In [111]:
```
# Load libraries
import numpy as np
import pandas as pd
import sys
import os


import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display
%matplotlib inline

import plotly.offline as py
import plotly.graph_objs as go
import plotly.tools as tls
py.init_notebook_mode()

from scipy.stats import skew, kurtosis
import warnings
warnings.filterwarnings('ignore')

from pandas import set_option
from tabulate import tabulate
```

In [112]:
```
class CFG:
    SEED = 2023
    imgdim1 = 20
    imgdim2 = 10

plt.style.use('fivethirtyeight')
plt.rcParams.update({'figure.figsize': (CFG.imgdim1, CFG.imgdim2)})
```

In [113]:
```
pd.set_option('display.max_columns', None)
```

In [114]:
```
data = pd.read_csv("diabetes.csv")
```

In [115]:
```
data.head()
```

Out[115]:

	id	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital
0	1	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	1	1
1	2	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	3
2	3	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	2
3	4	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	2
4	5	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	1

The below Features are removed as they represent unique row and do not contribute to model building

In [116]:
```
##Redundant Features
cols = ['id', 'encounter_id', 'patient_nbr','payer_code']
data[cols]
```

Out[116]:

	id	encounter_id	patient_nbr	payer_code
0	1	2278392	8222157	?
1	2	149190	55629189	?
2	3	64410	86047875	?
3	4	500364	82442376	?
4	5	16680	42519267	?
...	...	...	...	...
101761	101762	443847548	100162476	MC
101762	101763	443847782	74694222	MC
101763	101764	443854148	41088789	MC
101764	101765	443857166	31693671	MC
101765	101766	443867222	175429310	?

101766 rows × 4 columns

In [117]:
```
data = data.drop(columns = cols)
```

Let's have a look at data dimensionality, feature names, and feature types.

In [118]:
```
data.shape
```

Out[118]:
```
(101766, 47)
```

In [119]:
```
data.columns
```

Out[119]:
```
Index(['race', 'gender', 'age', 'weight', 'admission_type_id',
       'discharge_disposition_id', 'admission_source_id', 'time_in_hospital',
       'medical_specialty', 'num_lab_procedures', 'num_procedures',
       'num_medications', 'number_outpatient', 'number_emergency',
       'number_inpatient', 'diag_1', 'diag_2', 'diag_3', 'number_diagnoses',
       'max_glu_serum', 'A1Cresult', 'metformin', 'repaglinide', 'nateglinide',
       'chlorpropamide', 'glimepiride', 'acetohexamide', 'glipizide',
       'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose',
       'miglitol', 'troglitazone', 'tolazamide', 'examide', 'citoglipton',
       'insulin', 'glyburide.metformin', 'glipizide.metformin',
       'glimepiride.pioglitazone', 'metformin.rosiglitazone',
       'metformin.pioglitazone', 'change', 'diabetesMed', 'readmitted'],
      dtype='object')
```

In [120]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 47 columns):
 #   Column                    Non-Null Count   Dtype 
---  ------                    --------------   ----- 
 0   race                      101766 non-null  object
 1   gender                    101766 non-null  object
 2   age                       101766 non-null  object
 3   weight                    101766 non-null  object
 4   admission_type_id         101766 non-null  int64 
 5   discharge_disposition_id  101766 non-null  int64 
 6   admission_source_id       101766 non-null  int64 
 7   time_in_hospital          101766 non-null  int64 
 8   medical_specialty         101766 non-null  object
 9   num_lab_procedures        101766 non-null  int64 
 10  num_procedures            101766 non-null  int64 
 11  num_medications           101766 non-null  int64 
 12  number_outpatient         101766 non-null  int64 
 13  number_emergency          101766 non-null  int64 
 14  number_inpatient          101766 non-null  int64 
 15  diag_1                    101766 non-null  object
 16  diag_2                    101766 non-null  object
 17  diag_3                    101766 non-null  object
 18  number_diagnoses          101766 non-null  int64 
 19  max_glu_serum             101766 non-null  object
 20  A1Cresult                 101766 non-null  object
 21  metformin                 101766 non-null  object
 22  repaglinide               101766 non-null  object
 23  nateglinide               101766 non-null  object
 24  chlorpropamide            101766 non-null  object
 25  glimepiride               101766 non-null  object
 26  acetohexamide             101766 non-null  object
 27  glipizide                 101766 non-null  object
 28  glyburide                 101766 non-null  object
 29  tolbutamide               101766 non-null  object
 30  pioglitazone              101766 non-null  object
 31  rosiglitazone             101766 non-null  object
 32  acarbose                  101766 non-null  object
 33  miglitol                  101766 non-null  object
 34  troglitazone              101766 non-null  object
 35  tolazamide                101766 non-null  object
 36  examide                   101766 non-null  object
 37  citoglipton               101766 non-null  object
 38  insulin                   101766 non-null  object
 39  glyburide.metformin       101766 non-null  object
 40  glipizide.metformin       101766 non-null  object
 41  glimepiride.pioglitazone  101766 non-null  object
 42  metformin.rosiglitazone   101766 non-null  object
 43  metformin.pioglitazone    101766 non-null  object
 44  change                    101766 non-null  object
 45  diabetesMed               101766 non-null  object
 46  readmitted                101766 non-null  object
dtypes: int64(11), object(36)
memory usage: 36.5+ MB
```

In [121]:
```
data.describe()
```

Out[121]:

	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	num_procedures	num_medications	numbe
count	101766.000000	101766.000000	101766.000000	101766.000000	101766.000000	101766.000000	101766.000000	10
mean	2.024006	3.715642	5.754437	4.395987	43.095641	1.339730	16.021844	
std	1.445403	5.280166	4.064081	2.985108	19.674362	1.705807	8.127566	
min	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	
25%	1.000000	1.000000	1.000000	2.000000	31.000000	0.000000	10.000000	
50%	1.000000	1.000000	7.000000	4.000000	44.000000	1.000000	15.000000	
75%	3.000000	4.000000	7.000000	6.000000	57.000000	2.000000	20.000000	
max	8.000000	28.000000	25.000000	14.000000	132.000000	6.000000	81.000000	

In [122]:
```
data.describe(include=["object", "bool"])
```

Out[122]:

	race	gender	age	weight	medical_specialty	diag_1	diag_2	diag_3	max_glu_serum	A1Cresult	metformin	repaglinide	nateglinide	ch
count	101766	101766	101766	101766	101766	101766	101766	101766	101766	101766	101766	101766	101766	
unique	6	3	10	10	73	717	749	790	4	4	4	4	4	
top	Caucasian	Female	[70-80)	?	?	428	276	250	None	None	No	No	No	
freq	76099	54708	26068	98569	49949	6862	6752	11555	96420	84748	81778	100227	101063	

In [123]:
```
data["diabetesMed"].value_counts(normalize=True)
```

Out[123]:
```
Yes    0.770031
No     0.229969
Name: diabetesMed, dtype: float64
```

In [124]:
```
data["diabetesMed"].value_counts()
```

Out[124]:
```
Yes    78363
No     23403
Name: diabetesMed, dtype: int64
```

In [125]:
```
data["diabetesMed"] = data["diabetesMed"].map(lambda x: 1 if x == 'Yes' else 0)
```

In [126]:
```
data["diabetesMed"].mean()
```

Out[126]:
```
0.7700312481575379
```

Missing values in this dataset are represented by '?'. Only very few of the columns have missing values.

In [127]:
```
np.nan
```

Out[127]:
```
nan
```

In [128]:
```
data = data.replace("?", np.nan)
```

In [129]:
```
print("Check for the number of Duplicate Data")
data.duplicated().sum()
```

```
Check for the number of Duplicate Data
```

Out[129]:
```
0
```

In [130]:
```
missing = data.isna().sum()
print("Features with Missing Values")
100 * missing[missing>0]/data.shape[0]
```

```
Features with Missing Values
```

Out[130]:
```
race                 2.233555
weight              96.858479
medical_specialty   49.082208
diag_1               0.020636
diag_2               0.351787
diag_3               1.398306
dtype: float64
```

Six Features have missing values, a closer look at them to find a proper method to replacing such, either using mean, meadian or removing the features in entirety

Since weight is more than 70% missing we drop it

In [131]:
```
diag_1 = data.diag_1.mode()[0]
diag_2 = data.diag_2.mode()[0]
diag_3 = data.diag_3.mode()[0]
```

In [132]:
```
data['diag_1'] = data['diag_1'].apply(lambda x : diag_1 if x == np.nan else x)
data['diag_2'] = data['diag_1'].apply(lambda x : diag_2 if x == np.nan else x)
data['diag_3'] = data['diag_3'].apply(lambda x : diag_3 if x == np.nan else x)
```

In [133]:
```
data.diag_1 = data.diag_1.fillna(diag_1)
data.diag_2 = data.diag_2.fillna(diag_2)
data.diag_3 = data.diag_3.fillna(diag_3)
```

In [134]:
```
data.drop('weight', axis = 'columns', inplace = True)
```

Data Transformation

Discharge Disposition ID corresponding to [11 or 13 or 14 or 19 or 20 or 21] indicates patient has expired so there is no chance that it will readmit again so we will remove these records.

Discharge Disposition ID has lots of distinct values using domain knowledge we will convert them into small number of categories.

In [135]:
```
data['discharge_disposition_id'] = data['discharge_disposition_id'].apply(lambda x : 1 if int(x) in [6, 8, 9, 13]
                                                                          else ( 2 if int(x) in [3, 4, 5, 14, 22, 23, 24]
                                                                          else ( 10 if int(x) in [12, 15, 16, 17]
                                                                          else ( 11 if int(x) in [19, 20, 21]
                                                                          else ( 18 if int(x) in [25, 26]
                                                                          else int(x) )))))

data = data[~data.discharge_disposition_id.isin([11,13,14,19,20,21])]

data['admission_type_id'] = data['admission_type_id'].apply(lambda x : 1 if int(x) in [2, 7]
                                                            else ( 5 if int(x) in [6, 8]
                                                            else int(x) ))

data['admission_source_id'] = data['admission_source_id'].apply(lambda x : 1 if int(x) in [2, 3]
                                                                else ( 4 if int(x) in [5, 6, 10, 22, 25]
                                                                else ( 9 if int(x) in [15, 17, 20, 21]
                                                                else ( 11 if int(x) in [13, 14]
                                                                else int(x) ))))
```

In [136]:
```
ageDict = {'[0-10)' : 5,
'[10-20)' : 15,
'[20-30)' : 25,
'[30-40)' : 35,
'[40-50)' : 45,
'[50-60)' : 55,
'[60-70)' : 65,
'[70-80)' : 75,
'[80-90)' : 85,
'[90-100)' : 95}

data['age'] = data['age'].apply(lambda x : ageDict[x])
```

The medical_specialty feature, which is crucial, has too many distinct values, so when we apply one hot encoding, it will unnecessarily create a lot of features, according to our research. We used a frequency-based method and domain knowledge, such as the idea that all types of surgons should be included under the "surgon" category, to divide them up into fewer categories.

We failed to group below medical specialist so we grouped them into 'ungrouped' category.

- Endocrinology -- glands
- Gastroenterology --stomach
- Gynecology -- women reproduction system
- Hematology -- Blood
- Hematology/Oncology -- Blood
- Hospitalist -- one who takes care of admitted patients
- Oncology -- cancer
- Ophthalmology -- eye
- otolaryngology -- ears, nose, and throat
- Pulmonology -- respiratory
- Radiology -- diagnosing and treating injuries and diseases using medical imaging (radiology) procedures (exams/tests) such as X-rays

In [137]:
```
data['medical_specialty'].value_counts()
```

Out[137]:
```
InternalMedicine                   14328
Emergency/Trauma                    7449
Family/GeneralPractice              7302
Cardiology                          5296
Surgery-General                     3068
                                   ...
Perinatology                           1
Neurophysiology                        1
Psychiatry-Addictive                   1
Pediatrics-InfectiousDiseases          1
Surgery-PlasticwithinHeadandNeck       1
Name: medical_specialty, Length: 72, dtype: int64
```

In [138]:
```
high_frequency = ['InternalMedicine', 'Family/GeneralPractice', 'Cardiology', 'Surgery-General', 'Orthopedics', 'Orthopedics-
                  'Emergency/Trauma', 'Urology','ObstetricsandGynecology','Psychiatry','Pulmonology ','Nephrology','Radiologis

low_frequency = ['Surgery-PlasticwithinHeadandNeck','Psychiatry-Addictive','Proctology','Dermatology','SportsMedicine','Speec
                 'Neurophysiology','Resident','Pediatrics-Hematology-Oncology','Pediatrics-EmergencyMedicine','Dentistry','DCP
                 'Pediatrics-Pulmonology','Surgery-Pediatric','AllergyandImmunology','Pediatrics-Neurology','Anesthesiology','
                 'Endocrinology-Metabolism','PhysicianNotFound','Surgery-Colon&Rectal','OutreachServices',
                 'Surgery-Maxillofacial','Rheumatology','Anesthesiology-Pediatric','Obstetrics','Obsterics&Gynecology-Gynecolo

pediatrics = ['Pediatrics','Pediatrics-CriticalCare','Pediatrics-EmergencyMedicine','Pediatrics-Endocrinology','Pediatrics-He
              'Pediatrics-Neurology','Pediatrics-Pulmonology', 'Anesthesiology-Pediatric', 'Cardiology-Pediatric', 'Surgery-

psychic = ['Psychiatry-Addictive', 'Psychology', 'Psychiatry',  'Psychiatry-Child/Adolescent', 'PhysicalMedicineandRehabilita


neurology = ['Neurology', 'Surgery-Neuro',  'Pediatrics-Neurology', 'Neurophysiology']


surgery = ['Surgeon', 'Surgery-Cardiovascular',
          'Surgery-Cardiovascular/Thoracic', 'Surgery-Colon&Rectal', 'Surgery-General', 'Surgery-Maxillofacial',
             'Surgery-Plastic', 'Surgery-PlasticwithinHeadandNeck',  'Surgery-Thoracic',
             'Surgery-Vascular', 'SurgicalSpecialty', 'Podiatry']

ungrouped = ['Endocrinology','Gastroenterology','Gynecology','Hematology','Hematology/Oncology','Hospitalist','InfectiousDise
           'Oncology','Ophthalmology','Otolaryngology','Pulmonology','Radiology']


missing = ['MISS_VALUE']

colMedical = []

for val in data['medical_specialty'] :
    if val in pediatrics :
        colMedical.append('pediatrics')
    elif val in psychic :
        colMedical.append('psychic')
    elif val in neurology :
        colMedical.append('neurology')
    elif val in surgery :
        colMedical.append('surgery')
    elif val in high_frequency :
        colMedical.append('high_freq')
    elif val in low_frequency :
        colMedical.append('low_freq')
    elif val in ungrouped :
        colMedical.append('ungrouped')
    elif val in missing :
        colMedical.append('missing')
    else:
        colMedical.append('missing')

data['medical_specialty'] = colMedical
```

Feature Engineering

Domain Knowledge Type --> ICD Values --> Description

- Circulatory --> 390–459, 785 --> Diseases of the circulatory system
- Respiratory --> 460–519, 786 --> Diseases of the respiratory system
- Digestive --> 520–579, 787 --> Diseases of the digestive system
- Diabetes --> 250.xx --> Diabetes mellitus
- Injury --> 800–999 --> Injury and poisoning
- Musculoskeletal --> 710–739 --> Diseases of the musculoskeletal system and connective tissue
- Genitourinary --> 580–629, 788 --> Diseases of the genitourinary system
- Neoplasms --> 140–239 --> Neoplasms

- Pregnecy --> 630–679 --> Complications of pregnancy, childbirth, and the puerperium
- Other

In [139]:
```
data['diag_1'] = data['diag_1'].apply(lambda x : 'other' if (str(x).find('V') != -1 or str(x).find('E') != -1)
                                      else ('circulatory' if int(float(x)) in range(390, 460) or int(float(x)) == 785
                                      else     ('respiratory' if int(float(x)) in range(460, 520) or int(float(x)) == 786
                                      else     ('digestive'   if int(float(x)) in range(520, 580) or int(float(x)) == 787
                                      else     ('diabetes'    if int(float(x)) == 250
                                      else     ('injury'      if int(float(x)) in range(800, 1000)
                                      else ('musculoskeletal' if int(float(x)) in range(710, 740)
                                      else ('genitourinary'   if int(float(x)) in range(580, 630) or int(float(x)) == 788
                                      else ('neoplasms'       if int(float(x)) in range(140, 240)
                                      else ('pregnecy'        if int(float(x)) in range(630, 680)
                                      else 'other'))))))))))

data['diag_2'] = data['diag_2'].apply(lambda x : 'other' if (str(x).find('V') != -1 or str(x).find('E') != -1)
                                      else ('circulatory' if int(float(x)) in range(390, 460) or int(float(x)) == 785
                                      else     ('respiratory' if int(float(x)) in range(460, 520) or int(float(x)) == 786
                                      else     ('digestive'   if int(float(x)) in range(520, 580) or int(float(x)) == 787
                                      else     ('diabetes'    if int(float(x)) == 250
                                      else     ('injury'      if int(float(x)) in range(800, 1000)
                                      else ('musculoskeletal' if int(float(x)) in range(710, 740)
                                      else ('genitourinary'   if int(float(x)) in range(580, 630) or int(float(x)) == 788
                                      else ('neoplasms'       if int(float(x)) in range(140, 240)
                                      else ('pregnecy'        if int(float(x)) in range(630, 680)
                                      else 'other'))))))))))

data['diag_3'] = data['diag_3'].apply(lambda x : 'other' if (str(x).find('V') != -1 or str(x).find('E') != -1)
                                      else ('circulatory' if int(float(x)) in range(390, 460) or int(float(x)) == 785
                                      else     ('respiratory' if int(float(x)) in range(460, 520) or int(float(x)) == 786
                                      else     ('digestive'   if int(float(x)) in range(520, 580) or int(float(x)) == 787
                                      else     ('diabetes'    if int(float(x)) == 250
                                      else     ('injury'      if int(float(x)) in range(800, 1000)
                                      else ('musculoskeletal' if int(float(x)) in range(710, 740)
                                      else ('genitourinary'   if int(float(x)) in range(580, 630) or int(float(x)) == 788
                                      else ('neoplasms'       if int(float(x)) in range(140, 240)
                                      else ('pregnecy'        if int(float(x)) in range(630, 680)
                                      else 'other'))))))))))
```

Redmitted column has values like '>30' that is patient readmitted after 30 days and 'NO' that is patient not readmitted and '<30' that is patient readmitted before 30 days lets replace '>30'/'NO' with 0 and '<30' with 1

In [140]:
```
data['readmitted'] = data['readmitted'].apply(lambda x : 0 if (x == '>30' or x == 'NO') else 1)
```

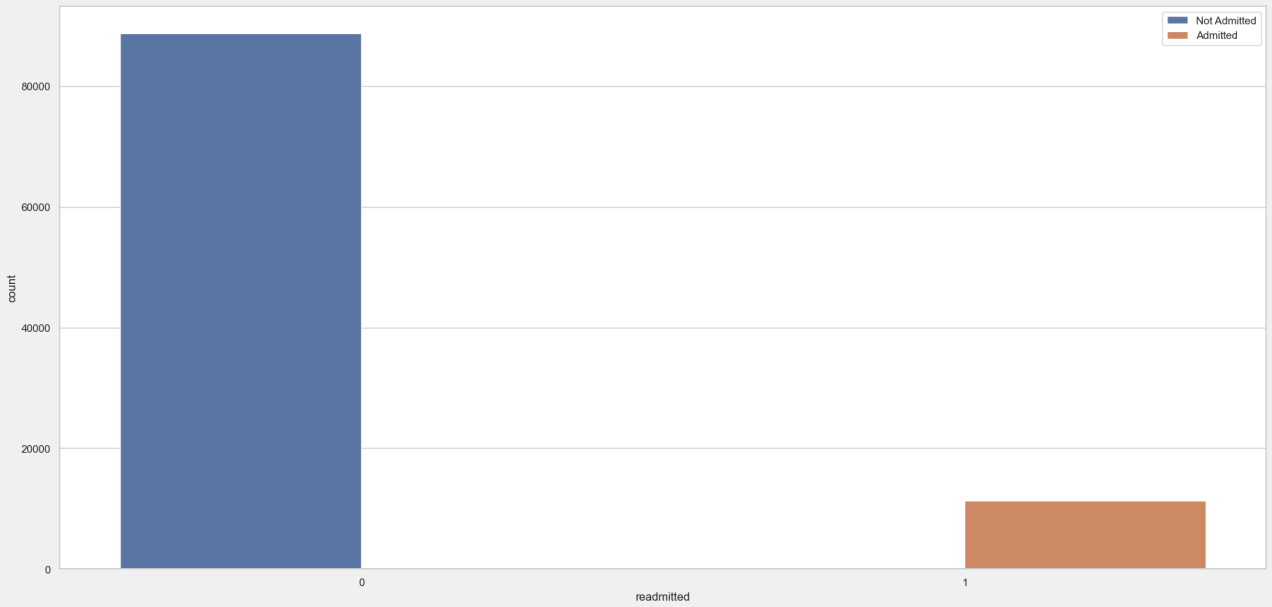
In [141]:
```
data.readmitted.value_counts()
```

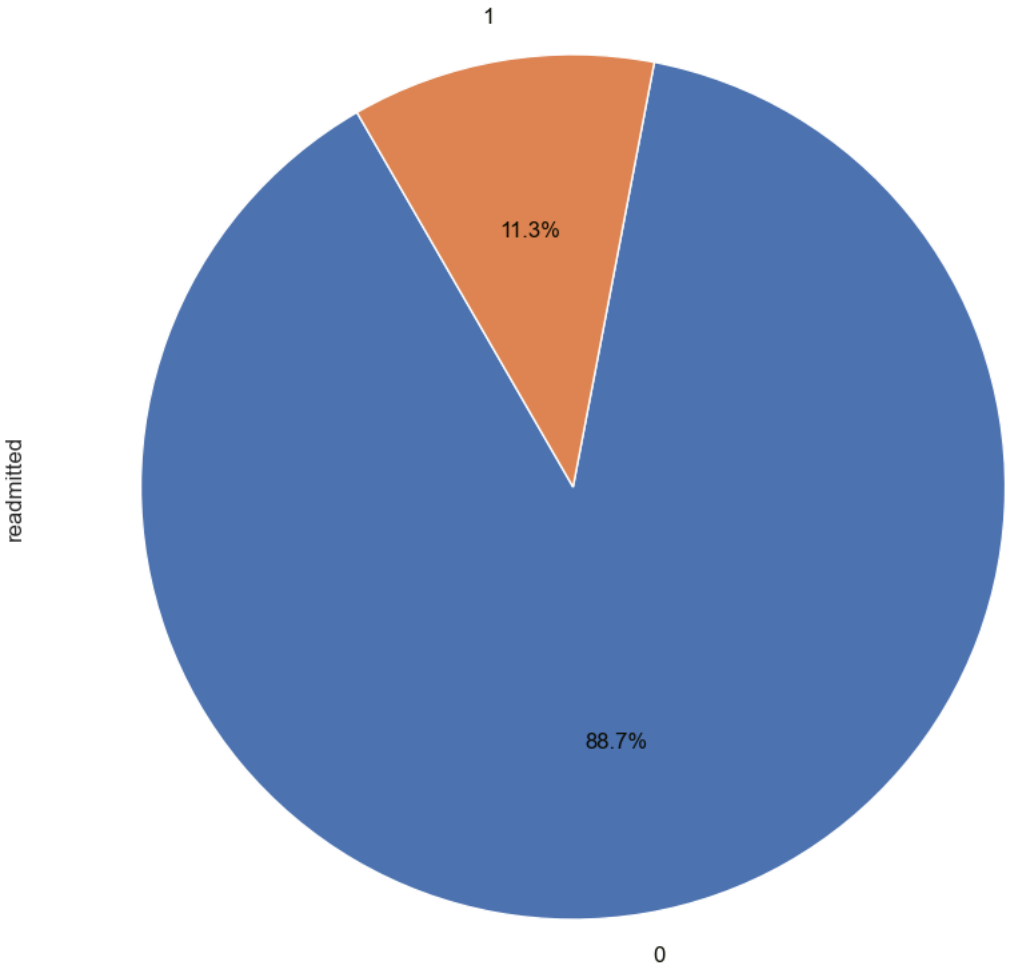
Out[141]:
```
0    88757
1    11357
Name: readmitted, dtype: int64
```

Data Visualization

In [142]:
```
plt.figure()
sns.set_theme(style="whitegrid")
ax = sns.countplot(x = 'readmitted', data = data, hue = 'readmitted')
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels = ['Not Admitted', 'Admitted'])
plt.figure()
data.readmitted.value_counts().plot.pie(autopct="%1.1f%%", startangle=120,
                                        textprops={'fontsize': 12, 'color':'#0a0a00'})
```

Out[142]: <Axes: ylabel='readmitted'>





If the frequency of person's visit to the hospital is high then we can think of that person to be less healthier and less healthier patient tends to readmit quickly lets create health_index variable.Higher the health_index lesser the chance that person will readmit (indirectely propotional)

Health_index = (1 / (number_emergency + number_inpatient + number_outpatient))

Severity of disease is high if patient is spending lots of time in hospital and going through number of complicated test so, lets create severity of disease as feature. To get probablistic interpretation lets divide it by total values.

severity_of_disease = (time_in_hospital + num_procedures + num_medications + num_lab_procedures + number_of_diagnoses)

Research has found that the patient which keep going through changes(up/down) in proportion of medications is tend to readmit so we have engineered new variable called as 'number_of_changes'. This captures number of medications whose proportion have changed for each patient.

In [143]:
```
data['health_index'] = data.apply(lambda x:  1 / (x['number_emergency'] + x['number_inpatient'] + x['number_outpatient'])
                                  if x['number_emergency'] != 0 or x['number_inpatient'] != 0 or x['number_outpatient'] != 0
                                  else 1, axis = 1)



total = data['time_in_hospital'].sum() + data['num_procedures'].sum() + data['num_medications'].sum()+\
                                 data['num_lab_procedures'].sum() + data['number_diagnoses'].sum()

data['severity_of_disease'] = (data['time_in_hospital'] + data['num_procedures'] +
                               data['num_medications'] + data['num_lab_procedures'] +
                               data['number_diagnoses']) / total

drugList = ['metformin','repaglinide','nateglinide','chlorpropamide','glimepiride','acetohexamide',
            'glipizide','glyburide','tolbutamide','pioglitazone','rosiglitazone','acarbose','miglitol',
            'troglitazone','tolazamide','examide','citoglipton','insulin','glyburide.metformin','glipizide.metformin',
            'glimepiride.pioglitazone','metformin.rosiglitazone','metformin.pioglitazone']


number_of_changes = []
for i in range(len(data)):
    changeCount = 0
    for col in drugList :
        if data.iloc[i][col] in ['Down', 'Up'] :
            changeCount += 1
    number_of_changes.append(changeCount)

data['number_of_changes'] = number_of_changes
```

Glucose Serum test : A blood glucose test is used to find out if your blood sugar levels are in the healthy range. It is often used to help diagnose and monitor diabetes.

'>200' : 200 = indicates diabetes

'>300' : 300 = Indicates diabetes

'Norm' : 100 = Normal

'None' : 0 = test was not taken

In [144]:
```
data['max_glu_serum'] = data['max_glu_serum'].apply(lambda x : 200 if x == '>200' else ( 300 if x == '>300' else ( 100 if x =
```

A1C test : An A1C test is a blood test that reflects your average blood glucose levels over the past 3 months

'>7' : 7

'>8' : 8

Norm : 5 = Normal

None : 0 = Test was not taken

In [145]:
```
data['A1Cresult'] = data['A1Cresult'].apply(lambda x : 7 if x == '>7' else (8 if  x == '>8' else ( 5 if x == 'Norm' else  0))
```

In [146]:
```
for col in drugList:
    data[col] = data[col].apply(lambda x : 10 if x == 'Up' else ( -10 if x == 'Down' else ( 0 if x == 'Steady' else  -20)))


data['change'] = data['change'].apply(lambda x : 1 if x == 'Ch' else -1)
```

In [147]:
```
## Engineered Features from domain knowledge
data['total_procedures'] = data['num_procedures'] + data['num_lab_procedures']
data['total_medical_interactions'] = data['number_outpatient'] + data['number_emergency'] + data['number_inpatient']
data['medication_ratio'] = data['num_medications'] / data['time_in_hospital']
data['avg_procedures_per_visit'] = data['total_procedures'] / (data['number_outpatient'] + data['number_inpatient'])
data['diagnoses_per_procedure'] = data['number_diagnoses'] / data['total_procedures']

data["time_in_hospital_per_procedure"] = data["time_in_hospital"] / data["num_procedures"]
data["number_medications_per_diagnosis"] = data["num_medications"] / data["number_diagnoses"]
data["average_lab_procedure_cost"] = data["num_lab_procedures"].mean()
data["emergency_room_visit_rate"] = data["number_emergency"] / data.shape[0]
data["inpatient_admission_rate"] = data["number_inpatient"] / data.shape[0]
```

In [148]:
```
data_checkpoint = data.copy()     ### storing the data to avoid redoing things again and again
```

In [149]:
```
data_checkpoint
```

Out[149]:

	race	gender	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	medical_specialty	num_lab_proce
0	Caucasian	Female	5	5	18	1	1	pediatrics	
1	Caucasian	Female	15	1	1	7	3	missing	
2	AfricanAmerican	Female	25	1	1	7	2	missing	
3	Caucasian	Male	35	1	1	7	2	missing	
4	Caucasian	Male	45	1	1	7	1	missing	
...	...	...	...	...	...	...	...	...	
101761	AfricanAmerican	Male	75	1	2	7	3	missing	
101762	AfricanAmerican	Female	85	1	2	4	5	missing	
101763	Caucasian	Male	75	1	1	7	1	missing	
101764	Caucasian	Female	85	1	2	7	10	surgery	
101765	Caucasian	Male	75	1	1	7	6	missing	

100114 rows × 59 columns

In [150]:
```
numerical_feature = [i for i in data.columns if data[i].dtypes == np.int64 or data[i].dtypes == float]
categorical_feature = [i for i in data.columns if data[i].dtypes != np.int64 or data[i].dtypes != float]
```

In [151]:
```
data[numerical_feature].corr()
```

Out[151]:

	age	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	num_lab_procedures	nu
age	1.000000	-0.005037	0.010197	0.039141	0.108258	0.017872	
admission_type_id	-0.005037	1.000000	0.078516	-0.188376	-0.023393	-0.158222	
discharge_disposition_id	0.010197	0.078516	1.000000	-0.001348	0.063721	0.000157	
admission_source_id	0.039141	-0.188376	-0.001348	1.000000	0.001874	0.160625	
time_in_hospital	0.108258	-0.023393	0.063721	0.001874	1.000000	0.319754	
num_lab_procedures	0.017872	-0.158222	0.000157	0.160625	0.319754	1.000000	
num_procedures	-0.029564	0.117560	0.007864	-0.195165	0.190051	0.051675	
num_medications	0.042493	0.100617	0.002174	-0.093780	0.464082	0.265123	
number_outpatient	0.023853	0.039430	-0.040647	0.008707	-0.009469	-0.007715	
number_emergency	-0.087552	-0.021599	-0.030333	0.071404	-0.009669	-0.001418	
number_inpatient	-0.043610	-0.039480	-0.000419	0.063186	0.073949	0.039162	
number_diagnoses	0.242649	-0.104142	-0.051554	0.115671	0.221034	0.150063	
max_glu_serum	0.019355	0.374252	0.028023	0.220969	0.027235	-0.126962	
A1Cresult	-0.131876	-0.061220	0.005056	0.049759	0.063165	0.255195	
metformin	-0.055550	0.018859	-0.003618	-0.037555	-0.003761	-0.043140	
repaglinide	0.050427	-0.016994	-0.016456	0.011884	0.034158	0.014294	
nateglinide	0.017769	-0.009945	-0.017676	-0.011398	0.005841	-0.006138	
chlorpropamide	0.013812	0.008291	0.029835	-0.004863	0.003097	-0.001742	
glimepiride	0.039058	-0.012056	-0.027846	-0.024316	0.015767	0.001304	
acetohexamide	0.001817	-0.001861	-0.000075	0.002063	0.010211	0.004036	
glipizide	0.055945	0.008522	-0.022969	0.007016	0.018229	0.016180	
glyburide	0.077644	0.003465	0.066376	-0.003623	0.025579	0.000354	
tolbutamide	0.010491	0.003909	-0.002750	0.002752	0.002277	0.001553	
pioglitazone	0.014622	0.013731	-0.023458	-0.015226	0.006860	-0.012297	
rosiglitazone	0.005199	0.016271	-0.008413	-0.022307	0.006577	-0.008258	
acarbose	0.007823	0.004763	0.007086	0.000151	0.006125	0.000359	
miglitol	0.011106	-0.003762	0.001298	0.002741	0.002591	-0.003905	
troglitazone	-0.001429	0.002261	0.007706	0.003574	0.003577	0.003829	
tolazamide	0.005379	0.007878	0.018924	0.001952	-0.003164	-0.000388	
examide	NaN	NaN	NaN	NaN	NaN	NaN	
citoglipton	NaN	NaN	NaN	NaN	NaN	NaN	
insulin	-0.076593	-0.021473	-0.091276	0.017741	0.098308	0.087214	
glyburide.metformin	-0.001206	-0.002137	-0.015573	-0.017781	-0.002748	-0.010445	
glipizide.metformin	0.002155	-0.002758	-0.001682	0.001962	-0.000609	-0.007518	
glimepiride.pioglitazone	-0.000165	-0.001861	-0.000923	0.002063	-0.002539	-0.000796	
metformin.rosiglitazone	0.002570	0.000726	-0.001306	-0.001737	-0.000585	0.001266	
metformin.pioglitazone	-0.000165	0.002888	-0.000075	-0.004519	0.001711	-0.003212	
change	-0.034316	0.009649	-0.056443	0.007882	0.107535	0.065342	
diabetesMed	-0.019746	-0.002387	-0.059461	0.001689	0.060810	0.034397	
readmitted	0.020842	-0.012852	0.021518	0.014770	0.045530	0.023385	
health_index	0.013586	0.000763	0.032938	-0.074438	-0.038580	-0.020088	
severity_of_disease	0.056422	-0.092757	0.004857	0.089964	0.540241	0.907092	
number_of_changes	-0.064529	0.016023	-0.037299	0.037322	0.160920	0.117639	
total_procedures	0.015186	-0.146831	0.000832	0.142544	0.333489	0.996309	
total_medical_interactions	-0.046575	-0.008804	-0.035016	0.068710	0.031532	0.016721	
medication_ratio	-0.065503	0.128349	-0.053593	-0.109825	-0.550720	-0.180448	
avg_procedures_per_visit	0.038691	-0.100708	0.012656	0.044288	0.176630	0.598783	
diagnoses_per_procedure	0.018014	0.001132	-0.032513	-0.016296	-0.085856	-0.484926	
time_in_hospital_per_procedure	0.095304	-0.100288	0.040594	0.142582	0.631085	0.210331	
number_medications_per_diagnosis	-0.098419	0.158916	0.035312	-0.163726	0.258679	0.124668	
average_lab_procedure_cost	NaN	NaN	NaN	NaN	NaN	NaN	
emergency_room_visit_rate	-0.087552	-0.021599	-0.030333	0.071404	-0.009669	-0.001418	
inpatient_admission_rate	-0.043610	-0.039480	-0.000419	0.063186	0.073949	0.039162	

Analysis to use spearsman correlation coefficient to check whether numerical features and readmitted column are dependant or independant if some features are found to be independant on readmitted we will simply remove them

As we can see that correlation is always close to zero but, spearman doesnt capture the non-linear relationships so, rather than using correlation coeff we will use pvalue to get "rejected features list" Here hypothesis testing is done assuming null hypothesis to be "variables are independant" so assuming significance level = alpha = 0.35 if pvalue < alpha then reject null hypothesis that is we accept variables are dependant

In [152]:
```
import scipy
```

In [153]:
```
rejected_features = []
accepted_features = []

for col in numerical_feature :
    rho , pval = scipy.stats.spearmanr(data['readmitted'], data[col])
    print(col, rho, pval)
    print("")


for col in numerical_feature :
    rho , pval = scipy.stats.spearmanr(data['readmitted'], data[col])
    if pval < 0.4 :
        accepted_features.append(col)
    else :
        rejected_features.append(col)

print("List of Features Rejected")
print(rejected_features)

data.drop(rejected_features, axis = 1, inplace=True)
```

```
medication_ratio -0.017808906952670737 1.7479796878953482e-08

avg_procedures_per_visit -0.11839843433449518 2.69069545490532e-309

diagnoses_per_procedure 0.010557807225970871 0.00083589838074872

time_in_hospital_per_procedure 0.017602517341443396 2.548290633507684e-08

number_medications_per_diagnosis 0.015500613884358694 9.354588963398245e-07

average_lab_procedure_cost nan nan

emergency_room_visit_rate 0.06412935666236855 1.0134729189365564e-91

inpatient_admission_rate 0.14089150016426985 0.0

List of Features Rejected
['nateglinide', 'acetohexamide', 'glipizide', 'miglitol', 'troglitazone', 'tolazamide', 'examide', 'citoglipton', 'glybur
ide.metformin', 'glipizide.metformin', 'glimepiride.pioglitazone', 'metformin.rosiglitazone', 'metformin.pioglitazone',
'average_lab_procedure_cost']
```

In [154]:
```
import statsmodels.api as sm

def statistical_analysis(df):
    """
    Compute the summary statistics for each numerical column which include
    Mean, Standard Deviation, Skew, Kurtosis, as well as the
    first order Autocorrelation estimate and its t-stat
    """
    summary = pd.DataFrame({
        "Mean" : df.mean(),
        "Std Dev": df.std(),
        "Skew" : df.apply(skew),
        "Kurtosis": df.apply(kurtosis)
    })

    acf_tstat = pd.DataFrame(columns = ['T-Stat'])

    for col in df.columns:
        acf = sm.tsa.stattools.acf(data[col], nlags = 1)[1]
        acf_tstat.loc[col] = [acf/ np.sqrt(len(df))]

    results = pd.concat([summary, acf_tstat], axis = 1)
    return results
```

In [155]:
```
result = statistical_analysis(df = data[[i for i in data.columns if data[i].dtypes != "object"]])
result
```

Out[155]:

	Mean	Std Dev	Skew	Kurtosis	T-Stat
age	65.830953	15.947425	-0.626691	0.274194	2.932362e-05
admission_type_id	1.783667	1.330994	1.441207	0.703495	2.781926e-04
discharge_disposition_id	2.088359	3.725944	4.127874	15.939251	9.792589e-04
admission_source_id	5.119314	2.880833	-0.570573	-1.390028	4.177205e-04
time_in_hospital	4.389646	2.974531	1.137864	0.870927	6.358810e-05
num_lab_procedures	42.943305	19.620940	-0.241456	-0.253291	2.155715e-04
num_procedures	1.330723	1.700286	1.326109	0.891054	8.314389e-05
num_medications	15.981821	8.092511	1.333098	3.523620	1.224501e-04
number_outpatient	0.369429	1.264006	8.817999	148.544463	6.898910e-05
number_emergency	0.198334	0.935537	22.841802	1185.155821	2.841709e-05
number_inpatient	0.632829	1.261833	3.626420	20.833069	3.114327e-05
number_diagnoses	7.409164	1.938288	-0.867576	-0.109544	3.350850e-04
max_glu_serum	9.075654	42.879610	5.283608	28.806573	1.484103e-04
A1Cresult	1.162685	2.635304	1.913712	1.852590	6.911284e-05
metformin	-15.975788	8.180076	1.606534	0.785317	5.147915e-08
repaglinide	-19.688555	2.532517	8.221573	67.758646	-6.452732e-06
chlorpropamide	-19.982520	0.605211	35.375575	1284.681756	-2.636540e-06
glimepiride	-18.958687	4.529928	4.239244	16.626671	3.144879e-05
glyburide	-17.864534	6.324902	2.735429	5.906418	2.868863e-05
tolbutamide	-19.995805	0.289634	69.024152	4762.333543	-6.630901e-07
pioglitazone	-18.534171	5.266930	3.368455	9.613615	1.751111e-05
rosiglitazone	-18.726951	4.926771	3.660947	11.666777	3.140684e-05
acarbose	-19.937971	1.124153	18.291449	338.020819	3.695772e-07
insulin	-9.417364	10.995387	0.384750	-1.359391	6.439308e-05
change	-0.072198	0.997395	0.144773	-1.979041	6.116113e-05
diabetesMed	0.771840	0.419648	-1.295568	-0.321504	3.437836e-05
readmitted	0.113441	0.317132	2.437855	3.943136	2.084673e-06
health_index	0.832090	0.296320	-1.330674	0.056182	8.968118e-05
severity_of_disease	0.000010	0.000004	0.184378	0.127745	1.393558e-04
number_of_changes	0.287542	0.487859	1.425330	1.433659	8.282556e-05
total_procedures	44.274028	19.781814	-0.217825	-0.208645	2.022932e-04
total_medical_interactions	1.200591	2.292775	5.334641	67.774970	6.664321e-05
medication_ratio	5.061163	3.806133	2.210864	6.984680	2.198539e-04
avg_procedures_per_visit	inf	NaN	NaN	NaN	NaN
diagnoses_per_procedure	0.379956	0.972591	6.516748	47.928591	1.468025e-04
time_in_hospital_per_procedure	inf	NaN	NaN	NaN	NaN
number_medications_per_diagnosis	2.282230	1.322322	2.459574	12.722960	1.818657e-04
emergency_room_visit_rate	0.000002	0.000009	22.841802	1185.155821	2.841709e-05
inpatient_admission_rate	0.000006	0.000013	3.626420	20.833069	3.114327e-05

In [ ]:
```
```

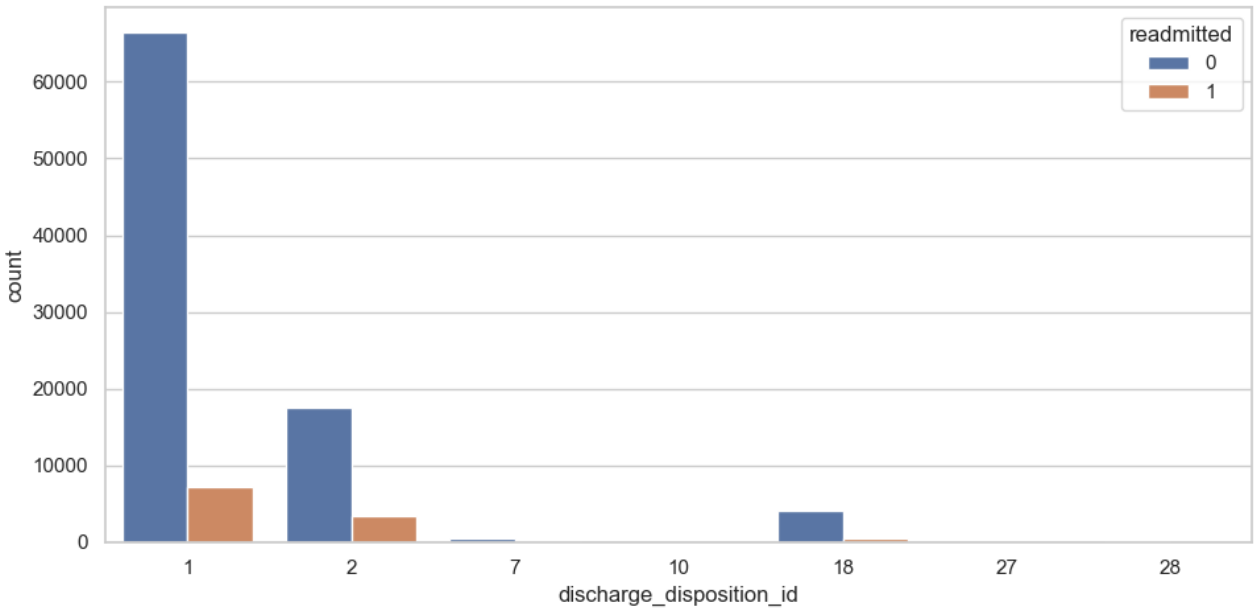
DATA VISUAIZATION

Univariate and Bivariate analysis of Different features

In [156]:
```
data2 = data.copy()
```

In [157]:
```
fig = plt.figure(figsize = (10, 5))

sns.countplot(x  = 'discharge_disposition_id', data = data2, hue = 'readmitted')
```

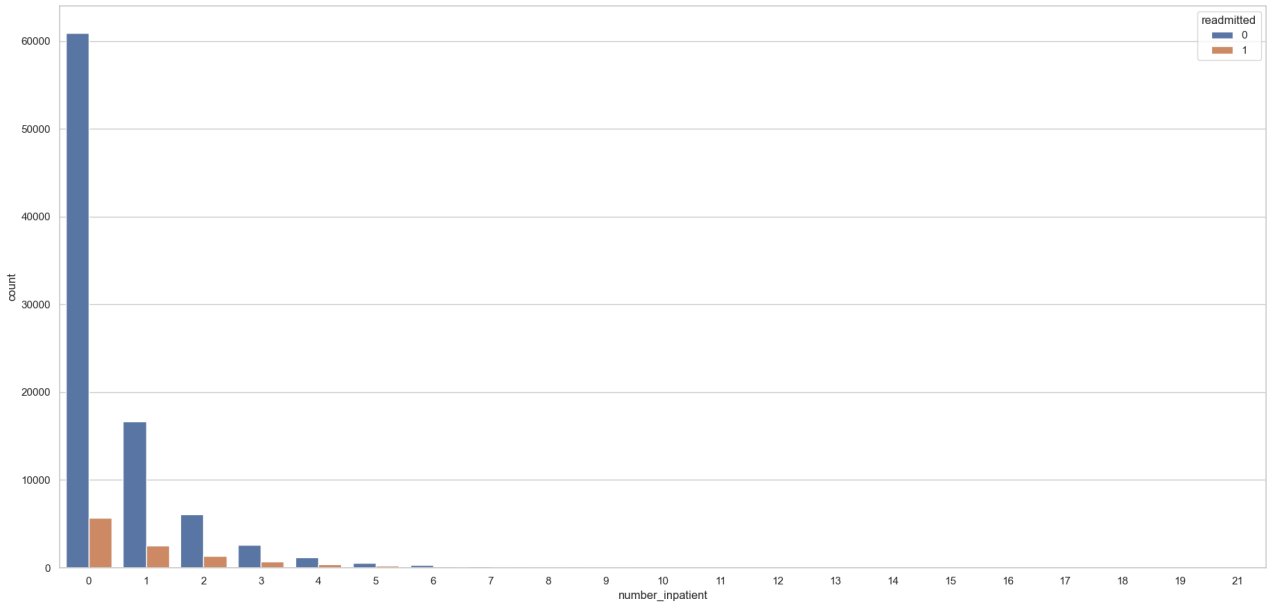
Out[157]: <Axes: xlabel='discharge_disposition_id', ylabel='count'>



- From the graph it clear that if discharge disposition id is 7 the patient wont readmit.

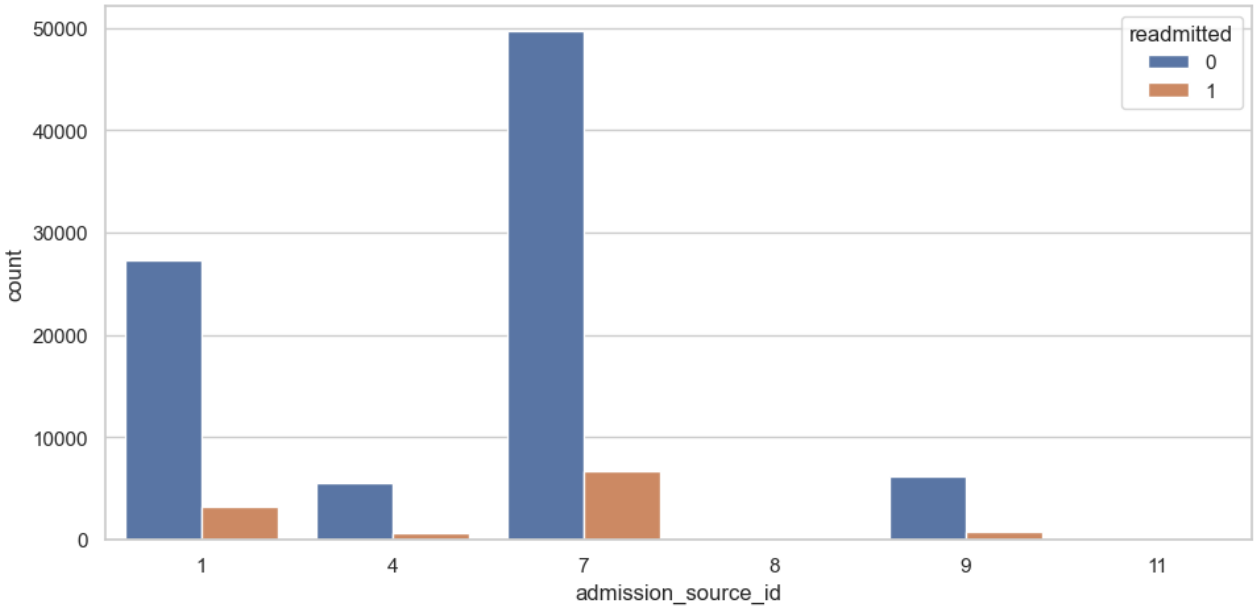
In [158]:
```
sns.countplot(x  = 'number_inpatient', data = data2, hue = 'readmitted')
```

Out[158]: <Axes: xlabel='number_inpatient', ylabel='count'>



- Number Inpatient: Most patient never admitted into the hospital and if patient has not admitted previously or admitted very few number of time there is very less chance that he will readmit.

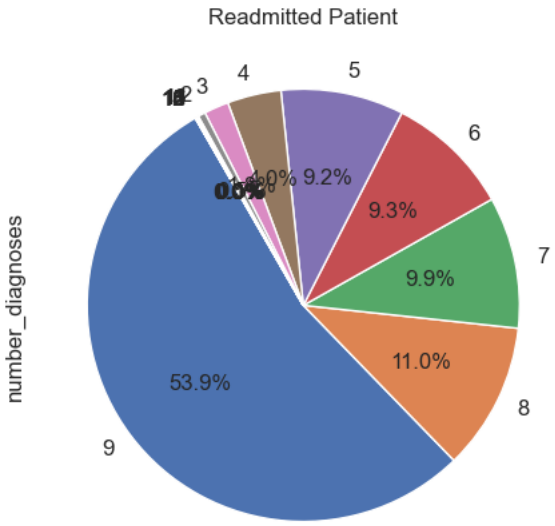
In [159]:
```
fig = plt.figure(figsize = (10, 5))
a = sns.countplot(x = 'admission_source_id', hue = 'readmitted', data = data2)
```

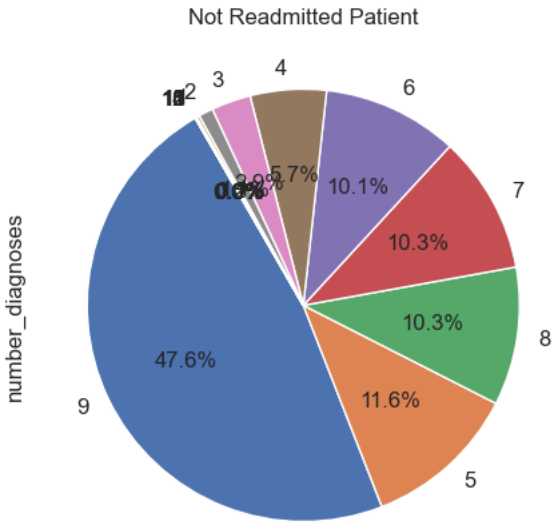


- Most of the patient who readmitted have admission source as 1 and so if some patient has source id as 1 he is more likely be going to readmit.

In [172]:
```
fig = plt.figure(figsize = (10, 5))


ax = data2[data2.readmitted == 1].number_diagnoses.value_counts().plot.pie(autopct="%1.1f%%", startangle=120,
                                        textprops={'fontsize': 12 },)
ax.set_title('Readmitted Patient')
fig = plt.figure(figsize = (10, 5))


ax = data2[data2.readmitted == 0].number_diagnoses.value_counts().plot.pie(autopct="%1.1f%%", startangle=120,
                                        textprops={'fontsize': 12},)
ax.set_title('Not Readmitted Patient')
```

Out[172]: Text(0.5, 1.0, 'Not Readmitted Patient')

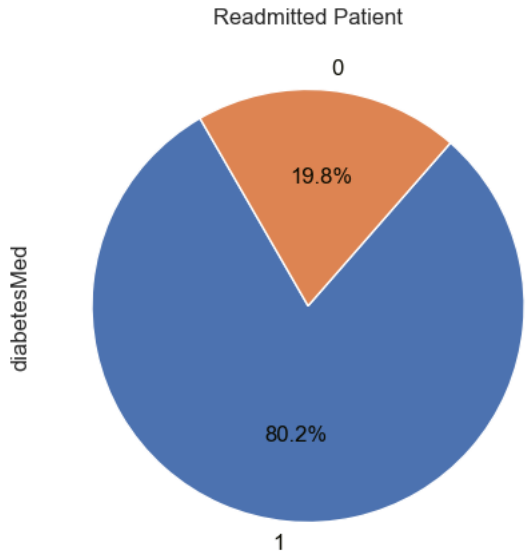


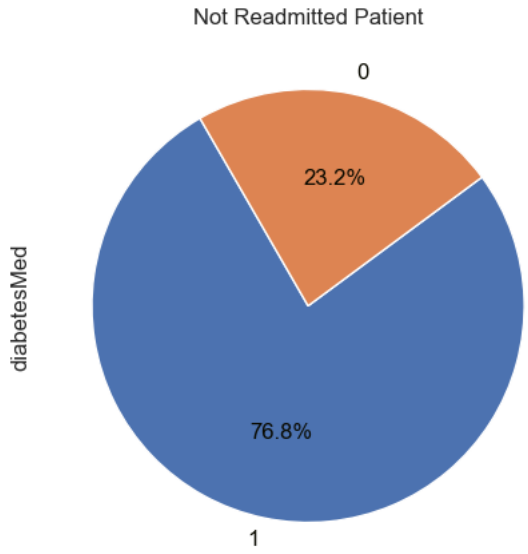


- Distribution of number of diagnosis done by readmitted and not readmitted patient is almost the same hence we are not able to conclude anything from it.

In [161]:
```
fig = plt.figure(figsize = (10, 5))

ax = data2[data2.readmitted == 1].diabetesMed.value_counts().plot.pie(autopct="%1.1f%%", startangle=120,
                                        textprops={'fontsize': 12, 'color':'#0a0a00'},)
ax.set_title('Readmitted Patient')
fig = plt.figure(figsize = (10, 5))


ax = data2[data2.readmitted == 0].diabetesMed.value_counts().plot.pie(autopct="%1.1f%%", startangle=120,
                                        textprops={'fontsize': 12, 'color':'#0a0a00'},)
ax.set_title('Not Readmitted Patient')
```

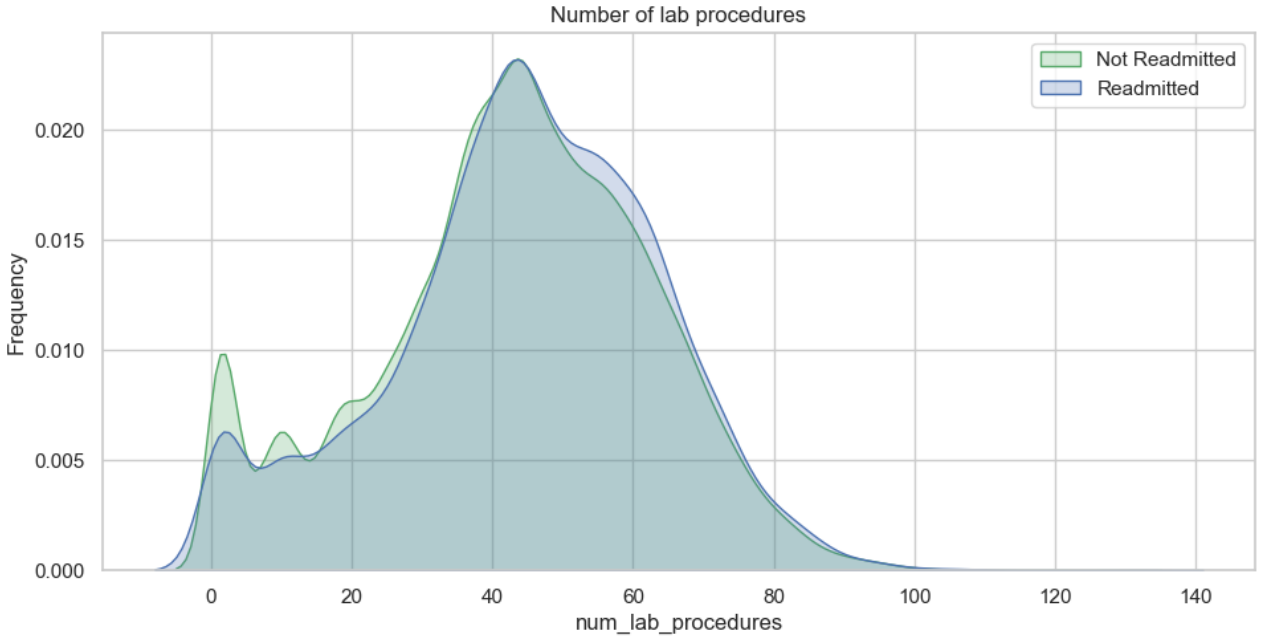
Out[161]: Text(0.5, 1.0, 'Not Readmitted Patient')





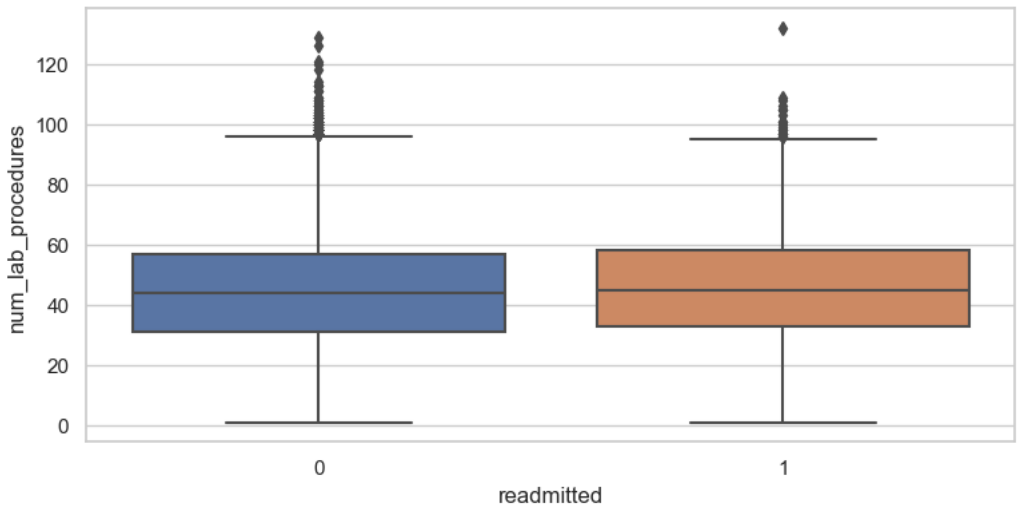
- This feature tells whether the patient has taken Diabetes Medication or not. In our dataset Number of patient taken Diabetes Medication and "readmitted" is almost same as number of patients taken Diabetes medication and "not readmitted".But by iteracting with other feature Diabetes Med might reveal lot of information that is useful for given task.

In [162]:
```
fig = plt.figure(figsize = (10, 5))
a = sns.kdeplot(data2.loc[(data2['readmitted'] == 0), "num_lab_procedures"] ,
                color = "g", shade = True, label = "Not Readmitted")

a = sns.kdeplot(data2.loc[(data2['readmitted'] == 1), "num_lab_procedures"] ,
                color = "b", shade = True, label = "Readmitted")

a.legend()
a.set_xlabel("num_lab_procedures")
a.set_ylabel("Frequency")
a.set_title("Number of lab procedures")
plt.show()
```



- Distribution Number of lab procedures for radmitted and not readmitted patient is exactely same. But it has high varience.High varience features are considered information rich features.

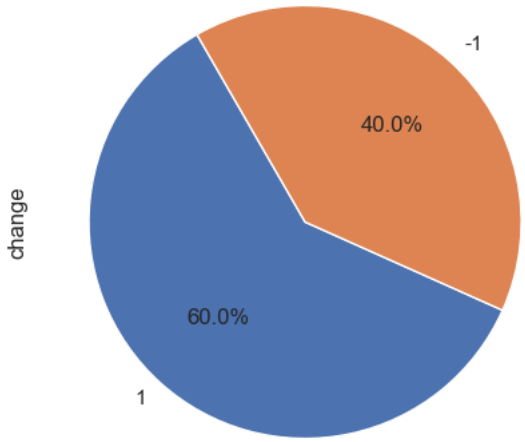
In [163]:
```
fig = plt.figure(figsize = (8, 4))
sns.boxplot(x = 'readmitted', y = 'num_lab_procedures', data = data2);
```



In [164]:
```
data2[data2.readmitted == 1].race.count
```

Out[164]:
```
<bound method Series.count of 11        AfricanAmerican
12              Caucasian
16        AfricanAmerican
46              Caucasian
50        AfricanAmerican
                ...
101699          Caucasian
101727          Caucasian
101732                NaN
101746          Caucasian
101750          Caucasian
Name: race, Length: 11357, dtype: object>
```

In [165]:
```
= plt.figure(figsize = (10, 5))
: data2[(data2.readmitted == 0) & (data2.diabetesMed == 1 )].change.value_counts().plot.pie(autopct="%1.1f%%", startangle=120)
```

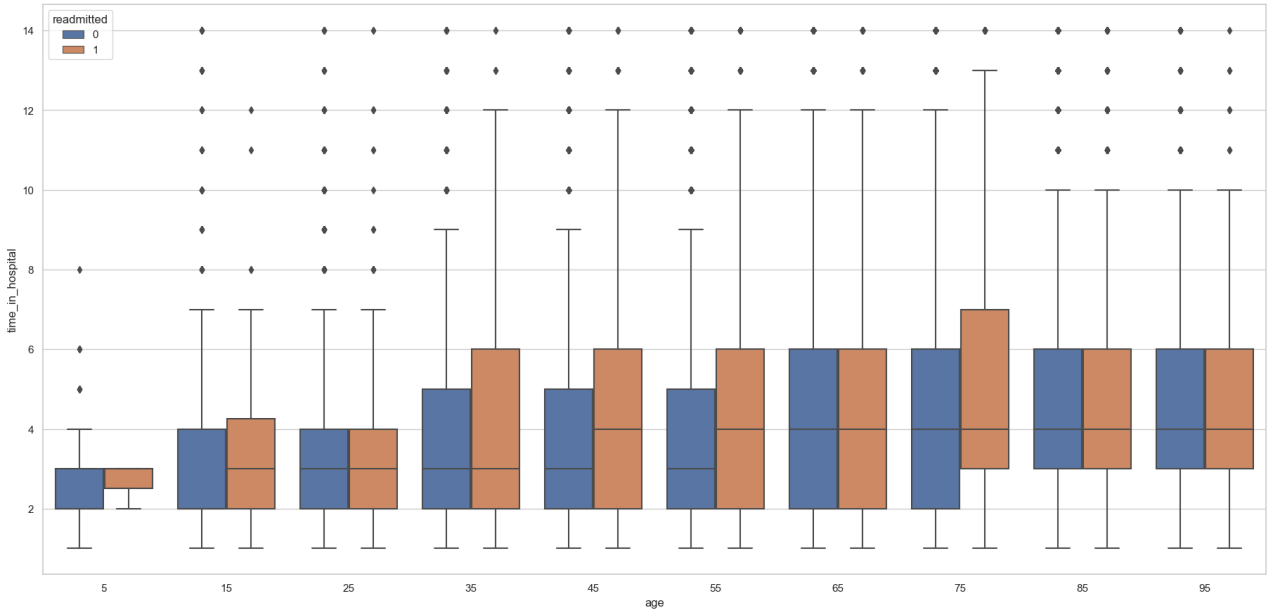


Between Change and DiabetesMed :

From pie chart it clear that if the patient has no changes in medication and has not taken any diabetic medication he has more chance of getting redmitted than the patient who has taken diabetesMed and has no changes in medication.

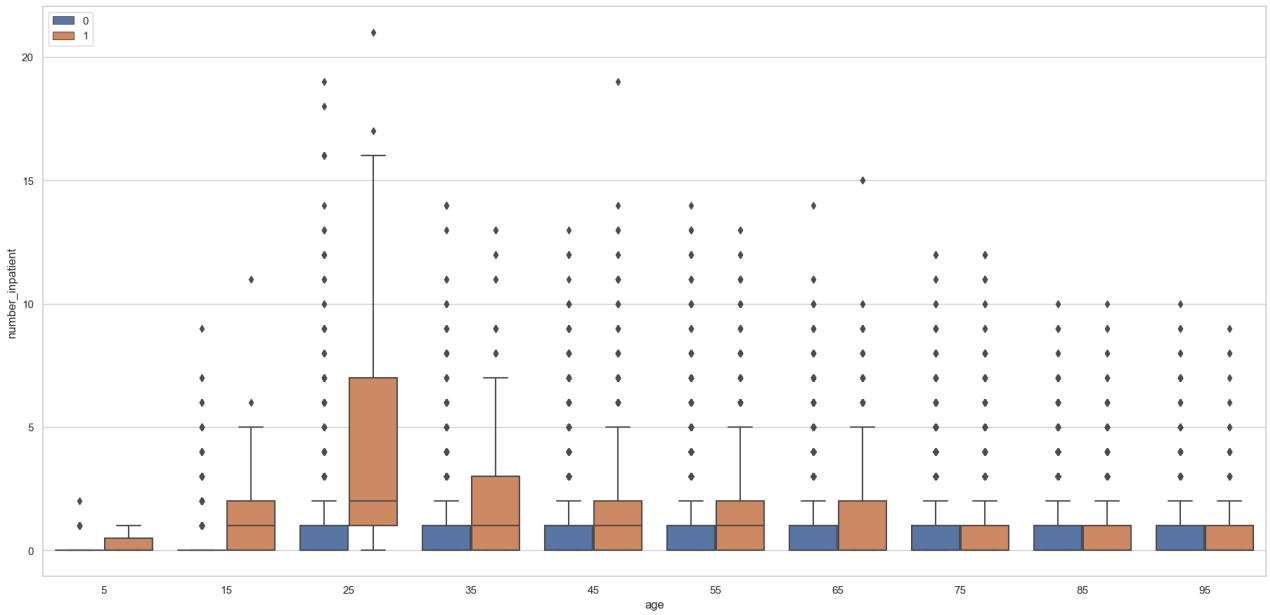
In [166]:
```
plt.figure(figsize = (20,10))
sns.boxplot(x='age', y = 'time_in_hospital', hue = 'readmitted', data = data2)
```

Out[166]:
```
<Axes: xlabel='age', ylabel='time_in_hospital'>
```



Younger people tend to admit in the hospital more often. The patient who have readmitted have high number of inpatient feature value can be seen for group age 25

In [167]:
```
plt.figure(figsize = (20,10))
ax = sns.boxplot(x='age', y = 'number_inpatient', hue = 'readmitted', data = data2)
ax.legend(loc = 'upper left');
```

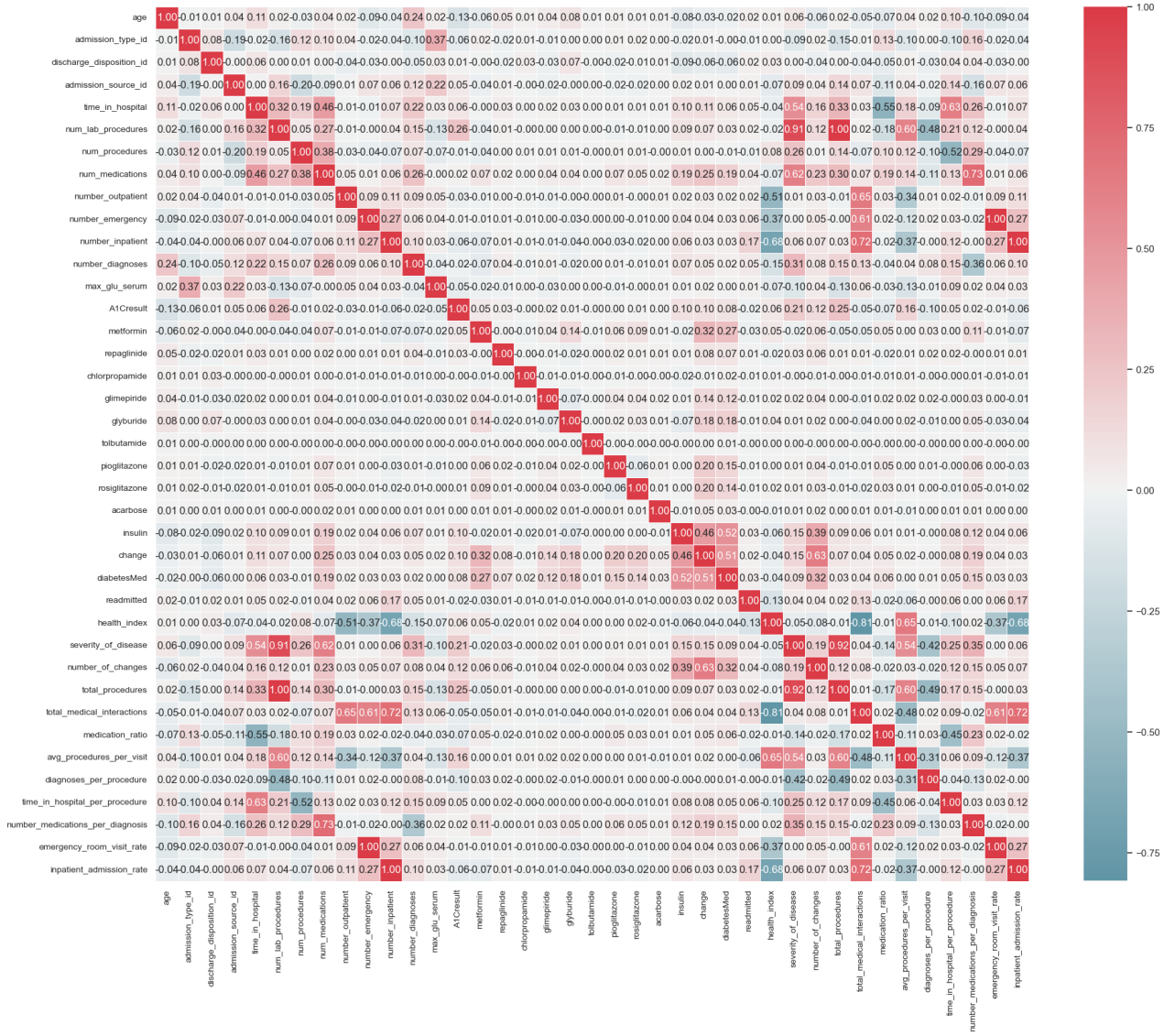


From pie chart we can infer that, If there is patient with source id = 5 and change feature has val as 0 then that patient has less chance of readmitting than the patient with admission id in 15 and has no change.

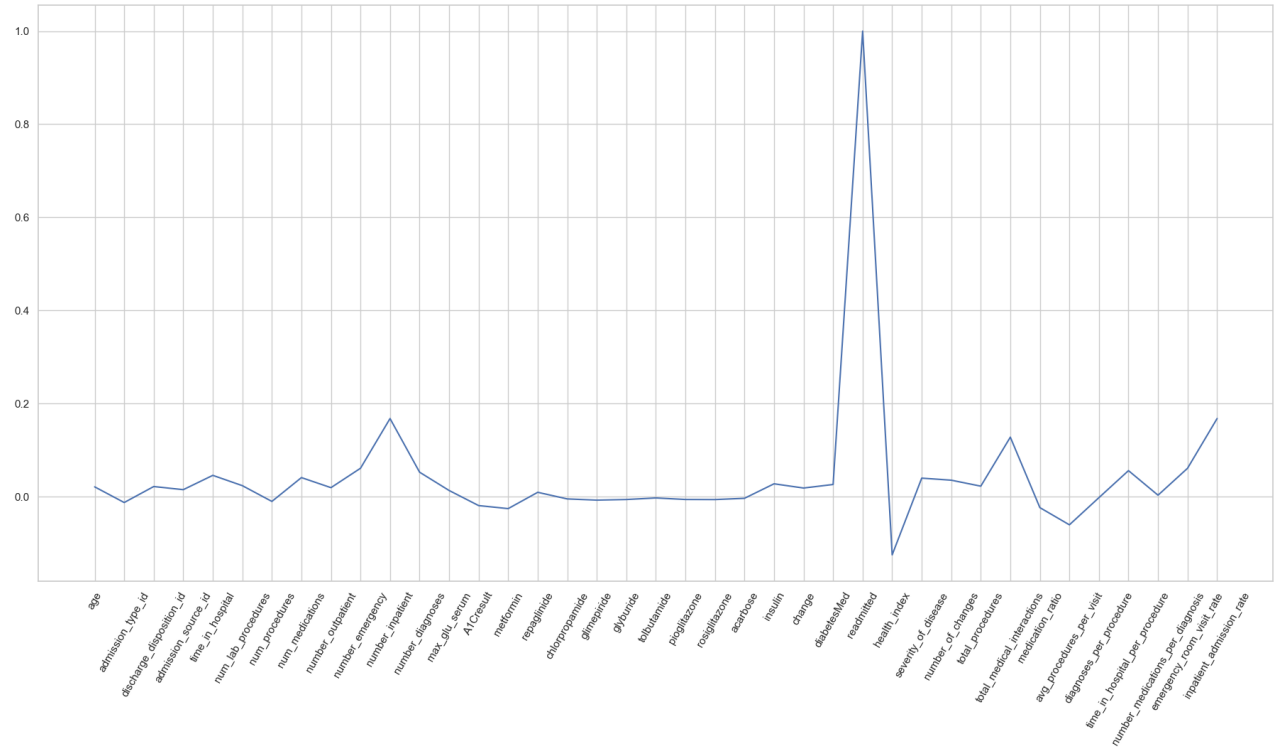
correlation analysis and statistical analysis

In [168]:

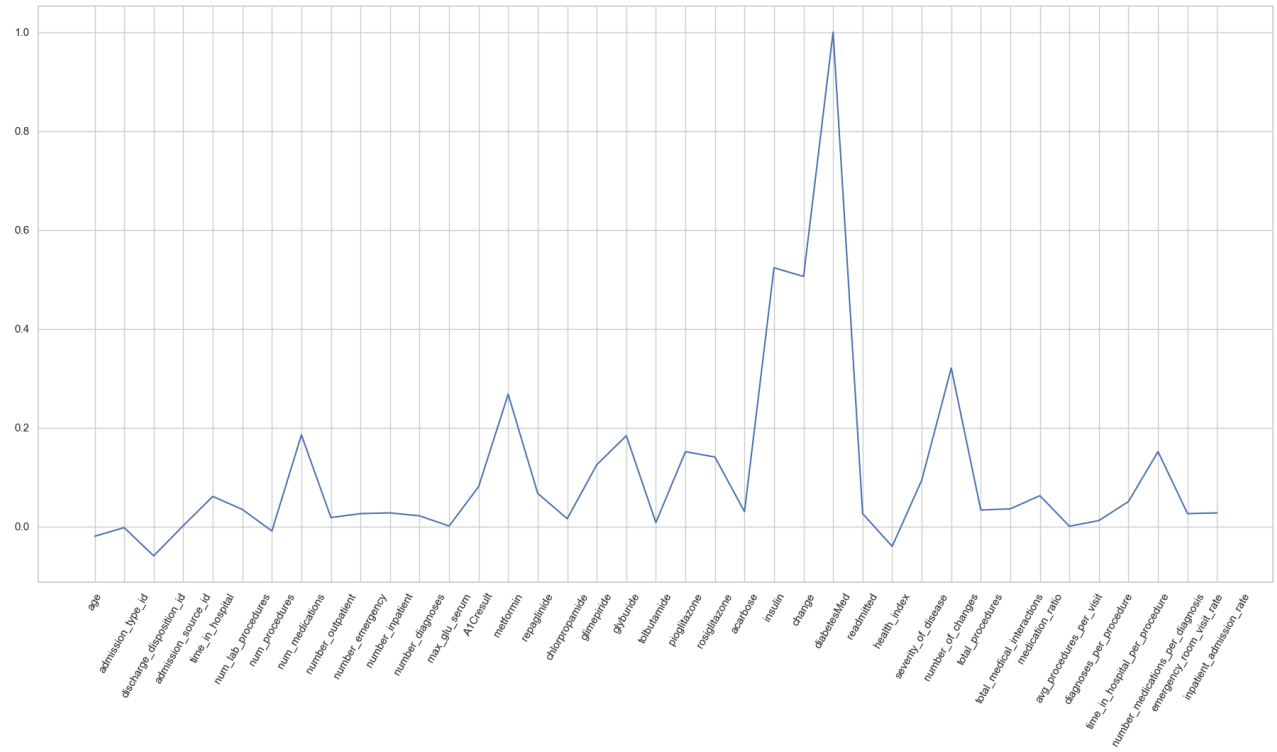
```
# Correlation
def HeatMap(df,x=True):
    correlations = df.corr()
    ## Create color map ranging between two colors
    cmap = sns.diverging_palette(220, 10, as_cmap=True)
    fig, ax = plt.subplots(figsize=(20, 20))
    fig = sns.heatmap(correlations, cmap=cmap, vmax=1.0, center=0, fmt='.2f',square=True, linewidths=.5, annot=x, cbar_kws={"
    fig.set_xticklabels(fig.get_xticklabels(), rotation = 90, fontsize = 10)
    fig.set_yticklabels(fig.get_yticklabels(), rotation = 0, fontsize = 10)
    plt.tight_layout()
    plt.show()

HeatMap(data2,x=True)
```



In [169]:
```
plt.figure(figsize = (20,10))
plt.xticks(rotation = 60)
plt.plot(data2.corr()['readmitted']);
```



In [170]:
```
plt.figure(figsize = (20,10))
plt.xticks(rotation = 60)
plt.plot(data2.corr()['diabetesMed']);
```



In [ ]:

In [ ]: