# **Part I: Research Question**

A1. Summarize one research question.

Is it possible to forecast patient readmission using chronic conditions, lifestyle factors, and healthcare services?

A2. Define the goals of the data analysis. The goal of this analysis is to predict the chances of a patient's readmission to the hospital by looking at chronic conditions, lifestyle factors, and healthcare services.

# **Part II: Method Justification**

B1. Summarize four assumptions of a logistic regression model.

1.Linearity of the Logit: The relationship between the log-odds of the dependent variable and the independent variables should be linear.

2.Independence of Observations: The observations in the dataset should be independent of each other.

3. Absence of Multicollinearity: In logistic regression, multicollinearity can make it difficult to assess the individual effects of independent variables and may lead to unstable coefficient estimates.

4. Large Sample Size for Stability: Logistic regression models perform better with a large sample size.

B2. Describe two benefits of using Python or R in support of 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 logistic regression an appropriate technique is to analyze the research question summarized in part I.

This research question aims to assess the probability of patient readmission to a healthcare facility, with 'ReAdmis' as the categorical dependent variable. It considers multiple independent variables, including patient health conditions, and the health care services. Given the categorical nature of the 'ReAdmis' variable and the need to predict readmission using multiple independent variables, logistic regression is an appropriate statistical method for addressing this research question.

# **Part III: Data Preparation**

C1.The file medical data is used for this analysis. After importing the data, missing values for each variable were checked, and there are no missing values for each column in the data. next, using the std() function the outliners were assessed. Outliers were found on variables such as TotalCharge, Additional\_charges, VitD\_levels,Initial\_days. The outliners on all four columns were treated using the zscore method.

Dummy variables were created to convert categorical variables into numeric values. Also, one hot encoding was used to create a numeric value for categorical variables with more than two options such as 'Services','Complication\_risk','Initial\_admin'

The following columns were removed because they are irrelevant to answer the research question. 'CaseOrder', 'Customer\_id', 'Interaction', 'UID', 'City', 'State','County', 'Zip', 'Lat', 'Lng','Population', 'Area', 'TimeZone'

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

The data includes 50 variables and 10,000 rows, with TotalCharge as the dependent variable.

the independent variables that are used for the analysis are 'ReAdmis', 'VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten',

'vitD\_supp', 'Soft\_drink', 'Initial\_admin', 'HighBlood', 'Stroke',\_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 Yealy income.

the average doctor visit 5 times a year and the average total cost is around 5312 and average addition 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 𝑖𝑛𝑐𝑜𝑚𝑒 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 5312𝑎𝑛𝑑 𝑎 𝑚𝑎𝑥𝑖𝑚𝑢𝑚 9180.

1

**import** pandas **as** pd

2

**import** numpy **as** np

3

**import** matplotlib.pyplot **as** plt

4

**import** seaborn **as** sns

6

**%**matplotlib inline

7

**from** scipy **import** stats

8

**import** statsmodels.api **as** sm

9

**from** sklearn.metrics **import** confusion\_matrix ,classification\_report,ConfusionMatrixDisplay,accuracy\_score

10

**from** statsmodels.stats.outliers\_influence **import** variance\_inflation\_factor

In [117]:

1

med\_data **=** pd.read\_csv('medical\_clean.csv')

In [118]:

1

med\_data.columns

Out[118]:

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 [119]:

1

med\_data **=** med\_data.drop(columns**=**['CaseOrder', 'Customer\_id', 'Interaction', 'UID','Job', 'Children', 'Age', 'Income', 'Marital', 'Gender', 'City', 'State',

2

'County', 'Zip', 'Lat', 'Lng','Population', 'Area', 'TimeZone' , 'Item1', 'Item2', 'Item3', 'Item4',

3

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

In [120]:

1

med\_data.columns

Out[120]:

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 [121]:

1

med\_data.dtypes

Out[121]:

ReAdmis object  
VitD\_levels float64  
Doc\_visits int64  
Full\_meals\_eaten int64  
vitD\_supp int64  
Soft\_drink object  
Initial\_admin object  
HighBlood object  
Stroke object  
Complication\_risk object  
Overweight object  
Arthritis object  
Diabetes object  
Hyperlipidemia object  
BackPain object  
Anxiety object  
Allergic\_rhinitis object  
Reflux\_esophagitis object  
Asthma object  
Services object  
Initial\_days float64  
TotalCharge float64  
Additional\_charges float64  
dtype: object

In [122]:

1

med\_data.isnull().sum() *# checking for missing data*

Out[122]:

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 [123]:

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

*# creating dummy variables*

8

**for** column **in** categorical\_columns:

9

med\_data[column] **=** med\_data[column].astype('category').cat.codes

In [124]:

1

med\_data **=** pd.get\_dummies(med\_data, columns**=**['Services','Complication\_risk','Initial\_admin'], drop\_first**=True**)

In [125]:

1

*# checking for outliers.*

2

med\_data.std()

Out[125]:

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 26.309341  
TotalCharge 2180.393838  
Additional\_charges 6542.601544  
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 [126]:

1

med\_data ['TotalCharge\_z']**=**stats.zscore(med\_data['TotalCharge'])

In [127]:

1

med\_data\_outliers\_TotalCharge **=** med\_data.query('TotalCharge\_z > 3 | TotalCharge\_z< -3')

In [128]:

1

med\_data ['Additional\_charges\_z']**=**stats.zscore(med\_data['Additional\_charges'])

In [129]:

1

med\_data\_outliers\_Additional\_charges **=** med\_data.query('Additional\_charges\_z > 3 | Additional\_charges\_z< -3')

In [130]:

1

med\_data ['Initial\_days\_z'] **=** stats.zscore(med\_data['Initial\_days'])

In [131]:

1

med\_data\_outliers\_Initial\_days **=** med\_data.query('Initial\_days\_z > 3 | Initial\_days\_z< -3')

In [132]:

1

med\_data.std() *# checking to see if outliers were treated.*

Out[132]:

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 26.309341  
TotalCharge 2180.393838  
Additional\_charges 6542.601544  
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  
TotalCharge\_z 1.000050  
Additional\_charges\_z 1.000050  
Initial\_days\_z 1.000050  
dtype: float64

med\_data['ReAdmis'].value\_counts()

In [133]:

1

med\_data['VitD\_levels'].describe()

Out[133]:

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 [134]:

1

med\_data['Doc\_visits'].describe()

Out[134]:

count 10000.000000  
mean 4.012200  
std 1.045734  
min 0.000000  
25% 3.000000  
50% 4.000000  
75% 5.000000  
max 8.000000  
Name: Doc\_visits, dtype: float64

In [135]:

1

med\_data['Full\_meals\_eaten'].describe()

Out[135]:

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 [136]:

1

med\_data['vitD\_supp'].value\_counts()

Out[136]:

vitD\_supp  
0 6702  
1 2684  
2 544  
3 64  
4 5  
5 1  
Name: count, dtype: int64

In [137]:

1

med\_data['Initial\_admin\_Emergency Admission'].value\_counts()

Out[137]:

Initial\_admin\_Emergency Admission  
True 5060  
False 4940  
Name: count, dtype: int64

In [138]:

1

med\_data['Initial\_admin\_Observation Admission'].value\_counts()

Out[138]:

Initial\_admin\_Observation Admission  
False 7564  
True 2436  
Name: count, dtype: int64

In [139]:

1

med\_data['Complication\_risk\_Low'].value\_counts()

Out[139]:

Complication\_risk\_Low  
False 7875  
True 2125  
Name: count, dtype: int64

In [140]:

1

med\_data['Complication\_risk\_Medium'].value\_counts()

Out[140]:

Complication\_risk\_Medium  
False 5483  
True 4517  
Name: count, dtype: int64

In [141]:

1

med\_data['Services\_CT Scan'].value\_counts()

Out[141]:

Services\_CT Scan  
False 8775  
True 1225  
Name: count, dtype: int64

In [142]:

1

med\_data['Services\_Intravenous'].value\_counts()

Out[142]:

Services\_Intravenous  
False 6870  
True 3130  
Name: count, dtype: int64

In [143]:

1

med\_data['Services\_MRI'].value\_counts()

Out[143]:

Services\_MRI  
False 9620  
True 380  
Name: count, dtype: int64

In [ ]:

1

In [144]:

1

med\_data['Overweight'].value\_counts()

Out[144]:

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

In [145]:

1

med\_data['Arthritis'].value\_counts()

Out[145]:

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

In [146]:

1

med\_data['Diabetes'].value\_counts()

Out[146]:

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

In [147]:

1

med\_data['Hyperlipidemia'].value\_counts()

Out[147]:

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

In [148]:

1

med\_data['BackPain'].value\_counts()

Out[148]:

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

In [149]:

1

med\_data['Anxiety'].value\_counts()

Out[149]:

Anxiety  
0 6785  
1 3215  
Name: count, dtype: int64

In [150]:

1

med\_data['Allergic\_rhinitis'].value\_counts()

Out[150]:

Allergic\_rhinitis  
0 6059  
1 3941  
Name: count, dtype: int64

In [151]:

1

med\_data['Reflux\_esophagitis'].value\_counts()

Out[151]:

Reflux\_esophagitis  
0 5865  
1 4135  
Name: count, dtype: int64

In [152]:

1

med\_data['Asthma'].value\_counts()

Out[152]:

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

In [153]:

1

med\_data['TotalCharge'].describe()

Out[153]:

count 10000.000000  
mean 5312.172769  
std 2180.393838  
min 1938.312067  
25% 3179.374015  
50% 5213.952000  
75% 7459.699750  
max 9180.728000  
Name: TotalCharge, dtype: float64

In [154]:

1

med\_data['Additional\_charges'].describe()

Out[154]:

count 10000.000000  
mean 12934.528587  
std 6542.601544  
min 3125.703000  
25% 7986.487755  
50% 11573.977735  
75% 15626.490000  
max 30566.070000  
Name: Additional\_charges, dtype: float64

In [155]:

1

med\_data['Asthma'].value\_counts()

Out[155]:

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

In [156]:

1

*#Describtibve ananlysis*

2

med\_data.describe()

Out[156]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **ReAdmis** | **VitD\_levels** | **Doc\_visits** | **Full\_meals\_eaten** | **vitD\_supp** | **Soft\_drink** | **HighBlood** | **Stroke** | **Overweight** | **Arthritis** | **...** | **Anxiety** | **Allergic\_rhinitis** | **Reflux\_esophagitis** | **Asthma** | **Initial\_days** | **TotalCharge** | **Additional\_charges** | **TotalCharge\_z** | **Additional\_charges\_z** | **Initial\_days\_z** |
| **count** | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | 10000.000000 | ... | 10000.000000 | 10000.000000 | 10000.000000 | 10000.00000 | 10000.000000 | 10000.000000 | 10000.000000 | 1.000000e+04 | 1.000000e+04 | 1.000000e+04 |
| **mean** | 0.366900 | 4985.993300 | 4.012200 | 1.001400 | 0.398900 | 0.257500 | 0.409000 | 0.199300 | 0.709400 | 0.357400 | ... | 0.321500 | 0.394100 | 0.413500 | 0.28930 | 34.455299 | 5312.172769 | 12934.528587 | 1.364242e-16 | -2.238210e-17 | -1.818989e-16 |
| **std** | 0.481983 | 2878.750357 | 1.045734 | 1.008117 | 0.628505 | 0.437279 | 0.491674 | 0.399494 | 0.454062 | 0.479258 | ... | 0.467076 | 0.488681 | 0.492486 | 0.45346 | 26.309341 | 2180.393838 | 6542.601544 | 1.000050e+00 | 1.000050e+00 | 1.000050e+00 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 1.001981 | 1938.312067 | 3125.703000 | -1.547440e+00 | -1.499299e+00 | -1.271601e+00 |
| **25%** | 0.000000 | 2493.750000 | 3.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 7.896215 | 3179.374015 | 7986.487755 | -9.782202e-01 | -7.563181e-01 | -1.009543e+00 |
| **50%** | 0.000000 | 4984.500000 | 4.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 35.836244 | 5213.952000 | 11573.977735 | -4.504951e-02 | -2.079630e-01 | 5.249140e-02 |
| **75%** | 1.000000 | 7478.250000 | 5.000000 | 2.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 1.000000 | 1.000000 | ... | 1.000000 | 1.000000 | 1.000000 | 1.00000 | 61.161020 | 7459.699750 | 15626.490000 | 9.849754e-01 | 4.114718e-01 | 1.015117e+00 |
| **max** | 1.000000 | 9975.000000 | 8.000000 | 7.000000 | 5.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | ... | 1.000000 | 1.000000 | 1.000000 | 1.00000 | 71.981490 | 9180.728000 | 30566.070000 | 1.774335e+00 | 2.695017e+00 | 1.426416e+00 |

8 rows × 23 columns

# **C3. Univariate and Bivariate visualizations**

before ploting the graphs, catagorical variables will converted to numerical values as follows

In [157]:

1

med\_data.dtypes

Out[157]:

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  
TotalCharge\_z float64  
Additional\_charges\_z float64  
Initial\_days\_z float64  
dtype: object

In [158]:

1

med\_data **=** med\_data.astype(int)

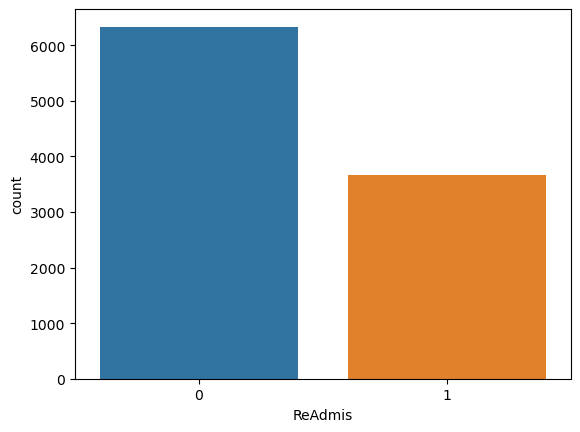
In [159]:

1

sns.countplot(x**=**'ReAdmis', data **=**med\_data)

Out[159]:

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



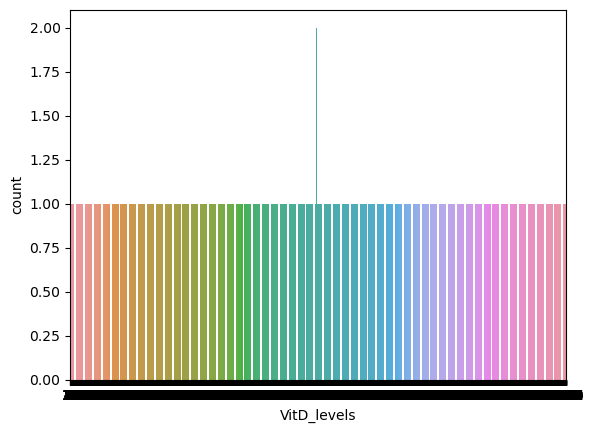
In [160]:

1

sns.countplot(x**=**'VitD\_levels', data **=**med\_data)

Out[160]:

<Axes: xlabel='VitD\_levels', ylabel='count'>



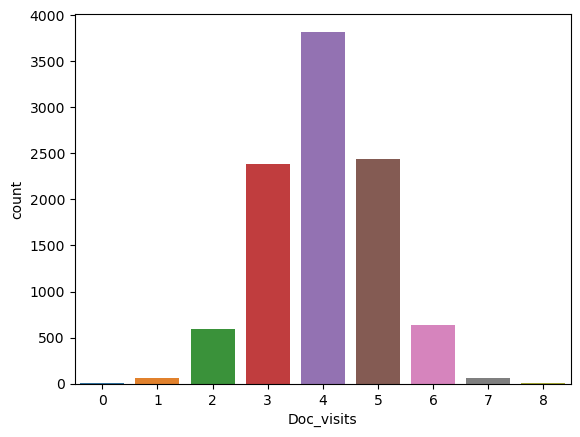
In [161]:

1

sns.countplot(x**=**'Doc\_visits', data **=**med\_data)

Out[161]:

<Axes: xlabel='Doc\_visits', ylabel='count'>



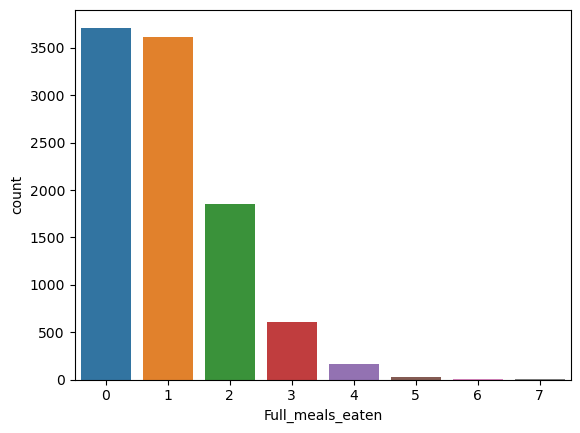
In [162]:

1

sns.countplot(x**=**'Full\_meals\_eaten', data **=**med\_data)

Out[162]:

<Axes: xlabel='Full\_meals\_eaten', ylabel='count'>



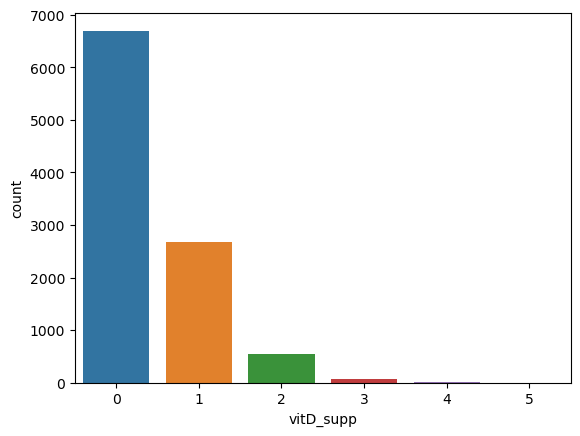
In [163]:

1

sns.countplot(x**=**'vitD\_supp', data **=**med\_data)

Out[163]:

<Axes: xlabel='vitD\_supp', ylabel='count'>



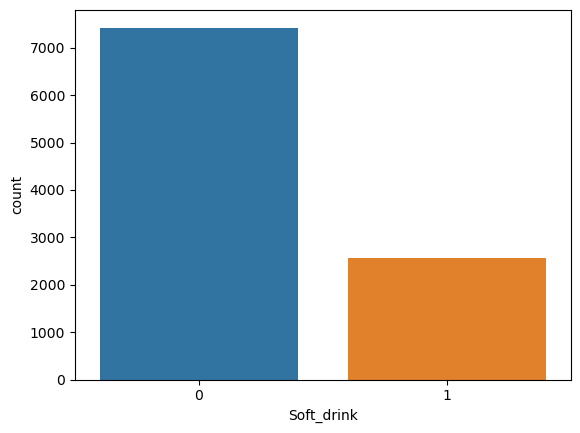
In [164]:

1

sns.countplot(x**=**'Soft\_drink', data **=**med\_data)

Out[164]:

<Axes: xlabel='Soft\_drink', ylabel='count'>



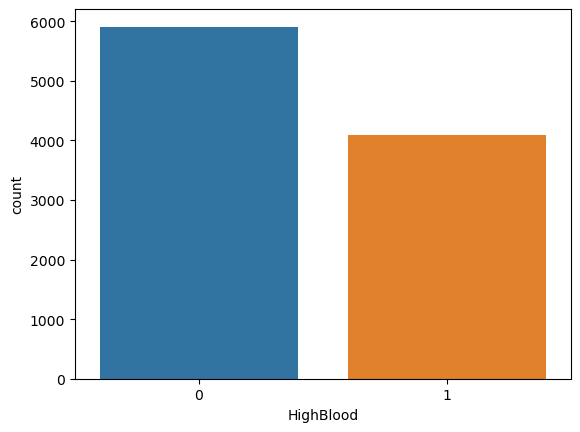
In [165]:

1

sns.countplot(x**=**'HighBlood', data **=**med\_data)

Out[165]:

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



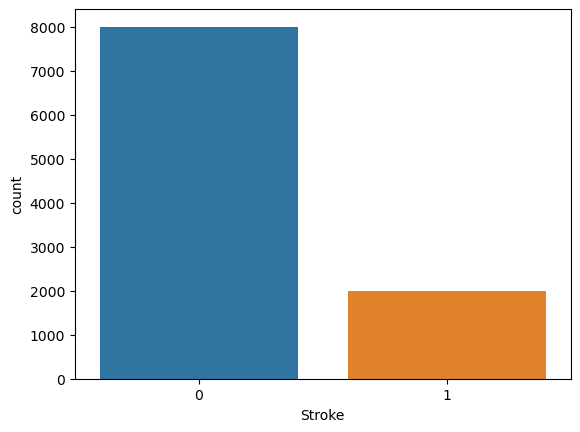
In [166]:

1

sns.countplot(x**=**'Stroke', data **=**med\_data)

Out[166]:

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



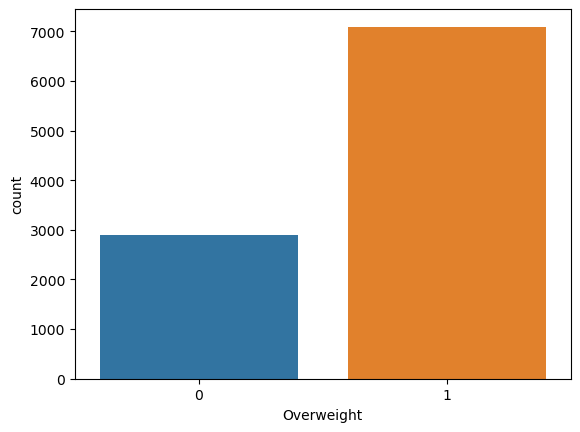
In [167]:

1

sns.countplot(x**=**'Overweight', data **=**med\_data)

Out[167]:

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



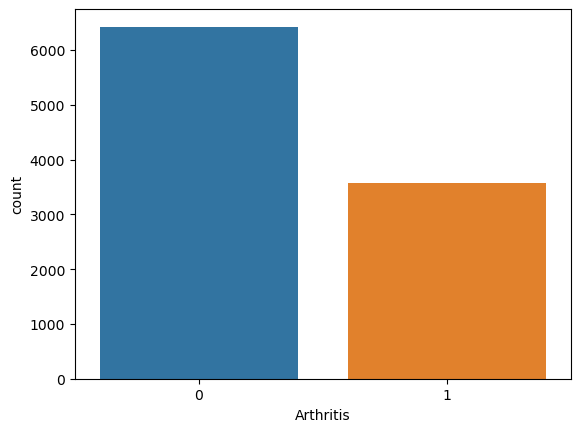
In [168]:

1

sns.countplot(x**=**'Arthritis', data **=**med\_data)

Out[168]:

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



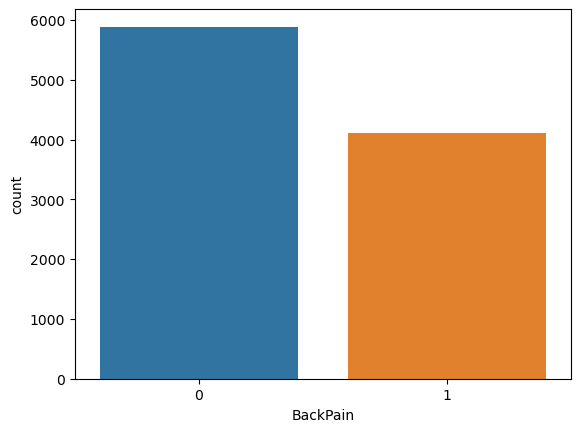
In [169]:

1

sns.countplot(x**=**'BackPain', data **=**med\_data)

Out[169]:

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



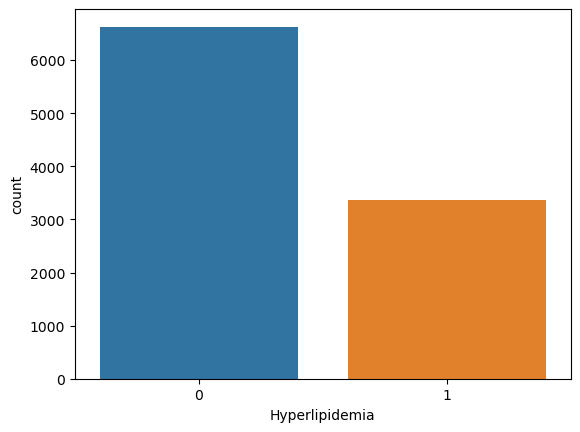
In [170]:

1

sns.countplot(x**=**'Hyperlipidemia', data **=**med\_data)

Out[170]:

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



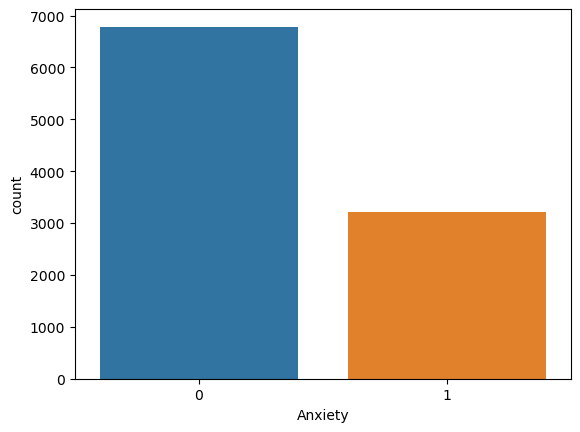
In [171]:

1

sns.countplot(x**=**'Anxiety', data **=**med\_data)

Out[171]:

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



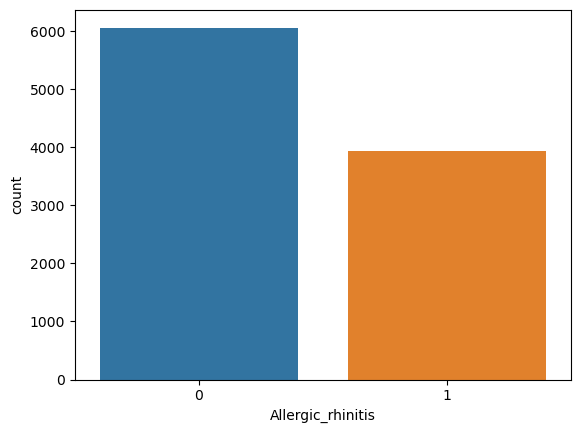
In [172]:

1

sns.countplot(x**=**'Allergic\_rhinitis', data **=**med\_data)

Out[172]:

<Axes: xlabel='Allergic\_rhinitis', ylabel='count'>



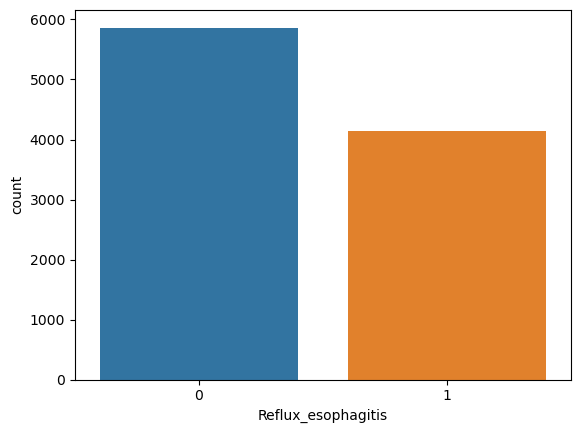
In [173]:

1

sns.countplot(x**=**'Reflux\_esophagitis', data **=**med\_data)

Out[173]:

<Axes: xlabel='Reflux\_esophagitis', ylabel='count'>



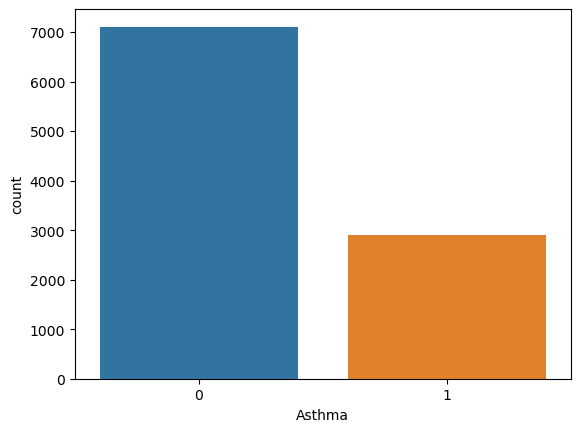
In [174]:

1

sns.countplot(x**=**'Asthma', data **=**med\_data)

Out[174]:

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



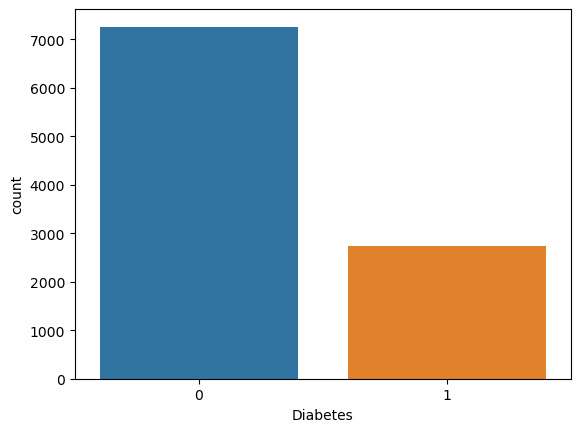
In [175]:

1

sns.countplot(x**=**'Diabetes', data **=**med\_data)

Out[175]:

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



In [176]:

1

med\_data.columns

Out[176]:

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', 'TotalCharge\_z',  
 'Additional\_charges\_z', 'Initial\_days\_z'],  
 dtype='object')

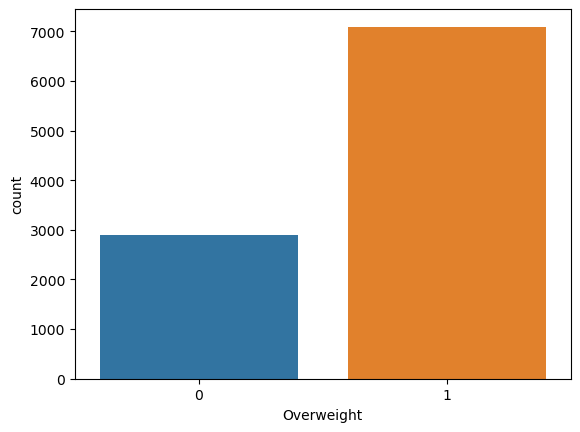
In [177]:

1

sns.countplot(x**=**'Overweight', data **=**med\_data)

Out[177]:

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



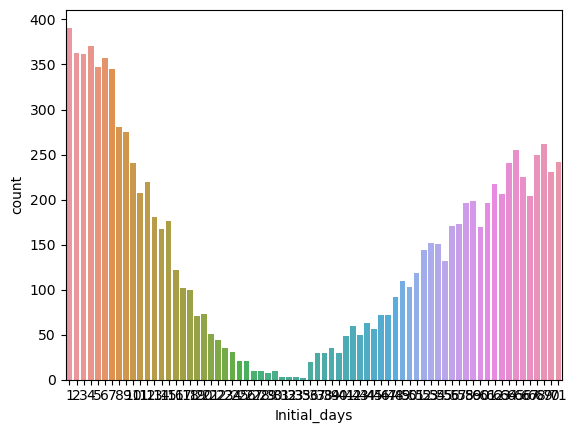
In [178]:

1

sns.countplot(x**=**'Initial\_days', data **=**med\_data)

Out[178]:

<Axes: xlabel='Initial\_days', ylabel='count'>



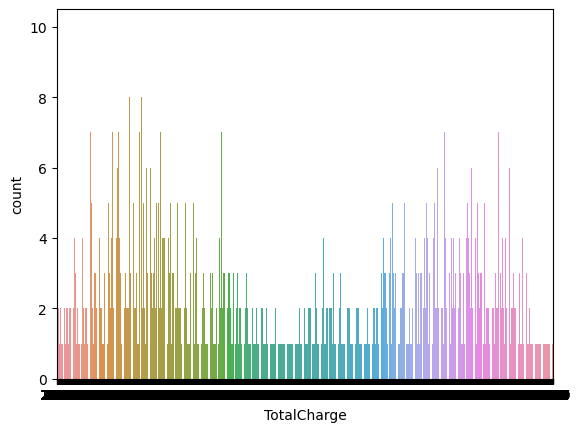
In [179]:

1

sns.countplot(x**=**'TotalCharge', data **=**med\_data)

Out[179]:

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



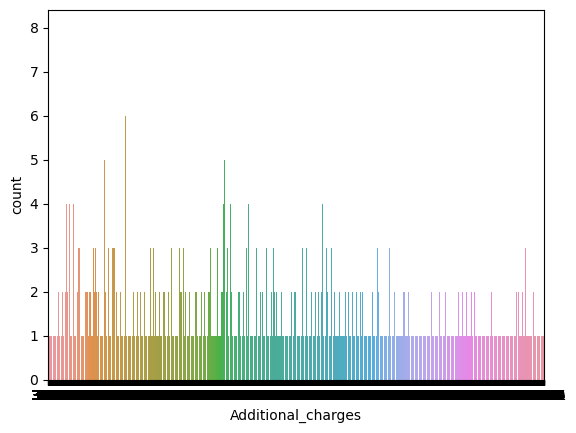
In [180]:

1

sns.countplot(x**=**'Additional\_charges', data **=**med\_data)

Out[180]:

<Axes: xlabel='Additional\_charges', ylabel='count'>



In [181]:

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

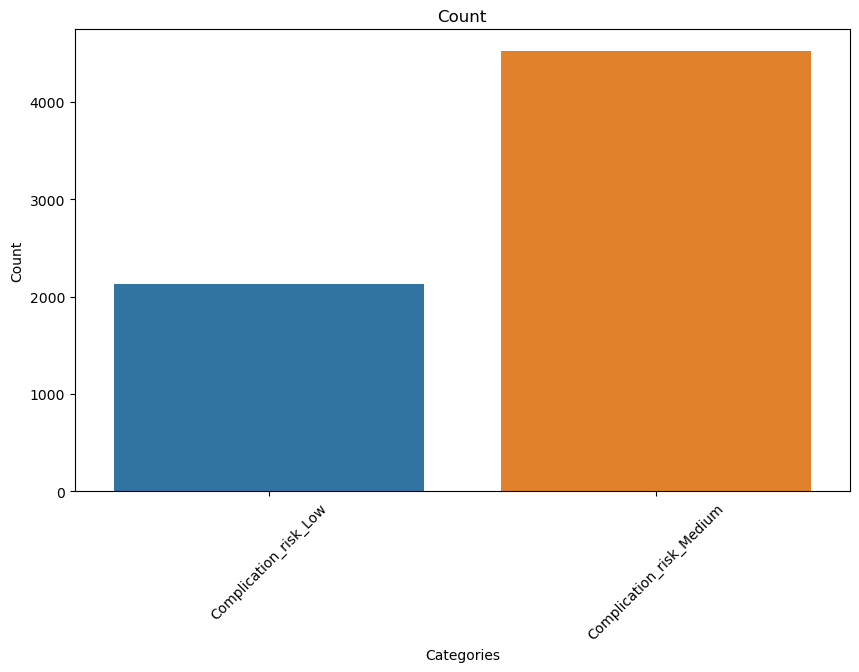
plt.ylabel('Count')

8

plt.title('Count')

9

plt.show()



In [182]:

1

2

selected\_columns **=** ['Services\_CT Scan', 'Services\_Intravenous',

3

'Services\_MRI',]

4

5

category\_counts **=** med\_data[selected\_columns].sum()

6

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

7

sns.barplot(x**=**category\_counts.index, y**=**category\_counts.values)

8

plt.xticks(rotation**=**45)

9

plt.xlabel('Categories')

10

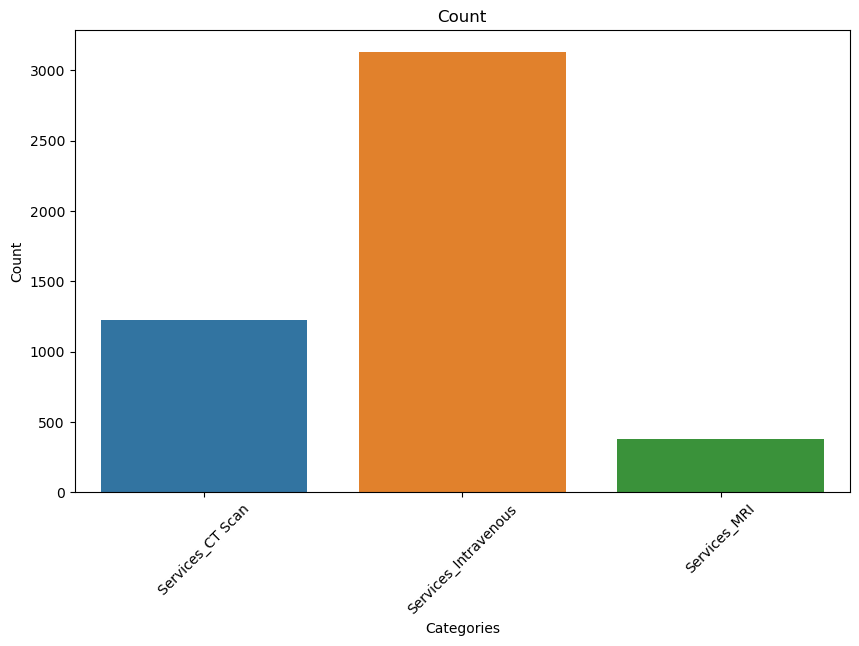
plt.ylabel('Count')

11

plt.title('Count')

12

plt.show()



In [183]:

1

selected\_columns **=** ['Initial\_admin\_Emergency Admission','Initial\_admin\_Observation Admission']

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

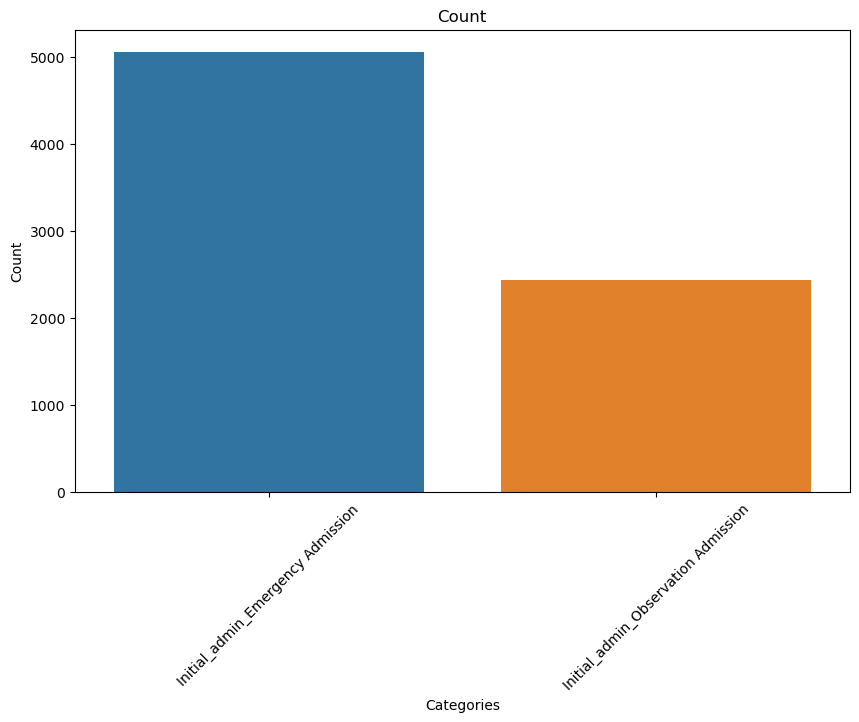
plt.ylabel('Count')

8

plt.title('Count')

9

plt.show()



In [ ]:

1

*#Barient visiulization*

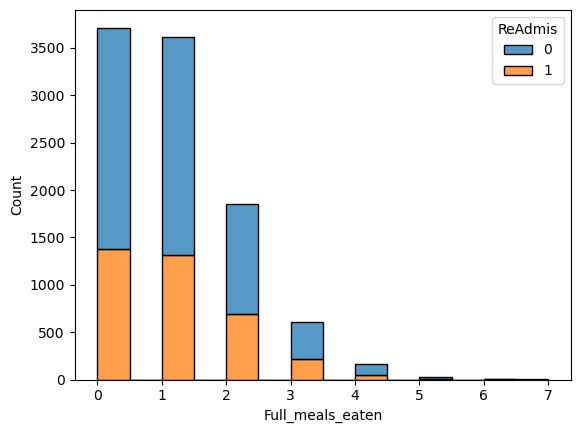
In [184]:

1

sns.histplot(binwidth**=**0.5, x**=** "Full\_meals\_eaten", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[184]:

<Axes: xlabel='Full\_meals\_eaten', ylabel='Count'>



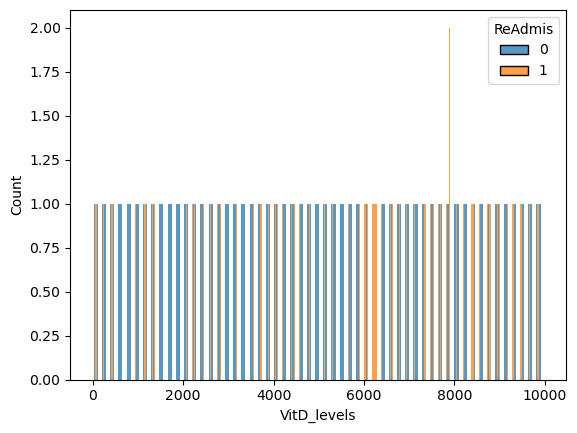
In [185]:

1

sns.histplot(binwidth**=**0.5, x**=** "VitD\_levels", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count",multiple**=**"stack")

Out[185]:

<Axes: xlabel='VitD\_levels', ylabel='Count'>



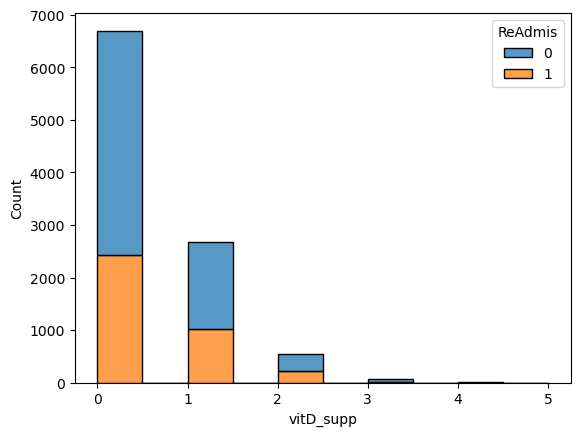
In [186]:

1

sns.histplot(binwidth**=**0.5, x**=** "vitD\_supp", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count",multiple**=**"stack")

Out[186]:

<Axes: xlabel='vitD\_supp', ylabel='Count'>



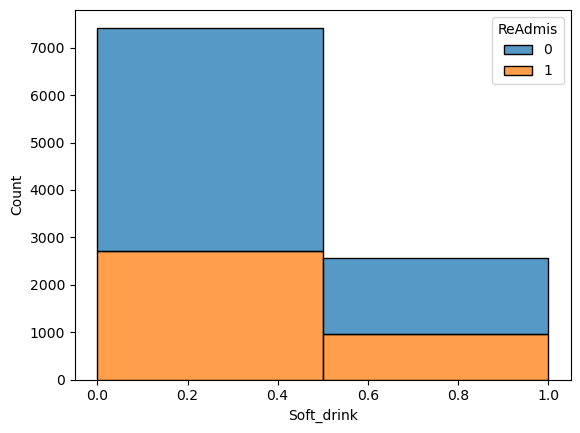
In [187]:

1

sns.histplot(binwidth**=**0.5, x**=** "Soft\_drink", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[187]:

<Axes: xlabel='Soft\_drink', ylabel='Count'>



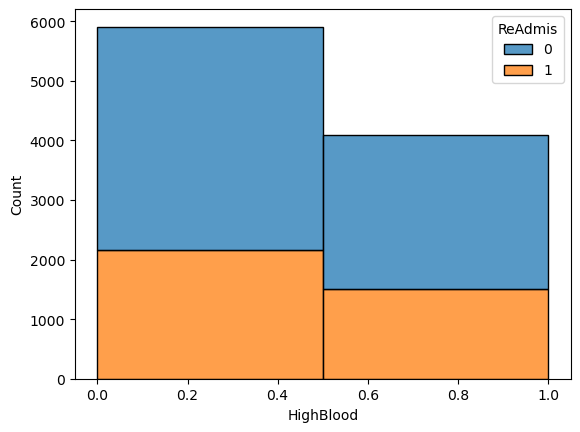
In [188]:

1

sns.histplot(binwidth**=**0.5, x**=** "HighBlood", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[188]:

<Axes: xlabel='HighBlood', ylabel='Count'>



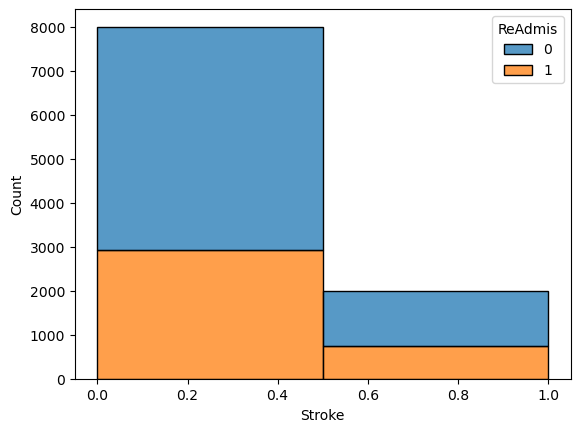
In [189]:

1

sns.histplot(binwidth**=**0.5, x**=** "Stroke", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[189]:

<Axes: xlabel='Stroke', ylabel='Count'>



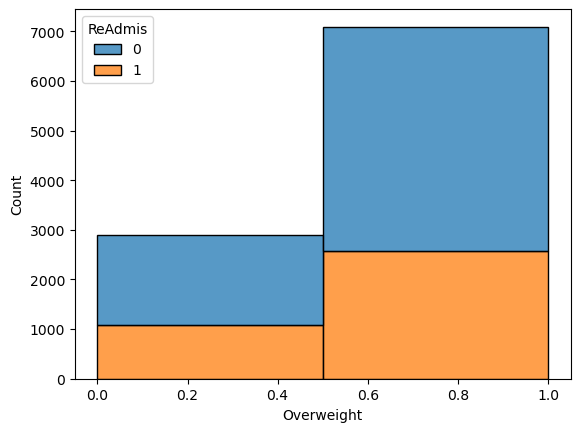
In [190]:

1

sns.histplot(binwidth**=**0.5, x**=** "Overweight", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count",multiple**=**"stack")

Out[190]:

<Axes: xlabel='Overweight', ylabel='Count'>



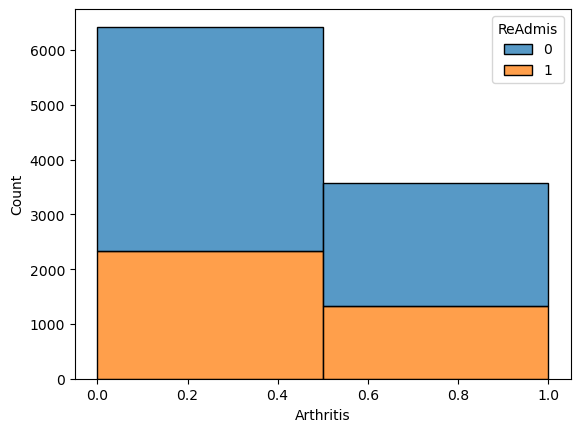
In [191]:

1

sns.histplot(binwidth**=**0.5, x**=** "Arthritis", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[191]:

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



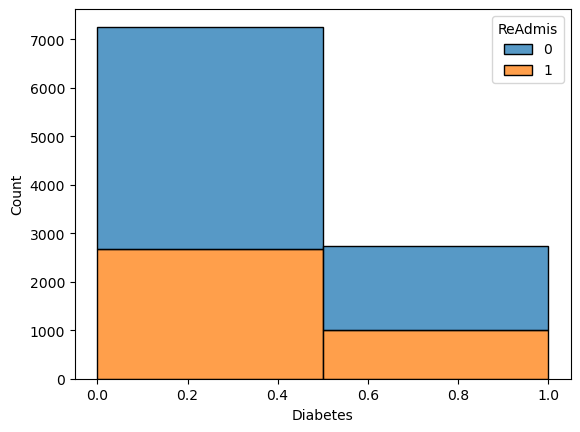
In [192]:

1

sns.histplot(binwidth**=**0.5, x**=** "Diabetes", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[192]:

<Axes: xlabel='Diabetes', ylabel='Count'>



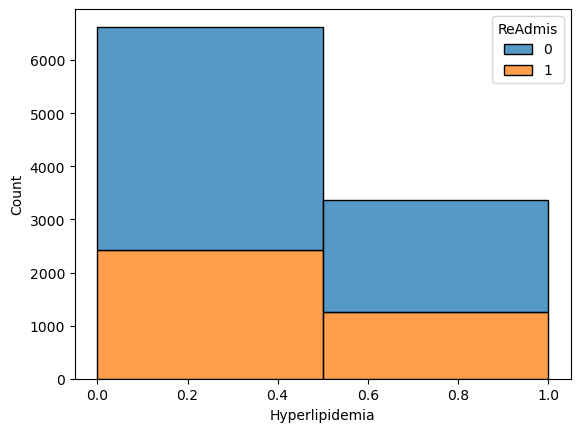
In [193]:

1

sns.histplot(binwidth**=**0.5, x**=** "Hyperlipidemia", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count",multiple**=**"stack")

Out[193]:

<Axes: xlabel='Hyperlipidemia', ylabel='Count'>



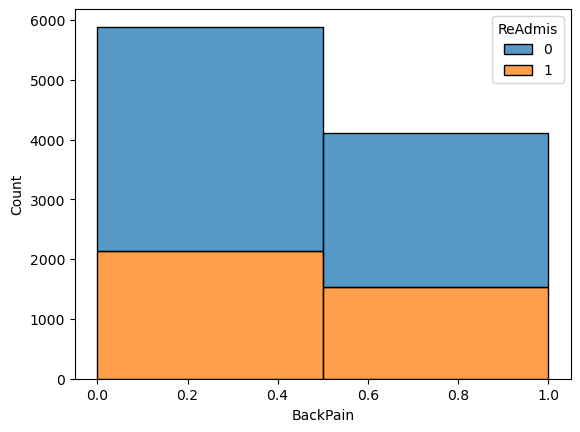
In [194]:

1

sns.histplot(binwidth**=**0.5, x**=** "BackPain", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[194]:

<Axes: xlabel='BackPain', ylabel='Count'>



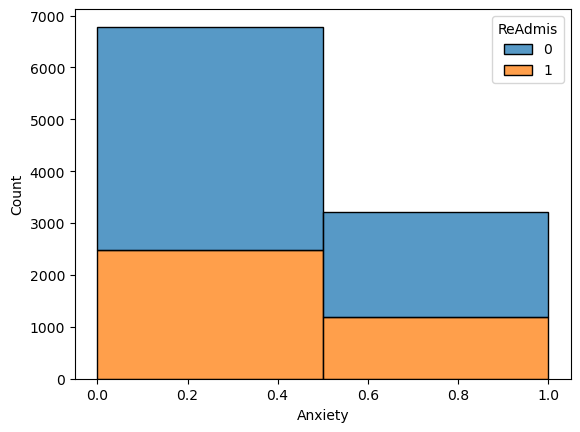
In [195]:

1

sns.histplot(binwidth**=**0.5, x**=** "Anxiety", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[195]:

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



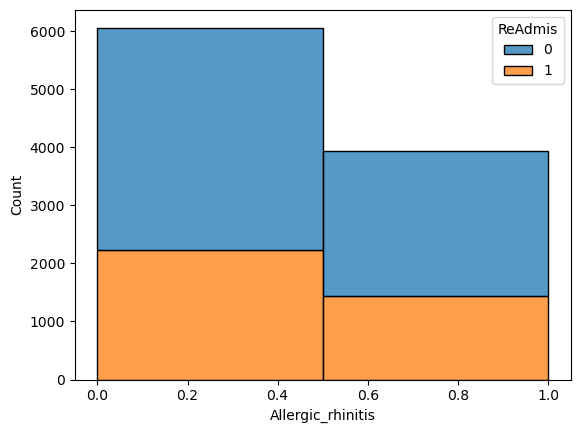
In [196]:

1

sns.histplot(binwidth**=**0.5, x**=** "Allergic\_rhinitis", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[196]:

<Axes: xlabel='Allergic\_rhinitis', ylabel='Count'>



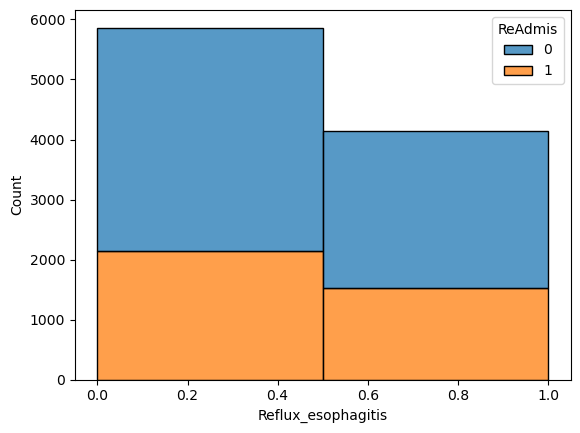
In [197]:

1

sns.histplot(binwidth**=**0.5, x**=** "Reflux\_esophagitis", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[197]:

<Axes: xlabel='Reflux\_esophagitis', ylabel='Count'>



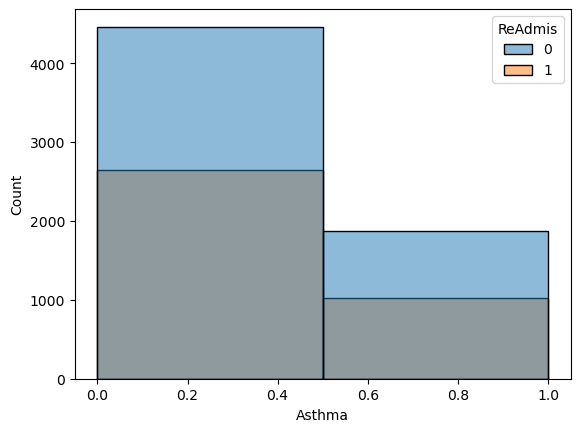
In [198]:

1

sns.histplot(binwidth**=**0.5, x**=** "Asthma", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count")

Out[198]:

<Axes: xlabel='Asthma', ylabel='Count'>



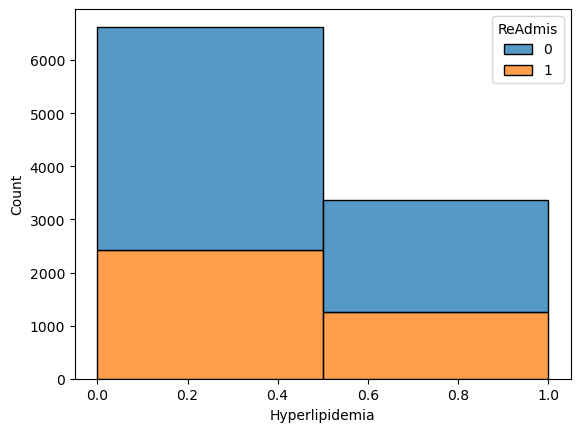
In [199]:

1

sns.histplot(binwidth**=**0.5, x**=** "Hyperlipidemia", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

Out[199]:

<Axes: xlabel='Hyperlipidemia', ylabel='Count'>



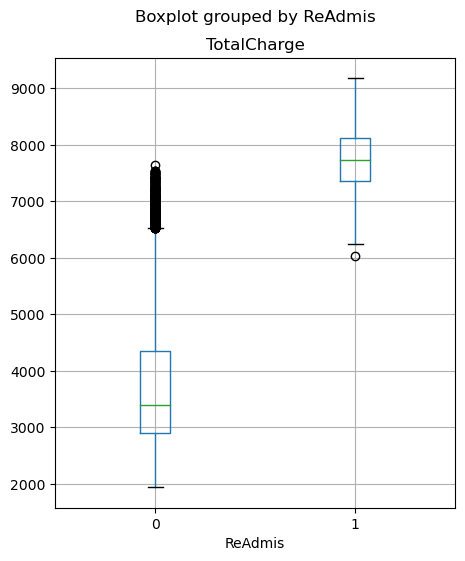
In [200]:

1

med\_data.boxplot(column **=**'TotalCharge', by**=** 'ReAdmis', figsize **=** ( 5,6))

Out[200]:

<Axes: title={'center': 'TotalCharge'}, xlabel='ReAdmis'>



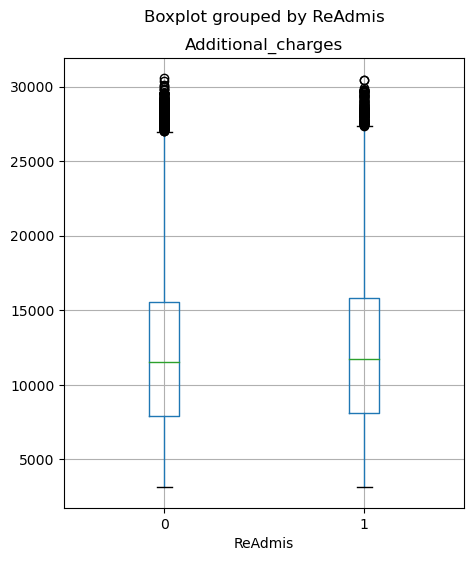
In [201]:

1

med\_data.boxplot(column **=**'Additional\_charges', by**=** 'ReAdmis', figsize **=** ( 5,6))

Out[201]:

<Axes: title={'center': 'Additional\_charges'}, xlabel='ReAdmis'>



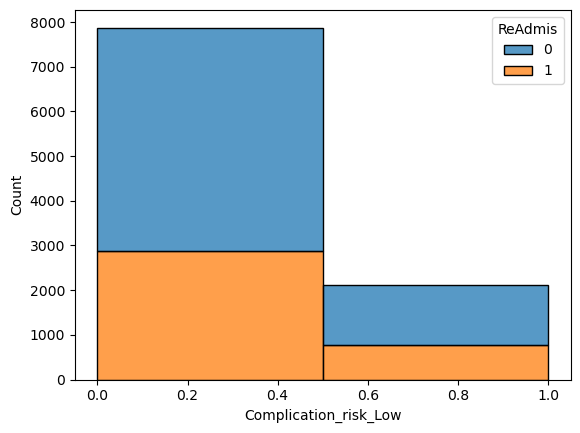
In [202]:

1

sns.histplot(binwidth**=**0.5, x**=** "Complication\_risk\_Low", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

2

plt.show();



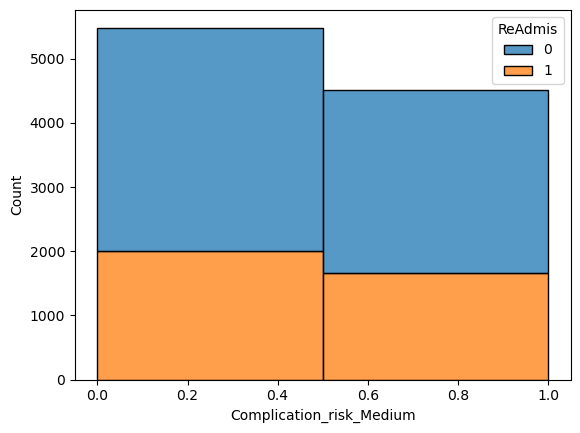
In [203]:

1

sns.histplot(binwidth**=**0.5, x**=** "Complication\_risk\_Medium", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

2

plt.show();



In [204]:

1

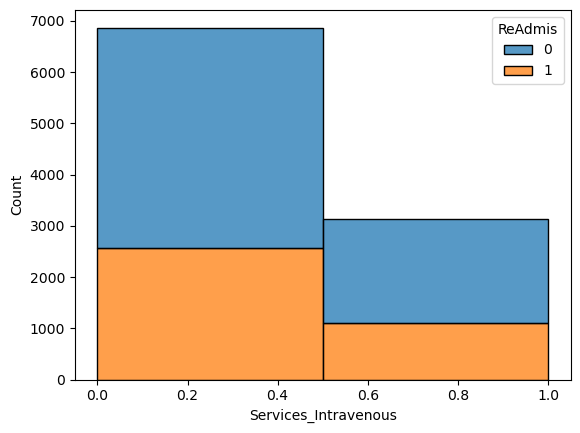
sns.histplot(binwidth**=**0.5, x**=** "Services\_Intravenous", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

2

plt.show();

3

4



In [205]:

1

sns.histplot(binwidth**=**0.5, x**=** "Services\_MRI", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

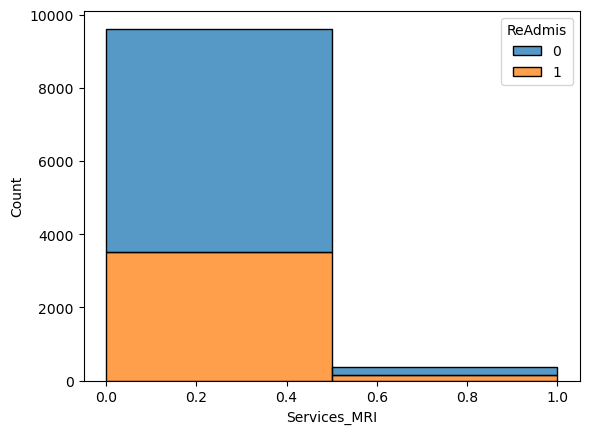
2

plt.show();

3

4

5



In [206]:

1

sns.histplot(binwidth**=**0.5, x**=** "Services\_CT Scan", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

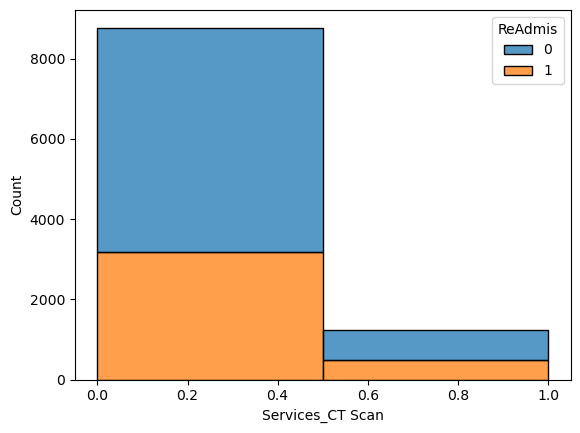
2

plt.show();

3

4

5



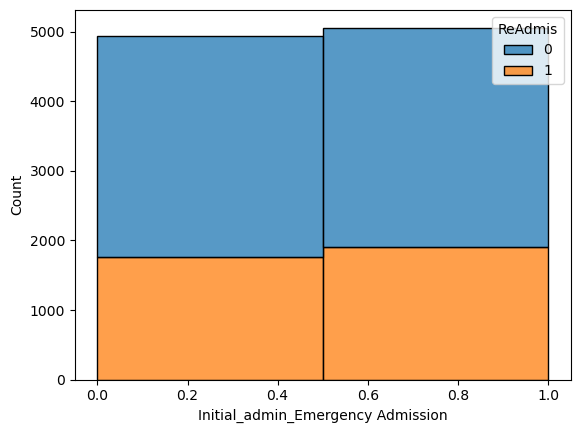
In [207]:

1

sns.histplot(binwidth**=**0.5, x**=** "Initial\_admin\_Emergency Admission", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

2

plt.show();



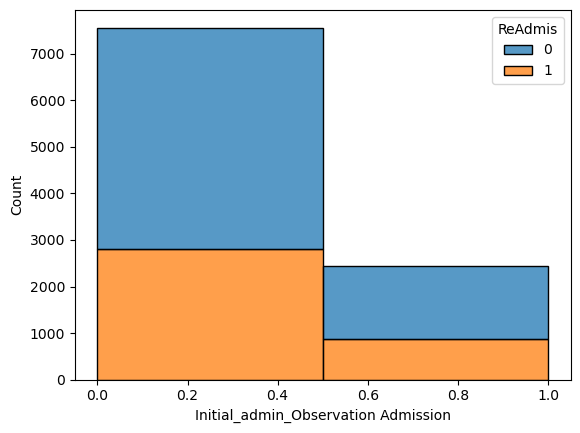
In [208]:

1

sns.histplot(binwidth**=**0.5, x**=** "Initial\_admin\_Observation Admission", hue**=**"ReAdmis", data**=**med\_data, stat**=**"count", multiple**=**"stack")

2

plt.show();



In [209]:

1

*#C4 Describe your data transformation goals that align with your research question and the steps used to transform the data to achieve the goals, including the annotated code.*

As stated 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. 22 independent variables are prepared for the initial model. All the categorical variables are converted to a numerical value. for categorical variables with more than two potions (i.e. 'Services','Complication\_risk','Initial\_admin'), one hot encoding is used to give variables numerical values.

In [210]:

1

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

2

3

med\_data.to\_csv('MSDA208\_Task2\_prepared\_Data.csv')

# **Part IV: Model Comparison and Analysis**

In [211]:

1

*# D1. Initial model*

The initial model has 24 variables, while the reduced model has 17. The reduced model is less complex, suggesting a closer representation.

Goodness of Fit: Both models have high Pseudo R-squared values (around 0.949), indicating a good fit to the data.

Log-Likelihood: The log-likelihood values show that both models have negative log-likelihood values, but the reduced model has a slightly lower value, suggesting a marginal improvement in fit.

The LLR p-value is 0.000 for both models, indicating that both models provide a statistically significant improvement over the null model (LL-Null).

The reduced model, despite having fewer predictors, performs similarly in terms of goodness of fit and statistical significance. This suggests that the removed predictors in the reduced model may not contribute significantly to explaining the variability in the dependent variable.

In [212]:

1

y **=** med\_data['ReAdmis']

3

independent\_vars **=** ['VitD\_levels', 'Doc\_visits',

5

'Full\_meals\_eaten', 'vitD\_supp','Soft\_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis',

6

'Diabetes','Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis',

7

'Reflux\_esophagitis', 'Asthma','Services\_CT Scan','TotalCharge' ,

8

'Services\_Intravenous', 'Services\_MRI', 'Complication\_risk\_Low',

9

'Complication\_risk\_Medium', 'Initial\_admin\_Emergency Admission',

10

'Initial\_admin\_Observation Admission']

11

12

X **=** med\_data[independent\_vars]

13

X **=** sm.add\_constant(med\_data[independent\_vars])

In [213]:

1

model **=** sm.Logit(y, X)

2

results **=** model.fit()

3

4

Optimization terminated successfully.  
 Current function value: 0.033252  
 Iterations 14

In [214]:

1

2

print(results.summary())

Logit Regression Results   
==============================================================================  
Dep. Variable: ReAdmis No. Observations: 10000  
Model: Logit Df Residuals: 9975  
Method: MLE Df Model: 24  
Date: Mon, 18 Dec 2023 Pseudo R-squ.: 0.9494  
Time: 10:21:34 Log-Likelihood: -332.52  
converged: True LL-Null: -6572.9  
Covariance Type: nonrobust LLR p-value: 0.000  
=======================================================================================================  
 coef std err z P>|z| [0.025 0.975]  
-------------------------------------------------------------------------------------------------------  
const -122.8571 6.934 -17.719 0.000 -136.447 -109.268  
VitD\_levels 1.934e-05 3.39e-05 0.570 0.569 -4.71e-05 8.58e-05  
Doc\_visits 0.0086 0.097 0.089 0.929 -0.182 0.199  
Full\_meals\_eaten 0.0586 0.102 0.572 0.567 -0.142 0.259  
vitD\_supp -0.1172 0.162 -0.722 0.470 -0.435 0.201  
Soft\_drink 0.3177 0.241 1.321 0.187 -0.154 0.789  
HighBlood -1.1240 0.217 -5.181 0.000 -1.549 -0.699  
Stroke 1.6798 0.267 6.285 0.000 1.156 2.204  
Overweight -0.3114 0.227 -1.373 0.170 -0.756 0.133  
Arthritis -2.6487 0.261 -10.137 0.000 -3.161 -2.137  
Diabetes -0.9244 0.233 -3.974 0.000 -1.380 -0.469  
Hyperlipidemia -1.4533 0.231 -6.303 0.000 -1.905 -1.001  
BackPain -1.2796 0.217 -5.907 0.000 -1.704 -0.855  
Anxiety -2.6231 0.265 -9.884 0.000 -3.143 -2.103  
Allergic\_rhinitis -1.4348 0.222 -6.450 0.000 -1.871 -0.999  
Reflux\_esophagitis -1.4811 0.224 -6.613 0.000 -1.920 -1.042  
Asthma -1.3766 0.233 -5.897 0.000 -1.834 -0.919  
Services\_CT Scan 1.5822 0.360 4.396 0.000 0.877 2.288  
TotalCharge 0.0182 0.001 17.738 0.000 0.016 0.020  
Services\_Intravenous 0.0355 0.227 0.157 0.875 -0.409 0.480  
Services\_MRI 2.7430 0.502 5.463 0.000 1.759 3.727  
Complication\_risk\_Low 5.6278 0.413 13.613 0.000 4.818 6.438  
Complication\_risk\_Medium 7.1450 0.467 15.303 0.000 6.230 8.060  
Initial\_admin\_Emergency Admission -6.7873 0.453 -14.997 0.000 -7.674 -5.900  
Initial\_admin\_Observation Admission 0.8133 0.279 2.914 0.004 0.266 1.360  
=======================================================================================================  
  
Possibly complete quasi-separation: A fraction 0.82 of observations can be  
perfectly predicted. This might indicate that there is complete  
quasi-separation. In this case some parameters will not be identified.

In [215]:

1

y **=** med\_data['ReAdmis']

2

3

4

independent\_vars **=**['VitD\_levels', 'Doc\_visits', 'Full\_meals\_eaten',

5

'vitD\_supp','Soft\_drink', 'HighBlood', 'Stroke', 'Overweight', 'Arthritis',

6

'Diabetes','Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis',

7

'Reflux\_esophagitis', 'Asthma','Services\_CT Scan','TotalCharge' ,

8

'Services\_Intravenous', 'Services\_MRI', 'Complication\_risk\_Low',

9

'Complication\_risk\_Medium', 'Initial\_admin\_Emergency Admission',

10

'Initial\_admin\_Observation Admission']

11

In [216]:

1

X2 **=** sm.add\_constant(med\_data[independent\_vars])

2

3

In [217]:

1

y\_pred **=** results.predict(X2)

In [218]:

1

y **=** y.astype(int)

2

y\_pred **=** y\_pred.astype(int)

In [219]:

1

print("Unique values in y:", np.unique(y))

2

print("Unique values in y\_pred:", np.unique(y\_pred))

Unique values in y: [0 1]  
Unique values in y\_pred: [0]

In [220]:

1

y\_pred\_binary **=** (y\_pred **>** 0.5).astype(int)

In [221]:

1

classification\_rep **=** classification\_report(y, y\_pred\_binary)

)

In [222]:

1

confusion\_matrix1 **=** confusion\_matrix(y, y\_pred\_binary)

2

print(confusion\_matrix1)

[[6331 0]  
 [3669 0]]

In [223]:

1

print("Confusion Matrix:")

2

print(confusion\_matrix)

3

4

print("\nClassification Report:")

5

print(classification\_rep)

Confusion Matrix:  
<function confusion\_matrix at 0x000002166A8136A0>  
  
Classification Report:  
 precision recall f1-score support  
  
 0 0.63 1.00 0.78 6331  
 1 0.00 0.00 0.00 3669  
  
 accuracy 0.63 10000  
 macro avg 0.32 0.50 0.39 10000  
weighted avg 0.40 0.63 0.49 10000

In [224]:

1

disp **=** ConfusionMatrixDisplay(confusion\_matrix1, display\_labels**=**['Not Readmitted', 'Readmitted'])

2

disp.plot(cmap**=**plt.cm.Blues, values\_format**=**'d')

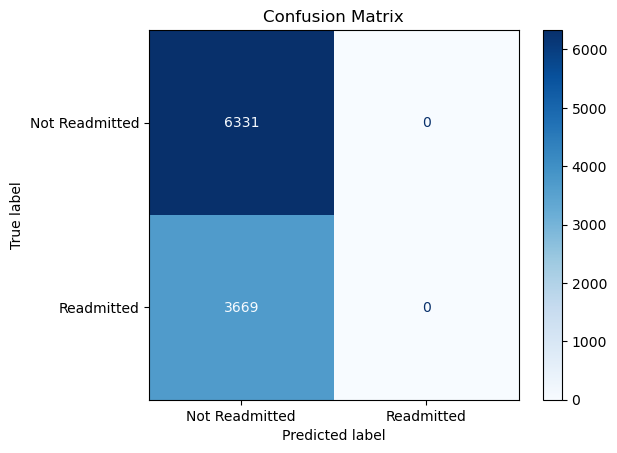
3

4

plt.title("Confusion Matrix")

5

plt.show()



In [225]:

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.*

Using the Backward Elimination method, variables that are contributing the least to the model are removed. In this case, the variables that have p value greater than 0.05 are Overweight ,Services\_Intravenous, VitD\_levels, Doc\_visits ,Full\_meals\_eaten ,vitD\_supp ,Soft\_drink. These variables will be excluded from the reduced model.

In [226]:

1

*#D3. reduced model after removing the columns.*

In [227]:

1

y **=** med\_data['ReAdmis']

2

3

independent\_vars **=** ['HighBlood', 'Stroke', 'Arthritis',

4

'Diabetes','Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis',

5

'Reflux\_esophagitis', 'Asthma','Services\_CT Scan','TotalCharge' ,

6

'Services\_MRI', 'Complication\_risk\_Low',

7

'Complication\_risk\_Medium', 'Initial\_admin\_Emergency Admission',

8

'Initial\_admin\_Observation Admission']

9

10

11

X **=** med\_data[independent\_vars]

12

X **=** sm.add\_constant(med\_data[independent\_vars])

13

14

15

In [228]:

1

model **=** sm.Logit(y, X)

2

results **=** model.fit()

3

4

Optimization terminated successfully.  
 Current function value: 0.033494  
 Iterations 14

In [229]:

1

2

print(results.summary())

Logit Regression Results   
==============================================================================  
Dep. Variable: ReAdmis No. Observations: 10000  
Model: Logit Df Residuals: 9982  
Method: MLE Df Model: 17  
Date: Mon, 18 Dec 2023 Pseudo R-squ.: 0.9490  
Time: 10:21:49 Log-Likelihood: -334.94  
converged: True LL-Null: -6572.9  
Covariance Type: nonrobust LLR p-value: 0.000  
=======================================================================================================  
 coef std err z P>|z| [0.025 0.975]  
-------------------------------------------------------------------------------------------------------  
const -121.2839 6.748 -17.972 0.000 -134.510 -108.057  
HighBlood -1.1247 0.214 -5.248 0.000 -1.545 -0.705  
Stroke 1.6527 0.265 6.245 0.000 1.134 2.171  
Arthritis -2.6285 0.259 -10.164 0.000 -3.135 -2.122  
Diabetes -0.8927 0.230 -3.877 0.000 -1.344 -0.441  
Hyperlipidemia -1.4403 0.229 -6.289 0.000 -1.889 -0.991  
BackPain -1.2510 0.214 -5.834 0.000 -1.671 -0.831  
Anxiety -2.5683 0.261 -9.841 0.000 -3.080 -2.057  
Allergic\_rhinitis -1.4172 0.220 -6.435 0.000 -1.849 -0.986  
Reflux\_esophagitis -1.4472 0.221 -6.558 0.000 -1.880 -1.015  
Asthma -1.3570 0.232 -5.853 0.000 -1.811 -0.903  
Services\_CT Scan 1.5519 0.348 4.463 0.000 0.870 2.233  
TotalCharge 0.0180 0.001 17.977 0.000 0.016 0.020  
Services\_MRI 2.7000 0.489 5.526 0.000 1.742 3.658  
Complication\_risk\_Low 5.5848 0.409 13.663 0.000 4.784 6.386  
Complication\_risk\_Medium 7.0544 0.457 15.443 0.000 6.159 7.950  
Initial\_admin\_Emergency Admission -6.7117 0.445 -15.081 0.000 -7.584 -5.839  
Initial\_admin\_Observation Admission 0.7654 0.276 2.777 0.005 0.225 1.306  
=======================================================================================================  
  
Possibly complete quasi-separation: A fraction 0.82 of observations can be  
perfectly predicted. This might indicate that there is complete  
quasi-separation. In this case some parameters will not be identified.

In [230]:

1

y **=** med\_data['ReAdmis']

In [231]:

1

reduced\_model\_vars **=** ['HighBlood', 'Stroke', 'Arthritis',

2

'Diabetes','Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis',

3

'Reflux\_esophagitis', 'Asthma','Services\_CT Scan','TotalCharge' ,

4

'Services\_MRI', 'Complication\_risk\_Low',

5

'Complication\_risk\_Medium', 'Initial\_admin\_Emergency Admission',

6

'Initial\_admin\_Observation Admission']

7

8

9

In [232]:

1

2

X3 **=** sm.add\_constant(med\_data[reduced\_model\_vars])

In [233]:

1

y\_pred **=** results.predict(X3)

In [234]:

1

y **=** y.astype(int)

2

y\_pred **=** y\_pred.astype(int)

In [235]:

1

print("Unique values in y:", np.unique(y))

2

print("Unique values in y\_pred:", np.unique(y\_pred))

Unique values in y: [0 1]  
Unique values in y\_pred: [0]

In [236]:

1

y\_pred\_binary **=** (y\_pred **>** 0.5).astype(int)

In [237]:

1

classification\_rep **=** classification\_report(y, y\_pred\_binary)

In [238]:

1

confusion\_matrix2 **=** confusion\_matrix(y, y\_pred\_binary)

2

In [239]:

1

print(confusion\_matrix2)

[[6331 0]  
 [3669 0]]

In [240]:

1

print("Confusion Matrix:")

2

print(confusion\_matrix2)

3

4

print("\nClassification Report:")

5

print(classification\_rep)

Confusion Matrix:  
[[6331 0]  
 [3669 0]]  
  
Classification Report:  
 precision recall f1-score support  
  
 0 0.63 1.00 0.78 6331  
 1 0.00 0.00 0.00 3669  
  
 accuracy 0.63 10000  
 macro avg 0.32 0.50 0.39 10000  
weighted avg 0.40 0.63 0.49 10000

In [241]:

1

disp **=** ConfusionMatrixDisplay(confusion\_matrix2, display\_labels**=**['Not Readmitted', 'Readmitted'])

2

disp.plot(cmap**=**plt.cm.Blues, values\_format**=**'d')

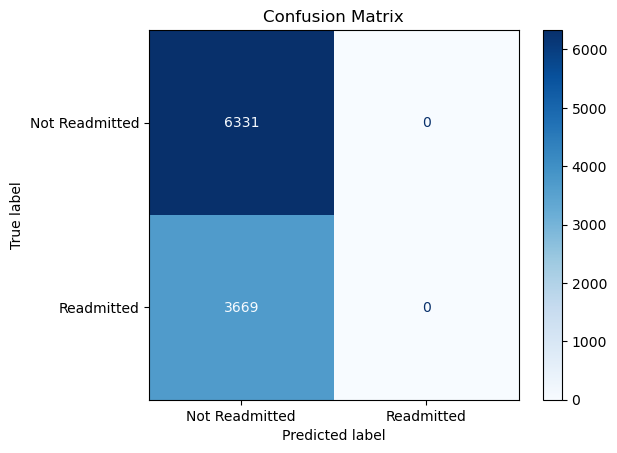
3

4

plt.title("Confusion Matrix")

5

plt.show()



In [242]:

1

accuracy **=** accuracy\_score(y, y\_pred\_binary)

In [243]:

1

print("Accuracy:", accuracy)

Accuracy: 0.6331

In [244]:

1

vif\_data**=** ['HighBlood', 'Stroke', 'Arthritis',

2

'Diabetes','Hyperlipidemia', 'BackPain', 'Anxiety', 'Allergic\_rhinitis',

3

'Reflux\_esophagitis', 'Asthma','Services\_CT Scan','TotalCharge' ,

4

'Services\_MRI', 'Complication\_risk\_Low',

5

'Complication\_risk\_Medium', 'Initial\_admin\_Emergency Admission',

6

'Initial\_admin\_Observation Admission']

7

In [245]:

1

vif\_data **=** pd.DataFrame()

2

vif\_data["feature"] **=** X.columns

3

vif\_data["VIF"] **=** [variance\_inflation\_factor(X.values, i)

4

**for** i **in** range(len(X.columns))]

5

6

print(vif\_data)

feature VIF  
0 const 16.762851  
1 HighBlood 1.001839  
2 Stroke 1.001171  
3 Arthritis 1.003092  
4 Diabetes 1.001775  
5 Hyperlipidemia 1.001645  
6 BackPain 1.003311  
7 Anxiety 1.002014  
8 Allergic\_rhinitis 1.001885  
9 Reflux\_esophagitis 1.001785  
10 Asthma 1.001338  
11 Services\_CT Scan 1.007725  
12 TotalCharge 1.024899  
13 Services\_MRI 1.006542  
14 Complication\_risk\_Low 1.291889  
15 Complication\_risk\_Medium 1.297764  
16 Initial\_admin\_Emergency Admission 1.504567  
17 Initial\_admin\_Observation Admission 1.495520

Most predictor variables have low VIF values, suggesting that multicollinearity is not a significant concern for individual predictors.

These variables have VIF values close to 1, indicating low multicollinearity. HighBlood, Stroke, Arthritis, Diabetes, Hyperlipidemia, BackPain, Anxiety, Allergic\_rhinitis, Reflux\_esophagitis, Asthma, Services\_CT Scan, TotalCharge, Services\_MRI

The following variables have higher VIF values, but they are still below 5. While they may suggest a moderate level of multicollinearity, it is not severe.

Complication\_risk\_Low, Complication\_risk\_Medium, Initial\_admin\_Emergency Admission, Initial\_admin\_Observation Admission

Type *Markdown* and LaTeX: 𝛼2*�2*

In [249]:

1

*#E3. Provide an executable error-free copy of the code*

In [250]:

1

med\_data.to\_csv('MSDA208\_PA\_task2\_E3.cvs')

# **Part V: Data Summary and Implications**

In [251]:

1

*#F1.*

In [252]:

1

*# a regression equation for the reduced model*

In(p/(1-p)) = −121.6056−1.1079×HighBlood+1.6548×Stroke−2.6233×Arthritis−0.8999×Diabetes− 1.4398×Hyperlipidemia−1.2672×BackPain−2.5751×Anxiety−1.4335×Allergic\_rhinitis−1.4700×Reflux\_esophagitis−1.3596× Asthma+1.5662×Services\_CT\_Scan+0.0180×TotalCharge+2.6915×Services\_MRI+5.5986×Complication\_risk\_Low+7.0666× Complication\_risk\_Medium−6.7370×Initial\_admin\_Emergency\_Admission+0.7856×Initial\_admin\_Observation\_Admission

In [253]:

1

*#.an interpretation of the coefficients of the reduced model*

\*\*Interpretation: Holding all other variables constant

HighBlood: Coefficient: -1.1079 Interpretation: readmission decrease by 1.1079 for individuals with High Blood compared to those without High Blood.

Stroke: Coefficient: 1.6548 readmission increase by 1.6548 for individuals with a history of Stroke.

Arthritis: Coefficient: -2.6233 Interpretation: readmission decreased by 2.6233 for individuals with Arthritis compared to those without Arthritis.

Diabetes: Coefficient: -0.8999 Interpretation: readmission decrease by 0.8999 for individuals with Diabetes compared to those without Diabetes.

Hyperlipidemia: Coefficient: -1.4398 Interpretation: readmission decrease by 1.4398 for individuals with Hyperlipidemia compared to those without Hyperlipidemia.

BackPain: Coefficient: -1.2672 Interpretation: readmission decrease by 1.2672 for individuals with Back Pain compared to those without Back Pain.

Anxiety: Coefficient: -2.5751 Interpretation: readmission decrease by 2.5751 for individuals with Anxiety compared to those without Anxiety.

Allergic\_rhinitis: Coefficient: -1.4335 Interpretation: readmission decrease by 1.4335 for individuals with Allergic Rhinitis compared to those without Allergic Rhinitis.

Reflux\_esophagitis: Coefficient: -1.4700 Interpretation: readmission decrease by 1.4700 for individuals with Reflux Esophagitis compared to those without Reflux Esophagitis.

Asthma: Coefficient: -1.3596 Interpretation: readmission decreased by 1.3596 for individuals with Asthma compared to those without Asthma.

Services\_CT Scan: Coefficient: 1.5662 Interpretation: readmission increase by 1.5662 for individuals who had a CT Scan compared to those who did not.

TotalCharge: Coefficient: 0.0180 Interpretation: a one-unit increase in Total Charge is associated with an increase in readmission by 0.0180.

Services\_MRI: Coefficient: 2.6915 Interpretation: readmission increase by 2.6915 for individuals who had an MRI compared to those who did not.

Complication\_risk\_Low: Coefficient: 5.5986 Interpretation: readmission increase by 5.5986 for individuals with a Low Complication Risk compared to those with a Medium Complication Risk.

Complication\_risk\_Medium: Coefficient: 7.0666 Interpretation: readmission increase by 7.0666 for individuals with a Medium Complication Risk compared to those with a Low Complication Risk.

Initial\_admin\_Emergency Admission: Coefficient: -6.7370 Interpretation: readmission decrease by 6.7370 for individuals admitted through Emergency Admission compared to those with other initial admission types.

Initial\_admin\_Observation Admission: Coefficient: 0.7856 Interpretation: readmission increase by 0.7856 for individuals admitted through Observation Admission compared to those with other initial admission types.

In [254]:

1

*# the statistical and practical significance of the reduced model*

The likelihood Ratio Test (LLR) p-value is 0.0 which indicates the model is statistically significant. The model is also practically significant because columns such as Initial\_admin\_Observation Admission, Complication\_risk\_Medium, Complication\_risk\_Low, Services\_MRI, TotalCharge, Services\_CT Scan that are suggesting the likelihood of hospital readmission for a patient.

In [255]:

1

*#The limitations of the data analysis*

The analysis relies on a dataset of 10,000 patients, which might be inadequate for making precise outcome predictions. Another limitation is the complexity of medical conditions and the use of coefficients to interpret readmission rates. For instance, Diabetes has a coefficient of -0.8999, suggesting a decrease in readmission by 0.8999 for individuals with Diabetes compared to those without Diabetes. However, having diabetes likely increases a patient's readmission rate.

In [256]:

1

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

1

Based on the analysis, we recommend that stakeholders examine the 17 independent variables that impact readmission rates.

2

We have identified variables such as Complication\_risk\_Medium, Services\_MRI, Services\_CT Scan, and Stroke that are highly associated with an increase in readmission rates. We advise stakeholders to provide education and resources for patients with a history of stroke and complication risk. Additionally, consider implementing an outreach program to ensure patients are connected to a primary care provider and actively engaged in managing their health.

# **Part VI: Demonstration**

In [ ]:

1

*#G* [*https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=041f2878-ea53-4185-95d5-b0dd004b4260*](https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=041f2878-ea53-4185-95d5-b0dd004b4260)

In [ ]:

1

*#HGeeksforGeeks. (n.d.). Detecting Multicollinearity with VIF in Python. GeeksforGeeks.* [*https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/*](https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/)

2

3

Smith, J. (2021, October 15). Building a Logistic Regression **in** Python: Step by Step. Towards Data Science. [https:**//**towardsdatascience.com**/**building**-**a**-**logistic**-**regression**-in-**python**-**step**-**by**-**step**-**becd4d56c9c8](https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8)

4

5

Leung, K. (2022, September 13). *Assumptions of logistic regression, clearly explained*. Medium. https://towardsdatascience.com/assumptions-of-logistic-regression-clearly-explained-44d85a22b290

GeeksforGeeks. (n.d.). Detecting Multicollinearity with VIF in Python. GeeksforGeeks. <https://www.geeksforgeeks.org/detecting-multicollinearity-with-vif-python/>

2

ResidentMario. (2021, July 12). Bivariate Plotting **with** Pandas. Kaggle. [https:**//**www.kaggle.com**/**code**/**residentmario**/**bivariate**-**plotting**-with-**pandas**/**notebook](https://www.kaggle.com/code/residentmario/bivariate-plotting-with-pandas/notebook).

6

1

3

Smith, J. (2021, October 15). Building a Logistic Regression in Python: Step by Step. Towards Data Science. <https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8>

4