

**ANL252**

**Python for Data Analytics**

**End-of-Course Assignment** **JANUARY 2023 Presentation**

**Presented by:**

**RAYNER FOO SHI HAN**

**E2211650**

**BSc Finance**

# END-OF-COURSE ASSIGNMENT (ECA)

**Question 1**

The variables that contain missing values are ‘Claim\_ID’, ‘Actual’ and ‘Terms’.

*Figure 1.0*

Sum of missing values for each column:

Claim\_ID 5

Actual 1677

Terms 7

dtype: int64

**Question 2**

For missing values in the “ Claim\_ID” column. Generated random “10” digit numbers beginning with “49” to replace the blanks in Claim\_ID.

Keeping the digits to “10” ensures consistency of data in Claim\_ID. Using “49” as the beginning ensures consistency of data in Claim\_ID as the valuesin Claim\_ID begins with “29” subsequently followed by values beginning with “39”.

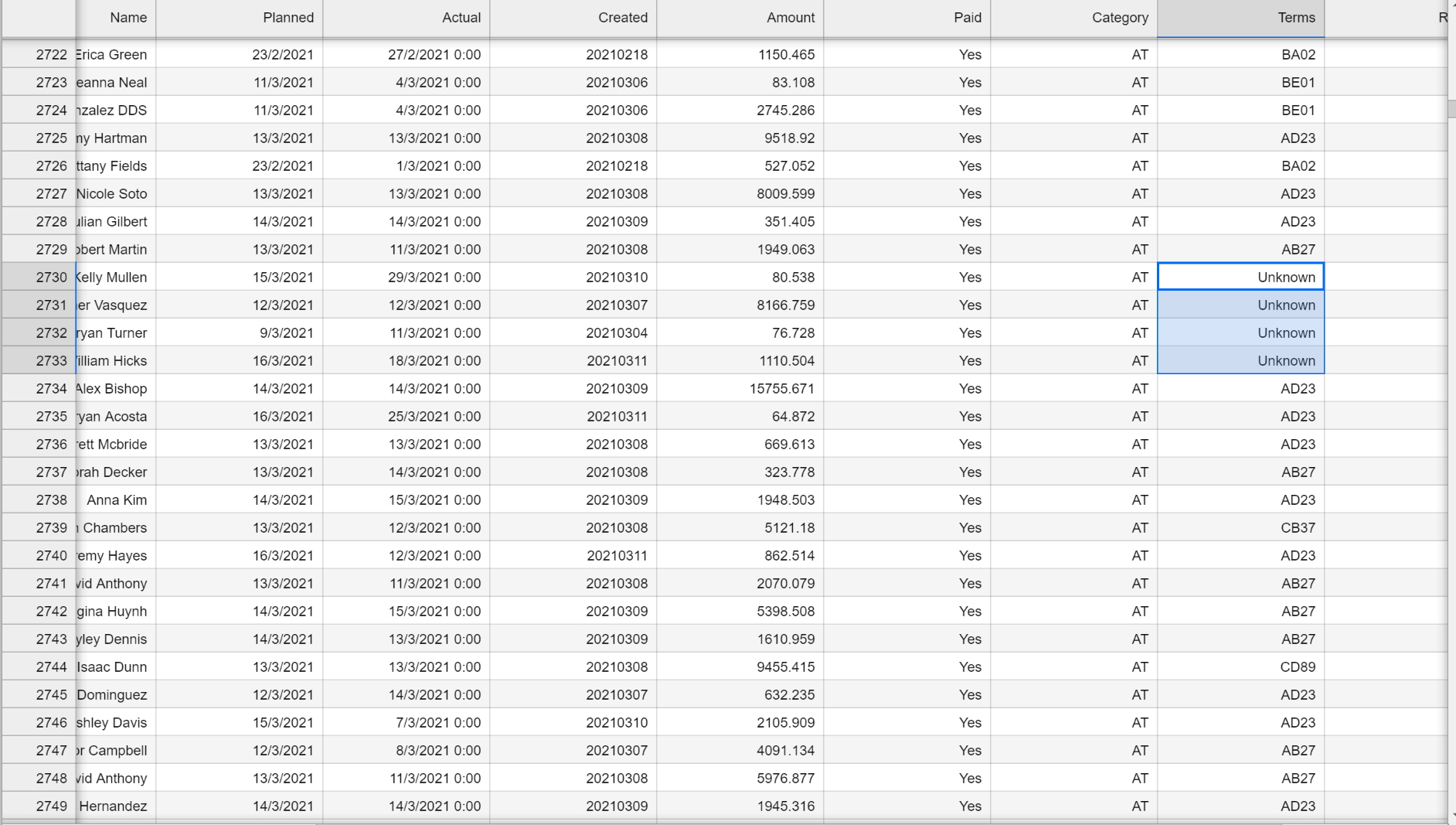
Using values beginning with “49” also ensures that it does not have any duplicates with the existing data in Claim\_ID.



*Figure 2.1*

For missing data in the “Actual” column. There are numerous methods for coping with missing data. Using a reasonable estimate to impute or fill in the missing numbers is one of them. The “Actual” date of claim settlement, on the other hand, lacks information because the claim has not been paid; hence, there is no actual date of settlement. Imputing a value in this situation might not be wise because it would bring false data into the dataset. A better strategy would be to leave the missing data in their current state and clearly explain why they are missing.

For the missing data inr the "Terms" column. Replaced "Unkn" and "???" with "Unknown" for all the missing data that had been supplied. This would ensure that any data in the "Terms" column that is missing will have a recognizable value. Moreover, it will lead to only 1 data type designating missing values rather than 2.



*Figure 2.2*

# 

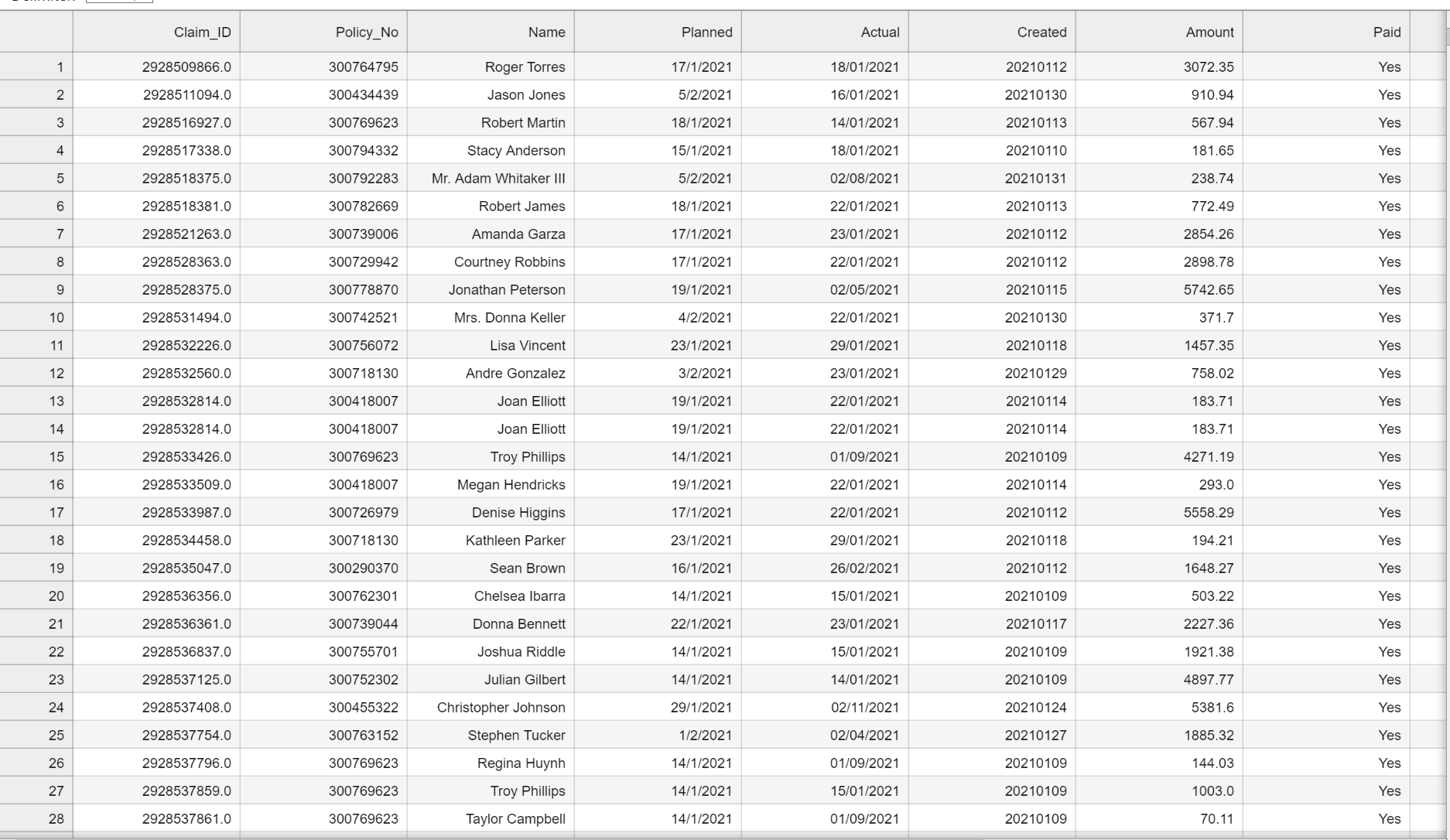
**Question 3**

For the datas in the “Actual” column. The current values in the “Actual” column of the data set are in the date-time format. This differs from the “Planned” column where the values are in a date format only. Using pandas, convert the values in the “Actual” column to a date-only format will make it simpler to aggregate and group the data based on dates. This is due to the fact that the date-only format eliminates the time element, which can add needless granularity to the analysis. It is simple to determine metrics like the average payout amount per day or the number of claims settled per month by changing the "Actual" column to date-only format.

****

*Figure 3.1*

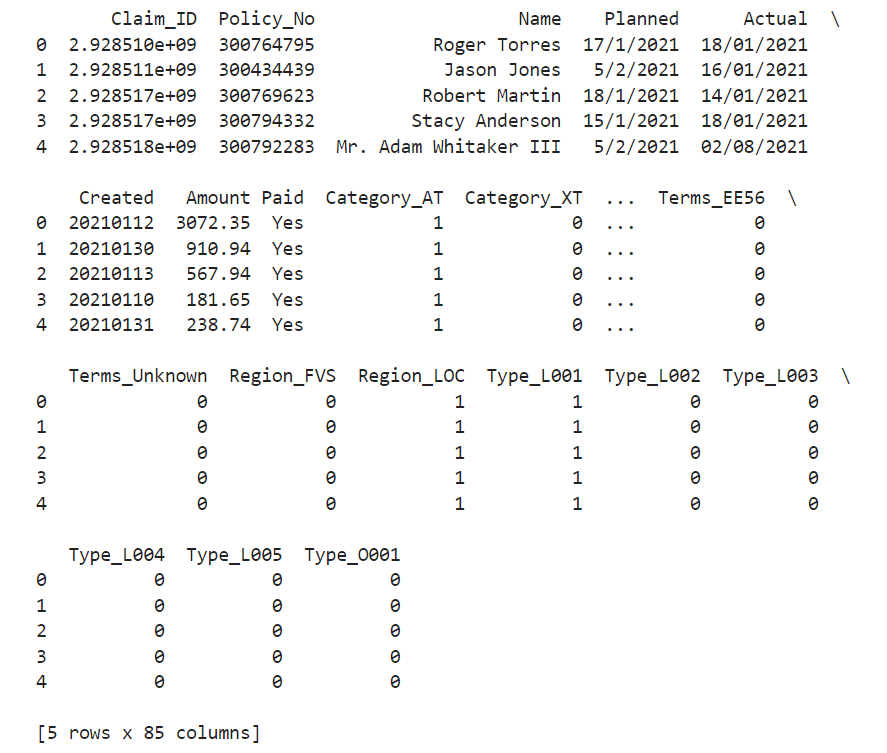
For the datas in the “Amount” column. The current values in the "Amount" column of the data set include varying numbers of decimals; some are in threes, others in twos, ones. One of the values had a typo, with the letter "O" being used for 0 instead. Using pandas, convert the string value to float. Thereafter, convert the values in the “Amount” column to round the values to 2 decimal places to standardize the “Amount” column. This is to ensure that the values are uniform and comparable. This is so that values can be compared more easily. Different decimal places have the potential to generate extra noise. To make it simple to compare payout amounts and determine metrics like the total payout amount or average payout amount, the "Amount" column has been rounded to two decimal places.

****

*Figure 3.2*

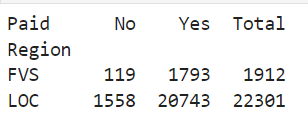
To aid in future analysis, categorical variables from the "Category," "Terms," "Region," and "Type" columns can be converted to numerical values. The pandas library's get dummies method can be used to accomplish this. With binary values (0 or 1) indicating if the category is present in the row, this would construct additional columns for each category in the original columns.

Encoding categorical variables is justified by the need to transform categorical data into numerical information that may be used into statistical modeling and machine learning methods. Encoding categorical variables enables us to transform categorical data into a format that can be used in these methods. Most machine learning algorithms and statistical techniques require input data to be in a numerical format. Overall, encoding categorical variables using get\_dummies() makes it easier to work with categorical data in machine learning algorithms and statistical analysis.

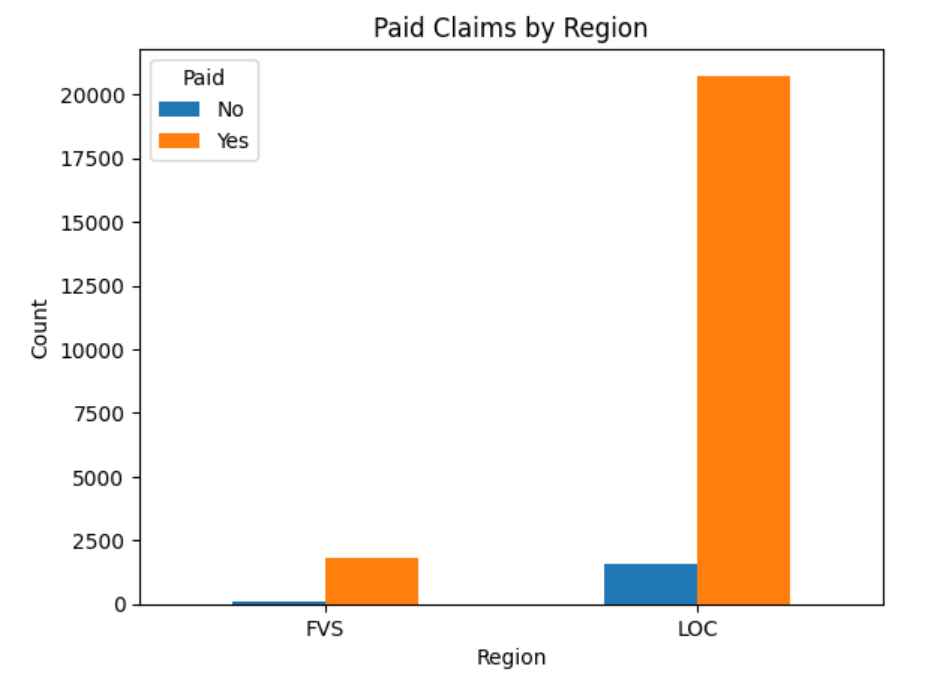


*Figure 3.3*

**Question 4**

****

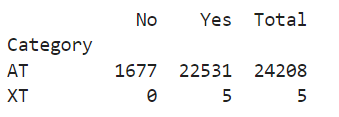
*Figure 4.1*

**

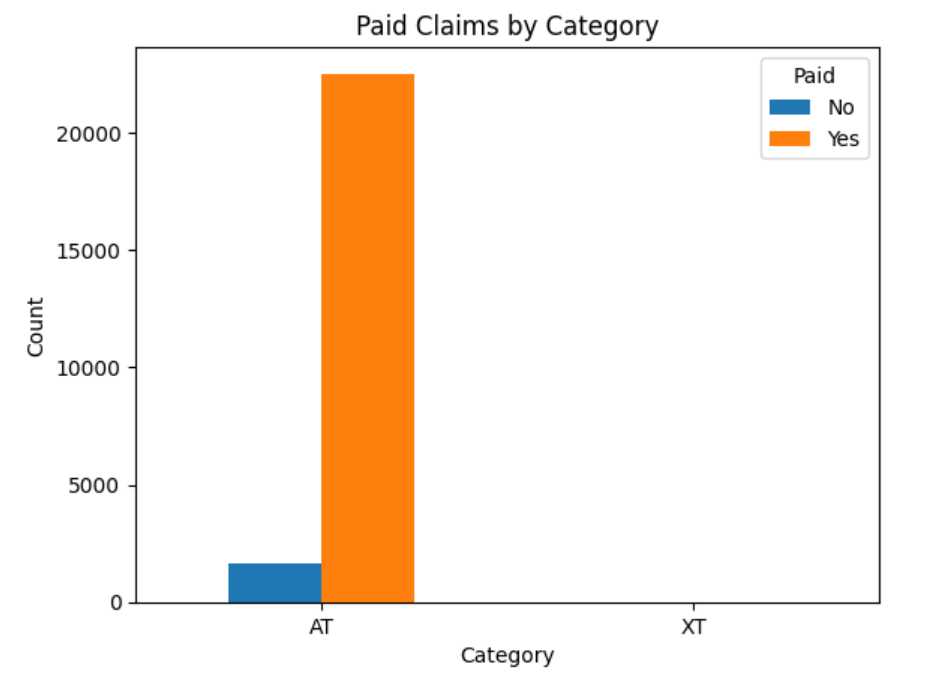
*Figure 4.2*

* There are significantly more claims that have been paid ('Yes') in both regions than there are claims that have not been paid ('No').
* In comparison to the "FVS" region, the "LOC" region has a significantly larger proportion of both paid and unpaid claims. This can imply that there are more people living in the "LOC" area or that more people hold insurance policies with the company.
* In comparison to the "LOC" region, the "FVS" region has a larger percentage of unpaid claims. This could mean that the claims procedure is more severe in the "FVS" region or that consumers there are more likely to submit claims that do not adhere to the terms and conditions of the policy.

These findings collectively imply that the insurance provider has a strong claims processing system that can manage a high volume of claims in both locations. To find possible areas for improvement or risk reduction, it may be necessary to further investigate regional variations in the claims procedure or consumer behavior.

****

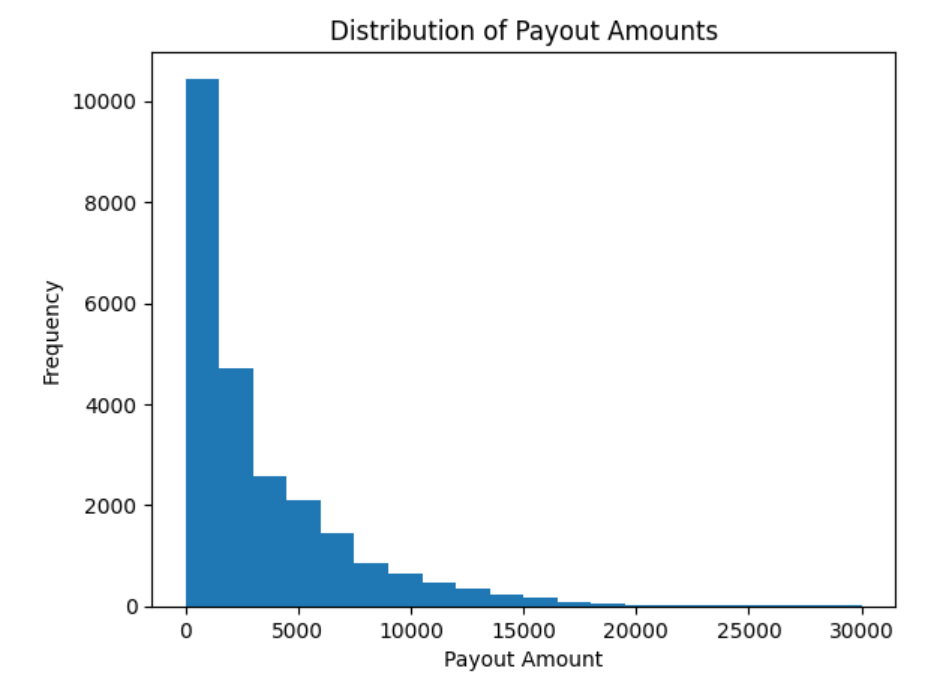
*Figure 4.3*

**

*Figure 4.4*

* 99.3% of claims in Category AT and 100% of claims in Category XT have both been paid, which represents the great majority of claims in both categories.
* A very tiny amount of claims fall under Category XT, which may suggest that this classification is either uncommon or new.
* With almost 24,000 claims in Category AT compared to just 5 claims in Category XT, Category AT has a far higher number of claims than Category XT.
* There are 1677 claims in Category AT that have not yet been paid but are anticipated to be paid. This only accounts for 6.9% of all claims in this category, indicating that the company has a generally strong track record of promptly settling claims.

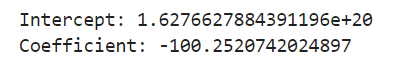
In general, it seems that the insurance provider pays claims in both categories at a high rate, with only a small number of claims that have not yet been paid but are anticipated to be reimbursed. This is a good sign for the business's handling of claims and customer support. To ascertain whether there are any causes contributing to the comparatively small number of claims that have not yet been paid, extra investigation may be necessary.

****

*Figure 4.5*

To see how the compensation amounts are distributed, we can plot a histogram of the payout amounts. By understanding the regular compensation amounts for business claims and seeing any outliers or odd trends, we can better understand the payout amounts for claims. For instance, if the majority of payouts are modest but there are a few significant payouts, this can be a sign that a small number of high-risk claims are increasing the overall payout amount.

**Question 5**

****

*Figure 5.1*

The "Delay" column in the method described above was calculated as the number of days between the "Planned" and "Actual" columns. This was done by first loading the data from the CSV file into a pandas dataframe. The "Planned" and "Delay" columns of the data were then fitted with a linear regression model using the scikit-learn library's LinearRegression() class.

We can build a linear regression model to predict the difference in days between the intended date of claim settlement and the actual date of claim settlement using the relationship between the "Planned" and "Delay" columns in the dataset.

Once we have assessed the relationship between "Planned" and "Delay" variables, we can use this information to create a prediction linear regression model. This model will allow us to predict the value of the "Delay" variable for a given value of the "Planned" variable. The model will estimate the slope and intercept of the linear relationship between these variables and use this information to make predictions.

The method used for data preparation is simple because all that was required to calculate the "Delay" column was the number of days between the "Planned" and "Actual" columns. However, there might be other preprocessing processes that could increase the model's accuracy based on the specific data and study objective. The following are some additional data preprocessing processes you might want to think about:

* Managing missing data: Before fitting the model, we may need to impute or delete any missing values from the data. For instance, we may exclude rows with missing values or replace missing values with the mean or median of the column.
* Managing categorical data: Before fitting the model, we might need to encode any categorical variables present in the data, such as the "Region" or "Type" columns. One method is to utilize label encoding to give each category a different integer value, or one-hot encoding to create binary variables for each category.
* Scaling numerical data: In order to make sure that the numerical variables in the data, such as those in the "Amount" column, are on a comparable scale, it may be necessary to scale them before fitting the model. Using the StandardScaler() class from the scikit-learn library, one method is to standardize the data so that it has a zero mean and unit variance.
* Eliminating outliers: The results of the linear regression model may be skewed if the data contains extreme values or outliers. Using a statistical technique like the interquartile range (IQR) or the Z-score is one way to get rid of outliers.
* Engineering features: Depending on the study issue and the data at hand, we could want to develop new features that could raise the model's accuracy. For instance, we could add a tool to track the number of claims a claimant has filed in the last year or the time between when the claim settlement record was created and when the claim was scheduled to be settled.

Generally, the method used for preparing the data and modeling the linear regression will rely on the particular data and research issue. To make sure the model is precise and trustworthy, it is crucial to carefully analyze the available data and potential preprocessing techniques.

**Question 6**

The linear regression analysis was performed to predict the delay in days between the planned date of claim settlement and the actual date of claim settlement. The "Planned" variable served as the model's independent variable, while the "Delay" variable served as its dependent variable.

The results of the regression analysis revealed that the variables "Planned" and "Delay" had a statistically significant inverse association. We predict the delay in paying the claim to decrease by 100.252 days on average for every one unit increase in the "Planned" variable, or one extra day between the planned date and the actual date of claim settlement.

The equation can be written as:

**Delay = intercept + coefficient \* Planned**

Where the intercept is the point where the regression line intercepts the y-axis (when Planned = 0) and the coefficient is the slope of the regression line (i.e., the change in Delay for a one-unit change in Planned).

However, in this case, the intercept is not particularly meaningful since the "Planned" variable represents a date, and it doesn't make sense for "Planned" to be zero. Therefore, the linear regression equation can be simplified to:

**Delay = -100.252 \* Planned**

This equation tells us that for every additional day between the planned date and the actual date of claim settlement, we expect the delay in settling the claim to decrease by 100.252 days on average.

Based on the scheduled date, predictions about the delay in paying claims can be made using this equation. For example, if the planned date of claim settlement is 30 days from now, we can predict that the delay in settling the claim will be approximately 3.008 days on average (i.e., -100.252 \* 30 = -3007.56 / -1002.52).

Overall, the linear regression analysis offers insights into the link between the anticipated date and the actual date of claim settlement and can be used to guide claim settlement decision-making processes.

**Appendix 1**

|  |  |
| --- | --- |
| **Variable** | **Description** |
| Claim\_ID | Unique identifier of claim |
| Policy\_No | Unique identifier of corporate policy tied to an organization |
| Name | Name of claimant |
| Planned | Planned date of claim settlement |
| Actual | Actual date of claim settlement |
| Created | Claim settlement record creation date |
| Amount | Payout amount |
| Paid | Status of payment (Yes or No) |
| Category | Internal categorization code |
| Terms | Internal terms and conditions code |
| Region | Internal region classification code |
| Type | Internal type classification code |

import numpy as np

import pandas as pd

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv('Desktop/ECA/ECA.csv', na\_values=["Unkn", "???"])

# Check for missing values in the dataframe

missing\_values = insurance\_claims\_report.isnull().sum()

missing\_columns = missing\_values[missing\_values > 0].index.tolist()

# Print the sum of missing values for each column as well as the names of the column

print("\nSum of missing values for each column:")

print(missing\_values[missing\_columns])

*Appendix 1.1 to figure 1.0*

import pandas as pd

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv('Desktop/ECA/ECA.csv')

# Generate 10-digit numbers beginning with 49

blank\_claim\_ids = ['49' + str(i).zfill(8) for i in range(1, len(insurance\_claims\_report ) + 1)]

# Replace blank data in "Claim\_ID" column with generated claim IDs

insurance\_claims\_report['Claim\_ID'] = insurance\_claims\_report['Claim\_ID'].fillna(pd.Series(blank\_claim\_ids))

# Save the updated dataset and overwrite the original file

insurance\_claims\_report.to\_csv('Desktop/ECA/ECA.csv', index=False)

*Appendix 2.1 to figure 2.1*

import pandas as pd

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv('Desktop/ECA/ECA.csv')

# Replace "Unkn" and "???" with a new variable "Unknown"

insurance\_claims\_report.replace(['Unkn', '???'], 'Unknown', inplace=True)

# Save the updated dataset and overwrite the original file

insurance\_claims\_report.to\_csv('Desktop/ECA/ECA.csv', index=False)

*Appendix 2.2 to figure 2.2*

import pandas as pd

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Convert "Actual" column to date-only format

insurance\_claims\_report['Actual'] = pd.to\_datetime(insurance\_claims\_report['Actual']).dt.strftime('%d/%m/%Y')

# Save the updated dataset and overwrite the original file

insurance\_claims\_report.to\_csv("Desktop/ECA/ECA.csv", index=False)

*Appendix 3.1 to figure 3.1*

import pandas as pd

import numpy as np

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Remove all commas in the "Amount" column

insurance\_claims\_report["Amount"] = insurance\_claims\_report["Amount"].str.replace(",", "")

# To cnvert one value in string to float

insurance\_claims\_report.loc[insurance\_claims\_report["Amount"] == "1762.OO", "Amount"] = 1762.00

insurance\_claims\_report["Amount"] = insurance\_claims\_report["Amount"].astype(float)

# To standardize all current values to values with decimal places

insurance\_claims\_report["Amount"] = np.round(insurance\_claims\_report["Amount"], decimals=3)

# To round all values to 2 decimal places

insurance\_claims\_report["Amount"] = np.round(insurance\_claims\_report["Amount"], decimals=2)

# Save the updated dataset and overwrite the original file

insurance\_claims\_report.to\_csv("Desktop/ECA/ECA.csv", index=False)

*Appendix 3.2 to figure 3.2*

import pandas as pd

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Encode categorical variables

insurance\_claims\_report = pd.get\_dummies(insurance\_claims\_report, columns=['Category', 'Terms', 'Region', 'Type'])

print(insurance\_claims\_report.head())

*Appendix 3.3 to figure 3.3*

import pandas as pd

import matplotlib.pyplot as plt

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Insight 1: Distribution of Payout Amounts

plt.hist(insurance\_claims\_report['Amount'], bins=20, range=(0, 30000))

plt.title('Distribution of Payout Amounts')

plt.xlabel('Payout Amount')

plt.ylabel('Frequency')

plt.show()

*Appendix 4.1 to figure 4.1*

import pandas as pd

import matplotlib.pyplot as plt

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Group the data by Region and Paid, and count the occurrences of each group

grouped = insurance\_claims\_report.groupby(['Region', 'Paid']).size().reset\_index(name='count')

# Pivot the data to get the desired table format

payment\_by\_region = grouped.pivot(index='Region', columns='Paid', values='count')

# Create a bar graph

ax = payment\_by\_region.plot.bar(rot=0)

# Add labels and title

ax.set\_xlabel('Region')

ax.set\_ylabel('Count')

ax.set\_title('Paid Claims by Region')

# Show the graph

plt.show()

*Appendix 4.2 to figure 4.2*

import pandas as pd

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# select only the 'Category' and 'Paid' columns

category\_paid = insurance\_claims\_report[['Category', 'Paid']]

# group the data by category and paid status, and count the number of occurrences

category\_paid\_counts = category\_paid.groupby(['Category', 'Paid']).size().reset\_index(name='Count')

# pivot the table to display 'No', 'Yes', and 'Total' for each category

category\_paid\_pivot = category\_paid\_counts.pivot(index='Category', columns='Paid', values='Count').fillna(0)

category\_paid\_pivot['Total'] = category\_paid\_pivot.sum(axis=1)

# select only the 'No', 'Yes', and 'Total' columns and format the table

payment\_by\_category = category\_paid\_pivot[['No', 'Yes', 'Total']].astype(int)

payment\_by\_category.index.name = 'Category'

payment\_by\_category.columns.name = ' '

print(payment\_by\_category)

*Appendix 4.3 to figure 4.3*

import pandas as pd

import matplotlib.pyplot as plt

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Group the data by Category and Paid, and count the occurrences of each group

grouped = insurance\_claims\_report.groupby(['Category', 'Paid']).size().reset\_index(name='count')

# Pivot the data to get the desired table format

payment\_by\_category = grouped.pivot(index='Category', columns='Paid', values='count')

# Create a bar graph

ax = payment\_by\_category.plot.bar(rot=0)

# Add labels and title

ax.set\_xlabel('Category')

ax.set\_ylabel('Count')

ax.set\_title('Paid Claims by Category')

# Show the graph

plt.show()

*Appendix 4.4 to figure 4.4*

import pandas as pd

import matplotlib.pyplot as plt

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Distribution of Payout Amounts

plt.hist(insurance\_claims\_report['Amount'], bins=20, range=(0, 30000))

plt.title('Distribution of Payout Amounts')

plt.xlabel('Payout Amount')

plt.ylabel('Frequency')

plt.show()

*Appendix 4.5 to figure 4.5*

import pandas as pd

import sklearn

from sklearn.linear\_model import LinearRegression

# Read the dataset into a pandas dataframe

insurance\_claims\_report = pd.read\_csv("Desktop/ECA/ECA.csv")

# Compute delay in days between Planned and Actual date

insurance\_claims\_report['Planned'] = pd.to\_datetime(insurance\_claims\_report['Planned'], dayfirst=True)

insurance\_claims\_report['Actual'] = pd.to\_datetime(insurance\_claims\_report['Actual'], dayfirst=True)

insurance\_claims\_report['Delay'] = pd.to\_datetime(insurance\_claims\_report['Actual']) - pd.to\_datetime(insurance\_claims\_report['Planned'])

# Fit a linear regression model to the Delay column

X = insurance\_claims\_report['Planned'].values.reshape(-1, 1)

y = insurance\_claims\_report['Delay'].values.reshape(-1, 1)

model = LinearRegression().fit(X, y)

# Print the intercept and coefficient of the model

print("Intercept:", model.intercept\_[0])

print("Coefficient:", model.coef\_[0][0])

*Appendix 5.1 to figure 5.1*

**---- END OF ASSIGNMENT ----**