

## Tutorial: Aster Data Developer Express – Cube Root Function

Ade\_2\_CubeRootFunction

---

### Introduction:

Aster Data Developer Express (“ADE”) is an integrated development environment for creating, testing and deploying SQL-MapReduce applications in Java. The ADE is delivered through integration with the Eclipse Integrated Development Environment.

### Objectives:

This tutorial explains how to use Aster Data Developer Express (“ADE”) to build, test, and deploy a SQL-MapReduce application that performs a cube root function.

### Prerequisites:

Prior to starting this tutorial you should download and install:

1. [Eclipse version 3.5.1](#)
2. [Aster Data Developer Express Plug-In](#)

### Sample Files:

None.

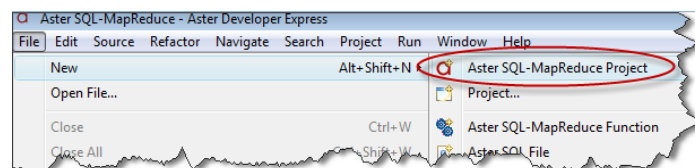
### Part 1: Create Your Aster Project in ADE

The ADE project serves two purposes:

- It's the container that holds your SQL-MapReduce code, related libraries, and tests.
- If you have an *nCluster* implementation (not required for this tutorial) it provides *nCluster* connection details to connect to your database(s).

#### *To create your project:*

1. Run Eclipse and select the command File: New: Aster Project.



2. The first screen of the New Project Wizard appears. Type a Project name (use letters, numerals, and hyphens only)

### Part 2: Create Your SQL-MapReduce Function in ADE

There are two main steps in creating a new function:

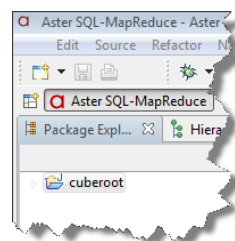
Create the SQL-MapReduce Function Stub Code, below.

- Implement Your `operateOn...()` Method

#### *Create the SQL-MapReduce Function Stub Code*

To create the SQL-MapReduce function stub code:

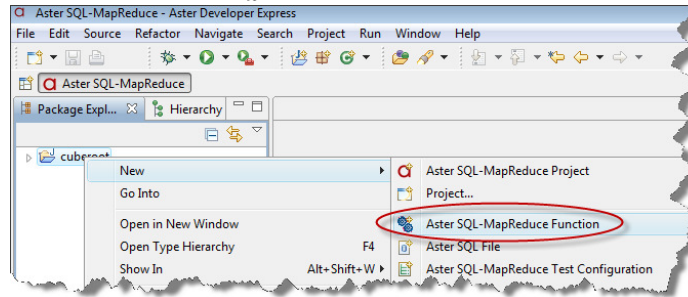
1. In Eclipse's Package Explorer, click on the name of your ADE project.



## Tutorial: Aster Data Developer Express – Cube Root Function

Ade\_2\_CubeRootFunction

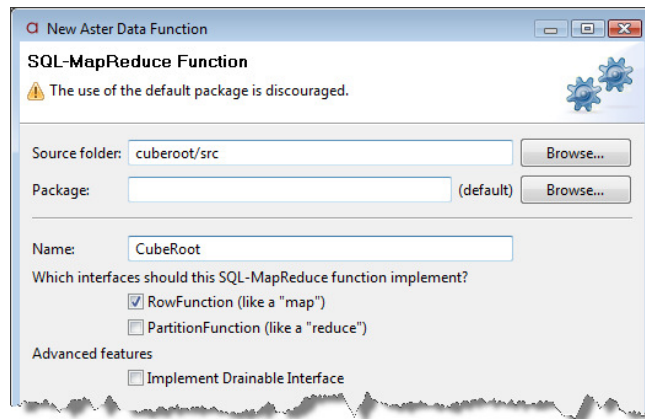
### 2. Choose **New: Aster SQL-MR Function**.



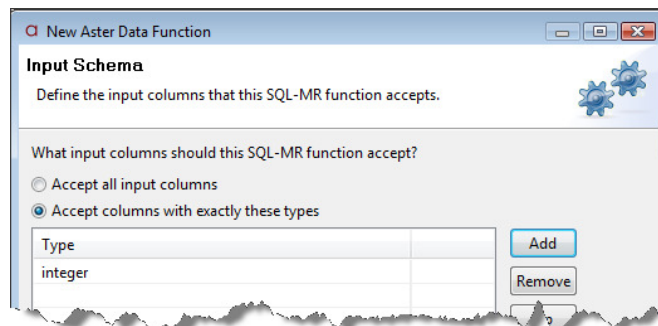
This launches the *ADE Function Wizard*, which will ask you a number of questions about what sort of function you want to create.

### 3. In the first window of the function wizard,

- Type a **Name** for the function (we'll call our example, "CubeRoot").
- Click **RowFunction** or **PartitionFunction** to specify the type of function. See ["Introduction to SQL-MapReduce Functions"](#) for an explanation of row and partition functions.
- Click **Next**.



### 4. In the second wizard window, *Input Schema*, you specify the types of columns your function can take as input.



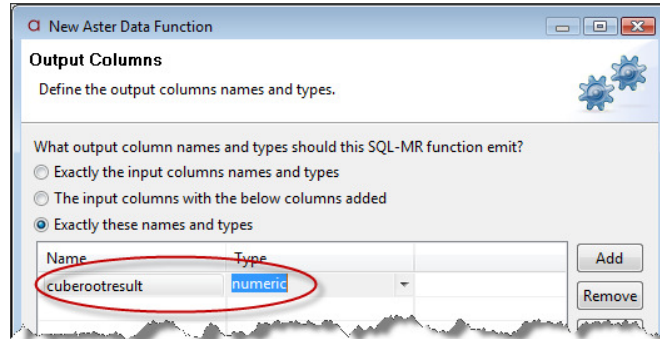
Choose **Accept all input columns** to accept all types (you can add restrictions in your SQL-MapReduce Java code later) or choose **Accept columns with exactly**

## Tutorial: Aster Data Developer Express – Cube Root Function

Ade\_2\_CubeRootFunction

**these types** to specify allowed column types. In this example, since we're just taking the integers from our `to_root` table as input, we'll click **Add** and choose `int`. Click **Next**.

5. In the third wizard screen, *Output Columns*, you choose your output columns and types.

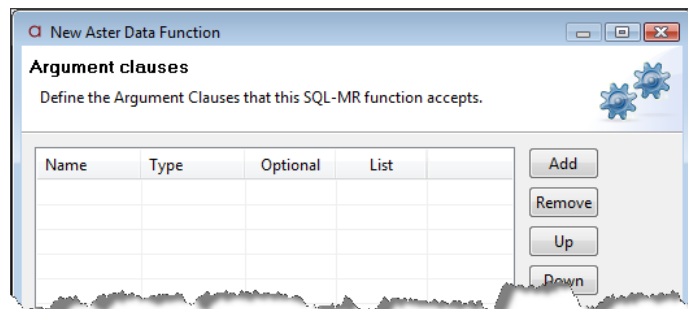


You can choose to:

- return all the input columns, meaning the types and names of the output columns match those of the input (choose **Exactly the input columns' names and types**); or
- return all the input columns *plus additional columns*, (choose **The input columns with the below columns added**); or
- it can return a set of columns you specify (choose **Exactly these names and types**). If you choose this option, the output columns need not match the names or types of the input columns.

In this example, we'll choose the third option, because we just want to return a floating point number that is the cube root of the input value. To do this we check the third checkbox, **Exactly these names and types**, and click the **Add** button. In the row that appears, click on the **Name** field and edit it, and click on the **Type** field and choose the datatype. Click **Add** to add more columns as needed.

6. Click **Next**.
7. In the next wizard screen, *Argument clauses*, you can set up your function to accept arguments from the query that calls the function. These are typically parameter settings that you provide so that the user can specify how the function should operate, in the context of his or her query.

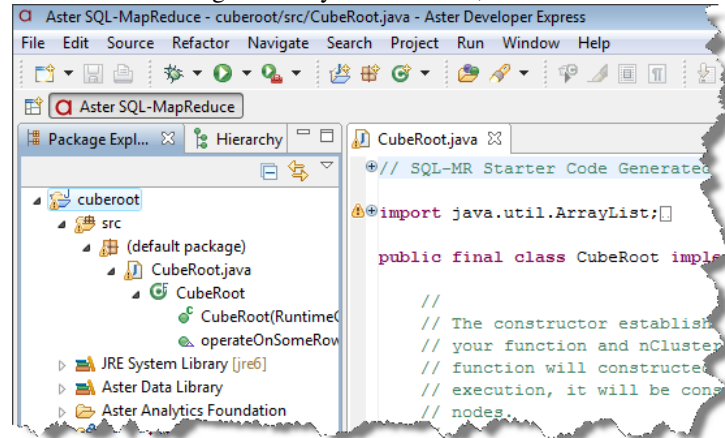


## Tutorial: Aster Data Developer Express – Cube Root Function

Ade\_2\_CubeRootFunction

Click the **Add** button, and click in the cells of the table to give the argument a **Name** and a data **Type**. If the argument is **Optional**, click that cell. To provide a list of acceptable values for the argument, type them in the **List** cell, separated by commas.

8. Click **Finish**. ADE generates your stub code, which is visible in the Package Explorer.



9. In Eclipse, click the save icon or click **File: Save** to save your function stub code.

### *Implement Your operateOn...() Method*

Your stub code contains two important methods:

- The constructor specifies mapping between input and output columns. In most cases you can use the constructor as-is.
- The row function (`operateOnSomeRows()`) or partition function (`operateOnPartition()`) is where you will implement the core logic of your SQL-MapReduce function. Your main task in building an SQL-MapReduce function is to implement this function, which is a row function in our example.

Implement the `operateOnSomeRows()` or `operateOnPartition()` function:

1. Before you implement your function, you may find it useful to create a TestRunner test file in Eclipse that will let you test your code locally (no `nCluster` needed) as you work on it. To create the test file, see Part 3: Test Your SQL-MapReduce Function in ADE for instructions.
2. In Eclipse, open your SQL-MapReduce function source file for editing. In this example, the file is called `CubeRoot.java`.
3. In the row or partition function, find this while loop and implement your function there.

```
while (inputIterator.advanceToNextRow()) {}
```

Do the following:

- a. Delete the `ClientVisibleException`. This is provided as a hint that the implementation has not yet been done. Delete the following lines:

```
throw new ClientVisibleException(  
    "operateOnSomeRows() needs to be implemented!");
```

## Tutorial: Aster Data Developer Express – Cube Root Function

Ade\_2\_CubeRootFunction

---

- b. Where you deleted the exception code, write the implementation of your function.  
For example, here's what our sample **cube root row function** looks like:

```
public void operateOnSomeRows(RowIterator inputIterator,
    RowEmitter outputEmitter) {
    //
    // This method will be called once for each set of rows.
    //
    while (inputIterator.advanceToNextRow()) {
        //
        // Construct output rows here using calls like outputEmitter.addInt(),
        // outputEmitter.addString(), or outputEmitter.addFromRow(). Once
        // finished building the row, call outputEmitter.emitRow().
        //
        int value = inputIterator.getIntAt(0);
        double root = Math.round(Math.pow(value, 1.0/3));

        outputEmitter.addDouble(root);
        outputEmitter.emitRow();
    }
}
```

## Part 3: Test Your SQL-MapReduce Function in ADE

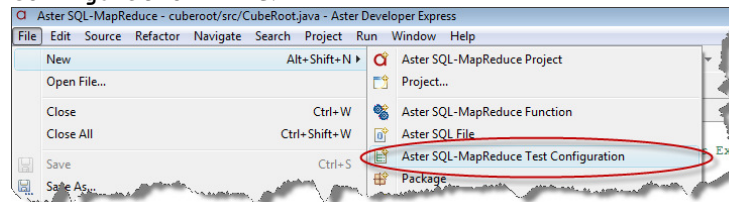
ADE includes a feature called TestRunner that lets you create test file in Eclipse that can test your code locally, without an nCluster connection. There are three main steps in this section:

- Create Your Tests, below.
- Create Your Input Data
- Run Your Test

### Create Your Tests

Create your tests in TestRunner as follows:

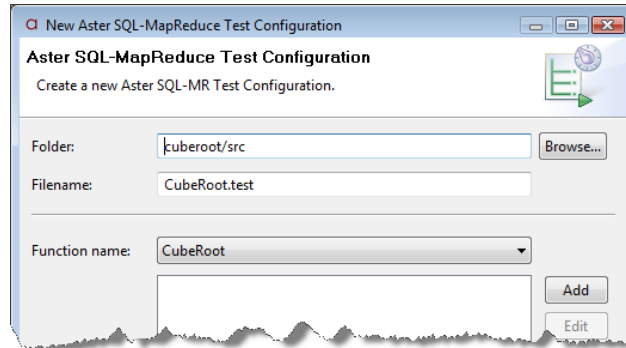
1. Select the menu command **File: New: Aster SQL-MapReduce Test Configuration File**.



2. The New Test Runner wizard appears. In the first window of the wizard, you can typically accept the default values. The **Folder** is where the test will be saved, the **Filename** is the name of the file with test data and settings, and the **Function name** is the name you gave your SQL-MapReduce function (not an individual method, but the class) in In the first window of the function wizard, .

## Tutorial: Aster Data Developer Express – Cube Root Function

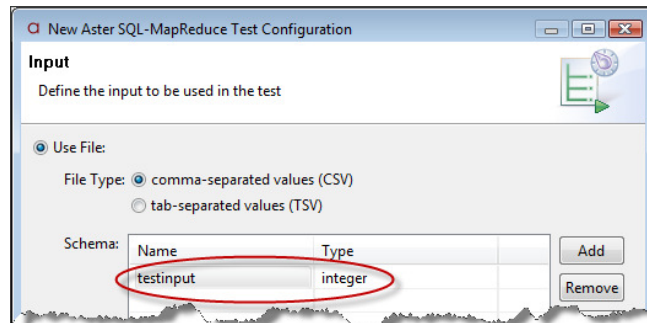
Ade\_2\_CubeRootFunction



If your function takes arguments, click the **Add** button next to the **Argument Clauses** list and add a name/value pair for each argument your test will pass to the function.

Click **Next**.

3. In the second screen of the wizard, *Input*, we define the format and source of the input data the test will use. We can choose to use a static data file, or we can choose to consume the output of another test as our input.



For this example, we'll use a static, csv-formatted data input file. Click **Use File**, and for **File Type**, choose **CSV**.

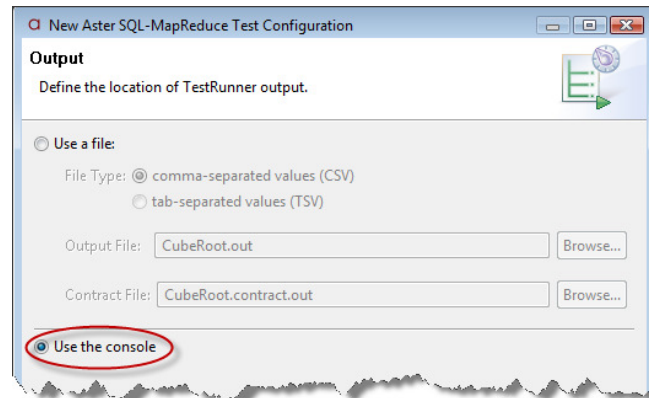
Our example function, cube root, takes just one integer column as input. To add an input field for this single column, click **Add**, and in the Schema table, click on each cell to give the field a **Name** and a data **Type**.

Click **Next**.

4. In the third screen of the wizard, *Output*, we specify the format and destination of the test output. You can direct the output to a file or to the Eclipse console.

## Tutorial: Aster Data Developer Express – Cube Root Function

Ade\_2\_CubeRootFunction



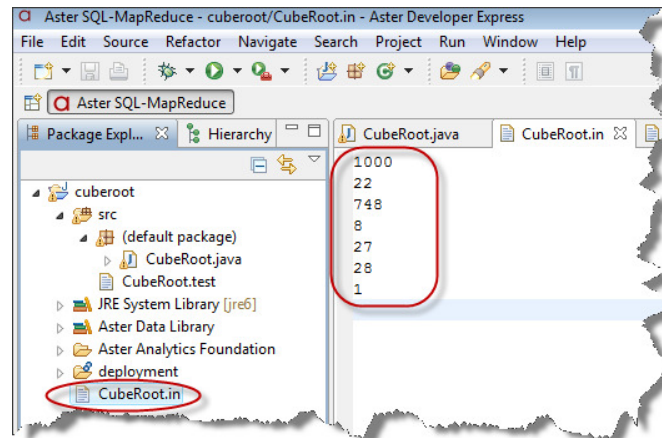
For this example, we'll send the results to the console.

5. Click **Finish**. ADE generates
  - the test file (its name ends in the `.test` suffix) and
  - if specified, the input data file (its name ends in the `.in` suffix).

These files are displayed in the Eclipse workspace.

### Create Your Input Data

If your test uses an input data file, you simply edit that file in Eclipse. In this example, find `CubeRoot.in` at the bottom of the Package Explorer tree, double-click it to edit it, type your test input values in the editing pane, and click **Save**.



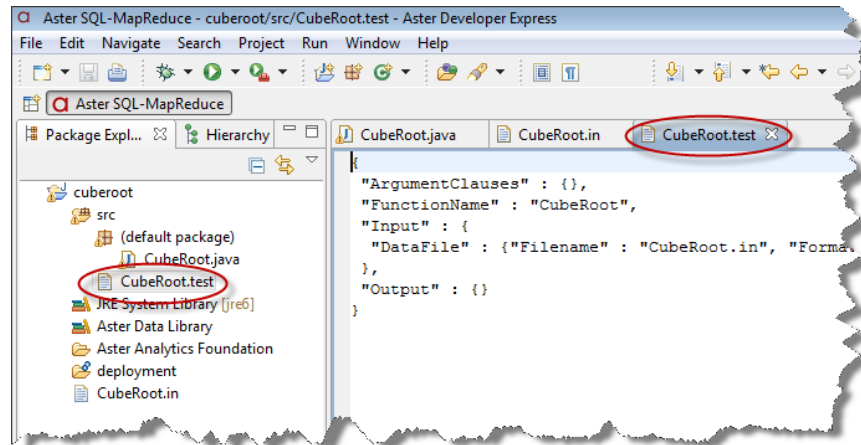
### Run Your Tests

To check your SQL-MapReduce function, run the test in Eclipse as follows.

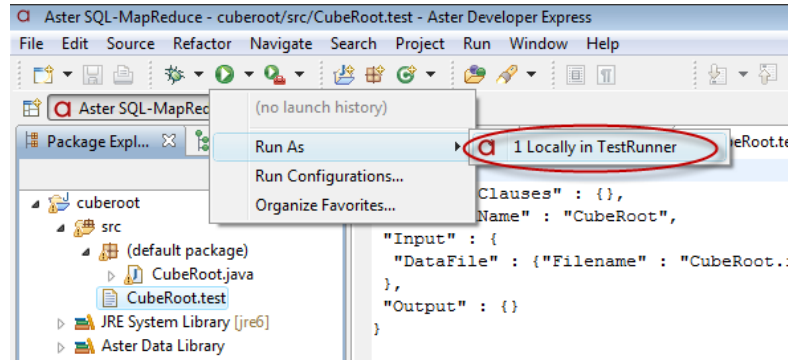
1. Find your test in Eclipse. In this example, we click on `CubeRoot.test` in the `src` part of the Package Explorer tree.

## Tutorial: Aster Data Developer Express – Cube Root Function

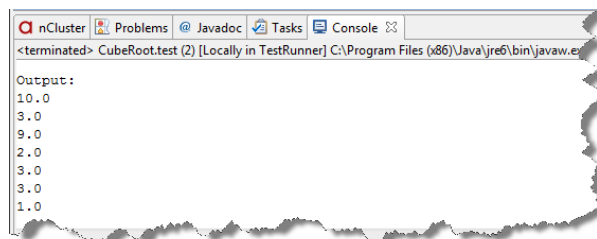
Ade\_2\_CubeRootFunction



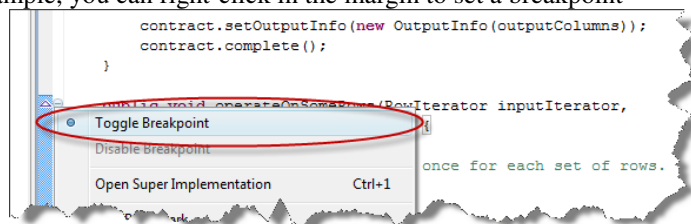
2. In the Run menu, choose **Run As: Locally in TestRunner**.



The results of the test appear in the Console near the bottom of the Eclipse window.



When testing your function locally with a .test file, you can use the debugging tools in Eclipse. For example, you can right-click in the margin to set a breakpoint

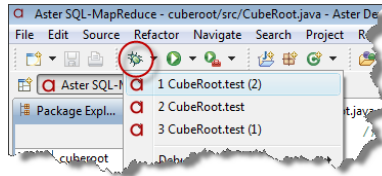


And then debug using your test file.

## Tutorial: Aster Data Developer Express – Cube Root Function

Ade\_2\_CubeRootFunction

---



Running with breakpoints launches the Debug perspective in Eclipse, where you can step through your code to pinpoint errors.

## Summary

At the end of this tutorial the participants understand how to:

- Create your Aster project
- Create your SQL-MapReduce function
- Test your SQL-MapReduce function

## Questions & Clarifications

If you need additional information or have any feedback/comments, please let us know at [info@asterdata.com](mailto:info@asterdata.com).