Nefisa Hassen D208 PA task1

Part I: Research Question

A1. What factors influence TotalCharge during a hospital stay for a patient?
A2. Goal of the Data Analysis:
The goal of this analysis is to understand the relationships between the dependent variable TotalCharge and independent variables. It is also to identify the factors that influence the total charges patients pay during their hospital stay. The analysis will provide insight into what affects healthcare costs to healthcare administrators.

Part II: Method Justification

B1. Summarize the four assumptions of a multiple linear regression model:
Homogeneity of variance or homoscedasticity: constant variance of the residuals.
Multicollinearity: two or more of the predictors correlate strongly with each other.
Normality: normal distribution of the error.
Linearity: the regression line should represent the points.
B2. Describe two benefits of using Python for various phases of the analysis: For this analysis, Python will be used. Python has many libraries and packages needed for analysis and offers greater visualization tools.
B3. Explain why multiple linear regression an appropriate technique is to use for analyzing the research question summarized in Part I:
The research question is investigating factors contributing to TotalCharge. We have one dependent variable, which is TotalCharge, and we are examining multiple independent variables that can potentially influence the total amount patients pay during their hospital stay. Independent variables include 'VitD_levels,' 'Doc_visits,' 'Full_meals_eaten,' 'vitD_supp,' 'Soft_drink,' 'Initial_admin,' 'HighBlood,' 'Stroke,' 'Complication_risk,' 'Overweight,' 'Arthritis,' 'Diabetes,' 'Hyperlipidemia,' 'BackPain,' 'Anxiety,' 'Allergic_rhinitis,' 'Reflux_esophagitis,' 'Asthma,' 'Services,' and 'Initial_days.'

Part III: Data Preparation

C1. On this analysis the file "medical_data" is used. After importing, the data was explored to identify and address any missing values in each variable. There were no missing values detected in any of the columns. Subsequently, the std() function was used to assess outliers. outliers were found in four variables: TotalCharge, Additional_charges, VitD_levels, and Initial_days. The zscore method was applied to treat outliers in all four columns. Also, categorical variables were transformed into numeric values through dummy variables. For categorical variables with more than two options, such as 'Services,' 'Complication_risk,' and 'Initial_admin,' one-hot encoding was implemented to generate numeric values. Lastly, the following variables are not needed to answer the research question and were removed. 'CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',

'Children', 'Age', 'Income', 'Marital', 'Gender','Item1', 'Item2', 'Item3', 'Item4',

'Item5', 'Item6', 'Item7', 'Item8'

In [1]:

```
1
import pandas as pd
2
import numpy as np
```

```
import matplotlib.pyplot as plt

import seaborn as sns

from statistics import stdev

%matplotlib inline

from scipy import stats

import statsmodels.api as sm
```

In [2]:
```
med_data = pd.read_csv('medical_clean.csv')
```

In [3]:

Out[3]:

```
med_data.shape
```

Out[5]:

```
(10000, 50)
```

In [6]:
```
med_data.columns
```

Out[6]:

```
Index(['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',
       'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',
       'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis',
       'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
```

```
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days',
       'TotalCharge', 'Additional_charges', 'Item1', 'Item2', 'Item3', 'Item4',
       'Item5', 'Item6', 'Item7', 'Item8'],
      dtype='object')
```

In [7]:

```
# dropping irevelvant columns.

med_data = med_data.drop(columns=['CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State',

    'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job',

    'Children', 'Age', 'Income', 'Marital', 'Gender','Item1', 'Item2', 'Item3', 'Item4',

    'Item5', 'Item6', 'Item7', 'Item8' ])
```

In [8]:

```
med_data.columns # checking to see of the columns are removed from the file.
```

Out[8]:

```
Index(['ReAdmis', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke',
       'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes',
       'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
       'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days',
       'TotalCharge', 'Additional_charges'],
      dtype='object')
```

In [9]:

```
med_data.isnull().sum() # cheking for missing data
```

Out[9]:

```
ReAdmis              0
VitD_levels          0
Doc_visits           0
Full_meals_eaten     0
vitD_supp            0
Soft_drink           0
Initial_admin        0
HighBlood            0
Stroke               0
Complication_risk    0
Overweight           0
Arthritis            0
Diabetes             0
Hyperlipidemia       0
BackPain             0
Anxiety              0
Allergic_rhinitis    0
Reflux_esophagitis   0
```

```
Asthma              0
Services            0
Initial_days        0
TotalCharge         0
Additional_charges  0
dtype: int64
```

In [10]:

```
1
```

Changing categorical variable into numeric value.

```
1
categorical_columns = [
2
  'ReAdmis', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
3
  'Soft_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis', 'Diabetes',
4
  'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
5
  'Reflux_esophagitis', 'Asthma'
6
]
7


8
for column in categorical_columns:
9
  med_data[column] = med_data[column].astype('category').cat.codes
```

In [12]:

```
1
med_data.dtypes
```

Out[12]:

```
ReAdmis              int8
VitD_levels         int16
Doc_visits           int8
Full_meals_eaten     int8
vitD_supp            int8
Soft_drink           int8
Initial_admin      object
HighBlood            int8
Stroke               int8
Complication_risk  object
Overweight           int8
Arthritis            int8
Diabetes             int8
Hyperlipidemia       int8
BackPain             int8
Anxiety              int8
Allergic_rhinitis    int8
Reflux_esophagitis   int8
```

```
Asthma                int8
Services             object
Initial_days        float64
TotalCharge         float64
Additional_charges  float64
dtype: object
```

Code for one hot encoding

```
1
```

```
med_data = pd.get_dummies(med_data, columns=['Services','Complication_risk','Initial_admin'],
drop_first=True)
```

In [14]:

```
1
```

```
med_data.dtypes
```

Out[14]:

```
ReAdmis                             int8
VitD_levels                        int16
Doc_visits                          int8
Full_meals_eaten                    int8
vitD_supp                           int8
Soft_drink                          int8
HighBlood                           int8
Stroke                              int8
Overweight                          int8
Arthritis                           int8
Diabetes                            int8
Hyperlipidemia                      int8
BackPain                            int8
Anxiety                             int8
Allergic_rhinitis                   int8
Reflux_esophagitis                  int8
Asthma                              int8
Initial_days                      float64
TotalCharge                         float64
Additional_charges                  float64
Services_CT Scan                     bool
Services_Intravenous                 bool
Services_MRI                         bool
Complication_risk_Low                bool
Complication_risk_Medium             bool
Initial_admin_Emergency Admission    bool
Initial_admin_Observation Admission  bool

dtype: object
```

#changing the bool into intiger

```
1
```

```
med_data = med_data.astype(int)
```

In [17]:

```
1
```

```
med_data.dtypes #  now  all the columns are integers
```

Out[17]:

```
ReAdmis                                int32
VitD_levels                            int32
Doc_visits                             int32
Full_meals_eaten                       int32
vitD_supp                              int32
Soft_drink                             int32
HighBlood                              int32
Stroke                                 int32
Overweight                             int32
Arthritis                              int32
Diabetes                               int32
Hyperlipidemia                         int32
BackPain                               int32
Anxiety                                int32
Allergic_rhinitis                      int32
Reflux_esophagitis                     int32
Asthma                                 int32
Initial_days                           int32
TotalCharge                            int32
Additional_charges                     int32
Services_CT Scan                       int32
Services_Intravenous                   int32
Services_MRI                           int32
Complication_risk_Low                  int32
Complication_risk_Medium               int32
Initial_admin_Emergency Admission      int32
Initial_admin_Observation Admission    int32
dtype: object
```

In [18]:

```
1
```

```
# checking for outliers.
```

```
2
```

```
med_data.std()
```

Out[18]:

```
ReAdmis                    0.481983
VitD_levels             2878.750357
Doc_visits                 1.045734
Full_meals_eaten           1.008117
vitD_supp                  0.628505
Soft_drink                 0.437279
HighBlood                  0.491674
Stroke                     0.399494
Overweight                 0.454062
Arthritis                  0.479258
Diabetes                   0.445930
Hyperlipidemia             0.472777
```

```
BackPain                              0.492112
Anxiety                              0.467076
Allergic_rhinitis                      0.488681
Reflux_esophagitis                       0.492486
Asthma                               0.453460
Initial_days                         26.301628
TotalCharge                          2180.391406
Additional_charges                      6542.600277
Services_CT Scan                         0.327879
Services_Intravenous                      0.463738
Services_MRI                         0.191206
Complication_risk_Low                       0.409097
Complication_risk_Medium                  0.497687
Initial_admin_Emergency Admission         0.499989
Initial_admin_Observation Admission       0.429276
dtype: float64
```

In [19]:

```
1
```

 # as showen above the columuns inital_days. TotalCharge,AddtionalCharge have outliners and theses columnes are treated using the Zscare method as below.

In [20]:

```
1
med_data ['VitD_levels_z']=stats.zscore(med_data['VitD_levels'])
```

In [21]:

```
1
med_data_outliers_VitD_levels = med_data.query('VitD_levels_z > 3 | VitD_levels_z< -3')
```

In [22]:

```
1
med_data ['Initial_days_z']=stats.zscore(med_data['Initial_days'])
```

In [23]:

```
1
med_data_outliers_Initial_days = med_data.query('Initial_days_z > 3 | Initial_days_z< -3')
```

In [24]:

```
1
med_data ['TotalCharge_z']=stats.zscore(med_data['TotalCharge'])
```

In [25]:

```
1
med_data_outliers_TotalCharg = med_data.query('TotalCharge_z > 3 | TotalCharge_z< -3')
```

In [26]:

```
1
med_data['Additional_charges_z'] = stats.zscore(med_data['Additional_charges']) # z varaible
```

In [27]:

```
1
med_data_outliers_addCharg = med_data.query('Additional_charges_z > 3 | Additional_charges_z < -3') # new dataset created
```

In [28]:

```
1
med_data.std() # no more outliners
```

Out[28]:

```
ReAdmis                                0.481983
VitD_levels                         2878.750357
Doc_visits                             1.045734
Full_meals_eaten                       1.008117
vitD_supp                              0.628505
Soft_drink                             0.437279
HighBlood                              0.491674
Stroke                                 0.399494
Overweight                             0.454062
Arthritis                              0.479258
Diabetes                               0.445930
Hyperlipidemia                         0.472777
BackPain                               0.492112
Anxiety                                0.467076
Allergic_rhinitis                      0.488681
Reflux_esophagitis                     0.492486
Asthma                                 0.453460
Initial_days                          26.301628
TotalCharge                         2180.391406
Additional_charges                  6542.600277
Services_CT Scan                       0.327879
Services_Intravenous                   0.463738
Services_MRI                           0.191206
Complication_risk_Low                  0.409097
Complication_risk_Medium               0.497687
Initial_admin_Emergency Admission      0.499989
Initial_admin_Observation Admission    0.429276
VitD_levels_z                          1.000050
Initial_days_z                         1.000050
TotalCharge_z                          1.000050
Additional_charges_z                   1.000050
dtype: float64
```

C2. Describe the dependent variable and all independent variables using summary statistics

**The following columns have been excluded from the summary statistics as they are unnecessary for addressing the research question and have been removed in a previous step. I do have a summary statistics code that was executed prior to their deletion. ('CaseOrder', 'Customer_id', 'Interaction', 'UID', 'City', 'State', 'County', 'Zip', 'Lat', 'Lng', 'Population', 'Area', 'TimeZone', 'Job', 'Children', 'Age', 'Income', 'Marital', 'Gender','Item1', 'Item2', 'Item3', 'Item4', 'Item5', 'Item6', 'Item7', 'Item8')

The data includes 50 variables and 10,000 rows, with TotalCharge as the dependent variable.
The independent variables used for the analysis are 'ReAdmis', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'Initial_admin', 'HighBlood', 'Stroke', 'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Services', 'Initial_days', 'TotalCharge', 'Additional_charges'.

The average patient is a 53-year-old with 2 children and has approximately 40,000 yearly income. The average doctor visit is 5 times a year. The average total cost is around 5312 and the average additional charge is 12934 per patient. The minimum age of a patient is 18 years old and the maximum is 89 years old.The minimum income 154 *per year and the maximum inco*me is 207,249. The minimum doctor visit is once a year and the maximum 9 times a year. looking at the total charge, on average a patient pays $5312 *and a maximu*m is $9180.

In [29]:
```
#Describtibve ananlysis
```

In [30]:
```
med_data.columns #for the following columns, more summary statistics  will be explored below
```

Out[30]:
```
Index(['ReAdmis', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp',
       'Soft_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis',
       'Diabetes', 'Hyperlipidemia', 'BackPain', 'Anxiety',
       'Allergic_rhinitis', 'Reflux_esophagitis', 'Asthma', 'Initial_days',
       'TotalCharge', 'Additional_charges', 'Services_CT Scan',
       'Services_Intravenous', 'Services_MRI', 'Complication_risk_Low',
       'Complication_risk_Medium', 'Initial_admin_Emergency Admission',
       'Initial_admin_Observation Admission', 'VitD_levels_z',
       'Initial_days_z', 'TotalCharge_z', 'Additional_charges_z'],
      dtype='object')
```

In [31]:
```
med_data['Initial_admin_Observation Admission'].value_counts()
```

Out[31]:
```
Initial_admin_Observation Admission
0    7564
1    2436
Name: count, dtype: int64
```

In [32]:
```
med_data['Initial_admin_Emergency Admission'].value_counts()
```

Out[32]:
```
Initial_admin_Emergency Admission
1    5060
0    4940
Name: count, dtype: int64
```

In [33]:
```
med_data['Anxiety'].value_counts()
```

Out[33]:
```
Anxiety
0    6785
```

```
1    3215
Name: count, dtype: int64
```

In [34]:
```
med_data['Allergic_rhinitis'].value_counts()
```

Out[34]:
```
Allergic_rhinitis
0    6059
1    3941
Name: count, dtype: int64
```

In [35]:
```
med_data['Reflux_esophagitis'].value_counts()
```

Out[35]:
```
Reflux_esophagitis
0    5865
1    4135
Name: count, dtype: int64
```

In [ ]:
```

```

In [36]:
```
med_data['Services_CT Scan'].value_counts()
```

Out[36]:
```
Services_CT Scan
0    8775
1    1225
Name: count, dtype: int64
```

In [37]:
```
med_data['Services_Intravenous'].value_counts()
```

Out[37]:
```
Services_Intravenous
0    6870
1    3130
Name: count, dtype: int64
```

In [38]:
```
med_data['Services_MRI'].value_counts()
```

Out[38]:
```
Services_MRI
0    9620
1     380
Name: count, dtype: int64
```

In [39]:
```
med_data['Asthma'].value_counts()
```

Out[39]:

```
Asthma
0    7107
1    2893
Name: count, dtype: int64
```

In [40]:
```
med_data['Initial_days'].describe()
```

Out[40]:
```
count    10000.000000
mean        33.956000
std         26.301628
min          1.000000
25%          7.000000
50%         35.500000
75%         61.000000
max         71.000000
Name: Initial_days, dtype: float64
```

In [41]:
```
med_data['Soft_drink'].value_counts()
```

Out[41]:
```
Soft_drink
0    7425
1    2575
Name: count, dtype: int64
```

In [42]:
```
med_data['ReAdmis'].value_counts()
```

Out[42]:
```
ReAdmis
0    6331
1    3669
Name: count, dtype: int64
```

In [43]:
```
med_data['VitD_levels'].describe()
```

Out[43]:
```
count    10000.000000
mean      4985.993300
std       2878.750357
min          0.000000
25%       2493.750000
50%       4984.500000
75%       7478.250000
max       9975.000000
Name: VitD_levels, dtype: float64
```

In [44]:
```
med_data['Full_meals_eaten'].describe()
```

Out[44]:

```
count    10000.000000
mean         1.001400
std          1.008117
min          0.000000
25%          0.000000
50%          1.000000
75%          2.000000
max          7.000000
Name: Full_meals_eaten, dtype: float64
```

In [45]:

```
1
med_data['vitD_supp'].value_counts()
```

Out[45]:

```
vitD_supp
0    6702
1    2684
2     544
3      64
4       5
5       1
Name: count, dtype: int64
```

In [46]:

```
1
med_data['HighBlood'].value_counts()
```

Out[46]:

```
HighBlood
0    5910
1    4090
Name: count, dtype: int64
```

In [47]:

```
1
med_data['Stroke'].value_counts()
```

Out[47]:

```
Stroke
0    8007
1    1993
Name: count, dtype: int64
```

In [48]:

```
1
med_data['Complication_risk_Medium'].value_counts()
```

```
2
```

Out[48]:

```
Complication_risk_Medium
0    5483
1    4517
Name: count, dtype: int64
```

In [49]:

```
1
```

```
med_data['Complication_risk_Low'].value_counts()
```

Out[49]:

```
Complication_risk_Low
0    7875
1    2125
Name: count, dtype: int64
```

In [50]:

```
1
```

```
med_data['Overweight'].value_counts()
```

Out[50]:

```
Overweight
1    7094
0    2906
Name: count, dtype: int64
```

In [51]:

```
1
```

```
med_data['Arthritis'].value_counts()
```

Out[51]:

```
Arthritis
0    6426
1    3574
Name: count, dtype: int64
```

In [52]:

```
1
```

```
med_data['Diabetes'].value_counts()
```

Out[52]:

```
Diabetes
0    7262
1    2738
Name: count, dtype: int64
```

In [53]:

```
1
```

```
med_data['Hyperlipidemia'].value_counts()
```

Out[53]:

```
Hyperlipidemia
0    6628
1    3372
Name: count, dtype: int64
```

In [54]:

```
1
```

```
med_data['BackPain'].value_counts()
```

Out[54]:

```
BackPain
0    5886
1    4114
Name: count, dtype: int64
```

In [55]:

```
1
```

```
med_data['Additional_charges'].describe()
```

Out[55]:

```
count    10000.000000
mean     12934.032300
std       6542.600277
min       3125.000000
25%       7985.750000
50%      11573.500000
75%      15626.000000
max      30566.000000
Name: Additional_charges, dtype: float64
```

In [56]:

```
1
med_data['TotalCharge'].describe()
```

Out[56]:

```
count    10000.000000
mean      5311.673500
std       2180.391406
min       1938.000000
25%       3179.000000
50%       5213.500000
75%       7459.250000
max       9180.000000
Name: TotalCharge, dtype: float64
```

In [57]:

```
1
```

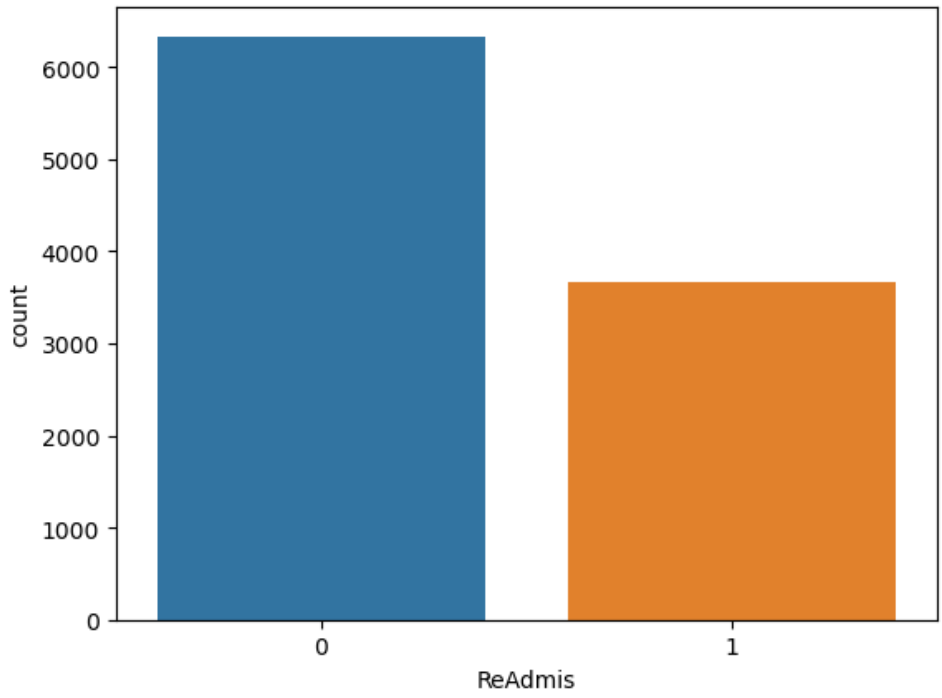
 # C3. Univariate and Bivariate visualizations

In [63]:

```
1
sns.countplot(x='ReAdmis', data =med_data)
```

Out[63]:

```
<Axes: xlabel='ReAdmis', ylabel='count'>
```

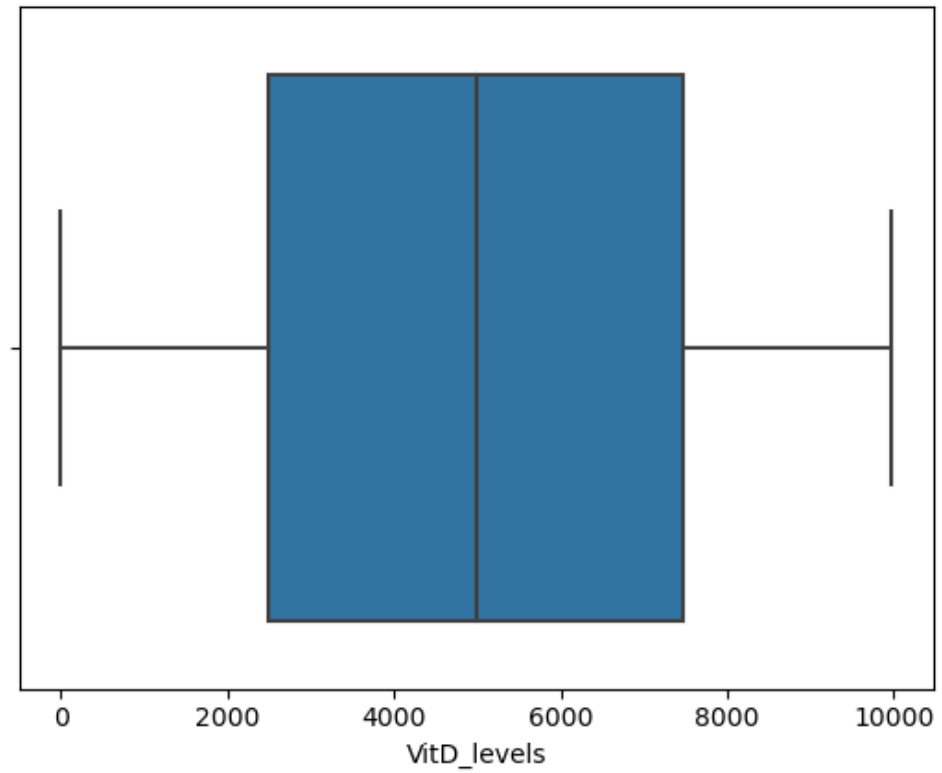


In [64]:

```
1 sns.boxplot(x='VitD_levels', data =med_data)
```

Out[64]:

```
<Axes: xlabel='VitD_levels'>
```

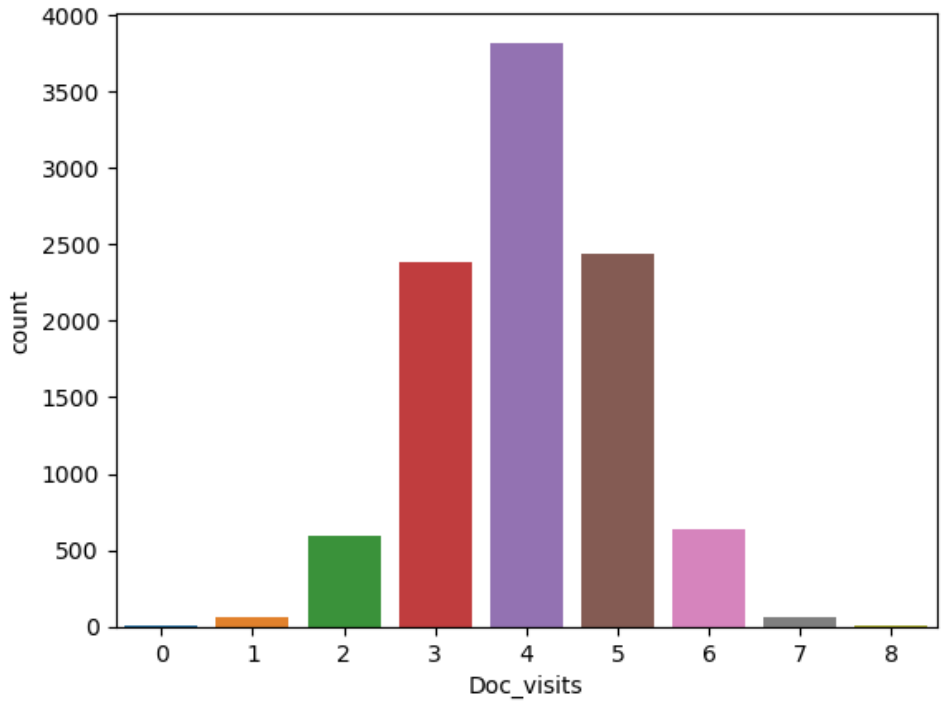


In [65]:

```
1
sns.countplot(x='Doc_visits', data =med_data)
```

Out[65]:

```
<Axes: xlabel='Doc_visits', ylabel='count'>
```

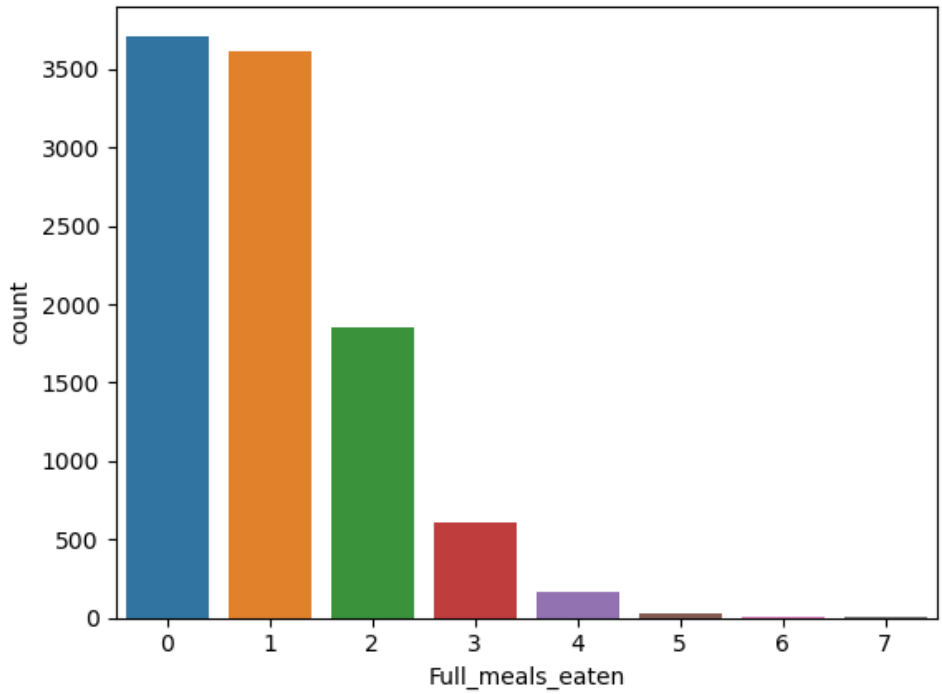


In [66]:

```
1
sns.countplot(x='Full_meals_eaten', data =med_data)
```

Out[66]:

```
<Axes: xlabel='Full_meals_eaten', ylabel='count'>
```

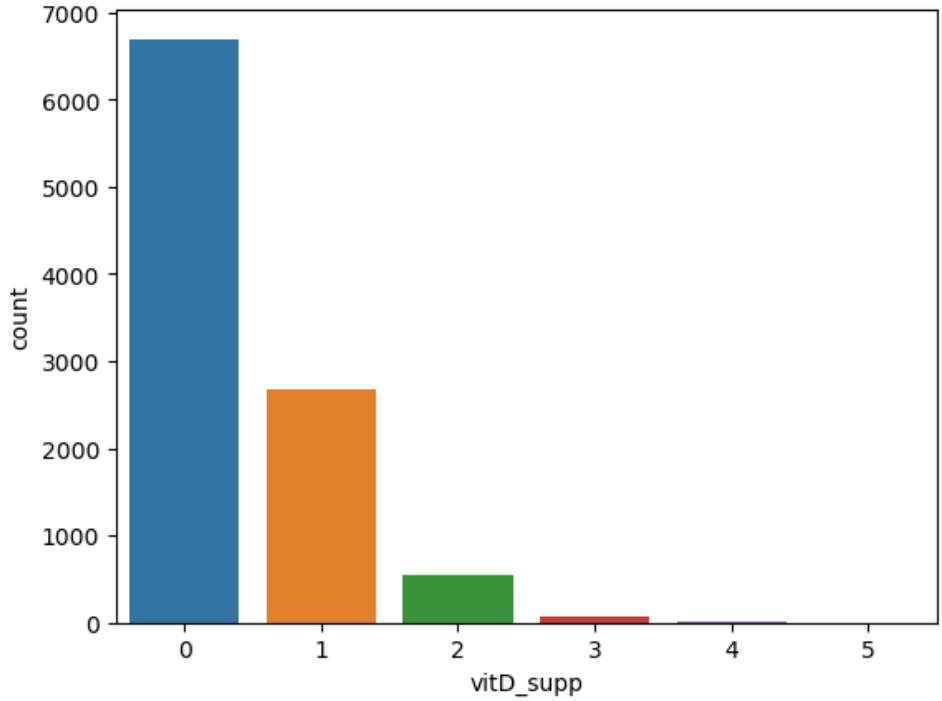


In [67]:

```
1
sns.countplot(x='vitD_supp', data =med_data)
```

Out[67]:

```
<Axes: xlabel='vitD_supp', ylabel='count'>
```

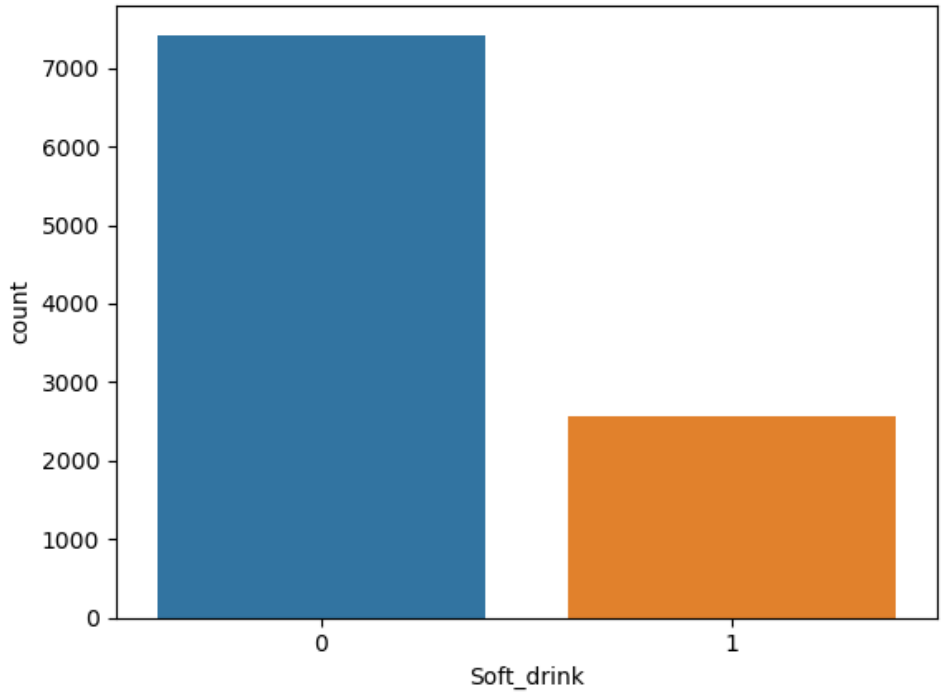


In [68]:

```
1
sns.countplot(x='Soft_drink', data =med_data)
```

Out[68]:

<Axes: xlabel='Soft_drink', ylabel='count'>

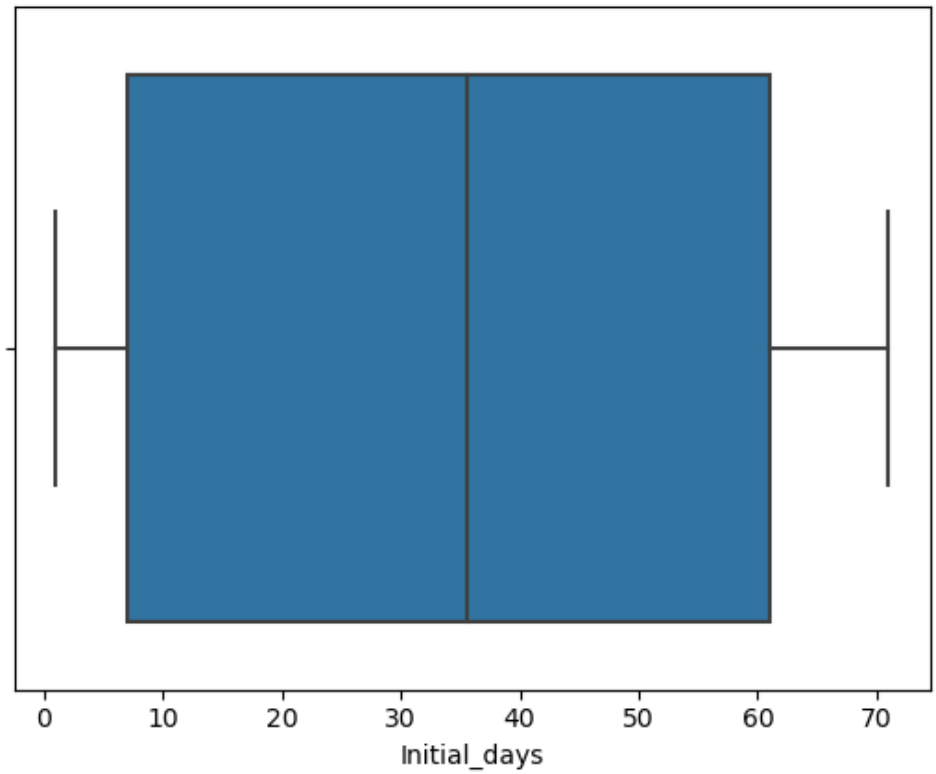


In [69]:

```
1
sns.boxplot(x='Initial_days', data =med_data)
```

Out[69]:

<Axes: xlabel='Initial_days'>



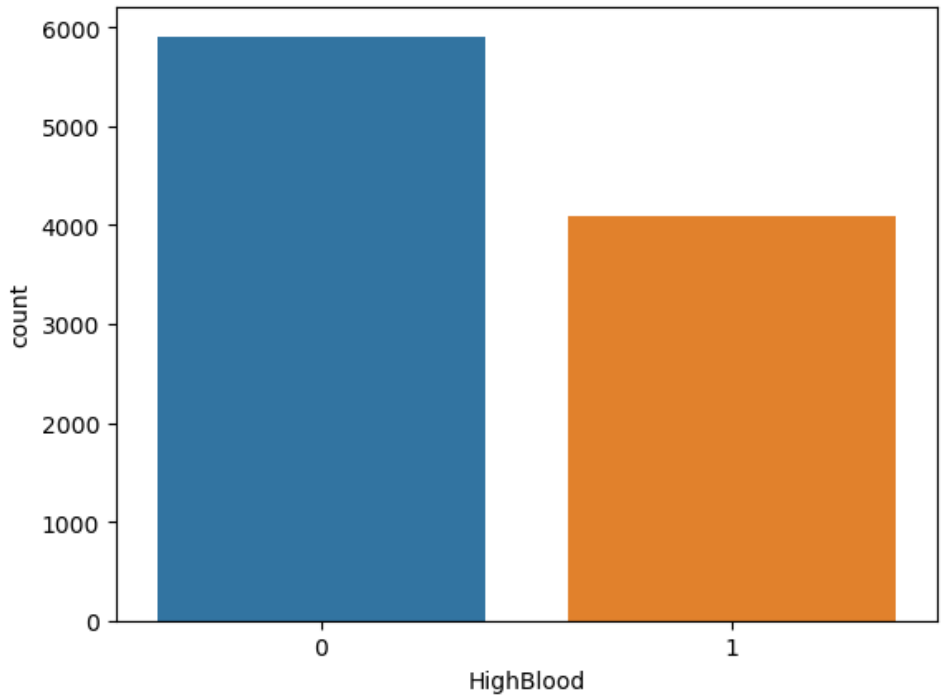
In [70]:

```
1
```

```
sns.countplot(x= 'HighBlood', data = med_data)
```

Out[70]:

```
<Axes: xlabel='HighBlood', ylabel='count'>
```

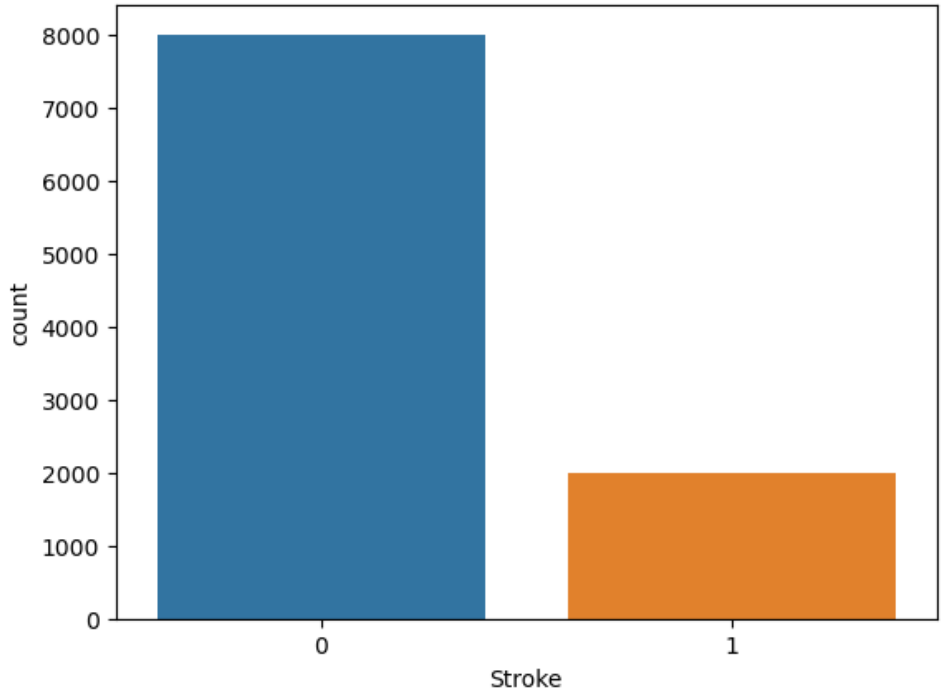


In [71]:

```
1
sns.countplot(x= 'Stroke', data = med_data)
```

Out[71]:

```
<Axes: xlabel='Stroke', ylabel='count'>
```

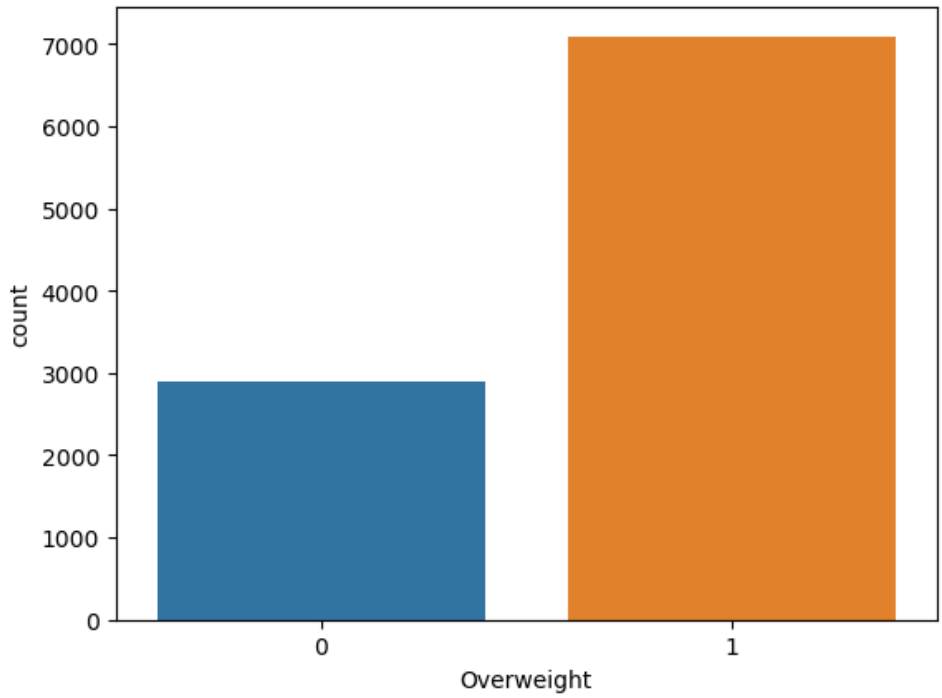


In [72]:

```
1
sns.countplot(x= 'Overweight', data = med_data)
```

Out[72]:

```
<Axes: xlabel='Overweight', ylabel='count'>
```



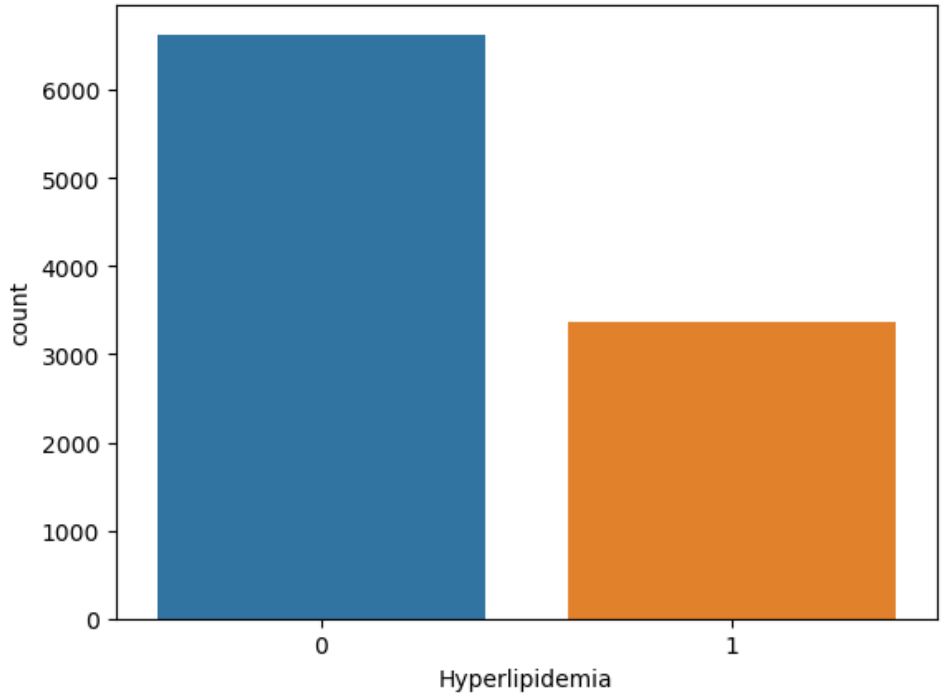
In [73]:

```
1
```

```
sns.countplot(x= 'Hyperlipidemia', data = med_data)
```

Out[73]:

```
<Axes: xlabel='Hyperlipidemia', ylabel='count'>
```



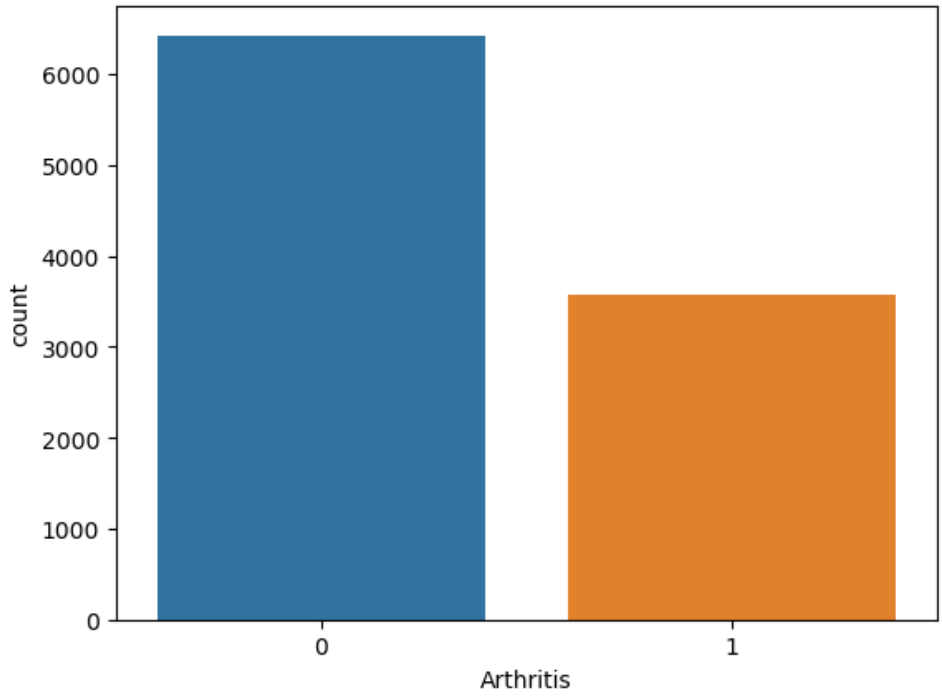
In [74]:

```
1
```

```
sns.countplot(x= 'Arthritis', data = med_data)
```

Out[74]:

<Axes: xlabel='Arthritis', ylabel='count'>

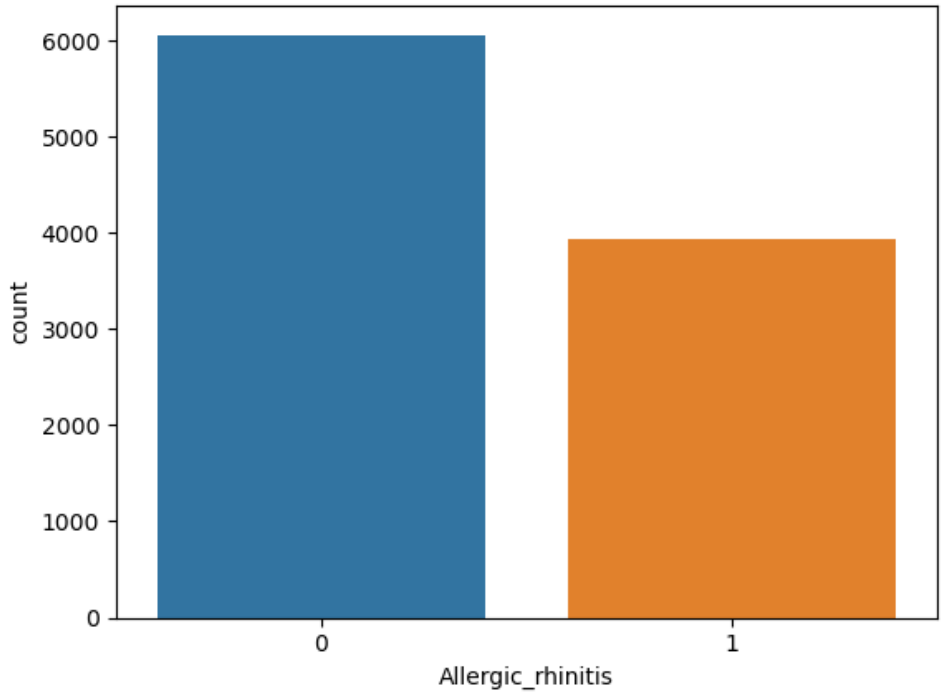


In [75]:

```
1
sns.countplot(x= 'Allergic_rhinitis', data = med_data)
```

Out[75]:

<Axes: xlabel='Allergic_rhinitis', ylabel='count'>

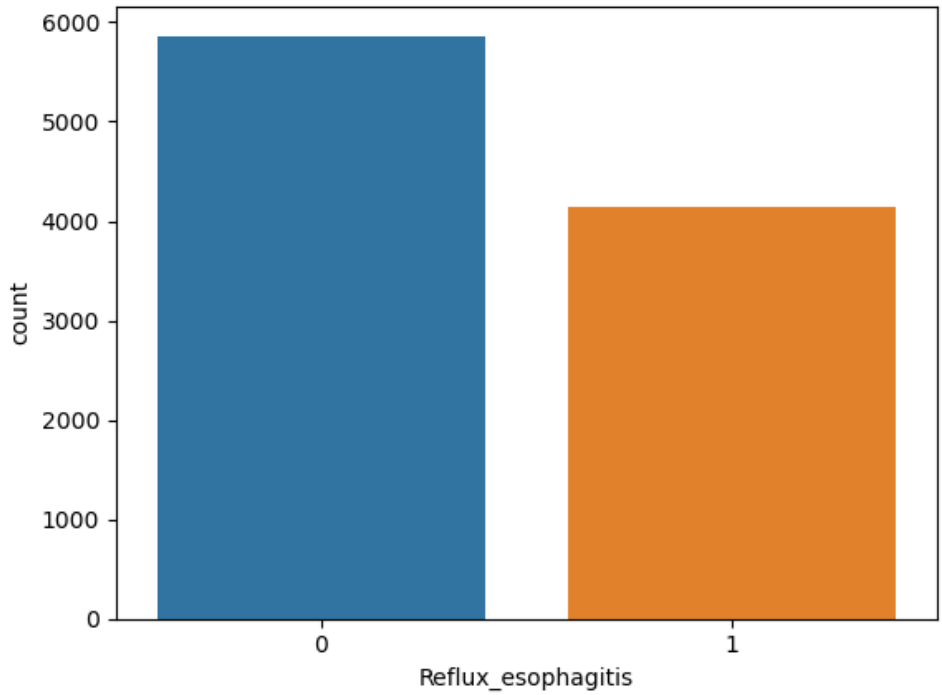


In [76]:

```
1
sns.countplot(x= 'Reflux_esophagitis', data = med_data)
```

Out[76]:

```
<Axes: xlabel='Reflux_esophagitis', ylabel='count'>
```

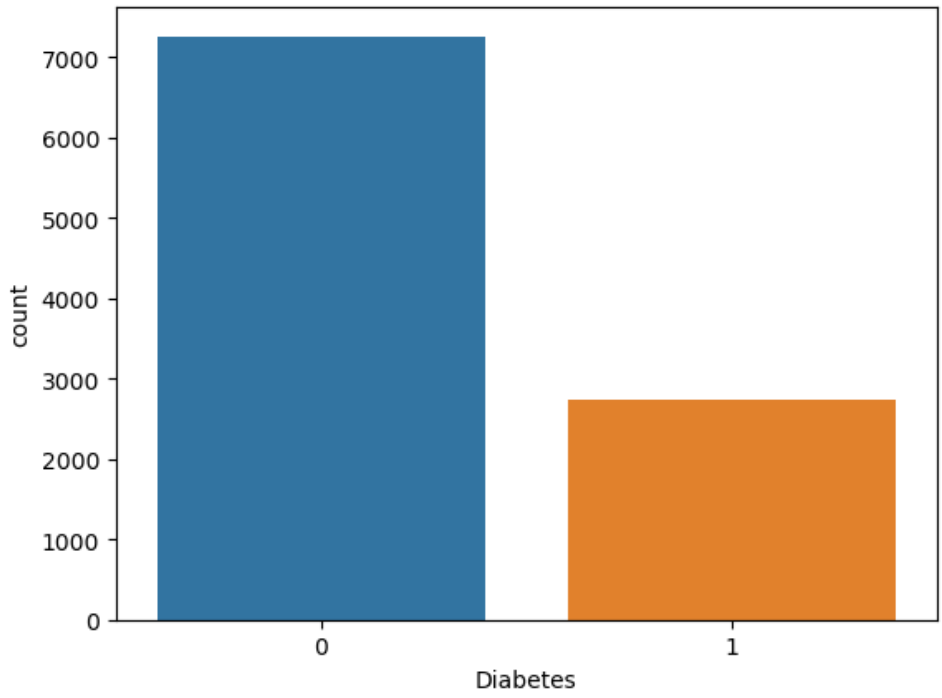


In [77]:

```
1
sns.countplot(x= 'Diabetes', data = med_data)
```

Out[77]:

```
<Axes: xlabel='Diabetes', ylabel='count'>
```

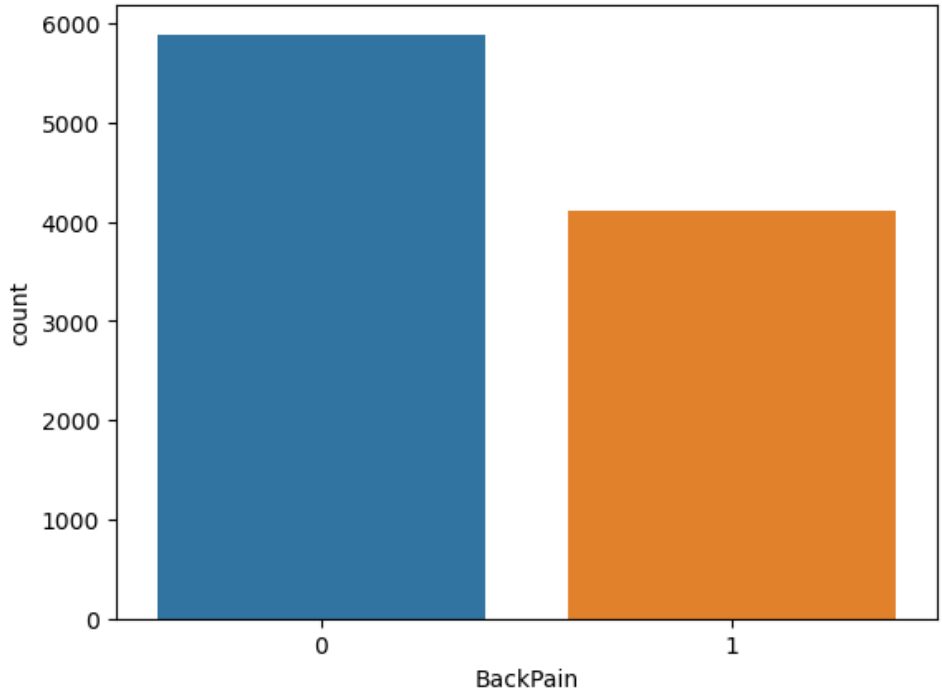


In [78]:

```
sns.countplot(x= 'BackPain', data = med_data)
```

Out[78]:

```
<Axes: xlabel='BackPain', ylabel='count'>
```

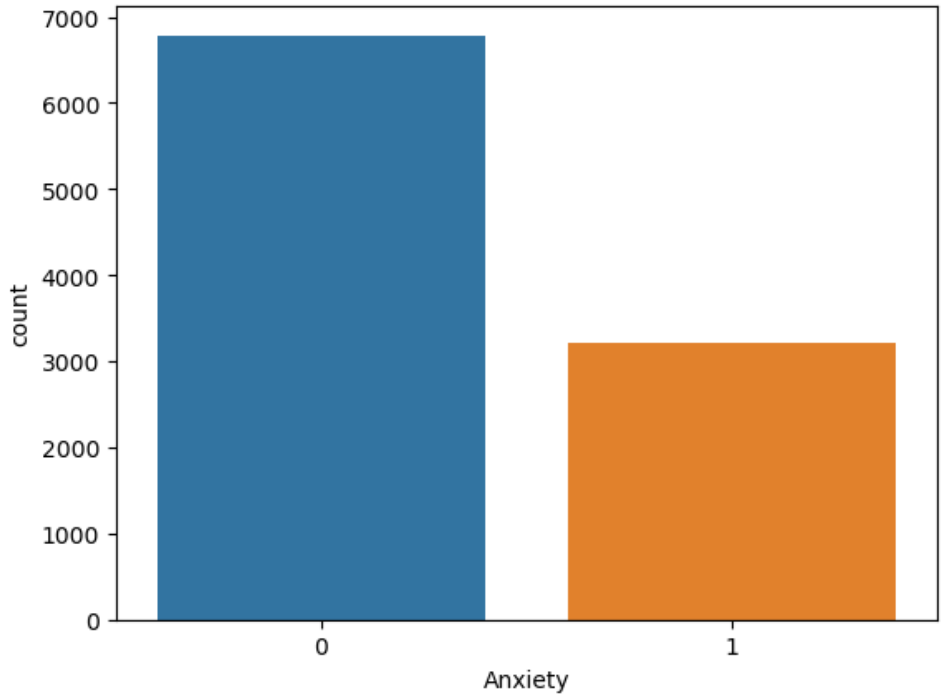


In [79]:

```
sns.countplot(x= 'Anxiety', data = med_data)
```

Out[79]:

<Axes: xlabel='Anxiety', ylabel='count'>

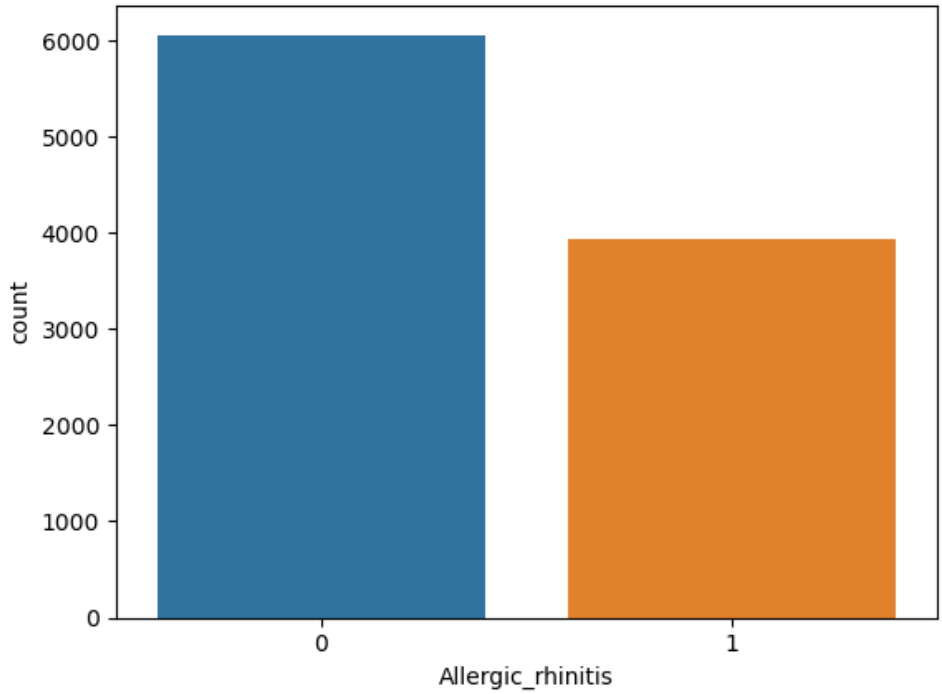


In [80]:

```
1
sns.countplot(x= 'Allergic_rhinitis', data = med_data)
```

Out[80]:

<Axes: xlabel='Allergic_rhinitis', ylabel='count'>

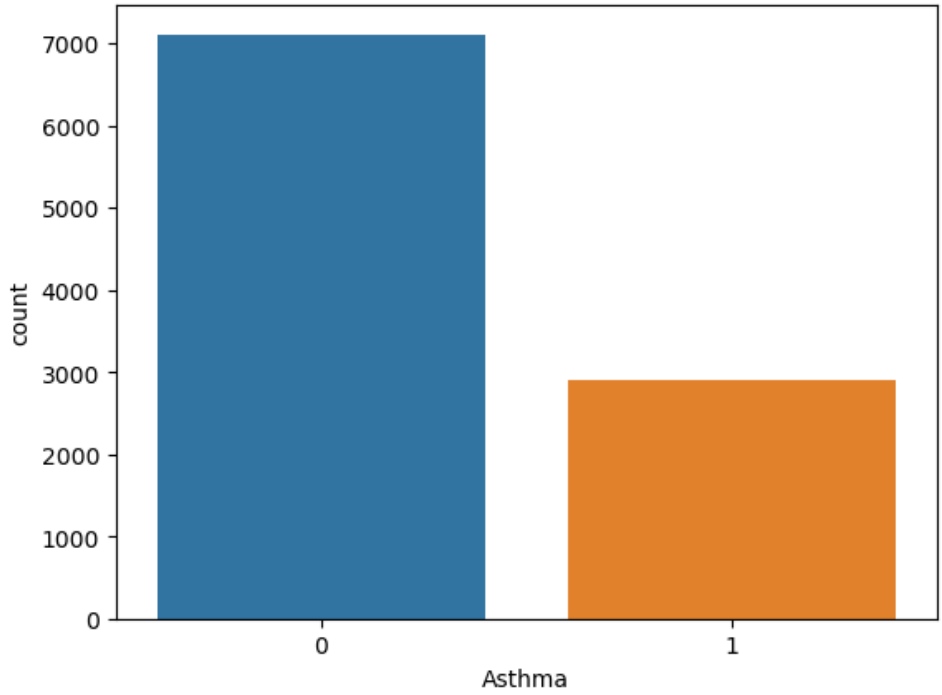


In [81]:

```
1
sns.countplot(x= 'Asthma', data = med_data)
```

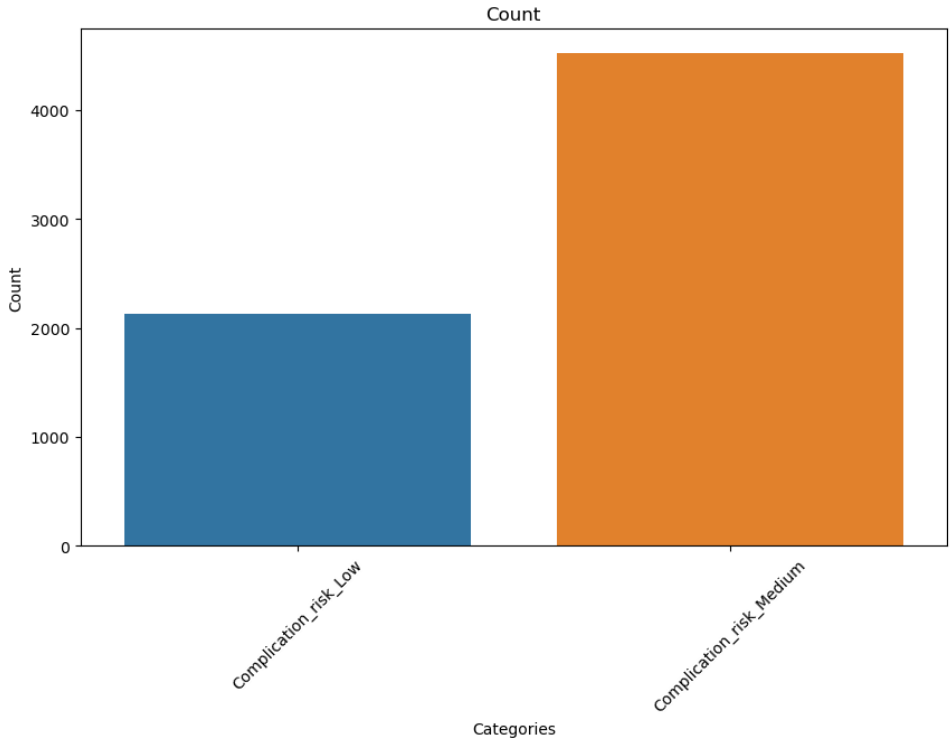
Out[81]:

```
<Axes: xlabel='Asthma', ylabel='count'>
```



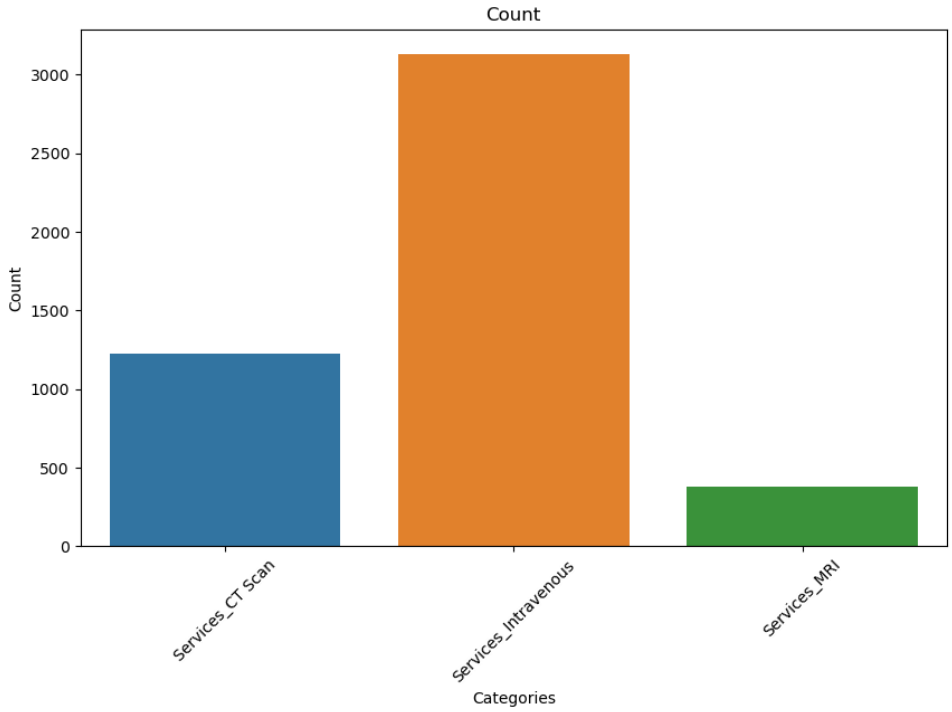
In [82]:

```
1
selected_columns = ['Complication_risk_Low', 'Complication_risk_Medium']
2
category_counts = med_data[selected_columns].sum()
3
plt.figure(figsize=(10, 6))
4
sns.barplot(x=category_counts.index, y=category_counts.values)
5
plt.xticks(rotation=45)
6
plt.xlabel('Categories')
7
plt.ylabel('Count')
8
plt.title('Count')
9
plt.show()
```



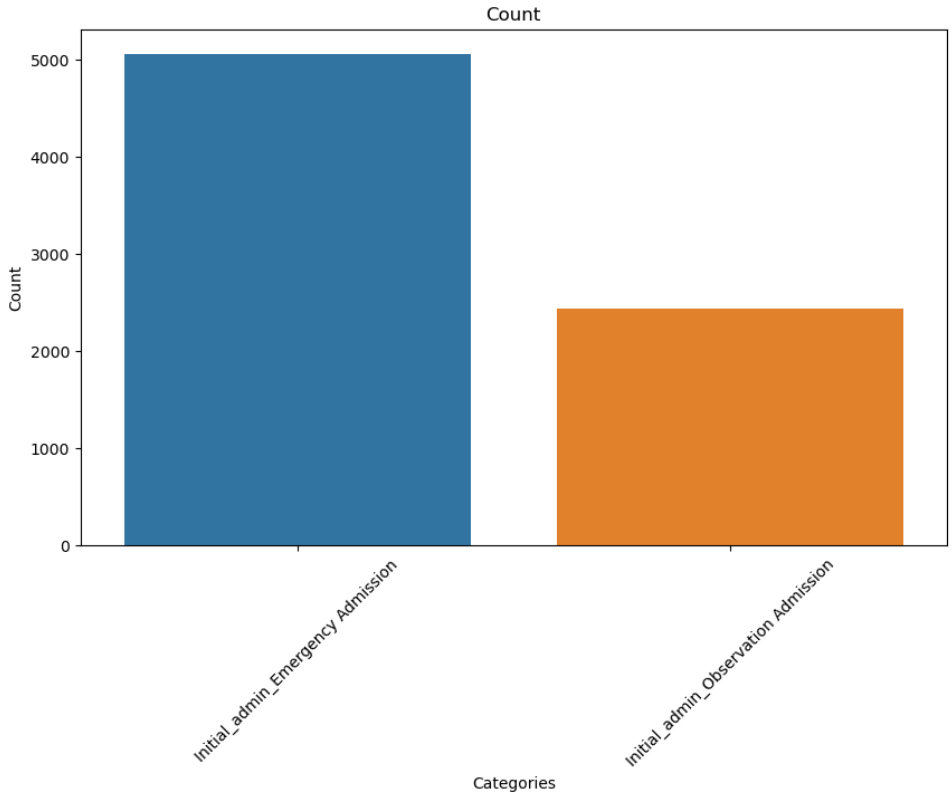
In [83]:

```
selected_columns = ['Services_CT Scan', 'Services_Intravenous',
    'Services_MRI',]


category_counts = med_data[selected_columns].sum()
plt.figure(figsize=(10, 6))
sns.barplot(x=category_counts.index, y=category_counts.values)
plt.xticks(rotation=45)
plt.xlabel('Categories')
plt.ylabel('Count')
plt.title('Count')
plt.show()
```



In [84]:

```
selected_columns = ['Initial_admin_Emergency Admission','Initial_admin_Observation Admission']

category_counts = med_data[selected_columns].sum()

plt.figure(figsize=(10, 6))

sns.barplot(x=category_counts.index, y=category_counts.values)

plt.xticks(rotation=45)

plt.xlabel('Categories')

plt.ylabel('Count')

plt.title('Count')

plt.show()
```



In [85]:

```
1
```

 # bivariant visualization

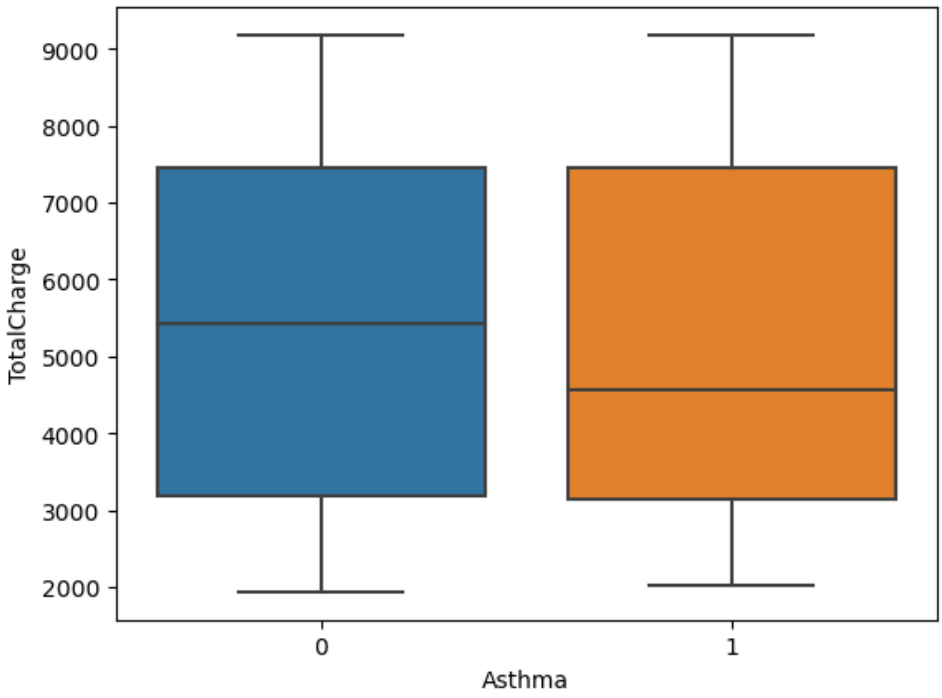
In [86]:

```
1
```

```
sns.boxplot(x=med_data['Asthma'], y=med_data['TotalCharge'])
```

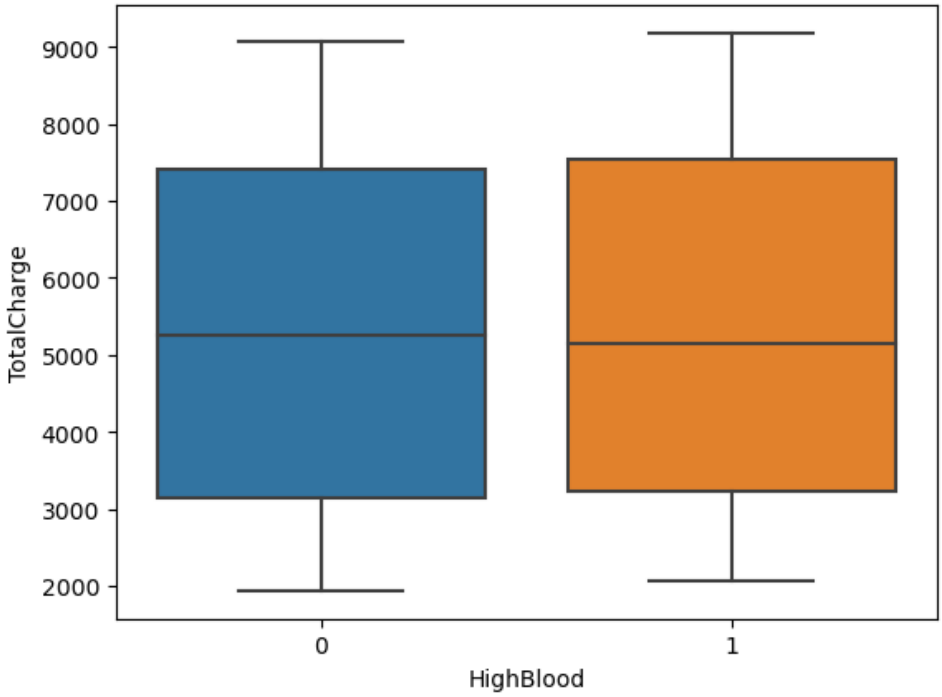
```
2
```

```
plt.show()
```



In [87]:

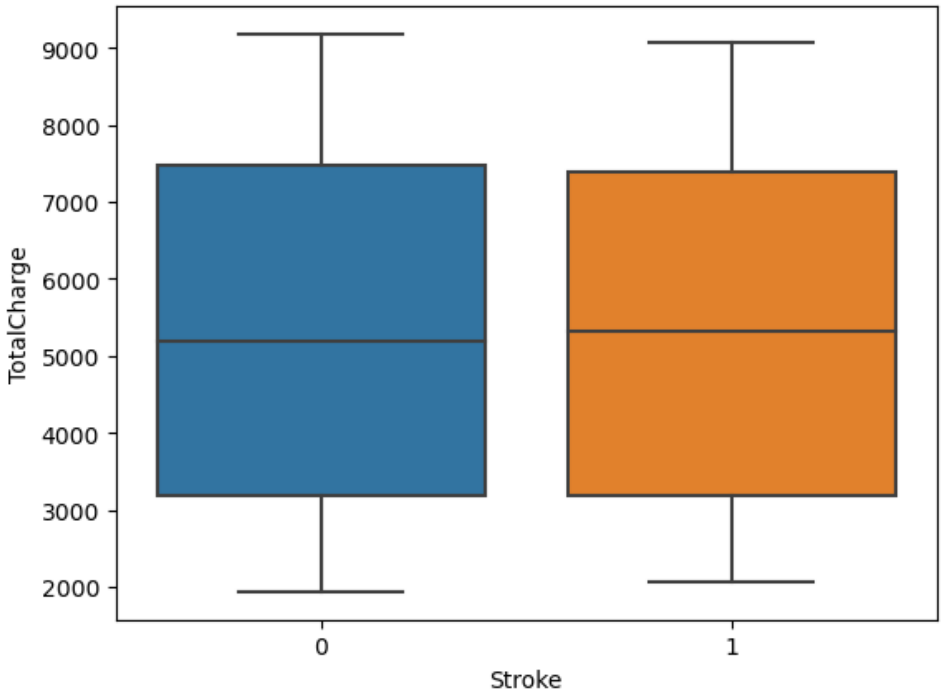
```
1 sns.boxplot(x=med_data['HighBlood'], y=med_data[ 'TotalCharge'])
2 plt.show();
```



In [88]:

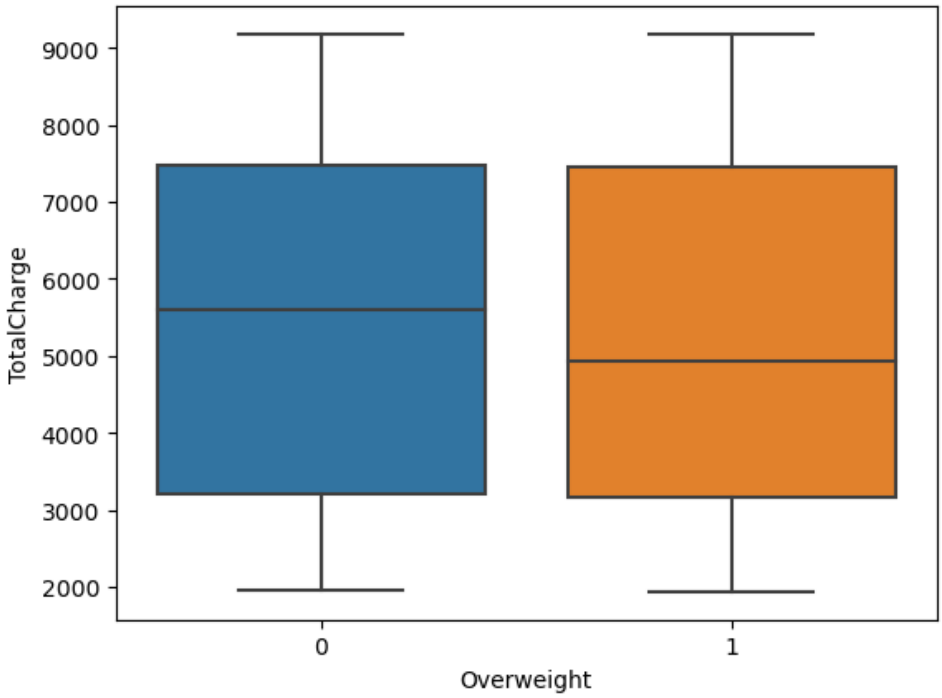
```
1 sns.boxplot(x=med_data['Stroke'], y=med_data['TotalCharge'])
```

```
2
plt.show();
```



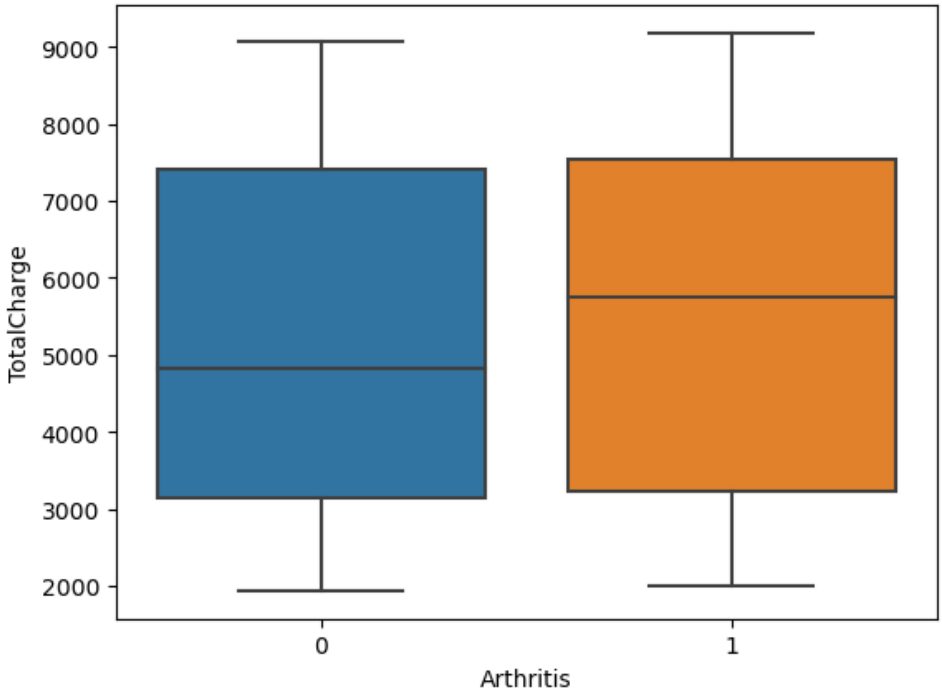
In [89]:

```
1
sns.boxplot(x=med_data['Overweight'], y=med_data['TotalCharge'])
2
plt.show();
```



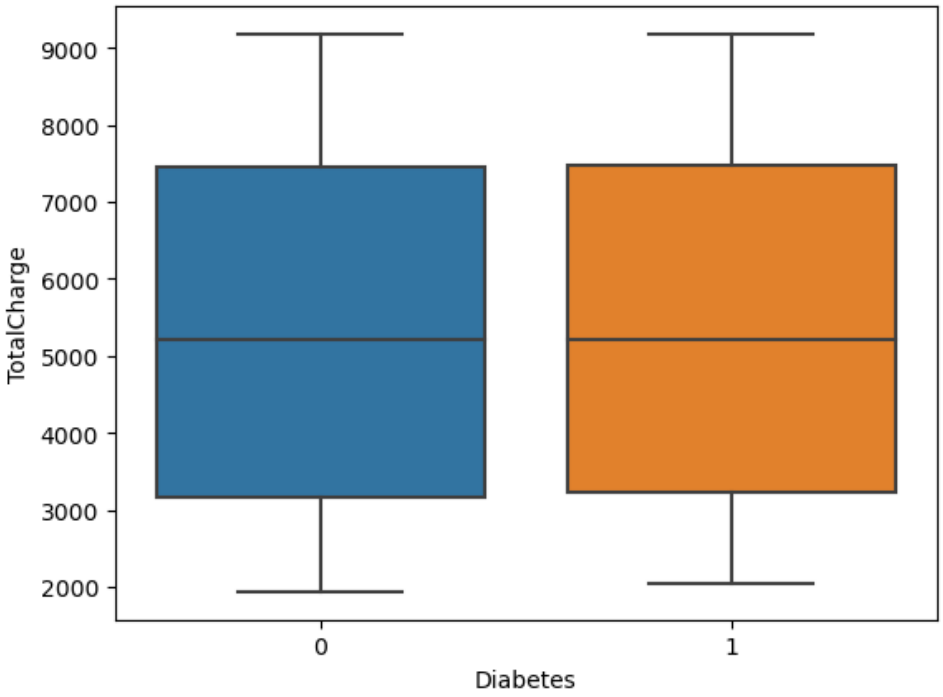
In [90]:

```
1
sns.boxplot(x=med_data['Arthritis'], y=med_data['TotalCharge'])
2
plt.show();
```



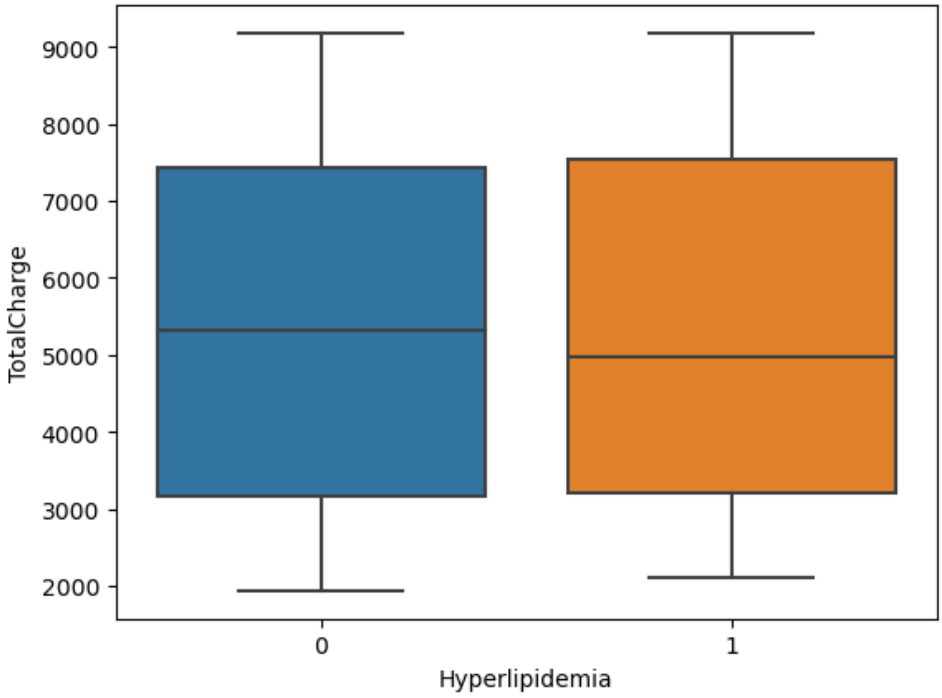
In [91]:

```
1
sns.boxplot(x=med_data['Diabetes'], y=med_data['TotalCharge'])
2
plt.show();
```



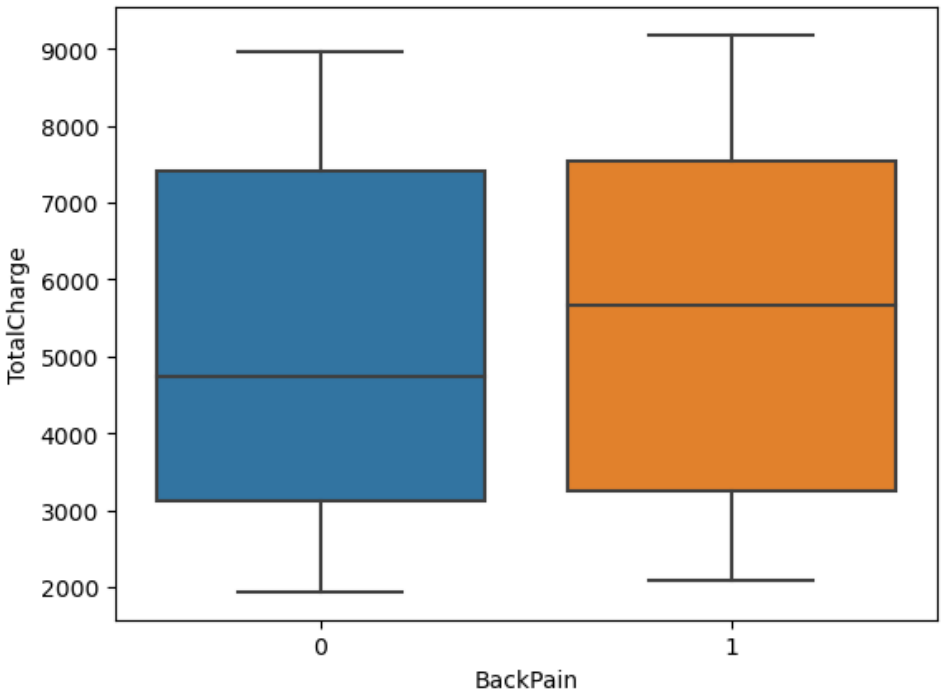
In [92]:

```
sns.boxplot(x=med_data['Hyperlipidemia'], y=med_data['TotalCharge'])

plt.show();
```



In [93]:

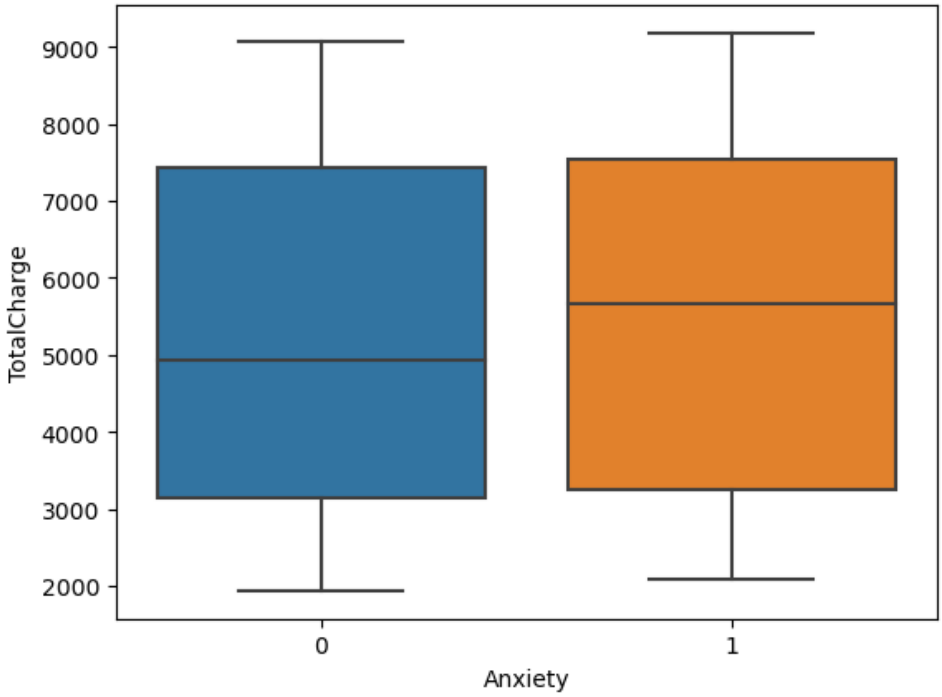
```
sns.boxplot(x=med_data['BackPain'], y=med_data['TotalCharge'])
```




```
plt.show();
```

In [94]:

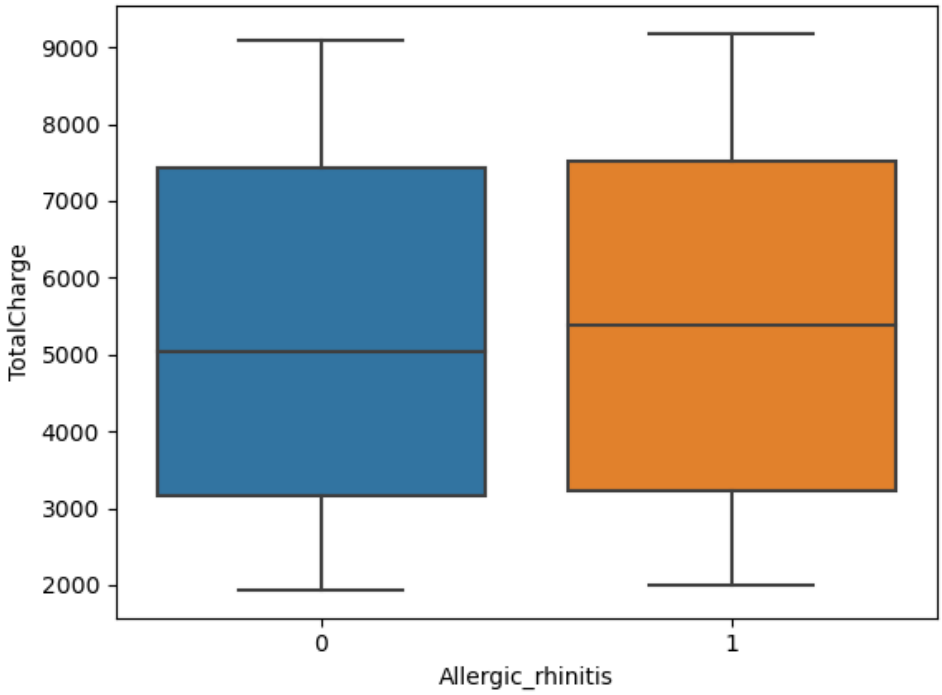
```
sns.boxplot(x=med_data['Anxiety'], y=med_data['TotalCharge'])

plt.show();
```



In [95]:

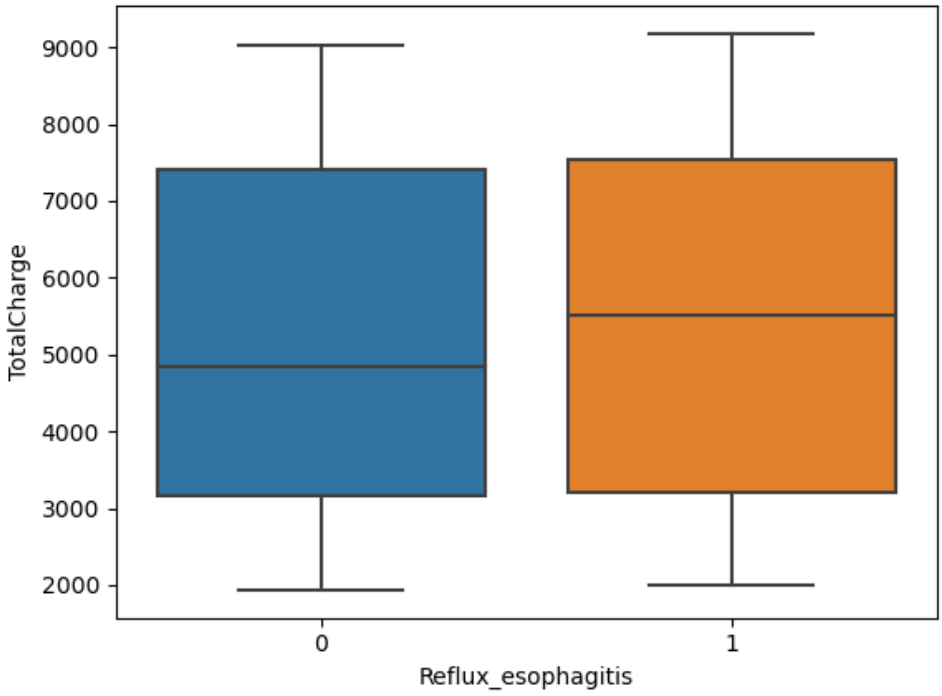
```
sns.boxplot(x=med_data['Allergic_rhinitis'], y=med_data['TotalCharge'])
```

```
2
plt.show();
```



In [96]:

```
1
sns.boxplot(x=med_data['Reflux_esophagitis'], y=med_data['TotalCharge'])
2
plt.show();
```



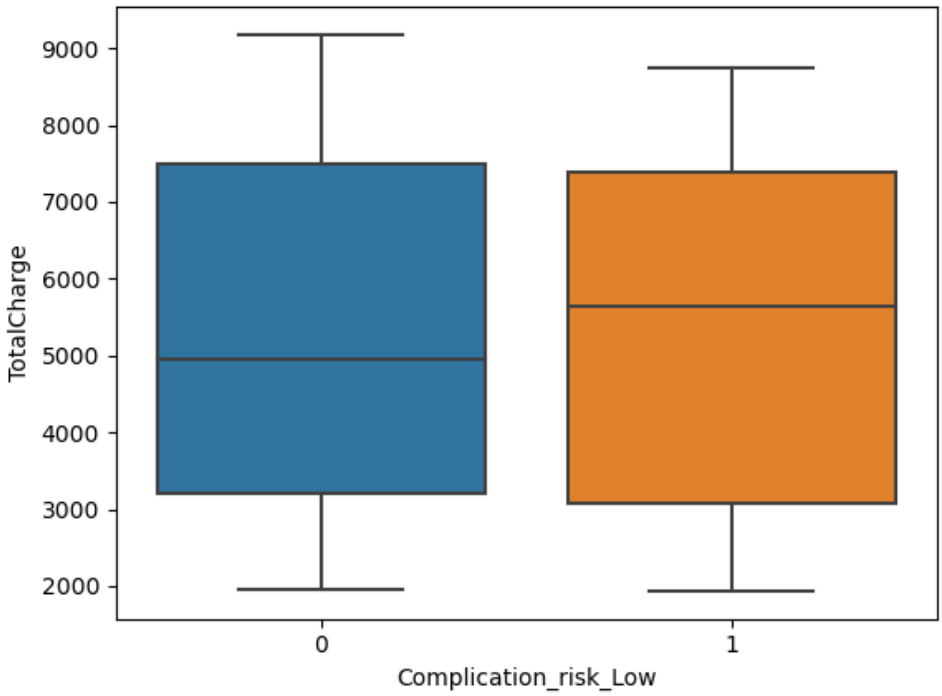
In [97]:

1

#bivarent visualization for one hot encoded variable.

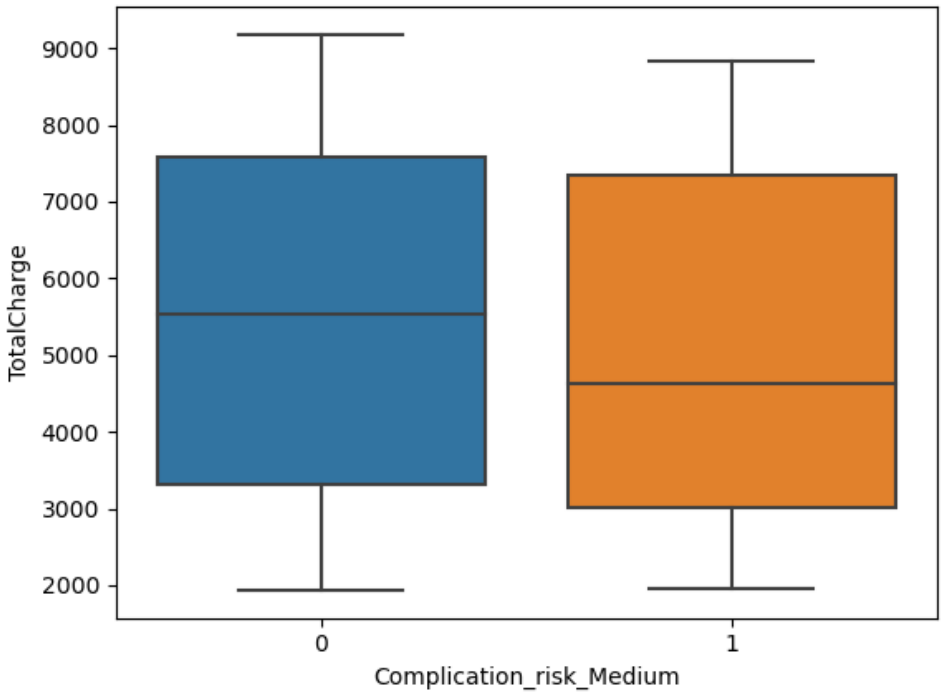
In [98]:

```
1
sns.boxplot(x=med_data['Complication_risk_Low'], y=med_data['TotalCharge'])
2
plt.show();
```



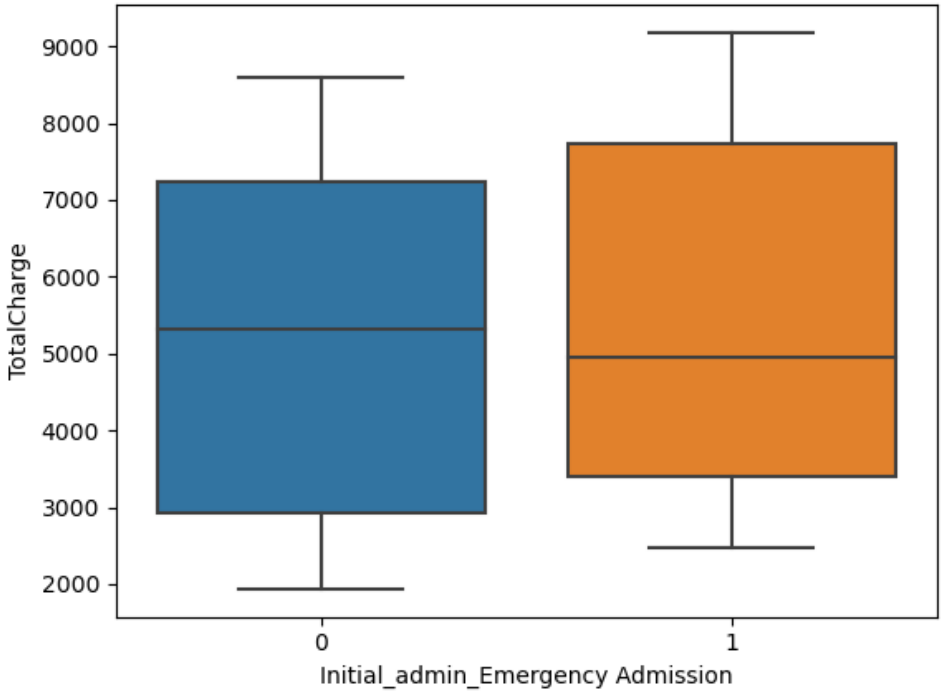
In [99]:

```
1
sns.boxplot(x=med_data['Complication_risk_Medium'], y=med_data['TotalCharge'])
2
plt.show();
```



In [100]:

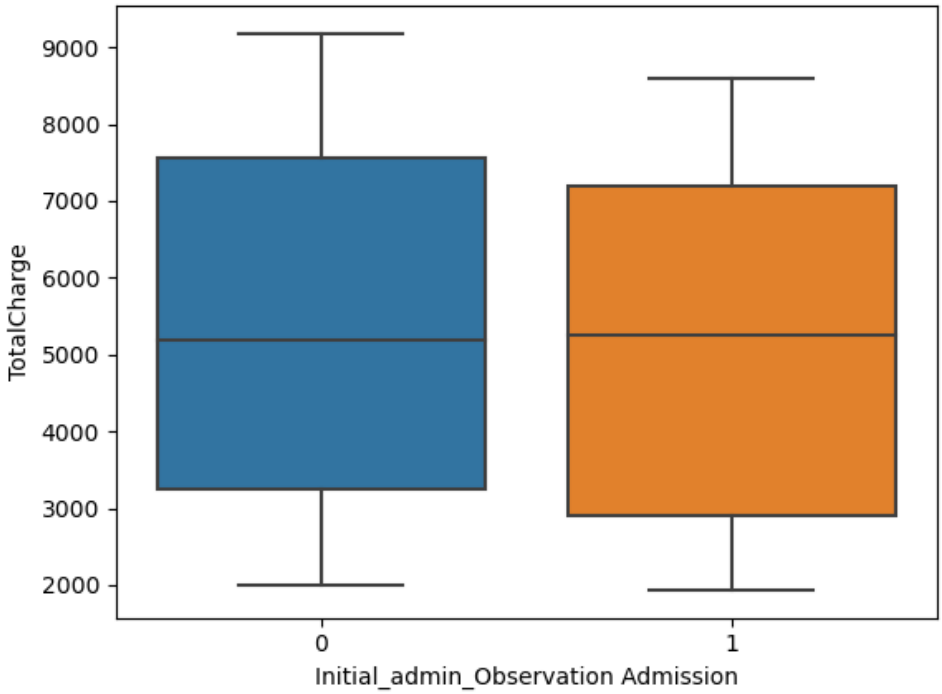
```
1
sns.boxplot(x=med_data['Initial_admin_Emergency Admission'], y=med_data['TotalCharge'])
2
plt.show();
```



In [101]:

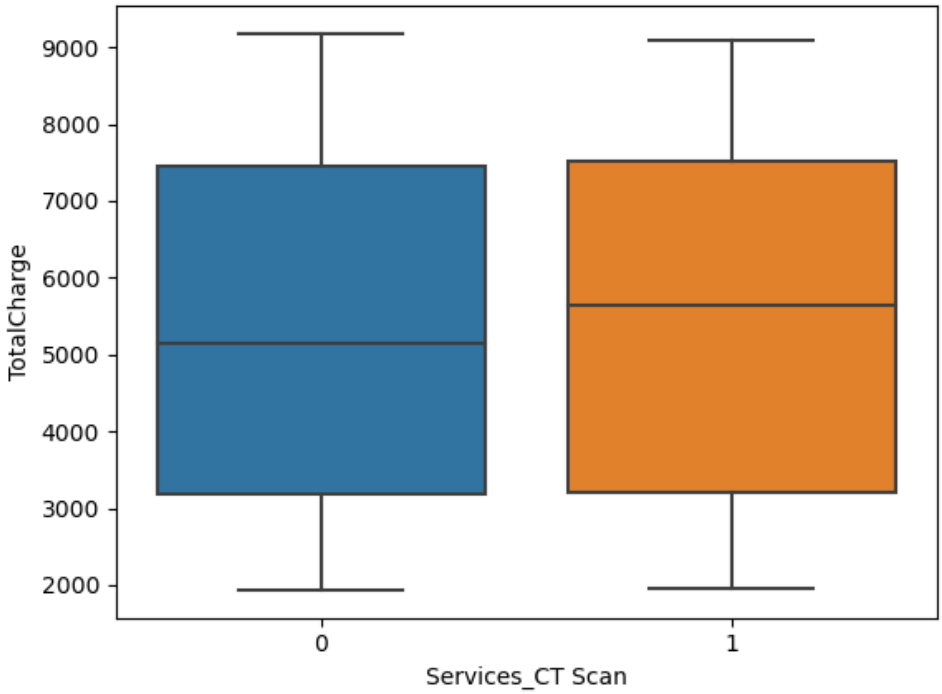
```
1
sns.boxplot(x=med_data['Initial_admin_Observation Admission'], y=med_data['TotalCharge'])
```

```
2
plt.show();
```



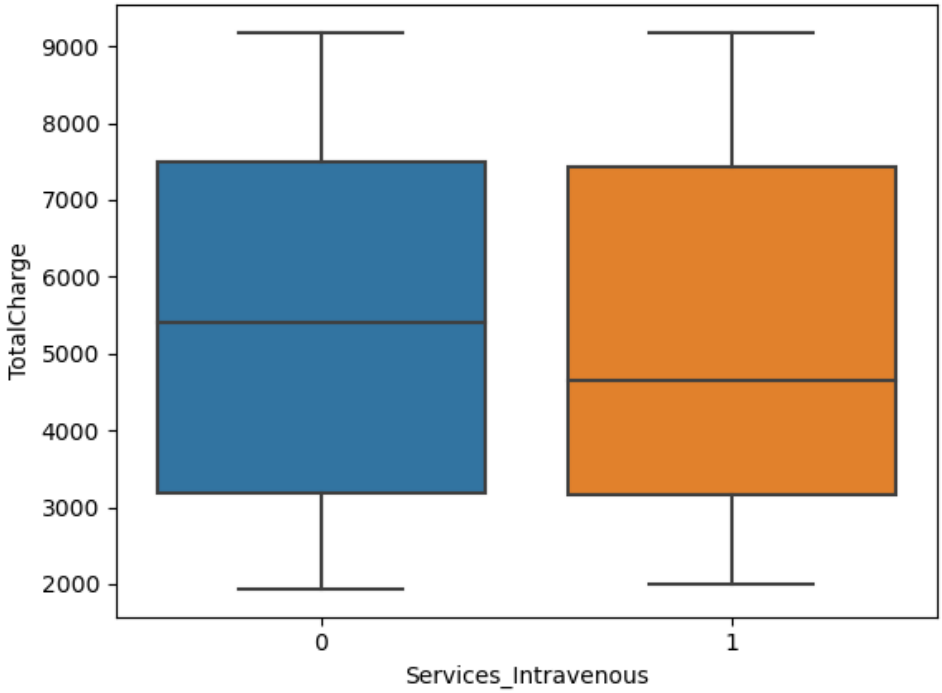
In [102]:

```
1
sns.boxplot(x=med_data['Services_CT Scan'], y=med_data['TotalCharge'])
2
plt.show();
```



In [103]:

```
1
sns.boxplot(x=med_data['Services_Intravenous'], y=med_data['TotalCharge'])
2
plt.show();
```

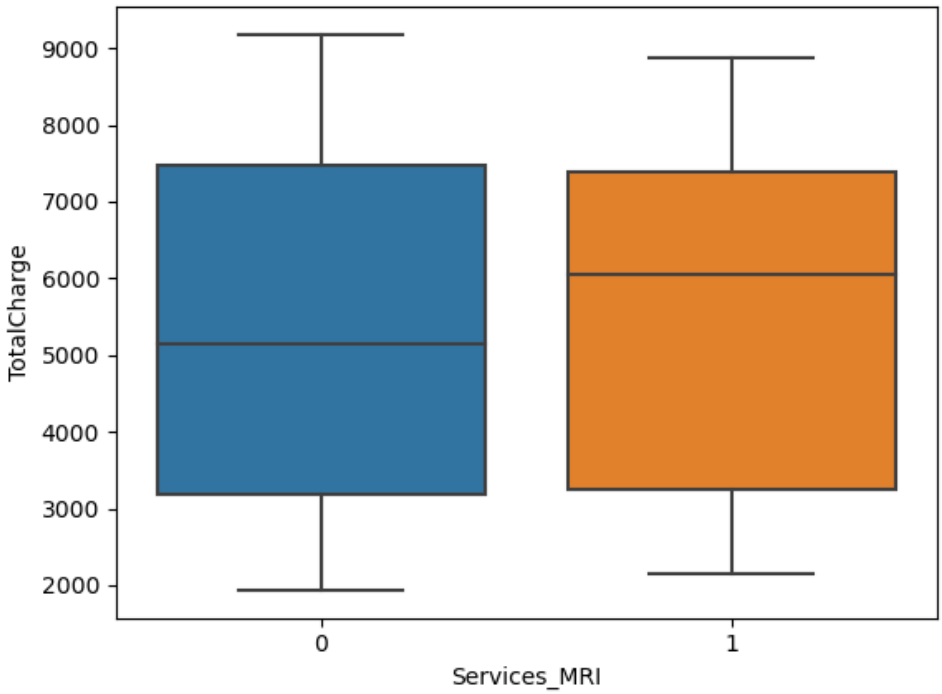


In [104]:

```
1
sns.boxplot(x=med_data['Services_MRI'], y=med_data['TotalCharge'])
2
```

Out[104]:

```
<Axes: xlabel='Services_MRI', ylabel='TotalCharge'>
```



In [106]:

```
1
```

#C4. Describe your data transformation goals

2

As said in C2, the data is clean. It has no missing values. The columns with outliners are treated using the z-score method. Columns that are irrelevant to the research question are removed.
24 independent variables are prepared for the first model. All the categorical variables are converted to a numerical value. For categorical variables with more than two options (i.e. 'Services','Complication_risk','Initial_admin'), one hot encoding is used to give variables numerical values.

In [107]:

```
1
```

#C5. Provide the prepared data set as a CSV file.

2

3

```
med_data.to_csv('MSDA208_Task1_prepared_Data.csv')
```

Part IV: Model Comparison and Analysis

In [108]:

```
1
```

 # D1. Initial model

In [109]:

```
1
```

```
y = med_data['TotalCharge']


independent_vars = ['ReAdmis', 'VitD_levels', 'Doc_visits', 'Full_meals_eaten',
  'vitD_supp','Soft_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis',
  'Diabetes','Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
  'Reflux_esophagitis', 'Asthma','Services_CT Scan',
    'Services_Intravenous', 'Services_MRI', 'Complication_risk_Low',
    'Complication_risk_Medium', 'Initial_admin_Emergency Admission',
    'Initial_admin_Observation Admission']



X = med_data[independent_vars]
```

In [110]:

```
X = sm.add_constant(X)
```

In [111]:

```
model = sm.OLS(y, X)
results = model.fit()
```

In [112]:

```
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           TotalCharge   R-squared:                       0.732
Model:                            OLS   Adj. R-squared:                  0.732
Method:                 Least Squares   F-statistic:                     1136.
Date:                Fri, 15 Dec 2023   Prob (F-statistic):               0.00
Time:                        21:05:21   Log-Likelihood:                -84475.
No. Observations:               10000   AIC:                         1.690e+05
Df Residuals:                    9975   BIC:                         1.692e+05
Df Model:                          24
Covariance Type:            nonrobust
==============================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

const	3820.1552	65.940	57.934	0.000	3690.900	3949.410
ReAdmis	3808.1408	23.469	162.264	0.000	3762.137	3854.144
VitD_levels	-0.0035	0.004	-0.888	0.374	-0.011	0.004
Doc_visits	-13.6744	10.821	-1.264	0.206	-34.885	7.536
Full_meals_eaten	-13.7950	11.228	-1.229	0.219	-35.804	8.214
vitD_supp	22.3414	18.007	1.241	0.215	-12.955	57.638
Soft_drink	-15.5679	25.883	-0.601	0.548	-66.303	35.167
HighBlood	78.2364	23.009	3.400	0.001	33.134	123.339
Stroke	-14.0368	28.305	-0.496	0.620	-69.520	41.446
Overweight	-16.3372	24.912	-0.656	0.512	-65.170	32.495
Arthritis	127.4260	23.602	5.399	0.000	81.160	173.692
Diabetes	76.4792	25.377	3.014	0.003	26.735	126.224
Hyperlipidemia	61.7879	23.931	2.582	0.010	14.877	108.698
BackPain	111.1109	22.995	4.832	0.000	66.037	156.185
Anxiety	131.8782	24.210	5.447	0.000	84.423	179.334
Allergic_rhinitis	93.0397	23.147	4.020	0.000	47.667	138.412
Reflux_esophagitis	92.5114	22.964	4.029	0.000	47.498	137.525
Asthma	5.3183	24.952	0.213	0.831	-43.592	54.229
Services_CT Scan	-86.3690	35.880	-2.407	0.016	-156.701	-16.037
Services_Intravenous	-8.0221	25.522	-0.314	0.753	-58.050	42.006
Services_MRI	-16.0533	60.071	-0.267	0.789	-133.805	101.699
Complication_risk_Low	-335.9325	31.355	-10.714	0.000	-397.394	-274.471
Complication_risk_Medium	-437.8158	25.778	-16.984	0.000	-488.347	-387.285
Initial_admin_Emergency Admission	382.3006	27.640	13.832	0.000	328.121	436.480
Initial_admin_Observation Admission	-18.8716	32.212	-0.586	0.558	-82.013	44.269

Omnibus:	1949.565	Durbin-Watson:	1.272
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3300.631
Skew:	1.319	Prob(JB):	0.00
Kurtosis:	3.981	Cond. No.	3.55e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.55e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [113]:

```
1
```

#D2.Justify a statistically based feature selection procedure or a model evaluation metric to reduce the initial model in a way that aligns with the research question.

In the initial model, we have 24 independent variables. Using the feature selection method, we will be removing variables that have a p-value of 0.05 or greater because that value is not statistically significant. After removing the variables, there are 13 independent variables that have p-values less than 0.05 which will be used in the reduced Mutiple regression model.

In [114]:

```
1
```

#D3 a reduced linear regression model that follows the feature selection

Second model after removing columns with p value of greater than 0.05

In [115]:

```
y = med_data['TotalCharge']


independent_vars = ['ReAdmis', 'HighBlood', 'Arthritis',
  'Diabetes','Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis',
  'Reflux_esophagitis','Services_CT Scan', 'Complication_risk_Low',
    'Complication_risk_Medium', 'Initial_admin_Emergency Admission']



X = med_data[independent_vars]
```

In [116]:

```
X = sm.add_constant(X)
```

In [117]:

```
model = sm.OLS(y, X)
results = model.fit()
```

In [118]:

```
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           TotalCharge   R-squared:                       0.732
Model:                            OLS   Adj. R-squared:                  0.732
Method:                 Least Squares   F-statistic:                     2098.
Date:                Fri, 15 Dec 2023   Prob (F-statistic):               0.00
Time:                        21:05:42   Log-Likelihood:                -84479.
No. Observations:               10000   AIC:                         1.690e+05
Df Residuals:                    9986   BIC:                         1.691e+05
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const                       3713.6803     34.221    108.522      0.000    3646.601    3780.760
ReAdmis                     3808.7516     23.453    162.400      0.000    3762.779    3854.724
HighBlood                     77.2116     22.991      3.358      0.001      32.145     122.278
Arthritis                    127.0708     23.589      5.387      0.000      80.832     173.310
```

```
Diabetes                          75.5697     25.344      2.982      0.003      25.891     125.248
Hyperlipidemia                    63.2371     23.907      2.645      0.008      16.375     110.099
BackPain                         110.9386     22.976      4.829      0.000      65.902     155.975
Anxiety                          131.7433     24.194      5.445      0.000      84.318     179.169
Allergic_rhinitis                 93.5637     23.127      4.046      0.000      48.231     138.897
Reflux_esophagitis                93.2730     22.950      4.064      0.000      48.287     138.259
Services_CT Scan                 -83.3393     34.486     -2.417      0.016    -150.940     -15.739
Complication_risk_Low           -335.1653     31.334    -10.696      0.000    -396.587    -273.744
Complication_risk_Medium        -438.5073     25.750    -17.030      0.000    -488.982    -388.033
Initial_admin_Emergency Admission 390.5554    22.606     17.276      0.000     346.243     434.868
==============================================================================
Omnibus:                     1956.623   Durbin-Watson:                   1.271
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3318.938
Skew:                           1.322   Prob(JB):                         0.00
Kurtosis:                       3.988   Cond. No.                         6.57
==============================================================================

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

In [119]:

```
1
```

#E1. Explain your data analysis process by comparing the initial multiple linear regression model and reduced linear regression model, including the following element:

The first model was conducted with 24 independent variables, and it has the same R-square and adjusted R-square values of 0.732. This shows that 73% of the dependent variability is explained by the first model. The model exhibits robustness since it effectively accounts for a large portion of the variance in the dependent variable. In addition, the adjusted R-squared value closely aligns with the R-squared value.

After removing variables with p-values of 0.05 or greater, the reduced model is constructed with 13 independent variables. Although the reduced model has a significantly lower number of variables, R square value is the same as the first model which is 0.732 and the adjusted r square has 0.731. This shows that 73% of the dependent variability is explained by reduced model as well. The first model was overfitted, and in the reduced model we removed variables affecting it.

In [120]:

```
1
```

#E2

In [121]:

```
1
```

```
residuals = results.resid
```

In [122]:

```
1
```

```
mse = np.mean(residuals**2)
```

```
2
```

```
sre = np.sqrt(mse)
```

In [123]:

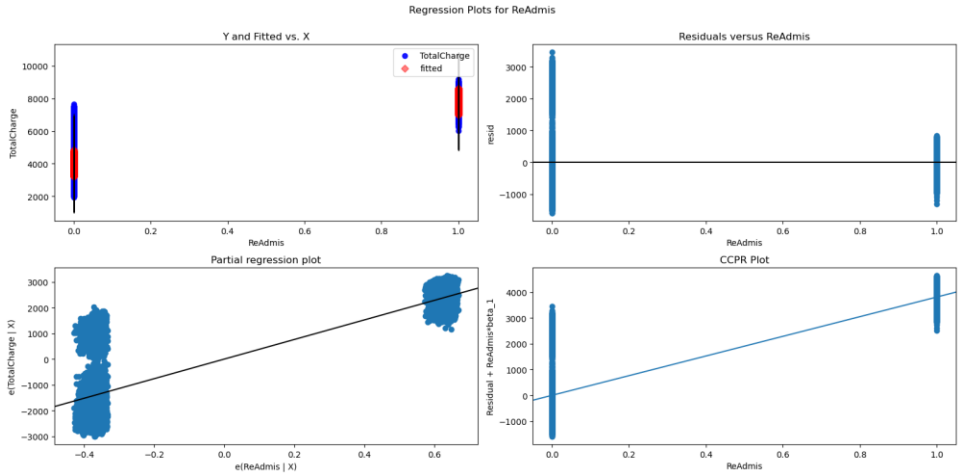
```
1
```

```
print("Standard Residual Error (SRE):", sre)
```

```
Standard Residual Error (SRE): 1128.8094377603525
```

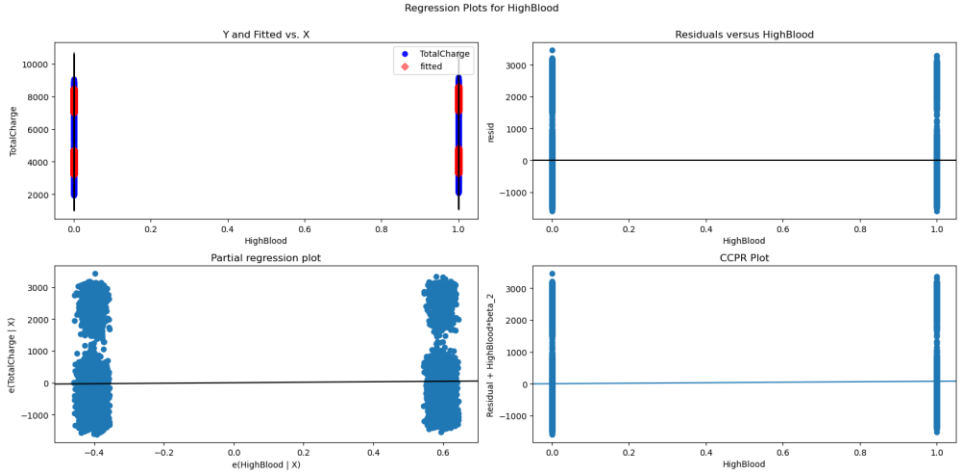
In [124]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'ReAdmis', fig=fig);
```



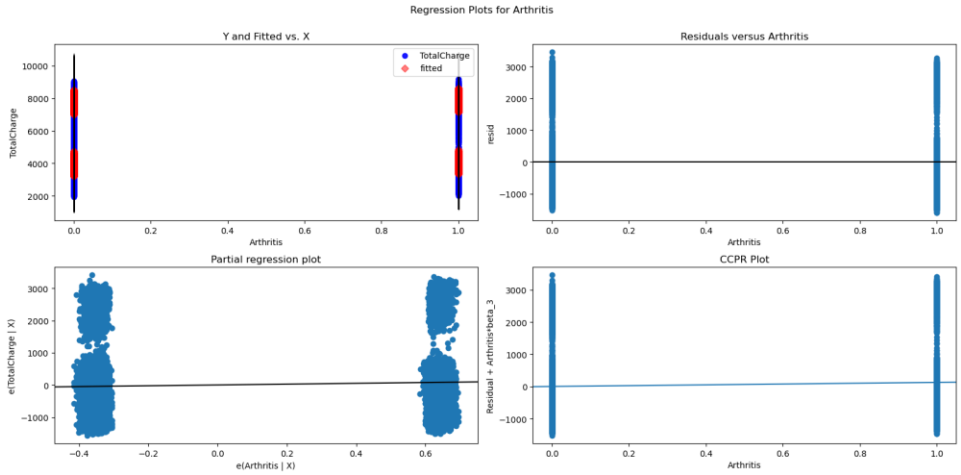
In [125]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'HighBlood', fig=fig);
```



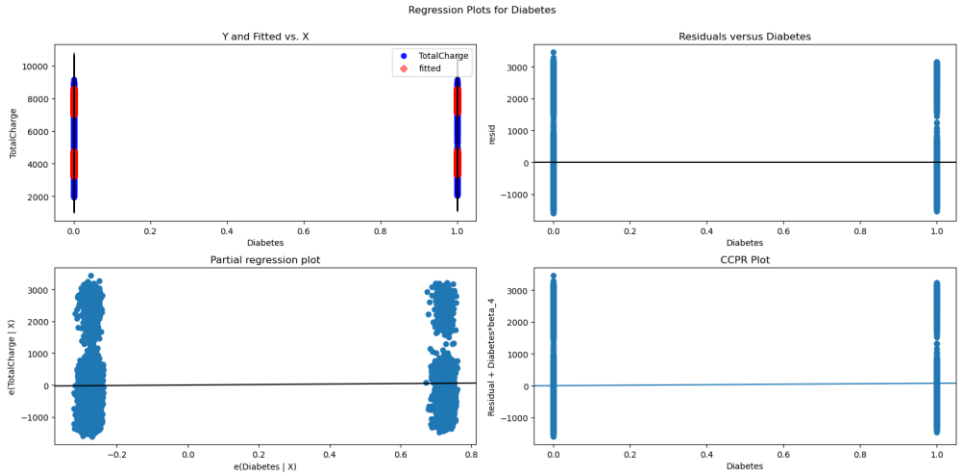
In [126]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Arthritis', fig=fig);
```



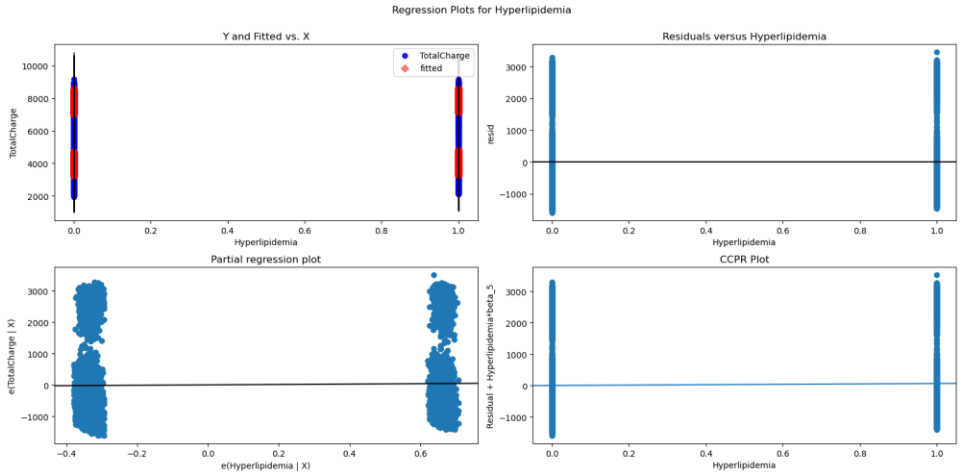
In [127]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Diabetes', fig=fig);
```



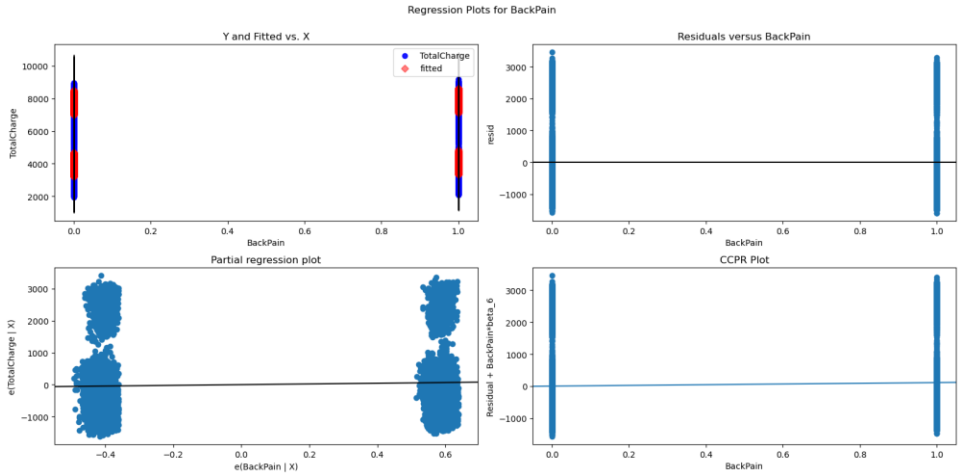
In [128]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Hyperlipidemia', fig=fig);
```



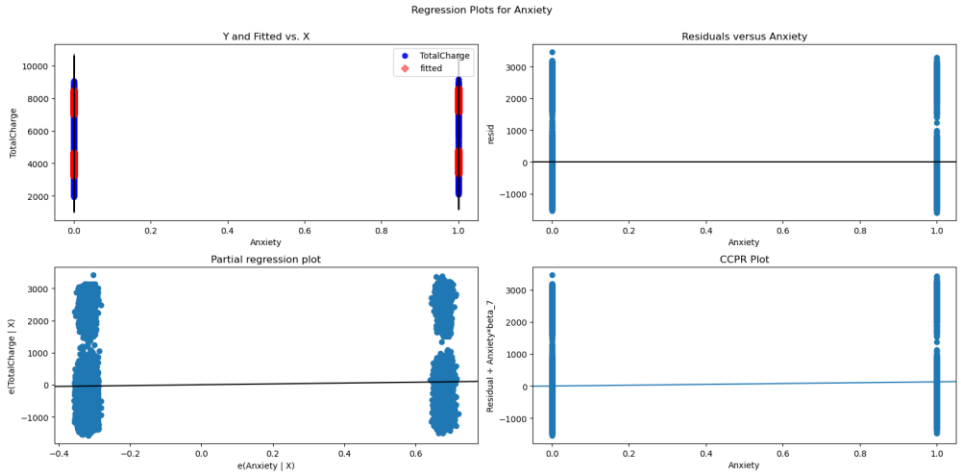
In [129]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'BackPain', fig=fig);
```



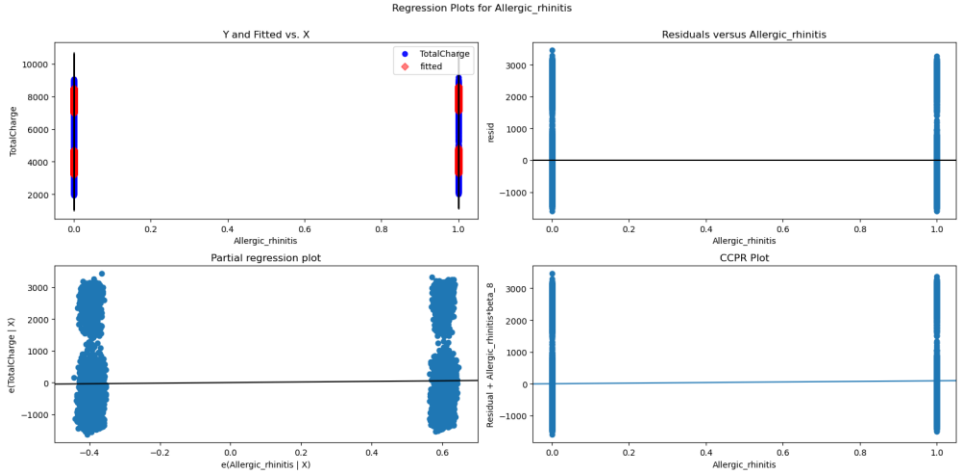
In [130]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Anxiety', fig=fig);
```



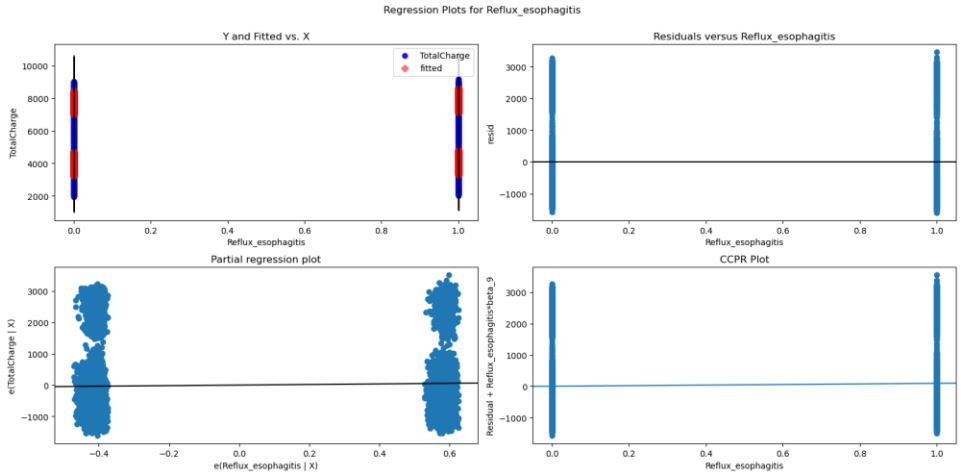
In [131]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Allergic_rhinitis', fig=fig);
```



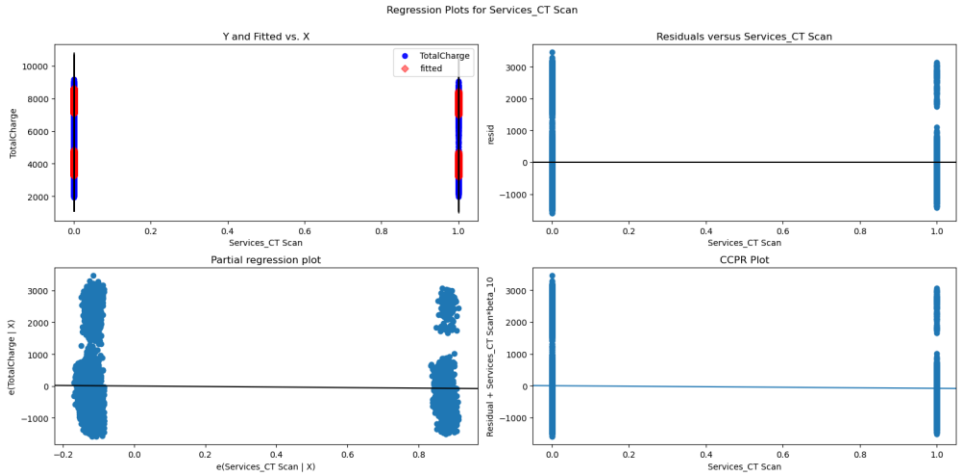
In [132]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Reflux_esophagitis', fig=fig);
```



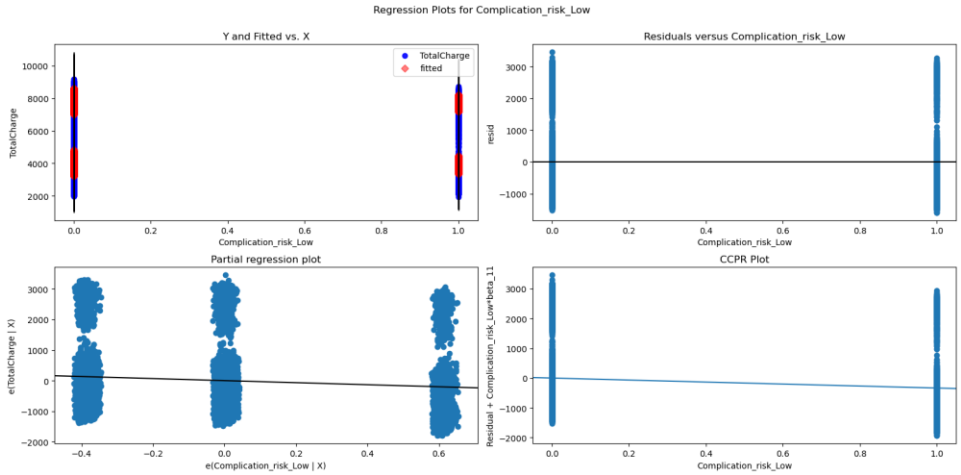
In [133]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Services_CT Scan', fig=fig);
```



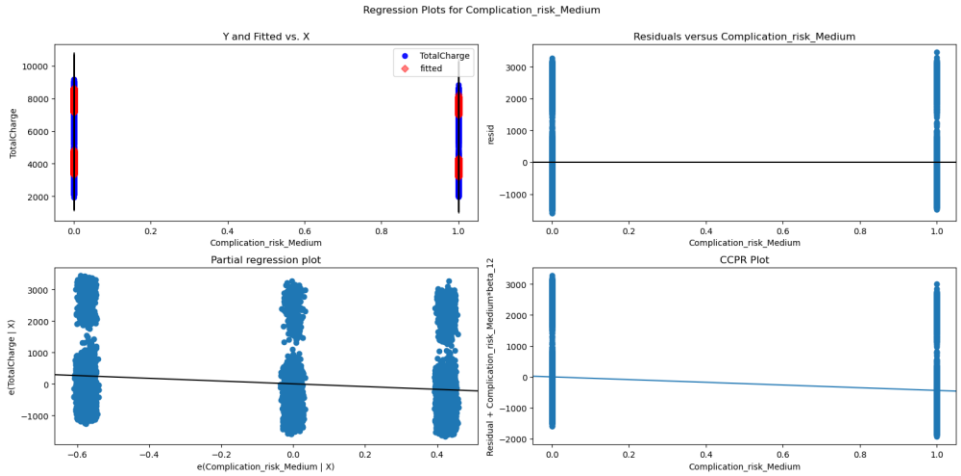
In [134]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Complication_risk_Low', fig=fig);
```



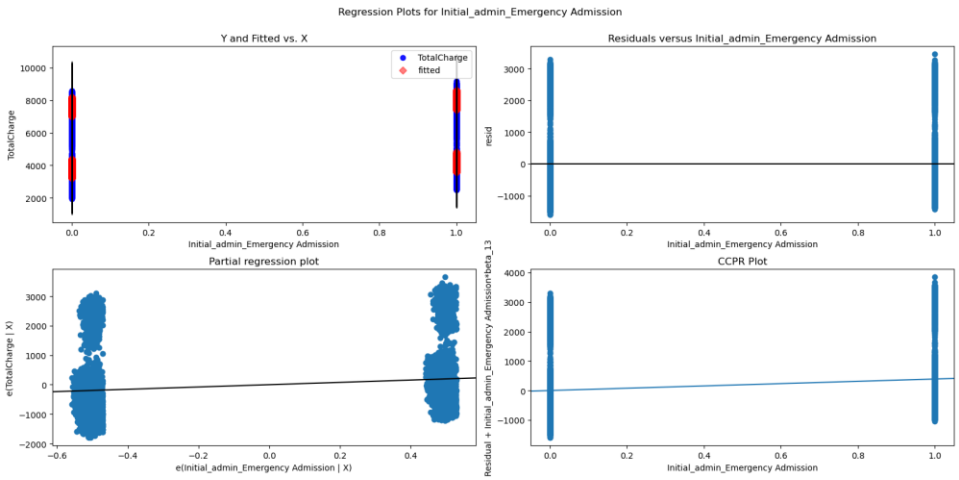
In [135]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Complication_risk_Medium', fig=fig);
```



In [136]:

```
1
fig=plt.figure(figsize=[16,8])
2
sm.graphics.plot_regress_exog(results,'Initial_admin_Emergency Admission', fig=fig);
```



In [137]:

```
1
```

#E3.Provide an executable error-free copy of the code used to support the implementation of the linear regression models using a Python or R file.

```
2
```

```
3
```

```
med_data.to_csv('MSDA208_PA_task1_E3.cvs')
```

In [ ]:

```
1
```

Part V: Data Summary and Implications

In [138]:

```
1
```

#F1. Discuss the results of your data analysis

y = 3714.1800 + 3808.7579 (ReAdmis)+ 77.2108 (HighBlood)+ 127.0744 (Arthritis)+ 75.5687 (Diabetes)+ 63.2353 (Hyperlipidemia) + 110.9397 (BackPain) + 131.7639Anxiety+ 93.5506 (Allergic_rhinitis)+ 93.2684(Reflux_esophagitis)+ ((-83.3542- Services_CT Scan) + (-335.1634 Complication_risk_Low)+ (-438.5156 Complication_risk_Medium)+ 390.5600 (Initial_admin_Emergency Admission*)

In [139]:

```
1
```

#an interpretation of the coefficients of the reduced model

```
2
```

**assuming everything else is constant
When a patent is readmisted , the total charge increases by 3808.7579.
For patients with high blood pressure, the total cost increases by 77.2108.
When "Arthritis" is 1, Y increases by 127.0744. For patients with arthritis, the total charge increases by 127.0744.
For patients with diabetes, the total charge increases by 75.5687.
For patients with hyperlipidemia, the total charge increases by 63.2353.
For patients with back pain, the total charge increases by 110.9397.
For patients with anxiety, the total charge increases by 131.7639.
For patients with allergic rhinitis, the total charge increases by 93.5506.

For patients with reflux esophagitis, the total charge increases by 93.2684.
For patients who use the "Services_CT Scan," their total charge decreases by 83.3542.
For patients with low complication risks, their total charge decreases by 335.1634.
For patients with medium complication risk, their total charge decreases by 438.5156.
For patients who had "Initial_admin_Emergency Admission," their total charge increased by 390.5600.

In [140]:
1

#The statistical and practical significance of the reduced model

The Prob(F-Statistic) value is 0.0, which indicates that the model is statistically significant. Additionally, the model is practically significant because the variables in the reduced model effectively explain changes in the dependent variable, total charge. The variables from the reduced model have a notable impact on the total charge that a patient incurs. For instance, a patient with diabetes or arthritis is more likely to require additional medical care, which consequently increases the total cost for that patient.

In [141]:
1

#The limitations of the data analysis

The analysis is based on a dataset of 10,000 patients, which may be insufficient to make accurate outcome predictions. Additionally, this analysis does not account for variations in health insurance coverage and patient's access to care. For instance, one of the independent variables, anxiety, appears to influence total charges in the reduced model. However, to gain a comprehensive understanding, it is crucial to know the specific type of insurance each patient owns or whether they have health insurance at all. Having knowledge of insurance types would provide valuable insights and facilitate the development of solutions to address this issue effectively.

In [142]:
1

#F2 Recommend a course of action based on your results.

We recommend that stakeholders examine the 13 independent variables in the reduced model that significantly affect TotalCharge for hospital stay for accurate and data-driven decision-making. We advise implementing health education outreach programs aimed at reducing hospital readmission rates for patients. Additionally, we recommend supporting health programs that offer resources for individuals with mental health and chronic diseases.

In [143]:
1

#G

#H

Atha, R. (2020, November 23). Multi-linear regression using python. Medium. https://medium.com/swlh/multi-linear-regression-using-python-44bd0d10082d Frost, J. (2023, March 21). How to interpret p-values and coefficients in regression analysis. Statistics By Jim. https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/ Scatter plot vs. Line Graph: A 2023 guide. Scatter Plot vs Line Graph A 2023 Guide Comments. (n.d.). https://ppcexpo.com/blog/scatter-plot-vs-line-graph#:~:text=A%20Scatter%20Plot%20can%20help,patterns%20of%20variables%20in%20data. Using and interpreting indicator (dummy) variables¶. Using and Interpreting Indicator (Dummy) Variables - Unifying Data Science. (n.d.). https://www.unifyingdatascience.org/html/interpreting_indicator_vars.html Wu, S. (2021, June 5). What are the best metrics to evaluate your regression model?. Medium. https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b

#J

#I

#J