

**ANL252 Python for Data Analytics**

**ECA JAN23 T05**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Mohamed Nurhaikal Bin Mohamed Shariff** | **J2170095** |

**Group: T05**

**Instructor's Name: Mr. Kumar Munish**

**Submission Date: 06/03/2023**

 ANL252, PYTHON FOR DATA ANALYTICS, J2170095, MOHAMED NURHAIKAL, 06 MARCH 2023.

**Section A**

**Question 1:**

# Import the library to be used

import pandas as pd

# read the excel file into a pandas dataframe

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

# Identify missing values in the dataframe

missing\_values = dfreceipt.isna()

# Count the number of missing values in each column

missing\_counts = missing\_values.sum()

# Print the missing value counts

print(missing\_counts)

The variables that contain missing values are “**Claim\_ID**” and “**Actual**”.

========================== End of Question 1 ==============================

**Question 2:**

# Import the library to be used

import pandas as pd

# Read the excel file into a pandas dataframe

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

# Replace the missing values in Claim\_ID with 0.

dfreceipt['Claim\_ID'].fillna(value=0, inplace=True)

By using data cleaning, this allows removing missing values and correcting errors in a data. For the missing data found in “Claim\_ID”, to replace it with a “0” as the user has already claimed the payment but did not key in the claim ID. The claim is still valid even though there is not a claim ID as under the column “Paid” indicates “Yes”. For the missing data found in “Actual”, the user has not claimed the payment thus there is no date nor has been paid as under the column “Paid” indicates “No”.

========================== End of Question 2 ==============================

**Question 3:**

1. Data Grouping:

Grouping data is often used to summarize and group rows of data based on the values in one or more columns. In this case, grouping the rows by the “Name” column shows the number of claims made by the different people.

# Import the library to be used

import pandas as pd

# Read the excel file into a pandas dataframe

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

# Group the data by the Name column

grouped\_dfreceipt = dfreceipt.groupby(by = ['Name']).mean()

grouped\_dfreceipt

1. Data Standardisation:

By using datetime library, the Actual column which has date “DD/MM/YYYY” and time “0:00”. The time is removed to match the %d/%m/%Y format found in the Planned column so that python can compare both the columns in future analysis.

# Import the library to be used

import pandas as pd

from datetime import datetime

# Read the excel file into a pandas dataframe

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

# Format the columns

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

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

1. Sort Data:

Sorting of data based on the values of certain variables aids in better understanding of the data. In this instant, by sorting them based on the column “Name” and “Paid”. This is to easily visualised if the person has been paid or not.

# Import the library to be used

import pandas as pd

# Read the excel file into a pandas dataframe

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

# Sort the data by Name and Paid in ascending order

sorted\_dfreceipt = dfreceipt.sort\_values(by=['Name', 'Paid'], ascending=True)

print(sorted\_dfreceipt)

========================== End of Question 3 ==============================

**Question 4:**

1. There are relatively high number of claims amounts between 100 to 5000 in the corporate claim processing. Which shows that their customers do not claim large amount of value but rather claim mostly below 5000.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Amount | Below 100 | 100 to 5000 | 5001 to 10000 | 10001 to 20000 |
| Number of claims | 1486 | 15679 | 3795 | 1452 |

1. The insurance company creates the claim process first subsequently key in the “Planned” date of repayment and also key in the “Actual” date of repayment. For example, the first entry, the claim created on the 12/01/2021 and planned to be process on the 17/01/2021 but was actually paid on the 18/01/2021.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Claim\_ID | Policy\_No | Name | Planned | Actual | Created |
| 2928509866 | 300764795 | Roger Torres | 17/1/2021 | 18/1/2021 0:00 | 20210112 |

1. Out of the 24213 corporate claims, 1677 of those are not claimed as they have no actual date of claiming. This shows that the company is actively clearing their insurance claims and not stalling any claim processes. The unclaimed values are either in the process or pending approval from the customers.

|  |  |  |
| --- | --- | --- |
| Total | Claimed (Yes) | Claimed (No) |
| 24213 | 22536 | 1677 |

========================== End of Question 4 ==============================

**Question 5:**

# Import libraries

import pandas as pd

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import mean\_squared\_error, r2\_score

from datetime import datetime

import numpy as np

# Load the data

data = pd.read\_csv("ECA.csv")

data.dropna(inplace=True)

# Remove the "0:00" substring from the date-time column

data['Actual'] = data['Actual'].str.replace(' 0:00', '')

# Print the modified DataFrame

print(data)

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

data['Actual'] = pd.to\_numeric(pd.to\_datetime(data['Actual'], format='%d/%m/%Y'))

data["Delay"] = pd.to\_datetime(data["Actual"]) - pd.to\_datetime(data["Planned"])

data["Delay"] = data["Delay"].apply(lambda x: x.days)

# Select relevant variables

X = data[["Planned", "Actual"]]

y = data["Delay"]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Build the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Evaluate the model on the test data

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Squared Error:", mse)

print("R-squared:", r2)

========================== End of Question 5 ==============================

**Question 6:**

The results of the output would be:

1. Mean Squared Error: 6.67511836470793e-24
2. R-squared: 1.0

The linear regression equation is, Delay = b0 + b1 \* Planned + b2 \* Actual

Where, b0, b1, and b2 are the intercept and coefficients for the variables Planned and Actual, respectively.