In [3]:

```
#import packages and clean data before running multiple regression analysis. Rename th
import numpy as np
import pandas as pd
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
pd.set_option('display.max_columns', None)
import pylab
from pylab import rcParams
import statsmodels.api as sm
import statistics
from scipy import stats
import sklearn
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
from scipy.stats import chisquare
from scipy.stats import chi2_contingency
df = pd.read_csv (r'C:\Users\fahim\Documents\0_WGUDocuments\d208\1medical_clean.csv')
df.rename(columns={'Item1':'Timely_admis','Item2':'Timely_treat',
 'Item3':'Timely_visits','Item4':'Reliability',
 'Item5':'Options','Item6':'Hrs_treat',
 'Item7':'Courteous','Item8':'Active_listen'},inplace=True)
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CaseOrder           10000 non-null  int64
 1   Customer_id         10000 non-null  object
 2   Interaction         10000 non-null  object
 3   UID                 10000 non-null  object
 4   City                10000 non-null  object
 5   State               10000 non-null  object
 6   County              10000 non-null  object
 7   Zip                 10000 non-null  int64
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64
 11  Area                10000 non-null  object
 12  TimeZone            10000 non-null  object
 13  Job                 10000 non-null  object
 14  Children            10000 non-null  int64
 15  Age                 10000 non-null  int64
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object
 18  Gender              10000 non-null  object
 19  ReAdmis             10000 non-null  object
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64
 22  Full_meals_eaten    10000 non-null  int64
 23  vitD_supp           10000 non-null  int64
 24  Soft_drink          10000 non-null  object
 25  Initial_admin       10000 non-null  object
 26  HighBlood           10000 non-null  object
 27  Stroke              10000 non-null  object
 28  Complication_risk   10000 non-null  object
 29  Overweight          10000 non-null  object
 30  Arthritis           10000 non-null  object
 31  Diabetes            10000 non-null  object
 32  Hyperlipidemia      10000 non-null  object
 33  BackPain            10000 non-null  object
 34  Anxiety             10000 non-null  object
 35  Allergic_rhinitis   10000 non-null  object
 36  Reflux_esophagitis  10000 non-null  object
 37  Asthma              10000 non-null  object
 38  Services            10000 non-null  object
 39  Initial_days        10000 non-null  float64
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Timely_admis        10000 non-null  int64
 43  Timely_treat        10000 non-null  int64
 44  Timely_visits       10000 non-null  int64
 45  Reliability         10000 non-null  int64
 46  Options             10000 non-null  int64
 47  Hrs_treat           10000 non-null  int64
 48  Courteous           10000 non-null  int64
 49  Active_listen       10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [4]:

```
#check if there are any missing data entries - if there are none then the output shoul
df.isna().any()
```

Out[4]:

```
CaseOrder             False
Customer_id           False
Interaction           False
UID                   False
City                  False
State                 False
County                False
Zip                   False
Lat                   False
Lng                   False
Population            False
Area                  False
TimeZone              False
Job                   False
Children              False
Age                   False
Income                False
Marital               False
Gender                False
ReAdmis               False
VitD_levels           False
Doc_visits            False
Full_meals_eaten      False
vitD_supp             False
Soft_drink            False
Initial_admin         False
HighBlood             False
Stroke                False
Complication_risk     False
Overweight            False
Arthritis             False
Diabetes              False
Hyperlipidemia        False
BackPain              False
Anxiety               False
Allergic_rhinitis     False
Reflux_esophagitis    False
Asthma                False
Services              False
Initial_days          False
TotalCharge           False
Additional_charges    False
Timely_admis          False
Timely_treat          False
Timely_visits         False
Reliability           False
Options               False
Hrs_treat             False
Courteous             False
Active_listen         False
dtype: bool
```

In [5]:

```
#check if there is any duplicate data entries present in columns
df[df.duplicated()]
```

Out[5]:

CaseOrder	Customer_id	Interaction	UID	City	State	County	Zip	Lat	Lng	Population	Area

In [6]:

```
# check if there are any duplicated columns in the data set - if there are none then t
df.columns.duplicated().any()
```

Out[6]:

```
False
```

In [7]:

```
# check if there are any duplicated rows in the data set - if there are none then the
df.duplicated().any()
```

Out[7]:

```
False
```

In [10]:

```
# remove demographic data from the data set since these entries won't be necessary for
df = df.drop(['CaseOrder','Customer_id','Interaction','UID','City','State','County','Z
```

In [11]:

```
# check to make sure that the columns for demographic data were dropped before proceed
df.head()
```

Out[11]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten	vi
0	1	53	86575.93	Divorced	Male	No	19.141466	6	0	
1	3	51	46805.99	Married	Female	No	18.940352	4	2	
2	3	53	14370.14	Widowed	Female	No	18.057507	4	1	
3	0	78	39741.49	Married	Male	No	16.576858	4	1	
4	1	22	1209.56	Widowed	Female	No	17.439069	5	0	

In [12]:

```
# convert categorical yes/no values to numeric 1/0 values
df = df.replace(to_replace = ['Yes','No'],value = [1,0])
df
```

Out[12]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten
0	1	53	86575.93	Divorced	Male	0	19.141466	6	0
1	3	51	46805.99	Married	Female	0	18.940352	4	2
2	3	53	14370.14	Widowed	Female	0	18.057507	4	1
3	0	78	39741.49	Married	Male	0	16.576858	4	1
4	1	22	1209.56	Widowed	Female	0	17.439069	5	0
...	...	...	...	...	...	...	...	...	...
9995	2	25	45967.61	Widowed	Male	0	16.980860	4	2
9996	4	87	14983.02	Widowed	Male	1	18.177020	5	0
9997	3	45	65917.81	Separated	Female	1	17.129070	4	2
9998	3	43	29702.32	Divorced	Male	1	19.910430	5	2
9999	8	70	62682.63	Separated	Female	1	18.388620	5	0

10000 rows × 36 columns

In [14]:

```
# convert the categorical variable of genders to a numeric variable
df['Gender'] = df['Gender'].replace(['Male','Female','Nonbinary'],[1,2,3])
df
```

Out[14]:

	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_meals_eaten
0	1	53	86575.93	Divorced	1	0	19.141466	6	0
1	3	51	46805.99	Married	2	0	18.940352	4	2
2	3	53	14370.14	Widowed	2	0	18.057507	4	1
3	0	78	39741.49	Married	1	0	16.576858	4	1
4	1	22	1209.56	Widowed	2	0	17.439069	5	0
...	...	...	...	...	...	...	...	...	...
9995	2	25	45967.61	Widowed	1	0	16.980860	4	2
9996	4	87	14983.02	Widowed	1	1	18.177020	5	0
9997	3	45	65917.81	Separated	2	1	17.129070	4	2
9998	3	43	29702.32	Divorced	1	1	19.910430	5	2
9999	8	70	62682.63	Separated	2	1	18.388620	5	0

10000 rows × 36 columns

In [15]:

```
# convert the non-married Marital status values to "Married/Not Married", then convert
#this will make the Marital variable easier to work with during regression analysis
df['Marital'] = df['Marital'].replace(['Divorced','Widowed','Separated','Never Marriec
df['Marital'] = df['Marital'].replace(['Married','Not Married'],[1,0])
```

In [32]:

```
# convert the Initial_Admin, Complication_risk, and Services variables into integers b
df['Initial_admin'] = df['Initial_admin'].replace(['Elective Admission','Observation A
df['Complication_risk'] = df['Complication_risk'].replace(['Low','Medium','High'],[1,2
df['Services'] = df['Services'].replace(['Blood Work','CT Scan','Intravenous','MRI'],[
df.info()
df.describe()
my_list = df.columns.values.tolist()
print(my_list)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 37 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Initial_days        10000 non-null  float64
 1   Children            10000 non-null  int64  
 2   Age                 10000 non-null  int64  
 3   Income              10000 non-null  float64
 4   Marital             10000 non-null  int64  
 5   Gender              10000 non-null  int64  
 6   ReAdmis             10000 non-null  int64  
 7   VitD_levels         10000 non-null  float64
 8   Doc_visits          10000 non-null  int64  
 9   Full_meals_eaten    10000 non-null  int64  
 10  vitD_supp           10000 non-null  int64  
 11  Soft_drink          10000 non-null  int64  
 12  Initial_admin       10000 non-null  int64  
 13  HighBlood           10000 non-null  int64  
 14  Stroke              10000 non-null  int64  
 15  Complication_risk   10000 non-null  int64  
 16  Overweight          10000 non-null  int64  
 17  Arthritis           10000 non-null  int64  
 18  Diabetes            10000 non-null  int64  
 19  Hyperlipidemia      10000 non-null  int64  
 20  BackPain            10000 non-null  int64  
 21  Anxiety             10000 non-null  int64  
 22  Allergic_rhinitis   10000 non-null  int64  
 23  Reflux_esophagitis  10000 non-null  int64  
 24  Asthma              10000 non-null  int64  
 25  Services            10000 non-null  int64  
 26  TotalCharge         10000 non-null  float64
 27  Additional_charges  10000 non-null  float64
 28  Timely_admis        10000 non-null  int64  
 29  Timely_treat        10000 non-null  int64  
 30  Timely_visits       10000 non-null  int64  
 31  Reliability         10000 non-null  int64  
 32  Options             10000 non-null  int64  
 33  Hrs_treat           10000 non-null  int64  
 34  Courteous           10000 non-null  int64  
 35  Active_listen       10000 non-null  int64  
 36  intercept           10000 non-null  int64  
dtypes: float64(5), int64(32)
memory usage: 2.8 MB
['Initial_days', 'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_l
evels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'Initial_admin',
'HighBlood', 'Stroke', 'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes', 'H
yperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'As
thma', 'Services', 'TotalCharge', 'Additional_charges', 'Timely_admis', 'Timely_trea
t', 'Timely_visits', 'Reliability', 'Options', 'Hrs_treat', 'Courteous', 'Active_list
en', 'intercept']
```

```
C:\Users\fahim\AppData\Local\Temp\ipykernel_21320\1287141306.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df['Initial_admin'] = df['Initial_admin'].replace(['Elective Admission','Observatio
n Admission','Emergency Admission'],[1,2,3])
C:\Users\fahim\AppData\Local\Temp\ipykernel_21320\1287141306.py:3: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df['Complication_risk'] = df['Complication_risk'].replace(['Low','Medium','High'],
[1,2,3])
C:\Users\fahim\AppData\Local\Temp\ipykernel_21320\1287141306.py:4: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df['Services'] = df['Services'].replace(['Blood Work','CT Scan','Intravenous','MR
I'],[1,2,3,4])
```

Out[32]:

	Initial_days	Children	Age	Income	Marital	Gender	ReAdmis	VitD_levels	Doc_visits	Full_r
0	10.585770	1	53	86575.93	0	1	0	19.141466	6	
1	15.129562	3	51	46805.99	1	2	0	18.940352	4	
2	4.772177	3	53	14370.14	0	2	0	18.057507	4	
3	1.714879	0	78	39741.49	1	1	0	16.576858	4	
4	1.254807	1	22	1209.56	0	2	0	17.439069	5	
...	...	...	...	...	...	...	...	...	...	
9995	51.561220	2	25	45967.61	0	1	0	16.980860	4	
9996	68.668240	4	87	14983.02	0	1	1	18.177020	5	
9997	70.154180	3	45	65917.81	0	2	1	17.129070	4	
9998	63.356900	3	43	29702.32	0	1	1	19.910430	5	
9999	70.850590	8	70	62682.63	0	2	1	18.388620	5	

10000 rows × 37 columns

In [37]:

```
# move the chosen target variable "Initial_days" to beginning of the columns
df=df[['Initial_days','Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis', 'Vi
# Confirm that the target variable was moved before exporting the prepared dataset
my_list = df.columns.values.tolist()
print(my_list)
# describe the dataframe to identify distribution of variables
df.describe()
```

```
['Initial_days', 'Children', 'Age', 'Income', 'Marital', 'Gender', 'ReAdmis', 'VitD_l
evels', 'Doc_visits', 'Full_meals_eaten', 'vitD_supp', 'Soft_drink', 'Initial_admin',
'HighBlood', 'Stroke', 'Complication_risk', 'Overweight', 'Arthritis', 'Diabetes', 'H
yperlipidemia', 'BackPain', 'Anxiety', 'Allergic_rhinitis', 'Reflux_esophagitis', 'As
thma', 'Services', 'TotalCharge', 'Additional_charges', 'Timely_admis', 'Timely_trea
t', 'Timely_visits', 'Reliability', 'Options', 'Hrs_treat', 'Courteous', 'Active_list
en']
```

Out[37]:

	Initial_days	Children	Age	Income	Marital	Gender	ReA
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00
mean	34.455299	2.097200	53.511700	40490.495160	0.202300	1.544600	0.36
std	26.309341	2.163659	20.638538	28521.153293	0.401735	0.539296	0.48
min	1.001981	0.000000	18.000000	154.080000	0.000000	1.000000	0.00
25%	7.896215	0.000000	36.000000	19598.775000	0.000000	1.000000	0.00
50%	35.836244	1.000000	53.000000	33768.420000	0.000000	2.000000	0.00
75%	61.161020	3.000000	71.000000	54296.402500	0.000000	2.000000	1.00
max	71.981490	10.000000	89.000000	207249.100000	1.000000	3.000000	1.00

In [20]:

```python
# now that all the modifications have been made, export the prepared dataset
df.to_csv(r'C:\Users\fahim\Documents\0_WGUDocuments\d208\1medical_clean-PREPAREDTASK1.
```

In [21]:

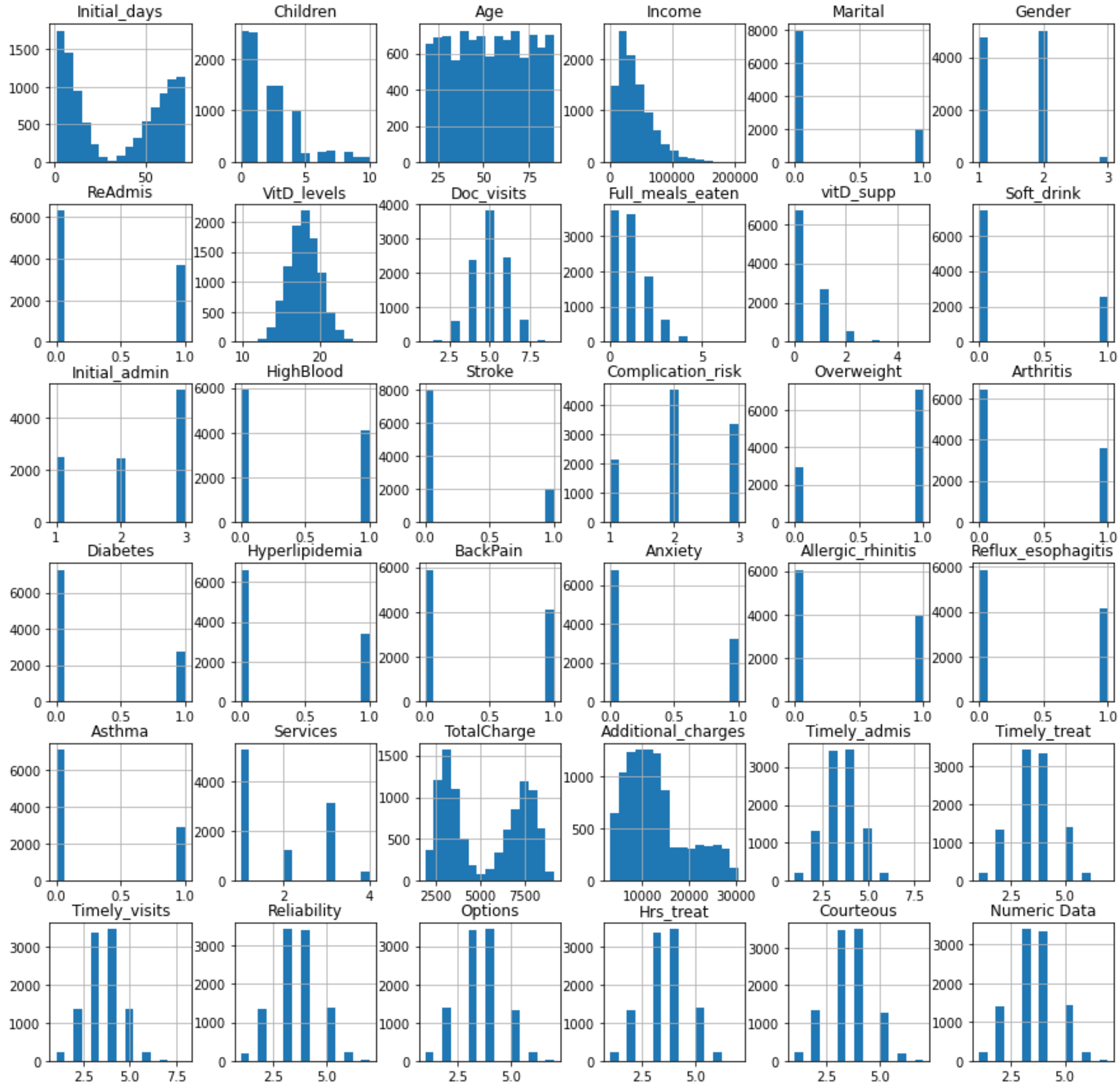
```python
# identify the columns for numerical data
NumericalData = df.select_dtypes(include = "number").columns
print (NumericalData)
```

```
Index(['Initial_days', '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', 'TotalCharge',
       'Additional_charges', 'Timely_admis', 'Timely_treat', 'Timely_visits',
       'Reliability', 'Options', 'Hrs_treat', 'Courteous', 'Active_listen'],
      dtype='object')
```

In [22]:

```python
# create histogram plots of the identified numerica data
fig = plt.figure(figsize=(10, 20))
ax = df[NumericalData].hist(bins = 15, figsize=(15,15))
plt.title('Numeric Data')
fig.tight_layout(h_pad=5, w_pad=5)
plt.show()
```

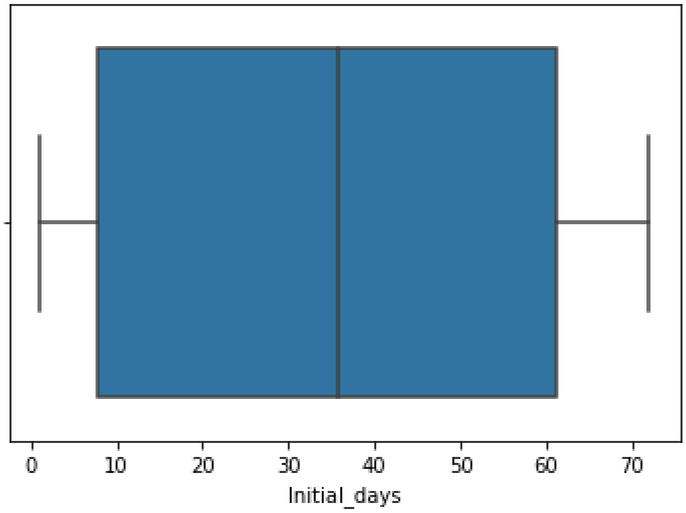
```
<Figure size 720x1440 with 0 Axes>
```



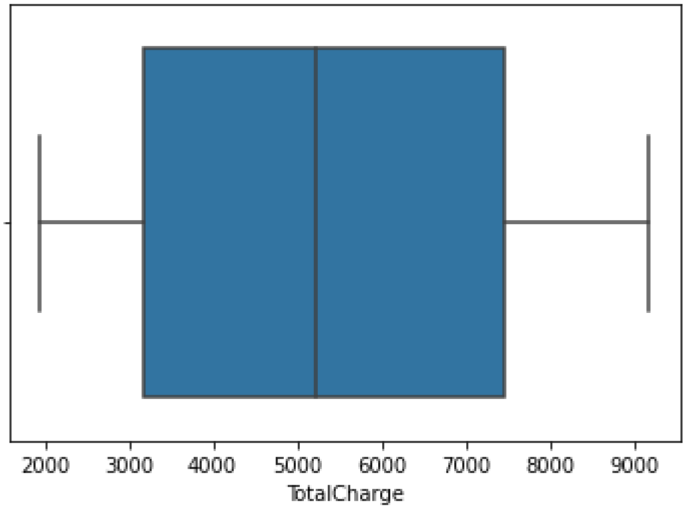
In [23]:

```
# create boxplots for the continuous variables: Initial_days, TotalCharge, and Doc_vis
sns.boxplot('Initial_days', data = df)
plt.show()
sns.boxplot('TotalCharge', data = df)
plt.show()
sns.boxplot('Doc_visits', data = df)
plt.show()
```

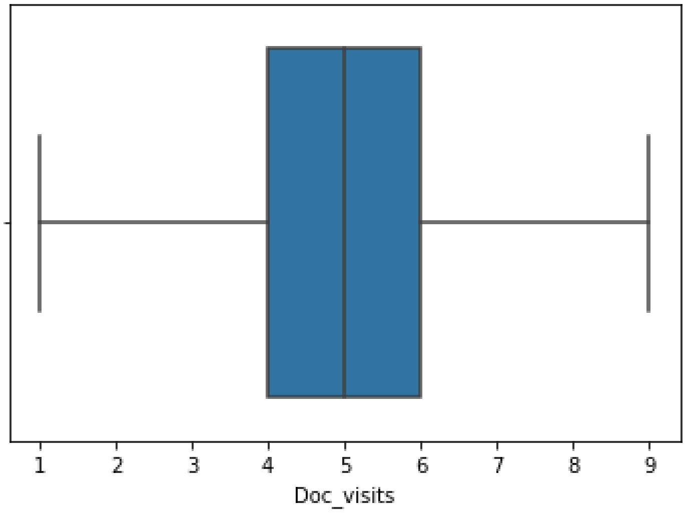
```
C:\Users\fahim\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_dec
orators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
C:\Users\fahim\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_dec
orators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

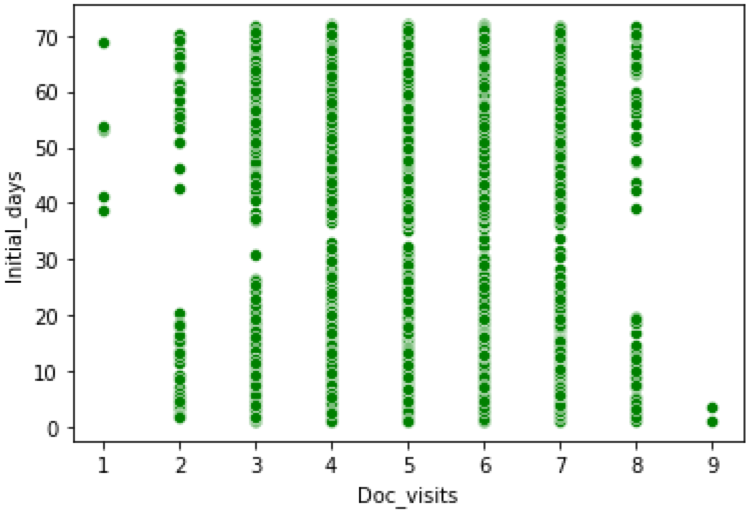


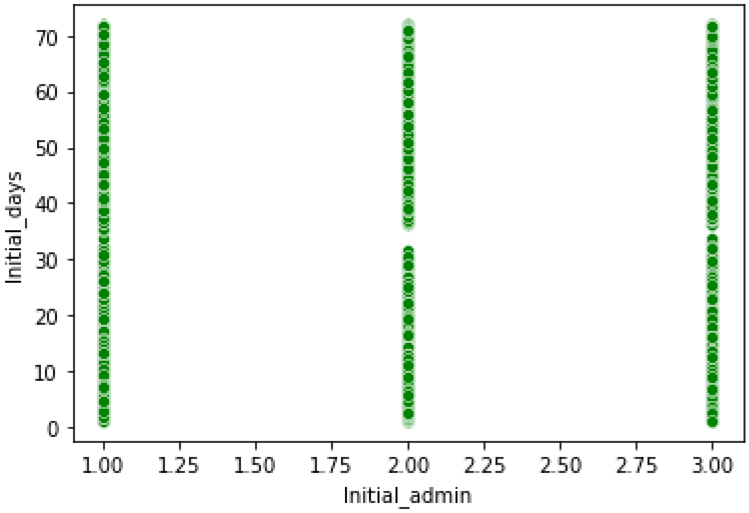
```
C:\Users\fahim\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\_dec
orators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From v
ersion 0.12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

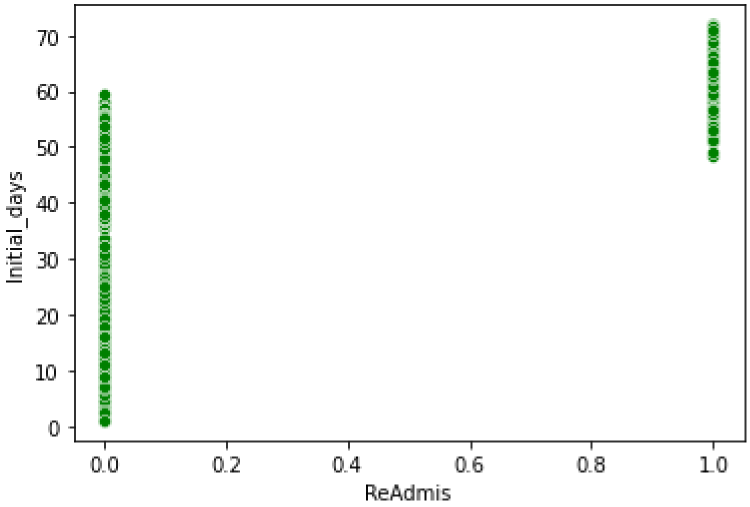


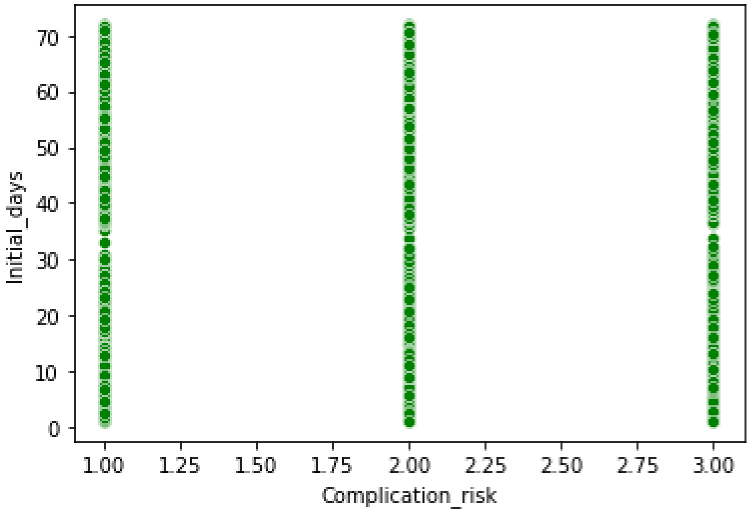
In [24]:

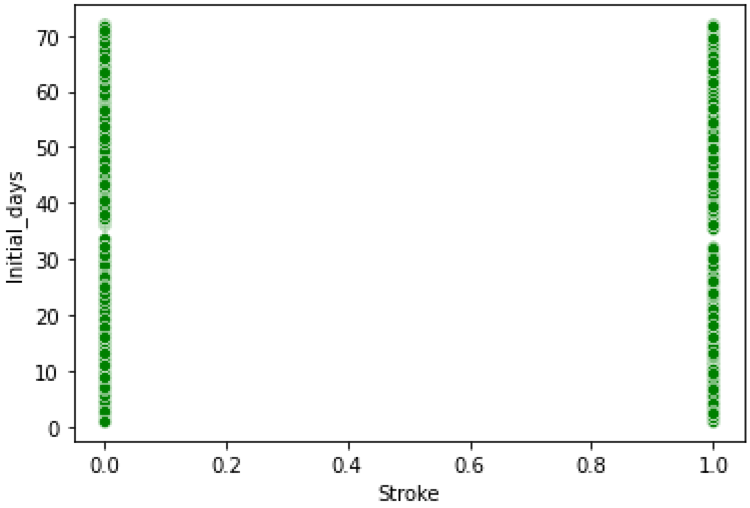
```
# Create scatterplots to demonstrate and identify relationships between the target var
sns.scatterplot(x=df['Doc_visits'],y=df['Initial_days'],color='green')
plt.show();
sns.scatterplot(x=df['Initial_admin'],y=df['Initial_days'],color='green')
plt.show();
sns.scatterplot(x=df['ReAdmis'],y=df['Initial_days'],color='green')
plt.show();
sns.scatterplot(x=df['Complication_risk'],y=df['Initial_days'],color='green')
plt.show();
sns.scatterplot(x=df['Stroke'],y=df['Initial_days'],color='green')
plt.show();
sns.scatterplot(x=df['Diabetes'],y=df['Initial_days'],color='green')
plt.show();
sns.scatterplot(x=df['Timely_admis'],y=df['Initial_days'],color='green')
plt.show();
sns.scatterplot(x=df['Timely_visits'],y=df['Initial_days'],color='green')
plt.show();
df['intercept'] = 1
lm_initialdays = sm.OLS(df['Initial_days'],df[['Age', 'ReAdmis', 'Doc_visits', 'Initia
```

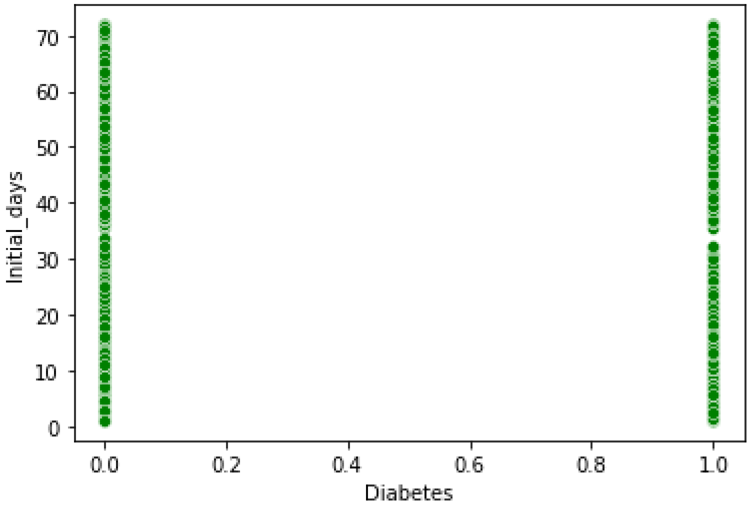


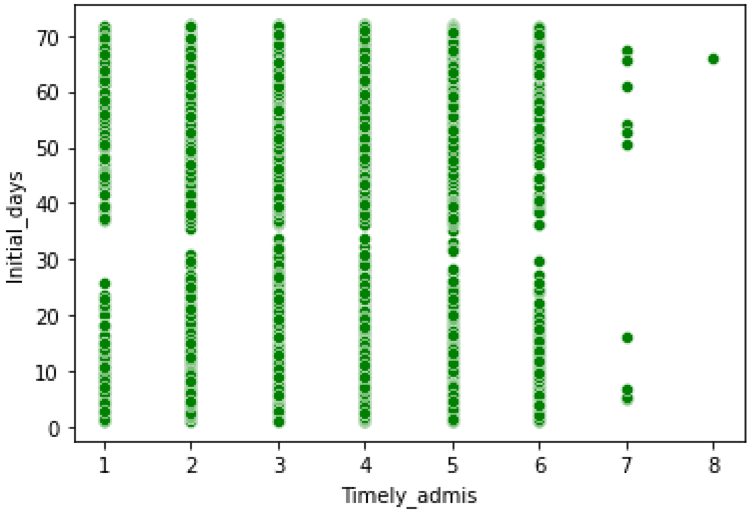


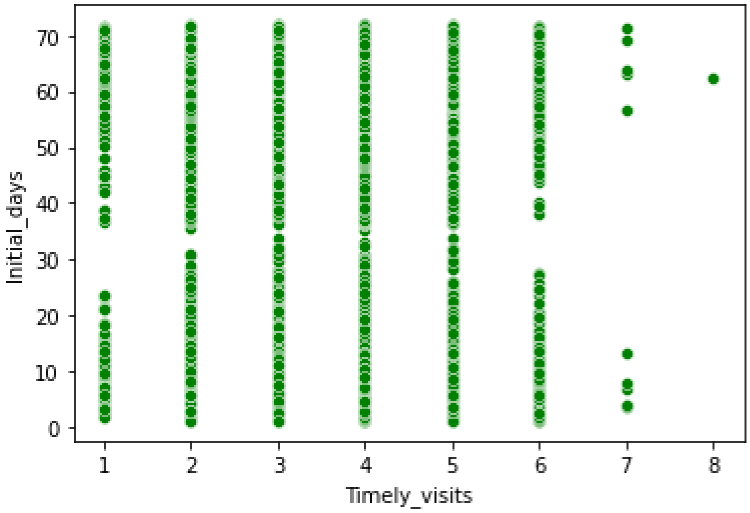

Initial_days
70
60
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
ReAdmis


Initial_days
70
60
50
40
30
20
10
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Complication_risk


Initial_days
70
60
50
40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
Stroke







```
C:\Users\fahim\AppData\Local\Temp\ipykernel_21320\3725291535.py:18: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df['intercept'] = 1
```

In [25]:
```
# retrieve the OLS Regression Results
```

```
print(lm_initialdays.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.994
Model:                            OLS   Adj. R-squared:                  0.994
Method:                 Least Squares   F-statistic:                 9.209e+04
Date:                Mon, 10 Oct 2022   Prob (F-statistic):               0.00
Time:                        17:21:13   Log-Likelihood:                -21027.
No. Observations:               10000   AIC:                         4.209e+04
Df Residuals:                    9980   BIC:                         4.224e+04
Df Model:                          19                                         
Covariance Type:            nonrobust                                         
==============================================================================
=
                         coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
Age                    0.0197      0.003      6.780      0.000       0.014       0.02
5
ReAdmis                0.9935      0.078     12.755      0.000       0.841       1.14
6
Doc_visits             0.0197      0.019      1.037      0.300      -0.018       0.05
7
Initial_admin         -3.3450      0.024   -137.279      0.000      -3.393      -3.29
7
HighBlood             -0.4696      0.112     -4.175      0.000      -0.690      -0.24
9
Stroke                 0.0806      0.050      1.617      0.106      -0.017       0.17
8
Complication_risk     -2.7188      0.028    -98.534      0.000      -2.773      -2.66
5
Diabetes              -0.9084      0.045    -20.395      0.000      -0.996      -0.82
1
Anxiety               -1.0138      0.043    -23.827      0.000      -1.097      -0.93
0
Allergic_rhinitis     -0.8316      0.041    -20.459      0.000      -0.911      -0.75
2
Reflux_esophagitis    -0.7705      0.040    -19.103      0.000      -0.850      -0.69
1
Asthma                 0.0236      0.044      0.540      0.589      -0.062       0.10
9
Services               0.0096      0.020      0.475      0.635      -0.030       0.04
9
TotalCharge            0.0119   1.73e-05    687.854      0.000       0.012       0.01
2
Additional_charges -9.473e-05   1.22e-05     -7.786      0.000      -0.000   -7.09e-0
5
Timely_admis          -0.0339      0.027     -1.279      0.201      -0.086       0.01
8
Timely_treat           0.0089      0.026      0.347      0.729      -0.041       0.05
9
Hrs_treat              0.0138      0.022      0.629      0.529      -0.029       0.05
7
Active_listen          0.0236      0.020      1.158      0.247      -0.016       0.06
4
intercept            -14.5799      0.186    -78.567      0.000     -14.944     -14.21
6
==============================================================================
Omnibus:                      284.824   Durbin-Watson:                   2.008
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              223.140
```

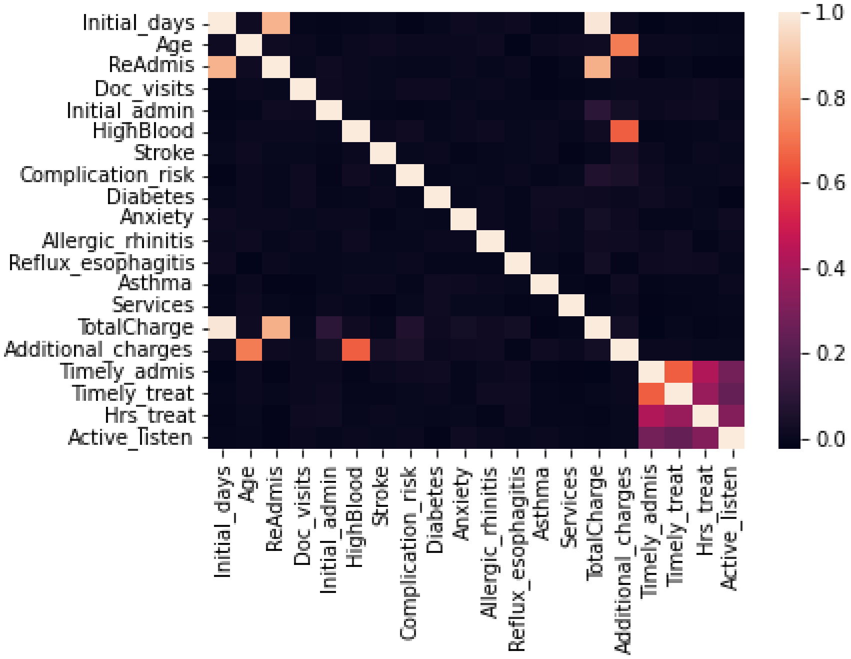
```
Skew:                           0.283   Prob(JB):                     3.51e-49
Kurtosis:                       2.535   Cond. No.                     1.47e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 1.47e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [26]:

```
# to address the strong multicollinearity, create heatmap and correlation matrix
medical_heatmap = df[['Initial_days','Age', 'ReAdmis', 'Doc_visits', 'Initial_admin',
#Initial model heatmap
sns.heatmap(medical_heatmap.corr(), annot=False)
plt.show
medical_heatmap.corr()
```

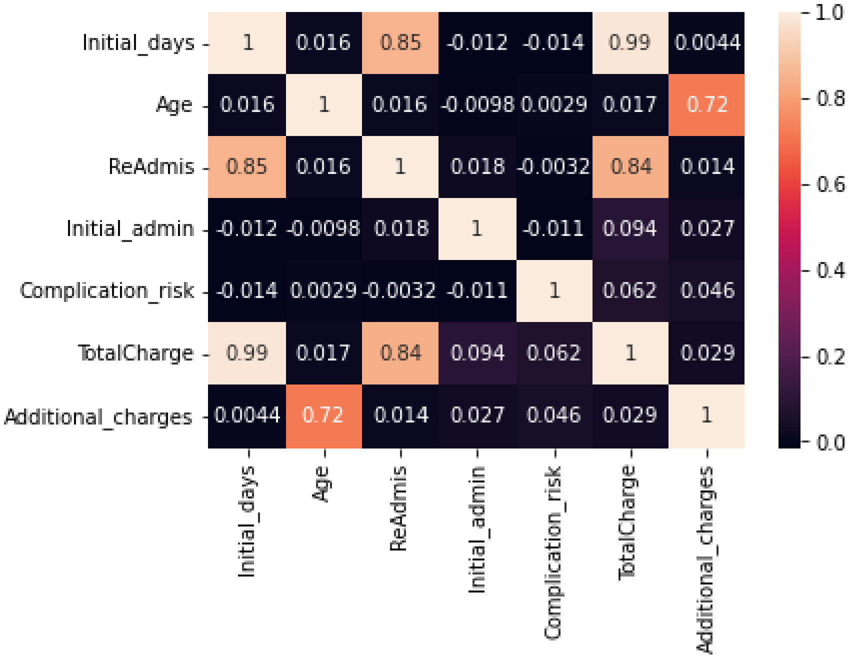
Out[26]:

	Initial_days	Age	ReAdmis	Doc_visits	Initial_admin	HighBlood	Stroke
Initial_days	1.000000	0.016264	0.850862	-0.006754	-0.012058	-0.006333	-0.002043
Age	0.016264	1.000000	0.015810	0.006898	-0.009763	0.007147	0.012035
ReAdmis	0.850862	0.015810	1.000000	0.000246	0.017522	0.002270	0.000918
Doc_visits	-0.006754	0.006898	0.000246	1.000000	0.012518	0.008967	-0.002230
Initial_admin	-0.012058	-0.009763	0.017522	0.012518	1.000000	0.001369	-0.008856
HighBlood	-0.006333	0.007147	0.002270	0.008967	0.001369	1.000000	0.007568
Stroke	-0.002043	0.012035	0.000918	-0.002230	-0.008856	0.007568	1.000000
Complication_risk	-0.014294	0.002887	-0.003236	0.012306	-0.011229	0.021368	0.001119
Diabetes	-0.002411	0.003694	-0.003058	0.012781	-0.009667	-0.005858	0.005792
Anxiety	0.011908	0.006130	0.002406	-0.001684	0.008305	0.008303	-0.013801
Allergic_rhinitis	0.003635	0.012092	-0.004651	0.002920	-0.005741	0.011709	-0.004837
Reflux_esophagitis	0.012237	-0.019609	0.005422	-0.005330	-0.004618	0.001150	-0.000054
Asthma	-0.013496	0.009229	-0.017133	-0.017989	-0.005956	0.006174	0.002443
Services	-0.007448	0.012016	-0.005578	-0.010785	0.003836	-0.003016	-0.016236
TotalCharge	0.987640	0.016876	0.843726	-0.005043	0.094157	0.019910	-0.003694
Additional_charges	0.004409	0.716854	0.013620	0.008072	0.026720	0.654316	0.035140
Timely_admis	-0.022258	0.005552	-0.016785	0.003680	0.006172	-0.011017	0.001948
Timely_treat	-0.007738	0.003967	-0.002423	0.006024	0.011959	-0.007745	-0.007706
Hrs_treat	-0.011752	-0.002087	-0.016894	0.012530	0.016487	-0.002369	0.004282
Active_listen	-0.008034	-0.003367	-0.016740	0.004571	-0.003092	0.002601	0.000040



In [27]:
```
# to narrow the results, remove the diagnosis and survey variables and create a reduce
medical_heatmap = df[['Initial_days','Age', 'ReAdmis', 'Initial_admin','Complication_r
sns.heatmap(medical_heatmap.corr(), annot=True)
plt.show
```

Out[27]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



	Initial_days	Age	ReAdmis	Initial_admin	Complication_risk	TotalCharge	Additional_charges
Initial_days	1	0.016	0.85	-0.012	-0.014	0.99	0.0044
Age	0.016	1	0.016	-0.0098	0.0029	0.017	0.72
ReAdmis	0.85	0.016	1	0.018	-0.0032	0.84	0.014
Initial_admin	-0.012	-0.0098	0.018	1	-0.011	0.094	0.027
Complication_risk	-0.014	0.0029	-0.0032	-0.011	1	0.062	0.046
TotalCharge	0.99	0.017	0.84	0.094	0.062	1	0.029
Additional_charges	0.0044	0.72	0.014	0.027	0.046	0.029	1

In [28]:
```
# create the reduced multiple regression model
df['intercept'] = 1
lm_initialdays_reduced = sm.OLS(df['Initial_days'],df[['Age', 'ReAdmis','Initial_admir
print(lm_initialdays_reduced.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           Initial_days   R-squared:                       0.993
Model:                            OLS   Adj. R-squared:                  0.993
Method:                 Least Squares   F-statistic:                 2.478e+05
Date:                Mon, 10 Oct 2022   Prob (F-statistic):               0.00
Time:                        17:45:37   Log-Likelihood:                -21843.
No. Observations:               10000   AIC:                         4.370e+04
Df Residuals:                    9993   BIC:                         4.375e+04
Df Model:                           6
Covariance Type:            nonrobust
=====================================================================================
=
                        coef    std err          t      P>|t|      [0.025      0.97
5]
-------------------------------------------------------------------------------------
-
Age                   0.0308      0.001     20.537      0.000       0.028       0.03
4
ReAdmis               1.2333      0.084     14.650      0.000       1.068       1.39
8
Initial_admin        -3.3141      0.026   -126.496      0.000      -3.365      -3.26
3
Complication_risk    -2.6898      0.030    -90.460      0.000      -2.748      -2.63
2
TotalCharge           0.0119   1.87e-05    633.739      0.000       0.012       0.01
2
Additional_charges   -0.0001   4.73e-06    -30.383      0.000      -0.000      -0.00
0
intercept           -15.6501      0.120   -130.377      0.000     -15.885     -15.41
5
==============================================================================
Omnibus:                      138.342   Durbin-Watson:                   1.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              112.748
Skew:                           0.189   Prob(JB):                     3.29e-25
Kurtosis:                       2.642   Cond. No.                     8.98e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 8.98e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

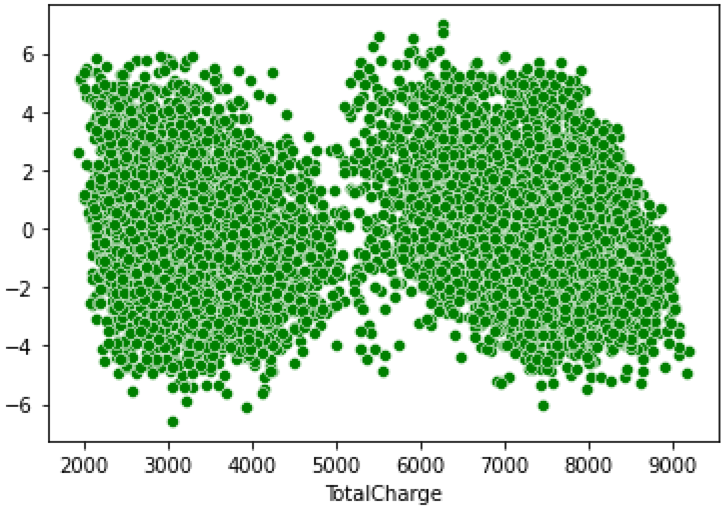
```
C:\Users\fahim\AppData\Local\Temp\ipykernel_21320\1666496053.py:2: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  df['intercept'] = 1
```

In [29]:

```
# load cleaned data to use for the residual plot; we will name the dataframe "regressi
regression_df = pd.read_csv(r'C:\Users\fahim\Documents\0_WGUDocuments\d208\1medical_cl
```

In [31]:

```
# create the residual plot
regression_df['intercept'] = 1
residuals = regression_df['Initial_days'] - lm_initialdays_reduced.predict(regression_
sns.scatterplot(x=regression_df['TotalCharge'],y=residuals,color='green')
plt.show();
```



In [ ]: