

**Python for Data Analytics**

**ANL252**

**ECA**

**January 2023**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **KOH REN YI DOMINIC** | **E1911548** |

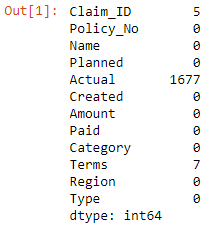
**Tutorial Group: ­­­­­­­­­­­­ T05**

**Instructor’s Name: Dr. Munish Kumar**

**Submission Date: 05/03/2023**

**Question 1**

Figure 1:



With this code, it was identified that under these following categories, there were missing values found:

1. Claim\_ID: 5 records with missing value.
2. Actual: 1677 records with missing value.
3. Terms: 7 records with missing value.

**Question 2**

Figure 2:

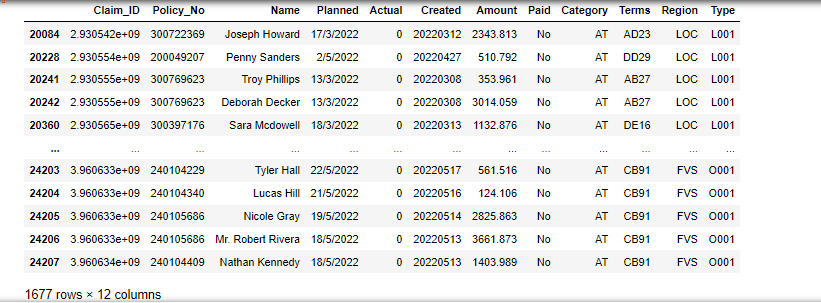


Figure 3:

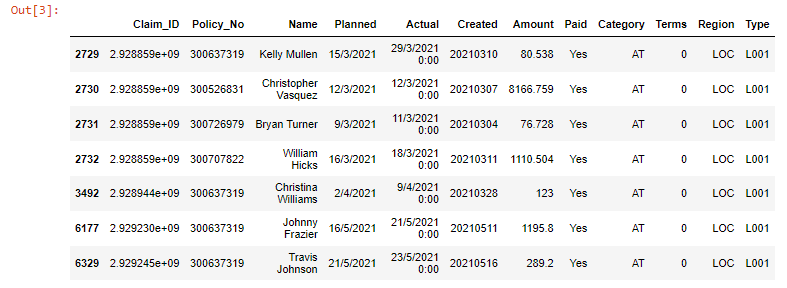
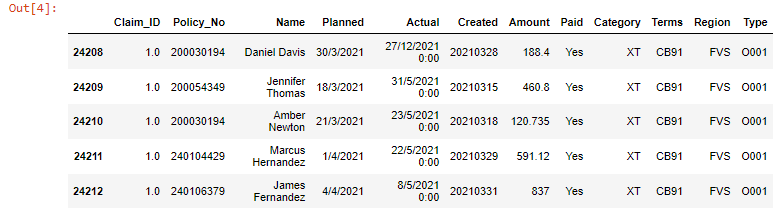


Figure 4:

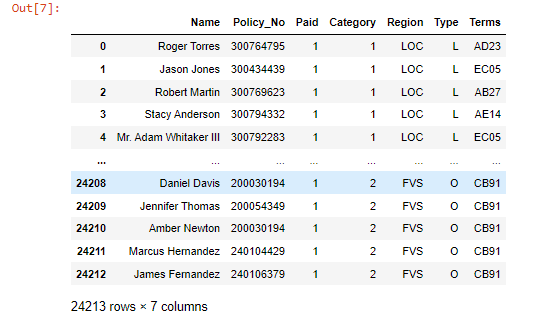


The following columns and parameters were treated in order as part of the data preparation:

1. Claim\_ID had those missing values being renamed to “1”. Despite not having any Claim\_ID, there are traceability as to when the actual settlement claim date was being performed, furthermore, having “Policy\_No” and “Amount” should allow the user to trace back to the actual Claim\_ID.
2. “Actual” and “Terms” which had missing values were renamed to “1”.

**Question 3:**

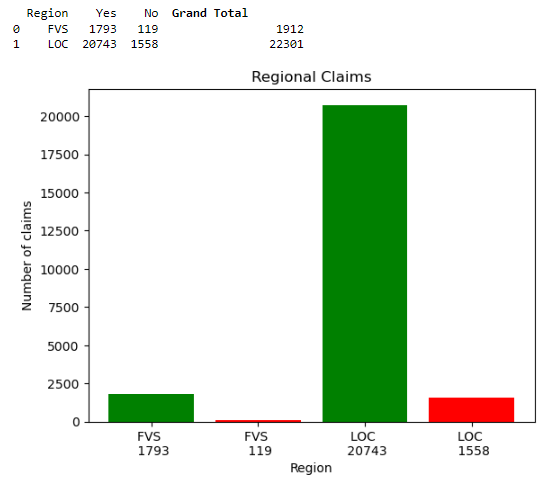
Figure 5:



1. The category “Paid” had their outputs “Yes” and “No” renamed to “1” and “0” respectively. This is to further understand if there are any interlink between both “Paid” and “Actual”. It was found that, these two categories are interlinked, if the actual date of claim settlement was not done, it further justifies that “Paid” was not done as well.
2. Reducing “Type” outputs from 6 to 2.
3. Displaying only the columns that is required (Total 7 columns).

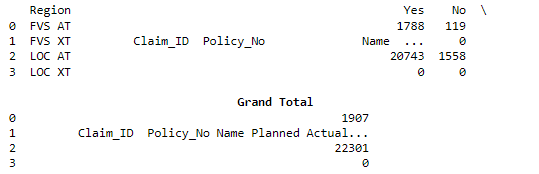
**Question 4:**

Figure 6:



Based on the above bar graph, it can be shown that the amount of claims being made total by each respective regions and the number of claims with payment being made that are processed. For FVS region, the overall % of claims that are not processed yet are standing at around 6% and for LOC is around 7%. As compared to the payments that have successfully being made, both regions overall payments are above 90%. The insurance company may want use this data to further understand from those respective regions as to why the outstanding claims are not being processed. Is it due to missing information such as “Terms”?

Figure 7.



Based on the data further broken down into “Category”, XT categories are shown as 100% processed under the Region FVS and under Region LOC, there is no claims that falls under XT. Hence, insurance companies are mainly having delayed processing of claims under “AT”. However, as sample size for XT is only 5 records, therefore, this may not be sufficient to state that XT categories will normally be processed quicker.

**Appendix – Figure 1**

In [1]:

#importing libraries

import sqlite3

import pandas as pd

#naming DF as dataframe name that is pulling data from ECA.csv

#na\_values is renaming the additional fields as missing values

df = pd.read\_csv('C:/Users/User/Desktop/ECA.csv',

na\_values = ("???","Unkn"))

#this code here is to list out the columns with the specified missing values.

#Blanks are included as it is a default missing value.

df.isnull().sum(axis=0)

#Output image

**Appendix – Figure 2 to Figure 4**

In [2]:

#replacing the values in "Paid" category for "Yes" and "No" as "1" and "0".

df['Paid'] = df['Paid'].replace(['Yes', 'No'], ['1', '0'])

#Filling up those specified missing values as "1"

df['Terms'] = df['Terms'].fillna(1)

#Filling up those specified missing values as "1"

df['Claim\_ID'] = df['Claim\_ID'].fillna(1)

#Filling up those specified missing values as "0"

df['Actual'] = df['Actual'].fillna(0)

#Printing out those category under Actual as 0. Total 1677 records, tallies as per question 1 Output image.

df.loc[(df['Actual'] == 0)]

In [3]:

#Printing out those terms specified as "1". Total 7 records, tallies as per question 1 Output image.

df.loc[(df['Terms'] == 1)]

In [4]:

#Printing out Claim ID specified as "1". Total 5 records, tallies as per question 1 Output image.

df.loc[(df['Claim\_ID'] == 1)]

**Appendix – Figure 5**

#replacing the values in "Category" and "Type" and "Paid".

df['Paid'] = df['Paid'].replace(['Yes', 'No'], ['1', '0'])

df['Category'] = df['Category'].replace(['AT', 'XT'], ['1', '2'])

df['Type'] = df['Type'].replace(['L001', 'L002', 'L003', 'L004', 'L005'], 'L')

df['Type'] = df['Type'].replace(['O001'], 'O')

#reducing the number of columns being shown.

df.loc[:,['Name', 'Policy\_No', 'Paid', 'Category', 'Region', 'Type', 'Terms']]

**Appendix – Figure 6**

#Filtering the dataframe by claims paid under the respective Regions either paid "1" or not paid "0"

Claims\_Paid\_Yes\_FVS = df[

(df['Paid'] == '1')

&

(df['Region'] == 'FVS')

]

Claims\_Paid\_No\_FVS = df[

(df["Paid"] == '0')

&

(df['Region'] == 'FVS')

]

Claims\_Paid\_Yes\_LOC = df[

(df["Paid"] == '1')

&

(df["Region"] == 'LOC')

]

Claims\_Paid\_No\_LOC = df[

(df["Paid"] == '0')

&

(df["Region"] == 'LOC')

]

# count the number of claims paid under "1" or "0"

num\_Claims\_Paid\_Yes\_FVS = len(Claims\_Paid\_Yes\_FVS)

num\_Claims\_Paid\_No\_FVS = len(Claims\_Paid\_No\_FVS)

num\_Claims\_Paid\_Yes\_LOC = len(Claims\_Paid\_Yes\_LOC)

num\_Claims\_Paid\_No\_LOC = len(Claims\_Paid\_No\_LOC)

#creating a table using dictionary of list

df = {

'Region': ['FVS', 'LOC'],

'Yes': [num\_Claims\_Paid\_Yes\_FVS, num\_Claims\_Paid\_Yes\_LOC],

'No': [num\_Claims\_Paid\_No\_FVS, num\_Claims\_Paid\_No\_LOC],

'\033[1mGrand Total\033[0m': [(num\_Claims\_Paid\_Yes\_FVS +

num\_Claims\_Paid\_No\_FVS),

(

num\_Claims\_Paid\_Yes\_LOC +

num\_Claims\_Paid\_No\_LOC

)

]

}

dftable = pd.DataFrame(df)

print (dftable)

#Data for bar graph

values = [

num\_Claims\_Paid\_Yes\_FVS,

num\_Claims\_Paid\_No\_FVS,

num\_Claims\_Paid\_Yes\_LOC,

num\_Claims\_Paid\_No\_LOC

]

labels = [

f'FVS \n {num\_Claims\_Paid\_Yes\_FVS}',

f'FVS \n {num\_Claims\_Paid\_No\_FVS}',

f'LOC \n {num\_Claims\_Paid\_Yes\_LOC}',

f'LOC \n {num\_Claims\_Paid\_No\_LOC}',

]

#creating bar graph and setting colors

plt.bar(labels, values)

colors = ['green', 'red', 'green', 'red']

plt.bar(labels, values, color=colors)

#adding x and y labels for the bar graph

plt.xlabel('Region')

plt.ylabel('Number of claims')

plt.title('Regional Claims')

#To show the bar graph

plt.show()

**Appendix – Figure 7**

#Filtering the dataframe by claims paid under the respective Regions either paid "1" or not paid "0"

Claims\_Paid\_Yes\_FVS\_AT = df[

(df["Paid"] == '1')

&

(df['Region'] == 'FVS')

&

(df['Category'] == '1')

]

Claims\_Paid\_No\_FVS\_AT = df[

(df['Paid'] == '0')

&

(df['Region'] == 'FVS')

&

(df['Category'] == '1')

]

#An error is showing here, it is supposed to reflect as 5 records that is "Yes" and "0" records as no for Category under "2"

Claims\_Paid\_Yes\_FVS\_XT = df[

(df['Paid'] == '1')

&

(df['Region'] == 'FVS')

&

(df['Category'] == '2')

]

Claims\_Paid\_No\_FVS\_XT = df[

(df['Paid'] == '0')

&

(df['Region'] == 'FVS')

&

(df['Category'] == '2')

]

Claims\_Paid\_Yes\_LOC\_AT = df[

(df['Paid'] == '1')

&

(df['Region'] == 'LOC')

&

(df['Category'] == '1')

]

Claims\_Paid\_No\_LOC\_AT = df[

(df['Paid'] == '0')

&

(df['Region'] == 'LOC')

&

(df['Category'] == '1')

]

Claims\_Paid\_Yes\_LOC\_XT = df[

(df['Paid'] == '1')

&

(df['Region'] == 'LOC')

&

(df['Category'] == '2')

]

Claims\_Paid\_No\_LOC\_XT = df[

(df['Paid'] == '0')

&

(df['Region'] == 'LOC')

&

(df['Category'] == '2')

]

# count the number of claims paid under "1" or "0"

num\_Claims\_Paid\_Yes\_FVS\_AT = len(Claims\_Paid\_Yes\_FVS\_AT)

num\_Claims\_Paid\_Yes\_FVS\_XT = len(Claims\_Paid\_Yes\_FVS\_XT)

num\_Claims\_Paid\_No\_FVS\_AT = len(Claims\_Paid\_No\_FVS\_AT)

num\_Claims\_Paid\_No\_FVS\_XT = len(Claims\_Paid\_No\_FVS\_XT)

num\_Claims\_Paid\_Yes\_LOC\_AT = len(Claims\_Paid\_Yes\_LOC\_AT)

num\_Claims\_Paid\_Yes\_LOC\_XT = len(Claims\_Paid\_Yes\_LOC\_XT)

num\_Claims\_Paid\_No\_LOC\_AT = len(Claims\_Paid\_No\_LOC\_AT)

num\_Claims\_Paid\_No\_LOC\_XT = len(Claims\_Paid\_No\_LOC\_XT)

#creating a table using dictionary of list

df = {

'Region': ['FVS AT', 'FVS XT', 'LOC AT', 'LOC XT'],

'Yes': [num\_Claims\_Paid\_Yes\_FVS\_AT, Claims\_Paid\_Yes\_FVS\_XT, num\_Claims\_Paid\_Yes\_LOC\_AT, num\_Claims\_Paid\_Yes\_LOC\_XT],

'No': [num\_Claims\_Paid\_No\_FVS\_AT, num\_Claims\_Paid\_No\_FVS\_XT, num\_Claims\_Paid\_No\_LOC\_AT, num\_Claims\_Paid\_No\_LOC\_XT],

'\033[1mGrand Total\033[0m': [(num\_Claims\_Paid\_Yes\_FVS\_AT +

num\_Claims\_Paid\_No\_FVS\_AT),

(

Claims\_Paid\_Yes\_FVS\_XT +

Claims\_Paid\_No\_FVS\_XT

),

(

num\_Claims\_Paid\_Yes\_LOC\_AT +

num\_Claims\_Paid\_No\_LOC\_AT

),

(

num\_Claims\_Paid\_Yes\_LOC\_XT +

num\_Claims\_Paid\_No\_LOC\_XT

)

]

}

dftable = pd.DataFrame(df)

print (dftable)