TestJoin

```java
package query;

import database.Database;
import org.junit.Rule;
import org.junit.Test;

import java.io.File;
import java.io.IOException;
import java.util.*;

import database.DatabaseException;
import testutil.TestUtils;
import databox.BoolDataBox;
import databox.DataBox;
import databox.FloatDataBox;
import databox.IntDataBox;
import databox.StringDataBox;
import databox.Type;
import table.Record;
import table.Schema;

import org.junit.rules.TemporaryFolder;

import static org.junit.Assert.*;

public class TestJoinOperator {

  @Rule
  public TemporaryFolder tempFolder = new TemporaryFolder();

  @Test(timeout=5000)
  public void testOperatorSchema() throws QueryPlanException, DatabaseException,
  IOException {
    TestSourceOperator sourceOperator = new TestSourceOperator();
    File tempDir = tempFolder.newFolder("joinTest");
    Database.Transaction transaction = new
    Database(tempDir.getAbsolutePath()).beginTransaction();
    JoinOperator joinOperator = new SNLJOperator(sourceOperator, sourceOperator,
    "int", "int", transaction);

    List<String> expectedSchemaNames = new ArrayList<String>();
    expectedSchemaNames.add("bool");
    expectedSchemaNames.add("int");
    expectedSchemaNames.add("string");
    expectedSchemaNames.add("float");
    expectedSchemaNames.add("bool");
    expectedSchemaNames.add("int");
    expectedSchemaNames.add("string");
    expectedSchemaNames.add("float");

    List<Type> expectedSchemaTypes = new ArrayList<Type>();
    expectedSchemaTypes.add(Type.boolType());
    expectedSchemaTypes.add(Type.intType());
    expectedSchemaTypes.add(Type.stringType(5));
    expectedSchemaTypes.add(Type.floatType());
    expectedSchemaTypes.add(Type.boolType());
    expectedSchemaTypes.add(Type.intType());
    expectedSchemaTypes.add(Type.stringType(5));
    expectedSchemaTypes.add(Type.floatType());

    Schema expectedSchema = new Schema(expectedSchemaNames, expectedSchemaTypes);

    assertEquals(expectedSchema, joinOperator.getOutputSchema());
  }

  @Test(timeout=5000)
  public void testSimpleJoin() throws QueryPlanException, DatabaseException,
    IOException {
      TestSourceOperator sourceOperator = new TestSourceOperator();
      File tempDir = tempFolder.newFolder("joinTest");
      Database.Transaction transaction = new
      Database(tempDir.getAbsolutePath()).beginTransaction();
      JoinOperator joinOperator = new SNLJOperator(sourceOperator, sourceOperator,
      "int", "int", transaction);

      Iterator<Record> outputIterator = joinOperator.iterator();
      int numRecords = 0;

      List<DataBox> expectedRecordValues = new ArrayList<DataBox>();
      expectedRecordValues.add(new BoolDataBox(true));
      expectedRecordValues.add(new IntDataBox(1));
      expectedRecordValues.add(new StringDataBox("abcde", 5));
      expectedRecordValues.add(new FloatDataBox(1.2f));
      expectedRecordValues.add(new BoolDataBox(true));
      expectedRecordValues.add(new IntDataBox(1));
      expectedRecordValues.add(new StringDataBox("abcde", 5));
      expectedRecordValues.add(new FloatDataBox(1.2f));
      Record expectedRecord = new Record(expectedRecordValues);


      while (outputIterator.hasNext()) {
        assertEquals(expectedRecord, outputIterator.next());
        numRecords++;
      }

      assertEquals(100*100, numRecords);
    }

    @Test(timeout=5000)
    public void testEmptyJoin() throws QueryPlanException, DatabaseException,
    IOException {
      TestSourceOperator leftSourceOperator = new TestSourceOperator();

      List<Integer> values = new ArrayList<Integer>();
      TestSourceOperator rightSourceOperator =
      TestUtils.createTestSourceOperatorWithInts(values);
      File tempDir = tempFolder.newFolder("joinTest");
      Database.Transaction transaction = new
      Database(tempDir.getAbsolutePath()).beginTransaction();
      JoinOperator joinOperator = new SNLJOperator(leftSourceOperator,
      rightSourceOperator, "int", "int", transaction);
      Iterator<Record> outputIterator = joinOperator.iterator();

      assertFalse(outputIterator.hasNext());
    }

  }
```