D206 Performance Assessment

Jose Aldana

Dr. Keiona Middleton

Part 1:

A:

My research question is “What conditions can cause anxiety?”. Anxiety itself is experienced by most if not all the population, and it can cause a multitude of other ailments. Seeing how correlated these fields would be worthwhile, since it would show if more attention were needed towards treating anxiety so that it may not be as debilitating in the future for the patient.

B:

I will be using the Medical Data set, and these are the variables associated with that data.:

- CaseOrder
    - Data Type: Qualitative
    - Description:A unique number that is used to find each case.
    - Example: 1
- Customer_id
    - Data Type: Qualitative
    - Description: A unique number that is used to identify each individual customer
    - Example: Z919181
- Interaction
    - Data Type: Qualitative
    - Description: Another unique ID used to track each distinct interaction with the hospital.
    - Example: a2057123-abf5-4a2c-abad-8ffe33512562
- UID
    - Data Type: Qualitative
    - Description: Like Interaction, used as a unique identifier for each unique interaction.
    - Example: e8e016144bfbe14974752d834f530e26
- City

    - Data Type: Qualitative
    - Description: The city of residence for the patient.
    - Example: Thompson
- State
    - Data Type: Qualitative
    - Description: The abbreviated state of residence for the patient.
    - Example: NY
- County
    - Data Type: Qualitative
    - Description: The County of residence for the patient.
    - Example: Jackson
- Zip
    - Data Type: Qualitative
    - Description: The Zip code of for each patient.
    - Example: 72760
- Lat
    - Data Type: Quantitative
    - Description: The GPS Coordinates for the patient.
    - Example: 43.54321
- Lng
    - Data Type: Quantitative
    - Description: The GPS Coordinates for the patient.
    - Example: -96.63772
- Population
    - Data Type: Quantitative
    - Description: The population within one mile of the patient.
    - Example: 2951
- Area
    - Data Type: Qualitative
    - Description: The type of area the patient lives in
    - Example: Suburban
- Timezone
    - Data Type: Qualitative
    - Description: The time zone of residence for the patient.
    - Example: America/Detroit
- Job
    - Data Type: Qualitative

  - Description: The patients employment.
  - Example: Actuary
- Children
  - Data Type: Quantitative
  - Description: How many children in the patient's household.
  - Example: 1
- Age
  - Data Type: Quantitative
  - Description: The patients age.
  - Example: 50
- Education
  - Data Type: Qualitative
  - Description: Level of education completed by patient
  - Example: Some College
- Employment
  - Data Type: Qualitative
  - Description: Type of employment the patient has.
  - Example: Part Time
- Income
  - Data Type: Quantitative
  - Description: The yearly income for the patient or the primary insurance holder.
  - Example: 401.86
- Marital
  - Data Type: Qualitative
  - Description: Marital Status of the patient.
  - Example: Single
- Gender
  - Data Type: Qualitative
  - Description: The patient's self-identified gender.
  - Example: Male
- ReAdmis
  - Data Type: Qualitative
  - Description: Whether the patient was readmitted within a month of their hospital visit.
  - Example: Yes
- VitD_levels

    - Data Type: Quantitative
    - Description: The patients vitamin d levels.
    - Example: 20.425926
- Doc_visits
    - Data Type: Quantitative
    - Description: How many times the primary doctor visited the patient on their initial visit.
    - Example: 4
- Full_meals_eaten
    - Data Type: Quantitative
    - Description: The number of full meals the patient ate on their initial visit.
    - Example: 2
- VitD_supp
    - Data Type: Quantitative
    - Description: The number of times a patient received Vitamin D supplements
    - Example: 1
- Soft_drink
    - Data Type: Qualitative
    - Description: Whether the patient has three or more soft drinks a day
    - Example: Yes
- Initial_admin
    - Data Type: Qualitative
    - Description: How was the patient admitted as?
    - Example: Emergency Admission.
- HighBlood
    - Data Type: Qualitative
    - Description: Whether the patient has high blood pressure
    - Example: Yes
- Stroke
    - Data Type: Qualitative
    - Description: Has the patient had a stroke in the past?
    - Example: Yes
- Complication_risk
    - Data Type: Qualitative
    - Description: The level of complication that a patient is categorized as.
    - Example: High
- Overweight

  - Data Type: Qualitative
  - Description: Is the patient Overweight?
  - Example: Yes
- Arthritis
  - Data Type: Qualitative
  - Description: Does the patient have arthritis?
  - Example: No
- Diabetes
  - Data Type: Qualitative
  - Description: Does the patient have Diabetes?
  - Example: Yes
- Hyperlipidemia
  - Data Type: Qualitative
  - Description: Does the patient have Hyperlipidemia?
  - Example: Yes
- BackPain
  - Data Type: Qualitative
  - Description: Does the patient have chronic back pain?
  - Example: Yes
- Anxiety
  - Data Type: Qualitative
  - Description: Does the patient have chronic Anxiety?
  - Example: Yes
- Allergic_rhinitis
  - Data Type: Qualitative
  - Description: Does the patient have allergic rhinitis?
  - Example: Yes
- Reflux_esophagitis
  - Data Type: Qualitative
  - Description: Does the patient have reflux esophagitis?
  - Example: Yes
- Asthma
  - Data Type: Qualitative
  - Description: Does the patient have asthma?
  - Example: Yes
- Services
  - Data Type: Qualitative

- Description: The services that the patient received on their initial visit.
  - Example: Bloodwork
- Initial_days
  - Data Type: Quantitative
  - Description: The number of days the patient spent in the hospital during the
  - Example: 7.0750833
- TotalCharge
  - Data Type: Quantitative
  - Description: The average amount the patient was charged per day during their initial visit.
  - Example: 4214.90535
- Additional_charges
  - Data Type: Quantitative
  - Description: The average amount charged per day on any additional resources such as medication or sedation.
  - Example: 17505.1925
- Item1
  - Data Type: Qualitative
  - Description: The satisfaction the patient had regarding timely admission, on a scale of one to eight.
  - Example: 1
- Item2
  - Data Type: Qualitative
  - Description: The satisfaction the patient had regarding timely treatment, on a scale of one to eight.
  - Example: 2
- Item3
  - Data Type: Qualitative
  - Description: The satisfaction the patient had when it comes to how timely their visit was, on a scale of one to eight.
  - Example: 3
- Item4
  - Data Type: Qualitative
  - Description: How reliable the patient felt they felt with their physician, on a scale of one to eight.
  - Example: 4
- Item5

- Data Type: Qualitative
    - Description: How satisfied the patient was with the care options they were provided, on a scale of one to eight.
    - Example: 5
- Item6
    - Data Type: Qualitative
    - Description: How satisfied the patient was with their hours of treatment, on a scale of one to eight.
    - Example: 6
- Item7
    - Data Type: Qualitative
    - Description: How courteous the staff was to the patient, on a scale of one to eight.
    - Example: 7
- Item8
    - Data Type: Qualitative
    - Description: How satisfied the patient was with how actively the doctor listened to their issues, on a scale of one to eight.
    - Example: 8

Part 2:

C

1. Methods for dealing with data quality issues: Firstly, importing the CSV file and installing the pandas library using ‘import pandas as pd” will allow me to use the correct methods for data quality issues:
    a. Using the data dictionary, as well as a general view of the data using the info , I am able to find any variables that may stand out. Such as misspellings or obviously incorrect inputs.
    b. I will be using the isnull, coupled with the sum function, in order to count the number of null or blank entries.
    c. To find and remove duplicates, I will be using the duplicated, value_counts and drop_duplicates functions in conjunction in order to find, count and remove any duplicates that would skew the results.
    d. Using tools such as histograms and box plots, I can find any outliers that might significantly alter the results of the data.
    e. Using the mean, median and mode of the field, I can impute any missing data, or if the data is not needed, I am able to remove it altogether.

f. To re-express categorical variables, I will use the value_counts function, as well as the replace function to replace the current values with a more appropriate term or value.

2. Why I use those Methods:
    a. Using the info function will give me a general understanding of the data, as well as the data types for each field. This can show me any blank data, as well as any data types that may not be the most effective for that field, such as an object data type, when it should be an integer.
    b. Counting the blank/ null entries will allow me to impute the data so that it may be more usable for any visualizations or PCA analysis.
    c. Using the the duplicated, value_counts and drop_duplicates functions lets me view the exact number of duplicates, and allows me to remove them immediately.
    d. Using visualization tools will allow me to immediately notice any outliers that need further investigation.
    e. While not completely accurate, using the mean, median or mode to impute allows me to use an approximate but useful dataset. If I feel that too many fields are being imputed, I can also elect to leave the data as it is or impute a default value.
    f. I chose to use the value_counts function to look for any categorial values that might need re-expression since it allows me to view if there are any incorrectly input entries while displaying the data type. I can then use the replace function to create a dictionary that will more accurately allow me to categorize a field, such as giving a value to a term to calculate any outliers.
3. Programming Language Used:
    a. I elected to use the python programming language, since I have had some experience with the language in previous projects/ classes. Python is also home to many libraries that can be used for different scenarios even outside of data analysis. It's easy to understand syntax also makes python my choice for this performance assessment. in this process, I mainly used two core packages: NumPy and Pandas. NumPy supplies essential mathematical functions required for data transformation, while Pandas allows me to use a data frame, which is a spreadsheet format within python, with built-in capabilities to manipulate and standardize data. Once the data is refined, we employ Principal Component Analysis (PCA) with Scikit-Learn's PCA module to delve into the principal components. Additionally, Seaborn assists in generating scree plots as part of the PCA analysis. Matplotlib also allows me to visualize useful data, such as using a histogram to find outliers.

4. Detection Code:

    a. Here are some screenshots of my code from a Jupyter Notebook, please see code attached to this submission for the journal.

In [23]:
```
#import necessary libraries

import numpy as np
import pandas as pd
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
#import csv file as df
df = pd.read_csv('/Users/josealdana/Desktop/D206_files/medical_csv/medical_raw_data.csv')
```

In [38]:
```
#view of the dataframe
df
```

Out[38]:

	Unnamed: 0	CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	...	TotalCharg
0	1	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Morgan	35621	34.34960	...	3191.04877
1	2	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	Jackson	32446	30.84513	...	4214.90534
2	3	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Minnehaha	57110	43.54321	...	2177.58676
3	4	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	Waseca	56072	43.89744	...	2465.11896
4	5	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	King William	23181	37.59894	...	1885.65513

In [17]:
```
#describe the statistics for the data
df.describe()
```

Out[17]:

	Unnamed: 0	CaseOrder	Zip	Lat	Lng	Population	Children	Age	Income	VitD_levels	...	Tota
count	10000.00000	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	7412.000000	7586.000000	7536.000000	10000.000000	...	10000
mean	5000.50000	5000.50000	50159.323900	38.751099	-91.243080	9965.253800	2.098219	53.295676	40484.438268	19.412675	...	5891
std	2886.89568	2886.89568	27469.588208	5.403085	15.205998	14824.758614	2.155427	20.659182	28664.861050	6.723277	...	3377
min	1.00000	1.00000	610.000000	17.967190	-174.209690	0.000000	0.000000	18.000000	154.080000	9.519012	...	1256
25%	2500.75000	2500.75000	27592.000000	35.255120	-97.352982	694.750000	0.000000	35.000000	19450.792500	16.513171	...	3253
50%	5000.50000	5000.50000	50207.000000	39.419355	-88.397230	2769.000000	1.000000	53.000000	33942.280000	18.080560	...	5852
75%	7500.25000	7500.25000	72411.750000	42.044175	-80.438050	13945.000000	3.000000	71.000000	54075.235000	19.789740	...	7614
max	10000.00000	10000.00000	99929.000000	70.560990	-65.290170	122814.000000	10.000000	89.000000	207249.130000	53.019124	...	21524

In [19]:
```
# Get an understanding of the data, and the data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 53 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0          10000 non-null  int64  
 1   CaseOrder           10000 non-null  int64  
 2   Customer_id         10000 non-null  object 
 3   Interaction         10000 non-null  object 
 4   UID                 10000 non-null  object 
 5   City                10000 non-null  object 
 6   State               10000 non-null  object 
 7   County              10000 non-null  object 
 8   Zip                 10000 non-null  int64  
 9   Lat                 10000 non-null  float64
 10  Lng                 10000 non-null  float64
 11  Population          10000 non-null  int64  
 12  Area                10000 non-null  object 
 13  Timezone            10000 non-null  object 
 14  Job                 10000 non-null  object 
 15  Children            7412 non-null   float64
 16  Age                 7586 non-null   float64
 17  Education           10000 non-null  object 
 18  Employment          10000 non-null  object 
 19  Income              7536 non-null   float64
 20  Marital             10000 non-null  object 
 21  Gender              10000 non-null  object 
 22  ReAdmis             10000 non-null  object 
 23  VitD_levels         10000 non-null  float64
```

In [26]:
```
#find any duplicates

df.duplicated()
```

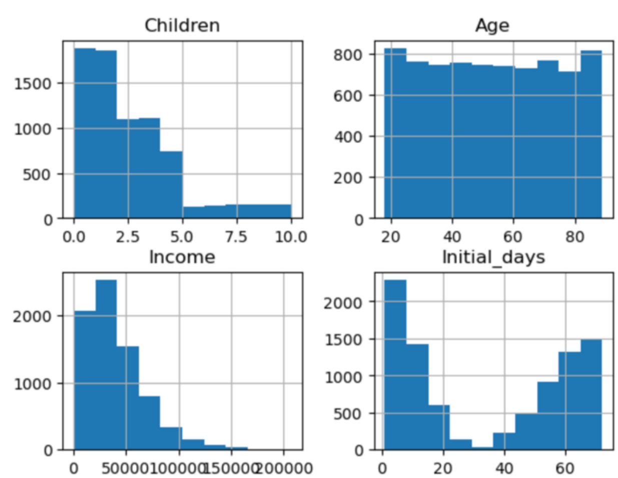
Out[26]:
```
0       False
1       False
2       False
3       False
4       False
        ...  
9995    False
9996    False
9997    False
9998    False
9999    False
Length: 10000, dtype: bool
```

In [29]:
```
# count the number of null entries
df.isnull().sum()
```

Out[29]:
```
Unnamed: 0              0
CaseOrder               0
Customer_id             0
Interaction             0
UID                     0
City                    0
State                   0
County                  0
Zip                     0
Lat                     0
Lng                     0
Population              0
Area                    0
Timezone                0
Job                     0
Children             2588
Age                  2414
Education               0
Employment              0
Income               2464
Marital                 0
Gender                  0
ReAdmis                 0
VitD_levels             0
Doc_visits              0
Full_meals_eaten        0
VitD_supp               0
Soft_drink           2467
Initial_admin           0
HighBlood               0
Stroke                  0
Complication_risk       0
Overweight            982
```

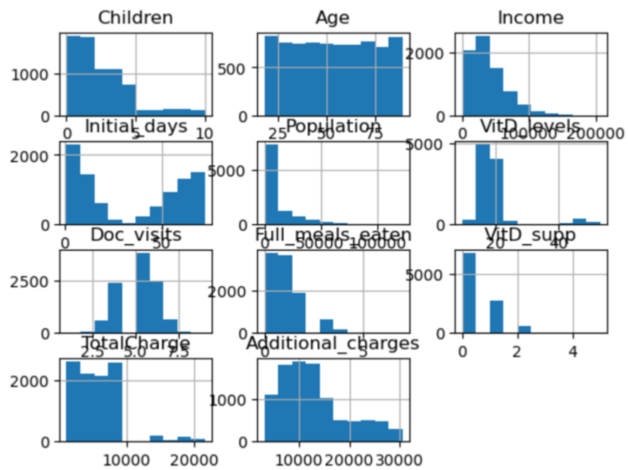
In [41]:
```
# this histogram will show me which imputation method to use for null values: median/mean
df[['Children', 'Age', 'Income', 'Initial_days']].hist()
```

Out[41]:
```
array([[<Axes: title={'center': 'Children'}>,
        <Axes: title={'center': 'Age'}>],
       [<Axes: title={'center': 'Income'}>,
        <Axes: title={'center': 'Initial_days'}>]], dtype=object)
```



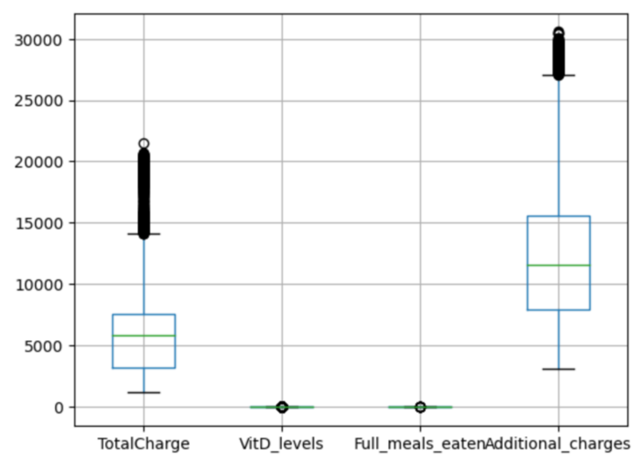
In [50]:
```python
# Finding any outliers for quantitative fields

df[['Children', 'Age', 'Income', 'Initial_days', 'Population', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'Vit
```

Out[50]:
```
array([[<Axes: title={'center': 'Children'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Income'}>],
       [<Axes: title={'center': 'Initial_days'}>,
        <Axes: title={'center': 'Population'}>,
        <Axes: title={'center': 'VitD_levels'}>],
       [<Axes: title={'center': 'Doc_visits'}>,
        <Axes: title={'center': 'Full_meals_eaten'}>,
        <Axes: title={'center': 'VitD_supp'}>],
       [<Axes: title={'center': 'TotalCharge'}>,
        <Axes: title={'center': 'Additional_charges'}>, <Axes: >]],
      dtype=object)
```



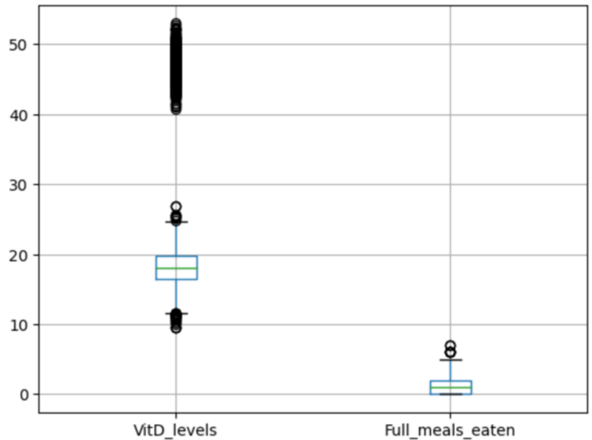
In [52]:
```python
#Further explore outliers using box plot

df[["TotalCharge", "VitD_levels", "Full_meals_eaten", "Additional_charges"
]].boxplot()
```

Out[52]: <Axes: >



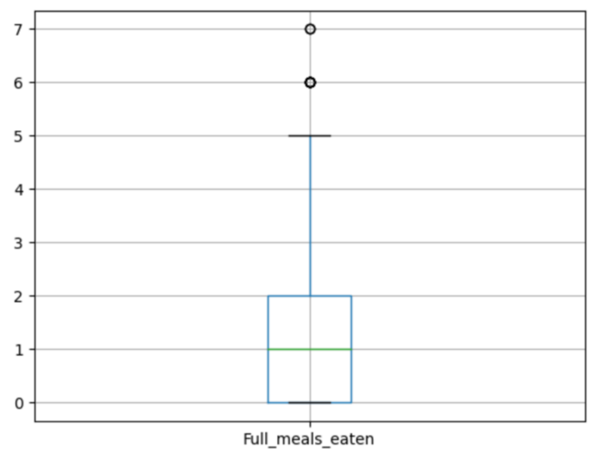
In [91]:
```
#let's further explore VitD_levels and Full_meals_eaten seperately

df[["VitD_levels", "Full_meals_eaten"
]].boxplot()
```

Out[91]: <Axes: >



In [92]:
```
#let's take an ever closer look at Full_meals_eaten
df[["Full_meals_eaten"
]].boxplot()
```

Out[92]: <Axes: >



In [93]:
```
#checking the top twenty values should give us an idea if this is an outlier

df.Full_meals_eaten.nlargest(n=20)
```

Out[93]:
```
958     7
4709    7
1231    6
2184    6
6068    6
7217    6
8144    6
9986    6
550     5
697     5
1148    5
1456    5
2315    5
2652    5
2746    5
2877    5
2919    5
4345    5
4902    5
5367    5
Name: Full_meals_eaten, dtype: int64
```

In [56]:
```
#the next few cells will verify that quantitative fields don't contain any mispelling/categorization errors.

df.Soft_drink.value_counts()
```

Out[56]:
```
No      5589
Yes     1944
Name: Soft_drink, dtype: int64
```

In [57]:
```
df.Area.value_counts()
```

Out[57]:
```
Rural       3369
Suburban    3328
Urban       3303
Name: Area, dtype: int64
```

In [58]:
```
df.Timezone.value_counts()
```

Out[58]:
```
America/New_York                  3889
America/Chicago                   3771
America/Los_Angeles                937
America/Denver                     612
America/Detroit                    262
America/Indiana/Indianapolis       151
America/Phoenix                    100
America/Boise                       86
America/Anchorage                   50
America/Puerto_Rico                 43
Pacific/Honolulu                    34
America/Menominee                   14
America/Nome                        12
America/Indiana/Vincennes            8
America/Kentucky/Louisville          6
America/Sitka                        6
America/Toronto                      5
America/Indiana/Marengo              3
America/Indiana/Tell_City            3
America/North_Dakota/Beulah          2
America/Yakutat                      1
America/Indiana/Winamac              1
America/Indiana/Knox                 1
America/North_Dakota/New_Salem       1
America/Indiana/Vevay                1
America/Adak                         1
Name: Timezone, dtype: int64
```

In [59]:
```
df.Education.value_counts()
```

Out[59]:
```
Regular High School Diploma                 2444
Bachelor's Degree                           1724
Some College, 1 or More Years, No Degree    1484
9th Grade to 12th Grade, No Diploma          832
Associate's Degree                           797
Master's Degree                              701
Some College, Less than 1 Year               642
Nursery School to 8th Grade                  552
GED or Alternative Credential                389
Professional School Degree                   208
No Schooling Completed                       133
Doctorate Degree                              94
Name: Education, dtype: int64
```

In [60]:
```
df.Marital.value_counts()
```

Out[60]:
```
Widowed          2045
Married          2023
Separated        1987
Never Married    1984
Divorced         1961
Name: Marital, dtype: int64
```

In [61]:
```
df.Gender.value_counts()
```

Out[61]:
```
Female                  5018
Male                    4768
Prefer not to answer     214
Name: Gender, dtype: int64
```

In [62]:
```
df.ReAdmis.value_counts()
```

Out[62]:
```
No     6331
Yes    3669
Name: ReAdmis, dtype: int64
```

In [63]:
```
df.Soft_drink.value_counts()
```

Out[63]:
```
No     5589
Yes    1944
Name: Soft_drink, dtype: int64
```

In [64]:
```
df.Initial_admin.value_counts()
```

Out[64]:
```
Emergency Admission      5060
Elective Admission       2504
Observation Admission    2436
Name: Initial_admin, dtype: int64
```

In [65]:
```
df.HighBlood.value_counts()
```

Out[65]:
```
No     5910
Yes    4090
Name: HighBlood, dtype: int64
```

In [66]:
```
df.Stroke.value_counts()
```

Out[66]:
```
No     8007
Yes    1993
Name: Stroke, dtype: int64
```

In [68]:
```
df.Complication_risk.value_counts()
```

Out[68]:
```
Medium    4517
High      3358
Low       2125
Name: Complication_risk, dtype: int64
```

In [69]:
```
df.Overweight.value_counts()
```

Out[69]:
```
1.0    6395
0.0    2623
Name: Overweight, dtype: int64
```

In [71]:
```
df.Arthritis.value_counts()
```
Out[71]:
```
No     6426
Yes    3574
Name: Arthritis, dtype: int64
```
In [72]:
```
df.Diabetes.value_counts()
```
Out[72]:
```
No     7262
Yes    2738
Name: Diabetes, dtype: int64
```
In [73]:
```
df.Hyperlipidemia.value_counts()
```
Out[73]:
```
No     6628
Yes    3372
Name: Hyperlipidemia, dtype: int64
```
In [74]:
```
df.BackPain.value_counts()
```
Out[74]:
```
No     5886
Yes    4114
Name: BackPain, dtype: int64
```
In [75]:
```
df.Anxiety.value_counts()
```
Out[75]:
```
0.0    6110
1.0    2906
Name: Anxiety, dtype: int64
```
In [76]:
```
df.Allergic_rhinitis.value_counts()
```
Out[76]:
```
No     6059
Yes    3941
Name: Allergic_rhinitis, dtype: int64
```
In [77]:
```
df.Asthma.value_counts()
```
Out[77]:
```
No     7107
Yes    2893
Name: Asthma, dtype: int64
```
In [79]:
```
df.Services.value_counts()
```
Out[79]:
```
Blood Work     5265
Intravenous    3130
CT Scan        1225
MRI             380
Name: Services, dtype: int64
```
In [80]:
```
#same thing for the item columns

df.Item1.value_counts()
```
Out[80]:
```
4    3455
3    3404
5    1377
2    1315
6     225
1     213
7      10
8       1
Name: Item1, dtype: int64
```
In [81]:
```
df.Item2.value_counts()
```
Out[81]:
```
3    3439
4    3351
5    1421
2    1360
1     213
6     204
7      12
Name: Item2, dtype: int64
```
In [82]:
```
df.Item3.value_counts()
```
Out[82]:
```
4    3464
3    3379
5    1358
2    1356
6     220
1     211
7      11
8       1
Name: Item3, dtype: int64
```

In [83]:
```
df.Item4.value_counts()
```
Out[83]:
```
3    3422
4    3394
5    1388
2    1346
6     231
1     207
7      12
Name: Item4, dtype: int64
```

In [84]:
```
df.Item5.value_counts()
```
Out[84]:
```
4    3446
3    3423
2    1380
5    1308
6     219
1     211
7      13
Name: Item5, dtype: int64
```

In [85]:
```
df.Item6.value_counts()
```
Out[85]:
```
4    3464
3    3371
5    1403
2    1319
6     220
1     213
7      10
Name: Item6, dtype: int64
```

In [86]:
```
df.Item7.value_counts()
```
Out[86]:
```
4    3487
3    3456
2    1345
5    1274
1     215
6     212
7      11
Name: Item7, dtype: int64
```

In [96]:
```
df.Population.nsmallest(n=20)
```
Out[96]:
```
42      0
44      0
86      0
171     0
241     0
407     0
447     0
500     0
520     0
555     0
563     0
650     0
655     0
1019    0
1065    0
1145    0
1220    0
1323    0
1353    0
1376    0
Name: Population, dtype: int64
```

```
In [98]: df.VitD_levels.nlargest(n=20)

Out[98]: 1963    53.019124
         1306    52.757599
         7157    52.370764
         7230    52.271584
         2615    52.156112
         3473    52.124137
         7527    52.117337
         580     52.063590
         1798    51.671571
         8681    51.659892
         447     51.430034
         2746    51.305309
         6938    51.216965
         9907    51.202138
         5043    51.030046
         7760    50.958467
         8357    50.927637
         2156    50.915519
         3386    50.888076
         5595    50.884049
         Name: VitD_levels, dtype: float64
```

```
In [104]: df.Income.describe()

Out[104]: count      7536.000000
          mean      40484.438268
          std       28664.861050
          min         154.080000
          25%       19450.792500
          50%       33942.280000
          75%       54075.235000
          max      207249.130000
          Name: Income, dtype: float64
```

```
In [87]: df.Item8.value_counts()

Out[87]: 3    3401
         4    3337
         5    1429
         2    1391
         6     221
         1     209
         7      12
         Name: Item8, dtype: int64
```

Part 3:

D.

1. Data Findings:

- Luckily there were no duplicates found in this dataset.
- The first column is an index column, which is unnecessary since Jupyter Notebook automatically indexes the data frame.
- The Zip code field is listed as an integer, but since it is categorical and not something that we will perform calculations with, it would be more useful as an object data type.
- The item 1-8 at the end of the dataset is not descriptive and need to be renamed.
- Item 1-8 are also shown to be integer data types, but a more effective data type would be category.
- The education field is an object data type, but it would be more effective as a categorical data type.
- All these fields would be more effective as a Boolean data type:

i. ReAdmis
ii. Soft_drink
iii. HighBlood
iv. Stroke
v. Overweight
vi. Arthritis
vii. Diabetes
viii. Hyperlipidemia
ix. Back_pain
x. Anxiety
xi. Allergic_rhinitis
xii. Reflux_esophagitis
xiii. Asthma

- There seemed to be a large difference between the largest and smallest number in the TotalCharge field. The minimum is 1256.75, and the maximum is 21524.22. This seems like a large gap, but in my opinion, this is plausible in a hospital, so these outliers will be left in.
- There were also many entries in the VitD_levels field that were considered outliers, but upon closer inspection, it seemed that the levels entered were accurate, and those patients indeed had a higher than usual Vitamin D level. Therefore, they will be left in.
- The Children and Age field are both float data types. When they would show better results as integers.
- Total Charge, Additional Charge and Income are rounded to the 6th decimal place, which seems excessive and will be brought to the 2nd decimal place.
- VitD_levels also have too many decimal places and will be brought down to the 2nd decimal place.
- The Children field was found to have 2588 entries missing.
- The Age field was found to have 2414 entries missing.
- The Income field was found to have 2464 entries missing.
- The Soft_drink field was found to have 2467 entries missing.
- The Overweight field was found to have 982 entries missing.
- The Anxiety field was found to have 984 entries missing.
- The Initial_days field was found to have 1056 entries missing.

2. Treatment of the data:

- I will be removing the first index column.
- The Zip code field will have its data type altered to object.

- Item 1-8 will be renamed so that they can provide more context on their respective columns. These names will be:
    i. ○ **Item1:** Time_admis
    ii. ○ **Item2:** Time_treat
    iii. ○ **Item3:** Time_visit
    iv. ○ **Item4:** Reliability
    v. ○ **Item5:** Options
    vi. ○ **Item6:** Hours_treat
    vii. ○ **Item7:** Courteous
    viii. ○ **Item8:** Doctor_listen
- Items 1-8 will be changed from an integer data type to a category data type.
- The fields listed in D1 that would be more effective as Boolean will be changed accordingly.
- Children and Age will be changed to an integer data type.
- 'TotalCharge', 'Additional_Charges', 'Initial_days' and VitD_levels will be changed to only display up to the 2nd decimal place.
- For the null fields that were found, I will be addressing them as such:
    i. Children- The distribution is skewed to the left; therefore, the most effective method of imputation would use the median.
    ii. Age- Age is a very important field for categorization, and while I feel that removing them would be the most effective and accurate method. That would remove about 25% of the total entries. Therefore, I feel that the most effective way of dealing with these missing values is to impute with the mean of that field.
    iii. Income- The histogram for the income field is skewed to the left, meaning that the most effective way to impute this data would be to use the median.
    iv. The Soft_drink, Overweight and Anxiety fields will be Boolean data types, therefore it can only be Yes or No. In my opinion, these fields are extremely important to a patient's health, and if they are left blank, that patient would most likely not have that field in mind. Meaning I feel comfortable filling in these missing values with a 0, or no.
    v. The Initial_days field's histogram is uniform, therefore the most effective method to impute would be to use the mean of that field.
- For the sake of keeping the data homogenous, I will be re expressing the Anxiety, and Overweight fields into Yes or No, as opposed to 1 or 0.

3. The work performed:

- I removed the Index column at the beginning of the csv file.

- I altered the data type for the Zip field to an Object since it would be easier to categorize.
- I renamed all the item fields at the end of the dataset so that they are more understandable when looking at the dataset. Their names were changed to:
    - i. **Item1:** Time_admis
    - ii. **Item2:** Time_treat
    - iii. **Item3:** Time_visit
    - iv. **Item4:** Reliability
    - v. **Item5:** Options
    - vi. **Item6:** Hours_treat
    - vii. **Item7:** Courteous
    - viii. **Item8:** Doctor_listen
- I corrected all the missing values as such:
    - Children- Used the median number of children to impute.
    - Age- Used the mean to impute.
    - Income- Used the median to impute.
    - Initial_days- Used the mean to impute.
    - Soft_drink- Imputed empty values with 'No'.
    - Overweight- Imputed empty values with '0'.
    - Anxiety- Imputed empty values with '0'.
- The 'Children' and 'Age' field were changed to integer.
- Item 1-8 were changed to a category data type.
- These columns were changed into Boolean data types:
        - ReAdmis
        - Soft_drink
        - HighBlood
        - Stroke
        - Overweight
        - Arthritis
        - Diabetes
        - Hyperlipidemia
        - Back_pain
        - Anxiety
        - Allergic_rhinitis
        - Reflux_esophagitis
        - Asthma
- But first the existing entries were changed from 'Yes' and 'No' into '1' and '0'. This was to ensure that it would be read correctly as a Boolean.
- These Booleans were then re-expressed back into 'Yes' or 'No', for better readability.

- 'TotalCharge', 'Additional_charges', 'Initial_days' and 'VitD_levels' were rounded down to two decimal places.
- These changes leave the entire dataset intact, while imputing any null values.

4. Treatment Code:

- This is the code used to treat the data quality issues:

```
In [126]: #I will drop the initial index column since it is not necessary
df = df.drop(df.columns[0], axis=1)

In [127]: columns_to_drop = [col for col in df.columns if col.startswith('unnam

In [128]: df = df.drop(columns=columns_to_drop)

In [129]: df
```

Out[129]:

	CaseOrder	Customer_id	Interaction	UID	City
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland
			5885f56b-		

```
In [130]: #The Zip code field will be changed to an Object data type, since it 
df['Zip'] = df['Zip'].astype('object')

In [131]: #The final 8 columns will be renamed to better understand them

new_column_names = {
    'Item1': 'Time_admis',
    'Item2': 'Time_treat',
    'Item3': 'Time_visit',
    'Item4': 'Reliability',
    'Item5': 'Options',
    'Item6': 'Hours_treat',
    'Item7': 'Courteous',
    'Item8': 'Doctor_listen'
}

In [132]: df = df.rename(columns=new_column_names)
```

In [313]:
```python
#The following columns will be changed to boolean for cohesion

boolean_columns = [
    'ReAdmis', 'Soft_drink', 'HighBlood', 'Stroke', 'Overweight',
    'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
    'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma'
]
```

In [316]:
```python
#Any 'Yes' or 'No' entries will be changed to 1 and 0.
#So that when changed to boolean they will correctly change

df[boolean_columns] = df[boolean_columns].replace({'Yes': 1, 'No': 0}
```

In [318]:
```python
df[boolean_columns] = df[boolean_columns].astype(bool)
```

In [320]:
```python
bool_dict = {False: 'No', True: 'Yes'}
```

In [321]:
```python
df[boolean_columns] = df[boolean_columns].replace(bool_dict)
```

In [288]:
```python
df[boolean_columns]
```

Out[288]:

	ReAdmis	Soft_drink	HighBlood	Stroke	Overweight	Arthritis	Diabetes	Hyperlipidemia
0	No	Yes	Yes	No	No	Yes	Yes	No
1	No	No	Yes	No	Yes	No	No	No
2	No	No	Yes	No	Yes	No	Yes	No
3	No	No	No	Yes	No	Yes	No	No
4	No	Yes	No	No	No	No	No	Yes
...	...	...	...	...	...	...	...	...
9993	No	No	Yes	No	Yes	No	No	Yes
9994	No	No	No	No	No	Yes	No	No
9995	No	No	Yes	No	Yes	No	No	No
9996	Yes	No	Yes	No	Yes	Yes	Yes	No
9998	Yes	No	No	No	Yes	No	No	No

In [76]:
```python
#The next few lines will fix the missing values

#Median will be used to impute the 'Children' field
median_children = df['Children'].median()
```

In [78]:
```python
df['Children'].fillna(median_children, inplace=True)
```

In [107]:
```python
df['Age'].fillna(mean_age, inplace=True)
```

In [108]:
```python
#Missing values from the 'Age' field will be imputed with the mean

mean_age = df['Age'].mean()
```

In [80]:
```python
#Missing values from the 'Income' field will be imputed with the Median
median_income = df['Income'].median()
```

In [81]:
```python
df['Income'].fillna(median_income, inplace=True)
```

```
In [390]: df['Initial_days'].fillna(mean_initial_days, inplace=True)
```

```
In [391]: #Missing boolean values will be imputed with a '0' or 'No' accordingl

          #'Soft_drink' will be filled in with 'No'
          df['Soft_drink'].fillna(False, inplace=True)
```

```
In [392]: #'Overweight' will be filled in with '0'
          df['Overweight'].fillna(False, inplace=True)
```

```
In [393]: #'Anxiety' will be filled in with '0'
          df['Anxiety'].fillna(False, inplace=True)
```

```
In [394]: #The children and Age fields would work best as integer data types
          df['Children'] = df['Children'].astype(int)
          df['Age'] = df['Age'].astype(int)
```

```
In [396]: #These cells will convert the final 8 columns into a category data ty
          columns_to_convert = ['Time_admis', 'Time_treat', 'Time_visit', 'Reli
```

```
In [397]: for column in columns_to_convert:
              df[column] = df[column].astype('category')
```

```
In [398]: #These cells will bring the decimal place down to 2 places

          df['TotalCharge'] = df['TotalCharge'].round(2)
```

```
In [399]: df['Additional_charges'] = df['Additional_charges'].round(2)
```

```
In [403]: df['Initial_days'] = df['Initial_days'].round(2)
```

```
In [400]: df['VitD_levels'] = df['VitD_levels'].round(2)
```

(For any code that does not fit into the screenshot fully, please see attached python file.)

5. Clean Data: Please see attached CSV file with clean data.
6. Disadvantages of Methods Used:

- Not being someone who understands what a medical institution might deem necessary data for a patient may have led to certain entries being changed or removed unnecessarily.
- The methods of imputation for missing values are also an approximation and can skew the data in an unexpected way.
- The much smaller dataset can affect the principal component analysis in the next section.

- The imputation for ‘Soft_drink’, ‘Overweight’ and ‘Anxiety’ may not accurately represent the patient. Since they might have forgotten to fill the field correctly.

7. Clean Dataset Issues.

A data analyst may struggle with getting a complete and accurate result from my clean dataset. Since all the null values were imputed. The imputation methods I applied could influence the identification of conditions associated with anxiety. If imputation introduces biases or inaccuracies, the correlation between variables and anxiety may not be accurate. For instance, imputing missing values using central tendency measures such as filling missing values in the ‘Anxiety’ field with ‘No’, could incorrectly categorize someone that may be suffering from the condition. This potential skew in the data may also impact any statistical analysis. Since, the biases introduced during imputation or data reduction may impact the statistical power of tests exploring relationships between various conditions and anxiety.

To gain valuable insight from the clean data, there needs to be an understanding of the potential biases in the dataset. Therefore, collaborating with domain experts who can provide insights into the specific conditions associated with anxiety can be valuable in mitigating the impact of data cleaning limitations on the pursuit of answers to the research question.

Part 4: PCA

E1. Perform PCA

In order to perform PCA, the variables used must be continuous. Meaning the relevant variables are:

- Latitude
- Longitude
- Children
- Age
- Population
- Income
- Vitamin D Levels
- Doctor Visits
- Full Meals Eaten
- Vitamin D Supplements
- Initial Days
- Total Charge

- Additional Charges

This is the loading matrix for these fields:

In [190]:
```
#This is a view of the loadings matrix
pca_load
```

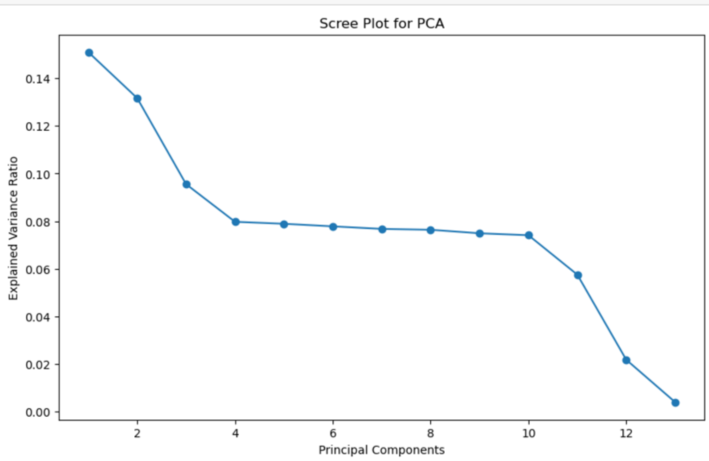
Out[190]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
0	-0.022612	-0.017581	-0.702290	0.113074	-0.039360	-0.036637	-0.036602	-0.063649	-0.016962	-0.093741	0.689778	0.005550	0.001273
1	0.001196	0.019392	0.342294	-0.371825	-0.135830	-0.499137	0.485761	-0.291881	0.121201	-0.024911	0.374242	-0.009489	0.000137
2	0.029971	-0.024484	0.604281	0.172014	0.137973	0.307929	-0.322302	0.144029	0.018025	-0.041071	0.602571	0.012588	0.004054
3	0.004681	0.011409	0.003486	0.193361	-0.248651	0.579625	0.612839	0.040521	0.298858	-0.312937	-0.010106	0.009195	0.003096
4	0.120715	0.695604	-0.003694	0.014438	-0.013343	-0.003052	-0.024366	-0.025985	-0.010408	0.000828	0.005069	0.706439	0.026323
5	0.000116	0.000425	0.029382	0.429834	0.442499	-0.082926	0.512376	0.154310	-0.410033	0.390916	0.064659	0.015279	0.000983
6	0.537222	-0.083799	-0.046582	-0.263335	0.324210	0.017148	0.067956	0.153856	-0.183402	-0.410444	-0.011081	0.028188	-0.544189
7	0.004749	-0.001989	-0.018061	0.281674	0.588728	-0.082201	-0.065758	-0.514146	0.518711	-0.143374	-0.092837	-0.006631	-0.000314
8	-0.010514	0.033863	-0.139749	-0.492160	0.301607	0.207395	0.078886	0.360858	0.470601	0.488106	0.082473	0.005969	0.004508
9	0.033749	0.009135	0.023302	0.348205	-0.087653	-0.511294	-0.007930	0.635942	0.394972	-0.216149	-0.025611	0.003660	0.002516
10	0.442931	-0.094255	0.017327	0.289548	-0.390885	-0.011591	-0.067842	-0.202470	0.215735	0.513896	0.012956	0.013275	-0.448658
11	0.695448	-0.115105	-0.026407	-0.018697	0.000378	0.003138	0.007291	-0.015021	-0.009815	0.009048	-0.004828	-0.032016	0.707489
12	0.119498	0.695938	-0.003590	0.026002	-0.000894	0.014814	-0.015526	-0.001949	-0.018506	0.001296	0.018936	-0.705873	-0.035719

E2. Which components should be retained?

A scree plot was used in order to determine which principal components should be retained, this is the visualization:

In [195]:
```
#This will plot the visualization

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_var_ratio) + 1), explained_var_ratio, marker='o', linestyle='-')
plt.title('Scree Plot for PCA')
plt.xlabel('Principal Components')
plt.ylabel('Explained Variance Ratio')
plt.show()
```



Based on this visualization, there is a clear elbow at principal component 4, therefore components 5-13 will not be used. It could be argued that since there is a substantial drop near the end of the plot, those principal components could be used as well. But it is possible that the end components merely contain small, specialized sources of data, and without proper domain knowledge, retaining them may not be advantageous.

E3. Benefits of this PCA.

An organization, specifically the hospital from this dataset, would benefit from the results of this PCA by identifying and retaining the most important features or variables. Which can be extremely helpful for a large dataset, such as one that contains thousands of entries with patient information. By reducing to the most important variables, there will be a general improvement in efficiency, which can then benefit tests such as predictive models. Additionally, this PCA can reveal patterns in patient data by finding combinations of symptoms that can co-occur and compare them to readmissions. Ultimately, these insights facilitate more targeted and effective healthcare strategies, enhancing both patient outcomes and organizational performance.

F. Panopto Recording.

The required Panopto Recording will be included with the final submission.

G. Code References.

There were no external code references used for this assessment.

H. References Cited.

There were no external sources used for the content of the assessment.