- Created By : Blessy Louis
- Created On: 28.07.2024

Task

Analyze wearable device data to correlate physical activity metrics with cardiovascular health outcomes. Utilize machine learning for predictive modeling and longitudinal analysis to identify trends over time. Provide personalized health recommendations based on findings to optimize cardiovascular fitness and overall well-being.

Import Libraries

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

The dataset contains information collected from wearable devices, capturing various aspects of physical activity and sedentary behavior over different days. Here's a breakdown of each column in the dataset:

1. **Id**: A unique identifier for each user or record. This helps distinguish data from different users or instances.
2. **ActivityDate**: The date on which the activity data was recorded. This allows for the analysis of trends over time.
3. **TotalSteps**: The total number of steps taken by the user on that particular day. This is a basic measure of daily activity.
4. **TotalDistance**: The total distance (typically in miles or kilometers) covered by the user on that day. This encompasses all types of movement, not just steps.
5. **TrackerDistance**: The distance recorded by the wearable device's tracker. This may differ slightly from TotalDistance due to differences in data sources or measurement methods.

6. **LoggedActivitiesDistance**: The distance covered during specific logged activities. Users might manually log activities like running or cycling, which is recorded here.
7. **VeryActiveDistance**: The distance covered during very active or high-intensity activities. This reflects the more strenuous part of the user's daily activity.
8. **ModeratelyActiveDistance**: The distance covered during moderately active or medium-intensity activities. This represents a mid-level of exertion.
9. **LightActiveDistance**: The distance covered during light activities. This could include casual walking or other low-intensity movements.
10. **SedentaryActiveDistance**: The distance (if any) recorded while being sedentary. This value is typically low or zero, as sedentary behavior involves minimal movement.
11. **VeryActiveMinutes**: The total minutes spent in very active or high-intensity activities throughout the day. This helps assess the intensity of the user's activity.
12. **FairlyActiveMinutes**: The total minutes spent in fairly active or moderate-intensity activities. This provides insight into the user's moderate activity level.
13. **LightlyActiveMinutes**: The total minutes spent in light activity throughout the day. This includes activities that are not very demanding but still contribute to overall movement.
14. **SedentaryMinutes**: The total minutes spent being sedentary or inactive. High values here indicate long periods of inactivity, which can have health implications.
15. **Calories**: The total number of calories burned by the user on that day. This is a key measure of energy expenditure and is influenced by all the other activity metrics.

Overall Dataset Purpose

The dataset aims to capture a comprehensive picture of an individual's daily physical activity and sedentary behavior. By analyzing this data, one can gain insights into patterns and trends that relate to overall health and fitness. For instance, understanding how different types of activities (very active, fairly active, light) contribute to calorie expenditure can help in creating personalized fitness and health recommendations.

Load Dataset

In [2]:
```
data = pd.read_csv('/content/dailyActivity_merged.csv')
```

In [3]:
```
# Display the first few rows of the dataset
data.head()
```

Out[3]:

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance	LightAct
0	1503960366	4/12/2016	13162	8.50	8.50	0.0	1.88	0.55	
1	1503960366	4/13/2016	10735	6.97	6.97	0.0	1.57	0.69	
2	1503960366	4/14/2016	10460	6.74	6.74	0.0	2.44	0.40	
3	1503960366	4/15/2016	9762	6.28	6.28	0.0	2.14	1.26	
4	1503960366	4/16/2016	12669	8.16	8.16	0.0	2.71	0.41	

In [4]:
```
# Check for missing values
print(data.isnull().sum())
```

```
Id                          0
ActivityDate                0
TotalSteps                  0
TotalDistance               0
TrackerDistance             0
LoggedActivitiesDistance    0
VeryActiveDistance          0
ModeratelyActiveDistance    0
LightActiveDistance         0
SedentaryActiveDistance     0
VeryActiveMinutes           0
FairlyActiveMinutes         0
LightlyActiveMinutes        0
SedentaryMinutes            0
Calories                    0
dtype: int64
```

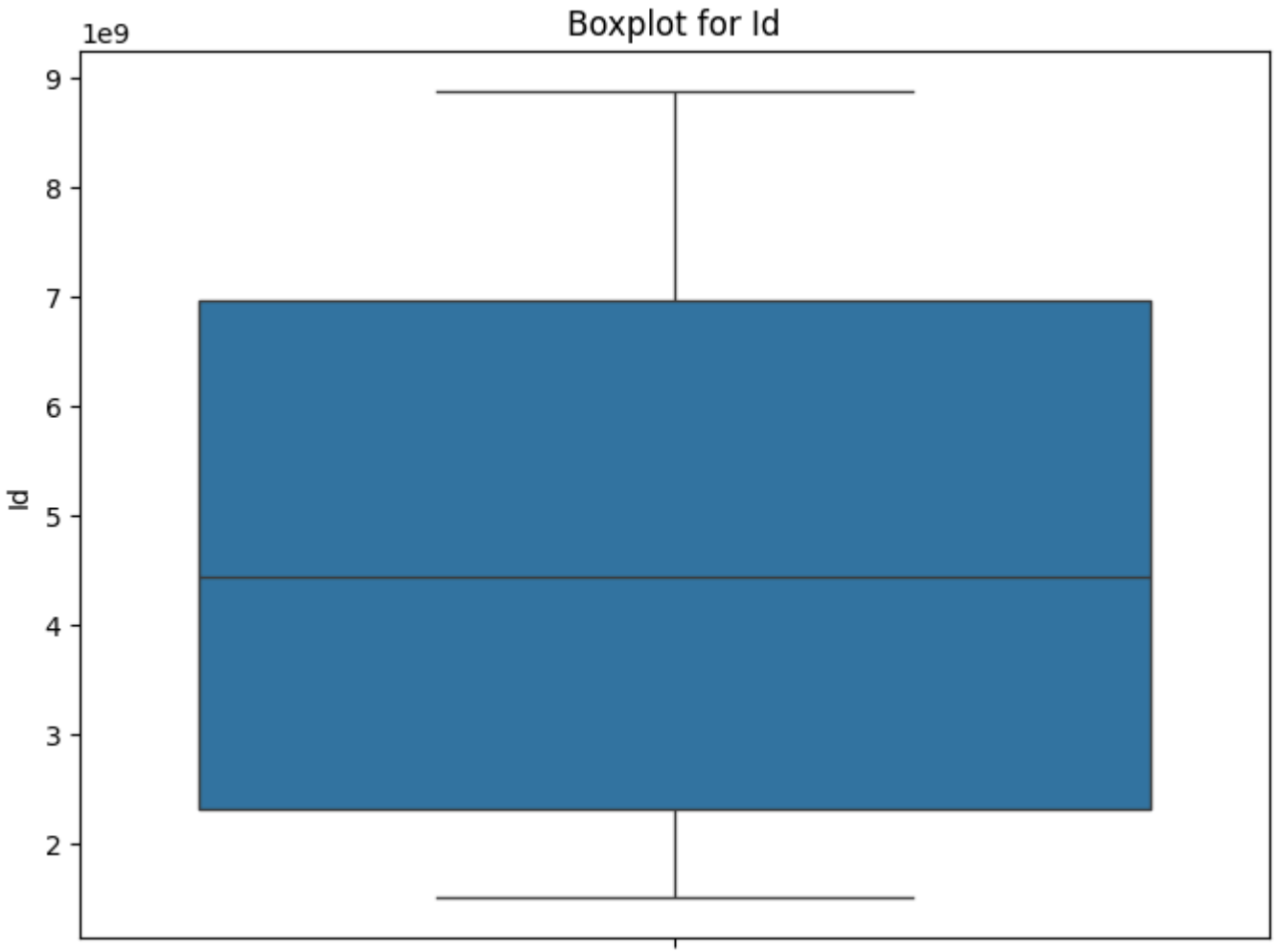
In [5]:
```
# Convert 'ActivityDate' to datetime
data['ActivityDate'] = pd.to_datetime(data['ActivityDate'])
```

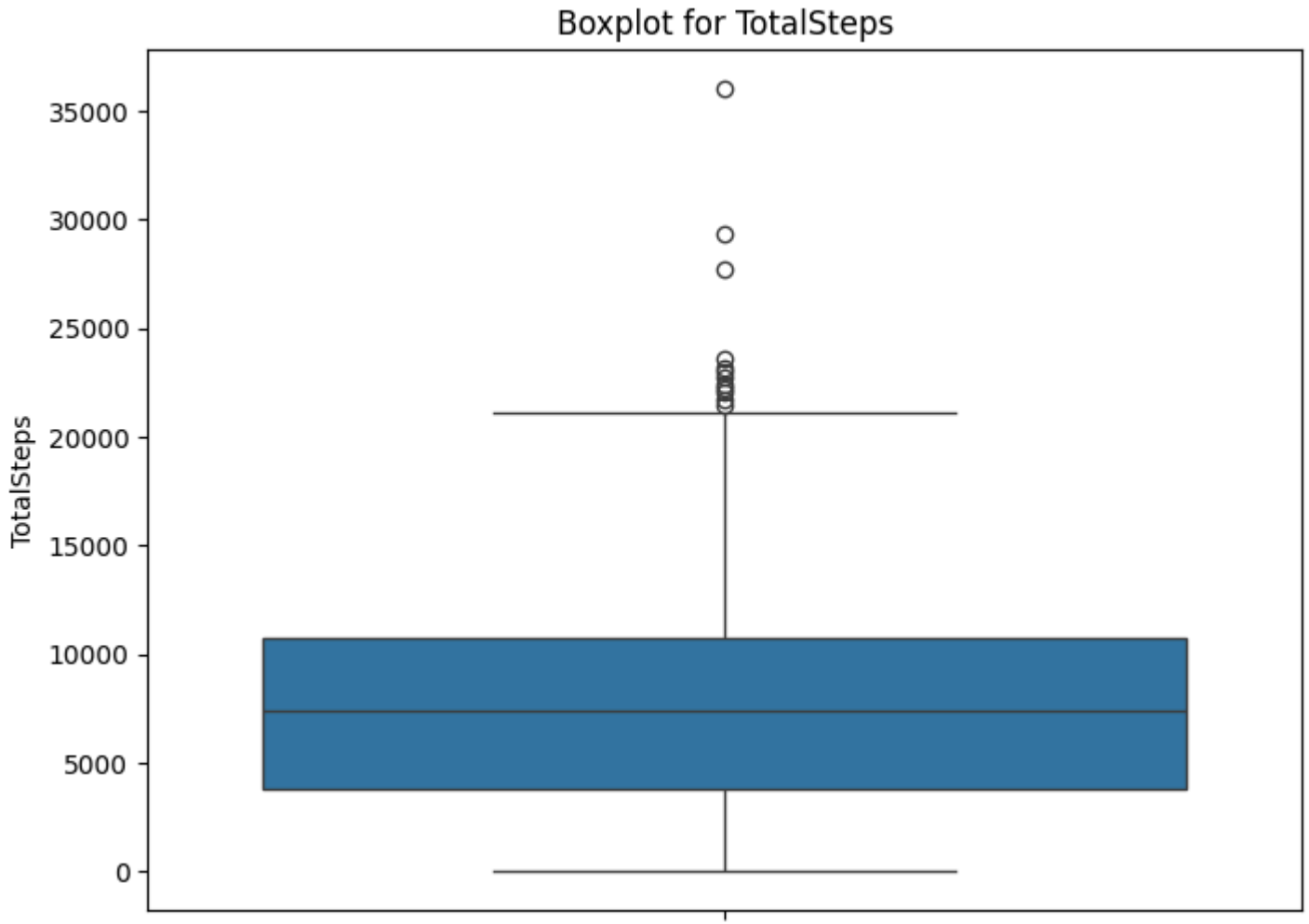
In [6]:
```
# Summary statistics
print(data.describe())
```

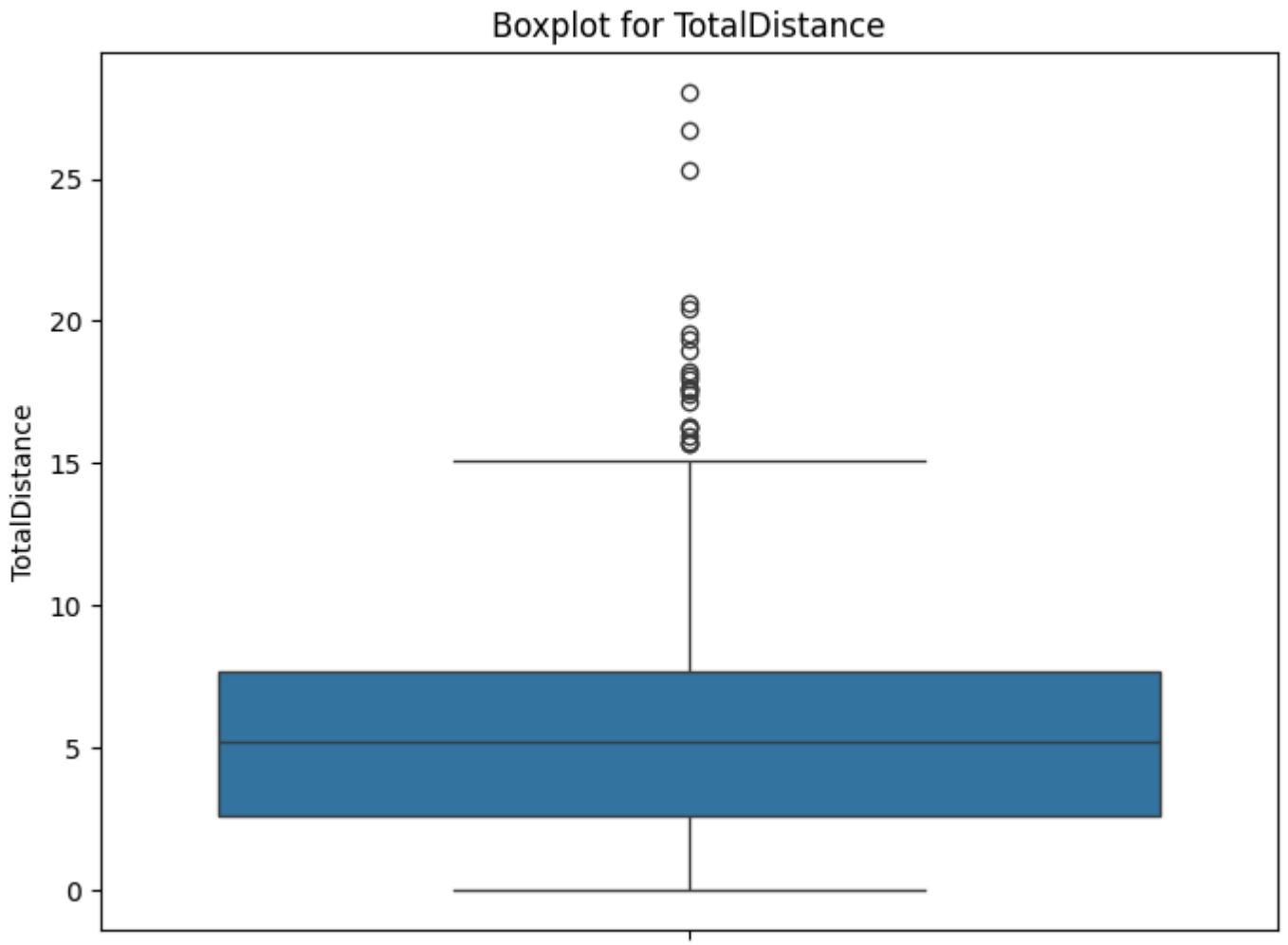
```
                 Id                    ActivityDate    TotalSteps  \
count  9.400000e+02                             940    940.000000
mean   4.855407e+09  2016-04-26 06:53:37.021276672   7637.910638
min    1.503960e+09            2016-04-12 00:00:00      0.000000
25%    2.320127e+09            2016-04-19 00:00:00   3789.750000
50%    4.445115e+09            2016-04-26 00:00:00   7405.500000
75%    6.962181e+09            2016-05-04 00:00:00  10727.000000
max    8.877689e+09            2016-05-12 00:00:00  36019.000000
std    2.424805e+09                            NaN   5087.150742

       TotalDistance  TrackerDistance  LoggedActivitiesDistance  \
count     940.000000       940.000000                940.000000
mean        5.489702         5.475351                  0.108171
min         0.000000         0.000000                  0.000000
25%         2.620000         2.620000                  0.000000
50%         5.245000         5.245000                  0.000000
75%         7.712500         7.710000                  0.000000
max        28.030001        28.030001                  4.942142
std         3.924606         3.907276                  0.619897

       VeryActiveDistance  ModeratelyActiveDistance  LightActiveDistance  \
count          940.000000                940.000000           940.000000
mean             1.502681                  0.567543             3.340819
min              0.000000                  0.000000             0.000000
25%              0.000000                  0.000000             1.945000
50%              0.210000                  0.240000             3.365000
75%              2.052500                  0.800000             4.782500
max             21.920000                  6.480000            10.710000
std              2.658941                  0.883580             2.040655

       SedentaryActiveDistance  VeryActiveMinutes  FairlyActiveMinutes  \
count               940.000000         940.000000           940.000000
mean                  0.001606          21.164894            13.564894
min                   0.000000           0.000000             0.000000
25%                   0.000000           0.000000             0.000000
50%                   0.000000           4.000000             6.000000
75%                   0.000000          32.000000            19.000000
max                   0.110000         210.000000           143.000000
std                   0.007346          32.844803            19.987404

       LightlyActiveMinutes  SedentaryMinutes     Calories
count            940.000000        940.000000   940.000000
mean             192.812766        991.210638  2303.609574
min                0.000000          0.000000     0.000000
```

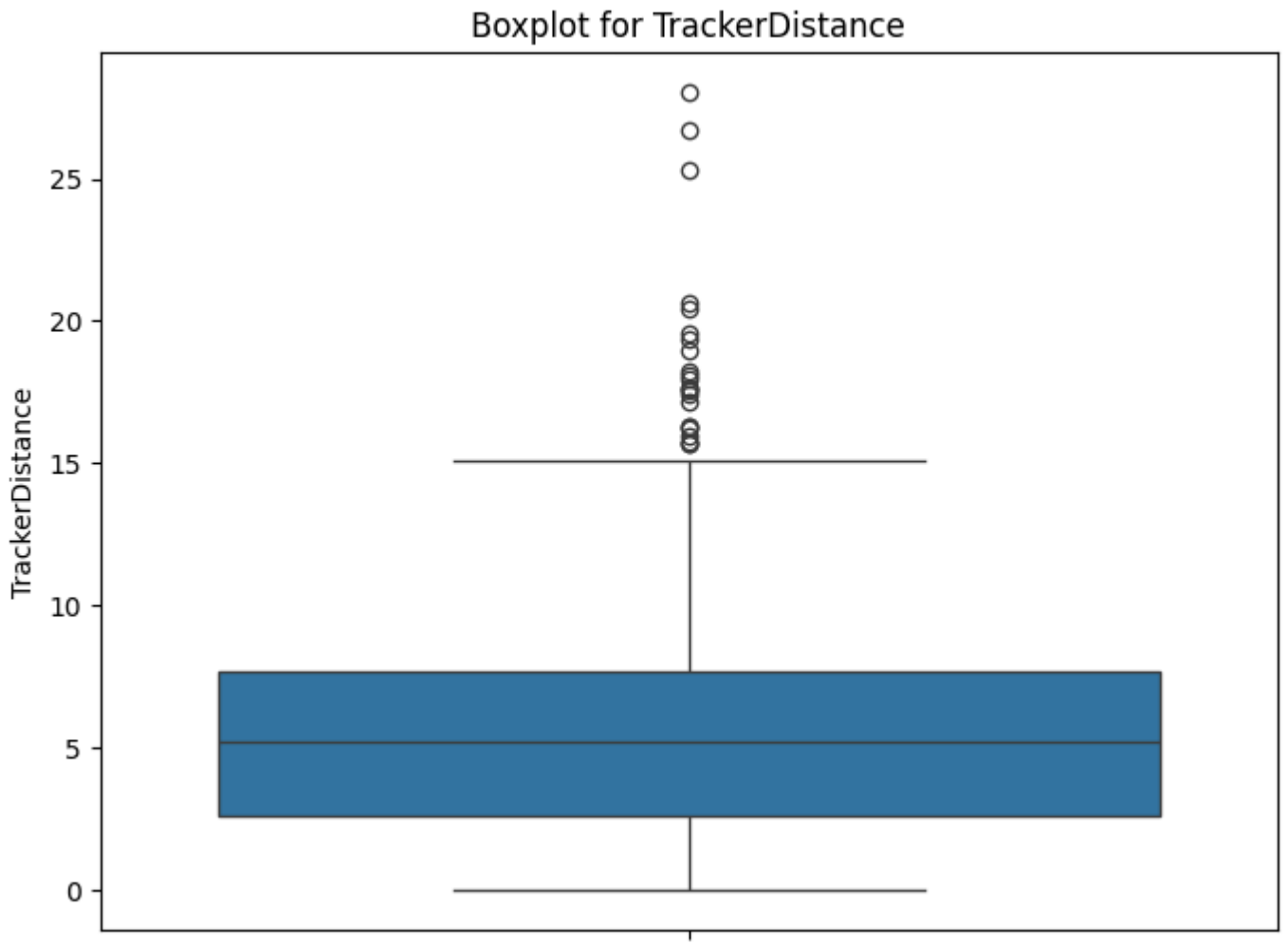
```
25%              127.000000        729.750000  1828.500000
50%              199.000000       1057.500000  2134.000000
75%              264.000000       1229.500000  2793.250000
max              518.000000       1440.000000  4900.000000
std              109.174700        301.267437   718.166862
```

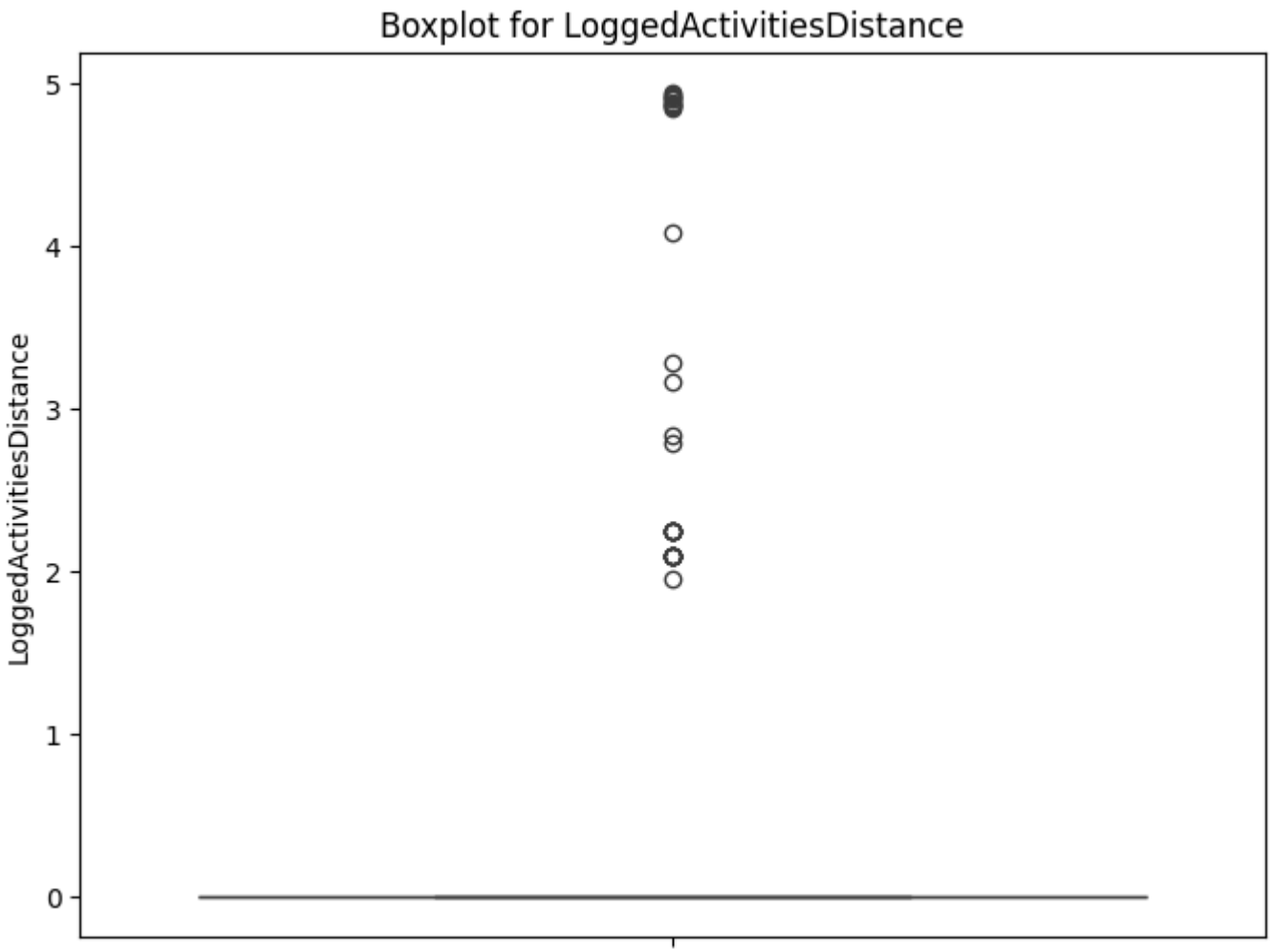
In [7]:
```
# Drop the 'ActivityDate' column for the boxplots
data_boxplot = data.drop(columns=['ActivityDate'])

# Iterate over each column and create a boxplot
for column in data_boxplot.columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=data_boxplot[column])
    plt.title(f'Boxplot for {column}')
    plt.show()
```

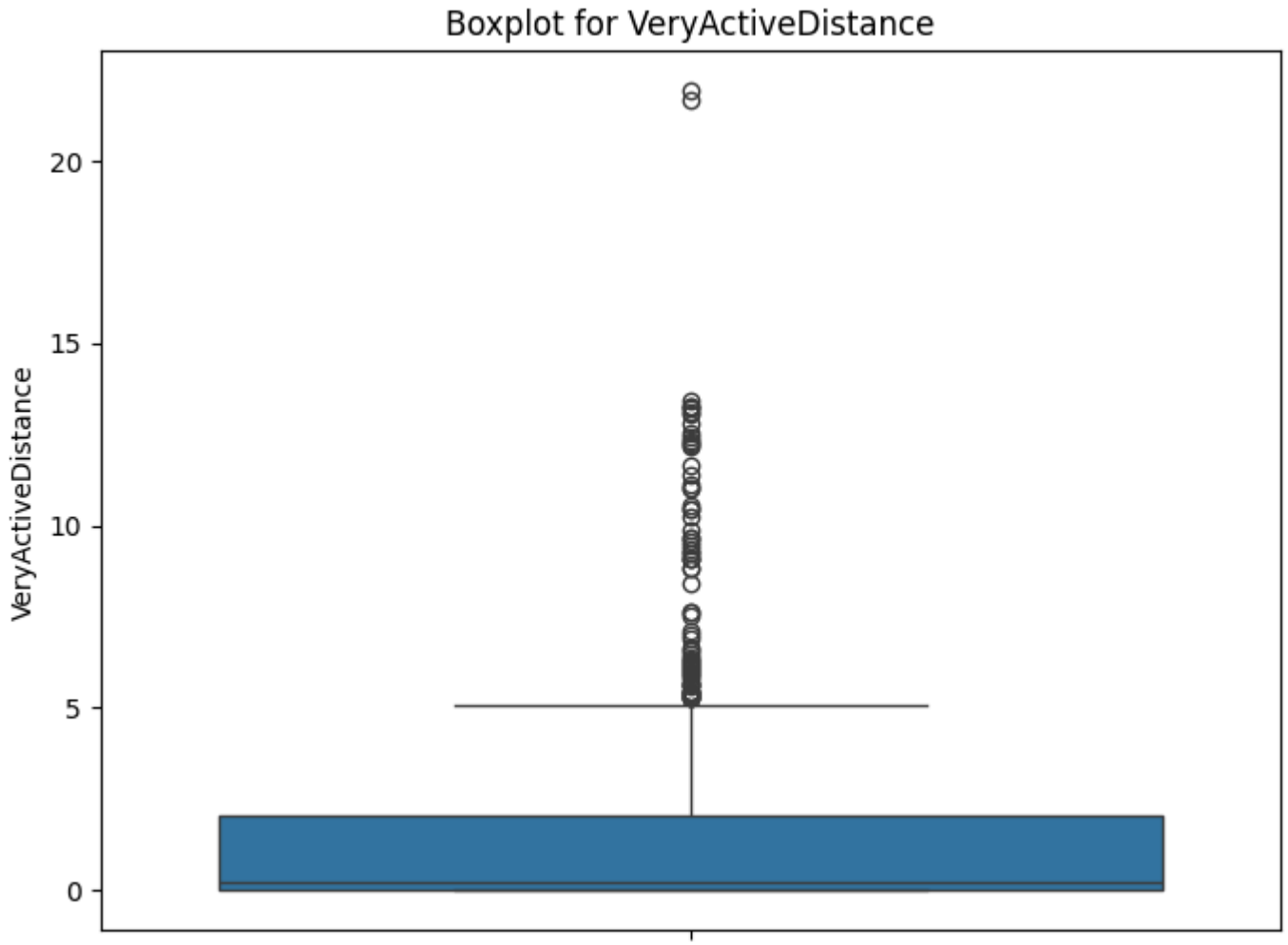

Boxplot for Id
1e9
9
8
7
6
5
4
3
2
Id

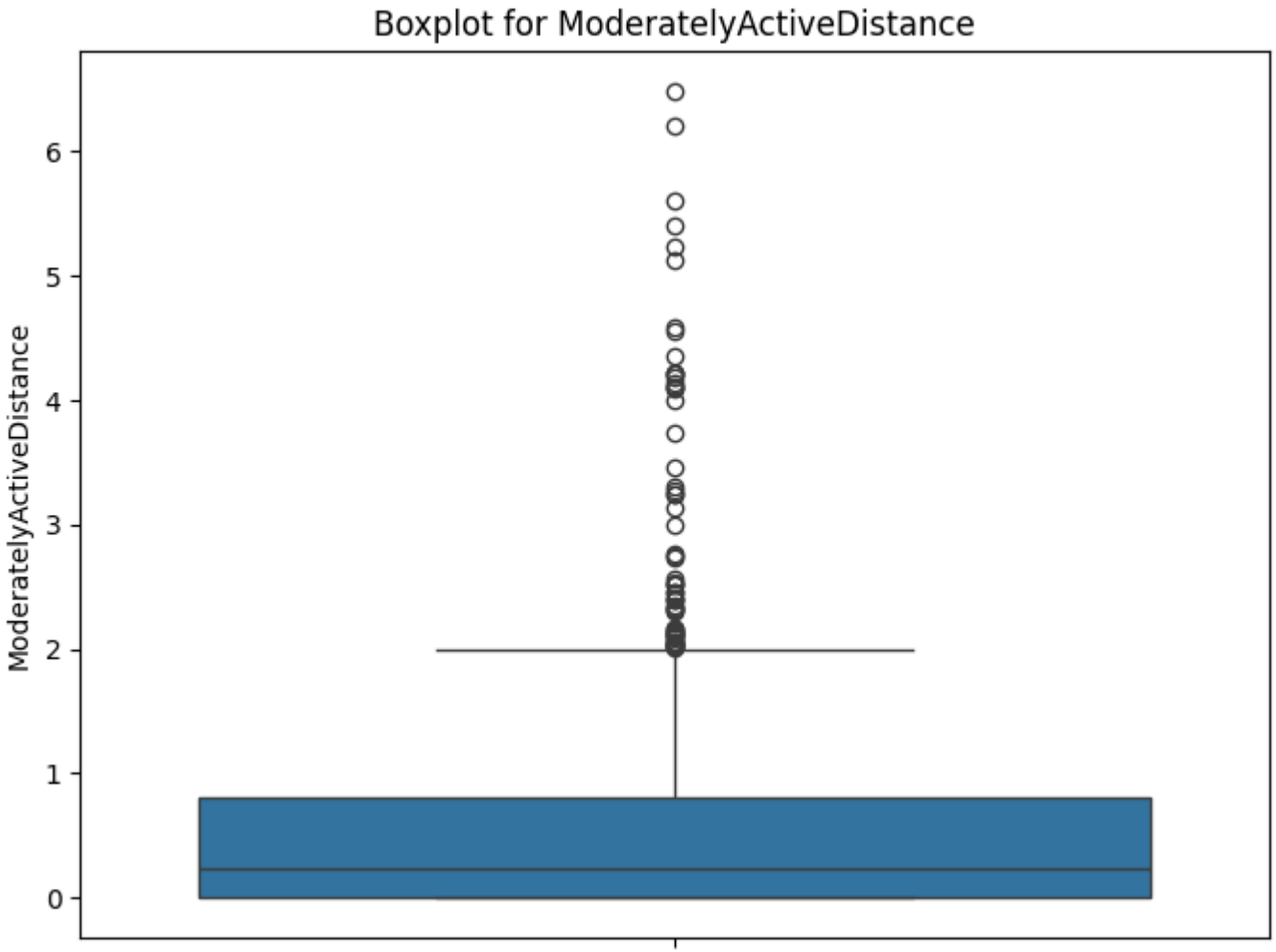

Boxplot for TotalSteps
TotalSteps
35000
30000
25000
20000
15000
10000
5000
0

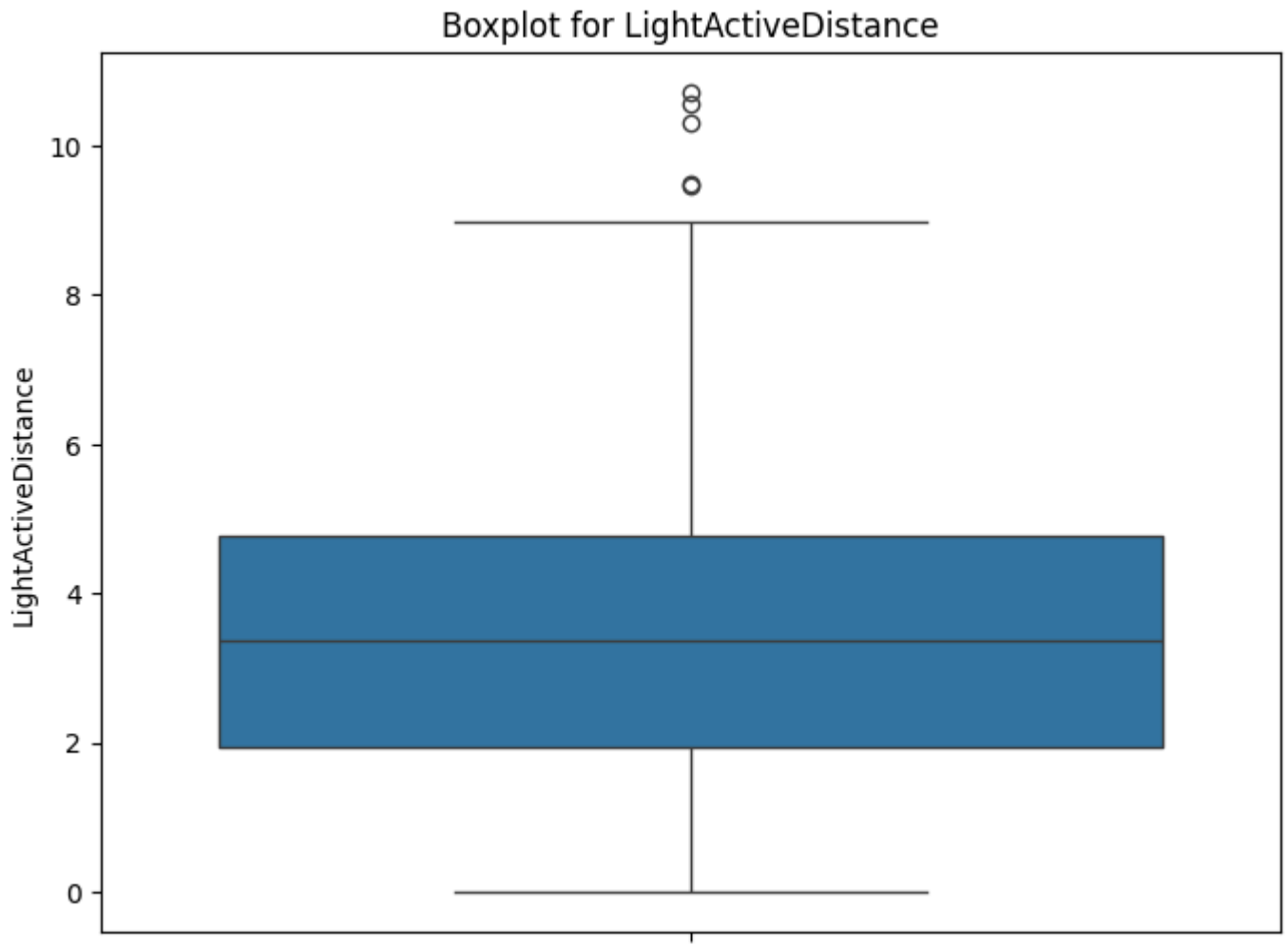

Boxplot for TotalDistance
TotalDistance
25
20
15
10
5
0


Boxplot for TrackerDistance
TrackerDistance
25
20
15
10
5
0

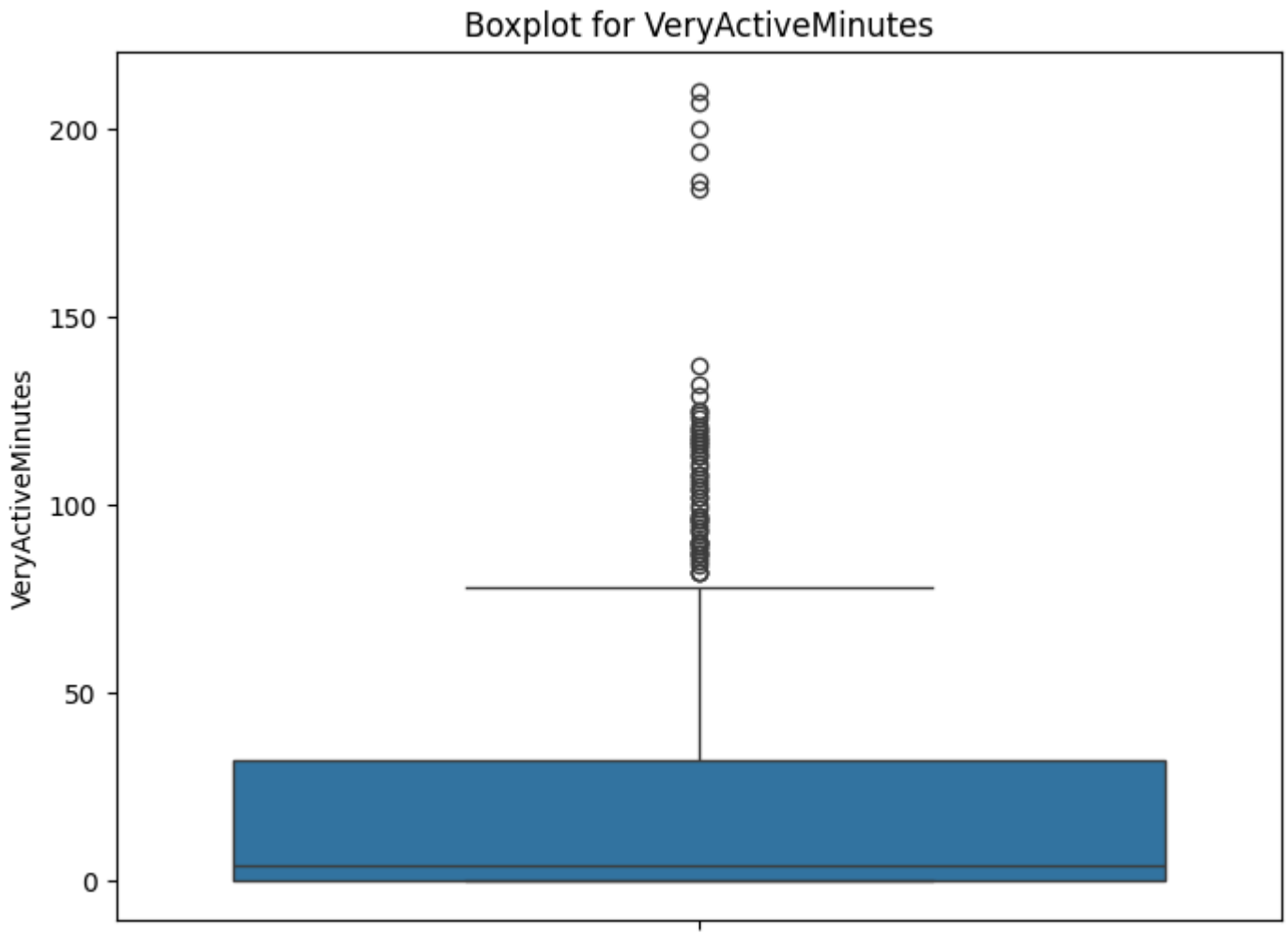

Boxplot for LoggedActivitiesDistance
LoggedActivitiesDistance
5
4
3
2
1
0

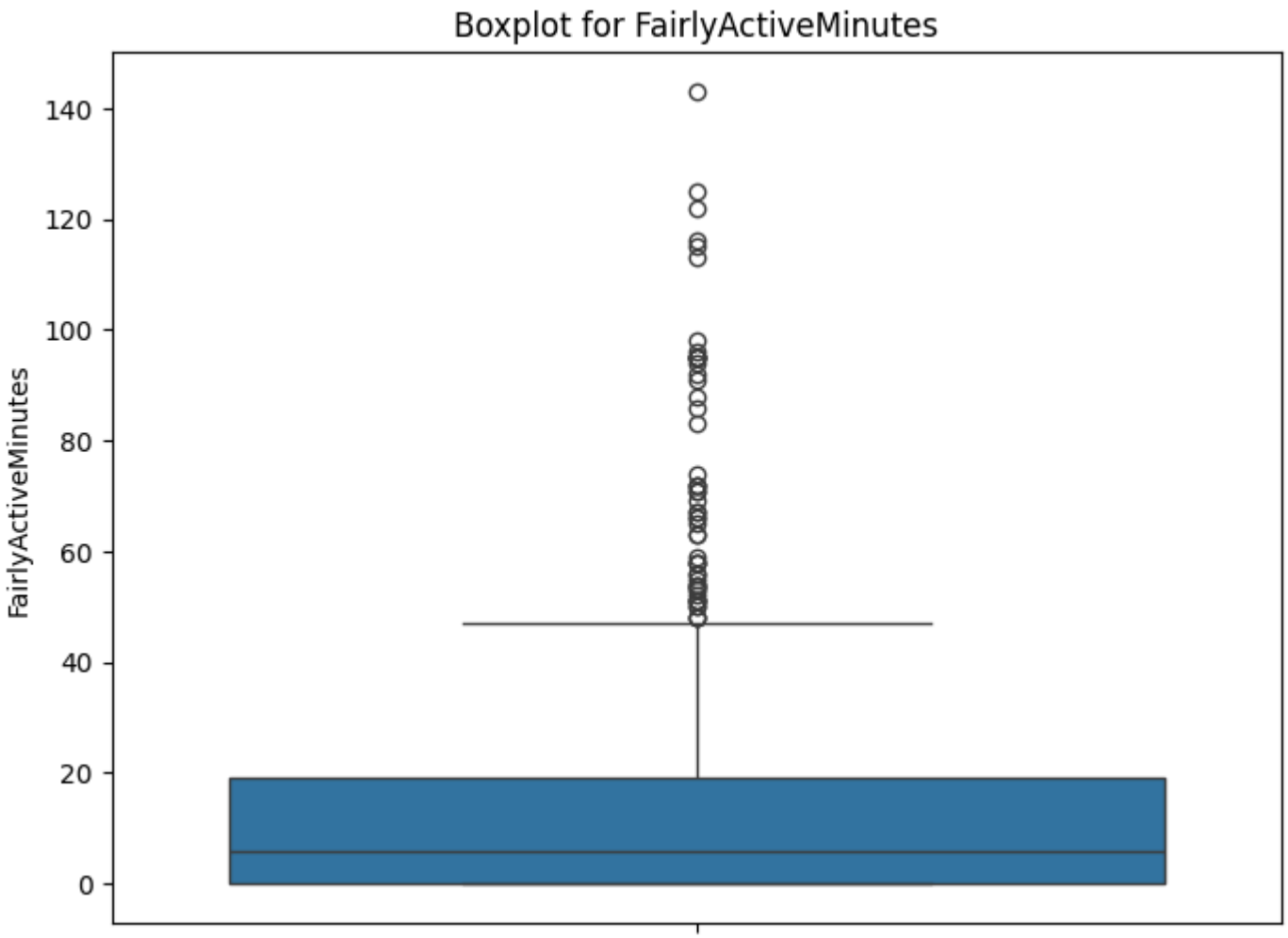

Boxplot for VeryActiveDistance
VeryActiveDistance
0
5
10
15
20

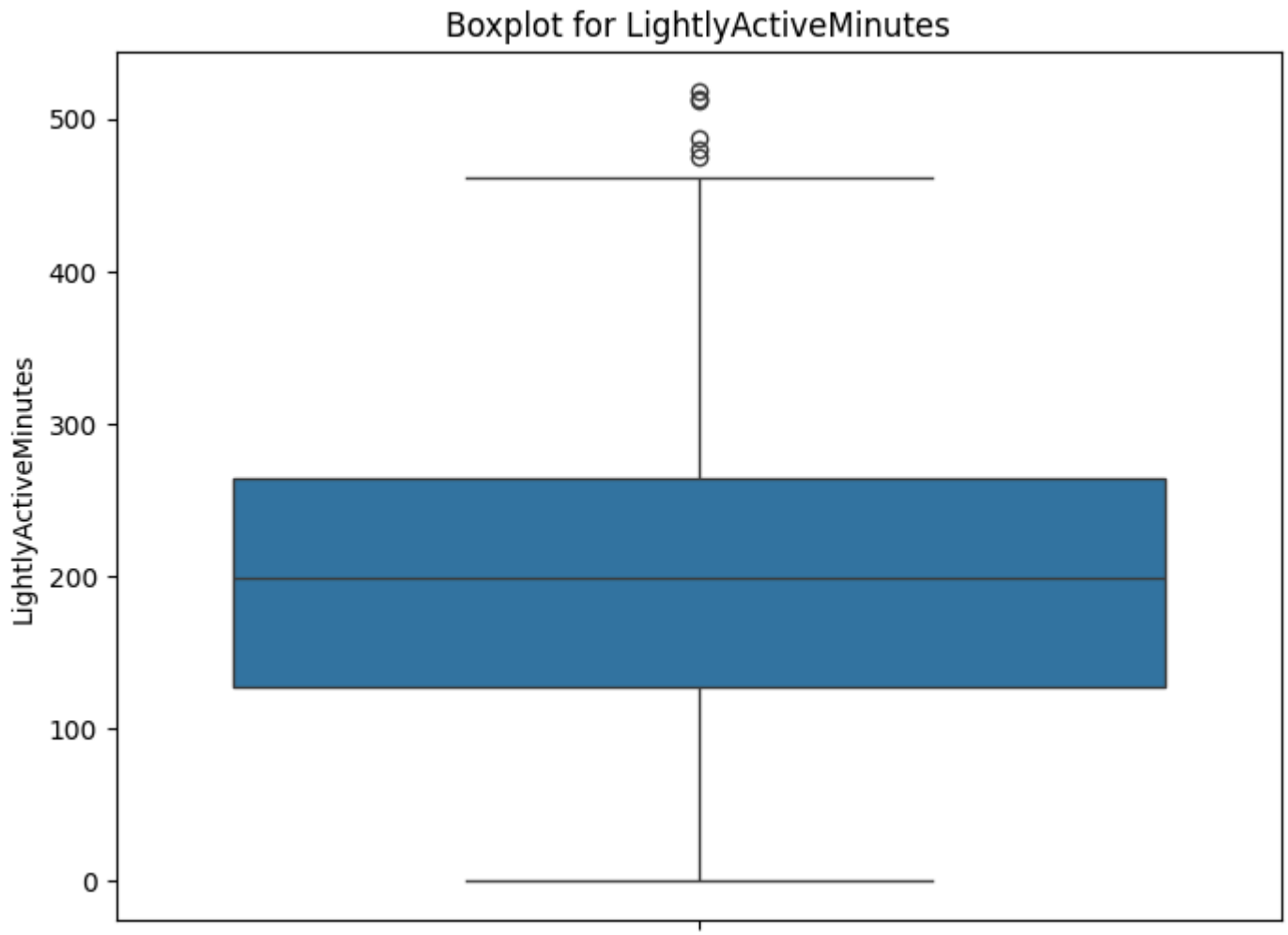

Boxplot for ModeratelyActiveDistance
ModeratelyActiveDistance
0
1
2
3
4
5
6

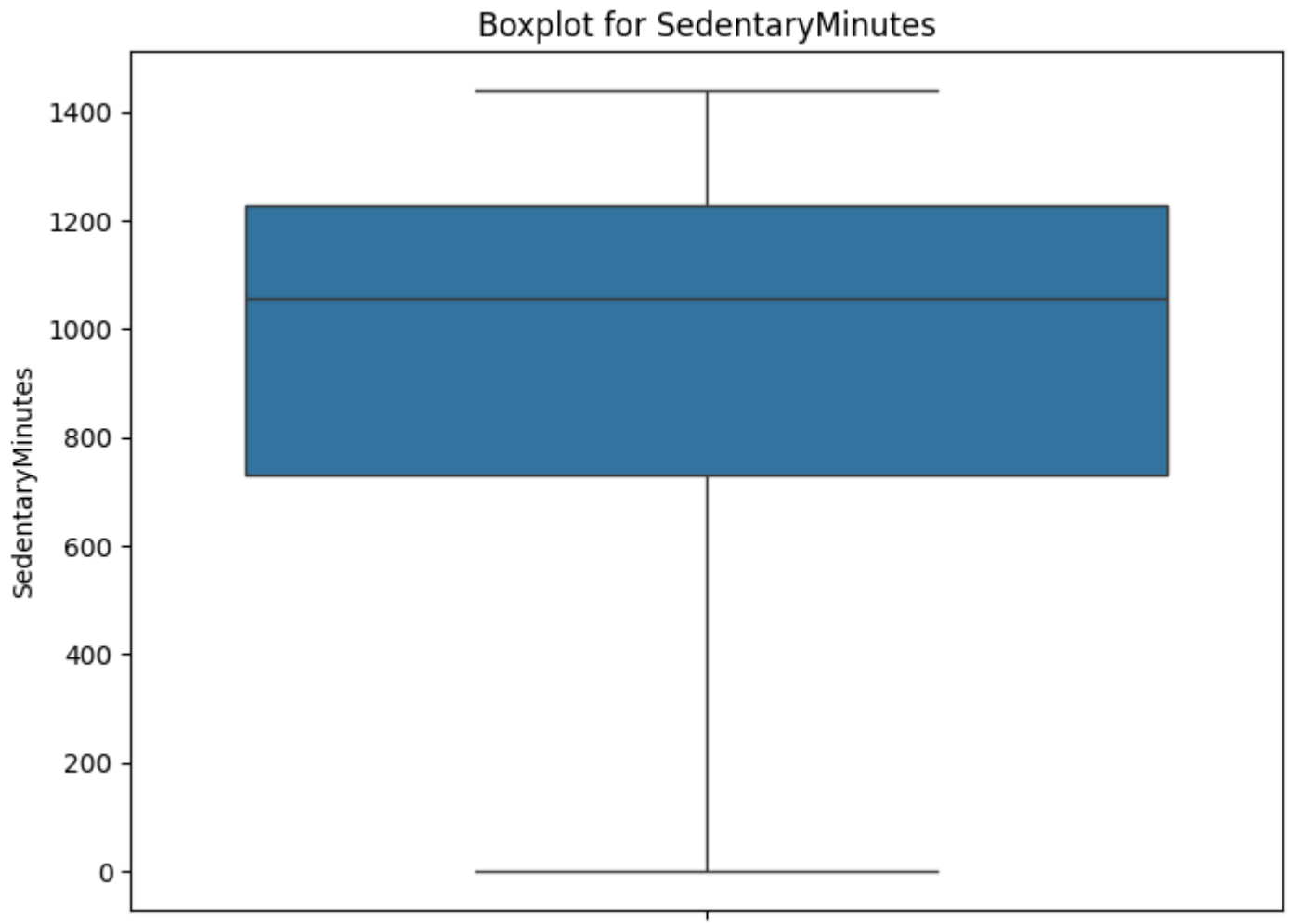

Boxplot for LightActiveDistance
LightActiveDistance
0
2
4
6
8
10

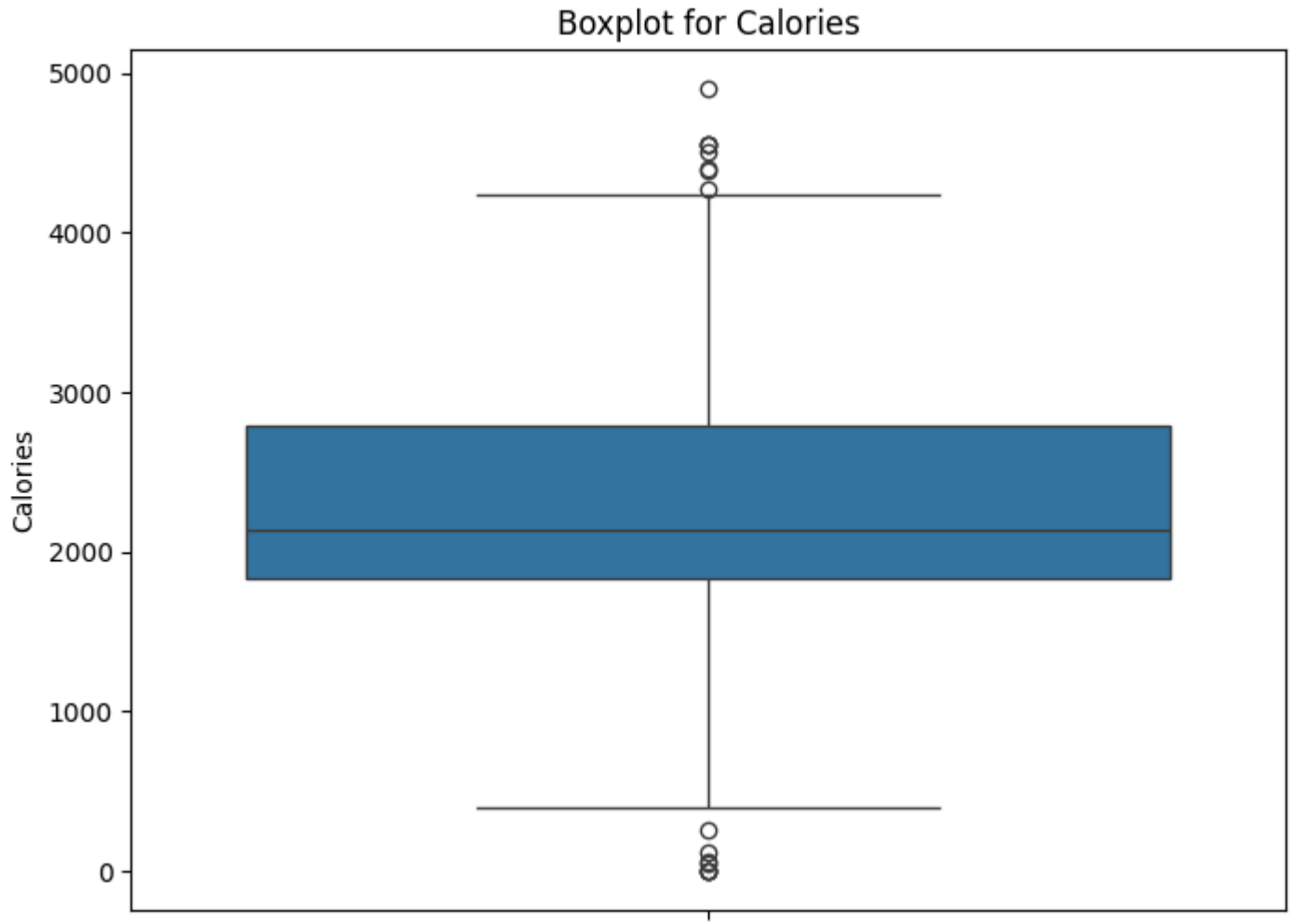

Boxplot for SedentaryActiveDistance
SedentaryActiveDistance
0.10
0.08
0.06
0.04
0.02
0.00


Boxplot for VeryActiveMinutes
VeryActiveMinutes
200
150
100
50
0

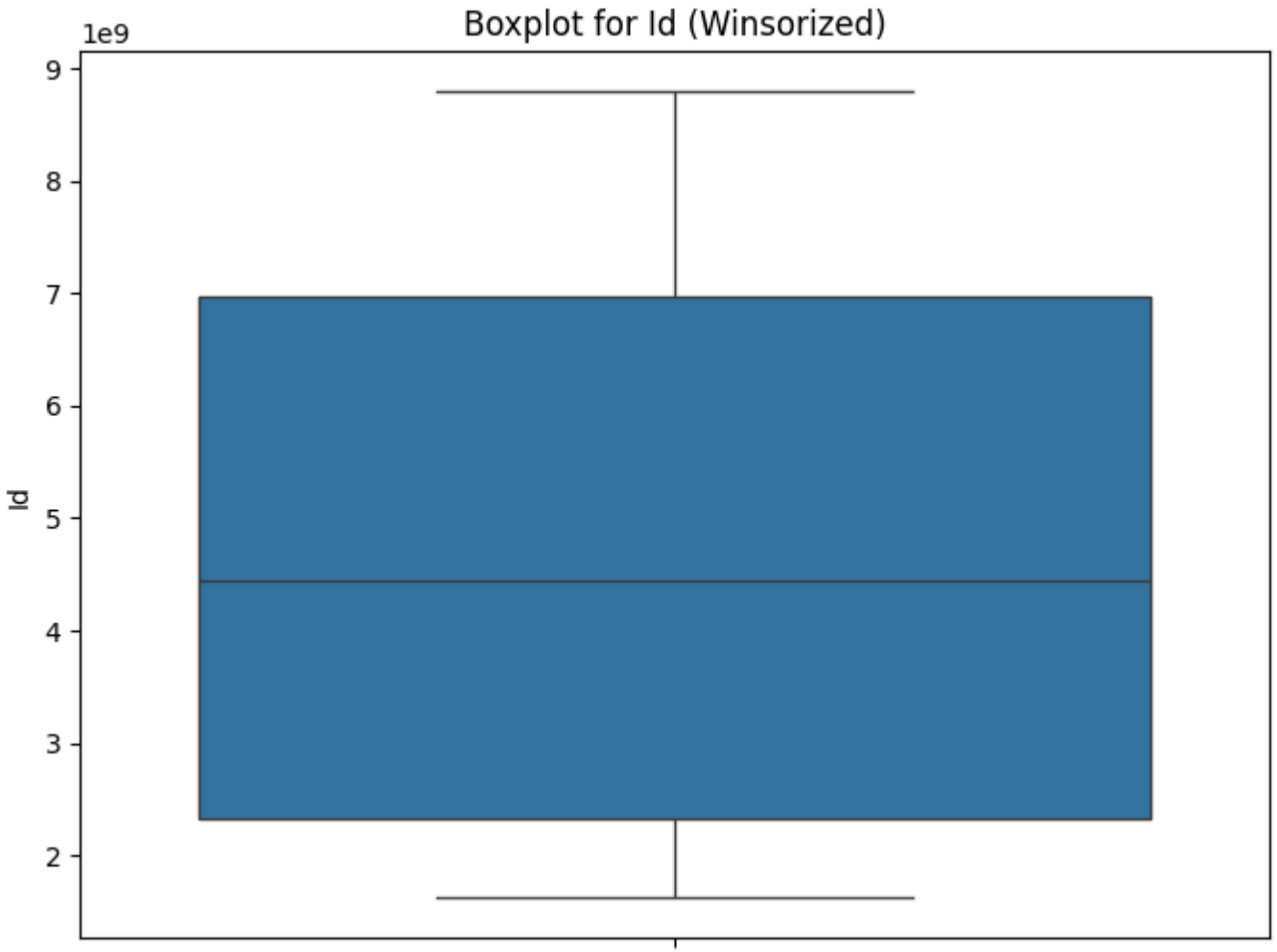

Boxplot for FairlyActiveMinutes
FairlyActiveMinutes
140
120
100
80
60
40
20
0

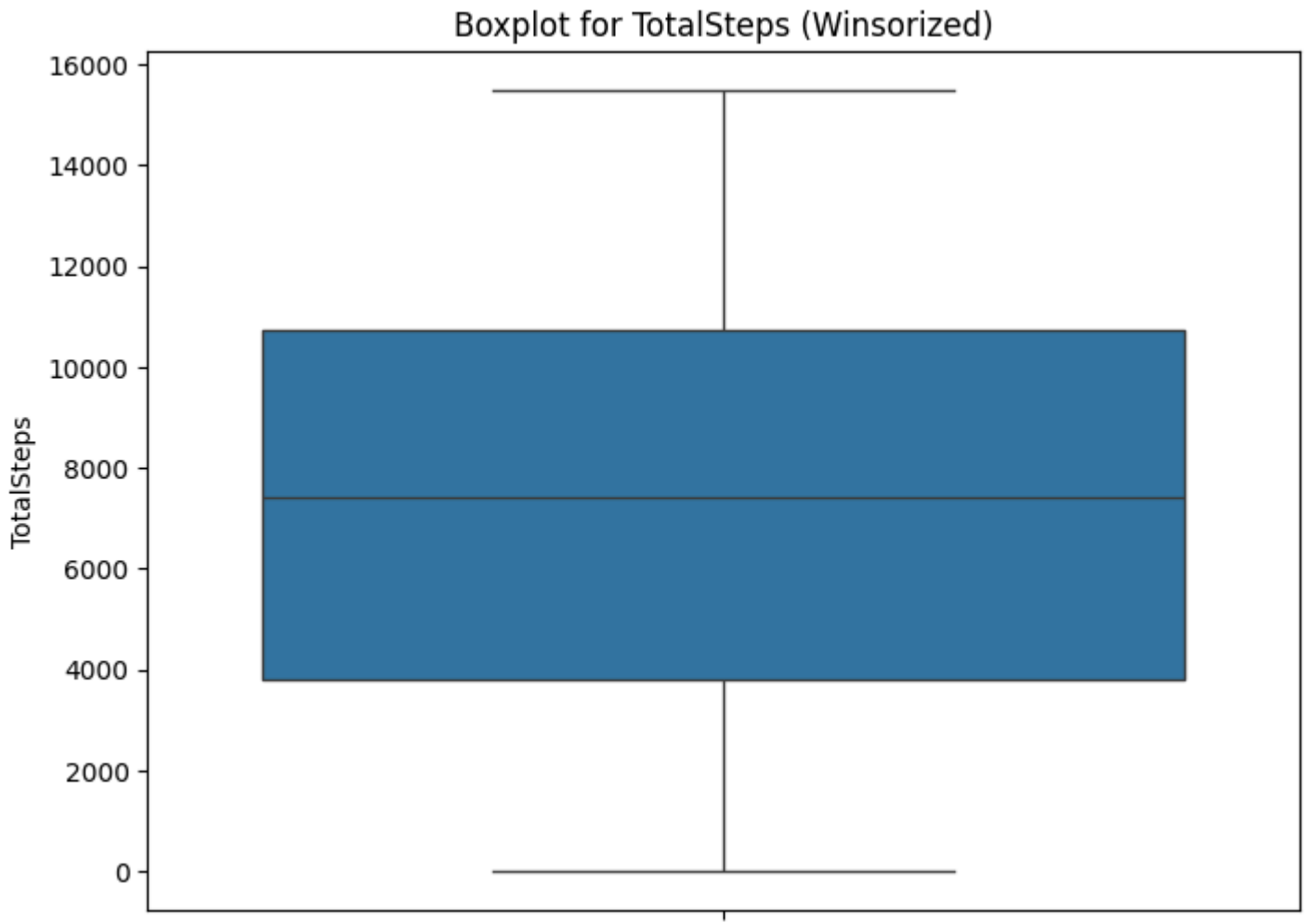

Boxplot for LightlyActiveMinutes
LightlyActiveMinutes
500
400
300
200
100
0

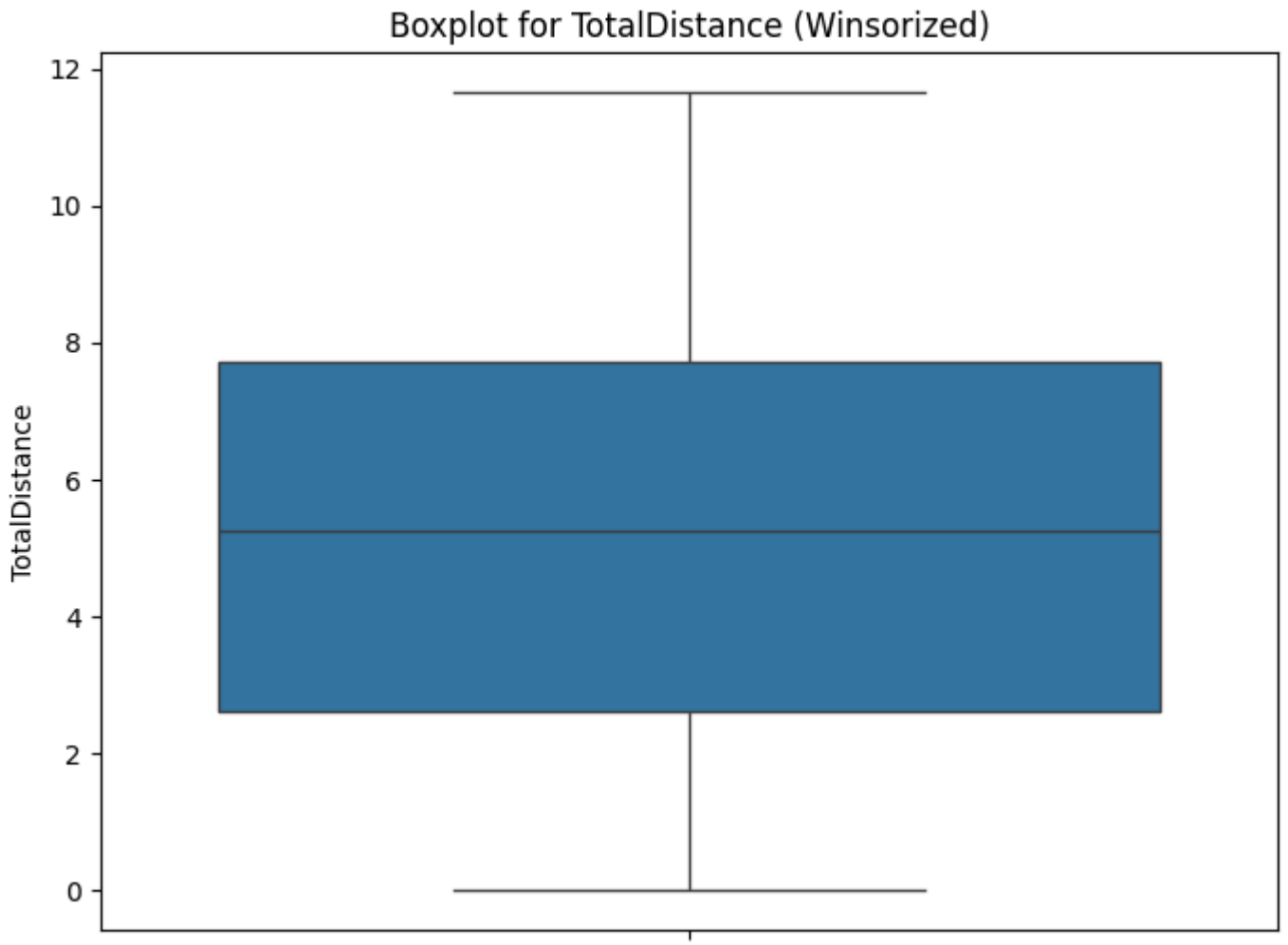

Boxplot for SedentaryMinutes
SedentaryMinutes
1400
1200
1000
800
600
400
200
0

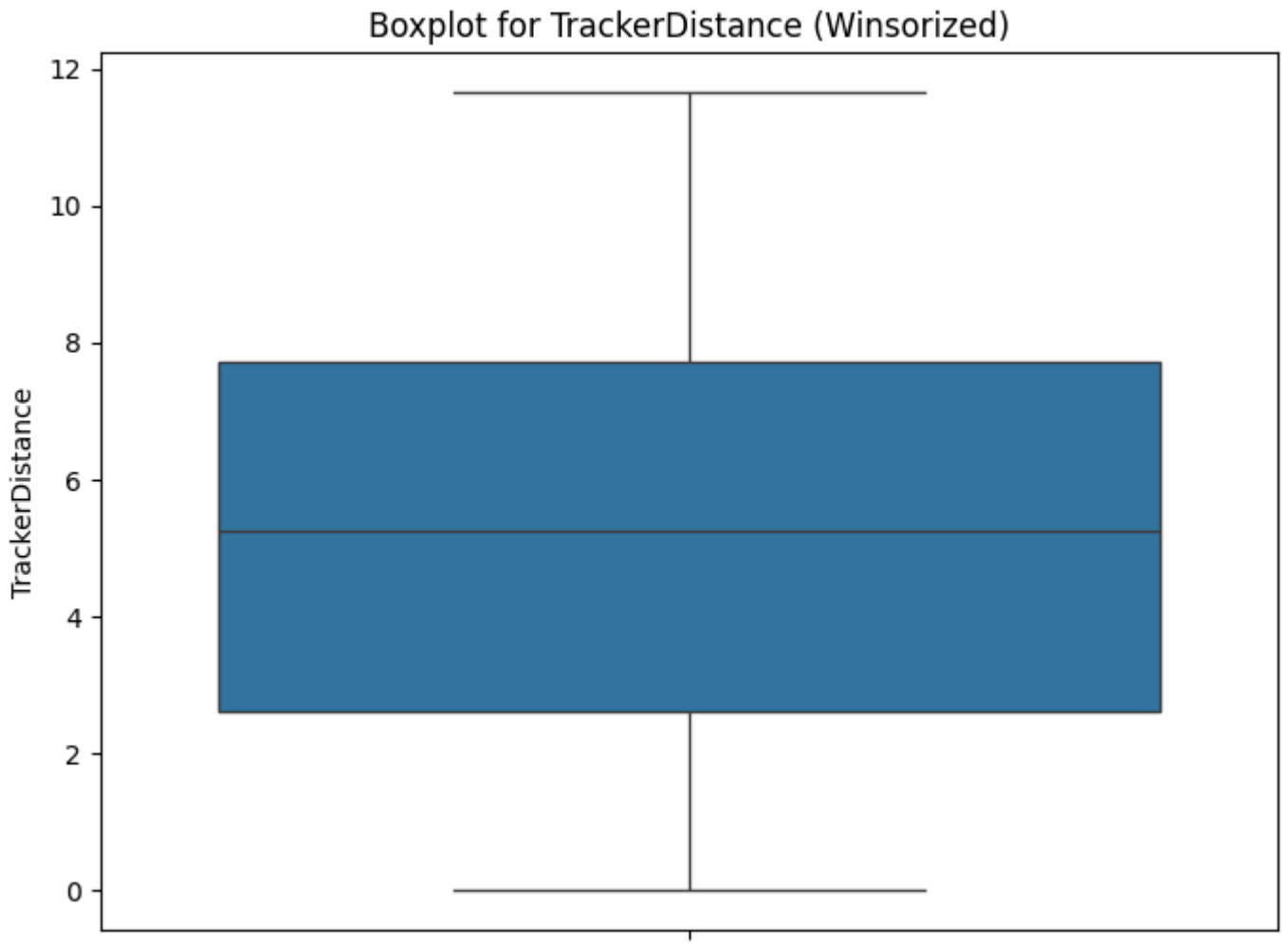


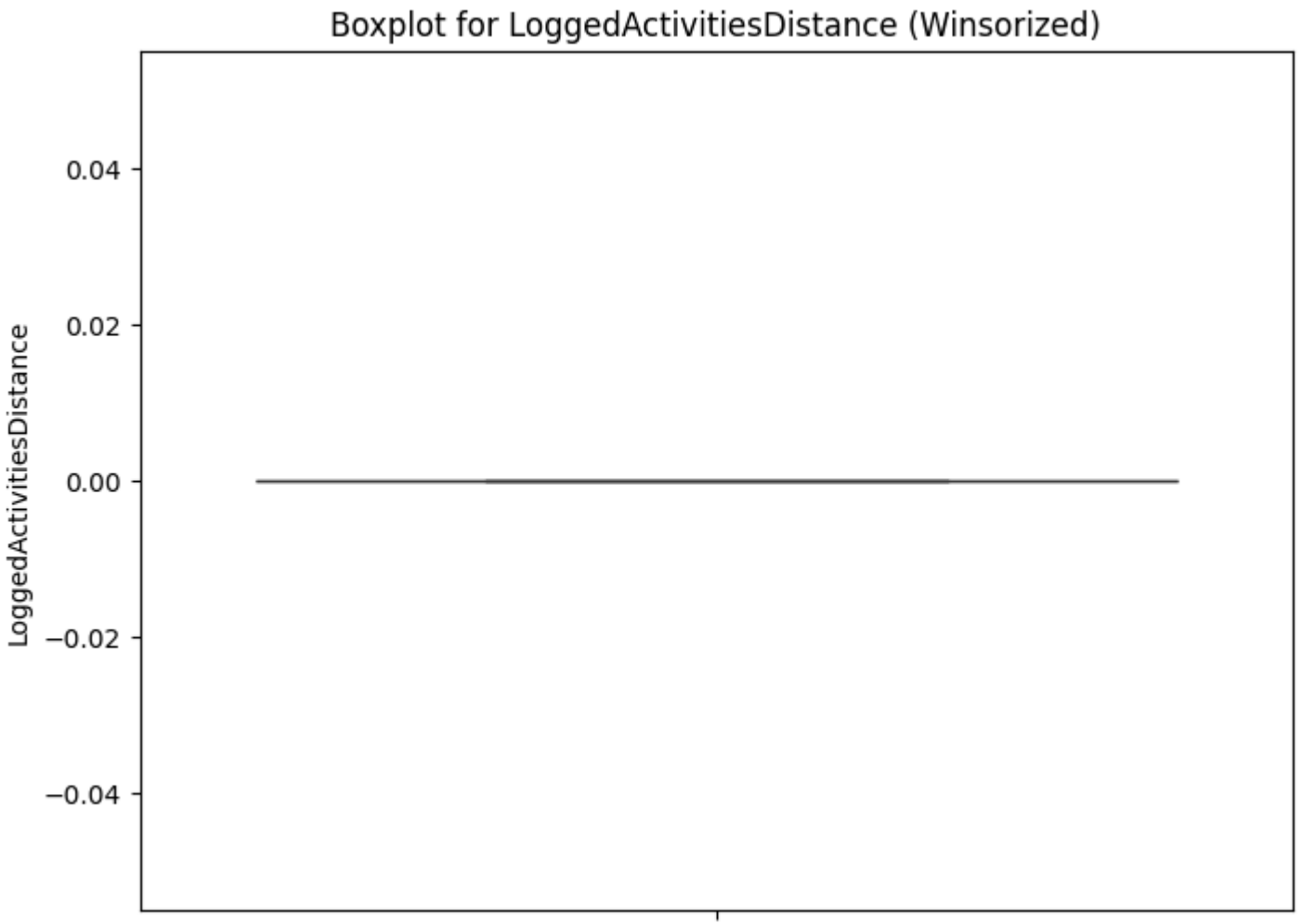
In [8]:
```
from scipy.stats.mstats import winsorize

# Apply winsorization to each column
winsorized_data = data_boxplot.apply(lambda x: winsorize(x, limits=[0.05, 0.05]) if x.dtype != 'object' else x)

# Iterate over each column and create a boxplot
for column in winsorized_data.columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=winsorized_data[column])
    plt.title(f'Boxplot for {column} (Winsorized)')
    plt.show()
```

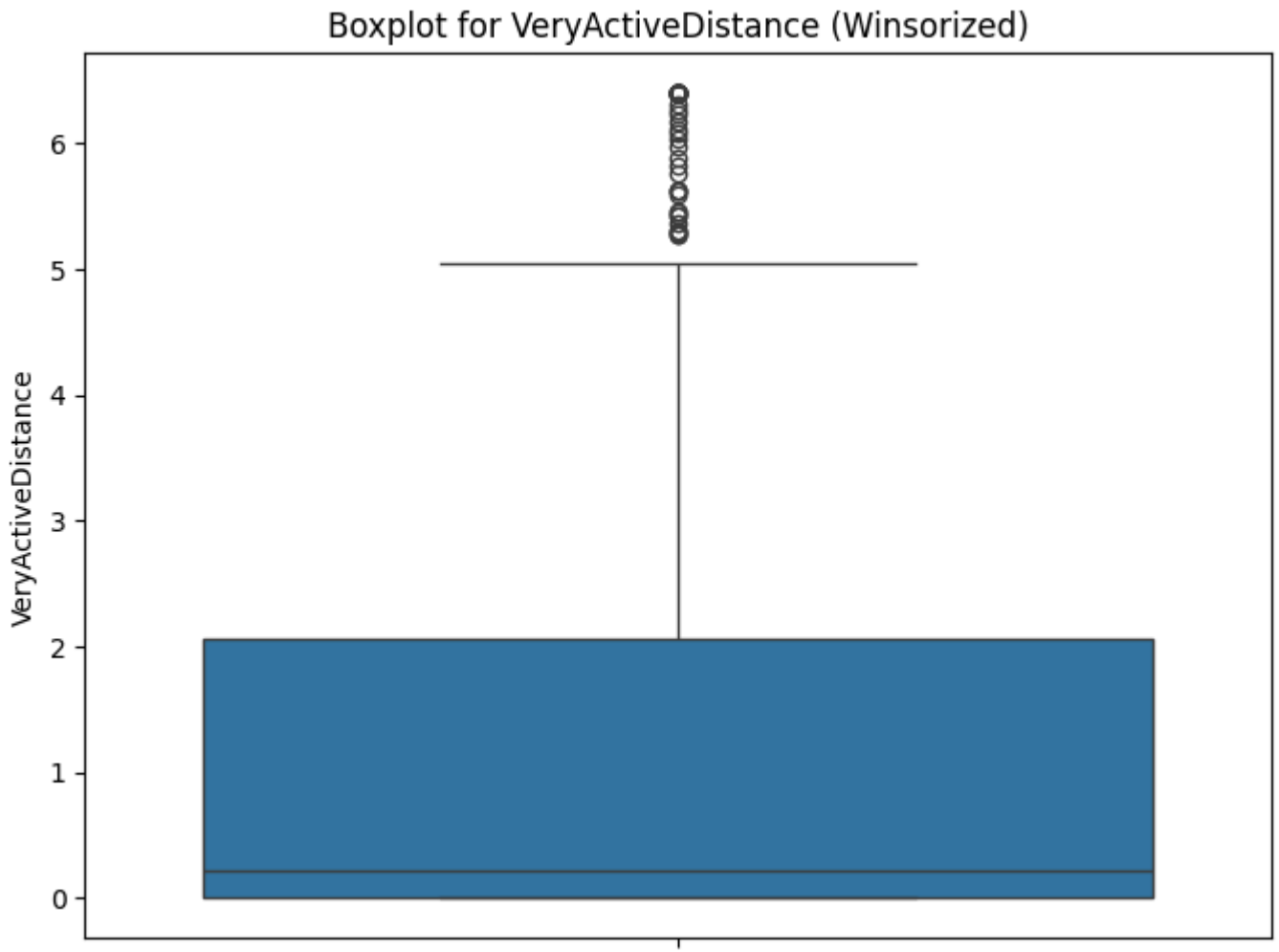

Boxplot for Id (Winsorized)
1e9
9
8
7
6
5
4
3
2
Id

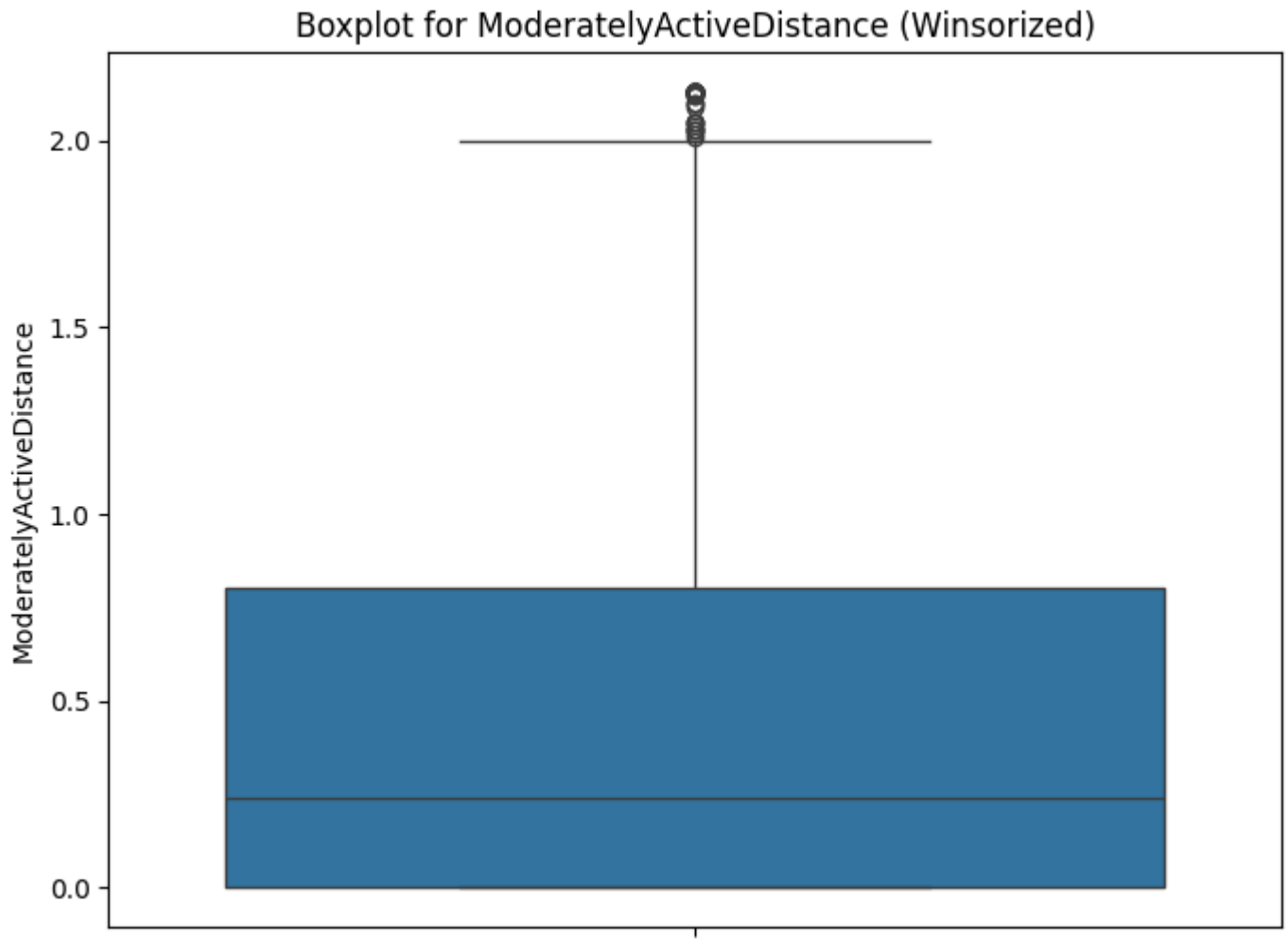

Boxplot for TotalSteps (Winsorized)
TotalSteps
16000
14000
12000
10000
8000
6000
4000
2000
0

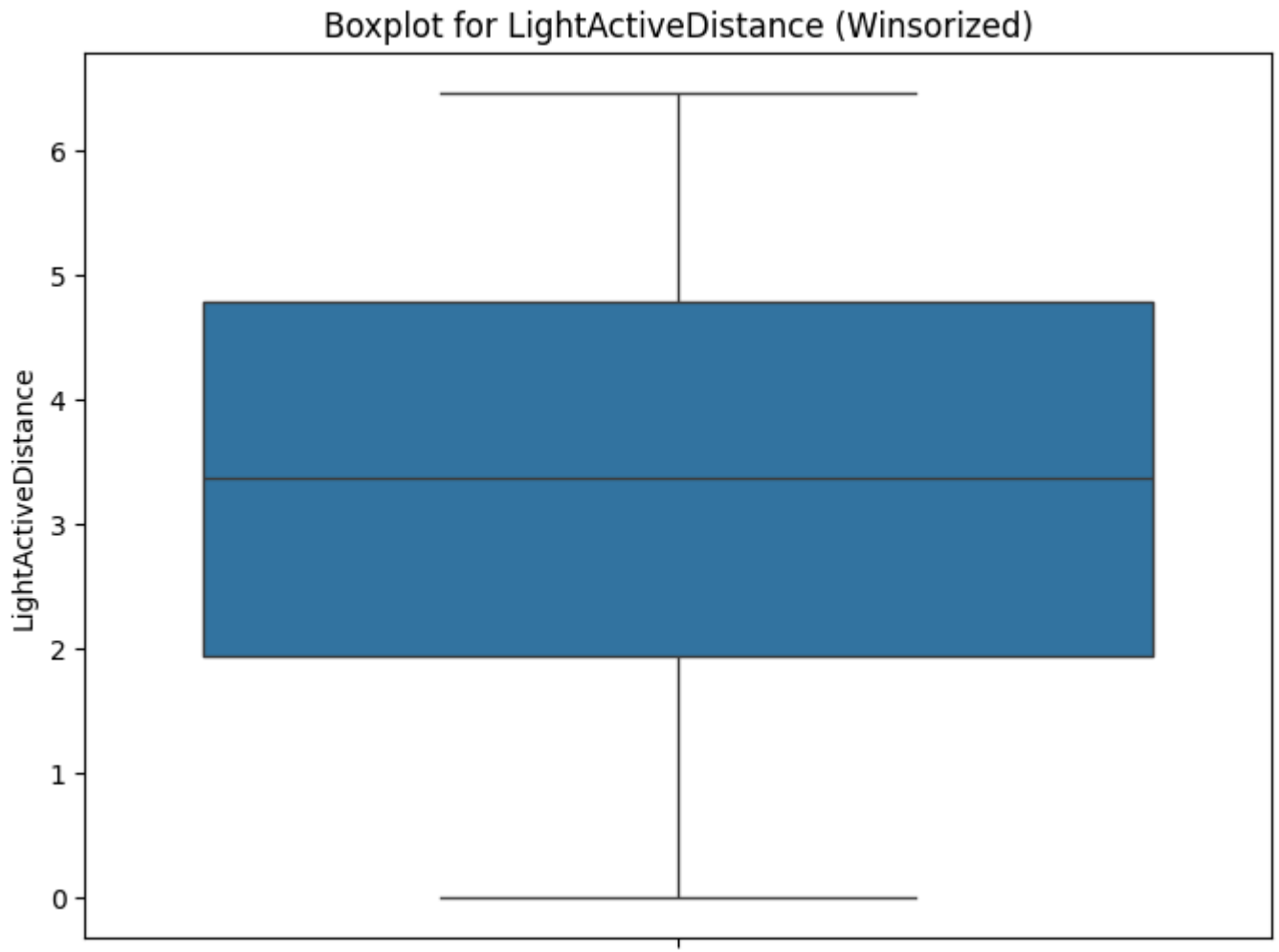

Boxplot for TotalDistance (Winsorized)
TotalDistance
12
10
8
6
4
2
0

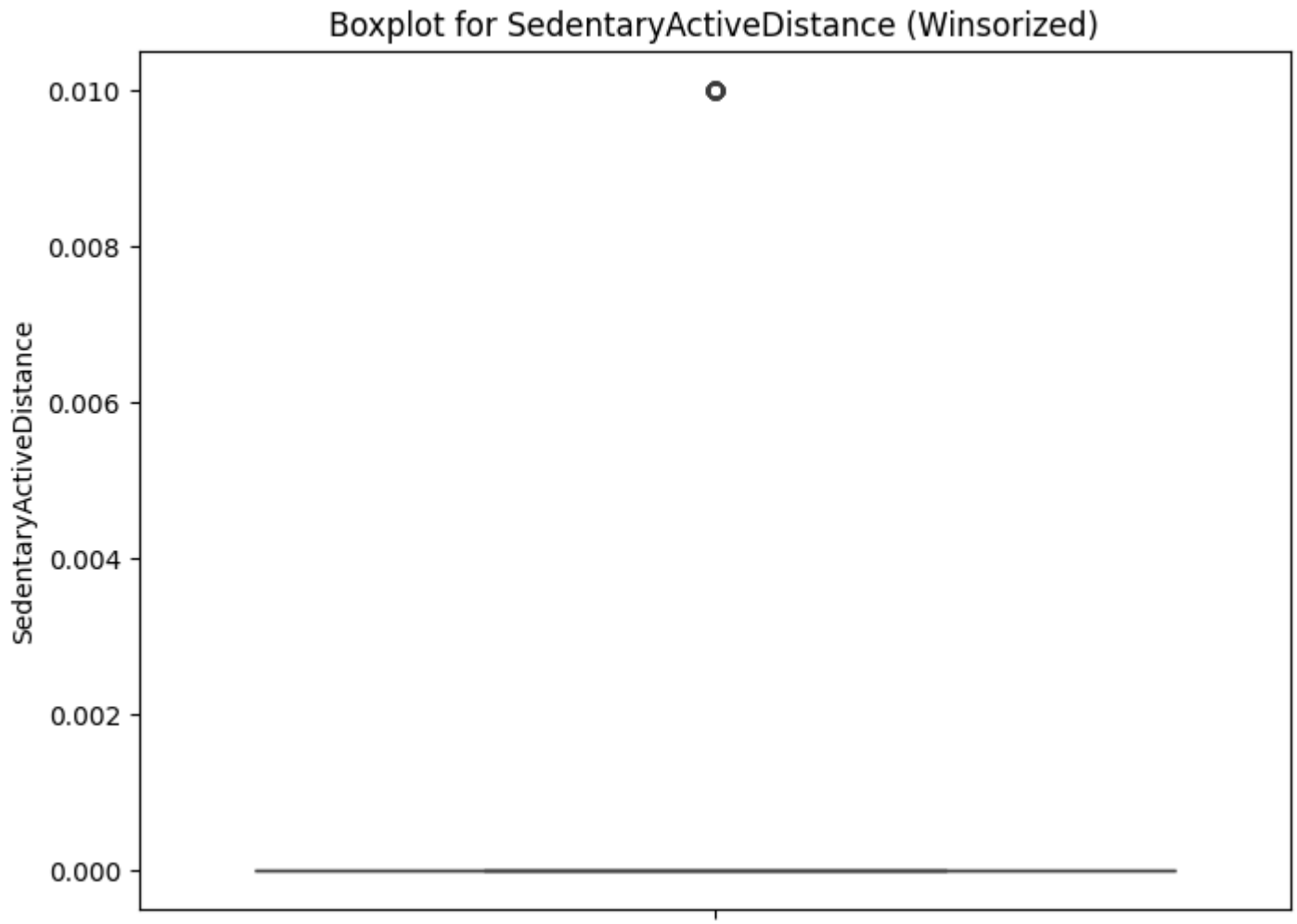

Boxplot for TrackerDistance (Winsorized)
TrackerDistance
12
10
8
6
4
2
0

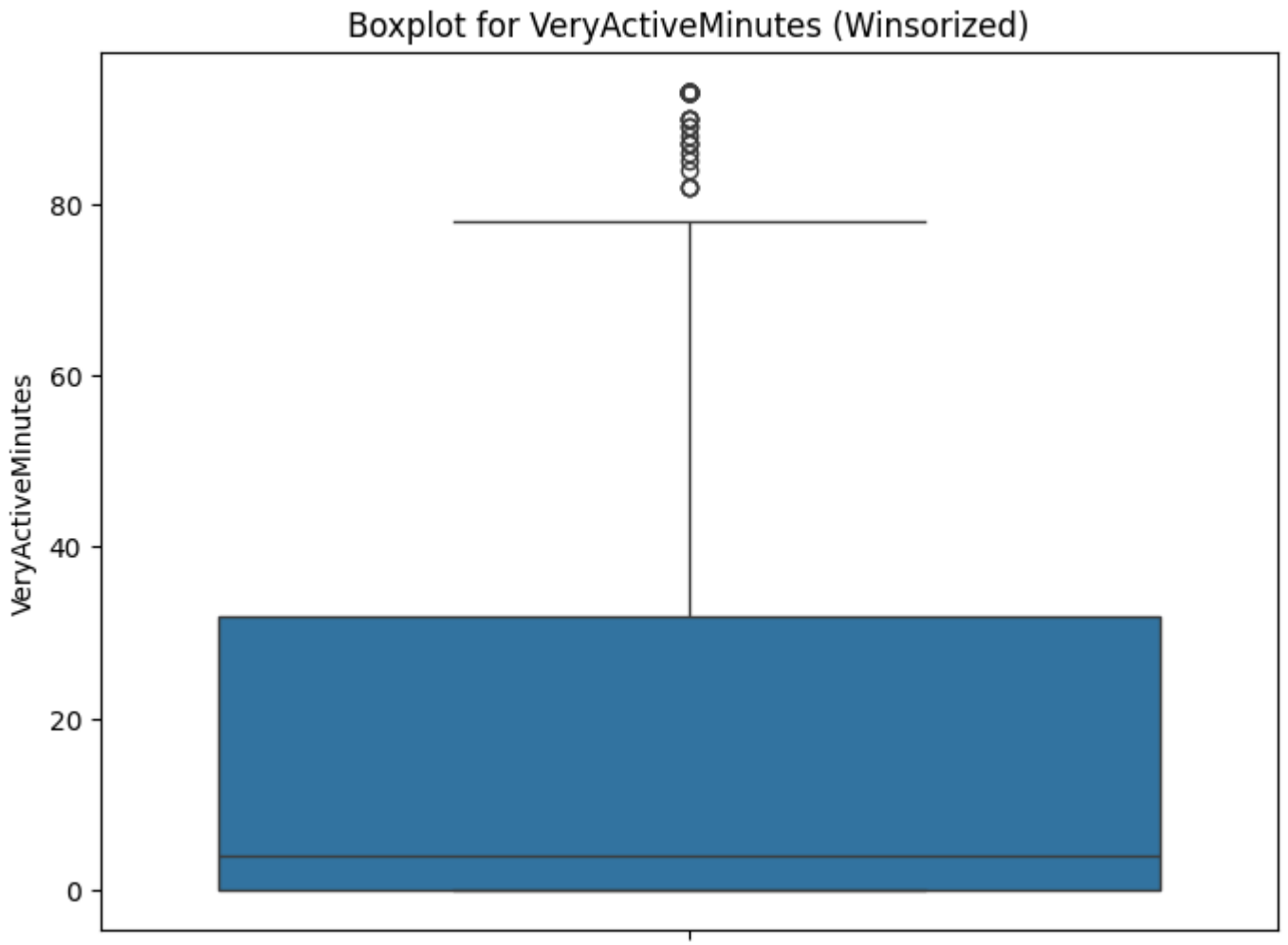

Boxplot for LoggedActivitiesDistance (Winsorized)
LoggedActivitiesDistance
0.04
0.02
0.00
−0.02
−0.04

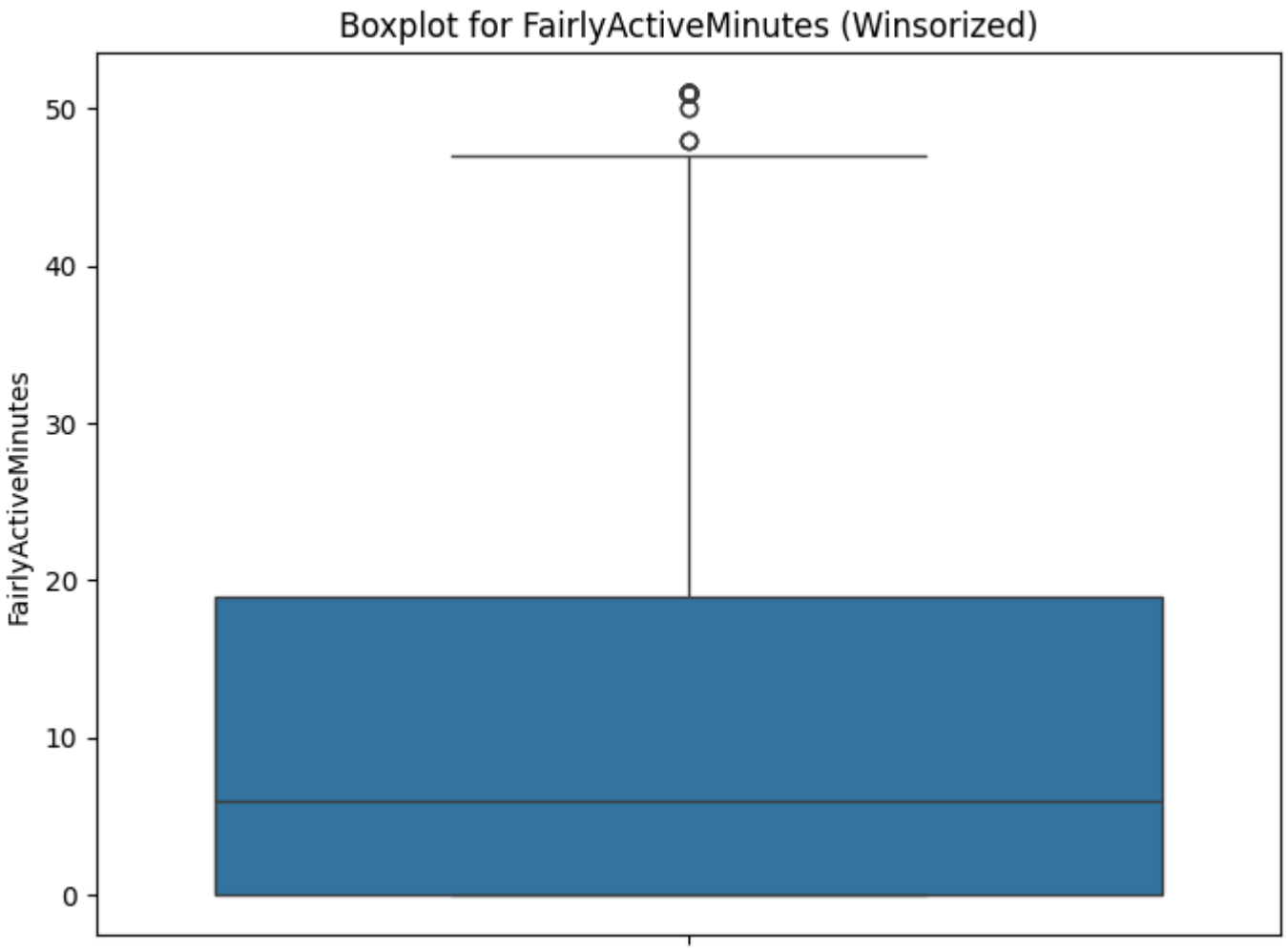

Boxplot for VeryActiveDistance (Winsorized)
VeryActiveDistance
0
1
2
3
4
5
6

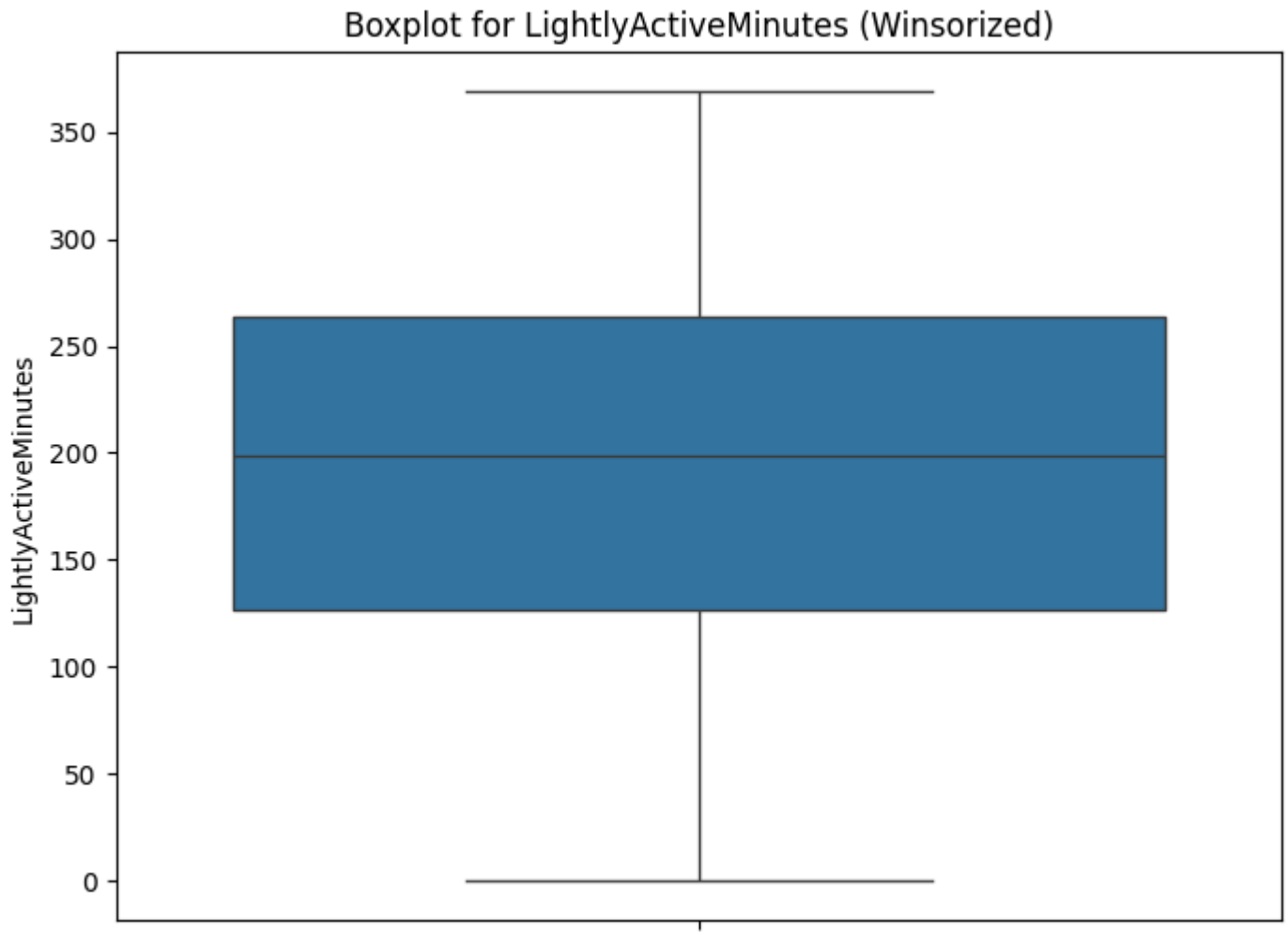

Boxplot for ModeratelyActiveDistance (Winsorized)
ModeratelyActiveDistance
2.0
1.5
1.0
0.5
0.0

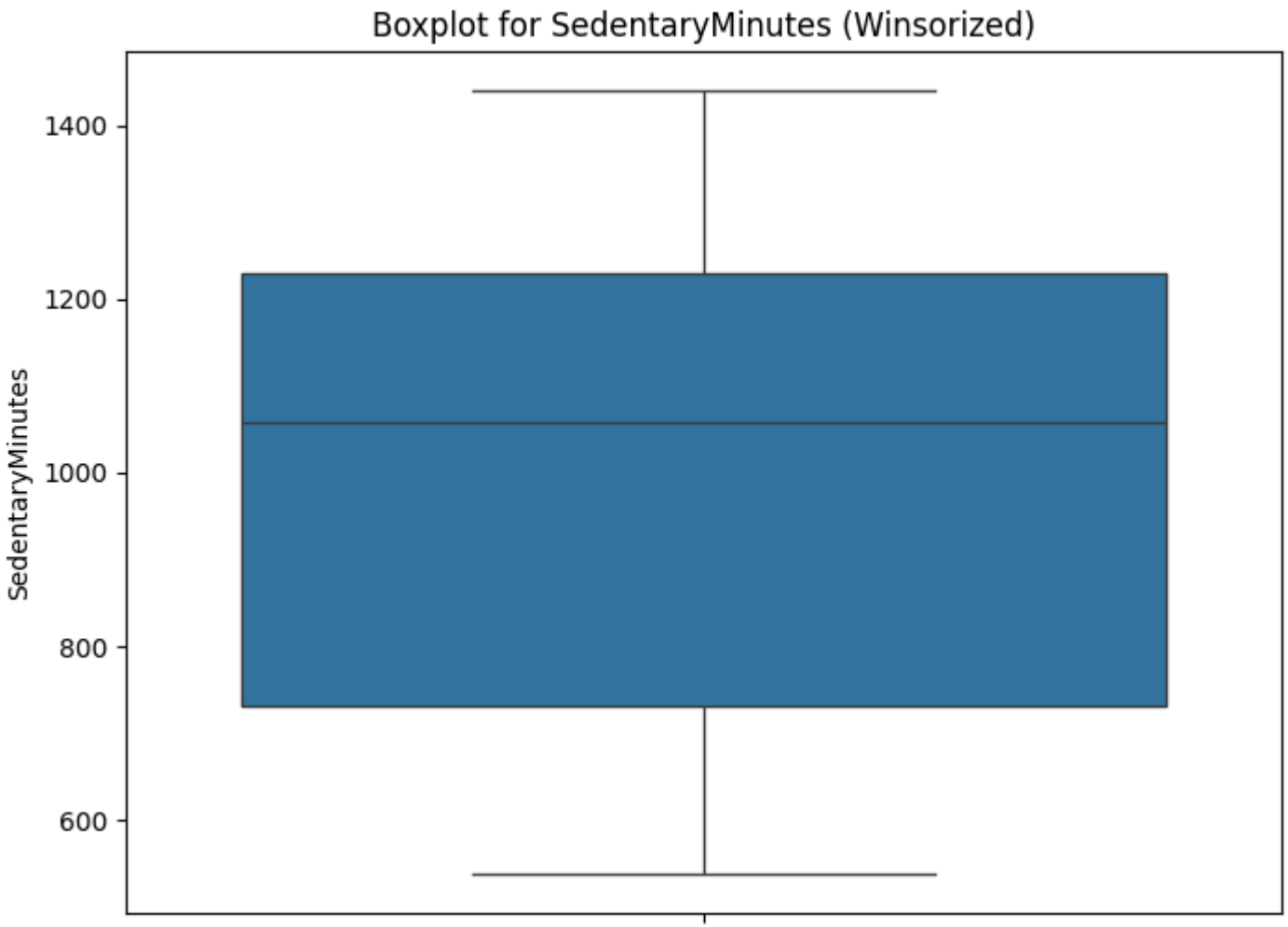

Boxplot for LightActiveDistance (Winsorized)
LightActiveDistance
0
1
2
3
4
5
6

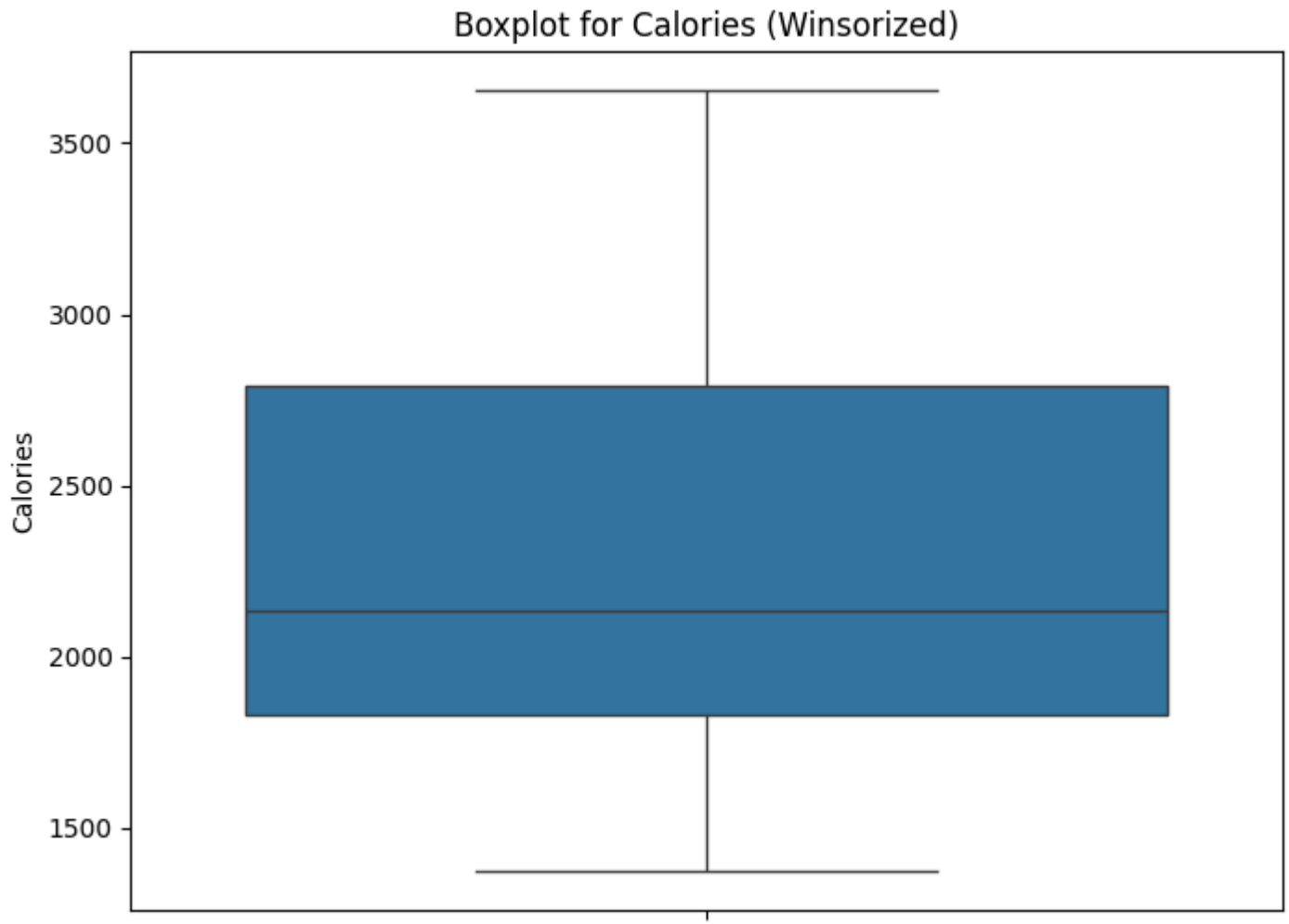

Boxplot for SedentaryActiveDistance (Winsorized)
SedentaryActiveDistance
0.010
0.008
0.006
0.004
0.002
0.000


Boxplot for VeryActiveMinutes (Winsorized)
VeryActiveMinutes
0
20
40
60
80

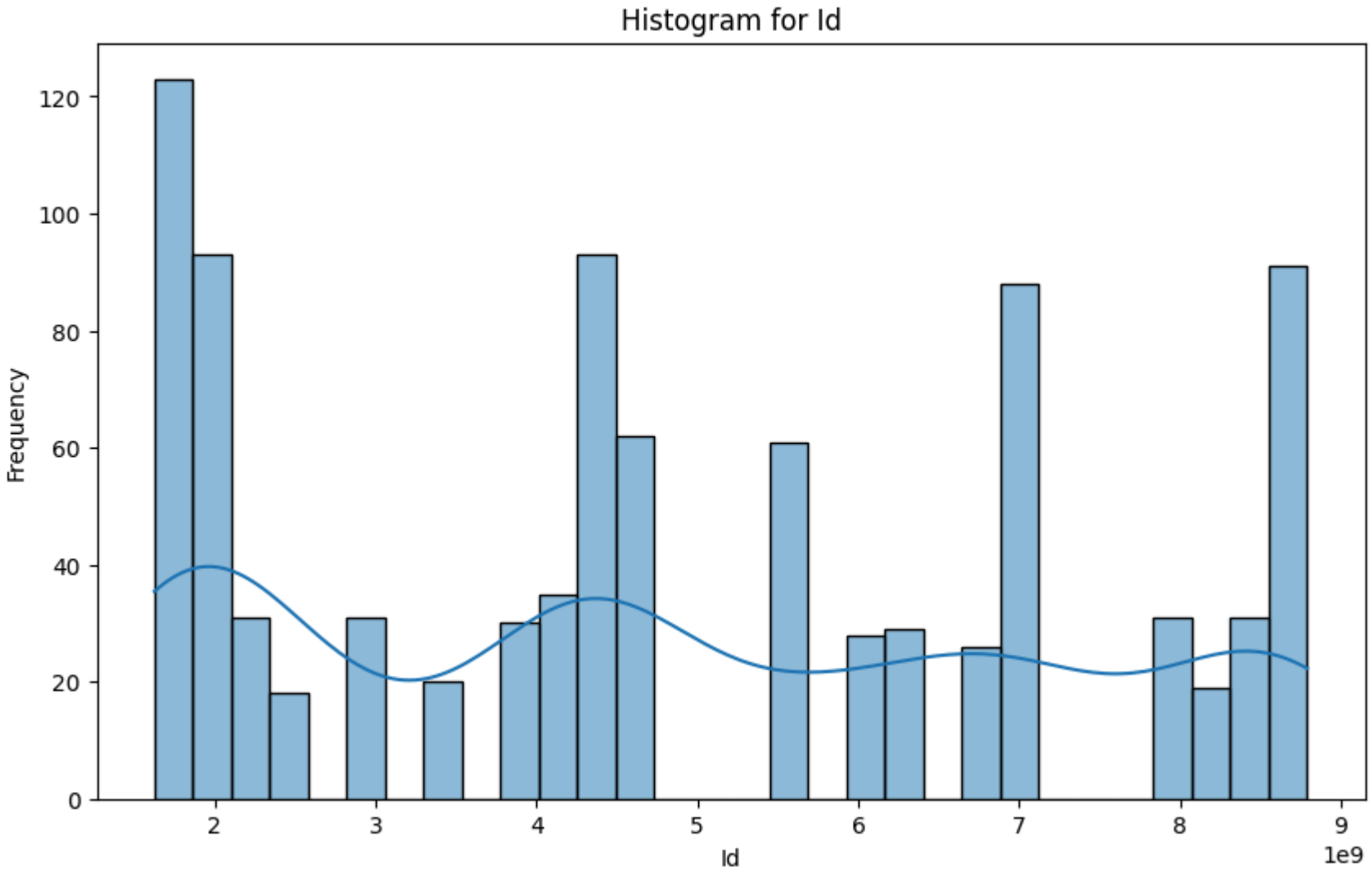

Boxplot for FairlyActiveMinutes (Winsorized)
FairlyActiveMinutes
0
10
20
30
40
50

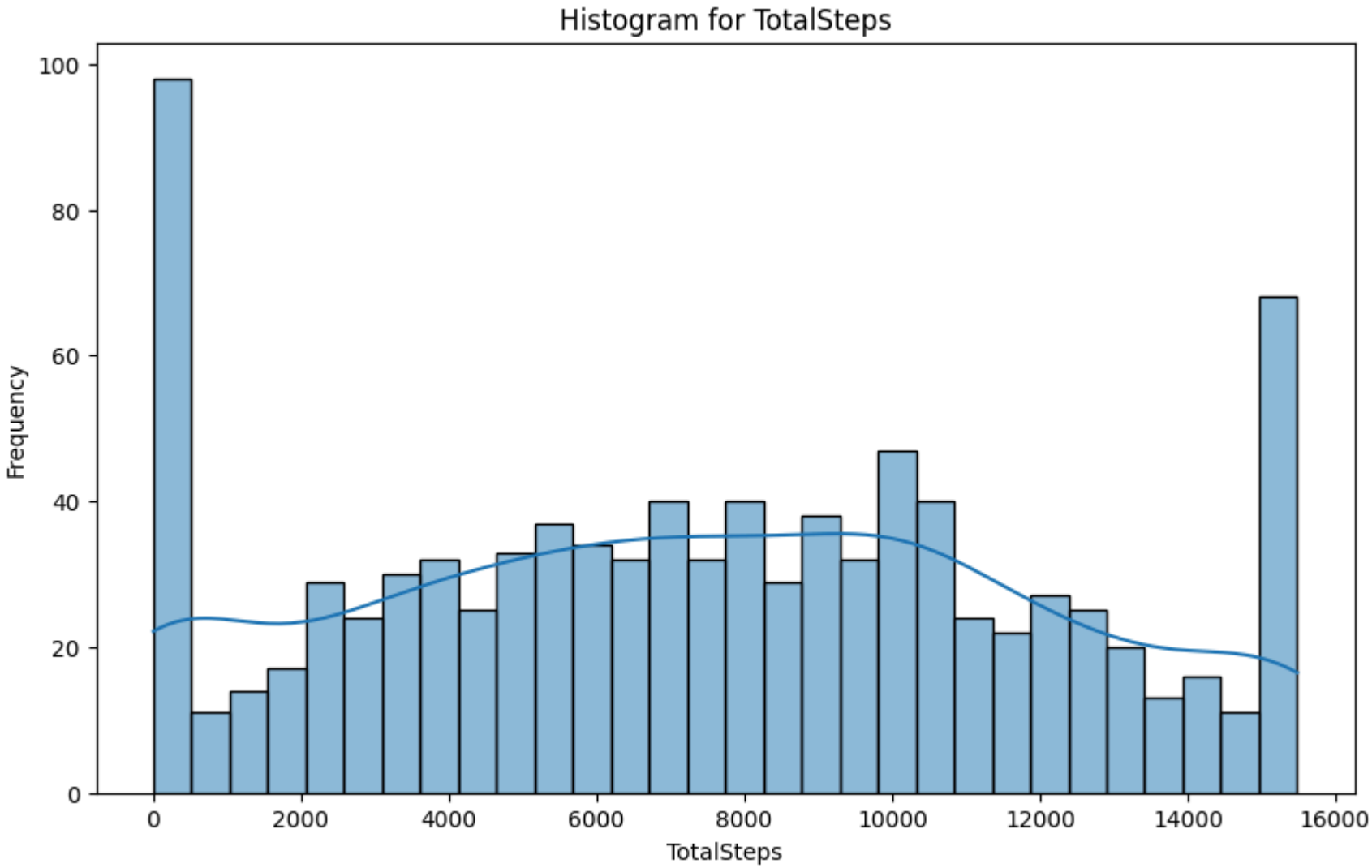

Boxplot for LightlyActiveMinutes (Winsorized)
LightlyActiveMinutes
350
300
250
200
150
100
50
0

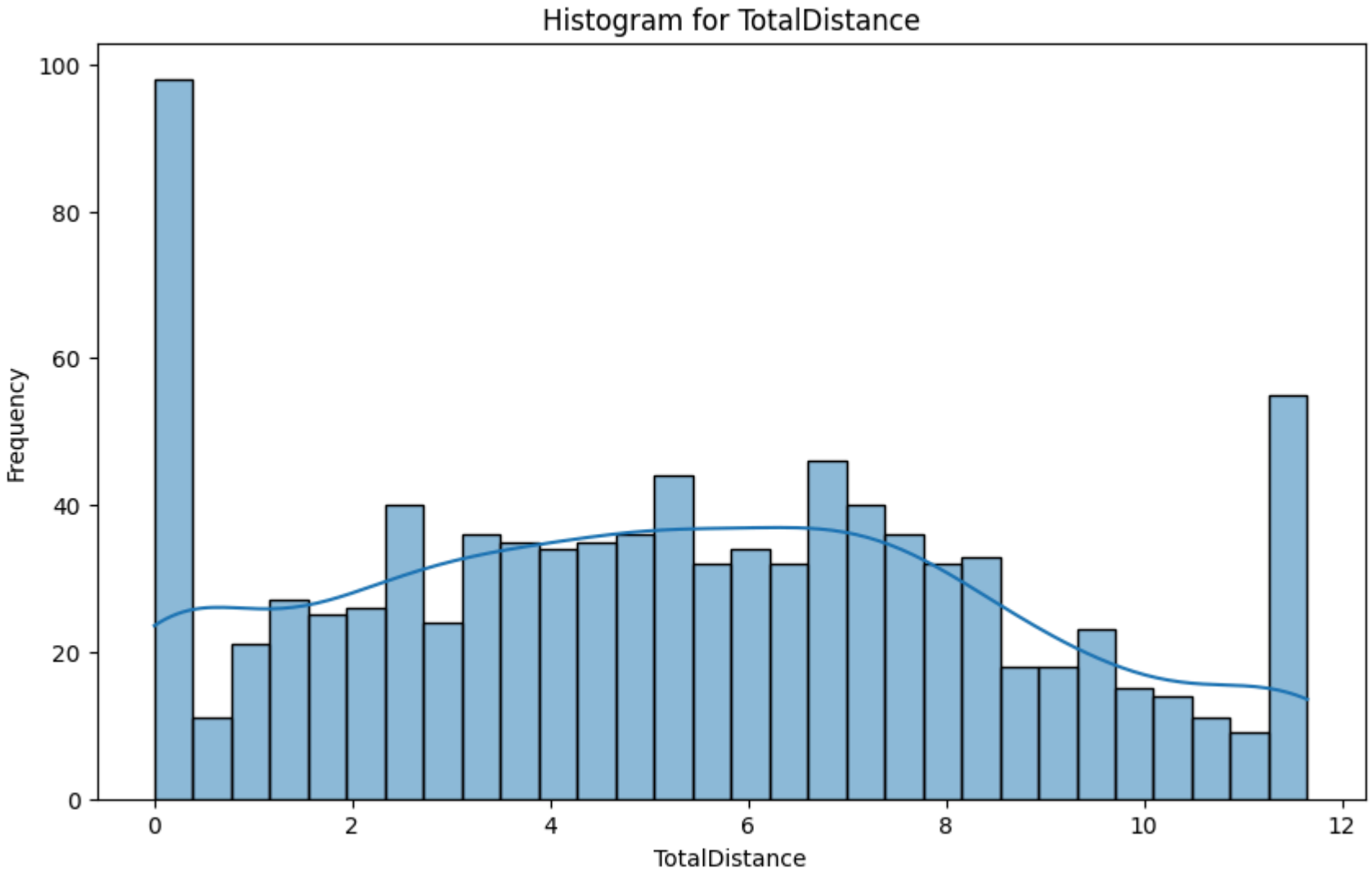

Boxplot for SedentaryMinutes (Winsorized)
SedentaryMinutes
1400
1200
1000
800
600

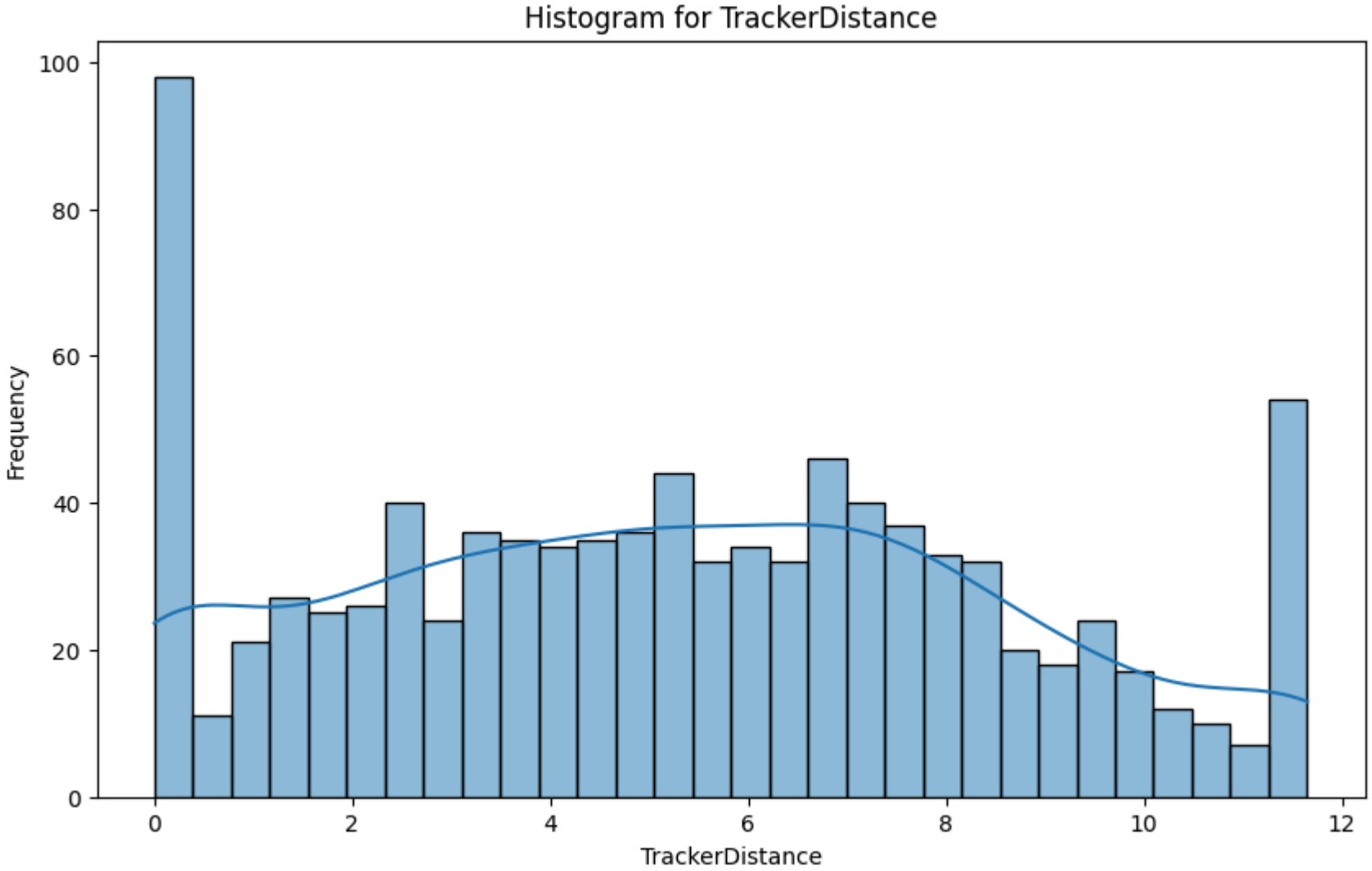


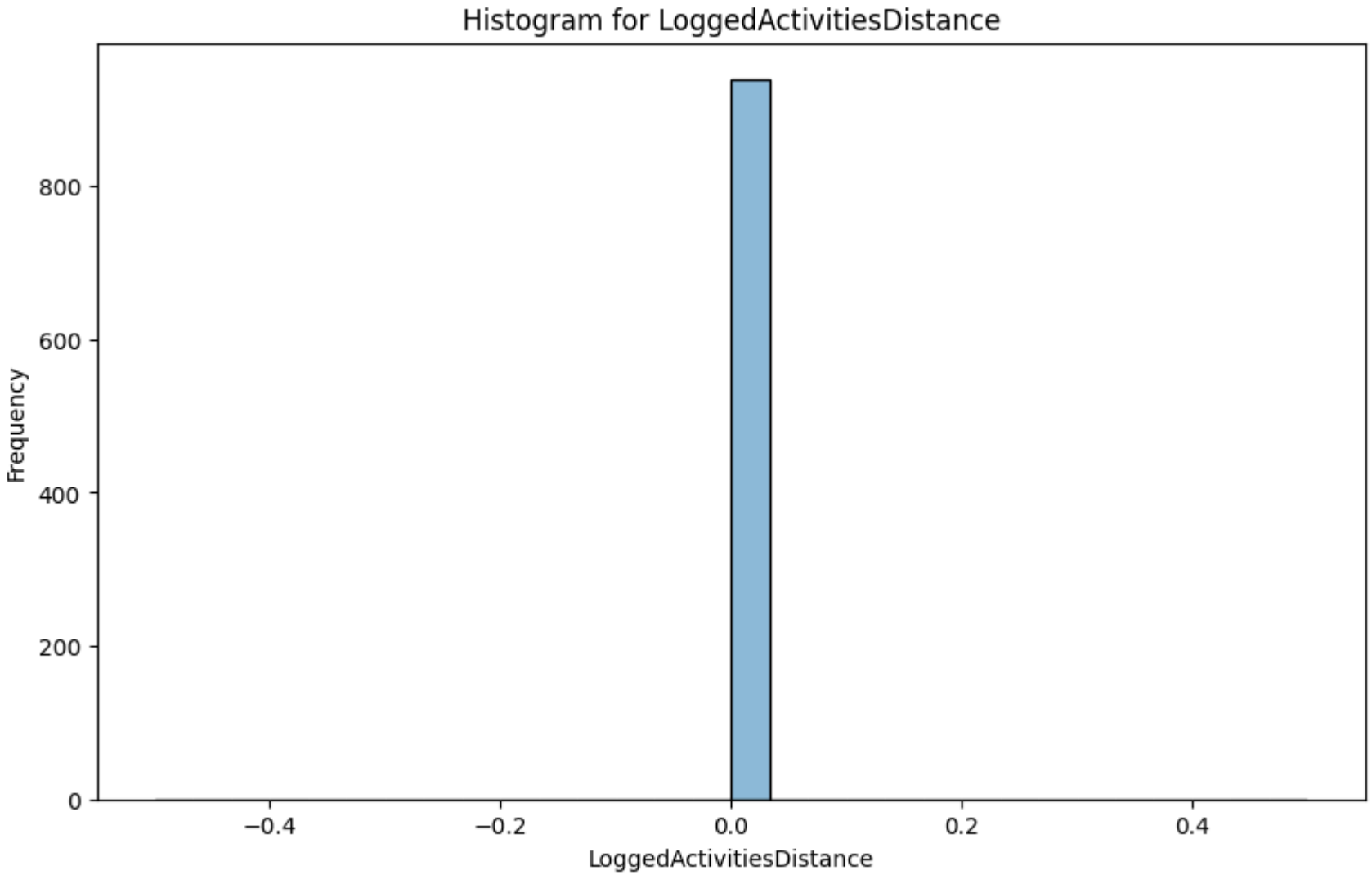
In [9]:
```
# Iterate over each column and create a histogram
for column in winsorized_data.columns:
    plt.figure(figsize=(10, 6))
    sns.histplot(winsorized_data[column], bins=30, kde=True)
    plt.title(f'Histogram for {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

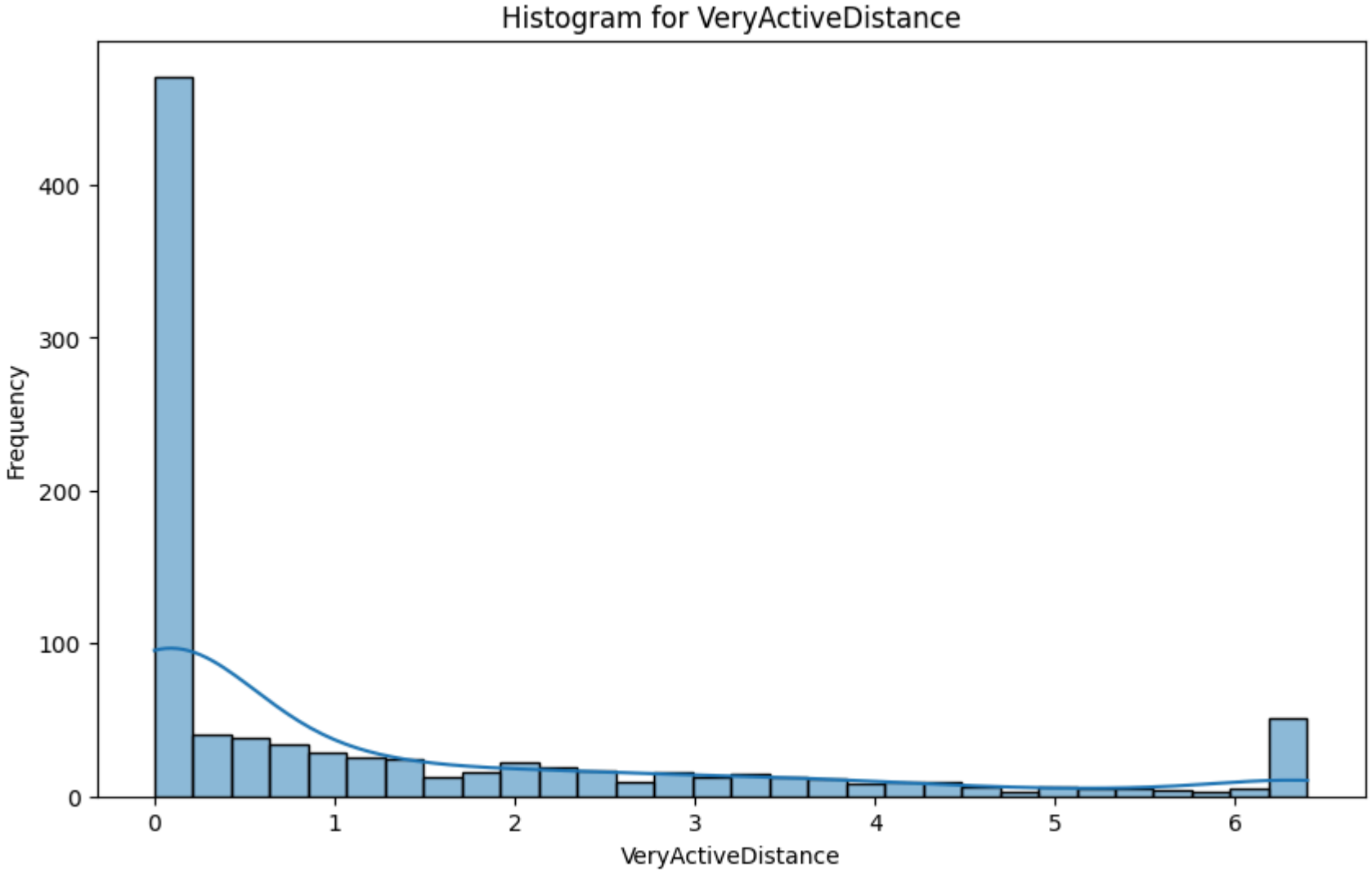

Histogram for Id
Frequency
0
20
40
60
80
100
120
2
3
4
5
6
7
8
9
Id
1e9

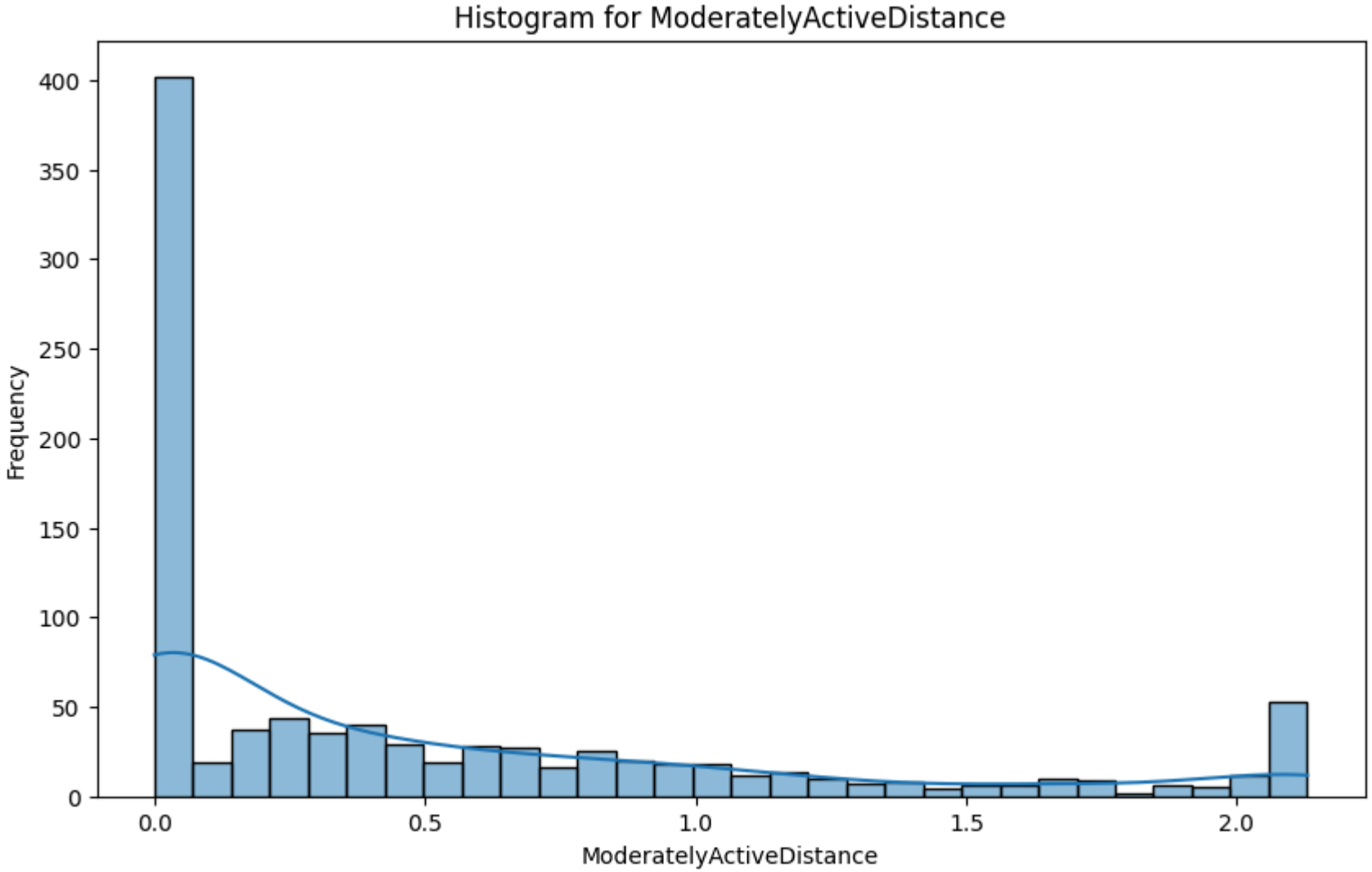

Histogram for TotalSteps
Frequency
TotalSteps
0
20
40
60
80
100
0
2000
4000
6000
8000
10000
12000
14000
16000

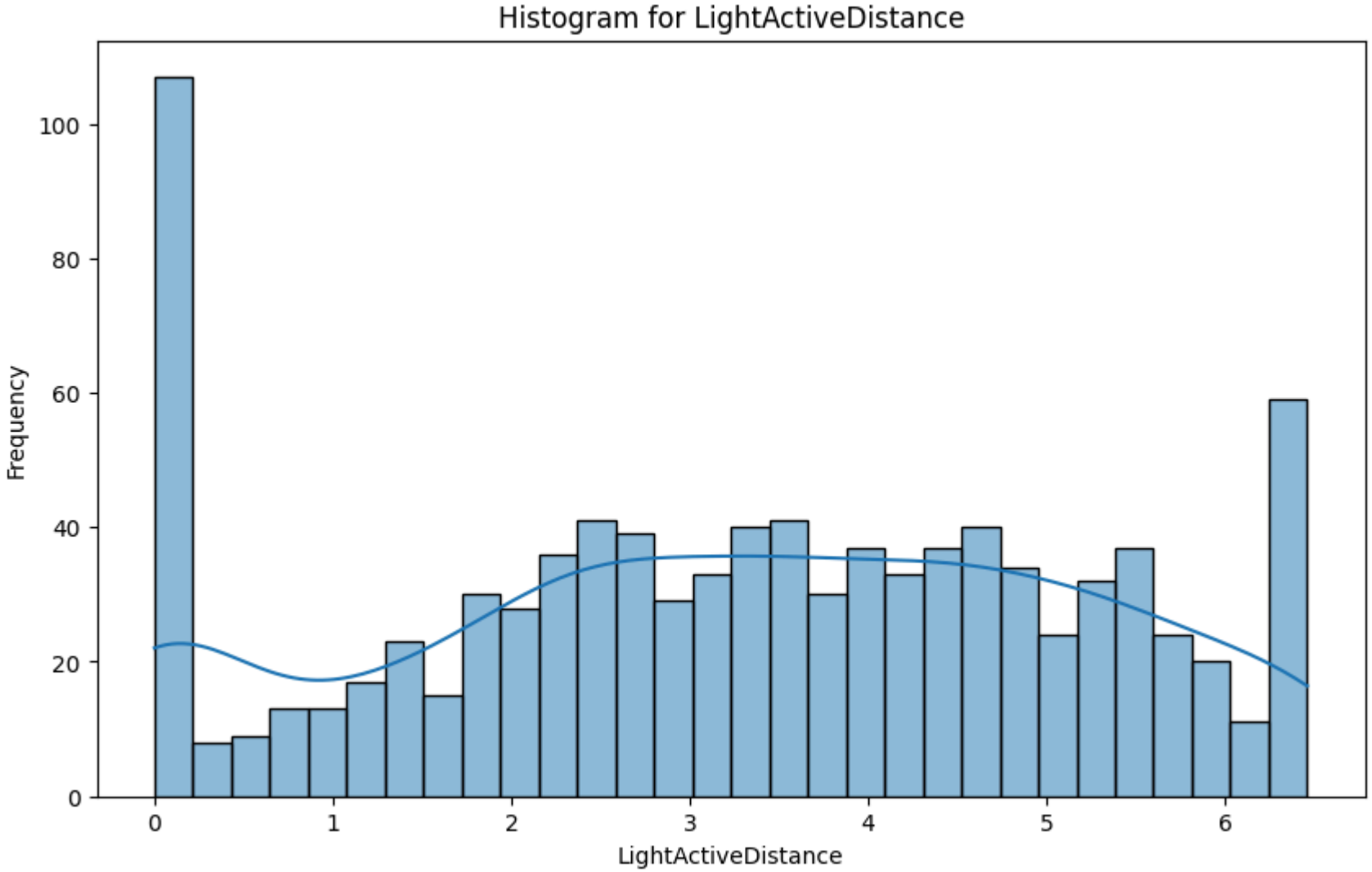

Histogram for TotalDistance
Frequency
TotalDistance
0
20
40
60
80
100
0
2
4
6
8
10
12

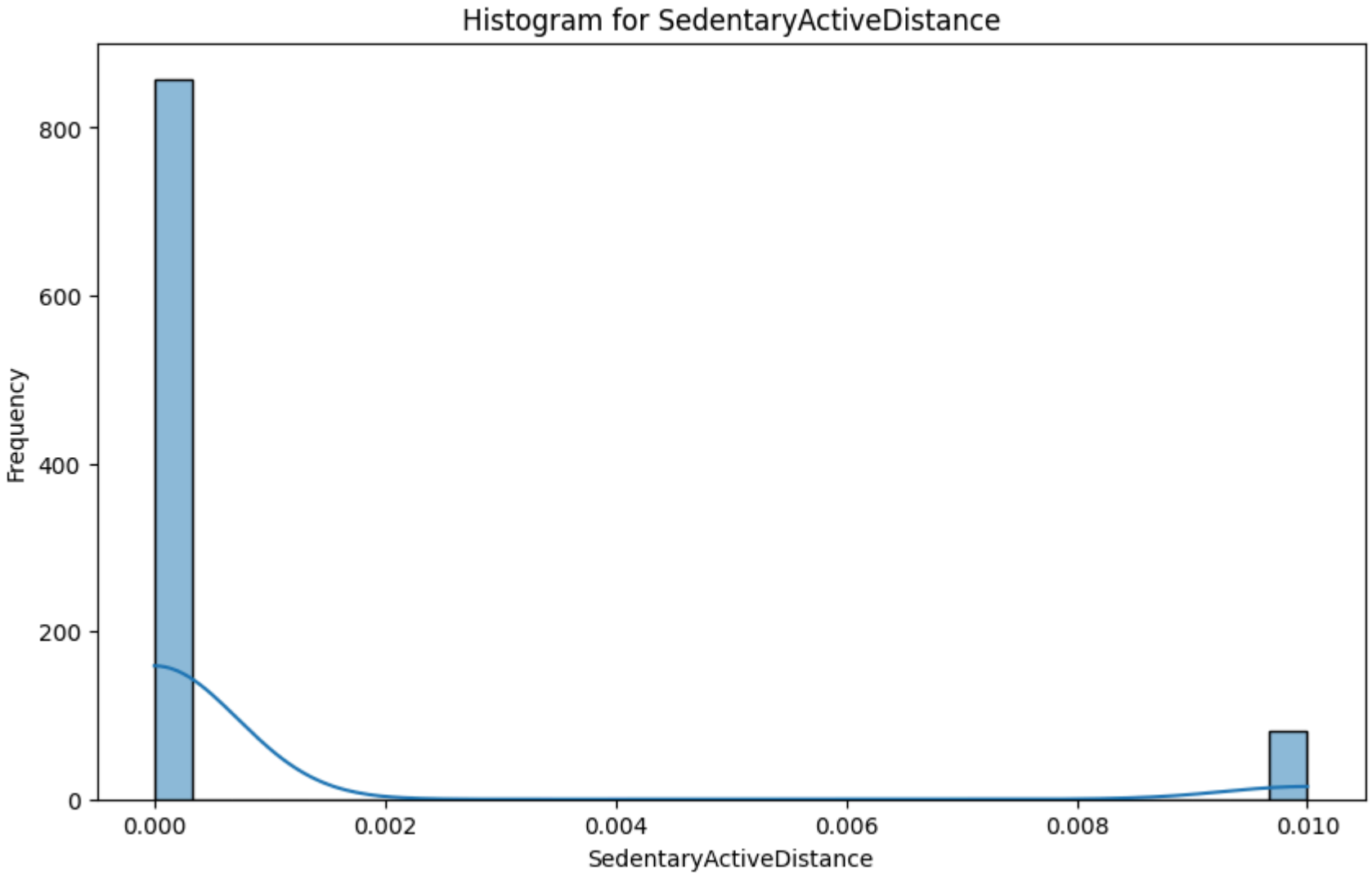

Histogram for TrackerDistance
Frequency
TrackerDistance
0
20
40
60
80
100
0
2
4
6
8
10
12

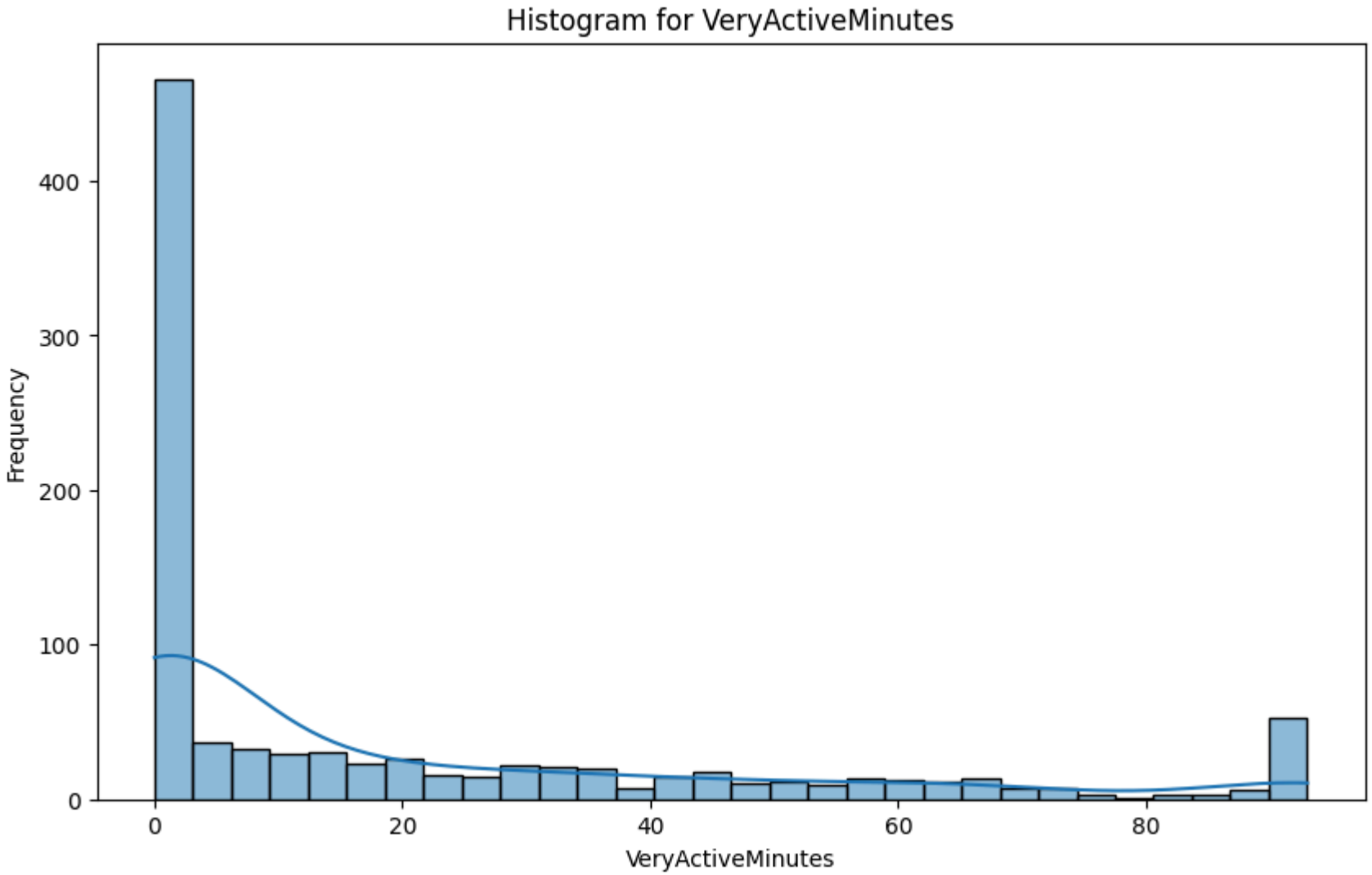

Histogram for LoggedActivitiesDistance
Frequency
800
600
400
200
0
−0.4
−0.2
0.0
0.2
0.4
LoggedActivitiesDistance

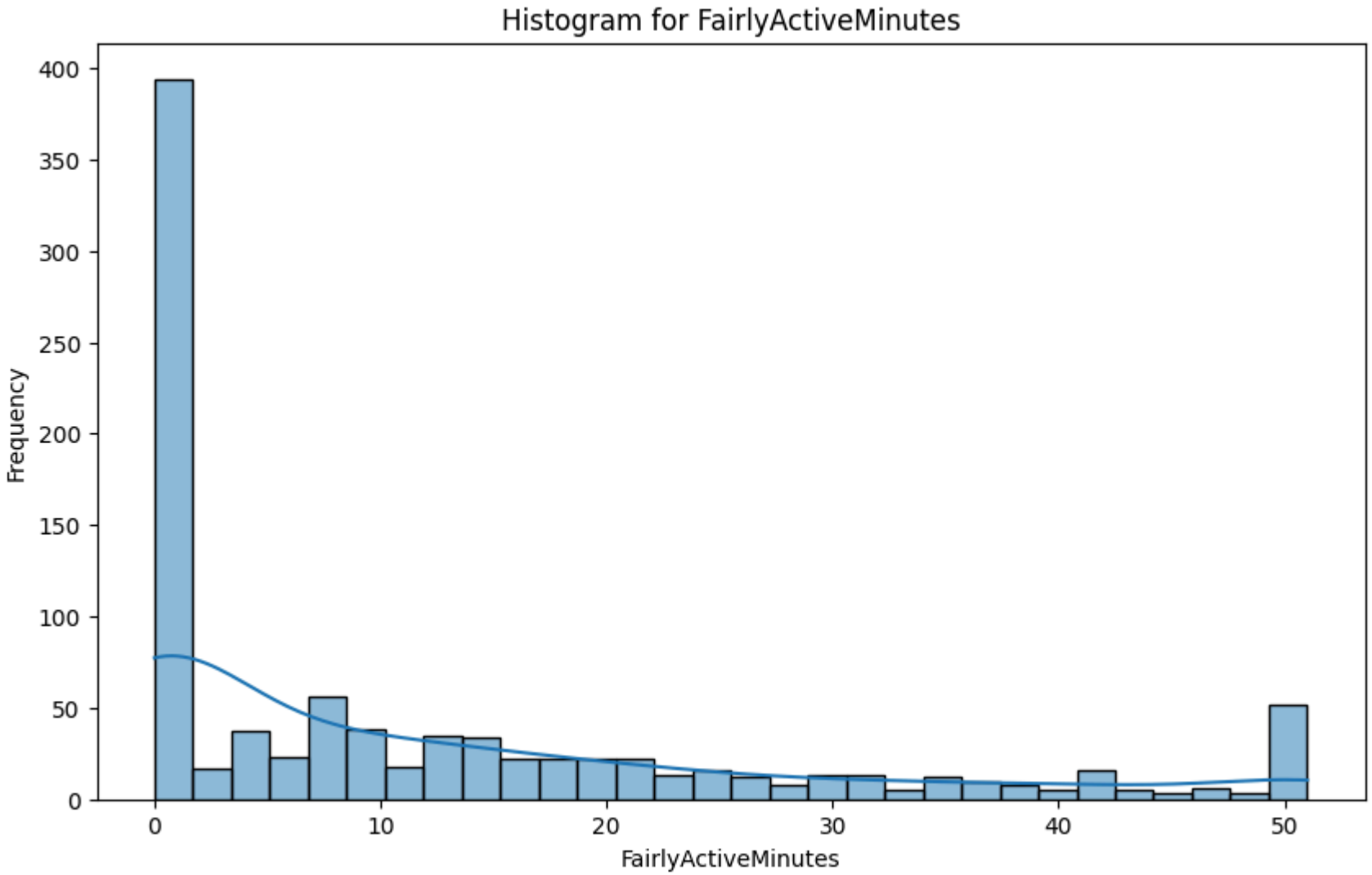

Histogram for VeryActiveDistance
Frequency
0
100
200
300
400
0
1
2
3
4
5
6
VeryActiveDistance

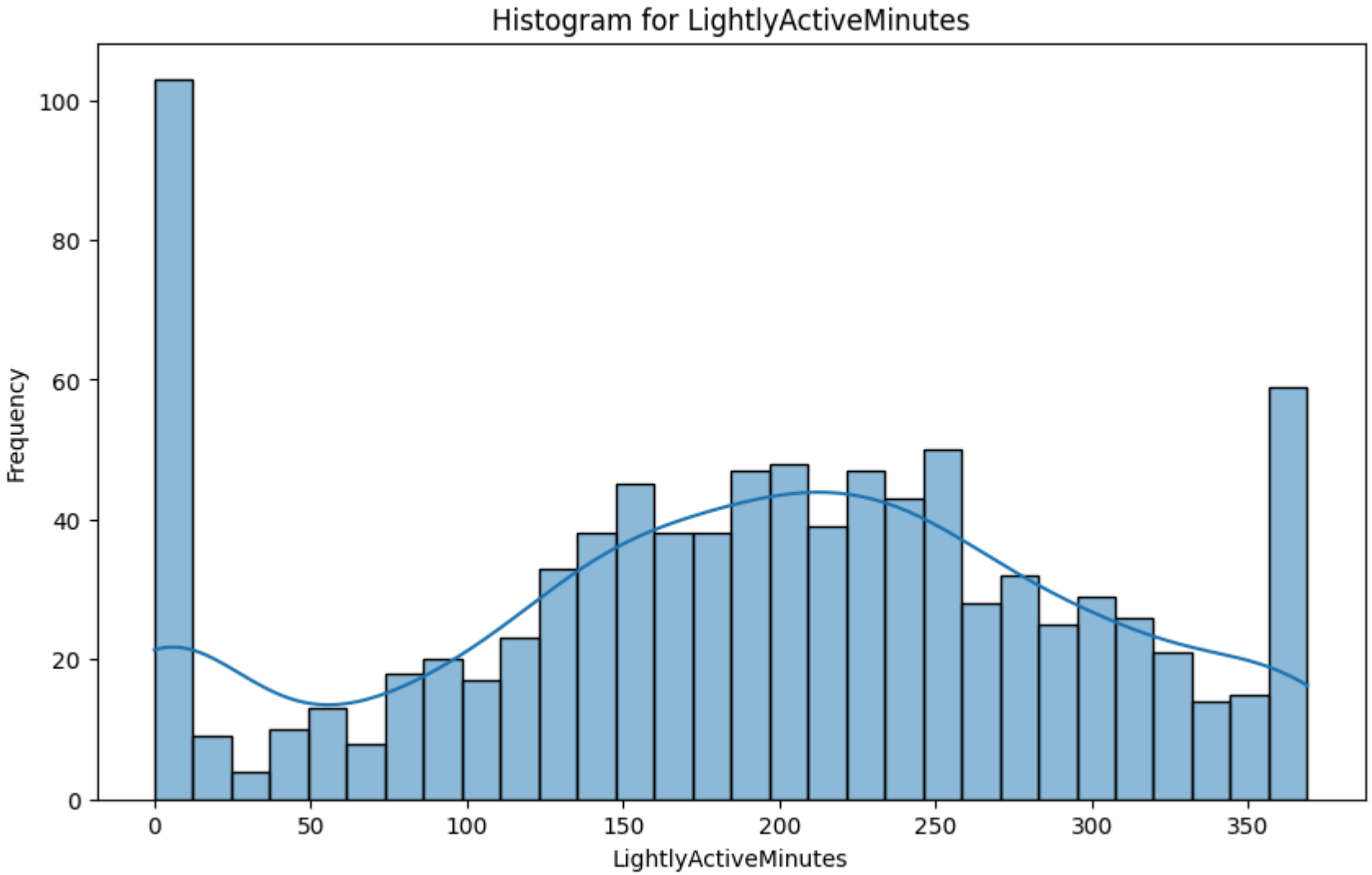

Histogram for ModeratelyActiveDistance
Frequency
400
350
300
250
200
150
100
50
0
0.0
0.5
1.0
1.5
2.0
ModeratelyActiveDistance

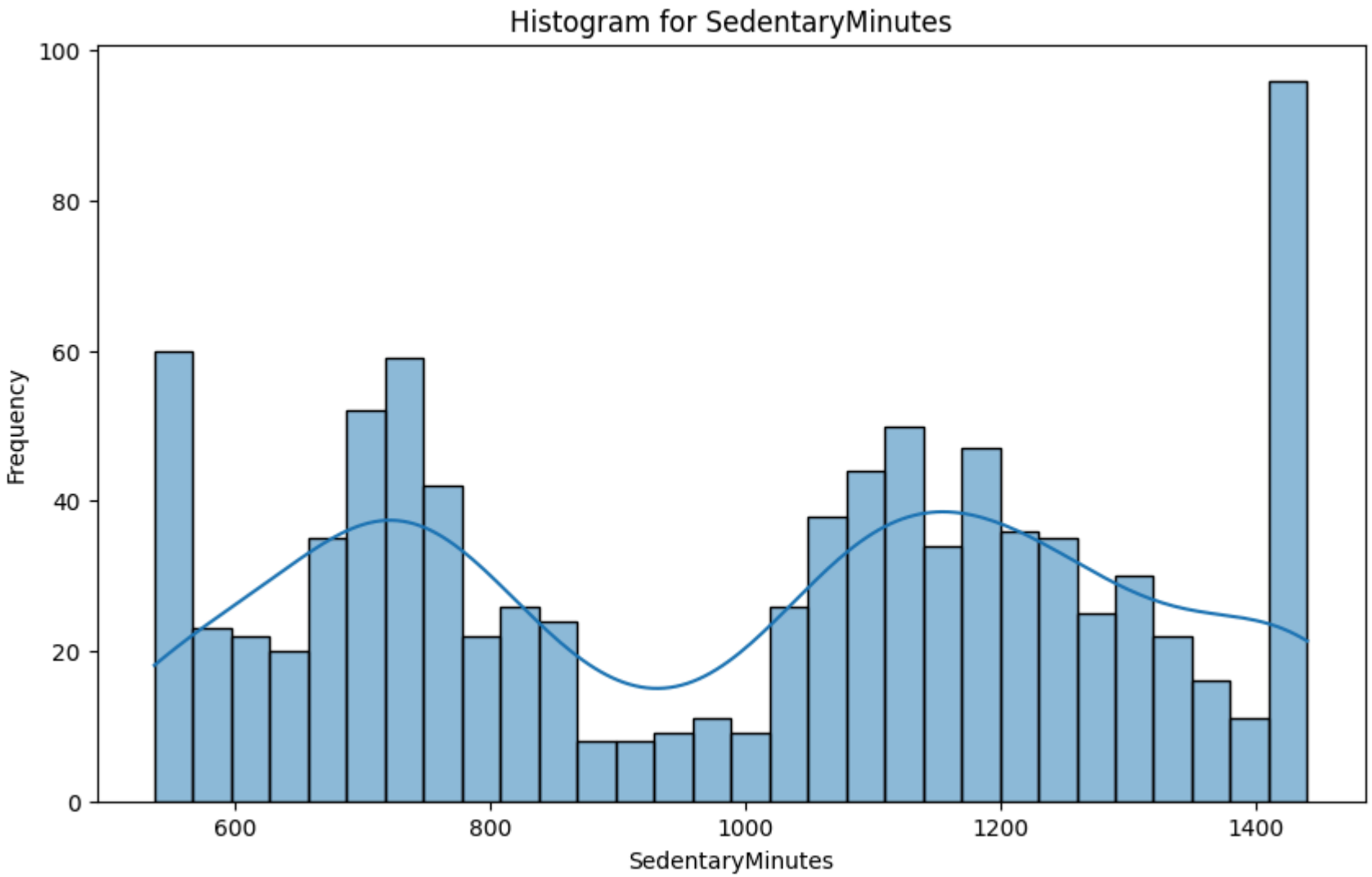

Histogram for LightActiveDistance
Frequency
LightActiveDistance
0
20
40
60
80
100
0
1
2
3
4
5
6

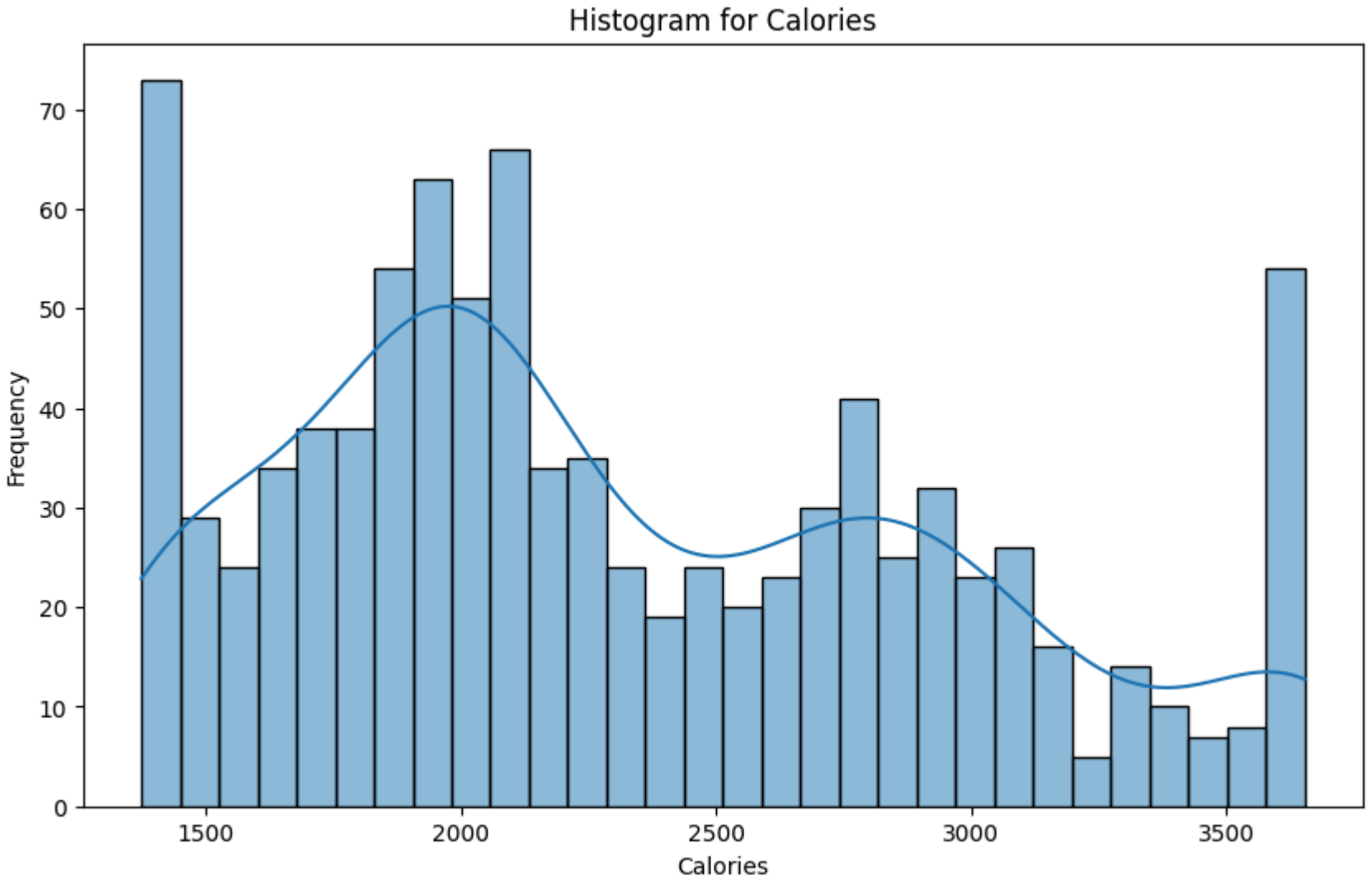

Histogram for SedentaryActiveDistance
Frequency
800
600
400
200
0
0.000
0.002
0.004
0.006
0.008
0.010
SedentaryActiveDistance


Histogram for VeryActiveMinutes
Frequency
0
100
200
300
400
0
20
40
60
80
VeryActiveMinutes

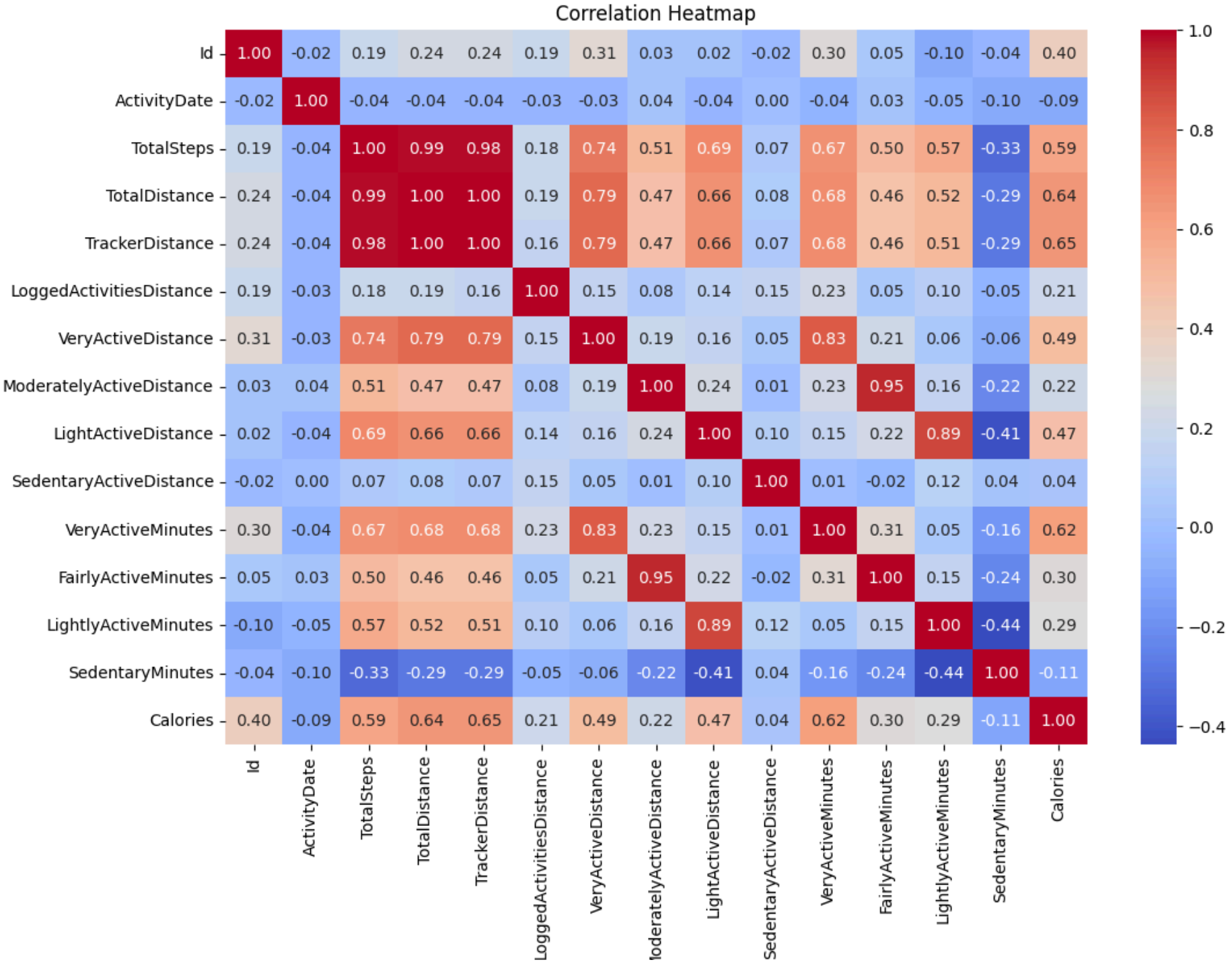

Histogram for FairlyActiveMinutes
Frequency
FairlyActiveMinutes
0
50
100
150
200
250
300
350
400
0
10
20
30
40
50


Histogram for LightlyActiveMinutes
Frequency
LightlyActiveMinutes
0
20
40
60
80
100
0
50
100
150
200
250
300
350

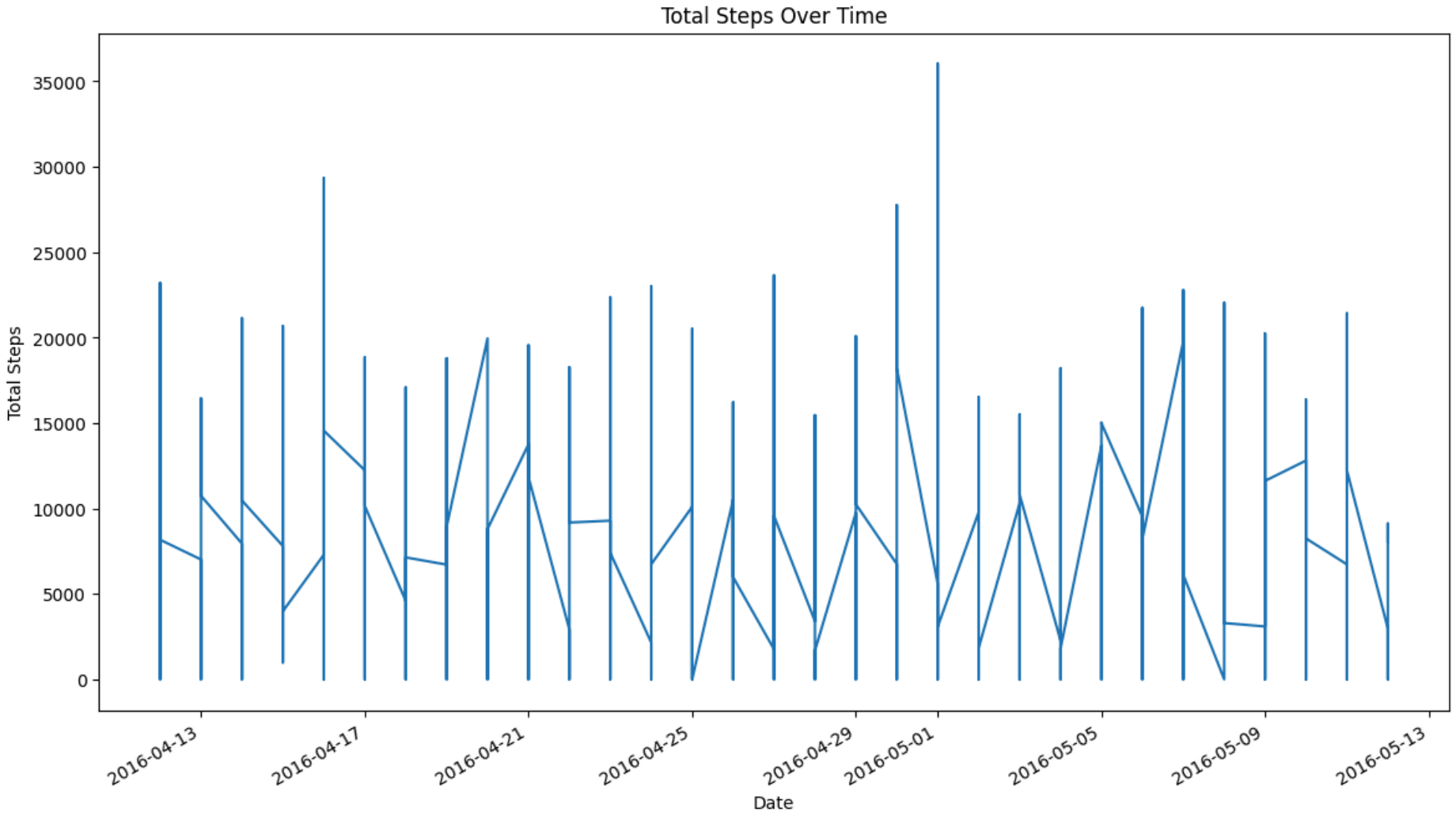

Histogram for SedentaryMinutes
Frequency
SedentaryMinutes
0
20
40
60
80
100
600
800
1000
1200
1400



In [10]:
```
# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```


Correlation Heatmap
Id 1.00 -0.02 0.19 0.24 0.24 0.19 0.31 0.03 0.02 -0.02 0.30 0.05 -0.10 -0.04 0.40
ActivityDate -0.02 1.00 -0.04 -0.04 -0.04 -0.03 -0.03 0.04 -0.04 0.00 -0.04 0.03 -0.05 -0.10 -0.09
TotalSteps 0.19 -0.04 1.00 0.99 0.98 0.18 0.74 0.51 0.69 0.07 0.67 0.50 0.57 -0.33 0.59
TotalDistance 0.24 -0.04 0.99 1.00 1.00 0.19 0.79 0.47 0.66 0.08 0.68 0.46 0.52 -0.29 0.64
TrackerDistance 0.24 -0.04 0.98 1.00 1.00 0.16 0.79 0.47 0.66 0.07 0.68 0.46 0.51 -0.29 0.65
LoggedActivitiesDistance 0.19 -0.03 0.18 0.19 0.16 1.00 0.15 0.08 0.14 0.15 0.23 0.05 0.10 -0.05 0.21
VeryActiveDistance 0.31 -0.03 0.74 0.79 0.79 0.15 1.00 0.19 0.16 0.05 0.83 0.21 0.06 -0.06 0.49
ModeratelyActiveDistance 0.03 0.04 0.51 0.47 0.47 0.08 0.19 1.00 0.24 0.01 0.23 0.95 0.16 -0.22 0.22
LightActiveDistance 0.02 -0.04 0.69 0.66 0.66 0.14 0.16 0.24 1.00 0.10 0.15 0.22 0.89 -0.41 0.47
SedentaryActiveDistance -0.02 0.00 0.07 0.08 0.07 0.15 0.05 0.01 0.10 1.00 0.01 -0.02 0.12 0.04 0.04
VeryActiveMinutes 0.30 -0.04 0.67 0.68 0.68 0.23 0.83 0.23 0.15 0.01 1.00 0.31 0.05 -0.16 0.62
FairlyActiveMinutes 0.05 0.03 0.50 0.46 0.46 0.05 0.21 0.95 0.22 -0.02 0.31 1.00 0.15 -0.24 0.30
LightlyActiveMinutes -0.10 -0.05 0.57 0.52 0.51 0.10 0.06 0.16 0.89 0.12 0.05 0.15 1.00 -0.44 0.29
SedentaryMinutes -0.04 -0.10 -0.33 -0.29 -0.29 -0.05 -0.06 -0.22 -0.41 0.04 -0.16 -0.24 -0.44 1.00 -0.11
Calories 0.40 -0.09 0.59 0.64 0.65 0.21 0.49 0.22 0.47 0.04 0.62 0.30 0.29 -0.11 1.00
Id
ActivityDate
TotalSteps
TotalDistance
TrackerDistance
LoggedActivitiesDistance
VeryActiveDistance
ModeratelyActiveDistance
LightActiveDistance
SedentaryActiveDistance
VeryActiveMinutes
FairlyActiveMinutes
LightlyActiveMinutes
SedentaryMinutes
Calories
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

In [11]:
```
# Trends over time (e.g., total steps)
plt.figure(figsize=(14, 8))
data.set_index('ActivityDate')['TotalSteps'].plot()
plt.title('Total Steps Over Time')
plt.xlabel('Date')
plt.ylabel('Total Steps')
plt.show()
```



Feature Engineering

In [12]:
```
# Creating new features
data['Weekday'] = data['ActivityDate'].dt.day_name()
data['TotalActiveMinutes'] = data['VeryActiveMinutes'] + data['FairlyActiveMinutes'] + data['LightlyActiveMinutes']

# Weekly aggregation
weekly_data = data.resample('W', on='ActivityDate').sum()
weekly_data['AverageSteps'] = weekly_data['TotalSteps'] / 7
```

Machine Learning Modeling

In [13]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, mean_absolute_percentage_error, explained_variance



# Define the feature matrix and target variable
X = data[['TotalSteps', 'TotalDistance', 'VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes']]
y = data['Calories']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a RandomForestRegressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
explained_var = explained_variance_score(y_test, y_pred)
medae = median_absolute_error(y_test, y_pred)
```

```
# Print evaluation metrics
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared (R²): {r2}')
print(f'Mean Absolute Percentage Error (MAPE): {mape}')
print(f'Explained Variance Score: {explained_var}')
print(f'Median Absolute Error: {medae}')
```

```
Mean Absolute Error (MAE): 323.06730856114865
Mean Squared Error (MSE): 189271.47669806003
Root Mean Squared Error (RMSE): 435.05341821213176
R-squared (R²): 0.5995728730580042
Mean Absolute Percentage Error (MAPE): 3.948409677979897e+16
Explained Variance Score: 0.5998322162545783
Median Absolute Error: 218.43000000000006
```

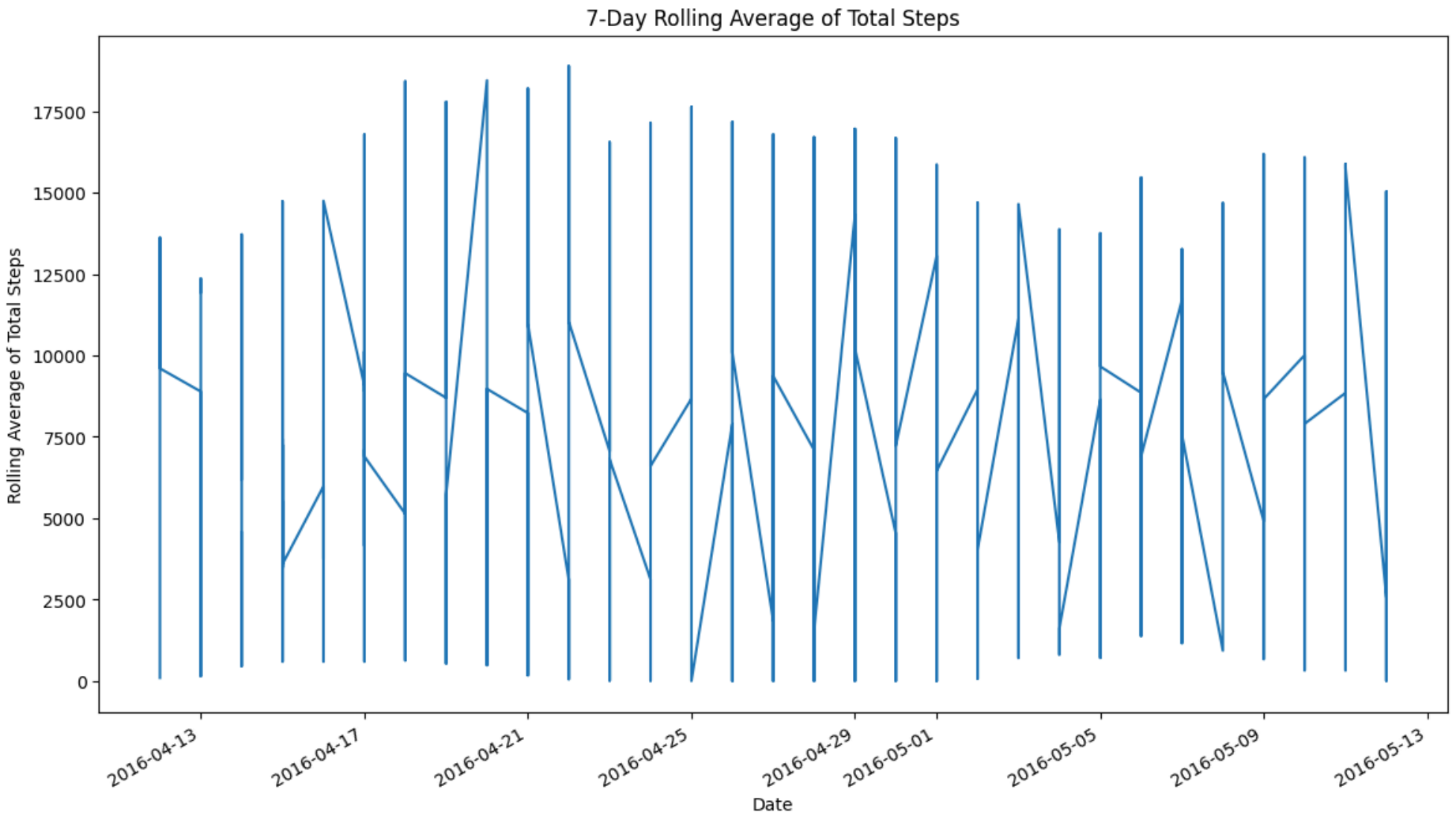
Interpretation

The model's performance, indicated by the evaluation metrics, shows that it has a mean absolute error (MAE) of approximately 323 calories, suggesting that, on average, the model's predictions deviate from the actual values by this amount. The mean squared error (MSE) is 189,271, highlighting the presence of some larger errors due to the squaring effect. The root mean squared error (RMSE) of 435 calories further underscores the typical magnitude of prediction errors. The R-squared value of 0.60 indicates that the model explains 60% of the variance in the target variable, which shows a moderate level of predictive power. However, the mean absolute percentage error (MAPE) is extremely high, implying issues with the model's accuracy on a relative scale. The explained variance score of 0.60 is consistent with the R-squared value, and the median absolute error of 218 calories suggests that half of the errors are below this threshold, indicating the model's predictions are somewhat reliable but leave room for improvement.

Longitudinal Analysis

In [14]:
```
# Calculate rolling averages to observe trends
data['RollingAvgSteps'] = data['TotalSteps'].rolling(window=7).mean()

# Plot the rolling average of total steps
plt.figure(figsize=(14, 8))
data.set_index('ActivityDate')['RollingAvgSteps'].plot()
plt.title('7-Day Rolling Average of Total Steps')
plt.xlabel('Date')
```

```
plt.ylabel('Rolling Average of Total Steps')
plt.show()
```



The graph shows the 7-day rolling average of total steps taken daily from April 13, 2016, to May 13, 2016. The data demonstrates significant variability, with the rolling average frequently oscillating between higher and lower values. This suggests that the individual or group being tracked experienced inconsistent daily activity levels, possibly due to changes in routine, varying exercise habits, or external factors influencing physical activity. For example, some days might have seen increased steps due to planned exercise or more active days, while other days show a sharp drop, indicating less movement. The lack of a clear upward or downward trend means there isn't a consistent improvement or decline in

daily steps over this period. Instead, the activity levels are quite erratic, which might reflect an irregular lifestyle or periodic bursts of high activity followed by lower activity days.

In [15]:
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.inspection import permutation_importance

# Load the dataset
data = pd.read_csv('/content/dailyActivity_merged.csv')

# Convert 'ActivityDate' to datetime
data['ActivityDate'] = pd.to_datetime(data['ActivityDate'])

# Define the feature matrix and target variable
X = data[['TotalSteps', 'TotalDistance', 'VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes']]
y = data['Calories']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a RandomForestRegressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Feature importance
importances = model.feature_importances_
features = X.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance': importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

print(importance_df)
```

```
                Feature  Importance
2     VeryActiveMinutes    0.404255
1         TotalDistance    0.351049
0            TotalSteps    0.123771
4  LightlyActiveMinutes    0.071097
3   FairlyActiveMinutes    0.049827
```

The feature importance values provide insights into which features are most influential in predicting the target variable (Calories burned). Here's an interpretation of the feature importance rankings:

1. **Very Active Minutes (Importance: 0.404255)**: This feature has the highest importance, indicating that the amount of time spent in very active minutes is the most significant predictor of calories burned. This makes sense because high-intensity activities typically burn more calories.

2. **Total Distance (Importance: 0.351049)**: The second most important feature is the total distance covered. This suggests that the distance traveled, likely through walking or running, plays a crucial role in determining the calories burned.

3. **Total Steps (Importance: 0.123771)**: Total steps taken is the third most important feature. While important, it has less influence compared to very active minutes and total distance, possibly because it includes both low-intensity and high-intensity activities.

4. **Lightly Active Minutes (Importance: 0.071097)**: This feature has a lower importance, indicating that time spent in light activity has some effect on calorie burn but is less significant compared to the more intense activity metrics.

5. **Fairly Active Minutes (Importance: 0.049827)**: The least important feature in this model is the time spent in fairly active minutes. While it still contributes to the prediction, its impact is minor compared to the other features.

Overall Interpretation

The model suggests that high-intensity activities (measured by very active minutes) and total distance traveled are the most crucial factors in predicting calories burned. Steps and light activity contribute to the prediction but are less significant. This insight can guide personalized recommendations, emphasizing the importance of increasing high-intensity activities and overall distance traveled to enhance cardiovascular fitness and overall calorie expenditure.

In [16]:
```
def generate_recommendations(user_data):
    recommendations = []

    if user_data['TotalSteps'] < 5000:
        recommendations.append("Try to increase your daily steps to at least 5000.")
    if user_data['VeryActiveMinutes'] < 20:
        recommendations.append("Incorporate more high-intensity activities, aiming for at least 20 minutes a day.")
    if user_data['LightlyActiveMinutes'] > 300:
        recommendations.append("Balance light activities with more moderate or high-intensity activities.")

    if not recommendations:
        recommendations.append("Keep up the good work! Continue with your current routine for maintaining cardiovascular health."

    return recommendations
```

```
# Example user data
user_data_example = {
    'TotalSteps': 4000,
    'TotalDistance': 2.5,
    'VeryActiveMinutes': 15,
    'FairlyActiveMinutes': 30,
    'LightlyActiveMinutes': 320
}

# Generate recommendations
recommendations = generate_recommendations(user_data_example)
for rec in recommendations:
    print(rec)
```

```
Try to increase your daily steps to at least 5000.
Incorporate more high-intensity activities, aiming for at least 20 minutes a day.
Balance light activities with more moderate or high-intensity activities.
```

Conclusion

This analysis utilized a subset of the available datasets in the provided corpus of dataset to explore the relationship between physical activity metrics and calories burned, a proxy for cardiovascular health outcomes. The features included were TotalSteps, TotalDistance, VeryActiveMinutes, FairlyActiveMinutes, and LightlyActiveMinutes. From the model evaluation, the key findings are:

1. **Feature Importance**:
    - **Very Active Minutes** and **Total Distance** are the most significant predictors of calories burned, highlighting the importance of high-intensity activities and overall distance traveled in determining energy expenditure.
    - **Total Steps** also contributes to the prediction but to a lesser extent.
    - **Lightly Active Minutes** and **Fairly Active Minutes** have minor impacts on the model's predictions.
2. **Model Performance**:
    - The model explains approximately 60% of the variance in calorie burn, as indicated by the R-squared value.
    - The mean absolute error (MAE) of about 323 calories and the root mean squared error (RMSE) of approximately 435 calories suggest that there is a moderate level of error in the predictions.

- The high mean absolute percentage error (MAPE) suggests that the model may have difficulty accurately predicting lower calorie burns or may be sensitive to extreme values.

Limitations

It is important to note that not all available features were used in this analysis. Features such as TrackerDistance, LoggedActivitiesDistance, VeryActiveDistance, ModeratelyActiveDistance, LightActiveDistance, and SedentaryMinutes were not included. This omission could limit the comprehensiveness of the analysis and the insights derived. Including these features might improve the model's accuracy and provide a more nuanced understanding of the relationship between physical activity and cardiovascular health outcomes.

Recommendations for Future Analysis

- **Include More Features**: Future analyses should incorporate all available features to capture the full range of activities and their impacts on health outcomes.
- **Explore Additional Models**: Different machine learning models and techniques might better capture the complexities in the data.
- **Longitudinal Analysis**: Analyzing data over longer periods can help identify trends and patterns that might not be apparent in a short-term analysis.
- **Personalized Recommendations**: Based on a more comprehensive model, personalized health recommendations can be made more accurately to optimize cardiovascular fitness and overall well-being.

In summary, while the current analysis provides valuable insights, a more comprehensive approach using all available data is necessary for a deeper understanding and more accurate predictions.