# XML Implementation of SQL in .NET

## Initialization Before the Class Constructor

```csharp  
private readonly string \_sqlFilePath;  
private readonly Dictionary<string, string> \_sqlQueries;  
```

## Inside the Constructor

```csharp  
this.\_sqlFilePath = this.\_configuration["SqlSeparationQueries:Assignment"];  
this.\_sqlQueries = LoadSqlQueries();  
```

## Load SQL Queries Method

```csharp  
private Dictionary<string, string> LoadSqlQueries()  
{  
 string absoluteSqlFilePath = Path.GetFullPath(Path.Combine(Directory.GetCurrentDirectory(), "src", \_sqlFilePath));  
 var xml = XElement.Load(absoluteSqlFilePath);  
  
 return xml.Elements("sql")  
 .ToDictionary(e => e.Attribute("name").Value, e => e.Value.Trim());  
}  
```

## Retrieving a Query from the XML

Instead of using the hard-coded SQL query string like:

```csharp  
// string Query = @"abc"  
```

You can access the query defined in the XML file as follows:

```csharp  
var query = \_sqlQueries["Assignment.query"];  
```

## XML File Structure

Create an XML file (`query.xml`) following the structure below:

```xml  
<sql-set>  
 <sql name="Assignment.GetAssignmentById">  
 <![CDATA[  
 -- put your query here  
 ]]>  
 </sql>  
</sql-set>  
```

## Handling Conditional Queries

To handle conditional queries within a method, you can define different queries in the XML and retrieve them as needed:

```csharp  
var queryCondition1 = \_sqlQueries["Assignment.querycondition1"];  
var queryCondition2 = \_sqlQueries["Assignment.querycondition2"];  
string finalQuery = string.Format(queryCondition1, otherFieldsOrPlaceholders);  
```

Also Change the property of the .xml file of copy to output directory to Copy always so that it can take the path during deployment