MileStone2

Name- Om Late

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data = pd.read_csv("/content/Health1.csv")
data.head()
```

	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	Gende of th Provide
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	M.D.	
1	3354385	1346202256	JONES	WENDY	P	M.D.	
2	3001884	1306820956	DUROCHER	RICHARD	W	DPM	I
3	7594822	1770523540	FULLARD	JASPER	NaN	MD	I
4	746159	1073627758	PERROTTI	ANTHONY	E	DO	I

5 rows × 27 columns

```
# original data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67325 entries, 0 to 67324
Data columns (total 27 columns):
 #   Column                                                        Non-Null Count  Dtype
---  ------                                                        --------------  -----
 0   index                                                         67325 non-null  int64
 1   National Provider Identifier                                  67325 non-null  int64
 2   Last Name/Organization Name of the Provider               67325 non-null  object
 3   First Name of the Provider                                64477 non-null  object
 4   Middle Initial of the Provider                            47580 non-null  object
 5   Credentials of the Provider                               62477 non-null  object
 6   Gender of the Provider                                    64478 non-null  object
 7   Entity Type of the Provider                               67325 non-null  object
 8   Street Address 1 of the Provider                          67325 non-null  object
 9   Street Address 2 of the Provider                          27507 non-null  object
 10  City of the Provider                                      67325 non-null  object
 11  Zip Code of the Provider                                  67325 non-null  float6
 12  State Code of the Provider                                67324 non-null  object
 13  Country Code of the Provider                              67324 non-null  object
 14  Provider Type                                             67324 non-null  object
 15  Medicare Participation Indicator                          67324 non-null  object
 16  Place of Service                                          67324 non-null  object
 17  HCPCS Code                                                67324 non-null  object
 18  HCPCS Description                                         67324 non-null  object
 19  HCPCS Drug Indicator                                      67324 non-null  object
 20  Number of Services                                        67324 non-null  object
 21  Number of Medicare Beneficiaries                          67324 non-null  object
 22  Number of Distinct Medicare Beneficiary/Per Day Services  67324 non-null  object
 23  Average Medicare Allowed Amount                           67324 non-null  object
 24  Average Submitted Charge Amount                           67324 non-null  object
 25  Average Medicare Payment Amount                           67324 non-null  object
 26  Average Medicare Standardized Amount                      67324 non-null  object
dtypes: float64(1), int64(2), object(24)
memory usage: 13.9+ MB
```

```
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)
```

```
data.head()
```

	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	Gende of th Provide
0	8774979	1891106191	UPADHYAYULA	SATYASREE	NaN	M.D.	
1	3354385	1346202256	JONES	WENDY	P	M.D.	
2	3001884	1306820956	DUROCHER	RICHARD	W	DPM	I
3	7594822	1770523540	FULLARD	JASPER	NaN	MD	I
4	746159	1073627758	PERROTTI	ANTHONY	E	DO	I

```
# 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()
```

	index	Full Name	National Provider Identifier	Credentials of the Provider	Gender of the Provider	City of the Provider	State Code of the Provider
0	8774979	SATYASREE UPADHYAYULA	1891106191	M.D.	F	SAINT LOUIS	MO
1	3354385	WENDY P JONES	1346202256	M.D.	F	FAYETTEVILLE	NC
2	3001884	RICHARD W DUROCHER	1306820956	DPM	M	NORTH HAVEN	CT
3	7594822	JASPER FULLARD	1770523540	MD	M	KANSAS CITY	MO
4	746159	ANTHONY E PERROTTI	1073627758	DO	M	JUPITER	FL

```
# Uniform format of credentials
data['Credentials of the Provider'] = data['Credentials of the Provider'].str.replace(r'\
data.head()
```

	index	Full Name	National Provider Identifier	Credentials of the Provider	Gender of the Provider	City of the Provider	State Code of the Provider
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	F	SAINT LOUIS	MO
1	3354385	WENDY P JONES	1346202256	MD	F	FAYETTEVILLE	NC
2	3001884	RICHARD W DUROCHER	1306820956	DPM	M	NORTH HAVEN	CT
3	7594822	JASPER FULLARD	1770523540	MD	M	KANSAS CITY	MO
4	746159	ANTHONY E PERROTTI	1073627758	DO	M	JUPITER	FL

Converting Object to Numeric

```
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: 67325 entries, 0 to 67324
Data columns (total 15 columns):
 #   Column                                                      Non-Null Count  Dtype
---  ------                                                      --------------  -----
 0   index                                                       67325 non-null  int64
 1   Full Name                                                   67325 non-null  object
 2   National Provider Identifier                                67325 non-null  int64
 3   Credentials of the Provider                                 62477 non-null  object
 4   Gender of the Provider                                      64478 non-null  object
 5   City of the Provider                                     67325 non-null  object
 6   State Code of the Provider                               67324 non-null  object
 7   Provider Type                                            67324 non-null  object
 8   Number of Services                                       65519 non-null  float6
 9   Number of Medicare Beneficiaries                         67039 non-null  float6
 10  Number of Distinct Medicare Beneficiary/Per Day Services  66316 non-null  float6
 11  Average Medicare Allowed Amount                          66821 non-null  float6
 12  Average Submitted Charge Amount                          62847 non-null  float6
 13  Average Medicare Payment Amount                          67004 non-null  float6
 14  Average Medicare Standardized Amount                     67006 non-null  float6
dtypes: float64(7), int64(2), object(6)
memory usage: 7.7+ MB
```

```
# missing values
print(data.isnull().sum())
```

```
index                                                       0
Full Name                                                   0
National Provider Identifier                                0
Credentials of the Provider                              4848
Gender of the Provider                                   2847
City of the Provider                                        0
State Code of the Provider                                  1
Provider Type                                               1
Number of Services                                       1806
Number of Medicare Beneficiaries                          286
Number of Distinct Medicare Beneficiary/Per Day Services    1009
Average Medicare Allowed Amount                           504
Average Submitted Charge Amount                          4478
Average Medicare Payment Amount                           321
Average Medicare Standardized Amount                      319
dtype: int64
```

```
# 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                              4848
Gender of the Provider                                   2847
City of the Provider                                        0
State Code of the Provider                                  1
Provider Type                                               1
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
```

```
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                                               1
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
```

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

```
0
```

```
data.head()
```

	index	Full Name	National Provider Identifier	Credentials of the Provider	Gender of the Provider	City of the Provider	State Code of the Provider
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	F	SAINT LOUIS	MO
1	3354385	WENDY P JONES	1346202256	MD	F	FAYETTEVILLE	NC
2	3001884	RICHARD W DUROCHER	1306820956	DPM	M	NORTH HAVEN	CT
3	7594822	JASPER FULLARD	1770523540	MD	M	KANSAS CITY	MO
4	746159	ANTHONY E PERROTTI	1073627758	DO	M	JUPITER	FL

```
def frequency_encode(df, columns):
  for column in columns:
    freq_encoding = df[column].value_counts() / len(df)
    new_column_name = column + '_Freq'
    # Check if column exists before inserting
    if new_column_name not in df.columns:
        df.insert(df.columns.get_loc(column) + 1, new_column_name, df[column].map(freq_en
  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()
```

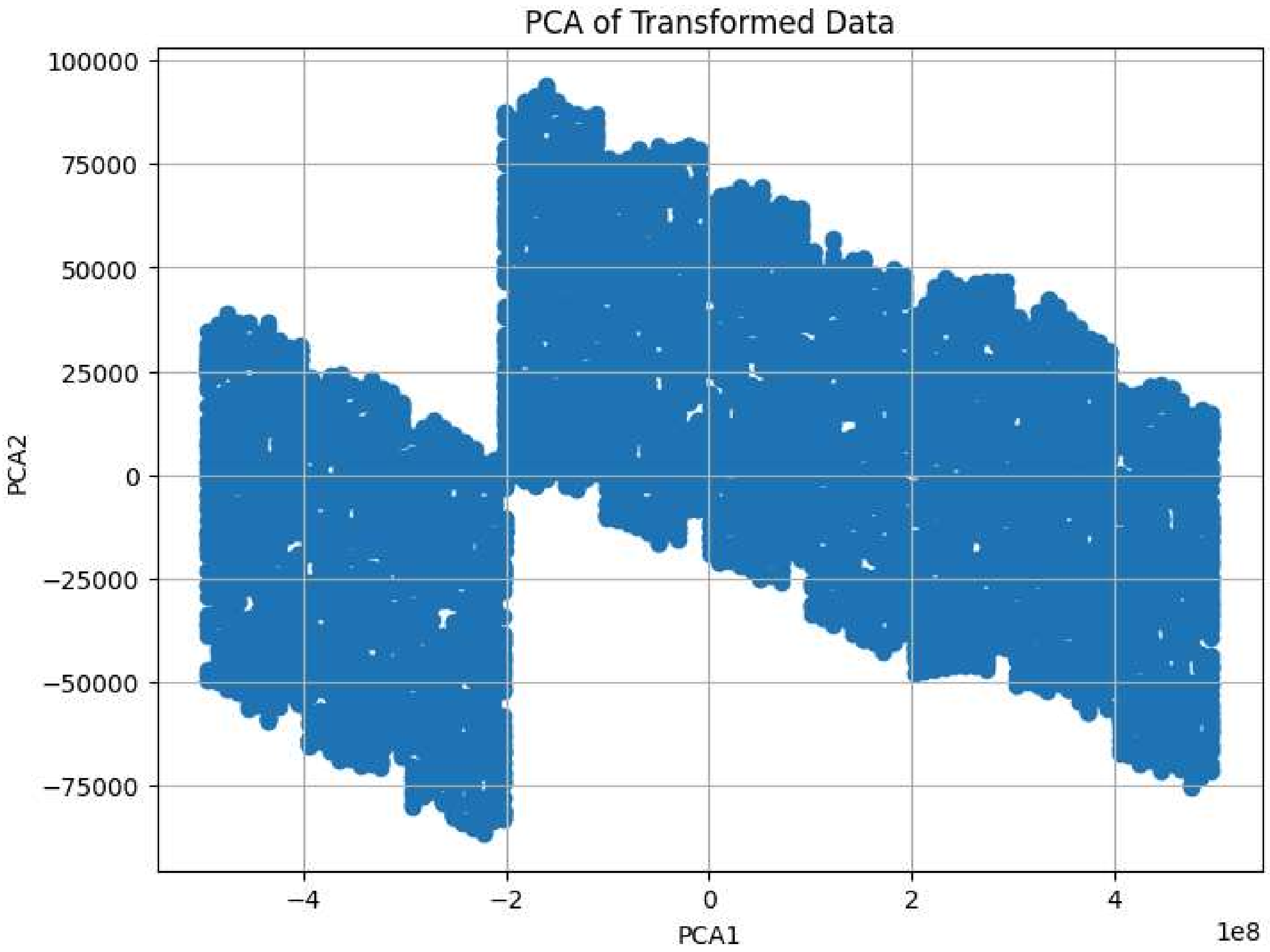
	index	Full Name	National Provider Identifier	Credentials of the Provider	Credentials of the Provider_Freq	Gender of the Provider	Gender o Provider_
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	0.735130	F	0.2
1	3354385	WENDY P JONES	1346202256	MD	0.735130	F	0.2
2	3001884	RICHARD W DUROCHER	1306820956	DPM	0.020160	M	0.7
3	7594822	JASPER FULLARD	1770523540	MD	0.735130	M	0.7
4	746159	ANTHONY E PERROTTI	1073627758	DO	0.064157	M	0.7

```
#Performing Standardization on Numerical Columns
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_column
data_copy=data.copy()
print("Standardized DataFrame:")
data.head()
```

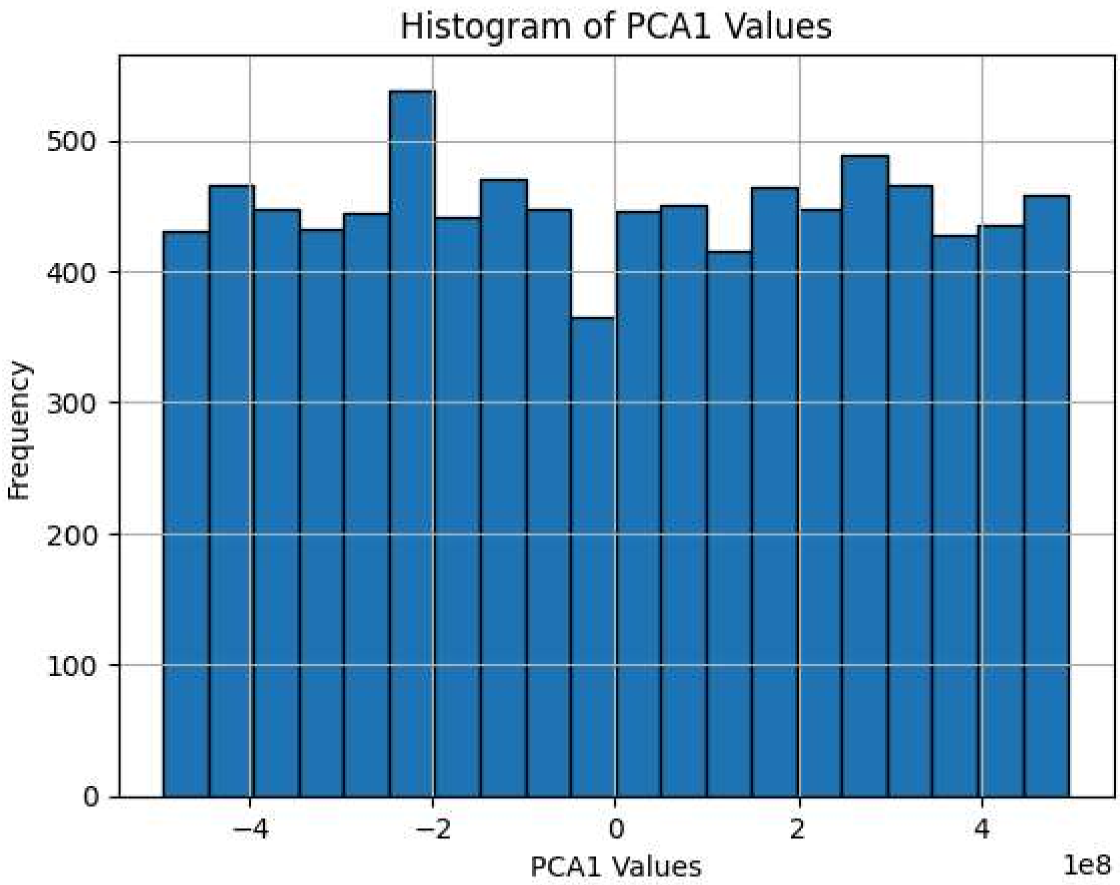
```
Standardized DataFrame:
```

	index	Full Name	National Provider Identifier	Credentials of the Provider	Credentials of the Provider_Freq	Gender of the Provider	Gender o Provider_
0	8774979	SATYASREE UPADHYAYULA	1891106191	MD	0.599764	F	-1.5
1	3354385	WENDY P JONES	1346202256	MD	0.599764	F	-1.5
2	3001884	RICHARD W DUROCHER	1306820956	DPM	-1.669524	M	0.6
3	7594822	JASPER FULLARD	1770523540	MD	0.599764	M	0.6
4	746159	ANTHONY E PERROTTI	1073627758	DO	-1.529881	M	0.6

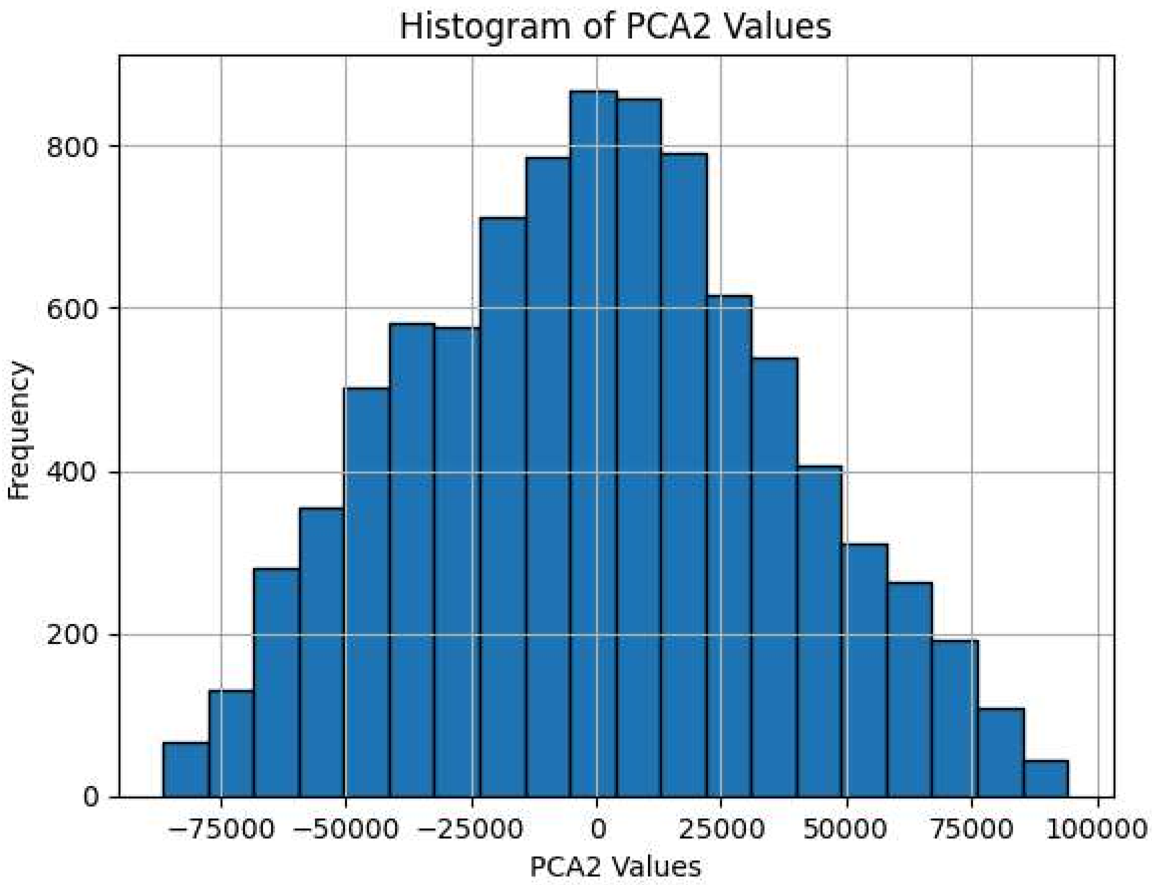
```
#Dimensionality Reduction using PCA
from sklearn.decomposition import PCA
df=data.copy()
# Imputation of categorical columns with mode
categorical_columns = ['Full Name',
                        'Credentials of the Provider',
                        'Gender of the Provider',
                        '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()
```



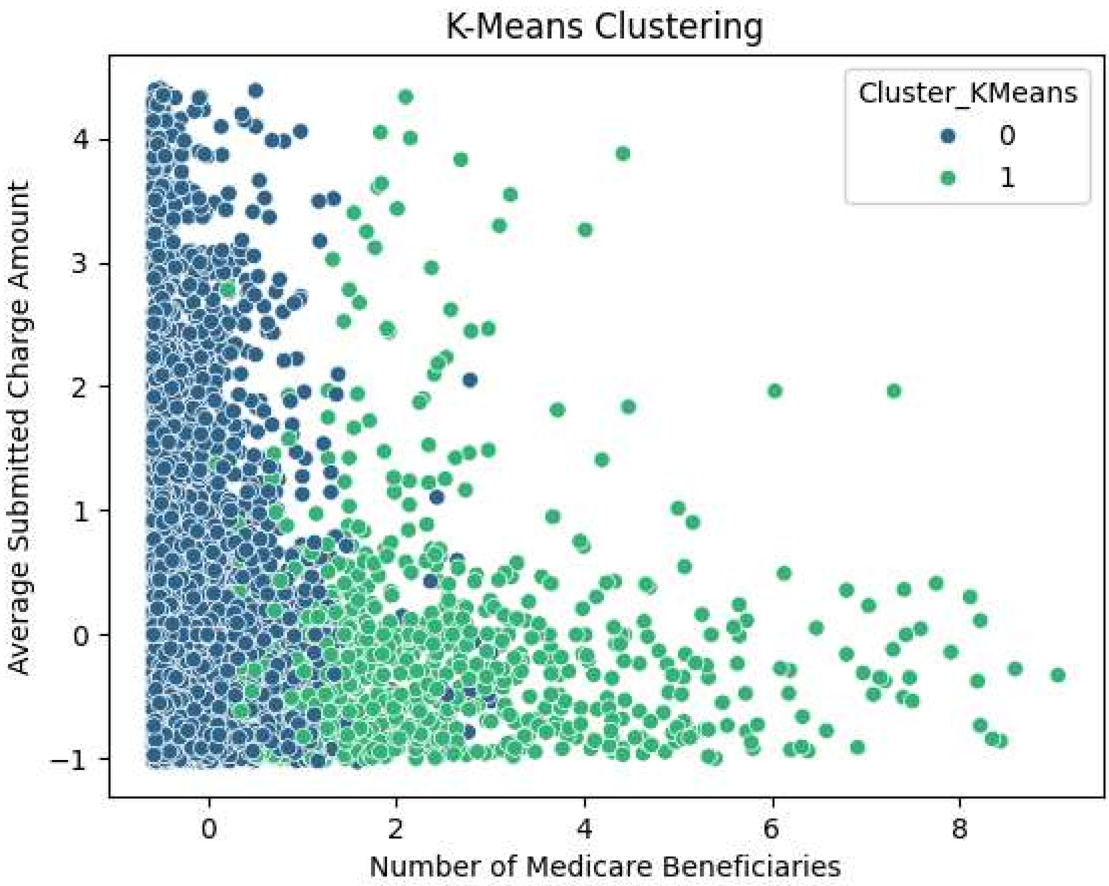
```
# Plot PCA1 as a histogram
plt.hist(pca_df['PCA1'], bins=20, edgecolor='black')
plt.xlabel('PCA1 Values')
plt.ylabel('Frequency')
plt.title('Histogram of PCA1 Values')
plt.grid(True)
plt.show()
```



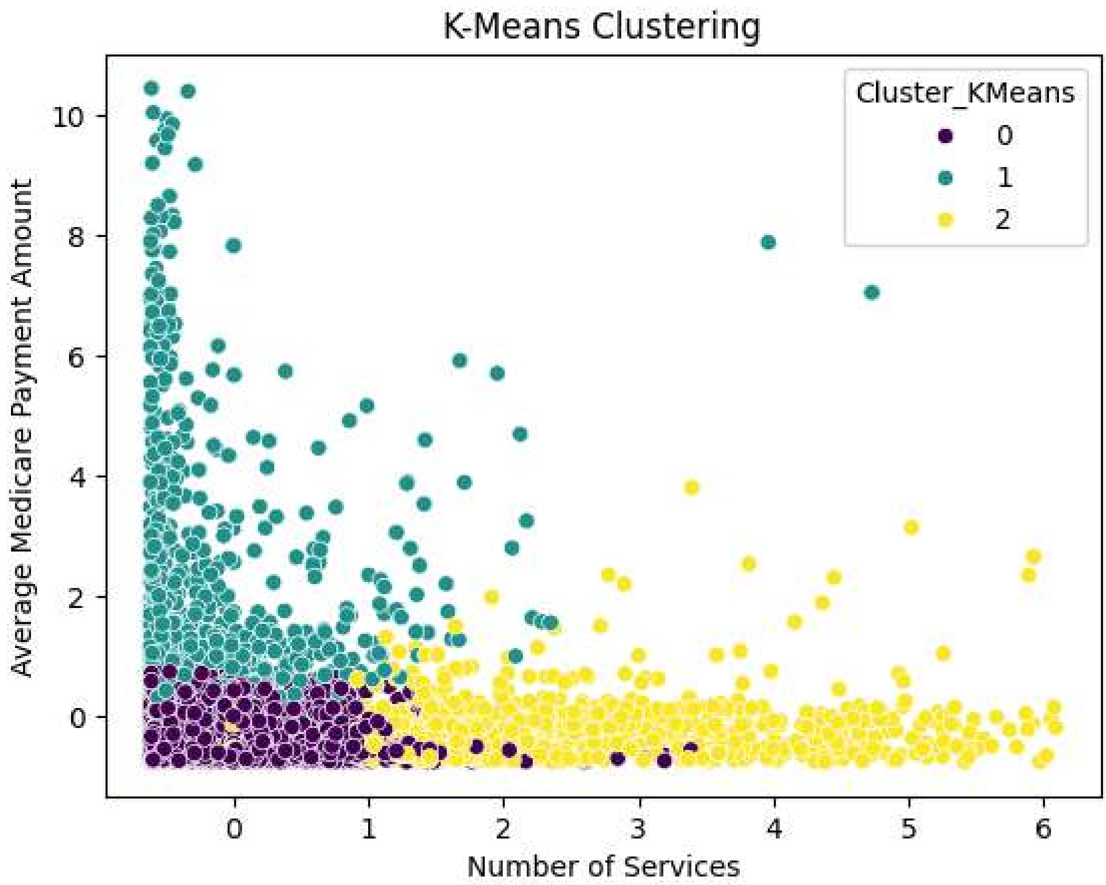
```
# Plot PCA2 as a histogram
plt.hist(pca_df['PCA2'], bins=20, edgecolor='black')
plt.xlabel('PCA2 Values')
plt.ylabel('Frequency')
plt.title('Histogram of PCA2 Values')
plt.grid(True)
plt.show()
```



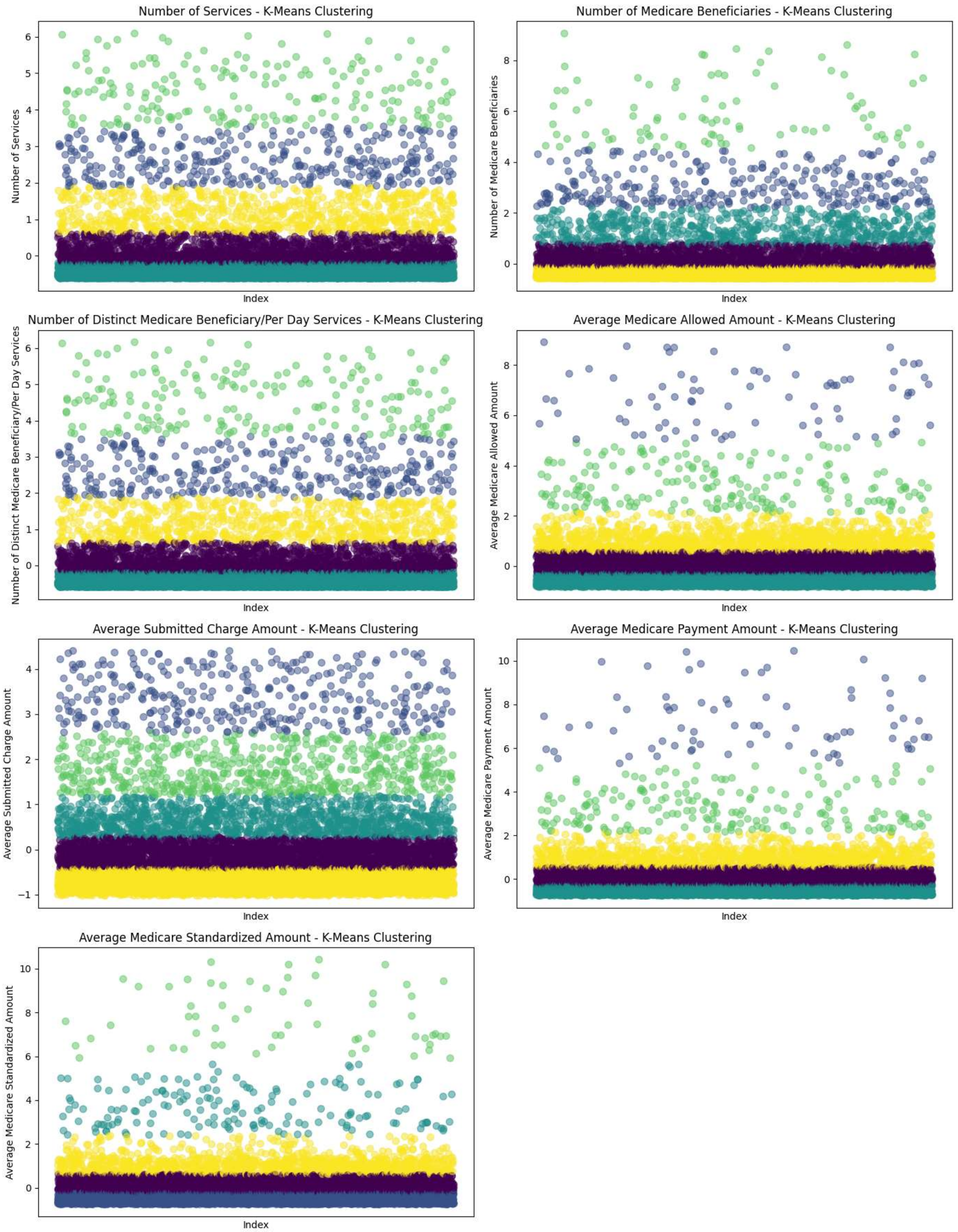
```
#CLUSTERING
#K MEANS CLUSTERING
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score
kmeans = KMeans(n_clusters=2, random_state=42)
data['Cluster_KMeans'] = kmeans.fit_predict(data[numeric_columns])
sns.scatterplot(data=data, x='Number of Medicare Beneficiaries', y='Average Submitted Cha
 hue='Cluster_KMeans', palette='viridis', legend='full')
plt.title('K-Means Clustering')
plt.show()
```



```
# 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', h
 palette='viridis', legend='full')
plt.title('K-Means Clustering')
plt.show()
```



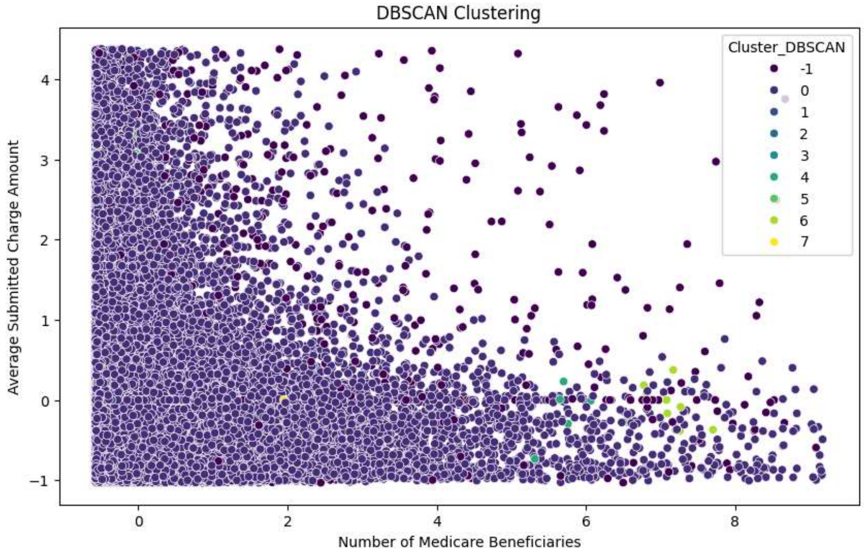
```
 #Algoplot of K-Means
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
k = 5
fig, axes = plt.subplots(4, 2, figsize=(14, 18))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
axes = axes.flatten()
for i, col in enumerate(numeric_columns):
    # 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')
    ax.set_xlabel('Index')
    ax.set_ylabel(col)
    ax.set_xticks([])
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()
```


Number of Services - K-Means Clustering
Number of Services
Index
6
5
4
3
2
1
0
Number of Medicare Beneficiaries - K-Means Clustering
Number of Medicare Beneficiaries
Index
8
6
4
2
0
Number of Distinct Medicare Beneficiary/Per Day Services - K-Means Clustering
Number of Distinct Medicare Beneficiary/Per Day Services
Index
6
5
4
3
2
1
0
Average Medicare Allowed Amount - K-Means Clustering
Average Medicare Allowed Amount
Index
8
6
4
2
0
Average Submitted Charge Amount - K-Means Clustering
Average Submitted Charge Amount
Index
4
3
2
1
0
−1
Average Medicare Payment Amount - K-Means Clustering
Average Medicare Payment Amount
Index
10
8
6
4
2
0
Average Medicare Standardized Amount - K-Means Clustering
Average Medicare Standardized Amount
Index
10
8
6
4
2
0

```
#DB SCAN CLUSTERING
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 Cha
 hue='Cluster_DBSCAN', palette='viridis', legend='full')
plt.title('DBSCAN Clustering')
plt.show()
```

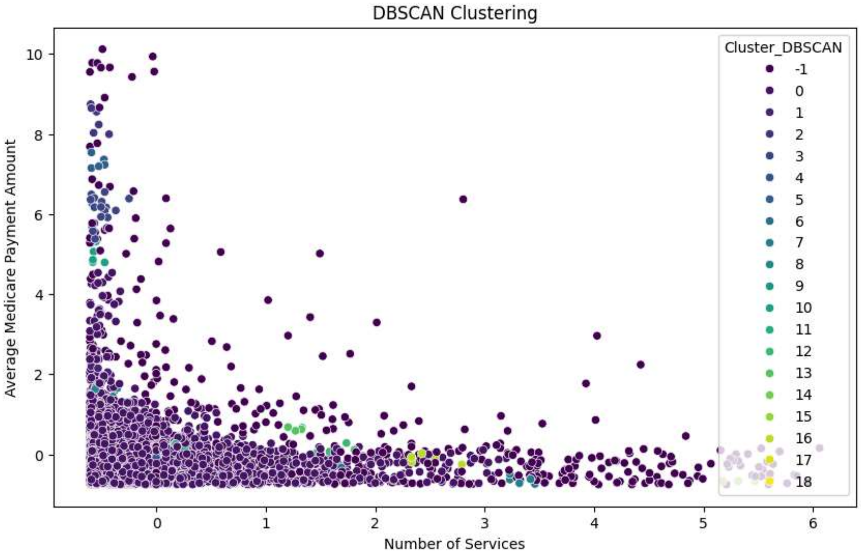
```
#DB SCAN CLUSTERING
from sklearn.cluster import DBSCAN
# Clustering using DBSCAN
dbscan = DBSCAN(eps=0.7, min_samples=6)
data['Cluster_DBSCAN'] = dbscan.fit_predict(data[numeric_columns])
# Number of noise points
num_noise_points = (data['Cluster_DBSCAN'] == -1).sum()
print(f"Number of noise points: {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: 703
```



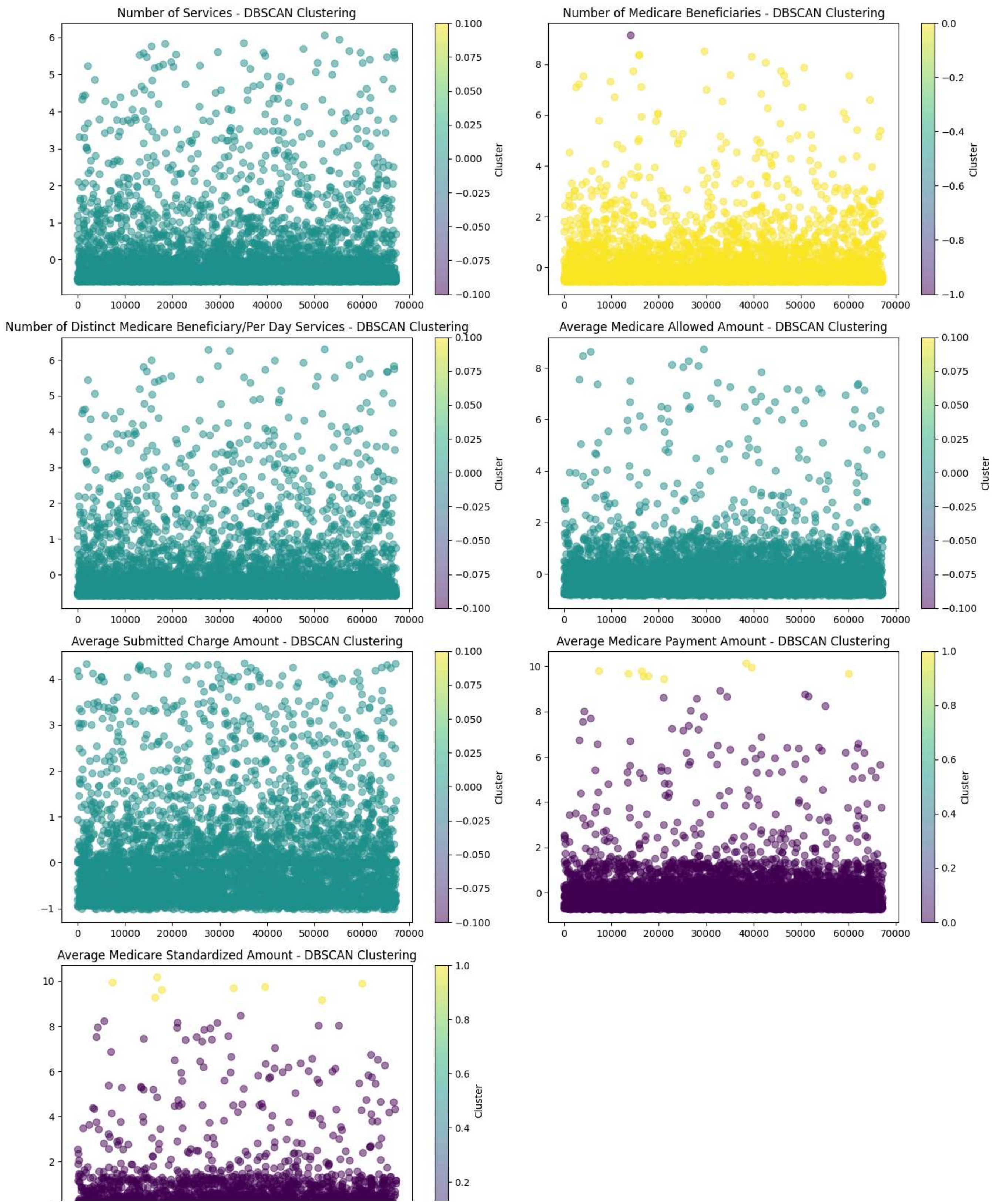
```
dbscan = DBSCAN(eps=0.5, min_samples=4)
data['Cluster_DBSCAN'] = dbscan.fit_predict(data[numeric_columns])
num_noise_points = (data['Cluster_DBSCAN'] == -1).sum()
print(f"Number of noise points: {num_noise_points}")
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Number of Services', y='Average Medicare Payment Amount',
 hue='Cluster_DBSCAN', palette='viridis', legend='full')
plt.title('DBSCAN Clustering')
plt.show()
```

```
Number of noise points: 445
```



```
#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()
```


Number of Services - DBSCAN Clustering
Cluster
0.100 0.075 0.050 0.025 0.000 −0.025 −0.050 −0.075 −0.100
Number of Medicare Beneficiaries - DBSCAN Clustering
Cluster
0.0 −0.2 −0.4 −0.6 −0.8 −1.0
Number of Distinct Medicare Beneficiary/Per Day Services - DBSCAN Clustering
Cluster
Average Medicare Allowed Amount - DBSCAN Clustering
Cluster
Average Submitted Charge Amount - DBSCAN Clustering
Cluster
Average Medicare Payment Amount - DBSCAN Clustering
Cluster
1.0 0.8 0.6 0.4 0.2 0.0
Average Medicare Standardized Amount - DBSCAN Clustering
Cluster
0 10000 20000 30000 40000 50000 60000 70000