1. LINQ Full Outer Join in C#

The Full Join retrieves all the matching records from both the data sources involved in the join and all the non-matching records from both data sources. The Unmatching data in such cases will take the default value, i.e., NULL.

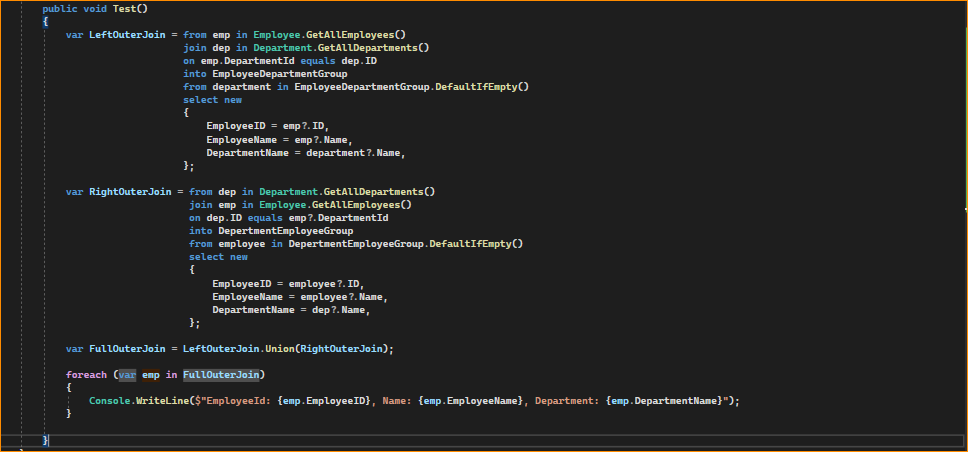
Like the Right Outer Join, LINQ does not support Full Outer Join directly. In LINQ, performing a full outer join between two collections or data sources can be more involved than inner joins or left joins. LINQ does not provide a built-in method for full outer joins like it does for inner and left joins. However, you can achieve the Full Outer Join in LINQ by performing the UNION of Left Outer Join and Right Outer Join.

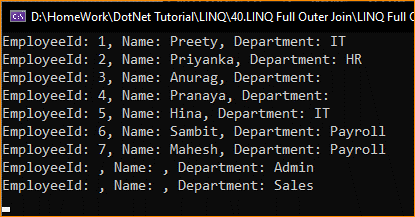
1. Examples to Understand Full Outer Join in LINQ:

Let us understand LINQ Full Join with some Examples using C# Language. We will use the following Employee and Department Data Sources to understand Full Outer Join. So, first, create a class file with the name Employee.cs and then copy and paste the following code into it. The following Employee class has 3 properties, i.e., Id, Name, and DepartmentId, and one method, i.e., GetAllEmployees(), which will return a collection of Employees, which will be one of our data sources.

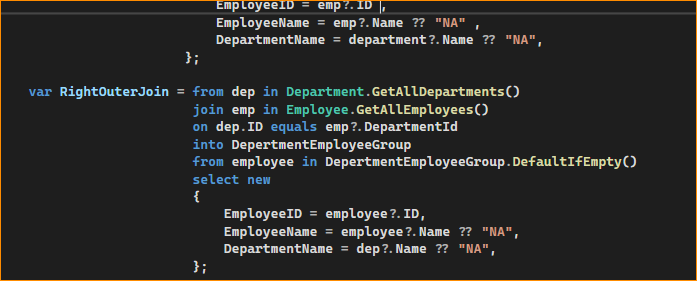
1. LINQ Full Outer Join Example using C#:

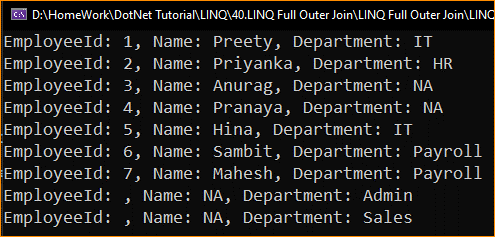
In the example below, we are fetching matchings and non-matching elements from both data sources using Full Join. As Full Outer Join is not directly supported in LINQ, we can still achieve the same by performing the **UNION operation** over the Left Outer Join and Right Outer Join shown in the example below.





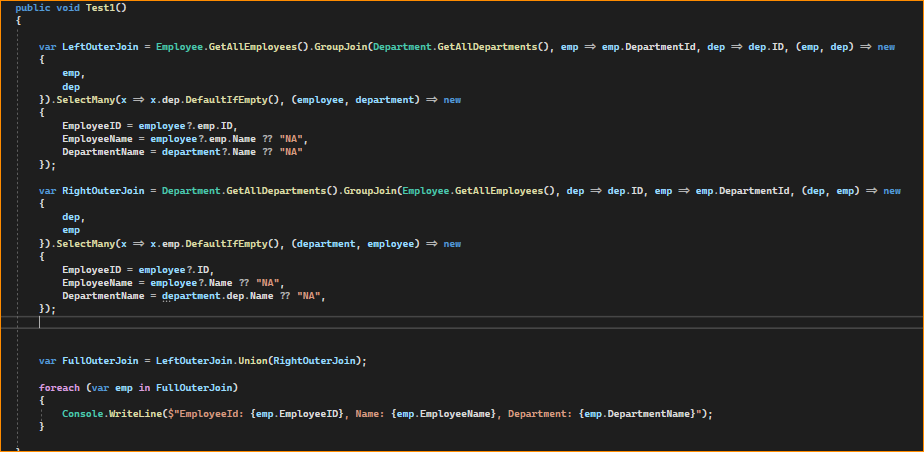
Now, instead of printing the Empty value, we want to store NA in the result set for the Name and Department string values if the value is not available. To do so, modify the Main method of the Program class as follows.

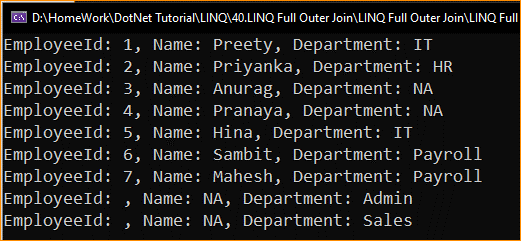




LINQ Full Join Example using Method Syntax:

In the example below, I show how to implement Full Join in LINQ using Method Syntax. The process is the same. First, Left Outer Join, then Right Outer Join, and finally, perform the UNION operation between the Left Outer Join and Right Outer Join, shown in the example below.





1. When to Use LINQ Full Outer Join in C#?

A full outer join is used when you want to retrieve all records from two datasets and match them based on a key. When records in one dataset do not have corresponding records in the other dataset, the result set will still include all records from both datasets, with null values in the columns from the dataset where a match isn’t found. Here are some scenarios where a full outer join is appropriate:

**Merging Data Sources:** When combining data from two different sources, you need to include all data from both, even if they don’t match.

**Data Comparison:** To compare two lists and find items that are exclusive to each list and those that appear in both lists.

**Data Synchronization:** When synchronizing two data sources, you might want to identify items only in one source or the other so you can update both sources to have the same data.

**Reporting:** In reporting scenarios, you need to show the presence or absence of data across two different datasets, such as showing all employees and their corresponding department information, including those without a department and departments without any employees.

**Data Cleaning:** During data cleaning operations, you might want to identify discrepancies between two sets of data so you can address data quality issues.

**Comprehensive Analysis:** When performing a comprehensive analysis that requires examining all possible combinations of data points between two datasets.

A LINQ full outer join is useful when you need to integrate, compare, or combine data from different sources while preserving all elements from both sources, regardless of whether they have matching keys or criteria.