ABUDHABI INDIAN SCHOOL

BRANCH-1, AL WATHBA

COMPUTER SCIENCE PRACTICAL FILE (2022- 2023)

SUBMITTED BY:

NAME: SYED ASHIQ AMEER

CLASS: 12 SECTION: A_M

EXAM NO:

ABUDHABI INDIAN SCHOOL,

BRANCH-1, AL WATHBA

CERTIFICATE

Certified that the work entered in this file is the bonafide work of Master/Miss.____ of Class ____________ Section______________, Examination No. _______________________, during the academic year 2022-2023.

____________________ ____________________

Teacher In-Charge *Principal*

Date of Examination:_________________

Internal Examiner :____________________

External Examiner :_____________________

INDEX

PROGRAM 1

DATE: 21/3/22

AIM:

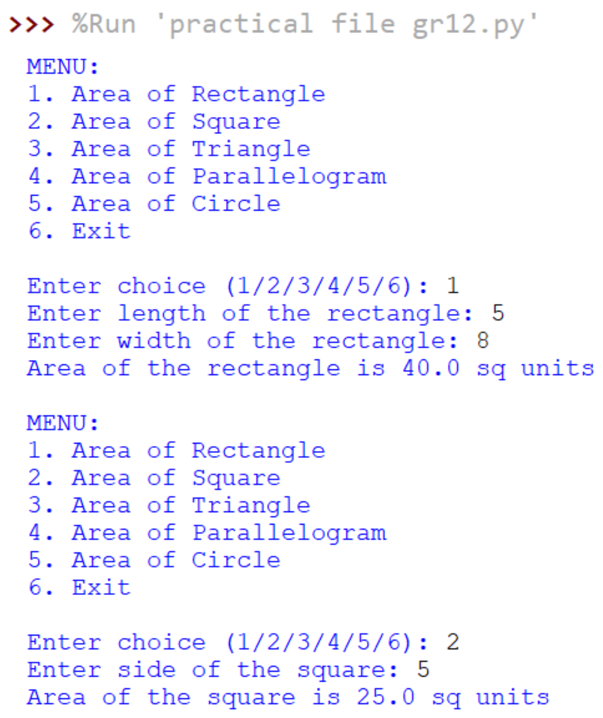
Create a menu driven program to calculate the area of the following shapes based on user’s choice:

1. RECTANGLE
2. SQUARE
3. TRIANGLE
4. PARALLELOGRAM
5. CIRCLE

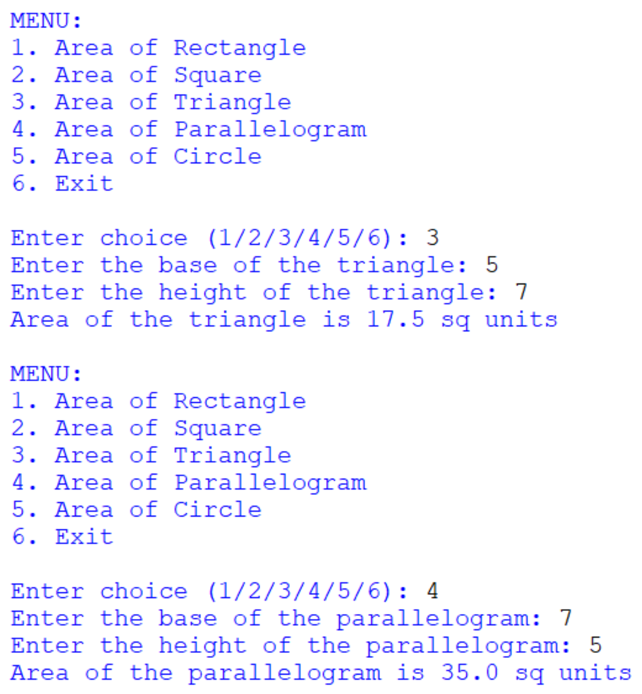
SOURCE CODE:

```
import math

def rectangle(l,b):
    ar = round(l*b,2)
    return ar

def square(s):
    ar = round(pow(s,2),2)
    return ar

def triangle(b,h):
    ar = round(0.5*b*h,2)
    return ar

def parallelogram(b,h):
    ar = round(b*h,2)
    return ar

def circle(r):
    ar = round(math.pi*pow(r,2),2)
    return ar

def menu():
    while True:
        print('MENU:')
        print('1. Area of Rectangle')
        print('2. Area of Square')
        print('3. Area of Triangle')
        print('4. Area of Parallelogram')
        print('5. Area of Circle')
        print('6. Exit')
        print()
        o=int(input('Enter choice (1/2/3/4/5/6): '))
        if o == 1:
            length = float(input('Enter length of the
rectangle: '))
            width = float(input('Enter width of the
rectangle: '))
            print('Area of the rectangle
is',rectangle(length,width),'sq units\n')
        elif o == 2:
            side = float(input('Enter side of the square:
'))
            print('Area of the square is',square(side),'sq
units\n')
        elif o == 3:
            base = float(input('Enter the base of the
triangle: '))
            height = float(input('Enter the height of the
triangle: '))
            print('Area of the triangle
is',triangle(base,height),'sq units\n')
        elif o == 4:
            base = float(input('Enter the base of the
parallelogram: '))
            height = float(input('Enter the height of the
parallelogram: '))
            print('Area of the parallelogram
is',parallelogram(base,height),'sq units\n')
        elif o == 5:
            radius = float(input('Enter the radius of the
circle: '))
            print('Area of the circle
is',circle(radius),'sq units\n')
        elif o == 6:
            print('Successfully Exited')
            break
        elif o == 7:
            print('Invalid Choice\n')


menu()
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 MENU:
 1. Area of Rectangle
 2. Area of Square
 3. Area of Triangle
 4. Area of Parallelogram
 5. Area of Circle
 6. Exit

 Enter choice (1/2/3/4/5/6): 1
 Enter length of the rectangle: 5
 Enter width of the rectangle: 8
 Area of the rectangle is 40.0 sq units

 MENU:
 1. Area of Rectangle
 2. Area of Square
 3. Area of Triangle
 4. Area of Parallelogram
 5. Area of Circle
 6. Exit

 Enter choice (1/2/3/4/5/6): 2
 Enter side of the square: 5
 Area of the square is 25.0 sq units
```



```
 MENU:
 1. Area of Rectangle
 2. Area of Square
 3. Area of Triangle
 4. Area of Parallelogram
 5. Area of Circle
 6. Exit

 Enter choice (1/2/3/4/5/6): 3
 Enter the base of the triangle: 5
 Enter the height of the triangle: 7
 Area of the triangle is 17.5 sq units

 MENU:
 1. Area of Rectangle
 2. Area of Square
 3. Area of Triangle
 4. Area of Parallelogram
 5. Area of Circle
 6. Exit

 Enter choice (1/2/3/4/5/6): 4
 Enter the base of the parallelogram: 7
 Enter the height of the parallelogram: 5
 Area of the parallelogram is 35.0 sq units
```

```
MENU:
1. Area of Rectangle
2. Area of Square
3. Area of Triangle
4. Area of Parallelogram
5. Area of Circle
6. Exit

Enter choice (1/2/3/4/5/6): 5
Enter the radius of the circle: 7
Area of the circle is 153.94 sq units

MENU:
1. Area of Rectangle
2. Area of Square
3. Area of Triangle
4. Area of Parallelogram
5. Area of Circle
6. Exit

Enter choice (1/2/3/4/5/6): 6
Successfully Exited
```

PROGRAM 2

DATE: 6/4/22

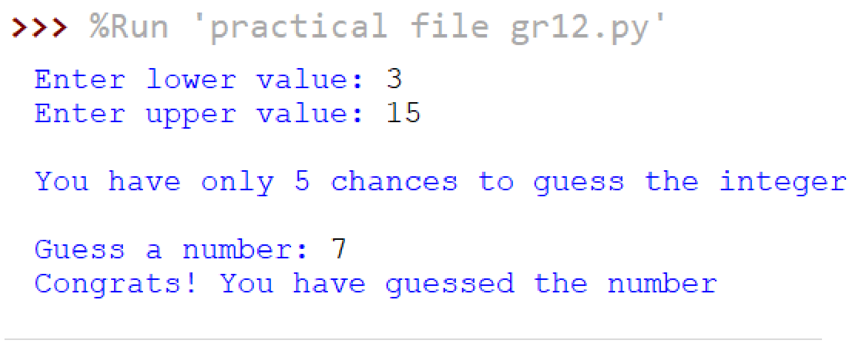
AIM:

Create Number Guessing Game in Python using random module.

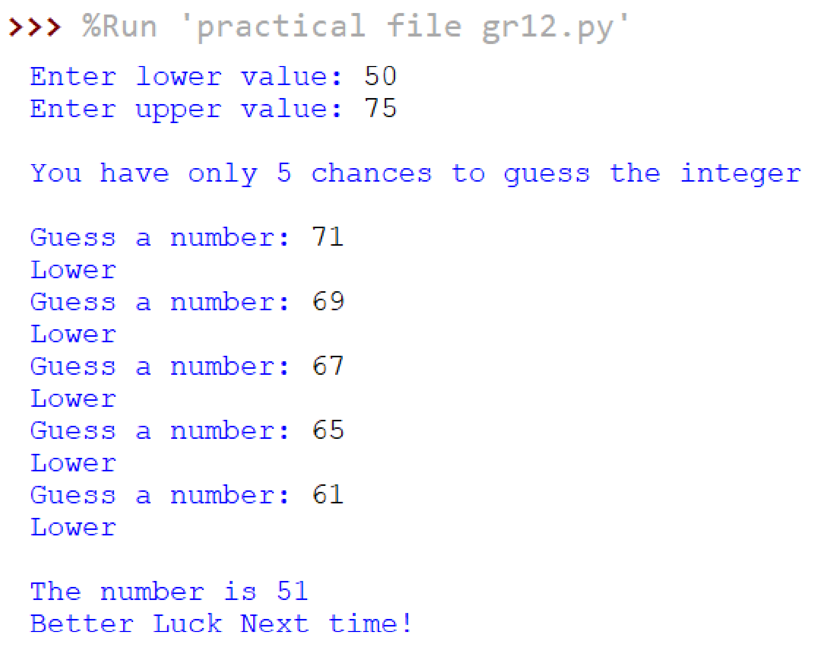
SOURCE CODE:

```
import random

lower = int(input('Enter lower value: '))
upper = int(input('Enter upper value: '))
no = random.randint(lower,upper)
print('\nYou have only 5 chances to guess the integer\n')
count = 0
win = False

while count < 5:
    count += 1
    guess = int(input('Guess a number: '))
    if no == guess:
        print('Congrats! You have guessed the number')
        win = True
        break
    elif no > guess:
        print('Higher')
    else:
        print('Lower')

if win == False:
    print('\nThe number is',no)
    print('Better Luck Next time!')
```

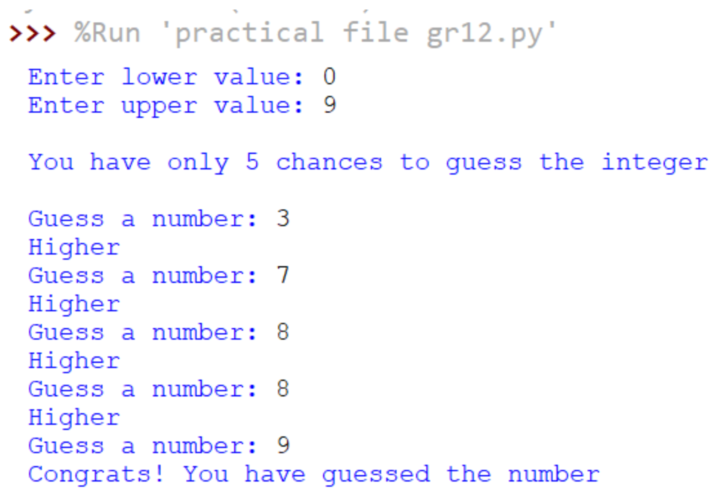
OUTPUT:



```
>>> %Run 'practical file gr12.py'
 Enter lower value: 3
 Enter upper value: 15

 You have only 5 chances to guess the integer

 Guess a number: 7
 Congrats! You have guessed the number
```



```
>>> %Run 'practical file gr12.py'
 Enter lower value: 50
 Enter upper value: 75

 You have only 5 chances to guess the integer

 Guess a number: 71
 Lower
 Guess a number: 69
 Lower
 Guess a number: 67
 Lower
 Guess a number: 65
 Lower
 Guess a number: 61
 Lower

 The number is 51
 Better Luck Next time!
```



```
>>> %Run 'practical file gr12.py'
 Enter lower value: 0
 Enter upper value: 9

 You have only 5 chances to guess the integer

 Guess a number: 3
 Higher
 Guess a number: 7
 Higher
 Guess a number: 8
 Higher
 Guess a number: 8
 Higher
 Guess a number: 9
 Congrats! You have guessed the number
```

Program 3

DATE: 11/4/22

AIM:

Write a Python program to accept a string from the user and create a menu to :-

a. Check whether it is palindrome or not.

b. Display number of characters,

c. No of words,

d. No.of lower case letters,

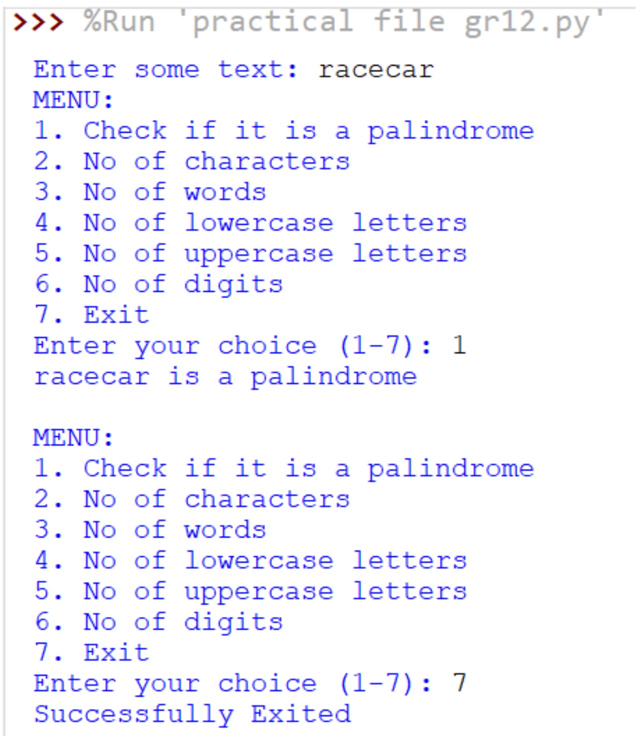
e. No. Of uppercase letters

f. No. of digits in the string.

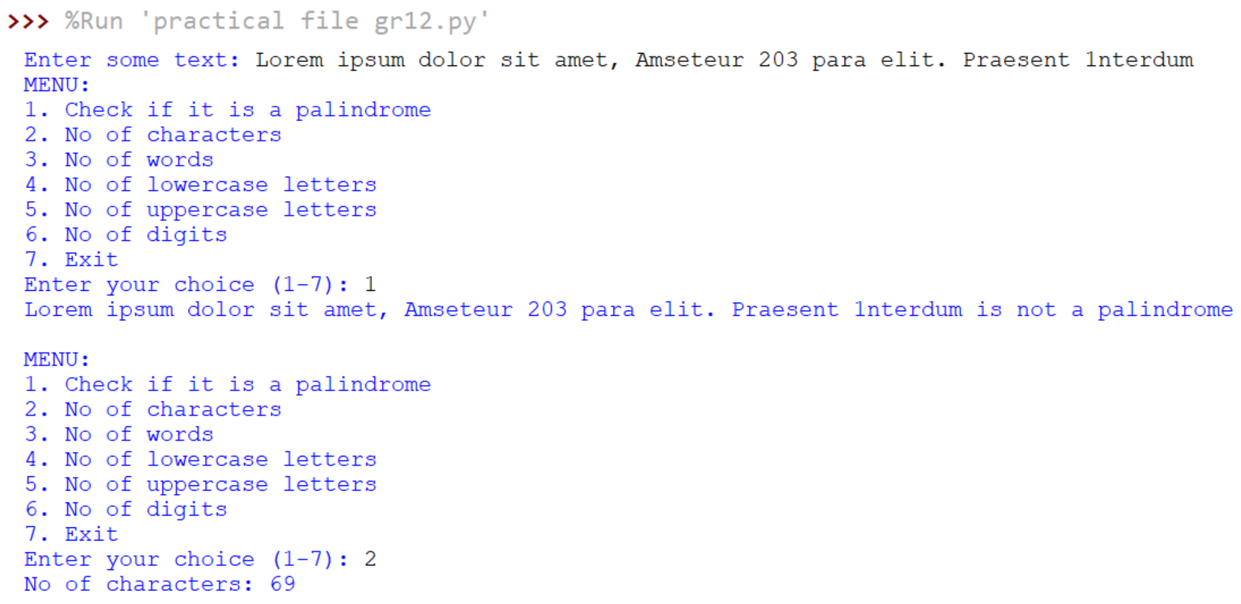
SOURCE CODE:

```
def palindrome(s):
    sp = s[::-1]
    if s == sp:
        return True
    else:
        return False

def char(s):
    c = len(s)
    return c

def words(s):
    split = s.split()
    c = len(split)
    return c

def lower(s):
    c = 0
    for i in s:
        if i.islower():
            c+=1
    return c

def upper(s):
    c = 0
    for i in s:
        if i.isupper():
            c+=1
    return c

def digits(s):
    c = 0
    for i in s:
        if i.isdigit():
            c+=1
    return c

def menu(s):
    while True:
        print('MENU:')
        print('1. Check if it is a palindrome')
        print('2. No of characters')
        print('3. No of words')
        print('4. No of lowercase letters')
        print('5. No of uppercase letters')
        print('6. No of digits')
        print('7. Exit')
        o = int(input('Enter your choice (1-7): '))
        if o == 1:
            if palindrome(s) == True:
                print(s,'is a palindrome\n')
            else:
                print(s,'is not a palindrome\n')
        elif o == 2:
            print('No of characters:',char(s),'\n')
        elif o == 3:
            print('No of words:',words(s),'\n')
        elif o == 4:
            print('No of lowercase 
letters:',lower(s),'\n')
        elif o == 5:
            print('No of uppercase 
letters:',upper(s),'\n')
        elif o == 6:
            print('No of digits:', digits(s),'\n')
        elif o == 7:
            print('Successfully Exited')
            break
        else:
            print('Invalid Option')
st = input('Enter some text: ')
menu(st)
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'

 Enter some text: racecar
 MENU:
 1. Check if it is a palindrome
 2. No of characters
 3. No of words
 4. No of lowercase letters
 5. No of uppercase letters
 6. No of digits
 7. Exit
 Enter your choice (1-7): 1
 racecar is a palindrome

 MENU:
 1. Check if it is a palindrome
 2. No of characters
 3. No of words
 4. No of lowercase letters
 5. No of uppercase letters
 6. No of digits
 7. Exit
 Enter your choice (1-7): 7
 Successfully Exited
```



```
>>> %Run 'practical file gr12.py'

 Enter some text: Lorem ipsum dolor sit amet, Amseteur 203 para elit. Praesent 1nterdum
 MENU:
 1. Check if it is a palindrome
 2. No of characters
 3. No of words
 4. No of lowercase letters
 5. No of uppercase letters
 6. No of digits
 7. Exit
 Enter your choice (1-7): 1
 Lorem ipsum dolor sit amet, Amseteur 203 para elit. Praesent 1nterdum is not a palindrome

 MENU:
 1. Check if it is a palindrome
 2. No of characters
 3. No of words
 4. No of lowercase letters
 5. No of uppercase letters
 6. No of digits
 7. Exit
 Enter your choice (1-7): 2
 No of characters: 69
```

```
MENU:
1. Check if it is a palindrome
2. No of characters
3. No of words
4. No of lowercase letters
5. No of uppercase letters
6. No of digits
7. Exit
Enter your choice (1-7): 3
No of words: 11

MENU:
1. Check if it is a palindrome
2. No of characters
3. No of words
4. No of lowercase letters
5. No of uppercase letters
6. No of digits
7. Exit
Enter your choice (1-7): 4
No of lowercase letters: 50

MENU:
1. Check if it is a palindrome
2. No of characters
3. No of words
4. No of lowercase letters
5. No of uppercase letters
6. No of digits
7. Exit
Enter your choice (1-7): 5
No of uppercase letters: 3

MENU:
1. Check if it is a palindrome
2. No of characters
3. No of words
4. No of lowercase letters
5. No of uppercase letters
6. No of digits
7. Exit
Enter your choice (1-7): 6
No of digits: 4

MENU:
1. Check if it is a palindrome
2. No of characters
3. No of words
4. No of lowercase letters
5. No of uppercase letters
6. No of digits
7. Exit
Enter your choice (1-7): 7
Successfully Exited
```

Program 4

DATE: 18/4/22

AIM:

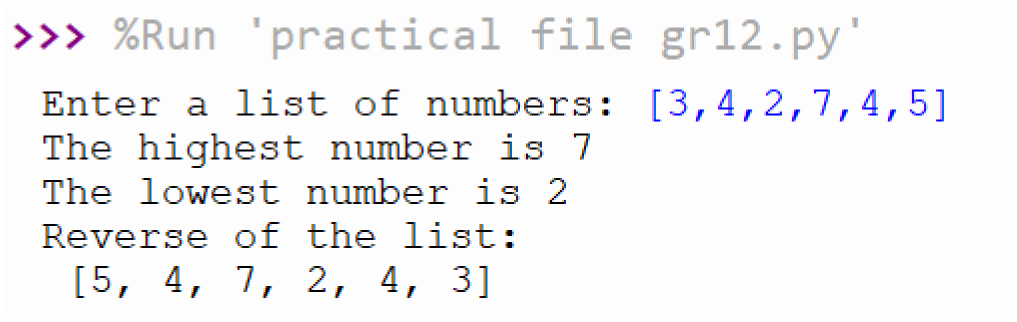
Create a Python program to accept a list of numbers from the user then display the highest number and

lowest number. Also display the reverse of the list.

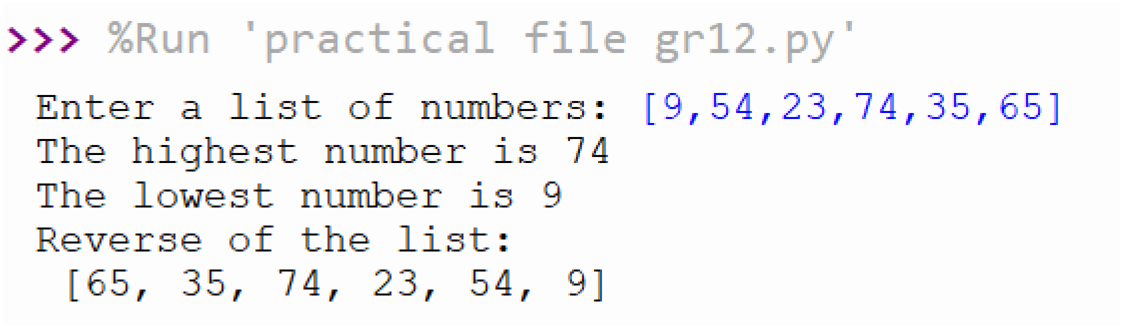
SOURCE CODE:

```
def maxmin(L):
    maxx,minn=L[0],L[0]
    for i in L:
        if i > maxx:
            maxx = i
        if i < minn:
            minn = i
    return maxx,minn

def reverse(L):
    return L[::-1]

l = eval(input('Enter a list of numbers: '))
mm = maxmin(l)
print('The highest number is',mm[0])
print('The lowest number is',mm[1])
rl = reverse(l)
print('Reverse of the list:\n',rl)
```

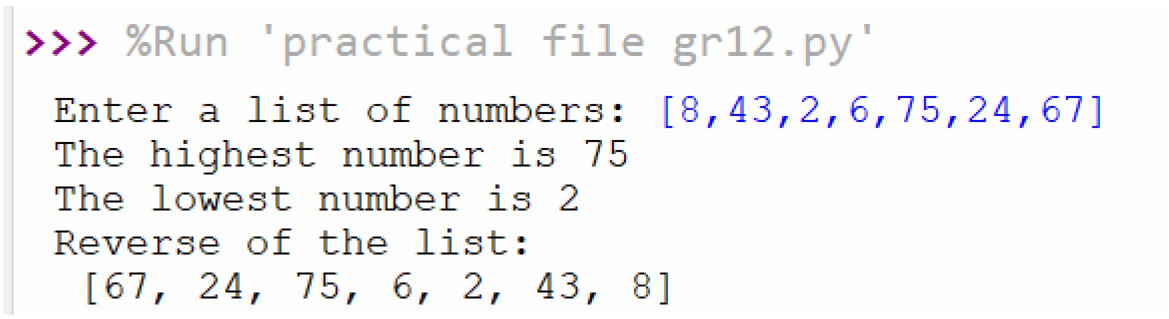
<u>OUTPUT:</u>



```
>>> %Run 'practical file gr12.py'

 Enter a list of numbers: [3,4,2,7,4,5]
 The highest number is 7
 The lowest number is 2
 Reverse of the list:
  [5, 4, 7, 2, 4, 3]
```



```
>>> %Run 'practical file gr12.py'

 Enter a list of numbers: [9,54,23,74,35,65]
 The highest number is 74
 The lowest number is 9
 Reverse of the list:
  [65, 35, 74, 23, 54, 9]
```



```
>>> %Run 'practical file gr12.py'

 Enter a list of numbers: [8,43,2,6,75,24,67]
 The highest number is 75
 The lowest number is 2
 Reverse of the list:
  [67, 24, 75, 6, 2, 43, 8]
```

Program 5

DATE: 25/4/22

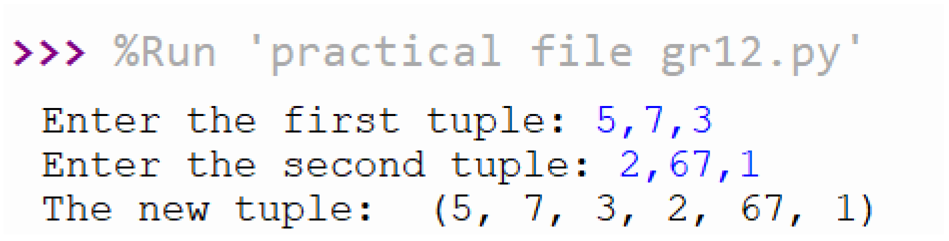
AIM:

Write a program that inputs two tuples and creates a third that contains the elements of first followed by all elements of second.

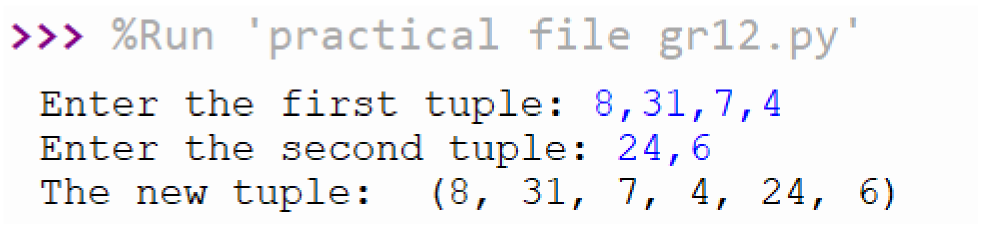
SOURCE CODE:

```
t1 = eval(input('Enter the first tuple: '))

t2 = eval(input('Enter the second tuple: '))

t3 = t1 + t2

print('The new tuple: ',t3)
```

<u>OUTPUT:</u>



```
>>> %Run 'practical file gr12.py'

 Enter the first tuple: 5,7,3
 Enter the second tuple: 2,67,1
 The new tuple:  (5, 7, 3, 2, 67, 1)
```



```
>>> %Run 'practical file gr12.py'

 Enter the first tuple: 8,31,7,4
 Enter the second tuple: 24,6
 The new tuple:  (8, 31, 7, 4, 24, 6)
```

Program 6

DATE: 27/4/22

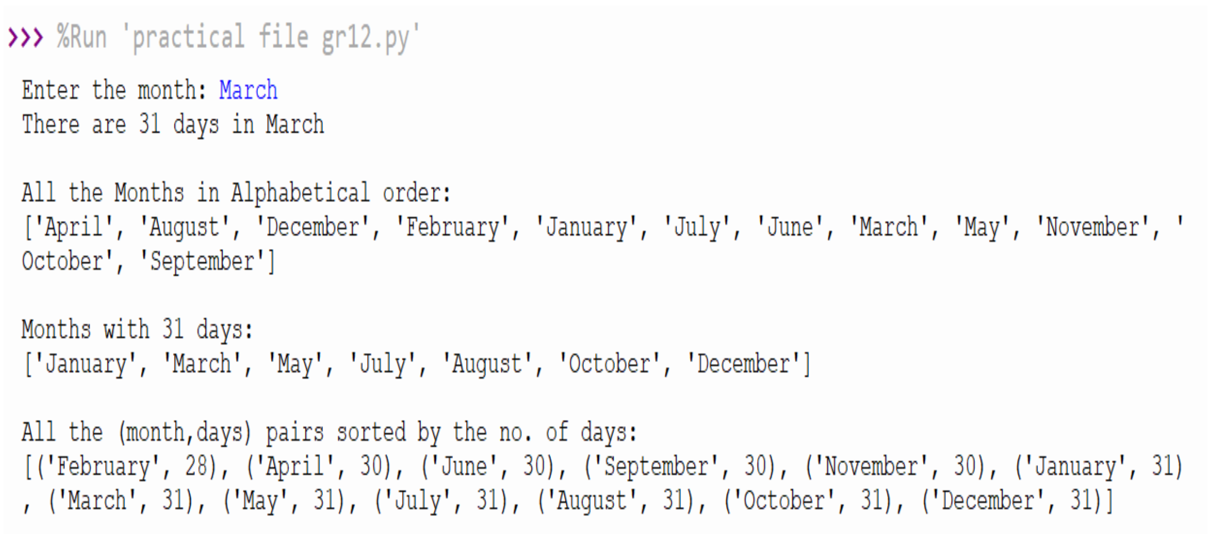
AIM:

Create a dictionary whose keys are month names and whose values are number of days in the corresponding months. Ask the user to enter a month name and use the dictionary to display the no. of days in the month. Display all the keys in Alphabetical order. Display all the months with 31 days. Display all key value pairs sorted by the number of days in each month.

SOURCE CODE:

```
d={'January':31,'February':28,'March':31,'April':30,'May':
31,'June':30,'July':31,'August':31,'September':30,'October
':31,'November':30,'December':31}

month=input('Enter the month: ')

for i in d:

    if i==month:

        days=d[i]

        f=0

        break

if f==0:

    print('There are',days,'days in',i)

else:

    print('Invalid month')

print()


print('All the Months in Alphabetical order:')

sort=sorted(d)

print(sort)

print()


l=[]

print('Months with 31 days:')

for i in d:
    if d[i]==31:
        l.append(i)
print(l)
print()

l.clear()
m,n=[],[]
print('All the (month,days) pairs sorted by the no. of
days:')
for j in d.items():
    if j[1]==28:
        l.append(j)
    elif j[1]==30:
        m.append(j)
    elif j[1]==31:
        n.append(j)
    else:
        break
lst=l+m+n
print(lst)
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 Enter the month: March
 There are 31 days in March

 All the Months in Alphabetical order:
 ['April', 'August', 'December', 'February', 'January', 'July', 'June', 'March', 'May', 'November', '
 October', 'September']

 Months with 31 days:
 ['January', 'March', 'May', 'July', 'August', 'October', 'December']

 All the (month,days) pairs sorted by the no. of days:
 [('February', 28), ('April', 30), ('June', 30), ('September', 30), ('November', 30), ('January', 31)
 , ('March', 31), ('May', 31), ('July', 31), ('August', 31), ('October', 31), ('December', 31)]
```

Program 7

DATE: 4/5/22

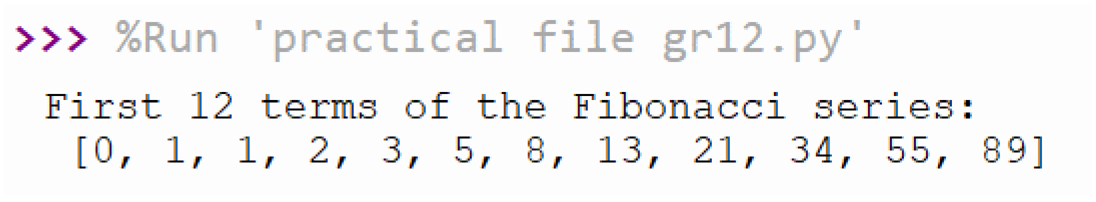
AIM:

Write a Python program to create a list for storing the first 12 terms of Fibonacci series.

SOURCE CODE:

```
def fibonacci(n):
    l=[]
    a,b=0,1
    for i in range(n):
        l.append(a)
        a,b=b,a+b
    return l

print('First 12 terms of the Fibonacci
series:\n',fibonacci(12)))
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
  First 12 terms of the Fibonacci series:
   [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Program 8

DATE: 9/5/22

AIM:

Create a Python program to have the following functions:

a. A function that takes a number as an argument and calculates its cube. If there is no value passed to

the function, it should calculate the cube of 5.

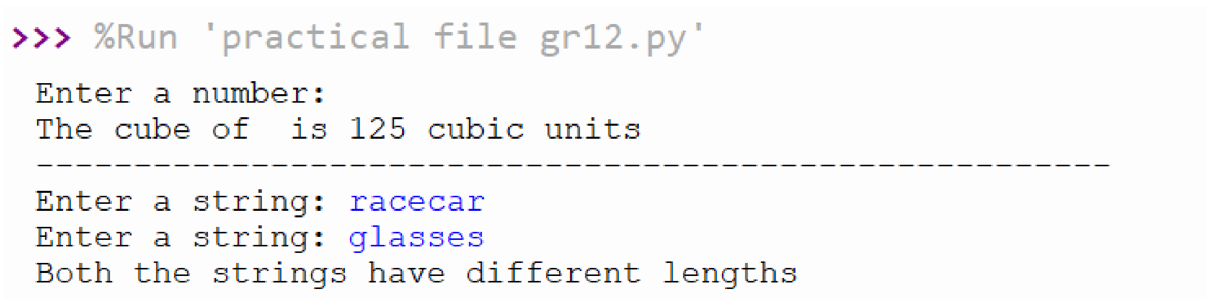
b. A function that receives two strings and checks whether they are the same length string or not. It

should return True if both are the same length string otherwise False

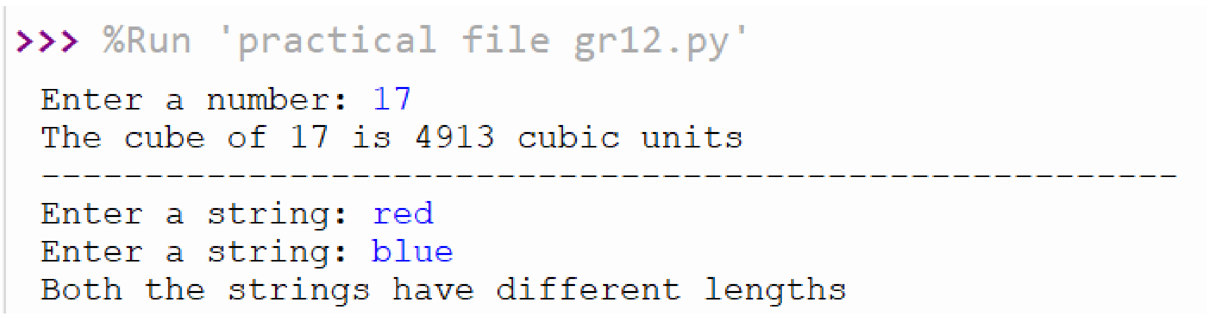
SOURCE CODE:

```
def func1(n=5):
    cube = n**3
    return cube

def func2(s1,s2):
    len1=len(s1)
    len2=len(s2)
    if s1 == s2:
        return True
    else:
        return False

num = input('Enter a number: ')
if num == '':
    c = func1()
else:
    c = func1(int(num))
print('The cube of',num,'is',c,'cubic units')

print('-----------------------------------------------------------
----')

st1 = input('Enter a string: ')
st2 = input('Enter a string: ')
if func2(st1,st2)==True:
    print('Both the strings have the same length')
else:
    print('Both the strings have different lengths')
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 Enter a number:
 The cube of  is 125 cubic units
 -------------------------------------------------------
 Enter a string: racecar
 Enter a string: glasses
 Both the strings have different lengths
```



```
>>> %Run 'practical file gr12.py'
 Enter a number: 17
 The cube of 17 is 4913 cubic units
 -------------------------------------------------------
 Enter a string: red
 Enter a string: blue
 Both the strings have different lengths
```

Program 9

DATE: 11/5/22

AIM:

Write a program to create a text file "Story.txt" and do the following:-

a. Display the number of blanks present in the file.

b. Display the number of lines present in the file.

c. Display the number of capital letters present in the file.

d. Display the number of words present in the file.

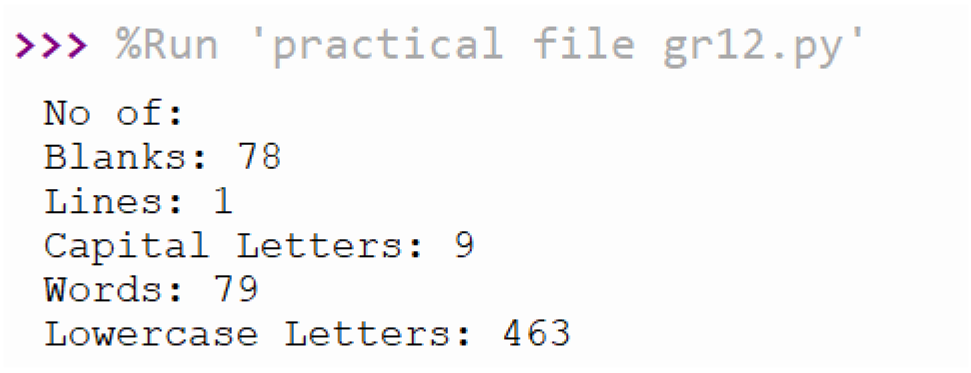
e. Display the number of lowercase letters present in the file.

SOURCE CODE:

```
def blank(r):
    c = 0
    for i in r:
        if i.isspace():
            c += 1
    return c

def lines(rl):
    c = len(rl)
    return c

def capitals(r):
    c = 0
    for i in r:
        if i.isupper():
            c += 1
    return c

def word(rl):
    c = 0
    for i in rl:
        split = i.split()
        c += len(split)
    return c
```

```
def lowercase(r):
    c = 0
    for i in r:
        if i.islower():
            c += 1
    return c

with open('story.txt','r') as f:
    readd = f.read()
    f.seek(0)
    rline = f.readlines()

    blanks = blank(readd)
    line = lines(rline)
    cap = capitals(readd)
    words = word(rline)
    lower = lowercase(readd)

    print('No of:')
    print('Blanks:',blanks)
    print('Lines:',line)
    print('Capital Letters:',cap)
    print('Words:',words)
    print('Lowercase Letters:',lower)
```

<u>OUTPUT:</u>



```
>>> %Run 'practical file gr12.py'
 No of:
 Blanks: 78
 Lines: 1
 Capital Letters: 9
 Words: 79
 Lowercase Letters: 463
```

Program 10

DATE: 16/5/22

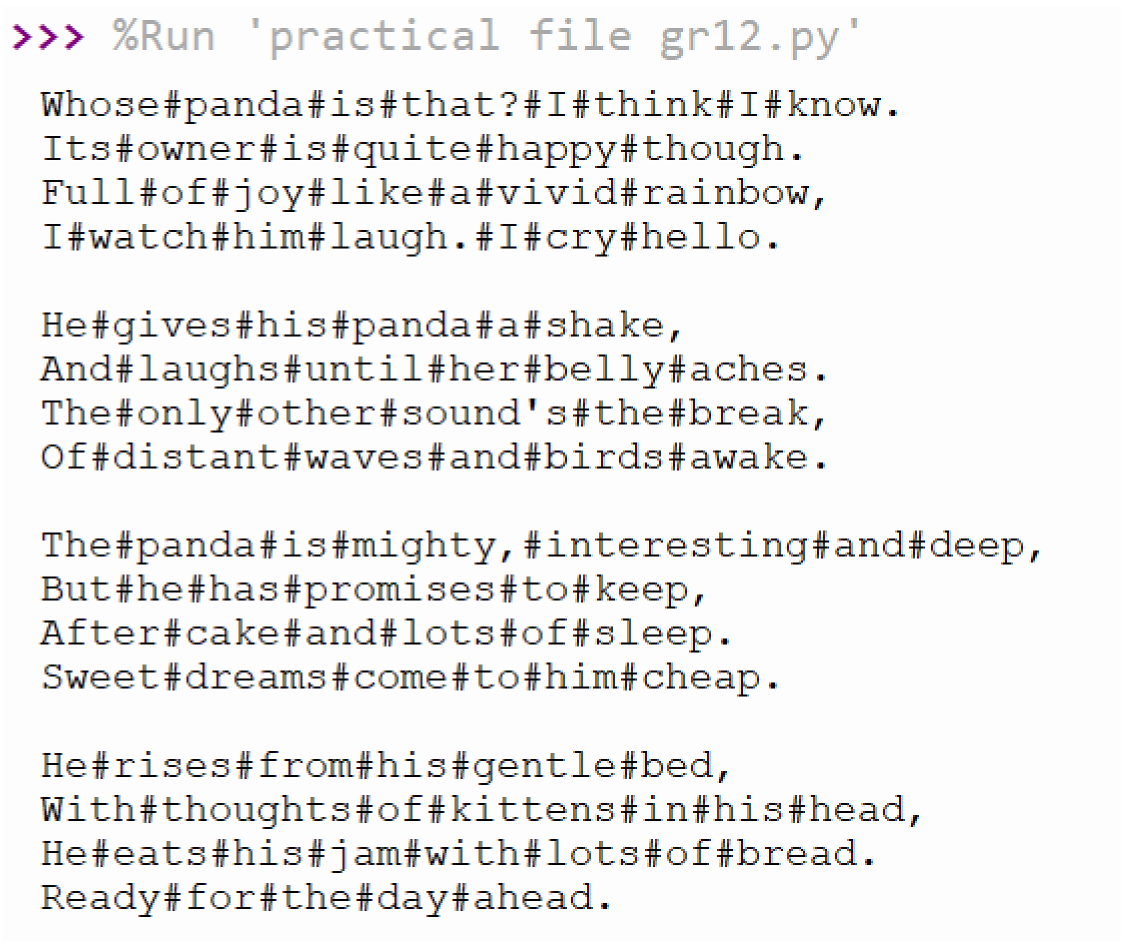
AIM:

Write a program to read the text file "Poem.txt" and display it content after replacing every space by #

SOURCE CODE:

```
def hashpoem():
    with open('poem.txt','r+') as f:
        rl = f.readlines()
        for i in rl:
            split = i.split()
            newline = '#'.join(split)
            print(newline)
hashpoem()
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 Whose#panda#is#that?#I#think#I#know.
 Its#owner#is#quite#happy#though.
 Full#of#joy#like#a#vivid#rainbow,
 I#watch#him#laugh.#I#cry#hello.

 He#gives#his#panda#a#shake,
 And#laughs#until#her#belly#aches.
 The#only#other#sound's#the#break,
 Of#distant#waves#and#birds#awake.

 The#panda#is#mighty,#interesting#and#deep,
 But#he#has#promises#to#keep,
 After#cake#and#lots#of#sleep.
 Sweet#dreams#come#to#him#cheap.

 He#rises#from#his#gentle#bed,
 With#thoughts#of#kittens#in#his#head,
 He#eats#his#jam#with#lots#of#bread.
 Ready#for#the#day#ahead.
```

Program 11

DATE: 25/5/22

AIM:

Create a text file "My School.TXT" with some information regarding your school. Read the content of the file

"My School.TXT" and display the following:

a. No of characters in the file.

b. Number of vowels present in the file.

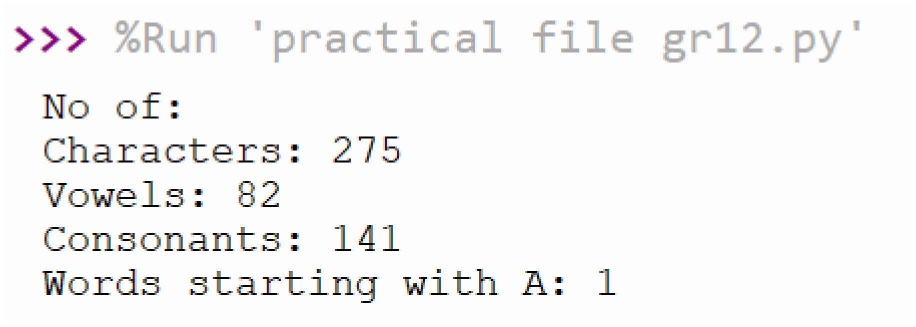
c. Number of consonants present in the file.

d. Number of words start with "A"

SOURCE CODE:

```
def strcount(txt):
    char = len(txt)
    vow,cons,A=0,0,0
    txtsplit = txt.split()
    for i in txt:
        if i.isalpha():
            if i in 'AEIOUaeiou':
                vow += 1
            else:
                cons += 1
    for j in txtsplit:
        if j[0]=='A':
            A+=1
    return char,vow,cons,A

with open('My School','r') as f:
    r = f.read()
    scount = strcount(r)
    print('No of:')
    print('Characters:',scount[0])
    print('Vowels:',scount[1])
    print('Consonants:',scount[2])
    print('Words starting with A:',scount[3])
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 No of:
 Characters: 275
 Vowels: 82
 Consonants: 141
 Words starting with A: 1
```

Program 12

DATE: 30/5/22

AIM:

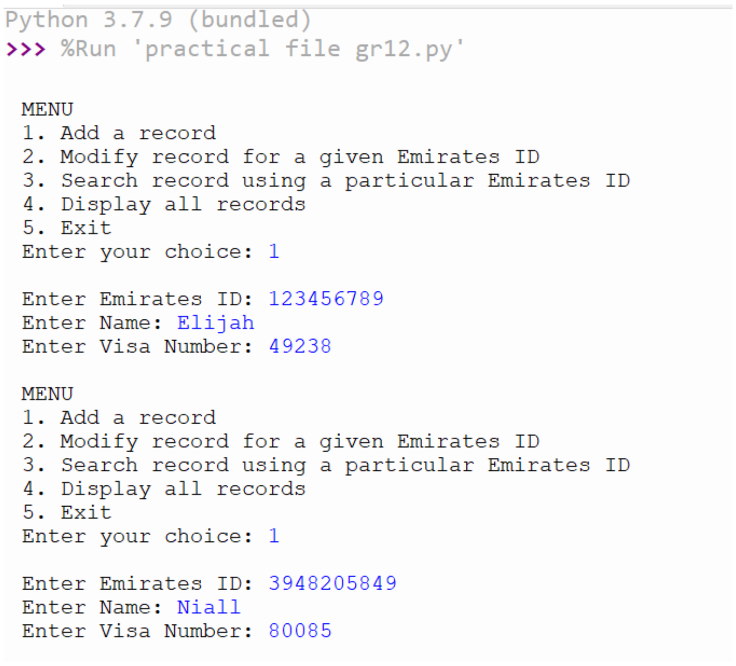
Write an interactive menu driven program to perform the following operations on a data file “UAE.dat” which stores the information such as Emirates_Id, Name, and Visa Number as a list.

1. Add a record
2. Modify record for a given Emirates_ID
3. Search record using a particular Emirates_ID
4. Display records

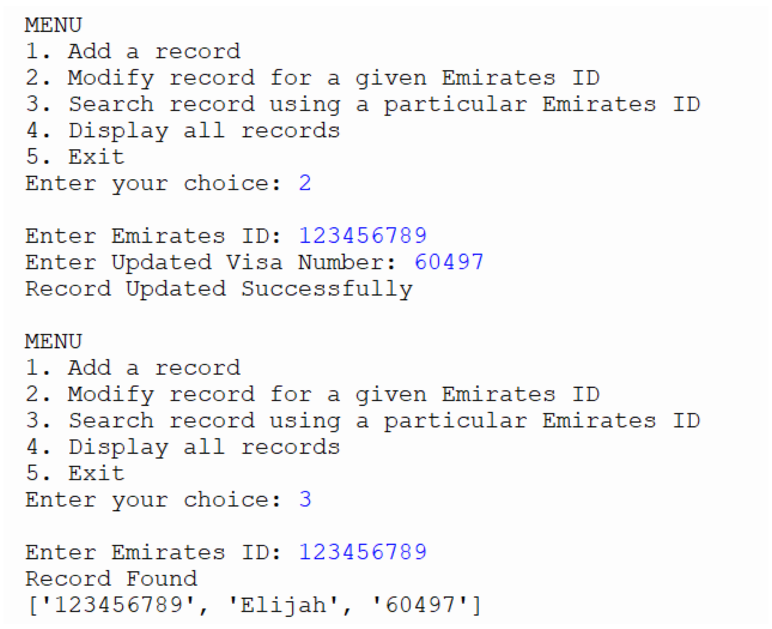
SOURCE CODE:

```
import pickle
with open("UAE.dat","wb") as f:
    rec=["Emirates_ID","Name","Visa Number"]
    pickle.dump(rec,f)

def menu():
    while True:
        c = int(input("\nMENU \n1. Add a record \n2.
Modify record for a given Emirates ID \n3. Search record
using a particular Emirates ID \n4. Display all records
\n5. Exit \nEnter your choice: "))

        if c == 1:
            add_rec()
        elif c == 2:
            modify_visa()
        elif c == 3:
            search_rec()
        elif c == 4:
            display_rec()
        elif c == 5:
            print("Session Closed Successfully")
            break
        else:
            print("Invalid Choice")

def add_rec():
    with open("UAE.dat","ab") as f:
        eid = input("\nEnter Emirates ID: ")
        name = input("Enter Name: ")
        vnum = input("Enter Visa Number: ")
        l = [eid,name,vnum]
        pickle.dump(l,f)

def modify_visa():
    with open("UAE.dat","rb+") as f:
        eid = input("\nEnter Emirates ID: ")
        flag = False
        try:
            while True:
                p = f.tell()
                rec = pickle.load(f)
                if rec[0] == eid:
                    f.seek(p)
                    vnum = input("Enter Updated Visa 
Number: ")
                    newrec = [rec[0],rec[1],vnum]
                    pickle.dump(newrec,f)
                    flag = True
        except EOFError:
            if flag == True:
                print("Record Updated Successfully")
            else:
                print("Record Invalid")

def search_rec():
    with open("UAE.dat","rb") as f:
        eid = input("\nEnter Emirates ID: ")
        try:
            r = []
            flag = False
            while True:
                rec = pickle.load(f)
                if rec[0] == eid:
                    r = rec
                    flag = True
        except EOFError:
            if flag==True:
                print("Record Found")
                print(r)
            else:
                print("Record Not Found")

def display_rec():
    with open("UAE.dat","rb") as f:
        f.seek(0)
        try:
            while True:
                rec=pickle.load(f)
                print(rec)
                print()
        except EOFError:
            print("\nEnd of File Reached")

menu()
```

OUTPUT:



```
Python 3.7.9 (bundled)
>>> %Run 'practical file gr12.py'

 MENU
 1. Add a record
 2. Modify record for a given Emirates ID
 3. Search record using a particular Emirates ID
 4. Display all records
 5. Exit
 Enter your choice: 1

 Enter Emirates ID: 123456789
 Enter Name: Elijah
 Enter Visa Number: 49238

 MENU
 1. Add a record
 2. Modify record for a given Emirates ID
 3. Search record using a particular Emirates ID
 4. Display all records
 5. Exit
 Enter your choice: 1

 Enter Emirates ID: 3948205849
 Enter Name: Niall
 Enter Visa Number: 80085
```



```
 MENU
 1. Add a record
 2. Modify record for a given Emirates ID
 3. Search record using a particular Emirates ID
 4. Display all records
 5. Exit
 Enter your choice: 2

 Enter Emirates ID: 123456789
 Enter Updated Visa Number: 60497
 Record Updated Successfully

 MENU
 1. Add a record
 2. Modify record for a given Emirates ID
 3. Search record using a particular Emirates ID
 4. Display all records
 5. Exit
 Enter your choice: 3

 Enter Emirates ID: 123456789
 Record Found
 ['123456789', 'Elijah', '60497']
```

```
MENU
1. Add a record
2. Modify record for a given Emirates ID
3. Search record using a particular Emirates ID
4. Display all records
5. Exit
Enter your choice: 4
['Emirates_ID', 'Name', 'Visa Number']

['123456789', 'Elijah', '60497']

['3948205849', 'Niall', '80085']


End of File Reached

MENU
1. Add a record
2. Modify record for a given Emirates ID
3. Search record using a particular Emirates ID
4. Display all records
5. Exit
Enter your choice: 5
Session Closed Successfully
```

Program 13

DATE: 6/6/22

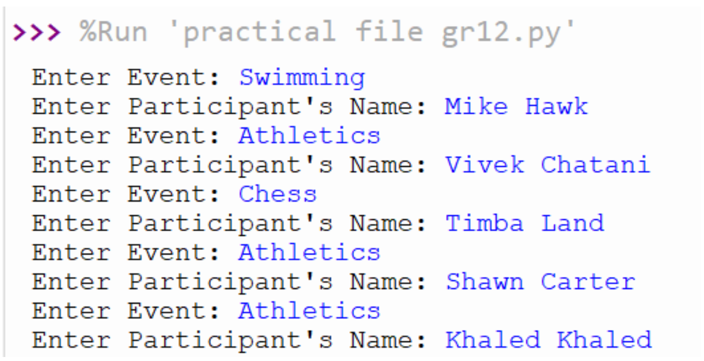
AIM:

Create a binary file “Sports.dat” to store the information of sports event in the following format : [Event, Participant Name]. Write a function that would read the content from the file “Sports.dat” and create a file named “Athletics.dat” copying only those record from Sports.dat where the event name is “Athletics”

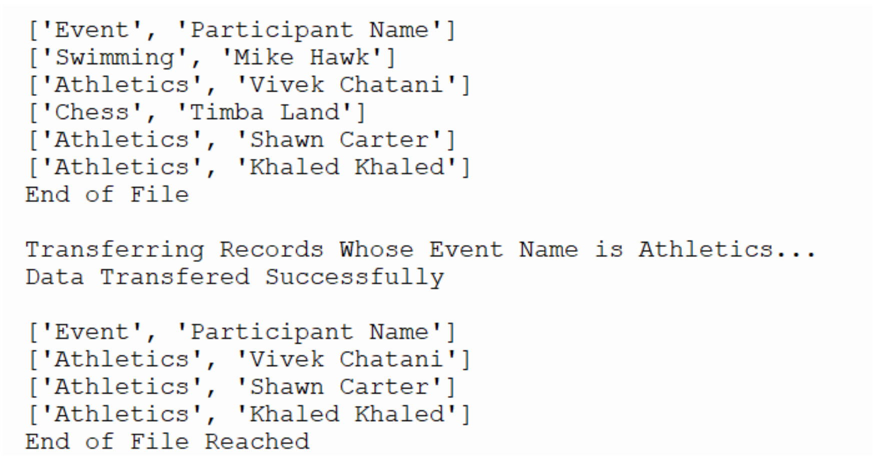
<u>SOURCE CODE:</u>

```
import pickle

def write_sports():
    with open('Sports.dat','wb') as f:
        pickle.dump(['Event','Participant Name'],f)
        for i in range(5):
            event=input('Enter Event: ')
            name=input('Enter Participant\'s Name: ')
            rec=[event,name]
            pickle.dump(rec,f)

def read_sports():
    print()
    with open('Sports.dat','rb') as f:
        try:
            while True:
                rec=pickle.load(f)
                print(rec)
        except EOFError:
            print('End of File\n')

def transfer():
    with open('Sports.dat','rb') as f:
        with open('Athletics.dat','wb') as f1:
            rec=pickle.load(f)
            pickle.dump(rec,f1)
            try:
                print('Transferring Records Whose Event
Name is Athletics...')
                while True:
                    rec=pickle.load(f)
                    if rec[0]=='Athletics':
                        pickle.dump(rec,f1)
            except EOFError:
                print('Data Transfered Successfully\n')

def read_athletics():
    with open('Athletics.dat','rb') as f:
        try:
            while True:
                rec=pickle.load(f)
                print(rec)
        except EOFError:
            print('End of File Reached')

write_sports()
read_sports()
transfer()
read_athletics()
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 Enter Event: Swimming
 Enter Participant's Name: Mike Hawk
 Enter Event: Athletics
 Enter Participant's Name: Vivek Chatani
 Enter Event: Chess
 Enter Participant's Name: Timba Land
 Enter Event: Athletics
 Enter Participant's Name: Shawn Carter
 Enter Event: Athletics
 Enter Participant's Name: Khaled Khaled
```



```
['Event', 'Participant Name']
['Swimming', 'Mike Hawk']
['Athletics', 'Vivek Chatani']
['Chess', 'Timba Land']
['Athletics', 'Shawn Carter']
['Athletics', 'Khaled Khaled']
End of File

Transferring Records Whose Event Name is Athletics...
Data Transfered Successfully

['Event', 'Participant Name']
['Athletics', 'Vivek Chatani']
['Athletics', 'Shawn Carter']
['Athletics', 'Khaled Khaled']
End of File Reached
```

Program 14

DATE: 13/6/22

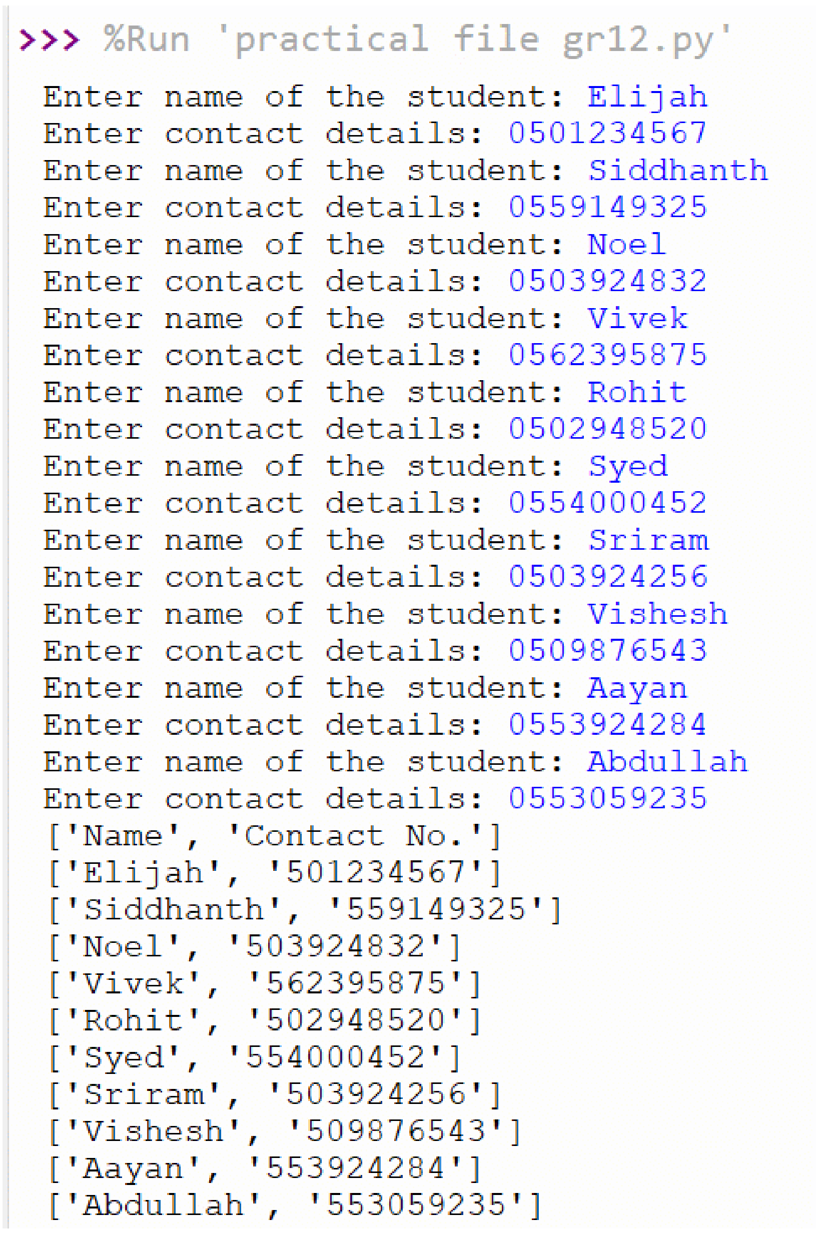
AIM:

Create a CSV file called "contact.csv" to store name and contact number of any 10 students. Write a function to display the content of the file.

SOURCE CODE:

```
import csv


def writerec():
    with open('contact.csv','w',newline='') as f:
        rec = csv.writer(f)
        rec.writerow(['Name','Contact No.'])
        for i in range(10):
            name = input('Enter name of the student: ')
            contact = int(input('Enter contact details: 
'))
            rec.writerow([name,contact])


def readrec():
    with open('contact.csv','r') as f:
        rec = csv.reader(f)
        for i in rec:
            print(i)


writerec()
readrec()
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 Enter name of the student: Elijah
 Enter contact details: 0501234567
 Enter name of the student: Siddhanth
 Enter contact details: 0559149325
 Enter name of the student: Noel
 Enter contact details: 0503924832
 Enter name of the student: Vivek
 Enter contact details: 0562395875
 Enter name of the student: Rohit
 Enter contact details: 0502948520
 Enter name of the student: Syed
 Enter contact details: 0554000452
 Enter name of the student: Sriram
 Enter contact details: 0503924256
 Enter name of the student: Vishesh
 Enter contact details: 0509876543
 Enter name of the student: Aayan
 Enter contact details: 0553924284
 Enter name of the student: Abdullah
 Enter contact details: 0553059235
 ['Name', 'Contact No.']
 ['Elijah', '501234567']
 ['Siddhanth', '559149325']
 ['Noel', '503924832']
 ['Vivek', '562395875']
 ['Rohit', '502948520']
 ['Syed', '554000452']
 ['Sriram', '503924256']
 ['Vishesh', '509876543']
 ['Aayan', '553924284']
 ['Abdullah', '553059235']
```

Program 15

DATE: 20/6/22

AIM:

Create a CSV file “books.csv” to store BOOKID, BOOKNAME, AUTHOR and PRICE as a tabular form: Write it as

a menu driven program to do the following:-

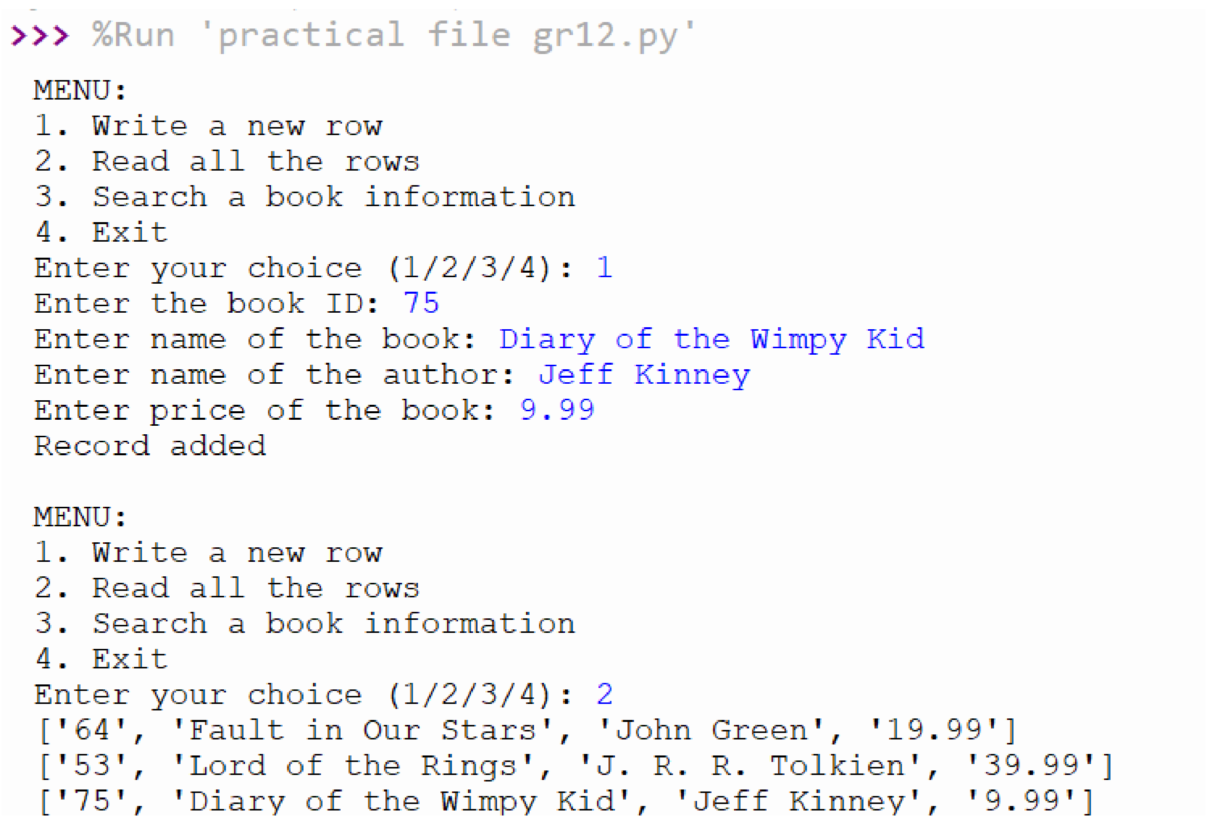
1. Write new row
2. Read all rows
3. Search a book information based on given bookid.

SOURCE CODE:

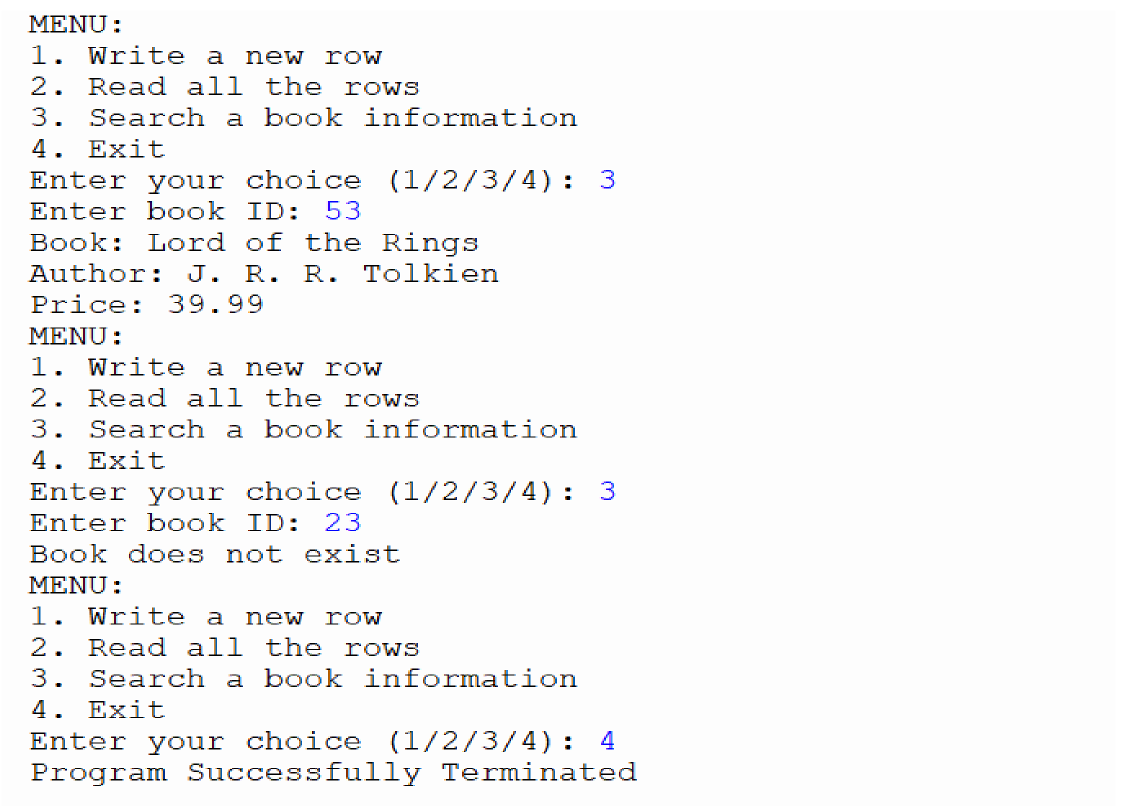
```
import csv

def menu():
    while True:
        print('MENU:')
        print('1. Write a new row')
        print('2. Read all the rows')
        print('3. Search a book information')
        print('4. Exit')
        o = int(input('Enter your choice (1/2/3/4): '))
        if o == 1:
            writerow()
        elif o == 2:
            readrows()
        elif o == 3:
            searchbook()
        elif o == 4:
            print('Program Successfully Terminated')
            break
        else:
            print('Invalid Option')

def writerow():
    with open('books.csv','a',newline='') as f:
        rec = csv.writer(f)
        b_id = int(input('Enter the book ID: '))
        book = input('Enter name of the book: ')
        author = input('Enter name of the author: ')
        price = float(input('Enter price of the book: '))
        rec.writerow([b_id,book,author,price])
        print('Record added\n')

def readrows():
    with open('books.csv','r') as f:
        rec = csv.reader(f)
        for i in rec:
            print(i)

def searchbook():
    with open('books.csv','r') as f:
        rec = csv.reader(f)
        b_id = int(input('Enter book ID: '))
        for i in rec:
            if str(b_id)==i[0]:
                print('Book:',i[1])
                print('Author:',i[2])
                print('Price:',i[3])
                break
        else:
            print('Book does not exist')
```

```
menu()
```

<u>OUTPUT:</u>



```
>>> %Run 'practical file gr12.py'
 MENU:
 1. Write a new row
 2. Read all the rows
 3. Search a book information
 4. Exit
 Enter your choice (1/2/3/4): 1
 Enter the book ID: 75
 Enter name of the book: Diary of the Wimpy Kid
 Enter name of the author: Jeff Kinney
 Enter price of the book: 9.99
 Record added

 MENU:
 1. Write a new row
 2. Read all the rows
 3. Search a book information
 4. Exit
 Enter your choice (1/2/3/4): 2
 ['64', 'Fault in Our Stars', 'John Green', '19.99']
 ['53', 'Lord of the Rings', 'J. R. R. Tolkien', '39.99']
 ['75', 'Diary of the Wimpy Kid', 'Jeff Kinney', '9.99']
```



```
MENU:
1. Write a new row
2. Read all the rows
3. Search a book information
4. Exit
Enter your choice (1/2/3/4): 3
Enter book ID: 53
Book: Lord of the Rings
Author: J. R. R. Tolkien
Price: 39.99
MENU:
1. Write a new row
2. Read all the rows
3. Search a book information
4. Exit
Enter your choice (1/2/3/4): 3
Enter book ID: 23
Book does not exist
MENU:
1. Write a new row
2. Read all the rows
3. Search a book information
4. Exit
Enter your choice (1/2/3/4): 4
Program Successfully Terminated
```

Program 16

DATE: 27/6/22

AIM:

Create a menu driven program to do the following stack (BOOK)operations:-

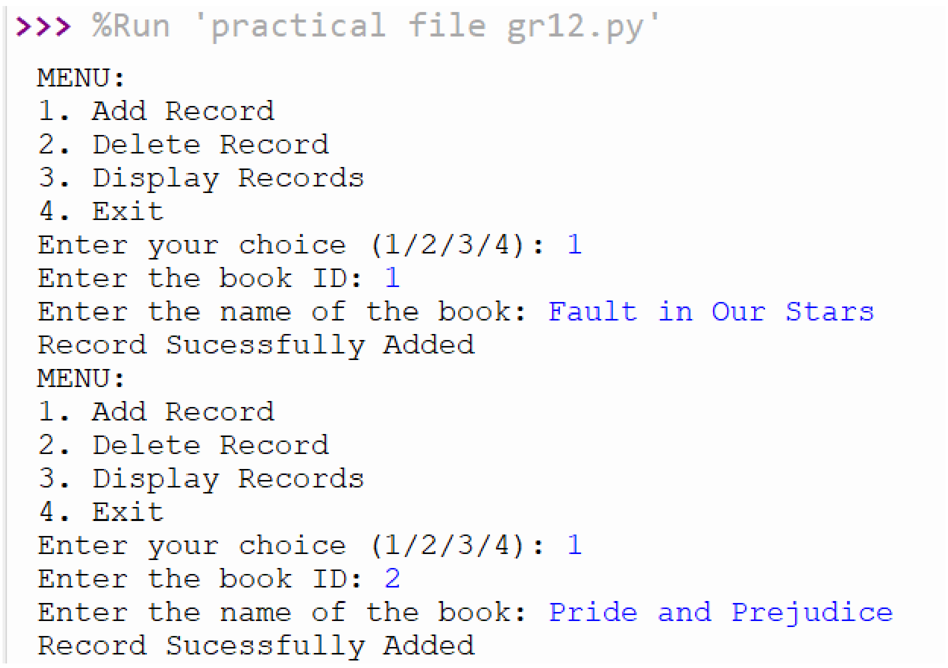
BOOK[BOOKID,BOOKNAME]

1. Push
2. Pop
3. Display
4. Exit

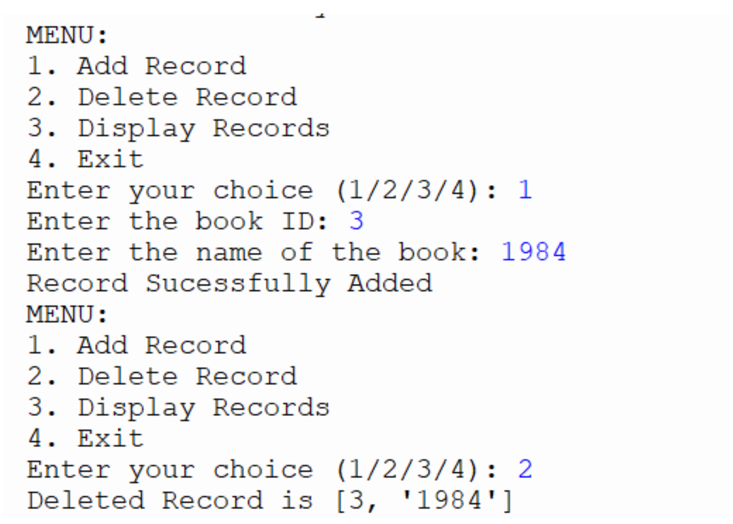
SOURCE CODE:

```
BOOK = []

def push(BOOK):
    b_id = int(input('Enter the book ID: '))
    b_name = input('Enter the name of the book: ')
    rec = [b_id,b_name]
    BOOK.append(rec)
    print('Record Sucessfully Added')

def pop(BOOK):
    if BOOK != []:
        print('Deleted Record is',BOOK.pop())
    else:
        print('No Record')

def display(BOOK):
    l = len(BOOK)
    print('Book ID\t\t Name')
    for i in range(l-1,-1,-1):
        print(BOOK[i][0],'\t\t',BOOK[i][1])

while True:
    print('MENU:')
    print('1. Add Record')
    print('2. Delete Record')
    print('3. Display Records')
    print('4. Exit')
    o = int(input('Enter your choice (1/2/3/4): '))
    if o == 1:
        push(BOOK)
    elif o == 2:
        pop(BOOK)
    elif o == 3:
        display(BOOK)
    elif o == 4:
        print('Program Successfully Terminated')
        break
    else:
        print('Invalid Choice')
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 MENU:
 1. Add Record
 2. Delete Record
 3. Display Records
 4. Exit
 Enter your choice (1/2/3/4): 1
 Enter the book ID: 1
 Enter the name of the book: Fault in Our Stars
 Record Sucessfully Added
 MENU:
 1. Add Record
 2. Delete Record
 3. Display Records
 4. Exit
 Enter your choice (1/2/3/4): 1
 Enter the book ID: 2
 Enter the name of the book: Pride and Prejudice
 Record Sucessfully Added
```



```
MENU:
1. Add Record
2. Delete Record
3. Display Records
4. Exit
Enter your choice (1/2/3/4): 1
Enter the book ID: 3
Enter the name of the book: 1984
Record Sucessfully Added
MENU:
1. Add Record
2. Delete Record
3. Display Records
4. Exit
Enter your choice (1/2/3/4): 2
Deleted Record is [3, '1984']
```

```
MENU:
1. Add Record
2. Delete Record
3. Display Records
4. Exit
Enter your choice (1/2/3/4): 3
Book ID          Name
2                Pride and Prejudice
1                Fault in Our Stars
MENU:
1. Add Record
2. Delete Record
3. Display Records
4. Exit
Enter your choice (1/2/3/4): 4
Program Successfully Terminated
```

Program 17

DATE: 31/8/22

AIM:

Create a menu driven database connectivity program to do the following transaction on a MySQL table EMP

(EMPNO,ENAME,JOB,SALARY,DOB,ADDRESS,DEPTNO).

1.Create

2.Insert

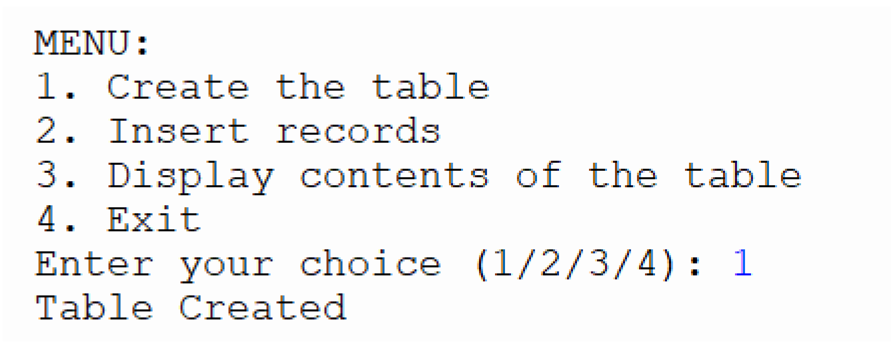
3.Display

4.Exit

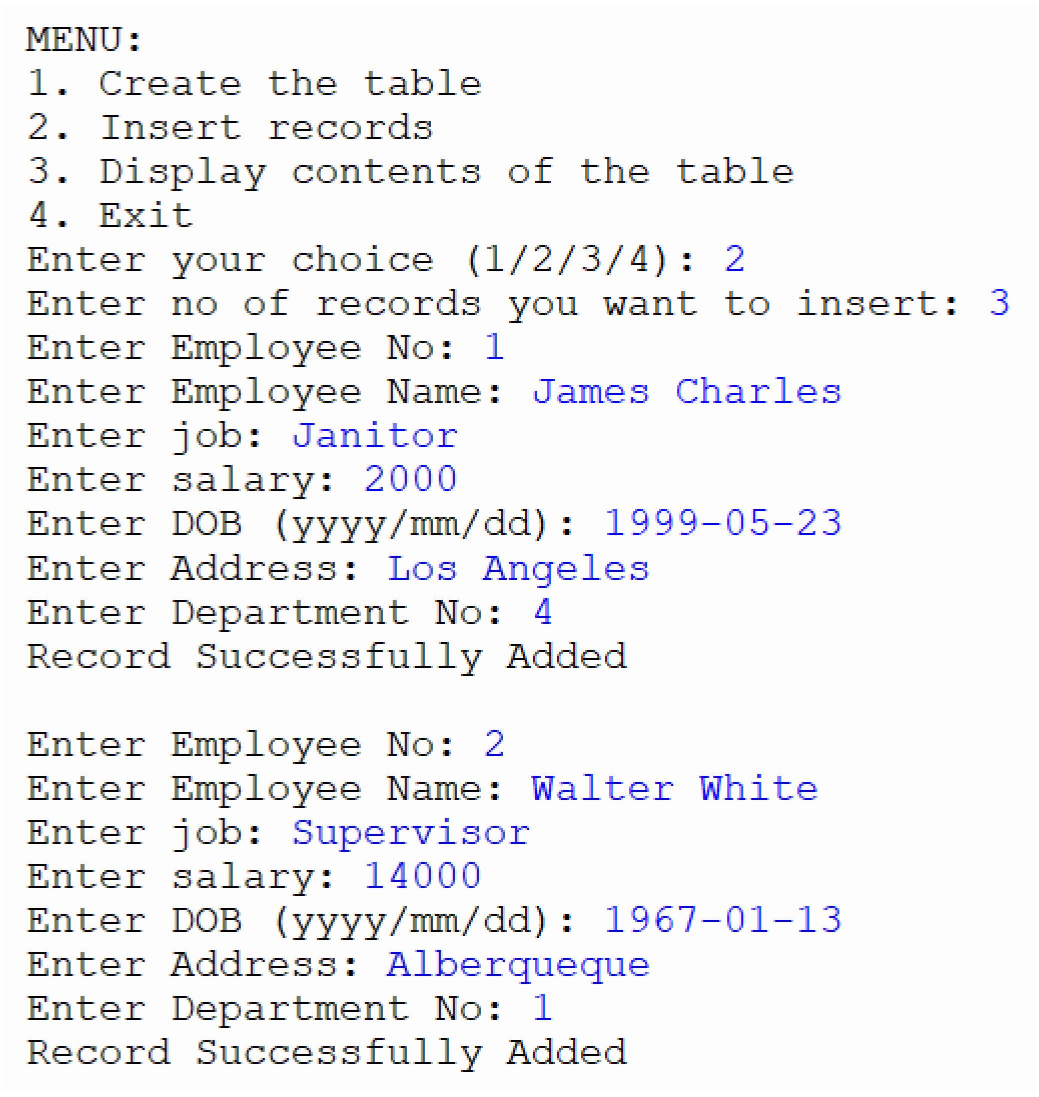
SOURCE CODE:

```
import mysql.connector
from datetime import date

mydb = mysql.connector.connect(host='localhost',user='root',passwd='Emsitchaf1',database='elconnect')
mycursor = mydb.cursor()

def insert():
    n = int(input('Enter no of records you want to insert: '))
    try:
        for i in range(n):
            EMPNO = int(input('Enter Employee No: '))
            ENAME = input('Enter Employee Name: ')
            JOB = input('Enter job: ')
            SALARY = int(input('Enter salary: '))
            DOB = input('Enter DOB (yyyy/mm/dd): ')
            ADDRESS = input('Enter Address: ')
            DEPTNO = int(input('Enter Department No: '))
            mycursor.execute('INSERT INTO EMP VALUES({},"{}","{}",{},"{}","{}",{})'.format(EMPNO,ENAME,JOB,SALARY,DOB,ADDRESS,DEPTNO))
            mydb.commit()
            print('Record Successfully Added\n')
    except:
        print('Error')

def display():
    mycursor.execute('SELECT * FROM EMP')
    x = mycursor.fetchall()
    for i in x:
        l = list(i)
        l[4]=date.isoformat(l[4])
        print(tuple(l))

while True:
    print('MENU:')
    print('1. Create the table')
    print('2. Insert records')
    print('3. Display contents of the table')
    print('4. Exit')
    o = int(input('Enter your choice (1/2/3/4): '))
    if o == 1:
        try:
            mycursor.execute('CREATE TABLE EMP(EMPNO
int(5),ENAME varchar(34),JOB varchar(15),SALARY int(5),DOB
date,ADDRESS varchar(50),DEPTNO int(2))')
            print('Table Created\n')
        except:
            print('Error\n')
    elif o == 2:
        insert()
    elif o == 3:
        display()
    elif o == 4:
        mydb.close()
        print('Program Successfully Terminated')
        break
    else:
        print('Invalid Choice')
```

OUTPUT:



```
MENU:
1. Create the table
2. Insert records
3. Display contents of the table
4. Exit
Enter your choice (1/2/3/4): 1
Table Created
```



```
MENU:
1. Create the table
2. Insert records
3. Display contents of the table
4. Exit
Enter your choice (1/2/3/4): 2
Enter no of records you want to insert: 3
Enter Employee No: 1
Enter Employee Name: James Charles
Enter job: Janitor
Enter salary: 2000
Enter DOB (yyyy/mm/dd): 1999-05-23
Enter Address: Los Angeles
Enter Department No: 4
Record Successfully Added

Enter Employee No: 2
Enter Employee Name: Walter White
Enter job: Supervisor
Enter salary: 14000
Enter DOB (yyyy/mm/dd): 1967-01-13
Enter Address: Alberqueque
Enter Department No: 1
Record Successfully Added
```

```
Enter Employee No: 3
Enter Employee Name: Steve Lacy
Enter job: Accountant
Enter salary: 10000
Enter DOB (yyyy/mm/dd): 2001-06-03
Enter Address: San Francisco
Enter Department No: 6
Record Successfully Added

MENU:
1. Create the table
2. Insert records
3. Display contents of the table
4. Exit
Enter your choice (1/2/3/4): 3
(1, 'James Charles', 'Janitor', 2000, '1999-05-23', 'Los Angeles', 4)
(2, 'Walter White', 'Supervisor', 14000, '1967-01-13', 'Alberqueque ', 1)
(3, 'Steve Lacy', 'Accountant', 10000, '2001-06-03', 'San Francisco', 6)
MENU:
1. Create the table
2. Insert records
3. Display contents of the table
4. Exit
Enter your choice (1/2/3/4): 4
Program Successfully Terminated
```

Program 18

DATE: 12/9/22

AIM:

Create a menu driven database connectivity program to do the following transaction on a MySQL table TEACHER (TID,TNAME,DESIGNATION,SALARY,DATEOFJOINING,PHONENUMBER,ADDRESS).

1.Create

2.Insert

3.Delete

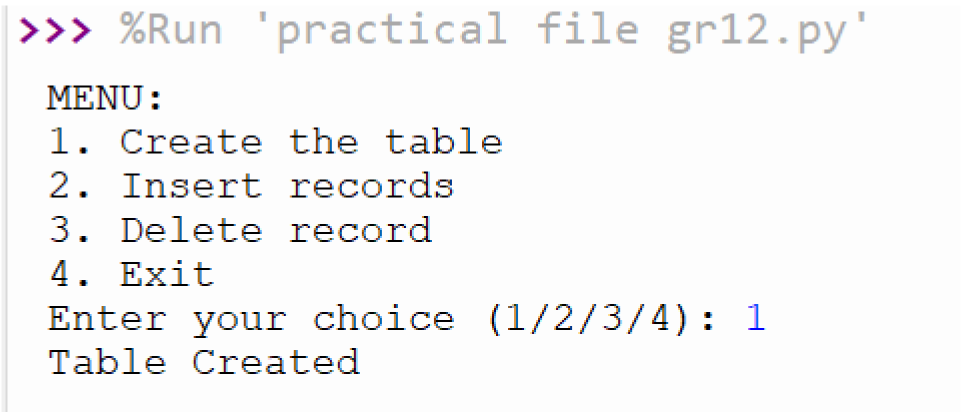
4.Exit,

SOURCE CODE:

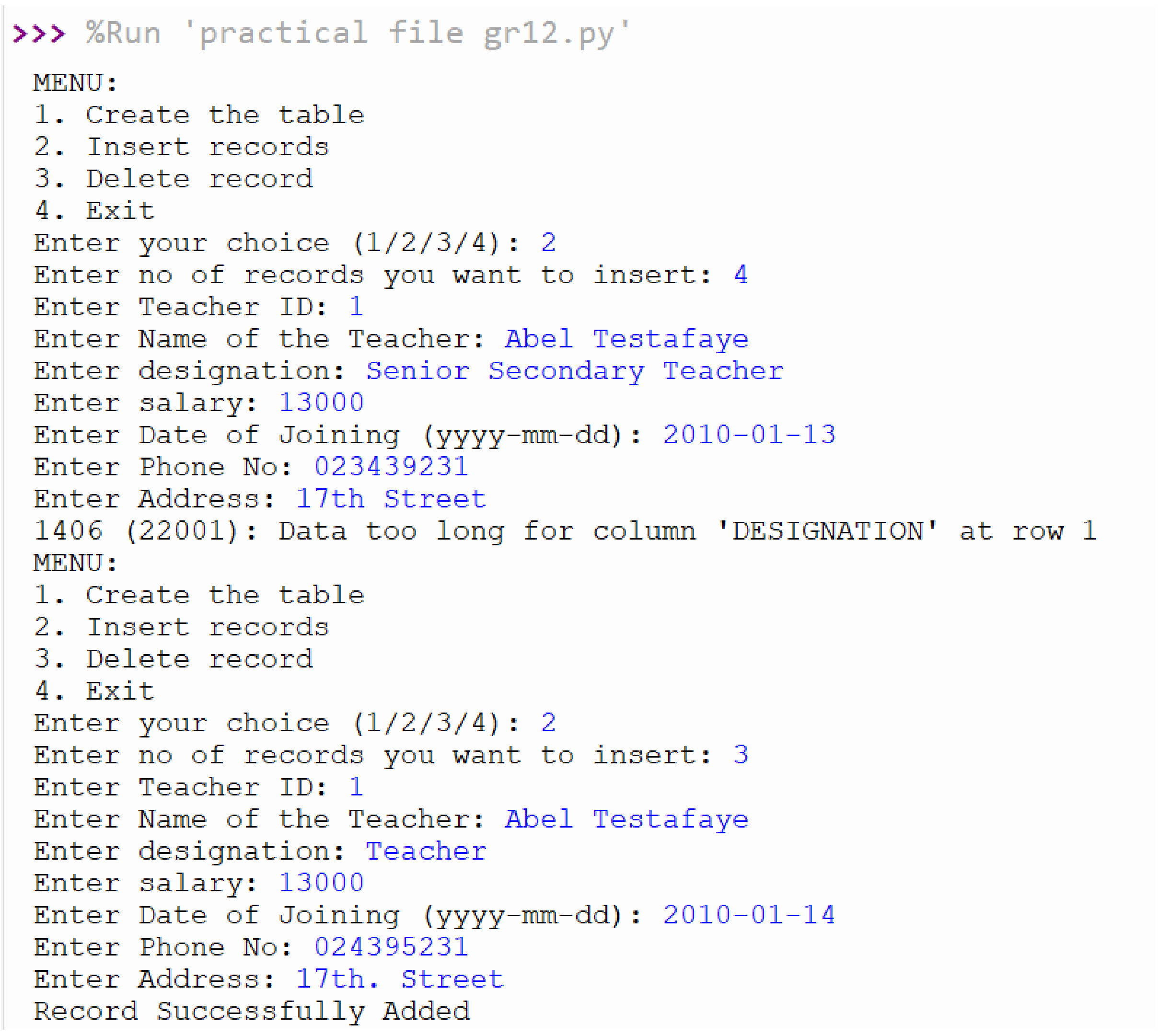
```
import mysql.connector


mydb =
mysql.connector.connect(host='localhost',user='root',passw
d='Emsitchaf1',database='elconnect')

mycursor = mydb.cursor()


def insert():

    n = int(input('Enter no of records you want to insert:
'))

    try:

        for i in range(n):

            TID = int(input('Enter Teacher ID: '))

            TNAME = input('Enter Name of the Teacher: ')

            DESIGNATION = input('Enter designation: ')

            SALARY = int(input('Enter salary: '))

            DOJ = input('Enter Date of Joining (yyyy-mm-
dd): ')

            PHONE = int(input('Enter Phone No: '))

            ADDRESS = input('Enter Address: ')

            mycursor.execute('INSERT INTO TEACHER
VALUES({},"{}","{}",{},"{}",{},"{}")'.format(TID,TNAME,DES
IGNATION,SALARY,DOJ,PHONE,ADDRESS))

            mydb.commit()

            print('Record Successfully Added\n')

    except Exception as e:
        print(e)


def delete():
    tid = int(input('Enter Teacher ID to be deleted: '))
    mycursor.execute('DELETE FROM TEACHER WHERE
TID={}'.format(tid))
    mydb.commit()
    if mycursor.rowcount == 0:
        print(tid,'does not exist')
    else:
        print('Successfully deleted')



while True:
    print('MENU:')
    print('1. Create the table')
    print('2. Insert records')
    print('3. Delete record')
    print('4. Exit')
    o = int(input('Enter your choice (1/2/3/4): '))
    if o == 1:
        try:
            mycursor.execute('CREATE TABLE TEACHER(TID
int(5),TNAME varchar(30),DESIGNATION varchar(15),SALARY
int(5),DATEOFJOINING date,PHONENUMBER int(10),ADDRESS
varchar(50))')
            print('Table Created\n')
        except:
```

```
        print('Error\n')
elif o == 2:
    insert()
elif o == 3:
    delete()
elif o == 4:
    mydb.close()
    print('Program Successfully Terminated')
    break
else:
    print('Invalid Choice')
```

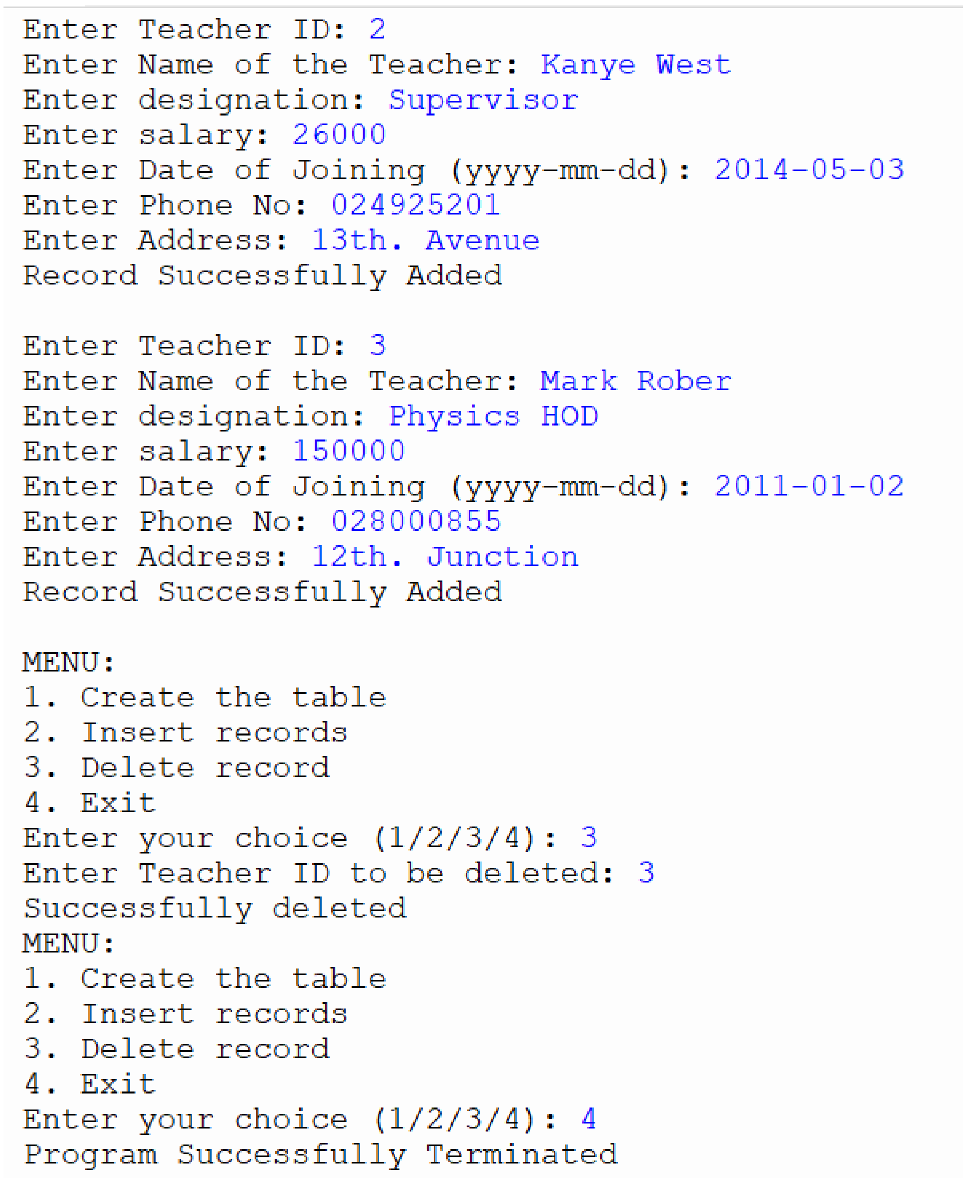
OUTPUT:



```
>>> %Run 'practical file gr12.py'
 MENU:
 1. Create the table
 2. Insert records
 3. Delete record
 4. Exit
 Enter your choice (1/2/3/4): 1
 Table Created
```



```
>>> %Run 'practical file gr12.py'
 MENU:
 1. Create the table
 2. Insert records
 3. Delete record
 4. Exit
 Enter your choice (1/2/3/4): 2
 Enter no of records you want to insert: 4
 Enter Teacher ID: 1
 Enter Name of the Teacher: Abel Testafaye
 Enter designation: Senior Secondary Teacher
 Enter salary: 13000
 Enter Date of Joining (yyyy-mm-dd): 2010-01-13
 Enter Phone No: 023439231
 Enter Address: 17th Street
 1406 (22001): Data too long for column 'DESIGNATION' at row 1
 MENU:
 1. Create the table
 2. Insert records
 3. Delete record
 4. Exit
 Enter your choice (1/2/3/4): 2
 Enter no of records you want to insert: 3
 Enter Teacher ID: 1
 Enter Name of the Teacher: Abel Testafaye
 Enter designation: Teacher
 Enter salary: 13000
 Enter Date of Joining (yyyy-mm-dd): 2010-01-14
 Enter Phone No: 024395231
 Enter Address: 17th. Street
 Record Successfully Added
```



```
Enter Teacher ID: 2
Enter Name of the Teacher: Kanye West
Enter designation: Supervisor
Enter salary: 26000
Enter Date of Joining (yyyy-mm-dd): 2014-05-03
Enter Phone No: 024925201
Enter Address: 13th. Avenue
Record Successfully Added

Enter Teacher ID: 3
Enter Name of the Teacher: Mark Rober
Enter designation: Physics HOD
Enter salary: 150000
Enter Date of Joining (yyyy-mm-dd): 2011-01-02
Enter Phone No: 028000855
Enter Address: 12th. Junction
Record Successfully Added

MENU:
1. Create the table
2. Insert records
3. Delete record
4. Exit
Enter your choice (1/2/3/4): 3
Enter Teacher ID to be deleted: 3
Successfully deleted
MENU:
1. Create the table
2. Insert records
3. Delete record
4. Exit
Enter your choice (1/2/3/4): 4
Program Successfully Terminated
```

Program 19

DATE: 2/11/22

AIM:

Create a menu driven database connectivity program to do the following transaction on a MySQL table PET

(NAME,OWNER,SPECIES,SEX,BIRTH,DEATH).

1.Create

2.Insert

3.Update

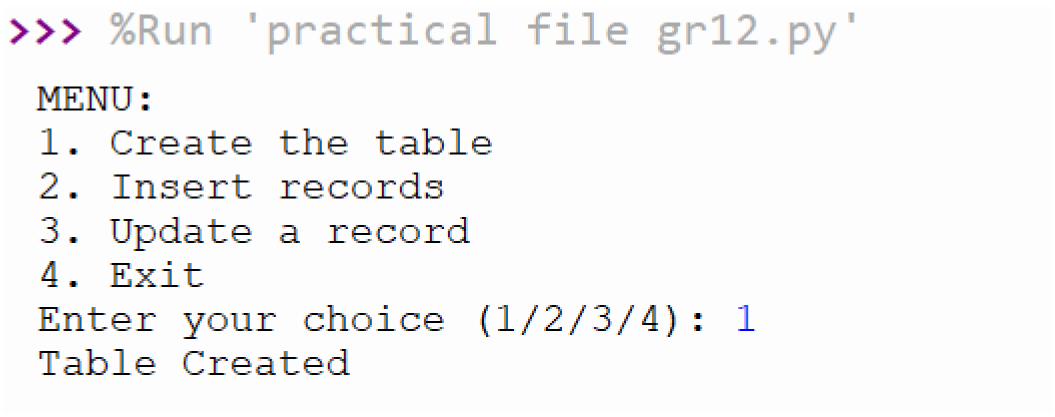
4.Exit

SOURCE CODE:

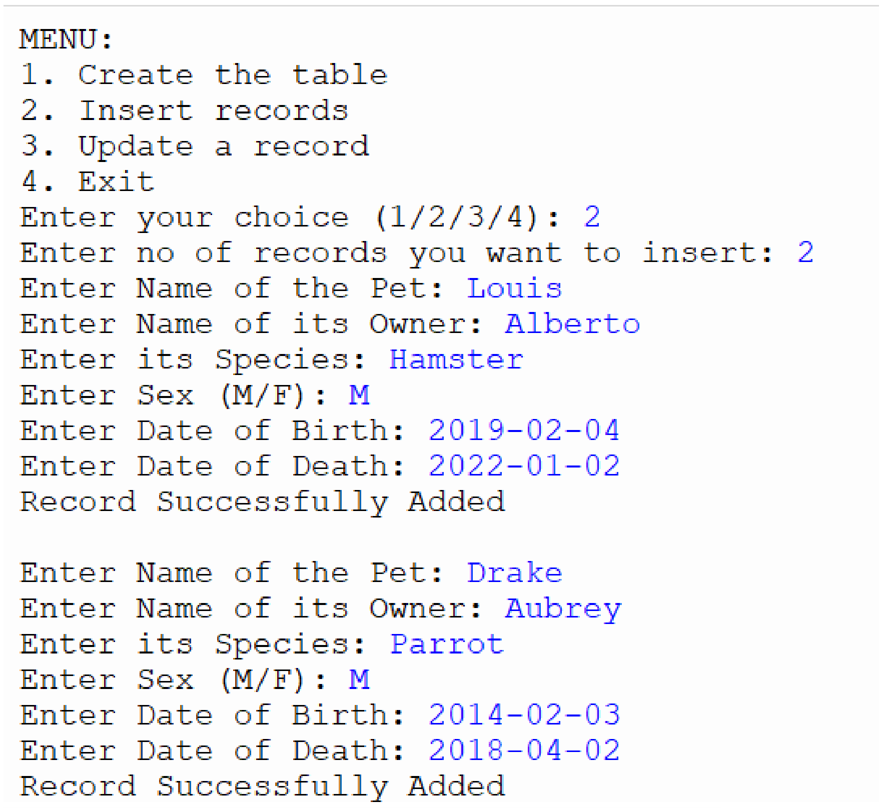
```
import mysql.connector


mydb =
mysql.connector.connect(host='localhost',user='root',passw
d='Emsitchaf1',database='elconnect')

mycursor = mydb.cursor()


def insert():

    n = int(input('Enter no of records you want to insert:
'))

    try:

        for i in range(n):

            NAME = input('Enter Name of the Pet: ')

            OWNER = input('Enter Name of its Owner: ')

            SPECIES = input('Enter its Species: ')

            SEX = input('Enter Sex (M/F): ')

            BIRTH = input('Enter Date of Birth: ')

            DEATH = input('Enter Date of Death: ')

            mycursor.execute('INSERT INTO PET
VALUES("{}","{}","{}","{}","{}","{}")'.format(NAME,OWNER,S
PECIES,SEX,BIRTH,DEATH))

            mydb.commit()

            print('Record Successfully Added\n')

    except Exception as e:

        print(e)
```

```
def update():
    name = input('Enter Name of the Pet to be updated: ')
    mycursor.execute('SELECT * FROM PET WHERE
NAME="{}"'.format(name))
    x = mycursor.fetchall()
    if x == []:
        print(name,'does not exist in the table PET')
        return
    while True:
        c = int(input('Which field to be updated?\n1.
Owner\n2. Species\n3. Sex\n4. Date of Birth\n5. Date of
Death\n6. ExitChoice (1-6): '))
        if c == 1:
            owner = input('Enter the new name of the
Owner: ')
            mycursor.execute('UPDATE PET SET OWNER="{}"
WHERE NAME="{}"'.format(owner,name))
            mydb.commit()
            print('Successfully Updated\n')
        elif c == 2:
            species = input('Enter the updated Species: ')
            mycursor.execute('UPDATE PET SET SPECIES="{}"
WHERE NAME="{}"'.format(species,name))
            mydb.commit()
            print('Successfully Updated\n')
        elif c == 3:
            sex = input('Enter the updated Sex(M/F): ')
            mycursor.execute('UPDATE PET SET SEX="{}"
WHERE NAME="{}"'.format(sex,name))

            mydb.commit()

            print('Successfully Updated\n')

        elif c == 4:

            birth = input('Enter the updated Date of
Birth: ')

            mycursor.execute('UPDATE PET SET BIRTH="{}"
WHERE NAME="{}"'.format(birth,name))

            mydb.commit()

            print('Successfully Updated\n')

        elif c == 5:

            death = input('Enter the updated Date of
Death: ')

            mycursor.execute('UPDATE PET SET DEATH="{}"
WHERE NAME="{}"'.format(death,name))

            mydb.commit()

            print('Successfully Updated\n')

        elif c == 6:

            print('Update Terminated\n')

            return

        else:

            print('Invalid Choice\n')


while True:

    print('MENU:')

    print('1. Create the table')

    print('2. Insert records')

    print('3. Update a record')
    print('4. Exit')
    o = int(input('Enter your choice (1/2/3/4): '))
    if o == 1:
        try:
            mycursor.execute('CREATE TABLE PET(NAME
varchar(25),OWNER varchar(25),SPECIES varchar(20),SEX
char(2),BIRTH date,DEATH date)')
            print('Table Created\n')
        except:
            print('Error\n')
    elif o == 2:
        insert()
    elif o == 3:
        update()
    elif o == 4:
        mydb.close()
        print('Program Successfully Terminated')
        break
    else:
        print('Invalid Choice')
```

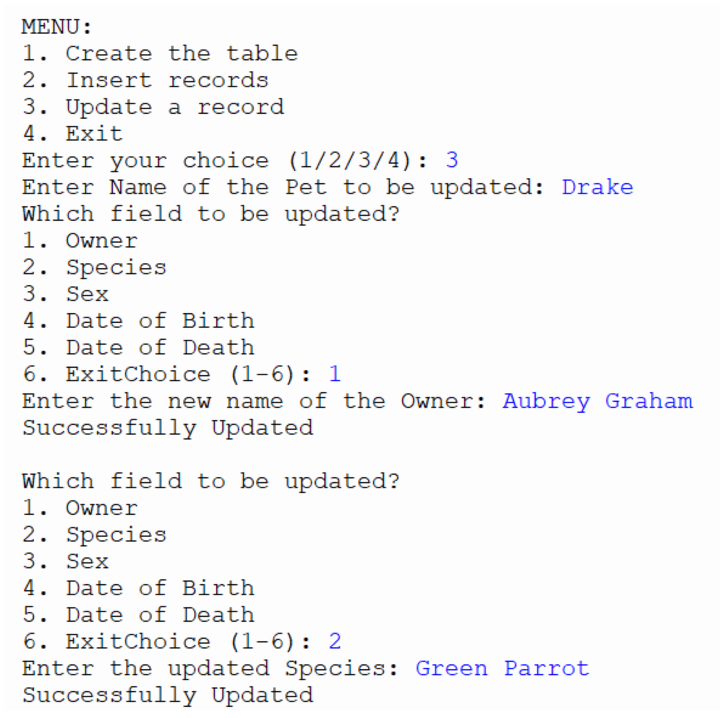
OUTPUT:



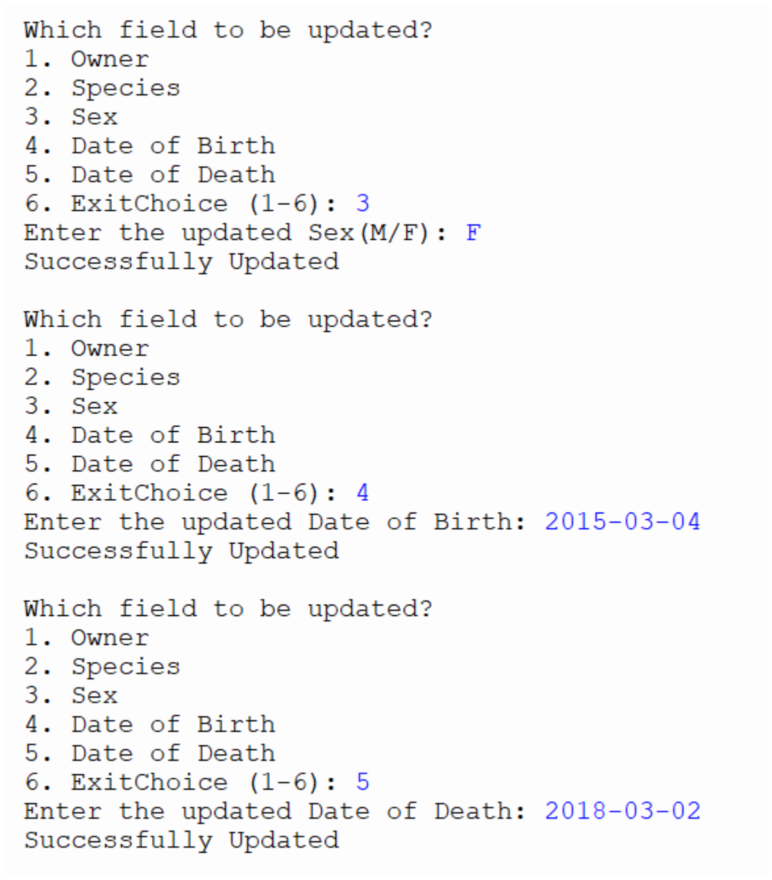
```
>>> %Run 'practical file gr12.py'

 MENU:
 1. Create the table
 2. Insert records
 3. Update a record
 4. Exit
 Enter your choice (1/2/3/4): 1
 Table Created
```



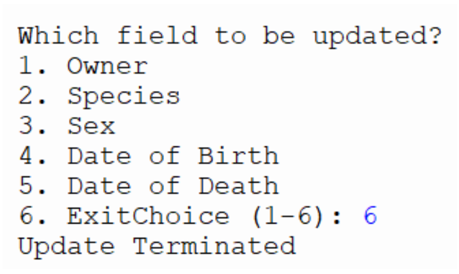
```
MENU:
1. Create the table
2. Insert records
3. Update a record
4. Exit
Enter your choice (1/2/3/4): 2
Enter no of records you want to insert: 2
Enter Name of the Pet: Louis
Enter Name of its Owner: Alberto
Enter its Species: Hamster
Enter Sex (M/F): M
Enter Date of Birth: 2019-02-04
Enter Date of Death: 2022-01-02
Record Successfully Added

Enter Name of the Pet: Drake
Enter Name of its Owner: Aubrey
Enter its Species: Parrot
Enter Sex (M/F): M
Enter Date of Birth: 2014-02-03
Enter Date of Death: 2018-04-02
Record Successfully Added
```



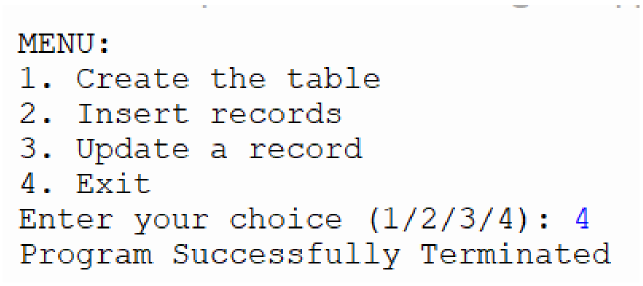
```
MENU:
1. Create the table
2. Insert records
3. Update a record
4. Exit
Enter your choice (1/2/3/4): 3
Enter Name of the Pet to be updated: Drake
Which field to be updated?
1. Owner
2. Species
3. Sex
4. Date of Birth
5. Date of Death
6. ExitChoice (1-6): 1
Enter the new name of the Owner: Aubrey Graham
Successfully Updated

Which field to be updated?
1. Owner
2. Species
3. Sex
4. Date of Birth
5. Date of Death
6. ExitChoice (1-6): 2
Enter the updated Species: Green Parrot
Successfully Updated
```



```
Which field to be updated?
1. Owner
2. Species
3. Sex
4. Date of Birth
5. Date of Death
6. ExitChoice (1-6): 3
Enter the updated Sex(M/F): F
Successfully Updated

Which field to be updated?
1. Owner
2. Species
3. Sex
4. Date of Birth
5. Date of Death
6. ExitChoice (1-6): 4
Enter the updated Date of Birth: 2015-03-04
Successfully Updated

Which field to be updated?
1. Owner
2. Species
3. Sex
4. Date of Birth
5. Date of Death
6. ExitChoice (1-6): 5
Enter the updated Date of Death: 2018-03-02
Successfully Updated
```



```
Which field to be updated?
1. Owner
2. Species
3. Sex
4. Date of Birth
5. Date of Death
6. ExitChoice (1-6): 6
Update Terminated
```



```
MENU:
1. Create the table
2. Insert records
3. Update a record
4. Exit
Enter your choice (1/2/3/4): 4
Program Successfully Terminated
```

Program 20

DATE: 9/11/22

AIM:

Create a menu driven database connectivity program to do the following transaction on a MySQL table MOV

(MOVNO,TITLE,TYPE,RATING,STARS,QTY,PRICE).

1.Create

2.Insert

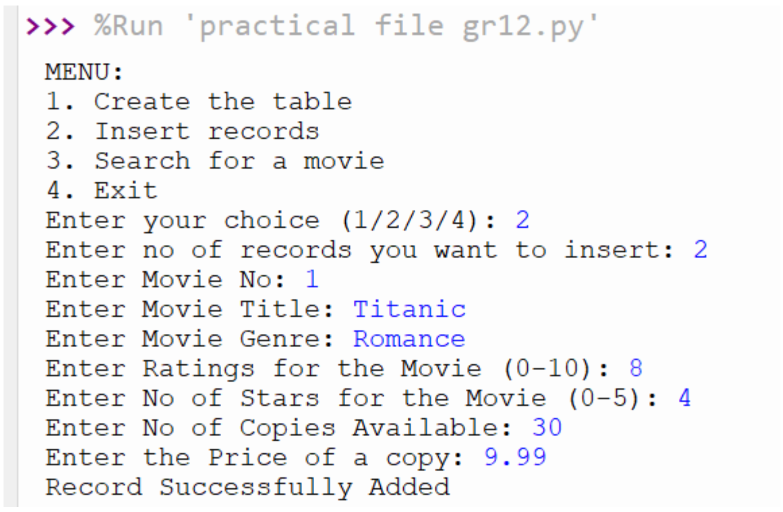
3.Search

4.Exit

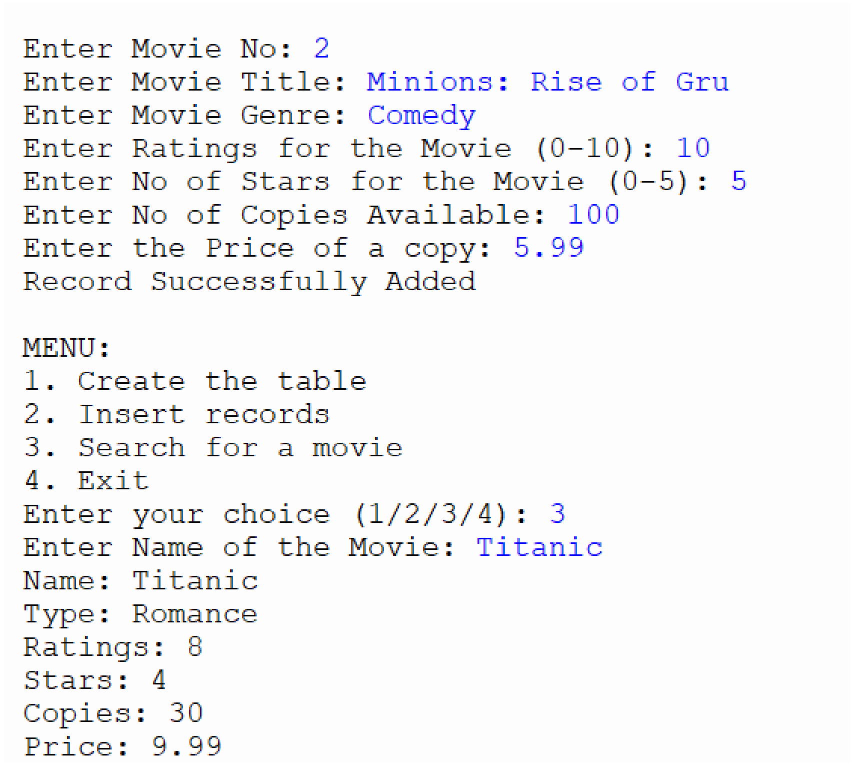
SOURCE CODE:

```
import mysql.connector


mydb =
mysql.connector.connect(host='localhost',user='root',passw
d='Emsitchaf1',database='elconnect')

mycursor = mydb.cursor()


def insert():

    n = int(input('Enter no of records you want to insert:
'))

    try:

        for i in range(n):

            MOVNO = int(input('Enter Movie No: '))

            TITLE = input('Enter Movie Title: ')

            TYPE = input('Enter Movie Genre: ')

            RATING = int(input('Enter Ratings for the
Movie (0-10): '))

            STARS = int(input('Enter No of Stars for the
Movie (0-5): '))

            QTY = int(input('Enter No of Copies Available:
'))

            PRICE = float(input('Enter the Price of a
copy: '))

            mycursor.execute('INSERT INTO MOV
VALUES({},"{}","{}",{},{},{},{})'.format(MOVNO,TITLE,TYPE,
RATING,STARS,QTY,PRICE))

            mydb.commit()
            print('Record Successfully Added\n')
    except:
        print('Error')

def search():
    movie = input('Enter Name of the Movie: ')
    mycursor.execute('SELECT * FROM MOV WHERE 
TITLE="{}"'.format(movie))
    x = mycursor.fetchone()
    if x == None:
        print(movie,'does not exist in table MOV')
    else:
        print('Name:',x[1])
        print('Type:',x[2])
        print('Ratings:',x[3])
        print('Stars:',x[4])
        print('Copies:',x[5])
        print('Price:',x[6])

while True:
    print('MENU:')
    print('1. Create the table')
    print('2. Insert records')
    print('3. Search for a movie')
    print('4. Exit')
    o = int(input('Enter your choice (1/2/3/4): '))
    if o == 1:
        try:
            mycursor.execute('CREATE TABLE MOV(MOVNO
int(3),TITLE varchar(30),TYPE varchar(15),RATING
int(2),STARS int(1),QTY int(4),PRICE float(4,2))')
            print('Table Created\n')
        except:
            print('Error\n')
    elif o == 2:
        insert()
    elif o == 3:
        search()
    elif o == 4:
        mydb.close()
        print('Program Successfully Terminated')
        break
    else:
        print('Invalid Choice')
```

OUTPUT:



```
>>> %Run 'practical file gr12.py'
 MENU:
 1. Create the table
 2. Insert records
 3. Search for a movie
 4. Exit
 Enter your choice (1/2/3/4): 2
 Enter no of records you want to insert: 2
 Enter Movie No: 1
 Enter Movie Title: Titanic
 Enter Movie Genre: Romance
 Enter Ratings for the Movie (0-10): 8
 Enter No of Stars for the Movie (0-5): 4
 Enter No of Copies Available: 30
 Enter the Price of a copy: 9.99
 Record Successfully Added
```



```
Enter Movie No: 2
Enter Movie Title: Minions: Rise of Gru
Enter Movie Genre: Comedy
Enter Ratings for the Movie (0-10): 10
Enter No of Stars for the Movie (0-5): 5
Enter No of Copies Available: 100
Enter the Price of a copy: 5.99
Record Successfully Added

MENU:
1. Create the table
2. Insert records
3. Search for a movie
4. Exit
Enter your choice (1/2/3/4): 3
Enter Name of the Movie: Titanic
Name: Titanic
Type: Romance
Ratings: 8
Stars: 4
Copies: 30
Price: 9.99
```

```
MENU:
1. Create the table
2. Insert records
3. Search for a movie
4. Exit
Enter your choice (1/2/3/4): 3
Enter Name of the Movie: Morbius
Morbius does not exist in table MOV
MENU:
1. Create the table
2. Insert records
3. Search for a movie
4. Exit
Enter your choice (1/2/3/4): 4
Program Successfully Terminated
```

Program 21

DATE: 14/11/22

AIM:

Refer the following tables and answer the following questions:

Column Name	Data Type	Size	Constraint
Admn No	Integer		Primary Key
Name	Char	20	Not Null
Age	Integer	2	
Department	Char	35	Not Null
DOJ	Date		
Gender	Char	1	

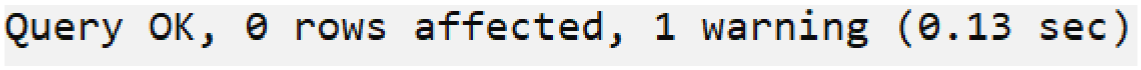
Admn No	Name	Age	Department	DOJ	Gender
100	Arpit	18	Computer	10/01/2017	M
105	Arun	19	History	24/03/2015	M
110	Ankita	17	Hindi	12/12/2016	F
112	Shilpa	20	Computer	01/07/2014	F
120	Sanjay	19	Maths	25/02/2017	M
135	Neelam	21	Hindi	31/07/2016	F

Q1. Create the table Student

QUERY:

CREATE TABLE Student(Admn_No INT PRIMARY KEY, Name CHAR(20) NOT NULL, Age INT(2), Department CHAR(35) NOT NULL, DOJ DATE, GENDER CHAR(1));

OUTPUT:



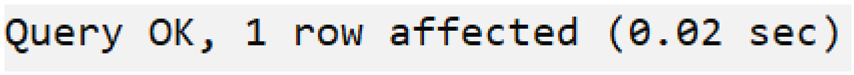
```
Query OK, 0 rows affected, 1 warning (0.13 sec)
```

Q2. Insert the above records in the table student

QUERY 1:

INSERT INTO Student VALUES(100,'Arpit',18,'Computer','2017-01-10','M');

OUTPUT:

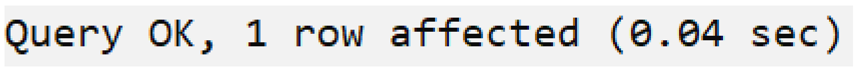


```
Query OK, 1 row affected (0.02 sec)
```

QUERY 2:

INSERT INTO Student VALUES(105,'Arun',19,'History','2015-03-24','M');

OUTPUT:

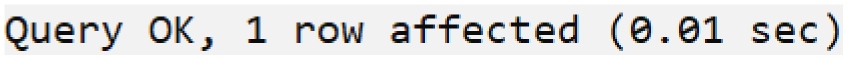


```
Query OK, 1 row affected (0.04 sec)
```

QUERY 3:

INSERT INTO Student VALUES(110,'Ankita',17,'Hindi','2016-12-12','F');

OUTPUT:

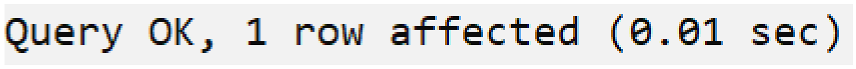


```
Query OK, 1 row affected (0.01 sec)
```

QUERY 4:

INSERT INTO Student VALUES(112,'Shilpa',20,'Computer','2014-07-01','F');

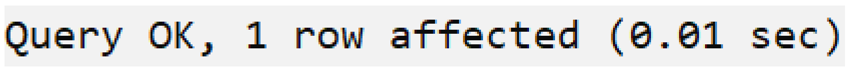
OUTPUT:



```
Query OK, 1 row affected (0.01 sec)
```

QUERY 5:

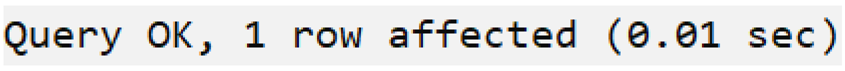
INSERT INTO Student VALUES(120,'Sanjay',19,'Maths','2017-02-25','M');



```
Query OK, 1 row affected (0.01 sec)
```

QUERY 6:

INSERT INTO Student VALUES(135,'Neelam',21,'Hindi','2016-07-31','F');



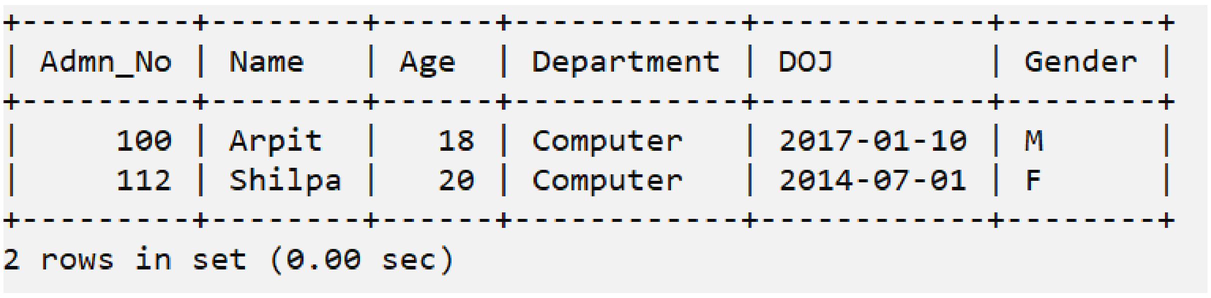
```
Query OK, 1 row affected (0.01 sec)
```

Q3. Display all information about students of ‘Computer’ department.

QUERY:

SELECT * FROM Student WHERE Department='Computer';

OUTPUT:



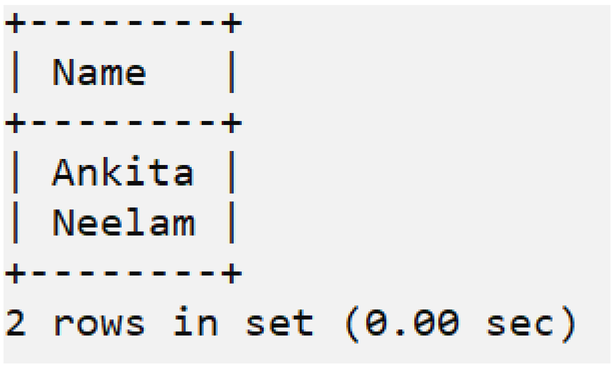
```
+---------+--------+------+------------+------------+--------+
| Admn_No | Name   | Age  | Department | DOJ        | Gender |
+---------+--------+------+------------+------------+--------+
|     100 | Arpit  |   18 | Computer   | 2017-01-10 | M      |
|     112 | Shilpa |   20 | Computer   | 2014-07-01 | F      |
+---------+--------+------+------------+------------+--------+
2 rows in set (0.00 sec)
```

Q4. Display the names of female students who are in Hindi department.

QUERY:

SELECT Name FROM Student WHERE Department='Hindi' AND Gender='F';

OUTPUT:



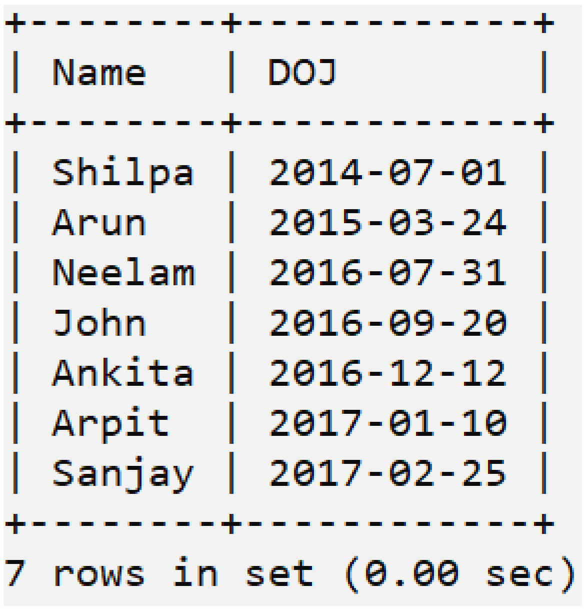
```
+--------+
| Name   |
+--------+
| Ankita |
| Neelam |
+--------+
2 rows in set (0.00 sec)
```

Q5. Display the names of all students with their date of joining in ascending order.

QUERY:

SELECT Name,DOJ FROM Student ORDER BY DOJ;

OUTPUT:



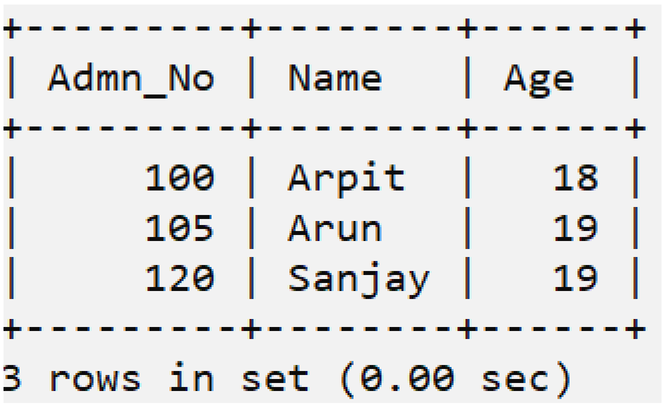
```
+--------+------------+
| Name   | DOJ        |
+--------+------------+
| Shilpa | 2014-07-01 |
| Arun   | 2015-03-24 |
| Neelam | 2016-07-31 |
| John   | 2016-09-20 |
| Ankita | 2016-12-12 |
| Arpit  | 2017-01-10 |
| Sanjay | 2017-02-25 |
+--------+------------+
7 rows in set (0.00 sec)
```

Q6. Display Admnno, Name and Age of male students only

<u>QUERY:</u>

SELECT Admn_No,Name,Age FROM Student WHERE Gender='M';

<u>OUTPUT:</u>



```
+---------+--------+------+
| Admn_No | Name   | Age  |
+---------+--------+------+
|     100 | Arpit  |   18 |
|     105 | Arun   |   19 |
|     120 | Sanjay |   19 |
+---------+--------+------+
3 rows in set (0.00 sec)
```

Q7. Count number of students who are above 19 years of age.

<u>QUERY:</u>

SELECT COUNT(*) FROM Student WHERE Age>=19;

<u>OUTPUT:</u>

```
+----------+
| COUNT(*) |
+----------+
|        4 |
+----------+
1 row in set (0.00 sec)
```

Q8. Insert a new row in the table with the following data.

Admn No : 145, Name : John, Age: 19, Department : Computer, DOJ: 2016-09-20, Gender: Male

QUERY:

INSERT INTO Student VALUES(145,'John',19,'Computer','2016-09-20','M');

OUTPUT:

```
Query OK, 1 row affected (0.01 sec)
```

Q9. Display the number of distinct departments.

QUERY:

SELECT COUNT(DISTINCT Department) FROM Student;

OUTPUT:

```
+----------------------------+
| COUNT(DISTINCT Department) |
+----------------------------+
|                          4 |
+----------------------------+
1 row in set (0.00 sec)
```

Q10. Display the number of students in each department.

QUERY:

SELECT Department,COUNT(*) FROM Student GROUP BY Department;

OUTPUT:

```
+------------+----------+
| Department | COUNT(*) |
+------------+----------+
| Computer   |        3 |
| History    |        1 |
| Hindi      |        2 |
| Maths      |        1 |
+------------+----------+
4 rows in set (0.00 sec)
```

Q11. Display the student details whose name start with 'A'

QUERY:

SELECT * FROM Student WHERE Name LIKE 'A%';

OUTPUT:

```
+---------+--------+------+------------+------------+--------+
| Admn_No | Name   | Age  | Department | DOJ        | Gender |
+---------+--------+------+------------+------------+--------+
|     100 | Arpit  |   18 | Computer   | 2017-01-10 | M      |
|     105 | Arun   |   19 | History    | 2015-03-24 | M      |
|     110 | Ankita |   17 | Hindi      | 2016-12-12 | F      |
+---------+--------+------+------------+------------+--------+
3 rows in set (0.00 sec)
```

Q12. Display the details of female students joined after 2016.

QUERY:

SELECT * FROM Student WHERE Gender='F' AND YEAR(DOJ)>2016;

OUTPUT:

```
Empty set (0.00 sec)
```

Program 22

DATE: 7/12/22

AIM:

Refer the following tables and answer the following questions:-

Table: Employee

EmpNo	Name	Salary	Zone	Age	Grade	Dept
1	Kishore	30000	West	28	A	10
2	Kritika	35000	Centre	30	A	10
3	Naveen	32000	West	40		20
4	Uday	38000	North	38	C	30
5	Nupur	32000	East	26		20
6	Moksh	37000	South	28	B	10
7	Shelly	36000	North	26	A	30

Table: Department

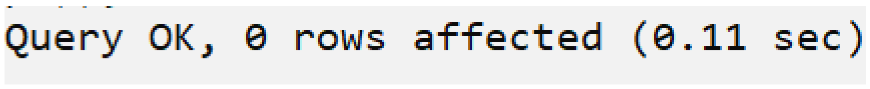
Dept	Dname	Minsal	Maxsal	HOD
10	Sales	25000	32000	1
20	Finance	30000	5000	5
30	Admin	25000	40000	7

Q1. Create the table Employee

QUERY:

CREATE TABLE Employee(EmpNo INT, Name VARCHAR(25),salary INT, Zone CHAR(6), Age INT, Grade CHAR(1),Dept INT, FOREIGN KEY(Dept) REFERENCES Department(Dept));

OUTPUT:



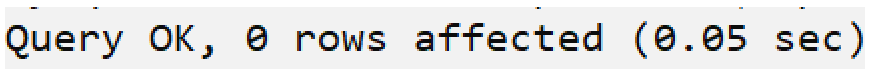
```
Query OK, 0 rows affected (0.11 sec)
```

Q2. Create the table Department

QUERY:

CREATE TABLE Department(Dept int primary key, Dname varchar(10), Minsal int, Maxsal int, HOD int);

OUTPUT:



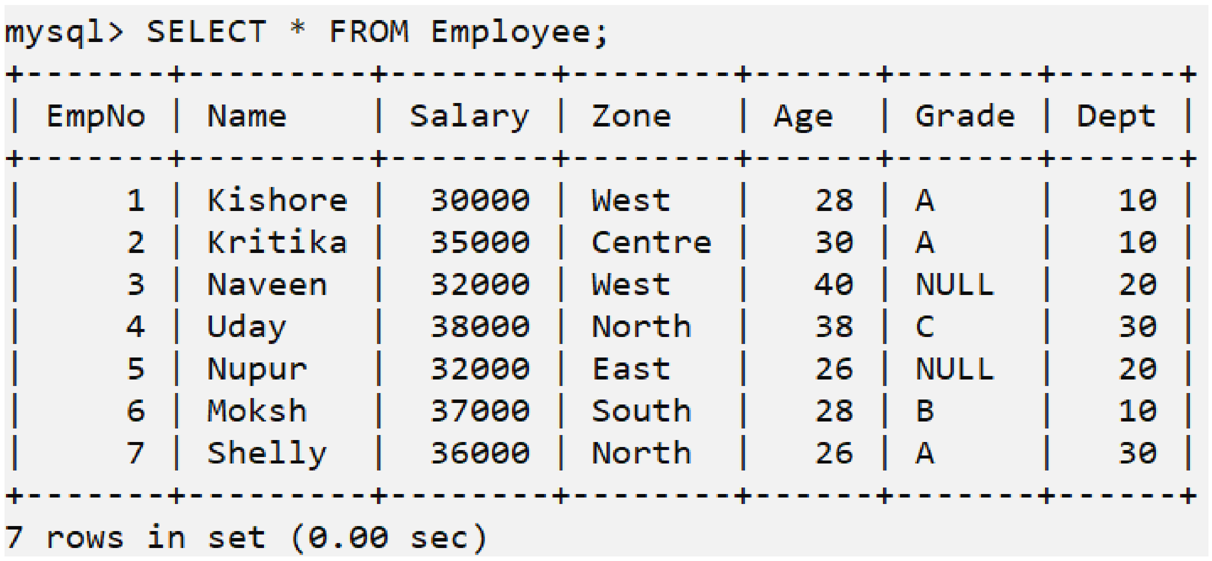
```
Query OK, 0 rows affected (0.05 sec)
```

Q3. Display the records of Employee table

QUERY:

SELECT * FROM Employee;

OUTPUT:



```
mysql> SELECT * FROM Employee;
+-------+---------+--------+--------+------+-------+------+
| EmpNo | Name    | Salary | Zone   | Age  | Grade | Dept |
+-------+---------+--------+--------+------+-------+------+
|     1 | Kishore |  30000 | West   |   28 | A     |   10 |
|     2 | Kritika |  35000 | Centre |   30 | A     |   10 |
|     3 | Naveen  |  32000 | West   |   40 | NULL  |   20 |
|     4 | Uday    |  38000 | North  |   38 | C     |   30 |
|     5 | Nupur   |  32000 | East   |   26 | NULL  |   20 |
|     6 | Moksh   |  37000 | South  |   28 | B     |   10 |
|     7 | Shelly  |  36000 | North  |   26 | A     |   30 |
+-------+---------+--------+--------+------+-------+------+
7 rows in set (0.00 sec)
```

Q4. Display the records of Department table

QUERY:

SELECT * FROM Department;

OUTPUT:

```
+------+---------+--------+--------+------+
| Dept | Dname   | Minsal | Maxsal | HOD  |
+------+---------+--------+--------+------+
|   10 | Sales   |  25000 |  32000 |    1 |
|   20 | Finance |  30000 |   5000 |    5 |
|   30 | Admin   |  25000 |  40000 |    7 |
+------+---------+--------+--------+------+
3 rows in set (0.00 sec)
```

Q5. Display the records of all employees with their annual salaries. Annual salary is calculated as salary*12.

QUERY:

SELECT Name, Salary*12 AS Annual_Salary FROM Employee;

OUTPUT:

```
+---------+---------------+
| Name    | Annual_Salary |
+---------+---------------+
| Kishore |        360000 |
| Kritika |        420000 |
| Naveen  |        384000 |
| Uday    |        456000 |
| Nupur   |        384000 |
| Moksh   |        444000 |
| Shelly  |        432000 |
+---------+---------------+
7 rows in set (0.00 sec)
```

Q6. Display the names of all employees working in North zone.

QUERY:

SELECT Name FROM Employee WHERE Zone='North';

OUTPUT:

```
+--------+
| Name   |
+--------+
| Uday   |
| Shelly |
+--------+
2 rows in set (0.00 sec)
```

Q7. Display the details of all employees whose grade is NULL

QUERY:

SELECT * FROM Employee WHERE Grade is NULL;

OUTPUT:

```
+-------+--------+--------+------+------+-------+------+
| EmpNo | Name   | Salary | Zone | Age  | Grade | Dept |
+-------+--------+--------+------+------+-------+------+
|     3 | Naveen |  32000 | West |   40 | NULL  |   20 |
|     5 | Nupur  |  32000 | East |   26 | NULL  |   20 |
+-------+--------+--------+------+------+-------+------+
2 rows in set (0.00 sec)
```

Q8. Display the names of various zones from the table Employee. A zone name should appear only once.

QUERY:

SELECT DISTINCT Zone as Zones FROM Employee;

OUTPUT:

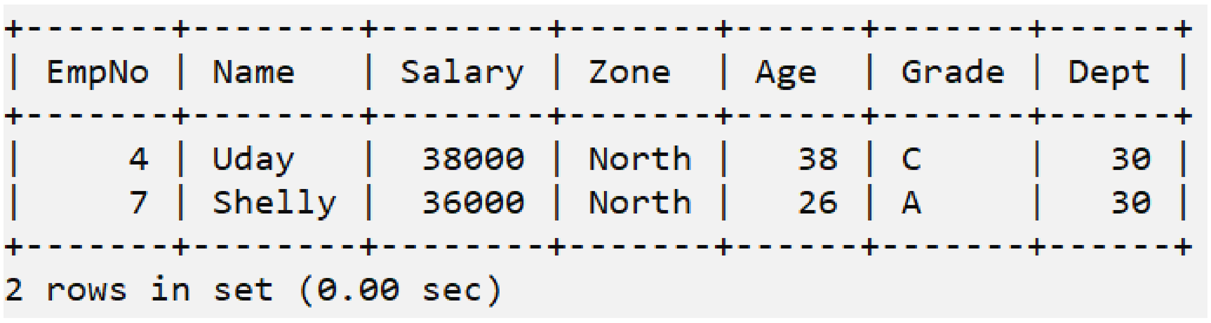
```
+--------+
| Zones  |
+--------+
| West   |
| Centre |
| North  |
| East   |
| South  |
+--------+
5 rows in set (0.00 sec)
```

Q9. Display the details of all employees who are getting a salary of more than 35000 in the department 30.

QUERY:

SELECT * FROM Employee WHERE Salary>35000 AND Dept=30;

OUTPUT:



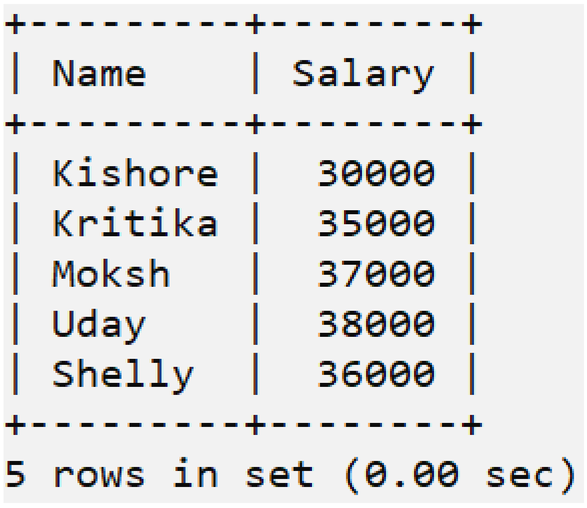
```
+-------+--------+--------+-------+------+-------+------+
| EmpNo | Name   | Salary | Zone  | Age  | Grade | Dept |
+-------+--------+--------+-------+------+-------+------+
|     4 | Uday   |  38000 | North |   38 | C     |   30 |
|     7 | Shelly |  36000 | North |   26 | A     |   30 |
+-------+--------+--------+-------+------+-------+------+
2 rows in set (0.00 sec)
```

Q10. Display the names and salaries of all the employees who are not working in department 20.

QUERY:

SELECT Name, Salary FROM Employee WHERE Dept!=20;

OUTPUT:



```
+---------+--------+
| Name    | Salary |
+---------+--------+
| Kishore |  30000 |
| Kritika |  35000 |
| Moksh   |  37000 |
| Uday    |  38000 |
| Shelly  |  36000 |
+---------+--------+
5 rows in set (0.00 sec)
```

Q11. Display the names and salaries of all the employees who are working neither in West zone nor in Centre zone.

QUERY:

SELECT Name, Salary FROM Employee WHERE Zone NOT IN('West','Centre');

OUTPUT:

```
+--------+--------+
| Name   | Salary |
+--------+--------+
| Uday   |  38000 |
| Nupur  |  32000 |
| Moksh  |  37000 |
| Shelly |  36000 |
+--------+--------+
4 rows in set (0.00 sec)
```

Q12. Display the names of all the employees who are working in department 20 or 30.(use IN operator)

QUERY:

SELECT Name FROM Employee WHERE Dept IN(20,30);

OUTPUT:

```
+--------+
| Name   |
+--------+
| Naveen |
| Nupur  |
| Uday   |
| Shelly |
+--------+
4 rows in set (0.00 sec)
```

Q13. Display the details of all employees whose grade is between A and C.

QUERY:

SELECT * FROM Employee WHERE Grade BETWEEN 'A' AND 'C';

OUTPUT:

```
+-------+---------+--------+--------+------+-------+-------
| EmpNo | Name    | Salary | Zone   | Age  | Grade | Dept
+-------+---------+--------+--------+------+-------+-------
|     1 | Kishore |  30000 | West   |   28 | A     |   10
|     2 | Kritika |  35000 | Centre |   30 | A     |   10
|     4 | Uday    |  38000 | North  |   38 | C     |   30
|     6 | Moksh   |  37000 | South  |   28 | B     |   10
|     7 | Shelly  |  36000 | North  |   26 | A     |   30
+-------+---------+--------+--------+------+-------+-------
5 rows in set (0.00 sec)
```

Q14. Display the name, salary and age of all the employees whose names contain ‘a’.

QUERY:

SELECT Name,Salary,Age FROM Employee WHERE Name LIKE '%a%';

OUTPUT:

```
+---------+--------+------+
| Name    | Salary | Age  |
+---------+--------+------+
| Kritika |  35000 |   30 |
| Naveen  |  32000 |   40 |
| Uday    |  38000 |   38 |
+---------+--------+------+
3 rows in set (0.00 sec)
```

Q15. Display the sum and average of the salaries of all the employees.

QUERY:

SELECT SUM(Salary),AVG(Salary) FROM Employee;

OUTPUT:

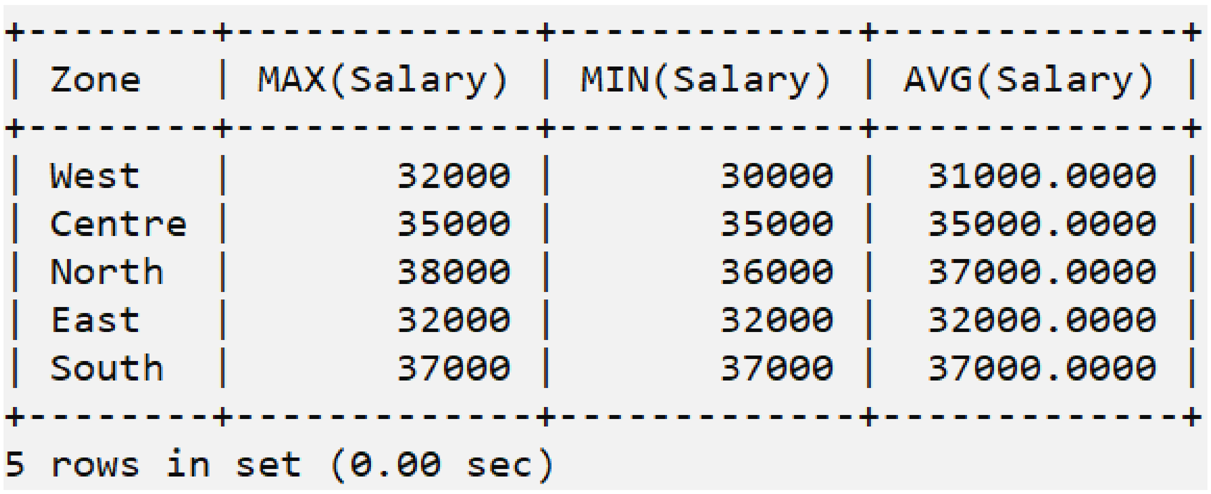
```
+-------------+-------------+
| SUM(Salary) | AVG(Salary) |
+-------------+-------------+
|      240000 |  34285.7143 |
+-------------+-------------+
1 row in set (0.00 sec)
```

Q16. Display the highest, lowest and average salary of each zone.

QUERY:

SELECT Zone,MAX(Salary),MIN(Salary),AVG(Salary) FROM Employee GROUP BY Zone;

OUTPUT:



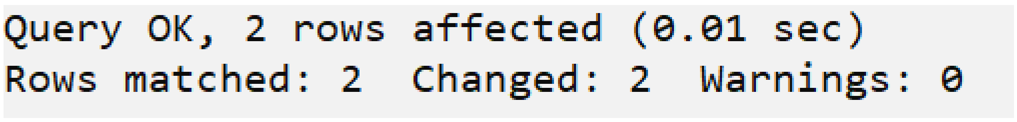
```
+--------+-------------+-------------+-------------+
| Zone   | MAX(Salary) | MIN(Salary) | AVG(Salary) |
+--------+-------------+-------------+-------------+
| West   |       32000 |       30000 |  31000.0000 |
| Centre |       35000 |       35000 |  35000.0000 |
| North  |       38000 |       36000 |  37000.0000 |
| East   |       32000 |       32000 |  32000.0000 |
| South  |       37000 |       37000 |  37000.0000 |
+--------+-------------+-------------+-------------+
5 rows in set (0.00 sec)
```

Q17. Put grade 'B' for all those whose grade is NULL.

QUERY:

UPDATE Employee SET Grade='B' WHERE Grade IS NULL;

OUTPUT:



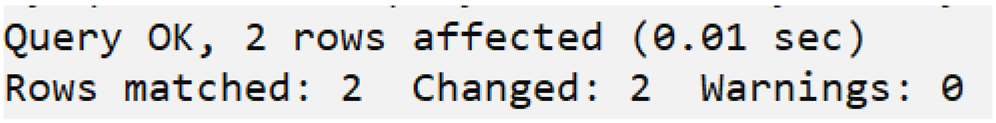
```
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

Q18. Increase the salary of all employees above 30 years of age by 10%.

QUERY:

UPDATE Employee SET Salary=Salary*(10/100)+Salary WHERE Age>30;

OUTPUT:



```
Query OK, 2 rows affected (0.01 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

Q19. Delete the records of all the employees whose grade is C and salary is below 30000.

QUERY:

DELETE FROM Employee WHERE Grade='C' AND Salary<30000;

OUTPUT:

```
Query OK, 0 rows affected (0.00 sec)
```

Q20. Add another column hiredate of type date in the Employee table.

QUERY:

ALTER TABLE Employee ADD hiredate DATE;

OUTPUT:

```
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Q21. Display the details of all employees who work in Sales department.

QUERY:

SELECT Employee.* FROM Employee,Department WHERE Employee.Dept=Department.Dept AND Dname='Sales';

OUTPUT:

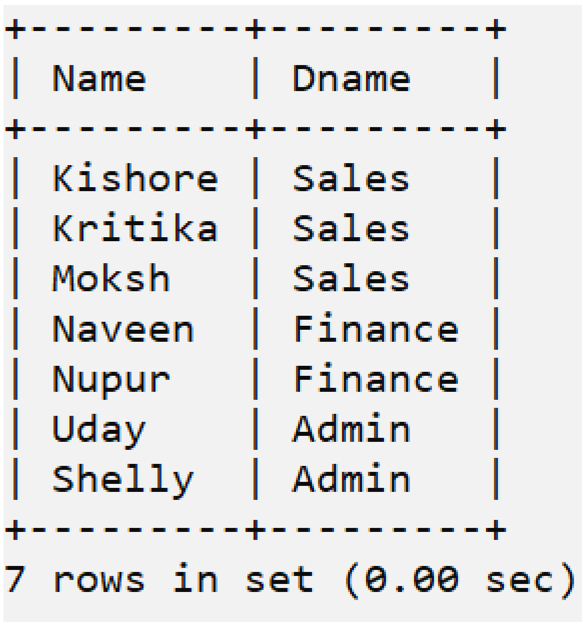
```
+-------+---------+--------+--------+------+-------+------+----------+
| EmpNo | Name    | Salary | Zone   | Age  | Grade | Dept | hiredate |
+-------+---------+--------+--------+------+-------+------+----------+
|     1 | Kishore |  30000 | West   |   28 | A     |   10 | NULL     |
|     2 | Kritika |  35000 | Centre |   30 | A     |   10 | NULL     |
|     6 | Moksh   |  37000 | South  |   28 | B     |   10 | NULL     |
+-------+---------+--------+--------+------+-------+------+----------+
3 rows in set (0.12 sec)
```

Q22. Display the name and department name of all the employees.

QUERY:

SELECT Name,Dname FROM Employee,Department WHERE Employee.Dept=Department.Dept;

OUTPUT:



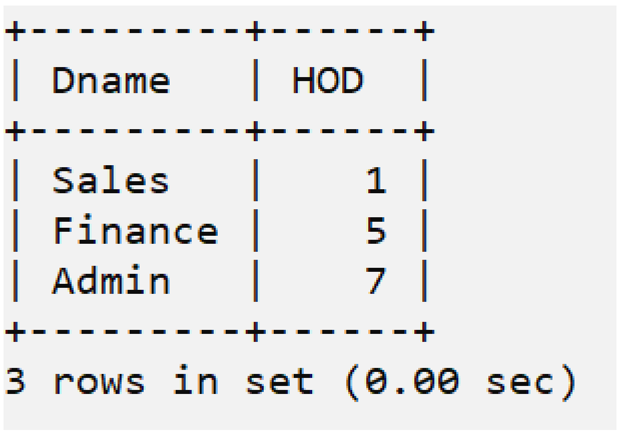
```
+---------+---------+
| Name    | Dname   |
+---------+---------+
| Kishore | Sales   |
| Kritika | Sales   |
| Moksh   | Sales   |
| Naveen  | Finance |
| Nupur   | Finance |
| Uday    | Admin   |
| Shelly  | Admin   |
+---------+---------+
7 rows in set (0.00 sec)
```

Q23. Display the names of the department and the name of the corresponding HOD for all the departments.

<u>QUERY:</u>

SELECT Dname,HOD FROM Department;

<u>OUTPUT:</u>



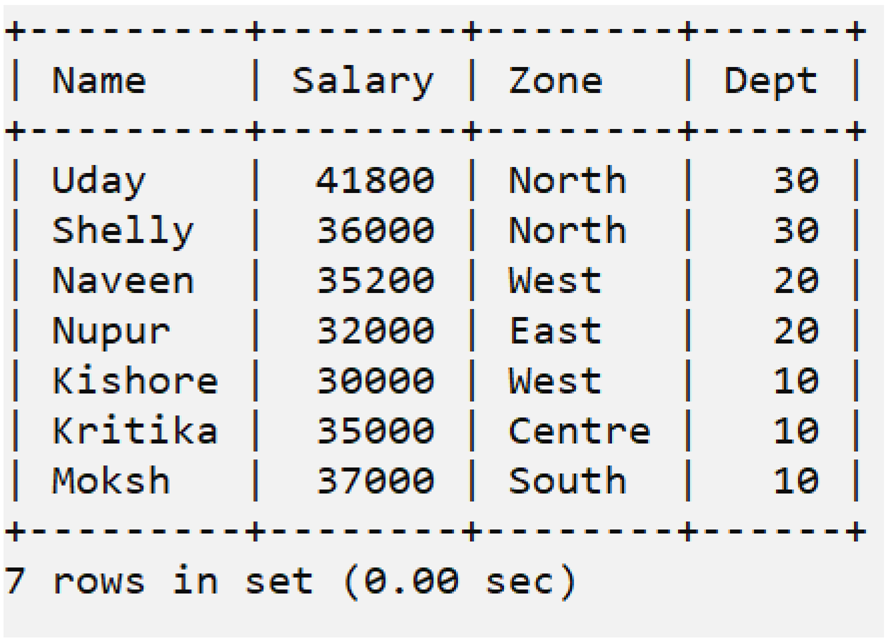
```
+---------+------+
| Dname   | HOD  |
+---------+------+
| Sales   |    1 |
| Finance |    5 |
| Admin   |    7 |
+---------+------+
3 rows in set (0.00 sec)
```

Q24. Display the name, salary and zone of all the employee’s department wise in descending order.

<u>QUERY:</u>

SELECT Name,Salary,Zone,Dept FROM Employee ORDER BY Dept DESC;

<u>OUTPUT:</u>



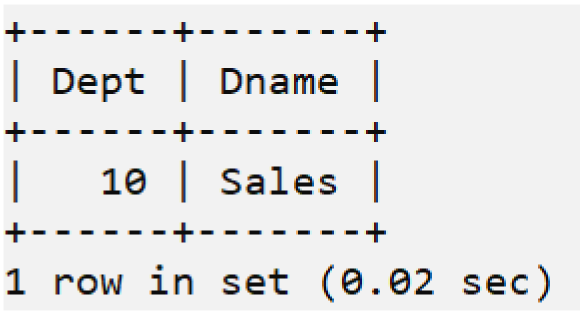
```
+---------+--------+--------+------+
| Name    | Salary | Zone   | Dept |
+---------+--------+--------+------+
| Uday    |  41800 | North  |   30 |
| Shelly  |  36000 | North  |   30 |
| Naveen  |  35200 | West   |   20 |
| Nupur   |  32000 | East   |   20 |
| Kishore |  30000 | West   |   10 |
| Kritika |  35000 | Centre |   10 |
| Moksh   |  37000 | South  |   10 |
+---------+--------+--------+------+
7 rows in set (0.00 sec)
```

Q25. Display the departments where the total number of employees is greater than 2.

QUERY:

SELECT Department.Dept,Dname FROM Department,Employee WHERE Department.Dept=Employee.Dept HAVING COUNT(Employee.Dept)>2;

OUTPUT:



```
+------+-------+
| Dept | Dname |
+------+-------+
|   10 | Sales |
+------+-------+
1 row in set (0.02 sec)
```