Q1: Library Management (3NF)

```
CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(100),
   Author VARCHAR(100),
   Publisher VARCHAR(100),
   YearPublished INT
);

CREATE TABLE Members (
   MemberID INT PRIMARY KEY,
   Name VARCHAR(100),
   Address VARCHAR(255),
   PhoneNumber VARCHAR(15)
);

CREATE TABLE Borrow (
   BorrowID INT PRIMARY KEY,
   BookID INT,
   MemberID INT,
   BorrowDate DATE,
   ReturnDate DATE,
   FOREIGN KEY (BookID) REFERENCES Books(BookID),
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID)
);

-- Sample Data & Output
INSERT INTO Books VALUES (1, 'DBMS Concepts', 'Korth', 'McGraw Hill', 2019);
INSERT INTO Members VALUES (101, 'Alice', 'Chennai', '9876543210');
INSERT INTO Borrow VALUES (1, 1, 101, '2025-04-25', NULL);

SELECT m.Name, b.Title, br.BorrowDate
FROM Borrow br
JOIN Books b ON br.BookID = b.BookID
JOIN Members m ON br.MemberID = m.MemberID;

-- Output:
-- | Name  | Title         | BorrowDate  |
-- | Alice | DBMS Concepts | 2025-04-25  |
```

Q2: Hospital Management (3NF)

```
CREATE TABLE Doctors (
   DoctorID INT PRIMARY KEY,
   Name VARCHAR(100),
   Specialization VARCHAR(100)
);

CREATE TABLE Patients (
   PatientID INT PRIMARY KEY,
   Name VARCHAR(100),
    Address VARCHAR(255)
);

CREATE TABLE Appointments (
    AppointmentID INT PRIMARY KEY,
    PatientID INT,
    DoctorID INT,
    AppointmentDate DATE,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);

-- Sample Data & Output
INSERT INTO Doctors VALUES (1, 'Dr. Smith', 'Cardiology');
INSERT INTO Patients VALUES (101, 'John', 'Chennai');
INSERT INTO Appointments VALUES (1001, 101, 1, '2025-04-28');

SELECT p.Name AS Patient, d.Name AS Doctor, a.AppointmentDate
FROM Appointments a
JOIN Patients p ON a.PatientID = p.PatientID
JOIN Doctors d ON a.DoctorID = d.DoctorID;

-- Output:
-- | Patient | Doctor    | AppointmentDate |
-- | John    | Dr. Smith | 2025-04-28      |
```

Q3: Arithmetic Operations with Exception Handling

```
DECLARE
    a NUMBER := 10;
    b NUMBER := 0;
    res NUMBER;
BEGIN
    res := a + b;
    DBMS_OUTPUT.PUT_LINE('Addition: ' || res);

    BEGIN
        res := a / b;
        DBMS_OUTPUT.PUT_LINE('Division: ' || res);
    EXCEPTION
        WHEN ZERO_DIVIDE THEN
            DBMS_OUTPUT.PUT_LINE('Division by zero is not allowed.');
    END;
END;

-- Output:
-- Addition: 10
-- Division by zero is not allowed.
```

Q4: Employee & Department Tables with Queries

```
CREATE TABLE Department (
```

```sql
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(100)
);

CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(100),
    Salary INT,
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
);

-- Sample Data
INSERT INTO Department VALUES (1, 'IT'), (2, 'HR');
INSERT INTO Employee VALUES (101, 'Alice', 50000, 1), (102, 'Bob', 60000, 1), (103, 'Carol', 40000, 2);

-- i) Specific Department
SELECT Name FROM Employee WHERE DeptID = 1;

-- ii) Nested Query for Highest Avg Salary
SELECT DeptID
FROM Employee
GROUP BY DeptID
ORDER BY AVG(Salary) DESC
FETCH FIRST 1 ROW ONLY;

-- iii) Inner Join
SELECT e.Name, d.DeptName
FROM Employee e
INNER JOIN Department d ON e.DeptID = d.DeptID;

-- iv) Outer Join
SELECT e.Name, d.DeptName
FROM Employee e
RIGHT OUTER JOIN Department d ON e.DeptID = d.DeptID;
```