# DAD 220 Database Documentation Template

## Step One: Create a Database

1. Navigate to your online integrated development environment (IDE). List and record the SQL commands that you used to complete this step here:

A screenshot of a computer

Description automatically generated

Description: To access the integrated development environment, I used the commands from the 3-2 assignment to change permissions so I can edit items once I’m in mysql. I changed the ownership with the following commands:

**chmod +x change\_perm.sh**

**./change\_perm.sh**

**mysql**

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:

A screenshot of a computer

Description automatically generated

A computer screen with a black screen

Description automatically generated

Description: For this query I created the database QuantigrationUpdates. I used the “CREATE DATABASE” statement to create a new database within mysql. I then used the “SHOW DATABASES” statement to list all the existing databases in the server. The command returned a set of databases including QauntigrationUpdates. This verified that the database was created successfully. Next, I used that statement “use QuantigrationUpdates” to enter the database in preparation for the next steps. I used the following commands to complete this step:

**CREATE DATABASE QuantigrationUpdates;**

**SHOW DATABASES;**

**use QuantigrationUpdates;**

1. Using the entity relationship diagram (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:

A screenshot of a computer

Description automatically generated

Description: For this query I used the “CREATE TABLE” statement to create a new table named Customers within the QuantigrationUpdates database. I created the table with the data from the (ERD) reference document. I designated the CustomerID column as the primary key to identify each customer record distinctively. I then used the “DESCRIBE” statement to see that the table I created was made successfully. I used the following commands to complete this step:

**CREATE TABLE Customers (CustomerID INT, FirstName VARCHAR(25), LastName VARCHAR(25), StreetAddress VARCHAR(50), City VARCHAR(50), State VARCHAR(25), ZipCode VARCHAR(10), Telephone VARCHAR(15), PRIMARY KEY(CustomerID));**

**DESCRIBE Customers;**

* 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:

A screenshot of a computer

Description automatically generated

Description: For this query I used the “CREATE TABLE” statement to create a new table named Orders within the QuantigrationUpdates database. I created the table with the data from the (ERD) reference document. I designated the CustomerID column as the foreign key to create a link between the Orders table and the Customers table; this action ensures that the values entered for CustomerID will correspond in both tables. I also set the OrderID as the primary key to identify each order number distinctively. I then used the “DESCRIBE” statement to see that the table I created was made successfully. I used the following commands to complete this step:

**CREATE TABLE Orders (OrderID INT, CustomerID INT, SKU VARCHAR(20), Description VARCHAR(50), PRIMARY KEY(CustomerID));**

**DESCRIBE Orders;**

* 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:

A screen shot of a computer

Description automatically generated

Description: For this query I used the “CREATE TABLE” statement to create a new table named RMA within the QuantigrationUpdates database. I created the table with the data from the (ERD) reference document. I designated the RMAID column as the primary key, which ensures each RMA has a unique identifier. The “NOT NULL” command ensures that the values entered are not null. The foreign key I used links the OrderID column in both the RMA, and Orders tables to ensure values entered must correspond. I then used the “DESCRIBE” statement to see that the table I created was made successfully. I used the following commands to complete this step:

**CREATE TABLE RMA (RMAID INT NOT NULL PRIMARY KEY, OrderID INT, Step VARCHAR(50), Status VARCHAR(15), Reason VARCHAR(15), FOREIGN KEY (OrderID) REFERENCES Orders(OrderID));**

**DESCRIBE RMA;**

## 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.

A screen shot of a computer

Description automatically generated

Description: For this step I used the “LOAD DATA INFILE” statement to import data from the designated csv files into the corresponding tables. The “INTO TABLE” statement specifies where the data will be inserted. The “LINES TERMINATED BY ‘\r\n’” specifies the ending format in the csv file and the “FIELDS TERMINATED BY ‘,’” indicates that the fields in the csv file are separated by commas. I used the following commands to complete these steps:

Customer Table: **LOAD DATA INFILE ‘/home/codio/workspace/customers.csv’ INTO TABLE Customers FIELDS TERMINATED BY ‘,’ LINES TERMINATED BY ‘\r\n’;**

Orders Table: **LOAD DATA INFILE ‘/home/codio/workspace/orders.csv’ INTO TABLE Orders FIELDS TERMINATED BY ‘,’ LINES TERMINATED BY ‘\r\n’;**

RMA Table: **LOAD DATA INFILE ‘/home/codio/workspace/rma.csv’ INTO TABLE RMA FIELDS TERMINATED BY ‘,’ LINES TERMINATED BY ‘\r\n’;**

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 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? **505**

A screen shot of a computer

Description automatically generated

Description: For this query I used the “COUNT(\*)” command to collect the data and return the count of the orders. I combined the Customers and Orders tables using an “INNER JOIN” command on the CustomerID column to ensure only customers with orders are retrieved. I used a “WHERE” command to specify the city and state, and also included the “UPPER()” function to collect the information no matter how it was inputted. The output of the query resulted in **505** orders placed by customers living in Framingham, Massachusetts. I used the following command to complete this step.

**SELECT COUNT(\*) FROM Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID WHERE UPPER(Customers.city) = 'FRAMINGHAM' AND UPPER(Customers.state) = 'MASSACHUSETTS';**

* + 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? **982**

A screen shot of a computer

Description automatically generated

Description: For this query I used the “COUNT(\*)” and “WHERE” command to count the number of customers whose state is specifically Massachusetts. I included the “UPPER()” function to disregard the case it was inputted as. The results showed that there are 982 records in Massachusetts. I used the following command to complete this step:

**SELECT COUNT(\*) 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:

**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 |

A screen shot of a computer

Description automatically generated

Description: For this query I inserted the information from the above table using the “INSERT INTO” command. This added 4 rows to the CustomerID column along with their corresponding information from the table provided. I verified this information was added to the Customers table correctly with a “SELECT \* FROM” and “WHERE” commands to see the information attached to the CustomerID numbers 100004, 100005, 100006, 100007 were implemented successfully. I used the following commands to complete this step:

**INSERT INTO Customers (CustomerID, FirstName, LastNAme, StreetAddress, City, State, ZipCode, Telephone) 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');**

**SELECT \* FROM Customers WHERE CustomerID IN (100004, 100005, 100006, 100007);**

**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 |

A screen shot of a computer

Description automatically generated

Description: For this query I inserted the information from the above table using the “INSERT INTO” command. This added 4 new records to the Orders table under the columns “OrderID”, “CustomerID”, “SKU”, and “Description”. I verified this information was added to the Orders table correctly with a “SELECT \* FROM” and “WHERE” commands to see the information attached to the OrderID numbers 1204305, 1204306, 1204307, 1204308 were implemented successfully. I used the following commands to complete this step:

**INSERT INTO Orders (OrderID, CustomerID, SKU, Description) 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');**

**SELECT \* FROM Orders WHERE OrderID IN (1204305, 1204306, 1204307, 1204308);**

* + 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”? **7**

A screen shot of a computer

Description automatically generated

Description: For this query I used the “SELECT COUNT(\*)” command to count the number of records in the Customer table “WHERE” the city name is Woonsocket. I used the “UPPER()” command to ensure that it pulls the information no matter how it was inputted. The results found that there are 7 customers. I used the following command to complete this step:

**SELECT COUNT(\*) FROM Customers WHERE UPPER(Customers.City) = '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?

**STATUS: Pending STEP: Awaiting customer Documentation**

A screenshot of a computer

Description automatically generated

Description: For this query I used the “SELECT \* FROM” command to pull the information from the RMA table “WHERE” the OrderID is 5175. The results of the query show that the current status of order 5175 is (Pending) and the current step is (Awaiting customer Documentation). These results are based off the order being defective. I used the following command to complete this step:

**SELECT \* 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?

**STATUS: Complete STEP: Credit Customer Account**

A screen shot of a computer

Description automatically generated

Description: For this query I used the “UPDATE” and “SET” commands to change the step and status of order 5175 in the RMA table. I then verified this by using the same command from the previous step to show the information from order 5175. The status has now been updated to (Complete), and the step has been updated to (Credit Customer Account). I used the following commands to complete this step.

**UPDATE RMA SET Status = 'Complete', Step = 'Credit Customer Account' WHERE OrderID = 5175;**

**SELECT \* FROM RMA WHERE OrderID = 5175;**

* + Delete RMA records.
    1. Write an SQL statement to delete all records with a reason of “Rejected.”
       1. How many records were deleted? **596**

A screen shot of a computer

Description automatically generated

Description: For this query I used the “DELETE FROM” command to delete the records within the Reason column that contained REJECTED within the RMA table. I used the “UPPER()” and “LIKE % %” commands to ensure that the records were deleted no matter how they were inputted. The UPPER command accounts for capitalization and the LIKE command combined with the % symbol accounts for any characters before or after the substring REJECTED. The query showed that 596 records were successfully deleted from the RMA table. I used the following command to complete this step:

**DELETE FROM RMA WHERE UPPER(Reason) LIKE '%REJECTED%';**

1. **Update your existing tables** 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:
   1. Rename all instances of “Customer” to “Collaborator.”

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Description: For this query I first used the “ALTER TABLE” command and dropped the foreign key from the orders table using its specific name Orders\_ibfk\_1, using the “DROP” command. I then realized I should change the name of the Customers table to Collaborators. Next, I dropped the primary key from the newly named Collaborators table with the “DROP” command. Then in the Orders table I changed the CustomerID to CollaboratorID and made sure to keep it as an INT input. I then did the same for the Collaborators table changing CustomerID to CollaboratorID and designated the CollaboratorID as the tables primary key. Lastly, I added CollaboratorID as a foreign key in the Orders table to link CollaboratorID within both the Orders table and the Collaborators table. I checked my work by using the “SHOW TABLES” command to see that the Orders table has changed successfully to Collaborators, and I used the “DESCRIBE” command on both the new Collaborators table and the Orders table to verify that all the instances of Customer have been changed to Collaborator. I used the following commands to complete this step:

**ALTER TABLE Orders DROP FOREIGN KEY Orders\_ibfk\_1;**

**RENAME TABLE Customers to Collaborators;**

**ALTER TABLE Collaborators DROP PRIMARY KEY;**

**ALTER TABLE Orders CHANGE CustomerID CollaboratorID INT;**

**ALTER TABLE Collaborators CHANGE CustomerID CollaboratorID INT;**

**ALTER TABLE Collaborators ADD PRIMARY KEY(CollaboratorID);**

**ALTER TABLE Orders ADD FOREIGN KEY(CollaboratorID) REFERENCES Collaborators(CollaboratorID);**

**SHOW TABLES;**

**DESCRIBE Orders;**

**DESCRIBE Collaborators;**

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.

A screen shot of a computer

Description automatically generated

Description: For this query I exported the data of the Orders table to a csv file named QuantigrationUpdatesOrdersData.csv with the “SELECT \* FROM Orders INTO OUTFILE” command. I used the “FIELDS TERMINATED BY ‘,’” and “LINES TERMINATED BY ‘\r\n’” commands to format the data with fields separated by commas and lines terminated by returns and newline characters. The query affected 37998 rows and successfully outputted the data from the Orders table into the newly designated csv file. I used the following command to complete this step:

**SELECT \* FROM Orders INTO OUTFILE '/home/codio/workspace/QuantigrationUpdatesOrdersData.csv' FIELDS TERMINATED BY ',' LINES TERMINATED BY '\r\n';**