Part I: Research Question

A1. Research Question

The data set I chose is medical_raw_data. The research question I will be addressing is which medical conditions (e.g., diabetes, stroke, anxiety) are most common among the population. My research question is highly relevant to healthcare organizations and public health agencies seeking to improve patient outcomes and optimize resource allocation. Understanding the prevalence of specific medical conditions allows these organizations to tailor their services and interventions effectively. For instance, if diabetes is a prevalent condition, healthcare providers can focus on diabetes management programs, preventive education, and community outreach initiatives.

Additionally, identifying trends in medical conditions helps forecast the demand for medical services and guide strategic planning. This information can also inform policymakers in developing targeted health campaigns, allocating funding, and addressing public health priorities to enhance overall community well-being. This question addresses a critical organizational need for data-driven healthcare management and policy formulation decision-making.

A2. Variables and Their Data Type

The dataset comprises 52 columns and 10,000 records, each representing a distinct patient. Each column variable—such as blood pressure, age, diabetes, and others—relates to an individual patient. The following is a description of each column variable:

- CaseOrder- Quantitative data, non-null, data type: int64, example row 1, row 2 respectively: 1, 2
  This column functions as an index that keeps the data in order.

- Customer_id- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: C412403, Z919181
  This column functions as a unique ID for each customer.

- Interaction- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: 8cd49b13-f45a-4b47-a2bd-173ffa932c2f, d2450b70-0337-4406-bdbb-bc1037f1734c

- UID- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: 3a83ddb66e2ae73798bdf1d705dc0932, 176354c5eef714957d486009feabf195

- City- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Eva, Marianna
  This column functions as the city location from which the patient is located.
- State- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: AL, FL
  This column functions as the state location from which the patient is located.

- County- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Morgan, Jackson
  This column functions as the county location from which the patient is located.

- Zip- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: 35621, 32446
  This column functions as the zip location from which the patient is located..

- Lat- Quantitative data, non-null, data type: float64, example row 1, row 2 respectively: 34.34960, 30.84513
  This column contains the GPS location of which the patient is from.

- Lng- Quantitative data, non-null, data type: float64t, example row 1, row 2 respectively: -86.72508, -85.22907
  This column contains the GPS location of which the patient is from.

- Population- Quantitative data, non-null, data type: int64, example row 1, row 2 respectively: 2951, 11303
  This column contains the population within one mile of the patient's residence.

- Area- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Suburban, Urban
  This column contains information on whether the area is urban, rural, or suburban.

- Timezone- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: America/Chicago, America/Chicago
  This column contains information on the time zone in which the patient lives.

- Job- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Psychologist/ sport and exercise, Community development worker
  This column provides information on the patient's job or the primary insurance holder's job.

- Children- Quantitative data, non-null, data type: object, example row 1, row 2 respectively: 1, 3
  This column contains the number of children that live in the same household as the patient.

- Age- Quantitative data, non-null, data type: float64, example row 1, row 2 respectively: 53, 51
  This column contains the patient's age.

- Education- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: College: Less than 1 year, Some College: 1 or More Years, No Degree
  This column contains information on the highest education received by the patient.

- Employment- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Full Time, Full Time
  This column contains the employment status of the patient.

- Income- Quantitative data, non-null, data type: float64, example row 1, row 2 respectively: 86575.93, 46805.99
  This column contains information on the patient's income or, in some cases, the income of the primary insurance holder.

- Marital- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Divorced, Married
  This column contains the marital status of the patient or primary insurance holder.

- Gender- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Male, Female
  This column contains information on the patient's gender: male, female, or nonbinary.

- ReAdmis- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: No, No
  This column contains information on whether the patient was readmitted to the hospital within a month of release.

- VitD_levels- Quantitative data, non-null, data type: float 64, example row 1, row 2 respectively: 17.80233049, 18.99463952
  This column contains information on the vitamin D levels of the patient measured in ng/mL.

- Doc_visits- Quantitative data, non-null, data type: int64, example row 1, row 2 respectively: 6, 4
  This column contains information on how often the doctor checked on a patient during the initial stay at the hospital.

- Full_meals_eaten- Quantitative data, non-null, data type: int64, example row 1, row 2 respectively: 0, 2
  This column contains information on how many meals the patient ate during their stay at the hospital, with partial meals counting as 0.

- VitD_supp- Quantitative data, non-null, data type: int64, example row 1, row 2 respectively: 0, 1
  This column contains information on how often vitamin D is given to the patient.

- Soft_drink- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: NA, No
  This column contains information on whether or not a patient drinks 3 sodas or more in a day.

- Initial_admin- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Emergency Admission, Emergency Admission
  This column contains information on the initial admission of the patient: emergency admission, elective admission, or observation.

- HighBlood- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Yes, Yes
  This column contains information on whether or not the patient has high blood pressure.

- Stroke- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: No, No
  This column contains information on whether or not the patient has had a stroke.

- Complication_risk- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Medium, High
  This column contains information on the patient's level of complication risk: low, medium, or high.

- Overweight- Qualitative data, non-null, data type: float64, example row 1, row 2 respectively: 0, 1
  This column contains information on whether or not the patient is overweight.

- Arthritis- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Yes, No
  This column contains information on whether or not the patient has arthritis.

- Diabetes- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Yes, No
  This column provides information on whether or not the patient has diabetes.

- Hyperlipidemia- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: No, No
  This column provides information on whether or not the patient has hyperlipidemia.

- BackPain- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Yes, No
  This column provides information on whether or not the patient has back pain.

- Anxiety- Qualitative data, non-null, data type: float64, example row 1, row 2 respectively: 1, NA
  This column provides information on whether or not the patient has anxiety.

- Allergic_rhinitis- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Yes, No
  This column provides information on whether or not the patient has allergic rhinitis.

- Reflux_esophagitis- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: No, Yes
  This column provides information on whether or not the patient has reflux esophagitis.

- Asthma- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Yes, No
  This column provides information on whether or not the patient has asthma.

- Services- Qualitative data, non-null, data type: object, example row 1, row 2 respectively: Blood Work, Intravenous
  This column provides information on what service the patient received: blood work, intravenous, CT Scan, MRI, etc.

- Initial_days- Quantitative data, non-null, data type: float64, example row 1, row 2 respectively: 10.58576971, 15.12956221
  This column provides information on how long the patient stayed on their first visit.

- TotalCharge- Quantitative data, non-null, data type: float64, example row 1, row 2 respectively: 3191.048774, 4212.905346
  This column provides information on the daily charges for the patient, and the total charge is divided by the number of days the patient is hospitalized, which gives an average value. It does not include any specialized treatments.

- Additional_charges- Quantitative data, non-null, data type: float64, example row 1, row 2 respectively: 17939.40342, 17612.99812
  This column provides information on the average amount charged for anything miscellaneous done to the patient.

  The following variables and items1-items8, are responses from a survey in which the patients rated on a scale from 1 to 8, where 1 is most important and 8 is least important.

- Item1- Timely admission, Qualitative data, non-null, data type: int64, example row 1, row 2 respectively: 3, 3

- Item2- Timely treatment, Qualitative data, non-null, data type: int64, example row 1, row 2 respectively: 3, 4

- Item3- Timely visits, Qualitative data, non-null, data type: int64, example row 1, row 2 respectively: 2, 3

- Item4- Reliability, Qualitative data, non-null, data type: int64, example row 1, row 2 respectively: 2, 4

- Item5- Options, Qualitative data, non-null, data type: int64, example row 1, row 2 respectively: 4, 4

- Item6- Hours of treatment, Qualitative data, non-null, data type: int64 example row 1, row 2 respectively: 3, 4

- Item7- Courteous staff, Qualitative data, non-null, data type: int64, example row 1, row 2 respectively: 3, 3

- Item8- Evidence of active listening from the doctor, Qualitative data, non-null, data type: int64, example row 1, row 2 respectively: 4, 3

Part II: Data-Cleaning Plan

B.1 Proposed Plan to Assess Data Quality

To assess the medical_raw_data.csv dataset, I will implement a structured exploratory data analysis (EDA) plan using pandas. After loading the dataset, I will systematically evaluate each column by checking its data type and number of entries, listing unique values, counting occurrences of each value, and identifying missing data. I will also compute summary statistics for each column to examine ranges, central tendencies, and variations, which will help identify anomalies or outliers.

This assessment will involve looping through all columns and applying these checks, providing detailed insights into data quality and structure. By identifying missing values, we can decide on corrective actions, such as imputation or removal of columns. Overall, this initial analysis will guide further data-cleaning steps and inform potential visualizations or analyses to explore relationships within the dataset.

B.2 Justification of Approach

The chosen approach for assessing the data quality is well-suited to the dataset's characteristics, which include numerical and categorical information across 10,000

entries related to medical, demographic, and behavioral aspects. The assessment techniques provide a comprehensive evaluation of various dimensions of data quality, such as completeness (through checks for missing values), accuracy (by verifying data types and consistency), and reliability (through duplicate detection and outlier identification). Each technique addresses specific issues that may arise in the dataset, ensuring a thorough and structured evaluation process. By progressing through each step logically, I can build a robust understanding of the dataset's quality, leading to more reliable and accurate insights in subsequent analyses.

B.3 Justification for Selected Programming Language and Libraries

I selected Python as the programming language for this data quality assessment because it is versatile, easy to use, and has strong community support, thus perfect for data analysis. The language has a vast choice of libraries specifically designed for data manipulation, analysis, and visualization, which causes an efficient approach to assessing data quality. The pandas' library is essential for data manipulation and exploration, with functions for cleaning and handling missing values. NumPy enhances numerical operations and supports mathematical computations, while Matplotlib and Seaborn are invaluable for creating visualizations that effectively represent data distributions, outliers, and relationships within the data. Although Scikit-learn is primarily a machine learning library, it offers valuable utilities for preprocessing and managing missing data, further supporting the cleaning process and the model sklearn.decomposition from Scikit-learn will be used for PCA. Overall, the combination of Python and these libraries enables a structured and effective methodology for assessing and enhancing the dataset's quality, ensuring that it is suitable for subsequent analysis.

Part III: Data Cleaning

C.1-C.2 Cleaning Findings and Justification of Mitigation Methods

The methods chosen to mitigate the data quality issues in the dataset are based on ensuring accuracy, consistency, efficiency, and the alignment of the data with its intended purpose. Here's a justification for each approach:

1. Correcting Data Types

Zip Codes (Convert to Strings): Zip codes should be used as identifiers. If kept as numerical values, their leading 0's will be dropped. For example, "00501" becomes "501". Thus, converting the data type to a string will correct the format.

Children, Age, Overweight, and Anxiety (Convert to Integers): These are naturally whole numbers, and storing them as floats introduces unnecessary decimal places, which could lead to confusion and wasted memory. Converting them to integers ensures clarity and prevents potential misinterpretation of the data.

Categorical Data (Convert Strings to Categories): Columns like Area, Education, Marital, Gender, and others represent categories with limited unique values. Storing them as string types is inefficient. Converting them to categorical types saves memory and speeds up operations like grouping, sorting, and filtering. This is critical for performance, especially in large datasets.

2. Addressing Inconsistent Data Values

Gender (Standardizing Values): The dataset includes the values Male, Female, and Prefer not to answer, but the data dictionary specifies Male, Female, and Non-Binary. It's essential to align with the data dictionary to maintain consistency across the dataset and ensure the data meets the expected definitions. This standardization also helps in creating accurate analyses and prevents data misinterpretation.

Timezones (Standardize to 9 US Time Zones): The dataset has 26 different time zones broken down by city, which adds unnecessary complexity. Standardizing to the nine official US time zones improves consistency and enables easier analysis by reducing fragmentation and complexity in the data. (Information of timezones from "United States Time Zone Map.")

3. Rounding Excessive Precision

VitD_levels, TotalCharge, Additional_charge (Reduce Decimal Precision): Storing values like VitD_levels to 6 decimal places provides excessive unnecessary precision for most practical analyses. Rounding these to 2 decimal places prevents false precision, simplifies calculations, and reduces memory usage without compromising the integrity of the data.

4. String Data Types

ReAdmis, Soft_drink, Diabetes, Asthma, and other health complications (Convert to binary integer): Many column entries have Yes/No values, which are strings.

Converting them to 1/0 respectively ensures more appropriate data handling and makes these fields easier to analyze, like in the case of PCA, which handles numeric values only.

5. Fixing Misleading Column Labels

Renaming Non-Descriptive Columns (e.g., Item1, Item2): Renaming these columns to more descriptive names will make the dataset easier to understand and interpret, especially for other users who might work with the dataset in the future. Descriptive labels improve clarity and reduce the risk of misinterpretation.

6. Replacing NaNs in Numerical Columns

Initial_days (Replace NaNs with 0): Since no value in the Initial_days column is less than 1, replacing NaN values with 0 is an optimal choice. This change reflects that if a value is missing, it likely means the patient was addmited for less than a day, thus, updating NaN values with 0's ensures that there is no ambiguity in this specific column.

7. Enforcing Consistent Naming Conventions

Standardizing Column Names (Pythonic Naming): Inconsistent naming conventions (e.g., using both uppercase and lowercase letters or inconsistent underscores) can make data harder to work with. Adopting a consistent, Pythonic convention (e.g., lowercase with underscores) makes manipulating the dataset programmatically easier, reducing errors and improving code readability.

These methods were chosen to improve data integrity, efficiency, and consistency. Correcting data types ensures the data is represented accurately (e.g., booleans for true/false values). Standardizing values and labels improves clarity and makes the data more accessible to work with, both manually and programmatically. Addressing excessive precision, non-descriptive labels, and inconsistent naming conventions enhances usability and reduces ambiguity. These methods ensure the dataset is clean, reliable, and ready for analysis.

C3. Summarry of Outcome from the Implementation of each Data-cleaning Step.

The data-cleaning process resulted in a well-prepared and consistent dataset for further analysis. Zip codes were standardized to five-character strings, ensuring uniformity for geographic comparisons. Time zones were grouped into broader categories (e.g., US - Eastern, US - Central), simplifying regional analysis. Yes/No

columns were converted to binary integers (1/0), making them compatible with numerical operations. String columns were transformed into categorical data types, optimizing memory usage and enhancing the performance of aggregation and grouping tasks.

Missing values in the children were replaced with zeros and converted to integer types, ensuring consistency without compromising the dataset's integrity. Gender categories were aligned with the data dictionary by replacing "Prefer not to answer" with "Non-Binary," promoting clarity and consistency. The ReAdmis column was converted to a boolean data type, simplifying the interpretation of readmission status as true/false values. Records in the Age and Income columns with missing values were dropped. I can not assume anyone's age or income for the range is vast and replacing these missing values might skew the data and give improper insights.

Numerical columns, including Vitamin D levels, total charges, and additional charges, were rounded to two decimal places, improving readability while maintaining data relevance. NaN values in Initial_days were filled with zeros, and the column was converted to integers to ensure consistency. Rows with missing values in critical fields were removed to maintain data completeness. Finally, column names were standardized into Pythonic formats, enhancing code readability and usability.

Overall, the data-cleaning efforts improved data quality, optimized memory usage, and ensured consistency across the dataset, making it reliable and ready for advanced analytics, modeling, and visualization.

C4. Summary of Limitations of the Data-cleaning Process.

While effective in improving data quality, the data-cleaning process has several limitations. First, replacing missing values with zeros in columns like Children and Initial_days may introduce bias, as these zeros might not accurately reflect real-world scenarios. Similarly, dropping rows with missing values (e.g., in income and age) may lead to the loss of valuable information and reduce sample size, potentially affecting the generalizability of insights.

Grouping time zones into broader categories simplifies analysis but might obscure regional nuances, such as the unique handling of Daylight Saving Time in certain areas (e.g., Arizona and Puerto Rico). Converting Yes/No columns to binary values assumes no other nuanced responses could overlook complex patient behaviors or conditions. The replacement of "Prefer not to answer" with "Non-Binary" in the gender column aligns with the data dictionary. Still, it may not fully capture individuals' intended identities, introducing a potential misrepresentation.

Furthermore, rounding continuous variables like Vitamin D levels and charges to two decimal places while improving readability could lead to minor precision loss. Additionally, transforming string columns to categorical data may limit flexibility for future updates if new categories emerge. Lastly, removing rows with NaN values may skew the dataset by underrepresenting specific demographics or patient groups, potentially impacting analysis accuracy and predictive models. Overall, while the data-cleaning steps ensure consistency and usability, they involve trade-offs that could influence the depth and reliability of the insights derived from the dataset.

C5. How Limitations Affect the Research Question

The limitations outlined in the data-cleaning process could impact the analysis of which medical conditions are most common, potentially leading to skewed or incomplete conclusions that affect healthcare decision-making. For instance, dropping rows with missing income or age data may exclude economically disadvantaged groups or specific age demographics, creating a bias in the sample that could misrepresent the proper distribution of medical conditions. Also, converting Yes/No responses to binary values assumes no nuance, potentially masking complexities in patient behaviors. For example, individuals managing chronic conditions like diabetes or anxiety may engage with healthcare differently, which a simple binary metric might fail to capture. This lack of nuance could hinder identifying subtle trends or developing targeted interventions. Third, replacing “Prefer not to answer” with “Non-Binary” in the gender column may introduce misclassification, which could skew insights into gender-related health trends. This misrepresentation might result in healthcare organizations missing opportunities to develop inclusive programs for underrepresented groups.

Finally, transforming string columns into categorical data may restrict future analysis if new health conditions or demographic categories arise. Removing rows with missing values could also disproportionately exclude vulnerable patient groups, limiting the dataset’s ability to reflect the full scope of health conditions within the population. These limitations could ultimately affect healthcare organizations' ability to make data-driven decisions, forecast service demand accurately, and allocate resources effectively. Misrepresentations in the prevalence of certain conditions may lead to misplaced priorities, resulting in underfunding critical health initiatives or overlooking emerging health risks.

D1. Code to Mitigate Data

Attached to my submission is a script file of the code used to mitigate the data quality.

D2. Provide a Copy of the Cleaned Data Set as a CSV file.

Attached to my submission is a copy of the cleaned data set as a CSV file. Below is a screenshot of some of the cleaned CSV file:

E1: Total Number of Principal Components and Loading Matrix

1. Total Number of Principal Components:

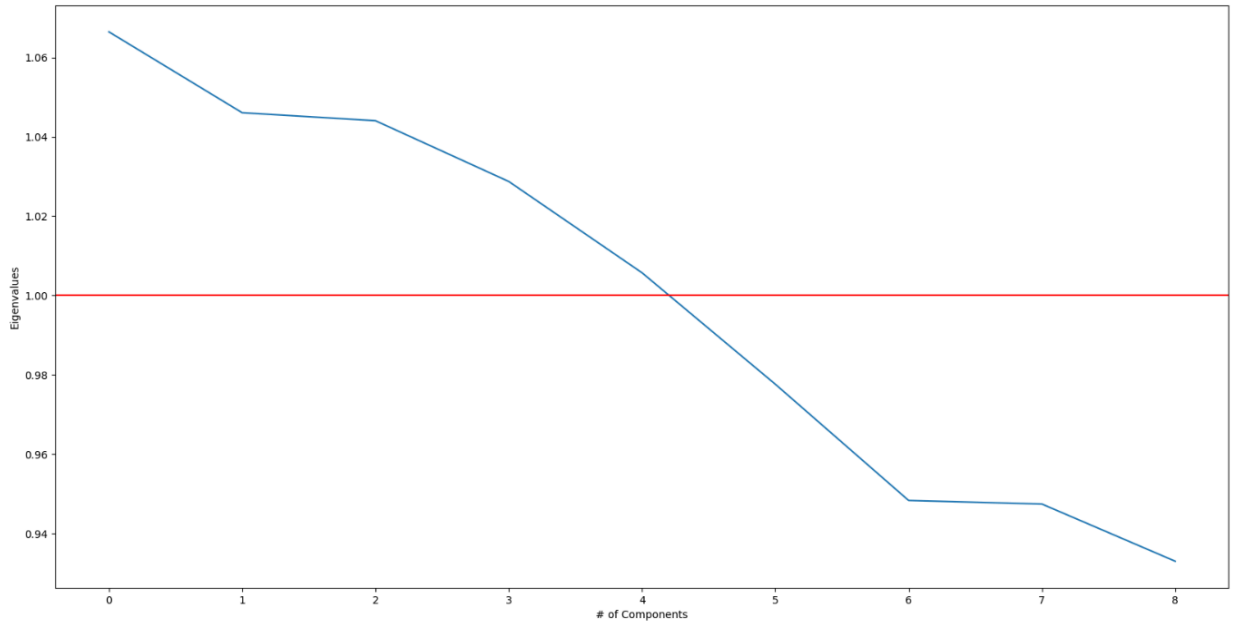
The variables I used for PCA are Population, Age, HighBlood, Stroke, Overweight, Arthritis, Diabetes, Reflux_esophagitis, and Asthma. Below is the output of the loading matrix.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Population	0.481113	-0.197517	0.062837	0.234676	0.293431	0.599459	0.425771	-0.208036	0.021962
Age	0.094407	0.208345	0.410866	-0.473997	-0.477606	-0.013362	0.458919	-0.192524	-0.279708
HighBlood	-0.112331	-0.400931	0.608331	0.089448	0.148409	0.085903	-0.195442	0.414065	-0.457730
Stroke	0.567053	-0.026116	0.192770	-0.333648	-0.151430	0.143247	-0.640715	-0.065957	0.266404
Overweight	-0.348864	-0.566943	0.000802	-0.257152	0.109461	-0.016628	-0.112480	-0.682507	-0.003982
Arthritis	-0.335955	0.171413	0.570487	0.327613	-0.121212	0.117331	0.020887	-0.122774	0.617250
Diabetes	-0.165140	0.632302	0.096931	-0.000645	0.387916	0.254372	-0.302179	-0.332420	-0.382790
Reflux_esophagitis	0.390046	0.041022	0.276634	0.218806	0.336798	-0.730438	0.088067	-0.258149	-0.020252
Asthma	-0.107103	0.062098	0.092925	-0.616115	0.591819	-0.016833	0.217354	0.292954	0.335920

E2: Justifying the Reduced Number of Principal Components with a Scree Plot

In PCA, it's common to reduce the dimensionality of the dataset by selecting only a few principal components that explain most of the variance in the data. For my research question, the most relevant data for PCA is age, population, and all the health problems recorded. The scree plot is a helpful tool where an "elbow" in the scree plot is noticeable, where the explained variance starts to level off, which indicates the following components contribute significantly less to explaining variance.

Scree Plot Screenshot:



E3: Organizational Benefits from PCA

Principal Component Analysis (PCA) offers several key data analysis and decision-making benefits. First, it enables data reduction by decreasing the dimensionality of large datasets while retaining most of the data's variability, leading to faster processing and analysis. Additionally, PCA contributes to noise reduction by filtering out less significant components, which can enhance predictive modeling performance and provide clearer insights. It also facilitates visualization by representing high-dimensional data in lower-dimensional spaces, such as 2D or 3D, helping to easily identify patterns, trends, and clusters. Moreover, PCA enhances interpretability by uncovering underlying structures and relationships within the data that may not be immediately apparent, thereby improving understanding and supporting informed decision-making. As a pre-processing step, PCA also supports further analysis by offering a cleaner and more interpretable feature set for machine learning techniques like clustering and regression. Lastly, it promotes resource optimization by reducing the number of features, enabling more efficient use of computational resources and lowering operational costs.

Part IV: Supporting Documents

F. Panopto Video

A recording on Panopto is attached with my submission under “Data Cleaning NUMx | D206 (Student Creators) [assignments].”

G. Web Sources With Code From Third Party

1. “Downcasting Behavior in replace Is Deprecated.” *GitHub*, 2023, github.com/pandas-dev/pandas/issues/57734.
2. “pandas.DataFrame.replace.” *pandas Documentation*, pandas, pandas.pydata.org/docs/reference/api/pandas.DataFrame.replace.html.

H. In text Citation

1. "United States Time Zone Map." *TimeTemperature*, 2023, www.timetemperature.com/tzus/time_zone.shtml.