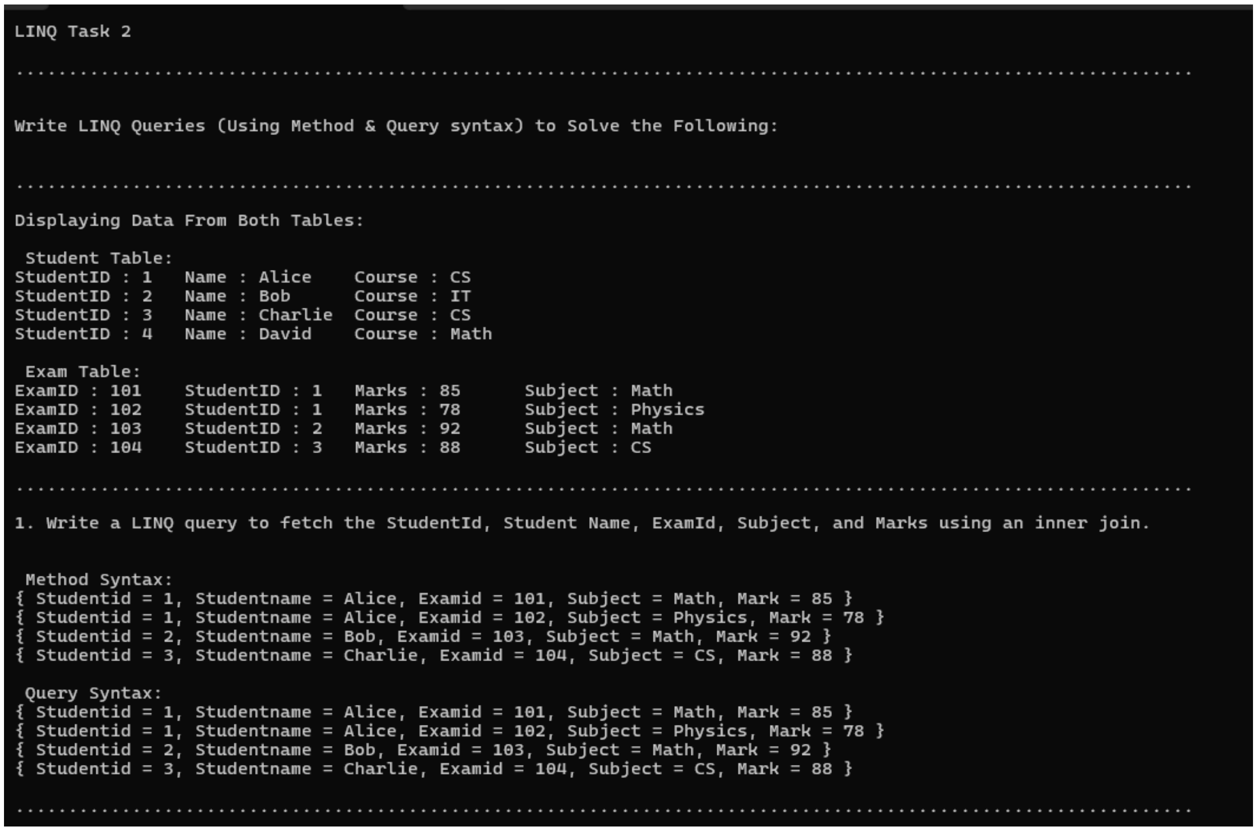
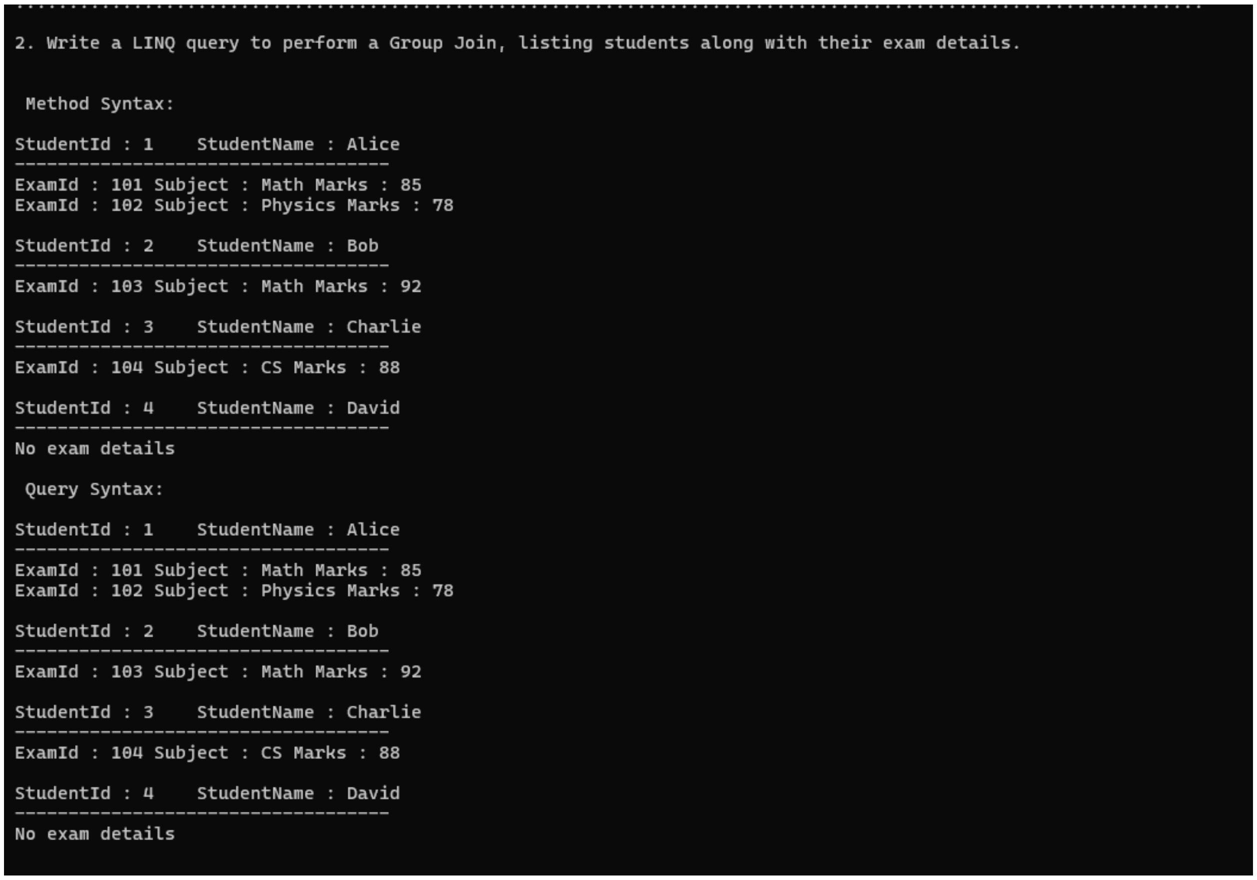
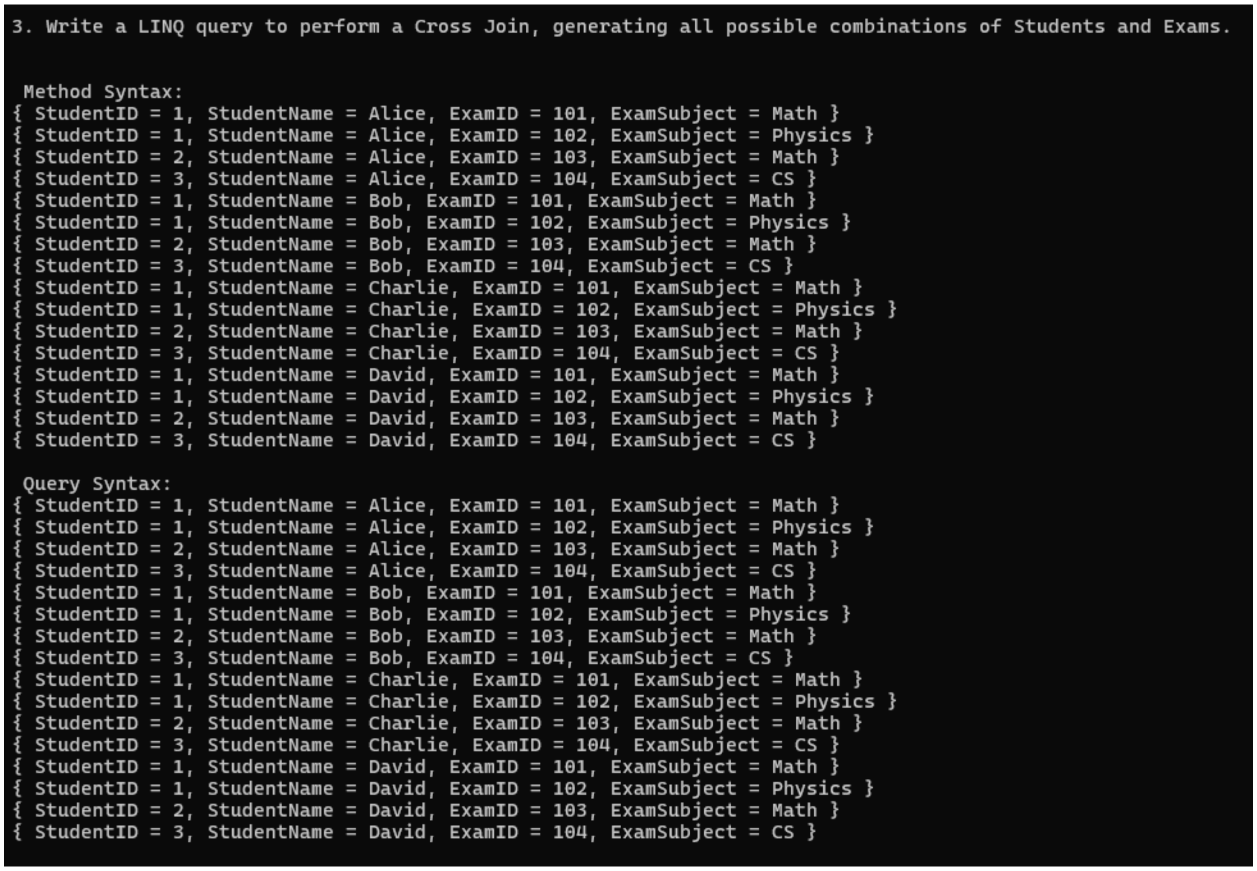


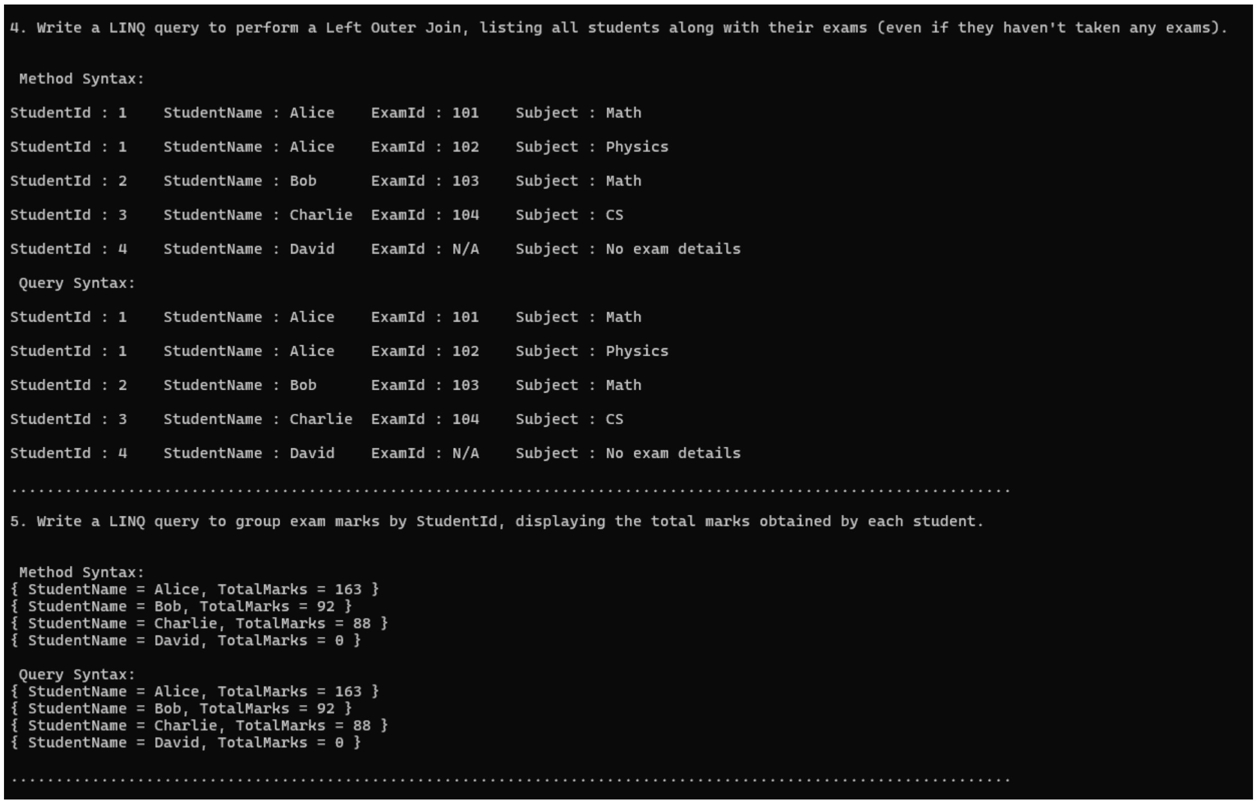
```
LINQ Task 2

..............................................................................................................


Write LINQ Queries (Using Method & Query syntax) to Solve the Following:


..............................................................................................................

Displaying Data From Both Tables:

 Student Table:
StudentID : 1   Name : Alice    Course : CS
StudentID : 2   Name : Bob      Course : IT
StudentID : 3   Name : Charlie  Course : CS
StudentID : 4   Name : David    Course : Math

 Exam Table:
ExamID : 101    StudentID : 1   Marks : 85      Subject : Math
ExamID : 102    StudentID : 1   Marks : 78      Subject : Physics
ExamID : 103    StudentID : 2   Marks : 92      Subject : Math
ExamID : 104    StudentID : 3   Marks : 88      Subject : CS

..............................................................................................................

1. Write a LINQ query to fetch the StudentId, Student Name, ExamId, Subject, and Marks using an inner join.


 Method Syntax:
{ Studentid = 1, Studentname = Alice, Examid = 101, Subject = Math, Mark = 85 }
{ Studentid = 1, Studentname = Alice, Examid = 102, Subject = Physics, Mark = 78 }
{ Studentid = 2, Studentname = Bob, Examid = 103, Subject = Math, Mark = 92 }
{ Studentid = 3, Studentname = Charlie, Examid = 104, Subject = CS, Mark = 88 }

 Query Syntax:
{ Studentid = 1, Studentname = Alice, Examid = 101, Subject = Math, Mark = 85 }
{ Studentid = 1, Studentname = Alice, Examid = 102, Subject = Physics, Mark = 78 }
{ Studentid = 2, Studentname = Bob, Examid = 103, Subject = Math, Mark = 92 }
{ Studentid = 3, Studentname = Charlie, Examid = 104, Subject = CS, Mark = 88 }

..............................................................................................................
```



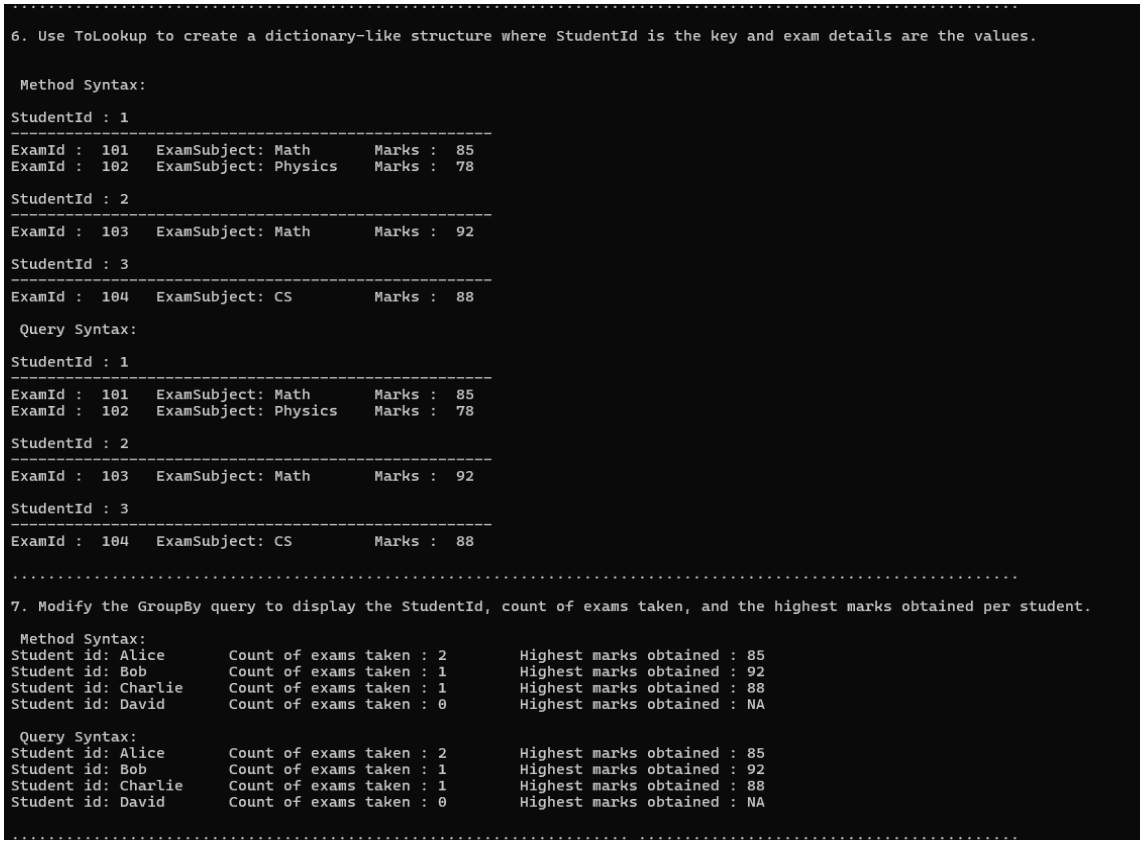
```
..............................................................................................................

2. Write a LINQ query to perform a Group Join, listing students along with their exam details.


 Method Syntax:

StudentId : 1    StudentName : Alice
-----------------------------------
ExamId : 101 Subject : Math Marks : 85
ExamId : 102 Subject : Physics Marks : 78

StudentId : 2    StudentName : Bob
-----------------------------------
ExamId : 103 Subject : Math Marks : 92

StudentId : 3    StudentName : Charlie
-----------------------------------
ExamId : 104 Subject : CS Marks : 88

StudentId : 4    StudentName : David
-----------------------------------
No exam details

 Query Syntax:

StudentId : 1    StudentName : Alice
-----------------------------------
ExamId : 101 Subject : Math Marks : 85
ExamId : 102 Subject : Physics Marks : 78

StudentId : 2    StudentName : Bob
-----------------------------------
ExamId : 103 Subject : Math Marks : 92

StudentId : 3    StudentName : Charlie
-----------------------------------
ExamId : 104 Subject : CS Marks : 88

StudentId : 4    StudentName : David
-----------------------------------
No exam details
```



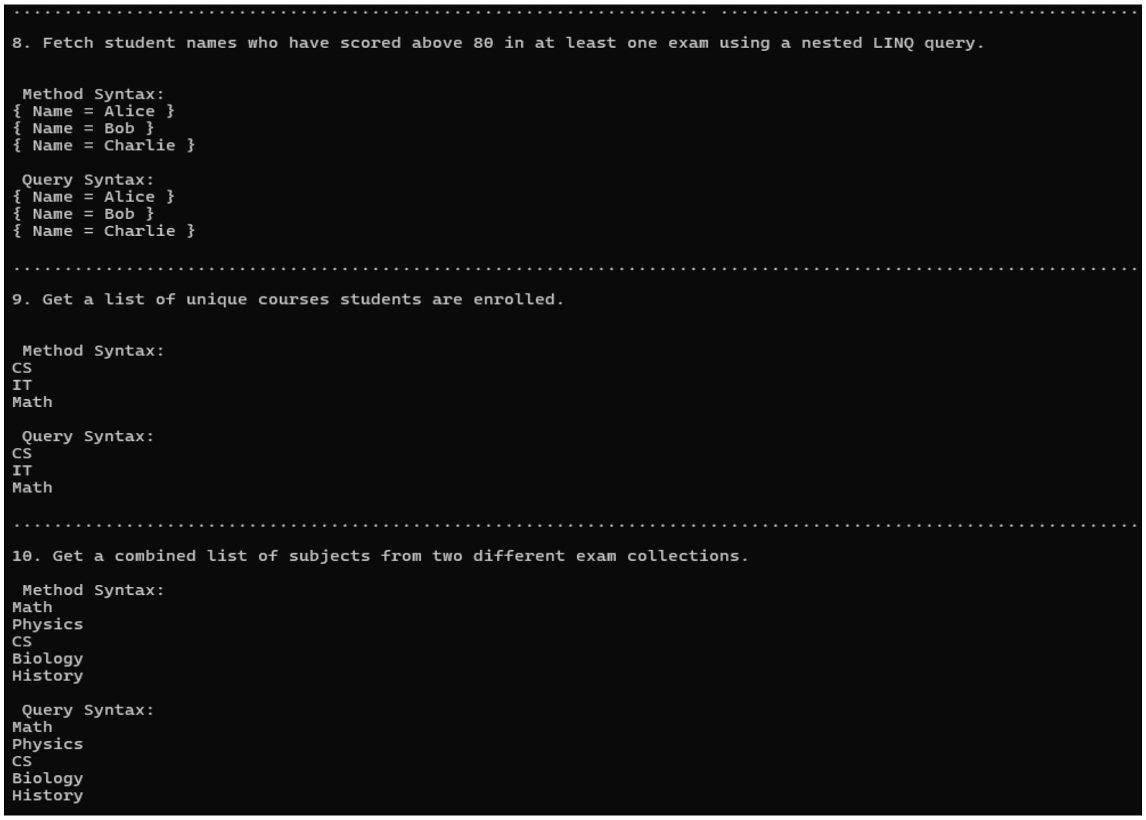
```
3. Write a LINQ query to perform a Cross Join, generating all possible combinations of Students and Exams.


 Method Syntax:
{ StudentID = 1, StudentName = Alice, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = Alice, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = Alice, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = Alice, ExamID = 104, ExamSubject = CS }
{ StudentID = 1, StudentName = Bob, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = Bob, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = Bob, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = Bob, ExamID = 104, ExamSubject = CS }
{ StudentID = 1, StudentName = Charlie, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = Charlie, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = Charlie, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = Charlie, ExamID = 104, ExamSubject = CS }
{ StudentID = 1, StudentName = David, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = David, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = David, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = David, ExamID = 104, ExamSubject = CS }

 Query Syntax:
{ StudentID = 1, StudentName = Alice, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = Alice, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = Alice, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = Alice, ExamID = 104, ExamSubject = CS }
{ StudentID = 1, StudentName = Bob, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = Bob, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = Bob, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = Bob, ExamID = 104, ExamSubject = CS }
{ StudentID = 1, StudentName = Charlie, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = Charlie, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = Charlie, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = Charlie, ExamID = 104, ExamSubject = CS }
{ StudentID = 1, StudentName = David, ExamID = 101, ExamSubject = Math }
{ StudentID = 1, StudentName = David, ExamID = 102, ExamSubject = Physics }
{ StudentID = 2, StudentName = David, ExamID = 103, ExamSubject = Math }
{ StudentID = 3, StudentName = David, ExamID = 104, ExamSubject = CS }
```



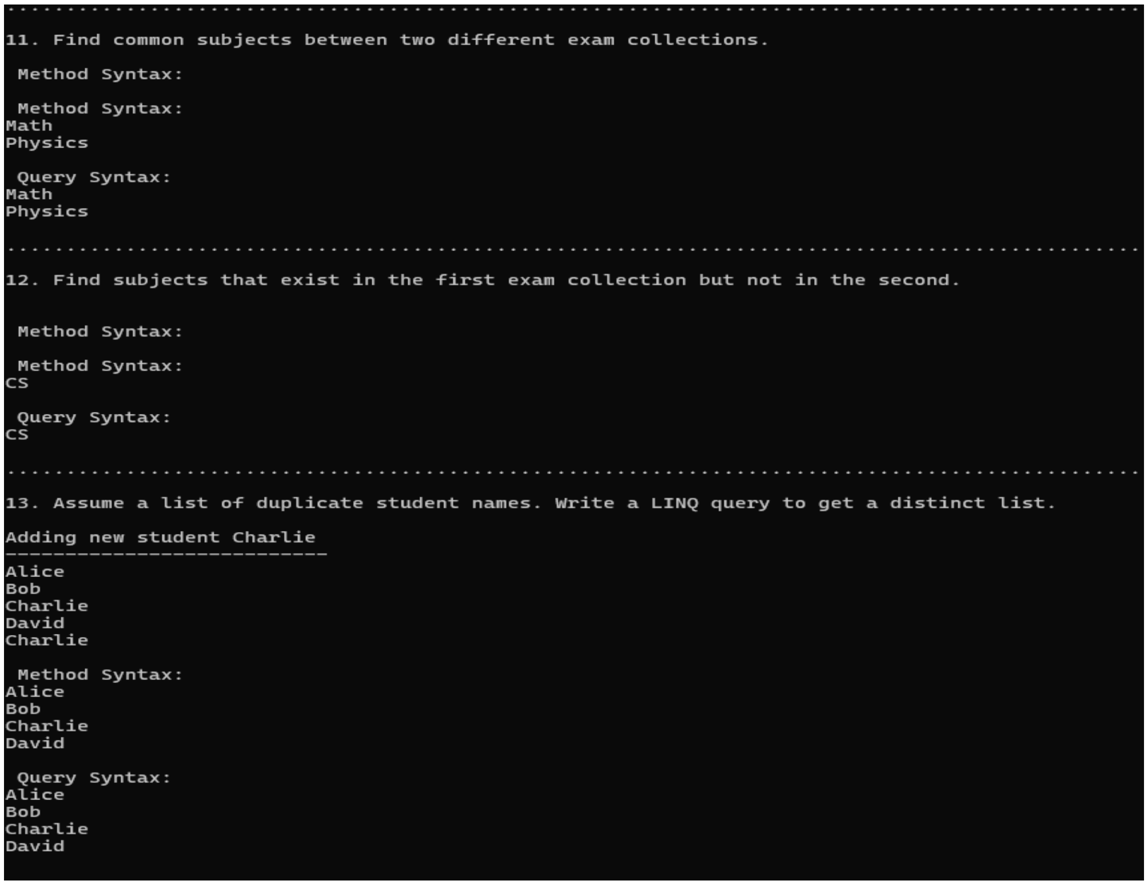
```
4. Write a LINQ query to perform a Left Outer Join, listing all students along with their exams (even if they haven't taken any exams).


 Method Syntax:

StudentId : 1    StudentName : Alice    ExamId : 101    Subject : Math

StudentId : 1    StudentName : Alice    ExamId : 102    Subject : Physics

StudentId : 2    StudentName : Bob      ExamId : 103    Subject : Math

StudentId : 3    StudentName : Charlie  ExamId : 104    Subject : CS

StudentId : 4    StudentName : David    ExamId : N/A    Subject : No exam details

 Query Syntax:

StudentId : 1    StudentName : Alice    ExamId : 101    Subject : Math

StudentId : 1    StudentName : Alice    ExamId : 102    Subject : Physics

StudentId : 2    StudentName : Bob      ExamId : 103    Subject : Math

StudentId : 3    StudentName : Charlie  ExamId : 104    Subject : CS

StudentId : 4    StudentName : David    ExamId : N/A    Subject : No exam details

........................................................................................................................

5. Write a LINQ query to group exam marks by StudentId, displaying the total marks obtained by each student.


 Method Syntax:
{ StudentName = Alice, TotalMarks = 163 }
{ StudentName = Bob, TotalMarks = 92 }
{ StudentName = Charlie, TotalMarks = 88 }
{ StudentName = David, TotalMarks = 0 }

 Query Syntax:
{ StudentName = Alice, TotalMarks = 163 }
{ StudentName = Bob, TotalMarks = 92 }
{ StudentName = Charlie, TotalMarks = 88 }
{ StudentName = David, TotalMarks = 0 }

........................................................................................................................
```



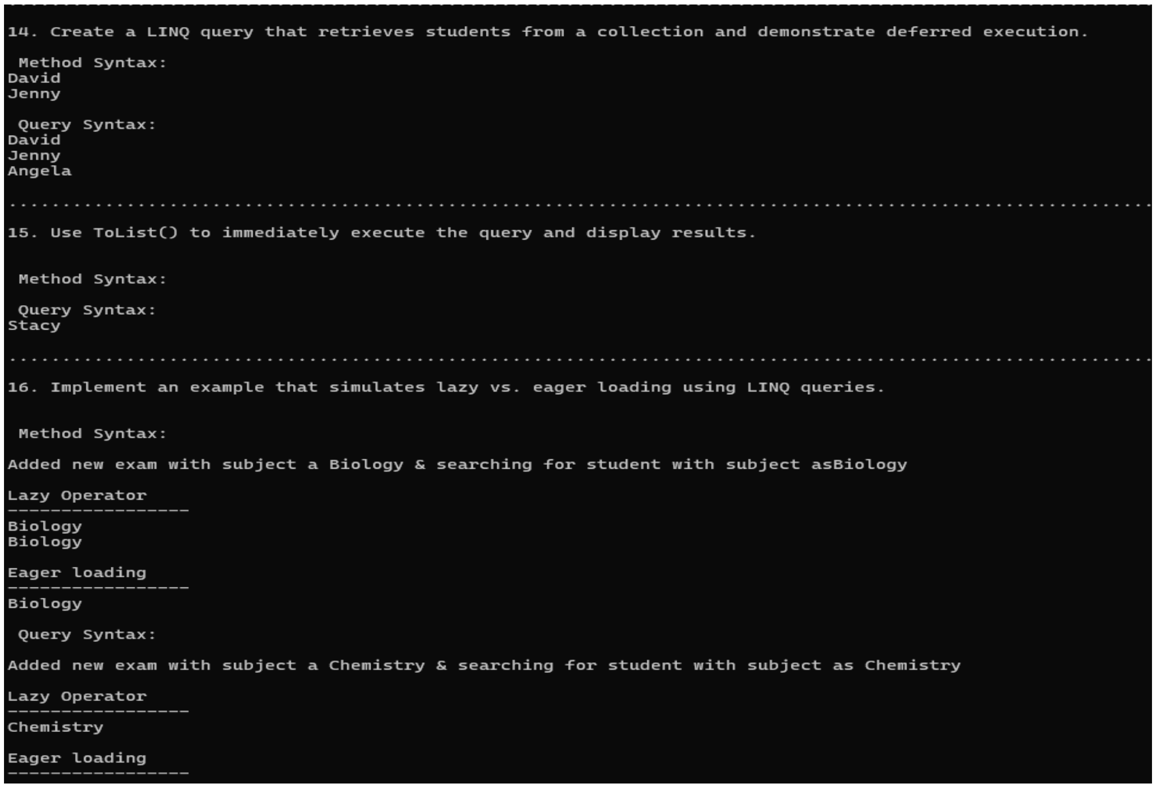
```
.............................................................................................................

6. Use ToLookup to create a dictionary-like structure where StudentId is the key and exam details are the values.


 Method Syntax:

StudentId : 1
----------------------------------------------------
ExamId :  101   ExamSubject: Math       Marks :  85
ExamId :  102   ExamSubject: Physics    Marks :  78

StudentId : 2
----------------------------------------------------
ExamId :  103   ExamSubject: Math       Marks :  92

StudentId : 3
----------------------------------------------------
ExamId :  104   ExamSubject: CS         Marks :  88

 Query Syntax:

StudentId : 1
----------------------------------------------------
ExamId :  101   ExamSubject: Math       Marks :  85
ExamId :  102   ExamSubject: Physics    Marks :  78

StudentId : 2
----------------------------------------------------
ExamId :  103   ExamSubject: Math       Marks :  92

StudentId : 3
----------------------------------------------------
ExamId :  104   ExamSubject: CS         Marks :  88

.............................................................................................................

7. Modify the GroupBy query to display the StudentId, count of exams taken, and the highest marks obtained per student.

 Method Syntax:
Student id: Alice       Count of exams taken : 2        Highest marks obtained : 85
Student id: Bob         Count of exams taken : 1        Highest marks obtained : 92
Student id: Charlie     Count of exams taken : 1        Highest marks obtained : 88
Student id: David       Count of exams taken : 0        Highest marks obtained : NA

 Query Syntax:
Student id: Alice       Count of exams taken : 2        Highest marks obtained : 85
Student id: Bob         Count of exams taken : 1        Highest marks obtained : 92
Student id: Charlie     Count of exams taken : 1        Highest marks obtained : 88
Student id: David       Count of exams taken : 0        Highest marks obtained : NA

.............................................................................................................
```



```
....................................................................................................

8. Fetch student names who have scored above 80 in at least one exam using a nested LINQ query.


 Method Syntax:
{ Name = Alice }
{ Name = Bob }
{ Name = Charlie }

 Query Syntax:
{ Name = Alice }
{ Name = Bob }
{ Name = Charlie }

....................................................................................................

9. Get a list of unique courses students are enrolled.


 Method Syntax:
CS
IT
Math

 Query Syntax:
CS
IT
Math

....................................................................................................

10. Get a combined list of subjects from two different exam collections.

 Method Syntax:
Math
Physics
CS
Biology
History

 Query Syntax:
Math
Physics
CS
Biology
History
```



```
...........................................................................................
11. Find common subjects between two different exam collections.

 Method Syntax:

 Method Syntax:
Math
Physics

 Query Syntax:
Math
Physics

...........................................................................................
12. Find subjects that exist in the first exam collection but not in the second.


 Method Syntax:

 Method Syntax:
CS

 Query Syntax:
CS

...........................................................................................
13. Assume a list of duplicate student names. Write a LINQ query to get a distinct list.

Adding new student Charlie
---------------------------
Alice
Bob
Charlie
David
Charlie

 Method Syntax:
Alice
Bob
Charlie
David

 Query Syntax:
Alice
Bob
Charlie
David
```



```
14. Create a LINQ query that retrieves students from a collection and demonstrate deferred execution.

 Method Syntax:
David
Jenny

 Query Syntax:
David
Jenny
Angela

...........................................................................................
15. Use ToList() to immediately execute the query and display results.


 Method Syntax:

 Query Syntax:
Stacy

...........................................................................................
16. Implement an example that simulates lazy vs. eager loading using LINQ queries.


 Method Syntax:

Added new exam with subject a Biology & searching for student with subject asBiology

Lazy Operator
-----------------
Biology
Biology

Eager loading
-----------------
Biology

 Query Syntax:

Added new exam with subject a Chemistry & searching for student with subject as Chemistry

Lazy Operator
-----------------
Chemistry

Eager loading
-----------------
```