]

\\ “**File Handling”**

**File Handling:-->** To manage the data, we need storage areas.

There are two types of Storage areas. 1. Temporary Storage Areas. 2. Permanent Storage Areas

Temporary Storage Areas:--> These storage areas are able to store data temporarily. EX: Buffers, Python Objects

Permanent Storage Areas --> These storage areas are able to store data permanently. EX: File System, DBMS, Data warehouses

**File System:** It is a permanent Storage Area, it able to manage data permanently and it is managed by local Operating system.If we want to use files in python applications we have to use the following steps.1. Create file2. Write Data in Files / Read data from Files 3. Close File.

1. Create File: -------------------->>file = open(fileName, mode)

EX: file = open("E:/abc/xyz/welcome.txt","w")

Note: If use "name" attribute from file we are able to get absolute path of the file. print(file.name)

2. Get Name of the File:---- ------->> name = os.path.basename(file.name)

3. Get Absolute path of the FIle: ------->> print(file.name) or print(os.path.abspath(file.name))

4. Get Parent location :----------->> os.path.dirname(file.name)

5. To get Mode of the file.-------->> print(file.mode)

6. To check readable file------------->> print(file.readable())

7. To check Writable------------------>> print(file.writable())

**import** os  
file = open(**"mukesh.txt"**,**"w"**)  
print(**"File Mode :"**,file.mode)  
print(**"File Name :"**,os.path.basename(file.name))  
print(**"File Dir :"**,os.path.dirname(file.name))  
print(**"Abs Path :"**,file.name)  
print(**"Abs Path :"**,os.path.abspath(file.name))  
print(**"Is Readable :"**,file.readable())  
print(**"Is Writable :"**,file.writable())  
file.write(**'mukesh i love you neelam\n'**)  
file.close()  
print(**'is file closed'**,file.closed)

If we want to Write Data in a File we have to use the following two functions.

1. write()

2. writelines()

Q)What is the difference between write() and writelines() function?

write() function is able to write single line of data.

writelines() function is able to write more than one line which are existed in list to a file.

f=open(**"D:\python class notes\editplusfile/Neelam.txt"**,**"w"**)  
list=[**"ram goto sheopur\nmohan eat 4 chapaties daily\nram and mohan both are friend"**]  
f.writelines(list)  
f.close()

To read the data from source file, we have to use following methods…

1. read()---------- ---------------->>it will use to read total data from source file

2. readline------------------------>>it will use to read only one line of data from source file

3. readlines()--------------------->>it will use to read multiple lines {list form}}of data from source file

4. read(no\_Of\_Characters)-->>it will use to read “n” no. of character from file.

f=open(**"D:\python class notes\editplusfile/Neelam.txt"**,**"r"**)  
data=f.read()  
*#data1=f.readline()  
#data2=f.readlines()  
#data3=f.read(10)  
#print(data1)  
#print(data2)  
#print(data3)*print(data)  
f.close()

Bydefault, files are existed in overriden mode, that is, at each and every write operation previous data will be overridden with new data. If we want to append new data to the old data which is existed in file previously we have to use 'a' as filemode.

f=open(**"D:\python class notes\editplusfile/Neelam.txt"**,**"a"**)  
list1=[**"lamer goto sheopur\nrohan eat 4 chapaties daily"**]  
f.writelines(list1)  
print(**"data append succesffully"**)  
f.close()

**FILE MODES**-> In File System , we are able to use the following modes to perform operations with files.

'w' -->it will used to write a data in complete override mode.

'r' --> it will used to read a data.

'a' --> it will used to write a data in append mode.

'r+'-->Read and Write, but, it will override the old data upto the length of new data.

'a+'--> it will allow write operation in Append mode and it will allow Read operations also.

'w+'--> it will allow write operation in complete override mode and same program it will allow to read data. 'x' --> It will create file, if file is existed then it will generate error.

In Python applications, if we will open the files and store the data and read date from file, we have to close file by close() function to avoid security issue. To Check whether a FILE is closed or not we have to use "closed" variable from File..

file = open(**"D:\python class notes\editplusfile/heloo.txt"**,**"w+"**)  
file.write(**"Welcome To Durgasoft"**)  
file.seek(0)  
print(file.read())  
print(file.closed)  
file.close()  
print(file.closed)

note-In general, in python applications, if we open the file then we have to close that file at the end of python application explicitly, in this context, developers may or may not close the files explicitly, if we want to close the files with out failure then we have to go for an alternative where PVM must close the files at the end of appliactions. To achieve this rejquired we have to create file with 'with' keyword

**with** open(**"D:\python class notes\editplusfile/heloo.txt"**,**"w+"**) **as** file:  
 file.write(**"Welcome To Durgasoft"**)  
 file.seek(0)  
 print(file.read())  
 print(**"File Closed? :"**,file.closed)  
print(**"File Closed? :"**,file.closed)

tell()---->>>it will get current location of the file pointer , we will use tell() function.

Seek(position)--->>it able to move file pointer to a particular location .

**with** open(**"D:\python class notes\editplusfile/heloo.txt"**,**"w+"**) **as** file:  
 file.write(**"durga software solution "**)  
 print(file.tell())  
 file.seek(0)  
 print(file.read(14))  
 print(file.tell())  
 print(file.seek(5))  
 print(file.read(9))  
 print(file.tell())

Q) write program to take file name as dynamic input and display the provided file content on console..

fileName = input(**"Enter File Name : "**)  
**with** open(fileName,**'r'**) **as** file:  
 data = file.read()  
 print(data)

op-

Enter File Name : D:\python class notes\editplusfile/heloo.txt

durga software solution

Q)Write a python program to take file name as dynamic input and to display no of lines, no of words and no of characters of the respective file?

fileName = input(**"Enter File Name : "**)  
**with** open(fileName,**'r'**) **as** file:  
 data = file.read()  
 file.seek(0)  
 datalines = file.readlines()  
 datawords=data.split()  
 charswithoutspace=0  
 **for** word **in** datawords:  
 charswithoutspace=charswithoutspace+len(word)  
 print(**"chars without space :"**,charswithoutspace)  
 *#print(data)* print(**"no of charecter :"**,len(data))  
 print(**"no of words :"**,len(datawords))  
 print(**"no of lines :"**,len(datalines))  
print(datalines)

op-:

Enter File Name : D:\python class notes\editplusfile/heloo.txt chars without space : 21 no of charecter : 23 no of words : 3 no of lines : 1 ['durga software solution']

Q)Write a python program to transfer data from one file to another file?

sourcefile=open(**"D:\python class notes\editplusfile/darling.txt"**,**"r"**)  
data=sourcefile.read()  
targetfile=open(**"D:\python class notes\editplusfile/home.txt"**,**"r+"**)  
targetfile.write(data)  
print(**"data transfer from source file to targetfile"**)  
data1=targetfile.read()  
print(data1)

If we want to work with binary data like images,...then we have to open the files with the file modes like 'rb', 'wb',.....

Q)Write a Python program to transfer image data from one file to another file?

sourcefile=open(**"D:\python class notes\editplusfile/mukesh1.jpeg"**,**"rb"**)  
data=sourcefile.read()  
targetfile=open(**"D:\python class notes\editplusfile/nilam.jpeg"**,**"wb"**)  
targetfile.write(data)  
print(**"image transfer from source file to targetfile"**)  
sourcefile.close()  
targetfile.close()

**CSV Files:----->>>** CSV --> Coma Seperated Values

CSV files are able to manage semi structered data, to manipulate CSV files data python has provided a seperate module in the form of 'csv'.

If we want to write data into CSV files, first we have to get Writer object from file then we have to write data to CSV file through Writer object.

EX

f = open(fileName,'w')

writer = csv.writer(f)

writer.writerow([Col1,col2....])

**import** csv  
file=open(**'D:\python\_exercise\student.csv'**,**'w'**,newline=**''**)  
writer=csv.writer(file)  
writer.writerow([**"sID"**,**"sNAME"**,**"sADD"**,**'sMARK'**])  
writer.writerow([1001,**'ram'**,**'mum'**,78])  
writer.writerow([1002,**'ramu'**,**'nvi-mum'**,74])  
writer.writerow([1012,**'kamu'**,**'swm'**,88])  
writer.writerow([1022,**'renu'**,**'swm'**,83])  
writer.writerow([1023,**'reetu'**,**'swm'**,56])  
writer.writerow([1003,**'reetu'**,**'bpl'**,66])  
print(**'file create'**)  
file.close()

Note: If we write data in CVS file, bydefault, data will be stored in CSV file with a line space between rows, if we want to remove line spaces between rows we have to use 'newline' attribute in open() function.

If we want to get data from CSV file then we have to get reader object from the source file by using reader() method.

Read csv file-:

**import** csv  
f=open(**'D:\python class notes\editplusfile/neelam.csv'**,**'r'**)  
reader=csv.reader(f)  
data=list(reader)  
print(**'accNo\taccholdername\tacctype\tbalance'**)  
print(**'---------------------------------------------'**)  
**for** row **in** data[1::]:  
 **for** col **in** row:  
 print(col, end=**' '**)  
 print()  
f.close()

**Creating ZIP file with no of files:**

-------------------------------------

Zipping : Gatherring all files(no.o file and folder) in to single file.

Unzipping: Unpacking (extracting) all

files and folders from Zip file to current location.

Advantages:

1. SImplifies Files transfermation from one location to location.

2. Optimize Memory utilization

**from** zipfile **import** ZipFile,ZIP\_DEFLATED,ZIP\_STORED  
f=ZipFile(**"D:\python class notes\editplusfile/kamu1.zip"**,**'w'**,ZIP\_DEFLATED)  
f.write(**"D:\python class notes\editplusfile/test.py"**)  
f.write(**"D:\python class notes\editplusfile/test2.py"**)  
f.write(**"D:\python class notes\editplusfile/test3.py"**)  
f.write(**"D:\python class notes\editplusfile/test4.py"**)  
print(**'file successfully zipped'**)  
f.close()

or

**from** zipfile **import** \*  
zip\_File1 = ZipFile(**"D:\python class notes\editplusfile\jkl/docs.zip"**,**'w'**,ZIP\_DEFLATED)  
zip\_File1.write(**"D:\python class notes\editplusfile\jkl/darling.txt"**)  
zip\_File1.write(**"D:\python class notes\editplusfile\jkl/heloo.txt"**)  
zip\_File1.write(**"D:\python class notes\editplusfile\jkl/home.txt"**)  
print(**"success"**)  
zip\_File1.close()

Extracting ZIP File(unzipping) and Displaying Data from ZIP File:

**from** zipfile **import** \*  
zip\_File1 = ZipFile(**"D:\jkl/nela.zip"**,**'r'**,ZIP\_STORED)  
*#zip\_File.write("D:\jkl/darling.txt")  
#zip\_File.write("D:\jkl/heloo.txt")  
#zip\_File.write("D:\jkl/home.txt")*zip\_File1.extract(**"jkl/darling.txt"**,path=**"D:\jkl"**)  
zip\_File1.extract(**"jkl/heloo.txt"**,path=**"D:\jkl"**)  
zip\_File1.extract(**"jkl/home.txt"**,path=**"D:\jkl"**)  
print(**"success"**)  
zip\_File1

**Pickling and Unpickling:**

By using python, we have to prepare two type application.

1. Distributed application

2. standalone application

Distributed application---: if we design and execute any application on the basis of client-server arch or by distributing application logic over multiple machine, then this type application called Distributed application.

standalone application-: If we design and execute any application without using client-server or without using distributed application logic over multiple machine, this type application as known as standalone application.

Note-In standalone application,to perform pickling/unpickling and to store pickled data,we will use flat file(normal txt file)

Pickling: the process of separating data from an object inorder to send an object to network, is called pickling.

Unpickling: the process of reconstructing an object on the basis of data, is called unpickling.

To perform Pickling and Unpickling, Python has provided predefined library in the form of 'pickle' module.

To perform pickling we have to use dump() function from pickle module.

To perform unpickling, we have to use load() function from pickle module.

Ex. Example on Pickling:

**import** pickle  
**class** Employee:  
 **def** \_\_init\_\_(self,eno,ename,esal,eaddr):  
 self.eno = eno  
 self.ename = ename  
 self.esal = esal  
 self.eaddr = eaddr  
 **def** getEmpDetails(self):  
 print(**"Employee Details"**)  
 print(**"----------------------"**)  
 print(**"Employee Number :"**,self.eno)  
 print(**"Employee Name :"**,self.ename)  
 print(**"Employee Salary :"**,self.esal)  
 print(**"Employee Address :"**,self.eaddr)  
file=open(**"E:\lic\mus/emp.txt"**,**'wb'**)  
emp1 = Employee(111,**"AAA"**,5000,**"Hyd"**)  
print(**"Employee Details Before Pickling"**)  
emp1.getEmpDetails()  
pickle.dump(emp1,file)  
print(**"Pickling is perfomred Successfully"**)  
**"**)#unpickling  
file = open(**"E:\lic\mus/emp.txt"**,**"rb"**)#unpickling  
obj = pickle.load(file)  
print(**"Employee Details After Unpickling"**)  
obj.getEmpDetails()

#note-different file check with project-pickle234 in pycharm or editplus-file(D:\jkl)

**Manipulation with directories-:**

**To get Current Working directory we have to use getcwd() function from 'os' module.**

**import os**dir=os.getcwd()  
print(dir)

**To create a directory we have to use mkdir() function from os module.**

**import** os  
os.mkdir(**'E:/bairthday/emp/tcs'**) #use for making folder in directory like tcs  
print(**'directory successfully making'**)

**To create Multiple directories in nested fashion we will use makedirs() function.**

**import** os  
os.makedirs(**'D:/bairthday/a/b'**)#use for complete directory—folder in folder  
print(**'directory successfully making'**)

**To remove a directory from the specified location we will use rmdir() function.**

**import os**os.rmdir(**'D:/bairthday/a/b/emp'**) #**#last folder of directory succesfully remove**  
print(**'last folder of directory succesfully remove'**)

**To delete more multiple directories(all folder) we will use removedirs() function.**

**import** os  
os.removedirs(**'D:/bairthday/a/b'**)  
print(**'Total folder(Directory) succesfully remove'**)

**To rename a directory we will use rename() function:**

**import** os  
os.rename(**'E:/bairthday/a/b'**,**'E:/bairthday/a/student'**,)  
print(**'last folder of dirctory rename succesfully remove'**)

**To rename multiple directories we will use renames() function.**

**import** os  
os.renames(**'E:/bairthday1/emp/tcs'**,**'E:/BDA/EMP\_X/wippro'**)  
print(**'all folder of dirctory rename succesfully '**)

**To get the content of a particular location we will use listdir() function**

**import** os  
content=os.listdir(**'E:/bairthday'**)  
print(**'all content show succesfully '**)  
print(content)

Note: listdir() is able to provide only files and folders of the specified location with out nested folders.

**If we want to get files and folders including nested folders(subfolders) we will use walk() function.**

**import** os  
content=os.walk(**'E:/bairthday'**)  
*#content=os.listdir('E:/bairthday')*print(**'all content show succesfully with subfolder '**)  
print(content)  
**for** dirpath,dirnames,filenames **in** content:  
 print(**"Dir Path :"**, dirpath)  
 print(**"dir names :"**, dirnames)  
 print(**"filenames :"**, filenames)  
 print(**"--------------------------------------"**)

--------------------------------------FILE HANDLING END--------------------------------------

**PDBC[Python Database Connectivity]**

PDBC--: it is the process of interacting with database from python application.

In enterprise applications, it is convention to manage data about the organizations like Employees details, customer details, Services details, products details,....

In enterprise application development , to manage these details we need to use Storage Areas.

There are two types of Storage Areas.

1. Temporary Storage Areas

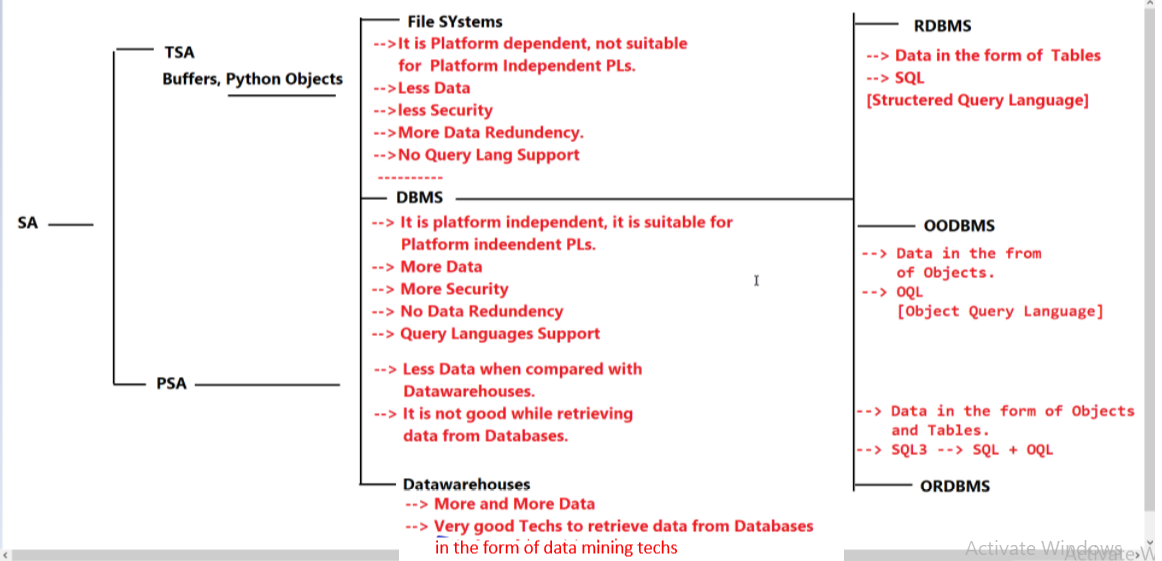
2. Permanent Storage Areas

There are three types of Permament Storage Areas.

1. File Systems

2. Database Management Systems

3. Data warehouses

****

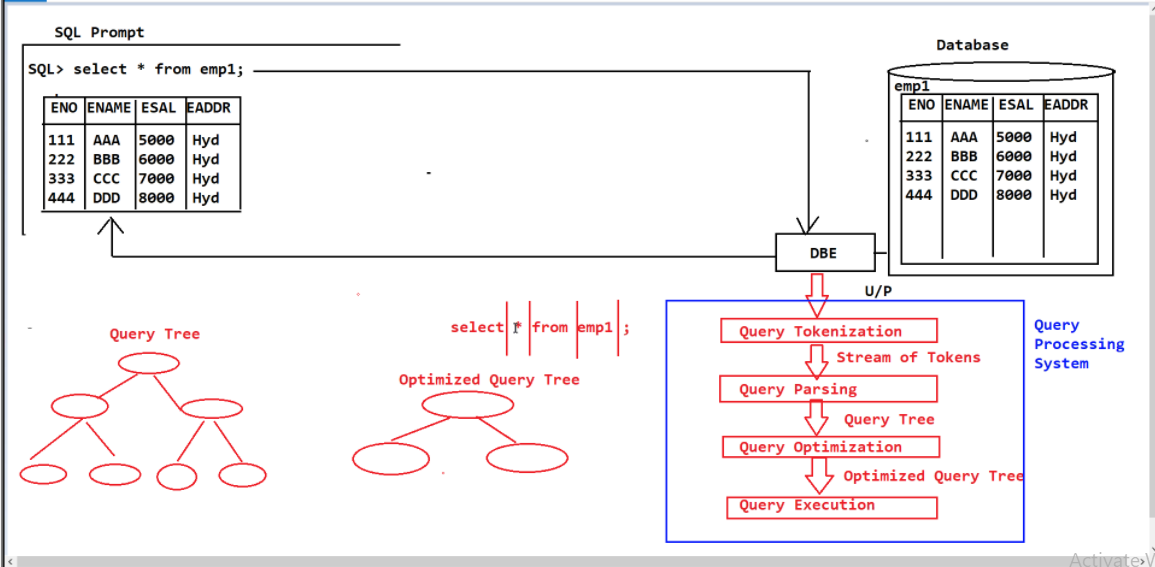
Q)What is the difference between Database and Database Management System?

1. Database is memory storage to store data. Database Management System is a Software System, it able to manage data by storing it and by retrieving it from Database.

2. Database is the collection of interrelated data. Database Management System is the collection of interrelated data and a set of rules and regulations to access data.

**Query Processing System:**

If we submit an SQL query to Database, Where at database , Database Engine will take query and Database will execute sql query by following the following actions. 1. Query Tokenization 2. Query Parsing 3. Query Optimization 4. Query Execution



1. Query Tokenization:--🡪 It able to devide the provided sql query into no of tokens and it will generate stream of tokens as an output.

2. Query Parsing: ---🡪The main intention of this pahse is to check the syntax errors in the provided sql query. This phase will take the stream of tokens as an input and it will constructor a Tree called as Query Truee, if query Tree is success then no syntax errors are existed in the provided sql query. If query tree is not success then there are some syntax errors.

3. Query Optimization :-🡪This phase will apply no of optimization algorithms on query tree to optimize the query tree inorder to reduce execution time and inorder to reduce memory utilization.

4. Query Execution-🡪 execute sql queries by using an interpreter and it will send results to the respective client.

**PDBC**-: It is a step by step procedure to interact with database from python application in order to perform database operation from python application. PDBC is a library, it will provide very good environment to connect with database from python application in order to perform database operation from python application. Generally in PDBC, we will write database logic in python application, we will send python represented database logic to database engine, where database engine has to execute database logic and it has to generate result to python application

Driver is an interface existed between python application and database, it can be used to map python API calls to database API calls and database API calls to python API calls.

Driver and Pdbc library are provided by python in the form of some explicitly module like … 1. Cx\_oracle/ cx\_Oracle\_types --->> for interacting with Oracle database. 2. Mysql.connector------------------->> for interacting with MYSQL database. 3. Pyodbc------------------->> for interacting with all database from python application.

If we are not using Ides to prepare PDBC application then we have to install PDBC libraries by using PIP command--- pip install cx\_oracle

If we use IDE like python, we have to add Cx\_oracle/ cx\_Oracle\_types to prject library.

**Steps to prepare PDBC Applications:**

1. Install Oracle Database 2. In Python Appliactions a)import cx\_Oracle module import cx\_Oracle as cxo

b)Create Connection between Python application and Database

con = cxo.connect("uname/password@DBServerIPAddrtess:DBPort/DBName")

EX: con = cxo.connect("system/durga@localhost:1521/xe")

c)Create Cursor inorder to hold result of the SQl query execution.

cursor = con.cursor()

d)Write and execute SQl query

cursor.execute(Query)

EX: cursor.execute("select \* from emp1")

e)If we execute non select sql queries like create or insert or update or delete or alter or drop then we must perform either commit or rollback operation.

con.commit() --> To store manipulations Permanently.

con.rollback() --> To remove manipulations and to get back previous state.

f)If we execute select sql query then get Results from Cursor

data1 = cursor.fetchone() --> To get only one row.

data2 = cursor.fetchmany(n) --> To get no of rows.

data3 = cursor.fetchall() --> To get all rows.

g)Close the connection and cursor

cursor.close()

con.close()

Ex1.PDBC Application to create table and insert data and display data in DB:

**import** cx\_Oracle **as** cxo  
con = cxo.connect(**"system/durga@localhost:1521/xe"**)  
cursor = con.cursor()  
cursor.execute(**"create table emp1(ENO number(3) primary key,ENAME varchar2(10),ESAL float(5),EADDR varchar2(10))"**)  
print(**"Table emp1 created Successfully"**)  
con.commit()

while True:

eno = int(input("Employee Number : "))

ename = input("Employee Name : ")

esal = float(input("Employee Salary : "))

eaddr = input("Employee Address : ")

cursor.execute("insert into emp1 values(%i,'%s',%f,'%s')"%(eno,ename,esal,eaddr))

print("Employee ",eno," Inserted Successfully")

option = input("Onemore Employee[yes/no]? : ")

if option == "yes":

continue

else:

break

con.commit()

cursor.execute("select \* from emp1")

data = cursor.fetchmany(3)

print("ENO\tENAME\tESAL\tEADDR")

print("------------------------------")

for row in data:

print(row[0],end="\t")

print(row[1],end="\t")

print(row[2],end="\t")

print(row[3],end="\n")

cursor.close()

con.close()

EX4: PDBC Application to perform Updations on database table:

**import** cx\_Oracle **as** cxo  
con = cxo.connect(**"system/durga@localhost:1521/xe"**)  
cursor = con.cursor()

#cursor.execute("update emp1 set ESAL = ESAL + 500 where ESAL < 10000")  
cursor.execute(**"delete from emp1 where ESAL < 10000"**)  
con.commit()  
print(**"Employee records Deleted Successfully"**)

**#drop table from database.**

**import** cx\_Oracle **as** cxo  
con = cxo.connect(**"system/durga@localhost:1521/xe"**)  
cursor = con.cursor()  
cursor.execute(**"drop table emp1"**)  
con.commit()  
print(**"emp1 table dropped from Db successfully"**)  
cursor.close()  
con.close()

**# PDBC with MYSQL Database-create table.**

**import** mysql.connector **as** ms  
con = ms.connect(host=**'localhost'**,database=**'pythondb'**,user=**'root'**,password=**'root'**)  
cursor = con.cursor()  
cursor.execute(**"create table nelam(sno int(3) primary key,Name varchar(20),Address varchar(50),qualification varchar(15), income double(8,2))"**)  
print(**"table created"**)  
con.commit()  
cursor.close()  
con.close()

**# insert data in table-:**

**import** mysql.connector **as** ms  
con = ms.connect(host=**'localhost'**,database=**'pythondb'**,user=**'root'**,password=**'root'**)  
cursor = con.cursor()  
*#cursor.execute("create table nelam(sno int(3) primary key,Name varchar(20),Address varchar(50),qualification varchar(15), income double(8,2))")  
#print("table created")  
#con.commit()  
#data inserterd in table  
#cursor.execute("insert into nelam values(101,'Ram','hyd','BE',50000)")*sql= **"insert into nelam(sno,Name,Address,qualification,income) VALUES(%s,%s,%s,%s,%s)"**records=[  
 (11,**'RAM'**,**'hyd'**,**'BE'**,34000),  
 (12,**'lemen'**,**'Mum'**,**'BBA'**,31000),  
 (13,**'Rita'**,**'Mum'**,**'BCA'**,29000),  
 (14,**'Nita'**,**'dlhi'**,**'BCA'**,26000),  
 (15,**'Sita'**,**'pune'**,**'BSC'**,23000),  
]  
cursor.executemany(sql,records)  
con.commit()  
print(**"records Inserted Successfully"**)  
cursor.close()  
con.close()

**#display the data with table..**

**import** mysql.connector **as** ms  
con = ms.connect(host=**'localhost'**,database=**'pythondb'**,user=**'root'**,password=**'root'**)  
cursor = con.cursor()  
cursor.execute(**"select\*from nelam"**)  
data=cursor.fetchall()  
con.commit()  
print(**"sno\t Name\t Adress\tQualification\tIncome"**)  
  
**for** row **in** data:  
 **for** col **in** row:  
 print(col,**" "**,end=**'\t'**)  
 print()  
cursor.close()  
con.close()

**#delete and update the data in table..**

import mysql.connector

con=mysql.connector.connect(host='localhost',database='pythondb',user='root',password='root')

cursor=con.cursor()

#cursor.execute("update nelam set income = income + 509 where income < 48000 ")

cursor.execute("delete from nelam where income > 30000.00 ")

con.commit()

print("data succesfully update")

cursor.close()

con.close()

**#drop table**

import mysql.connector

con=mysql.connector.connect(host='localhost',database='pythondb',user='root',password='root')

cursor=con.cursor()

cursor.execute("Drop table employees")

con.commit()

print("employees table dropped from pythondb successfully")

cursor.close()

con.close()

**## “MONGNODB”**

For create collection and Insert a docuemt in Collection andPrepare Python file and Write PDBC Code:

**a)Import pymongo module:**

import pymongo

**b)Create Client:**

client = pymongo.MongoClient() or client = pymongo.MongoClient("mongodb://127.0.0.1:27017")

**c)Create Database:**

empDB = client['employee']

**d)Create Collection:** empCollection = empDB.emp1

**e)Perform Database Operations:**

**1) To Insert Docuemtns:** empCollection.insert\_one()empCollection.insert\_many()empCollection.insert()

**2)To find documents:**  empCollection.find\_one() empCollection.find\_many() empCollection.find()

3)To update Cololection: empCollection.update\_one() empCollection.update\_many() empCollection.update()

4)To delete documents from Collection: empCollection.delete\_one() empCollection.delete\_many() empCollection.delete() EX: docuemnt = empCollection.insert\_one({'ENO':111,'ENAME':"AAA",'ESAL':5000,'EADDR':"Hyd"})Close Client. client.close()

**EX1: To create collection(document) and Insert a docuemt in Collection:**

import pymongo

client=pymongo.MongoClient()

db=client['durgadb']

empCollection=db.emp1

empCollection.insert\_one({'ENO':111,'ENAME':"AAA",'ESAL':5000,'EADDR':"Hyd"})

print("data insert succesfully")

client.close()

**for multiple data insert-:**

import pymongo

client = pymongo.MongoClient("mongodb://127.0.0.1:27017")

db=client['durgadb']

Collection=db['employee']

doclist=[]

while True:

eno=int(input("enter eno : "))

ename=input("enter ename : ")

eadd=input("enter eadd : ")

esal=int(input("enter esal : "))

document={'ENO':eno,"ENAME":ename,"EADD":eadd,"ESAL":esal}

doclist.append(document)

option=input("Onemore employees[yes/no]? : ")

if option == 'yes' :

continue

else:

break

Collection.insert\_many(doclist)

print("data insert successfully")

client.close()

**EX3: To Read Single and multiple Document from Collection:**

import pymongo

client=pymongo.MongoClient()

db=client['durgadb']

Collection=db['employee']

#result=Collection.find\_one({ "ENO" : 101}) ##use for display single document from collection

result = Collection.find()

for doc in result:

print(doc)

#print(result)

client.close()

**To perform Updations on docuemnts in Collection-:**

import pymongo

client = pymongo.MongoClient("mongodb://127.0.0.1:27017")

db=client['durgadb']

Collection=db['employee']

#Collection.update\_one({"ENO" : 111},{"$inc":{"ESAL":777}})

Collection.update\_many({"ESAL" :{"$gt":35000}},{"$inc":{"ESAL":777}})

print("data update successfully ...")

client.close()

**To delet the data from the document on collection-:**

import pymongo

client = pymongo.MongoClient("mongodb://127.0.0.1:27017")

db=client['durgadb']

Collection=db['employee']

#Collection.delete\_one({"ENO" : 101}) ##use for delete single document

Collection.delete\_many({"ESAL":{"$gt":40000}})

print("Document Deleted Successfully")

client.close()