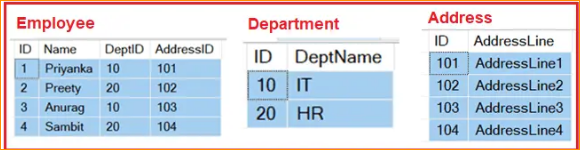
* If you have any experience in database systems like SQL Server, Oracle, MySQL, etc. then you may have familiar with SQL Joins. The LINQ Joins are not different. They are also used to merge the data from two or more data sources (tables or objects) into a single result set based on some common property. Using LINQ Joins, it is also possible to fetch the data based on some conditions.

1. What are LINQ Joins?

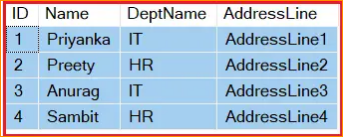
* As per the Microsoft documentation, **the joining of two data sources is the association of objects in one data source with objects that share a common attribute in another data source.**
* We can simplify the above definition as **Join operations are used to fetch the data from two or more data sources based on some common properties present in the data sources.**

1. Why do we need to perform Join Operations?

Let us understand why we need to perform Join Operations with an example. For example, let’s say we have the following three data sources (Employee, Department, and Address).



Now we need to fetch the data from the above three data sources as shown below.



As you can see the above result set contains the data from all three data sources. The most important point that you need to understand is while performing join you need the **common property**.

In this case, the common property between Employee and Department is **Department Id** which is present in both the data sources. In the same line, **Address ID** is the common property between the Employee and Address data sources. So scenarios like this we need to use LINQ Join to fetch the data. How to write the query to fetch the data we will discuss in our next article.

1. What are the Methods Available in LINQ to Perform the Join Operations?

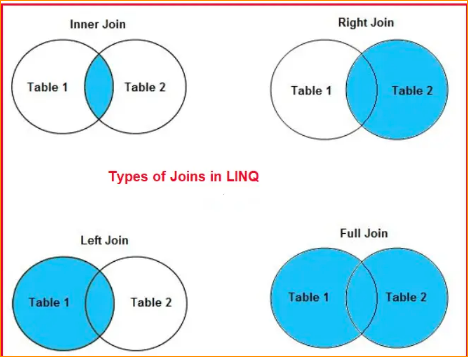
There are two methods available in LINQ to perform Join Operations. They are as follows:

**Join**: This operator is used to join two data sources or collections based on common property and return the data as a single result set.

**GroupJoin**: This operator is also used to join two data sources or collections based on a common key or property but returns the result as a group of sequences.

1. What are the different Types of LINQ Joins Available in C#?

We can perform different types of joins such as Inner Join, Left Join, Right Join, Full Join, and Cross Join in LINQ. For a better understanding, please have a look at the below image.



* **Inner Join**: The Inner Join is used to **return only the matching rows** from both the data sources involved in the **join by removing the non-matching records**.
* **Left Outer Join:** The LEFT OUTER JOIN is used to **retrieve all the matching rows from both the data sources involved in the join** as well as **non-matching rows from the left side data source**. In this case, **the un-matching data will take a null value**.
* **Right Outer Join:** The RIGHT OUTER JOIN is used to **retrieve all the matching rows from both the data sources involved in the join** as well as **non-matching rows from the right-side data source**. In this case, **the un-matching data will take NULL values.**
* **Full Outer Join:** The Full Outer **Join is used to retrieve all the matching rows as well as all the non-matching rows from both the data sources involved in the JOIN. The non-matching data in such cases will take the NULL values.**
* **Cross Join:** In Cross Join, each record of **a data source is joined with each record of the other data source.**