```
top_10_closed = sorted_closed[['FAC_NAME', 'ZIP_CD',
 ↪  'Suspected_Closure_Year']].head(10)
print(top_10_closed)
```

```
                                        FAC_NAME   ZIP_CD  \
0                          ABRAZO MARYVALE CAMPUS  85031.0
1        ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230.0
73                         AFFINITY MEDICAL CENTER  44646.0
15   ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS  12208.0
29        ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE  75662.0
132                        ALLIANCE LAIRD HOSPITAL  39365.0
147                      ALLIANCEHEALTH DEACONESS  73112.0
109                 ANNE BATES LEACH EYE HOSPITAL  33136.0
3          ARKANSAS VALLEY REGIONAL MEDICAL CENTER  81050.0
11             BANNER CHURCHILL COMMUNITY HOSPITAL  89406.0

     Suspected_Closure_Year
0                      2017
1                      2017
73                     2018
15                     2017
29                     2017
132                    2019
147                    2019
109                    2019
3                      2017
11                     2017
```

3. a. & b.

```
def identify_valid_closed_hospitals(df, list):
    valid_list = []
    potential_mergers = 0

    active_hospitals_count = (
    df[df['PGM_TRMNTN_CD'] == 0]
    .groupby(['year', 'ZIP_CD'])
    .size()
    .unstack(fill_value=0)
)

    for _, closed_hospital in list.iterrows():
        zip_code = closed_hospital['ZIP_CD']
        closure_year = closed_hospital['Suspected_Closure_Year']
        next_year = closure_year + 1
```

```
        if closure_year < 2019:
            current_year_count = active_hospitals_count.loc[closure_year,
↪ zip_code]
            next_year_count = active_hospitals_count.loc[next_year,
↪ zip_code]

            if current_year_count > next_year_count:
                valid_list.append(closed_hospital)
            else:
                potential_mergers += 1

        else:
            valid_list.append(closed_hospital)

    valid_closed_hospitals = pd.DataFrame(valid_list)
    num_valid_closed = valid_closed_hospitals.shape[0]

    return valid_closed_hospitals, num_valid_closed, potential_mergers

results = identify_valid_closed_hospitals(data_all, sorted_closed)

print(f'Number of potential mergers: {results[2]}')
print(f'Number of valid suspected hospital closures: {results[1]}')
print(results[0].head())
```

```
Number of potential mergers: 97
Number of valid suspected hospital closures: 77
     PRVDR_NUM                          FAC_NAME   ZIP_CD  \
132     250159           ALLIANCE LAIRD HOSPITAL  39365.0
147     370032          ALLIANCEHEALTH DEACONESS  73112.0
109     100240     ANNE BATES LEACH EYE HOSPITAL  33136.0
153     390302     BARIX CLINICS OF PENNSYLVANIA  19047.0
172     670097  BAYLOR EMERGENCY MEDICAL CENTER  75087.0

     Suspected_Closure_Year
132                    2019
147                    2019
109                    2019
153                    2019
172                    2019
```

I also notice a case where both closure year and following year have zero active hospitals. If so, it does not indicate a potential merger but rather an absence of active hospitals in that area. In this case, the code will be different by adding "current_year_count > 0" as one more condition into the "if current_year_count > next_year_count:" part. So it would be, "if current_year_count > 0 and current_year_count > next_year_count:", and the result would be:

Number of potential mergers: 31; Number of valid suspected hospital closures: 143.

I don't put this condition into my actual code part and get the result of merger 97, closure 77, which is also used in section5 plots, since the question focuses on whether there is a degree. This is also a limitation of the "first-pass" method mentioned in section6.

c.

```
valid_sorted_closed = results[0].sort_values(by='FAC_NAME')
top_10_closed = valid_sorted_closed[['FAC_NAME', 'ZIP_CD',
 ↪  'Suspected_Closure_Year']].head(10)
print(top_10_closed)
```

```
                                          FAC_NAME   ZIP_CD  \
132                          ALLIANCE LAIRD HOSPITAL  39365.0
147                        ALLIANCEHEALTH DEACONESS  73112.0
109                    ANNE BATES LEACH EYE HOSPITAL  33136.0
153                    BARIX CLINICS OF PENNSYLVANIA  19047.0
172                  BAYLOR EMERGENCY MEDICAL CENTER  75087.0
170  BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...  78613.0
145                       BELMONT COMMUNITY HOSPITAL  43906.0
135                           BIG SKY MEDICAL CENTER  59716.0
134              BLACK RIVER COMMUNITY MEDICAL CENTER  63901.0
160                       CARE REGIONAL MEDICAL CENTER  78336.0

     Suspected_Closure_Year
132                    2019
147                    2019
109                    2019
153                    2019
172                    2019
170                    2019
145                    2019
135                    2019
134                    2019
160                    2019
```

## Download Census zip code shapefile (10 pt)

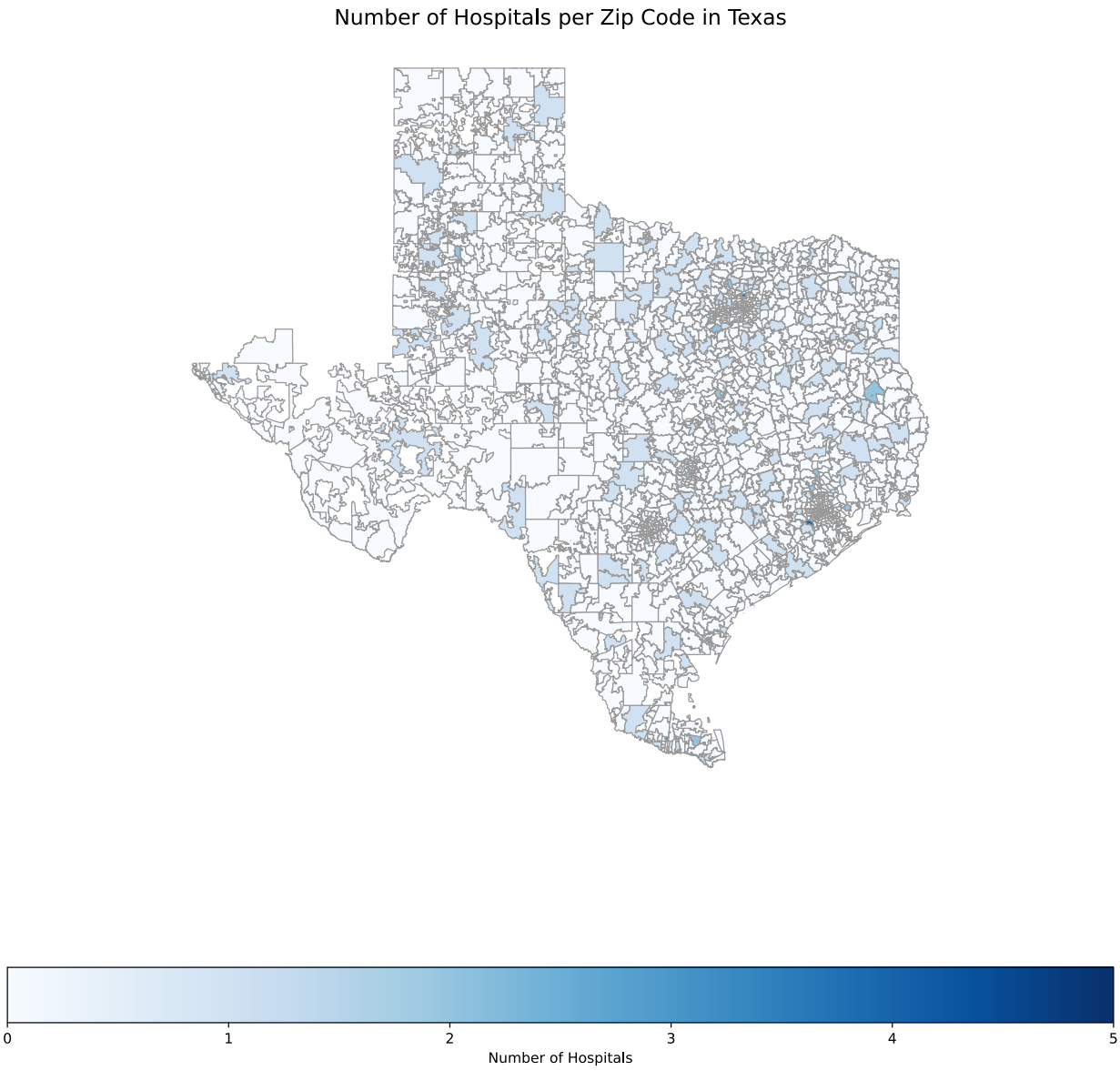
1. a. .dbf is a database file which contains attribute data in a tabular format. Each record in it corresponds to a geographical feature. .prj is a projection file which defines the coordinate system and projection information for the shapefile. .shp is a shapefile which is the main file containing the geometry of each feature, including the spatial data that represent the shapes of geographical features. .shx is a shape index file which provides an index of the geonmetry in the .shp file, helping to improve the loading of spatial data. .xml is a metadata file which contains metadata and provides descriptive information about the dataset.

b. .dbf: 6.4 MB .prj: 165 bytes .shp: 837.5 MB .shx: 265 KB .xml: 16 KB As seen, the .shp file is the largest, the .shx file and .xml file are relatively small, and the .prj file is the smallest.

2.

```
filepath = "data/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k.shp"
data_zip = gpd.read_file(filepath)
data_zip.head(100)

texas_prefixes = ('75', '76', '77', '78', '79')
zips_texas = data_zip[data_zip['ZCTA5'].str.startswith(texas_prefixes)]

estfile = 'data'
data = pd.read_csv(f'{estfile}/pos2016.csv')
hospitals_2016 = data[
    (data['PGM_TRMNTN_CD'] == 0) &
    (data['PRVDR_CTGRY_SBTYP_CD'] == 1) &
    (data['PRVDR_CTGRY_CD'] == 1)
].copy()

hospitals_2016['ZIP_CD'] = hospitals_2016['ZIP_CD'].astype(str).str[:5]
hospitals_2016['ZIP_CD'] = hospitals_2016['ZIP_CD'].str.zfill(5)
hospitals_texas =
 ↪  hospitals_2016[hospitals_2016['ZIP_CD'].str.startswith(texas_prefixes)].copy()

hospitals_per_zip =
 ↪  hospitals_texas.groupby('ZIP_CD')['PRVDR_NUM'].nunique().reset_index()
hospitals_per_zip.columns = ['ZIP_CD', 'hospital_count']

zips_texas = zips_texas.copy()
zips_texas = zips_texas.merge(hospitals_per_zip, left_on='ZCTA5',
 ↪  right_on='ZIP_CD', how='left')
zips_texas['hospital_count'] = zips_texas['hospital_count'].fillna(0)

fig, ax = plt.subplots(figsize=(14, 14))
zips_texas.plot(
    column='hospital_count',
    cmap='Blues',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.6',
    legend=True,
    legend_kwds={'label': "Number of Hospitals", 'orientation':
 ↪  "horizontal"}
)
ax.set_title('Number of Hospitals per Zip Code in Texas', fontsize=15)
```

```
ax.set_axis_off()
plt.show()
```




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

1.

```
data_zip['centroid'] = data_zip.geometry.centroid

zips_all_centroids = gpd.GeoDataFrame(data_zip[['ZCTA5', 'centroid']],
 ↪  geometry='centroid', crs=data_zip.crs)
```

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

```
/var/folders/29/92n8lbb16qsc1h27rdtp7j8c0000gn/T/ipykernel_11697/3330900437.py:1:
UserWarning:

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


Dimensions of zips_all_centroids: (33120, 2)
```

The resulting GeoDataFrame has 33,120 rows and 2 columns. ZCTA5: it is a distinct ZIP code area in U.S. Centroid: it is a geometry column that contains the geographic center point of the zip code area as a point geometry. Each point represents the central location within the boundaries of this row's zip code polygon.

2.

```
zips_texas_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(texas_prefixes)]
num_texas_zip = zips_texas_centroids['ZCTA5'].nunique()
print("The Number of unique zip codes in Texas:", num_texas_zip)

border_states_prefixes = texas_prefixes + ('70', '71', '72', '73', '74',
 ↪  '87', '88')
zips_texas_borderstates_centroids =
 ↪  zips_all_centroids[zips_all_centroids['ZCTA5'].str.startswith(border_states_prefixes)]
num_borderstate_zip = zips_texas_borderstates_centroids['ZCTA5'].nunique()
print("The Number of unique zip codes in Texas and bordering states:",
 ↪  num_borderstate_zip)
```

```
The Number of unique zip codes in Texas: 1935
The Number of unique zip codes in Texas and bordering states: 4057
```

3.

```
hospital_boarderstates =
 ↪  hospitals_2016[hospitals_2016['ZIP_CD'].str.startswith(border_states_prefixes)]
hospitals_per_zip =
 ↪  hospital_boarderstates['ZIP_CD'].value_counts().reset_index()
hospitals_per_zip.columns = ['ZIP_CD', 'hospital_count']

hospitals_per_zip['ZIP_CD'] =
 ↪  hospitals_per_zip['ZIP_CD'].astype(str).str.replace('.0', '',
 ↪  regex=False)
```