Nabin Thapa - 17 (I)Database Day 2 - Assignment

Part 1:

```sql
CREATE SCHEMA IF NOT EXISTS assignment2;
-- Grades table
CREATE TABLE assignment2.Grades (
   grade_id INT PRIMARY KEY,
   grade_name VARCHAR(10)
);
-- Students table
CREATE TABLE assignment2.Students (
   student_id INT PRIMARY KEY,
   student_name VARCHAR(50),
   student_age INT,
   student_grade_id INT,
   FOREIGN KEY (student_grade_id) REFERENCES assignment2.Grades(grade_id)
);
-- Courses table
CREATE TABLE assignment2.Courses (
   course_id INT PRIMARY KEY,
   course_name VARCHAR(50)
);
-- Enrollments table
CREATE TABLE assignment2.Enrollments (
   enrollment_id INT PRIMARY KEY,
   student_id INT,
   course_id INT,
   enrollment_date DATE,
   FOREIGN KEY (student_id) REFERENCES assignment2.Students(student_id),
   FOREIGN KEY (course_id) REFERENCES assignment2.Courses(course_id)
);
```

```sql
-- Insert into Grades table
INSERT INTO assignment2.Grades (grade_id, grade_name) VALUES
(1, 'A'),
(2, 'B'),
(3, 'C');
-- Insert into Courses table
INSERT INTO assignment2.Courses (course_id, course_name) VALUES
(101, 'Math'),
(102, 'Science'),
(103, 'History');
-- Insert into Students table
INSERT INTO assignment2.Students (student_id, student_name, student_age,
student_grade_id) VALUES
```

Nabin Thapa - 17 (I)Database Day 2 - Assignment

```
(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 assignment2.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.

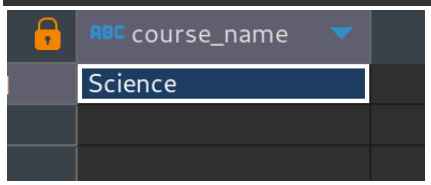
```
SELECT student_name FROM assignment2.Students s
WHERE  s.student_id IN(
  SELECT student_id FROM assignment2.Enrollments e, assignment2.Courses c
  WHERE  e.course_id = c.course_id
  AND c.course_name = 'Math'
);
```

	student_name
1	Alice
2	Charlie
3	David
4	Frank

Nabin Thapa - 17 (I)Database Day 2 - Assignment

2. List all courses taken by students named Bob.

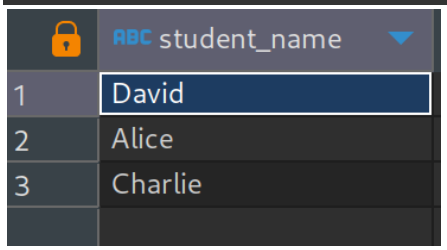
```
SELECT course_name FROM assignment2.Courses
WHERE course_id IN(
  SELECT e.course_id FROM assignment2.Enrollments e, assignment2.Students s
  WHERE  e.student_id = s.student_id
  AND s.student_name = 'Bob'
);
```



course_name
Science

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

```
SELECT student_name FROM assignment2.Students s
WHERE student_id IN(
  SELECT student_id FROM assignment2.Enrollments e
  GROUP BY student_id
  HAVING count(course_id) > 1
);
```



	student_name
1	David
2	Alice
3	Charlie

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

```
SELECT student_name FROM assignment2.Students s
WHERE student_grade_id IN(
  SELECT grade_id FROM assignment2.Grades g
  WHERE grade_name = 'A'
);
```

Nabin Thapa - 17 (I)Database Day 2 - Assignment

	student_name
1	Alice
2	Charlie
3	Eve
4	Henry

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

```
SELECT course_name,
   (SELECT COUNT(*)
    FROM assignment2.Enrollments e
    WHERE e.course_id = c.course_id) AS student_count
FROM assignment2.Courses c
```

	course_name	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 assignment2.Enrollments e
  WHERE c.course_id = e.course_id
)
FROM assignment2.Courses c
WHERE c.course_id IN (
  SELECT course_id
  FROM assignment2.Enrollments e
  GROUP BY course_id
  HAVING COUNT(e.student_id) = (
    SELECT COUNT(student_id) FROM assignment2.Enrollments
    GROUP BY course_id
    ORDER BY COUNT(student_id) DESC LIMIT 1
  )
);
```

Nabin Thapa - 17 (I)Database Day 2 - Assignment

	course_name	student_count
1	Math	4
2	Science	4

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

```
SELECT student_name FROM assignment2.Students s
WHERE student_id IN(
  SELECT student_id FROM assignment2.Enrollments e
  GROUP BY student_id
  HAVING COUNT(e.course_id) = (
    SELECT COUNT(*) FROM assignment2.Courses
  )
);
```

	student_name

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

```
SELECT student_name FROM assignment2.Students s
WHERE s.student_id NOT IN (
 SELECT student_id FROM assignment2.Enrollments
 GROUP BY student_id
 HAVING COUNT(course_id) > 0
);
```

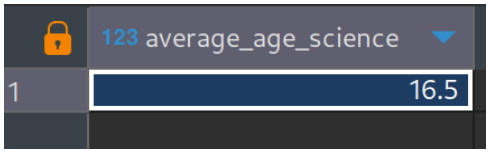
	student_name
1	Henry
2	Jack
3	Ivy

Nabin Thapa - 17 (I)Database Day 2 - Assignment

Nabin Thapa - 17 (I)Database Day 2 - Assignment

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

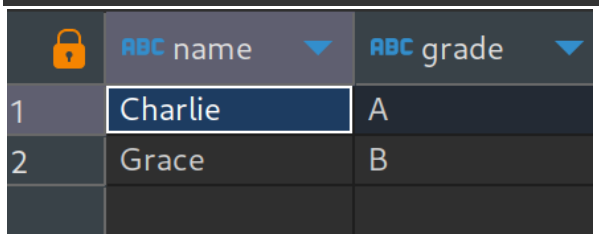
```
SELECT AVG(s.student_age) AS average_age_science
FROM assignment2.Students s WHERE student_id IN(
  SELECT student_id FROM assignment2.Enrollments e
  WHERE course_id IN (
    SELECT  course_id FROM assignment2.Courses c
    WHERE course_name = 'Science'
  )
);
```



	123 average_age_science
1	16.5

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

```
SELECT student_name,
(
  SELECT g.grade_name FROM assignment2.Grades g
  WHERE s.student_grade_id = g.grade_id
)
FROM assignment2.Students s
WHERE student_id IN (
  SELECT student_id  FROM assignment2.Enrollments
  WHERE course_id = (
    SELECT course_id FROM assignment2.Courses
    WHERE course_name = 'History'
  )
);
```



	ABC name	ABC grade
1	Charlie	A
2	Grace	B

Nabin Thapa - 17 (I)Database Day 2 - Assignment

2. 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 SCHEMA IF NOT EXISTS assignment2;

CREATE TABLE IF NOT EXISTS assignment2.publishers(
  publisher_id INT PRIMARY KEY,
  publisher_name VARCHAR(100),
  country VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS assignment2.books(
  book_id INT PRIMARY KEY,
  title vARCHAR(100),
  genre VARCHAR(50),
  publisher_id INT,
  publication_year DATE,
  FOREIGN KEY(publisher_id) REFERENCES assignment2.publishers(publisher_id)
);

CREATE TABLE IF NOT EXISTS assignment2.customers(
  customer_id INT PRIMARY KEY,
  customer_name VARCHAR(50) NOT NULL,
  email VARCHAR(150) UNIQUE,
  address VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS assignment2.authors(
  author_id INT PRIMARY KEY,
  author_name VARCHAR(50) NOT NULL,
  birth_date DATE,
  nationality VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS assignment2.orders(
  order_id INT PRIMARY KEY,
  order_date DATE DEFAULT CURRENT_DATE,
  customer_id INT,
  total_amount INT DEFAULT 1,
  FOREIGN KEY(customer_id) REFERENCES assignment2.customers(customer_id)
);
```

Nabin Thapa - 17 (I)Database Day 2 - Assignment

```sql
CREATE TABLE IF NOT EXISTS assignment2.book_authors(
  book_id INT,
  author_id INT,
  PRIMARY KEY(book_id, author_id),
  FOREIGN KEY(book_id) REFERENCES assignment2.books(book_id),
  FOREIGN KEY(author_id) REFERENCES assignment2.authors(author_id)
);

CREATE TABLE IF NOT EXISTS assignment2.order_items(
  order_id INT,
  book_id INT,
  PRIMARY KEY(order_id, book_id),
  FOREIGN KEY(book_id) REFERENCES assignment2.books(book_id),
  FOREIGN KEY(order_id) REFERENCES assignment2.orders(order_id)
);
```

ER DIAGRAM: