2.Create representation of document by calculating term frequency and inverse document frequency .

```
 #import the necessary libraries
import pandas as pd
import sklearn as sk
import math

first_sentence = "Data Science is the sexiest job of the 21st century"
second_sentence = "machine learning is the key for data science"
third_sentence = "machine learning is part of artificial intelligence"
fourth_sentence= "A data scientist has to think more than code "
fifth_sentence = "artificial intelligance is an emerging and promising
technology"
#split the sentences so that  each word have their own string
first_sentence = first_sentence.split(" ")
second_sentence = second_sentence.split(" ")
third_sentence = third_sentence.split(" ")
fourth_sentence = fourth_sentence.split(" ")
fifth_sentence = fifth_sentence.split(" ")
#join them to remove common duplicate words
total=
set(first_sentence).union(set(second_sentence).union(third_sentence).u
nion(fourth_sentence).union(fifth_sentence))
print(total)
```

```
{'', 'an', 'of', 'think', 'scientist', 'learning', 'data', 'and',
'intelligance', 'more', 'technology', 'artificial', 'has',
'intelligence', 'job', 'sexiest', 'part', 'the', 'to', 'A',
'promising', 'emerging', '21st', 'than', 'Data', 'for', 'century',
'machine', 'is', 'Science', 'code', 'key', 'science'}
```

```
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
wordDictC = dict.fromkeys(total, 0)
wordDictD = dict.fromkeys(total, 0)
wordDictE = dict.fromkeys(total, 0)
for word in first_sentence:
    wordDictA[word]+=1

for word in second_sentence:
    wordDictB[word]+=1

for word in third_sentence:
    wordDictC[word]+=1

for word in fourth_sentence:
    wordDictD[word]+=1
```

```
for word in fifth_sentence:
    wordDictE[word]+=1

#Now we put them in a dataframe and then view the result
pd.DataFrame([wordDictA, wordDictB, wordDictC, wordDictD, wordDictE])
```

```
      an  of  think  scientist  learning  data  and  intelligance
more  ...  \
0  0   0   1      0          0         0     0    0             0
0  ...
1  0   0   0      0          0         1     1    0             0
0  ...
2  0   0   1      0          0         1     0    0             0
0  ...
3  1   0   0      1          1         0     1    0             0
1  ...
4  0   1   0      0          0         0     0    1             1
0  ...

   than  Data  for  century  machine  is  Science  code  key  science

0     0     1    0        1        0   1        1     0    0        0

1     0     0    1        0        1   1        0     0    1        1

2     0     0    0        0        1   1        0     0    0        0

3     1     0    0        0        0   0        0     1    0        0

4     0     0    0        0        0   1        0     0    0        0


[5 rows x 33 columns]
```

## Term Frequency:

the number of times a term occurs in a document.The number of times a word appears in a document divded by the total number of words in the document it can be calculated by tf(t,d) = count of t in d / number of words in d here t is term and d is document .

Assume that a document has 20 words and 5 of them is the word “great”. The TF will be calculated as: tf: 5/20 = 0.25

```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)
    for word, count in wordDict.items():
        tfDict[word] = count/float(corpusCount)
    return(tfDict)
```

```
#running our sentences through the tf function:
tfFirst = computeTF(wordDictA, first_sentence)
tfSecond = computeTF(wordDictB, second_sentence)
tfThird = computeTF(wordDictC, third_sentence)
tfFourth = computeTF(wordDictD, fourth_sentence)
tfFifth = computeTF(wordDictE, fifth_sentence)

#Converting to dataframe for visualization
tf =pd.DataFrame([tfFirst, tfSecond, tfThird,tfFourth, tfFifth])
print(tf)
```

```
          an        of  think  scientist  learning   data    and  \
0  0.0  0.000  0.100000    0.0        0.0  0.000000  0.000  0.000
1  0.0  0.000  0.000000    0.0        0.0  0.125000  0.125  0.000
2  0.0  0.000  0.142857    0.0        0.0  0.142857  0.000  0.000
3  0.1  0.000  0.000000    0.1        0.1  0.000000  0.100  0.000
4  0.0  0.125  0.000000    0.0        0.0  0.000000  0.000  0.125

   intelligance  more  ...  than  Data    for  century   machine
is  \
0         0.000   0.0  ...   0.0   0.1  0.000      0.1  0.000000
0.100000
1         0.000   0.0  ...   0.0   0.0  0.125      0.0  0.125000
0.125000
2         0.000   0.0  ...   0.0   0.0  0.000      0.0  0.142857
0.142857
3         0.000   0.1  ...   0.1   0.0  0.000      0.0  0.000000
0.000000
4         0.125   0.0  ...   0.0   0.0  0.000      0.0  0.000000
0.125000

   Science  code    key  science
0      0.1   0.0  0.000    0.000
1      0.0   0.0  0.125    0.125
2      0.0   0.0  0.000    0.000
3      0.0   0.1  0.000    0.000
4      0.0   0.0  0.000    0.000

[5 rows x 33 columns]
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
filtered_sentence = [w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['', 'think', 'scientist', 'learning', 'data', 'intelligance',
'technology', 'artificial', 'intelligence', 'job', 'sexiest', 'part',
'A', 'promising', 'emerging', '21st', 'Data', 'century', 'machine',
'Science', 'code', 'key', 'science']
```

```
[nltk_data] Error loading stopwords: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```

```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)

    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items():
        idfDict[word] = math.log10(N / (float(val) + 1))

    return(idfDict)
#inputing our sentences in the log file
idfs = computeIDF([wordDictA, wordDictB, wordDictC, wordDictD,
wordDictE])
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
C:\Users\RAKSHA~1\AppData\Local\Temp/ipykernel_20580/1135201478.py in
<module>
      9     return(idfDict)
     10 #inputing our sentences in the log file
---> 11 idfs = computeIDF([wordDictA, wordDictB, wordDictC, wordDictD,
wordDictE])

NameError: name 'wordDictA' is not defined
```

```
def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items():
        tfidf[word] = val*idfs[word]
    return(tfidf)
#running our two sentences through the IDF:
idfFirst = computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
idfThird = computeTFIDF(tfThird, idfs)
idfFourth = computeTFIDF(tfFourth, idfs)
idfFifth = computeTFIDF(tfFifth, idfs)
#putting it in a dataframe
idf= pd.DataFrame([idfFirst, idfSecond, idfThird, idfFourth,
idfFifth])
print(idf)
```

```
                  and      Data   century       key     think
artificial  \
0  0.000000  0.000000  0.069897  0.069897  0.000000  0.000000
0.000000
1  0.000000  0.000000  0.000000  0.000000  0.087371  0.000000
0.000000
```

```
2  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
0.099853
3  0.069897  0.000000  0.000000  0.000000  0.000000  0.069897
0.000000
4  0.000000  0.087371  0.000000  0.000000  0.000000  0.000000
0.087371

   learning       job      code  ...      21st  scientist      than  \
0  0.000000  0.069897  0.000000  ...  0.069897   0.000000  0.000000
1  0.087371  0.000000  0.000000  ...  0.000000   0.000000  0.000000
2  0.099853  0.000000  0.000000  ...  0.000000   0.000000  0.000000
3  0.000000  0.000000  0.069897  ...  0.000000   0.069897  0.069897
4  0.000000  0.000000  0.000000  ...  0.000000   0.000000  0.000000

   intelligence   machine      data   science   sexiest  intelligance
\
0      0.000000  0.000000  0.000000  0.000000  0.069897      0.000000

1      0.000000  0.087371  0.087371  0.087371  0.000000      0.000000

2      0.099853  0.099853  0.000000  0.000000  0.000000      0.000000

3      0.000000  0.000000  0.069897  0.000000  0.000000      0.000000

4      0.000000  0.000000  0.000000  0.000000  0.000000      0.087371


        for
0  0.000000
1  0.087371
2  0.000000
3  0.000000
4  0.000000

[5 rows x 33 columns]
```