# DAD 220 Database Documentation Template

Tammie Pease

## Step One: Create a Database

1. Navigate to your online integrated development environment (IDE). Here, you will need to write the proper SQL commands in command line to create tables that demonstrate relationships based on the entity relationship diagram. List and record the SQL commands that you used to complete this step here:

>mysql

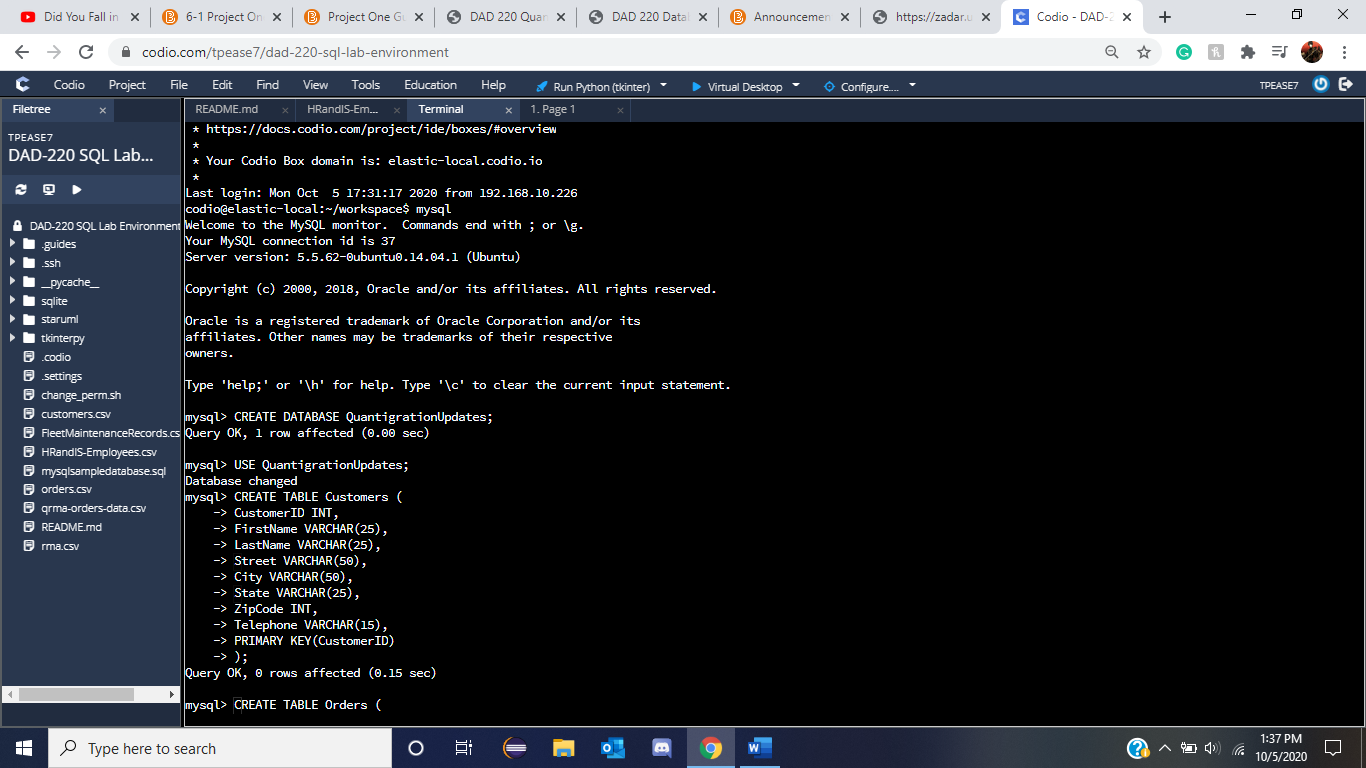
Mysql allows the users to access Codio and enter the IDE.   
“**ADDITIONAL NOTE:**Prompt 1 of this assignment is simply asking you to access the Codio development environment.  It is specifying that the Codio environment is where you will be performing the work.  You do not have to submit anything to address this prompt.” So I believe this was the main thing that was needed in this scenario and able to go into the main events.

1. Create a database schema called *QuantigrationUpdates*. List out the database name. Provide the SQL commands you ran against MySQL to successfully complete this in your answer:

>CREATE DATABASE QuantigrationUpdates;

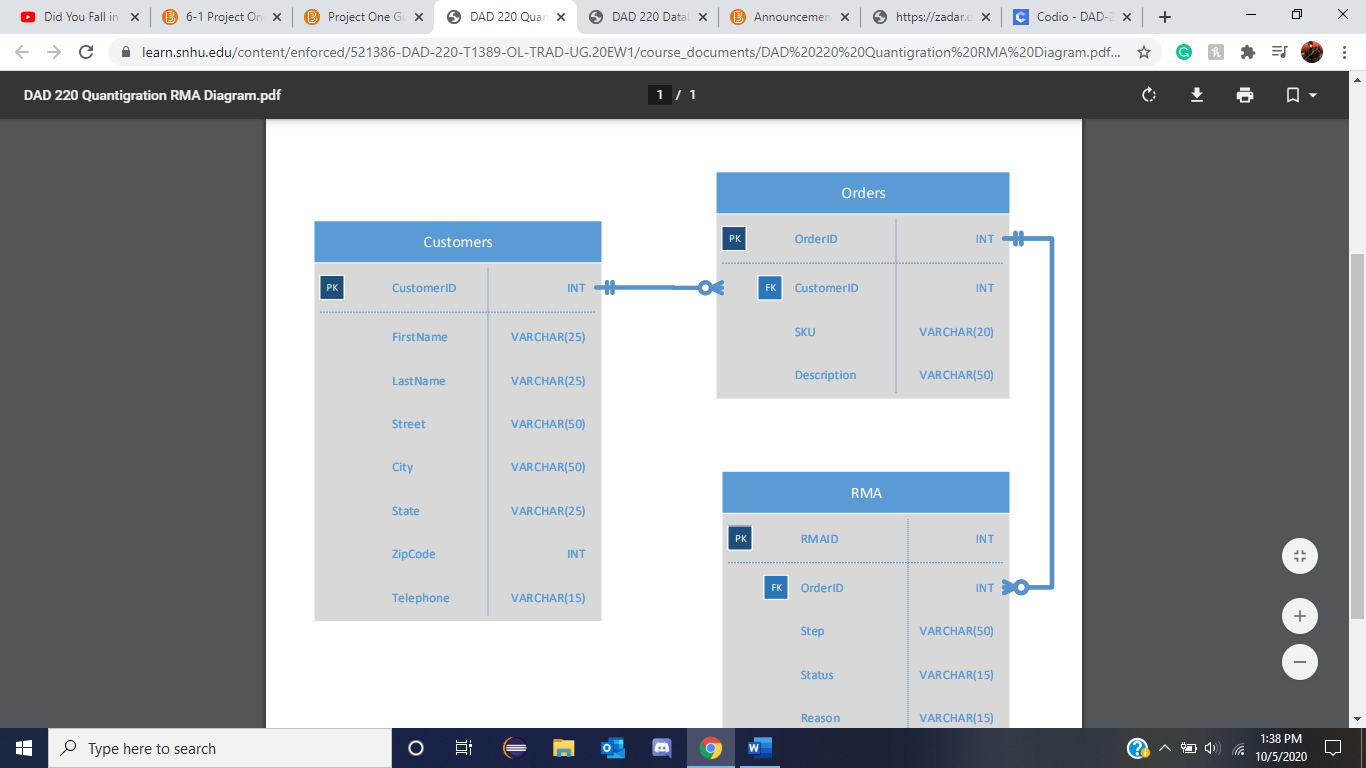
>USE QuantigrationUpdates;

Beginning prompt in order to create a database named *QuantigrationUpdates* then use this database in order to manipulate it and add the tables that are required.



Database changed shows that the access into its database was allowed and the creation was confirmed when the Query was ‘OK’.

1. Using the ERD as a reference, **create the following tables with the appropriate attributes and keys**:
   1. A table named **customers** in the *QuantigrationUpdates* database as defined on the project ERD. Provide the SQL commands you ran against MySQL to complete this successfully in your answer:



From the project ERD gives us the table’s contents along with the name that would be specified. Noted with this is that CustomerID has PK for primary key. Giving us the code:

CREATE TABLE Customers (

CustomerID INT,

FirstName VARCHAR(25),

LastName VARCHAR(25),

Street VARCHAR(50),

City VARCHAR(50),

State VARCHAR(25),

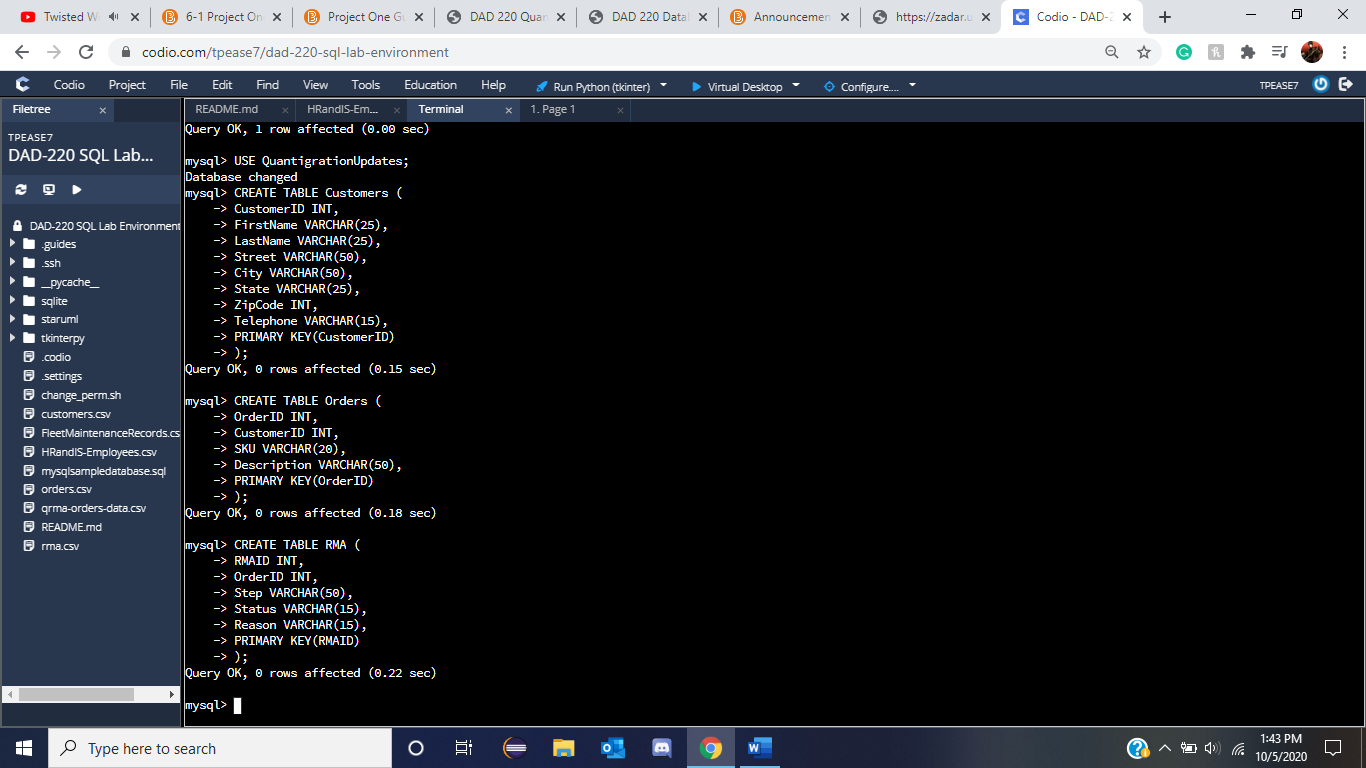
ZipCode INT,

Telephone VARCHAR(15),

PRIMARY KEY(CustomerID)

);

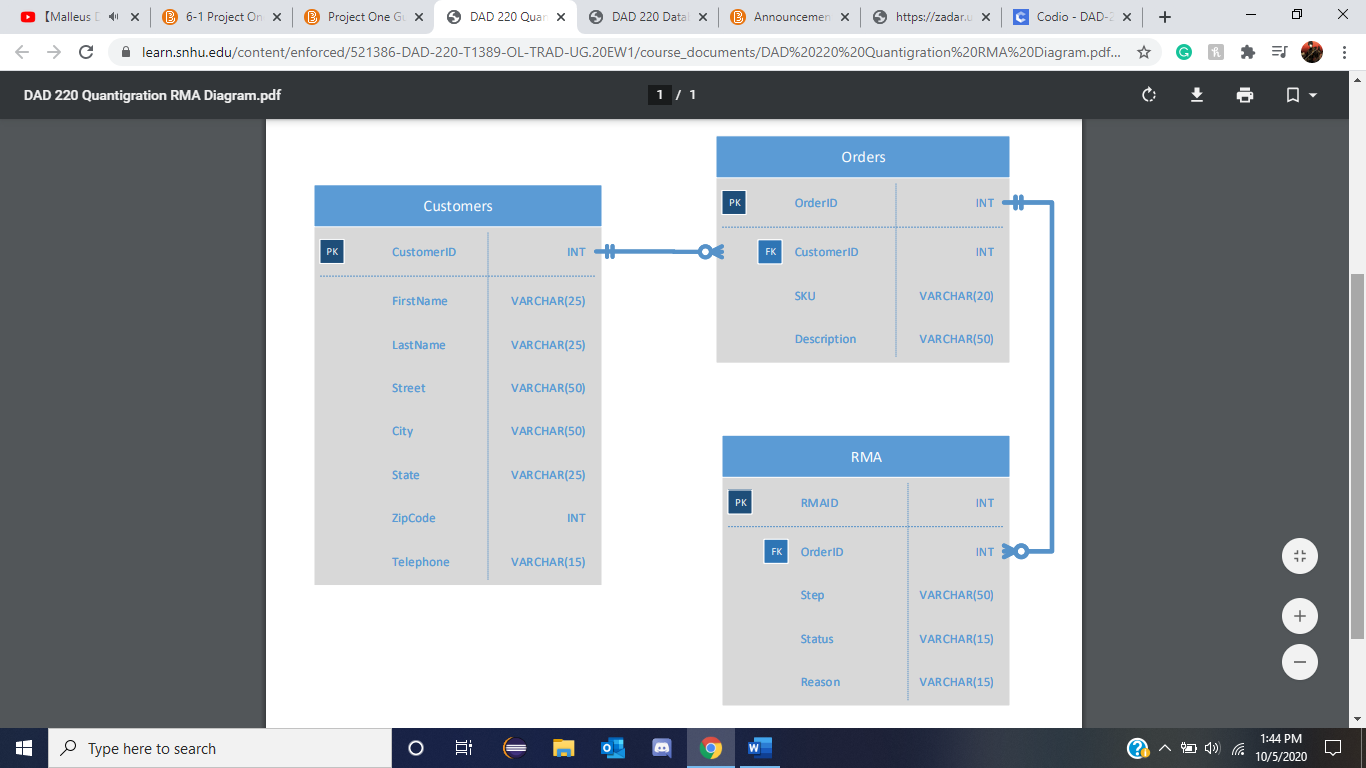
Confirmed within



It should be noted that with this code the Query was confirmed to be alright and allowed through along with the idea that this was safe and PRIMARY KEY(CustomerID) gives us that one as a primary key status.

* 1. A table named **orders** in the *QuantigrationUpdates* database as defined on the project ERD. Provide the SQL commands you ran against MySQL to complete this successfully in your answer:

Similar to ‘a’ in the above prompt when looking at the ERD of the project we can notice the required columns but also that there is a primary key with OrderID and also that there is a Foreign Key of CustomerID.



Giving us the usable codes of:

CREATE TABLE Orders (

OrderID INT,

CustomerID INT,

SKU VARCHAR(20),

Description VARCHAR(50),

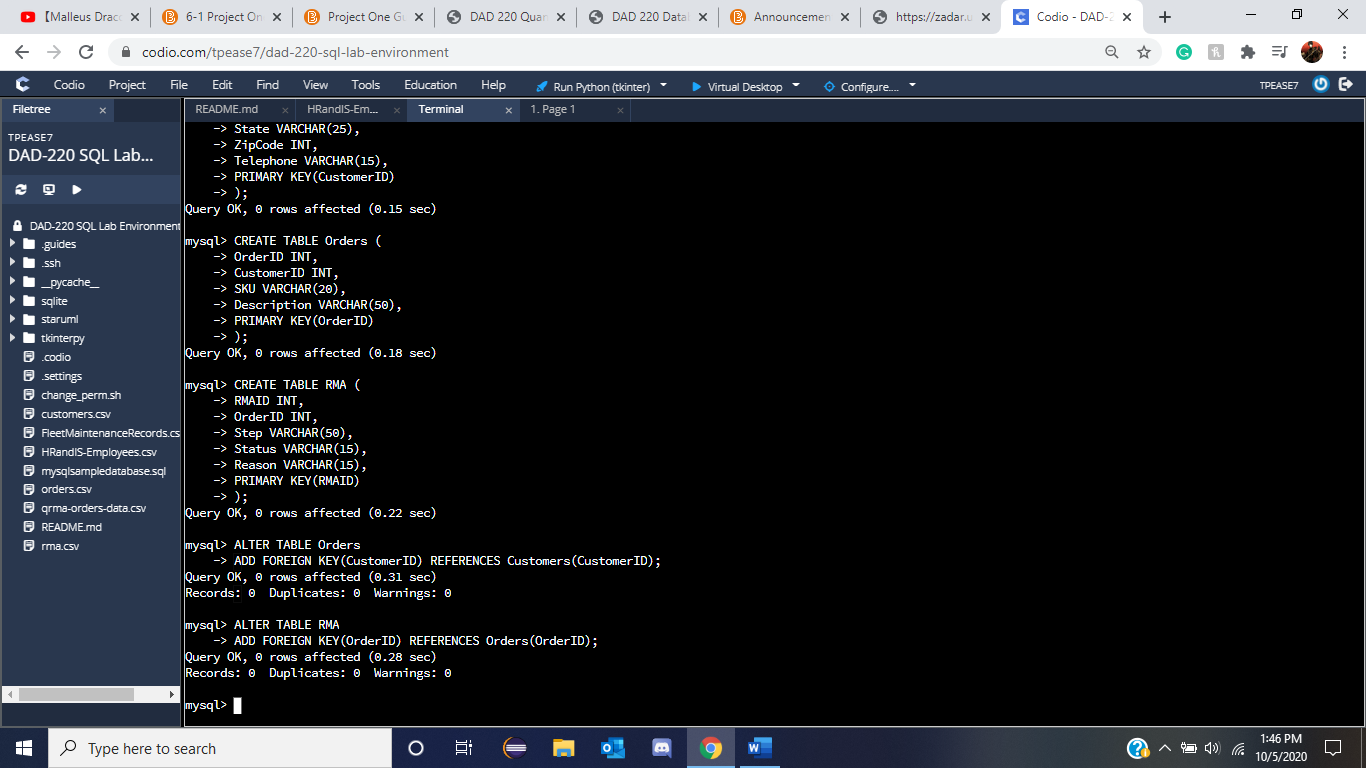
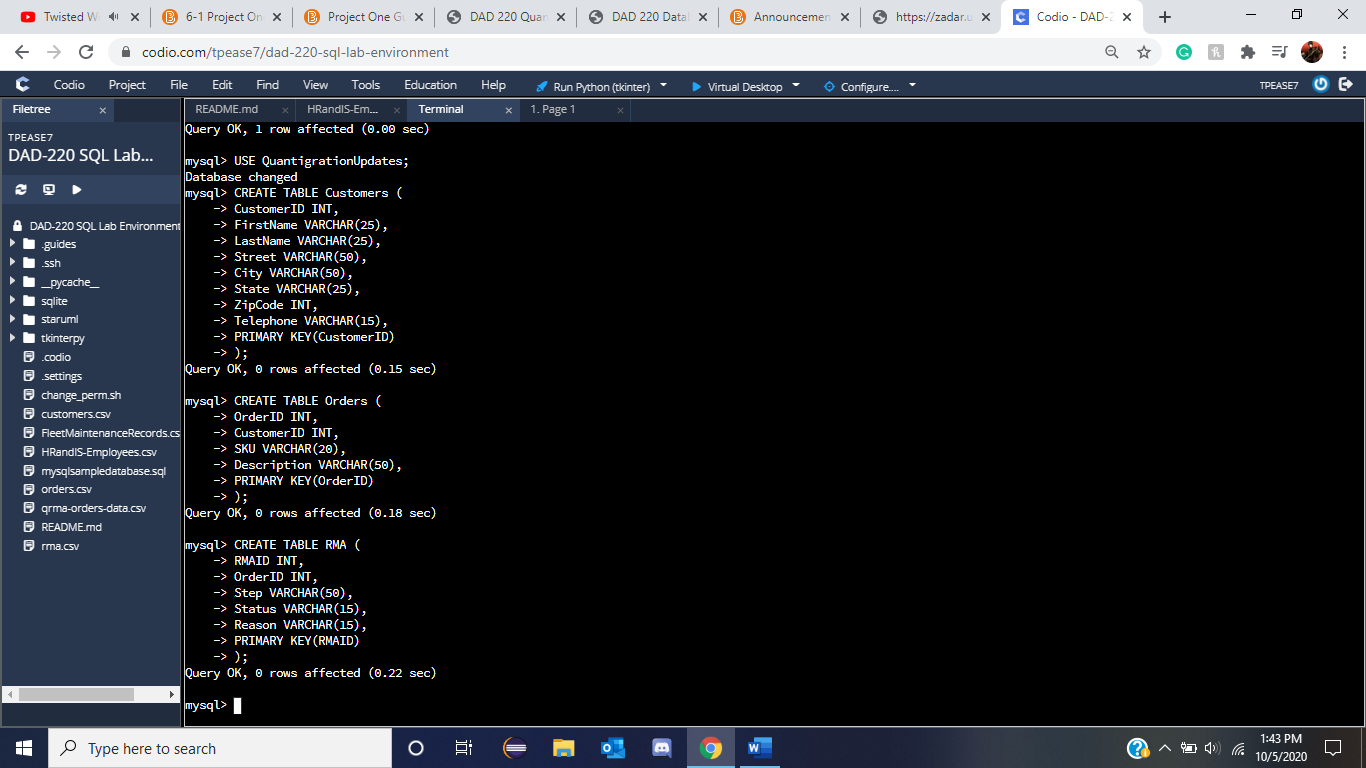
PRIMARY KEY(OrderID)

);

And to add the foreign key:

ALTER TABLE Orders

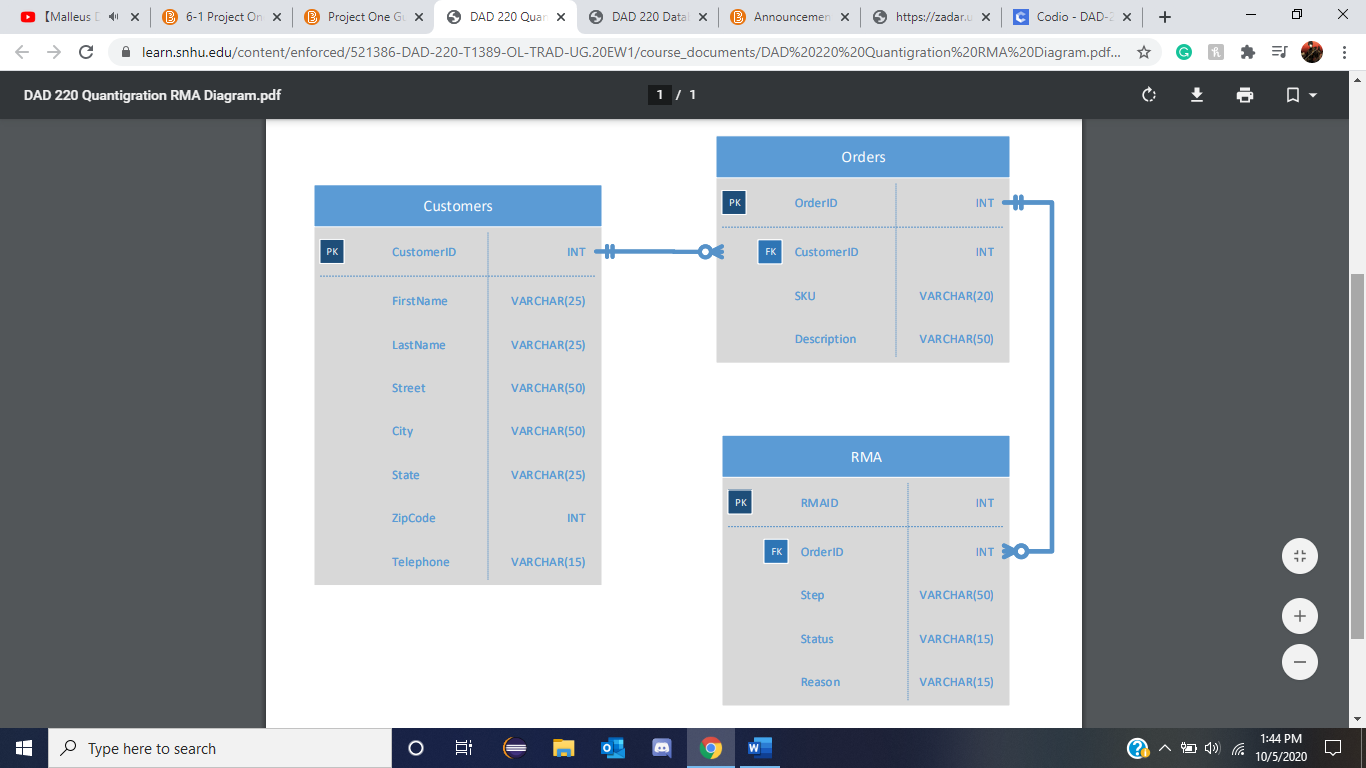
ADD FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID);



The create table Orders and the setting of PK OrderID are passed through via the guidelines however Alter Table Orders and ADD FOREIGN KEY CustomerID to reference the Customer’s table CustomersID allows them to be connected as shown in the main ERD files supplied fulfilling the requirements specified.

* 1. A table named **rma** in the *QuantigrationUpdates* database as defined on the project ERD. Provide the SQL commands you ran against MySQL to complete this successfully in your answer:

The ERD files show us the requirements with RMAID being the primary key and Order ID being the foreign key pulled from the orders Table.



With this information we can pull the idea that the following codes will be:

CREATE TABLE RMA (

RMAID INT,

OrderID INT,

Step VARCHAR(50),

Status VARCHAR(15),

Reason VARCHAR(15),

PRIMARY KEY(RMAID)

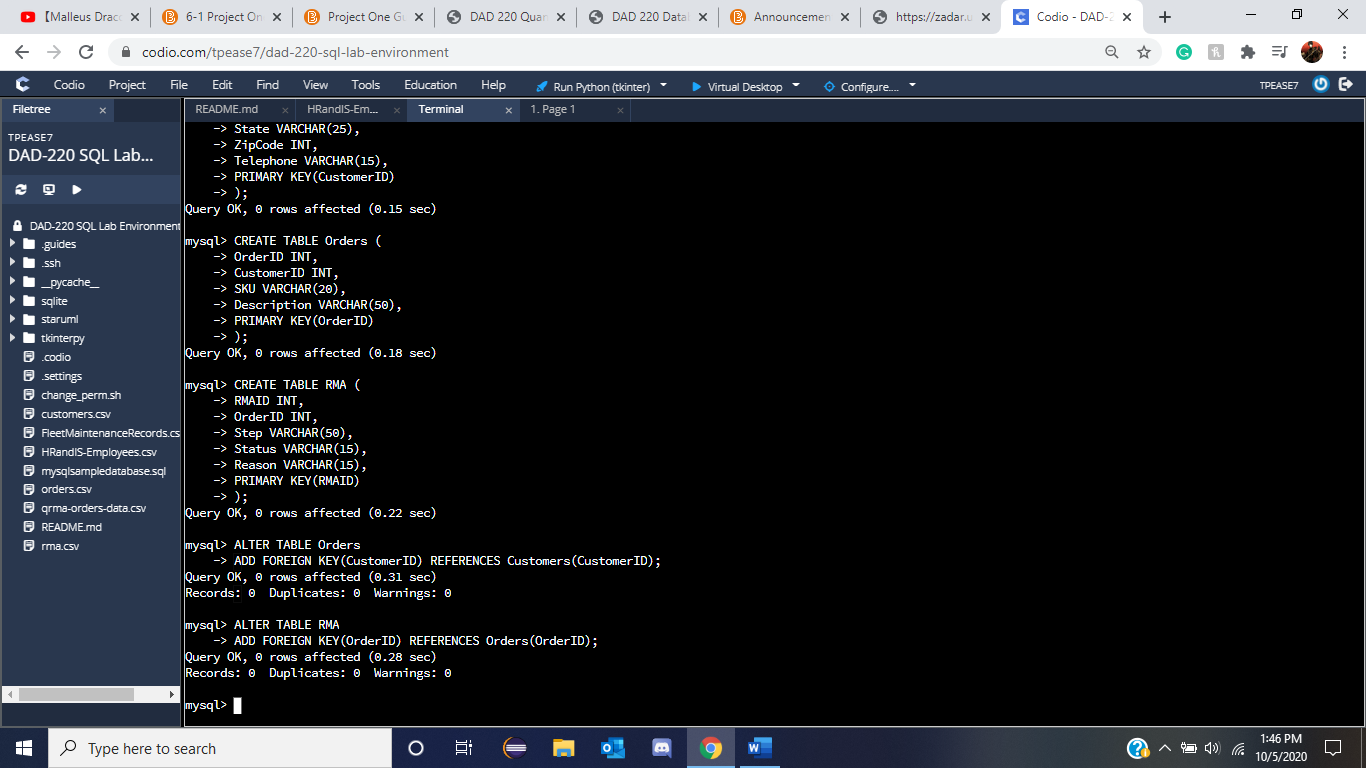
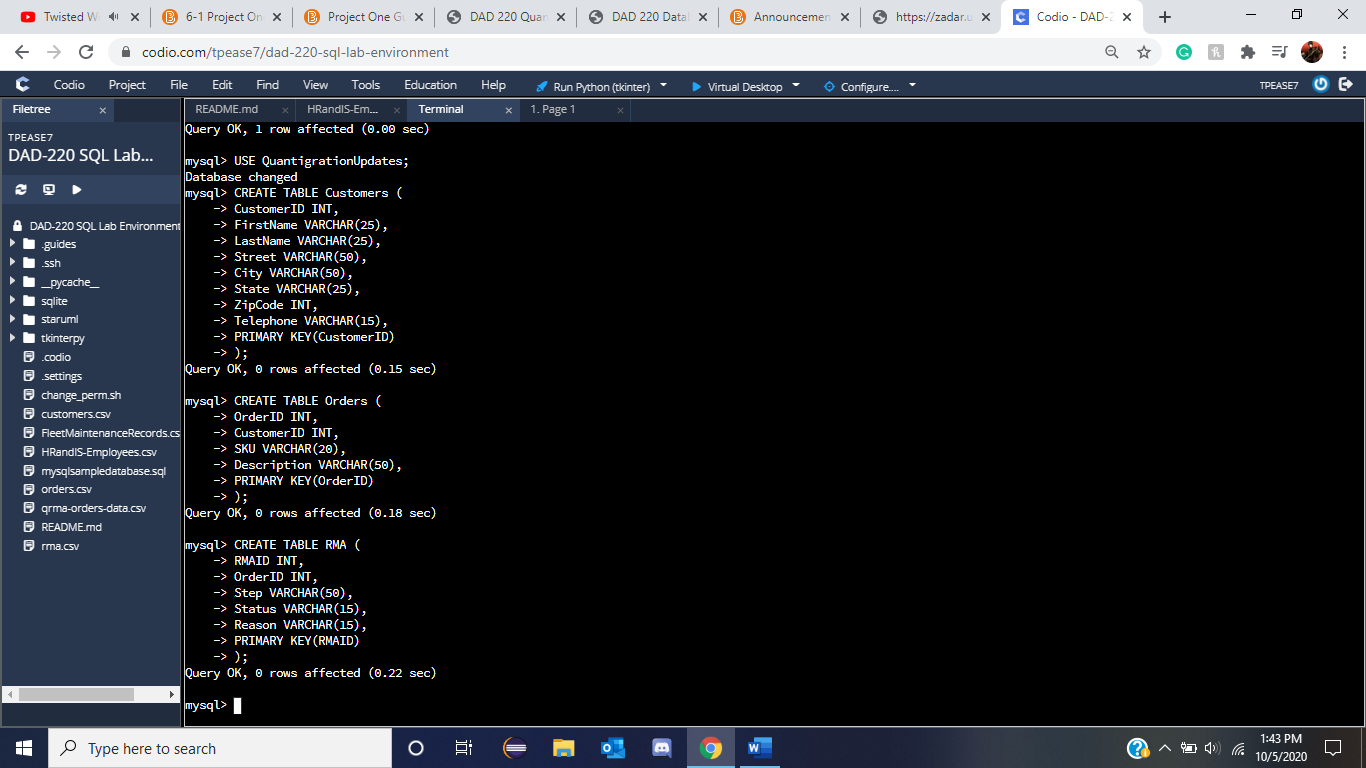
);

And for the foreign key addition similar to part Step One, Part 3 section b

ALTER TABLE RMA

ADD FOREIGN KEY(OrderID) REFERENCES Orders(OrderID);

Which gives us in mysql:



Creating the basis for the information that we will use and keep within these three tables shown in the ERD file giving us the basis we can start loading and manipulating information for the databases to keep.

1. **Update your existing table** from “Customer” to “Collaborator” using SQL based on this change in requirements. Provide the SQL commands you ran against MySQL to complete this successfully in your answer:

As stated in the Brightspace announcement: “Please do not rename the **Customers**table as specified in the provided **Database Documentation Template**.  Instead, create the **Collaborators**view as specified in the provided code file.”   
  
With this similar to the methods of creation in one of the earlier modules we will use the code to instead create something else without renaming it but allowing for a ‘Collaborators’ view to be made and invested into the database we have created here. The code itself will be:

CREATE VIEW Collaborators AS

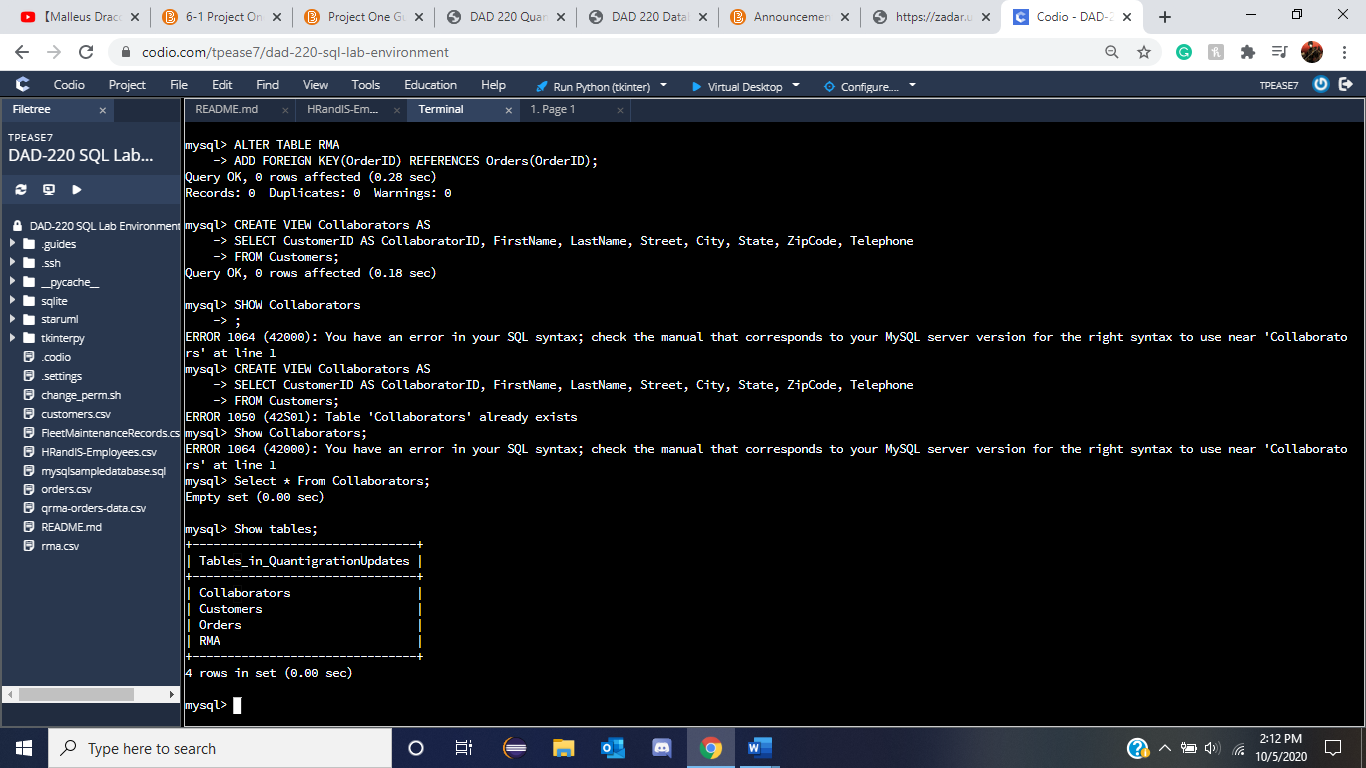
SELECT CustomerID AS CollaboratorID, FirstName, LastName, Street, City, State, ZipCode, Telephone

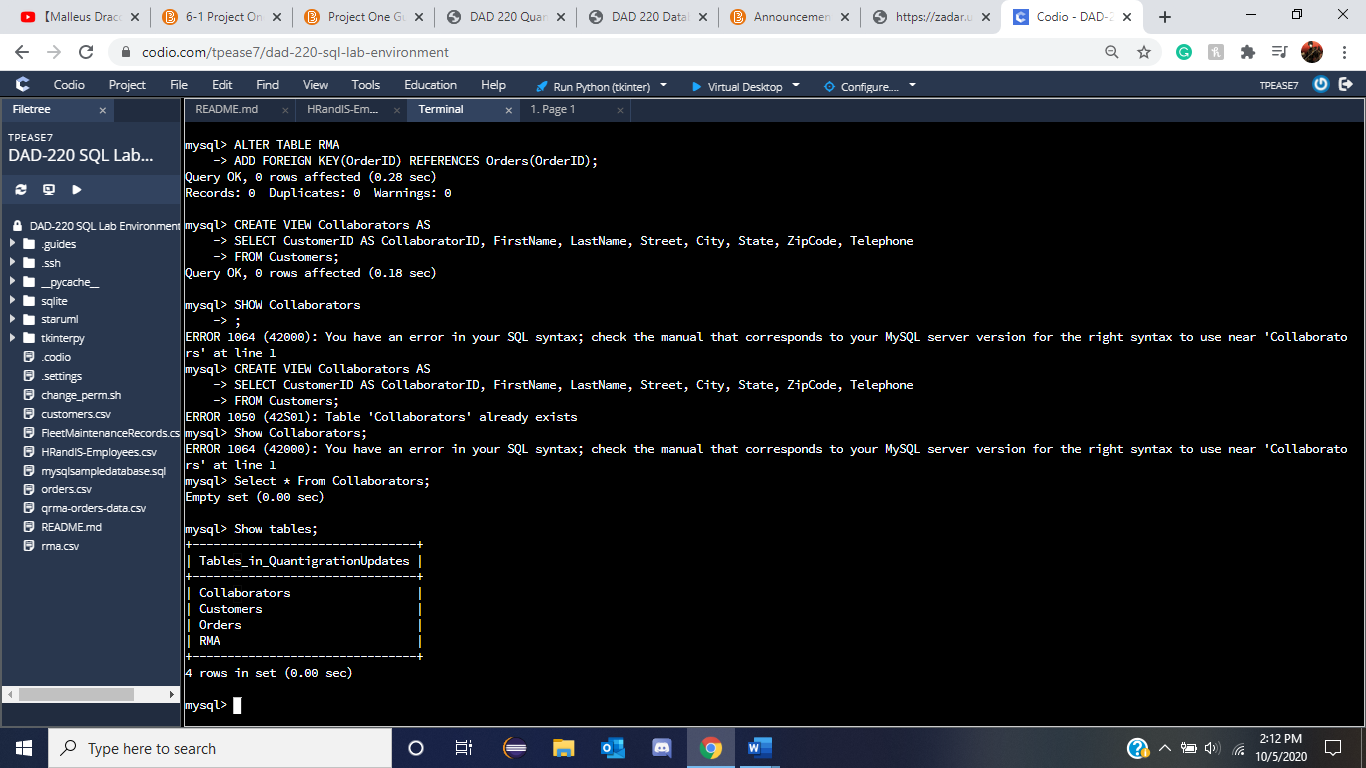
FROM Customers;

In which, we are then able to test if this code has been created and stayed with

Show tables;

That then confirms that this table of Collaborators has been made and is able to be used.

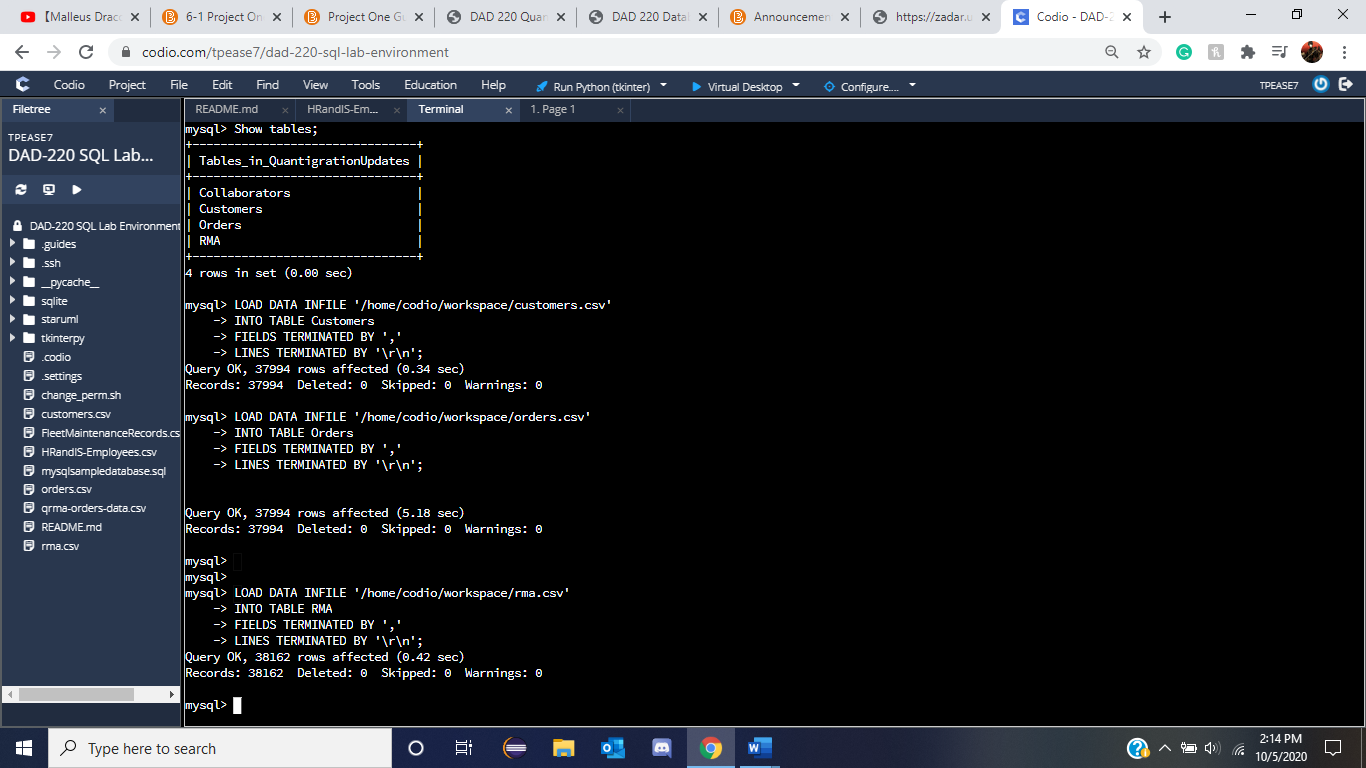




## Step Two: Load and Query the Data

1. **Import the data from each file into tables**.
   * Use the *QuantigrationUpdates* database, the three tables you created, and the three CSV files preloaded into Codio.
   * Use the import utility of your database program to load the data from each file into the table of the same name. You will perform this step three times, once for each table.

Code provided within the files are:



LOAD DATA INFILE ‘/home/codio/workspace/customers.csv’

INTO TABLE Customers

FIELDS TERMINATED BY ‘,’

LINES TERMINATED BY ‘\r\n’;

LOAD DATA INFILE ‘/home/codio/workspace/orders.csv’

INTO TABLE Orders

FIELDS TERMINATED BY ‘,’

LINES TERMINATED BY ‘\r\n’;

LOAD DATA INFILE ‘/home/codio/workspace/rma.csv’

INTO TABLE RMA

FIELDS TERMINATED BY ‘,’

LINES TERMINATED BY ‘\r\n’;

Signifying the changes and confirming they have happened into the beginning of Step Two, Part Two.

1. **Write basic queries against imported tables to organize and analyze targeted data.** For each query, replace the bracketed text with a screenshot of the query and its output. You should also include a brief, 1- to 3-sentence description of the output.
   * Write an SQL query that returns the count of orders for customers located only in the city of Framingham, Massachusetts.
     1. How many records were returned?

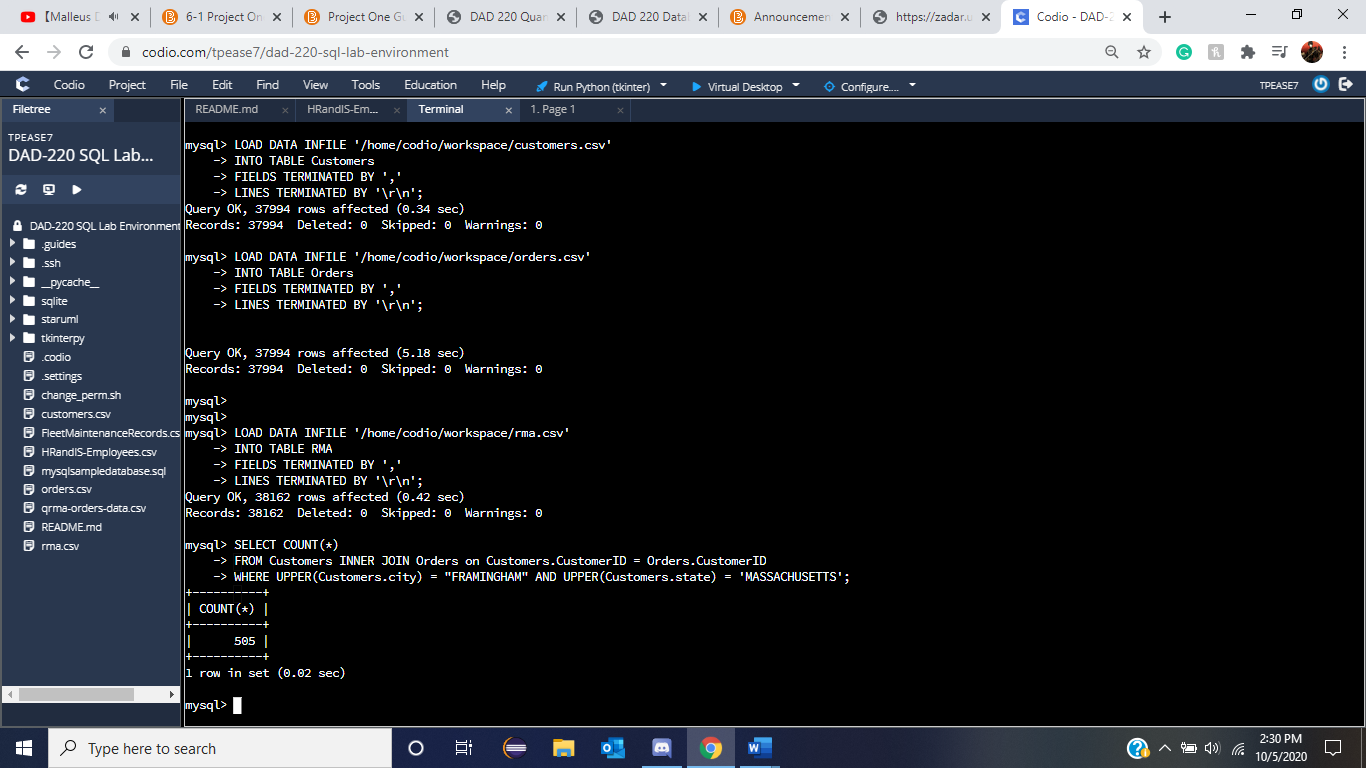
The code used was:

SELECT COUNT(\*)

FROM Customers INNER JOIN Orders on Customers.CustomerID = Orders.CustomerID

WHERE UPPER(Customers.city) = "FRAMINGHAM" AND UPPER(Customers.state) = 'MASSACHUSETTS';

Giving the result of:



With the hints given we needed to use Customer with an Inner Join for the best result with Orders where the customer ID’s lined up as they are both primary keys in their respective tables. After which the two requirements that both needed to be fulfilled stated with the AND was to confirm that both statements had to be true in order to be available. With this we set Customers.city and Customers.state to be Framingham, Massachusetts specifically. Which gave us 505 records.

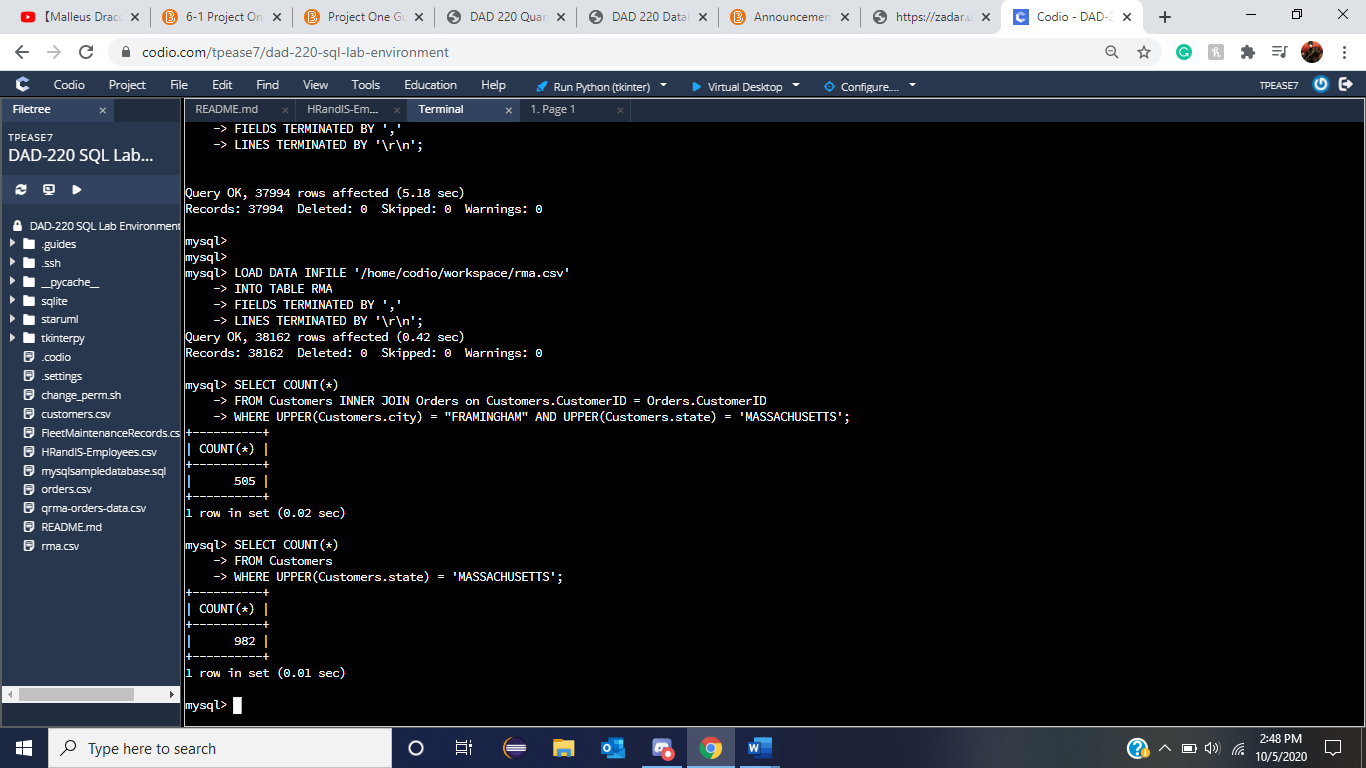
* + Write an SQL query to select all of the customers located in the state of Massachusetts.
    1. Use a WHERE clause to limit the number of records in the customers table to only those who are located in Massachusetts.
    2. Record an answer to the following question: How many records were returned?

SELECT COUNT(\*)

FROM Customers

WHERE UPPER(Customers.state) = 'MASSACHUSETTS';

Similar to the above prompt we needed to pull the number of customers from the state of Massachusetts. Using Upper allows the values to be upper case and compare it to the requirement to avoid failures of strange spellings or organization. With only one requirement we can select the count of (\*) from all the customer table where it meets the requirement. Giving us:



Or 982 files, showing the prior statement’s added Framingham was necessary to avoid the incorrect values. In order to look at all the files we can remove the COUNT and ‘(\*)’ with ‘\*’ which gives us

SELECT \*

FROM Customers

WHERE UPPER(Customers.state) = 'MASSACHUSETTS';



* + Write a SQL query to insert four new records into the orders and customers tables using the following data:
    1. Customers Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CustomerID** | **FirstName** | **LastName** | **StreetAddress** | **City** | **State** | **ZipCode** | **Telephone** |
| 100004 | Luke | Skywalker | 15 Maiden Lane | New York | NY | 10222 | 212-555-1234 |
| 100005 | Winston | Smith | 123 Sycamore Street | Greensboro | NC | 27401 | 919-555-6623 |
| 100006 | MaryAnne | Jenkins | 1 Coconut Way | Jupiter | FL | 33458 | 321-555-8907 |
| 100007 | Janet | Williams | 55 Redondo Beach Blvd | Torrence | CA | 90501 | 310-555-5678 |

The code itself will be:

INSERT INTO Customers VALUES

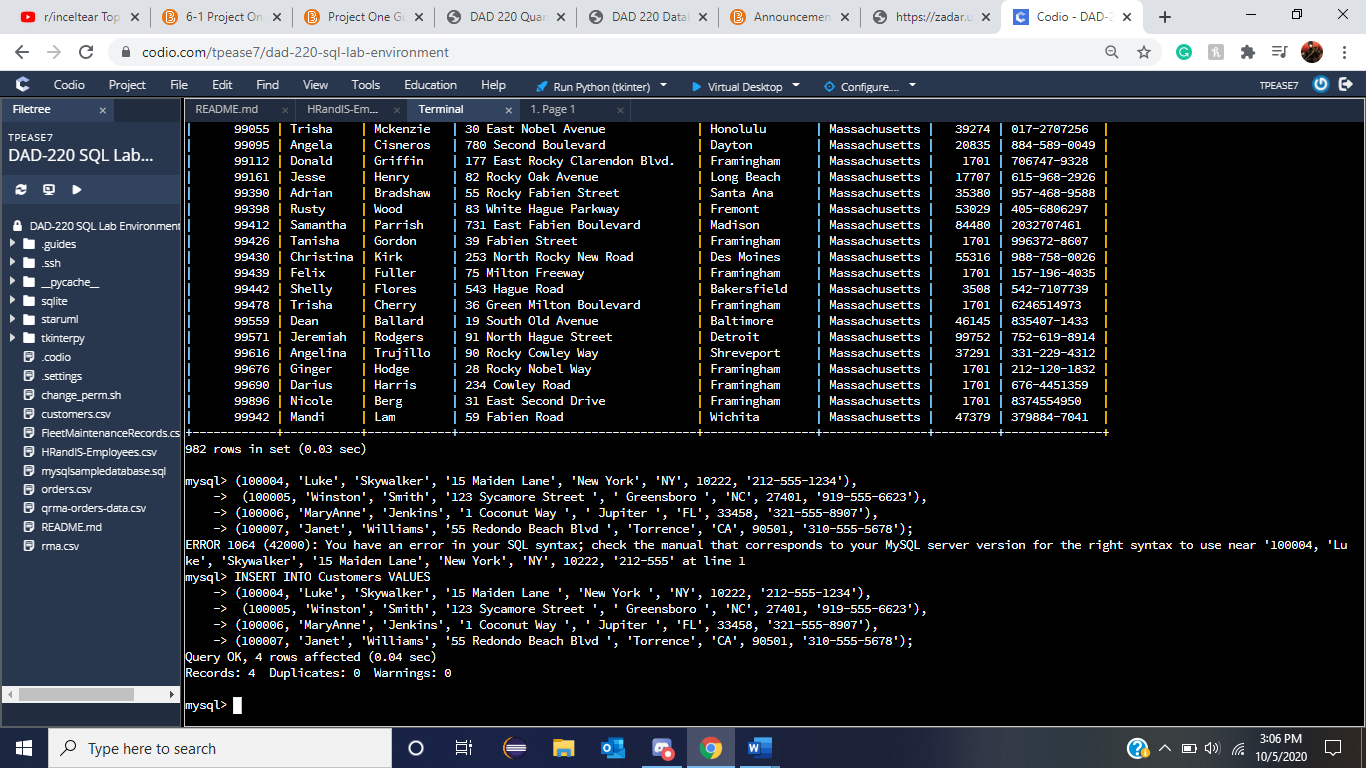
(100004, 'Luke', 'Skywalker', '15 Maiden Lane ', 'New York ', 'NY', 10222, '212-555-1234'),

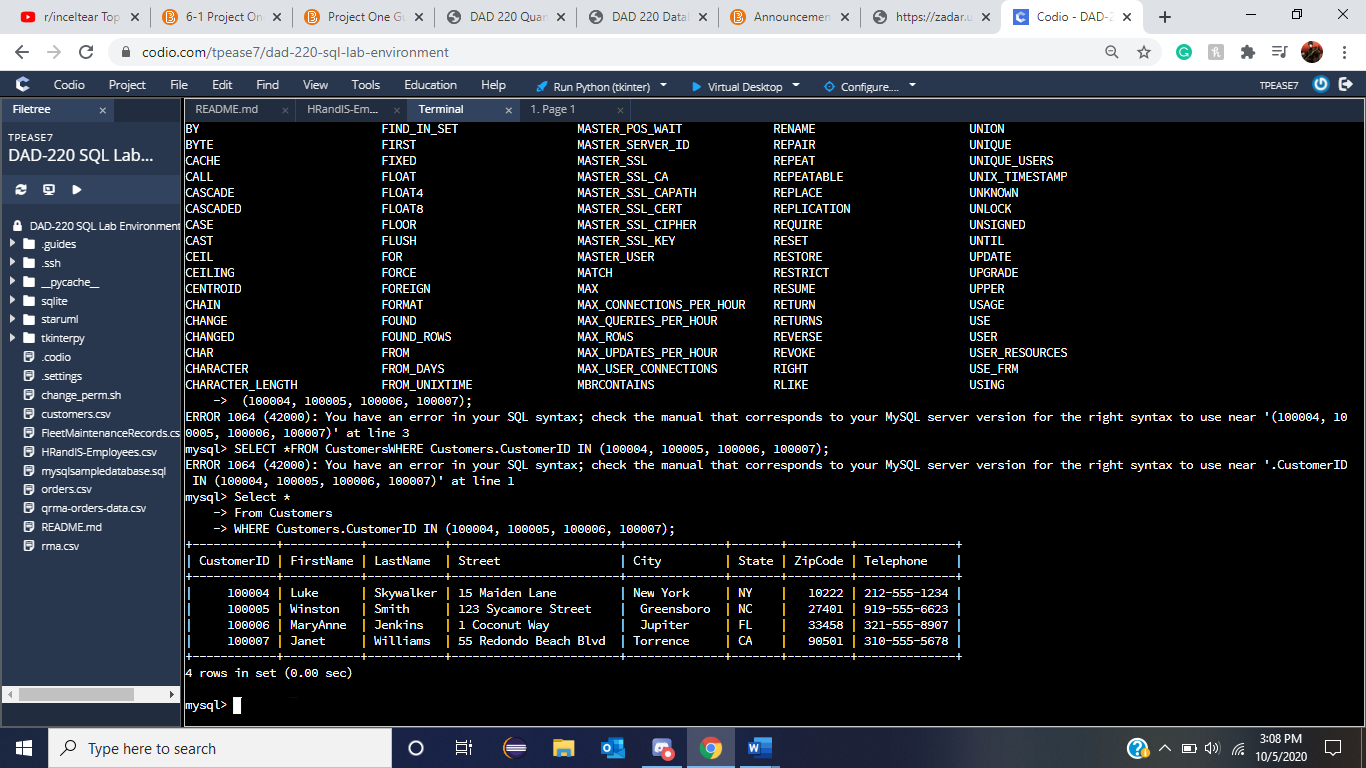
(100005, 'Winston', 'Smith', '123 Sycamore Street ', ' Greensboro ', 'NC', 27401, '919-555-6623'),

(100006, 'MaryAnne', 'Jenkins', '1 Coconut Way ', ' Jupiter ', 'FL', 33458, '321-555-8907'),

(100007, 'Janet', 'Williams', '55 Redondo Beach Blvd ', 'Torrence', 'CA', 90501, '310-555-5678');

With reference to the module 4 lab. We have to insert new values into the Customers table and those Values are ordered within the column as shown above in the new required data. We can then test to see if these values went in correctly with Select Where Customers.ID in those new IDS.





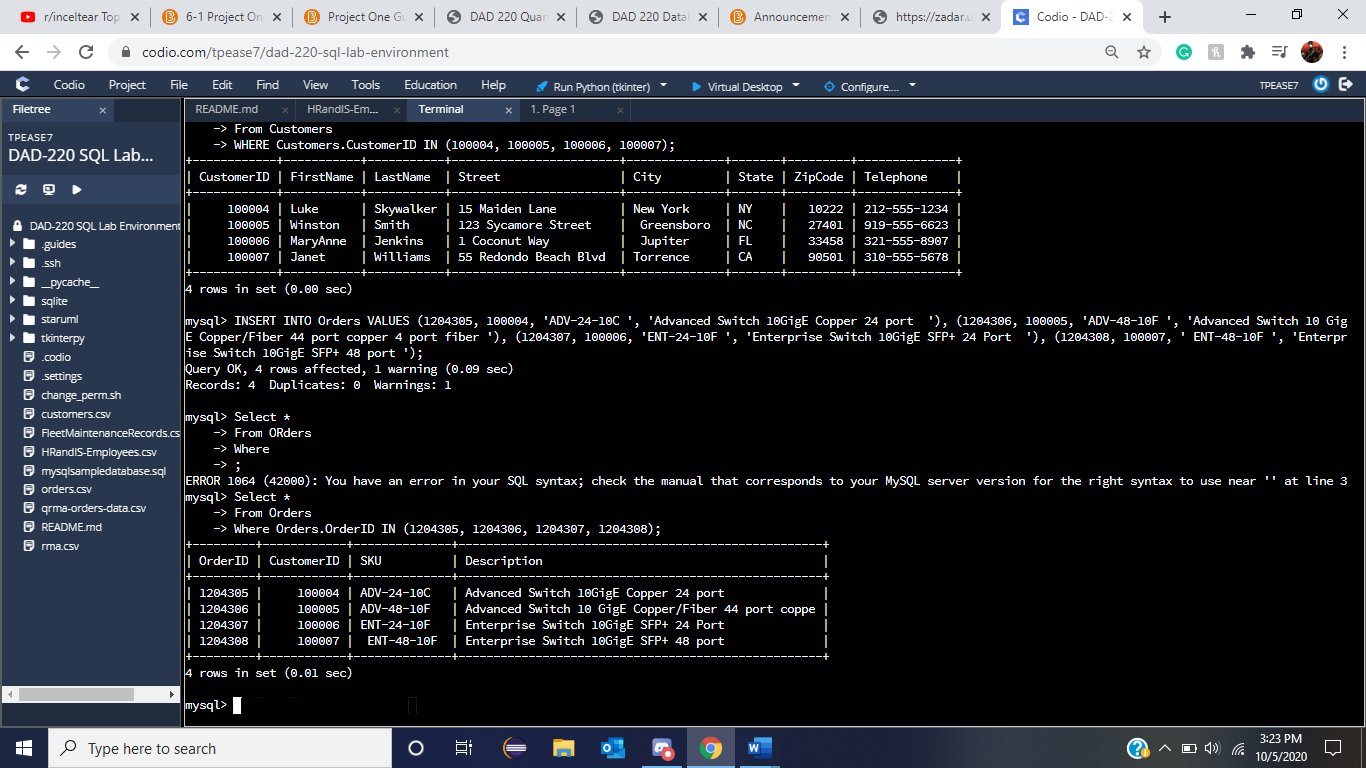
* + 1. Orders Table

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderID** | **CustomerID** | **SKU** | **Description** |
| 1204305 | 100004 | ADV-24-10C | Advanced Switch 10GigE Copper 24 port |
| 1204306 | 100005 | ADV-48-10F | Advanced Switch 10 GigE Copper/Fiber 44 port copper 4 port fiber |
| 1204307 | 100006 | ENT-24-10F | Enterprise Switch 10GigE SFP+ 24 Port |
| 1204308 | 100007 | ENT-48-10F | Enterprise Switch 10GigE SFP+ 48 port |

Similar to the above situation we used the INSERT INTO Orders Values and fulfilled them with the requirements of INT, INT, CHAR, CHAR. Giving us this code:

INSERT INTO Orders VALUES (1204305, 100004, 'ADV-24-10C ', 'Advanced Switch 10GigE Copper 24 port '), (1204306, 100005, 'ADV-48-10F ', 'Advanced Switch 10 GigE Copper/Fiber 44 port copper 4 port fiber '), (1204307, 100006, 'ENT-24-10F ', 'Enterprise Switch 10GigE SFP+ 24 Port '), (1204308, 100007, ' ENT-48-10F ', 'Enterprise Switch 10GigE SFP+ 48 port ');

And this testing code to make sure it went through.



Select \*

From Orders

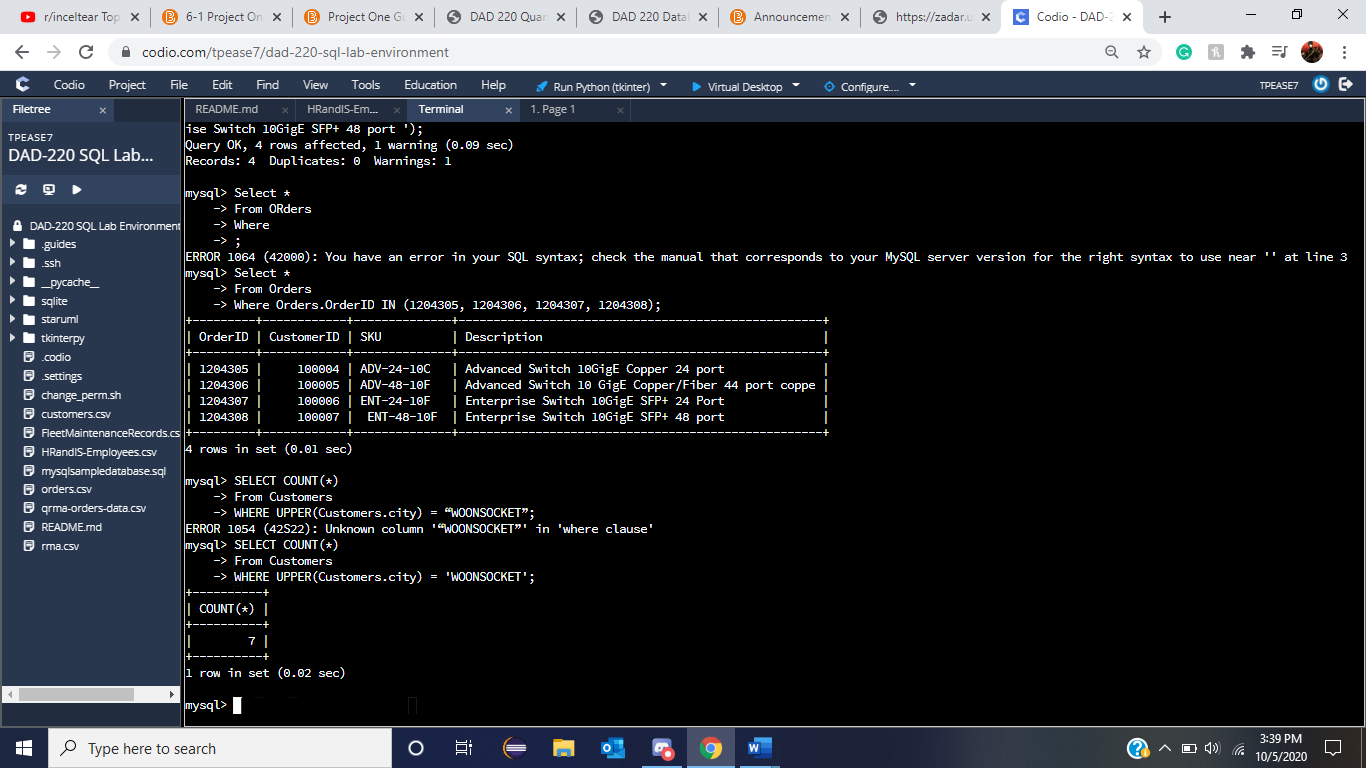
Where Orders.OrderID IN (1204305, 1204306, 1204307, 1204308);

Inserting values is important when it comes to the maintenance and organization of all the records. Because of this, making sure that each of the accepted values is equal to the ones given along with being kept in a certain order without extra specification makes it so that these values within each one will be put in or else it can all be rejected. Testing them out to make sure that these values were placed in correctly allows for the notice and customization of the values for showing that they can be inserted manually and also not just uploaded via other documents.

* + In the customers table, perform a query to count all records where the city is Woonsocket, Rhode Island.
    1. How many records are in the customers table where the field “city” equals “Woonsocket”?

SELECT COUNT(\*)

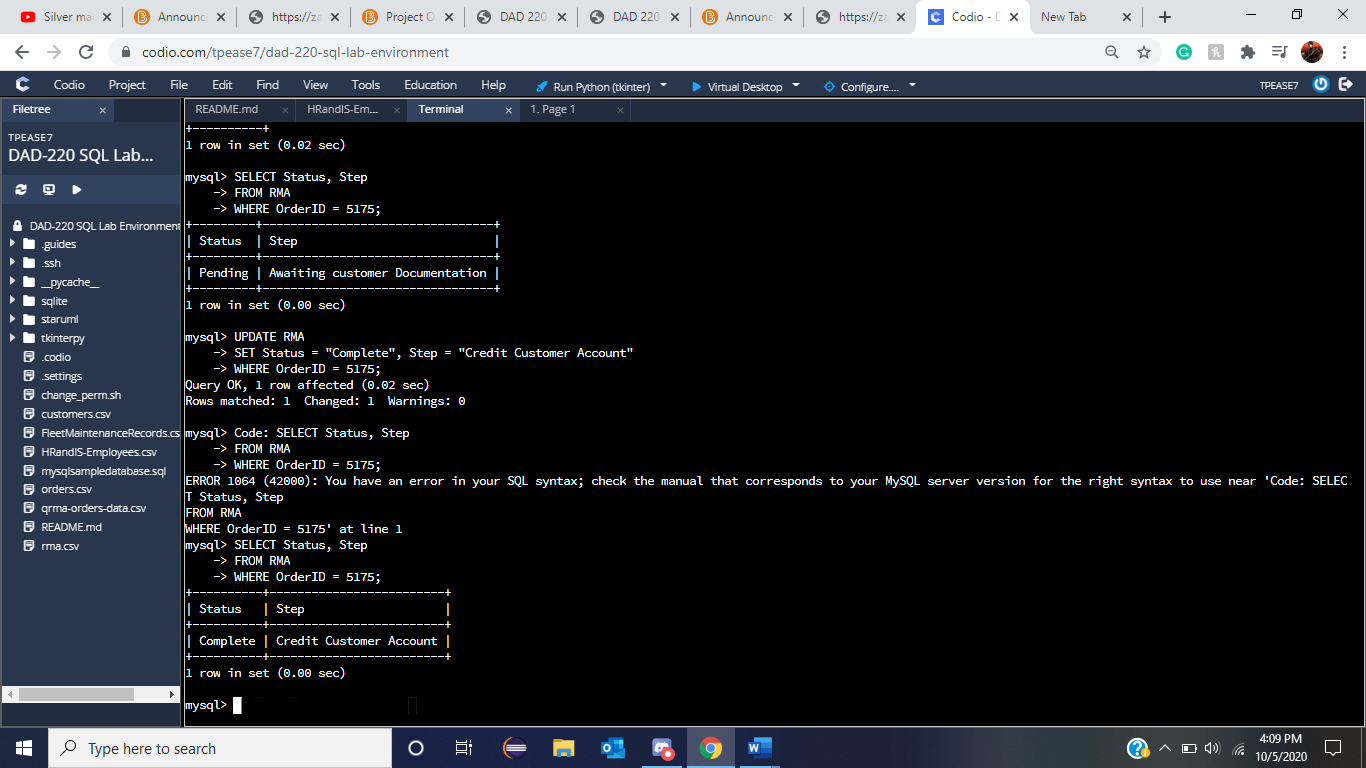
From Customers  
WHERE UPPER(Customers.city) = 'WOONSOCKET';



Following the prior search templates, we instead of having a city have the written request of searching for `Woonsocket’

* + In the rma database, update a customer’s records.
    1. Write an SQL statement to select the current fields of *status* and *step* for the record in the *rma* table with an *orderid* value of “5175.”
       1. What are the current status and step?

Similar to that of module 4 we were able to use the code of Selecting Status and Step From Table RMA where the requirements is that the Order ID is 5175. With only one OrderID it was the one we needed to see if it would work, this is useful when trying to see which orders are there or pulling up information on only one order if needed. Note that Select is not \* but instead just the status and Step from RMA pulling only the relevant info to what is needed.

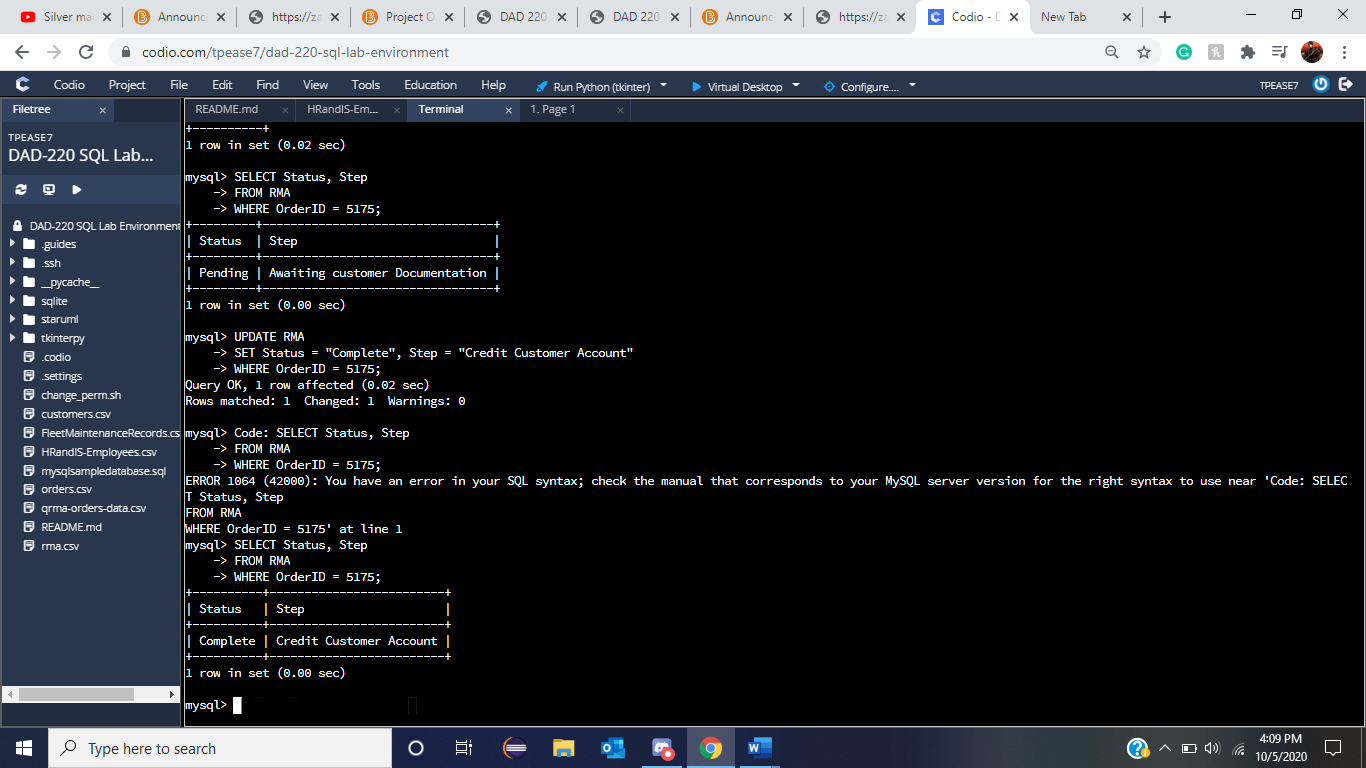


SELECT Status, Step

FROM RMA

WHERE OrderID = 5175;

* + 1. Write an SQL statement to update the *status* and *step* for the *orderid*, 5175 to *status* = “Complete” and *step* = “Credit Customer Account.”
       1. What are the updated *status* and *step* values for this record?



UPDATE RMA

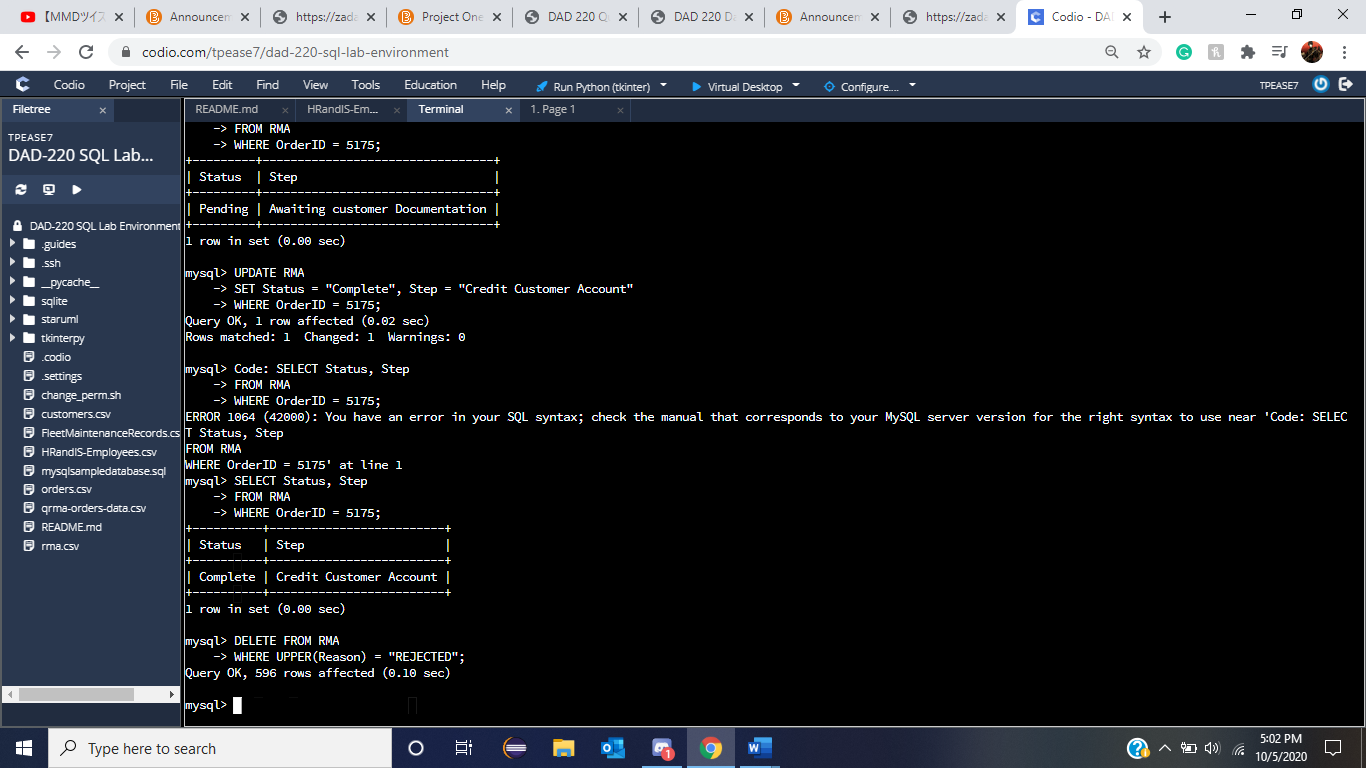
SET Status = “Complete”, Step = “Credit Customer Account”

WHERE OrderID = 5175;

This one is a little different to what we were doing beforehand as this one will have us Update the table similar to such of when we added the foreign key restraints onto each of the tables being created in the database. This Update allows for changes to be done throughout the table instead of solely some. UPDATE RMA sets the RMA ready to be changed, SET Status = “Complete”, Step = “Credit Customer Account” where the OrderID is 5175. Which means that the status will be changed to the new “Complete” Chars and Step to “Credit Customer Account” Chars but only where the ID is 5175. We can see if this occurred with the previous step and as shown the status and step of 5175 has now changed to what we placed into it at this part of the Documentation.

* + Delete rma records.
    1. Write an SQL statement to delete all records with a reason of “Rejected.”
       1. How many records were deleted?

As we are still within the RMA table we can pull a reason for them to be deleted and rejected as why keep information that is no longer useful to the company if it will never be used again. DELETE FROM RMA sets it so something will be deleted from RMA and the requirements will be set to WHERE UPPER(Reason) = REJECTED thus making sure all versions of rejected get confirmed and removed. Because of this we can see that with the query underneath that 596 rows were affected by this statement.



1. **Create an output file of the required query results.** Write an SQL statement to list the contents of the *orders* table and send the output to a file that has a .csv extension.

DELETE FROM RMA

WHERE UPPER(Reason) = “REJECTED”;

**Code:**

SELECT \*

FROM Orders

INTO OUTFILE '/home/codio/workspace/qupd-orders-data.csv'

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n';

There will be times when data will need to be exported after being edited and pulled into the databases and this example similar to module four and five makes it so that a file will be able to be exported out of the documents in order to accurately use them. After this statement was done another csv file was created on the side bar confirming that this file was created in this manner required.

