```sql
-- Section A: (10 Marks)

-- 1. Create Tables and Insert Data

CREATE TABLE Patients (
    PatientID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DOB DATE NOT NULL
);

CREATE TABLE Doctors (
    DoctorID INT PRIMARY KEY AUTO_INCREMENT,
    DoctorName VARCHAR(100) NOT NULL,
    Specialization VARCHAR(100)
);

CREATE TABLE Appointments (
    AppointmentID INT PRIMARY KEY AUTO_INCREMENT,
    PatientID INT,
    DoctorID INT,
    AppointmentDate DATE NOT NULL,
    FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);

-- Insert sample data
INSERT INTO Patients (FirstName, LastName, DOB) VALUES
('John', 'Doe', '1990-01-01'),
('Alice', 'Smith', '1985-05-12'),
('Bob', 'Brown', '1975-11-23'),
('Carol', 'Jones', '2000-07-30'),
('David', 'Lee', '1992-03-17');

INSERT INTO Doctors (DoctorName, Specialization) VALUES
('Dr. Shah', 'Cardiologist'),
('Dr. Mehta', 'Neurologist'),
('Dr. Iyer', 'Pediatrician'),
('Dr. Kapoor', 'Dermatologist'),
('Dr. Khan', 'Cardiologist');

INSERT INTO Appointments (PatientID, DoctorID, AppointmentDate) VALUES
(1, 1, '2023-07-01'),
(2, 2, '2023-08-15'),
(3, 1, '2023-07-01'),
(4, 3, '2023-09-10'),
(5, 5, '2023-07-01');

-- 2. Foreign Key constraints already added in table definition.

-- 3. Fetch patients with an appointment on '2023-07-01' or with a cardiologist
SELECT p.*
FROM Patients p
JOIN Appointments a ON p.PatientID = a.PatientID
```

```
JOIN Doctors d ON a.DoctorID = d.DoctorID
WHERE a.AppointmentDate = '2023-07-01' OR d.Specialization = 'Cardiologist';

-- 4. Fetch first 5 patients
SELECT * FROM Patients
LIMIT 5;

-- Section B: Intermediate SQL Queries (15 Marks)

-- 5. Appointments per doctor
SELECT d.DoctorName, COUNT(a.AppointmentID) AS TotalAppointments
FROM Doctors d
LEFT JOIN Appointments a ON d.DoctorID = a.DoctorID
GROUP BY d.DoctorID;

-- 6. Highest paid doctor (Assuming Salary column exists)
ALTER TABLE Doctors ADD Salary INT;

UPDATE Doctors SET Salary = 80000 WHERE DoctorID = 1;
UPDATE Doctors SET Salary = 90000 WHERE DoctorID = 2;
UPDATE Doctors SET Salary = 95000 WHERE DoctorID = 3;
UPDATE Doctors SET Salary = 85000 WHERE DoctorID = 4;
UPDATE Doctors SET Salary = 99000 WHERE DoctorID = 5;

SELECT DoctorName, Specialization
FROM Doctors
ORDER BY Salary DESC
LIMIT 1;

-- 7. Create view and update data
CREATE VIEW PatientAppointments AS
SELECT p.PatientID, p.FirstName, p.LastName, a.DoctorID, a.AppointmentDate
FROM Patients p
JOIN Appointments a ON p.PatientID = a.PatientID;

UPDATE PatientAppointments
SET AppointmentDate = '2024-01-01'
WHERE PatientID = 1;

-- 8. LEFT JOIN between Patients and Appointments
SELECT p.PatientID, p.FirstName, a.AppointmentID
FROM Patients p
LEFT JOIN Appointments a ON p.PatientID = a.PatientID;

-- 9. Temporary table for patients older than 60
CREATE TEMPORARY TABLE ElderlyPatients AS
SELECT * FROM Patients
WHERE TIMESTAMPDIFF(YEAR, DOB, CURDATE()) > 60;


-- Additional Solutions --

-- 10. Stored procedure to get DoctorName and total number of appointments
DELIMITER //
```

```
CREATE PROCEDURE GetDoctorDetails(IN inputDoctorID INT)
BEGIN
    SELECT d.DoctorName, COUNT(a.AppointmentID) AS TotalAppointments
    FROM Doctors d
    LEFT JOIN Appointments a ON d.DoctorID = a.DoctorID
    WHERE d.DoctorID = inputDoctorID
    GROUP BY d.DoctorID;
END //
DELIMITER ;

-- Section C: Advanced SQL Queries (15 Marks)

-- 11. WHILE loop to calculate appointment fees after a specific date
DELIMITER //
CREATE PROCEDURE CalculateFees(IN feePerAppointment INT, IN specDate DATE)
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE totalFee INT DEFAULT 0;
    DECLARE patID INT;
    DECLARE app_cursor CURSOR FOR
        SELECT PatientID FROM Appointments WHERE AppointmentDate > specDate;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN app_cursor;
    read_loop: LOOP
        FETCH app_cursor INTO patID;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SET totalFee = totalFee + feePerAppointment;
    END LOOP;
    CLOSE app_cursor;
    SELECT totalFee AS TotalFees;
END //
DELIMITER ;

-- 12. Stored function to calculate discounted bill
DELIMITER //
CREATE FUNCTION BILL_CALC(totalCost DECIMAL(10,2), discountPercent DECIMAL(5,2))
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE finalBill DECIMAL(10,2);
    SET finalBill = totalCost - (totalCost * discountPercent / 100);
    RETURN finalBill;
END //
DELIMITER ;

-- 13. Query to find doctors with more than 10 appointments
SELECT d.DoctorName, COUNT(a.AppointmentID) AS TotalAppointments
FROM Doctors d
JOIN Appointments a ON d.DoctorID = a.DoctorID
GROUP BY d.DoctorID
HAVING COUNT(a.AppointmentID) > 10;
```

```sql
-- 14. Trigger to log appointment updates
CREATE TABLE AppointmentLogs (
    LogID INT AUTO_INCREMENT PRIMARY KEY,
    AppointmentID INT,
    OldDate DATE,
    NewDate DATE,
    LogTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

DELIMITER //
CREATE TRIGGER LogAppointmentUpdate
BEFORE UPDATE ON Appointments
FOR EACH ROW
BEGIN
    IF OLD.AppointmentDate <> NEW.AppointmentDate THEN
        INSERT INTO AppointmentLogs (AppointmentID, OldDate, NewDate)
        VALUES (OLD.AppointmentID, OLD.AppointmentDate, NEW.AppointmentDate);
    END IF;
END //
DELIMITER ;

-- 15. Subquery to fetch the second most recent appointment
SELECT AppointmentDate, DoctorName
FROM Appointments a
JOIN Doctors d ON a.DoctorID = d.DoctorID
ORDER BY AppointmentDate DESC
LIMIT 1 OFFSET 1;
```