ADS ASSIGNMENT 1 WORKING WITH EDGAR DATASETS

REPORT PROBLEM 1 – DATA WRANGLING FROM EDGAR DATASET

PRIMARY TASKS INCLUDED IN 1st PROBLEM

1. Based on the user input, get CIK and accession number and generate a URL for the index.html file for that particular company.
2. After generating URL we search the 10-Q file which contains the tables which we need to scrape and generate a new URL for that 10-Q file of that company.
3. After generating the URL our main task is to scrape all the tables present on that URL having helpful information in them and then parsing them to structure the data and store it in CSV and store the zip file of all the tables in Amazon S3.
4. Now after this, we have automated this whole process using docker image so that we can replace any CIK and document accession number, insert their amazon keys and location and would be able to reuse the code.

TASK1:

```
#Creating URL using CIK and Accession number
cik = input("Enter a cik of a comapany:")
accNumber = input("Enter a documnet accession number:")

link2 = "http://www.sec.gov/Archives/edgar/data/" + cik + "/"+accNumber +"/"+ accNumber[:10]+ "-"+accNumber[10:12]+"-"+a
print(link2)
request = requests.get(link2)


if request.status_code == 200:
    print('Web site exists')
else:
    print('Web site does not exist',request.status_code)
```

```
Enter a cik of a comapany:51143
Enter a documnet accession number:000005114313000007
http://www.sec.gov/Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html
Web site exists
```

TASK2:

```
#Filtering , Pre-processing and Zipping the tables in 10-Q
http = urllib3.PoolManager()
response = http.request('GET', link1)
doc = BeautifulSoup(response.data)

tables = doc.findAll('table')
for table in tables:
    #find th = 'Type' in the selected table and its index.
    done = 'false'
    rows = table.findAll('tr')
    links = table.findAll('a')
    for row in rows:
        rows_th = row.findAll('th')
        rows_td = row.findAll('td')
        for row in rows_th:
            if row.text.lower() == 'type':
                type_ind = rows_th.index(row)

        col = [row.text for row in rows_td]
        type_col = col[type_ind::type_ind]

        for item in type_col:
            if str(item) == '10-Q':#Checking If 10-Q is found in any of the tables in the link
                aim = type_col.index(item)*type_ind
                for link in links:
                    if links.index(link) == aim:
                        final_link = "https://www.sec.gov"+ str(link).split('<a href="')[1].split('"')[0]
```

```
markup_type=markup_type)
```

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

TASK3:

```
table_list = pd.read_html(final_link)
print(final_link)
http = urllib3.PoolManager()
response = http.request('GET', final_link)
doc = BeautifulSoup(response.data)

tables = doc.findAll('table')

#Filtering the required tables based on Background colour
Required_Indexes = []

for j in range(0,len(tables)):
    rows = tables[j].findAll('td')
    for i in range(0,len(rows)):
        if 'background:' in str(rows[i]):
            Required_Indexes.append(j)
            break
for k in Required_Indexes:
    row_start_value=0
    for rows in range(0,len(table_list[k])-1):
        if table_list[k][rows:rows+1][0].notnull()[rows]:
            row_start_value=rows
            break
    s=[]
    for j in range(row_start_value,len(table_list[k])-1):
        s.append(table_list[k].iloc[j].dropna().tolist())
        for i in range(0,len(s)):
            if '$' in s[i]:
                s[i].remove('$')
    a= pd.DataFrame(s)
    #Saving the file to current woring directory
    cwd = os.getcwd()
    if not os.path.exists(cwd+"\\"+accNumber):
        os.makedirs(cwd+"\\"+accNumber)
    a.to_csv(cwd+"\\"+accNumber+"\\Table "+str(k+1)+ ".csv", sep=',', header= False,index= Fals
if __name__ == '__main__':
    cwd = os.getcwd()
    zipf = zipfile.ZipFile('%s.zip' %cik, 'w', zipfile.ZIP_DEFLATED)
    zipdir(accNumber, zipf)
    zipf.close()
```

Name	Size	Kind
51143.zip	51 KB	ZIP archive
000005114313000007	--	Folder

Initially the data is not well-organized and storing such data would lead to further problems in the process of analysis. The original data looks like this:

				Three Months Ended Sep		Nine Months Ended September 30,							
(Dollars in millions)			2013			2012			2013			2012	
Net income			$ 4041		$	3824		$	10299		$	10771	
Other comprehensive income/(loss), before tax:													
	Foreign currency translation adjustmer			382			501			-959			164
	Net changes related to available-for-sale securities:												
		Unrealized gains/(losses) arising during			3			11			0		13
		Reclassification of (gains)/losses to net			-5			-27			-5		-43
		Subsequent changes in previously impaired											
		securities arising during the period			1			-7			3		20
	Total net changes related to available-f			-1			-24			-1			-10
	Unrealized gains/(losses) on cash flow hedges:												
		Unrealized gains/(losses) arising during			-409			-54			-58		65
		Reclassification of (gains)/losses to net			-27			-112			-130		-246
	Total unrealized gains/(losses) on cash			-436			-165			-188			-181
	Retirement-related benefit plans:												
		Prior service costs/(credits)			0			0			33		0
		Net (losses)/gains arising during the pe			105			1			300		66
		Curtailments and settlements			0			-2			0		-1
		Amortization of prior service (credits)/(			-28			-37			-86		-112
		Amortization of net (gains)/losses			872			613			2623		1846
	Total retirement-related benefit plans			949			575			2869			1799
Other comprehensive income/(loss), b(			895			887			1721			1771	
Income tax (expense)/benefit related to items of													
		other comprehensive income			-91			-109			-933		-606
Other comprehensive income/(loss)			804			778			788			1165	
Total comprehensive inco			$ 4844		$	4601		$	11087		$	11936	
(Amounts may not add due to rounding.)													
(The accompanying notes are an integral part of the financial statements.)													

After performing cleaning and structuring operations the same data looks like this:

(DollarsÂ inÂ millionsÂ except perÂ shareÂ amounts)	2013	2012	2013	2012
Revenue:				
Services	14225	14626	42811	44279
Sales	8987	9642	27735	29424
Financing	509	479	1506	1500
Total revenue	23720	24747	72052	75203
Cost:				
Services	9098	9515	27950	29285
Sales	2975	3242	9108	10003
Financing	268	258	805	784
Total cost	12341	13016	37863	40072
Gross profit	11380	11732	34189	35131
Expense and other income:				
Selling, general and administrative	5255	5908	17512	17632
Research, development and engineering	1468	1534	4661	4722
Intellectual property and custom				
development income	-191	-303	-621	-847
Other (income) and expense	-62	-606	-214	-796
Interest expense	97	124	289	350
Total expense and other income	6567	6657	21627	21060
Income before income taxes	4812	5074	12562	14071
Provision for income taxes	772	1251	2263	3300
Net income	4041	3824	10299	10771
Earnings per share of common stock:				
Assuming dilution	3.68	3.33	9.27	9.27
Basic	3.7	3.36	9.35	9.38
Weighted-average number of common				
shares outstanding: (millions)				
Assuming dilution	1098.8	1149.3	1110.7	1161.8
Basic	1090.9	1137.2	1101.8	1148.4
Cash dividend per common share	0.95	0.85	2.75	2.45

```
inputLocation= input("Enter a appropriate location:")
accNumber='000005114313000007'
accessKey = "[redacted]"
secretAccessKey = '[redacted]'
locationlist = ('us-east-2','us-east-1','us-west-1','us-west-2','ap-south-1','ap-northeast-2','ap-northeast-3','ap-south
#locationlist = ('APNortheast','APSoutheast','ApSoutheast2','CNNorth1','EUCentral1','EU','SAEast','USWest','USWest2')

loc_link = ''

if inputLocation in locationlist:

    loc_link = 'boto.s3.connection.Location.' + inputLocation

#print(loc_link)
#try:
#conn = boto.connect_s3(AWS_ACCESS_KEY_ID,AWS_SECRET_ACCESS_KEY)
    conn = boto.s3.connect_to_region(inputLocation, aws_access_key_id = accessKey,
    aws_secret_access_key = secretAccessKey)

    ts = time.time()
    st = datetime.datetime.fromtimestamp(ts)
    bucket_name = accNumber.lower()+ "-"+str(st).replace(" ", "").replace("-", "").replace(":","").replace(".","")
    bucket = conn.create_bucket(bucket_name, location=inputLocation)
    print("bucket created")
    zipfile = zipf.filename
    print ("Uploading %s to Amazon S3 bucket %s", zipfile, bucket_name)

    #bucket = conn.get_bucket(bucket_name)
    #k = bucket.new_key(zipfile)
    k = Key(bucket)
    k.key = zipf.filename

    def percent_cb(complete, total):
            sys.stdout.write('.')
            sys.stdout.flush()

    k.set_contents_from_filename(zipfile)

    print('zip file uploaded')
    k.set_contents_from_filename(zipfile)
```

```
8='), ('x-amz-request-id', '6BDC42AD9FB5D1E3'), ('Date', 'Sat, 17 Feb 2018 00:11:04 GMT'), ('ETag', '"010a48e8ce65c77
219841adbe31392ec"'), ('Content-Length', '0'), ('Server', 'AmazonS3')]
zip file uploaded
```

TASK 4:

```
JAI@JAI-PC MINGW64 ~/Desktop/Northeastern/ads/Assignment 1
$ docker run jaisoni/doc62 python Problem_1.py year=2013 cik=51143 accNumber=000005114313000007 accessKey=[redacted] secretKey=[redacted]
h1m location=us-west-2
```

```
CIK= 51143
Accession Number= 000005114313000007
Access Key= [redacted]
Secret Access Key= [redacted]
Location= us-west-2
DEBUG - Starting new HTTP connection (1): www.sec.gov
DEBUG - http://www.sec.gov:80 "GET /Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html HTTP/1.1" 301 269
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html HTTP/1.1" 200 2860
Web site exists
DEBUG - Starting new HTTP connection (1): www.sec.gov
DEBUG - http://www.sec.gov:80 "GET /Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html HTTP/1.1" 301 269
DEBUG - Incremented Retry for (url='http://www.sec.gov/Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html'): Retry(total=2, connect=None, read=None,
redirect=None, status=None)
INFO - Redirecting http://www.sec.gov/Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html -> https://www.sec.gov/Archives/edgar/data/51143/00000511431
3000007/0000051143-13-000007-index.html
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /Archives/edgar/data/51143/000005114313000007/0000051143-13-000007-index.html HTTP/1.1" 200 11022
https://www.sec.gov/Archives/edgar/data/51143/000005114313000007/ibm13q3_10q.htm
DEBUG - Starting new HTTPS connection (1): www.sec.gov
DEBUG - https://www.sec.gov:443 "GET /Archives/edgar/data/51143/000005114313000007/ibm13q3_10q.htm HTTP/1.1" 200 None
DEBUG - Using access key provided by client.
DEBUG - Using secret key provided by client.
DEBUG - path=/
DEBUG - auth_path=/000005114313000007-20180217013759631698/
DEBUG - Method: PUT
DEBUG - Path: /
DEBUG - Data: <CreateBucketConfiguration><LocationConstraint>us-west-2</LocationConstraint></CreateBucketConfiguration>
DEBUG - Headers: {}
DEBUG - Host: 000005114313000007-20180217013759631698.s3-us-west-2.amazonaws.com:443
DEBUG - Port: 443
DEBUG - Params: {}
DEBUG - establishing HTTPS connection: host=000005114313000007-20180217013759631698.s3-us-west-2.amazonaws.com, kwargs={'timeout': 70, 'port': 443}
DEBUG - Token: None
DEBUG - StringToSign:
PUT
```

```
/usr/local/lib/python3.6/site-packages/urllib3/connectionpool.py:858: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verification is strong
ly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
/usr/local/lib/python3.6/site-packages/bs4/__init__.py:181: UserWarning: No parser was explicitly specified, so I'm using the best available HTML parser for this system ("lxml
"). This usually isn't a problem, but if you run this code on another system, or in a different virtual environment, it may use a different parser and behave differently.

The code that caused this warning is on line 123 of the file Problem_1.py. To get rid of this warning, change code that looks like this:

 BeautifulSoup(YOUR_MARKUP})

to this:

 BeautifulSoup(YOUR_MARKUP, "lxml")

  markup_type=markup_type))
/usr/local/lib/python3.6/site-packages/urllib3/connectionpool.py:858: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certificate verification is strong
ly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning)
/usr/local/lib/python3.6/site-packages/bs4/__init__.py:181: UserWarning: No parser was explicitly specified, so I'm using the best available HTML parser for this system ("lxml
"). This usually isn't a problem, but if you run this code on another system, or in a different virtual environment, it may use a different parser and behave differently.

The code that caused this warning is on line 151 of the file Problem_1.py. To get rid of this warning, change code that looks like this:

 BeautifulSoup(YOUR_MARKUP})

to this:

 BeautifulSoup(YOUR_MARKUP, "lxml")

  markup_type=markup_type))
Connected to S3
```

PROBLEM 2- MISSING VALUE ANALYSIS

INFORMATION ABOUT DATA AND PRIMARY TASKS INCLUDED IN 2nd PROBLEM

1. There are total of 15 fields in each of the CSV file of Log file dataset.
2. Total nulls are calculated for each of the CSV file.
3. Some of the fields have definite scope and if the data for that field is out of the scope or not null then the data is incorrect for that field.
4. There are some important fields on which the whole tuple depends for the analysis and those fields are: 'CIK', 'Accession Number', 'Date'. If these fields are NULL we have dropped the rows as it creates no meaning to analyze those records.

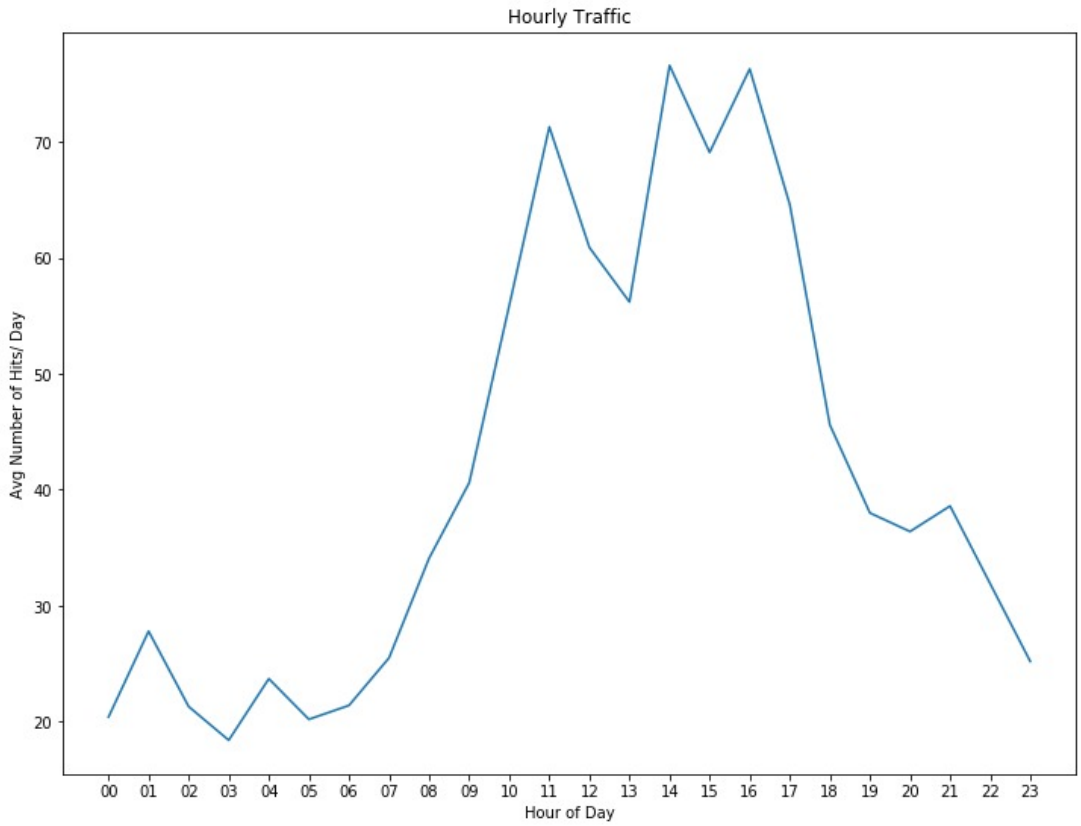
We used docker to automate this process to be applicable for any year and carry out missing value analysis on it and then storing it on Amazon S3 Bucket.

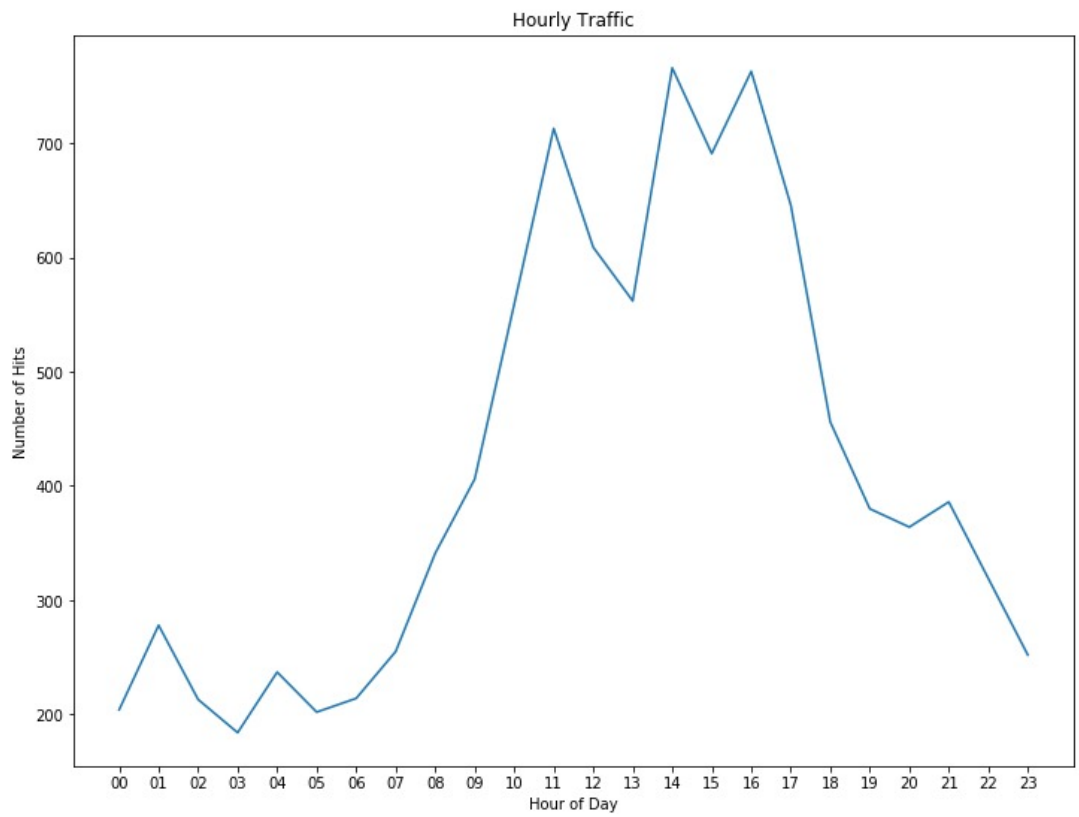
Some of the Code snippets and Visuals of the Missing Value Analysis are as follows:

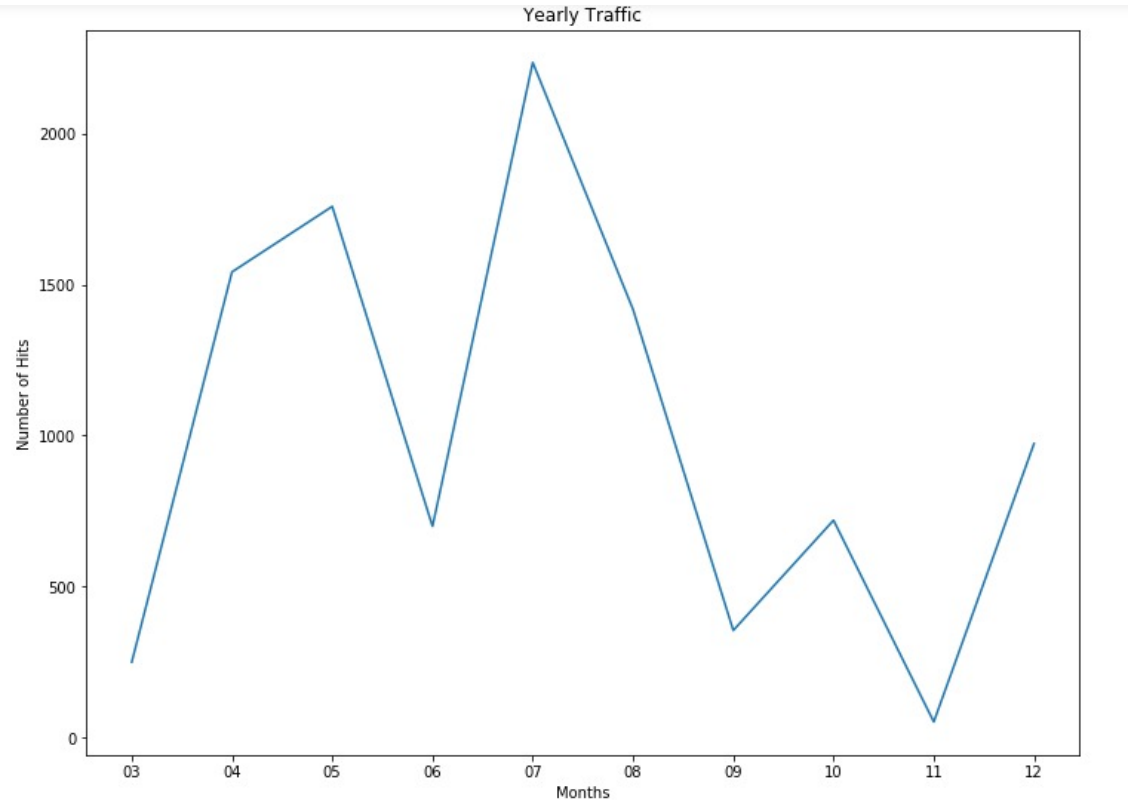
```
try:
    for key, val in all_csv_df_dict.items():
        df = all_csv_df_dict[key]
        #detecting null values
        null_count = df.isnull().sum()
        logging.info('Count of Null values for %s in all the variables:\n%s ', key, null_count)

        #remove rows which have no date, time, cik or accession

        df.dropna(subset=['cik'])
        df.dropna(subset=['accession'])
        df.dropna(subset=['date'])
        df.dropna(subset=['time'])

        # variable idx should be either 0 or 1
        incorrect_idx = (~df['idx'].isin([0.0,1.0])).sum()
        logging.info('There are %s idx which are not 0 or 1 in the log file %s', incorrect_idx, key)

        # variable norefer should be either 0 or 1
        incorrect_norefer = (~df['norefer'].isin([0.0,1.0])).sum()
        logging.info('There are %s norefer which are not 0 or 1 in the log file %s', incorrect_norefer, key)

        # variable noagent should be either 0 or 1
        incorrect_noagent = (~df['noagent'].isin([0.0,1.0])).sum()
        logging.info('There are %s noagent which are not 0 or 1 in the log file %s', incorrect_noagent, key)

        # variable date should be same as file name
        incorrect_size = (df['size'] <= 0.0).sum()
        logging.info('There are %s rows with size less than 0 in the log file %s', incorrect_size, key)

        # variable find should be either 0 or 10
        incorrect_find = (~df['find'].isin([0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0])).sum()
        logging.info('There are %s find which are not 0 throigh 10 in the log file %s', incorrect_find, key)

        # variable crawler should be either 0 or 1
        incorrect_crawler = (~df['crawler'].isin([0.0,1.0])).sum()
        logging.info('There are %s crawler which are not 0 or 1 in the log file %s', incorrect_crawler, key)
```

```
        #replace nan with the most used browser in data.
        max_browser = pd.DataFrame(df.groupby('browser').size().rename('cnt')).idxmax()[0]
        df['browser'] = df['browser'].fillna(max_browser)

        # replace nan idx with max idx
        max_idx = pd.DataFrame(df.groupby('idx').size().rename('cnt')).idxmax()[0]
        df['idx'] = df['idx'].fillna(max_idx)

        # replace nan code with max code
        max_code = pd.DataFrame(df.groupby('code').size().rename('cnt')).idxmax()[0]
        df['code'] = df['code'].fillna(max_code)

        # replace nan norefer with zero
        df['norefer'] = df['norefer'].fillna('1')

        # replace nan noagent with zero
        df['noagent'] = df['noagent'].fillna('1')

        # replace nan find with max find
        max_find = pd.DataFrame(df.groupby('find').size().rename('cnt')).idxmax()[0]
        df['find'] = df['find'].fillna(max_find)

        # replace nan crawler with zero
        df['crawler'] = df['crawler'].fillna('0')

        # replace nan extention with max extention
        max_extention = pd.DataFrame(df.groupby('extention').size().rename('cnt')).idxmax()[0]
        df['extention'] = df['extention'].fillna(max_extention)

        # replace nan extention with max extention
        max_zone = pd.DataFrame(df.groupby('zone').size().rename('cnt')).idxmax()[0]
        df['zone'] = df['zone'].fillna(max_zone)

        #Recovering extention with the accession for the incorrect extention
        for i in range(0,len(df)-1):
            temp = df[['accession','extention']].iloc[i][1]
            series = temp.split(".")
            if series[0] == "":
                df['extention'].iloc[i] = df['accession'].iloc[i]+testdata['extention'].iloc[i]


    logging.info('Rows removed where date, time, cik or accession were null.')
    logging.info('Recovered ip with the max ip used by that cik.')
    logging.info('NaN values in browser replaced with maximum count browser.')
    logging.info('NaN values in idx replaced with maximum count idx.')
    logging.info('NaN values in code replaced with maximum count code.')
    logging.info('NaN values in norefer replaced with 0.')
    logging.info('NaN values in noagent replaced with 0.')
    logging.info('NaN values in find replaced with maximum count find.')
    logging.info('NaN values in crawler replaced with 0.')
    logging.info('NaN values in extension replaced with maximum count extension.')
    logging.info('NaN values in zone replaced with maximum count zone.')
    logging.info('NaN values in size replaced with mean value of size.')

except Exception as e:
    logging.error(str(e))
    exit()
```

We have visualized the data to understand the patterns and trends present in it and gain helpful insights. Some of the visuals of analysis are as follows:


Hourly Traffic
Avg Number of Hits/ Day
70
60
50
40
30
20
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of Day


Hourly Traffic
Number of Hits
700
600
500
400
300
200
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Hour of Day



https://app.powerbi.com/groups/me/reports/28665e95-8e29-4c13-82ad-72b5053f5633/ReportSection

We have automated this whole process of carrying out missing value analysis on a particular CSV and storing it to the amazon s3 bucket using docker. The screenshot for the successful implementation of docker is as follows: