**Assignment-2(SQL QUERIES)**

**Part A: SQL DDL Statements (Design and Development)**

show databases;

create database employee;

use employee;

create table emp\_details(emp\_no int(10),emp\_name varchar(30),emp\_gender varchar(1),emp\_sal int(30));

show tables;

alter table emp\_details add emp\_dept varchar(20);

desc emp\_details;

insert into emp\_details values(1,'Ram','M',300000,'designing');

insert into emp\_details values(2,'Soham','M',300000,'designing');

insert into emp\_details values(3,'Mohan','M',250000,'management');

insert into emp\_details values(4,'Om','M',400000,'coding');

select \* from emp\_details;

create table emp\_info as select emp\_no,emp\_name,emp\_gender from emp\_details;

select \* from emp\_info;

truncate table emp\_info;

select \* from emp\_info;

drop table emp\_info;

select \* from emp\_info;

create view emp\_view1 as select \* from emp\_details;

create view emp\_view2 as select \* from emp\_details where emp\_dept="designing";

select \* from emp\_view1;

select \* from emp\_view2;

update emp\_details set emp\_dept="coding" where emp\_name="Mohan";

select \* from emp\_details;

drop view emp\_view1;

drop view emp\_view2;

create index emp\_ind on emp\_details(emp\_no,emp\_name);

show index from emp\_details;

**IMPORTANT POINTS:**

The main difference between the TRUNCATE, DELETE, and DROP commands in SQL is what they remove and how they do it:

* **TRUNCATE**Removes all rows from a table, but keeps the table structure intact. TRUNCATE is a DDL command that's fast and efficient because it doesn't log each row deletion. It's irreversible and releases the allocated storage space.
* **DELETE**Removes specific rows from a table based on conditions. DELETE is a DML command that's slower than TRUNCATE and DROP because it removes records individually. It's typically used with the SQL WHERE clause.
* **DROP**Permanently deletes a table or database object, including associated data and objects. DROP is a DDL command that removes the entire table along with its structure. It's irreversible and cannot be recovered without a backup.

In simple terms, an index in SQL is a tool used to quickly identify rows with specific column values. If there were no indexes, the SQL server would have to start with the first row and then go through the entire table until it discovers the relevant rows.

**Part B: SQL DML Statements (10 Queries)**

show databases;

create database student;

use student;

create table stud\_tab(stud\_id int(4),stud\_name varchar(20),stud\_dept varchar(10),stud\_dob date,stud\_address varchar(10));

desc stud\_tab;

insert into stud\_tab values(1,'Ram','Comp','2002-11-05','Pune');

insert into stud\_tab values(2,'Soham','IT','2002-09-03','Nashik');

insert into stud\_tab values(3,'Ramesh','Comp','2002-03-19','Pune');

insert into stud\_tab values(4,'Mohan','AI&DS','2002-02-22','Nagpur');

select \* from stud\_tab;

alter table stud\_tab add shift varchar(10);

update stud\_tab set shift='first' where stud\_id=1;

update stud\_tab set shift='second' where stud\_id=2;

update stud\_tab set shift='first' where stud\_id=3;

update stud\_tab set shift='first' where stud\_id=4;

select \* from stud\_tab;

insert into stud\_tab values(5,'Omkar','ENTC','2002-06-26','Pune','second');

select \* from stud\_tab;

delete from stud\_tab where stud\_address='Nagpur';

select \* from stud\_tab;

update stud\_tab set stud\_id=4 where stud\_name='Omkar';

select \* from stud\_tab;

select \* from stud\_tab where stud\_dob between '2002-01-01' and '2002-07-01';

alter table stud\_tab add stud\_fees int(15);

update stud\_tab set stud\_fees=15000 where stud\_id=1;

update stud\_tab set stud\_fees=20000 where stud\_id=2;

update stud\_tab set stud\_fees=20000 where stud\_id=3;

update stud\_tab set stud\_fees=15000 where stud\_id=4;

select \* from stud\_tab;

select \* from stud\_tab where stud\_fees=(select max(stud\_fees) from stud\_tab);

select sum(stud\_fees) from stud\_tab;

create table stud\_info as select stud\_id,stud\_name from stud\_tab;

select stud\_id from stud\_tab union select stud\_id from stud\_info;

**IMPORTANT POINTS:**

In SQL, the UNION operator combines the results from multiple queries into a single result set

**Assignment-3(all types of Join, Sub-Query and View:)**

show databases;

create database customer;

show databases;

use customer;

create table cust\_tab(id int(4),name varchar(20),quantity int(4),price int(10),item varchar(20));

show tables;

create table cust\_info(id int(4),name varchar(20),address varchar(20),mobile varchar(20));

desc cust\_tab;

desc cust\_info;

alter table cust\_tab add primary key(id);

alter table cust\_info add foreign key(id) references cust\_tab(id);

insert into cust\_tab values(1,'Ram',1,15,'Milk');

insert into cust\_tab values(2,'Soham',2,20,'Toast');

insert into cust\_tab values(3,'Mohan',4,5,'Parle-G');

insert into cust\_tab values(4,'Om',2,20,'Coca Cola');

select \* from cust\_tab;

alter table cust\_tab add totalprice int(4);

update cust\_tab set totalprice=quantity\*price where id=1;

update cust\_tab set totalprice=quantity\*price where id=2;

update cust\_tab set totalprice=quantity\*price where id=3;

update cust\_tab set totalprice=quantity\*price where id=4;

select \* from cust\_tab;

insert into cust\_info values(1,'Ram','Pune','9943569081');

insert into cust\_info values(2,'Soham','Pune','9978491281');

insert into cust\_info values(3,'Mohan','Nashik','8782356712');

insert into cust\_info values(4,'Om','Nagpur','7823450189');

select \* from cust\_info;

select cust\_tab.id,cust\_tab.name,cust\_tab.item,cust\_info.address from cust\_tab inner join cust\_info on cust\_tab.id=cust\_info.id;

select cust\_tab.id,cust\_tab.name,cust\_tab.item,cust\_info.address from cust\_tab left outer join cust\_info on cust\_tab.id=cust\_info.id;

select cust\_tab.id,cust\_tab.name,cust\_tab.item,cust\_info.address from cust\_tab right outer join cust\_info on cust\_tab.id=cust\_info.id;

create view mul\_view as select cust\_tab.id,cust\_tab.name,cust\_info.address from cust\_tab,cust\_info where cust\_info.id=cust\_tab.id;

select \* from mul\_view;

**Assignment-4(Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory.)**

create database library;

use library;

create table Borrower(Rollno int(4),Name varchar(20),DateofIssue date,NameofBook varchar(30),Status varchar(10));

insert into Borrower values(14,'Ram','2022-09-19','Operating System','I');

insert into Borrower values(27,'Soham','2022-07-24','Object Oriented Programming','I');

insert into Borrower values(34,'Mohan','2022-06-12','Microprocessor','I');

insert into Borrower values(48,'Om','2022-04-19','Mechanics','I');

select \* from Borrower;

create table Fine(Rollno int(4),Date date,Amount int(10));

delimiter //

create procedure calc\_Fine(in r int(10),in b varchar(30))

begin

declare doi date;

declare diff int(3);

select DateofIssue into doi from Borrower where Rollno=r and NameofBook=b;

select datediff(curdate(),doi) into diff;

if diff>=15 and diff<=30 then

insert into Fine values(r,curdate(),diff\*5);

end if;

if diff>30 then

insert into Fine values(r,curdate(),diff\*50);

end if;

end//

delimiter //

create procedure submit(in r int(2))

begin

update Borrower set Status='R' where Rollno=r;

delete from Fine where Rollno=r;

end//

call calc\_Fine(14,'Operating System');

select \* from Fine;

call calc\_Fine(27,'Object Oriented Programming');

call calc\_Fine(34,'Microprocessor');

call calc\_Fine(48,'Mechanics');

select \* from Fine;

call submit(14);

call submit(27);

call submit(48);

call submit(34);

select \* from Fine;

select \* from Borrower;

**Part 2: Circle and Area:**

DECLARE

Radius NUMBER(5); -- Variable to hold the radius value

Area NUMBER(7,2); -- Variable to store the calculated area with 2 decimal precision

Pi CONSTANT NUMBER(3,2) := 3.14; -- Defining a constant value for Pi

BEGIN

FOR Radius IN 3..7 LOOP

Area := Pi \* Radius \* Radius; -- Area calculation

INSERT INTO Circle (Radius, Area) VALUES (Radius, Area); -- Inserting into Circle table

END LOOP;

END;

/

**Assignment-5(Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.)**

create database Score;

use Score;

create table stud\_marks(name varchar(20),total\_marks int(5));

create table Result(roll\_no int(3) primary key,name varchar(20),class varchar(20));

insert into stud\_marks values('Suresh',995);

insert into stud\_marks values('Harish',865);

insert into stud\_marks values('Samart',920);

insert into stud\_marks values('Mohan',1000);

insert into stud\_marks values('Soham',745);

select \* from stud\_marks;

insert into Result(roll\_no,Name) values(1,'Suresh');

insert into Result(roll\_no,Name) values(2,'Harish');

insert into Result(roll\_no,Name) values(3,'Samart');

insert into Result(roll\_no,Name) values(4,'Mohan');

insert into Result(roll\_no,Name) values(5,'Soham');

select \* from Result;

delimiter //

create procedure proc\_Grade(in r int(2),out grade char(25))

begin

declare m int(4);

select total\_marks into m from stud\_marks where name=(select name from Result where roll\_no=r);

if m>=990 and m<=1500 then

select 'Distinction' into grade;

update Result set Class='Distinction' where Roll\_no=r;

elseif m>=900 and m<=989 then

select 'FirstClass' into grade;

update Result set Class='FirstClass' where Roll\_no=r;

elseif m>=825 and m<=899 then

select 'SecondClass' into grade;

update Result set Class='SecondClass' where Roll\_no=r;

else

select '--' into grade;

update Result set Class='--' where Roll\_no=r;

end if;

end //

delimiter //

create function func\_Grade(r int(2))

returns varchar(25)

deterministic

begin

declare grade varchar(25);

call proc\_Grade(r,grade);

return grade;

end //

select func\_Grade(1); //

select func\_Grade(2); //

select func\_Grade(3); //

select func\_Grade(4); //

select func\_Grade(5); //

select \* from Result; //

**IMPORTANT POINTS:**

In SQL, deterministic means a function always produces the same output when given the same input values and database state. Deterministic functions are useful because they can be indexed and are guaranteed to not cause side effects in the remote server.

**Assignment-6(Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)**

create database class;

use class;

create table O\_RollCall(roll\_no int(3),name varchar(20));

create table N\_RollCall(roll\_no int(3),name varchar(20));

insert into O\_RollCall values (1,'Himanshu');

insert into O\_RollCall values (2,'Ram');

insert into O\_RollCall values (3,'Soham');

insert into O\_RollCall values (5,'Mohan');

insert into O\_RollCall values (6,'Om');

insert into O\_RollCall values (9,'Yash');

insert into O\_RollCall values (11,'Mayur');

select \* from O\_RollCall;

select \* from N\_RollCall;

delimiter //

create procedure cursor\_proc\_p1()

begin

declare fin integer default 0;

declare old\_roll int(3);

declare old\_name varchar(20);

declare new\_roll int(3);

declare old\_csr cursor for select roll\_no,name from O\_RollCall;

declare new\_csr cursor for select roll\_no from N\_RollCall;

declare continue handler for not found set fin=1;

open old\_csr;

open new\_csr;

ss:loop

fetch old\_csr into old\_roll,old\_name;

fetch new\_csr into new\_roll;

if fin=1 then

leave ss;

end if;

if old\_roll<>new\_roll then

insert into N\_RollCall values(old\_roll,old\_name);

end if;

end loop;

close old\_csr;

close new\_csr;

end //

create procedure cursor\_proc\_p2(in r1 int)

begin

declare r2 int;

declare exit\_loop boolean;

declare c1 cursor for select roll\_no from O\_RollCall

where roll\_no>r1;

declare continue handler for not found set

exit\_loop=true;

open c1;

e\_loop:loop

fetch c1 into r2;

if not exists(select \* from N\_RollCall where roll\_no=r2)

then

insert into N\_RollCall select \* from O\_RollCall where roll\_no=r2;

end if;

if exit\_loop

then

close c1;

leave e\_loop;

end if;

end loop e\_loop;

end;//

call cursor\_proc\_p2(5); //

select \* from O\_RollCall; //

select \* from N\_RollCall; //

call cursor\_proc\_p2(3); //

call cursor\_proc\_p1(); //

select \* from O\_RollCall; //

select \* from N\_RollCall; //

**IMPORTANT POINTS:**

In SQL, a handler is a statement that tells an SQL procedure what to do when an error or warning occurs, or when no more rows are returned from a query

**Assignment-7(Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers)**

create database lib;

use lib;

create table library(bno int(5),bname varchar(40),author varchar(20),allowed\_days int(5));

create table library\_audit(bno int(5),old\_all\_days int(5),new\_all\_days int(5));

insert into library values(1,'Database Systems','Connally T',10);

insert into library values(2,'System Programming','John Donovan',20);

insert into library values(3,'Computer Network & Internet','Douglas E. Comer',18);

insert into library values(4,'Agile Project Management','Ken Schwaber',24);

insert into library values(5,'Python for Data Analysis','Wes McKinney',12);

select \* from library;

delimiter //

create trigger tr1

before update on library

for each row

begin

insert into library\_audit values(new.bno,old.allowed\_days,new.allowed\_days);

end //

update library set allowed\_days=15 where bno=1; //

update library set allowed\_days=25 where bno=2; //

update library set allowed\_days=13 where bno=3; //

update library set allowed\_days=19 where bno=4; //

update library set allowed\_days=17 where bno=5; //

select \* from library; //

select \* from library\_audit; //

**Assignment-8(Database Connectivity:)**

**How to Run the Program**

1. **Install the required library:**

**bash**

**Copy code**

pip install mysql-connector-python

1. **Modify the username, password, and database name in the connect\_mysql() function.**
2. **Run the script:**

**bash**

**Copy code**

python your\_script\_name.py

1. **Use the menu to add, view, edit, or delete records.**

import mysql.connector

# MySQL Database connection configuration

def connect\_mysql():

return mysql.connector.connect(

host="localhost", # Replace with your MySQL server host

user="your\_username", # Replace with your MySQL username

password="your\_password", # Replace with your MySQL password

database="your\_database" # Replace with your MySQL database name

)

# Connect to MySQL

conn = connect\_mysql()

cursor = conn.cursor()

# Create 'Circle' table if it doesn't exist

def create\_table():

cursor.execute("""

CREATE TABLE IF NOT EXISTS Circle (

Radius INT,

Area DECIMAL(7, 2)

)

""")

conn.commit()

print("Table 'Circle' created or already exists.")

# Add a new record

def add\_record(radius, area):

cursor.execute("INSERT INTO Circle (Radius, Area) VALUES (%s, %s)", (radius, area))

conn.commit()

print(f"Added record: Radius = {radius}, Area = {area}")

# Display all records

def display\_records():

cursor.execute("SELECT \* FROM Circle")

rows = cursor.fetchall()

print("Current Records:")

for row in rows:

print(row)

# Edit a record (by Radius)

def edit\_record(radius, new\_area):

cursor.execute("UPDATE Circle SET Area = %s WHERE Radius = %s", (new\_area, radius))

conn.commit()

print(f"Updated Radius {radius} with new Area {new\_area}")

# Delete a record (by Radius)

def delete\_record(radius):

cursor.execute("DELETE FROM Circle WHERE Radius = %s", (radius,))

conn.commit()

print(f"Deleted record with Radius {radius}")

# Close the connection

def close\_connection():

cursor.close()

conn.close()

print("Database connection closed.")

# Main Menu

def main():

create\_table() # Ensure the table exists

while True:

print("\nDatabase Navigation Operations:")

print("1. Add Record")

print("2. Display Records")

print("3. Edit Record")

print("4. Delete Record")

print("5. Exit")

choice = input("Enter your choice: ")

if choice == '1':

radius = int(input("Enter Radius: "))

area = float(input("Enter Area: "))

add\_record(radius, area)

elif choice == '2':

display\_records()

elif choice == '3':

radius = int(input("Enter Radius to edit: "))

new\_area = float(input("Enter new Area: "))

edit\_record(radius, new\_area)

elif choice == '4':

radius = int(input("Enter Radius to delete: "))

delete\_record(radius)

elif choice == '5':

break

else:

print("Invalid choice! Please try again.")

close\_connection()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**IMPORTANT POINTS:**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**is a conditional statement that ensures certain parts of the code only run when the script is executed directly, and not when it is imported as a module into another script.**

**Explanation in Detail**

1. **\_\_name\_\_ Variable:**
   * **Every Python script has a special built-in variable called \_\_name\_\_.**
   * **If the script is run directly, \_\_name\_\_ is set to "\_\_main\_\_".**
   * **If the script is imported as a module in another script, \_\_name\_\_ is set to the module’s name (the filename without .py).**
2. **Purpose of the if \_\_name\_\_ == "\_\_main\_\_": Block:**
   * **This block ensures that certain code will only execute when the script is run directly.**
   * **If the script is imported as a module in another script, the code inside this block will not execute automatically.**

**Assignment-9(MongoDB Queries: CRUD)**

show dbs

use book

show collections;

db.createCollection("library");

db.library.insert({"bid":1,"name":"C++"});

db.library.insert({"bid":2,"name":"SEPM","author":"Pressman"});

db.library.insert({"bid":3,"name":"CN","author":"Forouzan","cost":700});

db.library.find().pretty();

db.library.remove({"bid":1});

db.library.count();

db.library.find().pretty();

db.library.insert({"bid":1,"name":"C++"});

db.library.find().pretty();

db.library.find().sort({"bid":1})

db.library.insert({"bid":4,"name":"SPOS","author":"Pearson","cost":500});

db.library.find().pretty();

db.library.find().sort({"bid":1})

db.library.find({$and:[{"name":"CN"},{"cost":700}]}).pretty()

db.library.insert({"bid":5,"name":"TOC","author":"Addison-Wesley","cost":600});

db.library.insert({"bid":6,"name":"AI","author":"McGraw Hill Education","cost":800});

db.library.find().pretty();

db.library.find({$or:[{"cost":500},{"cost":800}]}).pretty()

db.library.find({"cost":{$ne:500}})

db.library.find({$nor:[{"cost":500},{"author":"Forouzan"}]})

db.library.find({"cost":{$not:{$gt:800}}})

db.library.insert({"bid":7,"name":"CC","author":"Wiley Publications","cost":400})

db.library.find()

db.library.update({'cost':400},{$set:{'cost':600}})

db.library.update({'cost':800},{$set:{'cost':1200}})

db.library.find().pretty();

**Assignment-10(MongoDB Aggregation and Indexing:)**

show dbs

use customer

db.cust\_table.insert({Item\_id:1,Cust\_Name:"Ram",Product:"Milk",Amount:40});

db.cust\_table.insert({Item\_id:2,Cust\_Name:"Ram",Product:"Parle\_G",Amount:50});

db.cust\_table.insert({Item\_id:3,Cust\_Name:"Mohan",Product:"Lays Chips",Amount:40});

db.cust\_table.insert({Item\_id:4,Cust\_Name:"Shivam",Product:"Mentos",Amount:10});

db.cust\_table.insert({Item\_id:5,Cust\_Name:"Mohan",Product:"Maggie",Amount:60});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$sum:"$Amount"}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$avg:"$Amount"}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$min:"$Amount"}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$max:"$Amount"}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$first:"$Amount"}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$last:"$Amount"}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$psuh:"$Amount"}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$sum:1}}});

db.cust\_table.aggregate({$group:{\_id:"$Cust\_Name","total":{$addToSet:"$Amount"}}});

db.cust\_table.createIndex({'Item\_id':1})

db.cust\_table.createIndex({'Item\_id':2})

db.cust\_table.createIndex({'Item\_id':4})

db.cust\_table.getIndexes()

db.cust\_table.dropIndex({'Item\_id':4})

db.cust\_table.dropIndex({'Item\_id':1})

db.cust\_table.getIndexes()

db.cust\_table.getIndexes()

**Assignment-11(Map reduces operations:)**

show dbs

use bill

db.pay.insert({Cust\_ID:"A123",Product:"Milk",Amount:40,Status:"P"});

db.pay.insert({Cust\_ID:"A123",Product:"Parle\_G",Amount:50,Status:"NP"});

db.pay.insert({Cust\_ID:"A123",Product:"Lays Chips",Amount:40,Status:"P"});

db.pay.insert({Cust\_ID:"B123",Product:"Mentos",Amount:10,Status:"P"});

db.pay.insert({Cust\_ID:"B123",Product:"Maggie",Amount:60,Status:"NP"});

db.pay.find()

db.pay.mapReduce(function(){emit(this.Cust\_ID,this.Amount);},function(key,values){return Array.sum(values)},{query:{"Status":"NP"},out:"Bill\_Amount"})

var mapFunc1=function(){emit(this.Cust\_ID,this.Amount);};

var reduceFunc1=function(keyCustID,valuePrices){return Array.sum(valuePrices);};

db.pay.mapReduce(mapFunc1,reduceFunc1,{out:"Map"})

db.Bill\_Amount.find()

db.Map.find()

**Assignment-12(Database Connectivity)**

package MongoDB;

import java.util.Scanner;

import com.mongodb.\*;

public class MongoDB {

public static void Stringdata(String data,DBCollection table) {

DBCursor cursor=table.find();

if(cursor.hasNext()) {

while(cursor.hasNext()) {

String Strdata=(String)cursor.next().get(data);

if(data=="Name") {

System.out.print(data+" :"+Strdata+"\t\t\t");

}

else {

System.out.print(data+":"+Strdata+"\t");

}

}

System.out.println(" ");

}

else {

System.out.print("");

}

}

public static void Intdata(String data,DBCollection table) {

DBCursor cursor=table.find();

if(cursor.hasNext()) {

while(cursor.hasNext()) {

int dataint=(Integer)cursor.next().get(data);

System.out.print(data+" :"+dataint+"\t\t\t\t");

}

System.out.println(" ");

}

else {

System.out.print("");

}

}

public static void main(String[] args) throws Exception {

MongoClient mongo=new MongoClient("localhost",27017);

System.out.println("Connected to the database successfully");

Scanner sc=new Scanner(System.in);

int n,age=0;

String name=" ",mobileno=" ",ans=" ",ans1=" ";

do {

DB db=mongo.getDB("Info");

DBCollection table=db.createCollection("Personal",null);

System.out.println("Enter the no of record you want to insert:");

n=sc.nextInt();

for(int i=1;i<=n;i++) {

BasicDBObject info= new BasicDBObject(i);

System.out.println("Enter Data"+(i));

System.out.print("Enter name:");

name=sc.next();

info.put("Name",name);

System.out.print("Enter age:");

age=sc.nextInt();

info.put("Age",age);

System.out.print("Enter Mobile Number:");

mobileno=sc.next();

info.put("Mobile Number",mobileno);

table.insert(info);

}

System.out.println("Insert Operation");

Stringdata("Name", table);

Stringdata("Mobile Number", table);

Intdata("Age", table);

System.out.print("Enter the no of record you want to delete:");

n=sc.nextInt();

for(int i=1;i<=n;i++) {

BasicDBObject info= new BasicDBObject(i);

System.out.println("Enter Data"+(i));

System.out.print("Enter name:");

name=sc.next();

info.put("Name",name);

table.remove(info);

}

System.out.println("Delete Operation");

Stringdata("Name", table);

Stringdata("Mobile Number", table);

Intdata("Age", table);

System.out.println("Do yo want to drop database: ");

ans=sc.next();

if(ans.equals("y")||ans.equals("Y")){

db.dropDatabase();

System.out.println("Database Droped ");

}

System.out.println("Do you want to continue:");

ans1=sc.next();

}while(ans1.equals("y")||ans1.equals("Y"));

}

}