Sanskriti School

Dr. S. Radhakrishnan Marg
Chanakyapuri
New Delhi - 110021

Python

Practical File

Term - 1

Submitted By:
Daksh Verma - XII B

Index

13	.Write a program to enter the following records in a binary file: Item No integer Item_Name string Qty integer Price float Number of records to be entered should be accepted from the user. Ask the user to input an Item No and delete the corresponding record from the file after displaying its contents in the following format and asking for confirmation: Item No: Item Name: Quantity: Price per item: Amount: After deletion, it should display the number of records still remaining in the file.	22
14	.Write a menu-driven program implementing user-defined functions to perform different functions on a csv file "employee" such as: (a)Write a single record to csv (b) Write all the records in one single go onto the csv. (c)Display the contents of the csv file.	23
15	.Write a program to create a CSV file to store usernames and passwords of 10 users. Now accept a username and password from the user and validate it from the file. Write functions (a)to create the file and add records to it (b) search for inputted username and password combination	24

16	A csv file is supposed to contain the following data (data only, no headings): RNo – integer Name – string Theory Marks – float Prac marks – float Write a menu driven program to perform the following operations on the file: (i) Create the file if it does not already exist (ii) Append record(s) to the file (iii) Display all the data from the file along with Total and Result of each student. Total is to be calculated as Theory+Prac, Result is "PASS" if Total is above 40, otherwise the Result is "FAIL". (iv) Increase the theory marks by 5 for all those students whose theory marks are between 35 and 41 (Excluding both). (v) Display the result summary in the following format: ● Students appeared: Total number of records in the file ● Max Total: Highest of total marks ● Min Total: Lowest of total marks ● Average Total: Average of total marks ● PASS: Total number of passing students ● FAIL: Total number of failed students ● PASS%: (PASS/Students appeared)*100	25
17	17.Create a products table(Pno, Pname, Rate, QOH) in MySQL and insert data. Implement the following SQL commands on the products table: (a)ALTER table to i. add new attribute - ROH type integer ii. modify data type- of Rate from integer to float iii. drop attribute- QOH from the table (b) Increase the rate of all products by 10% (c) Display data in ascending / descending order of Pname (d) Remove tuple(s) where Pname begins with 'T'. (e)Display the min, max, sum, count and average of the records based on field Rate	27
18	18. Refer to the tables given below and write queries that follow in MySQL. Table : Doctors DocID | DocName | Department | NoofOpdDays 101 | J K Mishra | Ortho | 3 102 | Mahesh tripathi | ENT | 4 103 | Ravi Kumar | Neuro | 5 104 | Mukesh Jain | Physio | 3 Table : Patients PatNo | PatName | Department | DocId 1 | Payal | ENT | 102 2 | Naveen | Ortho | 101 3 | Rakesh | Neuro | 103 4 | Atul | Physio | 104 (i) To display PatNo, PatName and corresponding DocName for each patient. (ii) To display the list of all doctors whose NoofOpdDays are more than 2. (iii) To display DocName, Department and PatName and DocId from both the tables. (iv) To display total no of different departments from Patients table. (v) To display all departments from Doctors table whose names end with 'o'.	30

19	19. Refer to the tables given below and write queries that follow in MySQL.	32
20	20. Consider the following tables STOCK and DEALERS and answer parts of this question:	35

FURNITURE

NO	ITEMNAME	TYPE	STOCKDATE	PRICE	DISCOUNT
1	Whitelotus	DoubleBed	23/02/02	30000	25
2	Pinkfeather	Babycot	20/01/02	7000	20
3	Dolphin	Babycot	19/02/02	9500	20
4	Decent	OfficeTable	01/01/02	25000	30
5	Comfortzone	DoubleBed	12/01/02	25000	25
6	Donald	Babycot	24/02/02	6500	15
7	RoyalFinish	OfficeTable	20/02/02	18000	30
8	Royaltiger	Sofa	22/02/02	31000	30
9	Econositting	Sofa	13/12/01	9500	25
10	EatingParadise	DiningTable	19/02/02	11500	25

ARRIVALS

NO	ITEMNAME	TYPE	STOCKDATE	PRICE	DISCOUNT
1	WoodComfort	DoubleBed	23/03/03	25000	25
2	OldFox	Sofa	20/02/03	17000	20
3	Micky	Babycot	21/02/03	7500	15

a) To show all information about the Baby cots from the FURNITURE table.

b) To list the ITEMNAME which are priced at more than 15000 from the FURNITURE table.

c) To list ITEMNAME and TYPE of those items, in which date of stock is before 22/01/02 from the FURNITURE table in descending of ITEMNAME.

d) To display ITEMNAME and DATEOFSTOCK of those items, in which the discount percentage is more than 25 from FURNITURE table.

e) To count the number of items, whose TYPE is "Sofa" from FURNITURE table.

f) To insert a new row in the ARRIVALS table with the following data: 14,"Velvet touch", "Double bed", {25/03/03}, 25000,30

g) Give the output of following SQL statements:

Note: Outputs of the above mentioned queries should be based on original data given in both the tables i.e., without considering the insertion done in (f) part of this question.

(i) Select COUNT(distinct TYPE) from FURNITURE;

(ii) Select MAX(DISCOUNT) from FURNITURE,ARRIVALS;

(iii) Select AVG(DISCOUNT) from FURNITURE where TYPE="Baby cot";

(iv) Select SUM(Price) from FURNITURE where STOCKDATE<{12/02/02};

Table:STOCK

ItemNo	Item	Dcode	Qty	UnitPrice	StockDate
5005	Ball Pen 0.5	102	100	16	31-Mar-10
5003	Ball Pen 0.25	102	150	20	01-Jan-10
5002	Gel Pen Premium	101	125	14	14-Feb-10
5006	Gel Pen Classic	101	200	22	01-Jan-09
5001	Eraser Small	102	210	5	19-Mar-09
5004	Eraser Big	102	60	10	12-Dec-09
5009	Sharpener Classic	103	160	8	23-Jan-09

Table: DEALERS

Dcode	Dname
101	Reliable Stationers
103	Classic Plastics
102	Clear Deals

Write SQL commands for the following statements:

(i) To display details of all Items in the Stock table in ascending order of StockDate.

(ii) To display ItemNo and Item name of those items from Stock table whose UnitPrice is more than 10.

(iii) To display the details of those items whose dealer code (Dcode) is 102 or Quantity in Stock (Qty) is more than 100 from the table Stock.

(iv) To display Maximum UnitPrice of items for each dealer individually as per Dcode from the table Stock.

21	21.Write SQL Commands for the following on the basis of the given table GRADUATE:	36
22	22.Write SQL queries for (a) to (d) and find outputs for SQL queries (e) to (h), which are based on the tables.	37

Table: GRADUATE

SL.No.	NAME	STIPEND	SUBJECT	AVERAGE	RANK
1	KARAN	400	PHYSICS	68	1
2	RAJ	450	CHEMISTRY	68	1
3	DEEP	300	MATHS	62	2
4	DIVYA	350	CHEMISTRY	63	1
5	GAURAV	500	PHYSICS	70	1
6	MANAV	400	CHEMISTRY	55	2
7	VARUN	250	MATHS	64	1
8	LIZA	450	COMPUTER	68	1
9	PUJA	500	PHYSICS	62	1
10	NISHA	300	COMPUTER	57	2

(i) List the names of those students who have obtained rank 1 sorted by NAME.

(ii) Display a list of all those names whose AVERAGE is greater than 65.

(iii) Display the names of those students who have opted COMPUTER as a SUBJECT with an AVERAGE of more than 60.

(iv) List the names of all the students in alphabetical order.

TRAINER

TID	Tname	City	HireDate	Salary
101	Sunaina	Mumbai	1998-10-15	90000
102	Anamika	Delhi	1994-12-24	80000
103	Deepti	Chandigarh	2001-12-21	82000
104	Meenakshi	Delhi	2002-12-25	78000
105	Richa	Mumbai	1996-01-12	95000
106	Maniprabha	Chennai	2001-12-12	69000

COURSE

CID	CNAME	FEES	STARTDATE	TID	
C201	AGDCA	12000	2018-07-02	101	
C202	ADCA	15000	2018-07-15	103	
C203	DCA	10000	2018-10-01	102	
C203	DDTP	9000	2018-09-15	104	
C205	DHN	20000	2018-08-01	101	
C206	O LEVEL	18000	2018-07-25	105	

a) Display the Trainer Name, City and Salary in descending order of their hiredate.

b) To display the TNAME and CITY of Trainer of joined the institute in the month of December 2001.

c) To display TNAME, HIREDATE, CNAME, STARTDATE from tables TRAINER and COURSE whose FEES is less than or equal to 10000

d) To display number of trainers from each city

e) SELECT TID, TNAME, FROM TRAINER WHERE CITY NOT IN('DELHI','MUMBAI');

f) SELECT DISTINCT TID FROM COURSE;

g) SELECT TID , COUNT (*), MIN(FEES) FROM COURSE GROUP BY TID HAVING COUNT(*)>1;

h) SELECT COUNT(*),SUM(FEES) FROM COURSE WHERE STARTDATE<'2018-09-15';

Sno.	Program

1	Write a random number generator that generates random numbers between 1 and 6 (simulates a dice).
	
	`import random`
	`def dice():`
	`    n=random.randint(1,6)`
	`    return(n)`
	
	`print(dice())`
	
	`4`

2	Write a function that takes a number n and then returns a randomly generated number having exactly n digits(not starting with zero) e.g. if n is 2 then function can randomly return a number 10-99 but 07,02 etc. are not valid two digit numbers.

```
import random
def randomnum(n):
    p = 10**(n-1)
    q= (10**(n))-1
    return(random.randint(p,q))

n = int(input("Number of digits : "))
print(randomnum(n))
```

```
Number of digits : 3
112
```

3

Create a modular program called matrix.py. It should contain functions which

- Return the sums of row elements as a tuple
- Return the Sums of column elements as a tuple
- Return Sums of diagonal elements
- Return transpose of a matrix.
- Takes two matrices as parameters and returns their sum
- Takes two matrices as parameters and returns their difference

For all functions it should accept square matrices as arguments. Demonstrate all functions by incorporating appropriate function calls in main program.

```
def sumrows(matrix):
    sumr=[]
    for i in matrix:

        sumr.append(sum(i))

    sumr=tuple(sumr)
    return(sumr)

print(sumrows([[1,2,3],[4,5,6],[7,8,9]]))
```

```
(6, 15, 24)
```

```
def sumcolumns(matrix):
    sumc=[]
    lis=[]
    rows=len(matrix)
    columns=len(matrix[0])
    for i in range(rows):

        for j in range(columns):

            lis.append(matrix[i][j])

    for i in range(rows):
        n=0
        for j in range(i,len(lis),columns):
            n+=lis[j]
        sumc.append(n)
    return(tuple(sumc))
print(sumcolumns([[1,2,3],[4,5,6],[7,8,9]]))
```

```
(12, 15, 18)
```

```
def sumdiag(matrix):
    sumd=0
    for i in range(len(matrix)):

        for j in range(len(matrix[i])):
            if i==j:
                sumd+=((matrix[i])[j])

    return(sumd)

print(sumdiag([[1,2,3],[4,5,6],[7,8,9]]))
```

```
15
```

```
def transpose(matrix):
    sumc=[]
    lis=[]
    rows=len(matrix)
    columns=len(matrix[0])
    for i in range(rows):

        for j in range(columns):

            lis.append(matrix[i][j])
    for i in range(rows):
        n=[]
        for j in range(i,len(lis),columns):
            n.append(lis[j])
        sumc.append(n)
    return(sumc)



print(transpose([[1,2,3],[4,5,6],[7,8,9]]))
```

```
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

```
def summatrix(m1,m2):
    result=[]

    r1=len(m1)
    c1=len(m1[0])
    r2=len(m2)
    c2=len(m2[0])

    if r1!=r2 or c1!=c2:
        print('The matrices cannot be added')

    else:
        for i in range(r1):
            lis=[]
            for j in range(c1):
                lis.append(m1[i][j] + m2[i][j])
            result.append(lis)

    return result

x= [[1,2,3],[4 ,5,6],[7 ,8,9]]
y = [[9,7,1],[6,5,4],[2,2,1]]
print(summatrix(x,y))
```

```
[[10, 9, 4], [10, 10, 10], [9, 10, 10]]
```

```
def difmatrix(m1,m2):
    result=[]

    r1=len(m1)
    c1=len(m1[0])
    r2=len(m2)
    c2=len(m2[0])

    if r1!=r2 or c1!=c2:
        print('Input not valid')

    else:
        for i in range(r1):
            lis=[]
            for j in range(c1):
                lis.append(m1[i][j] - m2[i][j])
            result.append(lis)

    return result

x= [[1,2,3],[4 ,5,6],[7 ,8,9]]
y = [[9,7,1],[6,5,4],[2,2,1]]
print(difmatrix(x,y))
```

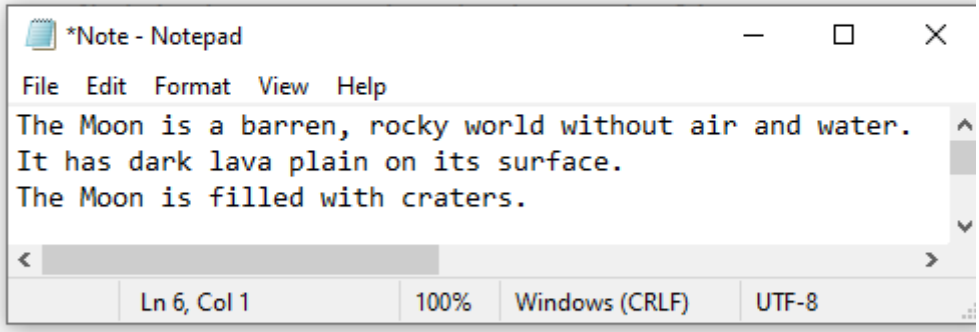
```
[[-8, -5, 2], [-2, 0, 2], [5, 6, 8]]
```

4	Read a text file line by line and display each word separated by a # symbol.

```
with open('Note.txt','r') as f1:
    i=f1.read()
    print(i.replace(' ','#'))
```

Shell

```
Python 3.7.9 (bundled)
```




```
>>> %Run 'Python Practical.py'

  The#Moon#is#a#barren,#rocky#world#without#air#and#water.#
  It#has#dark#lava#plain#on#its#surface.#
  The#Moon#is#filled#with#craters.
```

5	Read a text file and display the number of vowels/ consonants/ uppercase/ lowercase characters in the file.

*Note - Notepad

File Edit Format View Help

```
The Moon is a barren, rocky world without air and water.
It has dark lava plain on its surface.
The Moon is filled with craters.
```

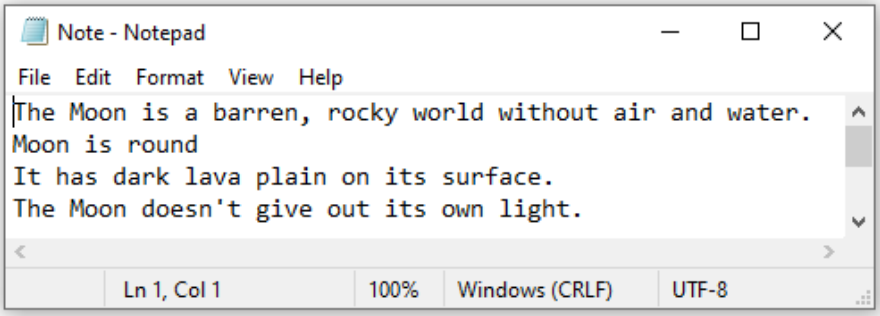
```
7  with open('Note.txt','r') as f1:
8      a=f1.read()
9
10     vowels=0
11     consonant=0
12     upper=0
13     lower=0
14     for i in a:
15         if i.isalpha() and i!='\n':
16             if i.lower() in ['a','e','i','o','u']:
17                 vowels+=1
18             if i.lower() not in ['a','e','i','o','u'] :
19                 consonant+=1
20             if i.isupper():
21                 upper+=1
22             if i.islower():
23                 lower+=1
24
25     print('Number of vowels            :',vowels)
26     print('Number of vowels            :',consonant)
27     print('Number of uppercase letters :',upper)
28     print('Number of lowercase letters :',lower)
29
30
31
```

Shell

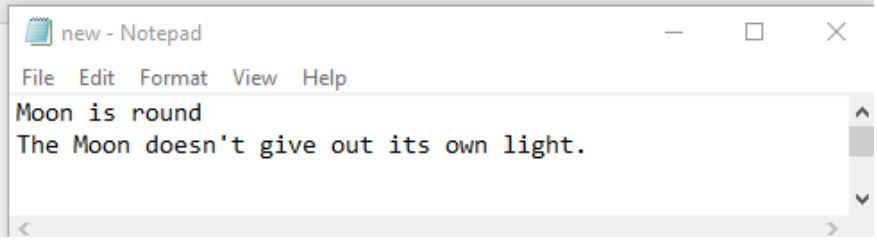
```
>>> %Run 'Python Practical.py'

  Number of vowels            : 38
  Number of vowels            : 62
  Number of uppercase letters : 5
  Number of lowercase letters : 95
```

6	Remove all the lines that contain the character `a' in a file and write it to another file.

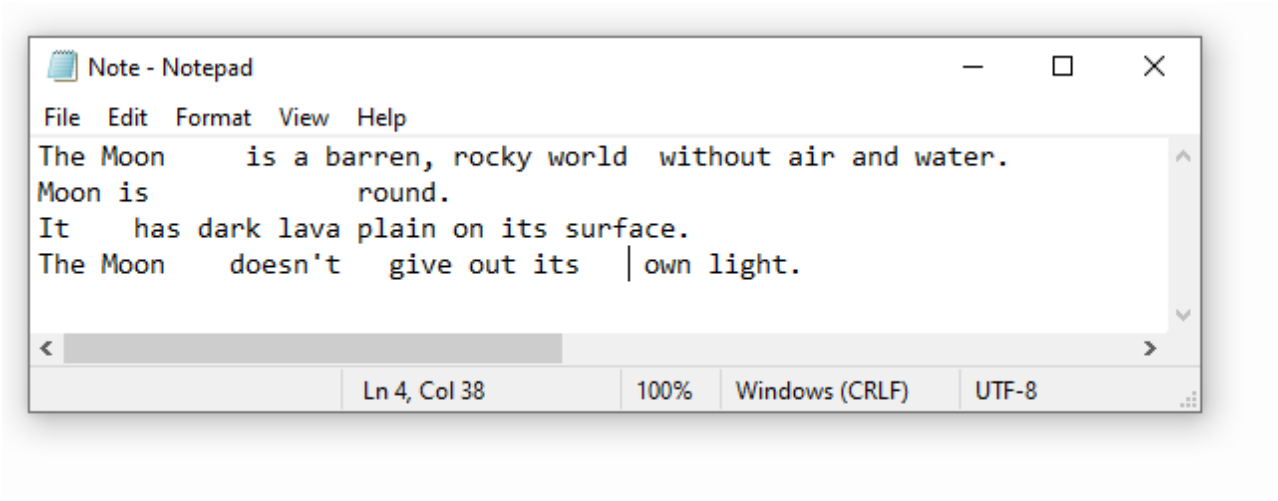
```
with open('Note.txt','r') as f1:
    a=f1.readlines()

    new=[]
    for i in a:
        count=0
        for j in i:
            if j == 'a':
                count+=1

        if count==0:
            new.append(i)


    f2=open('new.txt','w')
    f2.writelines(new)
    f2.close()
```



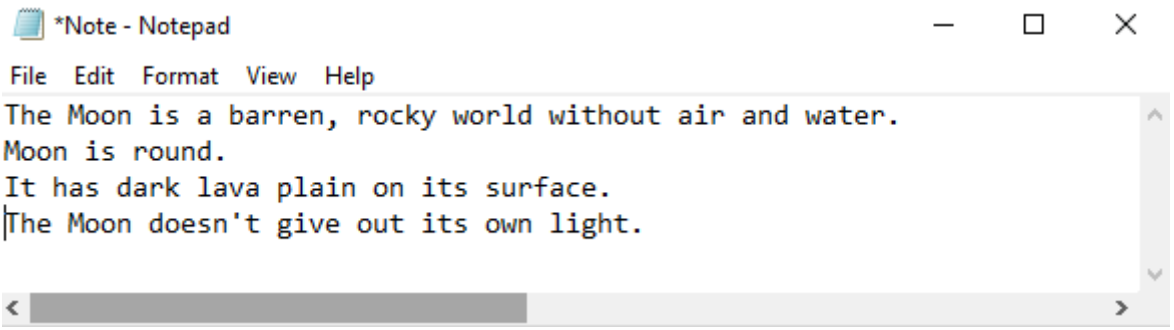

```
%Run 'Python Practical.py'
```



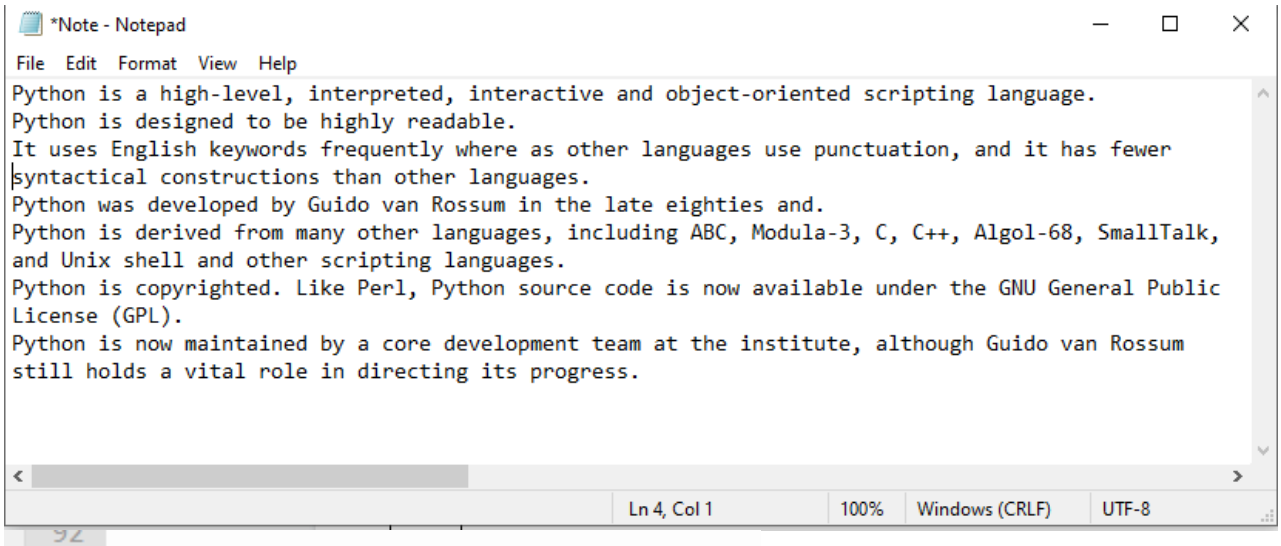
7 Read a text file line by line and replace all multiple consecutive spaces between words with single spaces.




```
69  f1 = open("Note.txt", "r")
70
71  read=f1.readlines()
72  lis=[]
73  for p in read:
74      string=''
75      a=p.split()
76      for i in a:
77          string+=i+' '
78      lis.append(string)
79
80  f1.close()
81  f2 = open("Note.txt", "w")
82  f2.writelines(lis)
83  f2.close()
84
```



8 Take a sample text file and find three most commonly occurring word(s) in it.




```
f1 = open("Note.txt", "r")
r=f1.read()
sp = r.split()
freq={}
for i in sp:
    if i not in freq:
        freq[i]=1
    else:
        freq[i]+=1

values=sorted(freq.values())

for i in range(-1,-4,-1):
    for k in freq:
        if freq[k] == values[i]:
            print(k,' : ',freq[k])
```

```
Shell
>>> %Run 'Python Practical.py'

  Python  :  7
  is  :  6
  and  :  6
```

9 Create a text file CRED.TXT and store 5 pairs of usernames and passwords in it. Then prompt the user to enter a username and password and validate it from the text file. If correct say “login successful” otherwise say “login denied”.

```
f1 = open("Cred.txt", "w")
for i in range(5):
    usr=input('Enter username:')
    pas=input('Enter password:')
    inp=usr+':'+pas+':\n'
    f1.write(inp)

f1.close()
```

```
120 f1 = open("Cred.txt", "w")
121 for i in range(5):
122     usr=input('Enter username:')
123     pas=input('Enter password:')
124     inp=usr+':'+pas+'\n'
125     f1.write(inp)
126
127 f1.close()
128
```

Shell

```
>>> %Run 'Python Practical.py'

  Enter username:a
  Enter password:1
  Enter username:b
  Enter password:2
  Enter username:c
  Enter password:3
  Enter username:d
  Enter password:4
  Enter username:e
  Enter password:5
```

*Cred - Notepad

File Edit Format View Help

```
a:1:
b:2:
c:3:
d:4:
e:5:
```

```
136 f1 = open("Cred.txt", "r")
137 usr=input('Enter username:')
138 pas=input('Enter password:')
139 s=0
140 r= f1.readlines()
141 for i in r:
142     q=i.split(':')
143     if q[0]==usr and q[1]==pas:
144
145         print('login successful')
146         s=1
147
148 if s==0:
149     print('login denied')
150
151 f1.close()
```

Shell

```
>>> %Run 'Python Practical.py'

  Enter username:a
  Enter password:3
  login denied

>>> %Run 'Python Practical.py'

  Enter username:d
  Enter password:4
  login successful
```

10	Create a binary file with name, roll number and marks. Search for a given roll number and display the name, if not found display appropriate message. Also, Input a roll number and update the marks.

```
import pickle

def addrec():
    with open('no.dat','ab') as fobj:
        x = input('Want to add students?(y/n)')
        while x == 'y':
            cod = int(input('enter roll no.'))
            nam = input('enter name')
            sal = float(input('marks'))
            lis = [cod,nam,sal]
            pickle.dump(lis,fobj)
            x = input('continue(y/n)')
            while x != 'y' and x!='n':
                x = input('continue(y/n)')
```

```
def findroll():
    fobj = open('no.dat','rb')
    find = int(input('enter roll no.: '))
    try:
        while True:
            lis = pickle.load(fobj)
            if lis[0] == find:
                break
        print(lis[0], lis[1], lis[2],sep = '\t')
        fobj.close()
    except:
        print('student not found')
        fobj.close()
```

```
def editarec():
    roll=int(input('Enter the roll of the student '))
    marks=input('Enter the new marks of the student')
    frobj=open('no.DAT','rb')
    fwobj=open('TEMP.DAT','wb')
    found=0
    try:
        while True:
            emp=pickle.load(frobj)
            if emp[0]==roll:
                emp[2]=marks
                found=1
            pickle.dump(emp,fwobj)
    except:
        frobj.close()
        fwobj.close()
    if found==0: print('Roll not found')
    os.remove('no.DAT')
    os.rename('TEMP.DAT','no.DAT')
```

11	11.Following is the structure of each record in a data file named "PRODUCT.DAT"{"prod_code": value, "prod_desc": value, "stock": value} The values for prod_code and prod_desc are strings, and the value for stock is an integer. Write a function in PYTHON to update the file with a new value of stock. The stock and the product_code, whose stock is to be updated, are to be input during the execution of the function.

```
import pickle
f = open("PRODUCT.dat","rb")
datalst = [ ]
while True :
    try :
        data = pickle.load(f)
        datalst += [data]
    except EOFError :
        break

f.close()

f = open("PRODUCT.dat","wb")
stock = int(input("Enter Stock which is to update :-"))
for i in datalst :
    if stock == i["stock"] :
        pcode = input("Enter product code :- ")
        pdesc = input("Enter product desc :- ")
        i["product code"] = pcode
        i [ "product desc" ] = pdesc
        pickle.dump(datalst,f)

f.close()
```

12	Assuming the tuple Vehicle as follows: (vehicletype, no_of_wheels), Where vehicletype is a string and no_of_wheels is an integer. Write a function showfile() to read all the records present in an already existing binary file SPEED.DAT and display them on the screen, also count the number of records present in the file.

```
def showrec():
    fobj = open('SPEED.dat','rb')
    count=0
    try:
        while True:
            lis = pickle.load(fobj)
            print(*lis,sep = '\t')
            count+=1
    except:
        fobj.close()
    print('Number of records is', count)
```

13

Write a program to enter the following records in a binary file:
Item No integer Item_Name string Qty integer Price float Number of records to be entered should be accepted from the user.
Ask the user to input an Item No and delete the corresponding record from the file after displaying its contents in the following format and asking for confirmation: Item No: Item Name: Quantity: Price per item: Amount: After deletion, it should display the number of records still remaining in the file

```
import pickle
import os

def addrec():
    with open('no.dat','ab') as fobj:
        x = input('How many records to be added')
        for i in range(X):
            no = int(input('enter roll no.'))
            nam = input('enter name')
            quan = int(input('marks'))
            price=float(input('Enter price'))
            lis = [no,nam,quan,sal]
            pickle.dump(lis,fobj)
```

```
def deletearec():
    code=int(input('Enter the Item No.'))
    frobj=open('no.DAT','rb')
    fwobj=open('TEMP.DAT','wb')
    found=0
    count=0
    try:
        while True:
            emp=pickle.load(frobj)
            if emp[0]==code:
                found=1
                print(*emp,sep='\t')
                c=input('confirm(y/n)')
                if c == 'y':
                    continue
            count+1
            pickle.dump(emp,fwobj)
    except:
        frobj.close()
        fwobj.close()
    if found==0: print('Item not found')
    print('Number of records left is :',count)
    os.remove('no.DAT')
    os.rename('TEMP.DAT','no.DAT')
```

14	.Write a menu-driven program implementing user-defined functions to perform different functions on a csv file “employee” such as: (a)Write a single record to csv (b) Write all the records in one single go onto the csv. (c)Display the contents of the csv file.

```
import csv
def singlerec():
    with open('employee.csv' , 'a',newline="" ) as f1:
        mywriter = csv.writer(f1,delimiter=',')
        mywriter.writerow(["Emp No","Name","Sal"])
        e=int(input('Emp no: '))
        n=input('Name : ')
        s=input('Sal : ')
        mywriter.writerow([e,n,s])

def multirec():
    roll=input("Roll No.:")

    f1=open('employee.csv' , 'a')
    new=[]
    x = input('Want to add records?(y/n)')
        while x == 'y':
            e=int(input('Emp no: '))
            n=input('Name : ')
            s=input('Sal : ')
            lis = [e,n,s]
            new.append(lis)
            x = input('continue(y/n)')
            while x != 'y' and x!='n':
                x = input('continue(y/n)')
    r_obj=csv.writer(f1)
    r_obj.writerows(new)

def show():
    with open('student.csv' , 'r',newline="" ) as f1:
        r_obj=csv.reader(f1,delimiter="!")
        count=0
        for row in r_obj:
            print(*row, sep=' , ')
```

15	Write a program to create a CSV file to store usernames and passwords of 10 users. Now accept a username and password from the user and validate it from the file. Write functions (a)to create the file and add records to it (b) search for inputted username and password combination

```
import csv
def create():
    with open('user.csv' , 'w',newline="" ) as f1:
        for i in range(10):

            mywriter = csv.writer(f1,delimiter=',')
            mywriter.writerow(["User","Password"])
            u=(input('Username: '))
            p=input('Password : ')
            mywriter.writerow([u,p])
def find():
    with open('user.csv' , 'r',newline="" ) as f1:
        r_obj=csv.reader(f1,delimiter=",")
        count=0
        u=(input('Username: '))
        p=input('Password : ')
        for row in r_obj:
            if row==[u,p]:
                print(*row, sep=' , ')
                print('Access granted')
```

16

A csv file is supposed to contain the following data (data only, no headings):
RNo – integer
Name – string
Theory Marks – float
Prac marks – float
Write a menu driven program to perform the following operations on the file:
(i) Create the file if it does not already exist
(ii) Append record(s) to the file
(iii) Display all the data from the file along with Total and Result of each student.
Total is to be calculated as Theory+Prac, Result is “PASS” if Total is above 40, otherwise the Result is “FAIL”.
(iv) Increase the theory marks by 5 for all those students whose theory marks are between 35 and 41 (Excluding both).
(v) Display the result summary in the following format:

```
import csv
def create():
    with open('student.csv' , 'w',newline="" ) as f1:
        mywriter = csv.writer(f1,delimiter=',')
        r=int(input('Roll no: '))
        n=input('Name : ')
        tm=float(input('Theory Marks : '))
        pm=float(input('Practical Marks : '))
        mywriter.writerow([r,n,tm,pm])

def append():
    with open('student.csv' , 'a',newline="" ) as f1:
        n=int(input('Enter the number of records to be appended :'))
        for i in range(n):
            mywriter = csv.writer(f1,delimiter=',')
            r=int(input('Roll no: '))
            n=input('Name : ')
            tm=float(input('Theory Marks : '))
            pm=float(input('Practical Marks : '))
            mywriter.writerow([r,n,tm,pm])
```

```
def disp():

    with open('student.csv' , 'r' ) as f1:
        r_obj=csv.reader(f1,delimiter=",")
        print('RollNO \t  Name \t Theory Marks \t Practical Marks \t Total')
        for i in r_obj:
            if float(i[2])+float(i[3])>40:
                print((*i,sep='\t','t','pass'))
            else:
                print((*i,sep='\t','t','fail'))



def incmarks():
    lis=[]
    with open('student.csv' , 'r' ) as f1:
        r_obj=csv.reader(f1,delimiter=",")
        for i in r_obj:
            if i[2]>35 and i[2]<41:
                i[2]+=5

            lis.append(i)
    with open('student.csv' , 'w' ) as f1:
        mywriter=csv.writer(f1)
        mywriter.writerows(lis)



def result():
     with open('student.csv' , 'r' ) as f1:
        r_obj=csv.reader(f1,delimiter=",")
        print('Students Appeared :',len(r_obj))
        pas=0
        fail=0
        mn=10**10
        mx=0
        sm=0
        for i in r_obj:
            sm+=float(i[2])+float(i[3])
            if float(i[2])+float(i[3])>mx:
                mx=float(i[2])+float(i[3])

            if float(i[2])+float(i[3])<mn:
                mn=float(i[2])+float(i[3])

            if float(i[2])+float(i[3])>40:
                pas+=1
            if float(i[2])+float(i[3])<=40:
                fail+=1
        print('Max     :',mx)
        print('Min     :',mn)
        print('Average :',sm/(len(r_obj)))
        print('Pass    :',pas)
        print('Fail    :',fail)
        print('Pass%   :', pas/(len(r_obj))*100)
```

17

Create a products table(Pno, Pname, Rate, QOH) in MySQL and insert data. Implement the following SQL commands on the products table:
(a)ALTER table to i. add new attribute - ROH type integer ii. modify data type- of Rate from integer to float iii. drop attribute- QOH from the table
(b) Increase the rate of all products by 10%
(c)Display data in ascending / descending order of Pname
(d) Remove tuple(s) where Pname begins with 'T'.
(e)Display the min, max, sum, count and average of the records based on field Rate

```
create table Product ( Pno int(50), Pname char(30), Rate int(50), QOH int(50)) ;

>>
insert into product values
(1,'Pen',10,40),
(2,'Pencil',5,50);
```

Product

Pno	Pname	Rate	QOH
1	Pen	10	40
2	Pencil	5	50

A
i)

```
alter table product
add ROH int;
```

Product

Pno	Pname	Rate	QOH	ROH
1	Pen	10	40	
2	Pencil	5	50	

ii)

```
Alter table product modify Rate float(10,2);
```

iii)

```
alter table product
DROP QOH;
```

Product

Pno	Pname	Rate	ROH
1	Pen	10	
2	Pencil	5	

B)

```
update product
set rate=1.1*rate;
```

Product

Pno	Pname	Rate	ROH
1	Pen	11	
2	Pencil	5.5	

C)

```
select * from product order by Pname desc;
```

Output

Pno	Pname	Rate	ROH
2	Pencil	5.5	
1	Pen	11	

D)

Before

```
insert into product values(3,'Tyres',500,87);
```

Product

Pno	Pname	Rate	ROH
1	Pen	11	
2	Pencil	5.5	
3	Tyres	500	87

After

```
delete from product where pname like 't%';
```

Product

Pno	Pname	Rate	ROH
1	Pen	11	
2	Pencil	5.5	

E)

```
select min(Rate), max(rate), sum(rate) , count(rate) , avg(rate) from product;
```

Output

min(Rate)	max(rate)	sum(rate)	count(rate)	avg(rate)
5.5	11	16.5	2	8.25

18

(i) To display PatNo, PatName and corresponding DocName for each patient.
(ii) To display the list of all doctors whose NoofOpdDays are more than 2.
(iii) To display DocName, Department and PatName and DocId from both the tables.
(iv) To display total no of different departments from Patients table.
(v) To display all departments from Doctors table whose names end with ‘o’.

```
mysql> select * from doctors;
+-------+-----------------+------------+-------------+
| DocId | DocName         | Department | NoofOpdDays |
+-------+-----------------+------------+-------------+
|   101 | J K Mishra      | Ortho      |           3 |
|   102 | Mahesh Tripathi | ENT        |           4 |
|   103 | Ravi Kumar      | Neuro      |           5 |
|   104 | Mukesh Jain     | Physio     |           3 |
+-------+-----------------+------------+-------------+
4 rows in set (0.00 sec)
```

```
mysql> select * from patients;
+-------+---------+------------+-------+
| PatNo | PatName | Department | DocID |
+-------+---------+------------+-------+
|     1 | Payal   | ENT        |   102 |
|     2 | Naveen  | Ortho      |   101 |
|     3 | Rakesh  | Neuro      |   103 |
|     4 | Atul    | Physio     |   104 |
+-------+---------+------------+-------+
4 rows in set (0.00 sec)
```

i)

```
mysql> SELECT Patients.PatNo,Patients.PatName,
    -> Doctors.DocName
    -> FROM Patients ,Doctors
    -> WHERE  Doctors.DocID =Patients.DocID;
+-------+---------+-----------------+
| PatNo | PatName | DocName         |
+-------+---------+-----------------+
|     2 | Naveen  | J K Mishra      |
|     1 | Payal   | Mahesh Tripathi |
|     3 | Rakesh  | Ravi Kumar      |
|     4 | Atul    | Mukesh Jain     |
+-------+---------+-----------------+
4 rows in set (0.01 sec)
```

ii)

```
mysql> SELECT DocName from Doctors where NoofOpdDays>2;
+-----------------+
| DocName         |
+-----------------+
| J K Mishra      |
| Mahesh Tripathi |
| Ravi Kumar      |
| Mukesh Jain     |
+-----------------+
4 rows in set (0.01 sec)
```

iii)

```
mysql> SELECT Doctors.DocName,
    -> Doctors.Department,Patients.PatName, Doctors.DocID
    -> FROM Patients, Doctors
    -> WHERE Doctors.DocID =Patients.DocID;
+-----------------+------------+---------+-------+
| DocName         | Department | PatName | DocID |
+-----------------+------------+---------+-------+
| J K Mishra      | Ortho      | Naveen  |   101 |
| Mahesh Tripathi | ENT        | Payal   |   102 |
| Ravi Kumar      | Neuro      | Rakesh  |   103 |
| Mukesh Jain     | Physio     | Atul    |   104 |
+-----------------+------------+---------+-------+
4 rows in set (0.00 sec)
```

iv)

```
mysql> SELECT COUNT(DISTINCT DEPARTMENT) from Patients;
+----------------------------+
| COUNT(DISTINCT DEPARTMENT) |
+----------------------------+
|                          4 |
+----------------------------+
1 row in set (0.00 sec)
```

v)

```
mysql> SELECT Department from Doctors where Department like '%o';
+------------+
| Department |
+------------+
| Ortho      |
| Neuro      |
| Physio     |
+------------+
3 rows in set (0.00 sec)
```

19

Refer to the tables given below and write queries that follow in MySQL.
a) To show all information about the Baby cots from the FURNITURE table.
b) To list the ITEMNAME which are priced at more than 15000 from the FURNITURE table.
c) To list ITEMNAME and TYPE of those items, in which date of stock is before 22/01/02 from the FURNITURE table in descending order of ITEMNAME.
d) To display ITEMNAME and DATEOFSTOCK of those items, in which the discount percentage is more than 25 from FURNITURE table.
e) To count the number of items, whose TYPE is "Sofa" from the FURNITURE table.
f) To insert a new row in the ARRIVALS table with the following data: 14,“Velvet touch”, "Double bed", {25/03/03}, 25000,30
g) Give the output of following SQL statements:
Note: Outputs of the above mentioned queries should be based on original data given in both the tables i.e., without considering the insertion done in (f) part of this question.
(i) Select COUNT(distinct TYPE) from FURNITURE;
(ii) Select MAX(DISCOUNT) from FURNITURE,ARRIVALS;
(iii) Select AVG(DISCOUNT) from FURNITURE where TYPE="Baby cot";
(iv) Select SUM(Price) from FURNITURE where STOCKDATE<{12/02/02};

```
mysql> select * from furniture;
+------+----------------+-------------+------------+-------+----------+
| No   | ItemName       | Type        | Stockdate  | Price | Discount |
+------+----------------+-------------+------------+-------+----------+
|    1 | Whitehouse     | DoubleBed   | 2023-02-02 | 30000 |       25 |
|    2 | Pinkfeather    | Babycot     | 2020-01-02 |  7000 |       20 |
|    3 | Dolphin        | Babycot     | 2019-02-02 |  9500 |       20 |
|    4 | Decent         | OfficeTable | 2001-01-02 | 25000 |       30 |
|    5 | Comfortzone    | DoubleBed   | 2012-01-02 | 25000 |       25 |
|    6 | Donald         | Babycot     | 2024-02-02 |  6500 |       15 |
|    7 | RoyalFinish    | OfficeTable | 2020-02-02 | 18000 |       30 |
|    8 | RoyalTiger     | Sofa        | 2022-02-02 | 31000 |       30 |
|    9 | Econositting   | Sofa        | 2013-12-01 |  9500 |       25 |
|   10 | EatingParadise | DiningTable | 2019-02-02 | 11500 |       25 |
+------+----------------+-------------+------------+-------+----------+
10 rows in set (0.00 sec)
```

```
mysql> select * from arrivals;
+------+-------------+------------+------------+-------+----------+
| NO   | ItemName    | Type       | Stockdate  | Price | Discount |
+------+-------------+------------+------------+-------+----------+
|    1 | WoodComfort |  DoubleBed | 2023-03-03 | 25000 |       25 |
|    2 | OldFox      | Sofa       | 2020-02-03 | 17000 |       20 |
|    3 | Micky       | Babycot    | 2021-02-03 |  7500 |       15 |
+------+-------------+------------+------------+-------+----------+
```

a)

```
mysql> Select * from FURNITURE where Type='Babycot';
+------+-------------+---------+------------+-------+----------+
| No   | ItemName    | Type    | Stockdate  | Price | Discount |
+------+-------------+---------+------------+-------+----------+
|    2 | Pinkfeather | Babycot | 2020-01-02 |  7000 |       20 |
|    3 | Dolphin     | Babycot | 2019-02-02 |  9500 |       20 |
|    6 | Donald      | Babycot | 2024-02-02 |  6500 |       15 |
+------+-------------+---------+------------+-------+----------+
3 rows in set (0.00 sec)
```

b)

```
mysql> Select ITEMNAME from FURNITURE where PRICE>15000;
+-------------+
| ITEMNAME    |
+-------------+
| Decent      |
| Comfortzone |
| RoyalFinish |
| RoyalTiger  |
+-------------+
4 rows in set (0.00 sec)
```

c)

```
mysql> select stockdate from furniture;
+------------+
| stockdate  |
+------------+
| 2020-01-02 |
| 2019-02-02 |
| 2001-01-02 |
| 2012-01-02 |
| 2024-02-02 |
| 2020-02-02 |
| 2022-02-02 |
| 2013-12-01 |
| 2019-02-02 |
+------------+
9 rows in set (0.00 sec)

mysql> Select ITEMNAME, TYPE from FURNITURE where STOCKDATE<=2022-01-02 order by ITEMNAME desc;
Empty set, 1 warning (0.00 sec)
```

d)

```
mysql> Select ITEMNAME, STOCKDATE DATEOFSTOCK from FURNITURE where DISCOUNT>25;
+-------------+-------------+
| ITEMNAME    | DATEOFSTOCK |
+-------------+-------------+
| Decent      | 2001-01-02  |
| RoyalFinish | 2020-02-02  |
| RoyalTiger  | 2022-02-02  |
+-------------+-------------+
3 rows in set (0.00 sec)
```

e)

```
mysql> Select count(itemname) from FURNITURE where TYPE='Sofa';
+-----------------+
| count(itemname) |
+-----------------+
|               2 |
+-----------------+
1 row in set (0.00 sec)
```

f)

```
mysql> insert into FURNITURE values(14, 'Velvet Touch', 'Double bed', '2025-03-03', 25000,30)
    -> ;
Query OK, 1 row affected (0.00 sec)

mysql> select * from furniture where No=14;
+------+-------------+------------+------------+-------+----------+
| No   | ItemName    | Type       | Stockdate  | Price | Discount |
+------+-------------+------------+------------+-------+----------+
|   14 | Velvet Touch | Double bed | 2025-03-03 | 25000 |       30 |
+------+-------------+------------+------------+-------+----------+
1 row in set (0.00 sec)
```

g)

i)

```
mysql> Select COUNT(distinct TYPE) from FURNITURE;
+----------------------+
| COUNT(distinct TYPE) |
+----------------------+
|                    5 |
+----------------------+
1 row in set (0.00 sec)
```

ii)

```
mysql> Select MAX(Discount) from furniture, arrivals;
ERROR 1052 (23000): Column 'Discount' in field list is ambiguous
```

iii)

```
mysql> select avg(discount) from furniture where type='Babycot';
+---------------+
| avg(discount) |
+---------------+
|       18.3333 |
+---------------+
1 row in set (0.00 sec)
```

iv)

```
+------------+
| SUM(Price) |
+------------+
|      50000 |
+------------+
1 row in set (0.00 sec)
```

20

Write SQL commands for the following statements:
(i) To display details of all Items in the Stock table in ascending order of StockDate.
(ii) To display ItemNo and Item name of those items from the Stock table whose UnitPrice is more than 10.
(iii) To display the details of those items whose dealer code (Dcode) is 102 or Quantity in Stock (Qty) is more than 100 from the table Stock.
(iv) To display Maximum UnitPrice of items for each dealer individually as per Dcode from the table Stock.

i)

```
mysql> Select * from STOCK order by StockDate;
+--------+-------------------+-------+------+-----------+------------+
| ItemNo | Item              | Dcode | Qty  | UnitPrice | StockDate  |
+--------+-------------------+-------+------+-----------+------------+
|   5006 | Gel Pen Classic   |   101 |  200 |        22 | 2009-01-01 |
|   5009 | Sharpener Classic |   103 |  160 |         8 | 2009-01-09 |
|   5001 | Eraser Small      |   102 |  210 |         5 | 2009-03-19 |
|   5004 | Eraser Big        |   102 |   60 |        10 | 2009-12-12 |
|   5003 | Ball Pen 0.25     |   102 |  150 |        20 | 2010-01-01 |
|   5002 | Gel Pen Premium   |   101 |  125 |        14 | 2010-02-14 |
|   5005 | Ball Pen 0.5      |   102 |  100 |        16 | 2010-03-31 |
+--------+-------------------+-------+------+-----------+------------+
7 rows in set (0.00 sec)
```

ii)

```
mysql> Select ItemNo, Item from STOCK where UnitPrice>10;
+--------+-----------------+
| ItemNo | Item            |
+--------+-----------------+
|   5002 | Gel Pen Premium |
|   5003 | Ball Pen 0.25   |
|   5005 | Ball Pen 0.5    |
|   5006 | Gel Pen Classic |
+--------+-----------------+
4 rows in set (0.00 sec)
```

iii)

```
mysql> Select * from STOCK where Dcode=102 OR Qty<10;
+--------+---------------+-------+------+-----------+------------+
| ItemNo | Item          | Dcode | Qty  | UnitPrice | StockDate  |
+--------+---------------+-------+------+-----------+------------+
|   5001 | Eraser Small  |   102 |  210 |         5 | 2009-03-19 |
|   5003 | Ball Pen 0.25 |   102 |  150 |        20 | 2010-01-01 |
|   5004 | Eraser Big    |   102 |   60 |        10 | 2009-12-12 |
|   5005 | Ball Pen 0.5  |   102 |  100 |        16 | 2010-03-31 |
+--------+---------------+-------+------+-----------+------------+
4 rows in set (0.00 sec)
```

iv)

```
mysql> Select dcode, Max(UnitPrice) from STOCK Group By Dcode;
+-------+----------------+
| dcode | Max(UnitPrice) |
+-------+----------------+
|   101 |             22 |
|   102 |             20 |
|   103 |              8 |
+-------+----------------+
3 rows in set (0.00 sec)
```

21

Write SQL Commands for the following on the basis of the given table GRADUATE:
(i) List the names of those students who have obtained rank 1 sorted by NAME.
(ii) Display a list of all those names whose AVERAGE is greater than 65.
(iii) Display the names of those students who have opted COMPUTER as a SUBJECT with an AVERAGE of more than 60.
(iv) List the names of all the students in alphabetical order.

i)

```
mysql> select NAME from GRADUATE where RANK=1 ORDER by NAME;
+--------+
| NAME   |
+--------+
| DIVYA  |
| GAURAV |
| KARAN  |
| LIZA   |
| PUJA   |
| RAJ    |
| VARUN  |
+--------+
```

ii)

```
mysql> Select average, NAME from GRADUATE where AVERAGE>65;
+---------+--------+
| average | NAME   |
+---------+--------+
|      68 | KARAN  |
|      68 | RAJ    |
|      70 | GAURAV |
|      68 | LIZA   |
+---------+--------+
4 rows in set (0.00 sec)
```

iii)

```
mysql> Select NAME from GRADUATE where SUBJECT='COMPUTER' AND AVERAGE>60;
+------+
| NAME |
+------+
| LIZA |
+------+
1 row in set (0.00 sec)
```

iv)

```
mysql> SELECT NAME FROM GRADUATE ORDER BY NAME;
+--------+
| NAME   |
+--------+
| DEEP   |
| DIVYA  |
| GAURAV |
| KARAN  |
| LIZA   |
| MANAV  |
| NISHA  |
| PUJA   |
| RAJ    |
| VARUN  |
+--------+
10 rows in set (0.00 sec)
```

22

a) Display the Trainer Name, City and Salary in descending order of their hiredate.
b) To display the TNAME and CITY of Trainer of joined the institute in the month of December 2001.
c) To display TNAME, HIREDATE, CNAME, STARTDATE from tables TRAINER and COURSE whose FEES is less than or equal to 10000
d) To display number of trainers from each city
e) SELECT TID, TNAME, FROM TRAINER WHERE CITY NOT IN('DELHI','MUMBAI');
f) SELECT DISTINCT TID FROM COURSE;
g) SELECT TID , COUNT (*), MIN(FEES) FROM COURSE GROUP BY TID HAVING COUNT(*)>1;
h) SELECT COUNT(*),SUM(FEES) FROM COURSE WHERE STARTDATE<'2018-09-15';

a)

```
mysql> SELECT Tname, City, Salary from TRAINER Order by hiredate desc;
+------------+------------+--------+
| Tname      | City       | Salary |
+------------+------------+--------+
| Meenakshi  | Delhi      |  78000 |
| Deepti     | Chandigarh |  82000 |
| Maniprabha | Chennai    |  69000 |
| Sunaina    | Mumbai     |  90000 |
| Richa      | Mumbai     |  95000 |
| Anamika    | Delhi      |  80000 |
+------------+------------+--------+
6 rows in set (0.00 sec)
```

b)

```
mysql> Select Tname, City from TRAINER where month(HireDate)=12 and year(hiredate)=2001;
+------------+------------+
| Tname      | City       |
+------------+------------+
| Deepti     | Chandigarh |
| Maniprabha | Chennai    |
+------------+------------+
2 rows in set (0.00 sec)
```

c)

```
mysql> Select Tname, HireDate, CName from TRAINER, COURSE where TRAINER.TID=COURSE.TID and FEES<=10000;
+-----------+------------+-------+
| Tname     | HireDate   | CName |
+-----------+------------+-------+
| Anamika   | 1994-12-24 | DCA   |
| Meenakshi | 2002-12-25 | DDTP  |
+-----------+------------+-------+
2 rows in set (0.00 sec)
```

d)

```
mysql> Select Count(TID) from TRAINER group by City;
+------------+
| Count(TID) |
+------------+
|          1 |
|          1 |
|          2 |
|          2 |
+------------+
4 rows in set (0.00 sec)
```

e)

```
mysql> select TID, TNAME from Trainer where City not in ('Delhi', 'Mumbai');
+-----+------------+
| TID | TNAME      |
+-----+------------+
| 103 | Deepti     |
| 106 | Maniprabha |
+-----+------------+
2 rows in set (0.00 sec)
```

f)

```
mysql> SELECT DISTINCT TID FROM COURSE;
+------+
| TID  |
+------+
|  101 |
|  102 |
|  103 |
|  104 |
|  105 |
+------+
5 rows in set (0.00 sec)
```

g)

```
mysql> SELECT TID, COUNT(*), MIN(FEES) FROM COURSE GROUP BY TID HAVING COUNT(*)>1;
+------+----------+-----------+
| TID  | COUNT(*) | MIN(FEES) |
+------+----------+-----------+
|  101 |        2 |     12000 |
+------+----------+-----------+
1 row in set (0.00 sec)
```

h)

```
mysql> SELECT COUNT(*), SUM(FEES) FROM COURSE WHERE STARTDATE<'2018-09-15';
+----------+-----------+
| COUNT(*) | SUM(FEES) |
+----------+-----------+
|        5 |     74000 |
+----------+-----------+
1 row in set (0.00 sec)
```

23	Write a menu driven Python program to implement a stack using a list data-structure.

```
stack = [1,2,3,4,5,6,7,8]
print("SELECT APPROPRIATE CHOICE")
print("1. PUSH Element into the stack")
print("2. POP Element from the stack")
print("3. Display elements of the stack")

choice = int(input("Enter the choice:"))
if choice == 1:
    i = input('Enter element to be pushed')
    stack.append(i)

elif choice == 2:
    if len(stack) == 0:
        print('The stack is empty')
    else:
        print('Element POP out from the stack is:')
        print(stack.pop())
elif choice == 3:
    if len(stack) == 0:
        print('The stack is empty')
    else:
        print("The size of the stack is: ",len(stack))
        print('The stack elements are :')
        print(*stack,sep='\n')
```

```
SELECT APPROPRIATE CHOICE
1. PUSH Element into the Stack
2. POP Element from the Stack
3. Display Elements of the Stack
Enter the Choice:3
The Size of the STACK is:  8
STACK elements are as follows:
1
2
3
4
5
6
7
8
```

```
SELECT APPROPRIATE CHOICE
1. PUSH Element into the stack
2. POP Element from the stack
3. Display elements of the stack
Enter the choice:2
Element POP out from the stack is:
8
```

24	Write a menu driven program to perform insert and delete operations on a stack containing member details as given below: MemberNo Integer, MemberName String, Age Integer

```
stack = [[1,'A',20],[2,'B',40]]
print("SELECT APPROPRIATE CHOICE")
print("1. PUSH Element into the stack")
print("2. POP Element from the stack")
print("3. Display elements of the stack")

choice = int(input("Enter the choice:"))
if choice == 1:
    no = int(input('Enter MemberNo '))
    na= input('Enter MemberNo ')
    ag= int(input('Enter Age'))
    stack.append([no,na,ag])

elif choice == 2:
    if len(stack) == 0:
        print('The stack is empty')
    else:
        print('Element POP out from the stack is:')
        print(stack.pop())
elif choice == 3:
    if len(stack) == 0:
        print('The stack is empty')
    else:
        print("The size of the stack is: ",len(stack))
        print('The stack elements are :')
        for i in stack:
            print(*i,sep='    ')
```

```
SELECT APPROPRIATE CHOICE
1. PUSH Element into the stack
2. POP Element from the stack
3. Display elements of the stack
Enter the choice:3
The size of the stack is:  2
The stack elements are :
1    A    20
2    B    40
```

```
SELECT APPROPRIATE CHOICE
1. PUSH Element into the stack
2. POP Element from the stack
3. Display elements of the stack
Enter the choice:1
Enter MemberNo 3
Enter MemberNo C
Enter Age60
```

```
SELECT APPROPRIATE CHOICE
1. PUSH Element into the stack
2. POP Element from the stack
3. Display elements of the stack
Enter the choice:2
Element POP out from the stack is:
[3, 'C', 60]
```

25	Write a menu driven program in Python to connect to a MySQL database-company, and perform add, modify, delete operations on a table employee. Also it should give options to view all or selective records from the table. CODE :

```
import mysql.connector as con
mycon=con.connect(host='localhost',user='root',passwd='welcome',database='employee')
cur=mycon.cursor()


def add():
    cur.execute("create table if not exists employee (id int(2) primary key, name varchar(25), age
        int(3), dept varchar(25), salary int(70))")
    id=(input('Enter id'))
    name=input('Enter Name')
    name="'"+name+"'"
    age=(input('Enter age'))
    dept=input('Enter dept')
    dept="'"+dept+"'"
    salary=(input('Enter salary'))

    command=('Insert into employee values (%s,%s,%s,%s,%s);'%(id,name,age,dept,salary))
    cur.execute(command)
    mycon.commit()


def selectall():
    sql='select * from employee'
    cur.execute(sql)
    for i in cur:
        print('id:' ,i[0],'    \t name:',i[1],'\t    age;',i[2])



def select():
    name=input('Enter Name :')
    command= ("select * from employee where name = '"+name+"'")
    cur.execute(command)
    print(*cur,sep=' \n')
```

```
def modify():
    dept=input('Enter Department Name :')
    command= ("update employee set salary=salary*1.1 where dept = '"+dept+"'")
    cur.execute(command)
    mycon.commit()


def delete():
    id=input('Enter id :')
    command= ("Delete from employee where id = '"+id+"'")
    cur.execute(command)
    mycon.commit()




print('Do you want add (1), modify(2), delete(3), view all (4) or selective records(5)')
x=int(input('Enter Answer'))

if x==1:
    add()
if x==2:
    modify()
if x==3:
    delete()
if x==4:
    selectall()
if x==5:
    select()
```

```
What would you like to do
add[1]
delete[2]
modify[3]
1
enter Employee no.: 1
enter Employee name: Hi
enter Employee salary: 123
Eno     Name    Salary
1       Hi      123
do you want to continue (y/n): y
What would you like to do
add[1]
delete[2]
modify[3]
3
enter Employee no.: 1
enter Employee salary: 1234
Eno     Name    Salary
1       Hi      1234
do you want to continue (y/n): y
What would you like to do
add[1]
delete[2]
modify[3]
2
enter Employee no.: 1
Eno     Name    Salary
do you want to continue (y/n):
```