**UNIVERSITY OF VISVESVARYA COLLEGE OF ENGINEERING**

*K R Circle – 560 001*

**DATABASE MANAGEMENT SYSTEM MANUAL**

|  |  |  |
| --- | --- | --- |
| **SL no.** | **Databse** | **Page no.** |
| **1** | **Library Database** | **2-5** |
| **2** | **Sales Order Database** | **6-9** |
| **3** | **Movie Database** | **10-13** |
| **4** | **College Database** | **14-18** |
| **5** | **Company Database** | **19-23** |

**Submitted by:**

Arshan A Shaik – U25UV22T006005

Kushal sathwik– U25UV22T0060

**Signature:**

**Dr. Samyama Gunjal G H**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1) LIBRARY DATABASE**

## DDL (Data Defination Language)

CREATE TABLE BOOK (

Book\_id INT,

Title VARCHAR(200),

Publisher\_Name VARCHAR(100),

Pub\_Year YEAR

);

CREATE TABLE BOOK\_AUTHORS (

Book\_id INT,

Author\_Name VARCHAR(100)

);

CREATE TABLE PUBLISHER (

Pub\_id INT,

Name VARCHAR(100),

Address VARCHAR(200),

Phone VARCHAR(15)

);

CREATE TABLE BOOK\_COPIES (

Book\_id INT,

Branch\_id INT,

No\_of\_Copies INT

);

CREATE TABLE BOOK\_LENDING (

Book\_id INT,

Branch\_id INT,

Card\_No INT,

Date\_Out DATE,

Due\_Date DATE

);

CREATE TABLE LIBRARY\_BRANCH (

Branch\_id INT,

Branch\_Name VARCHAR(100),

Address VARCHAR(200)

);

lter table book

add constraint book\_keysff primary key(book\_id);

ALTER TABLE BOOK\_AUTHORS

ADD CONSTRAINT PK\_BOOK\_AUTHORS PRIMARY KEY (Book\_id, Author\_Name);

ALTER TABLE PUBLISHER

ADD CONSTRAINT PK\_PUBLISHER PRIMARY KEY (Pub\_id);

ALTER TABLE BOOK\_COPIES

ADD CONSTRAINT PK\_BOOK\_COPIES PRIMARY KEY (Book\_id, Branch\_id);

ALTER TABLE BOOK\_LENDING

ADD CONSTRAINT PK\_BOOK\_LENDING PRIMARY KEY (Book\_id, Branch\_id, Card\_No);

ALTER TABLE LIBRARY\_BRANCH

ADD CONSTRAINT PK\_LIBRARY\_BRANCH PRIMARY KEY (Branch\_id);

#add foregin key constrainst

alter table publisher

add constraint fk\_pubslishercss foreign key(publisher\_name) references publisher (name);

ALTER TABLE BOOK\_AUTHORS

ADD CONSTRAINT FK\_BOOK\_AUTHORS\_BOOK FOREIGN KEY (Book\_id) REFERENCES BOOK(Book\_id);

ALTER TABLE BOOK\_COPIES

ADD CONSTRAINT FK\_BOOK\_COPIES\_BOOK FOREIGN KEY (Book\_id) REFERENCES BOOK(Book\_id);

ALTER TABLE BOOK\_COPIES

ADD CONSTRAINT FK\_BOOK\_COPIES\_BRANCH FOREIGN KEY (Branch\_id) REFERENCES LIBRARY\_BRANCH(Branch\_id);

ALTER TABLE BOOK\_LENDING

ADD CONSTRAINT FK\_BOOK\_LENDING\_BOOK FOREIGN KEY (Book\_id) REFERENCES BOOK(Book\_id);

ALTER TABLE BOOK\_LENDING

ADD CONSTRAINT FK\_BOOK\_LENDING\_BRANCH FOREIGN KEY (Branch\_id) REFERENCES LIBRARY\_BRANCH(Branch\_id);

## Data Manipulation Language (Inserting tuples)

INSERT INTO BOOK VALUES

(1, 'Database Systems', 'Pearson', 2018),

(2, 'Operating Systems', 'McGraw Hill', 2019),

(3, 'Computer Networks', 'Pearson', 2020),

(4, 'Data Structures', 'O Reilly', 2017),

(5, 'Machine Learning', 'Springer', 2021),

(8, 'Machine Learning', 'Springer', 2021);

INSERT INTO PUBLISHER VALUES

(101, 'Pearson', 'New York', '1234567890'),

(102, 'McGraw Hill', 'Chicago', '2345678901'),

(103, 'O Reilly', 'San Francisco', '3456789012'),

(104, 'Springer', 'Berlin', '4567890123'),

(105, 'Elsevier', 'London', '5678901234');

INSERT INTO BOOK\_AUTHORS VALUES

(1, 'Korth'),

(2, 'Silberschatz'),

(3, 'Tanenbaum'),

(4, 'Mark Allen Weiss'),

(5, 'Tom Mitchell');

INSERT INTO LIBRARY\_BRANCH VALUES

(10, 'Central Library', 'Main Street'),

(11, 'Science Block', 'North Wing'),

(12, 'Engineering Block', 'East Wing'),

(13, 'IT Block', 'South Wing'),

(14, 'Management Block', 'West Wing');

INSERT INTO BOOK\_COPIES VALUES

(1, 10, 5),

(2, 11, 3),

(3, 12, 4),

(4, 13, 2),

(5, 14, 6);

truncate book\_lending;

TRUNCATE TABLE BOOK\_LENDING;

INSERT INTO BOOK\_LENDING VALUES

(1, 10, 1001, '2025-01-10', '2025-01-24'),

(2, 11, 1001, '2025-02-15', '2025-03-01'),

(3, 12, 1001, '2025-03-01', '2025-03-15'),

(4, 13, 1001, '2025-04-05', '2025-04-19'),

(5, 11, 1002, '2025-01-12', '2025-01-26'),

(2, 11, 1002, '2025-02-18', '2025-03-04'),

(3, 12, 1002, '2025-03-10', '2025-03-24'),

(4, 13, 1002, '2025-04-12', '2025-04-26'),

(5, 14, 1002, '2025-05-15', '2025-05-29'),

(3, 12, 1003, '2025-03-05', '2025-03-19'),

(4, 13, 1004, '2025-03-07', '2025-03-21'),

(5, 14, 1005, '2025-03-09', '2025-03-23');

## QUIERES

1. **Retrieve details of the books in the library: id, title, name of the publisher, authors, number of**

**copies in each branch, etc.**

SELECT B.Book\_id,

B.Title,

B.Publisher\_Name,

A.Author\_Name,

C.Branch\_id,

C.No\_of\_Copies

FROM BOOK B

JOIN BOOK\_AUTHORS A ON B.Book\_id = A.Book\_id

JOIN BOOK\_COPIES C ON B.Book\_id = C.Book\_id;

select \* from book\_lending;



1. **Get the particulars of the borrowers who have borrowed more than 3 books from Jan 2019 to Jun**

**2019**

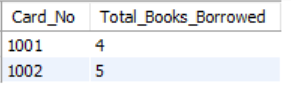
SELECT Card\_No, COUNT(\*) AS Total\_Books\_Borrowed

FROM BOOK\_LENDING

WHERE Date\_Out BETWEEN '2025-01-01' AND '2025-06-30'

GROUP BY Card\_No

HAVING COUNT(\*) > 3;



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

#step1: enable cascade while declaring foreign key This way, deleting a book from

# BOOK will automatically delete related records from other tables

ALTER TABLE BOOK\_AUTHORS

ADD CONSTRAINT FK\_BOOK\_AUTHORS\_BOOK FOREIGN KEY (Book\_id) REFERENCES BOOK(Book\_id) ON DELETE CASCADE;

ALTER TABLE BOOK\_COPIES

ADD CONSTRAINT FK\_BOOK\_COPIES\_BOOK FOREIGN KEY (Book\_id) REFERENCES BOOK(Book\_id) ON DELETE CASCADE;

ALTER TABLE BOOK\_LENDING

ADD CONSTRAINT FK\_BOOK\_LENDING\_BOOK FOREIGN KEY (Book\_id) REFERENCES BOOK(Book\_id) ON DELETE CASCADE;

DELETE FROM BOOK

WHERE

Book\_id = 3;

SELECT

\*

FROM

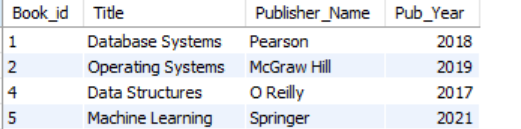
book

WHERE

book\_id = 3;

select \* from book where book\_id = 3;

Book table:



Book Authors table:

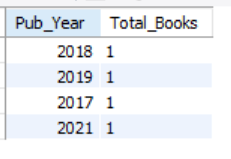


1. **Partition the BOOK table based on the year of publication. Demonstrate it’s working with a simple query**

SELECT Pub\_Year, COUNT(\*) AS Total\_Books

FROM BOOK

GROUP BY Pub\_Year;GROUP BY Pub\_Year;



1. **Create a view of all books and it’s number of copies that are currently available in the Library**

CREATE VIEW Book\_Availability AS

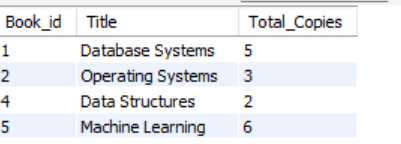
SELECT B.Book\_id, B.Title, SUM(BC.No\_of\_Copies) AS Total\_Copies

FROM BOOK B

JOIN BOOK\_COPIES BC ON B.Book\_id = BC.Book\_id

GROUP BY B.Book\_id, B.Title;

select \* from book\_Availability;



## 2) SALES ORDER DATABASE

### CREATION OF TABLES (DDL)

CREATE TABLE SALESMAN (

Salesman\_id INT PRIMARY KEY,

Name VARCHAR(50),

City VARCHAR(50),

Commission FLOAT

);

CREATE TABLE CUSTOMER (

Customer\_id INT PRIMARY KEY,

Cust\_Name VARCHAR(50),

City VARCHAR(50),

Grade INT,

Salesman\_id INT,

FOREIGN KEY (Salesman\_id) REFERENCES SALESMAN(Salesman\_id)

);

CREATE TABLE ORDERS (

Ord\_No INT PRIMARY KEY,

Purchase\_Amt FLOAT,

Ord\_Date DATE,

Customer\_id INT,

Salesman\_id INT,

FOREIGN KEY (Customer\_id) REFERENCES CUSTOMER(Customer\_id),

FOREIGN KEY (Salesman\_id) REFERENCES SALESMAN(Salesman\_id)

);

### INSERTING INTO TABLES (DML)

INSERT INTO SALESMAN VALUES

(1000, 'John', 'Bangalore', 0.15),

(1001, 'Alice', 'Delhi', 0.12),

(1002, 'Bob', 'Bangalore', 0.10),

(1003, 'Charlie', 'Mumbai', 0.18),

(1004, 'David', 'Kolkata', 0.20);

-- INSERT INTO CUSTOMER

INSERT INTO CUSTOMER VALUES

(2000, 'Amit', 'Bangalore', 200, 1000),

(2001, 'Sita', 'Bangalore', 250, 1000),

(2002, 'Ravi', 'Delhi', 150, 1001),

(2003, 'Meera', 'Delhi', 300, 1001),

(2004, 'Ramesh', 'Mumbai', 180, 1003),

(2005, 'Sunita', 'Kolkata', 210, 1004);

-- INSERT INTO ORDERS

INSERT INTO ORDERS VALUES

(3000, 5000, '2024-03-01', 2000, 1000),

(3001, 7000, '2024-03-01', 2001, 1000),

(3002, 3000, '2024-03-02', 2002, 1001),

(3003, 8000, '2024-03-02', 2003, 1001),

(3004, 4500, '2024-03-03', 2004, 1003),

(3005, 6000, '2024-03-03', 2005, 1004);

### QUERIES

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

SELECT COUNT(\*) AS count\_above\_bangalore\_avg

FROM CUSTOMER

WHERE GRADE > (SELECT AVG(GRADE)

FROM CUSTOMER WHERE CITY = 'BANGALORE');



1. **Find the names and numbers of all salesman who had more than one customer**

SELECT Name

FROM SALESMAN

WHERE Salesman\_id IN (

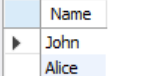
SELECT Salesman\_id

FROM CUSTOMER

GROUP BY Salesman\_id

HAVING COUNT(\*) > 1

);



1. **List all the salesman and indicate those who have and don’t have customers in their cities (use UNION operation)**

SELECT S.Salesman\_id, S.Name, 'Has Customer' AS Status

FROM SALESMAN S

WHERE S.Salesman\_id IN (

SELECT C.Salesman\_id

FROM CUSTOMER C

WHERE C.City = (SELECT City FROM SALESMAN WHERE Salesman\_id = C.Salesman\_id)

)

UNION

SELECT S.Salesman\_id, S.Name, 'No Customer' AS Status

FROM SALESMAN S

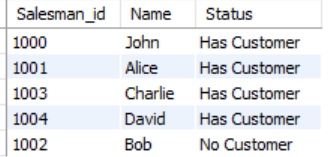
WHERE S.Salesman\_id NOT IN (

SELECT C.Salesman\_id

FROM CUSTOMER C

WHERE C.City = (SELECT City FROM SALESMAN WHERE Salesman\_id = C.Salesman\_id)

);



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

CREATE VIEW TopSalesman AS

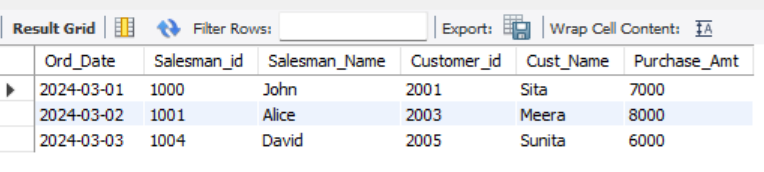
SELECT O.Ord\_Date, O.Purchase\_Amt, S.Name AS Salesman\_Name, C.Cust\_Name AS Customer\_Name

FROM ORDERS O

JOIN CUSTOMER C ON O.Customer\_id = C.Customer\_id

JOIN SALESMAN S ON O.Salesman\_id = S.Salesman\_id

WHERE O.Purchase\_Amt = (SELECT MAX(Purchase\_Amt) FROM ORDERS WHERE Ord\_Date = O.Ord\_Date);



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

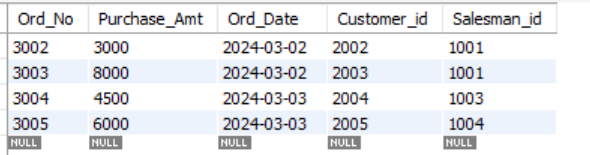
**deleted.**

DELETE FROM ORDERS

WHERE Salesman\_id = 1000;

DELETE FROM SALESMAN

WHERE Salesman\_id = 1000;



## 3) MOVIE DATABASE

### Creation of tables

CREATE TABLE ACTOR ( Act\_id INT PRIMARY KEY, Act\_Name VARCHAR(50), Act\_Gender VARCHAR(10) );

CREATE TABLE DIRECTOR ( Dir\_id INT PRIMARY KEY, Dir\_Name VARCHAR(50), Dir\_Phone VARCHAR(15) );

CREATE TABLE MOVIES ( Mov\_id INT PRIMARY KEY, Mov\_Title VARCHAR(100), Mov\_Year INT, Mov\_Lang

VARCHAR(20), Dir\_id INT REFERENCES DIRECTOR(Dir\_id) ON DELETE CASCADE);

CREATE TABLE MOVIE\_CAST ( Act\_id INT REFERENCES ACTOR(Act\_id) ON DELETE CASCADE, Mov\_id

INT REFERENCES MOVIES(Mov\_id) ON DELETE CASCADE, Role VARCHAR(50), PRIMARY KEY (Act\_id,

Mov\_id) );

CREATE TABLE RATING ( Mov\_id INT REFERENCES MOVIES(Mov\_id) ON DELETE CASCADE, Rev\_Stars

INT );

### Insertion of Tuples

INSERT INTO ACTOR VALUES

(1, 'Robert Downey Jr.', 'Male'),

(2, 'Scarlett Johansson', 'Female'),

(3, 'Chris Hemsworth', 'Male'),

(4, 'Leonardo DiCaprio', 'Male'),

(5, 'Emma Watson', 'Female'), (6, 'Tom Hanks', 'Male'),

(7, 'Angelina Jolie', 'Female'),

(8, 'Brad Pitt', 'Male');

INSERT INTO DIRECTOR VALUES

(1, 'ABCD', '1234567890'),

(2, 'XYZ', '9876543210'),

(3, 'Christopher Nolan', '5555555555'),

(4, 'Steven Spielberg', '6666666666'),

(5, 'James Cameron', '7777777777'),

(6, 'Martin Scorsese', '8888888888'),

(7, 'Quentin Tarantino', '9999999999'),

(8, 'Tim Burton', '4444444444'),

(10, 'XYZ', '1234567890');

INSERT INTO MOVIES VALUES

(1, 'Inception', 2010, 'English', 1),

(2, 'Avengers', 2012, 'English', 1),

(3, 'Titanic', 1997, 'English', 2),

(4, 'Interstellar', 2014, 'English', 3),

(5, 'Joker', 2019, 'English', 4),

(6, 'Fight Club', 1999, 'English', 5),

(7, 'Pulp Fiction', 1994, 'English', 6),

(8, 'The Dark Knight', 2008, 'English', 7),

(9, 'Movie XYZ1', 2020, 'English', 10),

(10, 'Movie XYZ2', 2021, 'English', 10);

INSERT INTO MOVIE\_CAST VALUES

(1, 1, 'Lead Role'),

(1, 2, 'Supporting'),

(2, 1, 'Supporting'), (2, 2, 'Heroine'),

(2, 3, 'Guest Role'),

(3, 2, 'Hero'),

(3, 4, 'Lead Role'),

(4, 2, 'Villain'),

(4, 5, 'Supporting'),

(4, 6, 'Villain'),

(5, 3, 'Lead Role'),

(5, 7, 'Guest Role'),

(6, 4, 'Supporting'),

(7, 5, 'Guest Role'),

(8, 6, 'Supporting');

INSERT INTO RATING VALUES

(1, 5),

(2, 4),

(3, 5),

(4, 3),

(5, 4),

(6, 5),

(7, 4),

(8, 5),

(9, 3),

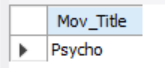
(10, 4);

### QUERIES

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

SELECT Mov\_Title FROM MOVIES

WHERE Dir\_id = (SELECT Dir\_id FROM DIRECTOR WHERE Dir\_Name = 'ABCD');



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

SELECT Mov\_Title

FROM MOVIES

WHERE Mov\_id IN (

SELECT Mov\_id

FROM MOVIE\_CAST

WHERE Act\_id IN (

SELECT Act\_id

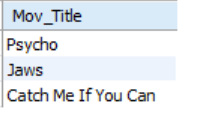
FROM MOVIE\_CAST

GROUP BY Act\_id

HAVING COUNT(DISTINCT Mov\_id) >= 2

)

);



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

**operation)**

SELECT Act\_Name

FROM ACTOR

WHERE Act\_id IN (

SELECT Act\_id

FROM MOVIE\_CAST MC

JOIN MOVIES M ON MC.Mov\_id = M.Mov\_id

WHERE M.Mov\_Year < 2000

)

AND Act\_id IN (

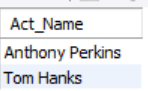
SELECT Act\_id

FROM MOVIE\_CAST MC

JOIN MOVIES M ON MC.Mov\_id = M.Mov\_id

WHERE M.Mov\_Year > 2015

);



1. **Find the titles of movies and number of stars for each movie that has at least one rating and**

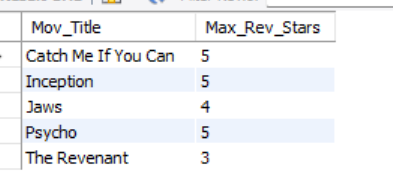
**find the highest number of stars that movie received. Sort the result my movie title.**

SELECT MOVIES.Mov\_Title, MAX(RATING.Rev\_Stars) AS Max\_Rev\_Stars

FROM MOVIES

JOIN RATING ON MOVIES.Mov\_id = RATING.Mov\_id

GROUP BY MOVIES.Mov\_Title

ORDER BY MOVIES.Mov\_Title;  


1. **Update rating of all movies directed by ‘XYZ’ to 5.**

UPDATE RATING

SET Rev\_Stars = 5

WHERE Mov\_id IN (

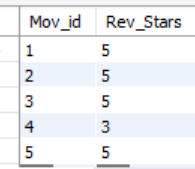
SELECT MOVIES.Mov\_id

FROM MOVIES

JOIN DIRECTOR ON MOVIES.Dir\_id = DIRECTOR.Dir\_id

WHERE DIRECTOR.Dir\_Name = 'Steven Spielberg'

);



## 4) COLLEGE DATABASE

**Creation of tables:**

CREATE TABLE STUDENT (

USN VARCHAR(20) PRIMARY KEY,

SName VARCHAR(100),

Address VARCHAR(255),

Phone VARCHAR(15),

Gender VARCHAR(10)

);

CREATE TABLE SEMSEC (

SSID INT PRIMARY KEY,

Sem INT,

Sec VARCHAR(1)

);

CREATE TABLE CLASS (

USN VARCHAR(20),

SSID INT,

PRIMARY KEY (USN, SSID),

FOREIGN KEY (USN) REFERENCES STUDENT(USN),

FOREIGN KEY (SSID) REFERENCES SEMSEC(SSID)

);

CREATE TABLE SUBJECT (

Subcode VARCHAR(10) PRIMARY KEY,

Title VARCHAR(100),

Sem INT,

Credits INT

);

CREATE TABLE IAMARKS (

USN VARCHAR(20),

Subcode VARCHAR(10),

SSID INT,

Test1 INT,

Test2 INT,

Test3 INT,

PRIMARY KEY (USN, Subcode, SSID),

FOREIGN KEY (USN) REFERENCES STUDENT(USN),

FOREIGN KEY (Subcode) REFERENCES SUBJECT(Subcode),

FOREIGN KEY (SSID) REFERENCES SEMSEC(SSID)

);

INSERTION INTO ROWS

INSERT INTO STUDENT VALUES

('1BI15CS101', 'Arun Kumar', 'Bangalore', '9876543210', 'Male'),

('1BI19CS112', 'Sunita Patel', 'Mysore', '9876543212', 'Female'),

('1BI19CS113', 'Vikram Singh', 'Delhi', '9876543213', 'Male'),

('1BI15CS116', 'Sameer Khan', 'Mumbai', '9876543216', 'Male'),

('1BI15CS123', 'Swati Gupta', 'Pune', '9876543223', 'Female');

-- Insert data into SEMSEC table

INSERT INTO SEMSEC VALUES

(1, 1, 'A'),

(12, 4, 'C'),

(22, 8, 'A'),

(23, 8, 'B'),

(24, 8, 'C');

-- Insert data into CLASS table

INSERT INTO CLASS VALUES

('1BI15CS101', 1),

('1BI19CS112', 12),

('1BI19CS113', 12),

('1BI15CS116', 22),

('1BI15CS123', 24);

-- Insert data into SUBJECT table

INSERT INTO SUBJECT VALUES

('CS101', 'Introduction to Programming', 1, 4),

('CS104', 'Database Management Systems', 4, 4),

('CS105', 'Operating Systems', 4, 4),

('CS110', 'Artificial Intelligence', 8, 4),

('CS111', 'Machine Learning', 8, 4);

-- Insert data into IAMARKS table

INSERT INTO IAMARKS VALUES

('1BI15CS101', 'CS101', 1, 18, 19, 17),

('1BI19CS112', 'CS104', 12, 18, 19, 15),

('1BI19CS113', 'CS105', 12, 15, 17, 19),

('1BI15CS116', 'CS110', 22, 19, 18, 20),

('1BI15CS123', 'CS111', 24, 14, 16, 15);

-- Add FinalIA column to IAMARKS table

ALTER TABLE IAMARKS ADD COLUMN FinalIA INT;

-- Update FinalIA values

UPDATE IAMARKS

SET FinalIA = (

CASE

WHEN Test1 <= Test2 AND Test1 <= Test3 THEN (Test2 + Test3) / 2

WHEN Test2 <= Test1 AND Test2 <= Test3 THEN (Test1 + Test3) / 2

WHEN Test3 <= Test1 AND Test3 <= Test2 THEN (Test1 + Test2) / 2

END

);

### QUERIES

1. **List all the student details studying in the fourth semester 'C' section:**

SELECT \*

FROM STUDENT

WHERE USN IN (

SELECT USN

FROM CLASS

WHERE SSID IN (

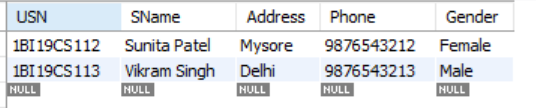
SELECT SSID

FROM SEMSEC

WHERE Sem = 4 AND Sec = 'C'

)

);



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

SELECT SS.Sem, SS.Sec, S.Gender, COUNT(\*) AS Count

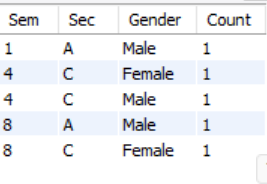
FROM STUDENT S

JOIN CLASS C ON S.USN = C.USN

JOIN SEMSEC SS ON C.SSID = SS.SSID

GROUP BY SS.Sem, SS.Sec, S.Gender

ORDER BY SS.Sem, SS.Sec;



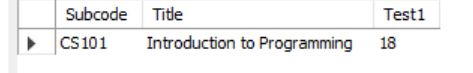
1. **Create a view of Test1 Marks of student USN '1XX1234' in all subjects:**

CREATE VIEW Student\_1BI15CS101\_Test1 AS

SELECT S.Subcode, S.Title, IA.Test1

FROM IAMARKS IA

JOIN SUBJECT S ON IA.Subcode = S.Subcode

WHERE IA.USN = '1BI15CS101';

**5. Categorize students based on the specified criteria for 8th semester A, B, and C sections:**

SELECT IA.USN, IA.Subcode, IA.FinalIA,

CASE

WHEN IA.FinalIA BETWEEN 17 AND 20 THEN 'Outstanding'

WHEN IA.FinalIA BETWEEN 12 AND 16 THEN 'Average'

WHEN IA.FinalIA < 12 THEN 'Weak'

END AS Category

FROM IAMARKS IA

WHERE IA.SSID IN (

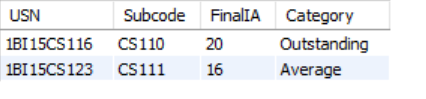
SELECT SSID

FROM SEMSEC

WHERE Sem = 8 AND Sec IN ('A', 'B', 'C')

)

ORDER BY IA.USN, IA.Subcode;

);

## 5) COMPANY DATABASE

**Creation of Tables:**

create database j\_db;

use j\_db;

CREATE TABLE EMPLOYEE (

SSN INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Address VARCHAR(255),

Sex CHAR(1),

Salary DECIMAL(10,2) CHECK (Salary > 0),

SuperSSN INT NULL,

DNo INT NULL

);

CREATE TABLE DEPARTMENT (

DNo INT PRIMARY KEY,

DName VARCHAR(100) UNIQUE NOT NULL,

MgrSSN INT UNIQUE NULL,

MgrStartDate DATE

);

-- Now add the foreign key constraints after both tables exist

ALTER TABLE EMPLOYEE ADD CONSTRAINT FK\_Emp\_Dept FOREIGN KEY (DNo) REFERENCES DEPARTMENT(DNo) ON DELETE CASCADE;

ALTER TABLE DEPARTMENT ADD CONSTRAINT FK\_Dept\_Mgr FOREIGN KEY (MgrSSN) REFERENCES EMPLOYEE(SSN) ON DELETE SET NULL;

-- Creating DLOCATION Table

CREATE TABLE DLOCATION (

DNo INT,

DLoc VARCHAR(100),

PRIMARY KEY (DNo, DLoc),

FOREIGN KEY (DNo) REFERENCES DEPARTMENT(DNo) ON DELETE CASCADE

);

-- Creating PROJECT Table

CREATE TABLE PROJECT (

PNo INT PRIMARY KEY,

PName VARCHAR(100) UNIQUE NOT NULL,

PLocation VARCHAR(100),

DNo INT,

FOREIGN KEY (DNo) REFERENCES DEPARTMENT(DNo) ON DELETE CASCADE

);

CREATE TABLE WORKS\_ON (

SSN INT,

PNo INT,

Hours DECIMAL(5,2) CHECK (Hours >= 0),

PRIMARY KEY (SSN, PNo),

FOREIGN KEY (SSN) REFERENCES EMPLOYEE(SSN) ON DELETE CASCADE,

FOREIGN KEY (PNo) REFERENCES PROJECT(PNo) ON DELETE CASCADE

);

-- Inserting Sample Data

INSERT INTO DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate) VALUES

(1, 'Accounts', NULL, '2020-01-01'),

(2, 'IT', NULL, '2021-06-15');

INSERT INTO EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo) VALUES

(101, 'John Scott', '123 Main St', 'M', 60000, NULL, 1),

(102, 'Alice Brown', '456 Oak St', 'F', 75000, NULL, 2),

(103, 'Michael Scott', '789 Pine St', 'M', 80000, NULL, 1);

UPDATE DEPARTMENT SET MgrSSN = 103 WHERE DNo = 1; -- Assigning Michael Scott as Manager of Accounts

INSERT INTO DLOCATION (DNo, DLoc) VALUES

(1, 'New York'),

(2, 'San Francisco');

INSERT INTO PROJECT (PNo, PName, PLocation, DNo) VALUES

(201, 'IoT', 'New York', 1),

(202, 'AI Research', 'San Francisco', 2);

INSERT INTO WORKS\_ON (SSN, PNo, Hours) VALUES

(101, 201, 20),

(102, 202, 25),

(103, 201, 30);

**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

WHERE P.DNo IN (

SELECT D.DNo

FROM DEPARTMENT D

JOIN EMPLOYEE E ON D.MgrSSN = E.SSN

WHERE E.Name LIKE '%Scott'

)

OR P.PNo IN (

SELECT W.PNo

FROM WORKS\_ON W

JOIN EMPLOYEE E ON W.SSN = E.SSN

WHERE E.Name LIKE '%Scott'

);



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

SELECT E.SSN, E.Name, E.Salary AS CurrentSalary,

E.Salary \* 1.10 AS NewSalary,

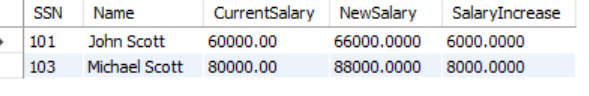
E.Salary \* 0.10 AS SalaryIncrease

FROM EMPLOYEE E

JOIN WORKS\_ON W ON E.SSN = W.SSN

JOIN PROJECT P ON W.PNo = P.PNo

WHERE P.PName = 'IoT';



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(Salary) AS TotalSalary,

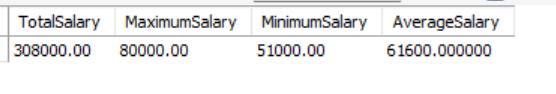
MAX(Salary) AS MaximumSalary,

MIN(Salary) AS MinimumSalary,

AVG(Salary) AS AverageSalary

FROM EMPLOYEE

WHERE DNo = (SELECT DNo FROM DEPARTMENT WHERE DName = 'Accounts');



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 P.PNo

FROM PROJECT P

WHERE P.DNo = 5 AND NOT EXISTS (

SELECT W.PNo

FROM WORKS\_ON W

WHERE W.PNo = P.PNo AND W.SSN = E.SSN

)

)



**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 D.DNo,

(SELECT COUNT(\*)

FROM EMPLOYEE E2

WHERE E2.DNo = D.DNo AND E2.Salary > 600000) AS HighSalaryCount

FROM DEPARTMENT D

WHERE (SELECT COUNT(\*)

FROM EMPLOYEE E1

WHERE E1.DNo = D.DNo) > 5;

