In [1]:

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docke
r-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list al
l files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets p
reserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved out
side of the current session
```

```
/kaggle/input/data-analysis/cdbrfss1999.csv
```

Data Analysis in Astronomy and Physics

Exercise Set 1

Xiongxiao Wang, Sakshi Pahujani, Mahak Sadhwani

1.Sampling

In [2]:

```
import os
os.getcwd() # get current working dictory
```

Out[2]:

```
'/kaggle/working'
```

In [3]:

```
os.chdir('/kaggle/') #change dirctory to kaggle
os.getcwd()
```

Out[3]:

```
'/kaggle'
```

In [4]:

```
os.listdir('/kaggle') # show the list of dictory 'kaggle'
```

Out[4]:

```
['src', 'lib', 'input', 'working']
```

In [5]:

```
os.listdir('/kaggle/input/data-analysis')
```

Out[5]:

```
['cdbrfss1999.csv']
```

Large data

- a. Take a sample of 30000 from this dataset and export it to an ASCII file. Make sure that your method allows to draw more than one sample from the population.
- b. Discuss your method to do so. Is your sampling a “good sample” in the sense that it is representative for the larger “population”?

solution:

- a.

1. Get the total number of data in file by counting the number of rows and minus 1
2. Set the desired sample size(30000)
3. select (n-s) numbers randomly from range(1,n+1)
4. skip the corresponding rows when reading the csv

- b. The method here is simple random sampling,each sample chosen by equal probability, so it's can be representative for larger "population"

In [6]:

```
import pandas as pd
import random

filename = 'input/data-analysis/cdbrfss1999.csv'
n = sum(1 for line in open(filename)) - 1 #number of rows in file (excludes heade
r)
s = 30000 #desired sample size
skip = sorted(random.sample(range(1,n+1),n-s)) #select (n-s) numbers randomly fro
m range(1,n+1). In next step, skip the corresponding row when reading the csv
cdbr = pd.read_csv(filename,skiprows = skip)
cdbr #display the 30000 samples by dataframe
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshel
l.py:3457: DtypeWarning: Columns (8) have mixed types.Specify dtype
option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[6]:

	STATE	GEOSTR	DENSTR	PSU	RECORD	IMONTH	IDAY	IYEAR	INTVID
0	1	1	1	10013	1	1	12	1999	33
1	1	1	1	10105	1	1	11	1999	40
2	1	1	1	10120	1	1	23	1999	33
3	1	1	1	10126	1	1	15	1999	6
4	1	1	1	10177	1	1	16	1999	0
...	...	...	...	...	...	...	...	...	...
29995	72	1	1	121031	1	12	19	1999	20
29996	72	1	1	121034	1	12	3	1999	6
29997	72	1	1	121048	1	12	5	1999	6
29998	72	1	1	121094	1	12	3	1999	6
29999	72	1	1	121095	1	12	1	1999	6

30000 rows × 282 columns

> Large columns

- a. Locate the columns corresponding to the variables genhlth, exerany, htf, hti, smoke100, weight, wtdesire, age, and sex.
- b. Reduce your sample to include only these variables and export it to an ASCII file.

solution: export this Dataframe into an ASCII file that include only these variables

In [7]:

```
Large_Columns_b = cdbr[['GENHLTH','EXERANY','HTF','HTI','SMOKE100','WEIGHT','WTD
ESIRE','AGE','SEX']]
filename_destination= 'working/Large_Columns_b.csv'
Large_Columns_b.to_csv(filename_destination)
Large_Columns_b  # display the Dataframe that question "Large Column b" required
```

Out[7]:

	GENHLTH	EXERANY	HTF	HTI	SMOKE100	WEIGHT	WTDESIRE	AGE	SEX
0	3	NaN	5	8	2	160	NaN	59	2
1	4	NaN	5	10	1	180	NaN	60	1
2	1	NaN	5	3	2	147	NaN	34	2
3	2	NaN	5	11	1	180	NaN	9	1
4	2	NaN	5	4	2	165	NaN	59	2
...	...	...	...	...	...	...	...	...	...
29995	4	NaN	5	8	1	190	NaN	58	1
29996	1	NaN	5	5	2	140	NaN	52	2
29997	2	NaN	5	2	2	145	NaN	42	2
29998	4	NaN	5	1	2	142	NaN	64	2
29999	3	NaN	5	0	2	127	NaN	67	2

30000 rows × 9 columns

c. How many cases and how many variables are there in your sample?

(c) 30,000 cases; 9 variables

> d. What type of variable is genhlth?

(d) categorical, ordinal

> e. What type of variable is weight?

(b) numerical, discrete

> f. What type of variable is smoke100?

(c) categorical (not ordinal)

> One Bar chart

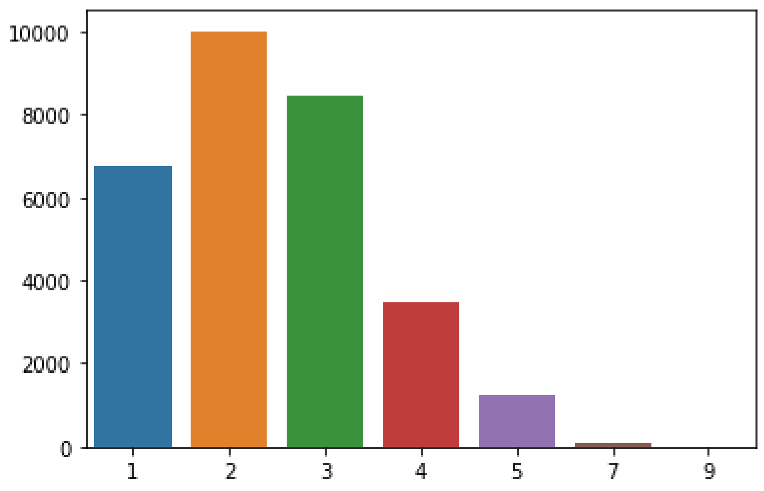
- > Take all genhlth entries from your sample and draw a bar chart to visualize how the cases are distributed across the possible categories.

In [8]:

```
# x axis represents General Health. 1,2,3,4,5,7,9 means Excellent, Very good, Goo
d, Fair, Poor, Don't know, Refused respectively
# y axis represents freuquency
import numpy as np
import seaborn as sns
genhlth = Large_Columns_b['GENHLTH']
genhlth_data_count = np.unique(genhlth,return_counts = True)
sns.barplot(x = genhlth_data_count[0],y = genhlth_data_count[1])
```

Out[8]:

```
<AxesSubplot:>
```



Two Bar Charts

- Combine the smoke100 with the genhlth entries from your sample and draw two bar charts, one showing the health of the smokers and a second one showing the health of the non-smokers.

In [9]:

```
genhlth_count_ifsmoke = Large_Columns_b.groupby(['SMOKE100','GENHLTH']).SMOKE100
.agg('count')
genhlth_count_ifsmoke.describe()
#genhlth_count_ifsmoke['SMOKE100'].describe()
#sns.barplot(x = 'SMOKE100', y = 'count', hue = 'GENHLTH',data = genhlth_count_ifs
moke)
```

Out[9]:

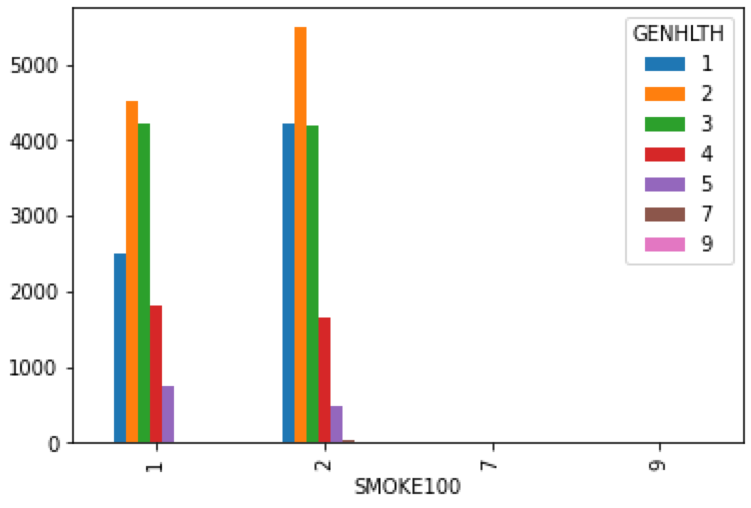
```
count      25.000000
mean     1200.000000
std      1835.563288
min         1.000000
25%         4.000000
50%        22.000000
75%      1815.000000
max      5481.000000
Name: SMOKE100, dtype: float64
```

In [10]:

```
#  x axis represents if smoke100, if so ,x=1; otherwise x=2. x=7,9 means 'not sur
e'/'refused' respectively. y axis means frequency
genhlth_count_ifsmoke.unstack().plot.bar()
```

Out[10]:

```
<AxesSubplot:xlabel='SMOKE100'>
```



> BMI

- Next let’s consider a new variable bmi that doesn’t show up directly in this data set: Body Mass Index (BMI).
- Compute the bmi for each case in your sample and add it to the sample (e.g. as additional column).
- Visualize the distribution of the BMI in your sample.

In [11]:

```
Large_Columns_b['BMI'] = Large_Columns_b['WEIGHT']/((Large_Columns_b['HTF']*12+L
arge_Columns_b['HTI'])*(Large_Columns_b['HTF']*12+Large_Columns_b['HTI']))*703
Large_Columns_b.drop(Large_Columns_b[Large_Columns_b.HTI > 11].index, inplace =
True) #drop the data whose height(iches) is 77 and 99(means Don't know and refused
respectively)
Large_Columns_b.drop(Large_Columns_b[Large_Columns_b.WEIGHT > 776].index, inplac
e = True)#drop the data whose weight is 777 and 999(means Don't know and refused r
espectively)
Large_Columns_b.drop(Large_Columns_b[Large_Columns_b.HTF > 7].index, inplace = T
rue)#drop the data whose height (feets) is 9(means refused)
BMI_data = Large_Columns_b['BMI']
BMI_data.drop(BMI_data[BMI_data == np.inf].index, inplace=True)# drop the data w
hose height is 0
BMI_data
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1: Set
tingWithCopyWarning:
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/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.7/site-packages/pandas/core/frame.py:4913: S
ettingWithCopyWarning:
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/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy
  errors=errors,
/opt/conda/lib/python3.7/site-packages/pandas/core/generic.py:4153:
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/pan
das-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-
copy
  self._update_inplace(obj)
```

Out[11]:

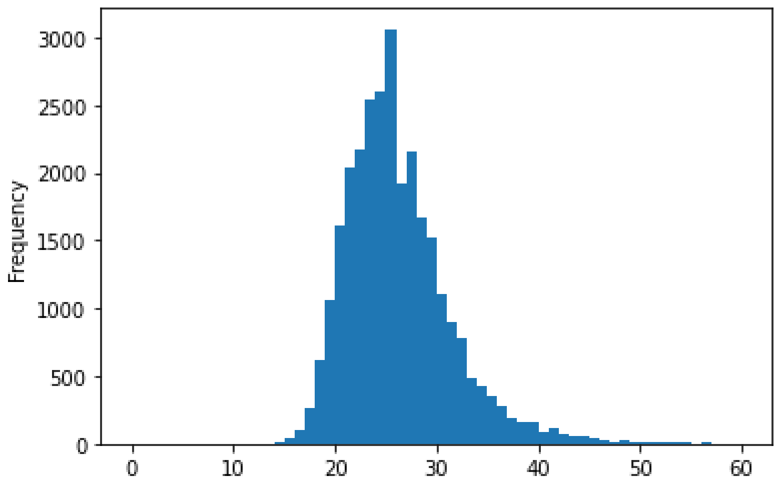
```
0        24.325260
1        25.824490
2        26.037037
3        25.102162
4        28.319092
           ...
29995    28.886246
29996    23.294675
29997    26.517950
29998    26.827734
29999    24.800278
Name: BMI, Length: 28830, dtype: float64
```

In [12]:

```
# It's a histogram, x axis represents BMI, y axis represents Frequency
BMI_data.plot.hist('BMI', bins =60 , range=(0,60))
```

Out[12]:

```
<AxesSubplot:ylabel='Frequency'>
```


Frequency