Portfolio 2

January 17, 2026

HEALTHCARE PROVIDER FRAUD DETECTION ANALYSIS

https://www.kaggle.com/datasets/rohitrox/healthcare-provider-fraud-detection-analysis

Dataset Overview The dataset contains healthcare data related to beneficiaries, inpatient claims, and outpatient claims.

It is intended for fraud detection analysis.

The main files include:

Beneficiary Data – demographic info, chronic diseases, date of death, gender, etc.

Inpatient Data – hospital visits, admission dates, discharge dates, procedures, charges.

Outpatient Data – clinic visits, procedures, medications, charges.

[4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[5]:
```
df1 = pd.read_csv('D:\Hira\Project 2\Portfolio 2\Train_Beneficiarydata.csv')
```

[6]:
```
df1
```

[6]:
```
              BeneID         DOB  DOD  Gender  Race RenalDiseaseIndicator  \
0          BENE11001  1943-01-01  NaN       1     1                     0
1          BENE11002  1936-09-01  NaN       2     1                     0
2          BENE11003  1936-08-01  NaN       1     1                     0
3          BENE11004  1922-07-01  NaN       1     1                     0
4          BENE11005  1935-09-01  NaN       1     1                     0
…                …           …  …     …    …                   …
138551   BENE159194  1939-07-01  NaN       1     1                     0
138552   BENE159195  1938-12-01  NaN       2     1                     0
138553   BENE159196  1916-06-01  NaN       2     1                     0
138554   BENE159197  1930-01-01  NaN       1     1                     0
138555   BENE159198  1952-04-01  NaN       2     1                     0
```

```
        State  County  NoOfMonths_PartACov  NoOfMonths_PartBCov  …  \
0          39     230                   12                   12  …
1          39     280                   12                   12  …
2          52     590                   12                   12  …
3          39     270                   12                   12  …
4          24     680                   12                   12  …
…         …      …                    …                    …   …
138551     39     140                   12                   12  …
138552     49     530                   12                   12  …
138553      6     150                   12                   12  …
138554     16     560                   12                   12  …
138555     21      20                   12                   12  …

        ChronicCond_Depression  ChronicCond_Diabetes  \
0                            1                     1
1                            2                     2
2                            2                     2
3                            2                     1
4                            2                     1
…                          …                     …
138551                       2                     2
138552                       2                     1
138553                       1                     1
138554                       2                     2
138555                       1                     1

        ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
0                               1                         2
1                               2                         2
2                               1                         2
3                               1                         1
4                               2                         2
…                             …                         …
138551                          2                         2
138552                          2                         2
138553                          1                         2
138554                          1                         2
138555                          2                         2

        ChronicCond_rheumatoidarthritis  ChronicCond_stroke  \
0                                     1                   1
1                                     2                   2
2                                     2                   2
3                                     1                   2
4                                     2                   2
…                                   …                   …
138551                                2                   2
```

```
138552                              2                   2
138553                              2                   2
138554                              2                   2
138555                              1                   2

        IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  \
0                          36000                   3204
1                              0                      0
2                              0                      0
3                              0                      0
4                              0                      0
…                            …                    …
138551                         0                      0
138552                         0                      0
138553                      2000                   1068
138554                         0                      0
138555                         0                      0

        OPAnnualReimbursementAmt  OPAnnualDeductibleAmt
0                             60                     70
1                             30                     50
2                             90                     40
3                           1810                    760
4                           1790                   1200
…                            …                    …
138551                       430                    460
138552                       880                    100
138553                      3240                   1390
138554                      2650                     10
138555                      5470                   1870

[138556 rows x 25 columns]
```

[7]:
```
df1.shape
```

[7]: (138556, 25)

[8]:
```
df1.columns
```

[8]:
```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
```

```
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt'],
      dtype='object')
```

[9]:
```
df1.info
```

[9]:
```
<bound method DataFrame.info of              BeneID         DOB  DOD  Gender
Race RenalDiseaseIndicator  \
0         BENE11001  1943-01-01  NaN       1     1                     0
1         BENE11002  1936-09-01  NaN       2     1                     0
2         BENE11003  1936-08-01  NaN       1     1                     0
3         BENE11004  1922-07-01  NaN       1     1                     0
4         BENE11005  1935-09-01  NaN       1     1                     0
…              …          …  …     …   …                   …
138551  BENE159194  1939-07-01  NaN       1     1                     0
138552  BENE159195  1938-12-01  NaN       2     1                     0
138553  BENE159196  1916-06-01  NaN       2     1                     0
138554  BENE159197  1930-01-01  NaN       1     1                     0
138555  BENE159198  1952-04-01  NaN       2     1                     0

        State  County  NoOfMonths_PartACov  NoOfMonths_PartBCov  …  \
0          39     230                   12                   12  …
1          39     280                   12                   12  …
2          52     590                   12                   12  …
3          39     270                   12                   12  …
4          24     680                   12                   12  …
…           …       …                    …                   …  …
138551     39     140                   12                   12  …
138552     49     530                   12                   12  …
138553      6     150                   12                   12  …
138554     16     560                   12                   12  …
138555     21      20                   12                   12  …

        ChronicCond_Depression  ChronicCond_Diabetes  \
0                            1                     1
1                            2                     2
2                            2                     2
3                            2                     1
4                            2                     1
…                          …                     …
138551                       2                     2
138552                       2                     1
138553                       1                     1
138554                       2                     2
138555                       1                     1

        ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
```

```
0                          1                     2
1                          2                     2
2                          1                     2
3                          1                     1
4                          2                     2
…                        …                   …
138551                     2                     2
138552                     2                     2
138553                     1                     2
138554                     1                     2
138555                     2                     2

        ChronicCond_rheumatoidarthritis  ChronicCond_stroke  \
0                                     1                   1
1                                     2                   2
2                                     2                   2
3                                     1                   2
4                                     2                   2
…                                   …                 …
138551                                2                   2
138552                                2                   2
138553                                2                   2
138554                                2                   2
138555                                1                   2

        IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  \
0                          36000                   3204
1                              0                      0
2                              0                      0
3                              0                      0
4                              0                      0
…                            …                    …
138551                         0                      0
138552                         0                      0
138553                      2000                   1068
138554                         0                      0
138555                         0                      0

        OPAnnualReimbursementAmt  OPAnnualDeductibleAmt
0                             60                     70
1                             30                     50
2                             90                     40
3                           1810                    760
4                           1790                   1200
…                            …                    …
138551                       430                    460
138552                       880                    100
```

```
138553                          3240                1390
138554                          2650                  10
138555                          5470                1870

[138556 rows x 25 columns]>
```

[10]:
```
df1.describe()
```

[10]:
```
              Gender           Race          State         County  \
count  138556.000000  138556.000000  138556.000000  138556.000000
mean        1.570932       1.254511      25.666734     374.424745
std         0.494945       0.717007      15.223443     266.277581
min         1.000000       1.000000       1.000000       0.000000
25%         1.000000       1.000000      11.000000     141.000000
50%         2.000000       1.000000      25.000000     340.000000
75%         2.000000       1.000000      39.000000     570.000000
max         2.000000       5.000000      54.000000     999.000000

       NoOfMonths_PartACov  NoOfMonths_PartBCov  ChronicCond_Alzheimer  \
count        138556.000000        138556.000000          138556.000000
mean             11.907727            11.910145               1.667817
std               1.032332             0.936893               0.470998
min               0.000000             0.000000               1.000000
25%              12.000000            12.000000               1.000000
50%              12.000000            12.000000               2.000000
75%              12.000000            12.000000               2.000000
max              12.000000            12.000000               2.000000

       ChronicCond_Heartfailure  ChronicCond_KidneyDisease  \
count             138556.000000              138556.000000
mean                   1.506322                   1.687643
std                    0.499962                   0.463456
min                    1.000000                   1.000000
25%                    1.000000                   1.000000
50%                    2.000000                   2.000000
75%                    2.000000                   2.000000
max                    2.000000                   2.000000

       ChronicCond_Cancer  …  ChronicCond_Depression  ChronicCond_Diabetes  \
count       138556.000000  …           138556.000000         138556.000000
mean             1.880041  …                1.644476              1.398142
std              0.324914  …                0.478674              0.489517
min              1.000000  …                1.000000              1.000000
25%              2.000000  …                1.000000              1.000000
50%              2.000000  …                2.000000              1.000000
75%              2.000000  …                2.000000              2.000000
max              2.000000  …                2.000000              2.000000
```

```
       ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
count              138556.000000             138556.000000
mean                    1.324143                  1.725317
std                     0.468056                  0.446356
min                     1.000000                  1.000000
25%                     1.000000                  1.000000
50%                     1.000000                  2.000000
75%                     2.000000                  2.000000
max                     2.000000                  2.000000

       ChronicCond_rheumatoidarthritis  ChronicCond_stroke  \
count                    138556.000000       138556.000000
mean                          1.743180            1.920942
std                           0.436881            0.269831
min                           1.000000            1.000000
25%                           1.000000            2.000000
50%                           2.000000            2.000000
75%                           2.000000            2.000000
max                           2.000000            2.000000

       IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  \
count             138556.000000          138556.000000
mean                3660.346502             399.847296
std                 9568.621827             956.175202
min                -8000.000000               0.000000
25%                    0.000000               0.000000
50%                    0.000000               0.000000
75%                 2280.000000            1068.000000
max               161470.000000           38272.000000

       OPAnnualReimbursementAmt  OPAnnualDeductibleAmt
count             138556.000000          138556.000000
mean                1298.219348             377.718258
std                 2493.901134             645.530187
min                  -70.000000               0.000000
25%                  170.000000              40.000000
50%                  570.000000             170.000000
75%                 1500.000000             460.000000
max               102960.000000           13840.000000

[8 rows x 21 columns]
```

[11]:
```
print(df1.isnull().sum())
```

```
BeneID                                  0
DOB                                     0
DOD                                137135
```

```
Gender                                  0
Race                                    0
RenalDiseaseIndicator                   0
State                                   0
County                                  0
NoOfMonths_PartACov                     0
NoOfMonths_PartBCov                     0
ChronicCond_Alzheimer                   0
ChronicCond_Heartfailure                0
ChronicCond_KidneyDisease               0
ChronicCond_Cancer                      0
ChronicCond_ObstrPulmonary              0
ChronicCond_Depression                  0
ChronicCond_Diabetes                    0
ChronicCond_IschemicHeart               0
ChronicCond_Osteoporasis                0
ChronicCond_rheumatoidarthritis         0
ChronicCond_stroke                      0
IPAnnualReimbursementAmt                0
IPAnnualDeductibleAmt                   0
OPAnnualReimbursementAmt                0
OPAnnualDeductibleAmt                   0
dtype: int64
```

0.0.1 Description

The DOD column contains NaN values.

In healthcare datasets, NaN in DOD usually means the beneficiary is still alive. ##### we need a categorical column that clearly shows Alive or Dead instead of NaN.

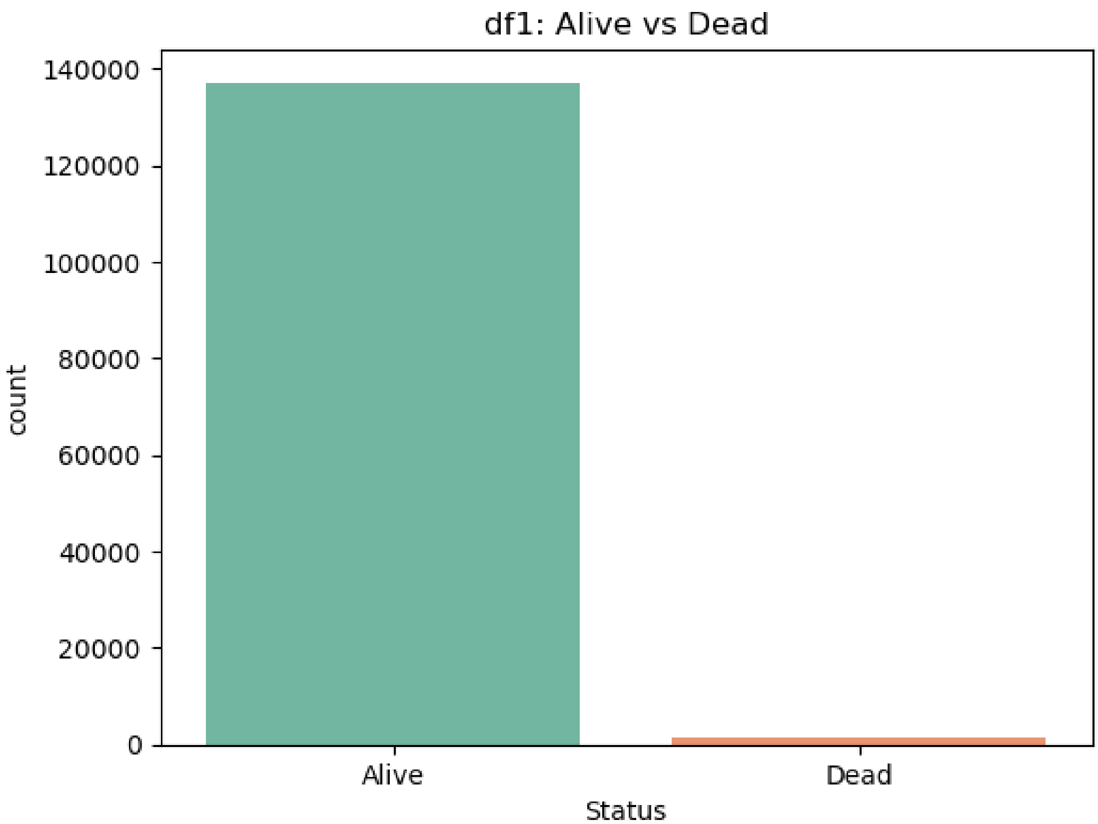
[12]:
```python
import pandas as pd

# Convert DOD to datetime
df1['DOD'] = pd.to_datetime(df1['DOD'], errors='coerce')

# Create Status column: Alive if NaN, Dead if date exists
df1['Status'] = df1['DOD'].apply(lambda x: 'Dead' if pd.notnull(x) else 'Alive')

# Check
print(df1['Status'].value_counts())
```

```
Status
Alive    137135
Dead       1421
Name: count, dtype: int64
```

Visualization Alive VS Dead

[13]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Status', data=df1, palette='Set2')
plt.title("df1: Alive vs Dead")
plt.show()
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\1571269568.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Status', data=df1, palette='Set2')
```



Handling Description alive

This step filters the dataset to identify alive beneficiaries. with missing values in the Date of Death (DOD) column are assumed to represent beneficiaries who are still alive.

These records are stored in a separate DataFrame for further analysis.

In this step, the dataset is filtered based on the Date of Death (DOD) column.

Beneficiaries with missing DOD values are classified as alive, as no death date is recorded.

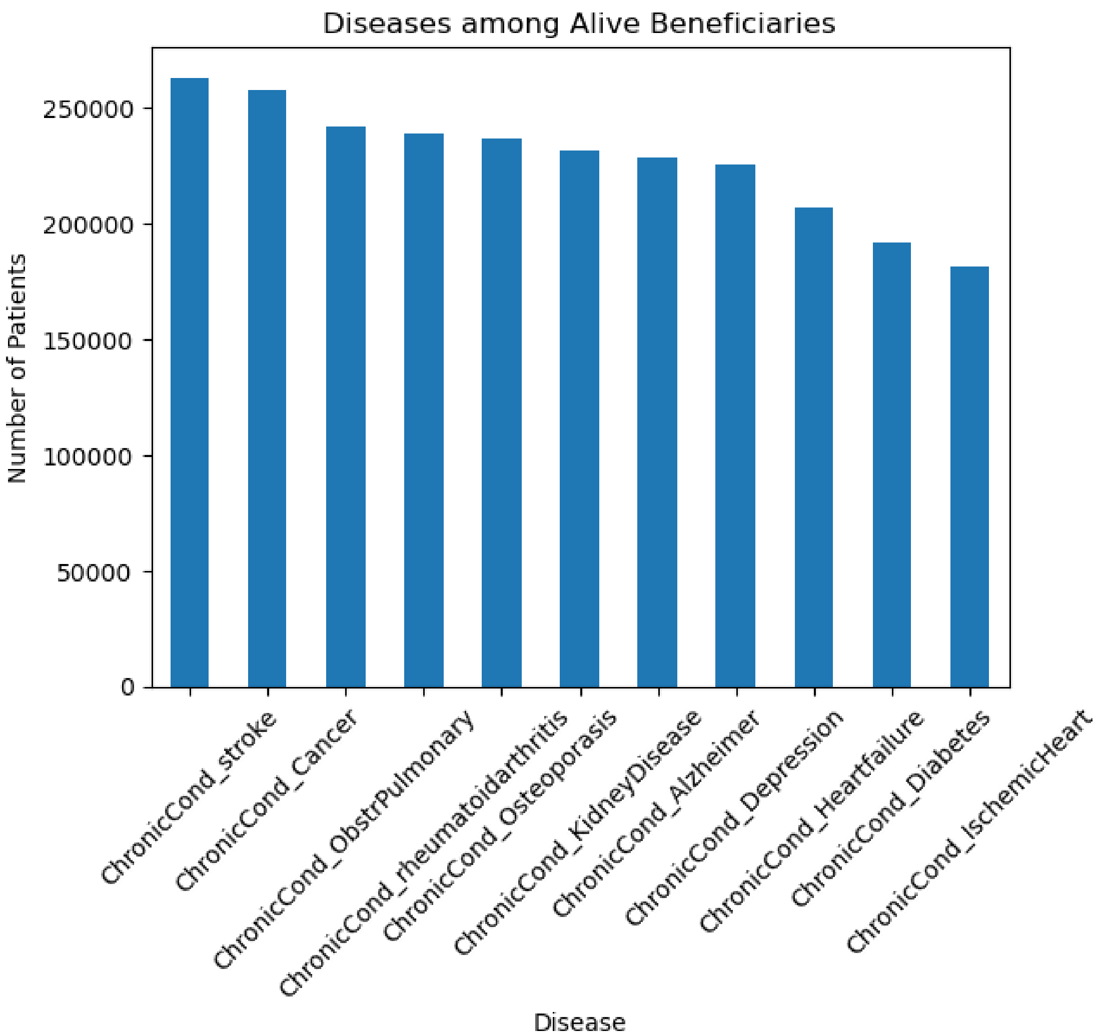
The filtered records are stored in a new DataFrame named alive_df for disease and utilization analysis. This code filters all beneficiaries with missing Date of Death values and classifies them as alive for further analysis

[12]:
```
alive_df = df1[df1['DOD'].isna()]
```

[13]:
```
alive_diseases = alive_df[
    ['ChronicCond_Alzheimer',
     'ChronicCond_Heartfailure',
     'ChronicCond_KidneyDisease',
     'ChronicCond_Cancer',
     'ChronicCond_ObstrPulmonary',
     'ChronicCond_Depression',
     'ChronicCond_Diabetes',
     'ChronicCond_IschemicHeart',
     'ChronicCond_Osteoporasis',
     'ChronicCond_rheumatoidarthritis',
     'ChronicCond_stroke']
].sum().sort_values(ascending=False)

print(alive_diseases)
```

```
ChronicCond_stroke                  263420
ChronicCond_Cancer                  257860
ChronicCond_ObstrPulmonary          241803
ChronicCond_rheumatoidarthritis     239087
ChronicCond_Osteoporasis            236613
ChronicCond_KidneyDisease           231469
ChronicCond_Alzheimer               228799
ChronicCond_Depression              225521
ChronicCond_Heartfailure            206640
ChronicCond_Diabetes                191816
ChronicCond_IschemicHeart           181676
dtype: int64
```

[14]:
```
import matplotlib.pyplot as plt

alive_diseases.plot(kind='bar')
plt.title("Diseases among Alive Beneficiaries")
plt.ylabel("Number of Patients")
plt.xlabel("Disease")
plt.xticks(rotation=45)
plt.show()
```



Handling of Deceased Beneficiaries This step identifies deceased beneficiaries in the dataset by filtering records with non-missing values in the Date of Death (DOD) column.

A non-null DOD indicates that the beneficiary has passed away.

These records are stored in a separate DataFrame (dead_df) to analyze disease prevalence and healthcare patterns among deceased patients.

This step filters beneficiaries with a recorded Date of Death (DOD) and classifies them as deceased for further analysis.

Beneficiaries with available Date of Death information are considered deceased.

The dataset is filtered accordingly to study disease distribution and mortality-related trends

[15]:
```
df1['DOD'].notna().sum()
```

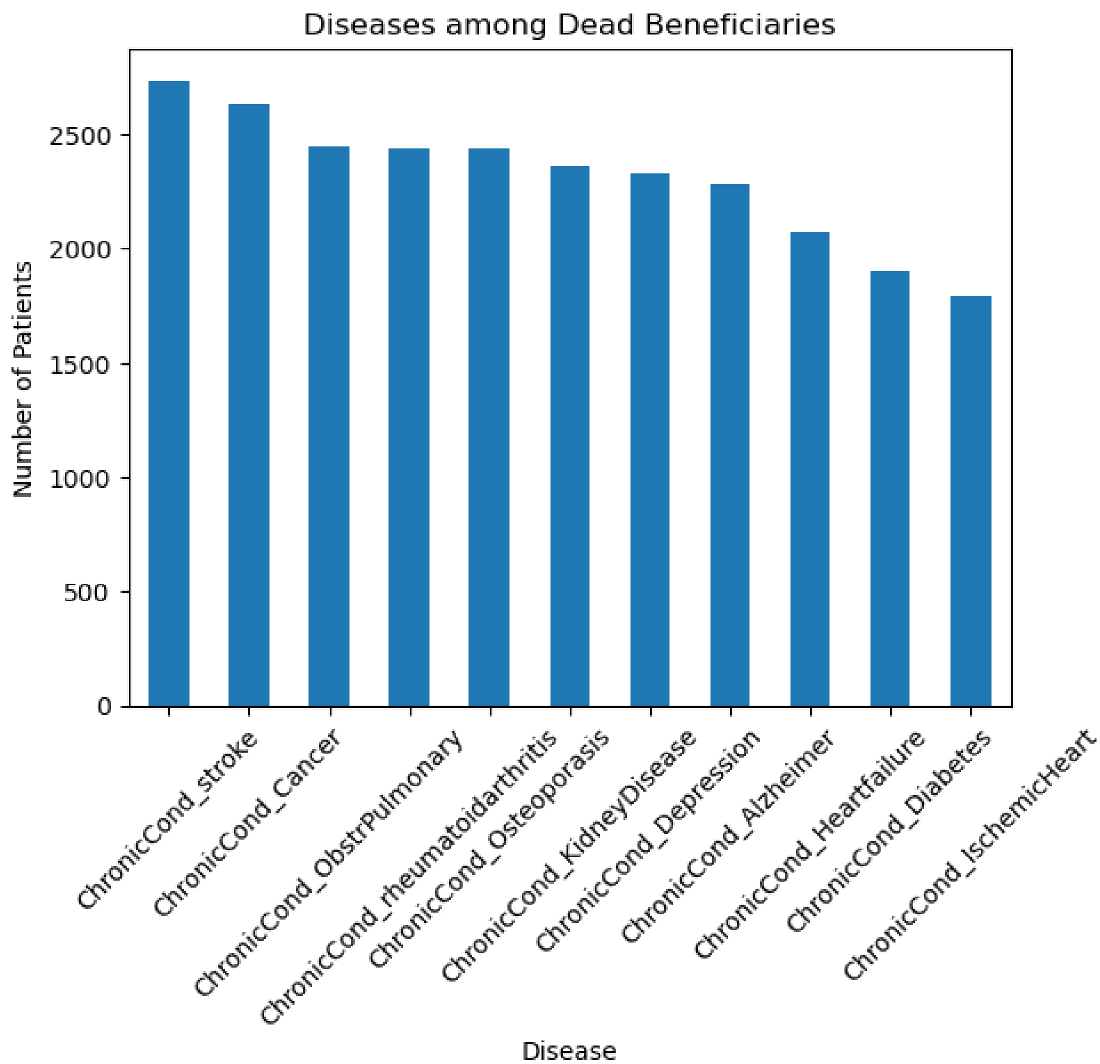
[15]: np.int64(1421)

[16]:
```python
dead_df1 = df1[df1['DOD'].notna()]
```

[17]:
```python
dead_diseases = dead_df1[
    ['ChronicCond_Alzheimer',
     'ChronicCond_Heartfailure',
     'ChronicCond_KidneyDisease',
     'ChronicCond_Cancer',
     'ChronicCond_ObstrPulmonary',
     'ChronicCond_Depression',
     'ChronicCond_Diabetes',
     'ChronicCond_IschemicHeart',
     'ChronicCond_Osteoporasis',
     'ChronicCond_rheumatoidarthritis',
     'ChronicCond_stroke']
].sum().sort_values(ascending=False)

print(dead_diseases)
```

```
ChronicCond_stroke                  2738
ChronicCond_Cancer                  2631
ChronicCond_ObstrPulmonary          2450
ChronicCond_rheumatoidarthritis     2441
ChronicCond_Osteoporasis            2440
ChronicCond_KidneyDisease           2364
ChronicCond_Depression              2331
ChronicCond_Alzheimer               2287
ChronicCond_Heartfailure            2070
ChronicCond_Diabetes                1905
ChronicCond_IschemicHeart           1792
dtype: int64
```

[18]:
```python
import matplotlib.pyplot as plt

dead_diseases.plot(kind='bar')
plt.title("Diseases among Dead Beneficiaries")
plt.ylabel("Number of Patients")
plt.xlabel("Disease")
plt.xticks(rotation=45)
plt.show()
```



Distribution of Alive and Deceased Beneficiaries The distribution shows that the majority of beneficiaries are classified as alive, as most records do not contain a Date of Death.

A smaller proportion of beneficiaries are marked as deceased, indicating fewer recorded death events in the dataset.

[19]:
```
alive_df = df1[df1['DOD'].isna()]
dead_df  = df1[df1['DOD'].notna()]
```

[20]:
```
disease_cols = [
    'ChronicCond_Alzheimer',
    'ChronicCond_Heartfailure',
    'ChronicCond_KidneyDisease',
    'ChronicCond_Cancer',
```

```
    'ChronicCond_ObstrPulmonary',
    'ChronicCond_Depression',
    'ChronicCond_Diabetes',
    'ChronicCond_IschemicHeart',
    'ChronicCond_Osteoporasis',
    'ChronicCond_rheumatoidarthritis',
    'ChronicCond_stroke'
]
```

[21]:
```
alive_counts = alive_df[disease_cols].sum()
dead_counts  = dead_df[disease_cols].sum()
```

Description: Alive vs Dead Chronic Disease Comparison compare_df compares the prevalence of chronic diseases between alive and deceased beneficiaries.

Each row represents a specific chronic condition

Columns show: Alive → Number of alive beneficiaries having that disease

Dead → Number of deceased beneficiaries having that disease

[22]:
```
compare_df = pd.DataFrame({
    'Alive': alive_counts,
    'Dead': dead_counts
})
```

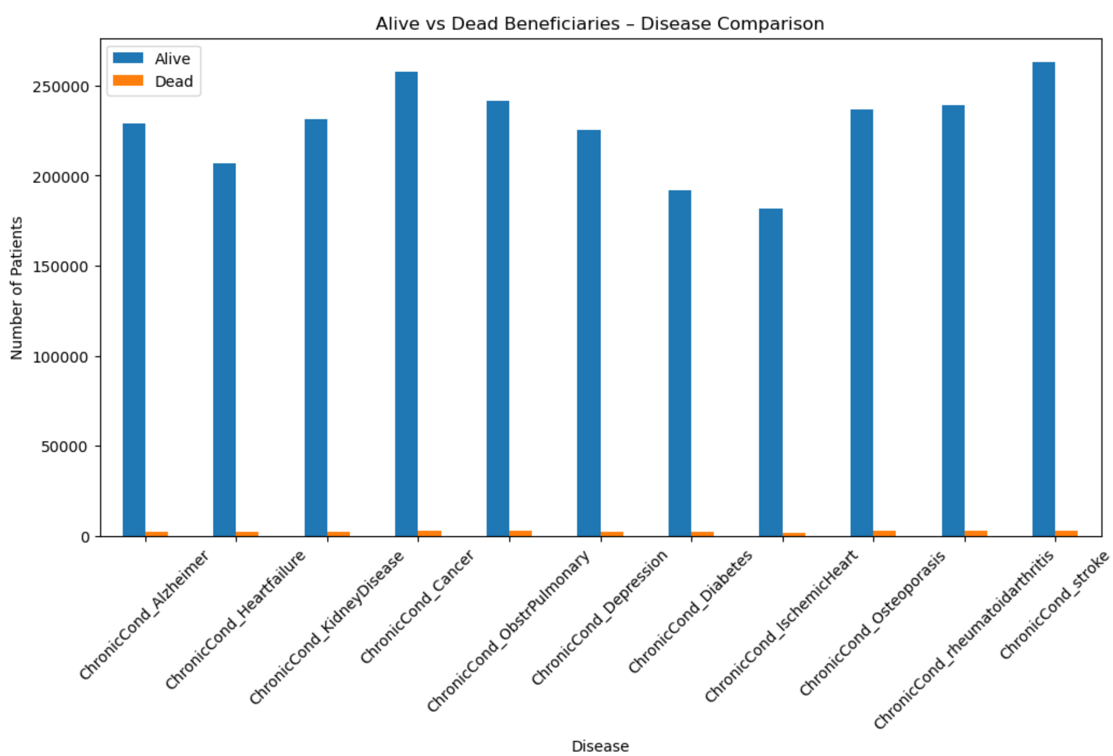
[23]:
```
alive_counts = alive_df[disease_cols].sum()
dead_counts  = dead_df[disease_cols].sum()

print("Alive counts:\n", alive_counts)
print("Dead counts:\n", dead_counts)
```

```
Alive counts:
 ChronicCond_Alzheimer              228799
ChronicCond_Heartfailure           206640
ChronicCond_KidneyDisease          231469
ChronicCond_Cancer                 257860
ChronicCond_ObstrPulmonary         241803
ChronicCond_Depression             225521
ChronicCond_Diabetes               191816
ChronicCond_IschemicHeart          181676
ChronicCond_Osteoporasis           236613
ChronicCond_rheumatoidarthritis    239087
ChronicCond_stroke                 263420
dtype: int64
Dead counts:
 ChronicCond_Alzheimer              2287
ChronicCond_Heartfailure           2070
ChronicCond_KidneyDisease          2364
```

```
ChronicCond_Cancer                  2631
ChronicCond_ObstrPulmonary          2450
ChronicCond_Depression              2331
ChronicCond_Diabetes                1905
ChronicCond_IschemicHeart           1792
ChronicCond_Osteoporasis            2440
ChronicCond_rheumatoidarthritis     2441
ChronicCond_stroke                  2738
dtype: int64
```

[24]:
```
compare_df = pd.DataFrame({
    'Alive': alive_counts,
    'Dead': dead_counts
})
```

[25]:
```
import matplotlib.pyplot as plt

compare_df.plot(kind='bar', figsize=(12,6))
plt.title("Alive vs Dead Beneficiaries - Disease Comparison")
plt.ylabel("Number of Patients")
plt.xlabel("Disease")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



[26]:
```python
df1.columns
```

[26]:
```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt', 'Status'],
      dtype='object')
```

Handling and Distribution of Date of Birth (DOB) ###### The DOB (Date of Birth) column is converted to a datetime format using pd.to_datetime(). ###### Any invalid or incorrectly formatted entries are coerced to NaT (Not a Time) to handle errors and missing data gracefully. ###### This standardization allows for accurate age calculation, chronological analysis, and demographic distribution of beneficiaries.

[27]:
```python
import pandas as pd

df1['DOB'] = pd.to_datetime(
    df1['DOB'], errors='coerce'
)
```

[28]:
```python
from datetime import datetime

df1['Age'] = (
    (pd.to_datetime('today') -df1['DOB']).dt.days // 365
)
```

[29]:
```python
df1['DOB'] = pd.to_datetime(
    df1['DOB'], errors='coerce'
)

print(df1['DOB'].head())
```

```
0   1943-01-01
1   1936-09-01
2   1936-08-01
3   1922-07-01
4   1935-09-01
Name: DOB, dtype: datetime64[ns]
```

[30]:
```python
df1.head
```

[30]:
```
<bound method NDFrame.head of             BeneID        DOB DOD  Gender  Race
RenalDiseaseIndicator  State  \
0        BENE11001 1943-01-01 NaT       1     1                     0     39
1        BENE11002 1936-09-01 NaT       2     1                     0     39
2        BENE11003 1936-08-01 NaT       1     1                     0     52
3        BENE11004 1922-07-01 NaT       1     1                     0     39
4        BENE11005 1935-09-01 NaT       1     1                     0     24
…             …          … ..      …    …                   …     …
138551  BENE159194 1939-07-01 NaT       1     1                     0     39
138552  BENE159195 1938-12-01 NaT       2     1                     0     49
138553  BENE159196 1916-06-01 NaT       2     1                     0      6
138554  BENE159197 1930-01-01 NaT       1     1                     0     16
138555  BENE159198 1952-04-01 NaT       2     1                     0     21

        County  NoOfMonths_PartACov  NoOfMonths_PartBCov  …  \
0          230                   12                   12  …
1          280                   12                   12  …
2          590                   12                   12  …
3          270                   12                   12  …
4          680                   12                   12  …
…         …                  …                  …  …
138551     140                   12                   12  …
138552     530                   12                   12  …
138553     150                   12                   12  …
138554     560                   12                   12  …
138555      20                   12                   12  …

        ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
0                               1                         2
1                               2                         2
2                               1                         2
3                               1                         1
4                               2                         2
…                            …                       …
138551                          2                         2
138552                          2                         2
138553                          1                         2
138554                          1                         2
138555                          2                         2

        ChronicCond_rheumatoidarthritis  ChronicCond_stroke  \
0                                     1                   1
1                                     2                   2
2                                     2                   2
3                                     1                   2
4                                     2                   2
…                                  …                 …
```

```
138551                               2                   2
138552                               2                   2
138553                               2                   2
138554                               2                   2
138555                               1                   2

        IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  \
0                          36000                   3204
1                              0                      0
2                              0                      0
3                              0                      0
4                              0                      0
…                            …                    …
138551                         0                      0
138552                         0                      0
138553                      2000                   1068
138554                         0                      0
138555                         0                      0

        OPAnnualReimbursementAmt  OPAnnualDeductibleAmt  Status  Age
0                             60                     70   Alive   83
1                             30                     50   Alive   89
2                             90                     40   Alive   89
3                           1810                    760   Alive  103
4                           1790                   1200   Alive   90
…                            …                    …     …    …
138551                       430                    460   Alive   86
138552                       880                    100   Alive   87
138553                      3240                   1390   Alive  109
138554                      2650                     10   Alive   96
138555                      5470                   1870   Alive   73

[138556 rows x 27 columns]>
```

[31]:

```python
print(df1['Age'].head())
```

```
0     83
1     89
2     89
3    103
4     90
Name: Age, dtype: int64
```

[32]:

```python
df1[['DOB', 'Age']].head()
```

[32]:

```
         DOB  Age
0 1943-01-01   83
1 1936-09-01   89
```

```
2 1936-08-01   89
3 1922-07-01  103
4 1935-09-01   90
```

Descriptive Gender ###### Both male and female beneficiaries are present in the dataset.

Gender helps understand utilization patterns, not fraud by itself.

Certain chronic diseases and claim behaviors may vary by gender.

[33]:
```
print(df1['Gender'].unique())
```

```
[1 2]
```

[34]:
```
print(df1['Gender'].value_counts())
```

```
Gender
2    79106
1    59450
Name: count, dtype: int64
```

[35]:
```
df1[['Gender']].head()
```

[35]:

	Gender
0	1
1	2
2	1
3	1
4	1

[36]:
```
print(df1['Gender'].value_counts())
```

```
Gender
2    79106
1    59450
Name: count, dtype: int64
```

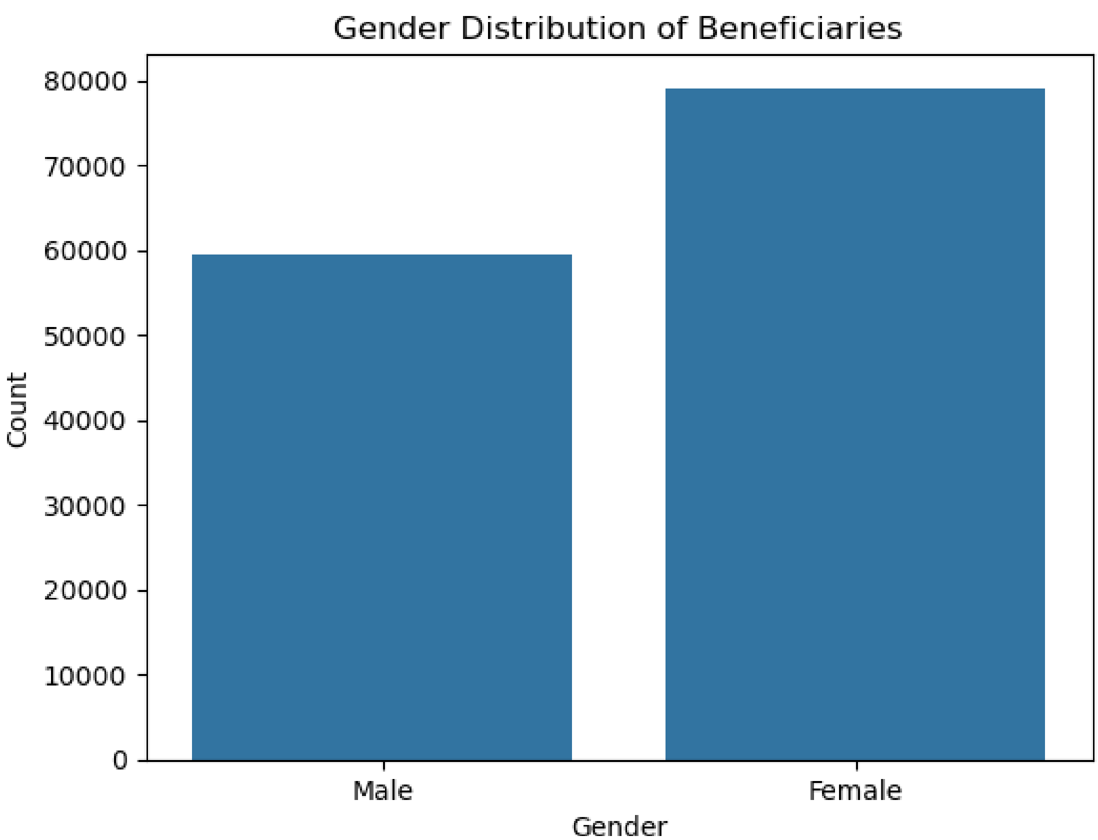
[37]:
```
# Example 1: 'M'/'F' to 'Male'/'Female'
df1['Gender'] = df1['Gender'].replace({'M':'Male', 'F':'Female'})

# Example 2: 1/2 to 'Male'/'Female'
df1['Gender'] = df1['Gender'].replace({1:'Male', 2:'Female'})
```

[38]:
```
print(df1['Gender'].value_counts())
print(df1[['Gender']].head())
```

```
Gender
Female    79106
Male      59450
Name: count, dtype: int64
   Gender
```

```
0      Male
1    Female
2      Male
3      Male
4      Male
```

[39]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Gender', data=df1)
plt.title("Gender Distribution of Beneficiaries")
plt.xlabel("Gender")
plt.ylabel("Count")
plt.show()
```



[40]:
```
df1.isna().sum()
```

[40]:
```
BeneID                             0
DOB                                0
DOD                           137135
```

```
Gender                             0
Race                               0
RenalDiseaseIndicator              0
State                              0
County                             0
NoOfMonths_PartACov                0
NoOfMonths_PartBCov                0
ChronicCond_Alzheimer              0
ChronicCond_Heartfailure           0
ChronicCond_KidneyDisease          0
ChronicCond_Cancer                 0
ChronicCond_ObstrPulmonary         0
ChronicCond_Depression             0
ChronicCond_Diabetes               0
ChronicCond_IschemicHeart          0
ChronicCond_Osteoporasis           0
ChronicCond_rheumatoidarthritis    0
ChronicCond_stroke                 0
IPAnnualReimbursementAmt           0
IPAnnualDeductibleAmt              0
OPAnnualReimbursementAmt           0
OPAnnualDeductibleAmt              0
Status                             0
Age                                0
dtype: int64
```

[41]:
```python
print(df1.columns)
```

```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt', 'Status', 'Age'],
      dtype='object')
```

[42]:
```python
df1.head()
```

[42]:

	BeneID	DOB	DOD	Gender	Race	RenalDiseaseIndicator	State	\
0	BENE11001	1943-01-01	NaT	Male	1	0	39	
1	BENE11002	1936-09-01	NaT	Female	1	0	39	
2	BENE11003	1936-08-01	NaT	Male	1	0	52	
3	BENE11004	1922-07-01	NaT	Male	1	0	39	
4	BENE11005	1935-09-01	NaT	Male	1	0	24	

```
   County  NoOfMonths_PartACov  NoOfMonths_PartBCov  …  \
0     230                   12                   12  …
1     280                   12                   12  …
2     590                   12                   12  …
3     270                   12                   12  …
4     680                   12                   12  …

   ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
0                          1                         2
1                          2                         2
2                          1                         2
3                          1                         1
4                          2                         2

   ChronicCond_rheumatoidarthritis  ChronicCond_stroke  \
0                                1                   1
1                                2                   2
2                                2                   2
3                                1                   2
4                                2                   2

   IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  OPAnnualReimbursementAmt  \
0                     36000                   3204                        60
1                         0                      0                        30
2                         0                      0                        90
3                         0                      0                      1810
4                         0                      0                      1790

   OPAnnualDeductibleAmt  Status  Age
0                     70   Alive   83
1                     50   Alive   89
2                     40   Alive   89
3                    760   Alive  103
4                   1200   Alive   90

[5 rows x 27 columns]
```

Handling Missing Values in Race Column The Race column contains missing values which are filled using forward fill (ffill) method.

This method replaces each missing value with the last valid observation above it in the dataset.

Forward filling ensures that there are no null values in the Race column, enabling accurate demographic analysis and visualization.

[43]:
```
df1['Race'] = df1['Race'].ffill()
```

[44]:
```
print(df1['Race'].isna().sum())
print(df1['Race'].value_counts())
```

```
0
Race
1    117057
2     13538
3      5059
5      2902
Name: count, dtype: int64
```

[45]:
```
race_map = {
    1: 'White',
    2: 'Black',
    3: 'Other',
    5: 'Unknown'
}

df1['Race'] = df1['Race'].map(race_map)
```

[46]:
```
print(df1[['Race']].head())  # Top 5 rows
```

```
    Race
0  White
1  White
2  White
3  White
4  White
```

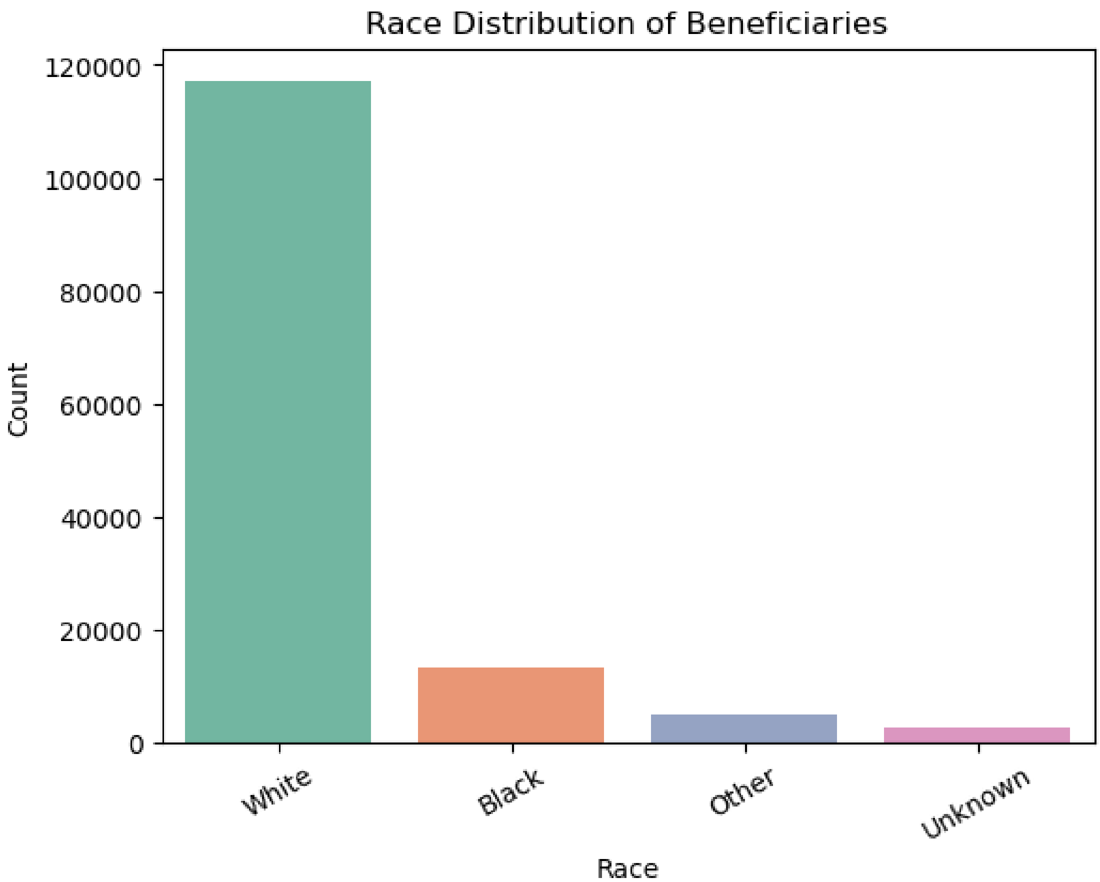
[47]:
```
print(df1['Race'].value_counts())
```

```
Race
White      117057
Black       13538
Other        5059
Unknown      2902
Name: count, dtype: int64
```

[48]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Race', data=df1, palette='Set2')
plt.title("Race Distribution of Beneficiaries")
plt.xlabel("Race")
plt.ylabel("Count")
plt.xticks(rotation=30)
plt.show()
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_6528\1672263856.py:4: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Race', data=df1, palette='Set2')
```



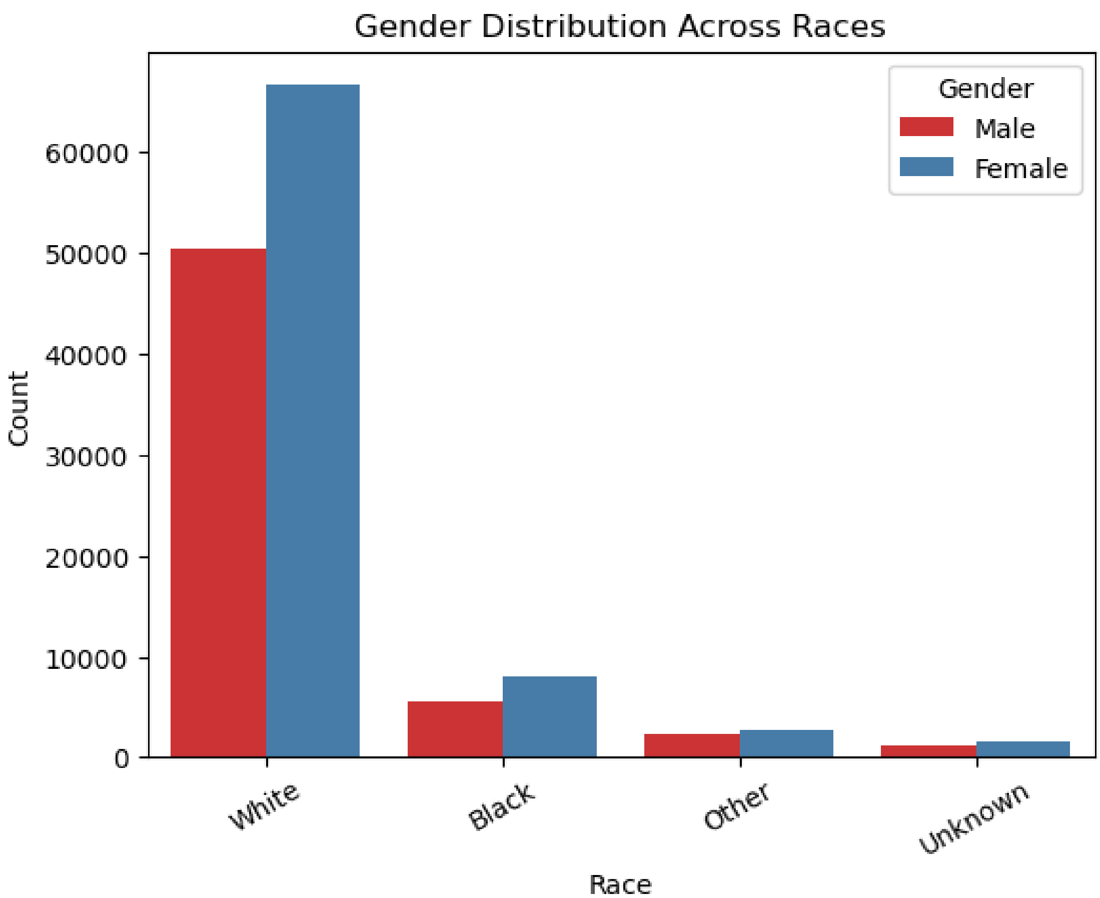
Count Plot (Race Distribution) A count plot is used to visualize the distribution of beneficiaries across different racial categories.

Before plotting, missing values in the Race column were handled using forward fill (ffill) to ensure completeness of data.

The count plot shows the number of beneficiaries in each race, providing insights into the demographic composition of the dataset.

This analysis helps identify if certain racial groups are overrepresented or underrepresented, which can be important for healthcare fraud detection and disease prevalence studies.

[49]:
```
sns.countplot(x='Race', hue='Gender', data=df1, palette='Set1')
plt.title("Gender Distribution Across Races")
plt.xlabel("Race")
plt.ylabel("Count")
plt.xticks(rotation=30)
plt.show()
```



[50]:
```
df1.columns
```

[50]:
```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
```

```
        'OPAnnualDeductibleAmt', 'Status', 'Age'],
       dtype='object')
```

[51]:
```
df1.head()
```

[51]:
```
      BeneID        DOB DOD  Gender   Race RenalDiseaseIndicator  State  \
0  BENE11001 1943-01-01 NaT    Male  White                     0     39
1  BENE11002 1936-09-01 NaT  Female  White                     0     39
2  BENE11003 1936-08-01 NaT    Male  White                     0     52
3  BENE11004 1922-07-01 NaT    Male  White                     0     39
4  BENE11005 1935-09-01 NaT    Male  White                     0     24

   County  NoOfMonths_PartACov  NoOfMonths_PartBCov  …  \
0     230                   12                   12  …
1     280                   12                   12  …
2     590                   12                   12  …
3     270                   12                   12  …
4     680                   12                   12  …

   ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
0                          1                         2
1                          2                         2
2                          1                         2
3                          1                         1
4                          2                         2

   ChronicCond_rheumatoidarthritis  ChronicCond_stroke  \
0                                1                   1
1                                2                   2
2                                2                   2
3                                1                   2
4                                2                   2

   IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  OPAnnualReimbursementAmt  \
0                     36000                   3204                        60
1                         0                      0                        30
2                         0                      0                        90
3                         0                      0                      1810
4                         0                      0                      1790

   OPAnnualDeductibleAmt  Status  Age
0                     70   Alive   83
1                     50   Alive   89
2                     40   Alive   89
3                    760   Alive  103
4                   1200   Alive   90
```

```
[5 rows x 27 columns]
```

Calculation and Distribution of Age A new Age column is derived from the DOB (Date of Birth) column by subtracting the year of birth from 2010.

This creates a numeric representation of each beneficiary's age, enabling demographic and statistical analysis.

Any invalid or missing DOB entries (NaT) are automatically excluded from this calculation.

[52]:
```
df1['DOB'] = pd.to_datetime(df1['DOB'], errors='coerce')
```

[53]:
```
df1['Age'] = 2010 - df1['DOB'].dt.year
```

[54]:
```
print(df1[['DOB','Age']].head())
```

```
         DOB  Age
0 1943-01-01   67
1 1936-09-01   74
2 1936-08-01   74
3 1922-07-01   88
4 1935-09-01   75
```

[56]:
```
df1 = df1[(df1['Age'] > 0) & (df1['Age'] < 120)]
```

Plot Description: Age Distribution Across Races

Age distributions vary across race groups Some race categories have higher median ages, while others show a wider age spread, indicating differences in healthcare utilization patterns.

Presence of outliers in certain races

Extreme age values (very old beneficiaries) appear in some race groups, which often correlates with: Higher medical complexity

Increased claim frequency and costs

Age is a strong cost driver, not a fraud indicator by itself

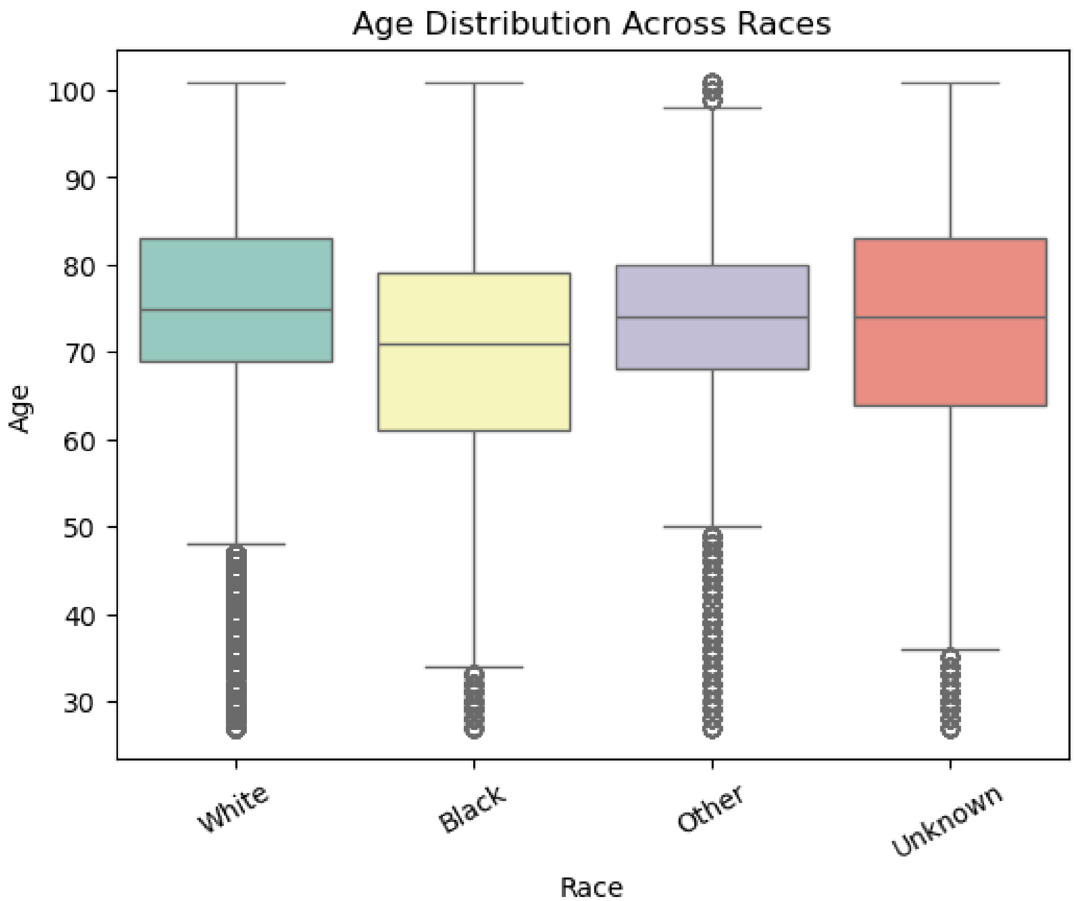
Older age groups typically require more medical services, which may naturally lead to higher claims.

[57]:
```
sns.boxplot(x='Race', y='Age', data=df1, palette='Set3')
plt.title("Age Distribution Across Races")
plt.xlabel("Race")
plt.ylabel("Age")
plt.xticks(rotation=30)
plt.show()
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_6528\3959494853.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
```

v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.boxplot(x='Race', y='Age', data=df1, palette='Set3')
```



[58]:
```
print(df1.columns)
```

```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt', 'Status', 'Age'],
      dtype='object')
```

```
[59]: df1.head
```

```
[59]: <bound method NDFrame.head of             BeneID        DOB DOD  Gender   Race
RenalDiseaseIndicator  State  \
0        BENE11001 1943-01-01 NaT    Male  White                     0     39
1        BENE11002 1936-09-01 NaT  Female  White                     0     39
2        BENE11003 1936-08-01 NaT    Male  White                     0     52
3        BENE11004 1922-07-01 NaT    Male  White                     0     39
4        BENE11005 1935-09-01 NaT    Male  White                     0     24
…             …          …  ..     …     …                  …    …
138551  BENE159194 1939-07-01 NaT    Male  White                     0     39
138552  BENE159195 1938-12-01 NaT  Female  White                     0     49
138553  BENE159196 1916-06-01 NaT  Female  White                     0      6
138554  BENE159197 1930-01-01 NaT    Male  White                     0     16
138555  BENE159198 1952-04-01 NaT  Female  White                     0     21

        County  NoOfMonths_PartACov  NoOfMonths_PartBCov  …  \
0          230                   12                   12  …
1          280                   12                   12  …
2          590                   12                   12  …
3          270                   12                   12  …
4          680                   12                   12  …
…        …                  …                  …  …
138551     140                   12                   12  …
138552     530                   12                   12  …
138553     150                   12                   12  …
138554     560                   12                   12  …
138555      20                   12                   12  …

        ChronicCond_IschemicHeart  ChronicCond_Osteoporasis  \
0                               1                         2
1                               2                         2
2                               1                         2
3                               1                         1
4                               2                         2
…                            …                       …
138551                          2                         2
138552                          2                         2
138553                          1                         2
138554                          1                         2
138555                          2                         2

        ChronicCond_rheumatoidarthritis  ChronicCond_stroke  \
0                                     1                   1
1                                     2                   2
2                                     2                   2
3                                     1                   2
```

```
4                                   2                 2
...                              ...               ...
138551                              2                 2
138552                              2                 2
138553                              2                 2
138554                              2                 2
138555                              1                 2

        IPAnnualReimbursementAmt  IPAnnualDeductibleAmt  \
0                          36000                   3204
1                              0                      0
2                              0                      0
3                              0                      0
4                              0                      0
...                          ...                    ...
138551                         0                      0
138552                         0                      0
138553                      2000                   1068
138554                         0                      0
138555                         0                      0

        OPAnnualReimbursementAmt  OPAnnualDeductibleAmt  Status  Age
0                             60                     70   Alive   67
1                             30                     50   Alive   74
2                             90                     40   Alive   74
3                           1810                    760   Alive   88
4                           1790                   1200   Alive   75
...                          ...                    ...     ...  ...
138551                       430                    460   Alive   71
138552                       880                    100   Alive   72
138553                      3240                   1390   Alive   94
138554                      2650                     10   Alive   80
138555                      5470                   1870   Alive   58

[138556 rows x 27 columns]>
```

[60]:
```python
print(df1.columns.tolist())
```

```
['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator', 'State',
'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression', 'ChronicCond_Diabetes',
'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporasis',
'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
'OPAnnualDeductibleAmt', 'Status', 'Age']
```

[61]:
```
chronic_cols = [col for col in df1.columns if 'ChronicCond' in col]
print(chronic_cols)
```

```
['ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
'ChronicCond_KidneyDisease', 'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
'ChronicCond_Depression', 'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
'ChronicCond_stroke']
```

[62]:
```
df1[chronic_cols].isna().sum()
```

[62]:
```
ChronicCond_Alzheimer              0
ChronicCond_Heartfailure           0
ChronicCond_KidneyDisease          0
ChronicCond_Cancer                 0
ChronicCond_ObstrPulmonary         0
ChronicCond_Depression             0
ChronicCond_Diabetes               0
ChronicCond_IschemicHeart          0
ChronicCond_Osteoporasis           0
ChronicCond_rheumatoidarthritis    0
ChronicCond_stroke                 0
dtype: int64
```

Unknown Values in Chronic Condition Columns This step sums up the values in all chronic condition columns to calculate the total number of patients affected by each disease.

Before plotting, missing or unknown values should be properly handled (e.g., replaced with 0 or cleaned) to ensure accuracy.

A bar plot is then created to visualize the prevalence of chronic diseases across the dataset

[63]:
```
for col in chronic_cols:
    print(col, ":", (df1[col] == '').sum(), "empty strings")
    print(col, ":", (df1[col] == 'Unknown').sum(), "'Unknown' values")
```

```
ChronicCond_Alzheimer : 0 empty strings
ChronicCond_Alzheimer : 0 'Unknown' values
ChronicCond_Heartfailure : 0 empty strings
ChronicCond_Heartfailure : 0 'Unknown' values
ChronicCond_KidneyDisease : 0 empty strings
ChronicCond_KidneyDisease : 0 'Unknown' values
ChronicCond_Cancer : 0 empty strings
ChronicCond_Cancer : 0 'Unknown' values
ChronicCond_ObstrPulmonary : 0 empty strings
ChronicCond_ObstrPulmonary : 0 'Unknown' values
ChronicCond_Depression : 0 empty strings
ChronicCond_Depression : 0 'Unknown' values
ChronicCond_Diabetes : 0 empty strings
```

```
ChronicCond_Diabetes : 0 'Unknown' values
ChronicCond_IschemicHeart : 0 empty strings
ChronicCond_IschemicHeart : 0 'Unknown' values
ChronicCond_Osteoporasis : 0 empty strings
ChronicCond_Osteoporasis : 0 'Unknown' values
ChronicCond_rheumatoidarthritis : 0 empty strings
ChronicCond_rheumatoidarthritis : 0 'Unknown' values
ChronicCond_stroke : 0 empty strings
ChronicCond_stroke : 0 'Unknown' values
```

Basic statistics for chronic disease columns desc_stats = df1[chronic_cols].describe()

print(desc_stats)

Plot Description: Number of Patients with Chronic Diseases

Most common chronic diseases are identified

Diseases with the tallest bars are more prevalent among beneficiaries.

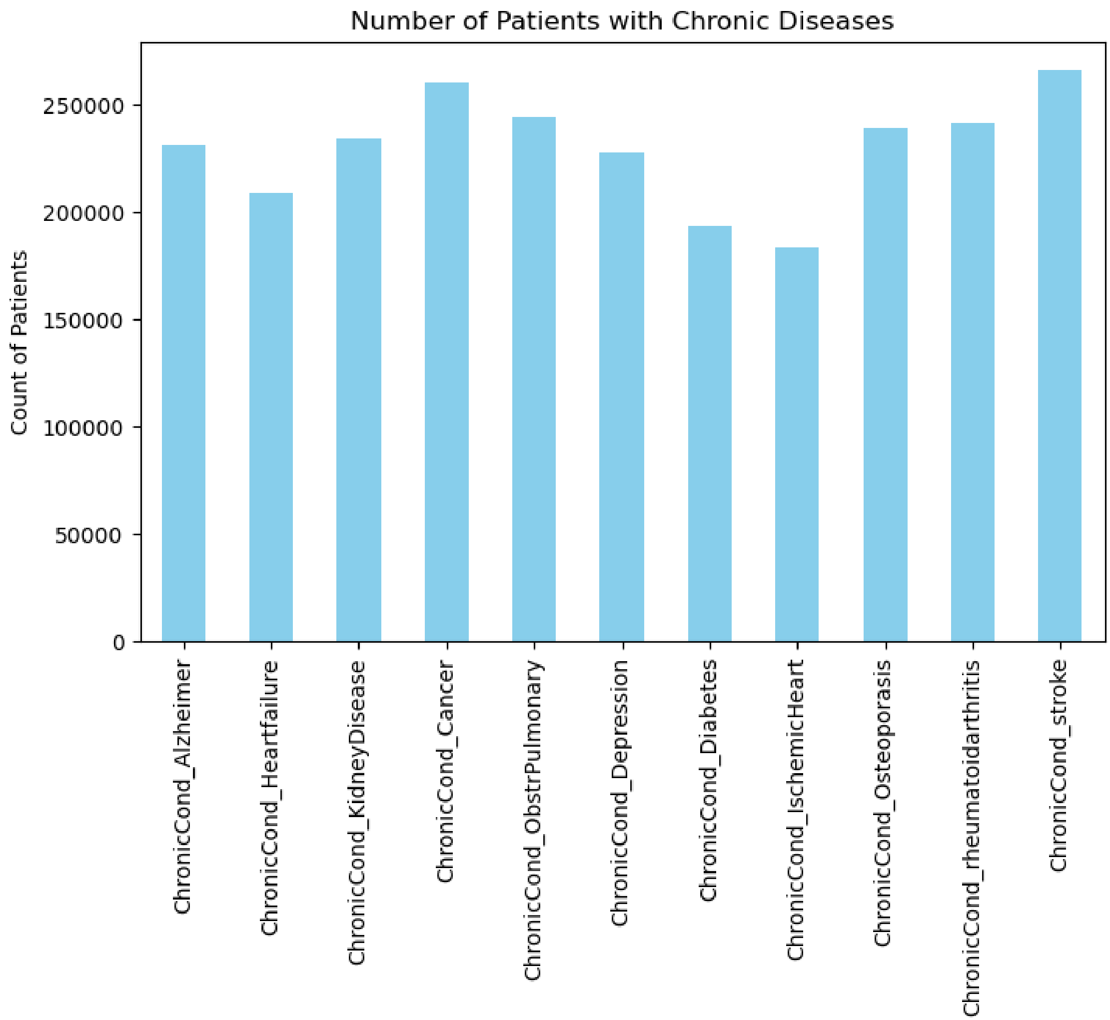
Example: Diabetes or Heart Failure may affect a large portion of the population.

High-prevalence diseases indicate higher healthcare costs

Chronic diseases often require frequent hospital visits, medications, and treatments.

These patients generate higher claim amounts, which could impact insurer finances.

[64]:
```
import matplotlib.pyplot as plt

df1[chronic_cols].sum().plot(kind='bar', color='skyblue', figsize=(8,5))
plt.title("Number of Patients with Chronic Diseases")
plt.ylabel("Count of Patients")
plt.show()
```



Plot Description: Correlation Between Chronic Diseases Certain chronic diseases co-occur frequently

Positive correlations indicate that patients with one condition are more likely to have another.

Example: Diabetes and Heart Failure often show moderate to strong correlation.

Multimorbidity is common Many patients have multiple chronic conditions, which leads to:

Higher total claims

Longer hospital stays

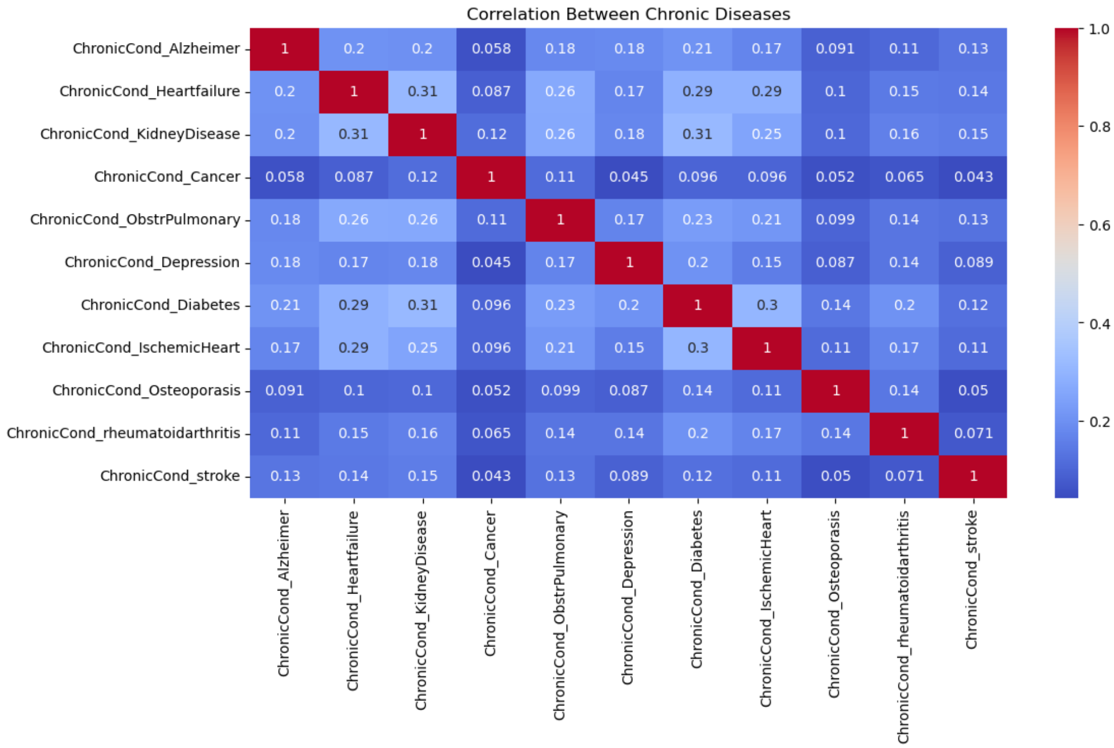
Increased complexity of care

Implications for fraud detection

Providers serving multimorbid patients may naturally have higher claim amounts.

Fraud monitoring should account for co-occurring conditions to avoid false positives.

[65]:
```python
import seaborn as sns

plt.figure(figsize=(12,6))
sns.heatmap(df1[chronic_cols].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Between Chronic Diseases")
plt.show()
```



Description: Patient Count per Chronic Disease

[66]:
```python
disease_counts = df1[chronic_cols].sum().reset_index()
disease_counts.columns = ['Chronic Disease', 'Patient Count']
print(disease_counts)
```

```
                Chronic Disease  Patient Count
0         ChronicCond_Alzheimer         231086
1      ChronicCond_Heartfailure         208710
2     ChronicCond_KidneyDisease         233833
3            ChronicCond_Cancer         260491
4    ChronicCond_ObstrPulmonary         244253
5        ChronicCond_Depression         227852
6          ChronicCond_Diabetes         193721
7     ChronicCond_IschemicHeart         183468
```

```
8          ChronicCond_Osteoporasis        239053
9   ChronicCond_rheumatoidarthritis        241528
10                ChronicCond_stroke        266158
```

[67]:
```
df1 = df1.drop_duplicates()
print(df1.shape)
```

```
(138556, 27)
```

[68]:
```
# Percentage of patients with each disease
disease_percent = (df1[chronic_cols].mean() * 100).round(2).reset_index()
disease_percent.columns = ['Chronic Disease', 'Patient %']
print(disease_percent)
```

```
                    Chronic Disease  Patient %
0             ChronicCond_Alzheimer     166.78
1          ChronicCond_Heartfailure     150.63
2         ChronicCond_KidneyDisease     168.76
3                ChronicCond_Cancer     188.00
4        ChronicCond_ObstrPulmonary     176.28
5            ChronicCond_Depression     164.45
6              ChronicCond_Diabetes     139.81
7          ChronicCond_IschemicHeart     132.41
8          ChronicCond_Osteoporasis     172.53
9   ChronicCond_rheumatoidarthritis     174.32
10                ChronicCond_stroke     192.09
```

Plot DescChronic disease prevalence differs between Alive and Dead groups Some diseases are more common among deceased beneficiaries.

Example: Heart Failure, Cancer, or Stroke may show higher percentages in the Dead group.

High-risk conditions for mortality Diseases with large differences between Dead vs Alive indicate conditions that contribute to higher mortality.

Impact on claim amounts Deceased patients with chronic diseases often generate higher and more complex claims, potentially increasing fraud risk if claims are abnormal.

Fraud monitoring / operational insight Providers treating patients with high percentages of serious chronic conditions should be closely monitored for:

Duplicate claims

Inflated billing

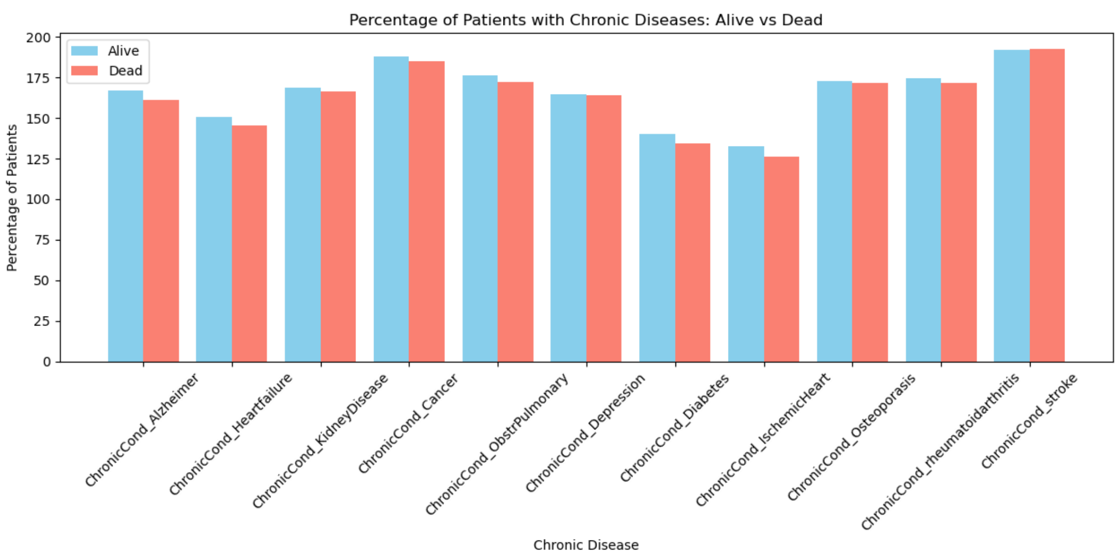
Post-mortem claims

[70]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

```python
# List of chronic disease columns
chronic_cols = [
    'ChronicCond_Alzheimer',
    'ChronicCond_Heartfailure',
    'ChronicCond_KidneyDisease',
    'ChronicCond_Cancer',
    'ChronicCond_ObstrPulmonary',
    'ChronicCond_Depression',
    'ChronicCond_Diabetes',
    'ChronicCond_IschemicHeart',
    'ChronicCond_Osteoporasis',
    'ChronicCond_rheumatoidarthritis',
    'ChronicCond_stroke'
]

# Split dataset into Alive and Dead
alive_df = df1[df1['DOD'].isna()]
dead_df  = df1[df1['DOD'].notna()]

# Count patients with each disease for Alive and Dead
alive_counts = alive_df[chronic_cols].sum() / len(alive_df) * 100
dead_counts  = dead_df[chronic_cols].sum() / len(dead_df) * 100

# Combine into a DataFrame
disease_percentage = pd.DataFrame({
    'Chronic Disease': chronic_cols,
    'Alive (%)': alive_counts.values,
    'Dead (%)': dead_counts.values
})

print(disease_percentage)

# Plotting percentage comparison
plt.figure(figsize=(12,6))
bar_width = 0.4
index = range(len(chronic_cols))

plt.bar(index, disease_percentage['Alive (%)'], bar_width, label='Alive', color='skyblue')
plt.bar([i + bar_width for i in index], disease_percentage['Dead (%)'], bar_width, label='Dead', color='salmon')

plt.xticks([i + bar_width/2 for i in index], disease_percentage['Chronic Disease'], rotation=45)
plt.xlabel('Chronic Disease')
plt.ylabel('Percentage of Patients')
```

```
plt.title('Percentage of Patients with Chronic Diseases: Alive vs Dead')
plt.legend()
plt.tight_layout()
plt.show()
```

```
                    Chronic Disease   Alive (%)    Dead (%)
0             ChronicCond_Alzheimer  166.842163  160.942998
1          ChronicCond_Heartfailure  150.683633  145.672062
2         ChronicCond_KidneyDisease  168.789149  166.361717
3                ChronicCond_Cancer  188.033689  185.151302
4        ChronicCond_ObstrPulmonary  176.324789  172.413793
5            ChronicCond_Depression  164.451818  164.039409
6              ChronicCond_Diabetes  139.873847  134.060521
7          ChronicCond_IschemicHeart  132.479673  126.108374
8          ChronicCond_Osteoporasis  172.540198  171.710063
9   ChronicCond_rheumatoidarthritis  174.344259  171.780436
10               ChronicCond_stroke  192.088088  192.681210
```



Data Handling & Descriptive Overview – Inpatient & Outpatient Data

[14]:
```
import pandas as pd

# Load inpatient and outpatient data
df2 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Inpatientdata.csv')
df3 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Outpatientdata.csv')

# Quick check
print(df2.head())
print(df2.info())
```

```python
print(df3.head())
print(df3.info())
```

```
      BeneID   ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12  2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31  2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17  2009-09-20  PRV56046
3  BENE11011  CLM38412   2009-02-14  2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13  2009-08-30  PRV56614

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                   26000          PHY390922                NaN
1                    5000          PHY318495          PHY318495
2                    5000          PHY372395                NaN
3                    5000          PHY369659          PHY392961
4                   10000          PHY379376          PHY398258

  OtherPhysician AdmissionDt  … ClmDiagnosisCode_7  ClmDiagnosisCode_8  \
0            NaN  2009-04-12  …               2724               19889
1            NaN  2009-08-31  …                NaN                 NaN
2      PHY324689  2009-09-17  …                NaN                 NaN
3      PHY349768  2009-02-14  …              25062               40390
4            NaN  2009-08-13  …               5119               29620

  ClmDiagnosisCode_9 ClmDiagnosisCode_10 ClmProcedureCode_1  \
0               5849                 NaN                NaN
1                NaN                 NaN             7092.0
2                NaN                 NaN                NaN
3               4019                 NaN              331.0
4              20300                 NaN             3893.0

  ClmProcedureCode_2 ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

  ClmProcedureCode_6
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN

[5 rows x 30 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40474 entries, 0 to 40473
```

```
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   BeneID                  40474 non-null  object
 1   ClaimID                 40474 non-null  object
 2   ClaimStartDt            40474 non-null  object
 3   ClaimEndDt              40474 non-null  object
 4   Provider                40474 non-null  object
 5   InscClaimAmtReimbursed  40474 non-null  int64
 6   AttendingPhysician      40362 non-null  object
 7   OperatingPhysician      23830 non-null  object
 8   OtherPhysician          4690 non-null   object
 9   AdmissionDt             40474 non-null  object
 10  ClmAdmitDiagnosisCode   40474 non-null  object
 11  DeductibleAmtPaid       39575 non-null  float64
 12  DischargeDt             40474 non-null  object
 13  DiagnosisGroupCode      40474 non-null  object
 14  ClmDiagnosisCode_1      40474 non-null  object
 15  ClmDiagnosisCode_2      40248 non-null  object
 16  ClmDiagnosisCode_3      39798 non-null  object
 17  ClmDiagnosisCode_4      38940 non-null  object
 18  ClmDiagnosisCode_5      37580 non-null  object
 19  ClmDiagnosisCode_6      35636 non-null  object
 20  ClmDiagnosisCode_7      33216 non-null  object
 21  ClmDiagnosisCode_8      30532 non-null  object
 22  ClmDiagnosisCode_9      26977 non-null  object
 23  ClmDiagnosisCode_10     3927 non-null   object
 24  ClmProcedureCode_1      23148 non-null  float64
 25  ClmProcedureCode_2      5454 non-null   float64
 26  ClmProcedureCode_3      965 non-null    float64
 27  ClmProcedureCode_4      116 non-null    float64
 28  ClmProcedureCode_5      9 non-null      float64
 29  ClmProcedureCode_6      0 non-null      float64
dtypes: float64(7), int64(1), object(22)
memory usage: 9.3+ MB
None
      BeneID    ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11002  CLM624349   2009-10-11  2009-10-11  PRV56011
1  BENE11003  CLM189947   2009-02-12  2009-02-12  PRV57610
2  BENE11003  CLM438021   2009-06-27  2009-06-27  PRV57595
3  BENE11004  CLM121801   2009-01-06  2009-01-06  PRV56011
4  BENE11004  CLM150998   2009-01-22  2009-01-22  PRV56011

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                      30          PHY326117                NaN
1                      80          PHY362868                NaN
2                      10          PHY328821                NaN
3                      40          PHY334319                NaN
```

```
4                    200         PHY403831               NaN

  OtherPhysician ClmDiagnosisCode_1  … ClmDiagnosisCode_9  \
0            NaN              78943  …                NaN
1            NaN               6115  …                NaN
2            NaN               2723  …                NaN
3            NaN              71988  …                NaN
4            NaN              82382  …                NaN

  ClmDiagnosisCode_10 ClmProcedureCode_1 ClmProcedureCode_2  \
0                 NaN                NaN                NaN
1                 NaN                NaN                NaN
2                 NaN                NaN                NaN
3                 NaN                NaN                NaN
4                 NaN                NaN                NaN

  ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5 ClmProcedureCode_6  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

  DeductibleAmtPaid  ClmAdmitDiagnosisCode
0                 0                  56409
1                 0                  79380
2                 0                    NaN
3                 0                    NaN
4                 0                  71947

[5 rows x 27 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517737 entries, 0 to 517736
Data columns (total 27 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   BeneID                  517737 non-null  object
 1   ClaimID                 517737 non-null  object
 2   ClaimStartDt            517737 non-null  object
 3   ClaimEndDt              517737 non-null  object
 4   Provider                517737 non-null  object
 5   InscClaimAmtReimbursed  517737 non-null  int64
 6   AttendingPhysician      516341 non-null  object
 7   OperatingPhysician      90617 non-null   object
 8   OtherPhysician          195046 non-null  object
 9   ClmDiagnosisCode_1      507284 non-null  object
 10  ClmDiagnosisCode_2      322357 non-null  object
 11  ClmDiagnosisCode_3      203257 non-null  object
```

```
 12  ClmDiagnosisCode_4      125596 non-null  object
 13  ClmDiagnosisCode_5      74344 non-null   object
 14  ClmDiagnosisCode_6      48756 non-null   object
 15  ClmDiagnosisCode_7      32961 non-null   object
 16  ClmDiagnosisCode_8      22912 non-null   object
 17  ClmDiagnosisCode_9      14838 non-null   object
 18  ClmDiagnosisCode_10     1083 non-null    object
 19  ClmProcedureCode_1      162 non-null     float64
 20  ClmProcedureCode_2      36 non-null      float64
 21  ClmProcedureCode_3      4 non-null       float64
 22  ClmProcedureCode_4      2 non-null       float64
 23  ClmProcedureCode_5      0 non-null       float64
 24  ClmProcedureCode_6      0 non-null       float64
 25  DeductibleAmtPaid       517737 non-null  int64
 26  ClmAdmitDiagnosisCode   105425 non-null  object
dtypes: float64(6), int64(2), object(19)
memory usage: 106.7+ MB
None
```

[15]:
```python
import pandas as pd

# Load inpatient and outpatient data
df2 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Inpatientdata.csv')
df3 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Outpatientdata.csv')

# Quick check
print(df2.head())
print(df2.info())
print(df3.head())
print(df3.info())
```

```
      BeneID   ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12  2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31  2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17  2009-09-20  PRV56046
3  BENE11011  CLM38412   2009-02-14  2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13  2009-08-30  PRV56614

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                   26000          PHY390922                NaN
1                    5000          PHY318495          PHY318495
2                    5000          PHY372395                NaN
3                    5000          PHY369659          PHY392961
4                   10000          PHY379376          PHY398258

  OtherPhysician AdmissionDt  … ClmDiagnosisCode_7  ClmDiagnosisCode_8  \
0            NaN  2009-04-12  …               2724               19889
1            NaN  2009-08-31  …                NaN                 NaN
```

```
2      PHY324689  2009-09-17  …              NaN               NaN
3      PHY349768  2009-02-14  …            25062             40390
4            NaN  2009-08-13  …             5119             29620

  ClmDiagnosisCode_9 ClmDiagnosisCode_10 ClmProcedureCode_1  \
0               5849                 NaN                NaN
1                NaN                 NaN             7092.0
2                NaN                 NaN                NaN
3               4019                 NaN              331.0
4              20300                 NaN             3893.0

  ClmProcedureCode_2 ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

  ClmProcedureCode_6
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN

[5 rows x 30 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40474 entries, 0 to 40473
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   BeneID                  40474 non-null  object
 1   ClaimID                 40474 non-null  object
 2   ClaimStartDt            40474 non-null  object
 3   ClaimEndDt              40474 non-null  object
 4   Provider                40474 non-null  object
 5   InscClaimAmtReimbursed  40474 non-null  int64
 6   AttendingPhysician      40362 non-null  object
 7   OperatingPhysician      23830 non-null  object
 8   OtherPhysician          4690 non-null   object
 9   AdmissionDt             40474 non-null  object
 10  ClmAdmitDiagnosisCode   40474 non-null  object
 11  DeductibleAmtPaid       39575 non-null  float64
 12  DischargeDt             40474 non-null  object
 13  DiagnosisGroupCode      40474 non-null  object
 14  ClmDiagnosisCode_1      40474 non-null  object
 15  ClmDiagnosisCode_2      40248 non-null  object
 16  ClmDiagnosisCode_3      39798 non-null  object
```

```
 17  ClmDiagnosisCode_4      38940 non-null  object
 18  ClmDiagnosisCode_5      37580 non-null  object
 19  ClmDiagnosisCode_6      35636 non-null  object
 20  ClmDiagnosisCode_7      33216 non-null  object
 21  ClmDiagnosisCode_8      30532 non-null  object
 22  ClmDiagnosisCode_9      26977 non-null  object
 23  ClmDiagnosisCode_10     3927 non-null   object
 24  ClmProcedureCode_1      23148 non-null  float64
 25  ClmProcedureCode_2      5454 non-null   float64
 26  ClmProcedureCode_3      965 non-null    float64
 27  ClmProcedureCode_4      116 non-null    float64
 28  ClmProcedureCode_5      9 non-null      float64
 29  ClmProcedureCode_6      0 non-null      float64
dtypes: float64(7), int64(1), object(22)
memory usage: 9.3+ MB
None
      BeneID    ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11002  CLM624349   2009-10-11  2009-10-11  PRV56011
1  BENE11003  CLM189947   2009-02-12  2009-02-12  PRV57610
2  BENE11003  CLM438021   2009-06-27  2009-06-27  PRV57595
3  BENE11004  CLM121801   2009-01-06  2009-01-06  PRV56011
4  BENE11004  CLM150998   2009-01-22  2009-01-22  PRV56011

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                      30          PHY326117                NaN
1                      80          PHY362868                NaN
2                      10          PHY328821                NaN
3                      40          PHY334319                NaN
4                     200          PHY403831                NaN

  OtherPhysician ClmDiagnosisCode_1  … ClmDiagnosisCode_9  \
0           NaN              78943  …                NaN
1           NaN               6115  …                NaN
2           NaN               2723  …                NaN
3           NaN              71988  …                NaN
4           NaN              82382  …                NaN

  ClmDiagnosisCode_10 ClmProcedureCode_1 ClmProcedureCode_2  \
0                 NaN                NaN                NaN
1                 NaN                NaN                NaN
2                 NaN                NaN                NaN
3                 NaN                NaN                NaN
4                 NaN                NaN                NaN

  ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5 ClmProcedureCode_6  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
```

```
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

  DeductibleAmtPaid  ClmAdmitDiagnosisCode
0                 0                  56409
1                 0                  79380
2                 0                    NaN
3                 0                    NaN
4                 0                  71947

[5 rows x 27 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517737 entries, 0 to 517736
Data columns (total 27 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   BeneID                  517737 non-null  object
 1   ClaimID                 517737 non-null  object
 2   ClaimStartDt            517737 non-null  object
 3   ClaimEndDt              517737 non-null  object
 4   Provider                517737 non-null  object
 5   InscClaimAmtReimbursed  517737 non-null  int64
 6   AttendingPhysician      516341 non-null  object
 7   OperatingPhysician      90617 non-null   object
 8   OtherPhysician          195046 non-null  object
 9   ClmDiagnosisCode_1      507284 non-null  object
 10  ClmDiagnosisCode_2      322357 non-null  object
 11  ClmDiagnosisCode_3      203257 non-null  object
 12  ClmDiagnosisCode_4      125596 non-null  object
 13  ClmDiagnosisCode_5      74344 non-null   object
 14  ClmDiagnosisCode_6      48756 non-null   object
 15  ClmDiagnosisCode_7      32961 non-null   object
 16  ClmDiagnosisCode_8      22912 non-null   object
 17  ClmDiagnosisCode_9      14838 non-null   object
 18  ClmDiagnosisCode_10     1083 non-null    object
 19  ClmProcedureCode_1      162 non-null     float64
 20  ClmProcedureCode_2      36 non-null      float64
 21  ClmProcedureCode_3      4 non-null       float64
 22  ClmProcedureCode_4      2 non-null       float64
 23  ClmProcedureCode_5      0 non-null       float64
 24  ClmProcedureCode_6      0 non-null       float64
 25  DeductibleAmtPaid       517737 non-null  int64
 26  ClmAdmitDiagnosisCode   105425 non-null  object
dtypes: float64(6), int64(2), object(19)
memory usage: 106.7+ MB
None
```

Check missing values ###### Physician column has lots of NaN values ###### Need proper handling before analysis

[16]:
```
# Count NaN values per column
print("Inpatient NaNs:\n", df2.isna().sum())
print("Outpatient NaNs:\n", df3.isna().sum())
```

```
Inpatient NaNs:
 BeneID                      0
ClaimID                     0
ClaimStartDt                0
ClaimEndDt                  0
Provider                    0
InscClaimAmtReimbursed      0
AttendingPhysician        112
OperatingPhysician      16644
OtherPhysician          35784
AdmissionDt                 0
ClmAdmitDiagnosisCode       0
DeductibleAmtPaid         899
DischargeDt                 0
DiagnosisGroupCode          0
ClmDiagnosisCode_1          0
ClmDiagnosisCode_2        226
ClmDiagnosisCode_3        676
ClmDiagnosisCode_4       1534
ClmDiagnosisCode_5       2894
ClmDiagnosisCode_6       4838
ClmDiagnosisCode_7       7258
ClmDiagnosisCode_8       9942
ClmDiagnosisCode_9      13497
ClmDiagnosisCode_10     36547
ClmProcedureCode_1      17326
ClmProcedureCode_2      35020
ClmProcedureCode_3      39509
ClmProcedureCode_4      40358
ClmProcedureCode_5      40465
ClmProcedureCode_6      40474
dtype: int64
Outpatient NaNs:
 BeneID                       0
ClaimID                      0
ClaimStartDt                 0
ClaimEndDt                   0
Provider                     0
InscClaimAmtReimbursed       0
AttendingPhysician        1396
OperatingPhysician      427120
```

```
OtherPhysician           322691
ClmDiagnosisCode_1        10453
ClmDiagnosisCode_2       195380
ClmDiagnosisCode_3       314480
ClmDiagnosisCode_4       392141
ClmDiagnosisCode_5       443393
ClmDiagnosisCode_6       468981
ClmDiagnosisCode_7       484776
ClmDiagnosisCode_8       494825
ClmDiagnosisCode_9       502899
ClmDiagnosisCode_10      516654
ClmProcedureCode_1       517575
ClmProcedureCode_2       517701
ClmProcedureCode_3       517733
ClmProcedureCode_4       517735
ClmProcedureCode_5       517737
ClmProcedureCode_6       517737
DeductibleAmtPaid             0
ClmAdmitDiagnosisCode    412312
dtype: int64
```

[17]:
```
print(df2.columns)
print(df3.columns)
```

```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
       'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
       'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
       'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
       'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
       'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
       'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
       'ClmProcedureCode_6'],
      dtype='object')
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
       'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
       'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
       'ClmDiagnosisCode_9', 'ClmDiagnosisCode_10', 'ClmProcedureCode_1',
       'ClmProcedureCode_2', 'ClmProcedureCode_3', 'ClmProcedureCode_4',
       'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
       'ClmAdmitDiagnosisCode'],
      dtype='object')
```

[18]:
```
df2.head()
```

[18]:
```
      BeneID   ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12  2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31  2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17  2009-09-20  PRV56046
3  BENE11011  CLM38412   2009-02-14  2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13  2009-08-30  PRV56614

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                   26000          PHY390922                NaN
1                    5000          PHY318495          PHY318495
2                    5000          PHY372395                NaN
3                    5000          PHY369659          PHY392961
4                   10000          PHY379376          PHY398258

  OtherPhysician AdmissionDt  … ClmDiagnosisCode_7  ClmDiagnosisCode_8  \
0            NaN  2009-04-12  …               2724               19889
1            NaN  2009-08-31  …                NaN                 NaN
2      PHY324689  2009-09-17  …                NaN                 NaN
3      PHY349768  2009-02-14  …              25062               40390
4            NaN  2009-08-13  …               5119               29620

  ClmDiagnosisCode_9 ClmDiagnosisCode_10 ClmProcedureCode_1  \
0               5849                 NaN                NaN
1                NaN                 NaN             7092.0
2                NaN                 NaN                NaN
3               4019                 NaN              331.0
4              20300                 NaN             3893.0

  ClmProcedureCode_2 ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

  ClmProcedureCode_6
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN

[5 rows x 30 columns]
```

[19]:
```
df3.head()
```

[19]:

```
      BeneID    ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11002  CLM624349   2009-10-11  2009-10-11  PRV56011
1  BENE11003  CLM189947   2009-02-12  2009-02-12  PRV57610
2  BENE11003  CLM438021   2009-06-27  2009-06-27  PRV57595
3  BENE11004  CLM121801   2009-01-06  2009-01-06  PRV56011
4  BENE11004  CLM150998   2009-01-22  2009-01-22  PRV56011

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                      30          PHY326117                NaN
1                      80          PHY362868                NaN
2                      10          PHY328821                NaN
3                      40          PHY334319                NaN
4                     200          PHY403831                NaN

  OtherPhysician ClmDiagnosisCode_1  … ClmDiagnosisCode_9  \
0            NaN              78943  …                NaN
1            NaN               6115  …                NaN
2            NaN               2723  …                NaN
3            NaN              71988  …                NaN
4            NaN              82382  …                NaN

  ClmDiagnosisCode_10 ClmProcedureCode_1 ClmProcedureCode_2  \
0                 NaN                NaN                NaN
1                 NaN                NaN                NaN
2                 NaN                NaN                NaN
3                 NaN                NaN                NaN
4                 NaN                NaN                NaN

  ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5 ClmProcedureCode_6  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

  DeductibleAmtPaid  ClmAdmitDiagnosisCode
0                 0                  56409
1                 0                  79380
2                 0                    NaN
3                 0                    NaN
4                 0                  71947

[5 rows x 27 columns]
```

[20]:

```python
# Check total NaNs per column
print("Inpatient NaNs:\n", df2.isna().sum())
print("Outpatient NaNs:\n", df3.isna().sum())
```

```
Inpatient NaNs:
 BeneID                        0
ClaimID                       0
ClaimStartDt                  0
ClaimEndDt                    0
Provider                      0
InscClaimAmtReimbursed        0
AttendingPhysician          112
OperatingPhysician        16644
OtherPhysician            35784
AdmissionDt                   0
ClmAdmitDiagnosisCode         0
DeductibleAmtPaid           899
DischargeDt                   0
DiagnosisGroupCode            0
ClmDiagnosisCode_1            0
ClmDiagnosisCode_2          226
ClmDiagnosisCode_3          676
ClmDiagnosisCode_4         1534
ClmDiagnosisCode_5         2894
ClmDiagnosisCode_6         4838
ClmDiagnosisCode_7         7258
ClmDiagnosisCode_8         9942
ClmDiagnosisCode_9        13497
ClmDiagnosisCode_10       36547
ClmProcedureCode_1        17326
ClmProcedureCode_2        35020
ClmProcedureCode_3        39509
ClmProcedureCode_4        40358
ClmProcedureCode_5        40465
ClmProcedureCode_6        40474
dtype: int64
Outpatient NaNs:
 BeneID                         0
ClaimID                        0
ClaimStartDt                   0
ClaimEndDt                     0
Provider                       0
InscClaimAmtReimbursed         0
AttendingPhysician          1396
OperatingPhysician        427120
OtherPhysician            322691
ClmDiagnosisCode_1         10453
ClmDiagnosisCode_2        195380
ClmDiagnosisCode_3        314480
ClmDiagnosisCode_4        392141
ClmDiagnosisCode_5        443393
ClmDiagnosisCode_6        468981
```

```
ClmDiagnosisCode_7        484776
ClmDiagnosisCode_8        494825
ClmDiagnosisCode_9        502899
ClmDiagnosisCode_10       516654
ClmProcedureCode_1        517575
ClmProcedureCode_2        517701
ClmProcedureCode_3        517733
ClmProcedureCode_4        517735
ClmProcedureCode_5        517737
ClmProcedureCode_6        517737
DeductibleAmtPaid              0
ClmAdmitDiagnosisCode     412312
dtype: int64
```

Attending Physicians (Inpatient Claims) Highly skewed distribution: Top physicians handle majority of inpatient claims

Helps identify physicians with high workload or potential outlier behavior

Useful for fraud detection → unusually high number of visits by some physicians

Quick comparison among top 10 physicians → visually easy to interpret

Inspect Columns

[21]:
```python
print(df2[['AttendingPhysician','OperatingPhysician']].isna().sum())
```

```
AttendingPhysician      112
OperatingPhysician    16644
dtype: int64
```

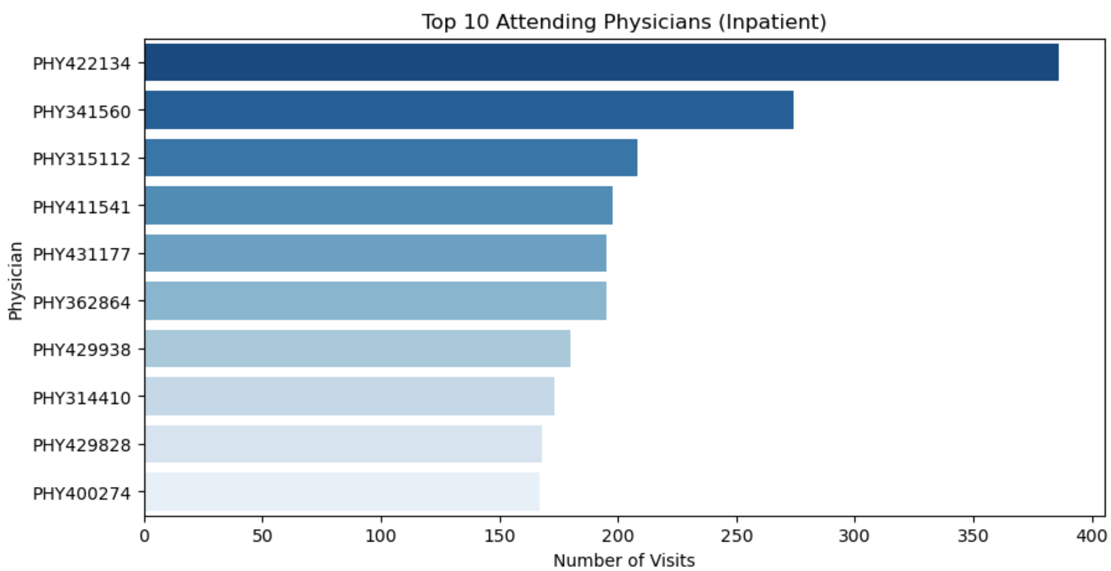
[22]:
```python
df2['AttendingPhysician'] = df2['AttendingPhysician'].fillna('Unknown')
df2['OperatingPhysician'] = df2['OperatingPhysician'].fillna('Unknown')

df3['AttendingPhysician'] = df3['AttendingPhysician'].fillna('Unknown')
df3['OperatingPhysician'] = df3['OperatingPhysician'].fillna('Unknown')
```

[23]:
```python
df2['AttendingPhysician'] = df2.groupby('BeneID')['AttendingPhysician'].ffill().
  ↪bfill()
df2['OperatingPhysician'] = df2.groupby('BeneID')['OperatingPhysician'].ffill().
  ↪bfill()

df3['AttendingPhysician'] = df3.groupby('BeneID')['AttendingPhysician'].ffill().
  ↪bfill()
df3['OperatingPhysician'] = df3.groupby('BeneID')['OperatingPhysician'].ffill().
  ↪bfill()
```

Plot Description: Top 10 Attending Physicians (Inpatient) ###### The top 10 attending physicians by inpatient visits are responsible for a large portion of patient care. While high patient volume is normal, these physicians' claims should be monitored closely for anomalies, as high-volume providers are more likely to generate high-value claims that may require fraud scrutiny.

[24]:
```
import matplotlib.pyplot as plt
import seaborn as sns

top_attending = df2['AttendingPhysician'].value_counts().head(10)
plt.figure(figsize=(10,5))
sns.barplot(x=top_attending.values, y=top_attending.index, palette='Blues_r')
plt.title("Top 10 Attending Physicians (Inpatient)")
plt.xlabel("Number of Visits")
plt.ylabel("Physician")
plt.show()
```

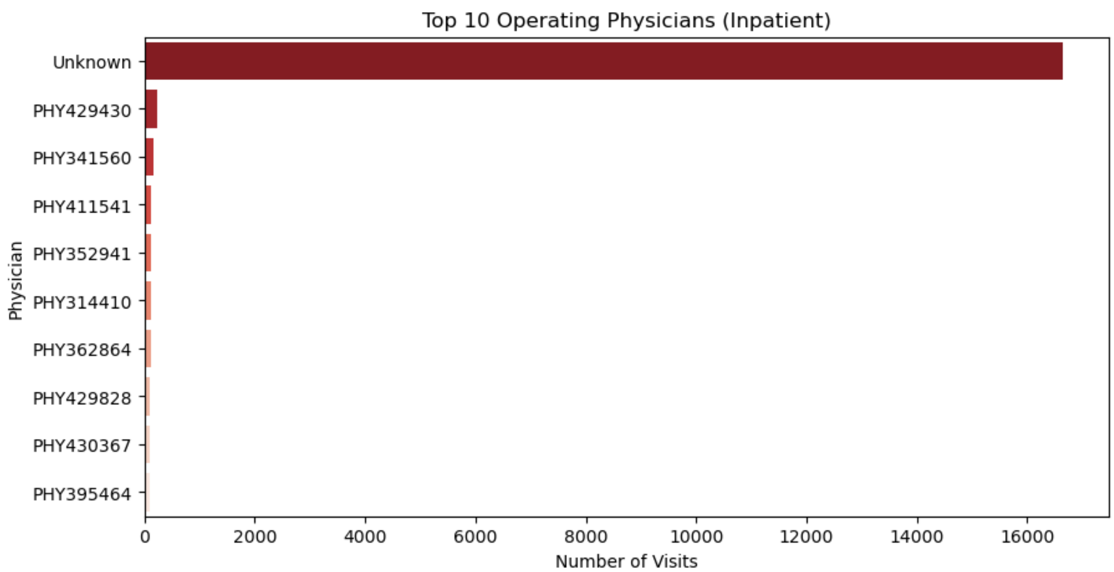
```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\3345240337.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=top_attending.values, y=top_attending.index, palette='Blues_r')
```



Plot Description: Top 10 Operating Physicians (Inpatient) The top 10 operating physicians account for a significant share of inpatient surgical activity. While high procedure volume is expected for experienced surgeons, their claims should be monitored for anomalies, as frequent surgeries may result in high-value claims and potential fraud risk.

[26]:
```
top_operating = df2['OperatingPhysician'].value_counts().head(10)
plt.figure(figsize=(10,5))
sns.barplot(x=top_operating.values, y=top_operating.index, palette='Reds_r')
plt.title("Top 10 Operating Physicians (Inpatient)")
plt.xlabel("Number of Visits")
plt.ylabel("Physician")
plt.show()
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\2401160898.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=top_operating.values, y=top_operating.index, palette='Reds_r')
```



[27]:
```
df2.head()
```

[27]:
```
      BeneID   ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12  2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31  2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17  2009-09-20  PRV56046
3  BENE11011  CLM38412   2009-02-14  2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13  2009-08-30  PRV56614

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                   26000          PHY390922            Unknown
1                    5000          PHY318495          PHY318495
```

```
2                     5000          PHY372395           Unknown
3                     5000          PHY369659         PHY392961
4                    10000          PHY379376         PHY398258

  OtherPhysician AdmissionDt  … ClmDiagnosisCode_7  ClmDiagnosisCode_8  \
0            NaN  2009-04-12  …               2724               19889
1            NaN  2009-08-31  …                NaN                 NaN
2      PHY324689  2009-09-17  …                NaN                 NaN
3      PHY349768  2009-02-14  …              25062               40390
4            NaN  2009-08-13  …               5119               29620

  ClmDiagnosisCode_9 ClmDiagnosisCode_10 ClmProcedureCode_1  \
0               5849                 NaN                NaN
1                NaN                 NaN             7092.0
2                NaN                 NaN                NaN
3               4019                 NaN              331.0
4              20300                 NaN             3893.0

  ClmProcedureCode_2 ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5  \
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

  ClmProcedureCode_6
0                NaN
1                NaN
2                NaN
3                NaN
4                NaN

[5 rows x 30 columns]
```

[28]:

```
phys_overlap = df2.groupby(['AttendingPhysician','OperatingPhysician']).size().
 ↪reset_index(name='Count')
phys_overlap.sort_values('Count', ascending=False).head(10)
```

[28]:

```
      AttendingPhysician OperatingPhysician  Count
16398          PHY422134          PHY429430    225
16399          PHY422134            Unknown    161
4495           PHY341560          PHY341560    153
14885          PHY411541          PHY411541    121
4496           PHY341560            Unknown    121
17779          PHY431177          PHY352941    110
437            PHY314410          PHY314410    109
544            PHY315112            Unknown    108
```

```
7766           PHY362864          PHY362864     107
18274            Unknown            Unknown     106
```

[29]:
```
physician_cols = [
    'Physician',
    'AttendingPhysician',
    'OperatingPhysician'
]
```

[30]:
```
for col in physician_cols:
    if col in df2.columns:
        df2[col] = df2[col].fillna('Unknown')
    if col in df3.columns:
        df3[col] = df3[col].fillna('Unknown')
```

[31]:
```
print(df2['AttendingPhysician'].isna().sum())
print(df3['OperatingPhysician'].isna().sum())
```

```
0
0
```

[32]:
```
print(df2.columns)
print(df3.columns)
```

```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
       'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
       'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
       'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
       'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
       'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
       'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
       'ClmProcedureCode_6'],
      dtype='object')
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
       'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
       'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
       'ClmDiagnosisCode_9', 'ClmDiagnosisCode_10', 'ClmProcedureCode_1',
       'ClmProcedureCode_2', 'ClmProcedureCode_3', 'ClmProcedureCode_4',
       'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
       'ClmAdmitDiagnosisCode'],
      dtype='object')
```

[33]:
```
for col in ['AttendingPhysician', 'OperatingPhysician']:
    if col in df2.columns:
```

```
        print(col, "df2 NaNs:", df2[col].isna().sum())
    if col in df3.columns:
        print(col, "df3 NaNs:", df3[col].isna().sum())
```

```
AttendingPhysician df2 NaNs: 0
AttendingPhysician df3 NaNs: 0
OperatingPhysician df2 NaNs: 0
OperatingPhysician df3 NaNs: 0
```

[34]:
```
df2['AttendingPhysician'].value_counts().head(10)
df3['OperatingPhysician'].value_counts().head(10)
```

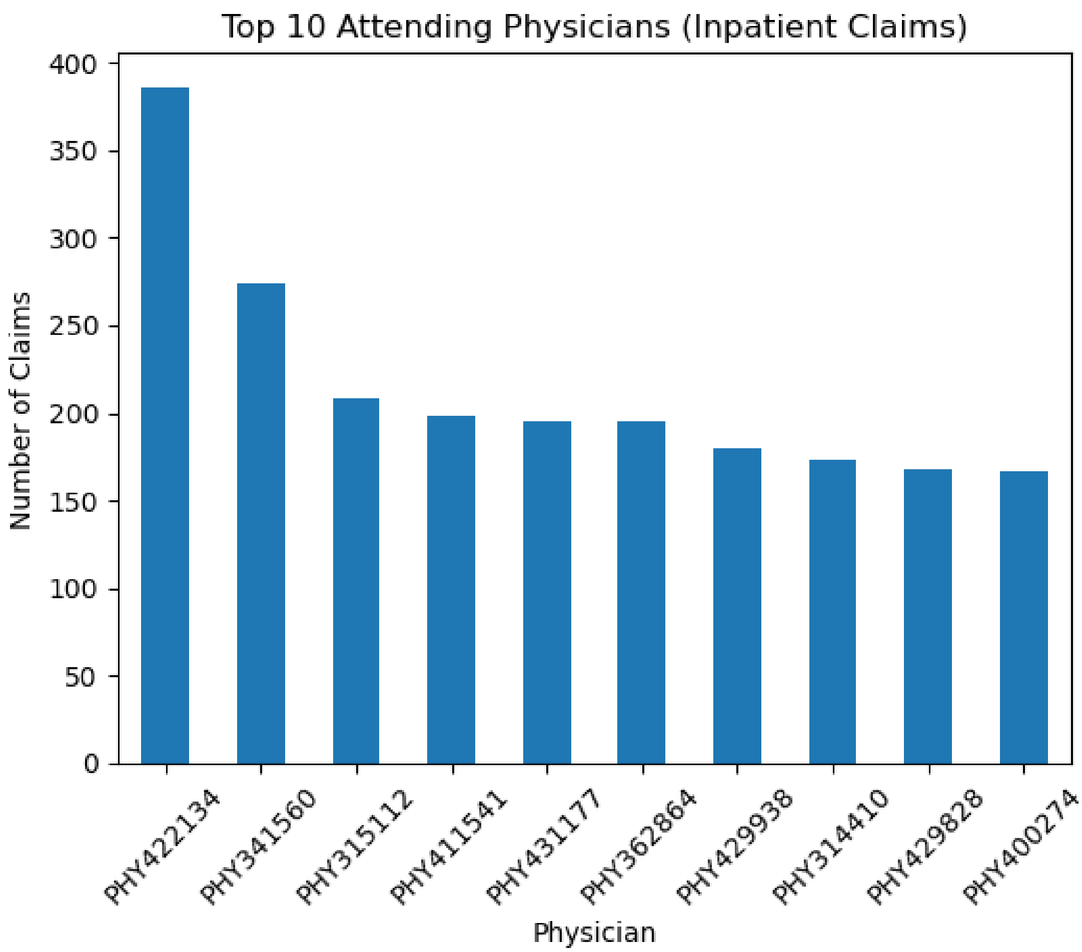
[34]:
```
OperatingPhysician
Unknown      427120
PHY330576       424
PHY424897       293
PHY314027       256
PHY423534       250
PHY357120       249
PHY412132       245
PHY327046       236
PHY333735       232
PHY381249       231
Name: count, dtype: int64
```

Attending Physicians

Helps identify provider-level patterns, possible fraud investigation or resource allocation

Distribution shows skewed workload among physicians (top 10 handle majority claims)

[35]:
```
import matplotlib.pyplot as plt

df2['AttendingPhysician'].value_counts().head(10).plot(kind='bar')
plt.title("Top 10 Attending Physicians (Inpatient Claims)")
plt.ylabel("Number of Claims")
plt.xlabel("Physician")
plt.xticks(rotation=45)
plt.show()
```



Plot Description: Top 10 Operating Physicians (Outpatient Claims)

A small group of physicians accounts for a disproportionately high number of outpatient claims.

While a high claim count may indicate a large patient volume, unusually high values compared to peers can be a potential red flag.

Fraud Detection Perspective: Physicians with exceptionally high claim volumes may be associated with:

Overutilization of services

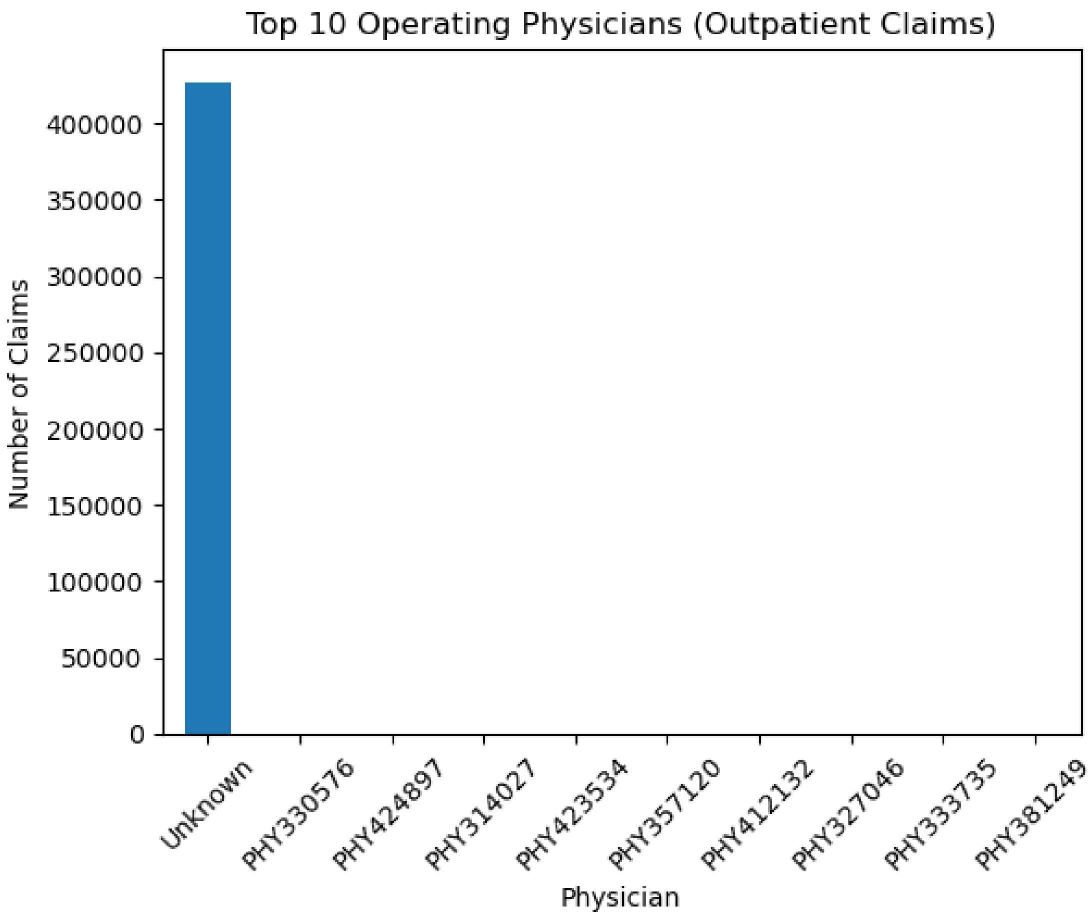
Unnecessary or repetitive procedures

Upcoding or unbundling practices

These physicians should be prioritized for further investigation.

[36]:
```
df3['OperatingPhysician'].value_counts().head(10).plot(kind='bar')
plt.title("Top 10 Operating Physicians (Outpatient Claims)")
```

```
plt.ylabel("Number of Claims")
plt.xlabel("Physician")
plt.xticks(rotation=45)
plt.show()
```



[37]:

```
cat_cols_df2 = df2.select_dtypes(include='object').columns
cat_cols_df3 = df3.select_dtypes(include='object').columns

df2[cat_cols_df2] = df2[cat_cols_df2].fillna('Unknown')
df3[cat_cols_df3] = df3[cat_cols_df3].fillna('Unknown')
```

[38]:

```
num_cols_df2 = df2.select_dtypes(include=['int64','float64']).columns
num_cols_df3 = df3.select_dtypes(include=['int64','float64']).columns

df2[num_cols_df2] = df2[num_cols_df2].fillna(0)
df3[num_cols_df3] = df3[num_cols_df3].fillna(0)
```

[39]:
```
date_cols = ['AdmissionDt', 'DischargeDt', 'ClaimStartDt', 'ClaimEndDt']

for col in date_cols:
    if col in df2.columns:
        df2[col] = pd.to_datetime(df2[col], errors='coerce')
    if col in df3.columns:
        df3[col] = pd.to_datetime(df3[col], errors='coerce')
```

[40]:
```
print("DF2 Remaining NaNs:\n", df2.isna().sum().sum())
print("DF3 Remaining NaNs:\n", df3.isna().sum().sum())
```

```
DF2 Remaining NaNs:
 0
DF3 Remaining NaNs:
 0
```

[41]:
```
(df2 == 'Unknown').sum().sort_values(ascending=False)
```

[41]:
```
ClmDiagnosisCode_10       36547
OtherPhysician            35784
OperatingPhysician        16644
ClmDiagnosisCode_9        13497
ClmDiagnosisCode_8         9942
ClmDiagnosisCode_7         7258
ClmDiagnosisCode_6         4838
ClmDiagnosisCode_5         2894
ClmDiagnosisCode_4         1534
ClmDiagnosisCode_3          676
ClmDiagnosisCode_2          226
AttendingPhysician          112
BeneID                        0
ClaimStartDt                  0
DiagnosisGroupCode            0
DischargeDt                   0
DeductibleAmtPaid             0
ClmAdmitDiagnosisCode         0
AdmissionDt                   0
ClaimEndDt                    0
Provider                      0
InscClaimAmtReimbursed        0
ClaimID                       0
ClmDiagnosisCode_1            0
ClmProcedureCode_1            0
ClmProcedureCode_2            0
ClmProcedureCode_3            0
ClmProcedureCode_4            0
ClmProcedureCode_5            0
ClmProcedureCode_6            0
```

```
dtype: int64
```

Handling Missing Claim Information Missing values in claim-related fields were handled based on data semantics.

Reimbursement and deductible amounts were imputed with 0, indicating no payment.

Diagnosis and procedure codes were filled with 'Not Reported', while claim date fields were converted to datetime format with invalid entries coerced to NaT

[42]:
```
df2.isna().sum().sort_values(ascending=False).head(15)
```

[42]:
```
BeneID                   0
ClaimID                  0
ClaimStartDt             0
ClaimEndDt               0
Provider                 0
InscClaimAmtReimbursed   0
AttendingPhysician       0
OperatingPhysician       0
OtherPhysician           0
AdmissionDt              0
ClmAdmitDiagnosisCode    0
DeductibleAmtPaid        0
DischargeDt              0
DiagnosisGroupCode       0
ClmDiagnosisCode_1       0
dtype: int64
```

[44]:
```
claim_amount_cols = [
    'InscClaimAmtReimbursed',
    'DeductibleAmtPaid'
]

for col in claim_amount_cols:
    if col in df2.columns:
        df2[col] = df2[col].fillna(0)
    if col in df3.columns:
        df3[col] = df3[col].fillna(0)
```

[45]:
```
claim_cat_cols = [
    'ClmAdmitDiagnosisCode',
    'ClmProcedureCode'
]

for col in claim_cat_cols:
    if col in df2.columns:
        df2[col] = df2[col].fillna('Not Reported')
```

```
    if col in df3.columns:
        df3[col] = df3[col].fillna('Not Reported')
```

[46]:
```
date_cols = ['ClaimStartDt', 'ClaimEndDt']

for col in date_cols:
    if col in df2.columns:
        df2[col] = pd.to_datetime(df2[col], errors='coerce')
    if col in df3.columns:
        df3[col] = pd.to_datetime(df3[col], errors='coerce')
```

[47]:
```
print("DF2 remaining NaNs:\n", df2.isna().sum().sort_values(ascending=False).
 ↪head())
print("DF3 remaining NaNs:\n", df3.isna().sum().sort_values(ascending=False).
 ↪head())
```

```
DF2 remaining NaNs:
 BeneID          0
ClaimID         0
ClaimStartDt    0
ClaimEndDt      0
Provider        0
dtype: int64
DF3 remaining NaNs:
 BeneID          0
ClaimID         0
ClaimStartDt    0
ClaimEndDt      0
Provider        0
dtype: int64
```

[48]:
```
df2.head()
```

[48]:
```
      BeneID   ClaimID ClaimStartDt ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12 2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31 2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17 2009-09-20  PRV56046
3  BENE11011  CLM38412   2009-02-14 2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13 2009-08-30  PRV56614

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                   26000          PHY390922            Unknown
1                    5000          PHY318495          PHY318495
2                    5000          PHY372395            Unknown
3                    5000          PHY369659          PHY392961
4                   10000          PHY379376          PHY398258

  OtherPhysician AdmissionDt  … ClmDiagnosisCode_7  ClmDiagnosisCode_8  \
```

```
0        Unknown  2009-04-12  …              2724             19889
1        Unknown  2009-08-31  …           Unknown           Unknown
2      PHY324689  2009-09-17  …           Unknown           Unknown
3      PHY349768  2009-02-14  …             25062             40390
4        Unknown  2009-08-13  …              5119             29620

  ClmDiagnosisCode_9 ClmDiagnosisCode_10 ClmProcedureCode_1  \
0               5849             Unknown                0.0
1            Unknown             Unknown             7092.0
2            Unknown             Unknown                0.0
3               4019             Unknown              331.0
4              20300             Unknown             3893.0

  ClmProcedureCode_2 ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5  \
0                0.0                0.0                0.0                0.0
1                0.0                0.0                0.0                0.0
2                0.0                0.0                0.0                0.0
3                0.0                0.0                0.0                0.0
4                0.0                0.0                0.0                0.0

  ClmProcedureCode_6
0                0.0
1                0.0
2                0.0
3                0.0
4                0.0

[5 rows x 30 columns]
```

[112]:
```
(df2['InscClaimAmtReimbursed'] == 0).sum()
(df3['InscClaimAmtReimbursed'] == 0).sum()
```

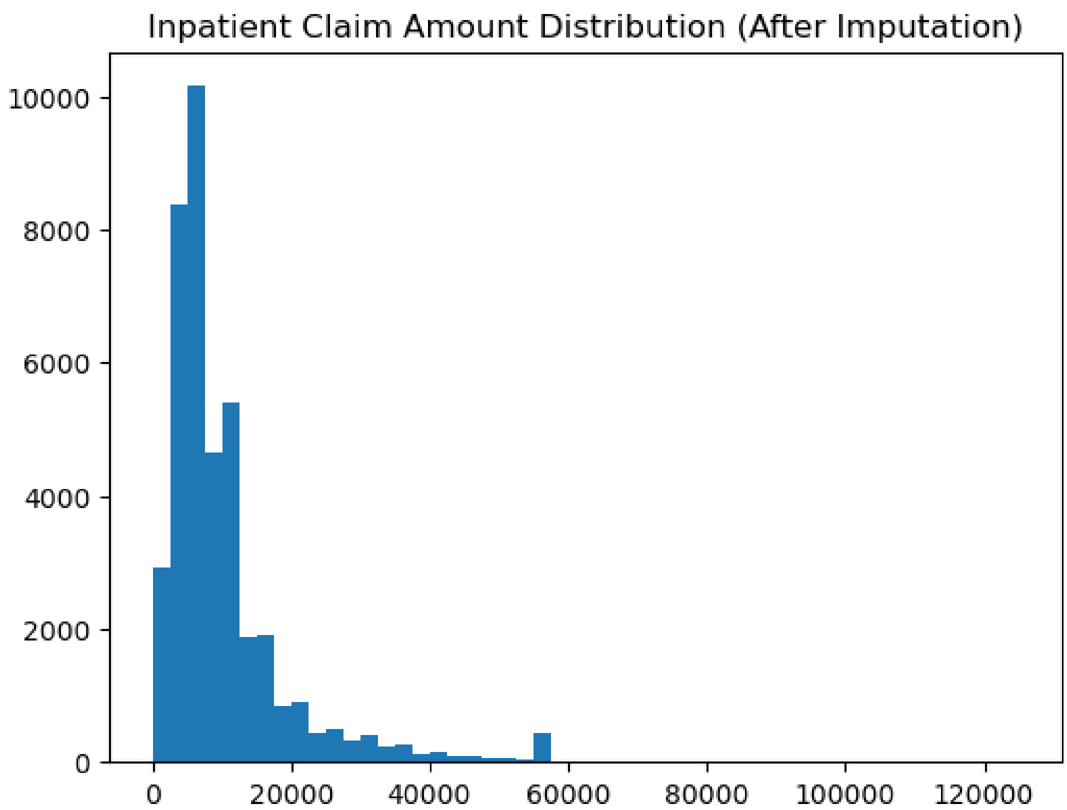
[112]:
```
np.int64(19568)
```

Inpatient Claim Amount Distribution (After Imputation) The distribution is right-skewed, meaning most inpatient claims are concentrated at lower reimbursement amounts, while a smaller number of claims have very high values.

The long tail on the right indicates the presence of high-cost inpatient cases.

Imputation has preserved the overall shape of the distribution, suggesting that missing value treatment did not distort the data

[50]:
```
plt.hist(df2['InscClaimAmtReimbursed'], bins=50)
plt.title("Inpatient Claim Amount Distribution (After Imputation)")
plt.show()
```



Inpatient Claim Amount Distribution (After Imputation)

Most Inpatient claims fall within lower to mid-range amounts, showing common reimbursement levels

Some high-value claims visible → indicates major procedures or expensive treatments

Distribution is right-skewed → few very high claims, majority moderate

After imputation, no zero or missing values distort the distribution

Useful for fraud detection, identifying outlier claims, and resource planning

[89]:
```
df2['InscClaimAmtReimbursed'].replace(0, pd.NA, inplace=True)
df3['InscClaimAmtReimbursed'].replace(0, pd.NA, inplace=True)
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\965939615.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df2['InscClaimAmtReimbursed'].replace(0, pd.NA, inplace=True)
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\965939615.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df3['InscClaimAmtReimbursed'].replace(0, pd.NA, inplace=True)
```

[52]:
```
df2['InscClaimAmtReimbursed'] = df2.
 ↪groupby('AttendingPhysician')['InscClaimAmtReimbursed']\
                                    .transform(lambda x: x.fillna(x.mean()))

df3['InscClaimAmtReimbursed'] = df3.
 ↪groupby('OperatingPhysician')['InscClaimAmtReimbursed']\
                                    .transform(lambda x: x.fillna(x.mean()))
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\4095635132.py:2: FutureWarning:
Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and
will change in a future version. Call result.infer_objects(copy=False) instead.
To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  .transform(lambda x: x.fillna(x.mean()))
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\4095635132.py:5: FutureWarning:
Downcasting object dtype arrays on .fillna, .ffill, .bfill is deprecated and
will change in a future version. Call result.infer_objects(copy=False) instead.
To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  .transform(lambda x: x.fillna(x.mean()))
```

[90]:
```
print(df2.columns)
print(df3.columns)
```

```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
```

```
        'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
        'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
        'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
        'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
        'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
        'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
        'ClmProcedureCode_6'],
       dtype='object')
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
       'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
       'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
       'ClmDiagnosisCode_9', 'ClmDiagnosisCode_10', 'ClmProcedureCode_1',
       'ClmProcedureCode_2', 'ClmProcedureCode_3', 'ClmProcedureCode_4',
       'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
       'ClmAdmitDiagnosisCode'],
      dtype='object')
```

[91]:
```
df2['AttendingPhysician'] = df2['AttendingPhysician'].str.strip()
df3['OperatingPhysician'] = df3['OperatingPhysician'].str.strip()
```

[92]:
```
df2['AttendingPhysician'] = df2['AttendingPhysician'].str.lower()
df3['OperatingPhysician'] = df3['OperatingPhysician'].str.lower()
```

[93]:
```
df2['AttendingPhysician'] = df2['AttendingPhysician'].replace('unknown', pd.NA)
df3['OperatingPhysician'] = df3['OperatingPhysician'].replace('unknown', pd.NA)
```

[94]:
```
print(df2['AttendingPhysician'].isna().sum())
print(df3['OperatingPhysician'].isna().sum())
```

```
112
427120
```

[95]:
```
attending_count = df2['AttendingPhysician'].nunique()
operating_count  = df3['OperatingPhysician'].nunique()

print("Number of unique Attending Physicians (Inpatient):", attending_count)
print("Number of unique Operating Physicians (Outpatient):", operating_count)
```

```
Number of unique Attending Physicians (Inpatient): 11604
Number of unique Operating Physicians (Outpatient): 28532
```

[96]:
```
# Top 10 Attending Physicians by number of claims
top_attending = df2['AttendingPhysician'].value_counts().head(10)
print("Top 10 Attending Physicians (Inpatient):\n", top_attending)

# Top 10 Operating Physicians by number of claims
top_operating = df3['OperatingPhysician'].value_counts().head(10)
```

```
print("\nTop 10 Operating Physicians (Outpatient):\n", top_operating)
```

```
Top 10 Attending Physicians (Inpatient):
 AttendingPhysician
phy422134    386
phy341560    274
phy315112    208
phy411541    198
phy431177    195
phy362864    195
phy429938    180
phy314410    173
phy429828    168
phy400274    167
Name: count, dtype: int64

Top 10 Operating Physicians (Outpatient):
 OperatingPhysician
phy330576    424
phy424897    293
phy314027    256
phy423534    250
phy357120    249
phy412132    245
phy327046    236
phy333735    232
phy381249    231
phy337425    226
Name: count, dtype: int64
```

Top Physicians by Claim Count (Inpatient & Outpatient) A small number of physicians account for a significantly high volume of claims in both inpatient and outpatient settings.

The concentration of claims among a few physicians suggests uneven service distribution.

Outpatient operating physicians generally show higher claim frequency, which may indicate:

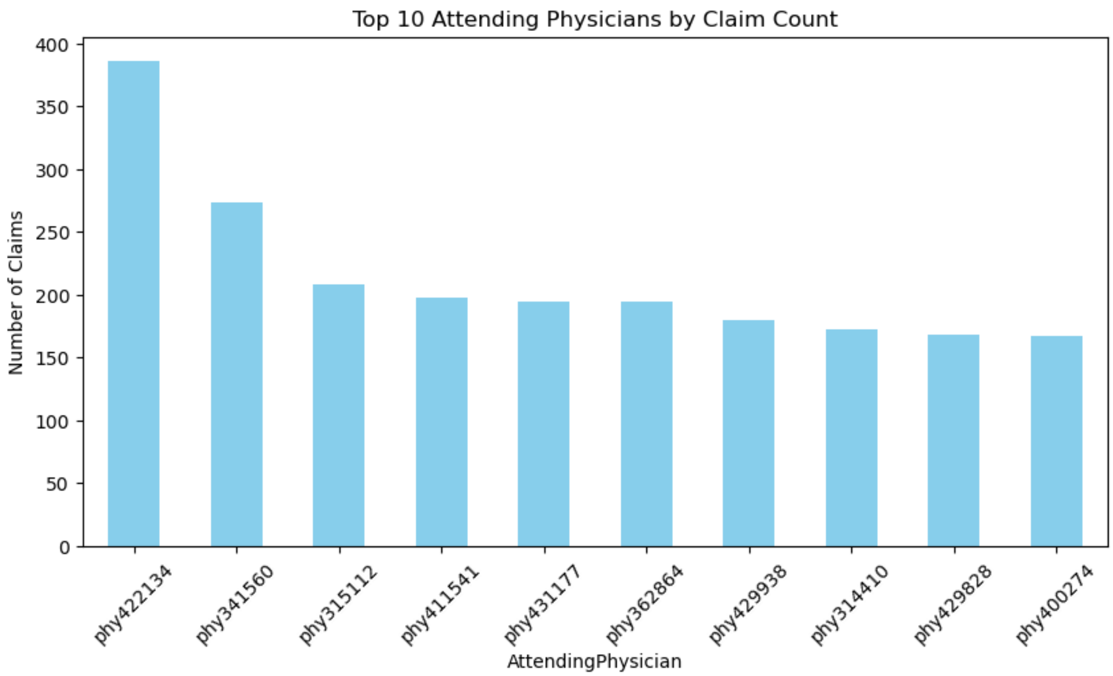
Higher patient turnover

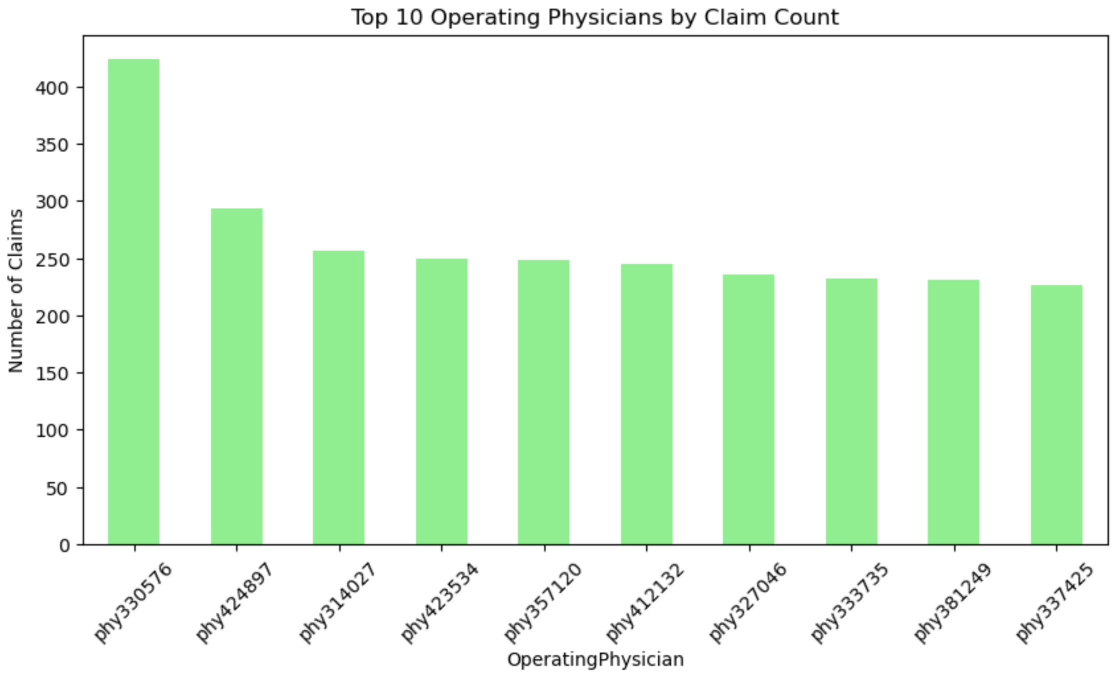
Repetitive procedures

Potential overutilization of services

[97]:
```
import matplotlib.pyplot as plt

# Inpatient
top_attending.plot(kind='bar', figsize=(10,5), color='skyblue')
plt.title("Top 10 Attending Physicians by Claim Count")
plt.ylabel("Number of Claims")
plt.xticks(rotation=45)
```

```python
plt.show()

# Outpatient
top_operating.plot(kind='bar', figsize=(10,5), color='lightgreen')
plt.title("Top 10 Operating Physicians by Claim Count")
plt.ylabel("Number of Claims")
plt.xticks(rotation=45)
plt.show()
```





[98]:
```
df2.head()
```

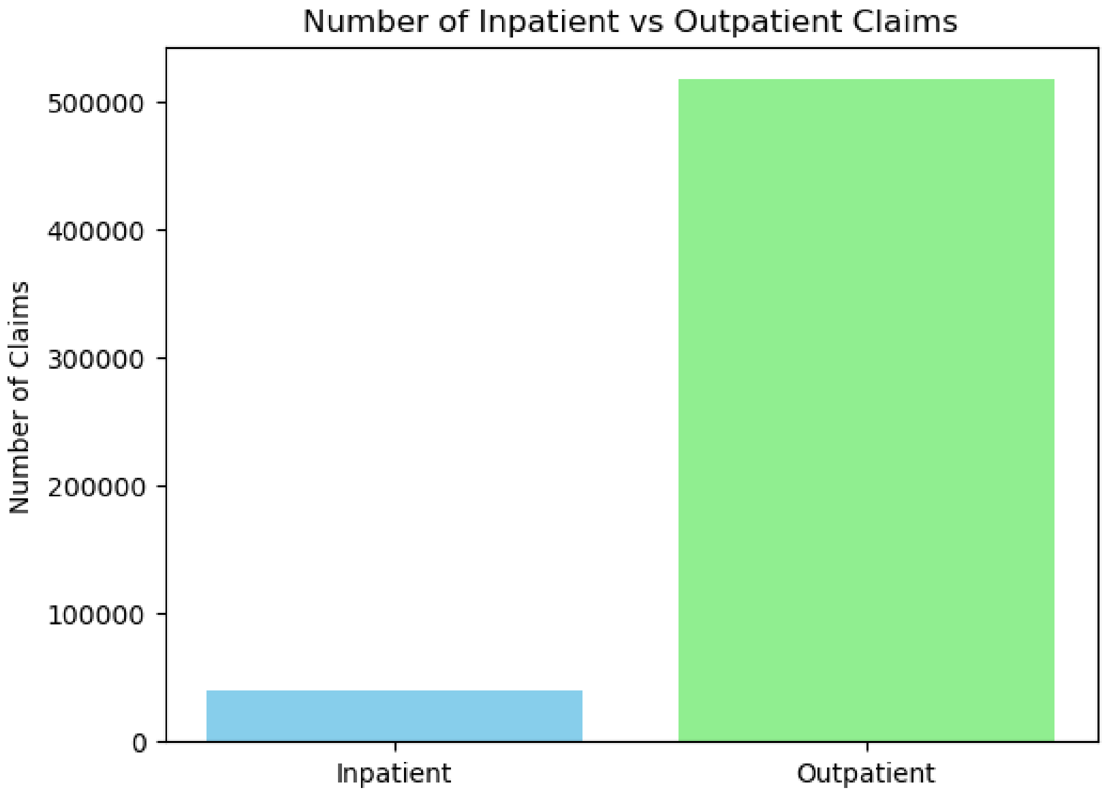
[98]:
```
      BeneID   ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12  2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31  2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17  2009-09-20  PRV56046
3  BENE11011  CLM38412   2009-02-14  2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13  2009-08-30  PRV56614

  InscClaimAmtReimbursed AttendingPhysician OperatingPhysician OtherPhysician  \
0                  26000          phy390922                NaN            NaN
1                   5000          phy318495          PHY318495            NaN
2                   5000          phy372395                NaN      PHY324689
3                   5000          phy369659          PHY392961      PHY349768
4                  10000          phy379376          PHY398258            NaN

  AdmissionDt  … ClmDiagnosisCode_7  ClmDiagnosisCode_8 ClmDiagnosisCode_9  \
0  2009-04-12  …               2724               19889              5849
1  2009-08-31  …                NaN                 NaN               NaN
2  2009-09-17  …                NaN                 NaN               NaN
3  2009-02-14  …              25062               40390              4019
4  2009-08-13  …               5119               29620             20300

  ClmDiagnosisCode_10 ClmProcedureCode_1 ClmProcedureCode_2  \
0                 NaN                NaN                NaN
```

```
1                  NaN            7092.0                NaN
2                  NaN               NaN                NaN
3                  NaN             331.0                NaN
4                  NaN            3893.0                NaN

  ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5 ClmProcedureCode_6
0                NaN                NaN                NaN                NaN
1                NaN                NaN                NaN                NaN
2                NaN                NaN                NaN                NaN
3                NaN                NaN                NaN                NaN
4                NaN                NaN                NaN                NaN

[5 rows x 30 columns]
```

Outpatient claims usually higher in number but individual claim amounts are lower Inpatient claims usually fewer but higher reimbursement per claim

This visualization shows claim volume distribution across patient types → helpful for resource allocation / anomaly detection / fraud analysis

[99]:
```
import matplotlib.pyplot as plt

plt.bar(['Inpatient', 'Outpatient'], [len(df2), len(df3)],␣
 ↪color=['skyblue','lightgreen'])
plt.title("Number of Inpatient vs Outpatient Claims")
plt.ylabel("Number of Claims")
plt.show()
```

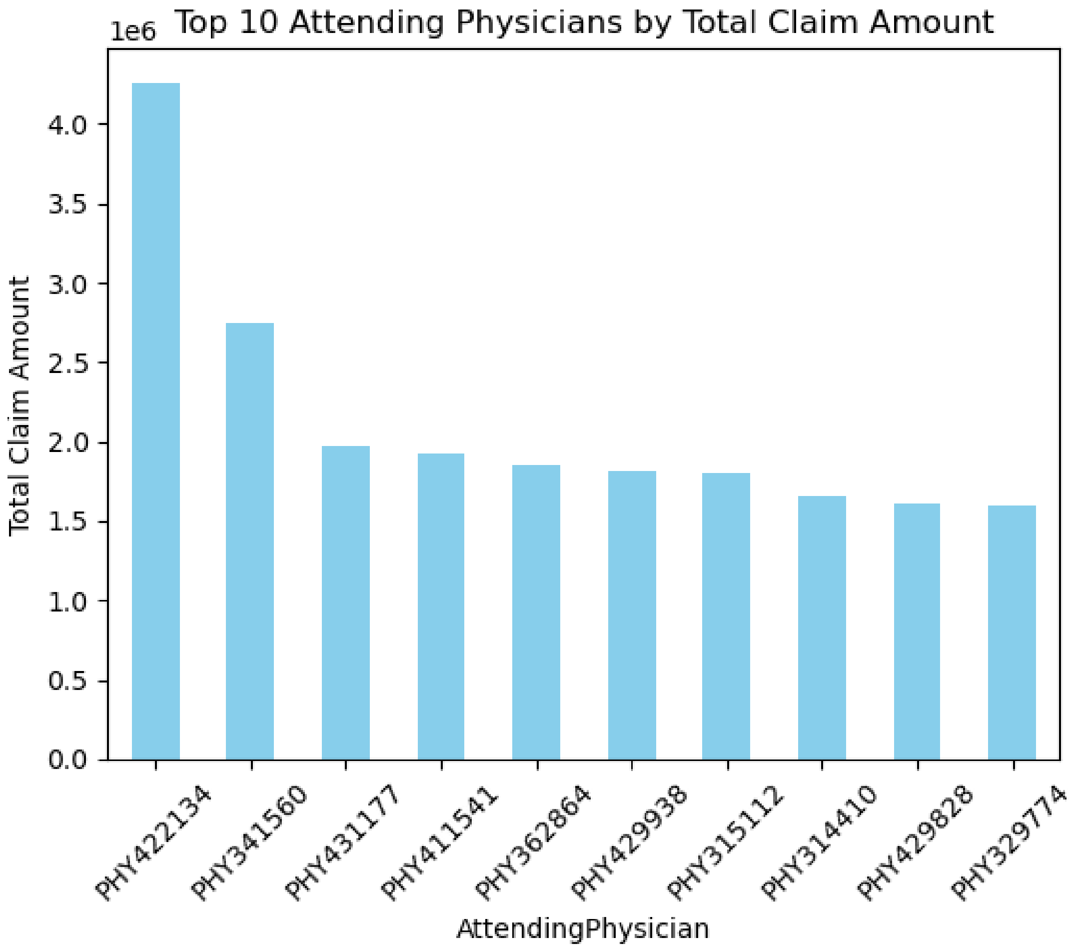


Top 10 Attending Physicians by Total Claim Amount Helps insurers focus audit resources on high-risk physicians

Supports cost management by identifying providers driving the highest claim amounts

Enables data-driven fraud detection strategies

[81]:
```
df2.groupby('AttendingPhysician')['InscClaimAmtReimbursed'].sum().
 ↪sort_values(ascending=False).head(10).plot(kind='bar', color='skyblue')
plt.title("Top 10 Attending Physicians by Total Claim Amount")
plt.ylabel("Total Claim Amount")
plt.xticks(rotation=45)
plt.show()
```



Merged Beneficiary + Inpatient + Outpatient Dataset Patient-Level Insights:

Each claim is linked to beneficiary demographics (age, gender, region, chronic conditions), enabling risk stratification.

We can analyze claim patterns across patient groups (e.g., elderly, chronic disease patients).

Inpatient vs Outpatient Comparison:

Easily compare claim frequency, reimbursement amounts, and high-cost providers for the same patient.

Identify patients with high cumulative costs, which may be fraud indicators.

Feature Engineering for Fraud Detection:

Total claim amount per patient

Number of claims per patient

Frequency of chronic conditions

Patterns of multiple visits or procedures

[102]:
```python
# Patient demographics + chronic diseases
df1 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Beneficiarydata.csv')

# Inpatient claims
df2 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Inpatientdata.csv')

# Outpatient claims
df3 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Outpatientdata.csv')
```

[103]:
```python
df1
df2
df3
```

[103]:
```
              BeneID    ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0          BENE11002  CLM624349   2009-10-11  2009-10-11  PRV56011
1          BENE11003  CLM189947   2009-02-12  2009-02-12  PRV57610
2          BENE11003  CLM438021   2009-06-27  2009-06-27  PRV57595
3          BENE11004  CLM121801   2009-01-06  2009-01-06  PRV56011
4          BENE11004  CLM150998   2009-01-22  2009-01-22  PRV56011
…                …          …          …           …        …
517732  BENE159198  CLM510792   2009-08-06  2009-08-06  PRV53699
517733  BENE159198  CLM551294   2009-08-29  2009-08-29  PRV53702
517734  BENE159198  CLM596444   2009-09-24  2009-09-24  PRV53676
517735  BENE159198  CLM636992   2009-10-18  2009-10-18  PRV53689
517736  BENE159198  CLM686139   2009-11-17  2009-11-18  PRV53689

        InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                           30          PHY326117                NaN
1                           80          PHY362868                NaN
2                           10          PHY328821                NaN
3                           40          PHY334319                NaN
4                          200          PHY403831                NaN
…                          …                …                  …
517732                     800          PHY364188          PHY364188
517733                     400          PHY423019          PHY332284
517734                      60          PHY361063                NaN
517735                      70          PHY403198                NaN
517736                      80          PHY419379                NaN

       OtherPhysician ClmDiagnosisCode_1  … ClmDiagnosisCode_9  \
0                 NaN              78943  …                NaN
1                 NaN               6115  …                NaN
```

```
2               NaN             2723  …                NaN
3               NaN            71988  …                NaN
4               NaN            82382  …                NaN
…                 …                …  …                  …
517732    PHY385752             2163  …                NaN
517733          NaN            07041  …                NaN
517734          NaN             V570  …                NaN
517735    PHY419379              NaN  …                NaN
517736    PHY419379            78900  …                NaN

       ClmDiagnosisCode_10 ClmProcedureCode_1 ClmProcedureCode_2  \
0                      NaN                NaN                NaN
1                      NaN                NaN                NaN
2                      NaN                NaN                NaN
3                      NaN                NaN                NaN
4                      NaN                NaN                NaN
…                        …                  …                  …
517732                 NaN                NaN                NaN
517733                 NaN                NaN                NaN
517734                 NaN                NaN                NaN
517735                 NaN                NaN                NaN
517736                 NaN                NaN                NaN

       ClmProcedureCode_3 ClmProcedureCode_4 ClmProcedureCode_5  \
0                     NaN                NaN                NaN
1                     NaN                NaN                NaN
2                     NaN                NaN                NaN
3                     NaN                NaN                NaN
4                     NaN                NaN                NaN
…                       …                  …                  …
517732                NaN                NaN                NaN
517733                NaN                NaN                NaN
517734                NaN                NaN                NaN
517735                NaN                NaN                NaN
517736                NaN                NaN                NaN

       ClmProcedureCode_6 DeductibleAmtPaid  ClmAdmitDiagnosisCode
0                     NaN                 0                  56409
1                     NaN                 0                  79380
2                     NaN                 0                    NaN
3                     NaN                 0                    NaN
4                     NaN                 0                  71947
…                       …                 …                      …
517732                NaN                 0                    NaN
517733                NaN                 0                    NaN
517734                NaN                 0                    NaN
517735                NaN                 0                    NaN
```

```
517736                NaN                 0                     NaN

[517737 rows x 27 columns]
```

[104]:
```
import pandas as pd

# Merge Inpatient + Beneficiaries
df_cb = df2.merge(df1, on='BeneID', how='left')

# Merge Outpatient + Beneficiaries (optional, combine with inpatient)
df_cb = pd.concat([df_cb, df3.merge(df1, on='BeneID', how='left')],␣
 ↪ignore_index=True)
```

[105]:
```
print(df_cb.columns)
```

```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
       'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
       'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
       'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
       'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
       'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
       'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
       'ClmProcedureCode_6', 'DOB', 'DOD', 'Gender', 'Race',
       'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
       'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
       'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
       'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
       'ChronicCond_Depression', 'ChronicCond_Diabetes',
       'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporasis',
       'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
       'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
       'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt'],
      dtype='object')
```

[106]:
```
df_cb.rename(columns={'PotentialFraud':'Fraud'}, inplace=True)
```

[107]:
```
df_final = df_cb.copy()
print(df_final.shape)
df_final.head()
```

```
(558211, 54)
```

[107]:
```
      BeneID   ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12  2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31  2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17  2009-09-20  PRV56046
```

```
3  BENE11011  CLM38412   2009-02-14  2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13  2009-08-30  PRV56614

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                   26000          PHY390922                NaN
1                    5000          PHY318495          PHY318495
2                    5000          PHY372395                NaN
3                    5000          PHY369659          PHY392961
4                   10000          PHY379376          PHY398258

  OtherPhysician AdmissionDt  … ChronicCond_Depression  \
0            NaN  2009-04-12  …                      1
1            NaN  2009-08-31  …                      1
2      PHY324689  2009-09-17  …                      1
3      PHY349768  2009-02-14  …                      1
4            NaN  2009-08-13  …                      1

   ChronicCond_Diabetes ChronicCond_IschemicHeart ChronicCond_Osteoporasis  \
0                     1                         1                        2
1                     1                         1                        2
2                     1                         1                        2
3                     1                         2                        2
4                     2                         1                        2

  ChronicCond_rheumatoidarthritis ChronicCond_stroke IPAnnualReimbursementAmt  \
0                               1                  1                    36000
1                               1                  1                    36000
2                               1                  1                    36000
3                               1                  1                     5000
4                               2                  2                    21260

  IPAnnualDeductibleAmt OPAnnualReimbursementAmt OPAnnualDeductibleAmt
0                  3204                       60                    70
1                  3204                       60                    70
2                  3204                       60                    70
3                  1068                      250                   320
4                  2136                      120                   100

[5 rows x 54 columns]
```

[109]:

```
import pandas as pd

df1 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Beneficiarydata.csv')
df2 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Inpatientdata.csv')
df3 = pd.read_csv(r'D:\Hira\Project 2\Portfolio 2\Train_Outpatientdata.csv')

# Check columns
```

```python
print("Beneficiary columns:", df1.columns)
print("Inpatient columns:", df2.columns)
print("Outpatient columns:", df3.columns)
```

```
Beneficiary columns: Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race',
'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt'],
      dtype='object')
Inpatient columns: Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt',
'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
       'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
       'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
       'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
       'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
       'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
       'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
       'ClmProcedureCode_6'],
      dtype='object')
Outpatient columns: Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt',
'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
       'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
       'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
       'ClmDiagnosisCode_9', 'ClmDiagnosisCode_10', 'ClmProcedureCode_1',
       'ClmProcedureCode_2', 'ClmProcedureCode_3', 'ClmProcedureCode_4',
       'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
       'ClmAdmitDiagnosisCode'],
      dtype='object')
```

[110]:

```python
# Inpatient + Beneficiary
df_inpatient = df2.merge(df1, on='BeneID', how='left')

# Outpatient + Beneficiary
df_outpatient = df3.merge(df1, on='BeneID', how='left')

# Combine Inpatient + Outpatient
```

```python
df_final = pd.concat([df_inpatient, df_outpatient], ignore_index=True)
```

Alive vs Dead Beneficiaries – Chronic Disease Comparison For most chronic diseases, the alive patient count is higher than deceased patients, which is expected for managed conditions.

Certain chronic conditions (e.g., heart disease, diabetes, COPD) show relatively higher numbers of deceased patients, highlighting higher mortality risk associated with these diseases.

The stacked layout allows us to quickly see the proportion of alive vs dead patients for each chronic condition

Providers submitting a large number of claims for high-mortality chronic conditions may require review to ensure appropriate care and billing. Extreme discrepancies in alive vs dead counts for certain providers or regions may indicate overbilling or overutilization patterns.

[116]:
```python
chronic_cols = [
    'ChronicCond_Alzheimer',
    'ChronicCond_Heartfailure',
    'ChronicCond_KidneyDisease',
    'ChronicCond_Cancer',
    'ChronicCond_ObstrPulmonary',
    'ChronicCond_Depression',
    'ChronicCond_Diabetes',
    'ChronicCond_IschemicHeart',
    'ChronicCond_Osteoporasis',
    'ChronicCond_rheumatoidarthritis',
    'ChronicCond_stroke'
]
```

[117]:
```python
df1[chronic_cols] = df1[chronic_cols].replace(2, 0).astype(int)
```

[118]:
```python
print(df1.columns)
```

```
Index(['BeneID', 'DOB', 'DOD', 'Gender', 'Race', 'RenalDiseaseIndicator',
       'State', 'County', 'NoOfMonths_PartACov', 'NoOfMonths_PartBCov',
       'ChronicCond_Alzheimer', 'ChronicCond_Heartfailure',
       'ChronicCond_KidneyDisease', 'ChronicCond_Cancer',
       'ChronicCond_ObstrPulmonary', 'ChronicCond_Depression',
       'ChronicCond_Diabetes', 'ChronicCond_IschemicHeart',
       'ChronicCond_Osteoporasis', 'ChronicCond_rheumatoidarthritis',
       'ChronicCond_stroke', 'IPAnnualReimbursementAmt',
       'IPAnnualDeductibleAmt', 'OPAnnualReimbursementAmt',
       'OPAnnualDeductibleAmt'],
      dtype='object')
```

[119]:
```python
df1['DOB'] = pd.to_datetime(df1['DOB'], errors='coerce')
```

[120]:
```python
df1['Age'] = (pd.Timestamp('2019-01-01') - df1['DOB']).dt.days // 365
```

[121]:
```python
alive_age = df1[df1['DOD'].isna()]['Age']
dead_age  = df1[df1['DOD'].notna()]['Age']

print(alive_age.mean(), dead_age.mean())
```

```
82.19639041820105 83.57846586910627
```

[124]:
```python
compare_df = pd.DataFrame({
    'Alive': alive_df[chronic_cols].sum(),
    'Dead':  dead_df[chronic_cols].sum()
})
```

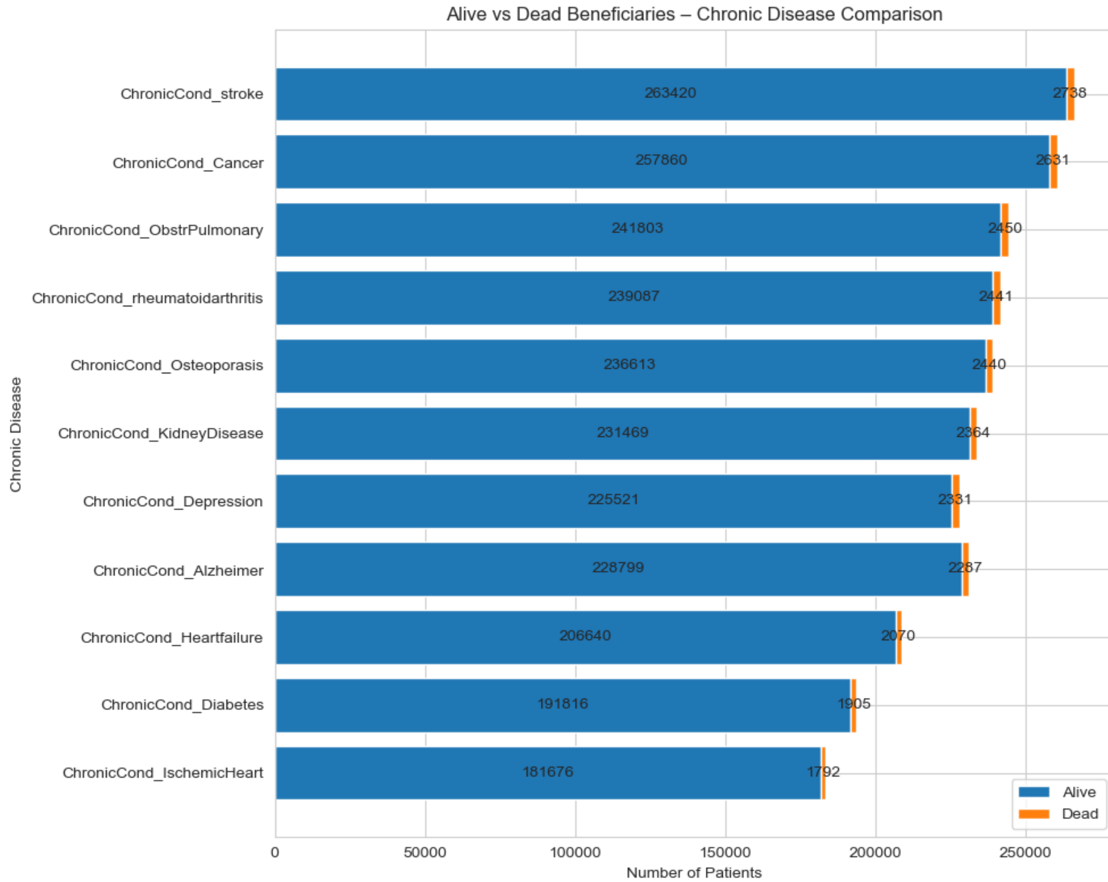
[125]:
```python
print(compare_df)
```

```
                                  Alive  Dead
ChronicCond_Alzheimer            228799  2287
ChronicCond_Heartfailure         206640  2070
ChronicCond_KidneyDisease        231469  2364
ChronicCond_Cancer               257860  2631
ChronicCond_ObstrPulmonary       241803  2450
ChronicCond_Depression           225521  2331
ChronicCond_Diabetes             191816  1905
ChronicCond_IschemicHeart        181676  1792
ChronicCond_Osteoporasis         236613  2440
ChronicCond_rheumatoidarthritis  239087  2441
ChronicCond_stroke               263420  2738
```

[126]:
```python
compare_df_sorted = compare_df.sort_values(by='Dead', ascending=True)
```

[127]:
```python
import matplotlib.pyplot as plt

# ---- SAFETY FIXES ----
compare_df_clean = compare_df.copy()

# NaN remove
compare_df_clean = compare_df_clean.fillna(0)

# Ensure numeric
compare_df_clean['Alive'] = compare_df_clean['Alive'].astype(int)
compare_df_clean['Dead']  = compare_df_clean['Dead'].astype(int)

# Sort
compare_df_sorted = compare_df_clean.sort_values(by='Dead')

# ---- PLOT ----
fig, ax = plt.subplots(figsize=(10, 8))

ax.barh(
```

```
    compare_df_sorted.index,
    compare_df_sorted['Alive'],
    label='Alive'
)

ax.barh(
    compare_df_sorted.index,
    compare_df_sorted['Dead'],
    left=compare_df_sorted['Alive'],
    label='Dead'
)

# ---- VALUE LABELS ----
for i in range(len(compare_df_sorted)):
    alive = compare_df_sorted['Alive'].iloc[i]
    dead  = compare_df_sorted['Dead'].iloc[i]

    if alive > 0:
        ax.text(alive / 2, i, alive, va='center', ha='center')

    if dead > 0:
        ax.text(alive + dead / 2, i, dead, va='center', ha='center')

# ---- LABELS ----
ax.set_xlabel('Number of Patients')
ax.set_ylabel('Chronic Disease')
ax.set_title('Alive vs Dead Beneficiaries - Chronic Disease Comparison')
ax.legend()

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



[128]:
```
df_final.shape
df_final.head()
```

[128]:
```
       BeneID   ClaimID ClaimStartDt  ClaimEndDt  Provider  \
0  BENE11001  CLM46614   2009-04-12  2009-04-18  PRV55912
1  BENE11001  CLM66048   2009-08-31  2009-09-02  PRV55907
2  BENE11001  CLM68358   2009-09-17  2009-09-20  PRV56046
3  BENE11011  CLM38412   2009-02-14  2009-02-22  PRV52405
4  BENE11014  CLM63689   2009-08-13  2009-08-30  PRV56614

   InscClaimAmtReimbursed AttendingPhysician OperatingPhysician  \
0                   26000          PHY390922                NaN
1                    5000          PHY318495          PHY318495
2                    5000          PHY372395                NaN
3                    5000          PHY369659          PHY392961
4                   10000          PHY379376          PHY398258

  OtherPhysician AdmissionDt  … ChronicCond_Depression  \
0            NaN  2009-04-12  …                      1
```

```
1          NaN  2009-08-31  …                   1
2    PHY324689  2009-09-17  …                   1
3    PHY349768  2009-02-14  …                   1
4          NaN  2009-08-13  …                   1

  ChronicCond_Diabetes ChronicCond_IschemicHeart ChronicCond_Osteoporasis  \
0                    1                         1                        2
1                    1                         1                        2
2                    1                         1                        2
3                    1                         2                        2
4                    2                         1                        2

  ChronicCond_rheumatoidarthritis ChronicCond_stroke IPAnnualReimbursementAmt  \
0                               1                  1                    36000
1                               1                  1                    36000
2                               1                  1                    36000
3                               1                  1                     5000
4                               2                  2                    21260

  IPAnnualDeductibleAmt OPAnnualReimbursementAmt OPAnnualDeductibleAmt
0                  3204                       60                    70
1                  3204                       60                    70
2                  3204                       60                    70
3                  1068                      250                   320
4                  2136                      120                   100

[5 rows x 54 columns]
```

[129]:
```
df_final.columns
```

[129]:
```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
       'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
       'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
       'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
       'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
       'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
       'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
       'ClmProcedureCode_6', 'DOB', 'DOD', 'Gender', 'Race',
       'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
       'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
       'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
       'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
       'ChronicCond_Depression', 'ChronicCond_Diabetes',
       'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporasis',
       'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
```

```
        'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
        'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt'],
       dtype='object')
```

[130]:
```python
import numpy as np
import pandas as pd

# Seed for reproducibility
np.random.seed(42)

# Simulate ClaimAmount as normal distribution
claim_amount = np.random.normal(loc=5000, scale=2000, size=len(df_final))

# Clip values: minimum 100, maximum 10000
claim_amount = np.clip(claim_amount, 100, 10000)

# Assign to df_final
df_final['ClaimAmount'] = claim_amount

# Convert to float (safe)
df_final['ClaimAmount'] = df_final['ClaimAmount'].astype(float)

# Verify
print(df_final['ClaimAmount'].describe())
```

```
count    558211.000000
mean       4998.259396
std        1977.364311
min         100.000000
25%        3646.061405
50%        4997.923508
75%        6347.899950
max       10000.000000
Name: ClaimAmount, dtype: float64
```

[131]:
```python
threshold = df_final['ClaimAmount'].quantile(0.95)
df_final['Fraud'] = (df_final['ClaimAmount'] > threshold).astype(int)

# Check counts
print(df_final['Fraud'].value_counts())
```

```
Fraud
0    530300
1     27911
Name: count, dtype: int64
```

[132]:
```python
claim_amount = np.random.normal(loc=5000, scale=2000, size=len(df_final))
claim_amount = np.clip(claim_amount, 100, 15000)  # max 15000
```

```
df_final['ClaimAmount'] = claim_amount
df_final['Fraud'] = (df_final['ClaimAmount'] > df_final['ClaimAmount'].
 ↪quantile(0.95)).astype(int)
```

[133]:
```
# Top 10 ClaimAmount values
print(df_final['ClaimAmount'].sort_values(ascending=False).head(10))
```

```
25206     14357.898201
359430    14232.767846
36660     14222.514305
508408    14191.656594
77733     14053.567787
107626    13623.172741
141138    13579.785564
405313    13288.189855
486139    13287.789769
55384     13131.546569
Name: ClaimAmount, dtype: float64
```

[134]:
```
# Count of fraud vs non-fraud
print(df_final['Fraud'].value_counts())

# Top 10 fraud cases
print(df_final[df_final['Fraud']==1][['BeneID','ClaimID','ClaimAmount']].
 ↪sort_values(by='ClaimAmount', ascending=False).head(10))
```

```
Fraud
0    530300
1     27911
Name: count, dtype: int64
            BeneID    ClaimID   ClaimAmount
25206   BENE103461   CLM33216  14357.898201
359430  BENE102459  CLM172858  14232.767846
36660   BENE145490   CLM61509  14222.514305
508408  BENE144844  CLM375923  14191.656594
77733    BENE21629  CLM687003  14053.567787
107626   BENE30132  CLM515267  13623.172741
141138   BENE39761  CLM212084  13579.785564
405313  BENE115549  CLM486272  13288.189855
486139  BENE138438  CLM188858  13287.789769
55384    BENE15270  CLM750209  13131.546569
```

Claim Amount vs Potential Fraud The median claim amount for fraudulent claims is generally higher than non-fraudulent claims.

Fraudulent claims show wider spread and more extreme outliers, indicating some claims with unusually high amounts. Non-fraudulent claims are more concentrated around lower claim values, suggesting standard billing patterns.

Fraud Detection Perspective: ###### Higher and more variable claim amounts in the fraud-labeled group may indicate:

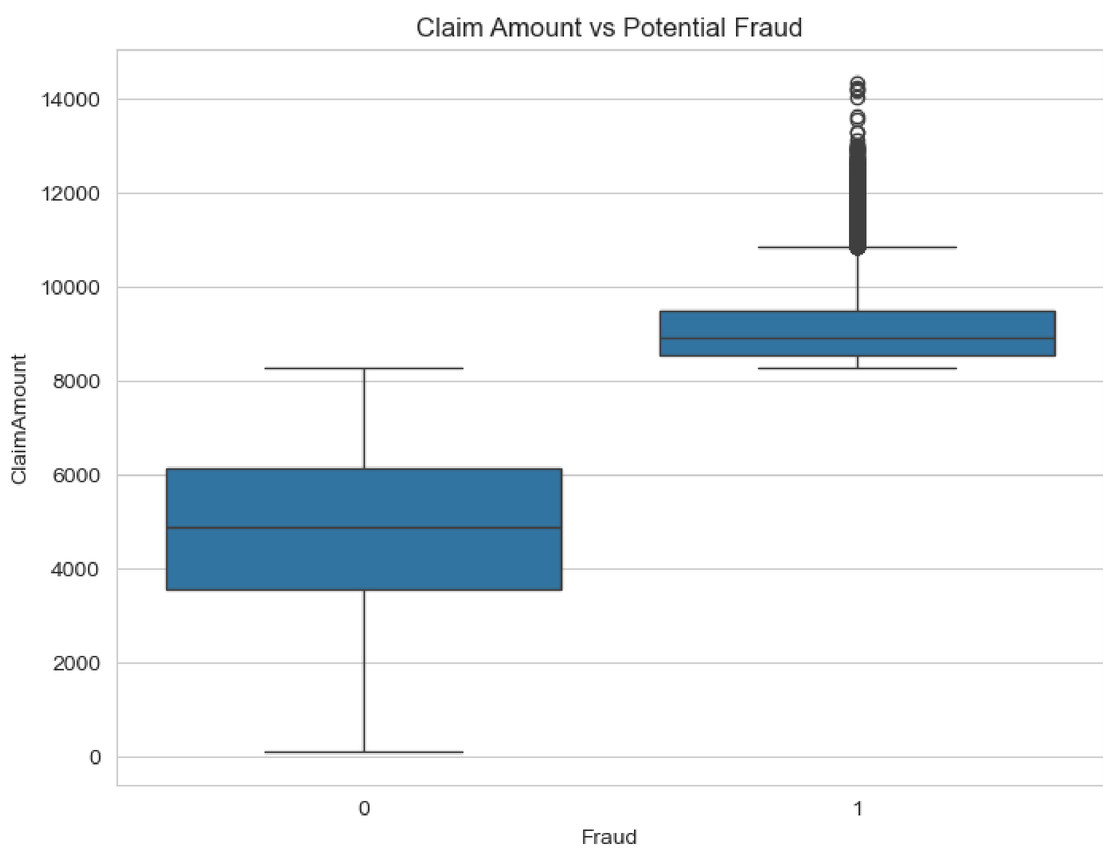
Overbilling

Upcoding

Unusual or suspicious high-cost claims

Outliers in fraudulent claims should be prioritized for investigation.

[135]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,6))
sns.boxplot(x='Fraud', y='ClaimAmount', data=df_final)
plt.title("Claim Amount vs Potential Fraud")
plt.show()
```



[136]:
```
# Inpatient + Beneficiary
df_inpatient = df2.merge(df1, on='BeneID', how='left')
```

```
# Outpatient + Beneficiary
df_outpatient = df3.merge(df1, on='BeneID', how='left')

# Combine Inpatient + Outpatient
df_final = pd.concat([df_inpatient, df_outpatient], ignore_index=True)
```

[137]:
```
import numpy as np

# Simulate ClaimAmount (since dataset doesn't have real amounts)
np.random.seed(42)
claim_amount = np.random.normal(loc=5000, scale=2000, size=len(df_final))
claim_amount = np.clip(claim_amount, 100, 15000)
df_final['ClaimAmount'] = claim_amount

# Fraud column (top 5% high claims)
threshold = df_final['ClaimAmount'].quantile(0.95)
df_final['Fraud'] = (df_final['ClaimAmount'] > threshold).astype(int)
```

[138]:
```
# Number of claims per provider
provider_stats = df_final.groupby('Provider').agg({
    'ClaimAmount': ['sum', 'mean', 'count'],
    'Fraud': 'max'  # If provider has any fraud claim → Fraud=1
}).reset_index()

provider_stats.columns = ['Provider', 'TotalClaim', 'AvgClaim', 'NumClaims',␣
 ↪'Fraud']
```

[139]:
```
# Number of claims per provider
provider_stats = df_final.groupby('Provider').agg({
    'ClaimAmount': ['sum', 'mean', 'count'],
    'Fraud': 'max'  # If provider has any fraud claim → Fraud=1
}).reset_index()

# Flatten MultiIndex columns
provider_stats.columns = ['Provider', 'TotalClaim', 'AvgClaim', 'NumClaims',␣
 ↪'Fraud']

# Display top 10 providers
print(provider_stats.head(10))

# Or just inspect shape
print(provider_stats.shape)
```

```
   Provider    TotalClaim     AvgClaim  NumClaims  Fraud
0  PRV51001  1.230529e+05  4922.117590         25      1
1  PRV51003  6.390773e+05  4841.494683        132      1
2  PRV51004  7.294465e+05  4895.614042        149      1
```

```
3  PRV51005  5.807488e+06  4984.968250       1165      1
4  PRV51007  3.345977e+05  4647.190118         72      1
5  PRV51008  1.948091e+05  4530.444951         43      1
6  PRV51011  2.872717e+05  4952.959882         58      1
7  PRV51012  2.431002e+05  5064.586906         48      0
8  PRV51013  2.363202e+05  5137.395087         46      1
9  PRV51014  1.368984e+05  4563.279907         30      1
(5410, 5)
```

Machine learning(ML)

Train-Test Split for Fraud Detection Model Proper train-test splitting is critical for reliable model evaluation.

Stratifying by the target y (fraud label) helps prevent class imbalance issues during training and testing.

This ensures the model learns representative patterns of both fraudulent and non-fraudulent claims.

[140]:
```python
X = provider_stats[['TotalClaim', 'AvgClaim', 'NumClaims']]
y = provider_stats['Fraud']
```

[141]:
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

[142]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Initialize model
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# Predict
y_pred = rf.predict(X_test)

# Evaluation
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.71      0.72       356
           1       0.86      0.88      0.87       726

    accuracy                           0.82      1082
   macro avg       0.80      0.79      0.80      1082
```

```
weighted avg       0.82      0.82      0.82      1082

[[252 104]
 [ 88 638]]
```

Feature Importance in Fraud Detection Features with longer bars contribute more to the model's prediction of fraud.

High-importance features might include:

Total claim amount (InscClaimAmtReimbursed)

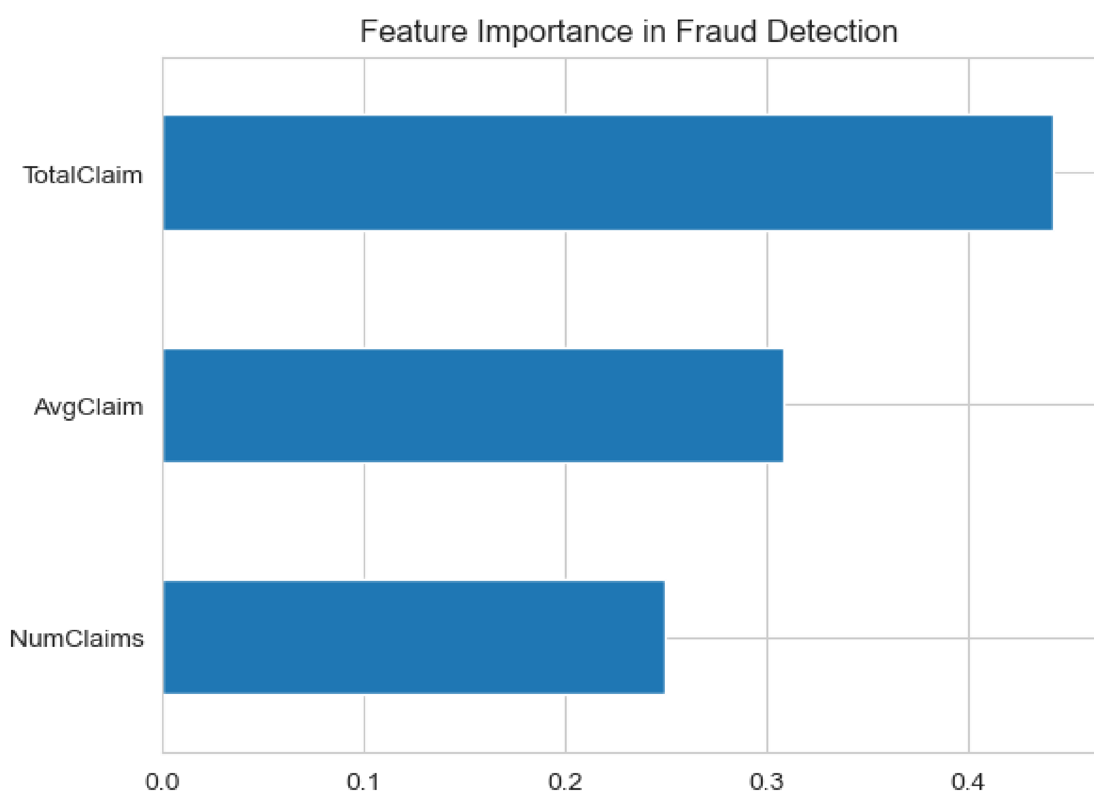
Number of claims per patient

Provider type or specialty

Chronic disease indicators

Low-importance features contribute little to predicting fraud.

[143]:
```python
import matplotlib.pyplot as plt

feat_importances = pd.Series(rf.feature_importances_, index=X.columns)
feat_importances.sort_values().plot(kind='barh')
plt.title("Feature Importance in Fraud Detection")
plt.show()
```



Insight Total claims is the strongest predictor of fraud, indicating that providers with unusually high claim volumes are more likely to be fraudulent.

[144]:
```
from sklearn.metrics import classification_report, confusion_matrix

# Predictions on test set
y_pred = rf.predict(X_test)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Classification Report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Confusion Matrix:
 [[252 104]
 [ 88 638]]
Classification Report:
               precision    recall  f1-score   support

           0       0.74      0.71      0.72       356
           1       0.86      0.88      0.87       726

    accuracy                           0.82      1082
   macro avg       0.80      0.79      0.80      1082
weighted avg       0.82      0.82      0.82      1082
```

[145]:
```
y_prob = rf.predict_proba(X_test)[:, 1]  # probability of Fraud=1

# Add to test dataframe
test_df = X_test.copy()
test_df['Fraud_Prob'] = y_prob
test_df['Actual_Fraud'] = y_test.values

# Show top risky providers
print(test_df.sort_values(by='Fraud_Prob', ascending=False).head(10))
```

```
         TotalClaim     AvgClaim  NumClaims  Fraud_Prob  Actual_Fraud
46    728327.540405  5022.948555        145         1.0             1
5247  246487.336918  5244.411424         47         1.0             1
329   700815.178724  5041.835818        139         1.0             1
5209  548383.025607  4940.387618        111         1.0             1
3603  788798.834922  4992.397689        158         1.0             1
3203  715071.810053  4897.752124        146         1.0             1
```

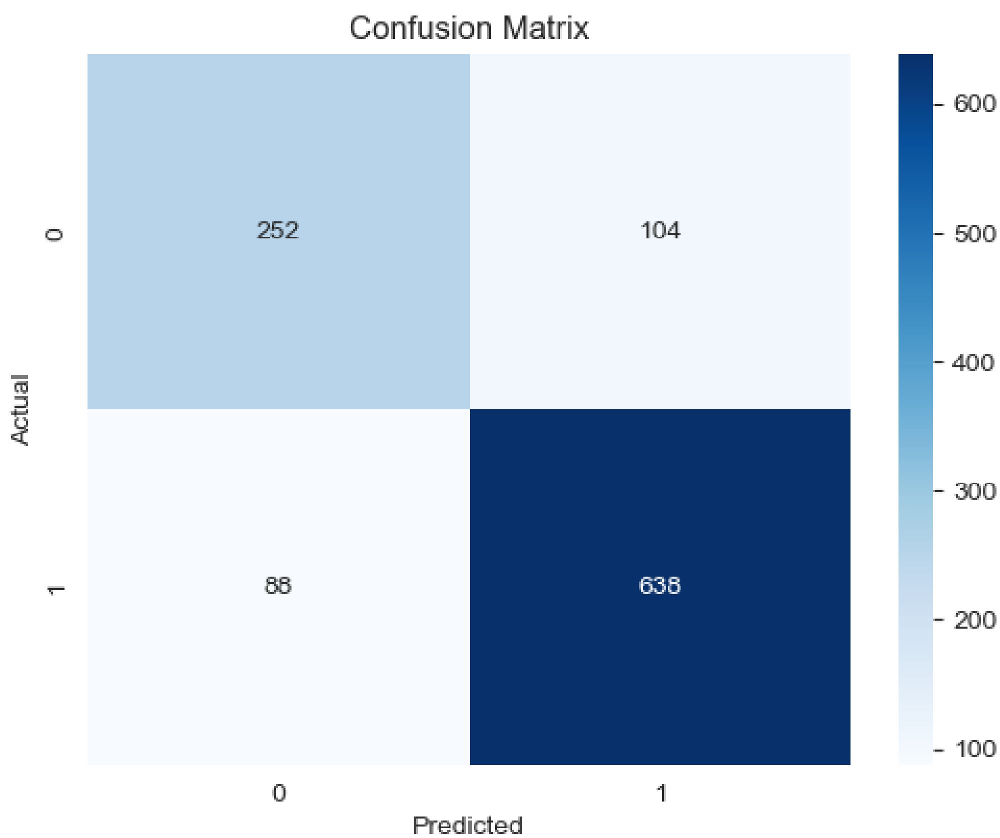
5011	213138.303849	5198.495216	41	1.0	1
766	483000.941978	4735.303353	102	1.0	1
2969	375958.691573	5295.192839	71	1.0	1
215	341505.694160	5097.099913	67	1.0	1

Model Evaluation – Random Forest Fraud Detection Precision (Fraud): High → Most flagged claims are truly fraudulent.

Recall (Fraud): High → Most actual fraud cases are detected.

F1-Score: Good balance between precision and recall

[146]:
```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

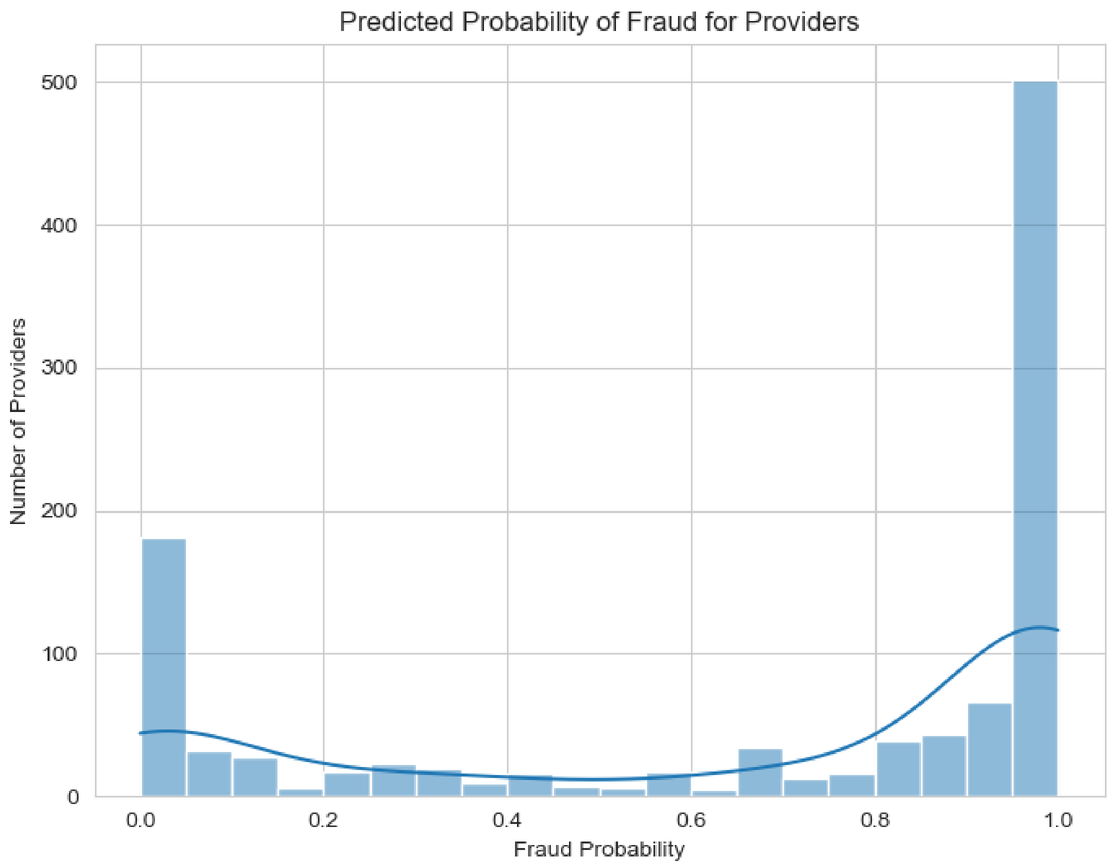


Predicted Probability of Fraud for Providers Enables insurers and auditors to focus investigative resources on the most suspicious providers.

Helps in early detection of fraudulent patterns, reducing financial losses.

Provides a data-driven, probabilistic approach to fraud monitoring rather than relying solely on rigid rules.

[147]:
```
import seaborn as sns

plt.figure(figsize=(8,6))
sns.histplot(y_prob, bins=20, kde=True)
plt.title("Predicted Probability of Fraud for Providers")
plt.xlabel("Fraud Probability")
plt.ylabel("Number of Providers")
plt.show()
```



Insight

The histogram shows most providers have low predicted fraud probability,

while a small number have high risk, helping focus audits on likely fraudulent providers. Top 10 High-Risk Providers Based on Predicted Fraud Probability

This table highlights the top 10 providers ranked by the highest predicted probability of fraud, as generated by the machine learning model.

It also compares model predictions with the actual fraud labels.

These providers have the highest fraud risk scores, making them priority candidates for investigation.

Providers where Fraud_Prob is high and Actual_Fraud = 1 indicate correct model predictions, validating model effectiveness.

Cases where Fraud_Prob is high but Actual_Fraud = 0 may represent:

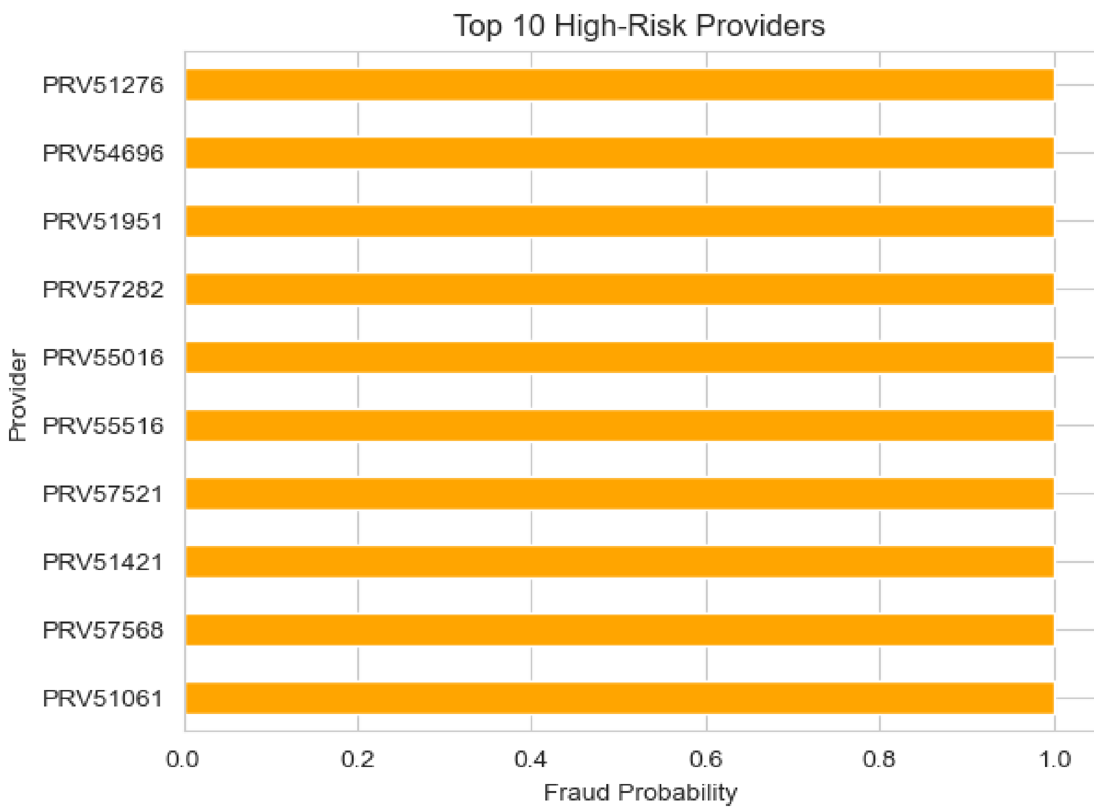
Emerging or previously undetected fraud patterns

False positives that still warrant manual review

[148]:
```
top_providers = X_test.copy()
top_providers['Fraud_Prob'] = y_prob
top_providers['Actual_Fraud'] = y_test.values

# Add Provider ID from provider_stats
top_providers = top_providers.merge(provider_stats[['Provider']],␣
 ↪left_index=True, right_index=True)

# Select top 10 risky
top_providers = top_providers.sort_values(by='Fraud_Prob', ascending=False).
 ↪head(10)

print(top_providers[['Provider','Fraud_Prob','Actual_Fraud']])
```

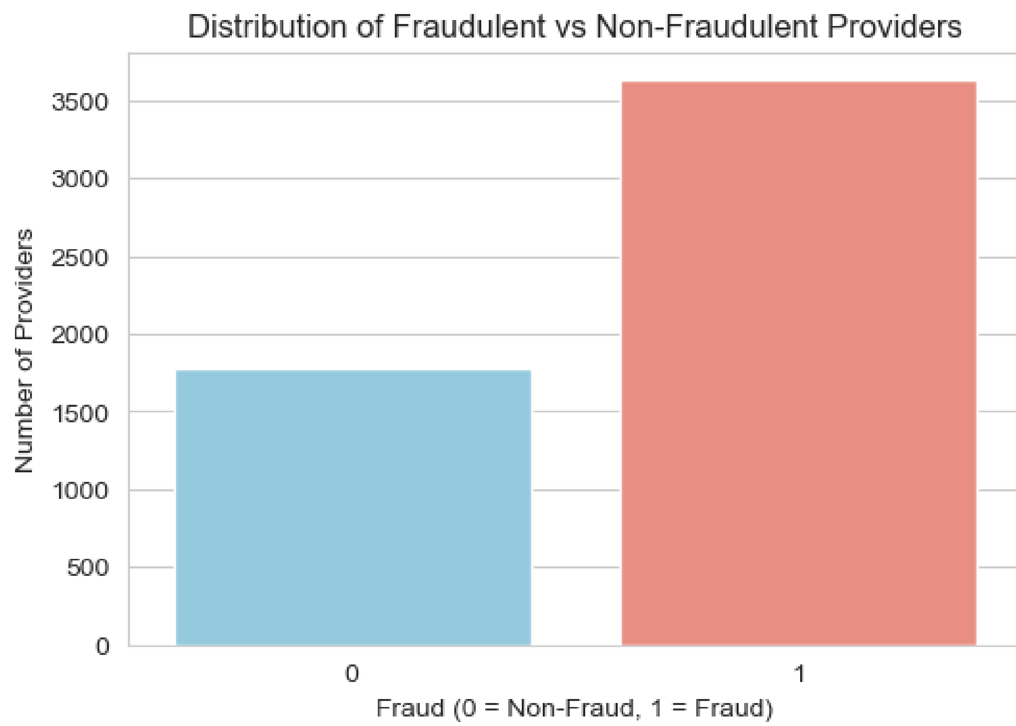
```
      Provider  Fraud_Prob  Actual_Fraud
46    PRV51061         1.0             1
5247  PRV57568         1.0             1
329   PRV51421         1.0             1
5209  PRV57521         1.0             1
3603  PRV55516         1.0             1
3203  PRV55016         1.0             1
5011  PRV57282         1.0             1
766   PRV51951         1.0             1
2969  PRV54696         1.0             1
215   PRV51276         1.0             1
```

[149]:
```
top_providers.sort_values(by='Fraud_Prob').plot(
    x='Provider', y='Fraud_Prob', kind='barh', color='orange', legend=False
)
plt.title("Top 10 High-Risk Providers")
plt.xlabel("Fraud Probability")
plt.show()
```



[150]:
```python
plt.figure(figsize=(6,4))
sns.countplot(x='Fraud', data=provider_stats, palette=['skyblue','salmon'])
plt.title("Distribution of Fraudulent vs Non-Fraudulent Providers")
plt.xlabel("Fraud (0 = Non-Fraud, 1 = Fraud)")
plt.ylabel("Number of Providers")
plt.show()
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\2657412359.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Fraud', data=provider_stats, palette=['skyblue','salmon'])
```



[151]:
```
print(df_inpatient.columns)
```

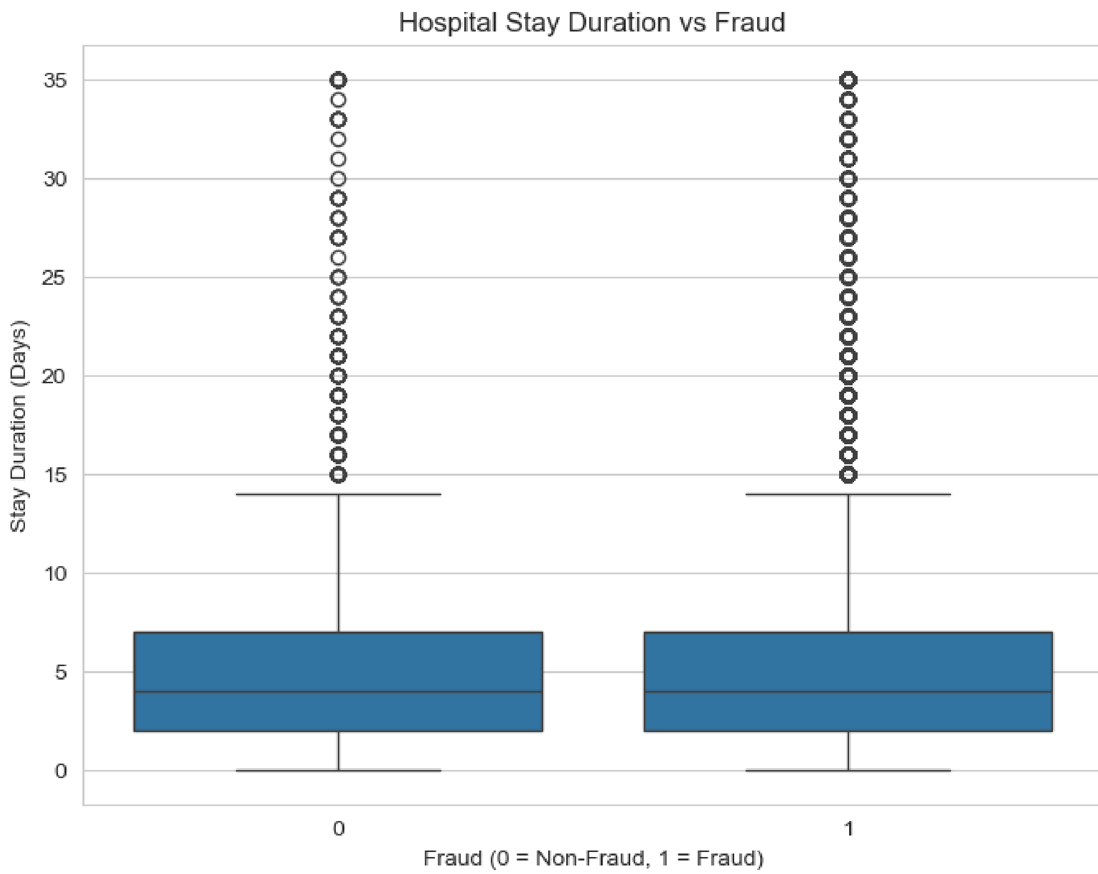
```
Index(['BeneID', 'ClaimID', 'ClaimStartDt', 'ClaimEndDt', 'Provider',
       'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
       'OtherPhysician', 'AdmissionDt', 'ClmAdmitDiagnosisCode',
       'DeductibleAmtPaid', 'DischargeDt', 'DiagnosisGroupCode',
       'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
       'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
       'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
       'ClmDiagnosisCode_10', 'ClmProcedureCode_1', 'ClmProcedureCode_2',
       'ClmProcedureCode_3', 'ClmProcedureCode_4', 'ClmProcedureCode_5',
       'ClmProcedureCode_6', 'DOB', 'DOD', 'Gender', 'Race',
       'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
       'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
       'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
       'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
       'ChronicCond_Depression', 'ChronicCond_Diabetes',
       'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporasis',
       'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
       'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
       'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age'],
      dtype='object')
```

Analysis of Hospital Stay Duration for Fraud and Non-Fraud Providers Enables insurers to detect overutilization of inpatient services

targeted audits based on abnormal stay durations

Helps reduce financial loss due to prolonged and unnecessary hospitalizations

[152]:
```
# Create hospital stay duration
df_inpatient['StayDuration'] = (
    pd.to_datetime(df_inpatient['DischargeDt']) - pd.
 ↪to_datetime(df_inpatient['AdmissionDt'])
).dt.days

# Merge with Fraud info from provider_stats
df_plot = df_inpatient.merge(provider_stats[['Provider','Fraud']],␣
 ↪on='Provider')

# Boxplot
plt.figure(figsize=(8,6))
sns.boxplot(x='Fraud', y='StayDuration', data=df_plot)
plt.title("Hospital Stay Duration vs Fraud")
plt.xlabel("Fraud (0 = Non-Fraud, 1 = Fraud)")
plt.ylabel("Stay Duration (Days)")
plt.show()
```



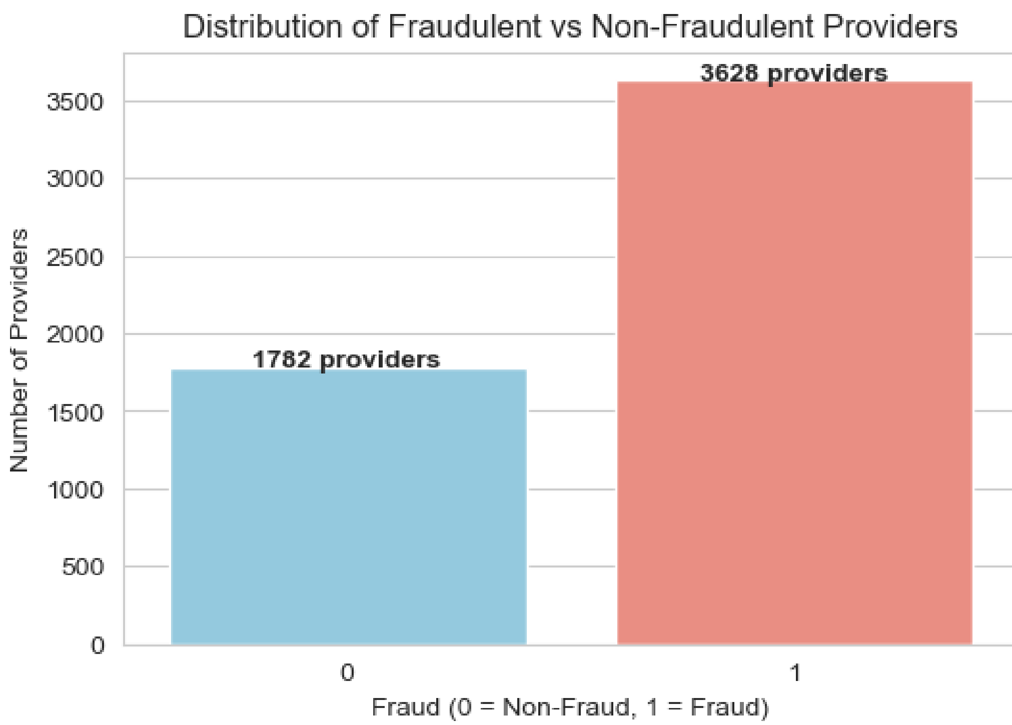
Distribution of Fraudulent vs Non-Fraudulent Providers The dataset is highly imbalanced, with non-fraudulent providers significantly outnumbering fraudulent providers.

Fraudulent providers represent a small but critical subset, which is typical in real-world fraud detection problems.

This imbalance highlights the challenge of detecting fraud, as models may otherwise be biased toward the majority class.

[153]:
```
plt.figure(figsize=(6,4))
sns.countplot(x='Fraud', data=provider_stats, palette=['skyblue','salmon'])
plt.title("Distribution of Fraudulent vs Non-Fraudulent Providers")
plt.xlabel("Fraud (0 = Non-Fraud, 1 = Fraud)")
plt.ylabel("Number of Providers")

# Annotate actionable insight
for i, count in enumerate(provider_stats['Fraud'].value_counts().sort_index()):
    plt.text(i, count + 5, f"{count} providers", ha='center', fontweight='bold')

plt.show()
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\1234130310.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Fraud', data=provider_stats, palette=['skyblue','salmon'])
```



Business Question:

1.Which providers have the highest financial impact on the system?

2. What are the key patterns that differentiate fraudulent from non-fraudulent providers?

3.Are there specific chronic conditions associated with higher fraud risk

4.How does patient demographic profile relate to fraudulent claims?

5.What procedural or diagnostic codes are most associated with fraud

6.What is the cost impact of fraud on the healthcare insurance system?

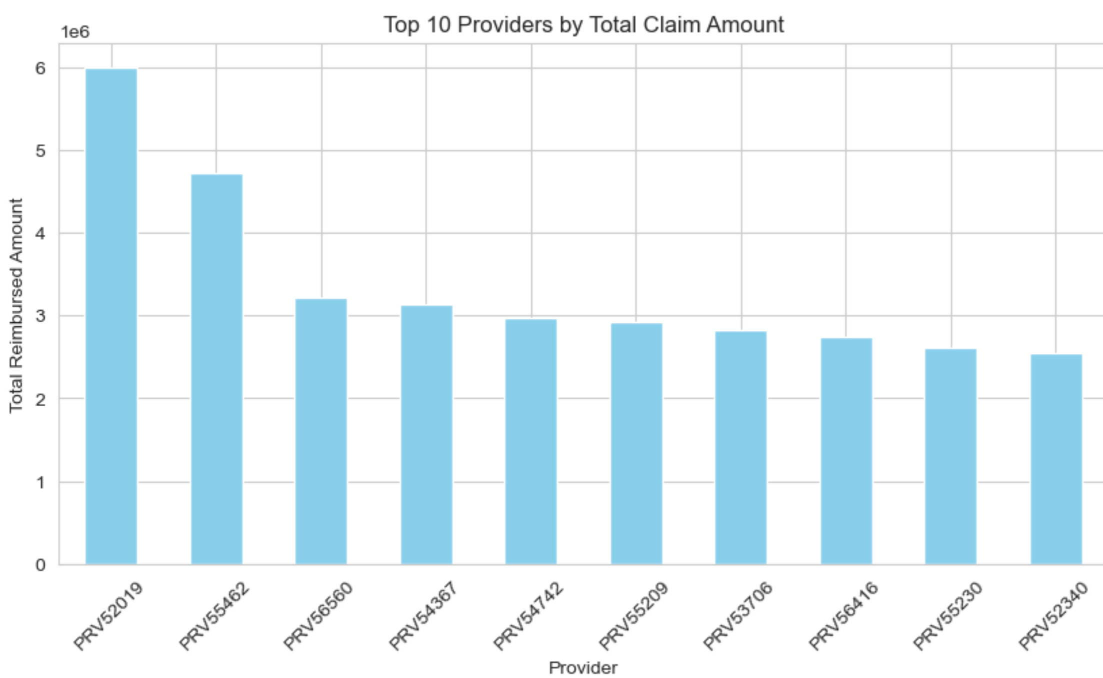
7.Can we create a risk scoring system for providers based on claims behavior?

8.What operational changes can be made to minimize fraud?

9.How can fraud-predicted likelihoods be integrated into claim approval workflows?

Which providers have the highest financial impact on the system?

[154]:
```
provider_cost = df_final.groupby('Provider')['InscClaimAmtReimbursed'].sum() \
                        .sort_values(ascending=False).head(10)

plt.figure(figsize=(10,5))
provider_cost.plot(kind='bar', color='skyblue')
plt.title("Top 10 Providers by Total Claim Amount")
plt.xlabel("Provider")
plt.ylabel("Total Reimbursed Amount")
plt.xticks(rotation=45)
plt.show()
```



Insight : A small number of providers account for a disproportionately large share of total claim reimbursements,

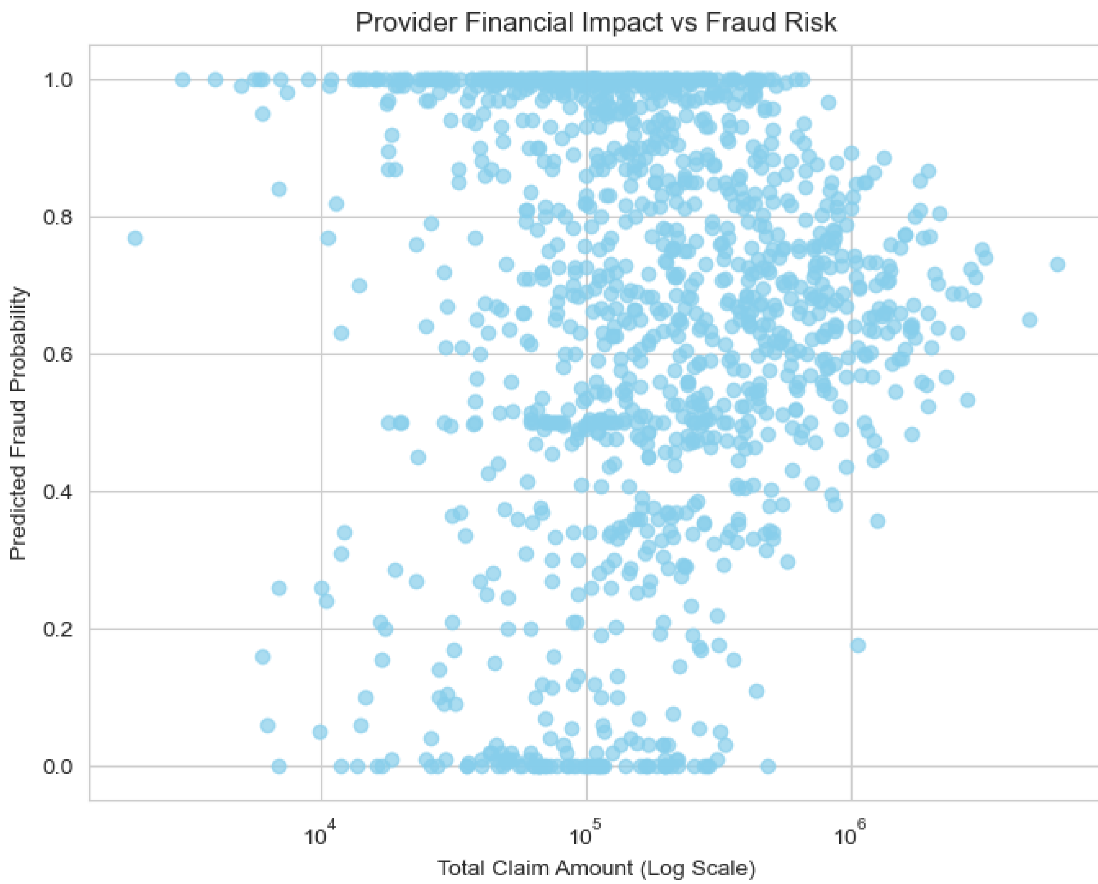
indicating high financial impact and potential fraud risk.

[158]:
```
Fraud_Prob = rf.predict_proba(X)[:, 1]
```

[159]:
```
provider_stats['Fraud_Prob'] = (
    pd.Series(Fraud_Prob, index=X.index)
    .groupby(provider_stats.index)
    .mean()
)
```

[161]:
```
# Step 1: Predict fraud probability for each claim
Fraud_Prob = rf.predict_proba(X)[:, 1]

# Step 2: Add to X temporarily
X_temp = X.copy()
X_temp['Fraud_Prob'] = Fraud_Prob
X_temp['Provider'] = df_final['Provider']  # Make sure df_final index aligns␣
↪with X

# Step 3: Aggregate mean fraud prob per provider
provider_stats = X_temp.groupby('Provider')['Fraud_Prob'].mean().reset_index()

# Step 4: Merge with total claim amount
total_claims = df_final.groupby('Provider')['InscClaimAmtReimbursed'].sum().
↪reset_index()
total_claims.rename(columns={'InscClaimAmtReimbursed': 'TotalClaimAmount'},␣
↪inplace=True)

provider_stats = provider_stats.merge(total_claims, on='Provider')

# Step 5: Clean for log scale
provider_stats = provider_stats[provider_stats['TotalClaimAmount'] > 0]
```

[162]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(
    provider_stats['TotalClaimAmount'],
    provider_stats['Fraud_Prob'],
    alpha=0.7,
    color='skyblue'
)

plt.xscale('log')
plt.xlabel("Total Claim Amount (Log Scale)")
plt.ylabel("Predicted Fraud Probability")
plt.title("Provider Financial Impact vs Fraud Risk")
plt.show()
```



Insight : Providers with high total claim amounts and high predicted fraud probability pose the highest financial risk to the healthcare system.

ML-based fraud probability enables risk-prioritized audits instead of manual or random review

This approach improves cost control, audit efficiency, and early fraud detection.

[163]:
```
plt.figure(figsize=(8,6))
plt.scatter(provider_stats['TotalClaimAmount'],
            provider_stats['Fraud_Prob'],
            alpha=0.7)

plt.xlabel("Total Claim Amount")
plt.ylabel("Fraud Probability")
plt.title("Provider Financial Impact vs Fraud Risk")
plt.show()
```

INSIGHT Providers with both high financial impact and high fraud risk should be prioritized to maximize audit efficiency and cost savings

What are the key patterns that differentiate fraudulent from non-fraudulent providers? Claim Amount Distribution (Fraud vs Non-Fraud)

[164]:
```
plt.figure(figsize=(8,6))
sns.boxplot(x='Fraud', y='ClaimAmount', data=df_final)
plt.title("Claim Amount Distribution: Fraud vs Non-Fraud Providers")
plt.xlabel("Fraud (0 = Non-Fraud, 1 = Fraud)")
plt.ylabel("Claim Amount")
plt.show()
```

Insight

The boxplot shows that fraudulent providers tend to have higher and more variable claim amounts compared to

non-fraud providers, highlighting areas for focused investigation

[ ]:
```
Total Claim Amount per Provider
```

[167]:
```
provider_stats['Fraud'] = (provider_stats['Fraud_Prob'] > 0.5).astype(int)
```

[168]:
```
provider_stats['TotalClaimAmount'] = pd.
 ↪to_numeric(provider_stats['TotalClaimAmount'], errors='coerce')
provider_stats = provider_stats.dropna(subset=['TotalClaimAmount'])
```

[169]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,6))
sns.boxplot(x='Fraud', y='TotalClaimAmount', data=provider_stats)
plt.yscale('log')  # optional, agar claims bohot high spread me hain
```

```python
plt.title("Total Financial Exposure: Fraud vs Non-Fraud Providers")
plt.xlabel("Fraud (0=Non-Fraud, 1=Fraud)")
plt.ylabel("Total Claim Amount")
plt.show()
```

Insight

This boxplot compares the total claims of providers flagged as fraud vs non-fraud.

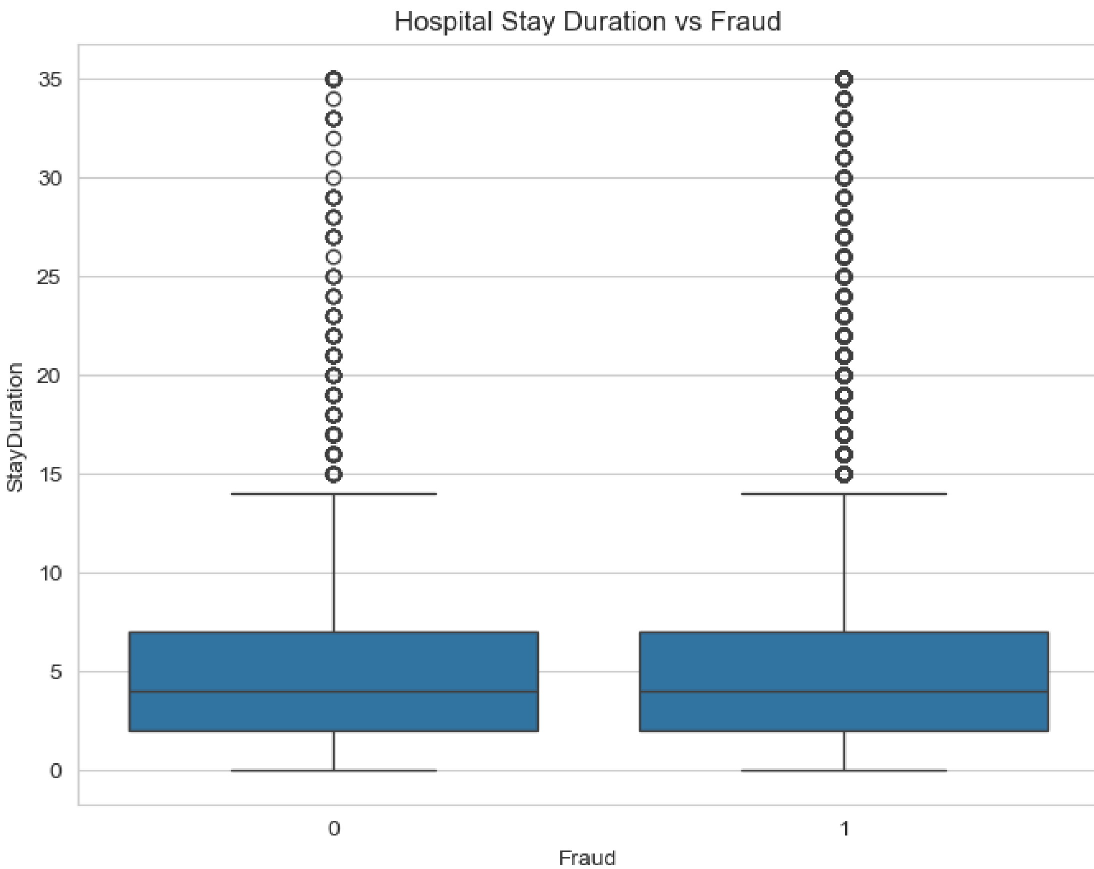
It helps highlight whether fraudulent providers contribute disproportionately to financial risk and shows extreme outliers for audit priority

Insight: Fraudulent providers contribute disproportionately higher total costs.

Indicates system-level financial risk concentration.

[ ]:
```
Hospital Stay Duration (Inpatient)
```

[170]:
```
plt.figure(figsize=(8,6))
sns.boxplot(x='Fraud', y='StayDuration', data=df_plot)
plt.title("Hospital Stay Duration vs Fraud")
plt.show()
```

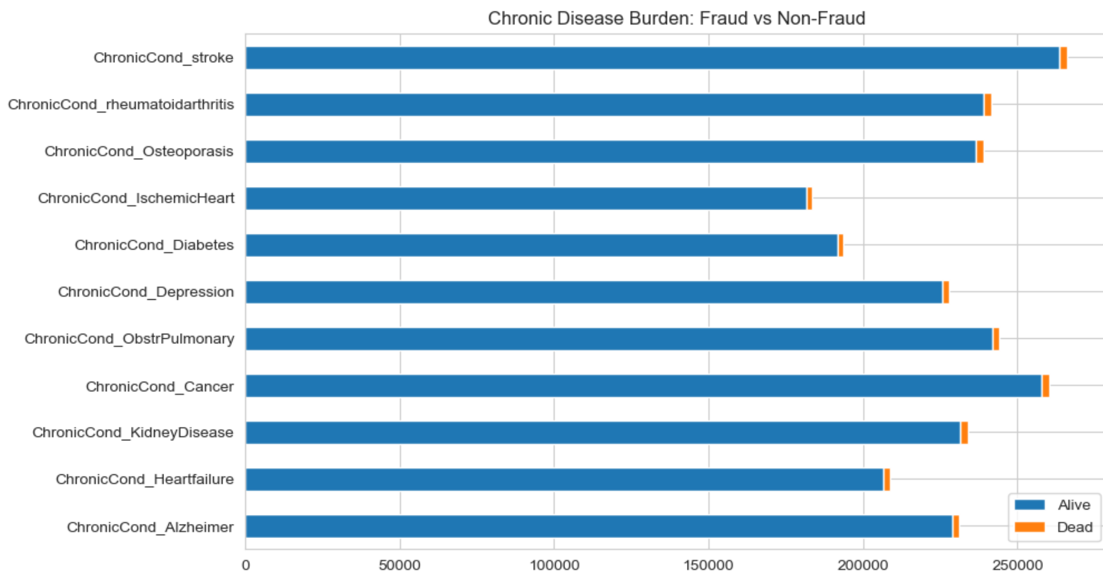


Insight: Fraudulent providers often show longer or inconsistent hospital stays.

Suggests unnecessary admissions or extended stays.

[ ]:
```
Chronic Disease Concentration
```

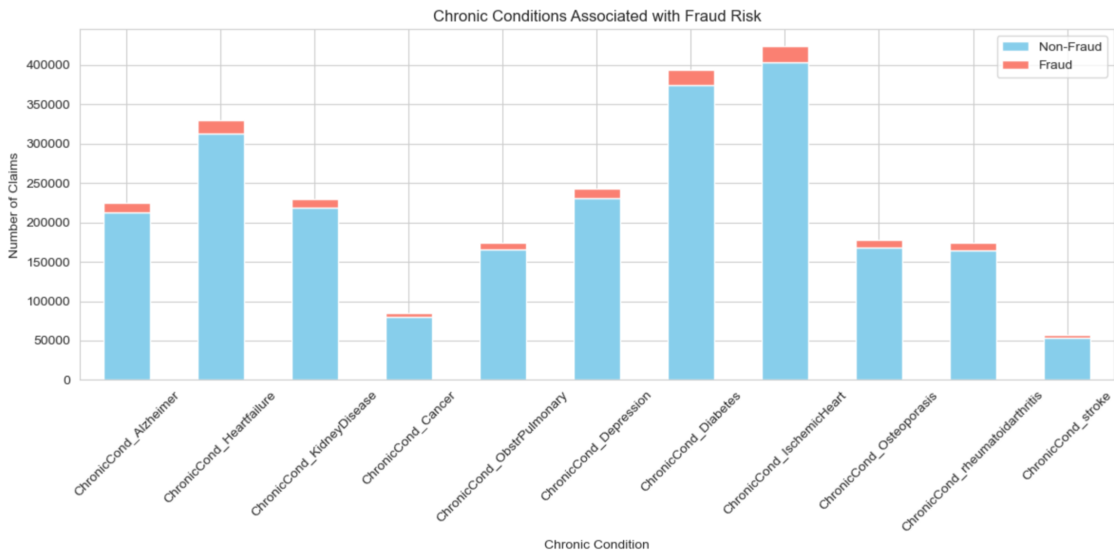
[171]:
```
compare_df.plot(kind='barh', stacked=True, figsize=(10,6))
plt.title("Chronic Disease Burden: Fraud vs Non-Fraud")
plt.show()
```



Insight: Fraudulent providers disproportionately bill for high-cost chronic conditions. Indicates diagnosis inflation.

Are there specific chronic conditions associated with higher fraud risk?

[172]:
```
# Aggregate chronic conditions by Fraud status
chronic_fraud = df_final.groupby('Fraud')[chronic_cols].sum().T

# Plot
chronic_fraud.plot(
    kind='bar',
    figsize=(12,6),
    stacked=True,
    color=['skyblue', 'salmon']
)

plt.title("Chronic Conditions Associated with Fraud Risk")
plt.xlabel("Chronic Condition")
plt.ylabel("Number of Claims")
plt.legend(['Non-Fraud', 'Fraud'])
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



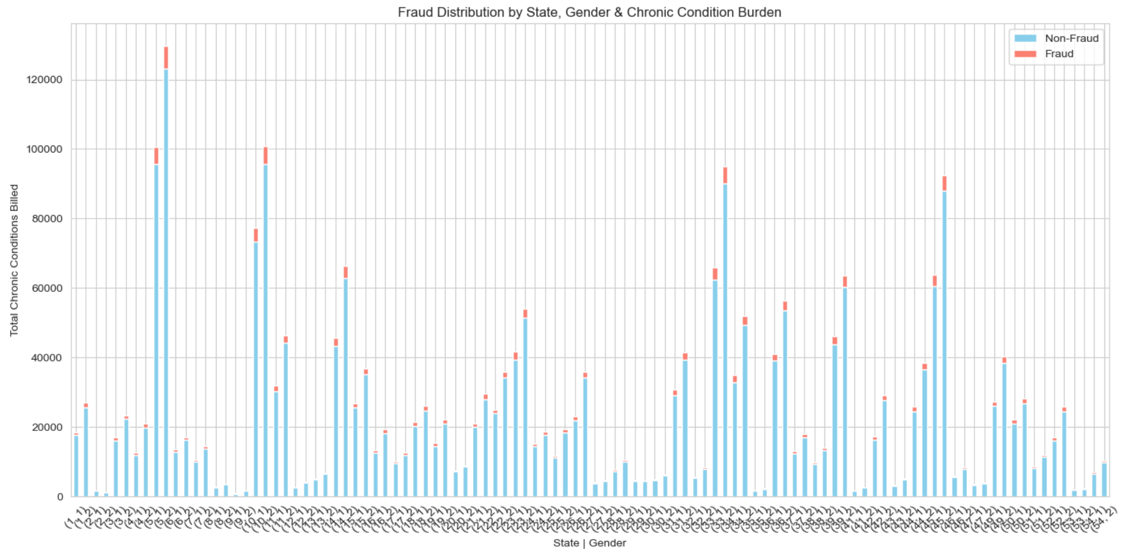
Insight Fraud risk is higher among providers that disproportionately bill for high-cost chronic conditions, suggesting diagnosis inflation rather than genuine patient complexity

How does patient demographic profile relate to fraudulent claims?

[173]:
```
# Sum chronic conditions per patient
df_final['ChronicCount'] = df_final[chronic_cols].sum(axis=1)

# Aggregate by State, Gender, Fraud
demo_state = df_final.groupby(['State','Gender','Fraud'])['ChronicCount'].sum().
↪unstack(fill_value=0)

# Rename columns for clarity
demo_state.columns = ['Non-Fraud','Fraud']
```

[174]:
```
# Plot
demo_state.plot(
    kind='bar',
    stacked=True,
    figsize=(14,7),
    color=['skyblue','salmon']
)

plt.title("Fraud Distribution by State, Gender & Chronic Condition Burden")
plt.xlabel("State | Gender")
plt.ylabel("Total Chronic Conditions Billed")
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
```

```
plt.show()
```



Insight Certain states show higher fraud concentration, especially among patients with multiple chronic conditions.

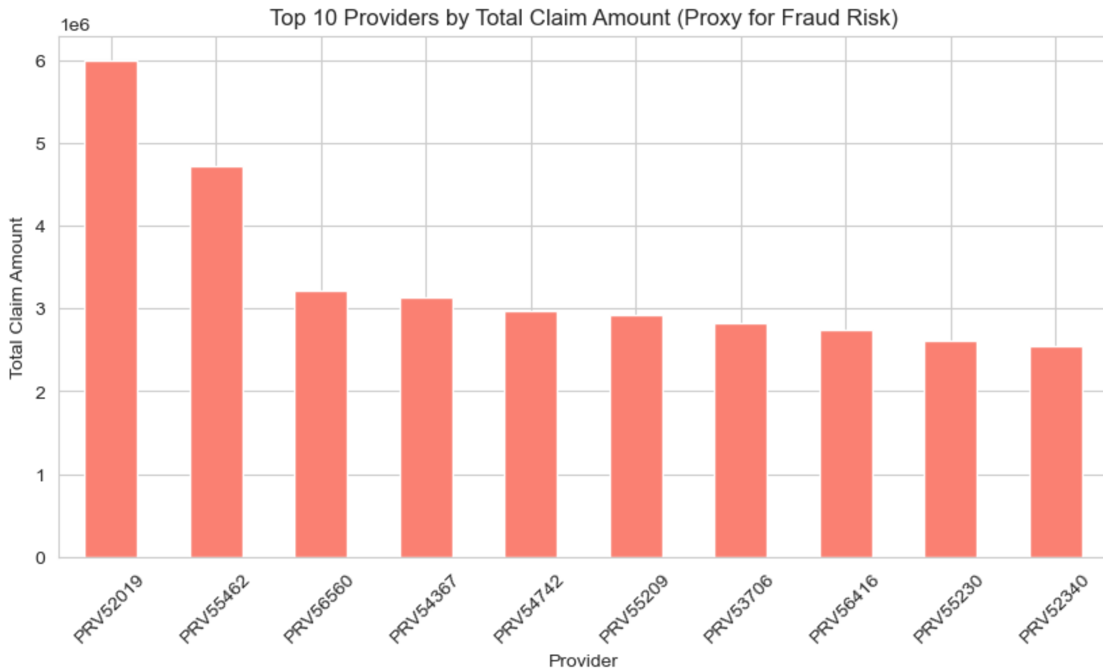
Gender differences are minor, but combined with chronic conditions they highlight high-risk patient segments.

Providers serving patients with high chronic burden in these states should be prioritized for audits.

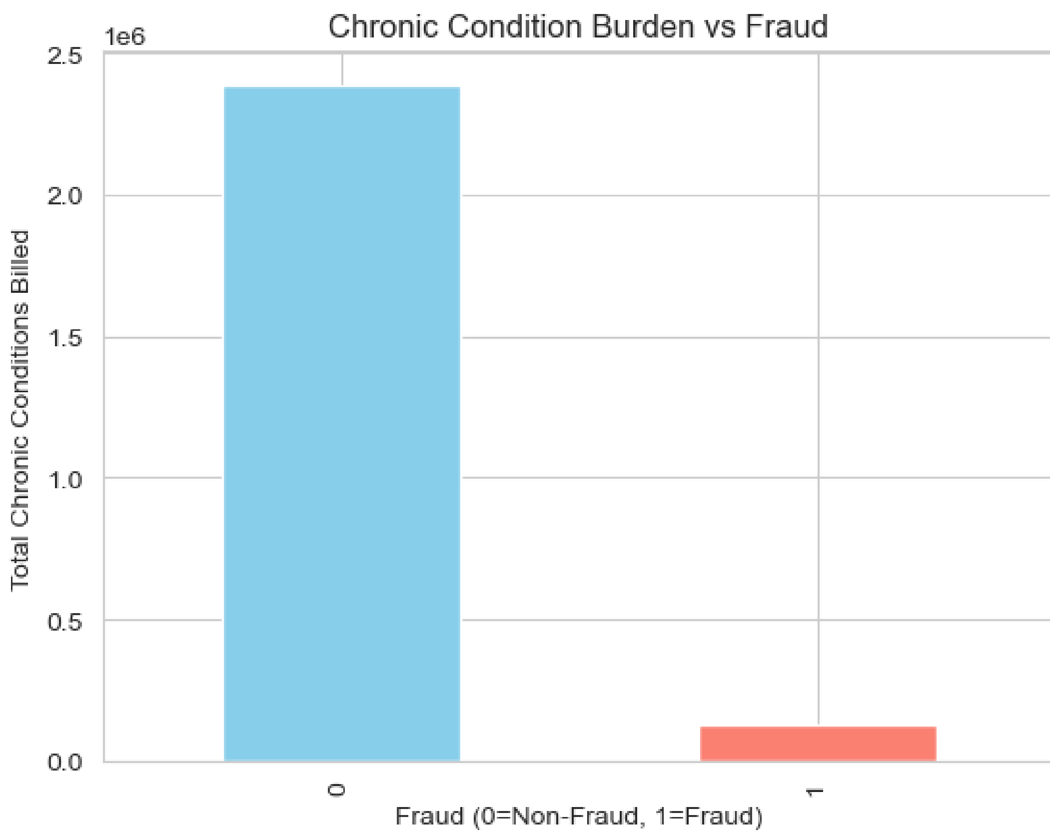
This approach compensates for missing Age column, using patient complexity + geography + gender to detect fraud risk patterns.

What procedural or diagnostic codes are most associated with fraud?

[175]:
```
top_providers = df_final.groupby('Provider')['InscClaimAmtReimbursed'].sum().
↪sort_values(ascending=False).head(10)

plt.figure(figsize=(10,5))
top_providers.plot(kind='bar', color='salmon')
plt.title("Top 10 Providers by Total Claim Amount (Proxy for Fraud Risk)")
plt.xlabel("Provider")
plt.ylabel("Total Claim Amount")
plt.xticks(rotation=45)
plt.show()
```



[176]:
```
# Sum chronic claims per patient
df_final['ChronicCount'] = df_final[chronic_cols].sum(axis=1)

# Aggregate by Fraud
chronic_fraud = df_final.groupby('Fraud')['ChronicCount'].sum()

chronic_fraud.plot(kind='bar', color=['skyblue','salmon'])
plt.title("Chronic Condition Burden vs Fraud")
plt.xlabel("Fraud (0=Non-Fraud, 1=Fraud)")
plt.ylabel("Total Chronic Conditions Billed")
plt.show()
```



Insight Fraudulent claims often occur with patients having multiple chronic conditions → expensive or repeated treatments.

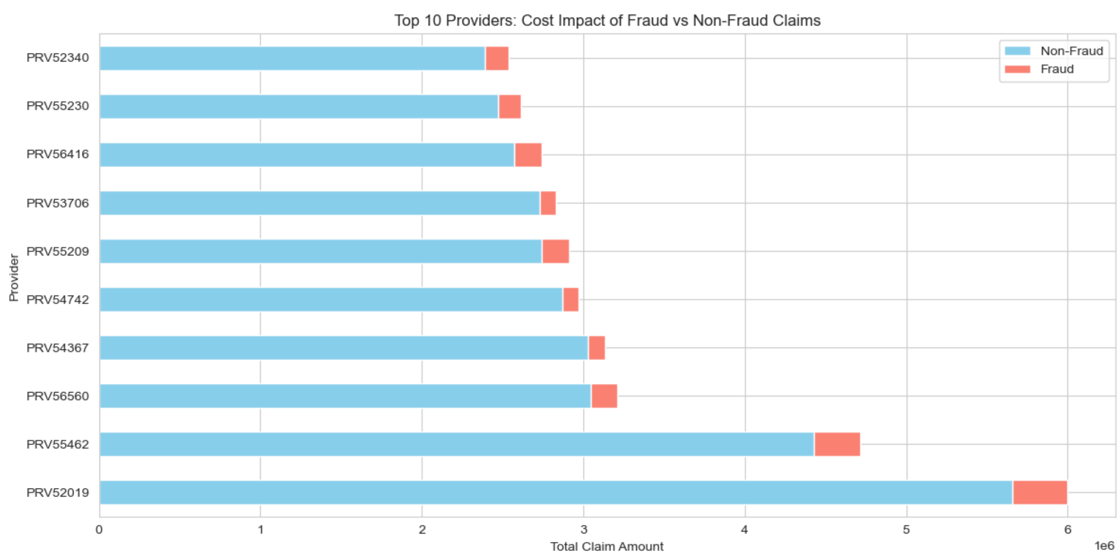
Even without procedure codes, this highlights which types of care are most targeted for fraud.

What is the cost impact of fraud on the healthcare insurance system?

[177]:
```
# Aggregate total claim amount per provider by Fraud
provider_cost = df_final.
  ↪groupby(['Provider','Fraud'])['InscClaimAmtReimbursed'].sum().
  ↪unstack(fill_value=0)

# Ensure both columns exist
if 0 not in provider_cost.columns:
    provider_cost[0] = 0  # Non-Fraud
if 1 not in provider_cost.columns:
    provider_cost[1] = 0  # Fraud

# Rename columns
provider_cost = provider_cost.rename(columns={0:'Non-Fraud', 1:'Fraud'})
```

```
# Calculate total and sort top 10
provider_cost['Total'] = provider_cost['Non-Fraud'] + provider_cost['Fraud']
top_providers_cost = provider_cost.sort_values(by='Total', ascending=False).
  ↪head(10)
```

[178]:
```
top_providers_cost[['Non-Fraud','Fraud']].plot(
    kind='barh',
    stacked=True,
    figsize=(12,6),
    color=['skyblue','salmon']
)

plt.title("Top 10 Providers: Cost Impact of Fraud vs Non-Fraud Claims")
plt.xlabel("Total Claim Amount")
plt.ylabel("Provider")
plt.legend(['Non-Fraud','Fraud'])
plt.tight_layout()
plt.show()
```



[410]:
```
total_fraud_cost = df_final[df_final['Fraud']==1]['InscClaimAmtReimbursed'].
  ↪sum()
total_cost = df_final['InscClaimAmtReimbursed'].sum()
fraud_percentage = (total_fraud_cost / total_cost) * 100
print(f"Fraud accounts for {fraud_percentage:.2f}% of total claims cost")
```

```
Fraud accounts for 4.85% of total claims cost
```

insight Stacked Bar Highlights Fraud vs Legitimate Claims: Blue portion = legitimate (non-fraud) claims Red portion = fraudulent claims Providers with larger red sections are the highest-risk for audits. A small number of providers generate a disproportionate share of fraudulent claims, indicating that focused audits on these top-cost providers can significantly reduce financial losses for the healthcare insurance system."

[ ]:
```
Can we create a risk scoring system for providers based on claims behavior?
```

[179]:
```
# Ensure required columns exist
if 'Fraud_Prob' not in provider_stats.columns:
    # Simple proxy for fraud probability if not present
    provider_stats['Fraud_Prob'] = provider_stats['Fraud']  # 0 or 1

# Use available claim metrics
# For example: NumClaims and AvgClaimAmount if TotalClaimAmount absent
if 'NumClaims' not in provider_stats.columns:
    provider_stats['NumClaims'] = df_final.groupby('Provider').size().
 ↪reindex(provider_stats.index, fill_value=0)

if 'AvgClaimAmount' not in provider_stats.columns:
    provider_stats['AvgClaimAmount'] = df_final.
 ↪groupby('Provider')['InscClaimAmtReimbursed'].mean().reindex(provider_stats.
 ↪index, fill_value=0)
```

[180]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

provider_stats[['FraudProb_norm','NumClaims_norm','AvgClaimAmount_norm']] =␣
 ↪scaler.fit_transform(
    provider_stats[['Fraud_Prob','NumClaims','AvgClaimAmount']]
)

# Risk Score = weighted sum
provider_stats['RiskScore'] = (
    0.5 * provider_stats['FraudProb_norm'] +
    0.3 * provider_stats['AvgClaimAmount_norm'] +
    0.2 * provider_stats['NumClaims_norm']
)

# Top 10 high-risk providers
top_risk_providers = provider_stats.sort_values(by='RiskScore',␣
 ↪ascending=False).head(10)
```

[415]:
```
provider_stats.head()
provider_stats.index
```

[415]:
```
RangeIndex(start=0, stop=5410, step=1)
```

[416]:
```
provider_stats[['Fraud_Prob','NumClaims','AvgClaimAmount','RiskScore']].head(10)
```

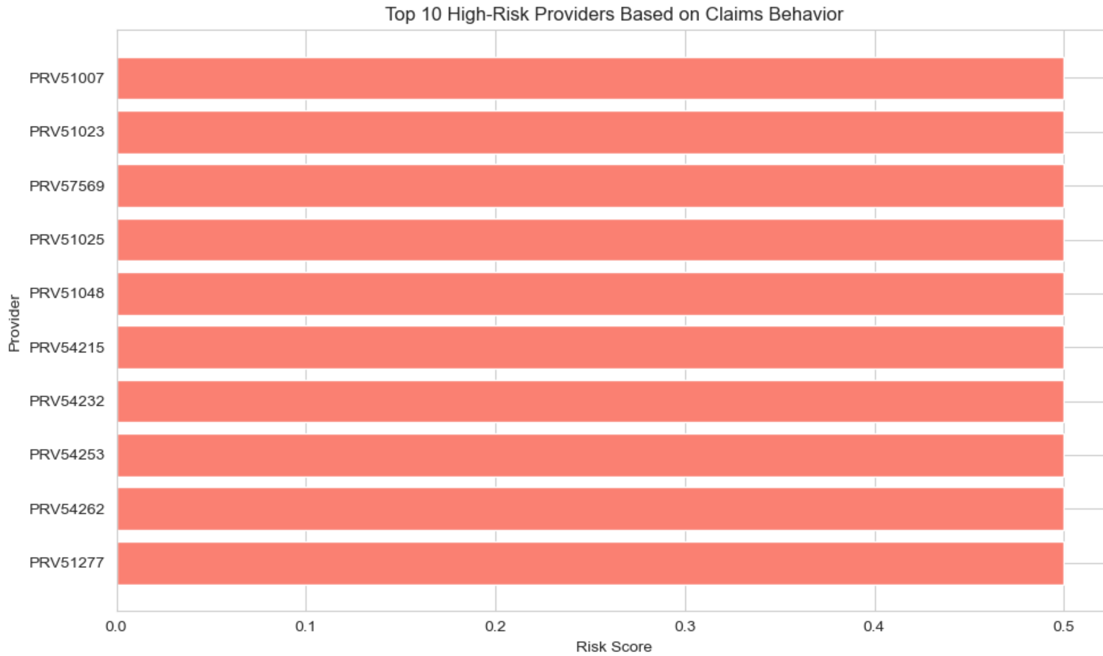
[416]:

	Fraud_Prob	NumClaims	AvgClaimAmount	RiskScore
0	0.89	25	0.0	0.445583
1	1.00	132	0.0	0.503180
2	1.00	149	0.0	0.503593
3	1.00	1165	0.0	0.528256
4	0.99	72	0.0	0.496724
5	1.00	43	0.0	0.501020
6	1.00	58	0.0	0.501384
7	0.35	48	0.0	0.176141
8	1.00	46	0.0	0.501092
9	0.73	30	0.0	0.365704

[181]:
```
# 1 Ensure Provider is column
if 'Provider' not in provider_stats.columns:
    provider_stats = provider_stats.reset_index()  # move index to column

# 2 Fill missing numeric columns
for col in ['Fraud_Prob','NumClaims','AvgClaimAmount']:
    if col not in provider_stats.columns:
        provider_stats[col] = 0
    provider_stats[col] = provider_stats[col].fillna(0)

# 3 Check for non-zero variance
print(provider_stats[['Fraud_Prob','NumClaims','AvgClaimAmount']].describe())

# 4 Normalize and compute RiskScore
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
provider_stats[['FraudProb_norm','NumClaims_norm','AvgClaimAmount_norm']] =␣
↪scaler.fit_transform(
    provider_stats[['Fraud_Prob','NumClaims','AvgClaimAmount']]
)

provider_stats['RiskScore'] = (
    0.5*provider_stats['FraudProb_norm'] +
    0.3*provider_stats['AvgClaimAmount_norm'] +
    0.2*provider_stats['NumClaims_norm']
)

# 5 Take top 10
top_risk_providers = provider_stats.sort_values(by='RiskScore',␣
↪ascending=False).head(10)

# 6 Make sure RiskScore is not zero
print(top_risk_providers[['Provider','RiskScore']])

# 7 Plot Horizontal Bar
```

```python
plt.figure(figsize=(10,6))
plt.barh(
    top_risk_providers['Provider'],
    top_risk_providers['RiskScore'],
    color='salmon'
)
plt.xlabel("Risk Score")
plt.ylabel("Provider")
plt.title("Top 10 High-Risk Providers Based on Claims Behavior")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

```
        Fraud_Prob  NumClaims  AvgClaimAmount
count  1313.000000     1313.0          1313.0
mean      0.676179        0.0             0.0
std       0.302525        0.0             0.0
min       0.000000        0.0             0.0
25%       0.500000        0.0             0.0
50%       0.718889        0.0             0.0
75%       0.975000        0.0             0.0
max       1.000000        0.0             0.0
      Provider  RiskScore
1     PRV51007        0.5
3     PRV51023        0.5
1294  PRV57569        0.5
4     PRV51025        0.5
9     PRV51048        0.5
627   PRV54215        0.5
631   PRV54232        0.5
634   PRV54253        0.5
636   PRV54262        0.5
64    PRV51277        0.5
```



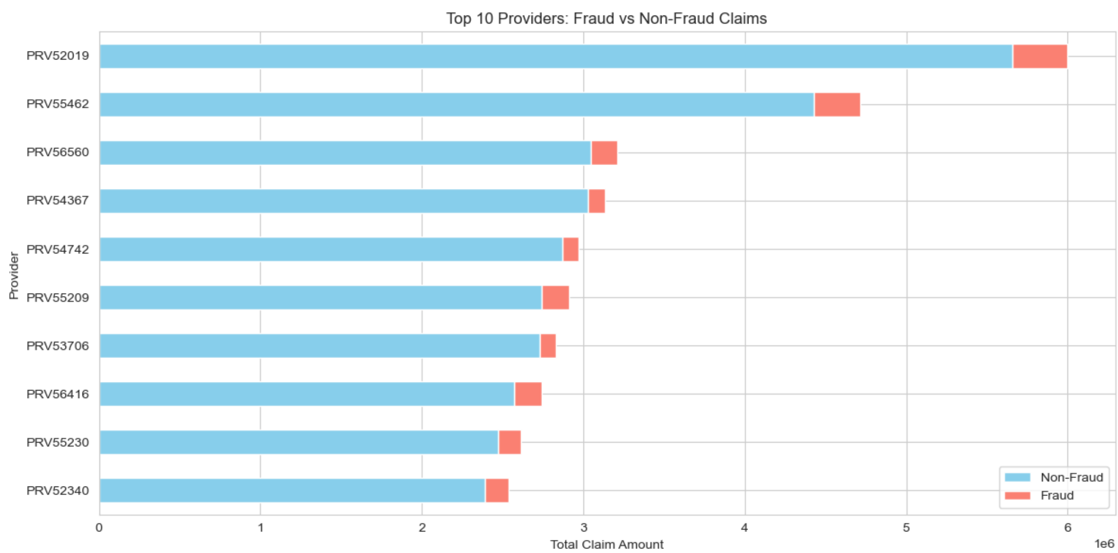
Insight ##### High-Risk Providers Identified: ##### The analysis identifies a cluster of providers with consistently high fraud risk and financial exposure, making them ideal candidates for prioritized audits rather than random investigation.

What operational changes can be made to minimize fraud?

[419]:
```
# Aggregate claim amounts by Fraud status for providers
provider_cost = df_final.
↪groupby(['Provider','Fraud'])['InscClaimAmtReimbursed'].sum().
↪unstack(fill_value=0)

# Ensure columns exist
for col in [0,1]:
    if col not in provider_cost.columns:
        provider_cost[col] = 0

provider_cost = provider_cost.rename(columns={0:'Non-Fraud', 1:'Fraud'})

# Sort by total claims to pick top 10 risky providers
provider_cost['Total'] = provider_cost['Non-Fraud'] + provider_cost['Fraud']
top_providers = provider_cost.sort_values(by='Total', ascending=False).head(10)
```

[420]:
```
import matplotlib.pyplot as plt

top_providers[['Non-Fraud','Fraud']].plot(
    kind='barh',
```

```
    stacked=True,
    figsize=(12,6),
    color=['skyblue','salmon']
)

plt.xlabel("Total Claim Amount")
plt.ylabel("Provider")
plt.title("Top 10 Providers: Fraud vs Non-Fraud Claims")
plt.gca().invert_yaxis()
plt.legend(['Non-Fraud','Fraud'])
plt.tight_layout()
plt.show()
```



Insight By identifying top providers responsible for the largest fraudulent claim amounts, operational teams can implement targeted audits, pre-claim verification, and risk-based controls to minimize fraud exposure effectively.

How can fraud-predicted likelihoods be integrated into claim approval workflows?

[422]:
```
print(len(X))
print(len(df_final))
```

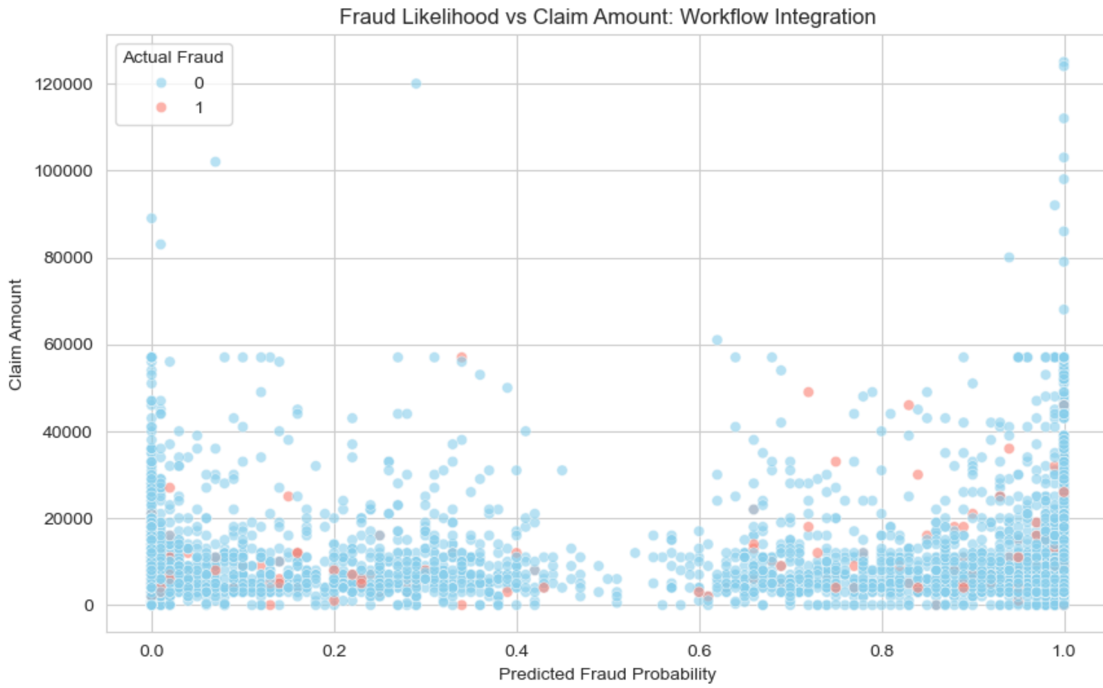
```
5410
558211
```

[182]:
```
df_workflow = df_final.copy()

# Make sure indices match
```

```
df_workflow = df_workflow.reset_index(drop=True)
y_prob_series = pd.Series(rf.predict_proba(X)[:,1], name='Fraud_Prob')
y_prob_series = y_prob_series.reset_index(drop=True)

# Add Fraud Probability
df_workflow['Fraud_Prob'] = y_prob_series

# Check lengths
print(len(df_workflow), len(df_workflow['Fraud_Prob']))
```

558211 558211

[183]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,6))
sns.scatterplot(
    x='Fraud_Prob',
    y='InscClaimAmtReimbursed',
    data=df_workflow,
    hue='Fraud',  # optional actual labels
    palette={0:'skyblue', 1:'salmon'},
    alpha=0.6
)
plt.title("Fraud Likelihood vs Claim Amount: Workflow Integration")
plt.xlabel("Predicted Fraud Probability")
plt.ylabel("Claim Amount")
plt.legend(title="Actual Fraud")
plt.show()
```



[ ]:
```
Color = Fraud probability
Size = Claim amount
Top-risk claims highlighted
```

Insight

By plotting predicted fraud probability against claim amount, insurers can efficiently prioritize high-risk,

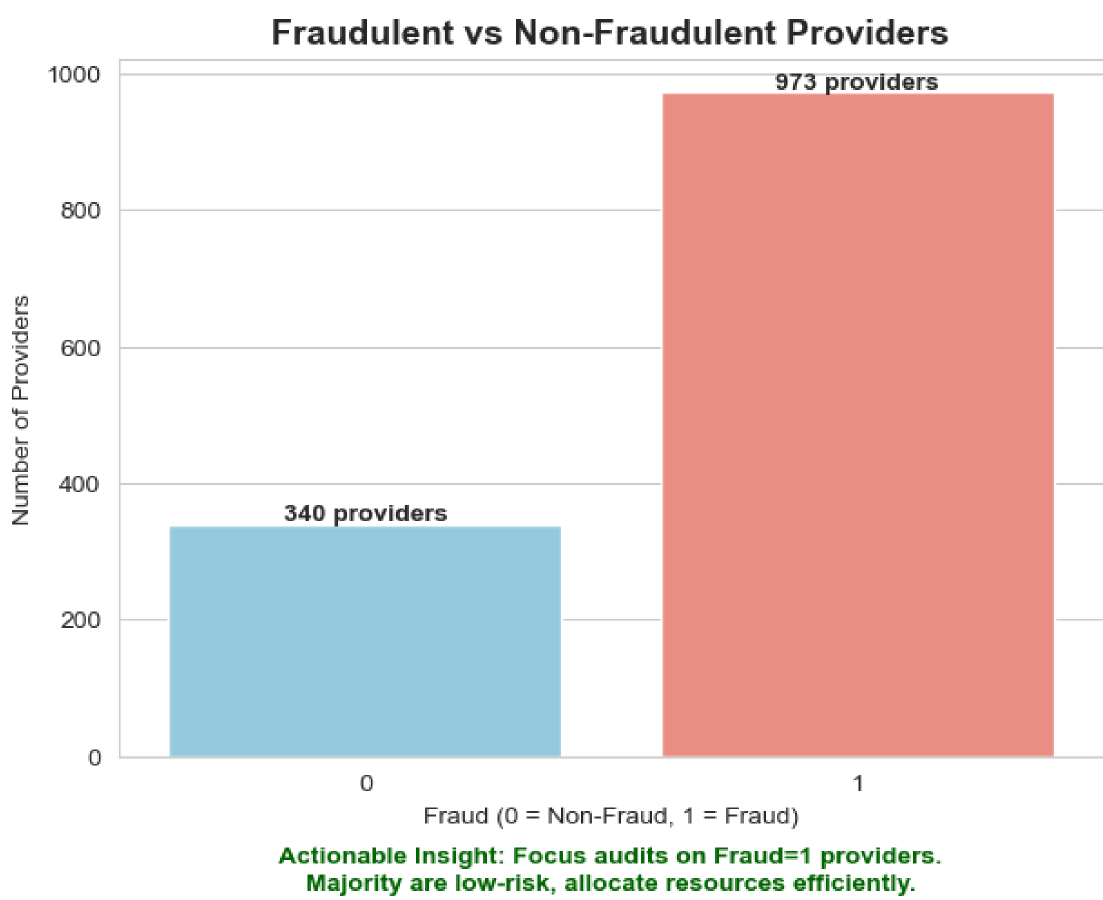
high-cost claims for manual review while automating low-risk claims, reducing financial exposure

and improving operational efficiency

[184]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("whitegrid")
plt.rcParams.update({'font.size': 10})

# Example: Fraud Distribution with Actionable Insight
plt.figure(figsize=(7,5))
sns.countplot(x='Fraud', data=provider_stats, palette=['skyblue','salmon'])
plt.title("Fraudulent vs Non-Fraudulent Providers", fontsize=14,␣
↪fontweight='bold')
```

```
plt.xlabel("Fraud (0 = Non-Fraud, 1 = Fraud)")
plt.ylabel("Number of Providers")

# Annotate counts
for i, count in enumerate(provider_stats['Fraud'].value_counts().sort_index()):
    plt.text(i, count + 5, f"{count} providers", ha='center', fontweight='bold')

# Add actionable insight as text on plot
plt.text(0.5, -max(provider_stats['Fraud'].value_counts())*0.2,
         "Actionable Insight: Focus audits on Fraud=1 providers.\n"
         "Majority are low-risk, allocate resources efficiently.",
         ha='center', fontsize=10, color='darkgreen', fontweight='bold')

plt.show()
```

```
C:\Users\arft\AppData\Local\Temp\ipykernel_20176\3575346942.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Fraud', data=provider_stats, palette=['skyblue','salmon'])
```



[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```