Contributors

Getting the source code

Download the latest stable MRQL source release from http://www.apache.org/dyn/closer.cgi/incubator/mrql and extract the files. You can get the latest source code using:

```
git clone https://git-wip-us.apache.org/repos/asf/incubator-mrql.git
```

To build MRQL using maven, use `mvn clean install` for Hadoop 2.* (Yarn) or `mvn -Dhadoop1 clean install` for Hadoop 1.*. To validate the installation use `mvn -DskipTests=false clean install`, which runs the queries in `tests/queries` in memory, local Hadoop mode, local Hama mode, Spark mode, and Flink mode.

Eclipse Project File Generation

We are using Maven so you can easily generate project files for our modules.

First you should make sure you have set up your workspace correctly with maven:

```
mvn -Declipse.workspace="/home/user/workspace/" eclipse:configure-workspace
```

Note: "/home/user/workspace/" should be the path to your workspace.

You can check if it was successful in Eclipse via:

```
Select Window > Preferences
Select Java > Build Path > Classpath Variables
```

If you see the **M2_REPO** variable, it worked correctly.

Now run the following commands:

```

% mvn clean install

% mvn eclipse:eclipse
```

You can now import the projects into your eclipse workspace via:

```
File -> Import -> Existing Projects into Workspace -> Choose your workspace as the root directory and import
the mrql-* projects.
```

On Window/Preferences/General/Editors/File Associations, add *.gen and associate it with the Java Editor.

Making Changes

Before you start, send a message to the MRQL developer mailing list, or file a bug report in Jira. Describe your proposed changes and check that they fit in with what others are doing and have planned for the project. Be patient, it may take folks a while to understand your requirements.

- Jira usage guidelines

Modify the source code and add some (very) nice features using your favorite IDE.

But take care about the following points

- All public classes and methods should have informative Javadoc comments.
- Contributions should pass existing unit tests.
- New unit tests should be provided to demonstrate bugs and fixes.

Creating a patch

Check to see what files you have modified with:

```
% git status
```

Add any new files with:

```
% git add any_files_you_created_modified_or_deleted
```

In order to create a patch, type

```
% git diff --cached > /tmp/MRQL-131.patch
```

Submitting a pull request to GitHub

The MRQL apache repository is cloned at the GitHub https://github.com/apache/incubator-mrql, which is connected to the MRQL JIRA issue manager. First, you need to create a GitHub account. Then, you need to fork the MRQL master by pushing the fork button on the MRQL master. Let say that your GitHub username is xxxxx. You should always bring your local fork up-to-date using the following git commands:

```
# clone your local fork (NOT the MRQL master)
git clone https://github.com/xxxxx/incubator-mrql.git current
cd current
# pull all recent changes from the MRQL master
git pull https://github.com/apache/incubator-mrql.git master
# bring the local fork up-to-date
git push origin master
```

Lets say that you want to create a pull request for the MRQL issue `[MRQL-1234] ...issue-title...`. You need to create a new branch of your local fork, say named MRQL-1234

```
# create a new branch inside your directory 'current'
git checkout -b MRQL-1234
# ... do some changes to the files ...
# store changes in the branch
git push origin MRQL-1234
# commit changes to the branch
git commit -a -m '[MRQL-1234] ...issue-title...'
```

Then go to your GitHub MRQL page and do a Pull Request. Use the same title `[MRQL-1234] ...issue-title...`.