Janera Dobson

Southern New Hampshire University

DAD-220-R5643

7-1 Final Project: Database Management Report

Throughout this course we covered structured query language (SQL) and how it can be used to manage database schemas, manipulate data, and analyze data. For this final project, I had to assume the role of a newly hired database engineer for an application startup. The initial release of the message app was a success, and the management team would like me to enhance the database to support a couple of new features in the app. My work is as follows:

**Task 1: Insert Record to the Person Table**Construct the SQL statement to add yourself to the Person table. **Note: You are required to add yourself as a new record in the “person” table. Use your first name and last name for one of the new records that you are inserting.**

I accessed Codio Terminal by computing the statement mysql followed by selecting my database I want to work within which was computed my “use messaging;” In order to insert my name in the person chart my statement was INSERT INTO person (first\_name, last\_name) > VALUES (“Janera”, “Dobson”); I then checked my work with the statement SELECT \* FROM person;A screenshot of a computer screen

Description automatically generated

**Task 2: Alter the Person Table**

Construct the SQL statement to alter the table named “person”. The columns, column data types, and column notes are provided in the previous section. You need to alter the table to include an additional column of your choice. This column should represent some property of a person. You can choose the data type for the column and any constraints on the column.

Considering mysql was already accessed and my database was already selected, all I needed to do was execute the task of altering the person table. I did so by ALTER TABLE person > ADD shoe\_size VARCHAR(2) DEFAULT NULL; To then check my work I computed the statement SELECT \* FROM person;A screenshot of a computer screen

Description automatically generated

**Task 3: Update Records in the Person Table**

Construct the SQL statement to update the existing record in the “person” table to use the new column that you created. Update your record (the record with your first and last name) in the Person table by setting some value to your new property.

In order to now update the records in the person table to use to use the column I created and the record with my first and last name within it; my sql statement was UPDATE person > SET shoe\_size = 7 > WHERE person\_id = 7;A screenshot of a computer screen

Description automatically generated

**Task 4: Delete Records from Person Table**

Construct the SQL statement to delete the record(s) from the “person” table where the first name is “Diana” and the last name is “Taurasi.”

The statement I executed in order to delete the record(s) from person where the first name is “Diana” and the last name is “Taurasi” was DELETE FROM person > WHERE person\_id = 6; The checked my work with SELECT \* FROM person; statement.

A screenshot of a computer screen

Description automatically generated

**Task 5: Alter the Contact List Table**

Construct the SQL statement to alter the table named “contact\_list”. The columns, column data types, and column notes are provided in the previous section. You need to alter the table to include an additional column named “favorite” with a data type of “varchar(10)”. This column is **not required**.

I altered the contact list table with the sql statement of: ALTER TABLE contact\_list > ADD favorite VARCHAR (10) NOT NULL; Then checked my work with SELECT \* FROM contact\_list; sql statement.

A screenshot of a computer screen

Description automatically generated

**Task 6: Update Records in the Contact List Table**

Construct the SQL statement to update the existing records in the “contact\_list” table to use the new column that you created. Update the record(s) in the table by setting Michael Phelps as everyone's favorite contact (contact\_id = 1). The value for the “favorite” column should be set to “y” for these records.

With this task I was sort of skeptical if I did it right. My sql statement was UPDATE contact\_list > SET favorite = “y” > WHERE contact\_id > 1; Then checked my work by this sql statement: SELECT \* FROM contact\_list;A screenshot of a computer screen

Description automatically generated

**Task 7: Update Records in the Contact List Table**

Construct the SQL statement to update the existing records in the “contact\_list” table to use the new column that you created. Update the remaining record(s) in the table by setting every contact who is NOT Michael Phelps (contact\_id <> 1) to not be a favorite. The value for the “favorite” column should be set to “n” for these records.

Here’s to why I really felt confused with task 6 considering now I’m overriding what I already just constructed, but I still did so how I was asked (so I hope). UPDATE contact\_list > SET favorite = “n” > WHERE contact\_id <> 1; I then followed up with SELECT \* FROM contact\_list to check my work.A screenshot of a computer screen

Description automatically generated

**Task 8: Insert Records to Contact List Table**

Construct the SQL statement to insert at least 3 new records in the “contact\_list” table. Make sure that you use the new column that you created in the previous step. **Note: You are required to add at least 3 new records with yourself as a new contact in the “contact\_list” table. Make sure that you provide a value (y or n) for the new “favorite” column.**

For some reason with this task I had to insert records into the contact list one at a time. I tried putting all the values together and now looking over my work I see that I was missing commas at the end of each value statement. An example of one of my statements was INSERT INTO contact\_list(connection\_id, person\_id, contact\_id, favorite) > VALUES (“15”, “7”, “7”, “y”); To check my work I computed SELECT \* FROM contact\_list;A screenshot of a computer screen

Description automatically generated

**Task 9: Create the Image Table**

Construct the SQL statement to create a table named “image”. The columns, column data types, and column notes are provided here. Create the “image” table according to these specifications.

My sql statement to create a table name image was: CREATE TABLE image( > image\_id INT(8) NOT NULL AUTO\_INCREMENT, > image\_name VARCHAR(50) NOT NULL, > image\_location VARCHAR(250) NOT NULL, > PRIMARY KEY(image\_id) > ) AUTO\_INCREMENT = 1;A screenshot of a computer

Description automatically generated

**Task 10: Create the Message-Image Intersection Table**

Construct the SQL statement to create an intersection table named “message\_image”. The columns, column data types, and column notes are provided here. Create the “message\_image” table according to these specifications.

My sql statement in order to create the message image intersection table was: CREATE TABLE message\_image ( > message\_id INT(8) NOT NULL, > image\_id INT(8) NOT NULL, PRIMARY KEY (message\_id) > ); Now because this intersected chart required both the message\_image and message\_id to be PRIMARY KEY, I had to perform a few extra codes in order to make that happen. The first was to drop the original primary key by ALTER TABLE message\_image > DROP primary key; and then compute another ALTER TABLE statement which was ALTER TABLE message\_id > ADD PRIMARY KEY(message\_id, image\_id);A screenshot of a computer

Description automatically generated

**Task 11: Insert Records to Image Table**

Construct the SQL statement to insert 5 new records in the “image” table.

In order to insert 5 new records in the image table I computed: INSERT INTO image (image\_name, image\_location) > VALUES > (“cat”, “window”), (“dog”, “bed”), (“balloon”, “store”), (“car”, “garage”), (“computer”, “library”); Then checked my code with the SELECT \* FROM image; sql statement. A screenshot of a computer

Description automatically generated

**Task 12: Insert Records to Message-Image Table**

Construct the SQL statement to insert 5 new records in the “message\_image” intersection table. **Note: You are required to add at least one record where at least one of Michael Phelp's messages includes at least one image. Also, you are required to add at least one message that has multiple images.**

The sql statement I constructed to insert 5 new records in the message\_image intersection table was: INSERT INTO message\_image(message\_id, image\_id) > VALUES > (“1”, “1”), (“3”, “5”), (“5”, “3”), (“2”, “2”), (“4”, “4,5”); Checked my work with SELECT \* FROM message\_image;A screenshot of a computer

Description automatically generated

**Task 13: Find All of the Messages that Michael Phelps Sent**Construct the SQL statement to find all of the messages that Michael Phelps sent. **Note: You must use the WHERE clause to set the conditions for this query.** Display the following columns:

- Sender's first name  
- Sender's last name  
- Receiver's first name - Receiver's last name - Message ID  
- Message  
- Message Timestamp

My sql statement for finding all the messages that Micheal Phelps sent was: SELECT > pSender.first\_name as ‘Sender\’s first name’, pSender.last\_name as ‘Sender\’s last name’, pReceiver.first\_name as ‘Receiver\’s first name’, pReceiver.last\_name as ‘Receiver\’s last name’, message.message\_id as ‘Message ID’, message.send\_datetime as ‘Message Timestamp’, > from message > join person pSender on pSender.person\_id = message.sender\_id > join pReceiver on pReceiver.person\_id = message.receiver\_id > WHERE sender\_id = 1; Note, I am just now realizing while putting together my report that throughout my code I continuously spelled receiver wrong. Lesson learned to take my time looking for grammatical errors next time.

A screenshot of a computer screen

Description automatically generated

**Task 14: Find the Number of Messages Sent for Every Person**Construct the SQL statement to find the number of messages sent for every person. **Note: You must use the WHERE clause to set the conditions for this query.** Display the following columns:

- Count of messages - Person ID  
- First Name  
- Last Name

The sql statement I constructed to find the number of messages sent for every person was: SELECT > Count(message) as “Count of Message” > , person.person\_id as “Person ID” > , person.first\_name as “First Name” > , person.last\_name as “Last Name” > from message > join person > ON message.message\_id = person.person\_id > GROUP BY person.person\_id, person.first\_name, person.last\_name; A screenshot of a computer screen

Description automatically generated

**Task 15: Find All of the Messages that Have At Least One Image Attached Using INNER JOINs**

Construct the SQL statement to find all of the messages that have at least one image attached using INNER JOINs. **Note: For messages with multiple images, display only the first image for the message.** Display the following columns:

- Message ID  
- Message  
- Message Timestamp - First Image Name  
- First Image Location

The last and final task required me to find all the messages that have at least one image attached using INNER JOINs. My statement was as follows: SELECT > message.message\_id as “Message ID” > , message.message as “Message” > , message.send\_datetime as “Message Timestamp” > , image\_name as “First Image Name” > , image\_location as “First Image Location” > from message > INNER JOIN image ON message\_id = image\_id;A screenshot of a computer screen

Description automatically generated