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

**INDEX**

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

**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**

**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:**

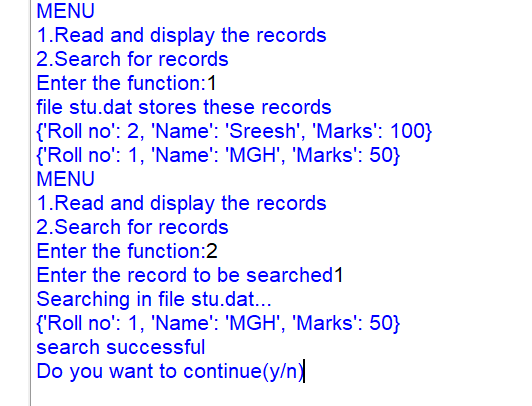
**read()**

**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**

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:

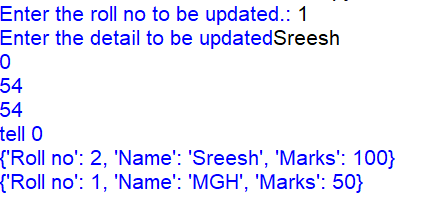
recs()

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)**

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:")

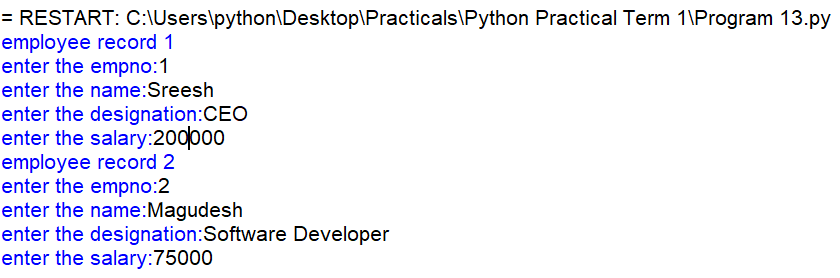
salary=int(input("enter the salary:"))

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**

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)")

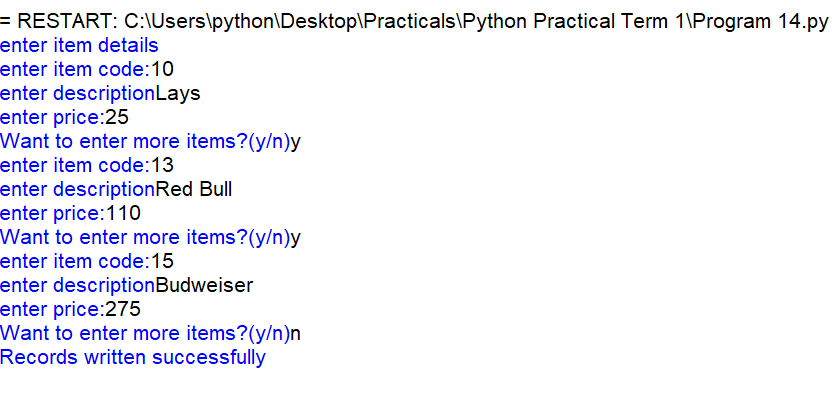
else:

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**

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()

from csv import reader

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:

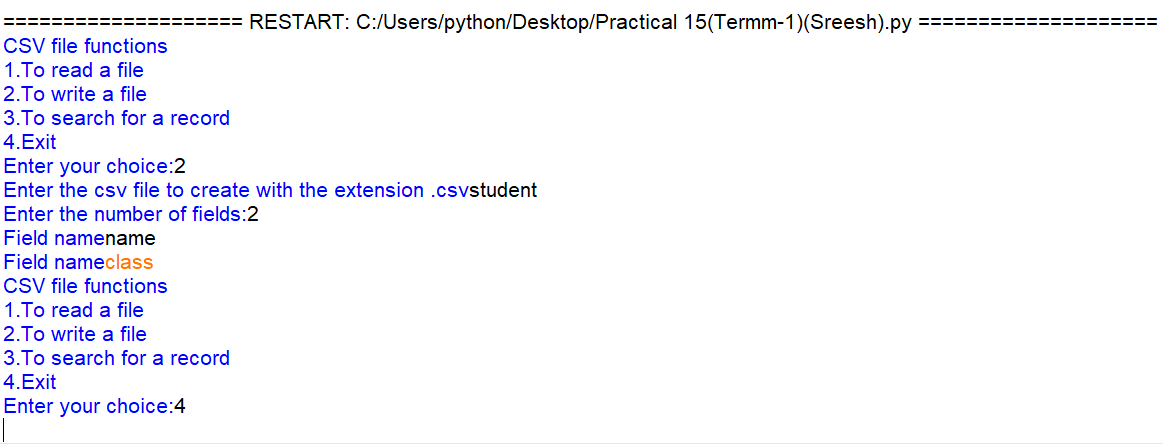
m=input("Enter the csv file in which you want the row to be searched with the extension .csv")

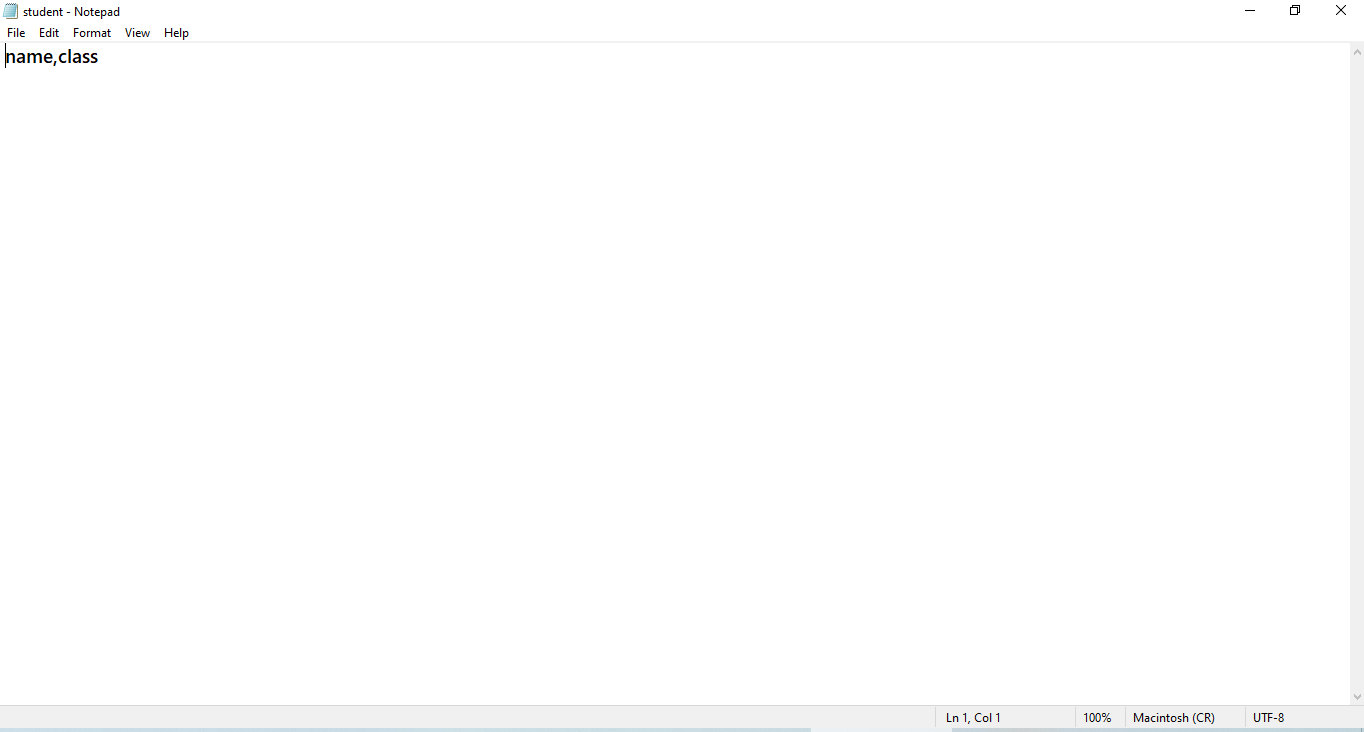
search(m)

if ch==4:

break

**Output:**

****

****

**Program 16: Encypting/Decrypting Messages**

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

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)

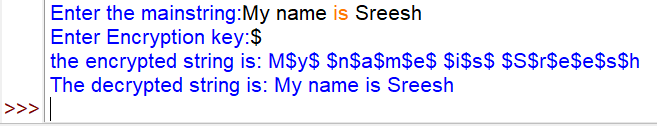
delst=decrypt(enstr,encryptstr)

destr="".join(delst)

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

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

**Output:**

****

**Program 17:Stacks**

**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()

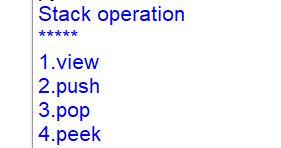
elif choice==4:

peek()

else:

print("wrong choice")

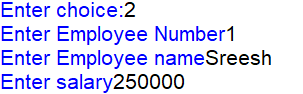
**Output:**

****

**View:**

****

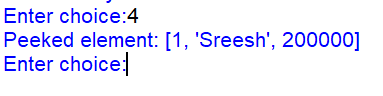
**Push:**

****

**Pop:**

****

**Peek:**

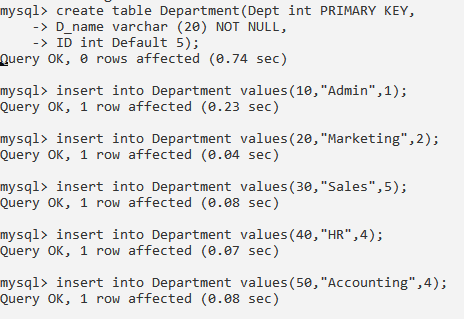
****

**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** |
| Dept | Integer | Primary key |
| D\_name | Varchar(20) | Not null |
| Salary | Float |  |
| Zone | Varchar(15) |  |
| Age | Integer | Greater than 20 |
| Grade | Char(1) |  |
| Dept | Integer | Foreign key(Department table) |

**Source code:**

****

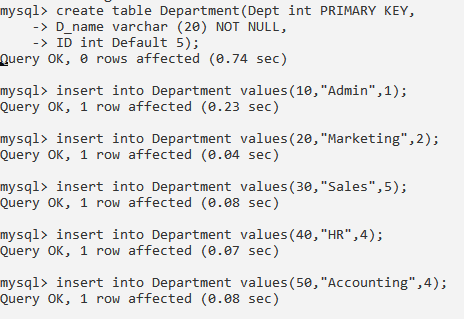
**Output:**

**Practical 1-Creating table and inserting record**

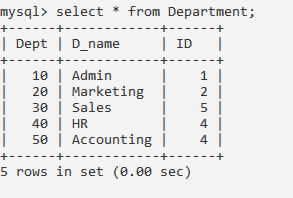
1.Create table department based on the following instance and populate the table.

|  |  |  |
| --- | --- | --- |
| **Column name** | **Data type** | **Constraints** |
| Dept | Integer | Primary key |
| D\_name | Varchar(20) | Not null |
| HOD | Integer | Default-5 |

**SOURCE CODE:**

****

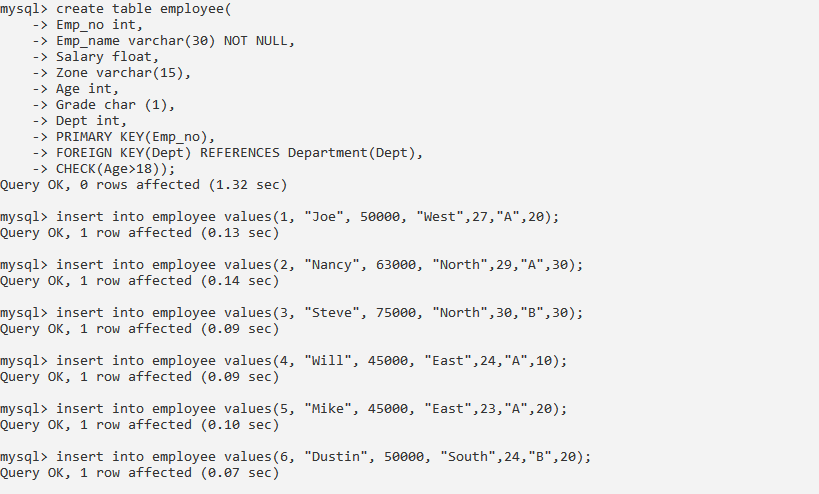
**OUTPUT:**

****

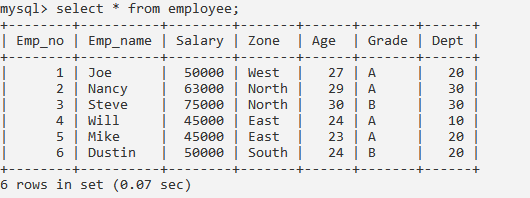
2.Create table Employee based on the following instance chart and populate the table.

|  |  |  |
| --- | --- | --- |
| **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:**

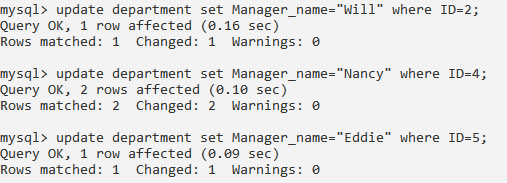


**OUTPUT:**

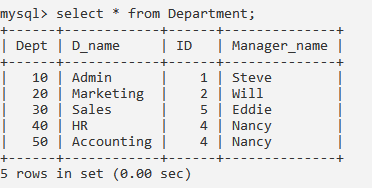
****

**Practical 2-To add column/s and update table**

**SOURCE CODE:**

****

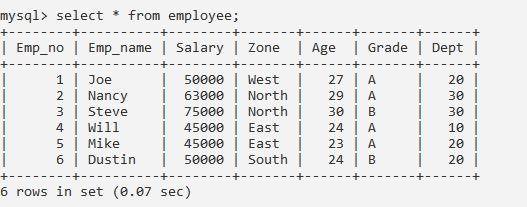
**OUTPUT:**

****

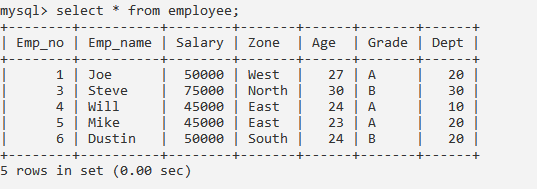
**Practical 3- To delete records based on criteria**

1.Delete all the records from Employee table belonging to North Zone and having Grade B

**SOURCE CODE:**

****

**OUTPUT:**

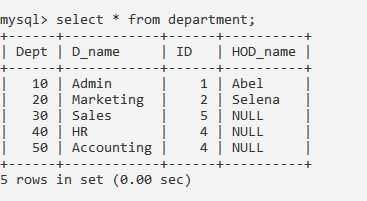
****

2.Delete column Manager\_name from table department.

**SOURCE CODE:**

****

**OUTPUT:**

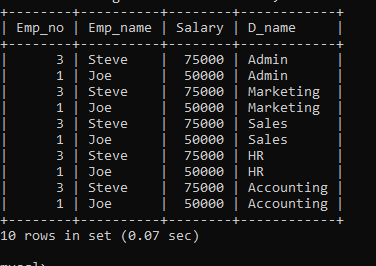
****

**Practical 4-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:**

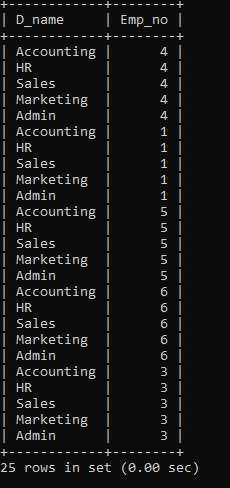
****

2.Display D\_name and corresponding Emp\_name from table Department and employee

**SOURCE CODE:**

****

**OUTPUT:**

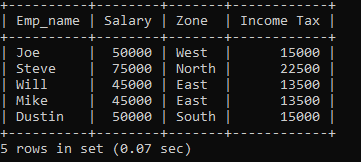
****

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

**SOURCE CODE:**

****

**OUTPUT:**

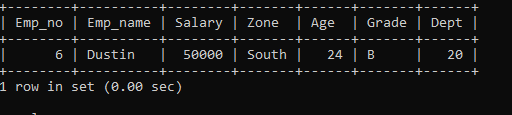
****

4.Display all details of employees of South zone whose salary is greater than 50000

**SOURCE CODE:**

****

**OUTPUT:**

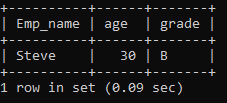
****

5.Display Emp\_name, age, grade of employees whose name starts with the character “S”

**SOURCE CODE:**



**OUTPUT:**

****

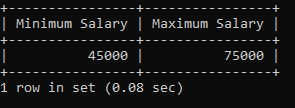
**Practical 5- Using functions**

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

**SOURCE CODE:**

****

**OUTPUT:**

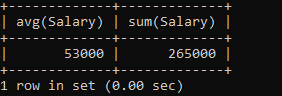
****

2.Display the average and total salary from table employee

**SOURCE CODE:**



**OUTPUT:**

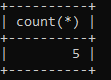
****

3.Count the number of records in the table employee.

**SOURCE CODE:**

****

**OUTPUT:**

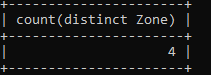
****

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

**SOURCE CODE:**



**OUTPUT:**



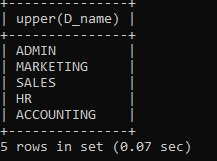
**Practical 6-Using functions**

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

**SOURCE CODE:**

****

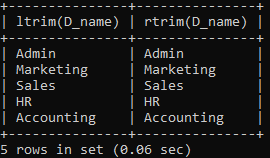
**OUTPUT:**

****

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

**SOURCE CODE:**

****

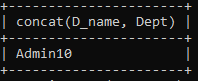
**OUTPUT:  
**

3.Concatenate D\_name and Dept of table Department having D\_name as ‘Computer’.

**SOURCE CODE:**

****

**OUTPUT:**

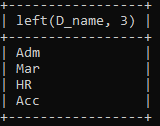
****

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

**SOURCE CODE:**

****

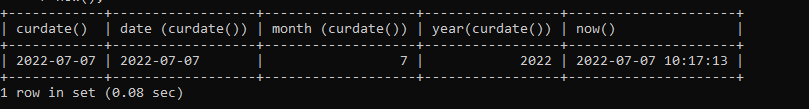
**OUTPUT:**

****

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

**SOURCE CODE:**

****

**OUTPUT:  
**

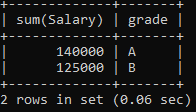
**Practical 7-Grouping data**

1.Display grade-wise total salary

**SOURCE CODE:**



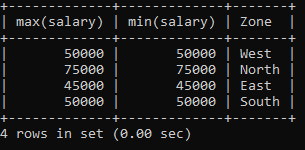
**OUTPUT:**



2.Display maximum and minimum salary in each zone.

**SOURCE CODE:  
**

**OUTPUT:**

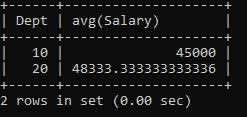
****

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

**SOURCE CODE:**

****

**OUTPUT:**

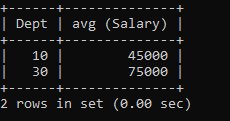
****

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

**SOURCE CODE:**

****

**OUTPUT:**

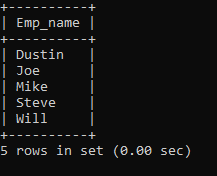
****

**Practical 8-Sorting data**

1.Display names from table employee in ascending order.

**SOURCE CODE:**

****

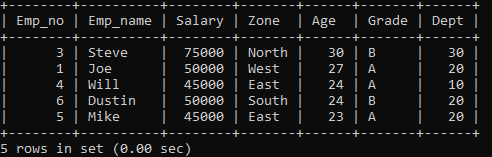
**OUTPUT:  
**

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

**SOURCE CODE:**

****

**OUTPUT:**

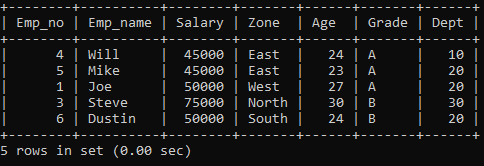
****

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

**SOURCE CODE:**

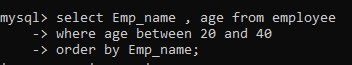
****

**OUTPUT:**

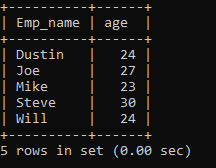
****

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

**SOURCE CODE:**

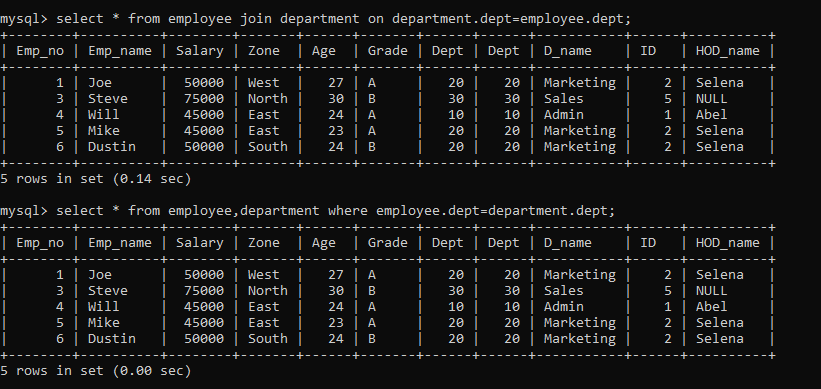
****

**OUTPUT:**

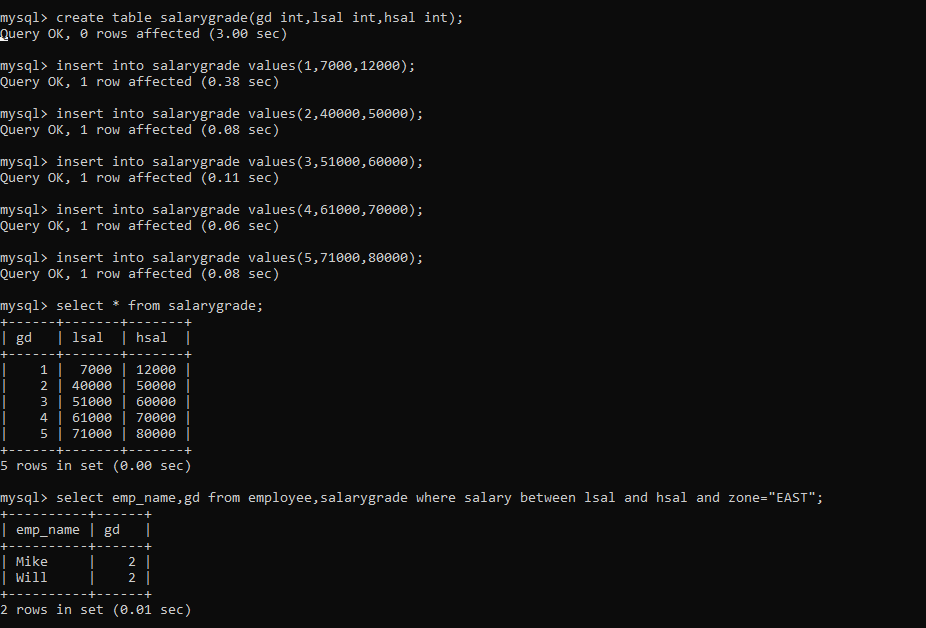
****

**Practical-9**

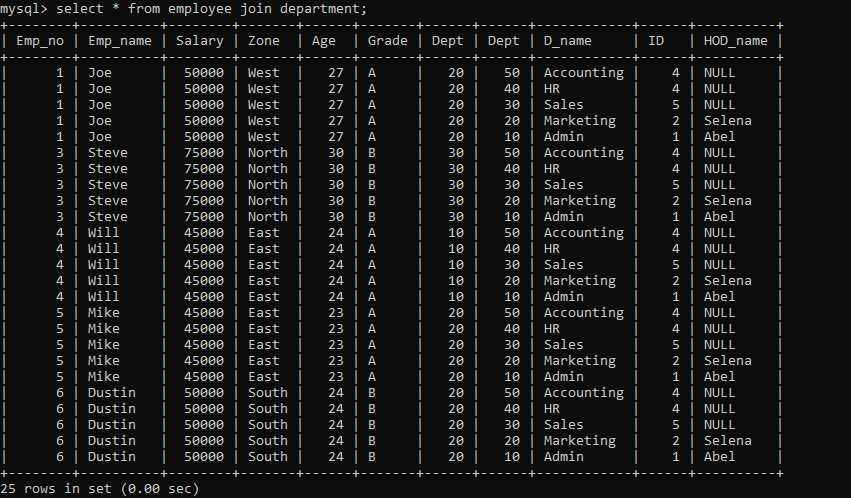
**Equi-join**

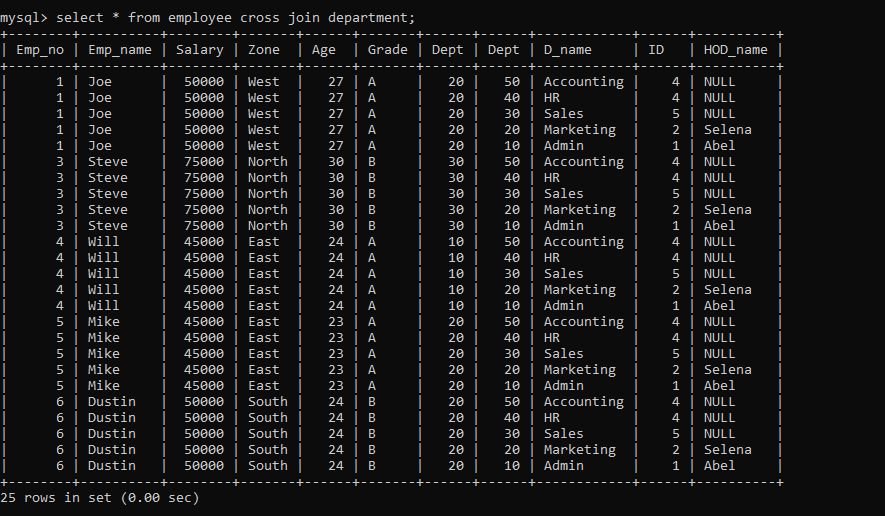
****

**Non-equi join**

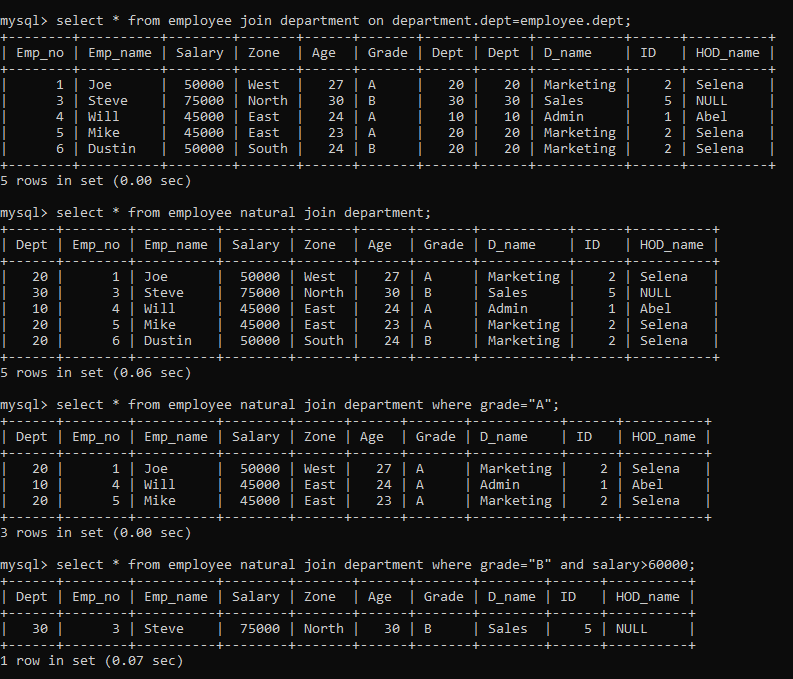
****

**Cartesian product**

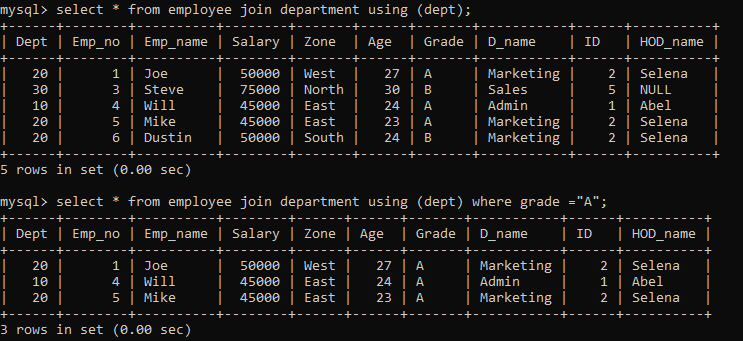
****

****

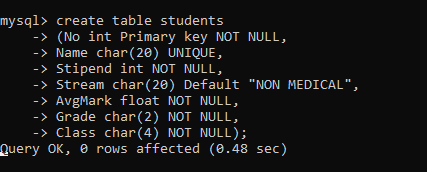
**Natural join**

****

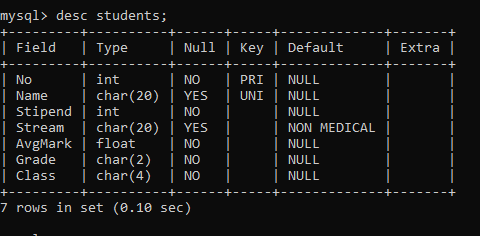
**Using clause instead of on**

****

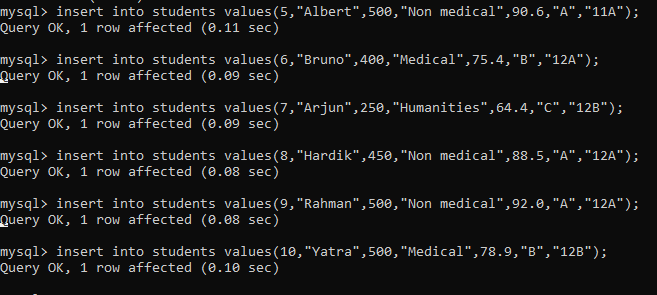
**CREATE TABLE WITH CONSTRAINT**

****

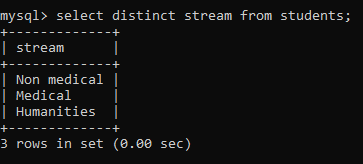
**DESC TABLE**



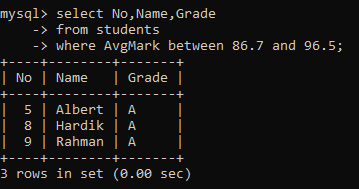
**INSERTING VALUES**

****

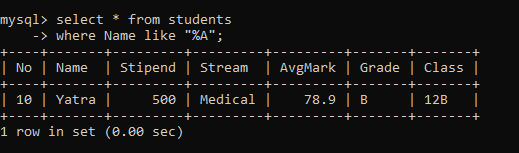
**SELECT(DISTINCT)**

****

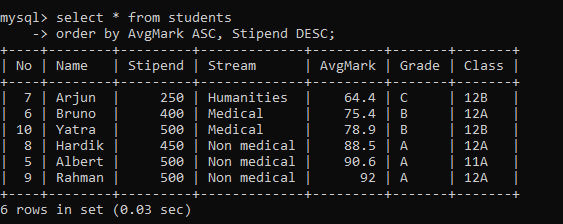
**WHERE CLAUSE USING(BETWEEN)**

****

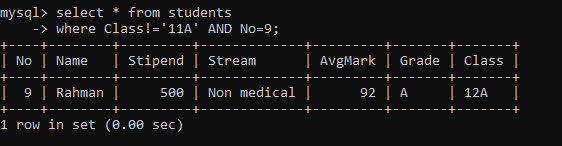
**WHERE CLAUSE USING(LIKE)**

****

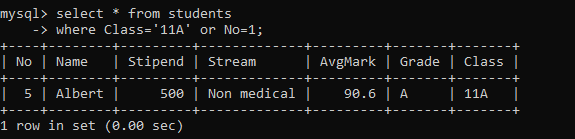
**WHERE CLAUSE USING(ASC AND DESC)**

****

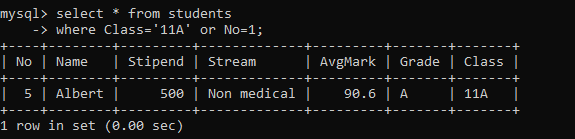
**WHERE CLAUSE USING ‘!=’**

****

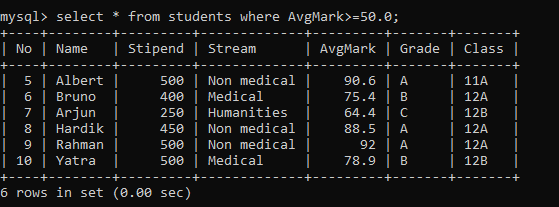
**WHERE CLAUSE USING ‘AND’**

****

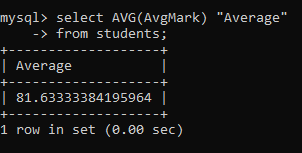
**WHERE CLAUSE USING ‘OR’**

****

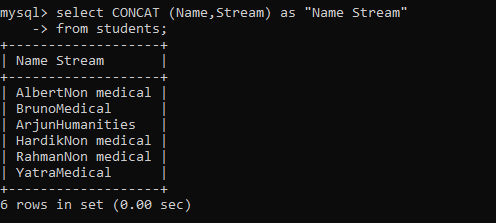
**WHERE CLAUSE USING (RELATIONS)**

****

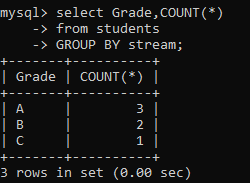
**USING FUNCTION(AVG)**

****

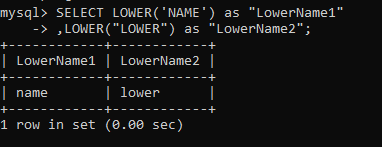
**USING FUNCTION(CONCAT)**

****

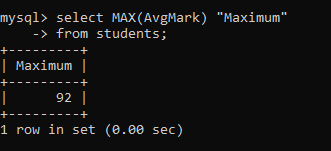
**USING FUNCTION(COUNT)**

****

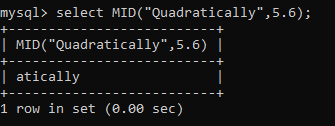
**USING FUNCTION(LOWER())**

****

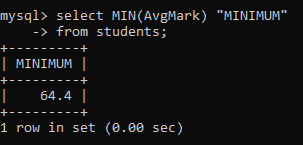
**USING FUNCTION(MAX())**

****

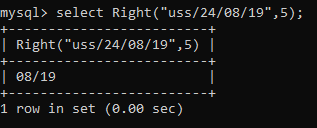
**USING FUNCTION(MID())**

****

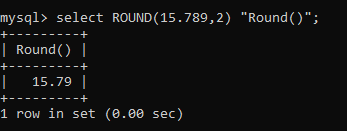
**USING FUNCTION(MIN())**

****

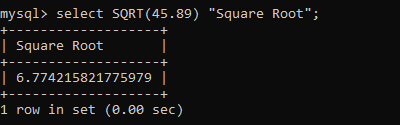
**USING FUNCTION(RIGHT)**

****

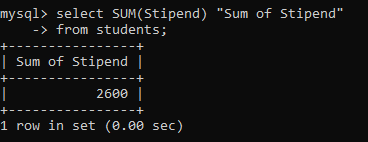
**USING FUNCTION(ROUND())**

****

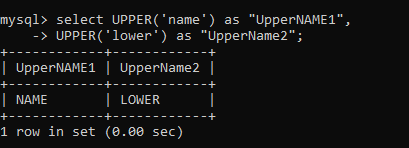
**USING FUNCTION(SQRT())**

****

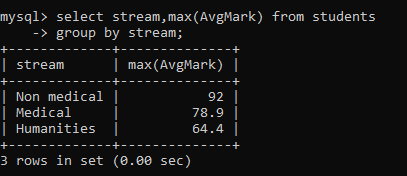
**USING FUNCTION(SUM())**

****

**USING FUNCTION(UPPER())**

****

**GROUP BY:**

****