SIT225: Data Analysis & interpretation

Run each cell to generate output and finally convert this notebook to PDF.

In [2]:
```
# Fill in student ID and name
#
student_id = "221435713"
student_first_last_name = "Anish Bansal"
print(student_id, student_first_last_name)
```

```
221435713 Anish Bansal
```

1. Descriptive Statistics

Descriptive statistics summarizes important features of a data set such as:

- Count
- Sum
- Standard deviation
- Percentile
- Average

In [4]:
```
# Make sure necessary packages are already installed.
!pip install pandas numpy seaborn

import pandas as pd
import numpy as np
import seaborn as sns

full_health_data = pd.read_csv("full_health_data.csv", header=0, sep=",")
print (full_health_data.describe())
```

```
Requirement already satisfied: pandas in /Users/jazz/anaconda3/lib/python
3.12/site-packages (2.2.2)
Requirement already satisfied: numpy in /Users/jazz/anaconda3/lib/python3.
12/site-packages (1.26.4)
Requirement already satisfied: seaborn in /Users/jazz/anaconda3/lib/python
3.12/site-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/jazz/anaco
nda3/lib/python3.12/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /Users/jazz/anaconda3/li
b/python3.12/site-packages (from pandas) (2023.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /Users/jazz/anac
onda3/lib/python3.12/site-packages (from seaborn) (3.8.4)
Requirement already satisfied: contourpy>=1.0.1 in /Users/jazz/anaconda3/l
ib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.
0)
Requirement already satisfied: cycler>=0.10 in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/jazz/anaconda3/
lib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.5
1.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/jazz/anaconda3/
lib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.
4)
Requirement already satisfied: packaging>=20.0 in /Users/jazz/anaconda3/li
b/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in /Users/jazz/anaconda3/lib/pyth
on3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/jazz/anaconda3/l
ib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.0.
9)
Requirement already satisfied: six>=1.5 in /Users/jazz/anaconda3/lib/pytho
n3.12/site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
         Duration  Average_Pulse   Max_Pulse  Calorie_Burnage  Hours_Work
\
count  163.000000     163.000000  163.000000       163.000000  163.000000
mean    64.263804     107.723926  134.226994       382.368098    4.386503
std     42.994520      14.625062   16.403967       274.227106    3.923772
min     15.000000      80.000000  100.000000        50.000000    0.000000
25%     45.000000     100.000000  124.000000       256.500000    0.000000
50%     60.000000     105.000000  131.000000       320.000000    5.000000
75%     60.000000     111.000000  141.000000       388.500000    8.000000
max    300.000000     159.000000  184.000000      1860.000000   11.000000

       Hours_Sleep
count   163.000000
mean      7.680982
std       0.663934
min       5.000000
25%       7.500000
50%       8.000000
75%       8.000000
max      12.000000
```

1.1 Percentile

25%, 50% and 75% - Percentiles

Observe the output of the above cell for 25%, 50% and 75% of all the columns. Let's explain for Average_Pulse:

- 25% of all of the training sessions have an average pulse of 100 beats per minute or lower. If we flip the statement, it means that 75% of all of the training sessions have an average pulse of 100 beats per minute or higher.
- 75% of all the training session have an average pulse of 111 or lower. If we flip the statement, it means that 25% of all of the training sessions have an average pulse of 111 beats per minute or higher.

In [15]:
```
import numpy as np

# Assuming full_health_data is your DataFrame
max_pulse = full_health_data["Max_Pulse"]

# Calculating percentiles for Max_Pulse
percentile_10_max = np.percentile(max_pulse, 10)
percentile_25_max = np.percentile(max_pulse, 25)
percentile_50_max = np.percentile(max_pulse, 50)
percentile_75_max = np.percentile(max_pulse, 75)

print("Max_Pulse Percentiles")
print("10% Percentile:", percentile_10_max)
print("25% Percentile:", percentile_25_max)
print("50% Percentile (Median):", percentile_50_max)
print("75% Percentile:", percentile_75_max)

# Output for comparison
print("\nAverage_Pulse Percentiles (for comparison):")
print("10% Percentile:", 92.2)
print("25% Percentile:", 100.0)
print("50% Percentile:", 105.0)
print("75% Percentile:", 111.0)
```

```
Max_Pulse Percentiles
10% Percentile: 120.0
25% Percentile: 124.0
50% Percentile (Median): 131.0
75% Percentile: 141.0

Average_Pulse Percentiles (for comparison):
10% Percentile: 92.2
25% Percentile: 100.0
50% Percentile: 105.0
75% Percentile: 111.0
```

Question: Calculate percentiles for Max_Pulse.

You should answer a follow up question in the activity sheet.

1.2 Standard Deviation

Standard deviation is a number that describes how spread out the observations are.

A mathematical function will have difficulties in predicting precise values, if the observations are "spread". Standard deviation is a measure of uncertainty.

A low standard deviation means that most of the numbers are close to the mean (average) value.

A high standard deviation means that the values are spread out over a wider range.

In [6]:
```
import numpy as np

# We can use the std() function from Numpy to find the standard deviation

std = np.std(full_health_data)
print(std)
```

```
Duration           42.862432
Average_Pulse      14.580131
Max_Pulse          16.353571
Calorie_Burnage   273.384624
Hours_Work          3.911718
Hours_Sleep         0.661895
dtype: float64
```

```
/Users/jazz/anaconda3/lib/python3.12/site-packages/numpy/core/fromnumeric.
py:3643: FutureWarning: The behavior of DataFrame.std with axis=None is de
precated, in a future version this will reduce over both axes and return a
scalar. To retain the old behavior, pass axis=0 (or do not pass axis)
  return std(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

1.2.1 Coefficient of variation

In the above cell, what does standard deviation numbers mean?

The coefficient of variation is used to get an idea of how large the standard deviation is.

Mathematically, the coefficient of variation is defined as:

$$Coefficient of Variation = Standard Deviation / Mean$$

In [7]:
```
cv = np.std(full_health_data) / np.mean(full_health_data)
print(cv)

# We see that the variables Duration and Calorie_Burnage has
# a high Standard Deviation compared to Max_Pulse, Average_Pulse and Hour
#
```

```
Duration          0.367051
Average_Pulse     0.124857
Max_Pulse         0.140043
Calorie_Burnage   2.341122
Hours_Work        0.033498
Hours_Sleep       0.005668
dtype: float64
```

```
/Users/jazz/anaconda3/lib/python3.12/site-packages/numpy/core/fromnumeric.
py:3643: FutureWarning: The behavior of DataFrame.std with axis=None is de
precated, in a future version this will reduce over both axes and return a
scalar. To retain the old behavior, pass axis=0 (or do not pass axis)
  return std(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

1.3 Variance

Variance is another number that indicates how spread out the values are.

In fact, if you take the square root of the variance, you get the standard deviation. Or the other way around, if you multiply the standard deviation by itself, you get the variance!

In [8]:
```
var = np.var(full_health_data)
print(var)
```

```
Duration            1837.188076
Average_Pulse        212.580225
Max_Pulse            267.439271
Calorie_Burnage    74739.152847
Hours_Work            15.301536
Hours_Sleep            0.438105
dtype: float64
```

```
/Users/jazz/anaconda3/lib/python3.12/site-packages/numpy/core/fromnumeric.
py:3785: FutureWarning: The behavior of DataFrame.var with axis=None is de
precated, in a future version this will reduce over both axes and return a
scalar. To retain the old behavior, pass axis=0 (or do not pass axis)
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

1.4 Correlation

Correlation measures the relationship between two variables.

A function has a purpose to predict a value, by converting input (x) to output (f(x)). We can say also say that a function uses the relationship between two variables for prediction.

Correlation Coefficient

The correlation coefficient measures the relationship between two variables.

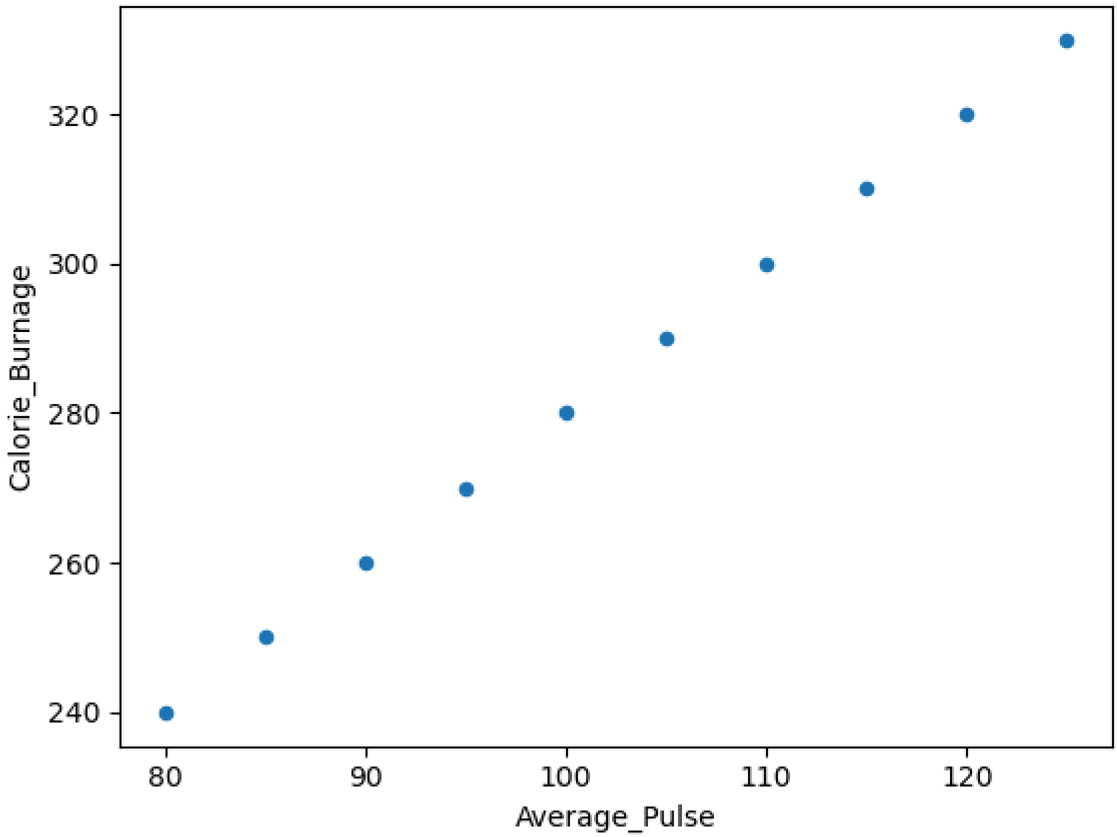
The correlation coefficient can never be less than -1 or higher than 1.

- 1 = there is a perfect linear relationship between the variables
- 0 = there is no linear relationship between the variables
- -1 = there is a perfect negative linear relationship between the variables

Perfect Linear Relationship (Correlation Coefficient = 1)

it exists a perfect linear relationship between Average_Pulse and Calorie_Burnage.

In [9]:
```
# Positive correlation
#

import matplotlib.pyplot as plt

def create_linear_health_data():
    data = [
        {'Duration':30, 'Average_Pulse':80, 'Max_Pulse':120,'Calorie_Burn
        {'Duration':45, 'Average_Pulse':85, 'Max_Pulse':120,'Calorie_Burn
        {'Duration':45, 'Average_Pulse':90, 'Max_Pulse':130,'Calorie_Burn
        {'Duration':60, 'Average_Pulse':95, 'Max_Pulse':130,'Calorie_Burn
        {'Duration':60, 'Average_Pulse':100, 'Max_Pulse':140,'Calorie_Bur
        {'Duration':60, 'Average_Pulse':105, 'Max_Pulse':140,'Calorie_Bur
        {'Duration':60, 'Average_Pulse':110, 'Max_Pulse':145,'Calorie_Bur
        {'Duration':45, 'Average_Pulse':115, 'Max_Pulse':145,'Calorie_Bur
        {'Duration':60, 'Average_Pulse':120, 'Max_Pulse':150,'Calorie_Bur
        {'Duration':45, 'Average_Pulse':125, 'Max_Pulse':150,'Calorie_Bur
    ]
    return data

health_data = pd.DataFrame.from_dict(create_linear_health_data())
health_data.plot(x ='Average_Pulse', y='Calorie_Burnage', kind='scatter')
plt.show()
```



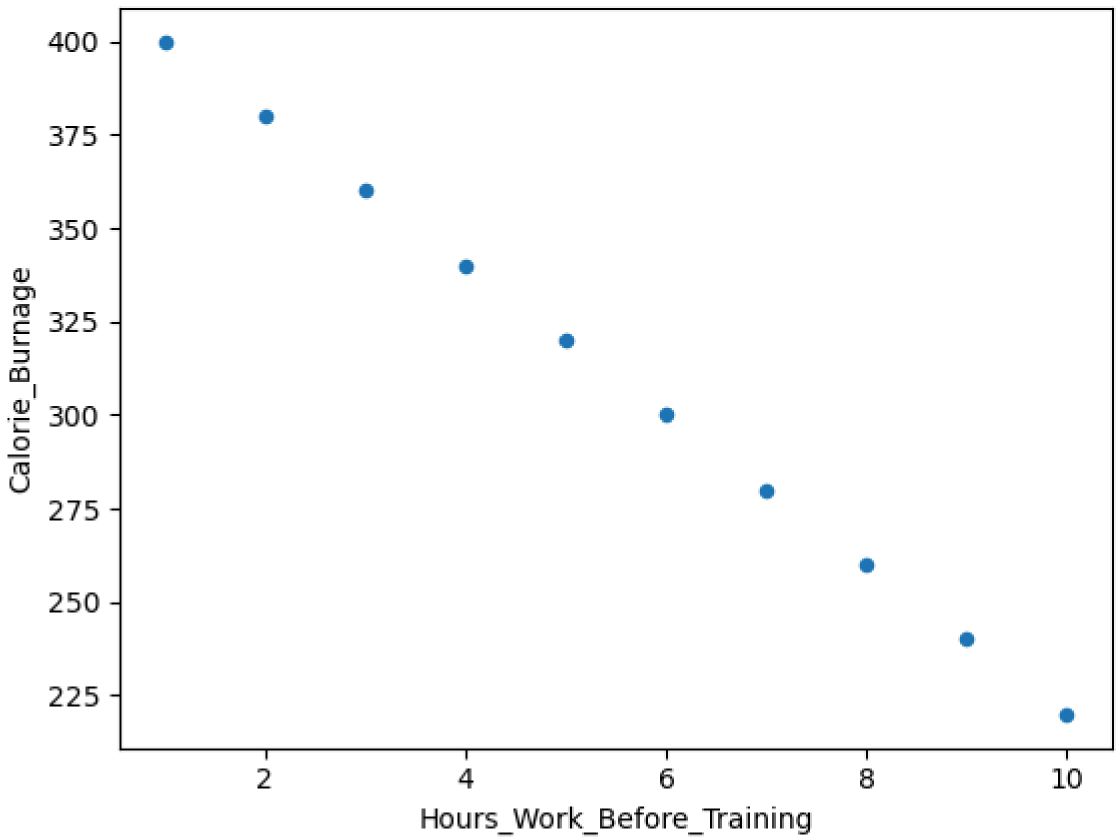
Perfect Negative Linear Relationship (Correlation Coefficient = -1)

We have plotted fictional data here. The x-axis represents the amount of hours worked at our job before a training session. The y-axis is Calorie_Burnage.

If we work longer hours, we tend to have lower calorie burnage because we are exhausted before the training session.

The correlation coefficient here is -1.

In [10]:
```
# Negative correlation
#
negative_corr = {'Hours_Work_Before_Training': [10,9,8,7,6,5,4,3,2,1],
'Calorie_Burnage': [220,240,260,280,300,320,340,360,380,400]}
negative_corr = pd.DataFrame(data=negative_corr)

negative_corr.plot(x ='Hours_Work_Before_Training', y='Calorie_Burnage',
plt.show()
```

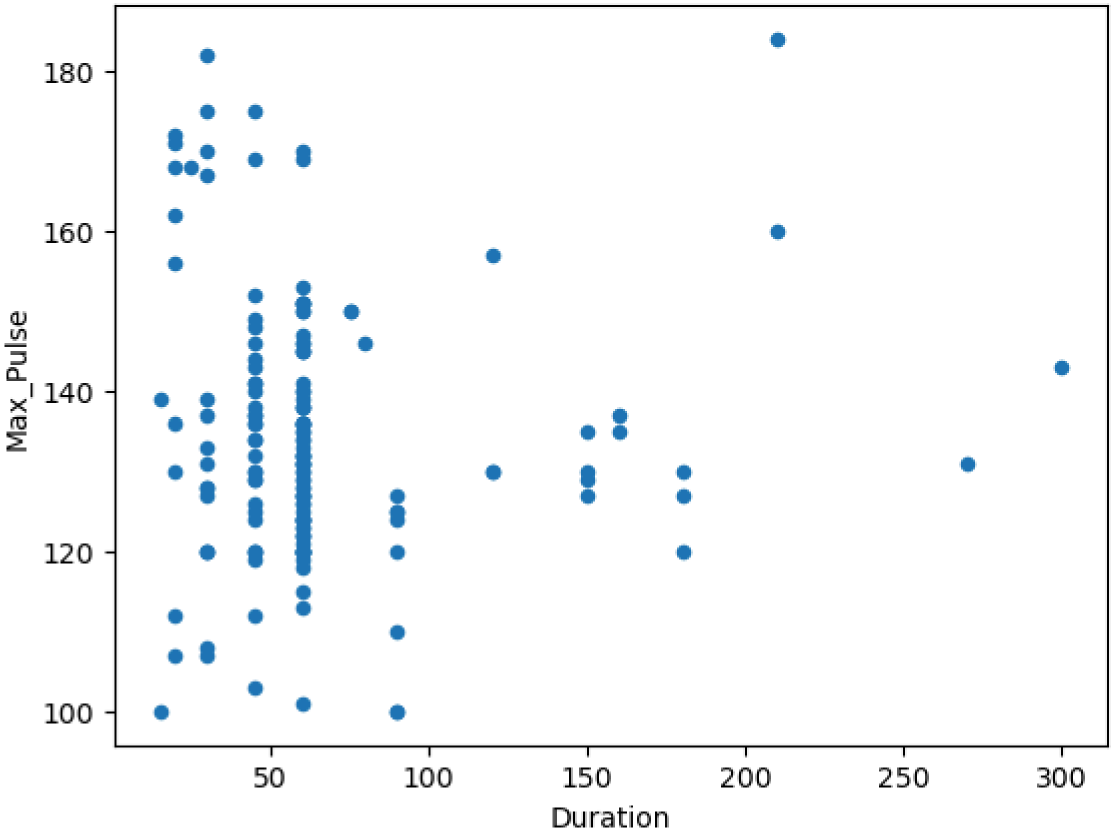


No Linear Relationship (Correlation coefficient = 0)

As you can see, there is no linear relationship between the two variables. It means that longer training session does not lead to higher Max_Pulse.

The correlation coefficient here is 0.

In [11]:
```
full_health_data.plot(x ='Duration', y='Max_Pulse', kind='scatter')
plt.show()
```



1.5 Correlation Matrix

A matrix is an array of numbers arranged in rows and columns.

A correlation matrix is simply a table showing the correlation coefficients between variables.

We can use the corr() function in Python to create a correlation matrix. We also use the round() function to round the output to two decimals:

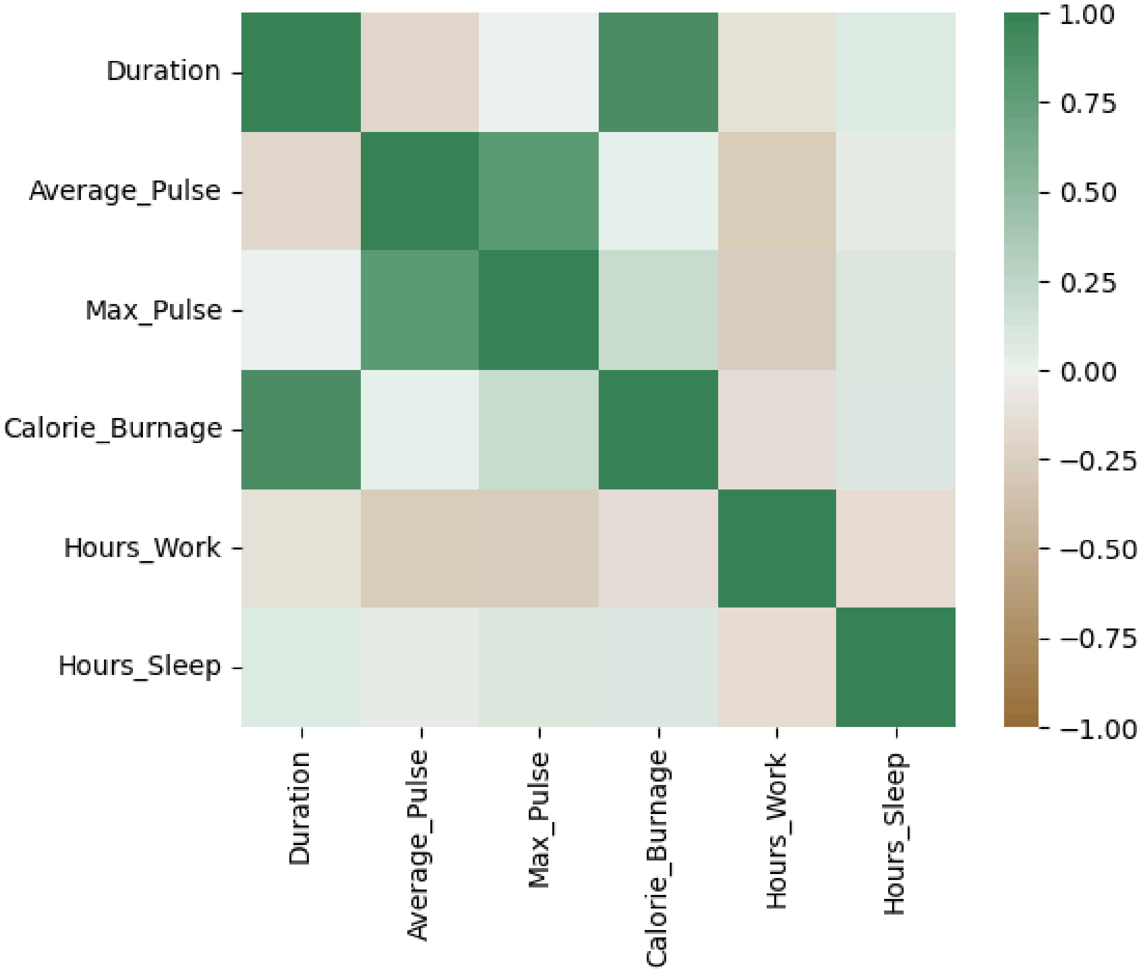
In [12]:
```
Corr_Matrix = round(full_health_data.corr(),2)
print(Corr_Matrix)

# Drop 2 columns - Hours_Work and Hours_Sleep to view the matrix nice.
#
health_part = full_health_data.drop(columns=['Hours_Work', 'Hours_Sleep']
Corr_Matrix = round(health_part.corr(),2)
print(Corr_Matrix)
```

```
                 Duration  Average_Pulse  Max_Pulse  Calorie_Burnage  \
Duration             1.00          -0.17       0.00             0.89
Average_Pulse       -0.17           1.00       0.79             0.02
Max_Pulse            0.00           0.79       1.00             0.20
Calorie_Burnage      0.89           0.02       0.20             1.00
Hours_Work          -0.12          -0.28      -0.27            -0.14
Hours_Sleep          0.07           0.03       0.09             0.08

                 Hours_Work  Hours_Sleep
Duration              -0.12         0.07
Average_Pulse         -0.28         0.03
Max_Pulse             -0.27         0.09
Calorie_Burnage       -0.14         0.08
Hours_Work             1.00        -0.14
Hours_Sleep           -0.14         1.00
                 Duration  Average_Pulse  Max_Pulse  Calorie_Burnage
Duration             1.00          -0.17       0.00             0.89
Average_Pulse       -0.17           1.00       0.79             0.02
Max_Pulse            0.00           0.79       1.00             0.20
Calorie_Burnage      0.89           0.02       0.20             1.00
```

Using a Heatmap

We can use a Heatmap to Visualize the Correlation Between Variables:

In [13]:
```
import matplotlib.pyplot as plt
import seaborn as sns

correlation_full_health = full_health_data.corr()

axis_corr = sns.heatmap(
    correlation_full_health,
    vmin=-1, vmax=1, center=0,
    cmap=sns.diverging_palette(50, 500, n=500),
    square=True
)

plt.show()
```



1.6 Correlation Does not imply Causality

Correlation measures the numerical relationship between two variables.

A high correlation coefficient (close to 1), does not mean that we can for sure conclude an actual relationship between two variables.

A classic example:

- During the summer, the sale of ice cream at a beach increases
- Simultaneously, drowning accidents also increase as well

Question: Does this mean that increase of ice cream sale is a direct cause of increased drowning accidents?

The question presented regarding the correlation between ice cream sales and drowning accidents helps highlight the concept that "Correlation does not imply causality." Even if two variables, such as ice cream sales and drowning incidents, show a high correlation, it does not mean one directly causes the other. In this case, it is incorrect to conclude that increasing ice cream sales is the cause of more drowning accidents. Instead, a more likely explanation is that both events increase during summer due to higher temperatures, with more people visiting beaches, which increases both ice cream consumption and swimming activities. The weather acts as a common factor influencing both variables, leading to an observed correlation, but no direct cause-and-effect relationship exists between them.

1.7 Linear Regression

The term regression is used when you try to find the relationship between variables.

In Machine Learning and in statistical modeling, that relationship is used to predict the outcome of events.

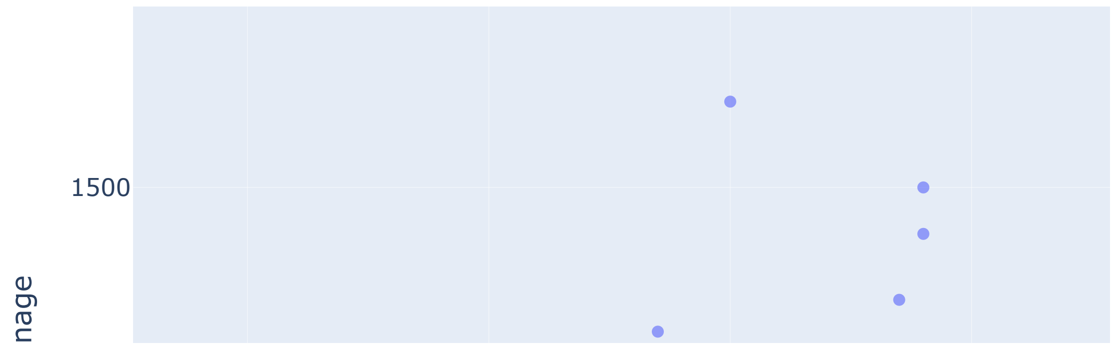
We will use Scikit-learn to train various regression models. Scikit-learn is a popular Machine Learning (ML) library that offers various tools for creating and training ML algorithms, feature engineering, data cleaning, and evaluating and testing models. It was designed to be accessible, and to work seamlessly with popular libraries like NumPy and Pandas.

We see how to apply a simple regression model for predicting Calorie_Burnage on various factors such as Average_Pulse or Duration.

In [14]:

```
!pip install seaborn plotly

import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from sklearn.linear_model import LinearRegression

df = full_health_data
X = df.Average_Pulse.values.reshape(-1, 1)

model = LinearRegression()
model.fit(X, df.Calorie_Burnage)

x_range = np.linspace(X.min(), X.max(), 100)
y_range = model.predict(x_range.reshape(-1, 1))

fig = px.scatter(df, x='Average_Pulse', y='Calorie_Burnage', opacity=0.65
fig.add_traces(go.Scatter(x=x_range, y=y_range, name='Regression Fit'))
fig.show()
```

```
Requirement already satisfied: seaborn in /Users/jazz/anaconda3/lib/python
3.12/site-packages (0.13.2)
Requirement already satisfied: plotly in /Users/jazz/anaconda3/lib/python
3.12/site-packages (5.22.0)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /Users/jazz/anacond
a3/lib/python3.12/site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /Users/jazz/anaconda3/lib/py
thon3.12/site-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /Users/jazz/anac
onda3/lib/python3.12/site-packages (from seaborn) (3.8.4)
Requirement already satisfied: tenacity>=6.2.0 in /Users/jazz/anaconda3/li
b/python3.12/site-packages (from plotly) (8.2.2)
Requirement already satisfied: packaging in /Users/jazz/anaconda3/lib/pyth
on3.12/site-packages (from plotly) (23.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/jazz/anaconda3/l
ib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.
0)
Requirement already satisfied: cycler>=0.10 in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/jazz/anaconda3/
lib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.5
1.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Users/jazz/anaconda3/
lib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.
4)
Requirement already satisfied: pillow>=8 in /Users/jazz/anaconda3/lib/pyth
on3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/jazz/anaconda3/l
ib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.0.
9)
Requirement already satisfied: python-dateutil>=2.7 in /Users/jazz/anacond
a3/lib/python3.12/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /Users/jazz/anaconda3/lib/p
ython3.12/site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /Users/jazz/anaconda3/li
b/python3.12/site-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in /Users/jazz/anaconda3/lib/pytho
n3.12/site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->s
eaborn) (1.16.0)
```



Question:

We have seen earlier how to apply a simple regression model for predicting Calorie_Burnage from Average_Pulse. There might be another candidate Duration in addition to Average_Pulse. You will need to repeat the above linear regression process to find relationsthip between Calorie_Burnage and Duration.

Comment on the both regression lines: Calorie_Burnage - Average_Pulse and Calorie_Burnage - Duration.

In [16]:

```
# Reshape the Duration column to use it in the model
X_duration = df.Duration.values.reshape(-1, 1)

# Fit the model using Duration as input and Calorie_Burnage as output
model_duration = LinearRegression()
model_duration.fit(X_duration, df.Calorie_Burnage)

# Create a range of Duration values for the regression line
x_range_duration = np.linspace(X_duration.min(), X_duration.max(), 100)
y_range_duration = model_duration.predict(x_range_duration.reshape(-1, 1)

# Plot the data points and the regression line for Duration
fig = px.scatter(df, x='Duration', y='Calorie_Burnage', opacity=0.65)
fig.add_traces(go.Scatter(x=x_range_duration, y=y_range_duration, name='R
fig.show()
```

In [17]:
```
# For Average_Pulse model
r_squared_avg_pulse = model.score(X, df.Calorie_Burnage)
print("R-squared for Average_Pulse vs Calorie_Burnage:", r_squared_avg_pu

# For Duration model
r_squared_duration = model_duration.score(X_duration, df.Calorie_Burnage)
print("R-squared for Duration vs Calorie_Burnage:", r_squared_duration)
```

```
R-squared for Average_Pulse vs Calorie_Burnage: 0.00030892484335254267
R-squared for Duration vs Calorie_Burnage: 0.788640826956281
```

In [ ]: