UNIVERSITY OF VISVESVARYA COLLEGE OF ENGINEERING

K R Circle – 560 001

DATABASE MANAGEMENT SYSTEM MANUAL

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;
```

Book_id	Title	Publisher_Name	Author_Name	Branch_id	No_of_Copies
1	Database Systems	Pearson	Korth	10	5
2	Operating Systems	McGraw Hill	Silberschatz	11	3
4	Data Structures	O Reilly	Mark Allen Weiss	13	2
5	Machine Learning	Springer	Tom Mitchell	14	6

2. **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;
```

Card_No	Total_Books_Borrowed
1001	4
1002	5

3. **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_id	Title	Publisher_Name	Pub_Year
1	Database Systems	Pearson	2018
2	Operating Systems	McGraw Hill	2019
4	Data Structures	O Reilly	2017
5	Machine Learning	Springer	2021

Book Authors table:

Book_id	Author_Name
1	Korth
2	Silberschatz
4	Mark Allen Weiss
5	Tom Mitchell

4. 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;
```

Pub_Year	Total_Books
2018	1
2019	1
2017	1
2021	1

5. 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;
```

Book_id	Title	Total_Copies
1	Database Systems	5
2	Operating Systems	3
4	Data Structures	2
5	Machine Learning	6

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');
```

count_above_bangalore_avg
2

2. **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
);
```

Name
John
Alice

3. **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)
);
```

Salesman_id	Name	Status
1000	John	Has Customer
1001	Alice	Has Customer
1003	Charlie	Has Customer
1004	David	Has Customer
1002	Bob	No Customer

4. **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);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Ord_Date	Salesman_id	Salesman_Name	Customer_id	Cust_Name	Purchase_Amt
2024-03-01	1000	John	2001	Sita	7000
2024-03-02	1001	Alice	2003	Meera	8000
2024-03-03	1004	David	2005	Sunita	6000

5. **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;
```

Ord_No	Purchase_Amt	Ord_Date	Customer_id	Salesman_id
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
NULL	NULL	NULL	NULL	NULL

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');
```

Mov_Title
Psycho

2. **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
   )
);
```

Mov_Title
Psycho
Jaws
Catch Me If You Can

3. **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
);
```

Act_Name
Anthony Perkins
Tom Hanks

4. **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;
```

Mov_Title	Max_Rev_Stars
Catch Me If You Can	5
Inception	5
Jaws	4
Psycho	5
The Revenant	3

5. **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'
);
```

Mov_id	Rev_Stars
1	5
2	5
3	5
4	3
5	5

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'
    )
);
```

USN	SName	Address	Phone	Gender
1BI19CS112	Sunita Patel	Mysore	9876543212	Female
1BI19CS113	Vikram Singh	Delhi	9876543213	Male
NULL	NULL	NULL	NULL	NULL

2. 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;
```

Sem	Sec	Gender	Count
1	A	Male	1
4	C	Female	1
4	C	Male	1
8	A	Male	1
8	C	Female	1

3. 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';
```

Subcode	Title	Test1
CS101	Introduction to Programming	18

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;
```

USN	Subcode	FinalIA	Category
1BI15CS116	CS110	20	Outstanding
1BI15CS123	CS111	16	Average

);

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'
);
```

PNo
201
NULL

2. **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';
```

SSN	Name	CurrentSalary	NewSalary	SalaryIncrease
101	John Scott	60000.00	66000.0000	6000.0000
103	Michael Scott	80000.00	88000.0000	8000.0000

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) 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');
```

TotalSalary	MaximumSalary	MinimumSalary	AverageSalary
308000.00	80000.00	51000.00	61600.000000

4. **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
  )
)
```

Name
John Scott
Alice Brown
Michael Scott
Robert King
Sophia Davis
Ethan White
Isabella Green

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;
```

DNo	HighSalaryCount
1	6