

CHAPTER – 2 LIBRARY DATABASE

1) Consider the following schema for a Library Database:

BOOK (Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS (Book_id, Author_Name)

PUBLISHER (Name, Address, Phone)

BOOK_COPIES (Book_id, Branch_id, No-of_Copies)

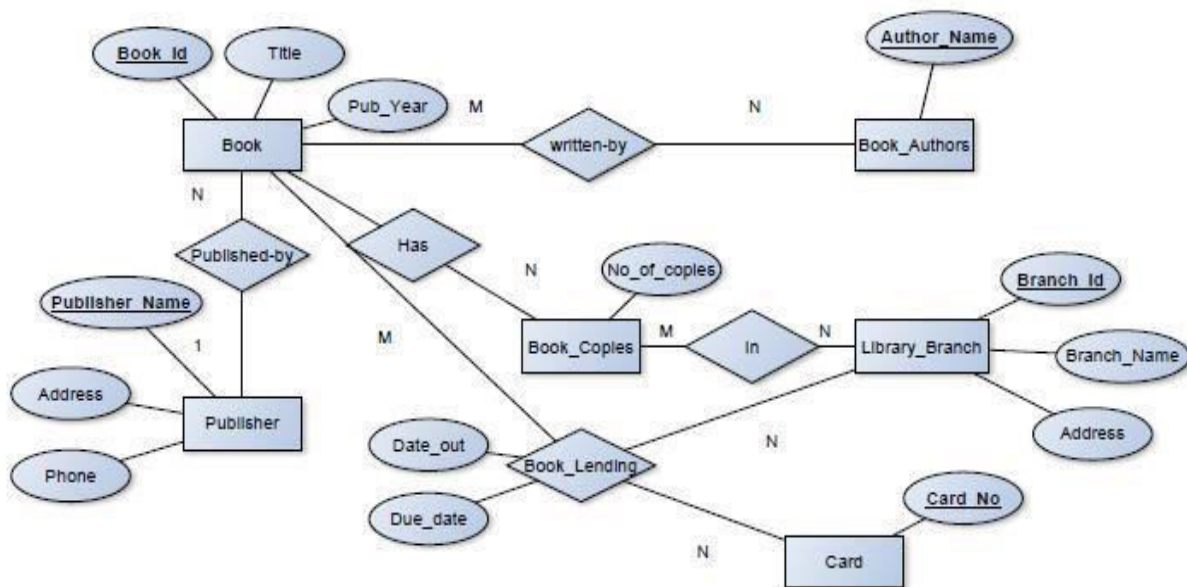
BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)

LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

Write SQL queries to

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library.

ER-Diagram:



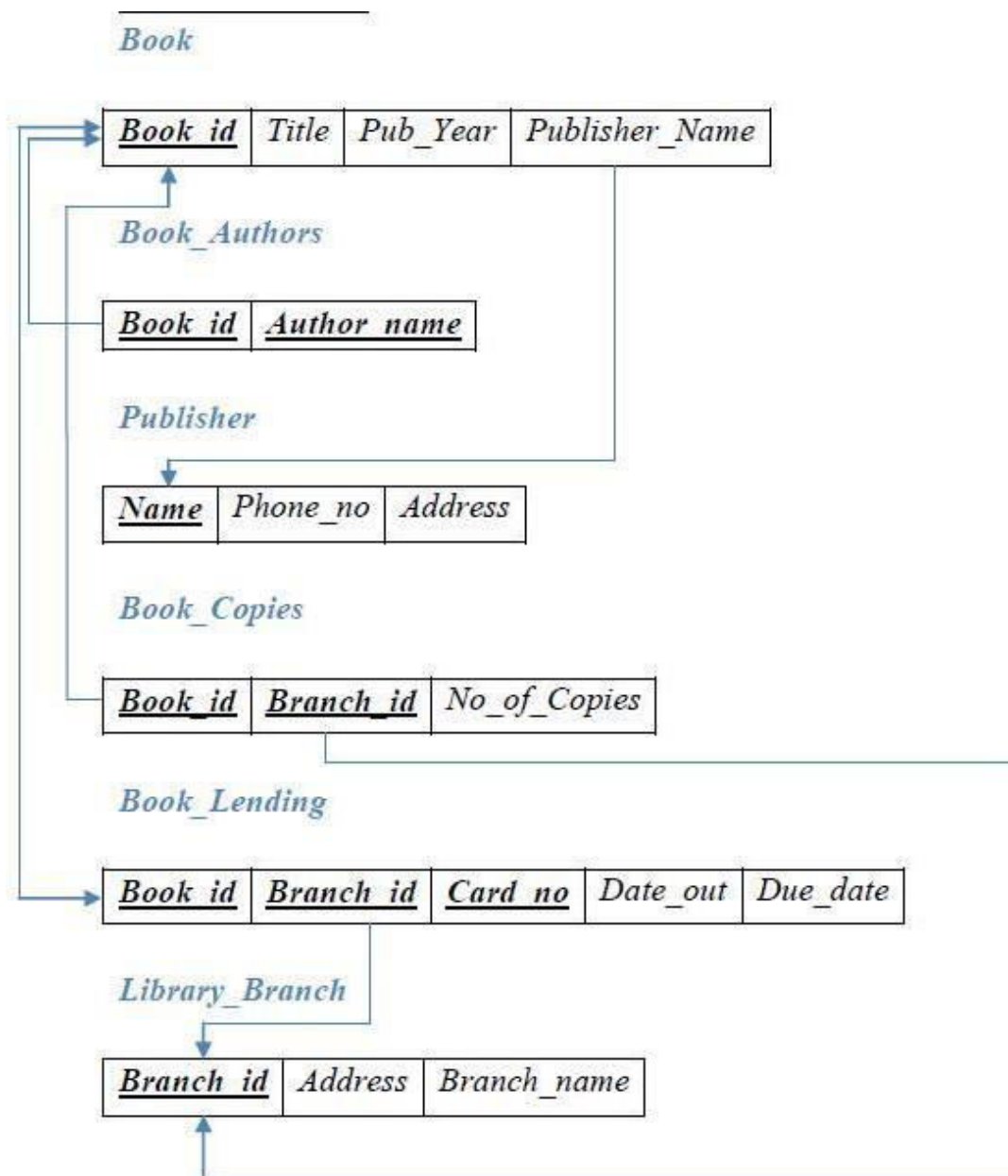
SCHEMA:

Table Creation:**PUBLISHER**

```
SQL> CREATE TABLE PUBLISHER(Pid int,  
    NAME VARCHAR(18) ADDRESS  
    VARCHAR(10),  
    PHONE VARCHAR(10), PRIMARY KEY(pid,name));
```

Table created.

BOOK

```
SQL> CREATE TABLE BOOK(  
    BOOK_ID INTEGER PRIMARY  
    KEY, TITLE VARCHAR(20),  
    PUBLISHER_NAME VARCHAR(20)REFERENCES PUBLISHER(NAME)ON  
    DELETE CASCADE,  
    PUB_YEAR NUMBER(4));
```

Table created.

BOOK_AUTHORS

```
SQL> CREATE TABLE BOOK_AUTHORS(  
    BOOK_ID INTEGER REFERENCES BOOK(BOOK_ID) ON DELETE  
    CASCADE, AUTHOR_NAME VARCHAR(20),  
    PRIMARY KEY(BOOK_ID));
```

Table created.

LIBRARY_BRANCH

```
SQL> CREATE TABLE  
    LIBRARY_BRANCH( BRANCH_ID  
    INTEGER PRIMARY KEY,  
    BRANCH_NAME VARCHAR(18),  
    ADDRESS VARCHAR(15));
```

Table created.

BOOK_COPIES

```
SQL> CREATE TABLE BOOK_COPIES(  
    BOOK_ID INTEGER REFERENCES BOOK(BOOK_ID) ON DELETE CASCADE,  
    BRANCH_ID INTEGER REFERENCES LIBRARY_BRANCH(BRANCH_ID) ON  
    DELETE CASCADE,
```

```
NO_OF_COPIES INTEGER,  
PRIMARY KEY(BOOK_ID,BRANCH_ID));
```

Table created.

BOOK_LENDING

```
SQL> CREATE TABLE BOOK_LENDING(  
    BOOK_ID INTEGER REFERENCES BOOK(BOOK_ID) ON DELETE CASCADE,  
    BRANCH_ID INTEGER REFERENCES LIBRARY_BRANCH(BRANCH_ID) ON  
    DELETE CASCADE,  
    CARD_NO  
    INTEGER,  
    DATE_OUT DATE,  
    DUE_DATE DATE,  
    PRIMARY KEY(BOOK_ID,BRANCH_ID,CARD_NO));
```

Table created.

Values for tables:

PUBLISHER

```
SQL> INSERT INTO PUBLISHER VALUES( '1001','PEARSON','BANGALORE','9875462530');
```

```
SQL> INSERT INTO PUBLISHER VALUES('1002','MCGRAW','NEWDELHI','7845691234');
```

```
SQL> INSERT INTO PUBLISHER VALUES('1003','SAPNA','BANGALORE','7845963210');
```

BOOK

```
SQL> INSERT INTO BOOK VALUES(1111,'SE','PEARSON',2005);
```

```
SQL> INSERT INTO BOOK VALUES(2222,'DBMS','MCGRAW',2004);
```

```
SQL> INSERT INTO BOOK VALUES(3333,'ANATOMY','PEARSON',2010);
```

```
SQL> INSERT INTO BOOK VALUES(4444,'ENCYCLOPEDIA','SAPNA',2010);
```

BOOK_AUTHORS

```
SQL> INSERT INTO BOOK_AUTHORS VALUES(1111,'SOMMERVILLE');
```

```
SQL> INSERT INTO BOOK_AUTHORS VALUES(2222,'NAVATHE');
```

```
SQL> INSERT INTO BOOK_AUTHORS VALUES(3333,'HENRY GRAY');
```

```
SQL> INSERT INTO BOOK_AUTHORS VALUES(4444,'THOMAS');
```

LIBRARY_BRANCH

```
SQL> INSERT INTO LIBRARY_BRANCH VALUES(11,'CENTRAL TECHNICAL','MG ROAD');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(22,'MEDICAL','BH ROAD');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(33,'CHILDREN','SS PURAM');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(44,'SECRETARIAT','SIRAGATE');
SQL> INSERT INTO LIBRARY_BRANCH VALUES(55,'GENERAL','JAYANAGAR');
```

BOOK_COPIES

```
SQL> INSERT INTO BOOK_COPIES VALUES(1111,11,5);
SQL> INSERT INTO BOOK_COPIES VALUES(3333,22,6);
SQL> INSERT INTO BOOK_COPIES VALUES(4444,33,10);
SQL> INSERT INTO BOOK_COPIES VALUES(2222,11,12);
SQL> INSERT INTO BOOK_COPIES VALUES(4444,55,3);
```

BOOK_LENDING

```
SQL> INSERT INTO BOOK_LENDING VALUES(2222,11,1,'10-JAN-2017','20-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(3333,22,2,'09-JUL-2017','12-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(4444,55,1,'11-APR-2017','09-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(2222,11,5,'09-AUG-2017','19-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(4444,33,1,'10-JUN-2017','15-AUG-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(1111,11,1,'12-MAY-2017','10-JUN-2017');
SQL> INSERT INTO BOOK_LENDING VALUES(3333,22,1,'10-JUL-2017','15-JUL-2017');
```

```
SQL> SELECT * FROM BOOK;
```

BOOK_ID	TITLE	PUBLISHER_NAME	PUB_YEAR
1111	SE	PEARSON	2005
2222	DBMS	MCGRAW	2004
3333	ANOTOMY	PEARSON	2010
4444	ENCYCLOPEDIA	SAPNA	2010

4 rows selected.

```
SQL> SELECT * FROM BOOK_AUTHORS;
```

BOOK_ID	AUTHOR_NAME
1111	SOMMERVILLE
2222	NAVATHE
3333	HENRY GRAY
4444	THOMAS

4 rows selected.

SQL> SELECT * FROM PUBLISHER;

Pid	NAME	ADDRESS	PHONE
1001	PEARSON	BANGALORE	9875462530
1002	MCGRAW	NEWDELHI	7845691234
1003	SAPNA	BANGALORE	7845963210

3 rows selected.

SQL> SELECT * FROM

BOOK_COPIES; BOOK_ID

BRANCH_ID NO_OF_COPIES

BOOK_ID	BRANCH_ID	NO_OF_COPIES
1111	11	5
3333	22	6
4444	33	10
2222	11	12
4444	55	3

5 rows selected.

SQL> SELECT * FROM BOOK_LENDING;

BOOK_ID	BRANCH_ID	CARD_NO	DATE_OUT	DUE_DATE
2222	11	1	10-JAN-17	20-AUG-17
3333	22	2	09-JUL-17	12-AUG-17
4444	55	1	11-APR-17	09-AUG-17
2222	11	5	09-AUG-17	19-AUG-17
4444	33	1	10-JUL-17	15-AUG-17
1111	11	1	12-MAY-17	10-JUN-17
3333	22	1	10-JUL-17	15-JUL-17

7 rows selected.

SQL> SELECT * FROM LIBRARY_BRANCH;

BRANCH ID	BRANCH NAME	ADDRESS
11	CENTRAL TECHNICAL	MG ROAD
22	MEDICAL	BH ROAD
33	CHILDREN	SS PURAM
44	SECRETARIAT	SIRAGATE
55	GENERAL	JAYANAGAR

5 rows selected.

Queries:

- 1) Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.

```

SELECT LB.BRANCH_NAME, B.BOOK_ID, TITLE,
       PUBLISHER_NAME, AUTHOR_NAME, NO_OF_COPIES
FROM BOOK B, BOOK_AUTHORS BA, BOOK_COPIES BC, LIBRARY_BRANCH
LB
WHERE B.BOOK_ID = BA.BOOK_ID
      AND BA.BOOK_ID =
      BC.BOOK_ID AND
      BC.BRANCH_ID =
      LB.BRANCH_ID
GROUP BY LB.BRANCH_NAME, B.BOOK_ID, TITLE,
       PUBLISHER_NAME, AUTHOR_NAME,
       NO_OF_COPIES;

```

BRANCH_NAME	BOOK_ID	TITLE	PUBLISHER_NAME	AUTHOR_NAME	NO_OF_COPIES
GENERAL	4444	ENCYCLOPEDIA	SAPNA	THOMAS	3
MEDICAL	3333	ANOTOMY	PEARSON	HENRY GRAY	6
CHILDREN	4444	ENCYCLOPEDIA	SAPNA	THOMAS	10
CENTRAL TECHNICAL	1111	SE	PEARSON	SOMMERVILLE	5
CENTRAL TECHNICAL	2222	DBMS	MCGRAW	NAVATHE	12

- 2) Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.

```

SELECT CARD_NO
FROM BOOK_LENDING
WHERE DATE_OUT BETWEEN '01-JAN-2017' AND '30-JUN-2017'
GROUP BY CARD_NO
HAVING COUNT(*) > 3;

```

```

CARD_NO
-----
1

```

3) Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

```
DELETE FROM BOOK  
WHERE BOOK_ID = '3333';
```

1 row deleted.

```
SQL> SELECT * FROM BOOK;
```

BOOK_ID	TITLE	PUBLISHER_NAME	PUB_YEAR
1111	SE	PEARSON	2005
2222	DBMS	MCGRAW	2004
4444	ENCYCLOPEDIA	SAPNA	2010

```
SQL> SELECT * FROM
```

```
BOOK_COPIES;
```

BOOK_ID	BRANCH_ID	NO_OF_COPIES
1111	11	5
4444	33	10
2222	11	12
4444	55	3

```
SQL> SELECT * FROM BOOK_LENDING;
```

BOOK_ID	BRANCH_ID	CARD_NO	DATE_OUT	DUE_DATE
2222	11	1	10-JAN-17	20-AUG-17
4444	55	1	11-APR-17	09-AUG-17
2222	11	5	09-AUG-17	19-AUG-17
4444	33	1	10-JUN-17	15-AUG-17
1111	11	1	12-MAY-17	10-JUN-17

4) Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

```
SELECT BOOK_ID, TITLE, PUBLISHER_NAME, PUB_YEAR  
FROM BOOK  
GROUP BY PUB_YEAR, BOOK_ID, TITLE, PUBLISHER_NAME;
```

BOOK_ID	TITLE	PUBLISHER_NAME	PUB_YEAR
2222	DBMS	MCGRAW	2004
1111	SE	PEARSON	2005
3333	ANOTOMY	PEARSON	2010
4444	ENCYCLOPEDIA	SAPNA	2010

- 5) Create a view of all books and its number of copies that are currently available in the Library.

```
CREATE VIEW BOOKS_AVAILABLE AS  
SELECT B.BOOK_ID, B.TITLE,  
C.NO_OF_COPIES FROM LIBRARY_BRANCH L,  
BOOK B, BOOK_COPIES C WHERE B.BOOK_ID  
= C.BOOK_ID AND  
L.BRANCH_ID=C.BRANCH_ID;
```

View created.

```
SQL> SELECT * FROM BOOKS_AVAILABLE;
```

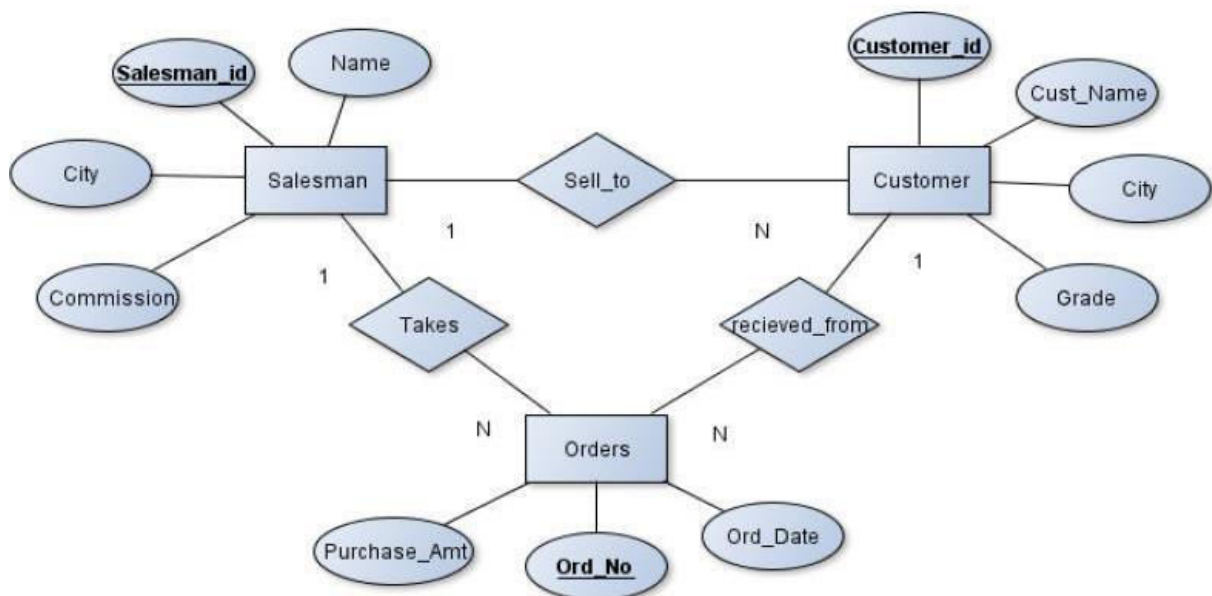
BOOK_ID	TITLE	NO_OF_COPIES
1111	SE	5
3333	ANATOMY	6
4444	ENCYCLOPEDIA	10
2222	DBMS	12
4444	ENCYCLOPEDIA	3

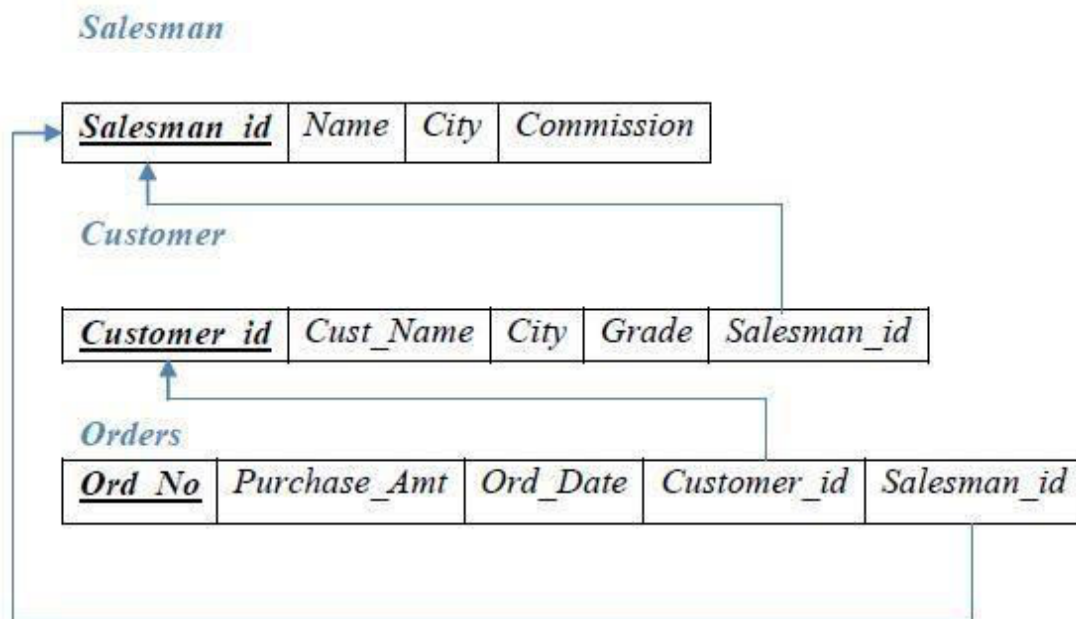
CHAPTER – 3

ORDER DATABASE

2) Consider the following schema for Order
Database: SALESMAN (Salesman_id, Name, City,
Commission)
CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)
ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)
Write SQL queries to

1. Count the customers with grades above Bangalore's average.
2. Find the name and numbers of all salesmen who had more than one customer.
3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)
4. Create a view that finds the salesman who has the customer with the highest order of a day.
5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

ER-Diagram:

SCHEMA:**Table Creation:****SALESMAN**

```

CREATE TABLE SALESMAN(
    SALESMAN_ID NUMBER(5) CONSTRAINT SALESMAN_SALID
    PRIMARY KEY, NAME VARCHAR(10) CONSTRAINT
    SALESMAN_NAME_NN NOT NULL,
    CITY VARCHAR(15) CONSTRAINT SALESMAN_CITY_NN
    NOT NULL, COMMISSION NUMBER(5));
  
```

Table created.

CUSTOMER

```

CREATE TABLE CUSTOMER(
    CUSTOMER_ID NUMBER(5) CONSTRAINT CUSTOMER_CUSTID_PK
    PRIMARY KEY, CUST_NAME VARCHAR(10) CONSTRAINT
    CUSTOMER_CUSTNAME_NN NOT NULL, CITY VARCHAR(10)
    CONSTRAINT CUSTOMER_CITY_NN NOT NULL,
    GRADE NUMBER(5) CONSTRAINT CUSTOMER_GRADE_NN NOT NULL,
    SALESMAN_ID NUMBER(5) CONSTRAINT CUSTOMER_SALEID_FK
    REFERENCES
    SALESMAN(SALESMAN_ID) ON DELETE SET NULL);
  
```

Table created.

ORDERS

```

CREATE TABLE ORDERS(
ORD_NO NUMBER(5) CONSTRAINT ORDERS_ODNO_PK
PRIMARY KEY, PURCHASE_AMT INTEGER CONSTRAINT
ORDERS_PAMT_NN NOT NULL, ORD_DATE DATE
CONSTRAINT ORDERS_ODATE_NN NOT NULL,
CUSTOMER_ID NUMBER(5) CONSTRAINT ORDERS_CUSTID_FK
REFERENCES
CUSTOMER(CUSTOMER_ID),
SALESMAN_ID NUMBER(5) CONSTRAINT ORDERS_SALEID_FK
REFERENCES
SALESMAN(SALESMAN_ID) ON DELETE CASCADE);

```

Table created.

Values for tables

```
SQL> INSERT INTO SALESMAN VALUES(&SALESMAN_ID,'&NAME','&CITY',&COMMISSION);
```

```
SQL> INSERT INTO CUSTOMER
VALUES(&CUSTOMER_ID,'&CUST_NAME','&CITY','&GRADE',&SALESMAN_ID);
```

```
SQL> INSERT INTO ORDERS
VALUES(&ORD_NO,&PURCHASE_AMT,'&ORD_DATE',&CUSTOMER_ID,&SALESMAN_ID);
```

```
SELECT * FROM SALESMAN;
```

SALESMAN_ID	NAME	CITY	COMMISSION
1000	RAJ	BENGALURU	50
2000	ASHWIN	TUMKUR	30
3000	BINDU	MUMBAI	40
4000	LAVANYA	BENGALURU	40
5000	ROHIT	MYSORE	60

```
SELECT * FROM CUSTOMER;
```

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
11	INFOSYS	BENGALURU	5	1000
22	TCS	BENGALURU	4	2000
33	WIPRO	MYSORE	7	1000
44	TCS	MYSORE	6	2000
55	ORACLE	TUMKUR	3	3000

```
SELECT * FROM ORDERS;
```

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
1	200000	12-APR-16	11	1000
2	300000	12-APR-16	11	2000
3	400000	15-APR-17	22	1000

- Count the customers with grades above Bangalore's average.

```

SELECT
COUNT(CUSTOMER_ID)
FROM CUSTOMER
WHERE GRADE>(SELECT AVG(GRADE)
              FROM CUSTOMER
              WHERE CITY LIKE '%BENGALURU');

COUNT(CUSTOMER_ID)
-----
3

```

- Find the name and numbers of all salesmen who had more than one customer.

```

SELECT NAME,
COUNT(CUSTOMER_ID) FROM
SALESMAN S, CUSTOMER C
WHERE S.SALESMAN_ID=C.SALESMAN_ID
GROUP BY NAME
HAVING COUNT(CUSTOMER_ID)>1;

```

NAME	COUNT(CUSTOMER_ID)
ASHWIN	2
RAJ	2

- List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)

```

(SELECT NAME
FROM SALESMAN S, CUSTOMER C
WHERE S.SALESMAN_ID=C.SALESMAN_ID AND
      S.CITY=C.CITY)

```

UNION

```

(SELECT NAME
FROM SALESMAN
WHERE SALESMAN_ID NOT IN(SELECT S1.SALESMAN_ID
                          FROM SALESMAN S1, CUSTOMER C1
                          WHERE S1.SALESMAN_ID=C1.SALESMAN_ID
                          AND
                          S1.CITY=C1.CITY));

```

NAME
ASHWIN
BINDU
LAVAN
YA RAJ
ROHIT

4. Create a view that finds the salesman who has the customer with the highest order of a day.

CREATE VIEWSALES_HIGHERORDER **AS SELECT**

SALESMAN_ID, PURCHASE_AMT

FROM ORDERS**WHERE** PURCHASE_AMT=(**SELECT MAX**(O.PURCHASE_AMT)**FROM** ORDERS O**WHERE** O.ORD_DATE='12-APR-16');

View created.

SELECT * FROM

SALES_HIGHERORDER;

SALESMAN_ID PURCHASE_AMT

2000	300000

5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

DELETE from salesman**WHERE** salesman_id = 1000;

1 row deleted.

SELECT * FROM SALESMAN;

SALESMAN_ID	NAME	CITY	COMMISSION
-----	-----	-----	
2000	ASHWIN	TUMKUR	30
3000	BINDU	MUMBAI	40
4000	LAVANYA	BENGALURU	40
5000	ROHIT	MYSORE	60

SELECT * FROM CUSTOMER;

CUSTOMER_ID	CUST_NAME	CITY	GRADE	SALESMAN_ID
-----	-----	-----	-----	
11	INFOSYS	BENGALURU	5	
22	TCS	BENGALURU	4	2000
33	WIPRO	MYSORE	7	
44	TCS	MYSORE	6	2000
55	ORACLE	TUMKUR	3	3000

SELECT * FROM ORDERS;

ORD_NO	PURCHASE_AMT	ORD_DATE	CUSTOMER_ID	SALESMAN_ID
-----	-----	-----	-----	
2	300000	12-APR-16	11	2000

CHAPTER – 4

MOVIE DATABASE

3) Consider the schema for Movie Database:

ACTOR (Act_id, Act_Name, Act_Gender)

DIRECTOR (Dir_id, Dir_Name,
Dir_Phone)

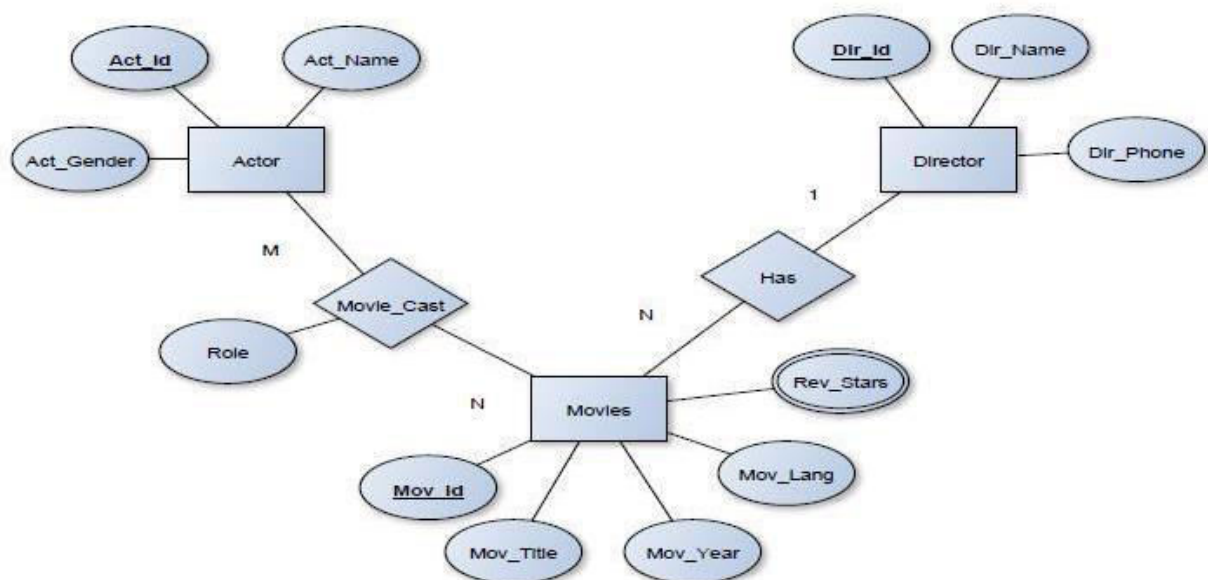
MOVIES (Mov_id, Mov_Title, Mov_Year, Mov_Lang,
Dir_id) MOVIE_CAST (Act_id, Mov_id, Role)

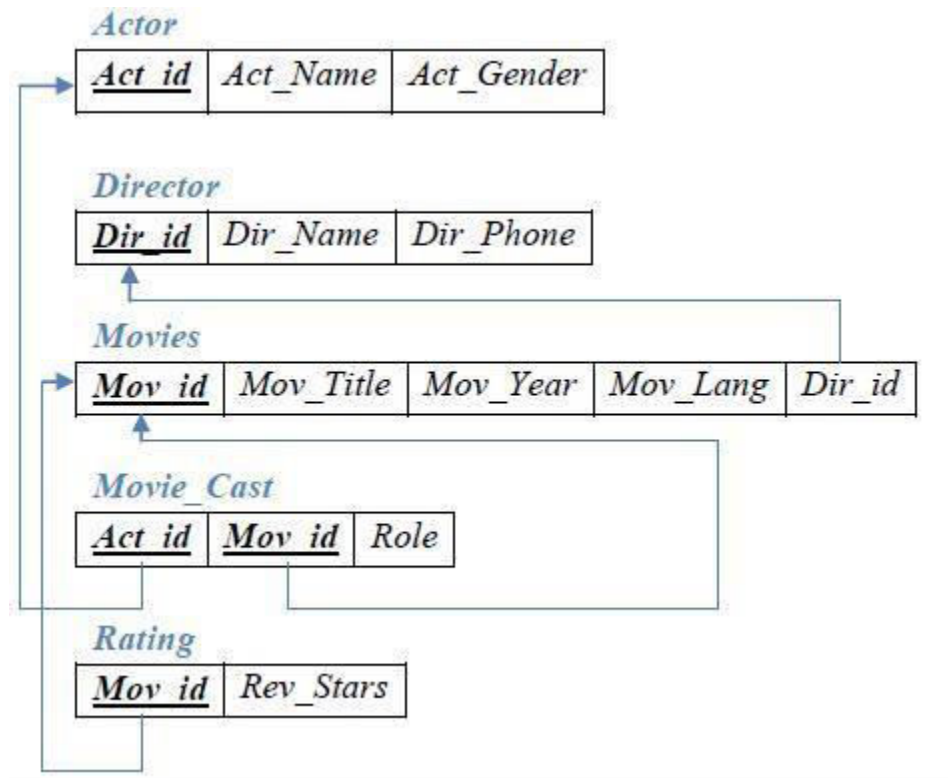
RATING (Mov_id,

Rev_Stars) Write SQL queries

to

1. List the titles of all movies directed by 'Hitchcock'.
2. Find the movie names where one or more actors acted in two or more movies.
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).
4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.
5. Update rating of all movies directed by 'Steven Spielberg' to 5.

ER-Diagram:

SCHEMA:**Table Creation:****ACTOR**

```

CREATE TABLE ACTOR(
  ACT_ID NUMBER(5) CONSTRAINT ACTOR_ACTID_PK PRIMARY
  KEY, ACT_NAME VARCHAR(18) CONSTRAINT
  ACTOR_ACTNAME_NN NOT NULL, ACT_GENDER VARCHAR(2)
  CONSTRAINT ACTOR_ACTGENDER_NN NOT NULL);
  
```

Table created.

DIRECTOR

```

CREATE TABLE DIRECTOR(
  DIR_ID NUMBER(5) CONSTRAINT DIRECTOR_DIRID_PK PRIMARY
  KEY, DIR_NAME VARCHAR(18) CONSTRAINT
  DIRECTOR_DIRNAME_NN NOT NULL, DIR_PHONE VARCHAR(10)
  CONSTRAINT DIRECTOR_DIRPHONE_NN NOT NULL);
  
```

Table created.

MOVIES

```
CREATE TABLE MOVIES(
MOV_ID NUMBER(5) CONSTRAINT MOVIES_MOVID_PK
PRIMARY KEY, MOV_TITLE VARCHAR(10) CONSTRAINT
MOVIES_MOVTITLE_NN NOT NULL, MOV_YEAR NUMBER(5)
CONSTRAINT MOVIES_MOVYEAR_NN NOT NULL,
MOV_LANG VARCHAR(10) CONSTRAINT
MOVIES_MOVLANG_NN NOT NULL,
DIR_ID NUMBER(5) CONSTRAINT MOVIES_DIRID_FK REFERENCES
DIRECTOR(DIR_ID));
```

Table created.

MOVIE_CAST

```
CREATE TABLE MOVIE_CAST(
ACT_ID NUMBER(5) CONSTRAINT MOVIECAST_ACTID_FK REFERENCES
ACTOR(ACT_ID), MOV_ID NUMBER(5) CONSTRAINT
MOVIECAST_MOVID_FK REFERENCES MOVIES(MOV_ID), ROLE
VARCHAR(10),
CONSTRAINT MOVIECAST_ACTID_MOVID_PK PRIMARY
KEY(ACT_ID,MOV_ID));
```

Table created.

RATING

```
CREATE TABLE RATING(
MOV_ID NUMBER(5) CONSTRAINT RATING_MOVID_FK REFERENCES
MOVIES(MOV_ID), REV_STARS NUMBER(1) CONSTRAINT
RATING_REVSTARS_NN NOT NULL, CONSTRAINT
RATING_MOVID_PK PRIMARY KEY(MOV_ID))
```

Table created.

Description of Schema:

SQL> DESC ACTOR

Name	Null?	Type
----- ACT_ID	NOT NULL	
NUMBER(5)		
ACT_NAME	NOT NULL	VARCHAR2(18)
ACT_GENDER	NOT NULL	VARCHAR2(2)

SQL> DESC DIRECTOR

Name	Null?	Type
-----	-----	----
DIR_ID	NOT NULL	
NUMBER(5)		
DIR_NAME	NOT NULL	VARCHAR2(18)
DIR_PHONE	NOT NULL	VARCHAR(10)

SQL> DESC MOVIES

Name	Null?	Type
-----	-----	----
MOV_ID	NOT NULL	
NUMBER(5)		
MOV_TITLE	NOT NULL	VARCHAR2(10)
MOV_YEAR	NOT NULL	NUMBER(5)
MOV_LANG	NOT NULL	
VARCHAR2(10) DIR_ID		
	NUMBER(5)	

SQL> DESC RATING

Name	Null?	Type
-----	-----	----
MOV_ID	NOT NULL	
NUMBER(5)		
REV_STARS	NOT NULL	NUMBER(1)

Values for tables:

SQL> INSERT INTO ACTOR VALUES(&ACT_ID,&ACT_NAME,&ACT_GENDER);

SQL> INSERT INTO DIRECTOR VALUES(&DIR_ID,&DIR_NAME,&DIR_PHONE);

SQL> INSERT INTO MOVIES
VALUES(&MOV_ID,&MOV_TITLE,&MOV_YEAR,&MOV_LANG,&DIR_ID);

SQL> INSERT INTO MOVIE_CAST VALUES(&ACT_ID,&MOV_ID,&ROLE);

SQL> INSERT INTO RATING VALUES(&MOV_ID,&REV_STARS);

SQL> SELECT * FROM ACTOR;

ACT_ID	ACT_NAME	AC
-----	-----	--
111	DEEPA SANNIDHI	F
222	SUDEEP	M
333	PUNEETH	M
444	DHIGANTH	M
555	ANGELA	F

SQL> SELECT * FROM DIRECTOR;

DIR_ID	DIR_NAME	DIR_PHONE
101	DWARKISH	112267809
102	RAJ MOULI	152358709
103	YOGARAJ	272337808
104	LANKESH	363445678
105	PAVAN KUMAR	385456809

SQL> SELECT * FROM MOVIES;

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
1111	LASTWORLD	2009	ENGLISH	104
2222	EEGA	2010	TELUGU	102
4444	PARAMATHMA	2012	KANNADA	103
3333	MALE	2006	KANNADA	103
5555	MANASARE	2010	KANNADA	103
6666	REAR WINDOW	1954	ENGLISH	101
7777	NOTORIOUS	1946	ENGLISH	101

SQL> SELECT * FROM MOVIE_CAST;

ACT_ID	MOV_ID	ROLE
222	2222	VILAN
333	4444	HERO
111	4444	HEROIN
444	3333	GUEST
444	5555	HERO
555	7777	MOTHER

SQL> SELECT * FROM RATING;

MOV_ID	REV_STARS
1111	3
2222	4
3333	3
5555	4
4444	5

1. List the titles of all movies directed by 'DWARKISH'.

```

SELECT MOV_TITLE
FROM MOVIES M, DIRECTOR D
WHERE D.DIR_ID=M.DIR_ID AND
      DIR_NAME=' DWARKISH ';

```

MOV_TITLE

 NOTORIO
 US REAR
 WINDOW

2. Find the movie names where one or more actors acted in two or more movies.

```
SELECT MOV_TITLE
FROM MOVIES M, MOVIE_CAST MC
WHERE M.MOV_ID=MC.MOV_ID AND
      MC.ACT_ID IN (SELECT ACT_ID
                    FROM MOVIE_CAST
                    GROUP BY ACT_ID
                    HAVING COUNT(MOV_ID)>=2);
```

MOV_TITLE

 MALE
 MANAS
 ARE

3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
(SELECT
  ACT_NAME
FROM ACTOR A
JOIN
  MOVIE_CAST C
ON
  A.ACT_ID=C.ACT_ID
JOIN MOVIES M
ON C.MOV_ID=M.MOV_ID
WHERE M.MOV_YEAR <
2000) INTERSECT
(SELECT
  ACT_NAME
FROM ACTOR A
JOIN
  MOVIE_CAST C
ON A.ACT_ID=C.ACT_ID JOIN
  MOVIES M
ON C.MOV_ID=M.MOV_ID
WHERE M.MOV_YEAR > 2015);
```

ACT_NAME

 DHIGANTH

4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

```

SELECT MOV_TITLE, REV_STARS
FROM MOVIES M, RATING R
WHERE M.MOV_ID=R.MOV_ID AND
REV_STARS>=1 ORDER BY MOV_TITLE

```

```

MOV_TITLE  REV_STARS
-----

```

```

EEGA                4
LASTWORLD           3
MALE                 3
MANASARE             4
PARAMATHMA          5

```

5. Update rating of all movies directed by 'Steven Spielberg' to 5.

```

UPDATE RATING
SET REV_STARS=5
WHERE MOV_ID IN (SELECT MOV_ID
                  FROM MOVIES M, DIRECTOR
                  D WHERE M.DIR_ID=D.DIR_ID
                  AND DIR_NAME='LANKESH');

```

1 row updated.

```

SELECT * FROM RATING

```

```

MOV_ID REV_STARS
-----

```

```

1111                5
2222                4
3333                3
5555                4
4444                5

```

CHAPTER - 5 AIRLINE**DATABASE**

1. 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 *aids* of all aircraft that can be used on routes from Bengaluru to New Delhi.

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

Create Table flights

(

no integer primary key,
fromplace varchar(20) not null,
toplace varchar(20) not null,
distance integer not null,
departs varchar(10)not null,

```

    arrives varchar(10) not null,
    price real
);

Create Table Aircraft
(
    aid int primary key,
    aname varchar(15),
    cruisingrange integer
);

Create Table employees
(
    eid int primary key,
    ename varchar(15),
    salary integer
);

Create Table certified
(
    eid int,
    aid int,
    primary key(eid,aid),
    foreign key (eid) references employees(eid),
    foreign key (aid) references aircraft(aid)
);

```

Insert the values into Relations

```

insert into flights values(255,'Bangalore','Mumbai', 200,'7:00 AM','9:00 AM',5000.00);
insert into flights values(256, 'Bangalore','Frankfurt', 400,'6:30 AM','11:30 AM',8000);
insert into flights values(257, 'Bangalore','Delhi',200,'9:30 AM','2:30 PM',5000);
insert into flights values(258,'Bangalore','Delhi',200,'10:40 AM','4:30 PM',6000);
insert into flights values(259,'Bangalore','Mangalore', 200,'11:00','7:30 PM',4000);
Select * from flights;

```

no	fromplace	toplace	distance	departs	arrives	price
255	Bangalore	Mumbai	200	7:00AM	9:00AM	5000
256	Bangalore	Frankfurt	400	6:30AM	11:30AM	8000
257	Bangalore	Delhi	200	9:30AM	2:30PM	5000
258	Bangalore	Delhi	200	10:40AM	4:30PM	6000
259	Bangalore	Mangalore	200	11:00AM	7:30PM	4000

```

insert into aircraft values(685,'Boeing15',1000);
insert into aircraft values(686,'Boeing10',2000);
insert into aircraft values(687,'Skytrain',1000);
insert into aircraft values(688,'AirBus',100);

```

```

Select * from aircraft;

```

aid	aname	cruisinrange
685	Boeing15	1000
686	Boeing10	2000
687	Skytrain	1000
688	AirBus	1500

insert into employees values(101,'Aruna', 90000);
 insert into employees values(102,'Asha',85000);
 insert into employees values(103,'Bhuvana',3000);
 insert into employees values(104,'Radha',4000);

Select * from employees;

eid	ename	salary
101	Aruna	90000
102	Asha	85000
103	Bhuvana	3000
104	Radha	4000

insert into certified values(101,685);
 insert into certified values(101,686);
 insert into certified values(101,687);
 insert into certified values(101,688);
 insert into certified values(102,685);
 insert into certified values(103,686);
 insert into certified values(103,687);

Select * from certified;

eid	aid
101	685
101	686
101	687
101	688
102	685
103	686
103	687

- 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 not in(select c.aid from certified c,employees e where c.eid=e.eid and e.salary<80000);

aname
Boeing15
AirBus

- 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(c.aid)>3;

eid	Max(a.cruisingrange)
101	2000

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

Select ename from employees where salary < (select min(price) from flights where fromplace='Bangalore' and toplace='Frankfurt');

ename
Bhuvana
Radha

- 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.

Select name,avgsal from(select a.aid,a.aname as name,avg(e.salary) as avgsal from aircraft a, certified c,employees e where a.aid=c.aid and c.eid=e.eid and cruisingrange>1000 group by a.aid,a.aname);

name	Avgsal
Boeing10	46500

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

Select distinct e.ename from employees e,certified c,aircraft a where a.aid=c.aid and c.eid =e.eid and a.aname like 'Boeing%';

ename
Asha
Bhuvana
Aruna

- vi) Find the *aids* 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(distance) from flights where fromplace='Bangalore' and toplace='Delhi');

aid
685
686
687

CHAPTER 6

BANKING ENTERPRISE DATABASE

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)

Write each of the following queries in SQL.

- Create the above tables by properly specifying the primary keys and the foreign keys
- Enter at least five tuples for each relation
- Find all the customers who have at least two accounts at the *Main* branch.
- Find all the customers who have an account at *all* the branches located in a specific city.
- Demonstrate how you delete all account tuples at every branch located in a specific city.
- Find the names of all depositors of a specific branch.
- Find the details of all loan holder of a specific branch.

Create:

```
CREATE TABLE branch
( branch_name VARCHAR(15),
  branch_city VARCHAR(15),
  assets NUMBER(10,2),
  PRIMARY KEY(branch_name)
);

CREATE TABLE account
( accno INTEGER(8),
  branch_name VARCHAR(15),
  balance NUMBER(10,2),
  PRIMARY KEY(accno),
  FOREIGN KEY(branch_name) REFERENCES branch(branch_name)ON DELETE
CASCADE );

CREATE TABLE customer
( customer_name VARCHAR(15),
  customer_street VARCHAR(15),
  customer_city VARCHAR(15),
  PRIMARY KEY(customer_name)
);

CREATE TABLE loan
( loan_number INTEGER(8),
  branc_hname VARCHAR(15),
  amount NUMBER(10,2),
  PRIMARY KEY(loan_number),
  FOREIGN KEY(branch_name) REFERENCES branch(branch_name) );
```

```
CREATE TABLE depositor
( customer_name VARCHAR(15),
  accno INTEGER,
  PRIMARY KEY(customer_name, accno),
  FOREIGN KEY(customer_name) REFERENCES customer(customer_name),
  FOREIGN KEY(accno) REFERENCES account(accno)
);
```

```
CREATE TABLE borrower
( customer_name VARCHAR(15),
  loan_number INTEGER(8),
  PRIMARY KEY(customer_name, loan_number),
  FOREIGN KEY(customer_name) REFERENCES customer(customer_name),
  FOREIGN KEY(loan_number) REFERENCES loan(loan_number)
);
```

INSERTIONS:

```
insert into branch (branch_name,branch_city,assets) values
  ("b1","c1",10000),
  ("b2","c2",20000),
  ("b3","c3",30000),
  ("b4","c4",40000),
  ("b5","c5",50000);
```

Query OK, 5 rows affected (0.06 sec)

Records: 5 Duplicates: 0 Warnings:0

mysql> SELECT * FROM branch;

```
+-----+-----+-----+
| branch_name | branch_city | assets |
+-----+-----+-----+
| b1          | c1          | 10000 |
| b2          | c2          | 20000 |
| b3          | c3          | 30000 |
| b4          | c4          | 40000 |
| b5          | c5          | 50000 |
+-----+-----+-----+
```

5 rows in set (0.00 sec)

```
INSERT INTO account (accno,branch_name,balance) VALUES
  (12,"b1",3000),
  (22,"b2",4000),
  (32,"b3",5000),
  (42,"b4",6000),
  (52,"b5",7000);
```

Query OK, 5 rows affected (0.06 sec)

Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM account;

```

+-----+-----+-----+
| accno | branch_name | balance |
+-----+-----+-----+
| 12 | b1 | 3000 |
| 22 | b2 | 4000 |
| 32 | b3 | 5000 |
| 42 | b4 | 6000 |
| 52 | b5 | 7000 |
+-----+-----+-----+

```

5 rows in set (0.00 sec)

```

INSERT INTO customer (customer_name,customer_street,customer_city) VALUES
    ("cust1","cstreet1","ccity1"),
    ("cust2","cstreet2","ccity2"),
    ("cust3","cstreet3","ccity3"),
    ("cust4","cstreet4","ccity4"),
    ("cust5","cstreet5","ccity5");

```

Query OK, 5 rows affected (0.07 sec)

Records: 5 Duplicates: 0 Warnings: 0

```

SELECT * FROM customer;

```

```

+-----+-----+-----+
| customer_name | customer_street | customer_city |
+-----+-----+-----+
| cust1 | cstreet1 | ccity1 |
| cust2 | cstreet2 | ccity2 |
| cust3 | cstreet3 | ccity3 |
| cust4 | cstreet4 | ccity4 |
| cust5 | cstreet5 | ccity5 |
+-----+-----+-----+

```

5 rows in set (0.00 sec)

```

INSERT INTO depositor (customer_name,accno) VALUES
    ("cust1",12),
    ("cust2",22),
    ("cust3",32),
    ("cust4",42),
    ("cust5",52);

```

Query OK, 5 rows affected (0.06 sec)

Records: 5 Duplicates: 0 Warnings: 0

```

SELECT * FROM depositor;

```

```

+-----+-----+
| customer_name | accno |
+-----+-----+
| cust1 | 12 |
| cust2 | 22 |
| cust3 | 32 |
| cust4 | 42 |
| cust5 | 52 |
+-----+-----+

```

5 rows in set (0.00 sec)

```
INSERT INTO loan (loan_number,branch_name,amount) VALUES
(10,"b1",10000),
(20,"b2",20000),
(30,"b3",30000),
(40,"b4",40000),
(50,"b5",50000);
```

Query OK, 5 rows affected (0.06 sec)
Records: 5 Duplicates: 0 Warnings: 0
select * from loan;

loan_number	branch_name	amount
10	b1	10000
20	b2	20000
30	b3	30000
40	b4	40000
50	b5	50000

5 rows in set (0.00 sec)

```
INSERT INTO borrower (customer_name,loan_number) VALUES
("cust1",10),
("cust2",20),
("cust3",30),
("cust4",40),
("cust5",50);
```

Query OK, 5 rows affected (0.05 sec)
Records: 5 Duplicates: 0 Warnings: 0
SELECT * FROM borrower;

customer_name	loan_number
cust1	10
cust2	20
cust3	30
cust4	40
cust5	50

5 rows in set (0.00 sec)

QUERIES:

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

```
SELECT customer_name
FROM depositor d,account a
WHERE d.accno=a.accno
AND a.branch_name='Main'
GROUP BY d.customer_name
HAVING COUNT(d.customer_name)>=2;
```

Empty set (0.00 sec)

updating can be done with the following commands.

```
UPDATE account SET branch_name='Main' WHERE branch_name="b1";
UPDATE account SET branch_name='Main' WHERE branch_name="b2";
UPDATE account SET customer_name='cust1' WHERE customer_name="cust2";
```

```
+-----+
| customer_name |
+-----+
| cust1        |
+-----+
```

1 row in set (0.00 sec)

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

```
SELECT d.customer_name
FROM account a,branch b,depositor d
WHERE b.branch_name=a.branch_name AND
a.accno=d.accno AND
b.branch_city='c3'
GROUP BY d.customer_name
HAVING COUNT(distinct b.branch_name)=(
SELECT COUNT(branch_name)
FROM branch
WHERE branch_city='c3');
```

```
+-----+
| customer_name |
+-----+
| cust3        |
+-----+
```

1 row in set (0.00 sec)

v. 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='c5');
```

Query OK, 1 row affected (0.04 sec)

```
SELECT * FROM account;
```

```
+-----+-----+-----+
| accno | branch_name | balance |
+-----+-----+-----+
| 12 | b1 | 3000 |
| 22 | b2 | 4000 |
| 32 | b3 | 5000 |
| 42 | b4 | 6000 |
+-----+-----+-----+
```

4 rows in set (0.00 sec)

BIBLIOGRAPHY

1. Elmasri and Navathe: Fundamentals of Database Systems, 5th Edition, Addison- Wesley, 2007
2. Raghu Ramakrishnan and Johannes Gehrke: Database Management Systems, 3rdEdition, McGraw-Hill, 2003.
3. Silberschatz, Korth and Sudharshan: Data base System Concepts, 5th Edition, Mc- GrawHill, 2006.
4. C.J. Date, A. Kannan, S. Swamynatham: A Introduction to Database Systems, 8thEdition, Pearson education, 2006.

VIVA QUESTIONS

1. Define Data.
2. Define Information.
3. Define Database.
4. Define DBMS.
5. What do you mean by processed data?
6. What do you mean by data management?
7. Which are the actions that are performed on the database?
8. Mention the different types of DBMS.
9. Define Data model.
10. Mention the different types of Data models.
11. Why database approach is advantageous than the file system approach?
12. Who is called as the father of RDBMS?
13. What do you mean by redundant data?
14. What do you mean by Data duplication?
15. Mention the different relational algebra operations.
16. Mention the different User interfaces provided by the database system.
17. Mention the different languages provided by the database system
18. What is the difference between select operation in relational algebra and in SQL?
19. What is the difference between JOIN and Cartesian product?
20. Mention the different types of Join operations.
21. What is the difference between EQUIJOIN and NATURAL JOIN?
22. What is the difference between OUTER JOIN and JOIN.?
23. What is the difference between OUTER UNION and UNION?
24. What do you mean by Union Compatibility.?
25. What do you mean by Type Compatibility?
26. Mention the different types of relational constraints.
27. Mention the different types of structural constraints
28. What do you mean by cardinality?
29. What do you mean by cardinality ratio?
30. What do you mean by degree of a relation?
31. What do you mean by entity integrity constraint?
32. What do you mean by referential integrity constraint?
33. What do you mean by NULL constraint?
34. What do you mean by unique constraint?
35. What do you mean by Check constraint?
36. Define functional dependency.
37. Define normalization.
38. Define normal form
39. Mention the different types of normal forms
40. What is the difference between 3NF and BCNF?
41. What do you mean by JOIN dependencies?
42. What do you mean by Inclusion dependencies?
43. What do you mean by Template dependencies?
44. What do you mean by Multivalued dependencies?
45. Define Project Join Normal form.
46. Define Domain Key Normal form.
47. Mention the informal guidelines for database design.

48. Define super key.
49. Define primary key.
50. Define foreign key.
51. Define unique key.
52. Define prime attribute.
53. Define trivial functional dependency.
54. When a FD is said to be fully FD?
55. Mention the different Armstrong's inference rules.
56. Why Armstrong's inference rules are said to be sound and complete?
57. Define denormalisation.
58. Define Transaction.
59. Mention the ACID properties.
60. Define schedule.
61. Is DBMS usage always advisable or some times we may depend on file base systems? Comment on the statement by describing the situation where DBMS is not a better option & file base systems is better.
62. Describe 3-level architecture of DBMS with details of languages associated at different levels plus the level of data independence.
63. How logical architecture of DBMS differs from physical architecture?
64. Create an E R diagram and relational schema to hold information about the situation in many institutions affiliated to some University, many teachers of different disciplines are teaching to many students enrolled in many courses offered by the university to the students through the institutions. Use concept of keys, aggregation, generalization, cardinality etc. in a proper way.
65. What is the utility of relational algebra & relational calculus? Name some software's based on these concepts?
66. Comment on the statement "Set theory has contributed a lot to RDBMS" support it with the help of suitable examples.
67. "Redundancy of data is many times beneficial" Justify the statement, also describe the situation when redundancy will mess up the current data base status, at that instance of time what actions you will prefer to take.
68. In Oracle we are having variety of versions Oracle 8, Oracle 9, etc, what does the associated number mean. Again we are having Oracle 8i, Oracle 9i etc, what does this "i" mean.
69. Describe the various file organization techniques? How a binary tree is different from B-tree and B+ tree? Under which situation we need to use B+ tree or B tree. Prove "Any relation which is in BCNF is in 3NF, but converse is not true"
70. Which functional dependencies are to be removed to achieve respective normal form? Discuss all the normal forms up to 4NF?
71. What is the mathematical basis of SQL? The SQL statement: select * from student will perform like projection or selection? Give details in support of your answer.