JDBC exercise  
1. Connect database to Java program  
2. Program to create database table using Java  
3. Program to insert, update, delete & select records  
4. Program to perform join on two tables  
5. Program to delete record from database  
6. Program to execute batch of SQL statements  
7. Program to insert date in mysql database  
8. Program to insert & read image  
9. Program to commit & rollback  
10. Program to execute SQL select query

**11. Database Access From a Programming Language**

line separator

This lab introduces you to database access from a programming language such as Java or C#. A sample set of exercises can be found here; although phrased using Java/JDBC, the exercise can be done using other languages, OBDC or ADO.NET APIs  
  
In this assignment, you will write a java program which takes any of the following commands as command line argument (not as input), and executes them using JDBC:

1. insert into relationname value1 value2 value3 ..  
     
   Inserts a tuple into the specified relation with the specified values; make sure you use a prepared statement; test it with input containing single quotes (if you run from command line, enclose those in double quotes so linux doesn't interpret the quotes) you can assume that all values are strings for simplicity
2. select from relationname  
     
   Prints all tuples from the specified relation. For the purpose of this assignment, you should assume that relationname is one of "instructor", "student" or "takes", and not use database metadata features to print the tuples.
3. select from relationname where "condition"  
     
   Executes a query with the specified condition. Note the use of double quotes so that the condition comes as a single command line parameter. Again assume relationname is one of those from the previous feature. So this is really a small addition to the code for the previous feature, do NOT make a separate copy of the code.
4. select from relationname1 relationname  
     
   Displays result of natural join of the two relations. This time, use the resultset metadata feature to display all values from the query result; don't worry about displaying column names, that is optional, we only care about values being displayed.

Note: In netbeans, to pass parameters to a program, you can right click on the project, choose configuration > customize. Or run this as a standalone java program instead of using netbeans (after you develop it using netbeans). Use the parameter argv to the main function, which provides you an array of Strings, with each word above part of one string. Use argv.length to figure out how many parameters are present.

For 0 extra credit: If you finish the above early and are bored, try adding column names to the select from relationname option; assume from is not a valid column name. You get no extra marks, except that if you goof up an earlier part, you may still get full marks on the assignment if you do this right (it's very unlikely you will do this part if you goof up an earlier part, anyway)

12. Create a Java project in your Eclipse workspace called jdbc-exercise-001. Place all your java source files for this exercise under the jdbc-exercise-001 project src folder in your Eclipse workspace.  
  
   
  
  
EXERCISE 1: MOVIE DATABASE  
  
Using the techniques and Java JDBC APIs define a movie database application that allows a user to look up information about a movie and provide a review for a movie.  Your application should provide the following features:  
  
   
  
•     List all movies that are rated a particular rating (Excellent, Good, Neutral, Poor, Very Poor) entered by the user.  
  
•     All a user to post a review for a particular movie.  A review consists of review text and a star rating with  
  
o 5 stars = Excellent  
  
o 4 stars = Good  
  
o 3 stars = Neutral  
  
o 2 stars = Poor  
  
o 1 star = Very Poor  
  
•         List all reviews for a movie and include the average star rating for the movie  
  
   
  
Your solution should read the database connection information (jdbc driver and connection url).     
  
CREATE TABLE MOVIE (MOVIE\_ID int not null primary key GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), MOVIE\_NAME varchar(100), RATED varchar(10));  
  
   
  
CREATE TABLE MOVIE\_REVIEW (REVIEW\_ID int not null primary key GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1), MOVIE\_ID int, REVIEW varchar(500), STARS int);  
  
INSERT INTO MOVIE(MOVIE\_NAME, RATED) values('Divergent', 'PG-13');  
  
   
  
INSERT INTO MOVIE(MOVIE\_NAME, RATED) values('Muppets Most Wanted', 'PG'); INSERT INTO MOVIE(MOVIE\_NAME, RATED) values('The LEGO Movie', 'PG'); INSERT INTO MOVIE(MOVIE\_NAME, RATED) values('Saving Mr. Banks', 'PG-13'); INSERT INTO MOVIE(MOVIE\_NAME, RATED) values('Her', 'R');