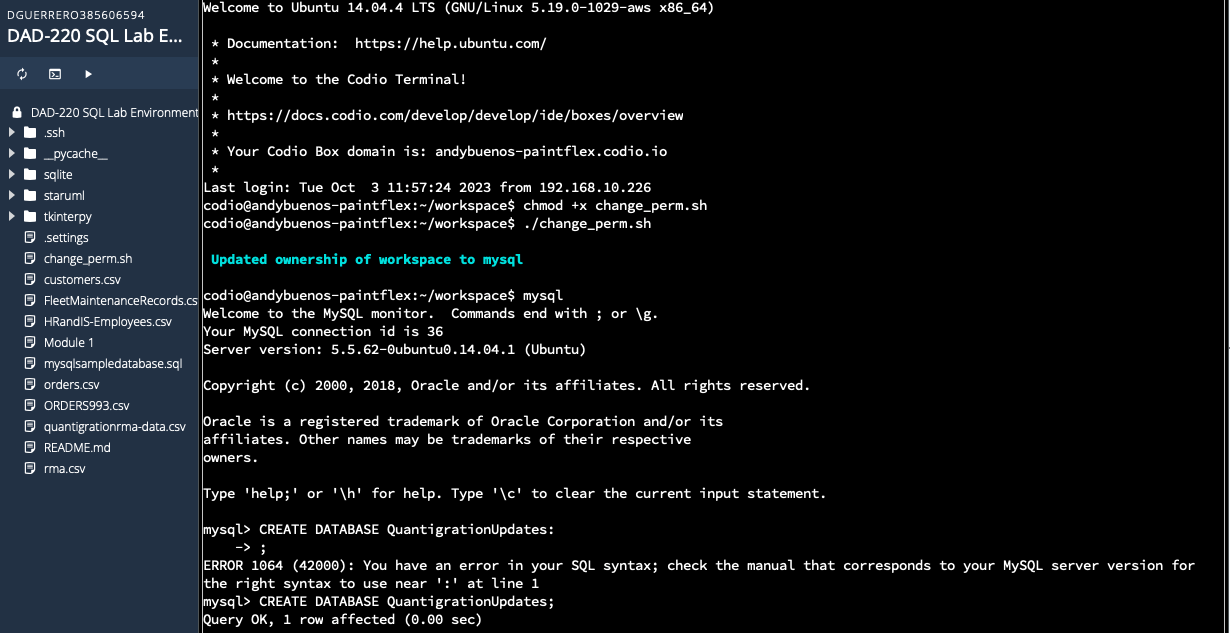
# DAD 220 Database Documentation Template

Complete these steps as you work through the directions for Project One. Replace the bracketed text with your screenshots and brief explanations of the work they capture. Each screenshot and its explanation should be sized to approximately one quarter of the page, with the description written below the screenshot. Follow these rules for each of the prompts and questions below. Review the example document located in the Project One Supporting Materials for assistance.

## 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:



Commands:

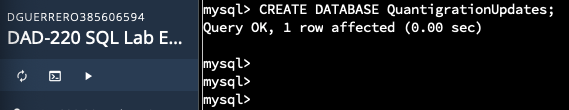
chmod +x change\_perm.sh

./change\_perm.sh

mysql;

Explanation: Commands used to change ownership of sql.

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 program

Description automatically generated

Commands:

CREATE DATABASE QuantigrationUpdates;

show databases;

Explanation:

Using the commands above I created a new database called QuantigrationUpdates with the CREATE command. To show proof that the databases was created I used the SHOW command where it displays all existing databases in the server.

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

Commands:

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;

Explanation:

The provided SQL code creates a table called "Customers" with columns for customer information. It includes a primary key constraint on the "CustomerID" column, ensuring each customer has a unique identifier. The table stores customer details such as first and last name, street address, city, state, ZIP code (as a variable character to accommodate various formats), and telephone number. This schema is designed for organizing and managing customer data in a database. Subsequently I used the “describe” command to show proof of table creation.

* 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 black screen with white text

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Commands:

CREATE TABLE Orders (

OrderID INT NOT NULL PRIMARY KEY,

CustomerID INT,

SKU VARCHAR(20),

Description VARCHAR(50),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

Describe Orders;

Explanation:

The provided SQL code creates a table called "Orders" to manage customer order data. It includes an "OrderID" as the primary key for unique order identification, a "CustomerID" column for associating each order with a specific customer, a "SKU" column for product identifiers, and a "Description" column for order details. The "FOREIGN KEY" constraint on "CustomerID" ensures that the values in this column correspond to valid customer IDs in the "Customers" table, maintaining data integrity and linking orders to their respective customers. Subsequently I used the “Describe” command to show proof of table creation.

* 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 screenshot of a computer

Description automatically generated

Commands:

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

Explanation:

This SQL code creates a table named "RMA" to manage Return Merchandise Authorization (RMA) records. The table includes an "RMAID" column as the primary key for unique RMA identification, an "OrderID" column to associate each RMA with a specific order, a "Step" column to track the current stage or step in the RMA process, a "Status" column to record the status of the RMA (e.g., "In Progress" or "Completed"), and a "Reason" column to provide a brief description of the reason for the RMA. The "FOREIGN KEY" constraint on "OrderID" ensures that the values in this column correspond to valid order IDs in the "Orders" table, establishing a relationship between RMAs and the orders they are linked to while maintaining data integrity. Same as the other tables I utilized the “Describe” command to show proof or table creation.

## 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 black screen with white text

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

Commands:

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';

Explanation:

These SQL statements utilize the **LOAD DATA INFILE** command to import data from CSV files into designated database tables. In the first code snippet, data is imported from the "customers.csv" file located at '/home/codio/workspace/' and is inserted into the "Customers" table. It is assumed that the CSV file employs a comma (',') as the field separator and each line concludes with a carriage return ('\r') followed by a line feed ('\n'). Similarly, the second and third snippets perform analogous operations, loading data from "orders.csv" and "rma.csv" files into the "Orders" and "RMA" tables, respectively. These statements facilitate the efficient transfer of data from external sources into the database, allowing for further analysis and processing.

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 black screen with white text

Description automatically generated

Command:

SELECT COUNT(\*)

FROM Customers

INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID

WHERE UPPER(Customers.City) = 'FRAMINGHAM' AND UPPER(Customers.State) = 'MASSACHUSETTS';

Explanation:

This SQL query counts the number of customers in Framingham, Massachusetts, who have placed orders. It accomplishes this by joining the "Customers" and "Orders" tables based on their shared "CustomerID" column. The query filters customers with 'FRAMINGHAM' as the city and 'MASSACHUSETTS' as the state, regardless of letter case. The result for this query is 505 orders.

* + 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?

928 RECORDS

A screenshot of a computer program

Description automatically generated

Command:

SELECT COUNT(\*)

FROM Customers

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

Explanation:

With this query the customers from the state of MASSACHUSETSS get selected and counted. The result of this query is 982 customers in the state of 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 screenshot of a computer screen

Description automatically generated

Commands:

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);

Explanation:

With the “INSERT” command we are inputting the data into the Customers table following the order the table was created. (CustomerID, FirstName, LastName, StreetAddress, City, State, ZipCode, Telephone). Afterward, I used the “SELECT \* FROM” query to specifically pull the information on the created customers with their unique ID.

**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 screenshot of a computer

Description automatically generated

Commands:

INSERT INTO Orders (OrderID, CustomerID, SKU, Description) VALUES

(1204305, 100004, 'ADV-24-10C', 'Advance Switch 10 GigE Copper 24 port'),

(1204306, 100005, 'ADV-48-10F', 'Advanced Switch 10 GigE Copper/Fiber 44 port copper 4 port fiber'),

(1204307, 100006, 'ENT-24-10', '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);

Explanation:

This SQL code inserts multiple rows of order data into the "Orders" table, with each row containing an order ID, customer ID, product SKU, and order description. With the “SELECT \* FROM” query I was able to verify that the orders where in fact added to the table.

* + 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”?

A screenshot of a computer program

Description automatically generated

Command:

SELECT COUNT(\*)

FROM Customers

WHERE UPPER(Customers.City) = 'WOONSOCKET';

Explanation:

This SQL code counts the number of customers in the "Customers" table whose city is 'WOONSOCKET', regardless of letter case, by applying the **UPPER** function to the "City" column for a case-insensitive comparison. The result of this query is 7 customers.

* + 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?

The status is “Pending” and the step is “Awaiting customer Documentation”.

A black screen with white text

Description automatically generated

Command:

SELECT \* FROM RMA WHERE OrderID = 5175;

Explanation:

With the query above we pulled all the necessary information on the OrderID 5175. This order came back with a status of “Pending” and awaiting customer documentation. The reason is “defective” and it looks like the company is still waiting on documents from the customer to show its defectiveness and issue the refund.

* + 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?

The updated status is “Complete” and the step is “Credit Customer Account”.

A screenshot of a computer screen

Description automatically generated

Commands:

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

SELECT \* FROM RMA WHERE OrderID = 5175;

Explanation:

With the query above we update the step from “Awaiting Customer Documentation” to “Credit Customer Account” and the status from “Pending” to “Complete” on 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 records were deleted.

A black background with white text

Description automatically generated

Command:

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

Explanation:

This query above deletes rows from the RMA table where the Reason column contains the substrings REJ. It effectively removed 596 records meeting this criteria.

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

Commands:

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

RENAME TABLE Customers TO Collaborators;

ALTER TABLE Collaborators DROP PRIMARY KEY;

ALTER TABLE Collaborators CHANGE CustomerID CollaboratorID INT;

ALTER TABLE Collaborators ADD PRIMARY KEY (CollaboratorID);

ALTER TABLE Orders CHANGE CustomerID CollaboratorID INT;

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

Describe Orders;

Describe Collaborators;

Describe Customers;

Explanation:

These SQL commands represent a sequence of actions designed to reshape the database structure. Initially, they involve removing a foreign key constraint called "Orders\_ibfk\_1" from the "Orders" table, effectively separating orders from their associated customers. Following this, the "Customers" table undergoes a name change to "Collaborators" to better reflect its intended function. To accommodate further alterations, the primary key in the "Collaborators" table is removed. Additionally, column names in both the "Collaborators" and "Orders" tables are modified, transforming "CustomerID" into "CollaboratorID" with an integer data type. To solidify these changes, fresh primary and foreign key constraints are introduced, establishing a new connection between the "CollaboratorID" in the "Orders" table and the corresponding field in the "Collaborators" table. In sum, these commands collectively redefine the database's structure and relationships to align with the updated naming conventions and data types.

The “DESCRIBE” commands are used to show that the “CustomerID” title has been change to “CollaboratorID” and it doesn’t exist in the database anymore.

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

Command:

SELECT \*

INTO OUTFILE '/home/codio/workspace/qrma-orders-data85.csv'

FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n'

FROM Orders;

Explanation:

This SQL code is one possible way of exporting the data of the Orders table into a csv file. My csv file is named qrma-orders-data85.csv. The data will be formatted with fields separated by commas and lines terminated by carriage return and newline character. 37998 rows were affected letting us know that was the amount of rows in the Orders table.