step : Imports & Reading Data

In [1]:
```
pip install pandas sqlite3 matplotlib seaborn
```

```
Requirement already satisfied: pandas in c:₩users₩seoin₩anaconda₩lib₩site-packages
(2.2.2)
Note: you may need to restart the kernel to use updated packages.
```

```
ERROR: Could not find a version that satisfies the requirement sqlite3 (from version
s: none)
ERROR: No matching distribution found for sqlite3
```

In [2]:
```
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats
pd.set_option('display.max_columns', None)
```

```
C:₩Users₩seoin₩anaconda₩Lib₩site-packages₩pandas₩core₩arrays₩masked.py:60: UserWarni
ng: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' curren
tly installed).
  from pandas.core import (
```

In [3]:
```
# Connect to SQLite database
conn = sqlite3.connect('cruise_data.db')
cursor = conn.cursor()

# Load CSV data into a DataFrame
df = pd.read_csv('C:/Users/seoin/Desktop/Tui/task_data/data.csv')

# Create a table in the SQLite database
df.to_sql('cruise_data', conn, if_exists='replace', index=False)

# Verify the data is loaded
query = "SELECT * FROM cruise_data"
df_sql = pd.read_sql_query(query, conn)
print(df_sql.head())
```

	Start Time	End Time	Vessel Name	Power Galley 1 (MW)	₩
0	2023-01-01T00:00:00	2023-01-01T00:05:00	Vessel 1	0.0946	
1	2023-01-01T00:05:00	2023-01-01T00:10:00	Vessel 1	0.0540	
2	2023-01-01T00:10:00	2023-01-01T00:15:00	Vessel 1	0.0439	
3	2023-01-01T00:15:00	2023-01-01T00:20:00	Vessel 1	0.0733	
4	2023-01-01T00:20:00	2023-01-01T00:25:00	Vessel 1	0.0780	

	Power Galley 2 (MW)	Power Service (MW)	HVAC Chiller 1 Power (MW)	₩
0	0.1384	5.4654	0.5074	
1	0.1370	5.4387	0.5158	
2	0.1785	5.5265	0.5117	
3	0.1725	5.5257	0.5177	
4	0.1397	5.4634	0.5169	

	HVAC Chiller 2 Power (MW)	HVAC Chiller 3 Power (MW)	Scrubber Power (MW)	₩
0	0.0	0.4979	0.4191	
1	0.0	0.4982	0.4204	
2	0.0	0.5032	0.4199	
3	0.0	0.5103	0.4188	
4	0.0	0.5100	0.4203	

	Sea Temperature (Celsius)	Boiler 1 Fuel Flow Rate (L/h)	₩
0	27.3000	0.0000	
1	27.3000	47.7695	
2	27.3000	77.2034	
3	27.3076	60.6369	
4	27.3518	55.2184	

	Boiler 2 Fuel Flow Rate (L/h)	Incinerator 1 Fuel Flow Rate (L/h)	₩
0	0.0	19.0090	
1	0.0	216.3180	
2	0.0	439.4300	
3	0.0	218.2797	
4	0.0	0.0000	

	Diesel Generator 1 Power (MW)	Diesel Generator 2 Power (MW)	₩
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	Diesel Generator 3 Power (MW)	Diesel Generator 4 Power (MW)	₩
0	0.0	7.3349	
1	0.0	7.3011	
2	0.0	7.3299	
3	0.0	7.3712	
4	0.0	7.3032	

	Latitude (Degrees)	Longitude (Degrees)	Relative Wind Angle (Degrees)	₩
0	17.72523	-65.45738	8.4428	
1	17.73088	-65.44803	41.3100	
2	17.73655	-65.43887	23.9997	
3	17.74202	-65.42980	14.5540	
4	17.74713	-65.42042	14.5632	

	True Wind Angle (Degrees)	Depth (m)	Relative Wind Direction (Degrees)	₩
0	10.9049	NaN	64.3112	
1	78.7817	NaN	62.8161	
2	33.6216	NaN	80.7356	
3	20.0348	NaN	75.9723	
4	20.0328	NaN	74.6509	

	True Wind Direction (Degrees)	Draft (m)	Speed Over Ground (knots)	₩

0	66.7735	7.8721	7.6300
1	64.3452	7.8713	7.5800
2	90.3574	7.8718	7.4379
3	81.4529	7.8710	7.3979
4	80.1204	7.8707	7.4343

	True Wind Speed (knots)	Relative Wind Speed (knots)	₩
0	19.5050	27.0579	
1	19.2968	26.8067	
2	19.4491	25.8380	
3	20.6231	27.6498	
4	20.4554	27.5341	

	Speed Through Water (knots)	Local Time (h)	Trim (m)	₩
0	7.8881	19.67367	-0.1425	
1	7.7438	19.75763	-0.1405	
2	7.6320	19.84158	-0.1450	
3	7.5080	19.92551	-0.1308	
4	7.5521	20.00947	-0.1269	

	Propulsion Power (MW)	Port Side Propulsion Power (MW)	₩
0	1.8691	0.8854	
1	1.8622	0.8737	
2	1.8036	0.8441	
3	1.8457	0.8543	
4	1.8399	0.8467	

	Starboard Side Propulsion Power (MW)	Bow Thruster 1 Power (MW)	₩
0	0.9837	0.0	
1	0.9885	0.0	
2	0.9595	0.0	
3	0.9914	0.0	
4	0.9932	0.0	

	Bow Thruster 2 Power (MW)	Bow Thruster 3 Power (MW)	₩
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	Stern Thruster 1 Power (MW)	Stern Thruster 2 Power (MW)	₩
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	Main Engine 1 Fuel Flow Rate (kg/h)	Main Engine 2 Fuel Flow Rate (kg/h)	₩
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

	Main Engine 3 Fuel Flow Rate (kg/h)	Main Engine 4 Fuel Flow Rate (kg/h)
0	0.0	1645.82000
1	0.0	1643.78999
2	0.0	1642.07000
3	0.0	1650.71000
4	0.0	1644.54000

Step : Data Understanding

In [4]:
```
df.shape
```

Out[4]:
```
(210240, 44)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210240 entries, 0 to 210239
Data columns (total 44 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   Start Time                            210240 non-null  object 
 1   End Time                              210240 non-null  object 
 2   Vessel Name                           210240 non-null  object 
 3   Power Galley 1 (MW)                   210224 non-null  float64
 4   Power Galley 2 (MW)                   210224 non-null  float64
 5   Power Service (MW)                    210222 non-null  float64
 6   HVAC Chiller 1 Power (MW)             210033 non-null  float64
 7   HVAC Chiller 2 Power (MW)             210033 non-null  float64
 8   HVAC Chiller 3 Power (MW)             210033 non-null  float64
 9   Scrubber Power (MW)                   210224 non-null  float64
 10  Sea Temperature (Celsius)             210224 non-null  float64
 11  Boiler 1 Fuel Flow Rate (L/h)         210224 non-null  float64
 12  Boiler 2 Fuel Flow Rate (L/h)         210224 non-null  float64
 13  Incinerator 1 Fuel Flow Rate (L/h)    210224 non-null  float64
 14  Diesel Generator 1 Power (MW)         210224 non-null  float64
 15  Diesel Generator 2 Power (MW)         210224 non-null  float64
 16  Diesel Generator 3 Power (MW)         210224 non-null  float64
 17  Diesel Generator 4 Power (MW)         210224 non-null  float64
 18  Latitude (Degrees)                    209900 non-null  float64
 19  Longitude (Degrees)                   209900 non-null  float64
 20  Relative Wind Angle (Degrees)         210226 non-null  float64
 21  True Wind Angle (Degrees)             210166 non-null  float64
 22  Depth (m)                             152746 non-null  float64
 23  Relative Wind Direction (Degrees)     210185 non-null  float64
 24  True Wind Direction (Degrees)         210166 non-null  float64
 25  Draft (m)                             209097 non-null  float64
 26  Speed Over Ground (knots)             209340 non-null  float64
 27  True Wind Speed (knots)               210166 non-null  float64
 28  Relative Wind Speed (knots)           210226 non-null  float64
 29  Speed Through Water (knots)           209299 non-null  float64
 30  Local Time (h)                        209900 non-null  float64
 31  Trim (m)                              209161 non-null  float64
 32  Propulsion Power (MW)                 210224 non-null  float64
 33  Port Side Propulsion Power (MW)       210224 non-null  float64
 34  Starboard Side Propulsion Power (MW)  210224 non-null  float64
 35  Bow Thruster 1 Power (MW)             210224 non-null  float64
 36  Bow Thruster 2 Power (MW)             210224 non-null  float64
 37  Bow Thruster 3 Power (MW)             210224 non-null  float64
 38  Stern Thruster 1 Power (MW)           210224 non-null  float64
 39  Stern Thruster 2 Power (MW)           210224 non-null  float64
 40  Main Engine 1 Fuel Flow Rate (kg/h)   210224 non-null  float64
 41  Main Engine 2 Fuel Flow Rate (kg/h)   210224 non-null  float64
 42  Main Engine 3 Fuel Flow Rate (kg/h)   210224 non-null  float64
 43  Main Engine 4 Fuel Flow Rate (kg/h)   210224 non-null  float64
dtypes: float64(41), object(3)
memory usage: 70.6+ MB
```

In [6]:
```
df.head()
```

Out[6]:

	Start Time	End Time	Vessel Name	Power Galley 1 (MW)	Power Galley 2 (MW)	Power Service (MW)	HVAC Chiller 1 Power (MW)	HVAC Chiller 2 Power (MW)	HVAC Chiller 3 Power (MW)	Scrubber Power (MW)	Tem
0	2023-01-01T00:00:00	2023-01-01T00:05:00	Vessel 1	0.0946	0.1384	5.4654	0.5074	0.0	0.4979	0.4191	
1	2023-01-01T00:05:00	2023-01-01T00:10:00	Vessel 1	0.0540	0.1370	5.4387	0.5158	0.0	0.4982	0.4204	
2	2023-01-01T00:10:00	2023-01-01T00:15:00	Vessel 1	0.0439	0.1785	5.5265	0.5117	0.0	0.5032	0.4199	
3	2023-01-01T00:15:00	2023-01-01T00:20:00	Vessel 1	0.0733	0.1725	5.5257	0.5177	0.0	0.5103	0.4188	
4	2023-01-01T00:20:00	2023-01-01T00:25:00	Vessel 1	0.0780	0.1397	5.4634	0.5169	0.0	0.5100	0.4203	

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Start Time', 'End Time', 'Vessel Name', 'Power Galley 1 (MW)',
       'Power Galley 2 (MW)', 'Power Service (MW)',
       'HVAC Chiller 1 Power (MW)', 'HVAC Chiller 2 Power (MW)',
       'HVAC Chiller 3 Power (MW)', 'Scrubber Power (MW)',
       'Sea Temperature (Celsius)', 'Boiler 1 Fuel Flow Rate (L/h)',
       'Boiler 2 Fuel Flow Rate (L/h)', 'Incinerator 1 Fuel Flow Rate (L/h)',
       'Diesel Generator 1 Power (MW)', 'Diesel Generator 2 Power (MW)',
       'Diesel Generator 3 Power (MW)', 'Diesel Generator 4 Power (MW)',
       'Latitude (Degrees)', 'Longitude (Degrees)',
       'Relative Wind Angle (Degrees)', 'True Wind Angle (Degrees)',
       'Depth (m)', 'Relative Wind Direction (Degrees)',
       'True Wind Direction (Degrees)', 'Draft (m)',
       'Speed Over Ground (knots)', 'True Wind Speed (knots)',
       'Relative Wind Speed (knots)', 'Speed Through Water (knots)',
       'Local Time (h)', 'Trim (m)', 'Propulsion Power (MW)',
       'Port Side Propulsion Power (MW)',
       'Starboard Side Propulsion Power (MW)', 'Bow Thruster 1 Power (MW)',
       'Bow Thruster 2 Power (MW)', 'Bow Thruster 3 Power (MW)',
       'Stern Thruster 1 Power (MW)', 'Stern Thruster 2 Power (MW)',
       'Main Engine 1 Fuel Flow Rate (kg/h)',
       'Main Engine 2 Fuel Flow Rate (kg/h)',
       'Main Engine 3 Fuel Flow Rate (kg/h)',
       'Main Engine 4 Fuel Flow Rate (kg/h)'],
      dtype='object')
```

Step : Data Preperation

In [8]:
```
# Convert time columns to datetime dtype
df['Start Time'] = pd.to_datetime(df['Start Time'])
df['End Time'] = pd.to_datetime(df['End Time'])

# Display info for only 'Start Time' and 'End Time' columns
df[['Start Time', 'End Time']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210240 entries, 0 to 210239
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   Start Time  210240 non-null  datetime64[ns]
 1   End Time    210240 non-null  datetime64[ns]
dtypes: datetime64[ns](2)
memory usage: 3.2 MB
```

Missing value check

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Start Time                                0
End Time                                  0
Vessel Name                               0
Power Galley 1 (MW)                      16
Power Galley 2 (MW)                      16
Power Service (MW)                       18
HVAC Chiller 1 Power (MW)               207
HVAC Chiller 2 Power (MW)               207
HVAC Chiller 3 Power (MW)               207
Scrubber Power (MW)                      16
Sea Temperature (Celsius)                16
Boiler 1 Fuel Flow Rate (L/h)            16
Boiler 2 Fuel Flow Rate (L/h)            16
Incinerator 1 Fuel Flow Rate (L/h)       16
Diesel Generator 1 Power (MW)            16
Diesel Generator 2 Power (MW)            16
Diesel Generator 3 Power (MW)            16
Diesel Generator 4 Power (MW)            16
Latitude (Degrees)                      340
Longitude (Degrees)                     340
Relative Wind Angle (Degrees)            14
True Wind Angle (Degrees)                74
Depth (m)                             57494
Relative Wind Direction (Degrees)        55
True Wind Direction (Degrees)            74
Draft (m)                              1143
Speed Over Ground (knots)               900
True Wind Speed (knots)                  74
Relative Wind Speed (knots)              14
Speed Through Water (knots)             941
Local Time (h)                          340
Trim (m)                               1079
Propulsion Power (MW)                    16
Port Side Propulsion Power (MW)          16
Starboard Side Propulsion Power (MW)     16
Bow Thruster 1 Power (MW)                16
Bow Thruster 2 Power (MW)                16
Bow Thruster 3 Power (MW)                16
Stern Thruster 1 Power (MW)              16
Stern Thruster 2 Power (MW)              16
Main Engine 1 Fuel Flow Rate (kg/h)      16
Main Engine 2 Fuel Flow Rate (kg/h)      16
Main Engine 3 Fuel Flow Rate (kg/h)      16
Main Engine 4 Fuel Flow Rate (kg/h)      16
dtype: int64
```

In [10]:
```
df.duplicated().sum()
```

Out[10]: 0

In [11]:
```
df[df['Power Galley 1 (MW)'].isna()]
```

Out[11]:

	Start Time	End Time	Vessel Name	Power Galley 1 (MW)	Power Galley 2 (MW)	Power Service (MW)	HVAC Chiller 1 Power (MW)	HVAC Chiller 2 Power (MW)	HVAC Chiller 3 Power (MW)	Scrubber Power (MW)	Temp
88578	2023-11-04 13:30:00	2023-11-04 13:35:00	Vessel 1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
140167	2023-05-02 16:35:00	2023-05-02 16:40:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
145233	2023-05-20 06:45:00	2023-05-20 06:50:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
161071	2023-07-14 06:35:00	2023-07-14 06:40:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
163498	2023-07-22 16:50:00	2023-07-22 16:55:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
163499	2023-07-22 16:55:00	2023-07-22 17:00:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
163500	2023-07-22 17:00:00	2023-07-22 17:05:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
165191	2023-07-28 13:55:00	2023-07-28 14:00:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
165767	2023-07-30 13:55:00	2023-07-30 14:00:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
165768	2023-07-30 14:00:00	2023-07-30 14:05:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
168077	2023-08-07 14:25:00	2023-08-07 14:30:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
168078	2023-08-07 14:30:00	2023-08-07 14:35:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
179568	2023-09-16 12:00:00	2023-09-16 12:05:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
179569	2023-09-16 12:05:00	2023-09-16 12:10:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
179572	2023-09-16 12:20:00	2023-09-16 12:25:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

	Start Time	End Time	Vessel Name	Power Galley 1 (MW)	Power Galley 2 (MW)	Power Service (MW)	HVAC Chiller 1 Power (MW)	HVAC Chiller 2 Power (MW)	HVAC Chiller 3 Power (MW)	Scrubber Power (MW)	Temp
209699	2023-12-30 02:55:00	2023-12-30 03:00:00	Vessel 2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [12]:
```
# Drop rows where 'Power Galley 1 (MW)' has missing values becasue most of the data
df = df.dropna(subset=['Power Galley 1 (MW)'])
```

In [13]:
```
df[df['HVAC Chiller 1 Power (MW)'].isna()]
```

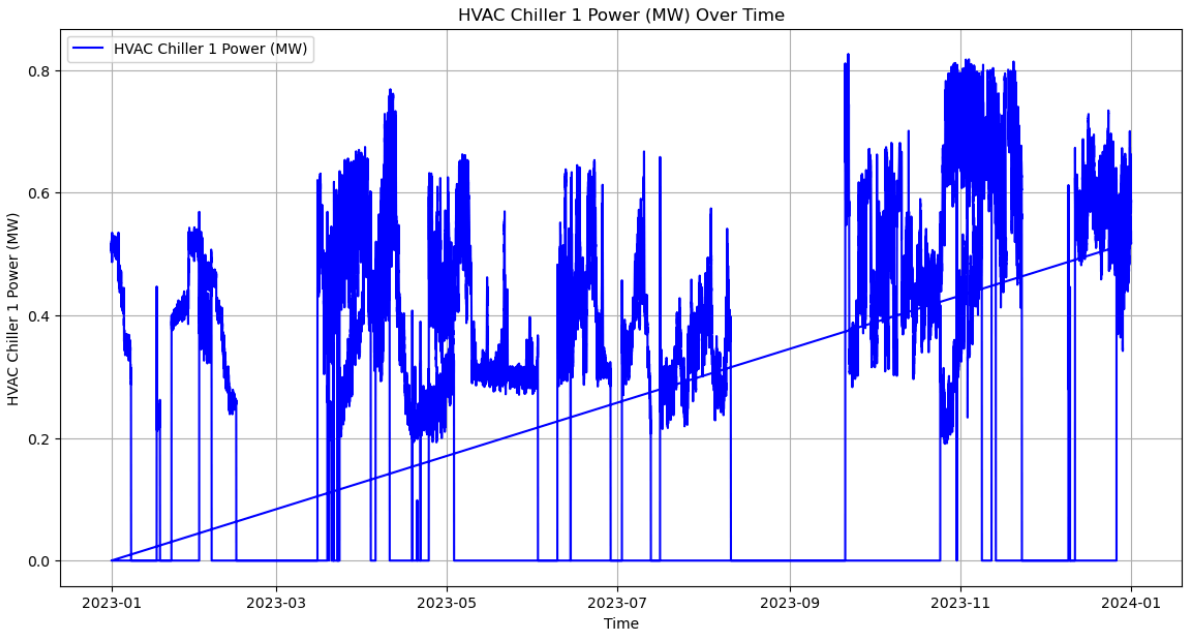
Out[13]:

	Start Time	End Time	Vessel Name	Power Galley 1 (MW)	Power Galley 2 (MW)	Power Service (MW)	HVAC Chiller 1 Power (MW)	HVAC Chiller 2 Power (MW)	HVAC Chiller 3 Power (MW)	Scrubber Power (MW)	Temp
197377	2023-11-17 08:05:00	2023-11-17 08:10:00	Vessel 2	0.0000	0.0918	4.4755	NaN	NaN	NaN	0.1543	
197378	2023-11-17 08:10:00	2023-11-17 08:15:00	Vessel 2	0.0000	0.0716	4.5178	NaN	NaN	NaN	0.1534	
197379	2023-11-17 08:15:00	2023-11-17 08:20:00	Vessel 2	0.0000	0.0638	4.4713	NaN	NaN	NaN	0.1530	
197380	2023-11-17 08:20:00	2023-11-17 08:25:00	Vessel 2	0.0071	0.0620	4.5521	NaN	NaN	NaN	0.1545	
197381	2023-11-17 08:25:00	2023-11-17 08:30:00	Vessel 2	0.0012	0.0420	4.4380	NaN	NaN	NaN	0.1539	
...	...	...	...	...	...	...	...	...	...	...	
197563	2023-11-17 23:35:00	2023-11-17 23:40:00	Vessel 2	0.0218	0.1772	5.5094	NaN	NaN	NaN	0.1577	
197564	2023-11-17 23:40:00	2023-11-17 23:45:00	Vessel 2	0.0487	0.1671	5.5004	NaN	NaN	NaN	0.1569	
197565	2023-11-17 23:45:00	2023-11-17 23:50:00	Vessel 2	0.0308	0.1678	5.5444	NaN	NaN	NaN	0.1581	
197566	2023-11-17 23:50:00	2023-11-17 23:55:00	Vessel 2	0.0497	0.1516	5.5213	NaN	NaN	NaN	0.1584	
197567	2023-11-17 23:55:00	2023-11-18 00:00:00	Vessel 2	0.0697	0.2012	5.6888	NaN	NaN	NaN	0.1585	

191 rows × 44 columns

In [14]:

```
# Ensure 'Start Time' is in datetime format
df['Start Time'] = pd.to_datetime(df['Start Time'])

# Plot the time series for 'HVAC Chiller 1 Power (MW)' to see how to handle missing v
plt.figure(figsize=(14, 7))
plt.plot(df['Start Time'], df['HVAC Chiller 1 Power (MW)'], label='HVAC Chiller 1 Po
plt.xlabel('Time')
plt.ylabel('HVAC Chiller 1 Power (MW)')
plt.title('HVAC Chiller 1 Power (MW) Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



In [15]:
```
# Sort the dataframe by 'Start Time' to ensure proper filling
df = df.sort_values(by='Start Time')

# Set 'Start Time' as the index
df.set_index('Start Time', inplace=True)

# Interpolate to fill missing values for chiller columns
#interpolation is a process of determining the unknown values that lie in between the
df['HVAC Chiller 1 Power (MW)'] = df['HVAC Chiller 1 Power (MW)'].interpolate()
df['HVAC Chiller 2 Power (MW)'] = df['HVAC Chiller 2 Power (MW)'].interpolate()
df['HVAC Chiller 3 Power (MW)'] = df['HVAC Chiller 3 Power (MW)'].interpolate()

# If you need to reset the index back to columns
df.reset_index(inplace=True)
```

In [16]:
```
df[df['Power Service (MW)'].isna()]
```

Out[16]:

	Start Time	End Time	Vessel Name	Power Galley 1 (MW)	Power Galley 2 (MW)	Power Service (MW)	HVAC Chiller 1 Power (MW)	HVAC Chiller 2 Power (MW)	HVAC Chiller 3 Power (MW)	Scrubber Power (MW)	Temp
148890	2023-09-16 12:15:00	2023-09-16 12:20:00	Vessel 2	0.03	0.080	NaN	0.000	0.384	0.0	0.773	
209145	2023-12-30 03:00:00	2023-12-30 03:05:00	Vessel 2	0.03	0.087	NaN	0.508	0.528	0.0	0.346	

In [17]:
```
# First forward fill, then backward fill
# the next available value after the missing data point replaces the missing value be
df['Power Service (MW)'] = df['Power Service (MW)'].ffill().bfill()
```

In [18]:
```
df[df['Speed Over Ground (knots)'].isna()]
```

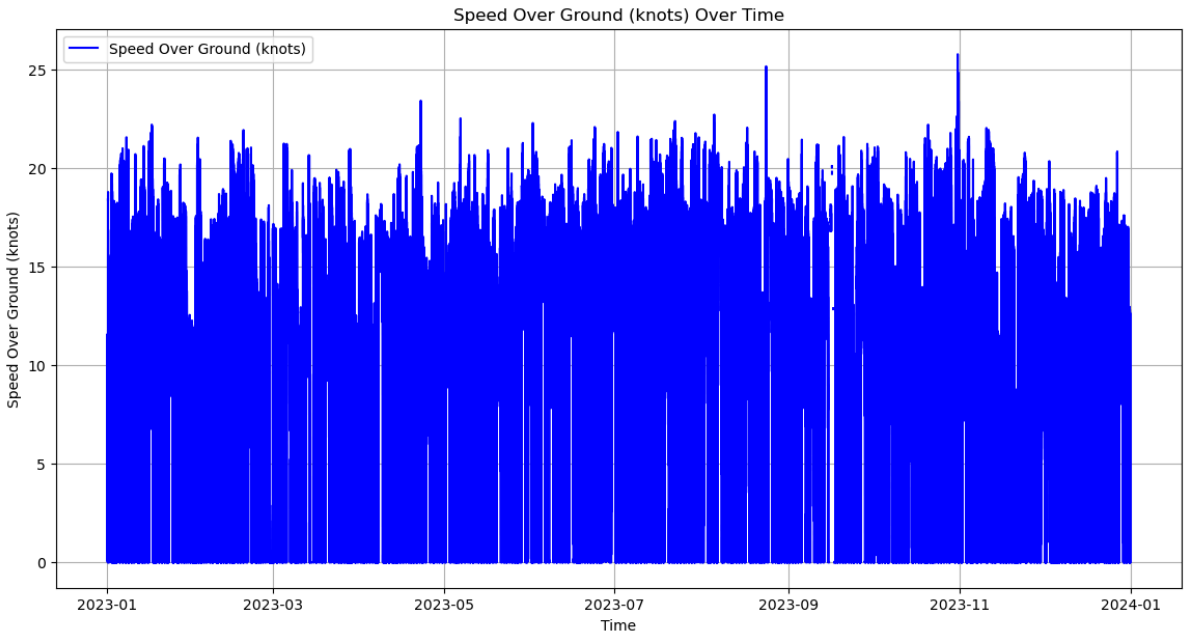
Out[18]:

	Start Time	End Time	Vessel Name	Power Galley 1 (MW)	Power Galley 2 (MW)	Power Service (MW)	HVAC Chiller 1 Power (MW)	HVAC Chiller 2 Power (MW)	HVAC Chiller 3 Power (MW)	Scrubber Power (MW)	Temp
137881	2023-08-28 09:30:00	2023-08-28 09:35:00	Vessel 2	0.1025	0.1235	-0.0400	0.0000	0.0	0.4086	0.0000	
137884	2023-08-28 09:35:00	2023-08-28 09:40:00	Vessel 2	0.1113	0.1194	-0.0400	0.0000	0.0	0.4061	0.0000	
137885	2023-08-28 09:40:00	2023-08-28 09:45:00	Vessel 2	0.0895	0.0986	-0.0400	0.0000	0.0	0.4006	0.0000	
137887	2023-08-28 09:45:00	2023-08-28 09:50:00	Vessel 2	0.1265	0.1295	-0.0400	0.0000	0.0	0.3959	0.0000	
137889	2023-08-28 09:50:00	2023-08-28 09:55:00	Vessel 2	0.1095	0.1345	-0.0400	0.0000	0.0	0.3969	0.0000	
...	...	...	...	...	...	...	...	...	...	...	
153941	2023-09-25 06:45:00	2023-09-25 06:50:00	Vessel 2	0.0281	0.1347	5.8139	0.4105	0.0	0.0000	0.7854	
153943	2023-09-25 06:50:00	2023-09-25 06:55:00	Vessel 2	0.0195	0.1565	5.8844	0.4183	0.0	0.0000	0.7832	
153945	2023-09-25 06:55:00	2023-09-25 07:00:00	Vessel 2	0.0317	0.1817	5.8343	0.4083	0.0	0.0000	0.7865	
153947	2023-09-25 07:00:00	2023-09-25 07:05:00	Vessel 2	0.0166	0.2210	5.9644	0.4213	0.0	0.0000	0.7857	
153949	2023-09-25 07:05:00	2023-09-25 07:10:00	Vessel 2	0.0486	0.2097	5.9657	0.4175	0.0	0.0000	0.7886	

886 rows × 44 columns

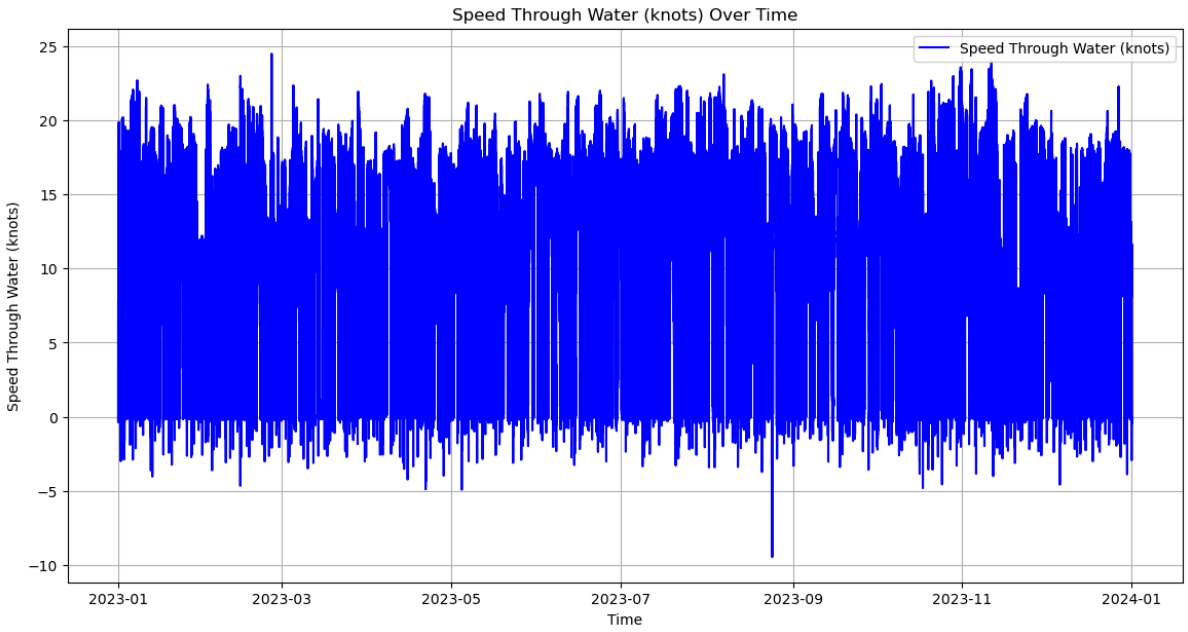
In [19]:

```
# Plot the time series for 'Speed Over Ground (knots)'
plt.figure(figsize=(14, 7))
plt.plot(df['Start Time'], df['Speed Over Ground (knots)'], label='Speed Over Ground
plt.xlabel('Time')
plt.ylabel('Speed Over Ground (knots)')
plt.title('Speed Over Ground (knots) Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



In [20]:
```
# Sort the dataframe by 'Start Time' to ensure proper filling
df = df.sort_values(by='Start Time')

# Set 'Start Time' as the index
df.set_index('Start Time', inplace=True)

# Interpolate to fill missing values
df['Speed Over Ground (knots)'] = df['Speed Over Ground (knots)'].interpolate()

# If you need to reset the index back to columns
df.reset_index(inplace=True)
```

In [21]:
```
# Set 'Start Time' as the index
df.set_index('Start Time', inplace=True)

# Interpolate to fill missing values
df['Speed Through Water (knots)'] = df['Speed Through Water (knots)'].interpolate()

# If you need to reset the index back to columns
df.reset_index(inplace=True)

# Plot the time series for 'Speed Through Water (knots)'
plt.figure(figsize=(14, 7))
plt.plot(df['Start Time'], df['Speed Through Water (knots)'], label='Speed Through W
plt.xlabel('Time')
plt.ylabel('Speed Through Water (knots)')
plt.title('Speed Through Water (knots) Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



In [22]:
```
# Set 'Start Time' as the index
df.set_index('Start Time', inplace=True)

# Interpolate to fill missing values
df['Speed Through Water (knots)'] = df['Speed Through Water (knots)'].interpolate()

# If you need to reset the index back to columns
df.reset_index(inplace=True)
```

In [23]:
```
# missing value check
df.isnull().sum()
```

Out[23]:

```
Start Time                                0
End Time                                  0
Vessel Name                               0
Power Galley 1 (MW)                       0
Power Galley 2 (MW)                       0
Power Service (MW)                        0
HVAC Chiller 1 Power (MW)                 0
HVAC Chiller 2 Power (MW)                 0
HVAC Chiller 3 Power (MW)                 0
Scrubber Power (MW)                       0
Sea Temperature (Celsius)                 0
Boiler 1 Fuel Flow Rate (L/h)             0
Boiler 2 Fuel Flow Rate (L/h)             0
Incinerator 1 Fuel Flow Rate (L/h)        0
Diesel Generator 1 Power (MW)             0
Diesel Generator 2 Power (MW)             0
Diesel Generator 3 Power (MW)             0
Diesel Generator 4 Power (MW)             0
Latitude (Degrees)                      326
Longitude (Degrees)                     326
Relative Wind Angle (Degrees)             0
True Wind Angle (Degrees)                60
Depth (m)                             57479
Relative Wind Direction (Degrees)        41
True Wind Direction (Degrees)            60
Draft (m)                              1127
Speed Over Ground (knots)                 0
True Wind Speed (knots)                  60
Relative Wind Speed (knots)               0
Speed Through Water (knots)               0
Local Time (h)                          326
Trim (m)                               1063
Propulsion Power (MW)                     0
Port Side Propulsion Power (MW)           0
Starboard Side Propulsion Power (MW)      0
Bow Thruster 1 Power (MW)                 0
Bow Thruster 2 Power (MW)                 0
Bow Thruster 3 Power (MW)                 0
Stern Thruster 1 Power (MW)               0
Stern Thruster 2 Power (MW)               0
Main Engine 1 Fuel Flow Rate (kg/h)       0
Main Engine 2 Fuel Flow Rate (kg/h)       0
Main Engine 3 Fuel Flow Rate (kg/h)       0
Main Engine 4 Fuel Flow Rate (kg/h)       0
dtype: int64
```

Sorting Vessels data to Vessel 1 & Vessel 2

In [24]:

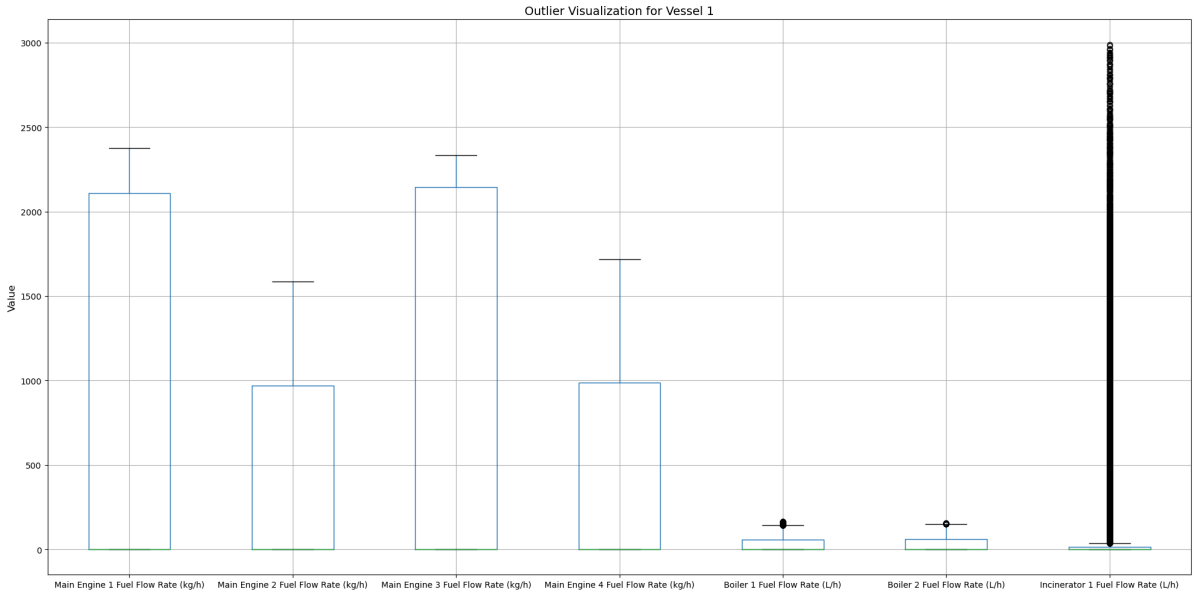
```
# Assuming 'Vessel Name' is the column indicating the vessel
vessel_1_data = df[df['Vessel Name'] == 'Vessel 1']
vessel_2_data = df[df['Vessel Name'] == 'Vessel 2']

# Verify the split
print(vessel_1_data.shape)
print(vessel_2_data.shape)
```

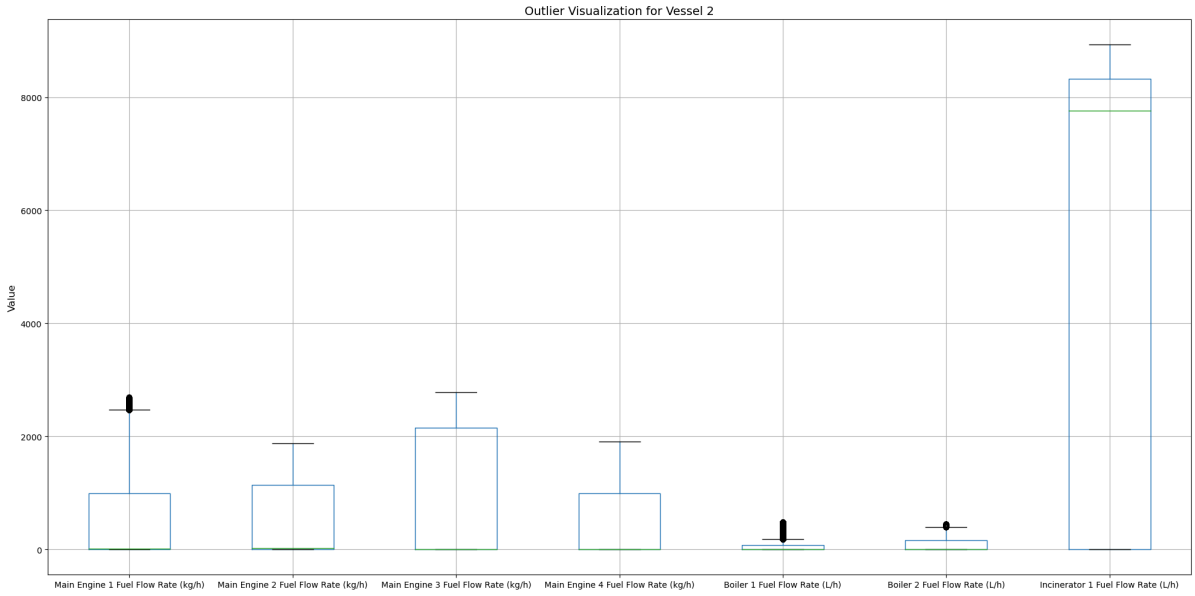
```
(105119, 44)
(105105, 44)
```

Outlier check of efficiency Analysis

In [25]:

```
# Define efficiency columns
efficiency_columns = [
    'Main Engine 1 Fuel Flow Rate (kg/h)', 'Main Engine 2 Fuel Flow Rate (kg/h)',
    'Main Engine 3 Fuel Flow Rate (kg/h)', 'Main Engine 4 Fuel Flow Rate (kg/h)',
    'Boiler 1 Fuel Flow Rate (L/h)', 'Boiler 2 Fuel Flow Rate (L/h)',
    'Incinerator 1 Fuel Flow Rate (L/h)'
]

# Assuming 'Vessel Name' is the column indicating the vessel
vessel_1_data = df[df['Vessel Name'] == 'Vessel 1'].copy()
vessel_2_data = df[df['Vessel Name'] == 'Vessel 2'].copy()

# Function to create box plots for outliers visualization
def plot_outliers(df, columns, vessel_name):
    plt.figure(figsize=(20, 10))
    df[columns].boxplot()
    plt.title(f'Outlier Visualization for {vessel_name}', fontsize=14)
    plt.ylabel('Value', fontsize=12)
    plt.xticks(rotation=0, fontsize=10)  # Change rotation to 0 for horizontal label
    plt.yticks(fontsize=10)
    plt.grid(True)
    plt.tight_layout()  # Adjust layout to make room for the labels
    plt.show()

# Plot outliers for Vessel 1
plot_outliers(vessel_1_data, efficiency_columns, 'Vessel 1')

# Plot outliers for Vessel 2
plot_outliers(vessel_2_data, efficiency_columns, 'Vessel 2')
```

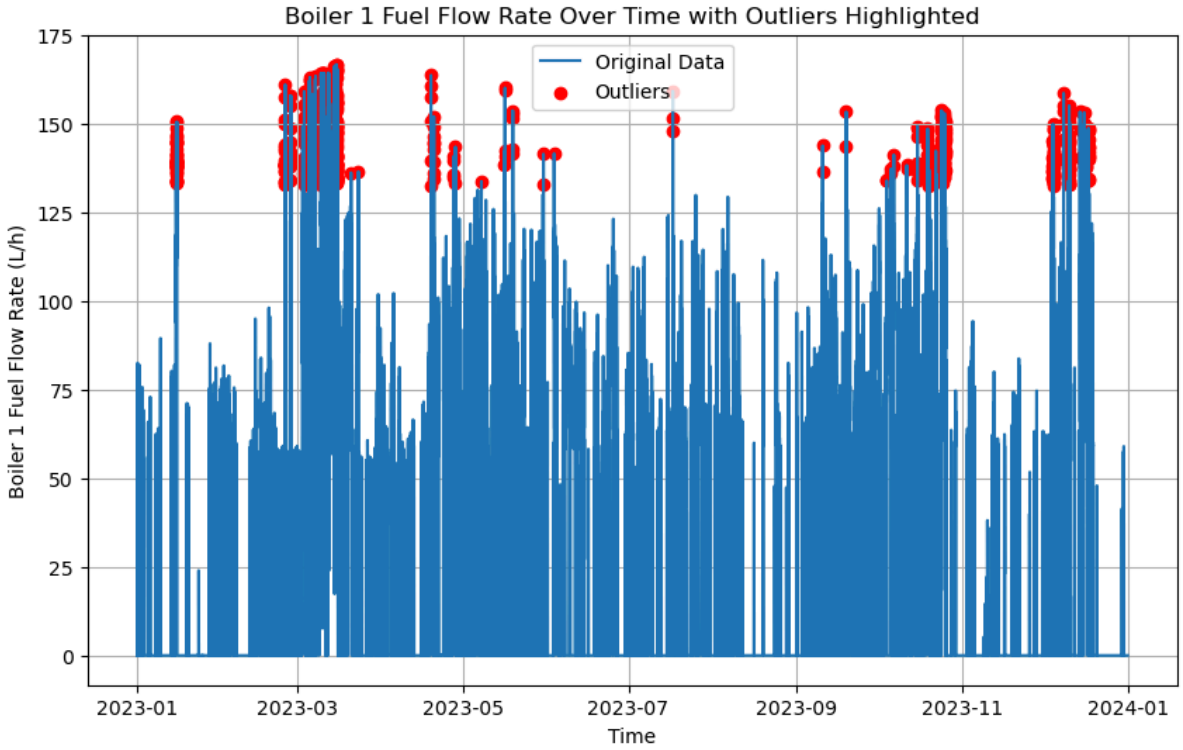




Each outlier check

In [26]:
```
# Define the column of interest for outlier detection
column_of_interest = 'Boiler 1 Fuel Flow Rate (L/h)'

# Assuming 'Vessel Name' is the column indicating the vessel
vessel_1_data = df[df['Vessel Name'] == 'Vessel 1'].copy()

# Function to identify and display outliers using Z-score method
def identify_outliers(df, column, z_thresh=3):
    z_scores = stats.zscore(df[column])
    abs_z_scores = np.abs(z_scores)
    outliers = df[abs_z_scores > z_thresh]
    return outliers

# Identify outliers in 'Boiler 1 Fuel Flow Rate (L/h)' for Vessel 1
boiler_1_outliers_vessel_1 = identify_outliers(vessel_1_data, column_of_interest)
```

In [27]:
```
# Ensure 'Start Time' is correctly set as the index
vessel_1_data.reset_index(inplace=True)
boiler_1_outliers_vessel_1.reset_index(inplace=True)

# Plot the original data
plt.figure(figsize=(10, 6))
plt.plot(vessel_1_data['Start Time'], vessel_1_data['Boiler 1 Fuel Flow Rate (L/h)']
plt.scatter(boiler_1_outliers_vessel_1['Start Time'], boiler_1_outliers_vessel_1['Bo
plt.xlabel('Time')
plt.ylabel('Boiler 1 Fuel Flow Rate (L/h)')
plt.title('Boiler 1 Fuel Flow Rate Over Time with Outliers Highlighted')
plt.legend()
plt.grid(True)
plt.show()
```



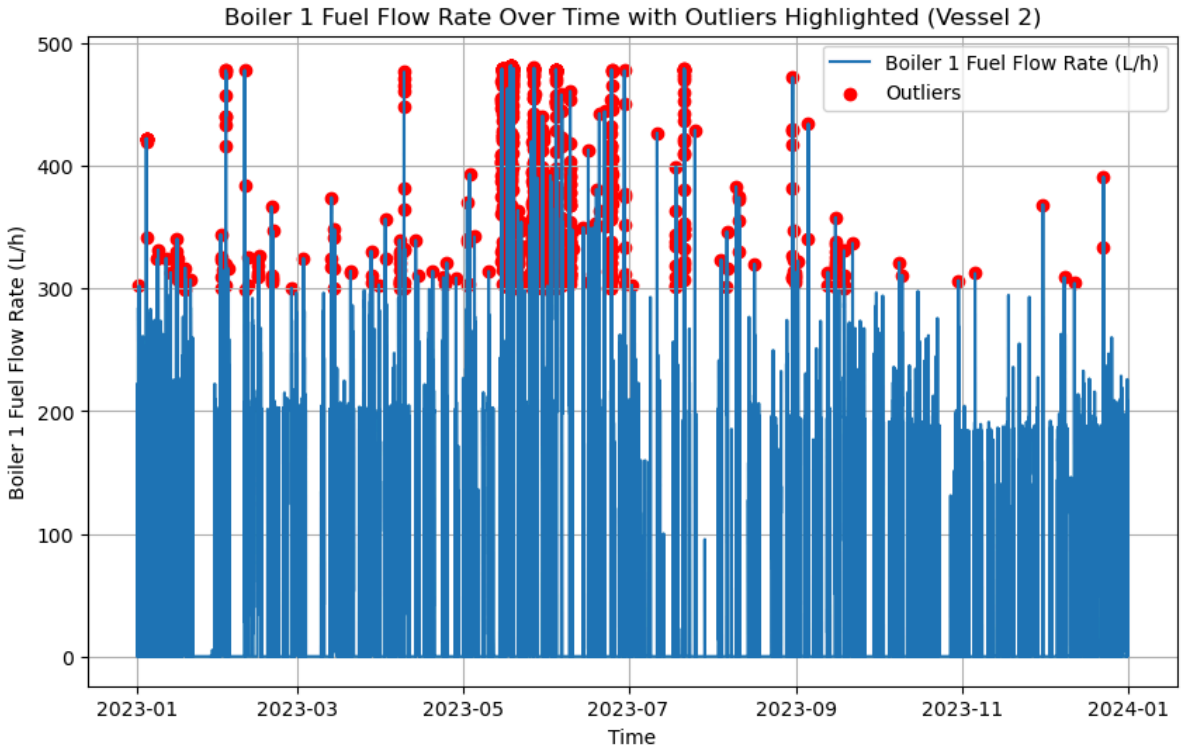
In [28]:
```
# Ensure 'Start Time' is correctly set as the index
vessel_1_data.set_index('Start Time', inplace=True)
boiler_1_outliers_vessel_1.set_index('Start Time', inplace=True)

# Statistical summary without outliers
data_without_outliers = vessel_1_data[~vessel_1_data.index.isin(boiler_1_outliers_ve
summary_with_outliers = vessel_1_data['Boiler 1 Fuel Flow Rate (L/h)'].describe()
summary_without_outliers = data_without_outliers['Boiler 1 Fuel Flow Rate (L/h)'].de

print("Summary with Outliers:₩n", summary_with_outliers)
print("₩nSummary without Outliers:₩n", summary_without_outliers)
```

```
Summary with Outliers:
 count    105119.000000
mean         24.052723
std          36.150448
min           0.000000
25%           0.000000
50%           0.000000
75%          57.318750
max         166.697000
Name: Boiler 1 Fuel Flow Rate (L/h), dtype: float64

Summary without Outliers:
 count    104304.000000
mean         23.124049
std          34.718068
min           0.000000
25%           0.000000
50%           0.000000
75%          57.100000
max         132.492000
Name: Boiler 1 Fuel Flow Rate (L/h), dtype: float64
```

Analysing with outliers becasue there are not much difference between mean, std, 75% and Max.

In [29]:
```
# Filter data for Vessel 2
vessel_2_data = df[df['Vessel Name'] == 'Vessel 2'].copy()
```

```
# Function to identify outliers
def identify_outliers(df, column):
    z_scores = stats.zscore(df[column])
    abs_z_scores = np.abs(z_scores)
    outliers = df[abs_z_scores > 3]
    return outliers

# Identify outliers in Boiler 1 Fuel Flow Rate for Vessel 2
boiler_1_outliers_vessel_2 = identify_outliers(vessel_2_data, 'Boiler 1 Fuel Flow Ra

# Plot the time series of Boiler 1 Fuel Flow Rate with outliers highlighted for Vesse
plt.figure(figsize=(10, 6))
plt.plot(vessel_2_data['Start Time'], vessel_2_data['Boiler 1 Fuel Flow Rate (L/h)']
plt.scatter(boiler_1_outliers_vessel_2['Start Time'], boiler_1_outliers_vessel_2['Bo
plt.xlabel('Time')
plt.ylabel('Boiler 1 Fuel Flow Rate (L/h)')
plt.title('Boiler 1 Fuel Flow Rate Over Time with Outliers Highlighted (Vessel 2)')
plt.legend()
plt.grid(True)
plt.show()
```



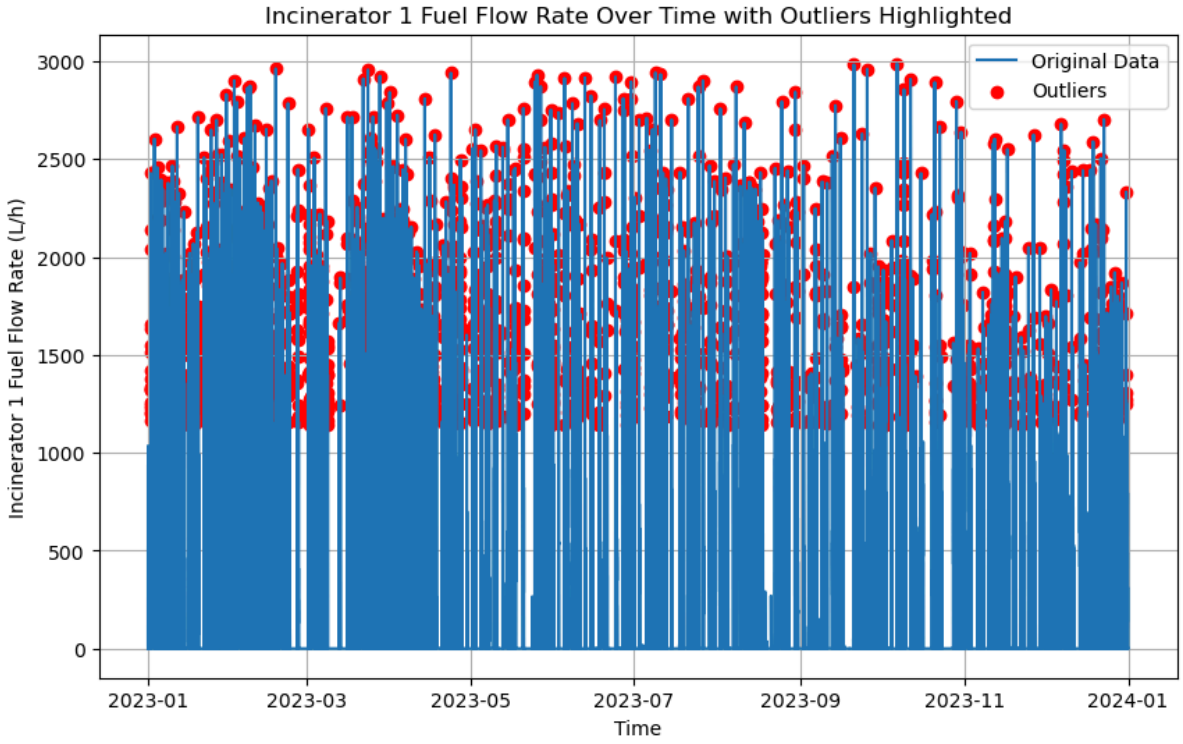
In [30]:
```
# Statistical summary without outliers
data_without_outliers_vessel_2 = vessel_2_data[~vessel_2_data.index.isin(boiler_1_ou
summary_with_outliers_vessel_2 = vessel_2_data['Boiler 1 Fuel Flow Rate (L/h)'].desc
summary_without_outliers_vessel_2 = data_without_outliers_vessel_2['Boiler 1 Fuel Fl

print("Summary with Outliers for Vessel 2:₩n", summary_with_outliers_vessel_2)
print("₩nSummary without Outliers for Vessel 2:₩n", summary_without_outliers_vessel_2
```

```
Summary with Outliers for Vessel 2:
 count    105105.000000
mean         48.085155
std          83.870875
min           0.000000
25%           0.000000
50%           0.000000
75%          74.948000
max         482.057000
Name: Boiler 1 Fuel Flow Rate (L/h), dtype: float64

Summary without Outliers for Vessel 2:
 count    104196.000000
mean         45.208148
std          78.136819
min           0.000000
25%           0.000000
50%           0.000000
75%          67.471950
max         299.521000
Name: Boiler 1 Fuel Flow Rate (L/h), dtype: float64
```

Analysing with outliers becasue outliers consistently shows a distribution

In [31]:

```python
# Define the column of interest for outlier detection
column_of_interest = 'Incinerator 1 Fuel Flow Rate (L/h)'

# Assuming 'Vessel Name' is the column indicating the vessel
vessel_1_data = df[df['Vessel Name'] == 'Vessel 1'].copy()

# Function to identify and display outliers using Z-score method
def identify_outliers(df, column, z_thresh=3):
    z_scores = stats.zscore(df[column])
    abs_z_scores = np.abs(z_scores)
    outliers = df[abs_z_scores > z_thresh]
    return outliers

# Identify outliers in 'Incinerator 1 Fuel Flow Rate (L/h)' for Vessel 1
incinerator_1_outliers_vessel_1 = identify_outliers(vessel_1_data, column_of_interes


# Ensure 'Start Time' is correctly set as the index
vessel_1_data.reset_index(inplace=True)
incinerator_1_outliers_vessel_1.reset_index(inplace=True)

# Plot the original data
plt.figure(figsize=(10, 6))
plt.plot(vessel_1_data['Start Time'], vessel_1_data[column_of_interest], label='Orig
plt.scatter(incinerator_1_outliers_vessel_1['Start Time'], incinerator_1_outliers_ve
plt.xlabel('Time')
plt.ylabel('Incinerator 1 Fuel Flow Rate (L/h)')
plt.title('Incinerator 1 Fuel Flow Rate Over Time with Outliers Highlighted')
plt.legend()
plt.grid(True)
plt.show()
```



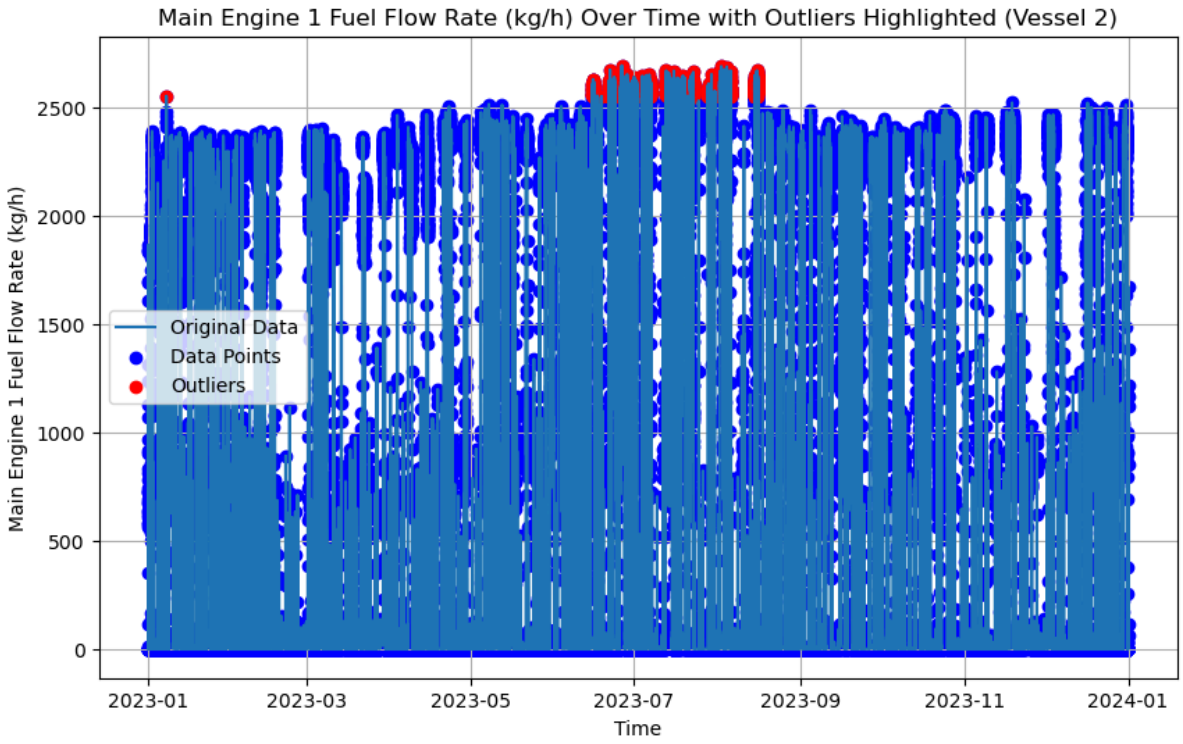
In [32]:
```
# Ensure 'Start Time' is correctly set as the index
vessel_1_data.set_index('Start Time', inplace=True)
incinerator_1_outliers_vessel_1.set_index('Start Time', inplace=True)

# Statistical summary without outliers
data_without_outliers = vessel_1_data[~vessel_1_data.index.isin(incinerator_1_outlie
summary_with_outliers = vessel_1_data[column_of_interest].describe()
summary_without_outliers = data_without_outliers[column_of_interest].describe()

print("Summary with Outliers:₩n", summary_with_outliers)
print("₩nSummary without Outliers:₩n", summary_without_outliers)
```

```
Summary with Outliers:
 count    105119.000000
mean        140.822075
std         333.396496
min           0.000000
25%           0.000000
50%           0.000000
75%          14.304000
max        2986.980000
Name: Incinerator 1 Fuel Flow Rate (L/h), dtype: float64

Summary without Outliers:
 count    102383.000000
mean        101.832632
std         226.663019
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max        1140.978000
Name: Incinerator 1 Fuel Flow Rate (L/h), dtype: float64
```

Analysing with outliers becasue outliers consistently shows a distribution

In [33]:
```
# Function to plot outliers for 'Main Engine 1 Fuel Flow Rate (kg/h)'
def plot_main_engine_outliers(df, column, vessel_name):
    plt.figure(figsize=(10, 6))
```

```
    plt.plot(df['Start Time'], df[column], label='Original Data')
    plt.scatter(df['Start Time'], df[column], c='blue', label='Data Points')
    outliers = df[df[column] > df[column].mean() + 2 * df[column].std()]
    plt.scatter(outliers['Start Time'], outliers[column], c='red', label='Outliers')
    plt.xlabel('Time')
    plt.ylabel(f'{column}')
    plt.title(f'{column} Over Time with Outliers Highlighted ({vessel_name})')
    plt.legend()
    plt.grid(True)
    plt.show()

# Assuming you have a DataFrame for Vessel 2 similar to vessel_2_data
# Example usage for 'Main Engine 1 Fuel Flow Rate (kg/h)' for 'Vessel 2'
plot_main_engine_outliers(vessel_2_data, 'Main Engine 1 Fuel Flow Rate (kg/h)', 'Vess
```



Analysing with outliers becasue outliers effects really minumum to overall distribution

Data Analysis

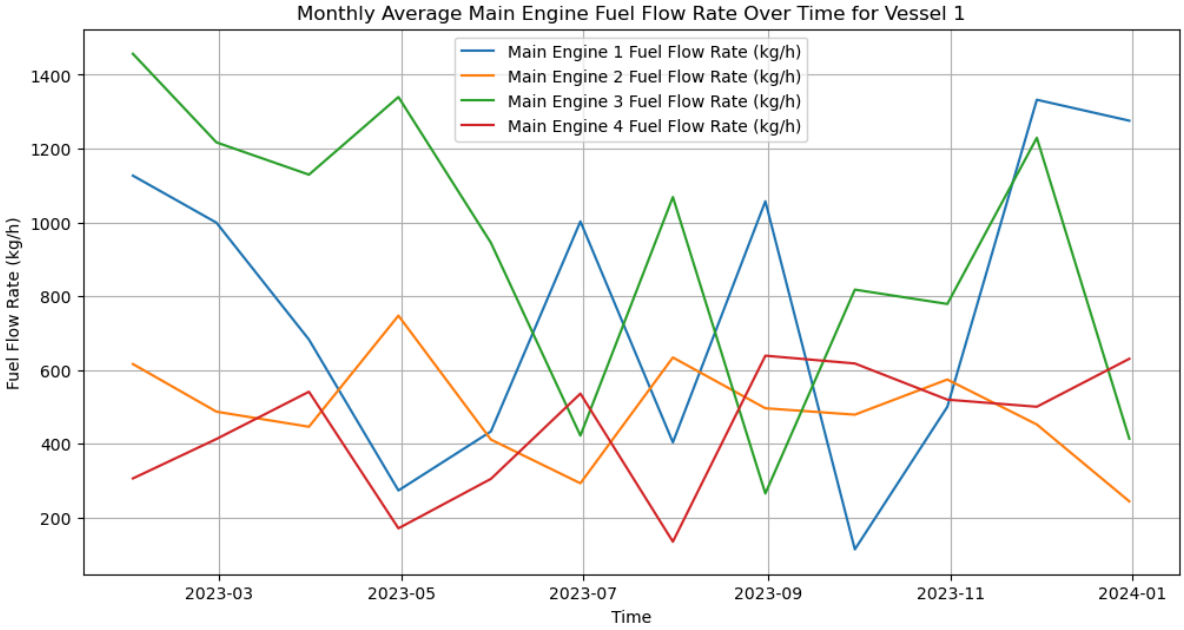
In [34]:

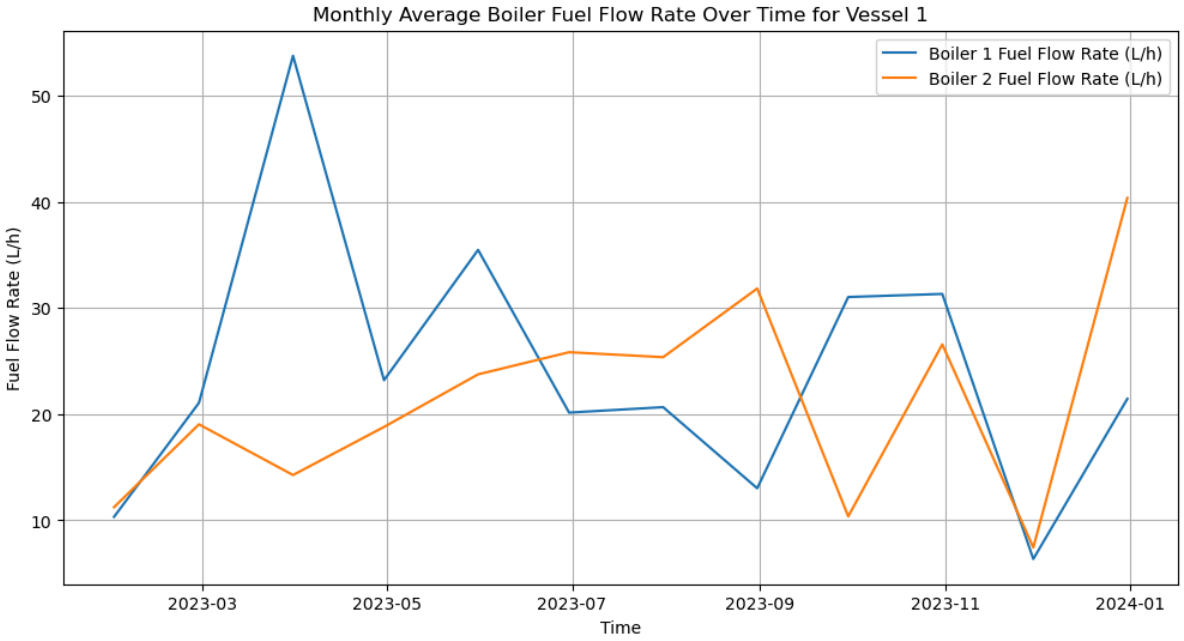
```
# Ensure 'Start Time' is in datetime format and set as index
df['Start Time'] = pd.to_datetime(df['Start Time'])
vessel_1_data = df[df['Vessel Name'] == 'Vessel 1'].copy()
vessel_2_data = df[df['Vessel Name'] == 'Vessel 2'].copy()
vessel_1_data.set_index('Start Time', inplace=True)
vessel_2_data.set_index('Start Time', inplace=True)

# Define efficiency columns
efficiency_columns = [
    'Main Engine 1 Fuel Flow Rate (kg/h)', 'Main Engine 2 Fuel Flow Rate (kg/h)',
    'Main Engine 3 Fuel Flow Rate (kg/h)', 'Main Engine 4 Fuel Flow Rate (kg/h)',
    'Boiler 1 Fuel Flow Rate (L/h)', 'Boiler 2 Fuel Flow Rate (L/h)',
    'Incinerator 1 Fuel Flow Rate (L/h)'
]

# Calculate monthly averages for Vessel 1
vessel_1_monthly_avg = vessel_1_data[efficiency_columns].resample('M').mean()
```

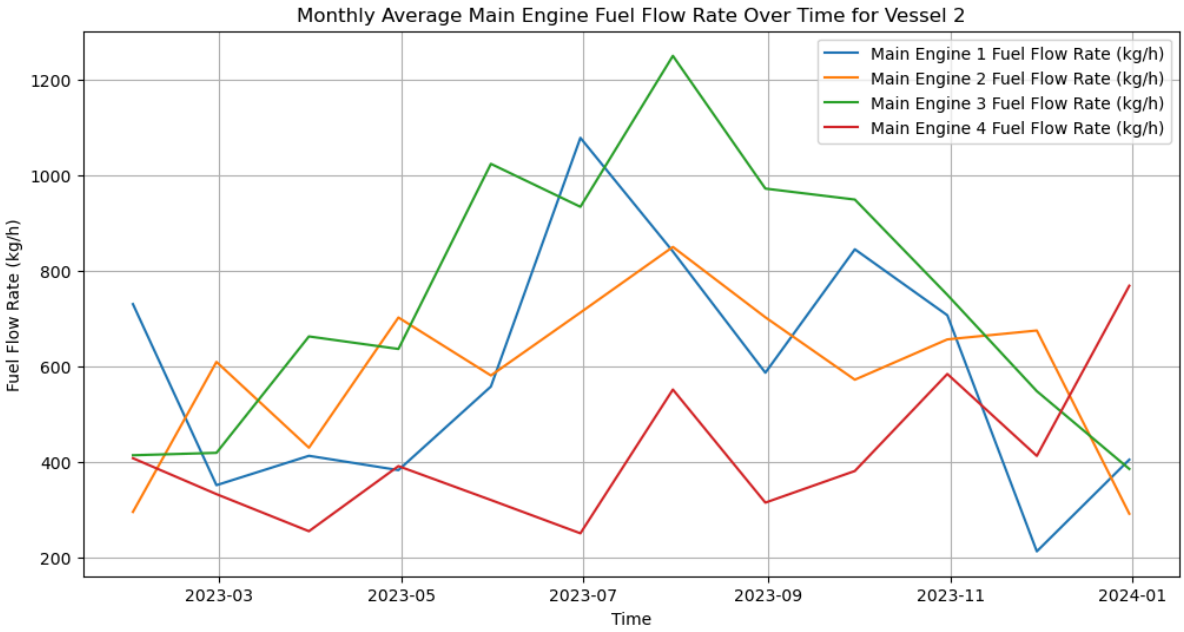
```
# Calculate monthly averages for Vessel 2
vessel_2_monthly_avg = vessel_2_data[efficiency_columns].resample('M').mean()

# Plotting monthly averages for Main Engine Fuel Flow Rates for Vessel 1
plt.figure(figsize=(12, 6))
plt.plot(vessel_1_monthly_avg.index, vessel_1_monthly_avg['Main Engine 1 Fuel Flow R
plt.plot(vessel_1_monthly_avg.index, vessel_1_monthly_avg['Main Engine 2 Fuel Flow R
plt.plot(vessel_1_monthly_avg.index, vessel_1_monthly_avg['Main Engine 3 Fuel Flow R
plt.plot(vessel_1_monthly_avg.index, vessel_1_monthly_avg['Main Engine 4 Fuel Flow R
plt.xlabel('Time')
plt.ylabel('Fuel Flow Rate (kg/h)')
plt.title('Monthly Average Main Engine Fuel Flow Rate Over Time for Vessel 1')
plt.legend()
plt.grid(True)
plt.show()

# Plotting monthly averages for Boiler Fuel Flow Rates for Vessel 1
plt.figure(figsize=(12, 6))
plt.plot(vessel_1_monthly_avg.index, vessel_1_monthly_avg['Boiler 1 Fuel Flow Rate (
plt.plot(vessel_1_monthly_avg.index, vessel_1_monthly_avg['Boiler 2 Fuel Flow Rate (
plt.xlabel('Time')
plt.ylabel('Fuel Flow Rate (L/h)')
plt.title('Monthly Average Boiler Fuel Flow Rate Over Time for Vessel 1')
plt.legend()
plt.grid(True)
plt.show()

# Plotting monthly averages for Main Engine Fuel Flow Rates for Vessel 2
plt.figure(figsize=(12, 6))
plt.plot(vessel_2_monthly_avg.index, vessel_2_monthly_avg['Main Engine 1 Fuel Flow R
plt.plot(vessel_2_monthly_avg.index, vessel_2_monthly_avg['Main Engine 2 Fuel Flow R
plt.plot(vessel_2_monthly_avg.index, vessel_2_monthly_avg['Main Engine 3 Fuel Flow R
plt.plot(vessel_2_monthly_avg.index, vessel_2_monthly_avg['Main Engine 4 Fuel Flow R
plt.xlabel('Time')
plt.ylabel('Fuel Flow Rate (kg/h)')
plt.title('Monthly Average Main Engine Fuel Flow Rate Over Time for Vessel 2')
plt.legend()
plt.grid(True)
plt.show()

# Plotting monthly averages for Boiler Fuel Flow Rates for Vessel 2
plt.figure(figsize=(12, 6))
plt.plot(vessel_2_monthly_avg.index, vessel_2_monthly_avg['Boiler 1 Fuel Flow Rate (
plt.plot(vessel_2_monthly_avg.index, vessel_2_monthly_avg['Boiler 2 Fuel Flow Rate (
plt.xlabel('Time')
plt.ylabel('Fuel Flow Rate (L/h)')
plt.title('Monthly Average Boiler Fuel Flow Rate Over Time for Vessel 2')
plt.legend()
plt.grid(True)
plt.show()

# Calculate monthly averages for Incinerator 1 Fuel Flow Rate (L/h) for Vessel 1
vessel_1_monthly_avg_incinerator = vessel_1_data['Incinerator 1 Fuel Flow Rate (L/h)

# Calculate monthly averages for Incinerator 1 Fuel Flow Rate (L/h) for Vessel 2
vessel_2_monthly_avg_incinerator = vessel_2_data['Incinerator 1 Fuel Flow Rate (L/h)

# Plotting monthly averages for Incinerator 1 Fuel Flow Rate (L/h) for Vessel 1
plt.figure(figsize=(12, 6))
plt.plot(vessel_1_monthly_avg_incinerator.index, vessel_1_monthly_avg_incinerator, l
plt.xlabel('Time')
plt.ylabel('Incinerator 1 Fuel Flow Rate (L/h)')
plt.title('Monthly Average Incinerator 1 Fuel Flow Rate Over Time for Vessel 1')
plt.legend()
```

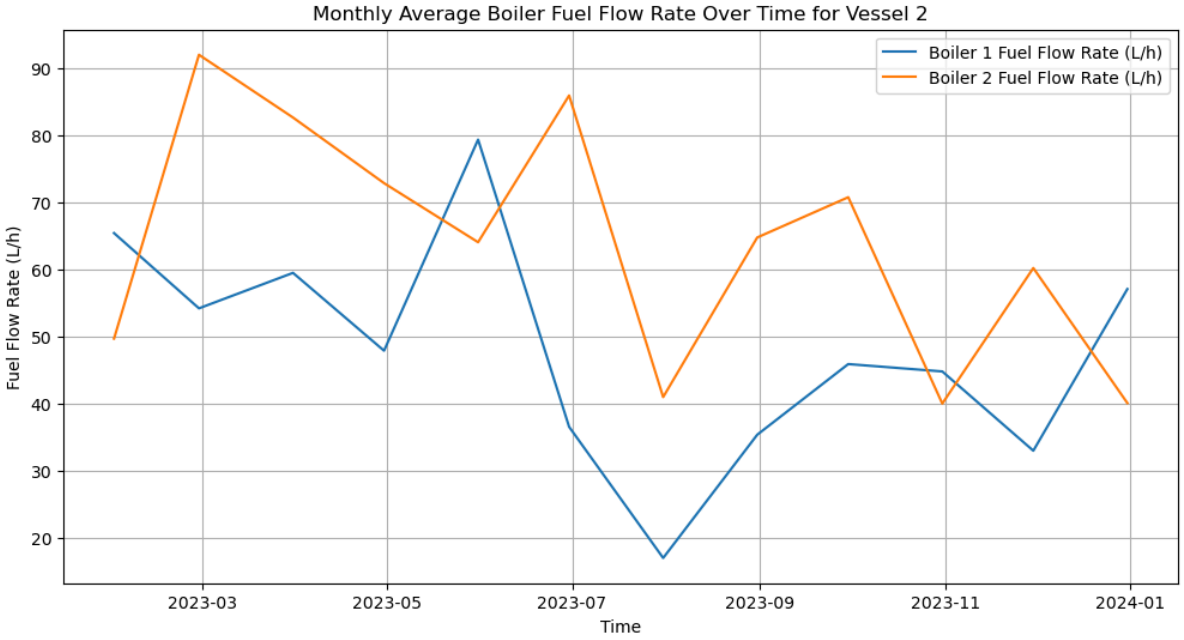
```
plt.grid(True)
plt.show()

# Plotting monthly averages for Incinerator 1 Fuel Flow Rate (L/h) for Vessel 2
plt.figure(figsize=(12, 6))
plt.plot(vessel_2_monthly_avg_incinerator.index, vessel_2_monthly_avg_incinerator, l
plt.xlabel('Time')
plt.ylabel('Incinerator 1 Fuel Flow Rate (L/h)')
plt.title('Monthly Average Incinerator 1 Fuel Flow Rate Over Time for Vessel 2')
plt.legend()
plt.grid(True)
plt.show()
```

```
C:\Users\seoin\AppData\Local\Temp\ipykernel_12856\2046262440.py:17: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  vessel_1_monthly_avg = vessel_1_data[efficiency_columns].resample('M').mean()
C:\Users\seoin\AppData\Local\Temp\ipykernel_12856\2046262440.py:20: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  vessel_2_monthly_avg = vessel_2_data[efficiency_columns].resample('M').mean()
```

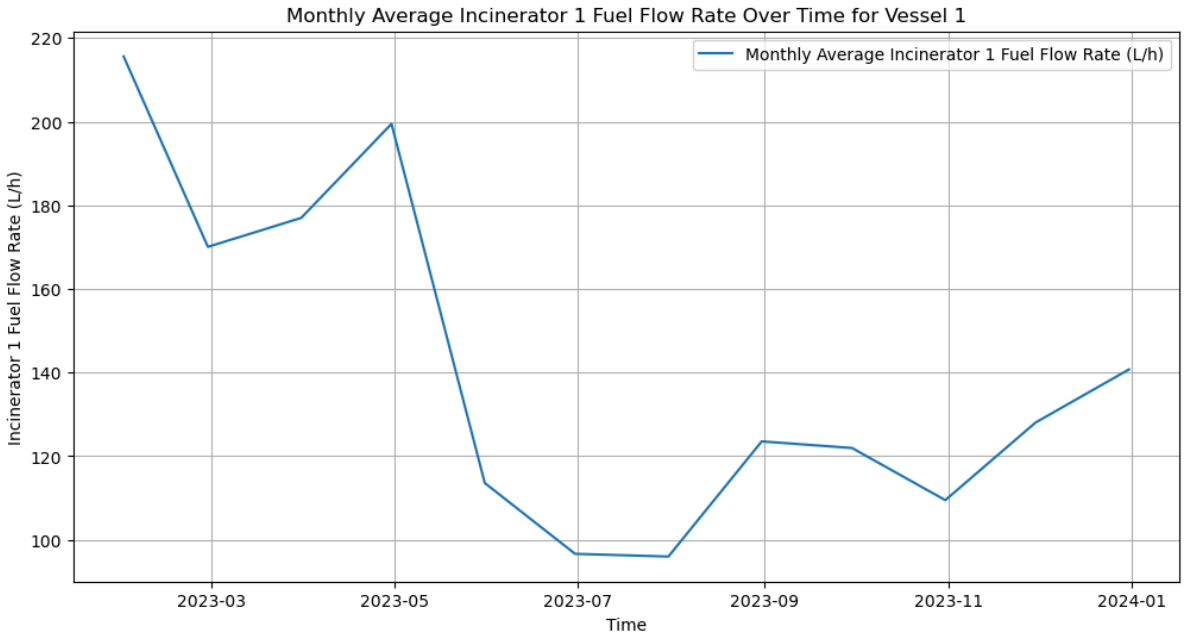


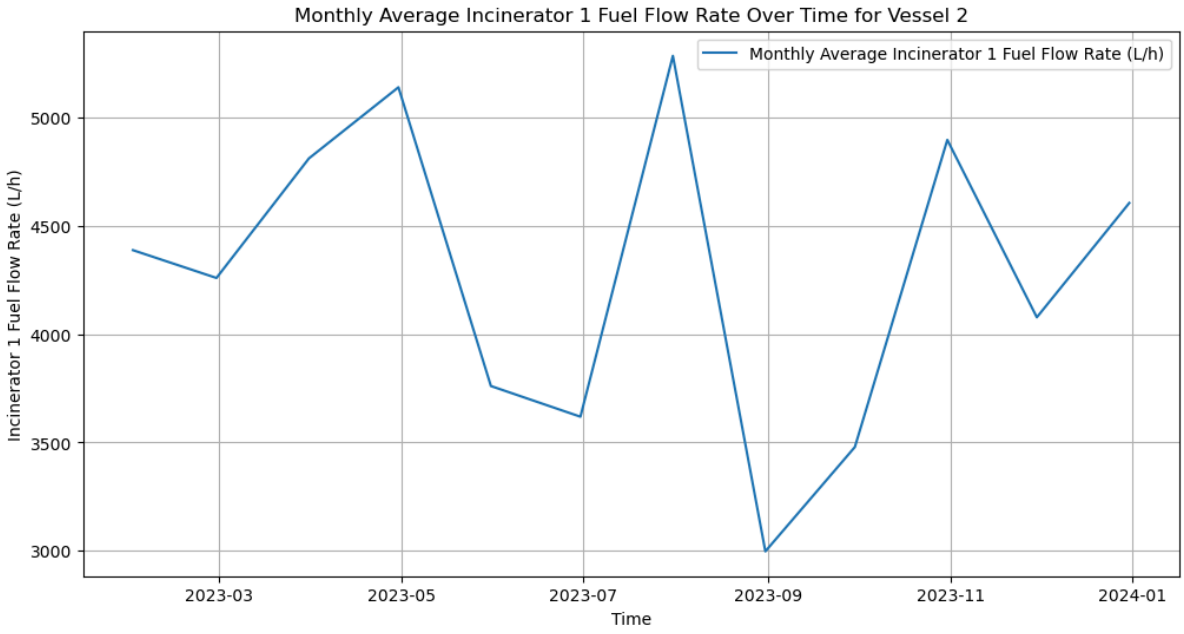




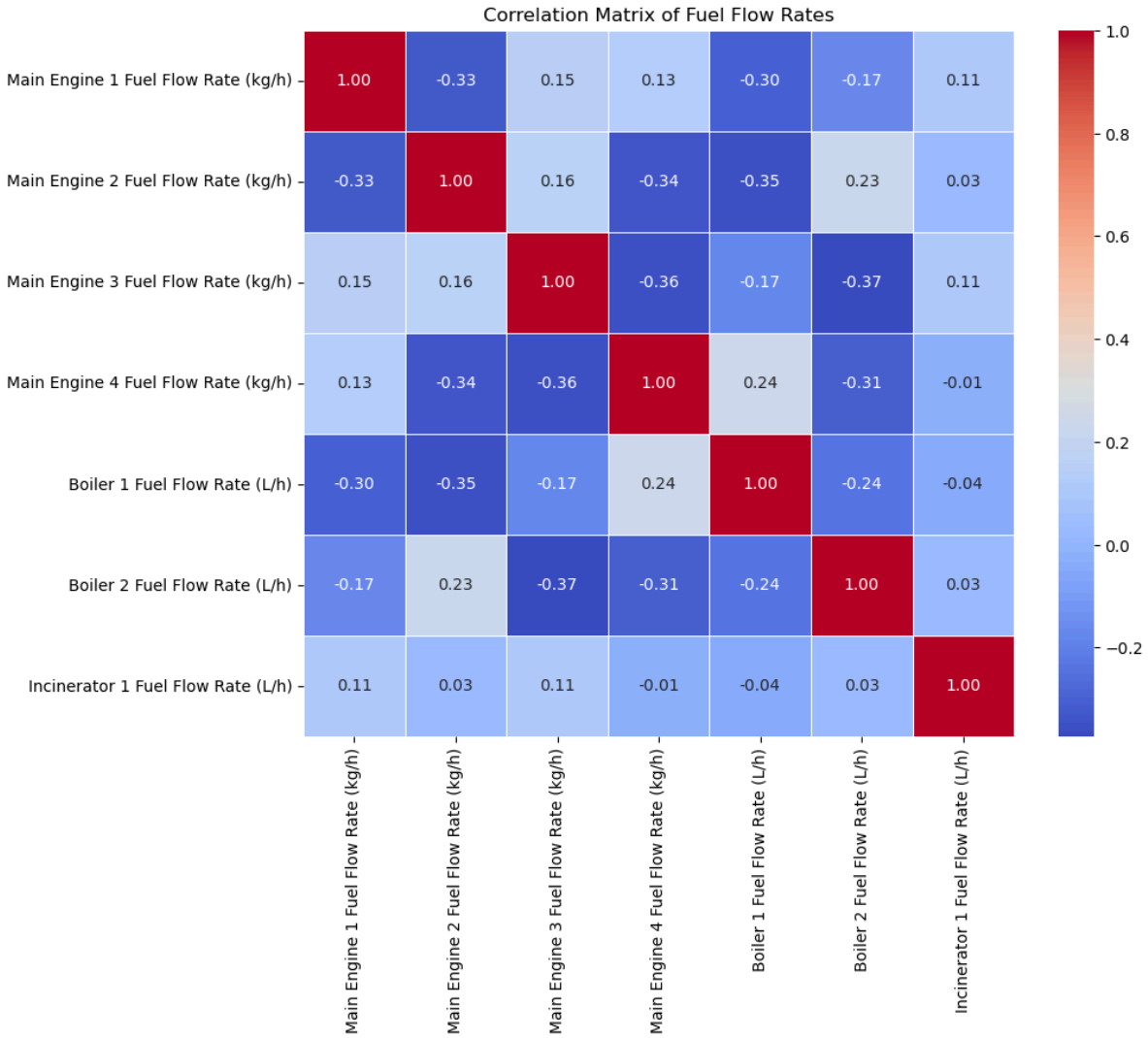



```
C:₩Users₩seoin₩AppData₩Local₩Temp₩ipykernel_12856₩2046262440.py:71: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  vessel_1_monthly_avg_incinerator = vessel_1_data['Incinerator 1 Fuel Flow Rate (L/
h)'].resample('M').mean()
C:₩Users₩seoin₩AppData₩Local₩Temp₩ipykernel_12856₩2046262440.py:74: FutureWarning:
'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  vessel_2_monthly_avg_incinerator = vessel_2_data['Incinerator 1 Fuel Flow Rate (L/
h)'].resample('M').mean()
```





In [35]:
```
# Calculate the correlation matrix
correlation_matrix = df[efficiency_columns].corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=
plt.title('Correlation Matrix of Fuel Flow Rates')
plt.show()
```



In [36]:

```
# Plotting and storing analysis results as DataFrames
results = []

# Efficiency Analysis for Vessel 1
efficiency_vessel_1 = vessel_1_monthly_avg.copy()
efficiency_vessel_1['Vessel'] = 'Vessel 1'
results.append(efficiency_vessel_1)

# Efficiency Analysis for Vessel 2
efficiency_vessel_2 = vessel_2_monthly_avg.copy()
efficiency_vessel_2['Vessel'] = 'Vessel 2'
results.append(efficiency_vessel_2)

# Combine all results
combined_results = pd.concat(results)

# Store combined results into SQLite database
combined_results.to_sql('efficiency_analysis', conn, if_exists='replace', index=True

# Verify data is stored correctly
query = "SELECT * FROM efficiency_analysis"
df_sql = pd.read_sql_query(query, conn)
print(df_sql.head())

# Close the connection
conn.close()
```

	Start Time	Main Engine 1 Fuel Flow Rate (kg/h)	₩
0	2023-01-31 00:00:00	1126.462229	
1	2023-02-28 00:00:00	999.000141	
2	2023-03-31 00:00:00	682.419954	
3	2023-04-30 00:00:00	273.527550	
4	2023-05-31 00:00:00	433.459121	

	Main Engine 2 Fuel Flow Rate (kg/h)	Main Engine 3 Fuel Flow Rate (kg/h)	₩
0	615.993675	1456.498448	
1	487.008028	1216.616883	
2	446.170489	1129.180494	
3	747.247725	1339.480324	
4	411.406155	944.593492	

	Main Engine 4 Fuel Flow Rate (kg/h)	Boiler 1 Fuel Flow Rate (L/h)	₩
0	306.167109	10.324022	
1	412.993884	21.067516	
2	541.052423	53.766229	
3	170.952196	23.203172	
4	304.915498	35.477180	

	Boiler 2 Fuel Flow Rate (L/h)	Incinerator 1 Fuel Flow Rate (L/h)	Vessel
0	11.235872	215.598515	Vessel 1
1	19.044405	170.054763	Vessel 1
2	14.267926	176.980471	Vessel 1
3	18.800144	199.465167	Vessel 1
4	23.754018	113.587768	Vessel 1

Efficiency Performance Trend Analysis

Introduction This report provides an in-depth analysis of the efficiency performance trends for two cruise ships, Vessel 1 and Vessel 2, based on their fuel flow rates for various engines and boilers. The analysis covers the period from early 2023 to early 2024, focusing on monthly average fuel flow rates for main engines, boilers, and incinerators.

Data Preparation

The dataset includes the following columns:

Main Engine 1 Fuel Flow Rate (kg/h) Main Engine 2 Fuel Flow Rate (kg/h) Main Engine 3 Fuel Flow Rate (kg/h) Main Engine 4 Fuel Flow Rate (kg/h) Boiler 1 Fuel Flow Rate (L/h) Boiler 2 Fuel Flow Rate (L/h) Incinerator 1 Fuel Flow Rate (L/h)

The dataset was processed to handle missing values using linear interpolation and ensure all necessary columns were in numeric format.

Analysis and Findings

Main Engine Fuel Flow Rates for Vessel 1:

- Monthly Average Trends: There is a noticeable decline in the fuel flow rates for Main Engine 1 and Main Engine 2 from January to July 2023. Main Engine 3 and Main Engine 4 show more variability with spikes in May, July, and December 2023. The variability suggests possible changes in operational patterns or maintenance activities during these periods.

- Interpretation: The decline in fuel flow rates for Main Engine 1 and Main Engine 2 may indicate improvements in operational efficiency or reduced load on these engines. The spikes in Main Engine 3 and Main Engine 4 might be due to increased demand or periods of higher operational intensity.

Boiler Fuel Flow Rates for Vessel 1:

- Monthly Average Trends: Both Boiler 1 and Boiler 2 show significant fluctuations throughout the year. Boiler 1 experienced peaks in April and August 2023, while Boiler 2 had peaks in August and December 2023.
- Interpretation: The peaks in fuel flow rates for the boilers may correspond to periods of increased onboard heating or hot water demand. The fluctuations suggest varying operational requirements and possibly seasonal changes impacting boiler usage.

Main Engine Fuel Flow Rates for Vessel 2:

- Monthly Average Trends: Main Engine 1 and Main Engine 2 fuel flow rates show an increasing trend from May to October 2023, followed by a decline towards the end of the year. Main Engine 3 and Main Engine 4 also exhibit similar variability with peaks in mid-2023.
- Interpretation: The increasing trend in mid-2023 indicates periods of higher operational activity or less efficient operation during these months. The decline at the end of the year could be attributed to reduced operational demand or efficiency improvements.

Boiler Fuel Flow Rates for Vessel 2:

- Monthly Average Trends: Boiler 1 fuel flow rate peaked in March 2023 and then showed a declining trend. Boiler 2 exhibited peaks in March and November 2023, with a general decline in other months.
- Interpretation: The peaks in Boiler 1 and Boiler 2 suggest periods of high demand for heating or other boiler-related services. The overall decline could be due to improved efficiency or reduced operational needs.

Incinerator Fuel Flow Rates:

- Vessel 1: The fuel flow rate for Incinerator 1 shows a declining trend from January to June 2023, followed by fluctuations for the rest of the year.
- Vessel 2: The fuel flow rate for Incinerator 1 fluctuates throughout the year, with peaks in mid-2023. Interpretation:

The declining trend for Vessel 1 suggests reduced incinerator usage or improved waste management practices. Fluctuations in Vessel 2 may indicate variable waste generation rates or differing operational needs.

Correlation Analysis of Fuel Flow Rates:

The correlation matrix shows the relationships between the fuel flow rates of different engines and boilers.

Key Insights:

There is a negative correlation between Main Engine 1 and Main Engine 2 fuel flow rates, indicating that when one engine's fuel flow rate increases, the other's tends to decrease. Main Engine 3 and Main Engine 4 have a moderate negative correlation with Boiler 1 and Boiler 2 fuel flow rates, suggesting that higher fuel usage in the main engines might be associated with lower fuel usage in the boilers. Incinerator 1 fuel flow rate shows weak correlations with other components, indicating independent operation.

Conclusion

Both vessels exhibit variability in fuel flow rates across different engines and boilers, indicating changes in operational patterns, demand, and possibly maintenance activities. Peaks in fuel flow rates often correspond to periods of higher operational activity or increased demand for specific services (e.g., heating). Declining trends in some fuel flow rates suggest improvements in efficiency or reduced operational demand over time. The correlation analysis highlights relationships between the fuel usage of different components, providing insights into operational dependencies and efficiencies.