George Washington University

Indicators of Heart Health

Avani Rao

GWID: G45297503

DATS6401: Visualization of Complex Data

Prof. Reza Jafari

03/11/2024

Dash Link: https://dashapp-txdi7nwftq-ue.a.run.app/

Table of Contents

Introduction

The dataset originally comes from the CDC (Centers of Disease control and Prevention) and is a major part of the Behavioral Risk Factor Surveillance System (BRFSS),which conducts annual telephone surveys to collect data on the health status of U.S. residents.

About Dataset

The dataset under discussion is derived from a key initiative by the Centers for Disease Control and Prevention (CDC) focused on understanding health trends in the United States. Heart disease is a prominent health issue affecting a diverse spectrum of the population and remains a leading cause of mortality. Nearly half the U.S. population is impacted by at least one of the primary risk factors—hypertension, elevated cholesterol, or smoking. Additionally, factors such as diabetes, obesity, sedentary lifestyle, and excessive alcohol consumption significantly contribute to heart disease risk. The imperative to pinpoint and mitigate the most influential risk factors is a critical component of public health strategies.

The data originates from the CDC's Behavioral Risk Factor Surveillance System (BRFSS), a comprehensive health survey that began in 1984 with a modest cohort of 15 states. Today, it has expanded to encompass all 50 states, the District of Columbia, and three U.S. territories. With over 400,000 adult interviews annually, the BRFSS stands as the most extensive health survey executed on an ongoing basis globally. It gathers invaluable insights via telephone surveys, aimed at monitoring health conditions and risk behaviors among adults in the U.S. This wealth of data serves as a cornerstone for applying advanced machine learning techniques to unveil patterns that may forecast an individual's likelihood of developing heart disease.

Objective

The analysis centers on identifying and understanding the factors that exert the most significant influence on heart disease prevalence.

Variables

There are 40 variables (columns) in this dataset.

- State: The U.S. state where the individual resides.
- Sex: Gender of the individual (Male or Female)
- GeneralHealth : Self-reported general health status of the individual.
- PhysicalHealthDays : Number of days in the past 30 days that physical health was not good.
- MentalHealthDays : Number of days in the past 30 days that mental health was not good.
- PhysicalActivities : Frequency of engaging in physical activities or exercises.
- SleepHours : Average number of hours of sleep per night.
- AgeCategory : Categorized age group of the individual
- HadHeartAttack : Whether the individual has had a heart attack.
- SmokerStatus : Current smoking status of the individual (smoker, former smoker, non-smoker).
- ECigaretteUsage : Whether the individual uses e-cigarettes.
- RaceEthnicityCategory : Categorized race or ethnicity of the individual.
- HeightInMeters : Height of the individual in meters.
- WeightInKilograms : Weight of the individual in kilograms.
- BMI : Body Mass Index calculated from height and weight.
- AlcoholDrinkers : Whether the individual consumes alcohol.

Before Cleaning

The dataset before cleaning looks like this

```
     State     Sex  ... HighRiskLastYear  CovidPos
0  Alabama  Female  ...               No        No
1  Alabama  Female  ...               No        No
2  Alabama  Female  ...               No       Yes
3  Alabama  Female  ...               No        No
4  Alabama  Female  ...               No        No

[5 rows x 40 columns]
                 State     Sex  ... HighRiskLastYear  CovidPos
445127  Virgin Islands  Female  ...               No       Yes
445128  Virgin Islands  Female  ...               No        No
445129  Virgin Islands  Female  ...               No        No
445130  Virgin Islands    Male  ...               No       Yes
445131  Virgin Islands    Male  ...               No        No

[5 rows x 40 columns]
```

Null Values in the Dataset	
Column	Missing Values
State	0
Sex	0
GeneralHealth	1198
PhysicalHealthDays	10927
MentalHealthDays	9067
LastCheckupTime	8308
PhysicalActivities	1093
SleepHours	5453
RemovedTeeth	11360
HadHeartAttack	3065
HadAngina	4405
HadStroke	1557
HadAsthma	1773
HadSkinCancer	3143
HadCOPD	2219
HadDepressiveDisorder	2812
HadKidneyDisease	1926
HadArthritis	2633
HadDiabetes	1087
DeafOrHardOfHearing	20647
BlindOrVisionDifficulty	21564

Contd…

DifficultyConcentrating	24240
DifficultyWalking	24012
DifficultyDressingBathing	23915
DifficultyErrands	25656
SmokerStatus	35462
ECigaretteUsage	35660
ChestScan	56046
RaceEthnicityCategory	14057
AgeCategory	9079
HeightInMeters	28652
WeightInKilograms	42078
BMI	48806
AlcoholDrinkers	46574
HIVTesting	66127
FluVaxLast12	47121
PneumoVaxEver	77040
TetanusLast10Tdap	82516
HighRiskLastYear	50623
CovidPos	50764

Table 1 - 1 Table of missing Nan Values

```
In [90]: print(df.describe().to_string())
       PhysicalHealthDays  MentalHealthDays     SleepHours  HeightInMeters  WeightInKilograms            BMI
count       246013.000000     246013.000000  246013.000000   246013.000000      246013.000000  246013.000000
mean             4.119055          4.167292       7.021312        1.705150          83.615522      28.668258
std              8.405803          8.102796       1.440698        0.106654          21.323232       6.514005
min              0.000000          0.000000       1.000000        0.910000          28.120000      12.020000
25%              0.000000          0.000000       6.000000        1.630000          68.040000      24.270000
50%              0.000000          0.000000       7.000000        1.700000          81.650000      27.460000
75%              3.000000          4.000000       8.000000        1.780000          95.250000      31.890000
max             30.000000         30.000000      24.000000        2.410000         292.570000      97.650000
```

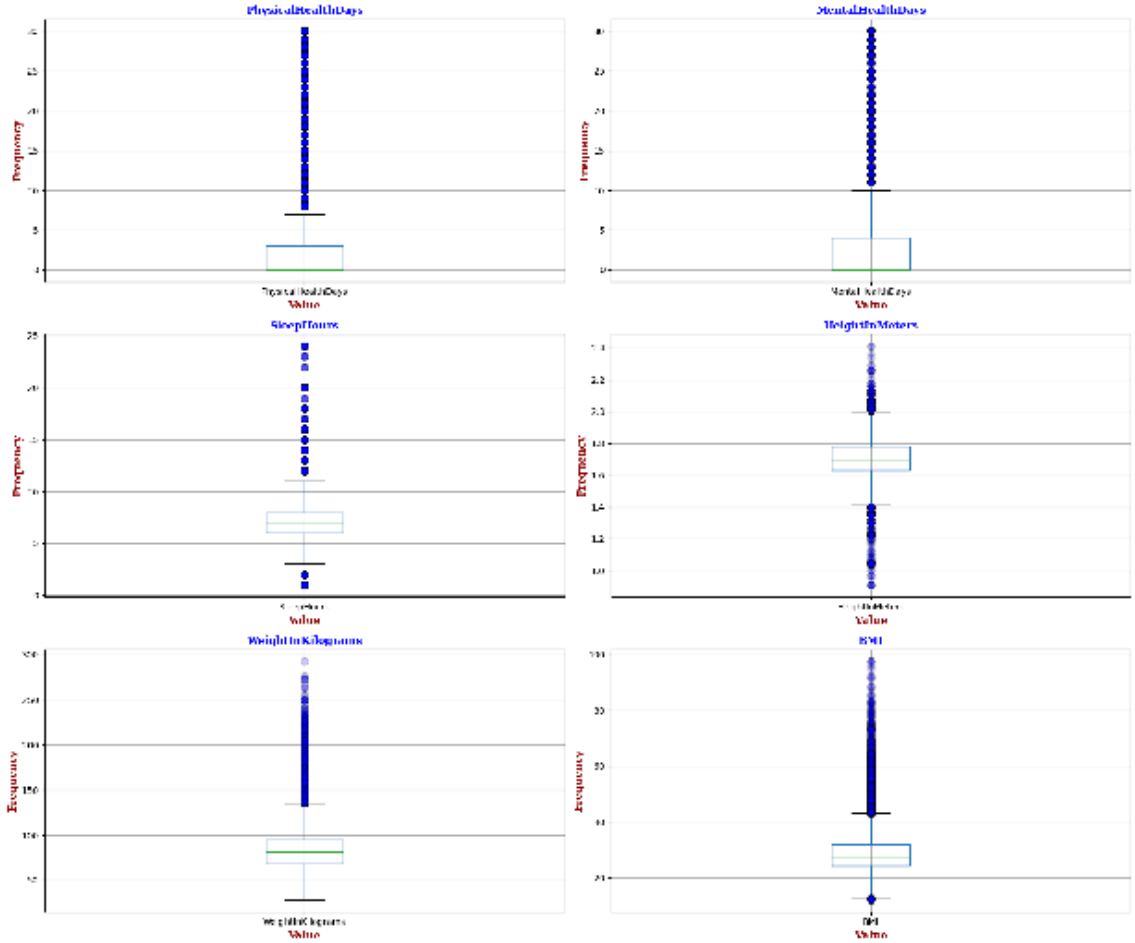
Table 2 - Descriptive statistics of the various variables

This table provides descriptive statistics for various health-related continuous variables after removing

duplicates and null values. There are 24,613 entries for each variable. On average, people report about 4 days of poor physical and mental health each month. Sleep averages around 7 hours per night. Height and weight data suggest an average BMI of approximately 28.7, indicating that the average individual is in the overweight category according to BMI classifications.

OUTLIERS Before removal

The analysis centers on identifying and understanding the factors that exert the most significant influence on heart disease prevalence.



PhysicalHealthDays: Most values cluster at the low end (near zero), indicating that many respondents report few days with poor physical health. The points above the box show a spread of outliers, which suggest that some respondents experience a high number of poor physical health days.

MentalHealthDays: This plot is similar to that for PhysicalHealthDays, with many values at zero and a similar pattern of outliers, indicating occasional high numbers of poor mental health days among respondents.

SleepHours: Sleep hours are more normally distributed, with a median around 7 hours. Outliers are present on both low and high ends, showing some respondents get very few or many more hours of sleep than average.

HeightInMeters: Height is fairly normally distributed around a median, with some outliers that are exceptionally short or tall.

WeightInKilograms: The distribution of weight has a right-skewed pattern, with a concentration of data points at lower weights and outliers suggesting a long tail of respondents with higher weights.

BMI: The distribution of BMI is also right-skewed, similar to weight, indicating that while most individuals have a BMI in the lower range, there are significant outliers on the higher end.

Lower and upper boound of each numeric column			
Column Name	Outliers Count	Lower Bound	Upper Bound
PhysicalHealthDays	38809	-4.5	7.5
MentalHealthDays	32714	-6.0	10.0
SleepHours	3488	3.0	11.0
HeightInMeters	830	1.4	2.01
WeightInKilograms	5940	27.23	136.06
BMI	7563	12.84	43.32

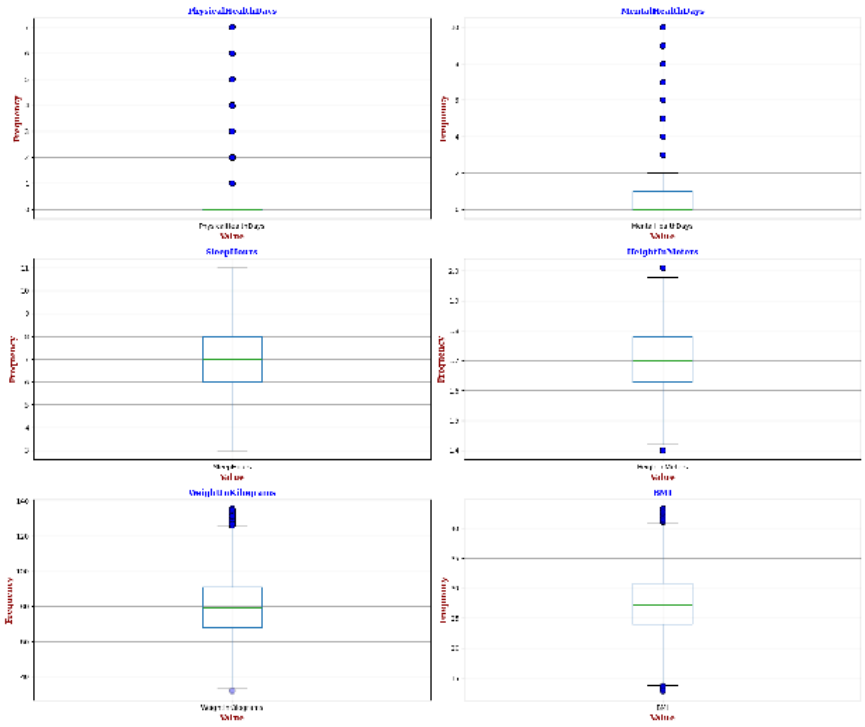
Table 1 - 2 Description of lower and upper bound of each numeric column

Data after dropping the outliers

	State	Sex	...	HighRiskLastYear	CovidPos
0	Alabama	Female	...	No	No
1	Alabama	Male	...	No	No
2	Alabama	Male	...	No	Yes
3	Alabama	Female	...	No	Yes
4	Alabama	Male	...	No	No
...	...	...	...	...	...
179788	Virgin Islands	Male	...	No	No
179789	Virgin Islands	Male	...	No	No
179790	Virgin Islands	Female	...	No	Yes
179791	Virgin Islands	Female	...	No	No
179792	Virgin Islands	Male	...	No	Yes

[179793 rows x 40 columns]

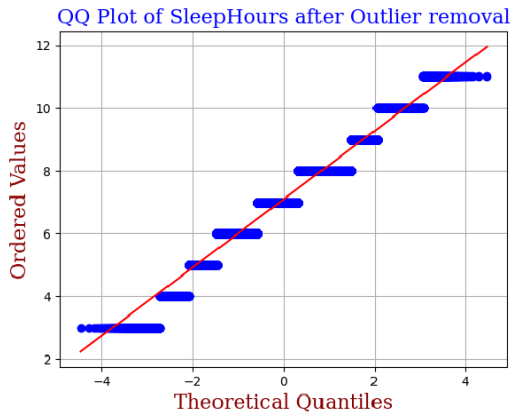
Table 1 - 4 Post cleaning outlier check



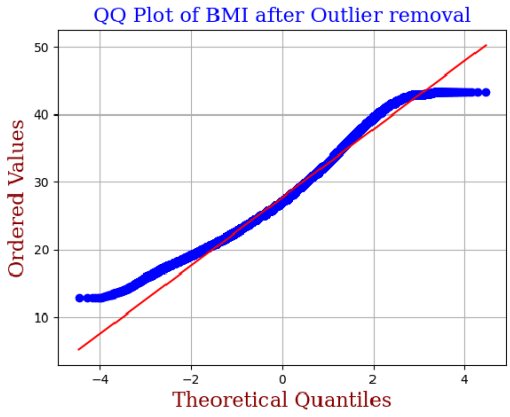
Check for normality after outlier detection

```
Column: PhysicalHealthDays
Statistics=79299.59, p-value=0.00
The data is not normally distributed (reject H0)

Column: MentalHealthDays
Statistics=80767.44, p-value=0.00
The data is not normally distributed (reject H0)

Column: SleepHours
Statistics=3387.70, p-value=0.00
The data is not normally distributed (reject H0)

Column: HeightInMeters
Statistics=5255.79, p-value=0.00
The data is not normally distributed (reject H0)

Column: WeightInKilograms
Statistics=4952.52, p-value=0.00
The data is not normally distributed (reject H0)

Column: BMI
Statistics=6970.07, p-value=0.00
```

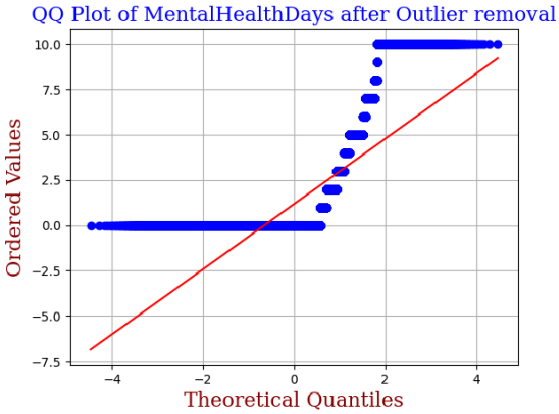
We can see the dataset is not normal



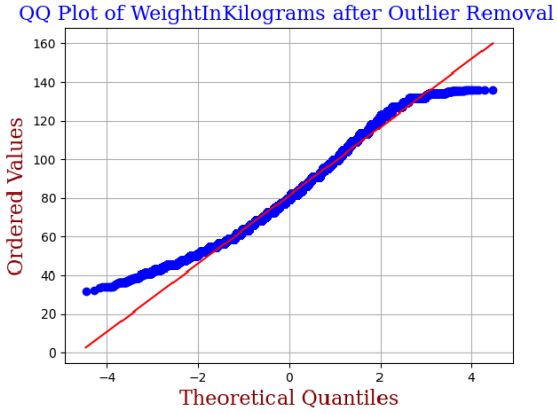
The points largely follow the reference line, indicating that sleep hours approximate a normal distribution after outlier removal, with slight deviations in the tails



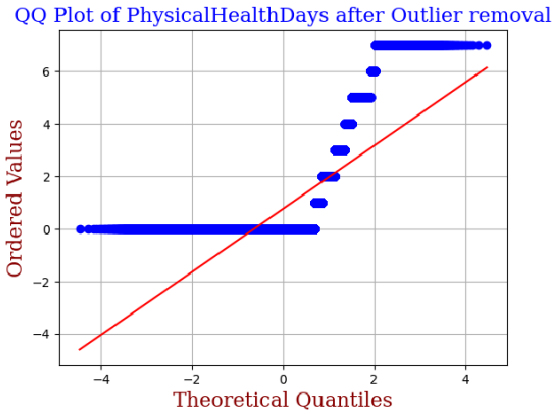
The curve in the plot suggests that BMI has a right-skewed distribution, even after removing outliers, with a heavier tail on the right side.



The sharp upward turn at the high end indicates that the distribution of Mental Health Days is highly right-skewed, which persists after outlier removal.



Like BMI, weight has a right-skewed distribution with a long right tail, which is common in biological data.

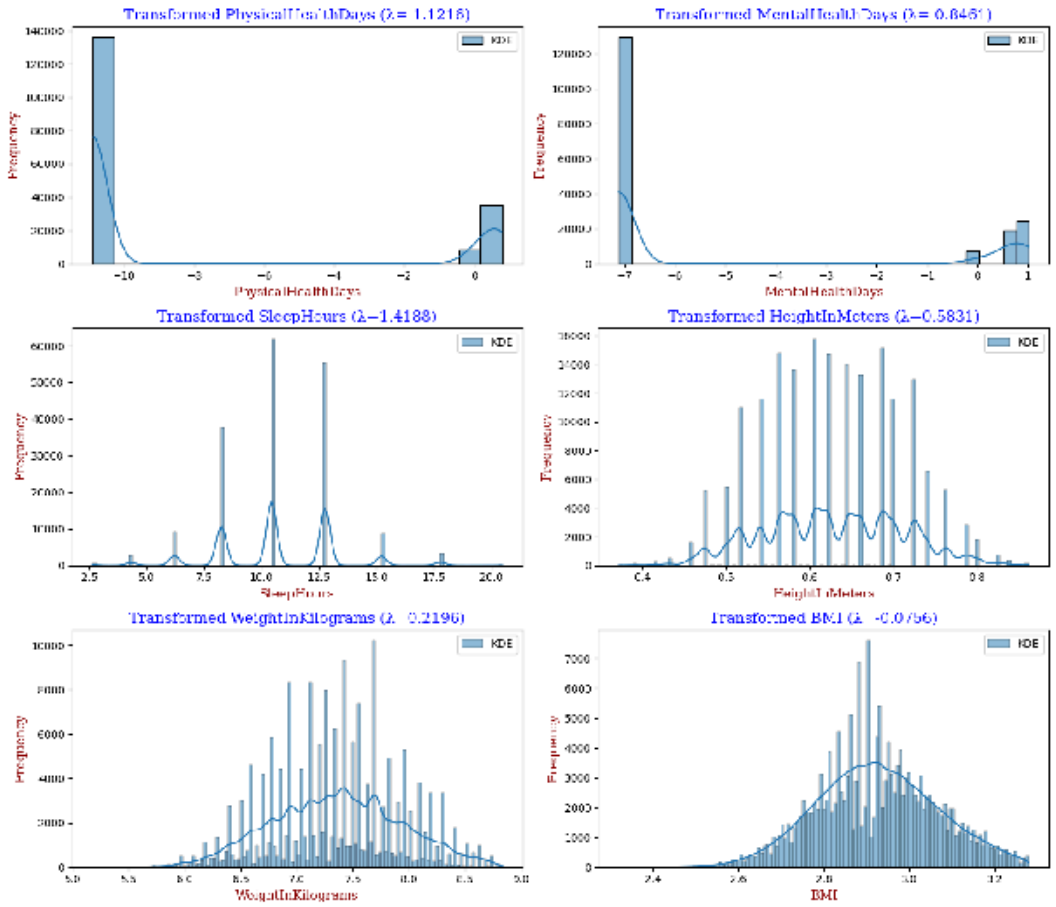


The plot indicates a right-skewed distribution with many values clustered at the lower end and some deviation from normality in the higher quantiles.

Inference from above plots - The QQ plots demonstrate that while some variables (like height and sleep hours) closely follow a normal distribution, others (like BMI, weight, and health days) show skewed distributions, even after the removal of outliers. These patterns are expected as real-world biological and health-related data often deviate from perfect normal.

Since data is not normal lets try applying box cox transformation to convert or not normal dataset to normal.

Subplots of the numeric columns after transforming the data using Box Cox Transformation.



The Box-Cox transformation is a statistical technique used to stabilize variance and make data more closely resemble a normal distribution. The transformation is defined as $Y(\lambda)=(Y\lambda-1)/\lambda$Y(λ)=(Yλ−1)/λ for $\lambda\neq0$λ≠0λ=0, and as log(*Y*)log(Y) for $\lambda=0$λ=0λ=0, where *Y*Y is the response variable and λλ is the transformation parameter.

The value of λλ is chosen to maximize the likelihood of the data fitting a normal distribution. A value of $\lambda=1$λ=1λ=1 indicates no transformation, $\lambda=0$λ=0λ=0 applies a logarithmic transformation, and other values indicate various powers of transformation.

The histograms and Kernel Density Estimate (KDE) plots represent the distribution of the variables after the Box-Cox transformation, with λλ values indicated for each. For instance:

- **PhysicalHealthDays** and **MentalHealthDays** with negative $\lambda\lambda$ values are heavily skewed to the right, with high kurtosis, indicating many values near zero and a long tail towards larger values.
- **SleepHours**, **HeightInMeters**, **WeightInKilograms**, and BMI show transformations aiming to normalize the data, reflected by the more symmetrical shape of their KDE curves. However, the multiple peaks in the transformed HeightInMeters distribution suggest discrete categories within the data that the transformation couldn't smooth into a single normal curve.

This is after applying the cox box the d'Agostino normality test , The D'Agostino's K-squared test is a statistical test that checks for normality in a dataset. The test combines skew and kurtosis to produce a test of normality, with the hypothesis that the data follows a normal distribution.

```
==================================================
da_k_squared test: PhysicalHealthDays dataset: statistics= 32496.40, p-value = 0.00
da_k_squared test: PhysicalHealthDays dataset is Not Normal
==================================================
==================================================
da_k_squared test: MentalHealthDays dataset: statistics= 67800.65, p-value = 0.00
da_k_squared test: MentalHealthDays dataset is Not Normal
==================================================
==================================================
da_k_squared test: SleepHours dataset: statistics= 1400.48, p-value = 0.00
da_k_squared test: SleepHours dataset is Not Normal
==================================================
==================================================
da_k_squared test: HeightInMeters dataset: statistics= 5342.17, p-value = 0.00
da_k_squared test: HeightInMeters dataset is Not Normal
==================================================
==================================================
da_k_squared test: WeightInKilograms dataset: statistics= 1734.41, p-value = 0.00
da_k_squared test: WeightInKilograms dataset is Not Normal
==================================================
==================================================
da_k_squared test: BMI dataset: statistics= 345.15, p-value = 0.00
da_k_squared test: BMI dataset is Not Normal
==================================================
```

The results indicate very large statistics and p-values of 0.00 for each variable tested, which strongly suggests the rejection of the null hypothesis that the dataset is normally distributed. In other words, for each of the variables PhysicalHealthDays, MentalHealthDays, SleepHours, HeightInMeters, WeightInKilograms, and BMI, the test concludes that the data do not follow a normal distribution.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction, simplifying the complexity of high-dimensional data while retaining most of the information. It identifies the directions (principal components) along which the variation in the data is maximum, transforming the original data into a new set of orthogonal features. PCA is often used to visualize high-dimensional data, reduce noise and redundancy, and prepare data for other machine learning tasks. The transformed features are linear combinations of the original variables, where the first few principal components typically capture most of the variance in the dataset.

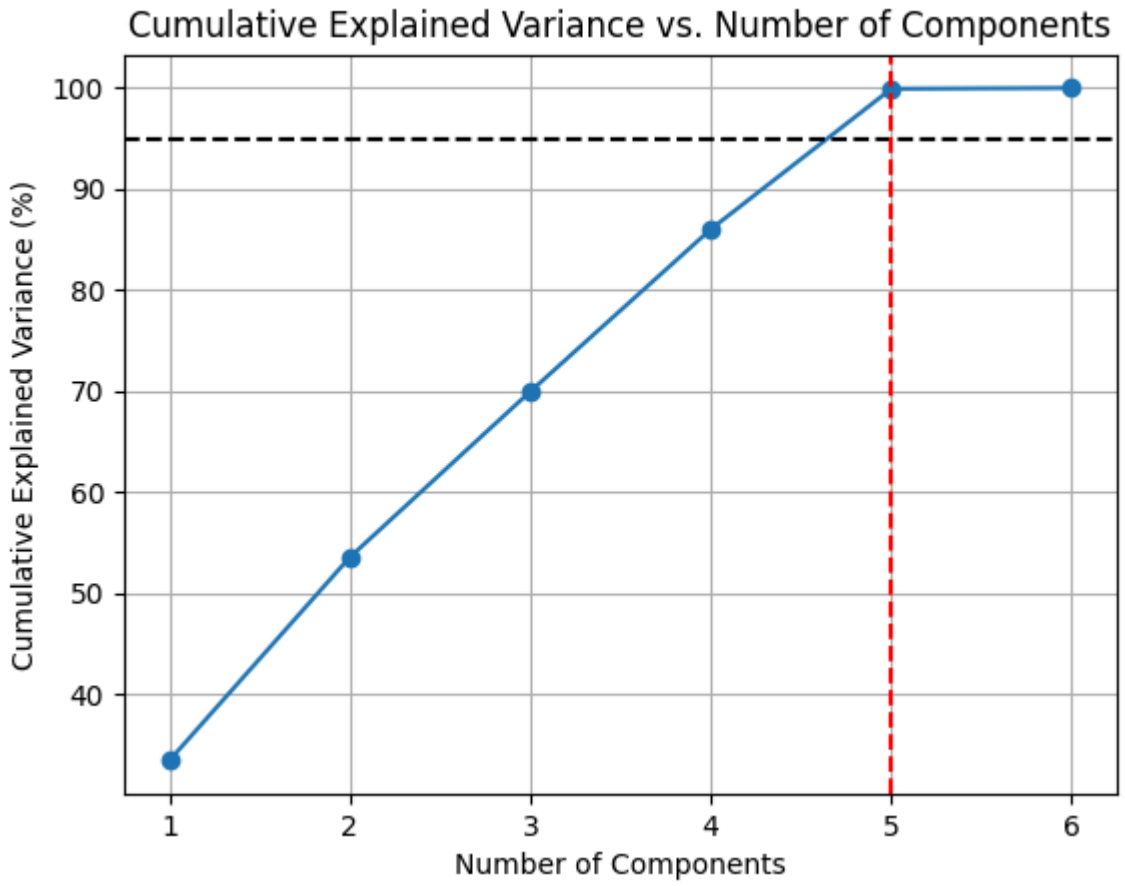
```
Explained Variance Ratio:
PC1: 33.41%
PC2: 20.13%
PC3: 16.41%
PC4: 16.07%
PC5: 13.90%
PC6: 0.09%

Cumulative Explained Variance:
PC1: 33.41%
PC2: 53.54%
PC3: 69.95%
PC4: 86.01%
PC5: 99.91%
PC6: 100.00%
```

Explained Variance Ratio: Each PC's explained variance ratio tells us how much information (variance) can be attributed to each principal component. For example, PC1 accounts for 33.41% of the variance in the data, and PC2 accounts for an additional 20.13%, and so forth.

Cumulative Explained Variance: This metric indicates the total variance captured by the first *n*n PCs. By PC5, 99.91% of the total variance in the data is captured, which suggests that most information can be represented by the first five PCs.

```
Number of features to be considered per the PCA analysis and assumed threshold (95% explained variance): 5
Explained Variance Ratio (Original Feature Space): [0.33408925 0.20130271 0.16406994 0.16065818 0.13902428 0.00085565]
Explained Variance Ratio (Reduced Feature Space): [0.33408925 0.20130271 0.16406994 0.16065818 0.13902428 0.00085565]
```

Optimum Number of Features: The second output indicates that to maintain 95% of the total variance, five features (principal components) should be retained.



Cumulative Explained Variance vs. Number of Components Graph: The graph visualizes how the cumulative explained variance increases with the number of principal components used. The red dashed line indicates the point where 95% of the total variance is accounted for, which corresponds to the point where five components are used.

```
Singular values for the reduced feature space: [600.33445128 466.00097039 420.70388275 416.30673238 387.26418639]
Condition number for the reduced feature space: 1.5501935690898938
Singular values for the original feature space: [600.33445128 466.00097039 420.70388275 416.30673238 387.26418639
  30.38156945]
Condition number for the original feature space: 19.759823543412423
```

Singular Values for the Reduced Feature Space: The singular values [600.334, 466.009, 420.703, 416.306, 387.264] for the reduced feature space indicate the importance of each principal component in the reduced space. The first singular value (600.334) is the largest, meaning the first principal component holds the most variance, and each subsequent principal component contributes less to the total variance.

Condition Number for the Reduced Feature Space: The condition number (1.5501) for the reduced feature space is relatively low, which suggests that the reduced feature space is well-conditioned and numerical calculations will be stable. This is desirable in data processing and means that the PCA has done a good job in decorrelating the features.

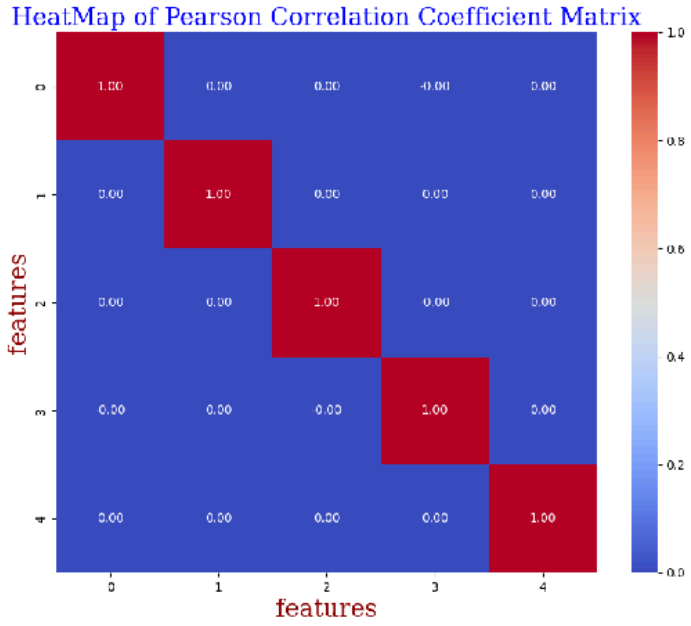
Singular Values for the Original Feature Space: Similar to the reduced space, the original singular values [600.334, 466.009, 420.703, 416.306, 387.264, 30.3815] tell us about the distribution of variance in the original data before dimensionality reduction. However, the presence of a very small singular value (30.3815) in comparison to the others suggests that there's a feature that does not contribute much variance, which is likely the one PCA has identified as being reducible.

Condition Number for the Original Feature Space: The condition number (19.7598) for the original feature space is significantly higher than in the reduced space, indicating that the original dataset was less stable for numerical operations. The higher condition number suggests that the original dataset may

have had features that were highly correlated or had redundant information, which PCA has successfully addressed by transforming the data into a space where these issues are mitigated.

Condition Number: The condition number provides an indication of the stability of a matrix (in this case, the matrix of features) to inversion or solving linear equations. A high condition number indicates potential numerical instability, while a lower condition number indicates a more stable system. The condition number is significantly lower in the reduced feature space compared to the original, which indicates an improvement in the numerical stability of the dataset after PCA transformation.

Overall, the PCA has effectively reduced the dimensionality of the data while retaining most of the variance, and also improved the condition number, suggesting an enhanced dataset for further analysis.

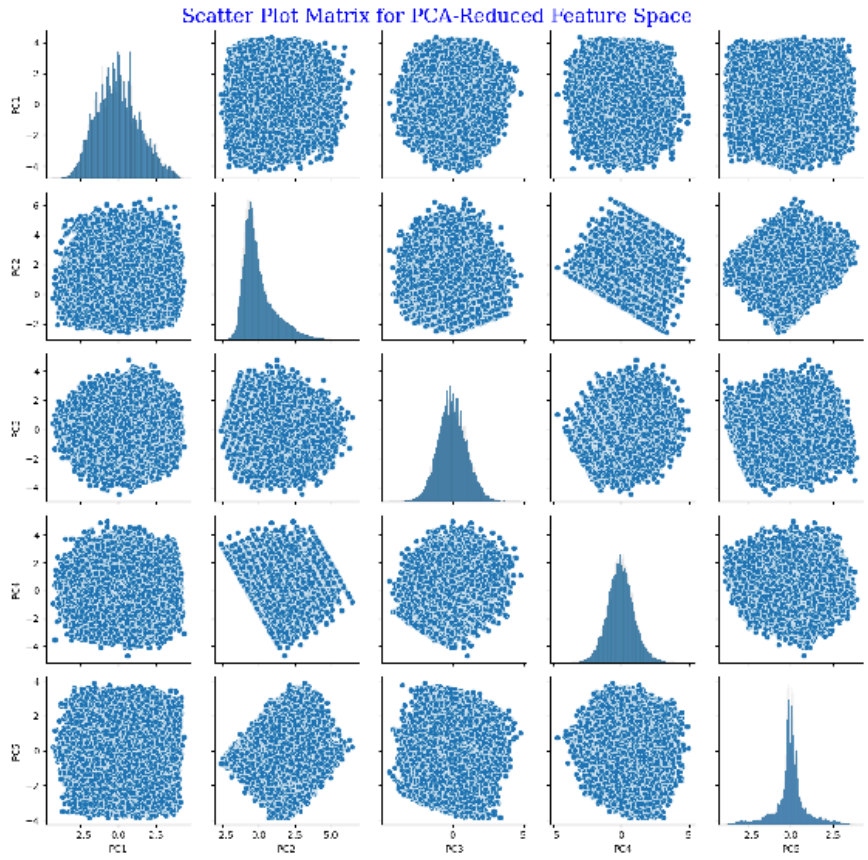


Heatmap of the Pearson correlation coefficient matrix for the PCA-reduced feature space. The color scale represents the strength of the correlation, with 1 indicating perfect positive correlation, -1 indicating perfect negative correlation, and 0 indicating no correlation. The diagonal, which shows the correlation of each PC with itself, is 1 as expected. Off-diagonal elements are all very close to 0,

indicating no correlation between different PCs. This is what you expect after PCA because it orthogonalized the feature space.

Each principal component is uncorrelated with the others (as shown by the near-zero off-diagonal values in the heatmap).

Comparison	Correlation Coefficient	Observations
PC1 vs PC1	1.0	Perfect positive correlation, as expected.
PC1 vs PC2	0.0	No correlation, indicating orthogonality.
PC1 vs PC3	0.0	No correlation, indicating orthogonality.
PC1 vs PC4	0.0	No correlation, indicating orthogonality.
PC1 vs PC5	0.0	No correlation, indicating orthogonality.
PC2 vs PC2	1.0	Perfect positive correlation, as expected.
PC2 vs PC3	0.0	No correlation, indicating orthogonality.
PC2 vs PC4	0.0	No correlation, indicating orthogonality.
PC2 vs PC5	0.0	No correlation, indicating orthogonality.
PC3 vs PC3	1.0	Perfect positive correlation, as expected.
PC3 vs PC4	0.0	No correlation, indicating orthogonality.
PC3 vs PC5	0.0	No correlation, indicating orthogonality.
PC4 vs PC4	1.0	Perfect positive correlation, as expected.
PC4 vs PC5	0.0	No correlation, indicating orthogonality.
PC5 vs PC5	1.0	Perfect positive correlation, as expected.



Scatter plot matrix for the PCA-reduced feature space. Each small graph shows the relationship between two principal components (PCs). The diagonal graphs are histograms, representing the distribution of each individual PC. Because PCA aims to remove correlation between the components, the scatter plots between different PCs (off-diagonal plots) should show no discernible pattern or correlation, which is consistent with what is seen in the graph: the plots appear as roughly circular clouds, indicating no relationship between different PCs.

- The distribution of each principal component is shown in the histograms on the diagonal of the scatter plot matrix, with varying degrees of spread.
- The spread and clustering in each histogram suggest different variances for each principal component, which is consistent with the idea that each principal component captures a different amount of the total variance in the data.

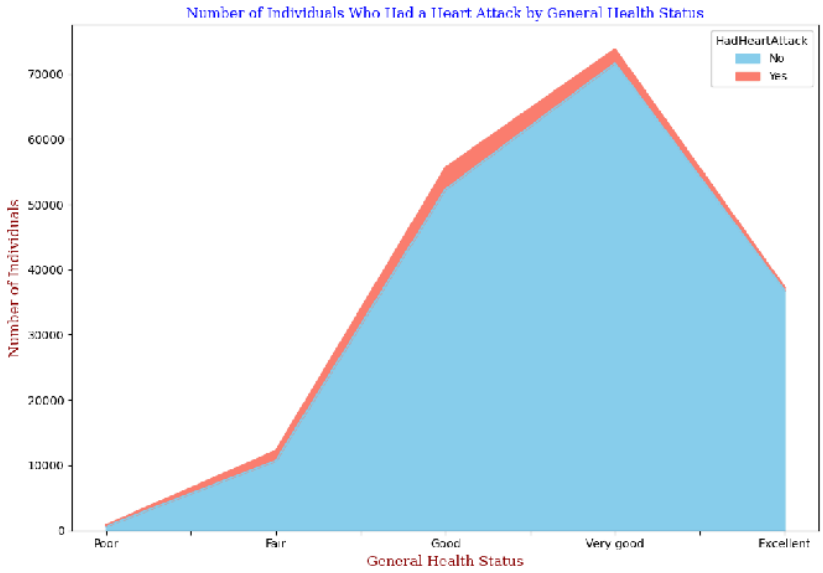
- These visuals confirm the effectiveness of PCA in creating features that capture the maximum variance in the data while ensuring these new features are uncorrelated with each other.

Statistics

Data Visualisation

Area Plot

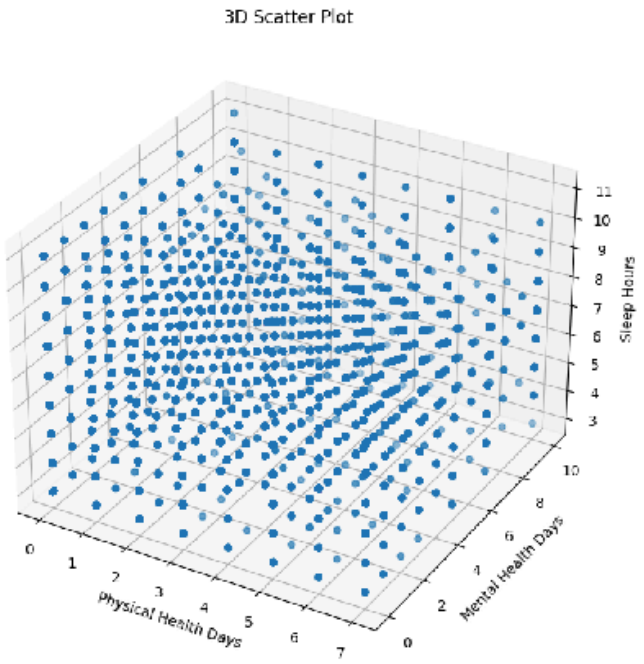
Compare the number of individuals with a 'HadHeartAttack' across different 'GeneralHealth' statuses



1. **General Health Distribution:** Most individuals in the dataset report being in 'Very good' or 'Excellent' general health. This could indicate a generally healthy population or a tendency for individuals to report their health status positively.
2. **Heart Attack Incidence:** There is a visible proportion of individuals across all categories of self-reported general health who have experienced a heart attack. Notably, even among those reporting 'Excellent' health, there are individuals who have had a heart attack.

3D PLOT

This plot shows the relationship between three different variables: **PhysicalHealthDays**,



MentalHealthDays, and **SleepHours**

Variable Relationships: Each point in the plot represents an individual's reported number of days with poor physical health, poor mental health, and their average sleep hours.

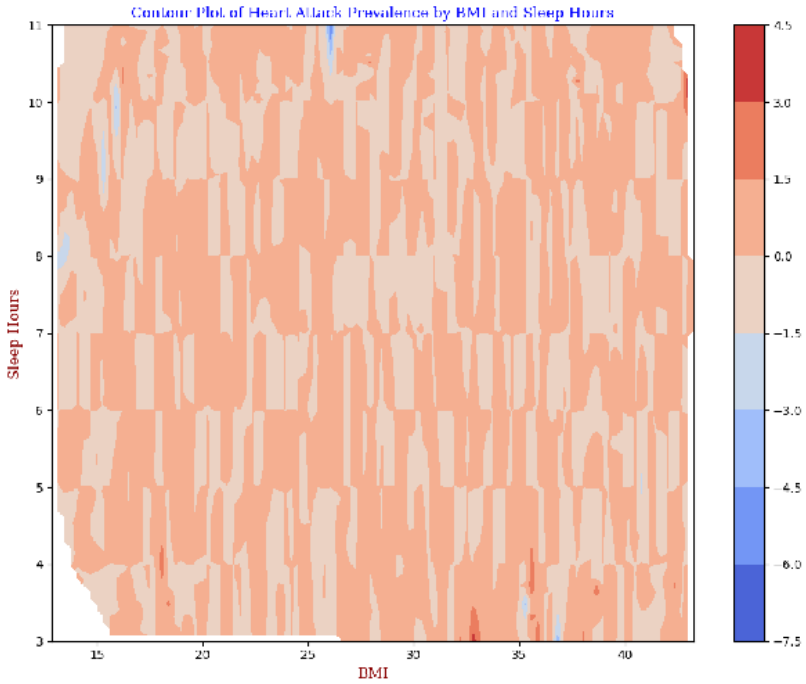
Clustering: If there are clusters of points, this might indicate groups of individuals with similar health metrics. For example, a cluster of points towards the origin (low on all three axes) might indicate individuals with good physical and mental health and sufficient sleep.

Contour Plot

Contour plots are a way to represent a three-dimensional surface on a two-dimensional plane. It is typically used to show the gradient (or potential change) of a variable z over two other variables x and y.

a contour plot to show the density of individuals across two key metrics, perhaps BMI and SleepHours, with contours indicating the prevalence of HadHeartAttack.

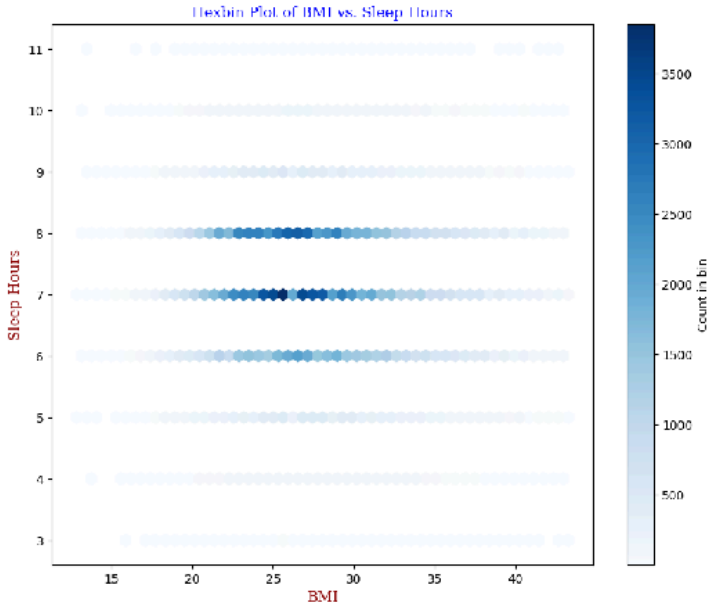
Contour plot aims to visualize the prevalence of heart attacks across two key metrics: BMI and Sleep Hours.



1. **Interpolation:** The contour plot uses cubic interpolation to estimate the prevalence of heart attacks, but the resulting image shows large areas of uniform color with sharp boundaries, which suggests that the interpolation might not be accurately capturing the underlying distribution.
2. **Data Density:** The plot does not appear to show a clear gradient or distinct regions of varying heart attack prevalence. Instead, it displays large blocks of color, indicating that the interpolation may be overfitting or not well-suited to the sparsity/distribution of the data.
3. **Color Mapping:** The color bar suggests that the interpolated values range from negative to positive, which is unusual for a binary outcome variable like heart attack prevalence. A binary outcome would typically be visualized with a clear distinction between the two categories.

Hexbin plot

Distribution of individuals by BMI and Sleep Hours

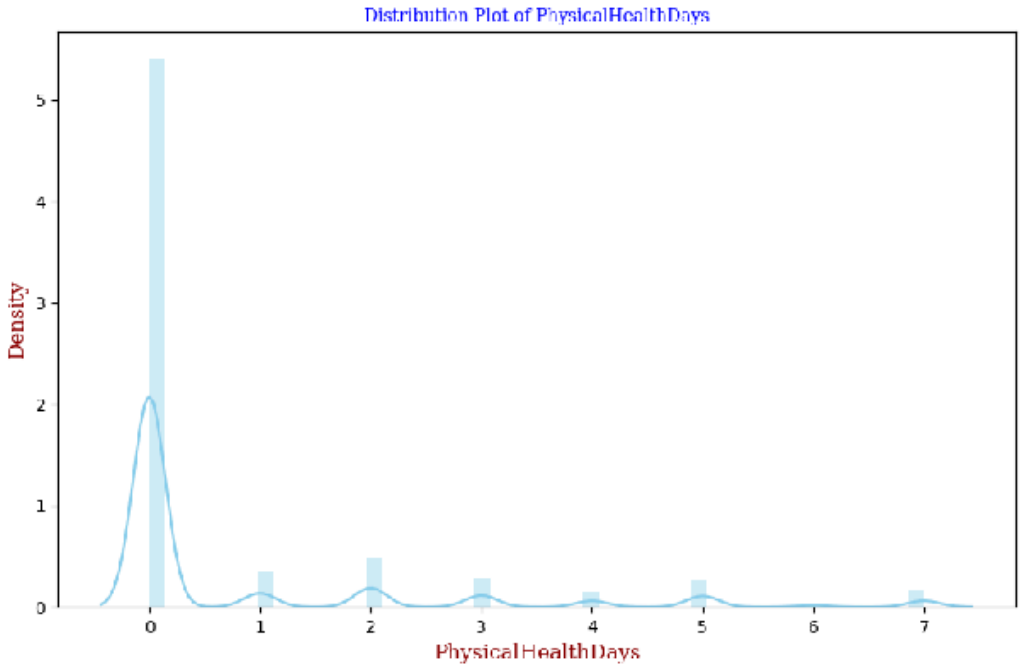


1. **Concentration of Data Points:** There is a clear concentration of data points around a BMI of 25 to 30 and Sleep Hours of about 7 to 8 hours. This is where the darkest hexagons are, indicating the most common BMI and sleep duration reported by individuals in the dataset.
2. **Sleep Duration:** Most individuals report getting between 6 to 9 hours of sleep, which is consistent with common sleep recommendations for adults.
3. **BMI Distribution:** The distribution of BMI values centers around the overweight threshold (BMI of 25 to 30), with fewer individuals reporting lower or higher BMIs.

This hexbin plot provides a visual summary of where the majority of individuals in the dataset fall concerning BMI and sleep duration and highlights the commonality of sleep duration reporting.

Distplot

Kernel Density Estimate (KDE) to show the density distribution of the number of days individuals reported their physical health as not good.

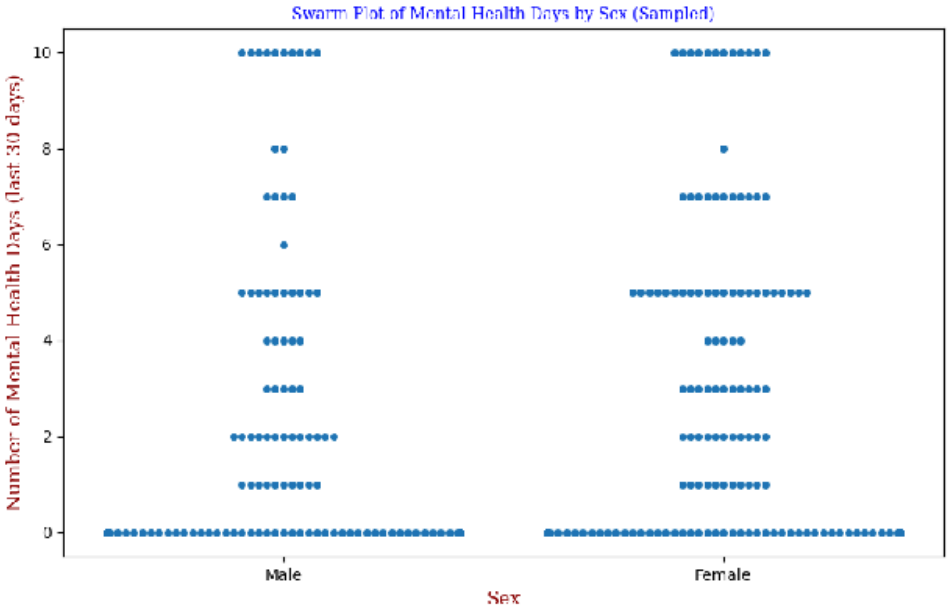


1. **Concentration at Zero:** The peak at zero suggests that a significant number of individuals reported no days with poor physical health, indicating a healthy sample or a tendency to report no health issues.
2. **Long Tail:** If the plot shows a long tail extending to the right, it would indicate that while most people report few poor physical health days, there is a smaller group that experiences poor physical health more frequently.
3. **KDE Curve:** The KDE curve helps to visualize the probability density of different values in PhysicalHealthDays. Peaks in the KDE curve represent the most common values, while valleys represent less common values.

Skewness: If the plot is not symmetric and has a longer tail on one side, this indicates skewness in the

data. A long right tail, as might be inferred from your description, would indicate positive skewness, meaning there are individuals with a high number of poor physical health days that are outliers compared to the rest of the population.

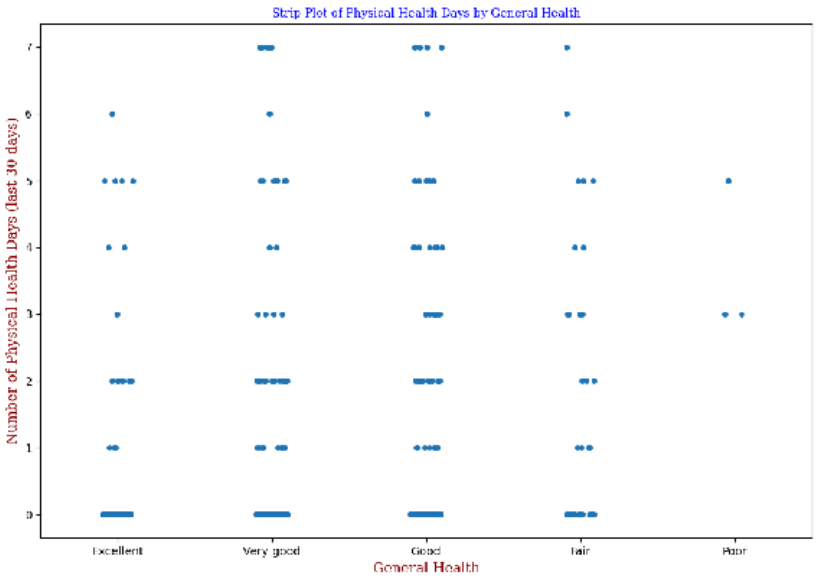
Swarm Plot



Distribution by Sex: The plot will show how MentalHealthDays are distributed between the sexes.

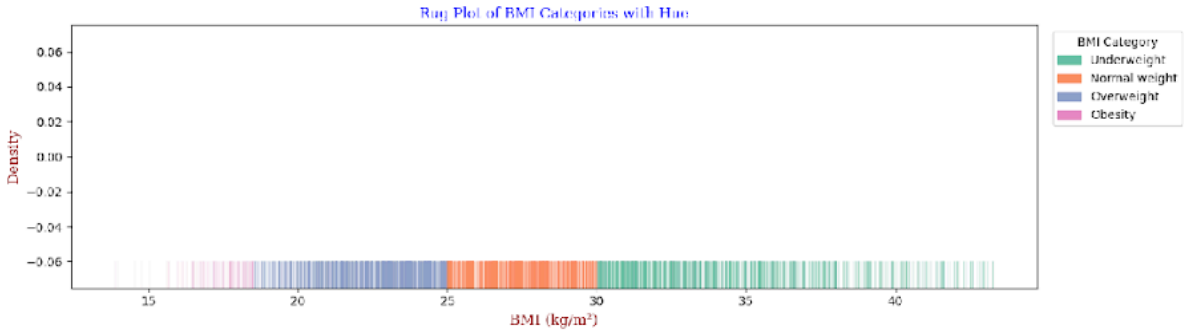
1. **Common Ranges:** By looking at where the points are most concentrated, you can determine the most common range of MentalHealthDays for each sex. If the dots are clustered towards the bottom, it suggests that most individuals report fewer mental health days.
2. **Outliers:** Points that stand apart from the main clusters could indicate individuals with an unusually high or low number of MentalHealthDays, which might warrant further investigation.

Strip Plot



1. **General Health Rating Impact:** An increase in the number and height of dots as you move from 'Excellent' to 'Poor' would demonstrate that people who rate their general health lower also tend to report more physical health days, suggesting a correlation between general health perception and actual health days reported.
2. **Outliers:** Individual dots that are separated from the main clusters could indicate outliers. For example, someone with an 'Excellent' general health rating but a high number of physical health days could be an outlier worth investigating.
3. **Dense Clustering at Lower Values:** If most of the dots across all health categories are clustered near the bottom of the y-axis, this indicates that a majority of individuals report few physical health problems regardless of their general health perception.
4. **Variability in Reports:** If the distribution of dots within categories like 'Fair' and 'Poor' is more spread out compared to 'Excellent' and 'Very good', this suggests greater variability in the number of physical health days reported by individuals with lower self-rated health statuses.

Rug Plot



The distribution of BMI values categorized into four distinct groups: Underweight, Normal weight, Overweight, and Obesity.

1. **BMI Distribution:** The rug plot indicates individual BMI values along the horizontal axis. Each short line represents a single data point from the sampled subset.
2. **Density of Categories:** The concentration of lines in different color segments gives a visual representation of the density of data points in each BMI category. A denser collection of lines suggests more individuals fall into that BMI category.

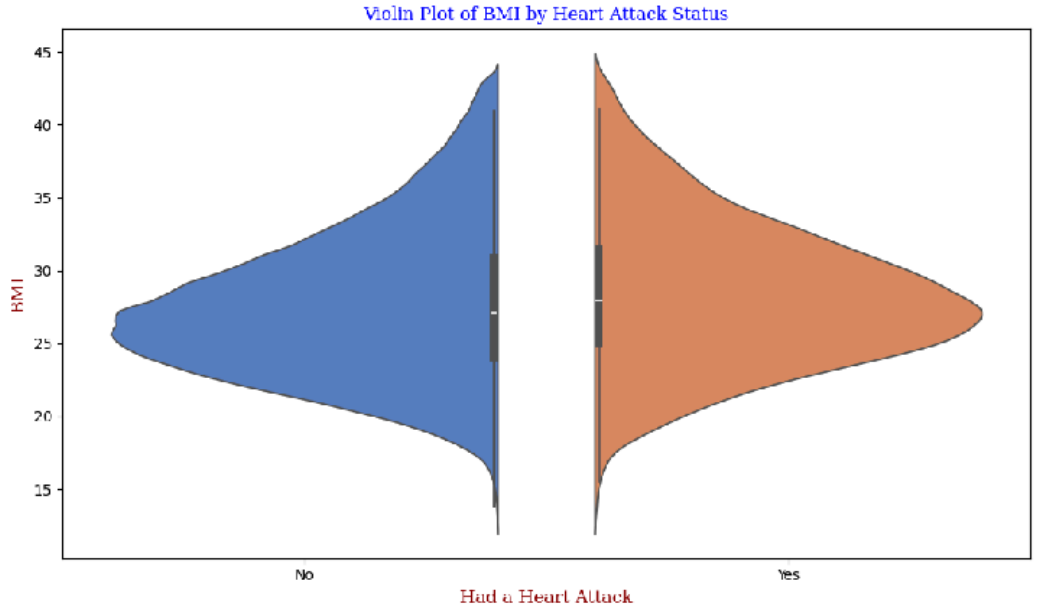
Range of BMI Values: The spread of each color shows the range of BMI values for each category. For example, the blue lines might represent the 'Underweight' category and only appear below the BMI value of 18.5.

Overlaps and Gaps: Overlaps between different colors can indicate transitional zones where BMI values are close to the thresholds between categories. Conversely, gaps or less dense areas may indicate fewer individuals with those BMI values.

Population Health: Generally, a heavier concentration of lines in the 'Normal weight' category would indicate a healthier population distribution, whereas a concentration in the 'Overweight' and 'Obesity'

categories may indicate a population with higher weight-related health risks.

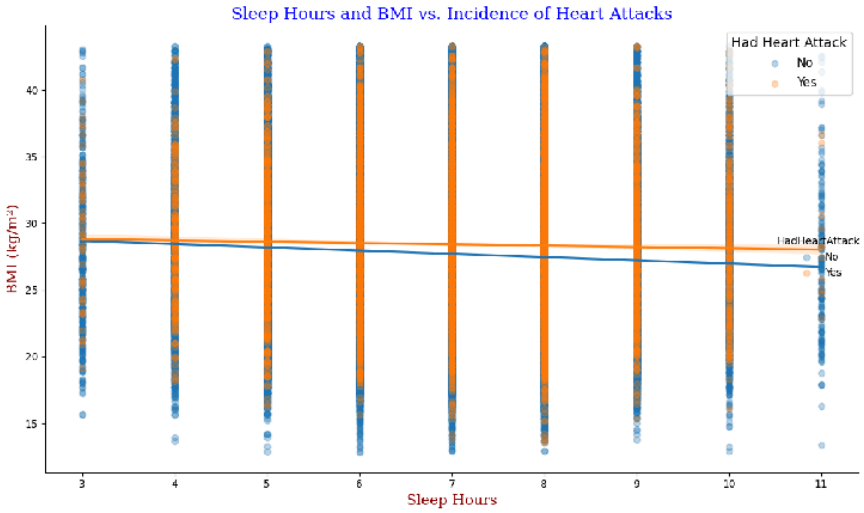
Violin Plot



The violin plot suggests a comparison between the BMI distributions of individuals based on whether they have had a heart attack or not if the violin on the 'Yes' side (indicating individuals who have had a heart attack) is broader or extends further towards higher BMI values than the 'No' side, it may imply that having a higher BMI is more common among those who have experienced a heart attack. Conversely, if the 'No' side is just as broad or broader at high BMI values, it might indicate that the relationship between BMI and heart attacks in this sample is not straightforward..

LM Plot

This scatter plot seems to plot individual data points for BMI against the number of sleep hours, with the color of the points indicating whether the individual has had a heart attack (orange for 'Yes', blue for 'No'.



Distribution of Data: Data points are spread across the range of sleep hours, indicating variability in sleep patterns among the individuals. BMI also varies, with a concentration of points in the overweight to obese categories (BMI > 25 kg/m²).

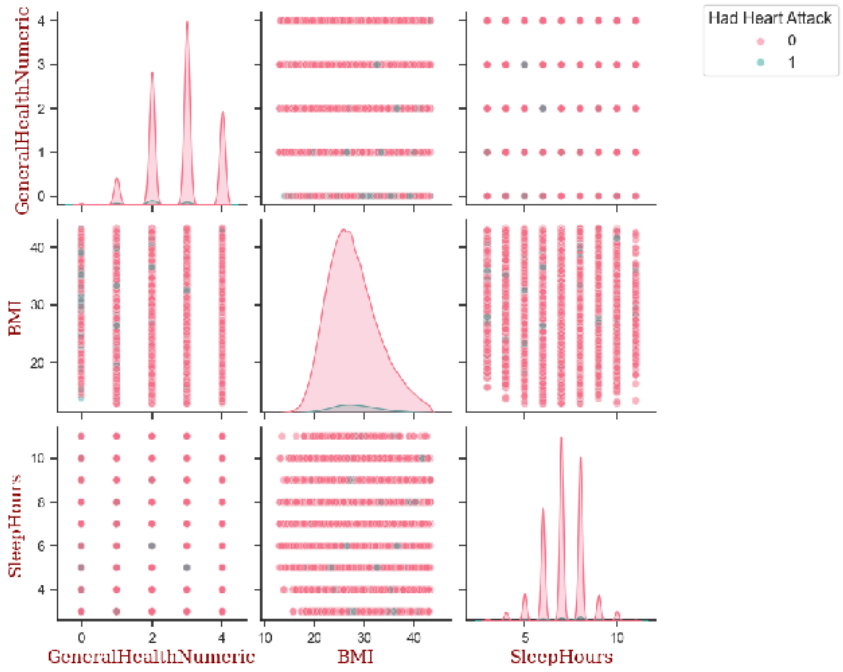
Concentration of Heart Attacks: The orange points represent individuals who have had a heart attack. At a glance, there doesn't appear to be a concentration of these points in any specific area of the plot, suggesting no immediate visible correlation between these two variables and heart attacks.

Sleep Hours: The distribution of sleep hours for individuals who have had a heart attack doesn't seem to significantly differ from those who haven't. Both groups have data points spread across the sleep hour axis.

BMI Values: Similarly, BMI values for individuals who have had a heart attack are spread throughout the range of BMI values and overlap significantly with those who haven't had a heart attack.

Pair Plot

GeneralHealthNumeric, **BMI**, and **SleepHours**, with the incidence of having had a heart attack (HadHeartAttack)



Distribution of Health Status: The top left distribution plot for GeneralHealthNumeric seems to show that fewer individuals report being in very poor health. However, the plot indicates that the distribution is not normal and suggests that the numeric encoding of general health status may not be uniformly distributed.

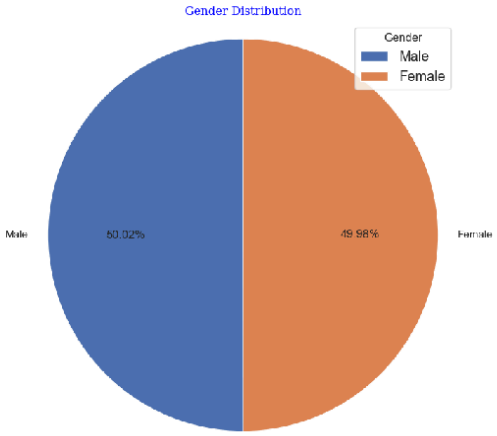
BMI Distribution: The distribution plot for BMI (middle of the diagonal) indicates a right-skewed distribution with most individuals concentrated in the overweight and obese categories, which is typical in many adult populations.

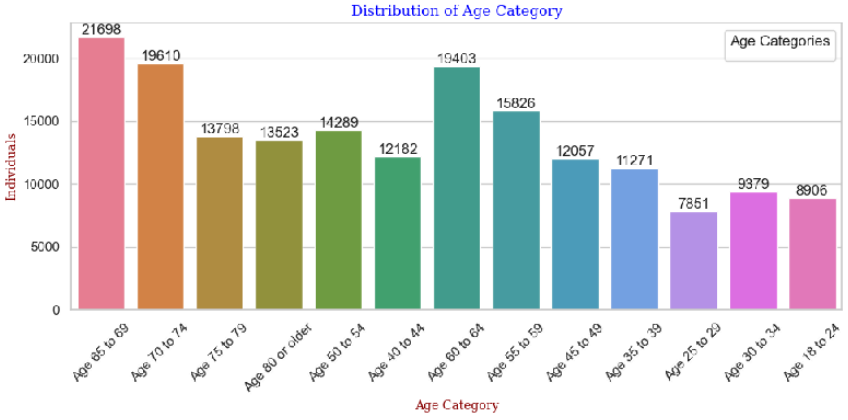
Sleep Hours Distribution: The bottom right distribution plot for SleepHours shows peaks around certain values, which may suggest that the majority of individuals report standard sleep hours, such as 7-8 hours per night, with fewer reporting very low or very high amounts of sleep.

Relationship Between Health Status and BMI: The scatter plot between GeneralHealthNumeric and BMI (center left) does not show any clear pattern or correlation. This could suggest that self-reported general health status is not directly related to BMI or that the relationship is not linear.

Gender and Age Distribution across the Dataset:

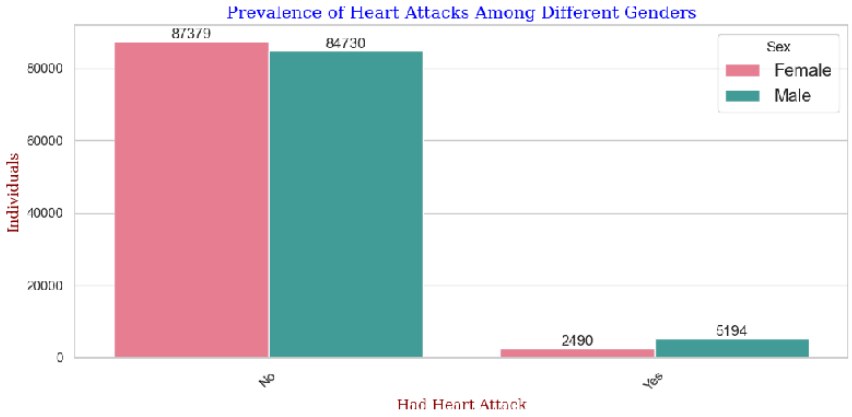
How does the prevalence of heart disease and major risk factors vary across different demographic groups (sex, age, race/ethnicity)





Above piechart shows that females are more than males in the data population.
Mostly individuals are aged around 65-69
Mostly individuals are aged around 65-69

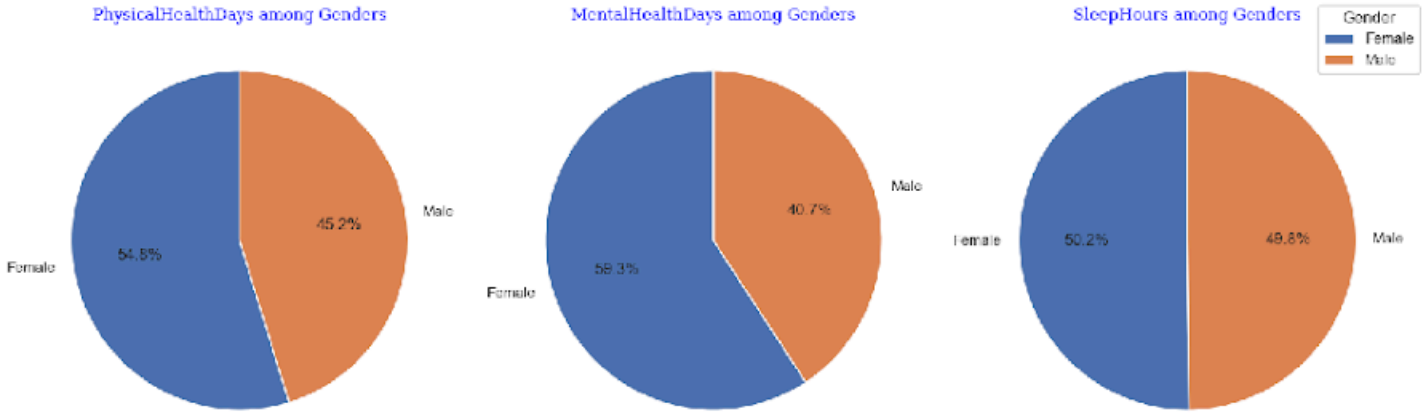
Prevalence of Heart Attacks Among Different Genders



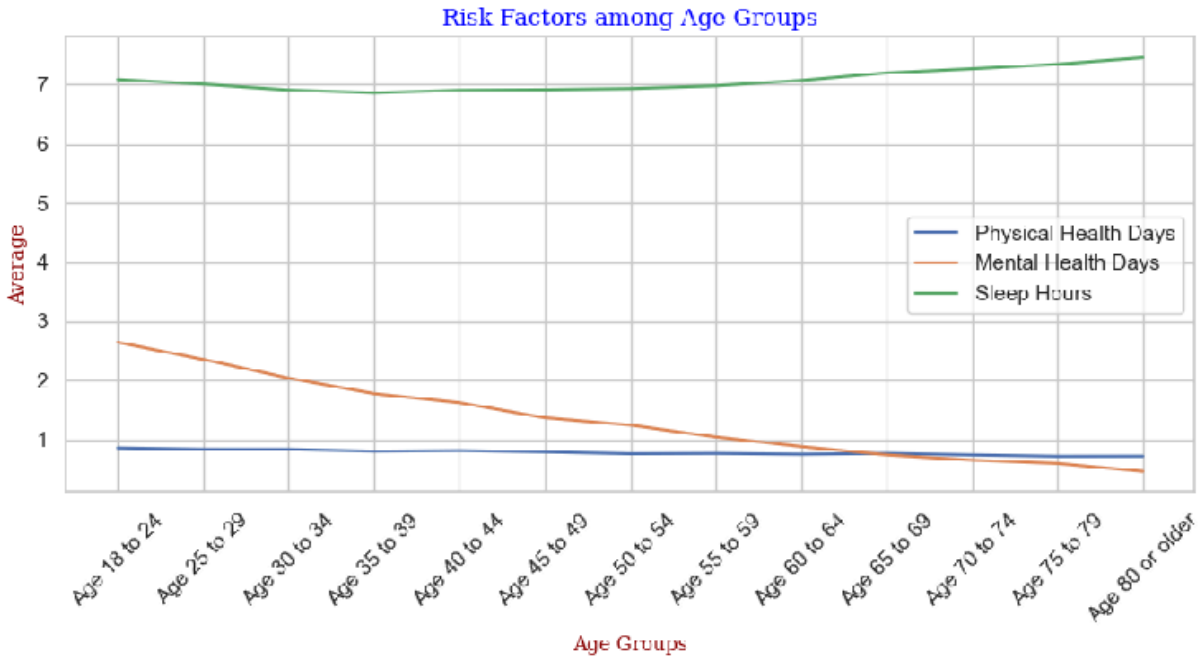
Many individuals did not experience any heart disease. However, among those who did, the majority were males

Demographic Variance

1. **WeightInKilograms and BMI (0.83):** There's a strong positive correlation
2. **HeightInMeters and WeightInKilograms (0.56):** A moderate positive correlation
3. **PhysicalHealthDays and MentalHealthDays (0.15):** A weak positive correlation implies a slight tendency for individuals with more physically unhealthy days to also have more mentally unhealthy days.



Risk Factors among different Age Groups:

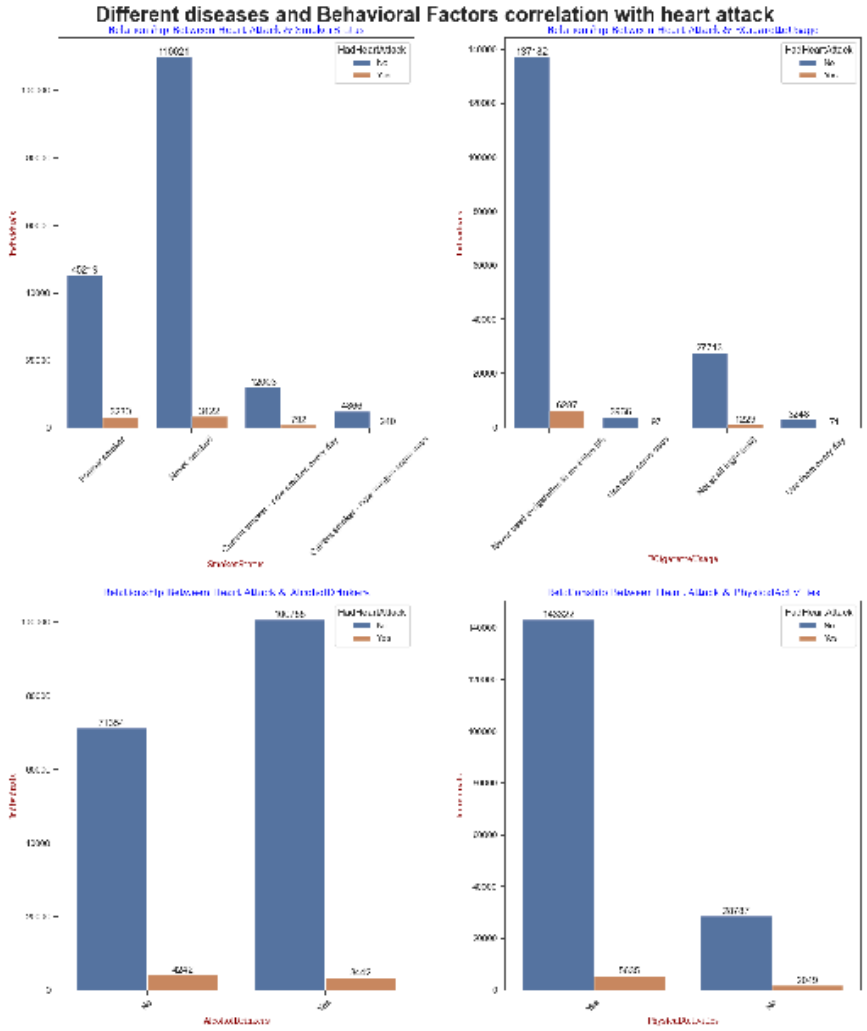


WeightInKilograms and BMI (0.83): There's a strong positive correlation

HeightInMeters and WeightInKilograms (0.56): A moderate positive correlation

PhysicalHealthDays and MentalHealthDays (0.15): A weak positive correlation implies a slight tendency for individuals with more physically unhealthy days to also have more mentally unhealthy days.

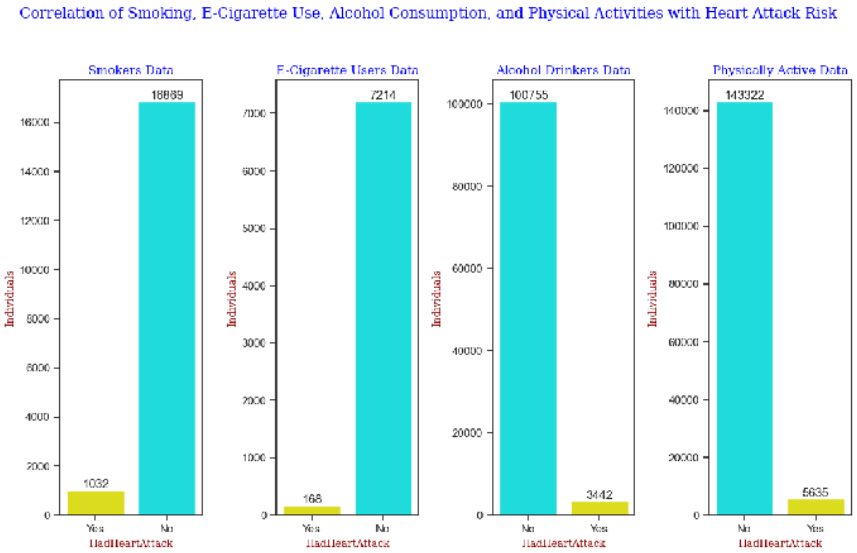
Explore the relationships between health behaviors (physical activity, alcohol consumption etc) and the prevalence of heart disease



The health behaviors may include Physical Activity, Alcohol Drinkers,smoking, e-cigarette use, relate to heart diseases.

Individuals who used to smoke in the past are often affected by heart attacks. Additionally, it's noteworthy that individuals who have never used e-cigarettes and do not consume alcohol also experience heart attacks,and individuals who are active in physical activity also experienced which is quite surprising

Investigating how the presence of these factors is connected to the prevalence of heart disease



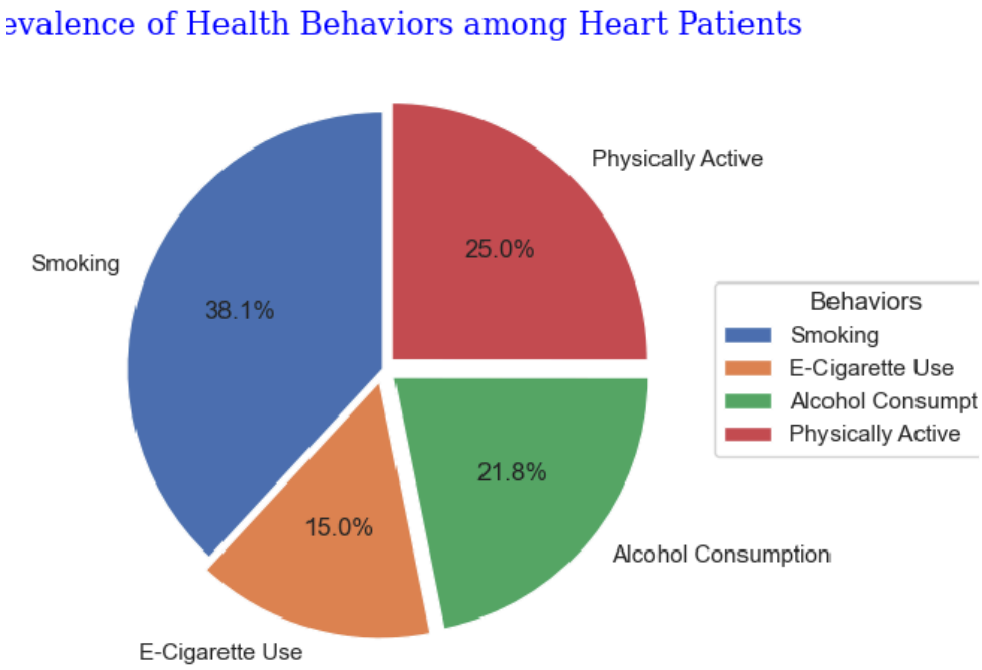
Smokers: Both smokers and non-smokers experience heart attacks, but most smokers haven't had one.

E-Cigarette Users: Like smoking, more e-cigarette users avoid heart attacks than suffer them, showing a link but no direct cause of heart attacks.

Alcohol Drinkers: Most people who drink alcohol don't have heart attacks, indicating a potential link but no clear cause.

Physically Active: Being active is more common among those without heart attacks, suggesting that exercise might help prevent them.

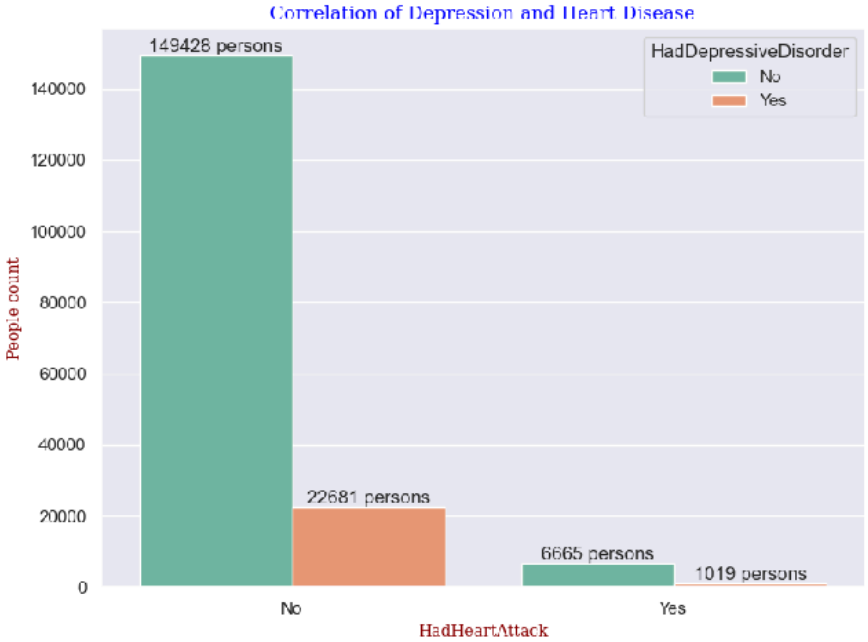
Prevalence of Health Behaviors among Heart Patients



Females experience a higher percentage of days with poor physical health and poor mental health compared to males. This means that, on average, females report more days of poor physical and mental health in the past 30 days than males.

The average number of sleep hours is quite similar between males and females.

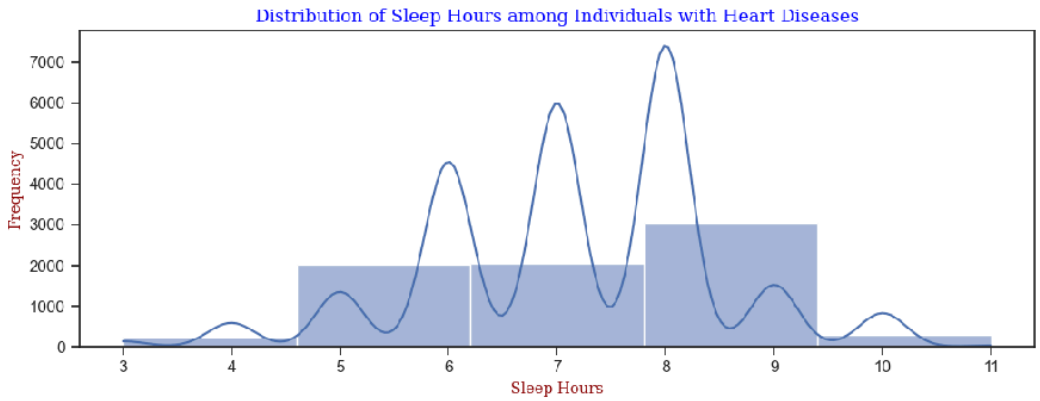
Effect of Mental Health on Heart Disease



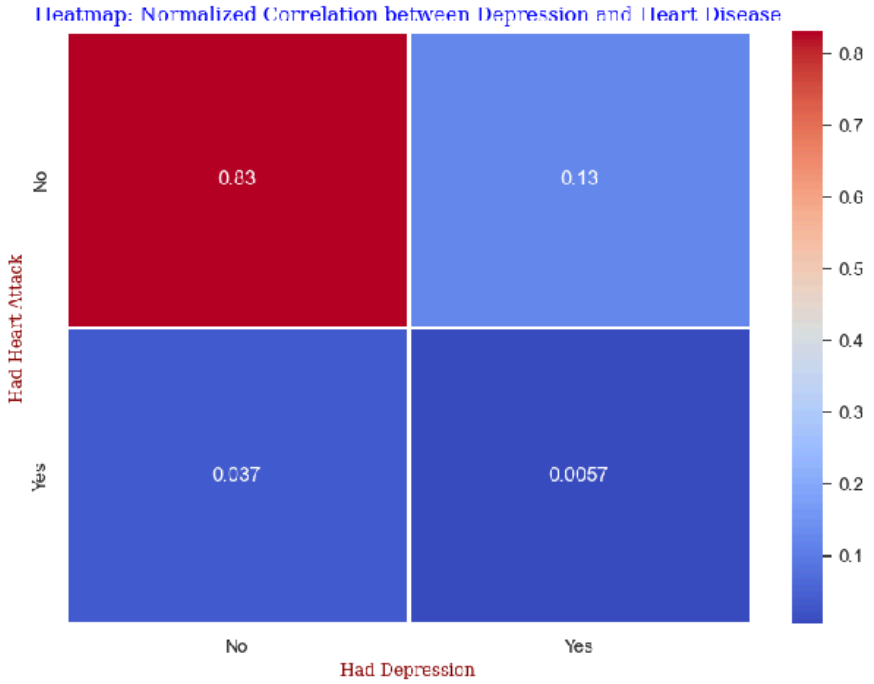
This chart shows that most of the people neither have heart disease nor depressive disorder.

Among people having heart problems, a significant big portion of 6000+ have depressive disorders also.

Distribution of Sleep Hours among Individuals with Heart Diseases



Prevalence of Health Behaviors among Heart Patients



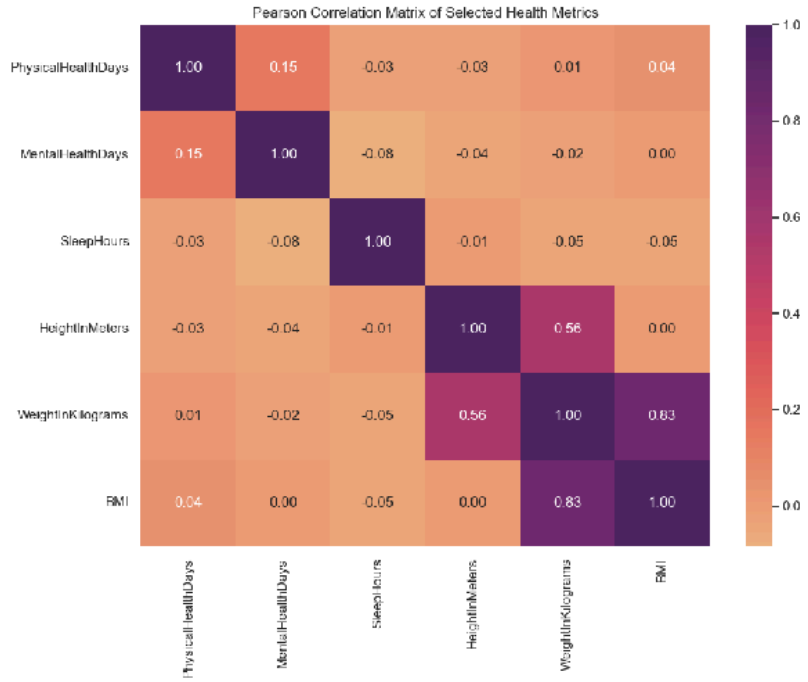
Mostly 83% people do not have depressive disorder and also have a good heart health.

13% people are suffering from depression but also don't have a heart disease.

Around 4% people don't have depressive disorder but facing heart disease.

Very few people i.e around 0.57% have faced depressive disorder before developing heart problems.

Correlation Heatmap of Numerical Columns



1. **WeightInKilograms and BMI (0.83):** There's a strong positive correlation
2. **HeightInMeters and WeightInKilograms (0.56):** A moderate positive correlation
3. **PhysicalHealthDays and MentalHealthDays (0.15):** A weak positive correlation implies a slight tendency for individuals with more physically unhealthy days to also have more mentally unhealthy days.

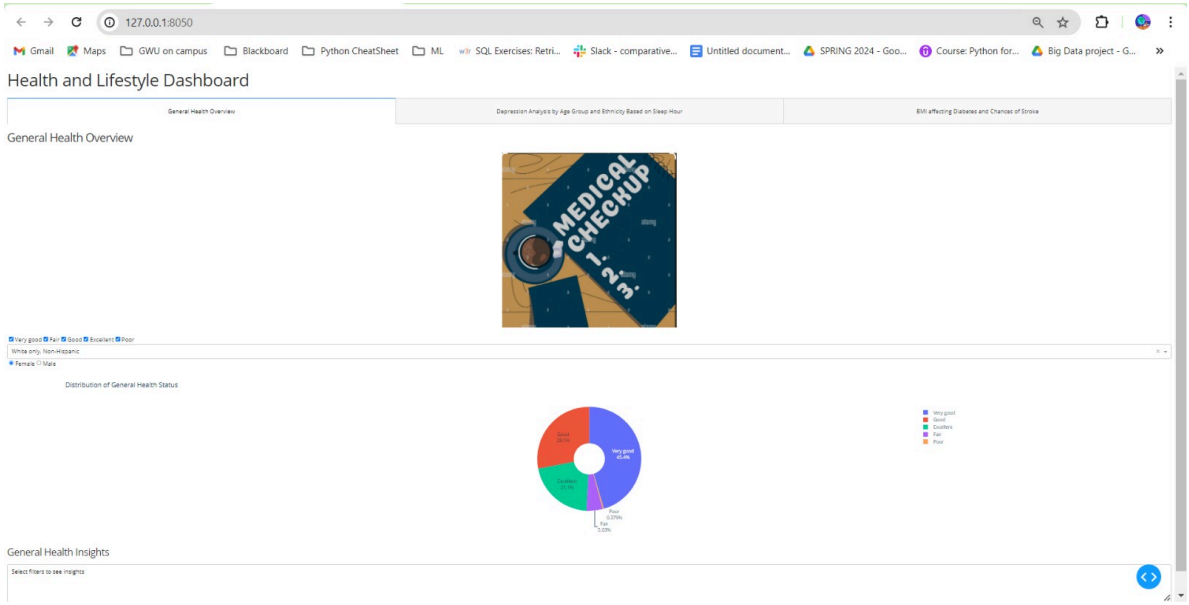
CONCLUSION

Study underscores the complexity of heart disease prevalence, which is influenced by a myriad of factors including age, sex, health behaviors, and lifestyle choices. Smoking was identified as a significant risk factor, but the surprising prevalence of heart disease among physically active individuals and non-users of e-cigarettes and alcohol suggests that heart disease is a multifaceted issue requiring a holistic approach to prevention and treatment.

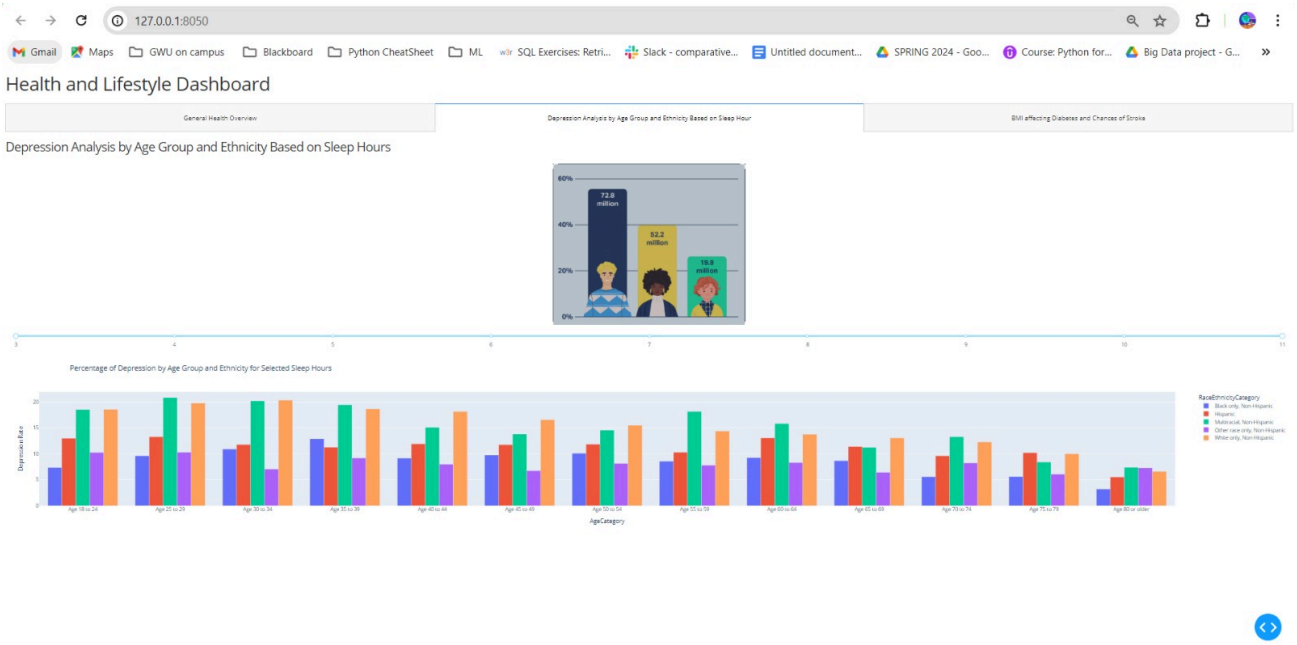
The color intensity of heatmap and the values, thereby indicate that having a depressive disorder is less common among those who had a heart attack than among those who did not.

Overall, the heatmap indicates that the strongest relationship is between not having depression and not having a heart attack. However, the presence of depression does not seem to have a strong correlation with the occurrence of a heart attack within this dataset.

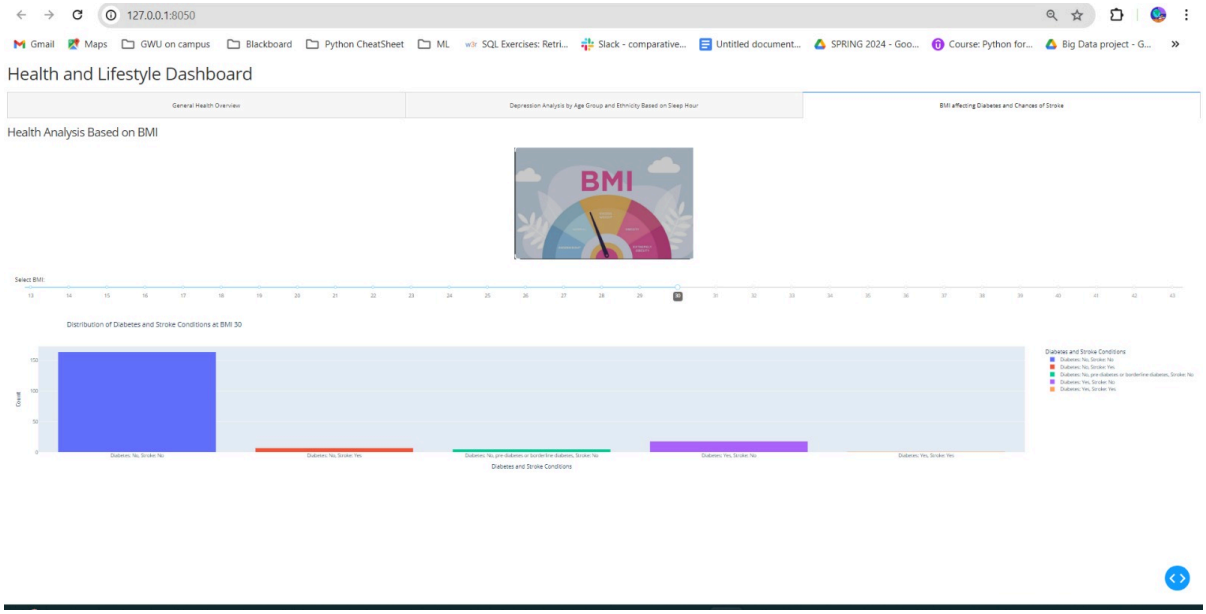
Dashboard



The dash tells us about general health overview of different ethnicity groups , and gender . A download button is for downloading the text noted by user



The depression analysis based on different age group and ethnicity tells us about the depression rate among them by changing the sleep range



Health analysis based on bmi tells us about the diabetes and stroke conditions faced by individuals in the dataset based on different BMI counts .

References

1. Indicators of heart Disease (2022 UPDATE). (2023, October 12). Kaggle. https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease/data

2. Le, P., Casper, M., & Vaughan, A. S. (2022). A dynamic visualization tool of local trends in heart disease and stroke mortality in the United States. Preventing Chronic Disease, 19. https://doi.org/10.5888/pcd19.220076

3. Comprehensive Analysis of Heart Disease Prediction: Machine Learning approach. (2022, October 7). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9972035

Appendix:

Static plot .py

```
#%%
import pandas as pd
u2 = "heart_2022_with_nans.csv"
data = pd.read_csv(u2)
#%%
print(data.head())
print(data.tail())
#%%
nan_summary = data.isna().sum()
print(nan_summary)
#%%
from prettytable import PrettyTable

# Create a PrettyTable object
table = PrettyTable()

# Set the column names
table.field_names = ["Column", "Missing Values"]

# Add rows with column names and missing values
for column, value in nan_summary.items():
    table.add_row([column, value])

# Set the title for the table
table.title = 'Null Values in the Dataset'

# Print the table
print(table)
#%%
data = data.dropna()
print(data)
#%%
data = data.drop_duplicates()
print(data)
nan_summary = data.isna().sum()
print(nan_summary)


#%%
df = data.copy()
print(df.info())
#%%
print(df.describe().to_string())
#%%
df.reset_index(drop=True, inplace=True)
print(df)
#%%

# Outliers
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
# Create box plots with subplots
fig, axes = plt.subplots(3, 2, figsize=(18, 15))  # Adjust the layout (3, 4) as
needed based on your number of columns

# Plot each column on a separate subplot
num_cols = df.select_dtypes(include=[np.number]).columns
ax_idx = 0  # index for axes
for col in num_cols:
   # Make sure not to go out of index range for axes
   if ax_idx < axes.size:
       ax = axes.flatten()[ax_idx]  # Get the current Axes object from the
flattened array of axes
       df.boxplot(column=[col], ax=ax, flierprops=dict(marker='o',
markerfacecolor='blue', markersize=8, linestyle='none', alpha=0.2))
       # Set title and labels with custom font properties
       ax.set_title(col, fontdict={'fontsize': 'large', 'fontweight': 'bold',
'color': 'blue', 'fontname': 'serif'})
       ax.set_xlabel('Value', fontdict={'fontsize': 'large', 'fontweight': 'bold',
'color': 'darkred', 'fontname': 'serif'})
       ax.set_ylabel('Frequency', fontdict={'fontsize': 'large', 'fontweight':
'bold', 'color': 'darkred', 'fontname': 'serif'})
       ax_idx += 1  # increment to next subplot axis

plt.tight_layout()
plt.show()
#%%
#Identifying Outliers
numerical_columns = df.select_dtypes(include=['int', 'float']).columns

for column in numerical_columns:
   # Calculate quartiles and IQR
   Q1 = df[column].quantile(0.25)
   Q3 = df[column].quantile(0.75)
   IQR = Q3 - Q1

   # Identify outliers
   lower_bound = Q1 - 1.5 * IQR
   upper_bound = Q3 + 1.5 * IQR

   outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]

   print(f"Outliers in {column}:")
   print(outliers.count())
   print(round(lower_bound,2))
   print(round(upper_bound,2))
   print("_----------------------------------------------------------------------")
#%%
from prettytable import PrettyTable
import pandas as pd
from prettytable import PrettyTable

# Assuming 'df' is your DataFrame
numerical_columns = df.select_dtypes(include=['int', 'float']).columns

# Create a PrettyTable object
```

```
table = PrettyTable()

# Add columns to the table
table.field_names = ["Column Name", "Outliers Count", "Lower Bound", "Upper Bound"]

for column in numerical_columns:
   # Calculate quartiles and IQR
   Q1 = df[column].quantile(0.25)
   Q3 = df[column].quantile(0.75)
   IQR = Q3 - Q1

   # Identify outliers
   lower_bound = Q1 - 1.5 * IQR
   upper_bound = Q3 + 1.5 * IQR

   outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
   outliers_count = outliers[column].count()  # Count of outliers

   # Add a row to the table with the data for this column
   table.add_row([
       column,
       outliers_count,
       round(lower_bound, 2),
       round(upper_bound, 2)
   ])
table.title=("Lower and upper boound of each numeric column")
# Print the table
print(table)

#%%
df['PhysicalHealthDays'].value_counts().sort_values()
#%%
df = df.drop(df[df['PhysicalHealthDays'] > 7.5].index)
df.reset_index(drop=True, inplace=True)
#%%
df.shape
#%%
df['PhysicalHealthDays'].value_counts().sort_values()
#%%
df['MentalHealthDays'].value_counts().sort_values()
#%%
df = df.drop(df[df['MentalHealthDays'] > 10].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df['MentalHealthDays'].value_counts().sort_values()

#%%
df['SleepHours'].value_counts().sort_values()
#%%
# Dropping records with sleep less than 3 hours
df = df.drop(df[df['SleepHours'] < 3].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df['SleepHours'].value_counts().sort_values()
```

```
#%%
df = df.drop(df[df['SleepHours'] >11].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df['SleepHours'].value_counts().sort_values()

#%%
df['HeightInMeters'].value_counts().sort_values()
#%%
df = df.drop(df[df['HeightInMeters'] <1.4].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df = df.drop(df[df['HeightInMeters'] >2.01].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df['HeightInMeters'].value_counts().sort_values()
#%%
df['WeightInKilograms'].value_counts().sort_values()
#%%
df = df.drop(df[df['WeightInKilograms'] <27.23].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df = df.drop(df[df['WeightInKilograms'] >136.06].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df['WeightInKilograms'].value_counts().sort_values()
#%%
df['BMI'].value_counts().sort_values()
#%%
df = df.drop(df[df['BMI'] <12.84].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df = df.drop(df[df['BMI'] > 43.32].index)
df.reset_index(drop=True, inplace=True)
df.shape
#%%
df['BMI'].value_counts().sort_values()
#%%
df.reset_index(drop=True, inplace=True)
print(df)
df.to_csv("heart_no_nans_clean_no_outliers.csv",index=False)
print(df)
print(df.columns)
#%%
#Post Cleaning Outlier check
# Outlier
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
# Create box plots with subplots
fig, axes = plt.subplots(3, 2, figsize=(18, 15))

# Plot each column on a separate subplot
num_cols = df.select_dtypes(include=[np.number]).columns
ax_idx = 0  # index for axes
for col in num_cols:
    # Make sure not to go out of index range for axes
    if ax_idx < axes.size:
        ax = axes.flatten()[ax_idx]  # Get the current Axes object from the
flattened array of axes
        df.boxplot(column=[col], ax=ax, flierprops=dict(marker='o',
markerfacecolor='blue', markersize=8, linestyle='none', alpha=0.2))
        # Set title and labels with custom font properties
        ax.set_title(col, fontdict={'fontsize': 'large', 'fontweight': 'bold',
'color': 'blue', 'fontname': 'serif'})
        ax.set_xlabel('Value', fontdict={'fontsize': 'large', 'fontweight': 'bold',
'color': 'darkred', 'fontname': 'serif'})
        ax.set_ylabel('Frequency', fontdict={'fontsize': 'large', 'fontweight':
'bold', 'color': 'darkred', 'fontname': 'serif'})
        ax_idx += 1  # increment to next subplot axis

plt.tight_layout()
plt.show()
#%%
#Check for Normality after outlier detection
import pandas as pd
from scipy.stats import normaltest
# Check normality for each numeric column
for col in df.select_dtypes(include=['float64', 'int64']).columns:
    # Remove NaN values from the column
    col_data = df[col].dropna()
    # Apply D'Agostino's K-squared test
    stat, p = normaltest(col_data)
    print(f"Column: {col}")
    print(f"Statistics={stat:.2f}, p-value={p:.2f}")

    # Determine if the column data is normal based on the p-value
    alpha = 0.01  # significance level
    if p > alpha:
        print("The data is normally distributed (fail to reject H0)")
    else:
        print("The data is not normally distributed (reject H0)")
    print()
#%%
#qq plot for sleep hours
import scipy.stats as stats
stats.probplot(df['SleepHours'], dist="norm", plot=plt)
plt.title('QQ Plot of SleepHours after Outlier removal', fontdict={'fontname':
'serif', 'color': 'blue', 'fontsize': 16})
plt.xlabel('Theoretical Quantiles', fontdict={'fontname': 'serif', 'color':
'darkred', 'fontsize': 16})
plt.ylabel('Ordered Values', fontdict={'fontname': 'serif', 'color': 'darkred',
'fontsize': 16})
plt.grid(True)
plt.show()
```

```
#%%
#qq plot for BMI
import scipy.stats as stats
stats.probplot(df['BMI'], dist="norm", plot=plt)
plt.title('QQ Plot of BMI after Outlier removal', fontdict={'fontname': 'serif',
'color': 'blue', 'fontsize': 16})
plt.xlabel('Theoretical Quantiles', fontdict={'fontname': 'serif', 'color':
'darkred', 'fontsize': 16})
plt.ylabel('Ordered Values', fontdict={'fontname': 'serif', 'color': 'darkred',
'fontsize': 16})
plt.grid(True)
plt.show()
#%%
#qq plot for PhysicalHealthDays
import scipy.stats as stats
stats.probplot(df['PhysicalHealthDays'], dist="norm", plot=plt)
plt.title('QQ Plot of PhysicalHealthDays after Outlier removal',
fontdict={'fontname': 'serif', 'color': 'blue', 'fontsize': 16})
plt.xlabel('Theoretical Quantiles', fontdict={'fontname': 'serif', 'color':
'darkred', 'fontsize': 16})
plt.ylabel('Ordered Values', fontdict={'fontname': 'serif', 'color': 'darkred',
'fontsize': 16})
plt.grid(True)
plt.show()
#%%
#qq plot for MentalHealthDays
import scipy.stats as stats
stats.probplot(df['MentalHealthDays'], dist="norm", plot=plt)
plt.title('QQ Plot of MentalHealthDays after Outlier removal',
fontdict={'fontname': 'serif', 'color': 'blue', 'fontsize': 16})
plt.xlabel('Theoretical Quantiles', fontdict={'fontname': 'serif', 'color':
'darkred', 'fontsize': 16})
plt.ylabel('Ordered Values', fontdict={'fontname': 'serif', 'color': 'darkred',
'fontsize': 16})
plt.grid(True)
plt.show()
#%%
#qq plot for WeightInKilograms
import scipy.stats as stats
stats.probplot(df['WeightInKilograms'], dist="norm", plot=plt)
plt.title('QQ Plot of WeightInKilograms after Outlier Removal ',
fontdict={'fontname': 'serif', 'color': 'blue', 'fontsize': 16})
plt.xlabel('Theoretical Quantiles', fontdict={'fontname': 'serif', 'color':
'darkred', 'fontsize': 16})
plt.ylabel('Ordered Values', fontdict={'fontname': 'serif', 'color': 'darkred',
'fontsize': 16})
plt.grid(True)
plt.show()
#%%
#qq plot for HeightInMeters
import scipy.stats as stats
stats.probplot(df['HeightInMeters'], dist="norm", plot=plt)
plt.title('QQ Plot of HeightInMeters after Outlier Removal ', fontdict={'fontname':
'serif', 'color': 'blue', 'fontsize': 16})
plt.xlabel('Theoretical Quantiles', fontdict={'fontname': 'serif', 'color':
'darkred', 'fontsize': 16})
plt.ylabel('Ordered Values', fontdict={'fontname': 'serif', 'color': 'darkred',
```

```
'fontsize': 16})
plt.grid(True)
plt.tight_layout()
plt.show()

#%%
import pandas as pd
url="heart_no_nans_clean_no_outliers.csv"
data=pd.read_csv(url)
print(data.head())
print(data.columns)
#%%
# Check for zero or negative values in numeric columns
numeric_columns = ['PhysicalHealthDays', 'MentalHealthDays', 'SleepHours',
'HeightInMeters',
                   'WeightInKilograms', 'BMI']

zero_negative_check = (data[numeric_columns] <= 0).sum()
zero_negative_check
#%%
from scipy.stats import boxcox
from scipy.special import boxcox1p

# Adjusting zero values by adding 0.1 for the relevant columns
data['PhysicalHealthDays'] = data['PhysicalHealthDays'].apply(lambda x: x + 0.1 if
x == 0 else x)
data['MentalHealthDays'] = data['MentalHealthDays'].apply(lambda x: x + 0.1 if x ==
0 else x)

# Dictionary to store Box-Cox transformed data and the lambda used
boxcox_transformed_data = {}
lambdas = {}

# Apply Box-Cox transformation
for column in numeric_columns:
    # Applying Box-Cox transformation
    fitted_data, fitted_lambda = boxcox(data[column])
    boxcox_transformed_data[column] = fitted_data
    lambdas[column] = fitted_lambda

# Convert transformed data to DataFrame for easier handling
transformed_df = pd.DataFrame(boxcox_transformed_data)
print(transformed_df)
# Display the lambda values used for each column
print(lambdas)
#%%
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import shapiro

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(14, 12))
axes = axes.flatten()

# Custom font settings for title and labels
title_font = {'fontname': 'serif', 'color': 'blue', 'fontsize': 14}
label_font = {'fontname': 'serif', 'color': 'darkred', 'fontsize': 12}
```

```
# Plot histograms for each transformed numeric column and include legends
for i, column in enumerate(numeric_columns):
   sns.histplot(transformed_df[column], kde=True, ax=axes[i], label='KDE')
   axes[i].set_title(f'Transformed {column} (λ={lambdas[column]:.4f})',
**title_font)
   axes[i].set_xlabel(f'{column}', **label_font)
   axes[i].set_ylabel('Frequency', **label_font)
   axes[i].legend()

plt.tight_layout()
plt.show()


from scipy.stats import normaltest

# Perform D'Agostino's K-squared normality test
def da_k_squared_test(x, title):
   # Ensure NaN values are dropped for normality test
   x_clean = x.dropna()
   stats, p = normaltest(x_clean)
   print('='*50)
   print(f'da_k_squared test: {title} dataset: statistics= {stats:.2f}, p-value =
{p:.2f}')
   alpha = 0.01  # Setting a stringent alpha level
   if p > alpha:
       print(f'da_k_squared test: {title} dataset is Normal')
   else:
       print(f'da_k_squared test: {title} dataset is Not Normal')
   print('='*50)

# Apply the defined function to each numeric column of the transformed data
for column in numeric_columns:
   da_k_squared_test(transformed_df[column], column)

#%%
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import pandas as pd
url="heart_no_nans_clean_no_outliers.csv"
df=pd.read_csv(url)
print(df.head())
print(df.shape)
# features  = ['PhysicalHealthDays', 'MentalHealthDays', 'SleepHours',
'HeightInMeters', 'WeightInKilograms', 'BMI']
numeric_cols = df.select_dtypes(include='float64').columns

X = df[numeric_cols]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA()
X_pca = pca.fit_transform(X_scaled)
explained_variance_ratio = pca.explained_variance_ratio_
cumulative_explained_variance = explained_variance_ratio.cumsum()
num_components = len(explained_variance_ratio)
print("Explained Variance Ratio:")
```

```
for i in range(num_components):
   print(f"PC{i+1}: {explained_variance_ratio[i]*100:.2f}%")
print("\nCumulative Explained Variance:")
for i in range(num_components):
   print(f"PC{i+1}: {cumulative_explained_variance[i]*100:.2f}%")
#%%
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import numpy as np

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA()
X_pca = pca.fit_transform(X_scaled)

cumulative_variance_ratio = pca.explained_variance_ratio_.cumsum()

n_components_threshold = np.argmax(cumulative_variance_ratio == 0.95) + 1
features_removed = X_scaled.shape[1] - n_components_threshold

pca_reduced = PCA()
X_pca_reduced = pca_reduced.fit_transform(X_scaled)

print(f"Number of features to be considered per the PCA analysis and assumed
threshold (95% explained variance):", features_removed)
print(f"Explained Variance Ratio (Original Feature Space):",
pca.explained_variance_ratio_)
print(f"Explained Variance Ratio (Reduced Feature Space):",
pca_reduced.explained_variance_ratio_)
#%%
import matplotlib.pyplot as plt

number_of_components = range(1, len(cumulative_variance_ratio) + 1)
cumulative_explained_variance = cumulative_variance_ratio * 100

index_95 = next(i for i, explained_variance in
enumerate(cumulative_explained_variance) if explained_variance >= 95)

plt.plot(number_of_components, cumulative_explained_variance, linestyle='-',
marker='o')

plt.axhline(y=95, color='black', linestyle='--')

plt.axvline(x=index_95 + 1, color='red', linestyle='--')

plt.annotate('95% Explained Variance', xy=(0, 95), xytext=(5, 90),
             arrowprops=dict(facecolor='black', shrink=0.05))

plt.annotate('Optimum Number of Features', xy=(index_95 + 1, 0), xytext=(index_95 +
5, 5),
             arrowprops=dict(facecolor='red', shrink=0.05))


plt.xlabel('Number of Components',font='serif',fontsize=20,color='darkred')
plt.ylabel('Cumulative Explained Variance
```

```
(%)',font='serif',fontsize=20,color='darkred')
plt.title('Cumulative Explained Variance vs. Number of
Components',font='serif',fontsize=20,color='blue')
plt.grid(True)
plt.show()


#%%
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import numpy as np

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

singular_values_original = np.linalg.svd(X_scaled, compute_uv=False)
condition_number_original = np.max(singular_values_original) /
np.min(singular_values_original)

pca = PCA(n_components=0.95)
X_reduced = pca.fit_transform(X_scaled)
singular_values_reduced = np.linalg.svd(X_reduced, compute_uv=False)
condition_number_reduced = np.max(singular_values_reduced) /
np.min(singular_values_reduced)

print("Singular values for the reduced feature space:", singular_values_reduced)
print("Condition number for the reduced feature space:", condition_number_reduced)
print("Singular values for the original feature space:", singular_values_original)
print("Condition number for the original feature space:",
condition_number_original)



correlation_matrix = np.corrcoef(X_reduced, rowvar=False)

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('HeatMap of Pearson Correlation Coefficient
Matrix',font='serif',fontsize=20,color='blue')
plt.xlabel("features",font='serif',fontsize=20,color='darkred')
plt.ylabel("features",font='serif',fontsize=20,color='darkred')
plt.show()



import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Convert the reduced PCA data into a DataFrame

# and 'X_reduced' is the PCA-reduced data

# If you do not have pca.components_, you can use a range to generate PC names
pc_names = [f"PC{i+1}" for i in range(X_reduced.shape[1])]
df_reduced = pd.DataFrame(X_reduced, columns=pc_names)
```

```
# Create the scatter plot matrix
sns.pairplot(df_reduced)
plt.suptitle('Scatter Plot Matrix for PCA-Reduced Feature
Space',font='serif',fontsize=20,color='blue')
plt.tight_layout()
plt.show()

#%%
from prettytable import PrettyTable
corr_table = PrettyTable(["Comparison", "Correlation Coefficient", "Observations"])
# Since the correlation matrix is symmetric and the diagonal is always 1, we only
show unique comparisons
comparisons = [
    ("PC1 vs PC1", 1.00, "Perfect positive correlation, as expected."),
    ("PC1 vs PC2", 0.00, "No correlation, indicating orthogonality."),
    ("PC1 vs PC3", 0.00, "No correlation, indicating orthogonality."),
    ("PC1 vs PC4", 0.00, "No correlation, indicating orthogonality."),
    ("PC1 vs PC5", 0.00, "No correlation, indicating orthogonality."),
    ("PC2 vs PC2", 1.00, "Perfect positive correlation, as expected."),
    ("PC2 vs PC3", 0.00, "No correlation, indicating orthogonality."),
    ("PC2 vs PC4", 0.00, "No correlation, indicating orthogonality."),
    ("PC2 vs PC5", 0.00, "No correlation, indicating orthogonality."),
    ("PC3 vs PC3", 1.00, "Perfect positive correlation, as expected."),
    ("PC3 vs PC4", 0.00, "No correlation, indicating orthogonality."),
    ("PC3 vs PC5", 0.00, "No correlation, indicating orthogonality."),
    ("PC4 vs PC4", 1.00, "Perfect positive correlation, as expected."),
    ("PC4 vs PC5", 0.00, "No correlation, indicating orthogonality."),
    ("PC5 vs PC5", 1.00, "Perfect positive correlation, as expected."),
]

# Populate the table with data
for comp in comparisons:
    corr_table.add_row(comp)

# Print the table
corr_table_str = corr_table.get_string()
print(corr_table_str)
#%%
# #STATISTICS
# import pandas as pd
#
# # Load the dataset
# df = pd.read_csv('heart_no_nans_clean_no_outliers.csv')
#
# # Calculate the mean, median, mode, standard deviation, and variance for the
dataset
# mean = df.mean()
# median = df.median()
# mode = df.mode().iloc[0]  # We take the first mode if there are multiple modes
# std_dev = df.std()
# variance = df.var()
#
# # Display the calculated statistics
# statistics_summary = pd.DataFrame({
#     'Mean': mean,
```

```
#     'Median': median,
#     'Mode': mode,
#     'Standard Deviation': std_dev,
#     'Variance': variance
# })
# print(statistics_summary)
# # For multivariate kernel density estimate, we can plot for two variables as an
example
# import seaborn as sns
# import matplotlib.pyplot as plt
#
# # We will arbitrarily choose the first two numerical columns for the KDE plot
# numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns[:2]
#
# # Generate KDE plot
# sns.jointplot(data=df, x=numerical_columns[0], y=numerical_columns[1],
kind='kde')
# plt.suptitle('Multivariate Kernel Density Estimate')
# plt.show()

#%%
#Area PLot
import pandas as pd
import matplotlib.pyplot as plt

df=pd.read_csv('heart_no_nans_clean_no_outliers.csv')

# We would first ensure that the 'GeneralHealth' categories are ordered from 'Poor'
to 'Excellent'
category_order = ['Poor', 'Fair', 'Good', 'Very good', 'Excellent']

# Ensure the GeneralHealth column is a categorical type with the categories in the
specified order
df['GeneralHealth'] = pd.Categorical(df['GeneralHealth'],
categories=category_order, ordered=True)

# Now we count the number of individuals who had a heart attack within each health
status category
# and plot the results as an area plot
heart_attack_counts =
df.groupby('GeneralHealth')['HadHeartAttack'].value_counts().unstack()

# Reordering the columns to make sure that 'No' is first if it is not
if heart_attack_counts.columns.tolist() != ['No', 'Yes']:
   heart_attack_counts = heart_attack_counts[['No', 'Yes']]

heart_attack_counts.plot(kind='area', stacked=True, figsize=(10, 7),
color=['skyblue', 'salmon'])

plt.title('Number of Individuals Who Had a Heart Attack by General Health
Status',font='serif',fontsize='large',color='blue')
plt.xlabel('General Health Status',font='serif',fontsize='large',color='darkred')
plt.ylabel('Number of Individuals',font='serif',fontsize='large',color='darkred')
plt.legend(title='HadHeartAttack')
plt.tight_layout()
plt.show()
```

```
#%%
#3D
from mpl_toolkits.mplot3d import Axes3D

# Assume df is your DataFrame and you have three numerical columns of interest
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df['PhysicalHealthDays'], df['MentalHealthDays'], df['SleepHours'])
ax.set_xlabel('Physical Health Days',font='serif',fontsize='large',color='darkred')
ax.set_ylabel('Mental Health Days',font='serif',fontsize='large',color='darkred')
ax.set_zlabel('Sleep Hours',font='serif',fontsize='large',color='darkred')

plt.title('3D Scatter Plot',font='serif',fontsize='large',color='blue')
plt.show()

#%%
import numpy as np

x = df['BMI']
y = df['SleepHours']
z = df['HadHeartAttack'].apply(lambda x: 1 if x == 'Yes' else 0)

# Create grid coordinates for contour plot
x_range = np.linspace(x.min(), x.max(), 100)
y_range = np.linspace(y.min(), y.max(), 100)
x_grid, y_grid = np.meshgrid(x_range, y_range)

# Interpolate z values on grid coordinates
from scipy.interpolate import griddata
z_grid = griddata((x, y), z, (x_grid, y_grid), method='cubic')

# Create the contour plot
plt.figure(figsize=(10, 8))
cp = plt.contourf(x_grid, y_grid, z_grid, cmap='coolwarm')
plt.colorbar(cp)

plt.title('Contour Plot of Heart Attack Prevalence by BMI and Sleep
Hours',font='serif',fontsize='large',color='blue')
plt.xlabel('BMI',font='serif',fontsize='large',color='darkred')
plt.ylabel('Sleep Hours',font='serif',fontsize='large',color='darkred')
plt.tight_layout()
plt.show()
#%%
#hexbin
import pandas as pd
import matplotlib.pyplot as plt

# Hexbin plot for BMI and Sleep Hours
plt.figure(figsize=(10, 8))
plt.hexbin(df['BMI'], df['SleepHours'], gridsize=50, cmap='Blues', mincnt=1)
plt.colorbar(label='Count in bin')
plt.title('Hexbin Plot of BMI vs. Sleep
Hours',font='serif',fontsize='large',color='blue')
```

```
plt.xlabel('BMI',font='serif',fontsize='large',color='darkred')
plt.ylabel('Sleep Hours',font='serif',fontsize='large',color='darkred')
plt.show()
#%%
#Cluster map
# import seaborn as sns
# import matplotlib.pyplot as plt
# from sklearn.preprocessing import StandardScaler
# import numpy as np
# import pandas as pd
#
# # Assuming df is your preprocessed DataFrame with the necessary columns
#
# # Select a smaller sample of the data if the DataFrame is very large
# sample_frac = 0.01  # for example, 1% of the data
# df_sample = df.sample(frac=sample_frac, random_state=42)
#
# # Select the columns for clustering
# health_vars = ['PhysicalHealthDays', 'MentalHealthDays', 'SleepHours', 'BMI',
'HadHeartAttack', 'HadStroke', 'HadDiabetes']
#
# # Create a new DataFrame for these variables and convert to float32
# cluster_df = df_sample[health_vars].astype(np.float32)
#
# # Convert binary 'Yes'/'No' to numerical 1/0 for clustering
# binary_cols = ['HadHeartAttack', 'HadStroke', 'HadDiabetes']
# for col in binary_cols:
#     cluster_df[col] = cluster_df[col].map({'Yes': 1, 'No': 0})
#
# # Standardize the data
# scaler = StandardScaler()
# cluster_scaled = scaler.fit_transform(cluster_df)
#
# # Now create the cluster map on the sampled and scaled data
# # Use single linkage to save memory if needed
# sns.clustermap(cluster_scaled, method='single', cmap='vlag', figsize=(12, 12))
# plt.show()

#%%

#%%
#Setting style and fonts for plotting
import seaborn as sns
def set_size_style(width, height, style=None):
   plt.figure(figsize=(width, height))
   if style != None:
       sns.set_style(style)

# Function for customizating the plot
def customize_plot(plot, title:str, xlabel:str,  ylabel:str, title_font:int,
label_font:int):
   plot.set_title(title, font="serif",fontsize = title_font, color='blue')
   plot.set_xlabel(xlabel,font="serif", fontsize = label_font, color='darkred')
   plot.set_ylabel(ylabel, font="serif",fontsize = label_font, color='darkred')
#%%
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

gender_count = df['Sex'].value_counts()
print(gender_count)
# Create pie chart
plt.figure(figsize=(8, 8))
plt.title("Gender Distribution", fontsize=14,font="serif", color='blue')
patches, texts, autotexts = plt.pie(gender_count, labels=gender_count.index,
radius=1, autopct='%.2f%%', startangle=90,)
plt.legend(patches, labels=gender_count.index, title="Gender", loc="best",
fontsize='large')
plt.axis('equal')
plt.show()
#%%
from prettytable import PrettyTable

gender_counts = {"Male": 89924, "Female": 89869}
gender_table = PrettyTable()
gender_table.field_names = ["Sex", "Count"]
for sex, count in gender_counts.items():
   gender_table.add_row([sex, count])
print(gender_table)
#%%
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

set_size_style(10, 5, 'whitegrid')

# Create the countplot
ax = sns.countplot(data=df, x='AgeCategory', palette="husl")
customize_plot(ax, "Distribution of Age Category", "Age Category", "Individuals",
12, 10)
for i in ax.containers:
   ax.bar_label(i, label_type='edge')
plt.xticks(rotation=45)
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels, title="Age Categories", loc="upper right")

plt.tight_layout()
plt.show()
#%%
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

set_size_style(10, 5, 'whitegrid')

ax = sns.countplot(data=df, x='HadHeartAttack', hue='Sex', palette="husl")
customize_plot(ax, "Prevalence of Heart Attacks Among Different Genders", "Had
Heart Attack", "Individuals", 14, 12)
for i in ax.containers:
   ax.bar_label(i, label_type='edge')
plt.xticks(rotation=45)
```

```
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels, title="Sex", loc="upper right", fontsize='large')

plt.tight_layout()

plt.show()

#%%
risk_factors = ['PhysicalActivities', 'SmokerStatus', 'ECigaretteUsage',
'HadDiabetes']
set_size_style(18, 16, 'whitegrid')


for i, risk_factor in enumerate(risk_factors, 1):
   plt.subplot(4, 1, i)
   ax = sns.countplot(data=df, x='AgeCategory', hue=risk_factor)
   customize_plot(ax, f"{risk_factor} Among Different Age Groups", "Age Category",
"Individuals", 15, 12)

   for j in ax.containers:
       ax.bar_label(j)
plt.tight_layout()
plt.show()
#%%
#Now let's look at some other common risk factors among different genders and age
groups, such as Physical Health Days, Mental Health Days, and Sleep Hours.
other_risk_factor_among_genders =
df.groupby('Sex').agg({'PhysicalHealthDays':'mean', 'MentalHealthDays':'mean',
'SleepHours':'mean'})
print(round(other_risk_factor_among_genders,2))
#creating pretty table
from prettytable import PrettyTable
pt = PrettyTable()
pt.field_names = ["Sex", "PhysicalHealthDays", "MentalHealthDays", "SleepHours"]
pt.title=("Mean of other common risk factors among gender")
pt.add_row(["Female", "0.84", "1.4", "7.12"])
pt.add_row(["Male", "0.70", "0.96", "7.07"])
print(pt)
#%%
#Other risk factors among gender
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for i, risk_factor in enumerate(other_risk_factor_among_genders.columns, 1):
   plt.subplot(1, 3, i)
   plt.title(f"{risk_factor} among Genders", fontsize =
'large',font='serif',color='blue' )
   plt.pie(other_risk_factor_among_genders[risk_factor],
labels=other_risk_factor_among_genders.index, autopct='%1.1f%%', startangle=90)
fig.legend(axes[0].patches, labels=other_risk_factor_among_genders.index,
loc='upper right', title='Gender')
plt.tight_layout()
plt.show()
#%%
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
# Aggregate the data by 'Sex' to get the mean values of the risk factors
```

```
other_risk_factor_among_genders =
df.groupby('Sex').agg({'PhysicalHealthDays':'mean', 'MentalHealthDays':'mean',
'SleepHours':'mean'}).reset_index()

# Now, melt the DataFrame to have a suitable format for sns.scatterplot
melted_data = other_risk_factor_among_genders.melt(id_vars='Sex', var_name='Risk
Factor', value_name='Mean Value')

# Use seaborn to plot the scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=melted_data, x='Risk Factor', y='Mean Value', hue='Sex',
style='Sex', s=100)
plt.title('Comparison of Risk Factors Among Genders')
plt.legend(title='Gender')
plt.tight_layout()
plt.show()
#%%
other_risk_factor_among_agegroups =
df.groupby('AgeCategory').agg({'PhysicalHealthDays':'mean',
'MentalHealthDays':'mean', 'SleepHours':'mean'})
print(other_risk_factor_among_agegroups)
from prettytable import PrettyTable

pretty_table = PrettyTable()

pretty_table.field_names = ["Age Category", "Average Physical Health Days",
"Average Mental Health Days", "Average Sleep Hours"]

for index, row in other_risk_factor_among_agegroups.iterrows():
    pretty_table.add_row([index, round(row['PhysicalHealthDays'], 2),
round(row['MentalHealthDays'], 2), round(row['SleepHours'], 2)])
pretty_table.title=("   Mean of other common risk factors among Age Groups  ")
print(pretty_table)

#plot the lineplot
set_size_style(9, 5, 'whitegrid')
plt.ylabel('Mean Values')
ax =sns.lineplot(data=other_risk_factor_among_agegroups, x='AgeCategory',
y='PhysicalHealthDays', label='Physical Health Days')
ax =sns.lineplot(data=other_risk_factor_among_agegroups, x='AgeCategory',
y='MentalHealthDays', label='Mental Health Days')
ax= sns.lineplot(data=other_risk_factor_among_agegroups, x='AgeCategory',
y='SleepHours', label='Sleep Hours')
plt.legend()
customize_plot(ax,"Risk Factors among Age Groups","Age Groups","Average",12,10)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

#%%import pandas as pd
from prettytable import PrettyTable
# relationships between health behaviors (physical activity, sleep, alcohol
consumption) and the prevalence of heart disease.
behavioral_factors = ['SmokerStatus', 'ECigaretteUsage', 'AlcoholDrinkers',
'PhysicalActivities']
for factor in behavioral_factors:
    print(df[factor].value_counts(normalize=True)*100, end = '\n\n')
```

```
distribution = df[factor].value_counts(normalize=True) * 100

table = PrettyTable()
table.field_names = ["Behavioral Factor", "Value", "Percentage"]

# Loop through each behavioral factor and fill the table
for factor in behavioral_factors:
    # Calculate the percentage distribution
    distribution = df[factor].value_counts(normalize=True) * 100

    # Add each value and its percentage to the table
    for value, percentage in distribution.items():
        table.add_row([factor, value, f"{percentage:.2f}%"])

table.title=("each behavioral factor, the categories within that factor , and the
percentage of the dataset that falls into each category")
print(table)

#%%
# explore the relation between each behaviorial factor with heart disease status.

set_size_style(15,18,'ticks')
for i,col in enumerate(behavioral_factors,1):
    plt.subplot(2,2,i)
    ax = sns.countplot(data=df, x=col, hue='HadHeartAttack')
    customize_plot(ax,f"Relationship Between Heart Attack & {col}" , col,
'Individuals', 12,10)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45)
    for container in ax.containers:
        ax.bar_label(container)
plt.suptitle('Different diseases and Behavioral Factors correlation with heart
attack', size=25, fontweight='bold')
plt.tight_layout(w_pad=2, h_pad = 2)
plt.show()

#%%
df['SmokerStatus'].replace({'Current smoker - now smokes some days' : 'Current
smoker(Some days)',
                                    'Current smoker - now smokes every day' :
'Current smoker(Every day)'}, inplace=True)

df['SmokerStatus'].unique()
df['ECigaretteUsage'].replace({'Not at all (right now)' : 'Not at all',
                                       'Never used e-cigarettes in my entire life'
: 'Never',
                                       'Use them every day' : 'Everyday',
                                       'Use them some days' : 'Somedays'},
inplace=True)

df['ECigaretteUsage'].unique()
# how the presence of these factors is connected to the prevalence of heart
disease.
SmokerData = df[(df['SmokerStatus'] == 'Current smoker(Every day)') |
(df['SmokerStatus'] == 'Current smoker(Some days)')]
ECigaretteData = df[(df['ECigaretteUsage'] == 'Everyday') | (df['ECigaretteUsage']
== 'Somedays')]
```

```
AlcoholDrinkersData = df[(df['AlcoholDrinkers'] == 'Yes')]
PhysicallyActiveData = df[(df['PhysicalActivities'] == 'Yes')]

colors = {"Yes": "Yellow", "No": "cyan"}  # Example colors
set_size_style(12 ,8,'ticks')
# Setup the figure and axes
plt.figure(figsize=(12, 8))
plt.suptitle('Correlation of Smoking, E-Cigarette Use, Alcohol Consumption, and
Physical Activities with Heart Attack Risk', fontname='serif', color='blue')

# Subplot 1: Smokers
plt.subplot(1, 4, 1)
ax1 = sns.countplot(data=SmokerData, x='HadHeartAttack', palette=colors)
for container in ax1.containers:
   ax1.bar_label(container)
customize_plot(ax1, "Smokers Data", 'HadHeartAttack', 'Individuals', 12, 10)

# Subplot 2: E-Cigarette Users
plt.subplot(1, 4, 2)
ax2 = sns.countplot(data=ECigaretteData, x='HadHeartAttack', palette=colors)
for container in ax2.containers:
   ax2.bar_label(container)
customize_plot(ax2, "E-Cigarette Users Data", 'HadHeartAttack', 'Individuals', 12,
10)

# Subplot 3: Alcohol Drinkers
plt.subplot(1, 4, 3)
ax3 = sns.countplot(data=AlcoholDrinkersData, x='HadHeartAttack', palette=colors)
for container in ax3.containers:
   ax3.bar_label(container)
customize_plot(ax3, "Alcohol Drinkers Data", 'HadHeartAttack', 'Individuals', 12,
10)

# Subplot 4: Physically Active
plt.subplot(1, 4, 4)
ax4 = sns.countplot(data=PhysicallyActiveData, x='HadHeartAttack', palette=colors)
for container in ax4.containers:
   ax4.bar_label(container)
customize_plot(ax4, "Physically Active Data", 'HadHeartAttack', 'Individuals', 12,
10)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])  # Adjust the layout to make room for the
suptitle
plt.show()


#%%
Smoker_heart_attack_percentage =
(SmokerData['HadHeartAttack'].value_counts(normalize=True) * 100).to_dict()
ECigarette_heart_attack_percentage =
(ECigaretteData['HadHeartAttack'].value_counts(normalize=True) * 100).to_dict()
Alcohol_heart_attack_percentage =
(AlcoholDrinkersData['HadHeartAttack'].value_counts(normalize=True) *
100).to_dict()
PhysicalActivities_heart_attack_percentage =
(PhysicallyActiveData['HadHeartAttack'].value_counts(normalize=True) *
100).to_dict()
```

```
behavior_percentages = {
    'Smoking': Smoker_heart_attack_percentage.get('Yes'),
    'E-Cigarette Use': ECigarette_heart_attack_percentage.get('Yes'),
    'Alcohol Consumption': Alcohol_heart_attack_percentage.get('Yes'),
    'Physically Active': PhysicalActivities_heart_attack_percentage.get('Yes')
}
fig, ax = plt.subplots()
wedges, texts, autotexts = ax.pie(
    behavior_percentages.values(),
    labels=behavior_percentages.keys(),
    autopct='%1.1f%%',
    startangle=90,
    explode=[0, 0.05, 0.05, 0.05]
)

plt.title('Prevalence of Health Behaviors among Heart Patients',
weight='bold',font='serif',fontsize='large',color='blue')

plt.legend(wedges, behavior_percentages.keys(), title="Behaviors", loc="center
left", bbox_to_anchor=(1, 0, 0.5, 1))
plt.tight_layout()
plt.show()
#%%
import matplotlib.pyplot as plt
import seaborn as sns

heart_attack_sleep_hours = df[df['HadHeartAttack'] == 'Yes']['SleepHours']
heart_attack_sleep_hours.describe()
set_size_style(10,4,'ticks')
ax =sns.histplot(heart_attack_sleep_hours, bins=5, kde=True)
customize_plot(ax,"Distribution of Sleep Hours among Individuals with Heart
Diseases" ,'Sleep Hours' , 'Frequency', 12,10)
plt.tight_layout()
plt.show()
#%%
mental_health_cols = df[['GeneralHealth', 'HadHeartAttack',
'HadDepressiveDisorder', 'MentalHealthDays']]
print(mental_health_cols)

set_size_style(8,6,'darkgrid')
ax = sns.countplot(data=mental_health_cols, x='HadHeartAttack',
hue='HadDepressiveDisorder', palette='Set2')
customize_plot(ax,'Correlation of Depression and Heart Disease', 'HadHeartAttack',
'People count', 12, 10)

for container in ax.containers:
    ax.bar_label(container, fmt='%g persons')
plt.tight_layout()
plt.show()
#%%
cross_pivot = pd.crosstab(index=mental_health_cols['HadHeartAttack'],
columns=mental_health_cols['HadDepressiveDisorder'], normalize=True)
print(cross_pivot)
set_size_style(8,6)
```

```
ax = sns.heatmap(data=cross_pivot,cmap='coolwarm', linewidths=1, annot=True)
customize_plot(ax, 'Heatmap: Normalized Correlation between Depression and Heart
Disease', 'Had Depression', 'Had Heart Attack', 12,10)
plt.tight_layout()
plt.show()
#%%
import seaborn as sns
import matplotlib.pyplot as plt

# Select relevant columns for correlation matrix
columns_of_interest = ['PhysicalHealthDays', 'MentalHealthDays', 'SleepHours',
'HeightInMeters', 'WeightInKilograms', 'BMI']
correlation_data = df[columns_of_interest].corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_data, annot=True, fmt=".2f", cmap='flare', cbar=True)
plt.title('Pearson Correlation Matrix of Selected Health Metrics')
plt.tight_layout()
plt.show()
#%%
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
#BMI and Heart Attack Incidence:
#A suitable plot for this analysis is a regression plot with scatter
representation. This type of plot can show the distribution of BMI across
individuals who have and have not had a heart attack and fit a regression line to
see the trend.
# Adding a legend to the regression plot for BMI vs. Incidence of Heart Attacks

# Mapping the 'HadHeartAttack' to numerical values for regression plot
df['HadHeartAttackNumeric'] = df['HadHeartAttack'].map({'Yes': 1, 'No': 0})

# Graph 5: Regression plot for BMI vs. Incidence of Heart Attacks

#%%
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as mpatches
# Using lmplot to show the relationship between Sleep Hours and BMI, colored by
Heart Attack incidence
lm = sns.lmplot(x='SleepHours', y='BMI', hue='HadHeartAttack', data=df,
logistic=False,
               height=7, aspect=1.5, scatter_kws={'alpha':0.3})

# Enhance the lmplot with title and axis labels
plt.title('Sleep Hours and BMI vs. Incidence of Heart Attacks',
fontdict={'fontname': 'serif', 'color': 'blue', 'size': 16})
plt.xlabel('Sleep Hours', fontdict={'fontname': 'serif', 'color': 'darkred',
'size': 14})
plt.ylabel('BMI (kg/m²)', fontdict={'fontname': 'serif', 'color': 'darkred',
'size': 14})

# Moving the legend to an appropriate position
plt.legend(title='Had Heart Attack', loc='upper right', title_fontsize='13',
fontsize='12')
```

```
plt.tight_layout()
plt.show()

#%%
# #for the multivariate analysis, we need to ensure 'GeneralHealth' is encoded
numerically
# # to be included in pairplot. We'll map it to ordinal values based on perceived
health ranking.
# import matplotlib.pyplot as plt
# import seaborn as sns
# import matplotlib.patches as mpatches
# health_mapping = {'Poor': 0, 'Fair': 1, 'Good': 2, 'Very good': 3, 'Excellent':
4}
# df['GeneralHealthNumeric'] = df['GeneralHealth'].map(health_mapping)
#
# # Preparing the subset of data for the multivariate analysis
# multivariate_data = df[['GeneralHealthNumeric', 'BMI', 'SleepHours',
'HadHeartAttackNumeric']]
# sns.set(style="ticks", color_codes=True)
# g = sns.pairplot(multivariate_data, vars=['GeneralHealthNumeric', 'BMI',
'SleepHours'],
#                  hue="HadHeartAttackNumeric", palette="husl",
#                  diag_kind='kde', height=2.5, plot_kws={'alpha':0.5})
#
# # Customizing the titles and labels
# g.fig.suptitle('Comprehensive Health Review: Interplay of General Health, BMI,
Sleep, and Heart Attack Incidence',
#                fontdict={'fontname': 'serif', 'color': 'blue', 'size': 20},
y=1.05)
# for ax in g.axes.flatten():
#     ax.set_xlabel(ax.get_xlabel(), fontdict={'fontname': 'serif', 'color':
'darkred', 'size': 14})
#     ax.set_ylabel(ax.get_ylabel(), fontdict={'fontname': 'serif', 'color':
'darkred', 'size': 14})
#
# # Adjusting the legend
# handles = g._legend_data.values()
# labels = g._legend_data.keys()
# g._legend.remove()
# g.fig.legend(handles=handles, labels=labels, loc='upper right', title='Had Heart
Attack', fontsize=12, title_fontsize=13)
#
# plt.show()
#%%
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.violinplot(x='HadHeartAttack', y='BMI', data=df, palette='muted', split=True)
plt.title('Violin Plot of BMI by Heart Attack
Status',font='serif',fontsize='large',color='blue')
plt.xlabel('Had a Heart Attack',font='serif',fontsize='large',color='darkred')
plt.ylabel('BMI',font='serif',fontsize='large',color='darkred')
plt.tight_layout()
plt.show()

#%%
```

```
#Rug PLOT

import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as mpatches
# Define the BMI categorization function that returns labels
def categorize_bmi_label(bmi):
    if bmi < 18.5:
        return 'Underweight'
    elif bmi < 25:
        return 'Normal weight'
    elif bmi < 30:
        return 'Overweight'
    else:
        return 'Obesity'

# Sample a subset of the data if it's too large to avoid overplotting
sampled_data = df.sample(frac=0.05, random_state=1)  # Adjust frac as needed

# Apply the categorization to the sampled data
temp_bmi_labels = sampled_data['BMI'].apply(categorize_bmi_label)

# Create the rug plot with sampled data and adjusted parameters
plt.figure(figsize=(14, 4))  # Increased figure size
sns.rugplot(x=sampled_data['BMI'], hue=temp_bmi_labels, palette='Set2', height=0.1,
alpha=0.1)

# Customize the plot
plt.title('Rug Plot of BMI Categories with
Hue',font='serif',fontsize='large',color='blue')
plt.xlabel('BMI (kg/m²)',font='serif',fontsize='large',color='darkred')
plt.ylabel('Density',font='serif',fontsize='large',color='darkred')
colors = sns.color_palette('Set2', n_colors=4)
categories = ['Underweight', 'Normal weight', 'Overweight', 'Obesity']
handles = [mpatches.Patch(color=colors[i], label=categories[i]) for i in range(4)]
plt.legend(handles=handles, title='BMI Category', bbox_to_anchor=(1.01, 1),
loc='upper left')

plt.tight_layout()
plt.show()

# #%%
# print(df.shape)
#
#%%
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
# Downsampling the dataset to 50000 data points
sampled_data = df.sample(n=500, random_state=42)

# Create the swarm plot with the sampled data
plt.figure(figsize=(10, 6))
swarm_plot_sampled = sns.swarmplot(x='Sex', y='MentalHealthDays',
```

```
data=sampled_data)
swarm_plot_sampled.set_title('Swarm Plot of Mental Health Days by Sex
(Sampled)',fontsize='large',color='blue',font='serif')
swarm_plot_sampled.set_xlabel('Sex',fontsize='large',color='darkred',font='serif')
swarm_plot_sampled.set_ylabel('Number of Mental Health Days (last 30
days)',fontsize='large',color='darkred',font='serif')

plt.show()
#%%
# Create a strip plot for PhysicalHealthDays vs. GeneralHealth
plt.figure(figsize=(12, 8))
strip_plot = sns.stripplot(x='GeneralHealth', y='PhysicalHealthDays',
data=sampled_data, order=['Excellent', 'Very good', 'Good', 'Fair', 'Poor'],
jitter=True)
strip_plot.set_title('Strip Plot of Physical Health Days by General
Health',fontsize='large',color='blue',font='serif')
strip_plot.set_xlabel('General
Health',fontsize='large',color='darkred',font='serif')
strip_plot.set_ylabel('Number of Physical Health Days (last 30
days)',fontsize='large',color='darkred',font='serif')

plt.show()


#
#%%

import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv("heart_no_nans_clean_no_outliers.csv")
# Choose the column for the distplot
column = 'PhysicalHealthDays'

# Create the distplot
plt.figure(figsize=(10, 6))
sns.distplot(df[column], kde=True, color='skyblue')
plt.xlabel(column,fontsize='large',color='darkred',font='serif')
plt.ylabel('Density',fontsize='large',color='darkred',font='serif')
plt.title('Distribution Plot of
{}'.format(column),fontsize='large',color='blue',font='serif')
plt.show()

#%%
#  #Clustermap
# import seaborn as sns
#
# # Create the cluster map
# plt.figure(figsize=(10, 8))
# sns.clustermap(df['PhysicalHealthDays'], cmap='viridis', figsize=(10, 8))
# plt.title('Cluster Map of
PhysicalHealthDays',fontsize='large',color='blue',font='serif')
# plt.show()
```

```
import dash
from dash import dcc, html, Input, Output , State
import plotly.express as px
import pandas as pd
import numpy as np
from dash.exceptions import PreventUpdate
from io import BytesIO
# Load your dataset
df = pd.read_csv('heart_no_nans_clean_no_outliers.csv')
external_stylesheets =['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash("My app",external_stylesheets = external_stylesheets)

# Container div to handle the centering
div_style = {
   'textAlign': 'center',  # Centers the content inside the div
   'width': '100%',  # Takes full width to allow central alignment inside any
parent
   'marginTop': '20px',  # Optional: Adds space above the image
   'marginBottom': '20px'  # Optional: Adds space below the image
}

# Image tag with updated styles
image_style = {
   'width': '15%',  # Sets the width of the image to 30% of the container# Makes
the image a block-level element
   'margin': 'auto',  # Centers the image horizontally within the container
   'verticalAlign': 'middle' , # Centers the image vertically
   'height': 'auto'  # Keeps the height proportional to the width to maintain
aspect ratio
}

# Define Tab 1 layout using dcc and html components
tab_1_layout = html.Div([
   html.H3('General Health Overview'),
   html.Div([
       html.Img(src='assets/general_health_overview.png', style=image_style)
   ], style=div_style),
   html.Div([
       dcc.Checklist(
           id='conditions-checklist',
           options=[{'label': x, 'value': x} for x in
df['GeneralHealth'].unique()],
           value=df['GeneralHealth'].unique().tolist(),
           inline=True
       ),
       dcc.Dropdown(
           id='race-ethnicity-dropdown',
           options=[{'label': race, 'value': race} for race in
df['RaceEthnicityCategory'].unique()],
           value=df['RaceEthnicityCategory'].unique()[0]
       ),
       dcc.RadioItems(
           id='gender-radioitems',
           options=[{'label': gender, 'value': gender} for gender in
```

```
df['Sex'].unique()],
            value=df['Sex'].unique()[0],
            inline=True
        ),

    ]),
    dcc.Loading(
        id="loading-general-health",
        children=[dcc.Graph(id='general-health-dist')],
        type="default"
    ),
    html.H4('General Health Insights'),
    dcc.Textarea(
        id='health-insights',
        value='Select filters to see insights',
        style={'width': '100%', 'height': 100},
    ),
      html.Button("Download Text", id="btn-download-text"),
      dcc.Download(id="download-text-file")
])


# Define Tab 2 layout
tab_2_layout  = html.Div([
    html.H3('Depression Analysis by Age Group and Ethnicity Based on Sleep Hours'),
    html.Div([
        html.Img(src='assets/Depression_among_ethnicity.png', style=image_style)
    ], style=div_style),
    html.Div([
        dcc.RangeSlider(
            id='sleep-hour-range-slider',
            min=int(df['SleepHours'].min()),
            max=int(df['SleepHours'].max()),
            step=0.1,
            marks={i: str(i) for i in range(int(df['SleepHours'].min()),
int(df['SleepHours'].max()) + 1)},
            value=[df['SleepHours'].min(), df['SleepHours'].max()],
        ),
    ]),
    dcc.Graph(id='depression-analysis-graph'),
])



#tab4
# Define Tab 4 layout
tab_4_layout= html.Div([
    html.H3('Health Analysis Based on BMI'),
    html.Div([
        html.Img(src='assets/BMI.png', style=image_style)
    ], style=div_style),
    html.Div([
        html.Label('Select BMI:'),
        dcc.Slider(
            id='bmi-slider',
            min=df['BMI'].min(),
```

```
            max=df['BMI'].max(),
            value=df['BMI'].min(),
            marks={str(i): f"{i}" for i in range(int(df['BMI'].min()),
int(df['BMI'].max())+1, 1)},  # Adjusted for granularity
            step=0.1,
            tooltip={"placement": "bottom", "always_visible": True}
        ),
    ], style={'padding': 20}),
    dcc.Graph(id='bmi-health-data-graph')
])



app.layout = html.Div([
    html.H1('Health and Lifestyle Dashboard'),
    dcc.Tabs(id='tabs', children=[
        dcc.Tab(label='General Health Overview', children=tab_1_layout),
        dcc.Tab(label='Depression Analysis by Age Group and Ethnicity Based on Sleep
Hour', children=tab_2_layout),
        # dcc.Tab(label='LifeStyle Factors',children=tab_3_layout),
        dcc.Tab(label=" BMI affecting Diabetes and Chances of
Stroke",children=tab_4_layout),
        # ... Add other tabs here
    ])
])


# Callback for updating the general health distribution graph
#callback tab_1
@app.callback(
    Output('general-health-dist', 'figure'),
    [
        Input('conditions-checklist', 'value'),
        Input('race-ethnicity-dropdown', 'value'),
        Input('gender-radioitems', 'value'),
    ]
)


def update_general_health_dist(selected_conditions, selected_race,
selected_gender):


    # Filter the DataFrame based on the selected inputs
    filtered_df = df[
        df['GeneralHealth'].isin(selected_conditions) &
        df['RaceEthnicityCategory'].isin([selected_race]) &
        df['Sex'].isin([selected_gender])
        ]

    # If the filtered DataFrame is not empty, create the pie chart
    if not filtered_df.empty:
        fig = px.pie(
            filtered_df,
            names='GeneralHealth',
            title='Distribution of General Health Status',
```

```
            hole=0.3
        )
        fig.update_traces(textinfo='percent+label')
        return fig
    else:
        return {
            'data': [],
            'layout': {
                'title': 'No data available for the selected criteria',
                'xaxis': {'visible': False},
                'yaxis': {'visible': False},
                'annotations': [{
                    'text': 'No data available',
                    'xref': 'paper',
                    'yref': 'paper',
                    'showarrow': False,
                    'font': {
                        'size': 28
                    }
                }]
            }
        }

@app.callback(
    Output("download-text-file", "data"),
    [Input("btn-download-text", "n_clicks")],
    [State("health-insights", "value")],  # Replace with the ID of your textarea
component
    prevent_initial_call=True
)
def download_text(n_clicks, text_value):
    if n_clicks is None:
        raise PreventUpdate

    # Convert the text to a file-like object
    return dict(content=text_value, filename="text_data.txt")




#callback_tab2
@app.callback(
    Output('depression-analysis-graph', 'figure'),
    [Input('sleep-hour-range-slider', 'value')]  # This input is triggered when the
slider value changes
)
def update_depression_analysis(sleep_hours_range):
    # Filter the DataFrame based on the selected sleep hours range
    filtered_df = df[(df['SleepHours'] >= sleep_hours_range[0]) & (df['SleepHours']
<= sleep_hours_range[1])]

    # Group by AgeCategory and RaceEthnicityCategory and count depression cases
    depression_stats = filtered_df.groupby(['AgeCategory', 'RaceEthnicityCategory',
'HadDepressiveDisorder'])[
        'HadDepressiveDisorder'].count().unstack(fill_value=0)
```

```
    depression_stats['Total'] = depression_stats.sum(axis=1)
    depression_stats['Depression Rate'] = (depression_stats['Yes'] /
depression_stats['Total']) * 100  # Calculate percentage of depression cases

    # Plot the results
    fig = px.bar(
        depression_stats.reset_index(),
        x='AgeCategory',
        y='Depression Rate',
        color='RaceEthnicityCategory',
        barmode='group',
        title='Percentage of Depression by Age Group and Ethnicity for Selected
Sleep Hours'
    )
    return fig


#callback_tab3


#callback tab4
# Callback to update Graph based on BMI slider value
@app.callback(
    Output('bmi-health-data-graph', 'figure'),
    [Input('bmi-slider', 'value')]
)
def update_bmi_figure(selected_bmi):
    # Filter the DataFrame for the selected BMI
    filtered_df = df[df['BMI'] == selected_bmi]

    # Creating a new column 'Condition' that combines diabetes and stroke status
    filtered_df['Condition'] = 'Diabetes: ' + filtered_df['HadDiabetes'] + ',
Stroke: ' + filtered_df['HadStroke']

    # Prepare data for the bar chart
    counts = filtered_df.groupby('Condition').size().reset_index(name='Count')
    fig = px.bar(counts, x='Condition', y='Count', color='Condition',
                 title=f"Distribution of Diabetes and Stroke Conditions at BMI
{selected_bmi}",
                 labels={'Condition': 'Diabetes and Stroke Conditions'})

    return fig




# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```