INFOSYS SPRINGBOARD INTERNSHIP

MileStone 2

Name: Rudrani Ghosh

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

data = pd.read_csv("/Users/rudranighosh/Downloads/Healthcare Providers.csv")
data.head()
```

```
/Users/rudranighosh/anaconda/anaconda3/lib/python3.11/site-packages/pandas/core/arrays/masked.py:60: UserWarning: Pandas r
equires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

Out[1]:

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider	Gender of the Provider	Entity Type of the Provider	Street Address 1 of the Provider	Street Address 2 of the Provider	...	HCPCS Code	HCPCS Description	HCPCS Drug Indicator	Num Servi
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	M.D.	F	I	1402 S GRAND BLVD	FDT 14TH FLOOR	...	99223	Initial hospital inpatient care, typically 70 ...	N	
1	3354385	1346202256	JONES	WENDY	P	M.D.	F	I	2950 VILLAGE DR	NaN	...	G0202	Screening mammography, bilateral (2-view study...	N	
2	3001884	1306820956	DUROCHER	RICHARD	W	DPM	M	I	20 WASHINGTON AVE	STE 212	...	99348	Established patient home visit, typically 25 m...	N	
3	7594822	1770523540	FULLARD	JASPER	NaN	MD	M	I	5746 N BROADWAY ST	NaN	...	81002	Urinalysis, manual test	N	
4	746159	1073627758	PERROTTI	ANTHONY	E	DO	M	I	875 MILITARY TRL	SUITE 200	...	96372	Injection beneath the skin or into muscle for ...	N	

5 rows × 27 columns

In [2]:
```
# information about the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 27 columns):
 #   Column                                                    Non-Null Count   Dtype  
---  ------                                                    --------------   -----  
 0   index                                                     100000 non-null  int64  
 1   National Provider Identifier                              100000 non-null  int64  
 2   Last Name/Organization Name of the Provider               100000 non-null  object 
 3   First Name of the Provider                                95745 non-null   object 
 4   Middle Initial of the Provider                            70669 non-null   object 
 5   Credentials of the Provider                               92791 non-null   object 
 6   Gender of the Provider                                    95746 non-null   object 
 7   Entity Type of the Provider                               100000 non-null  object 
 8   Street Address 1 of the Provider                          100000 non-null  object 
 9   Street Address 2 of the Provider                          40637 non-null   object 
 10  City of the Provider                                      100000 non-null  object 
 11  Zip Code of the Provider                                  100000 non-null  float64
 12  State Code of the Provider                                100000 non-null  object 
 13  Country Code of the Provider                              100000 non-null  object 
 14  Provider Type                                             100000 non-null  object 
 15  Medicare Participation Indicator                          100000 non-null  object 
 16  Place of Service                                          100000 non-null  object 
 17  HCPCS Code                                                100000 non-null  object 
 18  HCPCS Description                                         100000 non-null  object 
 19  HCPCS Drug Indicator                                      100000 non-null  object 
 20  Number of Services                                        100000 non-null  object 
 21  Number of Medicare Beneficiaries                          100000 non-null  object 
 22  Number of Distinct Medicare Beneficiary/Per Day Services  100000 non-null  object 
 23  Average Medicare Allowed Amount                           100000 non-null  object 
 24  Average Submitted Charge Amount                           100000 non-null  object 
 25  Average Medicare Payment Amount                           100000 non-null  object 
 26  Average Medicare Standardized Amount                      100000 non-null  object 
dtypes: float64(1), int64(2), object(24)
memory usage: 20.6+ MB
```

In [3]:
```
irrelevant_columns=['Entity Type of the Provider',
                    'Street Address 1 of the Provider',
                    'Street Address 2 of the Provider',
                    'Zip Code of the Provider',
                    'Medicare Participation Indicator',
                    'Place of Service',
                    'HCPCS Code',
                    'HCPCS Description',
                    'HCPCS Drug Indicator',
                    'Country Code of the Provider']

data=data.drop(columns=irrelevant_columns)
```

Columns that have no relevance in our assignment have been dropped

In [4]:
```
data.head()
```

Out[4]:

	index	National Provider Identifier	Last Name/Organization Name of the Provider	First Name of the Provider	Middle Initial of the Provider	Credentials of the Provider	Gender of the Provider	City of the Provider	State Code of the Provider	Provider Type	Number of Services	Number of Medicare Beneficiaries	Number of Distinct Medicare Beneficiary/Per Day Services
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	M.D.	F	SAINT LOUIS	MO	Internal Medicine	27	24	27
1	3354385	1346202256	JONES	WENDY	P	M.D.	F	FAYETTEVILLE	NC	Obstetrics & Gynecology	175	175	175
2	3001884	1306820956	DUROCHER	RICHARD	W	DPM	M	NORTH HAVEN	CT	Podiatry	32	13	32
3	7594822	1770523540	FULLARD	JASPER	NaN	MD	M	KANSAS CITY	MO	Internal Medicine	20	18	20
4	746159	1073627758	PERROTTI	ANTHONY	E	DO	M	JUPITER	FL	Internal Medicine	33	24	31

Data Preprocessing

In [5]:
```
# Merging the name columns into a single column
data['Full Name'] = data['First Name of the Provider'].fillna('') + ' ' + \
                    data['Middle Initial of the Provider'].fillna('') + ' ' + \
                    data['Last Name/Organization Name of the Provider'].fillna('')
data['Full Name'] = data['Full Name'].str.strip()

data = data.drop(columns=['Last Name/Organization Name of the Provider',
                          'First Name of the Provider',
                          'Middle Initial of the Provider'])

full_name_column = data.pop('Full Name')

data.insert(1, 'Full Name', full_name_column)


data.head()
```

Out[5]:

	index	Full Name	National Provider Identifier	Credentials of the Provider	Gender of the Provider	City of the Provider	State Code of the Provider	Provider Type	Number of Services	Number of Medicare Beneficiaries	Number of Distinct Medicare Beneficiary/Per Day Services	Average Medicare Allowed Amount	Average Submitted Charge Amount
0	8774979	SATYASREE UPADHYAYULA	1891106191	M.D.	F	SAINT LOUIS	MO	Internal Medicine	27	24	27	200.58777778	305.21111111
1	3354385	WENDY P JONES	1346202256	M.D.	F	FAYETTEVILLE	NC	Obstetrics & Gynecology	175	175	175	123.73	548.8
2	3001884	RICHARD W DUROCHER	1306820956	DPM	M	NORTH HAVEN	CT	Podiatry	32	13	32	90.65	155
3	7594822	JASPER FULLARD	1770523540	MD	M	KANSAS CITY	MO	Internal Medicine	20	18	20	3.5	5
4	746159	ANTHONY E PERROTTI	1073627758	DO	M	JUPITER	FL	Internal Medicine	33	24	31	26.52	40

A new column "Full Name" has been created to merge first name, middle name and last name

In [6]:
```
# Uniform format of credentials
data['Credentials of the Provider'] = data['Credentials of the Provider'].str.replace(r'\.', '', regex=True).str.upper()
data.head()
```

Out[6]:

	index	Full Name	National Provider Identifier	Credentials of the Provider	Gender of the Provider	City of the Provider	State Code of the Provider	Provider Type	Number of Services	Number of Medicare Beneficiaries	Number of Distinct Medicare Beneficiary/Per Day Services	Average Medicare Allowed Amount	Average Submitted Charge Amount
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	F	SAINT LOUIS	MO	Internal Medicine	27	24	27	200.58777778	305.21111111
1	3354385	WENDY P JONES	1346202256	MD	F	FAYETTEVILLE	NC	Obstetrics & Gynecology	175	175	175	123.73	548.8
2	3001884	RICHARD W DUROCHER	1306820956	DPM	M	NORTH HAVEN	CT	Podiatry	32	13	32	90.65	155
3	7594822	JASPER FULLARD	1770523540	MD	M	KANSAS CITY	MO	Internal Medicine	20	18	20	3.5	5
4	746159	ANTHONY E PERROTTI	1073627758	DO	M	JUPITER	FL	Internal Medicine	33	24	31	26.52	40

"Credentials of the Provider" column now follows a uniform format. Such that MD and M.D and M.D. are all treated as the same unit

Converting Object to Numeric Type

```
In [7]: numeric_columns = [
            'Number of Services',
            'Number of Medicare Beneficiaries',
            'Number of Distinct Medicare Beneficiary/Per Day Services',
            'Average Medicare Allowed Amount',
            'Average Submitted Charge Amount',
            'Average Medicare Payment Amount',
            'Average Medicare Standardized Amount'
        ]

        for column in numeric_columns:
            data[column] = pd.to_numeric(data[column], errors='coerce')


        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 15 columns):
 #   Column                                                    Non-Null Count   Dtype
---  ------                                                    --------------   -----
 0   index                                                     100000 non-null  int64
 1   Full Name                                                 100000 non-null  object
 2   National Provider Identifier                              100000 non-null  int64
 3   Credentials of the Provider                               92791 non-null   object
 4   Gender of the Provider                                    95746 non-null   object
 5   City of the Provider                                      100000 non-null  object
 6   State Code of the Provider                                100000 non-null  object
 7   Provider Type                                             100000 non-null  object
 8   Number of Services                                        97347 non-null   float64
 9   Number of Medicare Beneficiaries                          99595 non-null   float64
 10  Number of Distinct Medicare Beneficiary/Per Day Services  98500 non-null   float64
 11  Average Medicare Allowed Amount                           99255 non-null   float64
 12  Average Submitted Charge Amount                           93277 non-null   float64
 13  Average Medicare Payment Amount                           99534 non-null   float64
 14  Average Medicare Standardized Amount                      99530 non-null   float64
dtypes: float64(7), int64(2), object(6)
memory usage: 11.4+ MB
```

Looking for Missing Values and imputing them with Mean

```
In [8]: # missing values
        print(data.isnull().sum())
```

```
index                                                          0
Full Name                                                      0
National Provider Identifier                                   0
Credentials of the Provider                                 7209
Gender of the Provider                                      4254
City of the Provider                                           0
State Code of the Provider                                     0
Provider Type                                                  0
Number of Services                                          2653
Number of Medicare Beneficiaries                             405
Number of Distinct Medicare Beneficiary/Per Day Services    1500
Average Medicare Allowed Amount                              745
Average Submitted Charge Amount                             6723
Average Medicare Payment Amount                              466
Average Medicare Standardized Amount                         470
dtype: int64
```

```
In [9]: # Imputation of numeric missing values with mean
        data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].mean())

        print(data.isnull().sum())
```

```
index                                                          0
Full Name                                                      0
National Provider Identifier                                   0
Credentials of the Provider                                 7209
Gender of the Provider                                      4254
City of the Provider                                           0
State Code of the Provider                                     0
Provider Type                                                  0
Number of Services                                             0
Number of Medicare Beneficiaries                               0
Number of Distinct Medicare Beneficiary/Per Day Services       0
Average Medicare Allowed Amount                                0
Average Submitted Charge Amount                                0
Average Medicare Payment Amount                                0
Average Medicare Standardized Amount                           0
dtype: int64
```

Imputation of categorical columns with mode

In [10]:

```
categorical_columns = ['Credentials of the Provider',
                       'Gender of the Provider',
                       'City of the Provider',
                      'State Code of the Provider']

for column in categorical_columns:
    data[column].fillna(data[column].mode()[0], inplace=True)

print(data.isnull().sum())
```

```
index                                                       0
Full Name                                                   0
National Provider Identifier                                0
Credentials of the Provider                                 0
Gender of the Provider                                      0
City of the Provider                                        0
State Code of the Provider                                  0
Provider Type                                               0
Number of Services                                          0
Number of Medicare Beneficiaries                            0
Number of Distinct Medicare Beneficiary/Per Day Services    0
Average Medicare Allowed Amount                             0
Average Submitted Charge Amount                             0
Average Medicare Payment Amount                             0
Average Medicare Standardized Amount                        0
dtype: int64
```

Looking for Duplicate Values

In [11]:

```
# Check for duplicates
print(data.duplicated().sum())
```

```
0
```

In [12]:

```
data.head()
```

Out[12]:

	index	Full Name	National Provider Identifier	Credentials of the Provider	Gender of the Provider	City of the Provider	State Code of the Provider	Provider Type	Number of Services	Number of Medicare Beneficiaries	Number of Distinct Medicare Beneficiary/Per Day Services	Average Medicare Allowed Amount	Average Submitted Charge Amount	I
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	F	SAINT LOUIS	MO	Internal Medicine	27.0	24.0	27.0	200.587778	305.211111	15
1	3354385	WENDY P JONES	1346202256	MD	F	FAYETTEVILLE	NC	Obstetrics & Gynecology	175.0	175.0	175.0	123.730000	548.800000	11
2	3001884	RICHARD W DUROCHER	1306820956	DPM	M	NORTH HAVEN	CT	Podiatry	32.0	13.0	32.0	90.650000	155.000000	6
3	7594822	JASPER FULLARD	1770523540	MD	M	KANSAS CITY	MO	Internal Medicine	20.0	18.0	20.0	3.500000	5.000000	
4	746159	ANTHONY E PERROTTI	1073627758	DO	M	JUPITER	FL	Internal Medicine	33.0	24.0	31.0	26.520000	40.000000	1

Encoding some Categorical Columns using Frequency Encoder

In [13]:

```
def frequency_encode(df, columns):
    for column in columns:
        freq_encoding = df[column].value_counts() / len(df)
        new_column_name = column + '_Freq'
        df.insert(df.columns.get_loc(column) + 1, new_column_name, df[column].map(freq_encoding))
    return df

columns_to_encode=['Credentials of the Provider',
                   'Gender of the Provider',
                   'Provider Type',
                  'State Code of the Provider']

data = frequency_encode(data, columns_to_encode)

data.head()
```

Out[13]:

	index	Full Name	National Provider Identifier	Credentials of the Provider	Credentials of the Provider_Freq	Gender of the Provider	Gender of the Provider_Freq	City of the Provider	State Code of the Provider	State Code of the Provider_Freq	Provider Type	Provider Type_Freq	Number of Services	E
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	0.73827	F	0.29105	SAINT LOUIS	MO	0.01997	Internal Medicine	0.11366	27.0	
1	3354385	WENDY P JONES	1346202256	MD	0.73827	F	0.29105	FAYETTEVILLE	NC	0.03725	Obstetrics & Gynecology	0.01028	175.0	
2	3001884	RICHARD W DUROCHER	1306820956	DPM	0.01915	M	0.70895	NORTH HAVEN	CT	0.01403	Podiatry	0.02027	32.0	
3	7594822	JASPER FULLARD	1770523540	MD	0.73827	M	0.70895	KANSAS CITY	MO	0.01997	Internal Medicine	0.11366	20.0	
4	746159	ANTHONY E PERROTTI	1073627758	DO	0.06176	M	0.70895	JUPITER	FL	0.07263	Internal Medicine	0.11366	33.0	

Performing Standardization on Numerical Columns

In [14]:

```
from sklearn.preprocessing import StandardScaler

standardization_columns=['Number of Services',
                         'Number of Medicare Beneficiaries',
                         'Number of Distinct Medicare Beneficiary/Per Day Services',
                         'Average Medicare Allowed Amount',
                         'Average Submitted Charge Amount',
                         'Average Medicare Payment Amount',
                         'Average Medicare Standardized Amount',
                         'Credentials of the Provider_Freq',
                         'Gender of the Provider_Freq',
                         'State Code of the Provider_Freq' ]

# Standardization
standard_scaler = StandardScaler()
data[standardization_columns] = standard_scaler.fit_transform(data[standardization_columns])


data_copy=data.copy()

print("Standardized DataFrame:")
data.head()
```

```
Standardized DataFrame:
```

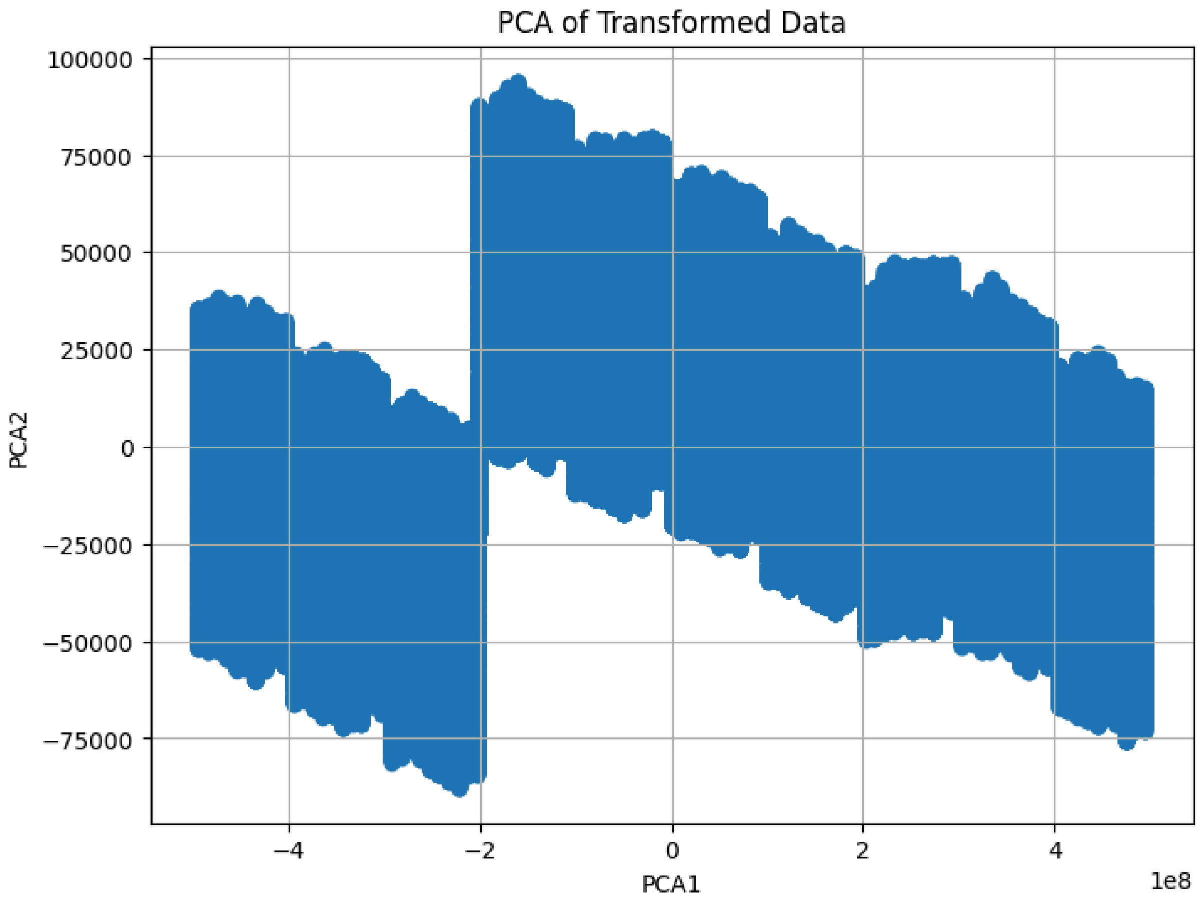
Out[14]:

	index	Full Name	National Provider Identifier	Credentials of the Provider	Credentials of the Provider_Freq	Gender of the Provider	Gender of the Provider_Freq	City of the Provider	State Code of the Provider	State Code of the Provider_Freq	Provider Type	Provider Type_Freq	Number of Services
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	0.594983	F	-1.560716	SAINT LOUIS	MO	-0.737342	Internal Medicine	0.11366	-0.497577
1	3354385	WENDY P JONES	1346202256	MD	0.594983	F	-1.560716	FAYETTEVILLE	NC	-0.004973	Obstetrics & Gynecology	0.01028	0.503328
2	3001884	RICHARD W DUROCHER	1306820956	DPM	-1.684316	M	0.640731	NORTH HAVEN	CT	-0.989093	Podiatry	0.02027	-0.463762
3	7594822	JASPER FULLARD	1770523540	MD	0.594983	M	0.640731	KANSAS CITY	MO	-0.737342	Internal Medicine	0.11366	-0.544917
4	746159	ANTHONY E PERROTTI	1073627758	DO	-1.549260	M	0.640731	JUPITER	FL	1.494517	Internal Medicine	0.11366	-0.456999

Dimensionality Reduction using PCA

In [15]:

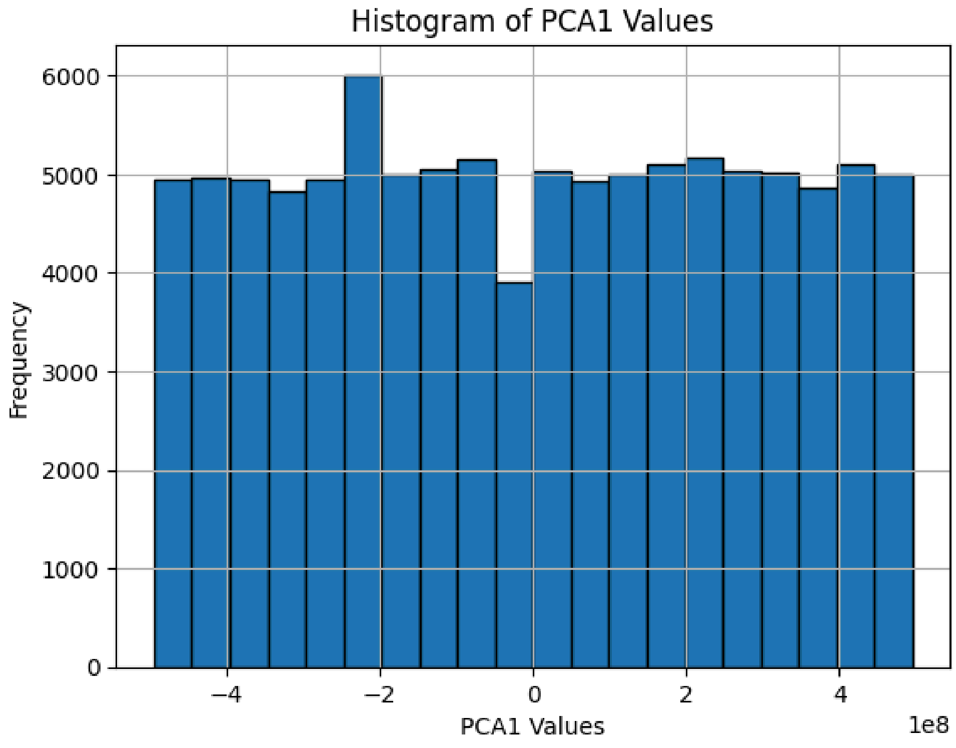
```
from sklearn.decomposition import PCA

df=data.copy()

# Imputation of categorical columns with mode
categorical_columns = ['Full Name',
                       'Credentials of the Provider',
                       'Gender of the Provider',
                       'City of the Provider',
                       'Provider Type',
                       'State Code of the Provider']

for column in df.columns:
    df[column].fillna(df[column].mode()[0], inplace=True)

df = df.drop(columns=categorical_columns)

pca = PCA(n_components=2)
pca_result = pca.fit_transform(df)

# DataFrame of PCA results
pca_df = pd.DataFrame(pca_result, columns=['PCA1', 'PCA2'])

# Scatter plot of PCA1 and PCA2
plt.figure(figsize=(8, 6))
plt.scatter(pca_df['PCA1'], pca_df['PCA2'])
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title('PCA of Transformed Data')
plt.grid(True)
plt.show()
```



The graph shows the scatterplot of the points of the first two principal components

In [16]:

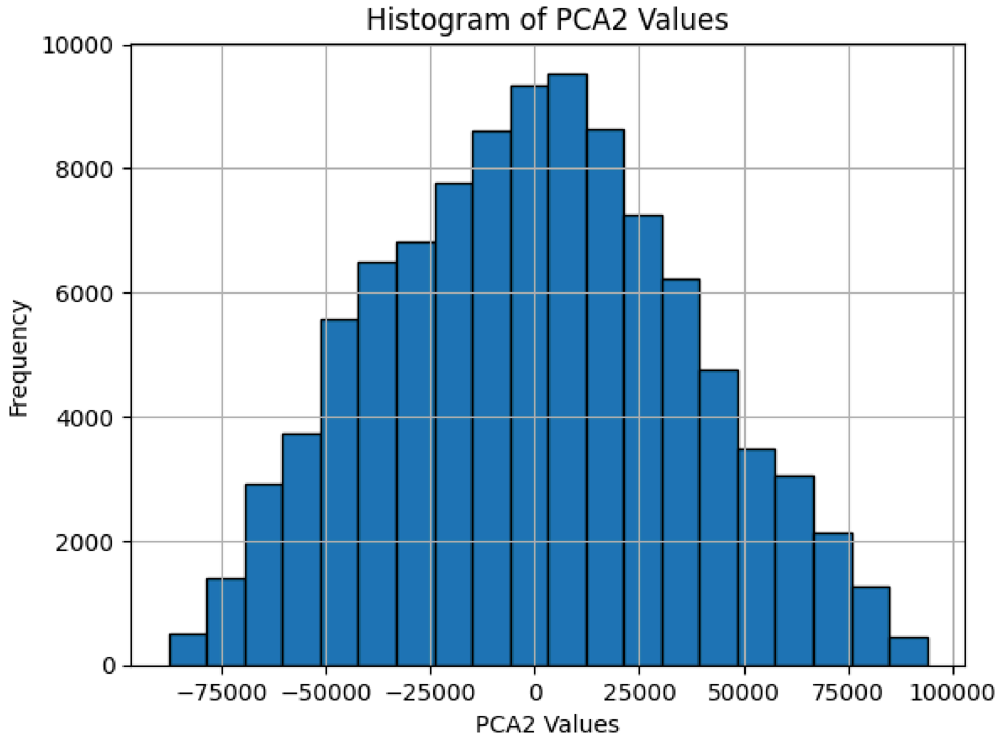
```
# Plot PCA1 as a histogram
plt.hist(pca_df['PCA1'], bins=20, edgecolor='black')
plt.xlabel('PCA1 Values')
plt.ylabel('Frequency')
plt.title('Histogram of PCA1 Values')
plt.grid(True)
plt.show()
```



The values of PCA1 are plotted into a histogram of 20 bins for better understanding of individual values.

In [17]:

```
# Plot PCA2 as a histogram
plt.hist(pca_df['PCA2'], bins=20, edgecolor='black')
plt.xlabel('PCA2 Values')
plt.ylabel('Frequency')
plt.title('Histogram of PCA2 Values')
plt.grid(True)
plt.show()
```



The values of PCA2 are plotted into a histogram of 20 bins for better understanding of individual values.

CLUSTERING

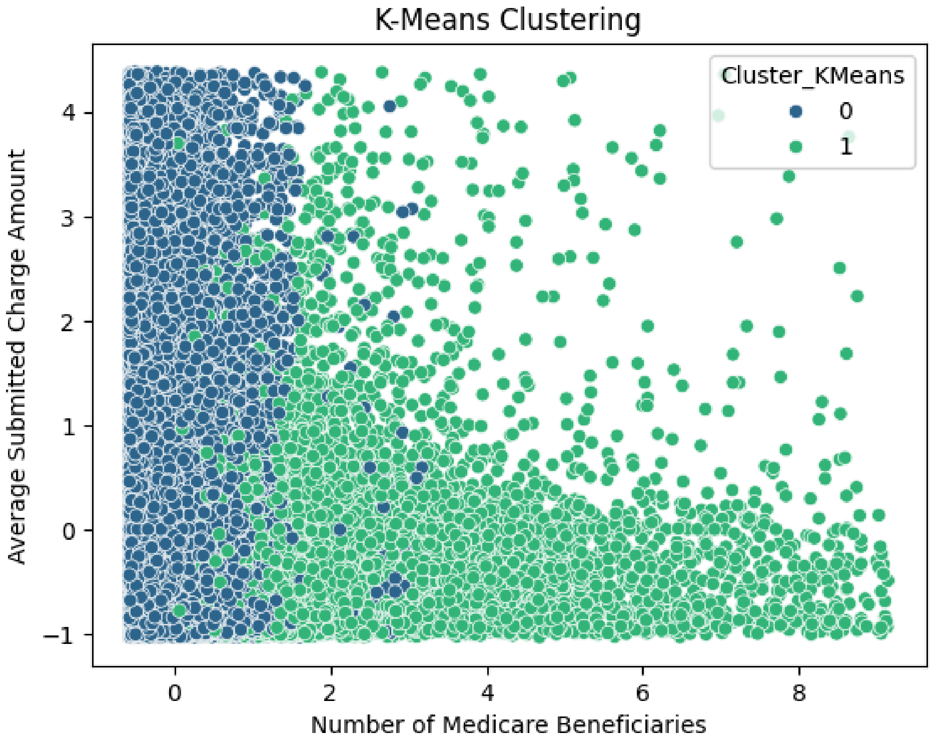
(i) K MEANS CLUSTERING

In [18]:

```
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score
```

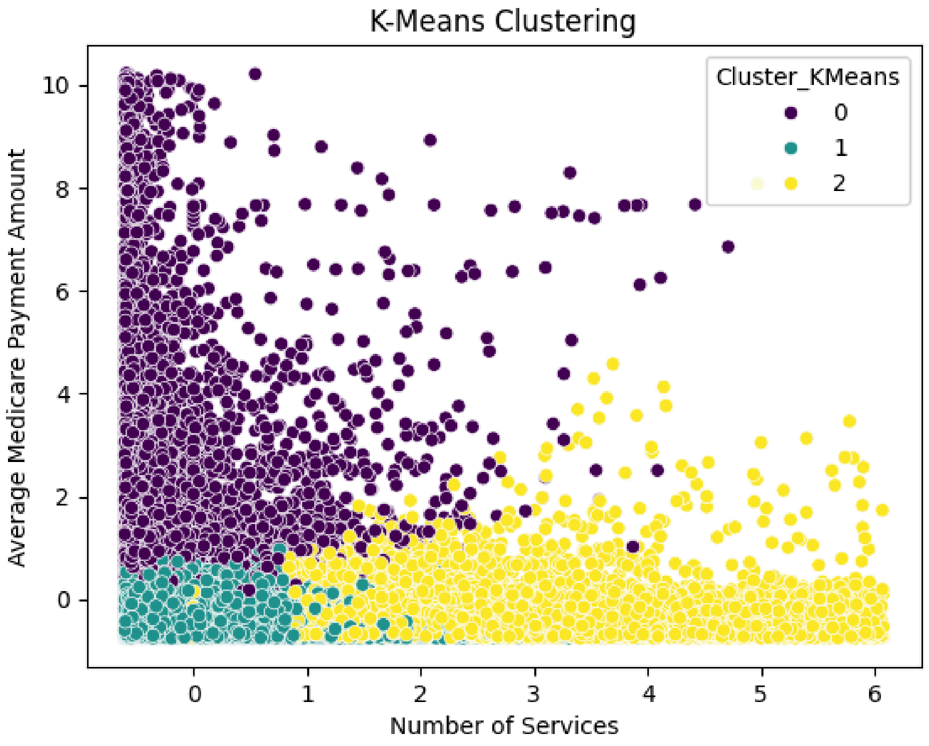
In [19]:

```
# Clustering using K-Means
kmeans = KMeans(n_clusters=2, random_state=42)
data['Cluster_KMeans'] = kmeans.fit_predict(data[numeric_columns])

sns.scatterplot(data=data, x='Number of Medicare Beneficiaries', y='Average Submitted Charge Amount',
                hue='Cluster_KMeans', palette='viridis', legend='full')
plt.title('K-Means Clustering')
plt.show()
```



Using K value as 2, we can see two very distinct clusters when we plot 'Average Submitted Charge Amount' against 'Number of Medicare Beneficiaries'

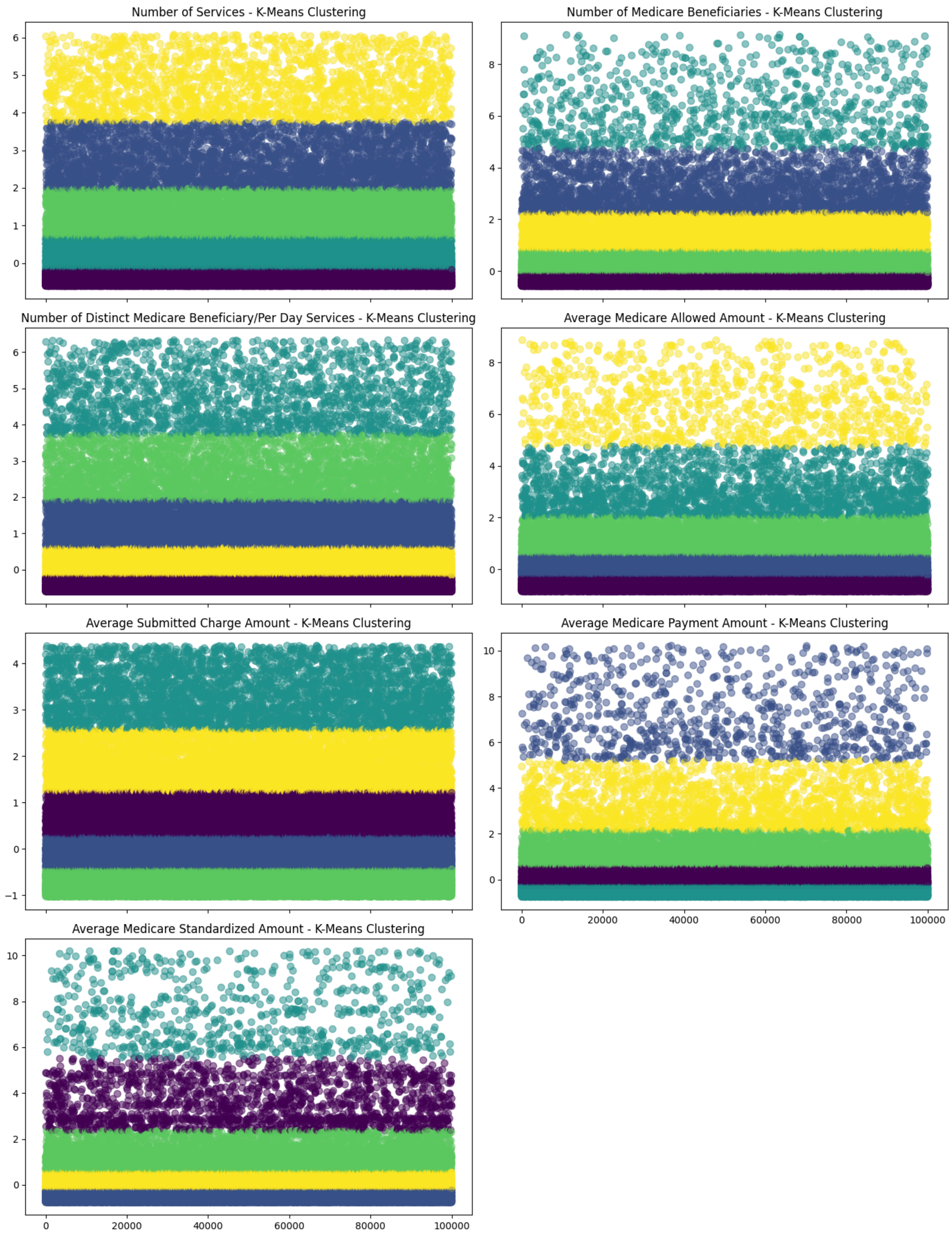
```
In [20]: # Clustering using K-Means
kmeans = KMeans(n_clusters=3, random_state=42)
data['Cluster_KMeans'] = kmeans.fit_predict(data[numeric_columns])
sns.scatterplot(data=data, x='Number of Services', y='Average Medicare Payment Amount', hue='Cluster_KMeans',
                palette='viridis', legend='full')
plt.title('K-Means Clustering')
plt.show()
```



Using K value as 3, three very distinct clusters can be seen when we plot "Average Medicare Payment Amount" against Number of Services

In [21]:

```
#Algoplot of K-Means

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np


k = 5

fig, axes = plt.subplots(4, 2, figsize=(14, 18))
fig.subplots_adjust(hspace=0.4, wspace=0.4)

axes = axes.flatten()

for i, col in enumerate(numeric_columns):
    # Perform K-Means clustering on the current column
    kmeans = KMeans(n_clusters=k, random_state=0)
    data['Cluster'] = kmeans.fit_predict(data[[col]])

    # Plot the column against its K-Means cluster assignments
    ax = axes[i]
    ax.scatter(data.index, data[col], c=data['Cluster'], s=50, alpha=0.5)
    ax.set_title(f'{col} - K-Means Clustering')

    if i < len(numeric_columns) - 2:
        ax.set_xticklabels([])

for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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



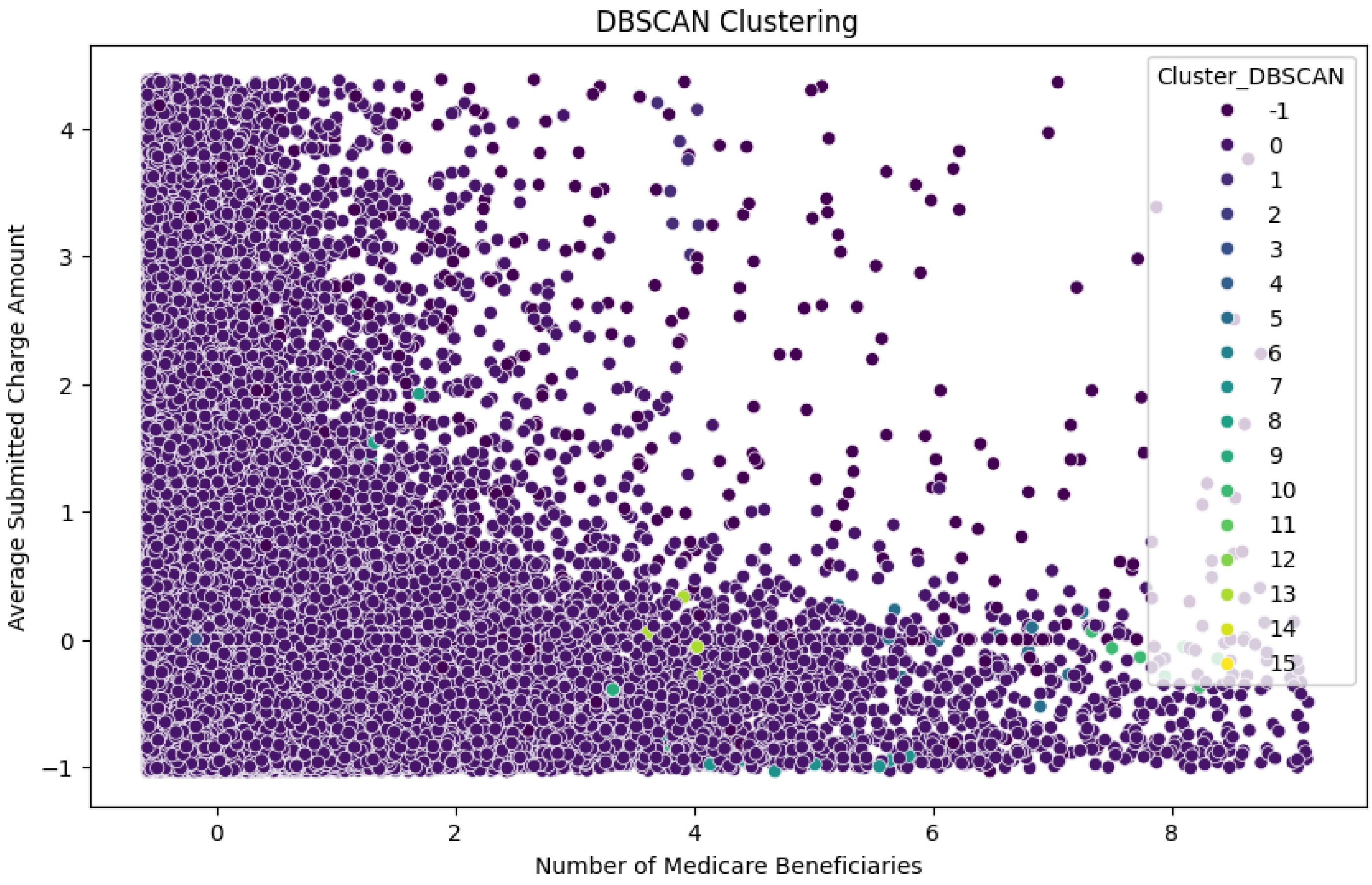
The above Algoplot of K Means Clustering shows the distribution of the data points across all the numeric columns, when K is set to 5

(ii) DB SCAN CLUSTERING

In [22]:

```
from sklearn.cluster import DBSCAN
# Clustering using DBSCAN
dbscan = DBSCAN(eps=0.7, min_samples=6)
data['Cluster_DBSCAN'] = dbscan.fit_predict(data[numeric_columns])

# Number of noise points
num_noise_points = (data['Cluster_DBSCAN'] == -1).sum()
print(f"Number of noise points: {num_noise_points}")

plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Number of Medicare Beneficiaries', y='Average Submitted Charge Amount',
                hue='Cluster_DBSCAN', palette='viridis', legend='full')
plt.title('DBSCAN Clustering')
plt.show()
```

```
Number of noise points: 788
```

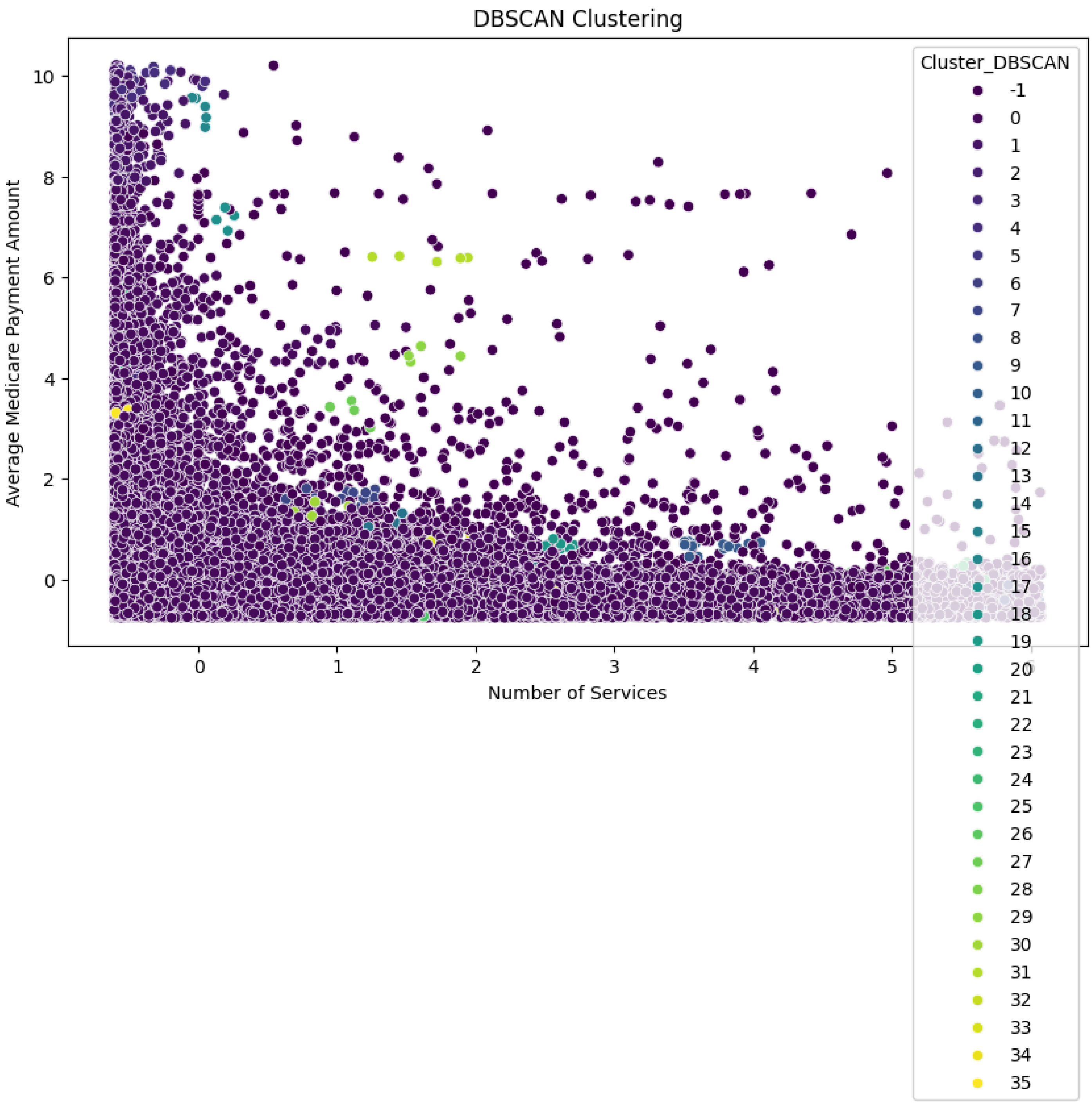


After setting the epsilon radius as 0.7 and minimum number of samples as 6, we found 303 noise points, and 17 clusters (-1 to 15)

In [23]:

```
dbscan = DBSCAN(eps=0.5, min_samples=4)
data['Cluster_DBSCAN'] = dbscan.fit_predict(data[numeric_columns])

num_noise_points = (data['Cluster_DBSCAN'] == -1).sum()
print(f"Number of noise points: {num_noise_points}")

plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Number of Services', y='Average Medicare Payment Amount',
                hue='Cluster_DBSCAN', palette='viridis', legend='full')
plt.title('DBSCAN Clustering')
plt.show()
```

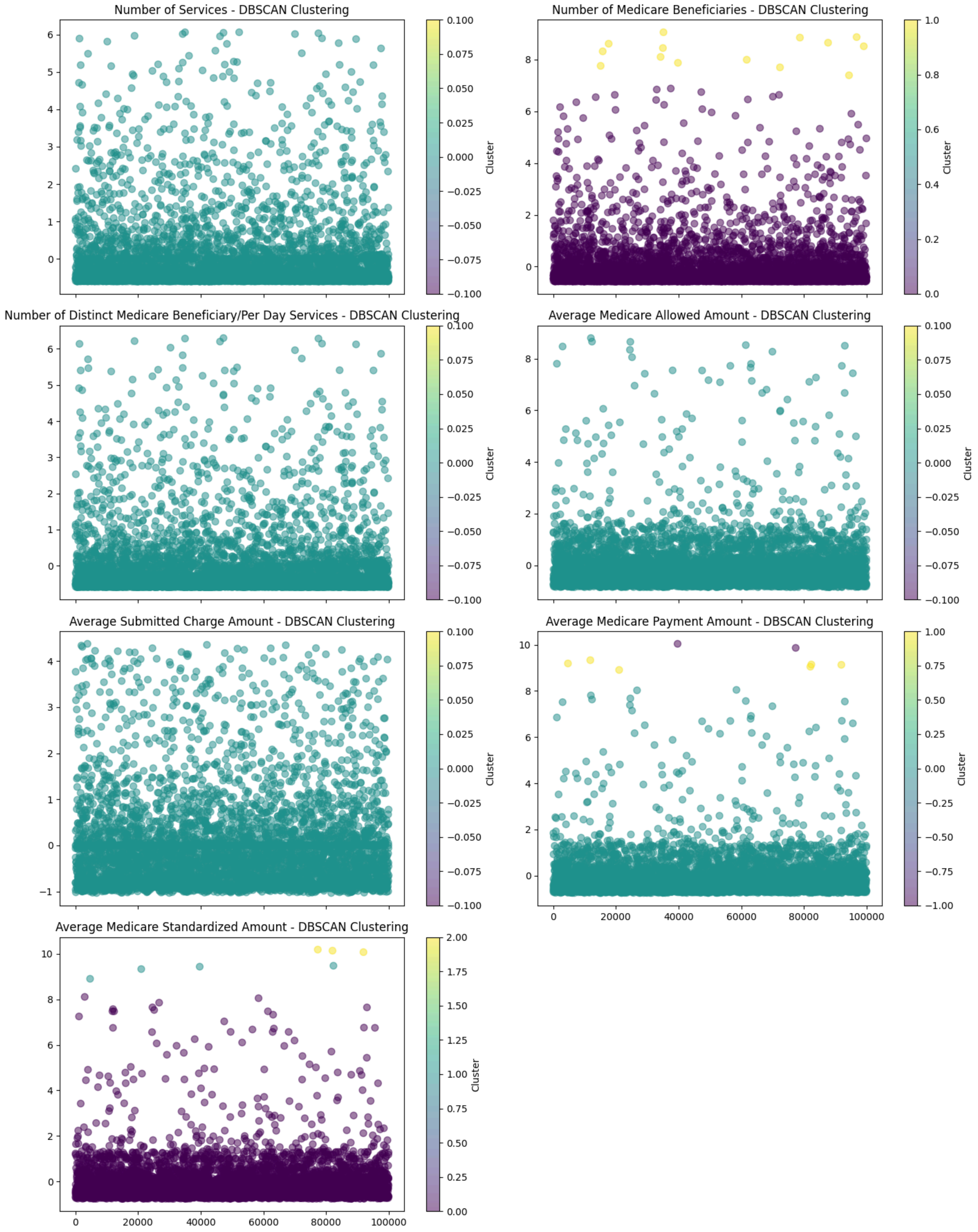
```
Number of noise points: 1395
```



After setting the epsilon radius as 0.5 and minimum number of samples as 4, we found 1395 noise points, and 37 clusters (-1 to 35)

In [24]:

```
#Algoplot of DBScan

eps = 0.5
min_samples = 3


data = data[numeric_columns].dropna()

data = data.sample(n=5000, random_state=42)

fig, axes = plt.subplots(4, 2, figsize=(14, 18))
fig.subplots_adjust(hspace=0.4, wspace=0.4)

axes = axes.flatten()

for i, col in enumerate(numeric_columns):
    # Perform DBSCAN clustering on the current column
    dbscan = DBSCAN(eps=eps, min_samples=min_samples)
    # Reshape data to 2D array for DBSCAN
    data_col = data[[col]].values.reshape(-1, 1)
    data['Cluster'] = dbscan.fit_predict(data_col)

    # Plot the column against its DBSCAN cluster assignments
    ax = axes[i]
    scatter = ax.scatter(data.index, data[col], c=data['Cluster'], cmap='viridis', s=50, alpha=0.5)
    ax.set_title(f'{col} - DBSCAN Clustering')

    cbar = plt.colorbar(scatter, ax=ax)
    cbar.set_label('Cluster')

    if i < len(numeric_columns) - 2:
        ax.set_xticklabels([])

for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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



Type *Markdown* and LaTeX: α^2

The above Algoplot of DBScan Clustering shows the distribution of the data points across all the numeric columns, when epsilon radius is set to 0.5 minimum number of samples is set to 3