Data loading, Storage and File Formats

UNIT III

Data Loading, Storage, and File Formats : Reading and Writing Data in Text Format, Reading Text Files in Pieces, Writing Data Out to Text Format, Manually Working with Delimited Formats, JSON Data, XML and HTML: Web Scraping, Binary Data Formats, Using HDF5 Format, Reading Microsoft Excel Files, Interacting with Databases, Storing and Loading Data in MongoDB

Reading and Writing of data in text formats:

Python become a beloved language for text and files because of its simple syntax for interaction with files, data structure and it convenient features like list,tuples, series, data frames, packing and unpacking.

Here we are going to taking the support of CSV, TEXT, EXCEL, JSON files to directly creating the data set instead of creating the data sets by using data structures.

CSV FILE: CSV file is delimited text file that uses commas to separate the values

- Each line of the file is data record, each data record contains one or more fields separated by commas.
- CSV stores the data in tabular format as a plane text ,various software programs and applications can be import or export from the CSV file.
- To directly import the CSV file into python environment, we need to use a method called read_csv(). Similarly we can also use a method called read_table() method.
- Firstly we have to create a MS EXCEL file in csv file format to create a data set.
  import pandas as pd
  data = pd.read_csv(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book1.csv')
  print(data)
  output:

	CSE	IT	CSD
0	501	1201	4401
1	502	1202	4402
2	503	1203	4403

Data loading, Storage and File Formats

- csv file format will never allow any beautifications to the current working file.
- Now we will work with read_table.

```
import pandas as pd

data = pd.read_table(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book1.csv')

print(data)
```

output:

```
   Beverages,cost
0         Pepsi,10
1      Coco cola,12
2     Thumbsup,13
3          Maaza,14
```

- we have any attribute called sep which is used to remove the comma in the read_table method.

```
import pandas as pd

data = pd.read_table(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book1.csv',sep = ',')

print(data)
```

output:

```
  Beverages  cost
```

Data loading, Storage and File Formats

	Beverages	cost
0	Pepsi	10
1	Coco cola	12
2	Thumbsup	13
3	Maaza	14

- we also have an attribute called header which is gives the index to the column names. Here we are going to assign the header with None to provide default (or) integers.

```
import pandas as pd

data = pd.read_csv(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book1.csv',header = None)

print(data)
```

output:

	0	1
0	Beverages	cost
1	Pepsi	10
2	Coco cola	12
3	Thumbsup	13
4	Maaza	14

Data loading, Storage and File Formats

- We are having a special attribute to change the column name which is called as names. We also have another to remove rows which is called skiprows and we have to assign some value to remove a specified number of rows.

```
import pandas as pd

data = pd.read_csv(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book1.csv',names = ["beveragers","price"],skiprows = 1)

print(data)
```

output:

	beveragers	price
0	Pepsi	10
1	Coco cola	12
2	Thumbsup	13
3	Maaza	14

Major advantages of the use of reading and writing different formats of files (CSV,JSON,TEXT,EXCEL…):

1. User friendly interface(working directly with raw data).
2. Easy retrievals.
3. Permanently stored.
4. Easy appending.

Reading text file in pieces:

- While processing a large file and to figure out the right set of arguments (or) rows, we want read a small pieces of the file through chunks. We have attribute like nrows and chunksize to reduce the large file into smaller chunks for easy processing of the data.

Data loading, Storage and File Formats

- Now let us see how the nrows will work.

```
import pandas as pd

data = pd.read_table(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book1.csv',nrows = 2)

print(data)
```

output:

	Beverages	cost
0	Pepsi	10
1	Coco cola	12

- Now let us try the chunksize attribute. Return TextFileReader object for iteration. See the IO Tools docs for more information on iterator and chunksize.

```
import pandas as pd

data = pd.read_table(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book1.csv',chunksize = 2,sep = ',')

print(data)
```

output:

```
<pandas.io.parsers.readers.TextFileReader object at 0x000001C9B255ACD0>
```

Writing data into the file (or) text format:

- Data can also be exported to delimited format. This can be done by using to_csv() method.
- Here we need to create a empty csv file in the current working directory, now let us consider the empty csv file as Book2.csv.

Data loading, Storage and File Formats

```
import pandas as pd
data =
pd.DataFrame({'a':[2020,2021,2022,2023,2024],'b':['JAN','FEB','MAR','APR','MAY'],'c':['M
ON','TUE','WED','THU','FRI']})
data.to_csv('C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book2.csv')
```

- After executing the code a new csv file is created in the current directory.
- To see the output for the above code go to current directory and open the Book2.csv.

```
import pandas as pd
import sys
data=pd.read_csv(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book3.csv')
print(data)
data.to_csv(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book5.csv')
k = data.to_csv(sys.stdout)
print(k)
output:
```

```
   a   b    c
0  1  11  111
1  2  22  222
2  3  33  333
,a,b,c
0,1,11,111
1,2,22,222
2,3,33,333
None
```

```
k1 = data.to_csv(sys.stdout,sep = '|')

print(k1)

output:
```

```
|a|b|c
0|1|11|111
1|2|22|222
2|3|33|333
None
```

Modification of the table:

- Now let us modify the existing table by opening Book3.csv and adding another column with name ‘d’ and filling the it with data along with NaN.

```
import pandas as pd
data=pd.read_csv(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book3.csv')
print(data)
```

output:

```
   a   b    c      d
0  1  11  111    NaN
1  2  22  222  2222.0
2  3  33  333    NaN
```

- Now we can manipulate the table by replacing NaN with NULL which is done by using na_rep attribute which is available in to_csv() method.

```
data.to_csv(sys.stdout,na_rep = "NULL")
print(data)
```

output:

```
   a   b    c      d
0  1  11  111    NaN
1  2  22  222  2222.0
2  3  33  333    NaN
```

- We can also change the NULL values with our required values.

```
data.to_csv(sys.stdout,na_rep = 77)

print(data)
```

output:

```
   a   b    c      d
0  1  11  111     77
1  2  22  222  2222.0
2  3  33  333     77
```

- We can also alter the table by removing the indexing and header by using index and header attributes which are available in to_csv() method. But to remove the index and header only by assigning the attribute with ‘False’.

```
data.to_csv(sys.stdout, sep='|', na_rep=77, index=False, header=False)
```

output:

```
1|11|111|77
2|22|222|2222.0
3|33|333|3333.0
```

Data loading, Storage and File Formats

Manually working delimited format:

- Normally we can also work manually to read (or) write the files without using the in built methods like read_csv(),read_table(),to_csv(). Now in this case we can use CSV module to work manually with delimited files.
- To work with the CSV module first we need to import the module. Now let us see how we should work manually with delimited files.

```
import pandas as pd

import csv

f = open(r'C:\Users\hp\Desktop\engineering textbooks\FDS NOTES\Book5.csv')

out = csv.reader(f)

for i in out:

    print(i)
```

output:

```
['', 'a', 'b', 'c']
['0', '1', '11', '111']
['1', '2', '22', '222']
['2', '3', '33', '333']
```

```
out=csv.reader(f, delimiter='|')
output:
['a,b,c,d']
['1,11,111,NaN']
['2,22,222,2222']
['3,33,333,3333']

line = list(csv.reader(open(r'C:\Users\hp\Desktop\engineering textbooks\FDS
NOTES\Book5.csv')))
print(line)
output:
[['', 'a', 'b', 'c'], ['0', '1', '11', '111'], ['1', '2', '22', '222'], ['2', '3', '33', '333']]
```

Data loading, Storage and File Formats

JSON[JAVA SCRIPT OBJECT NOTATION]:

- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format.
- It is easy for humans to read and write.
- It is easy for machines to parse and generate.
- JSON requires less memory to store the data.
- It is based on a subset of the JavaScript Programming Language Standard.
- The most commonly used message format for application irrespectively to language and interpretability.
- JSON is very similar to python dict object and it contains a group of key-value pairs.
- It has become one of the standard formats for sending data by HTTP request between web browsers and other applications.

JSON Module:

- Python has a built-in package called **JSON**, which can be used to work with JSON data. It's done by using the JSON module.
- JSON module is to convert the JSON form to python dict object and python dict object to JSON form.

Serialization:

- The process of converting an object from python supported form to either file supported (or) network supported form.
- It is also defined as
- For serialization purpose we use dump() and dumps() methods which are available in json module.

dump(): The dump() method is used when the Python objects have to be stored in a file.
dumps(): The dumps() is used when the objects are required to be in string format and is used for parsing, printing, etc, .

```
import json
sdetail = { 'sname':'kumar',
        'age':21,
        'per':94.67,
        'ismarried':False,
        'have':None}
injson = json.dumps(sdetail)
print(injson)
output:
{"sname": "kumar", "age": 21, "per": 94.67, "ismarried": false, "have": null}

import json
sdetail = { 'sname':'kumar',
        'age':21,
```

```
        'per':94.67,
        'ismarried':False,
        'have':None}
injson = json.dumps(sdetail,indent = 4)
print(injson)
output:
{
   "sname": "kumar",
   "age": 21,
   "per": 94.67,
   "ismarried": false,
   "have": null
}
```

- To create a JSON file with python dict object we use dump() method which is available in json module.

```
import json
sdetail = { 'sname':'kumar',
            'age':21,
            'per':94.67,
            'ismarried':False,
            'have':None}
injson = json.dumps(sdetail,indent = 4)
with open('studentinfo.json','w') as f:
   json.dump(sdetail,f,indent = 4)
```

- Now it creates a file (studentinfo.json) in the current working directory. To access the file we need to go to current directory and the file will directs to notepad.

Deserialization:

- The process of converting an object from either file support form (or) network supported form to python supported form.
- It is also defined as the process of decoding the data that is in JSON format into native data type.
- For deserialization purpose we use loads() and load() which are available in json module.

loads(): The loads() method is used to convert json string to python dict object.
load(): The load() method is used when we need to read a JSON format from a file and convert into python dict object.

```
import json
jobj='''{"sname":"kumar",
        "age":20,
        "per":80.2,
        "ismarried":false,
        "havegf":null}'''
```

Data loading, Storage and File Formats

```
inpyth=json.loads(jobj)
print(inpyth)
print('Student name::', inpyth['sname'])
print('Student age::', inpyth['age'])
print('Student percentage::', inpyth['per'])
print('Is married::', inpyth['ismarried'])
print('have Girl Friend::', inpyth['havegf'])
output:
{'sname': 'Khaleel', 'age': 20, 'per': 80.2, 'ismarried': False, 'havegf': None}
Student name:: Khaleel
Student age:: 20
Student percentage:: 80.2
Is married:: False
have Girl Friend:: None
```

Web Scraping: Web scraping is the process of gathering information from the Internet that means huge data.

XML Parser: XML parser is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XML documents.

- There are two types of XML parser-
    1. DOM –Document Object Model
    2. SAX-Simple API for XML

DOM: A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser:

- A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.
- DOM Parser has a tree based structure.
- It supports both read and write operations and the API is very simple to use.
- It is preferred when random access to widely separated parts of a document is required.

SAX: A SAX Parser implements SAX API. This API is an event based API and less intuitive.

Features of SAX Parser:

- It does not create any internal structure.
- Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.
- It is an event based parser, it works like an event handler in Java.
- It is simple and memory efficient.

Data loading, Storage and File Formats

- It is very fast and works for huge documents.

BeautifulSoup library:

- It is a python library which is used to perform web scraping and pulling the data out of HTML and XML files.
- It works with your favorite parser to provide idiomatic ways to navigating, searching and modifying the parser tree.
- It commonly saves programmers hours or days of work.

request module:

- The request module allows you to send HTTP requests using python.
- It is used to abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application.
- Now let us see how we perform HTML web scraping.

```
from bs4 import BeautifulSoup
import requests
url="https://kitsguntur.ac.in/site/kits.php"
xml=requests.get(url)
xml.content
k=BeautifulSoup(xml.content)
print(k)
```

output:

It displays all the tags of the above mentioned HTML web page.

- Let us also see how to work with XML web scraping.

```
from bs4 import BeautifulSoup
import requests
url="https://www.w3schools.com/xml/note.xml"
xml=requests.get(url)
xml.content
k=BeautifulSoup(xml.content)
t=k.find('note')
print(t.text)
```

output:

```
Tove
Jani
Reminder
  Don't forget me this weekend!
```

Data loading, Storage and File Formats

Binary data format:

- We have many ways to store the data in binary format but one of the easiest way is using the python in built pickle serialization.
- Using the pickle serialization the data is stored efficiently in the binary format.
- Generally pandas has conventional object like load(),loads(),dump(),dumps() to read and write the data from the disk.

Pickle:

- The pickle module implements binary protocols for serializing and de-serializing a Python object structure.
- Pickling is the process whereby a Python object hierarchy is converted into a byte stream.
- Python objects like list, dict are converted into "character stream".
- The ideal for character stream is, it contains all the information that is necessary to reconstruct the object in another python's script.

HDF5 format:

- HDF5 file stands for Hierarchical Data Format 5.
- It is an open-source file which comes in handy to store large amount of data.
- It stores data in a hierarchical structure within a single file. So if we want to quickly access a particular part of the file rather than the whole file
- One important feature is that it can attach metadata to every data in the file thus provides powerful searching and accessing.
- There are many numbers of tools that facilitate efficient reading and writing a large amount data in binary format in disk.
- There two interfaces to the HDF5 library in python is PyTable and h5Py.
- In the above two interface each take a different approaches to solve the problems.

h5Py:

- The h5py package is a Pythonic interface to the HDF5 binary data format.
- h5py provides easy-to-use high level interface, which allows you to store huge amounts of numerical data.
- Easily manipulate that data from NumPy. H5py uses straightforward NumPy and Python metaphors, like dictionary and NumPy array syntax.
- H5py provide direct and high level interface to the HDF5 API.

PyTable:

- PyTables is a Python package for storing and querying large tabular datasets in an efficient way.
- PyTables is built on top of the HDF5 library and the NumPy and numexpr packages.
- These provide the foundations for very compact storage and high performance data management.
- PyTables abstract main of the details of HDF5 to provide multiple flexible data contains table indexing, support for out of core computations.

Reading Microsoft Excel Files:

Data loading, Storage and File Formats

Interacting with database (MongoDB):

- MongoDB tutorial provides basic and advanced concepts of SQL.
- It is the most popular and trending database now-a-days.
- Even though we are having a lot of database software likeoracle,Mysql,DB2,Sybase, IBMDB,….. from different database venders like Oracle Cooperation, IBM, Microsoft etc..
- MongoDB is derived from the word "HUMONGOUS" which means Extremely very large.
- MongoDB is an cloud based database.
- MongoDB is not only the database but it is also the company i.e. vender of MongoDB.
- MongoDB also provides a lot of software programs.

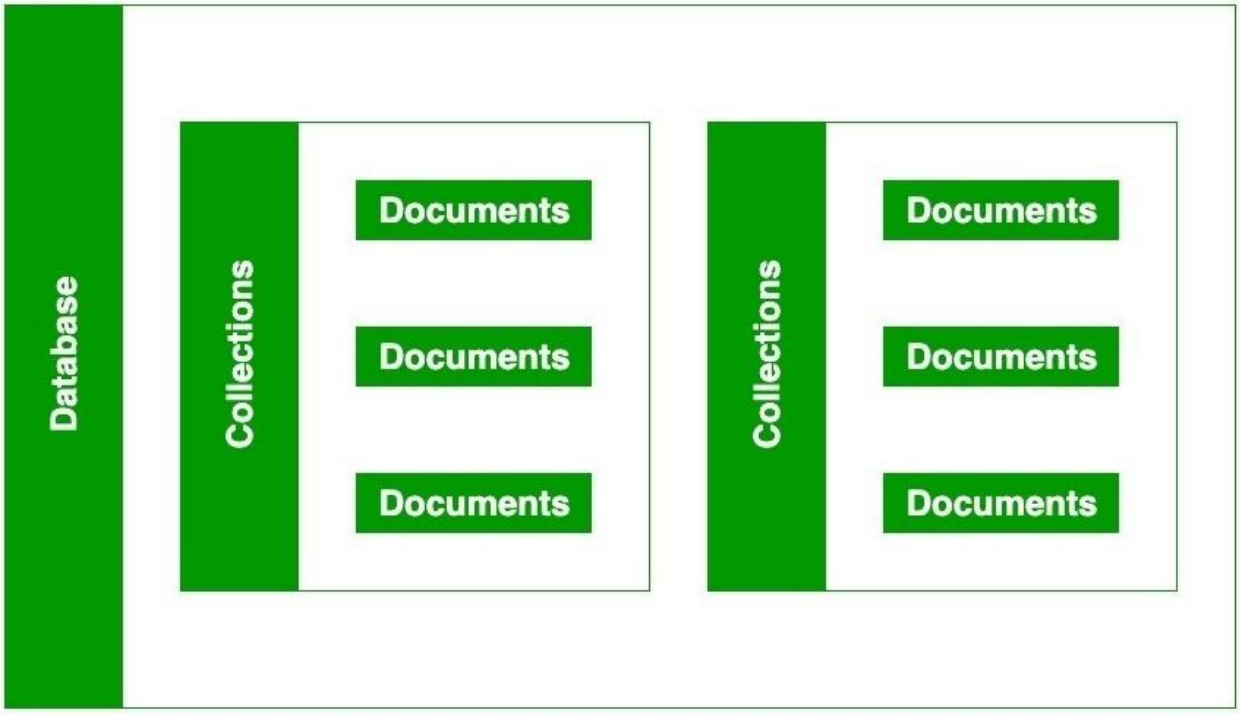
Reason for the use of MongoDB:

- We can use the MongoDB where we have the possibility of developing of an application that which requires a database.
- As a document database, MongoDB makes it easy for developers to store structured or unstructured data. It uses a JSON-like format to store documents.
- MongoDB can also handle high volume and can scale both vertically or horizontally to accommodate large data loads.
- MongoDB has become one of the most wanted databases in the world because it makes it easy for developers to store, manage, and retrieve data when creating applications with most programming languages.
- We can use the MongoDB in desktop application, client server application, enterprise application and also in web application.

Working of the MongoDB:

- MongoDB is a database server and the data is stored in these database.
- MongoDB environment gives a server that can start and then create multiple databases on it using MongoDB. Because of its NOSQL database, the data is stored in the collection and documents.
- Now let us see the relation between database, collection and documents

Data loading, Storage and File Formats



- MongoDB contains ‘N’ number of database. And each database contains ‘N’ number of collections. Finally each collection consists of ‘N’ number of documents.

Data representation in MongoDB:

- The data is represented in JSON format but internally the data is stored in BSON
- BSON is the binary encoded java script object notation.
- Here the BSON is a textual object notation widely used to transmit and store the data across web application. And the JSON is easier to understand as it is human-readable, but compared to BSON, it supports fewer data types.
- BSON have additional data type support like numberInt, objectID, …
- When the BSON is again converted into JSON we have a problem with additional supported data types.
- To solve this problem we get the output in EJSON(Extended JavaScript Object Notation).
- In the MongoDB all the conversions are done by the database engine which is included in the MongoDB.
- The biggest advantage of the MongoDB is of fast accessing of the data in the documents.

Data loading, Storage and File Formats

Key characteristics of MongoDB:

- All the information is available in the single document, it doesn't require join operation, it is fast retrieval.
- Documents are independent to each other and no scheme. Hence we can store unstructured like videos, audio, images…
- We can perform operation like editing, deleting and inserting documents very easily.
- Retrieval data is in the form of JSON which can be understandable by any programming language without any conversions.
- We can store very huge amount of data and hence scalability is more.
- The performance and flexibility is the biggest asset of MongoDB.