# **Laravel Query Builder Package Documentation**

**Introduction**

### **Overview**

The **Query Builder Package** is designed to simplify dynamic report generation for developers and non-technical users. It eliminates the need for writing complex SQL queries by providing an intuitive interface to construct queries effortlessly. This package allows users to select tables, apply conditions, join tables, group data, and generate custom reports with minimal effort.  
  
**Simplified Dynamic Report Generation**

The **Query Builder Package** is designed to make report generation easy for both developers and non-technical users. Writing SQL queries manually can be complex and time-consuming, especially when handling multiple tables, conditions, and calculations. This package simplifies the process with an intuitive interface that allows users to build queries without needing advanced database knowledge. By selecting tables, adding filters, applying conditions, and grouping data, users can generate custom reports effortlessly. This improves efficiency, reduces manual coding, and minimizes errors, ensuring quick and accurate data retrieval.

The primary goal of the **Query Builder Package** is to empower developers by providing a robust and efficient tool for generating reports dynamically. Traditional SQL query writing can be time-consuming and complex, especially when dealing with multiple joins, conditions, and aggregations. This package simplifies the process by offering an intuitive interface where developers can construct queries without deep SQL knowledge. It enhances productivity by reducing manual coding efforts, minimizing errors, and enabling faster report generation. With built-in support for selecting tables, applying conditions, joining tables, and grouping results, this package ensures that developers can retrieve and display relevant data efficiently.

### **Purpose of the Query Builder Package**

The **Laravel Query Builder Package** was created to provide developers with a streamlined and flexible way to generate dynamic reports. Instead of manually writing SQL queries, this package enables developers to construct queries using an intuitive interface, saving time and reducing errors. The goal is to enhance productivity by offering a user-friendly solution for building complex reports without requiring deep SQL expertise.

### 

### **Key Benefits**

* **Efficiency**: Reduces manual query writing and speeds up report generation.
* **Ease of Use**: Offers a user-friendly interface requiring minimal SQL knowledge.
* **Accuracy**: Minimizes errors in SQL query construction.
* **Flexibility**: Supports various query configurations, including table selection, joins, conditions, and aggregations.

## **Features**

* **Table Selection**: Choose the main table for querying.
* **Join Tables**: Add optional left joins to include related data.
* **Column Selection**: Specify columns to display in the results.
* **Conditions & Filters**: Apply WHERE clauses using operators and values.
* **Grouping & Aggregations**: Group results with SUM, GROUP\_CONCAT, and other functions.
* **Aliases**: Assign custom labels for grouped data.
* **Query Management**: Save, edit, delete, and execute queries dynamically.
* Support Additional Functions – Include AVG(), MIN(), MAX(), COUNT(), etc.
* Advanced Filtering – Add options such as HAVING and BETWEEN.
* ORDER BY – Implement sorting functionality in ASC/DESC order.
* Limit & Offset – Allow users to control the number of rows retrieved in query results.
* Table & Column Visibility – Configure visibility based on settings.

## **Usage Guide**

### **1. Access the Query Builder**

Navigate to:

http://127.0.0.1:8000/queries

**2. Middleware and Route Prefix**

Modify config/querybuilder.php to set middleware and route prefix:

return [

'middleware' => ['web', 'auth'], // Middleware for all QueryBuilder routes

'access\_route' => 'queries', // Prefix for web routes

];

### **3. Creating a Query**

1. **Select Main Table**: Choose a primary table.
2. **Define Joins (Optional)**: Select related tables for joining.
3. **Choose Columns**: Specify columns to retrieve.
4. **Set Conditions**: Define WHERE clauses.
5. **Apply Grouping**: Use GROUP BY with aggregation functions.
6. **Save the Query**: Store it for future execution.

### **4. Managing Queries**

* **View Queries**: Access saved queries in the query list.
* **Edit Queries**: Modify an existing query configuration.
* **Delete Queries**: Remove unwanted queries.
* **Execute Queries**: Run a saved query and display the results.

## **Screenshots**

### **Add Query Screen**

### 

### 

### **Query List Screen**

### **View Query Screen**

