PROTIUM_CUSTOMERS_DPD_LOAN HISTORY ANALYSIS PROJECT.

1.Importing the standard libraries.

2.Parsing the xml files for making them into the dataframes as of supporting to the data analysing.

3.Making the Percentages of trades done by the customers.

4.Calculating the sum of total distributed loans for each customer.

5.Finding the Maximum number of months for DPDs.

6.Extracting the analysed data into the Excel(xlsx) format.

****// IMPORTING THE LIBRARIES //****

In [1]:

```
import requests
import xml.etree.ElementTree as ET
import pandas as pd
import re
```

----------Reading the Data----------

In [2]:

```
#Parse xml tree.

tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer14235_loan14235_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer16475_loan16475_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer40409_loan40409_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer773504_loan774538_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer787561_loan788638_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer794397_loan795497_crif_report.html.xml")
```

```
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer898231_loan899591_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer1113697_loan1115483_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer1129550_loan1131339_crif_report.html.xml")
tree = ET.parse(r"C:\Users\yuvak\OneDrive\Desktop\Coding_data\Coding_data\customer1195586_loan1197471_crif_report.html.xml")
root = tree.getroot()
```

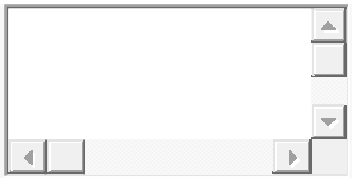
---------Creating Account Type and Combined Payment history---------

In [3]:

```
def CreateDF(root):
    """
    Function that creates DataFrame with Account Type and Combined payment history.
    param root: root of tree
    var loanDetails: list with Acc Type and payment history for single iter
    var trade_paymentHistory:  list of loanDetails for all iter
    returns trade_paymentHistory_DF: final dataframe
    """

    loanDetails = []
    trade_paymentHistory = []

    for element in root.iter('LOAN-DETAILS'):

        trade = element.find('ACCT-TYPE').text
        paymentHistory = element.find('COMBINED-PAYMENT-HISTORY').text
        loanDetails = [trade, paymentHistory]
        trade_paymentHistory.append(loanDetails)

    trade_paymentHistory_DF = pd.DataFrame(trade_paymentHistory, columns=['ACCT-TYPE', 'COMBINED-PAYMENT-HISTORY'])

    return(trade_paymentHistory_DF)
```

----------..Generating DPD Lists along with 30+DPD..---------

In [4]:



```
def GenerateDPDList(string):
    """
    Method that takes payment history as parameter and extracts DPD from each month and returns them as list
    var regex : generates str between "," and "/". this is date
    returns match: list of all regex
    """

    regex = '(?<=\,)(.*?)(?=\/)'
    match = re.findall(regex, string)
#print(match, len(match))

    return(match)

def FindDPD(lst):
    """
    Method that takes lits from GenerateDPDList and returns list 30+DPD
    """
    l = []
    for elm in lst:

#ignore strings and 000s.

        if elm not in ['DDD', 'XXX', 'STD', '000']:
#print(elm).
            if int(elm) >30:
#check for >30.
                l.append(int(elm))
    return(l)

#a= FindDPD(GenerateDPDList([])).
#print(FindDPD(GenerateDPDList([])), len(a)).
```

1.PERCENTAGE OF TRADES DONE BY CUSTOMERS.

In [5]:

```
#1) What percentage of trades are with 30+ DPD (more than 30 days past due) among all the trades availab
le?
#df is DataFrame with Account Type and Combined payment history.
df = CreateDF(root)
```

```
#print(df).

dpdMonths = []
totalMonth = []

for i in range(len(df)):

  elm = df.loc[i][1]
  if elm is None:
    totalMonth.append(0)
    dpdMonths.append(0)

  if elm is not None:

#print(elm).
#t1 is list of dpd for each trades payment history.
    tl = GenerateDPDList(elm)

#print(tl, len(tl)).
#list of total months for all trades. logic behind this is there will be months equal to all strings between "," an
d "/". that is len of list.
    totalMonth.append(len(tl))

#list of 30+dpds.
    dpd = FindDPD(tl)

#print(dpd, len(dpd)).
#list of 30+dpds months.
    dpdMonths.append(len(dpd))

df['Total months of Loan tenure'] = totalMonth
df['number of dpd occurance'] = dpdMonths

#print(df).

# summarise no of month data(total months, 30+dpd months) with trade(loan type).
DF = df.groupby('ACCT-TYPE').agg({'number of dpd occurance': 'sum', 'Total months of Loan tenure' : 'su
m'})

#print(DF).
DF['Percentage'] = DF['number of dpd occurance']*100/DF['Total months of Loan tenure']
```

2.SUM OF TOTAL DISTRUBUTED AMOUNT TO EACH.

In [6]:

```
#2)What is the sum of total disbursed amount for all loans  for each customer?
def FindSumOFAllDispersedLoanAmount(root):
    """
    Function takes root of tree as parameter and returns sum of disbursed amount to a customer throught t
heir credit lifecycle.
    """

    Sum = 0
    for time in root.iter('DISBURSED-AMT'):
        Sum += int(time.text.replace(',',''))
    return(Sum)

#for time in root.iter('SCORE-VALUE'):
#print(time.text).

#sum of total disbursed amount for all loans  for each customer
Sum = FindSumOFAllDispersedLoanAmount(root)
print(Sum)
```

```
8118705
```

3.MAXIMUM NUMBER OF MONTHS.

In [7]:

```
#3)What is the maximum number of months of 30+ due per trade was there?
#According to the Account-type having number of dpd occurances.
DF2 = df.groupby('ACCT-TYPE').agg({'number of dpd occurance': 'max'})
```

In [8]:

```
#Resulting data in the dataframe DF.
DF
```

Out[8]:

ACCT-TYPE	number of dpd occurance	Total months of Loan tenure	Percentage
Auto Loan (Personal)	0	11	0.000000
Credit Card	0	121	0.000000
Gold Loan	16	138	11.594203
Housing Loan	0	36	0.000000
Personal Loan	0	36	0.000000

In [9]:

#Resulting data in the dataframe DF2.

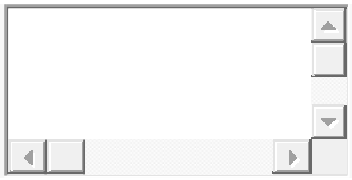
DF2

Out[9]:

ACCT-TYPE	number of dpd occurance
Auto Loan (Personal)	0
Credit Card	0
Gold Loan	3
Housing Loan	0
Personal Loan	0

CONVERTING THE DATA INTO EXCEL.

In [10]:



```
#Converting the data result into the excel .xlsx format.
DF.to_excel('DF.xlsx', sheet_name='sheet1', index=False)
DF2.to_excel('DF2.xlsx', sheet_name='sheet2', index=False)
```