CNT 4714 – Project Four – Spring 2019

Title: "Project Four: Developing A Three-Tier Distributed Web-Based Application"
Points: 100 points (bonus problem potentially adds 15 points – see page 9.)
Due Date: Sunday April 7, 2019 by 11:59 pm (WebCourses time)

Objectives: To incorporate many of the techniques you've learned so far this semester into a distributed three-tier web-based application which uses servlets and JSP technology running on a Tomcat container/server to access and maintain a persistent MySQL database using JDBC.

Description: In this assignment you will utilize a suppliers/parts/jobs/shipments database (creation/population script available on the course assignment page) as the back-end database. Front-end access to this database by the client will occur through a single page displayed in the client's web browser. The schema of the backend database consists of four tables with the following schemas for each table:

suppliers (snum, sname, status, city) //information about suppliers
parts (pnum, pname, color, weight, city) //information about parts
jobs (jnum, jname, numworkers, city) //information about jobs
shipments (snum, pnum, jnum, quantity) //suppliers ship parts to jobs in specific quantities

The first-tier (client-level front-end) of your application will be a JSP page that allows the client to enter SQL commands into a window (i.e. a form) and submit them to the server application for processing. The front-end (**and only the front-end**) will utilize JSP technology. The client front-end will provide the user a simple form in which they will enter a SQL command (any DML, DDL, or DCL command could theoretically be entered by the user, however we will restrict to queries, insert, update, replace, and delete commands). The front-end will provide only two buttons for the user, an Execute button that will cause the execution of the SQL command they enter, and a Reset button that simply clears any content in the form input area. The client front-end will run on any web-based browser that you would like to use. You can elect to have a default query or not, it is entirely your decision. The application will connect to the backend database as a root user client.

The second-tier servlet, in addition to handling the SQL command interface will also implement the business/application logic. This logic will increment by 5, the status of a supplier anytime that supplier is involved in the insertion/update of a shipment record in which the quantity is greater than or equal to 100. Note that any update of quantity >= 100 will affect any supplier involved in a shipment with a quantity >= 100. The example screen shots illustrate this case. An insert of a shipment tuple (S5, P6, J7, 400) will cause the status of every supplier who has a shipment with a quantity of 100 or greater to be increased by 5. In other words, even if a supplier's shipment is not directly affected by the update, their status will be affected if they have any shipment with quantity >= 100. **(See page 9 for a bonus problem that implements a modified version of this business rule.)** The business logic of the second tier will reside in the servlet on the Tomcat web-application server (server-side application). This means that the business logic is not to be implemented in the DBMS via a trigger.

The third-tier (back-end) is the persistent MySQL database described above and is under control of the MySQL DBMS server. All you need to do with the database is run the creation/population script. See the important note below concerning when/how to re-run this script for your final submission.

References:
Notes: Lecture Notes for MySQL installation and use. Documentation for MySQL available at: http://www.mysql.com. More information on JDBC can be found at: http://www.oracle.com/technetwork/java/javase/jdbc/index.html . More information on Tomcat can be found at http://tomcat.apache.org. Lecture Notes for Servlets. Lecture Notes for JSPs.

Restrictions:
Your source file shall begin with comments containing the following information:

```
/*  Name:
    Course: CNT 4714 – Spring 2019 – Project Four
    Assignment title:  A Three-Tier Distributed Web-Based Application
    Date:  April 7, 2019
*/
```

Input Specification: The suppliers/part/jobs/shipments database that is created/populated by the script `project4dbscript.sql`, is the back-end to this application. All other input comes from the front-end client submitted to the application server based servlet entered as either queries or updates to this database. The set of commands that you are to execute against this database are included in the `project4commands.sql` file available on the course homework page. I do not expect your front-end to execute the script. You'll need to execute the commands in this script one at a time in your application (copy and paste!). I've put them into a script file for convenience and so that you can run the script in the MySQL Workbench if you'd like to compare/see the result sets for each user command.

Output Specification: All output is generated by the servlet and should appear in the user's browser as a text/html page presented to the user. **IMPORTANT:** Be sure to re-run the `project4` database creation/population script before you begin creating your screen shots for submission. By doing so you will ensure that the database is in its initial state so that all update operations will produce the values we are expecting to see in your result outputs.

Deliverables:

(1) You should submit your entire `webapps` folder from Tomcat for this program. If you submit the entire folder, then all of the files necessary to execute your web application will be included with the directory structure intact. Submit this via WebCourses no later than **11:59pm Sunday April 7, 2019**.

(2) The following 14 screen shots must be submitted along with your webapps folder. (You can include the screenshots in the top-level of your webapps folder if you'd like, just be sure to include a note that you've done so.)

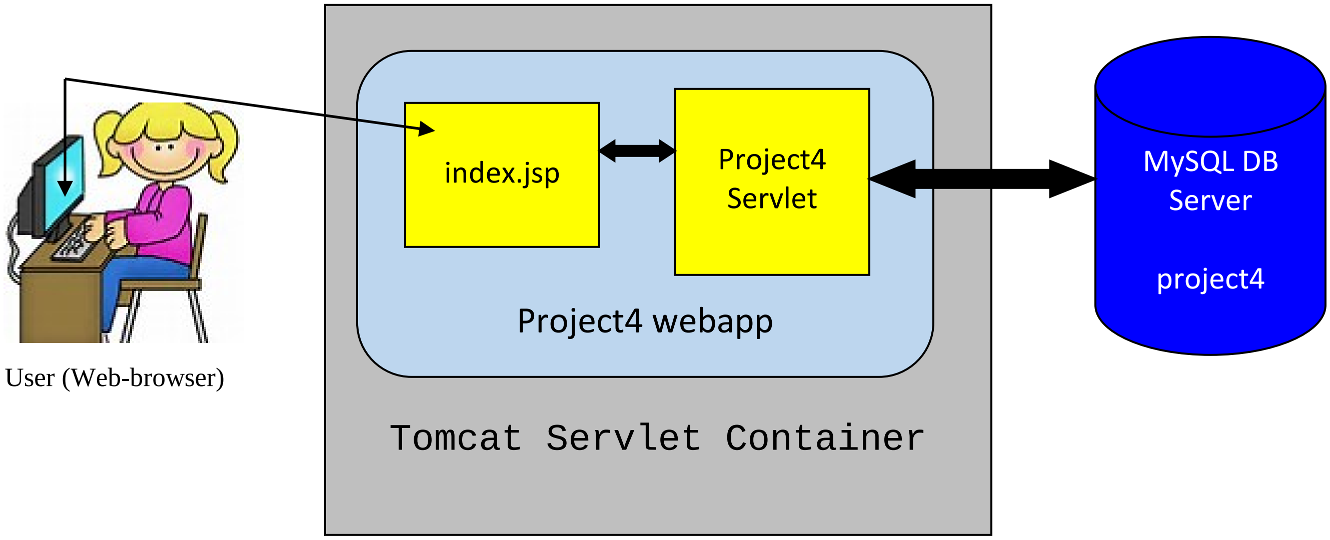
  a. Command 1
  b. Command 2A
  c. Command 2B – output may vary here
  d. Command 2C
  e. Command 3A – output may vary here
  f. Command 3B
  g. Command 3C – output may vary here
  h. Command 3D
  i. Command 4
  j. Command 5A
  k. Command 5B – output may vary here
  l. Command 5C

m. Command 5D
n. Command 6

Additional Information:
Be very careful when setting up the directory structures required for the web applications running under your server (Tomcat 9.0.10 or later). See the course notes on servlets for the exact directory structure that must be developed.

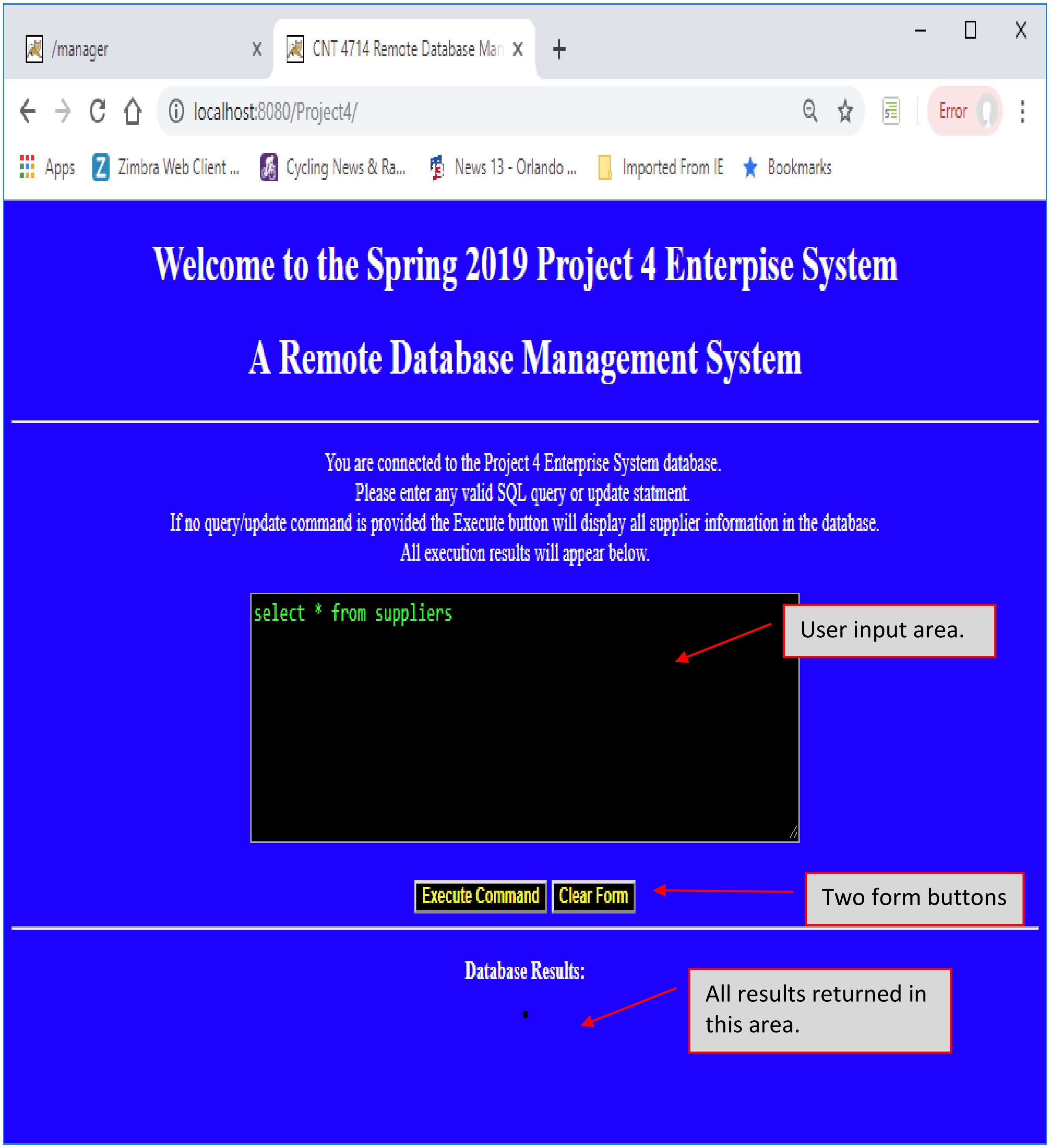
Important: Please name your webapp: **Project4.** Let the Amirreza and Abhianshu (the TAs) know if you are doing the bonus problem by attaching a note to your WebCourses submission.

Schematic Overview of Project Components:



Some screen shots illustrating the application.

Main client screen (initial configuration using a default query string):



Welcome to the Spring 2019 Project 4 Enterprise System

A Remote Database Management System

You are connected to the Project 4 Enterprise System database.
Please enter any valid SQL query or update statment.
If no query/update command is provided the Execute button will display all supplier information in the database.
All execution results will appear below.

```
select * from suppliers
```

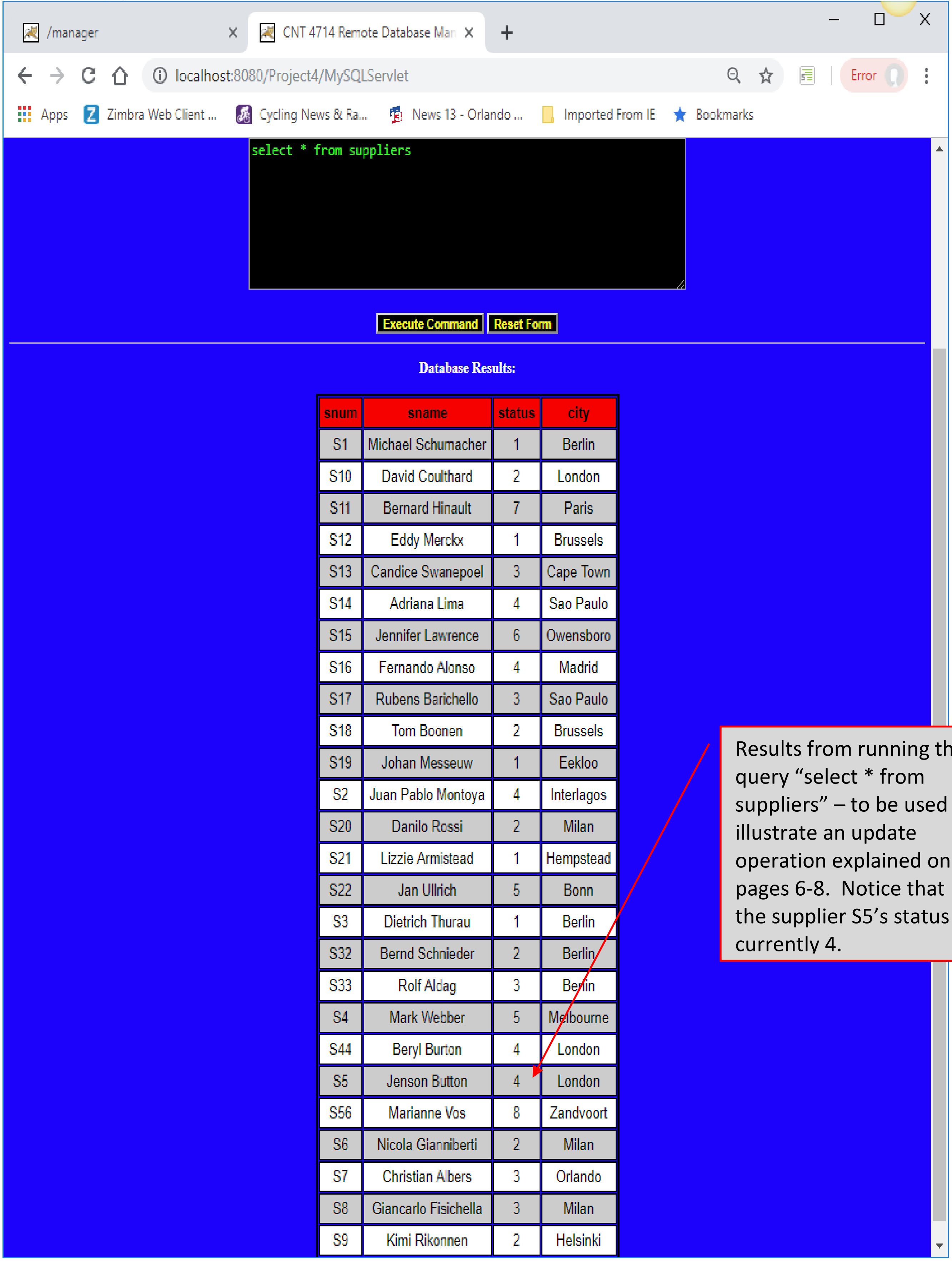
User input area.

Two form buttons

Database Results:

All results returned in this area.

Client simply clicks the “Execute Command” button and the SQL command in the form is executed:



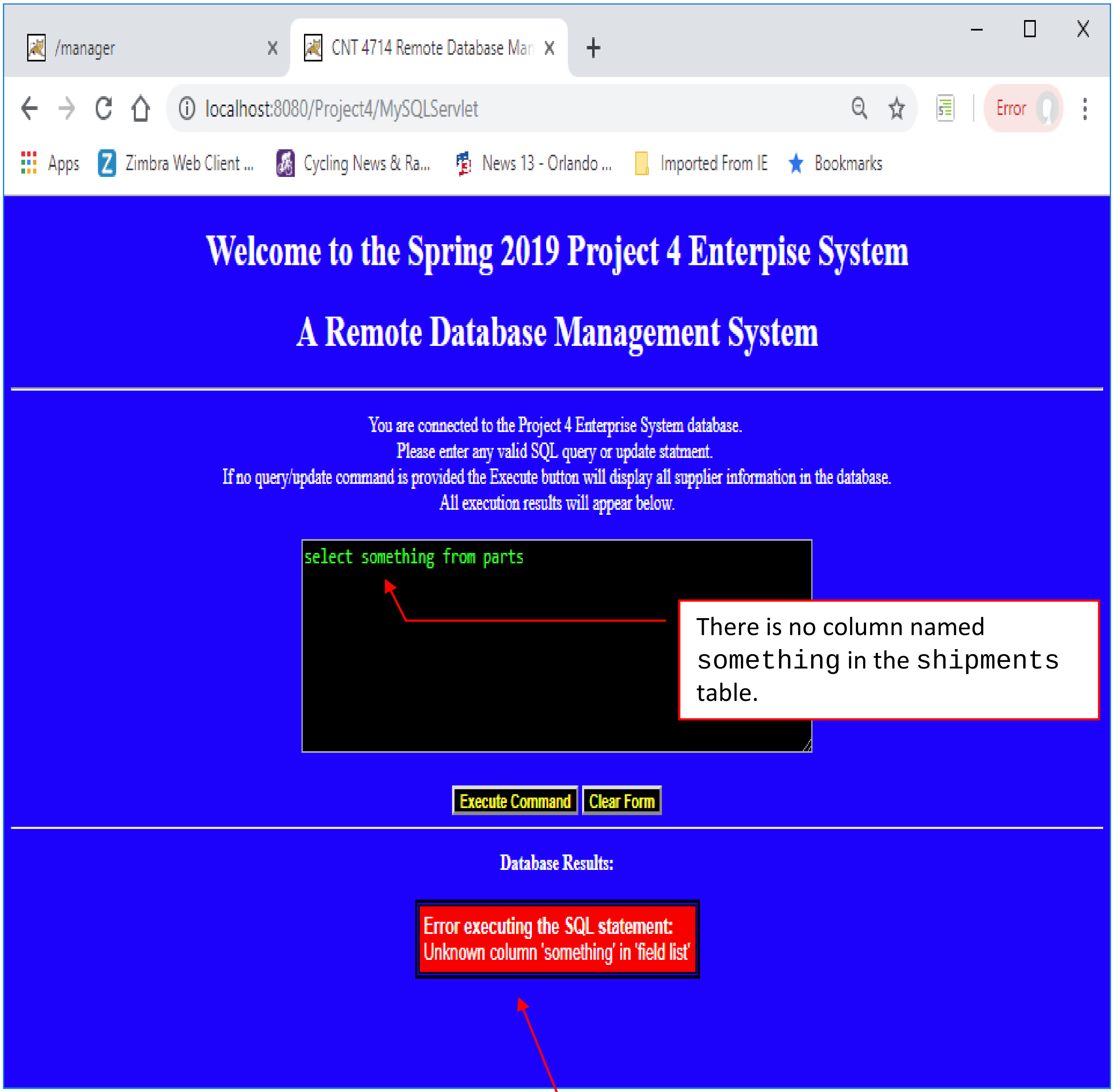
```
select * from suppliers
```

Database Results:

snum	sname	status	city
S1	Michael Schumacher	1	Berlin
S10	David Coulthard	2	London
S11	Bernard Hinault	7	Paris
S12	Eddy Merckx	1	Brussels
S13	Candice Swanepoel	3	Cape Town
S14	Adriana Lima	4	Sao Paulo
S15	Jennifer Lawrence	6	Owensboro
S16	Fernando Alonso	4	Madrid
S17	Rubens Barichello	3	Sao Paulo
S18	Tom Boonen	2	Brussels
S19	Johan Messeuw	1	Eekloo
S2	Juan Pablo Montoya	4	Interlagos
S20	Danilo Rossi	2	Milan
S21	Lizzie Armistead	1	Hempstead
S22	Jan Ullrich	5	Bonn
S3	Dietrich Thurau	1	Berlin
S32	Bernd Schnieder	2	Berlin
S33	Rolf Aldag	3	Berlin
S4	Mark Webber	5	Melbourne
S44	Beryl Burton	4	London
S5	Jenson Button	4	London
S56	Marianne Vos	8	Zandvoort
S6	Nicola Gianniberti	2	Milan
S7	Christian Albers	3	Orlando
S8	Giancarlo Fisichella	3	Milan
S9	Kimi Rikonnen	2	Helsinki

Results from running the query “select * from suppliers” – to be used to illustrate an update operation explained on pages 6-8. Notice that the supplier S5’s status is currently 4.

Client makes a mistake entering an SQL command:



Welcome to the Spring 2019 Project 4 Enterpise System

A Remote Database Management System

You are connected to the Project 4 Enterprise System database.
Please enter any valid SQL query or update statment.
If no query/update command is provided the Execute button will display all supplier information in the database.
All execution results will appear below.

```
select something from parts
```

There is no column named `something` in the `shipments` table.

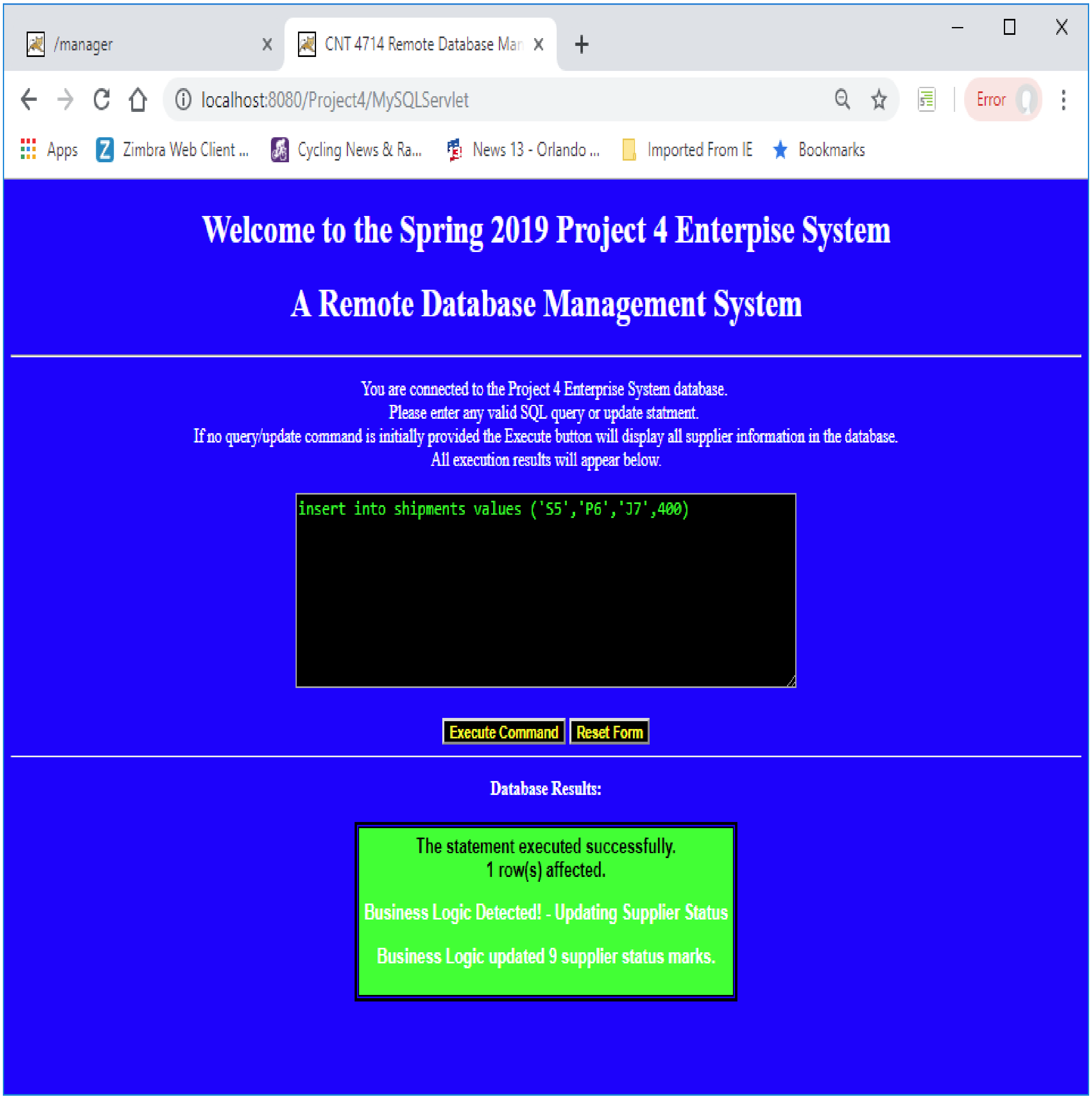
Database Results:

Error executing the SQL statement:
Unknown column 'something' in 'field list'

Error message is returned from MySQL indicating the problem with the operation.

Inserts and updates may cause changes to the supplier status field (business logic is triggered) as shown below:

Client issues the following insert command:



Welcome to the Spring 2019 Project 4 Enterprise System

A Remote Database Management System

You are connected to the Project 4 Enterprise System database.
Please enter any valid SQL query or update statment.
If no query/update command is initially provided the Execute button will display all supplier information in the database.
All execution results will appear below.

```
insert into shipments values ('S5','P6','J7',400)
```

Database Results:

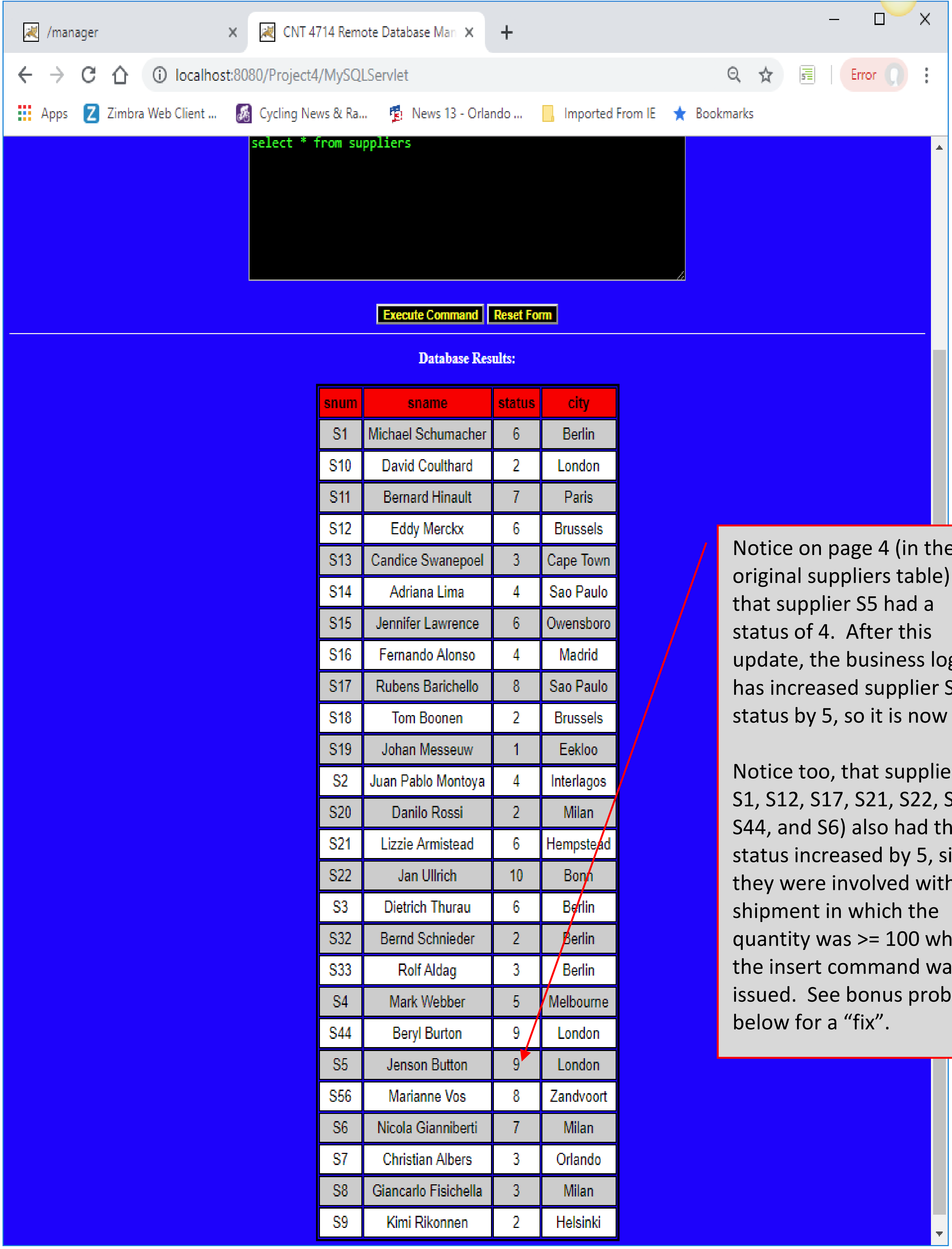
The statement executed successfully.
1 row(s) affected.

Business Logic Detected! - Updating Supplier Status

Business Logic updated 9 supplier status marks.

Alert message when an update to the quantity field in the shipments table has caused an update of a supplier's status in the supplier table. Note that in my application, I used this alert message any time the business logic was tested even if it did not trigger any updates. This means that this message would appear with different values even if no rows are updated.

After executing update command (the previous insert), client runs select * from suppliers.



```
select * from suppliers
```

Database Results:

snum	sname	status	city
S1	Michael Schumacher	6	Berlin
S10	David Coulthard	2	London
S11	Bernard Hinault	7	Paris
S12	Eddy Merckx	6	Brussels
S13	Candice Swanepoel	3	Cape Town
S14	Adriana Lima	4	Sao Paulo
S15	Jennifer Lawrence	6	Owensboro
S16	Fernando Alonso	4	Madrid
S17	Rubens Barichello	8	Sao Paulo
S18	Tom Boonen	2	Brussels
S19	Johan Messeuw	1	Eekloo
S2	Juan Pablo Montoya	4	Interlagos
S20	Danilo Rossi	2	Milan
S21	Lizzie Armistead	6	Hempstead
S22	Jan Ullrich	10	Bonn
S3	Dietrich Thurau	6	Berlin
S32	Bernd Schnieder	2	Berlin
S33	Rolf Aldag	3	Berlin
S4	Mark Webber	5	Melbourne
S44	Beryl Burton	9	London
S5	Jenson Button	9	London
S56	Marianne Vos	8	Zandvoort
S6	Nicola Gianniberti	7	Milan
S7	Christian Albers	3	Orlando
S8	Giancarlo Fisichella	3	Milan
S9	Kimi Rikonnen	2	Helsinki

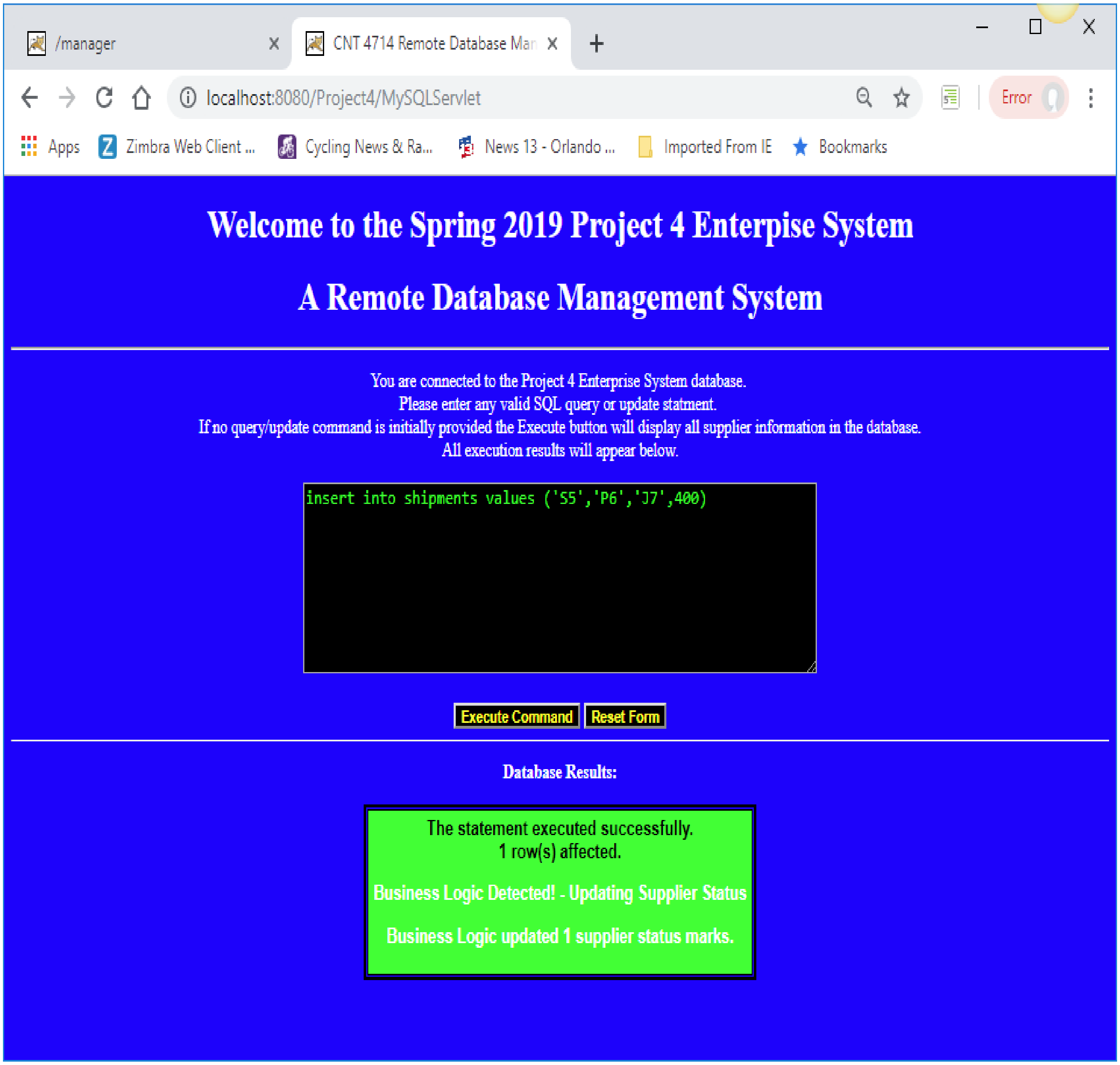
Notice on page 4 (in the original suppliers table) that supplier S5 had a status of 4. After this update, the business logic has increased supplier S5's status by 5, so it is now 9.

Notice too, that suppliers S1, S12, S17, S21, S22, S3, S44, and S6) also had their status increased by 5, since they were involved with a shipment in which the quantity was >= 100 when the insert command was issued. See bonus problem below for a "fix".

BONUS PROBLEM: 15 points

Instead of allowing any update/insert of a quantity >= 100 to affect any supplier with a shipment involving a quantity >= 100, adjust the business logic portion of your application so that an insert/update of a quantity greater than 100, causes a change to the status of only those suppliers directly affected by the update. For example, using the case shown above, when inserting the row (S5, P6, J7, 400) into the shipments table, only the status of supplier S5 should be increased by 5 (see screen shot below). However, an update such as: UPDATE shipments SET quantity = quantity + 50 WHERE pnum = “P3”, would increase by 5 the status of every supplier who ships part P3 in a quantity >= 100 after the update has been issued.

NOTE: If you elect to do the bonus problem, submit only this version of your application. Do not also submit the non-bonus problem version. Let Amirreza and Abhianshu (the TAs) know if you’ve elected to do the bonus problem or not.



Welcome to the Spring 2019 Project 4 Enterprise System

A Remote Database Management System

You are connected to the Project 4 Enterprise System database.
Please enter any valid SQL query or update statment.
If no query/update command is initially provided the Execute button will display all supplier information in the database.
All execution results will appear below.

```
insert into shipments values ('S5','P6','J7',400)
```

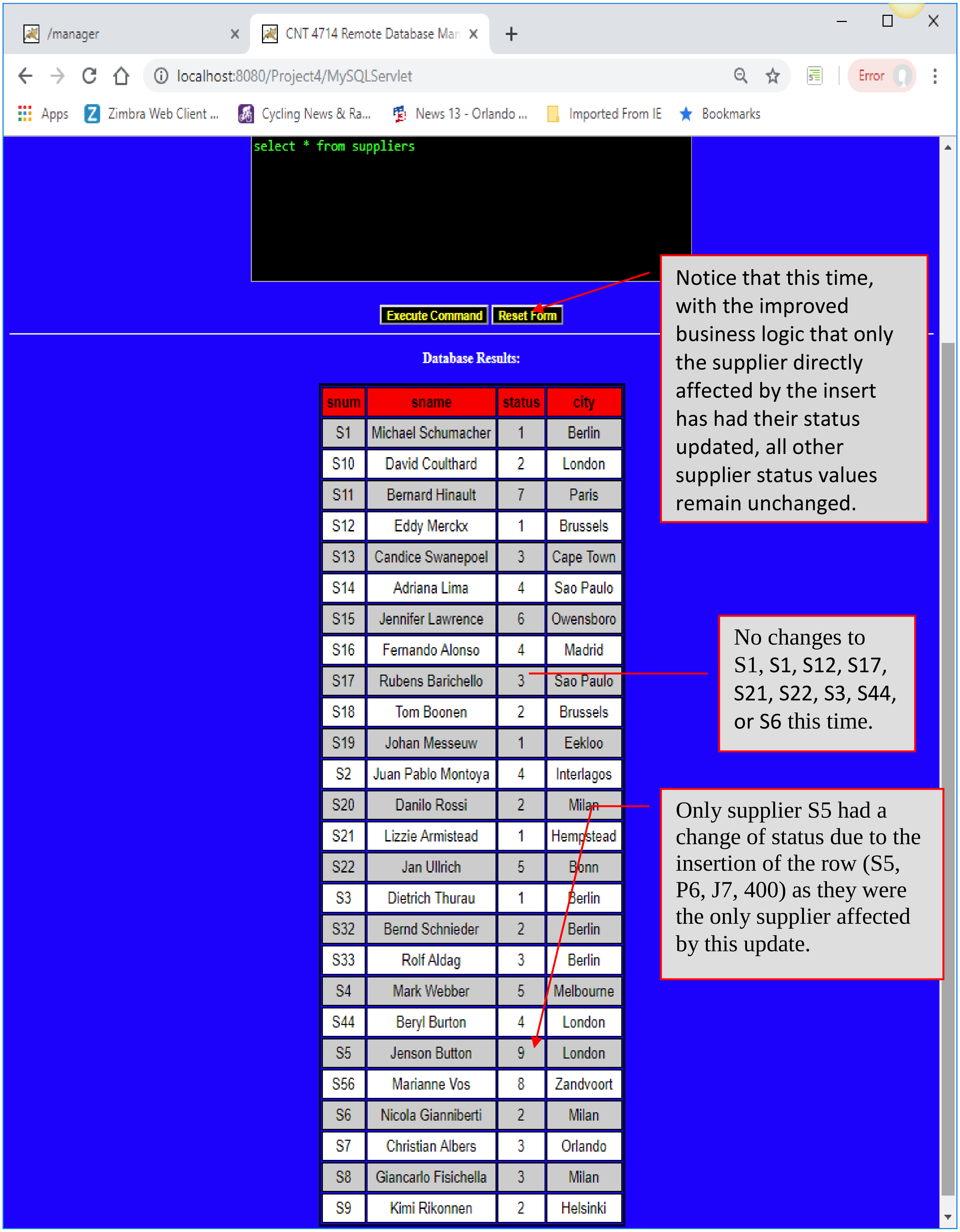
Database Results:

The statement executed successfully.
1 row(s) affected.

Business Logic Detected! - Updating Supplier Status

Business Logic updated 1 supplier status marks.

Immediately after issuing the update (insert above), the user reruns the select * from suppliers query:



```
select * from suppliers
```

Database Results:

snum	sname	status	city
S1	Michael Schumacher	1	Berlin
S10	David Coulthard	2	London
S11	Bernard Hinault	7	Paris
S12	Eddy Merckx	1	Brussels
S13	Candice Swanepoel	3	Cape Town
S14	Adriana Lima	4	Sao Paulo
S15	Jennifer Lawrence	6	Owensboro
S16	Fernando Alonso	4	Madrid
S17	Rubens Barichello	3	Sao Paulo
S18	Tom Boonen	2	Brussels
S19	Johan Messeuw	1	Eekloo
S2	Juan Pablo Montoya	4	Interlagos
S20	Danilo Rossi	2	Milan
S21	Lizzie Armistead	1	Hempstead
S22	Jan Ullrich	5	Bonn
S3	Dietrich Thurau	1	Berlin
S32	Bernd Schnieder	2	Berlin
S33	Rolf Aldag	3	Berlin
S4	Mark Webber	5	Melbourne
S44	Beryl Burton	4	London
S5	Jenson Button	9	London
S56	Marianne Vos	8	Zandvoort
S6	Nicola Gianniberti	2	Milan
S7	Christian Albers	3	Orlando
S8	Giancarlo Fisichella	3	Milan
S9	Kimi Rikonnen	2	Helsinki

Notice that this time, with the improved business logic that only the supplier directly affected by the insert has had their status updated, all other supplier status values remain unchanged.

No changes to S1, S1, S12, S17, S21, S22, S3, S44, or S6 this time.

Only supplier S5 had a change of status due to the insertion of the row (S5, P6, J7, 400) as they were the only supplier affected by this update.