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
```

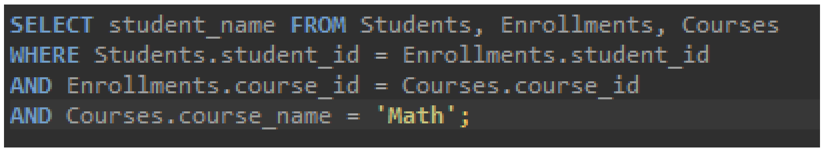
INSERT INTO Courses (course_id, course_name) VALUES
(101, 'Math'),
(102, 'Science'),
(103, 'History');

-- 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);

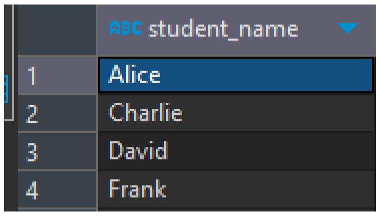
-- Insert into Enrollments table
INSERT INTO Enrollments (enrollment_id, student_id, course_id, enrollment_date) VALUES
(1, 1, 101, '2023-09-01'),
(2, 1, 102, '2023-09-01'),
(3, 2, 102, '2023-09-01'),
(4, 3, 101, '2023-09-01'),
(5, 3, 103, '2023-09-01'),
(6, 4, 101, '2023-09-01'),
(7, 4, 102, '2023-09-01'),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

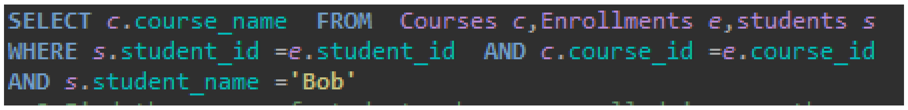


```
SELECT student_name FROM Students, Enrollments, Courses
WHERE Students.student_id = Enrollments.student_id
AND Enrollments.course_id = Courses.course_id
AND Courses.course_name = 'Math';
```

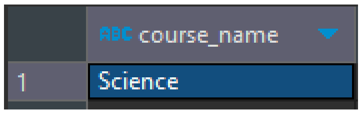


```
   student_name
1  Alice
2  Charlie
3  David
4  Frank
```

2. . List all courses taken by students named Bob.

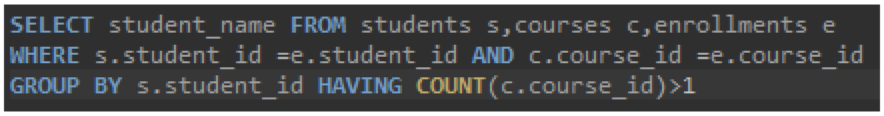


```
SELECT c.course_name  FROM  Courses c,Enrollments e,students s
WHERE s.student_id =e.student_id  AND c.course_id =e.course_id
AND s.student_name ='Bob'
```

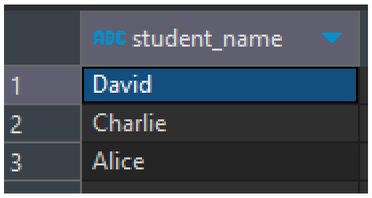


| | course_name |
|---|---|
| 1 | Science |

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

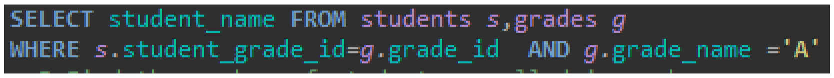


```
SELECT student_name FROM students s,courses c,enrollments e
WHERE s.student_id =e.student_id AND c.course_id =e.course_id
GROUP BY s.student_id HAVING COUNT(c.course_id)>1
```

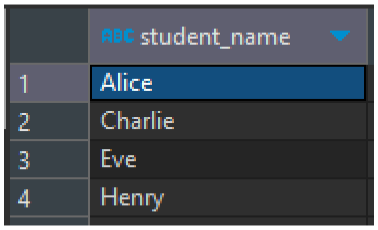


| | student_name |
|---|---|
| 1 | David |
| 2 | Charlie |
| 3 | Alice |

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

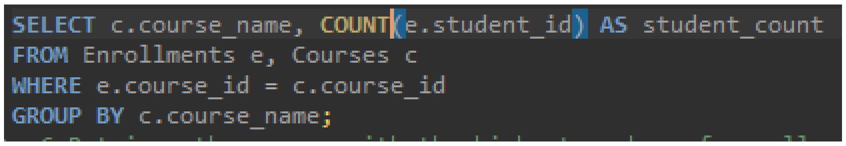


```
SELECT student_name FROM students s,grades g
WHERE s.student_grade_id=g.grade_id  AND g.grade_name ='A'
```

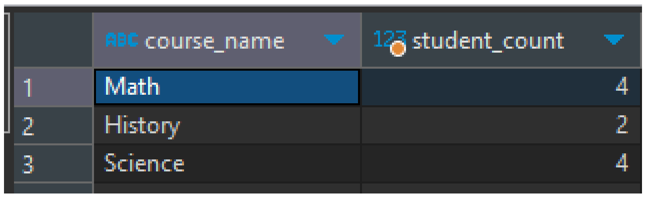


| | student_name |
|---|---|
| 1 | Alice |
| 2 | Charlie |
| 3 | Eve |
| 4 | Henry |

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

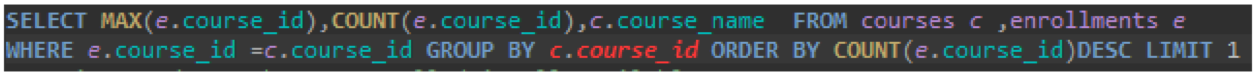


```
SELECT c.course_name, COUNT(e.student_id) AS student_count
FROM Enrollments e, Courses c
WHERE e.course_id = c.course_id
GROUP BY c.course_name;
```

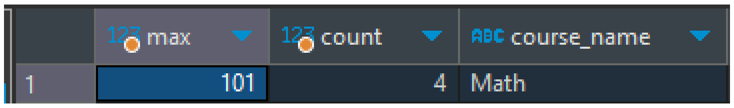


| | course_name | student_count |
|---|---|---|
| 1 | Math | 4 |
| 2 | History | 2 |
| 3 | Science | 4 |

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

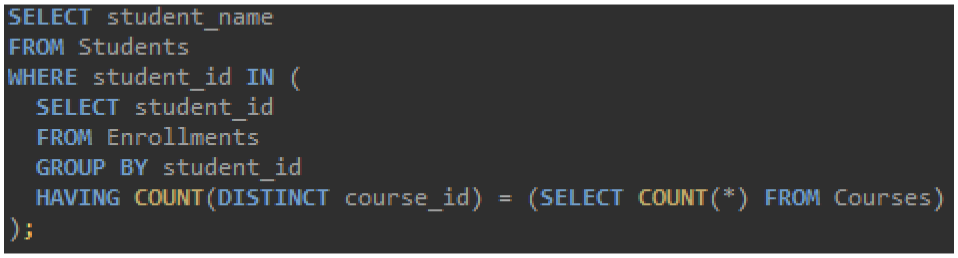


```
SELECT MAX(e.course_id),COUNT(e.course_id),c.course_name  FROM courses c ,enrollments e
WHERE e.course_id =c.course_id GROUP BY c.course_id ORDER BY COUNT(e.course_id)DESC LIMIT 1
```

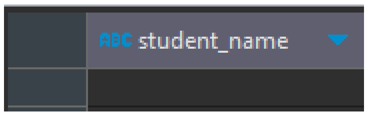


| | max | count | course_name |
|---|---|---|---|
| 1 | 101 | 4 | Math |

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

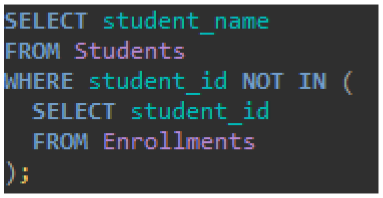


```sql
SELECT student_name
FROM Students
WHERE student_id IN (
  SELECT student_id
  FROM Enrollments
  GROUP BY student_id
  HAVING COUNT(DISTINCT course_id) = (SELECT COUNT(*) FROM Courses)
);
```

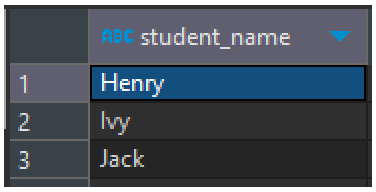


| | student_name |
|---|---|

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

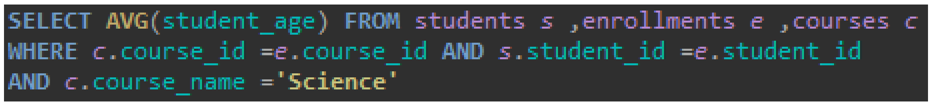


```sql
SELECT student_name
FROM Students
WHERE student_id NOT IN (
  SELECT student_id
  FROM Enrollments
);
```

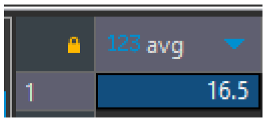


| | student_name |
|---|---|
| 1 | Henry |
| 2 | Ivy |
| 3 | Jack |

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

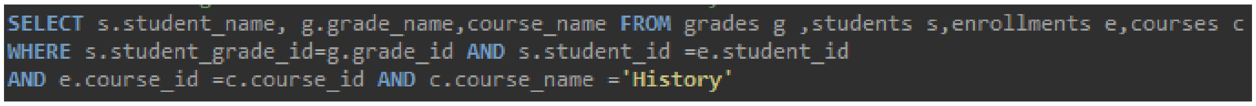


```sql
SELECT AVG(student_age) FROM students s ,enrollments e ,courses c
WHERE c.course_id =e.course_id AND s.student_id =e.student_id
AND c.course_name ='Science'
```

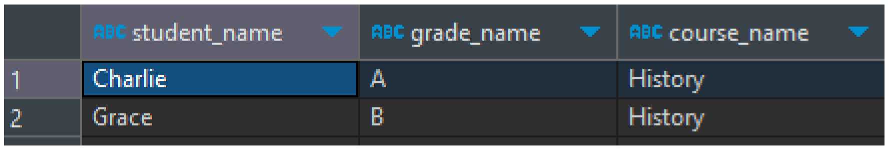


| | avg |
|---|---|
| 1 | 16.5 |

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

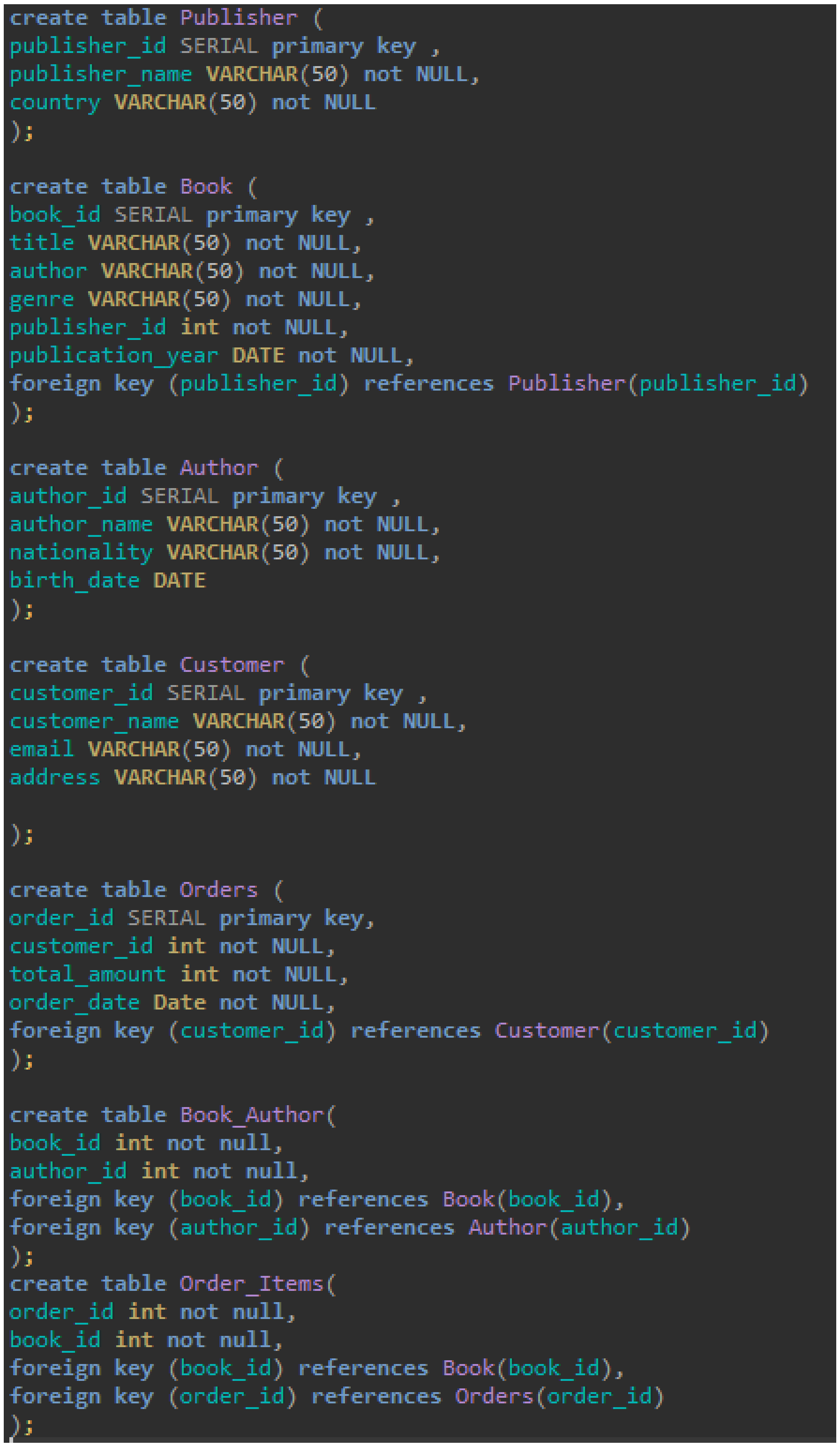


```sql
SELECT s.student_name, g.grade_name,course_name FROM grades g ,students s,enrollments e,courses c
WHERE s.student_grade_id=g.grade_id AND s.student_id =e.student_id
AND e.course_id =c.course_id AND c.course_name ='History'
```

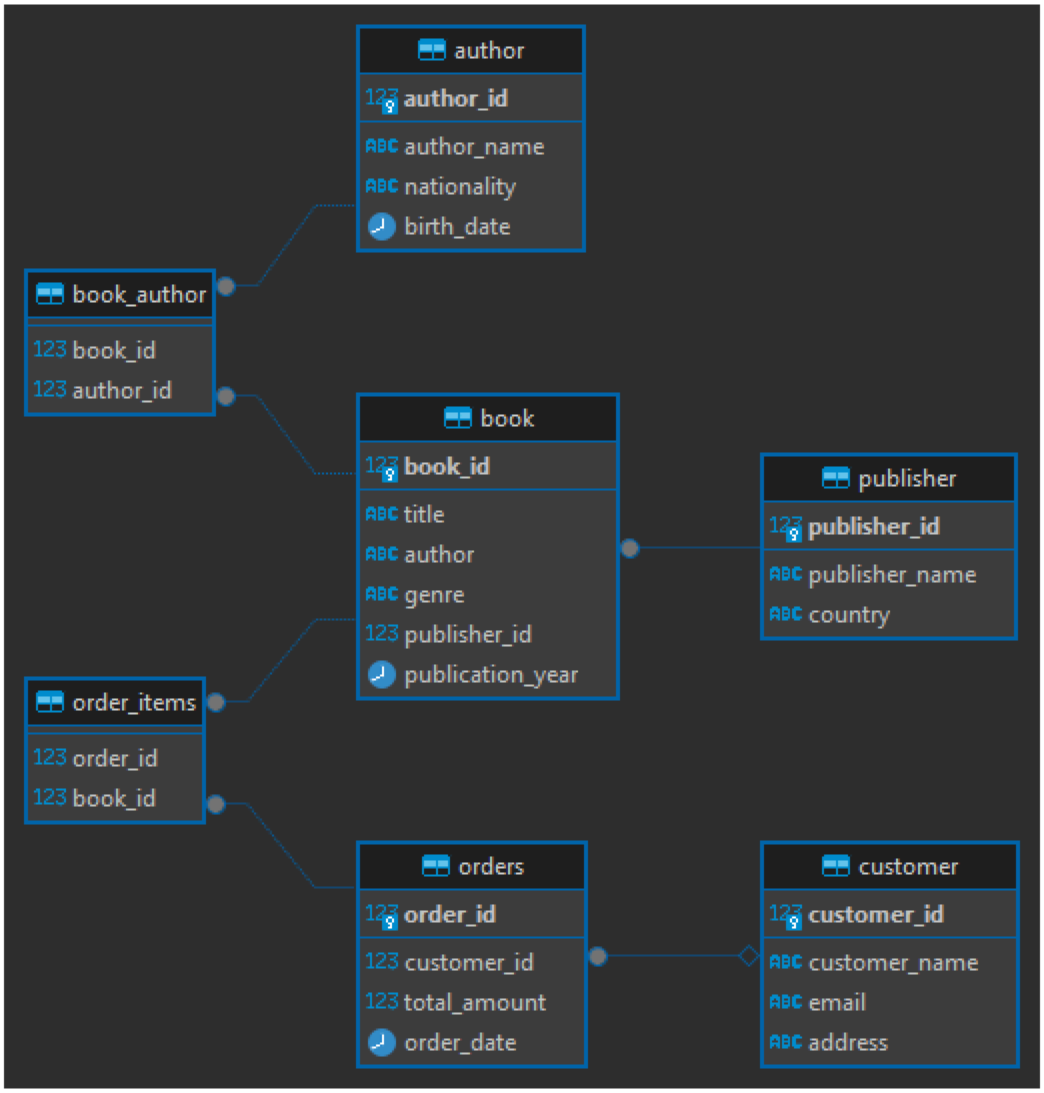


| | student_name | grade_name | course_name |
|---|---|---|---|
| 1 | Charlie | A | History |
| 2 | Grace | B | History |

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.



```sql
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)
);
```


author
author_id
author_name
nationality
birth_date
book_author
book_id
author_id
book
book_id
title
author
genre
publisher_id
publication_year
publisher
publisher_id
publisher_name
country
order_items
order_id
book_id
orders
order_id
customer_id
total_amount
order_date
customer
customer_id
customer_name
email
address