DAYANANDA SAGAR College OF ENGINEERING

(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)

Department of Computer Science & Engineering

2022-23

FOURTH SEMESTER

DATA BASE MANAGEMENT LABORATORY MANUAL

Sub Code: 21CS42

DSCE
DEPARTMENT OF COMPUTER SCIENCE

DAYANANDA SAGAR COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision and Mission of the Department

Vision

To provide a vibrant learning environment in computer science and engineering with focus on industry needs and research, for the students to be successful global professionals contributing to the society.

Mission

*** To adopt a contemporary teaching learning process with emphasis on hands on and Collaborative learning.**

*** To facilitate skill development through additional training and encourage student forums for enhanced learning.**

*** To collaborate with industry partners and professional societies and make the students industry ready.**

*** To encourage innovation through multidisciplinary research and development activities.**

*** To inculcate human values and ethics to groom the students to be responsible citizens.**

DSCE
DEPARTMENT OF COMPUTER SCIENCE

DAYANANDA SAGAR COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Code of Conduct in the Lab

Do's

Students shall

- Come prepared for the program to be developed in the laboratory.
- Report any broken plugs or exposed electrical wires to your faculty/laboratory technician immediately.
- Turn off the machine once you have finished using it.
- Maintain silence while working in the lab.
- Keep the Computer lab premises clean and tidy.
- Place backpacks under the table or computer counters.
- Treat fellow users of the laboratory, and all equipment within the laboratory, with the appropriate level of care and respect.

Don'ts

Students shall not

- Talk on cell phones in the lab.
- Eat or drink in the laboratory.
- Touch, connect or disconnect any plug or cable without the faculty/laboratory technician's permission.
- Install or download any software or modify or delete any system files on any lab computers.
- Read or modify other users' files.
- Meddle with other users' files.
- Leave their personal belongings unattended. We are not responsible for any theft.

Program 1

Setting up database and creating table

```
CONN SYSTEM/SYSTEM;

CREATE USER <username> IDENTIFIED BY <password>;

GRANT CONNECT, RESOURCE TO <username>;

COMMIT;

CREATE TABLE students(
                        id INT PRIMARY KEY,
                        Name VARCHAR(20),
                        Subject VARCHAR(20));

DESCRIBE students;

INSERT INTO students VALUES

(1,“Taylor”, “DBMS”), (2.“Bob”, “DBMS”), (3,“Michael”, “DBMS”),…;

SELECT * FROM students;

ALTER TABLE students ADD course INT;

DESC students;

ALTER TABLE students MODIFY/ALTER course VARCHAR(20);

SAVEPOINT s1;

ALTER TABLE students ADD salary INT;

SAVEPOINT s2;

ALTER TABLE students DROP COLUMN salary;

SELECT * FROM students;

ROLLBACK TO s2;

COMMIT;
```

CREATE TABLE student2 AS SELECT * FROM student1;

DROP TABLE student1;

TRUNCATE TABLE student2;

DESC student1;

DESC student2;

RENAME student2 TO student1;

DELETE FROM students WHERE name = ‘Ram’;

DELETE FROM students;

UPDATE students SET name = ‘Sita’ WHERE id = ‘3’;

SELECT * FROM students WHERE age > 18 AND age < 21;

SELECT * FROM students WHERE age = 18 AND subject = ‘DBMS’;

Operations on dual table

SELECT 50 + 60 FROM dual;

SELECT 100 – 60 FROM dual;

SELECT 100 / 50 FROM dual;

SELECT 10 * 20 FROM dual;

String Operations

SELECT name FROM students WHERE name LIKE ‘%A’;

SELECT name FROM students WHERE name LIKE ‘_a’;

Program 2

Constraints

1. NULL
2. NOT NULL
3. UNIQUE
4. PRIMARY KEY
5. FOREIGN KEY
6. CHECK
7. DEFAULT

STUDENTS

USN	NAME	SUBJECT	CONTACT
1DS21CS001	Taylor	Statistics	1234
1DS21CS002	Bob	DBMS	1253
1DS21CS003	Michael	OS	1243
1DS21CS004	Ed	Maths	1579
1DS21CS005	Selena	Statistics	1254
1DS21CS006	Robert	DBMS	1377
1DS21CS007	Ariana	DBMS	1677

FACULTY

FID	NAME	SUBJECT	USN
1001	Joe	Statistics	1DS21CS001
1001	Joe	Statistics	1DS21CS005
1002	Jack	DBMS	1DS21CS002
1002	Jack	DBMS	1DS21CS006
1002	Jack	DBMS	1DS21CS007

IF TABLES ARE BEING CREATED BEFORE OF THE FOLLOWING NAME, THEN CHANGE THE NAMES OF THE FOLLOWING TABLES AS PER NEED

```
CREATE TABLE students(usn VARCHAR(20) PRIMARY KEY,
                      Name VARCHAR(20),
                      Subject VARCHAR(20) NOT NULL,
```

```
                    Contact NUMBER UNIQUE);
CREATE TABLE faculty(fid NUMBER,
                    Name VARCHAR(20),
                    Subject VARCHAR(20),
                    Usn    VARCHAR(20),
                    FOREIGN KEY(usn) REFERENCES students(usn));
CREATE TABLE student1(usn VARCHAR(20),
                    Age NUMBER,
                    CHECK(age >= 18 AND age >= 50));
CREATE TABLE stud1(age NUMBER DEFAULT 18);
```

Aggregate Functions

For this students need to add a column 'salary' in the faculty relation

```
SELECT MAX(salary) AS maxs FROM faculty;
SELECT salary FROM faculty ORDER BY salary;
SELECT salary FROM faculty ORDER BY salary DESC;
```

Program 3

What are constraints?
Constraints are the rule enforced on data columns on table. These are used to limit the type of data
that can go into a table. This ensures the accuracy and reliability of the data in the database.

Some important constraints:
NULL, NOT NULL, UNIQUE, DEFAULT, PRIMARY KEY, FOREIGN KEY ,CHECK

NOT NULL Constraint:

By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such constraint on this column specifying that NULL is now not allowed for that column.
Note: A NULL is not the same as no data, rather, it represents unknown data.

UNIQUE Constraint:
The UNIQUE Constraint prevents two records from having identical values in a particular column. In the EMPLOYEE table, for example, you might want to prevent two or more people from having same phone number.

PRIMARY KEY Constraint:
A **primary key** is a single field or combination of fields that uniquely identify a record.
The fields that are part of the primary key cannot contain a NULL value and must be Unique.
Each table should have a primary key, and each table can have only ONE primary key.
Note: When multiple fields are used as a primary key, they are called a **composite key.**

FOREIGN KEY Constraint:

A foreign key is a key used to link two tables together. Foreign Key is a column or a combination of columns whose values match a Primary Key of another table.

The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

Note: The table with the foreign key is called the child table, while the table with primary key is called Primary key. The foreign key in the child table will generally reference a primary key in the parent table

CHECK Constraint:

The CHECK Constraint enables a condition to check the value being entered into a record. If the
condition evaluates to false, the record violates the constraint and isn't entered into the table
CREATE TABLE EMPLOYEE (EmpId int PRIMARY KEY, EmpName varchar(40) NOT NULL, EmpPosition varchar(10), Age int NOT NULL CHECK (Age>= 18),
Department varchar(10));

Sl.No Query

1. To create a new table stud with following information→ USN, SNAME, Branch, Sem

   Create table stud(USN number(10), sname varchar(10));

2. To increase the size of a column sname of size 10 to 15

   Syntax : Alter table table_name modify column_name data type(new size)

   Example: Alter table stud modify(sname varchar(15))

3. To add a new column in existing table

   Syntax: Alter table tablename add column_name data type

   Example: Alter table stud modify(branch varchar(10), sem number(1));

4. Rename an existing column in the table

   Sql> alter table tablename rename column oldcolumn to newcolumn;

5. Add a primary key to existing column. It is possible only if the table does not have any records.

   Sql> alter table tablename modify(column_name number(2) primary key);

6. To drop a column from existing table.

   alter table tablename drop column column_name ;

   Create table Dept and Employee with the following information

   Dept: Dnum, Dname, Dloc

   Employee: Empid, Ename, Job, mgr_Empid, Hiredate, sal, comm, dno

   Create table dept(dnum number(2) primary key, dname varchar(15), dloc varchar(10))

   Create table emp(empno number(4) primary key, ename varchar(15), job varchar(10), hiredate date, mgr number(4), sal number(5), comm number(4), dno, foreign key (dno) references dept(dnum));

   Following rows in the table.

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	Deptno
7369	SMITH	CLERK	7902	23-Apr-2022	800		20
7499	ALLEN	SALESMAN	7698	3-Jun-2021	1600	300	30
7566	JONES	MANAGER	7839	22-May-2012	2975		20
7521	WARD	SALESMAN	7698	12-Jun-2018	1250	500	30
7698	BLAKE	MANAGER	7839	12-Sep-2013	2850		30
7782	CLARK	MANAGER	7839	15-May -2021	2450		10
7788	SCOTT	ANALYST	7566	12-Sep-2013	3000		20
7839	KING	PRESIDENT		01-Jan-2010	5000		10
7844	TURNER	SALESMAN	7698	22-Dec-2019	1500	200	30
7876	ADAMS	CLERK	7788	2-Oct-2020	1000		20
7900	JAMES	CLERK	7698	13-Jan-2021	950		30
7934	MILLER	CLERK	7782	14-Nov-2020	1300		10
7902	FORD	ANALYST	7566	12-Feb-2017	3000		20
7654	MARTIN	SALESMAN	7698	14-Nov-2021	1250	1400	30

1. Display all the details of all Managers

select * from emp where job = ‘MANAGER’ ;

2. **List the emps who joined before 2000.**

select * from emp where hiredate < ’01-Jan-1999′ ;

3. **List the Empno, Ename, Sal, Daily Sal of all Employees in the ASC order of AnnSal.**

SELECT Empno, Ename, sal, Sal/30 DailySal FROM Emp ORDER BY Sal*12 ;

4. **Display the empno , ename, job, hiredate, exp of all Mgrs**

select empno, ename, sal, months_between(sysdate,hiredate)/12 Exp

from emp where job = ‘MANAGER’ ;

5. **List the empno, ename, sal, exp of all emps working for Mgr 7839.**

select empno, ename, sal, months_between(sysdate,hiredate)/12 Exp

from emp B where Mgr = 7839 ;

6. **Display the details of the emps whose Comm. Is more than their sal.**

select * from emp where comm > sal ;

7. **List the emps in the asc order of Designations**

select * from emp order by job ;

8. **List the emps along with their exp and daily sal is more than Rs.100**

Select emp.*, months_between(sysdate,hiredate)/12 Exp from emp where sal/30 > 100 ;

9. **List the emps who are either 'CLERK' or 'ANALYST' in the desc order**

Select * from emp where job in ('CLERK','ANALYST') order by job desc ;

10. **List the emps who joined on 1-May-81,31-Dec-81, 17-Dec-81, 19-Jan-80 in asc order of seniority.**

Select ename, hiredate, months_between(sysdate, hiredate)/12 EXP from emp where hiredate like '01-MAY-81′ or hiredate like '31-DEC-81′or hiredate like '17-DEC-81′ or hiredate like '19-JAN-80' order by hiredate desc ;

11. **List the emps who are working for the deptno 10 or 20**

Select * from emp where deptno in (10,20) ;

12. **List the emps who are joined in the year 2013**

Select * from emp where hiredate like '%2013' ;

13. **List the emps who are joined in the month of Aug 2017**

Select * from emp where hiredate like %AUG-2017' ;

14. **List the emps whose annul sal ranging from 22000 and 45000**

Select * from emp where sal*12 between 22000 and 45000 ;

15. **List the enames those are starting with 's' and with fire characters**

Select * from emp where ename like 'S____' ;

16. **List the emps those are having four chars and third char must be 'r'**

Select * from emp where ename like '__R_' ;

17. **List the 5 character names starting with 's' and ending with 'h'**

Select * from emp where ename like ‘S___H’ ;

18. **List the emps who joined in January**

Select * from emp where hiredate like ‘%JAN%’ ;

19. **List the emps who joined in the month of which second character is ‘a’**

Select * from emp where hiredate like ‘__-_A%’ ;

20. **List the emps whose sal is 4 digit number ending with zero**

Select * from emp where sal like ‘___0’ ;

21. **List the emps whose names having a character set ‘ll’ together**

Select * from emp where ename like ‘%LL%’ ;

22. **List the emps who does not belong to deptno 20**

Select * from emp where deptno <> 20 ;

23. **List all the emps except ‘president’ & ‘Mgr’ in asc order of salaries**

Select * from emp where job not in (‘PRESIDENT’,’MANAGER’)

order by sal ;

24. **List the emps who joined in before or after 1981**

Select * from emp where hiredate not like ‘%81’ ;

25. **List the emps whose empno not starting with digit 78**

Select * from emp where empno not like ‘78%’ ;

26. **List the emps who are working under ‘Mgr’**

Select * from emp where mgr in (select empno from emp where job = ‘MANAGER’) ;

27. **List the emps who joined in any year but not belongs to the month of March**

Select * from emp where hiredate not like ‘%MAR%’ ;

28. **List the total information of emp table along with dname and loc of all the emps working under**

‘Accounting’ & ‘Research’ in the asc deptno

SELECT EMP.*,DNAME,LOC FROM Emp, Dept WHERE Dname IN

(‘ACCOUNTING’,’RESEARCH’)AND EMP.DEPTNO = DEPT.DEPTNO

ORDER BY EMP.DEPTNO

29. **List the details of the emps whose salaries more than the employee BLAKE**

select A.* from emp A, emp B where A.sal > B.sal and B.ename = 'BLAKE'

OR

select * from emp where sal > (select sal from emp where ename = 'BLAKE')

30. **List the details of the emps whose job is same as ALLEN.**

select * from emp where job = (select job from emp where ename = 'ALLEN')

31. **List the emps who are senior to King**

select * from emp where hiredate < (select hiredate from emp where ename = 'KING')

32. Display the name of the employee who is drawing maximum salary

Select ename from emp where sal in(select max(sal) from emp);

33. Display top 5 rows from a table

Select * from emp where rownum<=5;

34. List the names,job,sal , dname deptwise

Select e.ename,e.job, e.sal,d.dname from emp e, dept d where e.deptno =d.deptno order by e.deptno;

Program 4

SAILORS

Tables used in this note:

Sailors(sid: integer, sname: string, rating: integer, age: real);Boats(bid: integer, bname: string, color: string); Reserves(sid: integer, bid: integer, day: date).

Sailors

Sid	Sname	Rating	Age
22	Dustin	7	45
29	Allen	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Scott	7	35
71	martin	10	16
74	Scott	9	40
85	Art	3	25.5
95	Bob	3	63.5

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

sid	bid	day
22	101	10-oct-2023
22	102	10-oct-2023
22	103	08-oct-2023
22	104	07-oct-2023
31	102	11-nov-2023
31	103	06-nov-2023
31	104	12-nov-2023
64	101	05-sep-2023
64	102	08-sep-2023
74	103	08-sep-2023

Figure 1: Instances of Sailors, Boats and Reserves

1. Create the Tables:

```
Create table boat (bid number (3) primary key, bname varchar (10), color varchar (6));
CREATE TABLE sailors (      sid number (2) not null, sname varchar (10),rating integer, age real,
                            CONSTRAINT PK_sailors PRIMARY KEY(sid) );
CREATE TABLE reserves (sid number(2), bid number(3), day date,
                            CONSTRAINT PK_reserves PRIMARY KEY(sid, bid, day),
                            FOREIGN KEY(sid)  REFERENCES  sailors(sid),
                            FOREIGN KEY(bid) REFERENCES boats(bid) );
```

2. Insert Data

```
INSERT INTO sailors
        ( sid, sname, rating,
age )VALUES ( 22, 'Dustin',
7, 45.0 )

INSERT INTO reserves
        ( sid, bid, day ) VALUES
( 22, 101, '1998-10-10')
```

Note the date can have one of the following formats: yyyy-mm-dd, mm-dd-yyyy and mm/dd/yyyy

In addition, DB2 allows to parse the date attribute using its month(), year() and day() functions.

e.g. select * from reserves where year(day) = 1998 and month(day) = 10

3. Simple SQL Query

The basic form of an SQL query: SELECT [DISTINCT] select-list FROM from-list WHERE qualification

Ex1: Using DISTINCT

Sname	age
Dustin	45
Brutus	33
Lubber	55.5
Andy	25.5
Rusty	35
Horatio	**35**
Zorba	16
Horatio	**35**
Art	25.5
Bob	63.5

sname	age
Andy	25.5
Art	25.5
Bob	63.5
Brutus	33
Dustin	45
Horatio	35
Lubber	55.5
Rusty	35
Zorba	16

```
SELECT sname,
ageFROM
sailors
```

or

```
SELECT S.sname,
S.ageFROM sailors
S
```

```
SELECT DISTINCT S.sname, S.age
FROM sailors AS S
```

Ex2. Find all information of sailors who have reserved boat number 101.

SELECT S.* FROM Sailors S, Reserves R WHERE S.sid = R.sid AND R.bid = 103

Or without using the range variables, S and R

```
SELECT Sailors.* FROM Sailors, Reserves WHERE Sailors.sid = Reserves.sid AND Reserves.bid
= 103
```

* can be used if you want to retrieve all columns.

Ex3. Find the names of sailors who have reserved a red boat, and list in the order of age.

SELECT S.sname, S.age FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = ‘red’

ORDER BY S.age ORDER BY S.age [ASC] (default)ORDER BY S.age DESC

Ex4. Find the names of sailors who have reserved at least one boat.

SELECT sname FROM Sailors S, Reserves RWHERE S.sid = R.sid

The join of Sailors and Reserves ensure that for each select sname, the sailor has made somereservation.

Ex5. Find the ids and names of sailors who have reserved two different boats on the same day.

SELECT DISTINCT S.sid, S.sname FROM Sailors S, Reserves R1, Reserves R2 WHERE S.sid = R1.sid AND S.sid = R2.sid AND R1.day = R2.day AND R1.bid <> R2.bid

Ex6. Using Expressions and Strings in the SELECT Command.

SELECT sname, age, rating + 1
as st FROM Sailors
WHERE 2* rating – 1 < 10 AND sname like **'B_%b'**

SQL provides for pattern matching through LIKE operator, along with the use of symbols:

% (which stands for **zero or more** arbitrary characters) and

_ (which stands for **exactly one**, arbitrary, characters)

4. Union, Intersect and Except

Note that Union, Intersect and Except can be used on only two tables that are **union-compatible**,that is, have the same number of columns and the columns, taken in order, have the same types.

Ex7. Find the ids of sailors who have reserved a red boat or a green boat.

SELECT R.sid
FROM Boats B, Reserves R
WHERE R.bid = B.bid AND B.color = 'red'
UNION

SELECT R2.sid

FROM Boats B2, Reserves R2
WHERE R2.bid = B2.bid AND B2.color = ‘green’

The answer contains: SID----------22 31 64 74

The default for UNION queries is that **duplicates *are* eliminated**. To retain duplicates, use UNION ALL.

Replace UNION with **UNION ALL**. The answer contains: 22 31 74 22 31 64 22 31Replace UNION with **INTERSECT**. The answer contains: 22 31.

Replace UNION with **EXCEPT**. The answer contains just the id 64.

6. Nested Query

IN and **NOT IN**

EXISTS and NOT EXISTS

UNIQUE and NOT

UNIQUE

op ANY

op ALL

EX8: Find the names of sailors who have reserved boat 103.

```
SELECT
S.sname
FROM Sailors
S
WHERE S.sid IN ( SELECT R.sid
                 FROM Reserves R
                 WHERE R.bid = 103
                 )
```

The inner subquery has been completely independent of the outer query.

(Correlated Nested Queries)

```
SELECT
S.sname
```

```
FROM Sailors
S
WHERE EXISTS (      SELECT *
                    FROM Reserves
                    R WHERE R.bid
                    = 103
                        AND R.sid = S.sid )
```

The inner query depends on the row that is currently being examined in the outer query.

EX9: Find the name and the age of the youngest sailor.

```
SELECT S.sname,
S.ageFROM Sailors
S
WHERE S.age <= ALL ( SELECT age FROM Sailors )
```

EX10: Find the names and ratings of sailor whose rating is better than some sailor called Horatio.

```
SELECT S.sname,
S.ratingFROM Sailors
S
WHERE S.rating > ANY ( SELECT S2.rating
                       FROM Sailors S2
                          WHERE S2.sname = 'Horatio')
```

Note that **IN** and **NOT IN** are equivalent to **= ANY** and **<> ALL**, respectively.

EX11: Find the names of sailors who have reserved all boats.

```
SELECT
S.sname
FROM Sailors
S
WHERE NOT EXISTS ( ( SELECT B.bid
                     FROM Boats
```

```
B)EXCEPT
( SELECT R.bid
  FROM Reserves R
  WHERE R.sid = S.sid        ) )
```

An alternative solution:

```
SELECT
S.sname
FROM Sailors
S
WHERE NOT EXISTS ( SELECT B.bid
                   FROM Boats B
                   WHERE NOT EXISTS ( SELECT R.bid
                                      FROM Reserves R
                                      WHERE R.bid = B.bid AND
                                      R.sid = S.sid ) )
```

7. Aggregation Operators

COUNT ([DISTINCT] A): The number of (unique) values in the A column.SUM ([DISTINCT] A): The sum of all (unique) values in the A column.
AVG ([DISTINCT] A): The average of all (unique) values in the A column.MAX (A): The maximum value in the A column.
MIN (A): The minimum value in the A column.

EX12: Count the number of different sailor names.

```
SELECT COUNT( DISTINCT S.sname ) FROM Sailors S
```

EX13: Calculate the average age of all sailors.

```
SELECT AVG(s.age) FROM Sailors S
```

EX14: Find the name and the age of the youngest sailor.

```
SELECT S.sname, S.age FROM Sailors S
WHERE S.age = (SELECT MIN(S2.age) FROM Sailors S2 )
```

SELECT [DISTINCT] select-list
FROM from-list

WHERE qualification GROUP
BY grouping-list HAVING
group-qualification

EX15: Find the average age of sailors for each rating level.

```
SELECT S.rating, AVG(S.age) AS
avg_ageFROM Sailors S
GROUP BY S.rating
```

Rating	avg_age
1	33
3	44.5
7	40
8	40.5
9	35
10	25.5

EX16: Find the average age of sailors for each rating level that has at least two sailors.

```
SELECT S.rating, AVG(S.age) AS
avg_ageFROM Sailors S
GROUP BY S.rating
HAVING COUNT(*) > 1
```

Rating	avg_age
3	44.5
7	40
8	40.5
10	25.5

EX16: An example shows difference between WHERE and HAVING:

```
SELECT S.rating, AVG(S.age) as
avg_ageFROM Sailors S
WHERE S.age >=40
GROUP BY S.rating
```

Rating	avg_age
3	63.5
7	45
8	55.5

```
SELECT S.rating, AVG(S.age) as
avg_ageFROM Sailors S
GROUP BY S.rating
HAVING AVG(S.age)
>= 40
```

Rating	avg_age
3	44.5
7	40
8	40.5

5. NULL value and OUTER JOIN

In the presence of *null* values, any row that evaluates to **false** or to **unknown** is eliminated

The two rows are **duplicates** if corresponding columns are either equal, or both contain *null*.(If we compare two *null* values using =, the result is unknown)

The arithmetic operation +, -, * and / all return *null* if one of their arguments is *null*.

Count(*) handle *null* values just like other values. All the other aggregate operations (COUNT, SUM,AVG, MAX, MIN, and variations using DISTINCT) simply discard null values

After: INSERT INTO sailors (sid, sname, rating, age)VALUES (99, 'Dan', null, 48.0) ,

An example of OUTER JOIN:

```
SELECT sailors.sid, sailors.sname, reserves.bid
FROM sailors LEFT OUTER JOIN reserves ON reserves.sid =
sailors.sidORDER BY sailors.sid
```

sid	sname	bid
22	Dustin	101
22	Dustin	102
22	Dustin	103
22	Dustin	104
29	Brutus	
31	Lubber	102
31	Lubber	103
31	Lubber	104
32	Andy	
58	Rusty	
64	Horatio	101
64	Horatio	102
71	Zorba	
74	Horatio	103
85	Art	
95	Bob	
99	Dan	

DSCE
DEPARTMENT OF COMPUTER SCIENCE

Program 5

BANKING ENTERPRISE DATABASE

Consider the following database for a banking enterprise.

BRANCH (branch-name: String, branch-city: String, assets: real)

ACCOUNTS (accno: int, branch-name: String, balance: real)

DEPOSITOR (customer-name: String, accno: int)

CUSTOMER(customer-name:string,customer-street:string,customer-city:string)

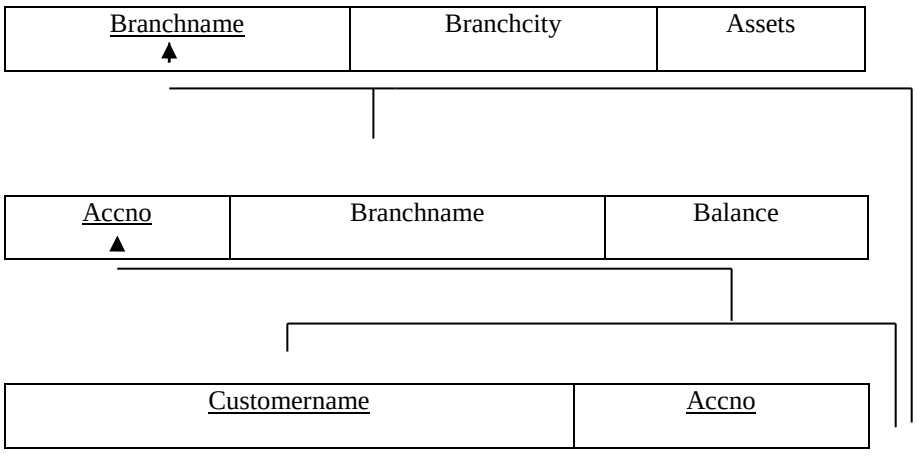
LOAN (loan-number: int, branch-name: String, amount: real)

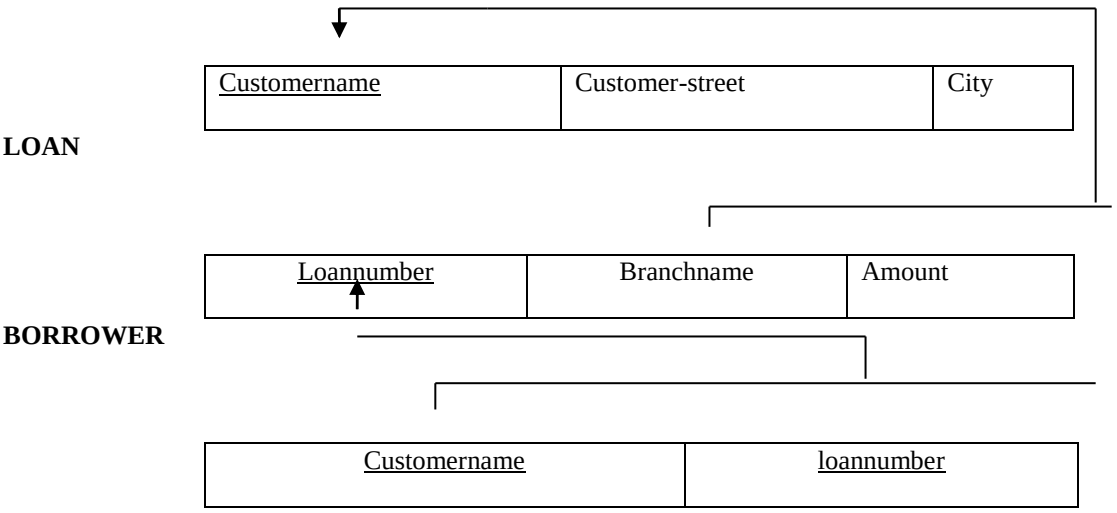
BORROWER (customer-name: String, loan-number: int)

a) **Create the tables for the schemas provided with primary keys and foreign keys.**
b) **Insert five tuples of values to each table.**
c) **Find all the customers who have at least two accounts at the *Main* branch.**
d) **Find all the customers who have an account at *all* the branches located in a specific city.**
e) **Demonstrate how you delete all account tuples at every branch located in a specific city.**

SCHEMA DIAGRAM

BRANCH





a) Create the above tables for the schemas provided with primary keys and foreign keys.

```
CREATE TABLE BRANCH
(BRANCH_NAME VARCHAR(20) PRIMARY KEY,
 BRANCH_CITY VARCHAR(10),
 ASSETS REAL);
```

Table created.

```
CREATE TABLE ACCOUNT
(ACCNO INT PRIMARY KEY,
 BRANCH_NAME VARCHAR(20),
 BALANCE REAL,
 FOREIGN KEY(BRANCH_NAME) REFERENCES BRANCH(BRANCH_NAME));
```

Table created.

```
CREATE TABLE CUSTOMER
(CUSTOMER_NAME VARCHAR(20) PRIMARY KEY,
 CUSTOMER_STREET VARCHAR(20),
```

```
 CUST_CITY VARCHAR(20));
```

Table created.

```
CREATE TABLE DEPOSITOR
(CUSTOMER_NAME VARCHAR(20),
 ACCNO INT,
 PRIMARY KEY(CUSTOMER_NAME,ACCNO),
 FOREIGN KEY(CUSTOMER_NAME) REFERENCES CUSTOMER(CUSTOMER_NAME),
 FOREIGN KEY(ACCNO) REFERENCES ACCOUNT(ACCNO)ON  DELETE CASCADE);
```

Table created.

```
CREATE TABLE LOAN
(LOAN_NO INT PRIMARY KEY,
  BRANCH_NAME VARCHAR(20),
  AMOUNT REAL,
  FOREIGN KEY(BRANCH_NAME) REFERENCES BRANCH(BRANCH_NAME));
```

Table created.

```
CREATE TABLE BORROWER
(CUSTOMER_NAME VARCHAR(20),
LOAN_NO INT,
FOREIGN KEY(CUSTOMER_NAME) REFERENCES CUSTOMER(CUSTOMER_NAME),
FOREIGN KEY(LOAN_NO) REFERENCES LOAN(LOAN_NO));
```

Table created.

b) Insert five tuples of values to each table.

c) Find all the customers who have at least two accounts at the *Main* branch.

```
SELECT CUSTOMER_NAME
FROM DEPOSITOR
WHERE ACCNO IN(SELECT ACCNO
FROM DEPOSITOR
WHERE ACCNO IN(  SELECT ACCNO
FROM ACCOUNT
WHERE BRANCH_NAME IN(SELECT BRANCH_NAME
FROM ACCOUNT
WHERE BRANCH_NAME='MAIN SBI VIJAY NAGAR'
GROUP BY BRANCH_NAME
HAVING COUNT(*) > 1)   ))
GROUP BY CUSTOMER_NAME HAVING COUNT(*) > 1;
```

d) Find all the customers who have an account at *all* the branches located in a specific city.

```
SELECT CUSTOMER_NAME
FROM BRANCH B,ACCOUNT A,DEPOSITOR D
WHERE B.BRANCH_NAME=A.BRANCH_NAME  AND  A.ACCNO=D.ACCNO AND
B.BRANCH_CITY='BANGALORE' GROUP BY CUSTOMER_NAME
HAVING COUNT(DISTINCT B.BRANCH_NAME)=(SELECT COUNT(BRANCH_NAME)
FROM BRANCH
WHERE BRANCH_CITY='BANGALORE');
```

e) Demonstrate how you delete all account tuples at every branch located in a specific city.

```
DELETE FROM ACCOUNT
WHERE BRANCH_NAME IN (SELECT BRANCH_NAME
 FROM BRANCH
 WHERE BRANCH_CITY='BANGALORE');
```