Studying Access to Healthcare Patterns using Medicare Health Insurance Data

Nicole Rachel Koshy[1], Prudhvi Raj Terli[1], Mothi Gowtham Ashok Kumar[1], Sony Priya Surapaneni[1], Vineela Deepthi Oguri[1], Divya Gottimukkula[1], Santosh Bokka[1]

[1]Indiana University, Indianapolis, IN 46202, USA

nickoshy@iu.edu, pterli@iu.edu, moashok@iu.edu, sonsurap@iu.edu, voguri@iu.edu, digott@iu.edu, sanbokk@iu.edu

Abstract. This project aims to determine if there is a difference in the access and utilization of Medicare health insurance services at various geographic, demographic, and epidemiological levels. We use inpatient and outpatient data from 2018-2020 to determine differences in access to services before and during the COVID pandemic. Since Medicare is most commonly availed by older citizens, we also use Post-Acute care and Hospice data to study differences in access to services based on gender, chronic conditions, duration of utilization, and Medicare payments. This data can be used to understand the most common services availed by older patients and to examine if the differences in cost of care and access can be attributed to a few key factors.

Keywords: Medicare • CMS • inpatient • outpatient • post-acute care • hospice • skilled nursing facility • inpatient rehabilitation facility • long term care facility • clustering • healthcare • health insurance • data analysis • data visualization • SQL • python.

1. Project Scope

1.1 Introduction

Medicare is a federally financed health insurance program for US citizens over the age of 65 years and younger individuals with disabilities or end-stage renal disease (Mues 2017[1]). Since almost all individuals who are enrolled in Medicare do not leave the program until they die, the data captured from hospitals participating in the program provides a wealth of information about the key features of the health services being provided to the elderly population, the most common diseases afflicting the elderly, the demographic dynamics of the people who have access and benefit from this program. Some examples of this include how Medicare data was used to determine that hospitals being given incentives to reduce patient readmissions fared better than hospitals that were penalized when they failed to do so (Hoffman 2020[2]). Another study concluded that reducing a certain subset of low value services that offer negligible clinical benefit to patients but cumulatively account for significant spending in the program would serve as an effective cost saving measure (Reid 2021[3]).

Medicare provides a variety of different coverages that are split into 4 main parts - A to D. Part A covers Hospital Insurance which covers inpatient (IPS) care, skilled nursing facilities, hospice, and home health care. Part B covers Medical Insurance, including outpatient (OPD) services, home health, medical equipment, and preventive services. Part D provides prescription coverage while Part C, which has been added in recent years, refers to the Medicare Advantage plans that offers coverage from private institutions as an alternative to the original Medicare plans[4].

Post-Acute Care (PAC) and Hospice Services, which is covered under parts A and B, are of 5 major types:

1. Skilled Nursing Facilities (SNF): In-patient treatment and rehabilitation facility with trained medical professionals that include licensed nurses, physical and occupational therapists among others [5].
2. Home Health (HH) Services: Health care services that can be provided at a patient's home for an illness or injury [6.]
3. Hospice (HOS) care: Services that focus on end-of-life care which can be provided at home or in facilities that provide care, comfort and quality of life to terminal patients[7].

4. Inpatient Rehabilitation Facility (IRF): Independent rehabilitation centers in acute care hospitals for patients that require round the clock nursing care [8].
5. Long Term Care (LTC) Facility: Facilities that provide various types of services required for patients who are unable to live independently [9].

For the purposes of this project, we will be using Inpatient, outpatient, post-acute care and hospice service provider data available from the Center for Medicare and Medicaid services (CMS) to study access to healthcare services across the US as well as within the state of Indiana.

1.2 Aim

Our aim is to utilize this data to elucidate the type of facilities available, costs associated with services provided, and the patterns of healthcare services available for some of the commonly treated indications through this program. To study this we asked the following questions:

1. Has there been a change in the inpatient and outpatient procedures patients access pre-pandemic (2018-2019) compared to during the pandemic (2020) which is discernable at various geographic levels (City, State)?
2. What is the difference in cost for the most frequently used procedures across cities within a State or between neighboring states?
3. Is there a disparity in the availability of inpatient and outpatient procedures across various geographic levels (such as Transplants, Hip and Knee Replacements)?
4. Can we use Hospital rating criteria and access to care for various demographic populations (percentage of people able to access services by ethnicity, medical conditions, total visits and discharges) to segregate facilities of a specific type (Inpatient Rehabilitation Facilities (IRF), Skilled Nursing Facilities (SNF), Home Health (HH), Hospice (HOS) and Long-Term Care (LTC) hospitals) into distinct clusters?

Based on our research questions, our hypothesis is below:

To study the distribution of services across various geographic and demographic levels

Null Hypothesis:

- There is no difference in the types of services available to patients across different geographic levels (cities, states, and across different states).
- There is no difference in the types of services available to patients across different demographic levels (based on ethnicity, and disease conditions).

Alternative Hypothesis:

- There is a discernible difference in the types of services available to patients across different geographic levels (cities, states, across different states).
- There is a discernible difference in the types of services available to patients across different demographic levels (based on ethnicity, and disease conditions).

For Clustering Analysis

Null Hypothesis: There are no subsets in the data that are more like each other than the rest of the data i.e., Hospital facilities cannot be segregated based on overall ratings, demographic access to care data, and hospital encounter criteria.

Alternative Hypothesis: There are subsets in the data that have higher similarity to each other than the rest of the data i.e., Hospital facilities can be segregated based on overall ratings, demographic access to care data, and hospital encounter criteria.

1.3 Purpose

In the past, Medicare data has been used to build models to predict clinical outcomes (MacKay 2021[11]), primary care practices (Wingrove 2020[10]).

The purpose of our study is to use Medicare data to study the distribution of patient services across cities and areas in a single state (example: Indiana) or a larger geographical area. The results generated from this study could be used to understand the average cost of care for major classes of diseases, the accessibility of facilities within an area that provide the most common services required by patients enrolled in the Medicare program.

2. Methodology

2.1 Steps of the Project

The focus of our project involves examining distribution of different types of medical facilities across the US to determine if there is a difference in the healthcare facilities

available to citizens based on geography, demographics and chronic conditions. The tools used are Python Jupyter notebook and phpMyAdmin.
The 4 main stages of our project are:

1. Data Collection
2. Data Extraction and Storage
3. Data Analysis
4. Data Visualization

2.2 Original Team Members and Responsibilities

Our team consists of people from medical, biotechnology and engineering backgrounds. Our credentials and original set of responsibilities are listed below in Table 1.

TEAM MEMBER	BACKGROUND	RESPONSIBILITIES
Sony Priya Surapaneni	Bachelor of Dental surgery, New to SQL and Python	Data Understanding
Divya Gottimukkula	Bachelor of Pharmacy, New to SQL and Python	Data Extraction
Santosh Bokka	Bachelor of Dental surgery, New to SQL and Python	Data Cleaning
Vineela Deepthi Oguri	Bachelor of pharmacy, Master's in pharmacy, New to SQL and Python	Exploratory Data Analysis
Prudhvi Raj Terli	Bachelor of Dental surgery, New to SQL and Python	Exploratory Data Analysis
Mothi Gowtham Ashok Kumar	Bachelor of Engineering, Experienced in SQL, Tableau, and beginner in Python	Descriptive Statistics, Hypothesis Testing
Nicole Koshy	MSc Bioinformatics, MS Biotech with experience in SQL and beginner level in Python and R	Visualization, Model Design, and Evaluation, Project Proposal & Report

Table 1: Original Responsibilities of Team Members for This Project

2.3 Actual Contributions from Individual Team Members

Since our team contained several Health Informatics students who are new to SQL and Python, they focused on understanding the data and finding patterns in the data that are relevant to our project while others in our group focused on the SQL and python processing of the data. Below is the updated table of our responsibilities.

TEAM MEMBER	RESPONSIBILITIES
Sony Priya Surapaneni	Data Understanding, Exploratory Data Analysis, Project Presentation, Inferring conclusions, Proof reading report
Divya Gottimukkula	Data Understanding, Exploratory Data Analysis, Project Presentation, Proof reading report
Santosh Bokka	Data Understanding, Exploratory Data Analysis, Project Presentation
Vineela Deepthi Oguri	Data Understanding, Exploratory Data Analysis, Project Presentation
Prudhvi Raj Terli	Data Extraction, SQL Data Analysis, Exploratory Data Analysis, Inferring conclusions, Proof reading report
Mothi Gowtham Ashok Kumar	Data Extraction, Exploratory Data Analysis, Python Visualization, Proof reading report
Nicole Koshy	Cleaning, SQL Analysis, Python Visualization, Hypothesis Testing, Model Design, and Evaluation, Project Proposal & Report

Table 2: Revised Responsibilities of Team Members for this Project

2.4 Project Challenges

While most of us in the group have some background in the healthcare domain, one challenge many members faced was understanding the datasets since we chose to use multiple datasets for our analysis. This significantly increased the time needed to come up with relevant research questions.

Cleaning the data was also a challenge since one of our datasets had 130 columns of which a subset of 60-80 were relevant for each subset in which columns had to be cleaned individually to replace suppressed data which took us almost 2 full days to complete.

Finally, a major hurdle we faced was running queries in SQL and Python. Even though the capacity of our group database was eventually increased, due to the large sizes of our datasets and multiple teams working on the same limited infrastructure, it was very common for SQL queries to require a long time to run, python notebooks not getting saved with updates due to frequent loss of connection to the server when using VPN for remote access. These issues have persisted to the very last days of our project and have required us to redo changes in codes more than once before they could be successfully saved.

3. Data Collection

The following datasets available at The Centers for Medicare and Medicaid Services were used for this study. For normalization by population, 2020 data from the US Census Bureau was taken:

1. **Medicare Inpatient Hospitals** – by Provider and Service – 2018 to 2020
   https://data.cms.gov/provider-summary-by-type-of-service/medicare-inpatient-hospitals/medicare-inpatient-hospitals-by-provider-and-service
2. **Medicare Outpatient Hospitals** – by Provider and Service – 2018 to 2020
   https://data.cms.gov/provider-summary-by-type-of-service/medicare-outpatient-hospitals/medicare-outpatient-hospitals-by-provider-and-service
3. **Medicare Post-Acute Care and Hospice** – by Geography & Provider
   https://data.cms.gov/provider-summary-by-type-of-service/medicare-post-acute-care-and-hospice-by-geography-provider
4. **US Census Bureau Data** – Statewide and City level
   https://www.census.gov/data/tables/time-series/demo/popest/2020s-total-cities-and-towns.html#ds

The dataset consists of the following raw data files:

1. MUP_IHP_RY21_P02_V10_DY18_PrvSvc.csv
2. MUP_IHP_RY21_P02_V10_DY19_PrvSvc_0.csv
3. MUP_IHP_RY22_P02_V10_Dy20_Geo.csv
4. MUP_OHP_R20_P04_V10_D18_Prov_Svc.csv
5. MUP_OUT_RY21_P04_V10_DY19_Prov_Svc.csv

6. MUP_OUT_RY22_P04_V10_DY20_Prov_Svc.csv
7. Medicare_Post_Acute_Care_and_Hospice_by_Geography_and_Provider_2020.csv
8. PACSNF_2020v2.csv
9. PACHOS_2020v2.csv
10. PACIRF_2020V2.csv
11. PACLTC_2020V2.csv
12. PACHH_2020V2.csv
13. USSTATEPOP_2020.csv
14. CITY_POP_2020.csv

4. Data Extraction and Storage

4.1 Data Extraction

For Datasets 1-6 (Inpatient & Outpatient Data) all files were input into SQL in their original format.

For 13-14 (Census data), a state abbreviation column was added so that the data could be cross-referenced during analysis.
The data from Dataset 7 was split into 5 separate tables, one for each type of PAC facility. Each new table had only the attributes from the original table that were relevant for them. This was determined from the PACPUF_Data_Dictionary. Datasets 8-12 were created from this primary file.

The main attributes used for analysis for each dataset were:
Datasets 1-3: DRG_CD, DRG_DESC, Prv_City, Prv_State, Avg_Mdcr_Pyt_Amt, TOT_DSCHRGS

Datasets 4-6: APC_Cd, APC_Desc, PRV_City, Prv_State, Bene_Cnt, Avg_Mdcr_Pymt_Amt

For Dataset 8, 10, 11, 12 : PRVDR_ID, PRVDR_NAME, PRVDR_CITY, STATE, BENE_DSTNCT_CNT, TOT_EPSD_STAY_CNT, TOT_SRVC_DAYS, TOT_CHRG_AMT, TOT_ALOWD_AMT, TOT_MDCR_PYMT_AMT, TOT_MDCR_STDZD_PYMT_AMT, BENE_AVG_AGE, BENE_MALE_PCT, BENE_FEML_PCT, BENE_RACE_WHT_PCT, BENE_RACE_BLACK_PCT, BENE_RACE_API_PCT, BENE_RACE_HSPNC_PCT,

BENE_RACE_NATIND_PCT, BENE_RACE_OTHR_PCT,
BENE_AVG_RISK_SCRE, BENE_CC_AF_PCT, BENE_CC_ALZHMR_PCT,
BENE_CC_ASTHMA_PCT, BENE_CC_CNCR_PCT, BENE_CC_CHF_PCT,
BENE_CC_CKD_PCT, BENE_CC_COPD_PCT, BENE_CC_DPRSSN_PCT,
BENE_CC_DBTS_PCT, BENE_CC_IHD_PCT, BENE_CC_OPO_PCT,
BENE_CC_RAOA_PCT, BENE_CC_SZ_PCT, BENE_CC_STROK_PCT

For Dataset 9: PRVDR_ID, PRVDR_NAME, PRVDR_CITY, STATE,
BENE_DSTNCT_CNT, TOT_EPSD_STAY_CNT, TOT_SRVC_DAYS,
TOT_CHRG_AMT, TOT_ALOWD_AMT, TOT_MDCR_PYMT_AMT,
TOT_MDCR_STDZD_PYMT_AMT, BENE_MALE_PCT, BENE_FEML_PCT,
BENE_RACE_WHT_PCT, BENE_RACE_BLACK_PCT, BENE_RACE_API_PCT,
BENE_RACE_HSPNC_PCT, BENE_RACE_NATIND_PCT,
BENE_RACE_OTHR_PCT, BENE_AVG_RISK_SCRE,
BENE_PRMRY_DX_CNCR_PCT, BENE_PRMRY_DX_COPD_PCT,
BENE_PRMRY_DX_RSPRTRYFAILR_PCT, BENE_PRMRY_DX_DMNT_PCT,
BENE_PRMRY_DX_STROK_PCT, BENE_PRMRY_DX_CHF_PCT,
BENE_PRMRY_DX_HYPRTNSN_PCT,
BENE_PRMRY_DX_OTHRCRDVSCLR_PCT,
BENE_PRMRY_DX_INFCTN_PCT, BENE_PRMRY_DX_ORTHO_PCT,
BENE_PRMRY_DX_INJURY_PCT, BENE_PRMRY_DX_MTR_NRL_PCT,
BENE_PRMRY_DX_DBTS_PCT

For Dataset 13: State, State_Abbr, Pop_2020
For Dataset 14: CITY, STATE, POPEST2020

The description of these attributes is provided below:

1. For Inpatient and Outpatient Data – Inpatient/Outpatient Service code and description, Provider City and State, Total patient counts per service per provider, Average Medicare amount payable per service per provider.
2. For PAC and Hospice Datasets for SNFs, HH, IRF and LTCs - Provider details, Provider level episode and service details, Medicare payment details Patient counts by gender, ethnicity, chronic conditions.
3. For PAC and Hospice Datasets for Hospices - Provider details, Provider level episode and service details, Medicare payment details Patient counts by gender, ethnicity, and primarily diagnosed disease.
4. For Population Data – City and State population numbers

4.2 Data Cleaning

Data Cleaning was performed only for Datasets 8-12 that contained suppressed data:
* - Replaced with 0.0 (These represents counts of lower than 10)
** - Replaced with 0.8755 (These represents counts where more than 75% of the counts are associated with a single chronic condition for a provider)

This cleaning was applied to all chronic condition and primary diagnosis columns. (BENE_CC_" DISEASE"_PCT, BENE_PRMRY_DX_"DISEASE"_PCT)

The general script for the same is mentioned below:

CLEANING SCRIPT format:

```
update TABLE_NAME
set COLUMN_NAME = '0' where COLUMN_NAME = '*'

update TABLE_NAME
set COLUMN_NAME= '0.8755' where COLUMN_NAME = '**'
```

The cleaned datasets were imported into MariaDB using Python scripts or direct imports, depending on the file size.

4.3 Data Import

Files were imported into MariaDB by 3 methods:

1. Direct Import
   CSV files within the file size limit allowed for the database, were uploaded directly into MariaDB through the Import tab. The files imported by this method are mentioned below:
   a. PACIRF_2020V2.csv
   b. PACLTC_2020V2.csv
   c. PACHH_2020V2.csv
   d. USSTATEPOP_2020.csv
   e. CITY_POP_2020.csv

2. Import using Python script for creation and upload

CSV Files that were larger in size than the allowed upload limit, were imported into the database using a python script for creating the table and then loading the file into the empty table.
A sample script of the same is provided below:

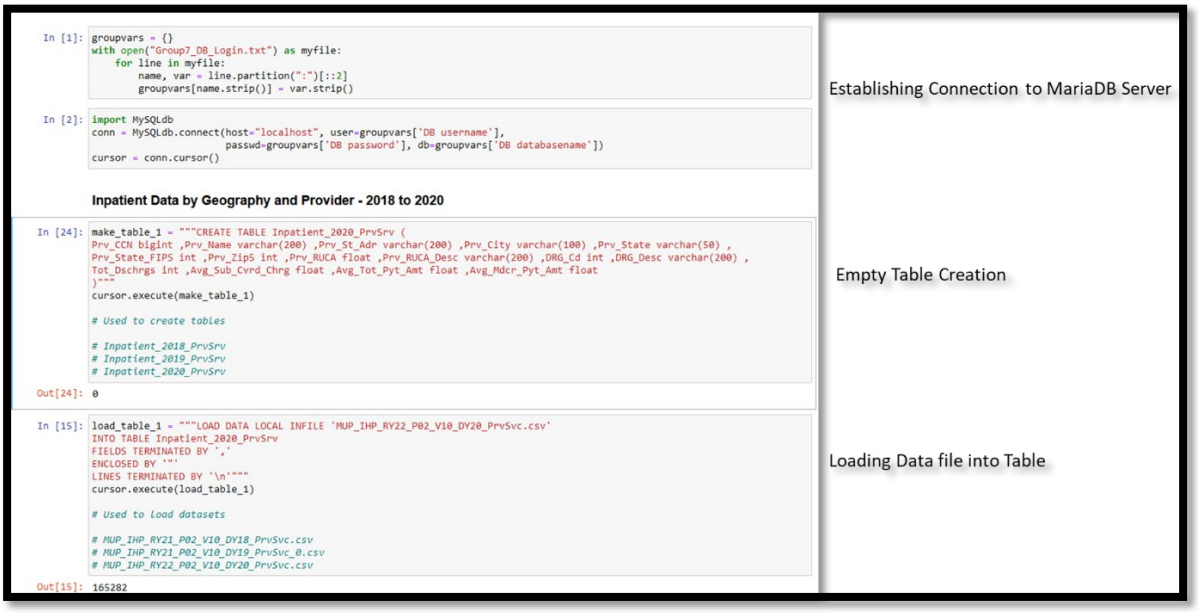

Figure 1: Table Creation and Data file upload script for importing the inpatient data files into the group database using Python Jupyter Notebook environment.

The tables imported by this method are mentioned below:

a. Inpatient_2018_PrvSrv
b. Inpatient_2019_PrvSrv
c. Inpatient_2020_PrvSrv
d. Outpatient_2018_PrvSrv
e. Outpatient_2019_PrvSrv
f. Outpatient_2020_PrvSrv
g. MD_PAC_and_Hospice_by_Geo_Prv_20
h. MD_PAC_SNF_2020
i. MD_PAC_HOS_2020

The codes for the creation of all additional tables are included in Appendix B – Section I.

3. Tables created from another table in the database
   Certain additional tables were created using SQL queries from existing tables for exploratory data analysis and data visualization. A sample script for the same is shown below:

SQL query/queries on database l501saptpurkFall22grp_07_db:

```
1 CREATE TABLE IHP_TOPDISCH_DISEASE_18 AS
2 (SELECT A.DRG_CD, A.DRG_DESC, SUM(A.TOT_DSCHRGS) AS TOT_DSCH_18
3 FROM Inpatient_2018_PrvSrv A
4 WHERE A.DRG_CD IN (871,291,470,177,193,872,189,378,392,690)
5 GROUP BY A.DRG_CD, A.DRG_DESC
6 ORDER BY SUM(A.TOT_DSCHRGS) DESC);
```

Figure 2: SQL Query to create Table IHP_TOPDISCH_DISEASE_18 in the MariaDB group Database.

The tables imported by this method are mentioned below:

a. IHP_TOPDISCH_DISEASE_18
b. IHP_TOPDISCH_DISEASE_19
c. IHP_TOPDISCH_DISEASE_20
d. IHP_TOPDISCH_STATE_18
e. IHP_TOPDISCH_STATE_19
f. IHP_TOPDISCH_STATE_20
g. OPD_TOPDISCH_DISEASE_18
h. OPD_TOPDISCH_DISEASE_19
i. OPD_TOPDISCH_DISEASE_20
j. OPD_TOPDISCH_STATE_18
k. OPD_TOPDISCH_STATE_19
l. OPD_TOPDISCH_STATE_20

The SQL queries for creating remaining tables are provided in Appendix A – Section I.

The final list of tables in the group database is provided below:

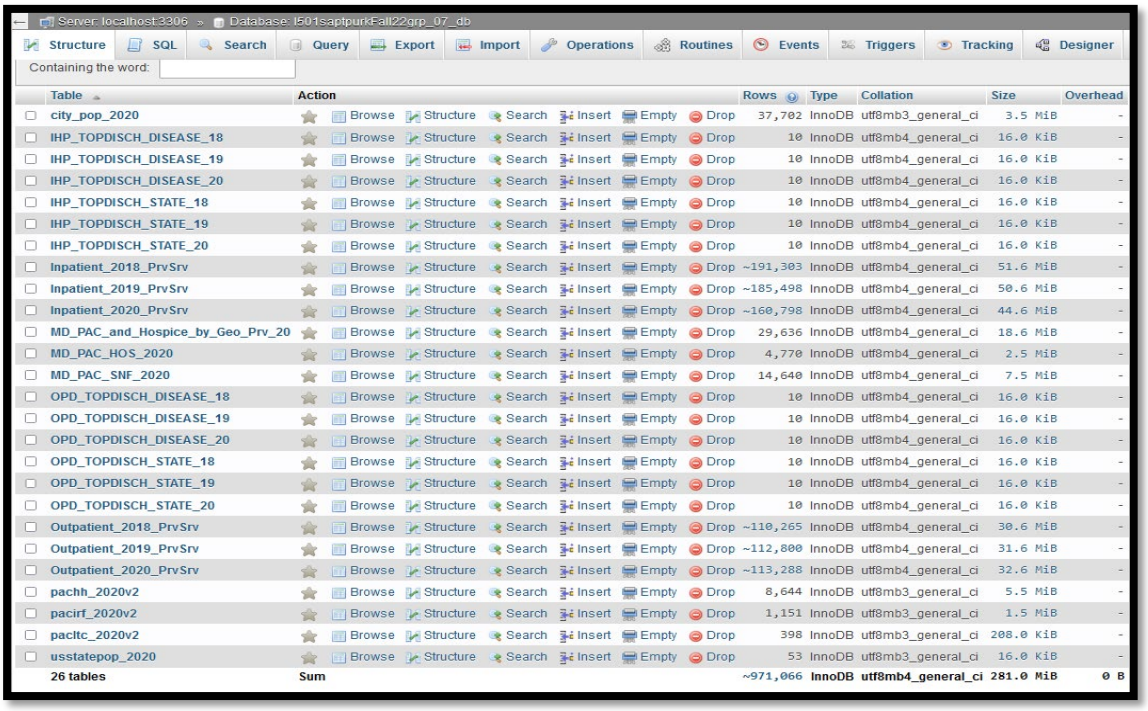

Table	Action	Rows	Type	Collation	Size	Overhead
city_pop_2020	Browse Structure Search Insert Empty Drop	37,702	InnoDB	utf8mb3_general_ci	3.5 MiB	-
IHP_TOPDISCH_DISEASE_18	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
IHP_TOPDISCH_DISEASE_19	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
IHP_TOPDISCH_DISEASE_20	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
IHP_TOPDISCH_STATE_18	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
IHP_TOPDISCH_STATE_19	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
IHP_TOPDISCH_STATE_20	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
Inpatient_2018_PrvSrv	Browse Structure Search Insert Empty Drop	~191,303	InnoDB	utf8mb4_general_ci	51.6 MiB	-
Inpatient_2019_PrvSrv	Browse Structure Search Insert Empty Drop	~185,498	InnoDB	utf8mb4_general_ci	50.6 MiB	-
Inpatient_2020_PrvSrv	Browse Structure Search Insert Empty Drop	~160,798	InnoDB	utf8mb4_general_ci	44.6 MiB	-
MD_PAC_and_Hospice_by_Geo_Prv_20	Browse Structure Search Insert Empty Drop	29,636	InnoDB	utf8mb4_general_ci	18.6 MiB	-
MD_PAC_HOS_2020	Browse Structure Search Insert Empty Drop	4,770	InnoDB	utf8mb4_general_ci	2.5 MiB	-
MD_PAC_SNF_2020	Browse Structure Search Insert Empty Drop	14,640	InnoDB	utf8mb4_general_ci	7.5 MiB	-
OPD_TOPDISCH_DISEASE_18	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
OPD_TOPDISCH_DISEASE_19	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
OPD_TOPDISCH_DISEASE_20	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
OPD_TOPDISCH_STATE_18	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
OPD_TOPDISCH_STATE_19	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
OPD_TOPDISCH_STATE_20	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_general_ci	16.0 KiB	-
Outpatient_2018_PrvSrv	Browse Structure Search Insert Empty Drop	~110,265	InnoDB	utf8mb4_general_ci	30.6 MiB	-
Outpatient_2019_PrvSrv	Browse Structure Search Insert Empty Drop	~112,800	InnoDB	utf8mb4_general_ci	31.6 MiB	-
Outpatient_2020_PrvSrv	Browse Structure Search Insert Empty Drop	~113,288	InnoDB	utf8mb4_general_ci	32.6 MiB	-
pachh_2020v2	Browse Structure Search Insert Empty Drop	8,644	InnoDB	utf8mb3_general_ci	5.5 MiB	-
pacirf_2020v2	Browse Structure Search Insert Empty Drop	1,151	InnoDB	utf8mb3_general_ci	1.5 MiB	-
pacltc_2020v2	Browse Structure Search Insert Empty Drop	398	InnoDB	utf8mb3_general_ci	208.0 KiB	-
usstatepop_2020	Browse Structure Search Insert Empty Drop	53	InnoDB	utf8mb3_general_ci	16.0 KiB	-
26 tables	**Sum**	**~971,066**	**InnoDB**	**utf8mb4_general_ci**	**281.0 MiB**	**0 B**

Figure 3: Complete list of tables in the group database in Maria DB

5. Data Analysis & Visualization

Python packages Pandas, NumPy, Matplotlib, Seaborn and Plotly were used for exploratory data analysis and visualization of all relevant datasets. We also used Scikitlearn and Plotly to implement Birch clustering and principal component analysis to determine if there were differences in the services availed for post-acute care and hospice services.

For every analysis, we accessed the data directly from the group database using a SQL Query and constructed a Data frame using pandas that was then used for visualization or analysis.

We used the same process of connecting to the database as mentioned in Figure 1. A list of all python packages we used for our analysis are shown below:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import datetime
import time
import seaborn as sns
%matplotlib inline
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode, iplot
import plotly.graph_objs as go
import chart_studio.plotly as py
from plotly import subplots
from sklearn.cluster import Birch, MiniBatchKMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Figure 4: List of Python Packages used for Data Analysis and Visualization

The process of analysis and the results obtained are discussed below:

5.1 Exploratory Data Analysis

5.1.1. Visualizing Distribution of Facilities across the US States Using Choropleths

In order to determine whether the distribution of hospital and hospice facilities was similar across different geographic levels across the US, we studied the distribution of facilities across the US at the State level.

The results are visualized in the form of 2 Choropleths for each type of hospital facility:

a. Number of Facilities per State

```
#Inpatient Facilites by State

dfihp = pd.read_sql_query(" SELECT DISTINCT Prv_State, COUNT(*) AS TOTAL_IHP FROM Inpatient_2020_PrvSrv
                          WHERE Prv_State not in ('Rndrng_Prvdr_State_Abrvtn') GROUP BY Prv_State
                          ORDER BY TOTAL_IHP DESC" , conn)
dfihp = pd.DataFrame(dfihp)
print(dfihp.head())

fig = go.Figure(data=go.Choropleth(
    locations=dfihp['Prv_State'],
    z = dfihp['TOTAL_IHP'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Blues',
    colorbar_title = "Number of Facilities",
))

fig.update_layout(
    title_text = 'Inpatient Facilities in Different States',
    geo_scope='usa',
)

fig.show()
```

Figure 5: Code for visualizing Number of Inpatient Facilities per State on a Choropleth

Codes for visualizing the other types of facilities (outpatient, SNFs, Hospice, IRF, Home Health and LTCs are provided in Appendix B – Section II.

Data for this analysis is plotted in terms of Number of Facilities per State.

b. Number of Facilities per State normalized to State population

```
#Inpatient Facilites by State normalized to population

dfihp = pd.read_sql_query("SELECT DISTINCT A.Prv_State, COUNT(A.Prv_State) AS TOTAL_IHP, B.Pop_2020 AS STATE_POP,
                          (COUNT(A.Prv_State)/B.Pop_2020)*1000000 AS IHP_RATIO FROM Inpatient_2020_PrvSrv A
                          JOIN usstatepop_2020 B ON A.Prv_State = B.State_Abbr
                          WHERE Prv_State not in ('Rndrng_Prvdr_State_Abrvtn') GROUP BY Prv_State
                          ORDER BY IHP_RATIO DESC" , conn)
dfihp = pd.DataFrame(dfihp)
print(dfihp.head())

fig = go.Figure(data=go.Choropleth(
    locations=dfihp['Prv_State'],
    z = dfihp['IHP_RATIO'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Blues',
    colorbar_title = "FACILITES to STATE POPULATION RATIO * (10^6)",
))

fig.update_layout(
    title_text = 'Inpatient Facilities in Different States (Normalized with State Population data)',
    geo_scope='usa',
)

fig.show()
```

Figure 6: Code for visualizing Number of Inpatient Facilities per State normalized with population data on a Choropleth

Codes for visualizing the other types of facilities (outpatient, SNFs, Hospice, IRF, Home Health and LTCs are provided in Appendix B – Section II.

Data for this analysis is plotted as a ratio of Number of Facilities per State normalized to State population which is multiplied by a factor of 10^6 for visualization.
The results are presented below as 2 choropleths for inpatient, outpatient, SNF and Hospice facilities.
For IRFs, LTCs and Home Health agencies a single choropleth showing the facilities to population ratio is provided.

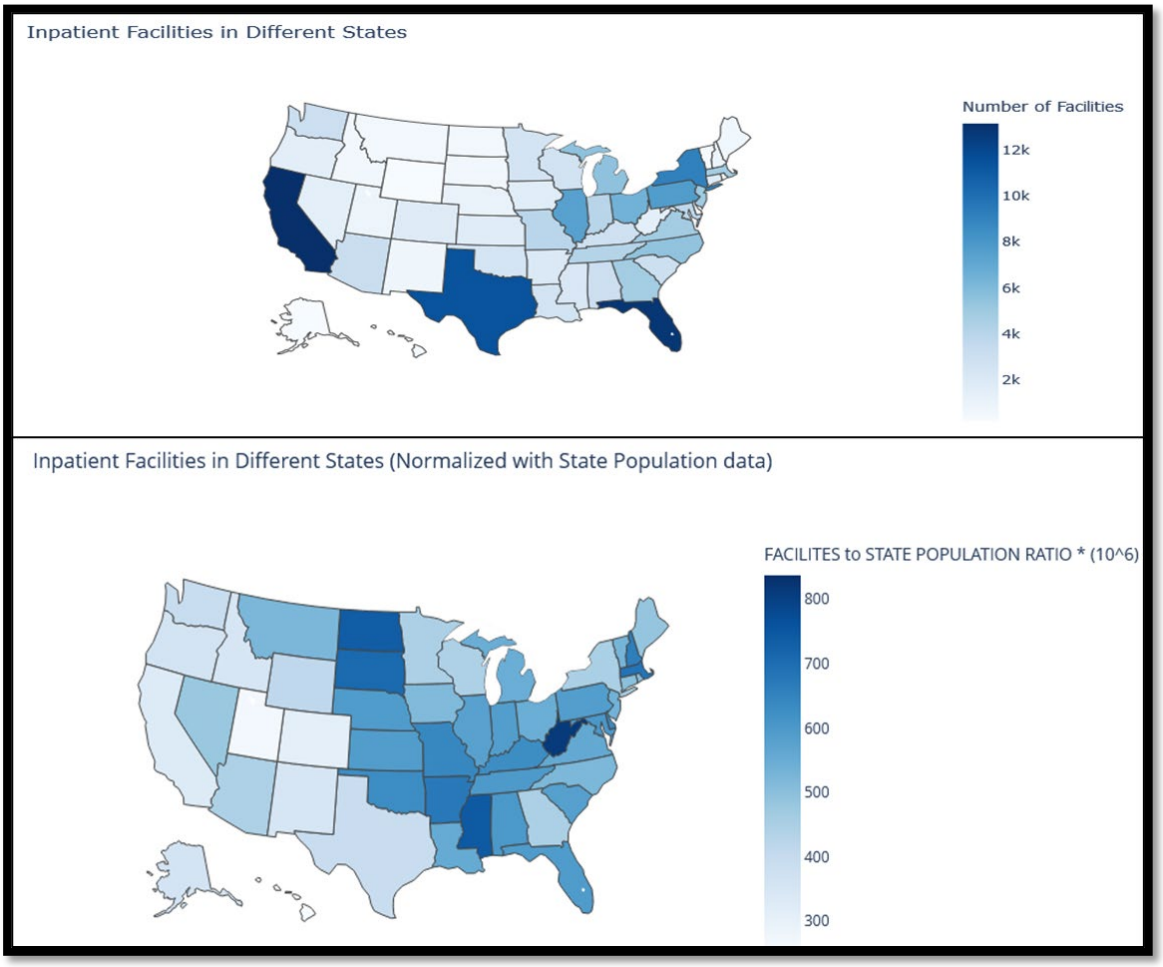

Figure 7: Choropleths showing the difference in distribution of inpatient facilities across the different states in the US in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

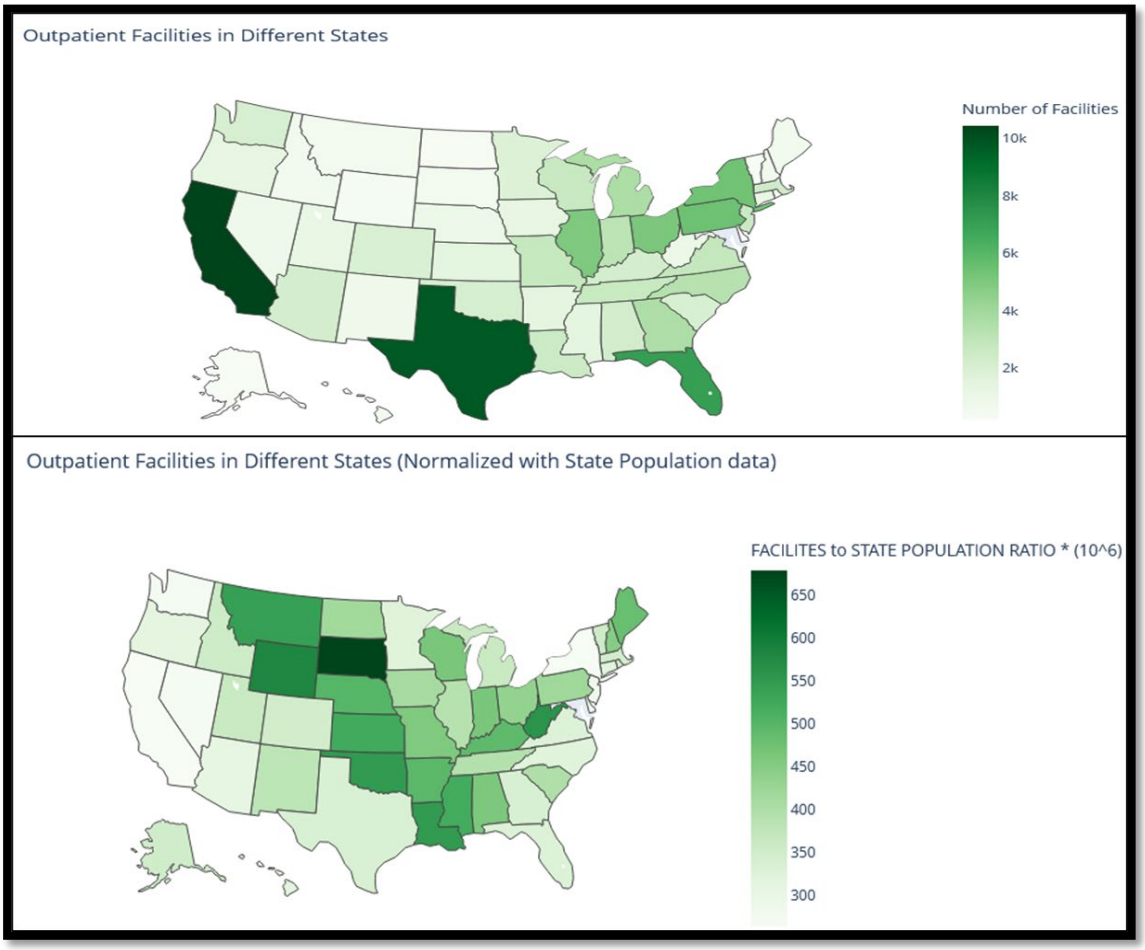

Figure 8: Choropleths showing the difference in distribution of outpatient facilities across the different states in the US in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

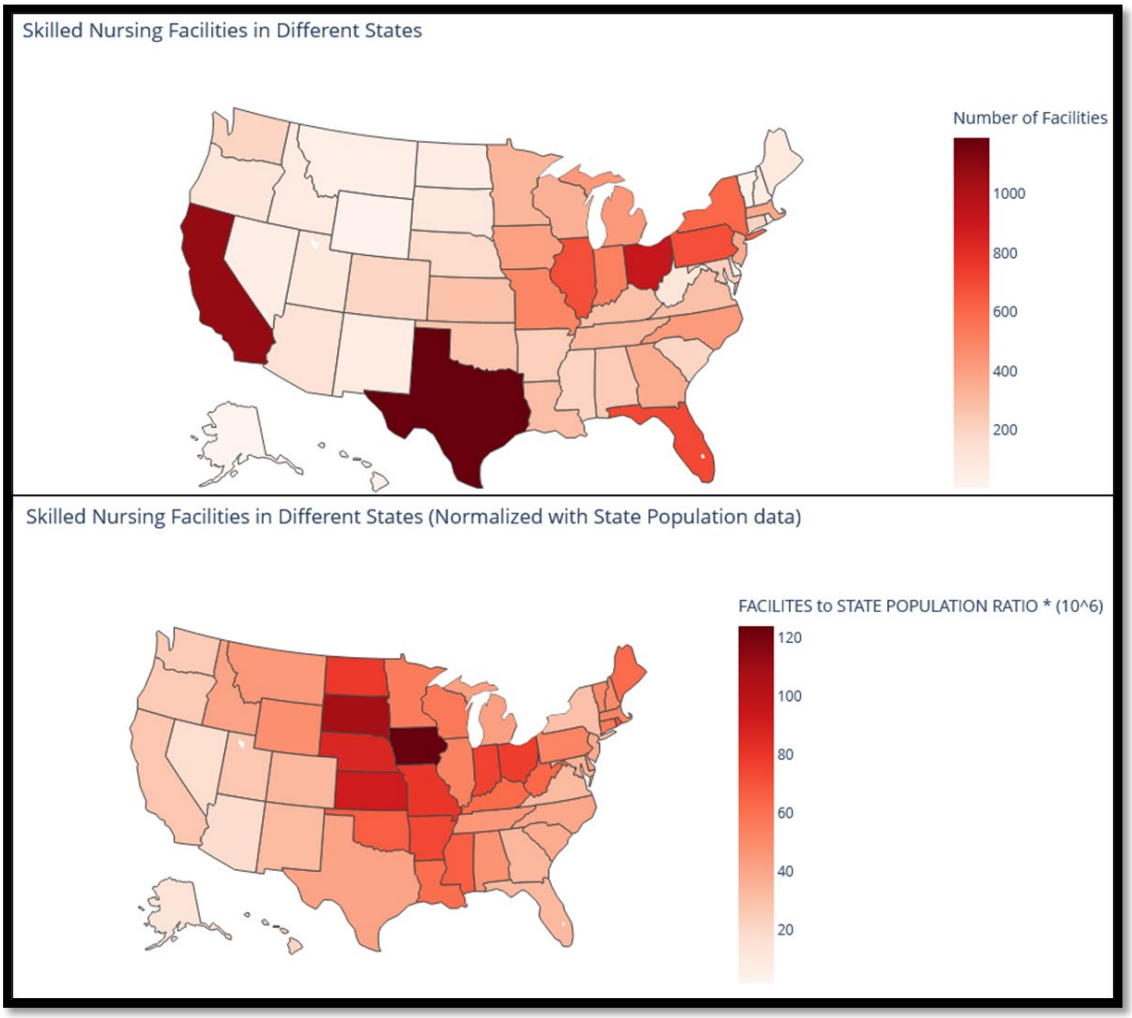

Figure 9: Choropleths showing the difference in distribution of Skilled Nursing facilities across the different states in the US in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

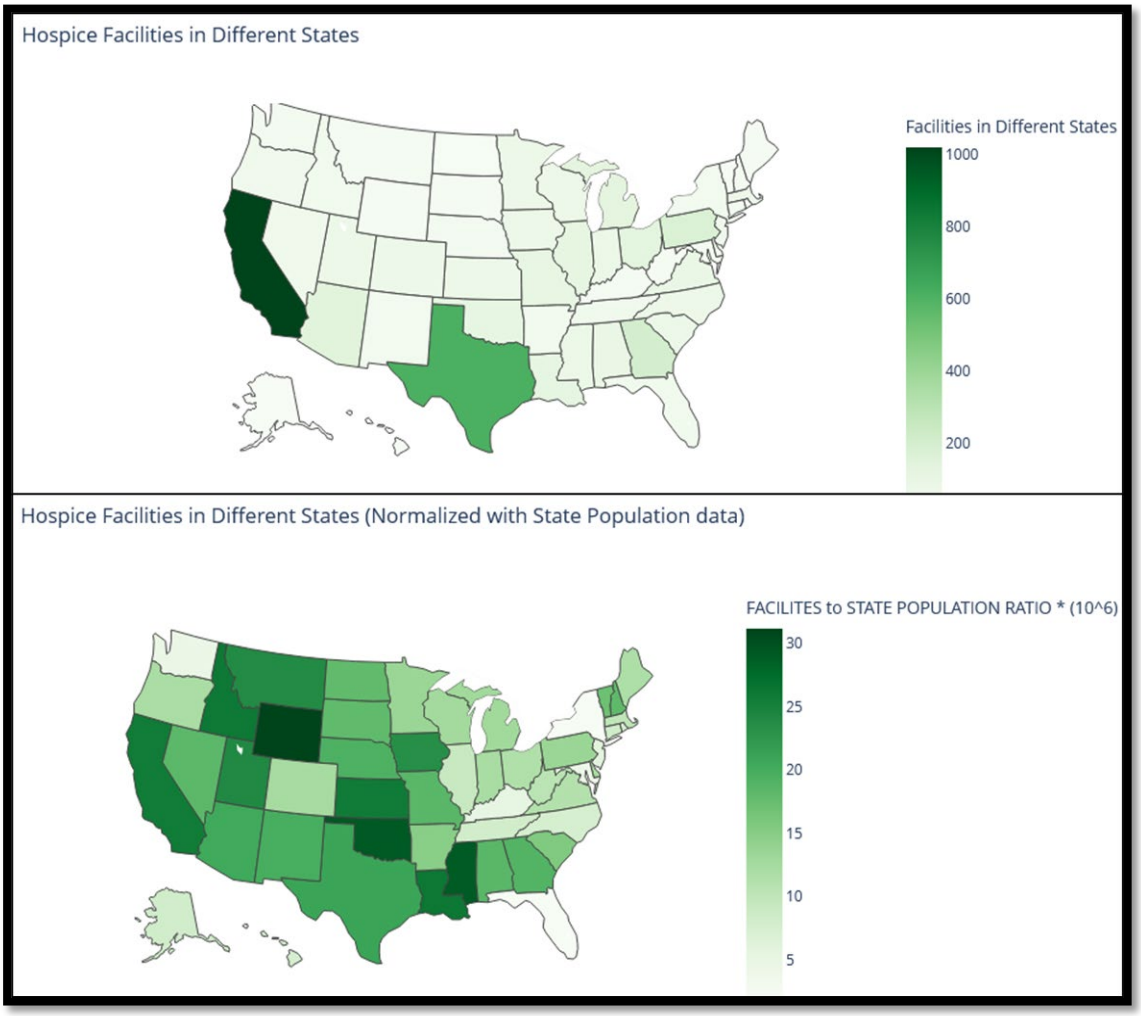

Figure 10: Choropleths showing the difference in distribution of Hospice facilities across the different states in the US in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

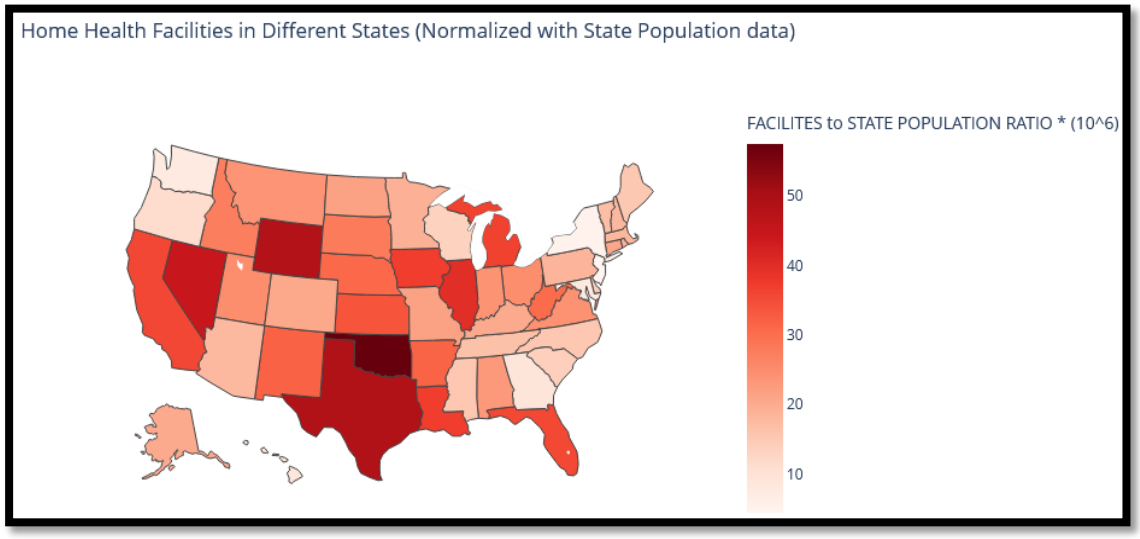

Figure 11: Choropleth showing the distribution of ratio of Home Health facilities normalized to population of each respective state.

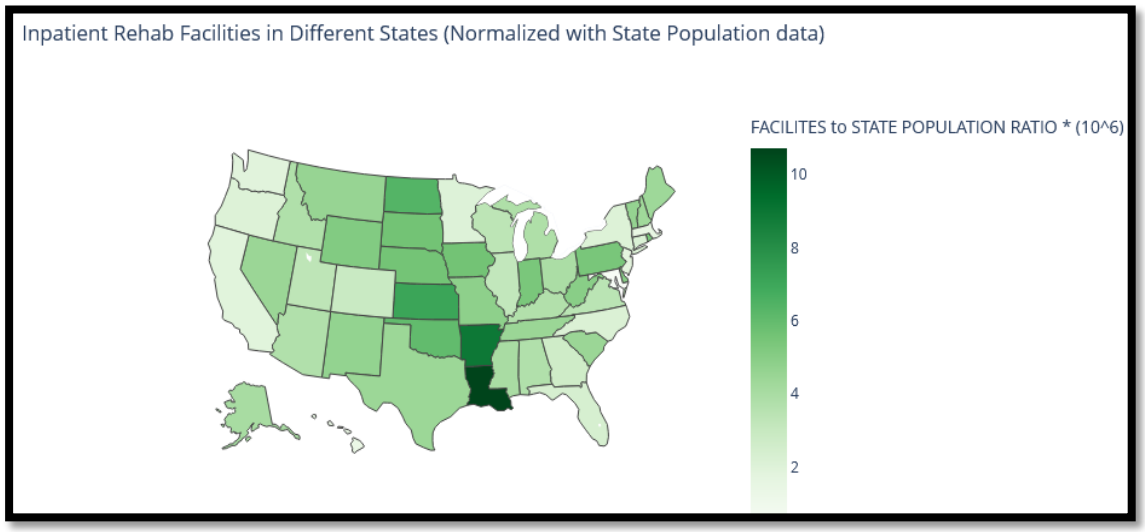

Figure 12: Choropleth showing the distribution of ratio of Inpatient Rehab facilities normalized to population of each respective state.

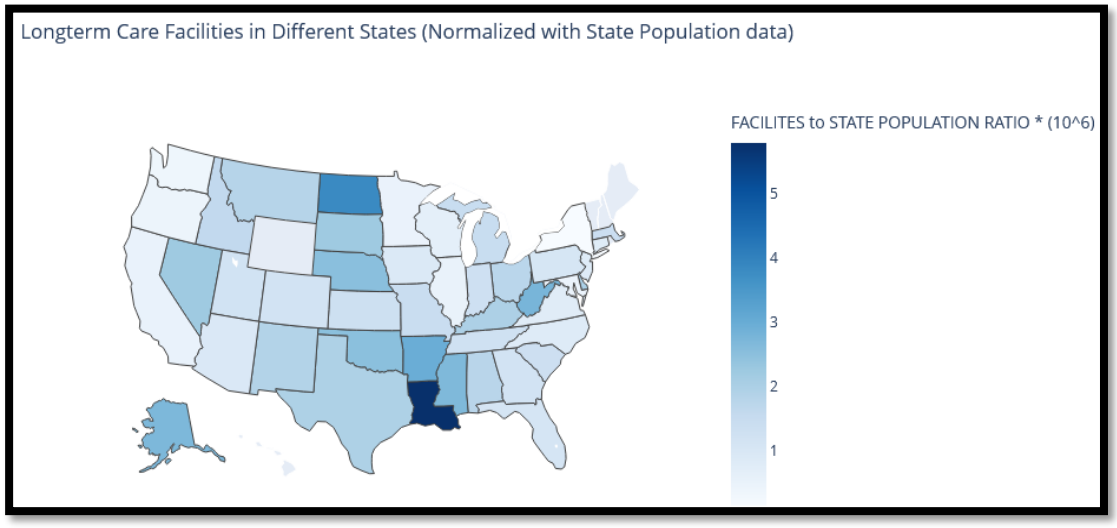

Figure 13: Choropleth showing the distribution of the ratio of -Term Care facilities normalized to population of each respective state.

From the results we can conclude that while the highest number of facilities are in the more populated states of California and Texas, when normalized to State population numbers as a metric for access to care, states with fewer people but a proportional number of facilities afford their residents with better access to facilities.

5.1.2 Visualizing Distribution of Facilities across Cities in Indiana Using Bar Graphs

To determine whether a similar pattern could be seen across cities in a state, we did a comparative analysis of Indiana State cities with the highest numbers of facilities versus cities with the highest facility number to city population ratio, to see if the same cities showed up in both cases.

The codes for the same are provided below:

a. Number of Facilities per City

```
#Inpatient Facilites In Indiana State

dfihp_ct = pd.read_sql_query(" SELECT DISTINCT Prv_City, COUNT(*) AS TOTAL_IHP FROM Inpatient_2020_PrvSrv
                             WHERE Prv_State in ('IN') GROUP BY Prv_City ORDER BY TOTAL_IHP DESC limit 20" , conn)
dfihp_ct = pd.DataFrame(dfihp_ct)

dfihp_ct.plot.barh('Prv_City','TOTAL_IHP')
plt.xlabel('CITY')
plt.ylabel('NUMBER OF FACILITES')
plt.title('INDIANA CITIES WITH HIGHEST NUMBER OF INPATIENT FACILITIES')
plt.show()
```

Figure 14: Code for visualizing Number of Inpatient Facilities per City in cities in Indiana State on a Horizontal Bar graph

Codes for visualizing the other types of facilities (outpatient, SNFs, Hospice, IRF, Home Health and LTCs are provided in Appendix B – Section III.

Data for this analysis is plotted in terms of Number of Facilities per City.

b. Number of Facilities per City normalized to City population

```
#Inpatient Facilites In Indiana State normalized to population

dfihp_ct = pd.read_sql_query("SELECT DISTINCT A.Prv_City as CITY, COUNT(A.Prv_City) AS TOTAL_IHP,
                             B.POPEST2020 AS CITY_POP, (COUNT(A.Prv_City)/B.POPEST2020)*10000 AS IHP_CITY_RATIO
                             FROM Inpatient_2020_PrvSrv A JOIN city_pop_2020 B ON A.Prv_State = B.STATE_ABBR
                             AND A.Prv_City = B.CITY WHERE A.Prv_State in ('IN') AND A.Prv_City = B.CITY
                             GROUP BY A.Prv_City ORDER BY IHP_CITY_RATIO DESC limit 20" , conn)
dfihp_ct = pd.DataFrame(dfihp_ct)
print(dfihp_ct.head())

dfihp_ct.plot.barh('CITY','IHP_CITY_RATIO')
plt.xlabel('FACILITES to CITY POPULATION RATIO * (10^4)')
plt.ylabel('CITY')
plt.yticks(fontsize = 10)
plt.title('INPATIENT FACILITES IN INDIANA CITIES (Normalized with City Population data)')
plt.tight_layout()
plt.show()
```

Figure 15: Code for visualizing Number of Inpatient Facilities per City in cities in Indiana State normalized with population data on a Horizontal Bar graph

Codes for visualizing the other types of facilities (outpatient, SNFs, Hospice, IRF, Home Health and LTCs are provided in Appendix B – Section III.

Data for this analysis is plotted as a ratio of Number of Facilities per City normalized to City population which is multiplied by a factor of 10^4 for visualization.
The results are presented below as 2 Bar Graphs for inpatient, outpatient, SNF and Hospice facilities.
For IRFs, LTCs and Home Health agencies a single Bar Graph showing the facilities ratio is provided.

	CITY	TOTAL_IHP	CITY_POP	IHP_CITY_RATIO
0	Munster	165	23895	69.052103
1	Lawrenceburg	30	5102	58.800470
2	Danville	41	10594	38.701152
3	Vincennes	54	16776	32.188841
4	Hobart	92	29669	31.008797

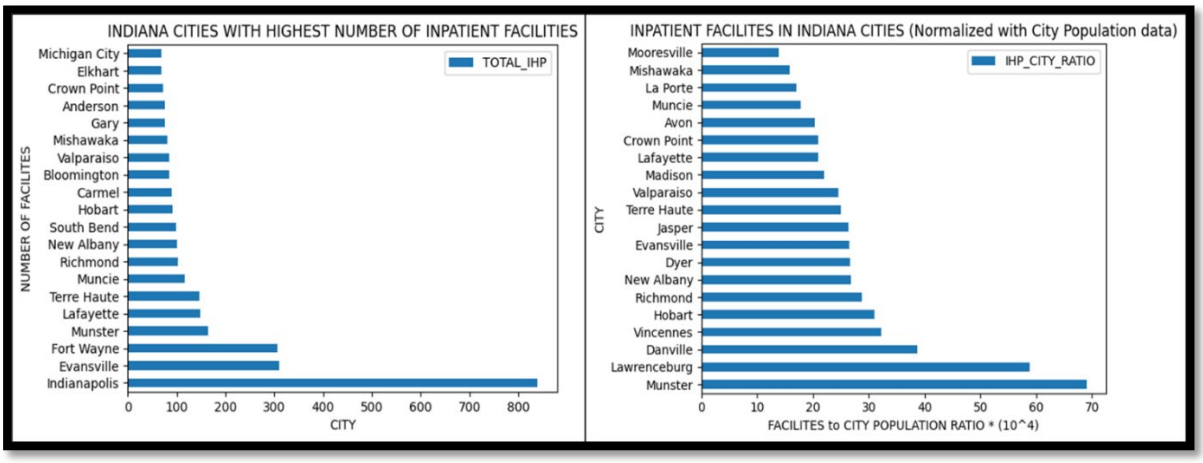

Figure 16: Bar Graphs showing the difference in distribution of inpatient facilities in different cities in Indiana in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

	CITY	TOTAL_OPD	CITY_POP	OPD_CITY_RATIO
0	Lawrenceburg	36	5102	70.560564
1	Newburgh	17	3358	50.625372
2	Munster	115	23895	48.127223
3	Knox	15	3587	41.817675
4	Danville	42	10594	39.645082

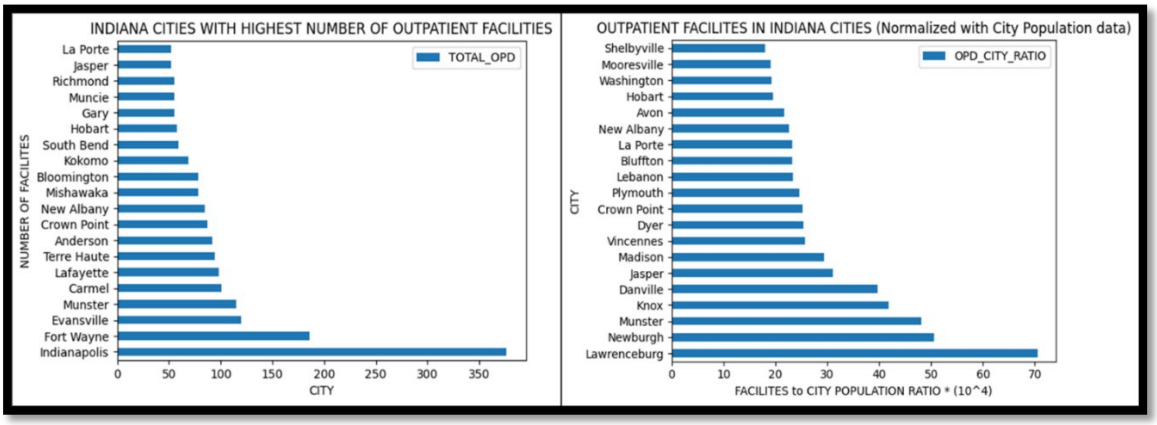

Figure 17: Bar Graphs showing the difference in distribution of outpatient facilities in different cities in Indiana in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

	CITY	TOTAL_OPD	CITY_POP	SNF_CITY_RATIO
0	LEAVENWORTH	1	264	37.878788
1	OAKTOWN	1	579	17.271157
2	NEWBURGH	5	3358	14.889815
3	NEW HARMONY	1	686	14.577259
4	KNIGHTSVILLE	1	689	14.513788

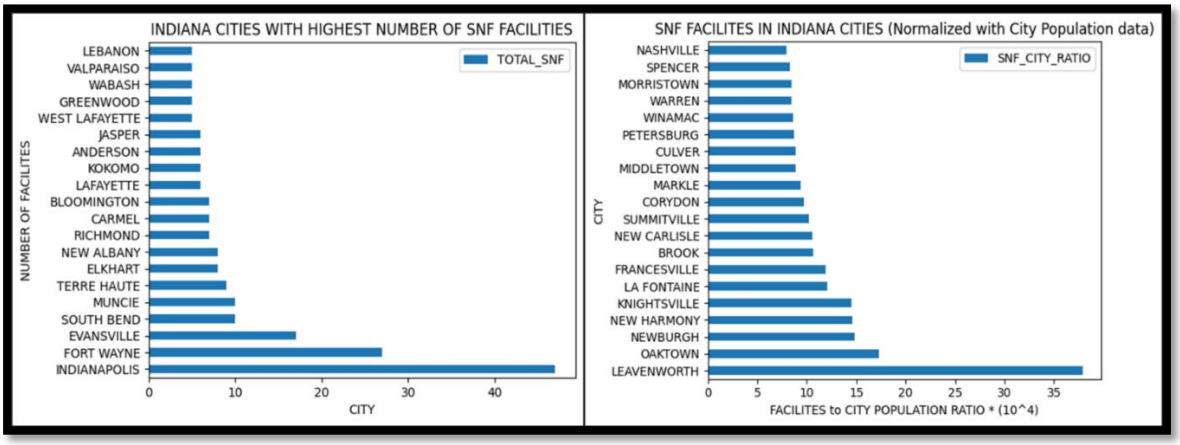

Figure 18: Bar Graphs showing the difference in distribution of SNF facilities in different cities in Indiana in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

	CITY	TOTAL_HOSPICE	CITY_POP	HSP_CITY_RATIO
0	LAWRENCEBURG	2	5102	3.920031
1	MITCHELL	1	3930	2.544529
2	BERNE	1	4141	2.414876
3	ROCHESTER	1	6280	1.592357
4	PORTLAND	1	6292	1.589320

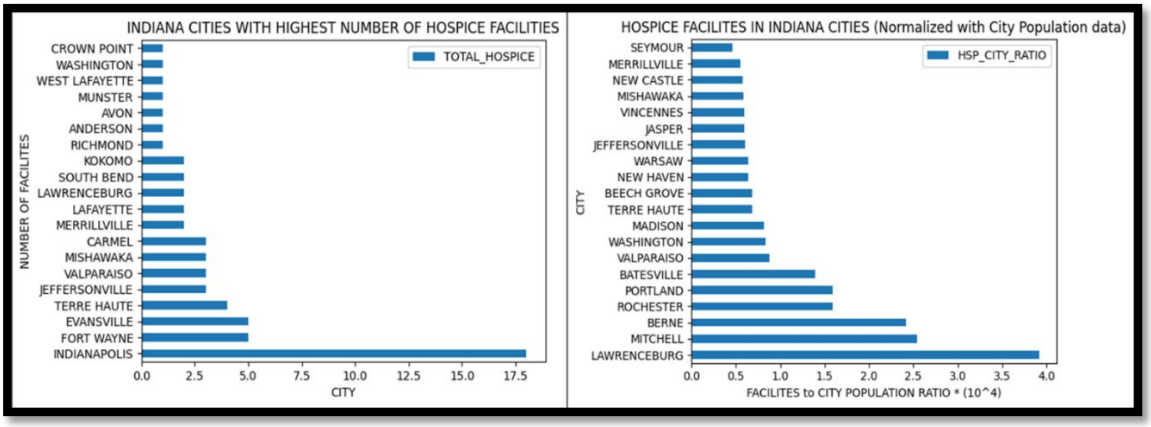

Figure 19: Bar Graphs showing the difference in distribution of Hospice facilities in different cities in Indiana in terms of number of Facilities (Plot 1) and when normalized with the population of each respective state (Plot 2).

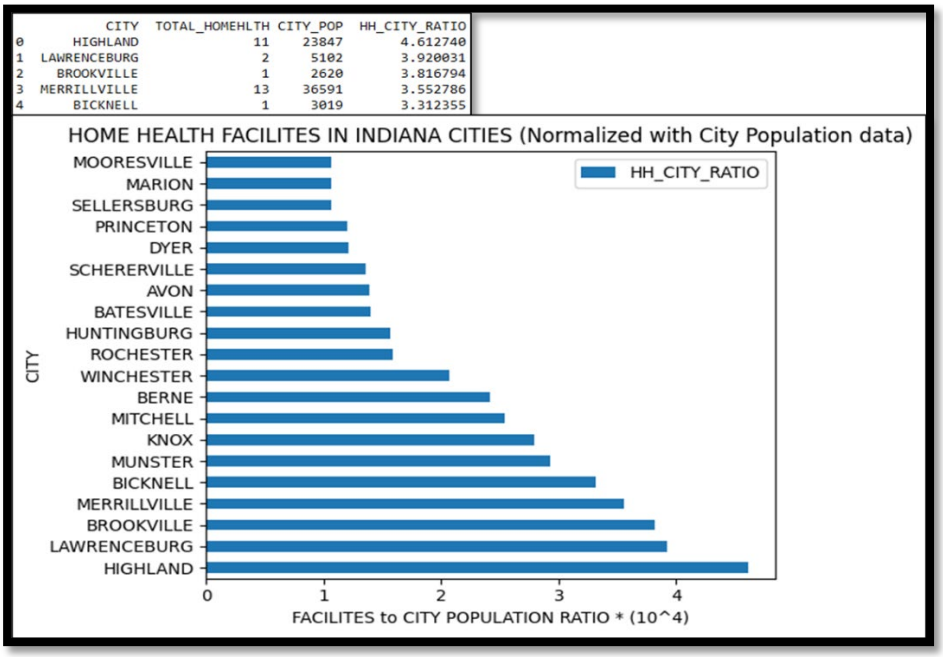

Figure 20: Bar Graph showing the distribution of ratio of Home Health facilities normalized to population of the respective city

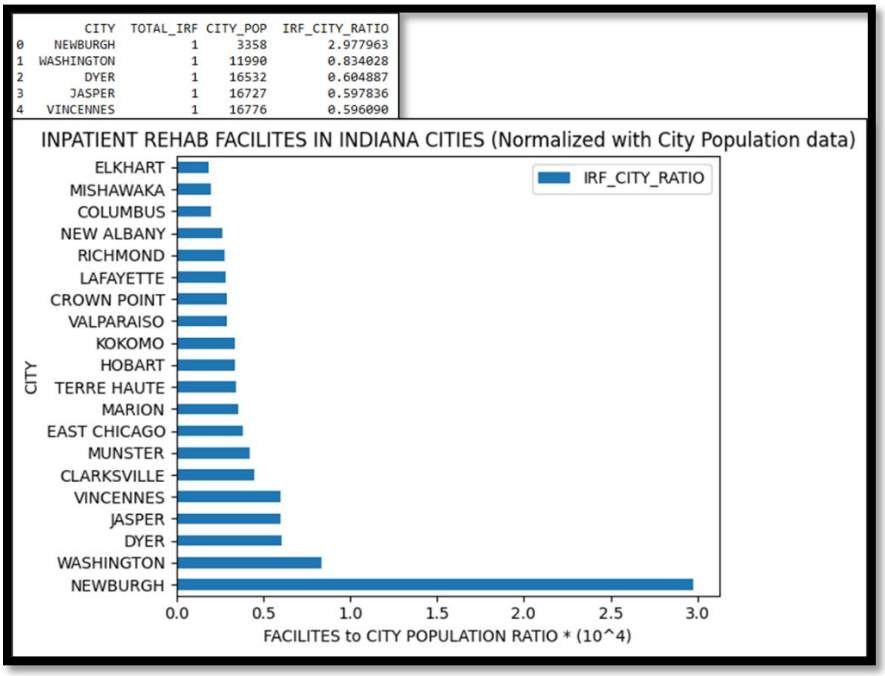

Figure 21: Bar Graph showing the distribution of ratio of IRF facilities normalized to population of the respective city

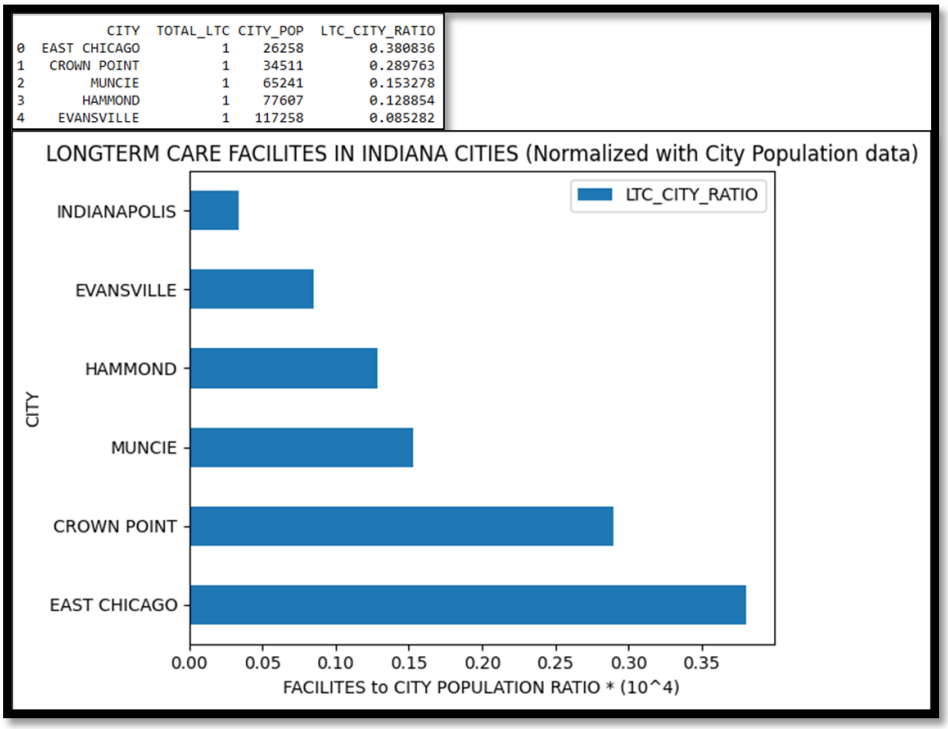

Figure 22: Bar Graph showing the distribution of ratio of LTC facilities normalized to population of the respective city

From the results, it is evident that a similar pattern to what was observed at the State level can also be seen at the city level within Indiana state.

5.1.3 Change in most availed inpatient and outpatient services before and during the COVID pandemic

One of our research questions was to ascertain whether there was a change in the most common indications patients came in for before and during the COVID pandemic i.e., between 2018 and 2020. To address this, we identified the inpatient and outpatient services with the highest patient counts (TOT_DSCHRGS column in table) at the state level across the US. We also looked at the changes in overall patient counts during this period to determine if a similar trend was observed. The codes for the same are provided below:

```
# TOP 10 DISEASES AND STATES FOR INPATIENT CONDITIONS

dfihptp = pd.read_sql_query("SELECT DISTINCT A.DRG_CD, A.DRG_DESC, SUM(A.TOT_DSCHRGS) AS TOT_DSCH_DIS_20
                            FROM Inpatient_2020_PrvSrv A GROUP BY A.DRG_CD, A.DRG_DESC ORDER BY SUM(A.TOT_DSCHRGS)
                            DESC LIMIT 10" , conn)
dfihptp = pd.DataFrame(dfihptp)
print(dfihptp.head())

dfihptp = pd.read_sql_query("SELECT DISTINCT A.Prv_City, SUM(A.TOT_DSCHRGS) AS TOT_DSCH_ST_20
                            FROM Inpatient_2020_PrvSrv A GROUP BY A.Prv_City ORDER BY SUM(A.TOT_DSCHRGS)
                            DESC LIMIT 10" , conn)
dfihptp = pd.DataFrame(dfihptp)
print(dfihptp.head())
```

```
   DRG_CD                                          DRG_DESC  TOT_DSCH_DIS_20
0     871  SEPTICEMIA OR SEVERE SEPSIS W/O MV >96 HOURS W...         586558.0
1     291                  HEART FAILURE AND SHOCK WITH MCC         329977.0
2     470  MAJOR HIP AND KNEE JOINT REPLACEMENT OR REATTA...         268571.0
3     177  RESPIRATORY INFECTIONS AND INFLAMMATIONS WITH MCC         179339.0
4     193            SIMPLE PNEUMONIA AND PLEURISY WITH MCC         128737.0
      Prv_City  TOT_DSCH_ST_20
0     New York         75638.0
1    Baltimore         50864.0
2      Houston         46163.0
3       Boston         46147.0
4  Springfield         44568.0
```

Figure 23: Code and partial output for determining top 10 inpatient services availed in the US and states with the maximum patient counts

```
# TOP 10 DISEASES FOR INPATIENT CONDITIONS

ihp_pt1 = pd.read_sql_query("SELECT DRG_CD, DRG_DESC, TOT_DSCH_18 from IHP_TOPDISCH_DISEASE_18 order by DRG_CD" , conn)
ihp_pt1 = pd.DataFrame(ihp_pt1)
#print(ihp_pt1)
ihp_pt2 = pd.read_sql_query("SELECT DRG_CD,TOT_DSCH_19 from IHP_TOPDISCH_DISEASE_19 order by DRG_CD" , conn)
ihp_pt2 = pd.DataFrame(ihp_pt2)
#print(ihp_pt2)
ihp_pt3 = pd.read_sql_query("SELECT DRG_CD,TOT_DSCH_20 from IHP_TOPDISCH_DISEASE_20 order by DRG_CD" , conn)
ihp_pt3 = pd.DataFrame(ihp_pt3)
#print(ihp_pt3)
ihp_pt = pd.concat([ihp_pt1, ihp_pt2['TOT_DSCH_19'] , ihp_pt3['TOT_DSCH_20']], axis=1)
ihp_pt
```

	DRG_CD	DRG_DESC	TOT_DSCH_18	TOT_DSCH_19	TOT_DSCH_20
0	177	RESPIRATORY INFECTIONS & INFLAMMATIONS W MCC	73413.0	72024.0	358678.0
1	189	PULMONARY EDEMA & RESPIRATORY FAILURE	147918.0	142779.0	213582.0
2	193	SIMPLE PNEUMONIA & PLEURISY W MCC	164685.0	147397.0	257474.0
3	291	HEART FAILURE & SHOCK W MCC	381898.0	393492.0	659954.0
4	378	G.I. HEMORRHAGE W CC	128431.0	122839.0	207508.0
5	392	"ESOPHAGITIS, GASTROENT & MISC DIGEST DISORDER...	141917.0	133569.0	204170.0
6	470	MAJOR HIP AND KNEE JOINT REPLACEMENT OR REATTA...	463050.0	420838.0	537142.0
7	690	KIDNEY & URINARY TRACT INFECTIONS W/O MCC	128310.0	126130.0	190072.0
8	871	SEPTICEMIA OR SEVERE SEPSIS W/O MV >96 HOURS W...	625930.0	607114.0	1173116.0
9	872	SEPTICEMIA OR SEVERE SEPSIS W/O MV >96 HOURS W...	159380.0	153356.0	251570.0

Figure 24: Code and output for determining patient counts for top 10 inpatient services availed in the US during 2018-2020

```
# TOP 10 DISEASES FOR INPATIENT CONDITIONS

X = ihp_pt['DRG_DESC']
y = ihp_pt['TOT_DSCH_18']
z = ihp_pt['TOT_DSCH_19']
a = ihp_pt['TOT_DSCH_20']

Y_axis = np.arange(len(X))

plt.barh(Y_axis - 0.2, y, 0.2, label = '2018',align='center')
plt.barh(Y_axis, z, 0.2, label = '2019',align='center')
plt.barh(Y_axis + 0.2, a, 0.2, label = '2020',align='center')

plt.yticks(Y_axis, X)
plt.ylabel("INPATIENT CONDITION")
plt.xlabel("TOTAL DISCHARGES")
plt.title("TOP 10 INPATIENT DISEASES WITH MAX PATIENT COUNTS ACROSS THE US")
plt.legend()
plt.show()
```

Figure 25: Code for visualizing top 10 inpatient services availed in the US during 2018-2020 on a horizontal bar graph

```
# TOP 10 STATES FOR INPATIENT CONDITIONS

ihp_pt4 = pd.read_sql_query("SELECT Prv_State, TOT_DSCH_18 from IHP_TOPDISCH_STATE_18 order by Prv_State" , conn)
ihp_pt4 = pd.DataFrame(ihp_pt4)
#print(ihp_pt4)
ihp_pt5 = pd.read_sql_query("SELECT Prv_State, TOT_DSCH_19 from IHP_TOPDISCH_STATE_19 order by Prv_State" , conn)
ihp_pt5 = pd.DataFrame(ihp_pt5)
#print(ihp_pt5)
ihp_pt6 = pd.read_sql_query("SELECT Prv_State, TOT_DSCH_20 from IHP_TOPDISCH_STATE_20 order by Prv_State" , conn)
ihp_pt6 = pd.DataFrame(ihp_pt6)
#print(ihp_pt6)
ihp_ptb = pd.concat([ihp_pt4, ihp_pt5['TOT_DSCH_19'] , ihp_pt6['TOT_DSCH_20']], axis=1)
```

	Prv_State	TOT_DSCH_18	TOT_DSCH_19	TOT_DSCH_20
0	CA	536190.0	526766.0	896576.0
1	FL	566195.0	548276.0	952976.0
2	IL	324291.0	316887.0	531734.0
3	MA	225035.0	222575.0	387404.0
4	MI	269411.0	254397.0	400720.0
5	NC	257082.0	249423.0	414592.0
6	NY	428538.0	426884.0	715314.0
7	OH	277340.0	262206.0	432632.0
8	PA	316478.0	307961.0	513138.0
9	TX	491290.0	472628.0	801114.0

Figure 26: Code and output for visualizing top 10 states availing inpatient services availed in the US during 2018-2020 on a horizontal bar graph

```
# TOP 10 STATES FOR INPATIENT CONDITIONS

X = ihp_ptb['Prv_State']
y = ihp_ptb['TOT_DSCH_18']
z = ihp_ptb['TOT_DSCH_19']
a = ihp_ptb['TOT_DSCH_20']

Y_axis = np.arange(len(X))

plt.barh(Y_axis - 0.2, y, 0.2, label = '2018',align='center')
plt.barh(Y_axis, z, 0.2, label = '2019',align='center')
plt.barh(Y_axis + 0.2, a, 0.2, label = '2020',align='center')


plt.yticks(Y_axis, X)
plt.ylabel("STATE")
plt.xlabel("TOTAL DISCHARGES")
plt.title("TOP 10 STATES WITH MAX INPATIENT CASES ACROSS THE US")
plt.legend()
plt.show()
```

Figure 27: Code for visualizing top 10 states availing inpatient services availed in the US during 2018-2020 on a horizontal bar graph

```
# TOP 10 DISEASES FOR OUTPATIENT CONDITIONS

opd_pt1 = pd.read_sql_query("SELECT APC_Cd, APC_Desc, TOT_DSCH_18 from OPD_TOPDISCH_DISEASE_18 order by APC_Cd" , conn)
opd_pt1 = pd.DataFrame(opd_pt1)
#print(opd_pt1)
opd_pt2 = pd.read_sql_query("SELECT APC_Cd, APC_Desc, TOT_DSCH_19 from OPD_TOPDISCH_DISEASE_19 order by APC_Cd" , conn)
opd_pt2 = pd.DataFrame(opd_pt2)
#print(opd_pt2)
opd_pt3 = pd.read_sql_query("SELECT APC_Cd, APC_Desc, TOT_DSCH_20 from OPD_TOPDISCH_DISEASE_20 order by APC_Cd" , conn)
opd_pt3 = pd.DataFrame(opd_pt3)
#print(opd_pt3)
opd_pt = pd.concat([opd_pt1, opd_pt2['TOT_DSCH_19'] , opd_pt3['TOT_DSCH_20']], axis=1)
opd_pt
```

	APC_Cd	APC_Desc	TOT_DSCH_18	TOT_DSCH_19	TOT_DSCH_20
0	5072	Level 2 Excision/ Biopsy/ Incision and Drainage	407260.0	414186.0	303525.0
1	5114	Level 4 Musculoskeletal Procedures	215778.0	210238.0	181656.0
2	5115	Level 5 Musculoskeletal Procedures	110308.0	138447.0	221411.0
3	5191	Level 1 Endovascular Procedures	345852.0	348311.0	269497.0
4	5193	Level 3 Endovascular Procedures	185780.0	164919.0	136172.0
5	5302	Level 2 Upper GI Procedures	191636.0	195551.0	161761.0
6	5361	Level 1 Laparoscopy and Related Services	183598.0	184621.0	154995.0
7	5373	Level 3 Urology and Related Services	148728.0	158195.0	129782.0
8	5491	Level 1 Intraocular Procedures	301074.0	279931.0	198974.0
9	8011	Comprehensive Observation Services	1285469.0	1285953.0	920002.0

Figure 28: Code and output for determining top 10 outpatient services availed in the US and states with the maximum patient counts

```
# TOP 10 DISEASES FOR OUTPATIENT CONDITIONS

X = opd_pt['APC_Desc']
y = opd_pt['TOT_DSCH_18']
z = opd_pt['TOT_DSCH_19']
a = opd_pt['TOT_DSCH_20']

Y_axis = np.arange(len(X))

plt.barh(Y_axis - 0.2, y, 0.2, label = '2018',align='center')
plt.barh(Y_axis, z, 0.2, label = '2019',align='center')
plt.barh(Y_axis + 0.2, a, 0.2, label = '2020',align='center')

plt.yticks(Y_axis, X)
plt.ylabel("OUTPATIENT CONDITION")
plt.xlabel("TOTAL DISCHARGES")
plt.title("TOP 10 OUTPATIENT DISEASES WITH MAX PATIENT COUNTS ACROSS THE US")
plt.legend()
plt.show()
```

Figure 29: Code for visualizing top 10 outpatient services availed in the US during 2018-2020 on a horizontal bar graph

```
#TOP 10 OUTPATIENT DISEASES

opd_pt4 = pd.read_sql_query("SELECT APC_Cd, APC_Desc, TOT_DSCH_18 from OPD_TOPDISCH_DISEASE_18
                            where APC_Desc not in ('Comprehensive Observation Services') order by APC_Cd" , conn)
opd_pt4 = pd.DataFrame(opd_pt4)
opd_pt5 = pd.read_sql_query("SELECT APC_Cd, APC_Desc, TOT_DSCH_19 from OPD_TOPDISCH_DISEASE_19
                            where APC_Desc not in ('Comprehensive Observation Services') order by APC_Cd" , conn)
opd_pt5 = pd.DataFrame(opd_pt5)
opd_pt6 = pd.read_sql_query("SELECT APC_Cd, APC_Desc, TOT_DSCH_20 from OPD_TOPDISCH_DISEASE_20
                            where APC_Desc not in ('Comprehensive Observation Services') order by APC_Cd" , conn)
opd_pt6 = pd.DataFrame(opd_pt6)
opd_pt = pd.concat([opd_pt4, opd_pt5['TOT_DSCH_19'] , opd_pt6['TOT_DSCH_20']], axis=1)
opd_pt


X2 = opd_pt['APC_Desc']
y2 = opd_pt['TOT_DSCH_18']
z2 = opd_pt['TOT_DSCH_19']
a2 = opd_pt['TOT_DSCH_20']

Y_axis = np.arange(len(X2))

plt.barh(Y_axis - 0.2, y2, 0.2, label = '2018',align='center')
plt.barh(Y_axis, z2, 0.2, label = '2019',align='center')
plt.barh(Y_axis + 0.2, a2, 0.2, label = '2020',align='center')

plt.yticks(Y_axis, X2)
plt.ylabel("OUTPATIENT CONDITION")
plt.xlabel("TOTAL DISCHARGES")
plt.title("TOP 10 OUTPATIENT DISEASES WITH MAX PATIENT COUNTS ACROSS THE US")
plt.legend()
plt.show()
```

Figure 30: Code for visualizing top 10 outpatient services availed in the US during 2018-2020 on a horizontal bar graph

```
#TOP 10 OUTPATIENT STATES

X = opd_ptb['Prv_State']
y = opd_ptb['TOT_DSCH_18']
z = opd_ptb['TOT_DSCH_19']
a = opd_ptb['TOT_DSCH_20']

Y_axis = np.arange(len(X))

plt.barh(Y_axis - 0.2, y, 0.2, label = '2018',align='center')
plt.barh(Y_axis, z, 0.2, label = '2019',align='center')
plt.barh(Y_axis + 0.2, a, 0.2, label = '2020',align='center')


plt.yticks(Y_axis, X)
plt.ylabel("STATE")
plt.xlabel("TOTAL DISCHARGES")
plt.title("TOP 10 STATES WITH MAX OUTPATIENT CASES ACROSS THE US")
plt.legend()
plt.show()
```

Figure 31: Code for visualizing top 10 states availing outpatient services availed in the US during 2018-2020 on a horizontal bar graph

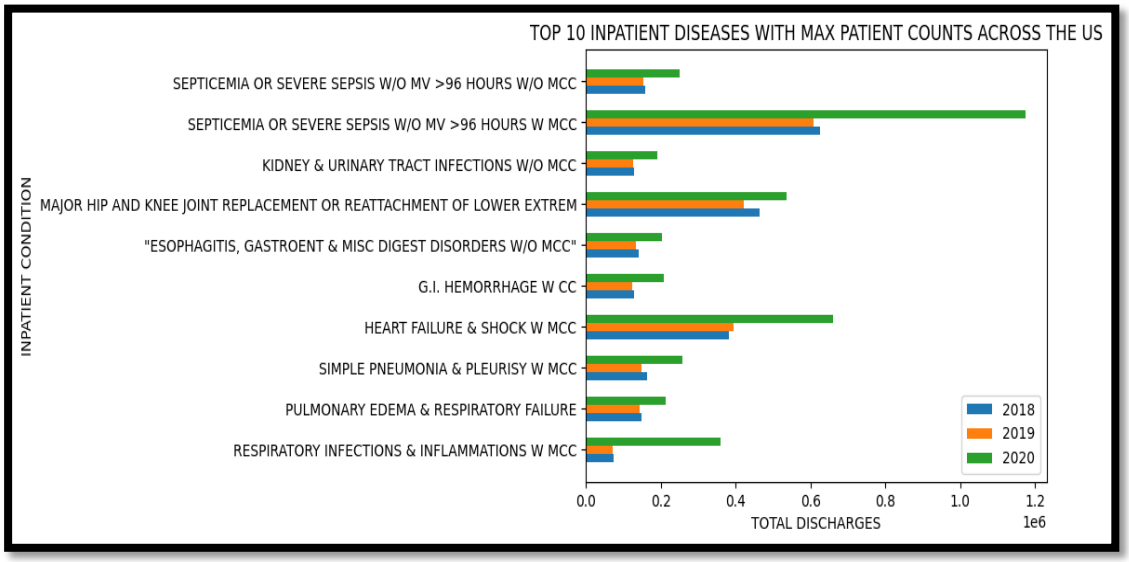

Figure 32: Changes in top 10 inpatient services in terms of patient counts accessed during 2018-2020 across the US

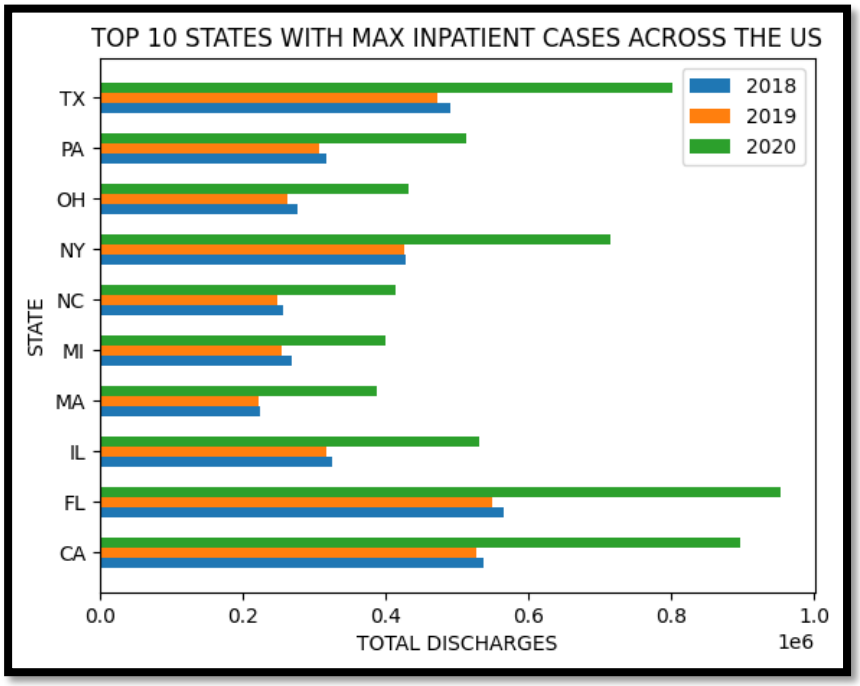

Figure 33: Changes in top 10 states availing inpatient services in terms of patient counts during 2018-2020 across the US.

From the inpatient data, we can conclude that 2020 saw a major spike in access to inpatient services related to sepsis, heart failure and respiratory infections which were commonly associated with COVID-19 infections.

There was also the expected spike in patient counts during 2020 compared with previous years, which correlates with the higher hospitalization rates seen during the pandemic.

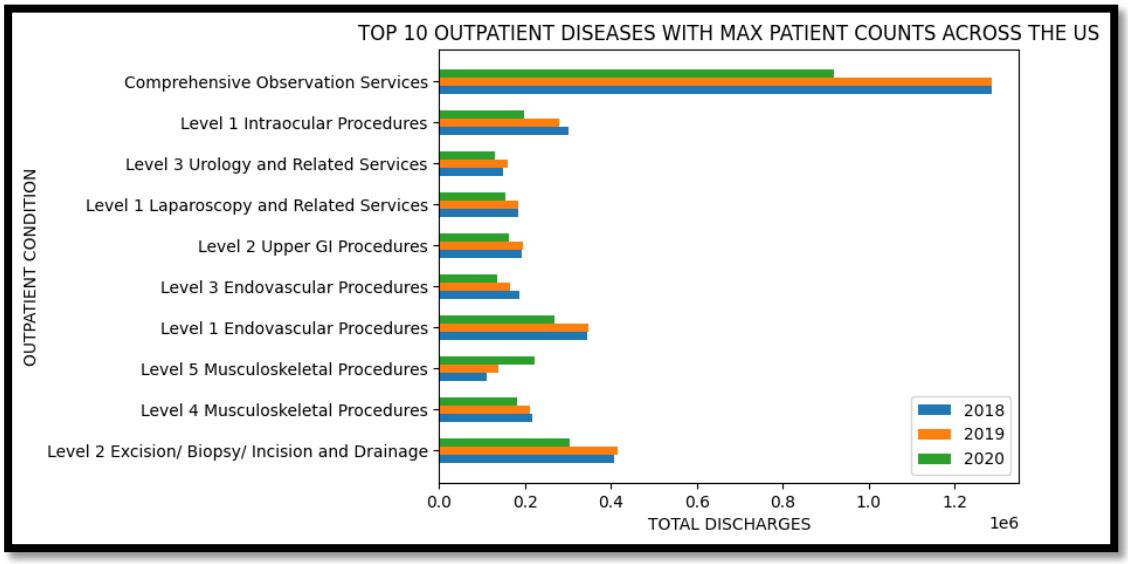

Figure 34: Changes in the top 10 outpatient services in terms of patient counts accessed during 2018-2020 across the US

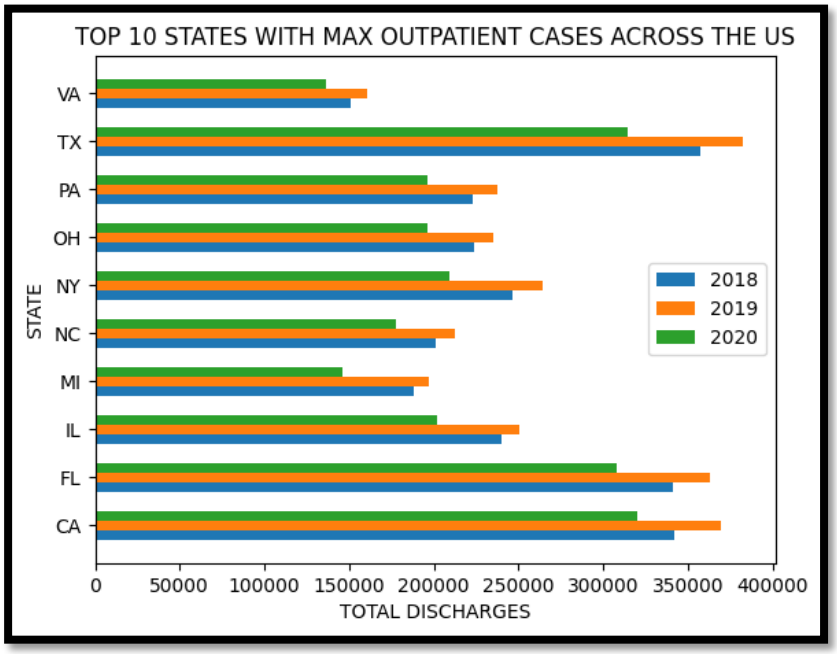

Figure 35: Changes in top 10 states availing outpatient services in terms of patient counts during 2018-2020 across the US

From the outpatient data, we can conclude that there was a slight decrease in access to services in 2020 as compared to previous years since people were advised to stay home and step out only in case of emergencies. The same trend was observed in patient counts across different states.

5.1.4 Difference in Cost for the Most Expensive Inpatient and Outpatient Services

To assess if there was a difference in cost of Medicare offered services across the US, we looked at the maximum, minimum and average cost of the most expensive inpatient and outpatient services in 2020.

We also performed a similar analysis for the most accessed services seen in the section 5.1.3 at the state level and within cities in the state of Indiana. The codes and the results for the same are provided below:

```
# Most expensive Inpatient Services

ihpxp = pd.read_sql_query("SELECT distinct ips.DRG_Desc, ROUND(AVG(ips.Avg_Mdcr_Pyt_Amt),2) AS AVG_COST,
                          ROUND(MIN(ips.Avg_Mdcr_Pyt_Amt),2) AS MIN_COST, ROUND(Max(ips.Avg_Mdcr_Pyt_Amt),2) AS MAX_COST
                          from Inpatient_2020_PrvSrv ips group by ips.DRG_Cd,ips.DRG_Desc
                          ORDER BY ROUND(AVG(ips.Avg_Mdcr_Pyt_Amt),2)  DESC limit 10" , conn)
ihpxp = pd.DataFrame(ihpxp)
print(ihpxp.head())

X = ihpxp ['DRG_Desc']
y = ihpxp ['MAX_COST']
z = ihpxp ['MIN_COST']

y_axis = np.arange(len(X))

plt.barh(y_axis - 0.2, y, 0.4, label = 'MAX COST')
plt.barh(y_axis + 0.2, z, 0.4, label = 'MIN COST')

plt.yticks(y_axis, X)
plt.ylabel("INPATIENT SERVICE")
plt.xlabel("COST OF SERVICES")
plt.title("COST OF MOST EXPENSIVE INPATIENT SERVICES")
plt.legend()
plt.show()
```

	DRG_Desc	AVG_COST	MIN_COST	MAX_COST
0	EXTENSIVE BURNS OR FULL THICKNESS BURNS WITH M...	349798.00	349798.00	349798.00
1	HEART TRANSPLANT OR IMPLANT OF HEART ASSIST SY...	255544.73	137219.94	472611.50
2	"ECMO OR TRACHEOSTOMY WITH MV >96 HOURS OR PRI...	152122.67	61340.73	434265.03
3	LUNG TRANSPLANT	120612.78	69232.64	217216.88
4	ALLOGENEIC BONE MARROW TRANSPLANT	117351.39	63064.36	204147.83

Figure 36: Code and partial output for assessing the difference in cost of the most expensive inpatient services covered by Medicare in the US on a horizontal Bar graph

```
# Difference in Cost of Most Frequently used Inpatient Services in the US

ihpxp2 = pd.read_sql_query("SELECT distinct ips.DRG_Desc, ROUND(AVG(ips.Avg_Mdcr_Pyt_Amt),2) AS AVG_COST,
                           ROUND(MIN(ips.Avg_Mdcr_Pyt_Amt),2) AS MIN_COST, ROUND(Max(ips.Avg_Mdcr_Pyt_Amt),2) AS MAX_COST
                           from Inpatient_2020_PrvSrv ips WHERE DRG_CD IN (871,291,470,177,193,872,189,378,392,690)
                           group by ips.DRG_Cd,ips.DRG_Desc ORDER BY ROUND(AVG(ips.Avg_Mdcr_Pyt_Amt),2)" , conn)
ihpxp2 = pd.DataFrame(ihpxp2)
print(ihpxp2.head())

X = ihpxp2 ['DRG_Desc']
y = ihpxp2 ['MAX_COST']
z = ihpxp2 ['MIN_COST']

y_axis = np.arange(len(X))

plt.barh(y_axis - 0.2, y, 0.4, label = 'MAX COST')
plt.barh(y_axis + 0.2, z, 0.4, label = 'MIN COST')

plt.yticks(y_axis, X)
plt.ylabel("INPATIENT SERVICE")
plt.xlabel("COST OF SERVICES")
plt.title("COST OF MOST COMMONLY USED INPATIENT SERVICES ACROSS THE US")
plt.legend()
plt.show()
```

	DRG_Desc	AVG_COST	MIN_COST	MAX_COST
0	ESOPHAGITIS, GASTROENT & MISC DIGEST DISORDERS...	5044.13	2053.00	40759.18
1	KIDNEY & URINARY TRACT INFECTIONS W/O MCC	5311.70	2670.93	44546.75
2	G.I. HEMORRHAGE W CC	6618.58	2318.61	42308.31
3	SEPTICEMIA OR SEVERE SEPSIS WITHOUT MV >96 HOU...	7010.97	2778.57	121897.31
4	PULMONARY EDEMA AND RESPIRATORY FAILURE	8717.98	4685.36	128772.18

Figure 37: Code and partial output for assessing the difference in cost of the most availed inpatient services covered by Medicare in the US on a horizontal Bar graph

```
# Difference in Cost of Most Frequently used Inpatient Services in Indiana State

ihpxp3 = pd.read_sql_query("SELECT distinct ips.DRG_Desc, ROUND(AVG(ips.Avg_Mdcr_Pyt_Amt),2) AS AVG_COST,
                           ROUND(MIN(ips.Avg_Mdcr_Pyt_Amt),2) AS MIN_COST, ROUND(Max(ips.Avg_Mdcr_Pyt_Amt),2) AS MAX_COST
                           from Inpatient_2020_PrvSrv ips WHERE DRG_CD IN (871,291,470,177,193,872,189,378,392,690)
                           AND Prv_State = 'IN' group by ips.DRG_Cd,ips.DRG_Desc
                           ORDER BY ROUND(AVG(ips.Avg_Mdcr_Pyt_Amt),2)" , conn)
ihpxp3 = pd.DataFrame(ihpxp3)
print(ihpxp3.head())

X = ihpxp3 ['DRG_Desc']
y = ihpxp3 ['MAX_COST']
z = ihpxp3 ['MIN_COST']

y_axis = np.arange(len(X))

plt.barh(y_axis - 0.2, y, 0.4, label = 'MAX COST')
plt.barh(y_axis + 0.2, z, 0.4, label = 'MIN COST')

plt.yticks(y_axis, X)
plt.ylabel("INPATIENT SERVICE")
plt.xlabel("COST OF SERVICES")
plt.title("COST OF MOST COMMONLY USED INPATIENT SERVICES IN INDIANA STATE")
plt.legend()
plt.show()
```

	DRG_Desc	AVG_COST	MIN_COST	MAX_COST
0	ESOPHAGITIS, GASTROENT & MISC DIGEST DISORDERS...	4527.40	2827.09	11482.83
1	KIDNEY & URINARY TRACT INFECTIONS W/O MCC	4710.44	3518.74	6671.33
2	G.I. HEMORRHAGE W CC	6116.09	4954.94	13402.48
3	SEPTICEMIA OR SEVERE SEPSIS WITHOUT MV >96 HOU...	6301.14	5035.43	13684.58
4	PULMONARY EDEMA AND RESPIRATORY FAILURE	7811.01	6302.72	15831.58

Figure 38: Code and partial output for assessing the difference in cost of the most availed inpatient services covered by Medicare within cities in Indiana on a horizontal Bar graph

```
# Most expensive Outpatient Services

ihpxp = pd.read_sql_query("SELECT distinct APC_Desc as Outpatient_Serv, ROUND(AVG(Avg_Mdcr_Pymt_Amt),2) AS AVG_COST,
                          ROUND(MIN(Avg_Mdcr_Pymt_Amt),2) AS MIN_COST, ROUND(Max(Avg_Mdcr_Pymt_Amt),2) AS MAX_COST
                          from Outpatient_2020_PrvSrv where (Avg_Mdcr_Pymt_Amt != 0.0) group by APC_Cd,APC_Desc
                          ORDER BY ROUND(AVG(Avg_Mdcr_Pymt_Amt),2) DESC limit 10" , conn)
ihpxp = pd.DataFrame(ihpxp)
print(ihpxp.head())

X = ihpxp ['Outpatient_Serv']
y = ihpxp ['MAX_COST']
z = ihpxp ['MIN_COST']

y_axis = np.arange(len(X))

plt.barh(y_axis - 0.2, y, 0.4, label = 'MAX COST')
plt.barh(y_axis + 0.2, z, 0.4, label = 'MIN COST')

plt.yticks(y_axis, X)
plt.ylabel("OUTPATIENT SERVICE")
plt.xlabel("COST OF SERVICES")
plt.title("COST OF MOST EXPENSIVE OUTPATIENT SERVICES")
plt.legend()
plt.show()
```

	Outpatient_Serv	AVG_COST	MIN_COST	MAX_COST
0	Level 4 Blood Product Exchange and Related Ser...	43148.73	37281.32	53797.48
1	Cochlear Implant Procedure	31393.97	22172.11	47138.93
2	Level 2 ICD and Similar Procedures	30404.31	10629.67	47202.44
3	Implantation Wireless PA Pressure Monitor	27200.47	23396.99	33072.17
4	Level 4 Neurostimulator and Related Procedures	26744.63	14485.75	41360.33

Figure 39: Code and partial output for assessing the difference in cost of the most expensive outpatient services covered by Medicare in the US on a horizontal Bar graph

```
# Difference in Cost of Most Frequently used Outpatient Services in the US

opxp2 = pd.read_sql_query("SELECT distinct APC_Desc as Outpatient_Serv, ROUND(AVG(Avg_Mdcr_Pymt_Amt),2) AS AVG_COST,
                          ROUND(MIN(Avg_Mdcr_Pymt_Amt),2) AS MIN_COST, ROUND(Max(Avg_Mdcr_Pymt_Amt),2) AS MAX_COST
                          from Outpatient_2020_PrvSrv where (Avg_Mdcr_Pymt_Amt != 0.0) AND
                          APC_Cd IN (8011,5072,5191,5115,5491,5114,5302,5361,5193,5373) group by APC_Cd,APC_Desc
                          ORDER BY ROUND(AVG(Avg_Mdcr_Pymt_Amt),2)" , conn)
opxp2 = pd.DataFrame(opxp2)
print(opxp2.head())

X = opxp2 ['Outpatient_Serv']
y = opxp2 ['MAX_COST']
z = opxp2 ['MIN_COST']

y_axis = np.arange(len(X))

plt.barh(y_axis - 0.2, y, 0.4, label = 'MAX COST')
plt.barh(y_axis + 0.2, z, 0.4, label = 'MIN COST')

plt.yticks(y_axis, X)
plt.ylabel("OUTPATIENT SERVICE")
plt.xlabel("COST OF SERVICES")
plt.title("COST OF MOST COMMONLY USED OUTPATIENT SERVICES ACROSS THE US")
plt.legend()
plt.show()
```

	Outpatient_Serv	AVG_COST	MIN_COST	MAX_COST
0	Level 2 Excision/ Biopsy/ Incision and Drainage	1075.79	437.40	1657.13
1	Level 2 Upper GI Procedures	1221.80	460.73	1875.00
2	Level 3 Urology and Related Services	1388.32	606.22	2137.77
3	Level 1 Intraocular Procedures	1588.50	276.48	2415.98
4	Comprehensive Observation Services	1806.92	59.13	3065.11

Figure 40: Code and partial output for assessing the difference in cost of the most availed outpatient services covered by Medicare in the US on a horizontal Bar graph

```
# Difference in Cost of Most Frequently used Outpatient Services in Indiana State

opxp3 = pd.read_sql_query("SELECT distinct APC_Desc as Outpatient_Serv, ROUND(AVG(Avg_Mdcr_Pymt_Amt),2) AS AVG_COST,
                          ROUND(MIN(Avg_Mdcr_Pymt_Amt),2) AS MIN_COST, ROUND(Max(Avg_Mdcr_Pymt_Amt),2) AS MAX_COST
                          from Outpatient_2020_PrvSrv where (Avg_Mdcr_Pymt_Amt != 0.0) AND
                          APC_Cd IN (8011,5072,5191,5115,5491,5114,5302,5361,5193,5373) AND
                          Prv_State = 'IN' group by APC_Cd,APC_Desc ORDER BY ROUND(AVG(Avg_Mdcr_Pymt_Amt),2)" , conn)
opxp3 = pd.DataFrame(opxp3)
print(opxp3.head())

X = opxp3 ['Outpatient_Serv']
y = opxp3 ['MAX_COST']
z = opxp3 ['MIN_COST']

y_axis = np.arange(len(X))

plt.barh(y_axis - 0.2, y, 0.4, label = 'MAX COST')
plt.barh(y_axis + 0.2, z, 0.4, label = 'MIN COST')

plt.yticks(y_axis, X)
plt.ylabel("OUTPATIENT SERVICE")
plt.xlabel("COST OF SERVICES")
plt.title("COST OF MOST COMMONLY USED OUTPATIENT SERVICES IN INDIANA STATE")
plt.legend()
plt.show()
```

	Outpatient_Serv	AVG_COST	MIN_COST	MAX_COST
0	Level 2 Excision/ Biopsy/ Incision and Drainage	1054.05	961.23	1148.50
1	Level 2 Upper GI Procedures	1185.83	980.92	1307.82
2	Level 3 Urology and Related Services	1365.18	1249.37	1443.70
3	Level 1 Intraocular Procedures	1500.68	276.48	1629.20
4	Comprehensive Observation Services	1759.91	1561.68	1965.49

Figure 41: Code and partial output for assessing the difference in cost of the most availed outpatient services covered by Medicare within cities in Indiana on a horizontal Bar graph

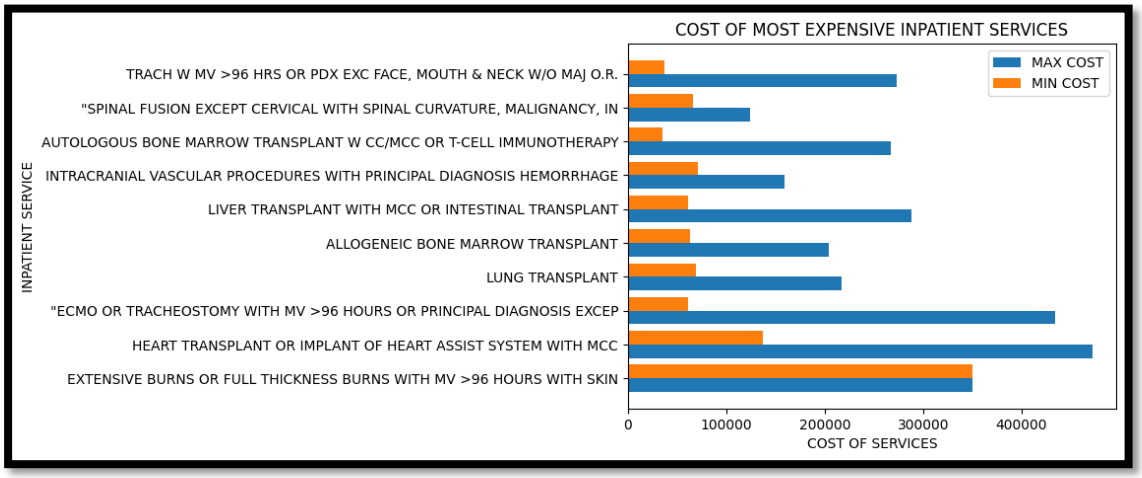

Figure 42: Differences in cost of the most expensive inpatient services covered by Medicare in the US visualized on a horizontal Bar graph

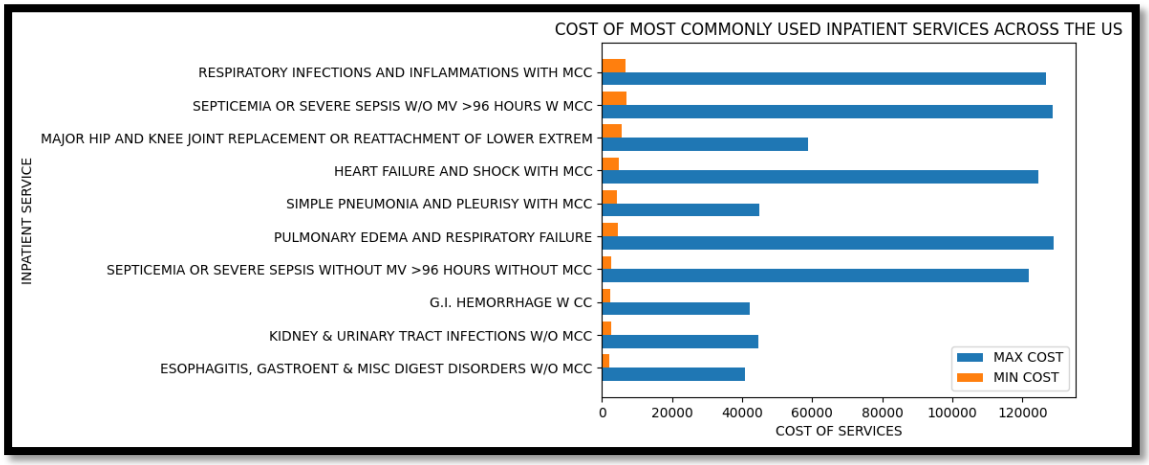

Figure 43: Differences in cost of the most availed inpatient services covered by Medicare in the US visualized on a horizontal Bar graph

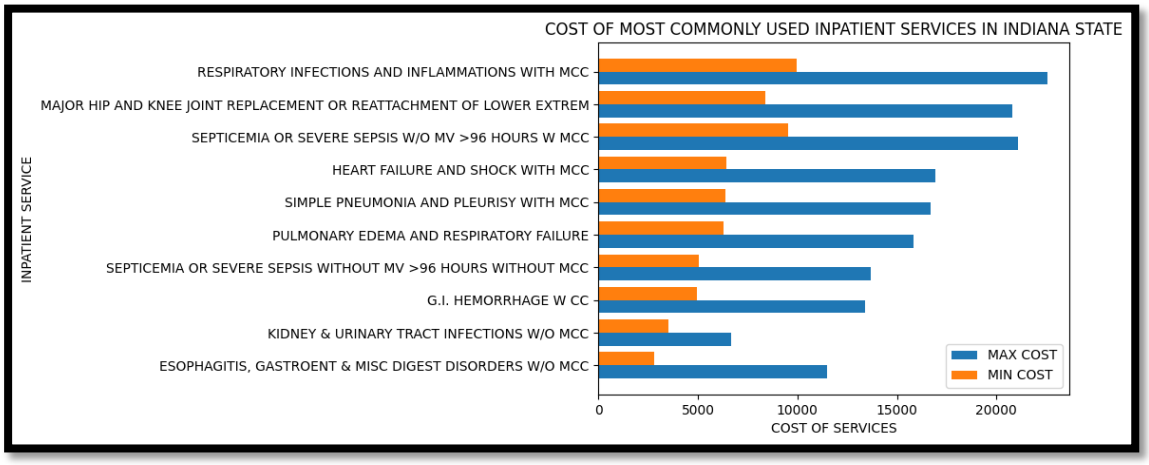

Figure 44: Differences in cost of the most availed inpatient services covered by Medicare within cities in Indiana visualized on a horizontal Bar graph

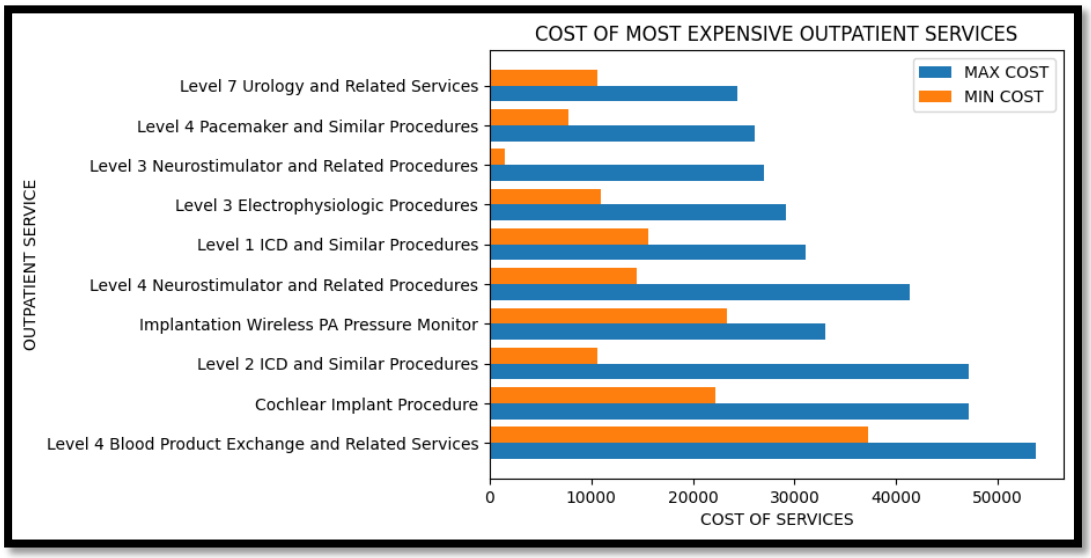

Figure 45: Differences in cost of the most expensive outpatient services covered by Medicare in the US visualized on a horizontal Bar graph

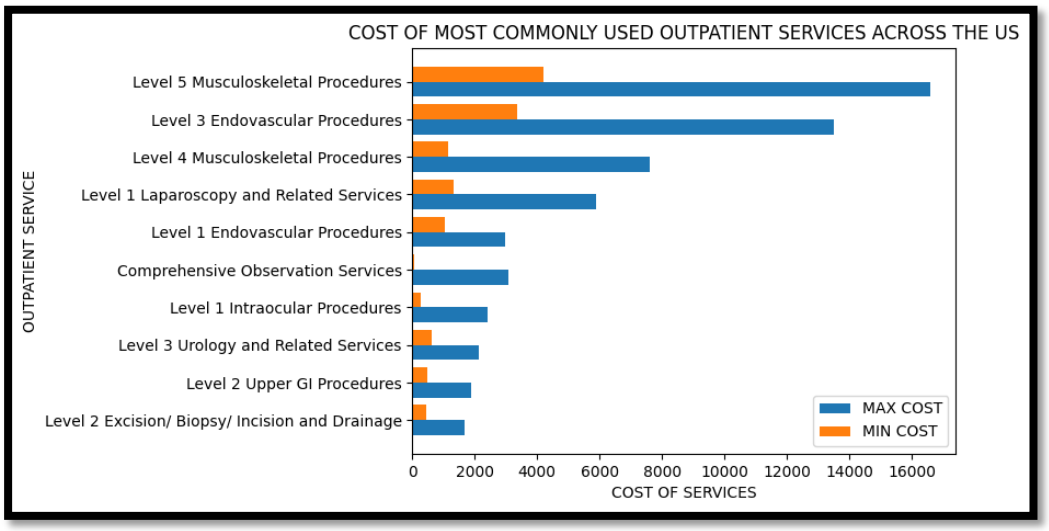

Figure 46: Differences in cost of the most availed outpatient services covered by Medicare in the US visualized on a horizontal Bar graph

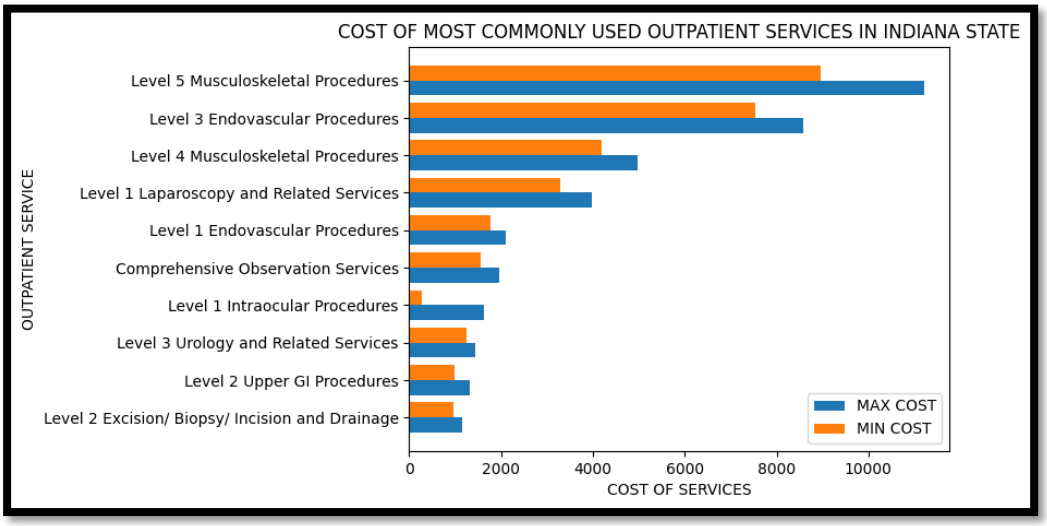

Figure 47: Differences in cost of the most availed outpatient services covered by Medicare within cities in Indiana visualized on a horizontal Bar graph

From the results, we can surmise that there is a steep difference in the cost of inpatient services across the US for the most expensive services as well as for those most availed. This difference can also be seen within cities in a state but to a lesser extent.

A similar pattern is seen with outpatient services to a lesser extent since outpatient services are not as expensive as inpatient services.

5.1.4 Distribution in PAC and Hospice Patients by Gender

Next, we assessed similar differences in access to PAC and Hospice services. We determined this by looking at distribution of patients by gender who availed SNF and Hospice facilities in different states and the top 20 cities with the highest male patient counts.

The code and the results for the same are provided below:

```
snf_mafm = pd.read_sql_query(" SELECT STATE, ROUND(((SUM(BENE_MALE_PCT)/COUNT(BENE_MALE_PCT))*100),2)
                               AS MALE_BENE,ROUND(((SUM(BENE_FEML_PCT)/COUNT(BENE_FEML_PCT))*100),2) AS FEMALE_BENE
                               FROM MD_PAC_SNF_2020 WHERE PRVDR_NAME != 'STATE TOTAL' AND STATE != 'NATIONAL TOTAL'
                               GROUP BY STATE ORDER BY ((SUM(BENE_MALE_PCT)/COUNT(BENE_MALE_PCT))*100) DESC" , conn)
snf_mafm = pd.DataFrame(snf_mafm)
snf_mafm.head()

trace1 = go.Bar(
    x=snf_mafm.STATE,
    y=snf_mafm.MALE_BENE,
    marker=dict(color='#e584f7'),
    name='Percent Male beneficiaries'
)
trace2 = go.Bar(
    x=snf_mafm.STATE,
    y=snf_mafm.FEMALE_BENE,
    marker=dict(color='#a0f784'),
    name='Percent Female beneficiaries'
)

data = [trace1, trace2]
layout = go.Layout(barmode='group', legend=dict(orientation='h'), title='Percent of Male and Female Patients in 2020 in
                  SNFs in different states')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```

Figure 48: Code for assessing differences in access to SNF facilities by gender at the state level, on a bar graph

```
snf_mafmct = pd.read_sql_query(" SELECT PRVDR_CITY, ROUND(((SUM(BENE_MALE_PCT)/COUNT(BENE_MALE_PCT))*100),2) AS MALE_BENE,
                                 ROUND(((SUM(BENE_FEML_PCT)/COUNT(BENE_FEML_PCT))*100),2) AS FEMALE_BENE
                                 FROM MD_PAC_SNF_2020 WHERE PRVDR_NAME != 'STATE TOTAL' AND STATE != 'NATIONAL TOTAL'
                                 GROUP BY PRVDR_CITY ORDER BY COUNT(BENE_MALE_PCT) DESC LIMIT 20" , conn)
snf_mafmct = pd.DataFrame(snf_mafmct)
snf_mafmct.head()

trace1 = go.Bar(
    x=snf_mafmct.PRVDR_CITY,
    y=snf_mafmct.MALE_BENE,
    marker=dict(color='#e584f7'),
    name='Percent Male beneficiaries'
)
trace2 = go.Bar(
    x=snf_mafmct.PRVDR_CITY,
    y=snf_mafmct.FEMALE_BENE,
    marker=dict(color='#a0f784'),
    name='Percent Female beneficiaries'
)

data = [trace1, trace2]
layout = go.Layout(barmode='group', legend=dict(orientation='h'), title='Percent of Male and Female Patients in 2020
                    in SNFs in the Top 20 Cities across the US')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```

Figure 49: Code for assessing differences in access to SNF facilities by gender in cities with the cities with the top 20 highest male patient counts, on a bar graph

The codes for differences in access to Hospice services are included in Appendix B – Section IV.

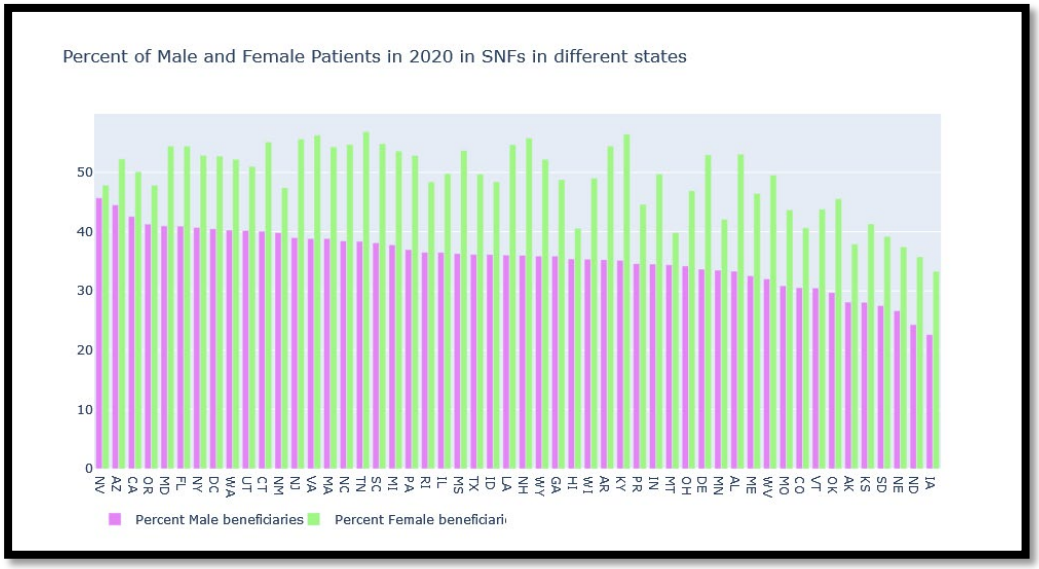

Figure 50: Differences in access to SNF facilities by gender at the state level, on a bar graph

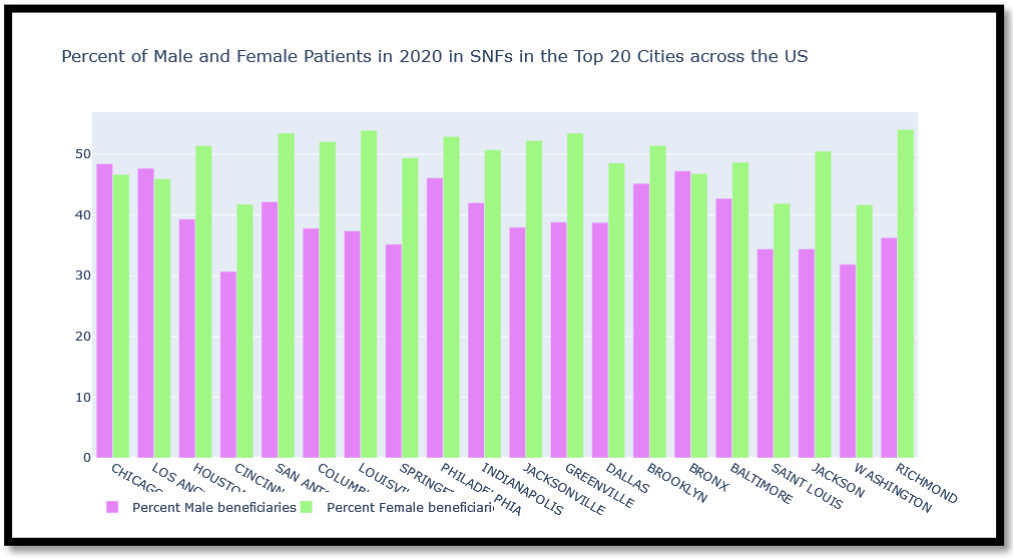

Figure 51: Differences in access to SNF facilities by gender in cities with the top 20 male patient counts, on a bar graph

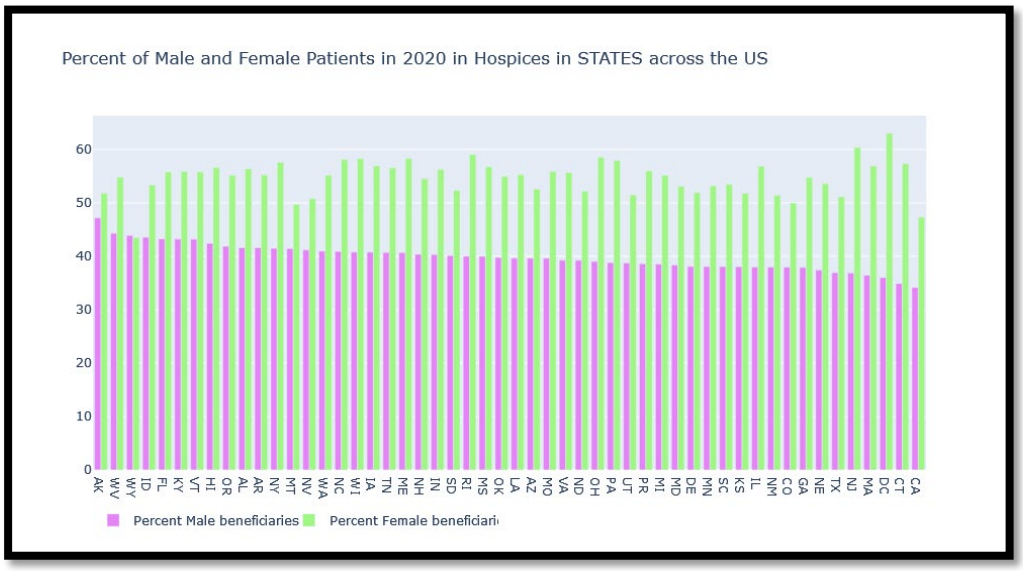

Figure 52: Differences in access to SNF facilities by gender at the state level, on a bar graph

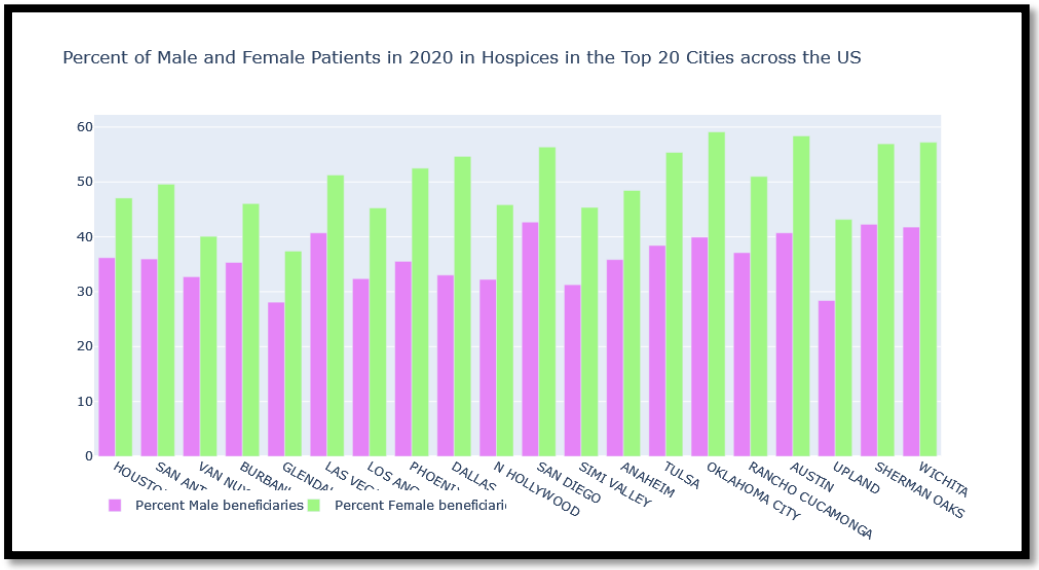

Figure 53: Differences in access to Hospice facilities by gender in cities with the top 20 male patient counts, on a bar graph

Based on the results we can conclude that female patients consistently access PAC and Hospice facilities more than male patients at all geographic levels.

5.1.5 Distribution in PAC and Hospice Patients by Ethnicity

In a similar fashion, we assessed differences in access to PAC and Hospice services by ethnicity at various geographic levels.

The code and the results for the same are provided below:

```
snf_ethn = pd.read_sql_query("SELECT ROUND(BENE_RACE_WHT_PCT*100,2) AS WHITE,
                              ROUND(BENE_RACE_BLACK_PCT*100,2) AS AFRICAN_AMERICAN,
                              ROUND(BENE_RACE_API_PCT*100,2) AS ASIAN_PACIFIC_ISLANDER,
                              ROUND(BENE_RACE_HSPNC_PCT*100,2) AS HISPANIC,
                              ROUND(BENE_RACE_NATIND_PCT*100,2) AS AMERICAN_INDIAN,
                              ROUND(BENE_RACE_OTHR_PCT*100,2) AS OTHER FROM MD_PAC_SNF_2020
                              WHERE STATE = 'NATIONAL TOTAL'" , conn)
snf_ethn = pd.DataFrame(snf_ethn)
snf_ethn = snf_ethn.T
snf_ethn.columns=['Percent']
snf_ethn = snf_ethn.assign(Race=['WHITE', 'AFRICAN_AMERICAN', 'ASIAN_PACIFIC_ISLANDER','HISPANIC','AMERICAN_INDIAN','OTHER'])
snf_ethn

fig, sc = plt.subplots(figsize=(8, 8))
sc.pie(snf_ethn['Percent'], labels=snf_ethn['Race'], autopct='%.1f%%')
sc.set_title("Access to SNF Level Care by Ethnicity At the National Level")
plt.show()
```

Figure 54: Code for assessing differences in access to SNF facilities by ethnicity at the national level using a pie chart

```
snf_ethn_st = pd.read_sql_query("SELECT STATE, ROUND(BENE_RACE_WHT_PCT*100,2) AS WHITE,
                                 ROUND(BENE_RACE_BLACK_PCT*100,2) AS AFRICAN_AMERICAN,
                                 ROUND(BENE_RACE_API_PCT*100,2) AS ASIAN_PACIFIC_ISLANDER,
                                 ROUND(BENE_RACE_HSPNC_PCT*100,2) AS HISPANIC,
                                 ROUND(BENE_RACE_NATIND_PCT*100,2) AS AMERICAN_INDIAN,
                                 ROUND(BENE_RACE_OTHR_PCT*100,2) AS OTHER FROM MD_PAC_SNF_2020
                                 WHERE SMRY_CTGRY = 'STATE'" , conn)
snf_ethn_st = pd.DataFrame(snf_ethn_st)
snf_ethn_st.head()
```

	STATE	WHITE	AFRICAN_AMERICAN	ASIAN_PACIFIC_ISLANDER	HISPANIC	AMERICAN_INDIAN	OTHER
0	AK	78.0	3.0	3.0	2.0	9.0	2.0
1	AL	80.0	18.0	0.0	0.0	0.0	0.0
2	AR	86.0	10.0	0.0	0.0	0.0	0.0
3	AZ	81.0	2.0	0.0	7.0	6.0	1.0
4	CA	62.0	8.0	9.0	16.0	0.0	1.0

Figure 55: Code and partial output to assess differences in access to SNF facilities by ethnicity in each state

```
snf_ethnin = pd.read_sql_query("SELECT ROUND(BENE_RACE_WHT_PCT*100,2) AS WHITE,
                                ROUND(BENE_RACE_BLACK_PCT*100,2) AS AFRICAN_AMERICAN,
                                ROUND(BENE_RACE_API_PCT*100,2) AS ASIAN_PACIFIC_ISLANDER,
                                ROUND(BENE_RACE_HSPNC_PCT*100,2) AS HISPANIC,
                                ROUND(BENE_RACE_NATIND_PCT*100,2) AS AMERICAN_INDIAN,
                                ROUND(BENE_RACE_OTHR_PCT*100,2) AS OTHER FROM MD_PAC_SNF_2020
                                WHERE STATE = 'IN' and SMRY_CTGRY =  'STATE'" , conn)
snf_ethnin = pd.DataFrame(snf_ethnin)
snf_ethnin
snf_ethnin = snf_ethnin.T
snf_ethnin.columns=['Percent']
snf_ethnin = snf_ethnin.assign(Race=['WHITE', 'AFRICAN_AMERICAN', 'ASIAN_PACIFIC_ISLANDER','HISPANIC','AMERICAN_INDIAN',
                                     'OTHER'])
snf_ethnin

fig, sc = plt.subplots(figsize=(8, 8))
sc.pie(snf_ethnin['Percent'], labels=snf_ethnin['Race'], autopct='%.1f%%')
sc.set_title("Access to SNF Level Care by Ethnicity in INDIANA State")
plt.show()
```

Figure 56: Code for assessing differences in access to SNF facilities by ethnicity in the state of Indiana, using a pie chart

The codes for differences in access to Hospice services are included in Appendix B – Section V.

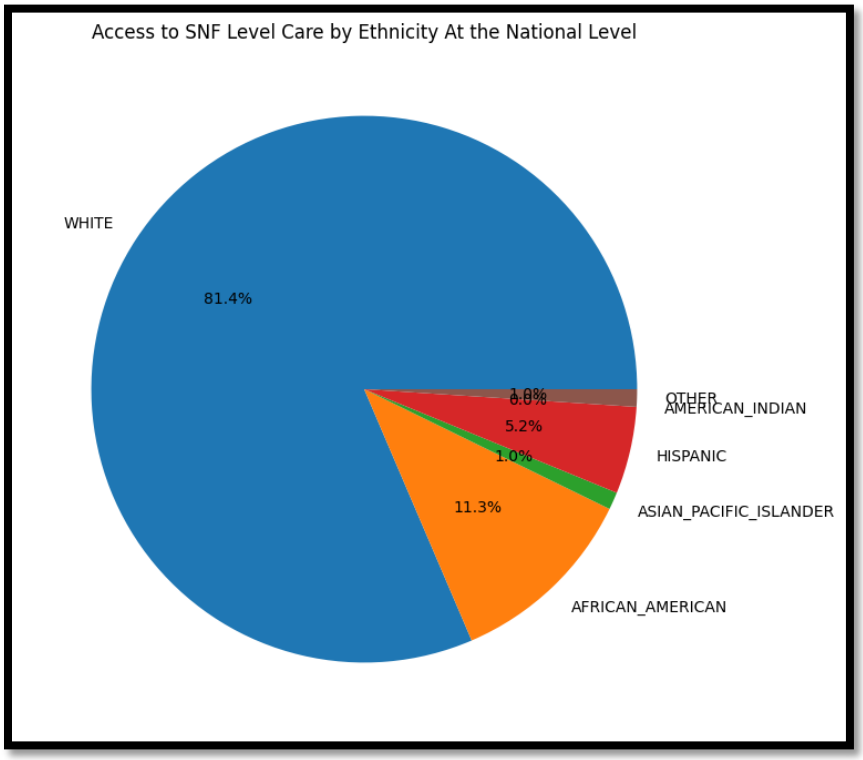

Figure 57: Differences in access to SNF facilities by ethnicity at the national level using a pie chart

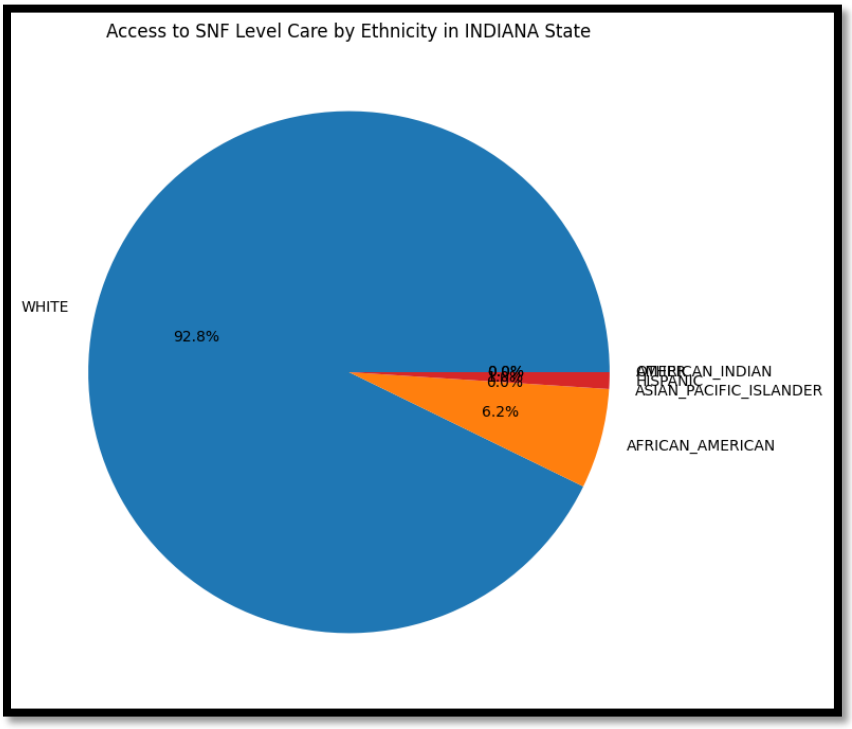

Figure 58: Differences in access to SNF facilities by ethnicity in Indiana State, using a pie chart

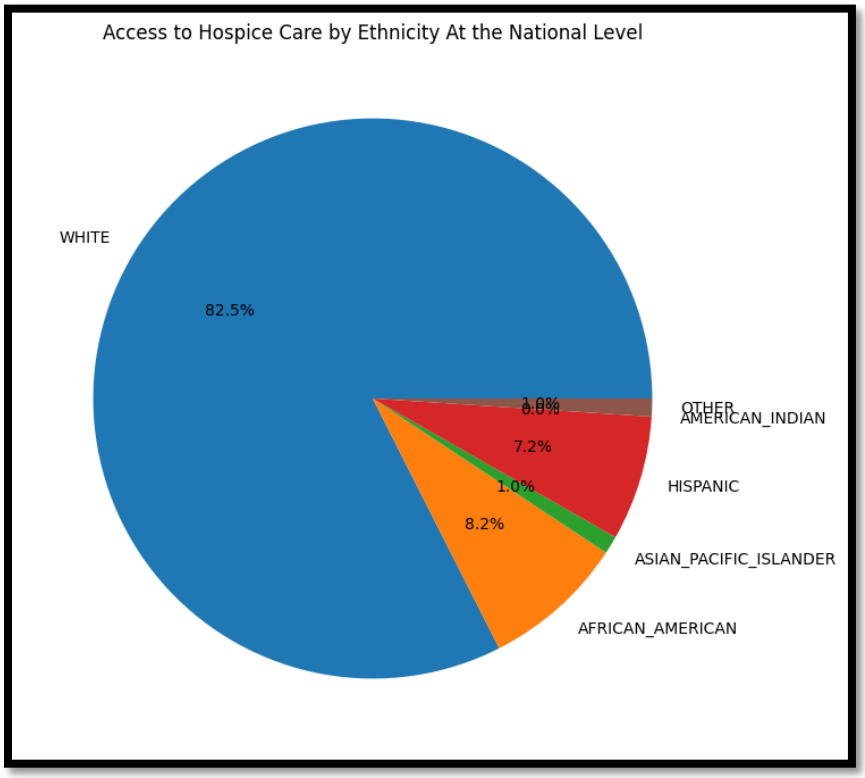

Figure 59: Differences in access to Hospice facilities by ethnicity at the national level using a pie chart

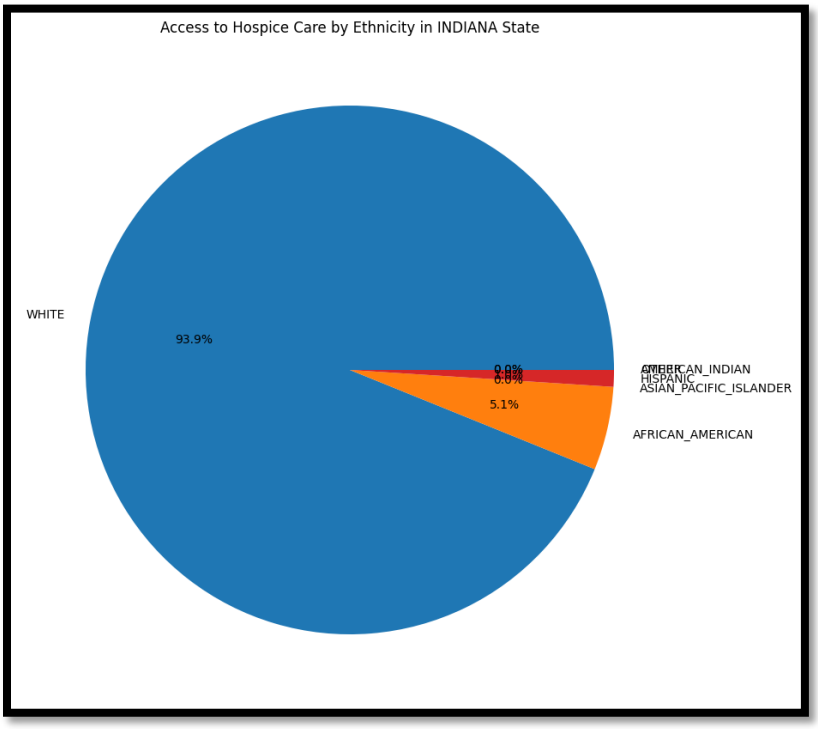

Figure 60: Differences in access to Hospice facilities by ethnicity in Indiana State, using a pie chart

The results demonstrate that more than 80% of PAC and Hospice facilities are accessed by White patients. African American and Hispanic patients are the next highest with some differences between national and Indiana state levels depending on facility type. Other ethnicities together make up less than 5% of the demographic at most geographic levels, except in certain specific states where some of these ethnicities have a larger presence.

5.2 Clustering Post-Acute Care and Hospice facilities to assess differences in access to care

Another one of our proposed research questions was whether we could show a difference in access to PAC and Hospice services available to Medicare patients that could be discerned by looking at characteristics of individual facilities such as details of hospital rating criteria, patient episode lengths, days of services offered, percentage of patients availing services by gender, ethnicity and chronic conditions treated.

We dropped the Hospital rating criteria early on in our analysis, since the data available provided ratings for a very small number of providers.

Using the other criteria mentioned above, we attempted to cluster the 5 different types of facilities into distinct groups with unique characteristics. The main steps involved in this process were:

a. Using Birch Clustering to segregate different types of facilities
b. Visualization of Cluster results based on the top 2 contributing components using Principal Component Analysis (PCA)
c. Studying distribution of certain features in individual clusters to attempt to identify features contributing most variation

We discuss each step in further detail for one type of facility below with accompanying codes and visualizations.
For the other four types, the summarized results and visualizations are included here, and the codes are included in Appendix B - Section VI.

5.2.1 Clustering SNF Facilities using Birch

1. For each facility subset we pulled a list of about 30-32 features and added them into a data frame.

```
snf_cl = pd.read_sql_query("SELECT PRVDR_ID,PRVDR_NAME,PRVDR_CITY,STATE,BENE_DSTNCT_CNT,TOT_EPSD_STAY_CNT,TOT_SRVC_DAYS,
                           TOT_CHRG_AMT,TOT_ALOWD_AMT,TOT_MDCR_PYMT_AMT,TOT_MDCR_STDZD_PYMT_AMT,BENE_AVG_AGE,BENE_MALE_PCT,
                           BENE_FEML_PCT,BENE_RACE_WHT_PCT,BENE_RACE_BLACK_PCT,BENE_RACE_API_PCT,BENE_RACE_HSPNC_PCT,
                           BENE_RACE_NATIND_PCT,BENE_RACE_OTHR_PCT,BENE_AVG_RISK_SCRE,BENE_CC_AF_PCT,BENE_CC_ALZHMR_PCT,
                           BENE_CC_ASTHMA_PCT,BENE_CC_CNCR_PCT,BENE_CC_CHF_PCT,BENE_CC_CKD_PCT,BENE_CC_COPD_PCT,
                           BENE_CC_DPRSSN_PCT,BENE_CC_DBTS_PCT,BENE_CC_IHD_PCT,BENE_CC_OPO_PCT,BENE_CC_RAOA_PCT,
                           BENE_CC_SZ_PCT,BENE_CC_STROK_PCT FROM MD_PAC_SNF_2020 WHERE SMRY_CTGRY = 'PROVIDER'" , conn)
snf_cl = pd.DataFrame(snf_cl)
snf_cl.head()
```

	PRVDR_ID	PRVDR_NAME	PRVDR_CITY	STATE	BENE_DSTNCT_CNT	TOT_EPSD_STAY_CNT	TOT_SRVC_DAYS	TOT_CHRG_AMT	TOT_ALOWD_AMT
0	25010	KETCHIKAN MED CTR NEW HORIZONS TRANSITIONAL CARE	KETCHIKAN	AK	21	47	693	1464798	502156
1	25018	PROVIDENCE TRANSITIONAL CARE CENTER	ANCHORAGE	AK	159	351	6233	14781270	4211956
2	25020	DENALI CENTER	FAIRBANKS	AK	75	144	2314	2565277	1483222
3	25021	HERITAGE PLACE	SOLDOTNA	AK	39	88	1337	1910608	935727
4	25025	PRESTIGE CARE & REHAB CENTER OF ANCHORAGE	ANCHORAGE	AK	172	357	5693	5859756	3972806

5 rows × 35 columns

Figure 61: Creating a data frame for 35 features for SNF facilities for cluster analysis

2. Provider identifier details such as Provider ID, Provider Name, City and State are dropped from the clustering data frame.

```
from sklearn.cluster import Birch, MiniBatchKMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_samples, silhouette_score

X = snf_cl
X.index = X.PRVDR_ID
X = X.drop(['PRVDR_ID', 'PRVDR_NAME', 'PRVDR_CITY', 'STATE'], axis=1)
X = X.dropna()

birch_silhouette_scores = []
for n_cluster in range(2, 8):

    birch_silhouette_scores.append(
        silhouette_score(X, Birch(n_clusters = n_cluster).fit_predict(X)))
k = np.arange(2,8)

plt.bar(k, birch_silhouette_scores)
plt.title('SNF Facilities - Assessing Cluster Numbers with BIRCH', fontsize = 10)
plt.xlabel('Number of Clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```

Figure 62: Dropping provider identifier details and running a loop to determine best number of clusters to segregate SNF facilities

3. To assess the best number of clusters for each dataset, a loop was used to assess which cluster number gave the best silhouette score. The results were visualized on a Bar graph.

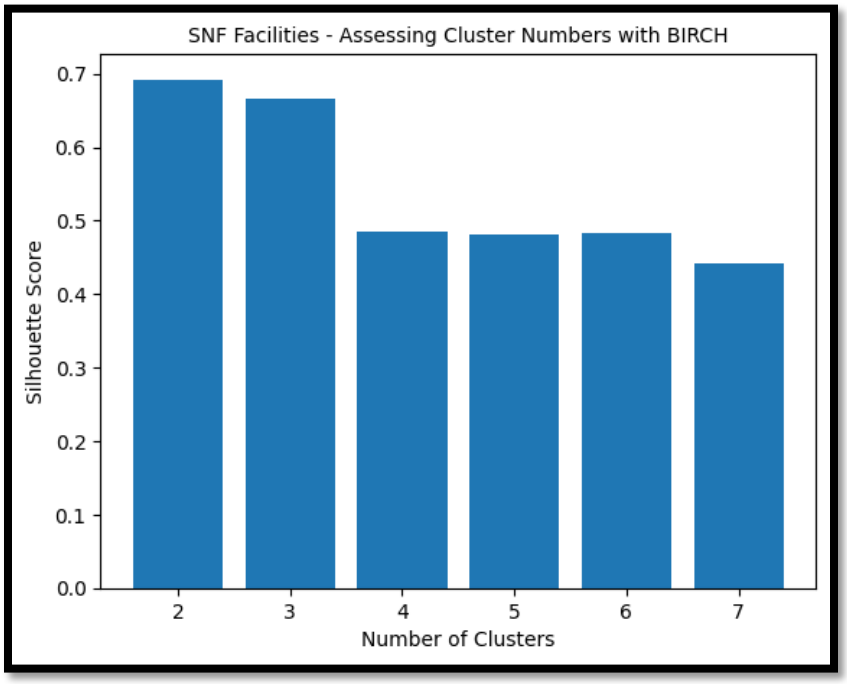

Figure 63: Visualizing the clustering loop results in determining the best number of clusters to segregate SNF facilities. N=2 gave the best results.

4. Clustering is redone with the appropriate number of clusters and the silhouette score is printed. The number of datapoints in each cluster are then determined.

```
# apply Birch clustering for SNF
from sklearn.cluster import Birch, MiniBatchKMeans
birch1 = Birch(branching_factor=50, n_clusters=2, threshold=1.5)

X['cluster1'] = birch1.fit_predict(X)

# re-fit our model
birch1.fit(X)

# Average silhouette score
silhouette_score_average1 = silhouette_score(X, birch1.predict(X))

print(silhouette_score_average1)
```

```
0.6919852248030971
```

```
#To print number of datapoints in each SNF Cluster

snf1 = (X[X['cluster1'] == 0])
snf2 = (X[X['cluster1'] == 1])

print('Size of SNF Clusters:', 'Cluster 1:', len(snf1), ', Cluster 2:', len(snf2))
```

```
Size of SNF Clusters: Cluster 1: 1912 , Cluster 2: 12527
```

Figure 64: Clustering is redone with an appropriate number of clusters (n=2 for SNFs) and the silhouette scores are printed. The number of data points in each cluster is also printed.

5.2.2 Cluster Visualization with Principal Component Analysis

1. To perform PCA, the features from the data frame are first transformed to a standardized scale to minimize variation due to difference in scale of various features.

Dimension Reduction with PCA

```
from sklearn.preprocessing import StandardScaler

xp = StandardScaler().fit_transform(X) # normalizing the features
print(xp.shape)
feat_cols = ['feature'+str(i) for i in range(xp.shape[1])]
norm_xp = pd.DataFrame(xp,columns=feat_cols)
norm_xp.tail()
```

```
(14439, 32)
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	...	feature22	feature23	feature24
14434	0.687630	1.058419	1.293282	0.871859	0.975487	0.884694	1.117964	0.439536	0.162843	0.474731	...	0.021443	0.814088	-0.358657
14435	-0.786400	-0.744985	-0.661468	-0.589314	-0.578463	-0.574546	-0.658367	0.227358	-2.211643	-2.373675	...	-1.904176	0.158425	0.174005
14436	-0.826784	-0.864892	-0.815286	-0.665301	-0.736935	-0.734409	-0.827418	1.288249	0.223727	0.427258	...	-1.320655	0.754483	-0.891318
14437	-0.362364	-0.212597	0.050672	-0.029139	0.051942	0.013990	-0.029691	0.015180	-0.141578	0.712098	...	-0.737134	0.873694	-0.944584
14438	-0.523901	-0.610689	-0.583476	-0.489076	-0.532504	-0.529082	-0.604997	1.288249	-0.202463	0.759572	...	-1.087246	0.158425	-0.252124

5 rows × 32 columns

Figure 65: Standardized scalar transformation of SNF features for PCA analysis.

2. A PCA loop is run by sequentially increasing the number of components until 90% of the variation is explained. The variation explained by each component in descending order and the final number of components required to explain 90% of the variation are printed.

```
# Determining Minimum number of components required to account for 90% explained variation

from sklearn.decomposition import PCA

for i in range(2,32):
    pca_xp = PCA(n_components=i)
    pcarun_xp = pca_xp.fit_transform(xp)
    fit_xp = pca_xp.fit(xp)
    X["cluster1"].unique()
    norm_xp["cluster1"] = X["cluster1"].values
    var_list = list(pca_xp.explained_variance_ratio_)
    exp_var = sum(var_list)
    if exp_var >= 0.9:
        print('Explained variation per principal component:', var_list)
        print('')
        print('Total variation explained with', i, 'components for SNF Clusters: ', exp_var)
        break
    i+=1
```

```
Explained variation per principal component: [0.29626352456649935, 0.13599800281525645, 0.11164933056184331, 0.04464113269304
578, 0.04161678946897887, 0.037224737143679486, 0.033208972673223806, 0.03239613378507404, 0.029898787600833695, 0.0238897069
0102609, 0.023492876576471384, 0.019031324094421084, 0.01776503941552845 2, 0.01553557976160553 5, 0.013745241732299488, 0.0133
03446975810977, 0.0130641823132953 1]

Total variation explained with 17 components for SNF Clusters:  0.902724809078893 1
```

Figure 66: Looped PCA analysis of scaled SNF features to determine the number of components that account for 90% of the explained variation. The explained variation for each principal component in descending order and the result is printed in the output.

3. A scatter plot is used to visualize the clusters with the top 2 principal components accounting for the maximum variation in the data.

```
# Visualize SNF Clusters

def cluster_visualization(df, columns, labels):

    df = df[columns]
    pca = PCA(17)
    pca.fit(df)
    X_PCA = pca.transform(df)
    X_PCA.shape
    print(X_PCA.shape)
    print('Explained variation for Top 2 principal components:', var_list[0] + var_list[1] )
    print('')

    x, y = X_PCA[:, 0], X_PCA[:, 1]

    colors = {0: 'red'
              ,1: 'blue'
             }

    names = {0: 'Cluster 1'
             ,1: 'Cluster 2'
            }

    df = pd.DataFrame({'x': x, 'y':y, 'label':labels})
    groups = df.groupby('label')

    fig, ax = plt.subplots(figsize=(8, 5))

    for name, group in groups:
        ax.plot(group.x, group.y, marker='o', linestyle='', ms=5,
                color=colors[name],label=names[name], mec='none')
        ax.set_aspect('auto')
        ax.tick_params(axis='x',which='both',bottom='off',top='off',labelbottom='off')
        ax.tick_params(axis= 'y',which='both',left='off',top='off',labelleft='off')

    ax.legend()
    ax.set_title("SNF Clustering Visualization - with Top 2 Principal Components")
    plt.show()

cluster_visualization(norm_xp, columns = norm_xp.columns[:-1], labels = norm_xp["cluster1"])
```

```
(14439, 17)
Explained variation for Top 2 principal components: 0.43226152738175705
```

Figure 67: Code for visualizing SNF clusters on a scatter plot using the top 2 principal components that account for the maximum amount of explained variation. The explained variation is printed in the output.

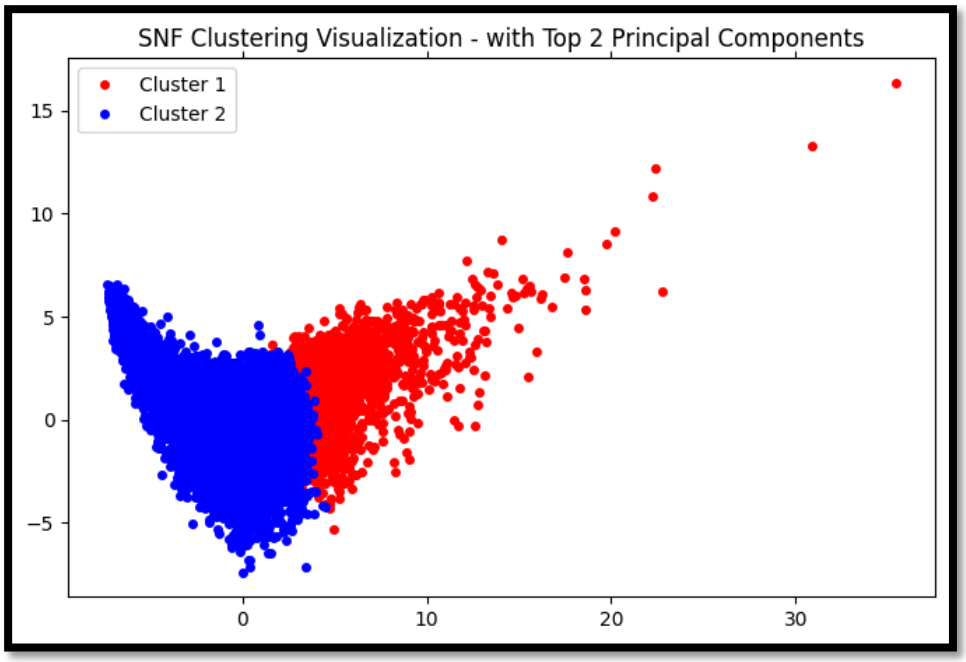

Figure 68: Scatter plot to visualize SNF clusters using the top 2 principal components that account for the maximum amount of explained variation

5.2.3 Studying Distribution of datapoints for Individual clusters to determine important components

For facility types where 2 or more clusters showed distinct distribution of datapoints, the distribution of the following features was studied for each cluster using a grouped boxplot:

1. Total Medicare charges
2. Distinct number of patients
3. Total Service days
4. Percent of Patients who have cancer as a chronic or primary diagnosed condition

```
# Analyze Distribution of Some Variables for Individual Clusters

import plotly.graph_objects as go

f1 = go.Box(y=snf1['TOT_CHRG_AMT'], name='Cluster 1')
f2 = go.Box(y=snf2['TOT_CHRG_AMT'], name='Cluster 2')
f3 = go.Box(y=snf1['BENE_DSTNCT_CNT'], name='Cluster 1')
f4 = go.Box(y=snf2['BENE_DSTNCT_CNT'], name='Cluster 2')
f5 = go.Box(y=snf1['TOT_SRVC_DAYS'], name='Cluster 1')
f6 = go.Box(y=snf2['TOT_SRVC_DAYS'], name='Cluster 2')
f7 = go.Box(y=snf1['BENE_CC_CNCR_PCT'], name='Cluster 1')
f8 = go.Box(y=snf2['BENE_CC_CNCR_PCT'], name='Cluster 2')


fig = subplots.make_subplots(rows=2, cols=2, print_grid=False, subplot_titles=('SNF Medicare Charges','SNF Patient Counts',
                                                                               'SNF Service Days','Cancer Patients'))

fig.append_trace(f1, 1, 1);
fig.append_trace(f2, 1, 1);
fig.append_trace(f3, 1, 2);
fig.append_trace(f4, 1, 2);
fig.append_trace(f5, 2, 1);
fig.append_trace(f6, 2, 1);
fig.append_trace(f7, 2, 2);
fig.append_trace(f8, 2, 2);


fig['layout'].update(height=800, width=800, showlegend=False);
iplot(fig, filename='simple-subplot');
```

Figure 69: Code for grouped Boxplot to visualize distribution of datapoints for certain features in the individual SNF clusters

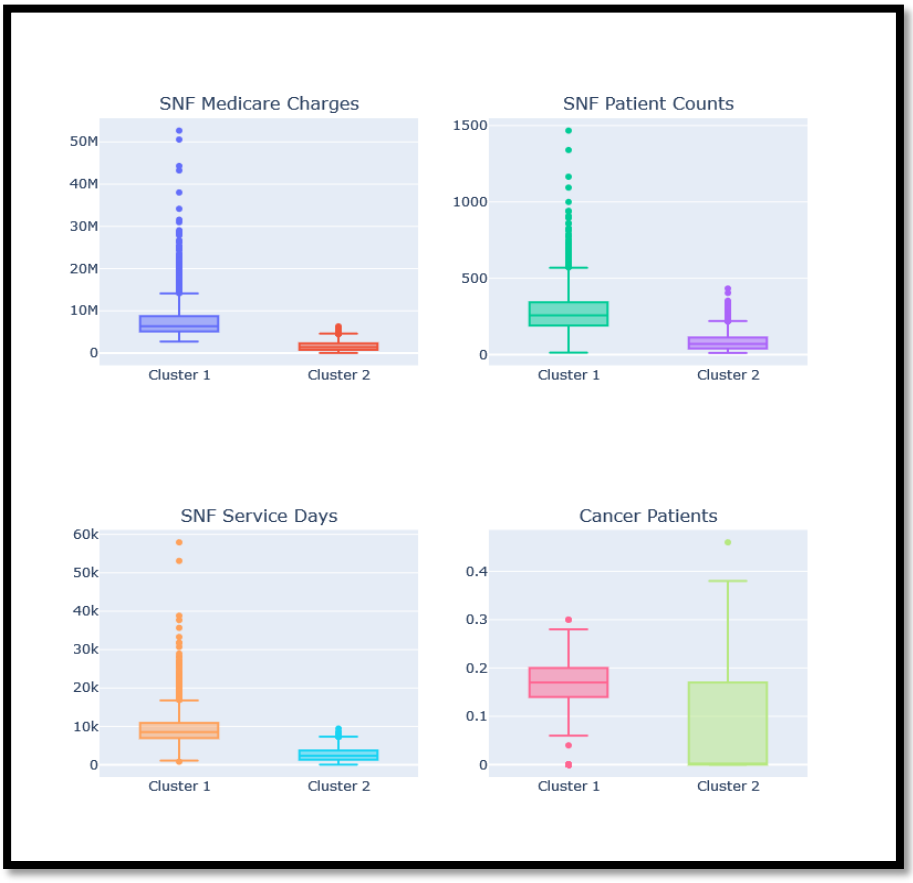

Figure 70: Grouped Boxplot to visualize distribution of datapoints for certain features in the individual SNF clusters

The clustering and PCA analysis for SNFs showed that there were 2 clusters of SNF facilities with some overlap, that showed differences in medicare charge, number of service days, and patient counts.

RESULTS FOR HOSPICE FACILITIES

	PRVDR_ID	PRVDR_NAME	PRVDR_CITY	STATE	BENE_DSTNCT_CNT	TOT_EPSD_STAY_CNT	TOT_SRVC_DAYS	TOT_CHRG_AMT	TOT_ALOWD_AMT
0	101500	COMMUNITY HOSPICE OF NORTHEAST FLORIDA INC	JACKSONVILLE	FL	6928	19590	422327	105703154	79398389
1	101502	HEARTLAND HOSPICE SERVICES	PLANTATION	FL	886	4268	111468	34766444	18056329
2	101504	HOSPICE BY THE SEA INC	BOCA RATON	FL	3320	10319	232574	67660868	46495815
3	101507	LIFEPATH HOSPICE	TAMPA	FL	5913	18013	397823	108821783	72256734
4	101508	SUNCOAST HOSPICE	CLEARWATER	FL	7705	23511	521160	195150195	96237637

5 rows × 33 columns

Figure 71: Data frame of 33 features for Hospice facilities for cluster analysis

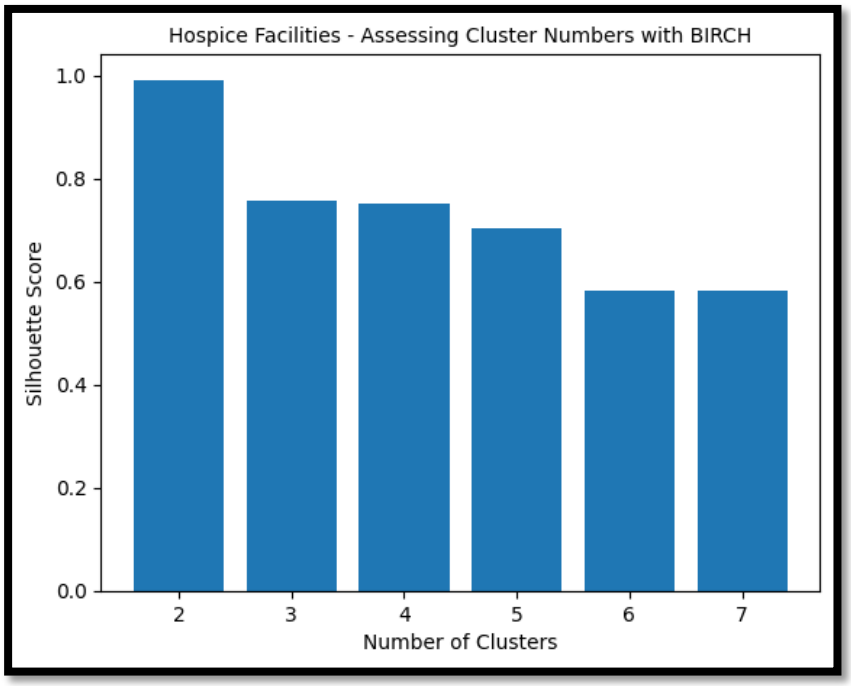

Figure 72: Visualizing clustering loop results in determining the best number of clusters to segregate Hospice facilities. N=2 gave the best results.

```
from sklearn.cluster import Birch, MiniBatchKMeans
birch2 = Birch(branching_factor=50, n_clusters=2, threshold=1.5)

X2['cluster2'] = birch2.fit_predict(X2)

birch1.fit(X2)

silhouette_score_average2 = silhouette_score(X2, birch1.predict(X2))

print(silhouette_score_average2)
```

```
0.9912003437764154
```

```
#To print number of datapoints in each Hospice Cluster

hos1 = (X2[X2['cluster2'] == 0])
hos2 = (X2[X2['cluster2'] == 1])

print('Size of Hospice Clusters:', 'Cluster 1:', len(hos1), ', Cluster 2:', len(hos2))
```

```
Size of Hospice Clusters: Cluster 1: 4689 , Cluster 2: 1
```

Figure 73: Clustering is redone with n=2 clusters for Hospice data with printed output of silhouette scores number of data points in each cluster after analysis.

```
# Determining Minimum number of components required to account for 90% explained variation

from sklearn.decomposition import PCA

for i in range(2,30):
    pca_xp2 = PCA(n_components=i)
    pcarun_xp2 = pca_xp2.fit_transform(xp2)
    fit_xp2 = pca_xp2.fit(xp2)
    X2["cluster2"].unique()
    norm_xp2["cluster2"] = X2["cluster2"].values
    var_list2 = list(pca_xp2.explained_variance_ratio_)
    exp_var2 = sum(var_list2)
    if exp_var2 >= 0.9:
        print('Explained variation per principal component:', var_list2)
        print('')
        print('Total variation explained with', i, 'components for Hospice Clusters: ', exp_var2)
        break
    i+=1
```

```
Explained variation per principal component: [0.2611089715632899, 0.08973329154713496, 0.060595521320192254, 0.05867131336041
3806, 0.04378443292283634, 0.04216158060732339, 0.03911911900844057, 0.0379512080039869l, 0.03686672605218474, 0.034156482875
98892, 0.032923833306650906, 0.031715140865281564, 0.030991904346482763, 0.02795316914736078, 0.025776332792674665, 0.0246179
64415543633, 0.023961253090050854]

Total variation explained with 17 components for Hospice Clusters:  0.9020882452258371
```

Figure 74: Looped PCA analysis showing a minimum of 17 components are required to account for 90% of explained variation in Hospice clusters.

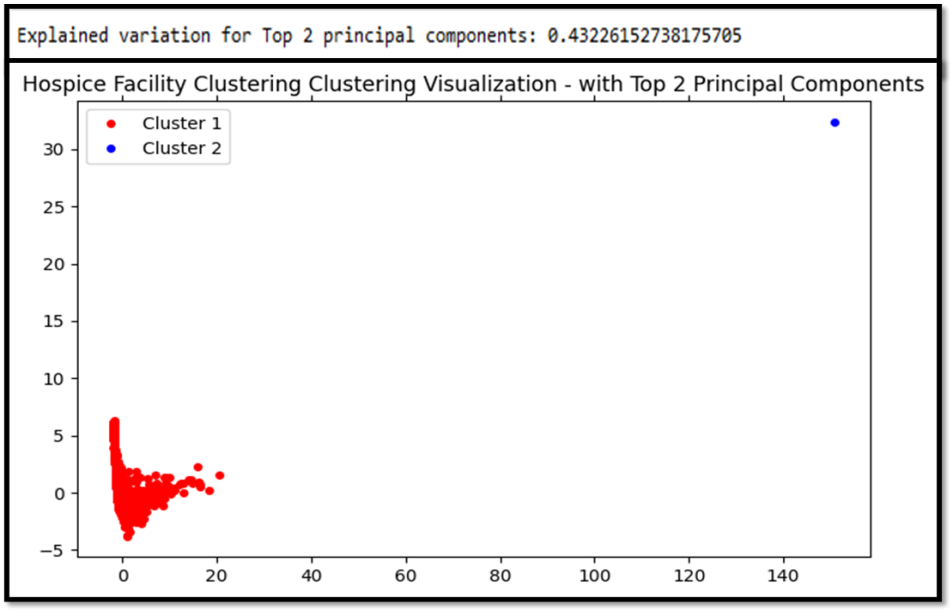

Figure 75: Scatter plot to visualize Hospice clusters using the top 2 principal components that account for the maximum amount of explained variation with printed output above the graph

The clustering and PCA analysis for Hospice facilities showed that even with significant variation in the top 2 principal components (26% and 8%), hospice facilities couldn't be segregated into more than a single cluster. This is bolstered by the fact that the 2nd cluster only included 1 provider.

RESULTS FOR HOME HEALTH FACILITIES

	PRVDR_ID	PRVDR_NAME	PRVDR_CITY	STATE	BENE_DSTNCT_CNT	TOT_EPSD_STAY_CNT	TOT_SRVC_DAYS	TOT_CHRG_AMT	TOT_ALOWD_AMT	TOT_
0	27001	PROVIDENCE IN HOME SERVICES	ANCHORAGE	AK	427	1027	26281	3299904	2218370	
1	27002	HOSPICE & HOME CARE OF JUNEAU	JUNEAU	AK	141	371	9664	812345	756322	
2	27006	FAIRBANKS MEMORIAL HOSPITAL HHA	FAIRBANKS	AK	144	360	9452	542215	674040	
3	27008	ANCORA HOME HEALTH & HOSPICE	WASILLA	AK	569	1611	42029	5818729	3705798	
4	27009	PETERSBURG MEDICAL CENTER HOME	PETERSBURG	AK	40	177	5071	525420	442058	

5 rows × 35 columns

Figure 76: Data frame of 35 features for Home Health facilities for cluster analysis

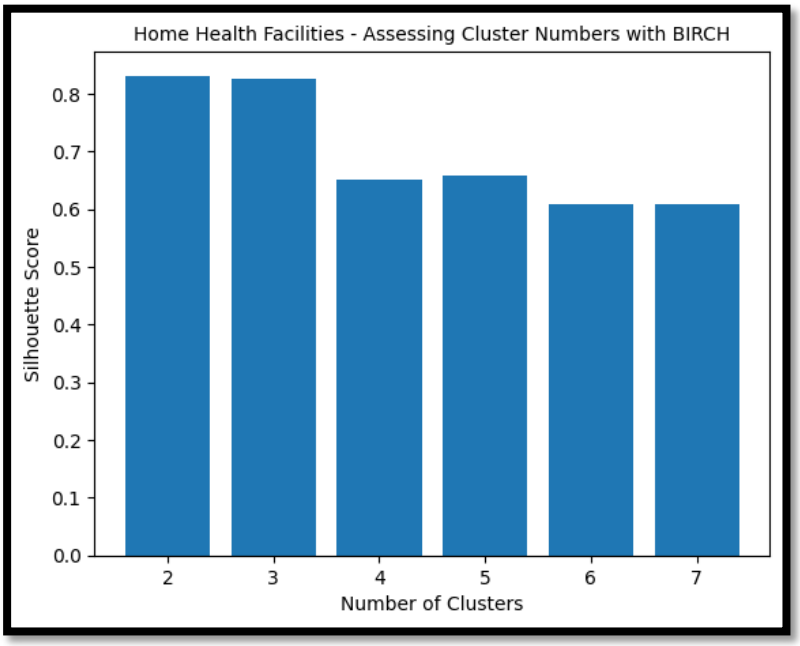

Figure 77: Visualizing clustering loop results to determine best number of clusters to segregate Home Health facilities. N=2 gave the best results.

```
from sklearn.cluster import Birch, MiniBatchKMeans
birch3 = Birch(branching_factor=50, n_clusters=2, threshold=1.5)

X3['cluster3'] = birch3.fit_predict(X3)

birch3.fit(X3)

silhouette_score_average3 = silhouette_score(X3, birch3.predict(X3))

print(silhouette_score_average3)
```

```
0.8312757494752304
```

```
#To print number of datapoints in each Home Health Cluster

hh1 = (X3[X3['cluster3'] == 0])
hh2 = (X3[X3['cluster3'] == 1])

print('Size of Home Health Clusters:', 'Cluster 1:', len(hh1), ', Cluster 2:', len(hh2))
```

```
Size of Home Health Clusters: Cluster 1: 426 , Cluster 2: 8165
```

Figure 78: Clustering redone with n=2 clusters for Home Health data with printed output of silhouette scores number of data points in each cluster after analysis.

```
# Determining Minimum number of components required to account for 90% explained variation

from sklearn.decomposition import PCA

for i in range(2,32):
    pca_xp3 = PCA(n_components=i)
    pcarun_xp3 = pca_xp3.fit_transform(xp3)
    fit_xp3 = pca_xp3.fit(xp3)
    X3["cluster3"].unique()
    norm_xp3["cluster3"] = X3["cluster3"].values
    var_list3 = list(pca_xp3.explained_variance_ratio_)
    exp_var3 = sum(var_list3)
    if exp_var3 >= 0.9:
        print('Explained variation per principal component:', var_list3)
        print('')
        print('Total variation explained with', i, 'components for Home Health clusters: ', exp_var3)
        break
    i+=1
```

```
Explained variation per principal component: [0.26813690779281374, 0.15745793050420742, 0.09502695352414708, 0.05587999905508
462, 0.04939919344975779, 0.042570626648277106, 0.03773829144685198, 0.03279413052487152 6, 0.02971278248529019, 0.02666367935
979024, 0.02586957419504 8883, 0.02025953224138717, 0.01773387195966031, 0.01643865923708606, 0.015039732368300765, 0.01323068
8702090073]

Total variation explained with 16 components for Home Health clusters:  0.9039525534946651
```

Figure 79: Looped PCA analysis of scaled Home Health facility features showing a minimum of 16 components are required to account for 90% of explained variation.

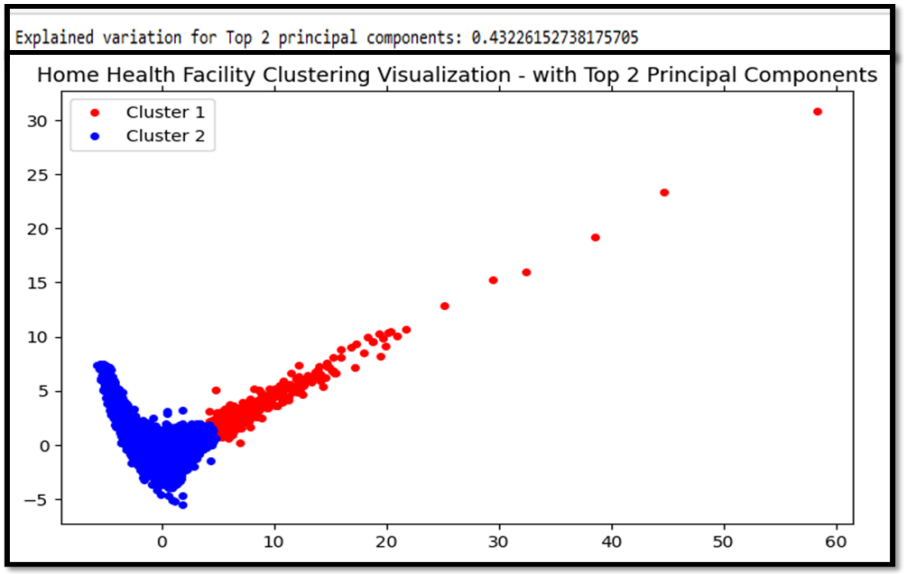

Figure 80: Scatter plot to visualize Home Health clusters using the top 2 principal components that account for the maximum amount of explained variation with printed output above the graph

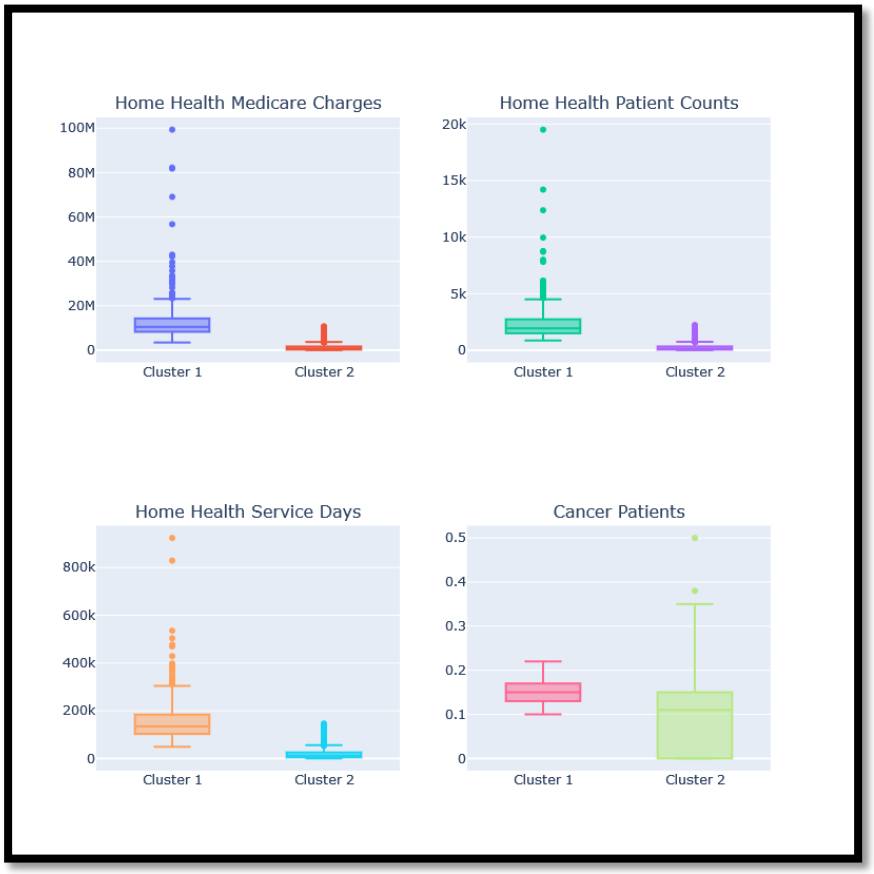

Figure 81: Grouped Boxplot to visualize distribution of datapoints for certain features in the individual Home Health clusters

The clustering and PCA analysis for Home Health agencies showed that variations in the features was sufficient to segregate these facilities into 2 clusters with some overlap, that showed differences in medicare charge, number of service days and patient counts.

RESULTS FOR INPATIENT REHAB FACILITIES

	PRVDR_ID	PRVDR_NAME	PRVDR_CITY	STATE	BENE_DSTNCT_CNT	TOT_EPSD_STAY_CNT	TOT_SRVC_DAYS	TOT_CHRG_AMT	TOT_ALOWD_AMT
0	20001	PROVIDENCE ALASKA MEDICAL CENTER	ANCHORAGE	AK	40	40	666	5007784	1278230
1	20017	ALASKA REGIONAL HOSPITAL	ANCHORAGE	AK	98	100	1425	11111476	3609570
2	10011	ST VINCENT'S EAST	BIRMINGHAM	AL	66	69	885	3705851	1268452
3	10029	EAST ALABAMA MEDICAL CENTER	OPELIKA	AL	157	168	2239	5437675	3661444
4	10033	UNIVERSITY OF ALABAMA HOSPITAL	BIRMINGHAM	AL	201	219	3173	16405267	5743019

5 rows × 35 columns

Figure 82: Data frame of 35 features for IRF facilities for cluster analysis

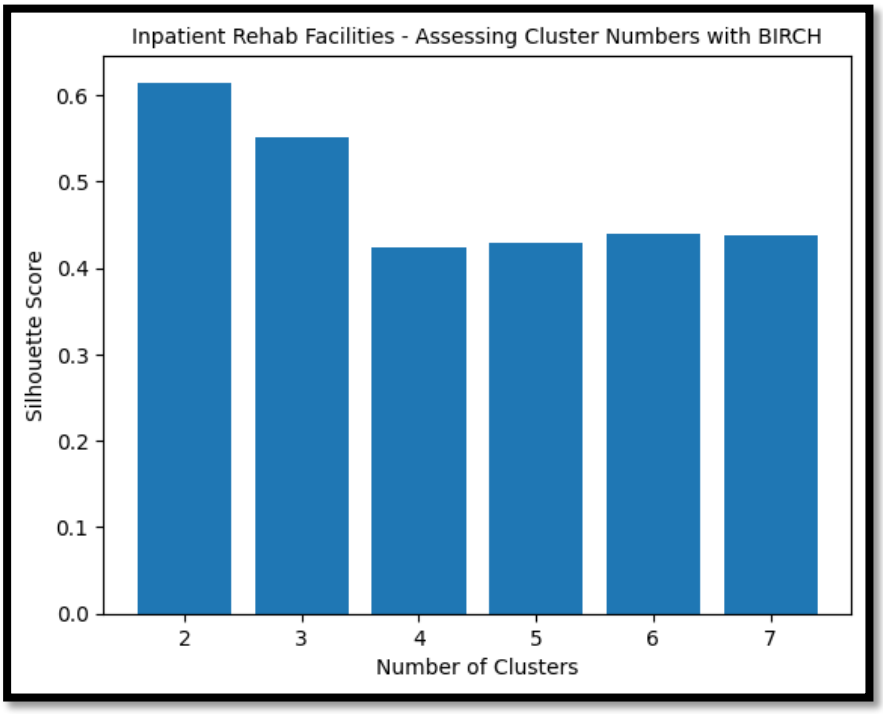

Figure 83: Visualizing clustering loop results to determine best number of clusters to segregate IRF facilities. N=2 gave the best results.

```
from sklearn.cluster import Birch, MiniBatchKMeans
birch4 = Birch(branching_factor=50, n_clusters=2, threshold=1.5)

X4['cluster4'] = birch4.fit_predict(X4)

birch4.fit(X4)

silhouette_score_average4 = silhouette_score(X4, birch4.predict(X4))

print(silhouette_score_average4)
```

```
0.6145210346110722
```

```
#To print number of datapoints in each IRF Cluster

irf1 = (X4[X4['cluster4'] == 0])
irf2 = (X4[X4['cluster4'] == 1])

print('Size of IRF Clusters:', 'Cluster 1:', len(irf1), ', Cluster 2:', len(irf2))
```

```
Size of IRF Clusters: Cluster 1: 191 , Cluster 2: 907
```

Figure 84: Clustering redone with n=2 clusters for IRF data with printed output of silhouette scores number of data points in each cluster after analysis.

```
# Determining Minimum number of components required to account for 90% explained variation

from sklearn.decomposition import PCA

for i in range(2,32):
    pca_xp4 = PCA(n_components=i)
    pcarun_xp4 = pca_xp4.fit_transform(xp4)
    fit_xp4 = pca_xp4.fit(xp4)
    X4["cluster4"].unique()
    norm_xp4["cluster4"] = X4["cluster4"].values
    var_list4 = list(pca_xp4.explained_variance_ratio_)
    exp_var4 = sum(var_list4)
    if exp_var4 >= 0.9:
        print('Explained variation per principal component:', var_list4)
        print('')
        print('Total variation explained with', i, 'components for IRF Clusters: ', exp_var4)
        break
    i+=1
```

```
Explained variation per principal component: [0.2771490137167606, 0.1430629186282524, 0.10589192754120497, 0.0649500747648972
7, 0.04550841445501567, 0.03708142469460432, 0.03352443178317382, 0.03142781799187788, 0.030533359480618128, 0.02371337353532
5263, 0.022065775597271818, 0.020042040831238127, 0.017490094387662275, 0.017183948606365622, 0.015629312222710686, 0.0147128
75829943592, 0.013019970506114927]

Total variation explained with 17 components for IRF Clusters:  0.9129867745730372
```

Figure 85: Looped PCA analysis of scaled IRF facility features showing a minimum of 17 components are required to account for 90% of explained variation.

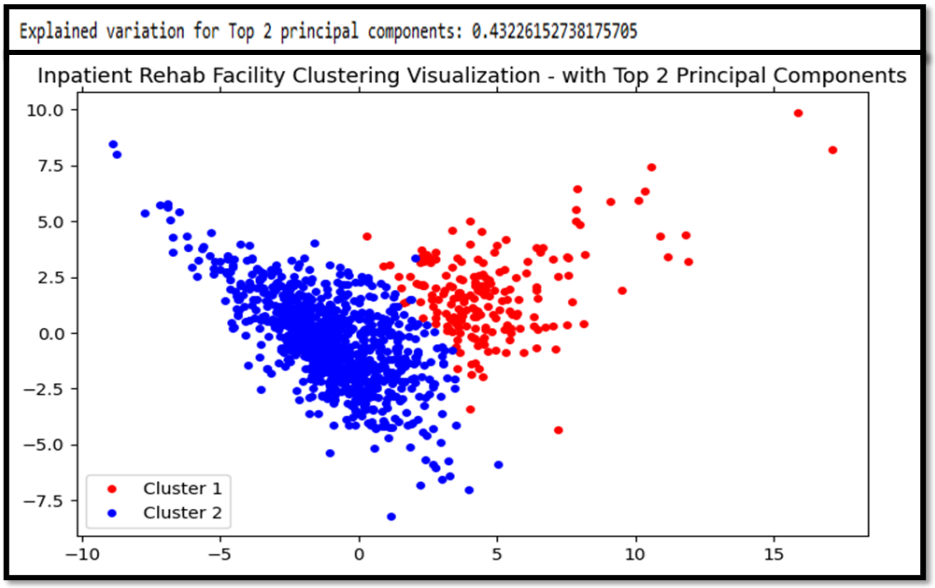

Figure 86: Scatter plot to visualize IRF clusters using the top 2 principal components that account for the maximum amount of explained variation with printed output above the graph

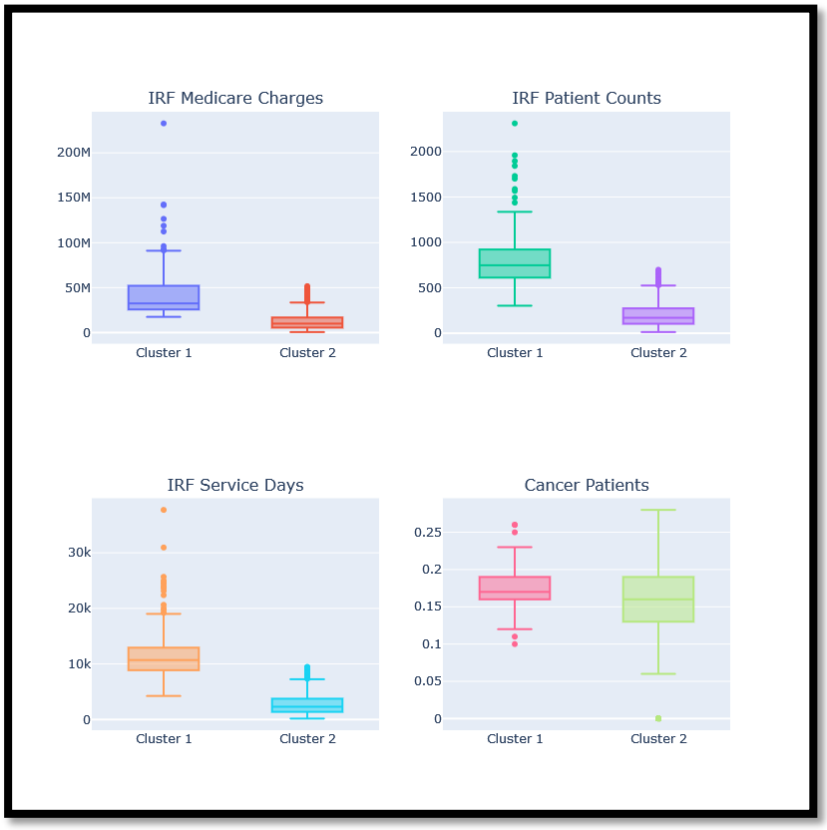

Figure 87: Grouped Boxplot to visualize distribution of datapoints for certain features in the individual IRF clusters

The clustering and PCA analysis for IRF facilities showed that variations in the features were sufficient to segregate these facilities into 2 clusters with some overlap, that showed differences in medicare charge, number of service days and patient counts.

RESULTS FOR LONG TERM CARE FACILITIES

	PRVDR_ID	PRVDR_NAME	PRVDR_CITY	STATE	BENE_DSTNCT_CNT	TOT_EPSD_STAY_CNT	TOT_SRVC_DAYS	TOT_CHRG_AMT	TOT_ALOWD_AMT
0	22001	ST ELIAS SPECIALTY HOSPITAL	ANCHORAGE	AK	91	96	4896	27069125	8941912
1	12006	INFIRMARY LTAC HOSPITAL	MOBILE	AL	142	146	3584	12530069	5488005
2	12007	NOLAND HOSPITAL MONTGOMERY II, LLC	MONTGOMERY	AL	92	96	2484	9743752	4022735
3	12008	SELECT SPECIALTY HOSPITAL - BIRMINGHAM	BIRMINGHAM	AL	154	157	4546	41612575	7526436
4	12009	NOLAND HOSPITAL BIRMINGHAM II, LLC	BIRMINGHAM	AL	165	171	4034	13730113	6142023

5 rows × 35 columns

Figure 88: Data frame of 35 features for LTC facilities for cluster analysis

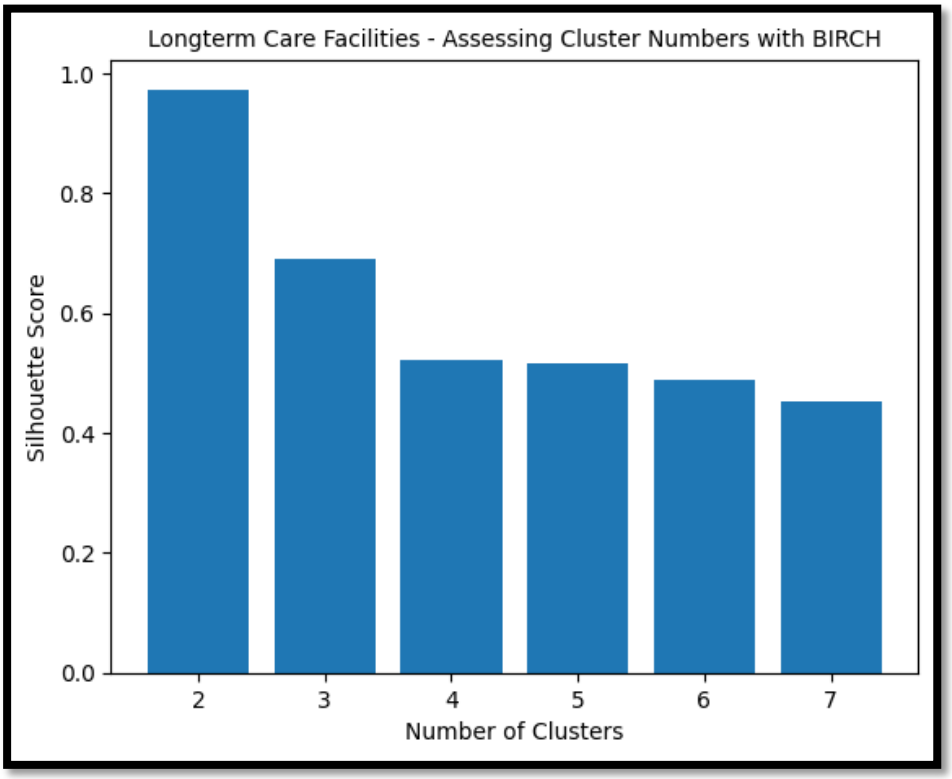

Figure 89: Visualizing clustering loop results to determine best number of clusters to segregate LTC facilities. N=2 gave the best results.

```
from sklearn.cluster import Birch, MiniBatchKMeans
birch5 = Birch(branching_factor=50, n_clusters=2, threshold=1.5)

X5['cluster5'] = birch5.fit_predict(X5)

birch5.fit(X5)

silhouette_score_average5 = silhouette_score(X5, birch5.predict(X5))

print(silhouette_score_average5)
```

```
0.9728511899645085
```

```
#To print number of datapoints in each LTC Cluster

ltc1 = (X5[X5['cluster5'] == 0])
ltc2 = (X5[X5['cluster5'] == 1])

print('Size of LTC Clusters:', 'Cluster 1:', len(ltc1), ', Cluster 2:', len(ltc2))
```

```
Size of LTC Clusters: Cluster 1: 351 , Cluster 2: 1
```

Figure 90: Clustering redone with n=2 clusters for LTC data with printed output of silhouette scores number of data points in each cluster after analysis.

```
# Determining Minimum number of components required to account for 90% explained variation

from sklearn.decomposition import PCA

for i in range(2,32):
    pca_xp5 = PCA(n_components=i)
    pcarun_xp5 = pca_xp5.fit_transform(xp5)
    fit_xp5 = pca_xp5.fit(xp5)
    X5["cluster5"].unique()
    norm_xp5["cluster5"] = X5["cluster5"].values
    var_list5 = list(pca_xp5.explained_variance_ratio_)
    exp_var5 = sum(var_list5)
    if exp_var5 >= 0.9:
        print('Explained variation per principal component:', var_list5)
        print('')
        print('Total variation explained with', i, 'components for LTC Clusters: ', exp_var5)
        break
    i+=1
```

```
Explained variation per principal component: [0.29708384994574755, 0.14966452488174373, 0.08332540302187992, 0.06013744112151
598, 0.04736650254085654, 0.04135498260832566, 0.03662405442054808, 0.035288356002439576, 0.02909143744786021, 0.028487919272
435765, 0.023814876537603526, 0.022680479927394068, 0.020012446740098246, 0.01827209499865776, 0.01505134820000009]

Total variation explained with 15 components for LTC Clusters:  0.9082557176671069
```

Figure 91: Looped PCA analysis of scaled LTC facility features showing a minimum of 15 components are required to account for 90% of explained variation.

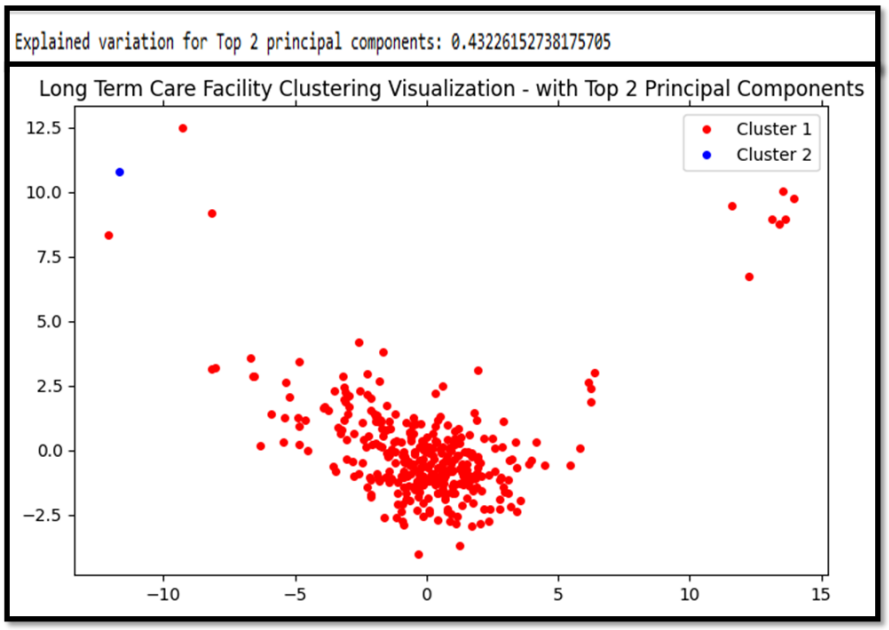

Figure 92: Scatter plot to visualize LTC clusters using the top 2 principal components that account for the maximum amount of explained variation with printed output above the graph

The clustering and PCA analysis for LTC facilities showed that even with significant variation in the top 2 principal components (30% and 14%), hospice facilities couldn't be segregated into more than a single cluster. This is bolstered by the fact that the 2nd cluster only included 1 provider.

6 Summary of Findings

6.1 **Research Question One:** Has there been a change in the inpatient and outpatient procedures patients access pre-pandemic (2018-2019) compared to during the pandemic (2020) which is discernable at various geographic levels (City, State)

Findings: From section 5.1.3 we can see that there was a discernible increase in inpatient hospitalizations in 2020 as compared to previous years, with a rise in patient counts seen across various several states. While outpatient services showed an opposing trend with patient numbers decrease in 2020, this drop wasn't nearly as steep as the increase in inpatient cases. Both trends can be attributed to the COVID 19 pandemic where inpatient hospitalizations for COVID related treatment and side effects increased while non-essential outpatient services that were considered less urgent were availed only in cases of emergency.

This proves our alternate hypothesis.

6.2 **Research Question Two:** What is the difference in cost for the most frequently used procedures across cities within a State or between neighboring states?

Findings: From Section 5.1.3 it is evident that there is a major difference in the cost of the same inpatient services being offered by different providers at the national and state levels. While a similar trend was once again seen with outpatient procedures, the cost difference was not nearly as pronounced since these procedures are generally not as expensive as inpatient procedures.

This proves our alternate hypothesis.

6.3 **Research Question Three:** Is there a disparity in the availability of inpatient and outpatient procedures across various geographic levels?

Findings: From Sections 5.1.1 and 5.1.2 it is evident that despite the states having the highest population counts also having the highest number of facilities in many cases, when this data is normalized with population data, it is clear that people living in lesser populated states have better access to facilities as compared to their counterparts.
This proves our alternate hypothesis.

6.4 **Research Question Four:** Can we use access to care parameters such as differences in gender, medicare costs, duration of services provided and patient episodes to segregate PAC and Hospice facilities into distinct clusters?

Findings: We answered this question with data from sections 5.1.4 and 5.1.5 and the clustering analysis.
Based on the trending analysis in sections 5.1.4 and 5.1.5, it is evident that PAC services are most frequently availed by 2 main demographics: Female and White patients.
Clustering analysis showed us that we could successfully only segregate 3 out of 5 types of PAC facilities into 2 clusters with distinct overlaps and some degree of overlap. These were the SNFs, Home Health and IRF facilities.
Clustering analysis yielded a silhouette score of between 0.69 to 0.83 for n=2 clusters, with an average of 15-17 features required to account for a variation of more than 90%.
In the case of the other two, even with a significant amount of variation in more than one principal component, the facilities couldn't successfully be clustered. Clustering analysis yielded a silhouette score

of between 0.97-0.99 for n=2 clusters, with an average of 15-17 features required to account for a variation of more than 90%.
While this trend goes against the alternate hypothesis, it is not an entirely surprising result since Hospice and Long terms care facilities cater to patients who are at the end of their life or are unable to live independently. As these limits the variety of services they need to provide their patients, it can be expected that these facilities would be standardized across the US and wouldn't vary significantly enough to

Therefore, overall, we can conclude that the alternate hypothesis was true in this case since 3 out 5 types had distinct enough features that proves that there is indeed a difference in the services available to patients that can be attributed to geography, demography and facility characteristics.

7 Limitations

While analyzing the disparity in access to care at geographic levels, while we have normalized the numbers with population data, a deeper analysis into the raw data of the number of facilities per region shows that while the numbers support our conclusion, this difference might be emphasized since the total area coverage of a region has not been considered. This was beyond the scope of our current abilities but would provide a more meaningful context for our results.

Additionally, in our clustering analysis, while we performed PCA analysis to visualize our clustering results, we were unable to take this a step further and determine exactly which principal components gave the most variations for all datasets. While we have performed distribution analysis, to answer this question, this type of analysis is

not ideal when working with unsupervised learning. We were also able to use only a single metric of the silhouette score to analyze the accuracy of our clustering method since our data didn't have any actual labels. These parameters would've added more depth to our analysis.

A. Appendix A – SQL Queries

Section I – SQL Queries for creating Tables for Data Analysis

```
CREATE TABLE IHP_TOPDISCH_DISEASE_18 AS
(SELECT A.DRG_CD, A.DRG_DESC, SUM(A.TOT_DSCHRGS) AS TOT_DSCH_18
FROM Inpatient_2018_PrvSrv A
WHERE A.DRG_CD IN (871,291,470,177,193,872,189,378,392,690)
GROUP BY A.DRG_CD, A.DRG_DESC
ORDER BY SUM(A.TOT_DSCHRGS) DESC);


CREATE TABLE IHP_TOPDISCH_DISEASE_19 AS
(SELECT B.DRG_CD, B.DRG_DESC, SUM(B.TOT_DSCHRGS) AS TOT_DSCH_19
FROM Inpatient_2019_PrvSrv B
WHERE B.DRG_CD IN (871,291,470,177,193,872,189,378,392,690)
GROUP BY B.DRG_CD, B.DRG_DESC
ORDER BY SUM(B.TOT_DSCHRGS) DESC);


CREATE TABLE IHP_TOPDISCH_DISEASE_20 AS
(SELECT C.DRG_CD, C.DRG_DESC, SUM(C.TOT_DSCHRGS) AS TOT_DSCH_20
FROM Inpatient_2020_PrvSrv C
WHERE C.DRG_CD IN (871,291,470,177,193,872,189,378,392,690)
GROUP BY C.DRG_CD, C.DRG_DESC
ORDER BY SUM(C.TOT_DSCHRGS) DESC);

--------------------------------------------------------------------------------

CREATE TABLE IHP_TOPDISCH_STATE_18 AS
(SELECT A.Prv_State, SUM(A.TOT_DSCHRGS) AS TOT_DSCH_18
FROM Inpatient_2018_PrvSrv A
WHERE A.Prv_State IN ('FL','CA','TX','NY','IL','PA','OH','NC','MI','MA')
GROUP BY A.Prv_State
ORDER BY SUM(A.TOT_DSCHRGS) DESC);


CREATE TABLE IHP_TOPDISCH_STATE_19 AS
(SELECT B.Prv_State, SUM(B.TOT_DSCHRGS) AS TOT_DSCH_19
FROM Inpatient_2019_PrvSrv B
WHERE B.Prv_State IN ('FL','CA','TX','NY','IL','PA','OH','NC','MI','MA')
GROUP BY B.Prv_State
ORDER BY SUM(B.TOT_DSCHRGS) DESC);
```

```
CREATE TABLE IHP_TOPDISCH_STATE_20 AS
(SELECT C.Prv_State, SUM(C.TOT_DSCHRGS) AS TOT_DSCH_20
FROM Inpatient_2020_PrvSrv C
WHERE C.Prv_State IN ('FL','CA','TX','NY','IL','PA','OH','NC','MI','MA')
GROUP BY C.Prv_State
ORDER BY SUM(C.TOT_DSCHRGS) DESC);

-----------------------------------------------------------------

CREATE TABLE OPD_TOPDISCH_DISEASE_18 AS
SELECT A.APC_Cd, A.APC_Desc, SUM(A.Bene_Cnt) AS TOT_DSCH_18
FROM Outpatient_2018_PrvSrv A
WHERE A.APC_Cd IN (8011,5072,5191,5115,5491,5114,5302,5361,5193,5373)
GROUP BY A.APC_Cd, A.APC_Desc
ORDER BY SUM(A.Bene_Cnt) DESC

CREATE TABLE OPD_TOPDISCH_DISEASE_19 AS
SELECT B.APC_Cd, B.APC_Desc, SUM(B.Bene_Cnt) AS TOT_DSCH_19
FROM Outpatient_2019_PrvSrv B
WHERE B.APC_Cd IN (8011,5072,5191,5115,5491,5114,5302,5361,5193,5373)
GROUP BY B.APC_Cd, B.APC_Desc
ORDER BY SUM(B.Bene_Cnt) DESC

CREATE TABLE OPD_TOPDISCH_DISEASE_20 AS
SELECT C.APC_Cd, C.APC_Desc, SUM(C.Bene_Cnt) AS TOT_DSCH_20
FROM Outpatient_2020_PrvSrv C
WHERE C.APC_Cd IN (8011,5072,5191,5115,5491,5114,5302,5361,5193,5373)
GROUP BY C.APC_Cd, C.APC_Desc
ORDER BY SUM(C.Bene_Cnt) DESC


-----------------------------------------------------------------------

CREATE TABLE OPD_TOPDISCH_STATE_18 AS
SELECT A.Prv_State, SUM(A.Bene_Cnt) AS TOT_DSCH_18
FROM Outpatient_2018_PrvSrv A
WHERE A.Prv_State IN ('CA','TX','FL','NY','IL','OH','PA','NC','MI','VA')
GROUP BY A.Prv_State
ORDER BY SUM(A.Bene_Cnt) DESC

CREATE TABLE OPD_TOPDISCH_STATE_19 AS
SELECT B.Prv_State, SUM(B.Bene_Cnt) AS TOT_DSCH_19
FROM Outpatient_2019_PrvSrv B
```

```
WHERE B.Prv_State IN ('CA','TX','FL','NY','IL','OH','PA','NC','MI','VA')
GROUP BY B.Prv_State
ORDER BY SUM(B.Bene_Cnt) DESC

CREATE TABLE OPD_TOPDISCH_STATE_20 AS
SELECT C.Prv_State, SUM(C.Bene_Cnt) AS TOT_DSCH_20
FROM Outpatient_2020_PrvSrv C
WHERE C.Prv_State IN ('CA','TX','FL','NY','IL','OH','PA','NC','MI','VA')
GROUP BY C.Prv_State
ORDER BY SUM(C.Bene_Cnt) DESC
```

Section II – SQL Queries for Exploratory Data Analysis

SQL Queries for Inpatient Data Analysis before Table creation

```
SELECT TOP 10 A.DRG_CD, A.DRG_DESC,
SUM(A.TOT_DSCHRGS) AS TOT_DSCH_DIS_18
FROM Inpatient_2018_PrvSrv A
GROUP BY A.DRG_CD, A.DRG_DESC
ORDER BY SUM(A.TOT_DSCHRGS) DESC

SELECT TOP 10 A.Prv_State,
SUM(A.TOT_DSCHRGS) AS TOT_DSCH_ST_18
FROM Inpatient_2018_PrvSrv A
GROUP BY A.Prv_State
ORDER BY SUM(A.TOT_DSCHRGS) DESC

SELECT TOP 10 A.DRG_CD, A.DRG_DESC,
SUM(A.TOT_DSCHRGS) AS TOT_DSCH_DIS_19
FROM Inpatient_2019_PrvSrv A
GROUP BY A.DRG_CD, A.DRG_DESC
ORDER BY SUM(A.TOT_DSCHRGS) DESC

SELECT TOP 10 A.Prv_State,
SUM(A.TOT_DSCHRGS) AS TOT_DSCH_ST_19
FROM Inpatient_2019_PrvSrv A
GROUP BY A.Prv_State
ORDER BY SUM(A.TOT_DSCHRGS) DESC
```

```
SELECT TOP 10 A.DRG_CD, A.DRG_DESC,
SUM(A.TOT_DSCHRGS) AS TOT_DSCH_DIS_20
FROM Inpatient_2020_PrvSrv A
GROUP BY A.DRG_CD, A.DRG_DESC
ORDER BY SUM(A.TOT_DSCHRGS) DESC

SELECT TOP 10 A.Prv_State,
SUM(A.TOT_DSCHRGS) AS TOT_DSCH_ST_20
FROM Inpatient_2020_PrvSrv A
GROUP BY A.Prv_State
ORDER BY SUM(A.TOT_DSCHRGS) DESC
```

SQL Queries for Outpatient Data Analysis before Table creation

```
SELECT TOP 11 A.APC_Cd, A.APC_Desc,
SUM(A.Bene_Cnt) AS TOT_DSCH_DIS_18
FROM Outpatient_2018_PrvSrv A
GROUP BY A.APC_Cd, A.APC_Desc
ORDER BY SUM(A.Bene_Cnt) DESC

SELECT TOP 10 A.Prv_State,
SUM(A.Bene_Cnt) AS TOT_DSCH_ST_18
FROM Outpatient_2018_PrvSrv A
GROUP BY A.Prv_State
ORDER BY SUM(A.Bene_Cnt) DESC

SELECT TOP 11 A.APC_Cd, A.APC_Desc,
SUM(A.Bene_Cnt) AS TOT_DSCH_DIS_19
FROM Outpatient_2019_PrvSrv A
GROUP BY A.APC_Cd, A.APC_Desc
ORDER BY SUM(A.Bene_Cnt) DESC

SELECT TOP 10 A.Prv_State,
SUM(A.Bene_Cnt) AS TOT_DSCH_ST_19
FROM Outpatient_2019_PrvSrv A
GROUP BY A.Prv_State
ORDER BY SUM(A.Bene_Cnt) DESC

SELECT TOP 11 A.APC_Cd, A.APC_Desc,
SUM(A.Bene_Cnt) AS TOT_DSCH_DIS_20
FROM Outpatient_2020_PrvSrv A
GROUP BY A.APC_Cd, A.APC_Desc
```

```
ORDER BY SUM(A.Bene_Cnt) DESC

SELECT TOP 10 A.Prv_State,
SUM(A.Bene_Cnt) AS TOT_DSCH_ST_20
FROM Outpatient_2020_PrvSrv A
GROUP BY A.Prv_State
ORDER BY SUM(A.Bene_Cnt) DESC
```

Appendix B – Python scripts

Section I: Table creation and Data File upload Scripts

Outpatient Data by Geography and Provider - 2018 to 2020

In [59]:
```
make_table_2 = """CREATE TABLE Outpatient_2018_PrvSrv (
PRV_CCN BIGINT, PRV_Name VARCHAR(200), PRV_ST_ADR VARCHAR(200), PRV_City VARCHAR(100), PRV_STATE VARCHAR(50),
PRV_State_FIPS INT, PRV_Zip5 INT, PRV_RUCA FLOAT, PRV_RUCA_Desc VARCHAR(200), APC_Desc VARCHAR(300),
Bene_Cnt INT, CAPC_Srvcs INT, Avg_Tot_Sub_Chrgs FLOAT, Avg_Mdcr_Alowd_Amt FLOAT, Avg_Mdcr_Pymt_Amt FLOAT,
Outlier_Srvcs INT, Avg_Mdcr_Outlier_Amt FLOAT
)"""
cursor.execute(make_table_2)

# Used to create tables

# Outpatient_2018_PrvSrv
# Outpatient_2019_PrvSrv
# Outpatient_2020_PrvSrv
```

Out[59]:
```
0
```

In [60]:
```
load_table_2 = """LOAD DATA LOCAL INFILE 'MUP_OHP_R20_P04_V10_D18_Prov_Svc.csv'
INTO TABLE Outpatient_2018_PrvSrv
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'"""
cursor.execute(load_table_2)

# Used to load datasets

# MUP_OUT_RY22_P04_V10_DY20_Prov_Svc.csv
# MUP_OUT_RY21_P04_V10_DY19_Prov_Svc.csv
# MUP_OHP_R20_P04_V10_D18_Prov_Svc.csv
```

Out[60]:
```
110478
```

In [27]:
```
conn.commit()
```

Figure 93: Table Creation and Data file upload script for importing the outpatient data files into the group database using Python Jupyter Notebook environment.

Post_Acute_Care_and_Hospice_by_Geography_and_Provider - 2020 Data

In [ ]:
```
make_table_3 = """CREATE TABLE MD_PAC_and_Hospice_by_Geo_Prv_20 (
YEAR int , YEAR_TYPE varchar(5) , SMRY_CTGRY varchar(10) , SRVC_CTGRY varchar(5) , PRVDR_ID varchar(10) , PRVDR_NAME varchar(100) ,
PRVDR_CITY varchar(20) , STATE varchar(20) , PRVDR_ZIP varchar(20) , BENE_DSTNCT_CNT varchar(20) , TOT_EPSD_STAY_CNT varchar(20) ,
TOT_SRVC_DAYS varchar(20) , BENE_LE_7_SRVC_DAYS_PCT varchar(5) , BENE_LE_30_SRVC_DAYS_PCT varchar(5) ,
BENE_GE_60_SRVC_DAYS_PCT varchar(5) , BENE_GE_180_SRVC_DAYS_PCT varchar(5) , TOT_CHRG_AMT varchar(20) ,
TOT_ALOWD_AMT varchar(20) , TOT_MDCR_PYMT_AMT varchar(20) , TOT_MDCR_STDZD_PYMT_AMT varchar(20) , TOT_OUTLIER_PYMT_AMT varchar(20) ,
BENE_MA_PCT varchar(5) , BENE_DUAL_PCT varchar(5) , BENE_RRL_PCT varchar(5) , BENE_AVG_AGE int , BENE_MALE_PCT varchar(5) ,
BENE_FEML_PCT varchar(5) , BENE_RACE_WHT_PCT varchar(5) , BENE_RACE_BLACK_PCT varchar(5) , BENE_RACE_API_PCT varchar(5) ,
BENE_RACE_HSPNC_PCT varchar(5) , BENE_RACE_NATIND_PCT varchar(5) , BENE_RACE_OTHR_PCT varchar(5) , BENE_AVG_RISK_SCRE float ,
BENE_AVG_CC_CNT int , BENE_CC_AF_PCT varchar(5) , BENE_CC_ALZHMR_PCT varchar(5) , BENE_CC_ASTHMA_PCT varchar(5) ,
BENE_CC_CNCR_PCT varchar(5) , BENE_CC_CHF_PCT varchar(5) , BENE_CC_CKD_PCT varchar(5) , BENE_CC_COPD_PCT varchar(5) ,
BENE_CC_DPRSSN_PCT varchar(5) , BENE_CC_DBTS_PCT varchar(5) , BENE_CC_HYPLPDMA_PCT varchar(5) , BENE_CC_HYPRTNSN_PCT varchar(5) ,
BENE_CC_IHD_PCT varchar(5) , BENE_CC_OPO_PCT varchar(5) , BENE_CC_RAOA_PCT varchar(5) , BENE_CC_SZ_PCT varchar(5) ,
BENE_CC_STROK_PCT varchar(5) , BENE_PRMRY_DX_CNCR_PCT varchar(5) , BENE_PRMRY_DX_COPD_PCT varchar(5) ,
BENE_PRMRY_DX_RSPRTRYFAILR_PCT varchar(5) , BENE_PRMRY_DX_DMNT_PCT varchar(5) , BENE_PRMRY_DX_STROK_PCT varchar(5) ,
BENE_PRMRY_DX_CHF_PCT varchar(5) , BENE_PRMRY_DX_HYPRTNSN_PCT varchar(5) , BENE_PRMRY_DX_OTHRCRDVSCLR_PCT varchar(5) ,
BENE_PRMRY_DX_INFCTN_PCT varchar(5) , BENE_PRMRY_DX_ORTHO_PCT varchar(5) , BENE_PRMRY_DX_INJURY_PCT varchar(5) ,
BENE_PRMRY_DX_MTR_NRL_PCT varchar(5) , BENE_PRMRY_DX_DBTS_PCT varchar(5) , BENE_PRMRY_DX_SKN_PCT varchar(5) ,
BENE_PRMRY_DX_AFTRCRE_PCT varchar(5) , NRSNG_VISITS_CNT varchar(13) , PT_VISITS_CNT varchar(13) ,
OT_VISITS_CNT varchar(13) , SLP_VISITS_CNT varchar(13) , MSW_VISITS_CNT varchar(13) , AIDE_VISITS_CNT varchar(13) ,
PHYSN_VISITS_CNT varchar(10) , TOT_NRSNG_MNTS varchar(20) , TOT_PT_MNTS varchar(20) , INDVDL_PT_MNTS varchar(20) ,
CNCRNT_GRP_PT_MNTS varchar(10) , COTRT_PT_MNTS varchar(10) , TOT_OT_MNTS varchar(15) , INDVDL_OT_MNTS varchar(15) ,
CNCRNT_GRP_OT_MNTS varchar(15) , COTRT_OT_MNTS varchar(15) , TOT_SLP_MNTS varchar(15) , INDVDL_SLP_MNTS varchar(15) ,
CNCRNT_GRP_SLP_MNTS varchar(15) , COTRT_SLP_MNTS varchar(15) , TOT_PT_MNTS_STAY varchar(15) , INDVDL_PT_MNTS_STAY varchar(15) ,
CNCRNT_GRP_PT_MNTS_STAY varchar(15) , TOT_OT_MNTS_STAY varchar(15) , INDVDL_OT_MNTS_STAY varchar(15) , CNCRNT_GRP_OT_MNTS_STAY varch
TOT_SLP_MNTS_STAY varchar(15) , INDVDL_SLP_MNTS_STAY varchar(15) , CNCRNT_GRP_SLP_MNTS_STAY varchar(15) , TOT_MSW_MNTS varchar(15) ,
TOT_AIDE_MNTS varchar(15) , NRSNG_MNTS_AVG_7_DAY_PRIOR_DEATH varchar(5) , MSW_MNTS_AVG_7_DAY_PRIOR_DEATH varchar(5) ,
AIDE_MNTS_AVG_7_DAY_PRIOR_DEATH varchar(5) , SRVC_SITE_DAYS_HOME_PCT varchar(5) , SRVC_SITE_DAYS_ASTD_LVG_FAC_PCT varchar(5) ,
SRVC_SITE_DAYS_LTCF_PCT varchar(5) , SRVC_SITE_DAYS_SNF_PCT varchar(5) , SRVC_SITE_DAYS_IP_PCT varchar(5) ,
SRVC_SITE_DAYS_IP_HOSPC_PCT varchar(5) , DSCHRG_CMNTY_SLFCR_PCT varchar(5) , DSCHRG_IP_PCT varchar(5) , DSCHRG_HH_PCT varchar(5) ,
DSCHRG_SNF_PCT varchar(5) , DSCHRG_IRF_PCT varchar(5) , DSCHRG_HOSPC_PCT varchar(5) , DSCHRG_DEATH_PCT varchar(5) ,
DSCHRG_UNK_PCT varchar(5) , DSCHRG_HOSPC_LV_PCT varchar(5) ,
HOSPC_RHC_DAYS_PCT varchar(5) , TOT_HH_LUPA_EPSDS_CNT varchar(5) , PDGM_CMNTY_ERLY_EPSD_PCT varchar(5) ,
PDGM_INST_ERLY_EPSD_PCT varchar(5) , PDGM_CMNTY_LTE_EPSD_PCT varchar(5) , PDGM_INST_LTE_EPSD_PCT varchar(5) ,
PDGM_NO_EPSD_PCT varchar(5) , PDGM_LOW_EPSD_PCT varchar(5) , PDGM_HIGH_EPSD_PCT varchar(5) , PDPM_PT_OT_TC_DAYS_PCT varchar(5) ,
PDPM_PT_OT_TD_DAYS_PCT varchar(5) , PDPM_SLP_SK_DAYS_PCT varchar(5) , PDPM_SLP_SL_DAYS_PCT varchar(5) ,
PDPM_NRSNG_ES3_DAYS_PCT varchar(5) , PDPM_NRSNG_ES2_DAYS_PCT varchar(5) , PDPM_NTA_NA_DAYS_PCT varchar(5) ,
PDPM_NTA_NB_DAYS_PCT varchar(5) , CMG_STROK_DAYS_PCT varchar(5) , CMG_INJURY_TRMA_DAYS_PCT varchar(5) ,
CMG_ORTHO_DAYS_PCT varchar(5) , CMG_ARTHRTS_DAYS_PCT varchar(5) , CMG_CRDVSCLR_PLMNRY_DAYS_PCT varchar(5) ,
CMG_NRLGCL_DAYS_PCT varchar(5) , CMG_OTHR_DAYS_PCT varchar(5)
)"""
cursor.execute(make_table_3)
```

In [5]:
```
load_table_3 = """LOAD DATA LOCAL INFILE 'Medicare_Post_Acute_Care_and_Hospice_by_Geography_and_Provider_2020.csv'
INTO TABLE MD_PAC_and_Hospice_by_Geo_Prv_20
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'"""
cursor.execute(load_table_3)
```

Out[5]: 29637

In [61]:
```
conn.commit()
```

Figure 94: Table Creation and Data file upload script for importing the Post-Acute care and Hospice data master file into the group database using Python Jupyter Notebook environment.

Skilled Nursing Facilitty_by_Geography_and_Provider - 2020 Data (Subset of Post_Acute_Care_and_Hospice_by_Geography_and_Provider)

In [11]:
```
make_table_4 = """CREATE TABLE MD_PAC_SNF_2020 (
YEAR smallint ,YEAR_TYPE nvarchar(50) ,SMRY_CTGRY nvarchar(50) ,SRVC_CTGRY nvarchar(50) ,PRVDR_ID nvarchar(50),
PRVDR_NAME nvarchar(100) ,PRVDR_CITY nvarchar(50) ,STATE nvarchar(50) ,PRVDR_ZIP nvarchar(50) ,BENE_DSTNCT_CNT int ,
TOT_EPSD_STAY_CNT int ,TOT_SRVC_DAYS int ,BENE_LE_7_SRVC_DAYS_PCT float ,BENE_LE_30_SRVC_DAYS_PCT float ,
BENE_GE_60_SRVC_DAYS_PCT float ,TOT_CHRG_AMT bigint ,TOT_ALOWD_AMT bigint ,TOT_MDCR_PYMT_AMT bigint ,TOT_MDCR_STDZD_PYMT_AMT bigint
BENE_MA_PCT float ,BENE_DUAL_PCT float ,BENE_RRL_PCT float ,BENE_AVG_AGE int ,BENE_MALE_PCT float ,BENE_FEML_PCT float ,BENE_RACE_WH
BENE_RACE_BLACK_PCT float ,BENE_RACE_API_PCT float ,BENE_RACE_HSPNC_PCT float ,BENE_RACE_NATIND_PCT float ,BENE_RACE_OTHR_PCT float
BENE_AVG_RISK_SCRE float ,BENE_AVG_CC_CNT int ,BENE_CC_AF_PCT float ,BENE_CC_ALZHMR_PCT float ,BENE_CC_ASTHMA_PCT float ,
BENE_CC_CNCR_PCT float ,BENE_CC_CHF_PCT float ,BENE_CC_CKD_PCT float ,BENE_CC_COPD_PCT float ,BENE_CC_DPRSSN_PCT float ,
BENE_CC_DBTS_PCT float ,BENE_CC_HYPLPDMA_PCT float ,BENE_CC_HYPRTNSN_PCT float ,BENE_CC_IHD_PCT float ,BENE_CC_OPO_PCT float ,
BENE_CC_RAOA_PCT float ,BENE_CC_SZ_PCT float ,BENE_CC_STROK_PCT float ,BENE_PRMRY_DX_CNCR_PCT float ,BENE_PRMRY_DX_COPD_PCT float ,
BENE_PRMRY_DX_RSPRTRYFAILR_PCT float ,BENE_PRMRY_DX_DMNT_PCT float ,BENE_PRMRY_DX_STROK_PCT float ,BENE_PRMRY_DX_CHF_PCT float ,
BENE_PRMRY_DX_HYPRTNSN_PCT float ,BENE_PRMRY_DX_OTHRCRDVSCLR_PCT float ,BENE_PRMRY_DX_INFCTN_PCT float ,BENE_PRMRY_DX_ORTHO_PCT floa
BENE_PRMRY_DX_INJURY_PCT float ,BENE_PRMRY_DX_MTR_NRL_PCT float ,BENE_PRMRY_DX_DBTS_PCT float ,BENE_PRMRY_DX_SKN_PCT float ,
BENE_PRMRY_DX_AFTRCRE_PCT float ,TOT_PT_MNTS_STAY int ,INDVDL_PT_MNTS_STAY int ,CNCRNT_GRP_PT_MNTS_STAY int ,TOT_OT_MNTS_STAY int ,
INDVDL_OT_MNTS_STAY int ,CNCRNT_GRP_OT_MNTS_STAY int ,TOT_SLP_MNTS_STAY int ,INDVDL_SLP_MNTS_STAY int ,CNCRNT_GRP_SLP_MNTS_STAY int
DSCHRG_CMNTY_SLFCR_PCT float ,DSCHRG_IP_PCT float ,DSCHRG_HH_PCT float ,DSCHRG_HOSPC_PCT float ,DSCHRG_DEATH_PCT float ,
DSCHRG_UNK_PCT float ,PDPM_PT_OT_TC_DAYS_PCT float ,PDPM_PT_OT_TD_DAYS_PCT float ,PDPM_SLP_SK_DAYS_PCT float ,PDPM_SLP_SL_DAYS_PCT f
PDPM_NRSNG_ES3_DAYS_PCT float ,PDPM_NRSNG_ES2_DAYS_PCT float ,PDPM_NTA_NA_DAYS_PCT float ,PDPM_NTA_NB_DAYS_PCT float
)"""
cursor.execute(make_table_4)
```

Out[11]:
```
0
```

In [12]:
```
load_table_4 = """LOAD DATA LOCAL INFILE 'PACSNF_2020v2.csv'
INTO TABLE MD_PAC_SNF_2020
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'"""
cursor.execute(load_table_4)
```

Out[12]:
```
14641
```

In [34]:
```
conn.commit()
```

Figure 95: Table Creation and Data file upload script for importing the Skilled Nursing Facility sub-dataset file into the group database using Python Jupyter Notebook environment.

Hospice Faciltity_by_Geography_and_Provider - 2020 Data (Subset of Post_Acute_Care_and_Hospice_by_Geography_and_Provider)

In [8]:
```
make_table_5 = """CREATE TABLE MD_PAC_HOS_2020 (
YEAR int ,YEAR_TYPE nvarchar(50) ,SMRY_CTGRY nvarchar(50) ,SRVC_CTGRY nvarchar(50) ,PRVDR_ID nvarchar(50) ,PRVDR_NAME nvarchar(50) ,
PRVDR_CITY nvarchar(50) ,STATE nvarchar(50) ,PRVDR_ZIP nvarchar(50) ,BENE_DSTNCT_CNT int ,TOT_EPSD_STAY_CNT int ,TOT_SRVC_DAYS int ,
BENE_LE_7_SRVC_DAYS_PCT float ,BENE_LE_30_SRVC_DAYS_PCT float ,BENE_GE_60_SRVC_DAYS_PCT float ,BENE_GE_180_SRVC_DAYS_PCT float ,
TOT_CHRG_AMT bigint ,TOT_ALOWD_AMT bigint ,TOT_MDCR_PYMT_AMT bigint ,TOT_MDCR_STDZD_PYMT_AMT bigint ,BENE_MA_PCT float ,
BENE_DUAL_PCT float ,BENE_RRL_PCT float ,BENE_AVG_AGE int ,BENE_MALE_PCT float ,BENE_FEML_PCT float ,BENE_RACE_WHT_PCT float ,
BENE_RACE_BLACK_PCT float ,BENE_RACE_API_PCT float ,BENE_RACE_HSPNC_PCT float ,BENE_RACE_NATIND_PCT float ,BENE_RACE_OTHR_PCT float
BENE_AVG_RISK_SCRE float ,BENE_PRMRY_DX_CNCR_PCT float ,BENE_PRMRY_DX_COPD_PCT float ,BENE_PRMRY_DX_RSPRTRYFAILR_PCT float ,
BENE_PRMRY_DX_DMNT_PCT float ,BENE_PRMRY_DX_STROK_PCT float ,BENE_PRMRY_DX_CHF_PCT float ,BENE_PRMRY_DX_HYPRTNSN_PCT float ,
BENE_PRMRY_DX_OTHRCRDVSCLR_PCT float ,BENE_PRMRY_DX_INFCTN_PCT float ,BENE_PRMRY_DX_ORTHO_PCT float ,BENE_PRMRY_DX_INJURY_PCT float
BENE_PRMRY_DX_MTR_NRL_PCT float ,BENE_PRMRY_DX_DBTS_PCT float ,NRSNG_VISITS_CNT int ,MSW_VISITS_CNT int ,AIDE_VISITS_CNT int ,
PHYSN_VISITS_CNT int ,TOT_NRSNG_MNTS int ,TOT_MSW_MNTS int ,TOT_AIDE_MNTS int ,NRSNG_MNTS_AVG_7_DAY_PRIOR_DEATH int ,
MSW_MNTS_AVG_7_DAY_PRIOR_DEATH int ,AIDE_MNTS_AVG_7_DAY_PRIOR_DEATH int ,SRVC_SITE_DAYS_HOME_PCT float ,SRVC_SITE_DAYS_ASTD_LVG_FAC_
SRVC_SITE_DAYS_LTCF_PCT float ,SRVC_SITE_DAYS_SNF_PCT float ,SRVC_SITE_DAYS_IP_PCT float ,SRVC_SITE_DAYS_IP_HOSPC_PCT float ,
DSCHRG_DEATH_PCT float ,DSCHRG_HOSPC_LV_PCT float ,HOSPC_RHC_DAYS_PCT float)"""
cursor.execute(make_table_5)
```

Out[8]:
```
0
```

In [9]:
```
load_table_5 = """LOAD DATA LOCAL INFILE 'PACHOS_2020v2.csv'
INTO TABLE MD_PAC_HOS_2020
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'"""
cursor.execute(load_table_5)
```

Out[9]:
```
4771
```

In [45]:
```
conn.commit()
```

Figure 96: Table Creation and Data file upload script for importing the Hospice facility sub-dataset file into the group database using Python Jupyter Notebook environment.

Section II: Code for Creating Choropleths to visualize number of facilities in different states across the US as Number of Facilities and with population normalization

```
#Outpatient Facilites by State

dfopd = pd.read_sql_query(" SELECT PRV_STATE AS STATE, COUNT(*) AS TOTAL_OPD FROM Outpatient_2020_PrvSrv
                          GROUP BY PRV_STATE ORDER BY TOTAL_OPD DESC" , conn)
dfopd  = pd.DataFrame(dfopd)
print(dfopd.head())

fig = go.Figure(data=go.Choropleth(
    locations=dfopd ['STATE'],
    z = dfopd ['TOTAL_OPD'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Greens',
    colorbar_title = "Number of Facilities",
))

fig.update_layout(
    title_text = 'Outpatient Facilities in Different States',
    geo_scope='usa',
)

fig.show()
```

Figure 97: Code for visualizing Number of Outpatient Facilities per State on a Choropleth

```
#SNF Facilites by State

dfsnf = pd.read_sql_query(" SELECT STATE, COUNT(*) AS TOTAL_SNF FROM MD_PAC_SNF_2020 where STATE != 'NATIONAL TOTAL'
                          GROUP BY STATE ORDER BY TOTAL_SNF DESC" , conn)
dfsnf = pd.DataFrame(dfsnf)
print(dfsnf.head())

fig = go.Figure(data=go.Choropleth(
    locations=dfsnf['STATE'], # Spatial coordinates
    z = dfsnf['TOTAL_SNF'].astype(float), # Data to be color-coded
    locationmode = 'USA-states', # set of locations match entries in `locations`
    colorscale = 'Reds',
    colorbar_title = "Number of Facilities",
))

fig.update_layout(
    title_text = 'Skilled Nursing Facilities in Different States',
    geo_scope='usa', # limite map scope to USA
)

fig.show()
```

Figure 98: Code for visualizing Number of Skilled Nursing Facilities per State on a Choropleth

```
#Hospice Facilites by State

dfhos = pd.read_sql_query("SELECT STATE, COUNT(*) AS TOTAL_HOSPICE FROM MD_PAC_HOS_2020 where STATE != 'NATIONAL TOTAL'
                          GROUP BY STATE ORDER BY TOTAL_HOSPICE DESC" , conn)
dfhos = pd.DataFrame(dfhos)
print(dfhos.head())

import plotly.graph_objects as go

import pandas as pd

fig = go.Figure(data=go.Choropleth(
    locations=dfhos['STATE'],
    z = dfhos['TOTAL_HOSPICE'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Greens',
    colorbar_title = "Number of Facilities",
))

fig.update_layout(
    title_text = 'Hospice Facilities in Different States',
    geo_scope='usa',
)

fig.show()
```

Figure 99: Code for visualizing Number of Hospice Facilities per State on a Choropleth

```
#Outpatient Facilites by State normalized to population

dfopd = pd.read_sql_query("SELECT A.PRV_STATE AS STATE, COUNT(A.PRV_STATE) AS TOTAL_OPD, B.Pop_2020 AS STATE_POP,
                          (COUNT(A.PRV_STATE)/B.Pop_2020)*1000000 AS OPD_RATIO FROM Outpatient_2020_PrvSrv A
                          JOIN usstatepop_2020 B ON A.PRV_STATE = B.State_Abbr GROUP BY PRV_STATE
                          ORDER BY OPD_RATIO DESC" , conn)
dfopd  = pd.DataFrame(dfopd)
print(dfopd.head())

fig = go.Figure(data=go.Choropleth(
    locations=dfopd ['STATE'],
    z = dfopd ['OPD_RATIO'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Greens',
    colorbar_title = "FACILITES to STATE POPULATION RATIO * (10^6)",
))

fig.update_layout(
    title_text = 'Outpatient Facilities in Different States (Normalized with State Population data)',
    geo_scope='usa',
)

fig.show()
```

Figure 100: Code for visualizing Number of Outpatient Facilities per State normalized with population data on a Choropleth

```
#SNF Facilites by State normalized to population

dfsnf = pd.read_sql_query("SELECT A.STATE, COUNT(A.STATE) AS TOTAL_SNF, B.Pop_2020 AS STATE_POP,
                          (COUNT(A.STATE)/B.Pop_2020)*1000000 AS SNF_RATIO FROM MD_PAC_SNF_2020 A
                          JOIN usstatepop_2020 B ON A.STATE = B.State_Abbr where A.STATE != 'NATIONAL TOTAL'
                          GROUP BY A.STATE ORDER BY SNF_RATIO DESC" , conn)
dfsnf = pd.DataFrame(dfsnf)
print(dfsnf.head())

fig = go.Figure(data=go.Choropleth(
    locations=dfsnf['STATE'],
    z = dfsnf['SNF_RATIO'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Reds',
    colorbar_title = "FACILITES to STATE POPULATION RATIO * (10^6)",
))

fig.update_layout(
    title_text = 'Skilled Nursing Facilities in Different States (Normalized with State Population data)',
    geo_scope='usa',
)

fig.show()
```

Figure 101: Code for visualizing Number of Skilled Nursing Facilities per State normalized with population data on a Choropleth

```
#Hospice Facilites by State normalized to population

dfhos = pd.read_sql_query("SELECT A.STATE, COUNT(A.STATE) AS TOTAL_HOSPICE, B.Pop_2020 AS STATE_POP,
                          (COUNT(A.STATE)/B.Pop_2020)*1000000 AS HSP_RATIO FROM MD_PAC_HOS_2020 A
                          JOIN usstatepop_2020 B ON A.STATE = B.State_Abbr where A.STATE != 'NATIONAL TOTAL'
                          GROUP BY A.STATE ORDER BY HSP_RATIO DESC" , conn)
dfhos = pd.DataFrame(dfhos)
print(dfhos.head())

import plotly.graph_objects as go

import pandas as pd

fig = go.Figure(data=go.Choropleth(
    locations=dfhos['STATE'],
    z = dfhos['HSP_RATIO'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Greens',
    colorbar_title = "FACILITES to STATE POPULATION RATIO * (10^6)",
))

fig.update_layout(
    title_text = 'Hospice Facilities in Different States (Normalized with State Population data)',
    geo_scope='usa',
)

fig.show()
```

Figure 102: Code for visualizing Number of Hospice Facilities per State normalized with population data on a Choropleth

```
# Home Health Facilites by State normalized to population

dfhh = pd.read_sql_query("SELECT A.STATE, COUNT(A.STATE) AS TOTAL_HOMEHLTH, B.Pop_2020 AS STATE_POP,
                         (COUNT(A.STATE)/B.Pop_2020)*1000000 AS HH_RATIO FROM pachh_2020v2 A
                         JOIN usstatepop_2020 B ON A.STATE = B.State_Abbr where A.STATE != 'NATIONAL TOTAL'
                         GROUP BY A.STATE ORDER BY HH_RATIO DESC" , conn)
dfhh = pd.DataFrame(dfhh)
print(dfhh.head())

import plotly.graph_objects as go

import pandas as pd

fig = go.Figure(data=go.Choropleth(
    locations=dfhh['STATE'],
    z = dfhh['HH_RATIO'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Reds',
    colorbar_title = "FACILITES to STATE POPULATION RATIO * (10^6)",
))

fig.update_layout(
    title_text = 'Home Health Facilities in Different States (Normalized with State Population data)',
    geo_scope='usa',
)

fig.show()
```

Figure 103: Code for visualizing Number of Home Health Facilities per State normalized with population data on a Choropleth

```
# Inpatient Rehab Facilites by State normalized to population

dfirf = pd.read_sql_query("SELECT A.STATE, COUNT(A.STATE) AS TOTAL_IRF, B.Pop_2020 AS STATE_POP,
                          (COUNT(A.STATE)/B.Pop_2020)*1000000 AS IRF_RATIO FROM pacirf_2020v2 A
                          JOIN usstatepop_2020 B ON A.STATE = B.State_Abbr where A.STATE != 'NATIONAL TOTAL'
                          GROUP BY A.STATE ORDER BY IRF_RATIO DESC" , conn)
dfirf = pd.DataFrame(dfirf)
print(dfirf.head())

import plotly.graph_objects as go

import pandas as pd

fig = go.Figure(data=go.Choropleth(
    locations=dfirf['STATE'],
    z = dfirf['IRF_RATIO'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Greens',
    colorbar_title = "FACILITES to STATE POPULATION RATIO * (10^6)",
))

fig.update_layout(
    title_text = 'Inpatient Rehab Facilities in Different States (Normalized with State Population data)',
    geo_scope='usa',
)

fig.show()
```

Figure 104: Code for visualizing Number of Inpatient Rehab Facilities per State normalized with population data on a Choropleth

```
# Longterm Care Facilites by State

dfltc = pd.read_sql_query("SELECT A.STATE, COUNT(A.STATE) AS TOTAL_LTC, B.Pop_2020 AS STATE_POP,
                          (COUNT(A.STATE)/B.Pop_2020)*1000000 AS LTC_RATIO FROM pacltc_2020v2 A
                          JOIN usstatepop_2020 B ON A.STATE = B.State_Abbr where A.STATE != 'NATIONAL TOTAL'
                          GROUP BY A.STATE ORDER BY LTC_RATIO DESC" , conn)
dfltc = pd.DataFrame(dfltc)
print(dfltc.head())

import plotly.graph_objects as go

import pandas as pd

fig = go.Figure(data=go.Choropleth(
    locations=dfltc['STATE'],
    z = dfltc['LTC_RATIO'].astype(float),
    locationmode = 'USA-states',
    colorscale = 'Blues',
    colorbar_title = "FACILITES to STATE POPULATION RATIO * (10^6)",
))

fig.update_layout(
    title_text = 'Longterm Care Facilities in Different States (Normalized with State Population data)',
    geo_scope='usa',
)

fig.show()
```

Figure 105: Code for visualizing Number of Long-Term Care Facilities per State normalized with population data on a Choropleth

Section III: Code for Creating Bar Graphs to visualize number of facilities in different cities across Indiana State as Number of Facilities and with population normalization

```
#Outpatient Facilites In Indiana State

dfopd_ct = pd.read_sql_query(" SELECT PRV_City, COUNT(*) AS TOTAL_OPD FROM Outpatient_2020_PrvSrv
                             WHERE Prv_State in ('IN') GROUP BY PRV_City ORDER BY TOTAL_OPD DESC limit 20" , conn)
dfopd_ct = pd.DataFrame(dfopd_ct)

dfopd_ct.plot.barh('PRV_City','TOTAL_OPD')
plt.xlabel('CITY')
plt.ylabel('NUMBER OF FACILITES')
plt.title('INDIANA CITIES WITH HIGHEST NUMBER OF OUTPATIENT FACILITIES')
plt.show()
```

Figure 106: Code for visualizing Number of Outpatient Facilities per City in cities in Indiana State on a Horizontal Bar graph

```
#SNF Facilites In Indiana State

snf_ct = pd.read_sql_query(" SELECT PRVDR_CITY, COUNT(*) AS TOTAL_SNF FROM MD_PAC_SNF_2020 where STATE = 'IN'
                           GROUP BY PRVDR_CITY ORDER BY TOTAL_SNF DESC limit 20" , conn)
snf_ct = pd.DataFrame(snf_ct)
snf_ct.head()

snf_ct.plot.barh('PRVDR_CITY','TOTAL_SNF')
plt.xlabel('CITY')
plt.ylabel('NUMBER OF FACILITES')
plt.title('INDIANA CITIES WITH HIGHEST NUMBER OF SNF FACILITIES')
plt.show()
```

Figure 107: Code for visualizing Number of SNF Facilities per City in cities in Indiana State on a Horizontal Bar graph

```
#Hospice Facilites In Indiana State

hos_ct = pd.read_sql_query(" SELECT PRVDR_CITY, COUNT(*) AS TOTAL_HOSPICE FROM MD_PAC_HOS_2020
                           where STATE = 'IN' GROUP BY PRVDR_CITY ORDER BY TOTAL_HOSPICE DESC LIMIT 20" , conn)
hos_ct = pd.DataFrame(hos_ct)
hos_ct.head()

hos_ct.plot.barh('PRVDR_CITY','TOTAL_HOSPICE')
plt.xlabel('CITY')
plt.ylabel('NUMBER OF FACILITES')
plt.title('INDIANA CITIES WITH HIGHEST NUMBER OF HOSPICE FACILITIES')
plt.show()
```

Figure 108: Code for visualizing Number of Hospice Facilities per City in cities in Indiana State on a Horizontal Bar graph

```
#Outpatient Facilites In Indiana State normalized to population

dfopd_ct = pd.read_sql_query("SELECT A.Prv_City as CITY, COUNT(A.Prv_City) AS TOTAL_OPD, B.POPEST2020 AS CITY_POP,
                             (COUNT(A.Prv_City)/B.POPEST2020)*10000 AS OPD_CITY_RATIO FROM Outpatient_2020_PrvSrv A
                             JOIN city_pop_2020 B ON A.Prv_State = B.STATE_ABBR AND A.Prv_City = B.CITY
                             WHERE A.Prv_State in ('IN') AND A.Prv_City = B.CITY GROUP BY A.Prv_City
                             ORDER BY OPD_CITY_RATIO DESC limit 20" , conn)
dfopd_ct = pd.DataFrame(dfopd_ct)
print(dfopd_ct.head())

dfopd_ct.plot.barh('CITY','OPD_CITY_RATIO')
plt.xlabel('FACILITES to CITY POPULATION RATIO * (10^4)')
plt.ylabel('CITY')
plt.yticks(fontsize = 10)
plt.title('OUTPATIENT FACILITES IN INDIANA CITIES (Normalized with City Population data)')
plt.tight_layout()
plt.show()
```

Figure 109: Code for visualizing Number of Outpatient Facilities per City in cities in Indiana State normalized to population data on a Horizontal Bar graph

```
#SNF Facilites In Indiana State normalized to population

snf_ct = pd.read_sql_query("SELECT A.PRVDR_CITY as CITY, COUNT(A.PRVDR_CITY) AS TOTAL_OPD, B.POPEST2020 AS CITY_POP,
                           (COUNT(A.PRVDR_CITY)/B.POPEST2020)*10000 AS SNF_CITY_RATIO FROM MD_PAC_SNF_2020 A
                           JOIN city_pop_2020 B ON A.STATE = B.STATE_ABBR AND A.PRVDR_CITY = B.CITY
                           where A.STATE = 'IN' AND A.PRVDR_CITY = B.CITY AND A.STATE = B.STATE_ABBR
                           GROUP BY A.PRVDR_CITY ORDER BY SNF_CITY_RATIO DESC limit 20" , conn)
snf_ct = pd.DataFrame(snf_ct)
print(snf_ct.head())

snf_ct.plot.barh('CITY','SNF_CITY_RATIO')
plt.xlabel('FACILITES to CITY POPULATION RATIO * (10^4)')
plt.ylabel('CITY')
plt.yticks(fontsize = 10)
plt.title('SNF FACILITES IN INDIANA CITIES (Normalized with City Population data)')
plt.tight_layout()
plt.show()
```

Figure 110: Code for visualizing Number of SNF Facilities per City in cities in Indiana State normalized to population data on a Horizontal Bar graph

```
#Hospice Facilites In Indiana State normalized to population

hos_ct = pd.read_sql_query("SELECT A.PRVDR_CITY as CITY, COUNT(A.PRVDR_CITY) AS TOTAL_HOSPICE, B.POPEST2020 AS CITY_POP,
                           (COUNT(A.PRVDR_CITY)/B.POPEST2020)*10000 AS HSP_CITY_RATIO FROM MD_PAC_HOS_2020 A
                           JOIN city_pop_2020 B ON A.STATE = B.STATE_ABBR AND A.PRVDR_CITY = B.CITY
                           AND A.STATE = B.STATE_ABBR where A.STATE = 'IN' AND A.PRVDR_CITY = B.CITY
                           AND A.STATE = B.STATE_ABBR GROUP BY A.PRVDR_CITY ORDER BY HSP_CITY_RATIO DESC LIMIT 20" , conn)
hos_ct = pd.DataFrame(hos_ct)
print(hos_ct.head())

hos_ct.plot.barh('CITY','HSP_CITY_RATIO')
plt.ylabel('CITY')
plt.xlabel('FACILITES to CITY POPULATION RATIO * (10^4)')
plt.yticks(fontsize = 10)
plt.title('HOSPICE FACILITES IN INDIANA CITIES (Normalized with City Population data)')
plt.tight_layout()
plt.show()
```

Figure 111: Code for visualizing Number of Hospice Facilities per City in cities in Indiana State normalized to population data on a Horizontal Bar graph

```
#Home Health Facilites In Indiana State normalized to population

hh_ct = pd.read_sql_query("SELECT A.PRVDR_CITY as CITY, COUNT(A.PRVDR_CITY) AS TOTAL_HOMEHLTH, B.POPEST2020 AS CITY_POP,
                          (COUNT(A.PRVDR_CITY)/B.POPEST2020)*10000 AS HH_CITY_RATIO FROM pachh_2020v2 A
                          JOIN city_pop_2020 B ON A.STATE = B.STATE_ABBR AND A.PRVDR_CITY = B.CITY
                          AND A.STATE = B.STATE_ABBR where A.STATE = 'IN' AND A.PRVDR_CITY = B.CITY
                          AND A.STATE = B.STATE_ABBR GROUP BY A.PRVDR_CITY ORDER BY HH_CITY_RATIO DESC LIMIT 20" , conn)
hh_ct = pd.DataFrame(hh_ct)
print(hh_ct.head())

hh_ct.plot.barh('CITY','HH_CITY_RATIO')
plt.ylabel('CITY')
plt.xlabel('FACILITES to CITY POPULATION RATIO * (10^4)')
plt.yticks(fontsize = 10)
plt.title('HOME HEALTH FACILITES IN INDIANA CITIES (Normalized with City Population data)')
plt.tight_layout()
plt.show()
```

Figure 112: Code for visualizing Number of Home Health Facilities per City in cities in Indiana State normalized to population data on a Horizontal Bar graph

```
#IRF Facilites In Indiana State normalized to population

irf_ct = pd.read_sql_query("SELECT A.PRVDR_CITY as CITY, COUNT(A.PRVDR_CITY) AS TOTAL_IRF, B.POPEST2020 AS CITY_POP,
                           (COUNT(A.PRVDR_CITY)/B.POPEST2020)*10000 AS IRF_CITY_RATIO FROM pacirf_2020v2 A
                           JOIN city_pop_2020 B ON A.STATE = B.STATE_ABBR AND A.PRVDR_CITY = B.CITY
                           AND A.STATE = B.STATE_ABBR where A.STATE = 'IN' AND A.PRVDR_CITY = B.CITY
                           AND A.STATE = B.STATE_ABBR GROUP BY A.PRVDR_CITY ORDER BY IRF_CITY_RATIO DESC LIMIT 20" , conn)
irf_ct = pd.DataFrame(irf_ct)
print(irf_ct.head())

irf_ct.plot.barh('CITY','IRF_CITY_RATIO')
plt.ylabel('CITY')
plt.xlabel('FACILITES to CITY POPULATION RATIO * (10^4)')
plt.yticks(fontsize = 10)
plt.title('INPATIENT REHAB FACILITES IN INDIANA CITIES (Normalized with City Population data)')
plt.tight_layout()
plt.show()
```

Figure 113: Code for visualizing Number of IRF Facilities per City in cities in Indiana State normalized to population data on a Horizontal Bar graph

```
# Longterm Care Facilites In Indiana State normalized to population

ltc_ct = pd.read_sql_query("SELECT A.PRVDR_CITY as CITY, COUNT(A.PRVDR_CITY) AS TOTAL_LTC, B.POPEST2020 AS CITY_POP,
                           (COUNT(A.PRVDR_CITY)/B.POPEST2020)*10000 AS LTC_CITY_RATIO FROM pacltc_2020v2 A
                           JOIN city_pop_2020 B ON A.STATE = B.STATE_ABBR AND A.PRVDR_CITY = B.CITY
                           AND A.STATE = B.STATE_ABBR where A.STATE = 'IN' AND A.PRVDR_CITY = B.CITY
                           AND A.STATE = B.STATE_ABBR GROUP BY A.PRVDR_CITY ORDER BY LTC_CITY_RATIO DESC LIMIT 20" , conn)
ltc_ct = pd.DataFrame(ltc_ct)
print(ltc_ct.head())

ltc_ct.plot.barh('CITY','LTC_CITY_RATIO')
plt.ylabel('CITY')
plt.xlabel('FACILITES to CITY POPULATION RATIO * (10^4)')
plt.yticks(fontsize = 10)
plt.title('LONGTERM CARE FACILITES IN INDIANA CITIES (Normalized with City Population data)')
plt.tight_layout()
plt.show()
```

Figure 114: Code for visualizing Number of LTC Facilities per City in cities in Indiana State normalized to population data on a Horizontal Bar graph

Section IV: Code for Visualizing Distribution of patients by gender

```
hos_mafm = pd.read_sql_query(" SELECT STATE, ROUND(((SUM(BENE_MALE_PCT)/COUNT(BENE_MALE_PCT))*100),2) AS MALE_BENE,
                              ROUND(((SUM(BENE_FEML_PCT)/COUNT(BENE_FEML_PCT))*100),2) AS FEMALE_BENE
                              FROM MD_PAC_HOS_2020 WHERE PRVDR_NAME != 'STATE TOTAL' AND STATE != 'NATIONAL TOTAL'
                              GROUP BY STATE ORDER BY ((SUM(BENE_MALE_PCT)/COUNT(BENE_MALE_PCT))*100) DESC" , conn)
hos_mafm = pd.DataFrame(hos_mafm)
hos_mafm.head()

trace1 = go.Bar(
    x=hos_mafm.STATE,
    y=hos_mafm.MALE_BENE,
    marker=dict(color='#e584f7'),
    name='Percent Male beneficiaries'
)
trace2 = go.Bar(
    x=hos_mafm.STATE,
    y=hos_mafm.FEMALE_BENE,
    marker=dict(color='#a0f784'),
    name='Percent Female beneficiaries'
)

data = [trace1, trace2]
layout = go.Layout(barmode='group', legend=dict(orientation='h'), title='Percent of Male and Female Patients in 2020
                   in Hospices in STATES across the US')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```

Figure 115: Code for assessing differences in access to Hospice facilities by gender at the state level, on a bar graph

```
hos_mafmct = pd.read_sql_query(" SELECT PRVDR_CITY, ROUND(((SUM(BENE_MALE_PCT)/COUNT(BENE_MALE_PCT))*100),2) AS MALE_BENE,
                               ROUND(((SUM(BENE_FEML_PCT)/COUNT(BENE_FEML_PCT))*100),2) AS FEMALE_BENE
                               FROM MD_PAC_HOS_2020 WHERE PRVDR_NAME != 'STATE TOTAL' AND STATE != 'NATIONAL TOTAL'
                               GROUP BY PRVDR_CITY ORDER BY COUNT(BENE_MALE_PCT) DESC LIMIT 20" , conn)
hos_mafmct = pd.DataFrame(hos_mafmct)
hos_mafmct.head()

trace1 = go.Bar(
    x=hos_mafmct.PRVDR_CITY,
    y=hos_mafmct.MALE_BENE,
    marker=dict(color='#e584f7'),
    name='Percent Male beneficiaries'
)
trace2 = go.Bar(
    x=hos_mafmct.PRVDR_CITY,
    y=hos_mafmct.FEMALE_BENE,
    marker=dict(color='#a0f784'),
    name='Percent Female beneficiaries'
)

data = [trace1, trace2]
layout = go.Layout(barmode='group', legend=dict(orientation='h'), title='Percent of Male and Female Patients in 2020
                  in Hospices in the Top 20 Cities across the US')
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='grouped-bar')
```

Figure 116: Code for assessing differences in access to Hospice facilities by gender in cities with the cities with the top 20 highest male patient counts, on a bar graph

Section V: Code for Visualizing Distribution of patients by Ethnicity

```
hos_ethn = pd.read_sql_query("SELECT ROUND(BENE_RACE_WHT_PCT*100,2) AS WHITE,
                              ROUND(BENE_RACE_BLACK_PCT*100,2) AS AFRICAN_AMERICAN,
                              ROUND(BENE_RACE_API_PCT*100,2) AS ASIAN_PACIFIC_ISLANDER,
                              ROUND(BENE_RACE_HSPNC_PCT*100,2) AS HISPANIC,
                              ROUND(BENE_RACE_NATIND_PCT*100,2) AS AMERICAN_INDIAN,
                              ROUND(BENE_RACE_OTHR_PCT*100,2) AS OTHER FROM MD_PAC_HOS_2020
                              WHERE STATE = 'NATIONAL TOTAL'" , conn)
hos_ethn = pd.DataFrame(hos_ethn)
hos_ethn = hos_ethn.T
hos_ethn.columns=['Percent']
hos_ethn = hos_ethn.assign(Race=['WHITE', 'AFRICAN_AMERICAN', 'ASIAN_PACIFIC_ISLANDER','HISPANIC','AMERICAN_INDIAN','OTHER'])
hos_ethn

fig, sc = plt.subplots(figsize=(8, 8))
sc.pie(hos_ethn['Percent'], labels=hos_ethn['Race'], autopct='%.1f%%')
sc.set_title("Access to Hospice Care by Ethnicity At the National Level")
plt.show()
```

Figure 117: Code for assessing differences in access to Hospice facilities by ethnicity at the national level using a pie chart

```
hos_ethn_st = pd.read_sql_query("SELECT STATE, ROUND(BENE_RACE_WHT_PCT*100,2) AS WHITE,
                                 ROUND(BENE_RACE_BLACK_PCT*100,2) AS AFRICAN_AMERICAN,
                                 ROUND(BENE_RACE_API_PCT*100,2) AS ASIAN_PACIFIC_ISLANDER,
                                 ROUND(BENE_RACE_HSPNC_PCT*100,2) AS HISPANIC,
                                 ROUND(BENE_RACE_NATIND_PCT*100,2) AS AMERICAN_INDIAN,
                                 ROUND(BENE_RACE_OTHR_PCT*100,2) AS OTHER FROM MD_PAC_HOS_2020
                                 WHERE SMRY_CTGRY = 'STATE'" , conn)
hos_ethn_st = pd.DataFrame(hos_ethn_st)
hos_ethn_st.head()
```

	STATE	WHITE	AFRICAN_AMERICAN	ASIAN_PACIFIC_ISLANDER	HISPANIC	AMERICAN_INDIAN	OTHER
0	AK	75.0	2.0	3.0	3.0	12.0	2.0
1	AL	80.0	18.0	0.0	0.0	0.0	0.0
2	AR	88.0	8.0	0.0	0.0	0.0	0.0
3	AZ	83.0	2.0	1.0	10.0	1.0	0.0
4	CA	66.0	5.0	8.0	17.0	0.0	1.0

Figure 118: Code and partial output to assess differences in access to Hospice facilities by ethnicity in each state

```
hos_ethnin = pd.read_sql_query("SELECT ROUND(BENE_RACE_WHT_PCT*100,2) AS WHITE,
                                ROUND(BENE_RACE_BLACK_PCT*100,2) AS AFRICAN_AMERICAN,
                                ROUND(BENE_RACE_API_PCT*100,2) AS ASIAN_PACIFIC_ISLANDER,
                                ROUND(BENE_RACE_HSPNC_PCT*100,2) AS HISPANIC,
                                ROUND(BENE_RACE_NATIND_PCT*100,2) AS AMERICAN_INDIAN,
                                ROUND(BENE_RACE_OTHR_PCT*100,2) AS OTHER FROM MD_PAC_HOS_2020
                                WHERE STATE = 'IN' and SMRY_CTGRY =  'STATE'" , conn)
hos_ethnin = pd.DataFrame(hos_ethnin)
hos_ethnin
hos_ethnin = hos_ethnin.T
hos_ethnin.columns=['Percent']
hos_ethnin = hos_ethnin.assign(Race=['WHITE', 'AFRICAN_AMERICAN', 'ASIAN_PACIFIC_ISLANDER','HISPANIC','AMERICAN_INDIAN',
                                     'OTHER'])
hos_ethnin

fig, sc = plt.subplots(figsize=(10, 10))
sc.pie(hos_ethnin['Percent'], labels=hos_ethnin['Race'], autopct='%.1f%%')
sc.set_title("Access to Hospice Care by Ethnicity in INDIANA State")
plt.show()
```

Figure 119: Code for assessing differences in access to Hospice facilities by ethnicity in the state of Indiana, using a pie chart

Section VI: Code for Clustering and PCA analysis of PAC and Hospice facility subsets

a) Hospice Agencies

Clustering Hospice Facilites by Stay Details, Medicare charges, Distribution of patients by Gender, Ethnicity, Chronic Conditions

```
hos_cl = pd.read_sql_query("SELECT DISTINCT PRVDR_ID,PRVDR_NAME,PRVDR_CITY,STATE,BENE_DSTNCT_CNT,TOT_EPSD_STAY_CNT,
                            TOT_SRVC_DAYS,TOT_CHRG_AMT,TOT_ALOWD_AMT, TOT_MDCR_PYMT_AMT, TOT_MDCR_STDZD_PYMT_AMT,
                            BENE_MALE_PCT,BENE_FEML_PCT,BENE_RACE_WHT_PCT, BENE_RACE_BLACK_PCT,BENE_RACE_API_PCT,
                            BENE_RACE_HSPNC_PCT,BENE_RACE_NATIND_PCT,BENE_RACE_OTHR_PCT, BENE_AVG_RISK_SCRE,
                            BENE_PRMRY_DX_CNCR_PCT,BENE_PRMRY_DX_COPD_PCT,BENE_PRMRY_DX_RSPRTRYFAILR_PCT,
                            BENE_PRMRY_DX_DMNT_PCT, BENE_PRMRY_DX_STROK_PCT,BENE_PRMRY_DX_CHF_PCT,
                            BENE_PRMRY_DX_HYPRTNSN_PCT,BENE_PRMRY_DX_OTHRCRDVSCLR_PCT,BENE_PRMRY_DX_INFCTN_PCT,
                            BENE_PRMRY_DX_ORTHO_PCT,BENE_PRMRY_DX_INJURY_PCT,BENE_PRMRY_DX_MTR_NRL_PCT,
                            BENE_PRMRY_DX_DBTS_PCT FROM MD_PAC_HOS_2020 WHERE SMRY_CTGRY = 'PROVIDER'
                            ORDER BY PRVDR_ID, STATE, PRVDR_CITY" , conn)
hos_cl = pd.DataFrame(hos_cl)
hos_cl.head()
```

Figure 120: Creating a data frame for 35 features for Hospice facilities for cluster analysis

```
X2 = hos_cl
X2.index = X2.PRVDR_ID
X2 = X2.drop(['PRVDR_ID', 'PRVDR_NAME', 'PRVDR_CITY', 'STATE'], axis=1)
X2 = X2.dropna()

birch_silhouette_scores = []

for n_cluster in range(2, 8):
    birch_silhouette_scores.append(
        silhouette_score(X2, Birch(n_clusters = n_cluster).fit_predict(X2)))

k = np.arange(2,8)

plt.bar(k, birch_silhouette_scores)
plt.title('Hospice Facilities - Assessing Cluster Numbers with BIRCH', fontsize = 10)
plt.xlabel('Number of Clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```

Figure 121: Dropping provider identifier details and running a loop to determine best number of clusters to segregate Hospice facilities

Dimension Reduction with PCA

```
from sklearn.preprocessing import StandardScaler

xp2 = StandardScaler().fit_transform(X2)
print(xp2.shape)
feat_cols2 = ['feature'+str(i) for i in range(xp2.shape[1])]
norm_xp2 = pd.DataFrame(xp2,columns=feat_cols2)
norm_xp2.tail()
```

```
(4690, 30)
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	...	feature20	feature21	feature22
4685	-0.437990	-0.402526	-0.385027	-0.380361	-0.355296	-0.355296	-0.348210	-2.997623	-3.165884	-0.401904	...	-0.770947	-0.856239	-0.481451
4686	-0.444224	-0.411119	-0.394648	-0.388348	-0.366109	-0.366109	-0.355962	-2.997623	-3.165884	0.170733	...	-0.770947	-0.856239	-0.481451
4687	-0.410562	-0.388562	-0.375300	-0.368150	-0.340295	-0.340295	-0.337745	0.076436	0.416190	-0.516431	...	-0.770947	-0.856239	-0.481451
4688	-0.445470	-0.418280	-0.403187	-0.397257	-0.376503	-0.376503	-0.363996	-2.997623	-3.165884	-2.806976	...	-0.770947	-0.856239	-0.481451
4689	-0.440484	-0.411477	-0.395039	-0.392098	-0.366457	-0.366457	-0.356473	-2.997623	-3.165884	-0.134673	...	-0.770947	-0.856239	-0.481451

5 rows × 30 columns

Figure 122: Standardized scalar transformation of Hospice features for PCA analysis.

```
# Visualize Hospice Clusters

def cluster_visualization(df, columns, labels):

    df = df[columns]
    pca = PCA(17)
    pca.fit(df)
    X_PCA = pca.transform(df)
    X_PCA.shape
    print('Explained variation for Top 2 principal components:', var_list[0] + var_list[1])

    x, y = X_PCA[:, 0], X_PCA[:, 1]

    colors = {0: 'red'
              ,1: 'blue'
             }

    names = {0: 'Cluster 1'
             ,1: 'Cluster 2'
            }

    df = pd.DataFrame({'x': x, 'y':y, 'label':labels})
    groups = df.groupby('label')

    fig, ax = plt.subplots(figsize=(8, 5))

    for name, group in groups:
        ax.plot(group.x, group.y, marker='o', linestyle='', ms=5,
                color=colors[name],label=names[name], mec='none')
        ax.set_aspect('auto')
        ax.tick_params(axis='x',which='both',bottom='off',top='off',labelbottom='off')
        ax.tick_params(axis= 'y',which='both',left='off',top='off',labelleft='off')

    ax.legend()
    ax.set_title("Hospice Facility Clustering Clustering Visualization - with Top 2 Principal Components")
    plt.show()

cluster_visualization(norm_xp2, columns = norm_xp2.columns[:-1], labels = norm_xp2["cluster2"])
```

```
Explained variation for Top 2 principal components: 0.43226152738175705
```

Figure 123: Code for visualizing Hospice clusters on a scatter plot using the top 2 principal components that account for the maximum amount of explained variation. The explained variation is printed in the output.

b) **Home Health Agencies**

Clustering Home Health Facilites by Stay Details, Medicare charges, Distribution of patients by Gender, Ethnicity, Chronic Conditions

```
hh_cl = pd.read_sql_query("SELECT PRVDR_ID,PRVDR_NAME,PRVDR_CITY,STATE,BENE_DSTNCT_CNT,TOT_EPSD_STAY_CNT,TOT_SRVC_DAYS,
                          TOT_CHRG_AMT,TOT_ALOWD_AMT,TOT_MDCR_PYMT_AMT,TOT_MDCR_STDZD_PYMT_AMT,BENE_AVG_AGE,BENE_MALE_PCT,
                          BENE_FEML_PCT,BENE_RACE_WHT_PCT,BENE_RACE_BLACK_PCT,BENE_RACE_API_PCT,BENE_RACE_HSPNC_PCT,
                          BENE_RACE_NATIND_PCT,BENE_RACE_OTHR_PCT,BENE_AVG_RISK_SCRE,BENE_CC_AF_PCT,BENE_CC_ALZHMR_PCT,
                          BENE_CC_ASTHMA_PCT,BENE_CC_CNCR_PCT,BENE_CC_CHF_PCT,BENE_CC_CKD_PCT,BENE_CC_COPD_PCT,
                          BENE_CC_DPRSSN_PCT,BENE_CC_DBTS_PCT,BENE_CC_IHD_PCT,BENE_CC_OPO_PCT,BENE_CC_RAOA_PCT,
                          BENE_CC_SZ_PCT,BENE_CC_STROK_PCT FROM pachh_2020v2 WHERE SMRY_CTGRY = 'PROVIDER'" , conn)
hh_cl = pd.DataFrame(hh_cl)
hh_cl.head()
```

Figure 124: Creating a data frame for 35 features for Home Health facilities for cluster analysis

```
from sklearn.cluster import Birch, MiniBatchKMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_samples, silhouette_score


X3 = hh_cl
X3.index = X3.PRVDR_ID
X3 = X3.drop(['PRVDR_ID', 'PRVDR_NAME', 'PRVDR_CITY', 'STATE'], axis=1)
X3 = X3.dropna()


birch_silhouette_scores = []

for n_cluster in range(2, 8):

    birch_silhouette_scores.append(
        silhouette_score(X3, Birch(n_clusters = n_cluster).fit_predict(X3)))

k = np.arange(2,8)

plt.bar(k, birch_silhouette_scores)
plt.title('Home Health Facilities - Assessing Cluster Numbers with BIRCH', fontsize = 10)
plt.xlabel('Number of Clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```

Figure 125: Dropping provider identifier details and running a loop to determine best number of clusters to segregate Home Health facilities

Dimension Reduction With PCA

```
from sklearn.preprocessing import StandardScaler

xp3 = StandardScaler().fit_transform(X3) # normalizing the features
print(xp3.shape)
feat_cols3 = ['feature'+str(i) for i in range(xp3.shape[1])]
norm_xp3= pd.DataFrame(xp3,columns=feat_cols3)
norm_xp3.tail()
```

```
(8591, 32)
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	...	feature22	feature23	feature24
8586	-0.288452	-0.305198	-0.309266	-0.326625	-0.287517	-0.287517	-0.305521	-0.639546	0.559104	0.159608	...	-0.027190	0.863491	0.026074
8587	-0.438394	-0.476084	-0.481729	-0.344885	-0.429441	-0.429441	-0.451256	0.136637	0.342125	0.364027	...	-1.611483	-0.509671	-0.346750
8588	-0.100645	0.018288	0.037295	-0.126069	-0.003431	-0.003431	-0.009692	-0.639546	0.414451	0.261818	...	-0.534164	0.574405	0.461036
8589	-0.518666	-0.586435	-0.597412	-0.493733	-0.546803	-0.546803	-0.572206	-0.898273	-2.550928	-2.702262	...	-3.576006	-1.955104	-2.459421
8590	-0.512608	-0.573824	-0.583571	-0.492346	-0.543642	-0.543642	-0.569138	-0.122091	-2.550928	-2.702262	...	0.289669	-1.955104	-2.459421

5 rows × 32 columns

Figure 126: Standardized scalar transformation of Home Health features for PCA analysis.

```
# Visualize Home Health Clusters

def cluster_visualization(df, columns, labels):

    df = df[columns]
    pca = PCA(16)
    pca.fit(df)
    X_PCA = pca.transform(df)
    X_PCA.shape
    print('Explained variation for Top 2 principal components:', var_list[0] + var_list[1])

    x, y = X_PCA[:, 0], X_PCA[:, 1]

    colors = {0: 'red'
              ,1: 'blue'
             }

    names = {0: 'Cluster 1'
             ,1: 'Cluster 2'
            }

    df = pd.DataFrame({'x': x, 'y':y, 'label':labels})
    groups = df.groupby('label')

    fig, ax = plt.subplots(figsize=(8, 5))

    for name, group in groups:
        ax.plot(group.x, group.y, marker='o', linestyle='', ms=5,
                color=colors[name],label=names[name], mec='none')
        ax.set_aspect('auto')
        ax.tick_params(axis='x',which='both',bottom='off',top='off',labelbottom='off')
        ax.tick_params(axis= 'y',which='both',left='off',top='off',labelleft='off')

    ax.legend()
    ax.set_title("Home Health Facility Clustering Visualization - with Top 2 Principal Components")
    plt.show()

cluster_visualization(norm_xp3, columns = norm_xp3.columns[:-1], labels = norm_xp3["cluster3"])
```

```
Explained variation for Top 2 principal components: 0.43226152738175705
```

Figure 127: Code for visualizing Home Health clusters on a scatter plot using the top 2 principal components that account for the maximum amount of explained variation. The explained variation is printed in the output.

```
# Analyze Distribution of Some Variables for Individual Clusters

import plotly.graph_objects as go

f1 = go.Box(y=hh1['TOT_CHRG_AMT'], name='Cluster 1')
f2 = go.Box(y=hh2['TOT_CHRG_AMT'], name='Cluster 2')
f3 = go.Box(y=hh1['BENE_DSTNCT_CNT'], name='Cluster 1')
f4 = go.Box(y=hh2['BENE_DSTNCT_CNT'], name='Cluster 2')
f5 = go.Box(y=hh1['TOT_SRVC_DAYS'], name='Cluster 1')
f6 = go.Box(y=hh2['TOT_SRVC_DAYS'], name='Cluster 2')
f7 = go.Box(y=hh1['BENE_CC_CNCR_PCT'], name='Cluster 1')
f8 = go.Box(y=hh2['BENE_CC_CNCR_PCT'], name='Cluster 2')


fig = subplots.make_subplots(rows=2, cols=2, print_grid=False, subplot_titles=('Home Health Medicare Charges',
                                                                               'Home Health Patient Counts',
                                                                               'Home Health Service Days','Cancer Patients'))
fig.append_trace(f1, 1, 1);
fig.append_trace(f2, 1, 1);
fig.append_trace(f3, 1, 2);
fig.append_trace(f4, 1, 2);
fig.append_trace(f5, 2, 1);
fig.append_trace(f6, 2, 1);
fig.append_trace(f7, 2, 2);
fig.append_trace(f8, 2, 2);


fig['layout'].update(height=800, width=800, showlegend=False);
iplot(fig, filename='simple-subplot');
```

Figure 128: Code for grouped Boxplot to visualize distribution of datapoints for certain features in the individual Home Health clusters

c) Inpatient Rehab Facilities

```
irf_cl = pd.read_sql_query("SELECT PRVDR_ID,PRVDR_NAME,PRVDR_CITY,STATE,BENE_DSTNCT_CNT,TOT_EPSD_STAY_CNT,TOT_SRVC_DAYS,
                           TOT_CHRG_AMT,TOT_ALOWD_AMT,TOT_MDCR_PYMT_AMT,TOT_MDCR_STDZD_PYMT_AMT,BENE_AVG_AGE,
                           BENE_MALE_PCT,BENE_FEML_PCT,BENE_RACE_WHT_PCT,BENE_RACE_BLACK_PCT,BENE_RACE_API_PCT,
                           BENE_RACE_HSPNC_PCT,BENE_RACE_NATIND_PCT,BENE_RACE_OTHR_PCT,BENE_AVG_RISK_SCRE,
                           BENE_CC_AF_PCT,BENE_CC_ALZHMR_PCT,BENE_CC_ASTHMA_PCT,BENE_CC_CNCR_PCT,BENE_CC_CHF_PCT,
                           BENE_CC_CKD_PCT,BENE_CC_COPD_PCT,BENE_CC_DPRSSN_PCT,BENE_CC_DBTS_PCT,BENE_CC_IHD_PCT,
                           BENE_CC_OPO_PCT,BENE_CC_RAOA_PCT,BENE_CC_SZ_PCT,BENE_CC_STROK_PCT FROM pacirf_2020v2
                           WHERE SMRY_CTGRY = 'PROVIDER'" , conn)
irf_cl = pd.DataFrame(irf_cl)
irf_cl.head()
```

Figure 129: Creating a data frame for 35 features for IRF facilities for cluster analysis

```
from sklearn.cluster import Birch, MiniBatchKMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_samples, silhouette_score


X4 = irf_cl
X4.index = X4.PRVDR_ID
X4 = X4.drop(['PRVDR_ID', 'PRVDR_NAME', 'PRVDR_CITY', 'STATE'], axis=1)
X4 = X4.dropna()


birch_silhouette_scores = []

for n_cluster in range(2, 8):

    birch_silhouette_scores.append(
        silhouette_score(X4, Birch(n_clusters = n_cluster).fit_predict(X4)))

k = np.arange(2,8)

plt.bar(k, birch_silhouette_scores)
plt.title('Inpatient Rehab Facilities - Assessing Cluster Numbers with BIRCH', fontsize = 10)
plt.xlabel('Number of Clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```

Figure 130: Dropping provider identifier details and running a loop to determine best number of clusters to segregate IRF facilities

Dimension Reduction with PCA

```
from sklearn.preprocessing import StandardScaler

xp4 = StandardScaler().fit_transform(X4)
print(xp4.shape)
feat_cols4 = ['feature'+str(i) for i in range(xp4.shape[1])]
norm_xp4= pd.DataFrame(xp4,columns=feat_cols4)
norm_xp4.tail()
```

```
(1098, 32)
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	...	feature22	feature23	feature24
1093	1.346517	1.460562	1.195376	0.187571	0.743497	0.723405	1.179851	0.337546	-0.899478	0.974024	...	0.020834	1.513271	0.878278
1094	2.201908	2.240807	2.219519	0.808129	1.990271	1.973554	2.404316	0.727678	-0.899478	0.974024	...	1.679768	0.894568	0.415982
1095	3.188368	3.238795	2.939962	1.290975	2.610396	2.608319	3.169396	-0.442718	-0.270416	0.359169	...	0.370083	0.894568	0.600901
1096	-0.257342	-0.281381	-0.394680	-0.451541	-0.324870	-0.324512	-0.284074	0.727678	0.107021	-0.009744	...	-1.376163	-0.254452	-0.601069
1097	0.580803	0.556324	0.770242	-0.102319	0.648755	0.651770	0.705847	-0.052586	-0.396229	0.482140	...	-1.376163	0.099092	-0.323691

5 rows × 32 columns

Figure 131: Standardized scalar transformation of IRF features for PCA analysis.

```
# Visualize IRF Clusters

def cluster_visualization(df, columns, labels):

    df = df[columns]
    pca = PCA(17)
    pca.fit(df)
    X_PCA = pca.transform(df)
    X_PCA.shape
    print('Explained variation for Top 2 principal components:', var_list[0] + var_list[1])

    x, y = X_PCA[:, 0], X_PCA[:, 1]

    colors = {0: 'red'
              ,1: 'blue'
             }

    names = {0: 'Cluster 1'
             ,1: 'Cluster 2'
            }

    df = pd.DataFrame({'x': x, 'y':y, 'label':labels})
    groups = df.groupby('label')

    fig, ax = plt.subplots(figsize=(8, 5))

    for name, group in groups:
        ax.plot(group.x, group.y, marker='o', linestyle='', ms=5,
                color=colors[name],label=names[name], mec='none')
        ax.set_aspect('auto')
        ax.tick_params(axis='x',which='both',bottom='off',top='off',labelbottom='off')
        ax.tick_params(axis= 'y',which='both',left='off',top='off',labelleft='off')

    ax.legend()
    ax.set_title("Inpatient Rehab Facility Clustering Visualization - with Top 2 Principal Components")
    plt.show()

cluster_visualization(norm_xp4, columns = norm_xp4.columns[:-1], labels = norm_xp4["cluster4"])
```

```
Explained variation for Top 2 principal components: 0.43226152738175705
```

Figure 132: Code for visualizing IRF clusters on a scatter plot using the top 2 principal components that account for the maximum amount of explained variation. The explained variation is printed in the output.

```
# Analyze Distribution of Some Variables for Individual Clusters

import plotly.graph_objects as go

f1 = go.Box(y=irf1['TOT_CHRG_AMT'], name='Cluster 1')
f2 = go.Box(y=irf2['TOT_CHRG_AMT'], name='Cluster 2')
f3 = go.Box(y=irf1['BENE_DSTNCT_CNT'], name='Cluster 1')
f4 = go.Box(y=irf2['BENE_DSTNCT_CNT'], name='Cluster 2')
f5 = go.Box(y=irf1['TOT_SRVC_DAYS'], name='Cluster 1')
f6 = go.Box(y=irf2['TOT_SRVC_DAYS'], name='Cluster 2')
f7 = go.Box(y=irf1['BENE_CC_CNCR_PCT'], name='Cluster 1')
f8 = go.Box(y=irf2['BENE_CC_CNCR_PCT'], name='Cluster 2')


fig = subplots.make_subplots(rows=2, cols=2, print_grid=False, subplot_titles=('IRF Medicare Charges','IRF Patient Counts',
                                                                               'IRF Service Days','Cancer Patients'))
fig.append_trace(f1, 1, 1);
fig.append_trace(f2, 1, 1);
fig.append_trace(f3, 1, 2);
fig.append_trace(f4, 1, 2);
fig.append_trace(f5, 2, 1);
fig.append_trace(f6, 2, 1);
fig.append_trace(f7, 2, 2);
fig.append_trace(f8, 2, 2);


fig['layout'].update(height=800, width=800, showlegend=False);
iplot(fig, filename='simple-subplot');
```

Figure 133: Code for grouped Boxplot to visualize distribution of datapoints for certain features in the individual IRF clusters

d) **Long Term Care Facilities**

```
ltc_cl = pd.read_sql_query("SELECT PRVDR_ID,PRVDR_NAME,PRVDR_CITY,STATE,BENE_DSTNCT_CNT,TOT_EPSD_STAY_CNT,TOT_SRVC_DAYS,
                            TOT_CHRG_AMT,TOT_ALOWD_AMT,TOT_MDCR_PYMT_AMT,TOT_MDCR_STDZD_PYMT_AMT,BENE_AVG_AGE,
                            BENE_MALE_PCT,BENE_FEML_PCT,BENE_RACE_WHT_PCT,BENE_RACE_BLACK_PCT,BENE_RACE_API_PCT,
                            BENE_RACE_HSPNC_PCT,BENE_RACE_NATIND_PCT,BENE_RACE_OTHR_PCT,BENE_AVG_RISK_SCRE,
                            BENE_CC_AF_PCT,BENE_CC_ALZHMR_PCT,BENE_CC_ASTHMA_PCT,BENE_CC_CNCR_PCT,BENE_CC_CHF_PCT,
                            BENE_CC_CKD_PCT,BENE_CC_COPD_PCT,BENE_CC_DPRSSN_PCT,BENE_CC_DBTS_PCT,BENE_CC_IHD_PCT,
                            BENE_CC_OPO_PCT,BENE_CC_RAOA_PCT,BENE_CC_SZ_PCT,BENE_CC_STROK_PCT FROM pacltc_2020v2
                            WHERE SMRY_CTGRY = 'PROVIDER'" , conn)
ltc_cl = pd.DataFrame(ltc_cl)
ltc_cl.head()
```

Figure 134: Creating a data frame for 35 features for LTC facilities for cluster analysis

```
from sklearn.cluster import Birch, MiniBatchKMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_samples, silhouette_score


X5 = ltc_cl
X5.index = X5.PRVDR_ID
X5 = X5.drop(['PRVDR_ID', 'PRVDR_NAME', 'PRVDR_CITY', 'STATE'], axis=1)
X5 = X5.dropna()


birch_silhouette_scores = []

for n_cluster in range(2, 8):

    birch_silhouette_scores.append(
        silhouette_score(X5, Birch(n_clusters = n_cluster).fit_predict(X5)))

k = np.arange(2,8)

plt.bar(k, birch_silhouette_scores)
plt.title('Longterm Care Facilities - Assessing Cluster Numbers with BIRCH', fontsize = 10)
plt.xlabel('Number of Clusters', fontsize = 10)
plt.ylabel('Silhouette Score', fontsize = 10)
plt.show()
```

Figure 135: Dropping provider identifier details and running a loop to determine best number of clusters to segregate LTC facilities

```
from sklearn.preprocessing import StandardScaler

xp5 = StandardScaler().fit_transform(X5) # normalizing the features
print(xp5.shape)
feat_cols5 = ['feature'+str(i) for i in range(xp5.shape[1])]
norm_xp5= pd.DataFrame(xp5,columns=feat_cols5)
norm_xp5.tail()
```

```
(352, 32)
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	...	feature22	feature23	feature24
347	-0.362900	-0.327780	-0.383970	-0.210477	-0.211699	-0.196968	-0.300487	-0.955950	-0.118913	0.414027	...	0.214566	-0.484966	-0.205001
348	0.152310	0.059563	0.240281	0.042746	0.168373	0.152007	0.424960	-0.955950	-0.118913	0.414027	...	0.214566	1.065164	0.424042
349	0.202575	0.124120	0.145100	0.076518	0.102570	0.121116	0.306452	-0.252722	-0.018294	0.301315	...	0.214566	1.065164	0.334179
350	-1.273941	-1.092533	-1.307927	-0.465069	-1.251399	-1.257831	-1.407065	-1.659177	-5.250473	-4.996187	...	-6.438485	-3.520637	-5.147479
351	-0.991204	-0.864100	-1.112921	-0.417692	-1.115165	-1.115931	-1.211848	-0.252722	0.484800	-0.262249	...	0.214566	0.935986	0.244316

5 rows × 32 columns

Figure 136: Standardized scalar transformation of LTC features for PCA analysis.

```
# Visualize LTC Clusters

def cluster_visualization(df, columns, labels):

    df = df[columns]
    pca = PCA(17)
    pca.fit(df)
    X_PCA = pca.transform(df)
    X_PCA.shape
    print('Explained variation for Top 2 principal components:', var_list[0] + var_list[1])

    x, y = X_PCA[:, 0], X_PCA[:, 1]

    colors = {0: 'red'
              ,1: 'blue'
             }

    names = {0: 'Cluster 1'
             ,1: 'Cluster 2'
            }

    df = pd.DataFrame({'x': x, 'y':y, 'label':labels})
    groups = df.groupby('label')

    fig, ax = plt.subplots(figsize=(8, 5))

    for name, group in groups:
        ax.plot(group.x, group.y, marker='o', linestyle='', ms=5,
                color=colors[name],label=names[name], mec='none')
        ax.set_aspect('auto')
        ax.tick_params(axis='x',which='both',bottom='off',top='off',labelbottom='off')
        ax.tick_params(axis= 'y',which='both',left='off',top='off',labelleft='off')

    ax.legend()
    ax.set_title("Long Term Care Facility Clustering Visualization - with Top 2 Principal Components")
    plt.show()

cluster_visualization(norm_xp5, columns = norm_xp5.columns[:-1], labels = norm_xp5["cluster5"])
```

```
Explained variation for Top 2 principal components: 0.43226152738175705
```

Figure 137: Code for visualizing LTC clusters on a scatter plot using the top 2 principal components that account for the maximum amount of explained variation. The explained variation is printed in the output.

References

1. Mues, K. E., Liede, A., Liu, J., Wetmore, J. B., Zaha, R., Bradbury, B. D., Collins, A. J., & Gilbertson, D. T. (2017). Use of the Medicare database in epidemiologic and health services research: a valuable source of real-world evidence on the older and disabled populations in the US. *Clinical epidemiology*, *9*, 267–277. https://doi.org/10.2147/CLEP.S105613
2. Hoffman, G. J., & Yakusheva, O. (2020). Association Between Financial Incentives in Medicare's Hospital Readmissions Reduction Program and Hospital Readmission Performance. *JAMA network open*, *3*(4), e202044. https://doi.org/10.1001/jamanetworkopen.2020.2044
3. Reid, R. O., Mafi, J. N., Baseman, L. H., Fendrick, A. M., & Damberg, C. L. (2021). Waste in the Medicare Program: a National Cross-Sectional Analysis of 2017 Low-Value Service Use and Spending. *Journal of general internal medicine*, *36*(8), 2478–2482. https://doi.org/10.1007/s11606-020-06061-0

4. Medicare Basics - https://www.medicare.gov/basics/get-started-with-medicare/medicare-basics/parts-of-medicare
5. Skilled Nursing Facilities - https://www.investopedia.com/terms/s/skilled-nursing-facility.asp#:~:text=A%20skilled%20nursing%20facility%20is%20an%20in-patient%20rehabilitation,physical%20and%20occupational%20therapists%2C%20speech%20pathologists%2C%20and%20audiologists.
6. Home Health coverage - https://www.medicare.gov/what-medicare-covers/whats-home-health-care#:~:text=Home%20health%20care%20is%20a,skilled%20nursing%20facility%20(SNF)
7. Hospice care - https://www.nia.nih.gov/health/what-are-palliative-care-and-hospice-care#hospice
8. Inpatient Rehabilitation Facilites - https://www.bacharach.org/what-is-an-irf/
9. Long term care facilities - https://www.cdc.gov/longtermcare/index.html
10. MacKay, E. J., Stubna, M. D., Chivers, C., Draugelis, M. E., Hanson, W. J., Desai, N. D., & Groeneveld, P. W. (2021). Application of machine learning approaches to administrative claims data to predict clinical outcomes in medical and surgical patient populations. PloS one, 16(6), e0252585. https://doi.org/10.1371/journal.pone.0252585
11. Wingrove, P., Liaw, W., Weiss, J., Petterson, S., Maier, J., & Bazemore, A. (2020). Using Machine Learning to Predict Primary Care and Advance Workforce Research. Annals of family medicine, 18(4), 334–340. https://doi.org/10.1370/afm.2550