

ANL252 PYTHON FOR DATA ANALYTICS

End-of-Course Assessment

B2271127

Ang Yong Xiang

06 MARCH 2023

Table of Contents

[Question 1 3](#_Toc128996045)

[Output for Q1 3](#_Toc128996046)

[Question 2 3](#_Toc128996047)

[Output (Replacement of Missing Value) 4](#_Toc128996048)

[Output (Removing All Rows With Missing Value) for Q2 4](#_Toc128996049)

[Question 3 5](#_Toc128996050)

[Q3 Output 5](#_Toc128996051)

[Question 4 6](#_Toc128996052)

[First Insight Output 6](#_Toc128996053)

[Second Insight Output 7](#_Toc128996054)

[Third Insight Output 8](#_Toc128996055)

[Question 5 9](#_Toc128996056)

[Question 6 9](#_Toc128996057)

[Appendix 10](#_Toc128996058)

[Q1 Solution 10](#_Toc128996059)

[Q2 Solution 10](#_Toc128996060)

[Solution for replacement of missing value for Q2 10](#_Toc128996061)

[Solution for removing all rows with missing value for Q2 11](#_Toc128996062)

[Question 3 Solution 11](#_Toc128996063)

[Question 4 Solution (1st insight - Top 5 products with the highest claims) 13](#_Toc128996064)

[2nd Insight - Compare total amount of claims paid in year 2021 n 2022 (Jan - June) 15](#_Toc128996065)

[3rd Insight – Compare Total Amount of Claim Paid in Year 2021 and 2022 (Jan - Mar) 18](#_Toc128996066)

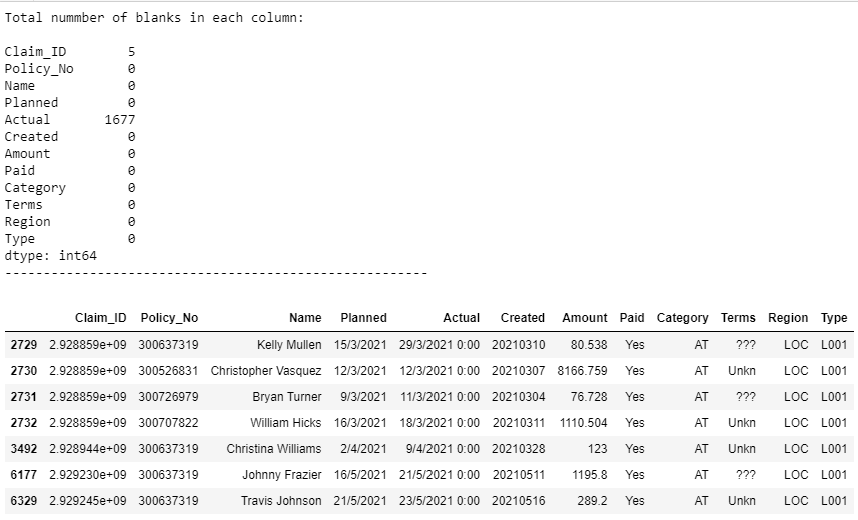
[Question 5 21](#_Toc128996067)

[References 22](#_Toc128996068)

# Question 1

**Claim\_ID**, **Actua**l and **Terms** variables have missing value. (Refer to appendix for the source codes)

## Output for Q1



# Question 2

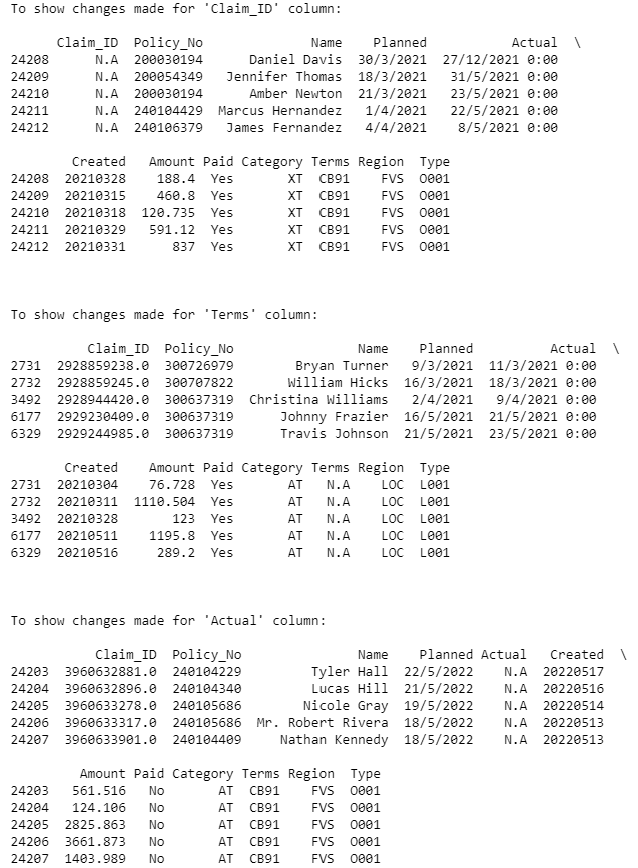
According to Scribbr, there are 3 different types of data. “Missing Completely at random (MCAR)”, “Missing at random (MAR)” and “Missing not at random (MNAR)”. MCAR refers to missing data are randomly distributed across the variable and unrelated to other variables. MAR refers to missing data are not randomly distributed but they are accounted for by other observed variables. Lastly, MNAR refers to missing data systematically differ from the observed values. (Bhandari (2022))

To deal with missing value and tidy up the data, there are different options available such as acceptance, listwise or pairwise deletion and imputation. Importantly, we need to determine the type of missing data because we can accept the data if it’s MCAR or MAR. As for MNAR data, may need more complex treatment.

Personally, I will decide what are the data analysis I would want to perform and choose a wise option to deal missing value without affecting accuracy of my analysis.

For question 2, I will choose “Pairwise Deletion” to deal with the missing value. By adopting pairwise deletion, I am able to keep more data by removing the data points that are missing from analysis. It conserves more of your data because all available data from cases are included. When performing analyses with multiple variables like correlation, only cases with complete data for each variable are included. For this question, I did both replacement of data and dropping of rows with missing data.

## Output (Replacement of Missing Value)

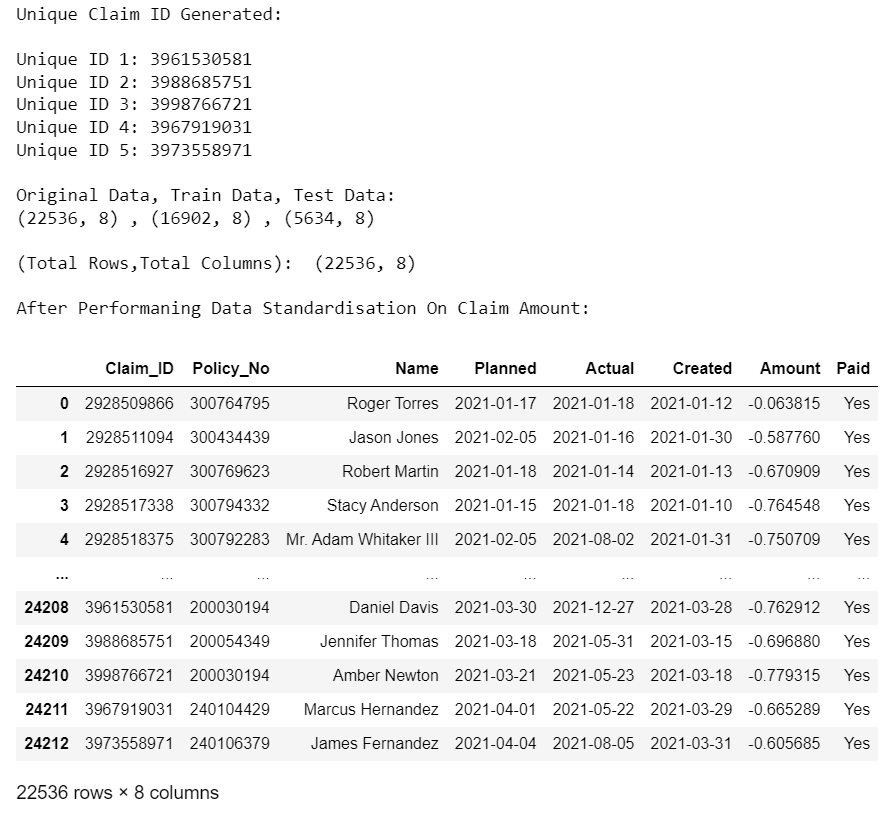


## Output (Removing All Rows With Missing Value) for Q2



# Question 3

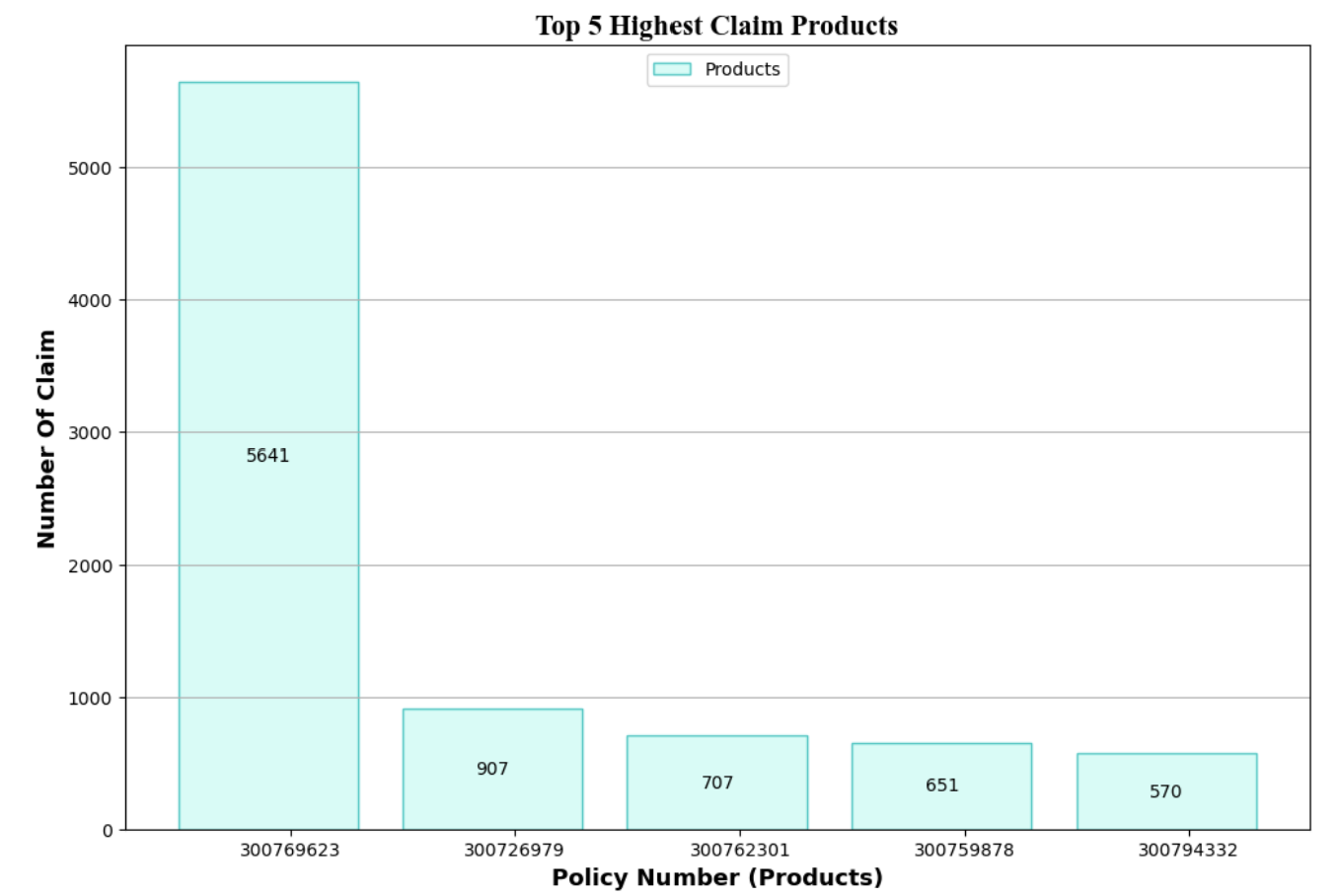
## Q3 Output



I have generated unique claim ID, replaced the missing values, dropped unnecessary columns, converted column data type for the selected columns and perform data standardisation for “Amount” column as well as filtered unnecessary rows for data analysis for the later portion. The purpose of generating unique claim id that is within reasonable value is to assign to the 5 rows with the missing claim id so that I can keep the data for analysis.

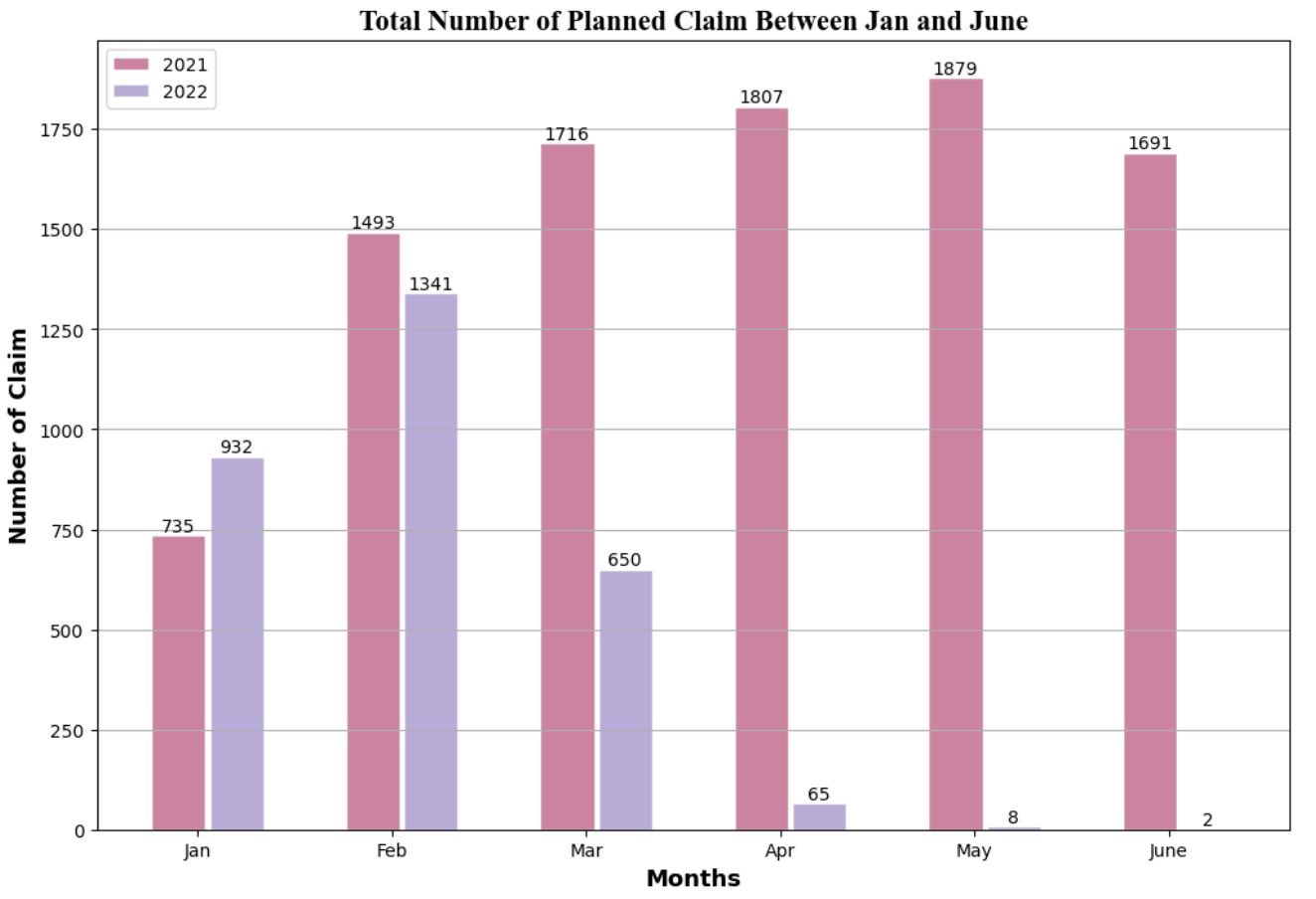
# Question 4

## First Insight Output



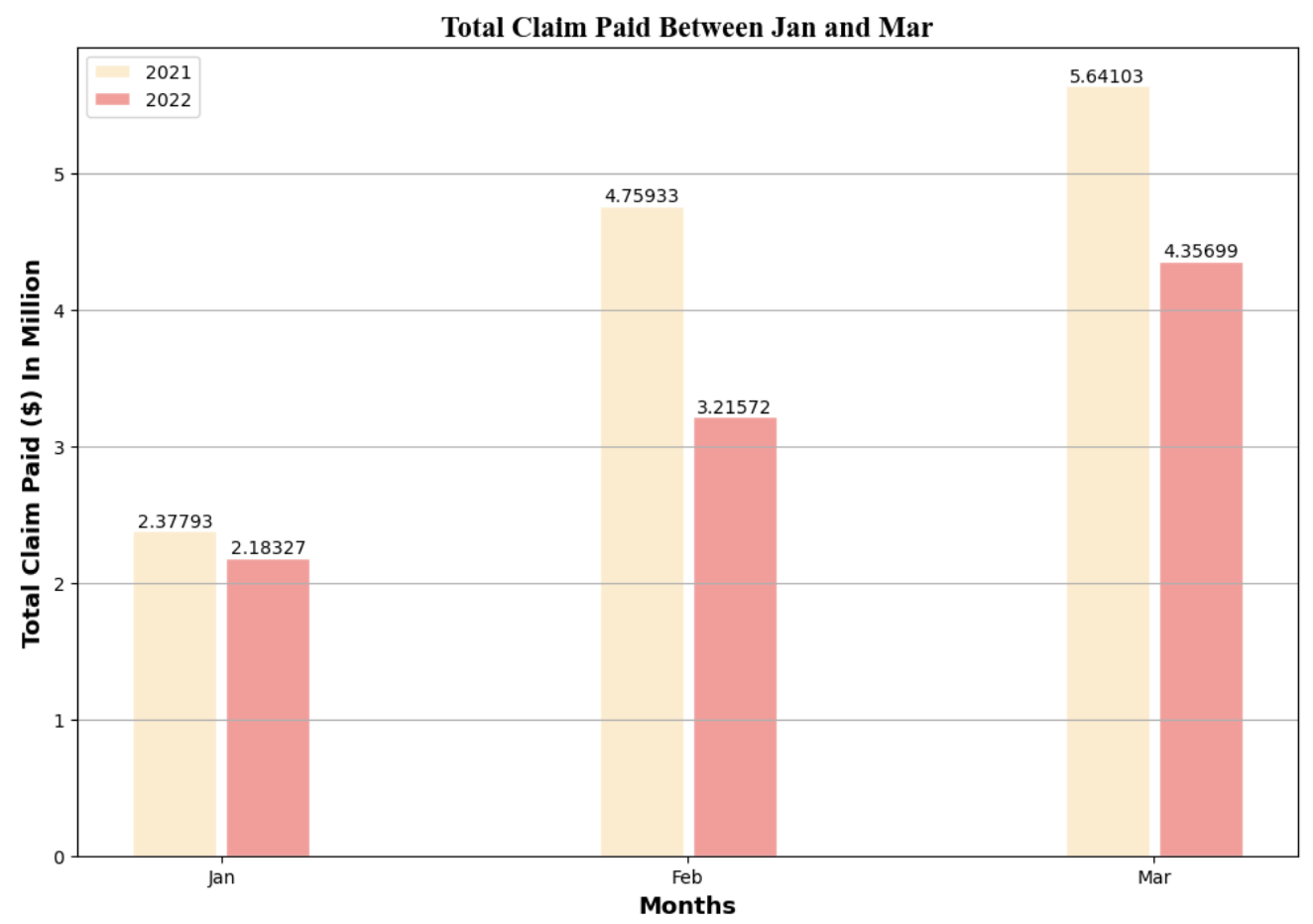
Based on the bar chart generated, the product with the policy number: 300769623 is the policy with the highest number of corporate claims being made. As such, we can deduce that this policy number might be an accident insurance as the chance of people having accidents will be higher as compared to occurrences such as death. Hence, accident insurances will have higher claim rates than people who hold insurances like life insurance. Furthermore, with this data, company can look into how can they improve the products in a way that can benefits the insured.

## Second Insight Output



Based on the bar chart generated, there is a high amount of planned claims made in Feb 2021 and 2022. We can deduce that Feb might be the company’s end of the financial year as most claims are usually made in this month. As such, most claims will be made by the company in this month before the financial year ends. Also, with the data above, company can plan ahead and analyse why there is a huge fall in the number of claims in 2022 and prepare for future.

## Third Insight Output



Based on the bar chart generated, there is a lower number of corporate claims paid in 2022 as compared to in 2021. We can deduce this occurrence is due to the pandemic, Covid 19 improving in 2022. Hence, there are lesser claims made a year after. The claims made are likely to be claims made using hospitalisation insurances as serious conditions of contracting Covid 19 may lead to one being hospitalised.

# Question 5



# Question 6

**Y = mx + c**

Regression line: Y = 0.573x + 6.772

# Appendix

## Q1 Solution

# Import Libraries

import numpy as np

import pandas as pd

# Import ECA.CSV File into DataFrame

dataFrame = pd.read\_csv('ECA.csv')

# To filter out data value with '????' and 'Unkn'

filteredData = dataFrame[dataFrame.isin(['???', 'Unkn']).any(1)]

# To retrieve and display the total number of NULL value for all columns

print('Total number of blanks in each column:\n')

print(dataFrame.isnull().sum())

print("-------------------------------------------------------")

# To Display filtered data frame

filteredData

## Q2 Solution

## Solution for replacement of missing value for Q2

# Import library

import pandas as pd

# Import ECA.CSV File into DataFrame

dataFrame = pd.read\_csv('ECA.csv',na\_values='na\_string', na\_filter=True)

# Define a string to search for and replace 'Unkn' & '???' with 'N.A'

str\_Unkn = 'Unkn'

str\_QnsMark = '???'

dataFrame['Terms'].replace(to\_replace=[str\_Unkn, str\_QnsMark], value='N.A', inplace=True)

dataFrame.fillna('N.A', inplace=True)

# Filtering the dataframe based on a condition

filtered\_ClaimID\_Df = dataFrame[(dataFrame['Claim\_ID'] == 'N.A')]

filtered\_Terms\_Df = dataFrame[(dataFrame['Terms'] == 'N.A')]

filtered\_Actual\_Df = dataFrame[(dataFrame['Actual'] == 'N.A')]

# To show the missing values in dataframe are replaced with "N.A"

print("To show changes made for 'Claim\_ID' column:\n")

print(filtered\_ClaimID\_Df.tail(5))

print("\n\n")

print("To show changes made for 'Terms' column:\n")

print(filtered\_Terms\_Df.tail(5))

print("\n\n")

print("To show changes made for 'Actual' column:\n")

print(filtered\_Actual\_Df.tail(5))

## Solution for removing all rows with missing value for Q2

# Import Library

import pandas as pd

# Import ECA.CSV File into DataFrame

dataFrame = pd.read\_csv('ECA.csv',na\_values='na\_string', na\_filter=True)

# Dropping the rows having NaN values

df = dataFrame.dropna()

# To reset the indices

df = df.reset\_index(drop = True)

# Selecting rows based on condition - Excluding rows with 'Unkn' &'???'

filtered\_Df = df[(df['Terms'] != 'Unkn') & (df['Terms'] != '???')]

# Display filtered Dataframe

filtered\_Df

## Question 3 Solution

# Import libraries

from sklearn.model\_selection import train\_test\_split

import random as rdm

import numpy as np

import pandas as pd

# Import ECA.CSV File into DataFrame

dataFrame = pd.read\_csv('ECA.csv')

# 1st implementation - to generate unique Claim ID for the existing missing values in last 5 rows of Claim\_ID column

def implementation1\_GenerateUniqueClaim\_ID():

maxClaim\_ID = dataFrame['Claim\_ID'].max() # Find maximum value of Claim\_ID

fixed\_digits = 10

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

while(int(random\_Claim\_ID) == int(maxClaim\_ID)):

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

return random\_Claim\_ID

# First Implementation is to replace the 5 missing claim id with generated unique claim IDs

# within range of 3960633901 and 4000000000. Purpose is to retain the existing rows for analysis

unique\_ClaimID\_1 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_2 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_3 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_4 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_5 = implementation1\_GenerateUniqueClaim\_ID()

# To display the 5 unique claim ids generated

print("Unique Claim ID Generated:\n")

print(f'Unique ID 1: ' + str(unique\_ClaimID\_1))

print(f'Unique ID 2: ' + str(unique\_ClaimID\_2))

print(f'Unique ID 3: ' + str(unique\_ClaimID\_3))

print(f'Unique ID 4: ' + str(unique\_ClaimID\_4))

print(f'Unique ID 5: ' + str(unique\_ClaimID\_5))

# To replace the 5 missing claim ID in dataframe

dataFrame.iloc[-1, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_5

dataFrame.iloc[-2, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_4

dataFrame.iloc[-3, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_3

dataFrame.iloc[-4, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_2

dataFrame.iloc[-5, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_1

# Changing Claim\_ID data type to integer type so that the claim\_ID can be seen clearly from the dataframe output

dataFrame['Claim\_ID'] = dataFrame['Claim\_ID'].astype('int64')

# Second implementation is to convert the String date for column "Planned", "Actual" and "Created" to a Date format,

# so that the data can be used for analysis in the later part. And remove the time in all rows for "Actual" column

# and perform Standardisation and split the data into training and testing

dataFrame['Created'] = pd.to\_datetime(dataFrame['Created'], format= '%Y%m%d')

dataFrame['Planned'] = pd.to\_datetime(dataFrame['Planned'], format='%d/%m/%Y')

dataFrame['Actual'] = pd.to\_datetime(dataFrame['Actual']).dt.date

# Third implementation is to dropping rows with missing data in "Actual" column and

# Remove unused columns like Category, Terms,Region and Type and rectify 1762.OO to 1762.00

dataFrame = dataFrame.dropna()

# Removing unused columns between column index 8 to 12

dataFrame.drop(dataFrame.iloc[:, 8:12], inplace=True, axis=1)

# Correct the value for index 3698

dataFrame.at[3698, 'Amount'] = 1762.00

# Convert "Amount" column to numeric data type

dataFrame['Amount'] = pd.to\_numeric(dataFrame['Amount'])

# Standardisation of data using pandas

dataFrame['Amount'] = (dataFrame['Amount'] - dataFrame['Amount'].mean()) / (dataFrame['Amount'].std())

# Split claim into the default proportion of 75% training and 25% testing data in sklearn

dataFrame\_Train, dataFrame\_Test = train\_test\_split(dataFrame, test\_size=0.25, random\_state=0)

# Display the original shape and the number of train and test data

print("\nOriginal Data, Train Data, Test Data:" )

print(dataFrame.shape,",",dataFrame\_Train.shape, ",", dataFrame\_Test.shape)

# Display the total rows and columns of filtered dataframe

print("\n(Total Rows,Total Columns): ", dataFrame.shape)

print("")

# Display the rectified value for index 3698

dataFrame.at[3698, 'Amount']

# Display the data frame

print("After Performaning Data Standardisation On Claim Amount: ")

dataFrame

## Question 4 Solution (1st insight - Top 5 products with the highest claims)

# Import Libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import random as rdm

# Import ECA.CSV File into DataFrame

dataFrame = pd.read\_csv('ECA.csv')

# 1st implementation - to generate unique Claim ID for the existing missing values in last 5 rows of Claim\_ID column

def implementation1\_GenerateUniqueClaim\_ID():

maxClaim\_ID = dataFrame['Claim\_ID'].max() # Find maximum value of Claim\_ID

fixed\_digits = 10

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

while(int(random\_Claim\_ID) == int(maxClaim\_ID)):

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

return random\_Claim\_ID

# First Implementation is to replace the 5 missing claim id with generated unique claim IDs in "Claim\_ID" column

unique\_ClaimID\_1 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_2 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_3 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_4 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_5 = implementation1\_GenerateUniqueClaim\_ID()

# To display the 5 unique ids generated

print(f'Unique ID 1: ' + str(unique\_ClaimID\_1))

print(f'Unique ID 2: ' + str(unique\_ClaimID\_2))

print(f'Unique ID 3: ' + str(unique\_ClaimID\_3))

print(f'Unique ID 4: ' + str(unique\_ClaimID\_4))

print(f'Unique ID 5: ' + str(unique\_ClaimID\_5))

dataFrame.iloc[-1, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_5

dataFrame.iloc[-2, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_4

dataFrame.iloc[-3, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_3

dataFrame.iloc[-4, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_2

dataFrame.iloc[-5, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_1

# Second implementation is to convert the String date for column "Planned", "Actual" and "Created" to a Date format,

# so that the data can be used for analysis in the later part. And remove the time in all rows of "Actual" column

dataFrame['Created'] = pd.to\_datetime(dataFrame['Created'], format= '%Y%m%d')

dataFrame['Planned'] = pd.to\_datetime(dataFrame['Planned'], format='%d/%m/%Y')

dataFrame['Actual'] = pd.to\_datetime(dataFrame['Actual']).dt.date

# Third implementation is to remove rows that don't have the actual date of claim settlement, and

# remove unused columns like Category, Terms, Region and Type

dataFrame.fillna('N.A', inplace=True)

# Displaying DataFrame based on a condition - excluding all rows with 'N.A'

dataFrame = dataFrame[(dataFrame['Actual'] != 'N.A')]

# Changing column data type to integer type

dataFrame['Claim\_ID'] = dataFrame['Claim\_ID'].astype('int64')

# Removing unused columns between column index 8 to 12

dataFrame.drop(dataFrame.iloc[:, 8:12], inplace=True, axis=1)

# Insight 1: To get the top 5 products with the highest number of claims.

# Extract 'Policy\_No' and 'Claim\_ID' columns data for analysis

specificData\_Df = dataFrame[['Policy\_No','Claim\_ID']]

# Group specificData by Policy\_No and sum the total number of frequency for each Policy\_No

grp\_SpecificData\_Df = specificData\_Df.groupby(['Policy\_No']).count().reset\_index()

# Sort grp\_SpecificData\_Df in ascending order

sort\_SpecificData\_Df = grp\_SpecificData\_Df.sort\_values(by=['Claim\_ID'], ascending=False)

top5\_Products = sort\_SpecificData\_Df.head()

# Set height of bar

top = top5\_Products

# set width of bar

barWidth = 0.1

fig,axis = plt.subplots(figsize =(12, 8))

# Set height of bar

fm = top5\_Products['Claim\_ID']

bar1 = np.arange(len(fm))

bar2 = [x + barWidth for x in bar1 - 0.1]

# Make the plot

plt.bar(bar2, fm, color ='#D0FCF6', edgecolor ='#00CBC6', label ='Products')

# Annotate, insert lable value for each bar

axis.bar\_label(axis.containers[0], label\_type='center')

# Adding Xticks,tile, x and y labels

plt.xticks([r + barWidth for r in range(len(fm))], top5\_Products['Policy\_No'])

plt.title("Top 5 Highest Claim Products",fontsize = 16,fontname="Times New Roman", fontweight="bold")

plt.xlabel('Policy Number (Products)', fontweight ='bold', fontsize = 13)

plt.ylabel('Number Of Claim', fontweight ='bold', fontsize = 13)

# Show horizontal Grid line

plt.grid(axis='y')

# Show lengend

plt.legend(loc='upper center')

# Show Chart

plt.show()

## 2nd Insight - Compare total amount of claims paid in year 2021 n 2022 (Jan - June)

# Import Libraries

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import random as rdm

# Import ECA.CSV File into DataFrame

dataFrame = pd.read\_csv('ECA.csv')

# 1st implementation - to generate unique Claim ID for the existing missing values in last 5 rows of Claim\_ID column

def implementation1\_GenerateUniqueClaim\_ID():

maxClaim\_ID = dataFrame['Claim\_ID'].max() # Find maximum value of Claim\_ID

fixed\_digits = 10

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

while(int(random\_Claim\_ID) == int(maxClaim\_ID)):

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

return random\_Claim\_ID

# First Implementation is to replace the 5 missing claim id with generated unique claim IDs in "Claim\_ID" column

unique\_ClaimID\_1 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_2 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_3 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_4 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_5 = implementation1\_GenerateUniqueClaim\_ID()

# Update the value of missing claim Id with the generated unique ID

dataFrame.iloc[-1, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_5

dataFrame.iloc[-2, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_4

dataFrame.iloc[-3, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_3

dataFrame.iloc[-4, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_2

dataFrame.iloc[-5, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_1

# Second implementation is to convert the String date for column "Planned", "Actual" and "Created" to a Date format,

# so that the data can be used for analysis in the later part. And remove the time in all rows of "Actual" column

dataFrame['Created'] = pd.to\_datetime(dataFrame['Created'], format= '%Y%m%d')

dataFrame['Planned'] = pd.to\_datetime(dataFrame['Planned'], format='%d/%m/%Y')

dataFrame['Actual'] = pd.to\_datetime(dataFrame['Actual']).dt.date

# Third implementation is to remove rows that don't have the actual date of claim settlement, and

# remove unused columns like Category, Terms, Region and Type

dataFrame.fillna('N.A', inplace=True)

# Displaying DataFrame based on a condition - excluding all rows with 'N.A'

dataFrame = dataFrame[(dataFrame['Actual'] != 'N.A')]

# Changing Claim\_ID column data type to integer type

dataFrame['Claim\_ID'] = dataFrame['Claim\_ID'].astype('int64')

# Removing unused columns between column index 8 to 12

dataFrame.drop(dataFrame.iloc[:, 8:12], inplace=True, axis=1)

# Extract 'Policy\_No' and 'Claim\_ID' columns data for analysis

specificData\_Df = dataFrame[['Claim\_ID','Planned']]

# Get year and month from the corresponding 'Actual' column

specificData\_Df['Month'] = pd.to\_datetime(specificData\_Df['Planned']).dt.month

# Convert the date to datetime64

specificData\_Df['Planned'] = pd.to\_datetime(specificData\_Df['Planned'], format='%Y-%m-%d')

# Filter data for 2021 - Jan to June

filtered\_specificData\_Df\_2021 = specificData\_Df[specificData\_Df['Planned'].dt.to\_period('M') < '2021-07']

# Filter data for 2022 - Jan to June

filtered\_specificData\_Df\_2022 = specificData\_Df[specificData\_Df['Planned'].dt.to\_period('M') > '2021-12']

# Group specificData by month and get the total counts for 2021

grp\_SpecificData\_Df\_2021 = filtered\_specificData\_Df\_2021.groupby(['Month']).count().reset\_index()

df\_2021 = grp\_SpecificData\_Df\_2021[['Month','Claim\_ID']]

finalised\_df\_2021 = df\_2021.head(6)

# Group specificData by month and get the total counts for 2022

grp\_SpecificData\_Df\_2022 = filtered\_specificData\_Df\_2022.groupby(['Month']).count().reset\_index()

df\_2022 = grp\_SpecificData\_Df\_2022[['Month','Claim\_ID']]

finalised\_df\_2022 = df\_2022.head(6)

# Plotting the dataframe

# set width of bar

barWidth = 0.3

fig,axis = plt.subplots(figsize =(12, 8))

# Set height of bar

yAxis\_2021 = finalised\_df\_2021['Claim\_ID']

yAxis\_2022 = finalised\_df\_2022['Claim\_ID']

# Set position of bar on X axis

bar1 = np.arange(len(yAxis\_2021))

bar2 = [x + barWidth for x in bar1 - 0.1]

bar3 = [x + barWidth for x in bar2]

# Make the plot

plt.bar(bar2,yAxis\_2021,color ='#D77FA1', width=barWidth - 0.02, edgecolor ='white', label ='2021')

plt.bar(bar3, yAxis\_2022, color ='#BAABDA', width=barWidth-0.02, edgecolor ='white', label ='2022')

# Annotate, insert lable value for each bar

axis.bar\_label(axis.containers[0], label\_type='edge')

axis.bar\_label(axis.containers[1], label\_type='edge')

# Adding Xticks

plt.title("Total Number of Planned Claim Between Jan and June",fontsize = 16,fontname="Times New Roman", fontweight="bold")

plt.xlabel('Months', fontweight ='bold', fontsize = 13)

plt.ylabel('Number of Claim', fontweight ='bold', fontsize = 13)

plt.xticks([r + barWidth for r in range(len(yAxis\_2021))], ['Jan', 'Feb', 'Mar','Apr','May','June'])

# Show horizontal Grid line

plt.grid(axis='y')

# Show lengend

plt.legend(loc='upper left')

# Show Chart

plt.show()

df\_Datatable = pd.DataFrame({

'Months': ['Jan','Feb','Mar','Apr','May','June'],

'2021': [735,1493,1716,1807,1879,1691],

'2022': [932,1341,650,65,8,2],

})

print("Data Table:")

df\_Datatable

## 3rd Insight – Compare Total Amount of Claim Paid in Year 2021 and 2022 (Jan - Mar)

# Import Libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import random as rdm

# Import ECA.CSV File into DataFrame

dataFrame = pd.read\_csv('ECA.csv')

# 1st implementation - to generate unique Claim ID for the existing missing values in last 5 rows of Claim\_ID column

def implementation1\_GenerateUniqueClaim\_ID():

maxClaim\_ID = dataFrame['Claim\_ID'].max() # Find maximum value of Claim\_ID

fixed\_digits = 10

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

while(int(random\_Claim\_ID) == int(maxClaim\_ID)):

random\_Claim\_ID = rdm.randrange(3960633901, 4000000000, fixed\_digits)

return random\_Claim\_ID

# First Implementation is to replace the 5 missing claim id with generated unique claim IDs in "Claim\_ID" column

unique\_ClaimID\_1 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_2 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_3 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_4 = implementation1\_GenerateUniqueClaim\_ID()

unique\_ClaimID\_5 = implementation1\_GenerateUniqueClaim\_ID()

# To display the 5 unique ids generated

print(f'Unique ID 1: ' + str(unique\_ClaimID\_1))

print(f'Unique ID 2: ' + str(unique\_ClaimID\_2))

print(f'Unique ID 3: ' + str(unique\_ClaimID\_3))

print(f'Unique ID 4: ' + str(unique\_ClaimID\_4))

print(f'Unique ID 5: ' + str(unique\_ClaimID\_5))

dataFrame.iloc[-1, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_5

dataFrame.iloc[-2, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_4

dataFrame.iloc[-3, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_3

dataFrame.iloc[-4, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_2

dataFrame.iloc[-5, dataFrame.columns.get\_loc('Claim\_ID')] = unique\_ClaimID\_1

# Second implementation is to convert the String date for column "Planned", "Actual" and "Created" to a Date format,

# so that the data can be used for analysis in the later part. And remove the time in all rows of "Actual" column

dataFrame['Created'] = pd.to\_datetime(dataFrame['Created'], format= '%Y%m%d')

dataFrame['Planned'] = pd.to\_datetime(dataFrame['Planned'], format='%d/%m/%Y')

dataFrame['Actual'] = pd.to\_datetime(dataFrame['Actual']).dt.date

# Third implementation is to remove rows that don't have the actual date of claim settlement, and

# remove unused columns like Category, Terms, Region and Type

dataFrame.fillna('N.A', inplace=True)

# Displaying DataFrame based on a condition - excluding all rows with 'N.A'

dataFrame = dataFrame[(dataFrame['Actual'] != 'N.A')]

# Changing column data type to integer type

dataFrame['Claim\_ID'] = dataFrame['Claim\_ID'].astype('int64')

# Removing unused columns between column index 8 to 12

dataFrame.drop(dataFrame.iloc[:, 8:12], inplace=True, axis=1)

# Extract 'Policy\_No' and 'Claim\_ID' columns data for analysis

specificData\_Df = dataFrame[['Claim\_ID','Planned','Amount']]

# Get year and month from the corresponding 'Actual' column

specificData\_Df['Month'] = pd.to\_datetime(specificData\_Df['Planned']).dt.month

# Convert the date to datetime64

specificData\_Df['Planned'] = pd.to\_datetime(specificData\_Df['Planned'], format='%Y-%m-%d')

# To correct the value for index 3698

specificData\_Df.at[3698, 'Amount'] = 1762.00

# Filter data for 2021 - Jan to Mar

filtered\_specificData\_Df\_2021 = specificData\_Df[specificData\_Df['Planned'].dt.to\_period('M') < '2021-07']

# Filter data for 2022 - Jan to Mar

filtered\_specificData\_Df\_2022 = specificData\_Df[specificData\_Df['Planned'].dt.to\_period('M') > '2021-12']

filtered\_specificData\_Df\_2022 = filtered\_specificData\_Df\_2022[filtered\_specificData\_Df\_2022['Planned'].dt.to\_period('M') < '2022-04']

# Converting "Amount" column to numeric and sum up the claim amt for each month in 2022, Jan to Mar

filtered\_specificData\_Df\_2022['Amount'] = pd.to\_numeric(filtered\_specificData\_Df\_2022['Amount'])

finalised\_df\_2022 = filtered\_specificData\_Df\_2022.groupby(['Month'])['Amount'].sum()

df\_2022 = finalised\_df\_2022.head(3)

# Converting "Amount" column to numeric and sum up the claim amt for each month in 2021, Jan to Mar

filtered\_specificData\_Df\_2021['Amount'] = pd.to\_numeric(filtered\_specificData\_Df\_2021['Amount'])

finalised\_df\_2021= filtered\_specificData\_Df\_2021.groupby(['Month'])['Amount'].sum()

df\_2021 = finalised\_df\_2021.head(3)

df\_ClaimAmt = pd.DataFrame({

'Months': ['Jan','Feb','Mar'],

'2021': [df\_2021.min()/1000000,df\_2021.median()/1000000,df\_2021.max()/1000000],

'2022': [df\_2022.min()/1000000,df\_2022.median()/1000000,df\_2022.max()/1000000],

})

# Plotting the dataframe

# set width of bar

barWidth = 0.2

fig,axis = plt.subplots(figsize =(12, 8))

# Set height of bar

yAxis\_2021 = df\_ClaimAmt['2021']

yAxis\_2022 = df\_ClaimAmt['2022']

# Set position of bar on X axis

bar1 = np.arange(len(yAxis\_2021))

bar2 = [x + barWidth for x in bar1 - 0.1]

bar3 = [x + barWidth for x in bar2]

# Make the plot

plt.bar(bar2,yAxis\_2021,color ='#FFEBCC', width=barWidth - 0.02, edgecolor ='white', label ='2021')

plt.bar(bar3, yAxis\_2022, color ='#FF9999', width=barWidth-0.02, edgecolor ='white', label ='2022')

# Annotate, insert lable value for each bar

axis.bar\_label(axis.containers[0], label\_type='edge')

axis.bar\_label(axis.containers[1], label\_type='edge')

# Adding Xticks

plt.title("Total Claim Paid Between Jan and Mar",fontsize = 16,fontname="Times New Roman", fontweight="bold")

plt.xlabel('Months', fontweight ='bold', fontsize = 13)

plt.ylabel('Amount Of Claim Paid ($) In Million', fontweight ='bold', fontsize = 13)

plt.xticks([r + barWidth for r in range(len(yAxis\_2021))], ['Jan', 'Feb', 'Mar'])

# Show horizontal Grid line

plt.grid(axis='y')

# Show lengend

plt.legend(loc='upper left')

# Show Chart

plt.show()

print("Data Table:")

df\_ClaimAmt

## Question 5

# Import Libraries

from sklearn.linear\_model import LinearRegression

import pandas as pd

import numpy as np

# Import CSV file

dataframe = pd.read\_csv('ECA.csv', na\_values=['Unkn', '???'])

# Count the number of missing values in each column

counts = dataframe.isna().sum()

# Drop rows with any missing value

dataframe = dataframe.dropna(axis = 0, how = "any")

# convert to datetime of Planned

dataframe['Planned'] = pd.to\_datetime(dataframe['Planned'], format='%d/%m/%Y')

# convert to yyyy-mm-dd format

dataframe['Planned'] = dataframe['Planned'].dt.strftime('%Y-%m-%d')

#convert datetime of Actual

dataframe['Actual'] = pd.to\_datetime(dataframe['Actual'], format='%d/%m/%Y %H:%M')

# convert to yyyy-mm-dd format

dataframe['Actual'] = dataframe['Actual'].dt.strftime('%Y-%m-%d')

modelR = LinearRegression()

plannedR = pd.to\_datetime(dataframe['Planned']).dt.day.values.reshape(-1, 1)

actualR = pd.to\_datetime(dataframe['Actual']).dt.day.values.reshape(-1, 1)

# Fit the model on the planned and actual data

modelR.fit(plannedR, actualR)

# Print the intercept and coefficients of the model

print("Intercept: ", modelR.intercept\_)

print("Coefficient: ", modelR.coef\_)

#Display dataframe

dataframe

# References

Bhandari, P. (2022, November 11). Missing Data | Types, Explanation, & Imputation. Scribbr. Retrieved March 1, 2023, from <https://www.scribbr.com/statistics/missing-data/>