Laboratory Manual

on

DATABASE MANAGEMENT SYSTEMS LABORATORY

[10CSL 57]

By

Mr. R. RAJKUMAR
Assistant Professor
Department of Information Science & Engineering

For V Semester CSE / ISE

Session: Aug – Nov 2012

RNS Institute of Technology
Rajarajeshwarinagar Post, Channasandra,
Bengaluru – 560098

DATABASE APPLICATIONS LABORATORY

Subject Code: 10CSL57 **I.A. Marks : 25**
Hours/Week : 03 **Exam Hours: 03**
Total Hours : 42 **Exam Marks: 50**

1. Consider the following relations:
Student (*snum:* integer, *sname:* string, *major:* string, *level:* string, *age:* integer)
Class (*name:* string, *meets at:* string, *room:* string, *fid:* integer)
Enrolled (*snum:* integer, *cname:* string)
Faculty (*fid*: integer, *fname:* string, *deptid:* integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Level is a two character code with 4 different values (example: Junior: JR etc)
Write the following queries in SQL. No duplicates should be printed in any of the answers.
i. Find the names of all Juniors (level = JR) who are enrolled in a class taught by Prof. Harshith
ii. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.
iii. Find the names of all students who are enrolled in two classes that meet at the same time.
iv. Find the names of faculty members who teach in every room in which some class is taught.
v. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

2. The following relations keep track of airline flight information:
Flights (*no:* integer, *from:* string, *to:* string, *distance:* integer, *Departs:* time, *arrives:* time, *price:* real)
Aircraft (*aid:* integer, *aname:* string, *cruisingrange:* integer)
Certified (*eid:* integer, *aid:* integer)
Employees (*eid:* integer, *ename:* string, *salary:* integer)

Note that the Employees relation describes pilots and other kinds of employees as well; Every pilot is certified for some aircraft, and only pilots are certified to fly.
Write each of the following queries in SQL:
i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80, 000.
ii. For each pilot who is certified for more than three aircrafts, find the *eid* and the maximum *cruisingrange* of the aircraft for which she or he is certified.
iii. Find the names of pilots whose *salary* is less than the price of the cheapest route from Bengaluru to Frankfurt.
iv. For all aircraft with *cruisingrange* over 1000 Kms,. Find the name of the aircraft and the average salary of all pilots certified for this aircraft.
v. Find the names of pilots certified for some Boeing aircraft.
vi. Find the *aid*s of all aircraft that can be used on routes from Bengaluru to New Delhi.

3. Consider the following database of student enrollment in courses & books adopted for each course.
STUDENT (regno: string, name: string, major: string, bdate:date)
COURSE (course #:int, cname:string, dept:string)
ENROLL (regno:string, course#:int, sem:int, marks:int)
BOOK _ ADOPTION (course# :int, sem:int, book-ISBN:int)
TEXT (book-ISBN:int, book-title:string, publisher:string, author:string)
i. Create the above tables by properly specifying the primary keys and the foreign keys.
ii. Enter at least five tuples for each relation.
iii. Demonstrate how you add a new text book to the database and make this book be adopted by some department.

iv. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.
v. List any department that has *all* its adopted books published by a specific publisher.
vi. Generate suitable reports.
vii. Create suitable front end for querying and displaying the results.

4. The following tables are maintained by a book dealer.
AUTHOR (author-id:int, name:string, city:string, country:string)
PUBLISHER (publisher-id:int, name:string, city:string, country:string)
CATALOG (book-id:int, title:string, author-id:int, publisher-id:int, category-id:int, year:int, price:int)
CATEGORY (category-id:int, description:string)
ORDER-DETAILS (order-no:int, book-id:int, quantity:int)

i. Create the above tables by properly specifying the primary keys and the foreign keys.
ii. Enter at least five tuples for each relation.
iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books is greater than the average price of the books in the catalog and the year of publication is after 2000.
iv. Find the author of the book which has maximum sales.
v. Demonstrate how you increase the price of books published by a specific publisher by 10%.
vi. Generate suitable reports.
vii. Create suitable front end for querying and displaying the results.

5. Consider the following database for a banking enterprise
BRANCH(branch-name:string, branch-city:string, assets:real)
ACCOUNT(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)
i. Create the above tables by properly specifying the primary keys and the foreign keys
ii. Enter at least five tuples for each relation
iii. Find all the customers who have at least two accounts at the *Main* branch.
iv. Find all the customers who have an account at *all* the branches located in a specific city.
v. Demonstrate how you delete all account tuples at every branch located in a specific city.
vi. Generate suitable reports.
vii. Create suitable front end for querying and displaying the results.

Instructions:
1. The exercises are to be solved in an RDBMS environment like Oracle or DB2.
2. Suitable tuples have to be entered so that queries are executed correctly.
3. Front end may be created using either VB or VAJ or any other similar tool.
4. The student need not create the front end in the examination. The results of the queries may be displayed directly.
5. Relevant queries other than the ones listed along with the exercises may also be asked in the examination.
6. Questions must be asked based on lots.

1. Consider the following relations:
Student (*snum:* integer, *sname:* string, *major:* string, *level:* string, *age:* integer)
Class (*name:* string, *meets_at:* string, *room:* string, *fid:* integer)
Enrolled (*snum:* integer, *cname:* string)
Faculty (*fid*: integer, *fname:* string, *deptid:* integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Level is a two character code with 4 different values (Junior: JR, Senior:SR, Super Senior: SS, Fresher: FR)

```
CREATE TABLE STUDENT
(SNUM       INTEGER      NOT NULL,
 SNAME      VARCHAR (15),
 MAJOR      VARCHAR (10),
 SLEVEL     CHAR (2)     NOT NULL,
 AGE        NUMBER (3),
 PRIMARY KEY (SNUM));

CREATE TABLE FACULTY
(FID        INTEGER      NOT NULL,
 FNAME      VARCHAR (12) NOT NULL,
 DEPTID     NUMBER (3),
 PRIMARY KEY (FID));

CREATE TABLE CLASS
(NAME       VARCHAR (12)  NOT NULL,
 MEETS_AT   VARCHAR (10),
 ROOM       VARCHAR (5),
 FID        INTEGER,
 PRIMARY KEY (NAME),
 FOREIGN KEY (FID) REFERENCES FACULTY (FID));

CREATE TABLE ENROLLED
(SNUM       INTEGER      NOT NULL,
 CNAME      VARCHAR (12) NOT NULL,
 PRIMARY KEY (SNUM, CNAME),
 FOREIGN KEY (SNUM) REFERENCES STUDENT (SNUM),
 FOREIGN KEY (CNAME) REFERENCES CLASS (NAME));

 INSERT INTO STUDENT VALUES (&SNUM, '&SNAME', '&MAJOR', '&SLEVEL', &AGE);
 INSERT INTO FACULTY VALUES (&FID, '&FNAME', &DEPTID);
 INSERT INTO CLASS VALUES ('&CNAME', '&MEETS_AT', '&ROOM', &FID);
 INSERT INTO ENROLLED VALUES (&SNUM, '&CNAME');
```

Write the following queries in SQL. No duplicates should be printed in any of the answers.

i) Find the names of all Juniors (level = JR) who are enrolled in a class taught by Prof. Harshith

```
SELECT DISTINCT S.SNAME
FROM STUDENT S, CLASS C, ENROLLED E, FACULTY F
WHERE S.SNUM = E.SNUM AND E.CNAME = C.NAME AND C.FID = F.FID AND
        F.FNAME = 'Prof. Harshith' AND S.LEVEL = 'JR';
```

ii) Find the names of all classes that either meet in room R128 or have five or more Students enrolled.

```
SELECT C.NAME
FROM CLASS C
WHERE C.ROOM = 'R128'
        OR C.NAME IN (SELECT E.CNAME
                      FROM ENROLLED E
                      GROUP BY E.CNAME
                      HAVING COUNT (*) >= 5);
```

iii) Find the names of all students who are enrolled in two classes that meet at the same time.

```
SELECT DISTINCT S.SNAME
FROM   STUDENT S
WHERE S.SNUM IN   (SELECT E1.SNUM
                   FROM    ENROLLED E1, ENROLLED E2, CLASS C1, CLASS C2
                   WHERE E1.SNUM = E2.SNUM AND E1.CNAME <> E2.CNAME
                   AND E1.CNAME = C1.NAME
                   AND E2.CNAME = C2.NAME AND C1.MEETS_AT = C2.MEETS_AT);
```

iv) Find the names of faculty members who teach in every room in which some class is taught.

```
SELECT DISTINCT F.FNAME
FROM FACULTY F
WHERE NOT EXISTS (( SELECT *
                    FROM   CLASS C
                    EXCEPT
                   (SELECT C1.ROOM
                    FROM   CLASS C1
                    WHERE C1.FID = F.FID));
```

v) Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

```
SELECT      DISTINCT F.FNAME
FROM        FACULTY F
WHERE       5 > (SELECT   COUNT (E.SNUM)
                 FROM     CLASS C, ENROLLED E
                 WHERE    C.CNAME = E.CNAME
                 AND      C.FID = F.FID);
```

2. The following relations keep track of airline flight information:
Flights (*flno:* integer, *from:* string, *to:* string, *distance:* integer, *Departs:* time, *arrives:* time, *price:* real)
Aircraft (*aid:* integer, *aname:* string, *cruisingrange:* integer)
Certified (*eid:* integer, *aid:* integer)
Employees (*eid:* integer, *ename:* string, *salary:* integer)

Note that the Employees relation describes pilots and other kinds of employees as well; Every pilot is certified for some aircraft, and only pilots are certified to fly.

```
CREATE TABLE FLIGHTS
(FLNO       INTEGER          PRIMARY KEY,
FFROM       VARCHAR(15)      NOT NULL,
TTO         VARCHAR(15)      NOT NULL,
DISTANCE    INTEGER,
DEPARTS     TIMESTAMP,
ARRIVES     TIMESTAMP,
PRICE       NUMBER(10,2));

CREATE TABLE AIRCRAFT
(AID              INTEGER    PRIMARY KEY,
ANAME             VARCHAR(10),
CRUISINGRANGE     INTEGER);

CREATE TABLE EMPLOYEES
(EID        INTEGER     PRIMARY KEY,
ENAME       VARCHAR(15),
SALARY      NUMBER(10,2));

CREATE TABLE CERTIFIED
(EID    INTEGER NOT NULL,
AID     INTEGER NOT NULL,
PRIMARY KEY (EID, AID),
FOREIGN KEY (EID) REFERENCES EMPLOYEES (EID),
FOREIGN KEY (AID) REFERENCES AIRCRAFT (AID));

INSERT INTO FLIGHTS VALUES (&FLNO, '&FFROM', '&TTO', &DISTANCE, '&DEPARTS',
'&ARRIVES', &PRICE);
INSERT INTO AIRCRAFT VALUES (&AID, '&ANAME', &CRUISRANGE);
INSERT INTO EMPLOYEES VALUES (&EID, '&ENAME', &SALARY);
INSERT INTO CERTIFIED VALUES (&EID, &AID);
```

Write each of the following queries in SQL:

i) Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

```
SELECT      DISTINCT A.ANAME
FROM        AIRCRAFT A
WHERE       A.AID IN    (SELECT     C.AID
                         FROM       CERTIFIED C, EMPLOYEES E
                         WHERE      C.EID = E.EID AND
                         NOT EXISTS (SELECT     *
                                     FROM       EMPLOYEES E1
                                     WHERE      E1.EID = E.EID AND E1.SALARY < 80000));
```

ii) For each pilot who is certified for more than three aircrafts, find the *eid* and the maximum *cruisingrange* of the aircraft for which she or he is certified.

```
SELECT      C.EID, MAX (A.CRUISINGRANGE)
FROM        CERTIFIED C, AIRCRAFT A
WHERE       C.AID = A.AID
GROUP BY    C.EID
HAVING      COUNT (*) > 3;
```

iii) Find the names of pilots whose *salary* is less than the price of the cheapest route from Bengaluru to Frankfurt.

```
SELECT    DISTINCT  E.ANAME
FROM      EMPLOYEE E
WHERE     E.SALARY < ( SELECT MIN (F.PRICE)
                       FROM  FLIGHTS F
                       WHERE  F.FFROM = 'Bengaluru' AND F.TTO = 'Frankfurt');
```

iv) For all aircraft with *cruisingrange* over 1000 Kms,. Find the name of the aircraft and the average salary of all pilots certified for this aircraft.

Observe that aid is the key for Aircraft, but the question asks for aircraft names; we deal with this complication by using an intermediate relation Temp;

```
SELECT    TEMP.NAME, TEMP.AVGSALARY
FROM      (SELECT     A.AID, A.ANAME AS NAME,
                      AVG (E.SALARY) AS AVGSALARY
          FROM        AIRCRAFT A, CERTIFIED C, EMPLOYEES E
          WHERE       A.AID = C.AID AND
                      C.EID = E.EID AND A.CRUISINGRANGE > 1000
          GROUP BY    A.AID, A.ANAME) AS TEMP;
```

v) Find the names of pilots certified for some Boeing aircraft.

```
SELECT    DISTINCT E.ENAME
FROM      EMPLOYEES E, CERTIFIED C, AIRCRAFT A
WHERE     E.EID =  C.EID AND
          C.AID = A.AID AND
          A.ANAME = 'Boeing';
```

vi) Find the *aid*s of all aircraft that can be used on routes from Bengaluru to New Delhi.

```
SELECT A.AID
FROM   AIRCRAFT A
WHERE A.CRUISINGRANGE   >   (SELECT MIN (F.DISTANCE)
                             FROM   FLIGHTS F
                             WHERE F.FFROM = 'Bengaluru' AND F.TTO = 'New Delhi');
```

3. Consider the following database of student enrollment in courses & books adopted for each course.
STUDENT (regno: string, name: string, major: string, bdate:date)
COURSE (course:int, cname:string, dept:string)
ENROLL (regno:string, course:int, sem:int, marks:int)
BOOK _ ADOPTION (course:int, sem:int, book-ISBN:int)
TEXT (book-ISBN:int, book-title:string, publisher:string, author:string)

i. Create the above tables by properly specifying the primary keys and the foreign keys.

```
CREATE TABLE SSTUDENT
(RREGNO     VARCHAR(30)       PRIMARY KEY,
 NAME       VARCHAR(30)       NOT NULL,
```

```
 MAJOR      VARCHAR(30)      NOT NULL,
 BDATE      DATE             NOT NULL);

CREATE TABLE CCOURSE
(COURSE     INTEGER      PRIMARY KEY,
 CCNAME     VARCHAR(30) NOT NULL,
 DEPT       VARCHAR(30) NOT NULL);

CREATE TABLE EENROLL
(RREGNO     VARCHAR(30) NOT NULL,
 COURSE     INTEGER      NOT NULL,
 SEM        INTEGER      NOT NULL,
 MARKS      INTEGER      NOT NULL,
 PRIMARY KEY (RREGNO, COURSE, SEM),
 FOREIGN KEY (RREGNO) REFERENCES SSTUDENT(RREGNO),
 FOREIGN KEY (COURSE) REFERENCES CCOURSE(COURSE));

CREATE TABLE TTEXT
(BOOKISBN   INTEGER          PRIMARY KEY,
 BOOKTITLE  VARCHAR(30)      NOT NULL,
 PUBLISHER  VARCHAR(30)      NOT NULL,
 AUTHOR     VARCHAR(30)      NOT NULL);

CREATE TABLE BBOOKADOPTION
(COURSE     INTEGER      NOT NULL,
 SEM        INTEGER      NOT NULL,
 BOOKISBN   INTEGER      NOT NULL,
 PRIMARY KEY (COURSE, SEM, BOOKISBN),
 FOREIGN KEY (COURSE) REFERENCES CCOURSE (COURSE),
 FOREIGN KEY (BOOKISBN) REFERENCES TTEXT (BOOKISBN));
```

ii. Enter at least five tuples for each relation.

```
INSERT INTO SSTUDENT VALUES('1RN10IS012','ANN','DATBASE','15-JAN-84');
INSERT INTO SSTUDENT VALUES('1 RN10CS012','MARY','DMS','25-FEB-84');
INSERT INTO SSTUDENT VALUES('1 RN10TC012','TOM','SSDT','11-DEC-84');
INSERT INTO SSTUDENT VALUES('1 RN10EE012','EVE','POWER GENERATION','1-APR-84');
INSERT INTO SSTUDENT VALUES('1 RN10EC012','GEORGE','POWER ELECTRONICS','5-NOV-84');

INSERT INTO CCOURSE VALUES(1,'DATABASE','CS');
INSERT INTO CCOURSE VALUES(2,'DMS','CS');
INSERT INTO CCOURSE VALUES(3,'SSDT','TC');
INSERT INTO CCOURSE VALUES(4,'POWER GENERATION','EE');
INSERT INTO CCOURSE VALUES(5,'POWER ELECTRONICS','EC');
INSERT INTO CCOURSE VALUES(6,'DATASTRUCTURE','CS');

INSERT INTO TTEXT VALUES(1,'DATABASE A SYSTEMATIC APPROACH','JOHN WILEY','R ASHOK KUMAR');
INSERT INTO TTEXT VALUES(2,'DMS FOR DUMMIES','JOHN WILEY','MADHUPRIYA');
INSERT INTO TTEXT VALUES(3,'SSDT NO ONE CAN TEACH BETTER','PEARSON','GAURA');
INSERT INTO TTEXT VALUES(4,'POWER GENERATION BIBLE','TMH','MEENA');
INSERT INTO TTEXT VALUES(5,'POWER OF POWER ELECTRONICS','O REILLY','GG THE GREAT');
INSERT INTO TTEXT VALUES(6,'POWER OF DATASTRUCTURES','JOHN WILEY','DENNISRITCHIE');
```

```
INSERT INTO TTEXT VALUES(7,'ELEMENTARY DATASTUCTURES1','JOHN WILEY','HERBERT SHIELD');
INSERT INTO TTEXT VALUES(8,'ELEMENTARY DATASTUCTURES2','JOHN WILEY','HERBERT SHIELD');
INSERT INTO TTEXT VALUES(9,'DATABASE','JOHN WILEY','MAYOR');

INSERT INTO EENROLL VALUES ('1RN10IS012', 1, 5, 98);
INSERT INTO EENROLL VALUES ('1RN10CS012', 2, 3, 88);
INSERT INTO EENROLL VALUES ('1RN10TC012', 3, 5, 88);
INSERT INTO EENROLL VALUES ('1RN10EE012', 4, 5, 88);
INSERT INTO EENROLL VALUES ('1RN10EC012', 5, 5, 88);

INSERT INTO BBOOKADOPTION VALUES (1, 5, 1);
INSERT INTO BBOOKADOPTION VALUES (1, 4, 9);
INSERT INTO BBOOKADOPTION VALUES (2, 3, 2);
INSERT INTO BBOOKADOPTION VALUES (3, 5, 3);
INSERT INTO BBOOKADOPTION VALUES (4, 5, 4);
INSERT INTO BBOOKADOPTION VALUES (5, 5, 5);
INSERT INTO BBOOKADOPTION VALUES (6, 4, 6);
INSERT INTO BBOOKADOPTION VALUES (6, 4, 7);
INSERT INTO BBOOKADOPTION VALUES (6, 4, 8);
```

iii. Demonstrate how you add a new text book to the database and make this book be adopted by some department.

```
INSERT INTO TTEXT VALUES (10, 'DATABASE FUNDAS', 'PEARSON', 'SCHIELD');
INSERT INTO BBOOKADOPTION VALUES (1, 3, 10);
```

iv. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the 'CS' department that use more than two books.

```
SELECT      C.COURSE, T.BOOKISBN, T.BOOKTITLE
FROM        CCOURSE C, BBOOKADOPTION BA, TTEXT T
WHERE       C.COURSE=BA.COURSE AND
            BA.BOOKISBN=T.BOOKISBN
            AND C.DEPT='CS' AND
            EXISTS
                  (SELECT     COUNT (COURSE)
                   FROM       BBOOKADOPTION
                   WHERE      COURSE=C.COURSE
                   GROUP BY   COURSE
                   HAVING     COUNT (COURSE)>=2)
                   ORDER BY   T.BOOKTITLE;
```

v. List any department that has *all* its adopted books published by a specific publisher.

```
SELECT      C.DEPT, T.BOOKTITLE, T.PUBLISHER
FROM        CCOURSE C, TTEXT T, BBOOKADOPTION BA
WHERE       C.COURSE=BA.COURSE AND T.BOOKISBN=BA.BOOKISBN
            AND
            T.PUBLISHER ='JOHN WILEY'AND
            T.PUBLISHER= ALL (SELECT      T1.PUBLISHER
                              FROM        CCOURSE C1, BBOOKADOPTION BA1, TTEXT T1
                              WHERE       BA1.BOOKISBN=T1.BOOKISBN AND
```

```
                                    BA1.COURSE=C1.COURSE AND
                                    C.DEPT=C1.DEPT);
```

vi. Generate suitable reports.
vii. Create suitable front end for querying and displaying the results.

4. The following tables are maintained by a book dealer.
AUTHOR (author-id:int, name:string, city:string, country:string)
PUBLISHER (publisher-id:int, name:string, city:string, country:string)
CATALOG (book-id:int, title:string, author-id:int, publisher-id:int, category-id:int, year:int, price:int)
CATEGORY (category-id:int, description:string)
ORDER-DETAILS (order-no:int, book-id:int, quantity:int)

i. Create the above tables by properly specifying the primary keys and the foreign keys.

```
CREATE TABLE AAUTHOR
( AUTHORID          INTEGER       PRIMARY KEY,
 NAME               VARCHAR(30) NOT NULL,
 CITY               VARCHAR(30) NOT NULL,
 COUNTRY            VARCHAR(30) NOT NULL);

CREATE TABLE PPUBLISHER
(PUBLISHERID        INTEGER       PRIMARY KEY,
 NAME               VARCHAR(30) NOT NULL,
 CITY               VARCHAR(30) NOT NULL,
 COUNTRY            VARCHAR(30) NOT NULL);

CREATE TABLE BOOKCATEGORY
(CATEGORYID         INTEGER       PRIMARY KEY,
  DESCRIPTION       VARCHAR(30) NOT NULL);

CREATE TABLE CCATALOG
(BOOKID INTEGER PRIMARY KEY,
 TITLE              VARCHAR(30) NOT NULL,
 AUTHORID           INTEGER       NOT NULL,
 PUBLISHERID        INTEGER       NOT NULL,
 CATEGORYID         INTEGER       NOT NULL,
 YEAROFPUBLISH      INTEGER       NOT NULL,
 PRICE              INTEGER       NOT NULL,
 FOREIGN KEY (AUTHORID) REFERENCES AAUTHOR(AUTHORID),
 FOREIGN KEY (PUBLISHERID) REFERENCES PPUBLISHER(PUBLISHERID),
 FOREIGN KEY (CATEGORYID) REFERENCES BOOKCATEGORY(CATEGORYID));

CREATE TABLE OORDERDETAILS
(ORDERNO    INTEGER       PRIMARY KEY,
  BOOKID    INTEGER       NOT NULL,
  QUANTITY  INTEGER       NOT NULL,
  FOREIGN KEY (BOOKID) REFERENCES CCATALOG(BOOKID));
```

ii. Enter at least five tuples for each relation.

```
INSERT INTO AAUTHOR VALUES (1,'NAVATHE','ARLINGTON','USA');
INSERT INTO AAUTHOR VALUES (2,'RAGHU RAMAKRISHNAN','CALIFORNIA','USA');
```

```
INSERT INTO AAUTHOR VALUES (3,'DHAMDHERE','MUMBAI','INDIA');
INSERT INTO AAUTHOR VALUES (4,'BJARNE','NEW JERSY','USA');
INSERT INTO AAUTHOR VALUES (5,'TANENBAUM','AMSTERDAM','NETHERLAND');

INSERT INTO PPUBLISHER VALUES (1,'JOHN WILEY','NEW YORK','USA');
INSERT INTO PPUBLISHER VALUES (2,'PEARSON','BANGALORE','INDIA');
INSERT INTO PPUBLISHER VALUES (3,'O REILLY','NEW JERSY','USA');
INSERT INTO PPUBLISHER VALUES (4,'TMH','CALCUTTA','INDIA');
INSERT INTO PPUBLISHER VALUES (5,'JOHN WILEY','NEW DELHI','INDIA');

INSERT INTO BOOKCATEGORY VALUES (1,'DATABASE MANAGEMENT');
INSERT INTO BOOKCATEGORY VALUES (2,'OPERATING SYSTEMS');
INSERT INTO BOOKCATEGORY VALUES (3,'C++');
INSERT INTO BOOKCATEGORY VALUES (4,'COMPUTER NETWORKS');
INSERT INTO BOOKCATEGORY VALUES (5,'C');

INSERT INTO CCATALOG VALUES (1,'FUNDAMENTALS OF DBMS',1,2,1,2004,500);
INSERT INTO CCATALOG VALUES (2,'PRINCIPLES OF DBMS',2,1,1,2004,400);
INSERT INTO CCATALOG VALUES (3,'OPERATING SYSTEMS',3,4,2,2004,200);
INSERT INTO CCATALOG VALUES (4,'C++ BIBLE',4,5,3,2003,500);
INSERT INTO CCATALOG VALUES (5,'COMPUTER NETWORKS',5,3,4,2002,250);
INSERT INTO CCATALOG VALUES (6,'FUNDAMENTALS OF C',1,2,5,2004,700);
INSERT INTO CCATALOG VALUES (7,'OPERATING SYSTEMS 2',3,2,2,2001,600);

INSERT INTO OORDERDETAILS VALUES (1,1,1);
INSERT INTO OORDERDETAILS VALUES (2,2,1);
INSERT INTO OORDERDETAILS VALUES (3,3,1);
INSERT INTO OORDERDETAILS VALUES (4,4,1);
INSERT INTO OORDERDETAILS VALUES (5,5,1);
INSERT INTO OORDERDETAILS VALUES (6,6,7);
INSERT INTO OORDERDETAILS VALUES (7,7,9);
```

iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books is greater than the average price of the books in the catalog and the year of publication is after 2000.

```
SELECT      *
FROM        AAUTHOR A
WHERE       EXISTS
                (SELECT     A1.AUTHORID,COUNT(A1.AUTHORID)
                FROM        AAUTHOR A1,CCATALOG C
                WHERE       A1.AUTHORID=C.AUTHORID AND
                            A.AUTHORID=A1.AUTHORID AND
                            C.YEAROFPUBLISH > 2000 AND
                            C.PRICE  >  (SELECT     AVG(PRICE)
                                        FROM        CCATALOG)
                                        GROUP BY    A1.AUTHORID
                                        HAVING      COUNT(A1.AUTHORID) >=2);
```

iv. Find the author of the book which has maximum sales.

```
SELECT      DISTINCT A.NAME
FROM        AAUTHOR A, CCATALOG C, OORDERDETAILS ODM
```

```
WHERE   A.AUTHORID=C.AUTHORID AND ODM.BOOKID=C.BOOKID AND
        EXISTS
                (SELECT     OD.BOOKID,SUM(OD.QUANTITY)
                 FROM       OORDERDETAILS OD
                WHERE       OD.BOOKID=ODM.BOOKID
                GROUP BY    BOOKID
                HAVING      SUM(OD.QUANTITY)>= ALL
                                        (SELECT     SUM(QUANTITY)
                                        FROM        OORDERDETAILS
                                        GROUP BY    BOOKID));
```

v. Demonstrate how you increase the price of books published by a specific publisher by 10%.

```
UPDATE      CCATALOG
SET         PRICE = (1.1) * PRICE
WHERE       AUTHORID = (SELECT      AUTHORID
                        FROM        AAUTHOR
                        WHERE       NAME = 'NAVATHE');
```

vi. Generate suitable reports.
vii. Create suitable front end for querying and displaying the results

5. Consider the following database for a banking enterprise
BRANCH(branch-name:string, branch-city:string, assets:real)
ACCOUNT(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)

i. Create the above tables by properly specifying the primary keys and the foreign keys

```
CREATE TABLE BBRANCH
(BRANCHNAME     VARCHAR(30)     PRIMARY KEY,
 BRANCHCITY     VARCHAR(30)     NOT NULL,
 ASSETS         NUMBER(10,2)    NOT NULL);

CREATE TABLE BBANKACCOUNT
(ACCNO          NUMBER(5)       PRIMARY KEY,
 BRANCHNAME     VARCHAR(30)     NOT NULL,
 BALANCE        NUMBER(10,2),
 FOREIGN KEY (BRANCHNAME) REFERENCES BBRANCH (BRANCHNAME));

CREATE TABLE BBANKCUSTOMER
(CUSTOMERNAME   VARCHAR(30)     PRIMARY KEY,
 CUSTOMERSTREET VARCHAR(30)     NOT NULL,
 CUSTOMERCITY   VARCHAR(30)     NOT NULL);

CREATE TABLE DDEPOSITOR
(CUSTOMERNAME   VARCHAR(30)     NOT NULL,
 ACCNO          NUMBER(5)       NOT NULL,
 PRIMARY KEY (CUSTOMERNAME, ACCNO),
 FOREIGN KEY (CUSTOMERNAME) REFERENCES BBANKCUSTOMER (CUSTOMERNAME),
 FOREIGN KEY (ACCNO) REFERENCES BBANKACCOUNT (ACCNO) ON DELETE CASCADE);
```

```
CREATE TABLE LLOAN
(LOANNUMBER     INTEGER          PRIMARY KEY,
 BRANCHNAME     VARCHAR(30)      NOT NULL,
 AMOUNT         NUMBER(10,2)     NOT NULL,
 FOREIGN KEY (BRANCHNAME) REFERENCES BBRANCH (BRANCHNAME));

CREATE TABLE BBORROWER
(CUSTOMERNAME   VARCHAR(30)      NOT NULL,
 LOANNUMBER     INTEGER          NOT NULL,
 PRIMARY KEY (CUSTOMERNAME, LOANNUMBER),
 FOREIGN KEY (CUSTOMERNAME) REFERENCES BBANKCUSTOMER (CUSTOMERNAME),
 FOREIGN KEY (LOANNUMBER) REFERENCES LLOAN (LOANNUMBER));
```

ii. Enter at least five tuples for each relation

```
INSERT INTO BBRANCH VALUES('CHAMRAJPET','BANGALORE',50000);
INSERT INTO BBRANCH VALUES('RESIDENCY ROAD','BANGALORE',10000);
INSERT INTO BBRANCH VALUES('M G ROAD','BANGALORE',100000);
INSERT INTO BBRANCH VALUES('CP','DELHI',100000);
INSERT INTO BBRANCH VALUES('JANTARMANTAR','DELHI',100000);

INSERT INTO BBANKACCOUNT VALUES(1,'CHAMRAJPET',2000);
INSERT INTO BBANKACCOUNT VALUES(2,'RESIDENCY ROAD',5000);
INSERT INTO BBANKACCOUNT VALUES(3,'M G ROAD',6000);
INSERT INTO BBANKACCOUNT VALUES(4,'CP',9999);
INSERT INTO BBANKACCOUNT VALUES(5,'JANTARMANTAR',999);
INSERT INTO BBANKACCOUNT VALUES(6,'M G ROAD',999);
INSERT INTO BBANKACCOUNT VALUES(8,'RESIDENCY ROAD',999);
INSERT INTO BBANKACCOUNT VALUES(9,'CP',10000);
INSERT INTO BBANKACCOUNT VALUES(10,'RESIDENCY ROAD',5000);
INSERT INTO BBANKACCOUNT VALUES(11,'JANTARMANTAR',9999);

INSERT INTO BBANKCUSTOMER VALUES('ANNE','BULL TEMPLE ROAD','BANGALORE');
INSERT INTO BBANKCUSTOMER VALUES('DANNY','BANNERGATTA ROAD','BANGALORE');
INSERT INTO BBANKCUSTOMER VALUES('TOM','J C ROAD','BANGALORE');
INSERT INTO BBANKCUSTOMER VALUES('NICK','CP','DELHI');
INSERT INTO BBANKCUSTOMER VALUES('ROVER','JANTARMANTAR','DELHI');

INSERT INTO DDEPOSITOR VALUES('ANNE',1);
INSERT INTO DDEPOSITOR VALUES('DANNY',2);
INSERT INTO DDEPOSITOR VALUES('TOM',3);
INSERT INTO DDEPOSITOR VALUES('NICK',4);
INSERT INTO DDEPOSITOR VALUES('ROVER',5);
INSERT INTO DDEPOSITOR VALUES('ANNE',6);
INSERT INTO DDEPOSITOR VALUES('ANNE',8);
INSERT INTO DDEPOSITOR VALUES('NICK',9);
INSERT INTO DDEPOSITOR VALUES('DANNY',10);
INSERT INTO DDEPOSITOR VALUES('NICK',11);

INSERT INTO LLOAN VALUES(1,'CHAMRAJPET',1000);
INSERT INTO LLOAN VALUES(2,'RESIDENCY ROAD',2000);
INSERT INTO LLOAN VALUES(3,'M G ROAD',3000);
```

```
INSERT INTO LLOAN VALUES(4,'CP',4000);
INSERT INTO LLOAN VALUES(5,'JANTARMANTAR',5000);

INSERT INTO BBORROWER VALUES('ANNE',1);
INSERT INTO BBORROWER VALUES('ANNE',2);
INSERT INTO BBORROWER VALUES('TOM',3);
INSERT INTO BBORROWER VALUES('NICK',4);
INSERT INTO BBORROWER VALUES('ROVER',5);
```

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

```
SELECT      *
FROM        BBANKCUSTOMER C
WHERE EXISTS
            (SELECT     DP.CUSTOMERNAME, COUNT (DP.CUSTOMERNAME)
             FROM       DDEPOSITOR DP,BBANKACCOUNT BA
             WHERE      DP.ACCNO=BA.ACCNO AND
                        C.CUSTOMERNAME=DP.CUSTOMERNAME AND
                        BA.BRANCHNAME='RESIDENCY ROAD'
             GROUP BY   DP.CUSTOMERNAME
             HAVING     COUNT(DP.CUSTOMERNAME)>=2);
```

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

```
SELECT      *
FROM        BBANKCUSTOMER BC
WHERE       NOT EXISTS
                (SELECT     BRANCHNAME
                FROM        BBRANCH
                WHERE       BRANCHCITY='DELHI'
                 MINUS
                SELECT      BA.BRANCHNAME
                FROM        DDEPOSITOR D,BBANKACCOUNT BA
                WHERE       D.ACCNO=BA.ACCNO AND
                            BC.CUSTOMERNAME=D.CUSTOMERNAME );
```

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

```
DELETE FROM     BBANKACCOUNT
WHERE           BRANCHNAME IN
                        (SELECT BRANCHNAME
                         FROM BBRANCH
                         WHERE BRANCHCITY='BANGALORE');
```

vi. Generate suitable reports.
vii. Create suitable front end for querying and displaying the results.