WEEK 3 - Assignment 2.3

In [1]:

```
# DSC530-T302
# Week 3 - 2.3
# 1.2 Programming Assignment
# Author: Aarti Ramani
# Created Date: 12/16/2022
# Purpose: Program to match the pregnancy numbers in NSFG pregnancy data and respon
# ******************************************************************************
# Change#:1 (Week 3)
# Change(s) Made: Version 1.0
# Date of Change: 12/16/2022
# Author: Aarti Ramani
# Change Approved by: N/A
# Date Moved to Production: N/A
# ******************************************************************************
import thinkstats2
import numpy as np
from collections import defaultdict
from os.path import basename, exists
import thinkplot


def MakePregMap(df):
    """Make a map from caseid to list of preg indices.

    df: DataFrame

    returns: dict that maps from caseid to list of indices into `preg`
    """
    d = defaultdict(list)
    for index, caseid in df.caseid.iteritems():
        d[caseid].append(index)
    return d


def CleanFemResp(df):
    """Recodes variables from the respondent frame.

    df: DataFrame
    """
    pass


def CleanFemPreg(df):
    """Recodes variables from the pregnancy frame.

    df: DataFrame
    """
    # mother's age is encoded in centiyears; convert to years
    df.agepreg /= 100.0
```

```
    # birthwgt_lb contains at least one bogus value (51 lbs)
    # replace with NaN
    df.loc[df.birthwgt_lb > 20, 'birthwgt_lb'] = np.nan

    # replace 'not ascertained', 'refused', 'don't know' with NaN
    na_vals = [97, 98, 99]
    df.birthwgt_lb.replace(na_vals, np.nan, inplace=True)
    df.birthwgt_oz.replace(na_vals, np.nan, inplace=True)
    df.hpagelb.replace(na_vals, np.nan, inplace=True)

    df.babysex.replace([7, 9], np.nan, inplace=True)
    df.nbrnaliv.replace([9], np.nan, inplace=True)

    # birthweight is stored in two columns, lbs and oz.
    # convert to a single column in lb
    # NOTE: creating a new column requires dictionary syntax,
    # not attribute assignment (like df.totalwgt_lb)
    df['totalwgt_lb'] = df.birthwgt_lb + df.birthwgt_oz / 16.0

    # due to a bug in ReadStataDct, the last variable gets clipped;
    # so for now set it to NaN
    df.cmintvw = np.nan


def ReadFemPreg(dct_file='2002FemPreg.dct',
                dat_file='2002FemPreg.dat.gz'):
    """Reads the NSFG pregnancy data.

    dct_file: string file name
    dat_file: string file name

    returns: DataFrame
    """
    dct = thinkstats2.ReadStataDct(dct_file)
    df = dct.ReadFixedWidth(dat_file, compression='gzip')
    CleanFemPreg(df)
    return df


def ReadFemResp(dct_file='2002FemResp.dct',
                dat_file='2002FemResp.dat.gz',
                nrows=None):
    """Reads the NSFG respondent data.

    dct_file: string file name
    dat_file: string file name

    returns: DataFrame
    """
    dct = thinkstats2.ReadStataDct(dct_file)
    df = dct.ReadFixedWidth(dat_file, compression='gzip', nrows=nrows)
    CleanFemResp(df)
    return df


def Mode(hist, hist_key):
```

```
    # print(hist_key)
    maxfreq = hist[hist_key].value_counts(ascending=False).head(1)
    # print(maxfreq.values)
    # print(maxfreq.index.values)
    return maxfreq


def AllModes(hist, hist_key):
    allfreq = hist[hist_key].value_counts(ascending=False)
    # print(allfreq)
    return allfreq


def main():
    try:
        preg = ReadFemPreg()
        resp = ReadFemResp()
        live = preg[preg.outcome == 1]
        try:
            searchcolumn = str(input('Please enter the column for which '
                                     'you would like to get the frequency : '))
        except RuntimeError as err:
            print('Invalid user input : ', err)
        else:
            columnname = [col for col in live.columns if searchcolumn == col]
            if len(columnname) > 0:
                try:
                    maxfreq = Mode(live, columnname[0])
                except RuntimeError as err:
                    print('Error in function <Mode>:', err)
                else:
                    if maxfreq.count() > 0:
                        print('Most frequent value in', columnname[0], ' is : '
                              , maxfreq.index.values,
                              ' with a count of : ', maxfreq.values)
                    else:
                        print('Function did not return any frequency for selected c
                    try:
                        allfreq = AllModes(live, columnname[0])
                        if allfreq.count() > 0:
                            # print('Most frequent value in', columnname[0], ' is :
                            #allfreq.index.values, ' with a count of : ', allfreq.v
                            print("Most frequent value in column :", '\033[1m' + co
                                  + '\033[0m', '\n')
                            print('{:15}''{:1}'.format("Value", "Count"))
                            print('{:15}''{:1}'.format("---------", "-----------"))
                            for i in allfreq.index:
                        # Add every key in the sorted list to the sorted-dictionary
                                print("{:15} {:1}".format(str(i), str(allfreq[i])))
                        else:
                            print('Function did not return any frequency for select
                    except RuntimeError as err:
                        print('Error in function <AllModes>:', err)
            else:
                print('Column does not exist.')
    except RuntimeError as err:
```

```
        print('We ran into an issue. ', err)


if __name__ == '__main__':
    main()
```

```
Please enter the column for which you would like to get the frequency : pregnum
Most frequent value in pregnum  is :  [3]  with a count of :  [2401]
Most frequent value in column : pregnum

Value          Count
---------      -----------
3               2401
2               2189
4               1612
5               950
1               851
6               512
7               296
8               169
9               92
10              37
11              19
12              9
14              7
19              4
```