AI - Claim Processing

February 26, 2020

0.0.1 Problem

AI in Claim Processing:

The insurance industry is dominated by large global firms that deal with thousands of customers filing insurance claims every day. Claims processing is a huge part of the insurance business process and improving turnaround time for each claim is critical to reducing operational costs at insurance firms.

As such, insurance carriers might find it challenging to improve their claims processes due to the sheer scale of incoming claims. In other words, it's difficult for them to pick up on patterns within their claims data that they may want to act on. This is a classic case for artificial intelligence.

0.0.2 Solution

At Acceltree, we created AI-based POC to smooth the health insurance claim processing and find out the pattern from the submitted cases.

We consider pre-approved-denied.csv which contains all claim approved and denied cases.

Insurer_Status is the target attribute with claim approved/denied status.

Total No of Approved Cases:243

Total No of Denied Cases:41

pre-approved-denied.csv conatins total 50 attributes:

ClmID, PriBenefGrade, BenefAge, Age_Band, Sex, Lenght of Stay, ClmAmount, ClmApprovedAmt, Settled_Amt_Band, Approved Amt Before deduction, Incurred Amount, Basic_Sum_Insured, Sum_Insured , Balance Sum Insured, Ailment_code, Ailment_Grp, Illness, Procedure Type(Surgical/Non-Surgical), HospID, HospName, CityName, HOSPITAL STATE, HOSPITAL IS IN PPN Y/N, RoomCategory, Room Charges Claimed, Room Charges Paid, Room Charges Claimed Per Day, Room Charges Paid Per Day, Nursing Charges Claimed, Nursing Charges Paid, Other Hospital Charges Claimed, Other Hospital Charges Paid, Lab Charges Claimed, Lab Charges Paid, Medi Charges Claimed, Medi Charges Paid, Misc Charges Claimed, Misc Charges Paid, Surgery Charges Claimed, Surgery Charges Paid, Consultant Charges

Claimed, Consultant Charges Paid, Package Rate Claimed, Package Rate Paid, Excess of Defined Ailment Limit, Policy Excess, Hospital Discount, Patient Paid Amount, Deduction Amt, Insurer_Status.

```
In [1]: import pandas as pd
        df = pd.read_csv("pre-approved-denied.csv")
        df.head(5)
```

```
Out[1]:       ClmID      PriBenefGrade  BenefAge Age_Band Sex  Lenght of Stay  \
        0  14301382          Associate         0     0-18   M               4
        1  14324426          Associate         0     0-18   M               3
        2  14353580   Senior Associate         0     0-18   F               4
        3  14354256   Senior Associate         0     0-18   M               2
        4  14362279                  0        29    26-30   F               6

           ClmAmount  ClmApprovedAmt Settled_Amt_Band  Approved Amt Before deduction  \
        0      12873           11030      10001-25000                          12873
        1      12046           10112      10001-25000                          11894
        2      25847           23130      10001-25000                          25700
        3      17209           13965      10001-25000                          15609
        4      76610           75449     50001-100000                          76610

                ...        Consultant Charges Claimed  Consultant Charges Paid  \
        0       ...                              2400                     2400
        1       ...                              1600                     1600
        2       ...                                 0                        0
        3       ...                                 0                        0
        4       ...                                 0                        0

           Package Rate Claimed  Package Rate Paid Excess of Defined Ailment Limit  \
        0                     0                  0                               0
        1                     0                  0                               0
        2                     0                  0                               0
        3                     0                  0                               0
        4                     0                  0                               0

          Policy Excess Hospital Discount Patient Paid Amount  Deduction Amt  \
        0             0              1843                   0              0
        1             0              1782                   0            152
        2             0              2570                   0            147
        3             0               476                1168           1600
        4             0              1161                   0              0

          Insurer_Status
        0       Approved
        1       Approved
        2       Approved
        3       Approved
```

```
4        Approved

[5 rows x 50 columns]
```

In [2]:
```
# Need to decode categorical attributes
# function to obtain Categorical Features

def _get_categorical_features(df):
    feats = [col for col in list(df.columns) if df[col].dtype == 'object']
    return feats

# function to factorize categorical features
def _factorize_categoricals(df, cats):
    for col in cats:
        df[col], _ = pd.factorize(df[col])
    return df
```

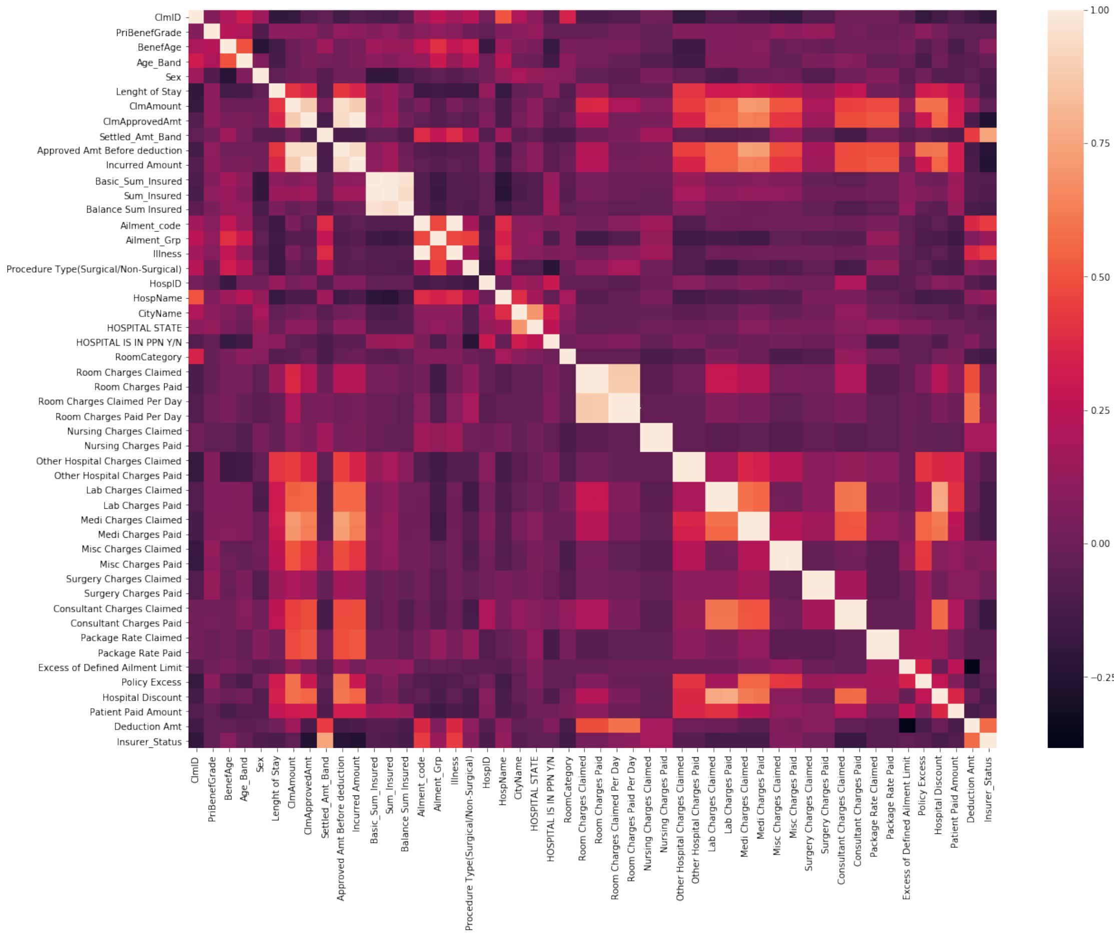
In [3]:
```
df_cats = _get_categorical_features(df)
```

In [4]:
```
train_df = _factorize_categoricals( df, df_cats)
train_df.head()
train_df.head()
train_df.to_csv('CleanDataNew.csv',index=False)
```

0.0.3 Statistical Analysis

In [27]:
```
# Find the correlation of attributes with Insurer_Status
# The attributes having positive correlations with target attribute i.e Insurer_Status
# can be good predictors.
import matplotlib.pyplot as plt
import seaborn as sns
corr=train_df.corr()
plt.figure(figsize=(20,15))
sns.heatmap(corr,
        xticklabels=corr.columns,
        yticklabels=corr.columns)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f22e5254358>
```



Further apply the statistical analyisis to get the description of data

```
In [6]: import pandas as pd
        df = pd.read_csv("CleanDataNew.csv")
```

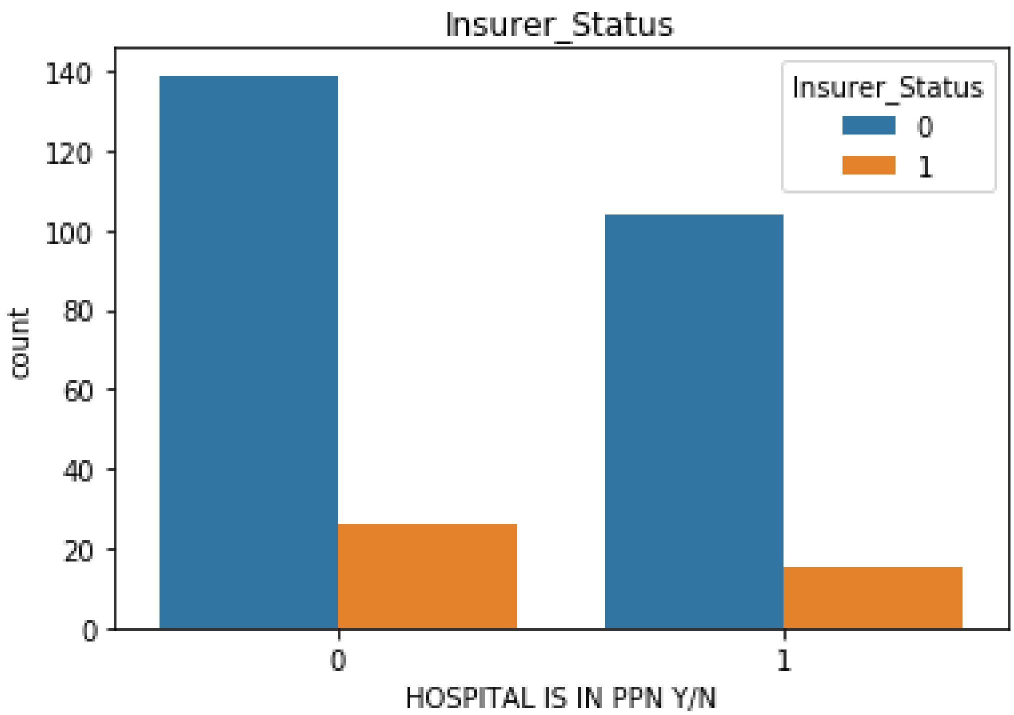
0.0.4 Insights from data

```
In [7]: # Hospitals which are in PPN (Preferred Provider Network)
        # 0 -PPN
        # 1-NOT PPN
        df.groupby(['HOSPITAL IS IN PPN Y/N','Insurer_Status'])['Insurer_Status'].count()
```

```
Out[7]: HOSPITAL IS IN PPN Y/N  Insurer_Status
        0                       0                 139
                                1                  26
        1                       0                 104
                                1                  15
        Name: Insurer_Status, dtype: int64
```

In [8]:
```
fig, ax = plt.subplots()
sns.countplot('HOSPITAL IS IN PPN Y/N',hue='Insurer_Status',data=df)
ax.set_title('Insurer_Status')
```

Out[8]:
```
Text(0.5, 1.0, 'Insurer_Status')
```



Insight: Hospitals in PPN are having more approved and denied cases.

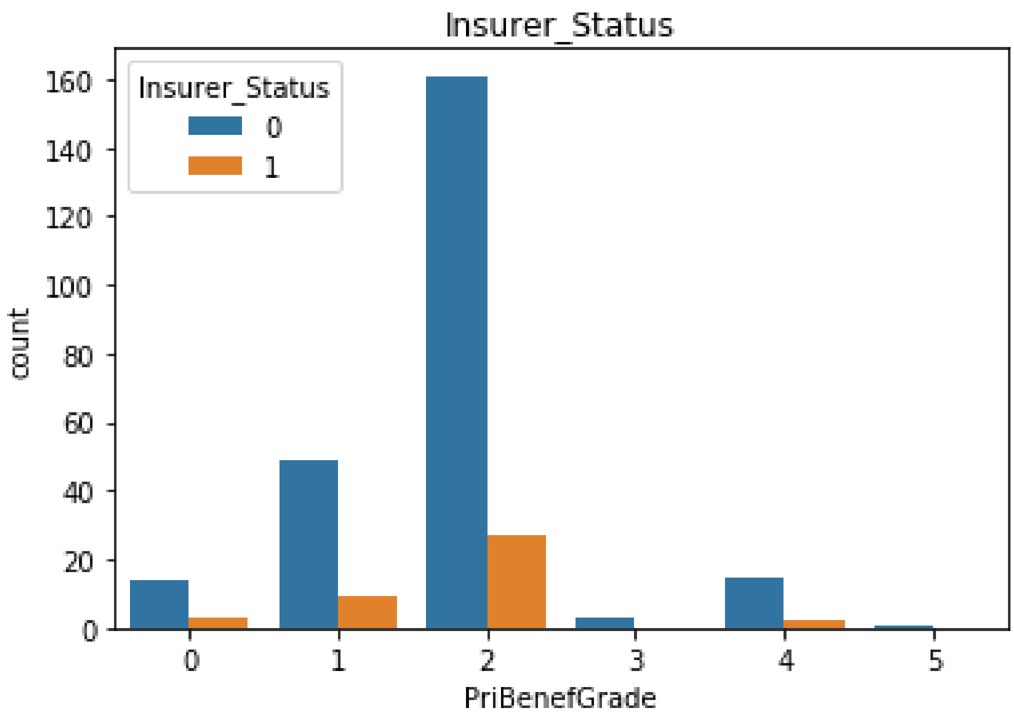
In [9]:
```
# Analysis with primary benefitiary grade
# the grades are like Associate, Director/expert, Manager, senior associate etc.
df.groupby(['PriBenefGrade','Insurer_Status'])['Insurer_Status'].count()

#fig, ax = plt.subplots()
#sns.countplot('HOSPITAL IS IN PPN Y/N',hue='Insurer_Status',data=df)
#ax.set_title('Insurer_Status')
df["PriBenefGrade"].value_counts()
```

Out[9]:
```
2    188
1     58
4     17
0     17
3      3
5      1
Name: PriBenefGrade, dtype: int64
```

In [10]:
```
fig, ax = plt.subplots()
sns.countplot('PriBenefGrade',hue='Insurer_Status',data=df)
ax.set_title('Insurer_Status')
```

Out[10]:
```
Text(0.5, 1.0, 'Insurer_Status')
```



Insight: The PriBenefGrade having Grade=2 i.e Director/Expert having more approved and denied cases.

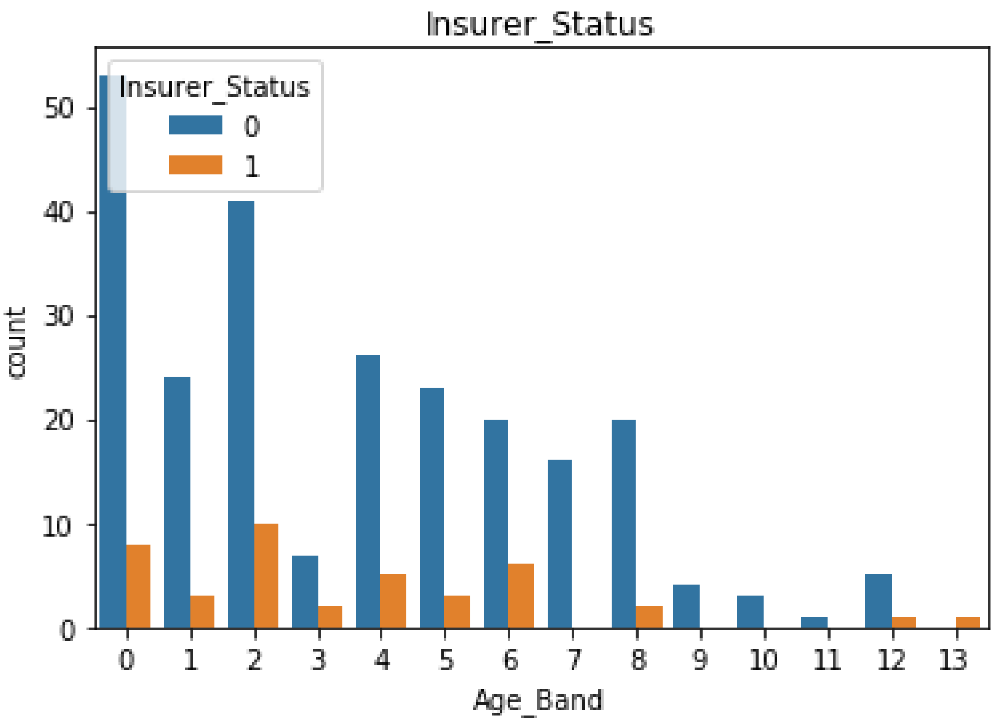
In [11]:
```
# Analysis with age-band
df.groupby(['Age_Band','Insurer_Status'])['Insurer_Status'].count()
```

Out[11]:
```
Age_Band  Insurer_Status
0         0                 53
          1                  8
1         0                 24
          1                  3
2         0                 41
          1                 10
3         0                  7
          1                  2
4         0                 26
          1                  5
5         0                 23
          1                  3
```

```
6          0                  20
           1                   6
7          0                  16
8          0                  20
           1                   2
9          0                   4
10         0                   3
11         0                   1
12         0                   5
           1                   1
13         1                   1
Name: Insurer_Status, dtype: int64
```

In [12]:
```python
fig, ax = plt.subplots()
sns.countplot('Age_Band',hue='Insurer_Status',data=df)
ax.set_title('Insurer_Status')
```

Out[12]:
```
Text(0.5, 1.0, 'Insurer_Status')
```



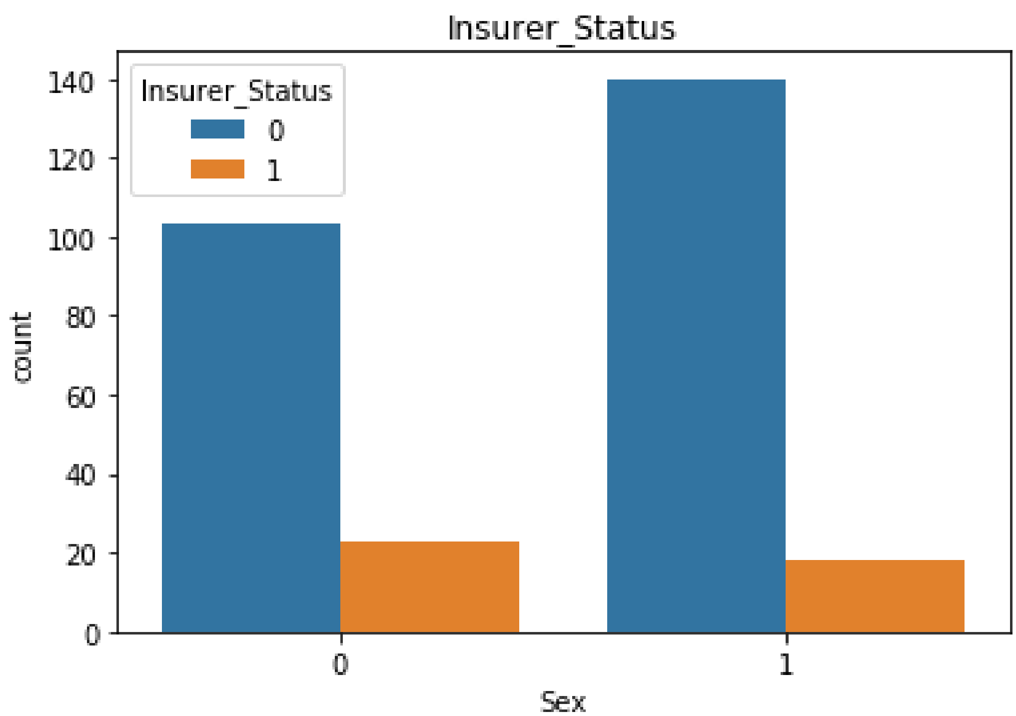
In [13]:
```python
# Sex-wise analysis
df.groupby(['Sex','Insurer_Status'])['Insurer_Status'].count()
```

Out[13]:
```
Sex   Insurer_Status
0     0                   103
```

```
     1                    23
1    0                   140
     1                    18
Name: Insurer_Status, dtype: int64
```

In [14]:
```python
fig, ax = plt.subplots()
sns.countplot('Sex',hue='Insurer_Status',data=df)
ax.set_title('Insurer_Status')
```

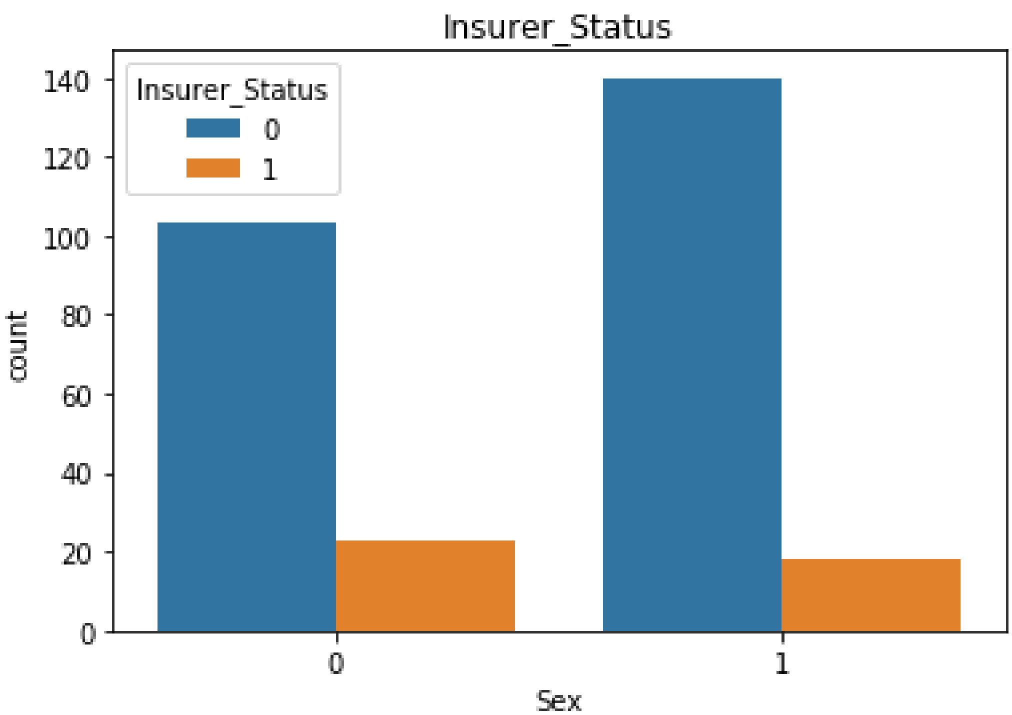
Out[14]: `Text(0.5, 1.0, 'Insurer_Status')`



Insight: Female(1) having more approved cases and male(0) having more denied cases.

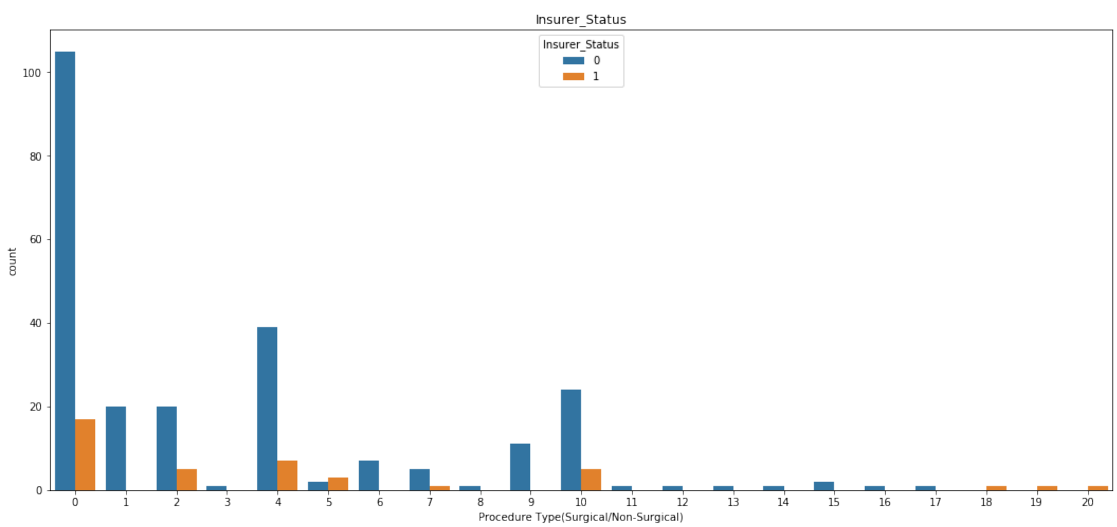
In [15]:
```python
fig, ax = plt.subplots()
sns.countplot('Sex',hue='Insurer_Status',data=df)
ax.set_title('Insurer_Status')
```

Out[15]: `Text(0.5, 1.0, 'Insurer_Status')`




```
In [16]: fig, ax = plt.subplots(figsize=(18,8))
         sns.countplot('Procedure Type(Surgical/Non-Surgical)',hue='Insurer_Status',data=df)
         ax.set_title('Insurer_Status')
```

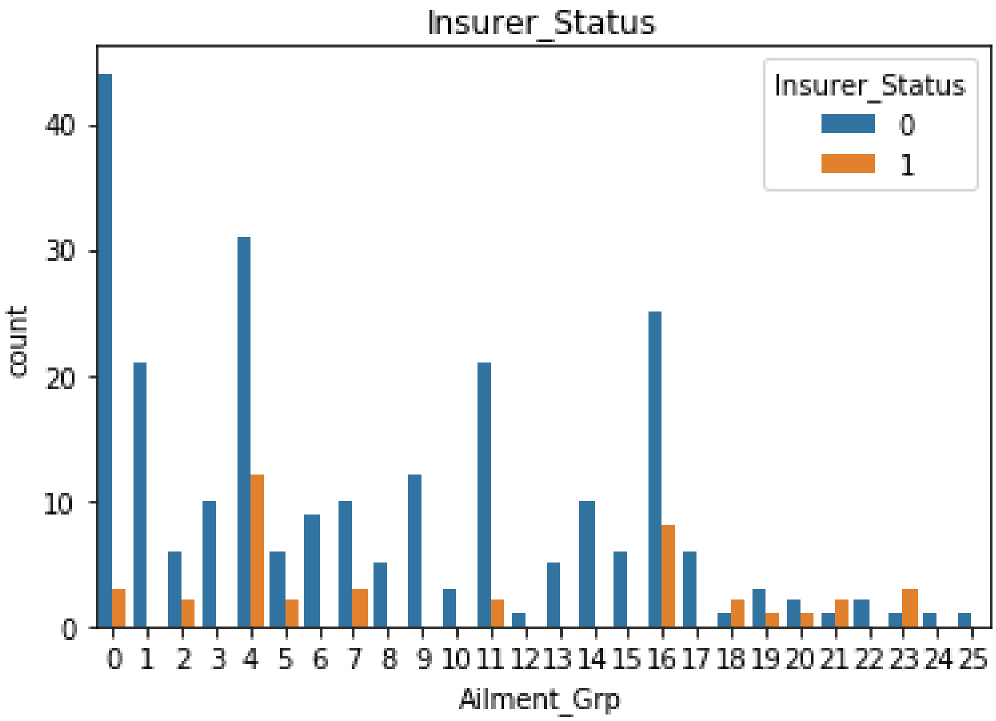
```
Out[16]: Text(0.5, 1.0, 'Insurer_Status')
```



Insight: The 9 procedure types out of 20 having denied cases.

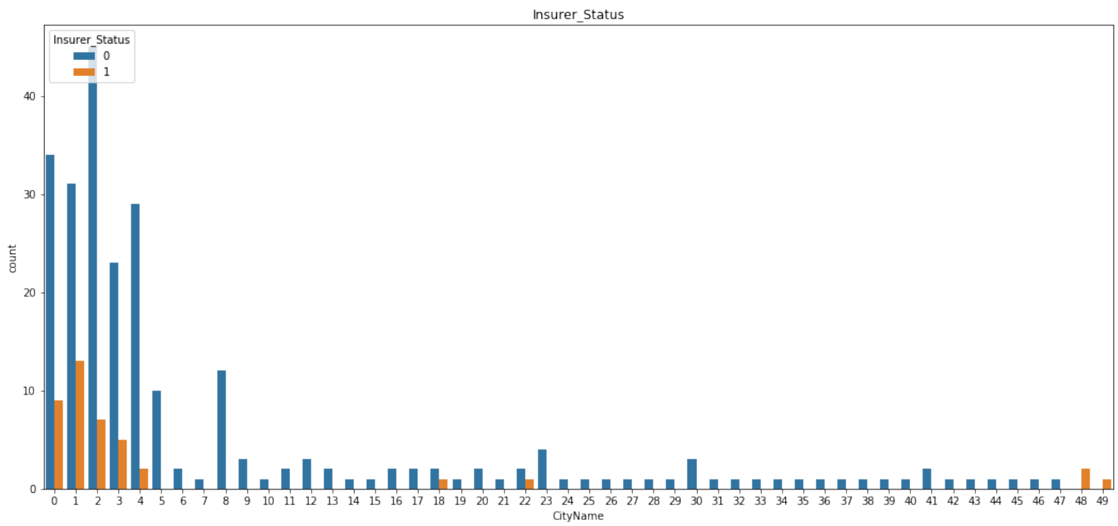
```
In [17]: fig, ax = plt.subplots()
         sns.countplot('Ailment_Grp',hue='Insurer_Status',data=df)
         ax.set_title('Insurer_Status')
```

```
Out[17]: Text(0.5, 1.0, 'Insurer_Status')
```



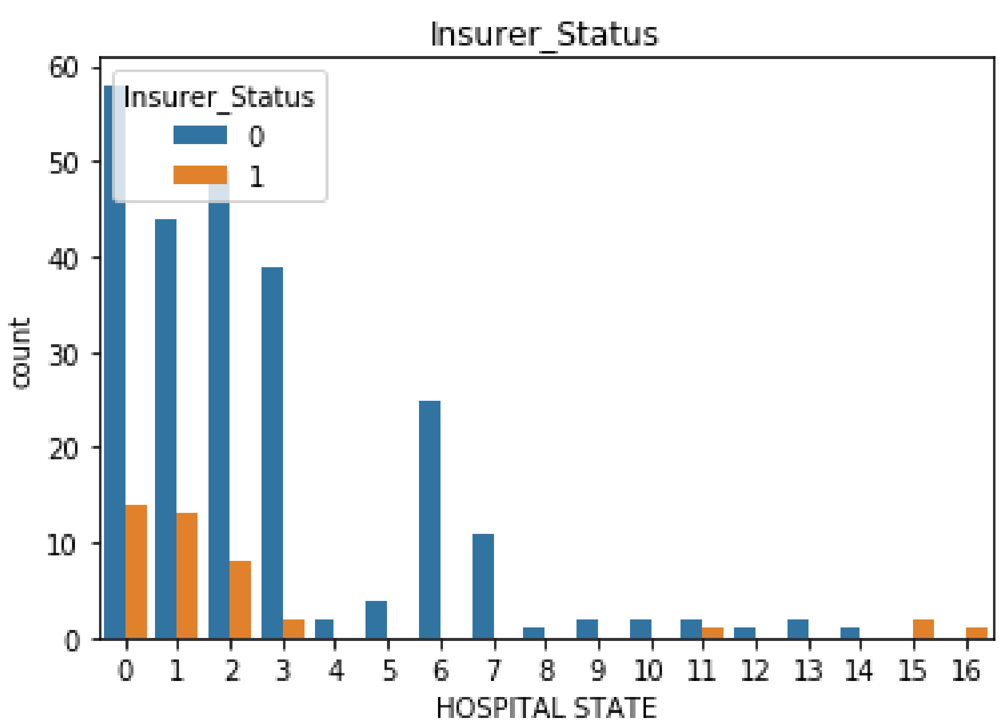

```
In [18]: fig, ax = plt.subplots(figsize=(18,8))
         sns.countplot('CityName',hue='Insurer_Status',data=df)
         ax.set_title('Insurer_Status')
```

```
Out[18]: Text(0.5, 1.0, 'Insurer_Status')
```



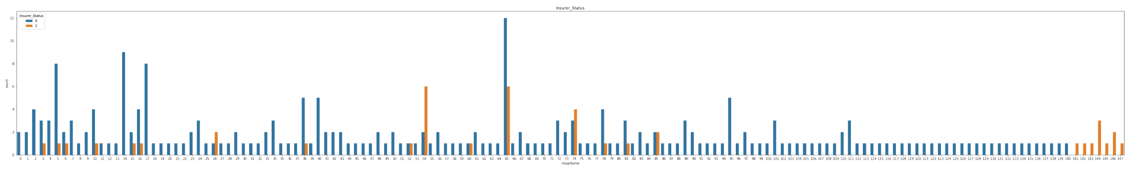
In [19]:
```
fig, ax = plt.subplots()
sns.countplot('HOSPITAL STATE',hue='Insurer_Status',data=df)
ax.set_title('Insurer_Status')
```

Out[19]: Text(0.5, 1.0, 'Insurer_Status')



In [20]:
```python
fig, ax = plt.subplots(figsize=(60,8))
sns.countplot('HospName',hue='Insurer_Status',data=df)
ax.set_title('Insurer_Status')
```

Out[20]: Text(0.5, 1.0, 'Insurer_Status')



Insight: Few hospitals are having denied cases. Hospital No. 26,74 are having more denied cases than approved cases. Some hospitals are having only denied cases.

0.0.5 Predictive Model

Further we apply predictived model considering the target attribute Insurer_Status (Approved/Denied)

In [21]:
```python
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, precision_score, rec
df= pd.read_csv("CleanDataNew.csv")
```

In [22]:
```python
X_train = df.drop('Insurer_Status',axis=1)
Y_train = df['Insurer_Status']


X_train, X_test, Y_train, Y_test = train_test_split(X_train,Y_train,test_size=0.03,rand
X_train.shape, X_test.shape

X_train_columns = X_train.columns


scaler = StandardScaler()

scaler.fit(X_train)

scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

In [23]:
```python
print("----------------   SVM   ----------------")
print("\n")
```

```
svm = SVC()

svm.fit(scaled_X_train, Y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(scaled_X_train, Y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
     .format(svm.score(scaled_X_test, Y_test)))

pred_svm = svm.predict(scaled_X_test)
print(classification_report(Y_test, pred_svm))
```

```
----------------   SVM   ----------------


Accuracy of SVM classifier on training set: 0.97
Accuracy of SVM classifier on test set: 0.89
             precision    recall  f1-score   support

          0       0.86      1.00      0.92         6
          1       1.00      0.67      0.80         3

avg / total       0.90      0.89      0.88         9
```

Conclusion: The Accuracy of SVM classifier on training set: 97% and Accuracy of SVM classifier on test set: 89%. However the accuracy for approved cases is 100% and denied cases is 67%. The reason behind this is the class imbalancing since we have 41 denied cases and 241 approved cases.

We can overcome this by upsampling

Next, we demonstrate the results after upsampling.

In [24]:
```
from sklearn.utils import resample
df= pd.read_csv("CleanDataNew.csv")
df_majority = df[df.Insurer_Status==0]
df_minority = df[df.Insurer_Status==1]
# Upsample minority class


df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=243,    # to match majority class
                                 random_state=123) # reproducible results




df_upsampled = pd.concat([df_majority, df_minority_upsampled])
```

In [25]:
```
X_train = df_upsampled.drop('Insurer_Status',axis=1)
Y_train = df_upsampled['Insurer_Status']


X_train, X_test, Y_train, Y_test = train_test_split(X_train,Y_train,test_size=0.02,rand
X_train.shape, X_test.shape

X_train_columns = X_train.columns


scaler = StandardScaler()

scaler.fit(X_train)

scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

In [26]:
```
print("----------------   SVM   ----------------")
print("\n")


svm = SVC()

svm.fit(scaled_X_train, Y_train)
print('Accuracy of SVM classifier on training set: {:.2f}'
     .format(svm.score(scaled_X_train, Y_train)))
print('Accuracy of SVM classifier on test set: {:.2f}'
     .format(svm.score(scaled_X_test, Y_test)))

pred_svm = svm.predict(scaled_X_test)
print(classification_report(Y_test, pred_svm))
```

```
----------------   SVM   ----------------


Accuracy of SVM classifier on training set: 0.99
Accuracy of SVM classifier on test set: 1.00
             precision    recall  f1-score   support

          0       1.00      1.00      1.00         6
          1       1.00      1.00      1.00         4

avg / total       1.00      1.00      1.00        10
```

Conclusion: We get the 100% accuarcy on approved and denied cases, but there may be the chance of overfitting of data.

Takeaway is with the help of AI and data science we can analyze the data and derive meaningful insights from data. Further with the help of machine learning techniques we can develop predictive models and optimise claims processing at scale without dramatically increasing operational costs.