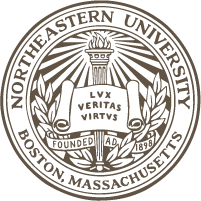
[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiAxdP4_qrZAhUlxoMKHZDACFUQjRwIBw&url=https://www.northeastern.edu/guidelines/resources/index.html&psig=AOvVaw3ZXiw7Od7CjTCeUGCehvMB&ust=1518889630579473)

ADVANCE IN DATA SCIENCE

AND

ARCHITECTURE

Data Analysis Edgar data from text files

SUBMITTED BY

DIVYA PRIYA EMMANUEL

GAURANG DAVDA

NIRANJHANI VASUDEVAN

**Abstract**

This report summarizes the implementation of Data wrangling executed on EDGAR website. The following task are performed as part of the assignment :

Section 1 : Extracting and Analysing log files from EDGAR website with given year

Section 2 : Handling missing values and examine summary metrics for the dataset extracted

from log files

Section 3 : Dockerize the process

**Section 1 - Parse the file**

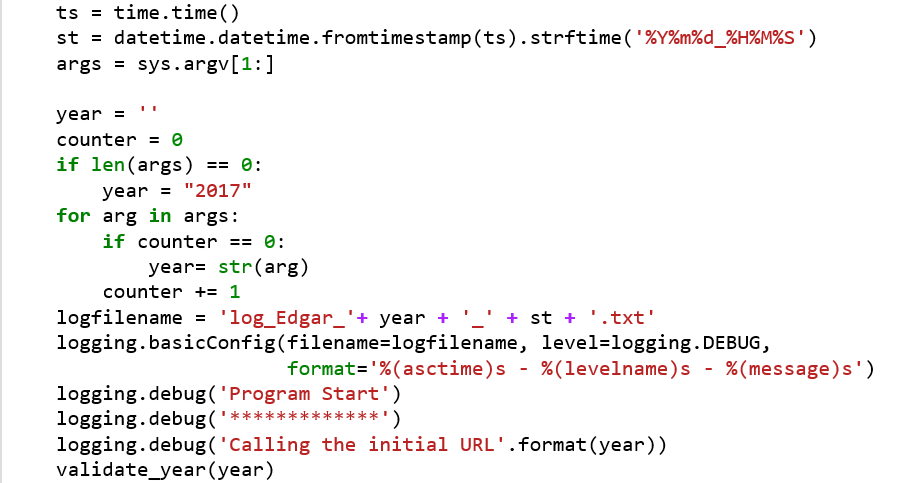
**STEP 1: Analysis of the log files**

EDGAR log files records and stores user access statistics for SEC.gov website in CSV format extracted from Apache log files. The log files contains certain limitations, including existence of lost or damaged files, and in addition it may contain inaccuracies or other errors.

* Log files contains lot of NaN
* There was abnormal values observed in some columns
* Some values are randomly populated
* Lot of rows are repeated with same values
* It was difficult to analysis dataset with that large volume of size
* There are lot of float datatype hold by columns although the value is contain integer

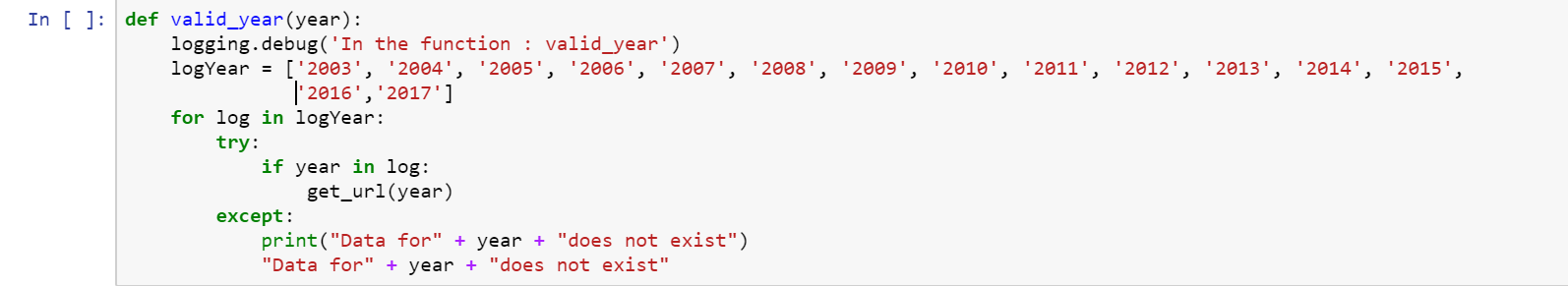
**STEP 2: Get the year for which Data cleaning to be performed**

The below code helps us to get a particular year value for which data cleaning to be performed. The EDGAR log file contains log file for each day from 2003 to 2017 as on when this report was created.



**STEP 3: Validation of the input year**

In order to get a valid year , this step makes sure that the given year is present in the EDGAR log file data set or not. If the given year is invalid, the program never runs further and logs a file regarding the same.



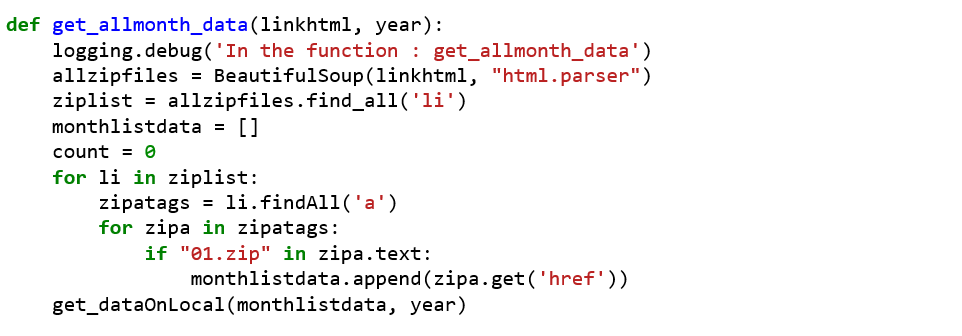
**STEP 4: Create all log file URL by hitting the input year URL**

This step creates and hits the URL with the given year. It parse the HTML content to obtain all the ‘href’ belonging to the particular year.



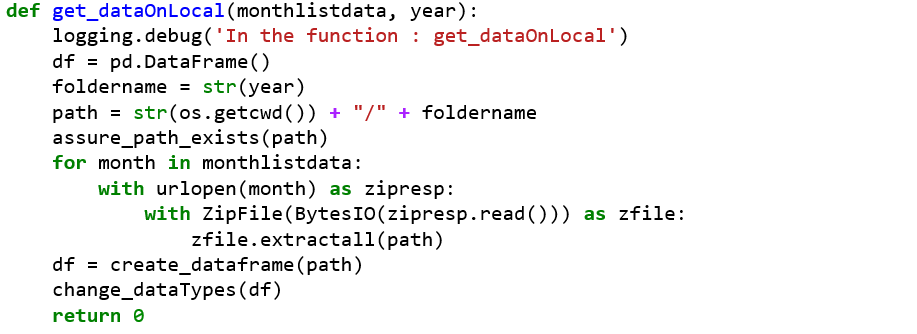
**STEP 4: Function to acquire first day log file of every month**

All the log files present in EDGAR website as CSV file in a ZIP. So when we hit the actual URL ZIP files are downloaded. As per the question, this function extracts log file of every month in a given year.

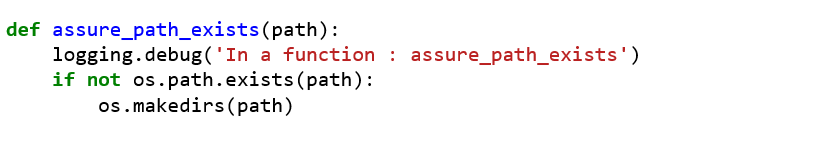


**STEP 5: Dataframe creation**

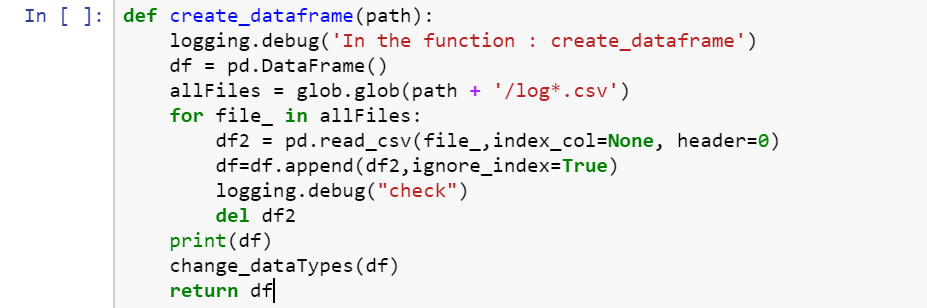
A directory is created with a folder in which zip data are extracted. There are two functions included in this code in order to create Dataframe and validate whether the directory path created or not.



This validates the directory creation :



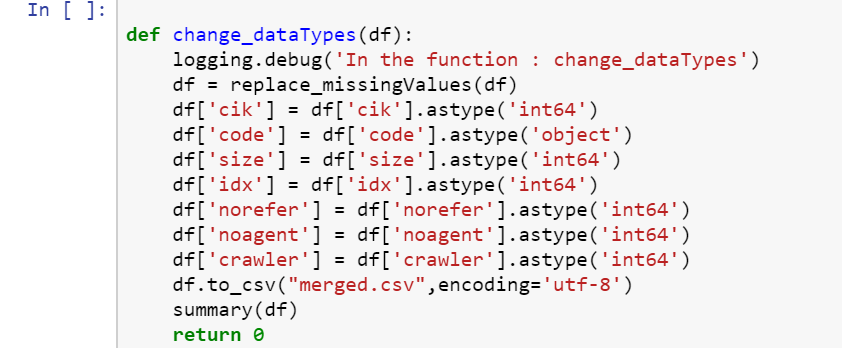
This is the function which creates the dataframe by merging log files one by one. It is done by reading the CSV present in the directory created by ‘get\_datalocal’ function.



**Section 2 – Handling Missing data**

**STEP 1: Changing Datatype**

The data gets messy mostly when it is read from a CSV file. The frequent affected feature is the datatype of the data. Hence by inspecting the created dataframe, most of the int datatypes were changed to float. Below code changes the unwanted datatype :



**STEP 2: Fix the NaN in all 15 columns**

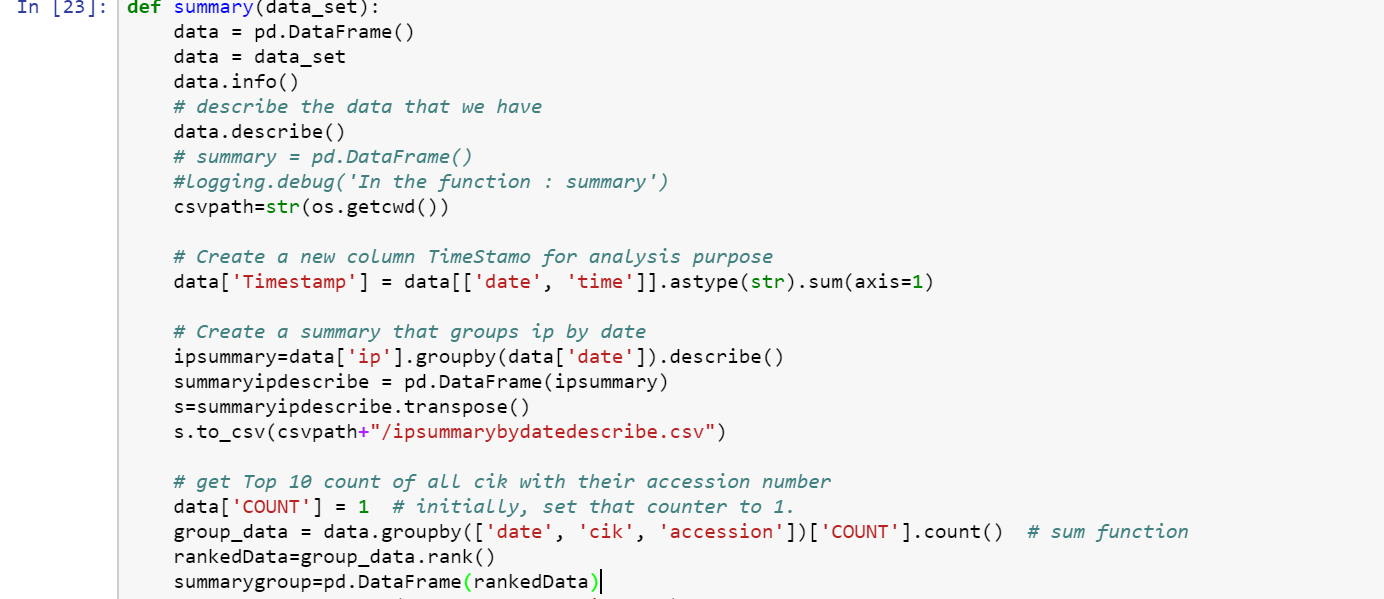
Each of the column is handled separately with generic code irrespective of any csv files with different dataset.

* cik, ip, accession, date and time column: These columns are most important so if any value missing in one of mentioned then the corresponding row will be dropped from the dataframe
* zone – replace NaN with the frequent occurrence in the column.
* browser – replace NaN with ‘missing’ string.
* extension – replace NaN by grouping the column with the ip and fill the frequent occurrence.
* Code – replace NaN with the “unknown\_status string”
* size – replace NaN with 0
* idx – check if the value of idx is either 0 or 1(Categorical information). And fill the NaN by frequent occureence in the column
* norefer - check if the value of norefer is either 0 or 1(Categorical information). And fill the NaN by frequent occureence in the column
* noagent - check if the value of noagent is either 0 or 1(Categorical information). And fill the NaN by frequent occureence in the column
* find - check if the value is between 0 to 10. And fill the NaN 0. We do this because the value 0 doesn’t have any action oriented provided by EDGAR website.



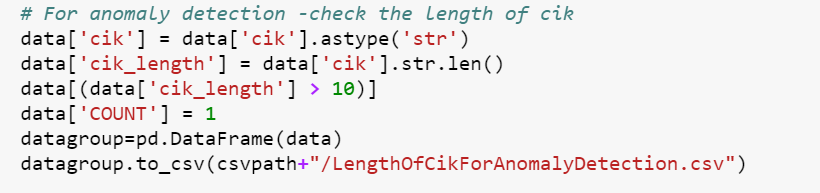
**STEP 3: Summary metrics**

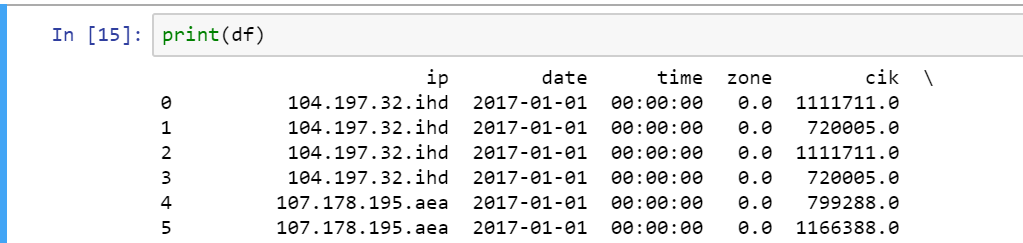
* Data type of each columns along with the memory used by the data in the data frame
* Describe – it gives the total count of the rows, mean value and std deviation etc. . for each column that has integer as a data type
* Grouping ip by date
* Get the top 10 company that logs the file in the given year.



**STEP 4: Anomalies present**

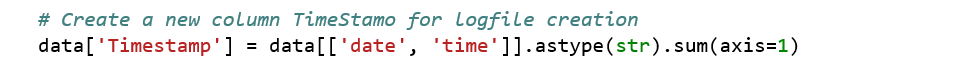
* As per EDGAR website each cik number should be a 10 digit long. But we have observed that in our dataset there are variations in the length of CIK number.
* Brower column should have 3 character string which describes the browser used by each user. We noticed that some of the values in the column are containing irrelevant integers followed by decimal in a ip address format.

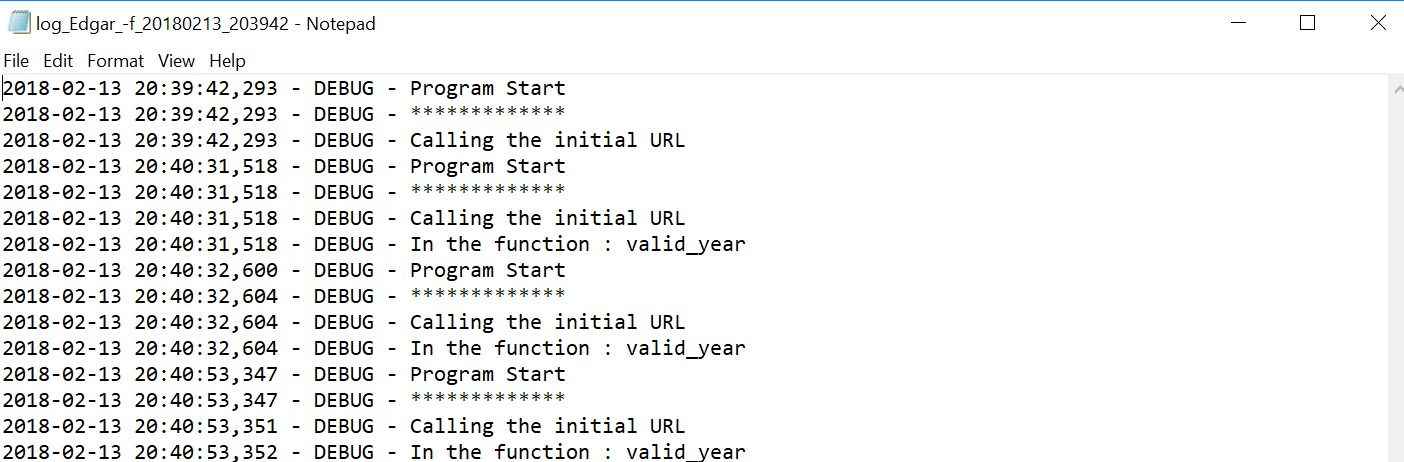




**STEP 5: Capturing each action in a log file**

A log file is created to capture each task performed in Section 1 process along with timestamp which will make analysis at ease if any issue faced by a user. The file also logs the last point of failure based on the given inputs.

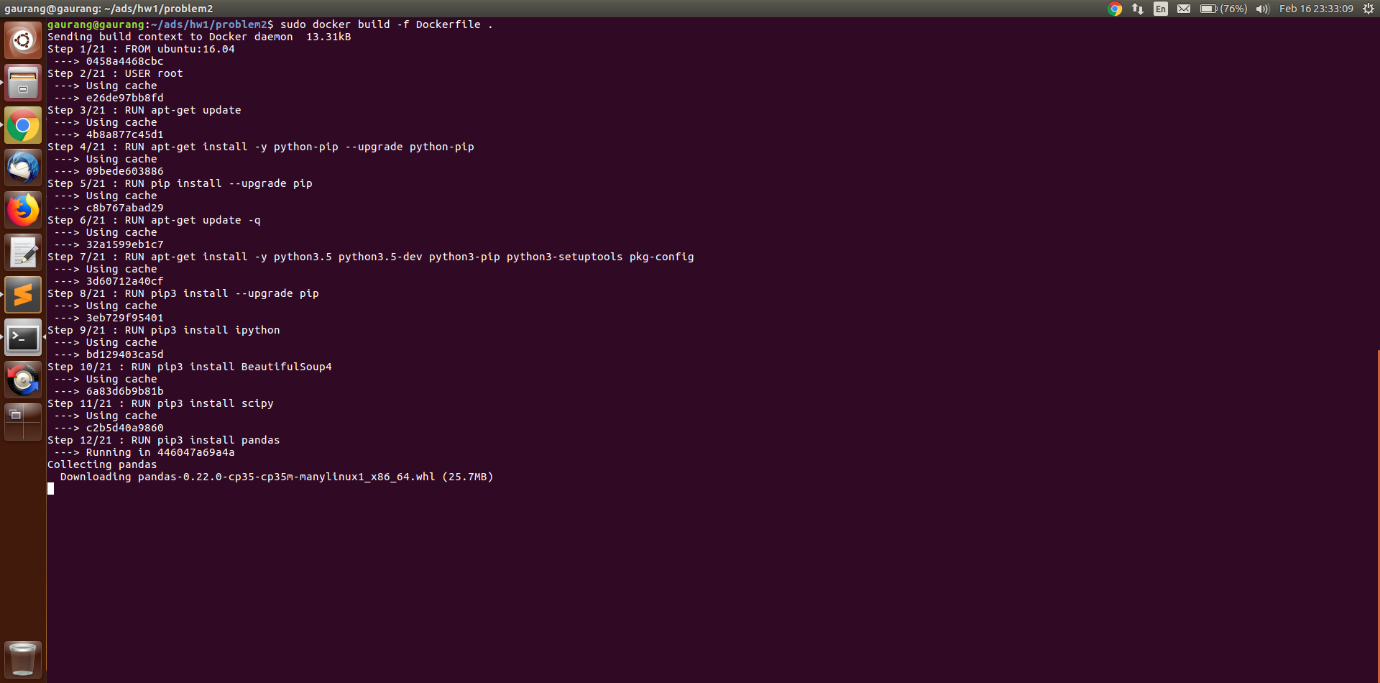




**Section 3 – Dockerize the pipeline**

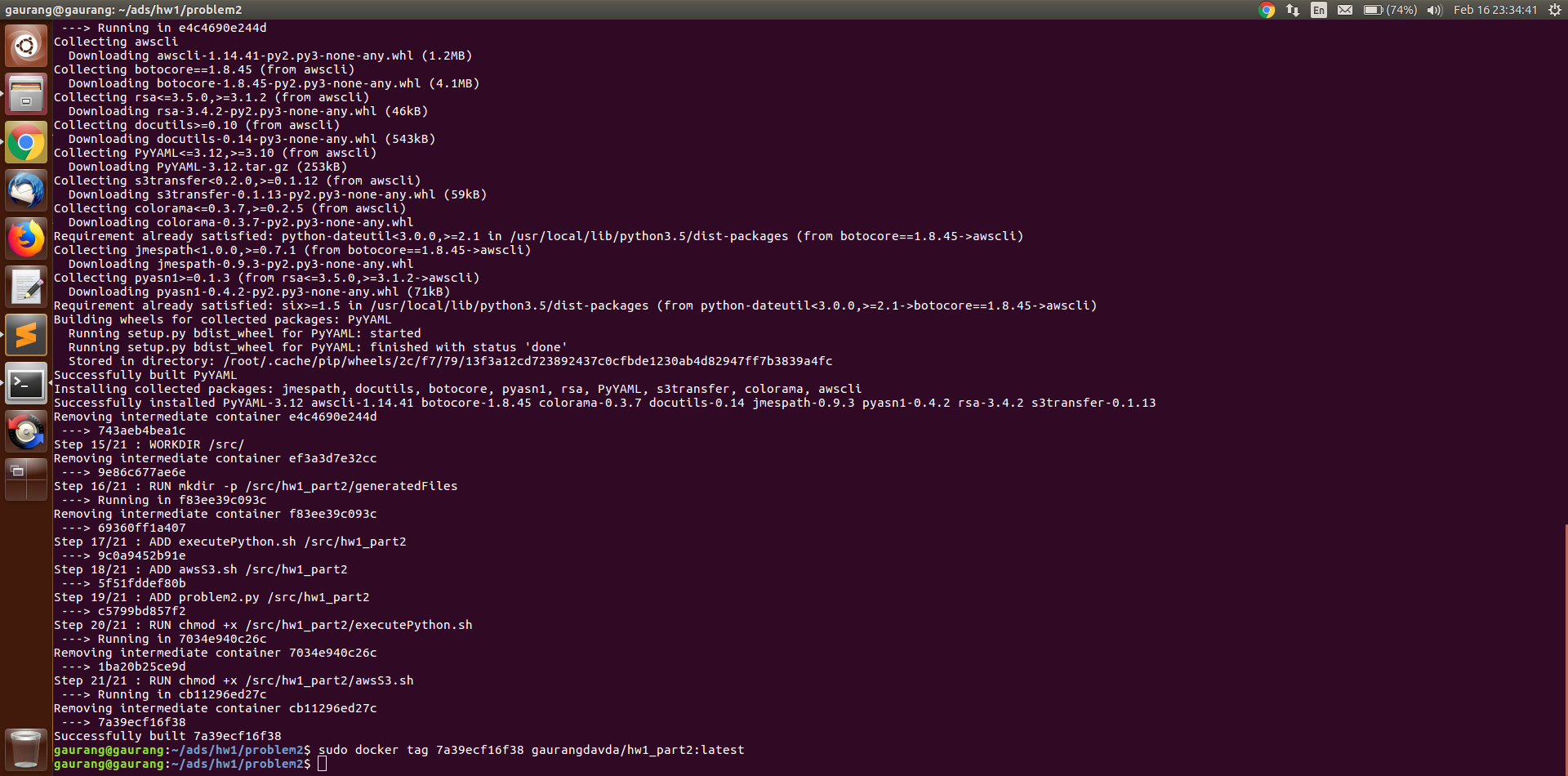
**STEP 1: Build the docker image from docker file**

sudo docker -f Dokerfile



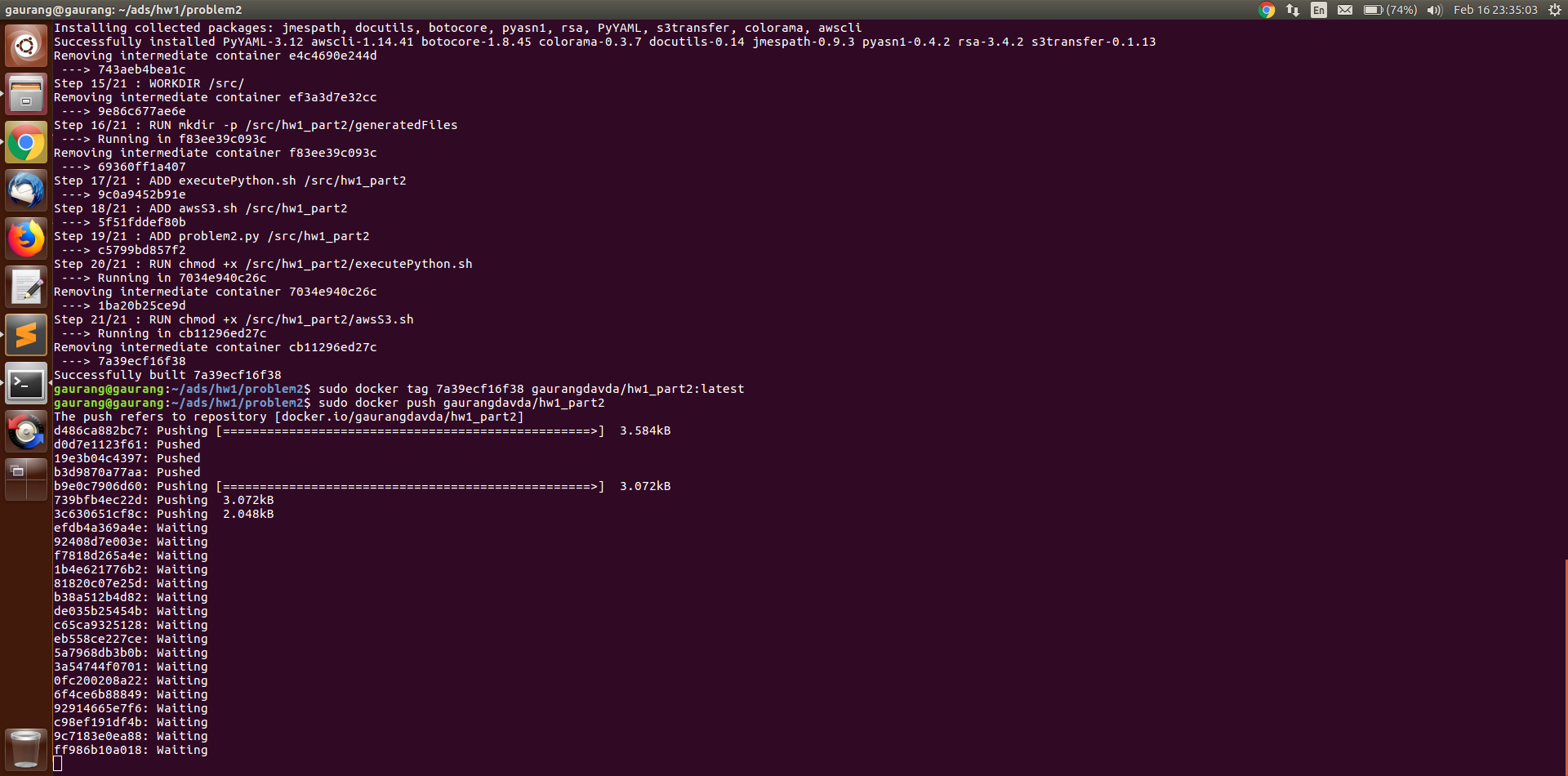
**STEP 2: Tag the build image to your repository name using**

sudo docker tag <tag of built file > <docker username >/<reponame>:<version>



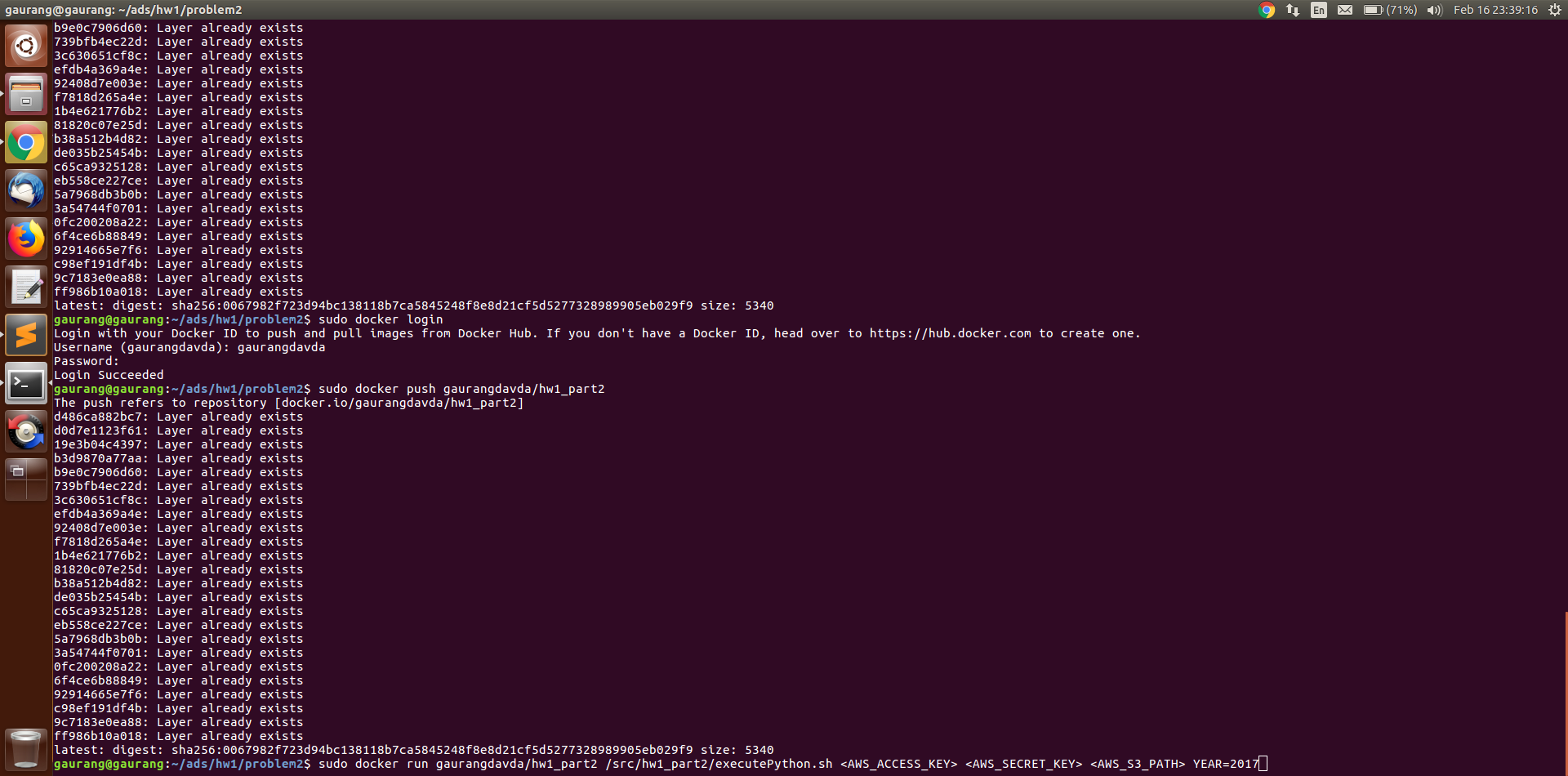
**STEP 3: Push the docker image from local to docker hub**

sudo docker push <docker username>/<reponame>



**STEP 4: Run the docker image using**

sudo docker run <docker username>/<reponame> <path of the file to be executed> <parameter of shell file>



**Section 4 – Summary**

In the analysis of EDGAR log data , we came across various difficulties while figuring out the possible data patterns from the dataset. Following are the studies we obtained during the process :

* There were multiple hits by same ip using different browsers
* At any particular instant, we observed different companies use same ip for different document request
* There is lot of data loss in 'Browser' column
* Analysing 1st day of first month gives very little pattern to understand data
* CIK column is not a 10 digit number as per "https://old.datahub.io/dataset/edgar"
* There were records with same value expect change in number that represented document request

There can be number of other patterns that will aid to detail the log data inconsistency. One assumption that we concluded by studying the 'ip' pattern, is that there is a probability that it indicates that the log hit is by either a normal human users or computer programs that acts in seconds speed. With this assumption, we can get new data pattern but it requires supporting models that distinguish a human user with a robot.