ADO.NET ile LINQ (Language Integrated Query) kullanarak veritabanı işlemleri yapabiliriz. LINQ, C# diline entegre edilmiş bir sorgu dili ve SQL'e benzer bir yapı sunar. LINQ kullanarak veritabanı sorguları yaparken, LINQ to SQL veya Entity Framework gibi ORM (Object-Relational Mapping) araçları da kullanılabilir. Ancak, doğrudan ADO.NET kullanarak LINQ sorguları yapabiliriz.

ADO.NET ile LINQ Örneği

Aşağıdaki örnekte, doğrudan ADO.NET kullanarak LINQ sorguları yapacağız. Bu örnekte SqlConnection, SqlCommand, SqlDataReader gibi ADO.NET sınıflarını kullanarak LINQ sorgularını yürüteceğiz.

1. LINQ ile Veri Okuma Örneği

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
public class Program
{
   public static void Main()
   {
      // Veritabanı bağlantı dizesi
      string connectionString = "Server=myServerAddress;Database=myDataBase;User
Id=myUsername;Password=myPassword;";
      try
      {
         // Bağlantı oluşturma
         using (SqlConnection connection = new SqlConnection(connectionString))
         {
            // Bağlantıyı açma
```

```
            connection.Open();
            Console.WriteLine("Bağlantı açıldı.");
            // LINQ sorgusu ile veri okuma
            string sqlQuery = "SELECT * FROM Products";
            SqlCommand command = new SqlCommand(sqlQuery, connection);
            // SqlDataReader kullanarak veri okuma
            using (SqlDataReader reader = command.ExecuteReader())
            {
                // LINQ sorgusu ile verileri okuma
                var products = from IDataRecord r in reader
                               select new
                               {
                                   ProductId = Convert.ToInt32(r["ProductId"]),
                                   Name = r["Name"].ToString(),
                                   Price = Convert.ToDecimal(r["Price"])
                               };
                // Verileri ekrana yazdırma
                foreach (var product in products)
                {
                    Console.WriteLine($"ProductId: {product.ProductId}, Name: {product.Name}, Price:
{product.Price}");
                }
            }
            // Bağlantıyı kapatma
            connection.Close();
            Console.WriteLine("Bağlantı kapatıldı.");
```

```
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Hata: " + ex.Message);
    }
  }
}
```

Yukarıdaki örnekte:

*SqlConnection, SqlCommand ve SqlDataReader sınıfları kullanılarak veritabanı bağlantısı ve sorgu yürütme işlemleri gerçekleştirilmiştir.

*SqlCommand ile SQL sorgusu yürütülmüş ve SqlDataReader ile sonuçlar okunmuştur.

*from IDataRecord r in reader kullanarak LINQ sorgusu ile SqlDataReader nesnesinden veri okunmuştur.

*select new { ... } ile her satırı bir anonim nesne olarak döndürülmüş ve foreach döngüsü ile ekrana yazdırılmıştır.

2. LINQ ile Veri Ekleme Örneği

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
public class Program
{
  public static void Main()
  {
    // Veritabanı bağlantı dizesi
    string connectionString = "Server=myServerAddress;Database=myDataBase;User
Id=myUsername;Password=myPassword;";
```

```
        try
        {
            // Bağlantı oluşturma
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                // Bağlantıyı açma
                connection.Open();
                Console.WriteLine("Bağlantı açıldı.");
                // LINQ ile veri ekleme
                string newProductName = "Monitor";
                decimal newProductPrice = 199.99M;
                // LINQ sorgusu ile veri ekleme
                string insertQuery = $"INSERT INTO Products (Name, Price) VALUES ('{newProductName}',
{newProductPrice})";
                SqlCommand command = new SqlCommand(insertQuery, connection);
                int rowsAffected = command.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Console.WriteLine("Ürün başarıyla eklendi.");
                }
                else
                {
                    Console.WriteLine("Ürün eklenirken bir hata oluştu.");
                }
                // Bağlantıyı kapatma
                connection.Close();
```

```
            Console.WriteLine("Bağlantı kapatıldı.");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Hata: " + ex.Message);
    }
  }
}
```

Yukarıdaki örnekte:

*SqlCommand nesnesi ile LINQ sorgusu oluşturulmuş ve ExecuteNonQuery() metodu ile sorgu yürütülmüştür.

*if (rowsAffected > 0) ile sorgunun başarı durumu kontrol edilmiştir.

*Veri ekleme işlemi yapılmıştır.

Bu örnekler, ADO.NET ile LINQ kullanarak SQL sorgularını doğrudan C# kodları içinde nasıl yapabileceğinizi göstermektedir. LINQ to SQL veya Entity Framework gibi ORM araçları da kullanarak daha gelişmiş LINQ sorguları yapabilirsiniz.

İşte bir adet gelişmiş bir LINQ örneği.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
public class Product
{
   public int ProductId { get; set; }
   public string Name { get; set; }
```

```
    public decimal Price { get; set; }
}
public class Program
{
    // Veritabanı bağlantı dizesi
    private static string connectionString = "Server=myServerAddress;Database=myDataBase;User
Id=myUsername;Password=myPassword;";
    public static void Main()
    {
        // Veritabanından ürünleri okuma ve LINQ ile sorgular yapma
        List<Product> products = ReadProducts();
        DisplayProducts(products);
        // Yeni ürün ekleyip tekrar listeleme
        AddProduct("Monitor", 199.99M);
        products = ReadProducts();
        DisplayProducts(products);
    }
    // Products tablosundan veri okuma
    private static List<Product> ReadProducts()
    {
        List<Product> productList = new List<Product>();

        try
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
```

```
        connection.Open();
        Console.WriteLine("Bağlantı açıldı.");
        string sqlQuery = "SELECT * FROM Products";
        SqlCommand command = new SqlCommand(sqlQuery, connection);
        using (SqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                Product product = new Product
                {
                    ProductId = Convert.ToInt32(reader["ProductId"]),
                    Name = reader["Name"].ToString(),
                    Price = Convert.ToDecimal(reader["Price"])
                };
                productList.Add(product);
            }
        }
        connection.Close();
        Console.WriteLine("Bağlantı kapatıldı.");
    }
}
catch (Exception ex)
{
    Console.WriteLine("Hata: " + ex.Message);
}
```

```
        return productList;
    }
    // Ürünleri ekrana yazdırma
    private static void DisplayProducts(List<Product> products)
    {
        Console.WriteLine("Ürünler:");
        foreach (var product in products)
        {
            Console.WriteLine($"ProductId: {product.ProductId}, Name: {product.Name}, Price:
{product.Price}");
        }
        Console.WriteLine();
    }
    // Yeni ürün ekleme
    private static void AddProduct(string name, decimal price)
    {
        try
        {
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                Console.WriteLine("Bağlantı açıldı.");
                string insertQuery = $"INSERT INTO Products (Name, Price) VALUES ('{name}', {price})";
                SqlCommand command = new SqlCommand(insertQuery, connection);
                int rowsAffected = command.ExecuteNonQuery();
                if (rowsAffected > 0)
```

```
            {
                Console.WriteLine("Ürün başarıyla eklendi.");
            }
            else
            {
                Console.WriteLine("Ürün eklenirken bir hata oluştu.");
            }
            connection.Close();
            Console.WriteLine("Bağlantı kapatıldı.");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Hata: " + ex.Message);
    }
  }
}
```

Bu örnekte:

*Product sınıfı, ProductId, Name ve Price özelliklerini içerir.

*ReadProducts metodu, Products tablosundan veri okur ve List<Product> olarak döndürür.

*DisplayProducts metodu, List<Product>'i ekrana yazdırır.

*AddProduct metodu, yeni bir ürün ekler.

Bu örnekte ADO.NET kullanarak LINQ sorgularını doğrudan C# kodlarında kullanarak veri işlemleri yapabilirsiniz. LINQ ile veriye sorgular yaparken, verileri C# nesneleri olarak işleyebilir ve yönetebilirsiniz.