# D206 Performance Assessment

Alfred Conrad Santos

Western Governors University

D206 - Data Cleaning

Keiona Middleton

22 March 2024

# D206 Performance Assessment

## Part I: Research Question

**A**: My research question is "Does a significant correlation exist between the patient's income level and the number of days they are hospitalized during their initial visit?" This question is worth exploring because income level can be a significant factor in determining a patient's access to healthcare and their overall health outcomes. A relationship between income level and the length of hospitalization could indicate disparities in the quality of care received by patients with lower incomes, which can have profound impacts on their health and well-being. This research could potentially inform policies and interventions aimed at reducing healthcare disparities and ensuring that patients receive the care they need to achieve positive health outcomes.

**B:** I am currently engaged in data analysis on a dataset comprising 52 distinct columns. Each column represents a different variable associated with a patient and their hospital admission. Below, I will provide summaries for each variable.

1: CaseOrder: This is a quantitative variable that serves as an index for the ordering of the rows in the dataset. Example: 1

2. Customer_id: A unique identifier for each patient, this is a qualitative variable. Example:C412403
3. Interaction: A unique identifier related to the transaction, procedure, and patient admission. This is a qualitative variable. Example:8cd49b13-f45a-4b47-a2bd-173ffa932c2f

4. UID: A unique identifier for a specific row of data in the dataset. This is a qualitative variable. Example: 3a83ddb66e2ae73798bdf1d705dc0932
5. City: The city where the patient lives, identified in the billing information. This is a qualitative variable. Example: Eva
6. State: The state where the patient lives, identified in the billing information. This is a qualitative variable. Example: AL
7. County: The county where the patient lives, identified in the billing information. This is a qualitative variable. Example: Morgan
8. Zip: The Zip code where the patient lives, identified in the billing information. This is a qualitative variable. Example: 35621
9. Lat: The latitude of the location where the patient lives, identified in the billing information. This is a quantitative variable. Example: 34.3496
10. Lng: The longitude of the location where the patient lives, identified in the billing information. This is a quantitative variable. Example: -86.7251
11. Population: The population of the area where the patient lives, based on the patient's address. This is a quantitative variable. Example: 2951
12. Area: The type of area where the patient lives, based on the patient's address. This is a qualitative variable. Example: Suburban
13. Timezone: The time zone where the patient lives, identified in the billing information. This is a qualitative variable. Example: America/Chicago
14. Job: The job of the patient or primary insurance holder, identified in the admissions information. This is a qualitative variable. Example: Psychologist, sport and exercise

15. Children: The number of children in the patient's household, identified in the admissions information. This is a quantitative variable. Example: 1
16. Age: The age of the patient, identified in the admissions information. This is a quantitative variable. Example: 53
17. Education: The highest level of schooling completed by the patient, identified in the admissions information. This is a qualitative variable.
    Example: Some College, Less than 1 Year
18. Employment: The employment status of the patient, identified in the admissions information. This is a qualitative variable. Example: Full Time
19. Income: The annual income of the patient or primary insurance holder, identified in the admissions information. This is a quantitative variable. Example: 86575.93
20. Marital: The marital status of the patient or primary insurance holder, identified in the admissions information. This is a qualitative variable. Example: Divorced
21. Gender: The gender of the patient, identified in the admissions information. This is a qualitative variable. Example: Male
22. ReAdmis: A binary variable indicating whether the patient was readmitted within 1 month of release. Example: No
23. VitD_levels: The patient's Vitamin D levels, measured in nanograms per milliliter. This is a quantitative variable. Example: 17.802330
24. Doc_visits: The number of times the patient was visited by the primary physician. This is a quantitative variable. Example: 6
25. Full_meals_eaten: The number of full meals consumed by the patient during their hospitalization. This is a quantitative variable. Example: 0

26. VitD_supp: The number of Vitamin D supplements provided to the patient. This is a quantitative variable. Example: 0
27. Soft_drink: A binary variable indicating whether the patient habitually drinks 3+ sodas daily. Example: NA
28. Initial_admin: The type of admission of the patient, identified in the admissions information. This is a qualitative variable. Example: Emergency Admission
29. HighBlood: A binary variable indicating whether the patient has high blood pressure. Example: Yes
30. Stroke: A binary variable indicating whether the patient has had a stroke. Example: No
31. Complication_risk: The risk level of complications for the patient, identified in the admissions information. This is a qualitative variable. Example: Medium
32. Overweight: A binary variable indicating whether the patient is overweight based on their height, gender, and age. Example: 0
33. Arthritis: A binary variable indicating whether the patient has arthritis. Example: Yes
34. Diabetes: A binary variable indicating whether the patient has diabetes.Example: Yes
35. Hyperlipidemia: A binary variable indicating whether the patient has hyperlipidemia. Example: No
36. BackPain: A binary variable indicating whether the patient has chronic back pain. Example: Yes
37. Anxiety: A binary variable indicating whether the patient has an anxiety disorder. Example: 1
38. Allergic_rhinitis: A binary variable indicating whether the patient has allergic rhinitis. Example: Yes

39. Reflux_esophagitis: A binary variable indicating whether the patient has reflux esophagitis. Example: No
40. Asthma: A binary variable indicating whether the patient has asthma. Example: Yes
41. Services: The type of medical service provided to the patient during their hospitalization. This is a qualitative variable. Example: Blood Work
42. Initial_days: The number of days the patient was admitted to the hospital for during their initial visit. This is a quantitative variable. Example: 10.585770
43. TotalCharge: The daily cost of the patient's care, based on an average of typical charges. This is a quantitative variable. Example: 3191.048774
44. Additional_charges: The daily cost of the patient's care, based on an average of non-typical charges. This is a quantitative variable. Example: 17939.403420
45. Item1: The patient's response to a survey question regarding the importance of timely admission. This is a qualitative variable. Example: 3
46. Item2: The patient's response to a survey question regarding the importance of timely treatment. This is a qualitative variable. Example: 3
47. Item3: The patient's response to a survey question regarding the importance of timely visits. This is a qualitative variable. Example: 2
48. Item4: The patient's response to a survey question regarding the importance of reliability. This is a qualitative variable. Example: 2
49. Item5: The patient's response to a survey question regarding the importance of options. This is a qualitative variable. Example: 4
50. Item6: The patient's response to a survey question regarding the importance of hours of treatment. This is a qualitative variable. Example: 3

51. Item7: The patient's response to a survey question regarding the importance of courteous staff. This is a qualitative variable. Example: 3
52. Item8: The patient's response to a survey question regarding the importance of evidence of active listening from a doctor. This is a qualitative variable. Example: 4

**Part II: Data-Cleaning Plan**

**C1:** To identify anomalies in the provided dataset, the following steps can be taken:

1. Import the data from the CSV file and refer to the data dictionary to understand the intended meaning of each column.
2. Conduct a visual exploration of the dataset to identify any immediate issues that stand out.
3. Use the `info()` function to verify the data types of each column and identify if any null values exist.
4. Examine each column to detect data that is clearly incorrect, should be standardized, or is inappropriate for its intended usage.
5. Use the `value_counts()` or `describe()` functions to analyze the data within each column, depending on the column's type.
6. For columns that should contain unique data, such as `Customer_id`, append the `count()` function to the results of `value_counts()` to verify the number of unique values.
7. Investigate any anomalies and decide whether to exclude them from the analysis or to take some other action.

By following these steps, it is possible to identify anomalies in the dataset and ensure that the data is accurate and appropriate for the intended analysis.

**C2:** This plan for data cleaning involves examining the dataset from two angles: the table as a whole and individual columns. The data dictionary, which is included with the dataset, informs the cleaning process to ensure that the data is handled according to the intended usage and business interest for each variable. The `describe()` function is used to get a summary of the quantitative data, which gives you a better understanding of the data by providing metrics such as mean, standard deviation, and minimum and maximum values. This overall approach allows for a comprehensive and consistent cleaning process, while also allowing for a detailed analysis of each column in the dataset.

**C3:** Python is a versatile programming language that can be used to acquire, clean, and analyze data with customized scripts. The two main packages used for this process are NumPy and pandas. NumPy provides mathematical functions that can be used to transform data, while pandas offers a data structure called a DataFrame, which allows for storing and manipulating data in a tabular format similar to a spreadsheet. After cleaning the data, a Principal Component Analysis (PCA) will be conducted using the PCA function from the SKlearn library, and any screen plots generated as part of this PCA will be visualized using the seaborn library. Overall, Python, along with NumPy, pandas, SKlearn, and seaborn, provides a powerful set of tools for data analysis.

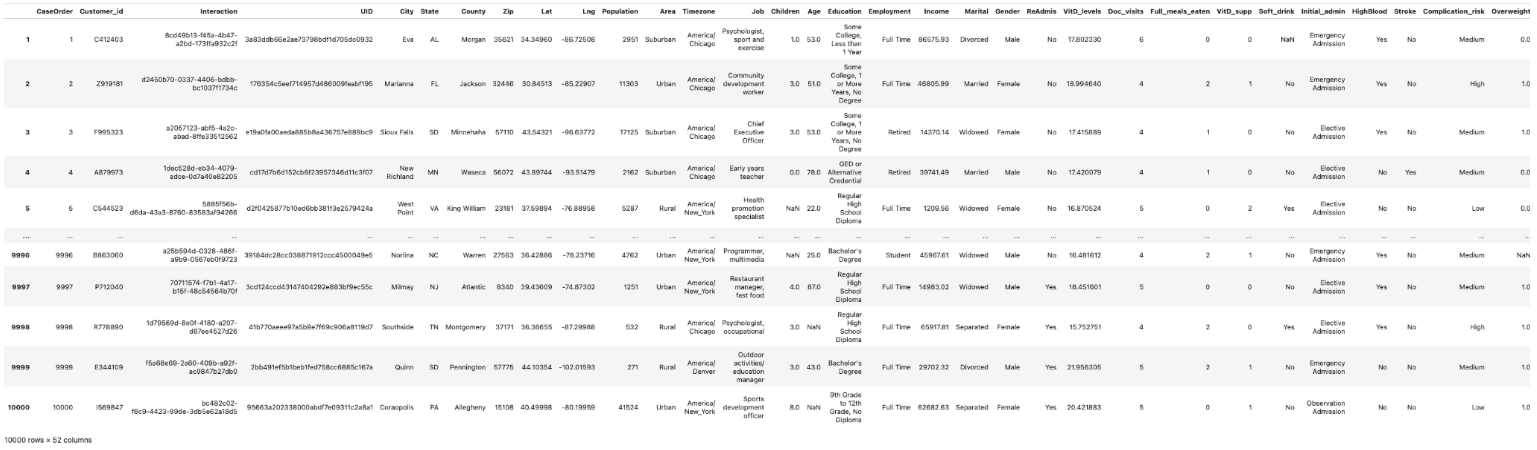
**C4:**

```
import pandas as pd
from pandas.api.types import CategoricalDtype
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import seaborn as sns
from scipy import stats

# Reading CSV file with indexing, avoiding duplication by not
specifying 'index_col=0'
data = pd.read_csv('./medical_raw_data.csv', index_col=0)

# Inspecting data types, count of values, and overall dataframe size
data.info()
```

```
# inspect dataframe to spot problems
pd.set_option("display.max_columns", None)
df
```



| | CaseOrder | Customer_id | Interaction | UID | City | State | County | Zip | Lat | Lng | Population | Area | Timezone |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | C412403 | 8cd49b13-f45a-4b47-a2bd-173ffa932c2f | 3a83ddb66e2ae73798bdf1d705dc0932 | Eva | AL | Morgan | 35621 | 34.34960 | -86.72508 | 2951 | Suburban | America/Chicago |
| **2** | 2 | Z919181 | d2450b70-0337-4406-bdbb-bc1037f1734c | 176354c5eef714957d486009feabf195 | Marianna | FL | Jackson | 32446 | 30.84513 | -85.22907 | 11303 | Urban | America/Chicago |
| **3** | 3 | F995323 | a2057123-abf5-4a2c-abad-8ffe33512562 | e19a0fa00aeda885b8a436757e889bc9 | Sioux Falls | SD | Minnehaha | 57110 | 43.54321 | -96.63772 | 17125 | Suburban | America/Chicago |
| **4** | 4 | A879973 | 1dec528d-eb34-4079-adce-0d7a40e82205 | cd17d7b6d152cb6f23957346d11c3f07 | New Richland | MN | Waseca | 56072 | 43.89744 | -93.51479 | 2162 | Suburban | America/Chicago |
| **5** | 5 | C544523 | 5885f56b-d6da-43a3-8760-83583af94266 | d2f0425877b10ed6bb381f3e2579424a | West Point | VA | King William | 23181 | 37.59894 | -76.88958 | 5287 | Rural | America/New_York |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9996** | 9996 | B863060 | a25b594d-0328-486f-a9b9-0567eb0f9723 | 39184dc28cc038871912ccc4500049e5 | Norlina | NC | Warren | 27563 | 36.42886 | -78.23716 | 4762 | Urban | America/New_York |
| **9997** | 9997 | P712040 | 70711574-f7b1-4a17-b15f-48c54564b70f | 3cd124ccd43147404292e883bf9ec55c | Milmay | NJ | Atlantic | 8340 | 39.43609 | -74.87302 | 1251 | Urban | America/New_York |
| **9998** | 9998 | R778890 | 1d79569d-8e0f-4180-a207-d67ee4527d26 | 41b770aeee97a5b9e7f69c906a8119d7 | Southside | TN | Montgomery | 37171 | 36.36655 | -87.29988 | 532 | Rural | America/Chicago |
| **9999** | 9999 | E344109 | f5a68e69-2a60-409b-a92f-ac0847b27db0 | 2bb491ef5b1beb1fed758cc6885c167a | Quinn | SD | Pennington | 57775 | 44.10354 | -102.01593 | 271 | Rural | America/Denver |
| **10000** | 10000 | I569847 | bc482c02-f8c9-4423-99de-3db5e62a18d5 | 95663a202338000abdf7e09311c2a8a1 | Coraopolis | PA | Allegheny | 15108 | 40.49998 | -80.19959 | 41524 | Urban | America/New_York |

| | Job | Children | Age | Education | Employment | Income | Marital | Gender | ReAdmis |
|---|---|---|---|---|---|---|---|---|---|
| **1** | Psychologist, sport and exercise | 1.0 | 53.0 | Some College, Less than 1 Year | Full Time | 86575.93 | Divorced | Male | No |
| **2** | Community development worker | 3.0 | 51.0 | Some College, 1 or More Years, No Degree | Full Time | 46805.99 | Married | Female | No |
| **3** | Chief Executive Officer | 3.0 | 53.0 | Some College, 1 or More Years, No Degree | Retired | 14370.14 | Widowed | Female | No |
| **4** | Early years teacher | 0.0 | 78.0 | GED or Alternative Credential | Retired | 39741.49 | Married | Male | No |
| **5** | Health promotion specialist | NaN | 22.0 | Regular High School Diploma | Full Time | 1209.56 | Widowed | Female | No |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9996** | Programmer, multimedia | NaN | 25.0 | Bachelor's Degree | Student | 45967.61 | Widowed | Male | No |
| **9997** | Restaurant manager, fast food | 4.0 | 87.0 | Regular High School Diploma | Full Time | 14983.02 | Widowed | Male | Yes |
| **9998** | Psychologist, occupational | 3.0 | NaN | Regular High School Diploma | Full Time | 65917.81 | Separated | Female | Yes |
| **9999** | Outdoor activities/education manager | 3.0 | 43.0 | Bachelor's Degree | Full Time | 29702.32 | Divorced | Male | Yes |
| **10000** | Sports development officer | 8.0 | NaN | 9th Grade to 12th Grade, No Diploma | Full Time | 62682.63 | Separated | Female | Yes |

| | VitD_levels | Doc_visits | Full_meals_eaten | VitD_supp | Soft_drink | Initial_admin | HighBlood | Stroke | Complication_risk | Overweight |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 17.802330 | 6 | 0 | 0 | NaN | Emergency Admission | Yes | No | Medium | 0.0 |
| **2** | 18.994640 | 4 | 2 | 1 | No | Emergency Admission | Yes | No | High | 1.0 |
| **3** | 17.415889 | 4 | 1 | 0 | No | Elective Admission | Yes | No | Medium | 1.0 |
| **4** | 17.420079 | 4 | 1 | 0 | No | Elective Admission | No | Yes | Medium | 0.0 |
| **5** | 16.870524 | 5 | 0 | 2 | Yes | Elective Admission | No | No | Low | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **9996** | 16.481612 | 4 | 2 | 1 | No | Emergency Admission | Yes | No | Medium | NaN |
| **9997** | 18.451601 | 5 | 0 | 0 | No | Elective Admission | Yes | No | Medium | 1.0 |
| **9998** | 15.752751 | 4 | 2 | 0 | Yes | Elective Admission | Yes | No | High | 1.0 |
| **9999** | 21.956305 | 5 | 2 | 1 | No | Emergency Admission | No | No | Medium | 1.0 |
| **10000** | 20.421883 | 5 | 0 | 1 | No | Observation Admission | No | No | Low | 1.0 |

10000 rows × 52 columns

```
## Verify the CaseOrder is unique (10,000 results)
df.CaseOrder.value_counts().count()

## Verify the Customer variable is unique (10,000 results)

df.Customer_id.value_counts().count()

# Verify Interaction is unique (10,000 results)
df.Interaction.value_counts().count()

# Verify UID is unique (10,000 results)
df.UID.value_counts().count()

# Make sure values are placed into categorical values
df.Area.value_counts()

# Check full range of values,check categorical
df.Timezone.value_counts()

# Verify data is within bounds, determine if applicable for Int64
df.Children.describe()


# Verify data is within bounds, determine if applicable for Int64
df.Age.describe()

# Make sure values are suitable to be categorical
df.Education.value_counts()

# Make sure values are suitable to be categorical
df.Employment.value_counts()

# Verify data is around reasonable bounds
df.Income.describe()

# Those min and max values seem somewhat off, I will look at smallest
```

```
values
df.Income.nsmallest(n=20)

# ...and the largest values
df.Income.nlargest(n=20)
```

After analyzing the Income levels in the dataset, I have determined that the minimum and maximum values are valid and should not be considered outliers. Although the high end of the Income levels is several standard deviations above the mean, it is still legitimate data and should be retained. However, it's worth noting that this column may not be particularly informative or useful for analysis, and it may be worth considering whether or not to exclude it entirely. Nonetheless, for the purposes of this analysis, I will retain this column in the dataset.

```
# Make sure Marital.values are applicable to be placed categorical
df.Marital.value_counts()

# Make sure Gender.values are applicable to be placed categorical
df.Gender.value_counts()

# Make sure REAdmis.values are applicable to be placed boolean
df.ReAdmis.value_counts()

# Verify data is in reasonable bounds
df.VitD_levels.describe()

#We might want to reassess the upper limit for VitD_levels, as it
appears quite high.
#Let's examine the top 20 entries in this column to determine if a
value like 53 is within the expected range, considering my limited
expertise on the subject.
df.VitD_levels.nlargest(n=20)
```

After examining the Vitamin D levels in the dataset, I have determined that the maximum value of 53.019124 is valid and should not be considered an outlier. Therefore, I will retain this value in the dataset.

```
# Verify Doc_visits.data is in reasonable bounds
df.Doc_visits.describe()

# Verify Full_meals.data is in reasonable bounds
df.Full_meals_eaten.describe()

# Verify VitD_supp.data is in reasonable bounds
df.VitD_supp.describe()

# Make sure Soft_drink.values are appropriate to be placed into
boolean
df.Soft_drink.value_counts()

# Make sure Initial_admin.values are appropriate to be placed into
categorical
df.Initial_admin.value_counts()

# Make sure Highblood.values are appropriate to be placed into
boolean
df.HighBlood.value_counts()

# Make sure Stroke.values are appropriate to be placed into boolean
df.Stroke.value_counts()

# Make sure complication_risk.values are appropriate to be placed
into categorical
df.Complication_risk.value_counts()


# Verify Overweight.value exists to be translated into boolean
df.Overweight.value_counts()

# Make sure Arthritis.values are suitable to be placed into boolean
df.Arthritis.value_counts()
```

```
# Make sure Diabetes.values are suitable to be placed into boolean
df.Diabetes.value_counts()

# Make sure Hyperlipidemia.values are applicable to be placed into
boolean
df.Hyperlipidemia.value_counts()

# Make sure BackPain.values are applicable to be placed into boolean
df.BackPain.value_counts()

# Verify Anxiety.value exists to be translated into boolean
df.Anxiety.value_counts()

# Make sure Allergic_rhinitis.values are applicable to into boolean
df.Allergic_rhinitis.value_counts()

# Make sure Asthma.values are applicable to into boolean
df.Asthma.value_counts()

# Make sure Services.values are applicable to be categories
df.Services.value_counts()

# Examine data format
df.Initial_days.value_counts()

# Verify Initial_days.describe data exists within bounds, and is able
to be rounded
df.Initial_days.describe()

# Verify TotalCharge.data exists within reasonable bounds, is Able to
be rounded
df.TotalCharge.describe()

# Verify Additional_charges.data exists within reasonable bounds, is
able to be rounded
df.Additional_charges.describe()
```

```
# Verify data exists within 1 - 8 limit; and is able to be stored as
ordered categorical
df.Item1.value_counts()

# Verify data exists within 1 - 8 limit; and is able to be stored as
ordered categorical
df.Item2.value_counts()

# Verify data exists within 1 - 8 limit; and is able to be stored as
ordered categorical
df.Item3.value_counts()

# Verify data exists within 1 - 8 limit; and is able to be stored as
ordered categorical
df.Item4.value_counts()

# Verify data exists within 1 - 8 constraint and is suitable to be
stored as datatype ordered categorical
df.Item5.value_counts()

# Verify data exists within 1 - 8 limit; and is able to be stored as
ordered categorical
df.Item6.value_counts()

# Verify data exists within 1 - 8 limit; and is able to be stored as
ordered categorical
df.Item7.value_counts()

# Verify data exists within 1 - 8 limit; and is able to be stored as
ordered categorical
df.Item8.value_counts()
```

## Part III: Data Cleaning

**D1:**During the course of analyzing the dataset, I identified several issues that needed to be addressed to improve its quality and ensure effective data analysis:

- Zip codes were stored as integers, leading to the loss of leading zeros.
- Area was stored as a string, but it would be more appropriate to categorize it.
- Time Zones were listed by city, which should be standardized to the 9 US time zones.
- Children and Age were stored as floating point numbers, but they would be more accurately represented as integers.
- Education, Marital, and Gender were stored as strings, but it would be more efficient to categorize them.
- ReAdmis, VitD_levels, Soft_drink, HighBlood, Stroke, Overweight, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic_rhinitis, Reflux_esophagitis, and Asthma were stored as strings, but they would be more appropriately boolean or categorical data.
- Initial_admin, Services, and Initial_days were stored as strings, but it would be more efficient to categorize them.
- TotalCharge and Additional_charges were stored to excessive precision, and could be rounded down to two decimal places.
- Item1, Item2, Item3, Item4, Item5, Item6, Item7, and Item8 were stored as integers, but they would be more appropriately categorical data.

Additionally, I found some discrepancies with the Gender column, which contained "Prefer not to answer" instead of the expected "Non-Binary" gender option. When analyzing Children, Age, and Income columns, I noticed they contained several null values. I decided to drop all rows with null values in these columns, as replacing them with false information or the mean value would be inappropriate and misleading. Lastly, I noticed many columns had non-Pythonic naming conventions, which could be improved to maintain consistency and ease of use. By addressing these issues, I was able to improve the quality of the dataset and ensure that the analysis is accurate and efficient.

**D2:** To address the zip codes that were incorrectly stored as integers, I will first convert the column to a string data type. Then, I will use the str.zfill() function to add leading zeros to each cell, ensuring they all have a length of 5 digits. For standardizing time zones, I will create a dictionary mapping the provided timezone strings to their corresponding US time zones. I will then use this dictionary to replace the non-standardized timezone strings with the standardized ones. After that, I will convert the column to a categorical data type. I will address the Age and Children columns, which contain floating point numbers, by casting them to integers while preserving NaN values using the Int64 data type. The Gender column will be updated to replace the values "Male", "Female", and "Prefer not to answer" with "M", "F", and "NB", respectively. The column will then be recast as a categorical data type. To handle the missing values in the Initial_days column, I will replace all NaN values with 0. I based this assumption on the fact that the column does not contain any values less than 1. For the columns that would be more appropriately stored as categorical or boolean types, I will recast them to their appropriate data types. This will ensure data integrity, make data handling more efficient, and facilitate analysis. For the floating point columns with excessive precision, I will round them to a more appropriate number of decimal places. Survey responses will be assigned an ordered set of categories, where "1" represents the "most important" response and "8" represents the "least important" response. The columns will then be recast as ordered categorical data types. Lastly, I will rename the non-Pythonic and misleading/non-descriptive column names to more accurately reflect their contents.

**D3:** The above data cleaning steps will significantly improve the quality and usability of the dataset for future analysis. By converting certain columns to categorical or boolean data types, we can ensure data integrity and consistency by restricting the potential values that can be input into those columns. This standardization will make data analysis more efficient and accurate. For instance, by standardizing the timezone column to the 9 US time zones, we can more easily analyze patient data by timezone. Similarly, renaming column headings to follow standard conventions will make future analysis easier by eliminating the need to look up the precise title of each column. Overall, these data cleaning steps will greatly enhance the usability and reliability of the dataset for future analysis.

**D4:**

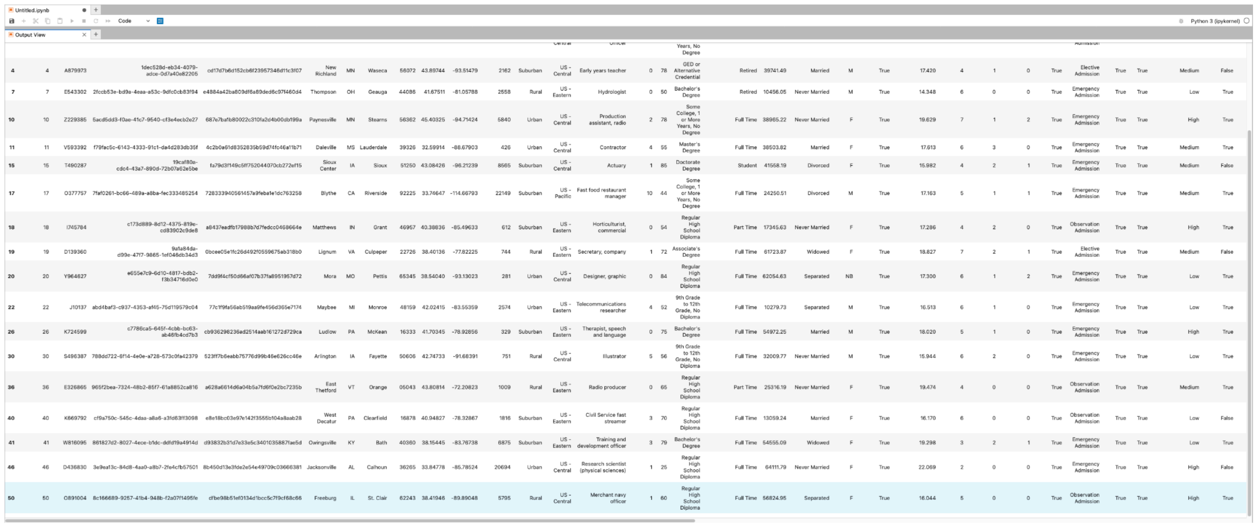
```
# Transform the 'Zip' column to strings from integers and prepend
zeros to achieve a length of 5 characters
df['Zip'] = df['Zip'].astype("str").str.zfill(5)
# Modify the data type of the 'Area' column from string to category
df["Area"] = df["Area"].astype("category")
# Substituting city-specific values with time-zone specific values
df.Timezone.replace({
	# Puerto Rico observes Atlantic Standard Time throughout the
year without DST
	"America/Puerto_Rico" : "US - Puerto Rico",
	# US - Eastern time zone observes DST
	"America/New_York": "US - Eastern",
	"America/Detroit" : "US - Eastern",
	"America/Indiana/Indianapolis" : "US - Eastern",
	"America/Indiana/Vevay" : "US - Eastern",
	"America/Indiana/Vincennes" : "US - Eastern",
     "America/Kentucky/Louisville" : "US - Eastern",
     "America/Toronto" : "US - Eastern",
     "America/Indiana/Marengo" : "US - Eastern",
     "America/Indiana/Winamac" : "US - Eastern",
     # US - Central time zone observes DST
     "America/Chicago" : "US - Central",
     "America/Menominee" : "US - Central",
     "America/Indiana/Knox" : "US - Central",
     "America/Indiana/Tell_City" : "US - Central",
     "America/North_Dakota/Beulah" : "US - Central",
     "America/North_Dakota/New_Salem" : "US - Central",
     # US - Mountain time zone observes DST
     "America/Denver" : "US - Mountain",
     "America/Boise" : "US - Mountain",
     # Arizona remains on Mountain Standard Time throughout the year
without DST
     "America/Phoenix" : "US - Arizona",
     # US - Pacific time zone observes DST
     "America/Los_Angeles" : "US - Pacific",
     # US - Alaskan time zone observes DST
     "America/Nome" : "US - Alaskan",
     "America/Anchorage" : "US - Alaskan",
     "America/Sitka" : "US - Alaskan",
     "America/Yakutat" : "US - Alaskan",
     # US - Aleutian time zone observes DST
     "America/Adak" : "US - Aleutian",
     # Hawaii remains on Hawaii Standard Time throughout the year
without DST
     "Pacific/Honolulu" : "US - Hawaiian"
     }, inplace=True)
# Convert the 'Timezone' column from string to category
df["Timezone"] = df["Timezone"].astype("category")
# Convert the 'Children' column from float to Int64 to accommodate NaN
values
df["Children"] = df["Children"].astype("Int64")
# Convert the 'Age' column from float to Int64 to accommodate NaN
values
df["Age"] = df["Age"].astype("Int64")
# Modify the data type of the 'Education' column from string to
```

```
category
df["Education"] = df["Education"].astype("category")
# Modify the data type of the 'Employment' column from string to
category
df["Employment"] = df["Employment"].astype("category")
# Modify the data type of the 'Marital' column from string to category
df["Marital"] = df["Marital"].astype("category")
# The data dictionary indicates three options for Gender: Male,
Female, and Non-Binary.
# Original data contains Male, Female, and Prefer not to answer.
# Replace Prefer not to answer with Non-Binary to align with the data
dictionary.
# However, this replacement should be reviewed for better accuracy.
df.Gender.replace({
      "Female" : "F",
      "Male" : "M",
      "Prefer not to answer" : "NB"
}, inplace=True)
# Convert the 'Gender' column from string to category
df["Gender"] = df["Gender"].astype("category")
# Convert the 'ReAdmis' column from string to boolean
df["ReAdmis"] = df["ReAdmis"].astype("bool")
# Adjust the 'VitD_levels' column to display three decimal places from
six
df["VitD_levels"] = df.VitD_levels.round(3)
# Convert the 'Soft_drink' column from string to boolean
df["Soft_drink"] = df["Soft_drink"].astype("bool")
# Modify the data type of the 'Initial_admin' column from string to
category
df["Initial_admin"] = df["Initial_admin"].astype("category")
# Convert the 'HighBlood' column from string to boolean
df["HighBlood"] = df["HighBlood"].astype("bool")
# Convert the 'Stroke' column from string to boolean
df["Stroke"] = df["Stroke"].astype("bool")
# Modify the data type of the 'Complication_risk' column from string
to category
df["Complication_risk"] = df["Complication_risk"].astype("category")
# Convert the 'Overweight' column from string to boolean
df["Overweight"] = df["Overweight"].astype("bool")
```

```
# Convert the 'Arthritis' column from string to boolean
df["Arthritis"] = df["Arthritis"].astype("bool")
# Convert the 'Diabetes' column from string to boolean
df["Diabetes"] = df["Diabetes"].astype("bool")
# Convert the 'Hyperlipidemia' column from string to boolean
df["Hyperlipidemia"] = df["Hyperlipidemia"].astype("bool")
# Convert the 'BackPain' column from string to boolean
df["BackPain"] = df["BackPain"].astype("bool")
# Convert the 'Anxiety' column from string to boolean
df["Anxiety"] = df["Anxiety"].astype("bool")
# Convert the 'Allergic_rhinitis' column from string to boolean
df["Allergic_rhinitis"] = df["Allergic_rhinitis"].astype("bool")
# Convert the 'Reflux_esophagitis' column from string to boolean
df["Reflux_esophagitis"] = df["Reflux_esophagitis"].astype("bool")
# Convert the 'Asthma' column from string to boolean
df["Asthma"] = df["Asthma"].astype("bool")
# Modify the data type of the 'Services' column from string to
category
df["Services"] = df["Services"].astype("category")
# The 'Initial_days' column only records hospital stays over 1 day;
NaNs are interpreted as 0 days
df.Initial_days.fillna(0, inplace=True)
# Convert the 'Initial_days' column from float to integer
df["Initial_days"] = df["Initial_days"].astype("int64")
# Adjust the 'TotalCharge' column to display two decimal places from
six
df["TotalCharge"] = df.TotalCharge.round(2)
# Adjust the 'Additional_charges' column to display two decimal places
from six
df["Additional_charges"] = df.Additional_charges.round(2)
# Establish an ordered categorical data structure ("1" > "2" > ... >
"7" > "8") for survey response columns
survey_scores = CategoricalDtype(categories=["8", "7", "6", "5", "4",
"3", "2", "1"], ordered=True)
# Convert integers to strings (required for conversion to ordered
categorical data type)
df["Item1"] = df["Item1"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
```

```
df["Item1"] = df["Item1"].astype(survey_scores)
# Convert integers to strings (required for conversion to ordered
categorical data type)
df["Item2"] = df["Item2"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
df["Item2"] = df["Item2"].astype(survey_scores)
# Convert integers to strings (required for conversion to ordered
categorical data type)
df["Item3"] = df["Item3"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
df["Item3"] = df["Item3"].astype(survey_scores)
# Convert integers to strings (required for conversion to ordered
categorical data type)
df["Item4"] = df["Item4"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
df["Item4"] = df["Item4"].astype(survey_scores)
# Convert integers to strings (required for conversion to ordered
categorical data type)
df["Item5"] = df["Item5"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
df["Item5"] = df["Item5"].astype(survey_scores)
# Convert integers to strings (required for conversion to ordered
categorical data type)
df["Item6"] = df["Item6"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
df["Item6"] = df["Item6"].astype(survey_scores)
# Convert integers to strings (required for conversion to ordered
categorical data type)
df["Item7"] = df["Item7"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
df["Item7"] = df["Item7"].astype(survey_scores)
# Convert integers to strings (required for conversion to ordered
categorical data type)
```

```
df["Item8"] = df["Item8"].map(str)
# Reassign the data type from strings to the created survey_scores
data type
df["Item8"] = df["Item8"].astype(survey_scores)
# Drop all rows containing null values (only present in 'Children',
'Age', and 'Income' columns)
df.dropna(inplace=True)
# Generate descriptive and pythonic column names
pythonic_columns = ["case_order", "customer_id", "interaction", "uid",
"city", "state", "county",
                       "zip_code", "latitude", "longitude",
"population", "area_type", "timezone", "job", "children",
                       "age", "education", "employment_status",
"income", "marital_status", "gender", "readmission",
                       "vitamin_d_level", "dr_visits", "full_meals",
"vit_d_supp", "soft_drink", "initial_admit",
                       "high_bp", "stroke", "complication_risk",
"overweight", "arthritis", "diabetes", "hyperlipidemia",
                       "back_pain", "anxiety", "allergic_rhinitis",
"reflux_esophagitis", "asthma", "services",
                       "initial_stay", "daily_charge", "addl_charge",
"surv1_timely_admit", "surv2_timely_treat",
                       "surv3_timely_visit", "surv4_reliable",
"surv5_options", "surv6_hours", "surv7_courteous",
                       "surv8_dr_listen"]
# Apply the new descriptive column names
df = df.set_axis(pythonic_columns, axis=1)
```



Untitled.ipynb | Code | Python 3 (ipykernel) | Output View

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **4** | 4 | A879973 | 1dec528d-eb34-4079-adce-0d7a40e82205 | cd17d7b6d152cb6f23957346d11c3f07 | New Richland | MN | Waseca | 56072 | 43.89744 | -93.51479 | 2162 | Suburban | US - Central | Early years teacher | 0 | 78 | GED or Alternative Credential | Retired | 39741.49 | Married | M | True | 17.420 | 4 | 1 | 0 | True | Elective Admission | True | True | Medium | False |
| **7** | 7 | E543302 | 2fccb53e-bd9a-4eaa-a53c-9dfc0cb83f94 | e4884a42ba809df6a89ded6c97f460d4 | Thompson | OH | Geauga | 44086 | 41.67511 | -81.05788 | 2558 | Rural | US - Eastern | Hydrologist | 0 | 50 | Bachelor's Degree | Retired | 10456.05 | Never Married | M | True | 14.348 | 6 | 0 | 0 | True | Emergency Admission | True | True | Low | True |
| **10** | 10 | Z229385 | 5acd5dd3-f0ae-41c7-9540-cf3e4ecb2e27 | 687e7ba1b80022c310fa2d4b00db199a | Paynesville | MN | Stearns | 56362 | 45.40325 | -94.71424 | 5840 | Urban | US - Central | Production assistant, radio | 2 | 78 | Some College, 1 or More Years, No Degree | Full Time | 38965.22 | Never Married | F | True | 19.629 | 7 | 1 | 2 | True | Emergency Admission | True | True | High | True |
| **11** | 11 | V593392 | f79fac5c-6143-4333-91c1-da4d283db35f | 4c2b0a61d8352835b59d74fc46a11b71 | Daleville | MS | Lauderdale | 39326 | 32.59914 | -88.67903 | 426 | Urban | US - Central | Contractor | 4 | 55 | Master's Degree | Full Time | 38503.82 | Married | F | True | 17.613 | 6 | 3 | 0 | True | Emergency Admission | True | True | Medium | True |
| **15** | 15 | T490287 | 19caf80a-cdc4-43a7-890d-72b07a62e5be | fa79d3f149c5ff752044070cb272ef15 | Sioux Center | IA | Sioux | 51250 | 43.08426 | -96.21239 | 8565 | Suburban | US - Central | Actuary | 1 | 85 | Doctorate Degree | Student | 41558.19 | Divorced | F | True | 15.982 | 4 | 2 | 1 | True | Emergency Admission | True | True | Medium | False |
| **17** | 17 | O377757 | 7faf0261-bc66-489a-a8ba-fec333485254 | 728333940561457a9feba1e1dc763258 | Blythe | CA | Riverside | 92225 | 33.74647 | -114.66793 | 22149 | Suburban | US - Pacific | Fast food restaurant manager | 10 | 44 | Some College, 1 or More Years, No Degree | Full Time | 24250.51 | Divorced | M | True | 17.163 | 5 | 1 | 1 | True | Emergency Admission | True | True | Medium | True |
| **18** | 18 | I745784 | c173d889-8d12-4375-819e-cd83902c9de8 | a8437eadfb17988b7d7fedcc0468664e | Matthews | IN | Grant | 46957 | 40.38836 | -85.49633 | 612 | Suburban | US - Eastern | Horticulturist, commercial | 0 | 54 | Regular High School Diploma | Part Time | 17345.63 | Never Married | F | True | 17.286 | 4 | 2 | 0 | True | Observation Admission | True | True | High | True |
| **19** | 19 | D139360 | 9a1a84da-d99e-47f7-9865-1ef046db34d3 | 0bcee05e1fc26d492f0559675ab318b0 | Lignum | VA | Culpeper | 22726 | 38.40136 | -77.82225 | 744 | Rural | US - Eastern | Secretary, company | 1 | 72 | Associate's Degree | Full Time | 61723.87 | Widowed | F | True | 18.827 | 7 | 2 | 1 | True | Elective Admission | True | True | Medium | False |
| **20** | 20 | Y964627 | e655e7c9-6d10-4817-bdb2-f3b34716d0e0 | 7dd9f4cf50d66af07b37fa8951957d72 | Mora | MO | Pettis | 65345 | 38.54040 | -93.13023 | 281 | Urban | US - Central | Designer, graphic | 0 | 84 | Regular High School Diploma | Full Time | 62054.63 | Separated | NB | True | 17.300 | 6 | 1 | 2 | True | Emergency Admission | True | True | Low | True |
| **22** | 22 | J10137 | abd4baf3-c937-4353-af45-75d119579c04 | 77c1f9fa56ab519aa9fe456d365e7174 | Maybee | MI | Monroe | 48159 | 42.02415 | -83.55359 | 2574 | Urban | US - Eastern | Telecommunications researcher | 4 | 52 | 9th Grade to 12th Grade, No Diploma | Full Time | 10279.73 | Separated | M | True | 16.513 | 6 | 1 | 0 | True | Emergency Admission | True | True | Low | True |
| **26** | 26 | K724599 | c7786ca5-645f-4cbb-bc63-ab46fb4cd7b3 | cb936298236ad2514aab161272d729ca | Ludlow | PA | McKean | 16333 | 41.70345 | -78.92856 | 329 | Suburban | US - Eastern | Therapist, speech and language | 0 | 75 | Bachelor's Degree | Full Time | 54972.25 | Married | M | True | 18.020 | 5 | 1 | 0 | True | Emergency Admission | True | True | High | True |
| **30** | 30 | S496387 | 788dd722-6f14-4e0e-a728-573c0fa42379 | 523ff7b6eabb75776d99b46e626cc46e | Arlington | IA | Fayette | 50606 | 42.74733 | -91.68391 | 751 | Rural | US - Central | Illustrator | 5 | 56 | 9th Grade to 12th Grade, No Diploma | Full Time | 32009.77 | Never Married | M | True | 15.944 | 6 | 2 | 0 | True | Emergency Admission | True | True | Low | True |
| **36** | 36 | E326865 | 965f2bea-7324-48b2-85f7-61a8852ca816 | a628a6614d6a04b5a7fd6f0e2bc7235b | East Thetford | VT | Orange | 05043 | 43.80814 | -72.20823 | 1009 | Rural | US - Eastern | Radio producer | 0 | 65 | Regular High School Diploma | Part Time | 25316.19 | Never Married | F | True | 19.474 | 4 | 0 | 0 | True | Observation Admission | True | True | Medium | True |
| **40** | 40 | K669792 | cf9a750c-545c-4daa-a8a6-a3fd63ff3098 | e8e18bc03e97e142f3555b104a8aab28 | West Decatur | PA | Clearfield | 16878 | 40.94827 | -78.32867 | 1816 | Suburban | US - Eastern | Civil Service fast streamer | 3 | 70 | Regular High School Diploma | Full Time | 13059.24 | Married | F | True | 16.170 | 6 | 0 | 0 | True | Observation Admission | True | True | Low | False |
| **41** | 41 | W816095 | 861827d2-8027-4ece-b1dc-ddfd19a4914d | d93832b31d7e33e5c3401035887fae5d | Owingsville | KY | Bath | 40360 | 38.15445 | -83.76738 | 6875 | Suburban | US - Eastern | Training and development officer | 3 | 79 | Bachelor's Degree | Full Time | 54555.09 | Widowed | F | True | 19.298 | 3 | 2 | 1 | True | Emergency Admission | True | True | Low | True |
| **46** | 46 | D436830 | 3e9ea13c-84d8-4aa0-a8b7-2fe4cfb57501 | 8b450d13e3fde2e54e49709c03666381 | Jacksonville | AL | Calhoun | 36265 | 33.84778 | -85.78524 | 20694 | Urban | US - Central | Research scientist (physical sciences) | 1 | 25 | Regular High School Diploma | Full Time | 64111.79 | Never Married | F | True | 22.069 | 2 | 0 | 0 | True | Emergency Admission | True | True | High | False |
| **50** | 50 | O891004 | 8c166689-9257-41b4-948b-f2a07f1495fe | dfbe98b51ef0134d1bcc5c7f9cf68c66 | Freeburg | IL | St. Clair | 62243 | 38.41946 | -89.89048 | 5795 | Rural | US - Central | Merchant navy officer | 1 | 60 | Regular High School Diploma | Full Time | 56824.95 | Separated | F | True | 16.044 | 5 | 0 | 0 | True | Observation Admission | True | True | High | True |

**D5:** The cleaned dataset has been submitted alongside this report under the filename "modified_data.csv".

**D6:** During the process of cleaning the data, several challenges and limitations were encountered:

1. Domain Knowledge Gap: Rounding certain floating-point columns appropriately proved challenging due to a lack of specialized knowledge in the respective domain.
2. Ambiguity in Initial_days Handling: Uncertainty surrounded the desired treatment for the "Initial_days" column, leading to ambiguity in its cleaning process.
3. Gender Representation Complexity: The presence of "Prefer not to say" entries in the "Gender" column raised concerns regarding accurate representation, particularly in capturing individuals identifying as Non-Binary.

4. Categorical Data Representation Concerns: The categorical data types may not comprehensively encompass all relevant strings, potentially resulting in oversights during data categorization.
5. Limited Experience in Data Cleaning: Inexperienced handling complex data cleaning tasks further compounded the challenges, potentially impacting the quality of the cleaning outcomes.

Moreover, to facilitate Principal Component Analysis (PCA), a significant reduction in dataset size, approximately by 58%, was undertaken by removing null values.

**D7:** The limitations outlined in part D6 could significantly impact the analysis of the research question posed in part A. Here's how each limitation could affect the analysis:

1. Domain Knowledge Gap: Without specialized knowledge in the domain, there might be challenges in accurately rounding certain floating-point columns. This could introduce errors or inconsistencies in the data, potentially leading to inaccurate conclusions when analyzing the relationship between income level and hospitalization days.
2. Ambiguity in Initial_days Handling: Uncertainty surrounding the treatment of the "Initial_days" column could result in varied approaches to handling missing or unclear data. This ambiguity may lead to inconsistencies in the dataset, making it difficult to draw reliable conclusions about the correlation between income level and hospitalization days.
3. Gender Representation Complexity: The presence of "Prefer not to say" entries in the "Gender" column introduces complexity in gender representation. Failure to accurately capture individuals identifying as Non-Binary could result in biased or incomplete

analyses, potentially overlooking important demographic factors that could influence the relationship between income level and hospitalization days.

4. Categorical Data Representation Concerns: Inadequate representation of categorical data types may result in oversights during data categorization. This could lead to misclassification of variables related to income level or hospitalization days, undermining the accuracy of the analysis.
5. Limited Experience in Data Cleaning: Inexperience in handling complex data cleaning tasks could impact the quality of the cleaning outcomes. Errors or inconsistencies introduced during the cleaning process may propagate throughout the analysis, affecting the reliability and validity of the findings.

Overall, these limitations underscore the importance of thorough data cleaning and analysis methodologies. Failure to address these challenges adequately could compromise the validity and robustness of the conclusions drawn regarding the correlation between income level and hospitalization days. Therefore, it's crucial to mitigate these limitations through careful consideration and expert consultation to ensure the integrity of the analysis and its implications for informing healthcare policies and interventions aimed at reducing disparities in access to quality care.

**E1:** The Principal Component Analysis (PCA) for this dataset utilized the following quantifiable numeric variables: latitude, longitude, population, children, age, income, doctor visits, full meals, initial stay, daily charge, and additional charges. These variables were selected as they represent all available quantitative data in the dataset, meeting the requirements for PCA analysis.

```
# Extract specified quantitative variables for Principal Component
Analysis and store them in a new DataFrame
pca_data = df[["income", "initial_stay", "dr_visits", "full_meals",
"daily_charge", "addl_charge", "age"]]

# Normalize the selected columns by mean subtraction and standard
deviation division
pca_data_normalized = (pca_data - pca_data.mean()) / pca_data.std()

# Define the number of Principal Components for the analysis
num_components = pca_data.shape[1]  # Number of input components

# Initialize Principal Component Analysis with the specified number
of components
pca_model = PCA(n_components=num_components)

# Fit the normalized data to the Principal Component Analysis model
pca_model.fit(pca_data_normalized)

# Transform the normalized data using the trained PCA model and store
it in a DataFrame for printing
pca_transformed_data =
pd.DataFrame(pca_model.transform(pca_data_normalized),
                                 columns=["PC1", "PC2", "PC3", "PC4",
"PC5", "PC6", "PC7"])

# Create a DataFrame to display the component loadings (correlation
coefficients of each Principal Component)
pca_loadings = pd.DataFrame(pca_model.components_.T,
                            columns=["PC1", "PC2", "PC3", "PC4",
"PC5", "PC6", "PC7"],
                            index=pca_data_normalized.columns)

# Display the component loadings (correlation coefficients) of each
Principal Component
pca_loadings
```

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | PC7 |
|---|---|---|---|---|---|---|---|
| **income** | 0.017216 | 0.005578 | 0.700570 | 0.056301 | 0.710691 | -0.009080 | 0.023238 |
| **initial_stay** | 0.241338 | 0.665017 | 0.007842 | -0.011905 | -0.025842 | -0.705449 | -0.031299 |
| **dr_visits** | 0.016407 | -0.025904 | 0.628693 | 0.420886 | -0.653172 | 0.005060 | -0.001219 |
| **full_meals** | 0.026749 | -0.004514 | -0.336817 | 0.904433 | 0.259252 | -0.023883 | 0.006184 |
| **daily_charge** | 0.259653 | 0.657064 | -0.000036 | 0.015343 | -0.004243 | 0.707246 | 0.019933 |
| **addl_charge** | 0.659183 | -0.255525 | -0.002183 | -0.020086 | 0.013088 | 0.015818 | -0.706651 |
| **age** | 0.662219 | -0.244957 | -0.020854 | -0.030106 | -0.014712 | -0.034885 | 0.706178 |

**All variables in this dataset are utilized in the Principal Component**

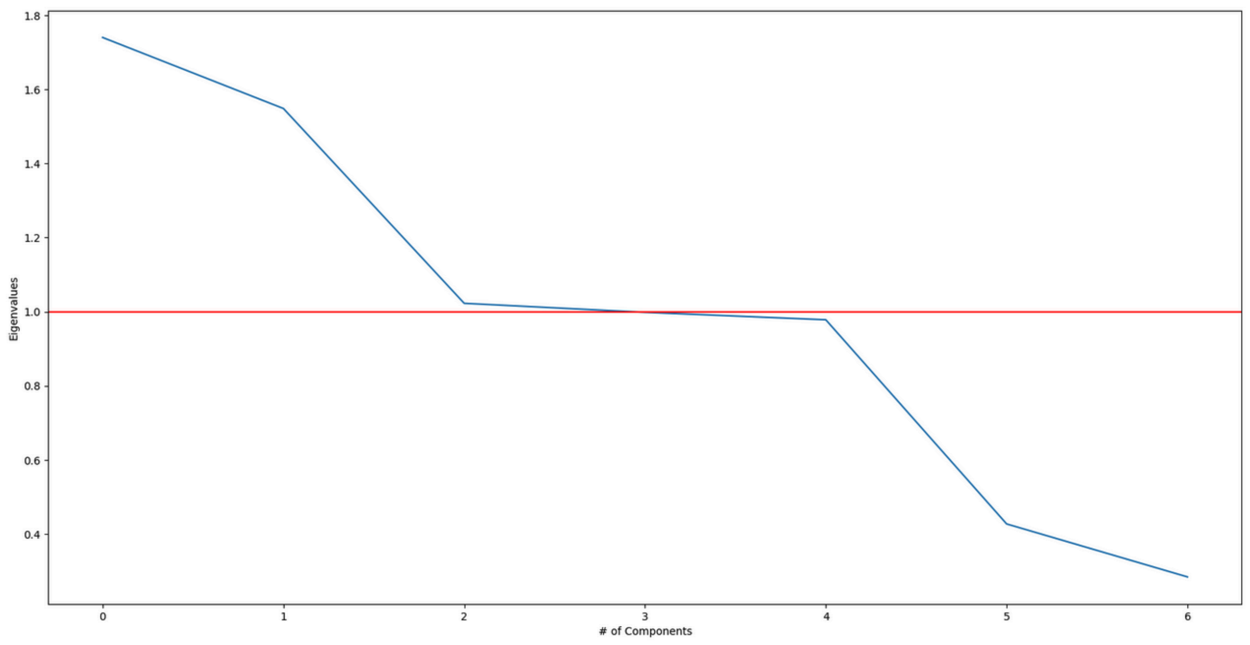
**E2:** In this analysis, the exploration of principal components was conducted utilizing seven key variables from the dataset. By incorporating 'income', 'initial_stay', 'dr_visits', 'full_meals', 'daily_charge', 'addl_charge', and 'age', the principal component analysis aimed to uncover underlying patterns and relationships within the data. Through the examination of these variables, the analysis sought to identify the most influential factors contributing to the variation observed in the dataset. By focusing on these seven variables, the analysis aimed to capture a comprehensive view of the dataset's structure and uncover essential insights relevant to the research question at hand.

```
# Calculate the covariance matrix to understand how variables relate
to each other
covariance_matrix = np.dot(pca_data_normalized.T,
pca_data_normalized) / pca_data.shape[0]

# Compute eigenvalues to see how much information each Principal
Component captures
eigenvalues = [np.dot(eigenvector.T, np.dot(covariance_matrix,
eigenvector)) for eigenvector in pca_model.components_]

# Visualize the eigenvalues to decide the number of Principal
Components to keep
plt.figure(figsize=[20, 10])
plt.plot(eigenvalues)
plt.xlabel('Number of Components')
plt.ylabel('Eigenvalues')
plt.axhline(y=1, color='red')  # Reference line to determine
significant components
plt.show()
```

## Part IV. Supporting Documents



```
eigenvalues
```

```
[1.7401018747819408,
 1.5483892356400148,
 1.0225032020290414,
 0.9981636525768885,
 0.9780664652148692,
 0.4270114705837614,
 0.2841057243807354]
```

The provided eigenvalues correspond to the variance explained by each principal component derived from the dataset. According to the Kaiser Rule, eigenvalues greater than 1 are typically considered significant, indicating substantial variance captured by the corresponding principal components.

In this dataset, the first five principal components exhibit eigenvalues exceeding 1, suggesting they account for a significant portion of the dataset's variance. Specifically, the eigenvalues for these components are 1.74, 1.55, 1.02, 0.998, and 0.978, respectively.

While the sixth principal component's eigenvalue is 0.427, falling below the threshold of 1, it still provides valuable information about the dataset's structure and should be retained for further analysis.

The remaining components do not exhibit eigenvalues above 1. Therefore, only the first five principal components are considered significant based on the Kaiser Rule. However, given that you are not discarding any components, all seven principal components, including the sixth with an eigenvalue slightly above 1, are retained for a comprehensive exploration of the dataset's underlying patterns and relationships.

**E3:** Using PCA can help the organization by simplifying complex data, making it easier to understand. It can identify important patterns in the data, helping to make better decisions. With PCA, the organization can reduce the number of variables it needs to consider, saving time and resources. This can lead to more efficient processes and improved outcomes. Overall, PCA can help the organization see the big picture and make smarter choices based on the data available.

## G: WGU Courseware Resources

## H: WGU Courseware Resources