Kohei Inagaki and Toshiyuki Kindaichi

```
import altair as alt
alt.renderers.enable("png")
```

```
RendererRegistry.enable('png')
```

PS4: Due Sat Nov 2 at 5:00PM Central. Worth 100 points. We use (*) to indicate a problem that we think might be time consuming.

Style Points (10 pts)

Please refer to the minilesson on code style **here**.

Submission Steps (10 pts)

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): Toshiyuki Kindaichi (12410291)
    - Partner 2 (name and cnet ID): Kohei Inagaki (12351305)
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: TK, and KI
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point)
6. Late coins used this pset (Toshiyuki Kindaichi): 1. Late coins left after submission: 2
7. Late coins used this pset (Kohei Inagaki): 1. Late coins left after submission: 3
8. 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.

9. (Partner 1): push `ps4.qmd` and `ps4.pdf` to your github repo.
10. (Partner 1): submit `ps4.pdf` via Gradescope. Add your partner on Gradescope.
11. (Partner 1): tag your submission in Gradescope

Important: Repositories are for tracking code. **Do not commit the data or shapefiles to your repo.** The best way to do this is with `.gitignore`, which we have covered in class. If you do accidentally commit the data, Github has a guide. The best course of action depends on whether you have pushed yet. This also means that both partners will have to download the initial raw data and any data cleaning code will need to be re-run on both partners' computers.

Download and explore the Provider of Services (POS) file (10 pts)

1.

- PRVDR_CTGRY_SBTYP_CD: To identify the types of providers.
- PRVDR_CTGRY_CD: To identify hospitals
- FAC_NAME: To check the name of providers in the Medicare and/or Medicaid programs.
- PRVDR_NUM: To get the CMS certification number to idetify each provider.
- PGM_TRMNTN_CD: TO check the status of providers, especially for 'active' status.
- ZIP_CD: Zip code to identify the physical address of providers. we can make use of zip information with GeoDataFrame to conduct geometric analysis.

2. a.

```
import os
import pandas as pd
import altair as alt
import geopandas as gpd
import matplotlib.pyplot as plt
from shapely.geometry import Point
import pyproj
import time
from shapely.ops import nearest_points
import matplotlib.colors as mcolors
```

```
file_path =
↪  "C:\\Users\\sumos\\OneDrive\\   \\Harris\\2024 \\Python2\\PS\\PS4\\data"
```

```
# Import data
df_2016 = pd.read_csv(f"{file_path}\\pos2016.csv")

# Add the year columns to dataframe for easier data processing in the next
 ↪  problems
df_2016['YEAR'] = '2016'
# When we try to filter the dataset with code 0, no result (0) are displayed.
 ↪  Check the value in the variable 'PRVDR_CTGRY_SBTYP_CD', 'PRVDR_CTGRY_CD',
 ↪  and 'ZIP_CD'

codes_check = ['PRVDR_CTGRY_SBTYP_CD', 'PRVDR_CTGRY_CD']
for column in codes_check:
    unique_codes = df_2016[column].unique()
    print(f"Unique codes in {column}: {unique_codes}")

# Check the datatypes as well
print(df_2016.dtypes)
```

```
Unique codes in PRVDR_CTGRY_SBTYP_CD: [ 1. nan 11.  2.  5.  6.  4.  3. 20.
24.]
Unique codes in PRVDR_CTGRY_CD: [ 1 21 19 16  9 12 14  3  2  4  8  5 10 15 11
6 17  7]
PRVDR_CTGRY_SBTYP_CD    float64
PRVDR_CTGRY_CD            int64
FAC_NAME                 object
PRVDR_NUM                object
PGM_TRMNTN_CD             int64
ZIP_CD                  float64
YEAR                     object
dtype: object
```

We were able to check the datatypes, values(codes), and missing values. Therefore, we will modify them.

```
# Na Check
print(f'2016 - PRVDR_CTGRY_SBTYP_CD NA:',
 ↪  df_2016['PRVDR_CTGRY_SBTYP_CD'].isna().sum())
print(f'2016 - ZIP_CD NA:', df_2016['ZIP_CD'].isna().sum())

# Modify the data:
# 1) Replace NaN in subtype and zip with 0 (there are no code"0" indicating
 ↪  other category)
```

```
# 2)fix the data type: convert PRVDR_CTGRY_SBTYP_CD, PRVDR_CTGRY_CD, and
 ↪  ZIP_CD into str.

df_2016['PRVDR_CTGRY_CD'] = df_2016['PRVDR_CTGRY_CD'].astype(str)
df_2016['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  df_2016['PRVDR_CTGRY_SBTYP_CD'].fillna(0).astype(int).astype(str)
df_2016['ZIP_CD'] = df_2016['ZIP_CD'].fillna(0).astype(int).astype(str)
```

```
2016 - PRVDR_CTGRY_SBTYP_CD NA: 1007
2016 - ZIP_CD NA: 270
```

```
# Filter with code 1
df_short_term_2016 = df_2016[(df_2016['PRVDR_CTGRY_CD'] == '1') &
 ↪  (df_2016['PRVDR_CTGRY_SBTYP_CD'] == '1')]

# Count facility name
facilities_2016 = df_short_term_2016['FAC_NAME'].nunique()
print("Number of facilities (FAC_NAME) in 2016:", facilities_2016)
```

```
Number of facilities (FAC_NAME) in 2016: 6770
```

In the 2016 POS file, there are 6,770 short-term hospitals.

b.

The article 'A Look at Rural Closures and Implication for Access to Care' says 'There are nearly 5,000 short-term, acute care hospitals in the United States.' Therefore, the number of the facilities in the dataset classified as short-term hospital is larger. This could be because the definition of short-term in the dataset covers broader facilities than that mentioned in the article. The article focus on the emergency, so if the short-term in the dataset includes the facilities which cannot deal with urgent cases, the number of the facilities will be larger. Thus, the range of the short-term might generate the difference.

3.

```
# Import data
df_2017 = pd.read_csv(f"{file_path}\\pos2017.csv")
# Attribution; Ask Chatgpt how to deal with utf-8 code error
df_2018 = pd.read_csv(f"{file_path}\\pos2018.csv", encoding='latin1')
df_2019 = pd.read_csv(f"{file_path}\\pos2019.csv", encoding='latin1')
```

```
# Add the year columns to dataframe for easy data processing
df_2017['YEAR'] = '2017'
df_2018['YEAR'] = '2018'
df_2019['YEAR'] = '2019'
# Na Check
print(f'2017 - PRVDR_CTGRY_SBTYP_CD NA:',
 ↪  df_2017['PRVDR_CTGRY_SBTYP_CD'].isna().sum())
print(f'2017 - PRVDR_CTGRY_CD NA:', df_2017['PRVDR_CTGRY_CD'].isna().sum())
print(f'2017 - ZIP_CD NA:', df_2017['ZIP_CD'].isna().sum())

print(f'2018 - PRVDR_CTGRY_SBTYP_CD NA:',
 ↪  df_2018['PRVDR_CTGRY_SBTYP_CD'].isna().sum())
print(f'2018 - PRVDR_CTGRY_CD NA:', df_2018['PRVDR_CTGRY_CD'].isna().sum())
print(f'2018 - ZIP_CD NA:', df_2018['ZIP_CD'].isna().sum())

print(f'2019 - PRVDR_CTGRY_SBTYP_CD NA:',
 ↪  df_2019['PRVDR_CTGRY_SBTYP_CD'].isna().sum())
print(f'2019 - PRVDR_CTGRY_CD NA:', df_2019['PRVDR_CTGRY_CD'].isna().sum())
print(f'2019 - ZIP_CD NA:', df_2019['ZIP_CD'].isna().sum())

# Modify the data: Replace NaN with 0 and fix the data type
df_2017['PRVDR_CTGRY_CD'] =
 ↪  df_2017['PRVDR_CTGRY_CD'].fillna(0).astype(int).astype(str)
df_2017['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  df_2017['PRVDR_CTGRY_SBTYP_CD'].fillna(0).astype(int).astype(str)
df_2017['ZIP_CD'] = df_2017['ZIP_CD'].fillna(0).astype(int).astype(str)

df_2018['PRVDR_CTGRY_CD'] =
 ↪  df_2018['PRVDR_CTGRY_CD'].fillna(0).astype(int).astype(str)
df_2018['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  df_2018['PRVDR_CTGRY_SBTYP_CD'].fillna(0).astype(int).astype(str)
df_2018['ZIP_CD'] = df_2018['ZIP_CD'].fillna(0).astype(int).astype(str)

df_2019['PRVDR_CTGRY_CD'] =
 ↪  df_2019['PRVDR_CTGRY_CD'].fillna(0).astype(int).astype(str)
df_2019['PRVDR_CTGRY_SBTYP_CD'] =
 ↪  df_2019['PRVDR_CTGRY_SBTYP_CD'].fillna(0).astype(int).astype(str)
df_2019['ZIP_CD'] = df_2019['ZIP_CD'].fillna(0).astype(int).astype(str)

# Filter with code 01
df_short_term_2017 = df_2017[(df_2017['PRVDR_CTGRY_CD'] == '1') &
 ↪  (df_2017['PRVDR_CTGRY_SBTYP_CD'] == '1')]
```

```
df_short_term_2018 = df_2018[(df_2018['PRVDR_CTGRY_CD'] == '1') &
 ↪  (df_2018['PRVDR_CTGRY_SBTYP_CD'] == '1')]
df_short_term_2019 = df_2019[(df_2019['PRVDR_CTGRY_CD'] == '1') &
 ↪  (df_2019['PRVDR_CTGRY_SBTYP_CD'] == '1')]

# Count facility name
facilities_2017 = df_short_term_2017['FAC_NAME'].nunique()
print("Number of facilities (FAC_NAME) in 2017:", facilities_2017)
facilities_2018 = df_short_term_2018['FAC_NAME'].nunique()
print("Number of facilities (FAC_NAME) in 2018:", facilities_2018)
facilities_2019 = df_short_term_2019['FAC_NAME'].nunique()
print("Number of facilities (FAC_NAME) in 2019:", facilities_2019)
```

```
2017 - PRVDR_CTGRY_SBTYP_CD NA: 1008
2017 - PRVDR_CTGRY_CD NA: 0
2017 - ZIP_CD NA: 270
2018 - PRVDR_CTGRY_SBTYP_CD NA: 1008
2018 - PRVDR_CTGRY_CD NA: 0
2018 - ZIP_CD NA: 270
2019 - PRVDR_CTGRY_SBTYP_CD NA: 1008
2019 - PRVDR_CTGRY_CD NA: 0
2019 - ZIP_CD NA: 270
Number of facilities (FAC_NAME) in 2017: 6795
Number of facilities (FAC_NAME) in 2018: 6818
Number of facilities (FAC_NAME) in 2019: 6856
```

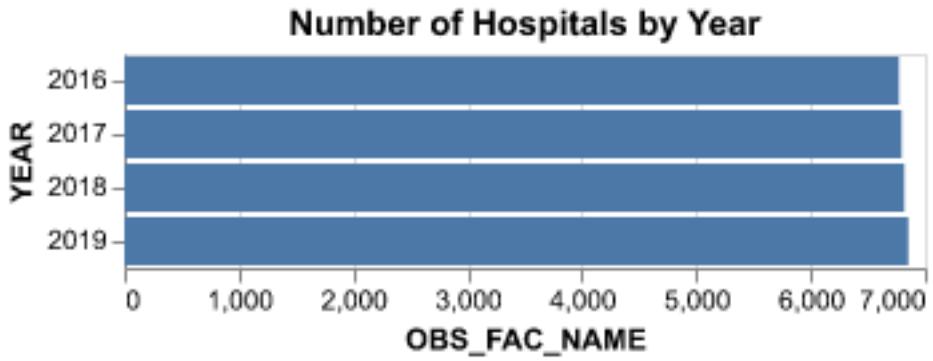
```
# Combine the dataframe with concat
combined_df = pd.concat([df_short_term_2016, df_short_term_2017,
 ↪  df_short_term_2018, df_short_term_2019], ignore_index=True)

# Check the result of combination
print(combined_df.head())
```

```
  PRVDR_CTGRY_SBTYP_CD PRVDR_CTGRY_CD                         FAC_NAME  \
0                    1              1  SOUTHEAST ALABAMA MEDICAL CENTER
1                    1              1            NORTH JACKSON HOSPITAL
2                    1              1     MARSHALL MEDICAL CENTER SOUTH
3                    1              1    ELIZA COFFEE MEMORIAL HOSPITAL
4                    1              1          MIZELL MEMORIAL HOSPITAL

  PRVDR_NUM  PGM_TRMNTN_CD ZIP_CD  YEAR
0    010001              0  36301  2016
```

1	010004	1	35740	2016
2	010005	0	35957	2016
3	010006	0	35631	2016
4	010007	0	36467	2016

```
# Count the observations (FAC_NAME) by year
observations_by_hospital =
 ↪  combined_df.groupby('YEAR')['FAC_NAME'].nunique().reset_index()
observations_by_hospital.columns = ['YEAR', 'OBS_FAC_NAME']
observations_by_hospital = observations_by_hospital.sort_values('YEAR')
# Plot the bar graph with Altair
chart_hos = alt.Chart(observations_by_hospital).mark_bar().encode(
    x = 'OBS_FAC_NAME:Q',
    y = 'YEAR:N'
).properties(
    title='Number of Hospitals by Year'
)
chart_hos.display()
```

```
C:\Users\sumos\anaconda3\Lib\site-packages\altair\utils\core.py:395:
FutureWarning:

the convert_dtype parameter is deprecated and will be removed in a future
version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
```

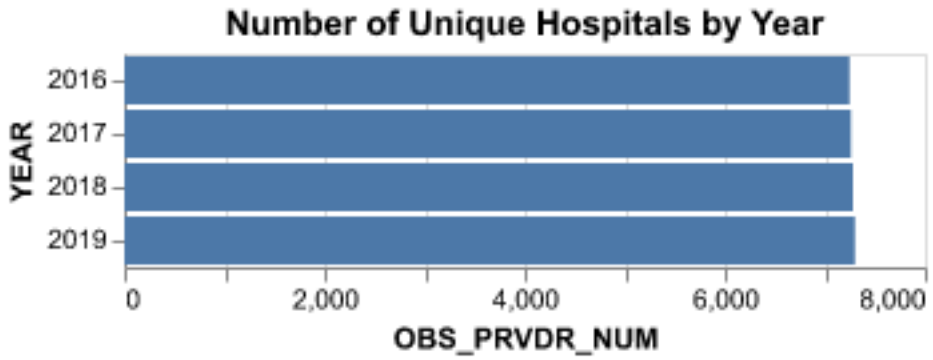


4. a.

```
# Count the observations by CMS
observations_by_cms =
 ↪  combined_df.groupby('YEAR')['PRVDR_NUM'].nunique().reset_index()
```

```
observations_by_cms.columns = ['YEAR', 'OBS_PRVDR_NUM']
observations_by_cms = observations_by_cms.sort_values('YEAR')

# Plot with Altair
chart_cms = alt.Chart(observations_by_cms).mark_bar().encode(
    x = 'OBS_PRVDR_NUM:Q',
    y = 'YEAR:N'
).properties(
    title='Number of Unique Hospitals by Year'
)
chart_cms.display()
```

```
C:\Users\sumos\anaconda3\Lib\site-packages\altair\utils\core.py:395:
FutureWarning:

the convert_dtype parameter is deprecated and will be removed in a future
version.  Do ``ser.astype(object).apply()`` instead if you want
``convert_dtype=False``.
```



The trend suggests that the number of short-term hospitals appears to increase year by year. Since the same hospital may hold multiple CMS certifications, the count based on CMS certification numbers is consistently expected to be higher than the count based on facility names. Anyway, it seems that the number of short-term hospitals increases in terms of name and CCN unless the closure hospitals are removed from the dataset.

On the other hand, we notice in the dataset that closed hospitals could not have been properly excluded from the dataset by using some specific codes such as termination code. Instead, some hospitals are simply labeled as just "(CLOSED)" in FAC_NAME in the dataset. This data processing not only leads to just increase the data observations but also makes it difficult to accurately identify which facility is closed using specific code in this dataset.

Identify hospital closures in POS file (15 pts) (*)

1.

```
active_2016 = df_short_term_2016[df_short_term_2016['PGM_TRMNTN_CD'] ==
↪  0][['PRVDR_NUM', 'FAC_NAME', 'ZIP_CD']].copy()

# Create the list for supspected closure
suspected_closures = pd.DataFrame()
active_hospitals = active_2016.copy()

# Record the suspected closure year for each year
for year, df_year in zip([2017, 2018, 2019], [df_short_term_2017,
↪  df_short_term_2018, df_short_term_2019]):
    active_in_year = df_year[df_year['PGM_TRMNTN_CD'] == 0]['PRVDR_NUM']
    closures_in_year =
↪  active_hospitals[~active_hospitals['PRVDR_NUM'].isin(active_in_year)].copy()
    closures_in_year['Suspected_Closure_Year'] = year
    # Add them to the list of supspected closure
    suspected_closures = pd.concat([suspected_closures, closures_in_year])
    # Update the active hospital list for next year
    active_hospitals =
↪  active_hospitals[active_hospitals['PRVDR_NUM'].isin(active_in_year)].copy()

# Drop the duplicate hospitals showing up in each year
closed_hospitals = suspected_closures.drop_duplicates(subset=['PRVDR_NUM'])

# Result
print("Total number of suspected closures from 2016 to 2019:",
↪  len(closed_hospitals))
print(closed_hospitals[['FAC_NAME', 'ZIP_CD', 'Suspected_Closure_Year']])
```

```
Total number of suspected closures from 2016 to 2019: 174
                                                  FAC_NAME ZIP_CD  \
2404                                ABRAZO MARYVALE CAMPUS  85031
6566           ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230
6759                          FALLBROOK HOSPITAL DISTRICT  92028
14798           ARKANSAS VALLEY REGIONAL MEDICAL CENTER  81050
14805                            KEEFE MEMORIAL HOSPITAL  80810
...                                                 ...    ...
133498                              TEXAS GENERAL HOSPITAL  75051
133501  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613
```

```
133508           LITTLE RIVER HEALTHCARE CAMERON HOSPITAL  76520
133511                     BAYLOR EMERGENCY MEDICAL CENTER  75087
133529                   TEXAS GENERAL HOSPITAL- VZRMC LP  75140

        Suspected_Closure_Year
2404                      2017
6566                      2017
6759                      2017
14798                     2017
14805                     2017
...                        ...
133498                    2019
133501                    2019
133508                    2019
133511                    2019
133529                    2019

[174 rows x 3 columns]
```

2.

```
sorted_closed_hospitals = closed_hospitals.sort_values(by='FAC_NAME')
print("First 10 rows of sorted suspected closures:")
print(sorted_closed_hospitals[['FAC_NAME',
 ↪  'Suspected_Closure_Year']].head(10))
```

```
First 10 rows of sorted suspected closures:
                                          FAC_NAME  Suspected_Closure_Year
2404                          ABRAZO MARYVALE CAMPUS                    2017
6566        ADVENTIST MEDICAL CENTER - CENTRAL VALLEY                    2017
86758                         AFFINITY MEDICAL CENTER                    2018
78926   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS                    2017
110054       ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE                    2017
66094                         ALLIANCE LAIRD HOSPITAL                    2019
93009                        ALLIANCEHEALTH DEACONESS                    2019
19313                   ANNE BATES LEACH EYE HOSPITAL                    2019
14798         ARKANSAS VALLEY REGIONAL MEDICAL CENTER                    2017
74314             BANNER CHURCHILL COMMUNITY HOSPITAL                    2017
```

3. a. b. c.

```
# Same process as Q1
active_2016 = df_short_term_2016[df_short_term_2016['PGM_TRMNTN_CD'] ==
 ↪  0][['PRVDR_NUM', 'FAC_NAME', 'ZIP_CD']].copy()

suspected_closures = pd.DataFrame()
active_hospitals = active_2016.copy()

for year, df_year in zip([2017, 2018, 2019], [df_short_term_2017,
 ↪  df_short_term_2018, df_short_term_2019]):
    active_in_year = df_year[df_year['PGM_TRMNTN_CD'] == 0]['PRVDR_NUM']
    closures_in_year =
 ↪  active_hospitals[~active_hospitals['PRVDR_NUM'].isin(active_in_year)].copy()
    closures_in_year['Suspected_Closure_Year'] = year
    suspected_closures = pd.concat([suspected_closures, closures_in_year])
    active_hospitals =
 ↪  active_hospitals[active_hospitals['PRVDR_NUM'].isin(active_in_year)].copy()

closed_hospitals = suspected_closures.drop_duplicates(subset=['PRVDR_NUM'])

# Create dictionary of each year for further comparison
data_by_year = {
    2016: df_short_term_2016,
    2017: df_short_term_2017,
    2018: df_short_term_2018,
    2019: df_short_term_2019
}

# Set the blank list for suspected hospitals of merger/acquision
potential_mergers = []

for _, row in closed_hospitals.iterrows():
    zip_code = row['ZIP_CD']
    year_of_closure = row['Suspected_Closure_Year']
    # Compare to the previous year data
    if year_of_closure <= 2019:
        # Dataframe for suspected closure for previous/current
        prev_year_df = data_by_year[year_of_closure - 1]
        current_year_df = data_by_year[year_of_closure]
        # Count the suspected closure in previous year per zip
        num_active_prev_year = len(prev_year_df[(prev_year_df['ZIP_CD'] ==
 ↪  zip_code) &  (prev_year_df['PGM_TRMNTN_CD'] == 0)])
        # Count the suspected closure in current year per zip
```

```
        num_active_current_year =
↪  len(current_year_df[(current_year_df['ZIP_CD'] == zip_code) &
↪  (current_year_df['PGM_TRMNTN_CD'] == 0)])
        # Add list if the number of suspected does not decrease
        if num_active_current_year >= num_active_prev_year:
            potential_mergers.append(row)

# Display the suspected merger/acquision iist
potential_mergers_df = pd.DataFrame(potential_mergers)
print("Number of potential mergers/acquisitions:", len(potential_mergers_df))

# Create the list without merger/acquision
corrected_closed_hospitals =
↪  closed_hospitals[~closed_hospitals['PRVDR_NUM'].isin(potential_mergers_df['PRVDR_NUM'])]
print("Number of corrected closures:", len(corrected_closed_hospitals))

# Sort with name and display the first 10 rows
sorted_corrected_closed_hospitals =
↪  corrected_closed_hospitals.sort_values(by='FAC_NAME')
print("First 10 rows of corrected sorted closures:")
print(sorted_corrected_closed_hospitals[['FAC_NAME', 'ZIP_CD',
↪  'Suspected_Closure_Year']].head(10))
```

```
Number of potential mergers/acquisitions: 8
Number of corrected closures: 166
First 10 rows of corrected sorted closures:
                                              FAC_NAME ZIP_CD  \
2404                          ABRAZO MARYVALE CAMPUS  85031
6566        ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230
86758                        AFFINITY MEDICAL CENTER  44646
78926   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS  12208
110054       ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE  75662
66094                         ALLIANCE LAIRD HOSPITAL  39365
93009                       ALLIANCEHEALTH DEACONESS  73112
19313                  ANNE BATES LEACH EYE HOSPITAL  33136
14798         ARKANSAS VALLEY REGIONAL MEDICAL CENTER  81050
74314             BANNER CHURCHILL COMMUNITY HOSPITAL  89406

        Suspected_Closure_Year
2404                      2017
6566                      2017
86758                     2018
```

```
78926                    2017
110054                   2017
66094                    2019
93009                    2019
19313                    2019
14798                    2017
74314                    2017
```

Download Census zip code shapefile (10 pt)

1. a.

```
# Import shape file
gd_zip = gpd.read_file(f"{file_path}\\gz_2010_us_860_00_500k.shp")
print(gd_zip.head())

# Check the datatypes as well
print(gd_zip.dtypes)
```

```
           GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
0  8600000US01040  01040  01040  ZCTA5      21.281
1  8600000US01050  01050  01050  ZCTA5      38.329
2  8600000US01053  01053  01053  ZCTA5       5.131
3  8600000US01056  01056  01056  ZCTA5      27.205
4  8600000US01057  01057  01057  ZCTA5      44.907

                                            geometry
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...
4  MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...
GEO_ID          object
ZCTA5           object
NAME            object
LSAD            object
CENSUSAREA     float64
geometry      geometry
dtype: object
```

The zip file has 5 types of files as follows: * dbf which has attribute information such as GEO_ID, ZCTA5(zip info), and CENSUSAREA (area info). * prj describes Coordinate Reference System (CRS) which has GEOGCS (Geographic Coordinate System), DATUM, UNIT (degree), etc,. * shp which has feature geometrics * shx which has positional index infromation * xml which has meta information such as sitation, abstract, and purpose.

b.

```
# Set the list of file names
shp_file_names = [
    "gz_2010_us_860_00_500k.dbf",
    "gz_2010_us_860_00_500k.prj",
    "gz_2010_us_860_00_500k.shp",
    "gz_2010_us_860_00_500k.shx",
    "gz_2010_us_860_00_500k.xml"
]

# Check the file size
for shp_file_name in shp_file_names:
    shp_fulll_path = os.path.join(file_path, shp_file_name)
    file_size = os.path.getsize(shp_fulll_path) / (1024 * 1024)
    print(f"{shp_file_name}: {file_size:.2f} MB")
```

```
gz_2010_us_860_00_500k.dbf: 6.13 MB
gz_2010_us_860_00_500k.prj: 0.00 MB
gz_2010_us_860_00_500k.shp: 798.74 MB
gz_2010_us_860_00_500k.shx: 0.25 MB
gz_2010_us_860_00_500k.xml: 0.01 MB
```

2.

Zip codse in Texas start from 733, 75, 76, 77, 78, or 79 based on Wikipedia.link.

```
# Restrict to Texas
texas_gd_zip = gd_zip[gd_zip['ZCTA5'].str.startswith(('75', '76', '77', '78',
↪  '79', '733'))]
# check the result
print(texas_gd_zip.head())
```

```
              GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
9207  8600000US78624  78624  78624  ZCTA5     708.041
9208  8600000US78626  78626  78626  ZCTA5      93.046
9209  8600000US78628  78628  78628  ZCTA5      73.382
```

```
9210  8600000US78631  78631  78631  ZCTA5     325.074
9211  8600000US78632  78632  78632  ZCTA5      96.278

                                                 geometry
9207  POLYGON ((-98.96423 30.49848, -98.96416 30.498...
9208  POLYGON ((-97.60944 30.57185, -97.61688 30.568...
9209  POLYGON ((-97.69285 30.57122, -97.69286 30.571...
9210  POLYGON ((-99.13053 30.36555, -99.13065 30.365...
9211  POLYGON ((-97.40946 29.75929, -97.40947 29.758...
```

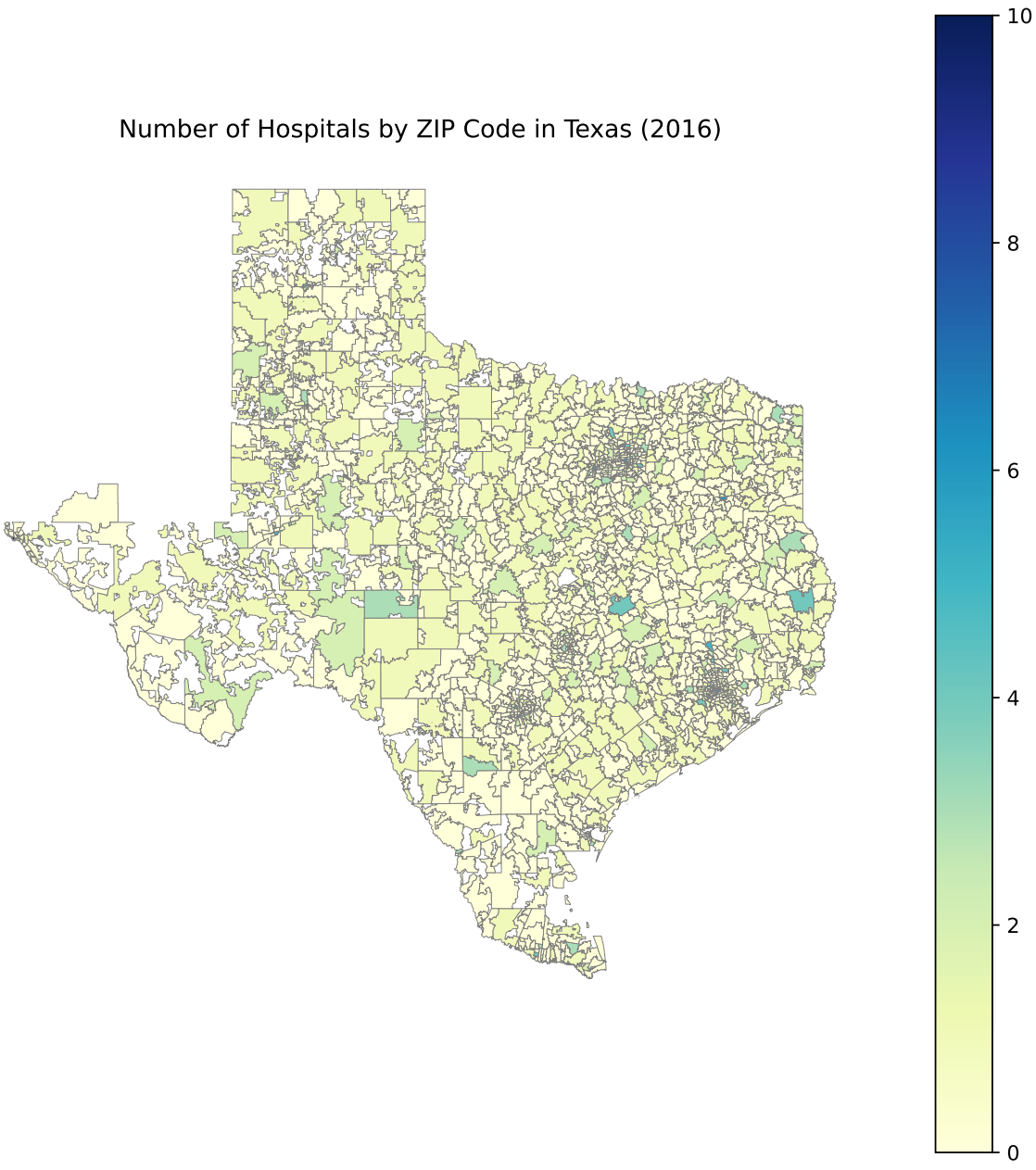
```
# Calculate the number of hospital per zip in 2016
hospital_2016_by_zip =
 ↪  df_short_term_2016.groupby('ZIP_CD').size().reset_index()
hospital_2016_by_zip.columns = ['ZIP_CD', 'NUM_OF_HSPTL']

# Merge the 2016 POS file and zip code shapefile
texas_gd_zip = texas_gd_zip.merge(hospital_2016_by_zip, left_on = 'ZCTA5',
 ↪  right_on = 'ZIP_CD', how = 'left')

# Replace Na with 0 in 'ZIP' and 'NUM_OF_HSPTL'
texas_gd_zip['ZIP_CD'] = texas_gd_zip['ZIP_CD'].fillna(0)
texas_gd_zip['NUM_OF_HSPTL'] = texas_gd_zip['NUM_OF_HSPTL'].fillna(0)

print(texas_gd_zip.head(3))
```

```
           GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
0  8600000US78624  78624  78624  ZCTA5     708.041
1  8600000US78626  78626  78626  ZCTA5      93.046
2  8600000US78628  78628  78628  ZCTA5      73.382

                                          geometry ZIP_CD  NUM_OF_HSPTL
0  POLYGON ((-98.96423 30.49848, -98.96416 30.498...  78624           1.0
1  POLYGON ((-97.60944 30.57185, -97.61688 30.568...  78626           1.0
2  POLYGON ((-97.69285 30.57122, -97.69286 30.571...      0           0.0
```

```
# Count the number of hospitals per zip in 2016
total_hospitals = texas_gd_zip['NUM_OF_HSPTL'].sum()
print("Total number of hospitals:", total_hospitals)

# Plot the choropleth
fig, ax = plt.subplots(1, 1, figsize = (10, 10))
texas_gd_zip.plot(column = 'NUM_OF_HSPTL', cmap = 'YlGnBu', linewidth = 0.2,
 ↪  ax = ax, edgecolor='0.5', legend = True)
```

```
ax.set_title("Number of Hospitals by ZIP Code in Texas (2016)")
ax.set_axis_off()
plt.show()
```

```
Total number of hospitals: 731.0
```



Calculate zip code's distance to the nearest hospital (20 pts) (*)

1.

```
# Inspect column names to identify the correct ZIP code column
print("Column names in zips_all:", gd_zip.columns)
print("Column names in zips_all_centroids:", gd_zip.columns)

# Set the correct ZIP code column
zip_code_column = 'ZCTA5'

# Create a GeoDataFrame for centroids of each ZIP code
zips_all_centroids = gd_zip.copy()
zips_all_centroids["geometry"] = zips_all_centroids["geometry"].centroid

# Display dimensions and columns
print("Dimensions of the GeoDataFrame:", zips_all_centroids.shape)
print("Columns in the GeoDataFrame:")
print(zips_all_centroids.columns)
print("\nColumn descriptions:")
for column in zips_all_centroids.columns:
    print(f"- {column}: contains {zips_all_centroids[column].dtype} type
     ↪  data, representing {column.lower().replace('_', ' ')}.")
```

```
Column names in zips_all: Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD',
'CENSUSAREA', 'geometry'], dtype='object')
Column names in zips_all_centroids: Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD',
'CENSUSAREA', 'geometry'], dtype='object')

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\2607317812.py:10:
UserWarning:

Geometry is in a geographic CRS. Results from 'centroid' are likely
incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a projected
CRS before this operation.


Dimensions of the GeoDataFrame: (33120, 6)
Columns in the GeoDataFrame:
Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA', 'geometry'],
dtype='object')

Column descriptions:
```

- GEO_ID: contains object type data, representing geo id.
- ZCTA5: contains object type data, representing zcta5.
- NAME: contains object type data, representing name.
- LSAD: contains object type data, representing lsad.
- CENSUSAREA: contains float64 type data, representing censusarea.
- geometry: contains geometry type data, representing geometry.

2.

```
# Step 2: Create GeoDataFrame subsets for Texas ZIP codes and bordering
 ↪  states
texas_prefixes = ('75', '76', '77', '78', '79', '733')
bordering_state_prefixes = texas_prefixes + ('73', '70', '72', '68', '71')  #
 ↪  Adding prefixes for nearby states

# Filter for Texas ZIP code centroids
zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids[zip_code_column].astype(str).str.startswith(texas_

# Filter for Texas and bordering states ZIP code centroids
zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids[zip_code_column].astype(str).str.startswith(border

# Print the number of unique ZIP codes in each subset
print("Number of unique ZIP codes in Texas:",
 ↪  zips_texas_centroids[zip_code_column].nunique())
print("Number of unique ZIP codes in Texas and bordering states:",
 ↪  zips_texas_borderstates_centroids[zip_code_column].nunique())

# Display the first few rows to verify
print("\nFirst few rows of Texas ZIP code centroids:")
print(zips_texas_centroids.head())
print("\nFirst few rows of Texas and bordering states ZIP code centroids:")
print(zips_texas_borderstates_centroids.head())
```

```
Number of unique ZIP codes in Texas: 1935
Number of unique ZIP codes in Texas and bordering states: 3827

First few rows of Texas ZIP code centroids:
              GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
9207  8600000US78624  78624  78624  ZCTA5     708.041
9208  8600000US78626  78626  78626  ZCTA5      93.046
9209  8600000US78628  78628  78628  ZCTA5      73.382
```

```
9210  8600000US78631  78631  78631  ZCTA5     325.074
9211  8600000US78632  78632  78632  ZCTA5      96.278

                        geometry
9207   POINT (-98.87707 30.2816)
9208  POINT (-97.59733 30.66535)
9209  POINT (-97.75112 30.64108)
9210  POINT (-99.30528 30.33772)
9211  POINT (-97.47045 29.69633)

First few rows of Texas and bordering states ZIP code centroids:
              GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
8607  8600000US68002  68002  68002  ZCTA5      72.401
8608  8600000US68004  68004  68004  ZCTA5      78.698
8609  8600000US68007  68007  68007  ZCTA5      35.921
8610  8600000US68017  68017  68017  ZCTA5      71.504
8611  8600000US68018  68018  68018  ZCTA5      43.733

                        geometry
8607  POINT (-96.33703 41.49689)
8608  POINT (-96.63226 42.01245)
8609   POINT (-96.19138 41.3698)
8610  POINT (-96.65488 41.05952)
8611   POINT (-96.6123 41.30543)
```

3.

Then, create a subset with ZIP codes having at least 1 hospital in 2016. In previous question, We have calulated the number of hospitals per ZIP code in 2016; 'hospital_2016_by_zip'

```
# Step 3: Create a subset with ZIP codes having at least 1 hospital in 2016
# Assuming we have already created zips_texas_borderstates_centroids in the
 ↪  previous steps

# Merge zips_texas_borderstates_centroids with hospital_2016_by_zip
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    hospital_2016_by_zip[hospital_2016_by_zip['NUM_OF_HSPTL'] >= 1],
    how='inner',
    left_on='ZCTA5',  # Assuming 'ZCTA5' is the ZIP code column in
 ↪  zips_texas_borderstates_centroids
    right_on='ZIP_CD'
)
```

```
# Display information about the resulting GeoDataFrame
print("Number of ZIP codes with at least 1 hospital:",
 ↪  len(zips_withhospital_centroids))
print("\nFirst few rows of zips_withhospital_centroids:")
print(zips_withhospital_centroids.head())
```

```
Number of ZIP codes with at least 1 hospital: 857

First few rows of zips_withhospital_centroids:
           GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
0  8600000US68047  68047  68047  ZCTA5     126.239
1  8600000US68071  68071  68071  ZCTA5      75.085
2  8600000US68122  68122  68122  ZCTA5      19.330
3  8600000US68124  68124  68124  ZCTA5       5.739
4  8600000US68130  68130  68130  ZCTA5       7.631

                     geometry ZIP_CD  NUM_OF_HSPTL
0  POINT (-96.74046 42.10671)  68047             1
1  POINT (-96.47116 42.23563)  68071             1
2  POINT (-96.05106 41.36825)  68122             2
3   POINT (-96.05159 41.2353)  68124             1
4  POINT (-96.19589 41.23423)  68130             1
```

An inner merge was performed using the following variables: 'ZCTA5': the ZIP code column from the zips_texas_borderstates_centroids GeoDataFrame 'ZIP_CD': the ZIP code column from the 'hospital_2016_by_zip' DataFrame This merge type was chosen to create a GeoDataFrame that includes only ZIP codes with at least one hospital in 2016. The inner merge ensures that only ZIP codes present in both dataframes are included in the final result. The purpose of this merge was to combine geographical data of ZIP codes in Texas and bordering states with hospital count information, specifically for ZIP codes that have at least one hospital in 2016.

4. a.

```
# Subtract 10 sample set from zips_texas_centroids
zips_texas_subset = zips_texas_centroids.sample(10)

# Estimate the time for 10 sample set
start_time = time.time()
for idx, row in zips_texas_subset.iterrows():
    point = row.geometry
    nearest = nearest_points(point,
 ↪  zips_withhospital_centroids.unary_union)[1]
```

```python
    zips_texas_subset.loc[idx, 'nearest_hospital_distance'] =
 ↪  point.distance(nearest)
end_time = time.time()
subset_time = end_time - start_time

print(f"Time taken for 10 zip codes: {subset_time:.2f} seconds")

# Convert time into seconds
estimated_time = subset_time * (len(zips_texas_centroids) / 10)
print(f"Estimated time for full dataset: {estimated_time:.2f} seconds")
```

```
Time taken for 10 zip codes: 0.01 seconds
Estimated time for full dataset: 2.13 seconds
```

```
C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:
```

```
The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1813684441.py:8:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.
```

b.

```python
start_time = time.time()

# Measure the full time for full set
for idx, row in zips_texas_centroids.iterrows():
    point = row.geometry
    nearest = nearest_points(point,
 ↪  zips_withhospital_centroids.unary_union)[1]
    zips_texas_centroids.loc[idx, 'nearest_hospital_distance'] =
 ↪  point.distance(nearest)
```

```
end_time = time.time()
full_time = end_time - start_time

print(f"Actual time for full dataset: {full_time:.2f} seconds")
print(f"Difference from estimation: {abs(full_time - estimated_time):.2f}
 ↪  seconds")
```

```
C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1935822623.py:6:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\anaconda3\Lib\site-packages\geopandas\geodataframe.py:1819:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1935822623.py:6:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1935822623.py:6:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\1935822623.py:6:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.
```

```
# Define the file path for the .prj file
prj_file_path = os.path.join(file_path, "gz_2010_us_860_00_500k.prj")

# Read the .prj file
with open(prj_file_path, 'r') as prj_file:
    prj_text = prj_file.read()

print(f"CRS from .prj file: {prj_text}")

# Get the units
crs = pyproj.CRS.from_string(prj_text)
units = crs.axis_info[0].unit_name

print(f"Units: {units}")
```

```
CRS from .prj file:
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,29
Units: Degree
```

We can see the unit is degree. Therefore, we convert it into mile with to_crs. (Attribution: ChatGPT)

```
# If the units are in degrees, reproject to a UTM zone (for Texas,
 ↪ EPSG:32614)
if units.lower() == 'degree':
    # Convert to EPSG:32614 which uses meters
    zips_texas_centroids = zips_texas_centroids.to_crs(epsg=32614)
    zips_withhospital_centroids =
 ↪ zips_withhospital_centroids.to_crs(epsg=32614)

# Calculate the distance to the nearest hospital in meters
zips_texas_centroids['nearest_hospital_distance_m'] = [
    point.distance(nearest_points(point,
 ↪ zips_withhospital_centroids.unary_union)[1])
    for point in zips_texas_centroids.geometry
]

# Convert meters to miles (1 mile = 1609.34 meters)
zips_texas_centroids['nearest_hospital_distance_miles'] =
 ↪ zips_texas_centroids['nearest_hospital_distance_m'] / 1609.34
```

```
# Display the results
print(zips_texas_centroids[['ZCTA5', 'nearest_hospital_distance_m',
 ↪  'nearest_hospital_distance_miles']].head())
```

```
C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\2031627860.py:9:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

      ZCTA5  nearest_hospital_distance_m  nearest_hospital_distance_miles
9207  78624                     0.000000                         0.000000
9208  78626                     0.000000                         0.000000
9209  78628                 12197.888778                         7.579436
9210  78631                 36489.177324                        22.673380
9211  78632                 15883.323167                         9.869464
```

5.

a.

```
# Calculate the distance to the nearest hospital in meters
zips_texas_centroids['nearest_hospital_distance_m'] =
 ↪  zips_texas_centroids.apply(
    lambda row: nearest_points(row.geometry,
     ↪  zips_withhospital_centroids.unary_union)[1].distance(row.geometry),
    axis=1
)

# The unit of the distance
crs = zips_texas_centroids.crs
print(f"The distance is in {crs.axis_info[0].unit_name}")
```

```
C:\Users\sumos\AppData\Local\Temp\ipykernel_41340\340823677.py:3:
DeprecationWarning:

The 'unary_union' attribute is deprecated, use the 'union_all()' method
instead.

The distance is in metre
```

b.

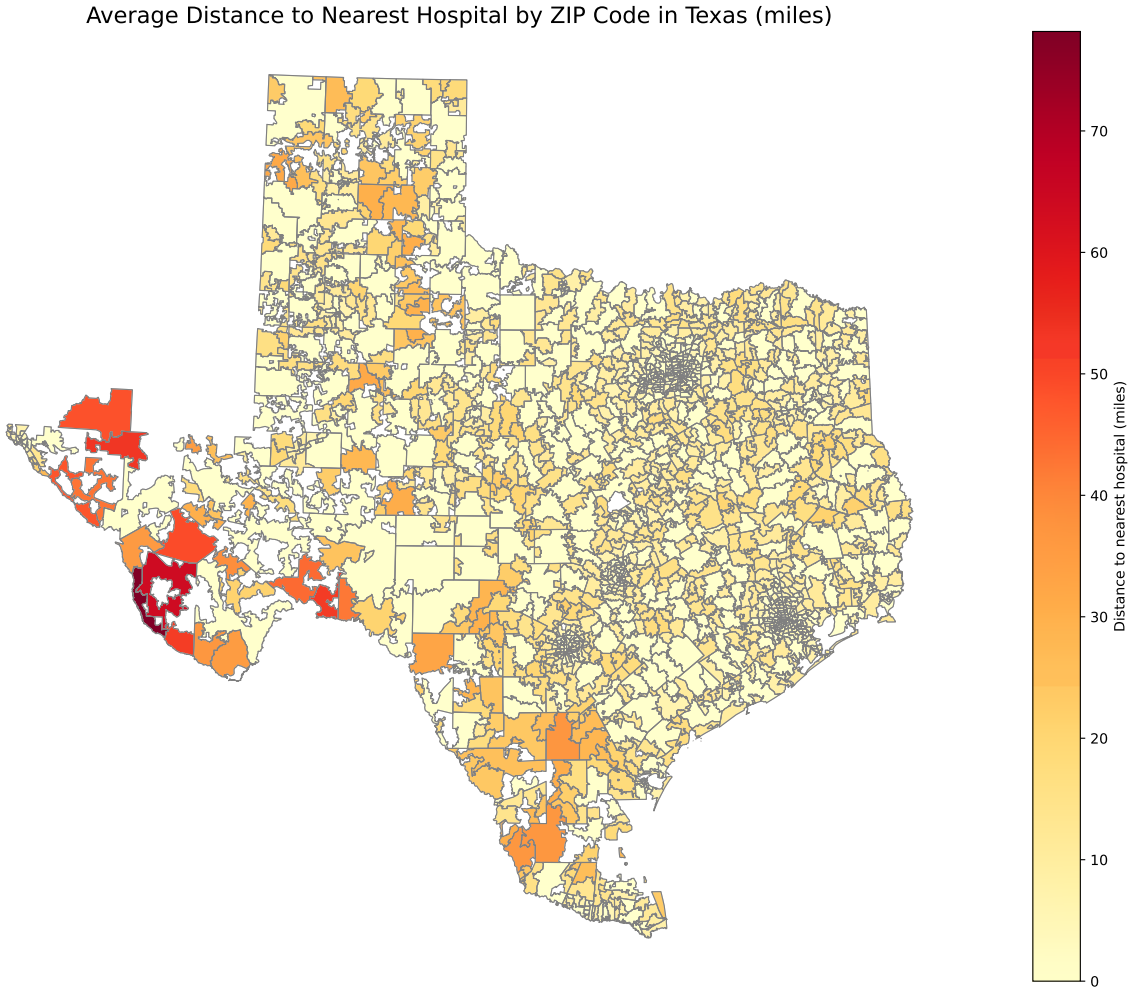
```
# Convert distances to miles (1 mile = 1609.34 meters)
zips_texas_centroids['nearest_hospital_distance_miles'] =
 ↪  zips_texas_centroids['nearest_hospital_distance_m'] / 1609.34

average_distance_miles =
 ↪  zips_texas_centroids['nearest_hospital_distance_miles'].mean()
print(f"Average distance to the nearest hospital in Texas ZIP codes:
 ↪  {average_distance_miles:.2f} miles")
```

```
Average distance to the nearest hospital in Texas ZIP codes: 8.19 miles
```

c.

```
# Convert 'texas_gd_zip' into same CRS as zips_texas_centroids
texas_gd_zip = texas_gd_zip.to_crs(zips_texas_centroids.crs)

# Merge and prepare data for visualization
texas_gd_zip_distance = texas_gd_zip.merge(
    zips_texas_centroids[['ZCTA5', 'nearest_hospital_distance_miles']],
    on='ZCTA5',
    how='left'
)

# Make plot
fig, ax = plt.subplots(1, 1, figsize=(15, 10))

# Use colormap ('YlOrRd') for mapping
texas_gd_zip_distance.plot(
    column='nearest_hospital_distance_miles',
    cmap='YlOrRd',
    linewidth=0.5,
    edgecolor='0.8',
    ax=ax,
    legend=True,
    legend_kwds={'label': 'Distance to nearest hospital (miles)'},
    missing_kwds={'color': 'lightgrey'}
)
# Plot the gray boundary of zip code
texas_gd_zip_distance.boundary.plot(
    color='gray',
    linewidth=0.8,
    ax=ax
```

```
)
# Set title
ax.set_title("Average Distance to Nearest Hospital by ZIP Code in Texas
 ↪  (miles)", fontsize=16)
ax.set_axis_off()
plt.tight_layout()
plt.show()
```



Effects of closures on access in Texas (15 pts)

1.

```
# Filter the closure dataframe with tx zip code
closures_tx =
 ↪  corrected_closed_hospitals[corrected_closed_hospitals['ZIP_CD'].astype(str).str.startswi
 ↪  '76', '77', '78', '79', '733'))]

# Count the number of closure in Tx per zip code
closures_tx_by_zip = closures_tx.groupby('ZIP_CD').size().reset_index()
closures_tx_by_zip.columns = ['TX_ZIP_CD', 'NUM_OF_CLSR']

# Display the table
pd.set_option('display.max_rows', None)
print("Number of ZIP codes vs. Number of closures:")
display(closures_tx_by_zip)
pd.reset_option('display.max_rows')
```

```
Number of ZIP codes vs. Number of closures:
```

	TX_ZIP_CD	NUM_OF_CLSR
0	75042	1
1	75051	1
2	75087	1
3	75140	1
4	75231	1
5	75235	1
6	75390	1
7	75601	1
8	75662	1
9	75835	1
10	75862	1
11	76502	1
12	76520	1
13	76531	1
14	76645	1
15	77035	1
16	77054	1
17	77065	1
18	77429	1
19	77479	1
20	77598	1
21	78017	1
22	78061	1

	TX_ZIP_CD	NUM_OF_CLSR
23	78336	1
24	785	1
25	78613	1
26	78734	1
27	78834	1
28	79520	1
29	79529	1
30	79553	1
31	79735	1
32	79761	1
33	79902	1

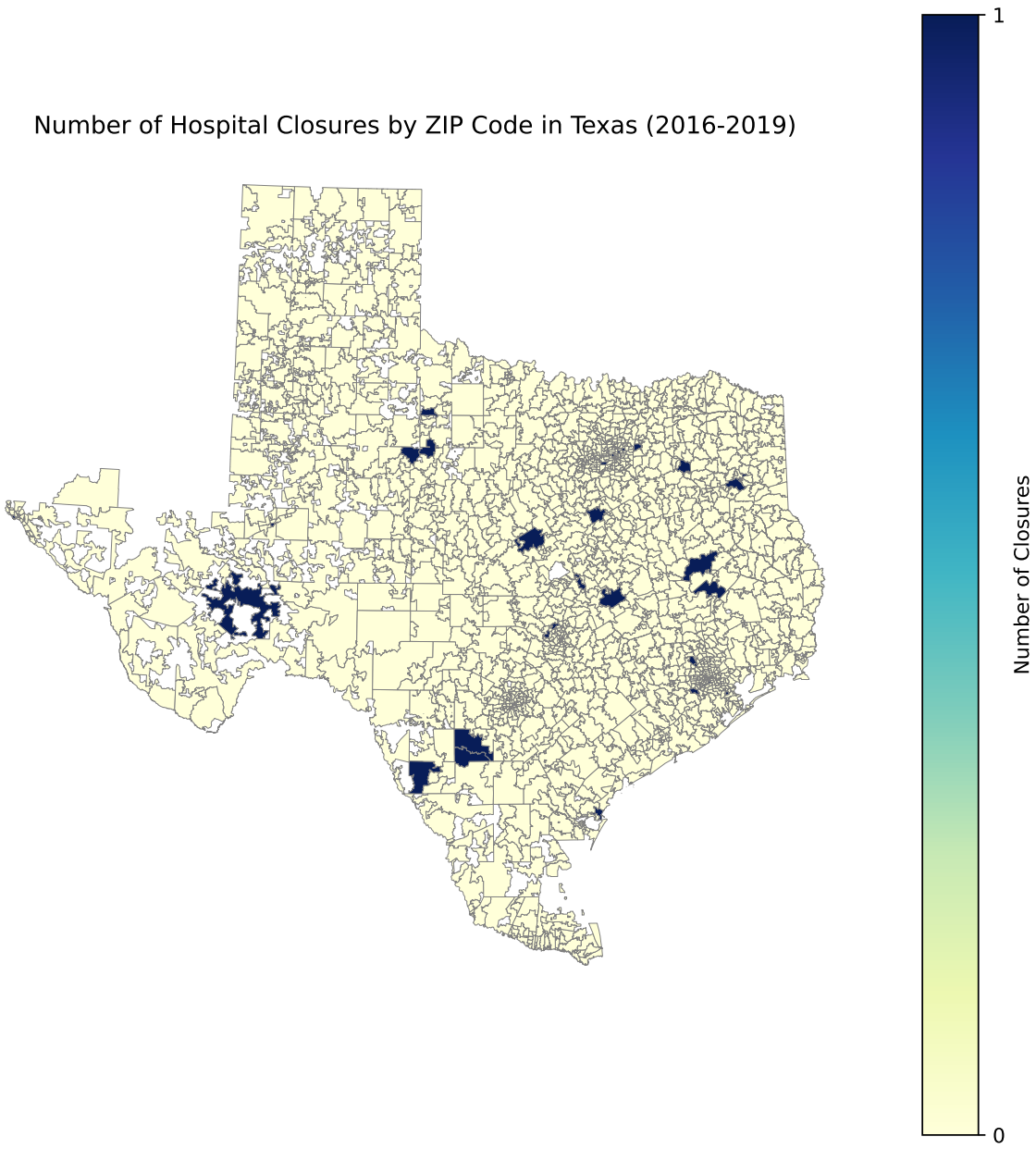
2.

```
# Count the zip directly affected
directly_affected_zip = closures_tx_by_zip['TX_ZIP_CD'].size
print("Directly affected ZIP codes in Texas:", directly_affected_zip)

# Merge the GeoDataFrame of Texas ZIP and direct zip info
texas_gd_zip = texas_gd_zip.merge(closures_tx_by_zip, left_on = 'ZCTA5',
 ↪  right_on = 'TX_ZIP_CD', how = 'left')
texas_gd_zip['NUM_OF_CLSR'] = texas_gd_zip['NUM_OF_CLSR'].fillna(0)
print(texas_gd_zip.head())
```

```
Directly affected ZIP codes in Texas: 34
           GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
0  8600000US78624  78624  78624  ZCTA5     708.041
1  8600000US78626  78626  78626  ZCTA5      93.046
2  8600000US78628  78628  78628  ZCTA5      73.382
3  8600000US78631  78631  78631  ZCTA5     325.074
4  8600000US78632  78632  78632  ZCTA5      96.278

                                            geometry ZIP_CD  NUM_OF_HSPTL  \
0  POLYGON ((503432.37 3374023.84, 503439.279 337...  78624           1.0
1  POLYGON ((633347.155 3382976.525, 632638.265 3...  78626           1.0
2  POLYGON ((625348.227 3382811.117, 625348.353 3...      0           0.0
3  POLYGON ((487456.881 3359299.162, 487445.337 3...      0           0.0
4  POLYGON ((653782.474 3293172.762, 653782.628 3...      0           0.0

  TX_ZIP_CD  NUM_OF_CLSR
```

```
0       NaN          0.0
1       NaN          0.0
2       NaN          0.0
3       NaN          0.0
4       NaN          0.0
```

```
# Plot the Choropleth
fig, ax_2 = plt.subplots(1, 1, figsize=(10, 10))
texas_gd_zip.plot(column = 'NUM_OF_CLSR', cmap = 'YlGnBu', linewidth = 0.2,
 ↪  ax = ax_2, edgecolor = '0.5', legend = True, legend_kwds={'label':
 ↪  "Number of Closures", 'ticks': [0, 1]})
ax_2.set_title("Number of Hospital Closures by ZIP Code in Texas
 ↪  (2016-2019)")
ax_2.set_axis_off()

plt.show()
```



3.

```
# Step1: Create a GeoDataFrame of the directly affected zip codes
directly_affected_gdf = texas_gd_zip[texas_gd_zip['NUM_OF_CLSR'] > 0].copy()

# Step2: Create a 10-mile buffer around them
```

```
buffered_zips = directly_affected_gdf.copy()
buffered_zips['geometry'] = buffered_zips.geometry.buffer(1609.34 * 10)

# Step3: Spatial join with the overall Texas zip code shapefile
indirectly_affected_gdf = gpd.sjoin(texas_gd_zip, buffered_zips, how="inner",
 ↪  predicate="intersects")
print(indirectly_affected_gdf.head())

# Remove direct ZIPs, just keeping indirect ZIPs
indirectly_affected_gdf = indirectly_affected_gdf[

     ↪  ~indirectly_affected_gdf['ZCTA5_left'].isin(directly_affected_gdf['ZCTA5'])
]

# Count the number of unique indirectly affected ZIP codes
indirectly_affected_zip_count =
 ↪  indirectly_affected_gdf['ZCTA5_left'].nunique()
print("Indirectly affected ZIP codes in Texas:",
 ↪  indirectly_affected_zip_count)
```

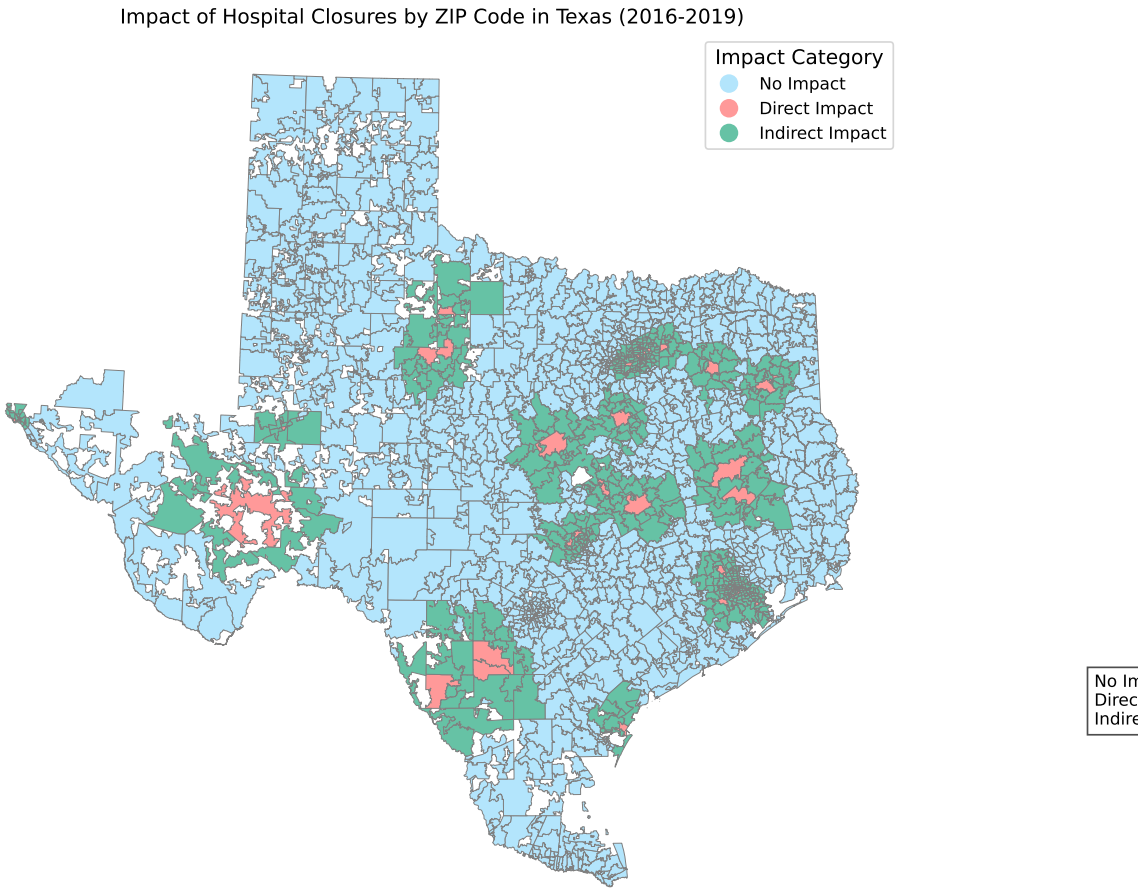
```
       GEO_ID_left ZCTA5_left NAME_left LSAD_left  CENSUSAREA_left  \
1   8600000US78626      78626     78626     ZCTA5           93.046
2   8600000US78628      78628     78628     ZCTA5           73.382
5   8600000US78633      78633     78633     ZCTA5           82.269
6   8600000US78634      78634     78634     ZCTA5           63.656
12  8600000US78641      78641     78641     ZCTA5          126.472

                                             geometry ZIP_CD_left  \
1   POLYGON ((633347.155 3382976.525, 632638.265 3...       78626
2   POLYGON ((625348.227 3382811.117, 625348.353 3...           0
5   POLYGON ((612164.247 3396525.074, 612111.995 3...           0
6   POLYGON ((634185.219 3378739.774, 634185.334 3...           0
12  POLYGON ((608214.847 3367889.82, 608203.041 33...           0

    NUM_OF_HSPTL_left TX_ZIP_CD_left  NUM_OF_CLSR_left  index_right  \
1                 1.0            NaN               0.0          862
2                 0.0            NaN               0.0          862
5                 0.0            NaN               0.0          862
6                 0.0            NaN               0.0          862
12                0.0            NaN               0.0           33

      GEO_ID_right ZCTA5_right NAME_right LSAD_right  CENSUSAREA_right  \
```

```
1   8600000US78613       78613      78613      ZCTA5            28.088
2   8600000US78613       78613      78613      ZCTA5            28.088
5   8600000US78613       78613      78613      ZCTA5            28.088
6   8600000US78613       78613      78613      ZCTA5            28.088
12  8600000US78734       78734      78734      ZCTA5            20.038

   ZIP_CD_right  NUM_OF_HSPTL_right TX_ZIP_CD_right  NUM_OF_CLSR_right
1         78613                 2.0          78613                1.0
2         78613                 2.0          78613                1.0
5         78613                 2.0          78613                1.0
6         78613                 2.0          78613                1.0
12        78734                 1.0          78734                1.0
Indirectly affected ZIP codes in Texas: 576
```

4.

```
# Step1: Make categories for not, directly, and indirectly affected
# Assign '0' for zip codes not affected
texas_gd_zip['impact_category'] = 0

# Assign '1' for zip codes directly affected
texas_gd_zip.loc[texas_gd_zip['NUM_OF_CLSR'] > 0, 'impact_category'] = 1

# Assign '2' for zip codes indirectly affected
texas_gd_zip.loc[texas_gd_zip['ZCTA5'].isin(indirectly_affected_gdf['ZCTA5_left']),
 ↪  'impact_category'] = 2

# Step2: Plot choropleth
# Color setting
cmap_4 = mcolors.ListedColormap(['#b3e5fc', '#ff9999', '#66c2a5'])
bounds_4 = [0, 1, 2, 3]
norm = mcolors.BoundaryNorm(bounds_4, cmap_4.N)

# Plot choropleth
fig, ax_4 = plt.subplots(1, 1, figsize=(10, 10))
texas_gd_zip.plot(column = 'impact_category', cmap = cmap_4, linewidth = 0.2,
 ↪  ax = ax_4, edgecolor = '0.5', legend = False, norm = norm)
ax_4.set_title("Impact of Hospital Closures by ZIP Code in Texas
 ↪  (2016-2019)")
ax_4.set_axis_off()

# Legend setting
legend_labels = ['No Impact', 'Direct Impact', 'Indirect Impact']
```

```
for color, label in zip(['#b3e5fc', '#ff9999', '#66c2a5'], legend_labels):
    ax_4.plot([], [], color = color, marker = 'o', linestyle = '', markersize
 ↪  = 10, label = label)
ax_4.legend(loc = "upper right", title = "Impact Category", fontsize = 10,
 ↪  title_fontsize = 12)

# Attribution; Ask ChatGPT how to display the statistical data info
no_impact = texas_gd_zip[texas_gd_zip['impact_category'] == 0].shape[0]
direct_impact = texas_gd_zip[texas_gd_zip['impact_category'] == 1].shape[0]
indirect_impact = texas_gd_zip[texas_gd_zip['impact_category'] == 2].shape[0]
stats_text = f"No Impact: {no_impact}\nDirect Impact:
 ↪  {direct_impact}\nIndirect Impact: {indirect_impact}"
plt.text(0.02, 0.02, stats_text, transform=ax.transAxes, fontsize=10,
 ↪  verticalalignment='bottom', bbox=dict(facecolor='white', alpha=0.7))

plt.show()
```



Reflecting on the exercise (10 pts)

1. The method may have several issues such as 1)Delayed Count Changes: Mergers or acquisitions might not immediately update in data, making a closed hospital seem active, 2)New Hospitals Masking Closures: New facilities in the same ZIP code can hide closures, keeping the total count stable. For better estimation, tracking termination code across years, instead of relying solely on active status, can reveal true satus of hospitals. In fact,the code 'PGM_TRMNUN_CD' has detailed values indicating current status of hospitals. In addition, comparing multi-year data, rather than examining closures year-by-year, will also show us how the steady status of hospitals.

2. Enhancing the current method involves using more comprehensive data sources, incorporating travel time and population metrics, assessing service availability, and considering the needs of vulnerable populations to provide a more accurate and thorough understanding of hospital closures' impacts.