In [6]:

```
import sqlite3

# Create connection
conn = sqlite3.connect('library.db')
cursor = conn.cursor()

# Drop tables if they exist (for rerunning)
cursor.executescript('''
DROP TABLE IF EXISTS Return_Transaction;
DROP TABLE IF EXISTS Issue_Transaction;
DROP TABLE IF EXISTS Library_Transaction;
DROP TABLE IF EXISTS Book_Copy;
DROP TABLE IF EXISTS Book;
DROP TABLE IF EXISTS Faculty;
DROP TABLE IF EXISTS Student;
DROP TABLE IF EXISTS Member;

-- Create Book table
CREATE TABLE Book (
    Book_ID INTEGER PRIMARY KEY,
    Title TEXT NOT NULL,
    Author TEXT NOT NULL,
    Publisher TEXT,
    Price REAL NOT NULL
);

-- Create Book_Copy table (Composite key)
CREATE TABLE Book_Copy (
    Book_ID INTEGER,
    Serial_Number INTEGER,
    Status TEXT CHECK (Status IN ('Issued', 'Available')),
    PRIMARY KEY (Book_ID, Serial_Number),
    FOREIGN KEY (Book_ID) REFERENCES Book(Book_ID)
);

-- Create Member table (Base entity)
CREATE TABLE Member (
    Member_ID INTEGER PRIMARY KEY,
    Name TEXT NOT NULL,
    Email TEXT,
    Address TEXT
);

-- Create Student table (inherits from Member)
CREATE TABLE Student (
    Member_ID INTEGER PRIMARY KEY,
    Course TEXT,
    Semester INTEGER,
    Max_Books INTEGER,
    FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID)
);

-- Create Faculty table (inherits from Member)
CREATE TABLE Faculty (
    Member_ID INTEGER PRIMARY KEY,
    Department TEXT,
    Designation TEXT,
    Max_Books INTEGER,
    FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID)
);

-- Create Transaction table (Base entity)
CREATE TABLE Library_Transaction (
    Transaction_ID INTEGER PRIMARY KEY,
    Member_ID INTEGER,
    Book_ID INTEGER,
    Serial_Number INTEGER,
    FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
    FOREIGN KEY (Book_ID, Serial_Number) REFERENCES Book_Copy(Book_ID, Serial_Number)
);

-- Create Issue_Transaction table (inherits from Transaction)
CREATE TABLE Issue_Transaction (
```

```
    Transaction_ID INTEGER PRIMARY KEY,
    DT_Issue DATE,
    To_Be_Returned_By DATE,
    FOREIGN KEY (Transaction_ID) REFERENCES Library_Transaction(Transaction_ID)
);

-- Create Return_Transaction table (inherits from Transaction)
CREATE TABLE Return_Transaction (
    Transaction_ID INTEGER PRIMARY KEY,
    DT_Return DATE,
    FOREIGN KEY (Transaction_ID) REFERENCES Library_Transaction(Transaction_ID)
);
''')

conn.commit()
print("Tables created successfully!")
```

```
Tables created successfully!
```

In [7]:
```
cursor.executescript('''
-- Insert into Book table
INSERT INTO Book VALUES (1, 'Database Systems', 'Elmasri', 'Pearson', 500);
INSERT INTO Book VALUES (2, 'Operating Systems', 'Silberschatz', 'McGraw Hill', 600);

-- Insert into Book_Copy table
INSERT INTO Book_Copy VALUES (1, 1, 'Available');
INSERT INTO Book_Copy VALUES (1, 2, 'Issued');
INSERT INTO Book_Copy VALUES (2, 1, 'Available');
INSERT INTO Book_Copy VALUES (2, 2, 'Issued');

-- Insert into Member table
INSERT INTO Member VALUES (101, 'Swapnamoy', 'swapna@gmail.com', 'Salt Lake');
INSERT INTO Member VALUES (102, 'Kristidhar', 'kristi@gmail.com', 'Garia');
INSERT INTO Member VALUES (103, 'Shahir', 'shahir@gmail.com', 'Jadavpur');
INSERT INTO Member VALUES (104, 'Abhi', 'abhi@gmail.com', 'Ballygunge');

-- Insert into Student table
INSERT INTO Student VALUES (101, 'CSE', 3, 5);
INSERT INTO Student VALUES (102, 'ECE', 4, 3);

-- Insert into Faculty table
INSERT INTO Faculty VALUES (103, 'CSE', 'Professor', 10);
INSERT INTO Faculty VALUES (104, 'EE', 'Lecturer', 8);

-- Insert into Transaction table
INSERT INTO Library_Transaction VALUES (1, 101, 1, 2); -- Swapnamoy issued book copy
INSERT INTO Library_Transaction VALUES (2, 103, 2, 2); -- Shahir issued book copy

-- Insert into Issue_Transaction table
INSERT INTO Issue_Transaction VALUES (1, '2025-03-01', '2025-03-08');
INSERT INTO Issue_Transaction VALUES (2, '2025-03-02', '2025-03-09');

-- Insert into Return_Transaction table (only one return)
INSERT INTO Return_Transaction VALUES (1, '2025-03-10');
''')

conn.commit()
print("Sample data inserted successfully!")
```

```
Sample data inserted successfully!
```

In [8]:
```
query = '''
SELECT B.Book_ID, B.Title,
       COUNT(*) AS Total_Copies,
       SUM(CASE WHEN BC.Status = 'Issued' THEN 1 ELSE 0 END) AS Copies_Issued
FROM Book B
JOIN Book_Copy BC ON B.Book_ID = BC.Book_ID
GROUP BY B.Book_ID;
'''
result = cursor.execute(query).fetchall()
print(result)
```

```
[(1, 'Database Systems', 2, 1), (2, 'Operating Systems', 2, 1)]
```

In [11]:
```
query = '''
SELECT M.Member_ID, M.Name, T.Book_ID, IT.To_Be_Returned_By
```

```
FROM Issue_Transaction IT
JOIN Library_Transaction T ON IT.Transaction_ID = T.Transaction_ID
JOIN Member M ON T.Member_ID = M.Member_ID
WHERE IT.To_Be_Returned_By < DATE('now')
AND NOT EXISTS (
    SELECT 1
    FROM Return_Transaction RT
    WHERE RT.Transaction_ID = IT.Transaction_ID
);
'''
result = cursor.execute(query).fetchall()
print(result)
```

```
[(103, 'Shahir', 2, '2025-03-09')]
```

In [13]:
```
query = '''
SELECT M.Member_ID, M.Name, T.Book_ID,
       RT.DT_Return,
       IT.To_Be_Returned_By,
       (JULIANDAY(RT.DT_Return) - JULIANDAY(IT.To_Be_Returned_By)) AS Delay_Days
FROM Issue_Transaction IT
JOIN Library_Transaction T ON IT.Transaction_ID = T.Transaction_ID
JOIN Return_Transaction RT ON T.Transaction_ID = RT.Transaction_ID
JOIN Member M ON T.Member_ID = M.Member_ID
WHERE RT.DT_Return > IT.To_Be_Returned_By;
'''
result = cursor.execute(query).fetchall()
print(result)
```

```
[(101, 'Swapnamoy', 1, '2025-03-10', '2025-03-08', 2.0)]
```

In [15]:
```
query = '''
SELECT S.Member_ID, M.Name
FROM Student S
LEFT JOIN Library_Transaction T ON S.Member_ID = T.Member_ID
LEFT JOIN Member M ON S.Member_ID = M.Member_ID
WHERE T.Transaction_ID IS NULL;
'''
result = cursor.execute(query).fetchall()
print(result)
```

```
[(102, 'Kristidhar')]
```

In [ ]:
```
query = '''
SELECT F.Member_ID, M.Name
FROM Faculty F
LEFT JOIN Library_Transaction T ON F.Member_ID = T.Member_ID
LEFT JOIN Member M ON F.Member_ID = M.Member_ID
WHERE T.Transaction_ID IS NULL;
'''
result = cursor.execute(query).fetchall()
print(result)
```

In [16]:
```
query = '''
SELECT T.Book_ID, B.Title, COUNT(T.Transaction_ID) AS Issue_Count
FROM Library_Transaction T
JOIN Issue_Transaction IT ON T.Transaction_ID = IT.Transaction_ID
JOIN Book B ON T.Book_ID = B.Book_ID
GROUP BY T.Book_ID;
'''
result = cursor.execute(query).fetchall()
print(result)
```

```
[(1, 'Database Systems', 1), (2, 'Operating Systems', 1)]
```