1. What is LINQ Cross Join?

**When combining two data sources (or you can say two collections) using Cross Join, each element in the first data source (i.e., first collection) will be mapped with each and every element in the second data source (i.e., second collection).** So, in simple words, we can say that the cross-join produces the Cartesian Products of the collections or data sources involved in the join. This is also known as a Cartesian product, where if the first collection has m elements and the second has n elements, the result **will have m \* n elements.**

In Cross Join, we don’t require the common key or property to specify the Join Condition. And moreover, there is no filtering of data. Each element of the first collection is paired with each element of the second collection. So, the total number of elements in the result set will be the product of the two data sources involved in the join. If the first data source contains 5 elements and the second data source contains 3 elements, then the resultant sequence will contain (5\*3) 15 elements.

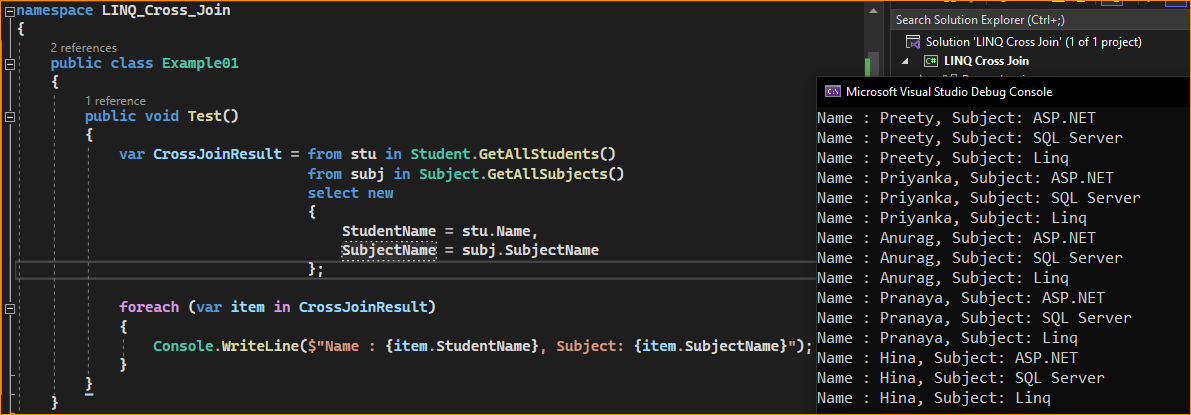
1. Examples to Understand LINQ Cross Join:

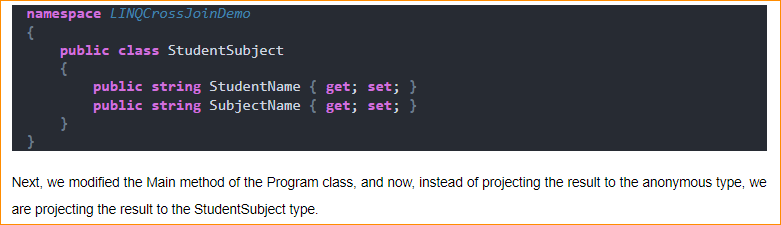
Let us understand How to Implement Cross Join in LINQ with Examples. The following Student and Subject model classes will be used to understand the LINQ Cross Join in C#. Please create a class file named Student.cs and copy and paste the following code. This class has two properties and one method returning a hard-coded collection of students, which will be one of our data sources.

Next, create another class file named Subject.cs and copy and paste the following code. This class has two properties and one method returning a hard-coded collection of subjects, which will be another data source.

1. Example to Understand LINQ Cross Join Using Query Syntax in C#

Let us see how to implement LINQ Cross Join Using Query Syntax in C#. The following example does the Cross Join between the Student with Subject Data Sources using Query Syntax. Here, we don’t need to use any “join” operator, or we don’t need to use the “on” operator, and we don’t need to specify the joining condition, i.e., no common property is required.







Points to Note

* A cross-join creates a combination of all elements from both collections, which can result in a very large result set, especially with large collections.
* Since there’s no join condition (like a matching key), cross joins don’t filter elements; they simply combine everything from both collections.
* Cross joins are useful in scenarios where you need to explore all possible combinations of elements from two sets, such as in product configuration, scheduling, or generating test data.

1. Example to Understand LINQ Cross Join Using Method Syntax in C#

To implement the LINQ Cross Join using Method Syntax, we need to use either the SelectMany() method or the Join() method, as shown in the example below.

