Final Report of Traineeship Program 2024

On

“Analyzing RunKeeper Fitness Data ”

MEDTOUREASY

25th June 2024

ACKNOWLEDGMENTS

The traineeship opportunity that I had with MedTourEasy was a great change for learning and understanding the intricacies of the subject of Data Visualizations in Data Analytics; and, for personal as well as professional development. I am very obliged for having a chance to interact with so many professionals who guided me throughout the traineeship project and made it a great learning curve for me.

Firstly, I express my deepest gratitude and special thanks to the Training & Development Team of MedTourEasy who gave me an opportunity to carry out my traineeship at their esteemed organization. Also, I express my thanks to the team for making me understand the details of the Data Analytics profile and training me in the

I would also like to thank the team of MedTourEasy and my colleagues who made the working environment productive and very conducive.

TABLE OF CONTENTS

ABSTRACT

One day, my old running friend and I were chatting about our running styles, training habits, and achievements, when I suddenly realized that I could take an in-depth analytical look at my training.

I have been using a popular GPS fitness tracker called Runkeeper for years and decided it was time to analyze my running data to see how I was doing.

Since 2012, I've been using the Runkeeper app, and it's great. One key feature: its excellent data export. Anyone who has a smartphone can download the app and analyze their data

1.1 About the Company

MedTourEasy, a global healthcare company, provides you the informational resources needed to evaluate your global options. MedTourEasy provides analytical solutions to our partner healthcare providers globally.

1.2 About the Project

Leverage the wealth of data accumulated over years using Runkeeper to assess my running performance.
Extract key trends, identify areas for improvement, and potentially set more targeted running goals.

Analysis of the Project: In this project, the "problem" I aimed to address was the absence of a comprehensive analysis of my running data. While I had been diligently tracking runs with Runkeeper, the data remained largely unanalysed.

I lacked insights into key aspects like:

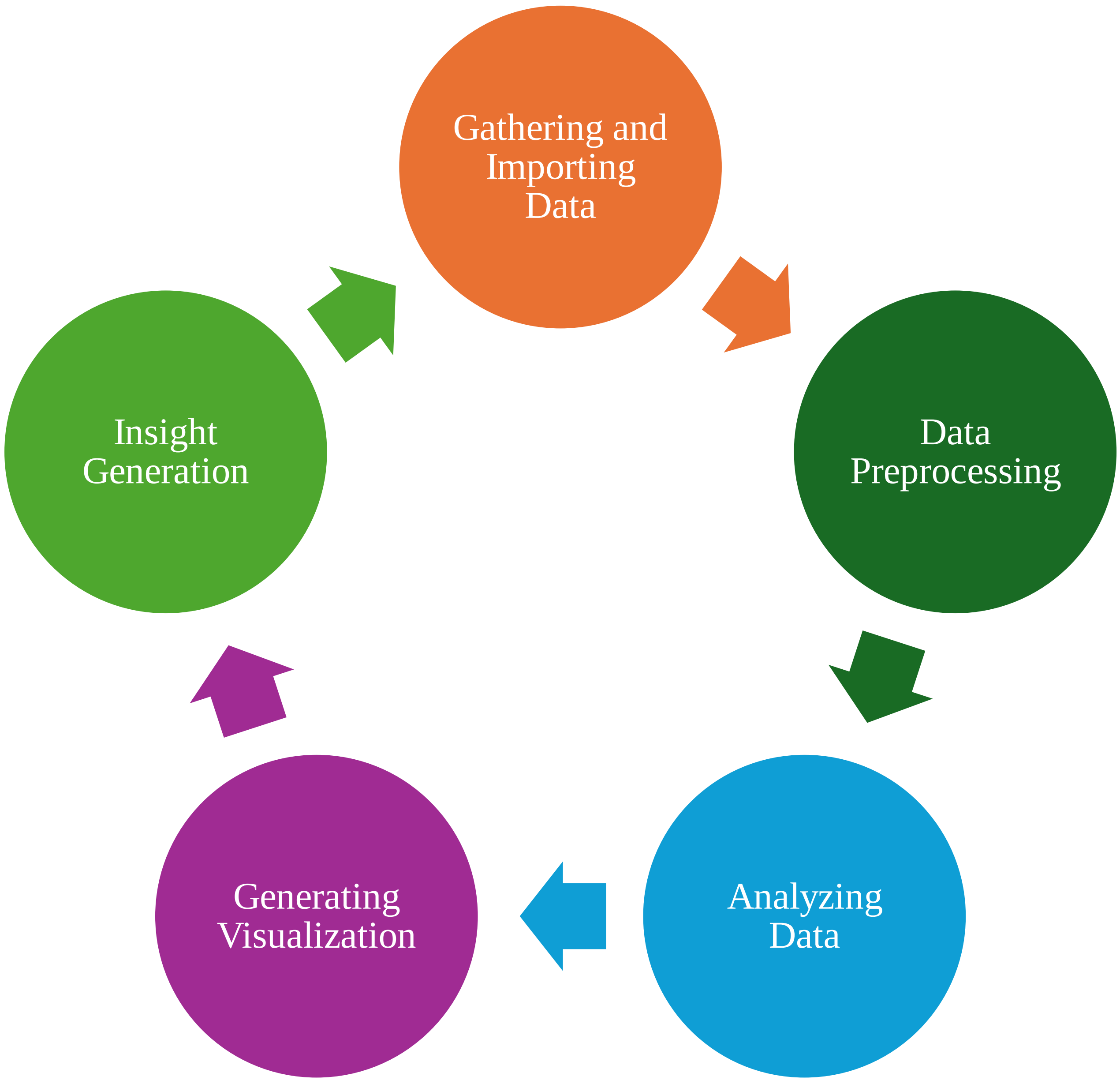
- Overall training volume trends (distance covered over time)
- Pace and speed variations and their connection to factors like distance or elevation
- Consistency in training frequency and how it might impact performance
- Identification of personal bests or longest runs

This lack of deeper understanding made it difficult to:

- Set targeted running goals based on my current performance level
- Identify areas for improvement in my training regimen

2. Methodology

2.1 Flow Of The Project:



2.2 Language and Platform Used:

2.3.1 Language: Python

Python is a powerful and versatile **high-level programming language**, known for its:

- **Readability:** Python's syntax is clear and concise, resembling natural language, making it easier to learn and write compared to some other languages.
- **Versatility:** Python is a **general-purpose language**, meaning it can be used for various tasks like web development, data science, machine learning, scripting, and automation.

While not the heart of your running data analysis project, Python plays a supporting role in data analysis workflows:

- Python's libraries like Pandas empower data cleaning and manipulation before feeding it into SQL databases.
- It shines in data visualization with Matplotlib and Seaborn, creating clear and informative charts from SQL query results.
- For projects demanding more than SQL's capabilities, Python offers Scikit-learn and other libraries for machine learning or advanced statistical analysis.

2.3.2 Tools: Google Collab

Google Collab offers a convenient, accessible, and powerful platform for data analysis projects, making it a popular choice for students, researchers, and data enthusiasts.

- **Free and Cloud-Based:** No software installation is needed. Access Collab from any device with a web browser and internet connection, making it highly portable.

- **Pre-Installed Libraries:** Essential data science libraries like Pandas, NumPy, Matplotlib, and Scikit-learn are pre-installed, saving you setup time.

- **Easy Data Visualization:** Create charts and graphs directly within your notebooks to visualize your data analysis results.

3. Implementation

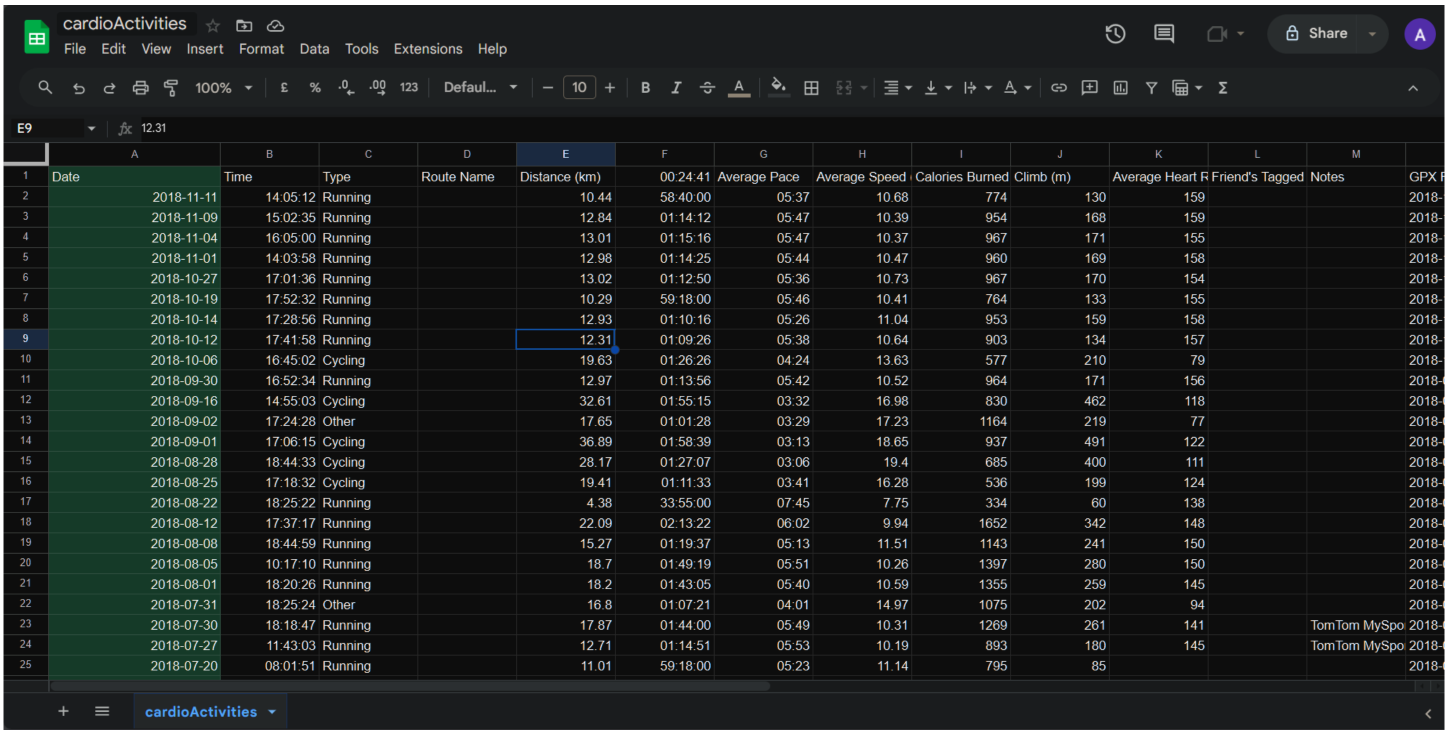
3.1 Gathering Requirements and Defining Problem Statement

This is the first step wherein the requirements are collected from the clients to understand the deliverables and goals to be achieved after which a problem statement is defined which must be adhered to while development of the project.

3.2 Data Collection and Importing

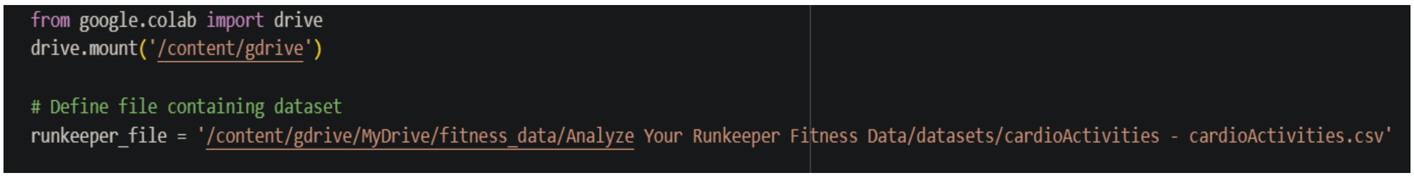
Data collection is a systematic approach for gathering and measuring information from a variety of sources to obtain a complete and accurate picture of an interest area. It helps an individual or organization to address specific questions, determine outcomes and forecast future probabilities and patterns.

- Data for this Project was collected from Runkeeper.
- It's a personal data, collected from an app.
- Exported data is in form of .csv file.



	A	B	C	D	E	F	G	H	I	J	K	L	M	
1	Date	Time	Type	Route Name	Distance (km)	00:24:41	Average Pace	Average Speed	Calories Burned	Climb (m)	Average Heart R	Friend's Tagged	Notes	GPX F
2	2018-11-11	14:05:12	Running		10.44	58:40:00	05:37	10.68	774	130	159			2018-
3	2018-11-09	15:02:35	Running		12.84	01:14:12	05:47	10.39	954	168	159			2018-
4	2018-11-04	16:05:00	Running		13.01	01:15:16	05:47	10.37	967	171	155			2018-
5	2018-11-01	14:03:58	Running		12.98	01:14:25	05:44	10.47	960	169	158			2018-
6	2018-10-27	17:01:36	Running		13.02	01:12:50	05:36	10.73	967	170	154			2018-
7	2018-10-19	17:52:32	Running		10.29	59:18:00	05:46	10.41	764	133	155			2018-
8	2018-10-14	17:28:56	Running		12.93	01:10:16	05:26	11.04	953	159	158			2018-
9	2018-10-12	17:41:58	Running		12.31	01:09:26	05:38	10.64	903	134	157			2018-
10	2018-10-06	16:45:02	Cycling		19.63	01:26:26	04:24	13.63	577	210	79			2018-
11	2018-09-30	16:52:34	Running		12.97	01:13:56	05:42	10.52	964	171	156			2018-
12	2018-09-16	14:55:03	Cycling		32.61	01:55:15	03:32	16.98	830	462	118			2018-
13	2018-09-02	17:24:28	Other		17.65	01:01:28	03:29	17.23	1164	219	77			2018-
14	2018-09-01	17:06:15	Cycling		36.89	01:58:39	03:13	18.65	937	491	122			2018-
15	2018-08-28	18:44:33	Cycling		28.17	01:27:07	03:06	19.4	685	400	111			2018-
16	2018-08-25	17:18:32	Cycling		19.41	01:11:33	03:41	16.28	536	199	124			2018-
17	2018-08-22	18:25:22	Running		4.38	33:55:00	07:45	7.75	334	60	138			2018-
18	2018-08-12	17:37:17	Running		22.09	02:13:22	06:02	9.94	1652	342	148			2018-
19	2018-08-08	18:44:59	Running		15.27	01:19:37	05:13	11.51	1143	241	150			2018-
20	2018-08-05	10:17:10	Running		18.7	01:49:19	05:51	10.26	1397	280	150			2018-
21	2018-08-01	18:20:26	Running		18.2	01:43:05	05:40	10.59	1355	259	145			2018-
22	2018-07-31	18:25:24	Other		16.8	01:07:21	04:01	14.97	1075	202	94			2018-
23	2018-07-30	18:18:47	Running		17.87	01:44:00	05:49	10.31	1269	261	141		TomTom MySpo	2018-
24	2018-07-27	11:43:03	Running		12.71	01:14:51	05:53	10.19	893	180	145		TomTom MySpo	2018-
25	2018-07-20	08:01:51	Running		11.01	59:18:00	05:23	11.14	795	85				2018-

Data importing is referred to as uploading the required data into the coding environment from internal sources (computer) or external sources (online websites and data repositories). This data can then be manipulated, aggregated, filtered according to the requirements and needs of the project.

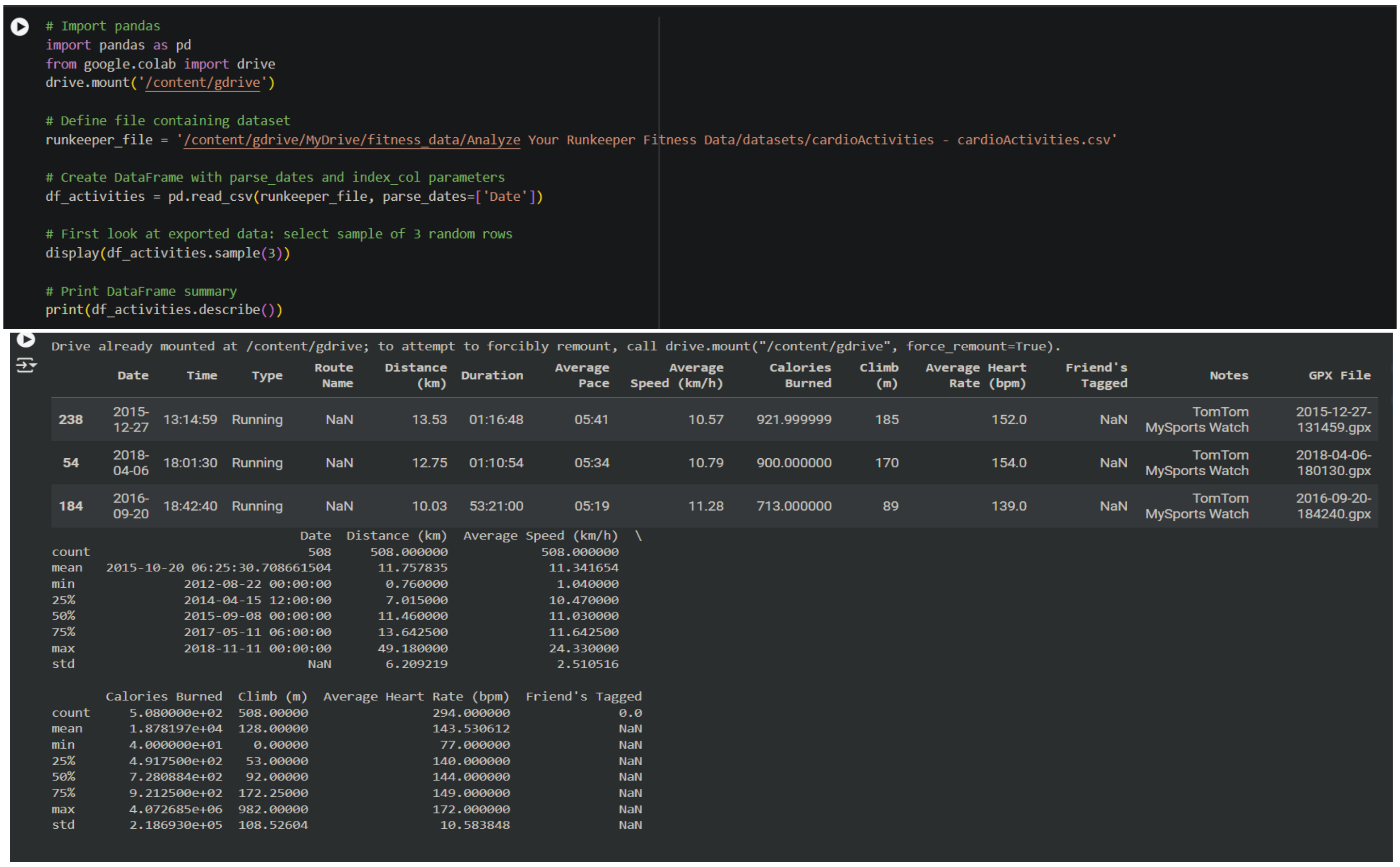


```
from google.colab import drive
drive.mount('/content/gdrive')

# Define file containing dataset
runkeeper_file = '/content/gdrive/MyDrive/fitness_data/Analyze Your Runkeeper Fitness Data/datasets/cardioActivities - cardioActivities.csv'
```

Checking data after importing in important, to see if there are no issues with the data:

This step ensures:

- **Data is correct with appropriate data types**
- **Columns and rows are correctly named.**
- **Makes visualizing data easier**



```
# Import pandas
import pandas as pd
from google.colab import drive
drive.mount('/content/gdrive')

# Define file containing dataset
runkeeper_file = '/content/gdrive/MyDrive/fitness_data/Analyze Your Runkeeper Fitness Data/datasets/cardioActivities - cardioActivities.csv'

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

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

# Print DataFrame summary
print(df_activities.describe())
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

	Date	Time	Type	Route Name	Distance (km)	Duration	Average Pace	Average Speed (km/h)	Calories Burned	Climb (m)	Average Heart Rate (bpm)	Friend's Tagged	Notes	GPX File
238	2015-12-27	13:14:59	Running	NaN	13.53	01:16:48	05:41	10.57	921.999999	185	152.0	NaN	TomTom MySports Watch	2015-12-27-131459.gpx
54	2018-04-06	18:01:30	Running	NaN	12.75	01:10:54	05:34	10.79	900.000000	170	154.0	NaN	TomTom MySports Watch	2018-04-06-180130.gpx
184	2016-09-20	18:42:40	Running	NaN	10.03	53:21:00	05:19	11.28	713.000000	89	139.0	NaN	TomTom MySports Watch	2016-09-20-184240.gpx

```
                               Date  Distance (km)  Average Speed (km/h)  \
count                           508     508.000000            508.000000
mean  2015-10-20 06:25:30.708661504      11.757835             11.341654
min             2012-08-22 00:00:00       0.760000              1.040000
25%             2014-04-15 12:00:00       7.015000             10.470000
50%             2015-09-08 00:00:00      11.460000             11.030000
75%             2017-05-11 06:00:00      13.642500             11.642500
max             2018-11-11 00:00:00      49.180000             24.330000
std                             NaN       6.209219              2.510516

       Calories Burned  Climb (m)  Average Heart Rate (bpm)  Friend's Tagged
count     5.080000e+02  508.00000                294.000000              0.0
mean      1.878197e+04  128.00000                143.530612              NaN
min       4.000000e+01    0.00000                 77.000000              NaN
25%       4.917500e+02   53.00000                140.000000              NaN
50%       7.280884e+02   92.00000                144.000000              NaN
75%       9.212500e+02  172.25000                149.000000              NaN
max       4.072685e+06  982.00000                172.000000              NaN
std       2.186930e+05  108.52604                 10.583848              NaN
```

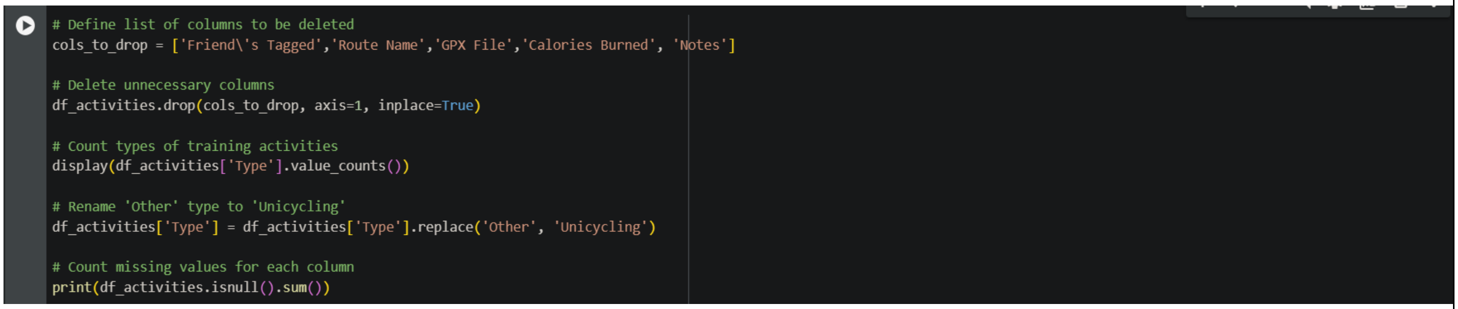
3.3 Data Preprocessing

“Quality data beats fancy algorithms”

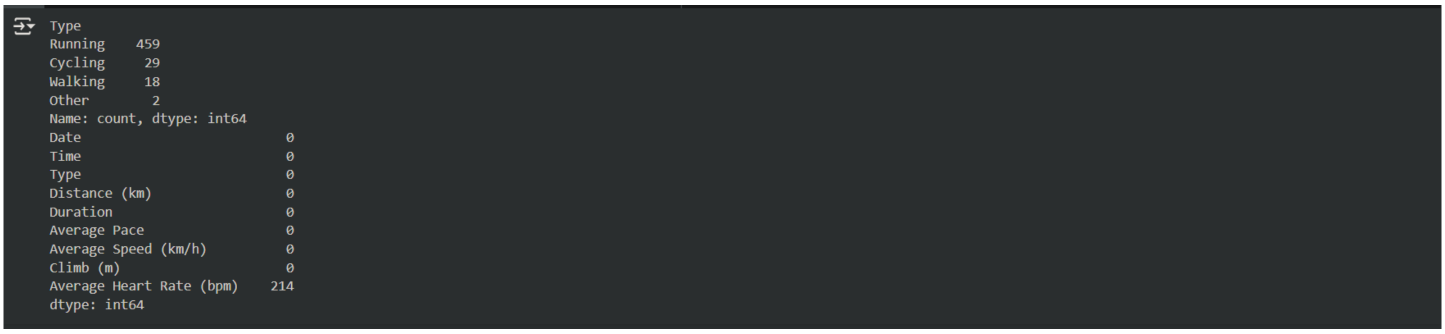
Data is the most imperative aspect of Analytics and Machine Learning. Everywhere in computing or business, data is required. But many a times, the data may be incomplete, inconsistent or may contain missing values when it comes to the real world. If the data is corrupted then the process may be impeded or inaccurate results may be provided. Hence, Data cleaning is considered a foundational element of the basic data science.

Data Cleaning means the process by which the incorrect, incomplete, inaccurate, irrelevant or missing part of the data is identified and then modified, replaced or deleted as needed.

- Remove columns not useful for our analysis.
- Replace the "Other" activity type to "Unicycling" because that was always the "Other" activity.
- Count missing values.



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

# Delete unnecessary columns
df_activities.drop(cols_to_drop, axis=1, inplace=True)

# Count types of training activities
display(df_activities['Type'].value_counts())

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

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



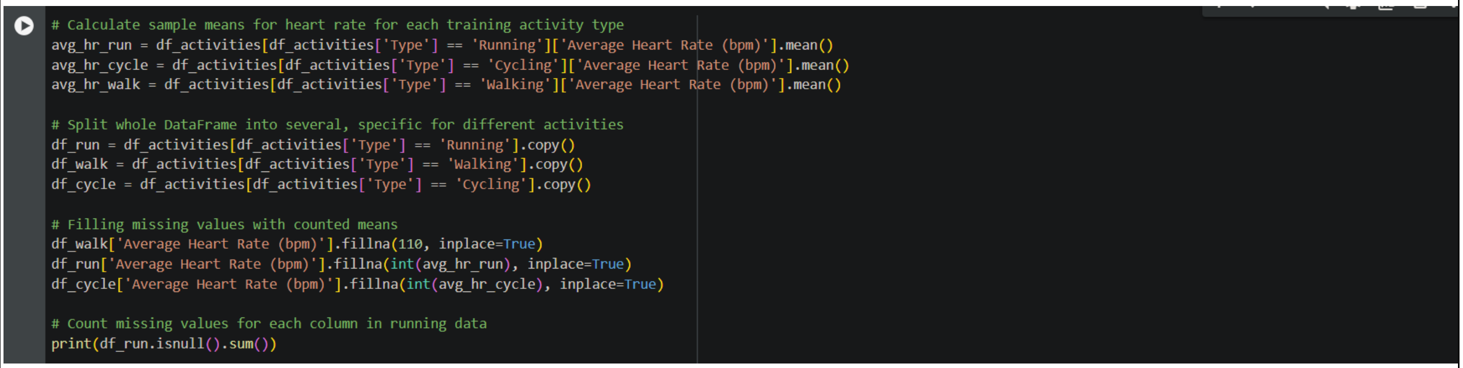
```
Type
Running     459
Cycling      29
Walking      18
Other         2
Name: count, dtype: int64
Date                        0
Time                        0
Type                        0
Distance (km)               0
Duration                    0
Average Pace                0
Average Speed (km/h) 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

Here our results shows that, there are 459 activity entry which is running within the dataset, similarly other activities number are also presented.

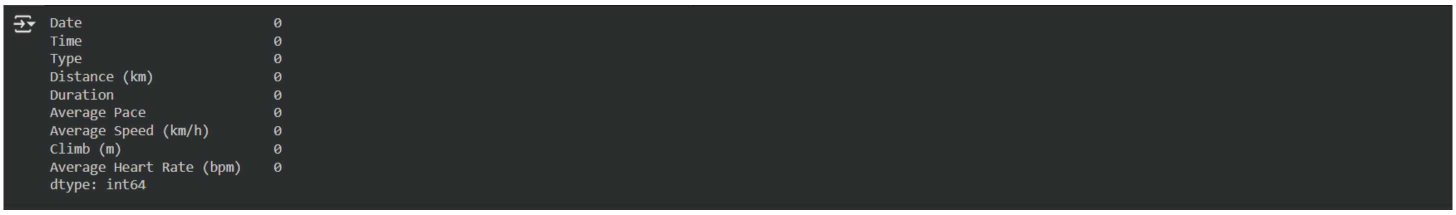
Next, we found number of null values in every column, and we see that all columns not useful for our analysis have been removed.

Missing Values: As we can see from the last output, there are 214 missing entries for my average heart rate.

We can fill in the missing values with an average value. This process is called mean imputation. When imputing the mean to fill in missing data, we need to consider that the average heart rate varies for different activities



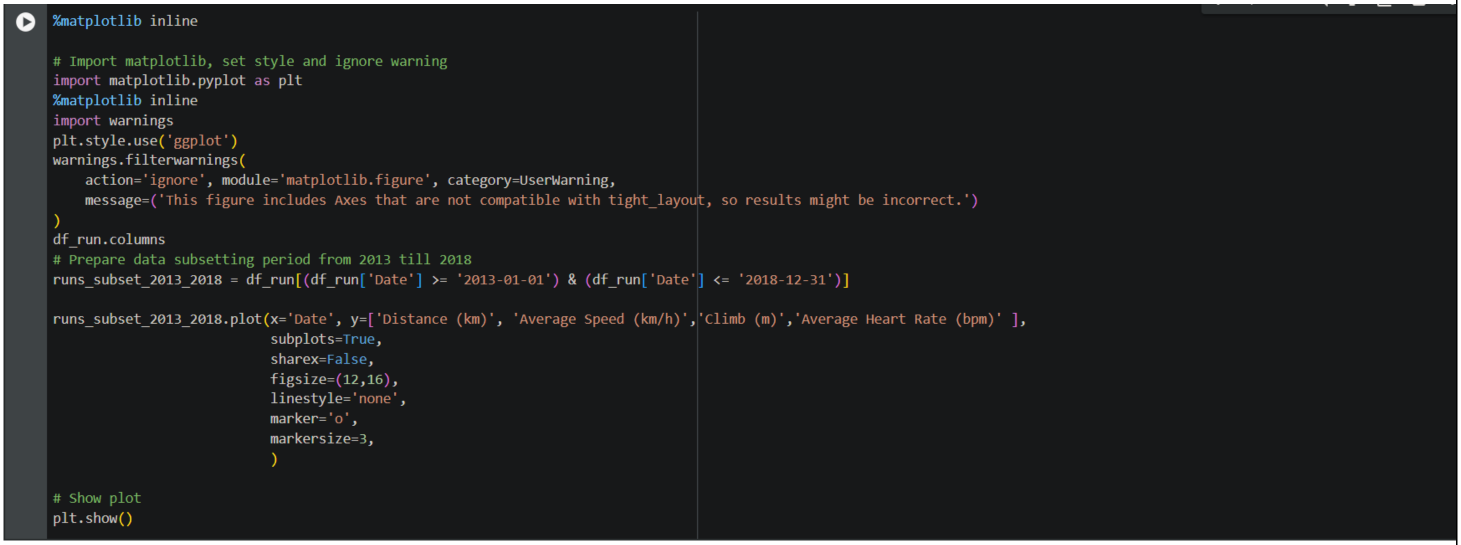
```python
# Calculate sample means for heart rate for each training activity type
avg_hr_run = df_activities[df_activities['Type'] == 'Running']['Average Heart Rate (bpm)'].mean()
avg_hr_cycle = df_activities[df_activities['Type'] == 'Cycling']['Average Heart Rate (bpm)'].mean()
avg_hr_walk = df_activities[df_activities['Type']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



```
Date                        0
Time                        0
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
```

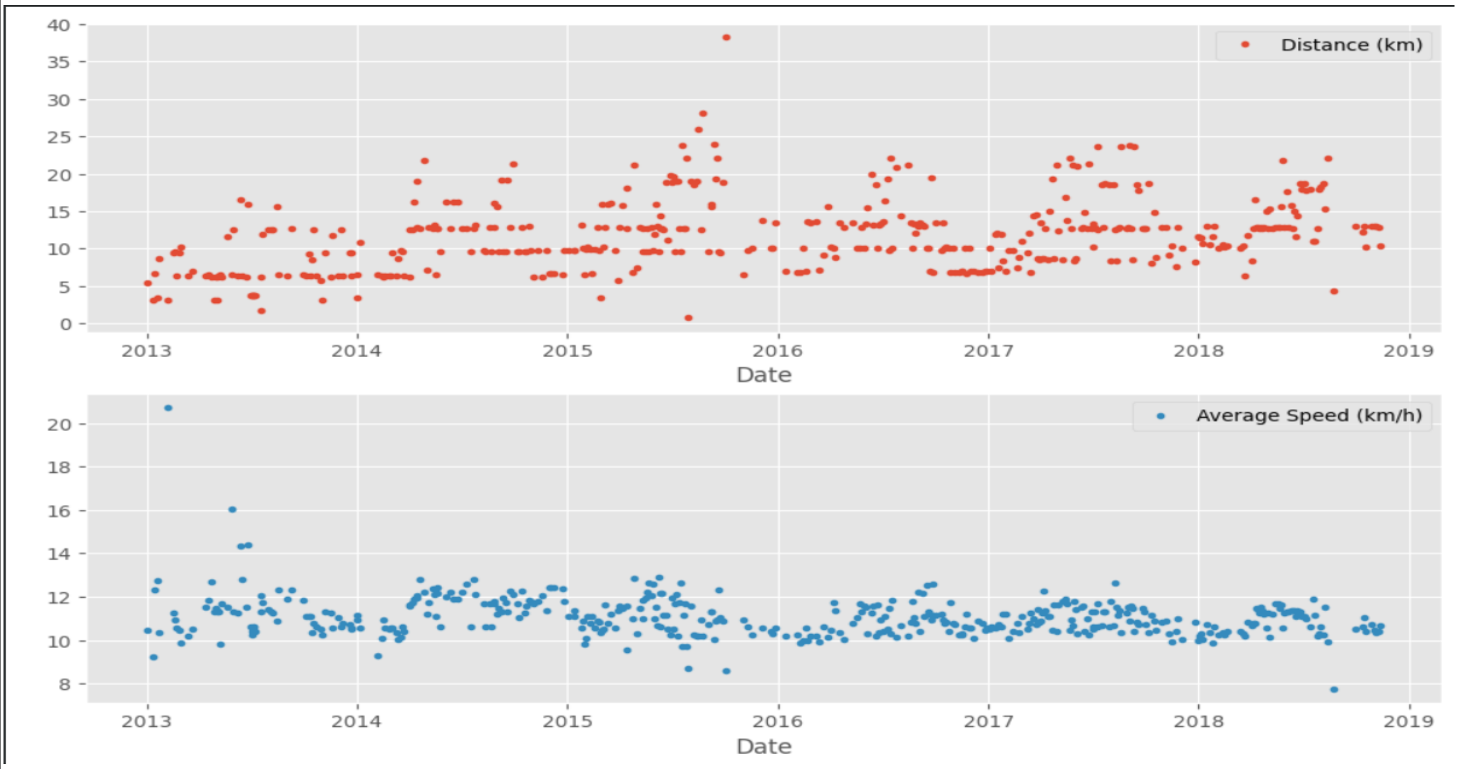
Now that we have removed all sorts of missing data, we can now begin analyzing and plotting data and finally generating insights which can be helpful in improving our health and training regiment.

Now, plotting data in beginning can make it easier foe us to see how data is changing with time or any other constraint. It gives us basic idea about the story data is trying to say.

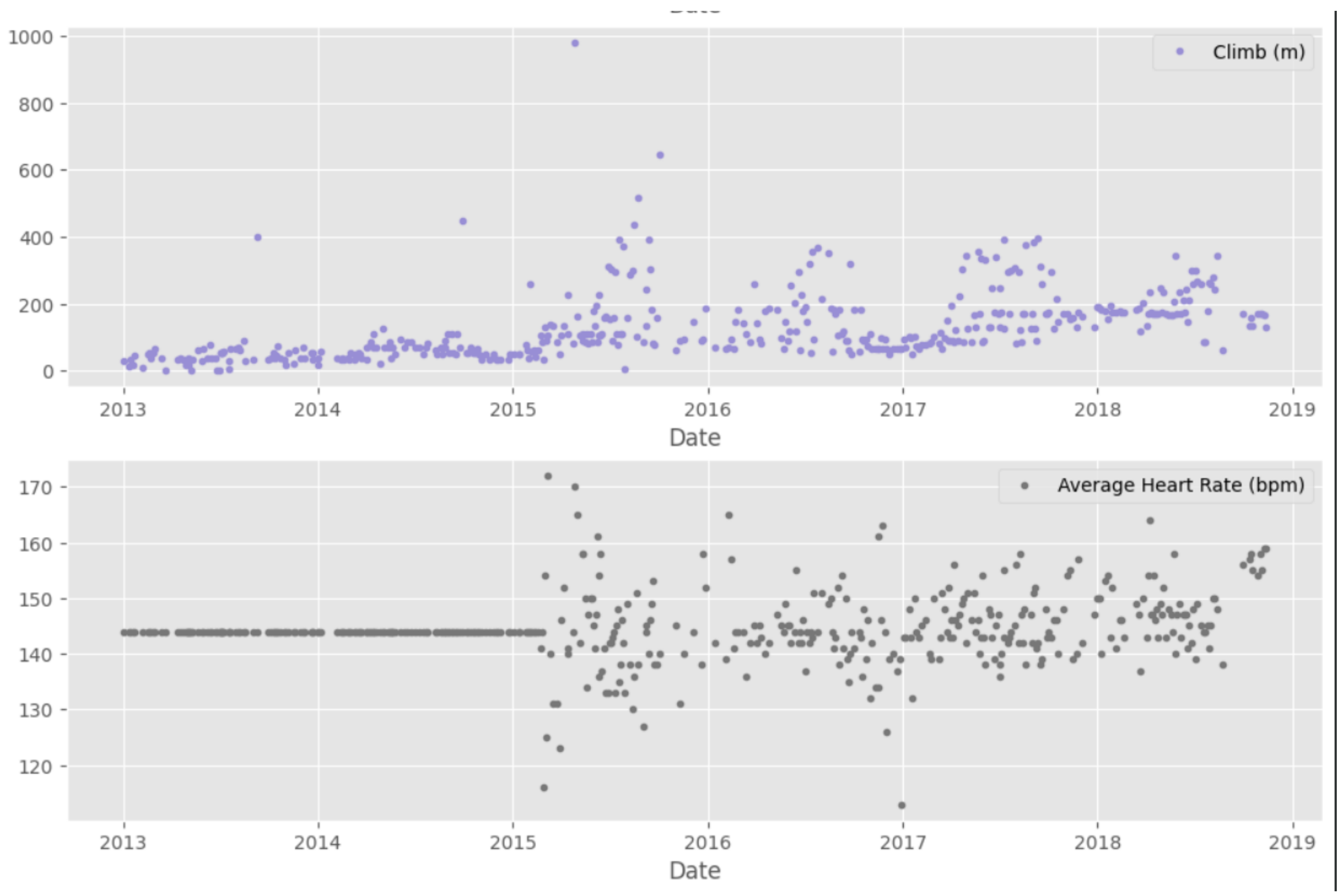


```
%matplotlib inline

# 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.')
)
df_run.columns
# Prepare data subsetting period from 2013 till 2018
runs_subset_2013_2018 = df_run[(df_run['Date'] >= '2013-01-01') & (df_run['Date'] <= '2018-12-31')]

runs_subset_2013_2018.plot(x='Date', y=['Distance (km)', 'Average Speed (km/h)','Climb (m)','Average Heart Rate (bpm)' ],
                           subplots=True,
                           sharex=False,
                           figsize=(12,16),
                           linestyle='none',
                           marker='o',
                           markersize=3,
                           )

# Show plot
plt.show()
```

Plot below shows us how distance and average speed of the runner varies from year to year, it helps in seeing progress within the data



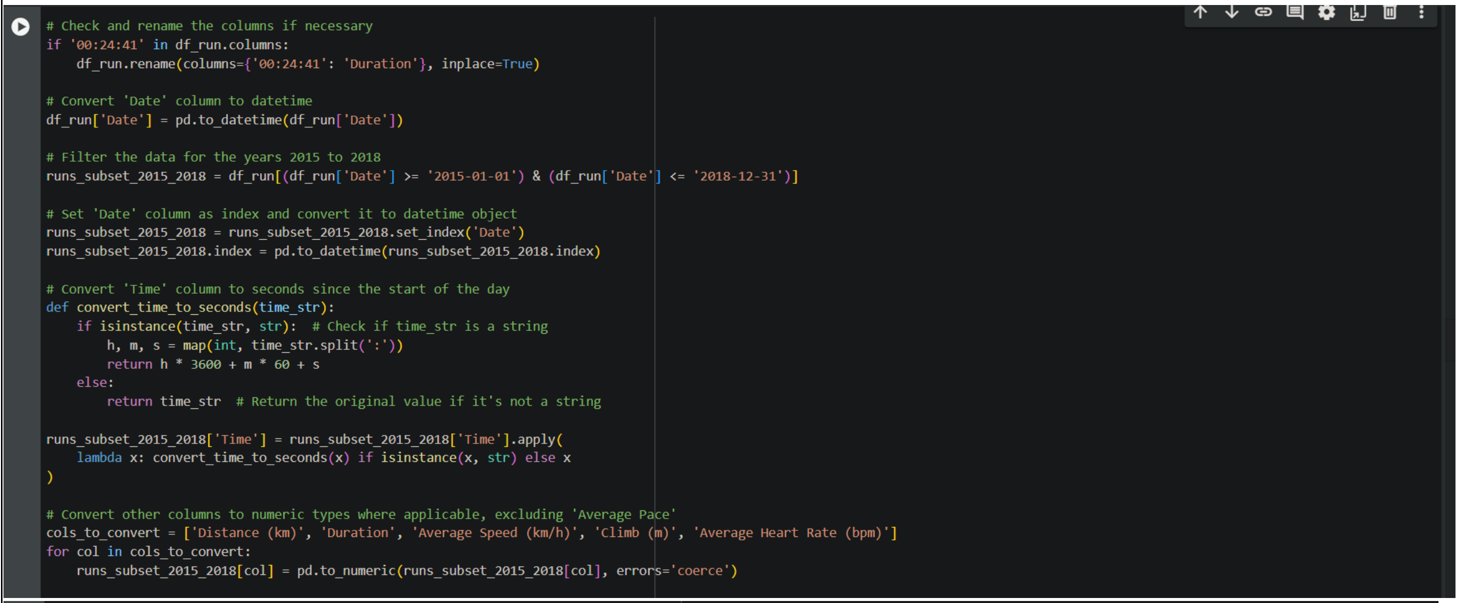
The plot below shows the climb(m) and average heart rate(bpm) of the runner, here climb shows the incline the runner has ran through and we also have the record of the heartbeat pattern.



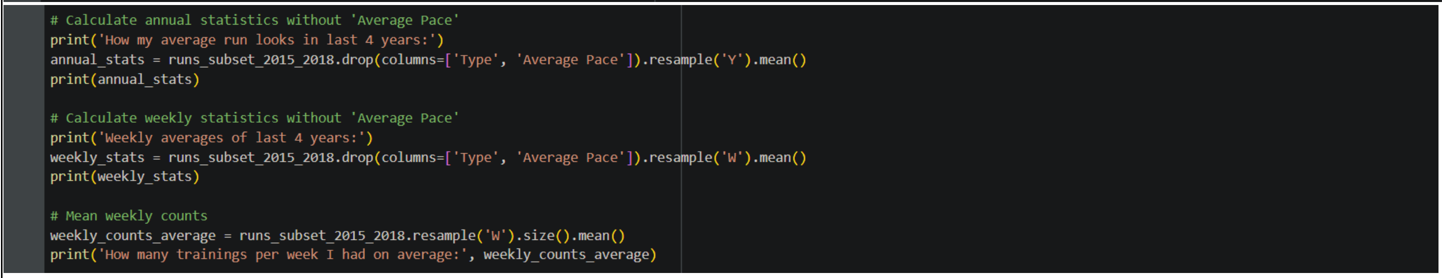
3.4 Analyzing Data: It is a crucial step where we analyze our data to find pattern, insights to come to a conclusion.
Here with the fitness data we have we need to answer few questions:

- What is your average distance?
- How fast do you run?
- Do you measure your heart rate?
- How often do you train?

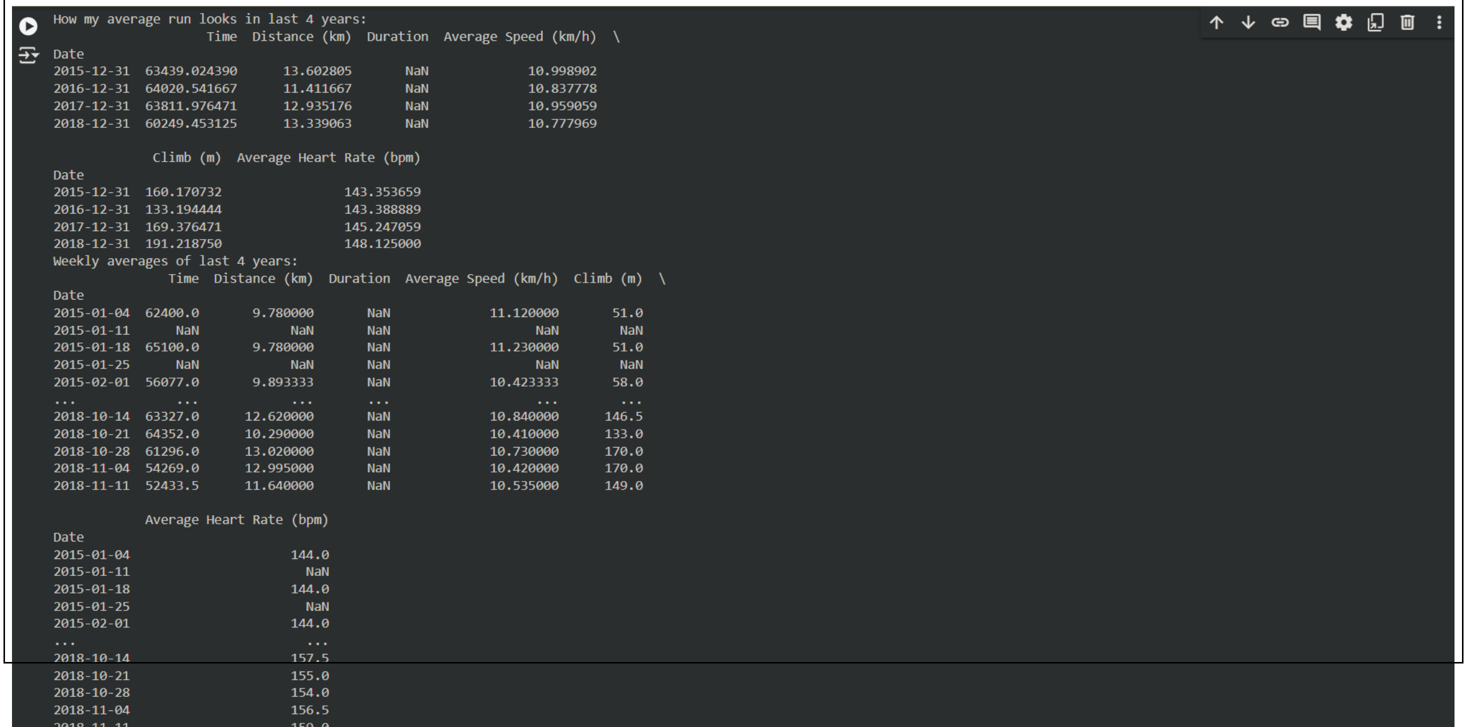
To answer such question we analyse the data



```
# Check and rename the columns if necessary
if '00:24:41' in df_run.columns:
    df_run.rename(columns={'00:24:41': 'Duration'}, inplace=True)

# Convert 'Date' column to datetime
df_run['Date'] = pd.to_datetime(df_run['Date'])

# Filter the data for the years 2015 to 2018
runs_subset_2015_2018 = df_run[(df_run['Date'] >= '2015-01-01') & (df_run['Date'] <= '2018-12-31')]

# Set 'Date' column as index and convert it to datetime object
runs_subset_2015_2018 = runs_subset_2015_2018.set_index('Date')
runs_subset_2015_2018.index = pd.to_datetime(runs_subset_2015_2018.index)

# Convert 'Time' column to seconds since the start of the day
def convert_time_to_seconds(time_str):
    if isinstance(time_str, str):  # Check if time_str is a string
        h, m, s = map(int, time_str.split(':'))
        return h * 3600 + m * 60 + s
    else:
        return time_str  # Return the original value if it's not a string

runs_subset_2015_2018['Time'] = runs_subset_2015_2018['Time'].apply(
    lambda x: convert_time_to_seconds(x) if isinstance(x, str) else x
)

# Convert other columns to numeric types where applicable, excluding 'Average Pace'
cols_to_convert = ['Distance (km)', 'Duration', 'Average Speed (km/h)', 'Climb (m)', 'Average Heart Rate (bpm)']
for col in cols_to_convert:
    runs_subset_2015_2018[col] = pd.to_numeric(runs_subset_2015_2018[col], errors='coerce')
```

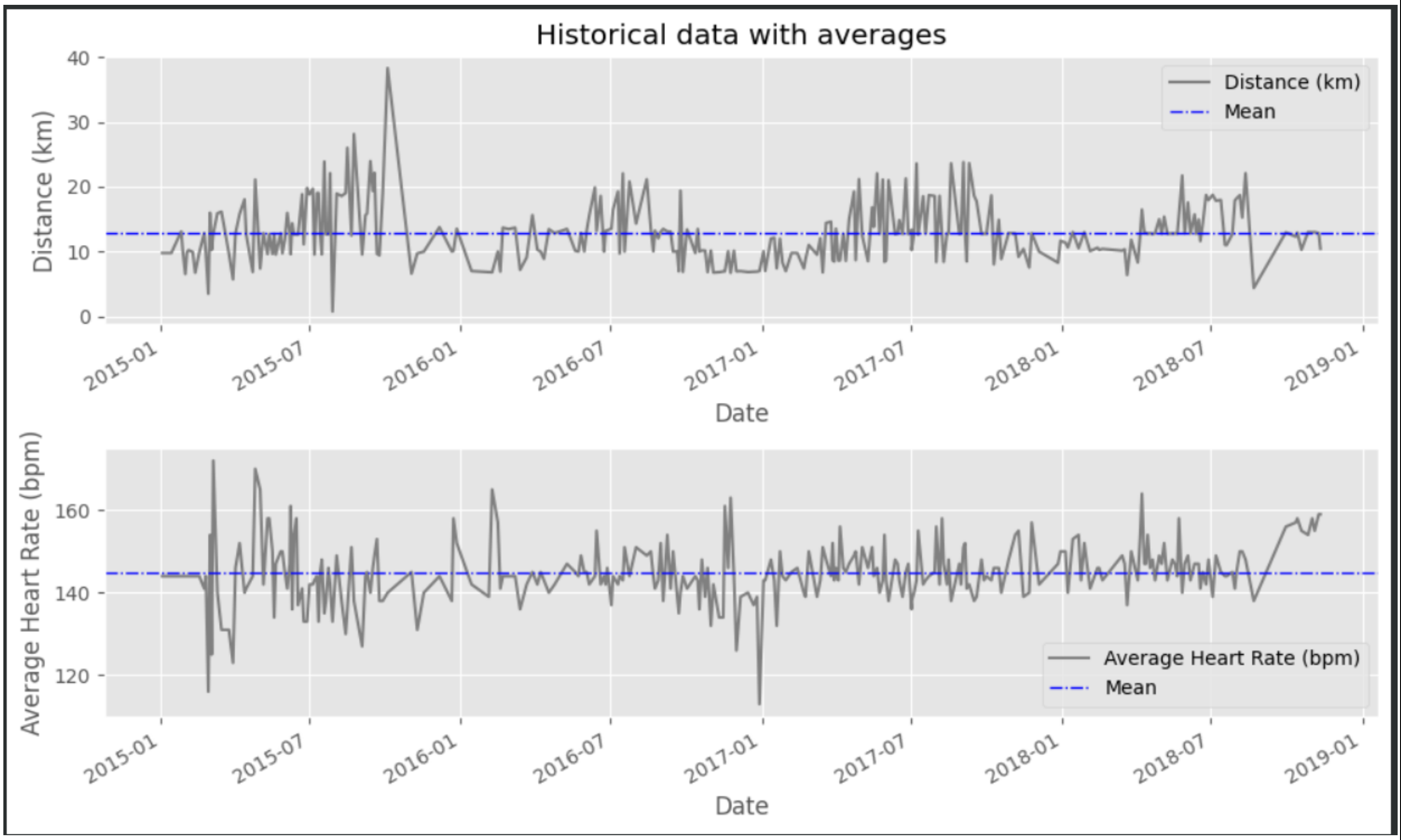


```
# Calculate annual statistics without 'Average Pace'
print('How my average run looks in last 4 years:')
annual_stats = runs_subset_2015_2018.drop(columns=['Type', 'Average Pace']).resample('Y').mean()
print(annual_stats)

# Calculate weekly statistics without 'Average Pace'
print('Weekly averages of last 4 years:')
weekly_stats = runs_subset_2015_2018.drop(columns=['Type', 'Average Pace']).resample('W').mean()
print(weekly_stats)

# Mean weekly counts
weekly_counts_average = runs_subset_2015_2018.resample('W').size().mean()
print('How many trainings per week I had on average:', weekly_counts_average)
```



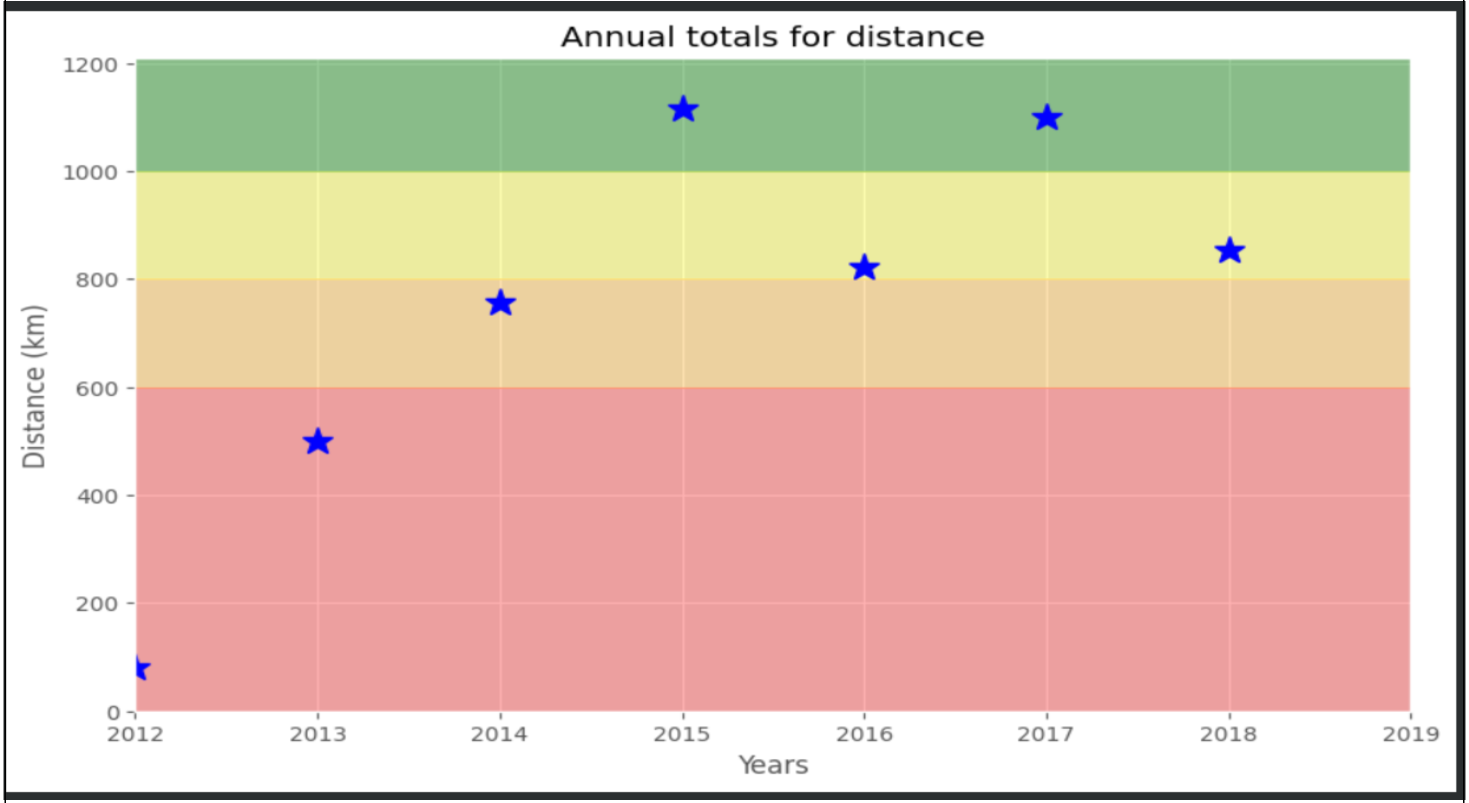
```
How my average run looks in last 4 years:
                    Time  Distance (km)  Duration  Average Speed (km/h)  \
Date
2015-12-31  63439.024390      13.602805       NaN             10.998902
2016-12-31  64020.541667      11.411667       NaN             10.837778
2017-12-31  63811.976471      12.935176       NaN             10.959059
2018-12-31  60249.453125      13.339063       NaN             10.777969

             Climb (m)  Average Heart Rate (bpm)
Date
2015-12-31  160.170732                143.353659
2016-12-31  133.194444                143.388889
2017-12-31  169.376471                145.247059
2018-12-31  191.218750                148.125000
Weekly averages of last 4 years:
               Time  Distance (km)  Duration  Average Speed (km/h)  Climb (m)  \
Date
2015-01-04  62400.0       9.780000       NaN             11.120000       51.0
2015-01-11      NaN            NaN       NaN                   NaN        NaN
2015-01-18  65100.0       9.780000       NaN             11.230000       51.0
2015-01-25      NaN            NaN       NaN                   NaN        NaN
2015-02-01  56077.0       9.893333       NaN             10.423333       58.0
...             ...            ...       ...                   ...        ...
2018-10-14  63327.0      12.620000       NaN             10.840000      146.5
2018-10-21  64352.0      10.290000       NaN             10.410000      133.0
2018-10-28  61296.0      13.020000       NaN             10.730000      170.0
2018-11-04  54269.0      12.995000       NaN             10.420000      170.0
2018-11-11  52433.5      11.640000       NaN             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                     150.0
```

3.5 Visualizing Data: Data becomes easier to understand one it is converted to imagery, as its easier to understand and see patterns in data as an image when compared to text report



This plot here shows the measure ‘of distance’ and ‘average heart rate’ of the runner from year 2015 to 2019, where gray colored graph shows the measurements, we have a dotted blue line showing the average of both measurements

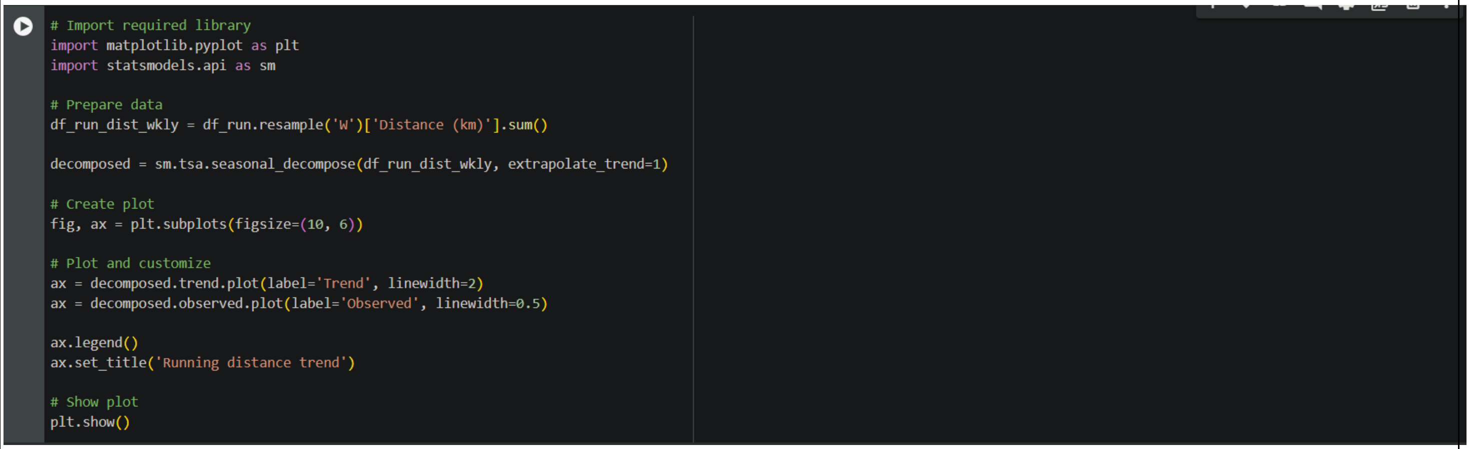
Now, after plotting and learning about data we realize every person has a goal to overcome during their fitness regime, thus we now analyze and plot data to see weather the runner was successful in improving himself



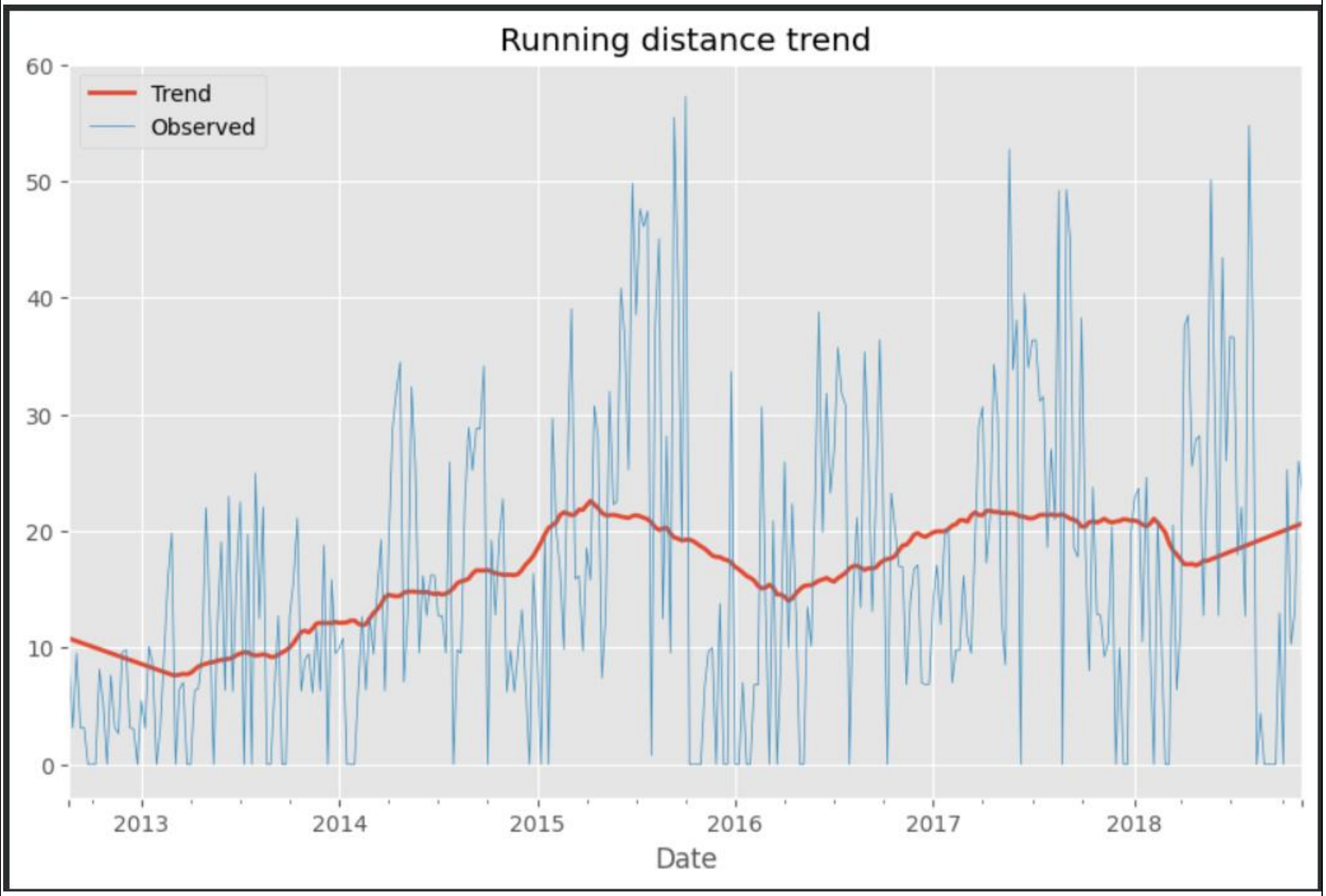
Now according to the visualization of the data we have here, runner had goal of running ‘1000 KM’ a year .

I have highlighted each years running distance with a blue star, now we can observe here 2 instances where runner accomplishes his goals thus their stars being present in green region.

Now we wish to see the improvement, this training regime brought us:



```
# Import required library
import matplotlib.pyplot as plt
import statsmodels.api as sm

# Prepare data
df_run_dist_wkly = df_run.resample('W')['Distance (km)'].sum()

decomposed = sm.tsa.seasonal_decompose(df_run_dist_wkly, extrapolate_trend=1)

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

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

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

# Show plot
plt.show()
```

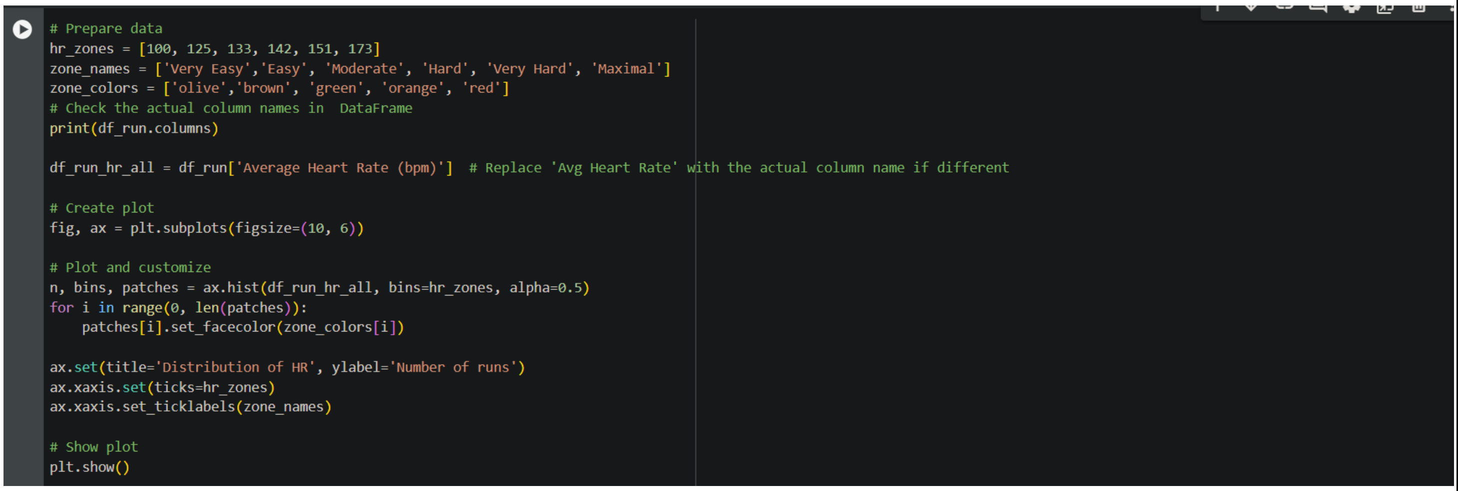


Now in this Viz we see a trend of how the runner showed increasing numbers of distance traveled by running which dropped at early quarter of 2015.

We can also observe runner improving his habits again and from early quarter of 2016 his stats improves again.

Now we also need a representation of intensity of the training runner has gone through, thus we plot heart rate of the runner as it is best metric available to measure the intensity of training.

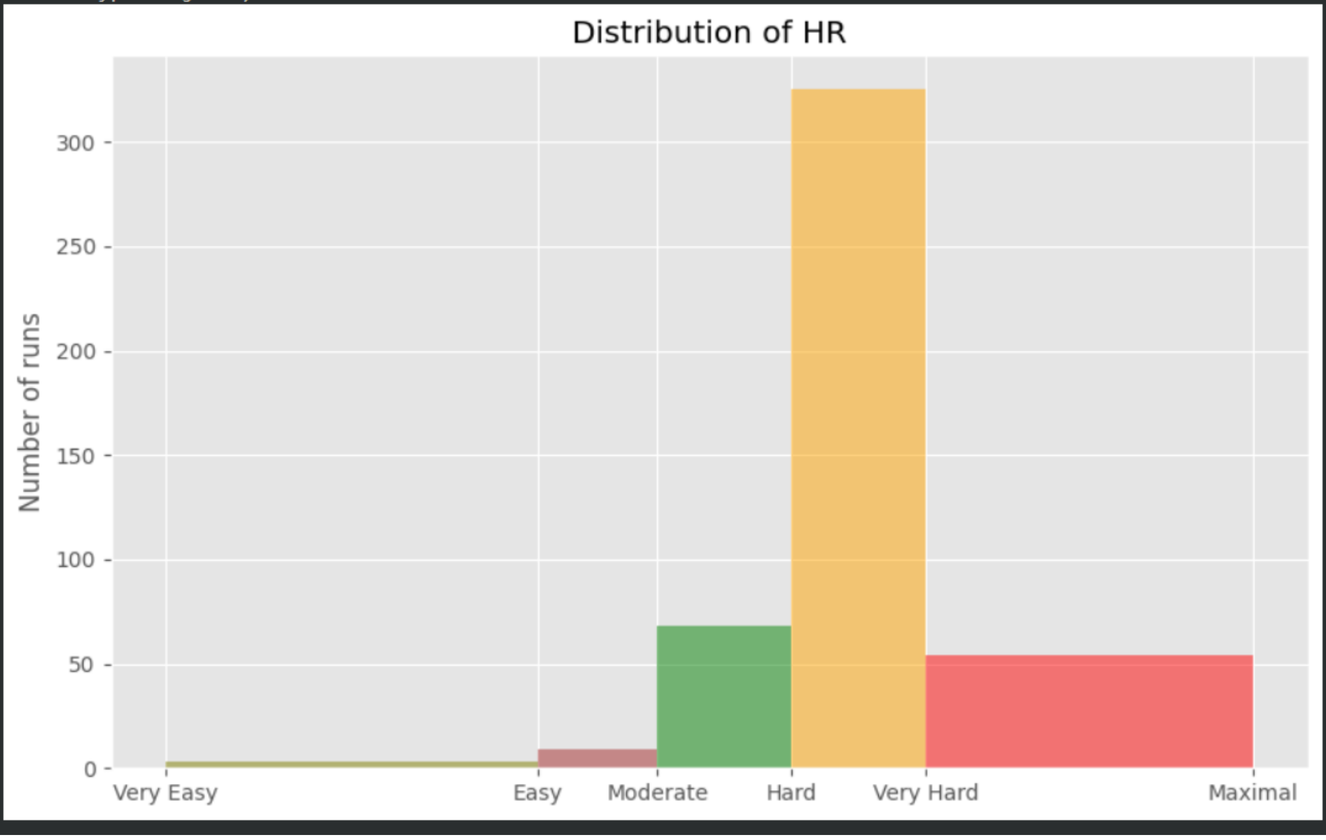
It means higher the heart rate, higher will be the intensity of training.



```
# Prepare data
hr_zones = [100, 125, 133, 142, 151, 173]
zone_names = ['Very Easy','Easy', 'Moderate', 'Hard', 'Very Hard', 'Maximal']
zone_colors = ['olive','brown', 'green', 'orange', 'red']
# Check the actual column names in  DataFrame
print(df_run.columns)

df_run_hr_all = df_run['Average Heart Rate (bpm)']  # Replace 'Avg Heart Rate' with the actual column name if different

# 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')
ax.xaxis.set(ticks=hr_zones)
ax.xaxis.set_ticklabels(zone_names)

# Show plot
plt.show()
```

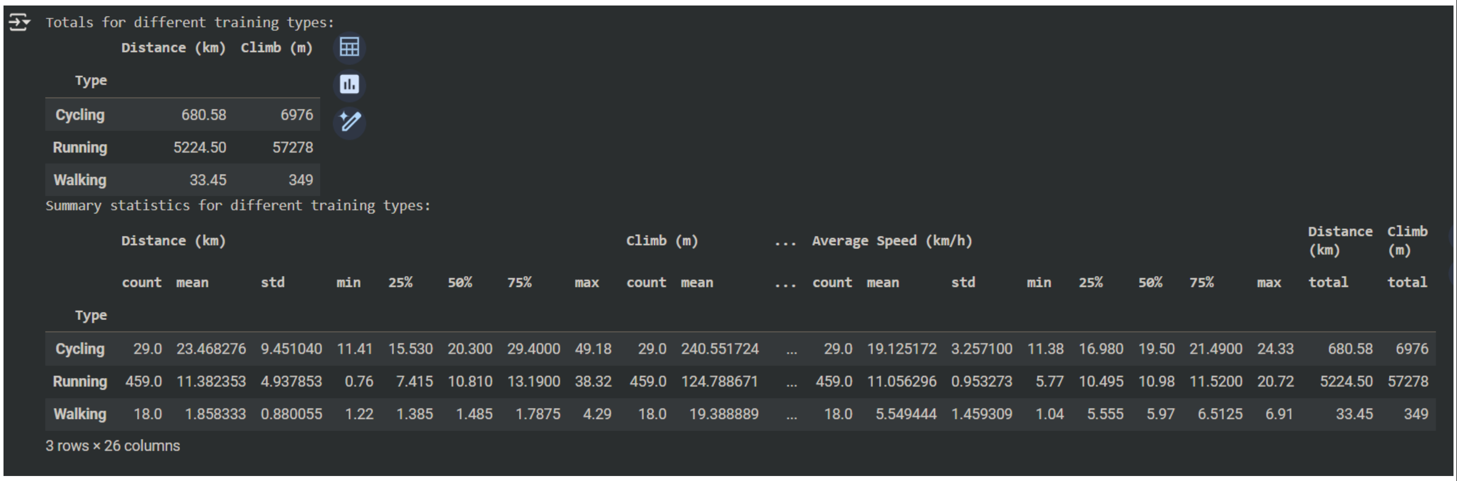
Here we plot for training intensity

```
Index(['Time', 'Type', 'Distance (km)', 'Duration', 'Average Pace',
       'Average Speed (km/h)', 'Climb (m)', 'Average Heart Rate (bpm)'],
      dtype='object')
```



Here we can see colors associated with the heart rate value ranging from light exercise to intense cardio activity.

Now we need a detailed summary of all the activities runner has gone through to make it easier to look, in one place



Totals for different training types:

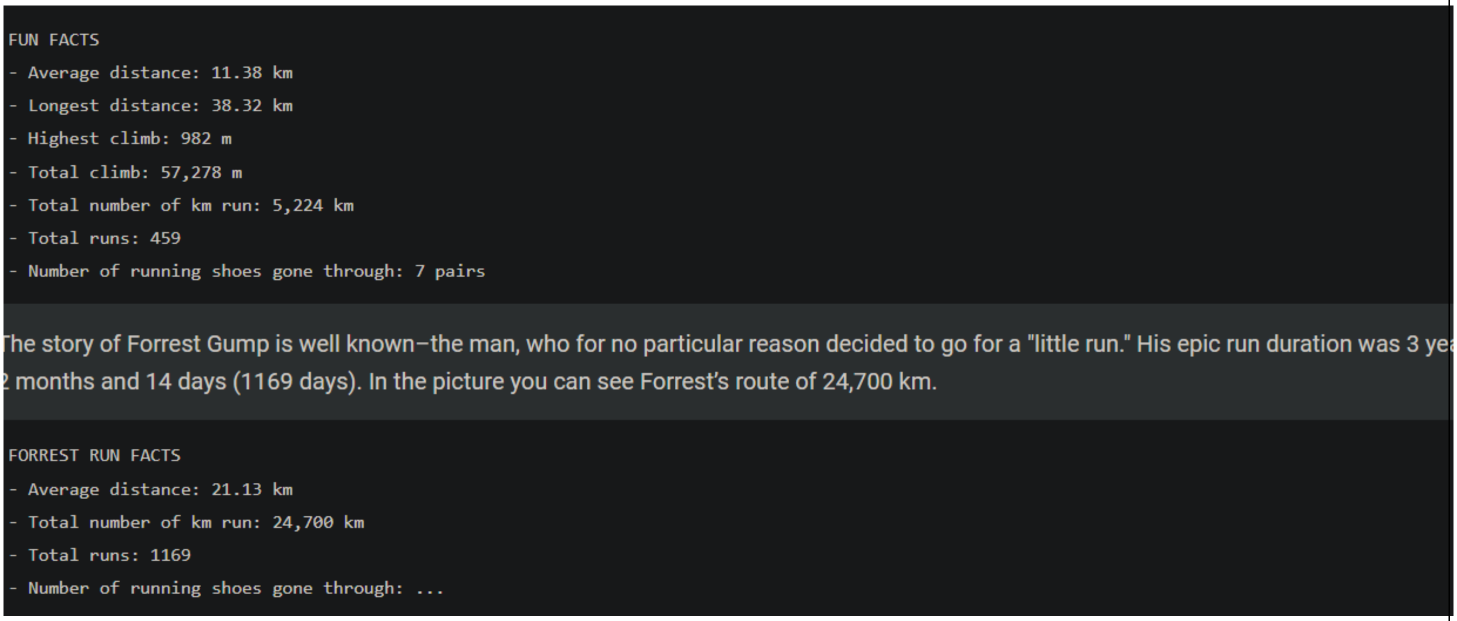
Type	Distance (km)	Climb (m)
Cycling	680.58	6976
Running	5224.50	57278
Walking	33.45	349

Summary statistics for different training types:

	Distance (km)								Climb (m)		...	Average Speed (km/h)								Distance (km)	Climb (m)
	count	mean	std	min	25%	50%	75%	max	count	mean	...	count	mean	std	min	25%	50%	75%	max	total	total
Type																					
Cycling	29.0	23.468276	9.451040	11.41	15.530	20.300	29.4000	49.18	29.0	240.551724	...	29.0	19.125172	3.257100	11.38	16.980	19.50	21.4900	24.33	680.58	6976
Running	459.0	11.382353	4.937853	0.76	7.415	10.810	13.1900	38.32	459.0	124.788671	...	459.0	11.056296	0.953273	5.77	10.495	10.98	11.5200	20.72	5224.50	57278
Walking	18.0	1.858333	0.880055	1.22	1.385	1.485	1.7875	4.29	18.0	19.388889	...	18.0	5.549444	1.459309	1.04	5.555	5.97	6.5125	6.91	33.45	349

3 rows × 26 columns

FUN FACT:

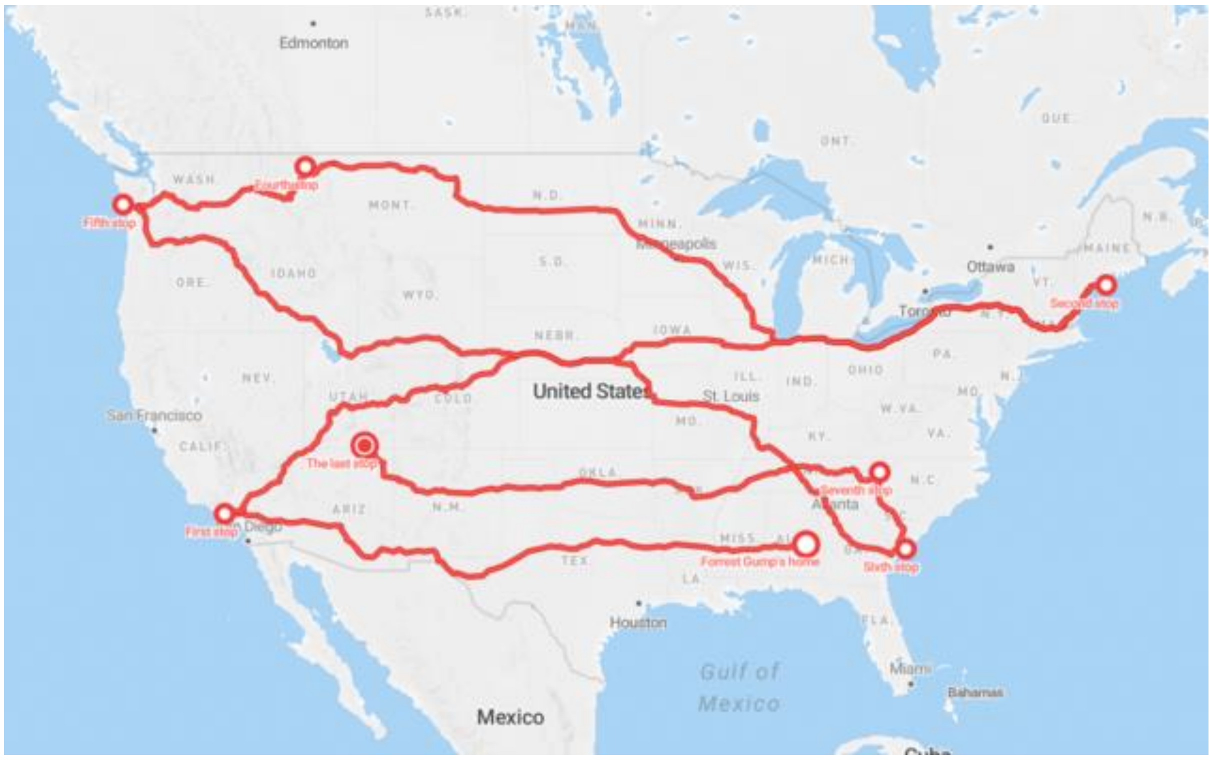


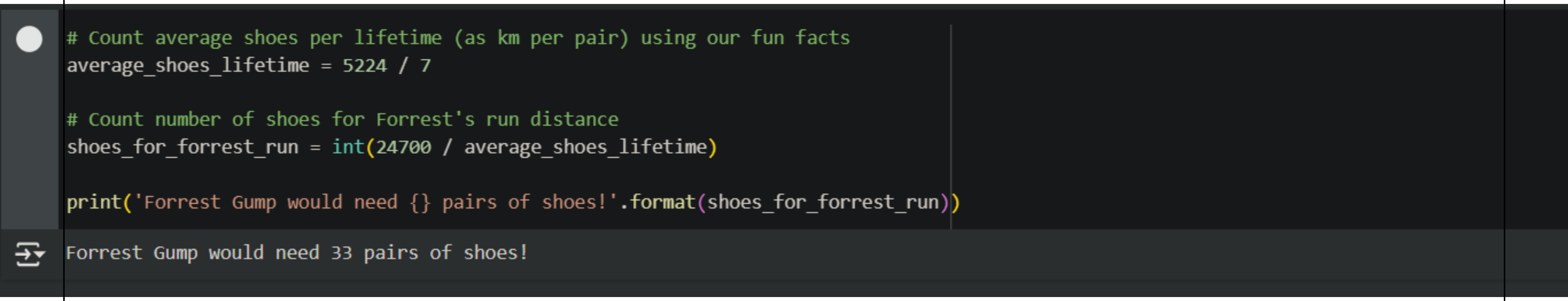
```
FUN FACTS
- Average distance: 11.38 km
- Longest distance: 38.32 km
- Highest climb: 982 m
- Total climb: 57,278 m
- Total number of km run: 5,224 km
- Total runs: 459
- Number of running shoes gone through: 7 pairs
```

he story of Forrest Gump is well known–the man, who for no particular reason decided to go for a "little run." His epic run duration was 3 ye
2 months and 14 days (1169 days). In the picture you can see Forrest's route of 24,700 km.

```
FORREST RUN FACTS
- Average distance: 21.13 km
- Total number of km run: 24,700 km
- Total runs: 1169
- Number of running shoes gone through: ...
```

Now we will compare runners stats with Forrest Gump, and find the distance travelled as well as number of times Gump would have to change his shoes





```
# Count average shoes per lifetime (as km per pair) using our fun facts
average_shoes_lifetime = 5224 / 7

# Count number of shoes for Forrest's run distance
shoes_for_forrest_run = int(24700 / average_shoes_lifetime)

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

```
Forrest Gump would need 33 pairs of shoes!
```

Thank you