

DAY 2:

A. 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.

```
select student_name from Students
where student_id in (
  select Enrollments.student_id
  from Enrollments where Enrollments.course_id =(select Courses.course_id from courses
  where Courses.course_name = 'Math')
);
```

	ABC student_name
1	Alice
2	Charlie
3	David
4	Frank

2. List all courses taken by students named Bob.

```
select c.course_name from courses c
where c.course_id in
(select e.course_id from enrollments e
where e.student_id=
(select s.student_id from students s
where s.student_name='Bob'));
```

	ABC course_name
1	Science

3. Find the names of students who are enrolled in more than one course.

```
select student_name from students
where student_id IN (
select student_id FROM enrollments
group by student_id having COUNT(course_id) > 1
);
```

	ABC student_name
1	Alice
2	Charlie
3	David

4. List all students who are in Grade A (grade_id = 1).

	ABC student_name
1	Alice
2	Charlie
3	Eve
4	Henry

```
select student_name from students
where student_grade_id = 1;
```

5. Find the number of students enrolled in each course.

```
select course_name,
(select COUNT(*) from Enrollments E
where E.course_id = C.course_id
) AS number_of_students
from Courses C;
```

	ABC course_name	123 student_count
1	Math	4
2	History	2
3	Science	4

6. Retrieve the course with the highest number of enrollments.

```
select course_name, (
select COUNT(*) from Enrollments E
WHERE E.course_id = C.course_id
) AS number_of_students
from Courses C order by number_of_students desc limit 1;
```

	ABC course_name	123 number_of_students
1	Math	4

7. List students who are enrolled in all available courses.

```
select student_name FROM Students where student_id IN
(
select student_id from Enrollments GROUP BY student_id
having COUNT(DISTINCT course_id) = (select COUNT(*) FROM Courses)
);
```

	ABC student_name

8. Find students who are not enrolled in any courses.

```
SELECT student_name FROM Students WHERE student_id
NOT IN (
SELECT student_id
FROM Enrollments
);
```

	ABC student_name
1	Henry
2	Ivy
3	Jack

9. Retrieve the average age of students enrolled in the Science course.

```
select AVG(student_age) as Avg_Age from Students
where student_id IN (
select student_id from Enrollments
where course_id = (
select course_id from courses
where course_name = 'Science'
)
);
```

	123 avg
1	16.5

10. Find the grade of students enrolled in the History course.

```
select student_name, (select grade_name from grades G
where S.student_grade_id = G.grade_id
) as Grade
from Students S
where student_id in (
select student_id from Enrollments E
where E.course_id = (
select course_id from courses
where course_name = 'History'
)
);
```

	ABC student_name ▼	ABC grade ▼
1	Charlie	A
2	Grace	B

B. 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.

```
create table Publisher (  
publisher_id SERIAL primary key ,  
publisher_name VARCHAR(50) not NULL,  
country VARCHAR(50) not NULL  
);  
  
create table Book (  
book_id SERIAL primary key ,  
title VARCHAR(50) not NULL,  
author VARCHAR(50) not NULL,  
genre VARCHAR(50) not NULL,  
publisher_id int not NULL,  
publication_year DATE not NULL,  
foreign key (publisher_id) references Publisher(publisher_id)  
);  
  
create table Author (  
author_id SERIAL primary key ,  
author_name VARCHAR(50) not NULL,  
nationality VARCHAR(50) not NULL,  
birth_date DATE  
);  
  
create table Customer (  
customer_id SERIAL primary key ,  
customer_name VARCHAR(50) not NULL,  
email VARCHAR(50) not NULL,  
address VARCHAR(50) not NULL  
);  
  
create table Orders (  
order_id SERIAL primary key,  
customer_id int not NULL,  
total_amount int not NULL,  
order_date Date not NULL,  
foreign key (customer_id) references Customer(customer_id)  
);  
  
create table Book_Author(  
book_id int not null,  
author_id int not null,  
foreign key (book_id) references Book(book_id),  
foreign key (author_id) references Author(author_id)  
);  
create table Order_Items(  
order_id int not null,  
book_id int not null,  
foreign key (book_id) references Book(book_id),  
foreign key (order_id) references Orders(order_id)  
);
```

