In [1]:
```
#Import the libraries needed
from pandas import Series, DataFrame
from pylab import rcParams
from scipy.stats import spearmanr
from sklearn import datasets
from sklearn import metrics
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import scale
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy
import seaborn as sns
import sklearn as sl
import sklearn.metrics as sm
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [2]:
```
pip install statsmodels
```

```
Requirement already satisfied: statsmodels in c:\users\brittany\anaconda3\lib\site-packages (0.13.2)
Requirement already satisfied: numpy>=1.17 in c:\users\brittany\anaconda3\lib\site-packages (from statsmodels) (1.21.5)
Requirement already satisfied: scipy>=1.3 in c:\users\brittany\anaconda3\lib\site-packages (from statsmodels) (1.7.3)
Requirement already satisfied: pandas>=0.25 in c:\users\brittany\anaconda3\lib\site-packages (from statsmodels) (1.4.3)
Requirement already satisfied: patsy>=0.5.2 in c:\users\brittany\anaconda3\lib\site-packages (from statsmodels) (0.5.2)
Requirement already satisfied: packaging>=21.3 in c:\users\brittany\anaconda3\lib\site-packages (from statsmodels) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\brittany\anaconda3\lib\site-packages (from packaging>=21.3-
>statsmodels) (3.0.4)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\brittany\anaconda3\lib\site-packages (from pandas>=0.25->stat
smodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\brittany\anaconda3\lib\site-packages (from pandas>=0.25->statsmodels)
(2022.1)
Requirement already satisfied: six in c:\users\brittany\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
#Load the dataset
df = pd.read_csv("medical clean 1.1.23.csv")
```

In [4]:
```
#Check total rows/columns
rows = df.shape[0]
cols = df.shape[1]
# display the number of rows and columns
print("Number of Rows: " + str(rows))
print("Number of Columns: " + str(cols))
```

```
Number of Rows: 10000
Number of Columns: 50
```

In [5]:
```
#check for nan and null values
df.isna().any()
```

Out[5]:
```
CaseOrder             False
Customer_id           False
Interaction           False
UID                   False
City                  False
State                 False
County                False
Zip                   False
Lat                   False
Lng                   False
Population            False
Area                  False
TimeZone              False
Job                   False
Children              False
Age                   False
Income                False
Marital               False
Gender                False
ReAdmis               False
VitD_levels           False
Doc_visits            False
Full_meals_eaten      False
vitD_supp             False
Soft_drink            False
Initial_admin         False
HighBlood             False
Stroke                False
Complication_risk     False
Overweight            False
Arthritis             False
Diabetes              False
Hyperlipidemia        False
BackPain              False
Anxiety               False
Allergic_rhinitis     False
Reflux_esophagitis    False
Asthma                False
Services              False
Initial_days          False
TotalCharge           False
Additional_charges    False
Item1                 False
Item2                 False
Item3                 False
Item4                 False
Item5                 False
Item6                 False
Item7                 False
Item8                 False
dtype: bool
```

In [6]:
```
df.isnull().any()
```

Out[6]:
```
CaseOrder             False
Customer_id           False
Interaction           False
UID                   False
City                  False
State                 False
County                False
Zip                   False
Lat                   False
Lng                   False
Population            False
Area                  False
TimeZone              False
Job                   False
Children              False
Age                   False
Income                False
Marital               False
Gender                False
ReAdmis               False
VitD_levels           False
Doc_visits            False
Full_meals_eaten      False
vitD_supp             False
Soft_drink            False
Initial_admin         False
HighBlood             False
Stroke                False
Complication_risk     False
Overweight            False
Arthritis             False
Diabetes              False
Hyperlipidemia        False
BackPain              False
Anxiety               False
Allergic_rhinitis     False
Reflux_esophagitis    False
Asthma                False
Services              False
Initial_days          False
TotalCharge           False
Additional_charges    False
Item1                 False
Item2                 False
Item3                 False
Item4                 False
Item5                 False
Item6                 False
Item7                 False
Item8                 False
dtype: bool
```

In [7]:
```
# count of unique values in each column
print(df.nunique())
```

```
CaseOrder             10000
Customer_id           10000
Interaction           10000
UID                   10000
City                   6072
State                    52
County                 1607
Zip                    8612
Lat                    8588
Lng                    8725
Population             5951
Area                      3
TimeZone                 26
Job                     639
Children                 11
Age                      72
Income                 9993
Marital                   5
Gender                    3
ReAdmis                   2
VitD_levels            9976
Doc_visits                9
Full_meals_eaten          8
vitD_supp                 6
Soft_drink                2
Initial_admin             3
HighBlood                 2
Stroke                    2
Complication_risk         3
Overweight                2
Arthritis                 2
Diabetes                  2
Hyperlipidemia            2
BackPain                  2
Anxiety                   2
Allergic_rhinitis         2
Reflux_esophagitis        2
Asthma                    2
Services                  4
Initial_days           9997
TotalCharge            9997
Additional_charges     9418
Item1                     8
Item2                     7
Item3                     8
Item4                     7
Item5                     7
Item6                     7
Item7                     7
Item8                     7
dtype: int64
```

In [10]:
```
#drop all columns and rows not being used
to_drop = ['CaseOrder', 'Customer_id','Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area',
df.drop(to_drop, inplace=True, axis=1)
```

In [15]:
```
#Change object to category for ReAdmis
df["ReAdmis"] = df["ReAdmis"].astype('category')
#Change object to category for Anxiety
df["Anxiety"] = df["Anxiety"].astype('category')
#Change object to category for Overweight
df["Overweight"] = df["Overweight"].astype('category')
#Change object to category for Arthritis
df["Arthritis"] = df["Arthritis"].astype('category')
#Change object to category for HighBlood
df["HighBlood"] = df["HighBlood"].astype('category')
#Change object to category for Stroke
df["Stroke"] = df["Stroke"].astype('category')
#Change object to category for Gender
df["Gender"] = df["Gender"].astype('category')
#Change object to category for Diabetes
df["Diabetes"] = df["Diabetes"].astype('category')
#Change object to category for Asthma
df["Asthma"] = df["Asthma"].astype('category')
#Change object to category for back pain
df["BackPain"] = df["BackPain"].astype('category')
#Change object to category for Hyperlipidemia
df["Hyperlipidemia"] = df["Hyperlipidemia"].astype('category')
```

In [16]:
```
#check data types
df.dtypes
```

Out[16]:
```
Gender            category
ReAdmis           category
Doc_visits           int64
HighBlood         category
Stroke            category
Overweight        category
Arthritis         category
Diabetes          category
Hyperlipidemia    category
BackPain          category
Anxiety           category
Asthma            category
dtype: object
```

In [18]:
```
#Run get dummies on categorical
pd.get_dummies(df, columns = ['Gender', 'ReAdmis', 'Anxiety', 'Overweight', 'HighBlood','Arthritis', 'Stroke', 'Diabetes', 'Asthm
```

Out[18]:

	Doc_visits	Gender_Female	Gender_Male	Gender_Nonbinary	ReAdmis_No	ReAdmis_Yes	Anxiety_No	Anxiety_Yes	Overweight_No	Overweight_Yes	...
0	6	0	1	0	1	0	0	1	1	0	...
1	4	1	0	0	1	0	1	0	0	1	...
2	4	1	0	0	1	0	1	0	0	1	...
3	4	0	1	0	1	0	1	0	1	0	...
4	5	1	0	0	1	0	1	0	1	0	...
...	...	...	...	...	...	...	...	...	...	...	...
9995	4	0	1	0	1	0	0	1	1	0	...
9996	5	0	1	0	0	1	1	0	0	1	...
9997	4	1	0	0	0	1	0	1	0	1	...
9998	5	0	1	0	0	1	1	0	0	1	...
9999	5	1	0	0	0	1	1	0	0	1	...

10000 rows × 24 columns

In [19]:
```
#create new df with the get dummies responses
dfupdated = pd.get_dummies(df, columns = ['ReAdmis', 'Anxiety', 'Overweight', 'Arthritis','HighBlood', 'Gender', 'Stroke', 'Diabe
```

In [20]:
```
#Drop columns to avoid dummy variable trap
to_drop = ['Gender_Male', 'Gender_Nonbinary','ReAdmis_No', 'Anxiety_No', 'Overweight_No', 'Stroke_No', 'Diabetes_No', 'Asthma_No'
dfupdated.drop(to_drop, inplace=True, axis=1)
```

In [21]:
```
#Run a describe
dfupdated.describe()
```

Out[21]:

	Doc_visits	ReAdmis_Yes	Anxiety_Yes	Overweight_Yes	Arthritis_Yes	HighBlood_Yes	Gender_Female	Stroke_Yes	Diabetes_Yes	Asthma_Yes	Hy
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.00000	
mean	5.012200	0.366900	0.321500	0.709400	0.357400	0.409000	0.501800	0.199300	0.27380	0.28930	
std	1.045734	0.481983	0.467076	0.454062	0.479258	0.491674	0.500022	0.399494	0.44593	0.45346	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	
25%	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	
50%	5.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.00000	0.00000	
75%	6.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.00000	1.00000	
max	9.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000	1.00000	

In [23]:
```
# Import statistics Library
import statistics
```

In [26]:
```
#view the mean
print(dfupdated.mean())
```

```
Doc_visits            5.0122
ReAdmis_Yes           0.3669
Anxiety_Yes           0.3215
Overweight_Yes        0.7094
Arthritis_Yes         0.3574
HighBlood_Yes         0.4090
Gender_Female         0.5018
Stroke_Yes            0.1993
Diabetes_Yes          0.2738
Asthma_Yes            0.2893
Hyperlipidemia_Yes    0.3372
BackPain_Yes          0.4114
dtype: float64
```

In [30]:
```
print(dfupdated['ReAdmis_Yes'].mode())
```

```
0    0
Name: ReAdmis_Yes, dtype: uint8
```

In [32]:
```
print(dfupdated['Anxiety_Yes'].mode())
```

```
0    0
Name: Anxiety_Yes, dtype: uint8
```

In [33]:
```
print(dfupdated['Overweight_Yes'].mode())
```

```
0    1
Name: Overweight_Yes, dtype: uint8
```

In [34]:
```
print(dfupdated['Arthritis_Yes'].mode())
```

```
0    0
Name: Arthritis_Yes, dtype: uint8
```

In [35]:
```
print(dfupdated['HighBlood_Yes'].mode())
```

```
0    0
Name: HighBlood_Yes, dtype: uint8
```

In [36]:
```
print(dfupdated['Asthma_Yes'].mode())
```

```
0    0
Name: Asthma_Yes, dtype: uint8
```

In [37]:
```
print(dfupdated['Gender_Female'].mode())
```

```
0    1
Name: Gender_Female, dtype: uint8
```

In [38]:
```
print(dfupdated['Stroke_Yes'].mode())
```

```
0    0
Name: Stroke_Yes, dtype: uint8
```

In [39]:
```
print(dfupdated['Diabetes_Yes'].mode())
```

```
0    0
Name: Diabetes_Yes, dtype: uint8
```

In [40]:
```
print(dfupdated['Hyperlipidemia_Yes'].mode())
```

```
0    0
Name: Hyperlipidemia_Yes, dtype: uint8
```

In [41]:
```
print(dfupdated['BackPain_Yes'].mode())
```

```
0    0
Name: BackPain_Yes, dtype: uint8
```

In [42]:
```
print(dfupdated['Doc_visits'].mode())
```

```
0    5
Name: Doc_visits, dtype: int64
```

In [43]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Doc_visits = dfupdated["Doc_visits"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Doc_visits)
spearmanr_coff
```

Out[43]: -0.0006162697767207155

In [44]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Gender_Female = dfupdated["Gender_Female"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Gender_Female)
spearmanr_coff
```

Out[44]: -0.011662563587899893

In [45]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Anxiety_Yes = dfupdated["Anxiety_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Anxiety_Yes)
spearmanr_coff
```

Out[45]: 0.0024062642580066325

In [46]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Overweight_Yes = dfupdated["Overweight_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Overweight_Yes)
spearmanr_coff
```

Out[46]: -0.008586002506313991

In [47]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Stroke_Yes = dfupdated["Stroke_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Stroke_Yes)
spearmanr_coff
```

Out[47]: 0.0009184539170458379

In [48]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Diabetes_Yes = dfupdated["Diabetes_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Diabetes_Yes)
spearmanr_coff
```

Out[48]: -0.00305812979631911

In [49]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Asthma_Yes = dfupdated["Asthma_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Asthma_Yes)
spearmanr_coff
```

Out[49]: -0.017132793168549827

In [50]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Hyperlipidemia_Yes = dfupdated["Hyperlipidemia_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Hyperlipidemia_Yes)
spearmanr_coff
```

Out[50]: 0.004306908097121766

In [51]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
BackPain_Yes = dfupdated["BackPain_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,BackPain_Yes)
spearmanr_coff
```

Out[51]: 0.013312789543203096

In [52]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
HighBlood_Yes = dfupdated["HighBlood_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,HighBlood_Yes)
spearmanr_coff
```

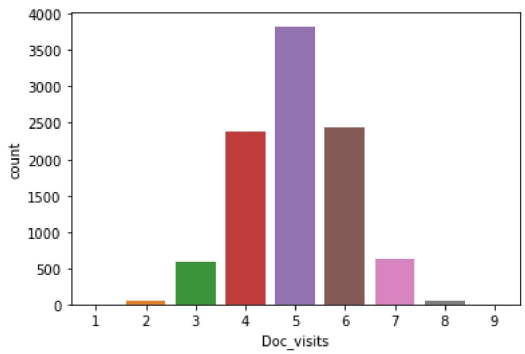
Out[52]: 0.0022700537328705066

In [53]:
```
#Check independence of each variable with Spearman Coefficient
ReAdmis_Yes= dfupdated["ReAdmis_Yes"]
Arthritis_Yes = dfupdated["Arthritis_Yes"]
#Find the Spearmen Cofficient.
spearmanr_coff, p_value = spearmanr(ReAdmis_Yes,Arthritis_Yes)
spearmanr_coff
```

Out[53]: 0.007663036548941127

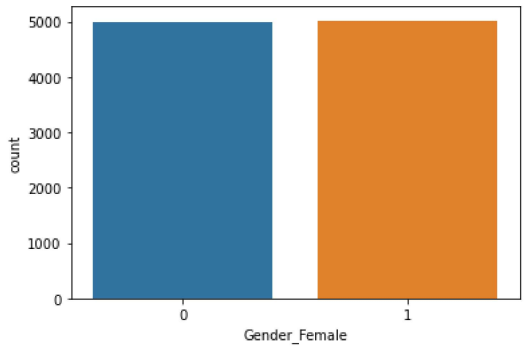
In [54]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Doc_visits', data=dfupdated)
```

Out[54]: <AxesSubplot:xlabel='Doc_visits', ylabel='count'>



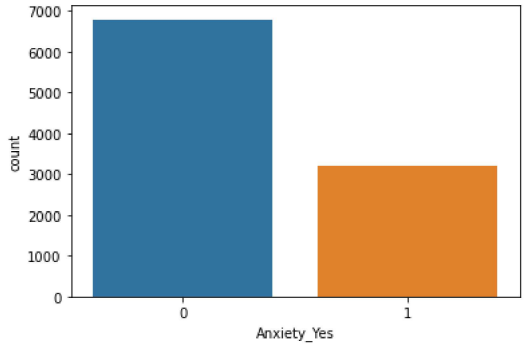
In [55]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Gender_Female', data=dfupdated)
```

Out[55]: <AxesSubplot:xlabel='Gender_Female', ylabel='count'>



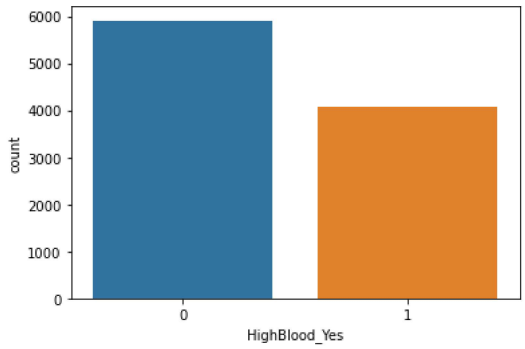
In [56]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Anxiety_Yes', data=dfupdated)
```

Out[56]: <AxesSubplot:xlabel='Anxiety_Yes', ylabel='count'>



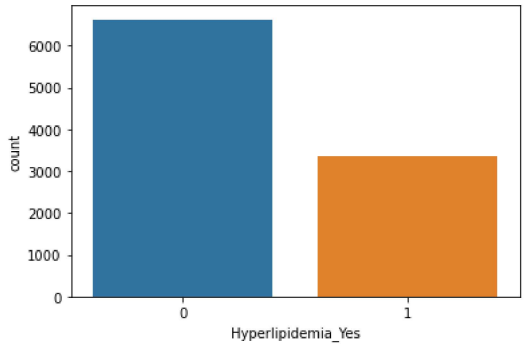
In [57]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='HighBlood_Yes', data=dfupdated)
```

Out[57]: <AxesSubplot:xlabel='HighBlood_Yes', ylabel='count'>



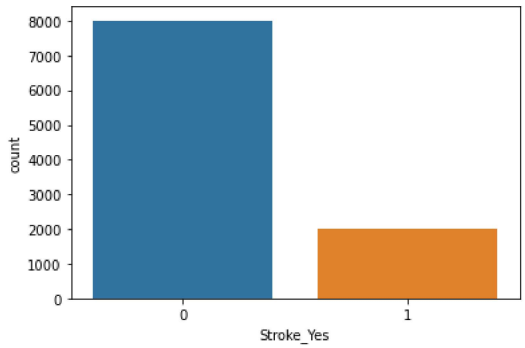
In [58]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Hyperlipidemia_Yes', data=dfupdated)
```

Out[58]: <AxesSubplot:xlabel='Hyperlipidemia_Yes', ylabel='count'>



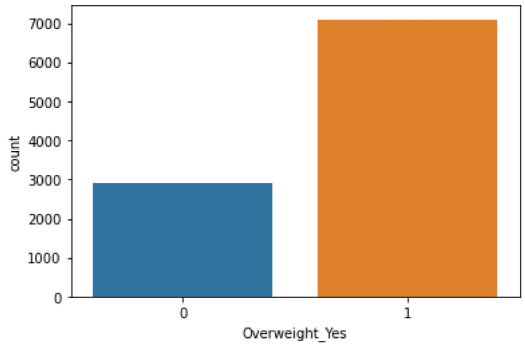
In [59]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Stroke_Yes', data=dfupdated)
```

Out[59]: <AxesSubplot:xlabel='Stroke_Yes', ylabel='count'>



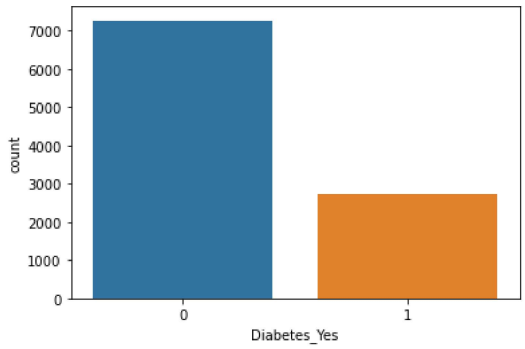
In [60]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Overweight_Yes', data=dfupdated)
```

Out[60]: <AxesSubplot:xlabel='Overweight_Yes', ylabel='count'>



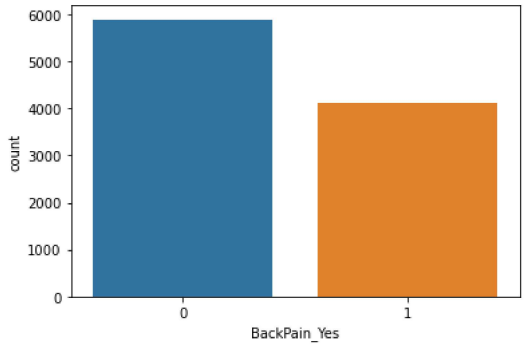
In [61]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Diabetes_Yes', data=dfupdated)
```

Out[61]: <AxesSubplot:xlabel='Diabetes_Yes', ylabel='count'>



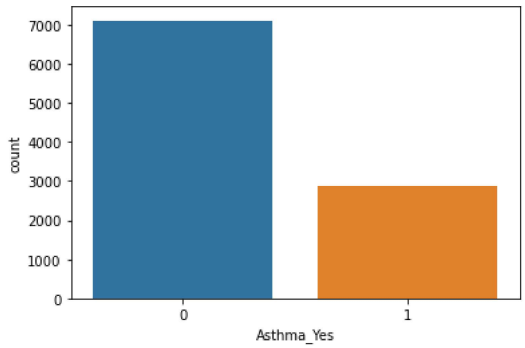
In [62]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='BackPain_Yes', data=dfupdated)
```

Out[62]: <AxesSubplot:xlabel='BackPain_Yes', ylabel='count'>



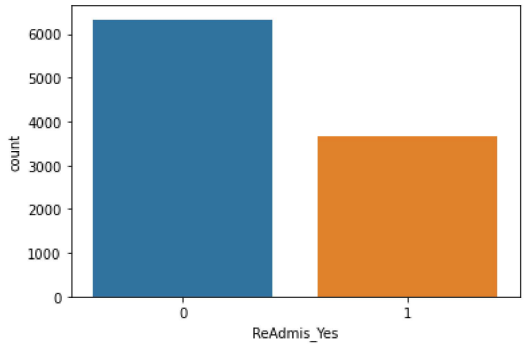
In [63]:
```
# Check the Prevalence of Each Classification Category with visual plot
sns.countplot(x='Asthma_Yes', data=dfupdated)
```

Out[63]: <AxesSubplot:xlabel='Asthma_Yes', ylabel='count'>



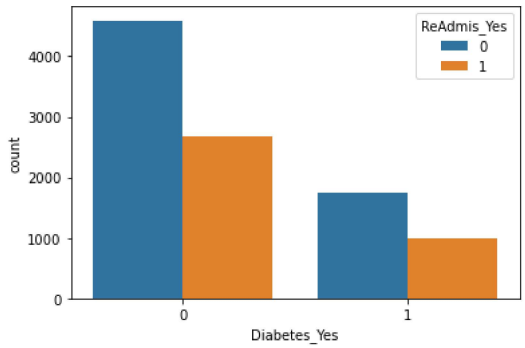
In [64]:
```
# Predictor Variable Plot
sns.countplot(x='ReAdmis_Yes', data=dfupdated)
```

Out[64]: <AxesSubplot:xlabel='ReAdmis_Yes', ylabel='count'>



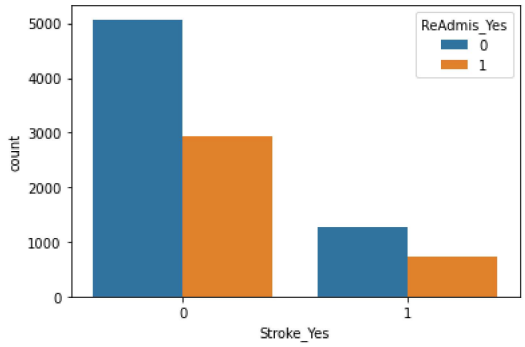
In [65]:
```
#Compare readmissions to patients who have diabetes
sns.countplot(x='Diabetes_Yes', hue='ReAdmis_Yes', data=dfupdated)
```

Out[65]: <AxesSubplot:xlabel='Diabetes_Yes', ylabel='count'>



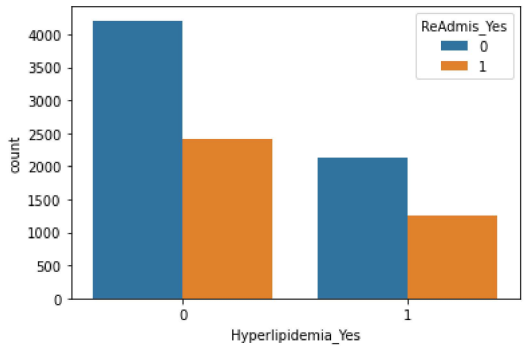
In [66]:
```
#Compare readmissions to patients who have had a stroke
sns.countplot(x='Stroke_Yes', hue='ReAdmis_Yes', data=dfupdated)
```

Out[66]: <AxesSubplot:xlabel='Stroke_Yes', ylabel='count'>



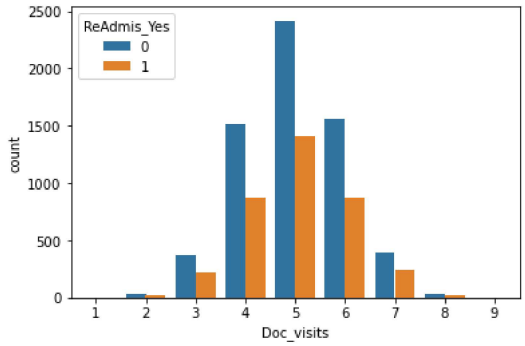
In [67]:
```
#Compare readmissions to patients who have hyperlipidemia
sns.countplot(x='Hyperlipidemia_Yes', hue='ReAdmis_Yes', data=dfupdated)
```

Out[67]: <AxesSubplot:xlabel='Hyperlipidemia_Yes', ylabel='count'>



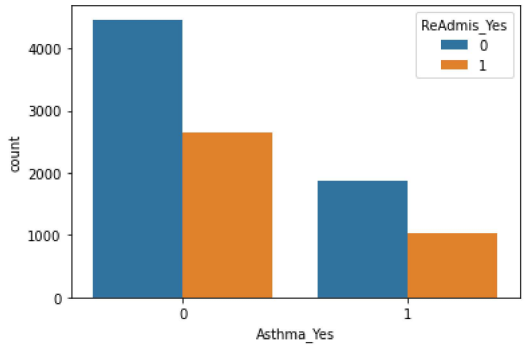
In [68]:
```
#Compare readmissions to doctor visits
sns.countplot(x='Doc_visits', hue='ReAdmis_Yes', data=dfupdated)
```

Out[68]: <AxesSubplot:xlabel='Doc_visits', ylabel='count'>



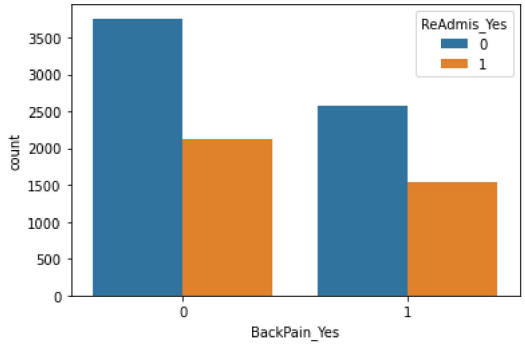
In [69]:
```
#Compare readmissions patients with asthma
sns.countplot(x='Asthma_Yes', hue='ReAdmis_Yes', data=dfupdated)
```

Out[69]: <AxesSubplot:xlabel='Asthma_Yes', ylabel='count'>



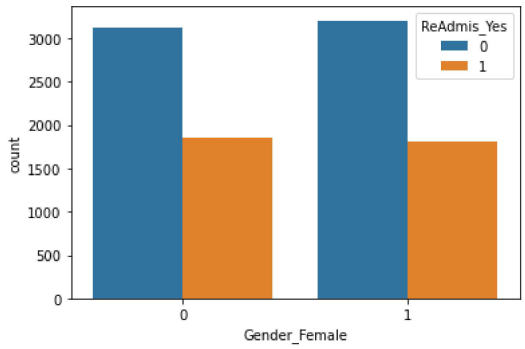
In [70]:
```
#Compare readmissions patients with back pain
sns.countplot(x='BackPain_Yes', hue='ReAdmis_Yes', data=dfupdated)
```

Out[70]: <AxesSubplot:xlabel='BackPain_Yes', ylabel='count'>



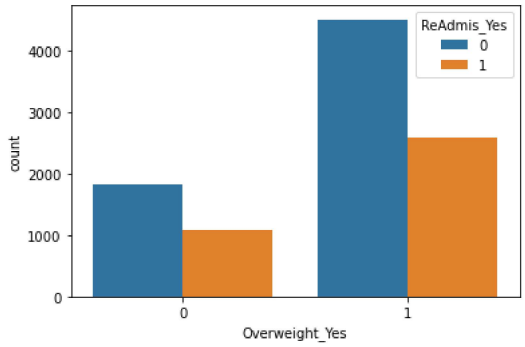
In [71]:
```
#Compare readmissions for female patients
sns.countplot(x='Gender_Female', hue='ReAdmis_Yes', data=dfupdated)
```

Out[71]: <AxesSubplot:xlabel='Gender_Female', ylabel='count'>



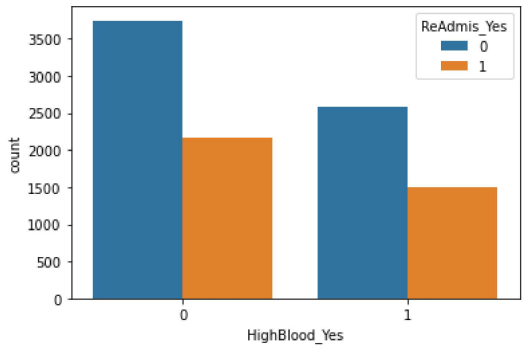
In [72]:
```
#Compare readmissions for overweight patients
sns.countplot(x='Overweight_Yes', hue='ReAdmis_Yes', data=dfupdated)
```

Out[72]: <AxesSubplot:xlabel='Overweight_Yes', ylabel='count'>

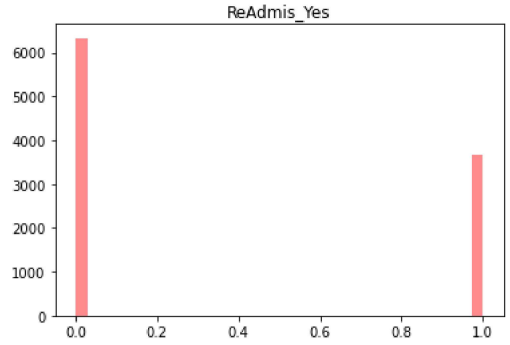


In [73]:
```
#Compare readmissions for patients with high blood pressure
sns.countplot(x='HighBlood_Yes', hue='ReAdmis_Yes', data=dfupdated)
```

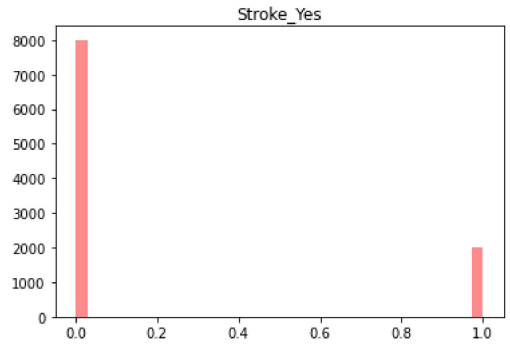
Out[73]: <AxesSubplot:xlabel='HighBlood_Yes', ylabel='count'>



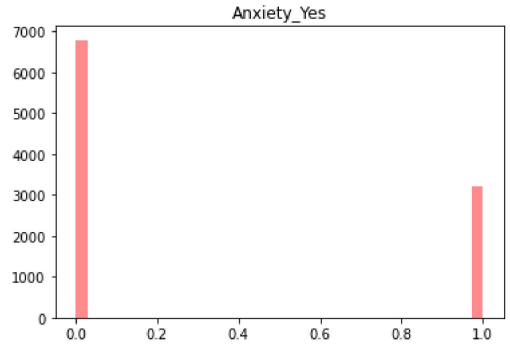
In [74]:
```
# plotting histogram
plt.hist(dfupdated['ReAdmis_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("ReAdmis_Yes")
plt.show()
```



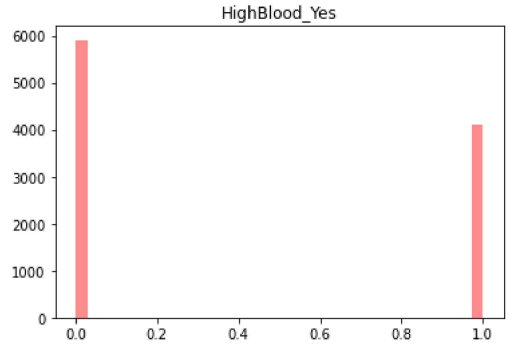
In [75]:
```
# plotting histogram
plt.hist(dfupdated['Stroke_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Stroke_Yes")
plt.show()
```



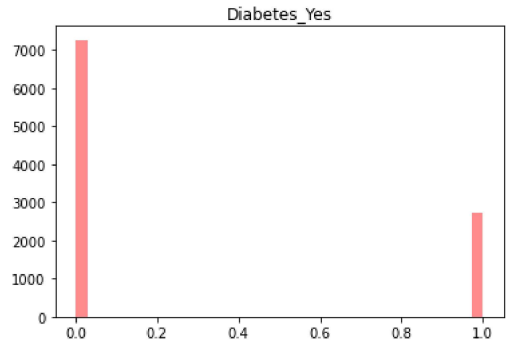
In [77]:
```
# plotting histogram
plt.hist(dfupdated['Anxiety_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Anxiety_Yes")
plt.show()
```



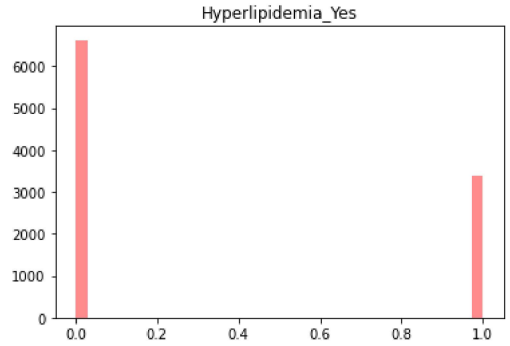
In [80]:
```
# plotting histogram
plt.hist(dfupdated['HighBlood_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("HighBlood_Yes")
plt.show()
```



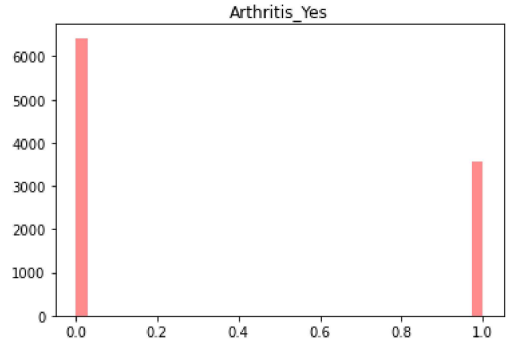
In [81]:
```
# plotting histogram
plt.hist(dfupdated['Diabetes_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Diabetes_Yes")
plt.show()
```



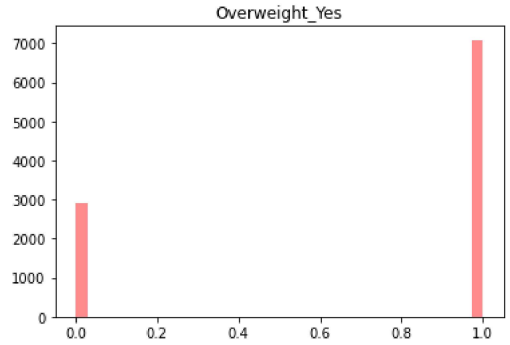
In [83]:
```
# plotting histogram
plt.hist(dfupdated['Hyperlipidemia_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Hyperlipidemia_Yes")
plt.show()
```



In [84]:
```
# plotting histogram
plt.hist(dfupdated['Arthritis_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Arthritis_Yes")
plt.show()
```

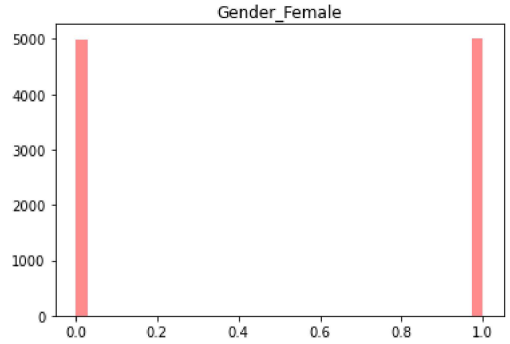


In [85]:
```
# plotting histogram
plt.hist(dfupdated['Overweight_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Overweight_Yes")
plt.show()
```

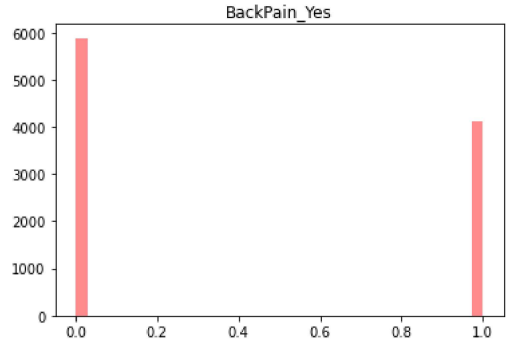


In [86]:
```
# plotting histogram
plt.hist(dfupdated['Gender_Female'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Gender_Female")
```

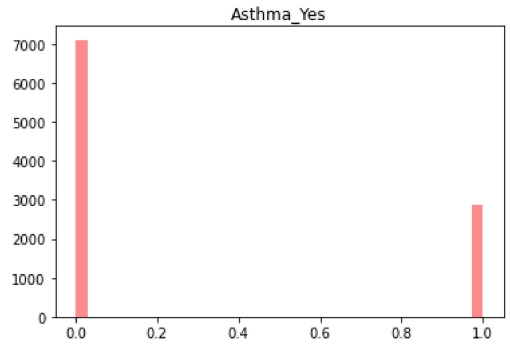
Out[86]:
```
Text(0.5, 1.0, 'Gender_Female')
```



In [87]:
```
# plotting histogram
plt.hist(dfupdated['BackPain_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("BackPain_Yes")
plt.show()
```

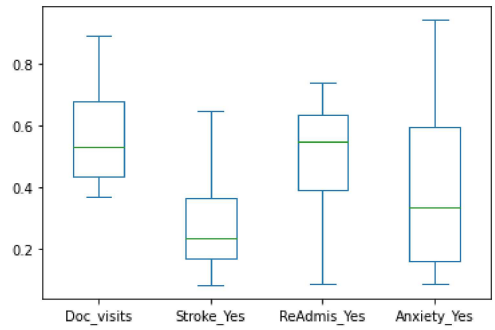


In [88]:
```
# plotting histogram
plt.hist(dfupdated['Asthma_Yes'],bins = 35,
         alpha = 0.45, color = 'red')
plt.title("Asthma_Yes")
plt.show()
```



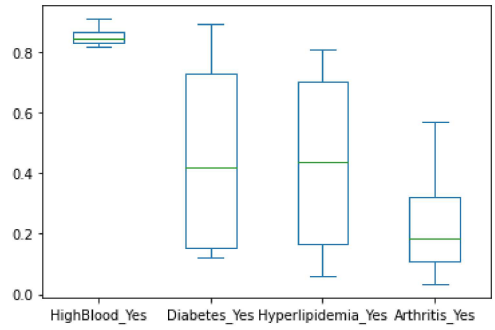
In [89]:
```
#boxplots
df = pd.DataFrame(data = np.random.random(size=(4,4)), columns = ['Doc_visits', 'Stroke_Yes', 'ReAdmis_Yes', 'Anxiety_Yes'])

df.plot(kind='box')
plt.show()
```



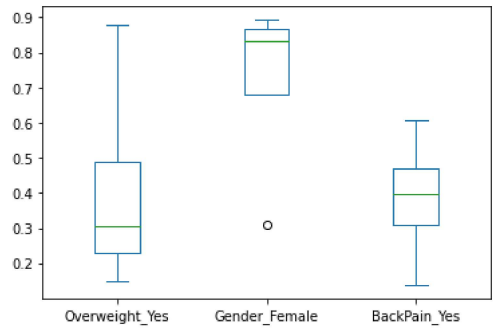
In [90]:
```
df = pd.DataFrame(data = np.random.random(size=(4,4)), columns = ['HighBlood_Yes', 'Diabetes_Yes', 'Hyperlipidemia_Yes', 'Arthrit

df.plot(kind='box')
plt.show()
```



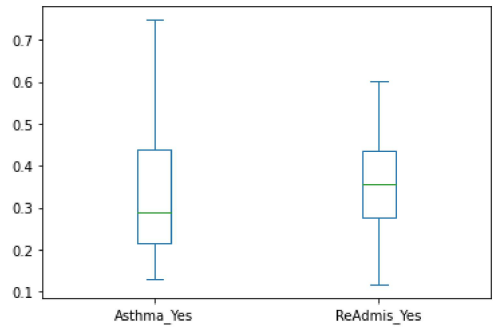
In [91]:
```
df = pd.DataFrame(data = np.random.random(size=(4,3)), columns = ['Overweight_Yes', 'Gender_Female', 'BackPain_Yes'])

df.plot(kind='box')
plt.show()
```

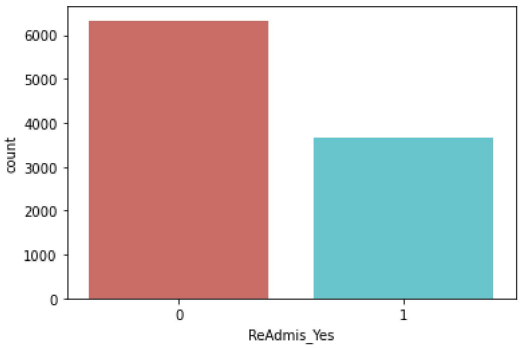


In [92]:
```
df = pd.DataFrame(data = np.random.random(size=(4,2)), columns = ['Asthma_Yes','ReAdmis_Yes'])

df.plot(kind='box')
plt.show()
```



In [93]:
```
#how does the Y data stack up?
dfupdated['ReAdmis_Yes'].value_counts()
sns.countplot(x='ReAdmis_Yes', data=dfupdated, palette='hls')
plt.show()
plt.savefig('count_plot')
```




```
<Figure size 432x288 with 0 Axes>
```

In [94]:
```
#review percentage of patients who readmitted/versus not readmitted
count_no_sub = len(dfupdated[dfupdated['ReAdmis_Yes']==0])
count_sub = len(dfupdated[dfupdated['ReAdmis_Yes']==1])
pct_of_no_sub = count_no_sub/(count_no_sub+count_sub)
print("patients who did not readmit", pct_of_no_sub*100)
pct_of_sub = count_sub/(count_no_sub+count_sub)
print("patients who readmitted", pct_of_sub*100)
```

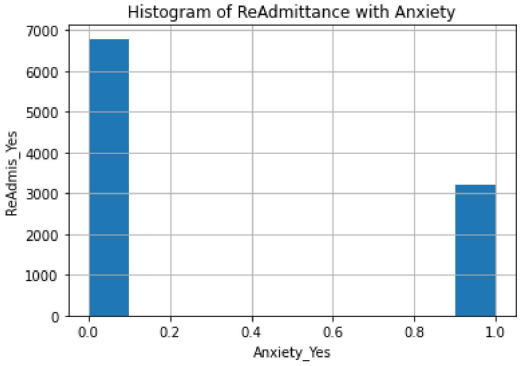
```
patients who did not readmit 63.31
patients who readmitted 36.69
```

In [95]:
```
#Check the means
dfupdated.groupby('ReAdmis_Yes').mean()
```

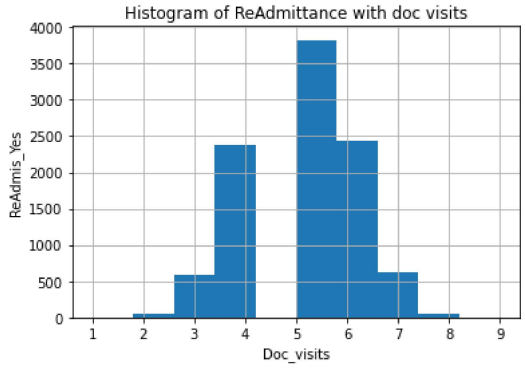
Out[95]:

ReAdmis_Yes	Doc_visits	Anxiety_Yes	Overweight_Yes	Arthritis_Yes	HighBlood_Yes	Gender_Female	Stroke_Yes	Diabetes_Yes	Asthma_Yes	Hyperlipidemia_
0	5.012004	0.320644	0.712368	0.354604	0.408150	0.506239	0.199021	0.274838	0.295214	0.335
1	5.012537	0.322976	0.704279	0.362224	0.410466	0.494140	0.199782	0.272009	0.279095	0.339

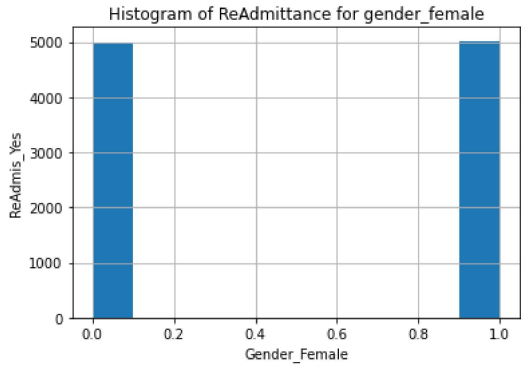
In [96]:
```
#review histogram visual aides
dfupdated.Anxiety_Yes.hist()
plt.title('Histogram of ReAdmittance with Anxiety')
plt.xlabel('Anxiety_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_anxiety')
```



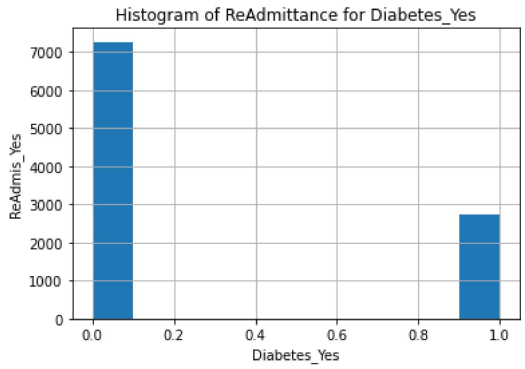
In [97]:
```
dfupdated.Doc_visits.hist()
plt.title('Histogram of ReAdmittance with doc visits')
plt.xlabel('Doc_visits')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_docvisits')
```



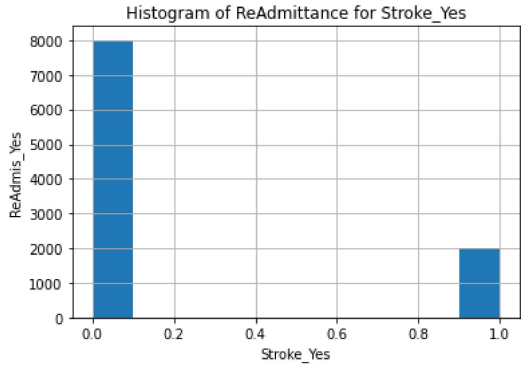
In [98]:
```
dfupdated.Gender_Female.hist()
plt.title('Histogram of ReAdmittance for gender_female')
plt.xlabel('Gender_Female')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_GenderFemale')
```



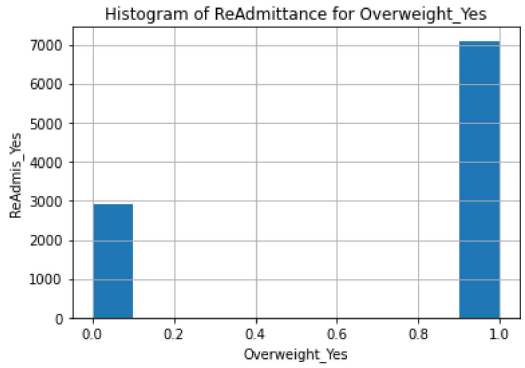
In [99]:
```
dfupdated.Diabetes_Yes.hist()
plt.title('Histogram of ReAdmittance for Diabetes_Yes')
plt.xlabel('Diabetes_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_DiabetesYes')
```



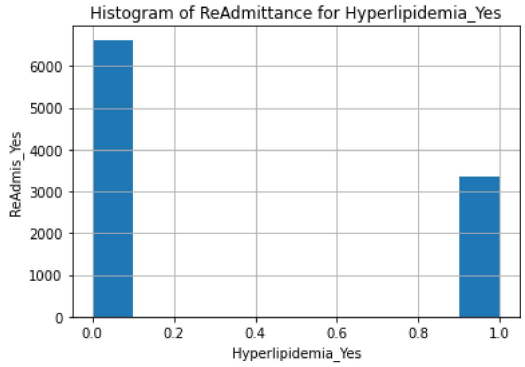
In [100]:
```
dfupdated.Stroke_Yes.hist()
plt.title('Histogram of ReAdmittance for Stroke_Yes')
plt.xlabel('Stroke_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_StrokeYes')
```



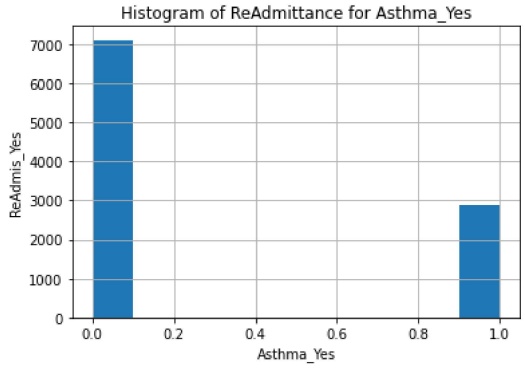
In [101]:
```
dfupdated.Overweight_Yes.hist()
plt.title('Histogram of ReAdmittance for Overweight_Yes')
plt.xlabel('Overweight_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_OverweightYes')
```



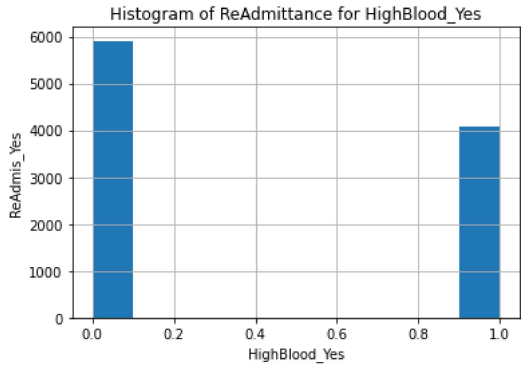
In [102]:
```
dfupdated.Hyperlipidemia_Yes.hist()
plt.title('Histogram of ReAdmittance for Hyperlipidemia_Yes')
plt.xlabel('Hyperlipidemia_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_HyperlipidemiaYes')
```



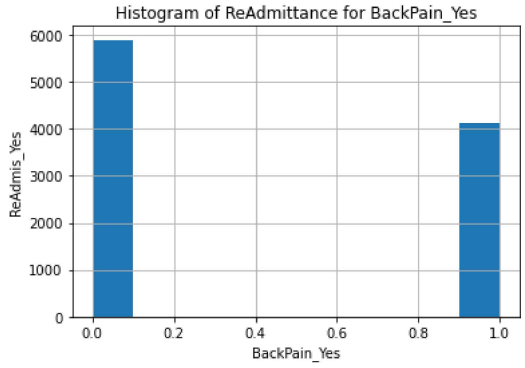
In [103]:
```
dfupdated.Asthma_Yes.hist()
plt.title('Histogram of ReAdmittance for Asthma_Yes')
plt.xlabel('Asthma_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_AsthmaYes')
```



In [104]:
```
dfupdated.HighBlood_Yes.hist()
plt.title('Histogram of ReAdmittance for HighBlood_Yes')
plt.xlabel('HighBlood_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_HighBloodYes')
```



In [105]:
```
dfupdated.BackPain_Yes.hist()
plt.title('Histogram of ReAdmittance for BackPain_Yes')
plt.xlabel('BackPain_Yes')
plt.ylabel('ReAdmis_Yes')
plt.savefig('hist_BackPainYes')
```



In [106]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + Stroke_Yes + HighBlood_Yes + Anxiety_Yes + Diabetes_Yes + Hyperlipidemia_Yes
```

```
Optimization terminated successfully.
         Current function value: 0.656892
         Iterations 4
```

In [107]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9988
Method:                           MLE   Df Model:                          11
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005985
Time:                        20:20:54   Log-Likelihood:                -6568.9
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.7251
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
Intercept              -0.5197      0.116     -4.480      0.000      -0.747      -0.292
Overweight_Yes         -0.0389      0.046     -0.853      0.394      -0.128       0.051
Stroke_Yes              0.0061      0.052      0.118      0.906      -0.096       0.108
HighBlood_Yes           0.0101      0.042      0.240      0.811      -0.073       0.093
Anxiety_Yes             0.0107      0.044      0.241      0.810      -0.076       0.098
Diabetes_Yes           -0.0127      0.047     -0.273      0.785      -0.104       0.079
Hyperlipidemia_Yes      0.0179      0.044      0.407      0.684      -0.068       0.104
Arthritis_Yes           0.0335      0.043      0.774      0.439      -0.051       0.118
Gender_Female          -0.0486      0.042     -1.171      0.242      -0.130       0.033
BackPain_Yes            0.0583      0.042      1.384      0.167      -0.024       0.141
Asthma_Yes             -0.0790      0.046     -1.717      0.086      -0.169       0.011
Doc_visits              0.0003      0.020      0.013      0.990      -0.039       0.039
=======================================================================================
```

In [108]:
```
#define the predictor variables and the response variable
X = dfupdated[['Overweight_Yes', 'Stroke_Yes', 'HighBlood_Yes', 'Anxiety_Yes', 'Diabetes_Yes', 'Hyperlipidemia_Yes', 'Arthritis_Y
y = dfupdated['ReAdmis_Yes']
```

In [109]:
```
#split the dataset into training (70%) and testing (30%) sets
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

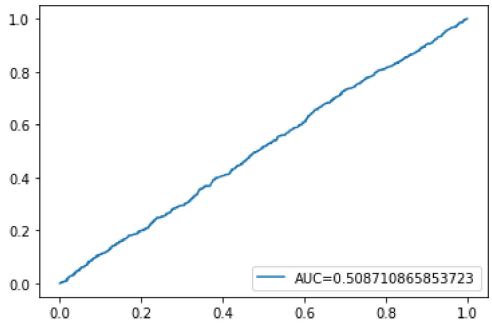
In [110]:
```
#instantiate the model
log_regression = LogisticRegression()
```

In [111]:
```
#fit the model using the training data
log_regression.fit(X_train,y_train)
```

Out[111]:
```
LogisticRegression
LogisticRegression()
```

In [112]:
```
#define metrics
y_pred_proba = log_regression.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
```

In [113]:
```
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.legend(loc=4)
plt.show()
```

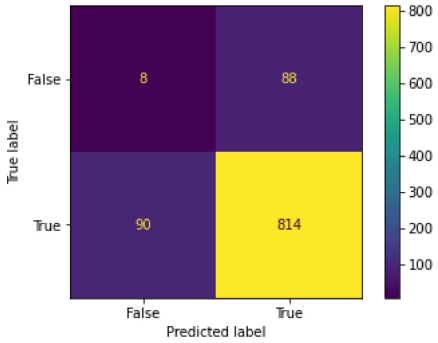


In [115]:
```
#plot confusion matrix
actual = np.random.binomial(1,.9,size = 1000)
predicted = np.random.binomial(1,.9,size = 1000)

confusion_matrix = metrics.confusion_matrix(actual, predicted)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```



In [116]:
```
#drop highest p-value
to_drop = ['Doc_visits']
dfupdated.drop(to_drop, inplace=True, axis=1)
```

In [117]:
```
#new dataframe for second model
dfupdated2 = dfupdated[['ReAdmis_Yes','Stroke_Yes', 'HighBlood_Yes', 'Anxiety_Yes', 'Hyperlipidemia_Yes', 'Arthritis_Yes', 'Overw
```

In [118]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + Stroke_Yes + HighBlood_Yes + Anxiety_Yes + Diabetes_Yes + Hyperlipidemia_Yes
```

```
Optimization terminated successfully.
         Current function value: 0.656892
         Iterations 4
```

In [119]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9989
Method:                           MLE   Df Model:                          10
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005985
Time:                        20:22:41   Log-Likelihood:                -6568.9
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.6418
=========================================================================================
                            coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Intercept                -0.5184      0.060     -8.646      0.000      -0.636      -0.401
Overweight_Yes           -0.0389      0.046     -0.853      0.394      -0.128       0.051
Stroke_Yes                0.0061      0.052      0.118      0.906      -0.096       0.108
HighBlood_Yes             0.0101      0.042      0.240      0.811      -0.073       0.093
Anxiety_Yes               0.0107      0.044      0.241      0.810      -0.076       0.098
Diabetes_Yes             -0.0127      0.047     -0.273      0.785      -0.104       0.079
Hyperlipidemia_Yes        0.0178      0.044      0.407      0.684      -0.068       0.104
Arthritis_Yes             0.0335      0.043      0.774      0.439      -0.051       0.118
Gender_Female            -0.0486      0.042     -1.171      0.242      -0.130       0.033
BackPain_Yes              0.0583      0.042      1.384      0.166      -0.024       0.141
Asthma_Yes               -0.0790      0.046     -1.718      0.086      -0.169       0.011
=========================================================================================
```

In [120]:
```
#new dataframe for third model
dfupdated3 = dfupdated[['ReAdmis_Yes', 'HighBlood_Yes', 'Anxiety_Yes', 'Hyperlipidemia_Yes', 'Arthritis_Yes', 'Overweight_Yes',
```

In [121]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + HighBlood_Yes + Anxiety_Yes + Diabetes_Yes + Hyperlipidemia_Yes + Arthritis_Y
```

```
Optimization terminated successfully.
         Current function value: 0.656892
         Iterations 4
```

In [122]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9990
Method:                           MLE   Df Model:                           9
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005974
Time:                        20:23:05   Log-Likelihood:                -6568.9
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.5489
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
Intercept              -0.5172      0.059     -8.767      0.000      -0.633      -0.402
Overweight_Yes         -0.0389      0.046     -0.853      0.394      -0.128       0.051
HighBlood_Yes           0.0102      0.042      0.241      0.810      -0.073       0.093
Anxiety_Yes             0.0106      0.044      0.239      0.811      -0.076       0.098
Diabetes_Yes           -0.0127      0.047     -0.272      0.786      -0.104       0.079
Hyperlipidemia_Yes      0.0178      0.044      0.405      0.686      -0.068       0.104
Arthritis_Yes           0.0334      0.043      0.772      0.440      -0.051       0.118
Gender_Female          -0.0486      0.042     -1.170      0.242      -0.130       0.033
BackPain_Yes            0.0583      0.042      1.384      0.166      -0.024       0.141
Asthma_Yes             -0.0790      0.046     -1.717      0.086      -0.169       0.011
=======================================================================================
```

In [123]:
```
#new dataframe for 4th model
Dfupdated4 = dfupdated[['ReAdmis_Yes', 'HighBlood_Yes', 'Hyperlipidemia_Yes', 'Arthritis_Yes', 'Overweight_Yes', 'Gender_Female',
```

In [124]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + HighBlood_Yes + Diabetes_Yes + Hyperlipidemia_Yes + Arthritis_Yes + Gender_Fe
```

```
Optimization terminated successfully.
         Current function value: 0.656895
         Iterations 4
```

In [125]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9991
Method:                           MLE   Df Model:                           8
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005931
Time:                        20:23:28   Log-Likelihood:                -6569.0
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.4536
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
Intercept              -0.5138      0.057     -8.971      0.000      -0.626      -0.402
Overweight_Yes         -0.0391      0.046     -0.856      0.392      -0.129       0.050
HighBlood_Yes           0.0102      0.042      0.243      0.808      -0.073       0.093
Diabetes_Yes           -0.0127      0.047     -0.273      0.785      -0.104       0.079
Hyperlipidemia_Yes      0.0176      0.044      0.402      0.688      -0.068       0.104
Arthritis_Yes           0.0335      0.043      0.775      0.438      -0.051       0.118
Gender_Female          -0.0485      0.042     -1.169      0.242      -0.130       0.033
BackPain_Yes            0.0584      0.042      1.386      0.166      -0.024       0.141
Asthma_Yes             -0.0789      0.046     -1.715      0.086      -0.169       0.011
=======================================================================================
```

In [126]:
```
#new dataframe for 5th model
Dfupdated5 = dfupdated[['ReAdmis_Yes', 'Hyperlipidemia_Yes', 'Arthritis_Yes', 'Overweight_Yes', 'Gender_Female', 'BackPain_Yes',
```

In [127]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + Diabetes_Yes + Hyperlipidemia_Yes + Arthritis_Yes + Gender_Female + BackPain_
```

```
Optimization terminated successfully.
         Current function value: 0.656898
         Iterations 4
```

In [128]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9992
Method:                           MLE   Df Model:                           7
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005886
Time:                        20:23:50   Log-Likelihood:                -6569.0
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.3563
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            -0.5098      0.055     -9.301      0.000      -0.617      -0.402
Overweight_Yes       -0.0388      0.046     -0.850      0.396      -0.128       0.051
Diabetes_Yes         -0.0128      0.047     -0.274      0.784      -0.104       0.079
Hyperlipidemia_Yes    0.0175      0.044      0.399      0.690      -0.068       0.104
Arthritis_Yes         0.0336      0.043      0.776      0.437      -0.051       0.118
Gender_Female        -0.0486      0.042     -1.171      0.242      -0.130       0.033
BackPain_Yes          0.0585      0.042      1.387      0.165      -0.024       0.141
Asthma_Yes           -0.0788      0.046     -1.713      0.087      -0.169       0.011
=====================================================================================
```

In [129]:
```
#new dataframe for 6th model
Dfupdated6 = dfupdated[['ReAdmis_Yes', 'Hyperlipidemia_Yes', 'Arthritis_Yes', 'Overweight_Yes', 'Gender_Female', 'BackPain_Yes',
```

In [130]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + Hyperlipidemia_Yes + Arthritis_Yes + Gender_Female + BackPain_Yes + Asthma_Ye
```

```
Optimization terminated successfully.
         Current function value: 0.656902
         Iterations 4
```

In [131]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9993
Method:                           MLE   Df Model:                           6
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005829
Time:                        20:24:13   Log-Likelihood:                -6569.0
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.2639
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            -0.5132      0.053     -9.622      0.000      -0.618      -0.409
Overweight_Yes       -0.0387      0.046     -0.848      0.397      -0.128       0.051
Hyperlipidemia_Yes    0.0174      0.044      0.396      0.692      -0.069       0.103
Arthritis_Yes         0.0335      0.043      0.774      0.439      -0.051       0.118
Gender_Female        -0.0487      0.042     -1.172      0.241      -0.130       0.033
BackPain_Yes          0.0586      0.042      1.391      0.164      -0.024       0.141
Asthma_Yes           -0.0790      0.046     -1.718      0.086      -0.169       0.011
=====================================================================================
```

In [132]:
```
#new dataframe for 7th model
Dfupdated7 = dfupdated[['ReAdmis_Yes', 'Arthritis_Yes', 'Overweight_Yes', 'Gender_Female', 'BackPain_Yes', 'Asthma_Yes']]
```

In [133]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + Arthritis_Yes + Gender_Female + BackPain_Yes + Asthma_Yes", data=Dfupdated7).
```

```
Optimization terminated successfully.
         Current function value: 0.656910
         Iterations 4
```

In [134]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9994
Method:                           MLE   Df Model:                           5
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005710
Time:                        20:24:34   Log-Likelihood:                -6569.1
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.1857
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      -0.5070      0.051     -9.943      0.000      -0.607      -0.407
Overweight_Yes -0.0388      0.046     -0.850      0.395      -0.128       0.051
Arthritis_Yes   0.0334      0.043      0.771      0.441      -0.051       0.118
Gender_Female  -0.0490      0.042     -1.179      0.238      -0.130       0.032
BackPain_Yes    0.0586      0.042      1.391      0.164      -0.024       0.141
Asthma_Yes     -0.0792      0.046     -1.722      0.085      -0.169       0.011
==============================================================================
```

In [135]:
```
#new dataframe for 8th model
Dfupdated8 = dfupdated[['ReAdmis_Yes', 'Overweight_Yes', 'Gender_Female', 'BackPain_Yes', 'Asthma_Yes']]
```

In [136]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Overweight_Yes + Gender_Female + BackPain_Yes + Asthma_Yes", data=Dfupdated8).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.656940
         Iterations 4
```

In [137]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9995
Method:                           MLE   Df Model:                           4
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0005258
Time:                        20:24:55   Log-Likelihood:                -6569.4
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.1406
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      -0.4947      0.048    -10.221      0.000      -0.590      -0.400
Overweight_Yes -0.0386      0.046     -0.847      0.397      -0.128       0.051
Gender_Female  -0.0493      0.042     -1.188      0.235      -0.131       0.032
BackPain_Yes    0.0580      0.042      1.376      0.169      -0.025       0.141
Asthma_Yes     -0.0794      0.046     -1.727      0.084      -0.170       0.011
==============================================================================
```

In [138]:
```
#new dataframe for 9th model
Dfupdated9 = dfupdated[['ReAdmis_Yes', 'Gender_Female', 'BackPain_Yes', 'Asthma_Yes']]
```

In [139]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Gender_Female + BackPain_Yes + Asthma_Yes", data=Dfupdated9).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.656975
         Iterations 4
```

In [140]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9996
Method:                           MLE   Df Model:                            3
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:               0.0004713
Time:                        20:25:15   Log-Likelihood:                -6569.8
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.1025
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept         -0.5217      0.036    -14.331      0.000      -0.593      -0.450
Gender_Female     -0.0494      0.042     -1.189      0.234      -0.131       0.032
BackPain_Yes       0.0576      0.042      1.368      0.171      -0.025       0.140
Asthma_Yes        -0.0800      0.046     -1.739      0.082      -0.170       0.010
==================================================================================
```

In [141]:
```
#new dataframe for 10th model
Dfupdated10 = dfupdated[['ReAdmis_Yes', 'BackPain_Yes', 'Asthma_Yes']]
```

In [142]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ BackPain_Yes + Asthma_Yes", data=Dfupdated10).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.657046
         Iterations 4
```

In [143]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                     9997
Method:                           MLE   Df Model:                            2
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:               0.0003637
Time:                        20:25:38   Log-Likelihood:                -6570.5
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                   0.09155
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept         -0.5463      0.030    -18.197      0.000      -0.605      -0.487
BackPain_Yes       0.0571      0.042      1.356      0.175      -0.025       0.140
Asthma_Yes        -0.0797      0.046     -1.732      0.083      -0.170       0.010
==================================================================================
```

In [144]:
```
#new dataframe for final model
DfupdatedFinal = dfupdated[['ReAdmis_Yes', 'Asthma_Yes']]
```

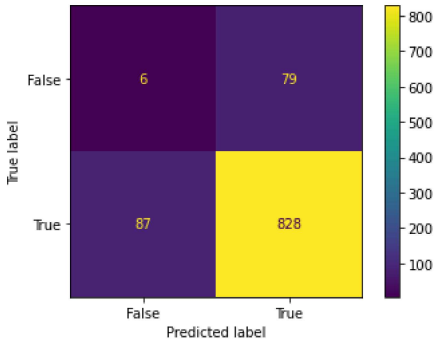
In [145]:
```
#Prepare and fit the model
log_reg = smf.logit("ReAdmis_Yes ~ Asthma_Yes", data=DfupdatedFinal).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.657138
         Iterations 4
```

In [146]:
```
#print the model
# Summary of results
print(log_reg.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:            ReAdmis_Yes   No. Observations:                10000
Model:                          Logit   Df Residuals:                    9998
Method:                           MLE   Df Model:                           1
Date:                Wed, 18 Jan 2023   Pseudo R-squ.:              0.0002240
Time:                        20:26:00   Log-Likelihood:                -6571.4
converged:                       True   LL-Null:                       -6572.9
Covariance Type:            nonrobust   LLR p-value:                    0.08618
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.5229      0.025    -21.310      0.000      -0.571      -0.475
Asthma_Yes    -0.0788      0.046     -1.713      0.087      -0.169       0.011
==============================================================================
```

In [147]:
```
#Run confusion matrix on final model
actual = np.random.binomial(1,.9,size = 1000)
predicted = np.random.binomial(1,.9,size = 1000)
confusion_matrix = metrics.confusion_matrix(actual, predicted)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
cm_display.plot()
plt.show()
```



In [ ]:
```
```