```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import dendrogram, linkage


df = pd.read_csv('/content/simulated_health_wellness_data (3).csv')
df.head()
```

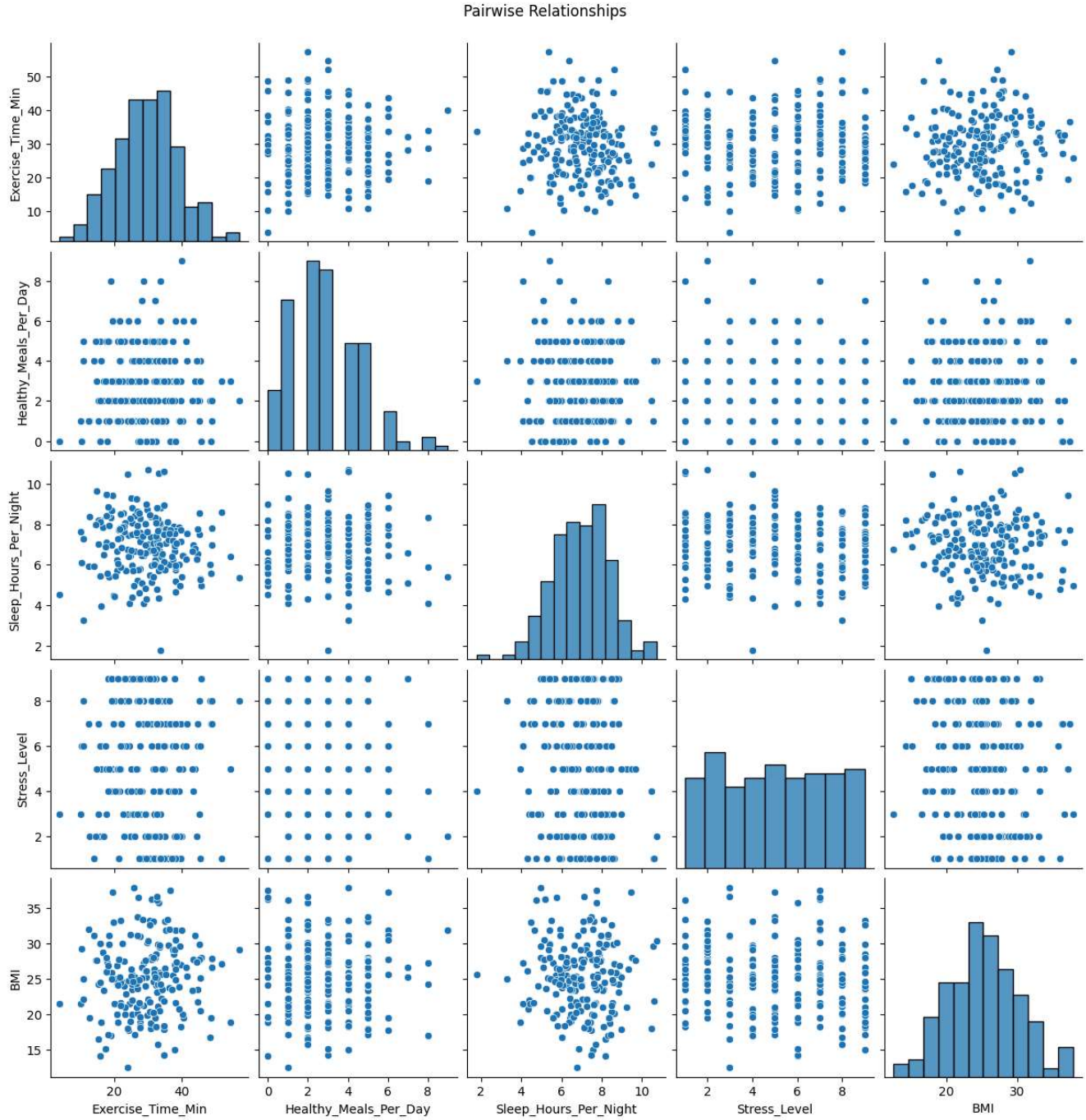
	Exercise_Time_Min	Healthy_Meals_Per_Day	Sleep_Hours_Per_Night	Stress_Level	BMI
0	34.967142	5	7.618856	2	33.068556
1	28.617357	8	4.105473	7	27.267672
2	36.476885	4	6.024123	1	23.779217
3	45.230299	1	8.565319	8	29.820436
4	27.658466	3	8.301648	3	30.947352

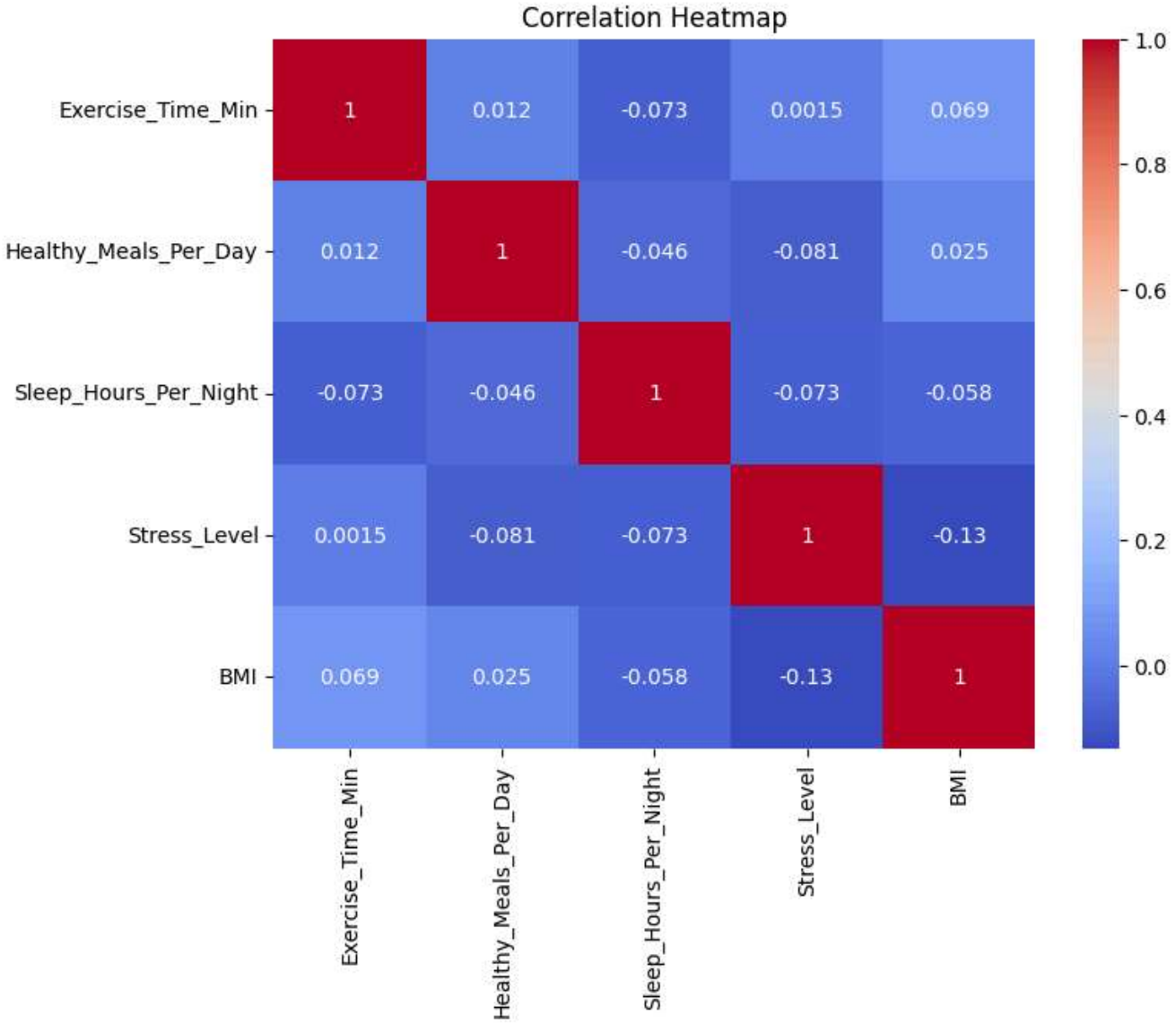
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print(df.describe())

sns.pairplot(df)
plt.suptitle("Pairwise Relationships", y=1.02)
plt.show()

plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

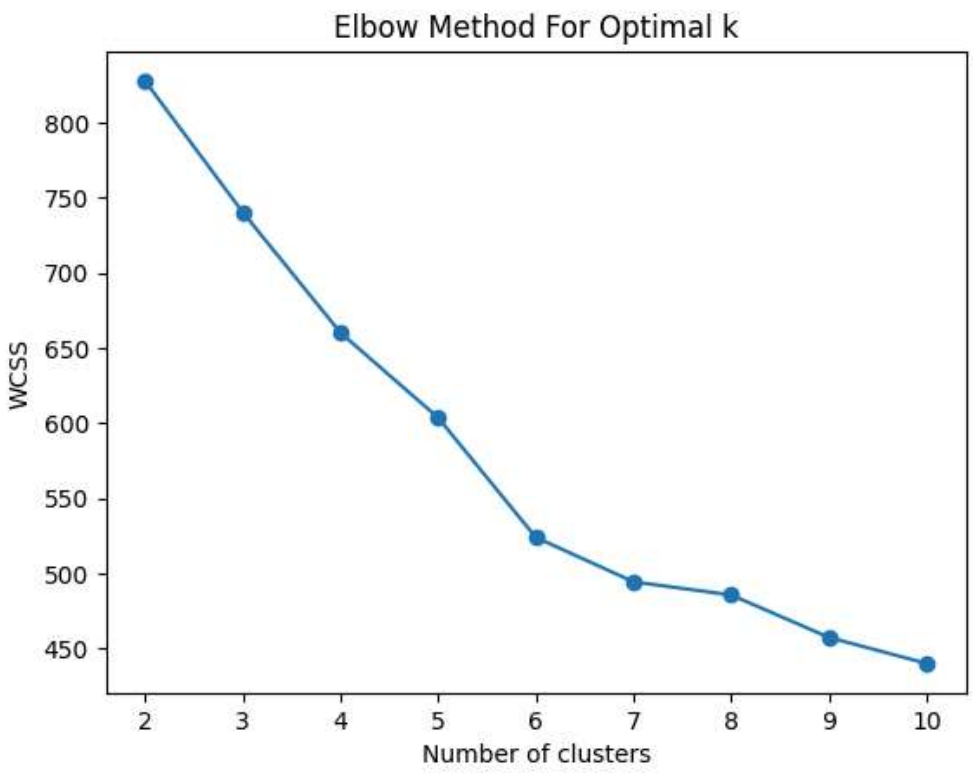
	Exercise_Time_Min	Healthy_Meals_Per_Day	Sleep_Hours_Per_Night	\
count	200.000000	200.000000	200.000000	
mean	29.592290	2.875000	6.933582	
std	9.310039	1.815449	1.422471	
min	3.802549	0.000000	1.778787	
25%	22.948723	2.000000	5.967243	
50%	29.958081	3.000000	6.972331	
75%	35.008525	4.000000	7.886509	
max	57.201692	9.000000	10.708419	

	Stress_Level	BMI
count	200.000000	200.000000
mean	4.995000	25.150008
std	2.605556	5.070778
min	1.000000	12.502971
25%	3.000000	21.458196
50%	5.000000	25.155662
75%	7.000000	28.011155
max	9.000000	37.898547



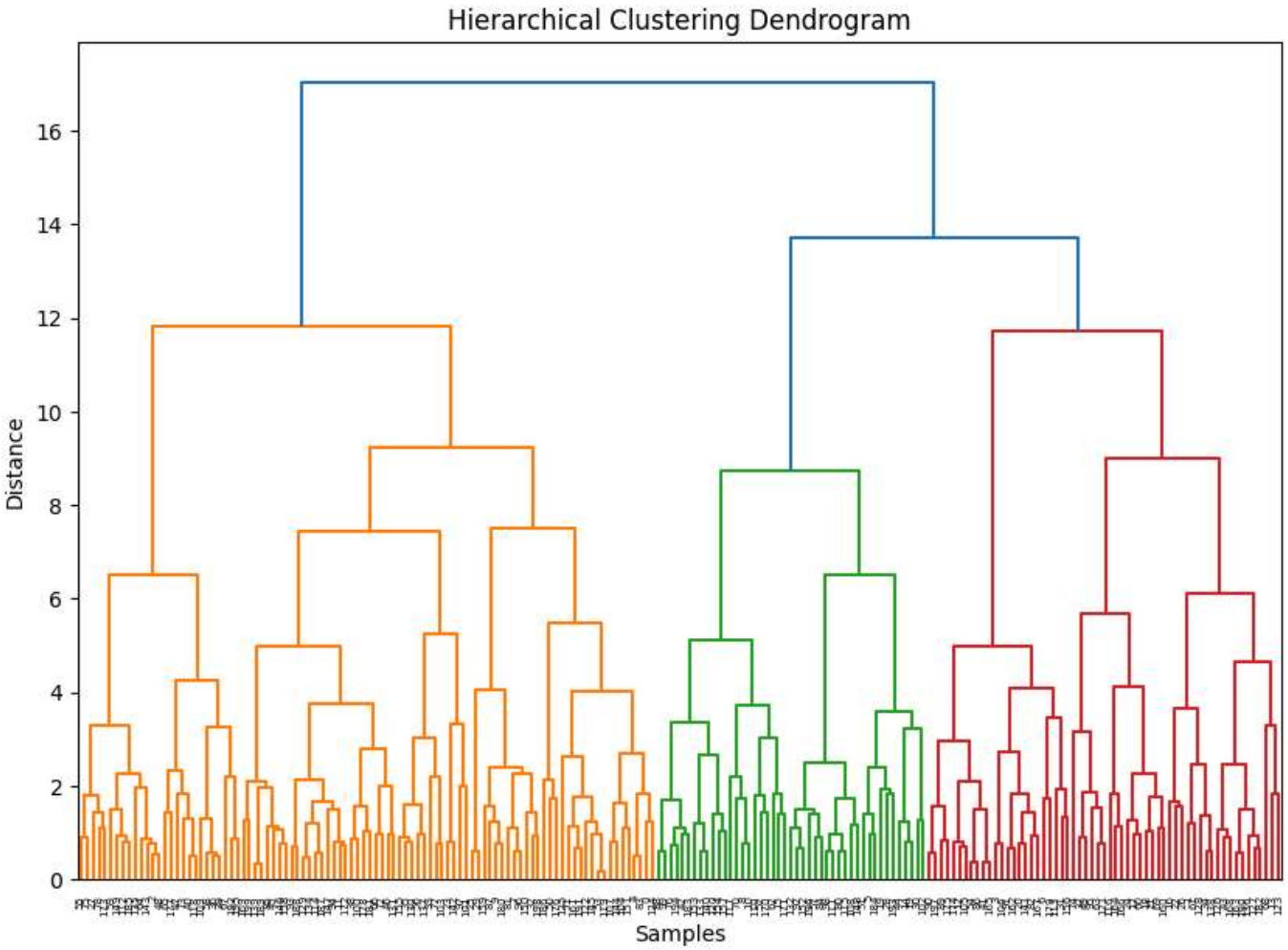

Correlation Heatmap
Exercise_Time_Min
Healthy_Meals_Per_Day
Sleep_Hours_Per_Night
Stress_Level
BMI
1
0.012
-0.073
0.0015
0.069
0.012
1
-0.046
-0.081
0.025
-0.073
-0.046
1
-0.073
-0.058
0.0015
-0.081
-0.073
1
-0.13
0.069
0.025
-0.058
-0.13
1
1.0
0.8
0.6
0.4
0.2
0.0

```
X = df.values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)



wcss = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(2, 11), wcss, marker='o')
plt.title('Elbow Method For Optimal k')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

# Use optimal k (e.g., 3)
kmeans = KMeans(n_clusters=3, random_state=42)
labels_kmeans = kmeans.fit_predict(X_scaled)

print("Silhouette Score (KMeans):", silhouette_score(X_scaled, labels_kmeans))
```




```
linked = linkage(X_scaled, method='ward')
plt.figure(figsize=(10, 7))
dendrogram(linked)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Samples')
plt.ylabel('Distance')
plt.show()
```

```
# Fit with chosen number of clusters
hierarchical = AgglomerativeClustering(n_clusters=3)
labels_hc = hierarchical.fit_predict(X_scaled)
print("Silhouette Score (Hierarchical):", silhouette_score(X_scaled, labels_hc))
```




```
Silhouette Score (Hierarchical): 0.13628495765267165
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

print("Explained Variance Ratio:", pca.explained_variance_ratio_)

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels_kmeans, cmap='viridis')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('PCA Result with KMeans Clusters')
plt.show()
```