Logo

Description automatically generated

**ANL252**

**Python for Data Analytics**

**T05**

End-of-Course Assessment

January 2023 Presentation

|  |  |
| --- | --- |
| Name | PI number |
| Damian Sim Ding Xian | E2281574 |
| Submission Date: | 6/3/2022 |

Table of Contents

[**Q1** 3](#_Toc128962074)

[**Q2** 3](#_Toc128962075)

[**Q3** 4](#_Toc128962076)

[**Q4** 5](#_Toc128962077)

[Chart 1: Bar Chart 5](#_Toc128962078)

[Chart 2: Scatter Plot 6](#_Toc128962079)

[Chart 3: Pie Chart 7](#_Toc128962080)

[**Q6** 9](#_Toc128962081)

[**Appendix** 10](#_Toc128962082)

[Q1 Codes 10](#_Toc128962083)

[Q2 Codes 10](#_Toc128962084)

[Q3 Codes 11](#_Toc128962085)

[Q4 Codes 13](#_Toc128962086)

[Codes for Bar Chart 13](#_Toc128962087)

[Codes for Scatter Plot 14](#_Toc128962088)

[Codes for Pie Chart 15](#_Toc128962089)

[Q5 Codes 18](#_Toc128962090)

[Codes for the Linear Regression table 18](#_Toc128962091)

[Codes for the Linear Regression Model 20](#_Toc128962092)

# **Q1**

The code shows that I have created a data frame for the imported ECA csv file. Afterwards, I created another object called ‘selected\_rows’ to show the rows that contains the specific missing data where the values contain blanks, ‘Unkn’ or ‘???’. The output shows that there is a total of 1689 rows which contains the missing values, and the variables of the missing data are **‘Claim\_ID’**, **‘Actual’** and **‘Terms’**.

# **Q2**

The codes show the output of the updated data frame after data cleaning. Firstly, I have decided to replace the values from ‘Claim\_ID’ that contains blank values to ‘Missing\_ID’. I chose not to delete the rows as ‘Claim\_ID’ is one of the unique identifiers in the dataset. It is not recommended to remove the entire row even if there is no unique identifier as the row of data still contains important information where the claim has been settled and paid which is important for the organisation. It would still be insightful in the analysis and removing these rows could result in a loss of important insights or patterns of the data. Moreover, there is another unique identifier called ‘Policy\_No’ which can still help to identify the rows that does not contain ‘Claim\_ID’. Hence, I thought it would be best to replace the ‘Claim\_ID’ to ‘Missing\_ID’.

Secondly, I have also replaced both the values ‘???’ and ‘Unkn’ from the variable ‘Terms’ to ‘Unknown’ instead of deleting the entire row. This is to ensure that the consistency of the data is being kept as I assume that both of the values ‘???’ and ‘Unkn’ are the same. Hence, I decided to replace it to the same value.

Lastly, I dropped and removed the rows where the claim has not been paid yet. This is because if I was in the view of the organisation, I would only want to have the claims that have been processed and paid, while the ones that have not been paid would be totally unecessary. Therefore, I removed it and loaded into a new CSV file and show the output.

# **Q3**

Data preparation tasks are important as it helps to ensure that the data is formatted properly, which would help to improve the quality and the structure of the data. The three data preparation tasks that can be used to further analyse the data are:

1. Data Formatting

Data formatting helps to ensure that the data used are in the right format based on the values which they have shown for analysis. It is important as it would help to ensure the consistency of the data and avoid having errors which would ensure that the data is interpreted properly. Based on the data given, ‘Actual’ only contains the date, however it is in a datetime format, while ‘Created’ is in another time format which is not consistent to the date format. Hence, I have formatted both ‘Actual’ and ‘Created’ to the same date format as ‘Planned’. I have also realised that ‘Amount’ contains inconsistent decimal places, hence I have decided to keep the values to 2 decimal places instead.

1. Data Grouping

Data grouping would help to find this data to see the claims that belong by the claimant. Data grouping is also essential as it helps to organise the huge amount of data in a meaningful way, where analysing and interpreting would be easier. It will also help to identify the patterns, trends or relationships which could not be shown immediately when looking just at individual data points.

Hence, I have decided to group the data by ‘Name’ as it would be easier to analyse the data and compare the data based on the same person. Therefore, grouping it by the names of the claimant could help the organisation to see important information such as how many claims and the amount of money had paid by the same person.

1. Data Cleaning

Data cleaning is also important as it would help to identify the errors and inconsistencies in the data which would improve the analysis of the data. Based on the CSV file, I found out that the column ‘Amount’ and ‘Type’ have what I assumed as errors where there is a letter O in some of the values. Hence, for the column ‘Amount’, I decided to replace the letter O to the number 0 as ‘Amount’ is showing the amount of money which could likely be an error made by someone. For the column ‘Type’ I replaced the letter O to the letter L as all of the other values in ‘Type’ only starts with the letter L.

# **Q4**Top of Form

## Chart 1: Bar Chart

Chart, bar chart

Description automatically generated

*Figure 1: Bar Chart showing the total amount of money claimed by Year.*

The bar chart above shows the total amount of money paid for the claims for each year which are 2020, 2021 and 2022. I have created a bar chart to show as it is commonly used to display categorical data, which in this case it is showing the amount of money paid by each year from ‘Actual’. I used ‘Actual’ rather than the other 2 date variables as it was stated from the appendix that it refers to the actual date of the claim payment which means the claim has been paid on that specific date.

Based on the chart, the lowest amount paid for the claims is in year 2020 with only a small amount of $326.37. This could mean that the organisation had a relatively low-risk in terms of claims settlements. The highest amount paid for the claims is in year 2021 with an amount of $65581158.20. This could mean that in 2021, there are huge amounts of high claims payments as it might be due to several reasons such as an increase in the number of claims filed, an increase in the severity of claims, or policy or procedure changes made by the organisation. And finally in 2022, the claims amounted has reduced which could mean that the organisation might have improved which could be providing additional training to employees, or revising the policies and procedures which would reduce more claims in the future.

## Chart 2: Scatter Plot

Chart

Description automatically generated

*Figure 2: Scatter Plot showing the relationship between Amount and Planned*

The scatter plot above shows the relationship between the amount of money paid for claims and the planned date for the claim settlement. The scatter plot is used as it helps to identify the patterns and relationships in the data. It can also be used to identify the outliers which are very different from the rest of the data.

Based on the scatter plot, it can be seen that the relationship between the amount and the planned date is strong, as the data points show that they are very close to one another. However, there is an outlier in the data.

## Chart 3: Pie Chart

Chart, pie chart

Description automatically generated

*Figure 3: Pie chart showing the amount of money paid for claims by Type.*

The pie chart above shows the amount of money paid for claims grouped by Type. I have decided to use the pie chart as it helps to show the proportions of the data. It would show how much each term contributes to the entire amount of money, and it is often used to highlight the largest and smallest proportions.

Based on the pie chart, it can be seen that most of the amount of money paid for the claims belong in type ‘L001’ which is 99% of the entire data while the rest of the types are only 1% of the entire data. This could mean that the claims filed are mostly common claims which is known as ‘L001’, while the rest of the claims could be rare cases with different reasons and impact.

**Q5**

**Text

Description automatically generated**

*Figure 4: Results of the Linear Regression Modelling*

*Chart, scatter chart

Description automatically generated*

*Figure 5: Linear Regression Model with the coefficients and intercept*

Figure 4 shows the results of the linear regression modelling table and Figure 5 shows the linear regression model with the coefficients and intercept. Both codes can be found in the appendix under **Q5 Codes**.

# **Q6**

Based on the linear regression results shown above, it shows that there is a negative slope which means that there is a moderate negative non-linear relationship between the delay in days and planned date. Hence, this could mean that the organisation might not have followed the claims payments properly and cause a huge delay in settling the claims. This could also support the reason why the claims are so high as customers could be complaining and requesting for higher claims. Moreover, based on the coefficients and intercept, the linear regression equation is Y = 88367.36 (2 d.p.) – 0.12 (2 d.p.)X.

# **Appendix**

## Q1 Codes

#import library

import pandas as pd

#load the CSV file into a data frame

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

#show the rows which contains missing data

selected\_rows = dataFrame[dataFrame.isnull().any(axis=1) | ((dataFrame == 'Unkn').any(axis=1) | (dataFrame == '???').any(axis=1))]

#show the variables which contains missing data

selected\_columns = dataFrame.loc[:, dataFrame.isnull().any() | (dataFrame == 'Unkn').any() | (dataFrame == '???').any()]

print(selected\_rows)

print(selected\_columns)

## Q2 Codes

#import library

import pandas as pd

#load the CSV file into a data frame

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

#replace the rows where there are no Claim\_ID to Missing\_ID

dataFrame['Claim\_ID'].fillna('Missing\_ID', inplace = True)

dataFrame['Terms'] = dataFrame['Terms'].replace({'???': 'Unknown', 'Unkn': 'Unknown'})

#drop the rows where the claim is not paid

dataFrame = dataFrame.drop(dataFrame[dataFrame['Paid'] == 'No'].index)

#load the updated data frame into a new csv file

dataFrame.to\_csv('ECA2.csv', index = False)

#load the updated CSV into a data frame

updateddataFrame = pd.read\_csv('ECA2.csv')

#show the output

updateddataFrame

## Q3 Codes

#import library

import pandas as pd

import datetime

#load the CSV file into a data frame

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

#replace the rows where there are missing Claim\_ID to Missing\_ID

dataFrame['Claim\_ID'].fillna('Missing\_ID', inplace = True)

dataFrame['Terms'] = dataFrame['Terms'].replace({'???': 'Unknown', 'Unkn': 'Unknown'})

#drop the rows where the claim is not paid

dataFrame = dataFrame.drop(dataFrame[dataFrame['Paid'] == 'No'].index)

#convert the 'Actual' column to datetime format

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

#format the 'Actual' column to dd/mm/yyyy format

dataFrame['Actual'] = dataFrame['Actual'].apply(lambda x: x.strftime('%d/%m/%Y'))

#convert the int 'Created' column to datetime, and format to dd/mm/yyyy

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

#convert the 'Claim\_ID' column to type string

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

# replace 'O' with '0' in the column 'Amount'

dataFrame['Amount'] = dataFrame['Amount'].str.replace('O', '0')

# convert the 'Amount' to float with 2 decimal places

dataFrame['Amount'] = dataFrame['Amount'].astype(float).round(2)

#replace 'O' with 'L' in the column 'Type'

dataFrame['Type'] = dataFrame['Type'].str.replace('O', 'L')

#load the updated data frame into a new csv file

dataFrame.to\_csv('ECA3.csv', index = False)

#group the dataframe by 'Name'

dataFrame = dataFrame.groupby('Name', as\_index = False)

#load the updated CSV into a data frame

updateddataFrame = pd.read\_csv('ECA3.csv')

#show the output

updateddataFrame

## Q4 Codes

### Codes for Bar Chart

#import library

import pandas as pd

import datetime

import matplotlib.pyplot as plt

#load the CSV file into a data frame

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

#replace the rows where there are missing Claim\_ID to Missing\_ID

dataFrame['Claim\_ID'].fillna('Missing\_ID', inplace = True)

dataFrame['Terms'] = dataFrame['Terms'].replace({'???': 'Unknown', 'Unkn': 'Unknown'})

#drop the rows where the claim is not paid

dataFrame = dataFrame.drop(dataFrame[dataFrame['Paid'] == 'No'].index)

#convert the 'Actual' column to datetime format

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

#format the 'Actual' column to dd/mm/yyyy format

dataFrame['Actual'] = dataFrame['Actual'].apply(lambda x: x.strftime('%d/%m/%Y'))

#convert the int 'Created' column to datetime, and format to dd/mm/yyyy

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

#convert the 'Claim\_ID' column to type string

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

# replace 'O' with '0' in the column 'Amount'

dataFrame['Amount'] = dataFrame['Amount'].str.replace('O', '0')

#convert the 'Amount' to float with 2 decimal places

dataFrame['Amount'] = dataFrame['Amount'].astype(float).round(2)

#replace 'O' with 'L' in the column 'Type'

dataFrame['Type'] = dataFrame['Type'].str.replace('O', 'L')

#load the updated data frame into a new csv file

dataFrame.to\_csv('ECA3.csv', index = False)

#group the dataframe by 'Name'

dataFrame = dataFrame.groupby('Name', as\_index = False)

#load the updated CSV into a data frame and convert the 'Created' column to datetime format

updateddataFrame = pd.read\_csv('ECA3.csv', parse\_dates=['Created'], dayfirst=True)

#extract the year from the 'Created' column

updateddataFrame['Created'] = pd.to\_datetime(updateddataFrame['Created']).dt.year

#group the data by year and calculate the total amount of money for each year

grouped\_data = updateddataFrame.groupby('Created')['Amount'].sum()

#create the bar chart using the grouped data

plt.bar(grouped\_data.index, grouped\_data.values)

plt.xlabel('Year')

plt.ylabel('Amount')

plt.title('Total Amount of Money Claimed by Year')

#show the labels for the total amount of money for each year

for i, v in enumerate(grouped\_data.values):

plt.annotate(str(v), xy=(grouped\_data.index[i], v), ha='center', va='bottom')

#output the bar chart

plt.show()

### Codes for Scatter Plot

#import library

import pandas as pd

import datetime

import matplotlib.pyplot as plt

#load the CSV file into a data frame

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

#replace the rows where there are missing Claim\_ID to Missing\_ID

dataFrame['Claim\_ID'].fillna('Missing\_ID', inplace = True)

dataFrame['Terms'] = dataFrame['Terms'].replace({'???': 'Unknown', 'Unkn': 'Unknown'})

#drop the rows where the claim is not paid

dataFrame = dataFrame.drop(dataFrame[dataFrame['Paid'] == 'No'].index)

#convert the 'Actual' column to datetime format

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

#format the 'Actual' column to dd/mm/yyyy format

dataFrame['Actual'] = dataFrame['Actual'].apply(lambda x: x.strftime('%d/%m/%Y'))

#convert the int 'Created' column to datetime, and format to dd/mm/yyyy

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

#convert the 'Claim\_ID' column to type string

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

# replace 'O' with '0' in the column 'Amount'

dataFrame['Amount'] = dataFrame['Amount'].str.replace('O', '0')

#convert the 'Amount' to float with 2 decimal places

dataFrame['Amount'] = dataFrame['Amount'].astype(float).round(2)

#load the updated data frame into a new csv file

dataFrame.to\_csv('ECA3.csv', index = False)

#load the updated CSV into a data frame and convert the 'Created' column to datetime format

updateddataFrame = pd.read\_csv('ECA3.csv')

#convert the planned date to datetime format

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

#extract the year from the 'Planned' column

updateddataFrame['Planned'] = updateddataFrame['Planned'].dt.year

#create a scatter plot of the payout amount vs. the planned date

plt.scatter(updateddataFrame['Planned'], updateddataFrame['Amount'])

#set the x-axis label

plt.xlabel('Planned Settlement Date')

#set the y-axis label

plt.ylabel('Payout Amount')

#set the chart title

plt.title('Relationship between Payout Amount and Planned Settlement Date')

#display the scatter plot

plt.show()

### Codes for Pie Chart

#import library

import pandas as pd

import datetime

import matplotlib.pyplot as plt

import seaborn as sns

#load the CSV file into a data frame

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

#replace the rows where there are missing Claim\_ID to Missing\_ID

dataFrame['Claim\_ID'].fillna('Missing\_ID', inplace = True)

dataFrame['Terms'] = dataFrame['Terms'].replace({'???': 'Unknown', 'Unkn': 'Unknown'})

#drop the rows where the claim is not paid

dataFrame = dataFrame.drop(dataFrame[dataFrame['Paid'] == 'No'].index)

#convert the 'Actual' column to datetime format

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

#format the 'Actual' column to dd/mm/yyyy format

dataFrame['Actual'] = dataFrame['Actual'].apply(lambda x: x.strftime('%d/%m/%Y'))

#convert the int 'Created' column to datetime, and format to dd/mm/yyyy

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

#convert the 'Claim\_ID' column to type string

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

# replace 'O' with '0' in the column 'Amount'

dataFrame['Amount'] = dataFrame['Amount'].str.replace('O', '0')

#replace 'O' with 'L' in the column 'Type'

dataFrame['Type'] = dataFrame['Type'].str.replace('O', 'L')

#convert the 'Amount' to float with 2 decimal places

dataFrame['Amount'] = dataFrame['Amount'].astype(float).round(2)

#load the updated data frame into a new csv file

dataFrame.to\_csv('ECA3.csv', index = False)

#group the dataframe by 'Name'

dataFrame = dataFrame.groupby('Name', as\_index = False)

#load the updated CSV into a data frame and convert the 'Created' column to datetime format

updateddataFrame = pd.read\_csv('ECA3.csv')

#extract the columns from the CSV file

amounts = updateddataFrame['Amount']

terms = updateddataFrame['Type']

#group the data by terms and aggregate the amounts

grouped\_data = updateddataFrame.groupby('Type')['Amount'].sum()

#generate random colors for each term

num\_terms = len(grouped\_data)

colors = sns.color\_palette("Set2", num\_terms)

#create pie chart

fig, ax = plt.subplots()

wedges, labels, autopct = ax.pie(grouped\_data, colors=colors, labels=grouped\_data.index, autopct='%1.1f%%')

#set title and legend

ax.set\_title('Amount of money paid for claims by Type')

#output the pie chart

plt.show()

## Q5 Codes

### Codes for the Linear Regression table

#import library

import pandas as pd

import datetime

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

from sklearn.linear\_model import LinearRegression

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

#load the CSV file into a data frame

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

#replace the rows where there are missing Claim\_ID to Missing\_ID

dataFrame['Claim\_ID'].fillna('Missing\_ID', inplace = True)

dataFrame['Terms'] = dataFrame['Terms'].replace({'???': 'Unknown', 'Unkn': 'Unknown'})

#drop the rows where the claim is not paid

dataFrame = dataFrame.drop(dataFrame[dataFrame['Paid'] == 'No'].index)

#convert the 'Actual' column to datetime format

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

#format the 'Actual' column to dd/mm/yyyy format

dataFrame['Actual'] = dataFrame['Actual'].apply(lambda x: x.strftime('%d/%m/%Y'))

#convert the int 'Created' column to datetime, and format to dd/mm/yyyy

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

#convert the 'Claim\_ID' column to type string

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

# replace 'O' with '0' in the column 'Amount'

dataFrame['Amount'] = dataFrame['Amount'].str.replace('O', '0')

#replace 'O' with 'L' in the column 'Type'

dataFrame['Type'] = dataFrame['Type'].str.replace('O', 'L')

#convert the 'Amount' to float with 2 decimal places

dataFrame['Amount'] = dataFrame['Amount'].astype(float).round(2)

#load the updated data frame into a new csv file

dataFrame.to\_csv('ECA3.csv', index = False)

#group the dataframe by 'Name'

dataFrame = dataFrame.groupby('Name', as\_index = False)

#load the updated CSV into a data frame and convert the 'Created' column to datetime format

updateddataFrame = pd.read\_csv('ECA3.csv')

#convert the date columns into datetime format

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

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

#create a new column for delay days

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

#select relevant features

x = updateddataFrame[['Delay Days']]

y = updateddataFrame['Amount']

#assign names to x

x.columns = ['Delay Days']

#split the data into both training and testing sets

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

#train a linear regression model

model = LinearRegression()

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

#evaluate the performance of the model

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

rmse = mean\_squared\_error(y\_test, y\_pred, squared=False)

mae = mean\_absolute\_error(y\_test, y\_pred)

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

print('RMSE:', rmse)

print('MAE:', mae)

print('R-squared:', r2)

#predict the delay days for a new claim

new\_claim = [[10]] # Specify the delay days for the new claim

new\_claim\_df = pd.DataFrame(new\_claim, columns=x.columns)

predicted\_claim\_amount = model.predict(new\_claim\_df)

print('Predicted Claim Amount:', predicted\_claim\_amount)

### Codes for the Linear Regression Model

#import library

import pandas as pd

import datetime as dt

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

#load the CSV file into a data frame

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

#replace the rows where there are missing Claim\_ID to Missing\_ID

dataFrame['Claim\_ID'].fillna('Missing\_ID', inplace = True)

dataFrame['Terms'] = dataFrame['Terms'].replace({'???': 'Unknown', 'Unkn': 'Unknown'})

#drop the rows where the claim is not paid

dataFrame = dataFrame.drop(dataFrame[dataFrame['Paid'] == 'No'].index)

#convert the 'Actual' column to datetime format

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

#format the 'Actual' column to dd/mm/yyyy format

dataFrame['Actual'] = dataFrame['Actual'].apply(lambda x: x.strftime('%d/%m/%Y'))

#convert the int 'Created' column to datetime, and format to dd/mm/yyyy

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

#convert the 'Claim\_ID' column to type string

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

# replace 'O' with '0' in the column 'Amount'

dataFrame['Amount'] = dataFrame['Amount'].str.replace('O', '0')

#replace 'O' with 'L' in the column 'Type'

dataFrame['Type'] = dataFrame['Type'].str.replace('O', 'L')

#convert the 'Amount' to float with 2 decimal places

dataFrame['Amount'] = dataFrame['Amount'].astype(float).round(2)

#load the updated data frame into a new csv file

dataFrame.to\_csv('ECA3.csv', index = False)

#group the dataframe by 'Name'

dataFrame = dataFrame.groupby('Name', as\_index = False)

#load the updated CSV into a data frame and convert the 'Created' column to datetime format

updateddataFrame = pd.read\_csv('ECA3.csv')

#convert the date columns into datetime format

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

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

#create a new column for delay days

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

#convert 'Planned' to ordinal values and create a new DataFrame with both columns

data = pd.DataFrame()

data['Planned'] = updateddataFrame['Planned'].apply(lambda x: x.toordinal())

data['Delay Days'] = updateddataFrame['Delay Days']

#fit the linear regression model

regressor = LinearRegression()

regressor.fit(data[['Planned']], data['Delay Days'])

#make predictions using the model

y\_pred = regressor.predict(data[['Planned']])

#convert ordinal values back to datetime format

data['Planned'] = pd.to\_datetime(data['Planned'].apply(lambda x: dt.date.fromordinal(x)))

#plot the data points and regression line

plt.scatter(data['Planned'], data['Delay Days'])

plt.plot(data['Planned'], y\_pred, color='red')

plt.xlabel('Planned date')

plt.ylabel('Delay (days)')

plt.title('Linear Regression Model')

plt.show()

#print the coefficients of the regression equation

print('Coefficients: \n', regressor.coef\_)

print('Intercept: \n', regressor.intercept\_)