Your Title

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

Style Points (10 pts)

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): Alejandra Silva - aosilva • Partner 2 (name and cnet ID): Guillermina Marto - gmarto
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: **___** **GM**
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.

```
import requests
import pandas as pd
import altair as alt

base_url = "https://data.cms.gov/data-api/v1/dataset/{uuid}/data"
uuid = "96ba2257-2080-49c1-9e5b-7726f9f83cad"

columns = [
    "PRVDR_CTGRY_CD",       # Provider Category Code
    "PRVDR_CTGRY_SBTYP_CD", # Provider Subtype Code
    "PRVDR_NUM",            # CMS Certification Number
    "PGM_TRMNTN_CD",        # Termination Code
    "FAC_NAME",             # Facility Name
    "ZIP_CD",               # ZIP Code
    "STATE_CD"              # State Abbreviation
]

columns_param = ",".join(columns)

offset = 0
limit = 5000  #  API allows size to be set to 5000

all_data = []

while True:
    params = {

        "column": columns_param,
        "size": limit,
        "offset": offset
    }

    url = base_url.format(uuid=uuid)
    response = requests.get(url, params=params)

    if response.status_code != 200:
        print(f"Error: {response.status_code}, {response.text}")
        break

    data = response.json()

    if not data:
        print("No more data available.")
        break

    all_data.extend(data)

    offset += limit
    print(f"Fetched {len(data)} rows, moving to next batch...")

df = pd.DataFrame(all_data)


df.to_csv("pos2016.csv", index=False)
```

```
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 5000 rows, moving to next batch...
Fetched 1557 rows, moving to next batch...
```

```
No more data available.
```

2.

```
df = pd.read_csv("pos2016.csv")


df_st_hospitals = df[
    (df["PRVDR_CTGRY_CD"] == 1) &
    (df["PRVDR_CTGRY_SBTYP_CD"] == 1)
]

num_hospitals = df_st_hospitals.shape[0]
print(f"Number of short-term hospitals reported in the data:
 ↪  {num_hospitals}")

print(df_st_hospitals)
```

```
Number of short-term hospitals reported in the data: 7245
        PRVDR_CTGRY_CD  PRVDR_CTGRY_SBTYP_CD PRVDR_NUM  PGM_TRMNTN_CD  \
0                    1                   1.0    010001              0
1                    1                   1.0    010004              1
2                    1                   1.0    010005              0
3                    1                   1.0    010006              0
4                    1                   1.0    010007              0
...                ...                   ...       ...            ...
133526               1                   1.0    670114              0
133527               1                   1.0    670115              0
133528               1                   1.0    670116              0
133529               1                   1.0    670117              0
133530               1                   1.0    670118              0

                                FAC_NAME   ZIP_CD STATE_CD
0       SOUTHEAST ALABAMA MEDICAL CENTER  36301.0       AL
1                 NORTH JACKSON HOSPITAL  35740.0       AL
2          MARSHALL MEDICAL CENTER SOUTH  35957.0       AL
3         ELIZA COFFEE MEMORIAL HOSPITAL  35631.0       AL
4               MIZELL MEMORIAL HOSPITAL  36467.0       AL
...                                  ...      ...      ...
133526             WEIMAR MEDICAL CENTER  78962.0       TX
133527      CLEVELAND EMERGENCY HOSPTIAL  77327.0       TX
133528                WISE HEALTH SYSTEM  76177.0       TX
133529  TEXAS GENERAL HOSPITAL- VZRMC LP  75140.0       TX
```

```
133530              FIRST TEXAS HOSPITAL  77070.0       TX

[7245 rows x 7 columns]
```

The number of short-term hospitals reported in the dataset for Q4 2016 is 7,245.

According to the American Hospital Association (AHA) Annual Survey, the estimated number of short-term hospitals is 4,500–5,000. Similarly, the CMS Hospital Compare dataset indicates around 4,800 hospitals.

The discrepancy could be due to the narrower definition used in our dataset and the timing of data collection, which only includes hospitals in Q4 2016. Additionally, the CMS dataset might not include hospitals that do not participate in Medicare or Medicaid, which could lead to lower numbers.

3.

```python
uuid_dict = {
    "2016Q4": "96ba2257-2080-49c1-9e5b-7726f9f83cad",
    "2017Q4": "d338dc0d-641c-486a-b586-88a662f36963",
    "2018Q4": "4ff7fcfb-2a40-4f76-875d-a4ac2aec268e",
    "2019Q4": "03cca0cc-13a0-4b8d-82c4-57185b6bbfbd"
}

columns = [
    "PRVDR_CTGRY_CD",          # Provider Category Code
    "PRVDR_CTGRY_SBTYP_CD",    # Provider Subtype Code
    "PRVDR_NUM",               # CMS Certification Number
    "PGM_TRMNTN_CD",           # Termination Code
    "FAC_NAME",                # Facility Name
    "ZIP_CD",                  # ZIP Code
    "STATE_CD"                 # State Abbreviation
]

columns_param = ",".join(columns)

combined_data = []

for year_quarter, uuid in uuid_dict.items():
    offset = 0
    limit = 5000
    all_data = []

    print(f"Fetching data for {year_quarter}...")
    while True:
        params = {
            "column": columns_param,
            "size": limit,
            "offset": offset
        }

        url = f"https://data.cms.gov/data-api/v1/dataset/{uuid}/data"
        response = requests.get(url, params=params)

        if response.status_code != 200:
            print(f"Error: {response.status_code}, {response.text}")
            break

        data = response.json()

        if not data:
            print("No more data available.")
            break

        all_data.extend(data)

        offset += limit
        print(f"Fetched {len(data)} rows for {year_quarter}, moving to next
         ↪  batch...")

    year_data = pd.DataFrame(all_data)
    year_data["Year"] = year_quarter[:4]

    # filtro por las condiciones
    year_data = year_data[
    (year_data["PRVDR_CTGRY_CD"] == "01") &
    (year_data["PRVDR_CTGRY_SBTYP_CD"] == "01")
]

    combined_data.append(year_data)

combined_df = pd.concat(combined_data, axis=0)

combined_df.to_csv("combined_data.csv", index=False)
```

```python
print(f"Total records retrieved across all years: {combined_df.shape[0]}")
```

```
Fetching data for 2016Q4...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 5000 rows for 2016Q4, moving to next batch...
Fetched 1557 rows for 2016Q4, moving to next batch...
No more data available.
Fetching data for 2017Q4...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 5000 rows for 2017Q4, moving to next batch...
Fetched 4110 rows for 2017Q4, moving to next batch...
No more data available.
Fetching data for 2018Q4...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 5000 rows for 2018Q4, moving to next batch...
Fetched 1696 rows for 2018Q4, moving to next batch...
No more data available.
Fetching data for 2019Q4...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 5000 rows for 2019Q4, moving to next batch...
Fetched 4206 rows for 2019Q4, moving to next batch...
No more data available.
```

```
Total records retrieved across all years: 29085
```

```
import altair as alt

# Plotting Number of Observations Per Year

combined_year_df =
 ↪  combined_df.groupby("Year").size().reset_index(name="Number of
 ↪  Observations")

obs_chart = alt.Chart(combined_year_df).mark_bar().encode(
    x=alt.X("Year:O", title="Year"),
    y=alt.Y("Number of Observations:Q", title="Number of Observations"),
    tooltip=["Year", "Number of Observations"]
).properties(
    title="Number of Observations Per Year"
)

obs_chart.display()


# Plotting Number of Unique Hospitals Per Year
unique_hospitals =
 ↪  combined_df.groupby("Year")["PRVDR_NUM"].nunique().reset_index(name="Number
 ↪  of Unique Hospitals")

unique_hospitals_chart = alt.Chart(unique_hospitals).mark_bar().encode(
    x=alt.X("Year:O", title="Year"),
    y=alt.Y("Number of Unique Hospitals:Q", title="Number of Unique
 ↪  Hospitals"),
    tooltip=["Year", "Number of Unique Hospitals"]
).properties(
    title="Number of Unique Hospitals Per Year"
)

unique_hospitals_chart.display()


#print("Observations Per Year:")
#print(observations_per_year)
#print("\nUnique Hospitals Per Year:")
#print(unique_hospitals_per_year)
```

```
# Compare the two plots to understand the structure of the data.
# Observations per year may be higher due to multiple records for the same
↪ hospital.
# Unique hospitals per year give an idea of how many distinct hospitals are
↪ in the dataset for each year.
```

```
/opt/anaconda3/lib/python3.12/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``.
```

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

```
/opt/anaconda3/lib/python3.12/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``.
```

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

4. a.
   b.

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

1. Termination code equal to 00=ACTIVE PROVIDER. The data contain only up to the code 07. The other codes apply to CLIA.

```
combined_df["PGM_TRMNTN_CD"] = combined_df["PGM_TRMNTN_CD"].astype(str)

inactive_codes = ["01", "02", "03", "04", "05", "06", "07"]

active_2016 = combined_df[(combined_df["Year"] == "2016") &
↪ (combined_df["PGM_TRMNTN_CD"] == "00")]

print(active_2016.head())
```

```
suspected_closures = []

for idx, hospital in active_2016.iterrows():
    provider_num = hospital["PRVDR_NUM"]
    facility_name = hospital["FAC_NAME"]
    zip_code = hospital["ZIP_CD"]

    for year in ["2017", "2018", "2019"]:
        yearly_data = combined_df[(combined_df["PRVDR_NUM"] == provider_num)
 ↪  & (combined_df["Year"] == year)]

        if yearly_data.empty or
         ↪  yearly_data["PGM_TRMNTN_CD"].isin(inactive_codes).any():
            suspected_closures.append({
                "Provider Number": provider_num,
                "Facility Name": facility_name,
                "ZIP Code": zip_code,
                "Year Closed": year
            })
            break

suspected_closures_df = pd.DataFrame(suspected_closures)
num_closures = len(suspected_closures_df)

display(f"Total suspected hospital closures: {num_closures}")
display(suspected_closures_df.head())
```

```
  PRVDR_CTGRY_CD PRVDR_CTGRY_SBTYP_CD PRVDR_NUM PGM_TRMNTN_CD  \
0             01                   01    010001            00
2             01                   01    010005            00
3             01                   01    010006            00
4             01                   01    010007            00
5             01                   01    010008            00

                          FAC_NAME ZIP_CD STATE_CD  Year
0  SOUTHEAST ALABAMA MEDICAL CENTER  36301       AL  2016
2     MARSHALL MEDICAL CENTER SOUTH  35957       AL  2016
3    ELIZA COFFEE MEMORIAL HOSPITAL  35631       AL  2016
4           MIZELL MEMORIAL HOSPITAL  36467       AL  2016
5       CRENSHAW COMMUNITY HOSPITAL  36049       AL  2016

'Total suspected hospital closures: 174'
```

	Provider Number	Facility Name	ZIP Code	Year Closed
0	010032	WEDOWEE HOSPITAL	36278	2019
1	010047	GEORGIANA MEDICAL CENTER	36033	2019
2	010146	RMC JACKSONVILLE	36265	2018
3	010172	NORTH ALABAMA SPECIALITY HOSPITAL	35611	2018
4	030001	ABRAZO MARYVALE CAMPUS	85031	2017

2.

```
sorted_closures = suspected_closures_df.sort_values(by="Facility Name")

top_10_closures = sorted_closures[["Facility Name", "Year Closed"]].head(10)

display(top_10_closures)
```

	Facility Name	Year Closed
4	ABRAZO MARYVALE CAMPUS	2017
10	ADVENTIST MEDICAL CENTER - CENTRAL VALLEY	2017
97	AFFINITY MEDICAL CENTER	2018
80	ALBANY MEDICAL CENTER / SOUTH CLINICAL CAMPUS	2017
140	ALLEGIANCE SPECIALTY HOSPITAL OF KILGORE	2017
62	ALLIANCE LAIRD HOSPITAL	2019
101	ALLIANCEHEALTH DEACONESS	2019
26	ANNE BATES LEACH EYE HOSPITAL	2019
21	ARKANSAS VALLEY REGIONAL MEDICAL CENTER	2017
69	BANNER CHURCHILL COMMUNITY HOSPITAL	2017

3.

```
combined_df["Year"] = combined_df["Year"].astype(int) # Ensure the Year
 ↪  column is properly converted to integer
combined_df["ZIP_CD"] = combined_df["ZIP_CD"].astype(str)  # ZIP code as
 ↪  string

# Check the data types to confirm they are as expected
print(combined_df.dtypes)  # This should show Year as int and ZIP_CD as
 ↪  object (string)

# Initialize list to store true closures
true_closures = []
```

```
# Loop through each suspected closure
for idx, row in suspected_closures_df.iterrows():
    provider_num = row["Provider Number"]
    zip_code = str(row["ZIP Code"])  # Convert ZIP code to string
    year_closed = int(row["Year Closed"])  # Ensure closure year is an
 ↪  integer

    # Count active hospitals in the closure year and the year after
    active_in_year = combined_df[(combined_df["ZIP_CD"] == zip_code) &
                                 (combined_df["Year"] == year_closed) &
                                 (combined_df["PGM_TRMNTN_CD"] ==
 ↪  "00")].shape[0]

    active_in_next_year = combined_df[(combined_df["ZIP_CD"] == zip_code) &
                                      (combined_df["Year"] == (year_closed +
 ↪  1)) &
                                      (combined_df["PGM_TRMNTN_CD"] ==
 ↪  "00")].shape[0]

    # Keep closure if the number of active hospitals decreases
    if active_in_next_year < active_in_year:
        true_closures.append(row)

# Convert true closures to DataFrame
true_closures_df = pd.DataFrame(true_closures)
num_true_closures = len(true_closures_df)

display(f"Total true hospital closures: {num_true_closures}")
display(true_closures_df)
```

```
PRVDR_CTGRY_CD          object
PRVDR_CTGRY_SBTYP_CD    object
PRVDR_NUM               object
PGM_TRMNTN_CD           object
FAC_NAME                object
ZIP_CD                  object
STATE_CD                object
Year                     int64
dtype: object
```

```
'Total true hospital closures: 26'
```

	Provider Number	Facility Name	ZIP Code
9	050153	O'CONNOR HOSPITAL	95128
16	050751	MIRACLE MILE MEDICAL CENTER	90036
24	100009	UNIVERSITY OF MIAMI HOSPITAL	33136
26	100240	ANNE BATES LEACH EYE HOSPITAL	33136
30	110004	HUTCHESON MEDICAL CENTER	30742
31	110039	TRINITY HOSPITAL OF AUGUSTA	30904
33	110187	CHESTATEE REGIONAL HOSPITAL	30533
34	130067	IDAHO DOCTORS HOSPITAL	83221
36	150175	THE HEART HOSPITAL AT DEACONESS GATEWAY LLC	47630
38	150182	FRANCISCAN HEALTH CARMEL	46032
42	180021	PINEVILLE COMMUNITY HOSPITAL	40977
47	190205	WOMEN'S AND CHILDREN'S HOSPITAL	70508
52	190268	LAFAYETTE GENERAL SURGICAL HOSPITAL	70503
53	190297	DOCTORS HOSPITAL AT DEER CREEK L L C	71446
65	260227	BLACK RIVER COMMUNITY MEDICAL CENTER	63901
85	330387	ROCKEFELLER UNIVERSITY HOSPITAL	10021
96	360141	NORTHSIDE MEDICAL CENTER	44501
101	370032	ALLIANCEHEALTH DEACONESS	73112
111	390061	LANCASTER REGIONAL MEDICAL CENTER	17604
115	390302	BARIX CLINICS OF PENNSYLVANIA	19047
125	440067	LAKEWAY REGIONAL HOSPITAL	37814
148	450766	UT SOUTHWESTERN UNIVERSITY HOSPITAL-ZALE LIPSHY	75390
149	450845	EL PASO SPECIALTY HOSPITAL	79902
150	450894	PINE CREEK MEDICAL CENTER LLP	75235
154	510039	OHIO VALLEY MEDICAL CENTER	26003
166	670087	BAYLOR SCOTT & WHITE EMERGENCY MEDICAL CENTER ...	78613

```
a.
b.
c.
```

Download Census zip code shapefile (10 pt)

1. a.
   b.
2.

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

1.

```python
import geopandas as gpd
```

2.
3.
4. a.
   b.
   c.
5. a.
   b.
   c.

Effects of closures on access in Texas (15 pts)

1.
2.
3.
4.

Reflecting on the exercise (10 pts)