Brenda Castaneda and Ahona Roy

Partner 1: Brenda Castaneda, brendac29 Partner 2: Ahona Roy, ahona1 "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: B.C., A.R "I have uploaded the names of anyone else other than my partner and I worked with on the problem set" Late coins used this pset: 1. Late coins left after submission: Ahona: 3/Brenda: 2

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

1.

```
#setup
import pandas as pd
import os
import geopandas as gpd
import altair as alt
import matplotlib.pyplot as plt
import time

#set working directory
os.chdir('C:\\Users\\Brenda\\Documents\\DAPII\\') #change to your personal
 ↪  directory
#read in data
hospitals_16 = pd.read_csv('pos2016.csv')
```

1. We pulled Provider Category Code, Provider Subcategory Code, Facility Name, Provider Number, State Code, Zip Code, and Termination Code.

2.

```
#add year column
hospitals_16['year'] = 2016

#subset short term hospitals 2016
short_term_16 = hospitals_16[(hospitals_16['PRVDR_CTGRY_CD'] == 1) &
 ↪  (hospitals_16['PRVDR_CTGRY_SBTYP_CD'] == 1)]
short_term_16.shape
```

```
(7245, 8)
```

a. There are 7,245 hospitals reported in this data. This number seems off as the article states there are "nearly 5,000 short-term, acute care hospitals in the U.S."
b. According to the American Hospital Association, there were 6,120 short-term community hospitals in January 2024. The difference between the number of hospitals in the data and the cross reference is discrepancies the in the data. There may be hospitals that have closed that are not accounted for and/or the same hospital might appear twice in the data. https://www.aha.org/statistics/fast-facts-us-hospitals

3.

```
#2017 data + add year column
hospitals_17 = pd.read_csv('pos2017.csv')
hospitals_17['year'] = 2017

#2018 data + add year column
hospitals_18 = pd.read_csv('pos2018.csv', encoding='ISO-8859-1')
#I had trouble loading data, so I copied error: "UnicodeDecodeError: 'utf-8'
 ↪  codec can't decode byte 0x98 in position 11674: invalid start byte" to
 ↪  CGPT to debug
hospitals_18['year'] = 2018

#2019 data + add year column
hospitals_19 = pd.read_csv('pos2019.csv', encoding='ISO-8859-1')
hospitals_19['year'] = 2019

#append dataframes
hospitals = pd.concat([hospitals_16, hospitals_17, hospitals_18,
 ↪  hospitals_19], ignore_index=True)
```

```
#subset short term hospitals
short_term = hospitals[(hospitals['PRVDR_CTGRY_CD'] == 1) &
 ↪  (hospitals['PRVDR_CTGRY_SBTYP_CD'] == 1)]
```

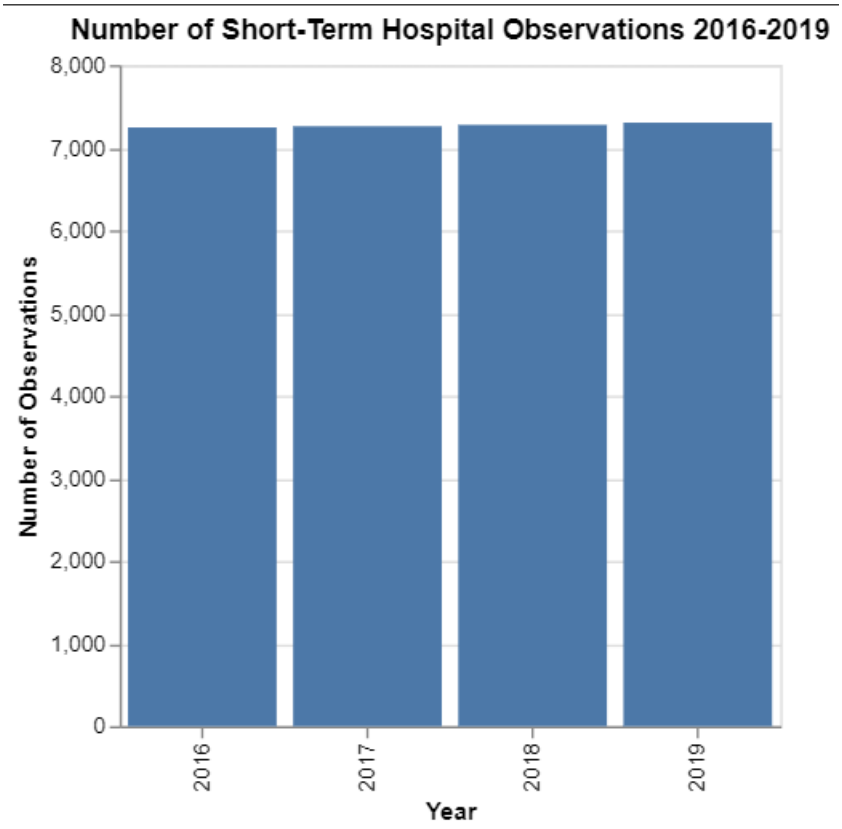
```
#group hospitals by year
hospitals_grouped =
 ↪  short_term.groupby('year').size().reset_index(name='count')

#plot number of short term observations for each year
observations = alt.Chart(hospitals_grouped).mark_bar().encode(
    x = alt.X('year:N', title = "Year"),
    y = alt.Y('count:Q', title = "Number of Observations")
).properties(
    title = "Number of Short-Term Hospital Observations 2016-2019",
    width = 300
)
display(observations)
```

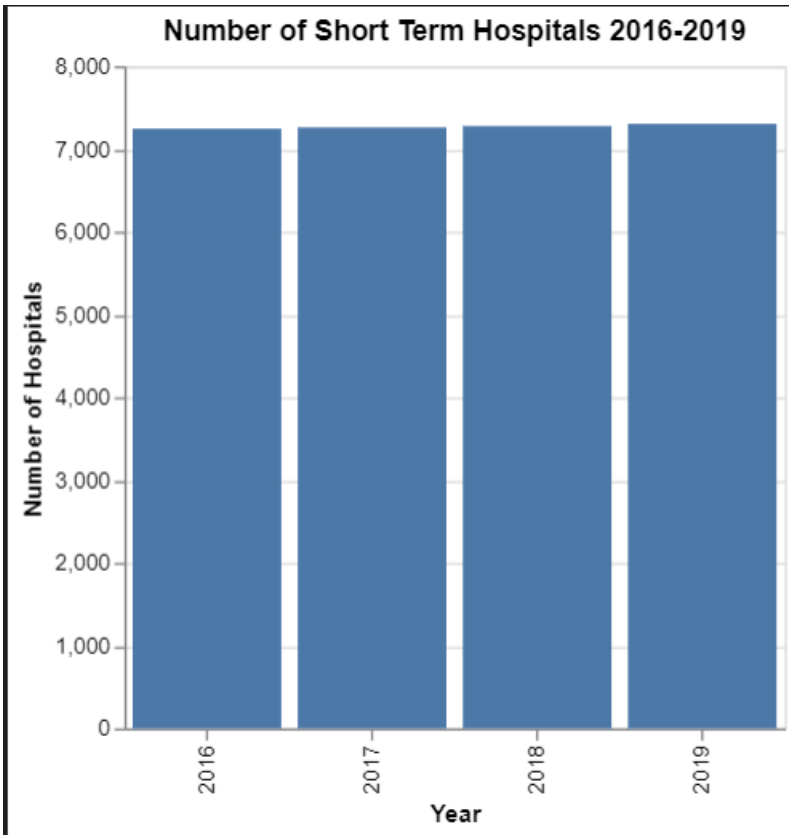
```
alt.Chart(...)
```

4. a.

```
#group by year and find unique CMS numbers
hospitals_unique =
 ↪  short_term.groupby('year')['PRVDR_NUM'].nunique().reset_index(name='unique_count')

#plot number of short term hospitals for each year
alt.Chart(hospitals_unique).mark_bar().encode(
    x = alt.X('year:N', title = "Year"),
    y = alt.Y('unique_count:Q', title = "Number of Hospitals")
).properties(
    title = "Number of Short Term Hospitals 2016-2019",
    width = 300
)
plt.show()
```

b. The number of observations in the previous plot and the number of unique hospitals is the same. This means that each row corresponds to a hospital with a unique CMS number.


Number of Short-Term Hospital Observations 2016-2019
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
Number of Observations
2016
2017
2018
2019
Year


Number of Short Term Hospitals 2016-2019
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
Number of Hospitals
2016
2017
2018
2019
Year

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

1.

```
#remove trailing zero and change to string type
short_term['ZIP_CD'] = short_term['ZIP_CD'].astype(str).str.replace('.0', '',
 ↪  regex=False)
#asked CGPT: How can i remove trailing zeros from data in a column

#reshape data so each row is a unique hospital-year status combo
hospitals_wide = short_term.pivot(index = ['PRVDR_NUM', 'ZIP_CD', 'FAC_NAME',
 ↪  'STATE_CD'], columns = 'year', values='PGM_TRMNTN_CD')
hospitals_wide.reset_index(inplace=True)

#filter hospitals that were active in 2016 and not active or missing by 2019
closures = hospitals_wide[
    (hospitals_wide[2016] ==0) & #active 2016
    ((hospitals_wide[2019] !=0) | (hospitals_wide[2019].isna())) #inactive or
 ↪  missing in 2019
]

#find the suspected year of closure
suspected_closures = []

for index, row in closures.iterrows():
    if row[2017] != 0 or pd.isna(row[2017]):
        suspected_closures.append((row['FAC_NAME'], row['ZIP_CD'], 2017))
    elif row[2018] != 0 or pd.isna(row[2018]):
        suspected_closures.append((row['FAC_NAME'], row['ZIP_CD'], 2018))
    elif row[2019] != 0 or pd.isna(row[2019]):
        suspected_closures.append((row['FAC_NAME'], row['ZIP_CD'], 2019))
suspected_closures = pd.DataFrame(suspected_closures, columns=['FAC_NAME',
 ↪  'ZIP_CD', 'YEAR_CLOSURE'])
suspected_closures.shape
```

```
C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\2677875424.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
```

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

```
(685, 3)
```

There are 685 hospitals that fit this definition.

2.

```
# Sort suspected closures by hospital name
suspected_closures = suspected_closures.sort_values(by='FAC_NAME')

#report first 10 rows
print(suspected_closures.head(10))
```

```
                                                FAC_NAME ZIP_CD  YEAR_CLOSURE
12                                ABRAZO MARYVALE CAMPUS  85031          2017
677                                AD HOSPITAL EAST, LLC  77380          2017
341                              ADIRONDACK MEDICAL CENTER  12983          2019
234  ADVENTIST HEALTHCARE WASHINGTON ADVENTIST HOSP...  20912          2019
447                               ADVENTIST MEDICAL CENTER  97216          2018
34                                ADVENTIST MEDICAL CENTER  93230          2017
39             ADVENTIST MEDICAL CENTER - CENTRAL VALLEY  93230          2017
38                    ADVENTIST MEDICAL CENTER - REEDLEY  93654          2017
409                               AFFINITY MEDICAL CENTER  44646          2018
352       ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS  12208          2017
```

3.

```
#count the number of active hospitals in each zip code
active_hospitals = closures.groupby('ZIP_CD').agg(
    active_count_16 = (2016, lambda x: (x == 0).sum()),
    active_count_17 = (2017, lambda x: (x == 0).sum()),
    active_count_18 = (2018, lambda x: (x == 0).sum()),
    active_count_19 = (2019, lambda x: (x == 0).sum())
).reset_index()

#merge the suspected closures df with active hospital count df
merged_closures = suspected_closures.merge(active_hospitals, on='ZIP_CD',
 ↪  how='left')

#filter zipcodes with number of active hospitals to find m/a
fake_closures = merged_closures[
```

```
    ((merged_closures['YEAR_CLOSURE'] == 2017) &
     (merged_closures['active_count_18'] >=
 ↪  merged_closures['active_count_17'])) |
    ((merged_closures['YEAR_CLOSURE'] == 2018) &
     (merged_closures['active_count_19'] >=
 ↪  merged_closures['active_count_18']))
]
fake_closures.shape
```

```
(407, 7)
```

a. There are 407 hospitals that fit the definition of being a merger/acquisition.
b.

```
#remove fake closures from sus closures to get real closures
real_closures =
 ↪  suspected_closures[~suspected_closures['ZIP_CD'].isin(fake_closures['ZIP_CD'])]
# asked CGPT: "how can I remove observation that are in a dataframe from
 ↪  another dataframe"
real_closures.shape
```

```
(270, 3)
```

After correcting for this, there are 270 "real closures" left.
c.

```
#sort by name and report top 10 rows
real_closures = real_closures.sort_values(by='FAC_NAME')

#report top 10 rows
print(real_closures.head(10))
```

```
                                        FAC_NAME ZIP_CD  YEAR_CLOSURE
341                          ADIRONDACK MEDICAL CENTER  12983          2019
234  ADVENTIST HEALTHCARE WASHINGTON ADVENTIST HOSP...  20912          2019
289                            ALLIANCE LAIRD HOSPITAL  39365          2019
425                           ALLIANCEHEALTH DEACONESS  73112          2019
286              BAPTIST MEM HOSP/ GOLDEN TRIANGLE INC  39701          2019
478                      BARIX CLINICS OF PENNSYLVANIA  19047          2019
651                            BAY AREA MEDICAL CENTER  54143          2019
675              BAY AREA REGIONAL MEDICAL CENTER, LLC  77598          2018
```

```
88       BAY MEDICAL CENTER SACRED HEART HEALTH SYSTEM  32401          2019
82               BAYHEALTH - MILFORD MEMORIAL HOSPITAL  19963          2019
```

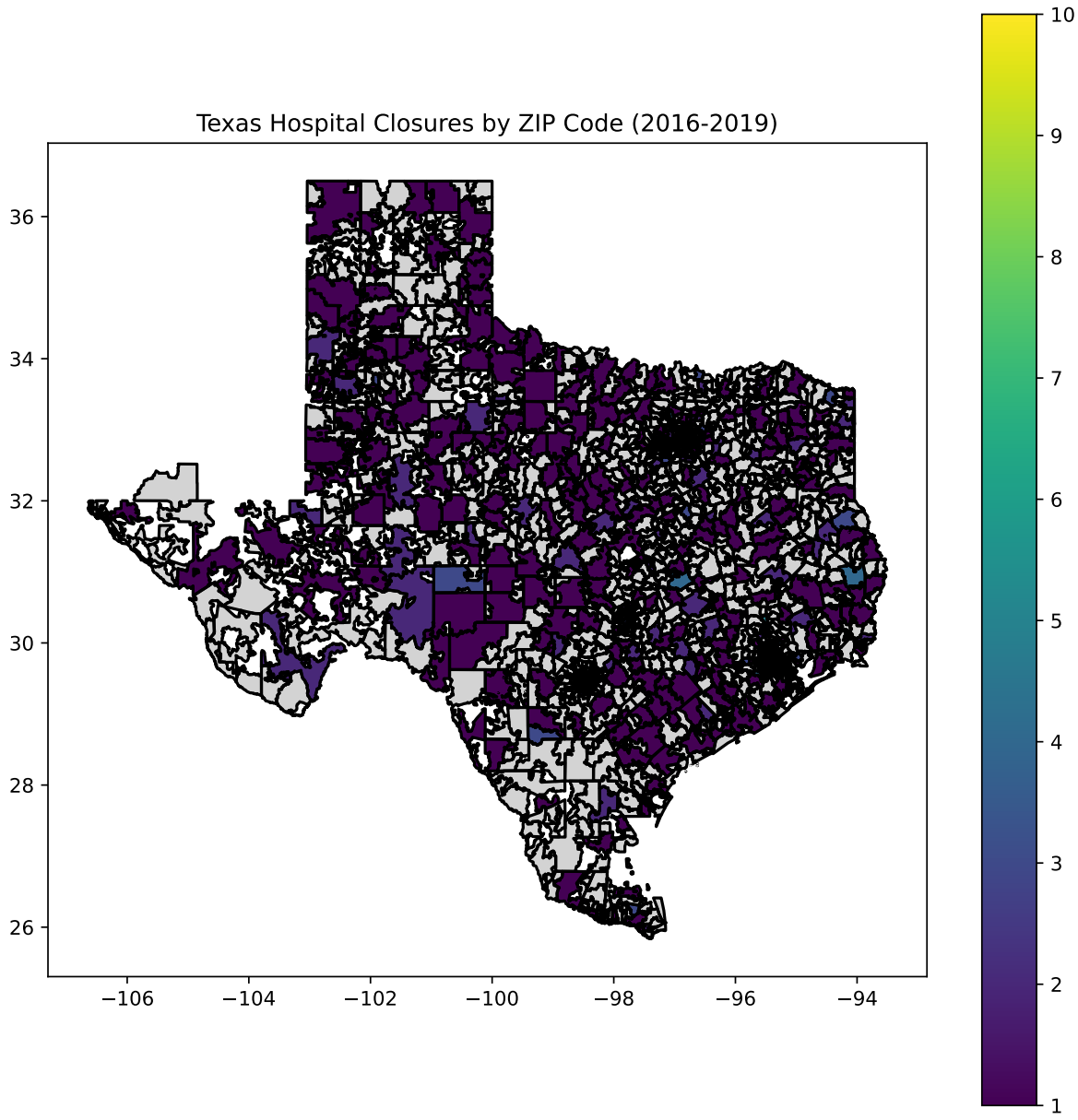
Download Census zip code shapefile (10 pt)

1. a. .dbf: Contains information about attributes of the shapes in the data. .prj: Contains information about the coordinate reference system. .shp: Contains the geometric feature data, it's the main data file and the largest. .shx: Contains the positional index of the geometric data. .xml: This file contains information about the data.
   b. The largest file is the .shp, about 0.8GB since it contains the main data file, followed by the .dbf file around 6,200 KB. The other files are much smaller, the .shx is 259KB, the .xml is 16KB, and the .prj is only 1KB.
2.

```
#load zip code data
zips = gpd.read_file('zips.shp')

#inspect columns
print(zips.columns) #ZCTA5 is zip code column

#restrict to TX zips
texas = zips[zips['ZCTA5'].str.startswith(tuple(['75', '76', '77', '78',
 ↪  '79']))]

#texas zips only
texas_hospitals = short_term_16[short_term_16['STATE_CD'] == 'TX']

#rename zip code column and change to string type
texas_hospitals.rename(columns={'ZIP_CD': 'ZCTA5'}, inplace=True)
texas_hospitals['ZCTA5'] =
 ↪  texas_hospitals['ZCTA5'].astype(str).str.replace('.0', '', regex=False)

#group by zip code
texas_hospitals =
 ↪  texas_hospitals.groupby('ZCTA5').size().reset_index(name='count')

#merge with texas zip code data
texas_merged = texas.merge(texas_hospitals, left_on='ZCTA5',
 ↪  right_on='ZCTA5', how='left')
```

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

C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\2046520361.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  texas_hospitals.rename(columns={'ZIP_CD': 'ZCTA5'}, inplace=True)
C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\2046520361.py:15:
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
  texas_hospitals['ZCTA5'] =
  texas_hospitals['ZCTA5'].astype(str).str.replace('.0', '', regex=False)
```

```
# plot choropleth of hospitals
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
texas_merged.boundary.plot(ax=ax, color='black')
texas_merged.plot(column='count', ax=ax, legend=True,
                  missing_kwds={'color': 'lightgrey', 'label': 'No
 ↪  closures'},
                  linewidth=0.2)
ax.set_title('Texas Hospital Closures by ZIP Code (2016-2019)')

# display map
plt.show()

# source, how to make choropleth using matplotlib:
 ↪  https://medium.com/@barua.aindriya/visualize-data-on-a-choropleth-map-with-geopandas-and-
```



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

1.

```
#calculate centroids
zips['centroid'] = zips.geometry.centroid
```

```
#create geodata frame with centroids
zips_all_centroids = gpd.GeoDataFrame(geometry=zips['centroid'])
zips_all_centroids['zip_code'] = zips['ZCTA5']

# get dimensions and columns
zips_all_centroids.shape
zips_all_centroids.head()
```

```
C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\3360450811.py:2:
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.

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

	geometry	zip_code
0	POINT (-72.64107 42.21257)	01040
1	POINT (-72.86985 42.28786)	01050
2	POINT (-72.71162 42.35349)	01053
3	POINT (-72.45805 42.19215)	01056
4	POINT (-72.3243 42.09165)	01057

The resulting dataframe has 33120 rows and 2 columns. The zip code column contains zip codes and the geometry column contains the pair of coordinate points of the center of the zip code area.

2.

```
#texas centroids
texas['centroid'] = texas.geometry.centroid
zips_texas_centroids=gpd.GeoDataFrame(geometry=texas['centroid'])
zips_texas_centroids['ZCTA5'] = texas['ZCTA5']

# unique tx zips
unique_texas_zips = zips_texas_centroids['ZCTA5'].nunique()
print("Unique ZIP codes in Texas:", unique_texas_zips)

# filter texas + border state
texas_borderstates = zips[zips['ZCTA5'].str.startswith(tuple([
```

```
'75', '76', '77', '78', '79', #texas
'87', '880', '881', '882', '883', '884', #new mexico
'73', '74', #oklahoma
'716', '717', '718', '719', '72', #arkansas
'70', '710', '711', '712', '713', '714', '715' #louisiana
]))]

#calculate tx + border state centroids
texas_borderstates['centroid'] = texas_borderstates.geometry.centroid
zips_texas_borderstates_centroids = gpd.GeoDataFrame(geometry =
 ↪  texas_borderstates['centroid'])
zips_texas_borderstates_centroids['ZCTA5'] = texas_borderstates['ZCTA5']

#unique zips in tx + border state
unique_border_states_zips =
 ↪  zips_texas_borderstates_centroids['ZCTA5'].nunique()
print("Unique ZIP codes in TX + border states:", unique_border_states_zips)
```

```
C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\1217432077.py:2:
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.

  texas['centroid'] = texas.geometry.centroid
C:\Users\Brenda\Documents\DAPII\.venv\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
  super().__setitem__(key, value)
C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\1217432077.py:20:
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.

  texas_borderstates['centroid'] = texas_borderstates.geometry.centroid

Unique ZIP codes in Texas: 1935
Unique ZIP codes in TX + border states: 4057
```

```
C:\Users\Brenda\Documents\DAPII\.venv\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
  super().__setitem__(key, value)
```

3.

```
#rename zip code column and change to string type
short_term_16.rename(columns={'ZIP_CD': 'ZCTA5'}, inplace=True)
short_term_16['ZCTA5'] = short_term_16['ZCTA5'].astype(str).str.replace('.0',
 ↪  '', regex=False)

#merge borderstate zips with hospitals data
zips_withhospitals_centroids =
 ↪  zips_texas_borderstates_centroids.merge(short_term_16, on = 'ZCTA5', how
 ↪  = 'inner')

#subset open hospitals in 2016
zips_withhospitals_centroids =
 ↪  zips_withhospitals_centroids[zips_withhospitals_centroids['PGM_TRMNTN_CD']==0]
```

```
C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\1796568577.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  short_term_16.rename(columns={'ZIP_CD': 'ZCTA5'}, inplace=True)
C:\Users\Brenda\AppData\Local\Temp\ipykernel_31236\1796568577.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
  short_term_16['ZCTA5'] =
  short_term_16['ZCTA5'].astype(str).str.replace('.0', '', regex=False)
```

This is an inner merge on the zip code variable (ZCTA5).

4. a.

```
#subset 10 zips
zips_10texas_centroids = zips_texas_centroids.head(10)

# Asked CGPT what this warning meant and how to resolve it:
# UserWarning: Geometry is in a geographic CRS. Results from 'sjoin_nearest'
 ↪  are likely incorrect.
# Use 'GeoSeries.to_crs()' to re-project geometries to a projected CRS before
 ↪  this operation.
# CGPT: use epsg to reproject CRS
zips_10texas_centroids = zips_10texas_centroids.to_crs(epsg=3857)
zips_withhospitals_centroids = zips_withhospitals_centroids.to_crs(epsg=3857)

#calculate distance
start_time = time.time() #start time
distance=gpd.sjoin_nearest(
    zips_10texas_centroids,
    zips_withhospitals_centroids,
    how='inner',
    distance_col = 'distance'
)
end_time = time.time() #end time
time_taken = end_time - start_time #how long did it take
print(f'The spatial join took: {time_taken} seconds')

#estimate time for entire procedure
estimated_time = (time_taken/10)*1935
print(f'The entire procedure will take about: {estimated_time} seconds')
```

```
The spatial join took: 0.0054781436920166016 seconds
The entire procedure will take about: 1.0600208044052124 seconds
```

b.

```
#reproject CRS
zips_texas_centroids = zips_texas_centroids.to_crs(epsg=3857)

#calculate distance
start_time = time.time() #start time
distance=gpd.sjoin_nearest(
    zips_texas_centroids,
```

```
    zips_withhospitals_centroids,
    how='inner',
    distance_col = 'distance'
)

end_time = time.time() #end time
time_taken = end_time - start_time #how long did it take
print(f'The spatial join took: {time_taken} seconds')
```

```
The spatial join took: 0.03196883201599121 seconds
```

I estimated the entire procedure would take longer than it did.

```
c. The units in the .prj file are in degrees. To convert degrees to miles, we
can multiply latitude by 69. To convert longitude to miles, mulitply
longitude degrees x 69 x cos(latitude). Since we reprojected the geographic
coordinates to CRS in meters in the previous stepts, we simply need to
convert the distance (in meters) to miles by dividing meters/1609.
https://sciencing.com/convert-degrees-latitude-miles-5744407.html
```

```
#convert distance column from meters to miles
distance['distance'] = distance['distance']/1609.34
```

5. a.

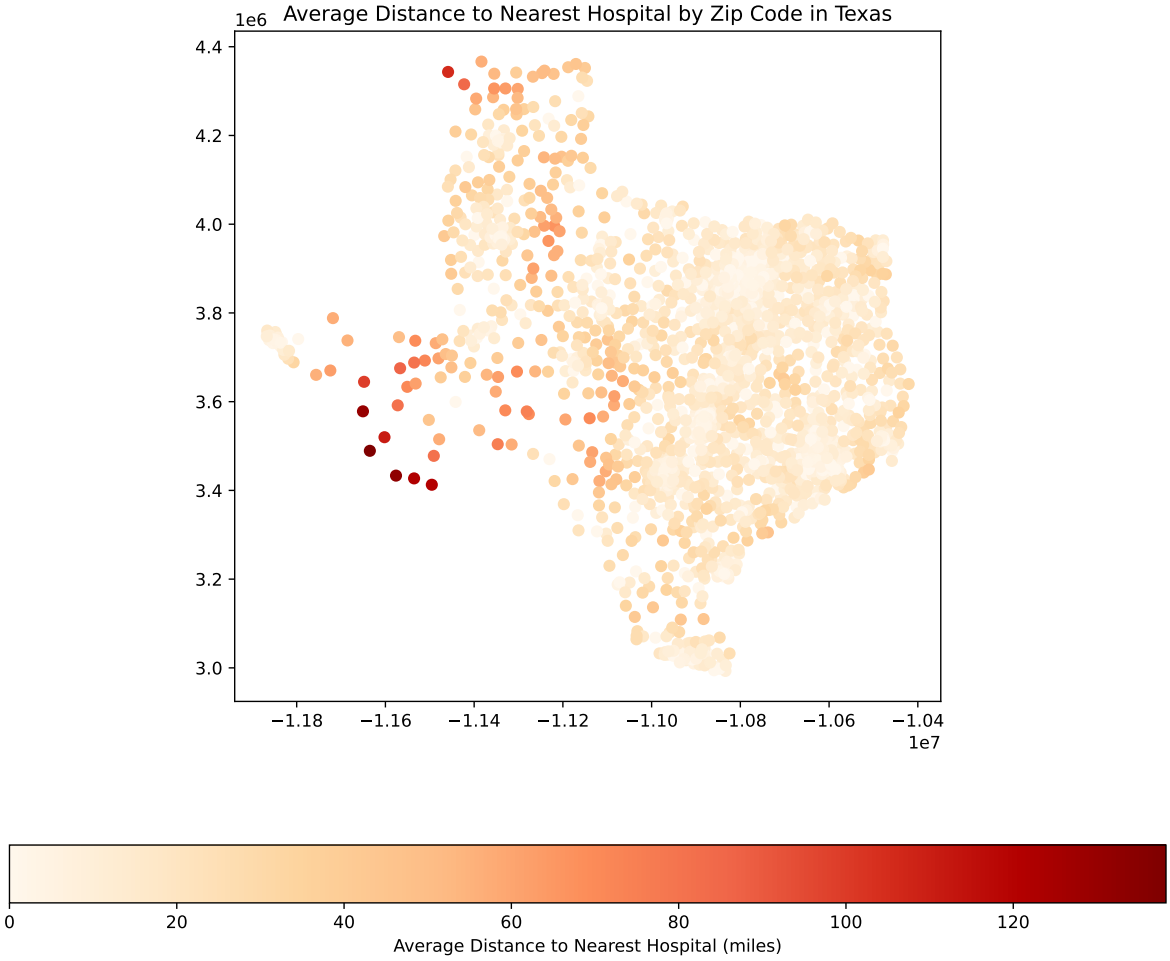
```
#avg distance distance to the nearest hospital for each tx zip
average_distance =
 ↪  distance.groupby('ZCTA5_left')['distance'].mean().reset_index()
```

The distance column in the previous question was converted to miles, therefore the units are in miles.

```
b. Yes, these values make sense.
c.
```

```
#merge tx zip centroids with average distance
merged = zips_texas_centroids.merge(
    average_distance,
    left_on='ZCTA5',
    right_on='ZCTA5_left',
    how='left'
```

```
)
#plot map of avg distance to nearest hospital for every zip code
fig, ax = plt.subplots(1, 1, figsize=(12, 10))
merged.plot(column='distance', ax=ax, legend=True,
            legend_kwds={'label': "Average Distance to Nearest Hospital
 ↪  (miles)",
                         'orientation': "horizontal"},
            cmap='OrRd',  # Choose a colormap
            missing_kwds={'color': 'lightgrey', 'label': 'No Data'})  #
 ↪  Handle missing values

ax.set_title('Average Distance to Nearest Hospital by Zip Code in Texas')
plt.show()
```



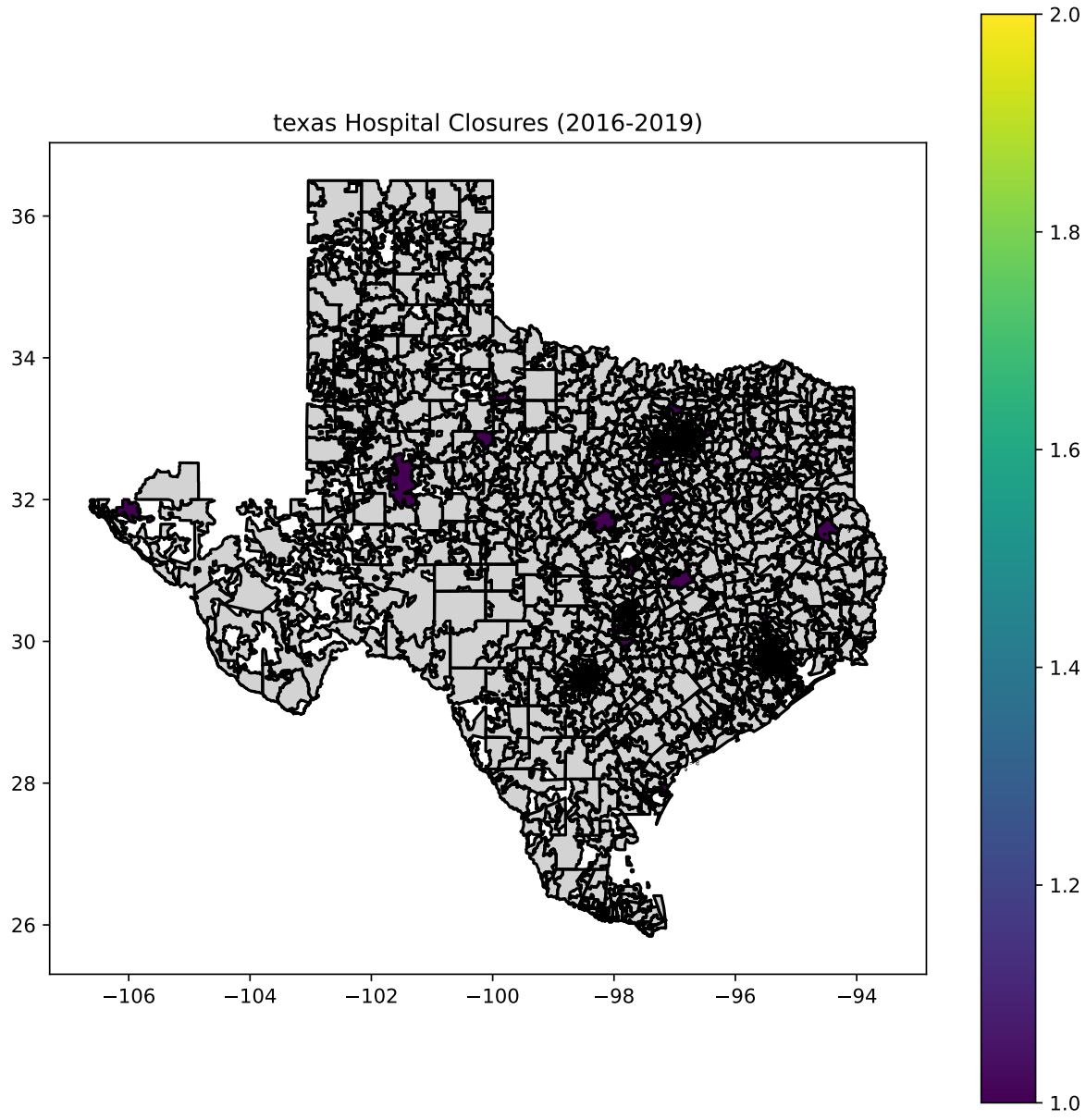
Effects of closures on access in Texas (15 pts)

1.

```
#subset texas hospital closures
texas_closures =
 ↪  real_closures[real_closures['ZIP_CD'].str.startswith(tuple(['75', '76',
 ↪  '77', '78', '79']))]

#group by zipcode and count closures per zip
texas_closures =
 ↪  texas_closures.groupby('ZIP_CD').size().reset_index(name='count')

texas_closures.rename(columns={'ZIP_CD': 'ZCTA5'}, inplace=True)
texas_closures['ZCTA5'] =
 ↪  texas_closures['ZCTA5'].astype(str).str.replace('.0', '', regex=False)
```

2.

```
#merge texas zips with texas closures
closures_merge = texas.merge(texas_closures, left_on = 'ZCTA5', right_on =
 ↪  'ZCTA5', how = 'left')

#plot choropleth of texas closures
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
closures_merge.boundary.plot(ax=ax, color='black')
closures_merge.plot(column='count', ax=ax, legend=True,
                    missing_kwds={'color': 'lightgrey', 'label': 'No
 ↪  closures'})

ax.set_title('texas Hospital Closures (2016-2019)')
plt.show()

#how many directly affected zips are there
directly_affected = closures_merge[closures_merge['count'].notnull()]
directly_affected.shape
```



There are 39 directly affected zip codes in Texas.

3.

```
# filter directly affected zips in tx
directly_affected = closures_merge[closures_merge['count'].notnull()]

# create gpd from directly affected
```

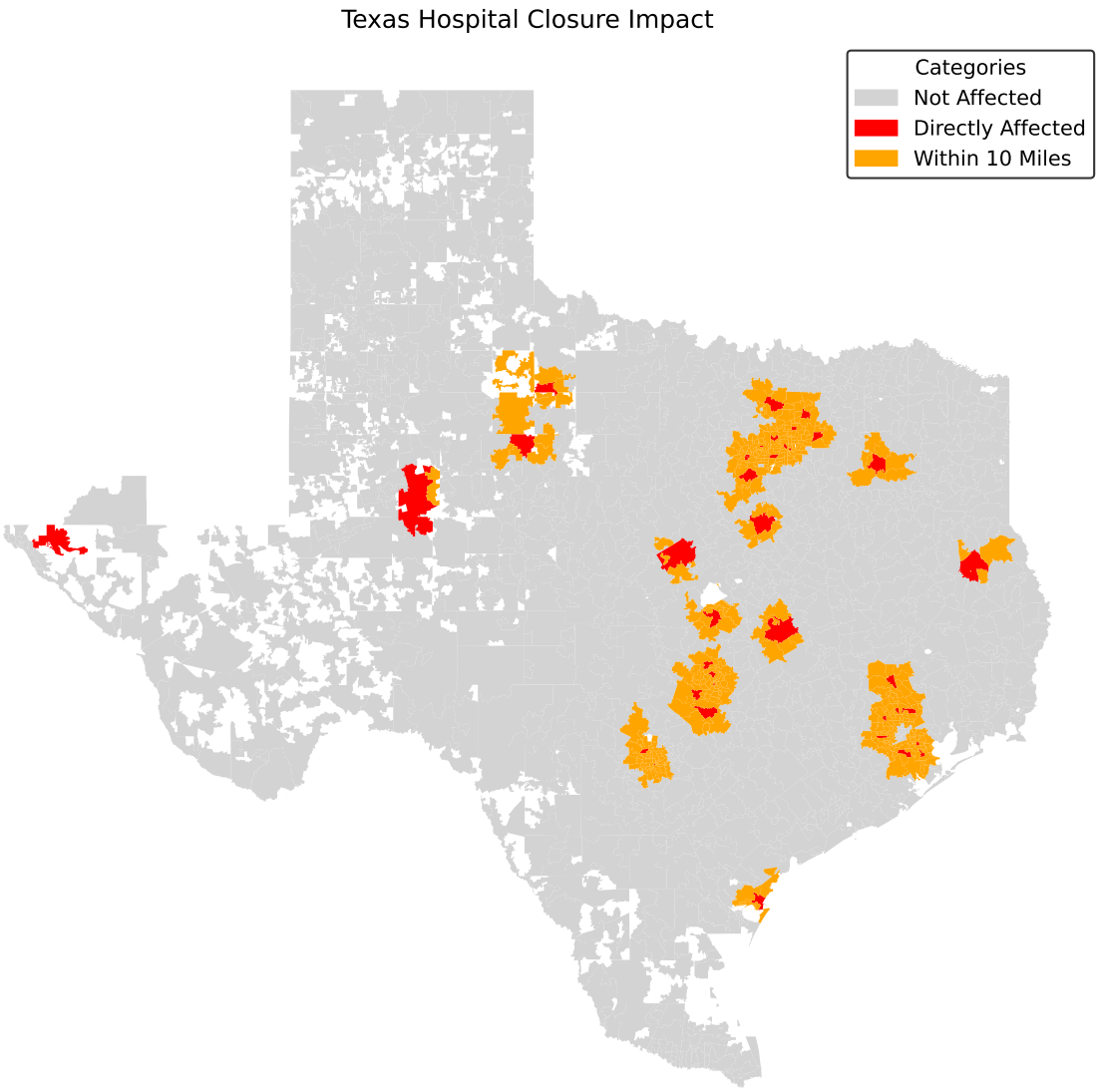
```
directly_affected_gdf = gpd.GeoDataFrame(directly_affected,
 ↪  geometry='centroid')
directly_affected_gdf = directly_affected_gdf.to_crs(epsg=3857)

# create 10 mile radius
buffer_distance = 10 * 1609.34
directly_affected_gdf['buffered_geometry'] =
 ↪  directly_affected_gdf.buffer(buffer_distance)

# reproject texas CRS
texas = texas.to_crs(epsg=3857)

indirectly_affected_zips = gpd.sjoin(texas,
 ↪  directly_affected_gdf.set_geometry('buffered_geometry'),
 ↪  predicate='intersects')
indirectly_affected_zips.head()

# Count indirectly affected zip codes
indirectly_affected_zip_codes =
 ↪  indirectly_affected_zips['ZCTA5_left'].unique()
num_indirectly_affected_zip_codes = len(indirectly_affected_zip_codes)

print(f'Total number of indirectly affected zip codes in TX:
 ↪  {num_indirectly_affected_zip_codes}')
```

```
Total number of indirectly affected zip codes in TX: 520
```

4.

```
import matplotlib.patches as mpatches

#create new column in texas zips to define categories
texas['category'] = 'Not Affected'
#categorize indirectly affected
texas.loc[texas['ZCTA5'].isin(indirectly_affected_zip_codes), 'category'] =
 ↪  'Within 10 Miles'
#categorize directly affected
texas.loc[texas['ZCTA5'].isin(directly_affected['ZCTA5']), 'category'] =
 ↪  'Directly Affected'

#set up color map
color_map = {
```

```
    'Not Affected': 'lightgray',
    'Directly Affected': 'red',
    'Within 10 Miles': 'orange'
}

#plot choropleth
fig, ax = plt.subplots(1, 1, figsize=(10, 10))
for category, color in color_map.items():
    texas[texas['category'] == category].plot(ax=ax, color=color)
handles = [mpatches.Patch(color=color, label=category) for category, color in
 ↪  color_map.items()]
ax.legend(handles=handles, title='Categories', loc='upper right',
 ↪  frameon=True, facecolor='white', edgecolor='black', fontsize=10)
#CGPT: I copied my plot code and asked "the legend does not show on my map,
 ↪  how can I display the legend?"
ax.set_title('Texas Hospital Closure Impact')
ax.set_axis_off()
plt.show()
```



Reflecting on the exercise (10 pts)

1. The first pass method, where we look at the number of hospitals that remain active over the years in each zip code might still not correctly capture real closures. For instance, this method may overlook hospitals that are in neighborhing zip codes if a merger between zip codes occurs. Our data also appears to record many closures in 2019 but we cannot verify if they were closed since we do not have 2020 data. We can analyze broader geographic areas instead of zip codes to more accurately capture real closures versus mergers/acquisitions in an area.

2. The closure of a hospital might not impact all ZIP codes equally. The zip codes do a good job in telling us where the closest hospital was. However, relying on centroids to measure the location of a hospital can be misleading as it does not give us information about the distribution of hospitals within a zip code. Regardless of the precise location of a hospital in a zipcode, if the hospital was within the zipcode, then the distance was marked by zero. Furthermore, looking at the closures doesn't give us more information about the accessibility of hospitals that remain open. A zip code can cover a large, irregular area, and its centroid might not accurately reflect the location of hospital facilities within it. One way we could improve this measure would be if we had access to the exact location of each hospital, then we could calculate exact distances, providing a more realistic measure of access.