In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/scipy/__init__.py:
146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this ve
rsion of SciPy (detected version 1.26.0
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
workouts2018 = pd.read_csv("workouts-2018.csv")
workouts2019 = pd.read_csv("workouts-2019.csv")
workouts2020 = pd.read_csv("workouts-2020.csv")
workouts2021 = pd.read_csv("workouts-2021.csv")
workouts2022 = pd.read_csv("workouts-2022.csv")
workouts2023 = pd.read_csv("workouts-2023.csv")
```

In [3]:
```
workouts2018.head()
```

Out[3]:

	Title	WorkoutType	WorkoutDescription	PlannedDuration	PlannedDistanceInMeters	W
0	Rest Day	Day Off	There was some real intensity in your sessions...	NaN	NaN	2
1	Rest Day	Day Off	Don't be tempted to squeeze and additional ses...	NaN	NaN	2
2	10-Mile Time Trial or Under/Over Intervals	Bike	Time: 50 mins - 1 hr 10 mins Warm-up: http://b...	1.166667	NaN	2
3	Rest Day	Day Off	If you have been short of time during this tra...	NaN	NaN	2
4	Rest Day	Day Off	This is an important session. Don't skip recov...	NaN	NaN	2

5 rows × 45 columns

In [4]:
```
data = pd.concat([workouts2018, workouts2019,
                  workouts2020, workouts2021,
                  workouts2022, workouts2023])
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4154 entries, 0 to 1104
Data columns (total 45 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Title                    4104 non-null   object 
 1   WorkoutType              4154 non-null   object 
 2   WorkoutDescription       760 non-null    object 
 3   PlannedDuration          1631 non-null   float64
 4   PlannedDistanceInMeters  96 non-null     float64
 5   WorkoutDay               4154 non-null   object 
 6   CoachComments            288 non-null    object 
 7   DistanceInMeters         3847 non-null   float64
 8   PowerAverage             1849 non-null   float64
 9   PowerMax                 1849 non-null   float64
 10  Energy                   1824 non-null   float64
 11  AthleteComments          1496 non-null   object 
 12  TimeTotalInHours         3943 non-null   float64
 13  VelocityAverage          3847 non-null   float64
 14  VelocityMax              0 non-null      float64
 15  CadenceAverage           3001 non-null   float64
 16  CadenceMax               0 non-null      float64
 17  HeartRateAverage         3353 non-null   float64
 18  HeartRateMax             3353 non-null   float64
 19  TorqueAverage            0 non-null      float64
 20  TorqueMax                0 non-null      float64
 21  IF                       3714 non-null   float64
 22  TSS                      3747 non-null   float64
 23  HRZone1Minutes           3344 non-null   float64
 24  HRZone2Minutes           3344 non-null   float64
 25  HRZone3Minutes           3344 non-null   float64
 26  HRZone4Minutes           3344 non-null   float64
 27  HRZone5Minutes           3344 non-null   float64
 28  HRZone6Minutes           3054 non-null   float64
 29  HRZone7Minutes           2725 non-null   float64
 30  HRZone8Minutes           0 non-null      float64
 31  HRZone9Minutes           0 non-null      float64
 32  HRZone10Minutes          0 non-null      float64
 33  PWRZone1Minutes          1824 non-null   float64
 34  PWRZone2Minutes          1824 non-null   float64
 35  PWRZone3Minutes          1824 non-null   float64
 36  PWRZone4Minutes          1824 non-null   float64
 37  PWRZone5Minutes          1824 non-null   float64
 38  PWRZone6Minutes          1824 non-null   float64
 39  PWRZone7Minutes          816 non-null    float64
 40  PWRZone8Minutes          0 non-null      float64
 41  PWRZone9Minutes          0 non-null      float64
 42  PWRZone10Minutes         0 non-null      float64
 43  Rpe                      1457 non-null   float64
 44  Feeling                  1442 non-null   float64
dtypes: float64(39), object(6)
memory usage: 1.5+ MB
```

In [6]:
```
#drop columns that only have null values
data = data.drop(columns=['VelocityMax', 'CadenceMax', 'TorqueAverage',
                          'TorqueMax', 'HRZone8Minutes', 'HRZone9Minutes',
                          'HRZone10Minutes', 'PWRZone8Minutes',
                          'PWRZone9Minutes', 'PWRZone10Minutes'])
```

In [7]:
```
data.dtypes
```

Out[7]:
```
Title                       object
WorkoutType                 object
WorkoutDescription          object
PlannedDuration            float64
PlannedDistanceInMeters    float64
WorkoutDay                  object
CoachComments               object
DistanceInMeters           float64
PowerAverage               float64
PowerMax                   float64
Energy                     float64
AthleteComments             object
TimeTotalInHours           float64
VelocityAverage            float64
CadenceAverage             float64
HeartRateAverage           float64
HeartRateMax               float64
IF                         float64
TSS                        float64
HRZone1Minutes             float64
HRZone2Minutes             float64
HRZone3Minutes             float64
HRZone4Minutes             float64
HRZone5Minutes             float64
HRZone6Minutes             float64
HRZone7Minutes             float64
PWRZone1Minutes            float64
PWRZone2Minutes            float64
PWRZone3Minutes            float64
PWRZone4Minutes            float64
PWRZone5Minutes            float64
PWRZone6Minutes            float64
PWRZone7Minutes            float64
Rpe                        float64
Feeling                    float64
dtype: object
```

In [8]:
```
data['WorkoutDay'] = pd.to_datetime(data['WorkoutDay'])
```

In [9]:
```
data['WorkoutDay'].head()
```

Out[9]:
```
0   2018-01-01
1   2018-01-02
2   2018-01-03
3   2018-01-04
4   2018-01-05
Name: WorkoutDay, dtype: datetime64[ns]
```

In [10]:
```
data['WorkoutType'].unique()
```

Out[10]:
```
array(['Day Off', 'Bike', 'Walk', 'Run', 'Strength', 'Swim', 'Other',
       'MTB', 'Custom', 'X-Train'], dtype=object)
```

In [11]:
```
#import to SQLLite to practice SQL queries
import sqlite3
conn = sqlite3.connect('cyclingdata.db')

data.to_sql('data', conn, if_exists='replace', index=False)
```

Out[11]:
```
4154
```

In [12]:
```
query = """
    SELECT Title, WorkoutType, WorkoutDescription, PlannedDuration, WorkoutDay
    FROM data
    WHERE TimeTotalinHours > 0
        AND WorkoutDescription != 'None'
        AND TSS != "NaN"
        AND Rpe != "NaN"
        AND WorkoutType = 'Bike'OR WorkoutType = 'MTB'
    """

result = pd.read_sql_query(query, conn)
result.head(10)
```

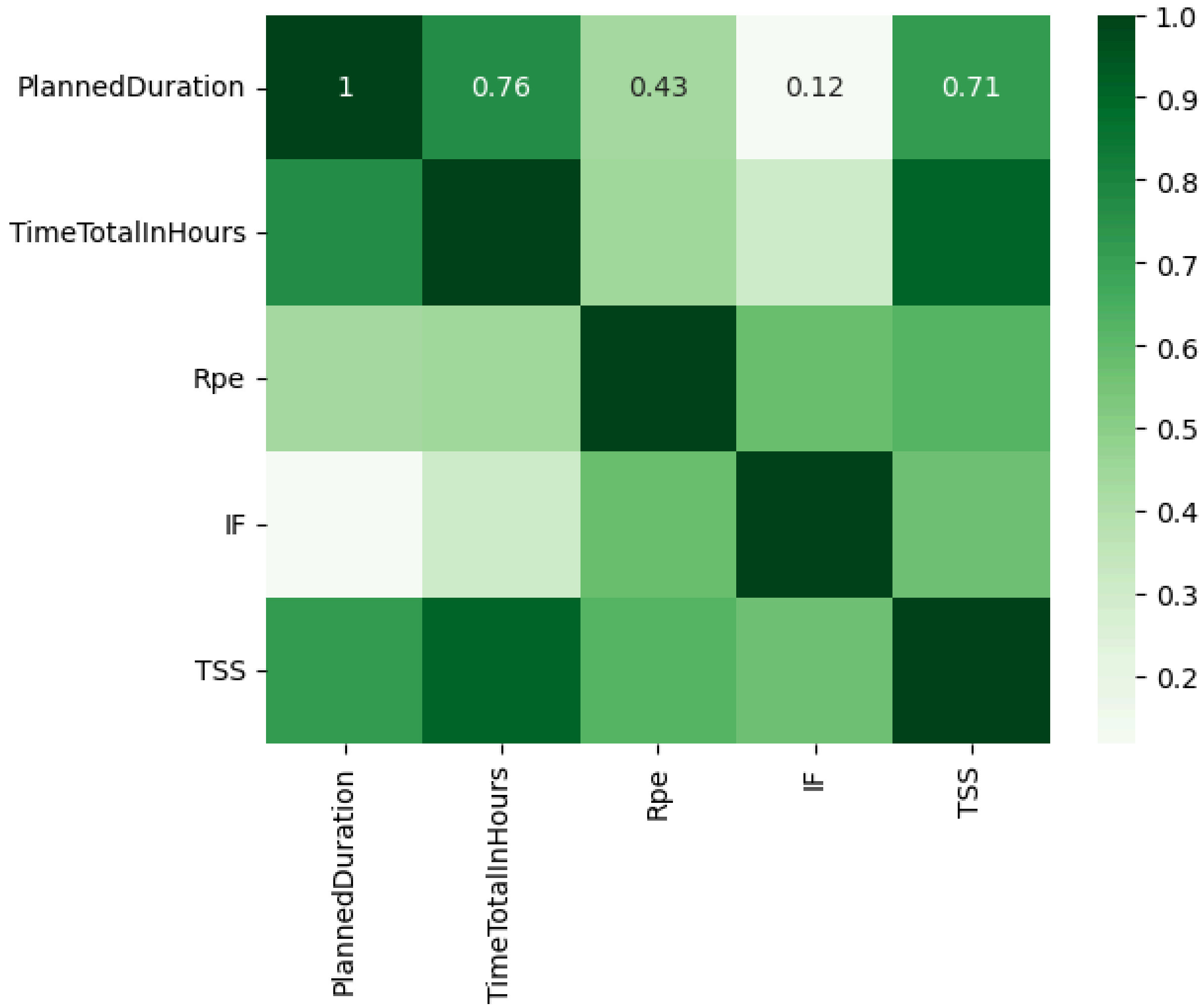
Out[12]:

	Title	WorkoutType	WorkoutDescription	PlannedDuration	WorkoutDay	TimeTotalInHo
0	First intervals of the year! =D	Bike	Duration: 75 minutes Course: Flat or rolling h...	1.250000	2019-02-05 00:00:00	1.2544
1	Zwift - Feb 5 & 7 Intervals	Bike	Duration: 75 minutes Course: Flat or rolling h...	1.250000	2019-02-07 00:00:00	1.263
2	Zwift - VoxTour Stage 3: Hannah Walker (E)	Bike	Duration: 90-120 minutes Course: Flat or rolli...	2.000000	2019-02-09 00:00:00	2.0052
3	Cycling	Bike	Duration: 90-120 minutes Course: Flat or rolli...	1.500000	2019-02-10 00:00:00	1.3258
4	1.5 Hour Ride	Bike	Duration: 90-120 minutes Course: Flat or rolli...	2.000000	2019-02-16 00:00:00	2.1080
5	2 Hour Ride	Bike	Duration: 90-120 minutes Course: Flat or rolli...	1.500000	2019-02-18 00:00:00	1.3255
6	Tuesday Intervals	Bike	Duration: 75 minutes Course: Flat or rolling h...	1.262467	2019-02-19 00:00:00	1.2558
7	"Funtervals"	Bike	Duration: 70 minutes Course: Flat or rolling h...	1.134722	2019-02-21 00:00:00	1.1366
8	Saturday Morning Pain Session	Bike	Duration: 75 minutes Course: Flat or rolling h...	1.200000	2019-02-23 00:00:00	1.2033
9	2 Hour Ride	Bike	Duration: 2 to 2.5 hours Course: Rolling hills...	1.500000	2019-02-24 00:00:00	1.4019

In [15]:
```
matrix = result.corr(numeric_only=True)
```

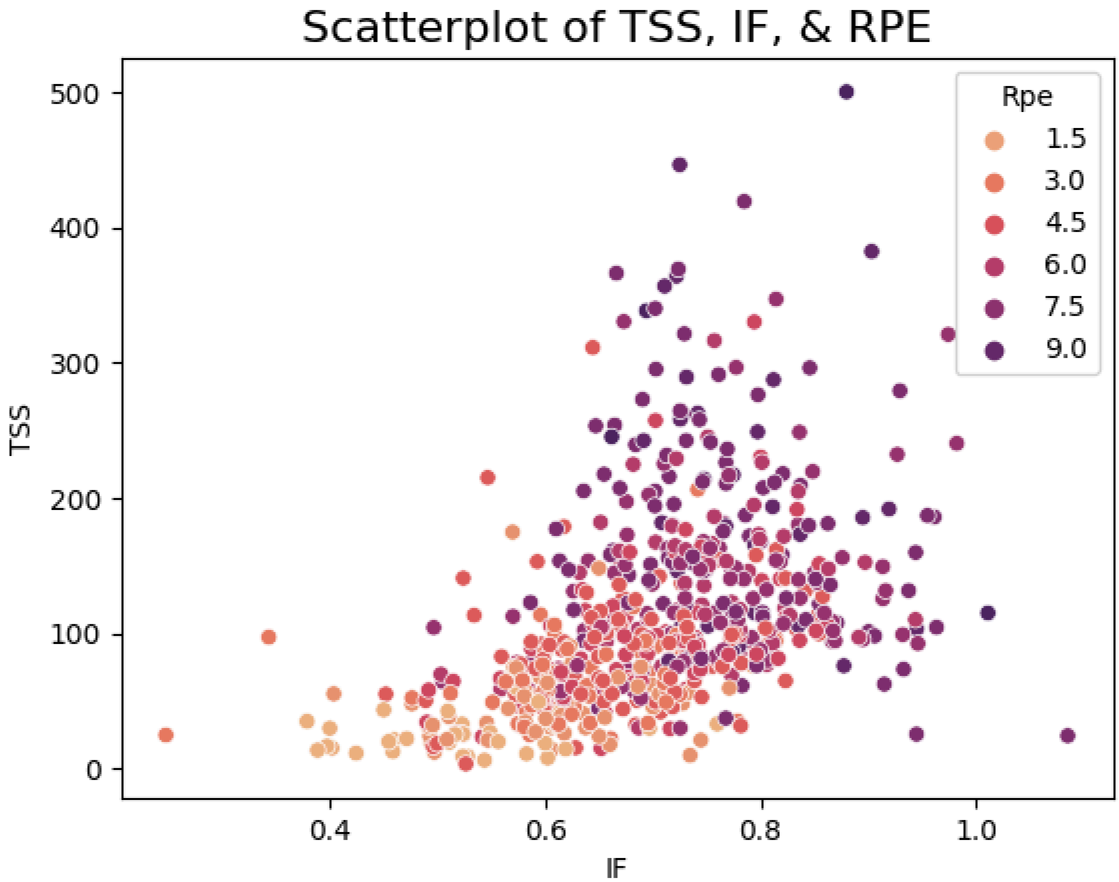
```
#plotting correlation matrix
sns.heatmap(matrix, cmap="Greens", annot=True)
```

Out[15]: <Axes: >



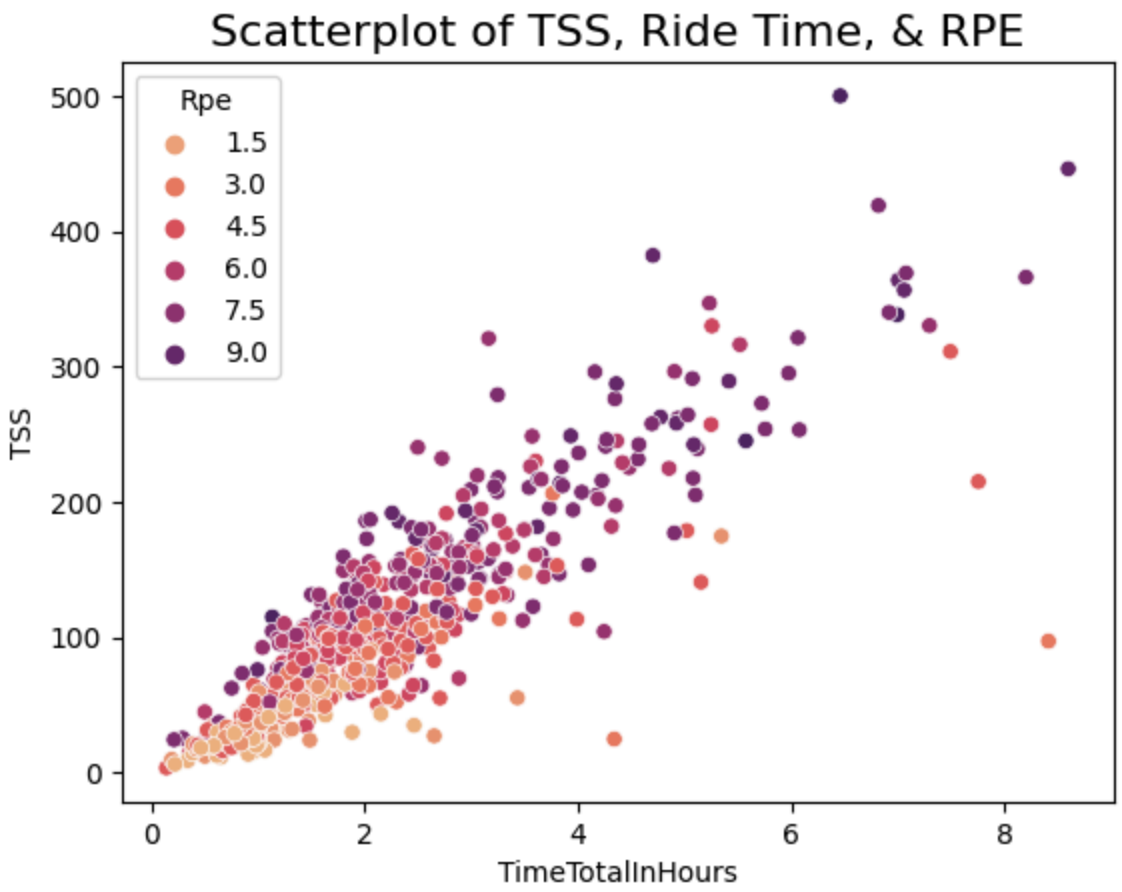
In [13]:
```
sns.scatterplot(data=result, x='IF', y='TSS', hue='Rpe', palette='flare')
plt.title('Scatterplot of TSS, IF, & RPE', fontsize=16)
plt.show()
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



In [14]:
```
sns.scatterplot(data=result, x='TimeTotalInHours', y='TSS', hue='Rpe', palette=
plt.title('Scatterplot of TSS, Ride Time, & RPE', fontsize=16)
plt.show()
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



In [15]:
```
numeric_df = result.drop(columns=result.select_dtypes(exclude=['number']).colu

numeric_df.corr()
```

Out[15]:

	PlannedDuration	TimeTotalInHours	Rpe	IF	TSS
PlannedDuration	1.000000	0.764733	0.433076	0.117775	0.707608
TimeTotalInHours	0.764733	1.000000	0.438718	0.306785	0.903844
Rpe	0.433076	0.438718	1.000000	0.582565	0.625136
IF	0.117775	0.306785	0.582565	1.000000	0.571928
TSS	0.707608	0.903844	0.625136	0.571928	1.000000

In [61]:
```
query = """
    SELECT *
    FROM data
    WHERE TimeTotalinHours > 0
        AND WorkoutDescription != 'None'
        AND TSS <> 'NaN'
        AND RPE <> 'NaN'
        AND Feeling <> 'NaN'
        AND WorkoutType = 'Bike'OR WorkoutType = 'MTB'
    """

result2 = pd.read_sql_query(query, conn)
result2.head()
```

Out[61]:

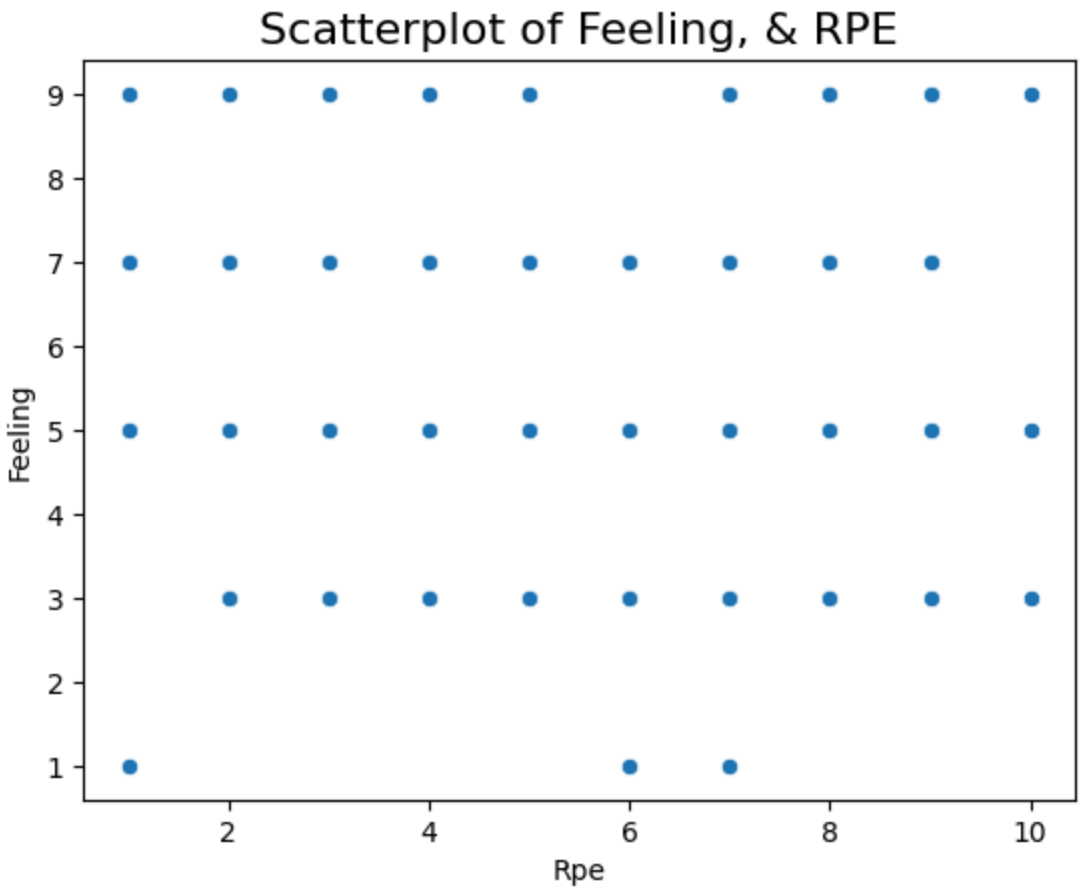
	Title	WorkoutType	WorkoutDescription	PlannedDuration	PlannedDistanceInMeters	Work
0	First intervals of the year! =D	Bike	Duration: 75 minutes Course: Flat or rolling h...	1.25	NaN	2 05T0
1	Zwift - Feb 5 & 7 Intervals	Bike	Duration: 75 minutes Course: Flat or rolling h...	1.25	NaN	2 07T0
2	Zwift - VoxTour Stage 3: Hannah Walker (E)	Bike	Duration: 90-120 minutes Course: Flat or rolli...	2.00	NaN	2 09T0
3	Cycling	Bike	Duration: 90-120 minutes Course: Flat or rolli...	1.50	NaN	2 10T0
4	1.5 Hour Ride	Bike	Duration: 90-120 minutes Course: Flat or rolli...	2.00	NaN	2 16T0

5 rows × 35 columns

In [62]:

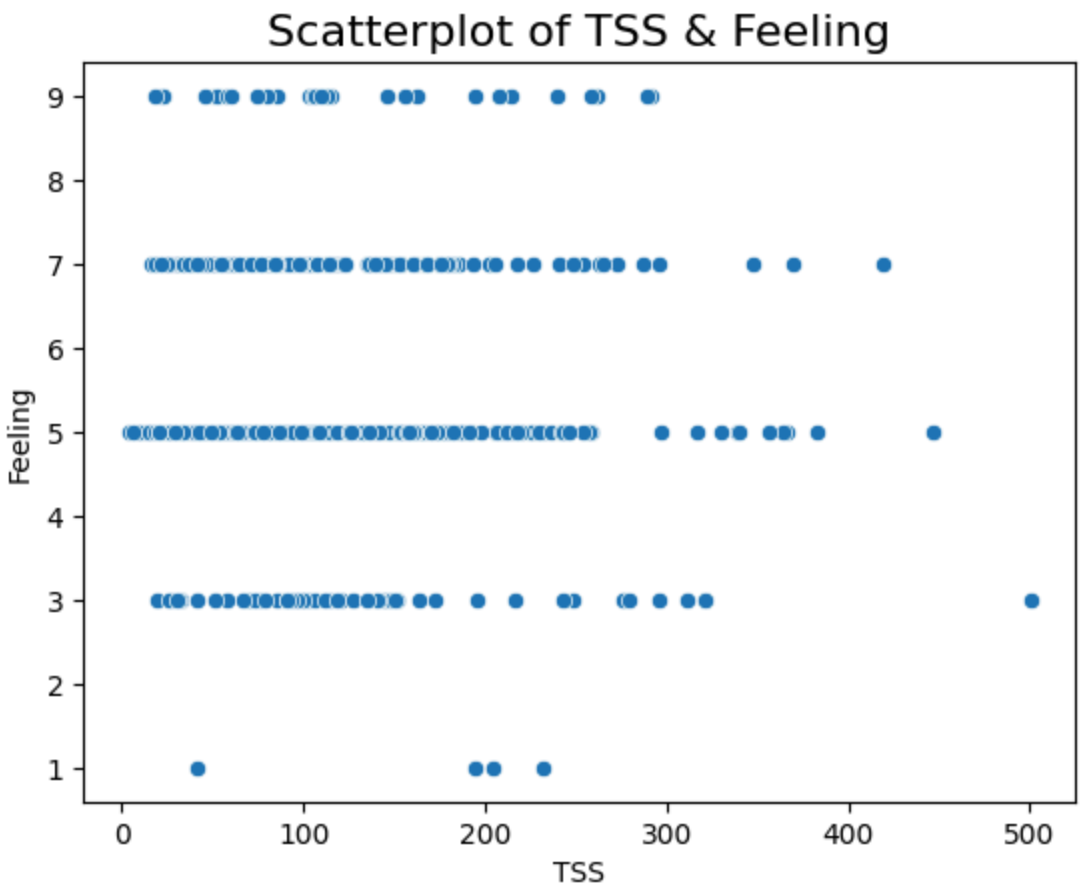
```
sns.scatterplot(data=result2, x='Rpe', y='Feeling')
plt.title('Scatterplot of Feeling, & RPE', fontsize=16)
plt.show()
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

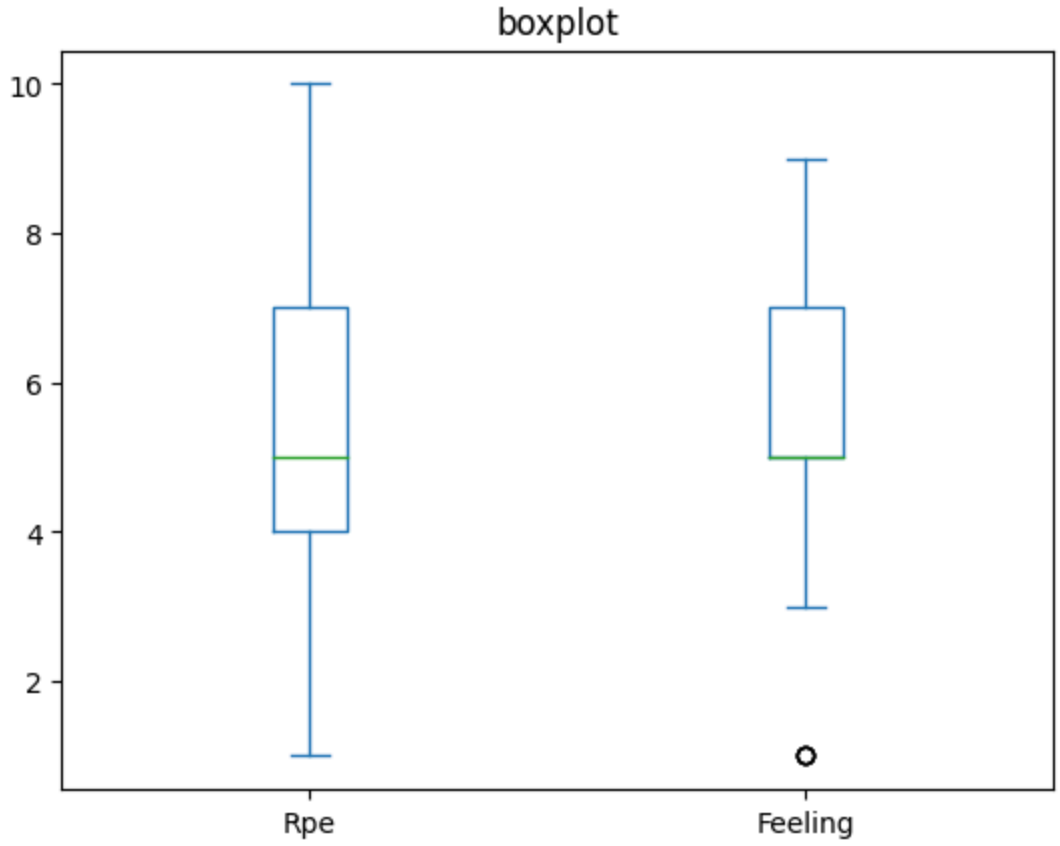


In [63]:
```
sns.scatterplot(data=result2, x='TSS', y='Feeling')
plt.title('Scatterplot of TSS & Feeling', fontsize=16)
plt.show()
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



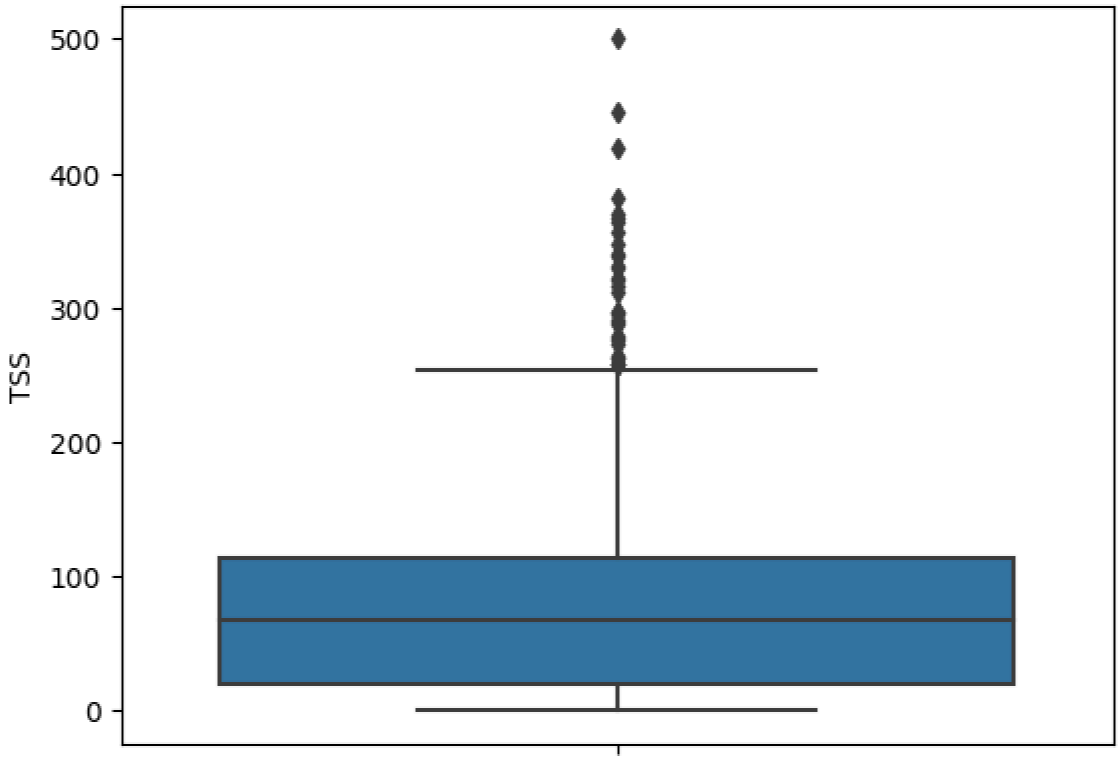
In [64]:
```
ax = result2[['Rpe', 'Feeling']].plot(kind='box', title='boxplot')
plt.show()
```



In [65]:
```
sns.boxplot(data=result2, y='TSS')
```

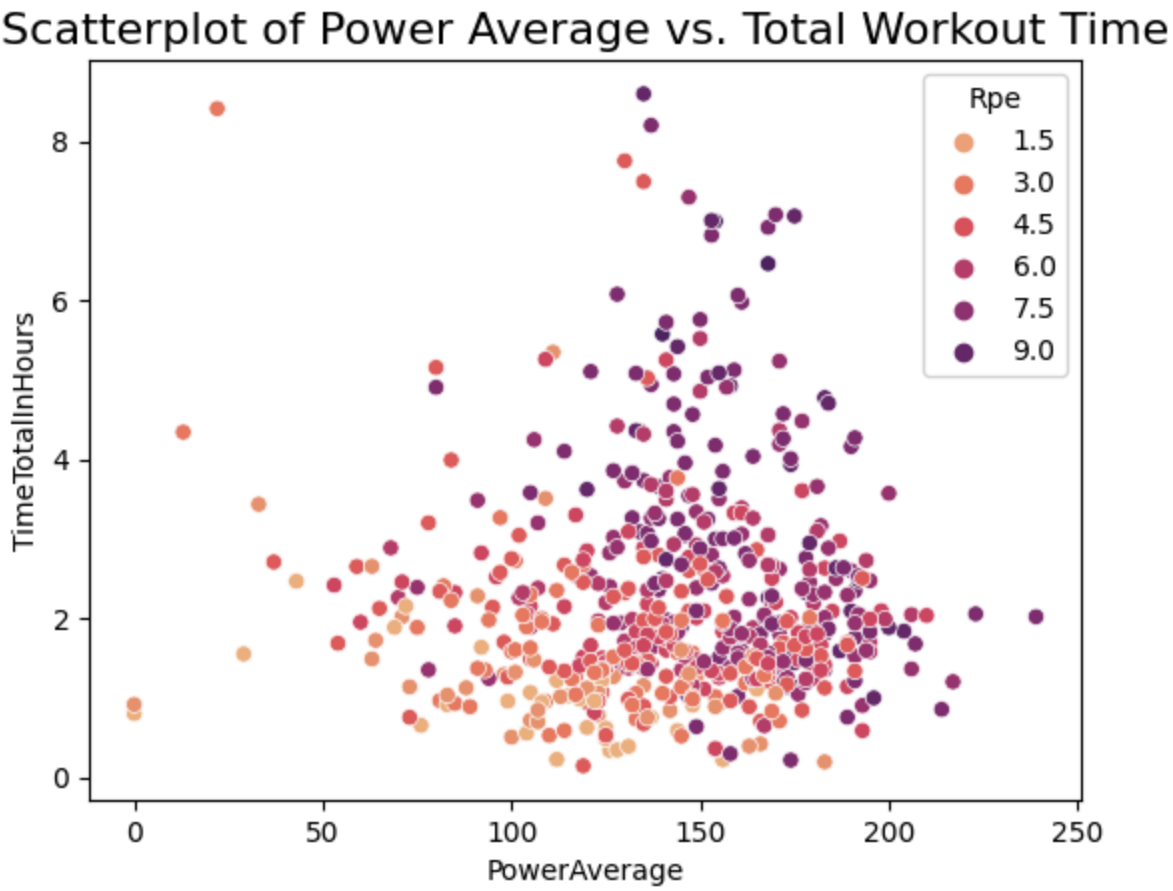
```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[65]: <Axes: ylabel='TSS'>



In [66]:
```
sns.scatterplot(data=result2, x='PowerAverage', y='TimeTotalInHours', hue='Rpe
plt.title('Scatterplot of Power Average vs. Total Workout Time', fontsize=16)
plt.show()
```

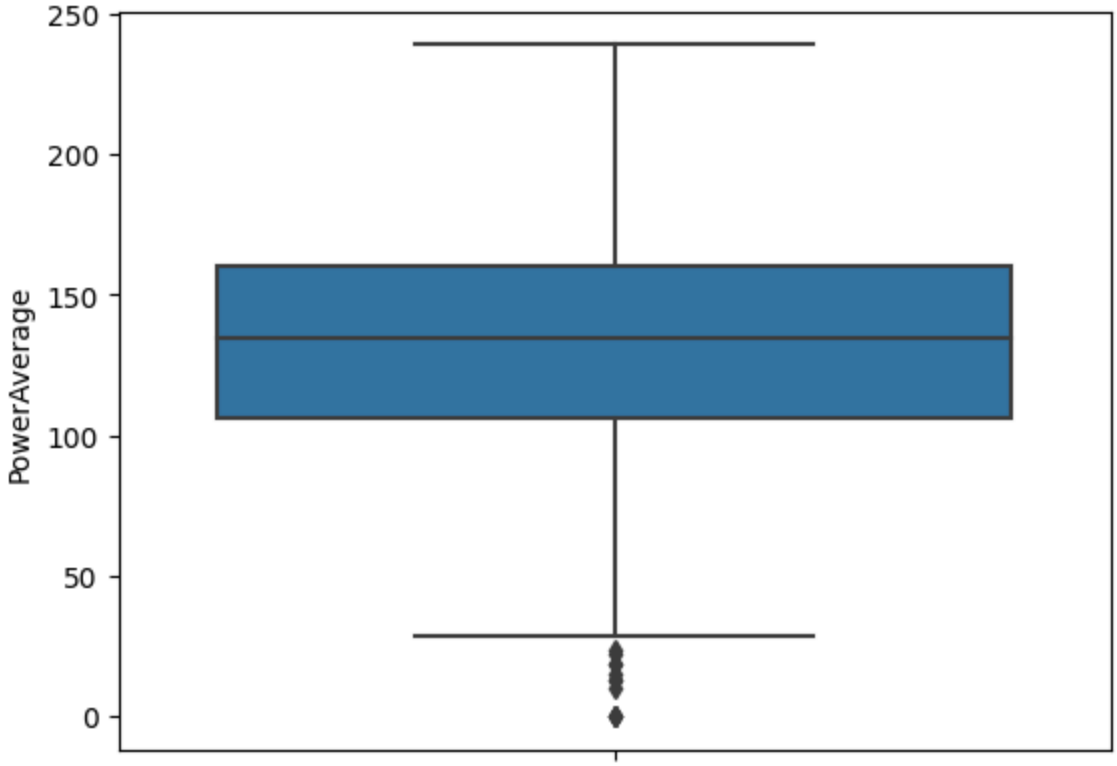
```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



In [67]:
```
sns.boxplot(data=result2, y='PowerAverage')
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

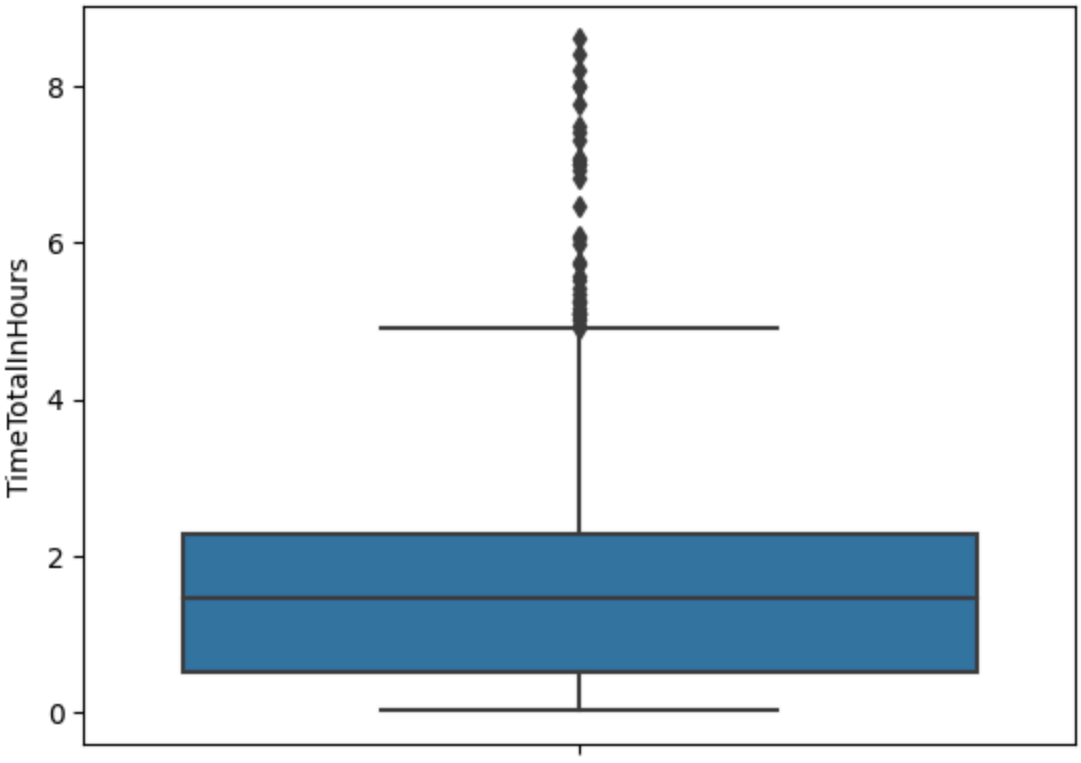
Out[67]:
```
<Axes: ylabel='PowerAverage'>
```



In [68]:
```
sns.boxplot(data=result2, y='TimeTotalInHours')
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.py:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[68]:
```
<Axes: ylabel='TimeTotalInHours'>
```

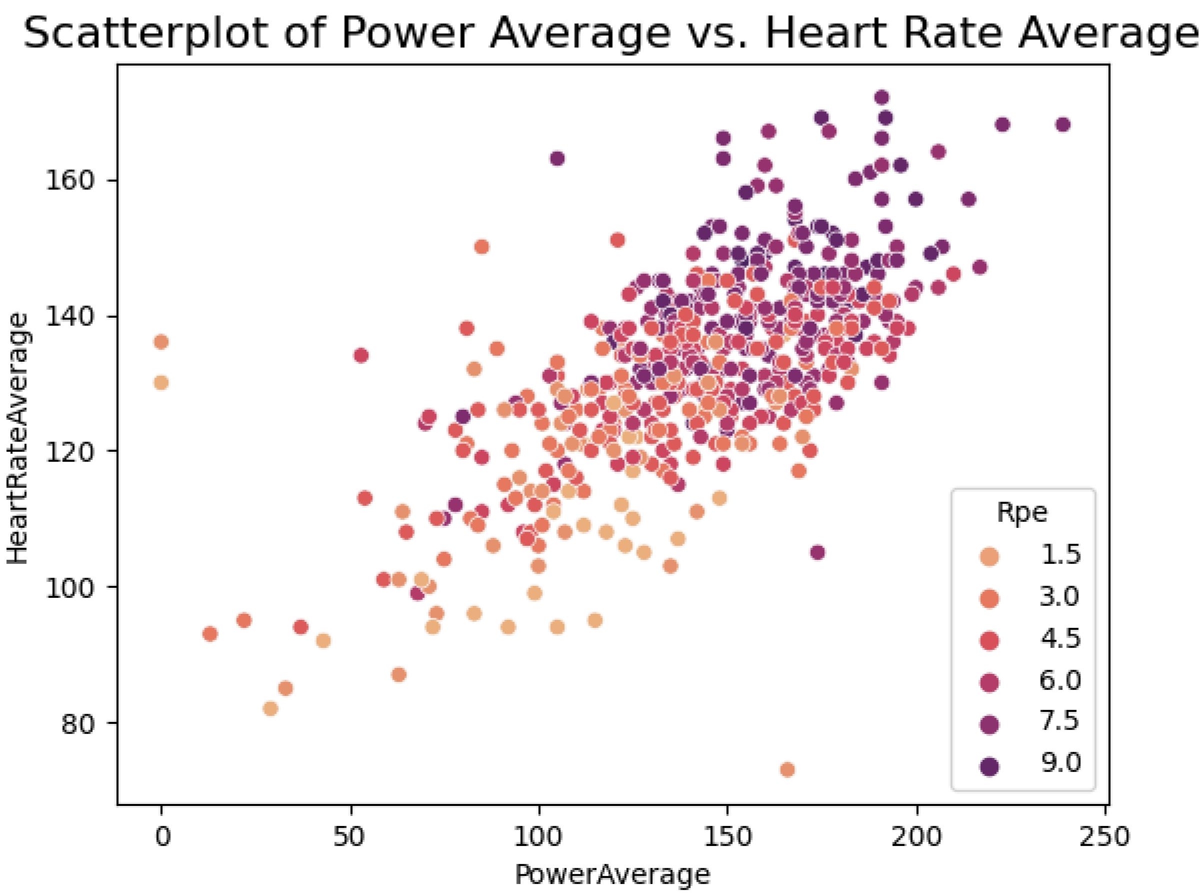


How does the subjective 'RPE' metric correlate to power and heart rate?

Does RPE increase linearly w/ average HR or Power output?

In [69]:
```
sns.scatterplot(data=result2, x='PowerAverage', y='HeartRateAverage', hue='Rpe
plt.title('Scatterplot of Power Average vs. Heart Rate Average', fontsize=16)
plt.show()
```

```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



Based on the above scatterplot, it does appear that as both average power and average heart rate increase, RPE increases.

Can we predict RPE based on duration, avg PWR, HR?

In [70]:
```
result2.Rpe.describe()
```

Out[70]:
```
count    628.000000
mean       5.253185
std        2.294416
min        1.000000
25%        4.000000
50%        5.000000
75%        7.000000
max       10.000000
Name: Rpe, dtype: float64
```

In [71]:
```
result2.describe()
```

Out[71]:

	PlannedDuration	PlannedDistanceInMeters	DistanceInMeters	PowerAverage	PowerMa
count	599.000000	45.0	1011.000000	892.000000	892.00000
mean	1.982015	0.0	25213.197613	131.110987	587.30829
std	1.091855	0.0	24115.943166	39.124567	236.03558
min	0.333333	0.0	0.000000	0.000000	0.00000
25%	1.416667	0.0	7224.285156	106.000000	441.50000
50%	1.500000	0.0	20301.810547	135.000000	593.50000
75%	2.500000	0.0	33675.865234	160.000000	699.25000
max	6.000000	0.0	163372.796875	239.000000	3209.00000

8 rows × 29 columns

In [72]:
```
result3 = result2[['PowerAverage', 'HeartRateAverage', 'TimeTotalInHours', 'Rp
```

In [73]:
```
result3.describe()
```

Out[73]:

	PowerAverage	HeartRateAverage	TimeTotalInHours	Rpe
count	892.000000	888.000000	1015.000000	628.000000
mean	131.110987	128.311937	1.670513	5.253185
std	39.124567	16.231367	1.417433	2.294416
min	0.000000	73.000000	0.027593	1.000000
25%	106.000000	119.000000	0.531354	4.000000
50%	135.000000	130.000000	1.470923	5.000000
75%	160.000000	139.000000	2.280475	7.000000
max	239.000000	172.000000	8.594491	10.000000

In [74]:
```
from pandas.api.types import CategoricalDtype

cat_type = CategoricalDtype(categories=[1, 2, 3, 4, 5, 6, 7, 8, 9], ordered=Tru

result3["Rpe"] = result3["Rpe"].astype(cat_type)
```

```
/var/folders/1p/n1yg43r91032z6dwjfvfltz00000gn/T/ipykernel_44300/3606035101.p
y:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  result3["Rpe"] = result3["Rpe"].astype(cat_type)
```

In [75]:
```
result3.isnull().sum()
```

Out[75]:
```
PowerAverage        136
HeartRateAverage    140
TimeTotalInHours     13
Rpe                 404
dtype: int64
```

In [76]:
```
#delete NA
result3.dropna(axis=0, inplace=True)
```

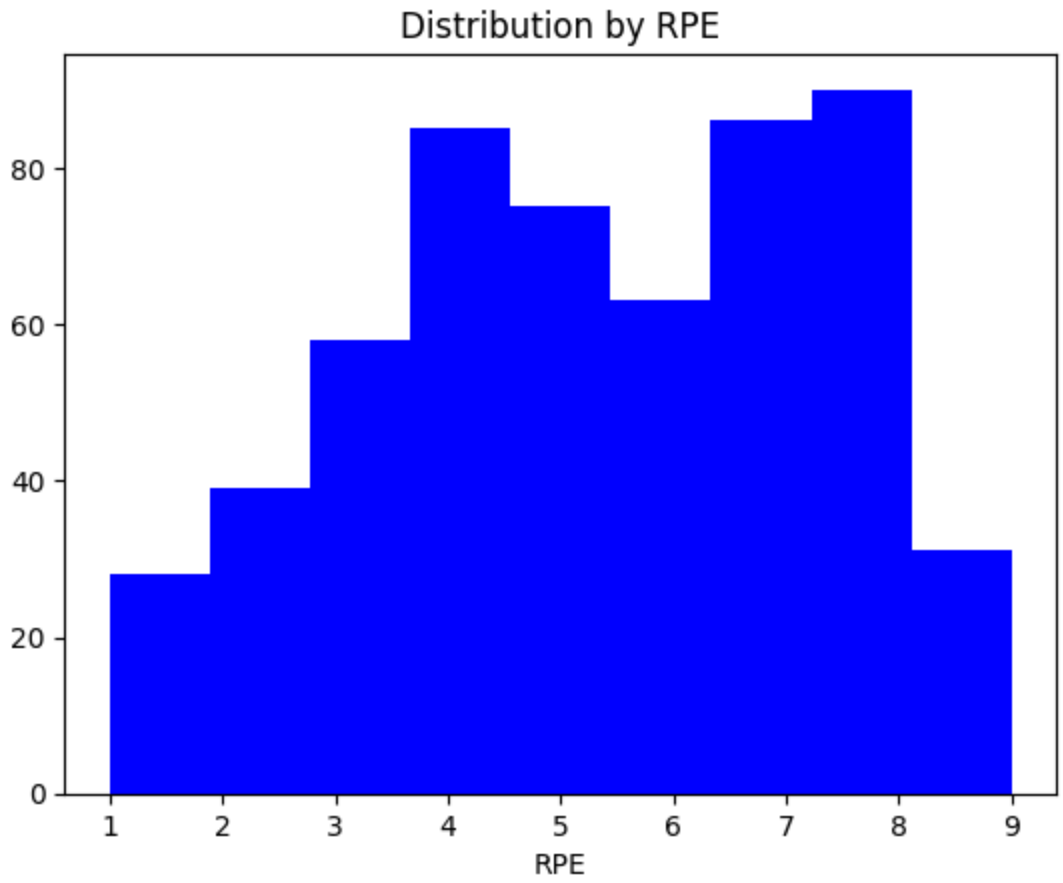
```
/var/folders/1p/n1yg43r91032z6dwjfvfltz00000gn/T/ipykernel_44300/3262627439.p
y:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  result3.dropna(axis=0, inplace=True)
```

In [77]:
```
result3.Rpe.describe()
```

Out[77]:
```
count     555
unique      9
top         8
freq       90
Name: Rpe, dtype: int64
```

In [78]:
```
plt.hist(result3['Rpe'],bins=9,color='b')
plt.xlabel('RPE')
plt.title('Distribution by RPE')
```

Out[78]:
```
Text(0.5, 1.0, 'Distribution by RPE')
```



In [79]:
```
from statsmodels.miscmodels.ordinal_model import OrderedModel

y = result3['Rpe']
X = result3[['PowerAverage', 'HeartRateAverage']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, rando

mod_probe = OrderedModel(y_train, X_train, distr='logit')
res_log = mod_probe.fit(method='bfgs')

predicted = res_log.model.predict(res_log.params, np.array(X_test))
preds = [np.argmax(x) for x in predicted]

res_log.summary()
```

```
Optimization terminated successfully.
         Current function value: 1.897649
         Iterations: 49
         Function evaluations: 54
         Gradient evaluations: 54
```

Out[79]:

OrderedModel Results

Dep. Variable:	Rpe	**Log-Likelihood:**	-631.92
Model:	OrderedModel	**AIC:**	1284.
Method:	Maximum Likelihood	**BIC:**	1322.
Date:	Thu, 07 Mar 2024		
Time:	10:45:33		
No. Observations:	333		
Df Residuals:	323		
Df Model:	2		

	coef	std err	z	P>|z|	[0.025	0.975]
PowerAverage	0.0057	0.004	1.603	0.109	-0.001	0.013
HeartRateAverage	0.0804	0.010	8.231	0.000	0.061	0.100
1/2	7.6266	1.058	7.210	0.000	5.553	9.700
2/3	0.1841	0.208	0.887	0.375	-0.223	0.591
3/4	0.1136	0.152	0.750	0.454	-0.183	0.411
4/5	-0.0021	0.126	-0.016	0.987	-0.250	0.246
5/6	-0.3550	0.141	-2.511	0.012	-0.632	-0.078
6/7	-0.5396	0.158	-3.424	0.001	-0.849	-0.231
7/8	-0.1643	0.138	-1.194	0.233	-0.434	0.105
8/9	0.6692	0.125	5.343	0.000	0.424	0.915

In [80]:
```
from sklearn import metrics

metrics.accuracy_score(y_test, preds)
```

Out[80]:
```
0.20270270270270271
```

In [81]:
```
from statsmodels.miscmodels.ordinal_model import OrderedModel

y = result3['Rpe']
X = result3[['PowerAverage', 'HeartRateAverage', 'TimeTotalInHours']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, rando

mod_probe = OrderedModel(y_train, X_train, distr='logit')
res_log = mod_probe.fit(method='bfgs')

predicted = res_log.model.predict(res_log.params, np.array(X_test))
preds = [np.argmax(x) for x in predicted]

res_log.summary()
```

```
Optimization terminated successfully.
         Current function value: 1.758149
         Iterations: 52
         Function evaluations: 57
         Gradient evaluations: 57
```

Out[81]:

OrderedModel Results

Dep. Variable:	Rpe	**Log-Likelihood:**	-585.46
Model:	OrderedModel	**AIC:**	1193.
Method:	Maximum Likelihood	**BIC:**	1235.
Date:	Thu, 07 Mar 2024		
Time:	10:45:35		
No. Observations:	333		
Df Residuals:	322		
Df Model:	3		

	coef	std err	z	P>|z|	[0.025	0.975]
PowerAverage	0.0121	0.004	3.406	0.001	0.005	0.019
HeartRateAverage	0.0804	0.010	8.159	0.000	0.061	0.100
TimeTotalInHours	0.7723	0.089	8.695	0.000	0.598	0.946
1/2	9.6937	1.112	8.720	0.000	7.515	11.873
2/3	0.3037	0.201	1.513	0.130	-0.090	0.697
3/4	0.2585	0.147	1.754	0.079	-0.030	0.547
4/5	0.1522	0.125	1.218	0.223	-0.093	0.397
5/6	-0.1903	0.141	-1.353	0.176	-0.466	0.085
6/7	-0.3715	0.157	-2.366	0.018	-0.679	-0.064
7/8	0.0148	0.137	0.108	0.914	-0.253	0.283
8/9	0.8158	0.123	6.648	0.000	0.575	1.056

In [82]:
```
from sklearn import metrics

metrics.accuracy_score(y_test, preds)
```

Out[82]:
```
0.2072072072072072
```

Because RPE is an ordered categorical variable, I used the ordinal model feature to do a multinomial logistic regression of sorts, where first, just average power and average heart rate were used to predict RPE, and then, with a second model, added the total ride time.

Though the accuracy metrics of both models are equally poor, the Log-Likelihood of the model with the total ride time is less, indicating some improvement in the model.

In [83]:
```
#more basic regression model

import statsmodels.api as sm

X = result3[['PowerAverage', 'HeartRateAverage', 'TimeTotalInHours']]
y = result3['Rpe']

# Add constant to the model
```

```
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print summary statistics
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Rpe   R-squared:                       0.554
Model:                            OLS   Adj. R-squared:                  0.551
Method:                 Least Squares   F-statistic:                     227.9
Date:                Thu, 07 Mar 2024   Prob (F-statistic):           4.01e-96
Time:                        10:45:36   Log-Likelihood:                -1008.5
No. Observations:                 555   AIC:                             2025.
Df Residuals:                     551   BIC:                             2042.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -7.5271      0.612    -12.296      0.000      -8.730      -6.325
PowerAverage   0.0110      0.002      4.525      0.000       0.006       0.016
HeartRateAverage     0.0751      0.006     12.587      0.000       0.063       0.087
TimeTotalInHours     0.6095      0.048     12.760      0.000       0.516       0.703
==============================================================================
Omnibus:                        1.449   Durbin-Watson:                   1.556
Prob(Omnibus):                  0.485   Jarque-Bera (JB):                1.536
Skew:                           0.103   Prob(JB):                        0.464
Kurtosis:                       2.844   Cond. No.                     1.91e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.91e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Here, the R-squared (and Adjusted R-squared) are close to .55, which indicate that 55% of the variability in RPE can be explained by workout length, average heart rate, and average power.

The F-statistic is 227.9 with a low p-value, which would indicate that the model as a whole is statistically significant.

When controlling for time and heart rate, power average is statistically significant and positively correlated with RPE. Heart Rate, and Total Time, however are not statistically significant when controlling for the other variables.

Do workouts with average power/ heart rate over tempo consistently have higher rpe?

In [86]:
```
# Create binary variables for average heart rate over 150 and average power ov
result3['heart_rate_over_150'] = (result3['HeartRateAverage'] > 150).astype(in
result3['power_over_220'] = (result3['PowerAverage'] > 220).astype(int)

# Create the regression model
X = result3[['heart_rate_over_150', 'power_over_220', 'TimeTotalInHours']]
y = result3['Rpe']

# Add constant to the model
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print summary statistics
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Rpe   R-squared:                       0.252
Model:                            OLS   Adj. R-squared:                  0.248
Method:                 Least Squares   F-statistic:                     61.82
Date:                Thu, 07 Mar 2024   Prob (F-statistic):           1.83e-34
Time:                        10:50:18   Log-Likelihood:                -1151.9
No. Observations:                 555   AIC:                             2312.
Df Residuals:                     551   BIC:                             2329.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.7462      0.158     23.691      0.000       3.436       4.057
heart_rate_over_150     2.2012      0.320      6.871      0.000       1.572       2.830
power_over_220     0.7276      1.403      0.519      0.604      -2.028       3.483
TimeTotalInHours     0.6505      0.061     10.609      0.000       0.530       0.771
==============================================================================
Omnibus:                        8.540   Durbin-Watson:                   1.562
Prob(Omnibus):                  0.014   Jarque-Bera (JB):                5.663
Skew:                          -0.088   Prob(JB):                       0.0589
Kurtosis:                       2.538   Cond. No.                         46.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
/var/folders/1p/n1yg43r91032z6dwjfvfltz00000gn/T/ipykernel_44300/2411831300.p
y:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  result3['heart_rate_over_150'] = (result3['HeartRateAverage'] > 150).astype
(int)
/var/folders/1p/n1yg43r91032z6dwjfvfltz00000gn/T/ipykernel_44300/2411831300.p
y:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  result3['power_over_220'] = (result3['PowerAverage'] > 220).astype(int)
```

Interestingly, this model has a lower R2 and explains only 25% of the variance in Rpe. It also has no statistically significant variables.

In [87]:
```
result3['power_over_220'].value_counts()
```

Out[87]:
```
power_over_220
0    553
1      2
Name: count, dtype: int64
```

In [88]:
```
result3['heart_rate_over_150'].value_counts()
```

Out[88]:
```
heart_rate_over_150
0    513
1     42
Name: count, dtype: int64
```

Again, we have a small data problem...perhaps repeat with 'endurance' ranges?

In [90]:
```
# Create binary variables for average heart rate over 150 and average power ov
result3['heart_rate_over_140'] = (result3['HeartRateAverage'] > 140).astype(in
result3['power_over_160'] = (result3['PowerAverage'] > 160).astype(int)

# Create the regression model
X = result3[['heart_rate_over_140', 'power_over_160', 'TimeTotalInHours']]
y = result3['Rpe']

# Add constant to the model
X = sm.add_constant(X)

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print summary statistics
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Rpe   R-squared:                       0.400
Model:                            OLS   Adj. R-squared:                  0.396
Method:                 Least Squares   F-statistic:                     122.2
Date:                Thu, 07 Mar 2024   Prob (F-statistic):           1.10e-60
Time:                        10:53:25   Log-Likelihood:                -1090.8
No. Observations:                 555   AIC:                             2190.
Df Residuals:                     551   BIC:                             2207.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.0769      0.154     19.920      0.000       2.774       3.380
heart_rate_over_140     1.7711      0.177      9.987      0.000       1.423       2.119
power_over_160     0.9367      0.168      5.575      0.000       0.607       1.267
TimeTotalInHours     0.6618      0.055     11.992      0.000       0.553       0.770
==============================================================================
Omnibus:                        1.805   Durbin-Watson:                   1.556
Prob(Omnibus):                  0.406   Jarque-Bera (JB):                1.679
Skew:                           0.044   Prob(JB):                        0.432
Kurtosis:                       2.745   Cond. No.                         7.76
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
/var/folders/1p/n1yg43r91032z6dwjfvfltz00000gn/T/ipykernel_44300/4050507655.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  result3['heart_rate_over_140'] = (result3['HeartRateAverage'] > 140).astype(int)
/var/folders/1p/n1yg43r91032z6dwjfvfltz00000gn/T/ipykernel_44300/4050507655.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  result3['power_over_160'] = (result3['PowerAverage'] > 160).astype(int)
```

In [91]:
```
result3['power_over_160'].value_counts()
```

Out[91]:
```
power_over_160
0    368
1    187
Name: count, dtype: int64
```

In [92]:
```
result3['heart_rate_over_140'].value_counts()
```

Out[92]:
```
heart_rate_over_140
0    399
1    156
Name: count, dtype: int64
```

Better data balance, but no statistically significant results.

Are workouts following rest days consistently higher or lower rpe? (Control for intensity using tss?)

In [39]:
```
query = """
    SELECT Title, WorkoutType, WorkoutDescription, PlannedDuration, WorkoutDay
    FROM data
    WHERE WorkoutType = 'Bike'OR WorkoutType = 'MTB' OR WorkoutType = 'Day Off
    """

result4 = pd.read_sql_query(query, conn)
result4.head(10)
```

Out[39]:

	Title	WorkoutType	WorkoutDescription	PlannedDuration	WorkoutDay	TimeTotalInHou
0	Rest Day	Day Off	There was some real intensity in your sessions...	NaN	2018-01-01T00:00:00	Na
1	Rest Day	Day Off	Don't be tempted to squeeze and additional ses...	NaN	2018-01-02T00:00:00	Na
2	10-Mile Time Trial or Under/Over Intervals	Bike	Time: 50 mins - 1 hr 10 mins Warm-up: http://b...	1.166667	2018-01-03T00:00:00	Na
3	Rest Day	Day Off	If you have been short of time during this tra...	NaN	2018-01-04T00:00:00	Na
4	Rest Day	Day Off	This is an important session. Don't skip recov...	NaN	2018-01-05T00:00:00	Na
5	Recovery ride	Bike	Warm-up: Ease yourself gently into today's ses...	2.000000	2018-01-06T00:00:00	Na
6	None	Bike	Time: 2 hr 30 mins - 3 hrs Warm-up: WU R from ...	3.000000	2018-01-07T00:00:00	Na
7	Cycling	Bike	None	NaN	2018-02-14T00:00:00	0.6997
8	Cycling	Bike	None	NaN	2018-02-25T00:00:00	0.0483
9	Cycling	Bike	None	NaN	2018-02-26T00:00:00	4.1241

In [43]:
```
result4['follows_rest_day'] = (result4['WorkoutType'].shift(1) == 'Day Off').as
```

In [50]:
```
#drop if RPE == NaN
result4 = result4.dropna(subset=['Rpe'])

#drop if TSS == NaN
result4 = result4.dropna(subset=['TSS'])
```

In [51]:
```
import statsmodels.api as sm

#regression model to predict 'rpe' based on 'follows_rest_day' and 'tss'
X = result4[['follows_rest_day', 'TSS']]
y = result4['Rpe']

#add constant to the model
X = sm.add_constant(X)

#fit the regression model
model = sm.OLS(y, X).fit()

#print summary statistics
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    Rpe   R-squared:                       0.300
Model:                            OLS   Adj. R-squared:                  0.299
Method:                 Least Squares   F-statistic:                     249.5
Date:                Thu, 07 Mar 2024   Prob (F-statistic):           6.97e-91
Time:                        10:23:57   Log-Likelihood:                -2409.7
No. Observations:                1166   AIC:                             4825.
Df Residuals:                    1163   BIC:                             4841.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.6459      0.097     37.750      0.000       3.456       3.835
follows_rest_day    -0.5340      0.783     -0.682      0.496      -2.071       1.003
TSS            0.0170      0.001     22.308      0.000       0.016       0.019
==============================================================================
Omnibus:                       22.823   Durbin-Watson:                   1.857
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               12.735
Skew:                          -0.014   Prob(JB):                      0.00172
Kurtosis:                       2.489   Cond. No.                     1.77e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.77e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

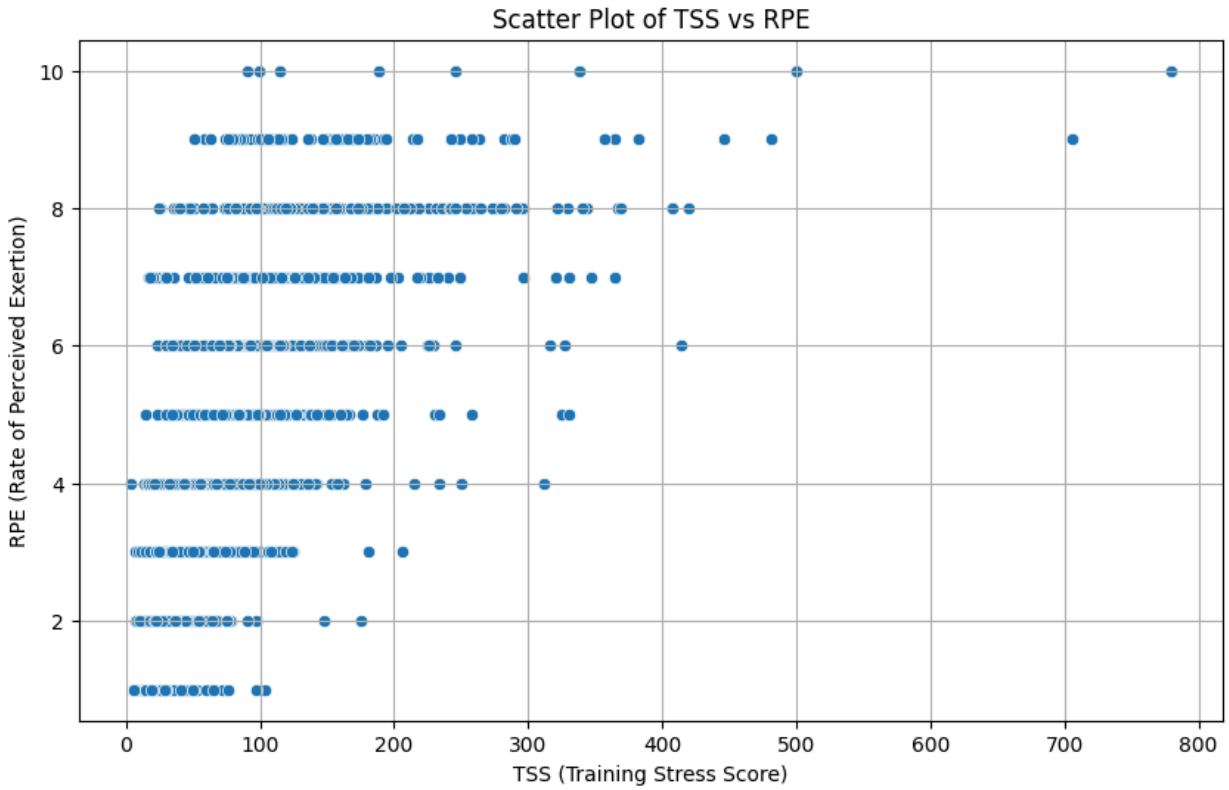
The F-statistisc of 249.4 with a low p-value indicates that the model as a whole is statistically significant. With an R-squared (and Adjusted R-squared) of .3, we can say that approximately 30% of the variability in RPE can be explained by the variables in the model.

More specifically, the coefficient of 'follows_rest_day' has a p-value close to .5, which is not statistically significant, indicating that when controlling for TSS, that whether a workout follows a rest day or not is not a significant indicator of RPE. On the other hand, TSS, when controlling for 'follows_rest_day', does seem to be positively correlated with 'Rpe.'

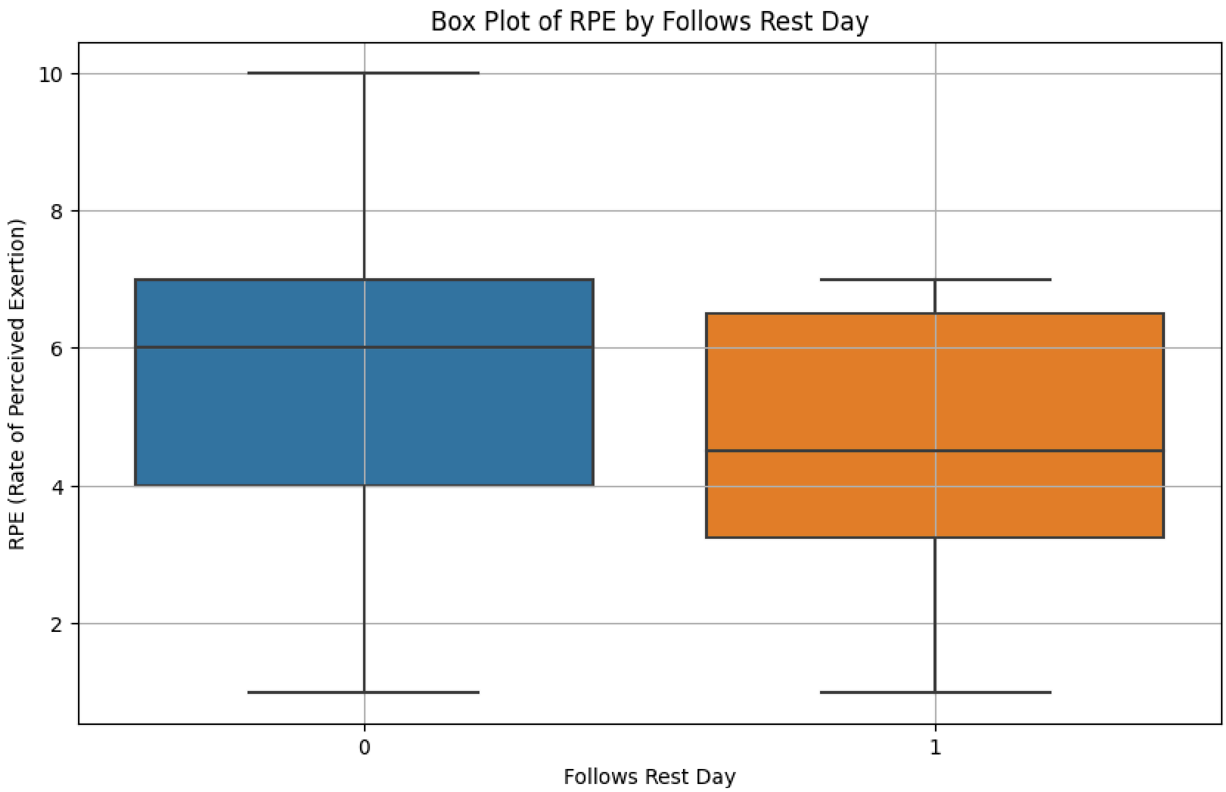
In [56]:
```
# Scatter plot of 'TSS' vs. 'Rpe'
plt.figure(figsize=(10, 6))
sns.scatterplot(x='TSS', y='Rpe', data=result4)
plt.title('Scatter Plot of TSS vs RPE')
plt.xlabel('TSS (Training Stress Score)')
plt.ylabel('RPE (Rate of Perceived Exertion)')
plt.grid(True)
plt.show()

# Box plot of 'Rpe' grouped by 'follows_rest_day'
plt.figure(figsize=(10, 6))
```

```
sns.boxplot(x='follows_rest_day', y='Rpe', data=result4)
plt.title('Box Plot of RPE by Follows Rest Day')
plt.xlabel('Follows Rest Day')
plt.ylabel('RPE (Rate of Perceived Exertion)')
plt.grid(True)
plt.show()

# Regression plot of 'TSS' vs. 'Rpe' with hue='follows_rest_day'
sns.lmplot(x='TSS', y='Rpe', data=result4, hue='follows_rest_day', scatter_kws=
plt.title('Regression Plot of TSS vs RPE with Follows Rest Day')
plt.xlabel('TSS (Training Stress Score)')
plt.ylabel('RPE (Rate of Perceived Exertion')
plt.grid(True)
plt.legend(title='Follows Rest Day', loc='upper left')
plt.show()
```

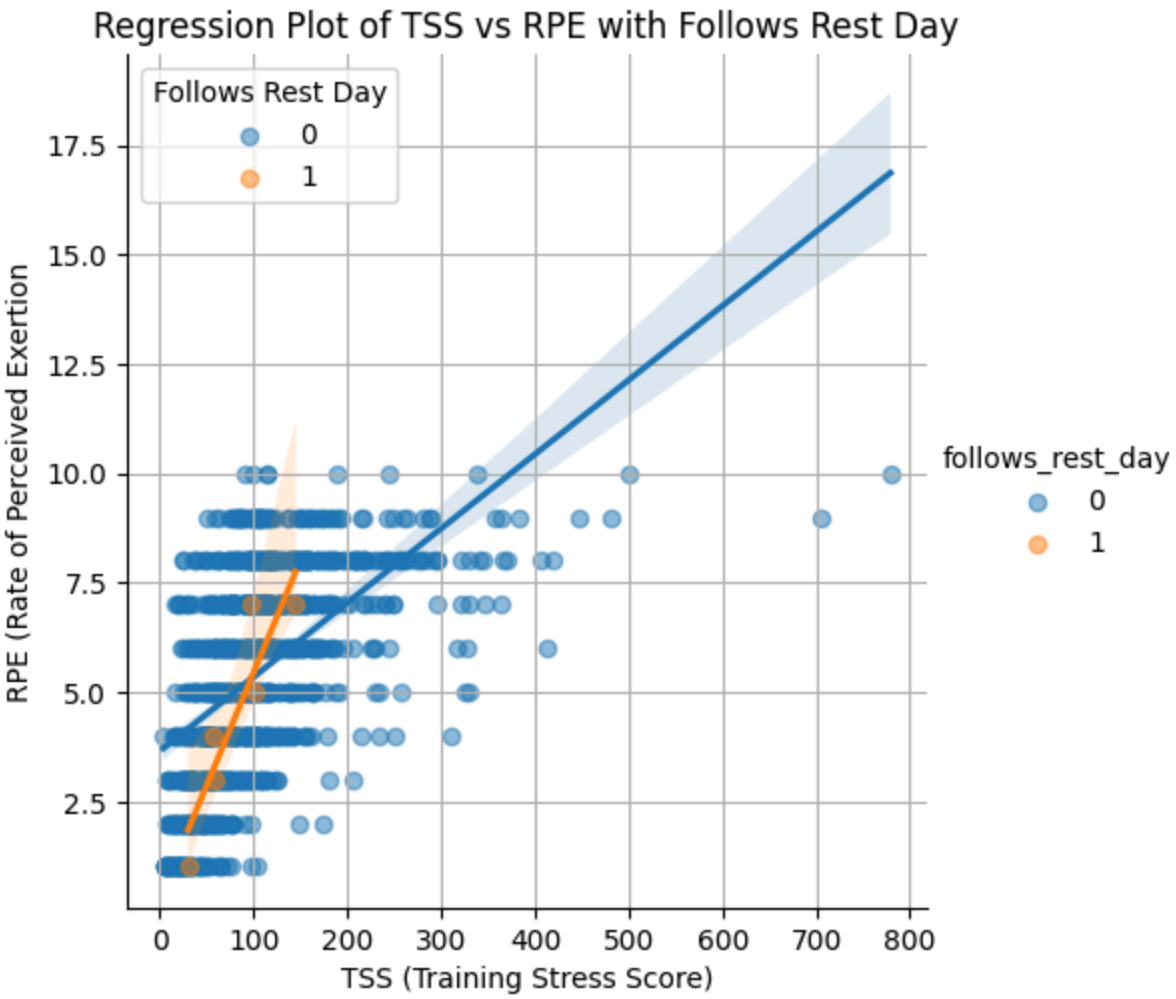
```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```



```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```




```
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/_oldcore.p
y:1498: FutureWarning: is_categorical_dtype is deprecated and will be removed
in a future version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
/Users/abigailsnyder/anaconda3/lib/python3.10/site-packages/seaborn/axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [ ]:
```
#1 in 'follows rest day' indcates yes, it follows a rest day
```

In [59]:
```
result4['follows_rest_day'].value_counts()
```

Out[59]:
```
follows_rest_day
0    1160
1       6
Name: count, dtype: int64
```

We have a small data problem here... which makes the model unreliable, as only 6 efforts that have both TSS and RPE ratings 'follow a rest day.'

In [ ]:
```

```