In [1]:
```
#Import data and packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA


file = 'C:/Users/cdric/OneDrive/Desktop/School/d212/medical_clean.csv'
data = pd.read_csv(file,na_values='NA') #replace NA values with NaN
data.head()
```

Out[1]:

	CaseOrder	Customer_id	Interaction	UID	City	State	Co
0	1	C412403	8cd49b13-f45a-4b47-a2bd-173ffa932c2f	3a83ddb66e2ae73798bdf1d705dc0932	Eva	AL	Mo
1	2	Z919181	d2450b70-0337-4406-bdbb-bc1037f1734c	176354c5eef714957d486009feabf195	Marianna	FL	Ja
2	3	F995323	a2057123-abf5-4a2c-abad-8ffe33512562	e19a0fa00aeda885b8a436757e889bc9	Sioux Falls	SD	Minne
3	4	A879973	1dec528d-eb34-4079-adce-0d7a40e82205	cd17d7b6d152cb6f23957346d11c3f07	New Richland	MN	W
4	5	C544523	5885f56b-d6da-43a3-8760-83583af94266	d2f0425877b10ed6bb381f3e2579424a	West Point	VA	W

5 rows × 50 columns

In [2]:
```
#Make sure that there are no null values
print(data.isnull().sum())
```

```
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              0
Age                   0
Income                0
Marital               0
Gender                0
ReAdmis               0
VitD_levels           0
Doc_visits            0
Full_meals_eaten      0
vitD_supp             0
Soft_drink            0
Initial_admin         0
HighBlood             0
Stroke                0
Complication_risk     0
Overweight            0
Arthritis             0
Diabetes              0
Hyperlipidemia        0
BackPain              0
Anxiety               0
Allergic_rhinitis     0
Reflux_esophagitis    0
Asthma                0
Services              0
Initial_days          0
TotalCharge           0
Additional_charges    0
Item1                 0
Item2                 0
Item3                 0
Item4                 0
Item5                 0
Item6                 0
Item7                 0
Item8                 0
dtype: int64
```

In [3]:

```
#These columns appear to be unique to the patient and procedures so we are checking if
print(data['CaseOrder'].is_unique)
print(data['Customer_id'].is_unique)
print(data['Interaction'].is_unique)
print(data['UID'].is_unique)
```

```
True
True
True
True
```

In [4]:
```
#https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=60fa4159-94ba-4f41-9ba8-c

##Steps: 1) Standardize, 2) Compute covariance, 3) Compute eigen vectors and values 4)
```

In [5]:
```
#Create set with only continuous variables
##Population, Income, VitD_levels, Initial_days, TotalCharge, Additional_charges, Age

cont_data=data[['Population', 'Income', 'VitD_levels', 'Initial_days', 'TotalCharge',
print(cont_data)
```

```
      Population    Income  VitD_levels  Initial_days  TotalCharge  \
0           2951  86575.93    19.141466     10.585770  3726.702860
1          11303  46805.99    18.940352     15.129562  4193.190458
2          17125  14370.14    18.057507      4.772177  2434.234222
3           2162  39741.49    16.576858      1.714879  2127.830423
4           5287   1209.56    17.439069      1.254807  2113.073274
...          ...       ...          ...           ...          ...
9995        4762  45967.61    16.980860     51.561220  6850.942000
9996        1251  14983.02    18.177020     68.668240  7741.690000
9997         532  65917.81    17.129070     70.154180  8276.481000
9998         271  29702.32    19.910430     63.356900  7644.483000
9999       41524  62682.63    18.388620     70.850590  7887.553000

      Additional_charges  Age
0           17939.403420   53
1           17612.998120   51
2           17505.192460   53
3           12993.437350   78
4            3716.525786   22
...                  ...  ...
9995         8927.642000   25
9996        28507.150000   87
9997        15281.210000   45
9998         7781.678000   43
9999        11643.190000   70

[10000 rows x 7 columns]
```

In [6]:
```
#Scale Data

scaler=StandardScaler()
scaled_data = scaler.fit_transform(cont_data)
scaled_data = pd.DataFrame(scaled_data, columns = ['Population', 'Income', 'VitD_level
                                                    'TotalCharge', 'Additional_charges'
print(scaled_data.head())
```

```
   Population    Income  VitD_levels  Initial_days  TotalCharge  \
0   -0.473168  1.615914     0.583603     -0.907310    -0.727185   
1    0.090242  0.221443     0.483901     -0.734595    -0.513228   
2    0.482983 -0.915870     0.046227     -1.128292    -1.319983   
3   -0.526393 -0.026263    -0.687811     -1.244503    -1.460517   
4   -0.315586 -1.377325    -0.260366     -1.261991    -1.467285   

   Additional_charges       Age  
0            0.765005 -0.024795  
1            0.715114 -0.121706  
2            0.698635 -0.024795  
3            0.009004  1.186592  
4           -1.408991 -1.526914  
```

In [7]:
```
#Extract cleaned data from Jupyter to desktop
scaled_data.to_csv(r'C:\Users\cdric\OneDrive\Desktop\School\d212\D212Task 2\scaled_mec
```

In [8]:
```
#Perform PCA
pca_all = PCA(n_components = 7, random_state=42)
pc=pca_all.fit_transform(scaled_data)
print(pc)
```

```
[[-1.12949397  0.65171606  0.70603764 ...  1.60387358 -0.56744533
   0.09987943]
 [-0.82819815  0.52529657 -0.14860596 ...  0.39259784 -0.5969337
   0.1261863 ]
 [-1.63037103  0.68340864 -0.61460571 ... -0.82203778 -0.49984235
  -0.16371901]
 ...
 [ 1.86880505 -0.27460709  0.84720546 ...  0.62818536 -0.55110214
  -0.01949778]
 [ 1.40213738 -1.04930053 -1.04038589 ...  0.64869864  0.18497355
  -0.00593244]
 [ 1.8843154   0.13434397  0.73373261 ... -0.10636015  0.74145682
  -0.11558188]]
```

In [9]:
```
#Create a dataframe to explain all continuous variable features.
pc_df = pd.DataFrame(pc,columns = ['PC1','PC2','PC3','PC4','PC5','PC6','PC7'])
print(pc_df)
```

```
           PC1       PC2       PC3       PC4       PC5       PC6       PC7
0    -1.129494  0.651716  0.706038  0.100780  1.603874 -0.567445  0.099879
1    -0.828198  0.525297 -0.148606  0.332045  0.392598 -0.596934  0.126186
2    -1.630371  0.683409 -0.614606  0.350770 -0.822038 -0.499842 -0.163719
3    -1.807875  1.068282  0.387090 -0.694423 -0.301055  0.833054 -0.120546
4    -2.142475 -1.799882 -0.932004 -0.621448 -1.008335 -0.084879 -0.134660
...        ...       ...       ...       ...       ...       ...       ...
9995  0.771136 -1.516410  0.385633 -0.506705 -0.013706 -0.549460  0.026960
9996  2.037951  2.638300 -0.691467 -0.624637 -0.292485 -0.538032 -0.176982
9997  1.868805 -0.274607  0.847205 -0.612837  0.628185 -0.551102 -0.019498
9998  1.402137 -1.049301 -1.040386 -0.357959  0.648699  0.184974 -0.005932
9999  1.884315  0.134344  0.733733  2.118223 -0.106360  0.741457 -0.115582

[10000 rows x 7 columns]
```

In [10]:
```
#Contribution to each of the PCs, D1
load = pd.DataFrame(pca_all.components_.T, columns = ['PC1','PC2','PC3','PC4','PC5','F
load
```

Out[10]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Population	0.023764	-0.027794	0.139508	0.913250	-0.380757	0.014048	-0.000910
Income	-0.020681	-0.019072	0.717051	0.171784	0.674929	0.002157	0.001291
VitD_levels	-0.001640	0.019142	-0.682130	0.368144	0.631501	-0.001922	-0.001545
Initial_days	0.701091	-0.089764	0.005694	-0.015491	0.018090	0.031477	-0.706274
TotalCharge	0.702221	-0.079253	0.003713	-0.012829	0.017349	-0.031550	0.706491
Additional_charges	0.084934	0.701346	0.025860	0.022484	-0.008450	-0.705905	-0.036789
Age	0.084541	0.701622	0.018925	0.004898	-0.001245	0.706758	0.026259

In [11]:
```
#Calc variance explained by all PCs
print('Variance explained by all 7 principal components =', sum(pca_all.explained_vari
```

```
Variance explained by all 7 principal components = 100.0
```

In [12]:
```
#Captured variance, less than one is not important and can be dropped
vary = pca_all.explained_variance_ratio_*100
var_df1 = pd.DataFrame(vary.round(2), columns = ['Captured Variance Per PC'], index =
var_df1
```

Out[12]:

	Captured Variance Per PC
PC1	28.47
PC2	24.49
PC3	14.47
PC4	14.30
PC5	14.06
PC6	4.05
PC7	0.17

In [13]:
```
#Eigenvalues to determine PCs
eigenvalues = pca_all.explained_variance_
eigen_df = pd.DataFrame(eigenvalues.round(4), columns = ['Eigenvalues per PC'], index
eigen_df
```

Out[13]:

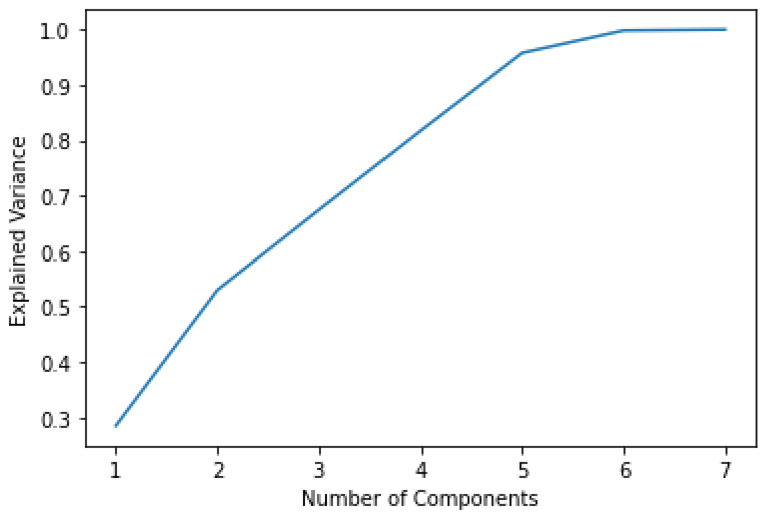
	Eigenvalues per PC
PC1	1.9929
PC2	1.7141
PC3	1.0128
PC4	1.0014
PC5	0.9842
PC6	0.2836
PC7	0.0117

In [14]:
```
#Cummulative sum
np.cumsum(pca_all.explained_variance_ratio_*100)
```

Out[14]:
```
array([ 28.46708594,  52.95208953,  67.41897126,  81.72330206,
        95.78202088,  99.83258854, 100.        ])
```

In [15]:
```
#Scree plot
#Components 1-5 are most significant

plt.plot(np.cumsum(pca_all.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Explained Variance')
plt.xticks(np.arange(len(np.cumsum(pca_all.explained_variance_ratio_))),
           np.arange(1, len(np.cumsum(pca_all.explained_variance_ratio_))+1))
plt.show()
```



In [16]:
```
#Explaining the PC variability.
print('Variance explained by the first pc =', np.cumsum(pca_all.explained_variance_rat
print('Variance explained by the first 2 pcs =', np.cumsum(pca_all.explained_variance_
print('Variance explained by the first 3 pcs =', np.cumsum(pca_all.explained_variance_
print('Variance explained by the first 4 pcs =', np.cumsum(pca_all.explained_variance_
print('Variance explained by all the pcs =', np.cumsum(pca_all.explained_variance_rati
```

```
Variance explained by the first pc = 28.467085941027054
Variance explained by the first 2 pcs = 52.95208953266629
Variance explained by the first 3 pcs = 67.41897125612338
Variance explained by the first 4 pcs = 81.72330206422662
Variance explained by all the pcs = 95.78202087830768
```

In [17]:
```
#Feature reduction to 5 variables since it makes up ~95.8% of variance
pc_5 = PCA(n_components = 5, random_state=42)
pc_5.fit(scaled_data)
var_pc5=pc_5.transform(scaled_data)

pca_5 =pc_5.explained_variance_ratio_*100
var_df1 = pd.DataFrame(pca_5.round(2), columns = ['Captured Variance per PC'],
                       index = ['PC1','PC2','PC3','PC4','PC5'])
var_df1
```

Out[17]:

	Captured Variance per PC
PC1	28.47
PC2	24.49
PC3	14.47
PC4	14.30
PC5	14.06

In [19]:

```
load = pd.DataFrame(pc_5.components_.T, columns = ['PC1','PC2','PC3','PC4','PC5'], inc
load
```

Out[19]:

	PC1	PC2	PC3	PC4	PC5
Population	0.023764	-0.027794	0.139508	0.913250	-0.380757
Income	-0.020681	-0.019072	0.717051	0.171784	0.674929
VitD_levels	-0.001640	0.019142	-0.682130	0.368144	0.631501
Initial_days	0.701091	-0.089764	0.005694	-0.015491	0.018090
TotalCharge	0.702221	-0.079253	0.003713	-0.012829	0.017349
Additional_charges	0.084934	0.701346	0.025860	0.022484	-0.008450
Age	0.084541	0.701622	0.018925	0.004898	-0.001245

In [ ]: