```python
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = '/mnt/data/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())

# Preprocessing: Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Dimensionality Reduction: Apply PCA
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

# Convert PCA results to a DataFrame for easier visualization
pca_df = pd.DataFrame(pca_data, columns=['PC1', 'PC2'])

# Clustering: Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(pca_data)

# Add cluster labels to the PCA DataFrame
pca_df['Cluster'] = clusters

# Visualization: Plot the clustered data
plt.figure(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=pca_df,
palette='viridis')
plt.title('Clusters in the PCA-reduced space')
plt.show()
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
Cell In[1], line 11
      9 # Load the dataset
     10 file_path = '/mnt/data/simulated_health_wellness_data.csv'
---> 11 data = pd.read_csv(file_path)
     13 # Display the first few rows of the dataset
```

```
     14 print("Dataset preview:")

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:1026, in read_csv(filepath_or_buffer, sep, delimiter, header,
names, index_col, usecols, dtype, engine, converters, true_values,
false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines,
parse_dates, infer_datetime_format, keep_date_col, date_parser,
date_format, dayfirst, cache_dates, iterator, chunksize, compression,
thousands, decimal, lineterminator, quotechar, quoting, doublequote,
escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision,
storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:620, in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:1620, in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
```

```
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/common.py:873,
in get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary
mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory:
'/mnt/data/simulated_health_wellness_data.csv'
```

```python
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = '/correct/path/to/your/simulated_health_wellness_data.csv'
# Update this path
try:
    data = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"File not found at path: {file_path}")
    raise

# Display the first few rows of the dataset
print("Dataset preview:")
```

```
print(data.head())

# Preprocessing: Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Dimensionality Reduction: Apply PCA
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

# Convert PCA results to a DataFrame for easier visualization
pca_df = pd.DataFrame(pca_data, columns=['PC1', 'PC2'])

# Clustering: Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(pca_data)

# Add cluster labels to the PCA DataFrame
pca_df['Cluster'] = clusters

# Visualization: Plot the clustered data
plt.figure(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=pca_df,
palette='viridis')
plt.title('Clusters in the PCA-reduced space')
plt.show()

File not found at path:
/correct/path/to/your/simulated_health_wellness_data.csv

---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
Cell In[4], line 12
     10 file_path =
'/correct/path/to/your/simulated_health_wellness_data.csv'  # Update
this path
     11 try:
---> 12     data = pd.read_csv(file_path)
     13 except FileNotFoundError:
     14     print(f"File not found at path: {file_path}")

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:1026, in read_csv(filepath_or_buffer, sep, delimiter, header,
names, index_col, usecols, dtype, engine, converters, true_values,
false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines,
parse_dates, infer_datetime_format, keep_date_col, date_parser,
```

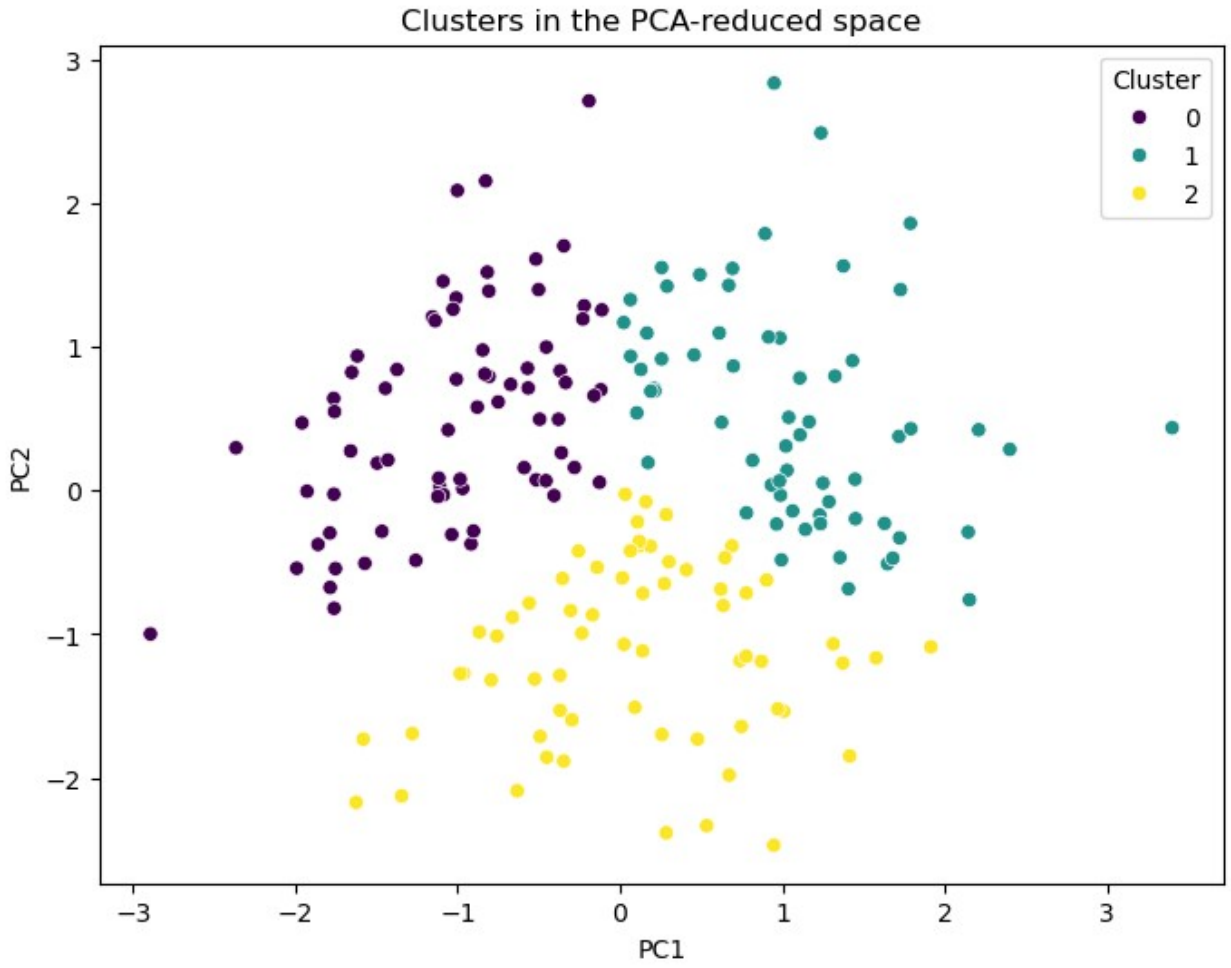
```
date_format, dayfirst, cache_dates, iterator, chunksize, compression,
thousands, decimal, lineterminator, quotechar, quoting, doublequote,
escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision,
storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:620, in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:1620, in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/parsers/readers.
py:1880, in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File
/opt/anaconda3/lib/python3.12/site-packages/pandas/io/common.py:873,
```

```
in get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary
mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory:
'/correct/path/to/your/simulated_health_wellness_data.csv'
```

```python
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path =
'/Users/kiran/Downloads/simulated_health_wellness_data.csv'
try:
    data = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"File not found at path: {file_path}")
    raise

# Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())

# Preprocessing: Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Dimensionality Reduction: Apply PCA
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)
```

```python
# Convert PCA results to a DataFrame for easier visualization
pca_df = pd.DataFrame(pca_data, columns=['PC1', 'PC2'])

# Clustering: Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(pca_data)

# Add cluster labels to the PCA DataFrame
pca_df['Cluster'] = clusters

# Visualization: Plot the clustered data
plt.figure(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=pca_df,
palette='viridis')
plt.title('Clusters in the PCA-reduced space')
plt.show()
```

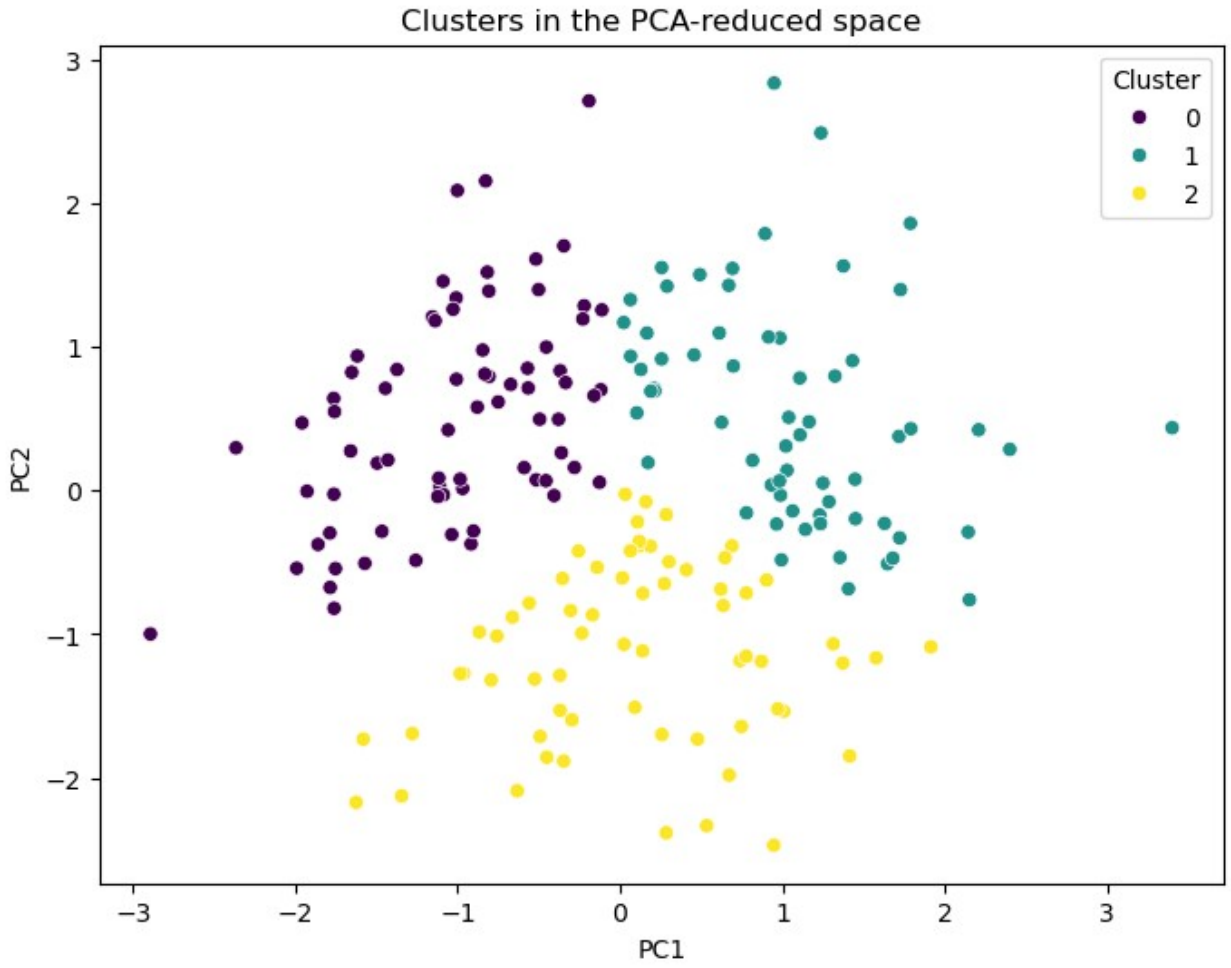
```
Dataset preview:
   Exercise_Time_Min  Healthy_Meals_Per_Day  Sleep_Hours_Per_Night  \
0          34.967142                      5               7.618856
1          28.617357                      8               4.105473
2          36.476885                      4               6.024123
3          45.230299                      1               8.565319
4          27.658466                      3               8.301648

   Stress_Level        BMI
0             2  33.068556
1             7  27.267672
2             1  23.779217
3             8  29.820436
4             3  30.947352
```




```python
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path =
'/Users/kiran/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())

# Preprocessing: Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
```

```python
# Dimensionality Reduction: Apply PCA
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

# Convert PCA results to a DataFrame for easier visualization
pca_df = pd.DataFrame(pca_data, columns=['PC1', 'PC2'])

# Clustering: Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(pca_data)

# Add cluster labels to the PCA DataFrame
pca_df['Cluster'] = clusters

# Visualization: Plot the clustered data
plt.figure(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=pca_df,
palette='viridis')
plt.title('Clusters in the PCA-reduced space')
plt.show()
```

```
Dataset preview:
   Exercise_Time_Min  Healthy_Meals_Per_Day  Sleep_Hours_Per_Night  \
0          34.967142                      5               7.618856
1          28.617357                      8               4.105473
2          36.476885                      4               6.024123
3          45.230299                      1               8.565319
4          27.658466                      3               8.301648

   Stress_Level        BMI
0             2  33.068556
1             7  27.267672
2             1  23.779217
3             8  29.820436
4             3  30.947352
```

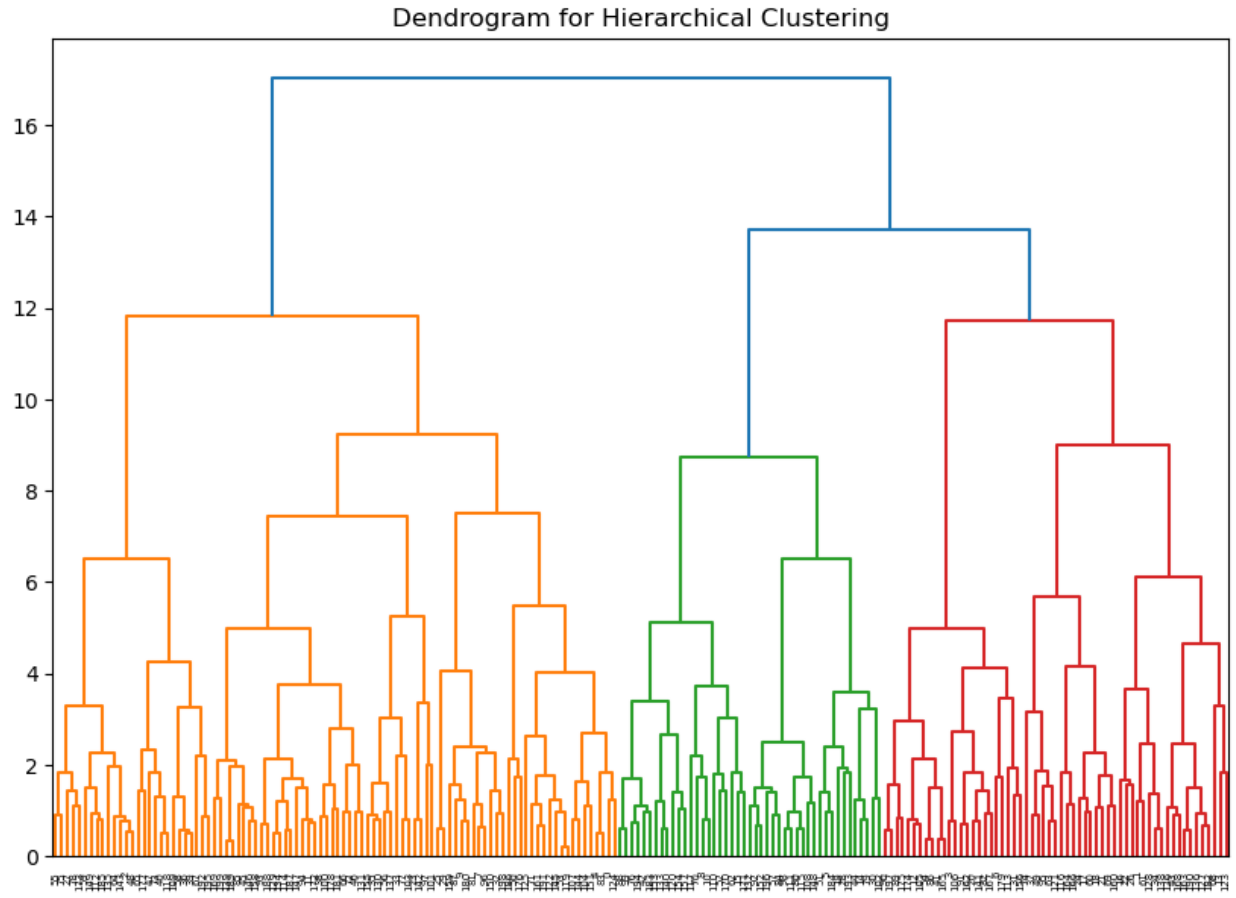



```
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path =
'/Users/kiran/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())

# Preprocessing: Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
```

```python
# K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_clusters = kmeans.fit_predict(scaled_data)
data['KMeans_Cluster'] = kmeans_clusters

# Hierarchical Clustering
# Generate the dendrogram to determine the optimal number of clusters
plt.figure(figsize=(10, 7))
dendrogram = sch.dendrogram(sch.linkage(scaled_data, method='ward'))
plt.title('Dendrogram for Hierarchical Clustering')
plt.show()

# Fit the Agglomerative Clustering model (Hierarchical Clustering)
hc = AgglomerativeClustering(n_clusters=3, affinity='euclidean',
linkage='ward')
hc_clusters = hc.fit_predict(scaled_data)
data['HC_Cluster'] = hc_clusters

# Analyze the clusters
print("\nK-Means Cluster Counts:")
print(data['KMeans_Cluster'].value_counts())

print("\nHierarchical Cluster Counts:")
print(data['HC_Cluster'].value_counts())

# Visualize K-Means Clusters
plt.figure(figsize=(8,6))
sns.scatterplot(x=data.iloc[:,0], y=data.iloc[:,1],
hue='KMeans_Cluster', palette='viridis', data=data)
plt.title('K-Means Clustering Visualization')
plt.show()

# Visualize Hierarchical Clusters
plt.figure(figsize=(8,6))
sns.scatterplot(x=data.iloc[:,0], y=data.iloc[:,1], hue='HC_Cluster',
palette='coolwarm', data=data)
plt.title('Hierarchical Clustering Visualization')
plt.show()
```

```
Dataset preview:
   Exercise_Time_Min  Healthy_Meals_Per_Day  Sleep_Hours_Per_Night  \
0          34.967142                      5               7.618856
1          28.617357                      8               4.105473
2          36.476885                      4               6.024123
3          45.230299                      1               8.565319
4          27.658466                      3               8.301648

   Stress_Level        BMI
0             2  33.068556
```

```
1              7  27.267672
2              1  23.779217
3              8  29.820436
4              3  30.947352
```




```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 34
     31 plt.show()
     33 # Fit the Agglomerative Clustering model (Hierarchical Clustering)
---> 34 hc = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
     35 hc_clusters = hc.fit_predict(scaled_data)
     36 data['HC_Cluster'] = hc_clusters

TypeError: AgglomerativeClustering.__init__() got an unexpected keyword argument 'affinity'
```

```python
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import scipy.cluster.hierarchy as sch

# Load the dataset
file_path =
'/Users/kiran/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())

# Preprocessing: Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Dimensionality Reduction: Apply PCA
pca = PCA(n_components=2)  # Reducing to 2 components
pca_data = pca.fit_transform(scaled_data)

# Convert PCA results to a DataFrame for easier visualization
pca_df = pd.DataFrame(pca_data, columns=['PC1', 'PC2'])

# Explained variance of PCA components
explained_variance = pca.explained_variance_ratio_
print(f"Explained variance by each component: {explained_variance}")

# Clustering on PCA-reduced data
# 1. K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_clusters = kmeans.fit_predict(pca_df)
pca_df['KMeans_Cluster'] = kmeans_clusters

# 2. Hierarchical Clustering
hc = AgglomerativeClustering(n_clusters=3, affinity='euclidean',
linkage='ward')
hc_clusters = hc.fit_predict(pca_df)
pca_df['HC_Cluster'] = hc_clusters

# Analyze the clusters
print("\nK-Means Cluster Counts on PCA Data:")
print(pca_df['KMeans_Cluster'].value_counts())

print("\nHierarchical Cluster Counts on PCA Data:")
```

```
print(pca_df['HC_Cluster'].value_counts())

# Visualization: PCA components with K-Means Clustering
plt.figure(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='KMeans_Cluster',
palette='viridis', data=pca_df)
plt.title('K-Means Clustering on PCA-Reduced Data')
plt.show()

# Visualization: PCA components with Hierarchical Clustering
plt.figure(figsize=(8,6))
sns.scatterplot(x='PC1', y='PC2', hue='HC_Cluster',
palette='coolwarm', data=pca_df)
plt.title('Hierarchical Clustering on PCA-Reduced Data')
plt.show()

# Interpretation: Loading the original features on PCA components
loadings = pd.DataFrame(pca.components_.T, columns=['PC1', 'PC2'],
index=data.columns)
print("\nPCA Component Loadings (How each original feature contributes
to the PCA components):")
print(loadings)

Dataset preview:
   Exercise_Time_Min  Healthy_Meals_Per_Day  Sleep_Hours_Per_Night  \
0          34.967142                      5               7.618856
1          28.617357                      8               4.105473
2          36.476885                      4               6.024123
3          45.230299                      1               8.565319
4          27.658466                      3               8.301648

   Stress_Level        BMI
0             2  33.068556
1             7  27.267672
2             1  23.779217
3             8  29.820436
4             3  30.947352
Explained variance by each component: [0.23691549 0.22082517]

---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[12], line 41
     38 pca_df['KMeans_Cluster'] = kmeans_clusters
     40 # 2. Hierarchical Clustering
---> 41 hc = AgglomerativeClustering(n_clusters=3,
affinity='euclidean', linkage='ward')
     42 hc_clusters = hc.fit_predict(pca_df)
     43 pca_df['HC_Cluster'] = hc_clusters
```

```
TypeError: AgglomerativeClustering.__init__() got an unexpected
keyword argument 'affinity'
```

```python
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# Load the dataset
file_path =
'/Users/kiran/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())

# Get basic information about the dataset
print("\nDataset Information:")
data.info()

# Descriptive statistics
print("\nDescriptive Statistics:")
print(data.describe())

# Check for missing values
print("\nMissing Values:")
print(data.isnull().sum())

# Exploratory Data Analysis
# 1. Histogram of each numerical feature
data.hist(figsize=(10, 8), bins=20, edgecolor='black')
plt.suptitle('Histograms of Numerical Features', fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# 2. Correlation heatmap to see relationships between variables
plt.figure(figsize=(10, 8))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
linewidths=0.5)
plt.title('Correlation Heatmap', fontsize=16)
plt.show()

# 3. Pairplot to analyze pairwise relationships between variables
sns.pairplot(data)
plt.suptitle('Pairplot of Variables', fontsize=16, y=1.02)
plt.show()
```

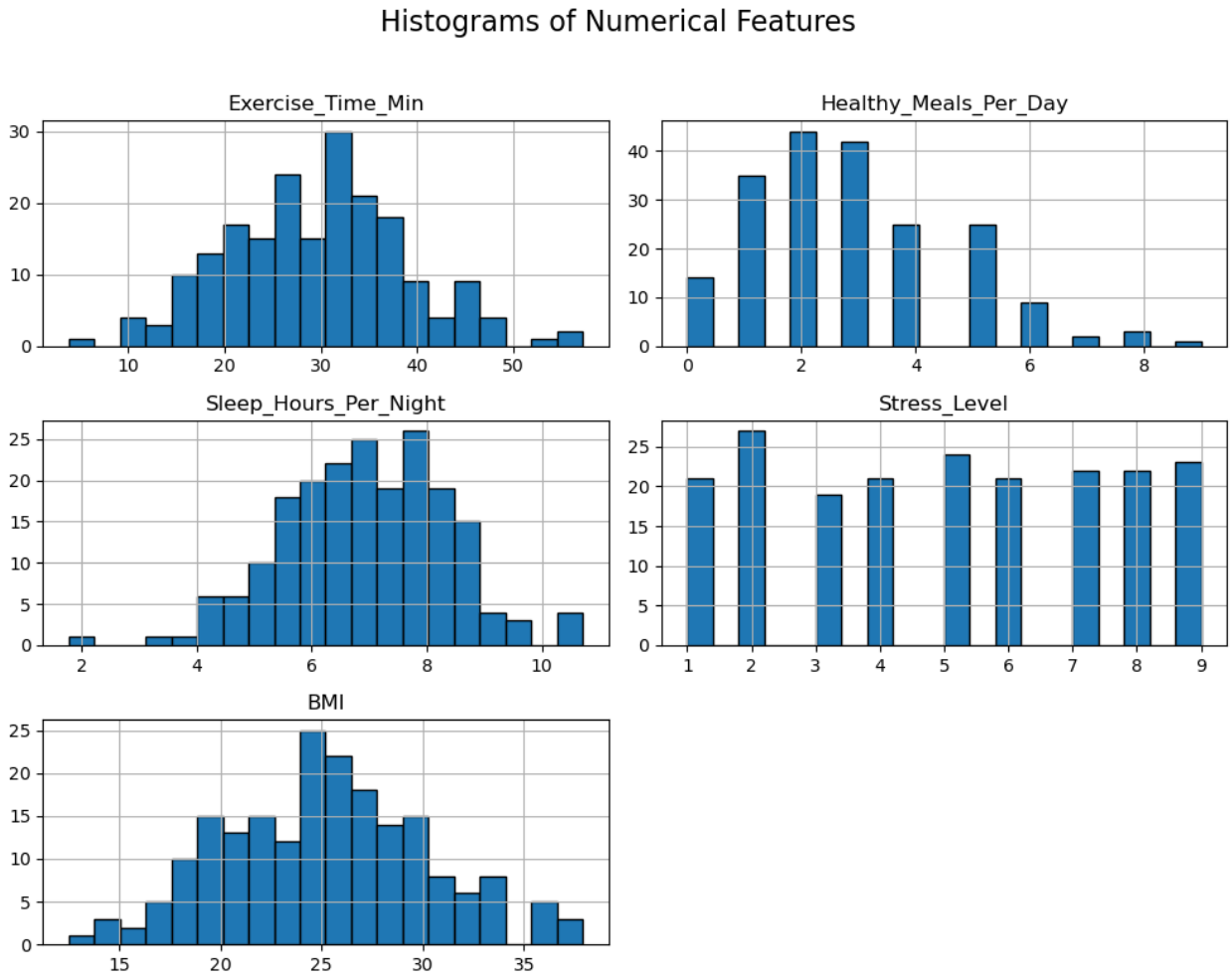
```python
# 4. Scatter plots using Plotly to visualize relationships between two variables interactively
# Example: Scatter plot of daily exercise time vs BMI
fig = px.scatter(data, x='daily_exercise_time', y='BMI',
                 title='Daily Exercise Time vs BMI',
                 labels={'daily_exercise_time': 'Daily Exercise Time (minutes)', 'BMI': 'Body Mass Index'})
fig.show()

# 5. Box plots using Seaborn to check for outliers in each feature
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, palette='Set2')
plt.title('Boxplot for Numerical Features to Detect Outliers', fontsize=16)
plt.xticks(rotation=90)
plt.show()

# 6. Plot the distribution of a specific column (e.g., Stress Level Score)
plt.figure(figsize=(8, 6))
sns.histplot(data['stress_level_score'], kde=True, bins=20)
plt.title('Distribution of Stress Level Score', fontsize=16)
plt.xlabel('Stress Level Score')
plt.ylabel('Frequency')
plt.show()

# 7. Correlation between daily exercise time and stress level score
plt.figure(figsize=(8, 6))
sns.scatterplot(x='daily_exercise_time', y='stress_level_score', data=data)
plt.title('Daily Exercise Time vs Stress Level Score', fontsize=16)
plt.xlabel('Daily Exercise Time (minutes)')
plt.ylabel('Stress Level Score')
plt.show()
```

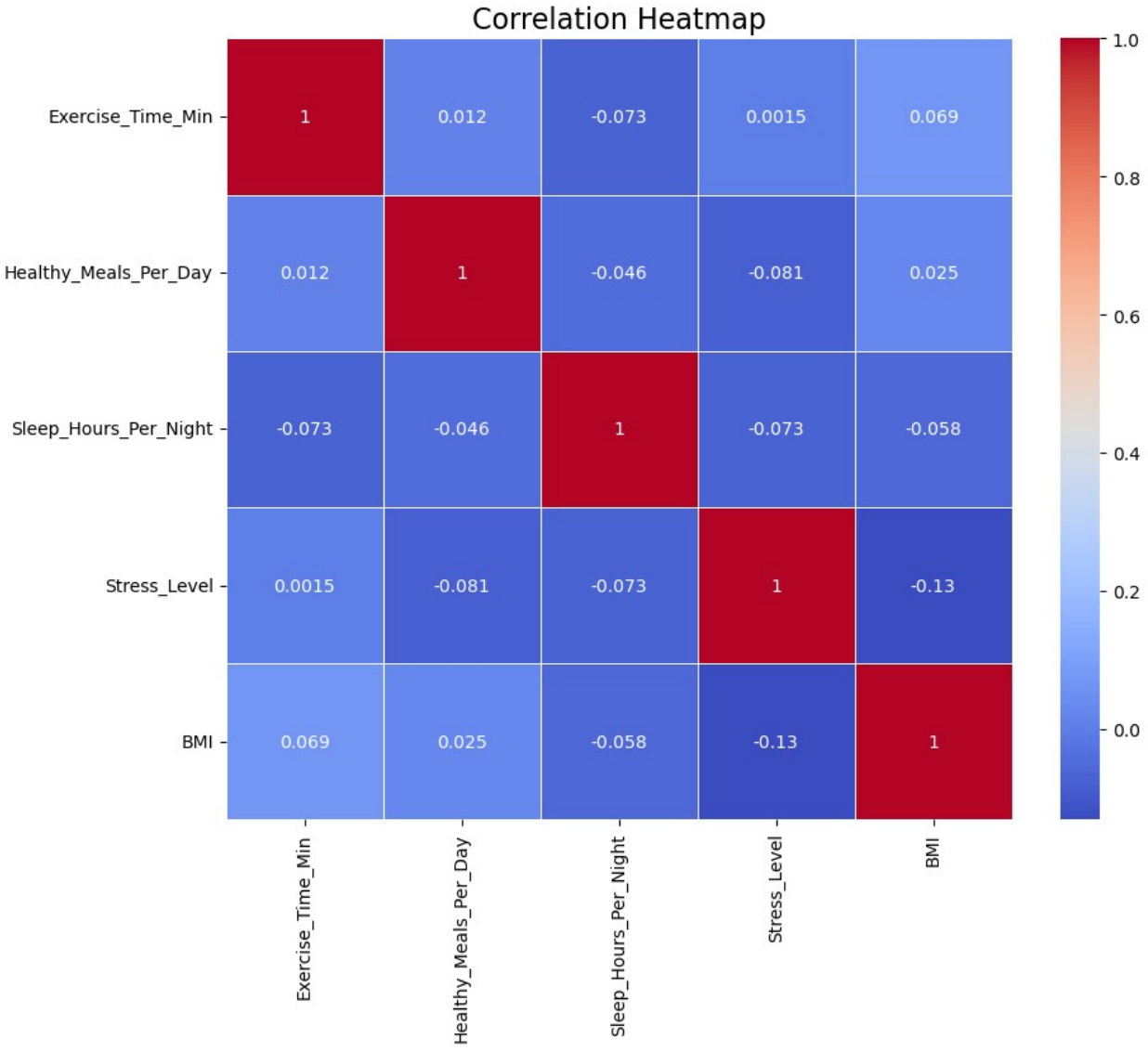
Dataset preview:

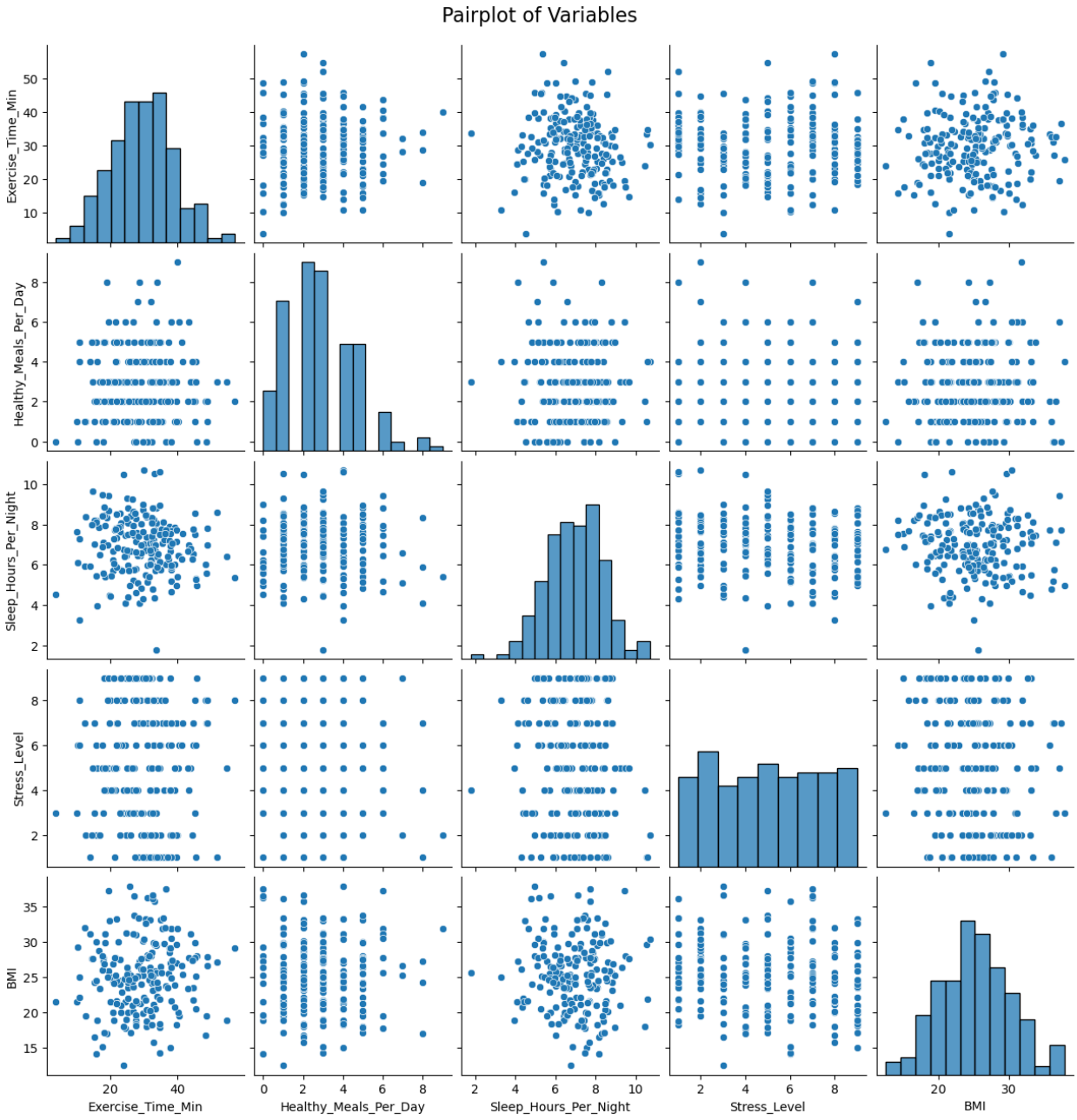
	Exercise_Time_Min	Healthy_Meals_Per_Day	Sleep_Hours_Per_Night	Stress_Level	BMI
0	34.967142	5	7.618856	2	33.068556
1	28.617357	8	4.105473	7	27.267672
2	36.476885	4	6.024123	1	23.779217
3	45.230299	1	8.565319	8	29.820436
4	27.658466	3	8.301648	3	30.947352

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Exercise_Time_Min      200 non-null    float64
 1   Healthy_Meals_Per_Day  200 non-null    int64  
 2   Sleep_Hours_Per_Night  200 non-null    float64
 3   Stress_Level           200 non-null    int64  
 4   BMI                    200 non-null    float64
dtypes: float64(3), int64(2)
memory usage: 7.9 KB

Descriptive Statistics:
       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

Missing Values:
Exercise_Time_Min        0
Healthy_Meals_Per_Day    0
Sleep_Hours_Per_Night    0
Stress_Level             0
```

```
BMI                             0
dtype: int64
```




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




```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[14], line 48
     44 plt.show()
     46 # 4. Scatter plots using Plotly to visualize relationships
between two variables interactively
     47 # Example: Scatter plot of daily exercise time vs BMI
---> 48 fig = px.scatter(data, x='daily_exercise_time', y='BMI',
     49                  title='Daily Exercise Time vs BMI',
```

```
     50                   labels={'daily_exercise_time': 'Daily
Exercise Time (minutes)', 'BMI': 'Body Mass Index'})
     51 fig.show()
     53 # 5. Box plots using Seaborn to check for outliers in each
feature

File
/opt/anaconda3/lib/python3.12/site-packages/plotly/express/_chart_type
s.py:66, in scatter(data_frame, x, y, color, symbol, size, hover_name,
hover_data, custom_data, text, facet_row, facet_col, facet_col_wrap,
facet_row_spacing, facet_col_spacing, error_x, error_x_minus, error_y,
error_y_minus, animation_frame, animation_group, category_orders,
labels, orientation, color_discrete_sequence, color_discrete_map,
color_continuous_scale, range_color, color_continuous_midpoint,
symbol_sequence, symbol_map, opacity, size_max, marginal_x,
marginal_y, trendline, trendline_options, trendline_color_override,
trendline_scope, log_x, log_y, range_x, range_y, render_mode, title,
template, width, height)
     12 def scatter(
     13     data_frame=None,
     14     x=None,
   (...)
     60     height=None,
     61 ) -> go.Figure:
     62     """
     63     In a scatter plot, each row of `data_frame` is represented
by a symbol
     64     mark in 2D space.
     65     """
---> 66     return make_figure(args=locals(), constructor=go.Scatter)

File
/opt/anaconda3/lib/python3.12/site-packages/plotly/express/_core.py:20
90, in make_figure(args, constructor, trace_patch, layout_patch)
   2087 layout_patch = layout_patch or {}
   2088 apply_default_cascade(args)
-> 2090 args = build_dataframe(args, constructor)
   2091 if constructor in [go.Treemap, go.Sunburst, go.Icicle] and
args["path"] is not None:
   2092     args = process_dataframe_hierarchy(args)

File
/opt/anaconda3/lib/python3.12/site-packages/plotly/express/_core.py:14
92, in build_dataframe(args, constructor)
   1489     args["color"] = None
   1490 # now that things have been prepped, we do the systematic
rewriting of `args`
-> 1492 df_output, wide_id_vars = process_args_into_dataframe(
   1493     args, wide_mode, var_name, value_name
   1494 )
```

```
   1496 # now that `df_output` exists and `args` contains only
references, we complete
   1497 # the special-case and wide-mode handling by further rewriting
args and/or mutating
   1498 # df_output
   1500 count_name = _escape_col_name(df_output, "count", [var_name,
value_name])

File
/opt/anaconda3/lib/python3.12/site-packages/plotly/express/_core.py:12
13, in process_args_into_dataframe(args, wide_mode, var_name,
value_name)
   1211         if argument == "index":
   1212             err_msg += "\n To use the index, pass it in
directly as `df.index`."
-> 1213         raise ValueError(err_msg)
   1214 elif length and len(df_input[argument]) != length:
   1215     raise ValueError(
   1216         "All arguments should have the same length. "
   1217         "The length of column argument `df[%s]` is %d, whereas
the "
   (...)
   1224         )
   1225     )

ValueError: Value of 'x' is not the name of a column in 'data_frame'.
Expected one of ['Exercise_Time_Min', 'Healthy_Meals_Per_Day',
'Sleep_Hours_Per_Night', 'Stress_Level', 'BMI'] but received:
daily_exercise_time
```

```python
# Import necessary libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import numpy as np

# Load the dataset
file_path =
'/Users/kiran/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())

# Preprocessing: Standardize the data
scaler = StandardScaler()
```

```
scaled_data = scaler.fit_transform(data)

# Function to calculate WCSS for K-Means
def calculate_wcss(data):
    wcss = []
    for n in range(1, 11):  # Trying different numbers of clusters
        kmeans = KMeans(n_clusters=n, random_state=42)
        kmeans.fit(data)
        wcss.append(kmeans.inertia_)  # Inertia is the WCSS
    return wcss

# K-Means Clustering before PCA
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_clusters = kmeans.fit_predict(scaled_data)

# Silhouette Score before PCA
silhouette_kmeans_before = silhouette_score(scaled_data,
kmeans_clusters)
print(f"Silhouette Score (K-Means) before PCA:
{silhouette_kmeans_before}")

# Calculate WCSS before PCA
wcss_before_pca = calculate_wcss(scaled_data)
print(f"Within-Cluster Sum of Squares (WCSS) before PCA:
{wcss_before_pca}")

# Hierarchical Clustering before PCA
hc = AgglomerativeClustering(n_clusters=3, affinity='euclidean',
linkage='ward')
hc_clusters = hc.fit_predict(scaled_data)

# Silhouette Score before PCA (Hierarchical Clustering)
silhouette_hc_before = silhouette_score(scaled_data, hc_clusters)
print(f"Silhouette Score (Hierarchical Clustering) before PCA:
{silhouette_hc_before}")

# Apply PCA to reduce dimensions
pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

# K-Means Clustering after PCA
kmeans_pca = KMeans(n_clusters=3, random_state=42)
kmeans_pca_clusters = kmeans_pca.fit_predict(pca_data)

# Silhouette Score after PCA (K-Means)
silhouette_kmeans_after = silhouette_score(pca_data,
kmeans_pca_clusters)
print(f"Silhouette Score (K-Means) after PCA:
{silhouette_kmeans_after}")
```

```
# Calculate WCSS after PCA
wcss_after_pca = calculate_wcss(pca_data)
print(f"Within-Cluster Sum of Squares (WCSS) after PCA:
{wcss_after_pca}")

# Hierarchical Clustering after PCA
hc_pca = AgglomerativeClustering(n_clusters=3, affinity='euclidean',
linkage='ward')
hc_pca_clusters = hc_pca.fit_predict(pca_data)

# Silhouette Score after PCA (Hierarchical Clustering)
silhouette_hc_after = silhouette_score(pca_data, hc_pca_clusters)
print(f"Silhouette Score (Hierarchical Clustering) after PCA:
{silhouette_hc_after}")

# Plot WCSS before and after PCA
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss_before_pca, label='Before PCA',
marker='o')
plt.plot(range(1, 11), wcss_after_pca, label='After PCA', marker='o')
plt.title('WCSS Before and After PCA')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.legend()
plt.show()

# Summary of Results
print("\nComparison Summary:")
print(f"Silhouette Score (K-Means) before PCA:
{silhouette_kmeans_before}")
print(f"Silhouette Score (K-Means) after PCA:
{silhouette_kmeans_after}")
print(f"Silhouette Score (Hierarchical Clustering) before PCA:
{silhouette_hc_before}")
print(f"Silhouette Score (Hierarchical Clustering) after PCA:
{silhouette_hc_after}")
```

```
Dataset preview:
   Exercise_Time_Min  Healthy_Meals_Per_Day  Sleep_Hours_Per_Night  \
0          34.967142                      5               7.618856
1          28.617357                      8               4.105473
2          36.476885                      4               6.024123
3          45.230299                      1               8.565319
4          27.658466                      3               8.301648

   Stress_Level        BMI
0             2  33.068556
1             7  27.267672
2             1  23.779217
3             8  29.820436
```

```
4             3  30.947352
Silhouette Score (K-Means) before PCA: 0.1516159911787657
Within-Cluster Sum of Squares (WCSS) before PCA: [999.9999999999998,
827.8082622275492, 740.46626764846, 660.2944852897326,
603.6685808745752, 524.0273006765655, 494.47180579109255,
485.68969307748915, 457.4529430729492, 440.13109309585354]

---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[16], line 44
     41 print(f"Within-Cluster Sum of Squares (WCSS) before PCA:
{wcss_before_pca}")
     43 # Hierarchical Clustering before PCA
---> 44 hc = AgglomerativeClustering(n_clusters=3,
affinity='euclidean', linkage='ward')
     45 hc_clusters = hc.fit_predict(scaled_data)
     47 # Silhouette Score before PCA (Hierarchical Clustering)

TypeError: AgglomerativeClustering.__init__() got an unexpected
keyword argument 'affinity'
```