (degree={degree})')
plt.xlabel('Year')
plt.ylabel('Daily Spot Price')
plt.title('Forecasting Daily Spot Prices with Polynomial Regression')
plt.legend()
plt.grid()
plt.show()

yearly_data = pd.DataFrame({'Year': years.flatten(), 'Value': values.flatten()})
forecasted_yearly_data = pd.DataFrame({'Year': forecast_years.flatten(), 'Forecasted Val

print("Yearly Actual and Forecasted Values:")
print(yearly_data)
print(forecasted_yearly_data)
```



Yearly Actual and Forecasted Values:

	Year	Value
0	2021	40.2380
1	2021	40.2085
2	2021	40.2380
3	2021	40.3678
4	2021	40.3678
...	...	...
3338	2009	11.8000
3339	2009	11.8000
3340	2009	11.8000
3341	2009	11.8000
3342	2009	11.8000

[3343 rows x 2 columns]

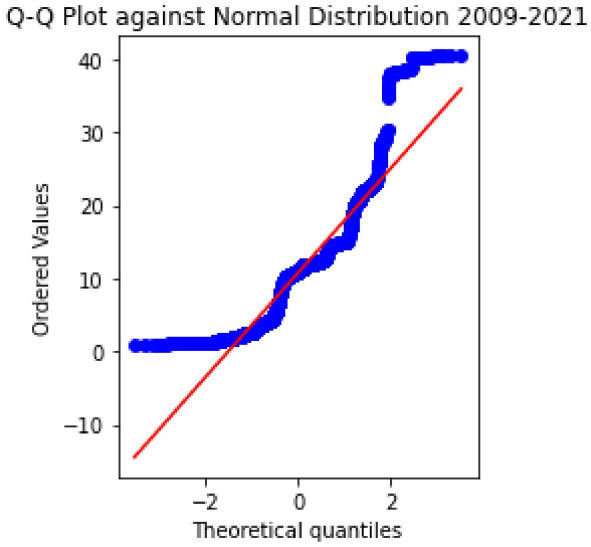
	Year	Forecasted Value
0	2022	19.051646
1	2023	20.257969

In [35]:

```
import pandas as pd
import matplotlib.pyplot as plt


excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)


dates = excel_data['Date']
spot_prices = excel_data['SpotPriceUSD']

# Plotting the actual daily spot prices from 2009 to 2021
plt.figure(figsize=(10, 6))
plt.plot(dates, spot_prices, color='blue', label='Actual Daily Spot Prices')
plt.xlabel('Date')
plt.ylabel('Daily Spot Price')
plt.title('Daily Spot Prices from 2009 to 2021')
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.show()
```

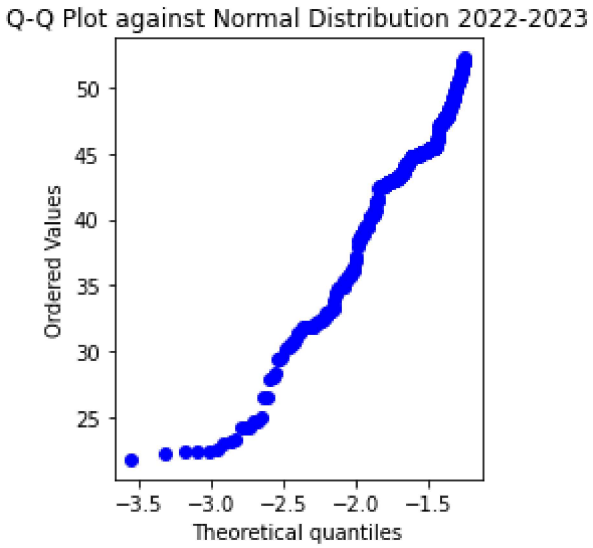
In [81]:

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats

#  daily spot price data from Excel
excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)

#  spot price data
spot_prices = excel_data['SpotPriceUSD']

#  Q-Q plot to compare against a normal distribution
plt.subplot(122)
stats.probplot(spot_prices, dist="norm", plot=plt)
plt.title('Q-Q Plot against Normal Distribution 2009-2021')

plt.tight_layout()
plt.show()
```



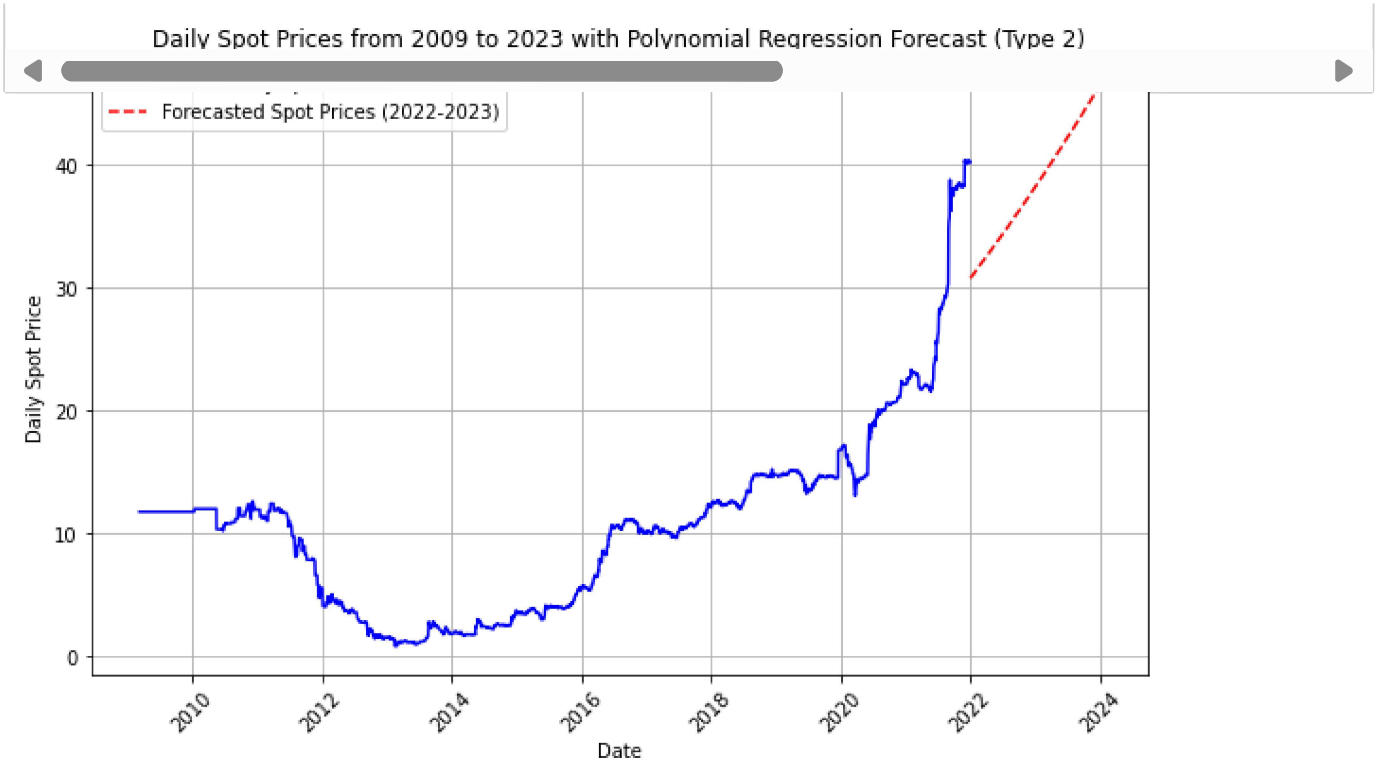
In [90]:

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats

#daily spot price data from Excel
excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)

# spot price data
spot_prices = excel_data['SpotPriceUSD']

# Q-Q plot to compare against a normal distribution
plt.subplot(122)
stats.probplot(spot_prices, dist="norm", plot=plt)
plt.title('Q-Q Plot against Normal Distribution 2022-2023')

plt.tight_layout()
plt.show()
```



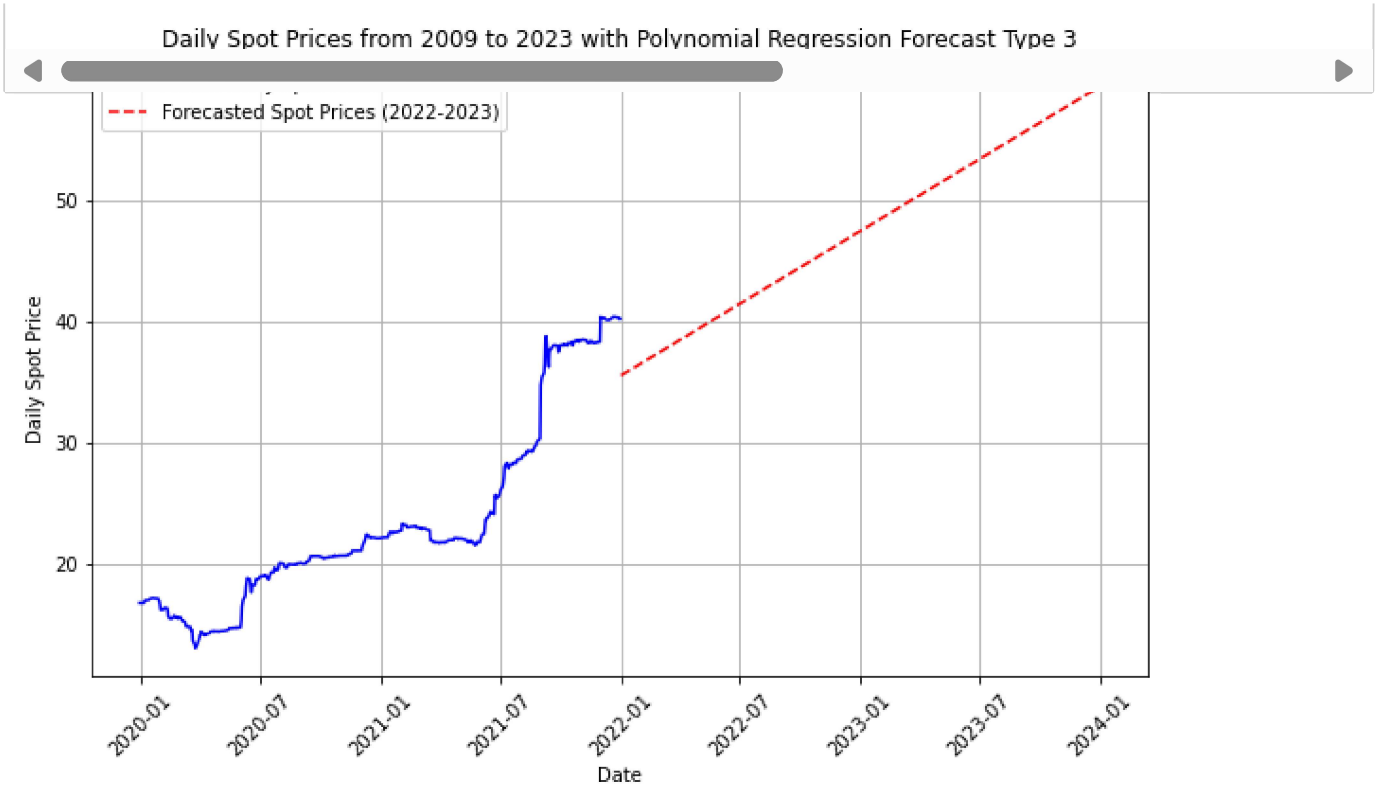
In [93]:

```
#2009-2021 data used

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# daily spot price data from Excel
excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)

# Extract date and spot price columns
dates = excel_data['Date']
spot_prices = excel_data['SpotPriceUSD']

#Polynomial Regression
days = (dates - dates.min()).dt.days.values  # Convert dates to days since the start
values = spot_prices.values
#  data for modeling
days = days.reshape(-1, 1)
values = values.reshape(-1, 1)

#  features
degree = 2  # the degree of the polynomial
poly_features = PolynomialFeatures(degree=degree)
X_poly = poly_features.fit_transform(days)

# Fit model
model = LinearRegression()
model.fit(X_poly, values)

#  features for forecasted days 2022 and 2023
forecast_days = np.arange(days.max() + 1, days.max() + 1 + 730).reshape(-1, 1)
forecast_poly = poly_features.transform(forecast_days)
forecast_values = model.predict(forecast_poly)

plt.figure(figsize=(10, 6))
plt.plot(dates, spot_prices, color='blue', label='Actual Daily Spot Prices')
plt.plot(forecasted_dates, forecast_values, color='red', linestyle='--', label='Forecast
plt.xlabel('Date')
plt.ylabel('Daily Spot Price')
plt.title('Daily Spot Prices from 2009 to 2023 with Polynomial Regression Forecast (Type
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.show()

# Create DataFrame for forecasted values
forecasted_dates = pd.date_range(start='2022-01-01', periods=730, freq='D')
forecasted_values_df = pd.DataFrame({'Date': forecasted_dates, 'Forecasted Spot Price':

# Export the forecasted data to an Excel file
forecasted_values_df.to_excel('Forecasted_Spot_Prices_2022_2023_1.xlsx', index=False)
```


Daily Spot Prices from 2009 to 2023 with Polynomial Regression Forecast (Type 2)
Forecasted Spot Prices (2022-2023)
Daily Spot Price
Date
0
10
20
30
40
2010
2012
2014
2016
2018
2020
2022
2024

In [91]:

```
#2020-2021 data used

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)

#date and spot price columns up to row 524 for 2020-2021
dates = excel_data['Date'][:525]
spot_prices = excel_data['SpotPriceUSD'][:525]

# prep for data for Polynomial Regression
days = (dates - dates.min()).dt.days.values  # Convert dates to days since the start
values = spot_prices.values

# Reshapefor modeling
days = days.reshape(-1, 1)
values = values.reshape(-1, 1)

# polynomial features
degree = 1  # the degree of the polynomial
poly_features = PolynomialFeatures(degree=degree)
X_poly = poly_features.fit_transform(days)

# Fitting for polynomial regression model
model = LinearRegression()
model.fit(X_poly, values)

# Generating polynomial features for forecasted days (2022 and 2023)
forecast_days = np.arange(days.max() + 1, days.max() + 1 + 730).reshape(-1, 1)
forecast_poly = poly_features.transform(forecast_days)
forecast_values = model.predict(forecast_poly)

# DataFrame for forecasted values
forecasted_dates = pd.date_range(start='2022-01-01', periods=730, freq='D')
forecasted_values_df = pd.DataFrame({'Date': forecasted_dates, 'Forecasted Spot Price':


forecasted_values_df.to_excel('Forecasted_Spot_Prices_2022_2023_2.xlsx', index=False)


plt.figure(figsize=(10, 6))
plt.plot(dates, spot_prices, color='blue', label='Actual Daily Spot Prices')
plt.plot(forecasted_dates, forecast_values, color='red', linestyle='--', label='Forecast
plt.xlabel('Date')
plt.ylabel('Daily Spot Price')
plt.title('Daily Spot Prices from 2009 to 2023 with Polynomial Regression Forecast Type
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.show()
```


Daily Spot Prices from 2009 to 2023 with Polynomial Regression Forecast Type 3
Forecasted Spot Prices (2022-2023)
Daily Spot Price
50
40
30
20
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
Date

In [69]:

```
import pandas as pd


excel_file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\
excel_data = pd.read_excel(excel_file_path)
print(excel_data.head())
```

```
        Date  PX_LAST  SpotPriceUSD  CHG_NET_1D     Date.1  \
0 2023-07-24    45.00       26.5500       -2.25 2022-01-01
1 2023-07-21    47.25       27.8775       -0.75 2022-01-02
2 2023-07-20    48.00       28.3200       -1.75 2022-01-03
3 2023-07-19    49.75       29.3525       -0.25 2022-01-04
4 2023-07-18    50.00       29.5000        0.05 2022-01-05

   Forecasted Spot PriceUSD_Type1  Forecasted Spot PriceUSD_Type2  \
0                       18.553219                       30.803059
1                       18.556568                       30.822118
2                       18.559918                       30.841183
3                       18.563268                       30.860255
4                       18.566618                       30.879334

   Forecasted Spot PriceUSD_Type3  Unnamed: 8  Unnamed: 9 Unnamed: 10  \
0                       35.563328         NaN         NaN         Max
1                       35.596016         NaN         NaN         Min
2                       35.628705         NaN         NaN      Median
3                       35.661394         NaN         NaN     Average
4                       35.694083         NaN         NaN         NaN

   Unnamed: 11  Unnamed: 12 Unnamed: 13  Unnamed: 14
0    88.500000          NaN         USD         0.59
1     1.450000          NaN         NaN          NaN
2    20.000000          NaN         NaN          NaN
3    23.923754          NaN         NaN          NaN
4          NaN          NaN         NaN          NaN
```

In [96]:

```
import pandas as pd
import matplotlib.pyplot as plt

file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\\Alter
df = pd.read_excel(file_path)

#  columns calculate correlations for
ref_column = 'SpotPriceUSD_QQ'
columns_to_compare = [
    'Forecasted Spot PriceUSD_Type1',
    'Forecasted Spot PriceUSD_Type2',
    'Forecasted Spot PriceUSD_Type3'
]

# Pearson correlation coefficients
correlation_coeffs = {}
for column in columns_to_compare:
    correlation_coeff = df[ref_column].corr(df[column])
    correlation_coeffs[column] = correlation_coeff

# Printing the correlation coefficients
for column, coeff in correlation_coeffs.items():
    print(f"Pearson Correlation Coefficient between {ref_column} and {column}: {coeff}")
```

```
Pearson Correlation Coefficient between SpotPriceUSD_QQ and Forecasted Spo
t PriceUSD_Type1: -0.6523550397202749
Pearson Correlation Coefficient between SpotPriceUSD_QQ and Forecasted Spo
t PriceUSD_Type2: -0.6632600716891396
Pearson Correlation Coefficient between SpotPriceUSD_QQ and Forecasted Spo
t PriceUSD_Type3: -0.6523550397202749
```

In [101]:

```
import pandas as pd
from sklearn.metrics import mean_absolute_error


file_path = "C:\\Users\\Rishab\\Documents\\UCL\\Masters Thesis\\Relevant journals\\Alter
df = pd.read_excel(file_path)


ref_column = 'SpotPriceUSD_QQ'
columns_to_compare = [
    'Forecasted Spot PriceUSD_Type1',
    'Forecasted Spot PriceUSD_Type2',
    'Forecasted Spot PriceUSD_Type3'
]

# Remove rows with missing values
df.dropna(subset=[ref_column] + columns_to_compare, inplace=True)

# Calculating the MAE values
mae_values = {}
for column in columns_to_compare:
    mae = mean_absolute_error(df[ref_column], df[column])
    mae_values[column] = mae


for column, mae in mae_values.items():
    print(f"Mean Absolute Error between {ref_column} and {column}: {mae}")
```

```
Mean Absolute Error between SpotPriceUSD_QQ and Forecasted Spot PriceUSD_T
ype1: 23.26410425872816
Mean Absolute Error between SpotPriceUSD_QQ and Forecasted Spot PriceUSD_T
ype2: 10.623623137652256
Mean Absolute Error between SpotPriceUSD_QQ and Forecasted Spot PriceUSD_T
ype3: 8.289042202009952
```

In [113]:

```
import pandas as pd
import numpy as np


data = {
    'Risk adjusted revenue forecast_ Expected Value optimization': [6437.92, 184751.08,
    'Risk adjusted revenue forecast_Weighted Sum MOO (75-25 Mangrove Seagrass Ratio)': [
    'Risk adjusted revenue forecast_Weighted Sum MOO (50-50 Mangrove Seagrass Ratio)': [
    'Risk adjusted revenue forecast_Weighted Sum MOO (25-75 Mangrove Seagrass Ratio)': [
}

df = pd.DataFrame(data)

#reference column
reference_column = df['Risk adjusted revenue forecast_ Expected Value optimization']

# increased revenue columns
increased_columns = [
    df['Risk adjusted revenue forecast_Weighted Sum MOO (75-25 Mangrove Seagrass Ratio)'
    df['Risk adjusted revenue forecast_Weighted Sum MOO (50-50 Mangrove Seagrass Ratio)'
    df['Risk adjusted revenue forecast_Weighted Sum MOO (25-75 Mangrove Seagrass Ratio)'
]

# Calculating annual returns for each column
returns = []
for increased_column in increased_columns:
    returns.append((increased_column - reference_column) / reference_column)

# Confidence level and time horizon
confidence_level = 0.95
time_horizon = 1
# VaR and ES for each increased revenue column
var_results = []
es_results = []

for return_series in returns:
    # Calculating VaR
    var = np.percentile(return_series, 100 * (1 - confidence_level))
    var_results.append(var)

    # Calculating ES
    losses = [r for r in return_series if r < var]
    es = -np.mean(losses)
    es_results.append(es)



for i, (var, es) in enumerate(zip(var_results, es_results)):
    print(f"{df.columns[i + 1]}:")
    print(f"VaR ({confidence_level * 100}% confidence, {time_horizon} year horizon): {va
    print(f"ES ({confidence_level * 100}% confidence, {time_horizon} year horizon): {es:
    print()
```

Risk adjusted revenue forecast_Weighted Sum MOO (75-25 Mangrove Seagrass R
atio):
VaR (95.0% confidence, 1 year horizon): 0.0898
ES (95.0% confidence, 1 year horizon): -0.0614

Risk adjusted revenue forecast_Weighted Sum MOO (50-50 Mangrove Seagrass R
atio):
VaR (95.0% confidence, 1 year horizon): 0.0834
ES (95.0% confidence, 1 year horizon): -0.0584

Risk adjusted revenue forecast_Weighted Sum MOO (25-75 Mangrove Seagrass R
atio):
VaR (95.0% confidence, 1 year horizon): 0.0770
ES (95.0% confidence, 1 year horizon): -0.0555

In [114]:

```
import pandas as pd
import numpy as np

data = {
    'BCS revenue forecast per annum': [30000.00, 450000.00, 1200000.00, 1200000.00, 1200
    'Risk adjusted revenue forecast_ Expected Value optimization': [6437.92, 184751.08,
    'Risk adjusted revenue forecast_Weighted Sum MOO (75-25 Mangrove Seagrass Ratio)': [
    'Risk adjusted revenue forecast_Weighted Sum MOO (50-50 Mangrove Seagrass Ratio)': [
    'Risk adjusted revenue forecast_Weighted Sum MOO (25-75 Mangrove Seagrass Ratio)': [
}

df = pd.DataFrame(data)

# Reference column
reference_column = df['BCS revenue forecast per annum']

#  revenue chnage columns (including the previous reference column)
decreased_columns = [
    df['Risk adjusted revenue forecast_ Expected Value optimization'],
    df['Risk adjusted revenue forecast_Weighted Sum MOO (75-25 Mangrove Seagrass Ratio)'
    df['Risk adjusted revenue forecast_Weighted Sum MOO (50-50 Mangrove Seagrass Ratio)'
    df['Risk adjusted revenue forecast_Weighted Sum MOO (25-75 Mangrove Seagrass Ratio)'
]

# Calculated Mean Absolute Error (MAE) for each column
mae_results = []

for decreased_column in decreased_columns:
    mae = np.mean(np.abs(decreased_column - reference_column))
    mae_results.append(mae)

# Printed MAE for each decreased revenue column
for i, mae in enumerate(mae_results):
    print(f"{df.columns[i + 1]}:")
    print(f"MAE: {mae:.4f}")
    print()
```

```
Risk adjusted revenue forecast_ Expected Value optimization:
MAE: 462110.8871

Risk adjusted revenue forecast_Weighted Sum MOO (75-25 Mangrove Seagrass R
atio):
MAE: 273068.7124

Risk adjusted revenue forecast_Weighted Sum MOO (50-50 Mangrove Seagrass R
atio):
MAE: 276770.0080

Risk adjusted revenue forecast_Weighted Sum MOO (25-75 Mangrove Seagrass R
atio):
MAE: 280471.3036
```

In [ ]:

```
```