PRACTICAL PROJECT FILE
COMPUTER SCIENCE
CLASS: XII Science

Department Of Computer Science
Rabindra Vidya Niketan, NH-49,
Keonjhar
Odisha

Name : Shrinibas Mahanta
ROLL No. : 33

CERTIFICATE

Name: Shrinibas Mahanta Class: XII Science

Roll No: 33

Exam : AISSCE 2022-23

Institution: Rabindra Vidya Niketan, Keonjhar, Odisha

*This is certified to be the bonafide work of the student in the **Computer Science Laboratory** during the academic year **2022-23**.*

*No. of practical certified _____ out of _____ in the subject of **Computer Science**.*

Teacher In-Charge

Date:_____

I N D E X

Teacher's Signature

PFQ01. Write a python program to read a string and print:

a) No. of alphabet
b) No. of uppercase
c) No. of lowercase
d) No. of digits

Program.

```
#Taking string as user input.
string1 = str(input("Enter the string to be evaluated:"))
# pre defining the calculating constants.
alpha = 0
upper = 0
lower = 0
digit = 0
# defining the logic for calculating:
#       1.alphabets
#       2.uppeercase
#       3.Lowercase
#       4.Digits.

for i in string1:
    if i.isalpha():
        alpha+=1
    if i.isupper():
        upper+=1
    if i.islower():
        lower+=1
    if i.isdigit():
        digit+=1
# Print statement to Print out calculated objects.
print(f"Number of alphabets: {alpha}\n",
      f"Number of uppercase: {upper}\n",
      f"Number of lowercase: {lower}\n",
      f"Number of digits: {digit}")
```

Output.

```
[sm2k4@fedora School Project codes] $ python3 stringeval.py
Enter the string to be evaluated:thisISASTring12345
Number of alphabets: 13
 Number of uppercase: 5
 Number of lowercase: 8
 Number of digits: 5
```

Signature

PFQ02. Write a python program to accept a dictionary by user containing 10 no.s names and classes as key - value pairs and display the dictionary.

Program.

```
# Initialising the dictionary.
dict1 = {}
i = int(input("Enter the number of names to store:"))
# creating a loop to take the name:value pairs as user input.
while i > 0:
    dict2 = {str(input("Enter the name:")):int(input("Enter the roll number:"))}
    dict1.update(dict2)
    i-=1
# To print the dictionary items.
print("The dictionary is:")
print(dict1)

print()

#storing the items in a list
list1 = dict1.items()
#loop to print the list items.
print("The dictionary items are:")
for i in list1:
    print(i)
```

Output.

```
[sm2k4@fedora School Project codes]$ python3 dictkv.py
Enter the number of names to store:10
Enter the name:sh
Enter the roll number:13
Enter the name:ab
Enter the roll number:14
Enter the name:sa
Enter the roll number:15
Enter the name:lk
Enter the roll number:16
Enter the name:pcj
Enter the roll number:17
Enter the name:abn
Enter the roll number:18
Enter the name:abh
Enter the roll number:19
Enter the name:sat
Enter the roll number:20
Enter the name:py
Enter the roll number:21
Enter the name:pu
Enter the roll number:22
The dictionary is:
{'sh': 13, 'ab': 14, 'sa': 15, 'lk': 16, 'pcj': 17, 'abn': 18, 'abh': 19, 'sat': 20, 'py': 21, 'pu': 2

The dictionary items are:
('sh', 13)
('ab', 14)
('sa', 15)
('lk', 16)
('pcj', 17)
('abn', 18)
('abh', 19)
('sat', 20)
('py', 21)
('pu', 22)
[sm2k4@fedora School Project codes]$
```

Signature

PFQ03. Write a python function to accept principal, rate and time by the user and find out the Simple Interest and Compound Interest.

Program.

```
# Defining the simple interest function.
def simpleI(p,r,t=1):
    return (p*r*t)/100
# Defining the compound interest function.
def compoundI(p,r,t=1):
    return (p*(1+r/100)**t)-p
# Taking principle,rate and time as user input../time is defaulted to 1 if no user input given.
x = int(input("Enter the principle amount:"))
y = int(input("Enter the rate of interest:"))
z = int(input("Enter the time of interest calculation:"))

#Print simple interest.
print(f"The simple interest calculated is: {simpleI(x, y,z)}")
#Print compound interest.
print(f"The compound interest calculated is:{compoundI(x, y,z)}")
```

Output.

```
[sm2k4@fedora School Project codes]$ python3 interest.py
Enter the principle amount:5000
Enter the rate of interest:5
Enter the time of interest calculation:2
The simple interest calculated is: 500.0
The compound interest calculated is:512.5
[sm2k4@fedora School Project codes]$
```

Signature

PFQ04. Write a python function to calculate the factorial of a given number.

Program.

```
def fact():
    num=int(input("Enter your number:"))
    fact=num
    while num != 1:
        num=num-1
        fact=fact*num
    return fact
print(f"The factorial of the number is :{fact()}")
```

Output.

```
[sm2k4@fedora School Project codes] $ python3 fact.py
Enter your number:5
The factorial of the number is :120
[sm2k4@fedora School Project codes] $ python3 fact.py
Enter your number:6
The factorial of the number is :720
```

Signature

PFQ05. Write a python function that accepts a string and calculate the number of uppercase and lowercase letters.

Program.

```
# Define the function to accept a string and calculate uppercase and lowercase characters.

def uplow(str1):
    up =0
    low=0
    #iterating through each character.
    for i in str1:
        #checking each character
        if i.isupper():
            up+=1
        if i.islower():
            low+=1

    return up,low
string1 = str(input("Enter the stirng:"))
print(f"The number of uppercase and lowercase characters are:{uplow(string1)}")
```

Output.

```
[sm2k4@fedora School Project codes]$ python3 uplow.py
Enter the string:HeAvEnHeLl
The number of uppercase and lowercase characters are:(5, 5)
[sm2k4@fedora School Project codes]$
```

Signature

PFQ06. Write a python function that checks whether a passed string is palindrome or not.

Program.

```
def strinv(str1):
    rev = str1[::-1]
    if str1 == rev:
        return 0
    else:
        return 1

string = str(input("Enter the string to be checked:"))

if strinv(string) == 0:
    print(f"The string {string} is a palindrome.")
else:
    print(f"The string {string} is not palindrome.")
```

Output.

```
[sm2k4@fedora School Project codes]$ python3 strpalindrome.py
Enter the string to be checked:hannah
The string hannah is a palindrome.
[sm2k4@fedora School Project codes]$ python3 strpalindrome.py
Enter the string to be checked:heaven
The string heaven is not palindrome.
[sm2k4@fedora School Project codes]$
```

Signature

PFQ07. Write a python function to count the total number of words in a text file.

Program.

```
def wrdcnt():
    file = open("text_file.txt","r")
    cont = file.read()
    words = 0
    for i in cont:
        if i.isspace():
            words+=1
        elif i == ".":
            words+=1
        else:
            continue
    return words
    file.close

print(f"The total number of words in the file are: {wrdcnt()}")
```

Output.

```
[sm2k4@fedora Count txt]$ python3 count.py
The total number of words in the file are: 20
[sm2k4@fedora Count txt]$
```

Signature

PFQ08. *Write a python function to read from text file "story.txt" and calculate and display the sum of all the even digits present in the file.*

Program.

```
def sumev():
    file = open("story.txt","r")
    cont = file.read()
    sumev = 0
    for i in cont:
        if i.isdigit() and int(i)%2 == 0 :
            sumev+=int(i)
        else:
            continue
    file.close()
    return sumev
print(f"The sum of all even numbers is: {sumev()}")
```

Output.

```
[sm2k4@fedora evencnt]$ python3 even.py
The sum of all even numbers is: 84
[sm2k4@fedora evencnt]$
```

Signature

PFQ09. Write a python function to count the number of Upper-Case alphabets in a text file "poem.txt".

Program.

```
def uppernum():
    file = open("poem.txt","r")
    cont = file.read()
    uppernum = 0
    for i in cont:
        if i.isupper():
            uppernum+=1
        else:
            continue
    file.close()
    return uppernum
print(f"The number of uppercase letters in the file 'poem.txt' are {uppernum()}")
```

Output.

```
[sm2k4@fedora upperpoem]$ python3 uppercnt.py
The number of uppercase letters in the file 'poem.txt' are 5
[sm2k4@fedora upperpoem]$
```

Signature

PFQ10. Write a python program to write and read data from a binary file consisting roll no, name, marks and Append more records in the same file.

Program.

```
import pickle
# a function to create/open a binary file and write content to it.
def write_bin():
    fo = open("binary.dat","wb")
    # number of file entry as a user input.
    ent = int(input("How many datas to store?: "))
    i = 0
    # loops through to get  userinput and dumps it in the binary file as a list.
    while i < ent :
        a = input("Name: " )
        b = int(input("Roll.no: "))
        c = int(input("Marks: "))
        dat = [a,b,c]
        pickle.dump(dat,fo)
        i+=1
    fo.close()
# function to append content to end of a binary file.
def append_bin():
    fo = open("binary.dat","ab")
    # number of file entry as a user input.
    ent = int(input("How many datas to Add?: "))
    i = 0
    # loops through to get user input and add it into the existing binary file by dumping it as a list.
    while i < ent :
        a = input("Name: " )
        b = int(input("Roll.no: "))
        c = int(input("Marks: "))
        dat = [a,b,c]
        pickle.dump(dat,fo)
        i+=1
    fo.close()
# function to read the content of a binary file.
def read_bin():
    fo = open("binary.dat","rb")
    # while true loops through and tries to load the file content from the binary file unless a exception is encountered.
    while True:
        try:
            x = pickle.load(fo)
            t = True
            # while true loops through the file content and prints the file content in the specified format.
            while t:
                name = x[0]
                roll = x[1]
                marks = x[2]
                print(f"Name of candidate: {name}\nRoll.no of candidate:{roll}\nMarks of candidate: {marks}")
                t =False
        # if exception is encountered breaks out of the loop and stops the function.
        except EOFError:
            break
    fo.close()
# Driver Code.
write_bin()
read_bin()
append_bin()
read_bin()
```

Output.

```
[sm2k4@fedora rwbin]$ python3 bin.py
How many datas to store?: 2
Name: sm
Roll.no: 33
Marks: 99
Name: sg
Roll.no: 100
Marks: 200
Name of candidate: sm
Roll.no of candidate:33
Marks of candidate: 99
Name of candidate: sg
Roll.no of candidate:100
Marks of candidate: 200
How many datas to Add?: 2
Name: ab
Roll.no: 25
Marks: 250
Name: lk
Roll.no: 29
Marks: 251
Name of candidate: sm
Roll.no of candidate:33
Marks of candidate: 99
Name of candidate: sg
Roll.no of candidate:100
Marks of candidate: 200
Name of candidate: ab
Roll.no of candidate:25
Marks of candidate: 250
Name of candidate: lk
Roll.no of candidate:29
Marks of candidate: 251
[sm2k4@fedora rwbin]$
```

Signature

PFQ11. Write a python program to write and read data from a binary file consisting roll no, name, marks and Search for a records from the file.

Program.

```
import pickle
# a function to create/open a binary file and write content to it.
def write_bin():
    fo = open("binary.dat","wb")
    # number of file entry as a user input.
    ent = int(input("How many datas to store?: "))
    i = 0
    # loops through to get  userinput and dumps it in the binary file as a list.
    while i < ent :
        a = input("Name: " )
        b = int(input("Roll.no: "))
        c = int(input("Marks: "))
        dat = [a,b,c]
        pickle.dump(dat,fo)
        i+=1
    fo.close()
# function to read the content of a binary file.
def read_bin():
    fo = open("binary.dat","rb")
    # while true loops through and tries to load the file content from the binary file unless a exception is encountered.
    while True:
        try:
            x = pickle.load(fo)
            t = True
            # while true loops through the file content and prints the file content in the specified format.
            while t:
                name = x[0]
                roll = x[1]
                marks = x[2]
                print(f"Name of candidate: {name}\nRoll.no of candidate:{roll}\nMarks of candidate: {marks}")
                t =False
        # if exception is encountered breaks out of the loop and stops the function.
        except EOFError:
            break
    fo.close()
# function to search the desired content of a file.
def search_bin():
    name = str(input("Enter the name to search for:"))
    fo = open("binary.dat","rb")
    # while true loops through and tries to load the file content from the binary file unless a exception is encountered.
    while True:
        try:
            x = pickle.load(fo)
            t = True
                else:
                    break
        # if exception is encountered breaks out of the loop and stops the function.
        except EOFError:
            break
    fo.close()

#Driver Code
write_bin()
read_bin()
search_bin()
```

Output.

```
[sm2k4@fedora rwsbin]$ python3 search.py
How many datas to store?: 5
Name: foo
Roll.no: 23
Marks: 43
Name: foob
Roll.no: 24
Marks: 44
Name: fooob
Roll.no: 25
Marks: 55
Name: fobo
Roll.no: 26
Marks: 66
Name: fboo
Roll.no: 33
Marks: 55
Name of candidate: foo
Roll.no of candidate:23
Marks of candidate: 43
Name of candidate: foob
Roll.no of candidate:24
Marks of candidate: 44
Name of candidate: fooob
Roll.no of candidate:25
Marks of candidate: 55
Name of candidate: fobo
Roll.no of candidate:26
Marks of candidate: 66
Name of candidate: fboo
Roll.no of candidate:33
Marks of candidate: 55
Enter the name to search for:fobo
The Searched Item Is:['fobo', 26, 66]
```

Signature

PFQ12. Write a python program to write and read data from a binary file consisting roll no, name, marks and Update a particular record in the file.

Program.

```
"""
function for writing onto the binary file.Cursor starts from the end.
"""
import pickle
def write_bin():
    fo = open("./binary.dat","wb+")
    lines = int(input("Number of records to enter:"))
    arr = []
    i = 0
    while i<lines:
        name = str(input("Enter Name:"))
        roll = str(input("Enter Roll:"))
        mark = str(input("Enter Marks:"))
        dat = [name,roll,mark]
        arr.append(dat)
        i+=1
    pickle.dump(arr,fo)
    fo.close()
"""
function for reading the existing binary file.Cursor starts from the begenning.
"""

def read_bin():
    fo = open("./binary.dat","rb+")
    file = pickle.load(fo)
    print("\nThe records are:")

    for i in file:
        name = i[0]
        roll = i[1]
        mark = i[2]
        print(f"\nName:{name}\nRoll:{roll}\nMark:{mark}\n")
    fo.close()

"""
function to update a record from the records.Cursor starts from the begenning.
"""
def update_bin():
    fo = open("./binary.dat","rb+")
    fo.seek(0)
    file = pickle.load(fo)
    wupd = str(input("Enter What To Update:"))
    for i in file:
        for j in i:
            if j == wupd:
                upd = str(input("Enter Update:"))
                index = i.index(j)
                i[index] = upd
    fo.seek(0)
    pickle.dump(file,fo)
    fo.close()


write_bin()
read_bin()
update_bin()
read_bin()
```

Output.

```
[sm2k4@fedora rwubin]$ python3 update.py
Number of records to enter:3
Enter Name:hl
Enter Roll:33
Enter Marks:44
Enter Name:hv
Enter Roll:55
Enter Marks:66
Enter Name:ad
Enter Roll:77
Enter Marks:88

The records are:

Name:hl
Roll:33
Mark:44


Name:hv
Roll:55
Mark:66


Name:ad
Roll:77
Mark:88

Enter What To Update:ad
Enter Update:hu

The records are:

Name:hl
Roll:33
Mark:44


Name:hv
Roll:55
Mark:66


Name:hu
Roll:77
Mark:88
```

Signature

PFQ13. Write a python program to write and read data from a c.s.v file consisting Item-No, Item-Name, Quantity and Price. Write functions to append new records and search for particular record using Item-No.

Program.

```
import csv

def write():
    # writing into csv files.

    file = open("csvfile.csv",'w')
    headings = ['itemNo','itemName','quantity','price']
    rows = []
    doAdd = "y"
    while doAdd == "y":
        no = input("Enter itemNo:")
        name = input("Enter itemName:")
        quantity = input("Enter Quantity:")
        price = input("Enter Price:")
        row = [no,name,quantity,price]
        rows.append(row)
        doAdd =input("Want to Add(y/n):")


    writer = csv.writer(file)
    writer.writerow(headings)
    writer.writerows(rows)
    file.close()

def read():
    # read from csv file.
    file = open("csvfile.csv",'r')
    reader = csv.reader(file)
    print("Reading CSV File:")
    headings = next(reader)

    print("File Format:",headings)
    rows = []

    for i in reader:
        for j in i:
            print(j,end=',')
        print()
    file.close()
```

```
def append():
    # appending  into existing csv files.

    file = open("csvfile.csv",'a')
    rows = []
    doAdd = "y"
    while doAdd == "y":
        no = input("Enter itemNo:")
        name = input("Enter itemName:")
        quantity = input("Enter Quantity:")
        price = input("Enter Price:")
        row = [no,name,quantity,price]
        rows.append(row)
        doAdd =input("Want to Add(y/n):")


    writer = csv.writer(file)
    writer.writerows(rows)
    file.close()


def scan():
    # Scanning from a existing CSV file.
    file = open("csvfile.csv",'r')
    ino = input("Enter the itemNo to search for:")
    reader = csv.reader(file)
    for i in reader:
        if ino == i[0]:
            print("The Searched column is:",i)
    file.close


#Driver Code

write()
append()
read()
scan()
```

Output.

```
sm2k4@fedora ~/c/s/S/csvwork (master)> python3 csvwork.py
Enter itemNo:01
Enter itemName:pen
Enter Quantity:5
Enter Price:100
Want to Add(y/n):y
Enter itemNo:02
Enter itemName:pencil
Enter Quantity:5
Enter Price:50
Want to Add(y/n):n
Enter itemNo:03
Enter itemName:eraser
Enter Quantity:5
Enter Price:50
Want to Add(y/n):n
Reading CSV File:
File Format: ['itemNo', 'itemName', 'quantity', 'price']
01,pen,5,100,
02,pencil,5,50,
03,eraser,5,50,
Enter the itemNo to search for:03
The Searched column is: ['03', 'eraser', '5', '50']
```

Signature

PFQ14.Write a Menu Driven program to implement a stack for book – details(Book-No, Book-Name). Implement PUSH, POP and DISPLAY operations.

Program.

```
import time

bookstr = []
def welcome():
    print("\nWelcome to Book Library.\n")
    print("Menu:")
    print(
        "1.Add Book\n"
        "2.Remove Last Added Book\n"
        "3.Display Books\n"
        "4.Exit\n"

        )

def main():
    menu = input("Enter The command from menu:")
    if menu =="1":
        push()
    elif menu =="2":
        print("Last added element removed.")
        pop()
    elif menu == "3":
        if len(bookstr) == 0:
            print("Empty library")
        else:
            print("The books are:")
            for i in bookstr:
                print(i[1])
    elif menu == "4":
        print("Exiting the program:")
        exit()
    else:
        print("Invalid command.")

def push():
    bookno = input("Enter Book-Number:")
    book = input("Enter Book-Name:")
    bookdet = [bookno, book]
    bookstr.append(bookdet)
    #print(bookstr)

def pop():
    try:
        bookstr.pop()
        #print(bookstr)
    except IndexError:
        print("There is nothing to remove.")
```

```
#Driver code:

while True:
    time.sleep(0.2)
    welcome()
    main()
```

Output.

```
  Welcome to Book Library.

  Menu:
  1.Add Book
  2.Remove Last Added Book
  3.Display Books
  4.Exit

  Enter The command from menu:1
  Enter Book-Number:1
  Enter Book-Name:The Lord of the Rings

  Welcome to Book Library.

  Menu:
  1.Add Book
  2.Remove Last Added Book
  3.Display Books
  4.Exit

  Enter The command from menu:1
  Enter Book-Number:2
  Enter Book-Name:CODE

Welcome to Book Library.

Menu:
1.Add Book
2.Remove Last Added Book
3.Display Books
4.Exit

Enter The command from menu:3
The books are:
The Lord of the Rings
CODE

Welcome to Book Library.

Menu:
1.Add Book
2.Remove Last Added Book
3.Display Books
4.Exit

Enter The command from menu:2
Last added element removed.

Welcome to Book Library.

Menu:
1.Add Book
2.Remove Last Added Book
3.Display Books
4.Exit

Enter The command from menu:4
Exiting the program:
```

Signature

PFQ15.Create a table ITEM with Code as the primary key. Decide your own data types. Insert records as shown in the table below and write the queries to perform the following functions on the table.

Table: ITEM

Code	ItemName	Company	Qty	Price	ExpiryDate
1002	Cake	Britannia	45	200	2013-01-12
1005	Biscuit	Britannia	90	100	2012-12-12
1006	Jam	Kissan	34	160	2013-01-23
1001	Jelly	Nestle	23	150	2012-11-21
1007	Sauce	Kissan	120	260	2013-02-15
1003	Maggi	Nestle	80	100	2013-02-10
1004	Chocolate	Cadbury	100	200	2012-12-27

Program.

```
MariaDB [sm2k4]> create table item(Code int(4) primary key, ItemName varchar(20), Company varchar(20), Qty int(4), Price int(6), ExpiryDate DATE);
Query OK, 0 rows affected (0.017 sec)

MariaDB [sm2k4]> insert into item values (1002, 'Cake', 'Britannia', 45, 200, '2013-01-12'), (1005, 'Biscuit','Britannia', 90, 100, '2012-12-12'), (1006, 'Jam', 'Kissan', 34, 160, '2013-01-23
'),(1001,'Jelly','Nestle',23,150,'2012-11-21'), (1007, 'Sauce', 'Kissan', 120, 260, '2013-02-15'), (1003, 'Maggi', 'Nestle', 80, 100, '2013-02-10'), (1004, 'Chocolate', 'Cadbury', 100, 200, '
2012-12-27');
Query OK, 7 rows affected (0.010 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

Output.

a. Display the details of the items in ascending order of Code.

```
MariaDB [sm2k4]> select * from item order by code;
+------+-----------+-----------+------+-------+------------+
| Code | ItemName  | Company   | Qty  | Price | ExpiryDate |
+------+-----------+-----------+------+-------+------------+
| 1001 | Jelly     | Nestle    |   23 |   150 | 2012-11-21 |
| 1002 | Cake      | Britannia |   45 |   200 | 2013-01-12 |
| 1003 | Maggi     | Nestle    |   80 |   100 | 2013-02-10 |
| 1004 | Chocolate | Cadbury   |  100 |   200 | 2012-12-27 |
| 1005 | Biscuit   | Britannia |   90 |   100 | 2012-12-12 |
| 1006 | Jam       | Kissan    |   34 |   160 | 2013-01-23 |
| 1007 | Sauce     | Kissan    |  120 |   260 | 2013-02-15 |
+------+-----------+-----------+------+-------+------------+
7 rows in set (0.000 sec)
```

b. Display Code and ItemName of items that have price in the 170 to 250 range.

```
MariaDB [sm2k4]> select Code,ItemName from item where Price between 170 and 250;
+------+-----------+
| Code | ItemName  |
+------+-----------+
| 1002 | Cake      |
| 1004 | Chocolate |
+------+-----------+
2 rows in set (0.000 sec)
```

c. Display the ItemName and Qty of all the products Expired in 2013.

```
MariaDB [sm2k4]> select ItemName,Qty from item where year(ExpiryDate) = '2013';
+----------+------+
| ItemName | Qty  |
+----------+------+
| Cake     |   45 |
| Maggi    |   80 |
| Jam      |   34 |
| Sauce    |  120 |
+----------+------+
4 rows in set (0.000 sec)
```

d. Increase the price of all the items by Rs. 5

```
MariaDB [sm2k4]> update item set price = price +5;
Query OK, 7 rows affected (0.012 sec)
Rows matched: 7  Changed: 7  Warnings: 0
```

e. Display the various company listed in the table.

```
+-----------+
| Company   |
+-----------+
| Nestle    |
| Britannia |
| Nestle    |
| Cadbury   |
| Britannia |
| Kissan    |
| Kissan    |
+-----------+
7 rows in set (0.000 sec)
```

Signature.

PFQ16. Create a table CLUB with CoachID as the primary key. Decide your own data types. Insert records as shown in the table below and write the queries for the following:

Table: CLUB

CoachID	CoachName	Sports	DateOfApp	Salary	Gender
1001	Ravindra	Karate	1990-03-27	12000	M
1002	Ambika	Karate	1998-01-20	30000	F
1003	Nitin	Squash	1998-02-19	15000	M
1004	Rohit	Basketball	1999-04-23	18000	M
1005	Mohan	Swimming	1998-02-24	30000	M
1006	Saumya	Swimming	2001-01-22	15000	F
1007	Garima	Karate	2010-02-27	5600	F
1008	Shailja	Basketball	2010-05-29	8500	F

Program.

```
7 rows in set (0.000 sec)

MariaDB [sm2k4]> create table club (CoachID int(5) primary key, CoachName varchar(20), Sports varchar(20), DateOfAppl DATE, Salary int(10), Gender varchar(1));
Query OK, 0 rows affected (0.033 sec)

MariaDB [sm2k4]> insert into club values (1001,'Rabindra','karate','1990-03-27',12000,'M'), (1002,'Ambika','karate','1998-01-20',30000,'F'), (1003,'Nitin','Squash','1998-02-19',15000,'M'), (1
004,'Rohit','Basketball','1999-04-23',18000,'M'), (1005,'Mohan','Swimming','1998-02-24',30000,'M'), (1006,'Saumya','Swimming','2001-01-22',15000,'F'), (1007,'Garima','karate','2010-02-27',560
0,'F'), (1008,'Shailja','Basketball','2010-05-29',8500,'F');
Query OK, 8 rows affected (0.010 sec)
Records: 8  Duplicates: 0  Warnings: 0
```

Output.

a. Display the sum of salaries of the female and the male coaches.

```
MariaDB [sm2k4]> select Gender,sum(Salary) from club where Gender = 'F';
+--------+-------------+
| Gender | sum(Salary) |
+--------+-------------+
| F      |       59100 |
+--------+-------------+
1 row in set (0.001 sec)

MariaDB [sm2k4]> select Gender,sum(Salary) from club where Gender = 'M';
+--------+-------------+
| Gender | sum(Salary) |
+--------+-------------+
| M      |       75000 |
+--------+-------------+
1 row in set (0.000 sec)
```

b. Display the maximum and minimum salaries of the Karate coaches.

```
MariaDB [sm2k4]> select max(Salary),min(Salary) from club where Sports = 'karate';
+-------------+-------------+
| max(Salary) | min(Salary) |
+-------------+-------------+
|       30000 |        5600 |
+-------------+-------------+
1 row in set (0.000 sec)
```

c. Display the numbers of male and female coaches.

```
MariaDB [sm2k4]> select count(Gender) from club where Gender = 'F';
+---------------+
| count(Gender) |
+---------------+
|             4 |
+---------------+
1 row in set (0.001 sec)

MariaDB [sm2k4]> select count(Gender) from club where Gender = 'M';
+---------------+
| count(Gender) |
+---------------+
|             4 |
+---------------+
1 row in set (0.000 sec)
```

d. Display the details of all coaches whose name end with ‘a’.

```
MariaDB [sm2k4]> select * from club where CoachName like '%a';
+---------+-----------+------------+------------+--------+--------+
| CoachID | CoachName | Sports     | DateOfAppl | Salary | Gender |
+---------+-----------+------------+------------+--------+--------+
|    1001 | Rabindra  | karate     | 1990-03-27 |  12000 | M      |
|    1002 | Ambika    | karate     | 1998-01-20 |  30000 | F      |
|    1006 | Saumya    | Swimming   | 2001-01-22 |  15000 | F      |
|    1007 | Garima    | karate     | 2010-02-27 |   5600 | F      |
|    1008 | Shailja   | Basketball | 2010-05-29 |   8500 | F      |
+---------+-----------+------------+------------+--------+--------+
5 rows in set (0.000 sec)
```

e. Display the details of all the coaches associated with sports starting with ‘S’.

```
MariaDB [sm2k4]> select CoachID,CoachName,Sports from club where CoachName like 'S%';
+---------+-----------+------------+
| CoachID | CoachName | Sports     |
+---------+-----------+------------+
|    1006 | Saumya    | Swimming   |
|    1008 | Shailja   | Basketball |
+---------+-----------+------------+
2 rows in set (0.001 sec)
```

Signature.

PFQ17. Create a table PATIENT with Pcode (patient code) as the primary key. Decide your own data types. Insert records as shown in the table and write the outputs for the following:

Table: PATIENT

Pcode	Name	Age	Dept	DOA	Charge	Gender
P1	Karan	24	Surgery	2010-07-07	5000	M
P2	Varun	45	Orthopedic	2010-12-19	8000	M
P3	Ravina	12	Orthopedic	2010-01-15	8000	F
P4	Ankita	36	Surgery	2009-04-16	12000	F
P5	Ketan	16	ENT	2009-07-31	25000	M
P6	Arvind	29	ENT	2010-07-07	15000	M
P7	Zugal	45	Cardiology	2010-10-20	14000	M

Program.

```
MariaDB [sm2k4]> create table patient (Pcode varchar(3),Name varchar(20),Age int, Dept varchar(20),DOA DATE, Charge int,Gender varchar(1));
Query OK, 0 rows affected (0.016 sec)

MariaDB [sm2k4]> insert into patient values ('P1','Karan',24,'Surgery','2010-07-07',5000,'M'), ('P2','Varun',45,'Orthopedic','2010-12-19',8000,'M'), ('P3','Ravina',12,'Orthopedic','2010-01-15
',8000,'F'), ('P4','Ankita',36,'Surgery','2009-04-16',12000,'F'), ('P5','Ketan',16,'ENT','2009-07-31',25000,'M'), ('P6','Arvind',29,'ENT','2010-07-07',15000,'M'), ('P7','Zugal',45,'Cardiology
','2010-10-20',14000, 'M');
Query OK, 7 rows affected (0.010 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

Output.

a. SELECT * FROM PATIENT WHERE AGE < 25 or AGE > 40;

```
MariaDB [sm2k4]> SELECT * FROM patient WHERE Age < 25 or Age > 40;
+-------+--------+------+------------+------------+--------+--------+
| Pcode | Name   | Age  | Dept       | DOA        | Charge | Gender |
+-------+--------+------+------------+------------+--------+--------+
| P1    | Karan  |   24 | Surgery    | 2010-07-07 |   5000 | M      |
| P2    | Varun  |   45 | Orthopedic | 2010-12-19 |   8000 | M      |
| P3    | Ravina |   12 | Orthopedic | 2010-01-15 |   8000 | F      |
| P5    | Ketan  |   16 | ENT        | 2009-07-31 |  25000 | M      |
| P7    | Zugal  |   45 | Cardiology | 2010-10-20 |  14000 | M      |
+-------+--------+------+------------+------------+--------+--------+
5 rows in set (0.000 sec)
```

b. SELECT * FROM PATIENT WHERE DEPT IN ('ENT', 'Surgery');

```
MariaDB [sm2k4]> SELECT * FROM patient WHERE Dept IN ('ENT', 'Surgery');
+-------+--------+------+---------+------------+--------+--------+
| Pcode | Name   | Age  | Dept    | DOA        | Charge | Gender |
+-------+--------+------+---------+------------+--------+--------+
| P1    | Karan  |   24 | Surgery | 2010-07-07 |   5000 | M      |
| P4    | Ankita |   36 | Surgery | 2009-04-16 |  12000 | F      |
| P5    | Ketan  |   16 | ENT     | 2009-07-31 |  25000 | M      |
| P6    | Arvind |   29 | ENT     | 2010-07-07 |  15000 | M      |
+-------+--------+------+---------+------------+--------+--------+
4 rows in set (0.000 sec)
```

c. SELECT MIN(CHARGE), MAX(CHARGE), SUM(CHARGE) FROM PATIENT;

```
MariaDB [sm2k4]> SELECT MIN(Charge), MAX(Charge), SUM(Charge) FROM patient;
+-------------+-------------+-------------+
| MIN(Charge) | MAX(Charge) | SUM(Charge) |
+-------------+-------------+-------------+
|        5000 |       25000 |       87000 |
+-------------+-------------+-------------+
1 row in set (0.000 sec)
```

d. SELECT COUNT(*) FROM PATIENT GROUP BY GENDER;

```
MariaDB [sm2k4]> select * from patient;
+-------+--------+------+------------+------------+--------+--------+
| Pcode | Name   | Age  | Dept       | DOA        | Charge | Gender |
+-------+--------+------+------------+------------+--------+--------+
| P1    | Karan  |   24 | Surgery    | 2010-07-07 |   5000 | M      |
| P2    | Varun  |   45 | Orthopedic | 2010-12-19 |   8000 | M      |
| P3    | Ravina |   12 | Orthopedic | 2010-01-15 |   8000 | F      |
| P4    | Ankita |   36 | Surgery    | 2009-04-16 |  12000 | F      |
| P5    | Ketan  |   16 | ENT        | 2009-07-31 |  25000 | M      |
| P6    | Arvind |   29 | ENT        | 2010-07-07 |  15000 | M      |
| P7    | Zugal  |   45 | Cardiology | 2010-10-20 |  14000 | M      |
+-------+--------+------+------------+------------+--------+--------+
7 rows in set (0.000 sec)
```

e. SELECT DEPT, COUNT(*) FROM PATIENT GROUP BY DEPT;

```
MariaDB [sm2k4]> SELECT Dept, COUNT(*) FROM patient GROUP BY Dept;
+------------+----------+
| Dept       | COUNT(*) |
+------------+----------+
| Cardiology |        1 |
| ENT        |        2 |
| Orthopedic |        2 |
| Surgery    |        2 |
+------------+----------+
4 rows in set (0.001 sec)
```

Signature.

PFQ18. Given the following tables for a database **LIBRARY**:

Table: BOOKS

Book_ID	Book_Name	Author_Name	Publishers	Price	Type	Quantity
C0001	Fast Cook	Lata kapoor	EPB	355	Cookery	5
F0001	The Tears	Willium Hopkins	First Publ.	650	Fiction	20
T0001	My First C++	Brain & Brooke	EPB	350	Text	10
T0002	C++ Brainworks	A.W. Rossaine	TDH	350	Text	15
F0002	Thunderbolts	Anna Roberts	First Publ.	750	Fiction	50

Table: ISSUED

Book_ID	Quantity_Issued
T0001	4
C0001	5
F0001	2

Write SQL queries for (a) to (f):

a. To show Book name, Author name and price of books of EPB Publishers.

```
MariaDB [sm2k4]> select Book_Name,Author_Name,Price from books where Publishers = 'EPB';
+-------------+---------------+-------+
| Book_Name   | Author_Name   | Price |
+-------------+---------------+-------+
| Fast Cook   | Lata Kapoor   |   355 |
| My First C++ | Brain & Brooke |   350 |
+-------------+---------------+-------+
2 rows in set (0.001 sec)
```

b. To list the names from books of Fiction type.

```
MariaDB [sm2k4]> select Book_Name from books where Type = 'Fiction';
+--------------+
| Book_Name    |
+--------------+
| The Tears    |
| Thunderbolts |
+--------------+
2 rows in set (0.000 sec)
```

c. To display the names and price of the books in descending order of their price.

```
MariaDB [sm2k4]> select Book_Name,Price from books order by price desc;
+----------------+-------+
| Book_Name      | Price |
+----------------+-------+
| Thunderbolts   |   750 |
| The Tears      |   650 |
| Fast Cook      |   355 |
| My First C++   |   350 |
| C++ Brainworks |   350 |
+----------------+-------+
5 rows in set (0.000 sec)
```

d. To increase the price of all books of First Publ. Publishers by 50.

```
MariaDB [sm2k4]> update books set Price = Price+50 where Publishers = 'First Publ.';
Query OK, 2 rows affected (0.010 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

e. To display the Book_ID, Book_Name, Quantity_Issued for all books which have been issued?

```
MariaDB [sm2k4]> select books.Book_ID,books.Book_Name,issued.Quantity_Issued from books,issued where books.Book_ID = issued.Book_ID
+---------+-------------+----------------+
| Book_ID | Book_Name   | Quantity_Issued |
+---------+-------------+----------------+
| C0001   | Fast Cook   |              5 |
| F0001   | The Tears   |              2 |
| T0001   | My First C++ |              4 |
+---------+-------------+----------------+
3 rows in set (0.000 sec)
```

f. Tim insert a new row in the table issued having the following data: 'F0002', 4.

```
MariaDB [sm2k4]> insert into issued values('F0002',4);
Query OK, 1 row affected (0.009 sec)
```

Give the output of the following queries based on the above tables:

->SELECT COUNT(DISTINCT Publishers)FROM BOOKS;

```
MariaDB [sm2k4]> SELECT COUNT(DISTINCT Publishers)FROM books;
+-----------------------------+
| COUNT(DISTINCT Publishers) |
+-----------------------------+
|                           3 |
+-----------------------------+
1 row in set (0.000 sec)
```

->SELECT SUM(Price) FROM BOOKS WHERE Quantity > 5;

```
MariaDB [sm2k4]> SELECT SUM(Price) FROM books WHERE Quantity > 5;
+------------+
| SUM(Price) |
+------------+
|       2200 |
+------------+
1 row in set (0.001 sec)
```

->SELECT Book_Name, Author_Name FROM BOOKS WHERE Price < 500;

```
MariaDB [sm2k4]> SELECT Book_Name, Author_Name FROM books WHERE Price < 500;
+----------------+----------------+
| Book_Name      | Author_Name    |
+----------------+----------------+
| Fast Cook      | Lata Kapoor    |
| My First C++   | Brain & Brooke |
| C++ Brainworks | A.W Rossaine   |
+----------------+----------------+
3 rows in set (0.000 sec)
```

->SELECT COUNT(*) FROM BOOKS;

```
MariaDB [sm2k4]> SELECT COUNT(*) FROM books;
+-----------+
| COUNT(*)  |
+-----------+
|         5 |
+-----------+
1 row in set (0.000 sec)
```