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):
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: **___** **___**
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.

```python
import requests
import pandas as pd

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
    "CCN",                   # 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  # Set the limit to the maximum allowed by the API (5000
 ↪  records)

all_data = []

while True:
    params = {
        "column": columns_param,
        "size": limit,  # API allows size to be set to 5000
        "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.")
```

```python
        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  PGM_TRMNTN_CD  \
0                    1                   1.0              0
1                    1                   1.0              1
2                    1                   1.0              0
3                    1                   1.0              0
4                    1                   1.0              0
...                ...                   ...            ...
133526               1                   1.0              0
133527               1                   1.0              0
133528               1                   1.0              0
133529               1                   1.0              0
133530               1                   1.0              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 6 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.

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 “CCN”, # 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 # Set the limit to the maximum allowed by the API (5000 records) 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]  # Extract year from 'year_quarter'

combined_data.append(year_data)
```

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

combined_df.to_csv(“combined_data_2017_2019.csv”, index=False)

Display information about the combined dataset

print(f”Total records retrieved across all years: {combined_df.shape[0]}“)

Step 2: Plotting Number of Observations Per Year

Group by ‘Year’ and count the number of observations

observations_per_year = combined_df.groupby(“Year”).size()

Plot the number of observations per year

observations_per_year.plot(kind=“bar”, title=“Number of Observations Per Year”) plt.xlabel(“Year”) plt.ylabel(“Number of Observations”) plt.show()