The code from your initial search form in MS Access demonstrates good logic and structure for filtering and displaying data, but it directly interacts with the underlying table (`mrl\_line\_items`) rather than the view or stored procedures, which can limit flexibility and maintainability, especially as your application grows.

### Key Points and Recommendations:

1. \*\*Direct SQL Queries\*\*: The code constructs SQL queries on the fly, which is effective for filtering but could be enhanced by delegating more of this logic to the PostgreSQL server. This reduces the workload on the Access client and leverages the server's processing power.

2. \*\*Dynamic Filtering\*\*: The `FilterListBox` function effectively builds and applies filters based on user input. This is a good approach but could benefit from server-side logic (e.g., using stored procedures) to handle more complex conditions or optimize performance.

3. \*\*Loading and Clearing Data\*\*: The `btnClear\_Click` and `Form\_Load` methods efficiently reset and load data, but they could be optimized by calling a server-side function or stored procedure that prepares the data before it's sent to Access.

### Recommendations for Improvement:

1. \*\*Use Views or Stored Procedures\*\*: Instead of querying the `mrl\_line\_items` table directly, use a view or stored procedure that encapsulates the query logic. This allows for easier updates and security improvements, as changes to the data retrieval logic can be made centrally on the server.

2. \*\*Enhance Filtering with Stored Procedures\*\*: Move the filtering logic to a stored procedure that takes parameters from the form (e.g., `jcn`, `niin`, etc.). This reduces the need to construct SQL queries in VBA and improves security by minimizing SQL injection risks.

3. \*\*Handling Large Data Sets\*\*: If the result set is large, consider implementing pagination within your stored procedures or views, similar to what we've discussed earlier. This way, only a portion of the data is retrieved at a time, which can greatly improve performance.

4. \*\*Update and Edit Operations\*\*: When moving from the search results to editing records, ensure that edits are made to the underlying tables through stored procedures, not directly to the view. This allows for more controlled and validated updates.

By implementing these changes, you will make your application more robust, scalable, and easier to maintain, while also improving performance by leveraging PostgreSQL’s strengths.