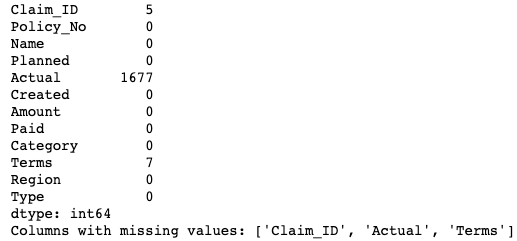
**Question 1**

By reading the dataset, the columns with missing values are “Claim\_ID,” “Actual,” and “Terms.”

The output table is shown below in Figure 1.1

**Figure 1.1**

***Output table showing the number of missing values in each colums***



**Question 2**

The missing data is treated by using the “dropna” function. If filling the missing data with mean, median, or mode is used, the data would not be accurate as the values of “Claim\_ID” and “Actual” are distinctive and unique. Therefore the “dropna” function is being used instead. The output is shown in Figure 2.2

**Figure 2.2**

***Output table of missing values in columns after treating the data***

Table

Description automatically generated with medium confidence

**Question 3**

**3.1**

One of the data preparation tasks that was chosen for further analysis would be converting the data of “Planned,” “Actual,” and “Created” from a string format to datetime format.

This would make it easier to analyze and manipulate the data by being able to calculate the time difference later on. The output table is shown in Figure 3.1.

**Figure 3.1**

***Output table of the formats of the values***

A picture containing table

Description automatically generated

**3.2**

Another data preparation task that was chosen would be calculating the time difference between the “Planned” and “Actual” dates. Calculating the time difference between those two variables and making a new column to the dataset would help us understand how closely the actual dates align with the planned dates and how long the claimer had to wait for their claims to get through. The output table is shown in Figure 3.2

**Figure 3.2**

***Data set with new columns named “Delay.”***

Table

Description automatically generated

**3.3**

The last data preparation task would be to create a new column for the quarters of the year. This is done by splitting the “Created” columns into four quarters of the year. This could help by splitting the data into four quarters of the year to check which quarters of the year the claimers submit more claims than the others. The output table is shown in Figure 3.3

**Figure 3.3**

***Output table of the new column “Quarter.”***

Table

Description automatically generated

**Question 4**

**4.1**

Based on figure 4.1 below, the company has six different types of claims. Many corporate claims come from type L001, with very few claims from the other types. And there is a claim called “O001,” which differs from the others. This could give the insurance company valuable insights if specific departments handle particular types of claims. The company could focus more on “L001” claims to not have any delays as it has the highest number of claims.

**Figure 4.1**

Chart, bar chart, histogram

Description automatically generated

**4.2**

By using the method “pandas.to\_datetime” and having the new column “Delay”, it is able to provide an output of the time difference between the actual date and the planned date of the claim, as shown in figure 4.2 below. The negative values of the x-axis are claims that went through before the planned date, while the positive values are claims that went through only after the planned date. Therefore, as shown in the bar graph above, many claims have come through near the planned date, and a few claims came in before the planned date. There are way lesser claims that came through after the planned date. It may be due to claims having issues and needing time to rectify before getting through, and the insurance company could implement ways to ensure that all the claims will be processed on time.

**Figure 4.2**

Chart

Description automatically generated

**4.3**

By splitting the “Created” columns into quarters, as shown in the graph below in figure 4.3, most claims created come from the 1st quarter of the year, followed by the 2nd quarter. The number of claims for the 3rd and 4th quarters is almost on par. Based on the data above, the insurance company can have a rough idea of the peak period of where claims are at the highest and implement more ideas to ensure claims are being processed on time.

**Figure 4.3**

**Chart, bar chart

Description automatically generated**

**Question 5**

The task is to predict the delay in days in processing claims.

This can be done using the previous data preparation tasks where a new column of “Delay” is made.

The value “x” would be the Payout amount of the claims, and the value “y” would be the delay of the claims.

The data is then split into two parts, one for training and one for testing.

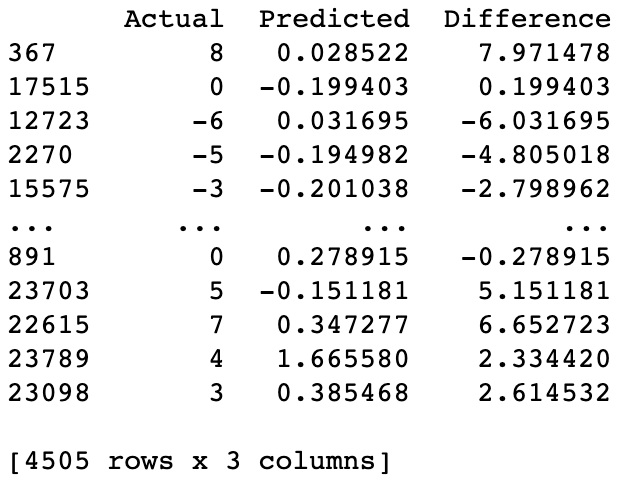
The data is split into 80% for training and 20% for testing.

Next, the regression model is trained using 80% of the training data, and it would predict the delays on the remaining 20% of test data.

Predictions are then made to check if the regression model is accurate in the prediction by comparing the actual delay values with the predicted delay values and showing the difference between those two values in the output table in Figure 5.1.

**Figure 5.1**

***Output table of the difference between actual and predicted values.***



Next, a plot of the predicted delays of the 20% of test data with the actual delays of the 20% of the test data is generated in figure 5.2.

**Figure 5.2**

**Plot on the predicted delays with the actual delays in days**

Chart, scatter chart

Description automatically generated

**Question 6**

**Figure 6.1**

***Output values from the regression model in Figure 5.2***

Text

Description automatically generated with medium confidence

Based on the linear regression model, it has a relatively high mean squared error which means that there is a large difference in the predicted delay of claims as compared to the actual delay of claims.

The linear regression equation would be y=0.0001x – 0.218

***Appendix***

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

import matplotlib.pyplot as plt

**Question 1**

##read in the dataset for missing values denoted by 'Unkn' and '???'

data = pd.read\_csv('/Users/keathchia/Documents/PYTHON/CODE.csv', na\_values=['Unkn', '???'])

##return the number of NaN values in all the columns of the dataset

print(data.isna().sum())

##printing the columns with missing values. Convert values to a list

missing\_columns = data.columns[data.isna().any()].tolist()

print("Columns with missing values:", missing\_columns)

**Question 2**

##Use dropna to delete the missing values

data.dropna(inplace=True)

print("number of missing values in columns after treating")

##print the missing values (should be zero for all variables after treating)

print(data.isnull().sum())

**Question 3**

#3.1#

# Convert all date columns to datetime format using panda to datetime

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

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

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

print(data.dtypes)

#3.2#

##calculating the time difference between Planned and Actual dates, making a new column in the data

data['Delay'] = (data['Actual'] - data['Planned']).dt.days

print(data)

#3.3#

##splitting “created” columns into quarters yearly

data['Quarter'] = data['Created'].dt.quarter

#rounding the values up to the nearest whole number

data['Quarter'].round()

print(data)

**Question 4**

#QUESTION 4.1#

#dictionary to count the frequency of each category

category\_counts = {}

for category in data['Type']:

if category in category\_counts:

category\_counts[category] += 1

else:

category\_counts[category] = 1

plt.bar(category\_counts.keys(), category\_counts.values())

#set plot title and axis labels

plt.title('Distribution of Categories of claims')

plt.xlabel('Type of claims')

plt.ylabel('Number of claims')

plt.show()

#QUESTION 4.2#

##dictionary to count the frequency of each category

category\_counts = {}

for category in data['Delay']:

if category in category\_counts:

category\_counts[category] += 1

else:

category\_counts[category] = 1

plt.bar(category\_counts.keys(), category\_counts.values(),width=10)

##set plot title and axis labels

plt.title('Distribution of Planned date of claim VS Actual date of claim')

plt.xlabel('Time Difference Of Planned Date Vs Actual date')

plt.ylabel('Number of claims')

plt.show()

#QUESTION 4.3#

#dictionary to count the frequency of each category

category\_counts = {}

for category in data['Quarter']:

if category in category\_counts:

category\_counts[category] += 1

else:

category\_counts[category] = 1

plt.bar(category\_counts.keys(), category\_counts.values())

# set plot title and axis labels

plt.title('Quarterly Distribution of Created Claims')

plt.xlabel('Quarters')

plt.ylabel('Number Of Claims')

plt.xticks([1,2,3,4],['Q1,Q2,Q3,Q4'])

plt.show()

**Question 5**

# Replace the invalid values of 1762.OO to 1762.00 as after running the code, there would be error if this value is chosen from the random testing

data['Amount'] = data['Amount'].replace('1762.OO', '1762.00')

x = data[['Amount']]

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=50)

r = LinearRegression()

r.fit(x\_train, y\_train)

# make predictions on the test set

y\_pred = r.predict(x\_test)

#show difference between actual and predicted

output=pd.DataFrame({'Actual':y\_test,'Predicted':y\_pred,'Difference':y\_test - y\_pred})

print(output)

# plot actual vs predicted values

plt.scatter(y\_test, y\_pred)

plt.xlabel('Actual Delay (days)')

plt.ylabel('Predicted Delay (days)')

plt.show()

**Question 6**

# evaluate the performance of the model with MSE, Intercept, and slope value

print('Mean squared error: %.2f' % mean\_squared\_error(y\_test, y\_pred))

print(f"intercept: {r.intercept\_}")

print(f"slope: {r.coef\_}")