In [2]:
```
import pandas as pd
#to see max rows by scrolling
pd.set_option('display.max_rows',None)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_8236\3739907820.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with oth
er libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

In [3]:
```
df = pd.read_excel(r'C:\Users\DELL\Desktop\Gridscope_Simulated_Data.xlsx')
df.head(10)
```

Out[3]:

	Sensor_ID	Timestamp	Fault_Type	Latitude	Longitude	Temperature	Wind_Speed	Pole_Health_Score	Communicat
0	SENSOR_1	2024-05-05 18:58:51.373	Line Break	48.916904	-136.200822	6.822513	66.096399	87	
1	SENSOR_2	2023-10-27 03:52:51.373	Line Break	30.326873	-30.918149	-20.408081	118.265484	62	
2	SENSOR_3	2024-04-09 14:12:51.373	Insulator Flashover	-25.471663	-37.032229	44.512369	65.442825	12	
3	SENSOR_4	2023-12-28 22:23:51.373	Vegetation Contact	73.246140	-36.430981	16.735903	116.220445	80	
4	SENSOR_5	2024-02-23 23:35:51.373	Lightning Strike	31.705666	-77.497165	-9.364069	76.987315	63	Device-t
5	SENSOR_6	2024-02-09 07:20:51.373	Vegetation Contact	-52.354753	-100.237611	31.894522	113.978402	62	
6	SENSOR_7	2023-10-20 05:40:51.373	Line Break	29.377679	12.304004	5.021616	69.319154	34	Device-t
7	SENSOR_8	2024-03-08 14:41:51.373	Insulator Flashover	28.685329	130.817174	1.567893	93.387686	25	Device-t
8	SENSOR_9	2023-07-01 23:53:51.373	Line Break	17.683781	-151.420648	24.081058	109.654604	53	
9	SENSOR_10	2023-09-04 21:58:51.373	Equipment Failure	17.229716	-100.671571	-23.125694	74.724815	28	

In [4]:
```
# Check for missing values
missing_values=df.isnull().sum()
missing_values
```

Out[4]:
```
Sensor_ID           0
Timestamp           0
Fault_Type          0
Latitude            0
Longitude           0
Temperature         0
Wind_Speed          0
Pole_Health_Score   0
Communication_Type  0
dtype: int64
```

In [5]:
```
df['Timestamp']=pd.to_datetime(df['Timestamp'])
df.dtypes
```

Out[5]:
```
Sensor_ID                     object
Timestamp             datetime64[ns]
Fault_Type                    object
Latitude                     float64
Longitude                    float64
Temperature                  float64
Wind_Speed                   float64
Pole_Health_Score              int64
Communication_Type            object
dtype: object
```

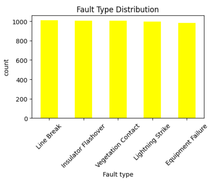
In [6]:
```
discriptive_stats=df.describe()
discriptive_stats
```

Out[6]:

	Timestamp	Latitude	Longitude	Temperature	Wind_Speed	Pole_Health_Score
count	5000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	2023-12-25 09:30:27.181000192	1.359573	-0.413938	10.339232	75.421367	50.221400
min	2023-06-26 07:43:51.373000	-89.974584	-179.997910	-29.988553	0.010258	1.000000
25%	2023-09-26 12:06:51.372999936	-44.170576	-92.726830	-9.643684	37.568199	25.000000
50%	2023-12-26 06:56:51.372999936	3.140578	0.547763	10.462031	75.792263	50.000000
75%	2024-03-26 01:38:06.372999936	47.008499	90.091019	30.230583	113.393260	75.000000
max	2024-06-25 06:16:51.373000	89.996496	179.914622	49.989231	149.990570	100.000000
std	NaN	52.410823	104.420086	23.185587	43.633872	28.729592

Fault Type Distribution

In [7]:
```
import matplotlib.pyplot as plt

# Plot the distribution of fault types
plt.figure(figsize=(5,3))
df['Fault_Type'].value_counts().plot(kind='bar',color='yellow')
plt.title('Fault Type Distribution')
plt.xlabel('Fault type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

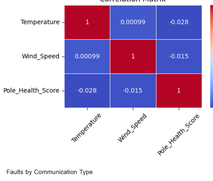


Faults Over Time

In [8]:
```
# Group data by date and count the number of faults
Faults_over_time = df['Timestamp'].dt.date.value_counts().sort_index()

# Plot the number of faults over time
plt.figure(figsize=(12,3))
Faults_over_time.plot(kind='line', color='green')
plt.xlabel('Number of Faults Over Time')
plt.ylabel('Count')
plt.grid(True)

plt.show()
```

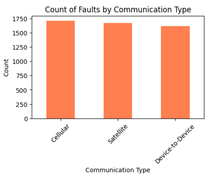
Correlation Analysis

In [9]:
```
# Calculate correlation matrix
corr_matrix = df[['Temperature', 'Wind_Speed', 'Pole_Health_Score']].corr()

# Plot correlation heatmap
import seaborn as sns

plt.figure(figsize=(5, 3))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.xticks(rotation=45)
plt.show()
```



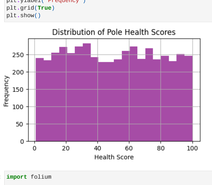
Faults by Communication Type

In [10]:
```
# Plot the count of faults by communication type

plt.figure(figsize=(5, 3))
df['Communication_Type'].value_counts().plot(kind='bar', color='coral')
plt.title('Count of Faults by Communication Type')
plt.xlabel('Communication Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



Pole Health Distribution

In [11]:
```
# Plot the distribution of pole health scores

plt.figure(figsize=(5, 3))
df['Pole_Health_Score'].plot(kind='hist', bins=20, color='purple', alpha=0.7)
plt.title('Distribution of Pole Health Scores')
plt.xlabel('Health Score')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



In [12]:
```
import folium

# Initialize a map centered at the mean location
m = folium.Map(location=[df['Latitude'].mean(), df['Longitude'].mean()], zoom_start=2)

# Add fault locations to the map
for _, row in df.iterrows():
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=f"Fault Type: {row['Fault_Type']}\nTemperature: {row['Temperature']}°C\nWind Speed: {row['
        icon=folium.Icon(color='red', icon='info-sign')
    ).add_to(m)

# Save the map to an HTML file
map_file_path = "fault_locations_map.html"
m.save(map_file_path)

map_file_path
```

Out[12]:
```
'fault_locations_map.html'
```

Time-Series Analysis of Faults

In [13]:
```
# Resample the data to get weekly counts of faults
weekly_faults = df.set_index('Timestamp').resample('W').size()

# Plot the weekly faults time series
plt.figure(figsize=(5,3))
weekly_faults.plot(kind='line', color='navy')
plt.title('Weekly Number of Faults')
plt.xlabel('Date')
plt.ylabel('Number of Faults')
plt.grid(True)
plt.show()
```