\usepackage{fvextra} \DefineVerbatimEnvironment{Highlighting}{Verbatim}{breaklines,commandchars=\\\{\}}

\RecustomVerbatimEnvironment{verbatim}{Verbatim}{ showspaces = false, showtabs = false, breaksymbolleft={}, breaklines }

PSet 4

AUTHOR
Charisma Lambert and Prashanthi Subbiah

PUBLISHED
November 2, 2024

PS4: Due Sat Nov 2 at 5:00PM Central. Worth 100 points. 1. This problem set is a paired problem set. 2. Play paper, scissors, rock to determine who goes first. Call that person Partner 1. • Partner 1 (name and cnet ID):Charisma Lambert, charisml • Partner 2 (name and cnet ID): Prashanthi Subbiah, prashanthis 3. Partner 1 will accept the ps4 and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted. 4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **CLPS**. 5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set here" (Ahona Roy) (1 point) 6. Late coins used this pset: **1** Late coins left after submission: **3** 7. Knit your ps4.qmd to an PDF file to make ps4.pdf, • The PDF should not be more than 25 pages. Use head() and re-size figures when appropriate. 8. (Partner 1): push ps4.qmd and ps4.pdf to your github repo. 9. (Partner 1): submit ps4.pdf via Gradescope. Add your partner on Gradescope. 10. (Partner 1): tag your submission in Gradescope

Section 1: Download and explore the Provider of Services (POS) file (10 pts) Partner 1

1.

```
import pandas as pd
import os
import csv
import warnings
warnings.filterwarnings("ignore")

base_path = r"/Users/charismalambert/Downloads"

health_path_16 = os.path.join(base_path, "pos2016.csv")
health_data_16 = pd.read_csv(health_path_16)
```

I pulled the following variables:
Provider code: PRVDR_CTGRY_CD and PRVDR_CTGRY_SBTYP_CD CMS certification number: PRVDR_NUM Termination code: PGM_TRMNTN_CD Facility Name: FAC_NAME Zipcode: ZIP_CD

2.

```
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
      (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

short_term_len_16 = len(short_term_16)
print(f"There are {short_term_len_16} hospitals reported in the 2016 data.")
```

```
There are 7245 hospitals reported in the 2016 data.
```

a. There are 7,245 hospitals reported in the 2016 data.
b. I found a report from the American Hospital Association that there were 5,534 hospitals registered in the US in 2016. I think it differs because their data does not contain outliers or fuzz, such as a if a hospital closed at any point in 2016 they likely removed it from their dataset, whereas our dataset might have it for the full year.

3.

```
# Repeat 3 steps for 2017- 2019: 1) load data, 2) filter for short-term, and 3) find
      number of hospitals for that year.
health_path_17 = os.path.join(base_path, "pos2017.csv")
health_data_17 = pd.read_csv(health_path_17)
short_term_17 =  health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
      (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "pos2018.csv")
health_data_18 = pd.read_csv(health_path_18, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
      (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "pos2019.csv")
health_data_19 = pd.read_csv(health_path_19, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["PRVDR_CTGRY_SBTYP_CD"] == 1) &
      (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

short_term_len_17 = len(short_term_17)
short_term_len_18 = len(short_term_18)
short_term_len_19 = len(short_term_19)
```

```
# Append the hospital data from 2016 – 2019 together
combined_df_final = pd.concat([short_term_16, short_term_17, short_term_18,
      short_term_19], ignore_index= True)
combined_df_final
```

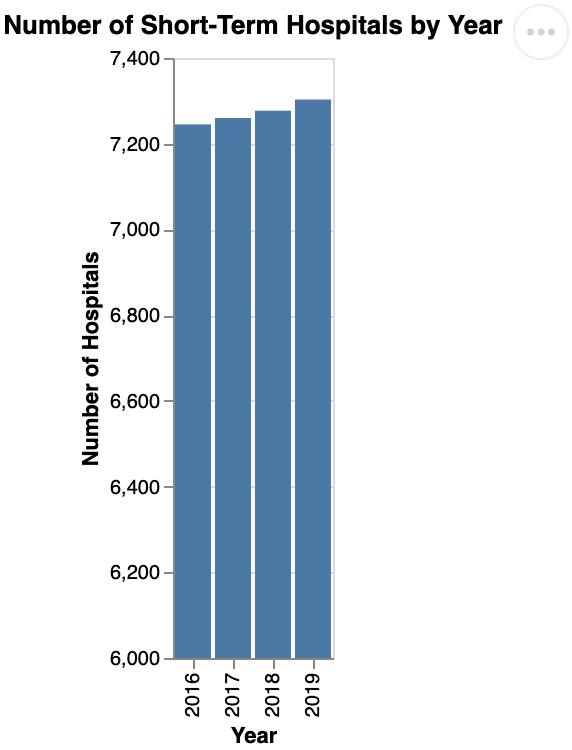
	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	CHOW_CNT	CHOW_DT	CITY_NAME	FAC_NAME
0	1.0	1	1	19730630.0	MULLENS	WYOMING COMM HOSP

	PRVDR_CTGRY_SBTYP_CD	PRVDR_CTGRY_CD	CHOW_CNT	CHOW_DT	CITY_NAME	FAC_NAME
						INC
1	1.0	1	1	19800401.0	STOCKTON	ST JOSEPHS PARKSIDE HOSPITAL
2	1.0	1	1	19800411.0	IRVING	PIONEER PARK MEDICAL CENTER
3	1.0	1	2	19800724.0	NEW ORLEANS	JO ELLEN SMITH MEMORIAL HOSPITAL
4	1.0	1	1	19800729.0	GORMAN	BLACKWELL HOSP S EASTLAND CO HOSP DIST
...	...	...	...	...	...	...
29080	1.0	1	0	NaN	CROCKETT	CROCKETT MEDICAL CENTER
29081	1.0	1	0	NaN	EL PASO	EL PASO LTA HOSPTIAL
29082	1.0	1	0	NaN	PFLUGERVILLE	BAYLOR SCOTT & WHITE MEDICAL CENTER ~ PFLUGERV...
29083	1.0	1	0	NaN	HOUSTON	THE HEIGHTS HOSPITAL
29084	1.0	1	0	NaN	SAN ANTONIO	SOUTHCROS HOSPITAL

29085 rows × 10 columns

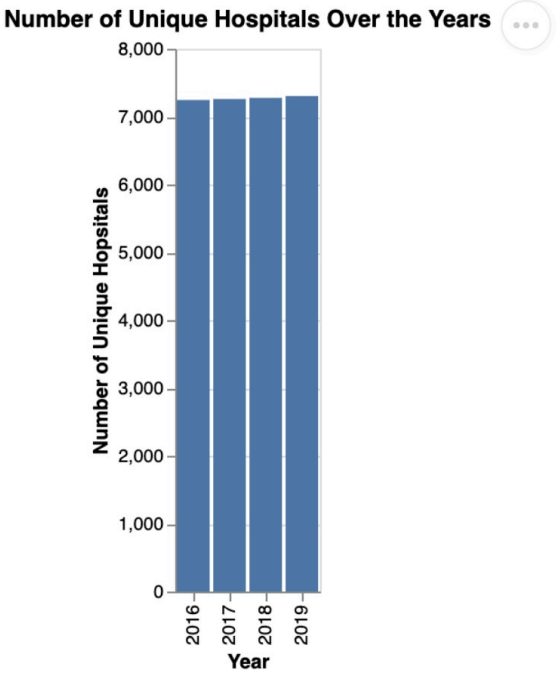
```
# Plot the number of observations by year
observations_by_year = combined_df_final.groupby("year").size().reset_index(name =
      "observations")

import altair as alt
obs_by_year = alt.Chart(observations_by_year).mark_bar().encode(
  x = alt.X("year:O", title = "Year"),
  y = alt.Y("observations:Q", title = "Number of Hospitals", scale = alt.Scale(domain
      = [6000, 7400], clamp = True), axis = alt.Axis(tickMinStep = 100))
```

```
  ).properties(
  title = "Number of Short-Term Hospitals by Year")

obs_by_year
```



4. a.

```
# Plot the number of unique hospitals
unique_hospitals_yr = combined_df_final.groupby("year")
      ["PRVDR_NUM"].nunique().reset_index()
unique_hospitals_yr.columns = ["year", "unique_hospitals"]

unique_hospitals_chart = alt.Chart(unique_hospitals_yr).mark_bar().encode(
  x = alt.X("year:O", title = "Year"),
  y = alt.Y("unique_hospitals:Q", title = "Number of Unique
      Hopsitals")).properties(title = "Number of Unique Hospitals Over the Years")
unique_hospitals_chart
```



b. Comparing the two graphs, I am seeing that the data is pretty consistent over the years– that there is an increase over the years. There is long-term stability of hospitals, with a slight increase from year to year, so there are more unique hospitals (new or mergers) but less hospitals than the year total.

Section 2: Identify hospital closures in POS file (15 pts) (*)
Partner 2
Q1

```
# Loading Libraries
import pandas as pd
import os
import csv
import warnings
warnings.filterwarnings("ignore")

# Q1
# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16)
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17)
health_data_17
short_term_17 =  health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017
short_term_17

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, encoding='latin1')
health_data_18.columns
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018
short_term_18

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, encoding='latin1')
health_data_19.columns
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019
short_term_19

# Concatanating into one dataframe, selecting required variables, and converting ZIP_CD to
        string
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
```

```
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].astype(str)

# Creating dataframe for active hospitals in 2016
certified_2016 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2016)]

# Creating dataframe for active hospitals in 2017
certified_2017 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2017)]

# Creating dataframe for active hospitals in 2018
certified_2018 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2018)]

# Creating dataframe for active hospitals in 2019
certified_2019 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2019)]

# Finding out which hospitals were closed by 2019 that were active in 2016, by PRVDR_NUM
hospitals_closed = pd.merge(certified_2016, certified_2019, on='PRVDR_NUM', how='left',
        indicator=True)
hospitals_closed = hospitals_closed[hospitals_closed['_merge'] == 'left_only'].drop(columns=
        ['_merge'])

# Filtering combined_df_final for entries of hospitals in hospitals_closed (by
        PGM_TRMNTN_CD), when status became not active, and the year this change happened
filtered_hospitals_closed = combined_df_final[(combined_df_final['PGM_TRMNTN_CD'] != 00) &

        (combined_df_final['PRVDR_NUM'].isin(hospitals_closed['PRVDR_NUM']))]

# Grouping filtered_hospitals_closed by name of hospital and summarizing year of
        termination/disappearance
last_active_years = (filtered_hospitals_closed.groupby('FAC_NAME')
                     .agg(year_terminated_disappear =('year', 'min'),
                          zip=('ZIP_CD', lambda x: x.unique()[0]))
                     .reset_index())

# Printing first 10
print(last_active_years.head(10))
print("The number of hospitals active in 2016 that are suspected to have closed by 2019 is",
        len(last_active_years["FAC_NAME"].unique()))
```

```
                                      FAC_NAME   year_terminated_disappear  \
0  (CLOSED) HEALTHSOUTH CHATTANOOGA REHAB HOSPITAL                       2019
1                          ABRAZO MARYVALE CAMPUS                       2017
2       ADVENTIST MEDICAL CENTER - CENTRAL VALLEY                       2017
3                          AFFINITY MEDICAL CENTER                       2018
4   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS                       2017
5        ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE                       2017
6                          ALLIANCE LAIRD HOSPITAL                       2019
7                         ALLIANCEHEALTH DEACONESS                       2019
8          ARKANSAS VALLEY REGIONAL MEDICAL CENTER                       2017
9              ASCENSION NE WISCONSIN MERCY CAMPUS                       2018

       zip
0  37404.0
```

```
1  85031.0
2  93230.0
3  44646.0
4  12208.0
5  75662.0
6  39365.0
7  73112.0
8  81050.0
9  54904.0
The number of hospitals active in 2016 that are suspected to have closed by 2019 is 177
```

Q2

```
# Loading Libraries
import pandas as pd
import os
import csv
import warnings
warnings.filterwarnings("ignore")

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16)
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016



base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17)
short_term_17 =  health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into one dataframe, selecting required variables, and converting ZIP_CD to
        string
```

```
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].astype(str)

# Creating dataframe for active hospitals in 2016
certified_2016 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2016)]

# Creating dataframe for active hospitals in 2017
certified_2017 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2017)]

# Creating dataframe for active hospitals in 2018
certified_2018 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2018)]

# Creating dataframe for active hospitals in 2019
certified_2019 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2019)]

# Finding out which hospitals were closed by 2019 that were active in 2016, by PRVDR_NUM
hospitals_closed = pd.merge(certified_2016, certified_2019, on='PRVDR_NUM', how='left',
        indicator=True)
hospitals_closed = hospitals_closed[hospitals_closed['_merge'] == 'left_only'].drop(columns=
        ['_merge'])

# Filtering combined_df_final for entries of hospitals in hospitals_closed (by
        PGM_TRMNTN_CD), when status became not active, and the year this change happened
filtered_hospitals_closed = combined_df_final[(combined_df_final['PGM_TRMNTN_CD'] != 00) &

        (combined_df_final['PRVDR_NUM'].isin(hospitals_closed['PRVDR_NUM']))]
```

```
# Grouping filtered_hospitals_closed by name of hospital and summarizing year of
        termination/disappearance
last_active_years = (filtered_hospitals_closed.groupby('FAC_NAME')
                     .agg(year_terminated_disappear =('year', 'min'),
                          zip=('ZIP_CD', lambda x: x.unique()[0]))
                     .reset_index())

# Sorting by hospital name and printing first 10 hospitals
last_active_years.sort_values(by='FAC_NAME', ascending=True)
last_active_years_1 = last_active_years[["FAC_NAME", "year_terminated_disappear"]]
first_10 = last_active_years.head(10)
print(first_10)
```

```
                                      FAC_NAME   year_terminated_disappear  \
0  (CLOSED) HEALTHSOUTH CHATTANOOGA REHAB HOSPITAL                       2019
1                          ABRAZO MARYVALE CAMPUS                       2017
2       ADVENTIST MEDICAL CENTER - CENTRAL VALLEY                       2017
3                          AFFINITY MEDICAL CENTER                      2018
4   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS                       2017
5        ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE                       2017
6                          ALLIANCE LAIRD HOSPITAL                      2019
7                        ALLIANCEHEALTH DEACONESS                       2019
8         ARKANSAS VALLEY REGIONAL MEDICAL CENTER                       2017
```

```
9              ASCENSION NE WISCONSIN MERCY CAMPUS                        2018
```

```
       zip
0  37404.0
1  85031.0
2  93230.0
3  44646.0
4  12208.0
5  75662.0
6  39365.0
7  73112.0
8  81050.0
9  54904.0
```

Q3a

```
# Loading Libraries
import pandas as pd
import os
import csv
import warnings
warnings.filterwarnings("ignore")

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
         4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16)


short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
(health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
         4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17)
short_term_17 =  health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
         (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
         4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
         (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
         4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
         (health_data_19["PRVDR_CTGRY_CD"] == 1)]
```

```
short_term_19["year"] = 2019

# Concatanating into one dataframe, selecting required variables, and converting ZIP_CD to
        string
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].astype(str)

# Creating dataframe for active hospitals in 2016
certified_2016 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2016)]

# Creating dataframe for active hospitals in 2017
certified_2017 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2017)]

# Creating dataframe for active hospitals in 2018
certified_2018 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2018)]

# Creating dataframe for active hospitals in 2019
certified_2019 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2019)]

# Finding out which hospitals were closed by 2019 that were active in 2016, by PRVDR_NUM
hospitals_closed = pd.merge(certified_2016, certified_2019, on='PRVDR_NUM', how='left',
        indicator=True)
hospitals_closed = hospitals_closed[hospitals_closed['_merge'] == 'left_only'].drop(columns=
        ['_merge'])

# Filtering combined_df_final for entries of hospitals in hospitals_closed (by
        PGM_TRMNTN_CD), when status became not active, and the year this change happened
filtered_hospitals_closed = combined_df_final[(combined_df_final['PGM_TRMNTN_CD'] != 00) &

        (combined_df_final['PRVDR_NUM'].isin(hospitals_closed['PRVDR_NUM']))]

# Grouping filtered_hospitals_closed by name of hospital and summarizing year of
        termination/disappearance
last_active_years = (filtered_hospitals_closed.groupby('FAC_NAME')
                     .agg(year_terminated_disappear =('year', 'min'),
                          zip=('ZIP_CD', lambda x: x.unique()[0]))
                     .reset_index())

# Grouping by ZIP_CD and year, and summarizing number of active hospitals by filtering for
        ['PGM_TRMNTN_CD'] == 00
active_hospitals_per_year = (combined_df_final[combined_df_final['PGM_TRMNTN_CD'] == 00]
                             .groupby(['ZIP_CD',
        'year']).size().reset_index(name='active_count')).reset_index()

active_hospitals_per_year =
        active_hospitals_per_year[active_hospitals_per_year["ZIP_CD"].isin(hospitals_closed['ZIP
        _CD_x'])]

# Created pivot table with columns for ZIP_CD, 2016, 2017, 2018, 2019, each summarizing
        number of active hospitals
pivoted_df = active_hospitals_per_year.pivot(index='ZIP_CD', columns='year',
        values='active_count').reset_index()
```

```
# To view entire dataframes
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

# Filling 0 for NAs
pivoted_df = pivoted_df.fillna(0)

# Dataframe shows zipcodes that saw either an increase or steady number
increased_hospitals = pivoted_df[(pivoted_df[2017] >= pivoted_df[2016]) |
                                 (pivoted_df[2018] >= pivoted_df[2017]) |
                                 (pivoted_df[2019] >= pivoted_df[2018])]

# List of unique ZIP_CD in above dataframe
no_decrease_zips = increased_hospitals['ZIP_CD'].unique()

# Merges no_decrease_zips with last_active_years to filter out (exclude) zipcodes that saw
        increase or no change during year of termination/disappearance/closure (final df is
        filtered_decreases_zips_df)
merged_df = last_active_years.merge(pivoted_df, how='left', left_on='zip',
        right_on='ZIP_CD')
filtered_decreases_zips_df = merged_df[~((merged_df['year_terminated_disappear'] == 2016) &
        (merged_df[2017] >= merged_df[2016]) |
                         (merged_df['year_terminated_disappear'] == 2017) &
        (merged_df[2018] >= merged_df[2017]) |
                         (merged_df['year_terminated_disappear'] == 2018) &
        (merged_df[2019] >= merged_df[2018]))]

# Filtering master dataframe (combined_df_final) for only those hospitals that are in
        filtered_decreases_zips_df
merg_aq =
        combined_df_final[(combined_df_final['FAC_NAME'].isin(filtered_decreases_zips_df['FAC_NAM
        E’])

# Dataframe with total number of hospitals in the corrected list of hospitals
provider_count = merg_aq.groupby('FAC_NAME') \
                        .agg(NUM_CMS=('PRVDR_NUM', 'count')) \
                        .reset_index()

# Dataframe with total number of hospitals that could be mergers of acquisitions
provider_count_1 = provider_count[(provider_count['NUM_CMS'] > 1)]

print("The number of hospitals that went through mergers or aquisitions is (Answer to 3a)",
        len(provider_count_1))

# Used BingChat with the following query "how do I create a pivot table that has a column
        for ZIP_CD and each year?"
# Used BingChat with the following query "how do I check that the increase/no change in
        active hospitals coincides with year of termination in that zipcode"
```

```
The number of hospitals that went through mergers or aquisitions is (Answer to 3a) 74
```

Q3b

```
# Loading Libraries
import pandas as pd
import os
import csv
import warnings
warnings.filterwarnings("ignore")

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16)
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17)
short_term_17 =  health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018
health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into one dataframe, selecting required variables, and converting ZIP_CD to
        string
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].astype(str)

# Creating dataframe for active hospitals in 2016
certified_2016 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2016)]

# Creating dataframe for active hospitals in 2017
certified_2017 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2017)]
```

```
# Creating dataframe for active hospitals in 2018
certified_2018 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2018)]

# Creating dataframe for active hospitals in 2019
certified_2019 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2019)]

# Finding out which hospitals were closed by 2019 that were active in 2016, by PRVDR_NUM
hospitals_closed = pd.merge(certified_2016, certified_2019, on='PRVDR_NUM', how='left',
        indicator=True)
hospitals_closed = hospitals_closed[hospitals_closed['_merge'] == 'left_only'].drop(columns=
        ['_merge'])

# Filtering combined_df_final for entries of hospitals in hospitals_closed (by
        PGM_TRMNTN_CD), when status became not active, and the year this change happened
filtered_hospitals_closed = combined_df_final[(combined_df_final['PGM_TRMNTN_CD'] != 00) &

        (combined_df_final['PRVDR_NUM'].isin(hospitals_closed['PRVDR_NUM']))]

# Grouping filtered_hospitals_closed by name of hospital and summarizing year of
        termination/disappearance
last_active_years = (filtered_hospitals_closed.groupby('FAC_NAME')
                     .agg(year_terminated_disappear =('year', 'min'),
                          zip=('ZIP_CD', lambda x: x.unique()[0]))
                     .reset_index())

# Grouping by ZIP_CD and year, and summarizing number of active hospitals by filtering for
        ['PGM_TRMNTN_CD'] == 00
active_hospitals_per_year = (combined_df_final[combined_df_final['PGM_TRMNTN_CD'] == 00]
                             .groupby(['ZIP_CD',
        'year']).size().reset_index(name='active_count')).reset_index()

active_hospitals_per_year =
        active_hospitals_per_year[active_hospitals_per_year["ZIP_CD"].isin(hospitals_closed['ZIP
        _CD_x'])]

# Created pivot table with columns for ZIP_CD, 2016, 2017, 2018, 2019, each summarizing
        number of active hospitals
pivoted_df = active_hospitals_per_year.pivot(index='ZIP_CD', columns='year',
        values='active_count').reset_index()

# To view entire dataframes
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

# Filling 0 for NAs
pivoted_df = pivoted_df.fillna(0)

# Dataframe shows zipcodes that saw either an increase or steady number
increased_hospitals = pivoted_df[(pivoted_df[2017] >= pivoted_df[2016]) |
                                 (pivoted_df[2018] >= pivoted_df[2017]) |
                                 (pivoted_df[2019] >= pivoted_df[2018])]

# List of unique ZIP_CD in above dataframe
no_decrease_zips = increased_hospitals['ZIP_CD'].unique()
```

```
# Merges no_decrease_zips with last_active_years to filter out (exclude) zipcodes that saw
        increase or no change during year of termination/disappearance/closure (final df is
        filtered_decreases_zips_df)
merged_df = last_active_years.merge(pivoted_df, how='left', left_on='zip',
        right_on='ZIP_CD')
filtered_decreases_zips_df = merged_df[~((merged_df['year_terminated_disappear'] == 2016) &
        (merged_df[2017] >= merged_df[2016]) |
                        (merged_df['year_terminated_disappear'] == 2017) &
        (merged_df[2018] >= merged_df[2017]) |
                        (merged_df['year_terminated_disappear'] == 2018) &
        (merged_df[2019] >= merged_df[2018]))]

# Filtering master dataframe (combined_df_final) for only those hospitals that are in
        filtered_decreases_zips_df
merg_aq =
        combined_df_final[(combined_df_final['FAC_NAME'].isin(filtered_decreases_zips_df['FAC_NAM
        E’])
# Dataframe with total number of hospitals in the corrected list of hospitals
provider_count = merg_aq.groupby('FAC_NAME') \
                        .agg(NUM_CMS=('PRVDR_NUM', 'count')) \
                        .reset_index()

# Dataframe with total number of hospitals that could be mergers of acquisitions
provider_count_1 = provider_count[(provider_count['NUM_CMS'] > 1)]

### b
print("The number of hospitals in the corrected list is (Answer to 3b)",
        len(provider_count))
```

```
The number of hospitals in the corrected list is (Answer to 3b) 81
```

Q3c

```
# Loading Libraries
import pandas as pd
import os
import csv
import warnings
warnings.filterwarnings("ignore")

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16)
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
```

```
health_data_17 = pd.read_csv(health_path_17)
short_term_17 =  health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into one dataframe, selecting required variables, and converting ZIP_CD to
        string
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].astype(str)

# Creating dataframe for active hospitals in 2016
certified_2016 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2016)]

# Creating dataframe for active hospitals in 2017
certified_2017 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2017)]
# Creating dataframe for active hospitals in 2018
certified_2018 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2018)]

# Creating dataframe for active hospitals in 2019
certified_2019 = combined_df_final[(combined_df_final["PGM_TRMNTN_CD"] == 00) &
        (combined_df_final["year"] == 2019)]

# Finding out which hospitals were closed by 2019 that were active in 2016, by PRVDR_NUM
hospitals_closed = pd.merge(certified_2016, certified_2019, on='PRVDR_NUM', how='left',
        indicator=True)
hospitals_closed = hospitals_closed[hospitals_closed['_merge'] == 'left_only'].drop(columns=
        ['_merge'])

# Filtering combined_df_final for entries of hospitals in hospitals_closed (by
        PGM_TRMNTN_CD), when status became not active, and the year this change happened
filtered_hospitals_closed = combined_df_final[(combined_df_final['PGM_TRMNTN_CD'] != 00) &

        (combined_df_final['PRVDR_NUM'].isin(hospitals_closed['PRVDR_NUM']))]

# Grouping filtered_hospitals_closed by name of hospital and summarizing year of
        termination/disappearance
last_active_years = (filtered_hospitals_closed.groupby('FAC_NAME')
                     .agg(year_terminated_disappear =('year', 'min'),
                          zip=('ZIP_CD', lambda x: x.unique()[0]))
                     .reset_index())
```

```
# Grouping by ZIP_CD and year, and summarizing number of active hospitals by filtering for
        ['PGM_TRMNTN_CD'] == 00
active_hospitals_per_year = (combined_df_final[combined_df_final['PGM_TRMNTN_CD'] == 00]
                             .groupby(['ZIP_CD',
        'year']).size().reset_index(name='active_count')).reset_index()

active_hospitals_per_year =
        active_hospitals_per_year[active_hospitals_per_year["ZIP_CD"].isin(hospitals_closed['ZIP
        _CD_x'])]

# Created pivot table with columns for ZIP_CD, 2016, 2017, 2018, 2019, each summarizing
        number of active hospitals
pivoted_df = active_hospitals_per_year.pivot(index='ZIP_CD', columns='year',
        values='active_count').reset_index()

# To view entire dataframes
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

# Filling 0 for NAs
pivoted_df = pivoted_df.fillna(0)

# Dataframe shows zipcodes that saw either an increase or steady number
increased_hospitals = pivoted_df[(pivoted_df[2017] >= pivoted_df[2016]) |
                                 (pivoted_df[2018] >= pivoted_df[2017]) |
                                 (pivoted_df[2019] >= pivoted_df[2018])]

# List of unique ZIP_CD in above dataframe
no_decrease_zips = increased_hospitals['ZIP_CD'].unique()
```

```
# Merges no_decrease_zips with last_active_years to filter out (exclude) zipcodes that saw
        increase or no change during year of termination/disappearance/closure (final df is
        filtered_decreases_zips_df)
merged_df = last_active_years.merge(pivoted_df, how='left', left_on='zip',
        right_on='ZIP_CD')
filtered_decreases_zips_df = merged_df[~((merged_df['year_terminated_disappear'] == 2016) &
        (merged_df[2017] >= merged_df[2016]) |
                        (merged_df['year_terminated_disappear'] == 2017) &
        (merged_df[2018] >= merged_df[2017]) |
                        (merged_df['year_terminated_disappear'] == 2018) &
        (merged_df[2019] >= merged_df[2018]))]

# Filtering master dataframe (combined_df_final) for only those hospitals that are in
        filtered_decreases_zips_df
merg_aq =
          combined_df_final[(combined_df_final['FAC_NAME'].isin(filtered_decreases_zips_df['FAC_
          NAME’])

# Dataframe with total number of hospitals in the corrected list of hospitals
provider_count = merg_aq.groupby('FAC_NAME') \
                        .agg(NUM_CMS=('PRVDR_NUM', 'count')) \
                        .reset_index()

# Dataframe with total number of hospitals that could be mergers of acquisitions
provider_count_1 = provider_count[(provider_count['NUM_CMS'] > 1)]

### c
provider_count.sort_values(by='FAC_NAME', ascending=True)
print("The answer to 3c is:")
print(provider_count.head(10))

# Used BingChat with the following query "how do I create a pivot table that has a column
        for ZIP_CD and each year?"
# Used BingChat with the following query "how do I check that the increase/no change in
        active hospitals coincides with year of termination in that zipcode"
```

```
The answer to 3c is:
                                          FAC_NAME  NUM_CMS
0     (CLOSED) HEALTHSOUTH CHATTANOOGA REHAB HOSPITAL        1
1                           ALLIANCE LAIRD HOSPITAL        4
2                         ALLIANCEHEALTH DEACONESS         4
3                       ATRIUM HEALTH KINGS MOUNTAIN         1
4                      BARIX CLINICS OF PENNSYLVANIA         4
5                    BAYLOR EMERGENCY MEDICAL CENTER        10
6  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER AT C        4
7                          BELMONT COMMUNITY HOSPITAL        4
8                              BIG SKY MEDICAL CENTER        4
9               BLACK RIVER COMMUNITY MEDICAL CENTER         4
```

Section 3: Download Census zip code shapefile (10 pt) Partner 1

1.

```
import zipfile

zip_file_path = "/Users/charismalambert/Downloads/gz_2010_us_860_00_500k.zip"
extraction_path = "extracted_files"

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extraction_path)

files = os.listdir(extraction_path)

file_info = []

for file in files:
    file_path = os.path.join(extraction_path, file)
    file_size = os.path.getsize(file_path)
    file_type = os.path.splitext(file)[1]

    file_info.append({
```

```
        "file_name": file,
        "file_type": file_type,
        "file_size_kb": file_size / 1024
    })

print("Answer to part b:")
for info in file_info:
    print(f" File: {info['file_name']}, Type: {info['file_type']}, Size:
        {info['file_size_kb']:.2f} KB")

# Citation:  Ran ChatGPT query on how to extract files from a zip file using Python and
        these are the steps it gave me.
```

```
Answer to part b:
 File: gz_2010_us_860_00_500k.prj, Type: .prj, Size: 0.16 KB
 File: gz_2010_us_860_00_500k.shx, Type: .shx, Size: 258.85 KB
 File: gz_2010_us_860_00_500k.shp, Type: .shp, Size: 817914.63 KB
 File: gz_2010_us_860_00_500k.dbf, Type: .dbf, Size: 6274.88 KB
 File: gz_2010_us_860_00_500k.xml, Type: .xml, Size: 15.27 KB
```

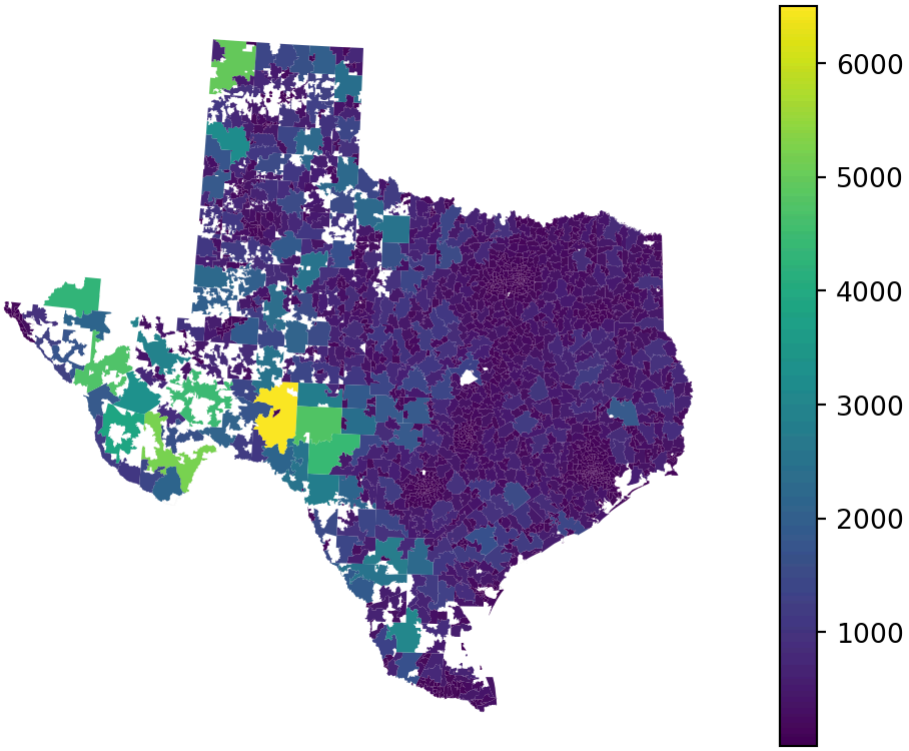
a. The five file types are:
.xml: Metadata file describing the dataset.
.shx: An index file that provides quick access to the shapes within the .shp file.
.shp: Contains geographic shapes representing spatial data.
.prj: Contains coordinate system and projection information.
.dbf: Stores tabular attribute data associated with each spatial feature inn the .shp file.

b. File: gz_2010_us_860_00_500k.prj, Type: .prj, Size: 0.16 KB
File: gz_2010_us_860_00_500k.shx, Type: .shx, Size: 258.85 KB
File: gz_2010_us_860_00_500k.shp, Type: .shp, Size: 817914.63 KB
File: gz_2010_us_860_00_500k.dbf, Type: .dbf, Size: 6274.88 KB
File: gz_2010_us_860_00_500k.xml, Type: .xml, Size: 15.27 KB

2.

```
# load zipcode shapefile
import geopandas as gdp

filepath = "/Users/charismalambert/Downloads/gz_2010_us_860_00_500k"
census_shp = gdp.read_file(filepath)

# restrict to Texas zip codes
census_shp["ZCTA5"] = census_shp["ZCTA5"].astype(str)
texas_zip = census_shp[census_shp["ZCTA5"].str.startswith(("75", "76", "77", "78",
        "79"))]

short_term_16["ZIP_CD"] = short_term_16["ZIP_CD"].astype(str)
hospitals_by_zip = short_term_16["ZIP_CD"].value_counts().reset_index()
hospitals_by_zip.columns = ["zip_code", "total_hospitals"]
hospitals_by_zipTX = texas_zip.merge(hospitals_by_zip, left_on = "ZCTA5", right_on =
        "zip_code", how = "left")
```

```
# choropleth of hospitals by zp code in Texas
hospitals_by_zipTX = hospitals_by_zipTX.to_crs("EPSG:5070")
hospitals_by_zipTX["area_km2"] = hospitals_by_zipTX.area/1000000
hospitals_by_zipTX.plot(column = "area_km2", legend = True).set_axis_off()

#Citation: Ran ChatGPT query on .prj file to get the .to_crs conversion.
```



Section 4: Calculate zip code's distance to the nearest hospital (20 pts) (*) Partner 2

Q1

```
# Loading Libraries
import geopandas as gpd
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
            gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500
            k.shp")
 print(zips_all_centroids.info())
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 33120 entries, 0 to 33119
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   GEO_ID      33120 non-null  object
 1   ZCTA5       33120 non-null  object
 2   NAME        33120 non-null  object
 3   LSAD        33120 non-null  object
 4   CENSUSAREA  33120 non-null  float64
 5   geometry    33120 non-null geometry
dtypes: float64(1), geometry(1), object(4)
memory usage: 1.5+ MB
None
```

Dimensions of the GeoDataFrame: 33120 rows, 6 columns (variables)

Each column and what they capture:

1. GEO_ID: geographic identifier that uniquely identifies an area
2. ZCTA5: Zip Code (first 5 digits with leading 0)
3. NAME: the same as ZCTA5
4. LSAD: the way in which each area is named (all are ZCTA5)
5. CENSUSAREA: the area of region
6. geometry: the shape of region

Q2

```
# Loading Libraries
import geopandas as gpd
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
         gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.
         shp")

# Filter Texas zip codes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
         & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
         == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
         (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

print("Number of unique ZipCodes for Texas = ", len(zips_texas_centroids.ZCTA5.unique()))

# Filter Texas and bordering states zipcodes
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
         "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
         (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
         == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0885") |
         (zips_all_centroids['GEO_ID'] == "8600000US0870") | (zips_all_centroids['GEO_ID']
         == "8600000US0871") | (zips_all_centroids['GEO_ID'] >= "8600000US0873") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0875") | (zips_all_centroids['GEO_ID']
         >= "8600000US0877") & (zips_all_centroids['GEO_ID'] <= "8600000US0885") |
         (zips_all_centroids['GEO_ID'] == "8600000US0730") | (zips_all_centroids['GEO_ID']
         == "8600000US0731") | (zips_all_centroids['GEO_ID'] >= "8600000US0734") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0741") | (zips_all_centroids['GEO_ID']
         >= "8600000US0743") & (zips_all_centroids['GEO_ID'] <= "8600000US0749") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0716") & (zips_all_centroids['GEO_ID']
         <= "8600000US0729") | (zips_all_centroids['GEO_ID'] >= "8600000US0700") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0701") | (zips_all_centroids['GEO_ID']
         >= "8600000US0703") & (zips_all_centroids['GEO_ID'] <= "8600000US0708") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0710") & (zips_all_centroids['GEO_ID']
         <= "8600000US0714") ]

print("Number of unique ZipCodes for Texas and Neighboring states = ",
         len(zips_texas_borderstates_centroids.ZCTA5.unique()))
```

```
Number of unique ZipCodes for Texas =  164
Number of unique ZipCodes for Texas and Neighboring states =  303
```

EXTRA CREDIT - Section 4, Q2 with function attempt

```
# Loading Libraries
import geopandas as gpd
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
           gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500
           k.shp")
```

```
# Filter Texas zip codes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
        & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
        == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
        (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

print("Number of unique ZipCodes for Texas and Neighboring states = ",
        len(zips_texas_centroids.ZCTA5.unique()))

# Texas and bordering states
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
        "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
        (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
        == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
        (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
        >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
        <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
        (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
        >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
        <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
        >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
        <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

print("Number of unique ZipCodes for Texas and Neighboring states = ",
        len(zips_texas_borderstates_centroids.ZCTA5.unique()))

from shapely.geometry import Polygon
# Function to check if two polygons intersect
def polygons_intersect(poly1, poly2):
    return poly1.intersects(poly2)

# Creating Texas zipcodes dataframe
texas_combined = zips_texas_centroids.unary_union

# Creating Texas prefixes dataframe
texas_prefixes = zips_texas_centroids.GEO_ID

# Creating Texas and Bordering zipcodes dataframe
bordering_prefixes = zips_texas_borderstates_centroids.GEO_ID

# Creating Bordering zipcodes dataframe
bordering_states_only =
        zips_texas_borderstates_centroids[~zips_texas_borderstates_centroids['GEO_ID'].isin(t

# Creating Texas and Bordering zipcodes dataframe
bordering_states_only['intersects_texas'] = bordering_states_only['geometry'].apply(
    lambda x: polygons_intersect(texas_combined, x)
)

# Filter for those that intersect
intersecting_zips = bordering_states_only[bordering_states_only['intersects_texas']]
```

```
print(intersecting_zips.head(5))

print(f"Intersecting zip codes: {intersecting_zips.GEO_ID.nunique()}")

# Used BingChat to aid in creating the function
```

```
Number of unique ZipCodes for Texas and Neighboring states =  164
Number of unique ZipCodes for Texas and Neighboring states =  303
            GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
816  8600000US07004  07004  07004  ZCTA5      10.286
817  8600000US07006  07006  07006  ZCTA5       9.247
824  8600000US07039  07039  07039  ZCTA5      13.803
826  8600000US07047  07047  07047  ZCTA5       5.134
827  8600000US07054  07054  07054  ZCTA5      13.724

                                          geometry  intersects_texas
816  POLYGON ((-74.32854 40.84467, -74.32881 40.845...              True
817  POLYGON ((-74.29835 40.83042, -74.29926 40.829...              True
824  POLYGON ((-74.36728 40.76107, -74.36753 40.761...              True
826  POLYGON ((-74.04586 40.75737, -74.04892 40.758...              True
827  POLYGON ((-74.38396 40.82499, -74.384 40.825, ...              True
Intersecting zip codes: 38
```

Q3

```
# Loading Libraries
import pandas as pd
import os
import csv
import geopandas as gpd
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
            gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500
            k.shp")
# Filter Texas Border States zip codes



zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
          "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
          (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
          == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
          (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
          >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
          (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
          <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
          (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
          >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
          (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
          <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
```

```
        (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
        >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
        <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

# Filter Texas zip codes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
        & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
        == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
        (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16, dtype={'ZIP_CD': str})
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17, dtype={'ZIP_CD': str})
short_term_17 = health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into combined_df_final and selecting required variables
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]

# Adding leading 0's in combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str.zfill(6)

# Dropping last digit from combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str[:-1]
```

```
#print(combined_df_final[['ZIP_CD', 'PGM_TRMNTN_CD', 'year']])

# Creating dataframe that has only the zip codes in texas and bordering hospitals that each
        have at least one hospital
zips_tb_with_hosp = combined_df_final[
    combined_df_final['ZIP_CD'].isin(zips_texas_borderstates_centroids['ZCTA5']) &
    (combined_df_final["year"] == 2016)]

zips_tb_with_hosp = zips_tb_with_hosp.groupby('ZIP_CD').agg(number_of_hospitals =
        ('PRVDR_NUM', 'count')).reset_index()
zips_tb_with_hosp = zips_tb_with_hosp[zips_tb_with_hosp['number_of_hospitals'] >= 1]

zips_tb_with_hosp_cent =
        zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zips
        _tb_with_hosp['ZIP_CD'])]
# Creating ZCTA5 in combined_df_final, which has all hospital data
combined_df_final['ZCTA5'] = combined_df_final['ZIP_CD']
combined_df_final_1 = combined_df_final

# Used an inner merge on ZCTA5
zips_withhospital_centroids = pd.merge(combined_df_final_1, zips_tb_with_hosp_cent,
        how='inner', on='ZCTA5')
print(zips_withhospital_centroids.head(5))
```

```
                      FAC_NAME  PRVDR_CTGRY_CD  PRVDR_CTGRY_SBTYP_CD  \
0           DELTA MEDICAL CENTER               1                   1.0
1  CONWAY REGIONAL MEDICAL CENTER              1                   1.0
2                   OZARK HEALTH               1                   1.0
3           DEWITT CITY HOSPITAL               1                   1.0
4        SALINE MEMORIAL HOSPITAL              1                   1.0

  PRVDR_NUM ZIP_CD  PGM_TRMNTN_CD  year  ZCTA5          GEO_ID   NAME   LSAD  \
0    040009  07202              1  2016  07202  8600000US07202  07202  ZCTA5
1    040029  07203              0  2016  07203  8600000US07203  07203  ZCTA5
2    040060  07203              7  2016  07203  8600000US07203  07203  ZCTA5
3    040064  07204              7  2016  07204  8600000US07204  07204  ZCTA5
4    040084  07201              0  2016  07201  8600000US07201  07201  ZCTA5

   CENSUSAREA                                           geometry
0       2.319  POLYGON ((-74.23332 40.65446, -74.23423 40.655...
1       2.651  POLYGON ((-74.24665 40.66319, -74.24532 40.663...
2       2.651  POLYGON ((-74.24665 40.66319, -74.24532 40.663...
3       1.232  POLYGON ((-74.24665 40.66319, -74.24735 40.663...
```

```
4        6.645  POLYGON ((-74.15304 40.6548, -74.16428 40.6550...
```

I did an inner merge, and merged on the variable 'ZCTA5', for zipcode.

Q4a

```
# Loading Libraries
import pandas as pd
import os
import csv
import geopandas as gpd
from shapely.geometry import Point, Polygon
from shapely.ops import nearest_points
import time
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
         gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.
         shp")

# Filter Texas Border States zip codes
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
         "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
         (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
         == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
         (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
         >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
         <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
         (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
         >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
         <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
         >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
         <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

# Filter Texas zip codes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
         & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
         == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
         (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
         4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16, dtype={'ZIP_CD': str})
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
         (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016
```

```
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17, dtype={'ZIP_CD': str})
short_term_17 = health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into combined_df_final and selecting required variables
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]

# Adding leading 0's to combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str.zfill(6)

# Dropping last digit from combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str[:-1]

# Creating dataframe that has only the zip codes in Texas and Bordering hospitals that each
        have at least one hospital
zips_tb_with_hosp = combined_df_final[
    combined_df_final['ZIP_CD'].isin(zips_texas_borderstates_centroids['ZCTA5']) &
    (combined_df_final["year"] == 2016)]

zips_tb_with_hosp = zips_tb_with_hosp.groupby('ZIP_CD').agg(number_of_hospitals =
        ('PRVDR_NUM', 'count')).reset_index()
zips_tb_with_hosp = zips_tb_with_hosp[zips_tb_with_hosp['number_of_hospitals'] >= 1]
zips_tb_with_hosp
zips_tb_with_hosp_cent =
        zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zips
        _tb_with_hosp['ZIP_CD'])]

# Creating ZCTA5 in combined_df_final, which has all hospital data
combined_df_final['ZCTA5'] = combined_df_final['ZIP_CD']
combined_df_final_1 = combined_df_final

## Used an inner merge on ZCTA5
zips_withhospital_centroids = pd.merge(combined_df_final_1, zips_tb_with_hosp_cent,
        how='inner', on='ZCTA5')
zips_withhospital_centroids
```

```
# Making sure all are geodataframes
if not isinstance(zips_withhospital_centroids, gpd.GeoDataFrame):
    zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
         geometry='geometry')

if 'geometry' not in zips_withhospital_centroids.columns:
    zips_withhospital_centroids['geometry'] = zips_withhospital_centroids.apply(
        lambda row: Point(row['longitude'], row['latitude']), axis=1)

# Ensure they use the same CRS
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=4326)
zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
         geometry='geometry')
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)

# Subset to 10 ZIP codes for testing
subset = zips_texas_centroids.head(10)
subset

# Calculating time for the join with the subset
def calculate_nearest(row, other_gdf, geom_col='geometry', src_col='ZCTA5'):
    if other_gdf.empty or row[geom_col] is None:
        return None, float('inf')
    other_geom_union = other_gdf.geometry.unary_union
    nearest_geom = nearest_points(row[geom_col], other_geom_union)[1]
    nearest_point = other_gdf.loc[other_gdf.geometry == nearest_geom]
    if nearest_point.empty:
        return None, float('inf')
    nearest_zip = nearest_point[src_col].values[0]
    distance = row[geom_col].distance(nearest_geom)
    return nearest_zip, distance

start_time = time.time()

subset['nearest_zip'], subset['distance_to_nearest'] = zip(
    *subset.apply(calculate_nearest, other_gdf=zips_withhospital_centroids, axis=1)
)

end_time = time.time()
time_taken = end_time - start_time

print(subset[['ZCTA5', 'nearest_zip', 'distance_to_nearest']])
print(f"Time taken for subset of 10 ZIP codes: {time_taken} seconds")

total_zip_codes = len(zips_texas_centroids)
estimated_time = (time_taken / 10) * total_zip_codes
print(f"Estimated time for the entire dataset: {estimated_time} seconds")

# Used BingChat with the following query: "how do I create a function that calculates the
         calculate the distance to the nearest zip code with at least one hospital?"
# Used BingChat with the following query: "how do I measure the time for a spatial join?"
```

```
     ZCTA5 nearest_zip distance_to_nearest
933  07501        None                 inf
934  07601        None                 inf
935  07624        None                 inf
936  07627        None                 inf
937  07642        None                 inf
938  07646        None                 inf
939  07660        None                 inf
940  07663        None                 inf
941  07675        None                 inf
942  07677        None                 inf
Time taken for subset of 10 ZIP codes: 46.294291257858276 seconds
Estimated time for the entire dataset: 759.2263766288758 seconds
```

Q4b

```
# Loading Libraries
import pandas as pd
import os
import csv
import geopandas as gpd
from shapely.geometry import Point, Polygon
from shapely.ops import nearest_points
import time
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
         gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.
         shp")

# Filter Texas Border States zip codes
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
         "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
         (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
         == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
         (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
         >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
         <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
         (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
         >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
         <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
         >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
         <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

# Filter Texas zip codes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
         & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
         == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
         (zips_all_centroids['GEO_ID'] == "8600000US0885") ]
```

```
# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16, dtype={'ZIP_CD': str})
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17, dtype={'ZIP_CD': str})
short_term_17 = health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into a dataframe and selecting required variables
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]

# Adding leading 0's in combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str.zfill(6)

# Dropping last digit from combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str[:-1]

# Creating dataframe that has only the zip codes in texas and bordering hospitals that each
        have at least one hospital
zips_tb_with_hosp = combined_df_final[
    combined_df_final['ZIP_CD'].isin(zips_texas_borderstates_centroids['ZCTA5']) &
    (combined_df_final["year"] == 2016)]

zips_tb_with_hosp = zips_tb_with_hosp.groupby('ZIP_CD').agg(number_of_hospitals =
        ('PRVDR_NUM', 'count')).reset_index()
zips_tb_with_hosp = zips_tb_with_hosp[zips_tb_with_hosp['number_of_hospitals'] >= 1]

zips_tb_with_hosp_cent =
        zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zips
        _tb_with_hosp['ZIP_CD'])]
```

```
# Creating ZCTA5 in combined_df_final, which has all hospital data
combined_df_final['ZCTA5'] = combined_df_final['ZIP_CD']
combined_df_final_1 = combined_df_final

# Used an inner merge on ZCTA5
zips_withhospital_centroids = pd.merge(combined_df_final_1, zips_tb_with_hosp_cent,
         how='inner', on='ZCTA5')

# Making sure all are geodataframes
if not isinstance(zips_withhospital_centroids, gpd.GeoDataFrame):
    zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
         geometry='geometry')

if 'geometry' not in zips_withhospital_centroids.columns:
    zips_withhospital_centroids['geometry'] = zips_withhospital_centroids.apply(
        lambda row: Point(row['longitude'], row['latitude']), axis=1)

# Ensure they use the same CRS
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=4326)
zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
         geometry='geometry')
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)

# Subset to 10 ZIP codes for testing
subset = zips_texas_centroids.head(10)

# Calculating time for the join with the subset
def calculate_nearest(row, other_gdf, geom_col='geometry', src_col='ZCTA5'):
    if other_gdf.empty or row[geom_col] is None:
        return None, float('inf')
    other_geom_union = other_gdf.geometry.unary_union
    nearest_geom = nearest_points(row[geom_col], other_geom_union)[1]
    nearest_point = other_gdf.loc[other_gdf.geometry == nearest_geom]
    if nearest_point.empty:
        return None, float('inf')
    nearest_zip = nearest_point[src_col].values[0]
    distance = row[geom_col].distance(nearest_geom)
    return nearest_zip, distance

start_time = time.time()

subset['nearest_zip'], subset['distance_to_nearest'] = zip(
    *subset.apply(calculate_nearest, other_gdf=zips_withhospital_centroids, axis=1)
)

end_time = time.time()
time_taken = end_time - start_time

print(subset[['ZCTA5', 'nearest_zip', 'distance_to_nearest']])
print(f"Time taken for subset of 10 ZIP codes: {time_taken} seconds")

total_zip_codes = len(zips_texas_centroids)
estimated_time = (time_taken / 10) * total_zip_codes
print(f"Estimated time for the entire dataset: {estimated_time} seconds")
```

```
# Calculating time for join for all zipcodes
start_time = time.time()

zips_texas_centroids['nearest_zip'], zips_texas_centroids['distance_to_nearest'] = zip(
    *zips_texas_centroids.apply(calculate_nearest, other_gdf=zips_withhospital_centroids,
         axis=1)
)

end_time = time.time()
time_taken = end_time - start_time

print(zips_texas_centroids[['ZCTA5', 'nearest_zip', 'distance_to_nearest']])
print(f"Time taken for all ZIP codes: {time_taken} seconds")
```

```
     ZCTA5 nearest_zip distance_to_nearest
933  07501        None                inf
934  07601        None                inf
935  07624        None                inf
936  07627        None                inf
937  07642        None                inf
938  07646        None                inf
939  07660        None                inf
940  07663        None                inf
941  07675        None                inf
942  07677        None                inf
Time taken for subset of 10 ZIP codes: 46.1674587726593 seconds
Estimated time for the entire dataset: 757.1463238716125 seconds
       ZCTA5 nearest_zip  distance_to_nearest
933    07501        None                 inf
934    07601        None                 inf
935    07624        None                 inf
936    07627        None                 inf
937    07642        None                 inf
...      ...         ...                 ...
32968  07652        None                 inf
32969  07726        None                 inf
32976  07801        None                 inf
32977  07869        None                 inf
32978  07974        None                 inf

[164 rows x 3 columns]
Time taken for all ZIP codes: 712.890095949173 seconds
```

It about the same time (a little less) to join all the zips than the estimated time.

Q4c

```
# Loading Libraries
import pandas as pd
import os
import csv
import geopandas as gpd
```

```
from shapely.geometry import Point, Polygon
from shapely.ops import nearest_points
import time
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
        gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.
        shp")

# Filter Texas Border States zip codes
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
        "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
        (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
        == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
        (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
        >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
        <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
        (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
        >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
        <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
        >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
        <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

# Filter Texas zip codes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
        & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
        == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
        (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16, dtype={'ZIP_CD': str})
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17, dtype={'ZIP_CD': str})
short_term_17 = health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
```

```
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating data into combined_df_final and selecting required variables
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]

# Adding leading 0's to combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str.zfill(6)

# Dropping to last digit from combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str[:-1]

# Creating dataframe that has only the zip codes in texas and bordering hospitals that each
        have at least one hospital
zips_tb_with_hosp = combined_df_final[
    combined_df_final['ZIP_CD'].isin(zips_texas_borderstates_centroids['ZCTA5']) &
    (combined_df_final["year"] == 2016)]

zips_tb_with_hosp = zips_tb_with_hosp.groupby('ZIP_CD').agg(number_of_hospitals =
        ('PRVDR_NUM', 'count')).reset_index()
zips_tb_with_hosp = zips_tb_with_hosp[zips_tb_with_hosp['number_of_hospitals'] >= 1]

zips_tb_with_hosp_cent =
        zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zips
        _tb_with_hosp['ZIP_CD'])]
# Creating ZCTA5 in combined_df_final, which has all hospital data
combined_df_final['ZCTA5'] = combined_df_final['ZIP_CD']
combined_df_final_1 = combined_df_final

# Used an inner merge on ZCTA5
zips_withhospital_centroids = pd.merge(combined_df_final_1, zips_tb_with_hosp_cent,
        how='inner', on='ZCTA5')

# Making sure all are geodataframes
if not isinstance(zips_withhospital_centroids, gpd.GeoDataFrame):
    zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
        geometry='geometry')

if 'geometry' not in zips_withhospital_centroids.columns:
    zips_withhospital_centroids['geometry'] = zips_withhospital_centroids.apply(
        lambda row: Point(row['longitude'], row['latitude']), axis=1)

# Ensure they use the same CRS
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=4326)
zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
        geometry='geometry')
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)
```

```
# Q4c
# in degrees
distance = gpd.sjoin_nearest(zips_texas_centroids, zips_withhospital_centroids,
how = 'inner',
distance_col = 'distance')

# in miles and degrees
distance['distance_miles'] = distance['distance'] * 69
print(distance[['ZIP_CD', 'distance_miles', 'distance']].head(5))
print("Unique entries for distance_miles are",distance['distance_miles'].unique())

# Used BingChat with the following query: "how do I convert degrees in geopatial data to
         miles?"
```

```
    ZIP_CD  distance_miles  distance
933  07505             0.0       0.0
933  07505             0.0       0.0
933  07505             0.0       0.0
933  07505             0.0       0.0
933  07505             0.0       0.0
Unique entries for distance_miles are [0.         3.02911584 4.64296213 0.25841988 1.99839517]
```

The original unit of distance is degrees. I multiplied the distance given in degrees by 69 to convert to miles.

Q5a

```
# Loading Libraries
import pandas as pd
import os
import csv
import geopandas as gpd
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
         gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500.

shp")

# Filter Texas Border States zip codes
```

```
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
"8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
(zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
        == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
        (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
        >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
        <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
        (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
        >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
        <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
        >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
        <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

# Filter Texas zip codes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
        & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
        == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
        (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16, dtype={'ZIP_CD': str})
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17, dtype={'ZIP_CD': str})
short_term_17 = health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating data into combined_df_final and selecting required variables
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)
```

```
combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]

# Adding leading 0's in combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str.zfill(6)

# Dropping the last digit from combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str[:-1]

# Creating dataframe that has only the zip codes in Texas and Bordering hospitals that each
        have at least one hospital
zips_tb_with_hosp = combined_df_final[
    combined_df_final['ZIP_CD'].isin(zips_texas_borderstates_centroids['ZCTA5']) &
    (combined_df_final["year"] == 2016)]

zips_tb_with_hosp = zips_tb_with_hosp.groupby('ZIP_CD').agg(number_of_hospitals =
        ('PRVDR_NUM', 'count')).reset_index()
zips_tb_with_hosp = zips_tb_with_hosp[zips_tb_with_hosp['number_of_hospitals'] >= 1]

zips_tb_with_hosp_cent =
        zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zips
        _tb_with_hosp['ZIP_CD'])]

# Creating ZCTA5 in combined_df_final, which has all hospital data
combined_df_final['ZCTA5'] = combined_df_final['ZIP_CD']
combined_df_final_1 = combined_df_final

# Used an inner merge on ZCTA5
zips_withhospital_centroids = pd.merge(combined_df_final_1, zips_tb_with_hosp_cent,
        how='inner', on='ZCTA5')

# Making sure all are geodataframes
if not isinstance(zips_withhospital_centroids, gpd.GeoDataFrame):
    zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
        geometry='geometry')

if 'geometry' not in zips_withhospital_centroids.columns:
    zips_withhospital_centroids['geometry'] = zips_withhospital_centroids.apply(
        lambda row: Point(row['longitude'], row['latitude']), axis=1)

# Creating dataframe with all Texas Zipcodes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
        & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
        == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
        (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

# Ensure both GeoDataFrames use the same coordinate reference system (CRS)
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=4326)
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)

# Perform spatial join to find average distance to the nearest hospitals
joined = gpd.sjoin_nearest(zips_texas_centroids, zips_withhospital_centroids, how='inner',
        distance_col='distance')
joined_1 = joined.groupby('ZIP_CD').agg(average_distance_miles =('distance',
        'first')).reset_index()

print(joined_1.head(5))
```

```
# Used BingChat with the following query: "how do I check if the dataframes are
        Geodataframes?"
```

	ZIP_CD	average_distance_miles
0	07004	0.0
1	07005	0.0
2	07006	0.0
3	07039	0.0
4	07046	0.0

The original unit of distance is degrees.

Q5b

```
# Loading Libraries
import pandas as pd
import os
import csv
import geopandas as gpd
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
        gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.
        shp")
# Filter Texas Border States zip codes
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
        "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
        (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
        == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
        (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
        >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
        <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
        (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
        >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
        <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
        >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
        (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
        <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
        (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

# Creating dataframe with all Texas Zipcodes
zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
        & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
        == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
        (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"
```

```
health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16, dtype={'ZIP_CD': str})
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17, dtype={'ZIP_CD': str})
short_term_17 = health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into 1 dataframe and selecting required variables
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)

combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]

# Adding leading 0's in combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str.zfill(6)

# Dropping last digit from combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str[:-1]

# Creating dataframe that has only the zip codes in Texas and Bordering hospitals that each
        have at least one hospital
zips_tb_with_hosp = combined_df_final[
    combined_df_final['ZIP_CD'].isin(zips_texas_borderstates_centroids['ZCTA5']) &
    (combined_df_final["year"] == 2016)]

zips_tb_with_hosp = zips_tb_with_hosp.groupby('ZIP_CD').agg(number_of_hospitals =
        ('PRVDR_NUM', 'count')).reset_index()
zips_tb_with_hosp = zips_tb_with_hosp[zips_tb_with_hosp['number_of_hospitals'] >= 1]

zips_tb_with_hosp_cent =
        zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zips
        _tb_with_hosp['ZIP_CD'])]
# Creating ZCTA5 in combined_df_final, which has all hospital data
combined_df_final['ZCTA5'] = combined_df_final['ZIP_CD']
```

```
combined_df_final_1 = combined_df_final

# Used an inner merge on ZCTA5
zips_withhospital_centroids = pd.merge(combined_df_final_1, zips_tb_with_hosp_cent,
         how='inner', on='ZCTA5')

# Making sure all are geodataframes
if not isinstance(zips_withhospital_centroids, gpd.GeoDataFrame):
    zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
         geometry='geometry')

if 'geometry' not in zips_withhospital_centroids.columns:
    zips_withhospital_centroids['geometry'] = zips_withhospital_centroids.apply(
        lambda row: Point(row['longitude'], row['latitude']), axis=1)

zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
         & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
         == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
         (zips_all_centroids['GEO_ID'] == "8600000US0885") ]



# Ensure both GeoDataFrames use the same coordinate reference system (CRS)
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=4326)
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)

# Perform spatial join to find the nearest hospitals
joined = gpd.sjoin_nearest(zips_texas_centroids, zips_withhospital_centroids, how='inner',
         distance_col='distance')

# Convert distance from degrees to miles (approximation: 1 degree = 69 miles)
joined['distance_miles'] = joined['distance'] * 69
#joined.head(5)

# Calculate the average distance for each zip code
average_distance = joined.groupby('ZIP_CD')['distance_miles'].mean().reset_index()
average_distance.columns = ['ZIP_CD', 'average_distance_miles']

# Merge the average distances back to the original zip code GeoDataFrame
zips_texas_centroids['ZIP_CD'] = zips_texas_centroids['ZCTA5']
zips_texas_centroids = zips_texas_centroids.merge(average_distance, on='ZIP_CD', how='left')
zips_texas_centroids_1 = zips_texas_centroids.groupby('ZIP_CD').agg(average_distance_miles =
         ('average_distance_miles', 'first')).reset_index()

print(zips_texas_centroids_1.head(10))

print("Unique entries for average_distance_miles are",
         zips_texas_centroids['average_distance_miles'].unique())

# Used BingChat with the following query: "how do I use something like this to calculate
         average distance: gpd.sjoin_nearest(zips_texas_centroids,
         zips_withhospital_centroids, how='inner', distance_col='distance')?"
```

```
  ZIP_CD  average_distance_miles
0  07501                    0.0
1  07502                    0.0
```

```
2  07503                    0.0
3  07504                    0.0
4  07505                    0.0
5  07506                    0.0
6  07508                    0.0
7  07512                    NaN
8  07513                    NaN
9  07514                    0.0
Unique entries for average_distance_miles are [0.                nan 0.22204391 0.03691713
0.92859243 0.30291158]
```

Yes, this makes sense (for the NaNs, assuming that they do not have hospitals), as this table shows the average distance from the centroid of the zipcode to the nearest hospital, which can be used to analyze how accessible hospitals are in Texas zipcodes.

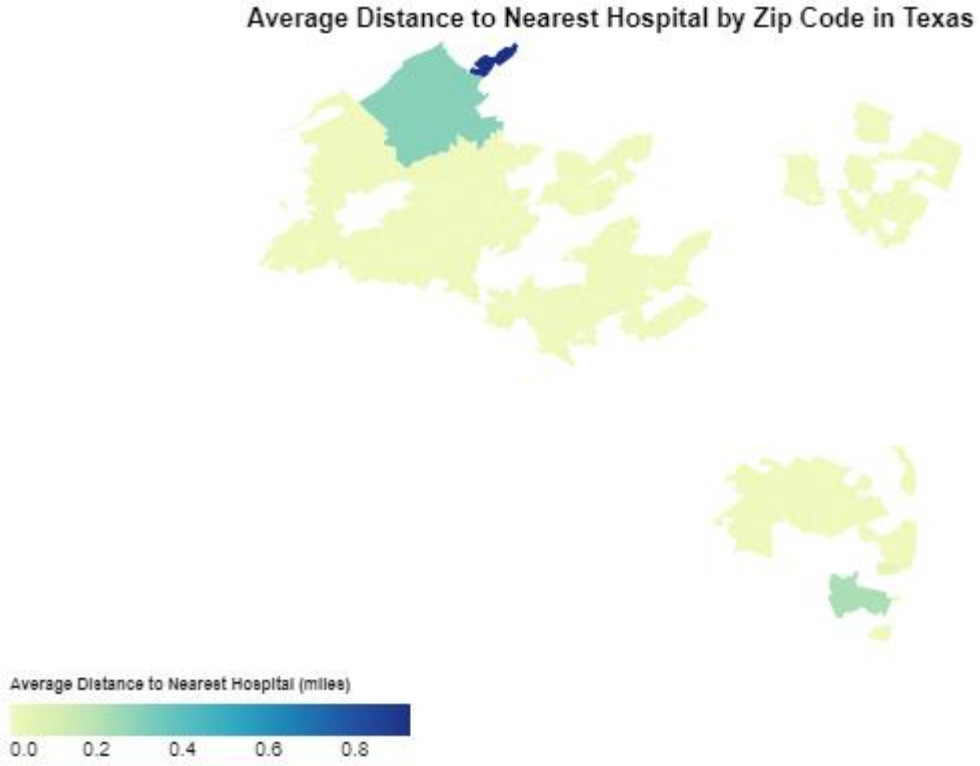
Q5c

```
# Loading Libraries
import pandas as pd
import os
import csv
import geopandas as gpd
import altair as alt
import warnings
warnings.filterwarnings("ignore")

# Load the GeoDataFrame
zips_all_centroids =
         gpd.read_file("C:/Users/prash/Downloads/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k

# Filter Texas Border States zip codes
zips_texas_borderstates_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >=
         "8600000US0750") & (zips_all_centroids['GEO_ID'] < "8600000US0800") |
         (zips_all_centroids['GEO_ID'] == "8600000US0733") | (zips_all_centroids['GEO_ID']
         == "8600000US0718") | (zips_all_centroids['GEO_ID'] == "8600000US0870") |
         (zips_all_centroids['GEO_ID'] == "8600000US0871") | (zips_all_centroids['GEO_ID']
         >= "8600000US0873") & (zips_all_centroids['GEO_ID'] <= "8600000US0875") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0877") & (zips_all_centroids['GEO_ID']
         <= "8600000US0885") | (zips_all_centroids['GEO_ID'] == "8600000US0730") |
         (zips_all_centroids['GEO_ID'] == "8600000US0731") | (zips_all_centroids['GEO_ID']
         >= "8600000US0734") & (zips_all_centroids['GEO_ID'] <= "8600000US0741") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0743") & (zips_all_centroids['GEO_ID']
         <= "8600000US0749") | (zips_all_centroids['GEO_ID'] >= "8600000US0716") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0729") | (zips_all_centroids['GEO_ID']
         >= "8600000US0700") & (zips_all_centroids['GEO_ID'] <= "8600000US0701") |
         (zips_all_centroids['GEO_ID'] >= "8600000US0703") & (zips_all_centroids['GEO_ID']
         <= "8600000US0708") | (zips_all_centroids['GEO_ID'] >= "8600000US0710") &
         (zips_all_centroids['GEO_ID'] <= "8600000US0714") ]

zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
         & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
         == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
         (zips_all_centroids['GEO_ID'] == "8600000US0885") ]

# Importing health data and filtering PRVDR_CTGRY_CD == 1 and PRVDR_CTGRY_SBTYP_CD == 1
base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"
```

```
health_path_16 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2016.csv")
health_data_16 = pd.read_csv(health_path_16, dtype={'ZIP_CD': str})
short_term_16 = health_data_16[(health_data_16["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_16["PRVDR_CTGRY_CD"] == 1)]
short_term_16["year"] = 2016

base_path = r"C:/Users/prash/OneDrive/Documents/Python II/PSet 4"

health_path_17 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2017.csv")
health_data_17 = pd.read_csv(health_path_17, dtype={'ZIP_CD': str})
short_term_17 = health_data_17[(health_data_17["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_17["PRVDR_CTGRY_CD"] == 1)]
short_term_17["year"] = 2017

health_path_18 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2018.csv")
health_data_18 = pd.read_csv(health_path_18, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_18 = health_data_18[(health_data_18["PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_18["PRVDR_CTGRY_CD"] == 1)]
short_term_18["year"] = 2018

health_path_19 = os.path.join(base_path, "C:/Users/prash/OneDrive/Documents/Python II/PSet
        4/POS_File_Hospital_Non_Hospital_Facilities_Q4_2019.csv")
health_data_19 = pd.read_csv(health_path_19, dtype={'ZIP_CD': str}, encoding='latin1')
short_term_19 = health_data_19[(health_data_19["ï»¿PRVDR_CTGRY_SBTYP_CD"] == 1) &
        (health_data_19["PRVDR_CTGRY_CD"] == 1)]
short_term_19["year"] = 2019

# Concatanating into 1 dataframe and selecting required variables
combined_df = pd.concat([short_term_16, short_term_17, short_term_18, short_term_19],
        ignore_index=True)

combined_df_final = combined_df[["FAC_NAME", "PRVDR_CTGRY_CD", "PRVDR_CTGRY_SBTYP_CD",
        "PRVDR_NUM", "ZIP_CD", "PGM_TRMNTN_CD", "year"]]

# Adding Leading 0's in combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str.zfill(6)

# Dropping the last digit from combined_df_final['ZIP_CD']
combined_df_final['ZIP_CD'] = combined_df_final['ZIP_CD'].str[:-1]

# Creating dataframe that has only the zip codes in texas and bordering hospitals that each
        have at least one hospital
zips_tb_with_hosp = combined_df_final[
    combined_df_final['ZIP_CD'].isin(zips_texas_borderstates_centroids['ZCTA5']) &
    (combined_df_final["year"] == 2016)]

zips_tb_with_hosp = zips_tb_with_hosp.groupby('ZIP_CD').agg(number_of_hospitals =
        ('PRVDR_NUM', 'count')).reset_index()
zips_tb_with_hosp = zips_tb_with_hosp[zips_tb_with_hosp['number_of_hospitals'] >= 1]

zips_tb_with_hosp_cent =
        zips_texas_borderstates_centroids[zips_texas_borderstates_centroids['ZCTA5'].isin(zips
        _tb_with_hosp['ZIP_CD'])]
# Creating ZCTA5 in combined_df_final, which has all hospital data
combined_df_final['ZCTA5'] = combined_df_final['ZIP_CD']
```

```
combined_df_final_1 = combined_df_final

# Used an inner merge on ZCTA5
zips_withhospital_centroids = pd.merge(combined_df_final_1, zips_tb_with_hosp_cent,
         how='inner', on='ZCTA5')

# Making sure all are geodataframes
if not isinstance(zips_withhospital_centroids, gpd.GeoDataFrame):
    zips_withhospital_centroids = gpd.GeoDataFrame(zips_withhospital_centroids,
         geometry='geometry')

if 'geometry' not in zips_withhospital_centroids.columns:
    zips_withhospital_centroids['geometry'] = zips_withhospital_centroids.apply(
        lambda row: Point(row['longitude'], row['latitude']), axis=1)


zips_texas_centroids = zips_all_centroids[ (zips_all_centroids['GEO_ID'] >= "8600000US0750")
         & (zips_all_centroids['GEO_ID'] < "8600000US0800") | (zips_all_centroids['GEO_ID']
         == "8600000US0733") | (zips_all_centroids['GEO_ID'] == "8600000US0718") |
         (zips_all_centroids['GEO_ID'] == "8600000US0885") ]


# Ensure both GeoDataFrames use the same coordinate reference system (CRS)
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=4326)
zips_withhospital_centroids = zips_withhospital_centroids.to_crs(epsg=4326)

# Perform spatial join to find the nearest hospitals
joined = gpd.sjoin_nearest(zips_texas_centroids, zips_withhospital_centroids, how='inner',
         distance_col='distance')

# Convert distance from degrees to miles (approximation: 1 degree = 69 miles)
joined['distance_miles'] = joined['distance'] * 69

# Calculate the average distance for each zip code
average_distance = joined.groupby('ZIP_CD')['distance_miles'].mean().reset_index()
average_distance.columns = ['ZIP_CD', 'average_distance_miles']

# Merge the average distances back to the original zip code GeoDataFrame
zips_texas_centroids['ZIP_CD'] = zips_texas_centroids['ZCTA5']
zips_texas_centroids = zips_texas_centroids.merge(average_distance, on='ZIP_CD', how='left')

# Plot the results
chart = alt.Chart(zips_texas_centroids).mark_geoshape().encode(
    color=alt.Color('average_distance_miles:Q', legend=alt.Legend(title="Average Distance to
         Nearest Hospital (miles)", titleFontSize=8)),
    tooltip=['ZCTA5', 'average_distance_miles']
).project(
    type='identity', reflectY=True
).properties(
    title='Average Distance to Nearest Hospital by Zip Code in Texas',
    width=600,
    height=300
).configure_legend(
    orient='bottom'
)
```

```
chart.display()

# Used BingChat to aid in creating a plot that shows the average distance between hospitals
        and zipcodes in altair
# Used BingChat to aid in how to use something like this to calculate average distance:
        gpd.sjoin_nearest(zips_texas_centroids, zips_withhospital_centroids, how='inner',
        distance_col='distance')
```



PSet4 Section 5

AUTHOR
Charisma Lambert Prashanthi Subbiah

PUBLISHED
November 3, 2024

IMPORTANT NOTE:

Our qmd was able to knit to show the code and output from Section 1 Q1 until Section 5 Q1. Section 5 Q2 onwards, the kernal kept crashing for on partners' systems. Thus, we wrote to Professor Maggie Shi, Professor Peter Ganong, and TA Ozzy Houck. We were instructed to comment out the codes of Q2-Q4 of Section 5. They had also mentioned that while grading, the grader should run these codes on their system, and if it runs, we would get full credit, and if the output does not fully match the solutions, we would get partial credit.

The codes for the following questions have been commented out as per these guidelines. Thank you!

Section 5: Effects of closures on access in Texas (15 pts) Partner 1

1.

```
closures_by_zip = merg_aq.groupby("ZIP_CD").agg(NUM_CMS = ("PRVDR_NUM",
         "count")).reset_index()
texas_closures_by_zip = closures_by_zip[closures_by_zip["ZIP_CD"].str.startswith(("75",
         "76", "77", "78", "79"))].reset_index()
print(texas_closures_by_zip)
```

2.

```
import geopandas as gdp
import matplotlib.pyplot as plt

# load .shp file
zip_gdf = gdp.read_file("/Users/charismalambert/Downloads/gz_2010_us_860_00_500k")
zip_gdf = zip_gdf[["ZCTA5", "geometry"]]
zip_gdf = zip_gdf.rename(columns= {"ZCTA5": "ZIP_CD"})
print("ZIP_GDF:", zip_gdf)

zip_gdf["ZIP_CD"] = zip_gdf["ZIP_CD"].astype(str)
aggregated_closures = texas_closures_by_zip.groupby('ZIP_CD').agg({'NUM_CMS':
         'sum'}).reset_index() # Query Corrected

# Merge the GeoDataFrame with the aggregated closures DataFrame
zip_and_geo = zip_gdf.merge(aggregated_closures, on="ZIP_CD", how="inner")  # Query
         Corrected
```

```
# Create the choropleth map
fig, ax = plt.subplots(figsize=(10, 10))
zip_and_geo.plot(column='NUM_CMS', ax=ax, legend=True, cmap='Blues', edgecolor='black')
plt.title('Choropleth Map of Closures by ZIP Code')

plt.show()
#Citation: For Section 5, Q2-4 I was unable to run and edit because my kernel kept crashing.
         I ran my code throguh ChatGPT and explained in my query the goal of each section
         and to correct for potiential error messages. The query corrected lines have been
         commented "query corrected." I have made a post to Ed and sent an email for help /
         potiential solution to complete Section 5 and submit for grading.
```

3.

```
import geopandas as gpd
import pandas as pd
from shapely.geometry import Point

# Load the Texas zip code shapefile
all_zips = gpd.read_file("/Users/charismalambert/Downloads/gz_2010_us_860_00_500k")

closures_gdf = gpd.GeoDataFrame(
    texas_closures_by_zip,
    geometry=gpd.points_from_xy(texas_closures_by_zip["ZIP_CD"].apply(lambda x: Point(x,
         0).x), texas_closures_by_zip["ZIP_CD"].apply(lambda x: Point(x, 0).y)),
    crs='EPSG:4326'
)  # Query Corrected

# Create a 10-mile buffer around the affected zip codes
texas_closures_by_zip['geometry'] = texas_closures_by_zipgeometry.buffer(10 * 1609.34)
closures_buffered = texas_closures_by_zip.dissolve().reset_index()  # Query Corrected
closures_buffered = gpd.GeoDataFrame(closures_buffered)  # Query Corrected

indirectly_affected = gpd.sjoin(all_zips, closures_buffered, how="inner", op="intersects")
         # Query Corrected
num_indirectly_affected_zips = indirectly_affected['ZCTA5'].nunique()

print(f'The number of indirectly affected zip codes in Texas is:
         {num_indirectly_affected_zips}')
#Citation: For Section 5, Q2-4 I was unable to run and edit because my kernel kept crashing.
         I ran my code throguh ChatGPT and explained in my query the goal of each section
         and to correct for potiential error messages. The query corrected lines have been
         commented "query corrected." I have made a post to Ed and sent an email for help /
         potiential solution to complete Section 5 and submit for grading.
```

4.

```
texas_zips = all_zips[all_zips["ZCTA5"].str.startswith(("75","76", "77", "78", "79"))]
texas_zips = texas_zips.rename(columns = {"ZCTA5": "ZIP_CD"})

texas_zips = texas_zips.to_crs(closures_buffered.crs)
indirectly_affected = gpd.sjoin(texas_zips, closures_buffered, how="left", op="intersects")
         # Query Corrected

texas_zips['category'] = 'Not Affected'
```

```
texas_zips.loc[texas_zips['ZIP_CD'].isin(texas_closures_by_zip['ZIP_CD']), 'category'] =
        'Directly Affected'
texas_zips.loc[indirectly_affected['index_right'].notna(), 'category'] = 'Indirectly
        Affected' # Query Corrected

color_map = {'Directly Affected': 'blue', 'Indirectly Affected': 'red', 'Not Affected':
        'green'}

fig, ax = plt.subplots(1, 1, figsize=(10, 8))
texas_zips.plot(column='category', color=texas_zips['category'].map(color_map), legend=True,
        ax=ax) # Query Corrected

ax.set_title('Texas Zip Codes Affected by Hospital Closures', fontsize=15)
ax.set_axis_off()
plt.show()

#Citation: For Section 5, Q2-4 I was unable to run and edit because my kernel kept crashing.
        I ran my code throguh ChatGPT and explained in my query the goal of each section
        and to correct for potiential error messages. The query corrected lines have been
        commented "query corrected." I have made a post to Ed and sent an email for help /
        potiential solution to complete Section 5 and submit for grading.
```

Reflecting on the exercise (10 pts)

Section 6

Partner 1: The “first-pass” method might be misidentifying hospitals, for example a hospital with a temporary closure that is actually still open. One way to do a better job at confirming hospital closures is to cross-reference our findings to state licensing databases and the facility website, there may be a history tab that can help us identify when a change occured.

Partner 2: I believe that this reflects there has not been too much of a decrease in access to hospitals in Texas zipcodes. In this exercise, one manages to filter out the cases of mergers or aquisitions from the closures, which helps mitigate error. However, this could be further improved if we are given more accurate data on where in particular the hospitals are situated, with latitudinal and longitudinal data on the hospitals themselves alongside that of the zipcode.