**DATABASE MANAGEMENT SYSTEMS LAB MANUAL**

| **Sl.**  **No** | **Program List** |
| --- | --- |
| 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 the details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc. 2. Get the particular borrowers who have borrowed more than 3 books from Jan 2017 to Jun 2017. 3. Delete a book in BOOK table and Update the contents of other tables using DML statements. 4. Create the view for BOOK table based on year of publication and demonstrate its working with a simple query. 5. Create a view of all books and its number of copies which are currently available in the Library. 6. Demonstrate the usage of view creation |
| 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 Bangalor’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).   1. Create a view that finds the salesman who has the customer with the highest   order of a day.   1. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted. 2. Create an index on (Customer (id)) to demonstrate the usage. |
| 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 2020 (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. |
| 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 fifth semester ‘B’ section. 2. Compute the total number of male and female students in each semester and in each section. 3. Create a view of Event 1 marks of student USN ‘01JST IS ’ in all subjects. 4. Calculate the Final IA (average of best two test marks) and update the corresponding table for all students. 5. Categorize students based on the following criterion:   If Final IA = 17 to 20 then CAT =‘Outstanding’  If Final IA = 12 to 16 then CAT = ‘Average’  If Final IA< 12 then CAT = ‘Weak’  Give these details only for 8th semester A, B, and C section students. |
| 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. (use NOT EXISTS operator).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. |
| 6 | Consider the schema of the call detail table to partitioned primary index:  CREATE TABLE calldetail (phone\_number DECIMAL(10) NOT NULL, call\_start TIMESTAMP, call\_duration INTEGER, call\_description VARCHAR(30))  PRIMARY INDEX (phone\_number, call\_start);  Demonstrate the query against this table be optimized by partitioning its primary index using partitioning techniques. |

**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 the details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
2. Get the particular borrowers who have borrowed more than 3 books from Jan 2020 to Jun 2022.
3. Delete a book in BOOK table and Update the contents of other tables using DML statements.
4. Create the view for BOOK table based on year of publication and demonstrate its working with a simple query.
5. Create a view of all books and its number of copies which are currently available in the Library.
6. Demonstrate the usage of view creation

### Table Creation:

**PUBLISHER**

SQL> **CREATE TABLE** PUBLISHER (

NAME VARCHAR (18) **PRIMARY KEY**,

ADDRESS VARCHAR (10),

PHONE VARCHAR (10));

Table created.

**BOOK**

SQL> **CREATE TABLE** BOOK (

BOOK\_ID INTEGER **PRIMARY KEY**,

TITLE VARCHAR (20),

PUBLISHER\_NAME VARCHAR (20) **REFERENCES** PUBLISHER(NAME)**ON DELETE CASADE**,

PUB\_YEAR INT (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('PEARSON','BANGALORE','9875462530');

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

SQL> INSERT INTO PUBLISHER VALUES('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,'ANOTOMY','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, '2017-01-10','2017-08-20'); |
| --- | --- | --- | --- | --- |
| SQL> | INSERT | INTO | BOOK\_LENDING | VALUES(3333,22,2, '2017-07-09','2017-08-12'); |
| SQL> | INSERT | INTO | BOOK\_LENDING | VALUES(4444,55,1, '2017-04-11','2017-08-09'); |
| SQL> | INSERT | INTO | BOOK\_LENDING | VALUES(2222,11,5, '2017-08-09','2017-08-19'); |
| SQL> | INSERT | INTO | BOOK\_LENDING | VALUES(4444,33,1, '2017-06-08','2017-08-15'); |
| SQL> | INSERT | INTO | BOOK\_LENDING | VALUES(1111,11,1, '2017-05-12','2017-06-10'); |
| SQL> | INSERT | INTO | BOOK\_LENDING | VALUES(3333,22,1, '2017-07-10','2017-07-15'); |

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

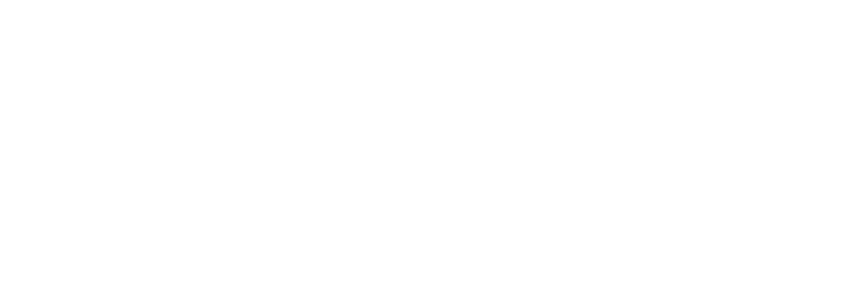
3333 HENRY GRAY

4444 THOMAS

4 rows selected.

| SQL>SELECT  NAME | \* FROM PUBLISHER;  ADDRESS |  | PHONE |
| --- | --- | --- | --- |
| PEARSON | BANGALORE |  | 9875462530 |
| MCGRAW | NEWDELHI |  | 7845691234 |
| SAPNA | BANGALORE |  | 7845963210 |

3 rows selected.

SQL> SELECT \* FROM BOOK\_COPIES; 

| 1111 |  | 11 |  | 5 |
| --- | --- | --- | --- | --- |
| 3333 |  | 22 |  | 6 |
| 4444 |  | 33 |  | 10 |
| 2222 |  | 11 |  | 12 |
| 4444 |  | 55 |  | 3 |

BOOK\_ID BRANCH\_ID NO\_OF\_COPIES

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 |

1. 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

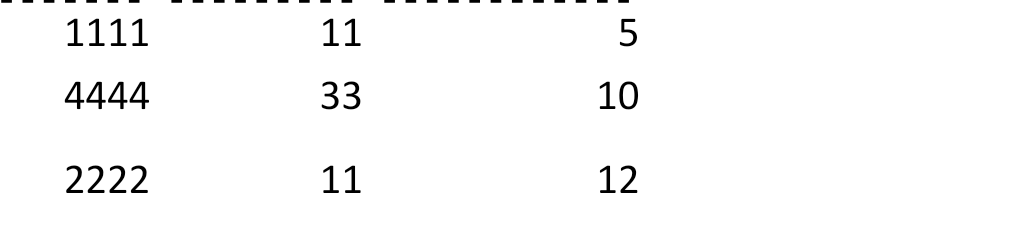
1. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

**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

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 |

1. Create the view for BOOK table based on year of publication and demonstrate its working

with a simple query.

CREATE VIEW V\_PUBLICATION AS SELECT BOOK\_ID, TITLE, PUBLISHER\_NAME, PUB\_YEAR FROM BOOK ORDER BY PUB\_YEAR;

View created.

SQL> SELECT \* FROM V\_ PUBLICATION ;

BOOK\_ID TITLE PUBLISHER\_NAME PUB\_YEAR

2222 DBMS MCGRAW 2004

1111 SE PEARSON 2005

3333 ANOTOMY PEARSON 2010

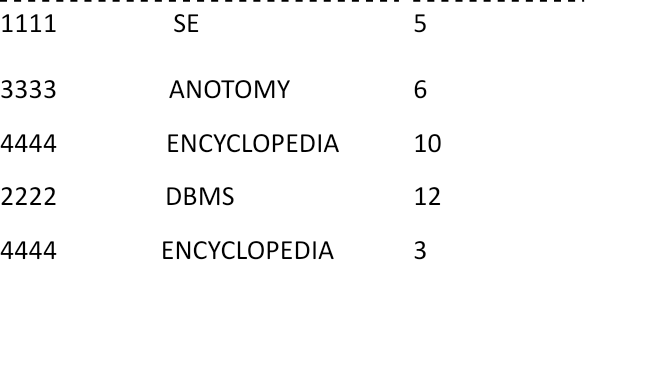
4444 ENCYCLOPEDIA SAPNA 2010

1. 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 ANDL. BRANCH\_ID=C.BRANCH\_ID;

View created.

SQL> SELECT \* FROM BOOKS\_AVAILABLE;

BOOK\_ID TITLE NO\_OF\_COPIES

**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 Bangalor’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).

1. Create a view that finds the salesman who has the customer with the highest order of a day.
2. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also

be deleted.

1. Create an index on (Customer (id)) to demonstrate the usage.

### Table Creation:

**SALESMAN**

CREATE TABLE SALESMAN (

SALESMAN\_ID INT(5) PRIMARY KEY,

NAME VARCHAR(10) NOT NULL,

CITY VARCHAR(15) NOT NULL,

COMMISSION INT(5));

Table created.

### CUSTOMER

CREATE TABLE CUSTOMER (

CUSTOMER\_ID INT(5) PRIMARY KEY,

CUST\_NAME VARCHAR(10) NOT NULL,

CITY VARCHAR(10) NOT NULL,

GRADE INT(5) NOT NULL,

SALESMAN\_ID INT(5),

FOREIGN KEY (SALESMAN \_ID) REFERENCES

SALESMAN(SALESMAN\_ID) ON DELETE SET NULL);

Table created.

### ORDERS

CREATE TABLE ORDERS (

ORD\_NO INT(5) PRIMARY KEY,

PURCHASE\_AMT INTEGER NOT NULL,

ORD\_DATE DATE NOT NULL,

CUSTOMER\_ID INT(5) ,

SALESMAN\_ID INT(5) ,

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID),

FOREIGN KEY (SALESMAN\_ID) REFERENCES SALESMAN(SALESMAN\_ID) ON DELETE CASCADE);

Table created.

### Values for tables

INSERT INTO SALESMAN VALUES (1000,'RAJ','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','MYSORE',60);

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

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

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

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

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

INSERT INTO ORDERS VALUES (1,200000,'12-04-16',11,1000);

INSERT INTO ORDERS VALUES (2,300000,'12-04-16',11,2000);

INSERT INTO ORDERS VALUES (3,400000,'15-04-17',22,1000);

**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 |  | 12-04-16 |  | 11 |  | 1000 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 |  | 300000 |  | 12-04-16 |  | 11 |  | 2000 |
| 3 |  | 400000 |  | 15-APR-17 |  | 22 |  | 1000 |

Queries:

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 **LIKE** '%BENGALURU');

COUNT(CUSTOMER\_ID)

3

1. 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

1. 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 LAVANYARAJ

ROHIT

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='12-04-16');

View created.

**SELECT** \* **FROM** SALES\_HIGHERODER; 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 12-04-16 11 2000

**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 2020 (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.

### Table Creation:

**ACTOR**

CREATE TABLE ACTOR (

ACT\_ID INT PRIMARY KEY,

ACT\_NAME VARCHAR(20) NOT NULL,

ACT\_GENDER CHAR(1) NOT NULL);

Table created.

### DIRECTOR

CREATE TABLE DIRECTOR(

DIR\_ID INT PRIMARY KEY,

DIR\_NAME VARCHAR(20) NOT NULL,

DIR\_PHONE INT NOT NULL);

Table created.

### MOVIES

CREATE TABLE MOVIES(

MOV\_ID INT PRIMARY KEY,

MOV\_TITLE VARCHAR(25) NOT NULL,

MOV\_YEAR INT NOT NULL,

MOV\_LANG VARCHAR(15) NOT NULL,

DIR\_ID INT NOT NULL,

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

Table created.

### MOVIE\_CAST

CREATE TABLE MOVIE\_CAST(

ACT\_ID INT NOT NULL,

MOV\_ID INT NOT NULL,

ROLE VARCHAR(10) NOT NULL,

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

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

Table created.

### RATING

CREATE TABLE RATING(

MOV\_ID INT,

REV\_STARS VARCHAR(25) NOT NULL,

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

Table created.

### Description of Schema:

SQL> DESC ACTOR;

Name Null Type

ACT\_ID NOT NULL INT

ACT\_NAME NOT NULL VARCHAR(20)

ACT\_GENDER NOT NULL CHAR(1)

SQL> DESC DIRECTOR;

Name Null Type

DIR\_ID NOT NULL INT

DIR\_NAME NOT NULL VARCHAR(20)

DIR\_PHONE NOT NULL INT

SQL> DESC MOVIES;

Name Null Type

MOV\_ID NOT NULL INT

MOV\_TITLE NOT NULL VARCHAR (25)

MOV\_YEAR NOT NULL INT

MOV\_LANG NOT NULL VARCHAR (15)

DIR\_ID NOT NULL INT

SQL> DESC RATING;

Name Null Type

MOV\_ID NOT NULL INT

REV\_STARS NOT NULL VARCHAR(25)

### Values for tables:

SQL> INSERT INTO ACTOR

VALUES(111,'DEEPA','F'),

(222,'SUDEEP','M'),

(333,'PUNEETH','M'),

(444,'DHIGANTH','M'),

(555,'RANI','F');

**SELECT \* FROM ACTOR;**

| ACT ID | ACT\_NAME | ACT\_GENDER |
| --- | --- | --- |
| 111 | DEEPA | F |
| 222 | SUDEEP | M |
| 333 | PUNEETH | M |
| 444 | DHIGANTH | M |
| 555 | RANI | F |

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 DIRECTOR;

| DIR\_ID |  | DIR\_NAME |  | DIR\_PHONE |
| --- | --- | --- | --- | --- |
| 101 |  | HITCHCOCK |  | 112267809 |
| 102 |  | RAJ MOULI |  | 152358709 |
| 103 |  | YOGARAJ |  | 272337808 |
| 104 |  | STEVEN SPIELBERG |  | 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 |  | 2016 |  | KANNADA |  | 103 |
| 5555 MANASARE |  | 1999 |  | 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 |

Queries

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

SELECT MOV\_TITLE FROM MOVIES

WHERE DIR\_ID = (SELECT DIR\_ID

FROM DIRECTOR

WHERE DIR\_NAME='HITCHCOCK');

MOV\_TITLE 

NOTORIOUS

REAR WINDOW

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 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;

MOV\_TITLE

MALE

MANASARE

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

(SELECT A.ACT\_NAME

FROM ACTOR A JOIN MOVIE\_CAST M ON A.ACT\_ID=M.ACT\_ID

JOIN MOVIES M1 ON M.MOV\_ID=M1.MOV\_ID

WHERE M1.MOV\_YEAR<2000 )

INTERSECT

(SELECT A.ACT\_NAME

FROM ACTOR A JOIN MOVIE\_CAST M ON A.ACT\_ID=M.ACT\_ID

JOIN MOVIES M1 ON M.MOV\_ID=M1.MOV\_ID

WHERE M1.MOV\_YEAR>2015 );

### ACT\_NAME

DHIGANTH

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,MAX(REV\_STARS)

FROM MOVIES

INNER JOIN RATING USING (MOV\_ID)

GROUP BY MOV\_TITLE

HAVING MAX(REV\_STARS)>0

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

WHERE DIR\_ID IN (SELECT DIR\_ID

FROM DIRECTOR

WHERE DIR\_NAME='STEVEN SPIELBERG'));

| SELECT  **MOV\_ID** |  | \* FROM RATING  **REV\_STARS** |
| --- | --- | --- |
| 1111 |  | 5 |
| 2222 |  | 4 |
| 3333 |  | 3 |
| 5555 |  | 4 |
| 4444 |  | 5 |

**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 fifth semester ‘B’ section.
2. Compute the total number of male and female students in each semester and in each section.
3. Create a view of Event 1 marks of student USN ‘01JST IS ’ in all subjects.
4. Calculate the Final IA (average of best two test marks) and update the corresponding table for all students.
5. Categorize students based on the following criterion:

If Final IA = 17 to 20 then CAT =‘Outstanding’

If Final IA = 12 to 16 then CAT = ‘Average’

If Final IA< 12 then CAT = ‘Weak’

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

### Table Creation:

**STUDENT**

CREATE TABLE STUDENT

(USN VARCHAR(10) PRIMARY KEY,

SNAME VARCHAR(25),

ADDRESS VARCHAR(25),

PHONE VARCHAR(10),

GENDER CHAR(1));

Table created.

**SEMSEC**

CREATE TABLE SEMSEC(

SSID VARCHAR(5) PRIMARY KEY,

SEM INT NOT NULL,

SEC VARCHAR(8) NOT NULL);

Table created.

**CLASS**

CREATE TABLE CLASS (

USN VARCHAR(10),

SSID VARCHAR(5), FOREIGN KEY(USN) REFERENCES STUDENT(USN), FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID));

Table created.

**SUBJECT**

CREATE TABLE SUBJECT(

SUBCODE VARCHAR(8) PRIMARY KEY,

TITLE VARCHAR(20),

SEM INT NOT NULL,

CREDITS INT );

Table created.

**IAMARKS**

CREATE TABLE IAMARKS (

USN VARCHAR(10),

SUBCODE VARCHAR(8),

SSID VARCHAR(5),

TEST1 INT,

TEST2 INT,

TEST3 INT,

FINALIA INT,FOREIGN KEY(USN)REFERENCES STUDENT(USN), FOREIGN KEY(SUBCODE) REFERENCES SUBJECT(SUBCODE), FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID));

Table created.

Values for tables:

**STUDENT:**

INSERTION TO BE DONE

SELECT \* FROM STUDENT;

| **USN SNAME** | **ADDRESS** | **PHONE** | **G** |
| --- | --- | --- | --- |
| 1cg15cs001 Abhi | tumkur | 9875698410 | M |
| 1cg15cs002 amulya | gubbi | 8896557412 | F |
| 1cg16me063 chethan | nittur | 7894759522 | M |
| 1cg14ec055 raghavi | sspuram | 9485675521 | F |
| 1cg15ee065 sanjay | bangalore | 9538444404 | M |
| **SEMSEC:** |  |  |  |
| INSERTION TO BE DONE |  |  |  |
| SELECT \* FROM SEMSEC; |  |  |  |
| **SSID SEM S**  **-** |  |  |  |
| 5A 5 A  3B 3 B |  |  |  |
| 7A 7 A |  |  |  |
| 2C 2 C  4B 4 B |  |  |  |
| 4C 4 C |  |  |  |
| **CLASS:** |  |  |  |
| INSERTION TO BE DONE. |  |  |  |
| SELECT \* FROM CLASS; |  |  |  |
| USN SSID |  |  |  |

1cg15cs001 5A

1cg15cs002 5A

1cg16me063 3B

1cg14ec055 7A

1cg15ee065 3B

1cg15ee065 4c

1cg15cs002 4c

| **SUBJECT:**  INSERTION TO BE DONE | |  | |  | |
| --- | --- | --- | --- | --- | --- |
| SELECT \* FROM SUBJECT; |  | |  | |
| **SUBCODE TITLE** | **SEM** | | **CREDITS** | |
| 15cs53 dbms | 5 | | 4 | |
| 15cs33 ds | 3 | | 4 | |
| 15cs34 co | 3 | | 4 | |
| 15csl58 dba | 52 | |  | |
| 10cs71 oomd | 7 | | 4 | |

**IAMARKS:**

| INSERTION TO BE DONE |  |  |
| --- | --- | --- |

SELECT \* FROM IAMARKS;

| USN |  | SUBCODE |  | SSID |  | TEST1 | TEST2 | TEST3 | FINALIA |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1cg15cs001 |  | 15cs53 |  | 5A |  | 18 | 19 | 15 | 19 |
| 1cg15cs002 |  | 15cs53 |  | 5A |  | 15 | 16 | 14 | 16 |
| 1cg16me063 |  | 15cs33 |  | 3B |  | 10 | 15 | 16 | 16 |
| 1cg14ec055 |  | 10cs71 |  | 7A |  | 18 | 20 | 21 | 21 |
| 1cg15ee065 |  | 15cs33 |  | 3B |  | 16 | 20 | 17 | 19 |
| 1cg15ee065 |  | 15cs53 |  | 4c |  | 19 | 20 | 18 | 20 |
| **Queries:** |  |  |  |  |  |  |  |  |  |

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

SELECT S.\*,SS.SEM,SS.SEC

-> FROM STUDENT S,SEMSEC SS,CLASS C WHERE S.USN=C.USN

-> AND SS.SSID=C.SSID

-> AND SS.SEM=4 AND SS.SEC='C';

USN SNAME ADDRESS PHONE G

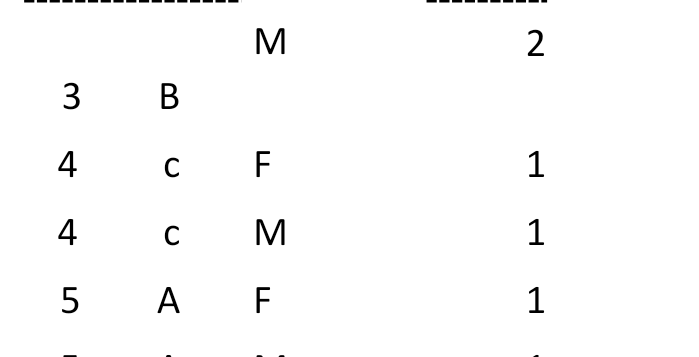


1cg15ee065 Sanjay bangalore 9538444404 M

1cg15cs002 Amulya gubbi 8896557412 F

1. Compute the total number of male and female students in each semester and in each section. mysql> SELECT SS.SEM,SS.SEC,S.GENDER,COUNT(S.GENDER) AS COUNT FROM STUDENT S,SEMSEC SS,CLASS C WHERE S.USN=C.USN

-> AND SS.SSID=C.SSID GROUP BY SS.SEM,SS.SEC,S.GENDER ORDER BY SEM;

SEM S G COUNT(\*) 

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

mysql> CREATE VIEW STUDENT\_TEST1\_MARKS\_V

-> AS SELECT TEST1, SUBCODE

-> FROM IAMARKS

-> WHERE USN='1BI15CS101';

Query OK, 0 rows affected (0.02 sec)

mysql> SELECT \* FROM STUDENT\_TEST1\_MARKS\_V;

SUBCODE TEST1

15cs33 16

15cs53 19

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

mysql> DELIMITER //

mysql> CREATE PROCEDURE AVG\_MARKS()

-> BEGIN

-> DECLARE C\_A INTEGER;

-> DECLARE C\_B INTEGER;

-> DECLARE C\_C INTEGER;

-> DECLARE C\_SUM INTEGER;

-> DECLARE C\_AVG INTEGER;

-> DECLARE C\_USN VARCHAR(10);

-> DECLARE C\_SUBCODE VARCHAR(8);

-> DECLARE C\_SSID VARCHAR(5);

->

-> DECLARE C\_IAMARKS CURSOR FOR

-> SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B, GREATEST(TEST3,TEST2) AS C, USN, SUBCODE, SSID

-> FROM IAMARKS

-> WHERE FINALIA IS NULL

-> FOR UPDATE;

->

-> OPEN C\_IAMARKS;

-> LOOP

->

-> FETCH C\_IAMARKS INTO C\_A, C\_B, C\_C, C\_USN, C\_SUBCODE, C\_SSID;

->

-> IF (C\_A != C\_B) THEN

-> SET C\_SUM=C\_A+C\_B;

-> ELSE

-> SET C\_SUM=C\_A+C\_C;

-> END IF;

->

-> SET C\_AVG=C\_SUM/2;

->

-> UPDATE IAMARKS SET FINALIA = C\_AVG

-> WHERE USN = C\_USN AND SUBCODE = C\_SUBCODE AND SSID = C\_SSID;

->

-> END LOOP;

-> CLOSE C\_IAMARKS;

-> END;

-> //

Query OK, 0 rows affected (0.01 sec)

**CALL AVG\_MARKS ();**

**SELECT \* FROM IAMARKS;**

**USN SUBCODE SSID TEST1 TEST2 TEST3 FINALIA**

| 1cg15cs001 15cs53 | 5A 18 |  | 19 |  | 15 | 19 |
| --- | --- | --- | --- | --- | --- | --- |
| 1cg15cs002 15cs53 | 5A 15 |  | 16 |  | 14 | 16 |
| 1cg16me063 15cs33 | 3B 10 |  | 15 |  | 16 | 16 |
| 1cg14ec055 10cs71 | 7A 18 |  | 20 |  | 21 | 21 |
| 1cg15ee065 15cs33 | 3B 16 |  | 20 |  | 17 | 19 |
| 1cg15ee065 15cs53 | 4c 19 |  | 20 |  | 18 | 20 |

DELIMITER;

1. 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.

mysql> SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER, IA.SUBCODE,

-> (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 = 8;

**USN SNAME ADDRESS PHONE G CAT**

**-** 

1cg14ec055 raghavi sspuram 9485675521 F WEAK

1. **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(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.

### Table Creation:

**DEPARTMENT**

| CREATE TABLE DEPARTMENT |
| --- |
|  | (DNO VARCHAR (20) PRIMARY KEY, |
|  | DNAME VARCHAR (20), |
|  | MGR\_SSN VARCHAR (20), |
|  | MGR\_START\_DATE DATE); |
|  |  |
|  | DESC DEPARTMENT; |

**EMPLOYEE**

| CREATE TABLE EMPLOYEE |
| --- |
|  | (SSN VARCHAR(20) PRIMARY KEY, |
|  | NAME VARCHAR(20), |
|  | ADDRESS VARCHAR(20), |
|  | SEX CHAR(1), |
|  | SALARY INTEGER, |
|  | SUPERSSN VARCHAR(20), |
|  | DNO VARCHAR(20), |
|  | FOREIGN KEY (SUPERSSN) REFERENCES EMPLOYEE (SSN), |
|  | FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNO)); |
|  |  |
|  | DESC EMPLOYEE; |

|  |
| --- |
| ALTER TABLE DEPARTMENT |
|  | ADD FOREIGN KEY (MGR\_SSN) REFERENCES EMPLOYEE(SSN); |
|  |  |

Table altered.

**DLOCATION**

| CREATE TABLE DLOCATION |
| --- |
|  | (DLOC VARCHAR(20), |
|  | DNO VARCHAR(20), |
|  | FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO), |
|  | PRIMARY KEY (DNO, DLOC)); |
|  |  |
|  | DESC DLOCATION; |

**PROJECT**

| CREATE TABLE PROJECT |
| --- |
|  | (PNO INTEGER PRIMARY KEY, |
|  | PNAME VARCHAR(20), |
|  | PLOCATION VARCHAR(20), |
|  | DNO VARCHAR(20), |
|  | FOREIGN KEY (DNO) REFERENCES DEPARTMENT(DNO)); |
|  |  |
|  | DESC PROJECT; |

**WORKS\_ON**

|  |
| --- |
| CREATE TABLE WORKS\_ON |
|  | (HOURS INTEGER, |
|  | SSN VARCHAR(20), |
|  | PNO INTEGER, |
|  | FOREIGN KEY (SSN) REFERENCES EMPLOYEE(SSN), |
|  | FOREIGN KEY (PNO) REFERENCES PROJECT(PNO), |
|  | PRIMARY KEY (SSN, PNO)); |
|  |  |
|  | DESC WORKS\_ON; |
|  |  |

### Values for tables:

**DEPARTMENT**

INSERT INTO DEPARTMENT VALUES(‘DNO’, ‘DNAME’, ‘MGRSSN’ ,'MGRSTARTDATE');

SELECT \* FROM DEPARTMENT;

DNO DNAME MGRSSN MGRSTARTD

- -

1 RESEARCH 111111 10-AUG-12

2 ACCOUNTS 222222 10-AUG-10

3 AI 333333 15-APR-12

4 NETWORKS 111111 18-MAY-14

5 BIGDATA 666666 21-JAN-10

5 rows selected.

**EMPLOYEE**

INSERT INTO EMPLOYEE VALUES('SSN','NAME','ADDRESS','SEX',SALARY,

'SUPERSSN' ,‘ ‘ DNO’);

SELECT \* FROM EMPLOYEE;

| SSN | NAME | ADDRESS | SEX | SALARY SUPERSSN | DNO |
| --- | --- | --- | --- | --- | --- |
| 111111 | RAJ | BENGALURU | M | 700000 | 1 |
| 222222 | RASHMI | MYSORE | F | 400000 111111 | 2 |
| 333333 | RAGAVI | TUMKUR | F | 800000 | 3 |
| 444444 | RAJESH | TUMKUR | M | 650000 333333 | 3 |
| 555555 | RAVEESH | BENGALURU | M | 500000 333333 | 3 |
| 666666 | SCOTT | ENGLAND | M | 700000 444444 | 5 |
| 777777 | NIGANTH | GUBBI | M | 200000 222222 | 2 |
| 888888 | RAMYA | GUBBI | F | 400000 222222 | 3 |
| 999999 | VIDYA | TUMKUR | F | 650000 333333 | 3 |
| 100000 | GEETHA | TUMKUR | F | 800000 | 3 |
| 10 rows | selected. |  |  |  |  |

**DLOCATION**

INSERT INTO DLOCATION VALUES(‘DNO’,'DLOC');

SELECT \* FROM DLOCATION;

DNO DLOC 

1. MYSORE
2. TUMKUR
3. GUBBI
4. DELHI
5. BENGALURU

**PROJECT**

INSERT INTO PROJECT VALUES(PNO,'PNAME','PLOCATION',’DNO’);

SELECT \* FROM PROJECT;

PNO PNAME PLOCATION DNO

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 111 IOT GUBBI  222 TEXTSPEECH GUBBI |  | 3 3 |  |  |  |
| 333 IPSECURITY DELHI |  | 4 |  |  |  |
| 444 TRAFICANAL BENGALURU |  | 5 |  |  |  |
| 555 CLOUDSEC DELHI |  | 1 |  |  |  |

5 rows selected.

**WORKS\_ON**

INSERT INTO WORKS\_ON VALUES ('SSN’, PNO, HOURS);

| SELECT \* FROM  SSN | WORKS\_ON;  PNO |  | HOURS |
| --- | --- | --- | --- |
| 666666 | 333 |  | 4 |
| 666666 | 111 |  | 2 |
| 111111 | 222 |  | 3 |
| 555555 | 222 |  | 2 |
| 333333 | 111 |  | 4 |
| 444444 | 111 |  | 6 |
| 222222 | 111 |  | 2 |

8 rows selected.

Queries

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 P.PNO |
| --- |
|  | FROM PROJECT P, DEPARTMENT D, EMPLOYEE E |
|  | WHERE E.DNO=D.DNO |
|  | AND D.MGR\_SSN=E.SSN |
|  | AND E.NAME LIKE '%SCOTT' |
|  | UNION |
|  | SELECT DISTINCT P1.PNO |
|  | FROM PROJECT P1, WORKS\_ON W, EMPLOYEE E1 |
|  | WHERE P1.PNO=W.PNO |
|  | AND E1.SSN=W.SSN |
|  | AND E1.NAME LIKE '%SCOTT'; |

PNO

111

333

444

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

| SELECT E.NAME, 1.1\*E.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'; |

| SSN |  | NAME |  | ADDRESS | SEX |  | SALARY SUPERSSN |  | DNO |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 111111 |  | RAJ |  | BENGALURU | M |  | 700000 |  | 1 |
| 222222 |  | RASHMI |  | MYSORE | F |  | 440000 111111 |  | 2 |
| 333333 |  | RAGAVI |  | TUMKUR | F |  | 880000 |  | 3 |
| 444444 |  | RAJESH |  | TUMKUR | M |  | 715000 333333 |  | 3 |
| 555555 |  | RAVEESH |  | BENGALURU | M |  | 500000 333333 |  | 3 |
| 666666 |  | SCOTT |  | ENGLAND | M |  | 770000 444444 |  | 5 |
| 777777 |  | NIGANTH |  | GUBBI | M |  | 200000 222222 |  | 2 |
| 888888 |  | RAMYA |  | GUBBI | F |  | 400000 222222 |  | 3 |
| 999999 |  | VIDYA |  | TUMKUR | F |  | 650000 333333 |  | 3 |
| 100000 |  | GEETHA |  | TUMKUR | F |  | 800000 |  | 3 |

10 rows selected.

1. 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(E.SALARY), MAX(E.SALARY), MIN(E.SALARY), AVG(E.SALARY) |
| --- |
|  | FROM EMPLOYEE E, DEPARTMENT D |
|  | WHERE E.DNO=D.DNO |
|  | AND D.DNAME='ACCOUNTS'; |



440000 200000 320000

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

|  |
| --- |
| SELECT E.NAME |
|  | FROM EMPLOYEE E |
|  | WHERE NOT EXISTS(SELECT PNO FROM PROJECT WHERE DNO='5' AND PNO NOT IN (SELECT |
|  | PNO FROM WORKS\_ON |
|  | WHERE E.SSN=SSN)); |
|  |  |

NAME

SCOTT

1. 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 D.DNO, COUNT(\*) |
| --- |
|  | FROM DEPARTMENT D, EMPLOYEE E |
|  | WHERE D.DNO=E.DNO |
|  | AND E.SALARY > 600000 |
|  | AND D.DNO IN (SELECT E1.DNO |
|  | FROM EMPLOYEE E1 |
|  | GROUP BY E1.DNO |
|  | HAVING COUNT(\*)>5) |
|  | GROUP BY D.DNO; |

DNO COUNT(SSN)

3 4

1. **Consider the schema of the call detail table to partitioned primary index:**

CREATE TABLE calldetail (phone\_number DECIMAL(10) NOT NULL, call\_start TIMESTAMP, call\_duration INTEGER, call\_description VARCHAR(30)) PRIMARY INDEX (phone\_number, call\_start);

Demonstrate the query against this table be optimized by partitioning its primary index using partitioning techniques.