1. **Consider the customer-sale scenario given below.**

**CUSTOMER(Cust id : integer, cust\_name: string)**

**ITEM(item\_id: integer, item\_name: string, price: integer)**

**SALE(bill\_no: integer, bill\_date: date, cust\_id: integer, item\_id: integer, qty\_sold: integer)**

For the above schema, perform the following:

a) Create the tables with the appropriate integrity constraints

b) Insert around 10 records in each of the tables

c) List all the bills for the current date with the customer names and item numbers

d) List the total Bill details with the quantity sold, price of the item and the final amount

e) List the details of the customer who have bought a product which has a price>200

f) Give a count of how many products have been bought by each customer

g) Give a list of products bought by a customer having cust\_id as 5

h) List the item details which are sold as of today

i) Create a view which lists out the bill\_no, bill\_date, cust\_id, item\_id, price, qty\_sold, amount.

1. create table customer1 (cust\_id number(5) primary key, cust\_name varchar2(15));
2. desc customer1
3. create table item(item\_id number(4) primary key, item\_name varchar2(15), price number);
4. create table sale(bill\_no number(5) primary key,bill\_date date, cust\_id number(5) references customer1(cust\_id), item\_id number(4) references item(item\_id),qty\_sold number(4));
5. desc customer1;

5. dsec item;

6. dsec sale;

7. insert into customer1 values(&custid,’&custname’);

8. select \* from customer1;

9 .insert into item values(&item\_id,’&item\_name’,&price)

10. select \* from item;

11. insert into Sale values(1450,to\_date(‘04-Jan-2008’,’dd-mm-yyyy’),100,2124.2)

12. select \* from sale;

QUERIES

1. List all the bills for the current date with the customer names and item numbers

select c.cust\_name, i.item\_id, s.bill\_no from customer1 c, item I, sale s

where c.cust\_id=s.cust\_id and s.bill\_date=to\_char(sysdate);

2. List the total Bill details with the quantity sold, price of the item and the final amount.

select i.price,s.item\_id, (i.price\*s.qty\_sold) as total from item I, sale s where i.item\_id=s.item\_id group by ( i.price ,s.item\_id,s.qty\_sold);

3. List the details of the customer who have bought a product which has a price>200

select c.cust\_id, c.cust\_name from customer1 c, sale s, item i where i.price>50 and

c.cust\_id=s.cust\_id and i.item\_id=s.item\_id;

4.Give a count of how many products have been bought by each customer

select cust\_id, count(item\_id) from sale group by cust\_id;

5.Give a list of products bought by a customer having cust\_id as 5

select i.item\_name from item i, sale s where s.cust\_id=101 and i.item\_id=s.item\_id;

6. List the item details which are sold as of today

select i.item\_id, i.item\_name from item I, sale s where i.item\_id=s.item\_id and s.bill\_date=to\_char(sysdate);

7. Create a view which lists out the bill\_no, bill\_date, cust\_id, item\_id, price, qty\_sold, amount

create view cust as (select s.bill\_no, s.bill\_date, c.cust\_id, i.item\_id, i.price, s.qty\_sold from customer1 c,sale s, item i where c.cust\_id=s.cust\_id and i.item\_id=s.item\_id);

select \* from cust;

2.**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, Programme\_id, No-of\_Copies)

BOOK\_LENDING(Book\_id, Programme\_id, Card\_No, Date\_Out, Due\_Date)

LIBRARY\_BRANCH (Programme\_id, Programme\_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 Programme, 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.

1. CREATE TABLE PUBLISHER

(NAME VARCHAR2 (20) PRIMARY KEY, PHONE INTEGER, ADDRESS VARCHAR2 (20));

2. CREATE TABLE BOOK

(BOOK\_ID INTEGER PRIMARY KEY, TITLE VARCHAR2 (20), PUB\_YEAR VARCHAR2 (20), PUBLISHER\_NAME REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

3. CREATE TABLE BOOK\_AUTHORS

(AUTHOR\_NAME VARCHAR2 (20), BOOK\_ID REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE, PRIMARY KEY (BOOK\_ID, AUTHOR\_NAME));

4. CREATE TABLE LIBRARY\_BRANCH

(BRANCH\_ID INTEGER PRIMARY KEY, BRANCH\_NAME VARCHAR2 (50),

ADDRESS VARCHAR2 (50));

5. CREATE TABLE BOOK\_COPIES

(NO\_OF\_COPIES INTEGER, BOOK\_ID REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE, BRANCH\_ID REFERENCES LIBRARY\_BRANCH (BRANCH\_ID) ON DELETE CASCADE, PRIMARY KEY (BOOK\_ID, BRANCH\_ID));

6. CREATE TABLE CARD

(CARD\_NO INTEGER PRIMARY KEY);

7. CREATE TABLE BOOK\_LENDING

(DATE\_OUT DATE, DUE\_DATE DATE, BOOK\_ID REFERENCES BOOK (BOOK\_ID) ON DELETE CASCADE,

BRANCH\_ID REFERENCES LIBRARY\_BRANCH (BRANCH\_ID) ON DELETE CASCADE,

CARD\_NO REFERENCES CARD (CARD\_NO) ON DELETE CASCADE, PRIMARY KEY (BOOK\_ID, BRANCH\_ID, CARD\_NO));

desc\*all;

INSERT INTO PUBLISHER VALUES (‗MCGRAW-HILL‘, 9989076587, ‗BANGALORE‘);

INSERT INTO PUBLISHER VALUES (‗PEARSON‘, 9889076565, ‗NEWDELHI‘);

INSERT INTO PUBLISHER VALUES (‗RANDOM HOUSE‘, 7455679345, ‗HYDRABAD‘);

INSERT INTO PUBLISHER VALUES (‗HACHETTE LIVRE‘, 8970862340, ‗CHENAI‘);

INSERT INTO PUBLISHER VALUES (‗GRUPO PLANETA‘, 7756120238, ‗BANGALORE‘);

INSERT INTO BOOK VALUES (1,‘DBMS‘,‘JAN-2017‘, ‗MCGRAW-HILL‘);

INSERT INTO BOOK VALUES (2,‘ADBMS‘,‘JUN-2016‘, ‗MCGRAW-HILL‘);

INSERT INTO BOOK VALUES (3,‘CN‘,‘SEP-2016‘, ‗PEARSON‘);

INSERT INTO BOOK VALUES (4,‘CG‘,‘SEP-2015‘, ‗GRUPO PLANETA‘);

INSERT INTO BOOK VALUES (5,‘OS‘,‘MAY-2016‘, ‗PEARSON‘);

INSERT INTO BOOK\_AUTHORS VALUES (‘NAVATHE‘, 1);

INSERT INTO BOOK\_AUTHORS VALUES (‘NAVATHE‘, 2);

INSERT INTO BOOK\_AUTHORS VALUES (‘TANENBAUM‘, 3);

INSERT INTO BOOK\_AUTHORS VALUES (‘EDWARD ANGEL‘, 4);

INSERT INTO BOOK\_AUTHORS VALUES (‘GALVIN‘, 5);

INSERT INTO LIBRARY\_BRANCH VALUES (10,‘RR NAGAR‘,‘BANGALORE‘);

INSERT INTO LIBRARY\_BRANCH VALUES (11,‘RNSIT‘,‘BANGALORE‘);

INSERT INTO LIBRARY\_BRANCH VALUES (12,‘RAJAJI NAGAR‘, ‘BANGALORE‘);

INSERT INTO LIBRARY\_BRANCH VALUES (13,‘NITTE‘,‘MANGALORE‘);

INSERT INTO LIBRARY\_BRANCH VALUES (14,‘MANIPAL‘,‘UDUPI‘);

INSERT INTO BOOK\_COPIES VALUES (10, 1, 10);

INSERT INTO BOOK\_COPIES VALUES (5, 1, 11);

INSERT INTO BOOK\_COPIES VALUES (2, 2, 12);

INSERT INTO BOOK\_COPIES VALUES (5, 2, 13);

INSERT INTO BOOK\_COPIES VALUES (7, 3, 14);

INSERT INTO BOOK\_COPIES VALUES (1, 5, 10);

INSERT INTO BOOK\_COPIES VALUES (3, 4, 11);

INSERT INTO CARD VALUES (100);

INSERT INTO CARD VALUES (101);

INSERT INTO CARD VALUES (102);

INSERT INTO CARD VALUES (103);

INSERT INTO CARD VALUES (104);

INSERT INTO BOOK\_LENDING VALUES (‘01-JAN-17‘,‘01-JUN-17‘, 1, 10, 101);

INSERT INTO BOOK\_LENDING VALUES (‘11-JAN-17‘,‘11-MAR-17‘, 3, 14, 101);

INSERT INTO BOOK\_LENDING VALUES (‘21-FEB-17‘,‘21-APR-17‘, 2, 13, 101);

INSERT INTO BOOK\_LENDING VALUES (‘15-MAR-17‘,‘15-JUL-17‘, 4, 11, 101);

INSERT INTO BOOK\_LENDING VALUES (‗12-APR-17‘,‘12-MAY-17‘, 1, 11, 104);

SELECT \* FROM PUBLISHER;

Select \* from all;

QUERIES

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

SELECT B.BOOK\_ID, B.TITLE, B.PUBLISHER\_NAME, A.AUTHOR\_NAME, C.NO\_OF\_COPIES, L.BRANCH\_ID FROM BOOK B, BOOK\_AUTHORS A, BOOK\_COPIES C, LIBRARY\_BRANCH L

WHERE B.BOOK\_ID=A.BOOK\_ID AND B.BOOK\_ID=C.BOOK\_ID AND L.BRANCH\_ID=C.BRANCH\_ID;

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 ‘01-JUL-2017‘ GROUP BY CARD\_NO HAVING COUNT (\*)>3;

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=3;

select \* from book;

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

CREATE VIEW V\_PUBLICATION AS SELECT PUB\_YEAR FROM BOOK;

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

CREATE VIEW V\_BOOKS AS SELECT B.BOOK\_ID, B.TITLE, C.NO\_OF\_COPIES

FROM BOOK B, BOOK\_COPIES C, LIBRARY\_BRANCH L WHERE B.BOOK\_ID=C.BOOK\_ID

AND C.BRANCH\_ID=L.BRANCH\_ID;

**3 Consider the Employee-pay scenario given below.**

EMPLOYEE(**emp\_id : integer**, emp\_name: string)

DEPARTMENT(**dept\_id: integer**, dept\_name:string)

PAYDETAILS(**emp\_id : integer**, **dept\_id: integer**, basic: integer, deductions: integer, additions: integer, DOJ: date)

PAYROLL**(emp\_id : integer**, pay\_date: date)

For the above schema, perform the following:

a) Create the tables with the appropriate integrity constraints

b) Insert around 10 records in each of the tables

c) List the employee details department wise

d) List all the employee names who joined after particular date

e) List the details of employees whose basic salary is between 10,000 and 20,000

f) Give a count of how many employees are working in each department

g) Give a names of the employees whose netsalary>10,000

1. create table employee(emp\_id int(5) primary key,emp\_name varchar2(25));
2. create table department(dept\_id int(5) primary key,dept\_name varchar2(20));
3. create table paydetails(emp\_id int(5) references employee(emp\_id),dept\_id int(5) reerences department(dept\_id),basic int(7,2),deductions int(5,2),additions int(5,2),doj date);
4. create table payroll(emp\_id int(5)references employee(emp\_id),pay\_date date);
5. desc employee;

6. desc department;

7. desc paydetails;

8. desc payroll;

9. insert into employee values(&emp\_id,’&emp\_name’);

select \* from employee;

10. insert  into department values(&dept\_id,’&dept\_name’);

select \* from department;

11. insert into paydeatils values(&emp\_id,&dept\_id, &basic,&deductions,&additions,&doj);

select \* from paydeatils;

12. insert into payroll values(&emp\_id,’&date’);

 select \* from payroll;

QUERIES

13. List all the employee names who joined after particular date

select e,empname from employee e,paydet p where e.empid=p.empid

and p.doj>=’05-mar-06’;

14.List the details of employees whose basic salary is between 10k and 20k

**select e.emp\_id , e.emp\_name,d.dept\_id , d.dept\_name , pd.basic from employee e , department d , paydetails pd , payroll pr where e.emp\_id=pd.emp\_id and d.dept\_id=pd.dept\_id and e.emp\_id=pr.emp\_id and pd.basic between 600 and 1000;**

15. Give a count of how many employees are working in each department

select count(empid),deptid from paydet group by deptid;

16. Give a names of the employees whose netsalary>10,000

select empname from employee where empid in(select empid from

paydet where basic-deduction>10000);

17. List the details for an employee\_id=5

select \* from employee where empid=5;

18. Create a view which lists out the emp\_name, department, basic, dedeuctions, netsalary

create view vw as select e.emp\_name , d.dept\_name , pd.basic,pd.deductions , (pd.basic+pd.additions-pd.deductions) netsalary from employee e, department d, paydetails pd,payroll pr where e.emp\_id=pd.emp\_id and d.dept\_id=pd.dept\_id and e.emp\_id=pr.emp\_id ;

select \* from vw ;

19. Create a view which lists the emp\_name and his netsalary

create view vew as select e.emp\_name , (pd.basic+pd.additions-pd.deductions) netsalary from employee e, department d, paydetails pd,payroll pr where e.emp\_id=pd.emp\_id and d.dept\_id=pd.dept\_id and e.emp\_id=pr.emp\_id ;

select \* from vew ;

4. Question 4

1. CREATE TABLE DEPARTMENT (DNO NUMBER (2), DNAME VARCHAR2 (20));

2. ALTER TABLE DEPARTMENT ADD PRIMARY KEY (DNO);

3. CREATE TABLE BRANCH (BCODE NUMBER (3), BNAME VARCHAR2 (25), DNO NUMBER (2));

4. ALTER TABLE BRANCH ADD PRIMARY KEY (BCODE);

5. ALTER TABLE BRANCH ADD FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO);

6. CREATE TABLE BRANCH\_COURSE (BCODE NUMBER(3),

CCODE NUMBER(4),

SEMESTER NUMBER(2));

7. ALTER TABLE BRANCH\_COURSE ADD PRIMARY KEY (BCODE, CCODE);

8. ALTER TABLE BRANCH\_COURSE ADD FOREIGN KEY (BCODE) REFERENCES BRANCH (BCODE);

9. ALTER TABLE BRANCH\_COURSE ADD FOREIGN KEY (CCODE) REFERENCES COURSE (CCODE);

10. CREATE TABLE STUDENT (ROLLNO NUMBER (5),

NAME VARCHAR2 (20),

DOB DATE, GENDER CHAR(2),

DOA DATE, BCODE NUMBER(3));

11. ALTER TABLE STUDENT ADD PRIMARY KEY (ROLLNO);

ALTER TABLE STUDENT ADD FOREIGN KEY (BCODE) REFERENCES BRANCH (BCODE);

ALTER TABLE ADD CONSTRAINT CHK CHECK (GENDER IN ('M','F'));

ALTER TABLE ADD CONSTRAINT CHK2 CHECK (DOA < TO\_DATE('31-4-2016,'DD-MM-YYYY');

12. CREATE TABLE COURSE (CCODE NUMBER (4), CNAME VARCHAR2 (25), CREDITS NUMBER (2), DNO NUMBER (2));

13. ALTER TABLE COURSE ADD PRIMARY KEY (CCODE);

14. ALTER TABLE COURSE ADD FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO));

15. CREATE TABLE ENROLLS (ROLLNO NUMBER (5), CCODE NUMBER (4), SESS VARCHAR2 (15), GRADE CHAR (2));

16. ALTER TABLE ENROLLS ADD PRIMARY KEY (ROLLNO, CCODE, SESS);

17. ALTER TABLE ENROLLS ADD FOREIGN KEY ROLLNO) REFERENCES STUDENT

(ROLLNO);

18. ALTER TABLE ENROLLS ADD FOREIGN KEY (CCODE) REFERENCES COURSE (CCODE);

19. INSERT INTO COURSE VALUES (1011, 'LINEAR ALGEBRA', 2,1);

20. INSERT INTO STUDENT VALUES ( 12001, 'RAMESH KAUSHIK', TO\_DATE( '3-4-1989',DD-MM-YYYY') ,'M' , TO\_DATE( '24-4-2016','DD-MM-YYYY'), 110);

21. INSERT INTO ENROLLS VALUES( 12001, 1112, ‘APRIL2013’,'D');

QUERIES

22. Develop a SQL query to list details of Departments that offer more than 3 branches.

SELECT \* FROM DEPARTMENT D WHERE D.DNO IN (SELECT B.DNO FROM BRANCH B GROUP BY B.DNO HAVING COUNT (B.DNO) > 3);

23. Develop a SQL query to list the details of Departments that offer more than 6 courses.

SELECT \* FROM DEPARTMENT D WHERE D.DNO IN (SELECT C.DNO FROM COURSE C GROUP BY C.DNO HAVING COUNT (C.CCODE) > 6);

24. Develop a SQL query to list the details of courses that are common for more than 3 branches.

SELECT \* FROM COURSE C WHERE C.CCODE IN (SELECT B.CCODE FROM BRANCH\_COURSE B GROUP BY B.CCODE HAVING COUNT (B.BCODE) > 3);

25. Develop a SQL query to list students who got ‘S’ in more than 2 courses during single enrollment.

SELECT \* FROM STUDENT S WHERE S.ROLLNO IN (SELECT E.ROLLNO FROM ENROLLS E WHERE E.GRADE = 'S' GROUP BY E.ROLLNO HAVING COUNT (E.GRADE) > 2);

26. Create a view that will keep track of the roll number, name and number of courses, a student has completed successfully.

CREATE VIEW STUDATA AS SELECT E.ROLLNO, S.NAME, COUNT (E.CCODE) AS CC FROM STUDENT S, ENROLLS E WHERE E.ROLLNO = S.ROLLNO AND E.GRADE ! = 'U' GROUP BY E.ROLLNO, S.NAME;

**5. 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.

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

CREATE TABLE ACTOR (

ACT\_ID NUMBER (3),

ACT\_NAME VARCHAR (20),

ACT\_GENDER CHAR (1),

PRIMARY KEY (ACT\_ID));

CREATE TABLE DIRECTOR (

DIR\_ID NUMBER (3),

DIR\_NAME VARCHAR (20),

DIR\_PHONE NUMBER (10),

PRIMARY KEY (DIR\_ID));

CREATE TABLE MOVIES (

MOV\_ID NUMBER (4),

MOV\_TITLE VARCHAR (25),

MOV\_YEAR NUMBER (4),

MOV\_LANG VARCHAR (12),

DIR\_ID NUMBER (3),

PRIMARY KEY (MOV\_ID),

FOREIGN KEY (DIR\_ID) REFERENCES DIRECTOR (DIR\_ID));

CREATE TABLE MOVIE\_CAST (

ACT\_ID NUMBER (3),

MOV\_ID NUMBER (4),

ROLE VARCHAR (10),

PRIMARY KEY (ACT\_ID, MOV\_ID),

FOREIGN KEY (ACT\_ID) REFERENCES ACTOR (ACT\_ID),

FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

CREATE TABLE RATING (

MOV\_ID NUMBER (4),

REV\_STARS VARCHAR (25),

PRIMARY KEY (MOV\_ID),

FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

desc\*all;

**Insertion of Values to Tables**

INSERT INTO ACTOR VALUES (301,‘ANUSHKA‘,‘F‘);

INSERT INTO ACTOR VALUES (302,‘PRABHAS‘,‘M‘);

INSERT INTO ACTOR VALUES (303,‘PUNITH‘,‘M‘);

INSERT INTO ACTOR VALUES (304,‘JERMY‘,‘M‘);

INSERT INTO DIRECTOR VALUES (60,‘RAJAMOULI‘, 8751611001);

INSERT INTO DIRECTOR VALUES (61,‘HITCHCOCK‘, 7766138911);

INSERT INTO DIRECTOR VALUES (62,‘FARAN‘, 9986776531);

INSERT INTO DIRECTOR VALUES (63,‘STEVEN SPIELBERG‘, 8989776530);

INSERT INTO MOVIES VALUES (1001,‘BAHUBALI-2‘, 2017, ‗TELAGU‘, 60);

INSERT INTO MOVIES VALUES (1002,‘BAHUBALI-1‘, 2015, ‗TELAGU‘, 60);

INSERT INTO MOVIES VALUES (1003,‘AKASH‘, 2008, ‗KANNADA‘, 61);

INSERT INTO MOVIES VALUES (1004,‘WAR HORSE‘, 2011, ‗ENGLISH‘, 63);

INSERT INTO MOVIE\_CAST VALUES (301, 1002, ‗HEROINE‘);

INSERT INTO MOVIE\_CAST VALUES (301, 1001, ‗HEROINE‘);

INSERT INTO MOVIE\_CAST VALUES (303, 1003, ‗HERO‘);

INSERT INTO MOVIE\_CAST VALUES (303, 1002, ‗GUEST‘);

INSERT INTO MOVIE\_CAST VALUES (304, 1004, ‗HERO‘);

INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);

INSERT INTO RATING VALUES (1004, 4);

SELECT \* FROM all;

QUERIES

**1. List the titles of all movies directed by ‘Hitchcock’.**

SELECT MOV\_TITLE FROM MOVIES

WHERE DIR\_ID IN (SELECT DIR\_ID FROM DIRECTOR WHERE DIR\_NAME = ‗HITCHCOCK’);

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

SELECT MOV\_TITLE FROM MOVIES M, MOVIE\_CAST MV

WHERE M.MOV\_ID=MV.MOV\_ID AND ACT\_ID IN (SELECT ACT\_ID FROM MOVIE\_CAST GROUP BY ACT\_ID HAVING COUNT (ACT\_ID)>1) GROUP BY MOV\_TITLE HAVING COUNT (\*)>1;

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

SELECT ACT\_NAME, MOV\_TITLE, MOV\_YEAR 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 NOT BETWEEN 2000 AND 2015;c

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, MAX (REV\_STARS) FROM MOVIES INNER JOIN RATING USING (MOV\_ID) GROUP BY MOV\_TITLE HAVING MAX (REV\_STARS)>0

ORDER BY MOV\_TITLE;

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

WHERE DIR\_ID IN (SELECT DIR\_ID

FROM DIRECTOR

WHERE DIR\_NAME = ‗STEVEN SPIELBERG‘));