**JSS PUBLIC SCHOOL,**

**OOTY**



**PRACTICLE FILE**

**2022-2023**

**COMPUTER SCIENCE**

**Certificate**

NAME : K.MAGUDESHWARAN

CLASS : XII

ROLL NO :

INSTITUTION : JSS PUBLIC SCHOOL

This is certified to be a bonafide work of the student in the Computer Laboratory during the academic year 2022-2023.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Teacher In-charge

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Examiner’s signature

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Principal

Date: **\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | Topic | Date | Pg No | Signature |
| 1. | Area of different shaped using Functions | 06/07/2022 |  |  |
| 2. | Library Functions | 13/07/2022 |  |  |
| 3. | Creating Module and Functions | 20/07/2022 |  |  |
| 4. | Simulating a dice random | 27/07/2022 |  |  |
| 5. | Reading/Writing from a file line by line | 29/07/2022 |  |  |
| 6. | Remove all lines that contain a specific character in a file and write it to another file | 2/08/2022 |  |  |
| 7. | To read a text file using built-in functions | 5/08/2022 |  |  |
| 8. | To read lines from a text file and to display the occurrence of a specific word. | 10/08/2022 |  |  |
| 9. | To find the most commonly occurring word in a text file | 12/08/2022 |  |  |
| 10. | Writing data to Binary file | 16/08/2022 |  |  |
| 11. | Read/Write/Search operations on a binary file | 19/09/2022 |  |  |
| 12. | Updating data in a binary file | 23/08/2022 |  |  |
| 13. | Creating a CSV files | 25/08/2022 |  |  |
| 14. | Writing into CSV file | 29/08/2022 |  |  |
| 15. | Read/Write/Search operations on CSV file | 1/09/2022 |  |  |
| 16. | Encrypting/Decrypting message | 7/09/2022 |  |  |
| 17. | Stacks | 13/09/2022 |  |  |
| 18. | Creating table and inserting record | 16/09/2022 |  |  |
| 19. | To add columns and update table | 20/09/2022 |  |  |
| 20. | To Delete record based on criteria | 23/09/2022 |  |  |
| 21. | Extracting data | 10/10/2022 |  |  |
| 22. | Using Functions-1 | 13/10/2022 |  |  |
| 23. | Using Functions-2 | 18/10/2022 |  |  |
| 24. | Grouping Data | 27/10/2022 |  |  |
| 25. | Sorting data | 09/11/2022 |  |  |
| 26. | SQL Joints | 16/11/2022 |  |  |
| 27. | Interface Python | 23/11/2022 |  |  |

**Program 1: Area of various shapes using functions**

**Index**

**Aim:** To easily calculate the area of different shapes and to reduce the program size and calculations by using functions.

def triangle(b,h):

area=0.5\*b\*h

return area

def square(a):

area=a\*a

return area

def rectangle(l,b):

area=l\*b

return area

def circle(r):

area=3.14\*r\*r

return area

y="yes"

while y=="yes":

print("MENU")

break

print("Which area do you want to find?")

print("1.area of triangle")

print("2.area of square")

print("3.area of rectangle")

print("4.area of circle")

print(" ")

x=int(input("Enter your choice:"))

if x==1:

base=int(input("Enter the length of the base:"))

height=int(input("Enter the height of the triangle:"))

a=triangle(base,height)

print(a)

if x==2:

side=int(input("Enter the length of the square:"))

a=square(side)

print(a)

if x==3:

length=int(input("Enter the length of the rectangle:"))

breadth=int(input("Enter the breadthof the rectangle:"))

a=rectangle(length,breadth)

print(a)

if x==4:

radius=int(input("Enter the radius of the circle:"))

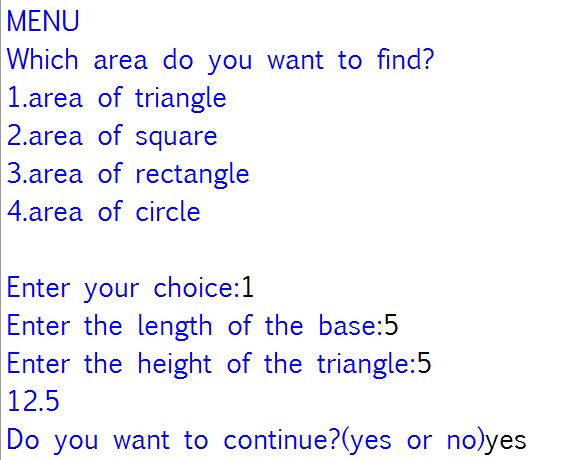
a=circle(radius)

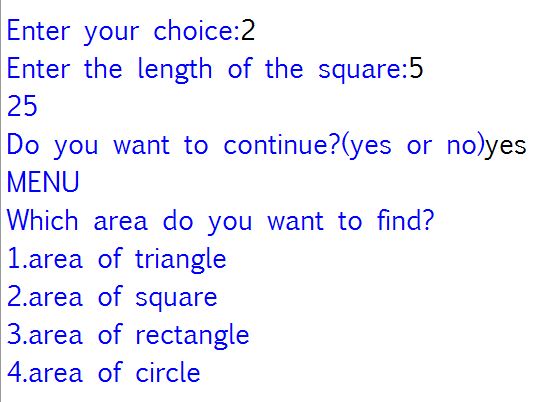
print(a)

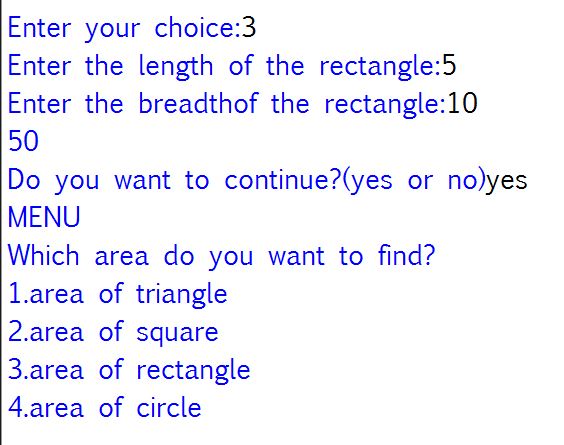
y=input("Do you want to continue?(yes or no)")

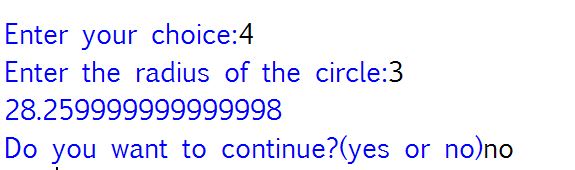
print("MENU")

**Output:**

****

****

****

****

**Program 2: Library Functions:**

**Aim:** To enable us to use various functions in the math and string module

import math

import string

y="yes"

while y.upper()=="YES":

print("1.Math module functions")

print("2.String module functions")

a=int(input("Enter function choice:"))

if a==1:

print("Choose functions:")

print("1.sqrt")

print("2.exp")

print("3.pow")

print("4.sin")

print("5.cos")

print("6.radians")

b=int(input("enter function: "))

if b==1:

x=int(input("Enter value to be calculated:"))

y=math.sqrt(x)

print(y)

elif b==2:

x=int(input("Enter value to be calculated:"))

y=math.exp(x)

print(y)

elif b==3:

x=int(input("Enter value to be calculated:"))

a=int(input("Enter power:"))

y=math.pow(x,a)

print(y)

elif b==4:

x=int(input("Enter value to be calculated:"))

y=math.sin(x)

print(y)

elif b==5:

x=int(input("Enter value to be calculated:"))

y=math.cos(x)

print(y)

elif b==6:

x=int(input("Enter value to be calculated:"))

y=math.radians(x)

print(y)

elif a==2:

print("Choose functions:")

print("1.lower")

print("2.upper")

print("3.title")

print("4.find")

print("5.replace")

b=int(input("Choose function:"))

if b==1:

x=input("Enter string:")

print(x.lower())

elif b==2:

x=input("Enter string:")

print(x.upper())

elif b==3:

x=input("Enter string:")

print(x.title())

elif b==4:

x=input("Enter string:")

y=input("Enter letter to be searched:")

print(x.find(y))

elif b==5:

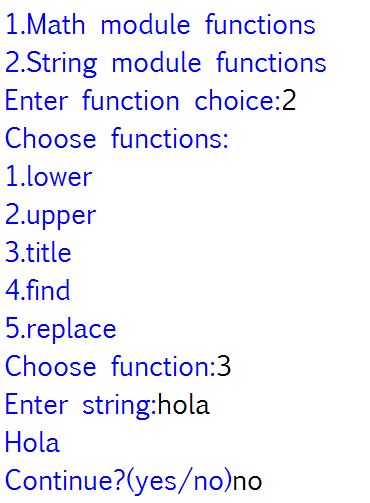
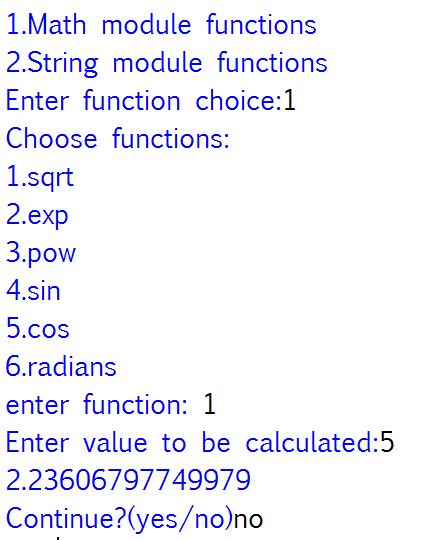
x=input("Enter string:")

y=input("Enter word to be replaced:")

z=input("Enter new word:")

print(x.replace(y,z))

y=input("Continue?(yes/no)")

****

**Program 3: Creating Modules and Packages:**

**Aim**: To easily calculate conversions by importing modules

#Length conversions

def miletokm(x):

km=x\*1.609344

return km

def kmtomile(x):

mile=x/1.609344

return mile

def feettoinches(x):

inches=x\*12

return inches

def inchestofeet(x):

feet=x/12

return feet

#Mass conversions

def kgtotonne(x):

tonne=x\*0.001

return tonne

def tonnetokg(x):

kg=x\*1000

return kg

def kgtopound(x):

pound=x\*2.20462

return pound

def poundtokg(x):

kg=x/2.20462

return kg

**Output:**

****

**Program 4: Simulating a dice random**

**Aim:**To randomly pick numbers using the random module and the program and it simulates the dice.

import random

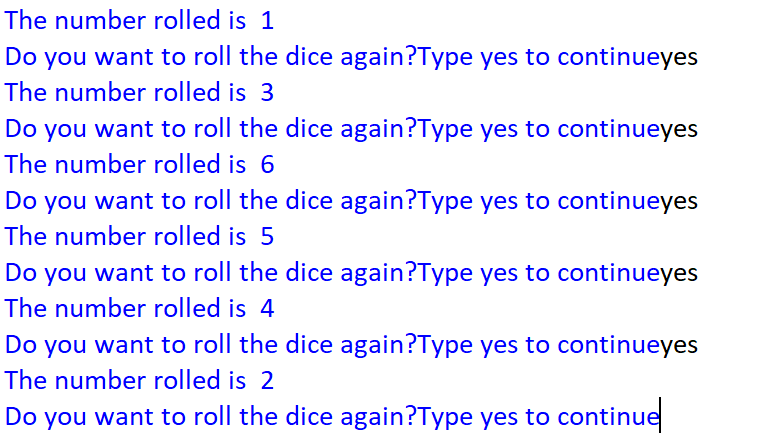
y="yes"

while y=="yes":

print("The number rolled is ", random.randint(1,6))

y=input("Do you want to roll the dice again?Type yes to continue")

**Output:**

****

**Program 5: Writing/Reading data from a file line by line**

**Aim:** To write the requiured data into the file and to read that written data line by line.

**Source Code:**

f1=open("write.txt","w")

a=int(input("Enter the name limit:"))

for i in range(a):

name=input("Enter name:")

f1.write(name)

f1.write("\n")

f1.close()

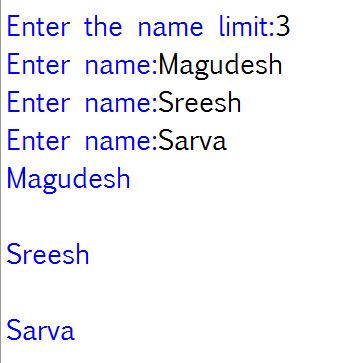
f1=open("write.txt","r")

for i in readlines():

print(i)

f1.close()

**Output:**

****

**Program 6- To Remove all the lines that contain a specific character in a file and write it into another file**

**Aim:** To write all the lines which do not contain a specific character into a new file.

**Source Code:**

f1=open("new.txt","r")

f2=open("new2.txt","w")

s=" "

c=0

for s in f1.read():

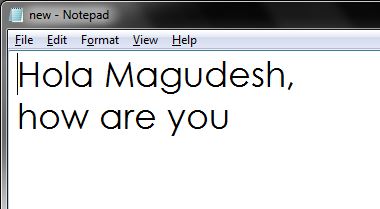
if "a" not in s:

f2.write(s)

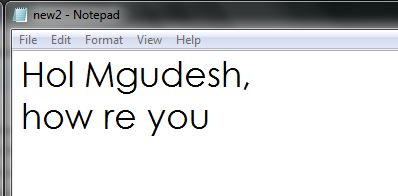
f1.close()

f2.close()

**Output:**

****

new.txt

****

new2.txt

**Program 7-To read a text file using built-in functions**

**Aim:** To display all the uppercase, lowercase, vowels consonants and special characters in the text file.

**Source code:**

f1=open("new.txt","r")

uc=lc=sc=vo=co=0

for i in f1.read():

if i.isalpha():

if i.isupper():

uc+=1

if i in ['a','e','i','o','u']:

vo+=1

else:

co+=1

elif i.islower():

lc+=1

if i in ['a','e','i','o','u']:

vo+=1

else:

co+=1

elif not i.isspace():

sc+=1

print("Number of upper case characters:",uc)

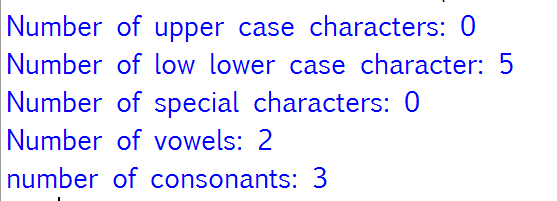
print("Number of low lower case character:",lc)

print("Number of special characters:",sc)

print("Number of vowels:",vo)

print("number of consonants:",co)

**Output:**

****

**Program 8- To find the most commonly occuring word in a text file**

**Aim:**To find the most commonly occuring word in a text file and listing the frequencies of words in text file

**Source code:**

myfile=open(r"C:\Users\computer\Desktop\new.txt","r")

contents=myfile.read()

wordlist=contents.split()

wordfreq=[]

high=0

word=''

existing=[]

for w in wordlist:

wcount=wordlist.count(w)

if w not in existing:

wordfreq.append([w,wcount])

existing.append(w)

if wcount>high:

high=wcount

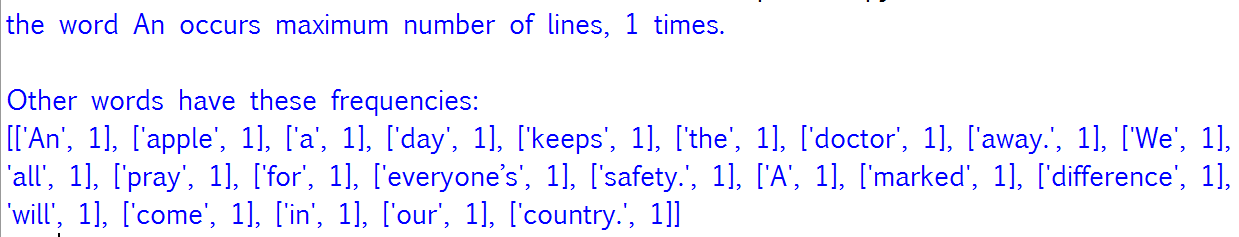
word=w

print("the word",word,"occurs maximum number of lines,",high,"times.")

print("\nOther words have these frequencies:")

print(wordfreq)

**Output:**

****

**Program 9-To read lines from a text file and display the occurrence of a specific word**

**Aim:** To read the file and to count the occurrence of a word that has been given as the input

**Source code:**

myfile=open(r"C:\Users\computer\Desktop\new.txt","r")

count=0

x=input("Enter the word:")

for i in myfile:

words=i.split()

for w in words:

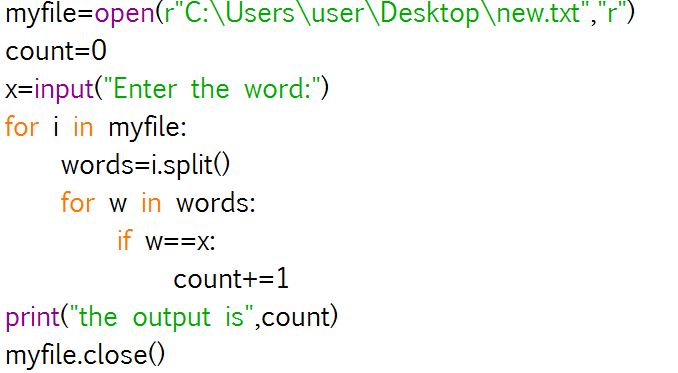
if w==x:

count+=1

print("the output is",count)

myfile.close()

**Output:**

****

new.txt

**Program 10- Writing data to binary file**

**Aim:** To get student data(roll no, name,marks) from user and write onto a binary file

**Source code:**

import pickle

stu={}

stufile=open("stu.dat","wb")

ans='y'

while ans =='y':

rno=int(input("Enter the roll number:"))

name=input("Enter name:")

marks=float(input("Enter marks:"))

stu['rollno']=rno

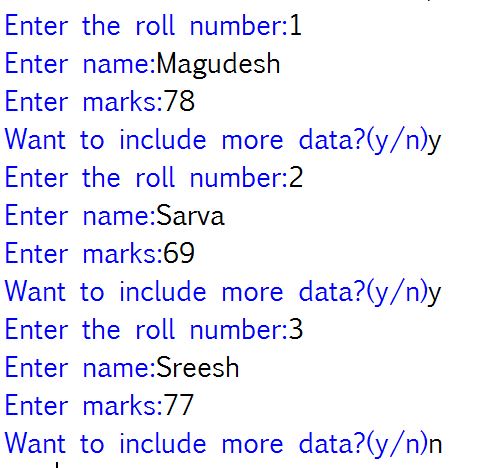
stu['name']=name

stu['marks']=marks

pickle.dump(stu,stufile)

ans=input("Want to include more data?(y/n)")

**Output:**

****

**Program11-Read/Write/Search operations on a binary file**

**Aim:** To read and display students details from a binary file and search for a record in the file

**Source code:**

import pickle

def read():

stu={}

fin=open("stu.dat","rb")

try:

print('file stu.dat stores these records')

while True:

stu=pickle.load(fin)

print(stu)

except EOFError:

fin.close()

def display():

fin=open(r"C:/Users/python/Documents/12th python/stu.dat",'rb')

found=False

a=int(input("Enter the record to be searched"))

try:

print("Searching in file stu.dat...")

while True:

stu=pickle.load(fin)

if stu['Roll no']==a:

print(stu)

found=True

except EOFError:

if found==False:

print("No such record found in the file")

else:

print("search successful")

fin.close()

answer='y'

while answer=='y':

print("MENU")

print("1.Read and display the records")

print("2.Search for records")

z=int(input("Enter the function:"))

if z==1:

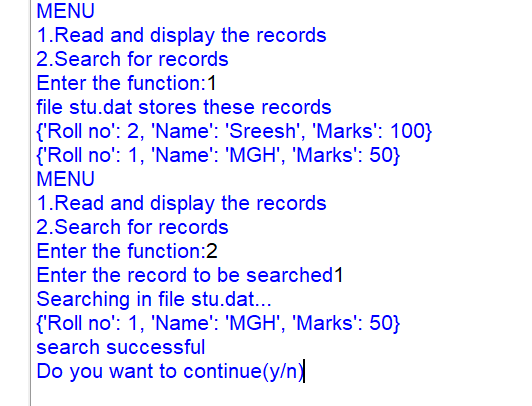
read()

if z==2:

display()

answer=input("Do you want to continue(y/n)")

**Output:**

****

**Program 12-Updating data to binary file**

**Aim:** To update details of a student on a binary file

**Source code:**

import pickle

stu={}

a=int(input('Enter the roll no to be updated.: '))

b=input('Enter the detail to be updated')

with open(r'C:\Users\python\Documents\Magudesh 12th\stu.dat','rb+') as file:

while True:

rpos=file.tell()

print(rpos)

stu=pickle.load(file)

if stu['Roll no']==a:

if b=='Name':

c=input('Enter the new name.: ')

stu[b]=c

print(stu)

elif b=='Roll no':

c=input('Enter the new roll no.: ')

stu[b]=c

elif b=='Marks':

c=float(input('Enter the new marks.: '))

stu[b]=c

print(file.seek(rpos))

pickle.dump(stu,file)

found=True

break

file.close()

def recs():

s={}

with open('C:\\Users\\python\\Documents\\Magudesh 12th\\stu.dat','rb') as t:

print('tell',t.tell())

while True:

s=pickle.load(t)

print(s)

if found==True:

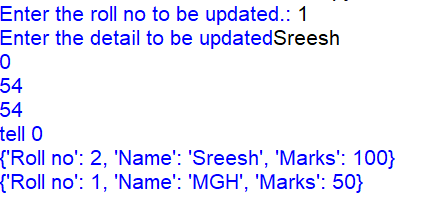
recs()

print('Record Updated')

else:

print('No record found')

**Output:**

****

**Program 13: Writing into CSV file**

**Aim**:To ceate a CSV file to store employee data(emp no, name , designation, salary)

**Source code:**

import csv

fh=open("employee.csv","w")

empwriter=csv.writer(fh)

empwriter.writerow(['empno','Name','designation','salary'])

for i in range(2):

print("employee record",i+1)

empno=int(input("enter the empno:"))

name=input("enter the name:")

designation=input("enter the designation:")

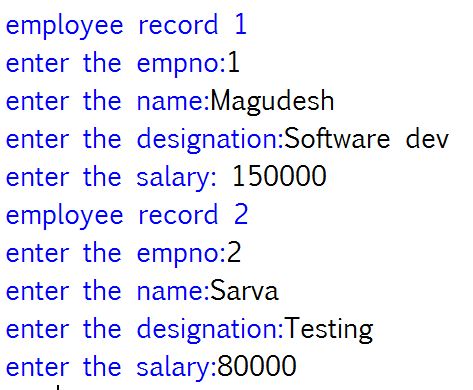
salary=int(input("enter the salary:"))

emprec=[empno,name,designation,salary]

empwriter.writerow(emprec)

fh.close()

**Output:**

****

**Program 14:Creating CSV file**

**Aim**:To get item details (code, description,price) for multiple items from the user

**Source code:**

import csv

fh=open("items.csv","w")

iwriter=csv.writer(fh)

answer='y'

itemrec=[['item\_name','description','price']]

print("enter item details")

while answer=='y':

iname=input("enter item code:")

desc=input("enter description")

price=float(input("enter price:"))

itemrec.append([iname,desc,price])

answer=input("Want to enter more items?(y/n)")

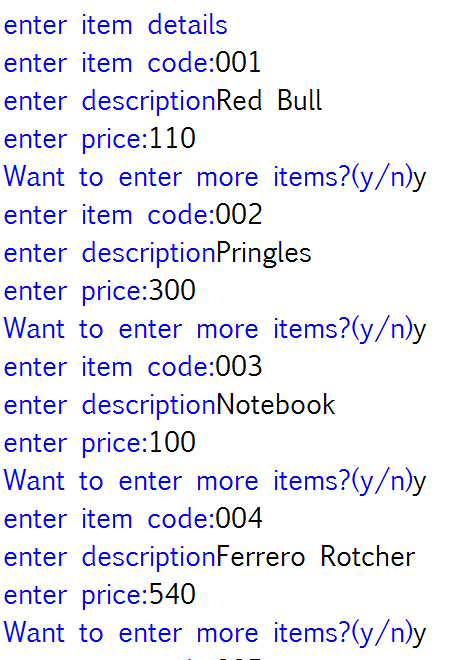
else:

iwriter.writerows(itemrec)

print("Records written successfully")

fh.close()

**Output**

****

**Program 15: Read/Write/Search operations on CSV file**

**Aim**:To create a Menu driven program to write, read and search records on a CSV file

**Source Code:**

**from csv import reader**

**def f\_csvread(b):**

**f=open(b,"r")**

**dt=reader(f)**

**data=list(dt)**

**f.close()**

**print(data)**

**from csv import writer**

**def f\_csvwrite(c):**

**f=open(c,"w")**

**dt=writer(f)**

**l=[]**

**n=int(input("Enter the number of fields:"))**

**for i in range(n):**

**x=input("Field name")**

**l.append(x)**

**dt.writerow(l)**

**f.close()**

**def search(m):**

**f=open(m,"R")**

**s=reader(f)**

**n=input("Enter data to be searched:")**

**print(s)**

**flag=0**

**for row in s:**

**for field in row:**

**if field==n:**

**print("Record found")**

**print(row)**

**print("\*\*\*\*\*")**

**flag=1**

**if flag==0:**

**print("Record not found")**

**print("\*\*\*\*\*")**

**f.close()**

**while True:**

**print("CSV file functions")**

**print("1.To read a file")**

**print("2.To write a file")**

**print("3.To search for a record")**

**print("4.Exit")**

**ch=int(input("Enter your choice:"))**

**if ch==1:**

**b=input("Enter the csv file wth the extension .csv")**

**f\_csvread(b)**

**if ch==2:**

**c=input("Enter the csv file to create with the extension .csv")**

**f\_csvwrite(c)**

**if ch==3:**

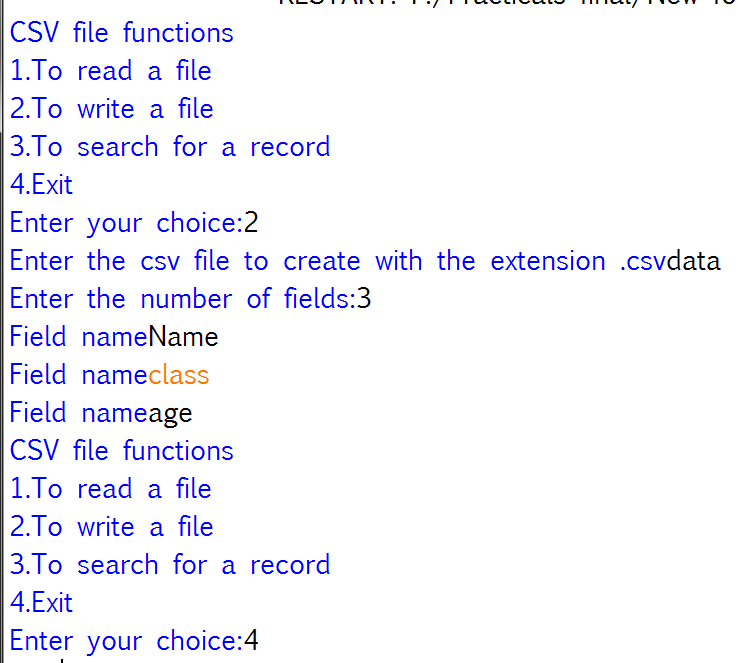
**m=input("Enter the csv file in which you want the row to be searched with the extension .csv")**

**search(m)**

**if ch==4:**

**break**

**Output:**

****

**Program 16: Encypting/Decrypting Messages**

**Aim**:To encrypt and decrypt strings that is given as the input

**Source Code:**

def encrypt (sttr,enkey):

return enkey. join (sttr)

def decrypt(sttr,enkey):

return sttr.split(enkey)

mainstring=input("Enter the mainstring:")

encryptstr=input("Enter Encryption key:")

enstr=encrypt(mainstring,encryptstr)

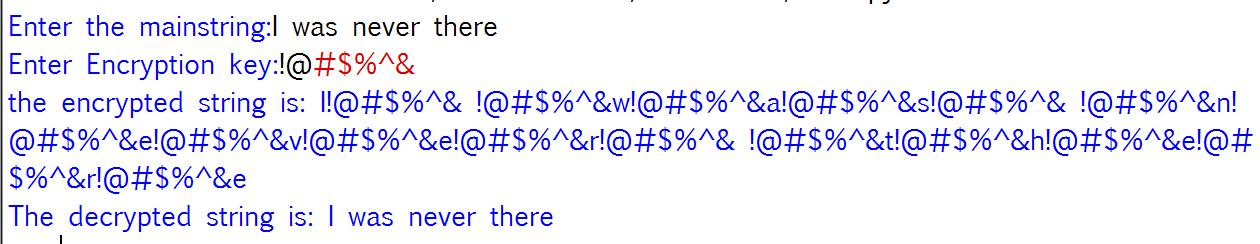
delst=decrypt(enstr,encryptstr)

destr="".join(delst)

print("the encrypted string is:",enstr)

print("The decrypted string is:",destr)

**Output:**

****

**Program 17:Stacks**

**Aim:** To write a menu driven program to perform push, pop, peek and display operations in stackusing list as per the structure given e\_no,e\_name,e\_sal

**Source code:**

stack=[]

def view():

if stack==[]:

print("stack empty")

else:

for x in range(len(stack)):

print(stack[x])

def push():

e\_no=int(input("Enter Employee Number"))

e\_name=input("Enter Employee name")

e\_sal=int(input("Enter salary"))

l=[e\_no,e\_name,e\_sal]

stack.append(l)

def pop():

if (stack==[]):

print("Stack empty")

else:

item=stack.pop(-1)

print("Deleted element:",item)

def peek():

item=stack[-1]

print("Peeked element:",item)

print ("Stack operation")

print("\*\*\*\*\*")

print("1.view")

print("2.push")

print("3.pop")

print("4.peek")

while True:

choice=int(input("Enter choice:"))

if choice==1:

view()

elif choice==2:

push()

elif choice==3:

pop()

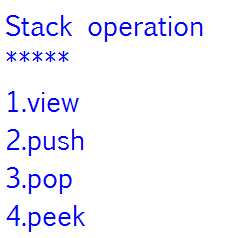
elif choice==4:

peek()

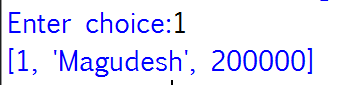
else:

print("wrong choice")

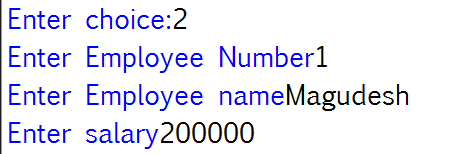
**Output:**

****

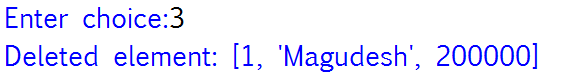
**1.)View**

****

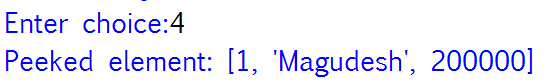
**2.)Push**

****

**3.)Pop**

****

**4.)Peek**

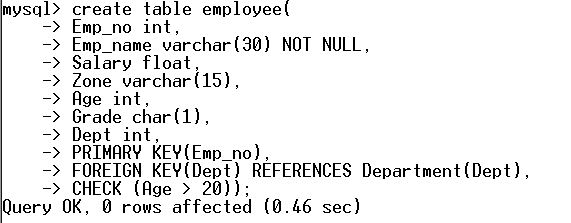
****

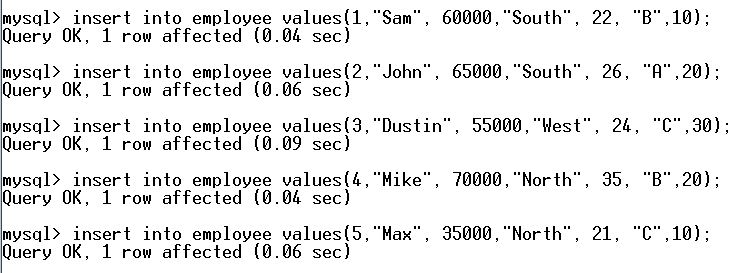
**Program 18:Creating tables and inserting records**

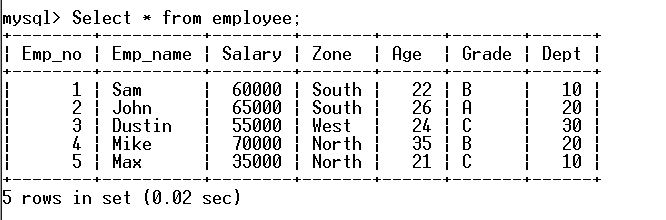
1. Create table Department based on the following instance chart and populate the table.

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Emp\_no | Integer | Primary Key |
| Emp\_name | Varchar(30) | Not Null |
| Salary | Float |  |
| Zone | Varchar(15) |  |
| Age | Integer | Greater than 20 |
| Grade | Char(1) |  |
| Dept | Integer | Foreign Key(Department table) |

**Source Code:**



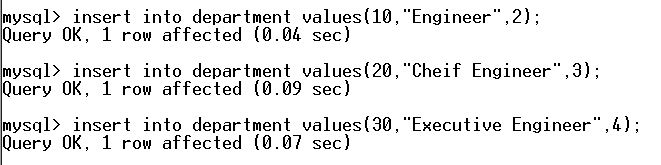
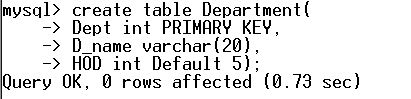


**Output:** 

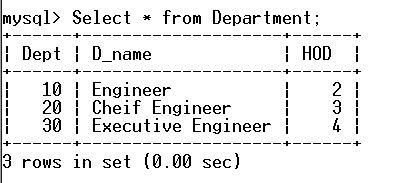
**2.Create table Department based on the following instance chart and populate the table**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Dept | Integer | Primary Key |
| D\_name | Varchar(20) | Not Null |
| HOD | Integer | Default-5 |

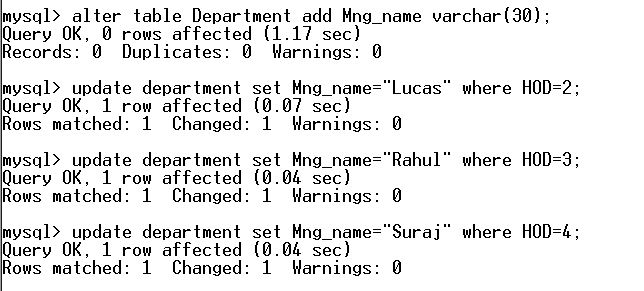
**Source Code:**



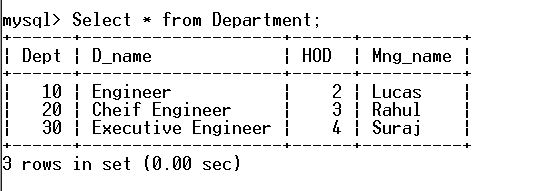
**Output:**



**Program 19- To add tables and update table**

**Source Code: **

**Output:**

****

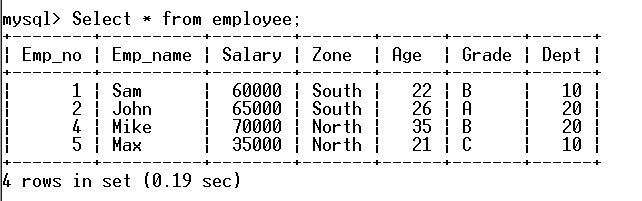
**Program 20-To delete records based on criteria**

1.Delete all the records from Employee table belonging to Central Zone and having grade A

**Source code:**

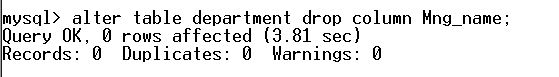
**PL-3.1**

**Output:**

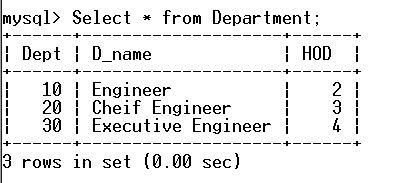
****

2.Delete column Mng\_Name from department table

**Source Code:**

****

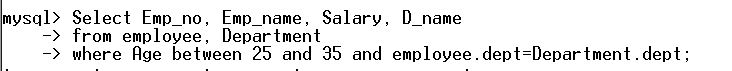
**Output:**

****

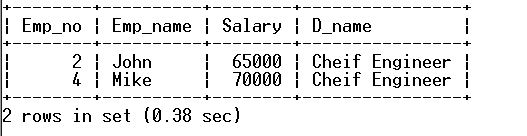
**Program 21 – Extracting Data**

1.Display Emp\_no, Emp\_name, Salary and corresponding D\_name of all the employees whose age is between 25 and 35(both values inclusive).

**Source Code:**

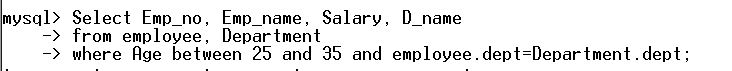
****

**Output:**

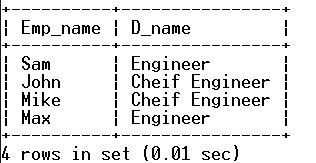
****

2.Display D\_name and corresponding Emp\_name from tables department and employee**.**

**Source code:**

****

**Output**

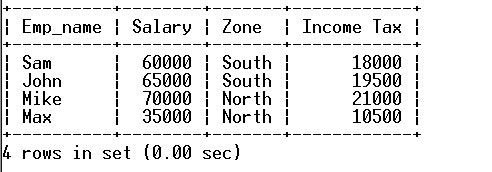
****

3.Display Emp\_name, Salary, Zone and income Tax of all the employees with appropriate column headings. (Income Tax to be calculated 30% of the salary)

**Source Code:**

**PL-4.3.3**

**Output:**

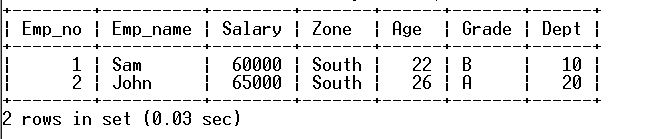
****

4.Display all details of employees of south zone whose salary is greater than 50,000.

**Source Code:**

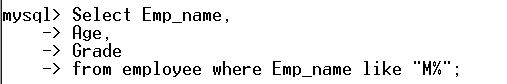
**PL-4.4.1**

**Output:**

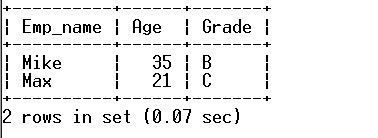
****

5.Display Emp\_Name, Age, grade of employees whose name started with the character ‘M’.

**Source Code:**

****

**Output:**

****

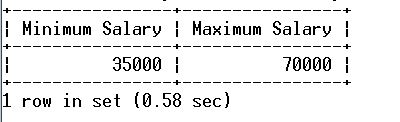
**Program 22-Using Functions-1**

1.Display maximum salary and minimum salary from table employee under appropriate column headings**.**

**Source Code:**

**PL-5.1.2**

**Output:**

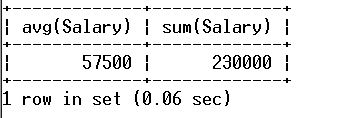
****

2.Display the average and total salary from table employee.

**Source Code:**

**PL-5.2.2**

**Output:**

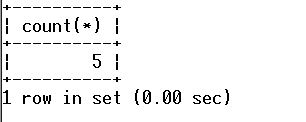
****

3.Count the number if employees in the table employees.

Source Code:

**PL-5.3.1**

**Output:**

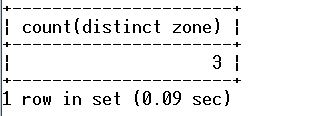
****

4.Count the number of distinct zones from table employee.

**Source Code:**

**28**

**Output:**

****

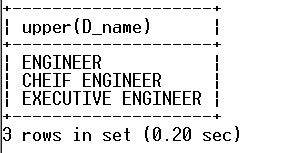
**Program 23-Using Functions-2**

1.Display D\_name of the table department in uppercase.

Source Code:

**30**

**Output:**

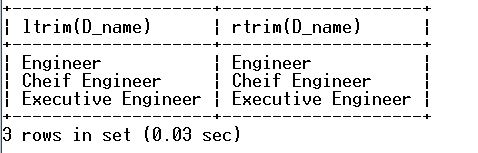
****

2.Removing leading and trailing spaces from D\_name field of department table.

**Source Code:**

**32**

**Output:**

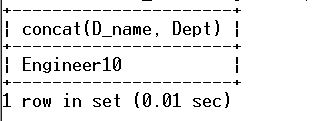
****

3.Concatenate D\_name and Dept of the table department having D\_Name as ‘Engineer’.

**Source Code:**

**34**

**Output:**

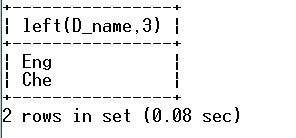
****

4.Display first three characters extracted from D\_name column of the table department whose Dept is not 30.

**Source Code:**

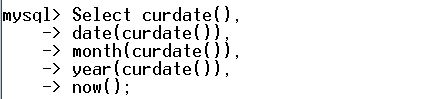
**36**

**Output:**

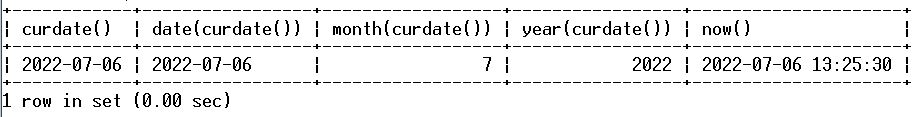
****

5.Display the date/time queries to return current date, date only, month only, year only, current date and time at which the function executes.

**Source Code:**

****

**Output:**

****

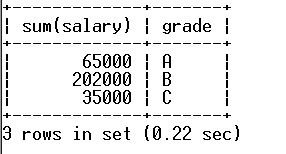
**Program 24-Grouping Data**

1.Display grade wise total salary

**Source Code:**

**40**

**Output:**

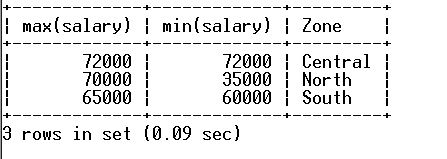
****

2.Display maximum and minimum salary in each zone.

Source Code:

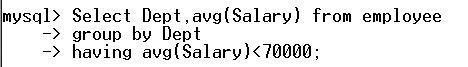
**42**

**Output:**

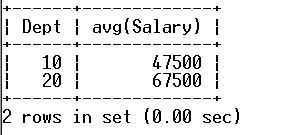
****

3.Display department wise average salary having average salary less than 70000.

**Source Code:**

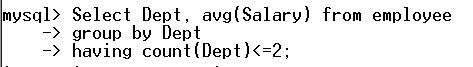
****

**Output:**

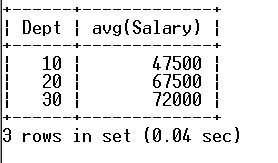
****

4.Display department wise average salary having dept count less than or equal to 2.

**Source Code:**

****

**Output:**

****

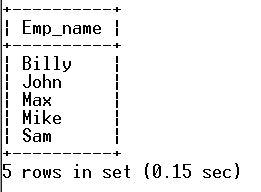
**Program 25-Sorting Data**

1.Display names from table employee in ascending order.

Source Code:

**48**

**Output:**

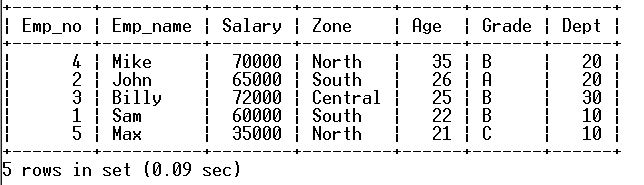
****

2.Display all the details of employees in descending order of age from table employees.

**Source Code:**

**50**

**Output:**

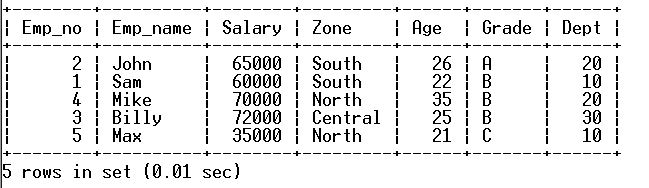
****

3.Display all the details of employees in descending order of grade and then by descending order of age from table employee.

**Source Code:**

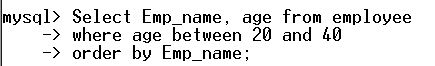
**52**

**Output:**

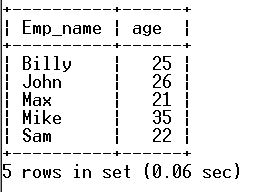
****

4.Display name and age of employees in ascending order of names where age is between 20 and 40.

**Source Code:**

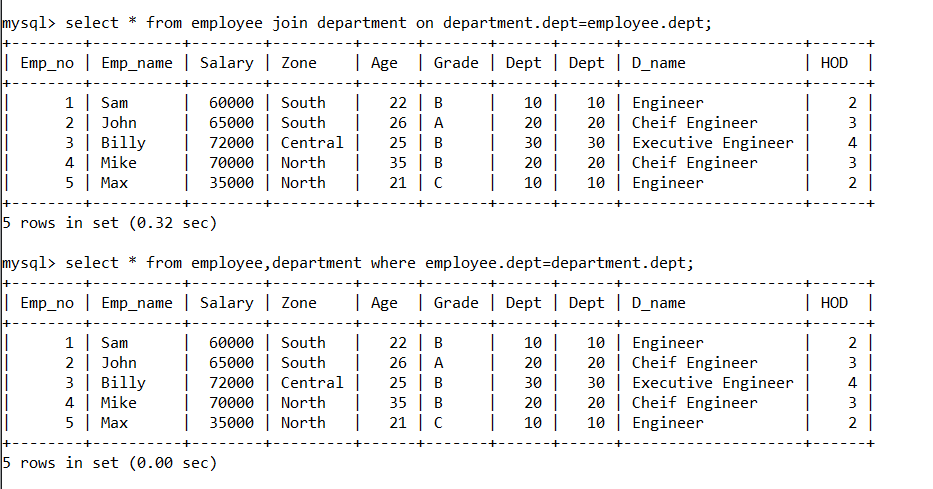
****

**Output:**

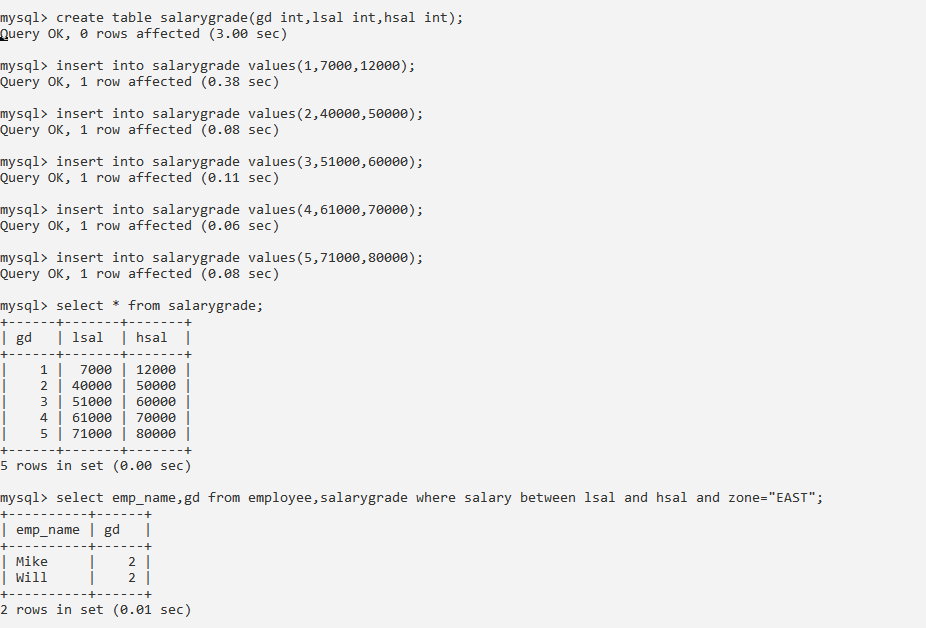
****

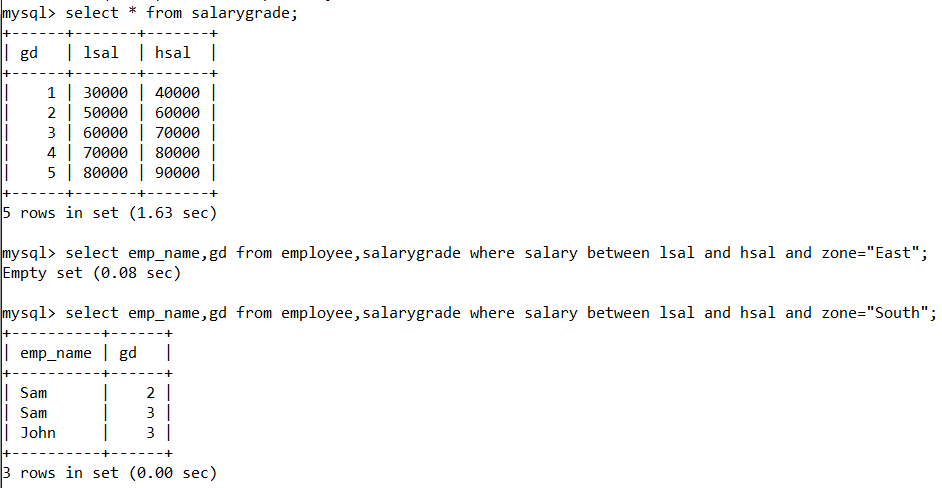
**Program 26: Joints**

**Equi-join**

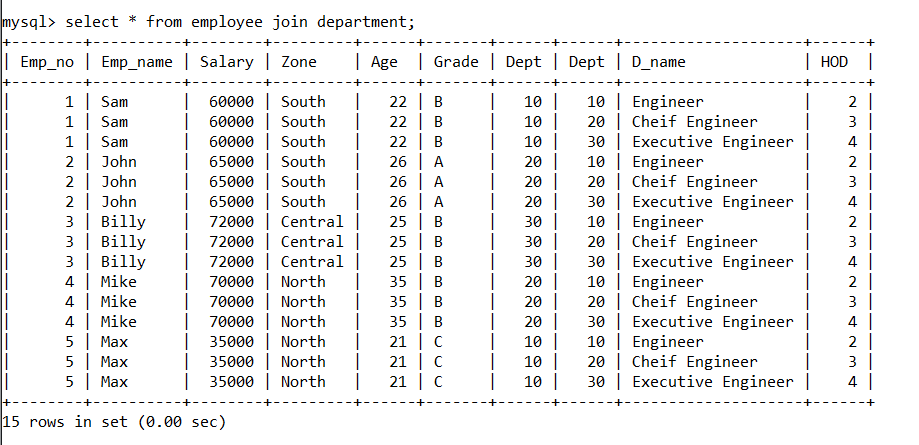
****

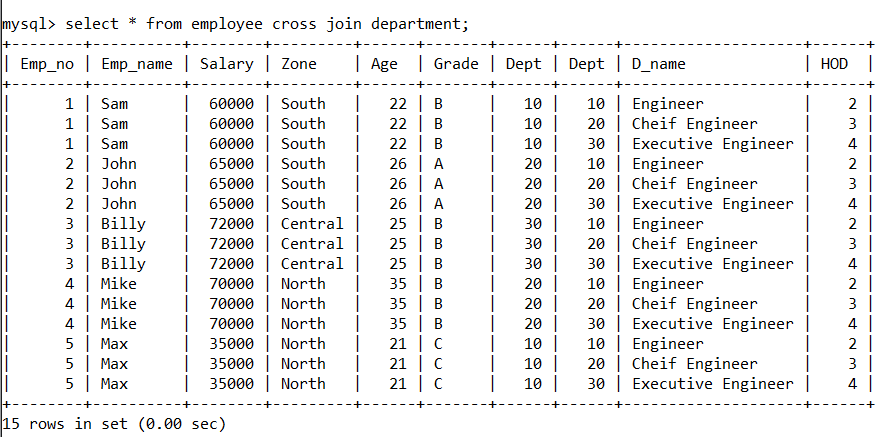
**Non-Equi join**

****

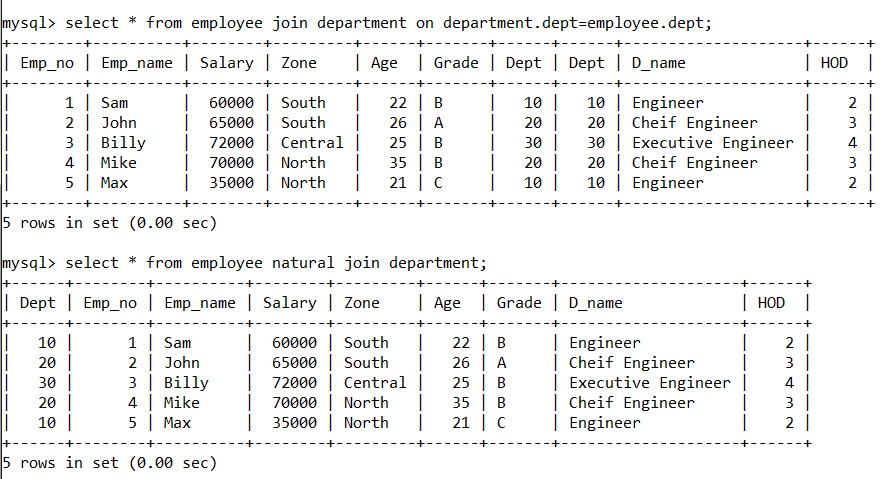
****

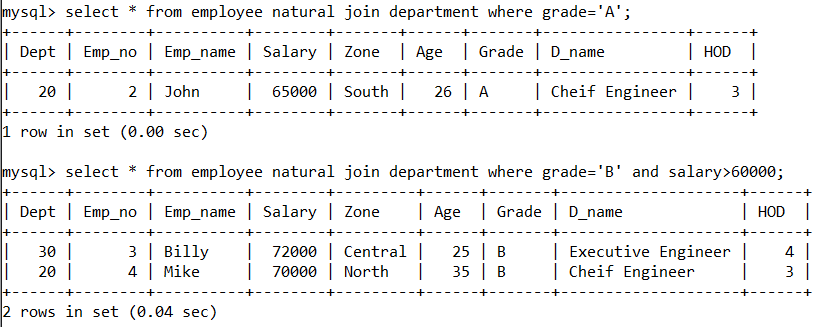
**Cartesian Product:**

****

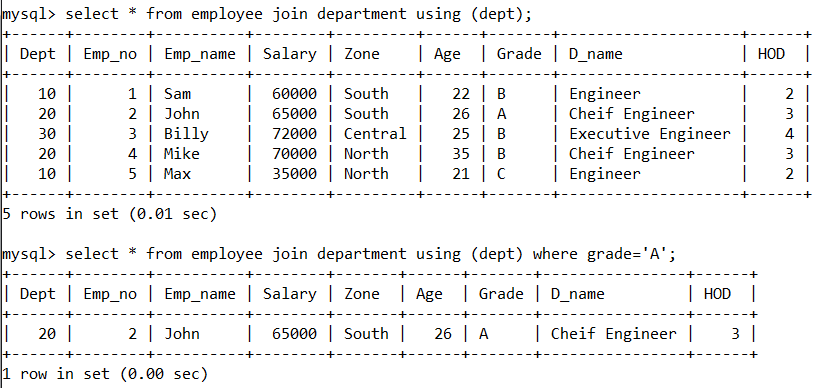
****

**Natural-Join**

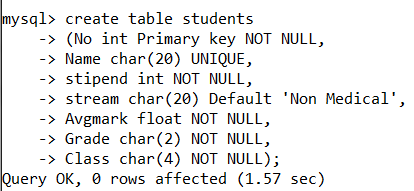
****

****

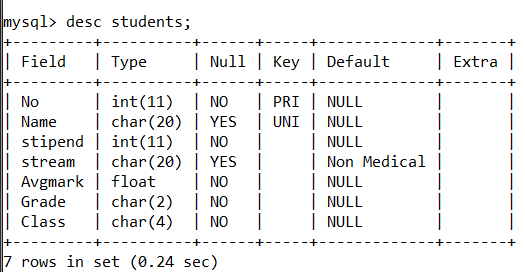
**‘Using’ clause instead of ‘on’**

****

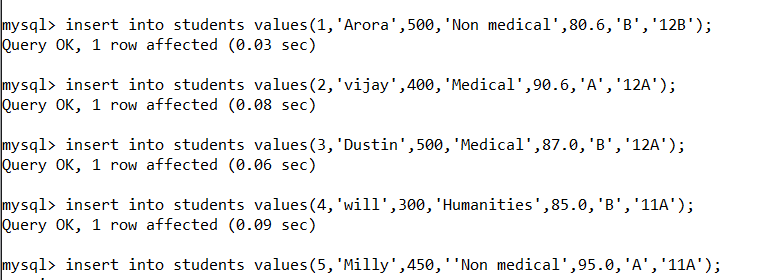
**Creating table with Constraint:**

****

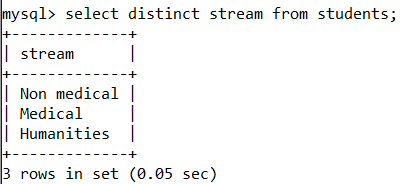
**Describing Table:**

****

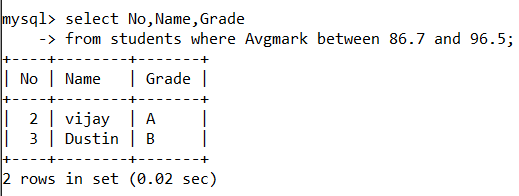
**Inserting Values:**

****

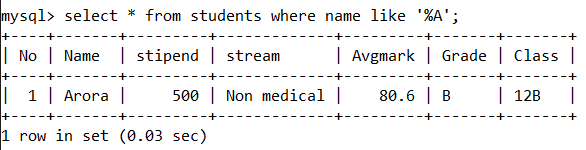
**Select (Distinct)**

****

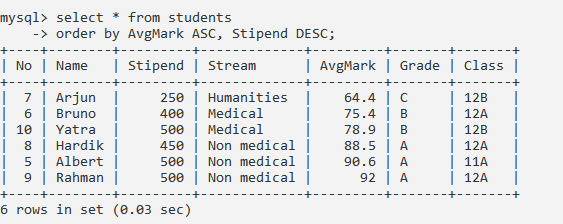
**‘Where’ and ‘Between’ Clause**

****

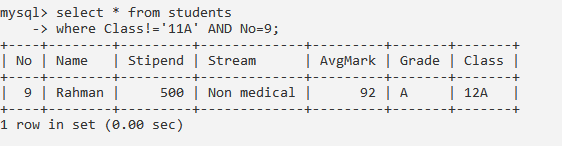
**‘Where’ and ‘Like’ Clause**

****

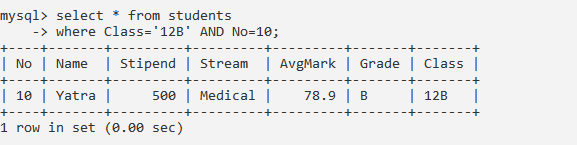
WHERE CLAUSE USING(ASC AND DESC)



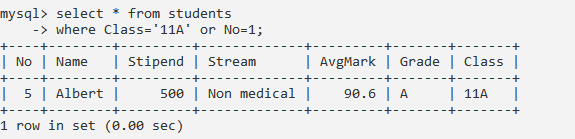
**Where clause using ‘ != ’**

****

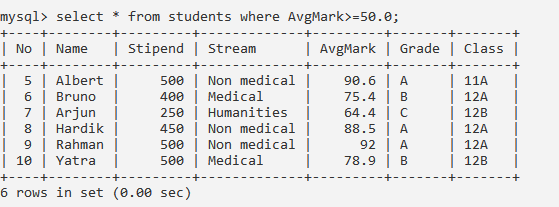
**Where Clause using ‘And’**

****

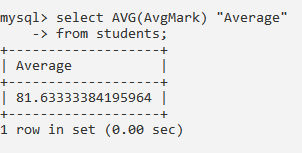
**Where Clause using ‘OR’**

****

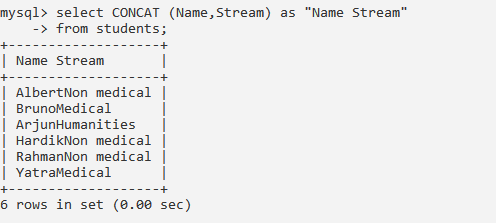
**Where clause using (Relations)**

****

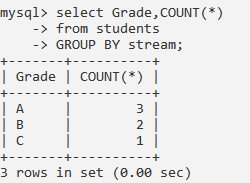
**Using Function(Avg)**

****

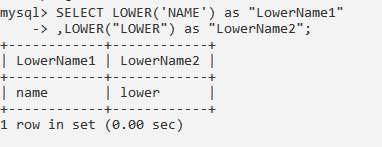
**Using Function (Concat)**

****

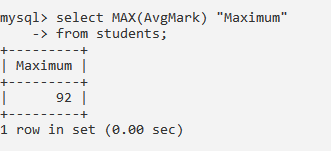
**Using Function(Count)**

****

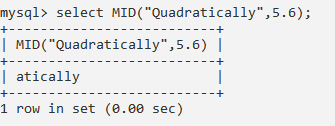
**Using function(lower())**

****

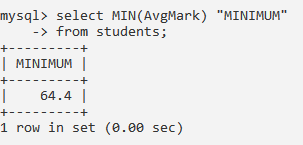
**Using Function (Max())**

****

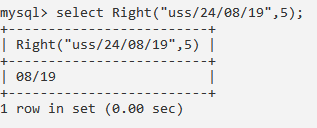
**Using Function(MID())**

****

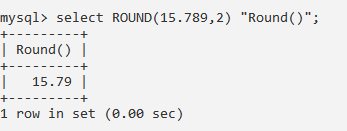
**Using Function(Min())**

****

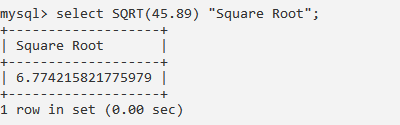
**Using Function(Right())**

****

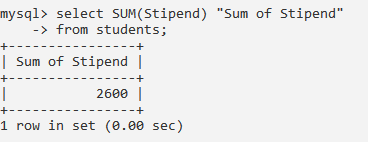
**Using Function(Round())**

****

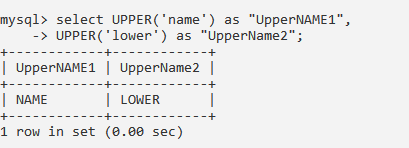
**Using Function(Sqrt())**

****

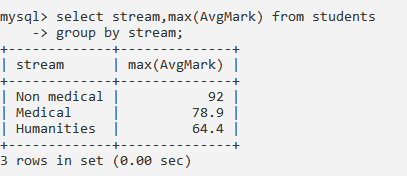
**Using function(sum())**

****

**Using Function (Upper())**

****

**Group By:**

****

**Program 27: Interface Python**

1.Creating Table

**Source code:**

import mysql.connector as mysql

conn = mysql.connect(host='localhost',user='root',

passwd='rootadmin',db='cspractical')

curs = conn.cursor()

def Create():

curs.execute("CREATE TABLE students(s\_no int primary key,name varchar(20),class varchar(4),s\_group varchar(4),marks float)")

conn.commit()

print('Table students Created')

Create()

**Output:**

****

2.)Adding Records

**Source code:**

def Adding():

a=int(input("Enter roll no"))

b=input('Enter Name')

c=input('Enter class')

d=input('Enter S\_group')

e=input('Enter Mark')

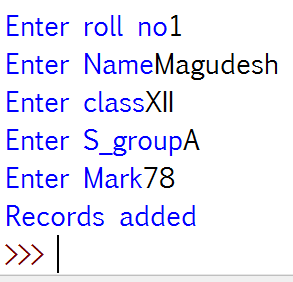
curs.execute("insert into students values({},'{}','{}','{}',{})".format(a,b,c,d,e))

conn.commit()

print('Records added')

Adding()

**Output:**



3.)Updating Records

**Source Code:**

def Update():

user\_id=int(input('Enter User ID.: '))

print('1. Name')

print('2. Class')

print('3. S\_group')

print('4. Marks')

a=int(input('Enter what you want to update.: '))

if a==1:

n=input('Enter New Name.: ')

curs.execute("UPDATE students set name='{}' where s\_no={}".format(n,user\_id))

print('Name Updated')

elif a==2:

c=input('Enter New Class.: ')

curs.execute("UPDATE students set class='{}' where s\_no={}".format(c,user\_id))

print('Class Updated')

elif a==3:

Sg=input('Enter New Name.: ')

curs.execute("UPDATE students set s\_group='{}' where s\_no={}".format(Sg,user\_id))

print('S\_group Updated')

elif a==4:

m=input('Enter New Mark.: ')

curs.execute("UPDATE students set marks={} where s\_no={}".format(m,user\_id))

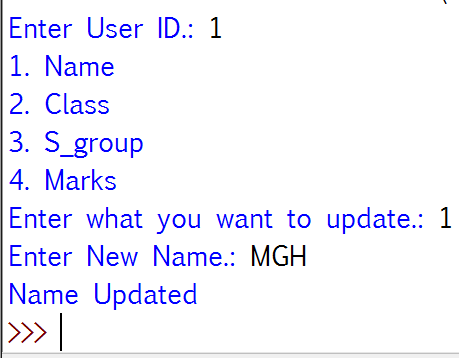
print('Marks Updated')

conn.commit()

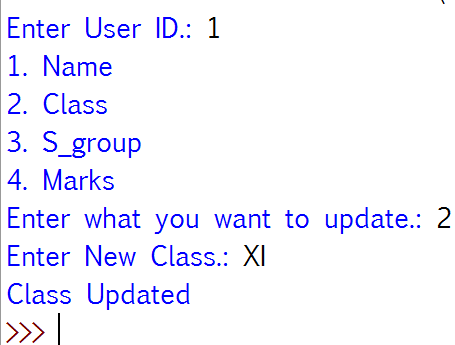
Update()

**Output:**

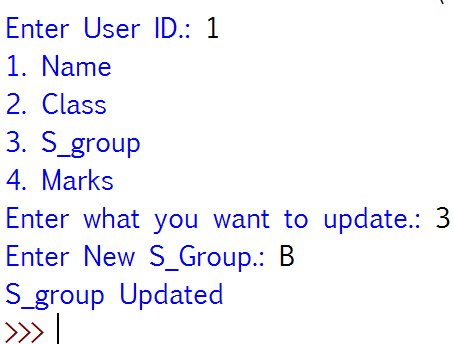
Updating Name



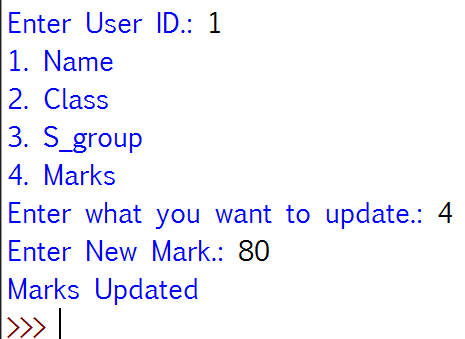
Updating Class



Updating S\_Group



Updating Marks



4.Searching Record

**Source code:**

def Search():

print('1. Search All')

print('2. Search Particular record')

a=int(input('Enter what you want to do.: '))

if a==1:

curs.execute('Select \* from students')

data=curs.fetchall()

for i in data:

print(i)

elif a==2:

name=input('Enter Name.: ')

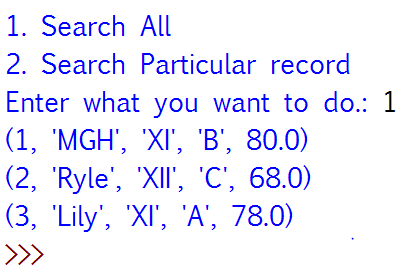
curs.execute('Select \* from students where name="{}"'.format(name))

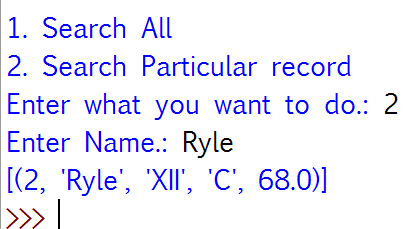
data=curs.fetchall()

print(data)

Search()

**Output:**

****

****