Lab Assignment 8: Data Management Using `pandas`, Part 1

DS 6001: Practice and Application of Data Science

Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

In this lab, you will be working with the 2017 Workplace Health in America survey which was conducted by the Centers for Disease Control and Prevention. According to the survey's guidence document:

> The Workplace Health in America (WHA) Survey gathered information from a cross-sectional, nationally representative sample of US worksites. The sample was drawn from the Dun & Bradstreet (D&B) database of all private and public employers in the United States with at least 10 employees. Like previous national surveys, the worksite served as the sampling unit rather than the companies or firms to which the worksites belonged. Worksites were selected using a stratified simple random sample (SRS) design, where the primary strata were ten multi-state regions defined by the Centers for Disease Control and Prevention (CDC), plus an additional stratum containing all hospital worksites.

The data contain over 300 features that report the industry and type of company where the respondents are employed, what kind of health insurance and other health programs are offered, and other characteristics of the workplaces including whether employees are allowed to work from home and the gender and age makeup of the workforce. The data are full of interesting information, but in order to make use of the data a great deal of data manipulation is required first.

Problem 0

Import the following libraries:

In [1]:
```
!pip install sidetable
```

```
Requirement already satisfied: sidetable in c:\users\hodge\anaconda3\lib\site-packag
es (0.9.1)
Requirement already satisfied: pandas>=1.0 in c:\users\hodge\anaconda3\lib\site-pack
ages (from sidetable) (2.1.4)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\hodge\anaconda3\lib\site
-packages (from pandas>=1.0->sidetable) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hodge\anaconda3\li
b\site-packages (from pandas>=1.0->sidetable) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hodge\anaconda3\lib\site-pac
kages (from pandas>=1.0->sidetable) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\hodge\anaconda3\lib\site-p
ackages (from pandas>=1.0->sidetable) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\hodge\anaconda3\lib\site-package
s (from python-dateutil>=2.8.2->pandas>=1.0->sidetable) (1.16.0)
```

In [2]:
```
import numpy as np
import pandas as pd
import sidetable
import sqlite3
import warnings
warnings.filterwarnings('ignore')
```

Problem 1

The raw data are stored in an ASCII file on the 2017 Workplace Health in America survey homepage. Load the raw data directly into Python without downloading the data onto your harddrive and display a dataframe with only the 14th, 28th, and 102nd rows of the data. [1 point]

In [3]:
```
import json
import requests
import io
from io import StringIO
```

In [4]:
```
#load the raw data directly into Python without downloading the data onto your hard
url = "https://www.cdc.gov/workplacehealthpromotion/data-surveillance/docs/whpps_12
r = requests.get(url)
r
```

Out[4]:
```
<Response [200]>
```

In [5]:
```
data = r.text
```

In [6]:
```
data_io = io.StringIO(data)

df = pd.read_csv(data_io, delimiter = '~')

#display 14th, 28th, and 102nd rows of the data
df.iloc[np.array([13, 27, 101])]
```

Out[6]:

	OC1	OC3	HI1	HI2	HI3	HI4	HRA1	HRA1A	HRA1B	HRA1E	...	WL3_05	E1_09
13	3	1.0	2.0	3.0	2.0	1.0	1.0	3.0	3.0	1.0	...	NaN	NaN
27	1	3.0	1.0	3.0	1.0	1.0	1.0	2.0	4.0	2.0	...	NaN	NaN
101	2	1.0	1.0	3.0	2.0	1.0	1.0	2.0	4.0	2.0	...	NaN	NaN

3 rows × 301 columns

> I'm not positive if this was the preferred way of getting the data, but It was the best way I knew how.

Problem 2

The data contain 301 columns. Create a new variable in Python's memory to store a working version of the data. In the working version, delete all of the columns except for the following:

- `Industry` : 7 Industry Categories with NAICS codes
- `Size` : 8 Employee Size Categories
- `OC3` Is your organization for profit, non-profit, government?
- `HI1` In general, do you offer full, partial or no payment of premiums for personal health insurance for full-time employees?
- `HI2` Over the past 12 months, were full-time employees asked to pay a larger proportion, smaller proportion or the same proportion of personal health insurance premiums?
- `HI3` : Does your organization offer personal health insurance for your part-time employees?
- `CP1` : Are there health education programs, which focus on skill development and lifestyle behavior change along with information dissemination and awareness building?
- `WL6` : Allow employees to work from home?
- Every column that begins `WD` , expressing the percentage of employees that have certain characteristics at the firm

[1 point]

In [7]:
```
wd_columns = [col for col in df.columns if col.startswith('WD')]
```

```
wd_string_list = [str(column) for column in wd_columns]

string_list = ['Industry', 'Size', 'OC3', 'HI1', 'HI2', 'HI3', 'CP1', 'WL6'] + wd_s

health_df = df[string_list]

health_df.head()
```

Out[7]:

	Industry	Size	OC3	HI1	HI2	HI3	CP1	WL6	WD1_1	WD1_2	WD2	WD3	WD4	W
0	7.0	7.0	3.0	2.0	1.0	2.0	1.0	1.0	25.0	20.0	85.0	60.0	40.0	
1	7.0	6.0	3.0	2.0	3.0	1.0	1.0	1.0	997.0	997.0	90.0	90.0	997.0	9
2	7.0	8.0	3.0	1.0	3.0	1.0	1.0	1.0	35.0	4.0	997.0	997.0	40.0	
3	7.0	4.0	2.0	1.0	2.0	1.0	2.0	2.0	50.0	15.0	50.0	85.0	75.0	
4	7.0	4.0	3.0	1.0	3.0	1.0	1.0	1.0	50.0	40.0	60.0	60.0	40.0	

Problem 3

The codebook for the WHA data contain short descriptions of the meaning of each of the columns in the data. Use these descriptions to decide on better and more intuitive names for the columns in the working version of the data, and rename the columns accordingly. [1 point]

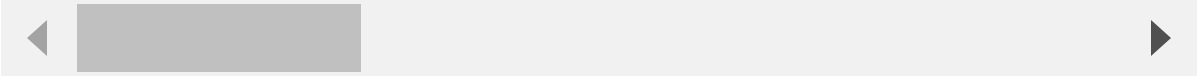
In [8]:

```
updated_columns = {'OC3' : 'Type',
            'HI1' : 'Insurance_Coverage',
            'HI2' : 'Payment_Size',
            'HI3' : 'Health_Insurance_Offered',
            'CP1' : 'Health_Education_Offered',
            'WL6' : 'At_Home_Workers',
            'WD1_1' : 'Workers_Under_30_Pct',
            'WD1_2' : 'Workers_60_and_Older_Pct',
            'WD2': 'Female_Workers_Pct',
            'WD3' : 'Hourly/Non-Exempt_Workers_Pct',
            'WD4' : 'Non-Daytime_Workers_Pct',
            'WD5' : 'Remote/Off-site_Workers_Pct',
            'WD6' : 'Bargaining/Unionized_Workers_Pct',
            'WD7' : 'Annual_Empoloyee_Turnover_Pct'}

health_df = health_df.rename(columns = updated_columns)

health_df
```

Out[8]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Offered	H
0	7.0	7.0	3.0	2.0	1.0	2.0	
1	7.0	6.0	3.0	2.0	3.0	1.0	
2	7.0	8.0	3.0	1.0	3.0	1.0	
3	7.0	4.0	2.0	1.0	2.0	1.0	
4	7.0	4.0	3.0	1.0	3.0	1.0	
...	...	...	...	...	...	...	
2838	6.0	5.0	4.0	1.0	3.0	1.0	
2839	6.0	5.0	4.0	2.0	3.0	1.0	
2840	6.0	8.0	4.0	2.0	3.0	1.0	
2841	6.0	8.0	4.0	2.0	3.0	1.0	
2842	6.0	8.0	4.0	2.0	3.0	1.0	

2843 rows × 16 columns



Problem 4

Using the codebook and this dictionary of NAICS industrial codes, place descriptive labels on the categories of the industry column in the working data. [1 point]

In [9]:
```
health_df['Industry'].unique()
```

Out[9]:
```
array([ 7.,  1.,  2.,  3.,  4.,  5.,  6., nan])
```

In [10]:
```
health_df['Industry']
```

Out[10]:
```
0       7.0
1       7.0
2       7.0
3       7.0
4       7.0
       ...
2838    6.0
2839    6.0
2840    6.0
2841    6.0
2842    6.0
Name: Industry, Length: 2843, dtype: float64
```

In [11]:
```
industry_mapping = {
    1: 'Agriculture, Forestry, Fishing and Hungting; Mining; Utilities; Constructio
    2: 'Wholesale Trade; Retail Trade; Transportation and Warehousing',
    3: 'Arts, Entertainment, and Recreation; Accomodation and Food Services; Other
    4: 'Information; Finance and Insurance; Real Estate Rental and Leasing; Profess
    5: 'Educational Services; Health Care and Social Assistance (excluding hospital
    6: 'Public Administration',
    7: 'General Medical and Surgical Hospitals; Psychiatric and Substance Abuse Hos
    'nan': np.nan
}
```

industry_mapping = { 1: {11: 'Agriculture, Forestry, Fishing and Hungting', 21: 'Mining', 22: 'Utilities', 23: 'Construction', 31-33: 'Manufacturing'}, 2: {42: 'Wholesale Trade', 44-45: 'Retail Trade', 48-49: 'Transportation and Warehousing'}, 3: {71: 'Arts, Entertainment, and Recreation', 72: 'Accomodation and Food Services', 81: 'Other Services (except Public Adminitration)'}, 4: {51: 'Information', 52: 'Finance and Insurance', 53: 'Real Estate Rental and Leasing', 54: 'Professional, Scientific, and Technical Services', 55: 'Management of Compoanies and Enterprises', 56: 'Adminitrative and Support and Waste...Services'}, 5: {61: 'Educational Services', 62: 'Health Care and Social Assistance (excluding hospitalworksites)'}, 6: {92: 'Public Administration'}, 7: {622110: 'General Medical and Surgical Hospitals', 622210: 'Psychiatric and Substance Abuse Hospitals', 622310: 'Specialty (except Psychiatric and Substance Abuse) Hospitals'}, 'nan': np.nan }

> I don't know any other way to solve this problem than treating this as a "Recoding Many Categorical Columns At Once" problem.

In [12]:
```
health_df['Industry'] = health_df['Industry'].map(industry_mapping)

health_df['Industry']
```

Out[12]:
```
0       General Medical and Surgical Hospitals; Psychi...
1       General Medical and Surgical Hospitals; Psychi...
2       General Medical and Surgical Hospitals; Psychi...
3       General Medical and Surgical Hospitals; Psychi...
4       General Medical and Surgical Hospitals; Psychi...
                              ...
2838                                Public Administration
2839                                Public Administration
2840                                Public Administration
2841                                Public Administration
2842                                Public Administration
Name: Industry, Length: 2843, dtype: object
```

> I had to call to .map() instead of .replace() because I was getting an attribute error that said

```
AttributeError: 'Series' object has no attribute '_replace_columnwise'
```

> I therefore had to continue with .map()

In [13]:
```
health_df.head()
```

Out[13]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Offered	Hea
0	General Medical and Surgical Hospitals; Psychi...	7.0	3.0	2.0	1.0	2.0	
1	General Medical and Surgical Hospitals; Psychi...	6.0	3.0	2.0	3.0	1.0	
2	General Medical and Surgical Hospitals; Psychi...	8.0	3.0	1.0	3.0	1.0	
3	General Medical and Surgical Hospitals; Psychi...	4.0	2.0	1.0	2.0	1.0	
4	General Medical and Surgical Hospitals; Psychi...	4.0	3.0	1.0	3.0	1.0	

Problem 5

Using the codebook, recode the "size" column to have three categories: "Small" for workplaces with fewer than 100 employees, "Medium" for workplaces with at least 100 but fewer than 500 employees, and "Large" for companies with at least 500 employees. [Note: Python dataframes have an attribute `.size` that reports the space the dataframe takes up in memory. Don't confuse this attribute with the column named "Size" in the raw data.] [1 point]

In [14]:
```
size_remapping = {1: 'Small',
                  2: 'Small',
                  3: 'Small',
                  4: 'Medium',
                  5: 'Medium',
                  6: 'Large',
                  7: 'Large',
                  8: 'Large'}

health_df['Size'] = health_df['Size'].map(size_remapping)

health_df['Size']
```

Out[14]:
```
0        Large
1        Large
2        Large
3       Medium
4       Medium
         ...
2838    Medium
2839    Medium
2840     Large
2841     Large
2842     Large
Name: Size, Length: 2843, dtype: object
```

In [15]:
```
health_df.head()
```

Out[15]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Offered
0	General Medical and Surgical Hospitals; Psychi...	Large	3.0	2.0	1.0	2.0
1	General Medical and Surgical Hospitals; Psychi...	Large	3.0	2.0	3.0	1.0
2	General Medical and Surgical Hospitals; Psychi...	Large	3.0	1.0	3.0	1.0
3	General Medical and Surgical Hospitals; Psychi...	Medium	2.0	1.0	2.0	1.0
4	General Medical and Surgical Hospitals; Psychi...	Medium	3.0	1.0	3.0	1.0

Problem 6

Use the codebook to write accurate and descriptive labels for each category for each categorical column in the working data. Then apply all of these labels to the data at once. Code "Legitimate Skip", "Don't know", "Refused", and "Blank" as missing values. [2 points]

In [16]:
```
updated_columns
```

Out[16]:
```
{'OC3': 'Type',
 'HI1': 'Insurance_Coverage',
 'HI2': 'Payment_Size',
 'HI3': 'Health_Insurance_Offered',
 'CP1': 'Health_Education_Offered',
 'WL6': 'At_Home_Workers',
 'WD1_1': 'Workers_Under_30_Pct',
 'WD1_2': 'Workers_60_and_Older_Pct',
 'WD2': 'Female_Workers_Pct',
 'WD3': 'Hourly/Non-Exempt_Workers_Pct',
 'WD4': 'Non-Daytime_Workers_Pct',
 'WD5': 'Remote/Off-site_Workers_Pct',
 'WD6': 'Bargaining/Unionized_Workers_Pct',
 'WD7': 'Annual_Empoloyee_Turnover_Pct'}
```

In [17]:
```
replace_map = {'Type':{1: 'For profit, public', 2: 'For profit, private', 3: 'Non-p
               'Insurance_Coverage': {1: 'Full insurance coverage offered', 2: 'Par
               'Payment_Size': {1: 'Larger', 2: 'Smaller', 3:'About the same'},
               'Health_Insurance_Offered': {1:'Yes', 2:'No'},
               'Health_Education_Offered': {1:'Yes', 2:'No'},
               'At_Home_Workers': {1:'Yes', 2:'No'}


}

#missing_values = {97:np.nan, 98:np.nan, 99:np.nan}
health_df = health_df.replace(replace_map)

health_df = health_df.replace([97, 98, 99], np.nan)

health_df.head()
```

Out[17]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Offered
0	General Medical and Surgical Hospitals; Psychi...	Large	Non-profit	Partial insurance coverage offered	Larger	No
1	General Medical and Surgical Hospitals; Psychi...	Large	Non-profit	Partial insurance coverage offered	About the same	Yes
2	General Medical and Surgical Hospitals; Psychi...	Large	Non-profit	Full insurance coverage offered	About the same	Yes
3	General Medical and Surgical Hospitals; Psychi...	Medium	For profit, private	Full insurance coverage offered	Smaller	Yes
4	General Medical and Surgical Hospitals; Psychi...	Medium	Non-profit	Full insurance coverage offered	About the same	Yes

Problem 7

The features that measure the percent of the workforce with a particular characteristic use the codes 997, 998, and 999 to represent "Don't know", "Refusal", and "Blank/Invalid" respectively. Replace these values with missing values for all of the percentage features at the same time. [1 point]

In [18]:
```
#missing_values = {97:np.nan, 98:np.nan, 99:np.nan}
health_df = health_df.replace([997, 998, 999], np.nan)

health_df.head()
```

Out[18]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Offered
0	General Medical and Surgical Hospitals; Psychi...	Large	Non-profit	Partial insurance coverage offered	Larger	No
1	General Medical and Surgical Hospitals; Psychi...	Large	Non-profit	Partial insurance coverage offered	About the same	Yes
2	General Medical and Surgical Hospitals; Psychi...	Large	Non-profit	Full insurance coverage offered	About the same	Yes
3	General Medical and Surgical Hospitals; Psychi...	Medium	For profit, private	Full insurance coverage offered	Smaller	Yes
4	General Medical and Surgical Hospitals; Psychi...	Medium	Non-profit	Full insurance coverage offered	About the same	Yes

Problem 8

Sort the working data by industry in ascending alphabetical order. Within industry categories, sort the rows by size in ascending alphabetical order. Within groups with the same industry and size, sort by percent of the workforce that is under 30 in descending numeric order. [1 point]

In [19]:
```
health_df = health_df.sort_values(by = ['Industry', 'Size', 'Workers_Under_30_Pct']
health_df
```

Out[19]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Of
1732	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	About the same	
1476	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	About the same	
1477	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	Smaller	
704	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Full insurance coverage offered	About the same	
1241	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Full insurance coverage offered	About the same	
...	...	...	...	...	...	
2604	Wholesale Trade; Retail Trade; Transportation ...	Small	Non-profit	Full insurance coverage offered	About the same	
2626	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, private	Partial insurance coverage offered	Larger	
2629	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, public	Full insurance coverage offered	Larger	
2631	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, private	Partial insurance coverage offered	Larger	
1662	NaN	NaN	NaN	NaN	NaN	

2843 rows × 16 columns

Problem 9

There is one row in the working data that has a `NaN` value for industry. Delete this row. Use a logical expression, and not the row number. [1 point]

In [20]:
```
health_df = health_df.dropna(subset = ['Industry'])
health_df
```

Out[20]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Of
1732	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	About the same	
1476	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	About the same	
1477	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	Smaller	
704	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Full insurance coverage offered	About the same	
1241	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Full insurance coverage offered	About the same	
...	...	...	...	...	...	
2595	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, private	Full insurance coverage offered	About the same	
2604	Wholesale Trade; Retail Trade; Transportation ...	Small	Non-profit	Full insurance coverage offered	About the same	
2626	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, private	Partial insurance coverage offered	Larger	
2629	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, public	Full insurance coverage offered	Larger	
2631	Wholesale Trade; Retail Trade;	Small	For profit, private	Partial insurance coverage offered	Larger	

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Of
	Transportation					
	...					

2842 rows × 16 columns

Problem 10

Create a new feature named `gender_balance` that has three categories: "Mostly men" for workplaces with between 0% and 35% female employees, "Balanced" for workplaces with more than 35% and at most 65% female employees, and "Mostly women" for workplaces with more than 65% female employees. [1 point]

In [21]:
```
health_df['gender_balance'] = pd.cut(health_df['Female_Workers_Pct'], bins = [-0.1,
health_df[['Female_Workers_Pct', 'gender_balance']]
```

Out[21]:

	Female_Workers_Pct	gender_balance
1732	50.0	Balanced
1476	30.0	Mostly men
1477	20.0	Mostly men
704	17.0	Mostly men
1241	50.0	Balanced
...	...	...
2595	2.0	Mostly men
2604	NaN	NaN
2626	NaN	NaN
2629	15.0	Mostly men
2631	NaN	NaN

2842 rows × 2 columns

Problem 11

Change the data type of all categorical features in the working data from "object" to "category". [1 point]

In [22]:
```
updated_columns
```

Out[22]:
```
{'OC3': 'Type',
 'HI1': 'Insurance_Coverage',
 'HI2': 'Payment_Size',
 'HI3': 'Health_Insurance_Offered',
 'CP1': 'Health_Education_Offered',
 'WL6': 'At_Home_Workers',
 'WD1_1': 'Workers_Under_30_Pct',
 'WD1_2': 'Workers_60_and_Older_Pct',
 'WD2': 'Female_Workers_Pct',
 'WD3': 'Hourly/Non-Exempt_Workers_Pct',
 'WD4': 'Non-Daytime_Workers_Pct',
 'WD5': 'Remote/Off-site_Workers_Pct',
 'WD6': 'Bargaining/Unionized_Workers_Pct',
 'WD7': 'Annual_Empoloyee_Turnover_Pct'}
```

In [23]:
```
health_df.dtypes
```

Out[23]:
```
Industry                          object
Size                              object
Type                              object
Insurance_Coverage                object
Payment_Size                      object
Health_Insurance_Offered          object
Health_Education_Offered          object
At_Home_Workers                   object
Workers_Under_30_Pct             float64
Workers_60_and_Older_Pct         float64
Female_Workers_Pct               float64
Hourly/Non-Exempt_Workers_Pct    float64
Non-Daytime_Workers_Pct          float64
Remote/Off-site_Workers_Pct      float64
Bargaining/Unionized_Workers_Pct float64
Annual_Empoloyee_Turnover_Pct    float64
gender_balance                  category
dtype: object
```

In [24]:
```
catcolumns = ['Type', 'Insurance_Coverage', 'Payment_Size', 'Health_Insurance_Offer

health_df[catcolumns] = health_df[catcolumns].astype('category')

health_df.dtypes
```

Out[24]:
```
Industry                            object
Size                                object
Type                              category
Insurance_Coverage                category
Payment_Size                      category
Health_Insurance_Offered          category
Health_Education_Offered          category
At_Home_Workers                   category
Workers_Under_30_Pct               float64
Workers_60_and_Older_Pct           float64
Female_Workers_Pct                 float64
Hourly/Non-Exempt_Workers_Pct      float64
Non-Daytime_Workers_Pct            float64
Remote/Off-site_Workers_Pct        float64
Bargaining/Unionized_Workers_Pct   float64
Annual_Empoloyee_Turnover_Pct      float64
gender_balance                    category
dtype: object
```

Problem 12

Filter the data to only those rows that represent small workplaces that allow employees to work from home. Then report how many of these workplaces offer full insurance, partial insurance, and no insurance. Use a function that reports the percent, cumulative count, and cumulative percent in addition to the counts. [1 point]

In [25]:
```
query_df = health_df.query("Size == 'Small' & At_Home_Workers == 'Yes'")
query_df
```

Out[25]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Of
900	Agriculture, Forestry, Fishing and Hungting; M...	Small	NaN	No insurance coverage offered	96.0	
2051	Agriculture, Forestry, Fishing and Hungting; M...	Small	For profit, private	Full insurance coverage offered	About the same	
542	Agriculture, Forestry, Fishing and Hungting; M...	Small	For profit, private	Partial insurance coverage offered	About the same	
1180	Agriculture, Forestry, Fishing and Hungting; M...	Small	For profit, private	Full insurance coverage offered	About the same	
2577	Agriculture, Forestry, Fishing and Hungting; M...	Small	For profit, private	Partial insurance coverage offered	About the same	
...	...	...	...	...	...	
1768	Wholesale Trade; Retail Trade; Transportation ...	Small	Other	Partial insurance coverage offered	NaN	
2109	Wholesale Trade; Retail Trade; Transportation ...	Small	NaN	Partial insurance coverage offered	Smaller	
2112	Wholesale Trade; Retail Trade; Transportation ...	Small	Other	Full insurance coverage offered	About the same	
2384	Wholesale Trade; Retail Trade; Transportation ...	Small	NaN	NaN	96.0	
2629	Wholesale Trade; Retail Trade;	Small	For profit, public	Full insurance coverage offered	Larger	

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Of
	Transportation					
	...					

709 rows × 17 columns

In [26]:
```
query_df.stb.freq(['Insurance_Coverage'])
```

Out[26]:

	Insurance_Coverage	count	percent	cumulative_count	cumulative_percent
0	Full insurance coverage offered	324	46.285714	324	46.285714
1	Partial insurance coverage offered	310	44.285714	634	90.571429
2	No insurance coverage offered	66	9.428571	700	100.000000

Problem 13

Anything that can be done in SQL can be done with `pandas`. The next several questions ask you to write `pandas` code to match a given SQL query. But to check that the SQL query and `pandas` code yield the same result, create a new database wsing the `sqlite3` package and input the cleaned WHA data as a table in this database. (See module 6 for a discussion of SQlite in Python.) [1 point]

In [27]:
```
import os
```

In [28]:
```
os.chdir("/Users/hodge/Desktop/UVA_Coding_Folder/DS6001")
```

In [29]:
```
health_db = sqlite3.connect("health.db")
```

In [30]:
```
health_df.to_sql('health', health_db, index=False, chunksize=1000, if_exists='repla
```

Out[30]: 2842

In [31]:
```
health_cursor = health_db.cursor()
```

In [32]:
```
health_cursor.execute("SELECT * FROM health")

my_df = health_cursor.fetchall()

colnames = [x[0] for x in health_cursor.description]

pd.DataFrame(my_df, columns = colnames)
```

Out[32]:

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Of
0	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	About the same	
1	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	About the same	
2	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Partial insurance coverage offered	Smaller	
3	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Full insurance coverage offered	About the same	
4	Agriculture, Forestry, Fishing and Hungting; M...	Large	For profit, private	Full insurance coverage offered	About the same	
...	...	...	...	...	...	
2837	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, private	Full insurance coverage offered	About the same	
2838	Wholesale Trade; Retail Trade; Transportation ...	Small	Non-profit	Full insurance coverage offered	About the same	
2839	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, private	Partial insurance coverage offered	Larger	
2840	Wholesale Trade; Retail Trade; Transportation ...	Small	For profit, public	Full insurance coverage offered	Larger	
2841	Wholesale Trade; Retail Trade;	Small	For profit, private	Partial insurance coverage offered	Larger	

	Industry	Size	Type	Insurance_Coverage	Payment_Size	Health_Insurance_Of
	Transportation					
	...					

2842 rows × 17 columns

Problem 14

Write `pandas` code that replicates the output of the following SQL code:

```
SELECT size, type, premiums AS insurance, percent_female FROM whpps
WHERE industry = 'Hospitals' AND premium_change='Smaller'
ORDER BY percent_female DESC;
```

For each of these queries, your feature names might be different from the ones listed in the query, depending on the names you chose in problem 3. [2 points]

In [33]:
```
health_cursor.execute("""
SELECT
    Size,
    Type,
    Insurance_Coverage AS insurance,
    Female_Workers_Pct
FROM
    health
WHERE
    Industry = 'General Medical and Surgical Hospitals; Psychiatric and Substance A
    AND payment_size = 'Smaller'
ORDER BY
    Female_Workers_Pct DESC;
""")

my_df = health_cursor.fetchall()

colnames = [x[0] for x in health_cursor.description]

pd.DataFrame(my_df, columns = colnames)
```

Out[33]:

	Size	Type	insurance	Female_Workers_Pct
0	Medium	Non-profit	Full insurance coverage offered	89.0
1	Large	Non-profit	Partial insurance coverage offered	80.0
2	Large	Non-profit	Partial insurance coverage offered	80.0
3	Small	Non-profit	Full insurance coverage offered	75.0
4	Medium	Non-profit	Partial insurance coverage offered	65.0
5	Medium	For profit, private	Full insurance coverage offered	50.0
6	Large	Non-profit	Partial insurance coverage offered	NaN
7	Medium	Non-profit	Full insurance coverage offered	NaN
8	Medium	None	Partial insurance coverage offered	NaN
9	Medium	Non-profit	Partial insurance coverage offered	NaN
10	Medium	Non-profit	Full insurance coverage offered	NaN

> My Industry in this case does not have 'Hospital'. This stems from my confusion with Problem 4 and not knowing exactly how to label the categories of the industry column. Whatever the case may be, I hope that I was able to query the table correclty.

Problem 15

Write `pandas` code that replicates the output of the following SQL code:

```
SELECT industry,
    AVG(percent_female) as percent_female,
    AVG(percent_under30) as percent_under30,
    AVG(percent_over60) as percent_over60
FROM whpps
GROUP BY industry
ORDER BY percent_female DESC;
```

[2 points]

In [34]:
```
health_cursor.execute("""
SELECT
    Industry,
    AVG(Female_Workers_Pct) as percent_female,
    AVG(Workers_Under_30_Pct) as percent_under30,
    AVG(Workers_60_and_Older_Pct) as percent_over60
FROM
    health
```

```
GROUP BY
    Industry
ORDER BY
    percent_female DESC;
""")

my_df = health_cursor.fetchall()

colnames = [x[0] for x in health_cursor.description]

pd.DataFrame(my_df, columns = colnames)
```

Out[34]:

	Industry	percent_female	percent_under30	percent_over60
0	Educational Services; Health Care and Social A...	78.354839	25.533333	11.349570
1	General Medical and Surgical Hospitals; Psychi...	75.944751	27.213793	16.489655
2	Arts, Entertainment, and Recreation; Accomodat...	53.236422	38.172638	11.270096
3	Information; Finance and Insurance; Real Estat...	49.365782	23.596970	12.465465
4	Public Administration	39.056738	21.015625	15.015385
5	Wholesale Trade; Retail Trade; Transportation ...	32.126016	29.108696	12.584034
6	Agriculture, Forestry, Fishing and Hungting; M...	20.328605	22.257143	10.690355

Problem 16

Write `pandas` code that replicates the output of the following SQL code:

```
SELECT gender_balance, premiums, COUNT(*)
FROM whpps
GROUP BY gender_balance, premiums
HAVING gender_balance is NOT NULL and premiums is NOT NULL;
```

[2 points]

In [35]:

```
health_cursor.execute("""
SELECT
    gender_balance,
    Insurance_Coverage,
    COUNT(*)
FROM
    health
GROUP BY
    gender_balance,
```

```
    Insurance_Coverage
HAVING
    gender_balance is NOT NULL and Insurance_Coverage is NOT NULL;
""")

my_df = health_cursor.fetchall()

colnames = [x[0] for x in health_cursor.description]

pd.DataFrame(my_df, columns = colnames)
```

Out[35]:

	gender_balance	Insurance_Coverage	COUNT(*)
0	Balanced	Full insurance coverage offered	226
1	Balanced	No insurance coverage offered	77
2	Balanced	Partial insurance coverage offered	271
3	Mostly men	Full insurance coverage offered	301
4	Mostly men	No insurance coverage offered	91
5	Mostly men	Partial insurance coverage offered	332
6	Mostly women	Full insurance coverage offered	251
7	Mostly women	No insurance coverage offered	95
8	Mostly women	Partial insurance coverage offered	298

Commit and Close Database

In [36]:

```
health_db.commit()
health_db.close()
```