

SQL File Evaluation (SQLFE) Tool

Usage

1. The SQLFE software is available for download through GitHub at:
<https://github.com/wagnerpj42/sql-file-evaluation>. Additional files, such as sample SQL submission files, a sample assignment properties file, and this documentation, are also available at this GitHub location.
2. Installation instructions are in the [Documentation-Install](#) document.
3. PRE-USAGE: INSTRUCTOR
 - a. A database must be set up in a supported database management system (DBMS), currently Oracle, MySQL 5.x, or MySQL 8.0, with the tables and sample data that support any queries in the assignment that you're giving. This database must be reachable in terms of network connectivity from the system running the SQLFE application.
 - i. All necessary database connector libraries must be added to the project, as discussed in the Documentation-Install document.
 - b. You must edit an Assignment Properties file (default name: assignmentProperties) with the specifications for each question in the SQL assignment, along with each question's specified answer, and specified tests for that question. Two examples of an Assignment Properties file can be found in the assignmentProperties-Oracle and assignmentProperties-MySQL files that are at the top level of the SQLFE distribution. A given question can be identified by an integer followed by a period (e.g. 1., 9., or 23.) or an integer followed by a letter followed by a period (e.g. 1a., 9b., 23c.), with either of the above possibilities optionally followed by a dash and integer (specifying a variant) before the period (e.g. 1-1., 1-2, 9a-1, 9a-2., 9a-3) if there are multiple possibilities for the question's answers (see section 5c below). There are two types of SQL tests that can be specified for each question answer:
 - i. Condition Tests – begin with COND, compare the submitted query using a condition built into the test (e.g. counting the number of SELECT keywords) and using an expression (e.g. ">= 2") to evaluate that test by the criteria specified in the expression. For certain COND tests (e.g. CondCompiles), the expression is empty (""), as the entire test condition is coded into the test method itself.
 - ii. Test Tests – begin with TEST, compare the submitted query to the instructor-specified query or queries in the assignment properties file. Such tests might compare the query text (TestQueryStringEqual), the result set generated by the query (e.g. TestResultSetEqualContent for equality regardless of column order), or other aspects of the query.
 - c. There are over thirty-five different SQL tests that you can use in specifying the weighted testing in your Assignment Properties file, with more detailed

information to be found in the [Documentation-SQLTests](#) document.

- d. Additional discussion of how to best configure your assignmentProperties file to help SQLFE evaluate submitted SQL queries as you prefer is given below.
- e. If you wish to run the JUnit tests on SQLFE, you must edit the default constructor in the file AbstractTest.java in the sqlfe.sqltests package and enter the type of data access object you want to use (currently, defaults to OracleDataAccessObject, but MySQL5xDataAccessObject and MySQL80DataAccessObject are also available), and enter the DBMS host, DBMS system id, DBMS user, and DBMS password information so that the test structure can access your database.

4. PRE-USAGE: STUDENTS

- a. SQLFE requires a certain file format to allow for proper parsing. An example template file named *StudentTemplate-LabTest2019* is included in the *setup* folder. A similar template file should be distributed to each student for an assignment. SQLFE uses a special comment marker (-- -- , which is the normal two dash SQL one line comment marker followed by a space and two more dashes) to indicate instructor comments in the template. This allows separate parsing of student comments which should use either the standard single-line SQL comment marker (--) or the standard multi-line SQL comment marker (/* to start, */ to end).
 - i. Note: currently, multi-line student comments marked by /* and */ must be on their own lines, though leading or trailing spaces are allowed.
- b. Students should be informed of the structure of the template file and the required structure of their answers and comments for the assignment. A sample instruction file that we used for the assignment involving the sample student submission files above and named "*LabTest Instructions 2019*" is included in the *setup* folder.

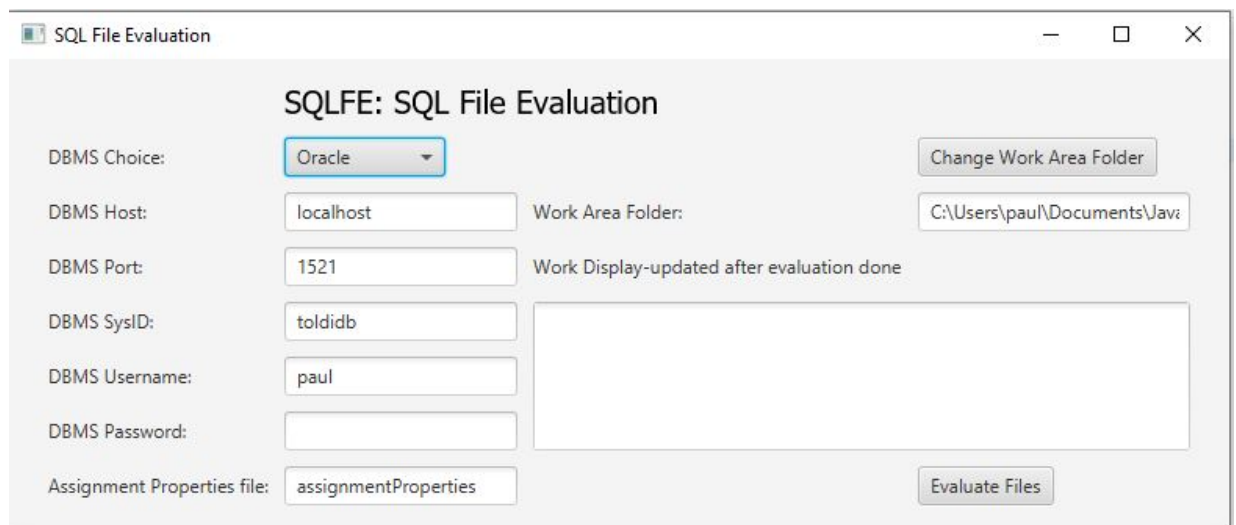
5. TESTING WITH SAMPLE FILES

- a. Sixty-six sample student submission files are provided in the *files-sample-Oracle* and *files-sample-MySQL* folders, and two sample assignmentProperties files (*assignmentProperties-Oracle* and *assignmentProperties-MySQL*) are provided at the top level as well. For testing, you must copy the sample submission files from one of the two above mentioned sample folders to the *files* folder depending on the DBMS you want to use. NOTE: you should also delete the placeholder samplefile.sql placeholder file. You can also copy one of the two sample assignmentProperties files depending on your DBMS to the default name *assignmentProperties*, or you can use one of the two full assignmentProperties file names directly in the GUI. After database setup as noted above and running the appropriate *BankingScript.sql* script from the *setup* folder in your database environment, you can test SQLFE and see the resulting evaluations in the *evaluations* folder by running the application as

described below.

6. USAGE

- a. The SQLFE application can be executed either
 - i. By running the available executable JAR file. This execution requires an available Java 8 JRE.
 - ii. Through an Integrated Development Environment (IDE). SQLFE has been tested and run on the Windows 10 operating system under the Eclipse and IntelliJ IDEA IDE environments. Under an IDE, SQLFE is run by opening the Main.java file in the sqlfe/general package and choosing Run/Run As/Java Application (in Eclipse) or Run/Run (in IntelliJ IDEA).
- b. In either execution case, you will see the following GUI window:

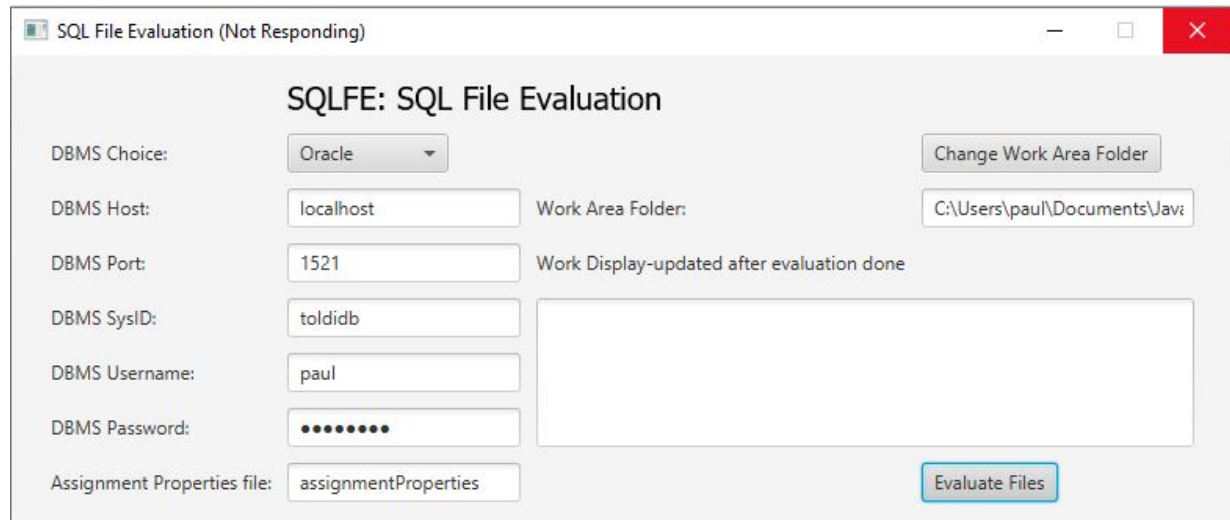


The screenshot shows a window titled "SQL File Evaluation" with a subtitle "SQLFE: SQL File Evaluation". The window contains several input fields and buttons. On the left, there are labels for "DBMS Choice:", "DBMS Host:", "DBMS Port:", "DBMS SysID:", "DBMS Username:", "DBMS Password:", and "Assignment Properties file:". The corresponding input fields are: a dropdown menu for "DBMS Choice" set to "Oracle", a text box for "DBMS Host" with "localhost", a text box for "DBMS Port" with "1521", a text box for "DBMS SysID" with "toldidb", a text box for "DBMS Username" with "paul", an empty text box for "DBMS Password", and a text box for "Assignment Properties file" with "assignmentProperties". On the right, there is a "Change Work Area Folder" button, a text box for "Work Area Folder" with "C:\Users\paul\Documents\Java", and a "Evaluate Files" button. A label "Work Display-updated after evaluation done" is positioned above a large empty rectangular area.

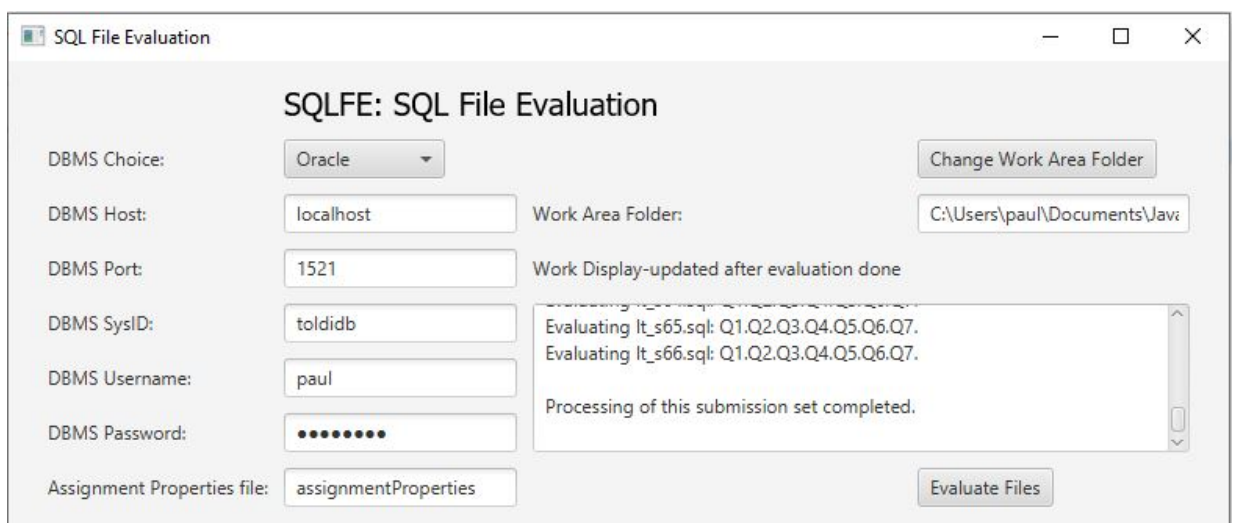
- c. You need to make the following choices:
 - i. DBMS Choice: Choose from the drop-down list between Oracle, MySQL 5.x, and MySQL 8.0, for the type of database that you've set up for your queries.
 - ii. DB Host: Enter the host name or ip address for the DBMS instance that supports your queries. The hostname may be in dot notation (e.g. dbsys.cs.uwec.edu) or localhost.
 - iii. DB Port: Enter the port name that your particular DBMS is using. Oracle normally defaults to 1521, and MySQL to 3306, but if you have multiple servers on a host and have configured your DBMS instance to use a different port, you may specify that here.
 - iv. DB System/ID: Enter the system id for the actual database instance that you're accessing. In Oracle, this may also be known as the SID or Service Name. In MySQL, this may be known as the schema.
 - v. DB Username: Enter the username for the account containing the data that you want to run the queries against.

- vi. DB Password: Enter the password corresponding to the username previously entered.
- vii. Assignment Properties file: Enter the name of the properties file representing the assignment specifications. This defaults to assignmentProperties (no file extension), but can be set to whatever you've named the file having this information, or for testing use one of the two provided sample files.
- viii. Work Area Folder: this is the top level folder for all SQLFE evaluation work. A default folder is provided by operating system, but you should choose a work folder specific to your evaluation situation and one that will not interfere with other work, as new directories may be created and files may be altered. This main work area folder must hold the following:
 - 1. The assignment properties file previously discussed.
 - 2. A folder named "*files*" that contains each submission file that you want to evaluate. For testing, you should copy all of the sample submission files from one of the two sample folders (Oracle or MySQL) to this *files* folder depending on which DBMS you are using.
 - 3. You may also create a folder named "*evaluations*" in this work area folder that will, after processing, contain:
 - An individual submission evaluation file for each submitted file (named based on the original file with a .OUT suffix),
 - A summary grades file (named AAA_grade_summary.out) that holds all of the total scores plus each individual question score for that student, which can be used for transferring scores to a grading database or spreadsheet as well as for review of student achievement,
 - A parsing problems file (named AAA_parsing_problems.out) noting which student submissions could not be parsed with an indication of the line area causing the problem, and
 - a student comments file (named AAA_student_comments.out) showing student answer comments that can be reviewed to help understand their answer.
 - 4. However, the "*evaluations*" folder will be automatically created under the main work area folder if SQLFE does not find it when SQLFE is first run.
- d. The first time you run SQLFE a configuration file (config.properties) is created that holds the GUI choices that you've made, with the exception of the database password, and those choices are retrieved as defaults for subsequent runs of SQLFE. You can always change any values directly in the text boxes.
- e. If all of the choices are specified, you can click the Start Evaluation button, and the SQL File Evaluation process will begin. Currently, the evaluation status is

not shown in real-time, but only appears after all evaluation work is completed. So, while the SQLFE system is running, you'll see the Evaluate Files button you just pressed highlighted in blue, but the work display area will remain empty:



- f. Once evaluation is completed, information on the parsing and evaluation of each submission parsing is displayed to the Work Display area. This can indicate submissions that were not able to be parsed, or questions that were not able to be evaluated.



- g. If a file cannot be parsed (possibly due to circumstances such as students not following the template format and rules or stray binary characters in the template file, which has happened to us when students downloaded the template file), an entry is made in the AAA_parse_problems.out file in the Evaluation folder, and no evaluation can/will be done, leading to a score of 0 for that student file. You can go back later to examine the student files that

had parse problems, manually fix those issues, and rerun SQLFE to generate scores for such files.

- h. When the parsing and evaluation is complete, as indicated in the Work Display window, you can close the SQLFE application and examine the evaluation documents in the evaluations folder.

7. HINTS/SUGGESTIONS FOR OPTIMAL USAGE

- a. **Tuning Using Multiple Runs:** The first run of SQLFE may not give you results that you think are correct. We find that when evaluating a new assignment, we make repeated runs of SQLFE, changing the assignment properties file tests and weightings based on examination of the results. Following are some notes on key concepts and details that may be useful.
- b. **Varying Tests and Weighting:** Different instructors have different standards for correctness. Normally we use multiple tests to give partial credit and to look for the presence of particular keywords that indicate understanding of the question, in addition to testing for result set content correctness. Below are some test configurations in the assignment properties file that show different instructor grading philosophies.
 - i. Evaluate a query only on correctness of the result set.
 - 1. Test set: `TestResultSetEqualContent 100`
 - ii. Evaluate a query on correctness of result set (80%), but allow partial credit (up to 20%) for basic query correctness (SELECT and FROM keywords present, query compiles regardless of result set generated).
 - 1. Test set: `CondCompiles 10 ""`
 `CondBasicContent 10 ""`
 `TestResultSetEqualContent 80`
 - iii. Evaluate a query on correctness of result set (80%), but allow partial (up to 20%) credit for basic correctness (SELECT and FROM keywords present, query compiles regardless of result set generated), but required that a nested subquery be used (requiring at least two select statements, deduction of 40% if not followed).
 - 1. Test set: `CondCompiles 10 ""`
 `CondBasicContent 10 ""`
 `CondSelectCount 40 ">= 2"`
 `TestResultSetEqualContent 40`
 - iv. Evaluate a query on correctness of result set (80%), but allow partial (up to 20%) credit for basic correctness (SELECT and FROM keywords present, query compiles regardless of result set generated), require results are sorted in ascending order (10% credit).
 - 1. Test set: `CondCompiles 10 ""`
 `CondBasicContent 10 ""`
 `CondOrderByCount 10 ">= 1"`
 `TestResultSetEqualContent 70`

- iv. Of course, many different combinations of tests can be used, though the weights for one test set must always add up to 100.

Multiple Possible Answer Specifications: We also find that we may want to specify multiple possible answers to the same question. This can occur because there are multiple ways of implementing a good answer using different keywords (e.g. a solution using a join vs. a solution using a nested subquery, or a solution using MINUS vs. a solution using NOT IN). This can also occur because a written question is vague, and different interpretations may lead to different answer queries and result sets. To support these situations, SQLFE allows you to specify multiple answers to one query within the answer properties file. Each solution must have a question identifier (e.g. 1 or 1a or 1b), with each answer option having a dash and digit (e.g. -1, -2, etc.) added after the main question identifier and before the ending period (overall example for this situation: we could have two answer solution variants for question 1a, that would be identified as 1a-1. and 1a-2. in the assignmentProperties file). Each of the multiple solutions can have different test sets for evaluation. A submitted answer query is compared to each of the possible solutions and the best result is awarded. An example follows:

```
6-1. (17 points)
SELECT DISTINCT CustID, FName, LName
FROM Customer C
JOIN Account A ON (C.CustID = A.Customer)
MINUS
SELECT CustID, FName, LName
FROM Customer C
JOIN Account A ON (C.CustID = A.Customer)
WHERE A.AccStatus = 'Active'
UNION
SELECT CustID, FName, LName
FROM Customer
WHERE NOT EXISTS
  (SELECT *
   FROM Account
    WHERE Customer = CustID);
```

```
CondCompiles 10 ""
CondBasicContent 15 ""
CondWhereCount 15 " >= 2"
CondTableCount 15 " >= 2"
CondSubselectDepthCount 10 " >= 2"
TestResultSetEqualContent 35
```

6-2. (17 points)

```
SELECT DISTINCT CustID, FName, LName
FROM Customer
WHERE Custid NOT IN
    (SELECT Customer
     FROM Account
     WHERE AccStatus = 'Active');
```

```
CondCompiles 10 ""
CondBasicContent 15 ""
CondWhereCount 15 " >= 1"
CondSelectCount 15 " >= 2"
TestColumnCount 10
TestResultSetEqualContent 35
```

8. Known Issues

- a. Counting keywords – the current count utilities do not always catch instances of a keyword that are not properly used in the SQL query as a keyword. This can happen for several reasons, including:
 - i. A keyword is embedded in a string literal
 1. Example: counting FROM in query SELECT * FROM Employees WHERE ename = 'FROMAGE';
 - ii. A keyword is embedded in a field or table name
 1. Example: counting IN in query SELECT ID, INSIDE_TEMP FROM Measurements;
 - iii. A keyword is embedded in another keyword (here MIN in MINUS)
 1. Example: SELECT MIN(SALARY) FROM Employees MINUS SELECT MIN(SALARY) FROM NewEmployees
 - iv. Mitigation of issue: When using tests with keyword counts, use tests with >= rather than = to ascertain at least minimum number of keyword instances exist