Task 1

```
# Import pandas
import pandas as pd

# Define file containing dataset
runkeeper_file = '/content/cardioActivities.csv'

# Create DataFrame with parse_dates and index_col parameters
df_activities = pd.read_csv(runkeeper_file, parse_dates=['Date'], index_col='Date')

# First look at exported data: select sample of 3 random rows
sample_rows = df_activities.sample(3)
print(sample_rows)

# Print DataFrame summary
df_activities.info()
```

```
                                Activity Id     Type Route Name  \
Date
2013-11-16 09:00:24  14e9f43f-6868-41dc-9c87-48b4206330fd  Running      NaN
2016-07-10 16:35:25  85014381-7900-49e2-88f0-42c75e0954c1  Running      NaN
2017-04-06 19:07:46  344ece5a-187e-418e-9287-0e8a11a07643  Running      NaN

                    Distance (km) Duration Average Pace  \
Date
2013-11-16 09:00:24         6.15    32:46       5:20
2016-07-10 16:35:25        19.27  1:49:48       5:42
2017-04-06 19:07:46         8.65    42:24       4:54

                    Average Speed (km/h)  Calories Burned  Climb (m)  \
Date
2013-11-16 09:00:24             11.26      424.000000       21
2016-07-10 16:35:25             10.53     1323.999999      319
2017-04-06 19:07:46             12.25      595.000000       87

                    Average Heart Rate (bpm)  Friend's Tagged  \
Date
2013-11-16 09:00:24                  NaN          NaN
2016-07-10 16:35:25                142.0          NaN
2017-04-06 19:07:46                156.0          NaN

                                Notes           GPX File
Date
2013-11-16 09:00:24               NaN              NaN
2016-07-10 16:35:25  TomTom MySports Watch  2016-07-10-163525.gpx
2017-04-06 19:07:46  TomTom MySports Watch  2017-04-06-190746.gpx
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 508 entries, 2018-11-11 14:05:12 to 2012-08-22 18:53:54
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
```

```
---  ------                  --------------  -----
 0   Activity Id             508 non-null    object
 1   Type                    508 non-null    object
 2   Route Name              1 non-null      object
 3   Distance (km)           508 non-null    float64
 4   Duration                508 non-null    object
 5   Average Pace            508 non-null    object
 6   Average Speed (km/h)    508 non-null    float64
 7   Calories Burned         508 non-null    float64
 8   Climb (m)               508 non-null    int64
 9   Average Heart Rate (bpm)  294 non-null  float64
 10  Friend's Tagged         0 non-null      float64
 11  Notes                   231 non-null    object
 12  GPX File                504 non-null    object
dtypes: float64(5), int64(1), object(7)
memory usage: 55.6+ KB
```

Task 2

```
# Define list of columns to be deleted
cols_to_drop = ['Friend\'s Tagged', 'Route Name', 'GPX File', 'Activity Id', 'Calories Burned', 'Notes']

# Delete unnecessary columns
df_activities.drop(columns=cols_to_drop, inplace=True)

# Count types of training activities
activity_counts = df_activities['Type'].value_counts()
print(activity_counts)

# Rename 'Other' type to 'Unicycling'
df_activities['Type'] = df_activities['Type'].str.replace('Other', 'Unicycling')

# Count missing values for each column
missing_values = df_activities.isnull().sum()
print(missing_values)
```

```
Type
Running    459
Cycling     29
Walking     18
Other        2
Name: count, dtype: int64
Type                      0
Distance (km)             0
Duration                  0
Average Pace              0
Average Speed (km/h)      0
Climb (m)                 0
Average Heart Rate (bpm)  214
dtype: int64
```

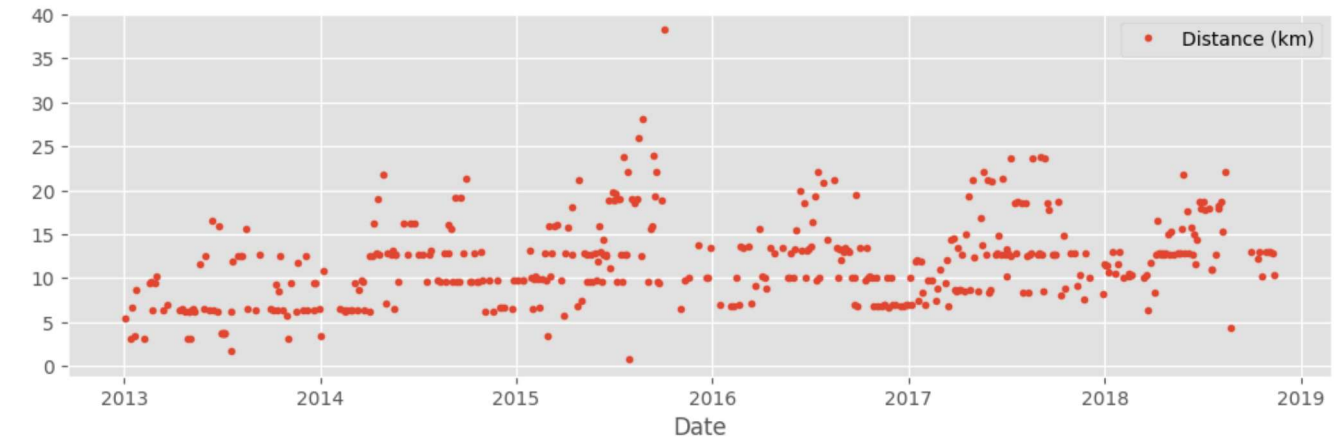
Task 3

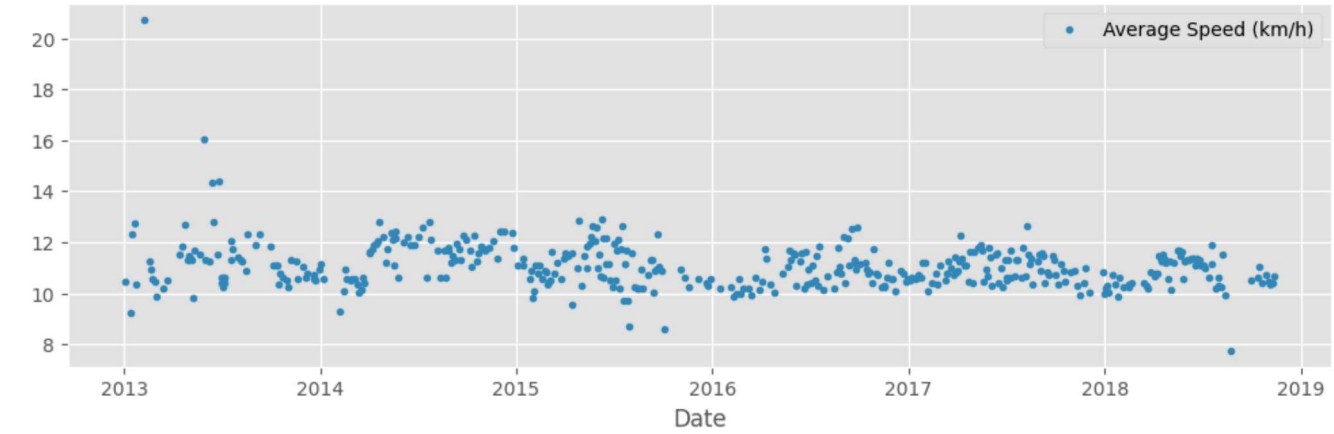
```
# Calculate sample means for heart rate for each training activity type
avg_hr_run = df_activities[df_activities['Type'] == 'Running']['Average Heart Rate (bpm)'].mean()
avg_hr_cycle = df_activities[df_activities['Type'] == 'Cycling']['Average Heart Rate (bpm)'].mean()

# Split whole DataFrame into several, specific for different activities
df_run = df_activities[df_activities['Type'] == 'Running'].copy()
df_walk = df_activities[df_activities['Type'] == 'Walking'].copy()
df_cycle = df_activities[df_activities['Type'] == 'Cycling'].copy()

# Filling missing values with counted means
df_walk['Average Heart Rate (bpm)'].fillna(110, inplace=True)
df_run['Average Heart Rate (bpm)'].fillna(int(avg_hr_run), inplace=True)
df_cycle['Average Heart Rate (bpm)'].fillna(int(avg_hr_cycle), inplace=True)

# Count missing values for each column in running data
missing_values_run = df_run.isnull().sum()
print(missing_values_run)
```

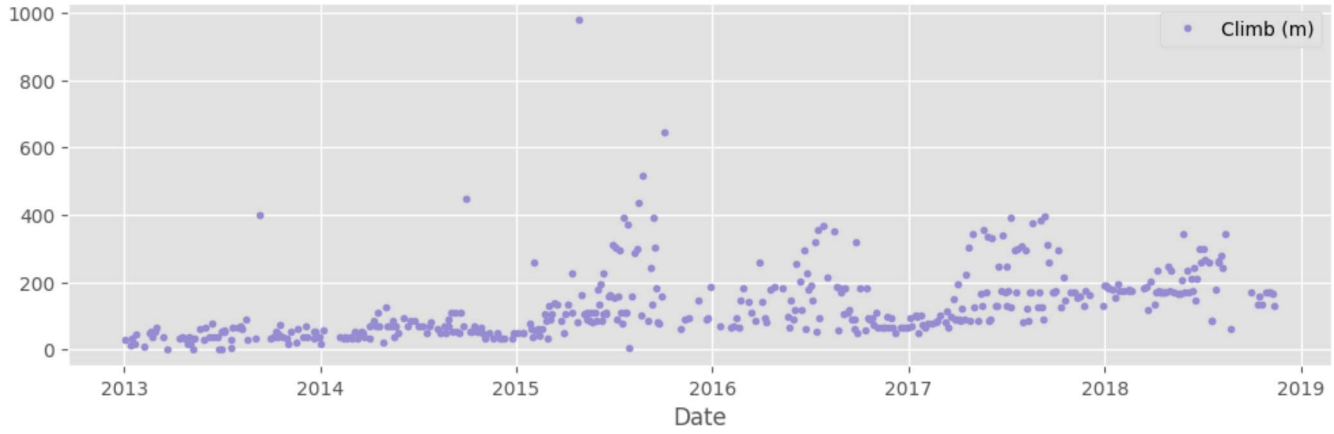
```
Type                    0
Distance (km)           0
Duration                0
Average Pace            0
Average Speed (km/h)    0
Climb (m)               0
Average Heart Rate (bpm)    0
dtype: int64
```

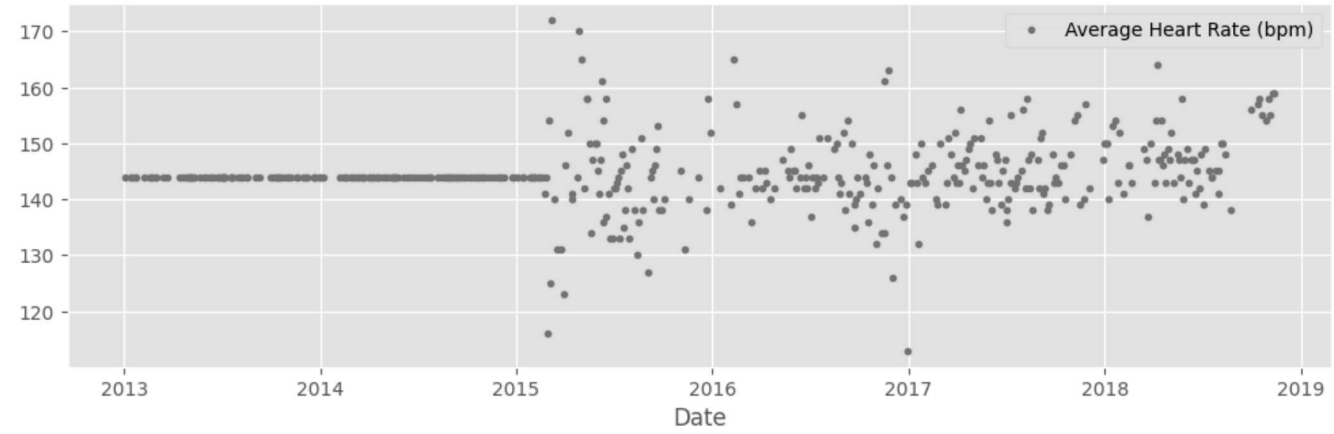
Task 4

```
%matplotlib inline

# Import matplotlib, set style and ignore warning
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
plt.style.use('ggplot')
warnings.filterwarnings(
   action='ignore', module='matplotlib.figure', category=UserWarning,
   message=('This figure includes Axes that are not compatible with tight_layout, so results might be incorrect.')
)

# Ensure the DataFrame is sorted by the date index
df_run.sort_index(inplace=True)

# Prepare data subsetting period from 2013 till 2018
runs_subset_2013_2018 = df_run['2013':'2018']

# Create, plot and customize in one step
runs_subset_2013_2018.plot(subplots=True,
                  sharex=False,
                  figsize=(12,16),
                  linestyle='none',
                  marker='o',
                  markersize=3,
                 )

# Show plot
plt.show()
```


Distance (km)
Date


Average Speed (km/h)
Date


Climb (m)
Date


Average Heart Rate (bpm)
Date

Task 5

```
# Prepare running data for the last 4 years
runs_subset_2015_2018 = df_run['2015':'2018']

# Exclude non-numeric columns and columns with string values representing time
numeric_cols = ['Distance (km)', 'Average Speed (km/h)', 'Climb (m)', 'Average Heart Rate (bpm)']

# Calculate annual statistics
print('How my average run looks in last 4 years:')
annual_stats = runs_subset_2015_2018[numeric_cols].resample('A').mean()
display(annual_stats)

# Calculate weekly statistics
print('Weekly averages of last 4 years:')
weekly_stats = runs_subset_2015_2018[numeric_cols].resample('W').mean()
display(weekly_stats)

# Mean weekly counts
weekly_counts = runs_subset_2015_2018['Distance (km)'].resample('W').count()
weekly_counts_average = weekly_counts.mean()
print('How many trainings per week I had on average:', weekly_counts_average)
```

How my average run looks in last 4 years:

Date	Distance (km)	Average Speed (km/h)	Climb (m)	Average Heart Rate (bpm)
2015-12-31	13.602805	10.998902	160.170732	143.353659
2016-12-31	11.411667	10.837778	133.194444	143.388889
2017-12-31	12.935176	10.959059	169.376471	145.247059
2018-12-31	13.339063	10.777969	191.218750	148.125000

Weekly averages of last 4 years:

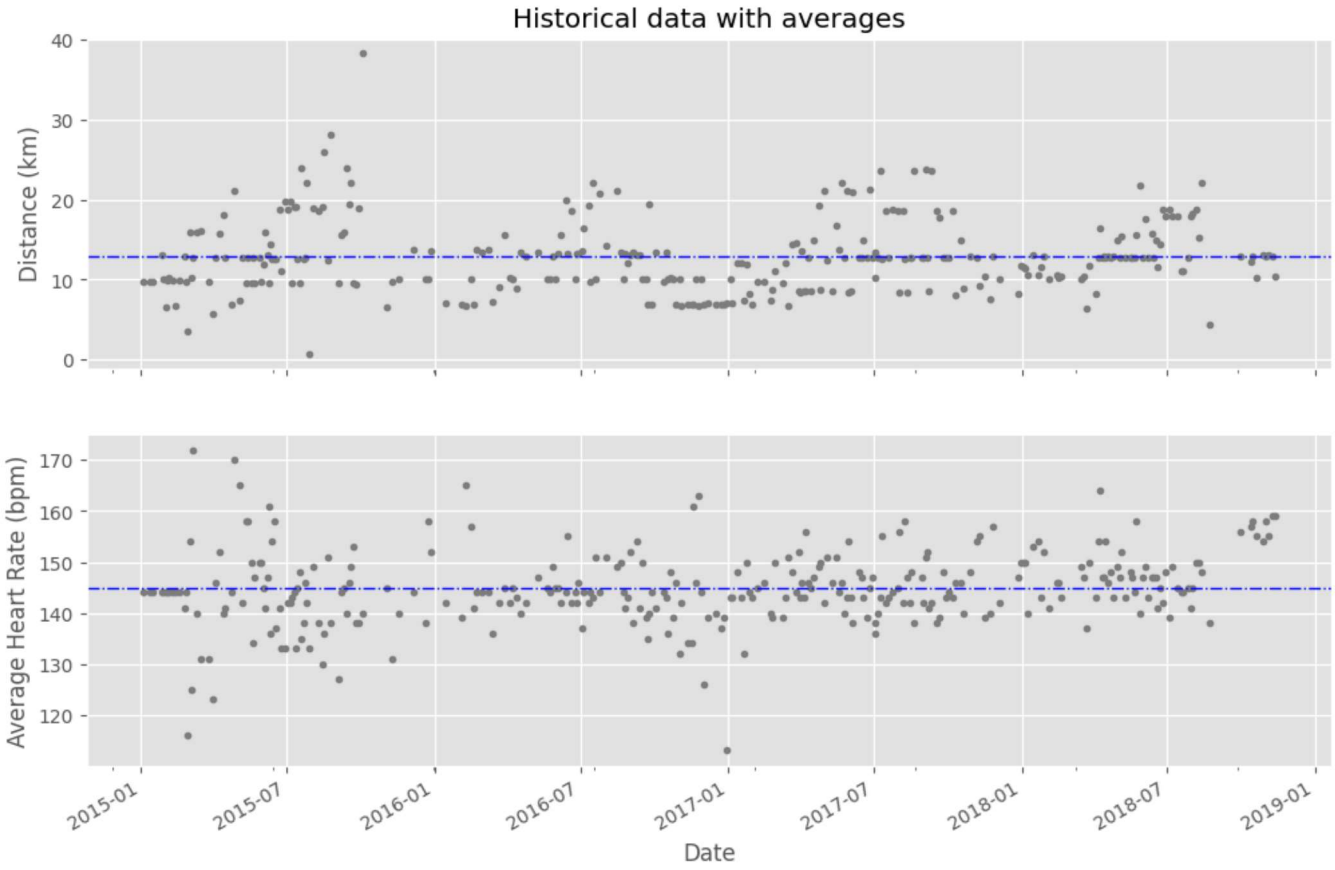
Date	Distance (km)	Average Speed (km/h)	Climb (m)	Average Heart Rate (bpm)
2015-01-04	9.780000	11.120000	51.0	144.0
2015-01-11	NaN	NaN	NaN	NaN
2015-01-18	9.780000	11.230000	51.0	144.0
2015-01-25	NaN	NaN	NaN	NaN
2015-02-01	9.893333	10.423333	58.0	144.0
...	...	...	...	...
2018-10-14	12.620000	10.840000	146.5	157.5
2018-10-21	10.290000	10.410000	133.0	155.0
2018-10-28	13.020000	10.730000	170.0	154.0
2018-11-04	12.995000	10.420000	170.0	156.5
2018-11-11	11.640000	10.535000	149.0	159.0

202 rows × 4 columns

How many trainings per week I had on average: 1.5

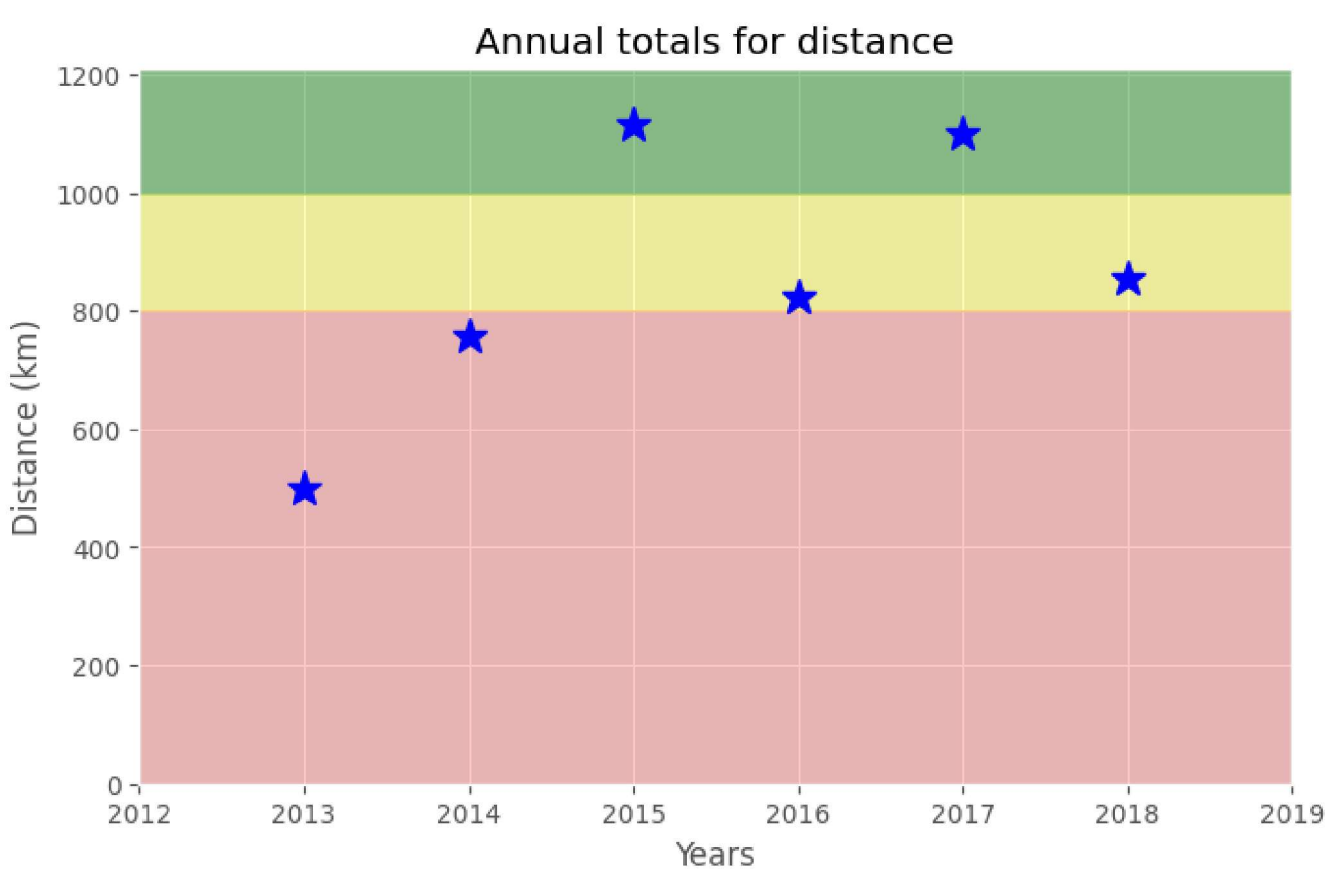
Task 6

```
# Prepare data
runs_subset_2015_2018 = df_run['2015':'2018']  # Correct order: from earlier to later date
runs_distance = runs_subset_2015_2018['Distance (km)']
runs_hr = runs_subset_2015_2018['Average Heart Rate (bpm)']

# Create plot
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True, figsize=(12, 8))

# Plot and customize first subplot
runs_distance.plot(ax=ax1, color='gray', marker='o', linestyle='none', markersize=3)
ax1.set(ylabel='Distance (km)', title='Historical data with averages')
ax1.axhline(runs_distance.mean(), color='blue', linewidth=1, linestyle='-.')

# Plot and customize second subplot
runs_hr.plot(ax=ax2, color='gray', marker='o', linestyle='none', markersize=3)
ax2.set(xlabel='Date', ylabel='Average Heart Rate (bpm)')
ax2.axhline(runs_hr.mean(), color='blue', linewidth=1, linestyle='-.')

# Show plot
plt.show()
```


Historical data with averages
Distance (km)
Average Heart Rate (bpm)
Date
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01

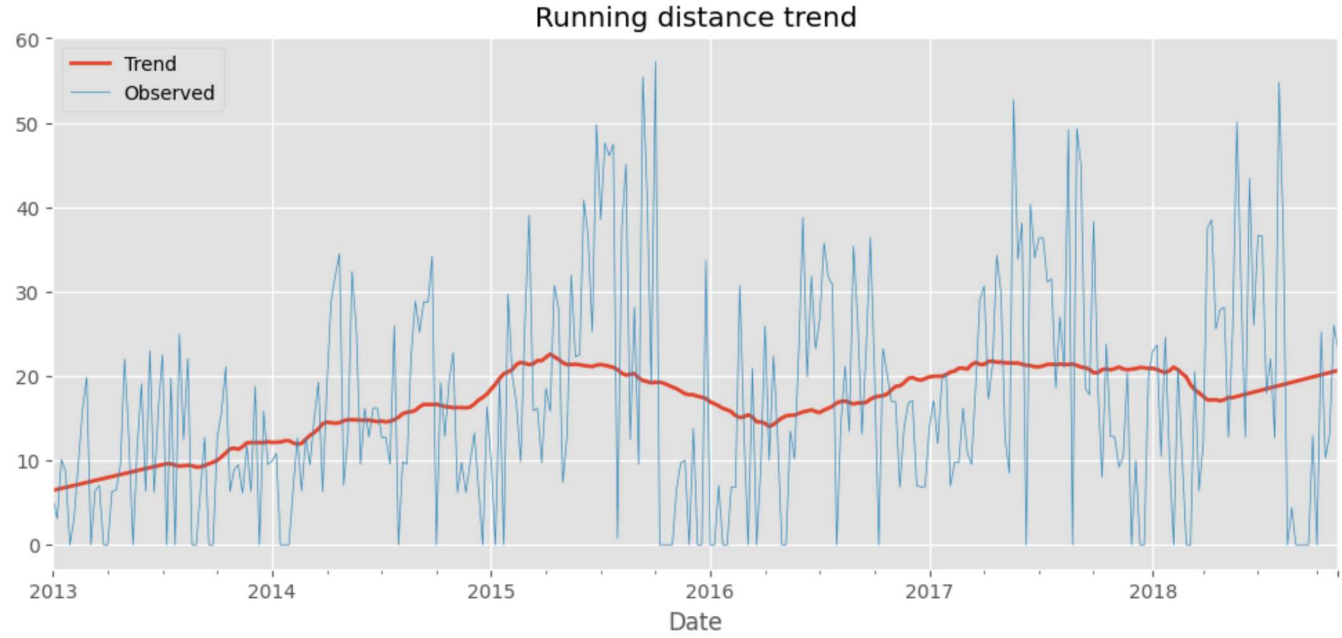
Task 7

```
# Assuming the previous tasks have been completed and df_run is already defined
# If df_run is not defined, load and prepare it as shown previously

# Prepare data: Subset the data from 2013 through 2018 and calculate annual totals
df_run_dist_annual = df_run['2013':'2018'].resample('A')['Distance (km)'].sum()

# Create plot
fig = plt.figure(figsize=(8, 5))

# Plot and customize
ax = df_run_dist_annual.plot(marker='x', markersize=14, linewidth=0, color='blue')
ax.set(ylim=[0, 1210],
       xlim=['2012','2019'],
       ylabel='Distance (km)',
       xlabel='Years',
       title='Annual totals for distance')

ax.axhspan(1000, 1210, color='green', alpha=0.4)
ax.axhspan(800, 1000, color='yellow', alpha=0.3)
ax.axhspan(0, 800, color='red', alpha=0.2)

# Show plot
plt.show()
```



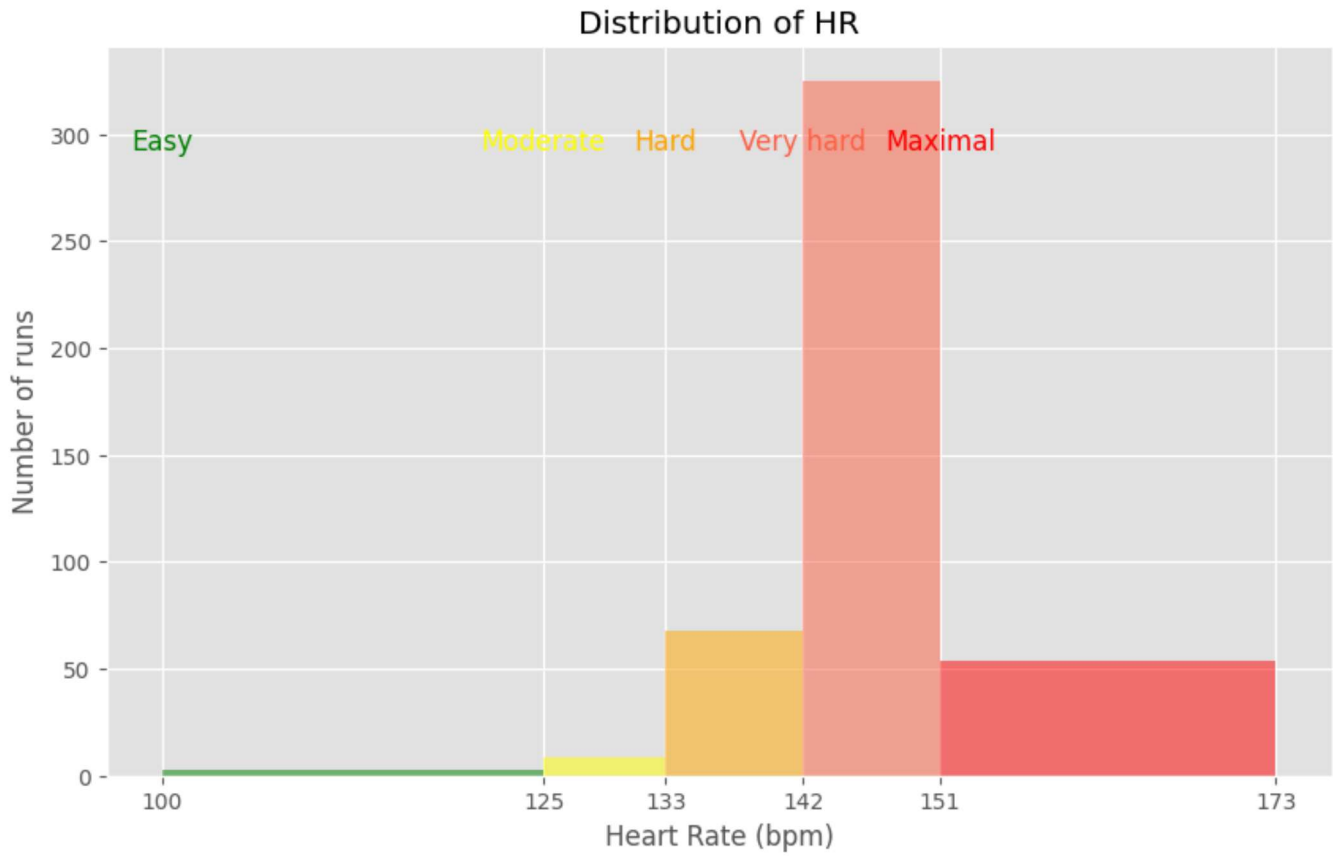
Task 8

```
# Import required library
import statsmodels.api as sm

# Prepare data: Subset the data from 2013 through 2018, select Distance (km), resample weekly, and fill NaN values
df_run_dist_wkly = df_run['2013':'2018'].resample('W')['Distance (km)'].sum().fillna(method='bfill')

# Apply seasonal decomposition
decomposed = sm.tsa.seasonal_decompose(df_run_dist_wkly, extrapolate_trend=1, period=52)

# Create plot
fig = plt.figure(figsize=(12, 5))

# Plot and customize
ax = decomposed.trend.plot(label='Trend', linewidth=2)
decomposed.observed.plot(ax=ax, label='Observed', linewidth=0.5)

ax.legend()
ax.set_title('Running distance trend')

# Show plot
plt.show()
```


Running distance trend
Trend
Observed
60
50
40
30
20
10
0
2013
2014
2015
2016
2017
2018
Date

Task 9

```
# Prepare data
hr_zones = [100, 125, 133, 142, 151, 173]
zone_names = ['Easy', 'Moderate', 'Hard', 'Very hard', 'Maximal']
zone_colors = ['green', 'yellow', 'orange', 'tomato', 'red']

# Assuming df_run_hr_all contains 'Average Heart Rate (bpm)' column from df_run DataFrame
df_run_hr_all = df_run['Average Heart Rate (bpm)']

# Create plot
fig, ax = plt.subplots(figsize=(10, 6))

# Plot and customize
n, bins, patches = ax.hist(df_run_hr_all, bins=hr_zones, alpha=0.5)
for i in range(0, len(patches)):
    patches[i].set_facecolor(zone_colors[i])

ax.set(title='Distribution of HR', ylabel='Number of runs', xlabel='Heart Rate (bpm)')
ax.xaxis.set(ticks=hr_zones)

# Adding zone names as text labels
for zone_name, zone_color, bin in zip(zone_names, zone_colors, bins):
    ax.text(bin, max(n) * 0.9, zone_name, color=zone_color, fontsize=12, ha='center')

# Show plot
plt.show()
```


Distribution of HR
Easy
Moderate
Hard
Very hard
Maximal
Number of runs
0
50
100
150
200
250
300
100
125
133
142
151
173
Heart Rate (bpm)

Task 10

```
# Concatenating three DataFrames
df_run_walk_cycle = pd.concat([df_run, df_walk, df_cycle])

# Define columns for distance, climb, and speed
dist_climb_cols = ['Distance (km)', 'Climb (m)']
speed_col = ['Average Speed (km/h)']

# Calculating total distance and climb in each type of activities
df_totals = df_run_walk_cycle.groupby('Type')[dist_climb_cols].sum()

print('Totals for different training types:')
display(df_totals)

# Calculating summary statistics for each type of activities
df_summary = df_run_walk_cycle.groupby('Type')[dist_climb_cols + speed_col].describe()

# Combine totals with summary
for i in dist_climb_cols:
    df_summary[(i, 'total')] = df_totals[i]

print('Summary statistics for different training types:')
display(df_summary)
```

Totals for different training types:

Type	Distance (km)	Climb (m)
Cycling	680.58	6976
Running	5224.50	57278
Walking	33.45	349

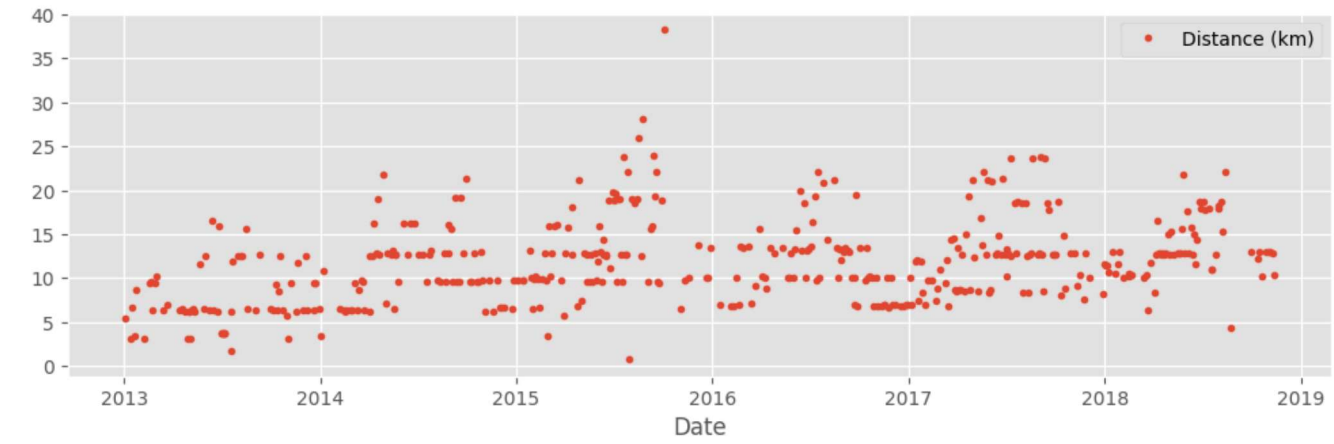
Summary statistics for different training types:

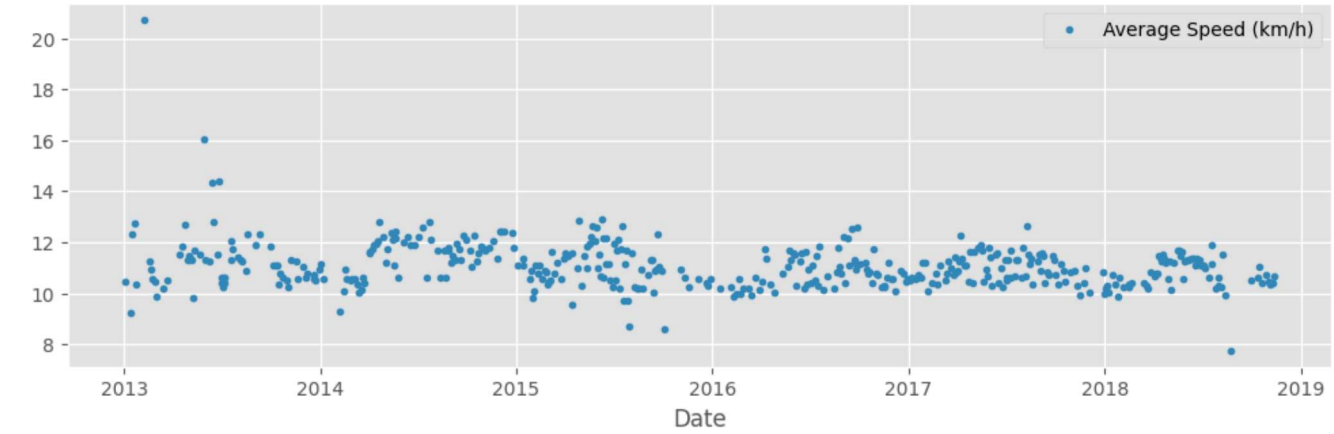
	Distance (km)								Climb (m)		...	Av
	count	mean	std	min	25%	50%	75%	max	count	mean	...	cou
Type												
Cycling	29.0	23.468276	9.451040	11.41	15.530	20.300	29.4000	49.18	29.0	240.551724	...	2
Running	459.0	11.382353	4.937853	0.76	7.415	10.810	13.1900	38.32	459.0	124.788671	...	45
Walking	18.0	1.858333	0.880055	1.22	1.385	1.485	1.7875	4.29	18.0	19.388889	...	1

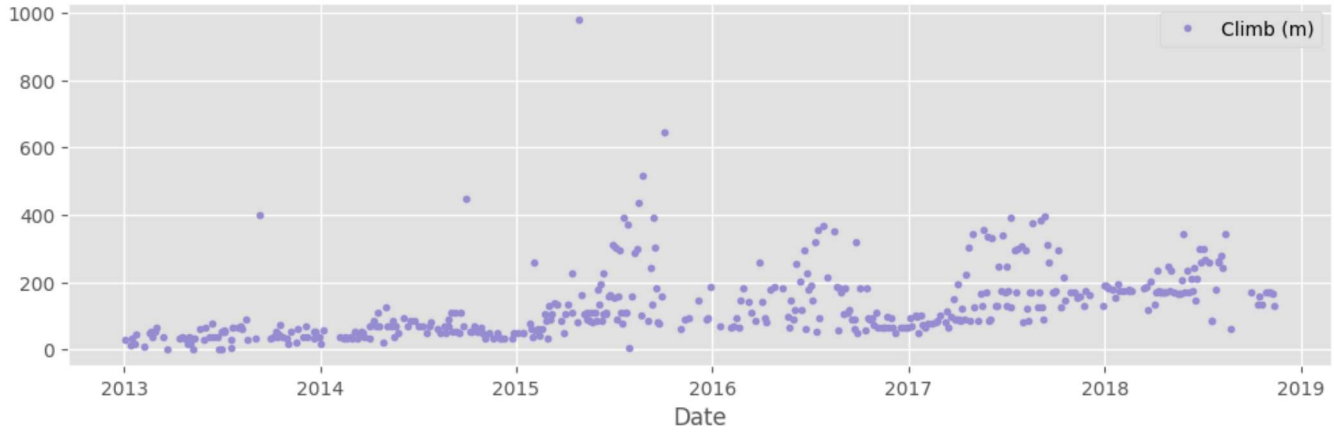
```
# Define average lifetime distance a pair of running shoes can cover (in km)
average_shoes_lifetime_km = 500

# Define Forrest Gump's total run distance (in km)
# Assuming Forrest Gump ran 19,024 miles (as per the movie), converting to km:
forrest_run_distance_km = 19024 * 1.60934

# Count number of shoes for Forrest's run distance
shoes_for_forrest_run = forrest_run_distance_km / average_shoes_lifetime_km

print('Forrest Gump would need {} pairs of shoes!'.format(int(shoes_for_forrest_run)))
```

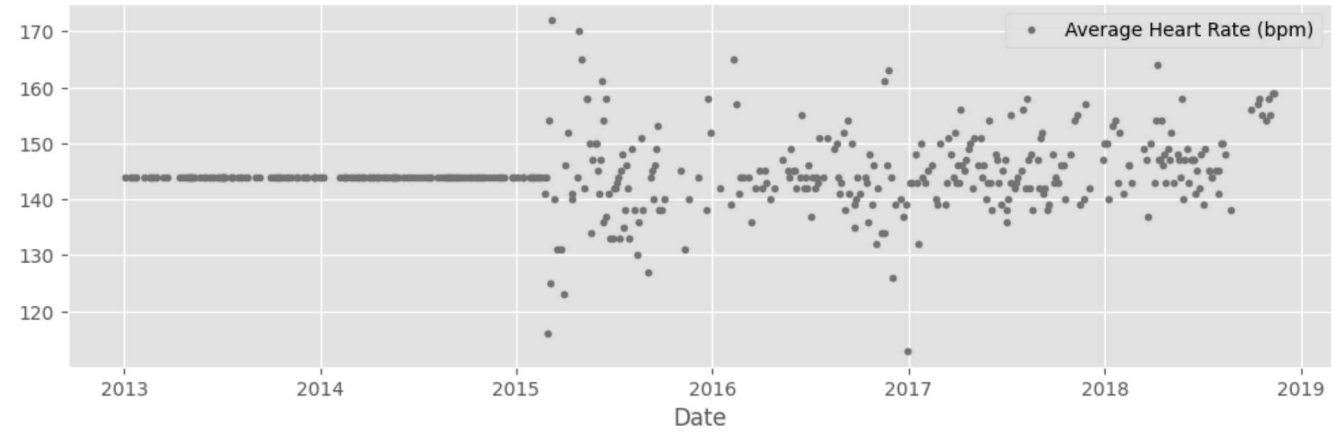
```
Forrest Gump would need 61 pairs of shoes!
```

```
runs_subset_2013_2018.plot(subplots=True, sharex=False, figsize=(12,16), linestyle='none', marker='o', markersize=
plt.show()
```


Distance (km)
Date


Average Speed (km/h)
Date

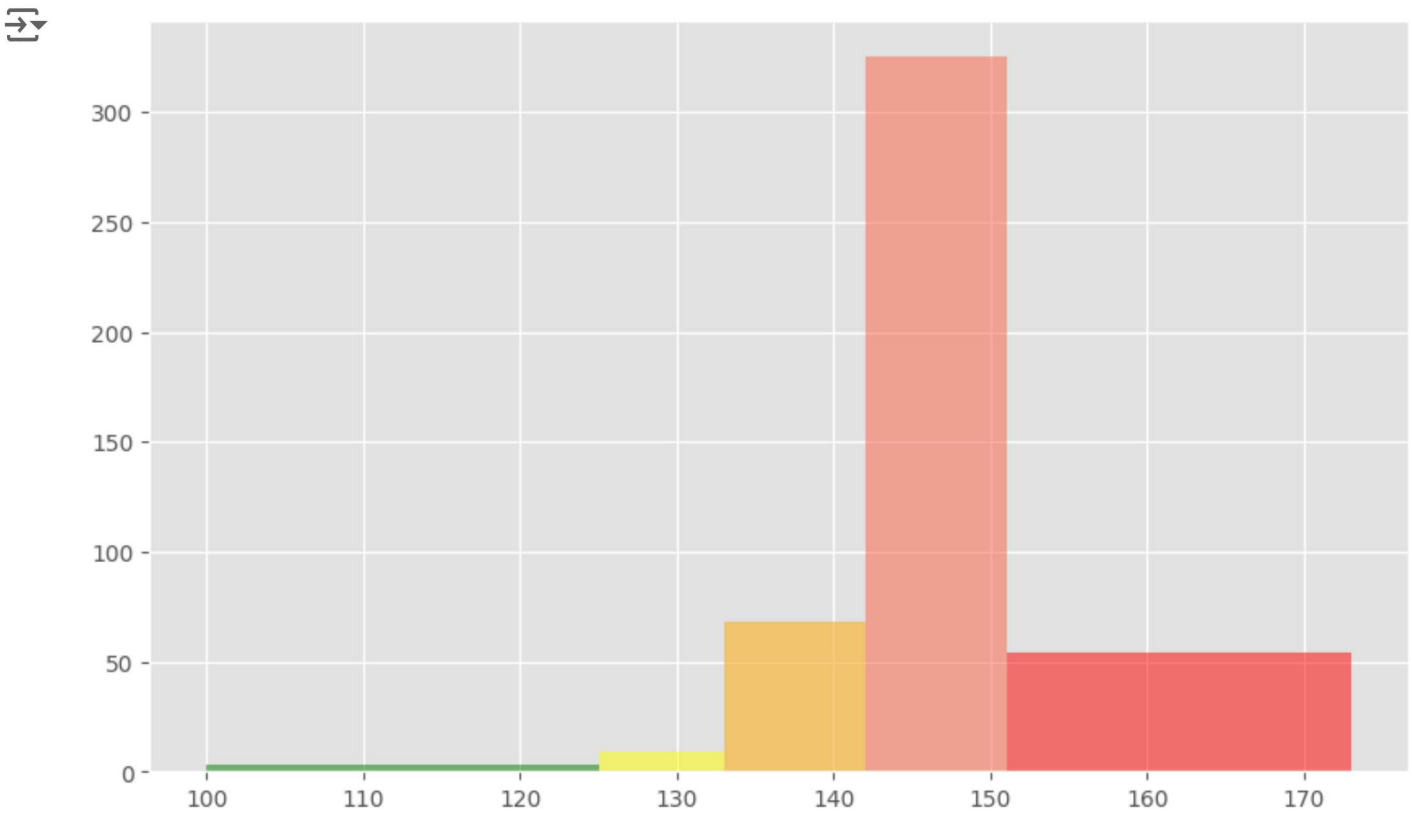

Climb (m)
Date


Average Heart Rate (bpm)
Date

```
annual_stats = runs_subset_2015_2018[numeric_cols].resample('A').mean()
weekly_stats = runs_subset_2015_2018[numeric_cols].resample('W').mean()
print (annual_stats)
print (weekly_stats)
```

```
            Distance (km)  Average Speed (km/h)   Climb (m)  \
Date
2015-12-31     13.602805          10.998902  160.170732
2016-12-31     11.411667          10.837778  133.194444
2017-12-31     12.935176          10.959059  169.376471
2018-12-31     13.339063          10.777969  191.218750

            Average Heart Rate (bpm)
Date
2015-12-31           143.353659
2016-12-31           143.388889
2017-12-31           145.247059
2018-12-31           148.125000
            Distance (km)  Average Speed (km/h)  Climb (m)  \
Date
2015-01-04      9.780000          11.120000      51.0
2015-01-11           NaN                NaN       NaN
2015-01-18      9.780000          11.230000      51.0
2015-01-25           NaN                NaN       NaN
2015-02-01      9.893333          10.423333      58.0
...                  ...                ...       ...
2018-10-14     12.620000          10.840000     146.5
2018-10-21     10.290000          10.410000     133.0
2018-10-28     13.020000          10.730000     170.0
2018-11-04     12.995000          10.420000     170.0
2018-11-11     11.640000          10.535000     149.0

            Average Heart Rate (bpm)
Date
2015-01-04                144.0
2015-01-11                  NaN
2015-01-18                144.0
2015-01-25                  NaN
2015-02-01                144.0
...                         ...
2018-10-14                157.5
2018-10-21                155.0
2018-10-28                154.0
2018-11-04                156.5
2018-11-11                159.0

[202 rows x 4 columns]
```

```
fig, ax = plt.subplots(figsize=(10, 6))
n, bins, patches = ax.hist(df_run_hr_all, bins=hr_zones, alpha=0.5)
for i in range(0, len(patches)):
    patches[i].set_facecolor(zone_colors[i])
plt.show()
```



```
import matplotlib.pyplot as plt

# Ensure the DataFrame is sorted by the date index
df_run.sort_index(inplace=True)

# Prepare data subsetting period from 2013 till 2018
runs_subset_2013_2018 = df_run['2013':'2018']

# Create, plot and customize in one step
runs_subset_2013_2018.plot(subplots=True,
                  sharex=False,
                  figsize=(12,16),
                  linestyle='none',
                  marker='o',
                  markersize=3)
plt.show()
```