

```
In [5]: import pandas as pd
import plotly.io as pio
import plotly.graph_objects as go
pio.templates.default = "plotly_white"
import plotly.express as px
data = pd.read_csv(r"C:\Users\nandini sharma\Downloads\Apple-Fitness-Data.csv")
print(data.head())
```

	Date	Time	Step Count	Distance	Energy Burned	\
0	2023-03-21	16:01:23	46	0.02543	14.620	
1	2023-03-21	16:18:37	645	0.40041	14.722	
2	2023-03-21	16:31:38	14	0.00996	14.603	
3	2023-03-21	16:45:37	13	0.00901	14.811	
4	2023-03-21	17:10:30	17	0.00904	15.153	

	Flights Climbed	Walking Double Support Percentage	Walking Speed
0	3	0.304	3.060
1	3	0.309	3.852
2	4	0.278	3.996
3	3	0.278	5.040
4	3	0.281	5.184

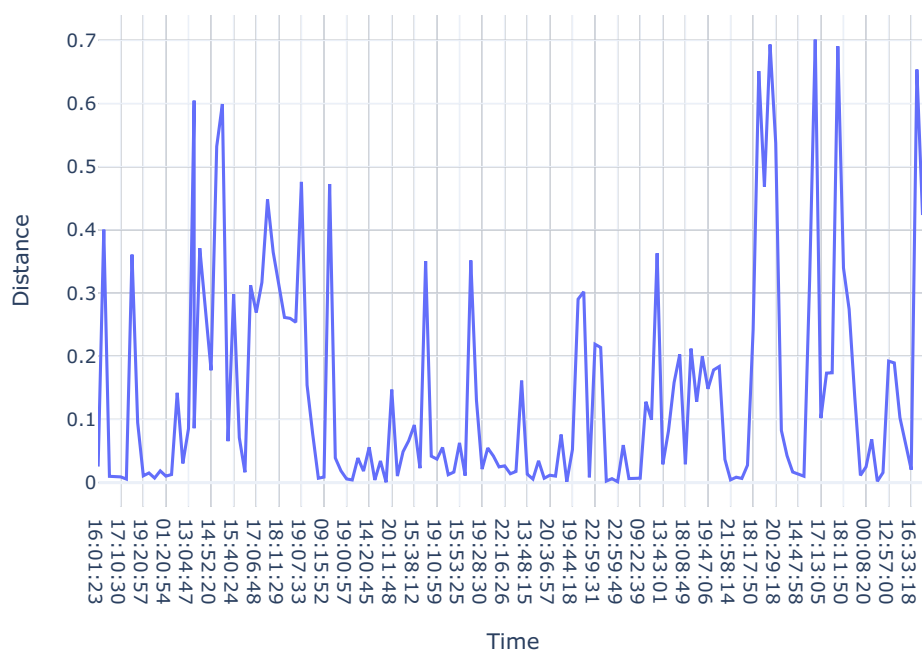
```
In [6]: print(data.isnull().sum())
```

```
Date          0
Time          0
Step Count    0
Distance      0
Energy Burned 0
Flights Climbed 0
Walking Double Support Percentage 0
Walking Speed 0
dtype: int64
```

```
In [7]: fig2 = px.line(data, x="Time",
                      y="Distance",
                      title="Distance Covered Over Time")
fig2.show()
```

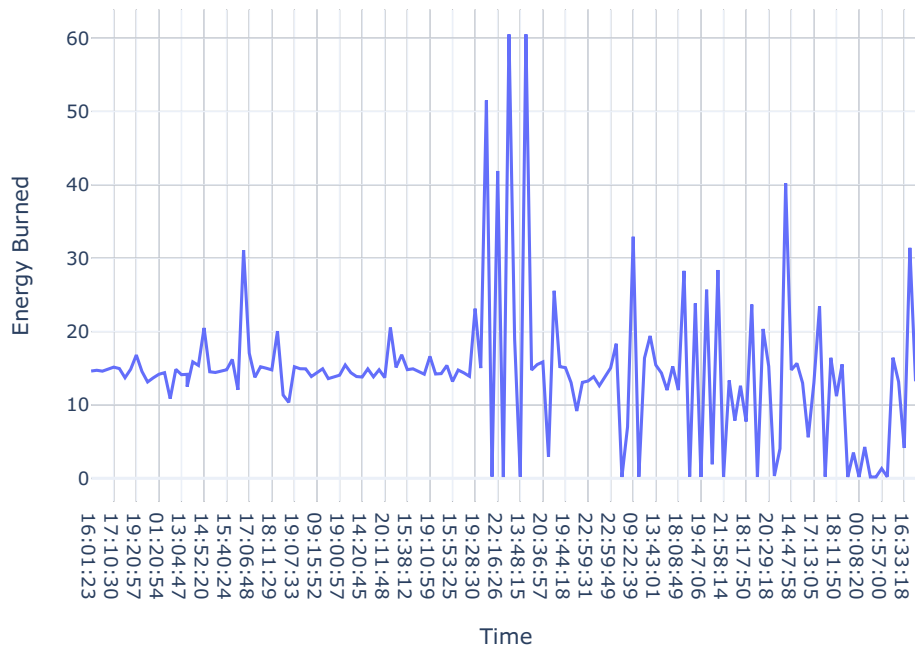


Distance Covered Over Time



```
In [8]: # Energy Burned Over Time
fig3 = px.line(data, x="Time",
               y="Energy Burned",
               title="Energy Burned Over Time")
fig3.show()
```

## Energy Burned Over Time



```
In [9]: # Walking Speed Over Time
fig4 = px.line(data, x="Time",
               y="Walking Speed",
               title="Walking Speed Over Time")
fig4.show()
```

```
In [10]: # Calculate Average Step Count per Day
average_step_count_per_day = data.groupby("Date")["Step Count"].mean().reset_index()

fig5 = px.bar(average_step_count_per_day, x="Date",
               y="Step Count",
               title="Average Step Count per Day")
fig5.update_xaxes(type='category')
fig5.show()
```

```
In [11]: # Calculate Walking Efficiency
data["Walking Efficiency"] = data["Distance"] / data["Step Count"]

fig6 = px.line(data, x="Time",
                y="Walking Efficiency",
                title="Walking Efficiency Over Time")
fig6.show()
```

```
In [12]: # Create Time Intervals
time_intervals = pd.cut(pd.to_datetime(data["Time"]).dt.hour,
                        bins=[0, 12, 18, 24],
                        labels=["Morning", "Afternoon", "Evening"],
                        right=False)

data["Time Interval"] = time_intervals

# Variations in Step Count and Walking Speed by Time Interval
fig7 = px.scatter(data, x="Step Count",
                  y="Walking Speed",
                  color="Time Interval",
```

```
        title="Step Count and Walking Speed Variations by Time Interval",
        trendline='ols')
fig7.show()
```

```
In [13]: # Reshape data for treemap
daily_avg_metrics = data.groupby("Date").mean().reset_index()

daily_avg_metrics_melted = daily_avg_metrics.melt(id_vars=["Date"],
                                                  value_vars=["Step Count", "Distance",
                                                             "Energy Burned", "Flights Climbed",
                                                             "Walking Double Support Percentage",
                                                             "Walking Speed"])

# Treemap of Daily Averages for Different Metrics Over Several Weeks
fig = px.treemap(daily_avg_metrics_melted,
                 path=["variable"],
                 values="value",
                 color="variable",
                 hover_data=["value"],
                 title="Daily Averages for Different Metrics")

fig.show()
```

C:\Users\nandini sharma\AppData\Local\Temp\ipykernel\_20860\1021872881.py:2: FutureWarning:

The default value of `numeric_only` in `DataFrameGroupBy.mean` is deprecated. In a future version, `numeric_only` will default to `False`. Either specify `numeric_only` or select only columns which should be valid for the function.

```
In [14]: # Select metrics excluding Step Count
metrics_to_visualize = ["Distance", "Energy Burned", "Flights Climbed",
                        "Walking Double Support Percentage", "Walking Speed"]

# Reshape data for treemap
daily_avg_metrics_melted = daily_avg_metrics.melt(id_vars=["Date"], value_vars=metrics_to_visualize)

fig = px.treemap(daily_avg_metrics_melted,
                 path=["variable"],
                 values="value",
                 color="variable",
                 hover_data=["value"],
                 title="Daily Averages for Different Metrics (Excluding Step Count)")
fig.show()
```

In [ ]: