Introduction to MySQL

Overview

This lab walks you through using MySQL. MySQL is a relational database that can be used as part of Web and other applications. This lab serves as a primer for using MySQL and will serve as a foundation when we discuss SQL injection attacks and possible mitigations.

Learning Outcomes:

At the completion of the lab you should be able to:

1. Connect to a MySQL database and show the tables within the SDEV AMI virtual machine
2. Create MySQL tables containing popular data types and constraints
3. Insert, update and delete data from MySQL database tables
4. Create and execute SQL Select statements and simple joins on MySQL tables

Lab Submission Requirements:

After completing this lab, you will submit a word (or PDF) document that meets all of the requirements in the description at the end of this document. In addition, your associated files should be submitted. You can submit multiple files in a zip file.

Virtual Machine Account Information

Your Virtual Machine has been preconfigured with all of the software you will need for this class. You have connected to this machine in the previous labs. Reconnect again using the Remote Desktop connection, your Administrator username and password.

Part 1 – Connect to a MySQL database and show the tables within the AWS Cloud VM

The Virtual Machine already has MySQL installed. A MySQL username has also been created along with a database to use for your applications and testing. We previously tested this earlier in the semester. Recall the following steps to connect to your MySQL database on your VM

1. Login to your VM using the remote desktop.
2. Open your command prompt and change to the Bitnami\WampStack-7.1.16-0\mysql\bin directory and enter the following command:

   ```
   mysql –u root –p
   ```

   When prompted enter the following mysql root password:

   ```
   sdev300vm99
   ```

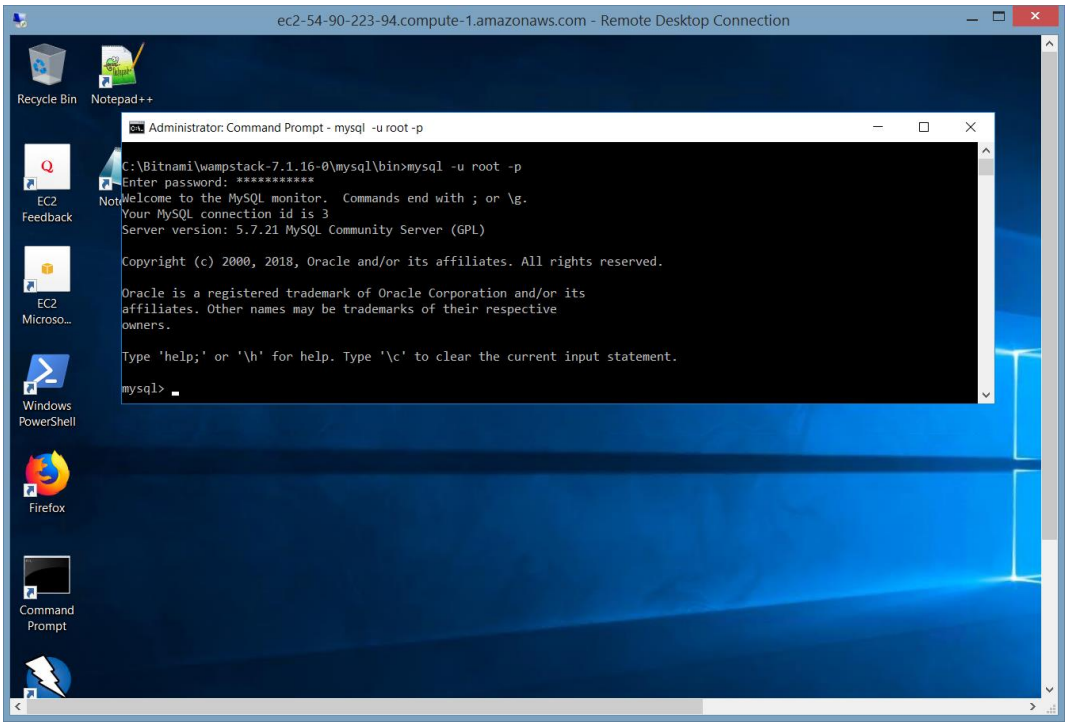

Figure 1 Connecting to MySQL on the VM

To display the available databases type the following at the mysql prompt:

```
show databases;
```

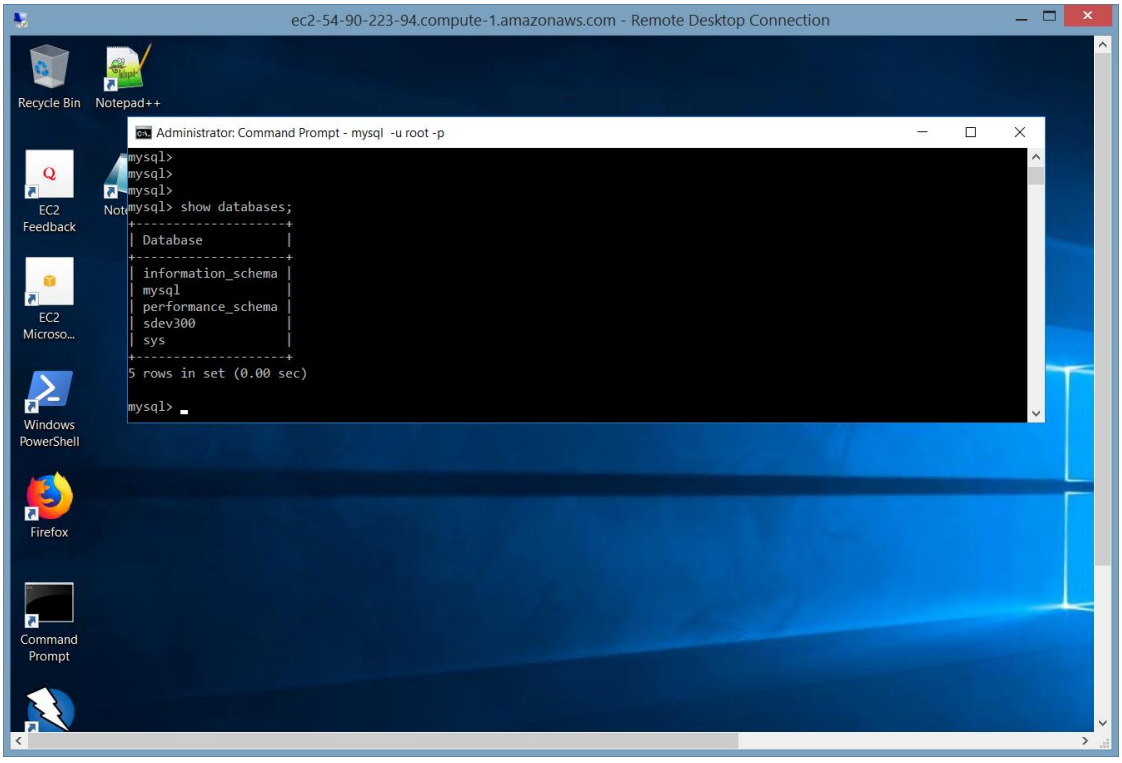

Figure 2 Show the available MySQL databases

3. The database we will be using for this course is sdev300. To use this database, type the following at the mysql prompt:

   use sdev300;

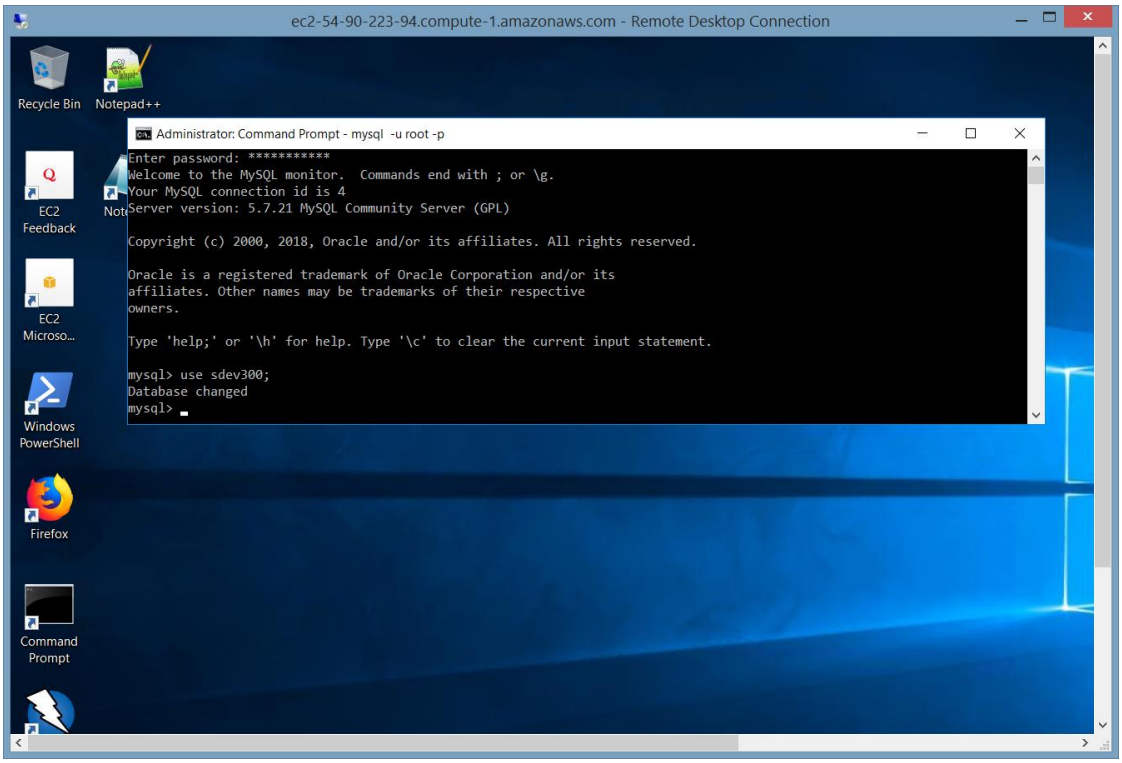

Figure 3 Selecting the SDEV300 database

Note: the Bitnami WAMP application provides a mysql-admin interface that you are welcome to use. For this course, instructions will only be provided for the command prompt mysql interactions.

Table 1 shows some additional handy commands to help navigate the databases and tables within MySQL. All of these commands are entered directly at the mysql> prompt.

Command	**Description**
show databases;	Lists the names of all available MySQL database schemas.
use database;	(e.g. use sdev300;) Moves to the database specified providing access to tables and all other objects within that database.
show tables;	Lists the tables for the current database.
exit;	Exits the MySQL database and returns to the command prompt.

When logged in as root, you can easily move back forth between database schemas. Keep in mind this is not typically the privileges a developer would be given. Usually, you would have much less visibility of other databases and even tables within that database. There is a separate database security course (SDEV 350) that addresses more details about users, roles and permissions that contribute to the very important and foundational principle of least privilege.

Part 2 Create MySQL tables containing popular data types and constraints

The reading for this week covered the foundations for creating and dropping tables using a variety of data types and constraints. In this exercise we will create three tables that could be used to represent a very simple student and course registration system. The tables all have primary keys. One table provides foreign keys to the other two tables.

When creating SQL commands to be executed in MySQL, it is always recommended to prepare them in a text editor and then either run the script or copy and paste into the MySQL application. Since this isn't a course in database design, we will just copy and paste from the Notepad++ text editor.

1. Create a folder named SQL to hold your SQL scripts in the documents folder (see figure 4).

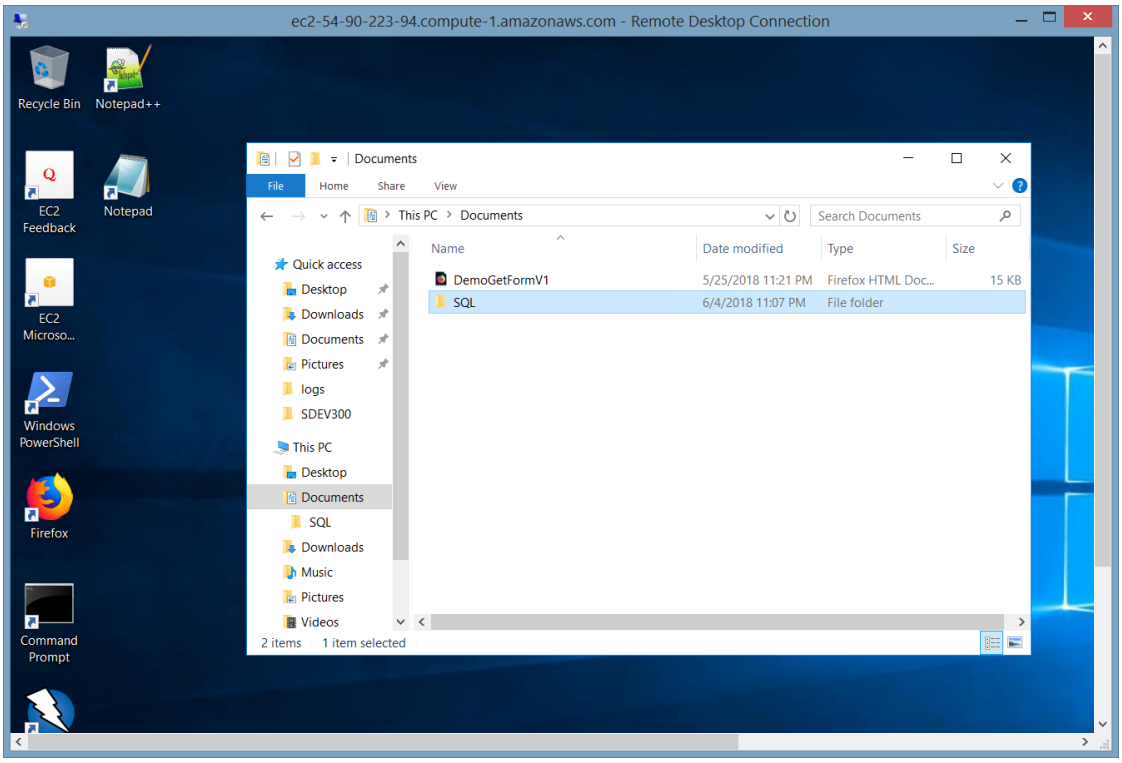

Figure 4 Create SQL Folder

2. Open up Notepad++ and copy and paste the following SQL code into a file named CreateTables.sql. (See figure 5)

```
use sdev300;

CREATE TABLE Students (
PSUsername varchar(30) primary key,
FirstName varchar(30),
LastName varchar(30),
EMail varchar(60)
);

CREATE TABLE Courses(
CourseID int primary key,
CourseDisc varchar(4),
CourseNum varchar(4),
CourseTitle varchar(75)
);

CREATE TABLE StudentCourses (
StudentCourseID int primary key,
CourseID int,
PSUsername varchar(30),
Constraint SC1 Foreign Key (CourseID) references Courses(CourseID) on
delete cascade,
Constraint SC2 Foreign Key (PSUsername) references Students(PSUsername)
On delete cascade
);
```

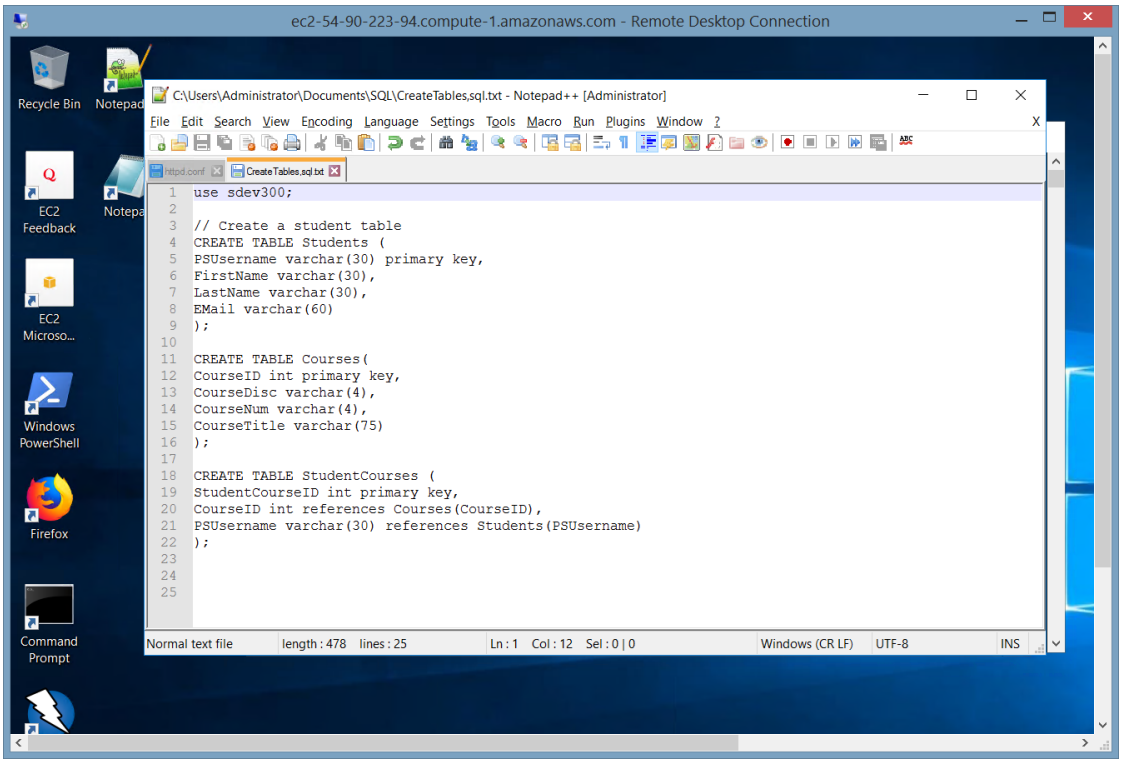

Figure 5 Creating Tables

3. Open your command prompt and change to the Bitnami\WampStack-7.1.16-0\mysql\bin directory and enter the following command:

```
mysql -u root -p
```

When prompted enter the following mysql root password:

```
sdev300vm99
```

4. At the mysql prompt begin copying and pasting the SQL code. You can copy it all at once. Use the show tables; command to demonstrate you successfully created the tables as shown in figure 6.

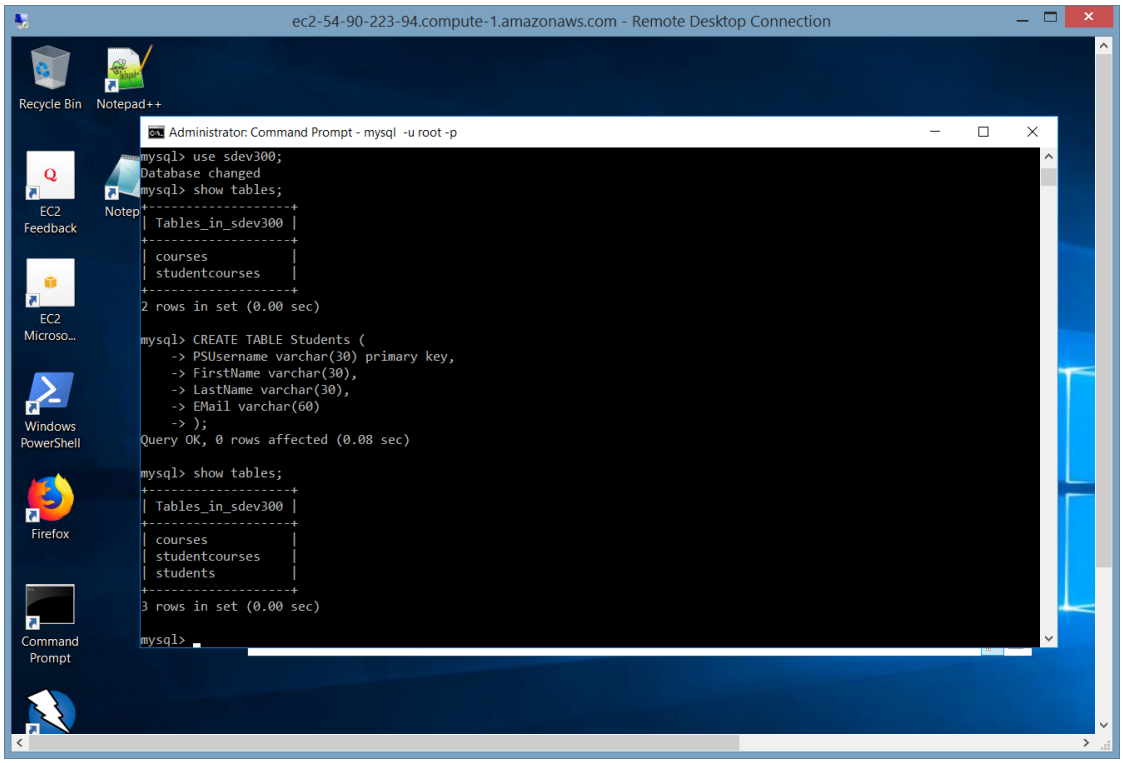

Figure 6 Running the SQL script

Note the Query OK output from MySQL is an indicator your SQL execution was successful.

5. You can also describe each table as shown in figure 7.

```
desc Courses;

desc StudentCourses;

desc Students;
```

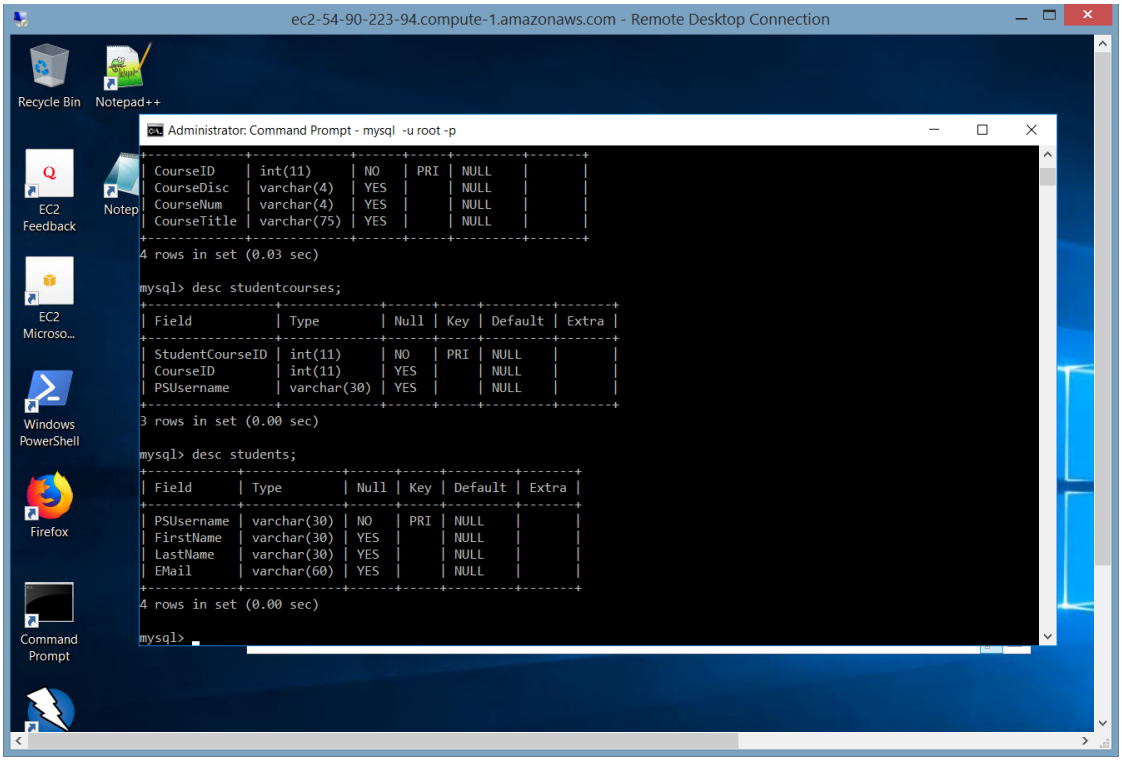

Figure 7 Describing the tables

Important note: On a Linux Operating system, MySQL table data and names are case sensitive. Hence (Students is not the same as students or STUDENTS). Case is not sensitive for MySQL databases on a windows machine. This fundamental different can be quite frustrating when developing on multiple Operating Systems. It is recommended you design assuming the table and object names will be case sensitive. It makes debugging much less challenging later.

You should experiment in MySQL by creating one or two of your own tables. Be sure to use several data types and add a primary key for each table and other constraints as needed. Use the MySQL reference documentation to discover additional functionality and datatypes as needed:

https://dev.mysql.com/doc/refman/8.0/en/

Part 3 Insert, update and delete data from MySQL database tables

Once tables have been created yor can insert records and then update the record or even delete the record. This exercise discusses how to use MySQL to populate and modify the records in your database.

We will once again, create the database scripts using the Notpad++ text editor.

1. Using Notepad++, copy and paste the following SQL code into a file named ModifyTables.sql

```
-- Use the appropriate database
   use sdev300;

-- Insert students
insert into Students
values ('jsmith', 'John','Smith','jsmith@students.umuc.edu');

insert into Students
values ('mjones', 'Mary','Jones','mjones@students.umuc.edu');

insert into Students
values ('jparsons', 'Jeff','Parsons','jparsons@students.umuc.edu');

-- Insert Courses
insert into Courses
values (1, 'SDEV','300','Secure Web Development');

insert into Courses
values (2, 'SDEV','360','Secure Software LifeCycle');

-- Insert student courses
insert into StudentCourses
values (1,1,'jsmith');

insert into StudentCourses
values (2,1,'mjones');

-- Update the Student data
update Students set Email = 'john.smith@students.umuc.edu'
where PSUsername = 'jsmith';

update Students set Email = 'mary.jones@students.umuc.edu'
where PSUsername = 'mjones';

update Students set Email = 'jeff.parsons@students.umuc.edu'
where PSUsername = 'jparsons';

-- delete the Parsons record
delete from Students
where PSUsername = 'jparsons';
```

2. If not already connected to MySQL, connect to your database and use the sdev300 database. Then copy and paste the SQL code into the MySQL command line as shown in figure 8.

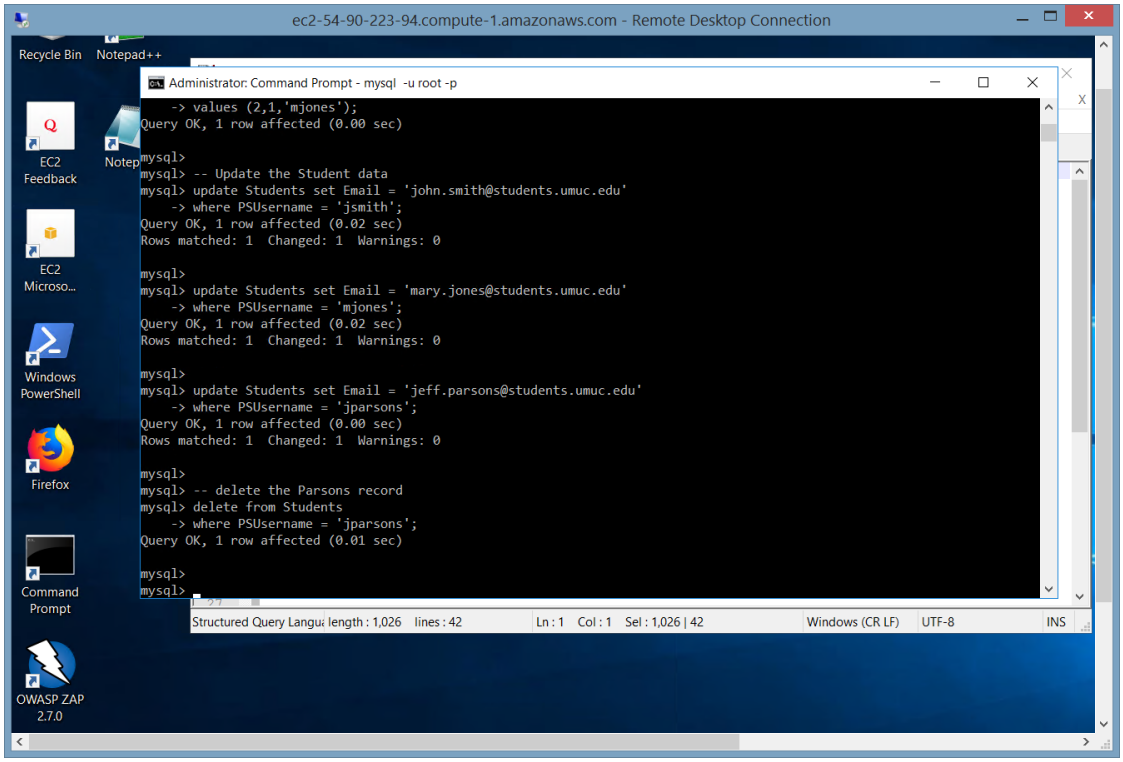

Figure 8 Changing the Table Data

Use the select * from tablename to show the contents for each table as shown in figure 9.

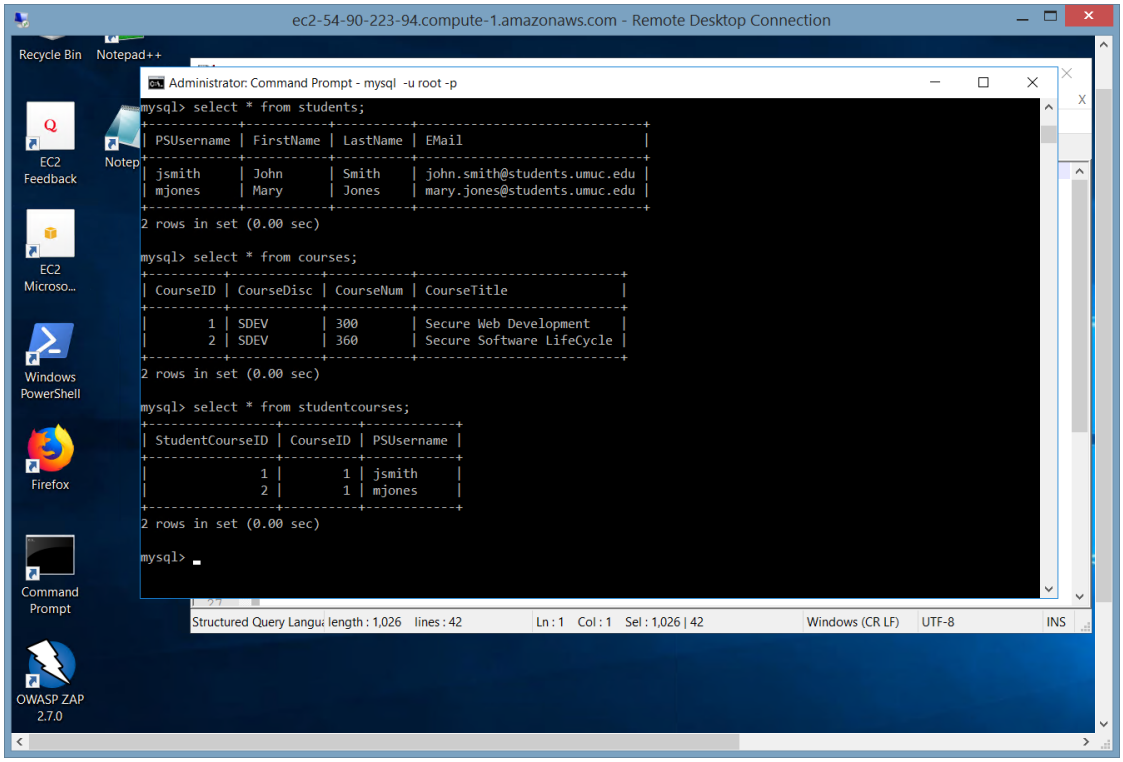

Figure 9 Selecting from each table

As you review the script and experiment by inserting, updating and deleting your own unique records, you should note following:

a. The primary key is a critical element and used to find records to update as well as delete. Notice the where clause is based on the primary key. When you build your own statements, be sure to consider this best practice.
b. The update statement uses the **set** clause. If you wanted to update multiple columns you would just use a comma between each new update. (e.g. set Email='NewEmail', lastname='NewLastname')
c. Be cautious with both deletes and updates. Be sure you use where clauses to filter to the specific rows or rows you want to modify or delete. Disastrous consequences could occur if you don't take caution here. For example, if you don't put the where clause for the updates, you could potentially change every record with your new data.

Be sure to experiment with multiple insert, update and delete statements to become comfortable with creating and updating data in your tables.

Part 4 Create and execute SQL Select statements and simple joins on MySQL tables

Once tables have been created and data populated, you can query the tables using the Select statement. The Select statement has many clauses, the examples below will emphasis the where and order by clauses.

1. Using Notepad++, copy and paste the following SQL code into a file named QueryTables.sql

```
-- Select all records and columns
select * from Students;

select * from Courses;

select * from StudentCourses;

-- Use the Where clause
select Email from Students
where PSUsername = 'jsmith';

select * from Courses
where CourseDisc Like ('SD%');

select PSUsername from StudentCourses
where CourseID = 1;

-- Order by
select * from Students
Order by LastName;

select * from StudentCourses
order by CourseID;

-- Joins to get more details
select A.PSUsername, CourseDisc from
```

```
StudentCourses A, Courses B, Students C
where A.PSUsername =C.PSUsername
and A.CourseID = B.CourseID;

select A.PSUsername, CourseDisc,CourseNum, CourseTitle from
StudentCourses A, Courses B, Students C
where A.PSUsername =C.PSUsername
and A.CourseID = B.CourseID
order by B.CourseID,A.PSUsername;
```

There are many ways and variations on a theme joining tables. Some joins can be quite complex. Joins that include multiple tables can be quite time consuming without proper analysis and design of SQL indices. However; since query optimization is not the focus of the course, we will acknowledge it is critical to an efficiently designed database application and use at most simple two table joins for most examples in this course.

Run the queries by copying and pasting into the MySQL command prompt as shown in figure 10.

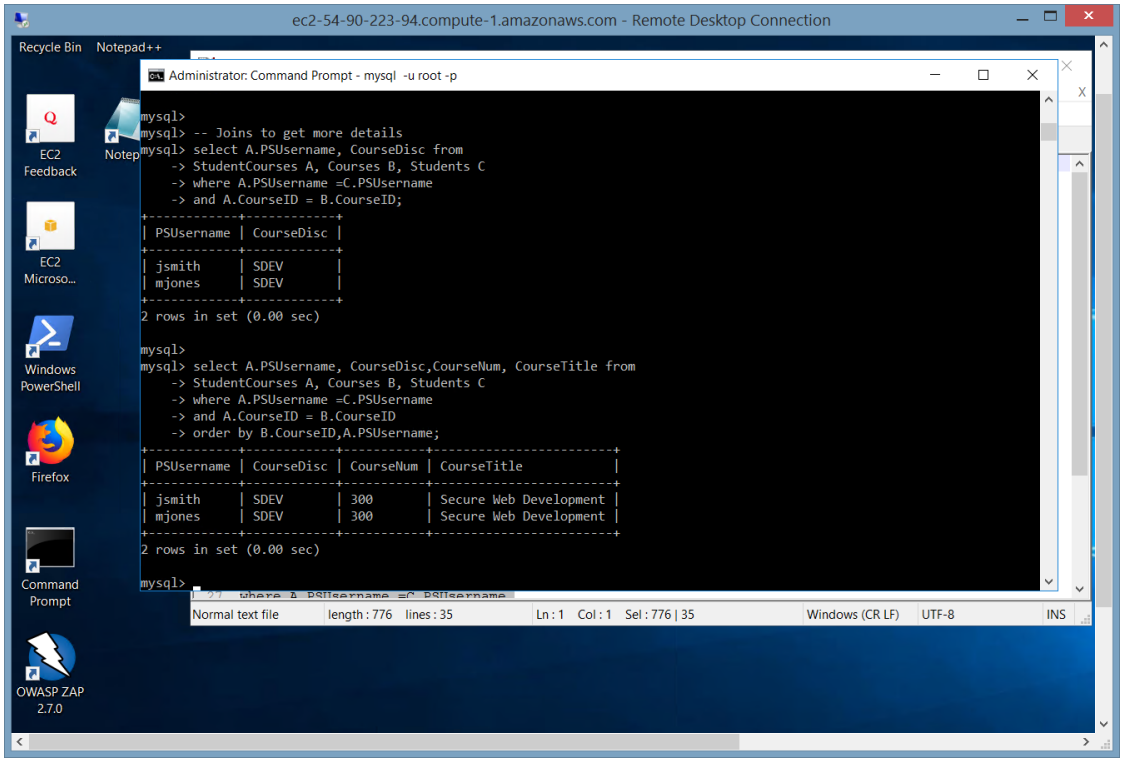

Figure 10 Running Joins and Select Queries

You should execute one or two queries at a time so you can experiment and analyze the results for each query.

Always test your SQL queries before implementing in Web application. You should confirm the expected results are output for each query.

The MySQL reference has more options to consider for becoming confident using SQL. As you experiment and analyze the SQL statements and results note the following:

a. Where clauses are critical to filtering and returning the exact rows you want.

b. If you use the **Like** clause you can also use the wild card character ('%') to provide results for any character.
c. You can use the Order by clause with multiple columns by using a comma between each column.
d. Joins can be tricky. For this course, we will work to keep no more than 3 table joins. Notice the pattern you need to join each table on the column that has the identical column. (e.g. PSUsername is found in both Students and StudentCourses). You may need to reference the alias (e.g. A or B in this case) to remove redundant namings. This is why we had to use order by B.CourseID, A.PSUsername. If we just used CourseID or PSUsername, the query would not know which table column to use. Joins Query statements may take some extra practice to become comfortable with.

Lab submission details:

As part of the submission for this Lab, you will design your own tables and populate them with data based on the following requirements. For each of the requirements, be sure to save the specific SQL statements that you used. Please label each SQL statement corresponding to the numbered requirements below:

1. Create a table named Faculty to store EMPLID, first name, last name, email, and year of birth, and Hire date. You should select the appropriate data types, sizes and constraints for the table. Hint: All tables need a Primary key.
2. Create a table named Courses to store CourseID, discipline name (e.g. SDEV), course number (e.g. 300), number of credits (e.g. 3), date first offered (e.g. June 10, 2010) and course title. You should select the appropriate data types, sizes and constraints for the table.
3. Create a table named FacultyCourses to store the Faculty and the Courses they have taught. You should design the table based on the Faculty and Courses tables you previously created. (Hint: Use Primary and Foreign key relationships)
4. Create Insert statements to populate at least 20 faculty records, 20 Course records, and 25 FacultyCourses records.
5. Create an update statement to update all Courses to 6 credits
6. Create an update statement to update any Faculty with a year of birth of 1994 to change it to 1993.
7. Write an appropriate SQL statement to delete any Faculty record whose Last name starts with the letter 'R' or the letter 'S'. (Hint: this should only be one SQL statement not two.)
8. Write an appropriate SQL statement to delete any Course record that was first offered in 2004
9. Use appropriate select statements to display all records in all 3 tables. The Faculty query should display the Faculty by last name in descending order and Course query should display the courses in ascending order by course title. The display order of the FacultyCourses table is not specified. Hint: you should create three separate select statements to satisfy this requirement.
10. Create a select statement to display all Faculty who have not taught at least 3 courses.
11. Create a select statement to display all Courses offered before 1999.
12. Use select and appropriate joins to display **all** columns from the Faculty and Course tables for each Faculty and Course in the FacultyCourse table. Note: this will be a 3-table join.

Create screen captures showing the successful running of each your scripts. Be sure to fully describe each screen capture. In addition, your SQL script should have comments listing the purpose of each group of similar statements.

For your deliverables, you should submit a zip file containing your word document (or PDF file) with screen captures of the application running successfully along with your SQL script file. You should include only one SQL script that includes all of your SQL statements.

Include your full name, class number, section, professor name, and date in the document.

Note: be sure to include your SQL file and Word (PDF) document for in your submission.

Grading Rubric:

Attribute	**Meets**
SQL statements	**85 points** Create a table named Faculty to store EMPLID, first name, last name, email, and year of birth, and Hire date. You should select the appropriate data types, sizes and constraints for the table. (5 points) Create a table named Courses to store CourseID, discipline name (e.g. SDEV), course number (e.g. 300), number of credits (e.g. 3), date first offered (e.g. June 10, 2010) and course title. You should select the appropriate data types, sizes and constraints for the table. (5 points) Create a table named FacultyCourses to store the Faculty and the Courses they have taught. You should design the table based on the Faculty and Courses tables you previously created. (10 points) Create Insert statements to populate at least 20 faculty records, 20 Course records, and 25 FacultyCourses records. (10 points) Create an update statement to update all Courses to 6 credits. (5 points) Create an update statement to update any Faculty with a year of birth of 1994 to change it to 1993. (5 points) Write an appropriate SQL statement to delete any Faculty record whose Last name starts with the letter 'R' or the letter 'S'. (5 points) Write an appropriate SQL statement to delete any Course record that was first offered in 2004. (5 points) Use appropriate select statements to display all records in all 3 tables. The Faculty query should display the Faculty by last name in descending order and Course query should display the courses in ascending order by course title. The display order of the FacultyCourses table is not specified. (10 points)

	Create a select statement to display all Faculty who have not taught at least 3 courses. (10 points) Create a select statement to display all Courses offered before 1999. (5 points) Use select and appropriate joins to display **all** columns from the Faculty and Course tables for each Faculty and Course in the FacultyCourse table. Note: this will be a 3-table join. (10 points)
Documentation and submission	**15 points** Fully describes each screen capture. (10 points) SQL script should have comments listing the purpose of each group of similar statements. (3 points) Include your full name, class number, section, professor name, and date in the document. (2 points) **SQL file not provided (-100)** **PDF or Word File with screen captures not provided (-100)**