National University of Computer & Emerging Sciences, Karachi
Fall-2025 School of Computing (BSCS, BSSE, BSCY, BSAI)
Assignment # 1

Subject: Database Systems -CS2005	**Post Date: 5/9/2025**
Total Marks: 40	**Due Date:** 21/9/2025
Course Instructors: Dr. Zulfiqar, Dr. Farrukh Salim, Basit Jasani, Ali Naseer, Atiya Jokhio, Javeria Farooq, Hajira Ahmed	

Instructions to be strictly followed.

- For all questions involving SQL Queries:
  - ***Submit the SQL Scripts in a .txt file.***
- It should be obvious that submitting your work after the due date will result in zero points being awarded.
- Plagiarism (copying/cheating) and late submissions result in a zero mark.

Question #1: Briefly answer the following questions **[10 points]**

a) What are the problems with File system data management?

a) Problems with File System Data Management

File-based systems (before DBMS) had many limitations:

1. **Data Redundancy & Inconsistency** – Same data stored in multiple files may differ (e.g., customer address updated in one file but not others).
2. **Difficulty in Accessing Data** – To retrieve specific information, complex application programs had to be written.
3. **Data Isolation** – Data scattered in different files with different formats makes integration hard.
4. **Integrity Problems** – Difficult to enforce constraints like *no negative salary*.
5. **Atomicity Issues** – Ensuring transactions (all-or-nothing operations) was very hard.
6. **Concurrent Access Anomalies** – File systems don't handle multiple users updating the same file properly.
7. **Security Problems** – Limited security mechanisms compared to DBMS (e.g., access control, views).

b) Explain the usage of the Composite Primary key with an example.

A **Composite Primary Key** is formed by combining two or more attributes to uniquely identify a record when one attribute alone is not sufficient.

c) what operations are performed by the application program when DBMS is used.

- **Request Data from DBMS** – By writing queries (SQL).
- **Manipulate Data** – Insert, update, delete, and retrieve records.
- **Perform Business Logic** – Validate user inputs, calculations, or workflows before/after DBMS operations.

- **Transaction Management** – Ensure atomicity, consistency, isolation, and durability (ACID).
- **Provide User Interface** – Display data in reports, dashboards, or input forms.

d) which independence is difficult to achieve in three schema architectures and why. Elaborate with an example.

In a three-schema architecture, achieving external-to-conceptual data independence(logical independence) is the most difficult because it requires complex transformations and mapping of data requests and responses between different user views and the unified logical structure, especially when multiple complex views and the central database logic are significantly different

e) A key is a superkey but not vice versa. Explain this statement with an example.

Superkey: A set of attributes that uniquely identifies a record.
Candidate Key: A minimal superkey (no extra attributes).
Primary Key: A chosen candidate key.
Statement meaning:
Every **key** is a **superkey** (it uniquely identifies records).
But not every **superkey** is a **key** (because it may have extra unnecessary attributes).

Question 2,part1

```
CREATE TABLE Members (
   MemberID INT PRIMARY KEY,                    -- PK constraint
   Name VARCHAR(100) NOT NULL,                  -- Cannot be NULL
   Email VARCHAR(100) UNIQUE NOT NULL,          -- Must be unique and not NULL
   JoinDate DATE DEFAULT CURRENT_DATE,          -- Default current date
   CHECK (LENGTH(Name) > 0)                     -- Name cannot be empty
);

CREATE TABLE Books (
   BookID INT PRIMARY KEY,                      -- PK constraint
   Title VARCHAR(200) NOT NULL,                 -- Cannot be NULL
   Author VARCHAR(100) NOT NULL,                -- Cannot be NULL
   CopiesAvailable INT DEFAULT 0 CHECK (CopiesAvailable >= 0)  -- Cannot be negative
);


CREATE TABLE IssuedBooks (
   IssueID INT PRIMARY KEY,                     -- PK constraint
   MemberID INT NOT NULL,                       -- FK constraint
   BookID INT NOT NULL,                         -- FK constraint
   IssueDate DATE DEFAULT CURRENT_DATE,         -- Default current date
   ReturnDate DATE,                             -- Nullable
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID)  -- FK to Members
      ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (BookID) REFERENCES Books(BookID)        -- FK to Books
      ON DELETE CASCADE ON UPDATE CASCADE,
   CHECK (ReturnDate IS NULL OR ReturnDate >= IssueDate) -- ReturnDate cannot be before IssueDate
);
```

Question 2,part2

Relations:

Relations	No of columns
Members	**3**
Books	**4**
Issued books	**5**

Column names	datatypes	constraints	Belongs to
Member ID	**INT**	**PK**	**Members**
Name	**VARCHAR**	**NOT NULL**	**Members**
Email	**VARCHAR**	**UNIQUE NOT NULL**	**Members**
Join date	**DATE**	**DEFAUKLT**	**Members**

Add values for books and issued books relations as well …….

Question 2,part 3

a)

```
-- Insert Members
INSERT INTO Members (MemberID, Name, Email, JoinDate)
VALUES
(1, 'Alice Brown', 'alice@example.com', '2025-01-15'),
(2, 'John Smith', 'john@example.com', '2025-02-01'),
(3, 'Sara Khan', 'sara@example.com', '2025-02-10');

-- Insert Books
INSERT INTO Books (BookID, Title, Author, CopiesAvailable)
VALUES
(101, 'Database Systems', 'Elmasri & Navathe', 5),
(102, 'Operating System Concepts', 'Silberschatz', 3),
(103, 'Clean Code', 'Robert C. Martin', 4);
```

b)

```
-- Issue a Book to a Member
INSERT INTO IssuedBooks (IssueID, MemberID, BookID, IssueDate, ReturnDate)
VALUES
(1001, 1, 101, CURRENT_DATE, NULL);

-- Update Available Copies of the Book
UPDATE Books
SET CopiesAvailable = CopiesAvailable - 1
WHERE BookID = 101;
```

c)

```
SELECT m.Name AS MemberName, b.Title AS BookTitle, i.IssueDate
FROM IssuedBooks i
JOIN Members m ON i.MemberID = m.MemberID
JOIN Books b ON i.BookID = b.BookID;
```

Question 2,part4

a)

```
-- This will cause a PRIMARY KEY violation
INSERT INTO Members (MemberID, Name, Email, JoinDate)
```

```
VALUES (1, 'Duplicate User', 'duplicate@example.com', '2025-03-01');
```
b)
```
-- This will cause a FOREIGN KEY violation
INSERT INTO IssuedBooks (IssueID, MemberID, BookID, IssueDate, ReturnDate)
VALUES (1002, 99, 101, CURRENT_DATE, NULL);
```

c)
```
-- This will cause a CHECK constraint violation
UPDATE Books
SET CopiesAvailable = -5
WHERE BookID = 101;
```
Question 2,part 5

1)we can adda a book reservation feature

2) we can associate a fine management system in which the borrower would be penalized for not returning book in time.

Question 2,part 6

a.
```
SELECT Name
FROM Members
WHERE MemberID NOT IN (
    SELECT DISTINCT MemberID
    FROM IssuedBooks
);
```

b.

```
SELECT Title, CopiesAvailable
FROM Books
WHERE CopiesAvailable = (
    SELECT MAX(CopiesAvailable) FROM Books
);
```

c.
```
SELECT Name, COUNT(*) AS TotalIssued
FROM Members m
JOIN IssuedBooks i ON m.MemberID = i.MemberID
GROUP BY Name
ORDER BY TotalIssued DESC
LIMIT 1;
```
d.
```
SELECT Title
FROM Books
WHERE BookID NOT IN (
    SELECT DISTINCT BookID
    FROM IssuedBooks
);
```
e.
```
SELECT DISTINCT m.Name
FROM Members m
JOIN IssuedBooks i ON m.MemberID = i.MemberID
```

```
WHERE i.ReturnDate IS NULL
  AND i.IssueDate < CURRENT_DATE - INTERVAL '30 DAY';
```

Question #3: Consider the following details given below and write each of the following queries in SQL. [15 points]

- Create a table Patient with attributes: Patient_ID, Name, Gender, DOB, Email, Phone, Address, Username, and Password.
- Create a table Doctor with attributes: Doctor_ID, Name, Specialization, Username, and Password.
- Create a table Appointment with attributes: Appointment_ID, Appointment_Date, Appointment_Time, Status, Clinic_Number, Patient_ID, and Doctor_ID.
- Create a table Prescription with attributes: Prescription_ID, Date, Doctor_Advice, Followup_Required, Patient_ID, and Doctor_ID.
- Create a table Invoice with attributes: Invoice_ID, Invoice_Date, Amount, Payment_Status, Payment_Method, and Patient_ID.
- Create a table Tests with attributes (Test_ID, Blood Test, X-Ray, MRI, CT Scan)

Apply some constraints while Creating a Patient table that includes all of the following:

- Patient_ID as **PRIMARY KEY**
- Name as **NOT NULL**
- Email as **UNIQUE**
- Gender with a **CHECK constraint** allowing only 'M' or 'F'

```
CREATE TABLE Patient (
    Patient_ID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Gender CHAR(1) CHECK (Gender IN ('M', 'F')),
    DOB DATE,
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(15),
    Address VARCHAR(255),
    Username VARCHAR(50),
    Password VARCHAR(50)
);
```

DML Queries:

a) Update the phone number and email of a patient in the Patient table

```
UPDATE Patient
SET Phone = '0300-1234567',
   Email = 'new_email@example.com'
    WHERE Patient_ID = 1;
```

b) Update the payment status of an invoice in the Invoice table from "Unpaid" to "Paid".

```
UPDATE Invoice
SET Payment_Status = 'Paid'
WHERE Invoice_ID = 101
  AND Payment_Status = 'Unpaid';
```

c) Delete all cancelled appointments from the Appointment table.

```
DELETE FROM Appointment
WHERE Status = 'Cancelled';
```

d) Delete an invoice from the Invoice table for a patient who has been refunded.

```
DELETE FROM Invoice
WHERE Patient_ID = 1
  AND Payment_Status = 'Refunded';
```

e) Select all appointments that are still "Booked".

```
SELECT *
FROM Appointment
WHERE Status = 'Booked';
```

f) Select all invoices that are "Unpaid".

```
SELECT *
FROM Invoice
WHERE Payment_Status = 'Unpaid';
```

g) Select all lab tests of type "Blood Test".

```
SELECT *
```

```
FROM Tests
WHERE Blood_Test = 'Yes';
```

h) Select all prescriptions issued on '2025-09-02'.

```
SELECT *
FROM Prescription
WHERE Date = '2025-09-02';
```

Advance SQL:

a) Show all patients with their doctors booked.

```
SELECT
   p.Patient_ID,
   p.Name AS Patient_Name,
   d.Doctor_ID,
   d.Name AS Doctor_Name,
   a.Appointment_Date,
   a.Appointment_Time,
   a.Status
FROM Patient p
JOIN Appointment a ON p.Patient_ID = a.Patient_ID
JOIN Doctor d ON a.Doctor_ID = d.Doctor_ID
WHERE a.Status = 'Booked';
```

b) Show all lab tests of patients and the doctor who requested them.

```
SELECT
   t.Test_ID,
   p.Name AS Patient_Name,
   d.Name AS Doctor_Name,
   t.Blood_Test,
   t.X_Ray,
   t.MRI,
   t.CT_Scan
FROM Tests t
JOIN Appointment a ON t.Test_ID = a.Appointment_ID   -- assumption: Test_ID matches Appointment_ID
JOIN Patient p ON a.Patient_ID = p.Patient_ID
JOIN Doctor d ON a.Doctor_ID = d.Doctor_ID;
```

c) Show prescriptions with medicines **only for patients named "Ali Khan"**.

```
SELECT
   pr.Prescription_ID,
   p.Name AS Patient_Name,
   pr.Date,
   pr.Doctor_Advice
FROM Prescription pr
JOIN Patient p ON pr.Patient_ID = p.Patient_ID
WHERE p.Name = 'Ali Khan'
  AND pr.Doctor_Advice IS NOT NULL;
```

d) Show prescriptions with doctors **where follow-up is required**.

```
SELECT
```

```
    pr.Prescription_ID,
    p.Name AS Patient_Name,
    d.Name AS Doctor_Name,
    pr.Date,
    pr.Doctor_Advice,
    pr.Followup_Required
FROM Prescription pr
JOIN Patient p ON pr.Patient_ID = p.Patient_ID
JOIN Doctor d ON pr.Doctor_ID = d.Doctor_ID
WHERE pr.Followup_Required = 'Yes';
```

Good Luck!