Your Title

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): Serhat Seflek serhat@uchicago.edu
    - Partner 2 (name and cnet ID): Zixuan Lan zixuanlan@uchicago.edu
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: SS and LZ
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: **__** Late coins left after submission: **__**
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

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

1.

Variable	Definition
PRVDR_CTGRY_SBTYP_CD	Identifies the subtype of the provider, within the primary category. Used to show the breakdown of provider categories, mainly for hospitals and SNFs.
PRVDR_CTGRY_CD	Identifies the type of provider participating in the Medicare/Medicaid program.
FAC_NAME	Name of the provider certified to participate in the Medicare and/or Medicaid programs.
PRVDR_NUM	Six or ten position identification number assigned to a certified provider (CMS Certification Number).
PGM_TRMNTN_CD	Indicates the current termination status for the provider.
ZIP_CD	Five-digit ZIP code for a provider's physical address.

2.

(a)

```
import pandas as pd
file_path = (r'C:\\Users\\shefo\\GitHub\\problem-set-4-serhat-ian\\pos2016.csv')
df = pd.read_csv(file_path)
pos2016  = df[(df["PRVDR_CTGRY_SBTYP_CD"]==1) & (df["PRVDR_CTGRY_CD"]==1)]
```

The number of short term hospital in the filtered data.

```
len(pos2016)
```

```
7245
```

(b)

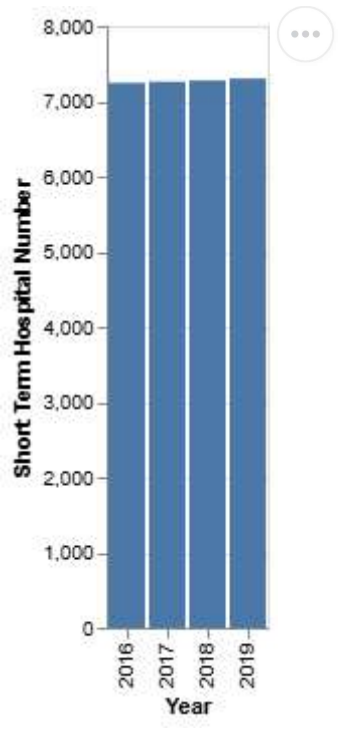
There are 7245 short term hospital in the data. However, the article provided in the problem set claims there were 5000. This difference could be arised from the authors" choice which type of short term hospital they used in their research or in other words, how they classified the short term hospitals.

3.

```
file_path = (r"C:\Users\shefo\GitHub\problem-set-4-serhat-ian\pos2017.csv")
df = pd.read_csv(file_path)
pos2017  = df[(df["PRVDR_CTGRY_SBTYP_CD"]==1) & (df["PRVDR_CTGRY_CD"]==1)]

file_path = (r"C:\Users\shefo\GitHub\problem-set-4-serhat-ian\pos2018.csv")
df = pd.read_csv(file_path)
pos2018  = df[(df["PRVDR_CTGRY_SBTYP_CD"]==1) & (df["PRVDR_CTGRY_CD"]==1)]

file_path = (r"C:\Users\shefo\GitHub\problem-set-4-serhat-ian\pos2019.csv")
df = pd.read_csv(file_path)
pos2019 = df[(df["PRVDR_CTGRY_SBTYP_CD"]==1) & (df["PRVDR_CTGRY_CD"]==1)]

pos2016["Year"] = 2016
pos2017["Year"] = 2017
pos2018["Year"] = 2018
pos2019["Year"] = 2019

pos2016["ZIP_CD"] = pos2016["ZIP_CD"].astype(int).astype(str)
pos2017["ZIP_CD"] = pos2017["ZIP_CD"].astype(int).astype(str)
pos2018["ZIP_CD"] = pos2018["ZIP_CD"].astype(int).astype(str)
pos2019["ZIP_CD"] = pos2019["ZIP_CD"].astype(int).astype(str)


combined_df = pd.concat([pos2016, pos2017, pos2018, pos2019], ignore_index=True)
```

```
import altair as alt

num_hospitals_per_year = combined_df.groupby("Year").size().reset_index(name="Count")
print(num_hospitals_per_year)


alt.Chart(num_hospitals_per_year).mark_bar().encode(
  x=alt.X("Year:O", title="Year"),
  y=alt.Y("Count:Q", title="Short Term Hospital Number")
)
```

```
   Year  Count
0  2016   7245
1  2017   7260
2  2018   7277
3  2019   7303
```

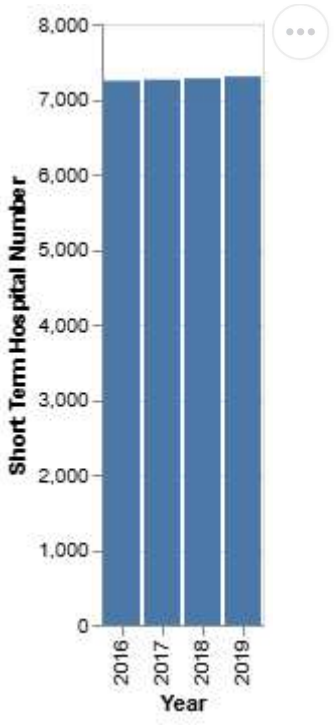


4.

(a)

```
unique_hospitals_per_year = combined_df.groupby("Year")["PRVDR_NUM"].nunique().reset_index(name="Unique_Hospitals
print(unique_hospitals_per_year)
alt.Chart(unique_hospitals_per_year).mark_bar().encode(
    x=alt.X("Year:O", title="Year"),
    y=alt.Y("Unique_Hospitals:Q", title="Short Term Hospital Number")
   )
```

```
   Year  Unique_Hospitals
0  2016              7245
1  2017              7260
2  2018              7277
3  2019              7303
```



(b)

The graphs and numbers are identical. It suggests the hospitals are unique with their provider number

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

1

```
active_2016 = pos2016[pos2016['PGM_TRMNTN_CD'] == 0]
active_2017 = pos2017[pos2017['PGM_TRMNTN_CD'] == 0]
active_2018 = pos2018[pos2018['PGM_TRMNTN_CD'] == 0]
active_2019 = pos2019[pos2019['PGM_TRMNTN_CD'] == 0]

def provider_in_year(df, provider_num):
    row = df[df['PRVDR_NUM'] == provider_num]
    return not row.empty and row['PGM_TRMNTN_CD'].values[0] == 0


def determine_closure_year(provider_num):

    if not provider_in_year(active_2017, provider_num):
        return 2017
    elif not provider_in_year(active_2018, provider_num):
        return 2018
    elif not provider_in_year(active_2019, provider_num):
        return 2019
    return None

active_2016_only = active_2016[active_2016['PGM_TRMNTN_CD'] == 0]


active_2016_only['Year_Closed'] = active_2016_only['PRVDR_NUM'].apply(determine_closure_year)


closed_hospitals = active_2016_only.dropna(subset=['Year_Closed']).reset_index(drop=True)


print(f"Number of suspected hospital closures: {len(closed_hospitals)}")
```

```
Number of suspected hospital closures: 747
```

2

```
print(closed_hospitals[['FAC_NAME', 'PRVDR_NUM', 'ZIP_CD', 'Year_Closed']].sort_values(by="FAC_NAME").head(10))
```

```
                                FAC_NAME PRVDR_NUM ZIP_CD  Year_Closed
132                  ABRAZO ARROWHEAD CAMPUS    030094  85308       2018.0
```

106	ABRAZO CENTRAL CAMPUS	030030	85015	2018.0
93	ABRAZO MARYVALE CAMPUS	030001	85031	2017.0
124	ABRAZO SCOTTSDALE CAMPUS	030083	85032	2018.0
138	ABRAZO WEST CAMPUS	030110	85395	2018.0
263	ADVENTIST MEDICAL CENTER	050121	93230	2018.0
299	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	050196	93230	2017.0
296	ADVENTIST MEDICAL CENTER - REEDLEY	050192	93654	2018.0
670	AFFINITY MEDICAL CENTER	360151	44646	2018.0
307	AHMC ANAHEIM REGIONAL MEDICAL CENTER	050226	92801	2018.0

3.

```
active_2016 = pos2016[(pos2016["PRVDR_CTGRY_CD"] == 1) & (pos2016["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2016["PGM_T
active_2017 = pos2017[(pos2017["PRVDR_CTGRY_CD"] == 1) & (pos2017["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2017["PGM_T
active_2018 = pos2018[(pos2018["PRVDR_CTGRY_CD"] == 1) & (pos2018["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2018["PGM_T
active_2019 = pos2019[(pos2019["PRVDR_CTGRY_CD"] == 1) & (pos2019["PRVDR_CTGRY_SBTYP_CD"] == 1) & (pos2019["PGM_T

closed_in_2017 = active_2016[active_2016["PRVDR_NUM"].isin(active_2017["PRVDR_NUM"]) == False].copy()
closed_in_2017["Year_Closed"] = 2017

closed_in_2018 = active_2016[(active_2016["PRVDR_NUM"].isin(active_2017["PRVDR_NUM"]) == True) &
                             (active_2016["PRVDR_NUM"].isin(active_2018["PRVDR_NUM"]) == False)].copy()
closed_in_2018["Year_Closed"] = 2018

closed_in_2019 = active_2016[(active_2016["PRVDR_NUM"].isin(active_2018["PRVDR_NUM"]) == True) &
                             (active_2016["PRVDR_NUM"].isin(active_2019["PRVDR_NUM"]) == False)].copy()
closed_in_2019["Year_Closed"] = 2019


closures_df = pd.concat([closed_in_2017, closed_in_2018, closed_in_2019], ignore_index=True)


def is_potential_merger(row, active_hospitals_by_year):
    zip_code = row["ZIP_CD"]
    year_closed = row["Year_Closed"]
    next_year = year_closed + 1


    if str(next_year) in active_hospitals_by_year:

        current_count = active_hospitals_by_year[str(year_closed)].loc[active_hospitals_by_year[str(year_closed)]
        next_count = active_hospitals_by_year[str(next_year)].loc[active_hospitals_by_year[str(next_year)]["ZIP_C


        return next_count >= current_count
    return False


active_hospitals_by_year = {
    "2016": active_2016,
    "2017": active_2017,
    "2018": active_2018,
    "2019": active_2019
}


closures_df["Is_Merger"] = closures_df.apply(is_potential_merger, axis=1, args=(active_hospitals_by_year,))
potential_mergers_df = closures_df[closures_df["Is_Merger"] == True]
confirmed_closures_df = closures_df[closures_df["Is_Merger"] == False]
```

(a)

```
print("Number of potential mergers/acquisitions:", len(potential_mergers_df))
```

```
Number of potential mergers/acquisitions: 665
```

(b)

```
print("Number of confirmed closures:", len(confirmed_closures_df))
```

```
Number of confirmed closures: 82
```

(c)

```
print("Sample of confirmed closures:\n", confirmed_closures_df[["FAC_NAME", "ZIP_CD", "PRVDR_NUM", "Year_Closed"]
```

```
Sample of confirmed closures:
                                              FAC_NAME ZIP_CD PRVDR_NUM  \
42                       ELIZA COFFEE MEMORIAL HOSPITAL  35631    010006
55                                    WEDOWEE HOSPITAL  36278    010032
66                          GEORGIANA MEDICAL CENTER  36033    010047
218                      SPARKS REGIONAL MEDICAL CENTER  72902    040055
219                           MERCY HOSPITAL FORT SMITH  72917    040062
249                    PACIFIC ALLIANCE MEDICAL CENTER  90012    050018
257                  SANTA CLARA VALLEY MEDICAL CENTER  95128    050038
263  CALIFORNIA PACIFIC MEDICAL CTR-PACIFIC CAMPUS ...  94115    050047
278         KAISER FOUNDATION HOSPITAL - SAN FRANCISCO  94115    050076
324                                   O'CONNOR HOSPITAL  95128    050153

     Year_Closed
42          2018
55          2018
66          2018
218         2018
219         2018
249         2018
257         2018
263         2018
278         2018
324         2018
```

Download Census zip code shapefile (10 pt)

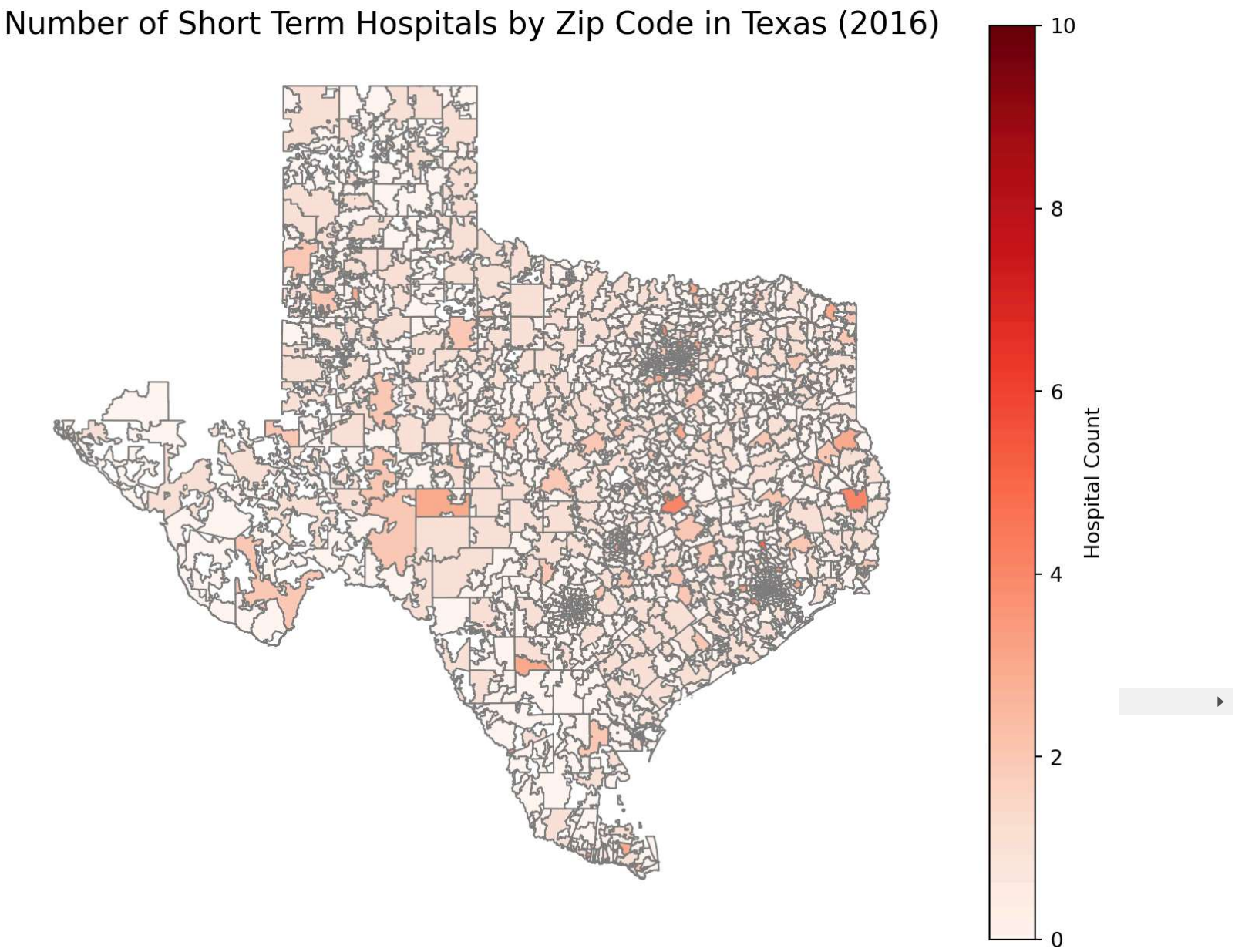
1.

File Type	Description	Typical File Size
`.dbf`	Stores tabular data with attributes for each feature.	Varies, but typically small (KBs to a few MBs).
`.prj`	Contains projection information for aligning the map.	Very small (usually a few KBs).
`.shp`	Holds geometry (points, lines, polygons) of the features.	Can be large (depends on data complexity, MBs to GBs).
`.shx`	An index file for quick access to shapes in the `.shp` file.	Small (usually a few KBs).
`.xml`	Metadata file with extra information about the dataset.	Small (a few KBs).

2.

```
import geopandas as gpd
import altair as alt
import matplotlib.pyplot as plt


shapefile_path = r"C:\Users\shefo\GitHub\problem-set-4-serhat-ian\gz_2010_us_860_00_500k.shp"
zip_gdf = gpd.read_file(shapefile_path)

zip_gdf["ZCTA5"] = zip_gdf["ZCTA5"].astype(str)


texas_zip_gdf = zip_gdf[zip_gdf["ZCTA5"].str.startswith(("75", "76", "77", "78", "79"))]


texas_hospitals = pos2016[pos2016["ZIP_CD"].str.startswith(("75", "76", "77", "78", "79"))]


hospital_counts = texas_hospitals.groupby("ZIP_CD").size().reset_index(name="Hospital_Count")



merged_gdf = texas_zip_gdf.merge(hospital_counts, left_on="ZCTA5", right_on="ZIP_CD", how="left")

merged_gdf["Hospital_Count"] = merged_gdf["Hospital_Count"].fillna(0)
```

```
#I got a very weird map with so many blank areas. So I asked to ChatGPT what was the problem. It said this proble

fig, ax = plt.subplots(figsize=(10, 8))

merged_gdf.plot(
    column="Hospital_Count",
    cmap="Reds",
    linewidth=0.8,
    edgecolor="gray",
    legend=True,
    legend_kwds={"label": "Hospital Count"},
    ax=ax
)

ax.set_title("Number of Short Term Hospitals by Zip Code in Texas (2016)", fontsize=15)
ax.axis("off")

plt.show()
```



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

1.

```
from shapely.geometry import shape, Point
import time
import fiona

zip_features = []

#My partner changed his path to my path, because I'm partner one and it should work on my computer too.
with fiona.open(r"C:\Users\shefo\GitHub\problem-set-4-serhat-ian\gz_2010_us_860_00_500k.shp") as src:
    for feature in src:
        zip_code = feature['properties']['ZCTA5']
        geometry = shape(feature['geometry'])
        centroid = geometry.centroid
        zip_features.append({"ZIP_CODE": zip_code, "geometry": geometry, "centroid": centroid})

zips_all_centroids = gpd.GeoDataFrame(zip_features, geometry="centroid")
```

```
print("Dimensions of zips_all_centroids:", zips_all_centroids.shape)
print("Columns in zips_all_centroids:", zips_all_centroids.columns)
```

```
Dimensions of zips_all_centroids: (33120, 3)
Columns in zips_all_centroids: Index(['ZIP_CODE', 'geometry', 'centroid'], dtype='object')
```

2.

```
texas_prefixes = ["75", "76", "77", "78", "79"]
#Oklahoma zip code prexies 73 and 74
#New Mexico 870-884
#Arkansas 716-729
#Lousiana 700-715
bordering_prefixes = texas_prefixes +  ["73", "74", "870", "871", "872", "873", "874", "875",
    "876", "877", "878", "879", "880", "881", "882", "883", "884",
    "700", "701", "702", "703", "704", "705", "706", "707", "708", "709",
    "710", "711", "712", "713", "714", "715", "716", "717", "718", "719",
    "720", "721", "722", "723", "724", "725", "726", "727", "728", "729"
]

zips_texas_centroids = zips_all_centroids[zips_all_centroids["ZIP_CODE"].str.startswith(tuple(texas_prefixes))]
print("Unique Texas ZIP codes:", zips_texas_centroids["ZIP_CODE"].nunique())
```

```
zips_texas_borderstates_centroids = zips_all_centroids[zips_all_centroids["ZIP_CODE"].str.startswith(tuple(border
print("Unique Texas + Bordering States ZIP codes:", zips_texas_borderstates_centroids["ZIP_CODE"].nunique())
```

```
Unique Texas ZIP codes: 1935
Unique Texas + Bordering States ZIP codes: 4057
```

3

```
zips_with_hospitals_2016 = active_2016["ZIP_CD"].unique()

zips_withhospital_centroids = zips_texas_borderstates_centroids[zips_texas_borderstates_centroids["ZIP_CODE"].isi
print("Dimensions of zips_withhospital_centroids:", zips_withhospital_centroids.shape)
```

```
Dimensions of zips_withhospital_centroids: (448, 3)
```

Explanation of Merge: We used an inner join (by filtering only those ZIPs that are in the hospital list) to identify ZIP codes with hospitals in 2016.

4.

```
zips_texas_centroids["nearest_hospital_distance"] = zips_texas_centroids["centroid"].apply(
    lambda x: zips_withhospital_centroids.distance(x).min())
```

(a)

```
subset_texas_centroids = zips_texas_centroids.iloc[:10]

# Timing the subset for distance calculations
start_time = time.time()
subset_texas_centroids["nearest_hospital_distance"] = subset_texas_centroids["centroid"].apply(
    lambda x: zips_withhospital_centroids.distance(x).min()
)
end_time = time.time()
subset_time = end_time - start_time
print("Time for 10 ZIP code calculations:", subset_time, "seconds")

# Estimate the time for the full procedure
estimated_time_full = subset_time * (len(zips_texas_centroids) / 10)
print("Estimated time for full procedure:", estimated_time_full, "seconds")
```

```
Time for 10 ZIP code calculations: 0.005631923675537109 seconds
Estimated time for full procedure: 1.0897772312164307 seconds
```

(b)

```
# Full distance calculation for all Texas ZIP codes
start_time = time.time()
zips_texas_centroids["nearest_hospital_distance"] = zips_texas_centroids["centroid"].apply(
    lambda x: zips_withhospital_centroids.distance(x).min()
)
end_time = time.time()
actual_time_full = end_time - start_time
print("Actual time for full calculation:", actual_time_full, "seconds")
```

```
Actual time for full calculation: 0.6089458465576172 seconds
```

(c)

```
#
from pyproj import CRS

# Read the .prj file content as a WKT string
with open(r"C:\Users\shefo\GitHub\problem-set-4-serhat-ian\gz_2010_us_860_00_500k.prj") as file:
    prj_contents = file.read()

print("Contents of the .prj file:")
print(prj_contents)
```

```
Contents of the .prj file:
GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],PRIMEM
["Greenwich",0],UNIT["Degree",0.017453292519943295]]
```

Looking the content of the .prj file, we can see the unit is degree. The conversion will be done with the information from this site:https://gis.stackexchange.com/questions/142326/calculating-longitude-length-in-miles

Texas longtitude will be taken approixametly as 30.

```
import math


def degrees_to_miles_longitude(distance_in_degrees, latitude=30):

    latitude_in_radians = latitude * 0.017453292519943295

    miles_per_degree_longitude = 69.172 * math.cos(latitude_in_radians)

    return distance_in_degrees * miles_per_degree_longitude
```

5.

```
average_distance_degrees = zips_texas_centroids["nearest_hospital_distance"].mean()
print("Average distance to nearest hospital (in degrees):", average_distance_degrees)
```

```
Average distance to nearest hospital (in degrees): 0.21101748566398393
```

(a)

This is in degrees.

(b)

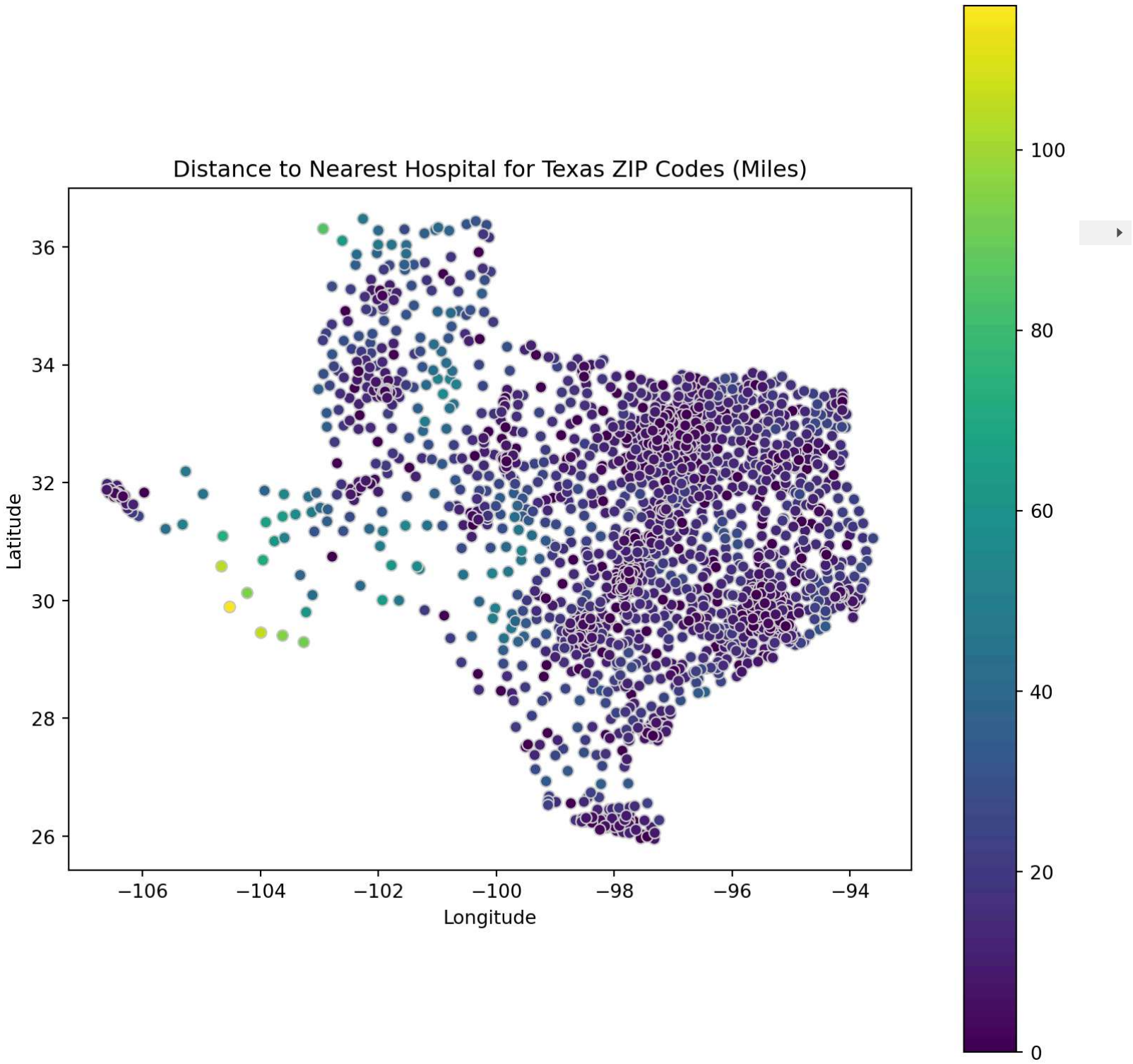
```
average_distance_miles = degrees_to_miles_longitude(average_distance_degrees, latitude=30)
print("Average distance to nearest hospital (in miles):", average_distance_miles)
```

```
Average distance to nearest hospital (in miles): 12.640941121268447
```

(c)

```
zips_texas_centroids["nearest_hospital_distance_miles"] = zips_texas_centroids["nearest_hospital_distance"].apply
    lambda x: degrees_to_miles_longitude(x, latitude=30)
)
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1, figsize=(10, 10))
zips_texas_centroids.plot(column="nearest_hospital_distance_miles", cmap="viridis", linewidth=0.8, ax=ax, edgecol
plt.title("Distance to Nearest Hospital for Texas ZIP Codes (Miles)")
plt.xlabel("Longitude")
```

```
plt.ylabel("Latitude")
plt.show()
```



Effects of closures on access in Texas (15 pts)

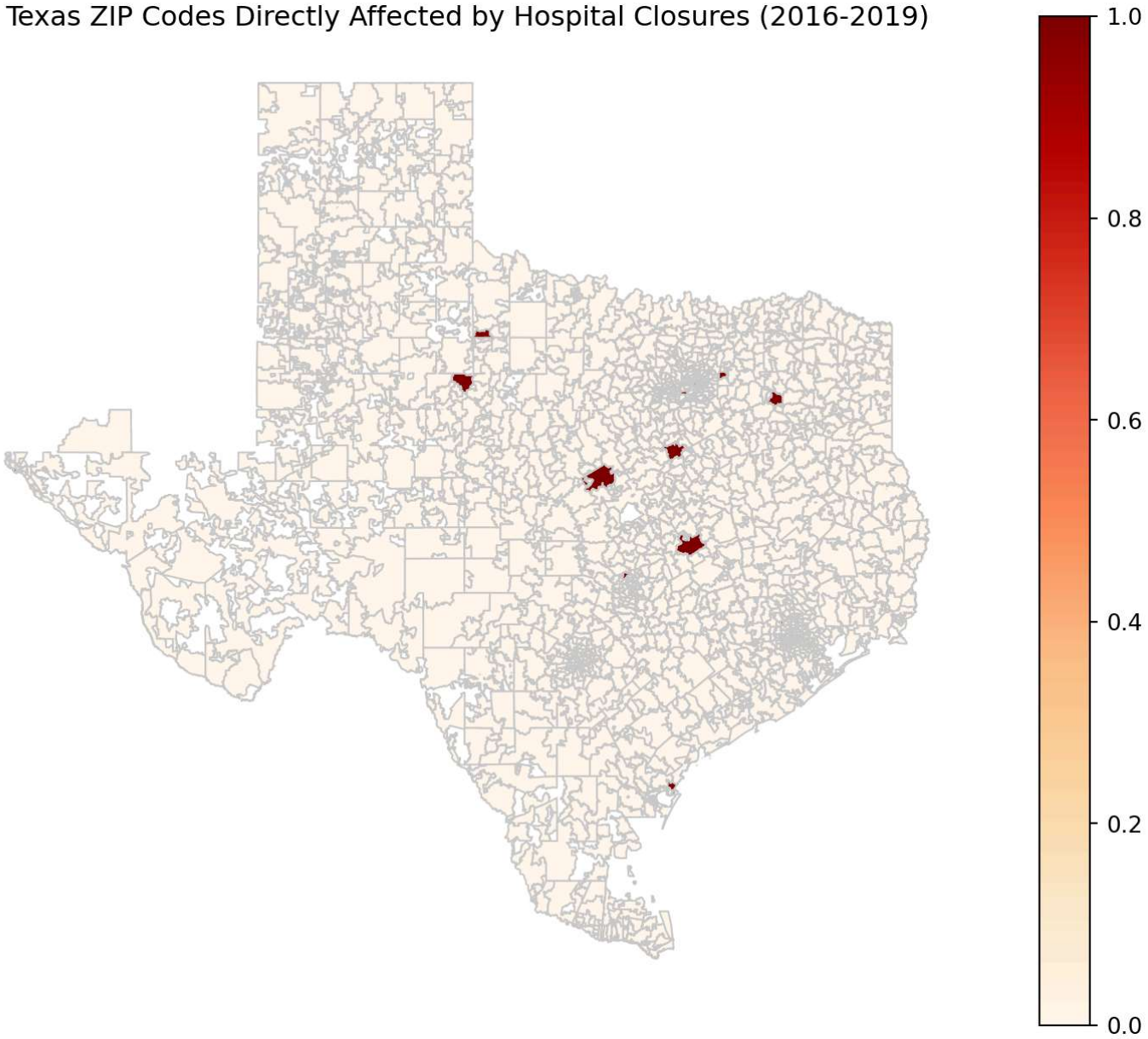
1.

```
texas_zip_prefixes = ["75", "76", "77", "78", "79"]

texas_closures_df = confirmed_closures_df[confirmed_closures_df["ZIP_CD"].astype(str).str[:2].isin(texas_zip_pref

closure_counts = texas_closures_df.groupby('ZIP_CD').size().reset_index(name='Closure_Count')

print(closure_counts)
```

	ZIP_CD	Closure_Count
0	75051	1
1	75087	1
2	75140	1
3	75235	1
4	75390	1
5	76520	1
6	76531	1
7	76645	1
8	77065	1
9	78336	1
10	78613	1
11	79520	1
12	79529	1
13	79902	1

2.

```
zip_gdf["ZCTA5"] = zip_gdf["ZCTA5"].astype(str)
texas_zip_prefixes = ["75", "76", "77", "78", "79"]
texas_zip_gdf = zip_gdf[zip_gdf["ZCTA5"].str.startswith(tuple(texas_zip_prefixes))]


texas_closure_geo = texas_zip_gdf.merge(closure_counts, left_on="ZCTA5", right_on = "ZIP_CD", how="left")
texas_closure_geo['Closure_Count'] = texas_closure_geo['Closure_Count'].fillna(0)

fig, ax = plt.subplots(1, 1, figsize=(10, 8))

texas_closure_geo.plot(
    column='Closure_Count',
    cmap='OrRd',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.8',
    legend=True
)
ax.set_title("Texas ZIP Codes Directly Affected by Hospital Closures (2016-2019)")
ax.set_axis_off()
plt.show()
```



3.

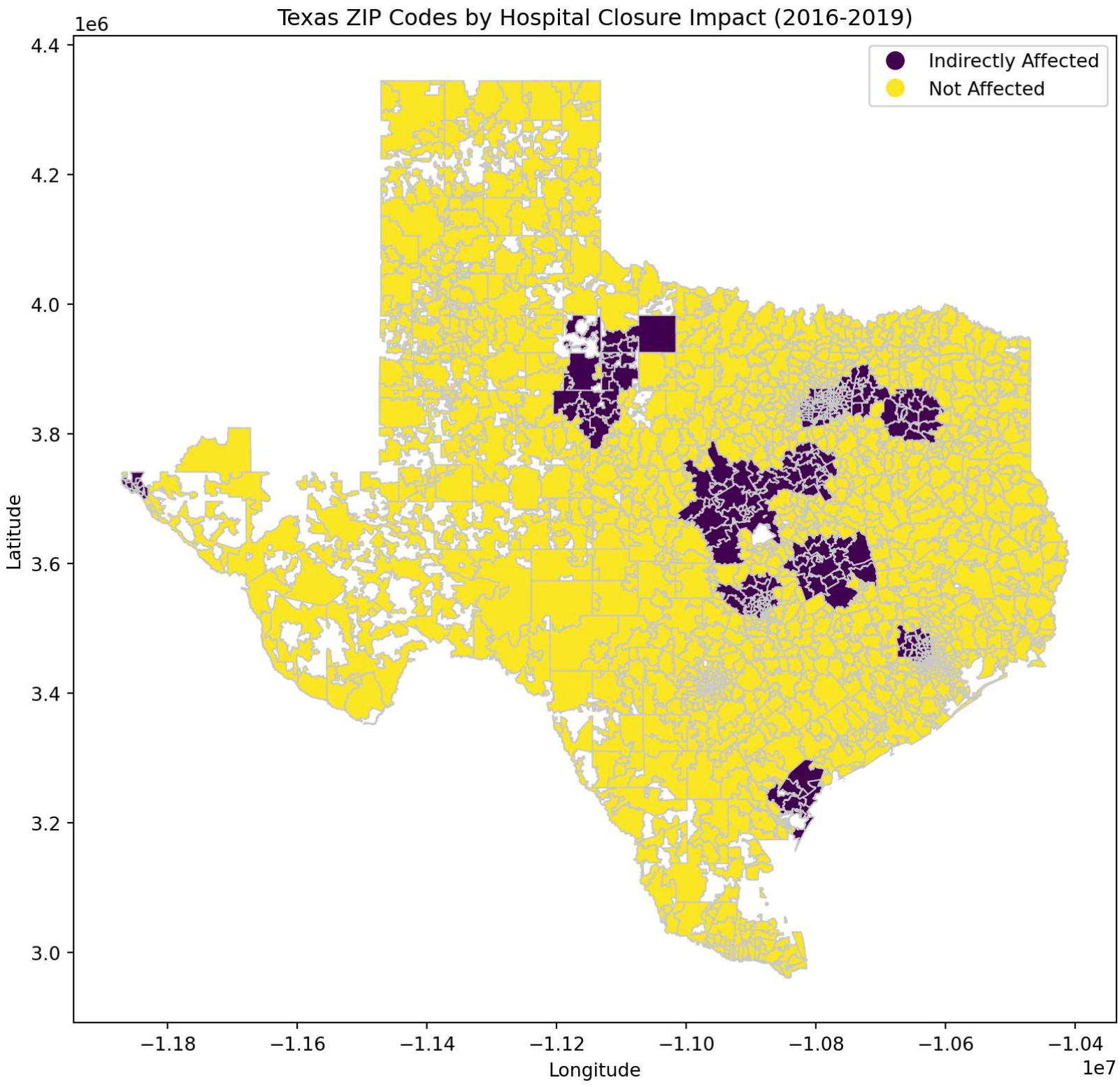
```
texas_closure_geo = texas_closure_geo.to_crs(epsg=3395)  # Using EPSG:3395 as an example (meters)
texas_zip_gdf = texas_zip_gdf.to_crs(texas_closure_geo.crs)

# Step 1: Filter directly affected ZIP codes (those with Closure_Count > 0)
directly_affected_zips = texas_closure_geo[texas_closure_geo['Closure_Count'] > 0]

# Step 2: Create a 10-mile buffer around each directly affected ZIP code (10 miles ≈ 16093.4 meters)
directly_affected_zips['buffer'] = directly_affected_zips.geometry.buffer(16093.4)

# Step 3: Convert buffered areas to a new GeoDataFrame
buffered_zips = gpd.GeoDataFrame(directly_affected_zips[['ZIP_CD', 'buffer']],
                                 geometry='buffer',
                                 crs=directly_affected_zips.crs)

# Step 4: Perform spatial join to find ZIP codes within the buffer area
# Ensure texas_zip_gdf has geometries of all Texas ZIP codes
indirectly_affected_zips = gpd.sjoin(texas_zip_gdf, buffered_zips, how='inner', predicate='intersects')

# Step 5: Get unique ZIP codes and count them
indirectly_affected_zips_list = indirectly_affected_zips[['ZIP_CD']].drop_duplicates().reset_index(drop=True)
```

```
indirectly_affected_count = indirectly_affected_zips_list['ZIP_CD'].nunique()

# Output results
print("Number of indirectly affected ZIP codes:", indirectly_affected_count)
print("List of indirectly affected ZIP codes:\n", indirectly_affected_zips_list)
```

```
Number of indirectly affected ZIP codes: 14
List of indirectly affected ZIP codes:
    ZIP_CD
0   78613
1   75140
2   75051
3   76645
4   79520
5   79529
6   79902
7   76531
8   76520
9   77065
10  75235
11  75087
12  75390
13  78336
```

4.

```
# Start with all Texas ZIP codes as 'unaffected'
texas_zip_gdf["closure_category"] = "Not Affected"

# Mark directly affected ZIP codes
texas_zip_gdf.loc[texas_zip_gdf["ZCTA5"].isin(directly_affected_zips["ZCTA5"]), "closure_category"] = "Directly A

# Mark indirectly affected ZIP codes (within 10 miles but not directly affected)
texas_zip_gdf.loc[texas_zip_gdf["ZCTA5"].isin(indirectly_affected_zips["ZCTA5"]), "closure_category"] = "Indirect

# Plot the choropleth map with different colors for each category
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_zip_gdf.plot(column="closure_category", cmap="viridis", linewidth=0.8, ax=ax, edgecolor="0.8", legend=True)
plt.title("Texas ZIP Codes by Hospital Closure Impact (2016-2019)")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```



Reflecting on the exercise

1

To improve accuracy for identifying hospital closures, we can cross check closures using multiple sources like financial reports, medical directories, and local news, rather than relying on one source which might have errors or outdated information. Tracking each hospital's status over several years would also help to distinguish permanent closures from temporary ones, like those due to renovations or short term issues. Additionally, local government and community reports often gives valuable, up-to-date insight on medical resources in the area, which could help verify the actual status of hospitals. Together, these methods create a clearer and more reliable picture of hospital closures, avoiding misinterpretations due to single-year or single-source data.

2

Potential issues and limitations include data accuracy and delays; sometimes hospitals appear "closed" due to data entry errors or mergers, and a single source might miss important context, like whether a closure is temporary. Additionally, a 10-mile impact radius affects densely populated areas differently than rural ones, where fewer people may be impacted. Hospital size and services also vary, so the closure of a large hospital likely has a bigger effect than a small clinic. Finally, alternative hospitals are often overlooked; some areas have other nearby options, while in others, a closed hospital may be the only one available. To improve accuracy, we could cross-check closures with additional data sources, look for long-term trends to confirm permanent closures, and consult community or government reports. Adjusting the impact radius based on population density, considering hospital size and services, and assessing other nearby hospitals would also provide a more accurate view. A more nuanced model, like the Two-Step Floating Catchment Area method (2SFCA), which considers both distance and population density, could further refine the impact assessment.