# 

# **ADVANCED DATA SCIENCE**

Assignment 1

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

Prof. Sri Krishnamurthy



Team Members:

Sayali Borse

Rishi Rajani

Komal Ambekar

# **Table of Contents:**

|  |  |
| --- | --- |
| **Sr. No.** | **Description** |
|  |  |
| 1 | Libraries |
| 2 | Logger |
| 3 | Problem 1: Data Wrangling Edgar Data set |
| 4 | Problem 2: Missing Data Analysis |

# **Libraries:**

1. Urllib: This is used for webscrapping
2. Urllib.request: This module is used for defining functions and classes for opening URLs
3. Zipfile: This module is used to manipulate zipfiles
4. Beautifulsoup: [Beautiful Soup](http://www.crummy.com/software/BeautifulSoup/) is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.
5. Csv: This is used to write and read the csv files
6. Logging: This module is used to track the logging details using the logger functions
7. Boto3: For handling Amazon Web Services S3
8. Pandas: This is used for data manipulation
9. Shutil: The **[shutil](https://docs.python.org/2/library/shutil.html" \l "module-shutil" \o "shutil: High-level file operations, including copying.)** module offers a number of high-level operations on files and collections of files. In particular, functions are provided which support file copying and removal.
10. Glob: The [**glob**](https://docs.python.org/2/library/glob.html#module-glob) module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order.
11. Os: This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see [**open()**](https://docs.python.org/2/library/functions.html#open), if you want to manipulate paths

# **Logger:**

Logging is a means of tracking events. An event is described by a descriptive message which can optionally contain variable data. We are generating log files at every stage of the code, we are generating timestamps, level names and customized error message.

We have use 4 levels of Logs:

1. DEBUG: It is used to return information to the log file that is created. Report events that occur during normal operation of a program
2. Error: Due to a more serious problem, the software has not been able to perform some function.
3. Info: Confirmation that things are working as expected.

# **Problem 1: Data Wrangling and EDA for Edgar Data set**

## **Tasks and Requirements:**

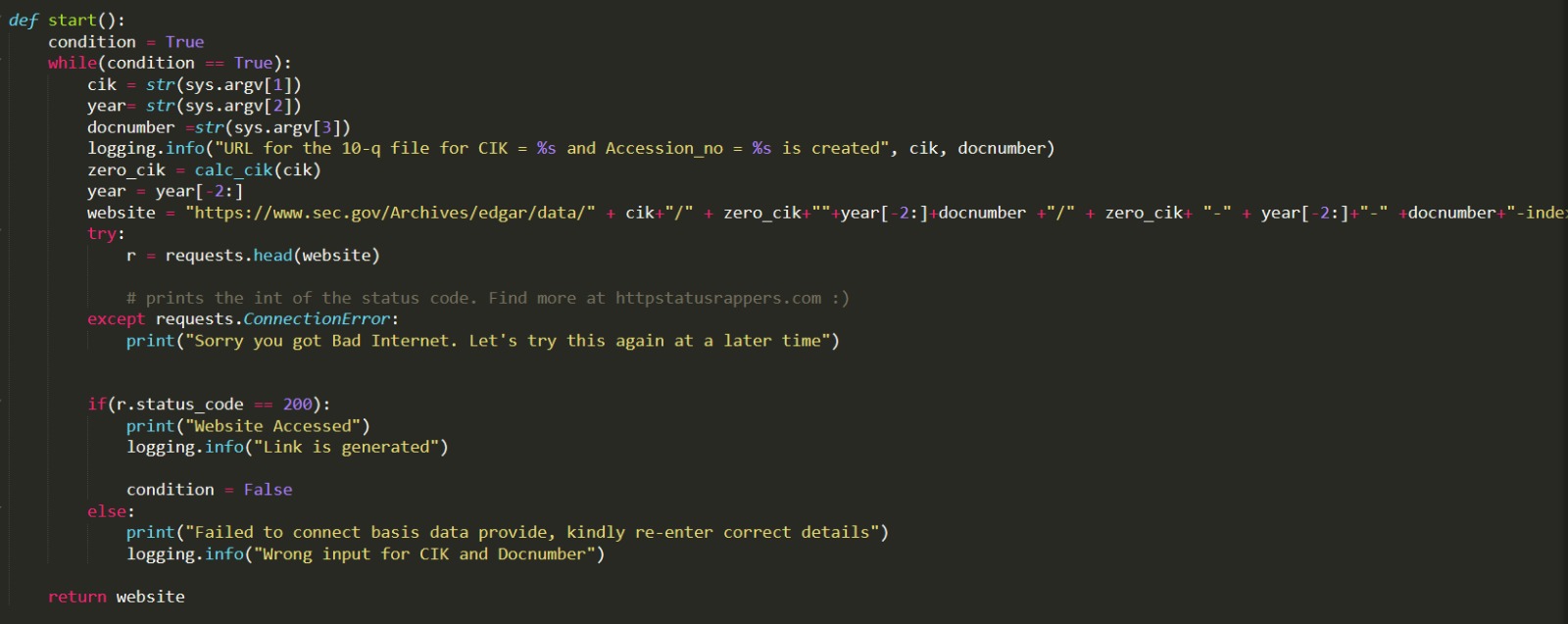
* The objective of this problem is to extract tables from 10Q fillings from <https://www.sec.gov/edgar/>
* Inputs given: CIK number, Accession code, Amazon access key and Amazon secret access key
* We are using urllib and urllib.request libraries to navigate and generate page specific URL.
* We are using BeautifulSoup libraries to handle html tags in Python and also to prettify the data content.
* Once URL is opened using the urllib.request.get() we have a foreach loop to run through all the <a> tags and <href> tags of the html page. If there is no 10-Q file the program exists.
* Once we get and open the 10-Q url link we extract the data by looking for table content which are part of the <div>tag.
* We clean the fetched data as it contains special characters like ‘$’ and ‘%’ values inside the table data. So we have to iterate through the tables and look for patterns.
* After data is cleaned every table is stored as individual csv file which is inside extracted\_csv folder.
* The program zips the folder and puts it inside Problem\_1.zip.
* We create a bucket and upload a zip file. The bucket name is always unique.

## **Screenshots:**

Eventually the entire code from our file comes down to running the below three functions. This code snippet is used to create the directory and scrape the website and generate the csv tables.



The below code snippet is used to take the CIK number and Accession number as input from the user which then generates the URL and accesses the website from which the tables should be extracted from 10Q fillings.



After the URL is generated and re-directed to the website. The tables from 10Q filings will be extracted and the csv files will be generated and zipped. 105 tables are extracted.



# **Problem 2: Missing Data Analysis:**

## **Tasks and Requirements:**

* Import all the libraries for the code and also initialize the log file as done in the problem 1.
* If the directories are not present create one to put the zip files and then clean the required directories.
* The input year must be between 2003 to 2018. If the year is not between this range the program will exit. There is no data for the first quarter of 2003.
* To the extract the csv, unzip all the downloaded log files.
* Load all the csv files into individual dataframe and combine into one csv file.
* Summary Metrics are as follows:
  + Compute mean size
  + Compute maximum used browser
  + Compute distinct count of ip per month i.e. per log file
  + Compute distinct count of cik per month i.e per log file
* Provide logging.info to all the variables and check for anomalies in the data and remove the row if found.
* Combine all dataframes and compute overall summary metric. Now export it to a csv.
* Create a bucket on S3 through the code and upload both the files.
* For each log file we have handled the missing value as the following:
  + The variables cik, ip, accession, data or time are most important for the log file in Edgar and we cannot dp correct analysis without these values.So skip the row if they consist null value.
  + Replacement for NaN according to the properties of variables.

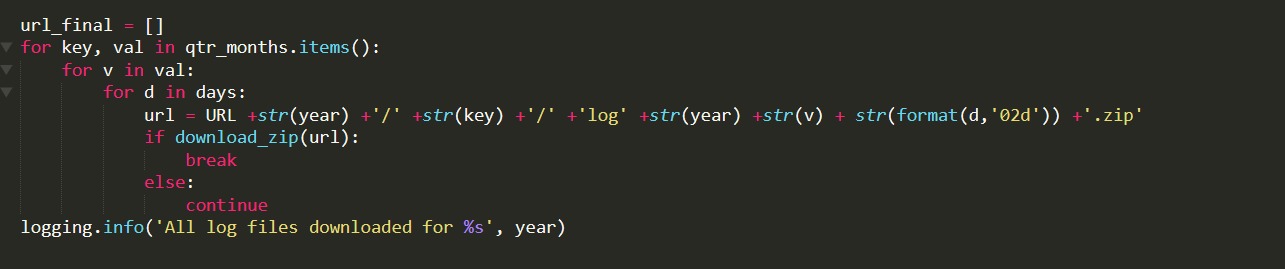
|  |  |  |
| --- | --- | --- |
| **VARIABLES** | **DATA** | **REPLACED WITH** |
| noagent | NaN | 1 |
| code | NaN | Most used code |
| browser | NaN | Most used browser |
| idx | NaN | Most used idx |
| norefer | NaN | 1 |
| find | NaN | Most used find |
| crawler | NaN | 0(assume it as empty) |
| extension | NaN | Most used extension |
| zone | NaN | Most used zone |
| size | NaN | Mean of the size |

## **Screenshots:**

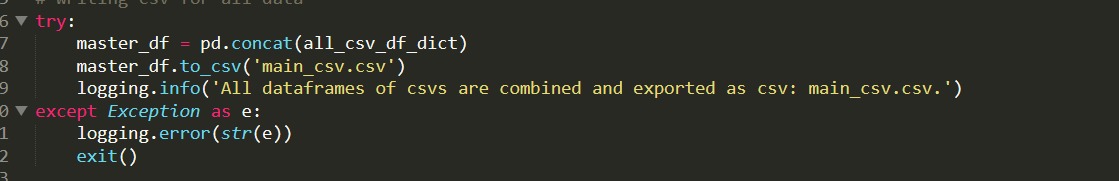
This try block checks if the zip files exist already in the folder. If the zipped folder exists then shutil.rmtree will remove the folder. If the zip folder do not exist then the downloaded zip folder will be unzipped. This is the directory cleanup.



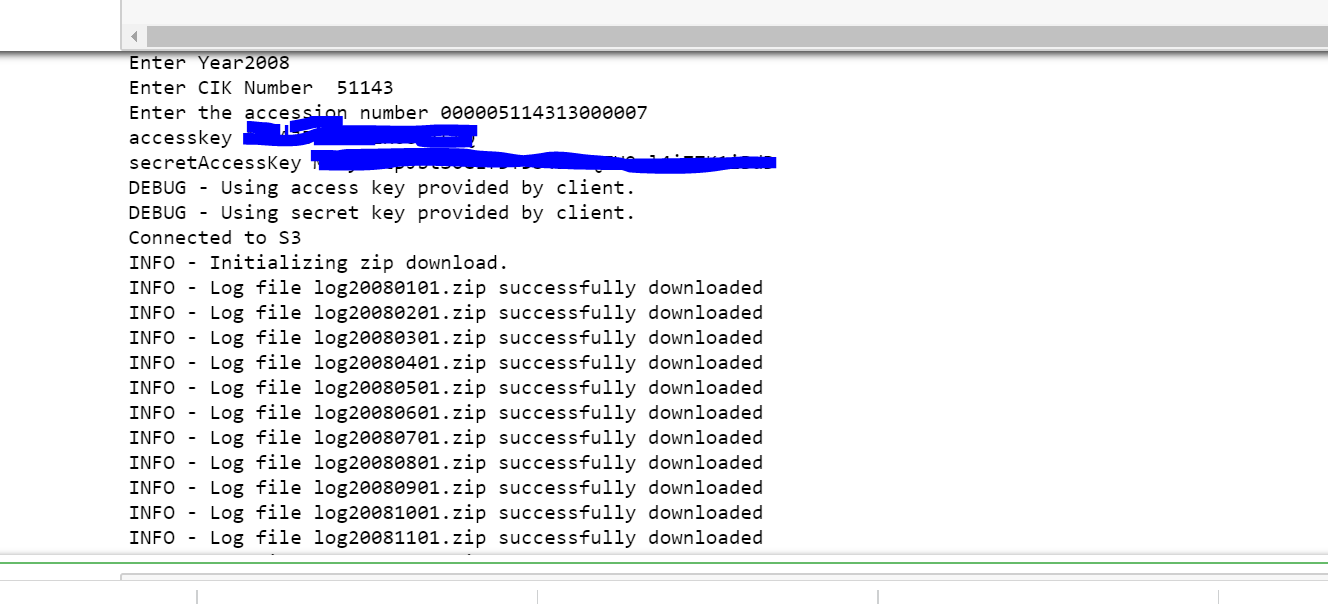
This code snippet is to download the log files from the URL when the user inputs the year for which he wants the log files. The log files downloaded will belong to the first day of every month.



All the downloaded and extracted csv files will be combined into one dataframe as main\_csv.csv. If the combining of the dataframe fails then it will throw an exception



**Output:**



Cleaning of Missing Data:

