# REPORT: Problem 1

# Data wrangling Edgar data from text files

## Part 1: Detailed steps of Data extraction, formatting and Upload

### 1> Using Config file to take user Input.

For this part we will ask user to create configuration file named: congif.ini

The format of the file will be:

```
[aws.data]

accessKey =

secretAccessKey =

inputLocation = us-east-1

cik = 51143

accessionNumber = 000005114313000007/0000051143-13-000007
```

### 2> Reading data from config file

Using folliwng code we can read the data from the file:

```
from configparser import ConfigParser

config = ConfigParser()


config_file = os.path.join(os.path.dirname(__file__), '/data/config.ini')

config.read(config_file)

default = config['aws.data']

accessKey = default['accessKey']

secretAccessKey = default['secretAccessKey']

inputLocation = default['inputLocation']

cik = default['cik']

accNum = default['accessionNumber']
```

### 3> Creating URL by given CIK and accession number:

```
url_start= "https://www.sec.gov/Archives/edgar/data/"
```

```
if not cik or not accNum:
    print('CIK and Accession number not given. Exiting the program')
else:
    print('CIK - %s' % (cik))
    print('Accession Number - %s' %( accNum))

url_final= url_start+cik.lstrip('0')+"/"+ accNum.replace('-','')+"/"+accNum+"-
index.html"

print('Final url is: %s'%(url_final))
logging.info("URL generated is: " + url_final)
```

## 4> Extraction 10q filing link from the given URL

```
#connect to a URL
website = urlopen(url_final)

#read html code
html = website.read()
soup=BeautifulSoup(html,"lxml")

#use soup to get all the links
url_10q=""

try:
    for link in soup.findAll('a'):
        print (link.get('href'))
        url_10qE= link.get('href')
        if url_10qE.endswith('10q.htm'):
            url_10q=url_10qE

    if url_10q is "":
        logging.info("Invalid URL!!!")
        print("Invalid URL!!!")
        exit()

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



print('10q url is: %s' %(url_10q))



url_10q= "https://www.sec.gov"+url_10q

print('Complete 10q url is: %s' %(url_10q))
```

5> Extracting tables from the given link and saving them in python:

```
page = urllib.request.urlopen(url_10q)
soup = BeautifulSoup(page, "lxml")

all_tables=soup.select("table")

my_tables=[]
for table in all_tables:
        my_tables.append([[td.text.replace("\n", " ").replace("\xa0"," ") for td in 
row.find_all("td")] for row in table.select("tr + tr")])
```

6> Formatting and extracting the table in .csv format

```
for tab in my_tables:
    if my_tables.index(tab) >=9 and my_tables.index(tab)<=109:
        with open(os.path.join('Extracted_csvs', str(my_tables.index(tab)-9) + 
'Tables.csv'), 'w') as f:
            writer = csv.writer(f)
            writer.writerows(tab)

# creating zip for every available file
def zipdir(path, ziph, refined_tables):
    for tab in my_tables:
        if my_tables.index(tab) >=9 and my_tables.index(tab)<=109:
            ziph.write(os.path.join('Extracted_csvs', str(my_tables.index(tab)-9) + 
'Tables.csv'))
```

```
        ziph.write(os.path.join('log_file.log'))
```

7> Zipping the folder and saving with log file.

```
zipf = zipfile.ZipFile('Log_File.zip', 'w', zipfile.ZIP_DEFLATED)

zipdir('/', zipf, my_tables)

zipf.close()
```

logging.info('csv and log file zipped')

8> Uploading the files in AWS S3 bucket.

```
  time_variable = time.time()

    timestamp_variable = datetime.datetime.fromtimestamp(time_variable)

    bucket_name = AWS_ACCESS_KEY_ID.lower() + str(timestamp_variable).replace(" ",
"").replace("-", "").replace(":","").replace( ".", "")

    bucket = conn.create_bucket(bucket_name, location=server_location)

    print("Bucket created")

    zipfile = 'Log_File.zip'

    print("Uploading %s to Amazon S3 bucket %s" %( zipfile, bucket_name))

    def percent_cb(complete, total):

        sys.stdout.write('.')

        sys.stdout.flush()


    k = Key(bucket)

    k.key = 'Log_File_1'

    k.set_contents_from_filename(zipfile,cb=percent_cb, num_cb=10)

    print("Zip File successfully uploaded to S3")
```

## Part 2: Handling the exception

### 1> When CIK or Accession Number if not found or wrong

```
if not cik or not accessionNumber:

    logging.warning(

        'CIK or AccessionNumber was not mentioned, assuming the values to be 51143 and
0000051143-13-000007 respectively. This is original data of Walmart')

    cik = '51143'

    accessionNumber = '0000051143-13-000007'

else:

    logging.info('CIK: %s and AccessionNumber: %s given'%( cik, accessionNumber))
```

## 2> Validating AWS keys:

```
if not accessKey or not secretAccessKey:
    logging.warning('Access Key and Secret Access Key not provided!!')
    print('Access Key and Secret Access Key not provided!!')
    exit()

AWS_ACCESS_KEY_ID = accessKey
AWS_SECRET_ACCESS_KEY = secretAccessKey

try:
    conn = boto.connect_s3(AWS_ACCESS_KEY_ID,
                           AWS_SECRET_ACCESS_KEY)

    print("Connected to S3")

except:
    logging.info("Amazon keys are invalid!!")
    print("Amazon keys are invalid!!")
    exit()
```