Assignment 4

FE 520 - Into to Python for Financial Applications

Parth Parab
MS CS '21
10444835

November 30, 2020

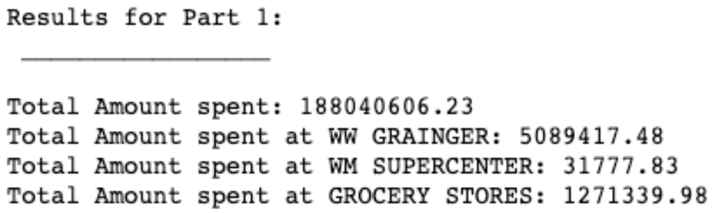
Q1. Credit Transaction data

We start with importing *res_purchase_2014.csv* into pandas data-frame

1. We drop all NaN values to avoid ambiguity and use string manipulating to remove (), $ and words from number cells before performing sum operation

2, 3 & 4. Here we loop through Data-frame column to find the value in question and set a counter to add its value

Results



```
Results for Part 1:
 __________________

Total Amount spent: 188040606.23
Total Amount spent at WW GRAINGER: 5089417.48
Total Amount spent at WM SUPERCENTER: 31777.83
Total Amount spent at GROCERY STORES: 1271339.98
```

Final output from Q1.py

(Continue)

Q2. Credit Transaction data

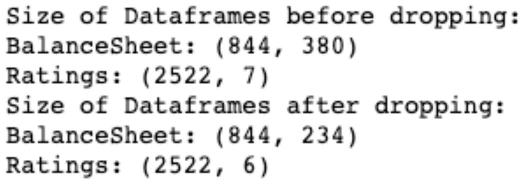
2.1 Read 'Energy.xlsx' and 'EnergyRating.xlsx' as BalanceSheet and Ratings(data-frame)

Here we start with importing the two xlsx files into pandas data-frame using the *pd.read_excel* function of pandas

2.2. Drop the column if more than 90% value in this column is 0 (or missing value).

The logic used here is to add all value 0(zero), isnull (for null) and isna (for NaN) and divide them by the length of the data-frame. If this value is greater than 0.9 (90%) then we drop the column

Result -



```
Size of Dataframes before dropping:
BalanceSheet: (844, 380)
Ratings: (2522, 7)
Size of Dataframes after dropping:
BalanceSheet: (844, 234)
Ratings: (2522, 6)
```

2.3. Replace all None or NaN with average value of each column.

The solution for this question follow-ups with 2.2 where we use the *fillna* function of pandas to replace all NaN/None values with the mean of all columns of the data frame respectively

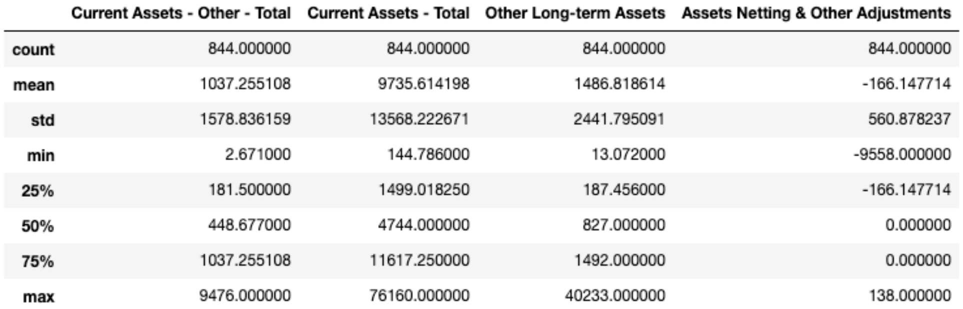
(Continue)

2.4. Normalize the table

We first use numpy to check for columns with numerical values - the result of this is a list of columns which has numerical values. We then use a lambda function to apply Normalization for each column using the formula *(x - np.min(x)) / (np.max(x) - np.min(x))*

2.5. Define an apply function to return the statistical information for variables = ['Current Assets - Other - Total', 'Current Assets - Total', 'Other Long-term Assets', 'Assets Netting & Other Adjustments'], you need to return a data-frame which has exactly same format with pandas method .describe().

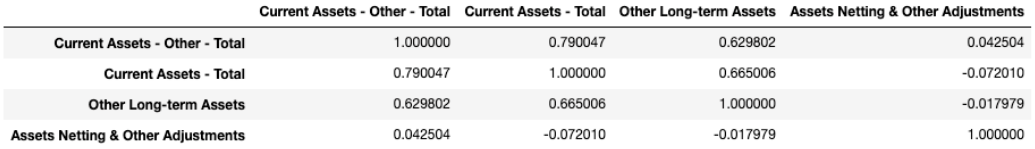
The describe function includes the following values - count, mean, std, min, max and quartiles of 25%, 50% and 75%. To go ahead with this representation we first create an empty data-frame with variables as column and index as describe values. We then take a variable *temp* which stores series of all values using the *apply* method which in-turn calls a function to perform all the calculations on the columns. The values of *temp* are then added in the data-frame through a loop.



	Current Assets - Other - Total	Current Assets - Total	Other Long-term Assets	Assets Netting & Other Adjustments
count	844.000000	844.000000	844.000000	844.000000
mean	1037.255108	9735.614198	1486.818614	-166.147714
std	1578.836159	13568.222671	2441.795091	560.878237
min	2.671000	144.786000	13.072000	-9558.000000
25%	181.500000	1499.018250	187.456000	-166.147714
50%	448.677000	4744.000000	827.000000	0.000000
75%	1037.255108	11617.250000	1492.000000	0.000000
max	9476.000000	76160.000000	40233.000000	138.000000

2.6. Calculate the correlation matrix for variables = ['Current Assets - Other - Total', 'Current Assets - Total', 'Other Long-term Assets', 'Assets Netting & Other Adjustments'].

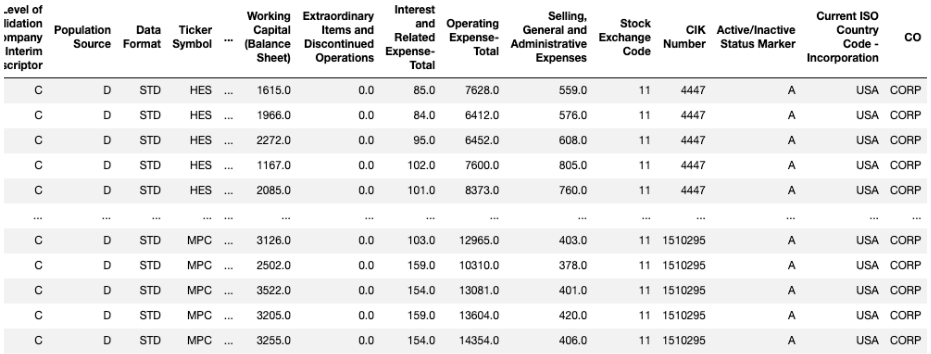
This is easily calculated using the *corr()* function of pandas



	Current Assets - Other - Total	Current Assets - Total	Other Long-term Assets	Assets Netting & Other Adjustments
Current Assets - Other - Total	1.000000	0.790047	0.629802	0.042504
Current Assets - Total	0.790047	1.000000	0.665006	-0.072010
Other Long-term Assets	0.629802	0.665006	1.000000	-0.017979
Assets Netting & Other Adjustments	0.042504	-0.072010	-0.017979	1.000000

2.7. If you look at column ('Company Name'), you will find some company name end with 'CORP', 'CO' or 'INC'. Create a new column (Name: 'CO') to store the last word of company name. (For example: 'CORP' or, 'CO' or 'INC') (Hint: using map function)

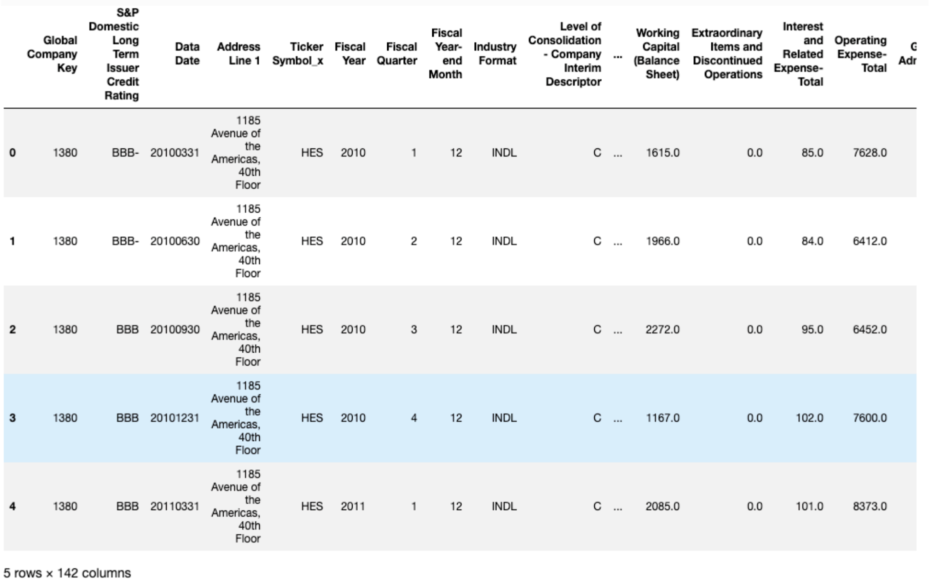
Here we use the split function to split the last word of the value and add it to a new column called 'CO' using a lambda function



Level of lidation ompany Interim scriptor	Population Source	Data Format	Ticker Symbol	...	Working Capital (Balance Sheet)	Extraordinary Items and Discontinued Operations	Interest and Related Expense-Total	Operating Expense-Total	Selling, General and Administrative Expenses	Stock Exchange Code	CIK Number	Active/Inactive Status Marker	Current ISO Country Code - Incorporation	CO
C	D	STD	HES	...	1615.0	0.0	85.0	7628.0	559.0	11	4447	A	USA	CORP
C	D	STD	HES	...	1966.0	0.0	84.0	6412.0	576.0	11	4447	A	USA	CORP
C	D	STD	HES	...	2272.0	0.0	95.0	6452.0	608.0	11	4447	A	USA	CORP
C	D	STD	HES	...	1167.0	0.0	102.0	7600.0	805.0	11	4447	A	USA	CORP
C	D	STD	HES	...	2085.0	0.0	101.0	8373.0	760.0	11	4447	A	USA	CORP
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
C	D	STD	MPC	...	3126.0	0.0	103.0	12965.0	403.0	11	1510295	A	USA	CORP
C	D	STD	MPC	...	2502.0	0.0	159.0	10310.0	378.0	11	1510295	A	USA	CORP
C	D	STD	MPC	...	3522.0	0.0	154.0	13081.0	401.0	11	1510295	A	USA	CORP
C	D	STD	MPC	...	3205.0	0.0	159.0	13604.0	420.0	11	1510295	A	USA	CORP
C	D	STD	MPC	...	3255.0	0.0	154.0	14354.0	406.0	11	1510295	A	USA	CORP

2.8. Merge (inner) Ratings and BalanceSheet based on 'data date' and 'Global Company Key', and name merged dataset 'Matched'.

We use the pandas function *merge* and set the key as *how="inner"* to merge the two dataframes with the respective columns

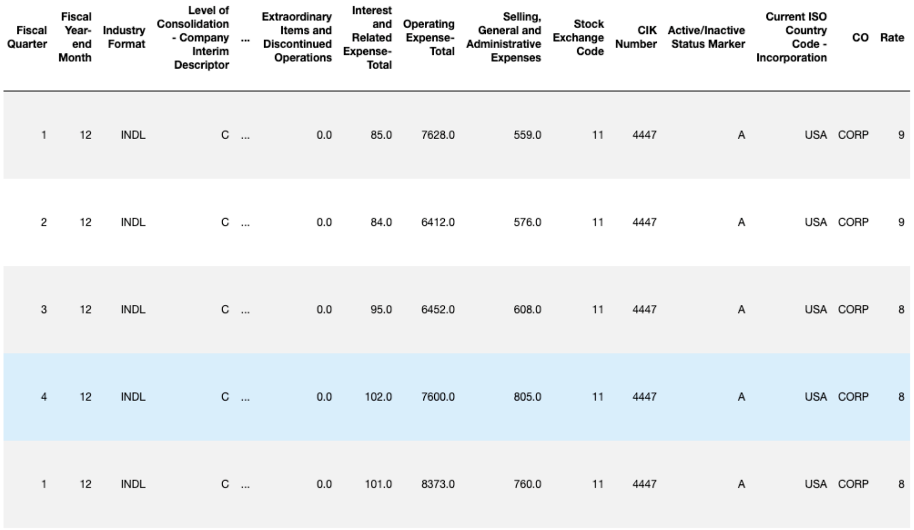


	Global Company Key	S&P Domestic Long Term Issuer Credit Rating	Data Date	Address Line 1	Ticker Symbol_x	Fiscal Year	Fiscal Quarter	Fiscal Year-end Month	Industry Format	Level of Consolidation - Company Interim Descriptor	...	Working Capital (Balance Sheet)	Extraordinary Items and Discontinued Operations	Interest and Related Expense-Total	Operating Expense-Total	G Adr
0	1380	BBB-	20100331	1185 Avenue of the Americas, 40th Floor	HES	2010	1	12	INDL	C	...	1615.0	0.0	85.0	7628.0	
1	1380	BBB-	20100630	1185 Avenue of the Americas, 40th Floor	HES	2010	2	12	INDL	C	...	1966.0	0.0	84.0	6412.0	
2	1380	BBB	20100930	1185 Avenue of the Americas, 40th Floor	HES	2010	3	12	INDL	C	...	2272.0	0.0	95.0	6452.0	
3	1380	BBB	20101231	1185 Avenue of the Americas, 40th Floor	HES	2010	4	12	INDL	C	...	1167.0	0.0	102.0	7600.0	
4	1380	BBB	20110331	1185 Avenue of the Americas, 40th Floor	HES	2011	1	12	INDL	C	...	2085.0	0.0	101.0	8373.0	

5 rows × 142 columns

2.9. Mapping - For dataset 'Matched', we have following mapping: AAA = 0 AA+ = 1 AA = 2 AA- =3 A+ = 4 A=5 A- = 6 BBB+ = 7 2 BBB = 8 BBB- = 9 BB+ = 10 BB = 11 others = 12 Using map function to create a new variable = 'Rate', which maps ratings to numerical ratings.

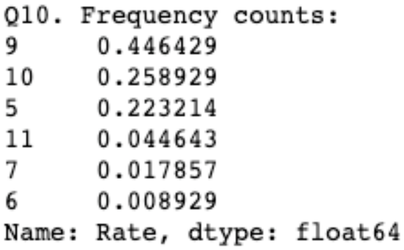
We start off by creating a dictionary with the above variables where each word (key) points to a number (value). We then use the lambda function to map the data-frame column with the dictionary into a new column called *'Rate'*



Fiscal Quarter	Fiscal Year-end Month	Industry Format	Level of Consolidation - Company Interim Descriptor	...	Extraordinary Items and Discontinued Operations	Interest and Related Expense-Total	Operating Expense-Total	Selling, General and Administrative Expenses	Stock Exchange Code	CIK Number	Active/Inactive Status Marker	Current ISO Country Code - Incorporation	CO	Rate
1	12	INDL	C	...	0.0	85.0	7628.0	559.0	11	4447	A	USA	CORP	9
2	12	INDL	C	...	0.0	84.0	6412.0	576.0	11	4447	A	USA	CORP	9
3	12	INDL	C	...	0.0	95.0	6452.0	608.0	11	4447	A	USA	CORP	8
4	12	INDL	C	...	0.0	102.0	7600.0	805.0	11	4447	A	USA	CORP	8
1	12	INDL	C	...	0.0	101.0	8373.0	760.0	11	4447	A	USA	CORP	8

2.10 Calculate the rating frequency of company whose name end with 'CO'. (Calculate the distribution of rating given the company name ending with 'CO', Hint, use map function)

We take two columns in consideration here - 'CO' and 'Rating' - we loop through the data-frame column 'CO' to find all values which have 'CO' and get the corresponding rating for that value from the 'Rating' Column and then calculate its frequency.



```
Q10. Frequency counts:
9     0.446429
10    0.258929
5     0.223214
11    0.044643
7     0.017857
6     0.008929
Name: Rate, dtype: float64
```

CODE APPENDIX

Q1.py

```
#!/usr/bin/env python
# coding: utf-8

# ## Q1. Credit Transaction data(40 points)

# This dataset is simulated individual credit card transactions by one company. Please use this dataset to answer following question. Please notice that you may need to observe the dataset and clean it before answering the following question.

# --> Import statements
import pandas as pd
import re

df = pd.read_csv("res_purchase_2014.csv", low_memory=False) #importing dataset res_purchase_2014.csv
print(df.head()) #displaying top 5 rows


# ### 1. What is total amount spending captured in this dataset?
# Hint: you may observe $ in front of the amount, which you need remove it (see.row 12), and () stands for negative value, which you need deduct the amount.

df = df.dropna() #dropping NA values
new = []
for value in df['Amount']:
    if(type(value) == str):
        value = value.strip()

        if(value[0] == '(' and value[-1] == ')'):
            value = str('-' + value[1:-1]) #if value contains ()

        if('$' in value):
            value = value.replace('$', '') #if value contains $

        if(re.search('[a-zA-Z]', value)):
            value = re.sub('[a-zA-Z]', '', value) #if value contains words

        value = float(value.strip()) #strip whitespace and convert str to float

    new.append(value)
    df['Amount'] =  new

    total = df['Amount'].sum() #sum all values
    print("Total Amount spent:",round(total,2)) #round to 2 digit


    # ### 2. How much was spend at WW GRAINGER?
    # Hint: All ‘WW GRAINGER’ contained in the ‘Vendor’.

    count = 0
    for i in range(len(df)):
        if(df['Vendor'][i] == 'WW GRAINGER'): #check for vendor with value WW GRAINGER
            count = count + df['Amount'][i] #if found then add to count

    print("Total Amount spent at WW GRAINGER:",round(count,2)) #round to 2 digit


    # ### 3. How much was spend at WM SUPERCENTER?
    # Hint: All ‘WM SUPERCENTER’ contained in the ‘Vendor’.

    count_sc = 0
    for i in range(len(df)):
        if(df['Vendor'][i] == 'WM SUPERCENTER'): #check for vendor with value WM
SUPERCENTER
            count_sc = count_sc + df['Amount'][i] #if found then add to count

    print("Total Amount spent at WM SUPERCENTER:",round(count_sc,2))  #round to 2
digit


    # ### 4. How much was spend at GROCERY STORES?
    # Hint: All ‘GROCERY STORES’ contained in the ‘Merchant Category Code’.

    count_gc = 0
    for i in range(len(df)):
        if('GROCERY STORES' in df['Merchant Category Code (MCC)'][i] ): #check for MCC
with value GROCERY STORES
            count_gc = count_gc + df['Amount'][i]

    print("Total Amount spent at GROCERY STORES:",round(count_gc,2)) #round to 2 digit
    # ### Final Output

    print('Results for Part 1: \n _________________\n')
    print("Total Amount spent:",round(total,2)) #round to 2 digit
    print("Total Amount spent at WW GRAINGER:",round(count,2)) #round to 2 digit
    print("Total Amount spent at WM SUPERCENTER:",round(count_sc,2))  #round to 2
digit
    print("Total Amount spent at GROCERY STORES:",round(count_gc,2)) #round to 2 digit
```

(Continue)

Q2.py

```
#!/usr/bin/env python
# coding: utf-8

# ## Q2 Data Processing with Pandas (60 points)
# In this practice, you are expected to play around Pandas and get familiar with it. The dataset is quarterly dataset downloading from WRDS. Please remember that you need to do data transformation based on the new dataset generated by previous step. Do not using other package other than numpy and pandas.

# --> Import statements
import pandas as pd
import numpy as np


# ### 1. Read’Energy.xlsx’and’EnergyRating.xlsx’as BalanceSheet and Ratings(dataframe)

BalanceSheet = pd.read_excel("Energy.xlsx") #importing dataset Energy.csv
print(BalanceSheet.head()) #displaying top 5 rows


Ratings = pd.read_excel("EnergyRating.xlsx") #importing dataset EnergyRating.csv
print(Ratings.head()) #displaying top 5 rows


# ### 2. drop the column if more than 90% value in this colnmn is 0 (or missing value).

def dropColumn(df):
    delList = []
    df = df.fillna(value=np.nan)
    for column in df.columns:
        if(((df[column] == 0).sum() + df[column].isnull().sum() + df[column].isna().sum())/len(df) > 0.9): # check if value in column is 0 or null or NaN then add
            delList.append(column)
    return delList

print('Size of Dataframes before dropping:')
print('BalanceSheet:',BalanceSheet.shape)
print('Ratings:',Ratings.shape)
print("-------------------")
```

```
    BalanceSheet = BalanceSheet.drop(columns=dropColumn(BalanceSheet)) #drop columns
with more than 90% 0 or null values
    Ratings = Ratings.drop(columns=dropColumn(Ratings))

    print('Size of Dataframes after dropping:')
    print('BalanceSheet:',BalanceSheet.shape)
    print('Ratings:',Ratings.shape)


    # ### 3. replace all None or NaN with average value of each column.

    BalanceSheet = BalanceSheet.fillna(value = BalanceSheet.mean())
    Ratings = Ratings.fillna(value = Ratings.mean())


    # ### 4. Normalize the table (Only need to normalize numerical parts)

    def normalize(df):
        num_cols = list(df.columns[df.dtypes.apply(lambda c: np.issubdtype(c,
np.number))])
        for col in num_cols:
            df[col].apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
        return df

    BalanceSheet = normalize(BalanceSheet)
    Ratings = normalize(Ratings)


    # ### 5. Define an apply function to return the statistical information for
variables = [’Current Assets - Other - Total’, ’Current Assets - Total’, ’Other Long-
term Assets’, ’Assets Netting & Other Adjustments’], you need to return a dataframe
which has exactly same format with pandas method .describe().
    #

    index = ["count", "mean", "std", "min", "25%", "50%", "75%", "max"]
    column_names = ["Current Assets - Other - Total", "Current Assets - Total",
            "Other Long-term Assets", "Assets Netting & Other Adjustments"]

    df = pd.DataFrame(columns = column_names, index = index)

    def res(column):
        data = [len(column), column.mean(), column.std(), column.min()] #calculate
data points
        q1, q2, q3 = column.quantile([.25, .5, .75]).to_list()
        data += [q1, q2, q3, column.max()]
        return data
    temp = BalanceSheet[column_names].apply(res, axis=0) #calling res function
    for i in df.columns:
        df[i] = temp[i] #adding back to dataframe
    print(df)


    # ### 6. Calculate the correlation matrix for variables = ['Current Assets - Other
- Total', 'Current Assets - Total', 'Other Long-term Assets', 'Assets Netting & Other
Adjustments'].

    correlation = BalanceSheet[column_names].corr()
    correlation


    # ### 7. If you look at column ('Company Name'), you will find some company name
end with 'CORP', 'CO' or 'INC'. Create a new column (Name: 'CO') to store the last
word of company name. (For example: 'CORP' or, 'CO' or 'INC') (Hint: using map
function)

    BalanceSheet["CO"] = BalanceSheet["Company Name"].map(lambda x: x.split()[-1])

    print(BalanceSheet)


    # ### 8. Merge (inner) Ratings and BalanceSheet based on 'datadate' and 'Global
Com- pany Key', and name merged dataset 'Matched'.

    Matched = pd.merge(Ratings, BalanceSheet, how="inner",on=["Data Date", "Global
Company Key"])
    Matched.head()


    # ### 9. Mapping
    # For dataset 'Matched', we have following mapping: AAA = 0
    # AA+ = 1
    # AA = 2
    #  AA- =3 A+ = 4 A=5
    # A- = 6 BBB+ = 7
    # 2
    # BBB = 8
    # BBB- = 9
    # BB+ = 10
    # BB = 11
    # others = 12
    # Using map function to create a new varible = ’Rate’, which maps ratings to
numerical ratings.
    ratings = {"AAA": 0, "AA+": 1, "AA": 2, "AA-": 3, "A+": 4, "A": 5, "A-": 6,"BBB+":
7, "BBB": 8, "BBB-": 9, "BB+": 10, "BB": 11}
    Matched["Rate"] = Matched['S&P Domestic Long Term Issuer Credit
Rating'].map(lambda x: ratings.get(x, 12))

    Matched.head()


    # ### 10. Calculate the rating frequency of company whose name end with ’CO’.
(Calcu- late the distribution of rating given the company name ending with ’CO’, Hint,
use map function)

    freq = Matched[Matched["CO"] == "CO"]["Rate"].value_counts() /
len(Matched[Matched["CO"] == "CO"])
    print("\nQ10. Frequency counts:")
    print(freq)
```