**Basic Queries**

1. **List all students along with their gender and date of birth.**
2. **Find the names of all courses along with their course codes.**
3. **Retrieve all teachers along with their department names.**
4. **Get the total fee amount and outstanding amount for each student.**

**Intermediate Queries**

1. **List all students enrolled in a specific course by course code.**
2. **Find all subjects taught by a particular teacher.**
3. **Show all mandatory subjects in each course.**
4. **Get the total amount paid by each student so far from the StudentFeePayment table.**
5. **List all students along with their contact details (phone, email, guardian name).**

**Advanced Queries and Functions**

1. **Write a query to calculate the average payment amount made by students.**
2. **Find students who have not paid their fees fully (OutstandingAmount > 0).**
3. **Write a query to show the number of students enrolled in each course.**
4. **Display the list of courses along with the number of mandatory and optional subjects in each.**

**Stored Procedure Questions**

1. **Write a stored procedure GetStudentFeeSummary that takes a StudentID and returns the total fee, total paid, and outstanding amount.**
2. **Write a stored procedure AddPayment that inserts a payment record for a student and updates the LastUpdated column in StudentFeeSummary.**

**Trigger Questions**

1. **Create a trigger that updates the LastUpdated column in StudentFeeSummary every time a payment is made (insert on StudentFeePayment).**
2. **Create a trigger that prevents deletion of a department if there are teachers assigned to it.**

**Function Questions**

1. **Create a scalar function that returns the age of a student given their StudentID (based on DOB).**
2. **Create a function that returns the total number of courses a student is enrolled in.**

**🔗 JOIN Queries**

**20.** List student names, their enrolled courses, and course codes.

**21.** Show all students along with their contact information and enrolled course names.

**22.** Retrieve teacher names along with the subject names they teach and the department they belong to.

**23.** List all subjects along with the course(s) they are part of and whether they are mandatory.

**24.** Find all students along with the total amount paid and outstanding fee (using StudentFeeSummary and StudentFeePayment).

**25.** List the student name, course name, and the department head of the department the course's teacher belongs to.

**📊 GROUP BY with HAVING**

**26.** Find the number of students enrolled in each course and show only those courses with more than 3 students.

**27.** Find the total fee paid by students, grouped by course.

**28.** List departments having more than one teacher assigned.

**29.** Show payment methods used and the total amount paid by each method, but only include methods where total payment exceeds 5000.

**30.** Get the average fee paid by students, grouped by gender, and only include genders where average is above 3000.

**🧮 SQL Functions**

**31.** Create a scalar function fn\_GetStudentAge that returns the age of a student given their StudentID.

**32.** Write a function fn\_TotalPaidByStudent that takes a StudentID and returns the total amount paid.

**33.** Create a function fn\_EnrolledCourseCount to return the number of courses a student is enrolled in.

**34.** Write a table-valued function that returns all subjects taught by a specific teacher.

**35.** Create a function to return the outstanding amount for a student based on real-time payment data (not the computed column).

**⚙️ Stored Procedures**

**36.** Create a stored procedure sp\_GetStudentDetails that takes a StudentID and returns personal info, course name, contact info, and fee summary.

**37.** Create a stored procedure sp\_AddStudentPayment that:

* Inserts a payment
* Updates the StudentFeeSummary table accordingly

**38.** Write a stored procedure to get all students who haven't paid any amount yet.

**39.** Create a procedure that returns all subjects in a course (by CourseID) along with whether they are mandatory.

**40.** Create a stored procedure to assign a teacher to a subject (inserts a record into TeacherSubject).

**🧨 Triggers**

**41.** Create a trigger trg\_UpdateFeeSummary that updates TotalPaidAmount and LastUpdated in StudentFeeSummary when a new record is inserted in StudentFeePayment.

**42.** Write a trigger that prevents deleting a student if there are fee payments associated.

**43.** Create a trigger that logs whenever a student contact info is updated (store logs in a new table StudentContactLog).

**44.** Write a trigger that prevents inserting a subject into SubjectCourse if it already exists for that course.