# C# LINQ Types of Execution

In C# LINQ, there are two types of execution: deferred execution and immediate execution. Let's explore each of them in more detail:

**Deferred Execution:** Deferred execution in LINQ postpones the execution of a query until the query results are actually enumerated. It allows for query reuse and retrieval of updated data each time the query results are iterated. When a LINQ query is defined, it creates a query variable that represents the query but does not execute it immediately.

Deferred execution provides flexibility and optimization opportunities because the query is not executed immediately, allowing for additional operations or filtering to be applied before the data is retrieved.

**Immediate Execution:** Immediate execution involves reading the data source and performing the operation immediately when the query is defined or invoked. The query is executed right away, and the results are returned. Examples of immediate execution operators include Count, Sum, Average, First, ToList, etc.

Immediate execution is useful when you want to get the final result immediately and don't need to reuse or iterate over the query multiple times.

Deferred execution in LINQ allows for query reuse, retrieval of updated data, and categorizes operators into streaming and non-streaming based on how they process the elements. Immediate execution, on the other hand, executes the query immediately and returns the final result.

Let's consider a web application that allows users to search for products based on various criteria, such as category, price range, and availability. The application retrieves product data from a database and provides search functionality.

**Example:**

```
public IEnumerable < Product > SearchProducts(string category, decimal
    minPrice, decimal maxPrice, bool availableOnly) {
  var query = dbContext.Products.AsQueryable();

  if (!string.IsNullOrEmpty(category)) {
    query = query.Where(p => p.Category == category);
  }

  query = query.Where(p => p.Price >= minPrice && p.Price <= maxPrice);

  if (availableOnly) {
    query = query.Where(p => p.Availability > 0);
  }

  return query.ToList(); // Deferred execution: Query is executed when
      enumerated
}
public int GetProductCount(string category) {
  object dbContext = null;
  return dbContext.Products.Count(p => p.Category == category); // Immediate
      execution: Query is executed immediately
}
```