```
# Step 1: Import the CSV file
file_path = 'gc3030.csv'  # Replace with your actual file path
df = pd.read_csv(file_path)
```

```
# Step 2: Rename columns
df = df.rename(columns={'watts': 'power', 'cad': 'cadence', 'secs': 'tim

# Step 3: Convert 'time' to datetime format
df['time'] = pd.to_datetime(df['time'], unit='s')
```

```
# Step 4: Calculate 30-second rolling averages for power
df['30s_avg_power'] = df['power'].rolling(window=30).mean()
```

```
df['30s_avg_power'] = df['power'].rolling(window=30).mean()
top_25_30s_avg = df.nlargest(25, '30s_avg_power')
print(top_25_30s_avg[['time', '30s_avg_power']])
```

```
                    time  30s_avg_power
7028  1970-01-01 02:08:53     505.366667
7027  1970-01-01 02:08:52     493.933333
7029  1970-01-01 02:08:54     492.000000
11692 1970-01-01 03:45:05     488.600000
7030  1970-01-01 02:08:55     487.833333
7033  1970-01-01 02:08:58     486.700000
11693 1970-01-01 03:45:06     486.166667
7031  1970-01-01 02:08:56     483.666667
7026  1970-01-01 02:08:51     483.133333
11691 1970-01-01 03:45:04     481.600000
7034  1970-01-01 02:08:59     481.066667
11694 1970-01-01 03:45:07     480.733333
7032  1970-01-01 02:08:57     479.233333
7035  1970-01-01 02:09:00     477.500000
11690 1970-01-01 03:45:03     474.233333
7025  1970-01-01 02:08:50     469.466667
11689 1970-01-01 03:45:02     464.566667
11569 1970-01-01 03:43:02     463.800000
7036  1970-01-01 02:09:01     462.566667
```

```
df['10min_intervals'] = pd.cut(df['time'], bins=pd.date_range(start=df['
top_5_intervals = df.groupby('10min_intervals').agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', 'max'),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
).nlargest(5, 'avg_power')

print(top_5_intervals)

ten_minute_intervals = pd.Grouper(key='time', freq='10T')
ten_minute_stats = df.groupby(ten_minute_intervals).agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
)
print("\nStatistics for 10-minute intervals:")
print(ten_minute_stats)
```

```
                                          avg_power  max_power  mi
10min_intervals
(1970-01-01 03:00:01, 1970-01-01 03:10:01]  322.192884      715.0
(1970-01-01 03:30:01, 1970-01-01 03:40:01]  318.859813      730.0
(1970-01-01 02:20:01, 1970-01-01 02:30:01]  287.451376      910.0
(1970-01-01 02:10:01, 1970-01-01 02:20:01]  265.524618      717.0
(1970-01-01 03:40:01, 1970-01-01 03:50:01]  246.037736      736.0

                                          avg_cadence
10min_intervals
(1970-01-01 03:00:01, 1970-01-01 03:10:01]    81.485185
(1970-01-01 03:30:01, 1970-01-01 03:40:01]    82.328704
(1970-01-01 02:20:01, 1970-01-01 02:30:01]    89.490909
(1970-01-01 02:10:01, 1970-01-01 02:20:01]    87.084890
(1970-01-01 03:40:01, 1970-01-01 03:50:01]    72.244395

Statistics for 10-minute intervals:
                     avg_power  max_power  min_power  avg_cadence
time
1970-01-01 00:00:00  170.780718      353.0   1.000000    77.300935
```

```
for i, interval in enumerate(top_5_intervals.index, 1):
    interval_data = df[df['10min_intervals'] == interval]
    interval_data.rename(columns={'power': 'avg_power'}, inplace=True)
    interval_data_stats = interval_data.groupby(pd.cut(interval_data['ti
        avg_power=('avg_power', lambda x: x[x != 0].mean()),
        max_power=('avg_power', 'max'),
        min_power=('avg_power', lambda x: x[x != 0].min()),
        avg_cadence=('cadence', lambda x: x[x != 0].mean())
    )
    print(f"Interval {i}:")
    print(interval_data_stats)
```

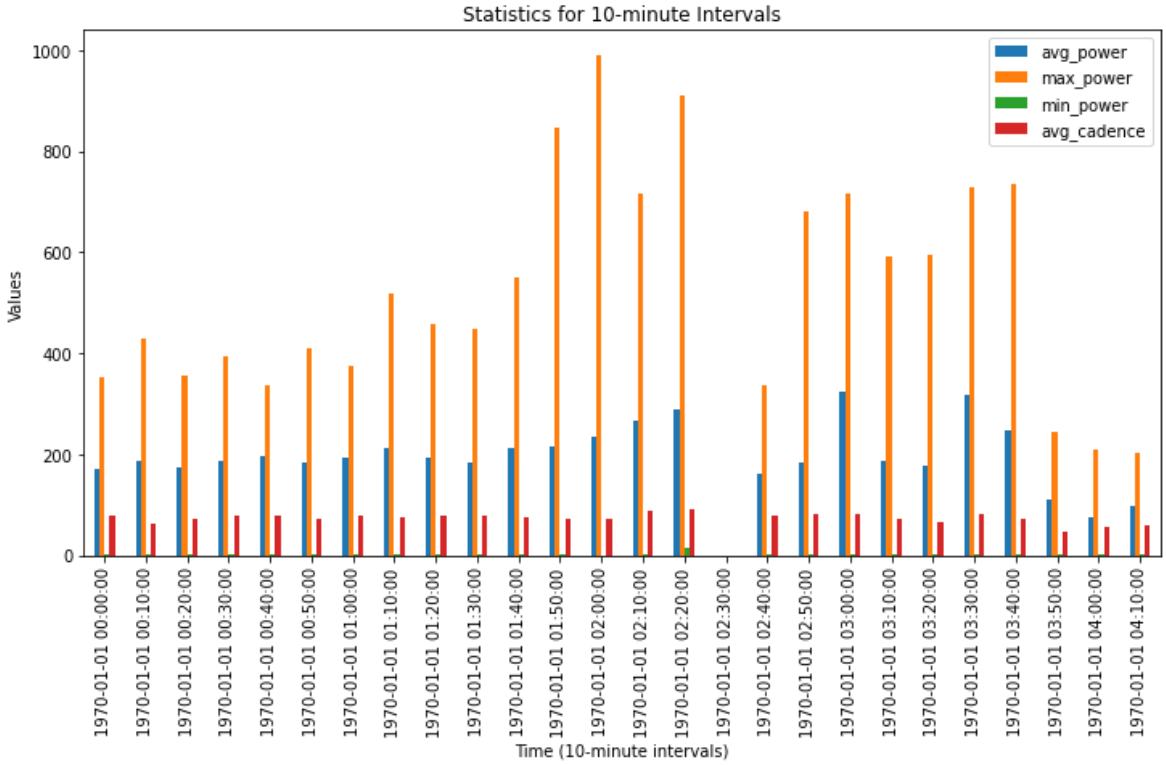
```
Interval 1:
                                                avg_power  max_power  mi
time
(1970-01-01 03:00:02, 1970-01-01 03:00:32]  220.400000      470.0
(1970-01-01 03:00:32, 1970-01-01 03:01:02]  362.933333      639.0
(1970-01-01 03:01:02, 1970-01-01 03:01:32]  231.931034      422.0
(1970-01-01 03:01:32, 1970-01-01 03:02:02]  381.000000      607.0
(1970-01-01 03:02:02, 1970-01-01 03:02:32]  241.727273      511.0
(1970-01-01 03:02:32, 1970-01-01 03:03:02]  357.517241      539.0
(1970-01-01 03:03:02, 1970-01-01 03:03:32]  249.653846      513.0
(1970-01-01 03:03:32, 1970-01-01 03:04:02]  391.466667      603.0
(1970-01-01 03:04:02, 1970-01-01 03:04:32]  213.173913      432.0
(1970-01-01 03:04:32, 1970-01-01 03:05:02]  385.566667      575.0
(1970-01-01 03:05:02, 1970-01-01 03:05:32]  252.615385      440.0
(1970-01-01 03:05:32, 1970-01-01 03:06:02]  377.000000      715.0
(1970-01-01 03:06:02, 1970-01-01 03:06:32]  267.761905      446.0
(1970-01-01 03:06:32, 1970-01-01 03:07:02]  429.400000      700.0
(1970-01-01 03:07:02, 1970-01-01 03:07:32]  238.928571      399.0
(1970-01-01 03:07:32, 1970-01-01 03:08:02]  398.821429      559.0
(1970-01-01 03:08:02, 1970-01-01 03:08:32]  223.192308      392.0
```

```
top_5_segments = ten_minute_stats.nlargest(5, 'avg_power')
print("\nTop 5 Unique 10-minute Segments:")
print(top_5_segments)
```

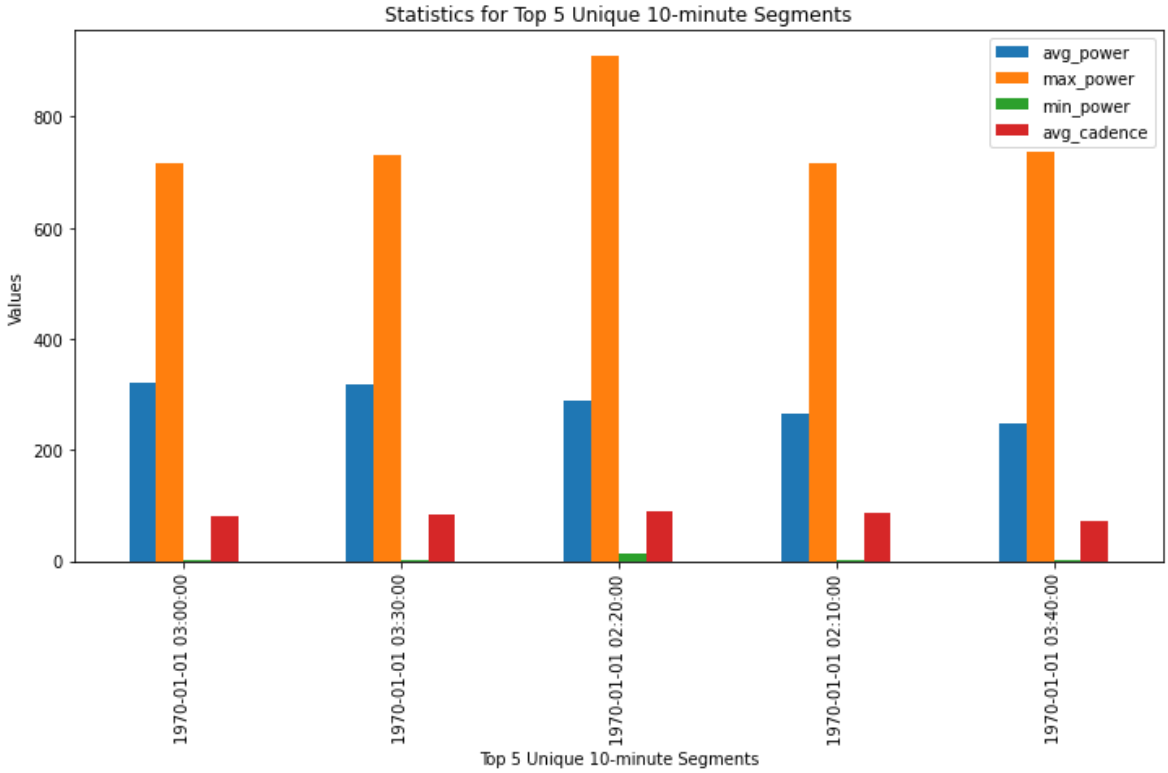
```
Top 5 Unique 10-minute Segments:
                     avg_power  max_power  min_power  avg_cadence
time
1970-01-01 03:00:00  322.136704      715.0        1.0    81.514815
1970-01-01 03:30:00  318.081776      730.0        1.0    82.231481
1970-01-01 02:20:00  287.334552      910.0       13.0    89.492754
1970-01-01 02:10:00  265.237691      717.0        1.0    87.069610
1970-01-01 03:40:00  246.633803      736.0        1.0    72.357143
```

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import timedelta

# Assuming df is the DataFrame from the previous code


# Plot statistics for separate 10-minute intervals
ten_minute_stats.plot(kind='bar', y=['avg_power', 'max_power', 'min_powe
plt.xlabel('Time (10-minute intervals)')
plt.ylabel('Values')
plt.title('Statistics for 10-minute Intervals')
plt.show()

# Plot statistics for the top 5 unique 10-minute segments
top_5_segments.plot(kind='bar', y=['avg_power', 'max_power', 'min_power'
plt.xlabel('Top 5 Unique 10-minute Segments')
plt.ylabel('Values')
plt.title('Statistics for Top 5 Unique 10-minute Segments')
plt.show()

# Plot statistics for 30-second intervals within 10-minute segments
df_30s_stats.plot(kind='bar', y=['avg_power', 'max_power', 'min_power',
plt.xlabel('Time (10-minute intervals)')
plt.ylabel('Values')
plt.title('Statistics for 30-second Intervals within 10-minute Segments'
plt.show()
```

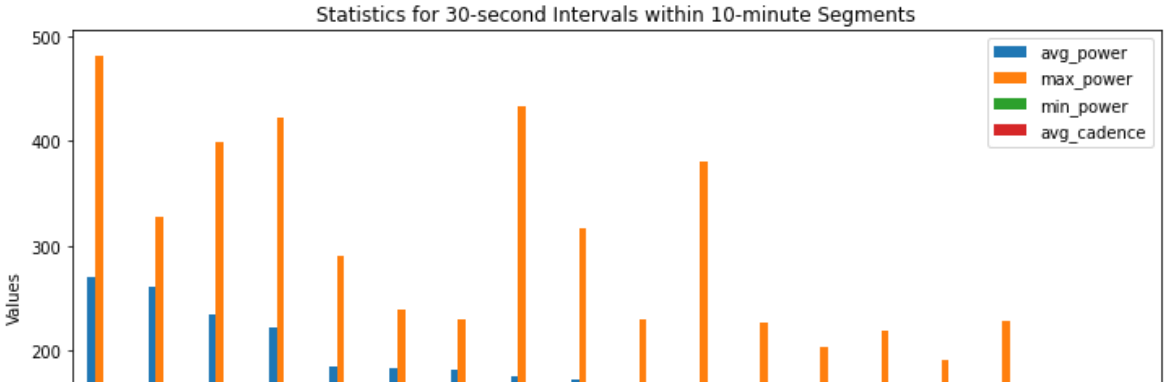
Download



Download



Download


Statistics for 30-second Intervals within 10-minute Segments
Values
500
400
300
200
avg_power
max_power
min_power
avg_cadence

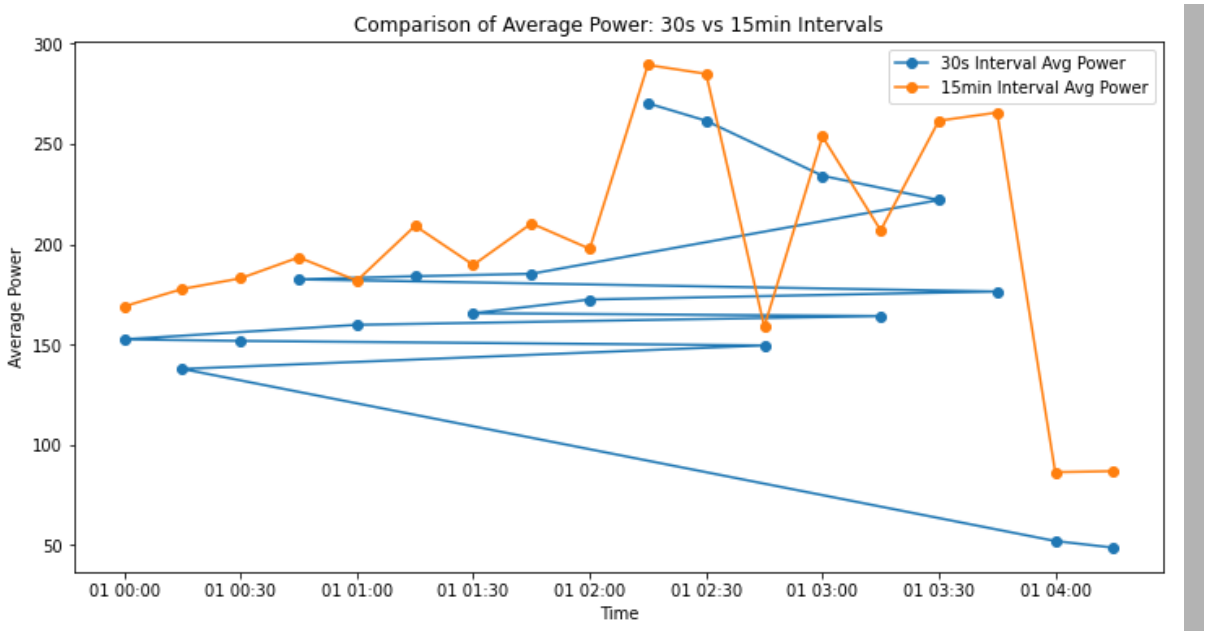
```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import timedelta

# Step 1: Import the CSV file
file_path = 'gc3030.csv'  # Replace with your actual file path
df = pd.read_csv(file_path)

# Step 2: Rename columns
df.rename(columns={'watts': 'power', 'cad': 'cadence', 'secs': 'time'},

# Step 3: Convert 'time' to datetime format
df['time'] = pd.to_datetime(df['time'], unit='s')

# Step 4: Identify the 25 highest 30-second power averages
df['30s_avg_power'] = df['power'].rolling(window=30).mean()
top_25_30s_avg_power = df.nlargest(25, '30s_avg_power')
print("Top 25 30-second Power Averages:")
print(top_25_30s_avg_power[['time', '30s_avg_power']])

# Step 5: Identify and show statistics for separate 15-minute intervals
fifteen_minute_intervals = pd.Grouper(key='time', freq='15T')
df['time_rounded'] = df['time'].dt.round('15T')
fifteen_minute_stats = df.groupby('time_rounded').agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
)

print("\nStatistics for 15-minute intervals:")
print(fifteen_minute_stats)

# Step 6: Identify and show statistics for separate 30-second intervals
thirty_second_intervals = pd.Grouper(key='time', freq='30S')
df['time_rounded_30s'] = df['time'].dt.round('30S')

thirty_second_stats = df.groupby(['time_rounded', 'time_rounded_30s']).a
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
)
print("\nStatistics for 30-second intervals within 15-minute segments:")
print(thirty_second_stats)

# Step 7: Identify and show statistics for the top 5 unique 15-minute se
top_5_segments = fifteen_minute_stats.nlargest(5, 'avg_power')
print("\nTop 5 Unique 15-minute Segments:")
print(top_5_segments)

# Step 8: Split 15-minute intervals into 30-second intervals and show so
df_30s_intervals = df.resample('30S', on='time').mean()
```

```
df_30s_intervals['time_rounded'] = df_30s_intervals.index.round('15T')

df_30s_stats = df_30s_intervals.groupby('time_rounded').agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
).sort_values(by='avg_power', ascending=False)  # Sort by highest averag
print("\nSorted Statistics for 30-second intervals within 15-minute segm
print(df_30s_stats)

# Step 9: Create individual datasets for each top 5 unique 15-minute int
#show the statistics for the 30 second intervals withing the top 5 uniqu
set1 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set2 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set3 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set4 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set5 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').

# Print the first few rows of each dataset
print("\nSet 1:")
print(set1.head())

print("\nSet 2:")
print(set2.head())

print("\nSet 3:")
print(set3.head())

print("\nSet 4:")
print(set4.head())

print("\nSet 5:")
print(set5.head())

# Plotting
plt.figure(figsize=(12, 6))

# Plot 30-second intervals
plt.plot(df_30s_stats.index, df_30s_stats['avg_power'], label='30s Inter

# Plot 15-minute intervals
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['avg_power'],

plt.title('Comparison of Average Power: 30s vs 15min Intervals')
plt.xlabel('Time')
plt.ylabel('Average Power')
plt.legend()
plt.show()

# Plotting
plt.figure(figsize=(12, 8))

# Plot Average Power
```

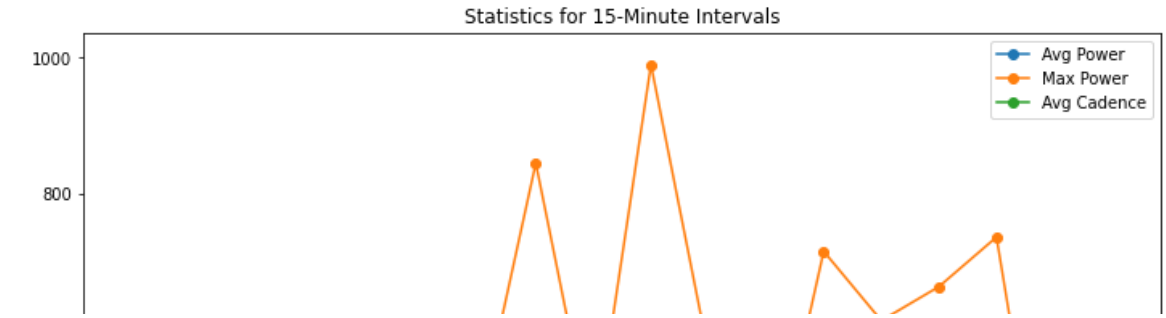
```
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['avg_power'],

# Plot Max Power
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['max_power'],

# Plot Average Cadence
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['avg_cadence']

plt.title('Statistics for 15-Minute Intervals')
plt.xlabel('Time')
plt.ylabel('Values')
plt.legend()
plt.show()
```

```
Top 25 30-second Power Averages:
                     time  30s_avg_power
7028  1970-01-01 02:08:53     505.366667
7027  1970-01-01 02:08:52     493.933333
7029  1970-01-01 02:08:54     492.000000
11692 1970-01-01 03:45:05     488.600000
7030  1970-01-01 02:08:55     487.833333
7033  1970-01-01 02:08:58     486.700000
11693 1970-01-01 03:45:06     486.166667
7031  1970-01-01 02:08:56     483.666667
7026  1970-01-01 02:08:51     483.133333
11691 1970-01-01 03:45:04     481.600000
7034  1970-01-01 02:08:59     481.066667
11694 1970-01-01 03:45:07     480.733333
7032  1970-01-01 02:08:57     479.233333
7035  1970-01-01 02:09:00     477.500000
11690 1970-01-01 03:45:03     474.233333
7025  1970-01-01 02:08:50     469.466667
11689 1970-01-01 03:45:02     464.566667
11569 1970-01-01 03:43:02     463.800000
```

Download



Download




```
# ///10minINT///

# Step 1: Import the CSV file
file_path = 'gc3030.csv'  # Replace with your actual file path
df = pd.read_csv(file_path)
```

```
# Step 2: Rename columns
df = df.rename(columns={'watts': 'power', 'cad': 'cadence', 'secs': 'tim

# Step 3: Convert 'time' to datetime format
df['time'] = pd.to_datetime(df['time'], unit='s')
```

```
# Step 4: Calculate 30-second rolling averages for power
df['30s_avg_power'] = df['power'].rolling(window=30).mean()
```

```
df['30s_avg_power'] = df['power'].rolling(window=30).mean()
top_25_30s_avg = df.nlargest(25, '30s_avg_power')
print(top_25_30s_avg[['time', '30s_avg_power']])
```

```
                    time  30s_avg_power
7028  1970-01-01 02:08:53     505.366667
7027  1970-01-01 02:08:52     493.933333
7029  1970-01-01 02:08:54     492.000000
11692 1970-01-01 03:45:05     488.600000
7030  1970-01-01 02:08:55     487.833333
7033  1970-01-01 02:08:58     486.700000
11693 1970-01-01 03:45:06     486.166667
7031  1970-01-01 02:08:56     483.666667
7026  1970-01-01 02:08:51     483.133333
11691 1970-01-01 03:45:04     481.600000
7034  1970-01-01 02:08:59     481.066667
11694 1970-01-01 03:45:07     480.733333
7032  1970-01-01 02:08:57     479.233333
7035  1970-01-01 02:09:00     477.500000
11690 1970-01-01 03:45:03     474.233333
7025  1970-01-01 02:08:50     469.466667
11689 1970-01-01 03:45:02     464.566667
11569 1970-01-01 03:43:02     463.800000
7036  1970-01-01 02:09:01     462.566667
```

```
df['10min_intervals'] = pd.cut(df['time'], bins=pd.date_range(start=df['
top_5_intervals = df.groupby('10min_intervals').agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', 'max'),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
).nlargest(5, 'avg_power')

print(top_5_intervals)

ten_minute_intervals = pd.Grouper(key='time', freq='10T')
ten_minute_stats = df.groupby(ten_minute_intervals).agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
)
print("\nStatistics for 10-minute intervals:")
print(ten_minute_stats)
```

```
                                                avg_power  max_power  mi
10min_intervals
(1970-01-01 03:00:01, 1970-01-01 03:10:01]  322.192884      715.0
(1970-01-01 03:30:01, 1970-01-01 03:40:01]  318.859813      730.0
(1970-01-01 02:20:01, 1970-01-01 02:30:01]  287.451376      910.0
(1970-01-01 02:10:01, 1970-01-01 02:20:01]  265.524618      717.0
```

```
(1970-01-01 03:40:01, 1970-01-01 03:50:01]  246.037736      736.0

                                            avg_cadence
10min_intervals
(1970-01-01 03:00:01, 1970-01-01 03:10:01]    81.485185
(1970-01-01 03:30:01, 1970-01-01 03:40:01]    82.328704
(1970-01-01 02:20:01, 1970-01-01 02:30:01]    89.490909
(1970-01-01 02:10:01, 1970-01-01 02:20:01]    87.084890
(1970-01-01 03:40:01, 1970-01-01 03:50:01]    72.244395

Statistics for 10-minute intervals:
                     avg_power  max_power  min_power  avg_cadence
time
1970-01-01 00:00:00  170.780718      353.0   1.000000    77.300935
```

```
for i, interval in enumerate(top_5_intervals.index, 1):
    interval_data = df[df['10min_intervals'] == interval]
    interval_data.rename(columns={'power': 'avg_power'}, inplace=True)
    interval_data_stats = interval_data.groupby(pd.cut(interval_data['ti
        avg_power=('avg_power', lambda x: x[x != 0].mean()),
        max_power=('avg_power', 'max'),
        min_power=('avg_power', lambda x: x[x != 0].min()),
        avg_cadence=('cadence', lambda x: x[x != 0].mean())
    )
    print(f"Interval {i}:")
    print(interval_data_stats)
```

```
Interval 1:
                                            avg_power  max_power  mi
time
(1970-01-01 03:00:02, 1970-01-01 03:00:32]  220.400000      470.0
(1970-01-01 03:00:32, 1970-01-01 03:01:02]  362.933333      639.0
(1970-01-01 03:01:02, 1970-01-01 03:01:32]  231.931034      422.0
(1970-01-01 03:01:32, 1970-01-01 03:02:02]  381.000000      607.0
(1970-01-01 03:02:02, 1970-01-01 03:02:32]  241.727273      511.0
(1970-01-01 03:02:32, 1970-01-01 03:03:02]  357.517241      539.0
(1970-01-01 03:03:02, 1970-01-01 03:03:32]  249.653846      513.0
(1970-01-01 03:03:32, 1970-01-01 03:04:02]  391.466667      603.0
(1970-01-01 03:04:02, 1970-01-01 03:04:32]  213.173913      432.0
(1970-01-01 03:04:32, 1970-01-01 03:05:02]  385.566667      575.0
(1970-01-01 03:05:02, 1970-01-01 03:05:32]  252.615385      440.0
(1970-01-01 03:05:32, 1970-01-01 03:06:02]  377.000000      715.0
(1970-01-01 03:06:02, 1970-01-01 03:06:32]  267.761905      446.0
(1970-01-01 03:06:32, 1970-01-01 03:07:02]  429.400000      700.0
(1970-01-01 03:07:02, 1970-01-01 03:07:32]  238.928571      399.0
(1970-01-01 03:07:32, 1970-01-01 03:08:02]  398.821429      559.0
(1970-01-01 03:08:02, 1970-01-01 03:08:32]  223.192308      392.0

<ipython-input-17-55ca3fcb93d2>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
```

```
  interval_data.rename(columns={'power': 'avg_power'}, inplace=True)
<ipython-input-17-55ca3fcb93d2>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
  interval_data.rename(columns={'power': 'avg_power'}, inplace=True)
<ipython-input-17-55ca3fcb93d2>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
  interval_data.rename(columns={'power': 'avg_power'}, inplace=True)
<ipython-input-17-55ca3fcb93d2>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/panda
```

```
top_5_segments = ten_minute_stats.nlargest(5, 'avg_power')
print("\nTop 5 Unique 10-minute Segments:")
print(top_5_segments)
```

```
Top 5 Unique 10-minute Segments:
                      avg_power  max_power  min_power  avg_cadence
time
1970-01-01 03:00:00  322.136704      715.0        1.0    81.514815
1970-01-01 03:30:00  318.081776      730.0        1.0    82.231481
1970-01-01 02:20:00  287.334552      910.0       13.0    89.492754
1970-01-01 02:10:00  265.237691      717.0        1.0    87.069610
1970-01-01 03:40:00  246.633803      736.0        1.0    72.357143
```

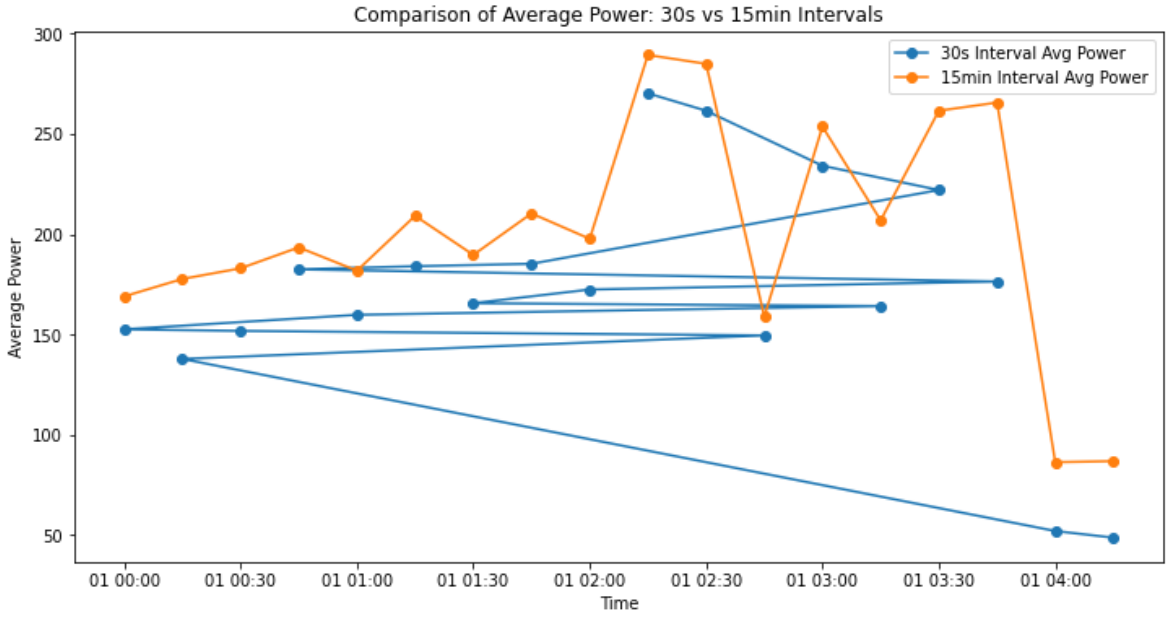
```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import timedelta

# Step 1: Import the CSV file
file_path = 'gc3030.csv'  # Replace with your actual file path
df = pd.read_csv(file_path)

# Step 2: Rename columns
df.rename(columns={'watts': 'power', 'cad': 'cadence', 'secs': 'time'},

# Step 3: Convert 'time' to datetime format
df['time'] = pd.to_datetime(df['time'], unit='s')

# Step 4: Identify the 25 highest 30-second power averages
df['30s_avg_power'] = df['power'].rolling(window=30).mean()
top_25_30s_avg_power = df.nlargest(25, '30s_avg_power')
print("Top 25 30-second Power Averages:")
print(top_25_30s_avg_power[['time', '30s_avg_power']])

# Step 5: Identify and show statistics for separate 15-minute intervals
fifteen_minute_intervals = pd.Grouper(key='time', freq='15T')
df['time_rounded'] = df['time'].dt.round('15T')
fifteen_minute_stats = df.groupby('time_rounded').agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
)

print("\nStatistics for 15-minute intervals:")
print(fifteen_minute_stats)

# Step 6: Identify and show statistics for separate 30-second intervals
thirty_second_intervals = pd.Grouper(key='time', freq='30S')
df['time_rounded_30s'] = df['time'].dt.round('30S')

thirty_second_stats = df.groupby(['time_rounded', 'time_rounded_30s']).a
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
)
print("\nStatistics for 30-second intervals within 15-minute segments:")
print(thirty_second_stats)

# Step 7: Identify and show statistics for the top 5 unique 15-minute se
top_5_segments = fifteen_minute_stats.nlargest(5, 'avg_power')
print("\nTop 5 Unique 15-minute Segments:")
print(top_5_segments)

# Step 8: Split 15-minute intervals into 30-second intervals and show so
df_30s_intervals = df.resample('30S', on='time').mean()
```

```
df_30s_intervals['time_rounded'] = df_30s_intervals.index.round('15T')

df_30s_stats = df_30s_intervals.groupby('time_rounded').agg(
    avg_power=('power', lambda x: x[x != 0].mean()),
    max_power=('power', lambda x: x[x != 0].max()),
    min_power=('power', lambda x: x[x != 0].min()),
    avg_cadence=('cadence', lambda x: x[x != 0].mean())
).sort_values(by='avg_power', ascending=False)  # Sort by highest averag
print("\nSorted Statistics for 30-second intervals within 15-minute segm
print(df_30s_stats)

# Step 9: Create individual datasets for each top 5 unique 15-minute int
#show the statistics for the 30 second intervals withing the top 5 uniqu
set1 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set2 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set3 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set4 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').
set5 = df_30s_stats[df_30s_stats.index.get_level_values('time_rounded').

# Print the first few rows of each dataset
print("\nSet 1:")
print(set1.head())

print("\nSet 2:")
print(set2.head())

print("\nSet 3:")
print(set3.head())

print("\nSet 4:")
print(set4.head())

print("\nSet 5:")
print(set5.head())

# Plotting
plt.figure(figsize=(12, 6))

# Plot 30-second intervals
plt.plot(df_30s_stats.index, df_30s_stats['avg_power'], label='30s Inter

# Plot 15-minute intervals
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['avg_power'],

plt.title('Comparison of Average Power: 30s vs 15min Intervals')
plt.xlabel('Time')
plt.ylabel('Average Power')
plt.legend()
plt.show()

# Plotting
plt.figure(figsize=(12, 8))

# Plot Average Power
```

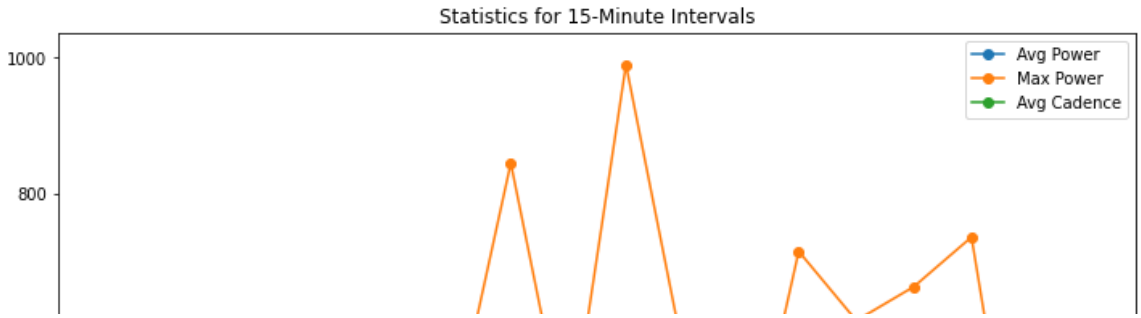
```
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['avg_power'],

# Plot Max Power
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['max_power'],

# Plot Average Cadence
plt.plot(fifteen_minute_stats.index, fifteen_minute_stats['avg_cadence']

plt.title('Statistics for 15-Minute Intervals')
plt.xlabel('Time')
plt.ylabel('Values')
plt.legend()
plt.show()
```

```
Top 25 30-second Power Averages:
                     time  30s_avg_power
7028  1970-01-01 02:08:53     505.366667
7027  1970-01-01 02:08:52     493.933333
7029  1970-01-01 02:08:54     492.000000
11692 1970-01-01 03:45:05     488.600000
7030  1970-01-01 02:08:55     487.833333
7033  1970-01-01 02:08:58     486.700000
11693 1970-01-01 03:45:06     486.166667
7031  1970-01-01 02:08:56     483.666667
7026  1970-01-01 02:08:51     483.133333
11691 1970-01-01 03:45:04     481.600000
7034  1970-01-01 02:08:59     481.066667
11694 1970-01-01 03:45:07     480.733333
7032  1970-01-01 02:08:57     479.233333
7035  1970-01-01 02:09:00     477.500000
11690 1970-01-01 03:45:03     474.233333
7025  1970-01-01 02:08:50     469.466667
11689 1970-01-01 03:45:02     464.566667
11569 1970-01-01 03:43:02     463.800000
```

Download


Comparison of Average Power: 30s vs 15min Intervals
30s Interval Avg Power
15min Interval Avg Power
Average Power
300
250
200
150
100
50
01 00:00
01 00:30
01 01:00
01 01:30
01 02:00
01 02:30
01 03:00
01 03:30
01 04:00
Time

Download


Statistics for 15-Minute Intervals
Avg Power
Max Power
Avg Cadence
1000
800