**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.

**Schema Creation**

Create schema Library;

Select the Library schema on the left pane by double click on it.(refresh may require)

**Table Creation:**

**PUBLISHER**

CREATE TABLE PUBLISHER( NAME VARCHAR(18) PRIMARY KEY, ADDRESS VARCHAR(10), PHONE VARCHAR(10));

**BOOK**

CREATE TABLE BOOK( BOOK\_ID INTEGER PRIMARY KEY, TITLE VARCHAR(20),

PUBLISHER\_NAME VARCHAR(20) REFERENCES PUBLISHER(NAME), PUB\_YEAR numeric);

**BOOK\_AUTHORS**

CREATE TABLE BOOK\_AUTHORS( BOOK\_ID INTEGER , AUTHOR\_NAME VARCHAR(20), PRIMARY KEY(BOOK\_ID), FOREIGN KEY (book\_id) REFERENCES Book (book\_id) ON DELETE CASCADE);

**LIBRARY\_BRANCH**

CREATE TABLE LIBRARY\_BRANCH( BRANCH\_ID INTEGER PRIMARY KEY, BRANCH\_NAME VARCHAR(18), ADDRESS VARCHAR(15));

**BOOK\_COPIES**

CREATE TABLE BOOK\_COPIES( BOOK\_ID INTEGER, BRANCH\_ID INTEGER , NO\_OF\_COPIES INTEGER, PRIMARY KEY(BOOK\_ID,BRANCH\_ID), FOREIGN KEY (book\_id) REFERENCES Book (book\_id) ON DELETE CASCADE, FOREIGN KEY (BRANCH\_ID) REFERENCES LIBRARY\_BRANCH(BRANCH\_ID) ON DELETE CASCADE);

**BOOK\_LENDING**

CREATE TABLE BOOK\_LENDING( BOOK\_ID INTEGER, BRANCH\_ID INTEGER, CARD\_NO INTEGER, DATE\_OUT DATE, DUE\_DATE DATE, PRIMARY KEY(BOOK\_ID,BRANCH\_ID,CARD\_NO) , FOREIGN KEY (book\_id) REFERENCES BOOK(BOOK\_ID) ON DELETE CASCADE, FOREIGN KEY (BRANCH\_ID) REFERENCES LIBRARY\_BRANCH(BRANCH\_ID) ON DELETE CASCADE);

**Values for tables:**

**PUBLISHER**

INSERT INTO PUBLISHER VALUES('PEARSON','BANGALORE','9875462530');

INSERT INTO PUBLISHER VALUES('MCGRAW','NEWDELHI','7845691234');

INSERT INTO PUBLISHER VALUES('SAPNA','BANGALORE','7845963210');

**BOOK**

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

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

INSERT INTO BOOK VALUES(3333,'ANOTOMY','PEARSON',2010);

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

**BOOK\_AUTHORS**

INSERT INTO BOOK\_AUTHORS VALUES(1111,'SOMMERVILLE');

INSERT INTO BOOK\_AUTHORS VALUES(2222,'NAVATHE');

INSERT INTO BOOK\_AUTHORS VALUES(3333,'HENRY GRAY');

INSERT INTO BOOK\_AUTHORS VALUES(4444,'THOMAS');

**LIBRARY\_BRANCH**

INSERT INTO LIBRARY\_BRANCH VALUES(11,'CENTRAL TECHNICAL','MG ROAD');

INSERT INTO LIBRARY\_BRANCH VALUES(22,'MEDICAL','BH ROAD');

INSERT INTO LIBRARY\_BRANCH VALUES(33,'CHILDREN','SS PURAM');

INSERT INTO LIBRARY\_BRANCH VALUES(44,'SECRETARIAT','SIRAGATE');

INSERT INTO LIBRARY\_BRANCH VALUES(55,'GENERAL','JAYANAGAR');

**BOOK\_COPIES**

INSERT INTO BOOK\_COPIES VALUES(1111,11,5);

INSERT INTO BOOK\_COPIES VALUES(3333,22,6);

INSERT INTO BOOK\_COPIES VALUES(4444,33,10);

INSERT INTO BOOK\_COPIES VALUES(2222,11,12);

INSERT INTO BOOK\_COPIES VALUES(4444,55,3);

**BOOK\_LENDING**

INSERT INTO BOOK\_LENDING VALUES(2222,11,1,'2017-01-10','2017-08-20');

INSERT INTO BOOK\_LENDING VALUES(3333,22,2,'2017-07-09','2017-08-12');

INSERT INTO BOOK\_LENDING VALUES(4444,55,1,'2017-04-11','2017-08-09');

INSERT INTO BOOK\_LENDING VALUES(2222,11,5,'2017-08-09','2017-08-19');

INSERT INTO BOOK\_LENDING VALUES(4444,33,1,'2017-06-10','2017-08-15');

INSERT INTO BOOK\_LENDING VALUES(1111,11,1,'2017-05-12','2017-06-10');

INSERT INTO BOOK\_LENDING VALUES(3333,22,1,'2017-07-10','2017-07-15');

**Queries:**

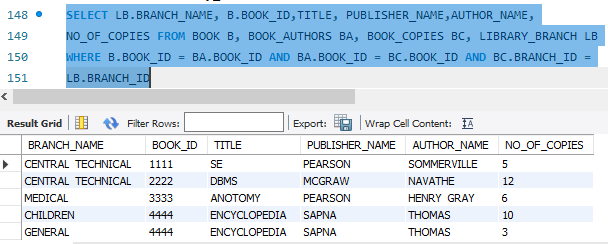
**1) Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.**

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;



**2) Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017**

**to Jun 2017.**

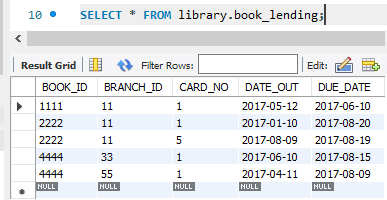
SELECT CARD\_NO, count(\*)

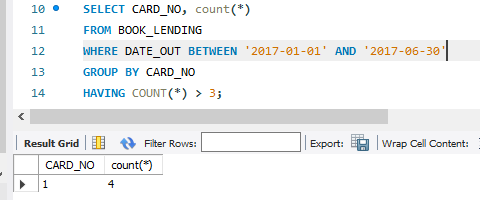
FROM BOOK\_LENDING

WHERE DATE\_OUT BETWEEN '2017-01-01' AND '2017-06-30'

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 = '3333';

BOOK\_AUTHORS

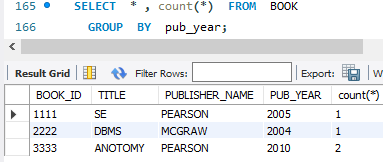
CREATE TABLE BOOK\_AUTHORS( BOOK\_ID INTEGER , AUTHOR\_NAME VARCHAR(20), PRIMARY KEY(BOOK\_ID), FOREIGN KEY (book\_id) REFERENCES Book (book\_id) ON DELETE CASCADE);

Here we set ‘ON DELETE CASCADE’ during the creation of table. So deletion of 1 row in ‘book’ table will delete a row in book\_authors relation too…

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

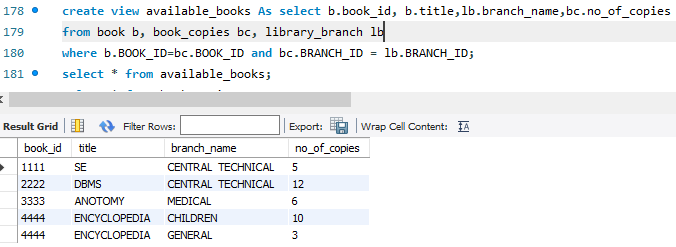
**SELECT \* , count(\*) FROM BOOK**

**GROUP BY pub\_year;**



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

First insert the deleted book on each respective table. Then we will get the following Result.



**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.

**Schema Creation**

Create schema ORDERS;

Select the ORDERS schema on the left pane by double click on it.(refresh may require)

**Table Creation:**

**SALESMAN**

CREATE TABLE SALESMAN(

SALESMAN\_ID integer PRIMARY KEY,

Nam VARCHAR(10) NOT NULL,

CITY VARCHAR(15) NOT NULL,

COMMISSION integer);

**CUSTOMER**

CREATE TABLE CUSTOMER(

CUSTOMER\_ID integer(5) ,

CUST\_NAME VARCHAR(10) NOT NULL,

CITY VARCHAR(10) NOT NULL,

GRADE int(5) NOT NULL,

SALESMAN\_ID integer,

PRIMARY KEY(CUSTOMER\_ID),

foreign key(SALESMAN\_ID) REFERENCES SALESMAN(SALESMAN\_ID) ON DELETE CASCADE);

**ORDERS**

CREATE TABLE ORDERS(

ORD\_NO INT PRIMARY KEY,

PURCHASE\_AMT INT NOT NULL,

ORD\_DATE DATE NOT NULL,

CUSTOMER\_ID integer ,

SALESMAN\_ID integer,

foreign key(SALESMAN\_ID) REFERENCES SALESMAN(SALESMAN\_ID) ON DELETE CASCADE,

foreign key(CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID) ON DELETE CASCADE);

**VALUES FOR TABLES**

**SALESMAN**

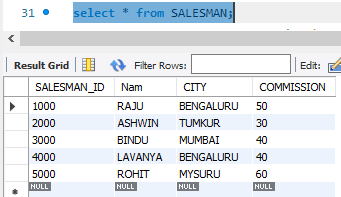
INSERT INTO SALESMAN VALUES(1000,'RAJU','BENGALURU',50);

INSERT INTO SALESMAN VALUES(2000,'ASHWIN','TUMKUR',30);

INSERT INTO SALESMAN VALUES(3000,'BINDU','MUMBAI',40);

INSERT INTO SALESMAN VALUES(4000,'LAVANYA','BENGALURU',40);

INSERT INTO SALESMAN VALUES(5000,'ROHIT','MYSURU',60);



**CUSTOMER**

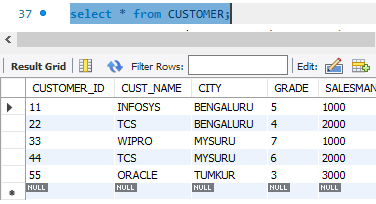
INSERT INTO CUSTOMER VALUES(11,'INFOSYS','BENGALURU',5,1000);

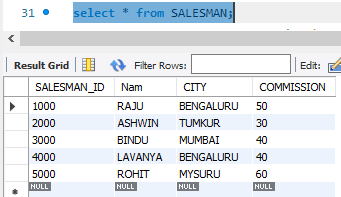
INSERT INTO CUSTOMER VALUES(22,'TCS','BENGALURU',4,2000);

INSERT INTO CUSTOMER VALUES(33,'WIPRO','MYSURU',7,1000);

INSERT INTO CUSTOMER VALUES(44,'TCS','MYSURU',6,2000);

INSERT INTO CUSTOMER VALUES(55,'ORACLE','TUMKUR',3,3000);

List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION 

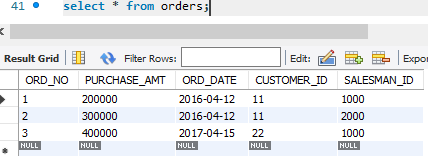
.

**ORDERS**

INSERT INTO orders VALUES(1,200000,'2016-04-12',11,1000);

INSERT INTO orders VALUES(2,300000,'2016-04-12',11,2000);

INSERT INTO orders VALUES(3,400000,'2017-04-15',22,1000);



1. Count the customers with grades above Bangalore’s average.

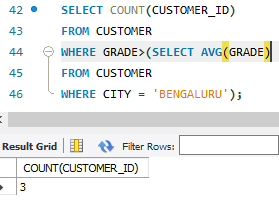
SELECT COUNT(CUSTOMER\_ID)

FROM CUSTOMER

WHERE GRADE>(SELECT AVG(GRADE)

FROM CUSTOMER

WHERE CITY = 'BENGALURU');



1. Find the name and numbers(of customers) of all salesmen who had more than one customer.

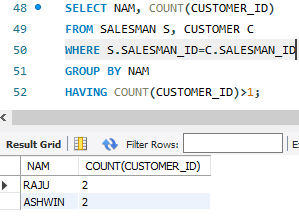
SELECT NAM, COUNT(CUSTOMER\_ID)

FROM SALESMAN S, CUSTOMER C

WHERE S.SALESMAN\_ID=C.SALESMAN\_ID

GROUP BY NAM

HAVING COUNT(CUSTOMER\_ID)>1;



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

(SELECT NAM

FROM SALESMAN S, CUSTOMER C

WHERE S.SALESMAN\_ID=C.SALESMAN\_ID AND

S.CITY=C.CITY)

UNION

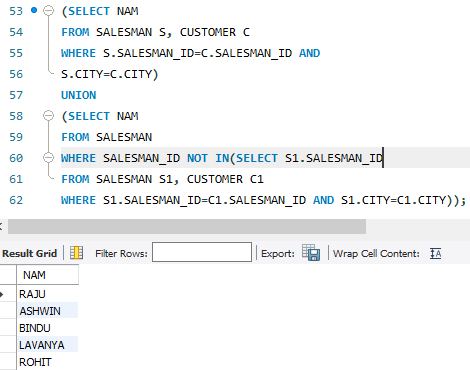
(SELECT NAM

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));



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

CREATE VIEW SALES\_HIGHERODER AS

SELECT SALESMAN\_ID, PURCHASE\_AMT

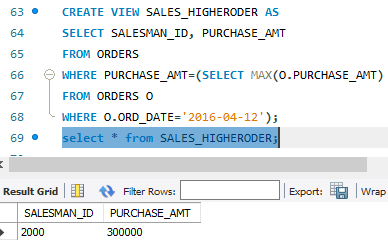
FROM ORDERS

WHERE PURCHASE\_AMT=(SELECT MAX(O.PURCHASE\_AMT)

FROM ORDERS O

WHERE O.ORD\_DATE='2016-04-12');

Select \* from SALES\_HIGHERODER;



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

DELETE from salesman

WHERE salesman\_id = 1000;

**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.

**Schema Creation**

Create schema movie;

Select the movie schema on the left pane by double click on it.(refresh may require)

**Table Creation:**

**ACTOR**

CREATE TABLE ACTOR(

ACT\_ID int(5) PRIMARY KEY,

ACT\_NAME VARCHAR(18) NOT NULL ,

ACT\_GENDER VARCHAR(2) NOT NULL);

**DIRECTOR**

CREATE TABLE DIRECTOR(

DIR\_ID int PRIMARY KEY,

DIR\_NAME VARCHAR(18) NOT NULL,

DIR\_PHONE VARCHAR(10) NOT NULL);

**MOVIES**

CREATE TABLE MOVIES(

MOV\_ID INT PRIMARY KEY,

MOV\_TITLE VARCHAR(10) NOT NULL,

MOV\_YEAR int NOT NULL,

MOV\_LANG VARCHAR(10) NOT NULL,

DIR\_ID INT NOT NULL,

foreign key(DIR\_ID) REFERENCES DIRECTOR(DIR\_ID)ON delete cascade);

**MOVIE\_CAST**

CREATE TABLE MOVIE\_CAST(

ACT\_ID INT NOT NULL ,

MOV\_ID INT NOT NULL ,

ROLE VARCHAR(10) NOT NULL,

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

foreign key(ACT\_ID) REFERENCES ACTOR(ACT\_ID) ON DELETE CASCADE,

foreign key (MOV\_ID) REFERENCES MOVIES(MOV\_ID)ON DELETE CASCADE);

**RATING**

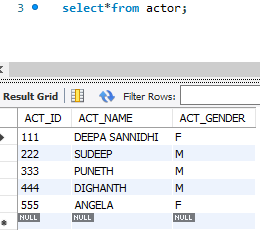
CREATE TABLE RATING(

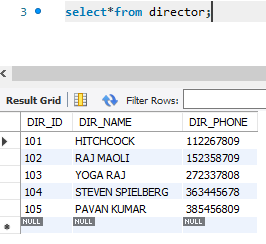
MOV\_ID int NOT NULL ,

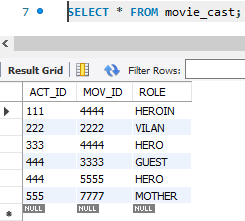
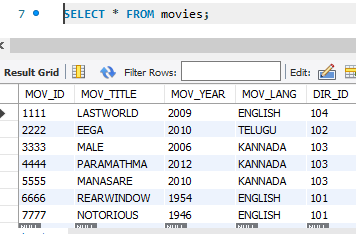
REV\_STARS int NOT NULL,

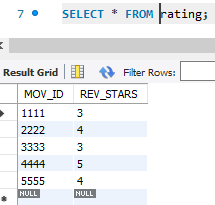
PRIMARY KEY(MOV\_ID),

foreign key(MOV\_ID) REFERENCES MOVIES(MOV\_ID) ON DELETE CASCADE);









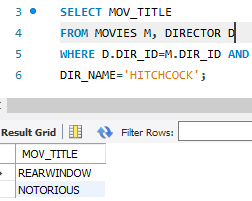
1. List the titles of all movies directed by ‘Hitchcock’.

SELECT MOV\_TITLE

FROM MOVIES M, DIRECTOR D

WHERE D.DIR\_ID=M.DIR\_ID AND

DIR\_NAME='HITCHCOCK';



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

SELECT MOV\_TITLE

FROM MOVIES M, MOVIE\_CAST MC

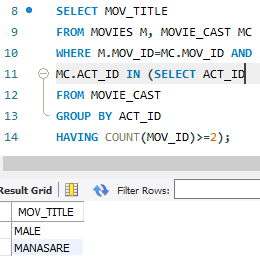
WHERE M.MOV\_ID=MC.MOV\_ID AND

MC.ACT\_ID IN (SELECT ACT\_ID

FROM MOVIE\_CAST

GROUP BY ACT\_ID

HAVING COUNT(MOV\_ID)>=2);



1. 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

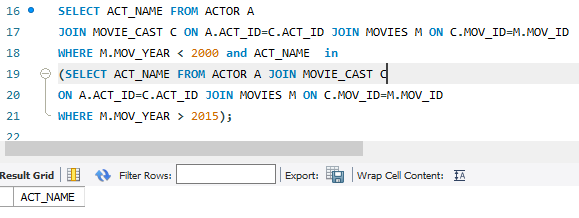
JOIN MOVIE\_CAST C ON A.ACT\_ID=C.ACT\_ID JOIN MOVIES M ON C.MOV\_ID=M.MOV\_ID

WHERE M.MOV\_YEAR < 2000 and ACT\_NAME in

(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

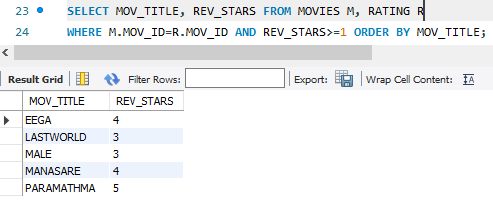
WHERE M.MOV\_YEAR > 2015);



1. 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;



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

UPDATE RATING

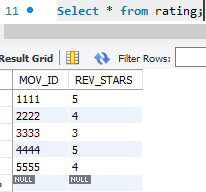
SET REV\_STARS=5

WHERE MOV\_ID IN (SELECT MOV\_ID

FROM MOVIES M, DIRECTOR D

WHERE M.DIR\_ID=D.DIR\_ID AND

DIR\_NAME='STEVEN SPIELBERG');



**COLLEGE DATABASE**

4). Consider the schema for College Database:

STUDENT (USN, SName, Address, Phone, Gender)

SEMSEC (SSID, Sem, Sec)

CLASS (USN, SSID)

SUBJECT (Subcode, Title, Sem, Credits)

IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

Write SQL queries to

1. List all the student details studying in fourth semester ‘C’ section.

2. Compute the total number of male and female students in each semester and in each section.

3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.

4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

5. Categorize students based on the following criterion:

If FinalIA = 17 to 20 then CAT = ‘Outstanding’

If FinalIA = 12 to 16 then CAT = ‘Average’

If FinalIA< 12 then CAT = ‘Weak’

Give these details only for 8th semester A, B, and C section students.

**CREATE TABLE STUDENT**

CREATE TABLE STUDENT

(USN VARCHAR(10) PRIMARY KEY,

SNAME VARCHAR(25),

ADDRESS VARCHAR(25),

PHONE VARCHAR(10),

GENDER CHAR(1));

**CREATE TABLE SEMSEC**

CREATE TABLE SEMSEC

(SSID VARCHAR(5) PRIMARY KEY,

SEM tinyint,

SEC CHAR(1));

**CREATE TABLE CLASS**

CREATE TABLE CLASS

(USN VARCHAR(10),

SSID VARCHAR(5),

PRIMARY KEY(USN,SSID),

FOREIGN KEY(USN) REFERENCES STUDENT(USN) on delete cascade, FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID)on delete cascade);

**CREATE TABLE SUBJECT**

CREATE TABLE SUBJECT

(SUBCODE VARCHAR(8) PRIMARY KEY, TITLE VARCHAR(20),

SEM tinyint, CREDITS tinyint);

**CREATE TABLE IAMARKS**

CREATE TABLE IAMARKS (USN VARCHAR(10), SUBCODE VARCHAR(8), SSID VARCHAR(5), TEST1 int, TEST2 int,TEST3 int, FINALIA int,

PRIMARY KEY(USN,SUBCODE,SSID),

FOREIGN KEY(USN) REFERENCES STUDENT(USN) on delete cascade,

FOREIGN KEY(SUBCODE) REFERENCES SUBJECT(SUBCODE) on delete cascade,

FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID) on delete cascade);

**STUDENT VALUES**

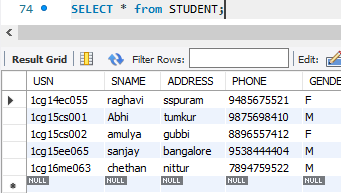
INSERT INTO STUDENT VALUES('1cg15cs001','Abhi','tumkur','9875698410','M');

INSERT INTO STUDENT VALUES('1cg15cs002','amulya','gubbi','8896557412','F');

INSERT INTO STUDENT VALUES('1cg16me063','chethan','nittur','7894759522','M');

INSERT INTO STUDENT VALUES('1cg14ec055','raghavi','sspuram','9485675521','F');

INSERT INTO STUDENT VALUES('1cg15ee065','sanjay','bangalore','9538444404','M');



**SEMSEC VALUES**

INSERT INTO SEMSEC VALUES('5A',5,'A');

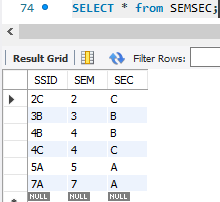
INSERT INTO SEMSEC VALUES('3B',3,'B');

INSERT INTO SEMSEC VALUES('7A',7,'A');

INSERT INTO SEMSEC VALUES('2C',2,'C');

INSERT INTO SEMSEC VALUES('4B',4,'B');

INSERT INTO SEMSEC VALUES('4C',4,'C');



**CLASS VALUES**

INSERT INTO CLASS VALUES('1cg15cs001','5A');

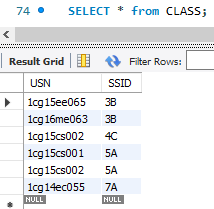
INSERT INTO CLASS VALUES('1cg15cs002','5A');

INSERT INTO CLASS VALUES('1cg16me063','3B');

INSERT INTO CLASS VALUES('1cg14ec055','7A');

INSERT INTO CLASS VALUES('1cg15ee065','3B');

INSERT INTO CLASS VALUES('1cg15cs002','4C');



**SUBJECT VALUES**

INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4);

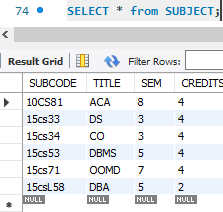
INSERT INTO SUBJECT VALUES ('15cs53','DBMS', 5, 4);

INSERT INTO SUBJECT VALUES ('15cs33','DS', 3, 4);

INSERT INTO SUBJECT VALUES ('15cs34','CO', 3, 4);

INSERT INTO SUBJECT VALUES ('15csL58','DBA', 5, 2);

INSERT INTO SUBJECT VALUES ('15cs71','OOMD', 7, 4);



**IAMARKS VALUES**

INSERT INTO IAMARKS VALUES ('1cg15cs001','15cs53','5A', 18,19,15,null);

INSERT INTO IAMARKS VALUES ('1cg15cs002','15cs53','5A', 15,16,14, null);

INSERT INTO IAMARKS VALUES ('1cg16me063','15cs33','3B', 10,15,16, null);

INSERT INTO IAMARKS VALUES ('1cg14ec055','15cs71','7A', 18,20,21, null);

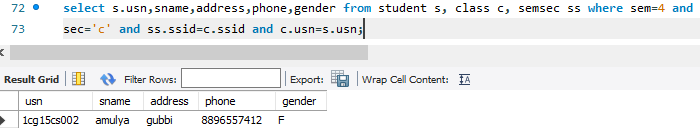
INSERT INTO IAMARKS VALUES ('1cg15ee065','15cs33','3B', 16,20,17, null);

INSERT INTO IAMARKS VALUES ('1cg15ee065','15cs53','4C', 19,20,18, null);

**1. List all the student details studying in fourth semester ‘C’ section.**

select s.usn,sname,address,phone,gender from student s, class c, semsec ss where sem=4 and

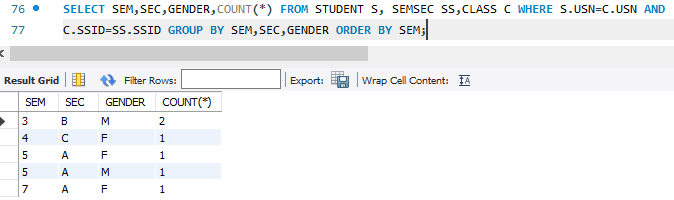
sec='c' and ss.ssid=c.ssid and c.usn=s.usn;



**2. Compute the total number of male and female students in each semester and in each section.**

SELECT SEM,SEC,GENDER,COUNT(\*) FROM STUDENT S, SEMSEC SS,CLASS C WHERE S.USN=C.USN AND

C.SSID=SS.SSID GROUP BY SEM,SEC,GENDER ORDER BY SEM;



1. **Create a view of Test1 marks of student USN ~~‘1BI15CS101’~~ ‘1cg15ee065’ in all subjects.**

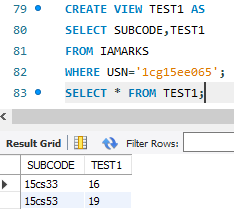
CREATE VIEW TEST1 AS

SELECT SUBCODE,TEST1

FROM IAMARKS

WHERE USN='1cg15ee065';

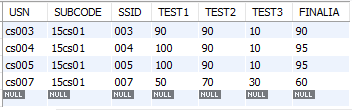
SELECT \* FROM TEST1;



1. **Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.**

UPDATE IAMARKS SET FINALIA = (TEST1+TEST2+TEST3-LEAST(TEST1,TEST2,TEST3)/2;

select \* from iamarks;



**5. Categorize students based on the following criterion:**

**If FinalIA = 17 to 20 then CAT = ‘Outstanding’ If FinalIA = 12 to 16 then CAT = ‘Average’**

**If FinalIA< 12 then CAT = ‘Weak’ Give these details only for ~~8th~~ (7th)semester A, B, and C section students.**

**SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,**

**CASE WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING' WHEN**

**SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,**

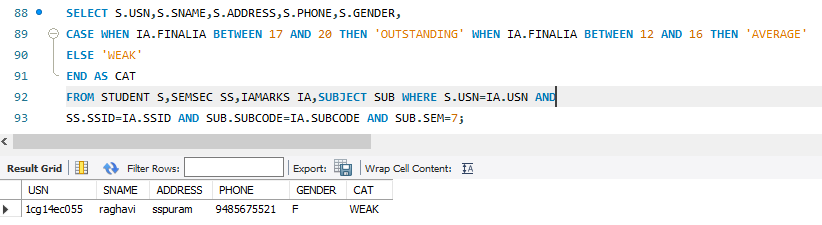
**CASE WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING' WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'**

**ELSE 'WEAK'**

**END AS CAT**

**FROM STUDENT S,SEMSEC SS,IAMARKS IA,SUBJECT SUB WHERE S.USN=IA.USN AND**

**SS.SSID=IA.SSID AND SUB.SUBCODE=IA.SUBCODE AND SUB.SEM=7;**



**5). Consider the schema for Company Database:**

**EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN,DNo)**

**DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate) DLOCATION (DNo,DLoc)**

**PROJECT (PNo, PName, PLocation, DNo) WORKS\_ON (SSN, PNo, Hours)**

**Write SQL queries to**

**1. Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’, either as a worker or as a manager of the department that controls the project.**

**2. Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent raise.**

**3. Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the maximum salary, the minimum salary, and the average salary in this department**

**4. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).**

**5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.**

**SCHEMA CREATION**

CREATE SCHEMA COMPANY;

**TABLE CREATION**

**TABLE DEPARTMENT**

CREATE TABLE DEPARTMENT(

DNO tinyint PRIMARY KEY, DNAME VARCHAR(15) NOT NULL, MGRSSN CHAR(10),

MGRSTARTDATE DATE);

**TABLE EMPLOYEE**

CREATE TABLE EMPLOYEE(

SSN CHAR(10) PRIMARY KEY,

NAME VARCHAR(18) NOT NULL, ADDRESS VARCHAR(18),

SEX VARCHAR(3), SALARY REAL, SUPER\_SSN CHAR(10),

DNO tinyint ,

FOREIGN KEY(DNO) REFERENCES DEPARTMENT(DNO));

**TABLE DLOCATION**

CREATE TABLE DLOCATION(

DLOC VARCHAR (20),

DNO tinyint,

PRIMARY KEY (DNO, DLOC),

foreign key(DNO) REFERENCES DEPARTMENT (DNO));

ALTER TABLE DEPARTMENT ADD CONSTRAINT FOREIGN KEY(MGRSSN) REFERENCES EMPLOYEE(SSN);

**TABLE PROJECT**

CREATE TABLE PROJECT(

PNO INTEGER PRIMARY KEY,

PNAME VARCHAR (20), PLOCATION VARCHAR (20), DNO tinyint,

foreign key(DNO) REFERENCES DEPARTMENT (DNO));

**TABLE WORKS\_ON**

CREATE TABLE WORKS\_ON(

SSN CHAR(10), PNO INTEGER, HOURS tinyint,

foreign key(SSN) REFERENCES EMPLOYEE (SSN),

foreign key(PNO) REFERENCES PROJECT(PNO),

PRIMARY KEY (SSN, PNO));

**INSERT VALUES**

DISABLE THE FOREIGN KEYS BEFORE INSERTING VALUES

An alternative to temporarily disable all the foreign keys: SET FOREIGN\_KEY\_CHECKS=0;

When you need to turn it on: SET FOREIGN\_KEY\_CHECKS=1;

INSERT INTO EMPLOYEE VALUES('111111','RAJ','BENGALURU','M',700000, null,1);

INSERT INTO EMPLOYEE VALUES('222222','RASHMI','MYSORE','F',400000,'111111',2);

INSERT INTO EMPLOYEE VALUES('333333','RAGAVI','TUMKUR','F',800000,null,3);

INSERT INTO EMPLOYEE VALUES('444444','RAJESH','TUMKUR','M',650000,'333333',3);

INSERT INTO EMPLOYEE VALUES('555555','RAVEESH','BENGALURU','M',500000,'333333',3);

INSERT INTO EMPLOYEE VALUES('666666','SCOTT','ENGLAND','M',700000,'444444',5);

INSERT INTO EMPLOYEE VALUES('777777','NIGANTH','GUBBI','M',200000,'222222',2);

INSERT INTO EMPLOYEE VALUES('888888','RAMYA','GUBBI','F',400000,'222222',3);

INSERT INTO EMPLOYEE VALUES('999999','VIDYA','TUMKUR','F',650000,'333333',3);

INSERT INTO EMPLOYEE VALUES('100000','GEETHA','TUMKUR','F',800000,'',3);

INSERT INTO DEPARTMENT VALUES(1,'RESEARCH','111111','2012-08-10');

INSERT INTO DEPARTMENT VALUES(2,'ACCOUNTS','222222','2010-08-10');

INSERT INTO DEPARTMENT VALUES(3,'AI','333333','2015-04-12');

INSERT INTO DEPARTMENT VALUES(4,'NETWORKS','111111','2018-05-14');

INSERT INTO DEPARTMENT VALUES(5,'BIGDATA','666666','2021-01-10');

INSERT INTO DLOCATION VALUES('MYSORE', 1);

INSERT INTO DLOCATION VALUES('TUMKUR', 1);

INSERT INTO DLOCATION VALUES('BENGALURU', 2);

INSERT INTO DLOCATION VALUES('GUBBI', 3);

INSERT INTO DLOCATION VALUES('DELHI', 4);

INSERT INTO DLOCATION VALUES('BENGALURU', 5);

INSERT INTO PROJECT VALUES(111,'IOT','GUBBI',3);

INSERT INTO PROJECT VALUES(222,'TEXTSPEECH','GUBBI',3);

INSERT INTO PROJECT VALUES(333,'IPSECURITY','DELHI',4);

INSERT INTO PROJECT VALUES(444,'TRAFICANAL', 'BENGALURU',5);

INSERT INTO PROJECT VALUES(555,'CLOUDSEC','DELHI',1);

INSERT INTO WORKS\_ON VALUES('666666',333,4);

INSERT INTO WORKS\_ON VALUES('666666',111,2);

INSERT INTO WORKS\_ON VALUES('111111',222,3);

INSERT INTO WORKS\_ON VALUES('555555',222,2);

INSERT INTO WORKS\_ON VALUES('333333',111,4);

INSERT INTO WORKS\_ON VALUES('444444',111,6);

INSERT INTO WORKS\_ON VALUES('222222',111,2);

1. Make a list of all project numbers for projects that involve an employee whose last name is

‘Scott’, either as a worker or as a manager of the department that controls the project.

(SELECT DISTINCT PNO

FROM PROJECT P, DEPARTMENT D, EMPLOYEE E WHERE P.DNO=D.DNO AND

SSN=MGRSSN AND

NAME='SCOTT')

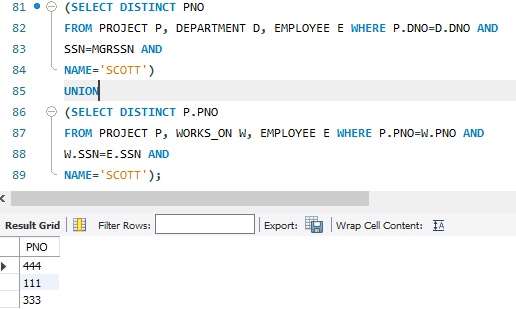
UNION

(SELECT DISTINCT P.PNO

FROM PROJECT P, WORKS\_ON W, EMPLOYEE E WHERE P.PNO=W.PNO AND

W.SSN=E.SSN AND

NAME='SCOTT');



2. Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10

percent raise.

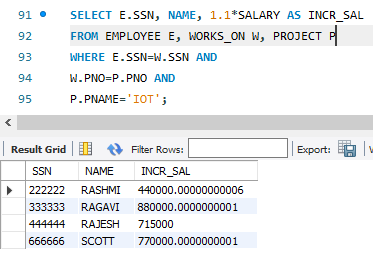
SELECT E.SSN, NAME, 1.1\*SALARY AS INCR\_SAL

FROM EMPLOYEE E, WORKS\_ON W, PROJECT P

WHERE E.SSN=W.SSN AND

W.PNO=P.PNO AND

P.PNAME='IOT';



3. Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the

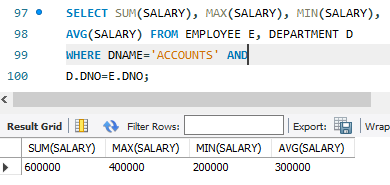
maximum salary, the minimum salary, and the average salary in this department.

SELECT SUM(SALARY), MAX(SALARY), MIN(SALARY),

AVG(SALARY) FROM EMPLOYEE E, DEPARTMENT D

WHERE DNAME='ACCOUNTS' AND

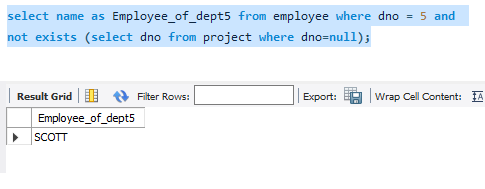
D.DNO=E.DNO;



1. Retrieve the name of each employee who works on all the projects controlled by department number 5 (use NOT EXISTS operator).

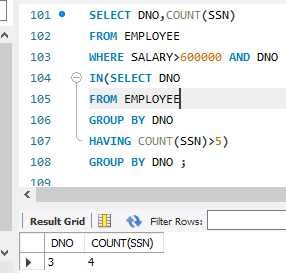
select name as Employee\_of\_dept5 from employee where dno = 5 and

not exists (select dno from project where dno=null);



5. For each department that has more than five employees, retrieve the department number

and the number of its employees who are making more than Rs. 6,00,000.



SELECT DNO,COUNT(SSN)

FROM EMPLOYEE

WHERE SALARY>600000 AND DNO

IN(SELECT DNO

FROM EMPLOYEE

GROUP BY DNO

HAVING COUNT(SSN)>5)

GROUP BY DNO ;