**Contributing to the SQRCT Project**

Thank you for your interest in contributing to the SQRCT (Strategic Quote Recovery & Conversion Tracker) system! Your help is valuable in maintaining and improving this Excel-based tool.

**Getting Started**

1. **Clone the Repository:** Make sure you have a local copy of this repository.
2. git clone <repository-url>
3. cd sqrct-repository
4. **Understand the Structure:** Familiarize yourself with the project structure outlined in the README.md and the system details in docs/ARCHITECTURE.md. Remember that this repository holds the *reference source code* (extracted VBA, M scripts) and documentation, while the operational system runs within multiple Excel (.xlsm/.xlsx) files.
5. **Required Software:** Ensure you have Microsoft Excel (with Power Query and VBA enabled) and Git installed.

**Branching Strategy**

We use a simple Git workflow:

1. **main** branch: Represents the stable, reviewed version of the extracted code and documentation. Direct pushes are discouraged.
2. **Feature/Bugfix Branches:** Create a new branch off main for any new feature development or bug fix (e.g., feature/improve-sync-logging, fix/pq-data-type-error).
3. **Pull Requests (PRs):** Once your changes are ready, push your branch to GitHub and open a Pull Request targeting the main branch.

**Coding Standards**

Consistency helps maintainability, especially with VBA and M code.

**VBA:**

* **Option Explicit**: Must be declared at the top of every module.
* **Variable Naming:** Use meaningful names (e.g., workbookPath instead of wp). Consider a consistent convention (e.g., camelCase likeThis or PascalCase LikeThis). Avoid overly generic names like temp, data, x.
* **Indentation:** Use consistent indentation (e.g., 4 spaces) for readability.
* **Comments:**
  + Use module headers to explain the purpose of the module.
  + Use procedure headers (before Sub or Function) to explain purpose, parameters, and return values.
  + Comment complex logic blocks, explaining the *why*, not just the *what*.
* **Error Handling:** Use On Error GoTo [Label] for structured error handling in main procedures. Avoid excessive use of On Error Resume Next without specific error checking immediately after. Log errors using the provided logging module (Module\_SyncTool\_Logger or equivalent).
* **Modularity:** Break down long procedures into smaller, single-purpose functions or subs. Avoid deeply nested loops/conditionals where possible.

**Power Query (M):**

* **Formatting:** Use the formatting applied by the Power Query editor (consistent indentation).
* **Step Naming:** Use descriptive step names (e.g., #"FilteredExpiredQuotes" instead of #"Filtered Rows").
* **Comments:** Use // for single-line comments or /\* ... \*/ for multi-line comments to explain complex transformations or logic. Add comments explaining the purpose of different sections within a query.
* **Avoid Hardcoding:** Parameterize file paths, server names, or other values where possible, rather than hardcoding them directly in the M script.

**Commit Messages**

Please follow the [Conventional Commits](https://www.conventionalcommits.org/en/v1.0.0/) format for clear and consistent commit history. Examples:

* feat: Add error logging to SyncTool file processor
* fix: Correct data type conversion in CSVQuotes query
* docs: Update ARCHITECTURE diagram for sync flow
* refactor: Improve lookup performance in Module\_Dashboard
* style: Apply consistent indentation to Module\_Utilities

**Managing Code Between Repository and Excel**

This is the most critical part for an Excel-based project using Git:

1. **Exporting Code:** Before committing changes, **manually export** the modified VBA modules (.bas), class modules (.cls), user forms (.frm), and worksheet/workbook code (.cls) from the relevant Excel file into the corresponding location within the src/ directory of this repository. Power Query M scripts should also be copied from the Advanced Editor into their respective .pq or .m files in src/power\_query/.
2. **Importing Code:** When pulling changes from the repository or switching branches, you need to **manually import** the updated code files from the repository back into the VBA editor and Power Query Advanced Editor of the relevant operational Excel file(s).
3. **Repository is King:** The code stored in this Git repository should always be considered the authoritative source of truth. Changes should ideally be made, tested, exported to the repo, and then committed. Avoid having significant uncommitted changes lingering only within the operational Excel files.
4. **Testing:** Thoroughly test any changes within the Excel environment *after* importing code from the repository.

*(Note: While tools exist to help automate VBA source control, they require separate setup.)*

**Pull Request (PR) Process**

1. Ensure your code adheres to the coding standards.
2. Test your changes thoroughly in the Excel environment.
3. Export the updated VBA/M code to the correct files in the repository.
4. Commit your changes with clear messages.
5. Push your feature/bugfix branch to GitHub.
6. Open a Pull Request (PR) against the main branch.
7. Fill out the PR template, explaining the changes made, the reason for them, and how they were tested. Link to any relevant issues.
8. Request reviews from relevant team members.
9. Address any feedback or requested changes.
10. Once approved and checks pass (if any configured), the PR can be merged into main.

**Reporting Issues**

* Use the GitHub Issues tab for this repository.
* Check if a similar issue already exists.
* Use the provided templates (bug\_report.md, feature\_request.md) to provide clear details.

Thank you for contributing!