DAY 2:

Create queries:

```
-- Students table
CREATE TABLE Students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50),
    student_age INT,
    student_grade_id INT,
    FOREIGN KEY (student_grade_id) REFERENCES Grades(grade_id)
);

-- Grades table
CREATE TABLE Grades (
    grade_id INT PRIMARY KEY,
    grade_name VARCHAR(10)
);

-- Courses table
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(50)
);

-- Enrollments table
CREATE TABLE Enrollments (
    enrollment_id INT PRIMARY KEY,
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);
```

Insert queries:

```
-- Insert into Grades table
INSERT INTO Grades (grade_id, grade_name) VALUES
(1, 'A'),
(2, 'B'),
(3, 'C');

-- Insert into Courses table
```

```
INSERT INTO Courses (course_id, course_name) VALUES
(101, 'Math'),
(102, 'Science'),
(103, 'History');

-- Insert into Students table
INSERT INTO Students (student_id, student_name, student_age, student_grade_id) VALUES
(1, 'Alice', 17, 1),
(2, 'Bob', 16, 2),
(3, 'Charlie', 18, 1),
(4, 'David', 16, 2),
(5, 'Eve', 17, 1),
(6, 'Frank', 18, 3),
(7, 'Grace', 17, 2),
(8, 'Henry', 16, 1),
(9, 'Ivy', 18, 2),
(10, 'Jack', 17, 3);

-- Insert into Enrollments table
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES
(1, 1, 101, '2023-09-01'),
(2, 1, 102, '2023-09-01'),
(3, 2, 102, '2023-09-01'),
(4, 3, 101, '2023-09-01'),
(5, 3, 103, '2023-09-01'),
(6, 4, 101, '2023-09-01'),
(7, 4, 102, '2023-09-01'),
(8, 5, 102, '2023-09-01'),
(9, 6, 101, '2023-09-01'),
(10, 7, 103, '2023-09-01');
```

Questions:

1. Find all students enrolled in the Math course.
2. List all courses taken by students named Bob.
3. Find the names of students who are enrolled in more than one course.
4. List all students who are in Grade A (grade_id = 1).
5. Find the number of students enrolled in each course.
6. Retrieve the course with the highest number of enrollments.
7. List students who are enrolled in all available courses.
8. Find students who are not enrolled in any courses.
9. Retrieve the average age of students enrolled in the Science course.
10. Find the grade of students enrolled in the History course.

Assignment:

Please design and create the necessary tables (`Books`, `Authors`, `Publishers`, `Customers`, `Orders`, `Book_Authors`, `Order_Items`) for an online bookstore database. Ensure each table includes appropriate columns, primary keys, and foreign keys where necessary. Consider the relationships between these tables and how they should be defined.

Conceptual Modeling:

1. **Identify Entities and Relationships:**
    - **Entities**:
        - Book (with attributes like `book_id`, `title`, `author`, `genre`, `publisher`, `publication_year`)
        - Author (with attributes like `author_id`, `author_name`, `birth_date`, `nationality`)
        - Publisher (with attributes like `publisher_id`, `publisher_name`, `country`)
        - Customer (with attributes like `customer_id`, `customer_name`, `email`, `address`)
        - Order (with attributes like `order_id`, `order_date`, `customer_id`, `total_amount`)
    - **Relationships**:
        - Books are written by Authors (`many-to-many` relationship)

        - Books are published by Publishers (`many-to-one` relationship)
        - Customers place Orders (`one-to-many` relationship)
        - Orders contain Books (`many-to-many` relationship)

2. **Conceptual Model Representation**:

    - Use an Entity-Relationship Diagram (ERD) to visually represent entities, attributes, and relationships.

Logical Schema Design:

1. **Translate Entities to Tables:**
    - **Tables**:
        - `Books` table (with columns: `book_id`, `title`, `genre`, `publisher_id`, `publication_year`)
        - `Authors` table (with columns: `author_id`, `author_name`, `birth_date`, `nationality`)
        - `Publishers` table (with columns: `publisher_id`, `publisher_name`, `country`)
        - `Customers` table (with columns: `customer_id`, `customer_name`, `email`, `address`)
        - `Orders` table (with columns: `order_id`, `order_date`, `customer_id`, `total_amount`)
        - `Book_Authors` table (to manage the `many-to-many` relationship between `Books` and `Authors`)
        - `Order_Items` table (to manage the `many-to-many` relationship between `Orders` and `Books`)
2. **Define Relationships and Constraints:**
    - **Primary Keys**:
        - `book_id` in `Books`
        - `author_id` in `Authors`
        - `publisher_id` in `Publishers`
        - `customer_id` in `Customers`
        - `order_id` in `Orders`
    - **Foreign Keys**:
        - `publisher_id` in `Books` references `publisher_id` in `Publishers`
        - `customer_id` in `Orders` references `customer_id` in `Customers`
        - `book_id` and `author_id` in `Book_Authors` reference `book_id` and `author_id` in `Books` and `Authors`, respectively
        - `order_id` and `book_id` in `Order_Items` reference `order_id` in

`Orders` and `book_id` in `Books`, respectively