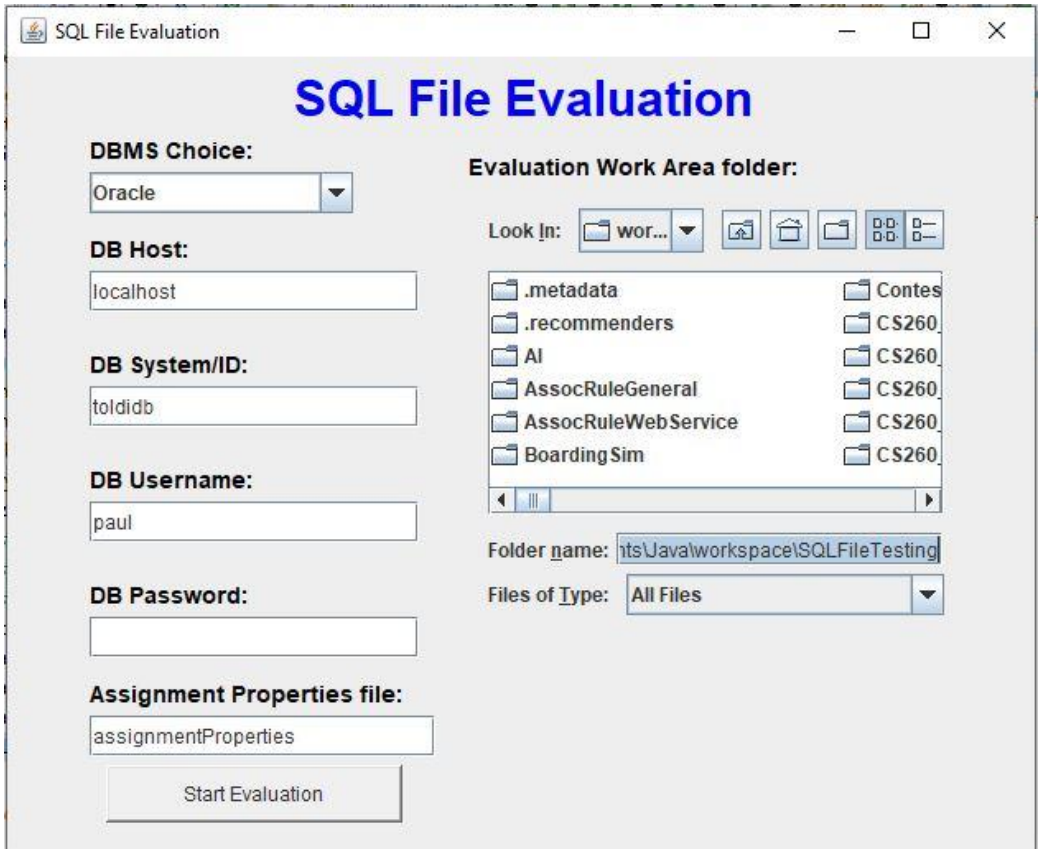
SQL File Evaluation (SQLFE) Tool
Usage (in IDE)

1. The SQLFE software, for now available only as Java source code to be loaded into an integrated development environment (IDE) such as Eclipse or IntelliJ IDEA, 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
    a. A database must be set up in a supported database management system (DBMS), currently Oracle or MySQL, 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 docoument.
    b. You must edit an Assignment Properties file (default name: assignmentProperties) with the specifications for each question in the SQL assignment. An example can be found in the assignmentProperties file that is 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 a number followed by a letter followed by a period (e.g. 1a., 9b., 23c.). There are two types of SQL tests:
        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.

    There are over thirty-five different SQL tests that you can use, with more detailed information found in the Documentation-SQLTests document.
    Additional discussion of how to best configure your assignmentProperties file to help SQLFE evaluate submitted SQL queries as you would is below.

4. USAGE
    a. SQLFE currently can only be executed 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.
    b. 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).
    c. You will see the following GUI window:



    d. You need to make the following choices:
        i. DBMS Choice: Choose from the drop-down list between Oracle and MySQL, 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 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.

iv. DB Username: Enter the username for the account containing the data that you want to run the queries against.
v. DB Password: Enter the password corresponding to the username previously entered.
vi. 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.
vii. Evaluation Work Area Folder: this is the top level folder for all SQLFE evaluation work. This 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.
    3. A folder named "evaluations" that will contain a submission evaluation file for each submitted file, as well as a summary grades file and a student comments file that can be reviewed.

e. 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 retrieve as defaults for subsequent runs of SQLFE. You can always change any values directly in the text boxes.
    i. NOTE: there is currently a bug in the Evaluation Work Area Folder saving and retrieval, in that any location retrieved is actually one level above the main folder you originally specified. At this point, it is necessary to go into the folder chooser and find your desired evaluation folder in the chooser, and double-click on that folder to select it.
f. If all of the choices are specified, you can click the Start Evaluation button, and you should see the SQL File Evaluation process begin.
g. Information on the progress of the evaluation is displayed to the IDE console, showing which submission is being worked on and which question is currently being evaluated. Any SQL errors are also sent to the console to help the instructor see possible problems with student-submitted queries.
h. When the evaluation is complete, as indicated in the console window, you can close the SQLFE application and examine the evaluation documents in the evaluations folder.

5. HINTS/SUGGESTIONS FOR OPTIMAL USAGE
    a. 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 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. 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 combination of tests can be used, though the weights must add up to 100.

c. **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 implanting 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, with each option having a dash and digit (e.g. -1, -2). 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 " >= 1"
CondTableCount 15 " >= 2"
TestColumnCount 10
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
```