**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/jmsjigyasa/Assignment1ADS](%20https:/github.com/jmsjigyasa/Assignment1ADS)

**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**

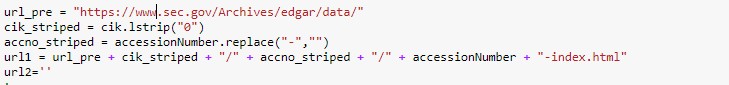
**Problem 1 Part 1 Abstract:**

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.

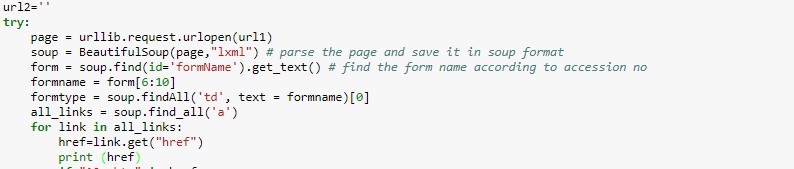
Process Overview

**Step 1**

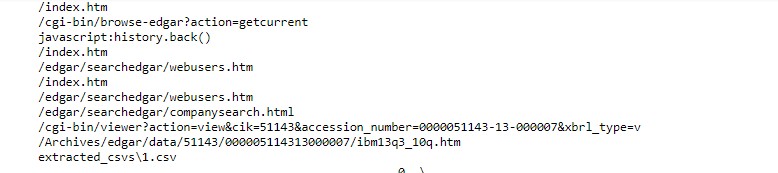
1. **Web Scraping:**
   1. Programmatically generated the URL (url\_pre + cik\_striped + "/" + accno\_striped + "/" + accessionNumber + "-index.html").
   2. The CIK and Accession Number was accessed using arguments. The library used was argparse
   3. We used the python library Beautiful Soup to parse 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.
   4. 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.



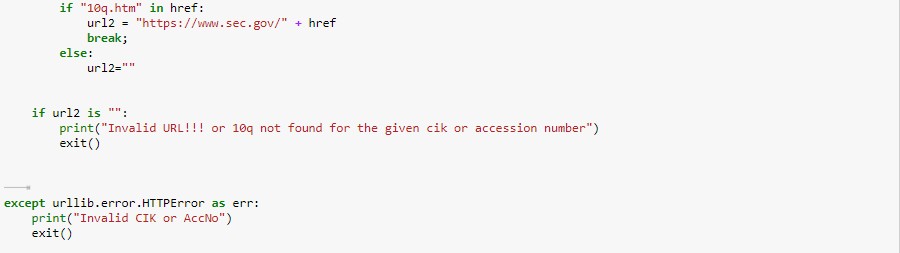
# Selecting all href links



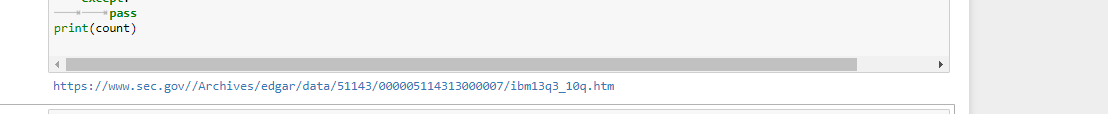
**List of href links generated**



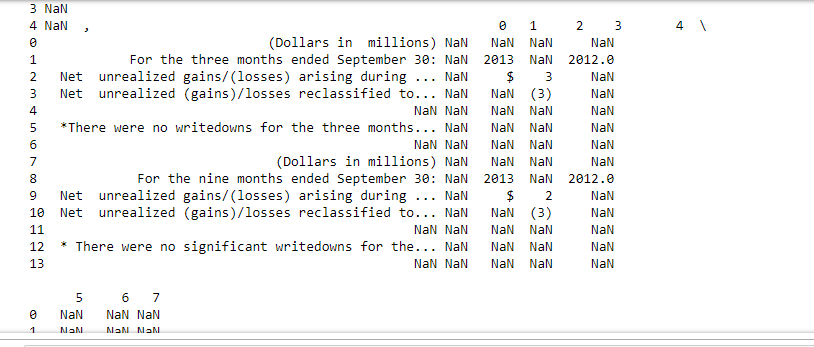
**Explicitly selecting the 10Q links and Exception Handling**



**10 Q URL generated**



# Reading from 10 Q file



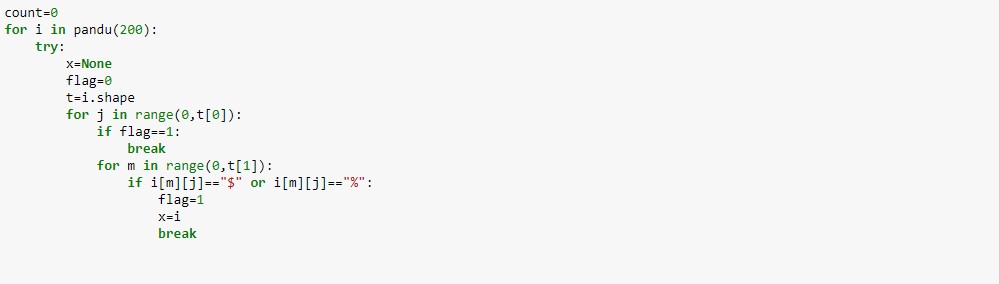
**Step 2**

**Data Cleansing**

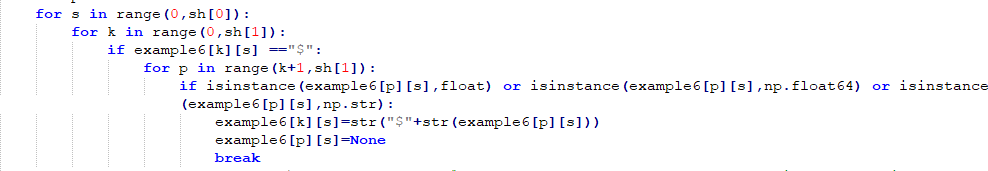
The data when inserted into the data frame was out of format, columns were not organized and consisted of NaN, None and empty strings.

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)

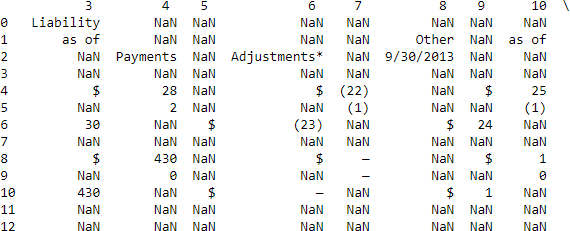
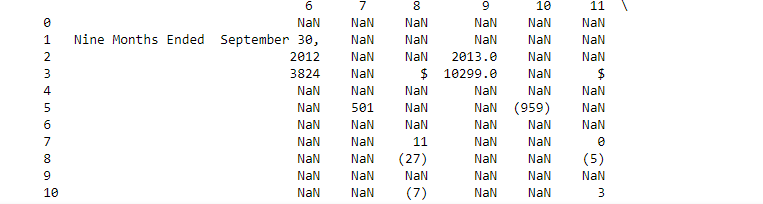
# Code to extrapolate all table fields with occurrences of ‘$” or ‘%’ (Used Pandas library and stored data in a Pandas Data Frame)



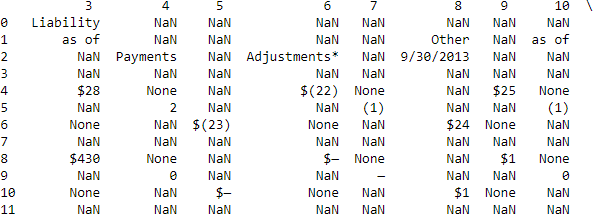
**Code to append ‘$’ sign to numbers preceding the ‘$’ cell (Used the Numpy Library)**



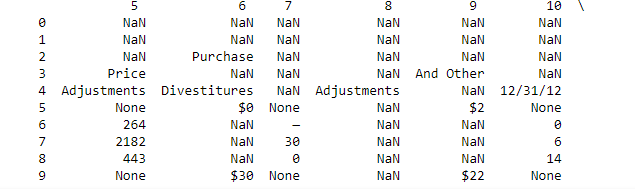
**Snippet before appending ‘$’ to the number**



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

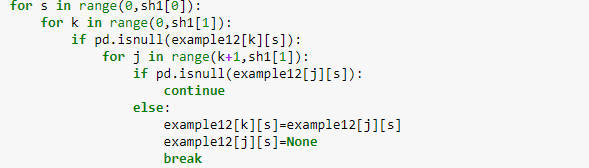


Eliminated all parenthesis (Used Regex expressions)

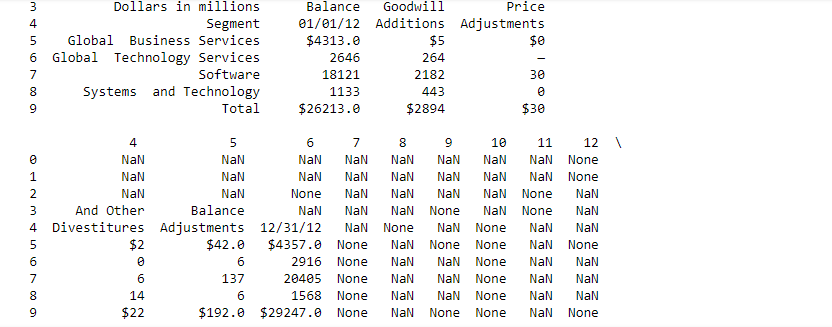


Rearranging columns values

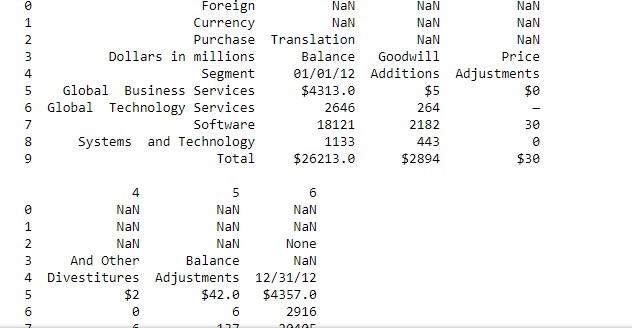
We re-arranged the values to their respective columns using the following logic (Used Pandas Library to check if a field was null (pd.isnull())



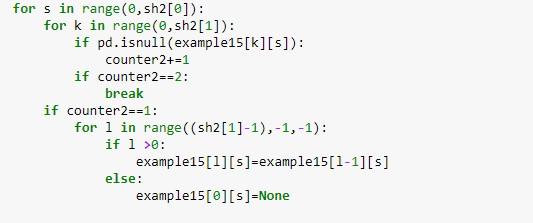
# Output after running the above for loop

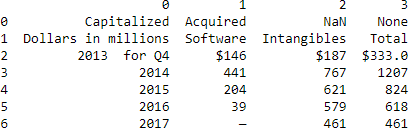


**All rows and columns that had only NaN or None values were eliminated (axis =0 and axis=1)**

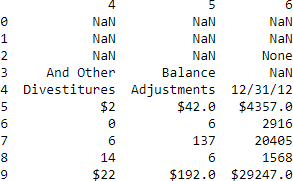


**We used the following code snippet to rearrange the columns such that the column values are aligned by the column names. (Pandas Library, Pandas Data Frame)**

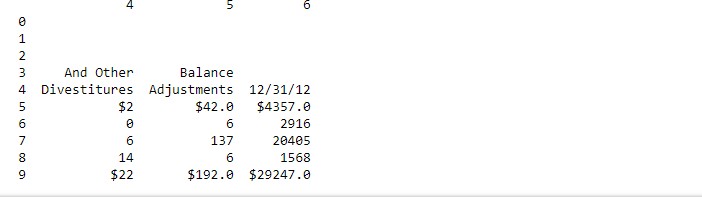




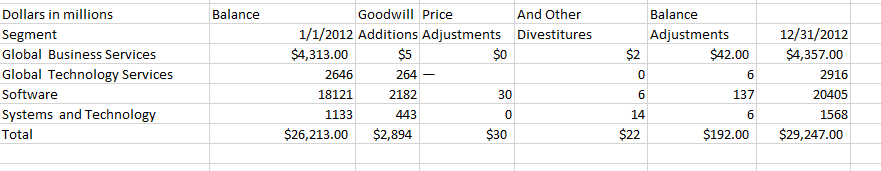
**Presence of NaN’s surrounding subheading’s mainly**



**Any occurrences of NaN were then replaced with empty strings**



**FINAL RESULT**



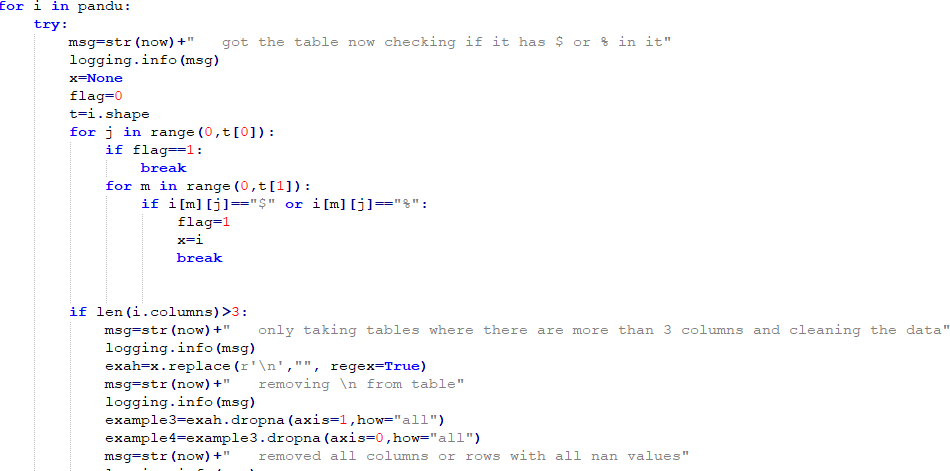
**Problem 1 Part 2 Python Libraries used**

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

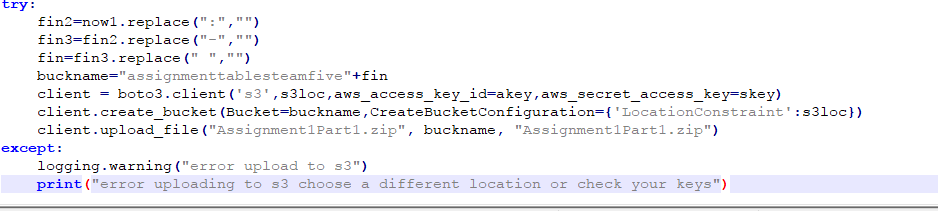
# Methodology

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 :
   1. Incorrect or blank user arguments – CIK,Accession Key, S3 loc, Amazon access keys
   2. Also occurs when these are found and program proceeds
   3. Creation of url, incorrect url or url not found
   4. Creation of directory to store table data
   5. Various stages of Data Cleaning process
   6. Table creation
   7. Table input into a csv
   8. Creation of zip file with log file and csvs

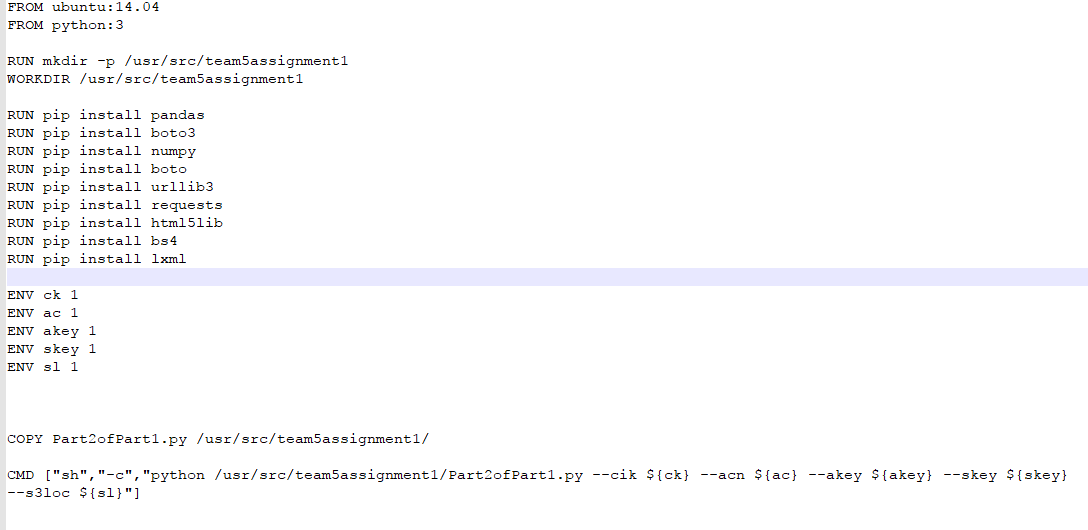
# Logging implemented at each step



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



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



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:
   1. Incorrect or blank CIK, Accession Number, Amazon Keys or S3 Location
   2. Blank year or incorrect year entered by user
   3. When stream of data is created to input into a csv
   4. 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

1. 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

# 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

# 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.

1. **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

1. File Size by Date:

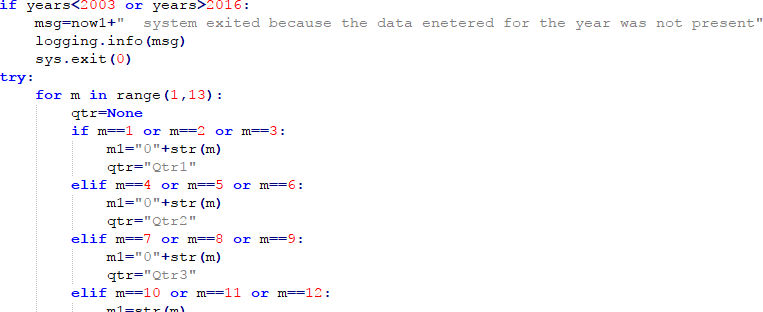
In the box plot for file size vs date also there are no outliers in the data. All values seem to occur within the minimum and maximum whiskers. Though it is evident that large number of values lie between the upper quartile and lower quartile ranges for the month of January where the size of the file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

1. Anomalies in File Size based on the dataset

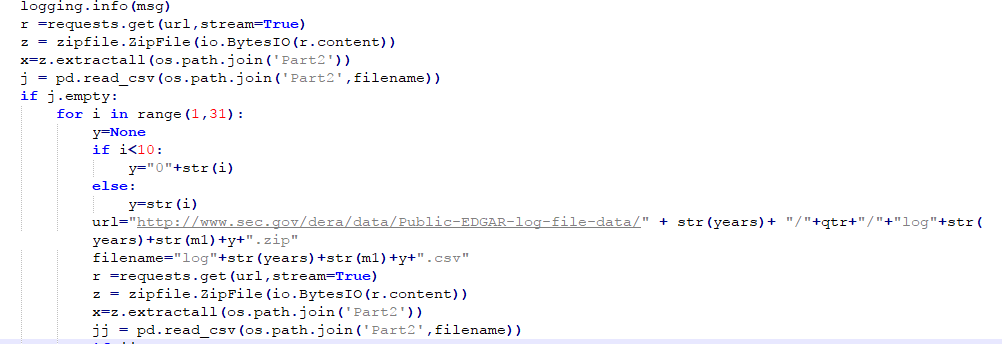
There are no file size outliers present in the data, but it is evident that there are values ranging from the lower quartile to the upper quartile. The maximum file size is greater than 175000 where as the minimum is 0 in the case of empty files

# Missing Data Analysis

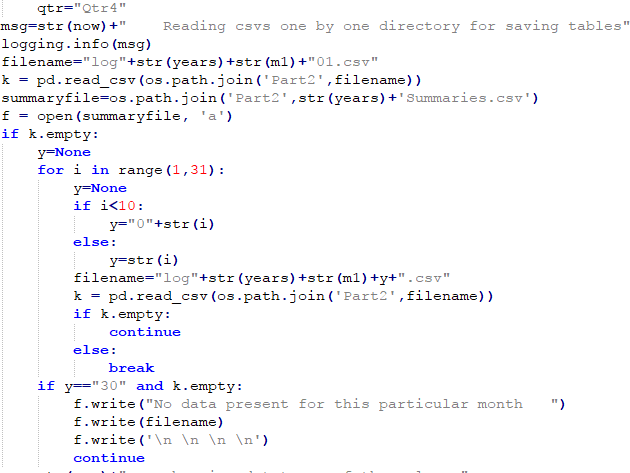
Iterating through the months to retrieve data for each month



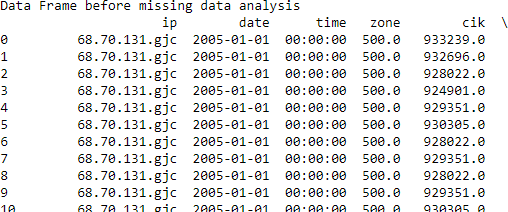
# Extraction from zip and reading the csv



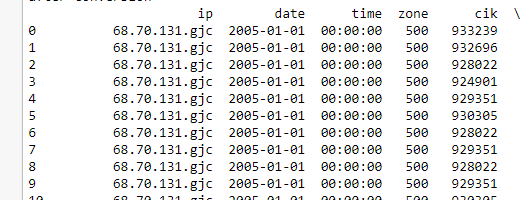
Checking for missing data for a month (then input data in Pandas Library data Frame. Other libraries used : zipfile, request, io, logging for logging, extraction and reading of zip file)



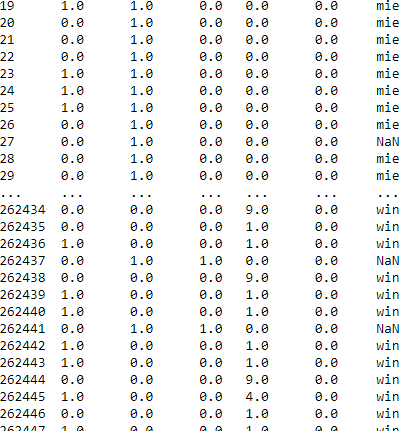
# Before changing data type of categorical values



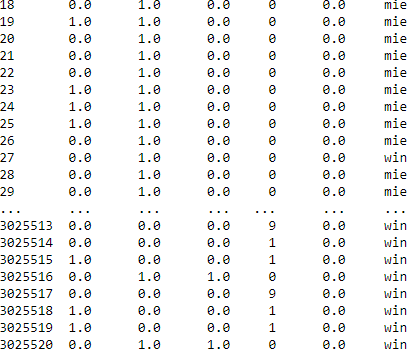
**After changing data type of categorical values**



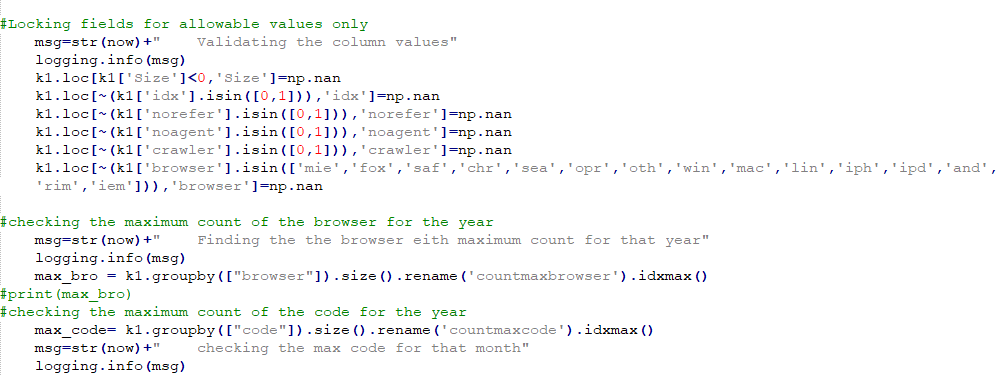
**Before handling missing values or NaN values**



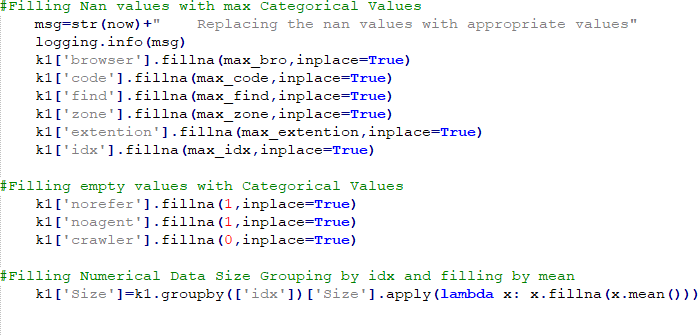
**After Handling missing data for categorical and numerical values**



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

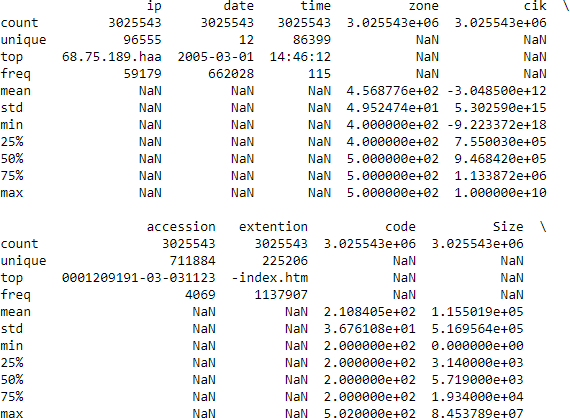


Filling missing values with Max occurred values and mean for the missing sizes

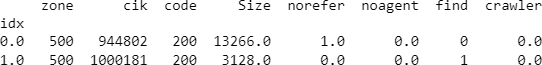


# Summary Metrics

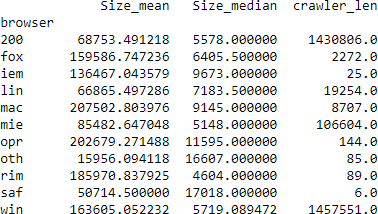
Summary of Dataset using describe()



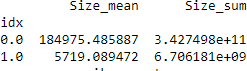
# Median of all columns with respect to idx



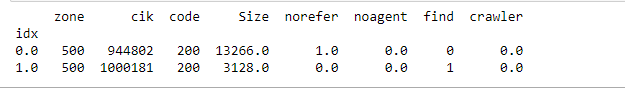
**Mean and Median file sizes for each browser were calculated**



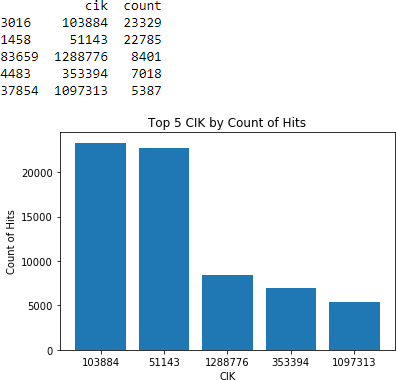
**File size with respect to idx**



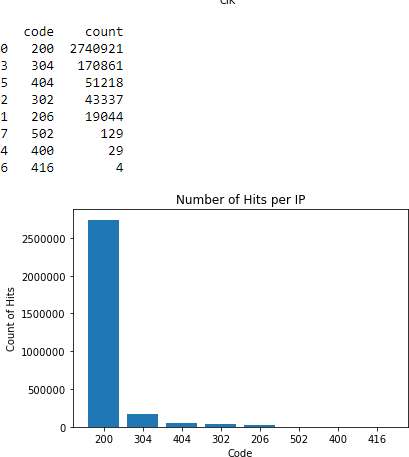
**Group by Idx**



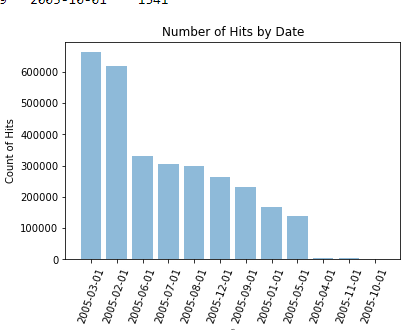
**Top 5 CIK based on the count of hits**



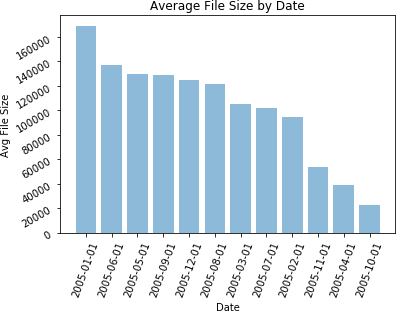
**Number of Hits on the basis of Code**



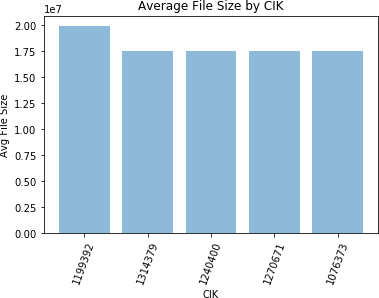
**Number of hits by Date**



**Average file Size by Date**



**Top 5 CIK by average file size**



**Time Series Graph of Traffic Activity through the day**

