SQL Query Builder Usage Guide

Introduction

The SqlQueryBuilder class provides a fluent interface for building SQL queries dynamically. It supports various SQL operations like SELECT, WHERE, JOIN, GROUP BY, ORDER BY, LIMIT, OFFSET, UNION, and more.

Installation & Setup

Ensure that your project has access to the SqlQueryBuilder class. You can include it in your project as a utility class.

```
// Example of instantiating the SqlQueryBuilder
var queryBuilder = new SqlQueryBuilder();
```

Basic Usage

1. SELECT Statement

```
var query = new SqlQueryBuilder()
    .Select("id", "name")
    .From("users")
    .Build();

Console.WriteLine(query);
// Output: SELECT id, name FROM users;
```

2. Using WHERE Clause

```
var query = new SqlQueryBuilder()
    .Select("id", "name")
    .From("users")
    .Where("age > 18")
    .Where("status = 'active'")
    .Build();
```

3. Ordering Results

```
var query = new SqlQueryBuilder()
    .Select("id", "name")
    .From("users")
    .OrderBy("name", descending: true)
    .Build();
```

4. Limiting & Offsetting Results

```
var query = new SqlQueryBuilder()
    .Select("id", "name")
    .From("users")
    .OrderBy("id")
    .Limit(10)
    .Offset(20)
    .Build();
```

Advanced Usage

5. Distinct Selection

```
var query = new SqlQueryBuilder()
    .Select("name")
    .Distinct()
    .From("users")
    .Build();
```

6. JOIN Operations

INNER JOIN

```
var query = new SqlQueryBuilder()
    .Select("users.id", "users.name", "orders.amount")
    .From("users")
    .Join("orders", "users.id = orders.user_id")
    .Build();
```

LEFT JOIN

```
var query = new SqlQueryBuilder()
    .Select("users.id", "users.name", "orders.amount")
    .From("users")
    .LeftJoin("orders", "users.id = orders.user_id")
    .Build();
```

7. GROUP BY and HAVING

```
var query = new SqlQueryBuilder()
    .Select("category", "COUNT(*) AS total")
    .From("products")
    .GroupBy("category")
    .Having("COUNT(*) > 5")
    .Build();
```

8. UNION Queries

```
var query1 = new SqlQueryBuilder()
    .Select("id", "name")
    .From("users")
    .Where("status = 'active'");

var query2 = new SqlQueryBuilder()
    .Select("id", "name")
    .From("admins")
    .Where("status = 'active'");

var unionQuery = query1.Union(query2).Build();
```

9. Complex Query Example

```
var query = new SqlQueryBuilder()
    .Select("users.id", "users.name", "SUM(orders.amount) AS total_spent")
    .Distinct()
    .From("users")
    .LeftJoin("orders", "users.id = orders.user_id")
    .Where("users.status = 'active'")
    .GroupBy("users.id", "users.name")
    .Having("SUM(orders.amount) > 100")
    .OrderBy("total_spent", descending: true)
    .Limit(10)
    .Offset(5)
    .Build();
```

Conclusion

The SqlQueryBuilder class simplifies SQL query construction in C# applications. It provides a flexible and fluent interface to build dynamic queries without manually concatenating strings.

For further enhancements, consider adding support for INSERT, UPDATE, and DELETE operations!