1.Exploratory Data Analysis (EDA) (20 pts):

6.Additional Insights (10 pts):

Loading the data set

In [2]:
```
import pandas as pd
import py7zr
from io import BytesIO
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#from sklearn.model_selection import cross_val_score
#from sklearn.model_selection import train_test_split

#from sklearn.metrics import accuracy_score, mean_squared_error, r2_score

#from sklearn.linear_model import LassoCV
#from sklearn.linear_model import LogisticRegressionCV
#import statsmodels.api as sm

#from sklearn.preprocessing import StandardScaler

# Path to your .7z file
file_path = 'C:\\Users\\meghs\\Downloads\\longyear-obese-hypertensive-40-57-medi

# Open the .7z file
with py7zr.SevenZipFile(file_path, mode='r') as z:
    all_files = z.getnames()
    # Assuming there's only one CSV file in the .7z archive
    csv_file_name = all_files[0]
    # Extract the CSV file to memory
    csv_file_dict = z.read([csv_file_name])
    csv_file = csv_file_dict[csv_file_name]

# Load the CSV data into a pandas DataFrame
df = pd.read_csv(csv_file)

# Now you can work with the DataFrame `df` as usual
file_path = 'C:\\Users\\meghs\\Downloads\\longyear-obese-hypertensive-40-75-scri

# Open the .7z file
with py7zr.SevenZipFile(file_path, mode='r') as z:
    all_files = z.getnames()
    # Assuming there's only one CSV file in the .7z archive
    csv_file_name = all_files[0]
    # Extract the CSV file to memory
    csv_file_dict = z.read([csv_file_name])
    csv_file = csv_file_dict[csv_file_name]
```

```
# Load the CSV data into a pandas DataFrame
df2 = pd.read_csv(csv_file)
```

```
C:\Users\meghs\AppData\Local\Temp\ipykernel_11944\3725068458.py:47: DtypeWarning:
Columns (9,10,29,43) have mixed types. Specify dtype option on import or set low_
memory=False.
  df2 = pd.read_csv(csv_file)
```

Information about df

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 623525 entries, 0 to 623524
Data columns (total 34 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   journey_id         623525 non-null  object
 1   episode_id         623525 non-null  object
 2   visit_id           621650 non-null  object
 3   encounter_id       623525 non-null  object
 4   claim_date         623525 non-null  object
 5   patient_state      623525 non-null  object
 6   patient_short_zip  623512 non-null  float64
 7   patient_age        616980 non-null  float64
 8   patient_gender     623393 non-null  object
 9   place_of_service   484555 non-null  object
 10  visit_type         600454 non-null  object
 11  payor              608936 non-null  object
 12  payor_channel      586086 non-null  object
 13  ref_npi            297501 non-null  float64
 14  hcp_npi            434653 non-null  float64
 15  hcp_taxonomy       428209 non-null  object
 16  hcp_specialty      428229 non-null  object
 17  hco_npi            617789 non-null  float64
 18  hcp_npi_list       434653 non-null  object
 19  hco_npi_list       617789 non-null  object
 20  diag_list          622655 non-null  object
 21  diag_1             622655 non-null  object
 22  diag_2             460595 non-null  object
 23  diag_3             365284 non-null  object
 24  diag_4             298233 non-null  object
 25  diag_5             205016 non-null  object
 26  rev_center_code    113926 non-null  float64
 27  rev_center_units   621558 non-null  float64
 28  proc_code          599612 non-null  object
 29  proc_modifier      165948 non-null  object
 30  proc_units         621331 non-null  float64
 31  line_charge        623525 non-null  float64
 32  claim_charge       623525 non-null  float64
 33  smart_allowed      623525 non-null  float64
dtypes: float64(11), object(23)
memory usage: 161.7+ MB
```

Information about df2

In [4]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 894623 entries, 0 to 894622
Data columns (total 48 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   journey_id                      894623 non-null  object 
 1   cohort_id                       894623 non-null  int64  
 2   claim_id                        894623 non-null  object 
 3   patient_gender                  894623 non-null  object 
 4   patient_state                   894623 non-null  object 
 5   patient_zip                     894501 non-null  float64
 6   patient_dob                     894619 non-null  object 
 7   pharmacy_npi                    839958 non-null  float64
 8   pharmacist_npi                  21578 non-null   float64
 9   prescriber_npi                  891155 non-null  object 
 10  primary_care_npi                3569 non-null    object 
 11  group_id                        273621 non-null  object 
 12  date_of_service                 894623 non-null  object 
 13  date_authorized                 784676 non-null  object 
 14  transaction_type                894623 non-null  object 
 15  date_prescription_written       894623 non-null  object 
 16  ndc11                           894623 non-null  int64  
 17  ndc11_original                  94203 non-null   float64
 18  number_of_refills_authorized    849519 non-null  float64
 19  diagnosis_code                  174705 non-null  object 
 20  diagnosis_code_type             111634 non-null  float64
 21  quantity_prescribed_original    322348 non-null  float64
 22  quantity_dispensed              894616 non-null  float64
 23  place_of_service                346197 non-null  float64
 24  is_service                      678323 non-null  object 
 25  level_of_service                217951 non-null  float64
 26  fill_number                     894623 non-null  float64
 27  days_supply                     894623 non-null  float64
 28  unit_of_measure                 594590 non-null  object 
 29  daw_code                        192144 non-null  object 
 30  prior_authorization_type_code   8916 non-null    float64
 31  is_compound_drug                784594 non-null  object 
 32  coupon_type                     2030 non-null    float64
 33  coupon_value_amount             5 non-null       float64
 34  pharmacy_submitted_cost         785081 non-null  float64
 35  patient_pay                     601820 non-null  float64
 36  copay_coinsurance               411383 non-null  float64
 37  pcn                             775703 non-null  object 
 38  bin                             786240 non-null  float64
 39  plan_pay                        491878 non-null  float64
 40  reject_code_1                   187279 non-null  object 
 41  reject_code_3                   41252 non-null   object 
 42  reject_code_4                   7408 non-null    object 
 43  reject_code_5                   1524 non-null    object 
 44  ndc                             894623 non-null  int64  
 45  active                          894623 non-null  bool   
 46  start_date                      894623 non-null  int64  
 47  end_date                        32448 non-null   float64
dtypes: bool(1), float64(21), int64(4), object(22)
memory usage: 321.6+ MB
```

EDA using diag(diagnosis colums) for data analysis

In [5]:
```
# 1. Select 'diag_' columns
diag_columns = [col for col in df.columns if col.startswith('diag_')]
```

```
df_diag = df[diag_columns]

# 2. Combine values
combined_list = df_diag.values.flatten().tolist()
combined_list = [x for x in combined_list if pd.notna(x)]

# 3. Process first three digits (same as before)
first_three_counts = {}
for diag_code in combined_list:
    first_three = diag_code[:3]
    first_three_counts[first_three] = first_three_counts.get(first_three, 0) + 1
```

In [6]:

```
import operator
# Sort the dictionary items by count (descending order)
sorted_counts = sorted(first_three_counts.items(), key=operator.itemgetter(1), r

for items in sorted_counts:
    print(items)
```

```
('E11', 233629)
('I10', 199537)
('E78', 113939)
('Z00', 95083)
('E66', 69689)
('Z68', 54013)
('M25', 51750)
('M54', 50447)
('Z12', 32396)
('N18', 31950)
('Z01', 30622)
('Z13', 29844)
('M79', 29497)
('G47', 28339)
('R10', 27659)
('Z79', 27149)
('E55', 27085)
('Z71', 26523)
('R07', 25175)
('E03', 24910)
('R73', 24776)
('N39', 22340)
('M17', 21592)
('D64', 19973)
('I25', 17483)
('Z20', 16871)
('R53', 16402)
('I50', 15359)
('R06', 15340)
('J44', 14592)
('F41', 13833)
('Z23', 13598)
('I48', 12605)
('K21', 12366)
('G89', 11753)
('Z11', 11720)
('I11', 11344)
('F33', 10927)
('N40', 10720)
('J45', 10699)
('E87', 10668)
('M19', 10218)
('U07', 10043)
('M47', 9806)
('R68', 9479)
('A41', 9416)
('D50', 9387)
('J96', 9330)
('R05', 9106)
('M51', 8942)
('F32', 8895)
('I12', 8183)
('L03', 7899)
('B35', 7878)
('R79', 7617)
('Z51', 7527)
('R51', 7323)
('K76', 7175)
('Z76', 7142)
('R42', 7133)
```

```
('N17', 7130)
('H25', 7047)
('L97', 7010)
('M48', 6829)
('M62', 6795)
('R00', 6751)
('F17', 6672)
('C50', 6663)
('R94', 6418)
('I70', 6262)
('Z86', 6222)
('M99', 5917)
('R19', 5885)
('J30', 5826)
('K57', 5809)
('R26', 5682)
('R60', 5627)
('K29', 5504)
('R11', 5477)
('R35', 5335)
('E83', 5316)
('Z99', 5282)
('D63', 5163)
('M75', 5106)
('F10', 5106)
('H40', 5093)
('K80', 5089)
('Z87', 5057)
('Z96', 5018)
('I69', 4976)
('N20', 4868)
('L60', 4813)
('Z98', 4744)
('J18', 4739)
('F20', 4725)
('F31', 4642)
('H35', 4563)
('I73', 4536)
('R09', 4418)
('D69', 4325)
('K59', 4186)
('M16', 4138)
('J06', 4082)
('I13', 4081)
('I87', 4013)
('C61', 4008)
('G40', 3974)
('I63', 3957)
('M06', 3921)
('R30', 3913)
('Z94', 3861)
('R74', 3859)
('R91', 3854)
('R31', 3793)
('Z09', 3770)
('H52', 3725)
('R97', 3673)
('Z47', 3664)
('E29', 3599)
('R55', 3503)
```

```
('F25', 3475)
('D68', 3462)
('R20', 3452)
('R80', 3414)
('K64', 3404)
('Z85', 3394)
('N25', 3365)
('I21', 3329)
('Z95', 3328)
('F43', 3316)
('E53', 3280)
('K74', 3271)
('R32', 3270)
('M81', 3236)
('R41', 3219)
('I83', 3218)
('M10', 3204)
('E04', 3199)
('Z03', 3165)
('N95', 3153)
('R39', 3026)
('D12', 3012)
('D72', 3005)
('G93', 2979)
('Z48', 2928)
('G56', 2907)
('S83', 2889)
('Z88', 2863)
('D25', 2808)
('G43', 2774)
('B20', 2756)
('N28', 2728)
('E10', 2724)
('N13', 2703)
('K92', 2697)
('S82', 2681)
('M72', 2656)
('I82', 2618)
('N30', 2617)
('K63', 2591)
('M50', 2566)
('M65', 2562)
('M20', 2542)
('L02', 2511)
('M15', 2508)
('E86', 2502)
('L89', 2502)
('M77', 2489)
('Z91', 2489)
('Z90', 2460)
('G62', 2454)
('K62', 2446)
('I49', 2407)
('L57', 2308)
('L84', 2308)
('R92', 2305)
('F11', 2305)
('R22', 2288)
('E05', 2260)
('R50', 2234)
```

```
('R29', 2199)
('J02', 2178)
('J20', 2171)
('I16', 2148)
('M43', 2147)
('K52', 2129)
('J98', 2127)
('M85', 2117)
('N81', 2072)
('M21', 2057)
('S09', 2027)
('R03', 2018)
('N63', 2017)
('C18', 2003)
('L30', 1975)
('N32', 1966)
('H04', 1936)
('L82', 1922)
('N52', 1921)
('C91', 1921)
('I42', 1909)
('M46', 1908)
('D48', 1905)
('C54', 1888)
('R52', 1885)
('L40', 1883)
('I47', 1870)
('H43', 1863)
('R21', 1856)
('K44', 1852)
('J34', 1850)
('N64', 1816)
('R13', 1806)
('J32', 1789)
('K43', 1788)
('Z78', 1786)
('M86', 1781)
('B96', 1781)
('J12', 1778)
('K31', 1776)
('I51', 1766)
('H61', 1762)
('E79', 1752)
('L98', 1750)
('S42', 1748)
('N92', 1725)
('I20', 1721)
('R33', 1706)
('I26', 1698)
('H02', 1626)
('T81', 1621)
('L85', 1590)
('N83', 1585)
('M70', 1580)
('M32', 1570)
('K70', 1560)
('M33', 1551)
('M05', 1548)
('D84', 1544)
('D75', 1538)
```

```
('C44', 1524)
('M76', 1521)
('C34', 1514)
('I95', 1511)
('K42', 1511)
('E88', 1508)
('S52', 1508)
('N76', 1503)
('H91', 1499)
('R76', 1493)
('C78', 1480)
('J90', 1471)
('K85', 1464)
('D51', 1460)
('K22', 1446)
('B37', 1443)
('D62', 1439)
('F03', 1427)
('G44', 1422)
('T84', 1418)
('K40', 1407)
('R82', 1388)
('J01', 1387)
('I89', 1368)
('E08', 1363)
('H53', 1358)
('N93', 1356)
('K56', 1351)
('H81', 1343)
('I67', 1334)
('B18', 1315)
('S39', 1302)
('H10', 1299)
('R63', 1295)
('F15', 1291)
('R56', 1289)
('Z45', 1281)
('E07', 1277)
('G45', 1266)
('R93', 1264)
('Z93', 1263)
('J40', 1261)
('Z30', 1260)
('N91', 1254)
('Z02', 1252)
('M13', 1247)
('R25', 1240)
('C22', 1230)
('Z46', 1227)
('I77', 1222)
('M23', 1210)
('I27', 1207)
('Z72', 1194)
('R16', 1188)
('E21', 1178)
('I65', 1177)
('S72', 1175)
('K75', 1161)
('E06', 1154)
('D61', 1152)
```

```
('E20', 1150)
('L29', 1145)
('C90', 1144)
('R18', 1144)
('T82', 1140)
('D22', 1138)
('B34', 1132)
('M53', 1122)
('H26', 1120)
('G57', 1112)
('F39', 1104)
('I35', 1099)
('S46', 1091)
('M35', 1071)
('C79', 1070)
('Z80', 1066)
('I34', 1066)
('S43', 1063)
('H90', 1042)
('I71', 1039)
('S92', 1003)
('G20', 1003)
('S01', 994)
('G63', 991)
('G82', 984)
('H93', 980)
('R27', 973)
('F29', 973)
('L91', 972)
('Z59', 970)
('R65', 964)
('N89', 961)
('S22', 951)
('M12', 937)
('G25', 934)
('D47', 933)
('C92', 928)
('S06', 927)
('L72', 920)
('H16', 919)
('Z89', 916)
('L81', 914)
('R59', 907)
('K25', 906)
('S93', 906)
('F51', 897)
('S00', 893)
('D70', 891)
('D49', 890)
('N31', 889)
('R14', 885)
('D17', 877)
('R78', 876)
('L08', 875)
('E13', 872)
('D35', 869)
('I44', 866)
('H11', 866)
('R69', 865)
('G35', 862)
```

```
('H92', 862)
('M67', 858)
('N10', 847)
('Z74', 841)
('E89', 841)
('J84', 816)
('F12', 811)
('R87', 809)
('H60', 805)
('N85', 800)
('C64', 796)
('K86', 792)
('C20', 790)
('K35', 790)
('N50', 785)
('N12', 784)
('S62', 781)
('S32', 776)
('C67', 767)
('C73', 764)
('K02', 762)
('A04', 761)
('I15', 759)
('G60', 757)
('S80', 756)
('T78', 747)
('E61', 743)
('J43', 740)
('S91', 738)
('K51', 736)
('N60', 734)
('C77', 733)
('C56', 726)
('R47', 719)
('M96', 718)
('B19', 712)
('L73', 710)
('Z04', 706)
('B02', 704)
('T14', 700)
('J22', 697)
('K83', 696)
('K30', 692)
('Z17', 690)
('I45', 687)
('R04', 683)
('B07', 683)
('G31', 681)
('K72', 680)
('M41', 679)
('C25', 679)
('K05', 677)
('J81', 671)
('S81', 670)
('E16', 669)
('Z28', 663)
('B95', 655)
('H66', 644)
('M94', 640)
('G81', 640)
```

```
('S02', 626)
('M22', 621)
('I61', 619)
('Z53', 616)
('G51', 616)
('K08', 611)
('E34', 610)
('K82', 609)
('T85', 608)
('F19', 602)
('S61', 601)
('K81', 597)
('W19', 588)
('L50', 585)
('R40', 577)
('M18', 576)
('T83', 572)
('K91', 562)
('G92', 561)
('F90', 560)
('C83', 557)
('J15', 553)
('S90', 550)
('M60', 550)
('C16', 550)
('I24', 549)
('G90', 548)
('R23', 544)
('N94', 544)
('R01', 543)
('M89', 543)
('H54', 541)
('Z43', 539)
('S31', 536)
('E27', 536)
('S63', 532)
('L65', 532)
('Z92', 528)
('N48', 528)
('N84', 525)
('M1A', 523)
('E44', 522)
('E23', 516)
('M24', 513)
('E56', 513)
('L20', 509)
('S29', 505)
('R77', 501)
('K65', 501)
('K58', 501)
('H57', 492)
('E46', 492)
('K50', 492)
('F13', 487)
('O09', 487)
('H34', 485)
('M71', 483)
('N43', 481)
('D46', 478)
('S16', 477)
```

```
('H33', 477)
('Z21', 477)
('D18', 476)
('C53', 472)
('N34', 471)
('K20', 467)
('Z82', 465)
('J10', 464)
('N41', 462)
('D05', 459)
('I31', 457)
('S33', 455)
('R57', 449)
('D89', 446)
('R49', 442)
('Z08', 442)
('I85', 440)
('K04', 438)
('S89', 438)
('B00', 437)
('F23', 432)
('Z32', 431)
('S20', 431)
('F52', 426)
('R44', 421)
('I62', 420)
('G83', 417)
('D23', 412)
('J41', 410)
('S86', 408)
('S69', 407)
('Z3A', 405)
('C85', 405)
('K66', 401)
('L21', 398)
('M31', 396)
('F34', 393)
('C55', 392)
('G30', 392)
('S13', 391)
('I96', 389)
('J69', 389)
('K12', 383)
('A40', 381)
('T86', 378)
('R45', 368)
('S99', 368)
('K60', 367)
('H01', 366)
('E28', 366)
('H69', 365)
('Z83', 362)
('H00', 361)
('G70', 360)
('A53', 357)
('D45', 356)
('R12', 353)
('B97', 353)
('C49', 353)
('J31', 349)
```

```
('D83', 346)
('N61', 342)
('Y92', 342)
('I80', 341)
('R62', 340)
('M14', 339)
('R70', 333)
('M45', 331)
('L90', 329)
('S76', 326)
('S60', 323)
('M00', 321)
('T40', 321)
('C43', 319)
('A08', 318)
('G50', 315)
('N99', 312)
('D53', 311)
('L76', 311)
('U09', 308)
('M26', 305)
('D24', 303)
('S30', 302)
('F02', 300)
('G61', 299)
('Z49', 299)
('S19', 298)
('E80', 297)
('D32', 296)
('I46', 296)
('J47', 296)
('L71', 295)
('H65', 294)
('S49', 291)
('Z29', 290)
('E26', 290)
('T43', 290)
('S70', 288)
('N23', 288)
('R89', 286)
('F14', 285)
('T87', 284)
('O10', 282)
('C15', 281)
('E75', 279)
('K13', 278)
('J39', 275)
('E72', 273)
('A09', 273)
('L23', 272)
('C80', 270)
('N49', 269)
('C07', 269)
('M87', 269)
('Z16', 268)
('K11', 268)
('K61', 267)
('N42', 267)
('I72', 265)
('K03', 264)
```

```
('H18', 263)
('H44', 262)
('Q66', 262)
('M84', 262)
('J80', 260)
('J11', 259)
('G95', 259)
('E43', 258)
('N80', 253)
('A49', 251)
('C71', 250)
('K73', 248)
('C82', 248)
('G37', 245)
('H47', 244)
('G91', 244)
('E22', 243)
('N73', 243)
('W01', 242)
('N21', 242)
('K26', 241)
('L53', 239)
('R15', 238)
('K27', 237)
('F45', 236)
('F40', 236)
('N19', 235)
('R61', 235)
('T50', 231)
('O24', 229)
('S50', 225)
('Q61', 223)
('T45', 219)
('F99', 219)
('R71', 219)
('N35', 217)
('L70', 216)
('N47', 216)
('J42', 216)
('I60', 213)
('N45', 209)
('Z34', 209)
('J00', 208)
('G58', 207)
('J38', 205)
('M40', 205)
('N04', 204)
('S40', 203)
('I36', 203)
('K46', 202)
('G24', 197)
('D21', 196)
('L25', 194)
('F79', 194)
('D80', 194)
('M80', 193)
('Z97', 190)
('S51', 190)
('N05', 187)
('J33', 187)
```

```
('N88', 186)
('F06', 186)
('L68', 186)
('K94', 186)
('J03', 184)
('G54', 182)
('J91', 181)
('G80', 180)
('D59', 180)
('C88', 180)
('Z41', 179)
('S23', 179)
('K90', 178)
('C21', 178)
('S79', 176)
('N87', 173)
('L28', 173)
('C81', 173)
('E63', 172)
('S05', 171)
('E01', 167)
('O99', 167)
('N90', 166)
('L11', 165)
('J95', 164)
('S21', 163)
('I08', 162)
('A60', 161)
('Z15', 161)
('F01', 160)
('T80', 160)
('M27', 160)
('D86', 158)
('H05', 157)
('F22', 157)
('R43', 157)
('F50', 156)
('K71', 154)
('Z63', 153)
('Z56', 153)
('J93', 150)
('N02', 150)
('I97', 149)
('W18', 148)
('Z60', 148)
('C57', 147)
('R90', 147)
('B36', 146)
('D52', 146)
('K55', 145)
('J94', 144)
('F42', 144)
('D41', 143)
('N62', 142)
('G72', 140)
('D13', 139)
('T24', 139)
('D16', 139)
('R58', 137)
('S59', 137)
```

```
('C23', 136)
('Z81', 136)
('B86', 135)
('I78', 134)
('F91', 134)
('Z22', 133)
('E74', 132)
('O34', 129)
('C31', 126)
('S71', 125)
('L93', 125)
('Y93', 124)
('D57', 123)
('L63', 123)
('I79', 122)
('S27', 122)
('T63', 121)
('M11', 121)
('I99', 120)
('T46', 120)
('R85', 120)
('Z77', 120)
('E65', 120)
('G64', 120)
('O35', 120)
('F05', 119)
('Z73', 117)
('D31', 117)
('D10', 117)
('M34', 117)
('D39', 116)
('L80', 115)
('S36', 112)
('I33', 112)
('I38', 112)
('Q05', 112)
('Q21', 111)
('H20', 111)
('H49', 110)
('F60', 109)
('L05', 109)
('J35', 109)
('F44', 109)
('S66', 109)
('D04', 108)
('C46', 108)
('N65', 107)
('H31', 107)
('I66', 107)
('D36', 106)
('A63', 106)
('S73', 106)
('T38', 106)
('S53', 106)
('Z65', 104)
('C7A', 103)
('H72', 102)
('R17', 101)
('I05', 99)
('I81', 99)
```

```
('G04', 98)
('D27', 98)
('A64', 98)
('E02', 97)
('I74', 97)
('B44', 97)
('S96', 96)
('C19', 95)
('S12', 95)
('L83', 94)
('C08', 94)
('I86', 93)
('D3A', 93)
('T88', 92)
('T42', 92)
('J09', 92)
('W57', 92)
('C7B', 91)
('C75', 90)
('Q28', 90)
('C41', 90)
('L66', 89)
('I07', 88)
('A54', 88)
('S98', 88)
('B17', 88)
('J04', 87)
('K95', 87)
('L10', 87)
('B45', 86)
('L92', 84)
('O26', 84)
('K14', 83)
('D33', 83)
('X50', 83)
('O20', 82)
('K28', 82)
('H17', 80)
('A02', 80)
('S67', 79)
('N82', 79)
('T79', 79)
('M66', 79)
('K06', 79)
('G99', 79)
('T18', 78)
('Z42', 78)
('H21', 78)
('D73', 78)
('C32', 77)
('N70', 77)
('R46', 76)
('S41', 76)
('D37', 76)
('C52', 76)
('F98', 74)
('L24', 74)
('E51', 74)
('A05', 74)
('R81', 74)
```

```
('C10', 74)
('F09', 73)
('B99', 73)
('R54', 72)
('B25', 72)
('H50', 71)
('A48', 71)
('Z66', 71)
('F59', 70)
('E24', 70)
('N75', 69)
('D58', 69)
('T22', 69)
('F70', 68)
('A59', 68)
('C76', 68)
('A15', 68)
('M95', 67)
('Z33', 67)
('R36', 67)
('D11', 67)
('H59', 66)
('H27', 66)
('F07', 66)
('T21', 66)
('Y84', 66)
('T16', 65)
('X58', 65)
('G97', 65)
('E67', 64)
('N15', 64)
('D56', 63)
('Q24', 63)
('Q82', 63)
('D34', 63)
('S14', 63)
('C66', 63)
('A88', 62)
('X32', 62)
('B15', 62)
('F63', 61)
('T39', 60)
('N26', 60)
('T07', 60)
('E85', 60)
('Y90', 59)
('Y81', 59)
('B38', 58)
('D44', 58)
('N11', 58)
('W10', 57)
('E09', 57)
('Z39', 57)
('H83', 56)
('N36', 56)
('T25', 56)
('E25', 55)
('I43', 55)
('N77', 55)
('M93', 55)
```

```
('F84', 55)
('L59', 55)
('L27', 54)
('T56', 53)
('G46', 53)
('G96', 53)
('R64', 53)
('W22', 52)
('T51', 52)
('Y83', 52)
('S88', 52)
('C95', 52)
('E64', 51)
('D66', 51)
('C62', 51)
('I28', 50)
('Z37', 50)
('F30', 50)
('T30', 50)
('O23', 50)
('N44', 49)
('Q25', 49)
('D03', 49)
('H46', 48)
('Q44', 48)
('G21', 48)
('K00', 48)
('A52', 48)
('R34', 48)
('B91', 47)
('T65', 47)
('K38', 47)
('T15', 46)
('N97', 46)
('H62', 46)
('Z14', 45)
('H70', 45)
('A56', 44)
('L64', 44)
('Z36', 44)
('L94', 44)
('M97', 44)
('I88', 44)
('Q87', 43)
('L01', 42)
('T31', 42)
('N72', 42)
('D07', 42)
('T67', 41)
('E71', 41)
('K45', 41)
('D09', 41)
('F69', 41)
('D14', 41)
('N08', 40)
('Y71', 40)
('M02', 40)
('A74', 40)
('D28', 39)
('C24', 39)
```

```
('A96', 39)
('J21', 39)
('C72', 39)
('Q27', 38)
('L56', 38)
('G71', 38)
('C48', 38)
('M01', 38)
('Z84', 37)
('H15', 37)
('J16', 36)
('Z64', 36)
('Q76', 36)
('B48', 36)
('Q23', 36)
('M49', 35)
('Q63', 34)
('S37', 34)
('J70', 34)
('M07', 34)
('M08', 34)
('Y99', 33)
('B33', 33)
('E73', 33)
('G06', 33)
('S24', 33)
('C11', 33)
('N46', 33)
('T17', 33)
('Z75', 32)
('T23', 32)
('W54', 32)
('Z31', 32)
('D81', 32)
('T19', 31)
('A07', 31)
('B69', 31)
('N53', 31)
('C51', 31)
('C01', 31)
('F48', 30)
('K37', 30)
('W26', 30)
('B30', 30)
('T20', 30)
('D42', 29)
('D30', 29)
('I30', 29)
('H51', 28)
('H28', 28)
('H42', 28)
('W55', 28)
('Z18', 28)
('I09', 28)
('O11', 28)
('M92', 28)
('H55', 27)
('I37', 27)
('O13', 27)
('J14', 27)
```

```
('M42', 27)
('Q84', 26)
('Z70', 26)
('D06', 26)
('J36', 26)
('O36', 26)
('Z52', 26)
('S35', 26)
('C02', 26)
('H74', 25)
('D55', 25)
('D15', 25)
('F28', 25)
('Z55', 25)
('A18', 25)
('O16', 25)
('S44', 25)
('D40', 25)
('B16', 24)
('Q78', 24)
('N03', 24)
('W07', 24)
('Z67', 24)
('A87', 24)
('O42', 24)
('L12', 24)
('W11', 24)
('Q99', 24)
('I00', 23)
('J86', 23)
('Q43', 23)
('E59', 23)
('L87', 23)
('K68', 23)
('F80', 22)
('O80', 22)
('T73', 22)
('O14', 22)
('Q45', 22)
('T44', 22)
('O90', 22)
('I75', 22)
('S10', 21)
('K09', 21)
('F64', 21)
('L74', 21)
('S34', 21)
('D29', 21)
('O46', 21)
('E50', 20)
('A46', 20)
('G98', 20)
('Q83', 20)
('Q55', 20)
('C17', 20)
('Q85', 19)
('H68', 19)
('E00', 19)
('F71', 19)
('F73', 19)
```

```
('M88', 19)
('L04', 19)
('Q38', 19)
('J99', 19)
('Q51', 19)
('L43', 19)
('L88', 19)
('O03', 19)
('M91', 19)
('O30', 19)
('Q68', 18)
('A69', 18)
('Q64', 18)
('G14', 18)
('G03', 18)
('N33', 18)
('M61', 18)
('R84', 18)
('O47', 18)
('F93', 18)
('C60', 18)
('C84', 17)
('C69', 17)
('H71', 17)
('D78', 17)
('J67', 17)
('M04', 16)
('J61', 16)
('F95', 16)
('R37', 16)
('B78', 16)
('Z62', 16)
('C4A', 16)
('T59', 16)
('C09', 16)
('L95', 16)
('T36', 15)
('A80', 15)
('N00', 15)
('L42', 15)
('G11', 15)
('D65', 15)
('B57', 15)
('C70', 15)
('R83', 14)
('J37', 14)
('T75', 14)
('S03', 14)
('N29', 14)
('S56', 14)
('W06', 14)
('L26', 14)
('J85', 14)
('T57', 13)
('S78', 13)
('H73', 13)
('B80', 13)
('J92', 13)
('B01', 13)
('Q75', 13)
```

```
('C37', 13)
('T48', 13)
('S85', 13)
('J13', 13)
('S54', 13)
('C94', 13)
('F16', 13)
('E77', 13)
('O48', 12)
('K41', 12)
('B89', 12)
('L55', 12)
('J82', 12)
('B85', 12)
('L13', 12)
('E60', 12)
('M83', 12)
('B87', 12)
('J66', 12)
('D60', 11)
('T68', 11)
('B49', 11)
('G08', 11)
('W45', 11)
('O98', 11)
('O86', 11)
('Q67', 11)
('M30', 11)
('N98', 11)
('Q74', 11)
('A19', 11)
('Q04', 10)
('W20', 10)
('A92', 10)
('W23', 10)
('Q90', 10)
('G12', 10)
('W17', 10)
('Q14', 10)
('Q98', 10)
('H80', 10)
('R48', 10)
('Q72', 10)
('E70', 10)
('S04', 10)
('H82', 10)
('O02', 10)
('G36', 9)
('S84', 9)
('B88', 9)
('W21', 9)
('Q40', 9)
('Z57', 9)
('G23', 9)
('Q10', 9)
('M90', 9)
('C14', 9)
('Y76', 9)
('C74', 9)
('C06', 8)
```

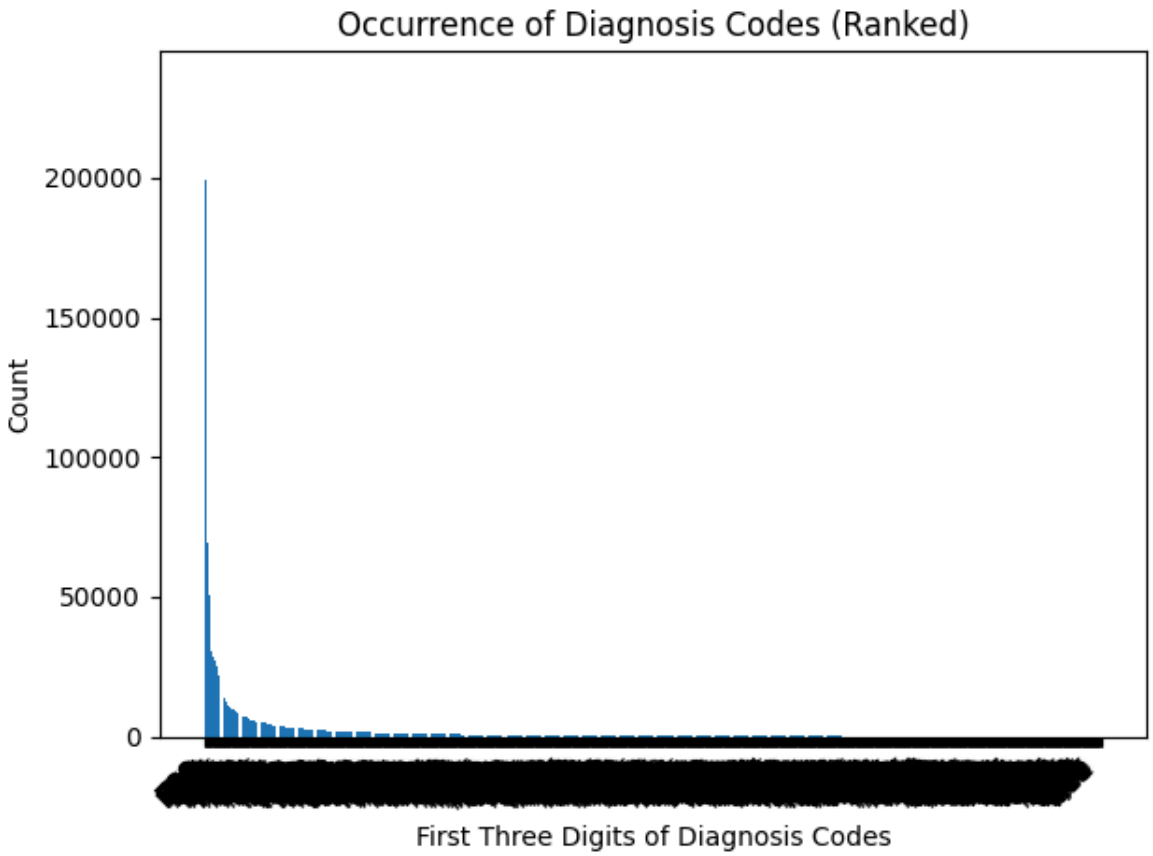
```
('B27', 8)
('W29', 8)
('A81', 8)
('I22', 8)
('Z40', 8)
('C68', 8)
('N14', 8)
('O82', 8)
('T69', 8)
('B40', 8)
('D19', 8)
('O69', 8)
('S64', 8)
('F24', 8)
('I76', 8)
('B08', 8)
('A20', 8)
('O71', 8)
('T70', 7)
('H30', 7)
('F21', 7)
('Q79', 7)
('D43', 7)
('O32', 7)
('L51', 7)
('Q65', 7)
('R86', 7)
('H36', 7)
('O62', 7)
('D38', 7)
('B59', 7)
('O21', 7)
('Z19', 7)
('F81', 6)
('X10', 6)
('X31', 6)
('H95', 6)
('S87', 6)
('Q62', 6)
('E58', 6)
('W25', 6)
('D71', 6)
('O92', 6)
('S28', 6)
('780', 6)
('Q26', 6)
('T76', 6)
('A51', 6)
('D74', 6)
('J68', 5)
('W46', 5)
('A31', 5)
('F54', 5)
('L54', 5)
('Q30', 5)
('B94', 5)
('T49', 5)
('T66', 5)
('W27', 5)
('I40', 5)
```

```
('T47', 5)
('B74', 5)
('B82', 4)
('X12', 4)
('L52', 4)
('T62', 4)
('L44', 4)
('Q39', 4)
('B04', 4)
('Q50', 4)
('Q77', 4)
('C65', 4)
('A37', 4)
('S26', 4)
('N01', 4)
('A42', 4)
('R75', 4)
('D26', 4)
('K36', 4)
('A00', 4)
('O44', 4)
('B58', 4)
('S97', 4)
('E45', 4)
('A50', 4)
('S55', 4)
('T41', 4)
('G00', 4)
('Q60', 3)
('L99', 3)
('Q70', 3)
('N07', 3)
('W50', 3)
('F55', 3)
('L67', 3)
('O76', 3)
('F88', 3)
('O28', 3)
('H67', 3)
('D00', 3)
('S45', 3)
('W00', 3)
('C30', 3)
('X00', 3)
('L86', 3)
('O70', 3)
('N27', 3)
('D20', 2)
('N71', 2)
('D82', 2)
('O60', 2)
('O40', 2)
('W08', 2)
('Y73', 2)
('W05', 2)
('O22', 2)
('B09', 2)
('X19', 2)
('Q33', 2)
('F72', 2)
```

```
('B60', 2)
('T61', 2)
('I23', 2)
('A32', 2)
('L41', 2)
('Q11', 2)
('S57', 2)
('A66', 2)
('T27', 2)
('S11', 2)
('A90', 2)
('I52', 2)
('E40', 2)
('Q35', 2)
('I5A', 2)
('O72', 2)
('S47', 2)
('T54', 2)
('C38', 2)
('Q31', 2)
('C03', 2)
('M63', 2)
('S68', 2)
('O31', 2)
('O01', 2)
('Q13', 2)
('A83', 2)
('S07', 2)
('B73', 2)
('D01', 2)
('Q93', 2)
('E54', 2)
('J05', 2)
('I06', 2)
('G59', 1)
('O73', 1)
('Q80', 1)
('A78', 1)
('A01', 1)
('A43', 1)
('F78', 1)
('C13', 1)
('Y28', 1)
('T74', 1)
('O68', 1)
('W31', 1)
('Q20', 1)
('Q54', 1)
('W51', 1)
('Q89', 1)
('Q15', 1)
('N06', 1)
('T37', 1)
('B77', 1)
('T58', 1)
('W34', 1)
('Q12', 1)
('K87', 1)
('Q22', 1)
('X39', 1)
```

```
('Y95', 1)
('B10', 1)
('A06', 1)
('X04', 1)
('F66', 1)
('W13', 1)
('F89', 1)
('F53', 1)
('L75', 1)
('J17', 1)
('F18', 1)
('W52', 1)
('W89', 1)
```

In [7]:
```python
# Extract labels and values
labels = [item[0] for item in sorted_counts]
values = [item[1] for item in sorted_counts]

# Create the bar chart
plt.bar(labels, values)

# Labels and title
plt.xlabel("First Three Digits of Diagnosis Codes")
plt.ylabel("Count")
plt.title("Occurrence of Diagnosis Codes (Ranked)")

# Optional: Rotate x labels if they are long or overlapping
if len(labels) > 5:
    plt.xticks(rotation=45, ha='right')

# Adjust layout for better display
plt.tight_layout()

# Show the plot
plt.show()
```



In [8]:
```
filtered_df = df[df['proc_code'] == "J3490"]


# 1. Select 'diag_' columns
diag_columns = [col for col in filtered_df.columns if col.startswith('diag_')]
df_diag = filtered_df[diag_columns]

# 2. Combine values
combined_list = df_diag.values.flatten().tolist()
combined_list = [x for x in combined_list if pd.notna(x)]

# 3. Process first three digits (same as before)
first_three_counts = {}
for diag_code in combined_list:
    first_three = diag_code[:3]
    first_three_counts[first_three] = first_three_counts.get(first_three, 0) + 1

sorted_counts = sorted(first_three_counts.items(), key=operator.itemgetter(1), r

for items in sorted_counts:
    print(items)
```

```
('I10', 249)
('E11', 229)
('E78', 139)
('N18', 100)
('I25', 91)
('M54', 81)
('E66', 75)
('Z79', 72)
('R07', 71)
('Z51', 64)
('I48', 56)
('J45', 50)
('G89', 49)
('M19', 46)
('M25', 44)
('R10', 44)
('D68', 40)
('F41', 37)
('Z99', 36)
('K21', 35)
('H25', 34)
('Z68', 34)
('J44', 33)
('Z30', 33)
('M17', 32)
('M47', 28)
('C50', 28)
('D50', 27)
('M79', 27)
('Z87', 26)
('Z88', 25)
('N25', 25)
('R51', 24)
('G47', 23)
('E03', 23)
('M48', 22)
('F33', 22)
('R06', 20)
('R94', 20)
('Z85', 19)
('Z20', 19)
('J20', 19)
('F17', 19)
('I50', 18)
('K29', 18)
('F32', 18)
('Z86', 17)
('M51', 17)
('I12', 17)
('R11', 16)
('M70', 16)
('N20', 16)
('I11', 16)
('K57', 15)
('K44', 15)
('K59', 15)
('C61', 15)
('Z90', 15)
('N13', 15)
('I49', 14)
```

```
('Z11', 14)
('Z01', 14)
('D63', 14)
('F31', 14)
('L02', 14)
('G43', 12)
('K80', 12)
('M16', 12)
('D64', 12)
('R05', 12)
('J30', 12)
('R26', 12)
('H40', 12)
('K43', 12)
('R19', 12)
('N39', 12)
('K40', 12)
('R52', 12)
('Z95', 11)
('C77', 11)
('D25', 11)
('N95', 11)
('T85', 11)
('K42', 11)
('R42', 10)
('I13', 10)
('K64', 10)
('M53', 10)
('N81', 10)
('L72', 10)
('F43', 10)
('T83', 10)
('C18', 10)
('E86', 9)
('Z12', 9)
('D12', 9)
('N40', 9)
('C55', 9)
('M75', 9)
('S01', 9)
('Z91', 9)
('R53', 8)
('C67', 8)
('I47', 8)
('U07', 8)
('K85', 8)
('Z45', 8)
('F10', 8)
('R60', 8)
('Z98', 8)
('C78', 8)
('N12', 8)
('M32', 8)
('K31', 8)
('K62', 8)
('J18', 7)
('S09', 7)
('E10', 7)
('L03', 7)
('N60', 7)
```

```
('M20', 7)
('H35', 7)
('K22', 7)
('M23', 7)
('K60', 6)
('J96', 6)
('C43', 6)
('G44', 6)
('Z48', 6)
('I83', 6)
('T81', 6)
('L98', 6)
('J32', 6)
('R20', 6)
('H18', 6)
('D17', 6)
('Z17', 6)
('C34', 6)
('G80', 6)
('K76', 6)
('N85', 6)
('T84', 6)
('S83', 5)
('N32', 5)
('Z23', 5)
('T78', 5)
('I67', 5)
('N84', 5)
('R41', 5)
('K74', 5)
('C22', 5)
('M60', 5)
('N48', 5)
('J34', 5)
('R16', 5)
('N28', 5)
('K86', 5)
('C54', 5)
('Z42', 4)
('Z92', 4)
('N64', 4)
('S02', 4)
('N93', 4)
('R00', 4)
('S70', 4)
('R74', 4)
('C16', 4)
('N17', 4)
('L25', 4)
('M96', 4)
('N83', 4)
('T82', 4)
('T86', 4)
('D18', 4)
('M65', 4)
('I35', 4)
('R03', 4)
('J43', 4)
('K51', 4)
('H02', 4)
```

```
('N50', 4)
('I31', 4)
('N87', 4)
('E87', 4)
('M81', 4)
('C83', 4)
('F20', 4)
('G56', 4)
('N44', 4)
('C56', 4)
('J15', 4)
('M94', 3)
('C79', 3)
('Z93', 3)
('J90', 3)
('N89', 3)
('N73', 3)
('K92', 3)
('H52', 3)
('I42', 3)
('E26', 3)
('I87', 3)
('H33', 3)
('K63', 3)
('M62', 3)
('Y83', 3)
('H53', 3)
('Z00', 3)
('I44', 3)
('I08', 3)
('G70', 3)
('J98', 3)
('Z49', 3)
('I81', 3)
('T80', 3)
('M86', 3)
('I89', 3)
('D84', 3)
('A41', 3)
('K83', 3)
('M89', 3)
('N49', 3)
('N30', 2)
('S67', 2)
('E88', 2)
('J39', 2)
('D28', 2)
('B02', 2)
('L76', 2)
('G25', 2)
('H54', 2)
('H81', 2)
('D11', 2)
('N35', 2)
('F25', 2)
('D57', 2)
('N61', 2)
('E83', 2)
('L90', 2)
('L97', 2)
```

```
('S51', 2)
('R79', 2)
('M10', 2)
('I15', 2)
('D46', 2)
('K61', 2)
('K20', 2)
('J12', 2)
('E04', 2)
('S32', 2)
('D23', 2)
('F42', 2)
('H90', 2)
('R50', 2)
('R29', 2)
('M21', 2)
('K56', 2)
('G61', 2)
('E29', 2)
('D75', 2)
('N52', 2)
('K06', 2)
('M43', 2)
('I65', 2)
('I16', 2)
('C20', 2)
('K95', 2)
('Q24', 2)
('R56', 2)
('C73', 2)
('C53', 2)
('H57', 2)
('B19', 2)
('B07', 2)
('B20', 2)
('Y92', 2)
('E89', 2)
('J95', 2)
('S52', 2)
('C62', 2)
('K82', 2)
('F12', 2)
('H43', 2)
('Z96', 2)
('Z41', 2)
('B95', 2)
('D27', 2)
('M50', 2)
('S60', 2)
('S90', 2)
('R09', 2)
('C10', 2)
('I71', 2)
('N10', 2)
('L05', 2)
('S63', 2)
('J02', 2)
('K27', 2)
('D35', 2)
('S31', 2)
```

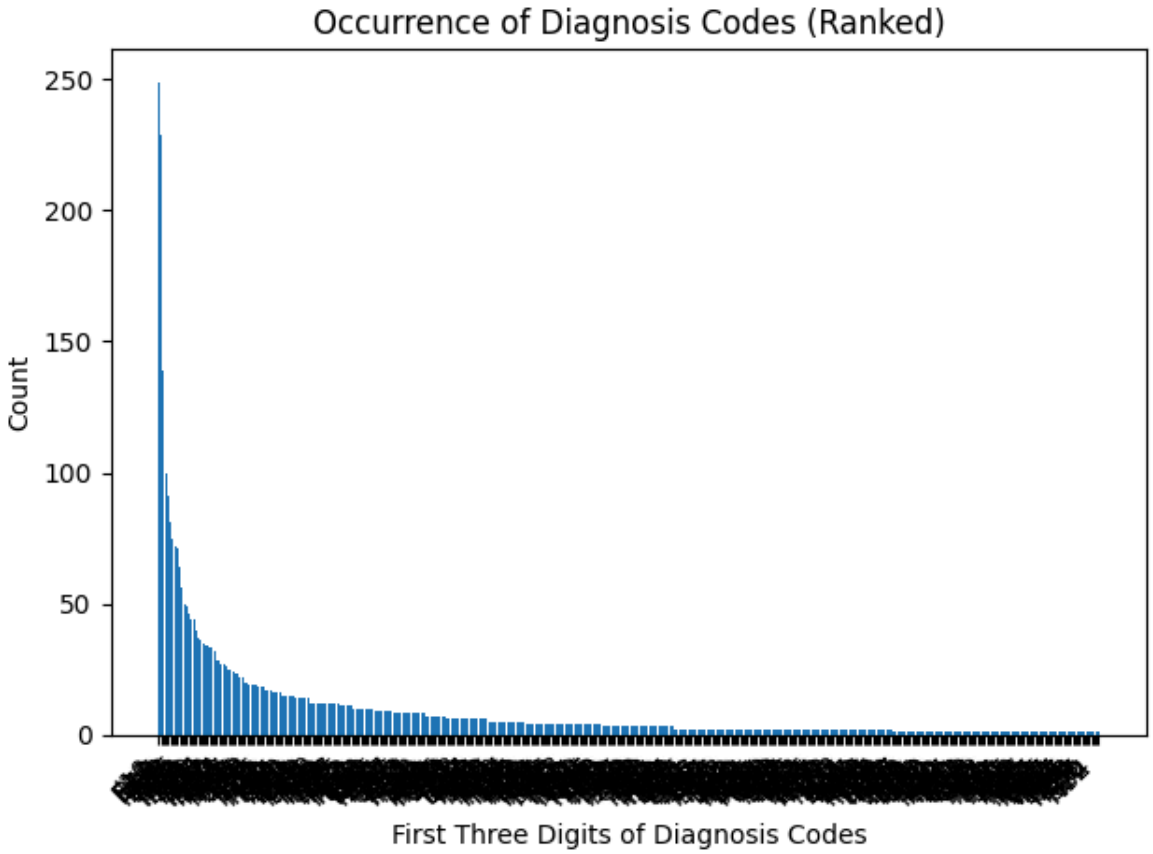
```
('C91', 2)
('K08', 2)
('R27', 2)
('S30', 2)
('R80', 2)
('D24', 2)
('C90', 2)
('G06', 2)
('M18', 2)
('R91', 2)
('J84', 2)
('N43', 2)
('C84', 2)
('J38', 2)
('N65', 2)
('D05', 2)
('H04', 1)
('H10', 1)
('Z73', 1)
('R45', 1)
('I43', 1)
('C57', 1)
('J06', 1)
('K04', 1)
('W22', 1)
('R30', 1)
('D72', 1)
('F11', 1)
('E55', 1)
('D51', 1)
('N36', 1)
('R33', 1)
('E27', 1)
('G62', 1)
('Z89', 1)
('R70', 1)
('E28', 1)
('L91', 1)
('L50', 1)
('S00', 1)
('M12', 1)
('K73', 1)
('H91', 1)
('B97', 1)
('K52', 1)
('R31', 1)
('L30', 1)
('Z71', 1)
('R93', 1)
('Z03', 1)
('B15', 1)
('R73', 1)
('Z78', 1)
('K05', 1)
('G24', 1)
('I45', 1)
('Q21', 1)
('C92', 1)
('H27', 1)
('R01', 1)
```

```
('F51', 1)
('I69', 1)
('N63', 1)
('K91', 1)
('E07', 1)
('R35', 1)
('I70', 1)
('H16', 1)
('R61', 1)
('B33', 1)
('T45', 1)
('K90', 1)
('Z80', 1)
('H21', 1)
('R18', 1)
('T46', 1)
('N41', 1)
('C31', 1)
('E80', 1)
('M22', 1)
('R47', 1)
('K13', 1)
('E21', 1)
('M13', 1)
('G35', 1)
('D69', 1)
('I27', 1)
('D80', 1)
('N23', 1)
('W01', 1)
('R97', 1)
('Z94', 1)
('D47', 1)
('R55', 1)
('J01', 1)
('N80', 1)
('G90', 1)
('M14', 1)
('L29', 1)
('K35', 1)
('I77', 1)
('F90', 1)
('S20', 1)
('N72', 1)
('I34', 1)
```

EDA-1 Distribution of Diagnosis Codes

In [9]:
```
# Extract labels and values
labels = [item[0] for item in sorted_counts]
values = [item[1] for item in sorted_counts]

# Create the bar chart
plt.bar(labels, values)

# Labels and title
plt.xlabel("First Three Digits of Diagnosis Codes")
plt.ylabel("Count")
plt.title("Occurrence of Diagnosis Codes (Ranked)")
```

```
# Optional: Rotate x labels if they are long or overlapping
if len(labels) > 5:
    plt.xticks(rotation=45, ha='right')

# Adjust layout for better display
plt.tight_layout()

# Show the plot
plt.show()
```



Insights :

- This is a skewed graph which depicts that only a small proportion of diagnosis codes occur very frequently.
- Approximately one diagnosis code stands out with a count of about 200,000 this suggest the code is extremely common in the dataset
- There is overall decling trend which suggests there are a small subset of common conditions, the rest of them are less frequently diagnosed

In [10]:
```
# 1. Select 'diag_' columns
diag_columns = [col for col in filtered_df.columns if col.startswith('diag_')]
df_diag = filtered_df[diag_columns]

# 2. Combine values
combined_list = df_diag.values.flatten().tolist()
combined_list = [x for x in combined_list if pd.notna(x)]

# 3. Process first three digits (same as before)
first_three_counts = {}
```

```
for diag_code in combined_list:
    first_three = diag_code[:3]
    first_three_counts[first_three] = first_three_counts.get(first_three, 0) + 1

sorted_counts = sorted(first_three_counts.items(), key=operator.itemgetter(1), r

for items in sorted_counts:
    print(items)
```

```
('I10', 249)
('E11', 229)
('E78', 139)
('N18', 100)
('I25', 91)
('M54', 81)
('E66', 75)
('Z79', 72)
('R07', 71)
('Z51', 64)
('I48', 56)
('J45', 50)
('G89', 49)
('M19', 46)
('M25', 44)
('R10', 44)
('D68', 40)
('F41', 37)
('Z99', 36)
('K21', 35)
('H25', 34)
('Z68', 34)
('J44', 33)
('Z30', 33)
('M17', 32)
('M47', 28)
('C50', 28)
('D50', 27)
('M79', 27)
('Z87', 26)
('Z88', 25)
('N25', 25)
('R51', 24)
('G47', 23)
('E03', 23)
('M48', 22)
('F33', 22)
('R06', 20)
('R94', 20)
('Z85', 19)
('Z20', 19)
('J20', 19)
('F17', 19)
('I50', 18)
('K29', 18)
('F32', 18)
('Z86', 17)
('M51', 17)
('I12', 17)
('R11', 16)
('M70', 16)
('N20', 16)
('I11', 16)
('K57', 15)
('K44', 15)
('K59', 15)
('C61', 15)
('Z90', 15)
('N13', 15)
('I49', 14)
```

```
('Z11', 14)
('Z01', 14)
('D63', 14)
('F31', 14)
('L02', 14)
('G43', 12)
('K80', 12)
('M16', 12)
('D64', 12)
('R05', 12)
('J30', 12)
('R26', 12)
('H40', 12)
('K43', 12)
('R19', 12)
('N39', 12)
('K40', 12)
('R52', 12)
('Z95', 11)
('C77', 11)
('D25', 11)
('N95', 11)
('T85', 11)
('K42', 11)
('R42', 10)
('I13', 10)
('K64', 10)
('M53', 10)
('N81', 10)
('L72', 10)
('F43', 10)
('T83', 10)
('C18', 10)
('E86', 9)
('Z12', 9)
('D12', 9)
('N40', 9)
('C55', 9)
('M75', 9)
('S01', 9)
('Z91', 9)
('R53', 8)
('C67', 8)
('I47', 8)
('U07', 8)
('K85', 8)
('Z45', 8)
('F10', 8)
('R60', 8)
('Z98', 8)
('C78', 8)
('N12', 8)
('M32', 8)
('K31', 8)
('K62', 8)
('J18', 7)
('S09', 7)
('E10', 7)
('L03', 7)
('N60', 7)
```

```
('M20', 7)
('H35', 7)
('K22', 7)
('M23', 7)
('K60', 6)
('J96', 6)
('C43', 6)
('G44', 6)
('Z48', 6)
('I83', 6)
('T81', 6)
('L98', 6)
('J32', 6)
('R20', 6)
('H18', 6)
('D17', 6)
('Z17', 6)
('C34', 6)
('G80', 6)
('K76', 6)
('N85', 6)
('T84', 6)
('S83', 5)
('N32', 5)
('Z23', 5)
('T78', 5)
('I67', 5)
('N84', 5)
('R41', 5)
('K74', 5)
('C22', 5)
('M60', 5)
('N48', 5)
('J34', 5)
('R16', 5)
('N28', 5)
('K86', 5)
('C54', 5)
('Z42', 4)
('Z92', 4)
('N64', 4)
('S02', 4)
('N93', 4)
('R00', 4)
('S70', 4)
('R74', 4)
('C16', 4)
('N17', 4)
('L25', 4)
('M96', 4)
('N83', 4)
('T82', 4)
('T86', 4)
('D18', 4)
('M65', 4)
('I35', 4)
('R03', 4)
('J43', 4)
('K51', 4)
('H02', 4)
```

```
('N50', 4)
('I31', 4)
('N87', 4)
('E87', 4)
('M81', 4)
('C83', 4)
('F20', 4)
('G56', 4)
('N44', 4)
('C56', 4)
('J15', 4)
('M94', 3)
('C79', 3)
('Z93', 3)
('J90', 3)
('N89', 3)
('N73', 3)
('K92', 3)
('H52', 3)
('I42', 3)
('E26', 3)
('I87', 3)
('H33', 3)
('K63', 3)
('M62', 3)
('Y83', 3)
('H53', 3)
('Z00', 3)
('I44', 3)
('I08', 3)
('G70', 3)
('J98', 3)
('Z49', 3)
('I81', 3)
('T80', 3)
('M86', 3)
('I89', 3)
('D84', 3)
('A41', 3)
('K83', 3)
('M89', 3)
('N49', 3)
('N30', 2)
('S67', 2)
('E88', 2)
('J39', 2)
('D28', 2)
('B02', 2)
('L76', 2)
('G25', 2)
('H54', 2)
('H81', 2)
('D11', 2)
('N35', 2)
('F25', 2)
('D57', 2)
('N61', 2)
('E83', 2)
('L90', 2)
('L97', 2)
```

```
('S51', 2)
('R79', 2)
('M10', 2)
('I15', 2)
('D46', 2)
('K61', 2)
('K20', 2)
('J12', 2)
('E04', 2)
('S32', 2)
('D23', 2)
('F42', 2)
('H90', 2)
('R50', 2)
('R29', 2)
('M21', 2)
('K56', 2)
('G61', 2)
('E29', 2)
('D75', 2)
('N52', 2)
('K06', 2)
('M43', 2)
('I65', 2)
('I16', 2)
('C20', 2)
('K95', 2)
('Q24', 2)
('R56', 2)
('C73', 2)
('C53', 2)
('H57', 2)
('B19', 2)
('B07', 2)
('B20', 2)
('Y92', 2)
('E89', 2)
('J95', 2)
('S52', 2)
('C62', 2)
('K82', 2)
('F12', 2)
('H43', 2)
('Z96', 2)
('Z41', 2)
('B95', 2)
('D27', 2)
('M50', 2)
('S60', 2)
('S90', 2)
('R09', 2)
('C10', 2)
('I71', 2)
('N10', 2)
('L05', 2)
('S63', 2)
('J02', 2)
('K27', 2)
('D35', 2)
('S31', 2)
```

```
('C91', 2)
('K08', 2)
('R27', 2)
('S30', 2)
('R80', 2)
('D24', 2)
('C90', 2)
('G06', 2)
('M18', 2)
('R91', 2)
('J84', 2)
('N43', 2)
('C84', 2)
('J38', 2)
('N65', 2)
('D05', 2)
('H04', 1)
('H10', 1)
('Z73', 1)
('R45', 1)
('I43', 1)
('C57', 1)
('J06', 1)
('K04', 1)
('W22', 1)
('R30', 1)
('D72', 1)
('F11', 1)
('E55', 1)
('D51', 1)
('N36', 1)
('R33', 1)
('E27', 1)
('G62', 1)
('Z89', 1)
('R70', 1)
('E28', 1)
('L91', 1)
('L50', 1)
('S00', 1)
('M12', 1)
('K73', 1)
('H91', 1)
('B97', 1)
('K52', 1)
('R31', 1)
('L30', 1)
('Z71', 1)
('R93', 1)
('Z03', 1)
('B15', 1)
('R73', 1)
('Z78', 1)
('K05', 1)
('G24', 1)
('I45', 1)
('Q21', 1)
('C92', 1)
('H27', 1)
('R01', 1)
```

```
('F51', 1)
('I69', 1)
('N63', 1)
('K91', 1)
('E07', 1)
('R35', 1)
('I70', 1)
('H16', 1)
('R61', 1)
('B33', 1)
('T45', 1)
('K90', 1)
('Z80', 1)
('H21', 1)
('R18', 1)
('T46', 1)
('N41', 1)
('C31', 1)
('E80', 1)
('M22', 1)
('R47', 1)
('K13', 1)
('E21', 1)
('M13', 1)
('G35', 1)
('D69', 1)
('I27', 1)
('D80', 1)
('N23', 1)
('W01', 1)
('R97', 1)
('Z94', 1)
('D47', 1)
('R55', 1)
('J01', 1)
('N80', 1)
('G90', 1)
('M14', 1)
('L29', 1)
('K35', 1)
('I77', 1)
('F90', 1)
('S20', 1)
('N72', 1)
('I34', 1)
```

In [11]:
```python
# 1. Select 'diag_' columns
#diag_columns = [col for col in df.columns if col.startswith('diag_')]
df_zip = df['patient_short_zip']

# 2. Combine values
combined_list = df_zip.values.flatten().tolist()
combined_list = [x for x in combined_list if pd.notna(x)]

# 3. Process first three digits (same as before)
zips = {}
for diag_code in combined_list:
    #first_three = diag_code[:3]
    zips[diag_code] = zips.get(diag_code, 0) + 1
```

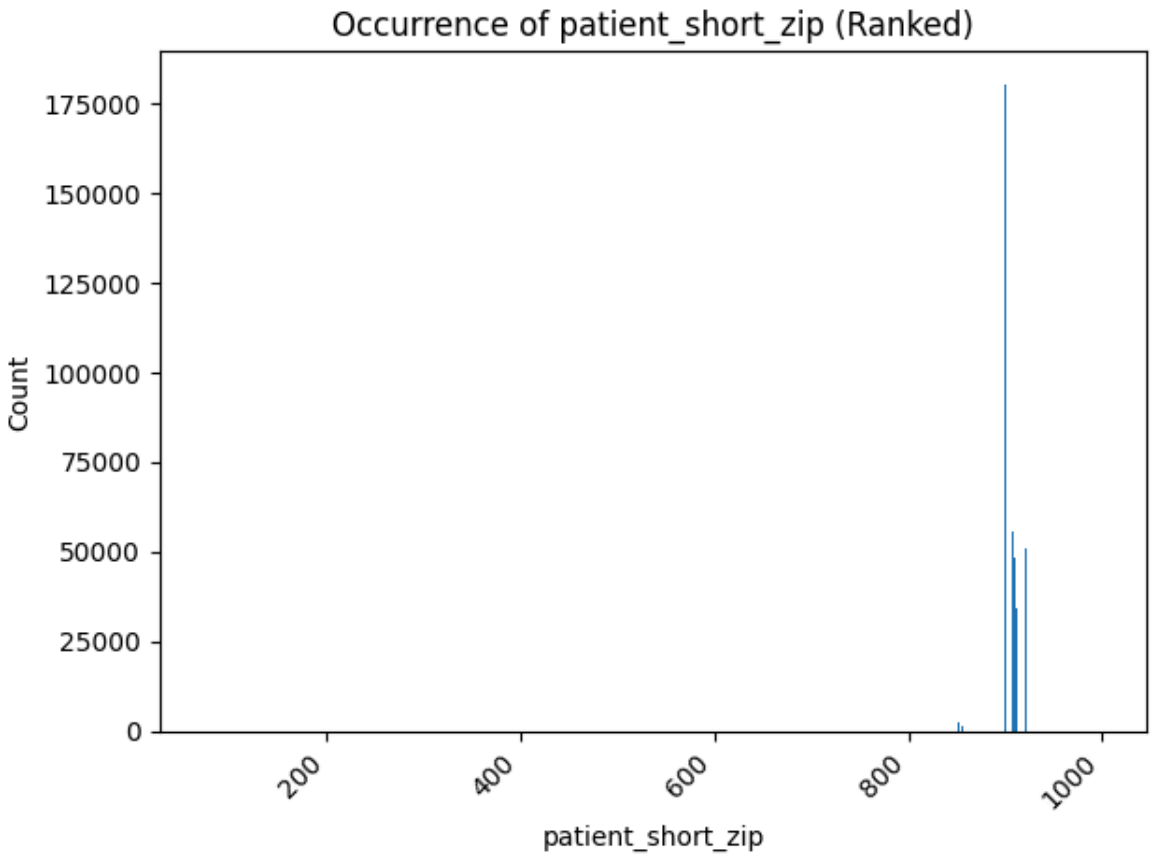
In [12]:
```
#import operator
# Sort the dictionary items by count (descending order)
sorted_counts = sorted(zips.items(), key=operator.itemgetter(1), reverse=True)

for items in sorted_counts:
    print(items)
```

```
(900.0, 180469)
(922.0, 166422)
(908.0, 55562)
(921.0, 51190)
(910.0, 48395)
(913.0, 34571)
(912.0, 34143)
(911.0, 32244)
(853.0, 6499)
(857.0, 2952)
(852.0, 2399)
(864.0, 1480)
(856.0, 1456)
(350.0, 1259)
(351.0, 637)
(352.0, 530)
(863.0, 495)
(859.0, 447)
(855.0, 430)
(995.0, 370)
(996.0, 359)
(860.0, 311)
(997.0, 291)
(344.0, 58)
(194.0, 42)
(850.0, 39)
(272.0, 33)
(891.0, 32)
(923.0, 25)
(304.0, 25)
(998.0, 23)
(320.0, 22)
(91.0, 22)
(629.0, 18)
(606.0, 18)
(770.0, 17)
(105.0, 15)
(787.0, 12)
(841.0, 12)
(895.0, 9)
(483.0, 9)
(333.0, 9)
(328.0, 8)
(527.0, 8)
(112.0, 8)
(370.0, 8)
(372.0, 7)
(773.0, 7)
(722.0, 7)
(170.0, 7)
(330.0, 6)
(924.0, 6)
(334.0, 6)
(528.0, 5)
(720.0, 5)
(752.0, 4)
(776.0, 4)
(524.0, 4)
(337.0, 4)
(612.0, 3)
```

```
(999.0, 3)
(88.0, 3)
(503.0, 3)
(619.0, 3)
(890.0, 3)
(933.0, 2)
(774.0, 2)
(631.0, 2)
(74.0, 2)
(478.0, 2)
(782.0, 2)
(523.0, 2)
(917.0, 2)
(554.0, 2)
(200.0, 2)
(442.0, 2)
(109.0, 1)
(481.0, 1)
(925.0, 1)
(610.0, 1)
(495.0, 1)
(321.0, 1)
(427.0, 1)
(616.0, 1)
(426.0, 1)
(600.0, 1)
(928.0, 1)
(800.0, 1)
(322.0, 1)
(315.0, 1)
(604.0, 1)
(753.0, 1)
(101.0, 1)
```

EDA-2 Distribution of The ZIP code for all the patients

In [13]:
```
# Extract labels and values
labels = [item[0] for item in sorted_counts]
values = [item[1] for item in sorted_counts]

# Create the bar chart
plt.bar(labels, values)

# Labels and title
plt.xlabel("patient_short_zip")
plt.ylabel("Count")
plt.title("Occurrence of patient_short_zip (Ranked)")

# Optional: Rotate x labels if they are long or overlapping
if len(labels) > 5:
    plt.xticks(rotation=45, ha='right')

# Adjust layout for better display
plt.tight_layout()

# Show the plot
plt.show()
```

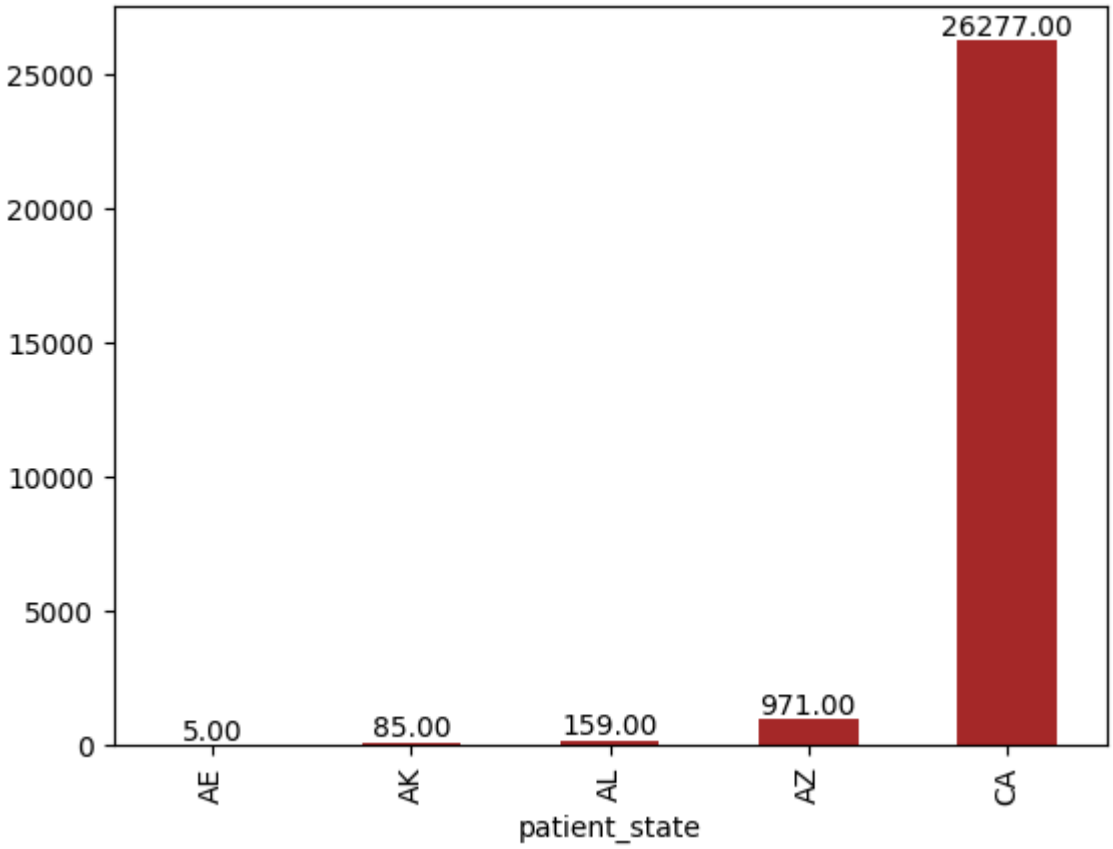


Insights :

- There is a one zip code that is very common in comparison to the others with a count of about 175000 occurances the zip code seems to be for the patients around 900 which indicates the state California
- This shows that the majority of the cases are California based

EDA-3 Distribution of patients on the basis of state

In [14]:
```
# Check the number of patients' states
patient_state_cnt = df.groupby('patient_state')['journey_id'].nunique()

# plot a bar plot
patient_state_cnt .plot.bar(color = 'brown')
for i, v in enumerate(patient_state_cnt):
    plt.text(i, v, f'{v:.2f}', ha='center', va='bottom')

plt.show()
```



Insights

- This is an elaboration of the previous plot, this elucidates that the majority of cases of the data are CA based
- This could give an overiview on state based trends , and could potentially be usefull for healthcare providers especially for resource allocation, and patient demographics.

Taking the obesity data based on the diag codes that start with E66

In [15]:
```
df_obesity_test = df[df['diag_1'].str.startswith("E66")|
      df['diag_2'].str.startswith("E66")|
      df['diag_3'].str.startswith("E66")|
      df['diag_4'].str.startswith("E66")|
      df['diag_5'].str.startswith("E66")]
```

In [16]:
```
df_obesity_test['diag'] = "Obesity"
df_obesity_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 63026 entries, 22 to 623506
Data columns (total 35 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   journey_id         63026 non-null  object 
 1   episode_id         63026 non-null  object 
 2   visit_id           62900 non-null  object 
 3   encounter_id       63026 non-null  object 
 4   claim_date         63026 non-null  object 
 5   patient_state      63026 non-null  object 
 6   patient_short_zip  63018 non-null  float64
 7   patient_age        62224 non-null  float64
 8   patient_gender     63023 non-null  object 
 9   place_of_service   50211 non-null  object 
 10  visit_type         59858 non-null  object 
 11  payor              60251 non-null  object 
 12  payor_channel      58314 non-null  object 
 13  ref_npi            24221 non-null  float64
 14  hcp_npi            49907 non-null  float64
 15  hcp_taxonomy       49357 non-null  object 
 16  hcp_specialty      49357 non-null  object 
 17  hco_npi            62810 non-null  float64
 18  hcp_npi_list       49907 non-null  object 
 19  hco_npi_list       62810 non-null  object 
 20  diag_list          63026 non-null  object 
 21  diag_1             63026 non-null  object 
 22  diag_2             62285 non-null  object 
 23  diag_3             60552 non-null  object 
 24  diag_4             56736 non-null  object 
 25  diag_5             44947 non-null  object 
 26  rev_center_code    9976 non-null   float64
 27  rev_center_units   62896 non-null  float64
 28  proc_code          59121 non-null  object 
 29  proc_modifier      8542 non-null   object 
 30  proc_units         62807 non-null  float64
 31  line_charge        63026 non-null  float64
 32  claim_charge       63026 non-null  float64
 33  smart_allowed      63026 non-null  float64
 34  diag               63026 non-null  object 
dtypes: float64(11), object(24)
memory usage: 17.3+ MB
```

```
C:\Users\meghs\AppData\Local\Temp\ipykernel_11944\2745376594.py:1: SettingWithCop
yWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_obesity_test['diag'] = "Obesity"
```

Understanding Gender Ratio

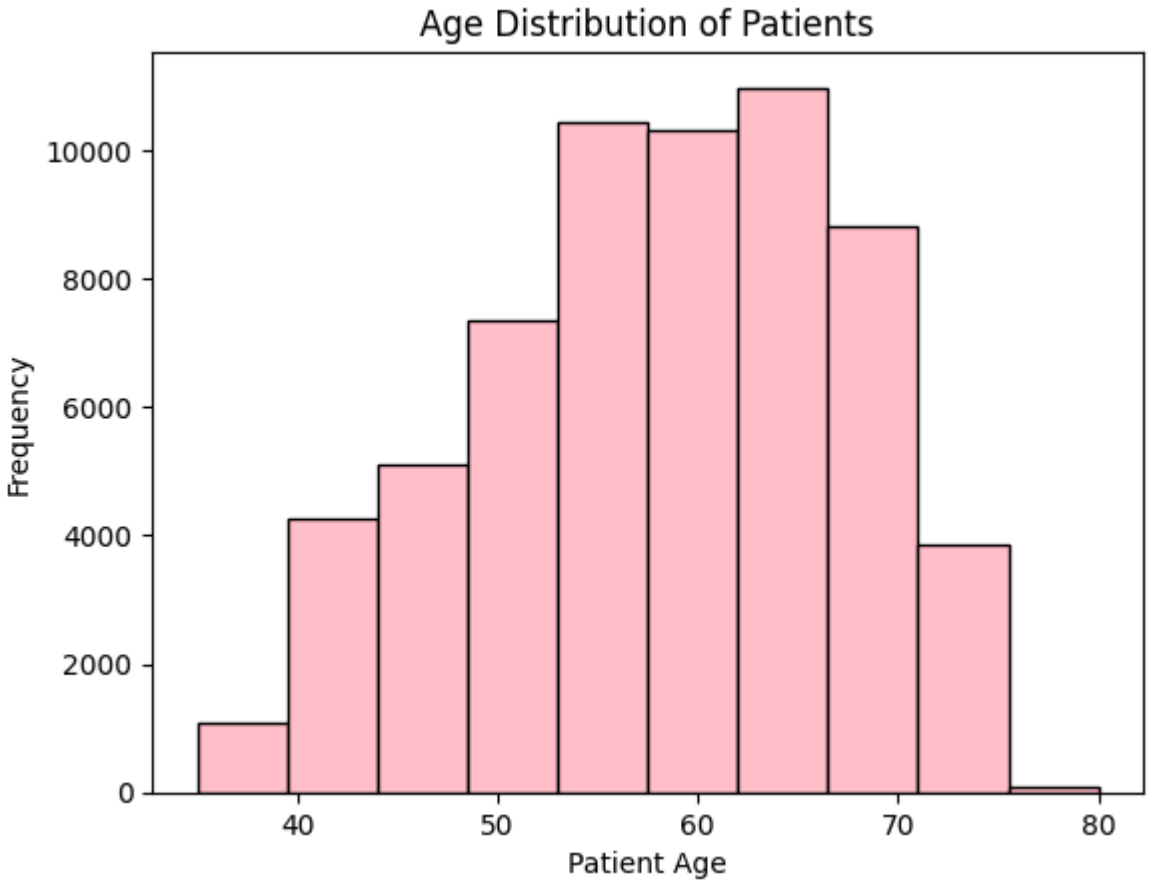
In [17]:
```
patients_gender_cnt = df_obesity_test.groupby(['diag', 'patient_gender'])['journ
total_cnt = patients_gender_cnt.groupby('diag')['count'].transform('sum')
patients_gender_cnt['gender_ratio'] = patients_gender_cnt['count']/total_cnt
patients_gender_cnt
```

Out[17]:

	diag	patient_gender	count	gender_ratio
0	Obesity	F	6422	0.587235
1	Obesity	M	4514	0.412765

EDA-4 Deep Dive in to the Age Demographics of the Patients

In [18]:
```
# Age distribution of patients
plt.hist(df_obesity_test['patient_age'], bins=10, color='pink', edgecolor='black

plt.xlabel('Patient Age')
plt.ylabel('Frequency')
plt.title('Age Distribution of Patients')

# Show the histogram
plt.show()
```

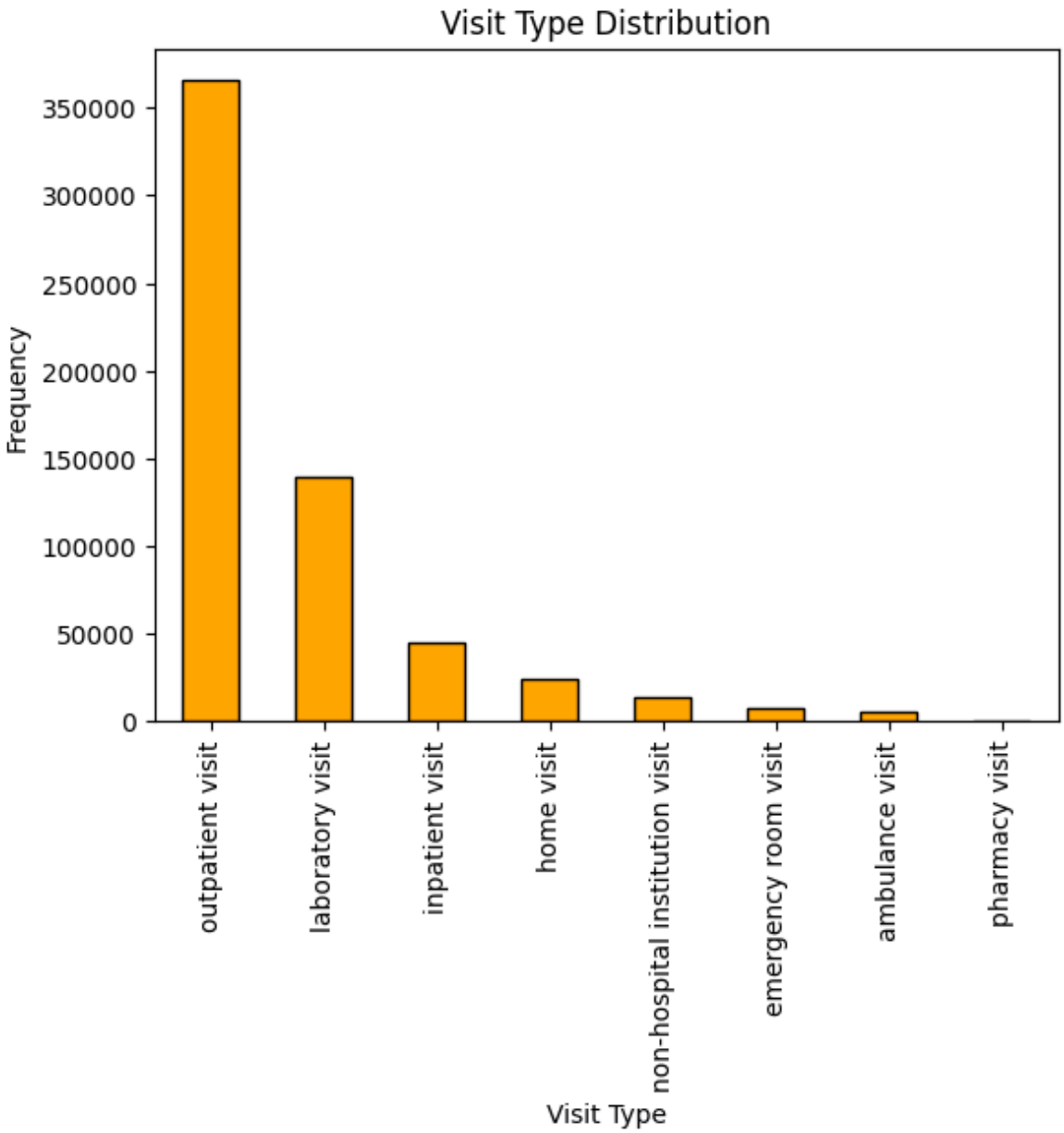


Insights:

- The most common age range seems to be between 55 and 65, which suggests that most people who are diagnosed as obsese pertian to this age bracket
- Histogram is roughly skewed towards older adults , and there are very few patients below the age of 40

EDA-5 Understanding the distribution of the Visit Type

In [19]:
```
#Visit Type Distribution

import matplotlib.pyplot as plt

# Count the occurrence of each visit type
visit_type_counts = df['visit_type'].value_counts()

# Create a bar chart
visit_type_counts.plot(kind='bar', color='orange', edgecolor='black')

plt.xlabel('Visit Type')
plt.ylabel('Frequency')
plt.title('Visit Type Distribution')

# Show the bar chart
plt.show()
```



Insights:

- "An outpatient visit refers to a medical service or treatment that does not require the patient to be admitted to a hospital for an overnight stay" This shows that the most

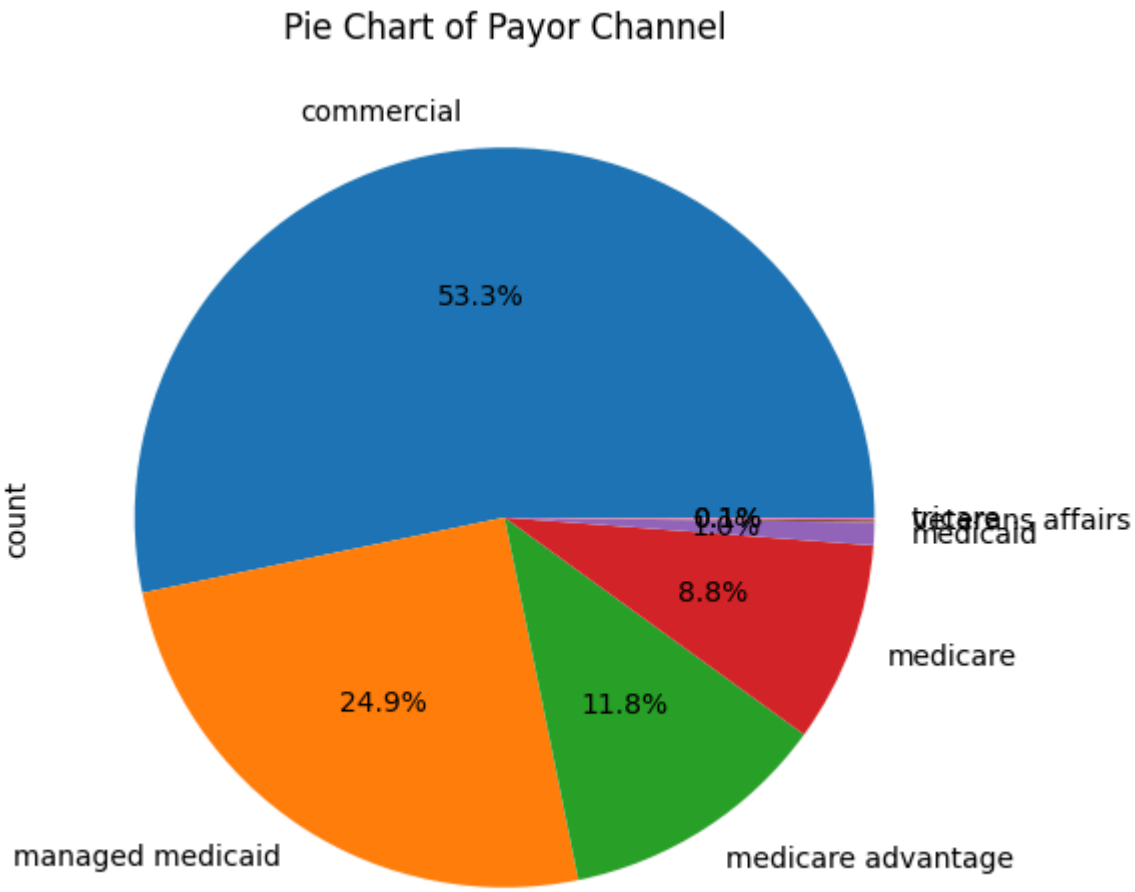
of the visit_type is outpatient_visit, based on the definition above it suggests that most of the patients do not stay overnight

- The second highest is laboratory visit where patients go for a diagnositic check or a health check up, which could mean that many patients are taking the initiate to get checked on a regular basis

- Lowest is ambulance_visit which could imply that the majority of the cases are low on severity and usually doesnt end up to be critical

EDA-6 Proportion by Payor Channel - Pie Chart

In [20]:
```
plt.figure(figsize=(12, 6))
df_obesity_test['payor_channel'].value_counts().plot(kind='pie', autopct='%1.1f%
plt.title('Pie Chart of Payor Channel')
plt.show()
```



Insights :

- A large portion is labeled as commercial about 53.3%.This suggests that over half of the patients in this dataset have commercial health insurance. This may imply that employee/commercial sponsorship might be common, where a larger organization sponsors for the insurance

- The orange and red segments represent managed medicaid at 24.9% and medicareat 11.8%. This indicate that a significant portion of the patient population is

covered by government-sponsored healthcare programs.
- Veterans affairs and tricare at 0.2%, indicating a very small proportion of patients in this dataset are receiving benefits through military or veteran health programs.

2. Data Preprocessing and Merging (15 pts)

In the previous module we have seen the basic information of df and df2, now I just want to check the number of missing values in both data frames

In [21]:
```
# Check the number of missing values in each column for df
print(df.isnull().sum())

# And for df2
print(df2.isnull().sum())
```

```
journey_id                0
episode_id                0
visit_id               1875
encounter_id              0
claim_date                0
patient_state             0
patient_short_zip        13
patient_age            6545
patient_gender          132
place_of_service     138970
visit_type            23071
payor                 14589
payor_channel         37439
ref_npi              326024
hcp_npi              188872
hcp_taxonomy         195316
hcp_specialty        195296
hco_npi                5736
hcp_npi_list         188872
hco_npi_list           5736
diag_list               870
diag_1                  870
diag_2               162930
diag_3               258241
diag_4               325292
diag_5               418509
rev_center_code      509599
rev_center_units       1967
proc_code             23913
proc_modifier        457577
proc_units             2194
line_charge               0
claim_charge              0
smart_allowed             0
dtype: int64
journey_id                          0
cohort_id                           0
claim_id                            0
patient_gender                      0
patient_state                       0
patient_zip                       122
patient_dob                         4
pharmacy_npi                    54665
pharmacist_npi                 873045
prescriber_npi                   3468
primary_care_npi               891054
group_id                       621002
date_of_service                     0
date_authorized                109947
transaction_type                    0
date_prescription_written           0
ndc11                               0
ndc11_original                 800420
number_of_refills_authorized    45104
diagnosis_code                 719918
diagnosis_code_type            782989
quantity_prescribed_original   572275
quantity_dispensed                  7
place_of_service               548426
is_service                     216300
```

```
level_of_service                676672
fill_number                          0
days_supply                          0
unit_of_measure                 300033
daw_code                        702479
prior_authorization_type_code   885707
is_compound_drug                110029
coupon_type                     892593
coupon_value_amount             894618
pharmacy_submitted_cost         109542
patient_pay                     292803
copay_coinsurance               483240
pcn                             118920
bin                             108383
plan_pay                        402745
reject_code_1                   707344
reject_code_3                   853371
reject_code_4                   887215
reject_code_5                   893099
ndc                                  0
active                               0
start_date                           0
end_date                        862175
dtype: int64
```

Mergeing Dataframes on journey_id and inner join

In [22]:
```
import pandas as pd

# Assuming df and df2 are your DataFrames and both contain a column named 'journ
merged_df = pd.merge(df, df2, on='journey_id', how='inner')

# Check the first few rows of the merged DataFrame
print(merged_df.head())
```

```
                          journey_id                         episode_id  \
0  00006390c96ebfffed580074c35a16a7  d5e15811af1d772f54af52f3560be0a5
1  00006390c96ebfffed580074c35a16a7  d5e15811af1d772f54af52f3560be0a5
2  00006390c96ebfffed580074c35a16a7  d5e15811af1d772f54af52f3560be0a5
3  00006390c96ebfffed580074c35a16a7  d5e15811af1d772f54af52f3560be0a5
4  00006390c96ebfffed580074c35a16a7  d5e15811af1d772f54af52f3560be0a5

                                       visit_id  \
0  f22b7ab885c44afae5449d74f8498f53aea1b2cfa79598...
1  f22b7ab885c44afae5449d74f8498f53aea1b2cfa79598...
2  f22b7ab885c44afae5449d74f8498f53aea1b2cfa79598...
3  f22b7ab885c44afae5449d74f8498f53aea1b2cfa79598...
4  f22b7ab885c44afae5449d74f8498f53aea1b2cfa79598...

                                     encounter_id  claim_date  \
0  d5a7dc61071d977cef9ff762973d360f4bcf15a19790d9...  2020-07-13
1  d5a7dc61071d977cef9ff762973d360f4bcf15a19790d9...  2020-07-13
2  d5a7dc61071d977cef9ff762973d360f4bcf15a19790d9...  2020-07-13
3  d5a7dc61071d977cef9ff762973d360f4bcf15a19790d9...  2020-07-13
4  d5a7dc61071d977cef9ff762973d360f4bcf15a19790d9...  2020-07-13

  patient_state_x  patient_short_zip  patient_age patient_gender_x  \
0              CA              922.0         55.0                M
1              CA              922.0         55.0                M
2              CA              922.0         55.0                M
3              CA              922.0         55.0                M
4              CA              922.0         55.0                M

  place_of_service_x  ...      bin plan_pay reject_code_1  reject_code_3  \
0               Home  ...  22659.0      NaN           NaN            NaN
1               Home  ...  22659.0      NaN           NaN            NaN
2               Home  ...  22659.0      NaN           NaN            NaN
3               Home  ...  22659.0      NaN           NaN            NaN
4               Home  ...  22659.0      NaN           NaN            NaN

   reject_code_4 reject_code_5          ndc  active start_date end_date
0            NaN           NaN  47335067381    True     190001      NaN
1            NaN           NaN  47335067381    True     190001      NaN
2            NaN           NaN  47335067381    True     190001      NaN
3            NaN           NaN  47335067381    True     190001      NaN
4            NaN           NaN  65862059805    True     190001      NaN

[5 rows x 81 columns]
```

In [23]:
```
print(merged_df.shape)
```

```
(29731051, 81)
```

Display the list of columns post merge

In [24]:
```
print(merged_df.columns.tolist())
```

```
['journey_id', 'episode_id', 'visit_id', 'encounter_id', 'claim_date', 'patient_s
tate_x', 'patient_short_zip', 'patient_age', 'patient_gender_x', 'place_of_servic
e_x', 'visit_type', 'payor', 'payor_channel', 'ref_npi', 'hcp_npi', 'hcp_taxonom
y', 'hcp_specialty', 'hco_npi', 'hcp_npi_list', 'hco_npi_list', 'diag_list', 'dia
g_1', 'diag_2', 'diag_3', 'diag_4', 'diag_5', 'rev_center_code', 'rev_center_unit
s', 'proc_code', 'proc_modifier', 'proc_units', 'line_charge', 'claim_charge', 's
mart_allowed', 'cohort_id', 'claim_id', 'patient_gender_y', 'patient_state_y', 'p
atient_zip', 'patient_dob', 'pharmacy_npi', 'pharmacist_npi', 'prescriber_npi',
'primary_care_npi', 'group_id', 'date_of_service', 'date_authorized', 'transactio
n_type', 'date_prescription_written', 'ndc11', 'ndc11_original', 'number_of_refil
ls_authorized', 'diagnosis_code', 'diagnosis_code_type', 'quantity_prescribed_ori
ginal', 'quantity_dispensed', 'place_of_service_y', 'is_service', 'level_of_servi
ce', 'fill_number', 'days_supply', 'unit_of_measure', 'daw_code', 'prior_authoriz
ation_type_code', 'is_compound_drug', 'coupon_type', 'coupon_value_amount', 'phar
macy_submitted_cost', 'patient_pay', 'copay_coinsurance', 'pcn', 'bin', 'plan_pa
y', 'reject_code_1', 'reject_code_3', 'reject_code_4', 'reject_code_5', 'ndc', 'a
ctive', 'start_date', 'end_date']
```

Out of these, I have taken only a subset of columns, as it leads to memory errors if all columns are considered

In [25]:
```
#Out of these columns just take a few subset columns of interest, from the merge

columns_taken_into_consideration = columns_of_interest = [
    'journey_id', 'claim_date', 'diag_1', 'diag_2', 'diag_3',
    'diag_4', 'diag_5', 'visit_type', 'patient_age', 'patient_gender_x',
    'prescriber_npi',  'quantity_dispensed', 'number_of_refills_authorized',
    'days_supply',  'date_of_service',
    'proc_code',
]

filtered_df = merged_df[columns_taken_into_consideration]

print(filtered_df)
```

```
                                journey_id  claim_date diag_1 diag_2 diag_3  \
0         00006390c96ebfffed580074c35a16a7  2020-07-13   I872    NaN    NaN
1         00006390c96ebfffed580074c35a16a7  2020-07-13   I872    NaN    NaN
2         00006390c96ebfffed580074c35a16a7  2020-07-13   I872    NaN    NaN
3         00006390c96ebfffed580074c35a16a7  2020-07-13   I872    NaN    NaN
4         00006390c96ebfffed580074c35a16a7  2020-07-13   I872    NaN    NaN
...                                    ...         ...    ...    ...    ...
29731046  1fff19f39322d5c25a60259f30d937a4  2023-08-07   B351    NaN    NaN
29731047  1fff19f39322d5c25a60259f30d937a4  2023-08-07   B351    NaN    NaN
29731048  1fff19f39322d5c25a60259f30d937a4  2023-08-07   B351    NaN    NaN
29731049  1fff19f39322d5c25a60259f30d937a4  2023-08-07   B351    NaN    NaN
29731050  1fff19f39322d5c25a60259f30d937a4  2023-08-07   B351    NaN    NaN

         diag_4 diag_5        visit_type  patient_age patient_gender_x  \
0           NaN    NaN        home visit         55.0                M
1           NaN    NaN        home visit         55.0                M
2           NaN    NaN        home visit         55.0                M
3           NaN    NaN        home visit         55.0                M
4           NaN    NaN        home visit         55.0                M
...         ...    ...               ...         ...              ...
29731046    NaN    NaN  laboratory visit         55.0                M
29731047    NaN    NaN  laboratory visit         55.0                M
29731048    NaN    NaN  laboratory visit         55.0                M
29731049    NaN    NaN  laboratory visit         55.0                M
29731050    NaN    NaN  laboratory visit         55.0                M

         prescriber_npi  quantity_dispensed  number_of_refills_authorized  \
0            1750479168                30.0                           3.0
1            1750479168                30.0                           3.0
2            1750479168                30.0                           3.0
3            1750479168                30.0                           3.0
4            1750479168                90.0                           0.0
...                 ...                 ...                           ...
29731046     1982176947                30.0                           3.0
29731047     1982176947                30.0                           3.0
29731048     1982176947                30.0                           3.0
29731049     1982176947                60.0                           3.0
29731050     1982176947                60.0                           3.0

          days_supply date_of_service proc_code
0                30.0      2022-04-15     A6549
1                30.0      2022-03-16     A6549
2                30.0      2022-05-17     A6549
3                30.0      2022-02-15     A6549
4                90.0      2022-02-14     A6549
...               ...             ...       ...
29731046         30.0      2023-06-18     87101
29731047         30.0      2023-03-13     87101
29731048         30.0      2023-05-18     87101
29731049         30.0      2023-02-10     87101
29731050         30.0      2023-03-13     87101

[29731051 rows x 16 columns]
```

In [26]:
```
print(filtered_df.shape)
```

```
(29731051, 16)
```

Segregating Values in Numberical and Categorical Coulmns for Imputing and checking missing values

In [27]:
```
# Count missing values in numerical columns
numerical_cols = filtered_df.select_dtypes(include=['int64', 'float64']).isnull(

# Count missing values in categorical columns
categorical_cols = filtered_df.select_dtypes(include=['object']).isnull().sum()

print("The numerical columns in the filtered dataset are" , numerical_cols)
print ("The categorical columns in the filtered dataset are" , categorical_cols)
```

```
The numerical columns in the filtered dataset are patient_age
262586
quantity_dispensed                  249
number_of_refills_authorized    1352038
days_supply                           0
dtype: int64
The categorical columns in the filtered dataset are journey_id                0
claim_date                 0
diag_1                 42226
diag_2               7625130
diag_3              12145982
diag_4              15393534
diag_5              19899997
visit_type           1249891
patient_gender_x        2907
prescriber_npi        106802
date_of_service            0
proc_code            1094279
dtype: int64
```

Dropping Missing Values

In [30]:
```
import numpy as np
import pandas as pd


# Remove  missing values
filtered_df_cleaned = filtered_df.dropna()
```

Counting Missing Values for Filtered Dataset to double check that missing values are dropped - The reason I choose this approach is that there are too many rows in the dataset and imputing all of them could lead to severly skewed results, and also by dropping values I can focus on the acuall data

In [ ]:
```
# Count missing values in numerical columns
missing_numerical_new = filtered_df_cleaned.select_dtypes(include=['int64', 'flo

# Count missing values in categorical columns
missing_categorical_new = filtered_df_cleaned.select_dtypes(include=['object']).

print("The numerical columns in the filtered dataset are" , missing_numerical_ne
print ("The categorical columns in the filtered dataset are" , missing_categoric
```

3. Causal Analysis Setup

Lets start with the definition of endogeneity - Endogeneity is when the X variables are correlated with the error term in the model. It is a factor that makes in hard to determine the pure cause and effect because there may be a hidden factor at play

Some Potential Endogenity issues for estimating treatment effect of Ozempic:

- Missing out on account for factors that affect the treatment, as we learnt in ML class that each and every confounding variable must be takening into consideration while estimating treatment effect if we miss to account for variables, it can result in inaccurate results - similar to the Cellphone and Murder case study discussed in class
- Selection Bias - Missing out on selecting a subset of the population for a study, there maybe a predefined bias in the sample of people who have taken Ozempic and who have not beyond what is observable

Strategy for Adressing Endogenity :

- Control for confounding variables, makes such each and every variables that affects the treatment is accounted for
- Propensity Score Matching - matching similar profile of people who take Ozempic and do not take Ozemptic to eliminate bias

4.Model Development

Stage-1 Using Lasso to select a subset of X variables

Lasso on Numerical Veraiables

In [53]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoCV
from sklearn.preprocessing import StandardScaler

df_numerical = filtered_df_cleaned[['patient_age', 'number_of_refills_authorized

# Using Proc code to determine treatemet
df_numerical['treatment'] = (df_numerical['proc_code'] == 'J3490').astype(int)
df_numerical = df_numerical.drop('proc_code', axis=1)  # Drop the original proc_

# Scale numerical variables
scaler = StandardScaler()
numerical_features = ['patient_age', 'number_of_refills_authorized', 'days_suppl
df_numerical[numerical_features] = scaler.fit_transform(df_numerical[numerical_f

# Split the dataset
```

```
X_num = df_numerical.drop('treatment', axis=1)
y_num = df_numerical['treatment']
X_train_num, X_test_num, y_train_num, y_test_num = train_test_split(X_num, y_num

# Lasso CV for numerical variables
lasso_cv_num = LassoCV(cv=5, random_state=42).fit(X_train_num, y_train_num)

# Identify important numerical features
print("Numerical features selected by LassoCV:")
for coef, feature in zip(lasso_cv_num.coef_, X_train_num.columns):
    if coef != 0:
        print(f"{feature}: {coef}")
```

```
Numerical features selected by LassoCV:
patient_age: -0.0010526708672630569
number_of_refills_authorized: 3.6732579943488286e-05
days_supply: -6.608513625688306e-05
quantity_dispensed: -8.85475318239684e-05
```

Insights for Numerical Variables :

-Patient Age: The coefficient for patient_age is negative, which suggests that as patient age increases, receiving treatment slightly decreases. The negative relationship is very small, indicating that age has a minor influence on the treatment decision.

-Number of Refills Authorized: The coefficient for number_of_refills_authorized is positive , indicating a slight increase in the probability of the treatment being prescribed with an increase in the number of refills

-Quantity Dispensed: The coefficient for quantity_dispensed is negative suggesting that a higher quantity of medication is associated with a slightly lower probability of receiving treatment.

Lasso on Categorical Variables

In [50]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoCV

#Using a sample due to Memory error
sampled_df = filtered_df_cleaned.sample(frac=0.1, random_state=42)  # Adjust fra

# Using Proc Code to determine treatment
sampled_df['treatment'] = (sampled_df['proc_code'] == 'J3490').astype(int)

# Categorical Variables
categorical_variables = ['journey_id', 'claim_date', 'diag_1', 'diag_2', 'diag_3

# Limiting
for col in categorical_variables:
    top_categories = sampled_df[col].value_counts().nlargest(20).index.tolist()
    category_mapping = {cat: i+1 for i, cat in enumerate(top_categories)}
    sampled_df[col] = sampled_df[col].map(category_mapping).fillna(0)

# LassoCV
X = sampled_df[categorical_variables]
y = sampled_df['treatment']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

lasso_cv = LassoCV(cv=5, random_state=42).fit(X_train, y_train)

# Categorical features from the sample
print("Categorical features selected by LassoCV from the sample:")
for i in range(len(lasso_cv.coef_)):
    if lasso_cv.coef_[i] != 0:
        print(X_train.columns[i] + ": " + str(lasso_cv.coef_[i]))
```

```
Categorical features selected by LassoCV from the sample:
journey_id: 1.5144818420916993e-05
claim_date: -0.000119349478475294
diag_1: -0.00017098923471139642
diag_2: -0.00015009704260416325
diag_3: 5.992292881420662e-05
diag_4: -0.00011210487468179148
diag_5: -2.6659362998336227e-05
visit_type: -0.001170914968689764
patient_gender_x: 0.0017145272904443643
prescriber_npi: 9.486060766700201e-05
date_of_service: -1.8286834156246342e-05
```

Insights :

- Visit Type : The negative coefficient implies that certain settings like outpatient or laboratory visits are linked to less health improvement, possibly due to these visits indicating routine checks or follow-ups without immediate treatment effects.
- Gender-Specific Outcomes: The positive coefficient points to gender differences in health outcomes, suggesting that healthcare interventions may need to be adapted based on gender to optimize effectiveness.
- Diagnostic Influence: Most diag_ variables having negative coefficients indicate the presence of conditions that challenge health improvement, with diag_3 being an exception, hinting at certain diagnoses that respond well to interventions, highlighting the necessity for diagnosis-specific care strategies.

Stage-2 Running LASSO with outcome - health improvement , treatment and X variables from the previous stage

In [56]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LassoCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

# One-hot encode selected categorical variables
categorical_vars = ['journey_id', 'claim_date', 'diag_1', 'diag_2', 'diag_3', 'd
X_categorical = pd.get_dummies(df_cleaned_copy[categorical_vars], drop_first=Tru

# Include numerical variables and treatment
X_numerical_and_treatment = df_cleaned_copy[['number_of_refills_authorized', 'da
scaler = StandardScaler()
X_numerical_and_treatment_scaled = scaler.fit_transform(X_numerical_and_treatmen
```

```
# Combine scaled numerical/treatment data with encoded categorical data
X_combined = pd.concat([pd.DataFrame(X_numerical_and_treatment_scaled, columns=[
                        X_categorical.reset_index(drop=True)], axis=1)

# Outcome - Health Improvement, which I have named as the change in effect due t
y = df_cleaned_copy['health_improvement']


X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.2

#LassoCV
lasso_cv = LassoCV(cv=5, random_state=42).fit(X_train, y_train)

# Model performance
y_pred = lasso_cv.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R^2 Score: {r2}")

# Coefficients to see the impact of each feature, including treatment, on health
print("Coefficients for each feature, including treatment:")
for coef, feature in zip(lasso_cv.coef_, X_train.columns):
    if coef != 0:
        print(f"{feature}: {coef}")
```

```
Mean Squared Error: 0.14785942517106415
R^2 Score: 0.011638353089595155
Coefficients for each feature, including treatment:
number_of_refills_authorized: -0.036759332433048346
days_supply: -0.018551360412333278
quantity_dispensed: 0.006128009558839102
treatment: -0.00015224549830800772
journey_id: -0.0007932462479701273
claim_date: 0.00045894966237880155
diag_1: 0.0011071471093506864
diag_2: 0.000190610742339016
diag_3: 0.0005181117794205426
diag_5: 0.0005900442114088223
visit_type: 0.0025755811852923877
patient_gender_x: -0.011581525507386344
prescriber_npi: -0.005245125277659626
date_of_service: -0.0019158817987380557
```

The logic behind determining the outcome of health improvement :

- The variable health_improvement is derived from observing changes in the number of refills . The underlying hypothesis is that a decrease in medication refills (or a negative change) might indicate an improvement in health, assuming that patients who are getting better require less medication over time.

- Binary Outcome, This outcome is represented as a binary variable, where a positive value (e.g., 1) indicates health improvement, and a negative value (e.g., 0) suggests no improvement. This simplification allows for a clear, direct assessment of treatment impact.

- Including the treatment variable directly as one of the predictors is standard in Double Lasso and allows for assessing its specific impact on health improvement, alongside other factors. The coefficient associated with the treatment variable in the LassoCV model directly indicates its influence on health improvementt.

5.Model Evaluation and Interpretation (15 pts)

- Mean Squared Error of 0.14: This value indicates the average squared difference between the observed actual outcomes and the outcomes predicted by the model. Overall, I beleive this MSE is good, indicating minimal MSE
- R^2 Score of 0.01 : R^2, is the ratio of explained varaiation to total varaiation, is quite low. An R^2 score close to 0 implies that the model does not explain much of the variability in health improvement, suggesting that the selected features, might not capture the full complexity of factors affecting the change in effect due to ozempic
- Treatment Effect -0.0001: The slight negative coefficient associated with the treatment variable suggests a minimal direct negative impact of Ozempic on immediate health improvement as defined.
- Diagnostic Codes and Prescriptions: Positive coefficients for diag_1, diag_2, diag_3, and diag_5 suggest that certain health conditions might see a slight improvement with respect to the context of this data.

Insights :

- The minimal direct impact of the treatment variable on health improvement suggests that Ozempic's effectiveness may be much more layered , maybe could be due to the fact that Ozempic alone will not suffice, maybe Ozempic with the addition of other activities and supplements may pose a bigger impact.
- The variation in health outcomes based on visit type, gender suggests that treatment effectiveness can vary widely based on patient characteristics and the nature of healthcare interactions, there is a need for tailored treatment approaches.
- The low e R^2 score suggests that all the confounding varaiables have not been taken into consideration, due to which there is a bias in the result and these results do not truly depict the correct treatment effect