In [ ]:
```
!pip install pandas matplotlib
```

In [20]:
```
import pandas as pd
import matplotlib. pyplot as plt
```

In [62]:
```
file_path = r"C:\Users\dell\OneDrive\Documents\Python Scripts\Drilling_Data_Sample.CSV"
def load_data(file_path):
    """Reads a CSV file into a DataFrame."""
    try:
        data = pd.read_csv(file_path)
        print("Data successfully loaded!")
        return data
    except Exception as e:
        print(f"Error loading data: {e}")
        return None

data = load_data(file_path)

if data is not None:
    print(data.head())
```

```
Data successfully loaded!
                    Time MD_DMEA MD_DBTM MD_ROP   MD_SWOB MD_TDRPM  \
0                     ms      ft      ft   ft/h  1000 lbf    c/min
1  24-03-2024 6:00:13 AM    2320    2320    197      20.1       99
2  24-03-2024 6:10:21 AM    2348    2348    147      17.2       97
3  24-03-2024 6:20:20 AM    2374    2374    128      17.3       99
4  24-03-2024 6:40:46 AM    2380    2380     39      13.8      100

      MD_TDTQR MD_BPOS   MD_HKLD MD_SPPA      MD_TFLO
0  1000 ft.lbf      ft  1000 lbf     psi      bbl/min
1           11      36       125    2351   24.9047619
2         10.4      36       125    2230           25
3          8.5      13       125    2815  25.02380952
4          6.8      84       125    2977  24.92857143
```

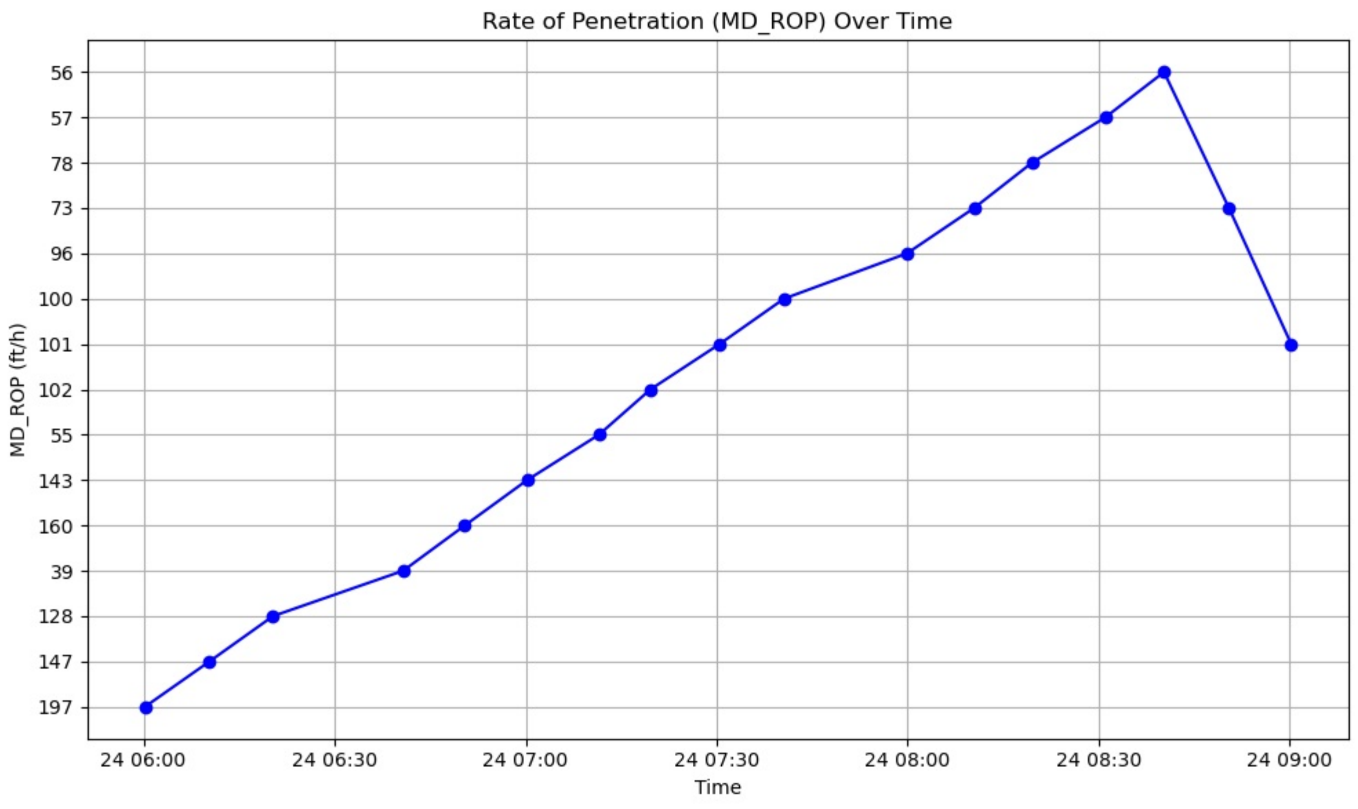
In [183…
```
def clean_data(data):
    """Cleans and wrangles the data by handling missing values, normalizing column names, and reporting types."
    data['Time'] = pd.to_datetime(data['Time'], errors='coerce')
    print("Cleaning data...")
    data = data.dropna()
    data.columns = [col.strip().replace(" ", "_").lower() for col in data.columns]
    print("Column names normalized.")

    for col in data.columns:
        dtype = data[col].dtype
        if dtype == 'object':
            continue
        elif pd.api.types.is_numeric_dtype(data[col]):
            data[col] = data[col].clip(upper=data[col].quantile(0.99))
        print("Data wrangling complete!")
        return data

print(data.head())
```

```
                 Time MD_DMEA MD_DBTM MD_ROP   MD_SWOB MD_TDRPM     MD_TDTQR  \
0                 NaT      ft      ft   ft/h  1000 lbf    c/min  1000 ft.lbf
1 2024-03-24 06:00:13    2320    2320    197      20.1       99           11
2 2024-03-24 06:10:21    2348    2348    147      17.2       97         10.4
3 2024-03-24 06:20:20    2374    2374    128      17.3       99          8.5
4 2024-03-24 06:40:46    2380    2380     39      13.8      100          6.8

  MD_BPOS   MD_HKLD MD_SPPA      MD_TFLO
0      ft  1000 lbf     psi      bbl/min
1      36       125    2351   24.9047619
2      36       125    2230           25
3      13       125    2815  25.02380952
4      84       125    2977  24.92857143
```

In [127…
```
def analyze_data(data):
    """Analyzes the data by displaying basic statistics."""

    print("Analyzing data...")

    print("\nSummary Statistics:")
    print(data.describe())

    print("\nData Types:")
    print(data.dtypes)

    if 'time' in data.columns:
        print("\n'Time' column is in datetime format:", isinstance(data['time'].dtype, pd.core.dtypes.dtypes.Da
    else:
        print("\n'No Time' column found.")
```

In [197…

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.plot(data['Time'], data['MD_ROP'], marker='o', linestyle='-', color='b')
plt.title("Rate of Penetration (MD_ROP) Over Time")
plt.xlabel("Time")
plt.ylabel("MD_ROP (ft/h)")
plt.grid(True)
plt.tight_layout()

# Save the plot as a PNG file
plt.savefig(r"C:\Users\dell\OneDrive\Documents\Python Scripts\md_rop_trend.png")
plt.show()
```



In [207…

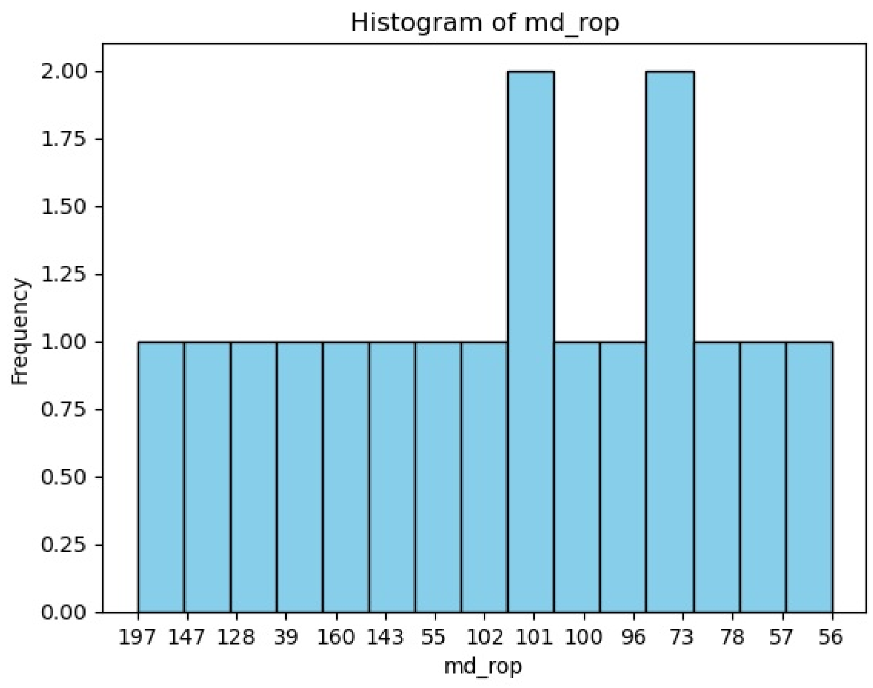
```
def custom_analysis(data, column_name):
    """Custom function to calculate the skewness of a column."""

    if column_name in data.columns:
        if pd.api.types.is_numeric_dtype(data[column_name]):
            skewness = data[column_name].skew()
            print(f"The skewness of '{column_name}' is: {skewness}")
        else:
            print(f"Column '{column_name}' is not numeric. Skewness calculation is not applicable.")
    else:
        print(f"Column '{column_name}' does not exist in the dataset.")
```

In [209…

```
# Main program
if __name__ == "__main__":
    # Example CSV for testing
    file_path = r"C:\Users\dell\OneDrive\Documents\Python Scripts\Drilling_Data_Sample.CSV"

    # Step 1: Load the data
    dataset = load_data(file_path)
    if dataset is not None:
        # Step 2: Clean and wrangle the data
        cleaned_data = clean_data(dataset)

        # Print the cleaned column names to confirm
        print("Column names after cleaning:", cleaned_data.columns)

        # Step 3: Analyze the data
        analyze_data(cleaned_data)

        # Step 4: Visualize data
        show_histogram(cleaned_data, 'md_rop')

        # Step 5: Perform custom analysis
        custom_analysis(cleaned_data, 'md_rop')
```

```
Data successfully loaded!
Cleaning data...
Column names normalized.
Data wrangling complete!
Column names after cleaning: Index(['time', 'md_dmea', 'md_dbtm', 'md_rop', 'md_swob', 'md_tdrpm',
       'md_tdtqr', 'md_bpos', 'md_hkld', 'md_sppa', 'md_tflo'],
      dtype='object')
Analyzing data...

Summary Statistics:
                                time
count                             17
mean   2024-03-24 07:32:43.176470528
min              2024-03-24 06:00:13
25%              2024-03-24 06:50:27
50%              2024-03-24 07:30:27
75%              2024-03-24 08:19:31
max              2024-03-24 09:00:14

Data Types:
time        datetime64[ns]
md_dmea             object
md_dbtm             object
md_rop              object
md_swob             object
md_tdrpm            object
md_tdtqr            object
md_bpos             object
md_hkld             object
md_sppa             object
md_tflo             object
dtype: object

'Time' column is in datetime format: False
Visualizing 'md_rop' as a histogram...
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_12956\1698306479.py:3: UserWarning: Could not infer format, so each e
lement will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected,
please specify a format.
  data['Time'] = pd.to_datetime(data['Time'], errors='coerce')
```



```
Column 'md_rop' is not numeric. Skewness calculation is not applicable.
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js