Assignment 1 Report

Overview

Process implemented in brief

1. Language used : Python
2. Process Followed : Data Ingestion, Data Wrangling, Data Cleansing, Exploratory Data Analysis
3. Tools used : Jupyter Notebook, boto 3, boto, Amazon S3 bucket

Steps to Run Docker for Part 1 and Part 2

Problem 1 Part 2 Docker Steps

1. docker build -f dock -t part2ofpart1 .
2. docker run -e ck=0000051143 -e ac=0000051143-13-000007 -e akey=your key -e skey=secret key -e sl=us-west-2 -ti part2ofpart1
3. docker tag b7dc3f7451d3 tushargl016/part2ofpart1:latest
4. docker pull tushargl016/part2ofpart1
5. docker run -e ck=0000051143 -e ac=0000051143-13-000007 -e akey= -e skey= -e sl=us-west-2 -ti tushargl016/part2ofpart1

Problem 2 Docker Steps

1. docker build -f dock -t assignmentonepart2 .
2. docker run -e year=2005 -e akey=your key -e skey=secret key -e sl=eu-west-1 -ti assignmentonepart2
3. docker tag b7dc3f7451d3 tushargl016/part2assignmentone:latest
4. docker pull tushargl016/part2assignmentone
5. docker run -e year=2005 -e akey= -e skey= -e sl=eu-west-1 -ti tushargl016/assignmentonepart2

akey – aws key, skey=secret key, sl=s3bucketregion,year=yearoflog

acn-accessionnumber, cik-cik number ,s3loc-aws s3 location

Submission

Github link : https://github.com/tushargl016/Team5assignment1

Problem 1 – Part1: is located in the folder name Part1ofPart1

Problem 1-Part2: is located in in the folder name Part2ofPart1

Problem 2: is located in Part 2

Problem 1 Part 1

Abstract:

Our approach to address Part 1 involved scraping the Edgar Data Website when Accession Number and CIK as input arguments. The form is parsed for the 10Q filing using Python Libraries and data is input into a data frame. We then implemented data cleansing which included extracting only tables with the data. Followed by aligning the columns, removing '$' signs which occupied individual columns, removing NaN, None and any occurrences of blank columns and after removing the Nan and None values realigned the columns by looping through the data frame.

Process Overview

Step 1

1. **Web Scraping:**
   a. Programmatically generated the URL (url_pre + cik_striped + "/" + accno_striped + "/" + accessionNumber + "-index.html").
   b. The CIK and Accession Number was accessed using arguments. The library used was argparse
   c. We used the python library Beautiful Soup to parse through the HTML form. If the 10Q file is not present it will provide a note showing that the 10Q file for the specified CIK does not exist.

   d. We initially tried loading data in the data frame using the Pandas read_table () command but it failed to extrapolate all tables. As a workaround we then put data from the 10Q form was put in the data frame using Pandas (pandas.read_html ()). We have also programmatically generated a note if an incorrect CIK and Accession Number is provided.

```
url_pre = "https://www.sec.gov/Archives/edgar/data/"
cik_striped = cik.lstrip("0")
accno_striped = accessionNumber.replace("-","")
url1 = url_pre + cik_striped + "/" + accno_striped + "/" + accessionNumber + "-index.html"
url2=''
```

Selecting all href links

```
url2=''
try:
    page = urllib.request.urlopen(url1)
    soup = BeautifulSoup(page,"lxml") # parse the page and save it in soup format
    form = soup.find(id='formName').get_text() # find the form name according to accession no
    formname = form[6:10]
    formtype = soup.findAll('td', text = formname)[0]
    all_links = soup.find_all('a')
    for link in all_links:
        href=link.get("href")
        print (href)
```

List of href links generated

```
/index.htm
/cgi-bin/browse-edgar?action=getcurrent
javascript:history.back()
/index.htm
/edgar/searchedgar/webusers.htm
/index.htm
/edgar/searchedgar/webusers.htm
/edgar/searchedgar/companysearch.html
/cgi-bin/viewer?action=view&cik=51143&accession_number=0000051143-13-000007&xbrl_type=v
/Archives/edgar/data/51143/000005114313000007/ibm13q3_10q.htm
extracted_csvs\1.csv
```

Explicitly selecting the 10Q links and Exception Handling

```
        if "10q.htm" in href:
            url2 = "https://www.sec.gov/" + href
            break;
        else:
            url2=""


    if url2 is "":
        print("Invalid URL!!! or 10q not found for the given cik or accession number")
        exit()


except urllib.error.HTTPError as err:
    print("Invalid CIK or AccNo")
    exit()
```

10 Q URL generated

```
    except:
        pass
print(count)
```

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

Reading from 10 Q file

```
3 NaN
4 NaN  ,                                                    0   1     2    3       4  \
0                              (Dollars in  millions) NaN   NaN  NaN     NaN
1           For the three months ended September 30: NaN  2013  NaN  2012.0
2   Net  unrealized gains/(losses) arising during ... NaN     $    3     NaN
3   Net  unrealized (gains)/losses reclassified to... NaN   NaN  (3)     NaN
4                                                 NaN NaN   NaN  NaN     NaN
5   *There were no writedowns for the three months... NaN   NaN  NaN     NaN
6                                                 NaN NaN   NaN  NaN     NaN
7                               (Dollars in millions) NaN   NaN  NaN     NaN
8            For the nine months ended September 30: NaN  2013  NaN  2012.0
9   Net  unrealized gains/(losses) arising during ... NaN     $    2     NaN
10  Net  unrealized (gains)/losses reclassified to... NaN   NaN  (3)     NaN
11                                                NaN NaN   NaN  NaN     NaN
12  * There were no significant writedowns for the... NaN   NaN  NaN     NaN
13                                                NaN NaN   NaN  NaN     NaN

      5     6   7
0   NaN   NaN NaN
1   NaN   NaN NaN
```

Step 2

Data Cleansing

The data when inserted into the data frame was out of format, columns were not organized and consisted of NaN, None and empty strings.

1. In order to iterate through the data frame we formulated a ‘for’ loop that would iterate through the data frame
2. In order to align the ‘$’ signs, we first retrieved all records that had “$’ or “%” by iterating through the shape of the data frame using a ‘for’ loop and stored it in variable (x)
3. Next, we analyzed, that all the columns were greater than 3 they were the tables we wanted to extract. Hence, using that condition we formulated a ‘for’ loop that would loop through the columns found in variable (x). If this loop found instances of a ‘$’ sign in a cell and a float number in the preceding cell it would append the ‘$’ sign found to the float number found.
4. Next we replaced all occurrences of opening and closing parenthesis, empty cell values, and any other ‘%’ in combination of ‘)’ using regex expressions and replaced these with NaN.
5. Next we dropped all ‘NaN’ and None values and reindexed the table
6. Though all records with None and NaN values were dropped , there yet existed None or NaN values in between in records

7. In order to eliminate this we formulated another ‘for’ loop. Using the current shape of the data frame we iterated through the data frame and if a float number or value was found after the NaN or None value it would move that particular column field to replace the null or none value. In the case it would find more that one none or null value preceding, the loop would try and find the next available float number and move it.

8. The above helped us achieve clarity in the data and helped us get rid of NaN and None values in between fields.
9. Following this we dropped any None or NaN that existed and reindexed the columns
10. The next challenge we faced was the tables that had subheading columns were occurring one index before the its actual position (Example: year 2013, 2014 and 2015).

11. In order to overcome this we formulated a for loop that would take the first column subheading and push it to the very next column, subsequently pushing each column to the very next index.
12. Next, if there were any other NaN present, were replaced with empty strings

13. Finally, the cleaned data was appended into a csv file that was created in a directory using os.path() and os.mkdir()

14. A new csv file would be created for each table as it iterated through the for loop (for loop 1)

Code to extrapolate all table fields with occurrences of ‘$” or ‘%’ (Used Pandas library and stored data in a Pandas Data Frame)

```
count=0
for i in pandu(200):
    try:
        x=None
        flag=0
        t=i.shape
        for j in range(0,t[0]):
            if flag==1:
                break
            for m in range(0,t[1]):
                if i[m][j]=="$" or i[m][j]=="%":
                    flag=1
                    x=i
                    break
```

Code to append '$' sign to numbers preceding the '$' cell (Used the Numpy Library)

```
for s in range(0,sh[0]):
    for k in range(0,sh[1]):
        if example6[k][s] =="$":
            for p in range(k+1,sh[1]):
                if isinstance(example6[p][s],float) or isinstance(example6[p][s],np.float64) or isinstance
                (example6[p][s],np.str):
                    example6[k][s]=str("$"+str(example6[p][s]))
                    example6[p][s]=None
                    break
```

Snippet before appending '$' to the number

	6	7	8	9	10	11	\
0	NaN	NaN	NaN	NaN	NaN	NaN	
1	Nine Months Ended September 30,	NaN	NaN	NaN	NaN	NaN	
2	2012	NaN	NaN	2013.0	NaN	NaN	
3	3824	NaN	$	10299.0	NaN	$	
4	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	501	NaN	NaN	(959)	NaN	
6	NaN	NaN	NaN	NaN	NaN	NaN	
7	NaN	NaN	11	NaN	NaN	0	
8	NaN	NaN	(27)	NaN	NaN	(5)	
9	NaN	NaN	NaN	NaN	NaN	NaN	
10	NaN	NaN	(7)	NaN	NaN	3	

	3	4	5	6	7	8	9	10	\
0	Liability	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	as of	NaN	NaN	NaN	NaN	Other	NaN	as of	
2	NaN	Payments	NaN	Adjustments*	NaN	9/30/2013	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	$	28	NaN	$	(22)	NaN	$	25	
5	NaN	2	NaN	NaN	(1)	NaN	NaN	(1)	
6	30	NaN	$	(23)	NaN	$	24	NaN	
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8	$	430	NaN	$	–	NaN	$	1	
9	NaN	0	NaN	NaN	–	NaN	NaN	0	
10	430	NaN	$	–	NaN	$	1	NaN	
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

After executing the ‘for’ loop to append ‘$’ sign to the number

	3	4	5	6	7	8	9	10	\
0	Liability	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	as of	NaN	NaN	NaN	NaN	Other	NaN	as of	
2	NaN	Payments	NaN	Adjustments*	NaN	9/30/2013	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	$28	None	NaN	$(22)	None	NaN	$25	None	
5	NaN	2	NaN	NaN	(1)	NaN	NaN	(1)	
6	None	NaN	$(23)	None	NaN	$24	None	NaN	
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8	$430	None	NaN	$–	None	NaN	$1	None	
9	NaN	0	NaN	NaN	–	NaN	NaN	0	
10	None	NaN	$–	None	NaN	$1	None	NaN	
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Eliminated all parenthesis (Used Regex expressions)

	5	6	7	8	9	10	\
0	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	Purchase	NaN	NaN	NaN	NaN	
3	Price	NaN	NaN	NaN	And Other	NaN	
4	Adjustments	Divestitures	NaN	Adjustments	NaN	12/31/12	
5	None	$0	None	NaN	$2	None	
6	264	NaN	–	NaN	NaN	0	
7	2182	NaN	30	NaN	NaN	6	
8	443	NaN	0	NaN	NaN	14	
9	None	$30	None	NaN	$22	None	

Rearranging columns values

We re-arranged the values to their respective columns using the following logic (Used Pandas Library to check if a field was null (pd.isnull())

```
for s in range(0,sh1[0]):
    for k in range(0,sh1[1]):
        if pd.isnull(example12[k][s]):
            for j in range(k+1,sh1[1]):
                if pd.isnull(example12[j][s]):
                    continue
                else:
                    example12[k][s]=example12[j][s]
                    example12[j][s]=None
                    break
```

Output after running the above for loop

3	Dollars in millions		Balance	Goodwill		Price				
4	Segment		01/01/12	Additions		Adjustments				
5	Global Business Services		$4313.0	$5		$0				
6	Global Technology Services		2646	264		—				
7	Software		18121	2182		30				
8	Systems and Technology		1133	443		0				
9	Total		$26213.0	$2894		$30				

	4	5	6	7	8	9	10	11	12	\
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	None	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	None	
2	NaN	NaN	None	NaN	NaN	NaN	NaN	None	NaN	
3	And Other	Balance	NaN	NaN	NaN	None	NaN	None	NaN	
4	Divestitures	Adjustments	12/31/12	NaN	None	NaN	None	NaN	NaN	
5	$2	$42.0	$4357.0	None	NaN	None	None	NaN	None	
6	0	6	2916	None	NaN	NaN	None	NaN	NaN	
7	6	137	20405	None	NaN	NaN	None	NaN	NaN	
8	14	6	1568	None	NaN	NaN	None	NaN	NaN	
9	$22	$192.0	$29247.0	None	NaN	None	None	NaN	None	

All rows and columns that had only NaN or None values were eliminated (axis =0 and axis=1)

0	Foreign	NaN	NaN	NaN	
1	Currency	NaN	NaN	NaN	
2	Purchase	Translation	NaN	NaN	
3	Dollars in millions	Balance	Goodwill	Price	
4	Segment	01/01/12	Additions	Adjustments	
5	Global Business Services	$4313.0	$5	$0	
6	Global Technology Services	2646	264	—	
7	Software	18121	2182	30	
8	Systems and Technology	1133	443	0	
9	Total	$26213.0	$2894	$30	

	4	5	6
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	NaN	NaN	None
3	And Other	Balance	NaN
4	Divestitures	Adjustments	12/31/12
5	$2	$42.0	$4357.0
6	0	6	2916
7	6	137	20405

We used the following code snippet to rearrange the columns such that the column values are aligned by the column names. (Pandas Library, Pandas Data Frame)

```
for s in range(0,sh2[0]):
    for k in range(0,sh2[1]):
        if pd.isnull(example15[k][s]):
            counter2+=1
        if counter2==2:
            break
    if counter2==1:
        for l in range((sh2[1]-1),-1,-1):
            if l >0:
                example15[l][s]=example15[l-1][s]
            else:
                example15[0][s]=None
```

	0	1	2	3
0	Capitalized	Acquired	NaN	None
1	Dollars in millions	Software	Intangibles	Total
2	2013 for Q4	$146	$187	$333.0
3	2014	441	767	1207
4	2015	204	621	824
5	2016	39	579	618
6	2017	–	461	461

Presence of NaN's surrounding subheading's mainly

	4	5	6
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	NaN	NaN	None
3	And Other	Balance	NaN
4	Divestitures	Adjustments	12/31/12
5	$2	$42.0	$4357.0
6	0	6	2916
7	6	137	20405
8	14	6	1568
9	$22	$192.0	$29247.0

Any occurrences of NaN were then replaced with empty strings

	4	5	6
0			
1			
2			
3	And Other	Balance	
4	Divestitures	Adjustments	12/31/12
5	$2	$42.0	$4357.0
6	0	6	2916
7	6	137	20405
8	14	6	1568
9	$22	$192.0	$29247.0

FINAL RESULT

Dollars in millions	Balance	Goodwill	Price	And Other	Balance	
Segment	1/1/2012	Additions	Adjustments	Divestitures	Adjustments	12/31/2012
Global Business Services	$4,313.00	$5	$0	$2	$42.00	$4,357.00
Global Technology Services	2646	264	—	0	6	2916
Software	18121	2182	30	6	137	20405
Systems and Technology	1133	443	0	14	6	1568
Total	$26,213.00	$2,894	$30	$22	$192.00	$29,247.00

Problem 1 Part 2

Python Libraries used

1. To pass CIK, Accession Number , Amazon keys as arguments – Library argparse
2. In order to log activities – Library logging
3. In order to create csv and zip file and create a folder – Library csv, zipfile, os, datetime
4. Amazon s3 – Used boto3
5. Data Ingestion and Cleaning : Libraries :
6. Numpy,urllib,urllib3,html5lib,zipfile,urllib.request,pandas,BeautifulSoup

Methodology

1. We passed Accession Key, CIK, Amazon s3 keys , S3 location as arguments by the user
2. Handled scenarios when CIK and Accession Key are blank or if the values are missing with respective –help option
3. We have accomadated logging. Logging details can be found in the file name 'Part2.log'
4. Logging occurs at different instances. To name a few :
   a. Incorrect or blank user arguments – CIK,Accession Key, S3 loc, Amazon access keys
   b. Also occurs when these are found and program proceeds
   c. Creation of url, incorrect url or url not found
   d. Creation of directory to store table data
   e. Various stages of Data Cleaning process
   f. Table creation
   g. Table input into a csv
   h. Creation of zip file with log file and csvs

Logging implemented at each step

```
for i in pandu:
    try:
        msg=str(now)+"   got the table now checking if it has $ or % in it"
        logging.info(msg)
        x=None
        flag=0
        t=i.shape
        for j in range(0,t[0]):
            if flag==1:
                break
            for m in range(0,t[1]):
                if i[m][j]=="$" or i[m][j]=="%":
                    flag=1
                    x=i
                    break


        if len(i.columns)>3:
            msg=str(now)+"   only taking tables where there are more than 3 columns and cleaning the data"
            logging.info(msg)
            exah=x.replace(r'\n',"", regex=True)
            msg=str(now)+"   removing \n from table"
            logging.info(msg)
            example3=exah.dropna(axis=1,how="all")
            example4=example3.dropna(axis=0,how="all")
            msg=str(now)+"   removed all columns or rows with all nan values"
```

5. Creation of zip file in ‘zipdir’. Tables and logfile will be present in the zipfile created.
6. In order to create a bucket in Amazon S3 we use boto3.client and upload the zip file created.

```
try:
    fin2=now1.replace(":","")
    fin3=fin2.replace("-","")
    fin=fin3.replace(" ","")
    buckname="assignmenttablesteamfive"+fin
    client = boto3.client('s3',s3loc,aws_access_key_id=akey,aws_secret_access_key=skey)
    client.create_bucket(Bucket=buckname,CreateBucketConfiguration={'LocationConstraint':s3loc})
    client.upload_file("Assignment1Part1.zip", buckname, "Assignment1Part1.zip")
except:
    logging.warning("error upload to s3")
    print("error uploading to s3 choose a different location or check your keys")
```

7. Process of accessing url, parsing form , data cleaning remains same as Problem 1 Part1
8. In order to docker image is created and the file is called ‘Dock ’

```
FROM ubuntu:14.04
FROM python:3

RUN mkdir -p /usr/src/team5assignment1
WORKDIR /usr/src/team5assignment1

RUN pip install pandas
RUN pip install boto3
RUN pip install numpy
RUN pip install boto
RUN pip install urllib3
RUN pip install requests
RUN pip install html5lib
RUN pip install bs4
RUN pip install lxml

ENV ck 1
ENV ac 1
ENV akey 1
ENV skey 1
ENV sl 1



COPY Part2ofPart1.py /usr/src/team5assignment1/

CMD ["sh","-c","python /usr/src/team5assignment1/Part2ofPart1.py --cik ${ck} --acn ${ac} --akey ${akey} --skey ${skey}
--s3loc ${sl}"]
```

Problem 2

Python Libraries used

1. To pass CIK, Accession Number , Amazon keys as arguments – Library argparse
2. In order to log activities – Library logging
3. In order to create csv and zip file and create a folder – Library csv, zipfile, os, datetime
4. Amazon s3 – Used boto3
5. Data Ingestion and Cleaning : Libraries : Numpy,urllib,urllib3,html5lib,zipfile,urllib.request,pandas,BeautifulSoup

Methodology

1. We passed Accession Key, CIK, Amazon s3 keys , S3 location as arguments by the user

2. Handled scenarios when CIK and Accession Key are blank or if the values are missing with respective –help option
3. We have accomadated logging. Logging details can be found in the file name 'Part2.log'
4. Exception Handling: To name a few:
   a. Incorrect or blank CIK, Accession Number, Amazon Keys or S3 Location
   b. Blank year or incorrect year entered by user
   c. When stream of data is created to input into a csv
   d. When data for the particular 1st day of the month is not present
5. Once missing data is analyzed , summary metrics are calculated and plots are constructed. We put these into a zip file (python library zipfile) located in a directory created using os.path()
6. Next, using boto3 library we programmatically upload the file into Amazon s3

Checking for Anomalies

1. Counted the null values for each variables
2. In the below section of missing data analysis for categorical values we identified missing values for norefer, noagent, idx. If any other value existed except 0 and 1 these were replaced by NaN.

Missing Data Analysis

1. User enters a year , for which log data is generated
2. Formulated a 'for ' loop that iterates through the 12 months. In the process the url for each month is generated programmatically.
3. URL is retrieved (using library request)
4. Next since each url downloads a zip file, we used the python library zipfile to extract the zip file and read into the log csv file for the particular month.

5. In the case that if the first day of the month is empty, we formulated a for loop that would iterated through each URL of the particular month to find the next possible day that contains the data. The respective day number is then appended to the filename
6. A directory is created using the os.path and the log file is also input into the same location.
7. We formulated the next for loop that iterated through all months. Through this for loop all missing data analysis is handled. This for loop also verifies if the files are not empty and then extracts them. The data is then put into a data frame (name=k) for missing data analysis. Here, we input the data in for each month into a summary file, if there is no data found for 30 days it will provide a note saying no data present for this month.
8. If it does find data we move ahead with further missing data analysis

Allowable values

Handling Categorical Fields

1. We found that categorical fields such as idx, zone, cik (though not exactly categorical), code, no agent, norefer, crawler, find were represented as float values (i.e. Had '.00' preceding it). Hence, we converted their values to integers.
2. We then dropped any occurrences of none or NAN values and reindexed the columns.
3. On testing for data from other years we realized that categorical values might have values other than those designated ie. 0 or 1.
4. Since, as per the Read Me some of these will have values 0 or 1 only, we locked these fields to only accept values for 0 or 1. If any other value occurred it was replaced by NaN
5. The above was done for idx, norefer, noagent, crawler.
6. We locked the names of browsers as well. If any other value occurred it was replaced by NaN

Handling Numeric fields

7. Next we locked the 'Size' to be only greater than 0 as Sizes cannot be negative. If any other value occurred it was replaced by NaN which would then be dropped using 'dropna' command

Further Missing Data Analysis

Handling missing Categorical Values

1. If categorical values were missing we calculated the max occurrence of each Categorical field and replaced it with the max occurrence value.
2. This was done for browser, code, find, zone, extention, idx
3. norefer and noagent missing fields were replaced by 1
4. crawler values were replaced by 0.

Handle missing Numerical Values: Missing fields for size was replaced by the mean size grouped by idx

If CIK, Accession, Ip, date are empty fields, those records were dropped

These filled missing values are then appended into the summary file

Summary Metrics for Year 2005

1. **describe ()** function was used to provide the summary of the data generated. This has been done on a monthly basis

   Observation: Only the Summary for size fits the analysis and is most fit for further analy sis. The mean, min, max and standard deviation provide an insight that sizes range from `3.140000e+03 to 8.453789e+07 and the mean is approximately midway at 1.155019e+05.` This is evident considering the log files each have large file size

2. **Mean and Median file sizes for each browser were calculated :**

   Observation:

   Browser Mac has the highest file size of 207502.803976. In comparison to all browsers the mean size is least for browser DoCoMo|KDDI|Cricket|Vodaphone. Medians for the browsers vary from 5148 to 17018 for MSIE and Safari

3. **Mean and Total File sizes were calculated on the basis idx :**

   Observation: For users that did not land on the index page their mean file size and total file size was greater than those users who landed on the index page. This could mainly be because most of the users would have been routed to the desired page received the file they desired over the few number of users that were routed back to the index page as they might not have found the desired page.

4. **Median was calculated on the basis of idx**: As observed earlier for all the columns and the median file size is greater for idx=0 over idx=1.

Analysis on the selected Year 2005

1. **The top 5 CIK based on the count of hits**: the highest CIK on the basis of most number of hits is CIK number 103884. On further research we found that the company with the top 1 CIK is Alexander J Corp.

2. **Number of hits on the basis of Code**: The count of hits is the largest for Code ‘200’. This implies that most of the hits were successful. This is closely followed by code ‘304’ as these are conditional GET statements

3. **Number of hits on the respective day of the month**: The number of hits were the largest for the month of March 2015 on evaluating the first day of the 12 months of 2005. This could be possible as a lot of companies submit their filings at this time around.

4. **Average file size on the basis on date**: The Average file size for month of 2005 was the largest for month of January.

5. **Top 5 cik based on file size**: Amongst the Top 5 CIK with the largest Average file Size , the Cik that topped the list was the CIK 1199392. As per the web this company is Emgold Mining Corp having the largest average file size.

6. **Average Traffic per hour:** This graph provides a trend into the time at which there is minimum traffic and maximum traffic. As per the graph there is least traffic at 12 am and most traffic between the time frame of 4pm and 5pm. Additionally, it also provides an insight that the traffic is approximately same at 4 am and 11 pm at night. This implies that users might be utilizing these filings the most between 4pm and 5pm.

Anomalies on the selected Year

Anomalies were identified using boxplots

1. File Size by idx
   Observation: On plotting a boxplot for file size by idx we observe that there are no outliers. There is no file size that is beyond the minimum or maximum whiskers of the boxplot. Though it is evident that the file size exceptionally shoots up for idx=0. This is because idx=0 implies it is not the index page and hence, the page could consists of filings which have large file sizes

2. File Size by Date:
   In the box plot for file size vs date also there are no outliers in the data. All values seem to occur within the minimum and maximum whiskers. Though it is evident that large number of values lie between the upper quartile and lower quartile ranges for the month of January where the size of the file is maximum

3. Anomalies in File Size based on the dataset

   There are no file size outliers present in the data, but it is evident that there are values ranging from the lower quartile to the upper quartile. The maximum file size is greater than 175000 where as the minimum is 0 in the case of empty files

Missing Data Analysis

Iterating through the months to retrieve data for each month

```
if years<2003 or years>2016:
    msg=now1+"  system exited because the data enetered for the year was not present"
    logging.info(msg)
    sys.exit(0)
try:
    for m in range(1,13):
        qtr=None
        if m==1 or m==2 or m==3:
            m1="0"+str(m)
            qtr="Qtr1"
        elif m==4 or m==5 or m==6:
            m1="0"+str(m)
            qtr="Qtr2"
        elif m==7 or m==8 or m==9:
            m1="0"+str(m)
            qtr="Qtr3"
        elif m==10 or m==11 or m==12:
            m1=str(m)
```

Extraction from zip and reading the csv

```
  logging.info(msg)
  r =requests.get(url,stream=True)
  z = zipfile.ZipFile(io.BytesIO(r.content))
  x=z.extractall(os.path.join('Part2'))
  j = pd.read_csv(os.path.join('Part2',filename))
  if j.empty:
      for i in range(1,31):
          y=None
          if i<10:
              y="0"+str(i)
          else:
              y=str(i)
          url="http://www.sec.gov/dera/data/Public-EDGAR-log-file-data/" + str(years)+ "/"+qtr+"/"+"log"+str(
          years)+str(m1)+y+".zip"
          filename="log"+str(years)+str(m1)+y+".csv"
          r =requests.get(url,stream=True)
          z = zipfile.ZipFile(io.BytesIO(r.content))
          x=z.extractall(os.path.join('Part2'))
          jj = pd.read_csv(os.path.join('Part2',filename))
```

Checking for missing data for a month (then input data in Pandas Library data Frame. Other libraries used : zipfile, request, io, logging for logging, extraction and reading of zip file)

```
    qtr="Qtr4"
msg=str(now)+"    Reading csvs one by one directory for saving tables"
logging.info(msg)
filename="log"+str(years)+str(m1)+"01.csv"
k = pd.read_csv(os.path.join('Part2',filename))
summaryfile=os.path.join('Part2',str(years)+'Summaries.csv')
f = open(summaryfile, 'a')
if k.empty:
    y=None
    for i in range(1,31):
        y=None
        if i<10:
            y="0"+str(i)
        else:
            y=str(i)
        filename="log"+str(years)+str(m1)+y+".csv"
        k = pd.read_csv(os.path.join('Part2',filename))
        if k.empty:
            continue
        else:
            break
    if y=="30" and k.empty:
        f.write("No data present for this particular month    ")
        f.write(filename)
        f.write('\n \n \n \n')
        continue
```

Before changing data type of categorical values

```
Data Frame before missing data analysis
                      ip        date      time   zone        cik  \
0          68.70.131.gjc  2005-01-01  00:00:00  500.0   933239.0
1          68.70.131.gjc  2005-01-01  00:00:00  500.0   932696.0
2          68.70.131.gjc  2005-01-01  00:00:00  500.0   928022.0
3          68.70.131.gjc  2005-01-01  00:00:00  500.0   924901.0
4          68.70.131.gjc  2005-01-01  00:00:00  500.0   929351.0
5          68.70.131.gjc  2005-01-01  00:00:00  500.0   930305.0
6          68.70.131.gjc  2005-01-01  00:00:00  500.0   928022.0
7          68.70.131.gjc  2005-01-01  00:00:00  500.0   929351.0
8          68.70.131.gjc  2005-01-01  00:00:00  500.0   928022.0
9          68.70.131.gjc  2005-01-01  00:00:00  500.0   929351.0
10         68.70.131.gjc  2005-01-01  00:00:00  500.0   930305.0
```

After changing data type of categorical values

```
                      ip        date      time  zone      cik  \
0          68.70.131.gjc  2005-01-01  00:00:00   500   933239
1          68.70.131.gjc  2005-01-01  00:00:00   500   932696
2          68.70.131.gjc  2005-01-01  00:00:00   500   928022
3          68.70.131.gjc  2005-01-01  00:00:00   500   924901
4          68.70.131.gjc  2005-01-01  00:00:00   500   929351
5          68.70.131.gjc  2005-01-01  00:00:00   500   930305
6          68.70.131.gjc  2005-01-01  00:00:00   500   928022
7          68.70.131.gjc  2005-01-01  00:00:00   500   929351
8          68.70.131.gjc  2005-01-01  00:00:00   500   928022
9          68.70.131.gjc  2005-01-01  00:00:00   500   929351
10         68.70.131.gjc  2005-01-01  00:00:00   500   930305
```

Before handling missing values or NaN values

19	1.0	1.0	0.0	0.0	0.0	mie
20	0.0	1.0	0.0	0.0	0.0	mie
21	0.0	1.0	0.0	0.0	0.0	mie
22	0.0	1.0	0.0	0.0	0.0	mie
23	1.0	1.0	0.0	0.0	0.0	mie
24	1.0	1.0	0.0	0.0	0.0	mie
25	1.0	1.0	0.0	0.0	0.0	mie
26	0.0	1.0	0.0	0.0	0.0	mie
27	0.0	1.0	0.0	0.0	0.0	NaN
28	0.0	1.0	0.0	0.0	0.0	mie
29	0.0	1.0	0.0	0.0	0.0	mie
...	...	...	...	...	...	...
262434	0.0	0.0	0.0	9.0	0.0	win
262435	0.0	0.0	0.0	1.0	0.0	win
262436	1.0	0.0	0.0	1.0	0.0	win
262437	0.0	1.0	1.0	0.0	0.0	NaN
262438	0.0	0.0	0.0	9.0	0.0	win
262439	1.0	0.0	0.0	1.0	0.0	win
262440	1.0	0.0	0.0	1.0	0.0	win
262441	0.0	1.0	1.0	0.0	0.0	NaN
262442	1.0	0.0	0.0	1.0	0.0	win
262443	1.0	0.0	0.0	1.0	0.0	win
262444	0.0	0.0	0.0	9.0	0.0	win
262445	1.0	0.0	0.0	4.0	0.0	win
262446	0.0	0.0	0.0	1.0	0.0	win
262447	1.0	0.0	0.0	1.0	0.0	win

After Handling missing data for categorical and numerical values

18	0.0	1.0	0.0	0	0.0	mie
19	1.0	1.0	0.0	0	0.0	mie
20	0.0	1.0	0.0	0	0.0	mie
21	0.0	1.0	0.0	0	0.0	mie
22	0.0	1.0	0.0	0	0.0	mie
23	1.0	1.0	0.0	0	0.0	mie
24	1.0	1.0	0.0	0	0.0	mie
25	1.0	1.0	0.0	0	0.0	mie
26	0.0	1.0	0.0	0	0.0	mie
27	0.0	1.0	0.0	0	0.0	win
28	0.0	1.0	0.0	0	0.0	mie
29	0.0	1.0	0.0	0	0.0	mie
...	...	...	...	...	...	...
3025513	0.0	0.0	0.0	9	0.0	win
3025514	0.0	0.0	0.0	1	0.0	win
3025515	1.0	0.0	0.0	1	0.0	win
3025516	0.0	1.0	1.0	0	0.0	win
3025517	0.0	0.0	0.0	9	0.0	win
3025518	1.0	0.0	0.0	1	0.0	win
3025519	1.0	0.0	0.0	1	0.0	win
3025520	0.0	1.0	1.0	0	0.0	win

Locking the allowable values for idx, zone, cik, code, no agent, norefer, crawler,size and finding max occurred values to replace missing values

```
#Locking fields for allowable values only
    msg=str(now)+"    Validating the column values"
    logging.info(msg)
    k1.loc[k1['Size']<0,'Size']=np.nan
    k1.loc[~(k1['idx'].isin([0,1])),'idx']=np.nan
    k1.loc[~(k1['norefer'].isin([0,1])),'norefer']=np.nan
    k1.loc[~(k1['noagent'].isin([0,1])),'noagent']=np.nan
    k1.loc[~(k1['crawler'].isin([0,1])),'crawler']=np.nan
    k1.loc[~(k1['browser'].isin(['mie','fox','saf','chr','sea','opr','oth','win','mac','lin','iph','ipd','and',
    'rim','iem'])),'browser']=np.nan

#checking the maximum count of the browser for the year
    msg=str(now)+"    Finding the the browser eith maximum count for that year"
    logging.info(msg)
    max_bro = k1.groupby(["browser"]).size().rename('countmaxbrowser').idxmax()
#print(max_bro)
#checking the maximum count of the code for the year
    max_code= k1.groupby(["code"]).size().rename('countmaxcode').idxmax()
    msg=str(now)+"    checking the max code for that month"
    logging.info(msg)
```

Filling missing values with Max occurred values and mean for the missing sizes

```
#Filling Nan values with max Categorical Values
    msg=str(now)+"    Replacing the nan values with appropriate values"
    logging.info(msg)
    k1['browser'].fillna(max_bro,inplace=True)
    k1['code'].fillna(max_code,inplace=True)
    k1['find'].fillna(max_find,inplace=True)
    k1['zone'].fillna(max_zone,inplace=True)
    k1['extention'].fillna(max_extention,inplace=True)
    k1['idx'].fillna(max_idx,inplace=True)

#Filling empty values with Categorical Values
    k1['norefer'].fillna(1,inplace=True)
    k1['noagent'].fillna(1,inplace=True)
    k1['crawler'].fillna(0,inplace=True)

#Filling Numerical Data Size Grouping by idx and filling by mean
    k1['Size']=k1.groupby(['idx'])['Size'].apply(lambda x: x.fillna(x.mean()))
```

Summary Metrics

Summary of Dataset using describe()

	ip	date	time	zone	cik	\
count	3025543	3025543	3025543	3.025543e+06	3.025543e+06	
unique	96555	12	86399	NaN	NaN	
top	68.75.189.haa	2005-03-01	14:46:12	NaN	NaN	
freq	59179	662028	115	NaN	NaN	
mean	NaN	NaN	NaN	4.568776e+02	-3.048500e+12	
std	NaN	NaN	NaN	4.952474e+01	5.302590e+15	
min	NaN	NaN	NaN	4.000000e+02	-9.223372e+18	
25%	NaN	NaN	NaN	4.000000e+02	7.550030e+05	
50%	NaN	NaN	NaN	5.000000e+02	9.468420e+05	
75%	NaN	NaN	NaN	5.000000e+02	1.133872e+06	
max	NaN	NaN	NaN	5.000000e+02	1.000000e+10	

	accession	extention	code	Size	\
count	3025543	3025543	3.025543e+06	3.025543e+06	
unique	711884	225206	NaN	NaN	
top	0001209191-03-031123	-index.htm	NaN	NaN	
freq	4069	1137907	NaN	NaN	
mean	NaN	NaN	2.108405e+02	1.155019e+05	
std	NaN	NaN	3.676108e+01	5.169564e+05	
min	NaN	NaN	2.000000e+02	0.000000e+00	
25%	NaN	NaN	2.000000e+02	3.140000e+03	
50%	NaN	NaN	2.000000e+02	5.719000e+03	
75%	NaN	NaN	2.000000e+02	1.934000e+04	
max	NaN	NaN	5.020000e+02	8.453789e+07	

Median of all columns with respect to idx

idx	zone	cik	code	Size	norefer	noagent	find	crawler
0.0	500	944802	200	13266.0	1.0	0.0	0	0.0
1.0	500	1000181	200	3128.0	0.0	0.0	1	0.0

Mean and Median file sizes for each browser were calculated

browser	Size_mean	Size_median	crawler_len
200	68753.491218	5578.000000	1430806.0
fox	159586.747236	6405.500000	2272.0
iem	136467.043579	9673.000000	25.0
lin	66865.497286	7183.500000	19254.0
mac	207502.803976	9145.000000	8707.0
mie	85482.647048	5148.000000	106604.0
opr	202679.271488	11595.000000	144.0
oth	15956.094118	16607.000000	85.0
rim	185970.837925	4604.000000	89.0
saf	50714.500000	17018.000000	6.0
win	163605.052232	5719.089472	1457551.0

File size with respect to idx

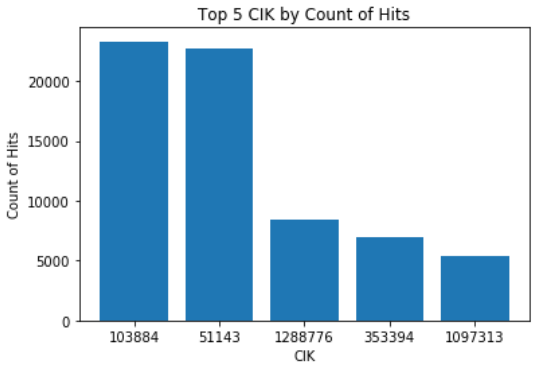
idx	Size_mean	Size_sum
0.0	184975.485887	3.427498e+11
1.0	5719.089472	6.706181e+09

Group by Idx

idx	zone	cik	code	Size	norefer	noagent	find	crawler
0.0	500	944802	200	13266.0	1.0	0.0	0	0.0
1.0	500	1000181	200	3128.0	0.0	0.0	1	0.0

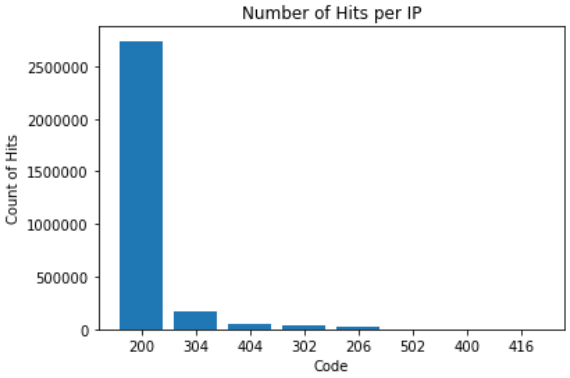
Top 5 CIK based on the count of hits

	cik	count
3016	103884	23329
1458	51143	22785
83659	1288776	8401
4483	353394	7018
37854	1097313	5387

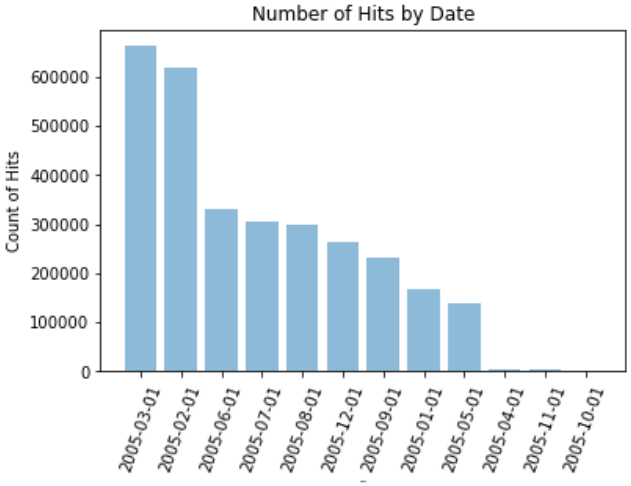


Number of Hits on the basis of Code

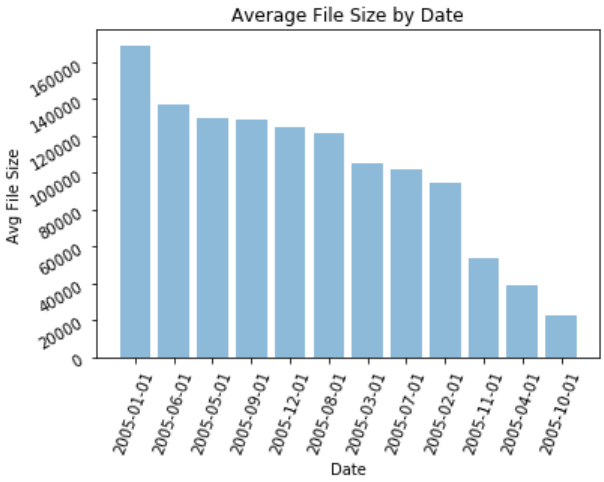
	code	count
0	200	2740921
3	304	170861
5	404	51218
2	302	43337
1	206	19044
7	502	129
4	400	29
6	416	4



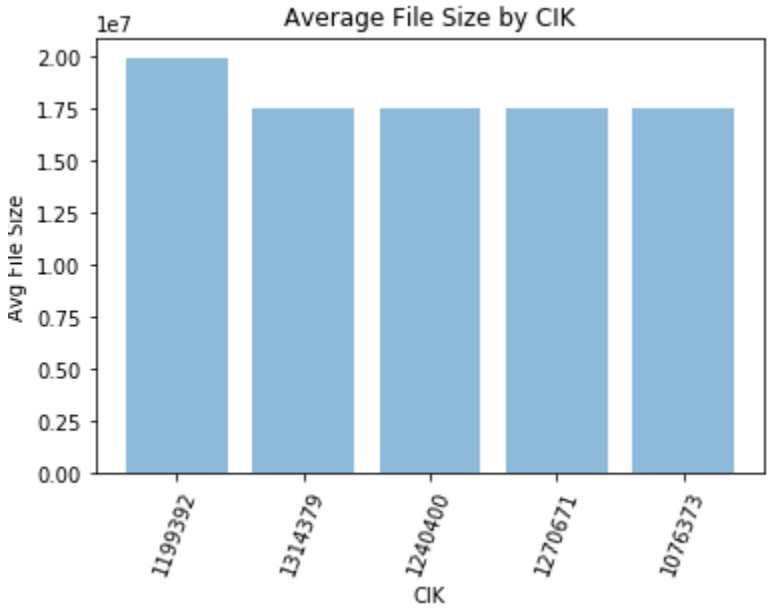
Number of hits by Date



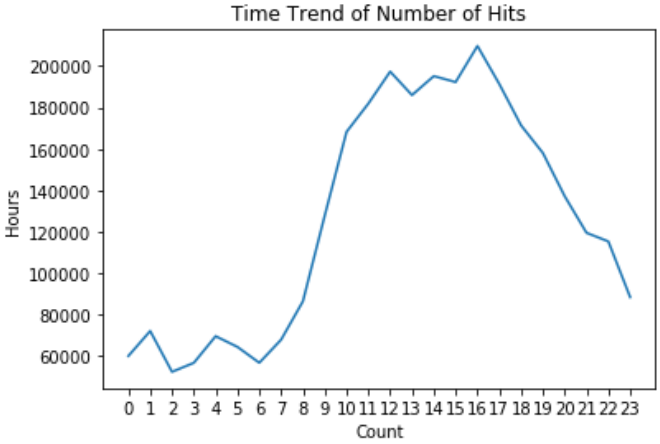
Average file Size by Date



Top 5 CIK by average file size



Time Series Graph of Traffic Activity through the day


Time Trend of Number of Hits
Hours
200000
180000
160000
140000
120000
100000
80000
60000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Count