```
A. Create queries:
```

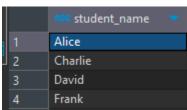
```
-- Students table
CREATE TABLE Students (
  student_id INT PRIMARY KEY,
  student_name VARCHAR(50),
  student_age INT,
  student grade id INT,
  FOREIGN KEY (student_grade_id) REFERENCES Grades(grade_id)
);
-- Grades table
CREATE TABLE Grades (
  grade_id INT PRIMARY KEY,
  grade_name VARCHAR(10)
);
-- 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, 'lvy', 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')
);
```



2. List all courses taken by students named Bob.

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
);

ADC student_name

1 Alice
2 Charlie
3 David
```

4. List all students who are in Grade A (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;
```



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  125 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
);

ADC student_name

1 Henry

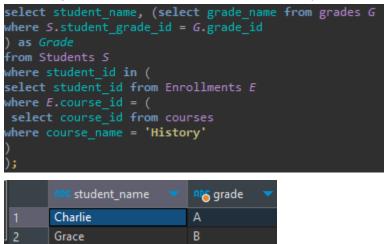
2 lvy

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
16.5
```

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



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,
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)
```

