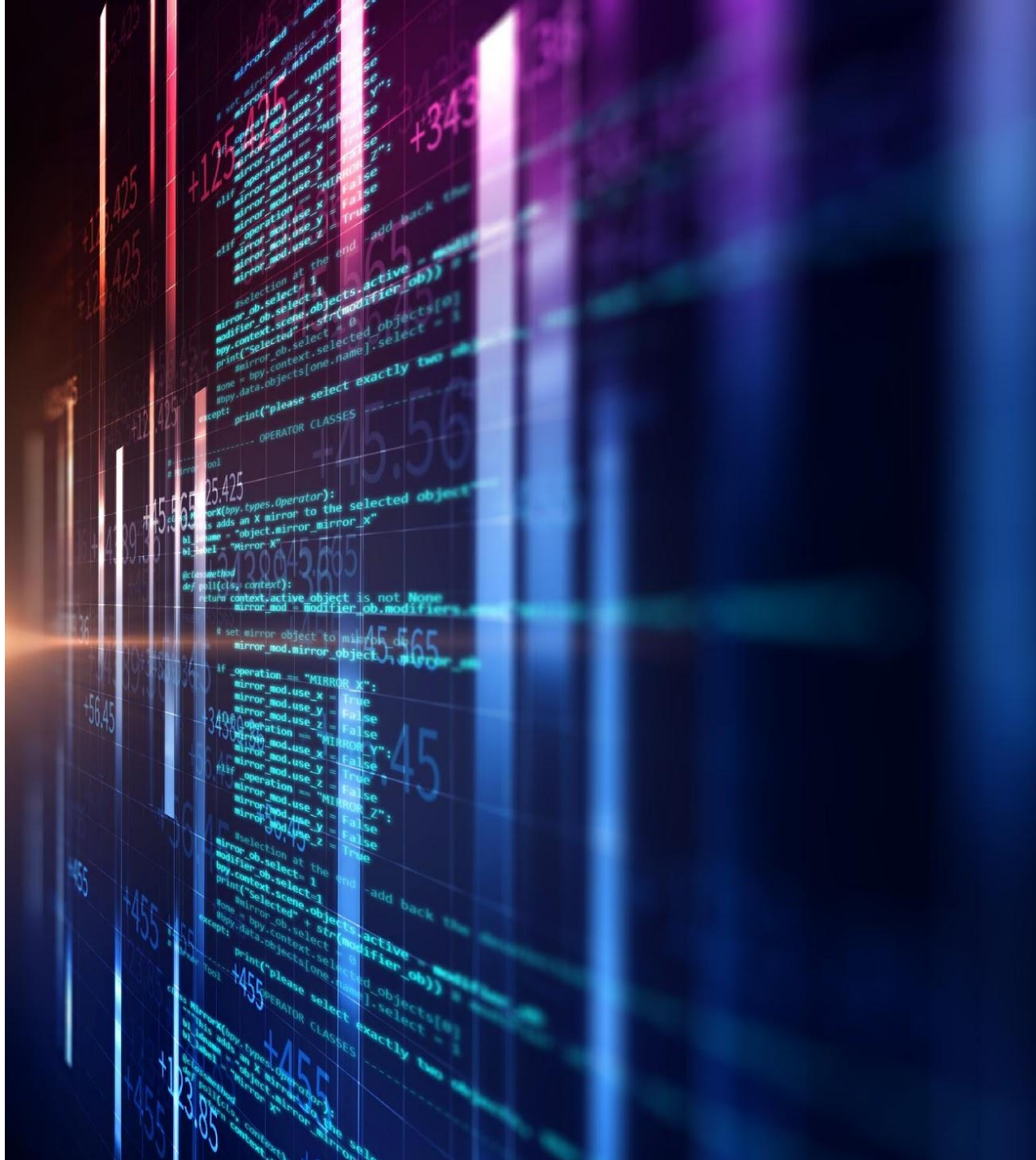
Connecting to Databases

Brandon Krakowsky

Penn Engineering



SQL & Databases Overview

What is SQL

- Structured Query Language
  - Some people pronounce is “sequel”
  - Others insist that “ess-cue-ell” is the only correct pronunciation
- A language for accessing and updating databases

What is a Database?

- A relational database (schema) is a collection of information stored in *tables* (entities) that relate to each other in some way

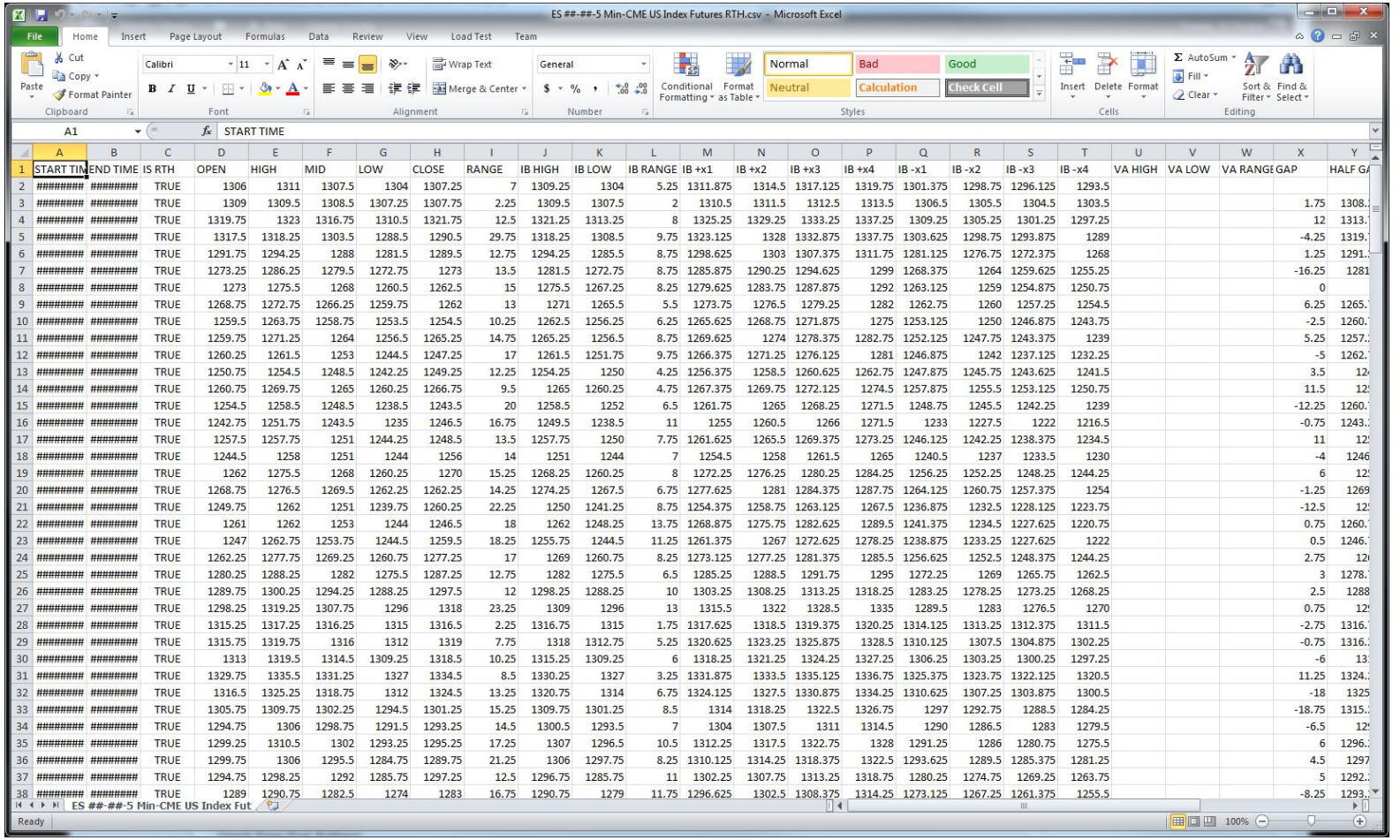
- Object oriented databases represent information in the form of *objects*

- This lecture will focus on accessing relational databases from Java

Relational Databases

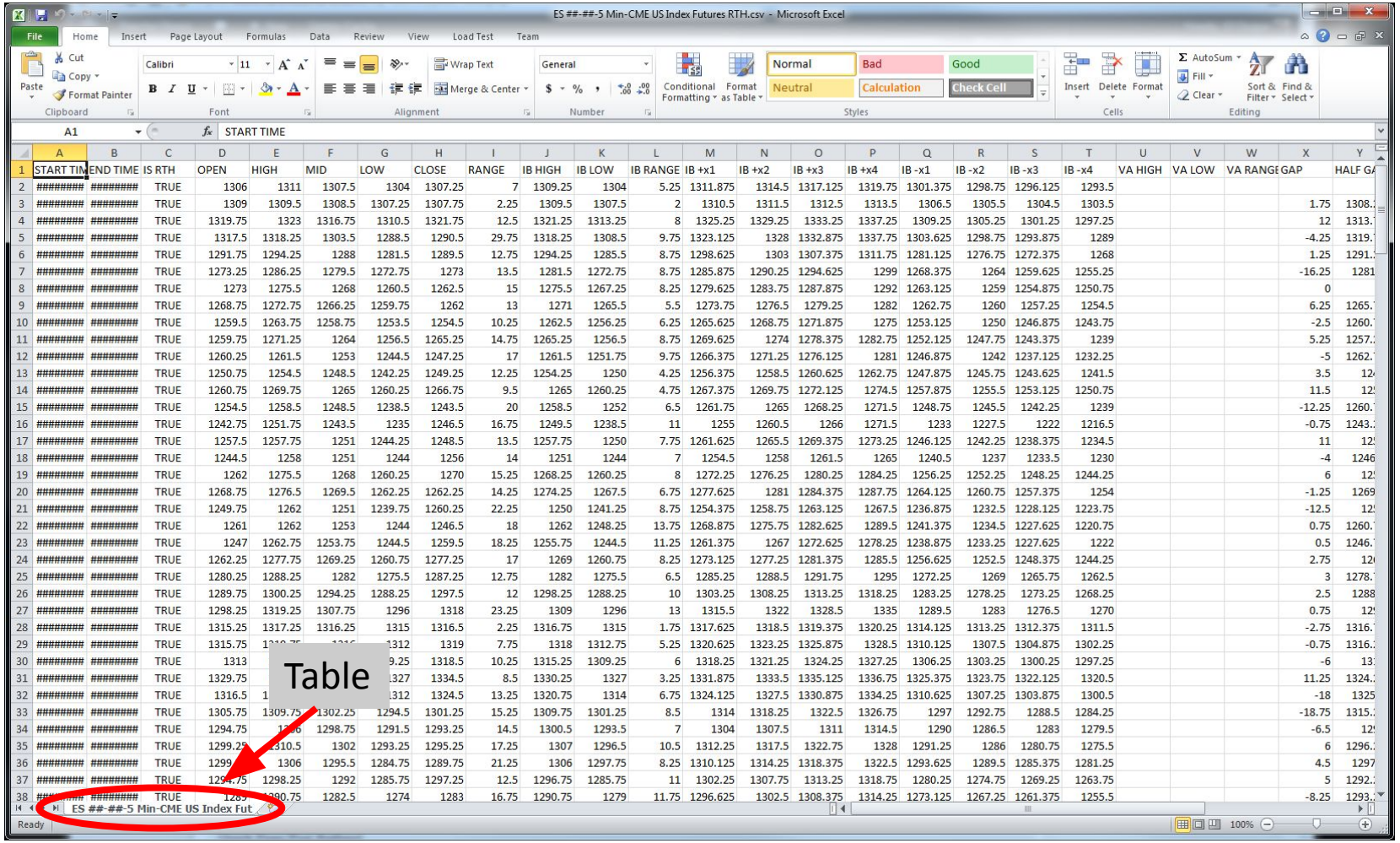
- Each *table* stores a particular type of thing (such as “Customers”)
  - A table consists of *rows* and *columns*
- A *column* (attribute) is a set of data values of a particular type
  - Each column defines a property of the entity (such as “Address”)
- A *row* is a single record in a table
  - It contains a single instance of the entity (such as one individual Customer)
- A *value* is a single column attribute for a single *row*

A Database is *Sort Of* Like Excel

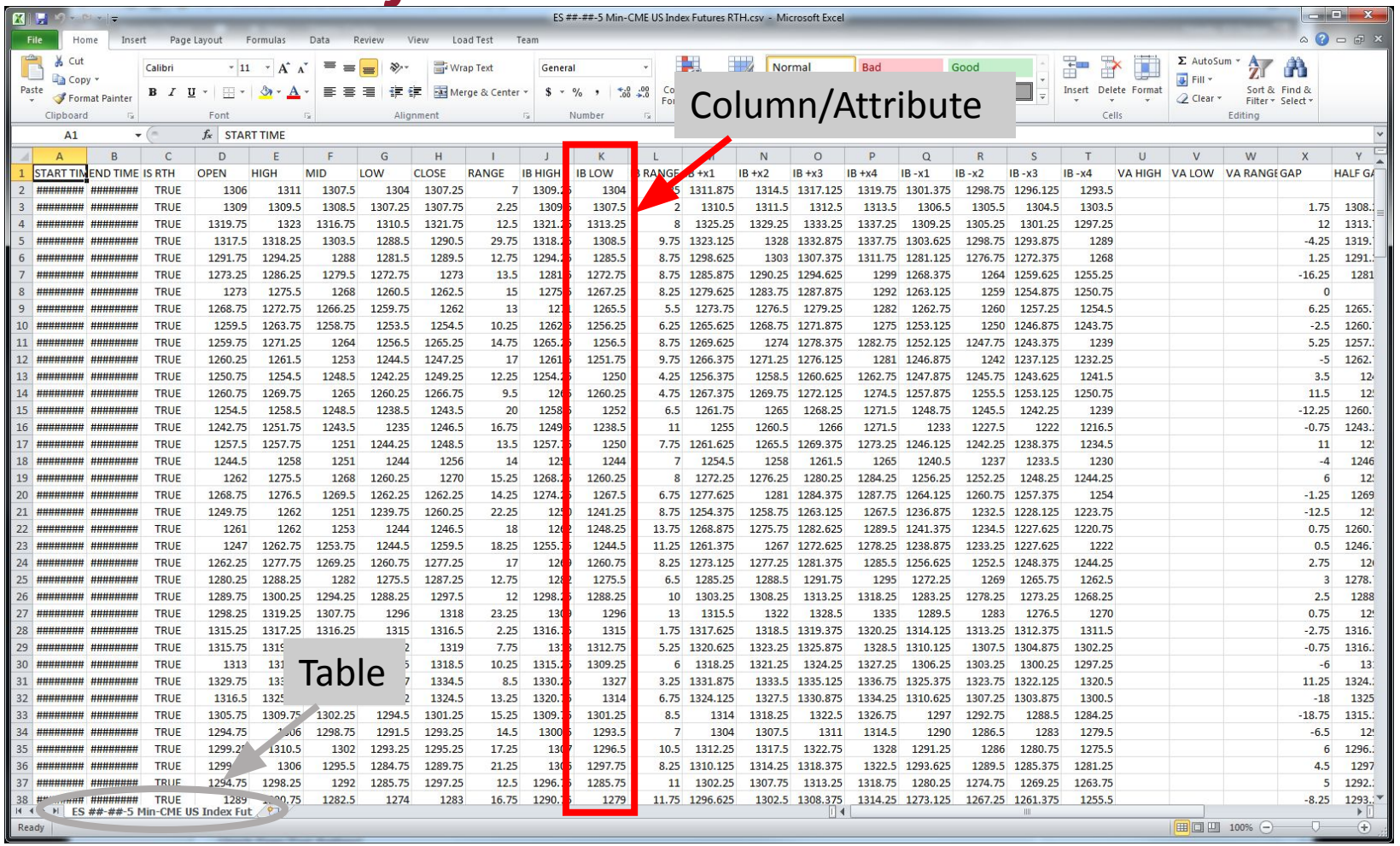


START TIME	END TIME	IS RTH	OPEN	HIGH	MID	LOW	CLOSE	RANGE	IB HIGH	IB LOW	IB RANGE	IB +x1	IB +x2	IB +x3	IB +x4	IB -x1	IB -x2	IB -x3	IB -x4	VA HIGH	VA LOW	VA RANGE	GAP	HALF GA
########	########	TRUE	1306	1311	1307.5	1304	1307.25	7	1309.25	1304	5.25	1311.875	1314.5	1317.125	1319.75	1301.375	1298.75	1296.125	1293.5					
########	########	TRUE	1309	1309.5	1308.5	1307.25	1307.75	2.25	1309.5	1307.5	2	1310.5	1311.5	1312.5	1313.5	1306.5	1305.5	1304.5	1303.5				1.75	1308.
########	########	TRUE	1319.75	1323	1316.75	1310.5	1321.75	12.5	1321.25	1313.25	8	1325.25	1329.25	1333.25	1337.25	1309.25	1305.25	1301.25	1297.25				12	1313.
########	########	TRUE	1317.5	1318.25	1303.5	1288.5	1290.5	29.75	1318.25	1308.5	9.75	1323.125	1328	1332.875	1337.75	1303.625	1298.75	1293.875	1289				-4.25	1319.
########	########	TRUE	1291.75	1294.25	1288	1281.5	1289.5	12.75	1294.25	1285.5	8.75	1298.625	1303	1307.375	1311.75	1281.125	1276.75	1272.375	1268				1.25	1291.
########	########	TRUE	1273.25	1286.25	1279.5	1272.75	1273	13.5	1281.5	1272.75	8.75	1285.875	1290.25	1294.625	1299	1268.375	1264	1259.625	1255.25				-16.25	1281
########	########	TRUE	1273	1275.5	1268	1260.5	1262.5	15	1275.5	1267.25	8.25	1279.625	1283.75	1287.875	1292	1263.125	1259	1254.875	1250.75				0	
########	########	TRUE	1268.75	1272.75	1266.25	1259.75	1262	13	1271	1265.5	5.5	1273.75	1276.5	1279.25	1282	1262.75	1260	1257.25	1254.5				6.25	1265.
########	########	TRUE	1259.5	1263.75	1258.75	1253.5	1254.5	10.25	1262.5	1256.25	6.25	1265.625	1268.75	1271.875	1275	1253.125	1250	1246.875	1243.75				-2.5	1260.
########	########	TRUE	1259.75	1271.25	1264	1256.5	1265.25	14.75	1265.25	1256.5	8.75	1269.625	1274	1278.375	1282.75	1252.125	1247.75	1243.375	1239				5.25	1257.
########	########	TRUE	1260.25	1261.5	1253	1244.5	1247.25	17	1261.5	1251.75	9.75	1266.375	1271.25	1276.125	1281	1246.875	1242	1237.125	1232.25				-5	1262.
########	########	TRUE	1250.75	1254.5	1248.5	1242.25	1249.25	12.25	1254.25	1250	4.25	1256.375	1258.5	1260.625	1262.75	1247.875	1245.75	1243.625	1241.5				3.5	12
########	########	TRUE	1260.75	1269.75	1265	1260.25	1266.75	9.5	1265	1260.25	4.75	1267.375	1269.75	1272.125	1274.5	1257.875	1255.5	1253.125	1250.75				11.5	12
########	########	TRUE	1254.5	1258.5	1248.5	1238.5	1243.5	20	1258.5	1252	6.5	1261.75	1265	1268.25	1271.5	1248.75	1245.5	1242.25	1239				-12.25	1260.
########	########	TRUE	1242.75	1251.75	1243.5	1235	1246.5	16.75	1249.5	1238.5	11	1255	1260.5	1266	1271.5	1233	1227.5	1222	1216.5				-0.75	1243.
########	########	TRUE	1257.5	1257.75	1251	1244.25	1248.5	13.5	1257.75	1250	7.75	1261.625	1265.5	1269.375	1273.25	1246.125	1242.25	1238.375	1234.5				11	12
########	########	TRUE	1244.5	1258	1251	1244	1256	14	1251	1244	7	1254.5	1258	1261.5	1265	1240.5	1237	1233.5	1230				-4	1246
########	########	TRUE	1262	1275.5	1268	1260.25	1270	15.25	1268.25	1260.25	8	1272.25	1276.25	1280.25	1284.25	1256.25	1252.25	1248.25	1244.25				6	12
########	########	TRUE	1268.75	1276.5	1269.5	1262.25	1262.25	14.25	1274.25	1267.5	6.75	1277.625	1281	1284.375	1287.75	1264.125	1260.75	1257.375	1254				-1.25	1269
########	########	TRUE	1249.75	1262	1251	1239.75	1260.25	22.25	1250	1241.25	8.75	1254.375	1258.75	1263.125	1267.5	1236.875	1232.5	1228.125	1223.75				-12.5	12
########	########	TRUE	1261	1262	1253	1244	1246.5	18	1262	1248.25	13.75	1268.875	1275.75	1282.625	1289.5	1241.375	1234.5	1227.625	1220.75				0.75	1260.
########	########	TRUE	1247	1262.75	1253.75	1244.5	1259.5	18.25	1255.75	1244.5	11.25	1261.375	1267	1272.625	1278.25	1238.875	1233.25	1227.625	1222				0.5	1246.
########	########	TRUE	1262.25	1277.75	1269.25	1260.75	1277.25	17	1269	1260.75	8.25	1273.125	1277.25	1281.375	1285.5	1256.625	1252.5	1248.375	1244.25				2.75	12
########	########	TRUE	1280.25	1288.25	1282	1275.5	1287.25	12.75	1282	1275.5	6.5	1285.25	1288.5	1291.75	1295	1272.25	1269	1265.75	1262.5				3	1278.
########	########	TRUE	1289.75	1300.25	1294.25	1288.25	1297.5	12	1298.25	1288.25	10	1303.25	1308.25	1313.25	1318.25	1283.25	1278.25	1273.25	1268.25				2.5	1288
########	########	TRUE	1298.25	1319.25	1307.75	1296	1318	23.25	1309	1296	13	1315.5	1322	1328.5	1335	1289.5	1283	1276.5	1270				0.75	12
########	########	TRUE	1315.25	1317.25	1316.25	1315	1316.5	2.25	1316.75	1315	1.75	1317.625	1318.5	1319.375	1320.25	1314.125	1313.25	1312.375	1311.5				-2.75	1316.
########	########	TRUE	1315.75	1319.75	1316	1312	1319	7.75	1318	1312.75	5.25	1320.625	1323.25	1325.875	1328.5	1310.125	1307.5	1304.875	1302.25				-0.75	1316.
########	########	TRUE	1313	1319.5	1314.5	1309.25	1318.5	10.25	1315.25	1309.25	6	1318.25	1321.25	1324.25	1327.25	1306.25	1303.25	1300.25	1297.25				-6	13
########	########	TRUE	1329.75	1335.5	1331.25	1327	1334.5	8.5	1330.25	1327	3.25	1331.875	1333.5	1335.125	1336.75	1325.375	1323.75	1322.125	1320.5				11.25	1324.
########	########	TRUE	1316.5	1325.25	1318.75	1312	1324.5	13.25	1320.75	1314	6.75	1324.125	1327.5	1330.875	1334.25	1310.625	1307.25	1303.875	1300.5				-18	1325
########	########	TRUE	1305.75	1309.75	1302.25	1294.5	1301.25	15.25	1309.75	1301.25	8.5	1314	1318.25	1322.5	1326.75	1297	1292.75	1288.5	1284.25				-18.75	1315.
########	########	TRUE	1294.75	1306	1298.75	1291.5	1293.25	14.5	1300.5	1293.5	7	1304	1307.5	1311	1314.5	1290	1286.5	1283	1279.5				-6.5	12
########	########	TRUE	1299.25	1310.5	1302	1293.25	1295.25	17.25	1307	1296.5	10.5	1312.25	1317.5	1322.75	1328	1291.25	1286	1280.75	1275.5				6	1296.
########	########	TRUE	1299.75	1306	1295.5	1284.75	1289.75	21.25	1306	1297.75	8.25	1310.125	1314.25	1318.375	1322.5	1293.625	1289.5	1285.375	1281.25				4.5	1297
########	########	TRUE	1294.75	1298.25	1292	1285.75	1297.25	12.5	1296.75	1285.75	11	1302.25	1307.75	1313.25	1318.75	1280.25	1274.75	1269.25	1263.75				5	1292.
########	########	TRUE	1289	1290.75	1282.5	1274	1283	16.75	1290.75	1279	11.75	1296.625	1302.5	1308.375	1314.25	1273.125	1267.25	1261.375	1255.5				-8.25	1293.

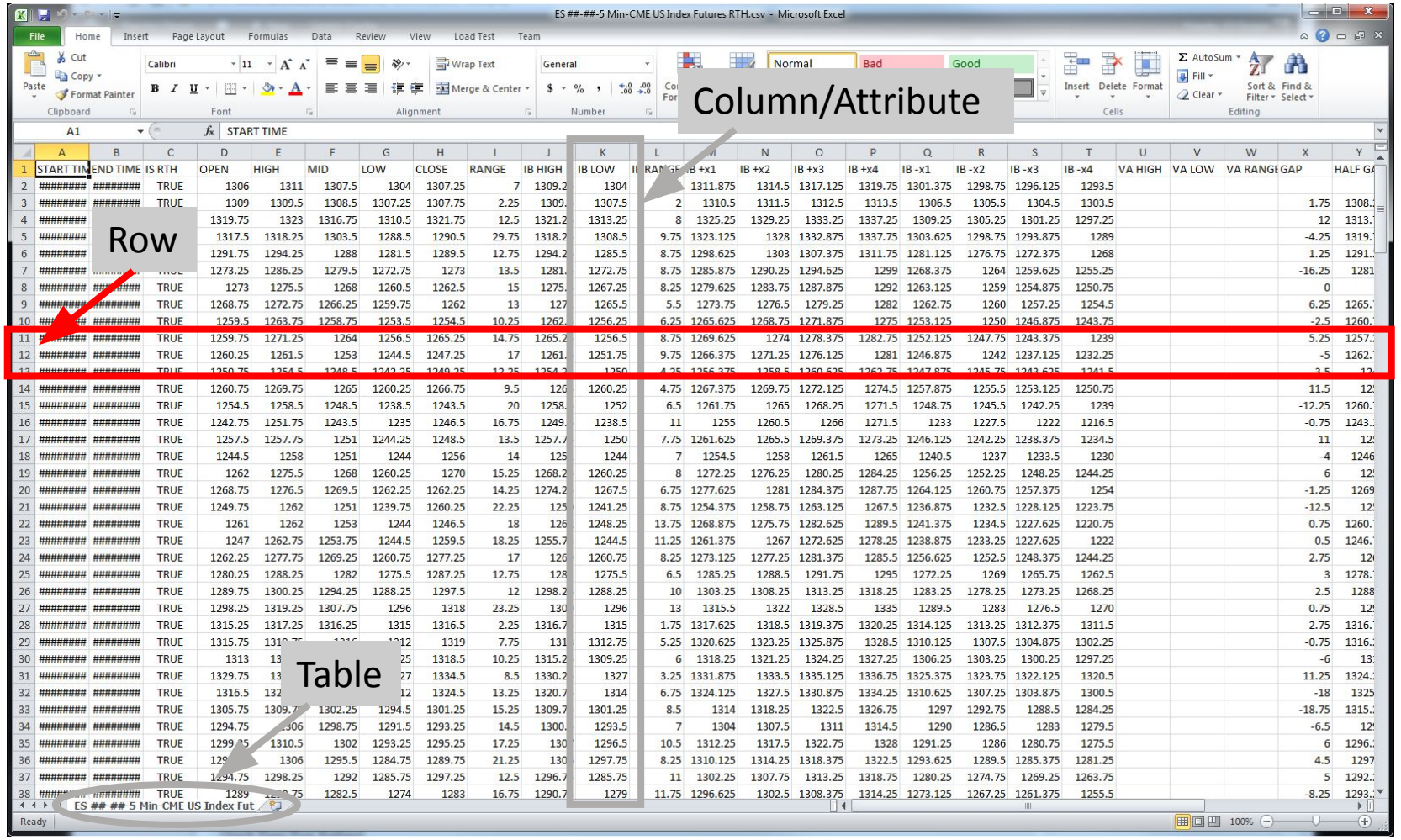
A Database is *Sort Of* Like Excel



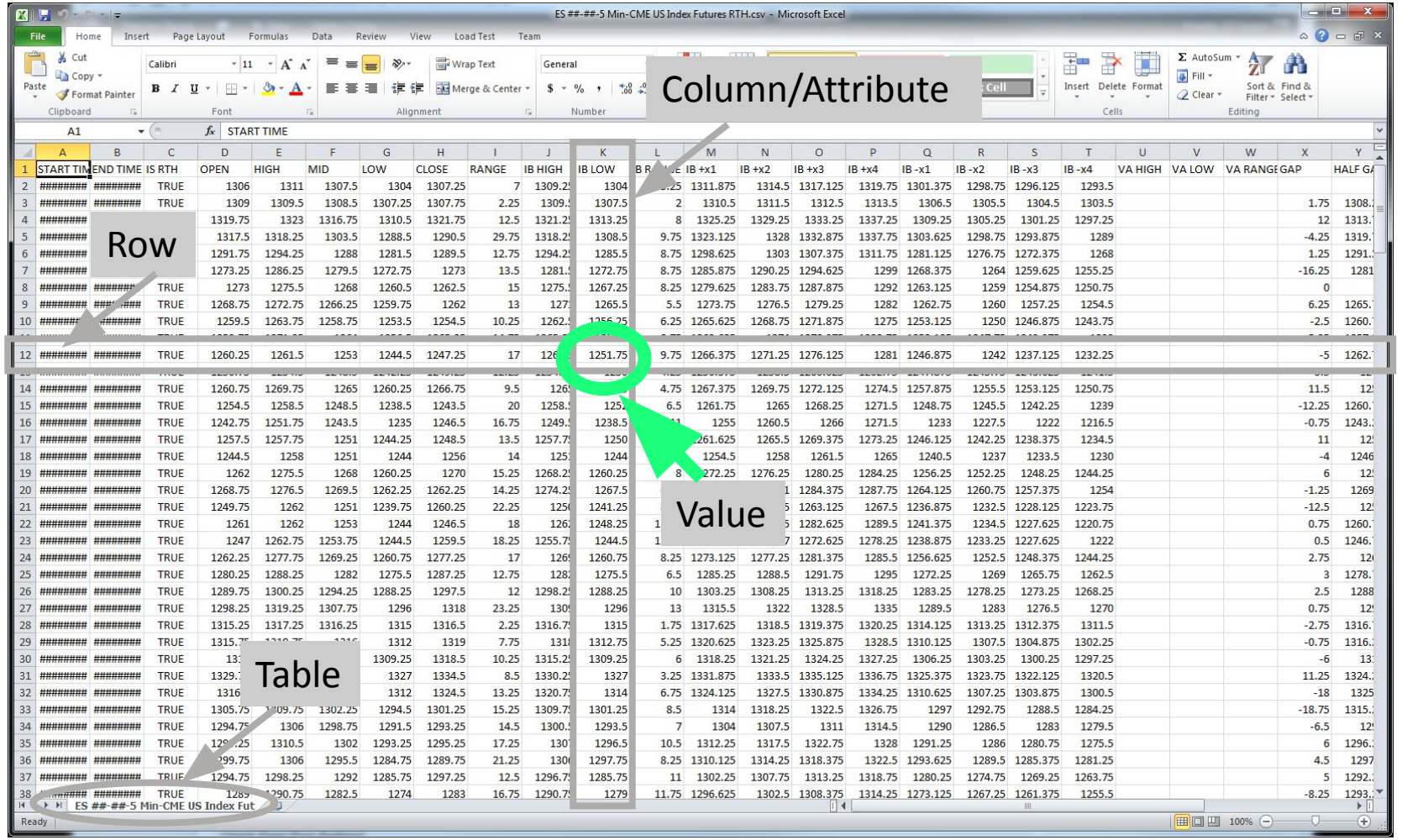
A Database is *Sort Of* Like Excel



A Database is *Sort Of* Like Excel



A Database is *Sort Of* Like Excel



How Do I Open a Database?

- You don't!
  - A database runs inside of a database management system (DBMS)
  - You "connect" to a database
- MySQL is one of the most commonly used open source DBMSs
  - It's free!
  - Great performance – very solid and stable
  - Works with just about any operating system
- For this lecture, the data will be hosted in MySQL and we'll connect to it from Java

JDBC

JDBC

- JDBC stands for Java Database Connectivity
- It's an API for accessing relational databases from Java
- JDBC allows you to:
  - Create a connection to a database in a DBMS (for example, MySQL)
  - Issue database queries
  - Make updates
  - Receive results

JDBC Driver

- To use JDBC, you'll need the database specific implementation of the JDBC driver
- To connect to MySQL, you have to use the JDBC driver from MySQL
- The MySQL JDBC driver is called MySQL Connector/J
  - You can find the latest version with installation instructions at this URL: https://dev.mysql.com/downloads/connector/j/
- Select your operating system or "platform independent", download the .tar.gz or .zip file, and extract the JAR file
  - A JAR (Java ARchive) is a package file typically used to aggregate multiple Java class files for distribution
  - For example, mysql-connector-java-<version>.jar
- Create a new Java Project and put the JAR file where Java can find it
  - Add the JAR file to your CLASSPATH, or
  - In Eclipse, go to: Project --> Properties --> Java Build Path --> Libraries --> Add External Jars…

Feedback Database Project

MySQL Server & Database

- To connect to a MySQL database from Java, first make sure your MySQL server is running
- Then create a database and table

```
1  CREATE DATABASE feedback;
2  USE feedback;
3
4  CREATE TABLE comments (
5      id INT NOT NULL AUTO_INCREMENT,
6      my_user VARCHAR(30) NOT NULL,
7      email VARCHAR(30),
8      webpage VARCHAR(100) NOT NULL,
9      datum DATETIME NOT NULL,
10     summary VARCHAR(40) NOT NULL,
11     comments VARCHAR(400) NOT NULL,
12     PRIMARY KEY (ID)
13 );
```

- The SQL code to the left creates a database “feedback” and then selects it for use
- It then creates a new table “comments” with 7 columns
- The “comments” table will store individual reviews of websites
- Each review will be a row in the table

DatabaseConnection Class: Database Credentials

DatabaseConnection.java

```java
package connection;
import java.sql.Connection;

/**
 * Manages database connection.
 * @author lbrandon
 */
public class DatabaseConnection {

    /**
     * JDBC database connection String.
     */
    private static String url = "jdbc:mysql://100.26.51.170:3306/feedback";

    //FOR DEMO PURPOSES ONLY
    //YOU SHOULD NEVER STORE HARD-CODED CREDENTIALS IN YOUR PROGRAM
    //Instead, you should store in environment variables,
    //encrypted configuration files,
    //or some other external authentication method.
    /**
     * Database username.
     */
    private static String username = "5e53b2";

    /**
     * Database password.
     */
    private static String password = "b72bca";
```

DatabaseConnection Class: Open Database

```
27      /**
28       * Opens a database connection.
29       * @return the db connection
30       */
31      public static Connection openDatabase() {
32
33          Connection connection = null;
34
35          try {
36              // Load the appropriate MySQL driver
37              Class.forName("com.mysql.cj.jdbc.Driver");
38
39              //create connection using JDBC driver
40              connection = DriverManager
41                      .getConnection(
42                              DatabaseConnection.url,
43                              DatabaseConnection.username,
44                              DatabaseConnection.password);
45          } catch (ClassNotFoundException e) {
46              // TODO Auto-generated catch block
47              e.printStackTrace();
48          } catch (SQLException e) {
49              // TODO Auto-generated catch block
50              e.printStackTrace();
51          }
52
53          return connection;
54      }
```

DatabaseConnection Class: Close Database

```
53      /**
54       * Closes given database connection.
55       * @param connection to close
56       */
57      public static void closeDatabase(Connection connection) {
58          try {
59              connection.close();
60          } catch (SQLException e) {
61              // TODO Auto-generated catch block
62              e.printStackTrace();
63          }
64      }
```

DatabaseQueries Class: Read From Database

DatabaseConnection.java | FeedbackDatabase.java | DatabaseQueries.java

```
1  package query;
2  import java.sql.Connection;
7
8  /**
9   * Manages database queries.
10  * @author lbrandon
11  *
12  */
13 public class DatabaseQueries {
14
15     /**
16      * Query database for given webpage using given connection.
17      * @param connection to use
18      * @param webpage to query for
19      */
20     public static void readFromDatabase(Connection connection, String webpage) {
21
22         try {
23             PreparedStatement preparedStatement = null;
24
25             if (webpage == null) {
26                 //PreparedStatement with SQL
27                 preparedStatement = connection
28                         .prepareStatement("SELECT * FROM comments");
29             } else {
30                 //if webpage is defined, filter SQL query with inserted variable
31                 preparedStatement = connection
32                         .prepareStatement("SELECT * FROM comments WHERE webpage = ?");
33                 preparedStatement.setString(1, webpage);
34             }
35
```

DatabaseQueries Class: Read From Database

```
35
36              //execute query and get result set
37              ResultSet resultSet = preparedStatement.executeQuery();
38
39              DatabaseQueries.printResultSet(resultSet);
40
41              resultSet.close();
42              preparedStatement.close();
43
44          } catch (SQLException e) {
45              // TODO Auto-generated catch block
46              e.printStackTrace();
47          }
48      }
49
```

DatabaseQueries Class: Read From Database

```
50      /**
51       * Query database using given connection.
52       * @param connection to use
53       */
54      public static void readFromDatabase(Connection connection) {
55          //call overloaded method readFromDatabase with null webpage argument
56          DatabaseQueries.readFromDatabase(connection, null);
57      }
58
```

DatabaseQueries Class: Write to Database

```
61      /**
62       * Writes given attributes to a record in the database using the given connection.
63       * @param connection to use
64       * @param my_user writing comments
65       * @param email of user
66       * @param webpage for review
67       * @param summary of webpage
68       * @param comments about webpage
69       * @return number of records inserted into the db
70       */
71      public static int writeToDatabase(Connection connection, String my_user, String email,
72              String webpage, String summary, String comments) {
73
74          int ret = 0;
75
76          try {
77              PreparedStatement preparedStatement = connection
78                      .prepareStatement("INSERT INTO comments VALUES (default, ?, ?, ?, ?, ?, ?)");
79
```

DatabaseQueries Class: Write to Database

```
79
80              //set parameter values via index, starting at 1
81              preparedStatement.setString(1, my_user);
82              preparedStatement.setString(2, email);
83              preparedStatement.setString(3, webpage);
84
85              //dynamically create date via Date object
86              long millis = System.currentTimeMillis();
87              preparedStatement.setDate(4, new java.sql.Date(millis));
88
89              preparedStatement.setString(5, summary);
90              preparedStatement.setString(6, comments);
91
92              //execute SQL and get return value (number of inserted rows)
93              ret = preparedStatement.executeUpdate();
94
95              preparedStatement.close();
96
97          } catch (SQLException e) {
98              // TODO Auto-generated catch block
99              e.printStackTrace();
100         }
101
102         return ret;
103     }
104
```

DatabaseQueries Class: Print Result Set

```
105     /**
106      * Prints given result set.
107      * @param resultSet to print
108      */
109     private static void printResultSet(ResultSet resultSet) {
110 
111         try {
112             while (resultSet.next()) {
113 
114                 // Get the column values via name
115                 String user = resultSet.getString("my_user");
116                 String email = resultSet.getString("email");
117                 String website = resultSet.getString("webpage");
118                 String summary = resultSet.getString("summary");
119                 Date date = resultSet.getDate("datum");
120                 String comment = resultSet.getString("comments");
121 
122                 // Note:
123                 // You can also get the column values via the column number which starts at 1
124                 // e.g. resultSet.getString(2);
125 
```

DatabaseQueries Class: Print Result Set

```
126                 //print column values
127                 System.out.println("User: " + user);
128                 System.out.println("Email: " + email);
129                 System.out.println("Website: " + website);
130                 System.out.println("Summary: " + summary);
131                 System.out.println("Date: " + date);
132                 System.out.println("Comment: " + comment);
133                 System.out.println("_____");
134             }
135         } catch (SQLException e) {
136             // TODO Auto-generated catch block
137             e.printStackTrace();
138         }
139     }
140 }
```

FeedbackDatabase Class: Query or Insert Records

DatabaseConnection.java | FeedbackDatabase.java

```
1  import java.sql.Connection;
6
7  /**
8   * Query or insert records into a feedback database, allowing users to review websites.
9   * @author lbrandon
10  *
11  */
12 public class FeedbackDatabase {
13
14     public static void main(String[] args) {
15
16         //open db connection
17         Connection connection = DatabaseConnection.openDatabase();
18
19         //create scanner for user input
20         Scanner scanner = new Scanner(System.in);
21
22         String input = null;
23         boolean usingDB = true;
24
```

FeedbackDatabase Class: Query or Insert Records

```
35
36          while (usingDB) {
37
38              //ask user what they want to do, query or input
39              System.out.println("What do you want to do? 'Query', 'Input', or 'Quit':");
40              input = scanner.nextLine().trim();
41
42              //switch statement: multi-way branch statement
43              switch(input.toLowerCase()) {
44
45                  //query database
46                  case "query":
47
48                      System.out.println("Which website or 'all'?:");
49                      input = scanner.nextLine().trim();
50
51                      //query entire database
52                      if ("all".equals(input)) {
53                          DatabaseQueries.readFromDatabase(connection);
54                      //query database with filter
55                      } else {
56                          DatabaseQueries.readFromDatabase(connection, input);
57                      }
58
59                      break;
```

FeedbackDatabase Class: Query or Insert Records

```
60
61                  case "input":
62
63                      //input record into database
64                      System.out.println("(Separated by a comma) enter your name, email, "
65                              + "webpage, a summary, and your comments:");
66                      input = scanner.nextLine().trim();
67
68                      //prepare attributes
69                      info = input.split(",");
70                      my_user = info[0].trim();
71                      email = info[1].trim();
72                      webpage = info[2].trim();
73                      summary = info[3].trim();
74                      comments = info[4].trim();
75
76                      DatabaseQueries.writeToDatabase(connection, my_user, email, webpage, summary, comments);
77                      DatabaseQueries.readFromDatabase(connection, webpage);
78
79                      break;
80
```

FeedbackDatabase Class: Query or Insert Records

```
79
80                  default:
81
82                      //quit the program
83                      usingDB = false;
84
85                      break;
86              }
87
88              System.out.println();
89          }
90
91          scanner.close();
92          DatabaseConnection.closeDatabase(connection);
93      }
94
```