Problem Set 4

Zijing Zhao & Zac(Zekai) Shen

PS4: Due Sat Nov 2 at 5:00PM Central. Worth 100 points.

Style Points (10 pts)

Submission Steps (10 pts)

Partner 1 (name and cnet ID): Zijing Zhao (zijingz) Partner 2 (name and cnet ID): Zac Shen (zekaishen)

"This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: Zijing Zhao, Zac Shen

Late coins used this pset: 1 Late coins left after submission: 1

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

```
import pandas as pd
import matplotlib.pyplot as plt
import geopandas as gpd
import warnings
# Suppress all warnings
warnings.filterwarnings("ignore")
```

1. Download this for 2016 and call it pos2016.csv. What are the variables you pulled? The Variables that we pulled: PRVDR_CTGRY_SBTYP_CD - Provider Category Subtype Code PRVDR_CTGRY_CD - Provider Category Code SSA_CNTY_CD - SSA County Code CRTFCTN_DT - Certification Date FAC_NAME - Facility Name PRVDR_NUM - CMS Certification Number RGN_CD - Region Code STATE_CD - State Abbreviation ST_ADR - Street Address PGM_TRMNTN_CD - Termination Code GNRL_CNTL_TYPE_CD - General Control Type Code ZIP_CD - ZIP Code

2. Import your pos2016.csv file. We want to focus on short-term hospitals. These are identified as facilities with provider type code 01 and subtype code 01. Subset your data to these facilities. a.How many hospitals are reported in this data? Does this number make sense? b.Cross-reference with other sources and cite the number you compared it to. If it differs, why do you think it could differ?

```
pos_2016 =
 ↪  pd.read_csv("/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2016.csv")

# Filter the data for short-term hospitals
short_term_2016 = pos_2016[(pos_2016['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos_2016['PRVDR_CTGRY_SBTYP_CD'] == 1)]

# Count the number of short-term hospitals
num_short_term_hospitals_2016 = short_term_2016.shape[0]
print(num_short_term_hospitals_2016)
```

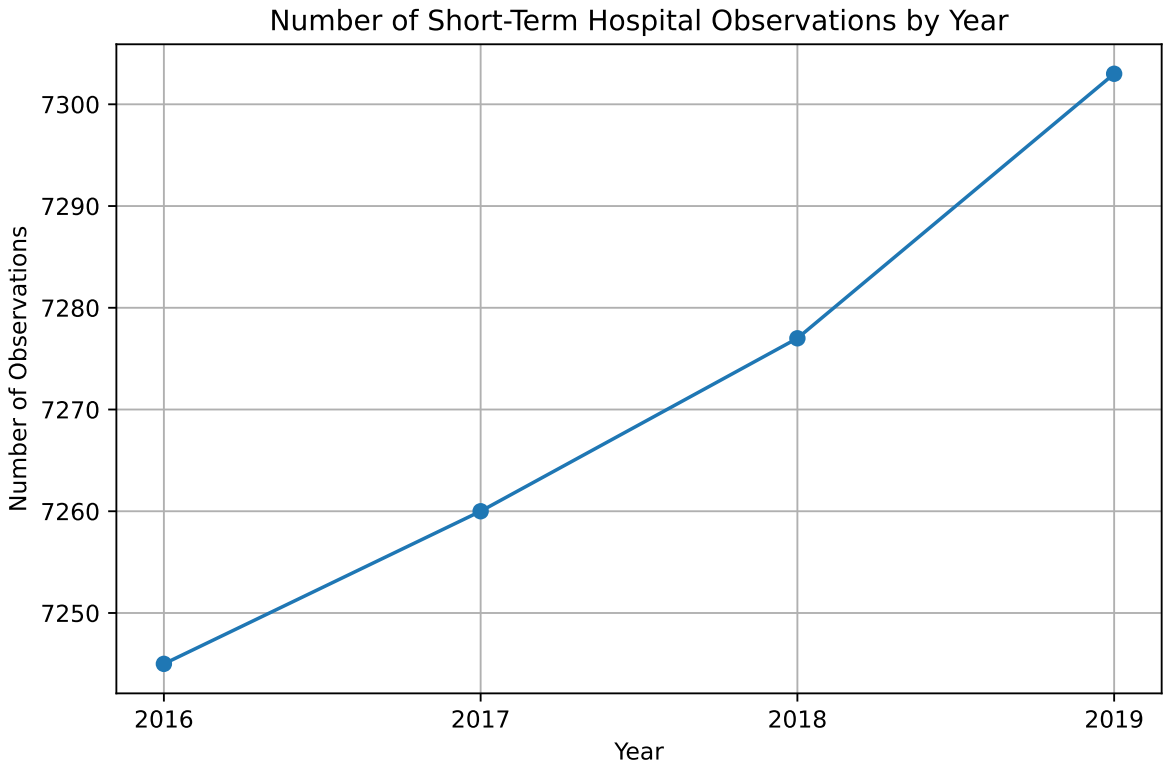
```
7245
```

This dataset reports a total of 7,245 short-term hospitals. We referred to the 4,840 community hospitals reported by the American Hospital Association (AHA) for 2016. (website: https://www.aha.org/system/files/2018-02/2018-aha-hospital-fast-facts.pdf). The number we have is notably higher than the AHA report. There are possible causes to the discrepancies: Definition Differences: The AHA defines community hospitals as nonfederal, short-term general, and other special hospitals accessible to the public. If our dataset includes additional categories—such as federal hospitals, specialty hospitals, or facilities not open to the general public—this would increase the total count. In addition, our dataset might encompass facilities beyond those considered community hospitals by the AHA, including certain specialty or non-general acute care hospitals. Lastly, our dataset may contain duplicates, outdated entries, or facilities that were closed or reclassified, leading to an inflated count.

3. Repeat the previous 3 steps with 2017Q4, 2018Q4, and 2019Q4 and then append them together. Plot the number of observations in your dataset by year.

```
pos_2017 =
 ↪  pd.read_csv("/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2017.csv")
pos_2018 =
 ↪  pd.read_csv("/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2018.csv",
 ↪  encoding='ISO-8859-1')
pos_2019 =
 ↪  pd.read_csv("/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2019.csv",
 ↪  encoding='ISO-8859-1')
```

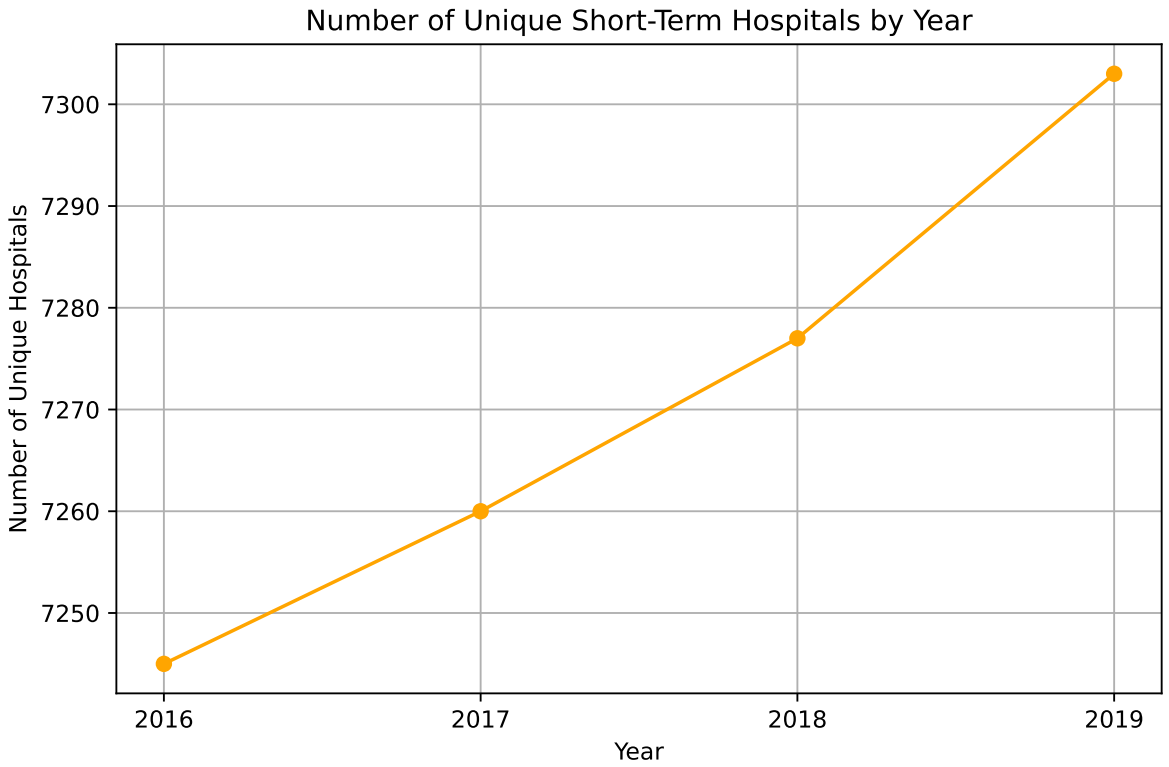
```
short_term_2016['Year'] = 2016

short_term_2017 = pos_2017[(pos_2017['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos_2017['PRVDR_CTGRY_SBTYP_CD'] == 1)]
short_term_2017['Year'] = 2017

short_term_2018 = pos_2018[(pos_2018['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos_2018['PRVDR_CTGRY_SBTYP_CD'] == 1)]
short_term_2018['Year'] = 2018

short_term_2019 = pos_2019[(pos_2019['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos_2019['PRVDR_CTGRY_SBTYP_CD'] == 1)]
short_term_2019['Year'] = 2019

# Append the datasets together
short_term_merge = pd.concat([short_term_2016, short_term_2017,
 ↪  short_term_2018, short_term_2019], ignore_index=True)

# Plot the number of observations by year
obs_by_year = short_term_merge['Year'].value_counts().sort_index()
plt.figure(figsize=(8, 5))
plt.plot(obs_by_year.index, obs_by_year.values, marker='o')
plt.xlabel('Year')
plt.ylabel('Number of Observations')
plt.title('Number of Short-Term Hospital Observations by Year')
plt.xticks(obs_by_year.index)
plt.grid(True)
plt.show()
```



4. Each hospital is identified by its CMS certification number.

    a. Plot the number of unique hospitals in your dataset per year.
    b. Compare this to your plot in the previous step. What does this tell you about the structure of the data?

```
unique_hospitals = short_term_merge.groupby('Year')['PRVDR_NUM'].nunique()

# Plot the number of unique hospitals per year
plt.figure(figsize=(8, 5))
plt.plot(unique_hospitals.index, unique_hospitals.values, marker='o',
 ↪  color='orange')
plt.xlabel('Year')
plt.ylabel('Number of Unique Hospitals')
plt.title('Number of Unique Short-Term Hospitals by Year')
plt.xticks(unique_hospitals.index)
plt.grid(True)
plt.show()
```



Comparing this plot to the previous plot of total observations per year, the unique hospital count aligns closely with the overall number of records each year, indicating that each observation generally represents a unique facility rather than multiple entries per hospital. The slight yearly increases suggest either new hospitals opening or previously unlisted hospitals being added to the dataset over time. This points to a stable dataset structure where each entry likely represents an individual hospital's record without significant duplication

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

1. Use this definition to create a list of all hospitals that were active in 2016 that were suspected to have closed by 2019. Record the facility name and zip of each hospital as well as the year of suspected closure (when they become terminated or disappear from the data). How many hospitals are there that fit this definition?

```
import pandas as pd

# Load data for 2016 to 2019
pos2016 =
 ↪  pd.read_csv('/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2016.csv',
 ↪  encoding='ISO-8859-1')
```

```
pos2017 =
 ↪  pd.read_csv('/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2017.csv',
 ↪  encoding='ISO-8859-1')
pos2018 =
 ↪  pd.read_csv('/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2018.csv',
 ↪  encoding='ISO-8859-1')
pos2019 =
 ↪  pd.read_csv('/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/pos2019.csv',
 ↪  encoding='ISO-8859-1')

# Filter short-term hospitals
short_term_2016 = pos2016[(pos2016['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2016['PRVDR_CTGRY_SBTYP_CD'] == 1)]
short_term_2017 = pos2017[(pos2017['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2017['PRVDR_CTGRY_SBTYP_CD'] == 1)]
short_term_2018 = pos2018[(pos2018['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2018['PRVDR_CTGRY_SBTYP_CD'] == 1)]
short_term_2019 = pos2019[(pos2019['PRVDR_CTGRY_CD'] == 1) &
 ↪  (pos2019['PRVDR_CTGRY_SBTYP_CD'] == 1)]

# Create a dataframe for active hospital in 2016
active_2016 = short_term_2016[pos2016['PGM_TRMNTN_CD'] == 0]

# Merge active hospitals with subsequent years
merged_2017 = active_2016.merge(short_term_2017[['PRVDR_NUM',
 ↪  'PGM_TRMNTN_CD']], on='PRVDR_NUM', how='left', suffixes=('', '_2017'))
merged_2018 = active_2016.merge(short_term_2018[['PRVDR_NUM',
 ↪  'PGM_TRMNTN_CD']], on='PRVDR_NUM', how='left', suffixes=('', '_2018'))
merged_2019 = active_2016.merge(short_term_2019[['PRVDR_NUM',
 ↪  'PGM_TRMNTN_CD']], on='PRVDR_NUM', how='left', suffixes=('', '_2019'))

# Identify closures
closures_2017 = merged_2017[(merged_2017['PGM_TRMNTN_CD_2017'] != 0) |
 ↪  merged_2017['PGM_TRMNTN_CD_2017'].isna()]
closures_2017['Year_Closed'] = 2017

closures_2018 = merged_2018[(merged_2018['PGM_TRMNTN_CD_2018'] != 0) |
 ↪  merged_2018['PGM_TRMNTN_CD_2018'].isna()]
closures_2018['Year_Closed'] = 2018

closures_2019 = merged_2019[(merged_2019['PGM_TRMNTN_CD_2019'] != 0) |
 ↪  merged_2019['PGM_TRMNTN_CD_2019'].isna()]
```

```
closures_2019['Year_Closed'] = 2019

# Combine all closures
closures = pd.concat([closures_2017, closures_2018,
 ↪  closures_2019]).drop_duplicates(subset=['PRVDR_NUM'])

# Count the number of closures
num_closures = closures.shape[0]

# Output the results
print(f"Number of hospitals suspected to have closed by 2019:
 ↪  {num_closures}")
print(closures[['FAC_NAME', 'ZIP_CD', 'Year_Closed']])
```

```
Number of hospitals suspected to have closed by 2019: 174
                                                FAC_NAME   ZIP_CD  Year_Closed
93                                ABRAZO MARYVALE CAMPUS  85031.0         2017
299            ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230.0         2017
381                          FALLBROOK HOSPITAL DISTRICT  92028.0         2017
528              ARKANSAS VALLEY REGIONAL MEDICAL CENTER  81050.0         2017
529                               KEEFE MEMORIAL HOSPITAL  80810.0         2017
...                                                  ...      ...          ...
3396                              TEXAS GENERAL HOSPITAL  75051.0         2019
3398  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613.0         2019
3404            LITTLE RIVER HEALTHCARE CAMERON HOSPITAL  76520.0         2019
3407                     BAYLOR EMERGENCY MEDICAL CENTER  75087.0         2019
3421                   TEXAS GENERAL HOSPITAL- VZRMC LP  75140.0         2019

[174 rows x 3 columns]
```

2. Sort this list of hospitals by name and report the names and year of suspected closure for the first 10 rows.

```
# Sort the closures by facility name
sorted_closures = closures.sort_values(by='FAC_NAME')

# Report the names and year of suspected closure for the first 10 rows
first_10_closures = sorted_closures[['FAC_NAME', 'Year_Closed']].head(10)

print("First 10 hospitals suspected to have closed by name and year of
 ↪  closure:")
print(first_10_closures)
```

```
First 10 hospitals suspected to have closed by name and year of closure:
                                          FAC_NAME  Year_Closed
93                            ABRAZO MARYVALE CAMPUS         2017
299       ADVENTIST MEDICAL CENTER - CENTRAL VALLEY         2017
2276                         AFFINITY MEDICAL CENTER         2018
2019  ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS         2017
2955       ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE         2017
1682                         ALLIANCE LAIRD HOSPITAL         2019
2345                       ALLIANCEHEALTH DEACONESS         2019
720                   ANNE BATES LEACH EYE HOSPITAL         2019
528          ARKANSAS VALLEY REGIONAL MEDICAL CENTER         2017
1801             BANNER CHURCHILL COMMUNITY HOSPITAL         2017
```

3. However, not all suspected hospital closures are true closures. For example, in the case of a merger, a CMS certification number will be appear to be "terminated," but then the hospital re-appear under a similar name/address with a new CMS certifi- cation number in the next year. As a first pass to address this, remove any suspected hospital closures that are in zip codes where the number of active hospitals does not decrease in the year after the suspected closure.

    a. Among the suspected closures, how many hospitals fit this definition of potentially being a merger/acquisition?

    b.After correcting for this, how many hospitals do you have left?

    c.Sort this list of corrected hospital closures by name and report the first 10 rows.

```python
# Count active hospitals by zip code for each year
active_by_zip_2016 = short_term_2016[short_term_2016['PGM_TRMNTN_CD'] ==
 ↪  0].groupby('ZIP_CD').size()
active_by_zip_2017 = short_term_2017[short_term_2017['PGM_TRMNTN_CD'] ==
 ↪  0].groupby('ZIP_CD').size()
active_by_zip_2018 = short_term_2018[short_term_2018['PGM_TRMNTN_CD'] ==
 ↪  0].groupby('ZIP_CD').size()
active_by_zip_2019 = short_term_2019[short_term_2019['PGM_TRMNTN_CD'] ==
 ↪  0].groupby('ZIP_CD').size()

# Identify potential mergers/acquisitions
potential_mergers = []
for index, row in closures.iterrows():
    zip_code = row['ZIP_CD']
    year_closed = row['Year_Closed']

    if year_closed == 2017:
```

```
        if active_by_zip_2017.get(zip_code, 0) <=
         ↪  active_by_zip_2018.get(zip_code, 0):
            potential_mergers.append(index)
    elif year_closed == 2018:
        if active_by_zip_2018.get(zip_code, 0) <=
         ↪  active_by_zip_2019.get(zip_code, 0):
            potential_mergers.append(index)
    elif year_closed == 2019:
        if active_by_zip_2019.get(zip_code, 0) <=
         ↪  active_by_zip_2019.get(zip_code, 0):
            potential_mergers.append(index)

# Remove potential mergers/acquisitions
corrected_closures = closures.drop(index=potential_mergers)

# Sort the corrected closures by facility name
sorted_corrected_closures = corrected_closures.sort_values(by='FAC_NAME')

# Report the first 10 rows
first_10_corrected_closures = sorted_corrected_closures[['FAC_NAME',
 ↪  'Year_Closed']].head(10)

# Output the results
num_potential_mergers = len(potential_mergers)
num_corrected_closures = corrected_closures.shape[0]

print(f"Number of hospitals potentially being a merger/acquisition:
 ↪  {num_potential_mergers}")
print(f"Number of hospitals left after correction: {num_corrected_closures}")
print("First 10 hospitals suspected to have closed by name and year of
 ↪  closure after correction:")
print(first_10_corrected_closures)
```

```
Number of hospitals potentially being a merger/acquisition: 173
Number of hospitals left after correction: 1
First 10 hospitals suspected to have closed by name and year of closure after
correction:
                      FAC_NAME  Year_Closed
797  TRINITY HOSPITAL OF AUGUSTA         2018
```

Download Census zip code shapefile (10 pt)

1. This is non-tabular data.

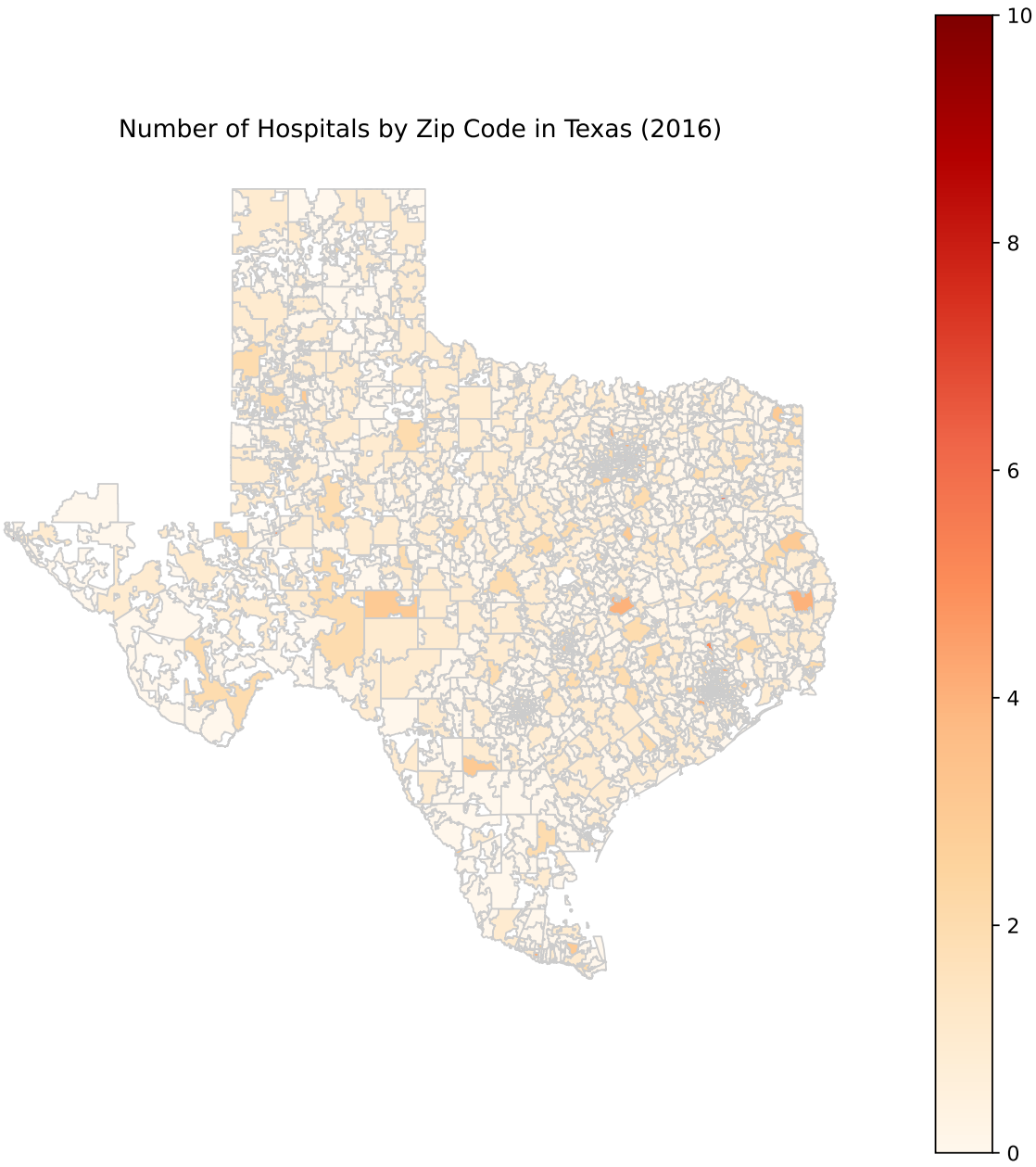
    a. Shapefile (.shp) contains the geometric information for each shape, such as points, lines, or polygons. This is the core file that stores vector data. Shapefile index (.shx) enables quick access to records in the .shp file. It's a positional index of the geometry to allow faster searching and loading of data. .dbf file contains attribute data in tabular format for each shape in the .shp file. It stores metadata, such as names, IDs, and other information associated with each shape. .prj is the projection file describes the coordinate system and projection information for the shapefile. This is essential for accurately placing the data on a map. .xml is an optional metadata file that provides additional descriptive information about the dataset, such as creation date, source, and field descriptions.

    b: gz_2010_us_860_00_500k.shp is the largest file at 837.5 MB, as it contains the primary vector data gz_2010_us_860_00_500k.dbf is next in size at 6.4 MB, as it holds the attribute data. gz_2010_us_860_00_500k.shx is smaller, at 265 KB, storing the index. gz_2010_us_860_00_500k.prj 165 bytes and gz_2010_us_860_00_500k.xml 16 KB, are relatively small, since they store text-based projection and metadata.

2. Load the zip code shapefile and restrict to Texas zip codes. (Hint: you can identify which state a zip code is in using the first 2-3 numbers in the zip code (Wikipedia link). Then calculate the number of hospitals per zip code in 2016 based on the cleaned POS file from the previous step. Plot a choropleth of the number of hospitals by zip code in Texas.)

```
zip_codes_gdf =
 ↪  gpd.read_file("/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/gz_2010_us_860_0

# Filter for Texas zip codes
texas_zip_codes = zip_codes_gdf[zip_codes_gdf['ZCTA5'].str.startswith(('75',
 ↪  '76', '77','78','79'))]

# Load the POS data for 2016 and filter to Texas
pos_2016_texas = short_term_2016.copy()
pos_2016_texas['ZIP_CD'] = pos_2016_texas['ZIP_CD'].astype(str).str[:5]
pos_2016_texas =
 ↪  pos_2016_texas[pos_2016_texas['ZIP_CD'].str.startswith(('75', '76',
 ↪  '77','78','79'))]

# Calculate the number of hospitals per zip code in Texas for 2016
hospitals_per_zip =
 ↪  pos_2016_texas.groupby('ZIP_CD').size().reset_index(name='hospital_count')
```

```
# Merge hospital counts with the Texas zip codes GeoDataFrame
texas_zip_codes = texas_zip_codes.rename(columns={'ZCTA5': 'ZIP_CD'})
texas_zip_codes = texas_zip_codes.merge(hospitals_per_zip, on='ZIP_CD',
 ↪  how='left').fillna(0)

# Plot a choropleth of the number of hospitals by zip code in Texas
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_zip_codes.plot(column='hospital_count', cmap='OrRd', linewidth=0.8,
 ↪  ax=ax, edgecolor='0.8', legend=True)
ax.set_title("Number of Hospitals by Zip Code in Texas (2016)")
ax.set_axis_off()
plt.show()
```



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

1. Create a GeoDataFrame for the centroid of each zip code nationally: zips_all_centroids. What are the dimensions of the resulting GeoDataFrame and what do each of the columns mean?

```
# Create a GeoDataFrame with the centroids of each zip code
zips_all_centroids = zip_codes_gdf.copy()
zips_all_centroids['geometry'] = zips_all_centroids['geometry'].centroid

# Display the dimensions and column names of the resulting GeoDataFrame
print(zips_all_centroids.head())
print("Dimensions of zips_all_centroids:", zips_all_centroids.shape)
print("Column names:", zips_all_centroids.columns)
```

```
           GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA
           geometry
0  8600000US01040  01040  01040  ZCTA5      21.281  POINT (-72.64107
42.21257)
1  8600000US01050  01050  01050  ZCTA5      38.329  POINT (-72.86985
42.28786)
2  8600000US01053  01053  01053  ZCTA5       5.131  POINT (-72.71162
42.35349)
3  8600000US01056  01056  01056  ZCTA5      27.205  POINT (-72.45805
42.19215)
4  8600000US01057  01057  01057  ZCTA5      44.907  POINT (-72.32430
42.09165)
Dimensions of zips_all_centroids: (33120, 6)
Column names: Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA',
'geometry'], dtype='object')
```

Dimensions: The shape property returns a tuple representing the number of rows (zip codes) and columns. Column meanings: ZCTA5 contains the 5-digit ZIP Code geometry: tells us it is a centroid point of the polygen, representing the center of each zip code polygon. GEO_ID: Name: name of the zip code polygen. LSAD County (ZCTA5) Censusarea: land area

2. Create two GeoDataFrames as subsets of zips_all_centroids. First, create all zip codes in Texas: zips_texas_centroids. Then, create all zip codes in Texas or a bordering state: zips_texas_borderstates_centroids, using the zip code prefixes to make these subsets. How many unique zip codes are in each of these subsets?

```
# Zip code prefixes for Texas and bordering states (Oklahoma, Arkansas,
 ↪  Louisiana, New Mexico)
texas_prefixes = ('75', '76', '77','78','79')
border_states_prefixes = ('70','71', '72', '73', '74','75', '76',
 ↪  '77','78','79','87', '88')

# Create the subset for Texas zip codes
```

```
zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(texas_prefixes)]

# Create the subset for Texas and bordering states zip codes
zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(border_states_prefixes)]

# Get the count of unique zip codes in each subset
num_texas_zips = zips_texas_centroids['ZCTA5'].nunique()
num_border_states_zips = zips_texas_borderstates_centroids['ZCTA5'].nunique()

print(num_texas_zips)
print(num_border_states_zips)
```

```
1935
4057
```

3. Then create a subset of zips_texas_borderstates_centroids that con- tains only the zip codes with at least 1 hospital in 2016. Call the resulting Geo- DataFrame zips_withhospital_centroids What kind of merge did you decide to do, and what variable are you merging on?

```
# Filter hospital data in 2016 and count hospitals by ZIP_CD

hospitals_2016 =
 ↪  pos_2016.groupby('ZIP_CD').size().reset_index(name='hospital_count')
hospitals_2016=hospitals_2016[pos2016['PGM_TRMNTN_CD'] == 0]
hospitals_2016['ZIP_CD'] =
 ↪  hospitals_2016['ZIP_CD'].astype(str).str.split('.').str[0].str.zfill(5)

# Rename the column in zips_texas_borderstates_centroids to match
 ↪  hospitals_2016_texas_border
zips_texas_borderstates_centroids =
 ↪  zips_texas_borderstates_centroids.rename(columns={'ZCTA5': 'ZIP_CD'})

# Perform a left merge on ZIP_CD to include only zip codes with at least one
 ↪  hospital
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    hospitals_2016, on='ZIP_CD', how='inner'
)
```

```
# Display dimensions of the resulting GeoDataFrame
zips_withhospital_centroids.shape
```

```
(1232, 7)
```

I used 'inner' merge so that it ensures for every zipcode, it satisfies both having at least one hospital and in texas or borderline states.

4. For each zip code in zips_texas_centroids, calculate the distance to the nearest zip code with at least one hospital in zips_withhospital_centroids. a.This is a computationally-intensive join. Before attempting to do the entire join, subset to 10 zip codes in zips_texas_centroids and try the join. How long did it take? Approximately how long do you estimate the entire procedure will take?

```
from shapely.ops import nearest_points
import time
# Subset to 10 zip codes in zips_texas_centroids
subset_zips_texas_centroids = zips_texas_centroids.head(10)

# Function to find the nearest point and calculate the distance
def calculate_nearest(row, other_gdf, point_column='geometry',
 ↪  other_point_column='geometry'):
    nearest_geom = nearest_points(row[point_column],
 ↪  other_gdf.unary_union)[1]
    distance = row[point_column].distance(nearest_geom)
    return distance

# Measure the time taken for the subset
start_time = time.time()

# Apply the function to calculate the distance for the subset
subset_zips_texas_centroids['nearest_hospital_distance'] =
 ↪  subset_zips_texas_centroids.apply(
    calculate_nearest, other_gdf=zips_withhospital_centroids, axis=1
)

end_time = time.time()
subset_time_taken = end_time - start_time
print(f'The time taken for calculating with the subset is
 ↪  {subset_time_taken:.2f}')
print(f'The estimate time for calculating the entire dataset is
 ↪  {(subset_time_taken/10)*len(zips_texas_centroids):.2f}')
```

```
The time taken for calculating with the subset is 0.01
The estimate time for calculating the entire dataset is 2.14
```

b.Now try doing the full calculation and time how long it takes. How close is it to your estimation?

```
# Measure the time taken for the full calculation
start_time = time.time()
# Apply the function to calculate the distance for the entire dataset
zips_texas_centroids['nearest_hospital_distance'] =
 ↪  zips_texas_centroids.apply(
    calculate_nearest, other_gdf=zips_withhospital_centroids, axis=1
)

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

# Print the time taken for the full calculation
print(f"Time taken for the full calculation: {full_time_taken:.2f} seconds")
```

```
Time taken for the full calculation: 1.92 seconds
```

The actual time taken is shorter than expected. Probably because the initialization of calculation won't multiply.

c.Look into the .prj file and report which unit it is in. Convert the given unit to miles, using an appropriate conversion you find online (estimates are okay).

```
from pyproj import CRS
file_path_prj =
 ↪  r'/Users/shenzekai/Documents/GitHub/problem-set-4-zac-tina/gz_2010_us_860_00_500k.prj'

with open(file_path_prj, 'r') as f:
    prj_wkt = f.read()

crs = CRS.from_wkt(prj_wkt)

print("WKT Projection:", crs.to_wkt())
```

```
WKT Projection: GEOGCRS["NAD83",DATUM["North American Datum
1983",ELLIPSOID["GRS
1980",6378137,298.257222101,LENGTHUNIT["metre",1]],ID["EPSG",6269]],PRIMEM["Greenwich",0,ANG
```

The unit it uses is degree. One degree equals to 69 miles.

5. Calculate the average distance to the nearest hospital for each zip code in Texas.

   a.What unit is this in?

```
print(zips_texas_centroids.crs)
```

```
EPSG:4269
```

```
It uses degrees.
```

```
b. Report the average distance in miles. Does this value make sense?
```

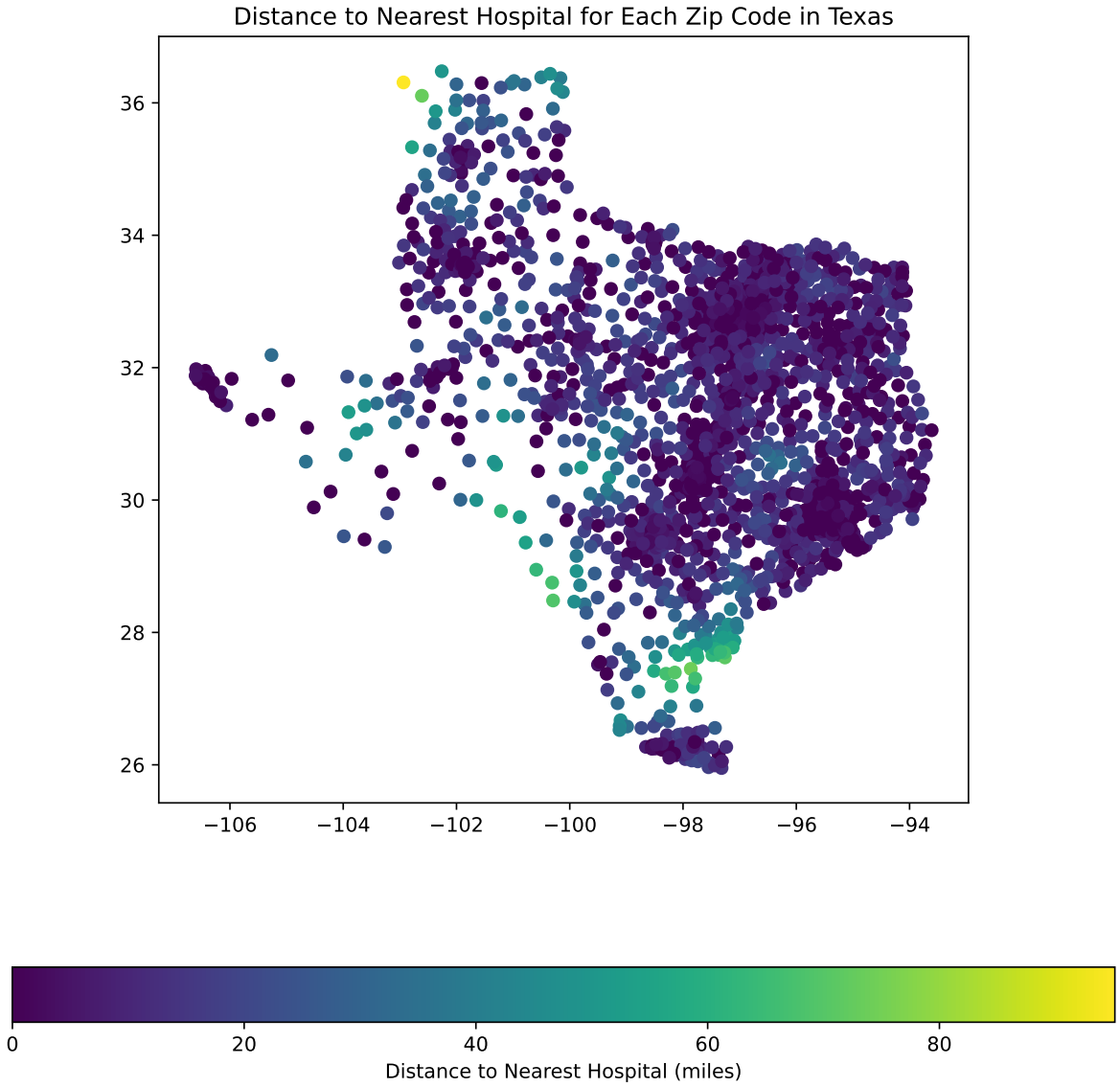
```
zips_texas_centroids['nearest_hospital_distance_miles'] =
 ↪  zips_texas_centroids['nearest_hospital_distance'] * 69
# Calculate the average distance in miles
average_distance_miles =
 ↪  zips_texas_centroids['nearest_hospital_distance_miles'].mean()
print(f"Average distance to the nearest hospital:
 ↪  {average_distance_miles:.2f} miles")
```

```
Average distance to the nearest hospital: 10.05 miles
```

```
The average distnce to the nearest hospital in Texas is 10 miles. It make
sense as it took around 20 min car drive to a hospital.
```

```
c. Map the value for each zip code.
```

```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
zips_texas_centroids.plot(column='nearest_hospital_distance_miles', ax=ax,
 ↪  legend=True,
                          legend_kwds={'label': "Distance to Nearest Hospital
 ↪  (miles)",
                                       'orientation': "horizontal"},
                          cmap='viridis')
plt.title("Distance to Nearest Hospital for Each Zip Code in Texas")
plt.show()
```



Effects of closures on access in Texas (15 pts)

1. Using the corrected hospital closures dataset from the first section, create a list of directly affected zip codes in Texas – that is, those with at least one closure in 2016-2019. Display a table of the number of closures by zipcode

```
closures['ZIP_CD'] =
 ↪  closures['ZIP_CD'].astype(str).str.split('.').str[0].str.zfill(5)
closures_texas = closures[closures['ZIP_CD'].str.startswith(('75', '76',
 ↪  '77','78','79'))]
```

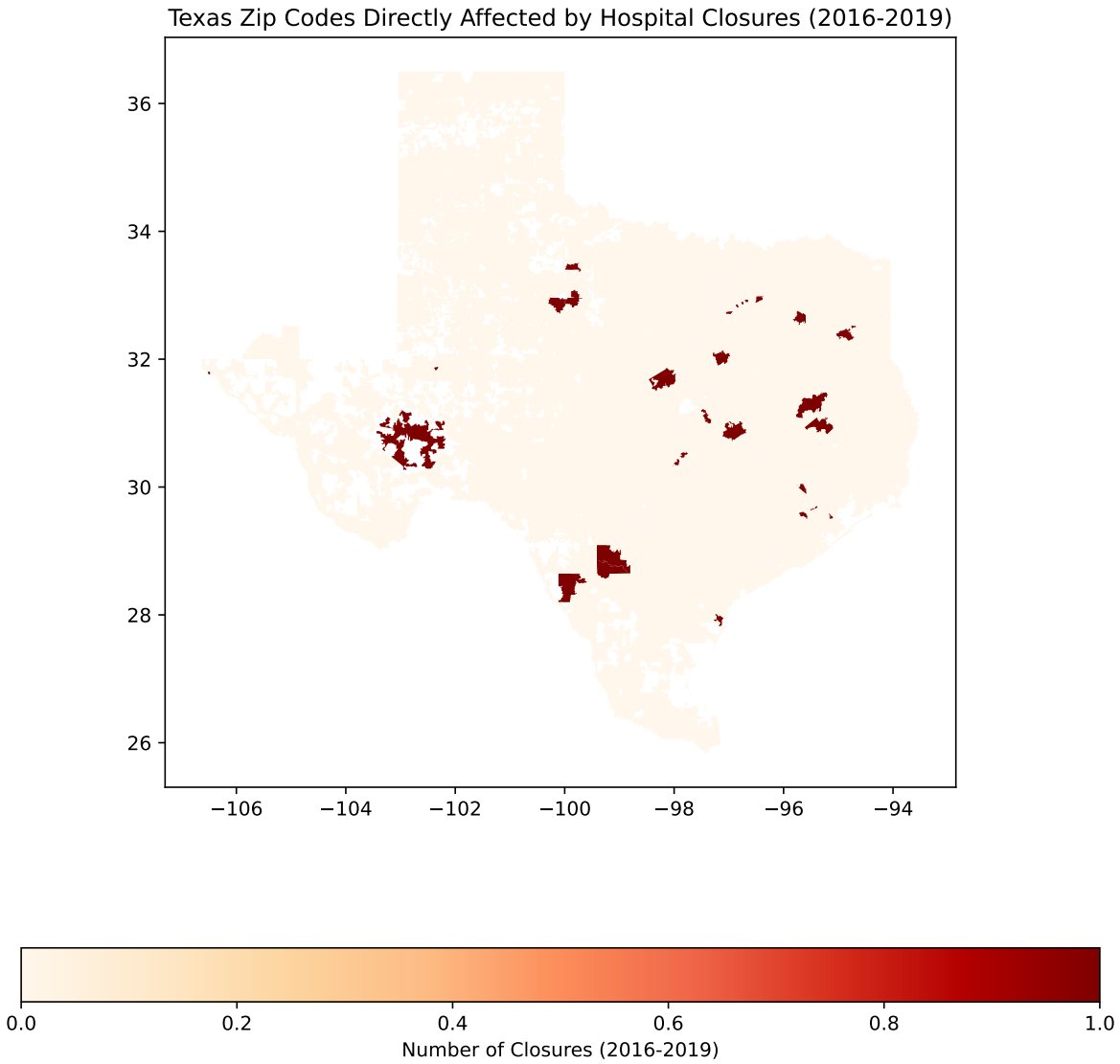
```
# Group by zip code and count the number of closures
TX_closures_by_zip =
 ↪  closures_texas.groupby('ZIP_CD').size().reset_index(name='closure_count')


# Display the table of the number of closures by zip code
print(TX_closures_by_zip)
```

```
   ZIP_CD  closure_count
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
23  78336              1
24  78613              1
25  78734              1
26  78834              1
27  79520              1
28  79529              1
29  79553              1
30  79735              1
31  79761              1
```

```
32  79902                1
```

2. Plot a choropleth of which Texas zip codes were directly affected by a closure in 2016-2019 – there was at least one closure within the zip code. How many directly affected zip codes are there in Texas?

```
texas_zip_codes['ZIP_CD'] = texas_zip_codes['ZIP_CD'].astype(str)

# Merge the closure data with the Texas zip code shapefile
texas_zip_closure = texas_zip_codes.merge(TX_closures_by_zip, on='ZIP_CD',
 ↪  how='left')

# Fill NaN values in closure_count with 0
texas_zip_closure['closure_count'] =
 ↪  texas_zip_closure['closure_count'].fillna(0)

# Plot the choropleth
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_zip_closure.plot(column='closure_count', ax=ax, legend=True,
                   legend_kwds={'label': "Number of Closures (2016-2019)",
                                'orientation': "horizontal"},
                   cmap='OrRd')
plt.title("Texas Zip Codes Directly Affected by Hospital Closures
 ↪  (2016-2019)")
plt.show()
```



3. Then identify all the indirectly affected zip codes: Texas zip codes within a 10-mile radius of the directly affected zip codes. To do so, first create a GeoDataFrame of the directly affected zip codes. Then create a 10-mile buffer around them. Then, do a spatial join with the overall Texas zip code shapefile. How many indirectly affected zip codes are there in Texas?

```
from shapely.geometry import Point

print(len(texas_zip_closure))
# Create a GeoDataFrame of the directly affected zip codes
```

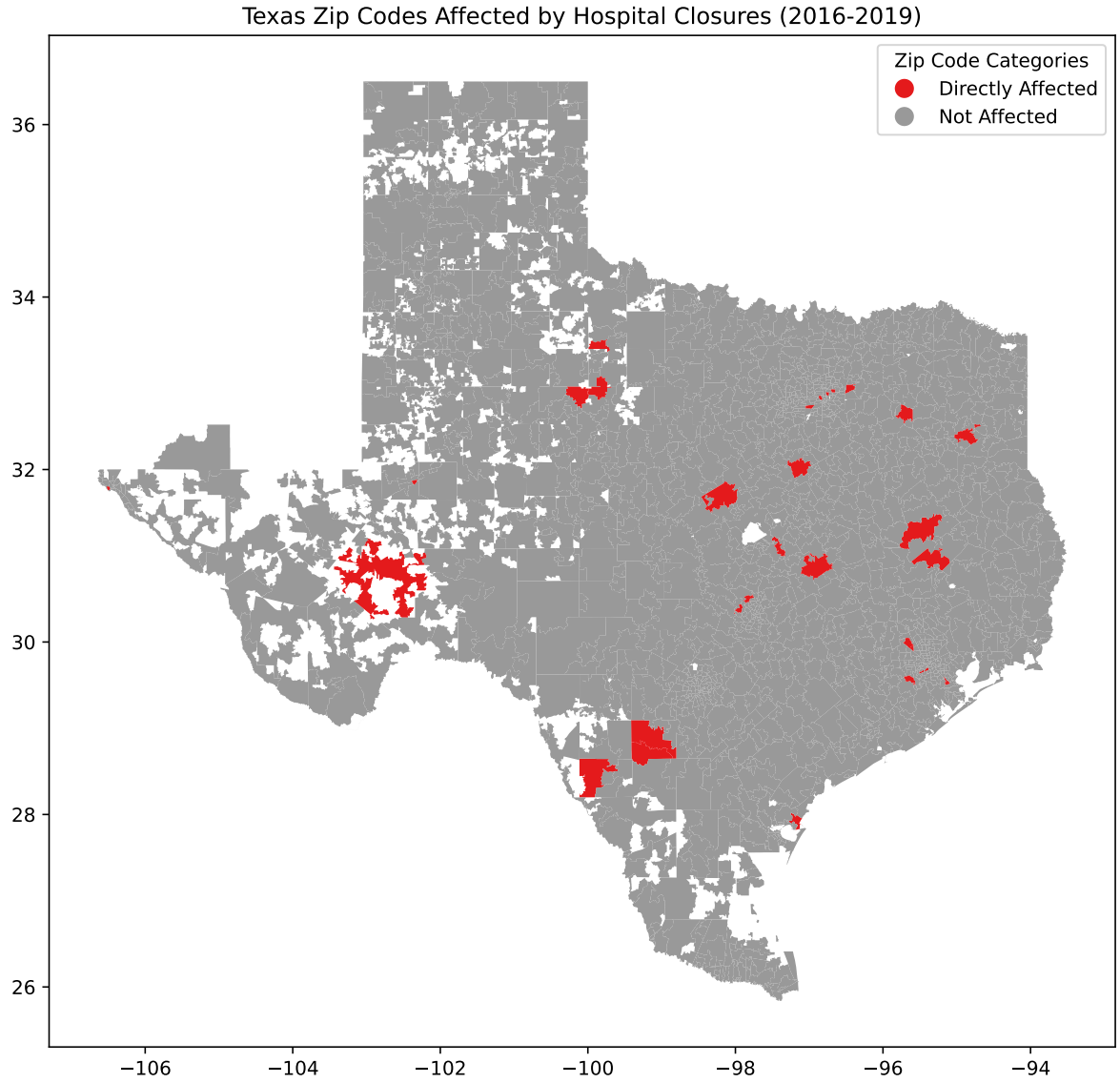
```
directly_affected_gdf = texas_zip_closure[texas_zip_closure['closure_count']
 ↪  > 0]
directly_affected_gdf.crs = "EPSG:4269"
directly_affected_gdf = directly_affected_gdf.to_crs(epsg=3857)

# Create a 10-mile buffer around the directly affected zip codes
# Convert miles to meters (1 mile = 1609.34 meters)
buffer_distance = 10 * 1609.34
buffered = directly_affected_gdf.buffer(buffer_distance)
buffered_gdf = gpd.GeoDataFrame(geometry=buffered)
buffered_gdf.crs = directly_affected_gdf.crs

# Perform a spatial join with the overall Texas zip code shapefile
indirectly_affected_gdf = gpd.sjoin(texas_zip_closure, buffered_gdf,
 ↪  how='inner', predicate='intersects')


# Remove duplicates to get unique indirectly affected zip codes
indirectly_affected_gdf =
 ↪  indirectly_affected_gdf.drop_duplicates(subset='ZIP_CD_left')

# Count the number of indirectly affected zip codes
num_indirectly_affected = len(indirectly_affected_gdf)
print(f"Number of indirectly affected zip codes: {num_indirectly_affected}")
```

```
1935
Number of indirectly affected zip codes: 0
```

4. Make a choropleth plot of the Texas zip codes with a different color for each of the 3 categories: directly affected by a closure, within 10 miles of closure but not directly affected, or not affected.

```
# Create a new column to categorize each zip code
texas_zip_closure['category'] = 'Not Affected'
texas_zip_closure.loc[texas_zip_closure['ZIP_CD'].isin(directly_affected_gdf['ZIP_CD']),
 ↪  'category'] = 'Directly Affected'
texas_zip_closure.loc[texas_zip_closure['ZIP_CD'].isin(indirectly_affected_gdf['ZIP_CD']),
 ↪  'category'] = 'Indirectly Affected'

# Plot the choropleth
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_zip_closure.plot(column='category', ax=ax, legend=True,
```

```
                        legend_kwds={'title': "Zip Code Categories"},
                        cmap='Set1')
plt.title("Texas Zip Codes Affected by Hospital Closures (2016-2019)")
plt.show()
```



Reflecting on the exercise (10 pts)

Partner1: The "first-pass" method we're using to address incorrectly identified clo- sures in the data is imperfect. Can you think of some potential issues that could arise still and ways to do a better job at confirming hospital closures?

Potential Issues: The unchanged total number of hospital in the zipcode area could be the result of a new established hospital. Improvements: Use cross-reference to look at if there are datasets that have recorded whether its a merger or closure.

Partner2: Consider the way we are identifying zip codes affected by closures. How well does this reflect changes in zip-code-level access to hospitals? Can you think of some ways to improve this measure?

It reflects the access of hospital influenced by hospital closures. However, it does not reflect more localized access problems (eg. population density). To improve, indexes such as hospitals per capita could be introduced to better evaluate the hospital access.