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

Problem Set 4

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

Style Points (10 pts)

Submission Steps (10 pts)

Partner 1: Kaijie(KJ) Wu, CNet ID: kaijie7 Partner 2: Griffin Sharps, CNet ID: gsharps "This submission is our work alone and complies with the 30538 integrity policy." Kaijie Wu & Griffin Sharps. Late coins used this pset: 1. Kaijie: 3 Late coins left after submission. Griffin: 2 Late coins left after submission.

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

1. The variables I pulled are: "PRVDR_CTGRY_SBTYP_CD", "PRVDR_CTGRY_CD", "FAC_NAME", "PRVDR_NUM", "PGM_TRMNTN_CD", and "ZIP_CD".

2. a. The unprocesssed data tells us that there were 7245 short-term hospitals in the US in 2016. However, there are 141557 unique entries in this dataset. Given that the American Hospital Association's report (https://www.aha.org/system/files/2018-02/2018-aha-hospital-fast-facts_0.pdf) lists there as only being 5534 hospitals in country in 2016, this data needs to be processed and cleaned to be useful.
   b. Accoding to the American Hospital Association's report , there were 4840 community hospitals in 2016. Because the report states that short-term hospitals are a subcategory of community hospitals, it does not make any sense of there to be more short-term hospitals than community hospitals. We suspect that this differs because the way hospitals close is not appropriately tracked. Closed and inactive hospitals are probably included in the unprocessed data.

```
### importing libraries
import pandas as pd
import altair as alt
import geopandas as gpd
import altair as alt
import json
import shapely
import random
import time
import matplotlib.pyplot as plt
```

```
### paths
# setting base path
base_path = r"C:\Users\james\Desktop\problem-set-4-kj-partner-ps4\pos{year}.csv"

# Create file paths for different years
file_path_2016 = base_path.format(year=2016)
file_path_2017 = base_path.format(year=2017)
file_path_2018 = base_path.format(year=2018)
file_path_2019 = base_path.format(year=2019)
```

```
### importing 2016 Provider of Service data
pos2016_data = pd.read_csv(file_path_2016)

# filtering for short-term hospitals
short_term_hospitals_2016 = pos2016_data[
    (pos2016_data['PRVDR_CTGRY_CD'] == 1) &
    (pos2016_data['PRVDR_CTGRY_SBTYP_CD'] == 1)
]

# counting and printing short-term hospitals
num_short_term_hospitals_2016 = short_term_hospitals_2016.shape[0]
print(num_short_term_hospitals_2016)

# counting an printing total hospitals
num_total_hospitals = pos2016_data['PRVDR_NUM'].nunique()
print(num_total_hospitals)
```

```
7245
141557
```

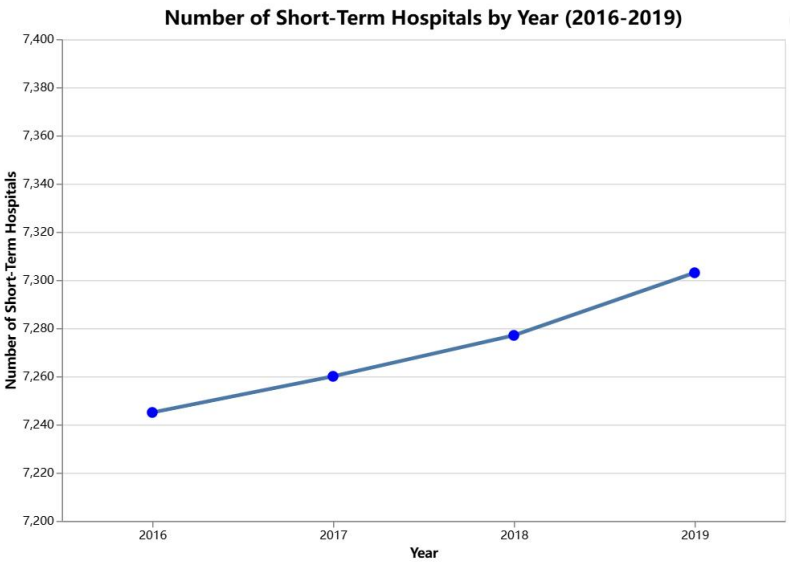
3. The number of short-term hospitals in 2017 was 7260. In 2018 it was 7277. And in 2019 it was 7303.

```
### 2017 Data
pos2017_data = pd.read_csv(file_path_2017)

short_term_hospitals_2017 = pos2017_data[(pos2017_data['PRVDR_CTGRY_CD'] == 1) & (pos2017_data['PRVDR_CTGRY_SBTYP_CD'] == 1)]

num_short_term_hospitals_2017 = short_term_hospitals_2017.shape[0]
print(num_short_term_hospitals_2017)

### 2018 Data
pos2018_data = pd.read_csv(file_path_2018, encoding='latin1')

short_term_hospitals_2018 = pos2018_data[(pos2018_data['PRVDR_CTGRY_CD'] == 1) & (pos2018_data['PRVDR_CTGRY_SBTYP_CD'] == 1)]

num_short_term_hospitals_2018 = short_term_hospitals_2018.shape[0]
print(num_short_term_hospitals_2018)

### 2019 Data
```

```
pos2019_data = pd.read_csv(file_path_2019, encoding='latin1')

short_term_hospitals_2019 = pos2019_data[(pos2019_data['PRVDR_CTGRY_CD'] == 1) & (pos2019_data['PRVDR_CTGRY_SBTYP_CD'] == 1)]

num_short_term_hospitals_2019 = short_term_hospitals_2019.shape[0]
print(num_short_term_hospitals_2019)
```

```
7260
7277
7303
```

```
### plot of number of short-term hospitals by year
yearly_counts_df = pd.DataFrame({
    'Year': [2016, 2017, 2018, 2019],
    'Count': [7245, 7260, 7277, 7303]
})

chart = alt.Chart(yearly_counts_df).mark_line(point=alt.OverlayMarkDef(size=80, color='blue')).encode(
    x=alt.X('Year:O', title='Year', axis=alt.Axis(labelAngle=0)),
    y=alt.Y('Count:Q', title='Number of Short-Term Hospitals', scale=alt.Scale(domain=[7200, 7400])),
    tooltip=[alt.Tooltip('Year', title='Year'), alt.Tooltip('Count', title='Hospitals')]
).properties(
    title="Number of Short-Term Hospitals by Year (2016-2019)",
    width=600,
    height=400
).configure_line(
    size=3
).configure_title(
    fontSize=16,
    anchor='middle',
)

chart.display()
```

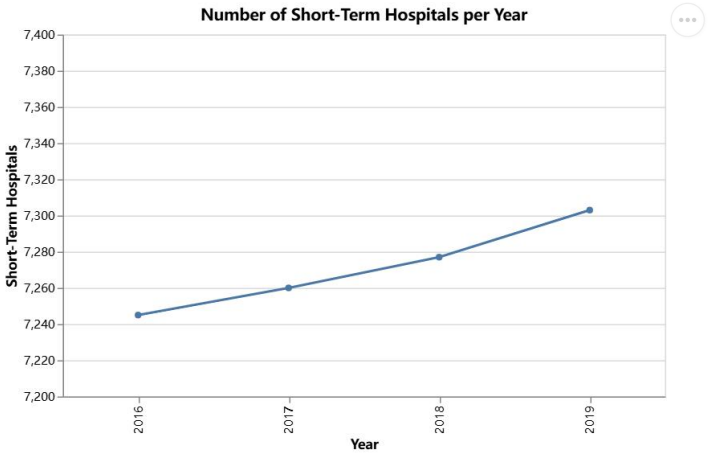


4. a.

```
## checking for duplicates

# Dictionary of file paths for each year
file_paths = {
    "2016": file_path_2016,
    "2017": file_path_2017,
    "2018": file_path_2018,
    "2019": file_path_2019
}

short_term_hospitals_per_year = {}

for year, file_path in file_paths.items():
    data = pd.read_csv(file_path, encoding='latin1')
    short_term_hospitals = data[(data['PRVDR_CTGRY_CD'] == 1) & (data['PRVDR_CTGRY_SBTYP_CD'] == 1)]
    short_term_hospitals_per_year[year] = short_term_hospitals['PRVDR_NUM'].nunique()

short_term_hospitals_df = pd.DataFrame(list(short_term_hospitals_per_year.items()), columns=['Year', 'Short-Term Hospitals'])

chart = alt.Chart(short_term_hospitals_df).mark_line(point=True).encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Short-Term Hospitals:Q', title='Short-Term Hospitals', scale=alt.Scale(domain=[7200,7400])),
    tooltip=['Year', 'Short-Term Hospitals']
).properties(
    title='Number of Short-Term Hospitals per Year',
    width=500,
    height=300
```

```
)
chart.display()
```

```
C:\Users\james\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``.
```



b. This and the previous plot are the same. This means that the number of short-term hospitals in both datasets is the same. This allows us to establish that no hospitals are duplicated in our datasets.

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

1. The suspected number of hospital closures is 174.

```
# Add a 'Year' column to each DataFrame to indicate the year of the data
short_term_hospitals_2016.loc[:, 'Year'] = 2016
short_term_hospitals_2017.loc[:, 'Year'] = 2017
short_term_hospitals_2018.loc[:, 'Year'] = 2018
short_term_hospitals_2019.loc[:, 'Year'] = 2019

# Combine the DataFrames for all years into a single DataFrame
df_pos_all_years = pd.concat([short_term_hospitals_2016, short_term_hospitals_2017, short_term_hospitals_2018, short_term_hospitals_2019], ignore_index=True)

# Identify hospitals that were active in 2016 (i.e. 'PGM_TRMNTN_CD' equals 0)
hospitals_2016_active = df_pos_all_years[(df_pos_all_years['Year'] == 2016) & (df_pos_all_years['PGM_TRMNTN_CD'] == 0)]

# Create an empty DataFrame to store information about suspected hospital closures
hospital_closure = pd.DataFrame(columns=['PRVDR_NUM', 'FAC_NAME', 'ZIP_CD', 'Year of Closure'])

# Iterate through each year from 2017 to 2019 to check for hospital closures
for year in range(2017, 2020):
     # Get a set of provider numbers for hospitals that are still active in the current year
    active_hospitals_current_year = set(df_pos_all_years[(df_pos_all_years['Year'] == year) &
                                                         (df_pos_all_years['PGM_TRMNTN_CD'] == 0)]['PRVDR_NUM'])
     # Initialize an empty DataFrame to store closure information for the current year
    year_closures = pd.DataFrame()
    # Iterate over each hospital that was active in 2016
    for _, hospital_info in hospitals_2016_active.iterrows():
        # Check if the hospital is no longer active in the current year
        if hospital_info['PRVDR_NUM'] not in active_hospitals_current_year:
            # Create a DataFrame entry for the hospital closure
            closure_entry = pd.DataFrame({
                'PRVDR_NUM': [hospital_info['PRVDR_NUM']],
                'FAC_NAME': [hospital_info['FAC_NAME']],
                'ZIP_CD': [hospital_info['ZIP_CD']],
                'Year of Closure': [year]
            })
            # Append the closure entry to the 'year_closures' DataFrame
            year_closures = pd.concat([year_closures, closure_entry], ignore_index=True)

    # Append the year's closure information to the main 'hospital_closure' DataFrame
    hospital_closure = pd.concat([hospital_closure, year_closures], ignore_index=True)

# Remove duplicate entries based on 'PRVDR_NUM' to ensure each hospital appears only once
hospital_closure.drop_duplicates(subset='PRVDR_NUM', inplace=True)

print(hospital_closure)
print(f"Total suspected closures: {len(hospital_closure)}")
```

```
C:\Users\james\AppData\Local\Temp\ipykernel_19488\781976808.py:2: 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-a-copy

C:\Users\james\AppData\Local\Temp\ipykernel_19488\781976808.py:3: 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-a-copy

C:\Users\james\AppData\Local\Temp\ipykernel_19488\781976808.py:4: 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-a-copy

C:\Users\james\AppData\Local\Temp\ipykernel_19488\781976808.py:5: 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-a-copy

C:\Users\james\AppData\Local\Temp\ipykernel_19488\781976808.py:38: FutureWarning:

The behavior of DataFrame concatenation with empty or all-NA entries is deprecated. In a future version, this will no longer exclude empty or all-NA columns when
determining the result dtypes. To retain the old behavior, exclude the relevant entries before the concat operation.
    PRVDR_NUM                                      FAC_NAME   ZIP_CD  \
0      030001                          ABRAZO MARYVALE CAMPUS  85031.0
1      050196          ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230.0
2      050435                       FALLBROOK HOSPITAL DISTRICT  92028.0
3      060036           ARKANSAS VALLEY REGIONAL MEDICAL CENTER  81050.0
4      060043                         KEEFE MEMORIAL HOSPITAL  80810.0
..        ...                                             ...      ...
303    670083                          TEXAS GENERAL HOSPITAL  75051.0
304    670087  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613.0
307    670094           LITTLE RIVER HEALTHCARE CAMERON HOSPITAL  76520.0
309    670097                    BAYLOR EMERGENCY MEDICAL CENTER  75087.0
311    670117                  TEXAS GENERAL HOSPITAL- VZRMC LP  75140.0

    Year of Closure
0              2017
1              2017
2              2017
3              2017
4              2017
..              ...
303            2019
304            2019
307            2019
309            2019
311            2019

[174 rows x 4 columns]
Total suspected closures: 174
```

2.

```
sorted_hospitals = hospital_closure.sort_values(by='FAC_NAME').head(10)

print(sorted_hospitals[['FAC_NAME', 'Year of Closure']])
```

```
                                    FAC_NAME Year of Closure
0                          ABRAZO MARYVALE CAMPUS            2017
1        ADVENTIST MEDICAL CENTER - CENTRAL VALLEY            2017
90                        AFFINITY MEDICAL CENTER            2018
15   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS            2017
29        ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE            2017
200                       ALLIANCE LAIRD HOSPITAL            2019
239                      ALLIANCEHEALTH DEACONESS            2019
164                 ANNE BATES LEACH EYE HOSPITAL            2019
3          ARKANSAS VALLEY REGIONAL MEDICAL CENTER            2017
11             BANNER CHURCHILL COMMUNITY HOSPITAL            2017
```

3. a. The number of potential merger/acquisition is 148.

```
# List to store CMS certification numbers of hospitals that might have been involved in a merger or acquisition
potential_mergers = []

# Iterate through each row in the hospital_closure DataFrame
for index, row in hospital_closure.iterrows():
    # Get the ZIP code and the year of suspected closure for the current hospital
    zip_code = row['ZIP_CD']
    closure_year = row['Year of Closure']

    # Get data for the year after the suspected closure for the same ZIP code
    # Filter for hospitals that are still active (PGM_TRMNTN_CD == 0)
    next_year_data = df_pos_all_years[(df_pos_all_years['Year'] == closure_year + 1) &
                                      (df_pos_all_years['ZIP_CD'] == zip_code) &
                                      (df_pos_all_years['PGM_TRMNTN_CD'] == 0)]
```

```
    # Get data for the year of the suspected closure for the same ZIP code
    current_year_data = df_pos_all_years[(df_pos_all_years['Year'] == closure_year) &
                                         (df_pos_all_years['ZIP_CD'] == zip_code) &
                                         (df_pos_all_years['PGM_TRMNTN_CD'] == 0)]

    # If the number of active hospitals in the next year is not less than the current year,
    # it suggests a potential merger or acquisition, so we add the hospital to potential_mergers
    if len(next_year_data) >= len(current_year_data):
        potential_mergers.append(row['PRVDR_NUM'])

# Filter out hospitals identified as potential mergers from hospital_closure
corrected_closures = hospital_closure[~hospital_closure['PRVDR_NUM'].isin(potential_mergers)]

# Count the number of potential mergers/acquisitions
mergers_count = len(potential_mergers)
print(f"Number of potential mergers/acquisitions: {mergers_count}")
```

```
Number of potential mergers/acquisitions: 148
```

b.

The number of corrected suspected hospital closures is 26.

```
# Calculate and print the number of hospitals remaining after removing potential mergers
remaining_hospitals_count = len(corrected_closures)
print(f"Remaining suspected closures after correction: {remaining_hospitals_count}")
```

```
Remaining suspected closures after correction: 26
```

c.

```
# Sort the corrected closures by facility name and display the first 10 entries
sorted_corrected_hospitals = corrected_closures.sort_values(by='FAC_NAME').head(10)
print(sorted_corrected_hospitals[['FAC_NAME', 'Year of Closure']])
```

```
                                      FAC_NAME Year of Closure
239                      ALLIANCEHEALTH DEACONESS            2019
164                    ANNE BATES LEACH EYE HOSPITAL            2019
253                    BARIX CLINICS OF PENNSYLVANIA            2019
304  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...            2019
203              BLACK RIVER COMMUNITY MEDICAL CENTER            2019
171                        CHESTATEE REGIONAL HOSPITAL            2019
191              DOCTORS HOSPITAL AT DEER CREEK L L C            2019
287                        EL PASO SPECIALTY HOSPITAL            2019
176                          FRANCISCAN HEALTH CARMEL            2019
168                          HUTCHESON MEDICAL CENTER            2019
```

Download Census zip code shapefile (10 pt)

1. a. The five file types are Shape File (.shp), Shape Index File (.shx), Attribute Data File (.dbf), Projection File (.prj), and Metadata File (.xml).

- A Shape File contains the geometry data:the shapes of geographical features.
- A Shape Index File is an index for that geometry data.
- An Attribute Data File holds attribute data in a dBase table format, where each row corresponds to a * shape in the shape file, providing details such as ZIP code and other metadata.
- A Projection File is the information about the coordinate system and projection used. It is critical for accurately aligning shapefile data on a map.
- A Metadata File provides the information about the shapefile, including its origin, date of creation, and any limitations on its usage.

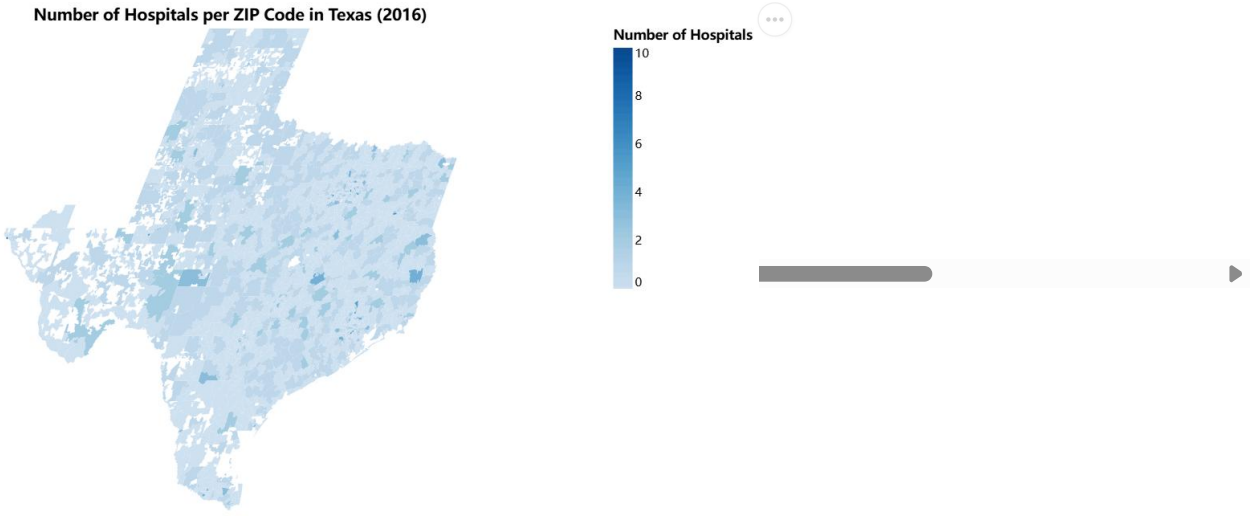
   b. By looking at the files we can see that:

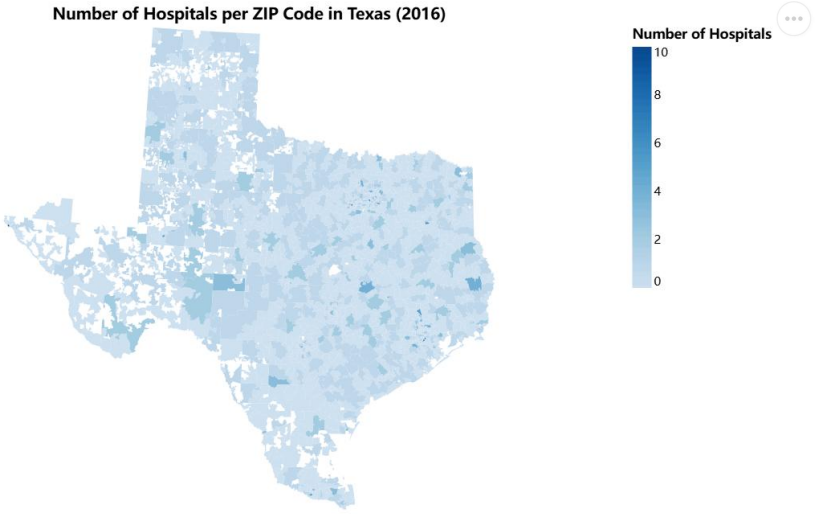
- The size of Attribute Data File is 6.4 MB.
- The size of Projection File is 165 bytes.
- The size of Shape File is 837.5 MB.
- The size of Shape Index File is 266 KB.
- The size of Metadata File is 16 KB.

2.

```
# Load the ZIP Code Shapefile using GeoPandas
zip_shapefile = gpd.read_file(r"C:\Users\james\Desktop\gz_2010_us_860_00_500k\gz_2010_us_860_00_500k.shp")

# Use a range to generate the prefixes and convert them to strings
texas_prefixes = tuple(str(i) for i in range(75, 80))

# Subset the GeoDataFrame using the generated prefixes
texas_zip_shapefile = zip_shapefile[zip_shapefile['ZCTA5'].str.startswith(texas_prefixes)].copy()

# Filter the POS data for the year 2016
pos_2016 = df_pos_all_years[(df_pos_all_years['Year'] == 2016)]
```

```
# Count the number of hospitals per ZIP code and create a DataFrame
hospital_counts = pos_2016['ZIP_CD'].value_counts().reset_index()
hospital_counts.columns = ['ZIP_CD', 'Hospital_Count']  # Rename columns for clarity

# Make a copy of the filtered GeoDataFrame to avoid warnings
texas_zip_shapefile = texas_zip_shapefile.copy()

# Convert 'ZCTA5' to integer format for merging
texas_zip_shapefile['ZIP_CD'] = texas_zip_shapefile['ZCTA5'].astype(int)

# Merge the Texas ZIP shapefile data with the hospital counts DataFrame
# This adds the 'Hospital_Count' column to the GeoDataFrame
texas_map_data = texas_zip_shapefile.merge(hospital_counts, how='left', left_on='ZIP_CD', right_on='ZIP_CD')

# Fill any NaN values in 'Hospital_Count' with 0
texas_map_data['Hospital_Count'] = texas_map_data['Hospital_Count'].fillna(0)

# Convert the merged GeoDataFrame to GeoJSON format for visualization in Altair
texas_geojson = json.loads(texas_map_data.to_json())

# Create a choropleth map using Altair to visualize the number of hospitals per ZIP code
choropleth = alt.Chart(alt.Data(values=texas_geojson['features'])).mark_geoshape().encode(
    color=alt.Color('properties.Hospital_Count:Q', scale=alt.Scale(scheme='blues'), title='Number of Hospitals'),
    tooltip=[
        alt.Tooltip('properties.ZCTA5CE10:O', title='ZIP Code'),
        alt.Tooltip('properties.Hospital_Count:Q', title='Hospital Count')
    ]
).transform_calculate(
    ZCTA5CE10='datum.properties.ZCTA5CE10',
    Hospital_Count='datum.properties.Hospital_Count'
).properties(
    width=600,
    height=400,
    title='Number of Hospitals per ZIP Code in Texas (2016)'
)

choropleth.display()

# This yields a skewed map of the state. ChatGPT suggests that this because we did not use a "geographic projection of the data". We corrected this using the albe

# Create a choropleth map using Altair with the 'albersUsa' projection
choropleth = alt.Chart(alt.Data(values=texas_geojson['features'])).mark_geoshape().encode(
    color=alt.Color('properties.Hospital_Count:Q', scale=alt.Scale(scheme='blues'), title='Number of Hospitals'),
    tooltip=[
        alt.Tooltip('properties.ZCTA5CE10:O', title='ZIP Code'),
        alt.Tooltip('properties.Hospital_Count:Q', title='Hospital Count')
    ]
).transform_calculate(
    ZCTA5CE10='datum.properties.ZCTA5CE10',
    Hospital_Count='datum.properties.Hospital_Count'
).properties(
    width=600,
    height=400,
    title='Number of Hospitals per ZIP Code in Texas (2016)'
).project(
    type='albersUsa'  # Apply the Albers USA projection
)

choropleth.display()
```





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

1.

```
zips_all_centroids = zip_shapefile.copy()  # Make a copy to avoid modifying the original GeoDataFrame

# Update the 'geometry' column to contain the centroids of each polygon
zips_all_centroids['geometry'] = zips_all_centroids['geometry'].centroid

# Display the dimensions of the resulting GeoDataFrame
print(zips_all_centroids.shape)
print(zips_all_centroids.columns)
```

```
C:\Users\james\AppData\Local\Temp\ipykernel_19488\148395807.py:4: 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.


(33120, 6)
Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA', 'geometry'], dtype='object')
```

The dimensions of the resulting GeoDataFrame are 33120 x 6.

A manual search of the .xml file shows that the columns are: * GEO_ID: Unique identifier for a geographic entity [alphanumeric] * ZCTA5: ZIP Code Tabulation Area [5-digit Census code] * Name: Name without translated Legal/Statistical Area Description (LSAD) [alphanumeric] * LSAD: Standard abbreviation translation of Legal/Statistical Area Description (LSAD) code as used on census maps [alpha] * CENSUSAREA: Area of entity before generalization in square miles [numeric] * geometry: the geometry of the centroid points

2.

```
# GeoDataFrame for all Texas zip codes
zips_texas_centroids = zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith((texas_prefixes))].copy()

# New tuple with all zip code prefixes
all_prefixes = tuple(
    str(i) for i in range(75, 80)   # 75-79
) + tuple(
    str(i) for i in range(870, 885) # 870-884
) + tuple(
    str(i) for i in range(73, 75)   # 73-74
) + tuple(
    str(i) for i in range(716, 730) # 716-729
) + tuple(
    str(i) for i in range(700, 716) # 700-715
)

# Subset the GeoDataFrame using the combined prefixes
zips_texas_borderstates_centroids = zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(all_prefixes)].copy()

print(zips_texas_centroids.shape)
print(zips_texas_borderstates_centroids.shape)

zips_texas_borderstates_centroids
```

```
(1935, 6)
(4057, 6)
```

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry
8870	8600000US70003	70003	70003	ZCTA5	7.019	POINT (-90.21397 29.99864)
8871	8600000US70030	70030	70030	ZCTA5	31.486	POINT (-90.43225 29.81731)
8872	8600000US70032	70032	70032	ZCTA5	1.748	POINT (-89.99779 29.95816)

	GEO_ID	ZCTA5	NAME	LSAD	CENSUSAREA	geometry
8873	8600000US70036	70036	70036	ZCTA5	4.379	POINT (-90.12115 29.70903)
8874	8600000US70038	70038	70038	ZCTA5	1.329	POINT (-89.39875 29.32533)
...	...	...	...	...	...	...
32917	8600000US78261	78261	78261	ZCTA5	29.865	POINT (-98.40189 29.6918)
32918	8600000US78368	78368	78368	ZCTA5	216.341	POINT (-97.8103 28.10544)
32919	8600000US78412	78412	78412	ZCTA5	8.798	POINT (-97.34297 27.70383)
32920	8600000US78557	78557	78557	ZCTA5	11.653	POINT (-98.24337 26.10675)
32921	8600000US78586	78586	78586	ZCTA5	176.313	POINT (-97.6311 26.10507)

4057 rows × 6 columns
There are 1935 zip codes in Texas and 4057 in Texas and its bordering states.

3.

```
# Convert 'ZIP_CD' to a string in the hospital_counts DataFrame to match the ZCTA5 column
hospital_counts['ZIP_CD'] = hospital_counts['ZIP_CD'].astype(int).astype(str)

# Merge the hospital_counts DataFrame with the zips_texas_borderstates_centroids GeoDataFrame
zips_withhospital_centroids = zips_texas_borderstates_centroids.merge(
    hospital_counts,
    how='left',
    left_on='ZCTA5',
    right_on='ZIP_CD'
)

# Fill any NaN values in the Hospital_Count column with 0
zips_withhospital_centroids['Hospital_Count'] = zips_withhospital_centroids['Hospital_Count'].fillna(0)

# Subset the merged GeoDataFrame to only include ZIP codes with at least one hospital
zips_withhospital_centroids = zips_withhospital_centroids[zips_withhospital_centroids['Hospital_Count'] > 0]
```

I used a left merge and merged on the variables ZCTA5 (left) and ZIP_CD (right). This ensures that we keep all of the zip codes in the zips_texas_borderstates_centroid GeoDataFrame.

4. a. It takes 0.13 seconds to run the test code using 10 zip codes. Given that there are 1935 zip codes in Texas, I would expect the full code to take 25.16 seconds to run.

```
# Start the timer
start_time = time.time()

# Your code chunk to be timed
# Select 10 random indices from the re-projected GeoDataFrame
random_indices = random.sample(range(len(zips_texas_centroids)), 10)

# Subset the re-projected GeoDataFrame to include only the selected random indices
zips_texas_sample = zips_texas_centroids.iloc[random_indices]

# Calculate distances for each point in the sample to all points in zips_withhospital_centroids_projected
zips_texas_sample.loc[:, 'Distance_to_Nearest_Hospital'] = [
    zips_withhospital_centroids.distance(point).min() for point in zips_texas_sample.geometry
]

# Print the results
print(zips_texas_sample[['ZCTA5', 'Distance_to_Nearest_Hospital']])

# End the timer
end_time = time.time()

# Calculate and print the elapsed time
elapsed_time = end_time - start_time
print(f"Elapsed time: {elapsed_time:.2f} seconds")
```

```
       ZCTA5  Distance_to_Nearest_Hospital
9973   76671                      0.189381
9587   76092                      0.000000
24943  75937                      0.160470
28660  78580                      0.000000
10619  78056                      0.246042
9831   76666                      0.217399
10202  77094                      0.000000
10815  78332                      0.000000
10227  77407                      0.074362
24861  78677                      0.180862
Elapsed time: 0.12 seconds

C:\Users\james\AppData\Local\Temp\ipykernel_19488\964348748.py:13: UserWarning:

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


C:\Users\james\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-a-copy
```

b.
It takes 8.74 seconds to run the full code. My estimation was far too pessimistic!

```
# Start the timer
start_time = time.time()

# Calculate distances for each point in zips_texas_centroids to all points in zips_withhospital_centroids and find the minimum distance
zips_texas_centroids['Distance_to_Nearest_Hospital'] = [
    zips_withhospital_centroids.distance(point).min() for point in zips_texas_centroids.geometry
]

# End the timer
end_time = time.time()

# Calculate and print the elapsed time
elapsed_time = end_time - start_time
print(f"Elapsed time: {elapsed_time:.2f} seconds")
```

```
C:\Users\james\AppData\Local\Temp\ipykernel_19488\783210012.py:6: UserWarning:

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


Elapsed time: 21.60 seconds
```

c.
The units given in the .prj file are degrees, which cannot be converted directly into miles because they rely on both latitude and longitude. As you can see later, I used the Albers USA projection(EPSG:5070), which I found when I made the Altair chart above. This gives distances in meters, which I can convert to miles using the variable I created below.
In this region of the US 1 degree is roughly equivalent to just over 54 miles.

```
# Specify the path to your .prj file
prj_file_path = r'C:\Users\james\Desktop\gz_2010_us_860_00_500k\gz_2010_us_860_00_500k.prj'

# Read the contents of the .prj file
with open(prj_file_path, 'r') as file:
    prj_content = file.read()

# Print the CRS information
print("Contents of the .prj file:")
print(prj_content)

# Reproject to a projected CRS, such as Albers USA (EPSG:5070)
texas_projected = texas_zip_shapefile.to_crs(epsg=5070)

# Now distances are in meters. Use the conversion factor to convert to miles
meters_to_miles = 0.000621371

# Example: Calculating distance between two points in miles
point1 = texas_projected.geometry.iloc[0]
point2 = texas_projected.geometry.iloc[1]
distance_in_meters = point1.distance(point2)
distance_in_miles = distance_in_meters * meters_to_miles

print(f"Distance in miles: {distance_in_miles:.2f}")
```

```
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",0.0174532925199433295]]
Distance in miles: 54.29
```

5.

```
# Reproject both GeoDataFrames to Albers Equal Area (EPSG:5070) for accurate distance calculations
zips_texas_centroids_projected = zips_texas_centroids.to_crs(epsg=5070)
zips_withhospital_centroids_projected = zips_withhospital_centroids.to_crs(epsg=5070)

# Calculate the average distance to all hospitals for each ZIP code in Texas
average_distances = []
for point in zips_texas_centroids_projected.geometry:
    # Calculate distances from the current point to all hospital centroids
    all_distances = zips_withhospital_centroids_projected.distance(point)
    # Calculate the average distance
    average_distance = all_distances.mean()
    average_distances.append(average_distance)

# Convert distances from meters to miles
meters_to_miles = 0.000621371
average_distances_in_miles = [distance * meters_to_miles for distance in average_distances]

# Add the average distances as a new column to the original GeoDataFrame
zips_texas_centroids['Average_Distance_to_Nearest_Hospital'] = average_distances_in_miles
```

a.
The units are originally in degrees. I updated the CRS so that they were then in meters. Then I converted meters to miles.

b.

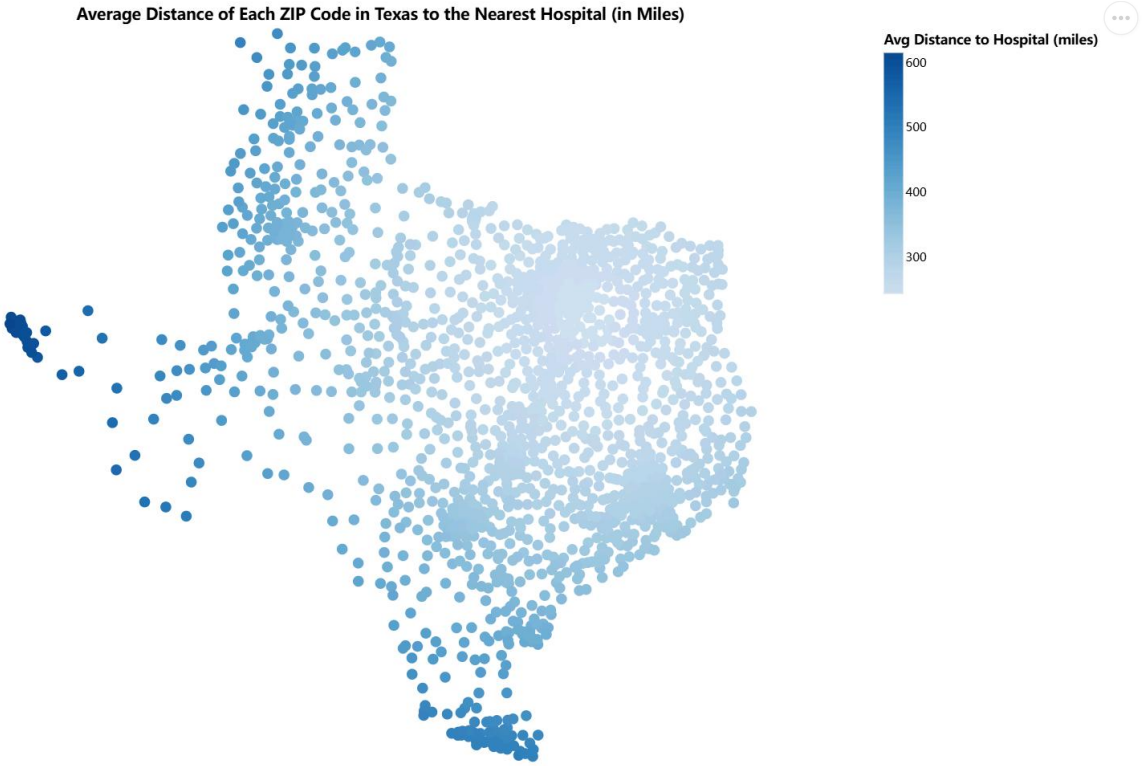
My calculation shows that the average distance from a Texas zipcode's centroid to a hospital in Texas is 317.04 miles. This does not make sense. However, I found this question confusing. There can only be one nearest hospital to each centroid--that is there is a single distance between it and the nearest centroid with a hospital. I posted about this repeatedly on Ed but got no response, even after 24 hours. If my understanding of the question is faulty, I'm not sure what I was supposed to do.

```
overall_average_distance = zips_texas_centroids['Average_Distance_to_Nearest_Hospital'].mean()

print(f"Overall average distance of all ZIP codes in Texas to a hospital: {overall_average_distance:.2f} miles")
```

```
Overall average distance of all ZIP codes in Texas to a hospital: 317.04 miles
```

c.

```
# Convert the updated GeoDataFrame to GeoJSON format
zips_texas_geojson = json.loads(zips_texas_centroids.to_json())

# Create a choropleth map using Altair
centroid_choropleth = alt.Chart(alt.Data(values=zips_texas_geojson['features'])).mark_geoshape().encode(
    color=alt.Color(
        'properties.Average_Distance_to_Nearest_Hospital:Q',
        scale=alt.Scale(scheme='blues'),
        title='Avg Distance to Hospital (miles)'
    ),
    tooltip=[
        alt.Tooltip('properties.ZCTA5:O', title='ZIP Code'),
        alt.Tooltip('properties.Average_Distance_to_Nearest_Hospital:Q', title='Avg Distance (miles)')
    ]
).transform_calculate(
    ZCTA5='datum.properties.ZCTA5',
    Average_Distance_to_Nearest_Hospital='datum.properties.Average_Distance_to_Nearest_Hospital'
).properties(
    width=800,
    height=600,
    title='Average Distance of Each ZIP Code in Texas to the Nearest Hospital (in Miles)'
).project(
    type='albersUsa'
)

centroid_choropleth.display()
```



Effects of closures on access in Texas (15 pts)

1.

```
# Filter the `corrected_closures` DataFrame for Texas ZIP codes
texas_closures = corrected_closures[corrected_closures['ZIP_CD'].astype(str).str.startswith(texas_prefixes)]

# List  all the unique ZIP codes in Texas affected by hospital closures
affected_zip_codes = texas_closures['ZIP_CD'].unique().tolist()

# Generate a comprehensive list of all ZIP codes in Texas
all_texas_zip_codes = zips_texas_centroids['ZCTA5'].unique()
```

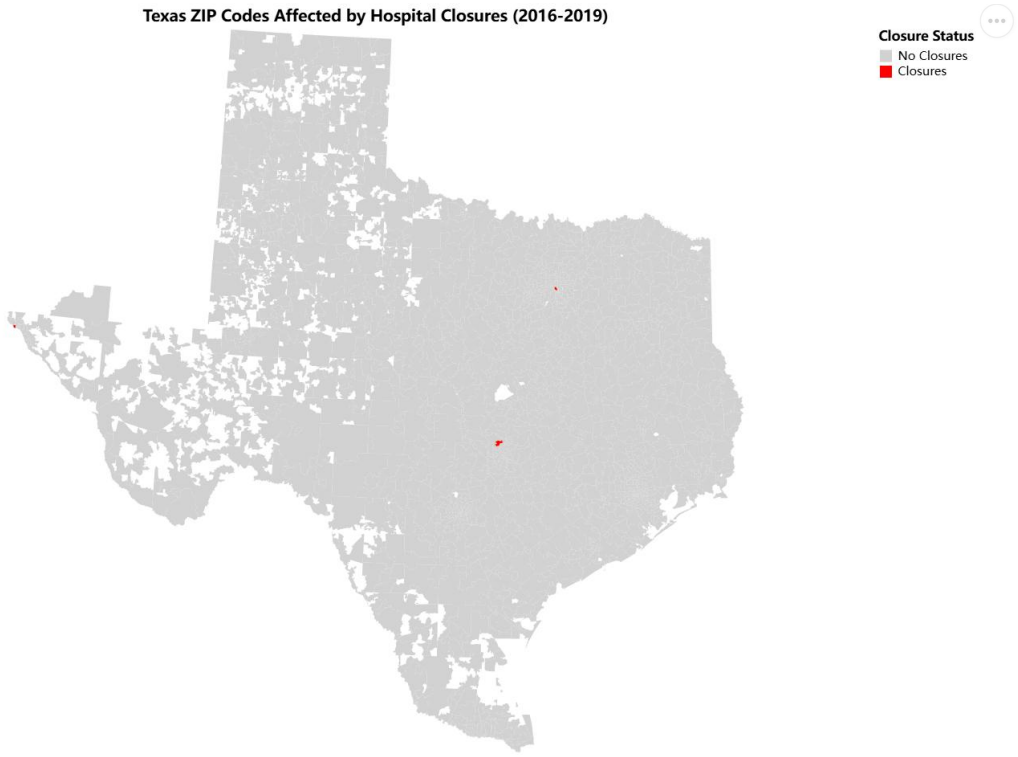
```
# Convert the list to a DataFrame
all_texas_zip_codes_df = pd.DataFrame(all_texas_zip_codes, columns=['ZIP_CD'])

# Count the number of closures by ZIP code
closures_by_zip = texas_closures['ZIP_CD'].value_counts().reset_index()
closures_by_zip.columns = ['ZIP_CD', 'Number_of_Closures']

# Convert 'ZIP_CD' in closures_by_zip to string and remove decimal points
closures_by_zip['ZIP_CD'] = closures_by_zip['ZIP_CD'].astype(int).astype(str).str.zfill(5)

# Perform the left merge
texas_closures_table = all_texas_zip_codes_df.merge(closures_by_zip, on='ZIP_CD', how='left')

# Fill NaN values in 'Number_of_Closures' with 0 and change
texas_closures_table['Number_of_Closures'] = texas_closures_table['Number_of_Closures'].fillna(0).astype(int)

# Convert all 'ZIP_CD' values in texas_closures_table to int for consistency
texas_closures_table['ZIP_CD'] = texas_closures_table['ZIP_CD'].astype(int)

print("List of affected ZIP codes in Texas:")
print(affected_zip_codes)

print("\nTable showing the number of hospital closures in Texas by ZIP code (including zero closures):")
print(texas_closures_table)
```

```
List of affected ZIP codes in Texas:
[75390.0, 79902.0, 75235.0, 78613.0]

Table showing the number of hospital closures in Texas by ZIP code (including zero closures):
      ZIP_CD  Number_of_Closures
0      78624                  0
1      78626                  0
2      78628                  0
3      78631                  0
4      78632                  0
...      ...                ...
1930   78261                  0
1931   78368                  0
1932   78412                  0
1933   78557                  0
1934   78586                  0

[1935 rows x 2 columns]
```

2. 4 zip codes in Texas were affected by hospital closures.

```
# Make a copy of texas_projected GeoDataFrame
zips_texas = texas_zip_shapefile.copy()

# Merge the GeoDataFrame with the closures table
texas_closures_choropleth_data = zips_texas.merge(
    texas_closures_table[['ZIP_CD', 'Number_of_Closures']],
    on='ZIP_CD',
    how='left'
)

# Update the 'Number_of_Closures' column to create a categorical distinction
texas_closures_choropleth_data['Closure_Status'] = texas_closures_choropleth_data['Number_of_Closures'].apply(
    lambda x: 'No Closures' if x == 0 else 'Closures'
)

# Convert the updated GeoDataFrame to GeoJSON for Altair
choropleth_geojson = json.loads(texas_closures_choropleth_data.to_json())

# Plotting the choropleth
texas_closures_choropleth = alt.Chart(alt.Data(values=choropleth_geojson['features'])).mark_geoshape().encode(
    color=alt.Color(
        'properties.Closure_Status:N',
        scale=alt.Scale(
            domain=['No Closures', 'Closures'],
            range=['lightgray', 'red']
        ),
        legend=alt.Legend(
            title='Closure Status',
            orient='right',
            symbolType='square'
        )
    ),
    tooltip=[
        alt.Tooltip('properties.ZIP_CD:O', title='ZIP Code'),
        alt.Tooltip('properties.Closure_Status:N', title='Closure Status')
    ]
).properties(
    width=800,
    height=600,
    title='Texas ZIP Codes Affected by Hospital Closures (2016-2019)'
).project(
    type='albersUsa'
)
```

```
# Count the number of ZIP codes with at least one closure
num_affected_zips = len(affected_zip_codes)

texas_closures_choropleth.display()

print(f"Number of ZIP codes in Texas directly affected by hospital closures: {num_affected_zips}")
```




```
Number of ZIP codes in Texas directly affected by hospital closures: 4
```

3.

```
# Create a GeoDataFrame with only directly affected ZIP codes
affected_zips_gdf = texas_closures_choropleth_data[
    texas_closures_choropleth_data['Number_of_Closures'] > 0
].copy()

affected_zips_gdf_projected = affected_zips_gdf.to_crs(epsg=5070)

# Conversion factor: miles to meters
miles_to_meters = 1609.34

# Create a buffered GeoDataFrame around the directly affected ZIP codes
buffered_zips_gdf = affected_zips_gdf_projected.copy()
buffered_zips_gdf['geometry'] = affected_zips_gdf_projected['geometry'].buffer(10 * miles_to_meters)

# Perform a spatial join to see which ZIP codes are directly or indirectly affected
# Note: Use the `texas_projected` rather than the texas_zip_shapefile GeoDataFrame to ensure distances are in meters
joined_buffered_gdf = gpd.sjoin(texas_projected, buffered_zips_gdf, how='left', predicate='intersects')

print(joined_buffered_gdf)

# Add a new column to indicate if a ZIP code is affected directly or indirectly
joined_buffered_gdf['Affected_Status'] = joined_buffered_gdf.apply(
    lambda row: 'Indirectly Affected' if pd.notnull(row.index_right) else 'Not Affected',
    axis=1
)

# Update the status for ZIP codes that are directly affected by closures
joined_buffered_gdf.loc[
    joined_buffered_gdf['ZIP_CD_left'].isin(affected_zips_gdf['ZIP_CD']),
    'Affected_Status'
] = 'Directly Affected'
# change the "ZIP_CD" to "ZIP_CD_left"

# Count the number of ZIP codes marked as 'Indirectly Affected'
num_indirectly_affected = joined_buffered_gdf[joined_buffered_gdf['Affected_Status'] == 'Indirectly Affected'].shape[0]

print(f"Number of indirectly affected ZIP codes in Texas: {num_indirectly_affected}")
```

```
          GEO_ID_left ZCTA5_left NAME_left LSAD_left  CENSUSAREA_left  \
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
...               ...        ...       ...       ...              ...
32917  8600000US78261      78261     78261     ZCTA5           29.865
```

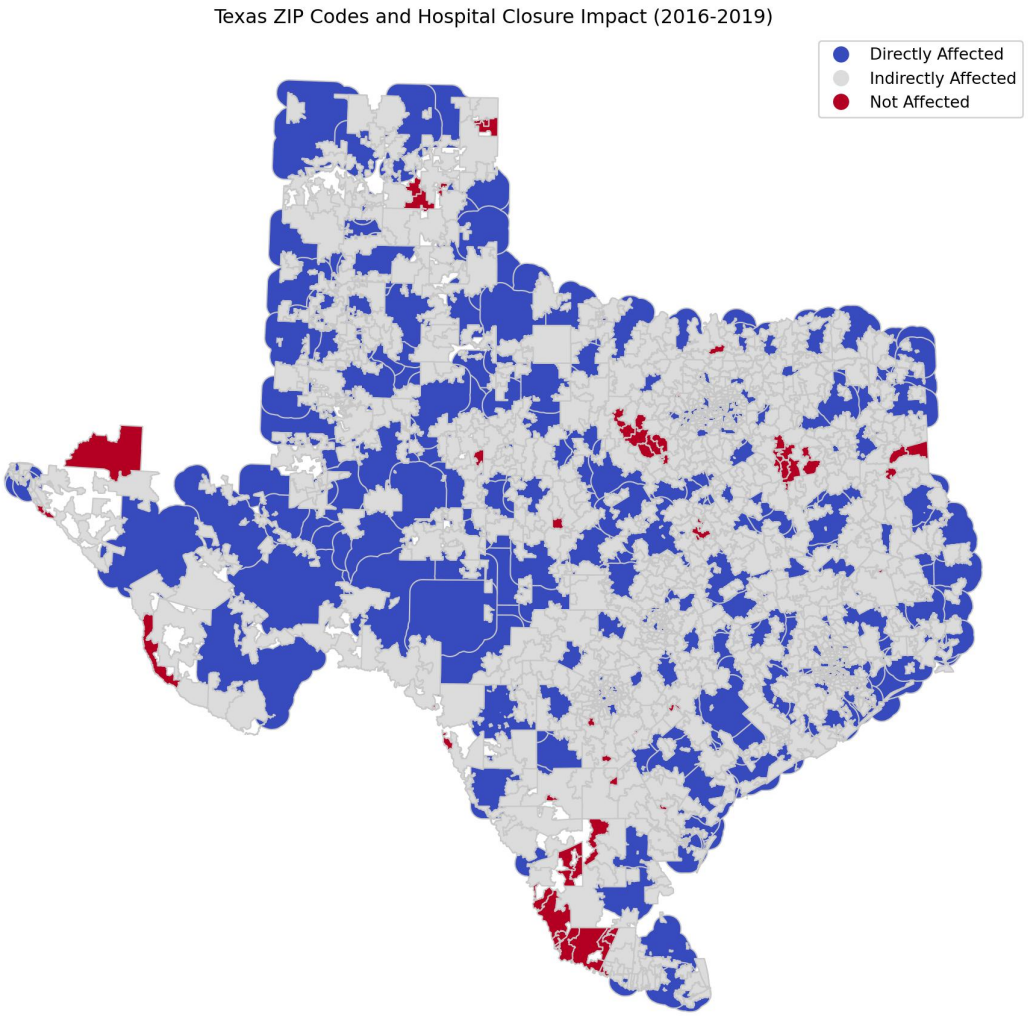
```
32918  8600000US78368      78368     78368     ZCTA5          216.341
32919  8600000US78412      78412     78412     ZCTA5            8.798
32920  8600000US78557      78557     78557     ZCTA5           11.653
32921  8600000US78586      78586     78586     ZCTA5          176.313

                                              geometry  ZIP_CD_left  \
9207   POLYGON ((-283928.442 829150.855, -283921.544 ...        78624
9208   POLYGON ((-154039.428 834177.743, -154758.105 ...        78626
9209   POLYGON ((-162023.498 834246.688, -162023.936 ...        78628
9210   POLYGON ((-300338.177 814903.567, -300350.066 ...        78631
9211   POLYGON ((-136239.104 743680.248, -136240.073 ...        78632
...                                                  ...          ...
32917  POLYGON ((-236303.891 741288.333, -236298.368 ...        78261
32918  POLYGON ((-182353.225 578314.536, -182213.379 ...        78368
32919  POLYGON ((-129571.289 516829.64, -129572.012 5...        78412
32920  POLYGON ((-222577.817 337608.335, -222584.563 ...        78557
32921  POLYGON ((-161212.554 350675.16, -160798.732 3...        78586

       index_right    GEO_ID_right ZCTA5_right NAME_right LSAD_right  \
9207           NaN             NaN         NaN        NaN        NaN
9208         862.0  8600000US78613       78613      78613      ZCTA5
9209         862.0  8600000US78613       78613      78613      ZCTA5
9210           NaN             NaN         NaN        NaN        NaN
9211           NaN             NaN         NaN        NaN        NaN
...            ...             ...         ...        ...        ...
32917          NaN             NaN         NaN        NaN        NaN
32918          NaN             NaN         NaN        NaN        NaN
32919          NaN             NaN         NaN        NaN        NaN
32920          NaN             NaN         NaN        NaN        NaN
32921          NaN             NaN         NaN        NaN        NaN

       CENSUSAREA_right  ZIP_CD_right  Number_of_Closures Closure_Status
9207                NaN           NaN                 NaN            NaN
9208             28.088       78613.0                 1.0       Closures
9209             28.088       78613.0                 1.0       Closures
9210                NaN           NaN                 NaN            NaN
9211                NaN           NaN                 NaN            NaN
...                 ...           ...                 ...            ...
32917               NaN           NaN                 NaN            NaN
32918               NaN           NaN                 NaN            NaN
32919               NaN           NaN                 NaN            NaN
32920               NaN           NaN                 NaN            NaN
32921               NaN           NaN                 NaN            NaN

[1997 rows x 16 columns]
Number of indirectly affected ZIP codes in Texas: 184
```

4.

```python
# Define file paths for shapefiles and hospital closure data
shapefile_path = r"C:\Users\james\Desktop\gz_2010_us_860_00_500k\gz_2010_us_860_00_500k.shp"
data_files = {
    2016: r"C:\Users\james\Desktop\problem-set-4-kj-partner-ps4\pos2016.csv",
    2017: r"C:\Users\james\Desktop\problem-set-4-kj-partner-ps4\pos2017.csv",
    2018: r"C:\Users\james\Desktop\problem-set-4-kj-partner-ps4\pos2018.csv",
    2019: r"C:\Users\james\Desktop\problem-set-4-kj-partner-ps4\pos2019.csv"
}

# Load the shapefile and filter it to include only Texas ZIP codes
gdf = gpd.read_file(shapefile_path)
gdf['ZCTA5'] = gdf['ZCTA5'].astype(str)
tx_gdf = gdf[gdf['ZCTA5'].str.startswith(('75', '76', '77', '78', '79'))]

# Set a consistent CRS for accurate distance measurements
tx_gdf = tx_gdf.to_crs(epsg=32614)

# Process and merge data from each year, focusing on short-term hospitals
yearly_data = {}
for year, file in data_files.items():
    data = pd.read_csv(file, encoding='ISO-8859-1', low_memory=False)
    data_filtered = data[(data['PRVDR_CTGRY_CD'] == 1) & (data['PRVDR_CTGRY_SBTYP_CD'] == 1)]
    data_filtered['Year'] = year
    yearly_data[year] = data_filtered[['PRVDR_NUM', 'FAC_NAME', 'PGM_TRMNTN_CD', 'ZIP_CD', 'Year']]

# Combine the filtered data and isolate closures specific to Texas
df_combined = pd.concat(yearly_data.values())
tx_closures = df_combined[df_combined['PGM_TRMNTN_CD'] != 0]

# Count the closures by ZIP code
closures_by_zip = tx_closures['ZIP_CD'].value_counts().reset_index()
closures_by_zip.columns = ['ZIP_CD', 'closure_count']

# Identify ZIP codes directly impacted by closures
directly_impacted = closures_by_zip[closures_by_zip['closure_count'] > 0]
directly_impacted['ZIP_CD'] = directly_impacted['ZIP_CD'].apply(lambda x: str(int(float(x))))

# Create GeoDataFrame for directly affected ZIP codes
directly_impacted_gdf = tx_gdf[tx_gdf['ZCTA5'].isin(directly_impacted['ZIP_CD'])].copy()
directly_impacted_gdf['category'] = 'Directly Affected'

# Create a 10-mile buffer around directly affected ZIP codes
```

```python
directly_impacted_gdf['geometry'] = directly_impacted_gdf.geometry.buffer(10 * 1609.34)

# Identify indirectly affected ZIP codes using spatial join
indirectly_impacted_gdf = gpd.sjoin(tx_gdf, directly_impacted_gdf, how="inner", predicate="intersects")
indirectly_impacted_gdf = indirectly_impacted_gdf[~indirectly_impacted_gdf['ZCTA5_left'].isin(directly_impacted_gdf['ZCTA5'])]
indirectly_impacted_gdf['category'] = 'Indirectly Affected'

# Identify ZIP codes that were not impacted
not_impacted_gdf = tx_gdf[~tx_gdf['ZCTA5'].isin(directly_impacted['ZIP_CD']) &
                          ~tx_gdf['ZCTA5'].isin(indirectly_impacted_gdf['ZCTA5_left'])]
not_impacted_gdf['category'] = 'Not Affected'

# Combine all ZIP code categories into one GeoDataFrame
combined_gdf = pd.concat([
    directly_impacted_gdf[['ZCTA5', 'geometry', 'category']],
    indirectly_impacted_gdf[['ZCTA5_left', 'geometry', 'category']].rename(columns={'ZCTA5_left': 'ZCTA5'}),
    not_impacted_gdf[['ZCTA5', 'geometry', 'category']]
])

# Ensure CRS consistency
combined_gdf = combined_gdf.to_crs(epsg=32614)

# Plot the map showing the impact of hospital closures
fig, ax = plt.subplots(1, 1, figsize=(12, 12))
combined_gdf.plot(column='category', cmap='coolwarm', linewidth=0.8, ax=ax, edgecolor='0.8', legend=True, categorical=True)

# Customize the plot
ax.set_title("Texas ZIP Codes and Hospital Closure Impact (2016-2019)")
ax.set_axis_off()

# Display the final plot
plt.show()
```

```
C:\Users\james\AppData\Local\Temp\ipykernel_19488\3371005749.py:23: 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-a-copy

C:\Users\james\AppData\Local\Temp\ipykernel_19488\3371005749.py:23: 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-a-copy

C:\Users\james\AppData\Local\Temp\ipykernel_19488\3371005749.py:23: 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-a-copy

C:\Users\james\AppData\Local\Temp\ipykernel_19488\3371005749.py:23: 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-a-copy

C:\Users\james\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-a-copy
```



Reflecting on the exercise (10 pts)

1. The first potential reason for the incorrect identified closure is temporary closurue or renovation. Hospital might be closed due to renovations for other temporary maintanace. They do not permanently shut down. It might be identified as permanent closure.

The second reason might be the mergers or acquisitions. This would lead to the change in CMS number. As a result, it would cause the incorrect closure identification.

The potential solution is to extend the observation period beyond a single year. A hospital appearing inactive for several consecutive years is more likely to be closed than one missing for a single year.

The second solution is to track the changes in CMS number. Track the changes in CMS numbers due to mergers or acquisitions would ensure the continuity in data.

2. This method does not reflect changes in zip-code-level access to hospitals well. As we can see from the earlier choropleths of the data, many zip codes do not have a hospital within them, or are far away from a hospital. These zip codes will be affected far more by nearby closures then zip codes that do contain hospitals or that are close to other zip codes that do.