# Table of Contents



# Import libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly_express as px
import statsmodels.formula.api as smf
%matplotlib inline
```

# Load first two data files

In [2]:

```
ben_summary_df = pd.read_excel(r"C:\Users\Corey
Moser\Documents\Programming\Python\My_Projects\MorganStanley_Case_Study\BeneficiarySummary_2008.xls
, sheet_name="Sheet2" )
```

### Explantory Text

- This file loaded abvove (referred throughout to throughout this Notebook as `ben_summary_df` contains Medicare beneficiary summary data from 2008.
- In addition to basic demographic data (de-identified of course), the file also contains flags indicating whether the beneficiary has been diagnosed with one of 11 chronic conditions (e.g. Alzheimer, heart failure, COPD, diabetes etc...).
- Finally the file includes a variety of payment info including inpatient and outpatient reimbursement amounts as well as carrier, beneficiary and primary payer responsibilities.

In [3]:

```
#Show the first 5 rows of this file
ben_summary_df.head()
```

Out[3]:

| | DESYNPUF_ID | Number of Claims | SuperUtilizer | BENE_BIRTH_DT | BENE_DEATH_DT | BENE_SEX_IDENT_CD | BENE_RACE_CD | BENE_ES |
|---|---|---|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 1 | N | 19230501 | NaN | 1 | 1 | |
| 1 | 00016F745862898F | 0 | N | 19430101 | NaN | 1 | 1 | |
| 2 | 0001FDD721E223DC | 0 | N | 19360901 | NaN | 2 | 1 | |
| 3 | 00021CA6FF03E670 | 0 | N | 19410601 | NaN | 1 | 5 | |
| 4 | 00024B3D2352D2D0 | 1 | N | 19360801 | NaN | 1 | 1 | |

5 rows × 34 columns

## Explantory Text

- This second file contains ( `claims_df` ) 76 variables. Each record pertains to a synthetic outpatient claim and includes everything from claim ID and physician present to diagnosis and revenue codes.
- Crucially, this file contains a common column ( `DESYNPUF_ID` ) which allows us to tally the number of outpatient claims attributed to each member in our member summary file. We only need this file to tally the number of claims by each member of in `ben_summary_df`

In [4]:

```
claims_df = pd.read_excel(r"C:\Users\Corey Moser\Documents\Programming\Python\My_Projects\MorganStanley_Case_Study\Claims_2008.xlsx")
```

In [5]:

```
#Display the first 5 rows of the claims data file
claims_df.head(n=5)
```

Out[5]:

| | DESYNPUF_ID | CLM_ID | SEGMENT | CLM_FROM_DT | CLM_THRU_DT | PRVDR_NUM | Year | CLM_PMT_AMT | NCH_PRMF |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 542192281063886 | 1 | 20080904 | 20080904 | 2600RA | 2008 | 50 | |
| 1 | 00024B3D2352D2D0 | 542242281386963 | 1 | 20080712 | 20080712 | 5200TV | 2008 | 30 | |
| 2 | 0002F28CE057345B | 542162280904893 | 1 | 20080423 | 20080423 | 3902NU | 2008 | 10 | |
| 3 | 0002F28CE057345B | 542192281407888 | 1 | 20080724 | 20080724 | 3902NU | 2008 | 60 | |
| 4 | 0002F28CE057345B | 542342281460715 | 1 | 20080727 | 20080727 | 3900RQ | 2008 | 200 | |

5 rows × 77 columns

## Explantory Text

Use the member ID column `DESYNPUF_ID`, which exists in both data sets, to get a count of how many times each member submitted a claim.

In [6]:

```
z = claims_df['DESYNPUF_ID'].value_counts() #Get a count of each value in the claims_df data set
z1 = z.to_dict() #convert this to a dictonary format
```

## Explantory Text

- We can confirm that this worked by running the cell below.
- The member with ID *776E90FCC80C722F* submitted 33 claims in 2008. This jives with what we see in Excel

In [7]:

```
i = 10
for k, v in z1.items():
    if i > 0:
        print(k,v)
        i -=1
```

```
776E90FCC80C722F 33
DA2C31A13EEA6994 33
DB998B20F3833F82 32
D16705EB362C7513 32
66B11951279DB594 32
7819D8FA21478516 31
F1B31EAE9B8F41C9 31
AD5D4E087BF57C05 31
6688888FAE4A668D 31
634C57A49CC11CD1 31
```

## Explantory Text

- **The following code "maps" the key value pairs to our benefits summary dataframe.**
- **Applying the number of claims to the corresponding member ID and creating a new column, `NumClaims` with that value**

In [8]:

```
ben_summary_df['NumClaims'] =  ben_summary_df['DESYNPUF_ID'].map(z1)
```

In [9]:

```
ben_summary_df[['DESYNPUF_ID','NumClaims']].head(n=5)
```

Out[9]:

| | DESYNPUF_ID | NumClaims |
|---|---|---|
| 0 | 00013D2EFD8E45D1 | 1.0 |
| 1 | 00016F745862898F | NaN |
| 2 | 0001FDD721E223DC | NaN |
| 3 | 00021CA6FF03E670 | NaN |
| 4 | 00024B3D2352D2D0 | 1.0 |

In [10]:

```
ben_summary_df =  ben_summary_df.replace({
    'NumClaims': np.nan}, 0)
```

In [11]:

```
ben_summary_df[['DESYNPUF_ID','NumClaims']].head(n=5)
```

Out[11]:

| | DESYNPUF_ID | NumClaims |
|---|---|---|
| 0 | 00013D2EFD8E45D1 | 1.0 |
| 1 | 00016F745862898F | 0.0 |
| 2 | 0001FDD721E223DC | 0.0 |
| 3 | 00021CA6FF03E670 | 0.0 |
| 4 | 00024B3D2352D2D0 | 1.0 |

# Analysis 1

### Run an OLS regression -- model the relationship between the number of claims and outpatient annual Medicare reimbursement amount

In [12]:

```
model1 = smf.ols('MEDREIMB_OP ~ NumClaims',
              data=ben_summary_df).fit()
model1_rsquare = model1.rsquared
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            MEDREIMB_OP   R-squared:                       0.389
Model:                            OLS   Adj. R-squared:                  0.389
Method:                 Least Squares   F-statistic:                 7.399e+04
Date:                Tue, 28 May 2019   Prob (F-statistic):               0.00
Time:                        07:42:20   Log-Likelihood:            -1.0084e+06
No. Observations:              116352   AIC:                         2.017e+06
Df Residuals:                  116350   BIC:                         2.017e+06
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -158.7680      5.020    -31.628      0.000    -168.607    -148.929
NumClaims    322.3482      1.185    272.011      0.000     320.025     324.671
==============================================================================
Omnibus:                   174707.222   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        101199704.771
Skew:                           9.267   Prob(JB):                         0.00
Kurtosis:                     146.286   Cond. No.                         5.26
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

## Explantory Text

**An $R^2$ of .389 tells us that number of claims accounts for nearly 40% in the variance of reimbursement amounts. And as the scatter plot below further verifies, the two are positively coorelated. Every additional claim increases the reimbursement amount by ~$322.**

In [13]:

```
px.scatter(ben_summary_df,
           x = 'NumClaims',
           y = 'MEDREIMB_OP',
           trendline= True,
           title = 'Reimbursement Amounts Increase with # of Claims',
           labels=dict(NumClaims='Number of Claims',
                       MEDREIMB_OP = "Reimbursement Amounts"
                       ))
```

# Analysis 2

## Who are our health care super users?

## Explantory Text

- **The Pareto Principle, also known as the 80-20 rule, has shown up across an outstanding variety of fields. It says that a small number of things (people, products, actions) results in a disproportionate number of outcomes.**
- **The healthcare corrollary is simply that 80% of healthcare costs can be attributed to 20% of the patient population.**
- **With this in mind, a logical place to look for a condition to target is to identify our so called "health care super users" and determine whether they suffer disproportionately from any chronic conditions.**

**Let's get summary statistics on the number of claims column. Below, we can see that the average number of claims is 2.42 and 75% members submit four or fewer claims. What about our super users?**

In [14]:

```
ben_summary_df['NumClaims'].describe()
```

Out[14]:

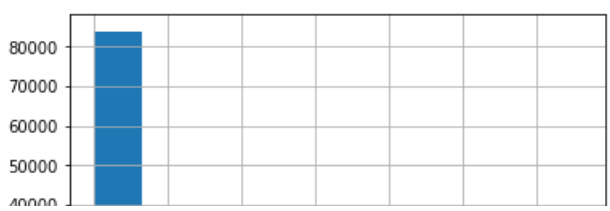
```
count    116352.000000
mean          2.422829
std           3.474703
min           0.000000
25%           0.000000
50%           1.000000
75%           4.000000
max          33.000000
Name: NumClaims, dtype: float64
```

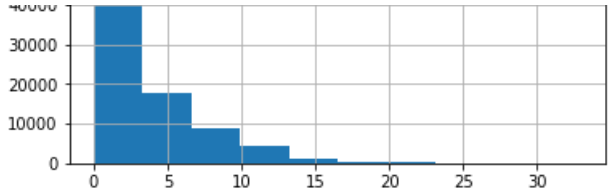
In [15]:

```
ben_summary_df['NumClaims'].hist().plot()
```

Out[15]:

```
[]
```






In [16]:

```
print("90% of users have", ben_summary_df['NumClaims'].quantile(0.90),"or fewer claims")
print("99% of users have", ben_summary_df['NumClaims'].quantile(0.99),"or fewer claims")
```

```
90% of users have 7.0 or fewer claims
99% of users have 14.0 or fewer claims
```

In [17]:

```
print("This means that", ben_summary_df[ben_summary_df['NumClaims'] > 7].count()[0],"users have mor
e than 7 claims")
print("And", ben_summary_df[ben_summary_df['NumClaims'] > 14].count()[0],"users have more than 14 c
laims. Let's call these our super users.")
```

```
This means that 11005 users have more than 7 claims
And 1134 users have more than 14 claims. Let's call these our super users.
```

**Create a new column called `IsSuperUtilizer`. If the value in `NumClaims` is greater than 14 than member receives a 'Y' to indicate Super User status. Else, an 'N'.**

In [18]:

```
#This function determines whether a value is greater than 14 and returns a Y if True.
def is_SuperUtilizer(row):
    if row['NumClaims'] > 14:
        val = 'Y'
    else:
        val = 'N'
    return val
```

In [19]:

```
#Function is applied column wise and the value is placed in a new column
ben_summary_df['IsSuperUtilizer'] = ben_summary_df.apply(is_SuperUtilizer, axis=1)
```

In [20]:

```
#This shows the function worked as intended
ben_summary_df[['DESYNPUF_ID','NumClaims','IsSuperUtilizer']].head(n=5)
```

Out[20]:

| | DESYNPUF_ID | NumClaims | IsSuperUtilizer |
|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 1.0 | N |
| 1 | 00016F745862898F | 0.0 | N |
| 2 | 0001FDD721E223DC | 0.0 | N |
| 3 | 00021CA6FF03E670 | 0.0 | N |
| 4 | 00024B3D2352D2D0 | 1.0 | N |

# Analysis 3

## Which chronic conditions are most prevalent among super users?

## Explantory Text

- **Now that we know who are super users are, let's see how often they appear to have a chronic condition.**
- **The screenshot below shows how these conditions are coded within our data so we know which columns to investigate**

In [21]:

```
from IPython.display import Image
Image(filename=r'C:\Users\Corey
Moser\Documents\Programming\Python\My_Projects\MorganStanley_Case_Study\CMS_MedicalCodes.JPG', wid
th=500)
```

Out[21]:

| | | |
|---|---|---|
| 13 | *SP_ALZHDMTA* | DESYNPUF: Chronic Condition: Alzheimer or related disorders or senile |
| 14 | *SP_CHF* | DESYNPUF: Chronic Condition: Heart Failure |
| 15 | *SP_CHRNKIDN* | DESYNPUF: Chronic Condition: Chronic Kidney Disease |
| 16 | *SP_CNCR* | DESYNPUF: Chronic Condition: Cancer |
| 17 | *SP_COPD* | DESYNPUF: Chronic Condition: Chronic Obstructive Pulmonary Disease |
| 18 | *SP_DEPRESSN* | DESYNPUF: Chronic Condition: Depression |
| 19 | *SP_DIABETES* | DESYNPUF: Chronic Condition: Diabetes |
| 20 | *SP_ISCHMCHT* | DESYNPUF: Chronic Condition: Ischemic Heart Disease |
| 21 | *SP_OSTEOPRS* | DESYNPUF: Chronic Condition: Osteoporosis |
| 22 | *SP_RA_OA* | DESYNPUF: Chronic Condition: rheumatoid arthritis and osteoarthritis (RA/OA) |
| 23 | *SP_STRKETIA* | DESYNPUF: Chronic Condition: Stroke/transient Ischemic Attack |

In [22]:

```
#Grab the columns shown above from our data
ben_summary_df.columns[14:25]
```

Out[22]:

```
Index(['SP_ALZHDMTA', 'SP_CHF', 'SP_CHRNKIDN', 'SP_CNCR', 'SP_COPD',
       'SP_DEPRESSN', 'SP_DIABETES', 'SP_ISCHMCHT', 'SP_OSTEOPRS', 'SP_RA_OA',
       'SP_STRKETIA'],
      dtype='object')
```

In [23]:

```
#Assign these to a variable so we can access them easily. Will use this momentarily
chronic_conditions = ben_summary_df.columns[14:25]
chronic_conditions
```

Out[23]:

```
Index(['SP_ALZHDMTA', 'SP_CHF', 'SP_CHRNKIDN', 'SP_CNCR', 'SP_COPD',
       'SP_DEPRESSN', 'SP_DIABETES', 'SP_ISCHMCHT', 'SP_OSTEOPRS', 'SP_RA_OA',
       'SP_STRKETIA'],
      dtype='object')
```

## Explantory Text

- **This pivot table shows us how diabetes affects our population and whether those affected are healthcare super users or not.**
- **Below we can see that 44,060 employees of 116,352 have diabetes.**
- **More importantly, of our super users 1,008 out of 1,134 have it; this is a huge percentage of that population.**

In [24]:

```
ben_summary_df.pivot_table(index='SP_DIABETES',
                           columns='IsSuperUtilizer',
                           values='DESYNPUF_ID',
                           aggfunc='count',
                           margins=True)
```

Out[24]:

| IsSuperUtilizer | N | Y | All |
|---|---|---|---|
| SP_DIABETES | | | |
| 1 | 43052 | 1008 | 44060 |
| 2 | 72166 | 126 | 72292 |
| All | 115218 | 1134 | 116352 |

## Explantory Text

- **Lets' create a series of pivot tables.For each, we'll show how each of our chronic conditions affect differents portions of our patient population and how prevalent our super users are among them.**

In [25]:

```
for i in chronic_conditions:
    print(ben_summary_df.pivot_table(index=i,
                            columns='IsSuperUtilizer',
                            values='DESYNPUF_ID',
                            aggfunc='count',
                            margins=True))
    print("")
    print("---------------")
    print("")
```

```
IsSuperUtilizer       N     Y     All
SP_ALZHDMTA
1                 21800   610   22410
2                 93418   524   93942
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_CHF
1                 32301   854   33155
2                 82917   280   83197
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_CHRNKIDN
1                 17985   701   18686
2                 97233   433   97666
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_CNCR
1                  7183   232    7415
2                108035   902  108937
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_COPD
1                 15208   535   15743
2                100010   599  100609
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_DEPRESSN
1                 24150   690   24840
2                 91068   444   91512
All              115218  1134  116352
```

```
---------------

IsSuperUtilizer       N     Y     All
SP_DIABETES
1                 43052  1008   44060
2                 72166   126   72292
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_ISCHMCHT
1                 47925  1017   48942
2                 67293   117   67410
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_OSTEOPRS
1                 19722   455   20177
2                 95496   679   96175
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_RA_OA
1                 17485   431   17916
2                 97733   703   98436
All              115218  1134  116352

---------------

IsSuperUtilizer       N     Y     All
SP_STRKETIA
1                  5036   187    5223
2                110182   947  111129
All              115218  1134  116352

---------------
```

### Explantory Text

- **Of our 11 chronic diseases, diabetes and ischemic heart disease are most prevalent, affecting ~89% of the super-user population.**
- **Said another way, having one of these two conditions makes someone much more likely to use a lot of health care.**
- **Both too affect large portions of our employee population regardless of whether someone is a health care super user (42% for ischemic heart disease and 37% for diabetes).**
- **It follow then that we can decrease healthcare usage by targeting the factors that lead to these conditions.**

# Load the prescription drug events data file

### Explantory Text

**We can also use prescription drug event data, which conveniently identifies members by the same `DESYNPUF_ID`, column to learn more about Rx usage**

In [26]:

```
#load the rx data set
rx_df = pd.read_csv(r"C:\Users\Corey
Moser\Documents\Programming\Python\My_Projects\MorganStanley_Case_Study\DE1_0_2008_to_2010_Prescri
 Drug_Events_Sample_1.csv")
```

```
rx_df.head()
```

```
C:\Users\Corey Moser\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2785:
DtypeWarning:

Columns (4) have mixed types. Specify dtype option on import or set low_memory=False.
```

Out[26]:

| | DESYNPUF_ID | PDE_ID | SRVC_DT | year | PROD_SRVC_ID | QTY_DSPNSD_NUM | DAYS_SUPLY_NUM | PTNT_PAY_AMT | TO |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 2.336640e+14 | 20080103 | 2008 | 247037252 | 30 | 20 | 10 | |
| 1 | 00013D2EFD8E45D1 | 2.336440e+14 | 20080105 | 2008 | 223039502 | 10 | 10 | 0 | |
| 2 | 00013D2EFD8E45D1 | 2.339740e+14 | 20080109 | 2008 | 364724812 | 120 | 30 | 10 | |
| 3 | 00013D2EFD8E45D1 | 2.335740e+14 | 20080123 | 2008 | 179180672 | 30 | 30 | 0 | |
| 4 | 00013D2EFD8E45D1 | 2.330240e+14 | 20080124 | 2008 | 58016005300 | 30 | 30 | 70 | |

## Explantory Text

**Similar to what we did earlier, we can get a count of how many time each user in our original file appears in the new data set. Essentially, how many prescription drug events did each member have in 2008?**

In [27]:

```
rx_counts = rx_df['DESYNPUF_ID'].value_counts() #Get a count of each value in the claims_df data set
rx_counts_dict = rx_counts.to_dict() #convert this to a dictonary format
```

In [28]:

```
ben_summary_df['NumRxEvents'] =  ben_summary_df['DESYNPUF_ID'].map(rx_counts_dict)
```

In [29]:

```
#View 10 entries in our rx_counts_dict to ensure we are getting the results we expect
j = 10
for k, v in rx_counts_dict.items():
    if j > 0:
        print(k,v)
        j -=1
```

```
29A392121473E854 144
01854EABDA796A16 130
04CD7DF7BB1A5968 128
1BA2FE3C2B0F337E 128
213DDA8B2B5FFDBE 128
15BFA405D4FD1BCA 126
15F6460FD971421B 125
1F006C9F51F91886 125
122426591B38C914 123
03D262AF75723EF7 121
```

In [30]:

```
#We need to recode employees who did not have an Rx claim. Numpy uses NaN (not a number) by default.
# The following code changes this to 0.
ben_summary_df =  ben_summary_df.replace({
    'NumRxEvents': np.nan}, 0)
```

In [31]:

```
ben_summary_df[['DESYNPUF_ID','NumRxEvents']].head(n=5)
```

Out[31]:

| | DESYNPUF_ID | NumRxEvents |
|---|---|---|
| 0 | 00013D2EFD8E45D1 | 80.0 |
| 1 | 00016F745862898F | 0.0 |
| 2 | 0001FDD721E223DC | 24.0 |
| 3 | 00021CA6FF03E670 | 0.0 |
| 4 | 00024B3D2352D2D0 | 0.0 |

In [32]:

```
ben_summary_df['NumRxEvents'].describe()
```

Out[32]:

```
count    116352.000000
mean          3.244723
std          11.403043
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max         144.000000
Name: NumRxEvents, dtype: float64
```

# Analysis 4

## What does the distribution of Rx Claims look like?

In [33]:

```
#Look at the first 5 values. Shows how how many had 0 through 5 prescription drug events.
# As expected we see a huge dropoff as the number of prescriptions increases
ben_summary_df['NumRxEvents'].value_counts().sort_index()[:5]
```

Out[33]:

```
0.0    102498
1.0       762
2.0       597
3.0       422
4.0       315
5.0       230
Name: NumRxEvents, dtype: int64
```

In [34]:

```
print(ben_summary_df['NumRxEvents'].value_counts().sort_index()[0],"members had no Rx events
while", ben_summary_df['NumRxEvents'].value_counts().sort_index()[1:].sum(), "had one or more Rx
events")
```

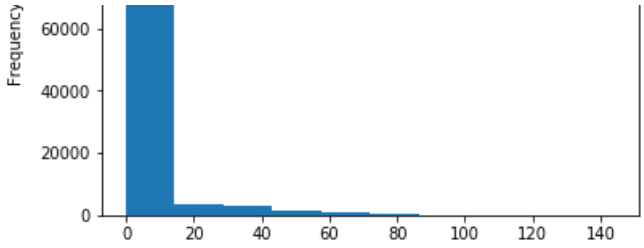
```
102498 members had no Rx events while 13854 had one or more Rx events
```

In [35]:

```
ben_summary_df['NumRxEvents'].plot.hist()
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2642d2d6550>
```




In [56]:

```
ben_summary_df['NumRxEvents'].describe()
```

Out[56]:

```
count    116352.000000
mean          3.244723
std          11.403043
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max         144.000000
Name: NumRxEvents, dtype: float64
```

### Explantory Text

- **Rx claim events are definitely right skewed.**
- **The vast majority of users had none while a few had over 100. The average was 3.24.**

## Analysis 5

### Does our Rx data help us better predict super users?

In [36]:

```
ben_summary_df.columns
```

Out[36]:

```
Index(['DESYNPUF_ID', 'Number of Claims', 'SuperUtilizer', 'BENE_BIRTH_DT',
       'BENE_DEATH_DT', 'BENE_SEX_IDENT_CD', 'BENE_RACE_CD', 'BENE_ESRD_IND',
       'SP_STATE_CODE', 'BENE_COUNTY_CD', 'BENE_HI_CVRAGE_TOT_MONS',
       'BENE_SMI_CVRAGE_TOT_MONS', 'BENE_HMO_CVRAGE_TOT_MONS',
       'PLAN_CVRG_MOS_NUM', 'SP_ALZHDMTA', 'SP_CHF', 'SP_CHRNKIDN', 'SP_CNCR',
       'SP_COPD', 'SP_DEPRESSN', 'SP_DIABETES', 'SP_ISCHMCHT', 'SP_OSTEOPRS',
       'SP_RA_OA', 'SP_STRKETIA', 'MEDREIMB_IP', 'BENRES_IP', 'PPPYMT_IP',
       'MEDREIMB_OP', 'BENRES_OP', 'PPPYMT_OP', 'MEDREIMB_CAR', 'BENRES_CAR',
       'PPPYMT_CAR', 'NumClaims', 'IsSuperUtilizer', 'NumRxEvents'],
      dtype='object')
```

### Explantory Text

**Let's create a new dataframe which will be a smaller version of our original data. The columns we'll use include:**

- **`DESYNPUF_ID` = member ID**
- **`NumRxEvents` = # of prescription drug events for each member**
- **`SP_ISCHMCHT` = Does this person have ischemic heart disease (yes/no)**
- **`SP_DIABETES` = Does this person have diabetes (yes/no)**
- **`SuperUtilizer` = Is this person a super utilizer (yes/no)**

In [37]:

```
df2 =
ben_summary_df[['DESYNPUF_ID','NumRxEvents','SP_ISCHMCHT','SP_DIABETES','SuperUtilizer']].copy()
```

In [38]:

```
df2.head(n=5)
```

Out[38]:

| | DESYNPUF_ID | NumRxEvents | SP_ISCHMCHT | SP_DIABETES | SuperUtilizer |
|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 80.0 | 2 | 2 | N |
| 1 | 00016F745862898F | 0.0 | 2 | 2 | N |
| 2 | 0001FDD721E223DC | 24.0 | 2 | 2 | N |
| 3 | 00021CA6FF03E670 | 0.0 | 2 | 2 | N |
| 4 | 00024B3D2352D2D0 | 0.0 | 2 | 2 | N |

In [39]:

```
#recode all 2s to 0. We want 1 to = yes (person has disease) and 0 = no (person does not)
df2['SP_ISCHMCHT'] = df2['SP_ISCHMCHT'].replace({
    2: 0})
```

In [40]:

```
df2.head(n=5)
```

Out[40]:

| | DESYNPUF_ID | NumRxEvents | SP_ISCHMCHT | SP_DIABETES | SuperUtilizer |
|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 80.0 | 0 | 2 | N |
| 1 | 00016F745862898F | 0.0 | 0 | 2 | N |
| 2 | 0001FDD721E223DC | 24.0 | 0 | 2 | N |
| 3 | 00021CA6FF03E670 | 0.0 | 0 | 2 | N |
| 4 | 00024B3D2352D2D0 | 0.0 | 0 | 2 | N |

In [41]:

```
#recode all 2s to 0. We want 1 to = yes (person has disease) and 0 = no (person does not)
df2['SP_DIABETES'] = df2['SP_DIABETES'].replace({
    2: 0})
df2.head(n=5)
```

Out[41]:

| | DESYNPUF_ID | NumRxEvents | SP_ISCHMCHT | SP_DIABETES | SuperUtilizer |
|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 80.0 | 0 | 0 | N |
| 1 | 00016F745862898F | 0.0 | 0 | 0 | N |
| 2 | 0001FDD721E223DC | 24.0 | 0 | 0 | N |
| 3 | 00021CA6FF03E670 | 0.0 | 0 | 0 | N |
| 4 | 00024B3D2352D2D0 | 0.0 | 0 | 0 | N |

In [42]:

```
#recode all Y and N to 1 and 0 respectively
df2['IsSuperUtilizer'] = df2['SuperUtilizer'].replace({
    'N': 0,
'Y':1})
```

In [43]:

```
model2 = smf.logit('IsSuperUtilizer ~ NumRxEvents + SP_ISCHMCHT + SP_DIABETES',
```

```
              data=df2).fit()
model2_rsquare = model2.prsquared
print(model2.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.047749
         Iterations 11
                           Logit Regression Results
==============================================================================
Dep. Variable:        IsSuperUtilizer   No. Observations:               116352
Model:                          Logit   Df Residuals:                   116348
Method:                           MLE   Df Model:                            3
Date:                Tue, 28 May 2019   Pseudo R-squ.:                  0.1292
Time:                        07:43:10   Log-Likelihood:                -5555.7
converged:                       True   LL-Null:                       -6379.9
                                        LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      -7.0133      0.109    -64.536      0.000      -7.226      -6.800
NumRxEvents     0.0076      0.002      4.283      0.000       0.004       0.011
SP_ISCHMCHT     1.6143      0.105     15.408      0.000       1.409       1.820
SP_DIABETES     1.8371      0.101     18.103      0.000       1.638       2.036
==============================================================================
```

### Explantory Text

- **As expected, those classified as having one of our two chronic illnesses increases the likelihood of that member being a health care super user.**
- **Surprisingly, the number of Rx events does not.</b>**

## Analysis 6

### Does total Rx spend by member coorelate with health care super users?

### Explantory Text

**Maybe instead of Rx events we want to find the gross drug cost? This might help us discern if one condition is more expensive to treat than another.**

In [44]:

```
rx_df.columns
```

Out[44]:

```
Index(['DESYNPUF_ID', 'PDE_ID', 'SRVC_DT', 'year', 'PROD_SRVC_ID',
       'QTY_DSPNSD_NUM', 'DAYS_SUPLY_NUM', 'PTNT_PAY_AMT', 'TOT_RX_CST_AMT'],
      dtype='object')
```

### Explantory Text

- **From our Rx dataset, we can group by member ID ( `DESYNPUF_ID` ) and sum on `TOT_RX_CST_AMT` , defined as the gross drug cost.**
- **I believe this is a better representation of the cost of a patient's Rx spend rather than the true amount they paid, which is influenced by copayments, coinsurance, deductibles, rebates etc..**

In [45]:

```
#group by member id, sum the gross drug amount and display the first 5 results
rx_df.groupby('DESYNPUF_ID')['TOT_RX_CST_AMT'].sum()[:5]
```

Out[45]:

```
DESYNPUF_ID
00013D2EFD8E45D1    6640
0001FDD721E223DC    1030
```

```
0001FDD721E223DC    1030
00036A21B65B0206     760
000489E7EAAD463F    1420
00048EF1F4791C68    1350
Name: TOT_RX_CST_AMT, dtype: int64
```

In [46]:

```
#.describe() outputs summary statistics on the groupby operation above
rx_df.groupby('DESYNPUF_ID')['TOT_RX_CST_AMT'].sum().describe()
```

Out[46]:

```
count    13868.000000
mean      1675.186761
std       1751.246494
min          0.000000
25%        360.000000
50%       1100.000000
75%       2382.500000
max      13330.000000
Name: TOT_RX_CST_AMT, dtype: float64
```

## Explantory Text

- **The average Rx spending among members in 2008 was &#36 1,675**
- **Many members had no Rx spend**
- **In one case, a member had Rx spending totaling &#36 13,300.**

In [47]:

```
#Add each employee's Rx spend to our df2 data frame
rx_spend = rx_df.groupby('DESYNPUF_ID')['TOT_RX_CST_AMT'].sum()
rx_spend_dict = rx_spend.to_dict()
```

In [48]:

```
df2['RxSpend'] = df2['DESYNPUF_ID'].map(rx_spend_dict)
```

In [49]:

```
df2.head()
```

Out[49]:

| | DESYNPUF_ID | NumRxEvents | SP_ISCHMCHT | SP_DIABETES | SuperUtilizer | IsSuperUtilizer | RxSpend |
|---|---|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 80.0 | 0 | 0 | N | 0 | 6640.0 |
| 1 | 00016F745862898F | 0.0 | 0 | 0 | N | 0 | NaN |
| 2 | 0001FDD721E223DC | 24.0 | 0 | 0 | N | 0 | 1030.0 |
| 3 | 00021CA6FF03E670 | 0.0 | 0 | 0 | N | 0 | NaN |
| 4 | 00024B3D2352D2D0 | 0.0 | 0 | 0 | N | 0 | NaN |

In [50]:

```
df2 = df2.replace({
    'RxSpend':np.NaN}, 0)
```

In [51]:

```
df2.head(n=5)
```

Out[51]:

| | DESYNPUF_ID | NumRxEvents | SP_ISCHMCHT | SP_DIABETES | SuperUtilizer | IsSuperUtilizer | RxSpend |
|---|---|---|---|---|---|---|---|
| 0 | 00013D2EFD8E45D1 | 80.0 | 0 | 0 | N | 0 | 6640.0 |
| 1 | 00016F745862898F | 0.0 | 0 | 0 | N | 0 | 0.0 |
| 2 | 0001FDD721E223DC | 24.0 | 0 | 0 | N | 0 | 1030.0 |
| 3 | 00021CA6FF03E670 | 0.0 | 0 | 0 | N | 0 | 0.0 |
| 4 | 00024B3D2352D2D0 | 0.0 | 0 | 0 | N | 0 | 0.0 |

In [52]:

```
model3 = smf.logit('IsSuperUtilizer ~ RxSpend',
              data=df2).fit()
model3_rsquare = model3.prsquared
print(model3.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.054521
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:        IsSuperUtilizer   No. Observations:               116352
Model:                          Logit   Df Residuals:                   116350
Method:                           MLE   Df Model:                            1
Date:                Tue, 28 May 2019   Pseudo R-squ.:                0.005679
Time:                        07:43:11   Log-Likelihood:                -6343.6
converged:                       True   LL-Null:                       -6379.9
                                        LLR p-value:                1.707e-17
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -4.6864      0.031   -149.484      0.000      -4.748      -4.625
RxSpend        0.0002   2.17e-05     10.031      0.000       0.000       0.000
==============================================================================
```

### Explantory Text

**According to out logistic regression output, Rx spend does not predict super user status. There are a number of reasons this could be the case:**

1. **We're chosen an incorrect measure of Rx spend.**
2. **Having a high number of medical events (in terms of count of claims) does not automatically correspond to expensive Rx treatment.**
3. **Incorrect code in the analysis is not fully capturing the relationship between the two.**

## Analysis 7

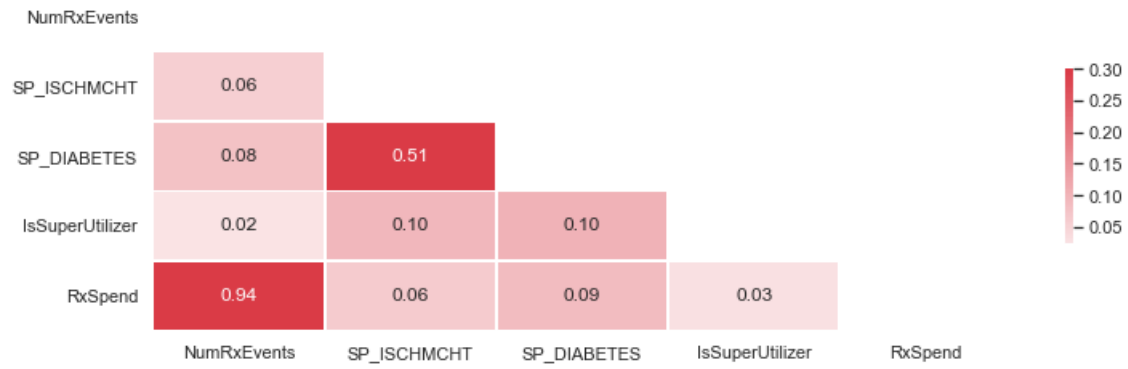
### Create a correlation matrix to visualize the relationship between the variables in our new dataframe

In [53]:

```
sns.set(style="white")
corr = df2.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(12, 4))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=False, linewidths=.5, annot=True, fmt=".2f", cbar_kws={"shrink": .5})
```

Out[53]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2642e0856a0>
```




### Explantory Text

- **Neither Rx spend nor NumRxEvents has a very high correlation with super users.**
- **Let's try one more analysis looking at the average Rx spend betweent the two groups.**

## Analysis 8

### How does average Rx spend differ among super users with one of our two target chronic conditions?

In [54]:

```
df2.pivot_table(index='SP_DIABETES',
                            columns='IsSuperUtilizer',
                            values='RxSpend',
                            aggfunc='mean',
                            margins=True)
```

Out[54]:

| IsSuperUtilizer | 0 | 1 | All |
|---|---|---|---|
| SP_DIABETES | | | |
| 0 | 143.665577 | 339.206349 | 144.006391 |
| 1 | 286.338381 | 463.253968 | 290.385837 |
| All | 196.976254 | 449.470899 | 199.437139 |

In [55]:

```
df2.pivot_table(index='SP_ISCHMCHT',
                            columns='IsSuperUtilizer',
                            values='RxSpend',
                            aggfunc='mean',
                            margins=True)
```

Out[55]:

| IsSuperUtilizer | 0 | 1 | All |
|---|---|---|---|
| SP_ISCHMCHT | | | |
| 0 | 155.821854 | 478.547009 | 156.381991 |
| 1 | 254.762441 | 446.125860 | 258.738915 |
| All | 196.976254 | 449.470899 | 199.437139 |

### Explantory Text

**As expected Rx spend is higher both for those with and without our target conditions and higher still among the super users**

# Closing Remarks

- **This notebook file is intended only to show the different approaches I might take toward investigating our initial question: how to identify a chronic condition within our employee population.**
- **Since I do not know what specific information is available for analysis I made a number of assumptions and chose certain analyses based on the data available to me from cms.gov.**
- **Of course there are many ways to approach the problem and certain methods above may not be appropriate when applied to employee enterprise data.**
- **I hope if nothing else this file demonstrates my analytical abilities and approach to answering a complex question.**
- **Further, I believe notebook-style analysis is the future of team-based analytical projects as they follow a more linear style of thinking and incorporate the best of both MS Word and Excel while better handling large data sets.**

In [ ]: