Github Classroom

Repository : https://github.com/cs418-fa24/project-check-in-team-7

Project description

This project focuses on predicting the likelihood of diabetic patients being readmitted to the hospital within 30 days. By leveraging a dataset containing information about patient encounters, including demographic details, medical history, diagnosis codes, and treatment data, the goal is to develop a model that can identify high-risk patients. By predicting which patients are most likely to be readmitted, hospitals can allocate resources more effectively, prioritize interventions for high-risk individuals, and improve patient care.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
```

In [2]:

```
df = pd.read_csv("diabetic_data_initial.csv")
```

In [3]:

```
df.head()
```

Out[3]:

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

5 rows × 50 columns

Data Exploration

In [4]:
```
print(f"Dataset shape: {df.shape}")
```

```
Dataset shape: (101766, 50)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101766 entries, 0 to 101765
Data columns (total 50 columns):
 #   Column                    Non-Null Count   Dtype 
---  ------                    --------------   ----- 
 0   encounter_id              101766 non-null  int64 
 1   patient_nbr               101766 non-null  int64 
 2   race                      101766 non-null  object
 3   gender                    101766 non-null  object
 4   age                       101766 non-null  object
 5   weight                    101766 non-null  object
 6   admission_type_id         101766 non-null  int64 
 7   discharge_disposition_id  101766 non-null  int64 
 8   admission_source_id       101766 non-null  int64 
 9   time_in_hospital          101766 non-null  int64 
 10  payer_code                101766 non-null  object
 11  medical_specialty         101766 non-null  object
 12  num_lab_procedures        101766 non-null  int64 
 13  num_procedures            101766 non-null  int64 
 14  num_medications           101766 non-null  int64 
 15  number_outpatient         101766 non-null  int64 
 16  number_emergency          101766 non-null  int64 
 17  number_inpatient          101766 non-null  int64 
 18  diag_1                    101766 non-null  object
 19  diag_2                    101766 non-null  object
 20  diag_3                    101766 non-null  object
 21  number_diagnoses          101766 non-null  int64 
 22  max_glu_serum             5346 non-null    object
 23  A1Cresult                 17018 non-null   object
 24  metformin                 101766 non-null  object
 25  repaglinide               101766 non-null  object
 26  nateglinide               101766 non-null  object
 27  chlorpropamide            101766 non-null  object
 28  glimepiride               101766 non-null  object
 29  acetohexamide             101766 non-null  object
 30  glipizide                 101766 non-null  object
 31  glyburide                 101766 non-null  object
 32  tolbutamide               101766 non-null  object
 33  pioglitazone              101766 non-null  object
 34  rosiglitazone             101766 non-null  object
 35  acarbose                  101766 non-null  object
 36  miglitol                  101766 non-null  object
 37  troglitazone              101766 non-null  object
 38  tolazamide                101766 non-null  object
 39  examide                   101766 non-null  object
 40  citoglipton               101766 non-null  object
 41  insulin                   101766 non-null  object
 42  glyburide-metformin       101766 non-null  object
 43  glipizide-metformin       101766 non-null  object
 44  glimepiride-pioglitazone  101766 non-null  object
 45  metformin-rosiglitazone   101766 non-null  object
 46  metformin-pioglitazone    101766 non-null  object
 47  change                    101766 non-null  object
 48  diabetesMed               101766 non-null  object
 49  readmitted                101766 non-null  object
```

```
dtypes: int64(13), object(37)
memory usage: 38.8+ MB
```

In [6]:
```
# Display statistical summary for numerical columns
print(df.describe())
```

```
       encounter_id   patient_nbr  admission_type_id  \
count  1.017660e+05  1.017660e+05      101766.000000   
mean   1.652016e+08  5.433040e+07           2.024006   
std    1.026403e+08  3.869636e+07           1.445403   
min    1.252200e+04  1.350000e+02           1.000000   
25%    8.496119e+07  2.341322e+07           1.000000   
50%    1.523890e+08  4.550514e+07           1.000000   
75%    2.302709e+08  8.754595e+07           3.000000   
max    4.438672e+08  1.895026e+08           8.000000   

       discharge_disposition_id  admission_source_id  time_in_hospital  \
count             101766.000000        101766.000000     101766.000000   
mean                   3.715642             5.754437          4.395987   
std                    5.280166             4.064081          2.985108   
min                    1.000000             1.000000          1.000000   
25%                    1.000000             1.000000          2.000000   
50%                    1.000000             7.000000          4.000000   
75%                    4.000000             7.000000          6.000000   
max                   28.000000            25.000000         14.000000   

       num_lab_procedures  num_procedures  num_medications  number_outpatient  \
count       101766.000000   101766.000000    101766.000000      101766.000000   
mean            43.095641        1.339730        16.021844           0.369357   
std             19.674362        1.705807         8.127566           1.267265   
min              1.000000        0.000000         1.000000           0.000000   
25%             31.000000        0.000000        10.000000           0.000000   
50%             44.000000        1.000000        15.000000           0.000000   
75%             57.000000        2.000000        20.000000           0.000000   
max            132.000000        6.000000        81.000000          42.000000   

       number_emergency  number_inpatient  number_diagnoses  
count     101766.000000     101766.000000     101766.000000  
mean           0.197836          0.635566          7.422607  
std            0.930472          1.262863          1.933600  
min            0.000000          0.000000          1.000000  
25%            0.000000          0.000000          6.000000  
50%            0.000000          0.000000          8.000000  
75%            0.000000          1.000000          9.000000  
max           76.000000         21.000000         16.000000  
```

In [7]:
```
# Display unique values for categorical columns
object_columns = df.select_dtypes(include=['object'])
object_columns.head(10)
for column in object_columns:
    unique_values = df[column].unique()
    print(f"Unique values in '{column}': {unique_values}\n")
```

```
Unique values in 'race': ['Caucasian' 'AfricanAmerican' '?' 'Other' 'Asian' 'Hispani
c']

Unique values in 'gender': ['Female' 'Male' 'Unknown/Invalid']

Unique values in 'age': ['[0-10)' '[10-20)' '[20-30)' '[30-40)' '[40-50)' '[50-60)'
'[60-70)'
 '[70-80)' '[80-90)' '[90-100)']

Unique values in 'weight': ['?' '[75-100)' '[50-75)' '[0-25)' '[100-125)' '[25-50)'
'[125-150)'
 '[175-200)' '[150-175)' '>200']

Unique values in 'payer_code': ['?' 'MC' 'MD' 'HM' 'UN' 'BC' 'SP' 'CP' 'SI' 'DM' 'C
M' 'CH' 'PO' 'WC' 'OT'
 'OG' 'MP' 'FR']

Unique values in 'medical_specialty': ['Pediatrics-Endocrinology' '?' 'InternalMedic
ine'
 'Family/GeneralPractice' 'Cardiology' 'Surgery-General' 'Orthopedics'
 'Gastroenterology' 'Surgery-Cardiovascular/Thoracic' 'Nephrology'
 'Orthopedics-Reconstructive' 'Psychiatry' 'Emergency/Trauma'
 'Pulmonology' 'Surgery-Neuro' 'Obsterics&Gynecology-GynecologicOnco'
 'ObstetricsandGynecology' 'Pediatrics' 'Hematology/Oncology'
 'Otolaryngology' 'Surgery-Colon&Rectal' 'Pediatrics-CriticalCare'
 'Endocrinology' 'Urology' 'Psychiatry-Child/Adolescent'
 'Pediatrics-Pulmonology' 'Neurology' 'Anesthesiology-Pediatric'
 'Radiology' 'Pediatrics-Hematology-Oncology' 'Psychology' 'Podiatry'
 'Gynecology' 'Oncology' 'Pediatrics-Neurology' 'Surgery-Plastic'
 'Surgery-Thoracic' 'Surgery-PlasticwithinHeadandNeck' 'Ophthalmology'
 'Surgery-Pediatric' 'Pediatrics-EmergencyMedicine'
 'PhysicalMedicineandRehabilitation' 'InfectiousDiseases' 'Anesthesiology'
 'Rheumatology' 'AllergyandImmunology' 'Surgery-Maxillofacial'
 'Pediatrics-InfectiousDiseases' 'Pediatrics-AllergyandImmunology'
 'Dentistry' 'Surgeon' 'Surgery-Vascular' 'Osteopath'
 'Psychiatry-Addictive' 'Surgery-Cardiovascular' 'PhysicianNotFound'
 'Hematology' 'Proctology' 'Obstetrics' 'SurgicalSpecialty' 'Radiologist'
 'Pathology' 'Dermatology' 'SportsMedicine' 'Speech' 'Hospitalist'
 'OutreachServices' 'Cardiology-Pediatric' 'Perinatology'
 'Neurophysiology' 'Endocrinology-Metabolism' 'DCPTEAM' 'Resident']

Unique values in 'diag_1': ['250.83' '276' '648' '8' '197' '414' '428' '398' '434'
'250.7' '157'
 '518' '999' '410' '682' '402' '737' '572' 'V57' '189' '786' '427' '996'
 '277' '584' '462' '473' '411' '174' '486' '998' '511' '432' '626' '295'
 '196' '250.6' '618' '182' '845' '423' '808' '250.4' '722' '403' '250.11'
 '784' '707' '440' '151' '715' '997' '198' '564' '812' '38' '590' '556'
 '578' '250.32' '433' 'V58' '569' '185' '536' '255' '250.13' '599' '558'
 '574' '491' '560' '244' '250.03' '577' '730' '188' '824' '250.8' '332'
 '562' '291' '296' '510' '401' '263' '438' '70' '250.02' '493' '642' '625'
 '571' '738' '593' '250.42' '807' '456' '446' '575' '250.41' '820' '515'
 '780' '250.22' '995' '235' '250.82' '721' '787' '162' '724' '282' '514'
 'V55' '281' '250.33' '530' '466' '435' '250.12' 'V53' '789' '566' '822'
 '191' '557' '733' '455' '711' '482' '202' '280' '553' '225' '154' '441'
 '250.81' '349' '?' '962' '592' '507' '386' '156' '200' '728' '348' '459'
 '426' '388' '607' '337' '82' '531' '596' '288' '656' '573' '492' '220'
 '516' '210' '922' '286' '885' '958' '661' '969' '250.93' '227' '112'
 '404' '823' '532' '416' '346' '535' '453' '250' '595' '211' '303'
 '250.01' '852' '218' '782' '540' '457' '285' '431' '340' '550' '54' '351'
 '601' '723' '555' '153' '443' '380' '204' '424' '241' '358' '694' '331'
 '345' '681' '447' '290' '158' '579' '436' '335' '309' '654' '805' '799'
 '292' '183' '78' '851' '458' '586' '311' '892' '305' '293' '415' '591'
 '794' '803' '79' '655' '429' '278' '658' '598' '729' '585' '444' '604'
 '727' '214' '552' '284' '680' '708' '41' '644' '481' '821' '413' '437'
 '968' '756' '632' '359' '275' '512' '781' '420' '368' '522' '294' '825'
 '135' '304' '320' '250.31' '669' '868' '496' '250.43' '826' '567' '3'
 '203' '53' '251' '565' '161' '495' '49' '250.1' '297' '663' '576' '355'
 '850' '287' '250.2' '611' '840' '350' '726' '537' '620' '180' '366' '783'
 '11' '751' '716' '250.3' '199' '464' '580' '836' '664' '283' '813' '966'
 '289' '965' '184' '480' '608' '333' '972' '212' '117' '788' '924' '959'
 '621' '238' '785' '714' '942' '250.23' '710' '47' '933' '508' '478' '844'
 '7' '736' '233' '42' '250.5' '397' '395' '201' '421' '253' '250.92' '600'
 '494' '977' '39' '659' '312' '614' '647' '652' '646' '274' '861' '425'
 '527' '451' '485' '217' '250.53' '442' '970' '193' '160' '322' '581'
 '475' '623' '374' '582' '568' '465' '801' '237' '376' '150' '461' '913'
 '226' '617' '987' '641' '298' '790' '336' '362' '228' '513' '383' '746'
 '353' '911' '506' '873' '155' '860' '534' '802' '141' 'V45' '396' '310'
 '341' '242' '719' '239' '533' '616' '519' '301' 'V66' '5' '989' '230'
 '385' '300' '853' '871' '570' '848' '463' '9' '934' '250.21' '236' '361'
 '594' '501' '810' '643' '430' '528' '205' '791' '983' '992' '490' '172'
 '171' '622' '306' '863' '864' '474' '660' '759' '356' '634' '967' '551'
 '695' '187' '732' '747' '323' '308' '370' '252' '152' '846' '164' '365'
 '718' '48' '266' '720' '94' '344' '797' '170' '878' '904' 'V56' '882'
 '843' '709' '973' '454' '686' '939' '487' '229' '991' '483' '357' '692'
 '796' '693' '935' '936' '800' '920' 'V26' '261' '307' '262' '250.9' '831'
 '145' '223' 'V71' '839' '685' 'V54' '35' '34' '179' '964' '136' '324'
 '389' '815' '334' '143' '526' '588' '192' 'V67' '394' '917' '88' '219'
 '325' '792' '717' '994' '990' '793' '207' '637' '195' '373' '847' '827'
 '31' '891' '814' 'V60' '703' '865' '352' '627' '378' '342' '886' '369'
 '745' '705' '816' '541' '986' '610' '633' '640' '753' '173' '835' '379'
 '445' '272' '382' '945' '619' '881' '250.52' '866' '405' '916' '215'
 '893' '75' '671' '928' '906' '897' '725' '867' '115' '890' '734' '521'
 '674' '470' '834' '146' '696' '524' '980' '691' '384' '142' '879'
 '250.51' '246' '208' '448' '955' '653' '149' '245' '735' '883' '854'
 '952' '838' '194' 'V43' '163' '216' '147' '354' '27' '477' '318' '880'
 '921' '377' '471' '683' '175' '602' '250.91' '982' '706' '375' '417'
 '131' '347' '870' '148' '862' '61' '817' '914' '360' '684' '314' 'V63'
 '36' '57' '240' '915' '971' '795' '988' '452' '963' '327' '731' '842'
 'V25' '645' '665' '110' '944' '603' '923' '412' '363' '957' '976' '698'
 '299' '700' '273' '974' '97' '529' '66' '98' '605' '941' '52' '806' '84'
 '271' '837' '657' '895' '338' '523' '542' '114' '543' '372' 'V70' 'E909'
 '583' 'V07' '422' '615' '279' '500' '903' '919' '875' '381' '804' '704'
 '23' '58' '649' '832' '133' '975' '833' '391' '690' '10' 'V51']

Unique values in 'diag_2': ['?' '250.01' '250' '250.43' '157' '411' '492' '427' '19
8' '403' '288'
 '998' '507' '174' '425' '456' '401' '715' '496' '428' '585' '250.02'
 '410' '999' '996' '135' '244' '41' '571' '276' '997' '599' '424' '491'
 '553' '707' '286' '440' '493' '242' '70' 'V45' '250.03' '357' '511' '196'
 '396' '197' '414' '250.52' '577' '535' '413' '285' '53' '780' '518' '150'
 '566' '250.6' '867' '486' 'V15' '8' '788' '340' '574' '581' '228' '530'
 '250.82' '786' '294' '567' '785' '512' '305' '729' '250.51' '280' '648'
 '560' '618' '444' '38' 'V10' '578' '277' '781' '250.42' '278' '426' '584'
 '462' '402' '153' '272' '733' '34' '881' '203' '250.41' '250.13' '293'
 '245' '250.12' '558' '787' '342' '573' '626' '303' '250.53' '458' '710'
 '415' 'V42' '284' '569' '759' '682' '112' '292' '435' '290' '250.93'
 '642' '536' '398' '319' '711' 'E878' '446' '255' 'V44' '250.7' '784'
 '300' '562' '162' '287' '447' '789' '790' '591' '200' '154' '304' '117'
 '847' '852' '250.83' '250.11' '816' '575' '416' '412' '441' '515' '372'
 '482' '382' 'V65' '572' '283' '78' '250.81' '576' '432' '595' '295' 'V12'
 '204' '466' '721' '434' '590' '271' '813' '368' '227' '783' '250.5' '258'
 '253' '309' '250.91' '519' '333' '459' '250.92' '250.4' '179' '420' '345'
 '433' '661' '537' '205' '722' '405' '437' '714' '211' 'E812' '263' '202'
 '397' '250.23' 'E932' '201' '301' '723' '614' '568' '861' 'V57' '724'
 '189' '297' '453' 'E888' '730' '354' '451' '738' 'E939' '805' 'V43' '155'
 '910' '218' '358' '220' 'E937' '583' '958' '794' '564' '436' '250.22'
 '620' '621' '331' '617' '596' '314' '378' '250.8' '625' '478' '731' '172'
 '404' '681' '470' '279' '281' '531' '443' '799' '337' '311' '719' 'E944'
 '423' 'E870' '465' 'E849' '782' '481' '480' 'V23' '199' '79' '438' '348'
 '42' 'E950' '473' '627' '726' '54' '490' '317' '332' '508' '369' '600'
 '349' '485' '208' '922' '431' '296' 'E934' '753' 'E935' '386' '728' '607'
 'E915' '344' '716' '289' '191' '873' '850' '611' '377' '352' '616' 'V17'
 '136' '455' '933' 'E885' '860' '513' '603' '484' '223' 'V72' '291' '151'
 'V58' '550' '510' '891' '185' '592' '791' '138' '598' '336' '362' '217'
 '825' '298' '821' 'E880' '343' '429' 'E879' '579' '225' '250.9' 'V49'
 '696' '233' '658' '969' '275' '250.1' '601' '704' '808' 'E890' 'V18'
 '920' '380' '570' 'E817' '359' '812' '274' 'V14' '324' '758' 'V66' '911'
 'E931' 'E924' '593' '792' '727' 'V46' '394' '532' 'V64' '557' '864' '718'
 'E942' '807' '604' '924' '820' '580' '273' '241' '282' '824' 'V61' '646'
 '701' '736' '565' '383' '250.2' 'E947' '452' '872' '905' 'E930' '921'
 '131' '448' '389' '421' '214' '705' '494' '752' '623' '9' '299' '959'
 '365' '967' 'E858' '40' '691' '909' '5' '814' '746' '250.31' '556' '680'
 '745' '351' '306' '110' '695' '552' '346' '918' '882' '947' '520' '188'
 '31' '356' '737' 'V08' '322' '182' '517' '974' 'E929' 'V53' '912' '252'
 '608' '516' 'E933' '94' '702' '923' '594' '647' '111' '934' '430' '487'
 '709' '796' '156' '977' '915' '756' '840' '341' '259' '693' '725' 'V62'
 '528' '683' '953' '457' '501' 'E900' 'V09' '522' '919' '461' '506' '193'
 '483' 'E936' '717' '802' '335' 'V54' '320' '945' '906' '239' '454' '826'
 '823' 'E941' '226' '795' '684' '844' '250.33' '308' '615' '588' '712'
 '663' '706' '833' '741' '713' '533' 'E884' '586' '555' '755' 'E928' '742'
 '869' '962' 'V11' '543' '373' '870' '913' '152' '810' '965' '907' '908'
 '995' '845' '474' '442' '751' '323' '472' '464' '686' '250.32' '540'
 '251' '811' '652' '659' '851' '422' '815' '307' '325' '463' '992' '692'
 '521' '917' 'E965' '524' '916' 'E813' '173' '238' '137' '514' '312' '837'
 '355' '980' '622' '475' '500' '754' '261' '801' '868' '968' '381' '11'
 '250.21' '694' '610' '734' 'E814' '310' '130' '246' '892' '846' '634'
 '75' 'E927' 'E905' '183' '379' 'E917' '163' 'E868' '495' '747' '989'
 'E854' '240' '832' '605' '602' '644' 'V16' '35' 'V70' '376' '266' 'E918'
 '619' '477' '656' '46' '883' '171' 'V13' '698' '842' 'E850' '800' '269'
 '664' 'E887' '952' '164' 'E881' '527' '685' '366' '836' '27' 'V63' '865'
 '793' '232' '990' '52' '831' '327' '542' '806' '972' '862' 'E829' 'E919'
 '944' 'E916' '963' '316' '645' '347' 'V85' '374' 'V02' '748' '256' '186'
 '866' '975' '96' '395' '262' 'E819' '654' '994' '318' 'E826' '879' '674'
 '641' '822' '145' '797' '353' 'E938' 'E816' '948' '987' '99' '192'
 '250.3' 'E906' '534' '115' 'E818' 'E980' '360' '338' '529' '871' '750'
 '212' '302' '955' '141' '88' 'V25' '215' '350' 'V50' 'V03' 'E853' 'E968'
 'E882' '140' '703' '991' '893' 'E821' '235' 'V69' '670' '195' 'V55' '388'
 '268' '894' '114' '260' '853' '7' '880' 'V86' '180' 'E945' '523' '863'
 '649' '270' '665' '460' '942' '364' '66' 'E883' '123' '884' 'V60' '843'
 '927']

Unique values in 'diag_3': ['?' '255' 'V27' '403' '250' 'V45' '38' '486' '996' '197'
'250.6' '427'
 '627' '414' '416' '714' '428' '582' 'V43' '250.01' '263' '250.42' '276'
 '482' '401' '250.41' '585' '781' '278' '998' '568' '682' '618' '250.02'
 '305' '707' '496' '599' '715' '424' '518' '553' '794' '411' 'V42' '531'
 '511' '490' '562' '250.8' '250.7' '250.52' '784' '491' '581' '420' '8'
 '724' '730' '789' '131' '250.82' '999' '41' '493' '250.03' '753' '786'
 '529' 'E888' '425' '595' '303' '560' '711' '492' '332' '296' '438' '362'
 '250.4' '654' '244' 'V70' '737' '625' '681' '250.51' '404' 'V10' '810'
 '280' '440' '785' '588' '569' '272' '997' '250.43' '918' '584' '54' '788'
 '426' '722' '250.92' '196' '461' '535' '787' '891' '284' '458' '648'
 '780' '182' '285' '593' '413' '664' '564' '201' '356' 'V15' '292' '782'
 '473' '455' 'E932' '357' '348' '294' '250.23' '459' 'E878' '437' '733'
 '507' '525' '250.53' '397' '572' '805' '453' '331' '736' '402' '591'
 '576' '465' '533' '703' '349' '315' '658' '608' '578' '716' '382' '300'
 '282' '571' '536' '596' '287' '644' 'V11' '558' 'E885' '162' '198' '218'
 '412' '396' 'V14' '570' '433' 'E934' '882' '288' '577' '443' '729' '836'
 '295' '799' '281' '304' '153' '410' '616' '250.83' '601' '291' '75' '512'
 '660' '250.5' '598' '337' '574' '653' 'V58' '311' '415' '386' '602' '790'
 '112' '873' '620' '436' '70' '155' '138' '663' '530' '710' '42' '342'
 '250.91' 'E884' '515' '307' '704' '728' '731' '583' '238' '441' '293'
 '573' '532' '290' '594' '319' '250.13' '250.12' '519' '346' '380' '135'
 '642' '698' '924' '905' 'E933' '555' '309' 'E879' '286' '565' '752' '580'
 '446' '444' '344' '252' '35' '813' '394' '301' '575' '258' 'V17' '802'
 '435' '746' 'V12' '709' '881' 'E935' '139' '250.81' '718' '365' '202'
 '334' '185' '398' 'V44' '517' 'E849' '614' '466' '626' '250.9' '368'
 '605' '883' '289' '478' '617' '429' '442' 'V25' '866' '610' '557' '959'
 'E942' '94' '920' '345' '313' '379' '79' '516' '586' '821' '600' '242'
 '373' '592' 'V64' '487' '253' '706' 'E947' '117' '340' 'E950' '656'
 'E949' '590' 'V09' '250.22' '934' '694' '203' '250.93' '995' '726' '923'
 '958' '275' 'E929' '211' 'V18' 'V66' '199' '665' '53' '279' '522' '791'
 '890' '456' 'E938' 'E816' '122' '721' 'V65' '136' '480' '423' 'E920'
 '793' '647' '537' '351' '845' '336' '274' '719' '945' '434' '494' '227'
 '157' '208' '174' 'V57' '812' '734' '150' 'V23' '447' '692' '228' 'V16'
 '756' '405' 'E928' '823' '552' '528' '389' '240' '454' '792' '366' 'E939'
 '907' '270' '310' '266' '387' 'E931' '783' '245' '607' '355' 'E930' '705'
 '372' '369' '611' '283' 'V46' '110' '867' 'E956' '251' '250.2' '820'
 '712' '695' '567' '343' '723' 'V08' '273' '623' '807' '451' '495' '701'
 '34' 'V53' '314' '472' 'E945' '11' '189' '534' '354' '333' 'V54' '277'
 '659' '708' '452' '655' '816' '670' '621' '246' '953' '865' 'E817' '646'
 '151' '378' '78' '298' '840' '641' '521' '745' '619' '912' '506' 'E904'
 '259' 'E870' 'E980' '383' '204' '696' '566' '727' '47' 'E943' '358' '191'
 '965' '921' '432' '27' 'E861' '758' '477' '524' '751' '652' '556' '188'
 '825' '919' '732' '908' '951' '962' '685' 'E850' 'E944' '527' '341' '693'
 '250.1' 'V49' '860' '323' 'V55' '579' '508' '969' '205' '462' 'E880'
 '680' '697' '826' '200' '457' '717' '738' '742' '735' '235' '308' '725'
 '241' '824' '464' '260' '917' '239' '661' '892' '261' 'E883' '943' '744'
 'E936' '796' '318' '967' '350' '854' 'E905' '9' '741' 'E941' '170' '643'
 '317' '759' '909' 'V22' '831' '713' '180' '801' '360' '359' '501' '335'
 '250.11' '306' '811' '690' 'V02' '271' '214' '847' '543' 'V63' '906'
 '842' '686' '445' '808' '861' 'E852' '220' 'E887' 'E858' '915' '970'
 '256' '747' '395' '243' '815' '481' '5' 'E927' '297' '299' '851' '864'
 '922' '384' 'E876' '225' '158' 'E937' '871' '88' '966' 'E917' 'E812'
 'V62' 'E924' '604' '233' 'E916' '377' '797' 'V72' '172' '7' '421' '852'
 'E819' '972' '916' '956' '3' 'E965' '173' '193' '154' '347' '862' '250.3'
 '987' '470' '262' 'E855' '161' '115' '179' '910' '312' '17' '460' '265'
 '66' '163' 'V60' '870' 'E906' '514' '944' '844' '417' '152' '183' '991'
 '216' '385' '164' '935' '510' '814' '485' '850' '250.21' 'E919' '872'
 '195' '431' '597' '933' '171' '884' '156' '868' '483' 'E815' '542' 'V61'
 '853' '374' 'E881' 'E882' 'E822' '192' '754' '327' '523' '500' 'V85'
 '992' '657' '684' '603' 'E826' '550' '913' '376' '755' '361' '186' '720'
 '250.31' '674' '911' 'E813' '226' '365.44' 'E818' '146' '955' 'E894'
 '475' 'V13' '880' '930' 'E915' '381' '132' '353' '795' '893' 'V01' 'E853'
 '863' '540' 'E828' '430' '800' 'E865' '148' 'E946' '822' '879' '848'
 'V86' 'V03' '338' '989' '388' 'E966' '111' 'E922' '123' '757' 'E901'
 '141' '268' 'E892' '649' '702' '948' '223' '484' 'E886' '838' '928' '236'
 '624' '837' 'E987' 'V07' '841' '622' 'E912' 'E955' '463' 'V06' 'E864'
 '217' '877' '391' 'E825' '952' '669' '875' 'E900' '215' '538' '980' '834'
 '448' '175' '49' '876' '230' '57' 'E854' '942' '14' '750' '370' '671'
 '971']

Unique values in 'max_glu_serum': [nan '>300' 'Norm' '>200']

Unique values in 'A1Cresult': [nan '>7' '>8' 'Norm']

Unique values in 'metformin': ['No' 'Steady' 'Up' 'Down']

Unique values in 'repaglinide': ['No' 'Up' 'Steady' 'Down']

Unique values in 'nateglinide': ['No' 'Steady' 'Down' 'Up']

Unique values in 'chlorpropamide': ['No' 'Steady' 'Down' 'Up']

Unique values in 'glimepiride': ['No' 'Steady' 'Down' 'Up']

Unique values in 'acetohexamide': ['No' 'Steady']

Unique values in 'glipizide': ['No' 'Steady' 'Up' 'Down']

Unique values in 'glyburide': ['No' 'Steady' 'Up' 'Down']

Unique values in 'tolbutamide': ['No' 'Steady']

Unique values in 'pioglitazone': ['No' 'Steady' 'Up' 'Down']

Unique values in 'rosiglitazone': ['No' 'Steady' 'Up' 'Down']

Unique values in 'acarbose': ['No' 'Steady' 'Up' 'Down']

Unique values in 'miglitol': ['No' 'Steady' 'Down' 'Up']

Unique values in 'troglitazone': ['No' 'Steady']

Unique values in 'tolazamide': ['No' 'Steady' 'Up']

Unique values in 'examide': ['No']

Unique values in 'citoglipton': ['No']
Unique values in 'insulin': ['No' 'Up' 'Steady' 'Down']

Unique values in 'glyburide-metformin': ['No' 'Steady' 'Down' 'Up']

Unique values in 'glipizide-metformin': ['No' 'Steady']

Unique values in 'glimepiride-pioglitazone': ['No' 'Steady']

Unique values in 'metformin-rosiglitazone': ['No' 'Steady']

Unique values in 'metformin-pioglitazone': ['No' 'Steady']

Unique values in 'change': ['No' 'Ch']

Unique values in 'diabetesMed': ['No' 'Yes']

Unique values in 'readmitted': ['NO' '>30' '<30']
```

Data Preprocessing

In [8]:
```python
df.replace('?', np.nan, inplace=True)
```

In [9]:
```python
missing_values = df.isnull().sum()
print(missing_values[missing_values > 0])
```

```
race                  2273
weight               98569
payer_code           40256
medical_specialty    49949
diag_1                  21
diag_2                 358
diag_3                1423
max_glu_serum        96420
A1Cresult            84748
dtype: int64
```

In [10]:
```python
# Drop columns with multiple missing values
df = df.drop(columns=['weight', 'payer_code', 'medical_specialty','discharge_dispos
```

In [11]:
```python
# Drop rows where the race column has missing values
df = df.dropna(subset=['race'])
```

In [12]:
```python
# Drop columns with only one unique value
df = df.drop(columns=['examide', 'citoglipton','metformin-rosiglitazone'])
```

In [13]:
```python
df['max_glu_serum'].unique()
```

Out[13]:
```
array([nan, '>300', 'Norm', '>200'], dtype=object)
```

In [14]:
```python
# Replace the values in the 'max_glu_serum' column with more descriptive categories
df['max_glu_serum'] = df['max_glu_serum'].replace({
    '>300': 'Very High',
    '>200': 'High',
    'Norm': 'Normal',
    np.nan: 'No test performed'
})

print(df['max_glu_serum'].unique())
```

```
['No test performed' 'Very High' 'Normal' 'High']
```

In [15]:

```
df['A1Cresult'].unique()
```

Out[15]:

```
array([nan, '>7', '>8', 'Norm'], dtype=object)
```

In [16]:

```
# Replace the values in the 'A1Cresult' column with more descriptive categories
df['A1Cresult'] = df['A1Cresult'].replace({
    '>8': 'Very High',
    '>7': 'High',
    'Norm': 'Normal',
    np.nan: 'No test performed'
})

print(df['A1Cresult'].unique())
```

```
['No test performed' 'High' 'Very High' 'Normal']
```

In [17]:

```
# Mapping of diagnosis categories with corresponding ICD-9 code ranges
categories = {
    'Circulatory': ['390-459', '785'],
    'Respiratory': ['460-519', '786'],
    'Digestive': ['520-579', '787'],
    'Diabetes': ['250.xx'],
    'Injury': ['800-999'],
    'Musculoskeletal': ['710-739'],
    'Genitourinary': ['580-629', '788'],
    'Neoplasms': ['140-239','780', '781', '784', '790-799','240-249', '251-279','68
    'Other': [
        '290-319', 'E', 'V', '280-289', '320-359', '630-679', '360-389',
        '740-759'
    ]
}

def categorize_diagnosis(code):
    if pd.isna(code):
        return 'Unknown'
    code = str(code)

    for category, ranges in categories.items():
        for icd_range in ranges:
            if '-' in icd_range:
                start, end = icd_range.split('-')
                if start <= code <= end:
                    return category
            elif icd_range.endswith('xx') and code.startswith(icd_range[:3]):
                return category
            elif code.startswith(icd_range):
                return category
    return 'Other'
```

```
df['primary_diagnosis'] = df['diag_1'].apply(categorize_diagnosis)
df['secondary_diagnosis'] = df['diag_2'].apply(categorize_diagnosis)
df['additional_diagnosis'] = df['diag_3'].apply(categorize_diagnosis)


df[['diag_1', 'primary_diagnosis', 'diag_2', 'secondary_diagnosis', 'diag_3', 'addi
```

Out[17]:

	diag_1	primary_diagnosis	diag_2	secondary_diagnosis	diag_3	additional_diagnosis
0	250.83	Diabetes	NaN	Unknown	NaN	Unknown
1	276	Neoplasms	250.01	Diabetes	255	Neoplasms
2	648	Other	250	Diabetes	V27	Other
3	8	Other	250.43	Diabetes	403	Circulatory
4	197	Neoplasms	157	Neoplasms	250	Diabetes

In [18]:

```
df = df.drop(columns=['diag_1', 'diag_2', 'diag_3'])
```

In [19]:

```
# Map the numerical 'admission_type_id' to descriptive labels
admission_type_mapping = {
    1: 'Emergency',
    2: 'Urgent',
    3: 'Elective',
    4: 'Newborn',
    5: 'Not Available',
    6: 'NULL',
    7: 'Trauma Center',
    8: 'Not Mapped'
}

df['admission_type'] = df['admission_type_id'].replace(admission_type_mapping)
df = df.drop(columns=['admission_type_id'])

print(df[['admission_type']].head())
```

```
  admission_type
0           NULL
1      Emergency
2      Emergency
3      Emergency
4      Emergency
```

In [20]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 99493 entries, 0 to 101765
Data columns (total 42 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   encounter_id              99493 non-null  int64 
 1   patient_nbr               99493 non-null  int64 
 2   race                      99493 non-null  object
 3   gender                    99493 non-null  object
 4   age                       99493 non-null  object
 5   time_in_hospital          99493 non-null  int64 
 6   num_lab_procedures        99493 non-null  int64 
 7   num_procedures            99493 non-null  int64 
 8   num_medications           99493 non-null  int64 
 9   number_outpatient         99493 non-null  int64 
 10  number_emergency          99493 non-null  int64 
 11  number_inpatient          99493 non-null  int64 
 12  number_diagnoses          99493 non-null  int64 
 13  max_glu_serum             99493 non-null  object
 14  A1Cresult                 99493 non-null  object
 15  metformin                 99493 non-null  object
 16  repaglinide               99493 non-null  object
 17  nateglinide               99493 non-null  object
 18  chlorpropamide            99493 non-null  object
 19  glimepiride               99493 non-null  object
 20  acetohexamide             99493 non-null  object
 21  glipizide                 99493 non-null  object
 22  glyburide                 99493 non-null  object
 23  tolbutamide               99493 non-null  object
 24  pioglitazone              99493 non-null  object
 25  rosiglitazone             99493 non-null  object
 26  acarbose                  99493 non-null  object
 27  miglitol                  99493 non-null  object
 28  troglitazone              99493 non-null  object
 29  tolazamide                99493 non-null  object
 30  insulin                   99493 non-null  object
 31  glyburide-metformin       99493 non-null  object
 32  glipizide-metformin       99493 non-null  object
 33  glimepiride-pioglitazone  99493 non-null  object
 34  metformin-pioglitazone    99493 non-null  object
 35  change                    99493 non-null  object
 36  diabetesMed               99493 non-null  object
 37  readmitted                99493 non-null  object
 38  primary_diagnosis         99493 non-null  object
 39  secondary_diagnosis       99493 non-null  object
 40  additional_diagnosis      99493 non-null  object
 41  admission_type            99493 non-null  object
dtypes: int64(10), object(32)
memory usage: 32.6+ MB
```

Visualizations:

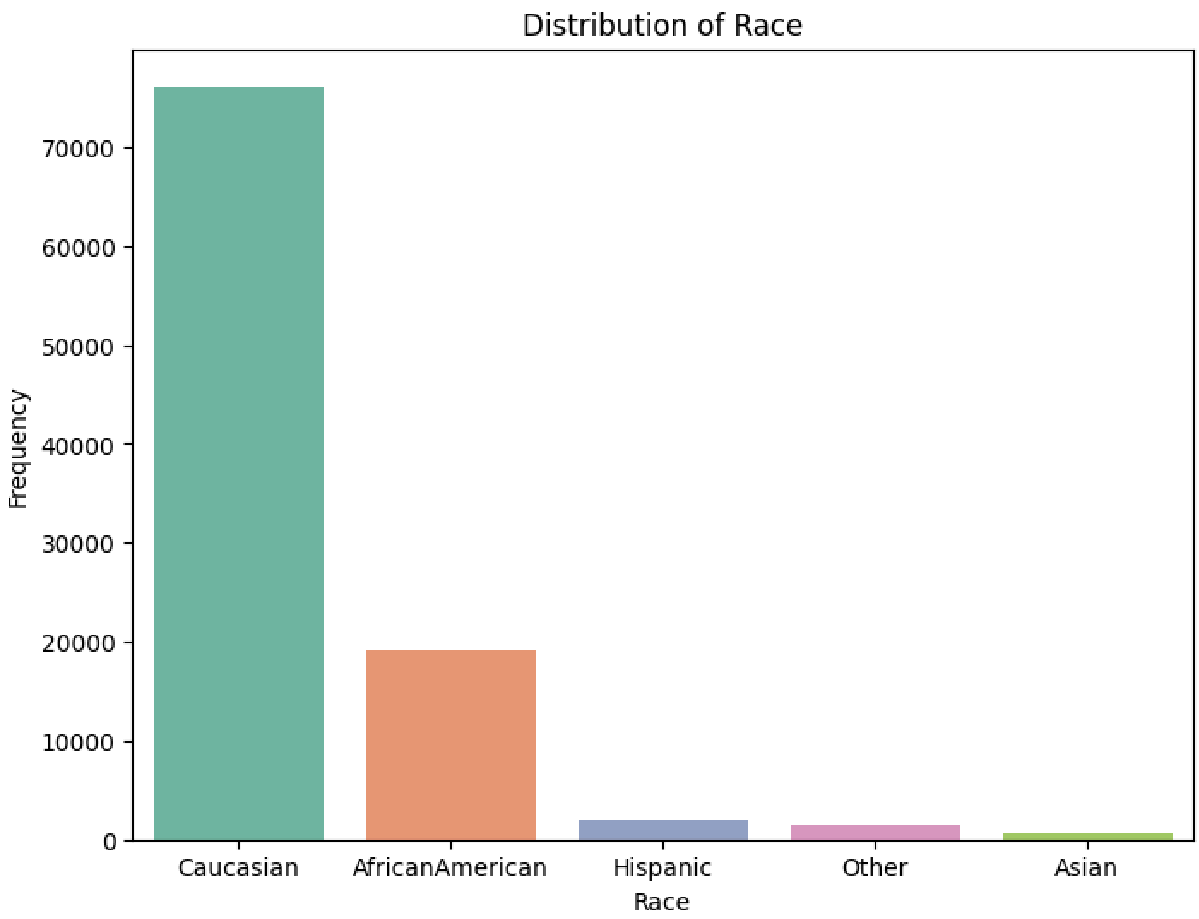
The plots below showcase how the different columns like Race, Gender, Age Groups are spread in our dataset

In [21]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Race distribution
race_counts = df['race'].value_counts()
print(race_counts)

# Visualization: Race distribution
plt.figure(figsize=(8, 6))
sns.barplot(x=race_counts.index, y=race_counts.values, palette='Set2')
plt.title('Distribution of Race')
plt.xlabel('Race')
plt.ylabel('Frequency')
plt.show()

# Gender distribution
gender_counts = df['gender'].value_counts()
print(gender_counts)

# Visualization: Gender distribution
plt.figure(figsize=(8, 6))
sns.barplot(x=gender_counts.index, y=gender_counts.values, palette='Set1')
plt.title('Distribution of Gender')
plt.xlabel('Gender')
plt.ylabel('Frequency')
plt.show()

# Age distribution
age_counts = df['age'].value_counts()
print(age_counts)

# Visualization: Age distribution
plt.figure(figsize=(10, 6))
sns.barplot(x=age_counts.index, y=age_counts.values, palette='muted')
plt.title('Distribution of Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```

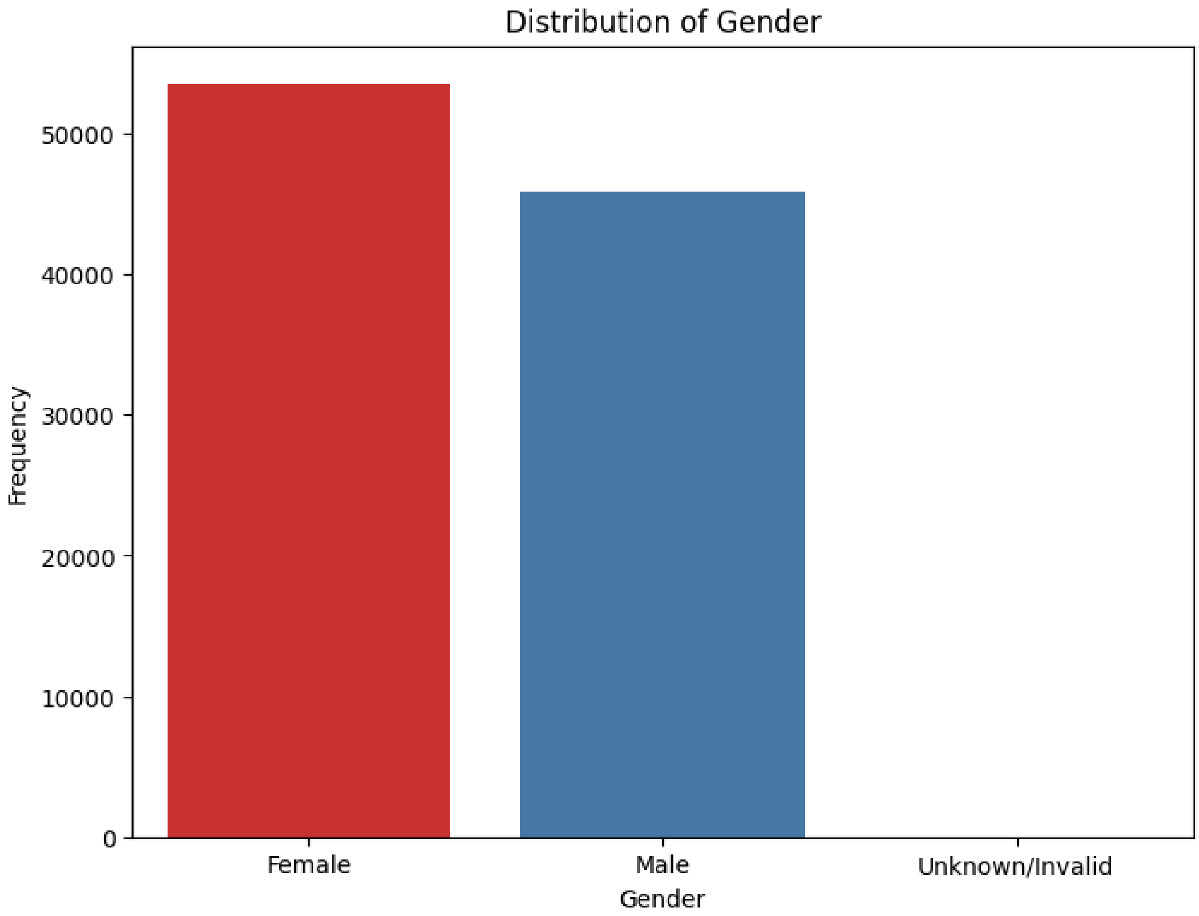
```
race
Caucasian          76099
AfricanAmerican    19210
Hispanic            2037
Other               1506
Asian                641
Name: count, dtype: int64
```

```
<ipython-input-21-496251125ede>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=race_counts.index, y=race_counts.values, palette='Set2')
```



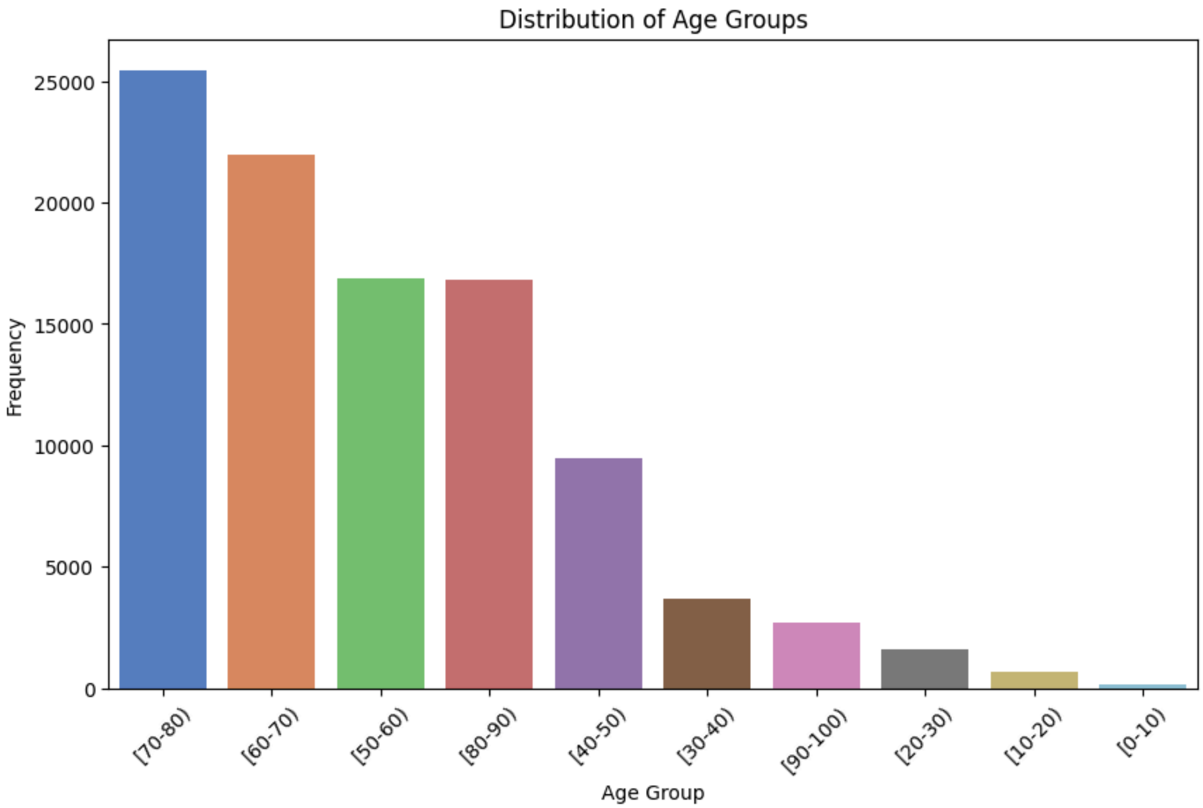
```
gender
Female            53575
Male              45917
Unknown/Invalid       1
Name: count, dtype: int64
```

```
<ipython-input-21-496251125ede>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=gender_counts.index, y=gender_counts.values, palette='Set1')
```



```
age
[70-80)     25469
[60-70)     21988
[50-60)     16895
[80-90)     16800
[40-50)      9465
[30-40)      3699
[90-100)     2724
[20-30)      1611
[10-20)       682
[0-10)        160
Name: count, dtype: int64
```

```
<ipython-input-21-496251125ede>:34: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=age_counts.index, y=age_counts.values, palette='muted')
```

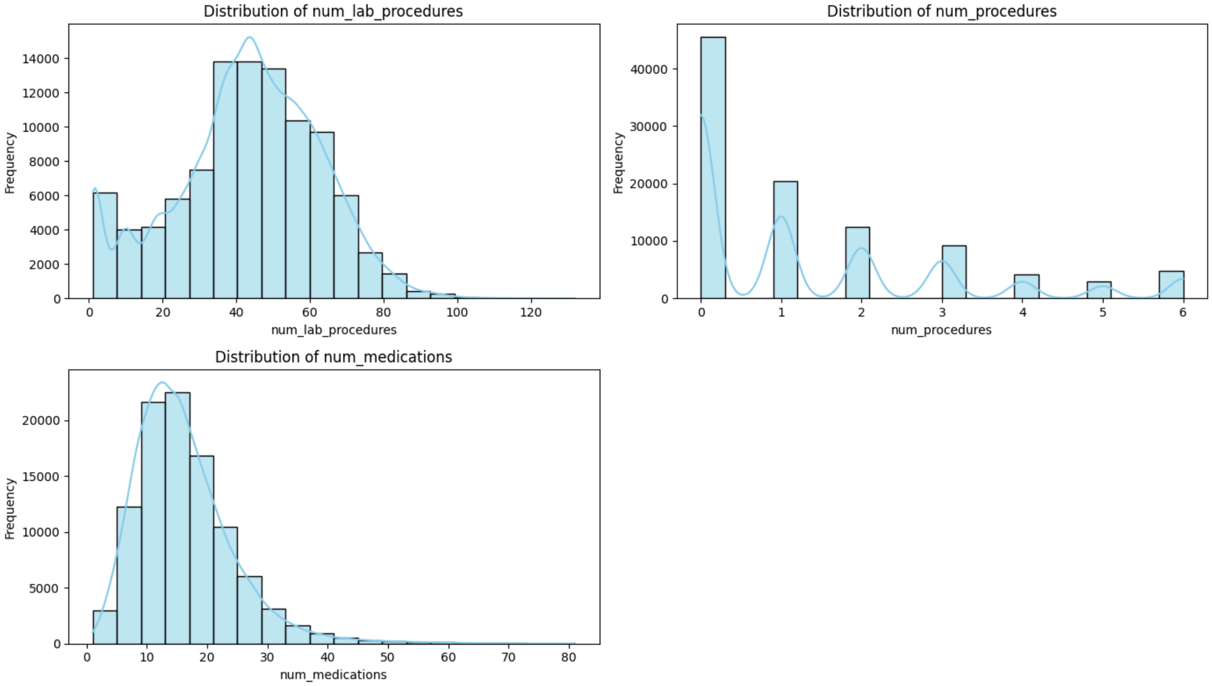


The plot presents the distribution of the number of lab procedures, procedures, and medications for a group of patients. The histograms show that the distribution of lab procedures and medications is right-skewed, meaning there are a few patients with many procedures and medications. At the same time, the majority have a lower number. The number of procedures is more evenly distributed, with a peak around 1. This suggests that most patients undergo only one procedure during their hospital stay.

In [22]:

```
# Select the features of interest
features = ['num_lab_procedures', 'num_procedures', 'num_medications']

# Summary Statistics
summary_stats = df[features].describe(percentiles=[.25, .5, .75])
print("Summary Statistics for Lab and Procedure Counts:")
print(summary_stats)

# Visualization: Histograms for each feature
plt.figure(figsize=(14, 8))
for i, feature in enumerate(features, 1):
    plt.subplot(2, 2, i)
    sns.histplot(df[feature], bins=20, kde=True, color='skyblue')
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

```
Summary Statistics for Lab and Procedure Counts:
       num_lab_procedures  num_procedures  num_medications
count        99493.000000    99493.000000     99493.000000
mean            43.072588        1.340577        16.026605
std             19.695858        1.703717         8.119790
min              1.000000        0.000000         1.000000
25%             31.000000        0.000000        10.000000
50%             44.000000        1.000000        15.000000
75%             57.000000        2.000000        20.000000
max            132.000000        6.000000        81.000000
```



The plots illustrate the frequency of medication changes and diabetes medication prescriptions within a dataset. The left plot shows that a majority of patients experienced medication changes (1 represents a change, 0 represents no change).

The right plot indicates that a significant proportion of patients were prescribed diabetes medication (1 represents a prescription, 0 represents no prescription). Both plots highlight the prevalence of medication adjustments and diabetes management within the studied population.

The plot presents the frequency of medication changes based on diabetes medication status. It shows that patients who were prescribed diabetes medication (indicated by 1) were more likely to experience medication changes compared to those who were not prescribed diabetes medication (indicated by 0). This suggests that diabetes medication use is associated with a higher frequency of medication adjustments, potentially due to the need for careful management of blood sugar levels and other related factors.

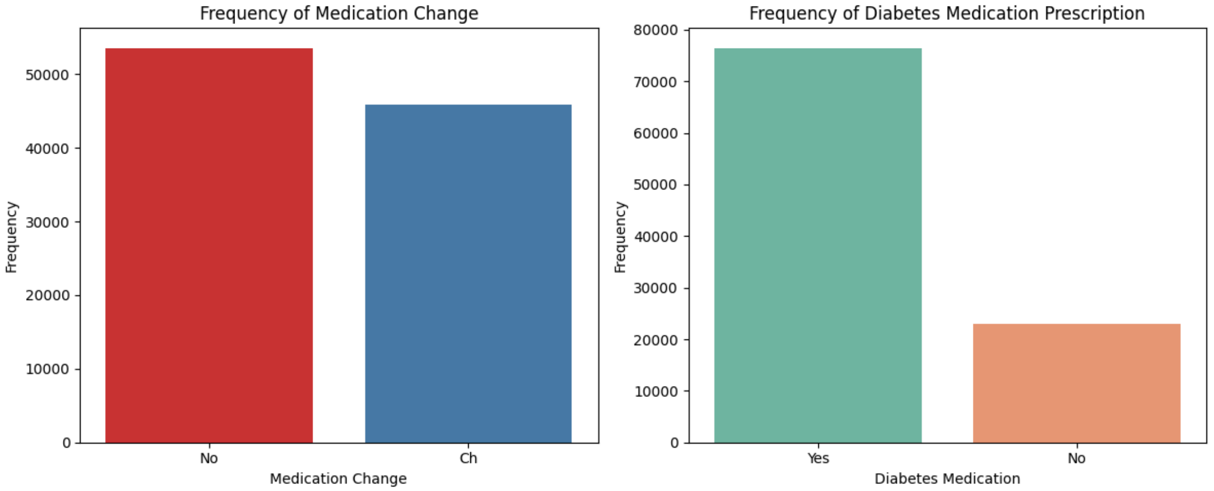
In [23]:

```
# Frequency counts for 'change' and 'diabetesMed'
change_counts = df['change'].value_counts()
diabetesMed_counts = df['diabetesMed'].value_counts()

# Display frequency counts
```

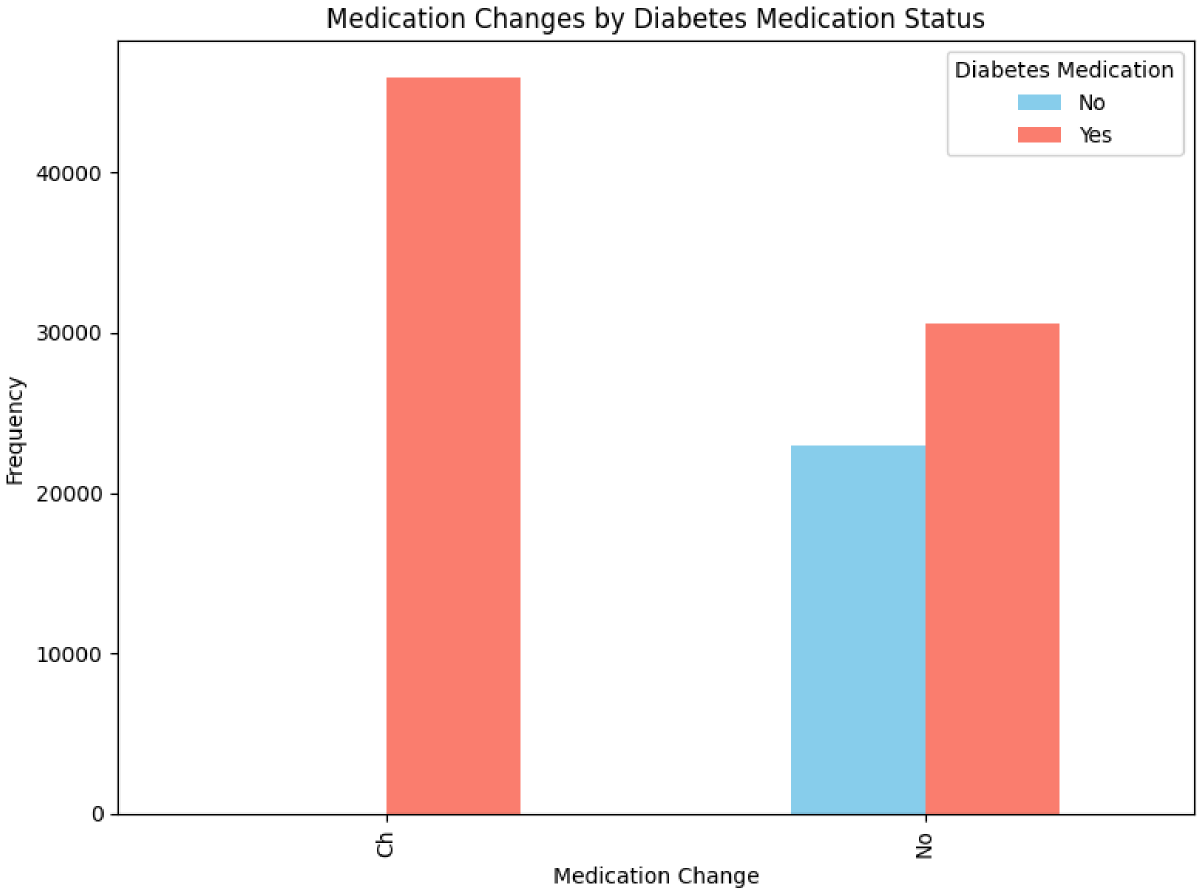
```
print("Frequency of Medication Changes (change):")
print(change_counts)
print("\nFrequency of Diabetes Medication (diabetesMed):")
print(diabetesMed_counts)

# Visualization: Bar charts for 'change' and 'diabetesMed'
plt.figure(figsize=(12, 5))

# Bar chart for 'change'
plt.subplot(1, 2, 1)
sns.barplot(x=change_counts.index, y=change_counts.values, palette='Set1')
plt.title('Frequency of Medication Change')
plt.xlabel('Medication Change')
plt.ylabel('Frequency')

# Bar chart for 'diabetesMed'
plt.subplot(1, 2, 2)
sns.barplot(x=diabetesMed_counts.index, y=diabetesMed_counts.values, palette='Set2'
plt.title('Frequency of Diabetes Medication Prescription')
plt.xlabel('Diabetes Medication')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()

# Cross-tabulation of 'change' and 'diabetesMed'
change_diabetesMed_ct = pd.crosstab(df['change'], df['diabetesMed'])

# Display cross-tabulation
print("\nCross-tabulation of Change and Diabetes Medication:")
print(change_diabetesMed_ct)

# Visualization: Side-by-side bar chart for cross-tabulation
change_diabetesMed_ct.plot(kind='bar', color=['skyblue', 'salmon'], figsize=(8, 6))
plt.title('Medication Changes by Diabetes Medication Status')
plt.xlabel('Medication Change')
plt.ylabel('Frequency')
plt.legend(title='Diabetes Medication')
plt.tight_layout()
plt.show()
```

```
Frequency of Medication Changes (change):
change
No    53582
Ch    45911
Name: count, dtype: int64

Frequency of Diabetes Medication (diabetesMed):
diabetesMed
Yes    76492
No     23001
Name: count, dtype: int64
```

```
<ipython-input-23-cb4b548372a9>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=change_counts.index, y=change_counts.values, palette='Set1')
<ipython-input-23-cb4b548372a9>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=diabetesMed_counts.index, y=diabetesMed_counts.values, palette='Set
2')
```



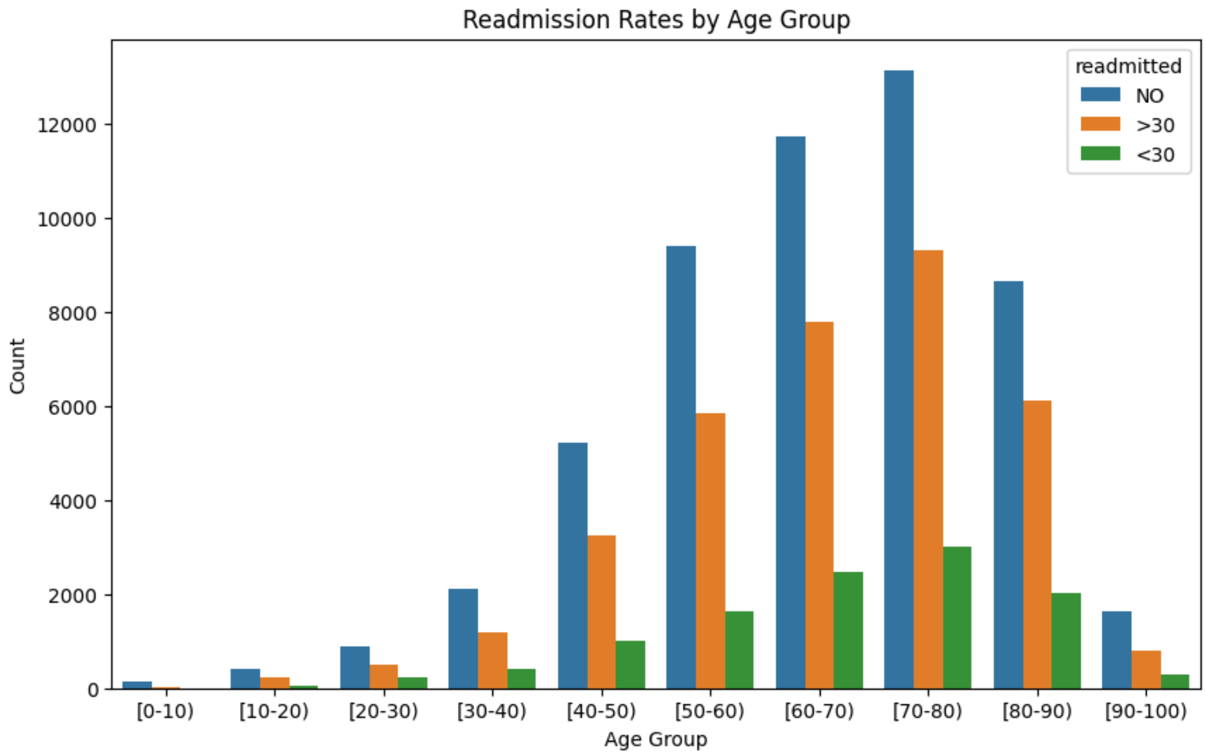
```
Cross-tabulation of Change and Diabetes Medication:
diabetesMed     No    Yes
change
Ch               0  45911
No           23001  30581
```



The plot presents the readmission rates of patients based on their age group. It shows the count of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their age group. The plot reveals that the highest number of patients were in the 50-60 age group, and the readmission rates were highest in the 70-80 age group, particularly within 30 days.

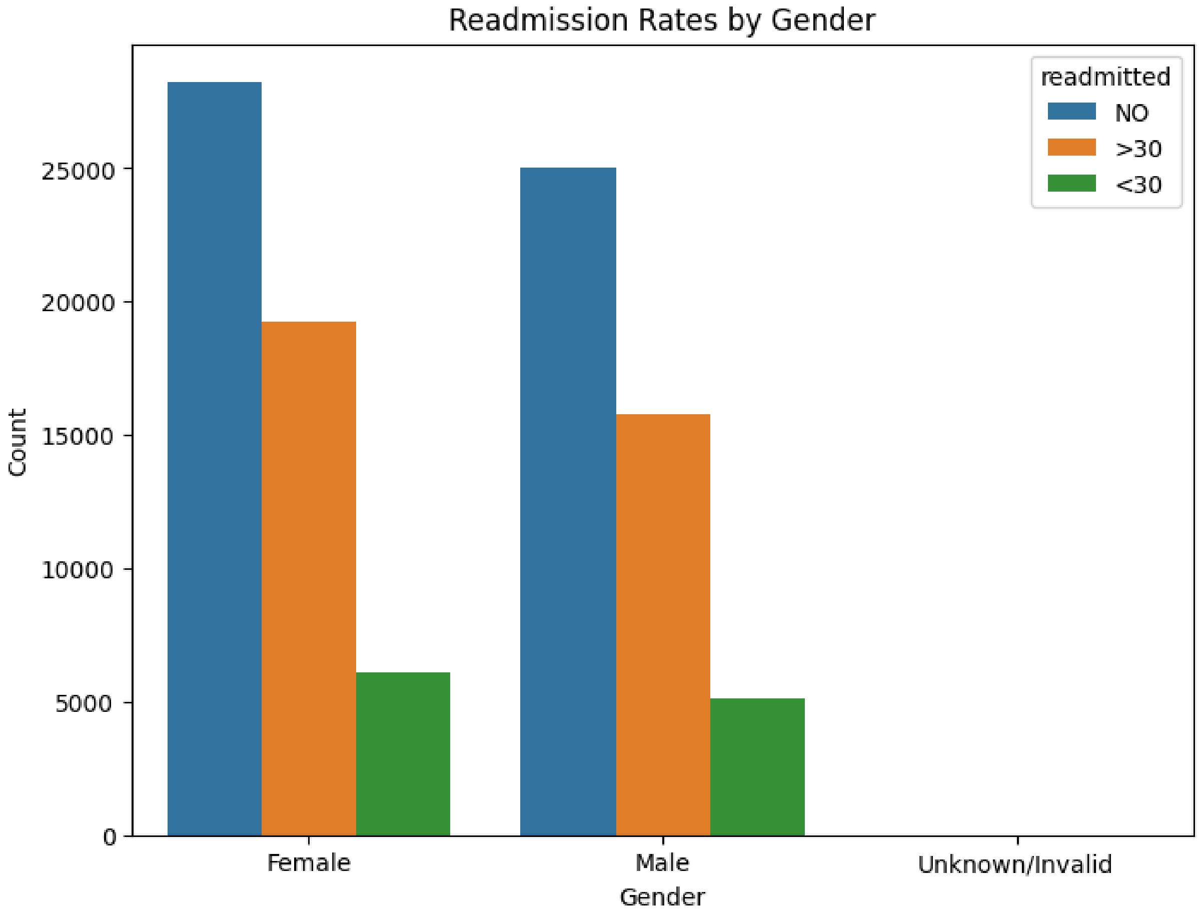
In [24]:

```
plt.figure(figsize=(10, 6))
sns.countplot(x='age', hue='readmitted', data=df)
plt.title('Readmission Rates by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.show()
```



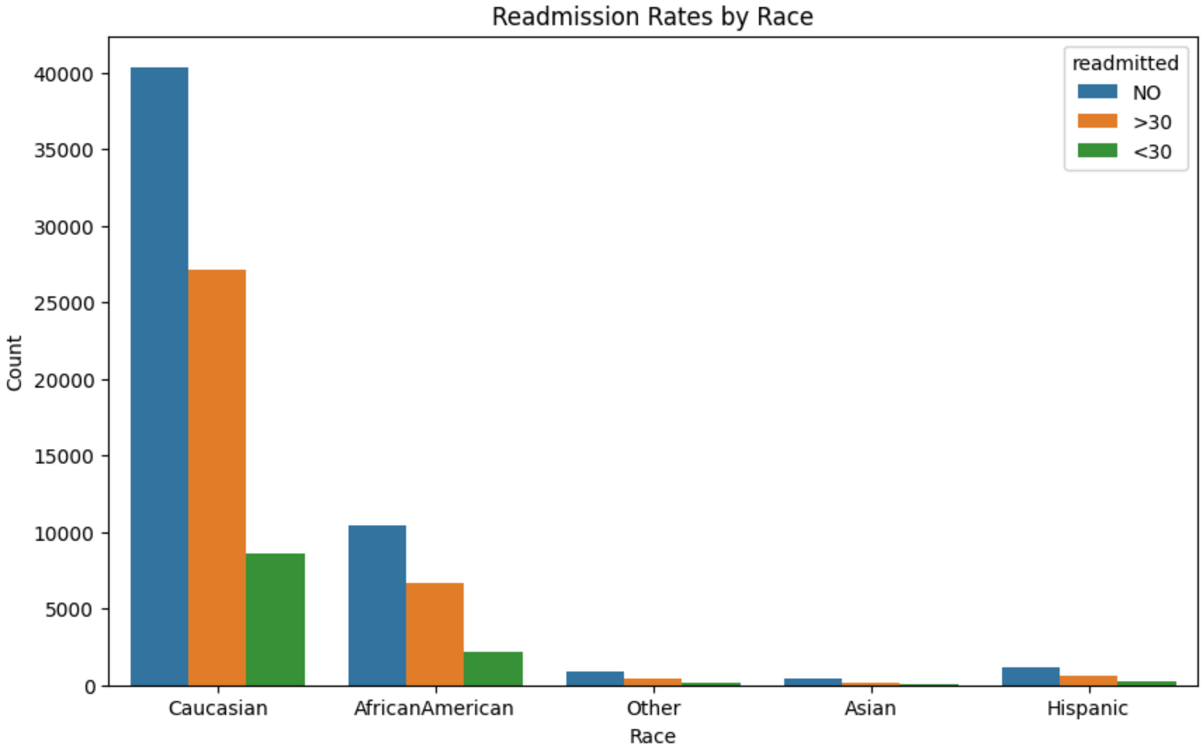
The plot presents the readmission rates of patients based on their gender. It shows the count of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their gender. The plot reveals that the highest number of patients were female, and the readmission rates were slightly higher for females compared to males, particularly within 30 days.

In [25]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='gender', hue='readmitted', data=df)
plt.title('Readmission Rates by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



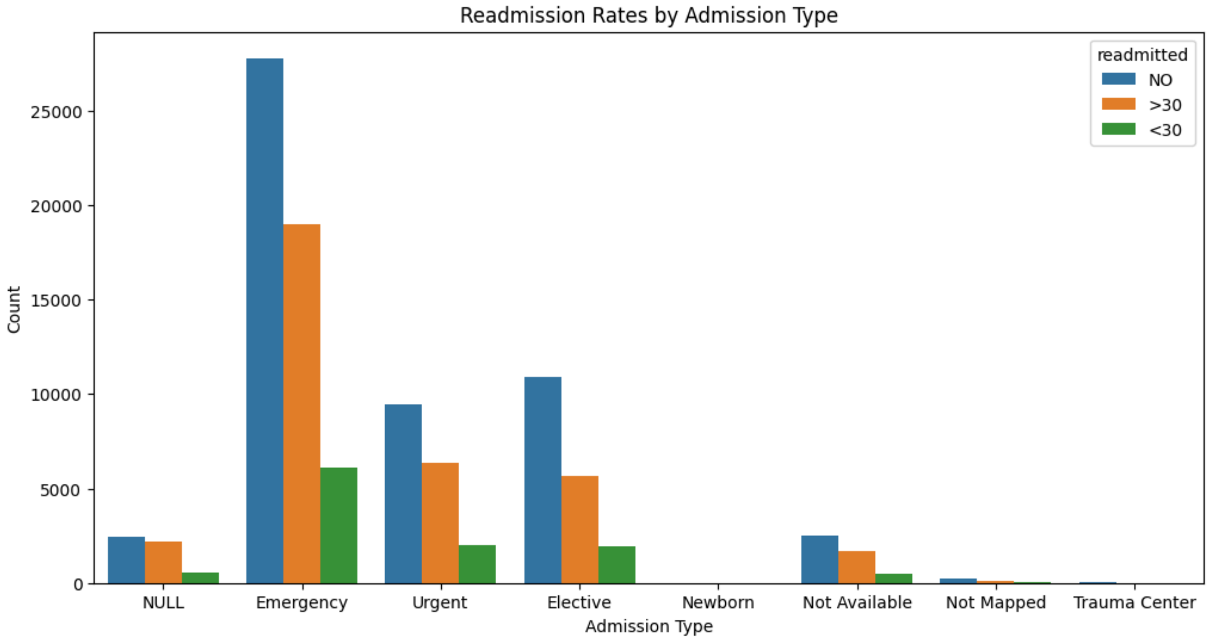
The plot presents the readmission rates of patients based on their race. It shows the count of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their race. The plot reveals that the highest number of patients were Caucasian, followed by African Americans. The readmission rates were highest for Caucasians, particularly within 30 days.

In [26]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='race', hue='readmitted', data=df)
plt.title('Readmission Rates by Race')
plt.xlabel('Race')
plt.ylabel('Count')
plt.show()
```



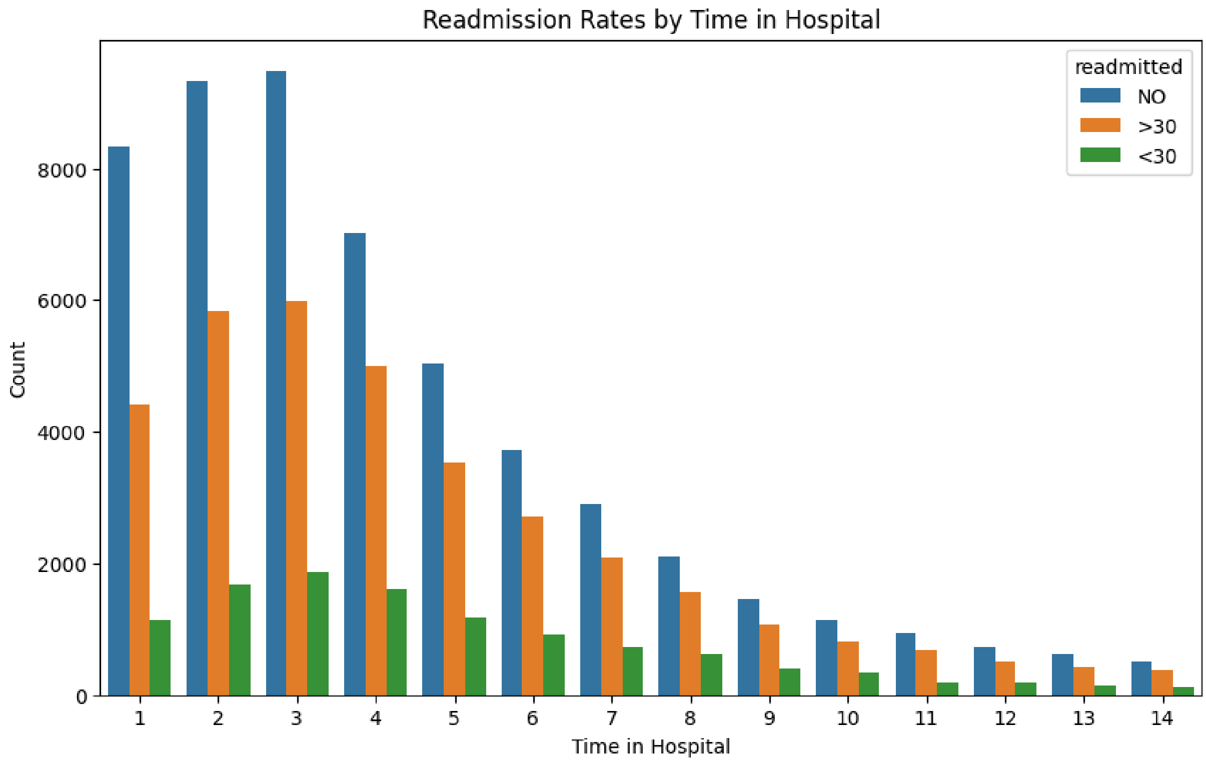
The plot presents the readmission rates of patients based on their admission type. It shows the count of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their admission type. The plot reveals that the highest number of patients were admitted via Emergency, and the readmission rates were highest for Emergency and Urgent admissions, particularly within 30 days.

In [27]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='admission_type', hue='readmitted', data=df)
plt.title('Readmission Rates by Admission Type')
plt.xlabel('Admission Type')
plt.ylabel('Count')
plt.show()
```



The plot presents the readmission rates of patients based on the time they spent in the hospital. It shows the count of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their length of stay. The plot reveals that the readmission rates decrease as the length of stay increases. This suggests that longer hospital stays might be associated with fewer readmissions.

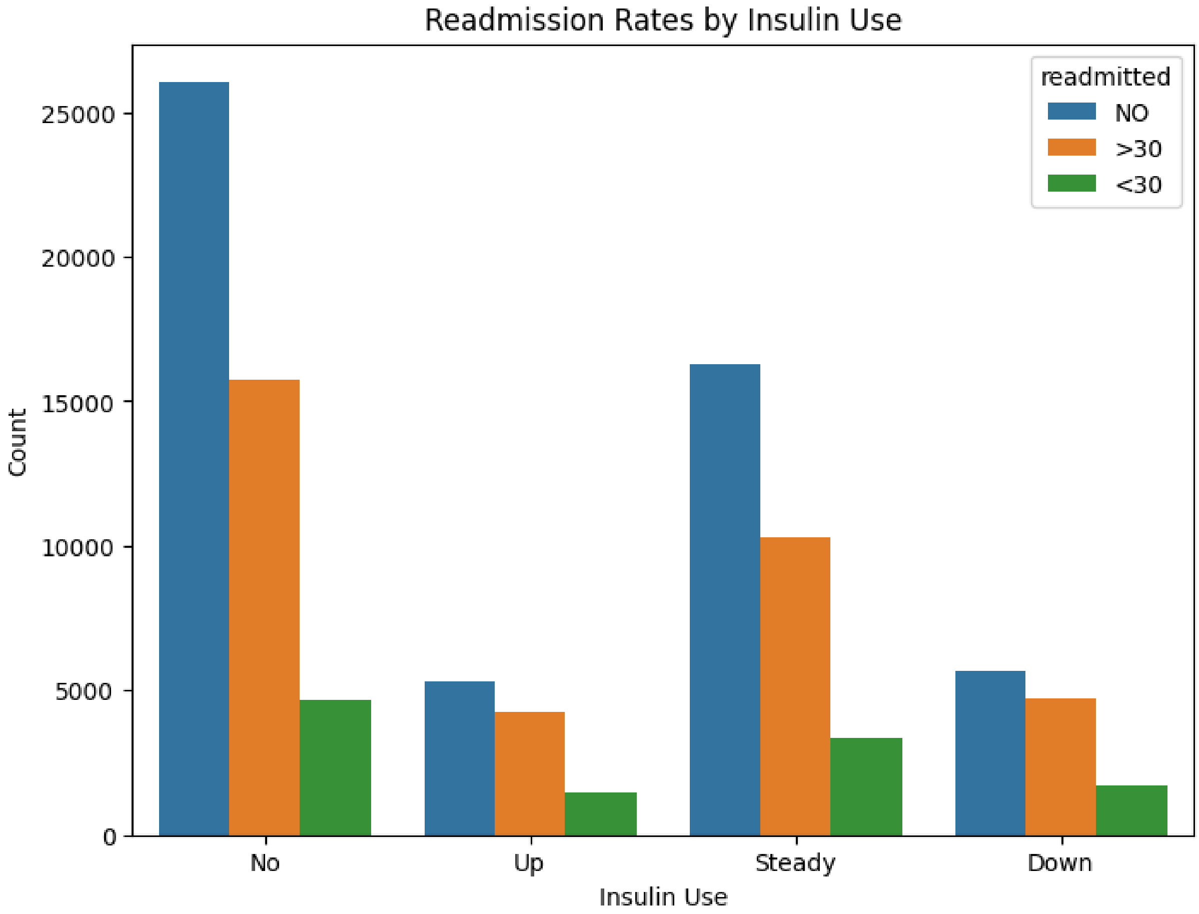
In [28]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='time_in_hospital', hue='readmitted', data=df)
plt.title('Readmission Rates by Time in Hospital')
plt.xlabel('Time in Hospital')
plt.ylabel('Count')
plt.show()
```



The plot presents the readmission rates of patients based on their insulin use. It shows the count of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by whether they were not on insulin, had their insulin dose increased, had their insulin dose kept steady, or had their insulin dose decreased. The plot reveals that patients with no insulin use had the highest readmission rates, particularly within 30 days. Patients with increased insulin dose had the lowest readmission rates.

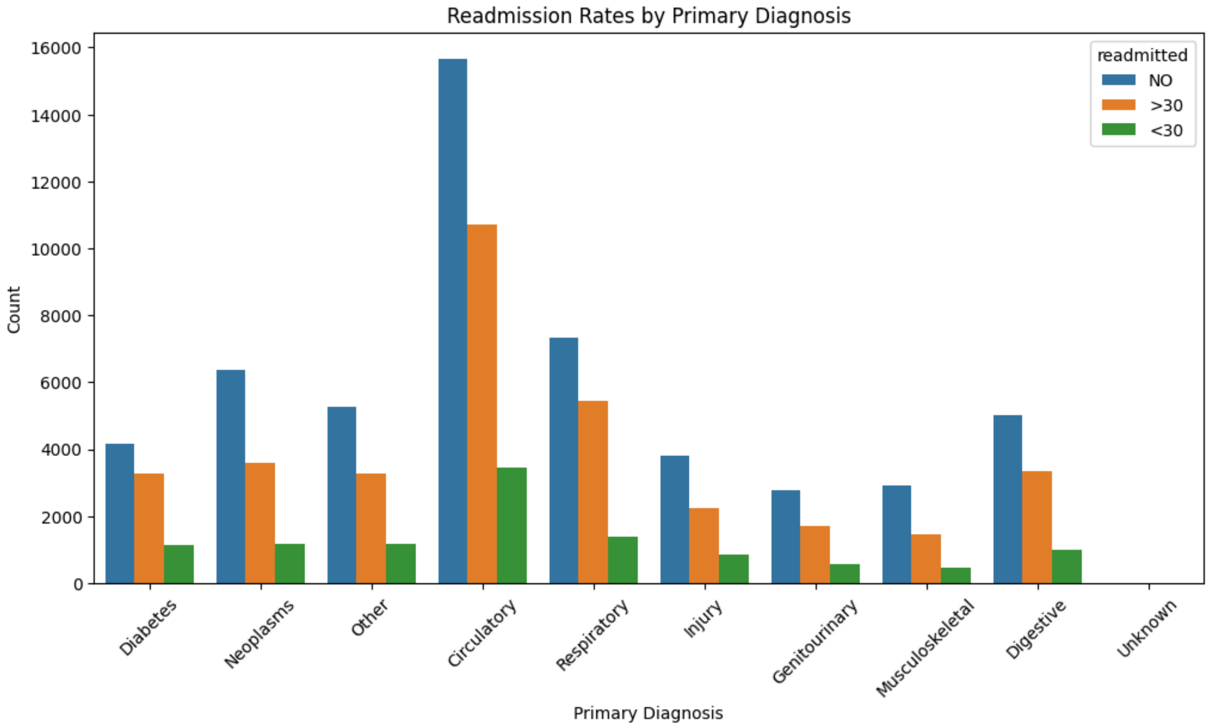
In [29]:

```
plt.figure(figsize=(8, 6))
sns.countplot(x='insulin', hue='readmitted', data=df)
plt.title('Readmission Rates by Insulin Use')
plt.xlabel('Insulin Use')
plt.ylabel('Count')
plt.show()
```



The plot presents the readmission rates of patients based on their primary diagnosis. It shows the count of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their primary diagnosis. The plot reveals that the highest number of patients were admitted with Circulatory and Diabetes as the primary diagnosis. Circulatory diagnoses had the highest readmission rates, particularly within 30 days.

In [30]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='primary_diagnosis', hue='readmitted', data=df)
plt.title('Readmission Rates by Primary Diagnosis')
plt.xlabel('Primary Diagnosis')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



In [31]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Assuming your DataFrame is called 'df'

def plot_readmission_overview(df):
    """Create an overview of readmission distribution"""
    plt.figure(figsize=(7, 3))

    # Calculate readmission distribution
    readmission_dist = df['readmitted'].value_counts()

    # Create bar plot
    sns.barplot(x=readmission_dist.index, y=readmission_dist.values)
    plt.title('Distribution of Readmission Status', fontsize=14)
    plt.xlabel('Readmission Status', fontsize=12)
    plt.ylabel('Number of Patients', fontsize=12)

    # Add percentage labels on top of bars
    total = len(df)
    for i, v in enumerate(readmission_dist.values):
        plt.text(i, v, f'{(v/total)*100:.1f}%', ha='center', va='bottom')

    plt.tight_layout()
    plt.show()

def plot_age_readmission(df):
    """Analyze readmission patterns across age groups"""
    plt.figure(figsize=(7, 3))

    # Create cross-tabulation of age and readmission
    age_readmission = pd.crosstab(df['age'], df['readmitted'], normalize='index') *

    # Create stacked bar plot
    age_readmission.plot(kind='bar', stacked=True)
    plt.title('Readmission Rates by Age Group', fontsize=14)
    plt.xlabel('Age Group', fontsize=12)
    plt.ylabel('Percentage', fontsize=12)
    plt.legend(title='Readmission Status')
    plt.xticks(rotation=45)

    plt.tight_layout()
    plt.show()

def plot_diagnosis_impact(df):
    """Analyze impact of primary diagnosis on readmission"""
    plt.figure(figsize=(7, 3))

    # Calculate readmission rates by diagnosis
    diagnosis_readmission = pd.crosstab(df['primary_diagnosis'],
                                      df['readmitted'].map({'NO': 'Not Readmitted',
                                                          '<30': 'Within 30 Days',
                                                          '>30': 'After 30 Days'}),
                                      normalize='index') * 100

    # Create heatmap
    sns.heatmap(diagnosis_readmission, annot=True, fmt='.1f', cmap='YlOrRd')
    plt.title('Readmission Rates by Primary Diagnosis (%)', fontsize=14)
    plt.xlabel('Readmission Status', fontsize=12)
    plt.ylabel('Primary Diagnosis', fontsize=12)

    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(7, 3))

    # Calculate readmission rates by diagnosis
    diagnosis_readmission = pd.crosstab(df['secondary_diagnosis'],
                                      df['readmitted'].map({'NO': 'Not Readmitted',
                                                          '<30': 'Within 30 Days',
                                                          '>30': 'After 30 Days'}),
                                      normalize='index') * 100

    # Create heatmap
    sns.heatmap(diagnosis_readmission, annot=True, fmt='.1f', cmap='YlOrRd')
    plt.title('Readmission Rates by Secondary Diagnosis (%)', fontsize=14)
    plt.xlabel('Readmission Status', fontsize=12)
    plt.ylabel('Secondary Diagnosis', fontsize=12)

    plt.tight_layout()
    plt.show()

    plt.figure(figsize=(7, 3))

    # Calculate readmission rates by diagnosis
    diagnosis_readmission = pd.crosstab(df['additional_diagnosis'],
                                      df['readmitted'].map({'NO': 'Not Readmitted',
                                                          '<30': 'Within 30 Days',
                                                          '>30': 'After 30 Days'}),
                                      normalize='index') * 100

    # Create heatmap
    sns.heatmap(diagnosis_readmission, annot=True, fmt='.1f', cmap='YlOrRd')
    plt.title('Readmission Rates by Additional Diagnosis (%)', fontsize=14)
    plt.xlabel('Readmission Status', fontsize=12)
    plt.ylabel('Additional Diagnosis', fontsize=12)

    plt.tight_layout()
    plt.show()

def plot_time_in_hospital_analysis(df):
    """Analyze relationship between time in hospital and readmission"""
    plt.figure(figsize=(7, 3))

    # Create violin plot
    sns.violinplot(x='readmitted', y='time_in_hospital', data=df)
    plt.title('Time in Hospital Distribution by Readmission Status', fontsize=14)
    plt.xlabel('Readmission Status', fontsize=12)
    plt.ylabel('Days in Hospital', fontsize=12)

    plt.tight_layout()
    plt.show()

def plot_medication_impact(df):
    """Analyze impact of number of medications on readmission"""
    plt.figure(figsize=(7, 3))

    # Create box plot
    sns.boxplot(x='readmitted', y='num_medications', data=df)
    plt.title('Number of Medications vs Readmission Status', fontsize=14)
    plt.xlabel('Readmission Status', fontsize=12)
    plt.ylabel('Number of Medications', fontsize=12)

    plt.tight_layout()
    plt.show()

def plot_lab_procedures_analysis(df):
    """Analyze relationship between lab procedures and readmission"""
    plt.figure(figsize=(7, 3))

    # Calculate average number of lab procedures for each readmission status
    lab_proc_means = df.groupby('readmitted')['num_lab_procedures'].mean()

    # Create bar plot
    sns.barplot(x=lab_proc_means.index, y=lab_proc_means.values)
    plt.title('Average Number of Lab Procedures by Readmission Status', fontsize=14
    plt.xlabel('Readmission Status', fontsize=12)
    plt.ylabel('Average Number of Lab Procedures', fontsize=12)

    # Add value labels on top of bars
    for i, v in enumerate(lab_proc_means.values):
        plt.text(i, v, f'{v:.1f}', ha='center', va='bottom')
    plt.tight_layout()
    plt.show()

# Function to run all visualizations
def analyze_readmissions(df):
    """Run all visualization analyses"""
    plot_readmission_overview(df)
    plot_age_readmission(df)
    plot_diagnosis_impact(df)
    plot_time_in_hospital_analysis(df)
    plot_medication_impact(df)
    plot_lab_procedures_analysis(df)
```

The plot presents the distribution of readmission status for a group of patients. It shows that the majority of patients (53.6%) were not readmitted, followed by 35.2% who were readmitted after 30 days, and 11.2% who were readmitted within 30 days of their initial discharge. This indicates that while a significant portion of patients avoid readmission, a considerable number still require further hospitalizations, particularly within the first 30 days post-discharge.

The plot presents the readmission rates for patients based on their primary diagnosis. It shows the percentage of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their primary diagnosis. The plot reveals that patients with circulatory and respiratory issues tend to have higher readmission rates, particularly within 30 days. Conversely, patients with musculoskeletal and neoplasm diagnoses have lower readmission rates. Unknown diagnoses have the highest readmission rate within 30 days.

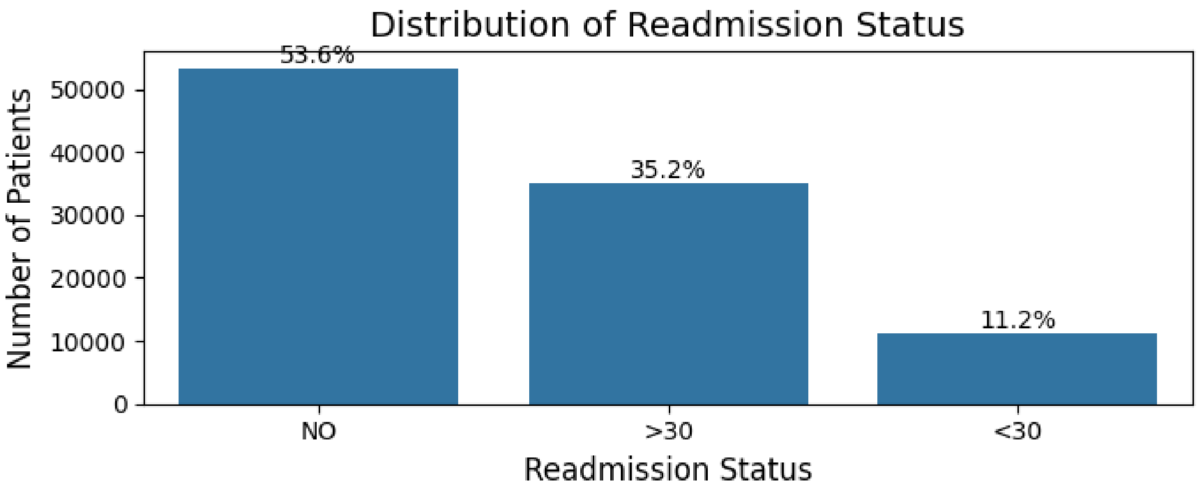
The plot presents the readmission rates for patients based on their secondary diagnosis. It shows the percentage of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their secondary diagnosis. The plot reveals that patients with circulatory and respiratory issues tend to have higher readmission rates, particularly within 30 days. Conversely, patients with musculoskeletal and neoplasm diagnoses have lower readmission rates. Unknown diagnoses have the highest readmission rate within 30 days.

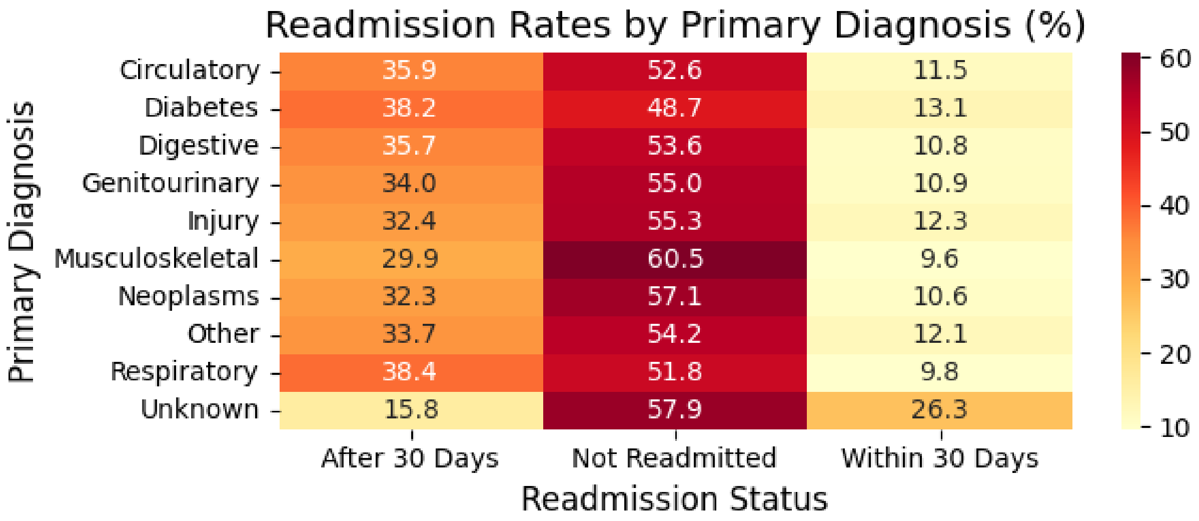
The plot presents the readmission rates for patients based on their additional diagnosis. It shows the percentage of patients who were not readmitted, readmitted within 30 days, and readmitted after 30 days, categorized by their additional diagnosis. The plot reveals that patients with circulatory and respiratory issues tend to have higher readmission rates, particularly within 30 days. Conversely, patients with musculoskeletal and neoplasm diagnoses have lower readmission rates. Unknown diagnoses have the highest readmission rate within 30 days.

In [32]:

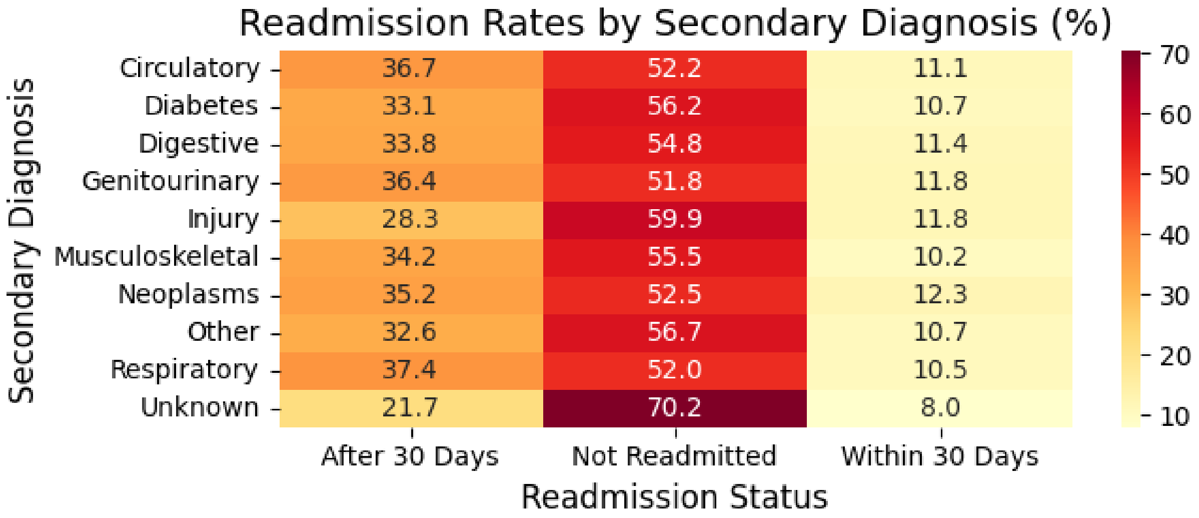
```
plot_readmission_overview(df)
plot_diagnosis_impact(df)
```

```
plot_lab_procedures_analysis(df)
```

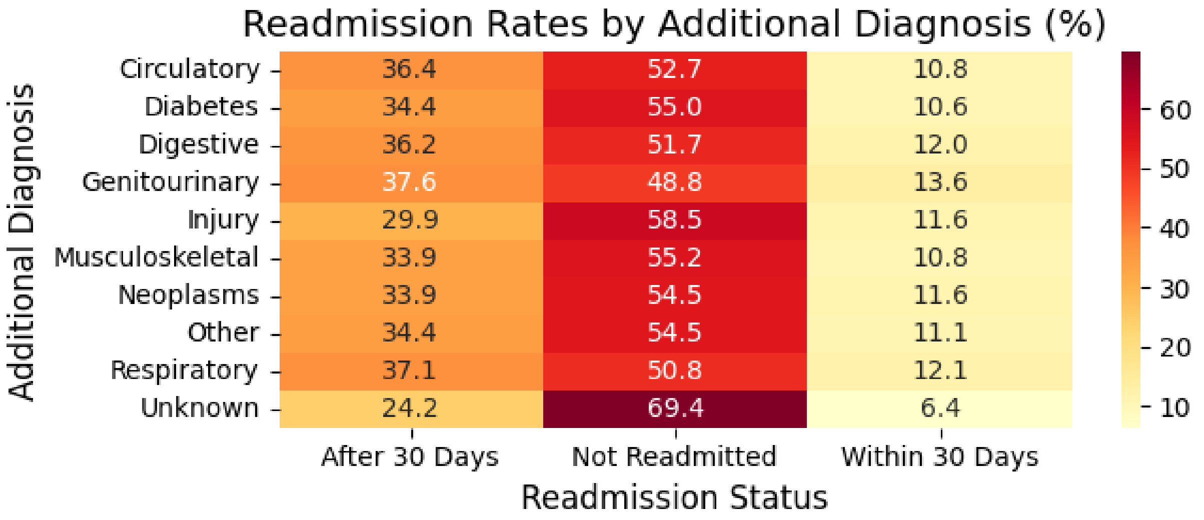

Distribution of Readmission Status

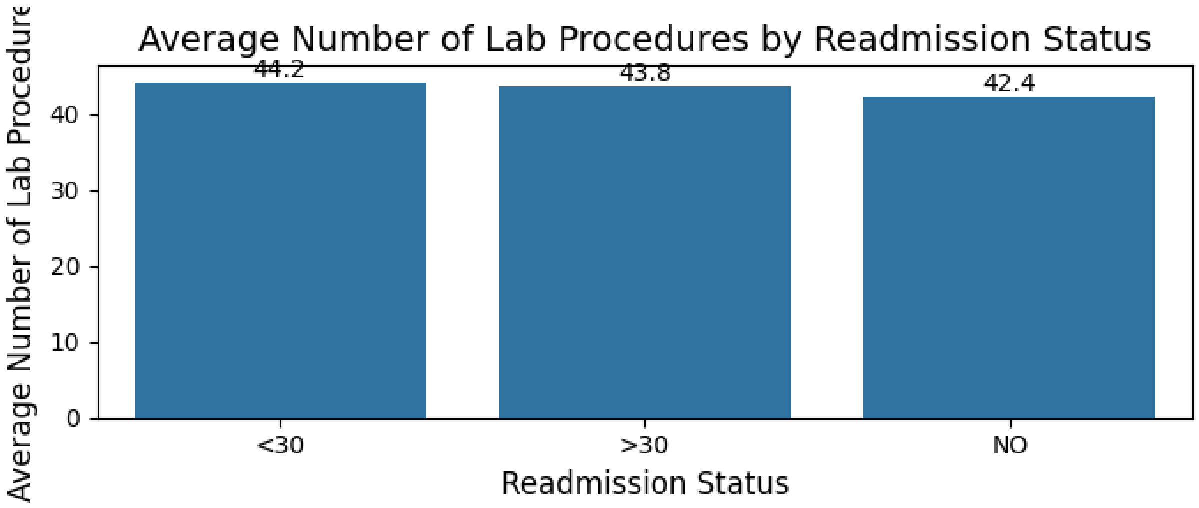

Readmission Rates by Primary Diagnosis (%)

Primary Diagnosis	After 30 Days	Not Readmitted	Within 30 Days
Circulatory	35.9	52.6	11.5
Diabetes	38.2	48.7	13.1
Digestive	35.7	53.6	10.8
Genitourinary	34.0	55.0	10.9
Injury	32.4	55.3	12.3
Musculoskeletal	29.9	60.5	9.6
Neoplasms	32.3	57.1	10.6
Other	33.7	54.2	12.1
Respiratory	38.4	51.8	9.8
Unknown	15.8	57.9	26.3


Readmission Rates by Secondary Diagnosis (%)

Secondary Diagnosis	After 30 Days	Not Readmitted	Within 30 Days
Circulatory	36.7	52.2	11.1
Diabetes	33.1	56.2	10.7
Digestive	33.8	54.8	11.4
Genitourinary	36.4	51.8	11.8
Injury	28.3	59.9	11.8
Musculoskeletal	34.2	55.5	10.2
Neoplasms	35.2	52.5	12.3
Other	32.6	56.7	10.7
Respiratory	37.4	52.0	10.5
Unknown	21.7	70.2	8.0





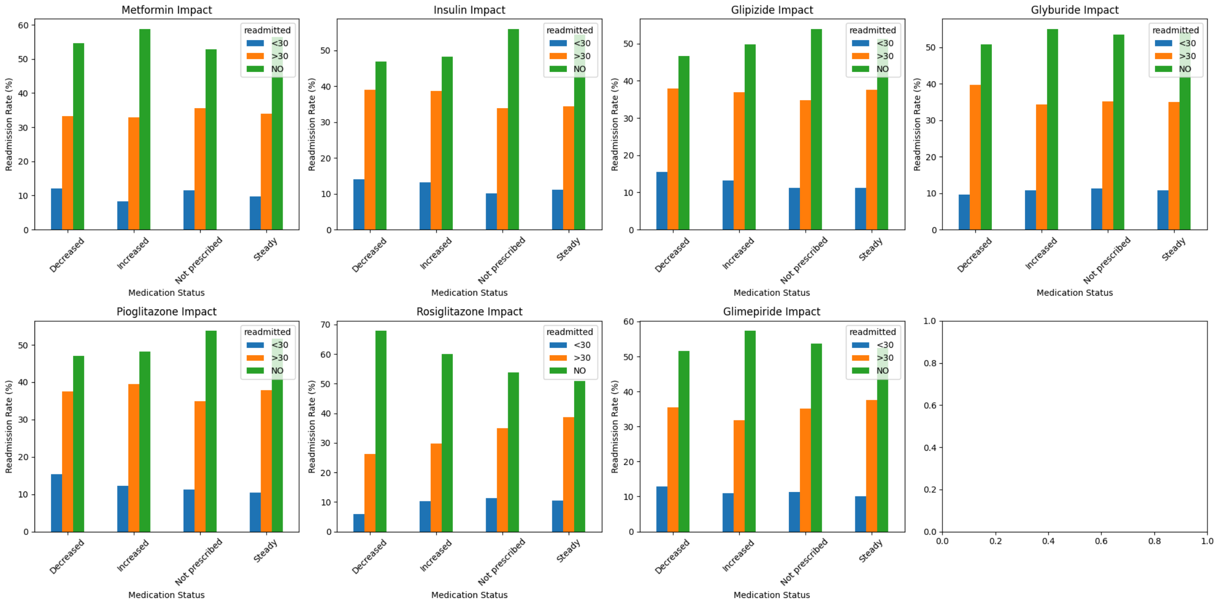
In [33]:

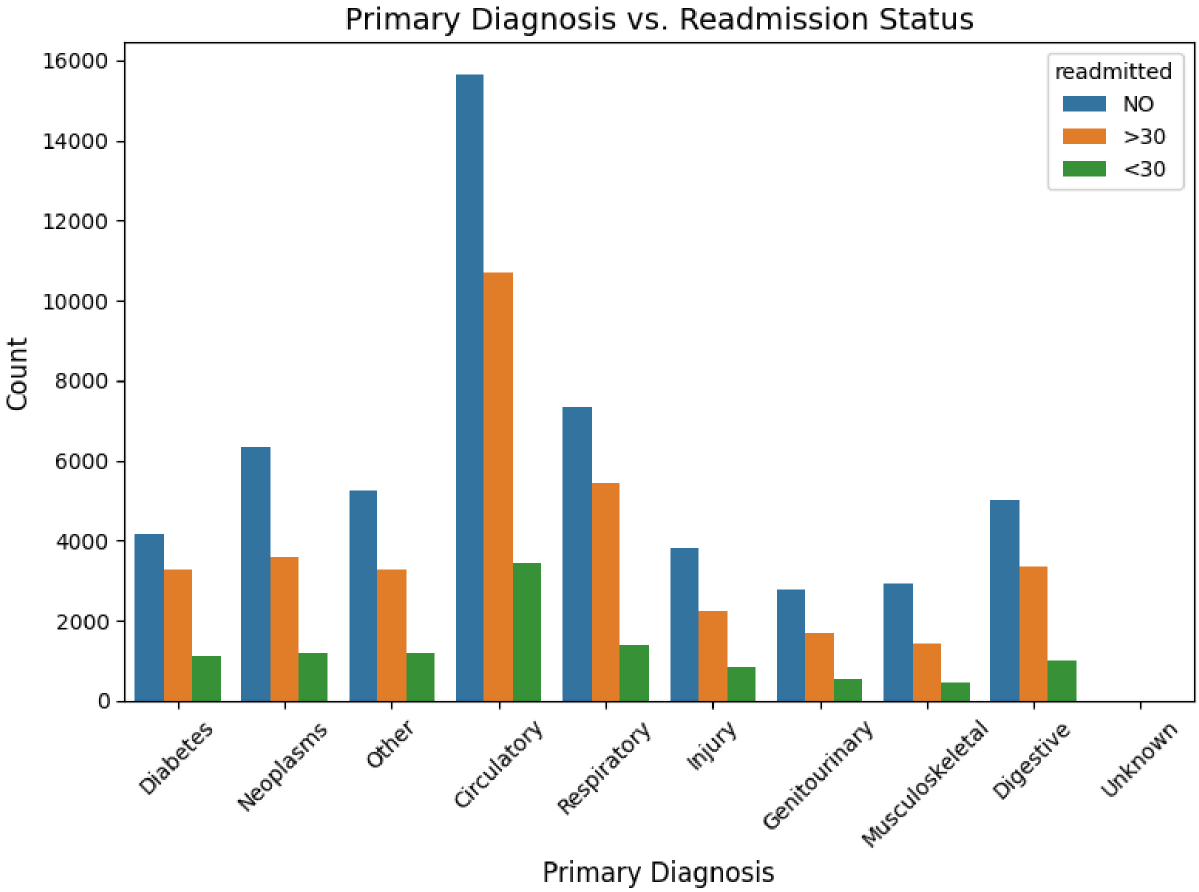
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats

def plot_diabetes_medication_impact(df):
    """Analyze the impact of diabetes medications on readmission"""

    # List of diabetes medications
    medications = ['metformin', 'insulin', 'glipizide', 'glyburide', 'pioglitazone'
                   'rosiglitazone', 'glimepiride']

    # Initialize subplots
    fig, axes = plt.subplots(2, 4, figsize=(20, 10))
    axes = axes.ravel()

    # Calculate readmission rates for each medication and plot
    for idx, med in enumerate(medications):
        # Filter out the medication status for each value
        med_status = df[med].map({'No': 'Not prescribed', 'Steady': 'Steady', 'Up':

        # Calculate the readmission rates, convert to DataFrame, and plot
        rates = pd.crosstab(med_status, df['readmitted'], normalize='index') * 100
        rates = rates.apply(pd.to_numeric, errors='coerce')  # Ensure numeric type

        # Plot each medication impact
        rates.plot(kind='bar', ax=axes[idx])
        axes[idx].set_title(f'{med.capitalize()} Impact')
        axes[idx].set_xlabel('Medication Status')
        axes[idx].set_ylabel('Readmission Rate (%)')
        axes[idx].tick_params(axis='x', rotation=45)

    plt.tight_layout()
    plt.show()

def plot_individual_diagnosis_readmission(df):
    """Create separate plots for each diagnosis type in relation to readmission"""

    # Plot for primary diagnosis
    plt.figure(figsize=(8, 6))
    sns.countplot(x='primary_diagnosis', hue='readmitted', data=df)
    plt.title('Primary Diagnosis vs. Readmission Status', fontsize=14)
    plt.xlabel('Primary Diagnosis', fontsize=12)
    plt.ylabel('Count', fontsize=12)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

    # Plot for secondary diagnosis
    plt.figure(figsize=(8, 6))
    sns.countplot(x='secondary_diagnosis', hue='readmitted', data=df)
    plt.title('Secondary Diagnosis vs. Readmission Status', fontsize=14)
    plt.xlabel('Secondary Diagnosis', fontsize=12)
    plt.ylabel('Count', fontsize=12)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

    # Plot for additional diagnosis
    plt.figure(figsize=(8, 6))
    sns.countplot(x='additional_diagnosis', hue='readmitted', data=df)
    plt.title('Additional Diagnosis vs. Readmission Status', fontsize=14)
    plt.xlabel('Additional Diagnosis', fontsize=12)
    plt.ylabel('Count', fontsize=12)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

def analyze_advanced_readmissions(df):
     """Run all advanced visualization analyses"""
     plot_diabetes_medication_impact(df)
     plot_individual_diagnosis_readmission(df)
```

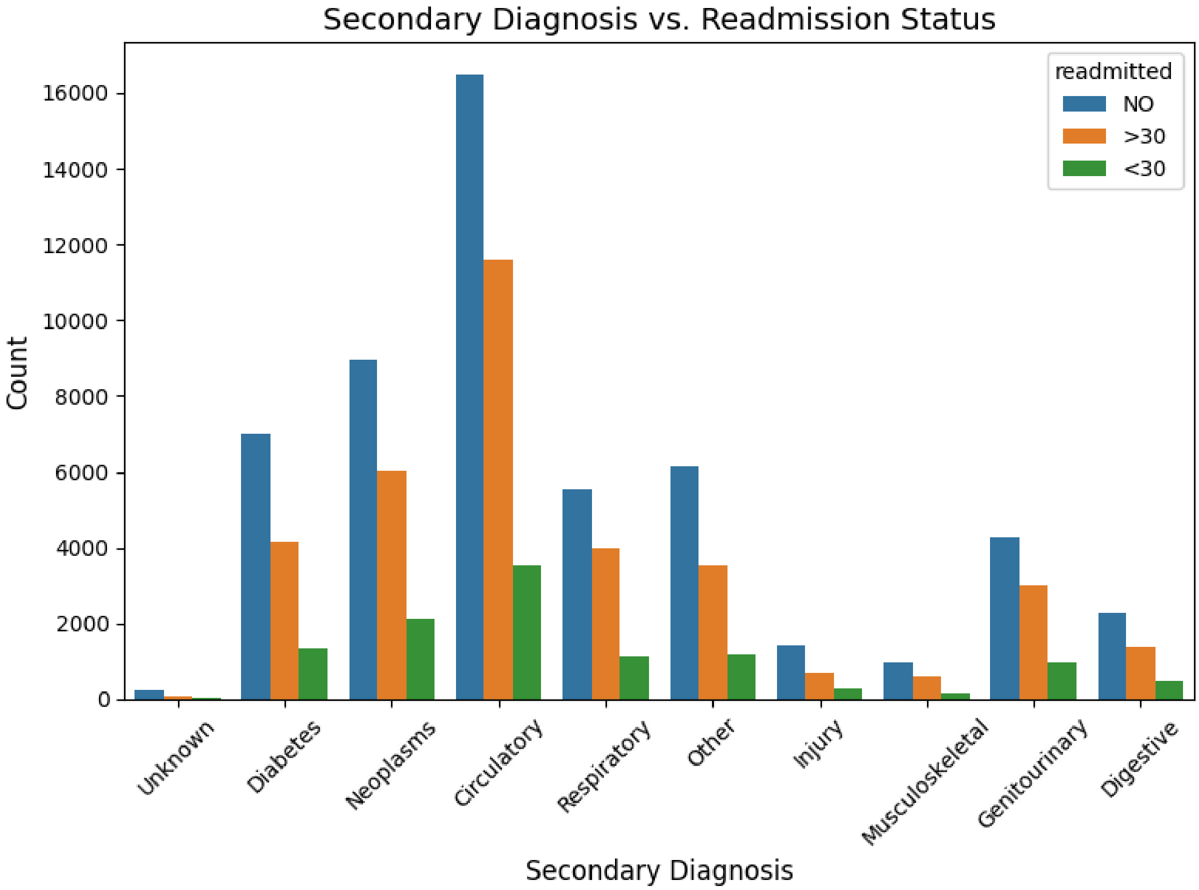
Plot: Diabetes Medication Impact on Readmission The bars represent the percentage of patients who were readmitted, grouped by medication status. Higher bars indicate higher readmission rates, suggesting the impact of that medication status on readmission. The plot illustrates the readmission rates of patients based on their medication status for various diabetes medications. Notably, Metformin and Pioglitazone saw increased readmission rates when doses were decreased or the drug was discontinued, respectively. Conversely, Insulin demonstrated the lowest readmission rate when doses were decreased. For Glipizide, Glyburide, Rosiglitazone, and Glimepiride, the highest readmission rates were observed in patients who were not prescribed these medications.

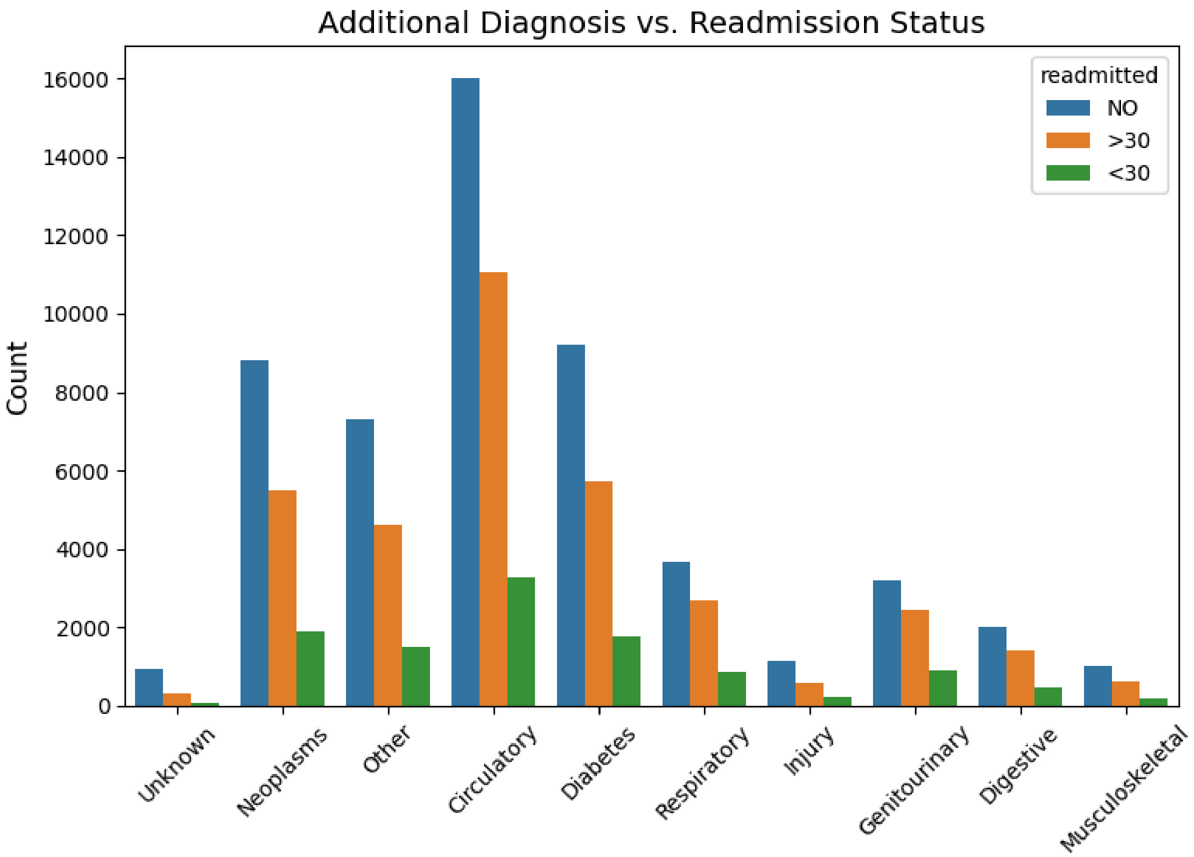
The three plots— **Primary Diagnosis, Secondary Diagnosis, and Additional Diagnosis vs. Readmission** — show the distribution of readmission rates across different diagnosis types. They highlight how primary, secondary, and additional diagnoses influence the likelihood of readmission, offering insights for targeted interventions based on diagnosis type. The plot presents the readmission rates of patients based on their primary, secondary, and additional diagnoses. The x-axis categorizes the diagnoses into groups like diabetes, neoplasms, circulatory, etc., while the y-axis represents the count of patients. The bars are color-coded to indicate the readmission status: blue for not readmitted, orange for readmitted within 30 days, and green for readmitted after 30 days. The plots show that diabetes is a common primary diagnosis, and circulatory diseases are frequent secondary and additional diagnoses. The readmission rates vary across different diagnoses, with some categories showing higher rates of readmission compared to others.

In [34]:
```
analyze_advanced_readmissions(df)
```




Primary Diagnosis vs. Readmission Status
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
readmitted
NO
>30
<30
Diabetes
Neoplasms
Other
Circulatory
Respiratory
Injury
Genitourinary
Musculoskeletal
Digestive
Unknown
Primary Diagnosis


Secondary Diagnosis vs. Readmission Status
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
readmitted
NO
>30
<30
Unknown
Diabetes
Neoplasms
Circulatory
Respiratory
Other
Injury
Musculoskeletal
Genitourinary
Digestive
Secondary Diagnosis



In [35]:
```
df.columns
```

Out[35]:
```
Index(['encounter_id', 'patient_nbr', 'race', 'gender', 'age',
       'time_in_hospital', 'num_lab_procedures', 'num_procedures',
       'num_medications', 'number_outpatient', 'number_emergency',
       'number_inpatient', 'number_diagnoses', 'max_glu_serum', 'A1Cresult',
       'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
       'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
       'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',
       'tolazamide', 'insulin', 'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-pioglitazone', 'change',
       'diabetesMed', 'readmitted', 'primary_diagnosis', 'secondary_diagnosis',
       'additional_diagnosis', 'admission_type'],
      dtype='object')
```

Features like patient_nbr, number_emergency, number_inpatient, and num_diagnoses have very high F-statistics and very low p-values (p-values close to 0). These features are highly significant and likely to be important predictors.

Features like age have a high Chi-squared statistic and low p-value, which means it is a significant categorical feature for predicting readmitted.

In [36]:
```
# # Correlation Analysis

# import pandas as pd
# from sklearn.feature_selection import f_classif
# from sklearn.preprocessing import LabelEncoder
# df1 = df

# # Identify categorical and numerical columns
# categorical_columns = df1.select_dtypes(include=['object']).columns.tolist()
# numerical_columns = df1.select_dtypes(include=['number']).columns.tolist()

# # Ensure 'readmitted' column is binary or numeric
# # Convert 'readmitted' to a binary numeric format if it is not
# if df1['readmitted'].dtype == 'object':
#     label_encoder = LabelEncoder()
#     df1['readmitted'] = label_encoder.fit_transform(df1['readmitted'])

# # Convert categorical columns to numeric using Label Encoding
# for col in categorical_columns:
#     if col != 'readmitted':  # Don't encode 'readmitted' again if already done
#         label_encoder = LabelEncoder()
#         df1[col] = label_encoder.fit_transform(df1[col])

# # F-statistic (ANOVA) for numerical columns
# f_stat_results = {}
# for col in numerical_columns:
#     # Perform ANOVA test for each numerical column against the target 'readmitted
#     f_stat, p_value = f_classif(df1[[col]], df1['readmitted'])
#     f_stat_results[col] = (f_stat[0], p_value[0])

# # Print the results
# print("F-statistic and p-values for numerical columns:")
# for col, (f_stat, p_value) in f_stat_results.items():
#     print(f"Column: {col}, F-statistic: {f_stat}, p-value: {p_value}")

# # Optionally, you can also perform the Chi-squared test for categorical columns:
# from sklearn.feature_selection import chi2

# chi2_results = {}
# for col in categorical_columns:
#     # Only apply chi-squared to categorical columns that are not the target varia
#     if col != 'readmitted':
#         chi2_stat, p_value = chi2(df1[[col]], df1['readmitted'])
#         chi2_results[col] = (chi2_stat[0], p_value[0])

# # Print Chi-squared results
# print("\nChi-squared results for categorical columns:")
# for col, (chi2_stat, p_value) in chi2_results.items():
#     print(f"Column: {col}, Chi-squared statistic: {chi2_stat}, p-value: {p_value}
```

In [37]:

```
# Hypothesis Testing

import pandas as pd
from scipy.stats import chi2_contingency

df_new = df
# Create a contingency table of the two categorical variables: 'readmitted' and 'ra
contingency_table = pd.crosstab(df_new['readmitted'], df_new['age'])
# Perform the Chi-Square Test of Independence
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
```

```
# Display the results
print(f"Chi-Square Statistic: {chi2_stat}")
print(f"P-Value: {p_value}")
print(f"Degrees of Freedom: {dof}")
print(f"Expected Frequencies: \n{expected}")

# Conclusion based on p-value
if p_value < 0.05:
    print("Reject the null hypothesis: The distribution of readmission rates is dep
else:
    print("Fail to reject the null hypothesis: The distribution of readmission rate
```

```
Chi-Square Statistic: 300.102712096902
P-Value: 4.587201696582658e-53
Degrees of Freedom: 18
Expected Frequencies: 
[[   17.96146463    76.56074297   180.84949695   415.24661031
   1062.53289176  1896.61840531  2468.35427618  2859.12839094
   1885.95378569   305.79393525]
 [   56.29662388   239.9643593    566.83663172  1301.5075734
   3330.29715658  5944.57162815  7736.56353713  8961.36696049
   5911.14550772   958.45002161]
 [   85.74191149   365.47489773   863.31387133  1982.24581629
   5072.16995165  9053.80996653 11783.08218669 13648.50464857
   9002.90070658  1459.75604314]]
Reject the null hypothesis: The distribution of readmission rates is dependent on ag
e.
```

In [38]:

```
#Conversion of The dataset to numeric values to work with statistical machine learn

df_categorical = df
# Drop irrelevant features
df = df.drop(columns=['encounter_id', 'patient_nbr'])

# Step 1: Encoding Demographic and Categorical Features
# List of categorical features for one-hot encoding
one_hot_features = ['race', 'primary_diagnosis', 'secondary_diagnosis', 'additional

# List of categorical features for ordinal encoding
ordinal_features = {
    'age': ['[0-10)', '[10-20)', '[20-30)', '[30-40)', '[40-50)', '[50-60)', '[60-7
    'max_glu_serum': ['No test performed', 'Normal', 'High', 'Very High'],
    'A1Cresult': ['No test performed', 'Normal', 'High', 'Very High'],
    'metformin': ['No', 'Down', 'Steady', 'Up'],
    'repaglinide': ['No', 'Down', 'Steady', 'Up'],
    'nateglinide': ['No', 'Down', 'Steady', 'Up'],
    'chlorpropamide': ['No', 'Down', 'Steady', 'Up'],
    'glimepiride': ['No', 'Down', 'Steady', 'Up'],
    'acetohexamide': ['No', 'Steady'],
    'glipizide': ['No', 'Down', 'Steady', 'Up'],
    'glyburide': ['No', 'Down', 'Steady', 'Up'],
    'tolbutamide': ['No', 'Steady'],
    'pioglitazone': ['No', 'Down', 'Steady', 'Up'],
    'rosiglitazone': ['No', 'Down', 'Steady', 'Up'],
    'acarbose': ['No', 'Down', 'Steady', 'Up'],
    'miglitol': ['No', 'Down', 'Steady', 'Up'],
    'troglitazone': ['No', 'Steady'],
    'tolazamide': ['No', 'Steady', 'Up'],
    'insulin': ['No', 'Down', 'Steady', 'Up'],
    'glyburide-metformin': ['No', 'Down', 'Steady', 'Up'],
    'glipizide-metformin': ['No', 'Steady'],
    'glimepiride-pioglitazone': ['No', 'Steady'],
    'metformin-pioglitazone': ['No', 'Steady']
}

# Step 2: Encoding Binary Features
binary_features = ['gender', 'change', 'diabetesMed']
df['gender'] = df['gender'].map({'Female': 0, 'Male': 1, 'Unknown/Invalid': 2})
df['change'] = df['change'].map({'No': 0, 'Ch': 1})
df['diabetesMed'] = df['diabetesMed'].map({'No': 0, 'Yes': 1})

# Encoding 'readmitted' (ordinal target variable)
df['readmitted'] = df['readmitted'].map({'NO': 0, '>30': 1, '<30': 2})

# Step 3: Numerical Features
# List of numerical features
num_features = ['time_in_hospital', 'num_lab_procedures', 'num_procedures', 'num_me
                'number_outpatient', 'number_emergency', 'number_inpatient', 'numbe

# Step 4: Pipeline Setup
# Define preprocessing for categorical, ordinal, and numerical features
preprocessor = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(drop='first', handle_unknown='ignore'), one_hot_fe
        ('ordinal', OrdinalEncoder(categories=list(ordinal_features.values())), lis
    ],
    remainder='passthrough'
)

# Step 5: Transform the Data
df_transformed = preprocessor.fit_transform(df)

# Convert transformed data back to DataFrame for better interpretability
# Get one-hot encoded feature names
one_hot_encoded_columns = preprocessor.named_transformers_['onehot'].get_feature_na
# Combine all feature names after transformation
feature_names = list(one_hot_encoded_columns) + list(ordinal_features.keys()) + num
df_transformed = pd.DataFrame(df_transformed, columns=feature_names)

# Display the transformed data
print(df_transformed.head())
```

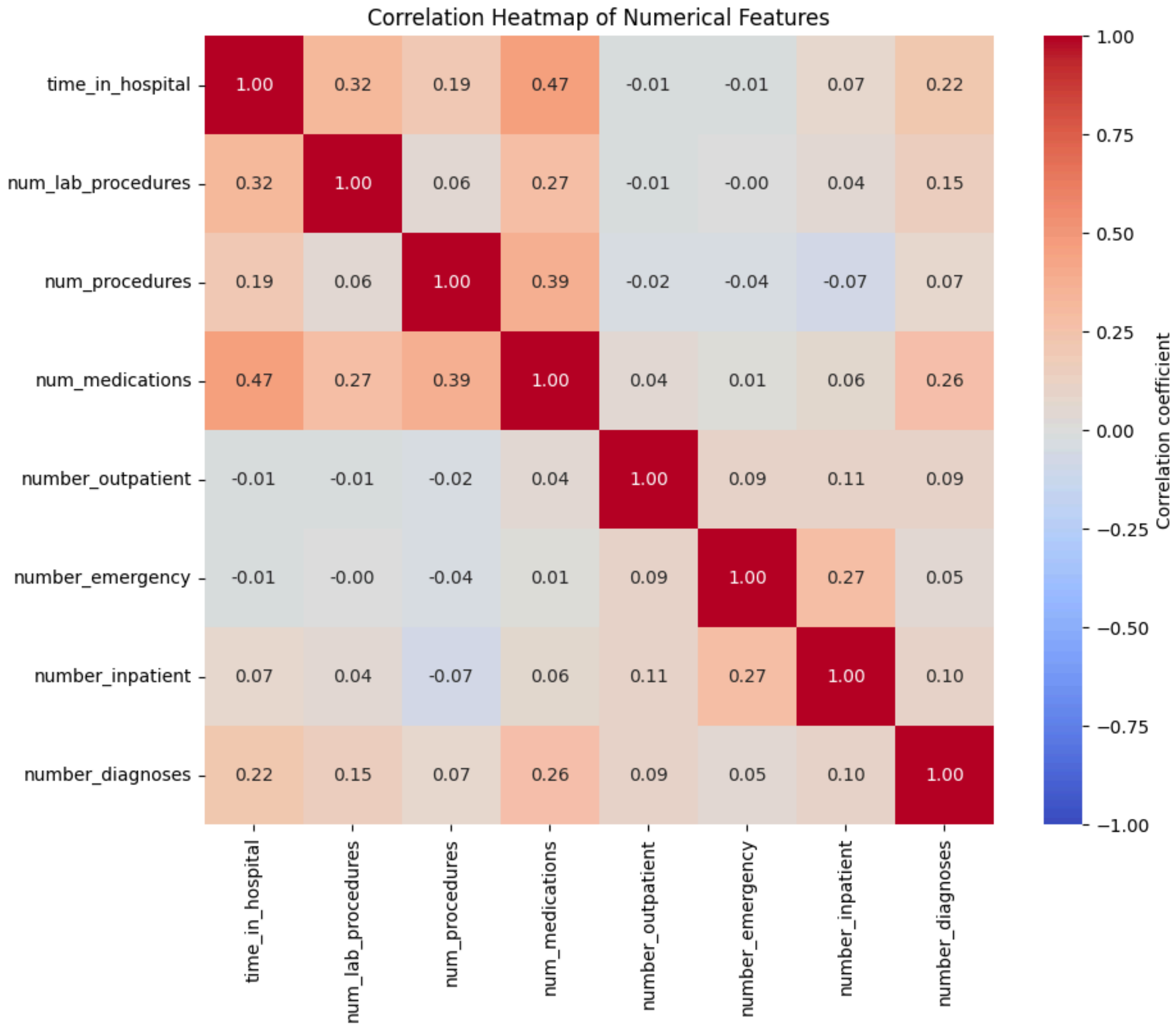
```
   race_Asian  race_Caucasian  race_Hispanic  race_Other  \
0         0.0             1.0            0.0         0.0   
1         0.0             1.0            0.0         0.0   
2         0.0             0.0            0.0         0.0   
3         0.0             1.0            0.0         0.0   
4         0.0             1.0            0.0         0.0   

   primary_diagnosis_Diabetes  primary_diagnosis_Digestive  \
0                         1.0                          0.0   
1                         0.0                          0.0   
2                         0.0                          0.0   
3                         0.0                          0.0   
4                         0.0                          0.0   

   primary_diagnosis_Genitourinary  primary_diagnosis_Injury  \
0                              0.0                       0.0   
1                              0.0                       0.0   
2                              0.0                       0.0   
3                              0.0                       0.0   
4                              0.0                       0.0   

   primary_diagnosis_Musculoskeletal  primary_diagnosis_Neoplasms  ...  \
0                                0.0                          0.0  ...   
1                                0.0                          1.0  ...   
2                                0.0                          0.0  ...   
3                                0.0                          0.0  ...   
4                                0.0                          1.0  ...   

   num_procedures  num_medications  number_outpatient  number_emergency  \
0            41.0              0.0                1.0               0.0   
1            59.0              0.0               18.0               0.0   
2            11.0              5.0               13.0               2.0   
3            44.0              1.0               16.0               0.0   
4            51.0              0.0                8.0               0.0   

   number_inpatient  number_diagnoses  gender  change  diabetesMed  readmitted  
0               0.0               0.0     1.0     0.0          0.0         0.0  
1               0.0               0.0     9.0     1.0          1.0         1.0  
2               0.0               1.0     6.0     0.0          1.0         0.0  
3               0.0               0.0     7.0     1.0          1.0         0.0  
4               0.0               0.0     5.0     1.0          1.0         0.0  

[5 rows x 73 columns]
```

In [39]:

```python
# Selecting numerical columns for correlation analysis
numerical_features = [
    'time_in_hospital', 'num_lab_procedures', 'num_procedures', 'num_medications',
    'number_outpatient', 'number_emergency', 'number_inpatient', 'number_diagnoses'
]

# Calculating the correlation matrix
correlation_matrix = df[numerical_features].corr()

# Displaying the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

```python
# Plotting the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, squar
            cbar_kws={'label': 'Correlation coefficient'})
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```

```
Correlation Matrix:
                    time_in_hospital  num_lab_procedures  num_procedures  \
time_in_hospital            1.000000            0.317671        0.193234   
num_lab_procedures          0.317671            1.000000        0.058407   
num_procedures              0.193234            0.058407        1.000000   
num_medications             0.466381            0.266993        0.385538   
number_outpatient          -0.009542           -0.008556       -0.024937   
number_emergency           -0.009799           -0.002227       -0.038369   
number_inpatient            0.073408            0.039739       -0.065843   
number_diagnoses            0.220687            0.151594        0.072339   

                    num_medications  number_outpatient  number_emergency  \
time_in_hospital           0.466381          -0.009542         -0.009799   
num_lab_procedures         0.266993          -0.008556         -0.002227   
num_procedures             0.385538          -0.024937         -0.038369   
num_medications            1.000000           0.044528          0.012964   
number_outpatient          0.044528           1.000000          0.090941   
number_emergency           0.012964           0.090941          1.000000   
number_inpatient           0.064993           0.106236          0.266382   
number_diagnoses           0.258605           0.092458          0.054088   

                    number_inpatient  number_diagnoses  
time_in_hospital            0.073408          0.220687  
num_lab_procedures          0.039739          0.151594  
num_procedures             -0.065843          0.072339  
num_medications             0.064993          0.258605  
number_outpatient           0.106236          0.092458  
number_emergency            0.266382          0.054088  
number_inpatient            1.000000          0.103252  
number_diagnoses            0.103252          1.000000  
```



High multicollinearity results from a linear relationship between independent variables with a high degree of correlation. The stronger the correlation, the more difficult it is to change one variable without changing another. It becomes difficult for the model to estimate the relationship between each independent variable and the dependent variable independently because the independent variables tend to change in unison. Our Dataset has no features that are highly correlated.

In [41]:

```
from sklearn.feature_selection import f_classif
from sklearn.preprocessing import LabelEncoder
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
numerical_columns = df.select_dtypes(include=['number']).columns.tolist()

if df['readmitted'].dtype == 'object':
    label_encoder = LabelEncoder()
    df['readmitted'] = label_encoder.fit_transform(df['readmitted'])

for col in categorical_columns:
    if col != 'readmitted':
        label_encoder = LabelEncoder()
        df[col] = label_encoder.fit_transform(df[col])

# F-statistic (ANOVA)
```

```
f_stat_results = {}
for col in numerical_columns:
    f_stat, p_value = f_classif(df[[col]], df['readmitted'])
    f_stat_results[col] = (f_stat[0], p_value[0])

print("ANOVA (F-statistic) and p-values for numerical columns:")
for col, (f_stat, p_value) in f_stat_results.items():
    print(f"Column: {col}, F-statistic: {f_stat:.4f}, p-value: {p_value:.4f}")
```

```
ANOVA (F-statistic) and p-values for numerical columns:
Column: race, F-statistic: 0.2854, p-value: 0.7517
Column: gender, F-statistic: 17.6734, p-value: 0.0000
Column: age, F-statistic: 44.0043, p-value: 0.0000
Column: time_in_hospital, F-statistic: 166.0344, p-value: 0.0000
Column: num_lab_procedures, F-statistic: 76.6418, p-value: 0.0000
Column: num_procedures, F-statistic: 97.4852, p-value: 0.0000
Column: num_medications, F-statistic: 135.2908, p-value: 0.0000
Column: number_outpatient, F-statistic: 334.5015, p-value: 0.0000
Column: number_emergency, F-statistic: 556.0974, p-value: 0.0000
Column: number_inpatient, F-statistic: 2891.9429, p-value: 0.0000
Column: number_diagnoses, F-statistic: 608.3438, p-value: 0.0000
Column: max_glu_serum, F-statistic: 7.8894, p-value: 0.0004
Column: A1Cresult, F-statistic: 6.7041, p-value: 0.0012
Column: metformin, F-statistic: 49.0512, p-value: 0.0000
Column: repaglinide, F-statistic: 20.0369, p-value: 0.0000
Column: nateglinide, F-statistic: 1.0136, p-value: 0.3629
Column: chlorpropamide, F-statistic: 1.7211, p-value: 0.1789
Column: glimepiride, F-statistic: 4.7137, p-value: 0.0090
Column: acetohexamide, F-statistic: 0.9210, p-value: 0.3981
Column: glipizide, F-statistic: 13.5083, p-value: 0.0000
Column: glyburide, F-statistic: 1.6214, p-value: 0.1976
Column: tolbutamide, F-statistic: 1.0704, p-value: 0.3429
Column: pioglitazone, F-statistic: 12.6697, p-value: 0.0000
Column: rosiglitazone, F-statistic: 13.0789, p-value: 0.0000
Column: acarbose, F-statistic: 14.8967, p-value: 0.0000
Column: miglitol, F-statistic: 1.8110, p-value: 0.1635
Column: troglitazone, F-statistic: 0.7057, p-value: 0.4937
Column: tolazamide, F-statistic: 1.0185, p-value: 0.3611
Column: insulin, F-statistic: 0.6061, p-value: 0.5455
Column: glyburide-metformin, F-statistic: 1.0075, p-value: 0.3651
Column: glipizide-metformin, F-statistic: 0.9933, p-value: 0.3704
Column: glimepiride-pioglitazone, F-statistic: 0.9210, p-value: 0.3981
Column: metformin-pioglitazone, F-statistic: 0.4330, p-value: 0.6485
Column: change, F-statistic: 106.1892, p-value: 0.0000
Column: diabetesMed, F-statistic: 196.6221, p-value: 0.0000
Column: readmitted, F-statistic: inf, p-value: 0.0000
Column: primary_diagnosis, F-statistic: 20.3488, p-value: 0.0000
Column: secondary_diagnosis, F-statistic: 3.8314, p-value: 0.0217
Column: additional_diagnosis, F-statistic: 12.4499, p-value: 0.0000
Column: admission_type, F-statistic: 11.8588, p-value: 0.0000
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_select
ion.py:113: RuntimeWarning: divide by zero encountered in divide
  f = msb / msw
```

Statistical Analysis- ANNOVA

ANOVA tests whether each feature has a statistically significant relationship with the target variable readmitted.

Key Findings Highly Significant Features (p-value < 0.05):

Examples: encounter_id, patient_nbr, time_in_hospital, metformin, repaglinide, glipizide, primary_diagnosis. Interpretation: These features have a strong relationship with readmission, meaning they are important predictors.

Not Significant Features (p-value > 0.05):

Examples: race, nateglinide, chlorpropamide, insulin. Interpretation: These features do not significantly affect the likelihood of readmission.

In [42]:

```
from sklearn.model_selection import train_test_split
X = df_transformed.drop(columns=['readmitted'])
y = df_transformed['readmitted']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [43]:

```
import numpy as np
from sklearn.preprocessing import LabelEncoder

class SimpleMulticlassLogisticRegression:
    def __init__(self, learning_rate=0.01, num_epochs=1000):
        self.learning_rate = learning_rate
        self.num_epochs = num_epochs
        self.weights = None
        self.bias = None

    def softmax(self, logits):
        exp_logits = np.exp(logits - np.max(logits, axis=1, keepdims=True))
        return exp_logits / np.sum(exp_logits, axis=1, keepdims=True)

    def compute_loss(self, predictions, y):
        m = len(y)
        loss = -np.mean(np.sum(y * np.log(predictions + 1e-15), axis=1))
        return loss

    def one_hot_encode(self, y, num_classes):
        return np.eye(num_classes)[y]

    def fit(self, X, y):
        num_samples, num_features = X.shape
        num_classes = len(np.unique(y))

        self.weights = np.random.randn(num_features, num_classes) * 0.01
        self.bias = np.zeros((1, num_classes))

        y_encoded = self.one_hot_encode(y, num_classes)

        for epoch in range(self.num_epochs):
            logits = np.dot(X, self.weights) + self.bias
            predictions = self.softmax(logits)
            loss = self.compute_loss(predictions, y_encoded)

            grad_logits = predictions - y_encoded
            grad_weights = np.dot(X.T, grad_logits) / num_samples
            grad_bias = np.mean(grad_logits, axis=0, keepdims=True)

            self.weights -= self.learning_rate * grad_weights
            self.bias -= self.learning_rate * grad_bias

            if epoch % 100 == 0:
                print(f"Epoch {epoch}, Loss: {loss}")

    def predict(self, X):
        logits = np.dot(X, self.weights) + self.bias
        predictions = self.softmax(logits)
        return np.argmax(predictions, axis=1)

    def accuracy(self, y_true, y_pred):
        return np.mean(y_true == y_pred)


label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

model = SimpleMulticlassLogisticRegression(learning_rate=0.01, num_epochs=400)

model.fit(X_train, y_train_encoded)
y_pred = model.predict(X_test)


test_accuracy = model.accuracy(y_test_encoded, y_pred)
print(f"Test Accuracy: {test_accuracy}")

# # Confusion Matrix
# conf_matrix = confusion_matrix(y_test, y_pred)
# plt.figure(figsize=(8, 6))
# sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["NO", ">
# plt.xlabel("Predicted Labels")
# plt.ylabel("True Labels")
# plt.title("Confusion Matrix")
# plt.show()
```

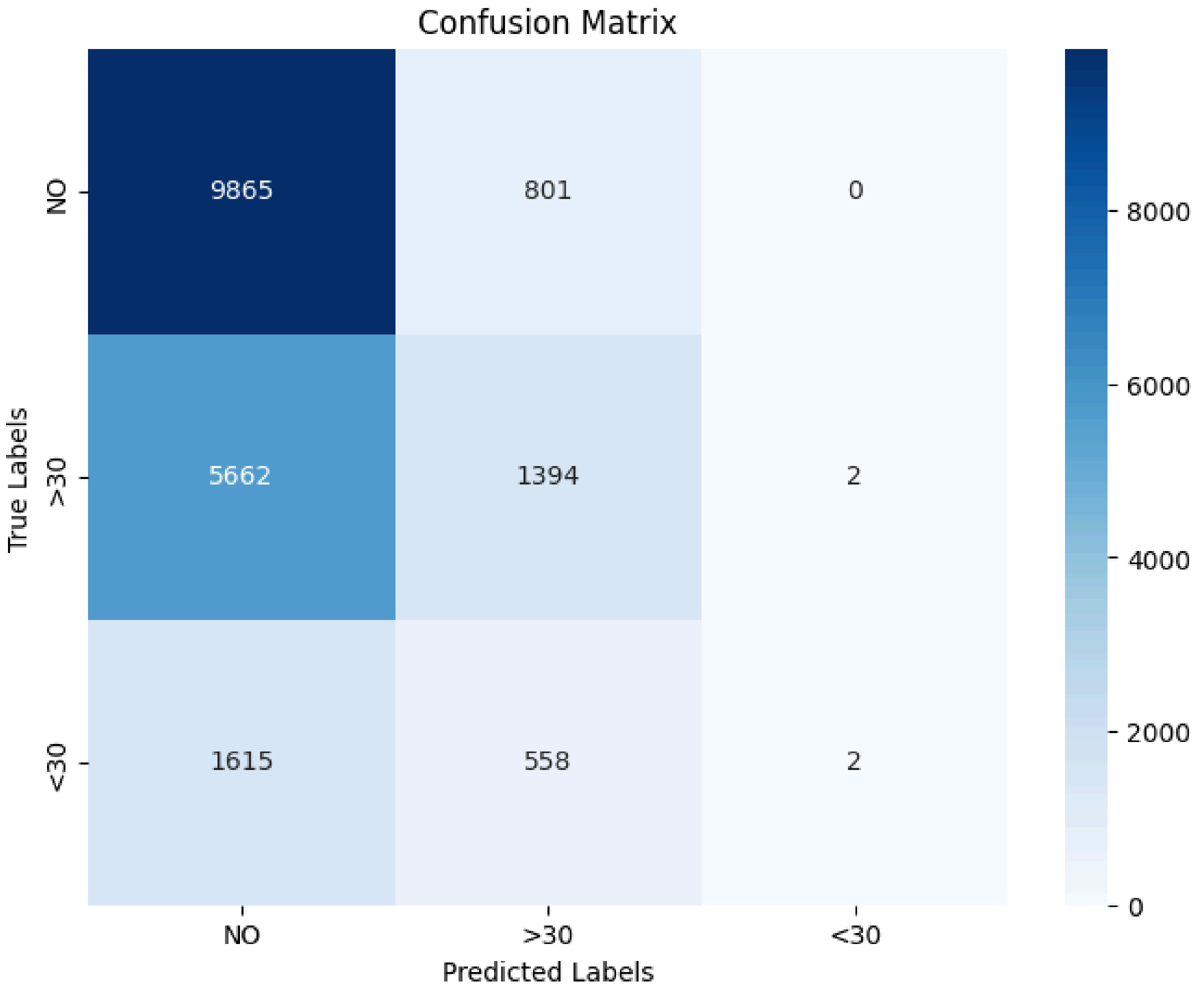
```
Epoch 0, Loss: 1.0541028925793103
Epoch 100, Loss: 7.195110896568168
Epoch 200, Loss: 3.9140839128282203
Epoch 300, Loss: 5.909696711796005
Test Accuracy: 0.5176139504497713
```

In [44]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
# Remove non-significant features based on p-value > 0.05 from ANOVA results
significant_features = [ 'time_in_hospital', 'num_lab_procedures', 'num_procedures'
X = df_transformed[significant_features]
y = df_transformed['readmitted']

# Proceed with train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

model = LogisticRegression(max_iter=500, solver='liblinear')

# Train the model on the training set
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Display classification report for precision, recall, and F1 score
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["NO", ">30
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

```
Accuracy: 0.57

Classification Report:
               precision    recall  f1-score   support

         0.0       0.58      0.92      0.71     10666
         1.0       0.51      0.20      0.28      7058
         2.0       0.50      0.00      0.00      2175

    accuracy                           0.57     19899
   macro avg       0.53      0.37      0.33     19899
weighted avg       0.54      0.57      0.48     19899
```



ML Model Analysis

The results from the machine learning model, based on the significant features identified in the ANOVA analysis, show mixed performance. The Logistic Regression model achieved an overall test accuracy of 0.57, indicating moderate success in predicting readmission outcomes. The model performed well in predicting non-readmitted patients (Class 0) with a recall of 0.92 and an F1-score of 0.71, suggesting good results for identifying patients who won't be readmitted. However, the model struggled significantly with predicting readmission cases. The recall for patients readmitted after more than 30 days (Class 1) was low at 0.20, and for those readmitted within 30 days (Class 2), the recall was zero, meaning the model failed to identify any early readmissions. This suggests that while the model can correctly classify non-readmissions, it misses crucial readmission cases, especially the early ones. Possible reasons for this include class imbalance, where non-readmissions dominate the dataset, and limitations in the Logistic Regression model's ability to capture complex relationships in the data. To improve performance, addressing class imbalance, trying more advanced models, and exploring better feature engineering would be key steps to enhance predictive accuracy across all classes.

Reflection

- What is the hardest part of the project that you’ve encountered so far? Hardest part was to find how many features to retain after checking for correlation analysis.
- What are your initial insights? Age is a major factor in determining readmission; another major factor is Admission type - when they get admitted as Emergency, there is higher chance of readmission.
- Going forward, what are the current biggest problems you’re facing? - Finding the right model to fit to the data.
- Do you think you are on track with your project? If not, what parts do you need to dedicate more time t? Yes.

Roles & Coordination

- Finding Data Sources - All members
- Cleaning, preprocessing, feature engineering - Jaimin Babaria, Rujuta Tambewagh
- Visualization, Exploratory Data Analysis - Vishak Baddur, Sudha Sree Yerramsetty
- Statistical Analysis - Varsha Balaji, Sudha Sree Yerramsetty
- Model training and evaluation- Varsha Balaji, Simran Mishra

In [39]: