ASSIGNMENT 1

Working with Edgar datasets:
Wrangling, Pre-processing and exploratory data analysis

COURSE: INFO7390
Advance Data Science & Architecture

PROFESSOR:
Srikanth Krishnamurthy

SUBMITTED BY:
TEAM 9
Amit Pingale - 001898697
Himani Solanki - 001899580
Shubham Patel - 001899476

Problem:1

Step 1: Creating URL from CIK code and Accession Number given by the user.
The URL string is generated by modifying the inputs given and examining the format.

```
In [3]: cik = str(cik)
        accession = str(accession)
        cik = cik.lstrip('0')
        acc = re.sub(r'[-]', r'', accession)
        url = 'https://www.sec.gov/Archives/edgar/data/' + cik + '/' + acc + '/' + accession + '/-index.htm'

        print(url)

        https://www.sec.gov/Archives/edgar/data/51143/000005114313000007/0000051143-13-000007/-index.htm

In [ ]:
```

Step 2: Importing urllib.request in python 3.2 to open the above generated URL.

From bs4 import BeautifulSoup to fetch the entire HTML content from the requested file and parse the DOM elements.

```
        https://www.sec.gov/Archives/edgar/data/51143/000005114313000007/0000051143-13-000007/-index.htm

In [4]: final_url = ""
        html = urllib.request.urlopen(url)
        soup = BeautifulSoup(html, "html.parser")
        all_tables = soup.find('table', class_='tableFile')
        tr = all_tables.find_all('tr')

        for row in tr:
            final_url = row.findNext("a").attrs['href']
            break
        next_url = "https://www.sec.gov" + final_url
        print(next_url)

        https://www.sec.gov/Archives/edgar/data/51143/000005114313000007/ibm13q3_10q.htm
```

Step 3: Getting the URL for 10-Q form using the fetched page.

```
        https://www.sec.gov/Archives/edgar/data/51143/000005114313000007/0000051143-13-000007/-index.htm

In [4]: final_url = ""
        html = urllib.request.urlopen(url)
        soup = BeautifulSoup(html, "html.parser")
        all_tables = soup.find('table', class_='tableFile')
        tr = all_tables.find_all('tr')

        for row in tr:
            final_url = row.findNext("a").attrs['href']
            break
        next_url = "https://www.sec.gov" + final_url
        print(next_url)

        https://www.sec.gov/Archives/edgar/data/51143/000005114313000007/ibm13q3_10q.htm
```

Step 4: **Fetched all table elements by scrapping the DOM object. Multiple tables are sorted and saved**

```
https://www.sec.gov/Archives/edgar/data/51143/000005114313000007/ibm13q3_10q.htm

In [ ]: htmlpage = urllib.request.urlopen(next_url)
        page = BeautifulSoup(htmlpage, "html.parser")
```

Step 5: **FETCH THE TABLE and removing all null values and commas**

```
In [5]: refined_tables=[]

        for tab in all_divtables:
            for tr in tab.find_all('tr'):
                f=0
                for td in tr.findAll('td'):
                    if('$' in td.get_text()):
                        refined_tables.append(tab)
                        f=1;
                        break;
                if(f==1):
                    break;
```

Step 6: **CREATE CSV FOR EACH DATA TABLE**

```
In [6]: for tab in refined_tables:
            records = []
            for tr in tab.find_all('tr'):
                rowString=[]
                for td in tr.findAll('td'):
                    p = td.find_all('p')
                    if len(p)>0:
                        for ps in p:
                            ps_text = ps.get_text().replace("\n"," ")
                            ps_text = ps_text.replace("\xa0","")
                            rowString.append(ps_text)
                    else:
                        td_text=td.get_text().replace("\n"," ")
                        td_text = td_text.replace("\xa0","")
                        rowString.append(td_text)
                records.append(rowString)
            with open(os.path.join('csvFile' , str(refined_tables.index(tab)) + 'tables.csv'), 'w') as f:
                writer = csv.writer(f)
                writer.writerows(records)
```

Step 7: **Zipping the table**

```
In [11]: def zipdir(path, ziph, refined_tables):
             # ziph is zipfile handle
             for tab in refined_tables:
                 ziph.write(os.path.join('csvFile', str(refined_tables.index(tab))+'tables.csv'))

         zipf = zipfile.ZipFile('csv.zip', 'w', zipfile.ZIP_DEFLATED)
         zipdir('/', zipf, refined_tables)
         zipf.close()
```

Problem:2

The document summarizes the design and implementation of the analysis performed on the Edger log files. This document is divided into four parts.

Part 1: Fetching and Analysis of Edgar log file.
Part 2: Handling Missing Values and compute summary metric for Edgar Log files
Part 3: Create the Tableau representation of the analysis performed on the log

Part 1: Fetching and Analysis of Edgar log file.

The EDGAR Log File Data Set contains information in the CSV format extracted from log files that record and store user access statistics for the sec.gov website. The logs are captured on daily basis and are stored in a zip format under respective year on the website.

Edgar log files consist of the following columns:

Each column in the log file store various information compromising of:

CIK: Edgar company CIK for filing purpose

Accession: Accession number to access the specified file

Extension: Consist of the file that is requested by supplying CIK and accession number

Code: Implies the response code from the server

Date: Log file creation file

Analyzing the log based on the description present on the variables we found that:

- There were missing values for column name browser and size
- There were many extensions without the file name
- Cik length was more than 10 whereas on site it is mentioned as 10Q
- For code 304 we have all the file size as 0

Part 2: Handling Missing Values and compute summary metric for Edgar Log files

This part of the documentation consists of getting a year from the user and then extracting all the log files of that particular year's every month's first day log file and unzipping it as they are present in the .zip format.

Step 1: Get the year from the user

The program takes the year as the parameter from the user. As there are only log file presented for the year 2003 to 2017, if the user gives a parameter which is out from that range, the program will show an error asking to enter a valid year.

If the year is valid, then we will fetch the url from the Edgar website which consist of all the log files associated for a particular year.

```
args = sys.argv[1:]

year = ''
counter = 0
if len(args) == 0:
    year = "2003"
for arg in args:
    if counter == 0:
        year= str(arg)
    counter += 1
```

Step 2: Download the Extracted Log file for all the 1st date of the Month

As we are using python 3.2 for accessing the URL, we need to import the following python library to access the Edgar website.

import urllib.request :- To open a requested URL

Next we will be hitting the actual URL which consist of the log files to be analyzed, we will be using BeautifulSoup to work with the HTML files.

from bs4 import BeautifulSoup : - To fetch the HTML content from the requested file

```
In [11]: monthlistdata = []
         count = 0
         for li in ziplist:
             zipatags = li.findAll('a')
             for zipa in zipatags:
                 if "01.zip" in zipa.text:
                     monthlistdata.append(zipa.get('href'))

In [12]: monthlistdata

Out[12]: ['http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr4/log20031201.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr4/log20031101.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr4/log20031001.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr3/log20030901.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr3/log20030801.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr3/log20030701.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030601.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030501.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr2/log20030401.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030301.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030201.zip',
          'http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/2003/Qtr1/log20030101.zip']
```

<u>Step 3:</u> Load the Log File in the DataFrame using Pandas

To load all the log file data and to do the analysis purpose into the program we use Python Library Pandas for creating the data frame.

```
In [17]: df
```

Out[17]:

	ip	date	time	zone	cik	accession	extention	code	size	idx	norefer	noagent
0	129.110.39.jca	2003-03-01	00:00:00	500.0	97349.0	0000097349-01-000006	-0002.txt	200.0	3726.0	0.0	0.0	0.0
1	61.115.76.jbf	2003-03-01	00:00:00	500.0	766351.0	0000950134-03-003149	.txt	200.0	995957.0	0.0	1.0	0.0
2	61.115.76.jbf	2003-03-01	00:00:01	500.0	902584.0	0000902584-03-000044	.txt	200.0	15520.0	0.0	1.0	0.0
3	61.115.76.jbf	2003-03-01	00:00:03	500.0	778207.0	9999999997-03-006003	.txt	200.0	1670.0	0.0	1.0	0.0
4	129.110.39.jca	2003-03-01	00:00:07	500.0	97349.0	0000097349-01-000006	-index.htm	200.0	4331.0	1.0	0.0	0.0
5	208.62.55.eib	2003-03-01	00:00:07	500.0	56824.0	0000950124-03-000077	-index.htm	200.0	2727.0	1.0	0.0	0.0
6	129.110.39.jca	2003-03-01	00:00:10	500.0	97349.0	0000097349-01-000006	-0003.txt	200.0	1211.0	0.0	0.0	0.0
7	208.62.55.eib	2003-03-01	00:00:12	500.0	56824.0	0000950124-03-000077	k73883e10vqza.txt	200.0	158260.0	0.0	0.0	0.0
8	129.110.39.jca	2003-03-01	00:00:13	500.0	97349.0	0000097349-01-000006	-index.htm	200.0	4331.0	1.0	0.0	0.0
9	148.139.130.hhi	2003-03-01	00:00:16	500.0	1108205.0	0000927016-02-005853	-index.htm	304.0	NaN	1.0	0.0	0.0
10	129.110.39.jca	2003-03-01	00:00:17	500.0	97349.0	0000097349-01-000006	-0003.txt	200.0	1211.0	0.0	0.0	0.0
11	129.110.39.jca	2003-03-01	00:00:19	500.0	97349.0	0000097349-01-000006	-index.htm	200.0	4331.0	1.0	0.0	0.0
12	61.115.76.jbf	2003-03-01	00:00:21	500.0	778207.0	0000950144-03-002432	.txt	200.0	1974869.0	0.0	1.0	0.0
13	67.81.137.eci	2003-03-01	00:00:21	500.0	857323.0	0000908662-03-000043	form_8-k.txt	200.0	2510.0	0.0	0.0	0.0
14	203.200.34.ejd	2003-03-01	00:00:31	500.0	1125051.0	0001002014-03-000092	-index.htm	200.0	2832.0	1.0	0.0	0.0
15	129.110.39.jca	2003-03-01	00:00:32	500.0	97349.0	0000097349-01-000006	-0004.txt	200.0	316657.0	0.0	0.0	0.0
16	148.139.130.hhi	2003-03-01	00:00:33	500.0	1108205.0	0000927016-02-005853	d8k.txt	304.0	NaN	0.0	0.0	0.0

<u>Step 4:</u> Change the data type of the column present in the Dataframe

After getting all the log file for a year, we will use the Python Library Pandas for creating a dataframe and load all the log file data into it for the analysis purpose.

```
In [18]: all_data = df
```

```
In [19]: new_data = pd.DataFrame()
         all_data['zone'] = all_data['zone'].astype('int64')
         all_data['cik'] = all_data['cik'].astype('int64')
```

```
In [20]:
         all_data['noagent'] = all_data['noagent'].astype('int64')
         all_data['norefer'] = all_data['norefer'].astype('int64')
```

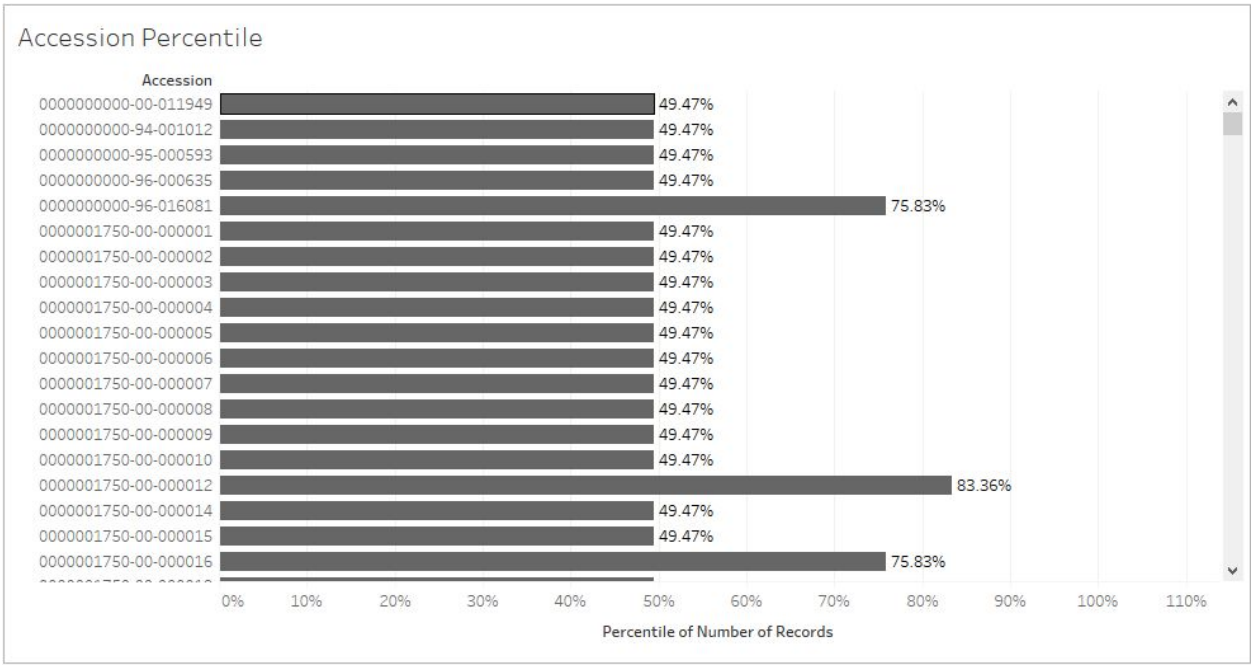
```
In [21]: all_data['code'] = all_data['code'].astype('int64')
```

```
In [22]:  all_data['idx'] = all_data['idx'].astype('int64')
```

```
In [23]: all_data['crawler'] = all_data['crawler'].astype('int64')
         all_data['find'] = all_data['find'].astype('int64')
```

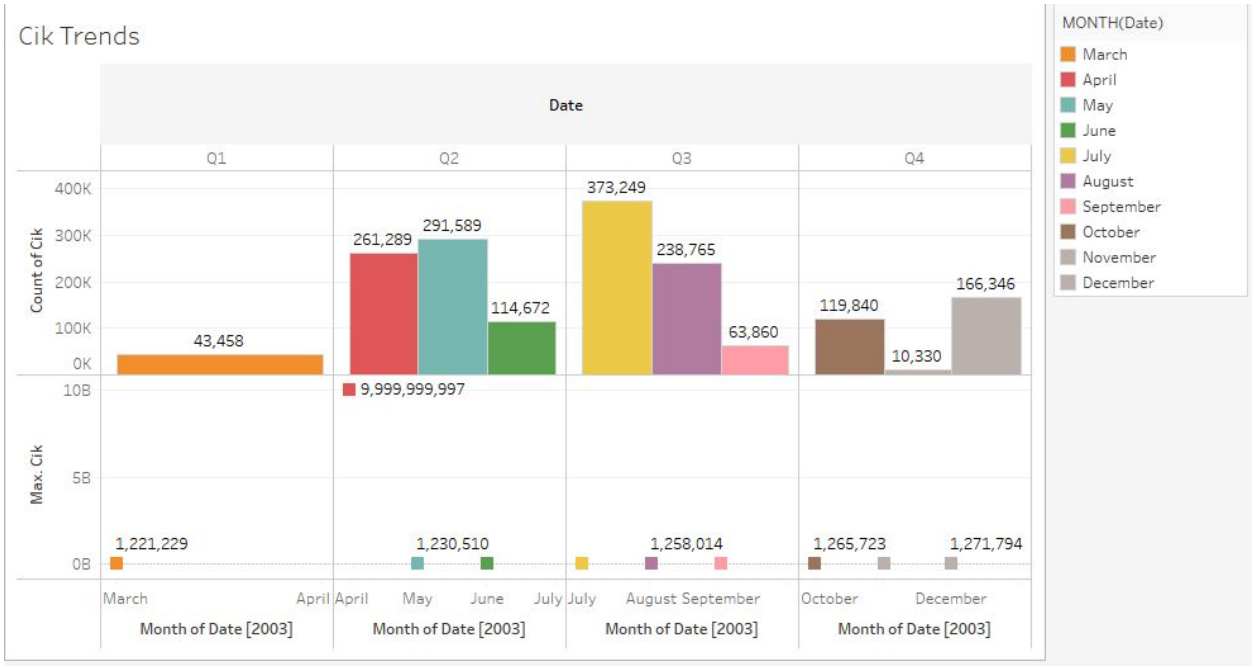
```
In [24]: data = pd.DataFrame()
```

Part 4: Create the Tableau representation of the analysis performed on the log

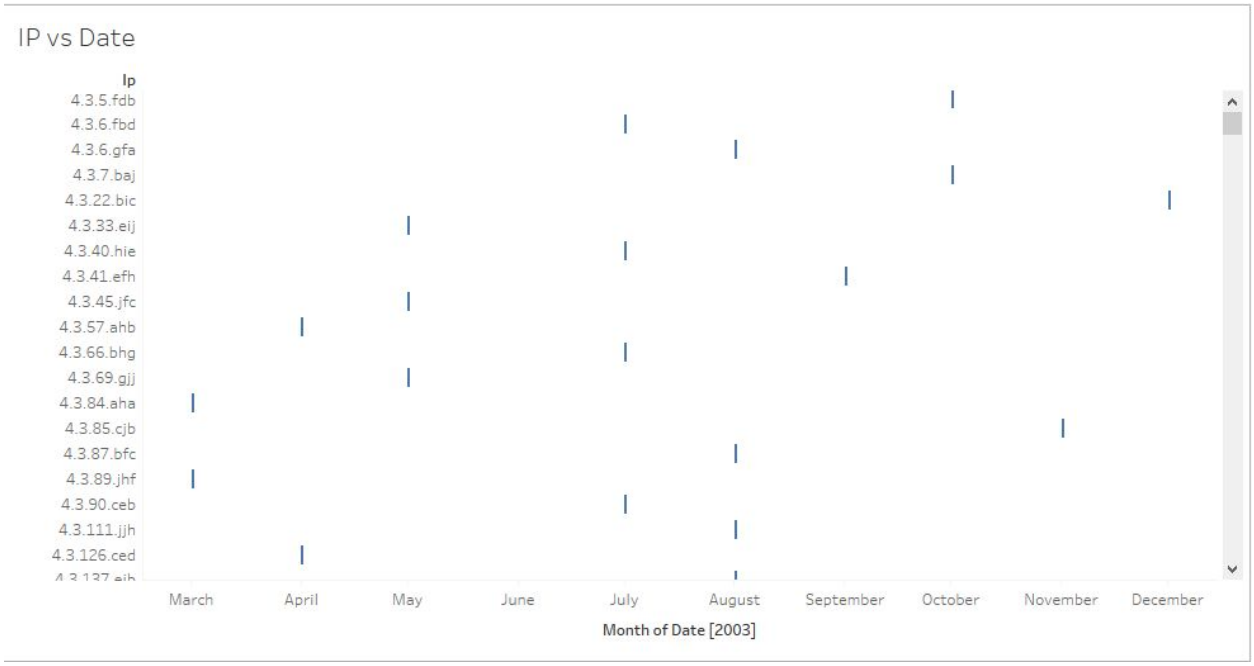


This tableau image shows the percentile of the number of records generated by the particular Accession Number.

This tableau image shows the size of the code each and every browser is accessing.



This tableau image shows the maximum length of Cik and also the count of Cik with respect to Date Quarters and Data month.



This tableau image shows the IP address vs month of the selected year.