Darin Maxey

DAD-220

Introduction to SQL

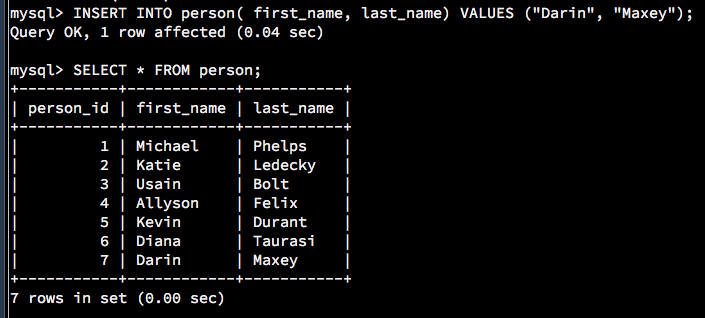
**Final Project**

**Task 1: Insert Record to the Person Table**

To insert data I used all the fields except the PRIMARY KEY field to insert test data. For example, I did not include the person\_id field when I inserted values to the person table. To see what the PRIMARY KEY column was I used “SHOW COLUMNS FROM person”. To insert my name into the person table I used “INSERT INTO” which my name then got assigned to person\_id 7 because person\_id is an auto incremented column.

Code:

INSERT INTO person( first\_name, last\_name) VALUES ("Darin", "Maxey");



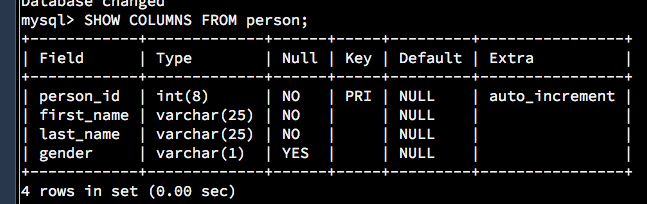
**Task 2: Alter the Person Table**

To alter the customer table I used ALTER TABLE and since I am altering the customer table I used “ALTER TABLE person”. The column I added was a gender column that only allows 1 character by using “VARCHAR(1)”. If this field is not populated it will default to NULL. I was able to confirm this using “SHOW COLUMNS FROM person” (the table name).

Code:

ALTER TABLE person ADD gender VARCHAR(1);



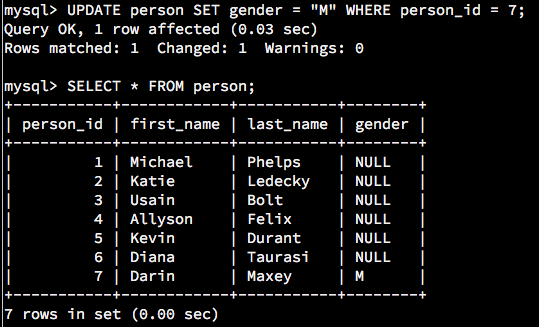


**Task 3: Update Records in the Person Table**

To update the record with my name I used UPDATE customer (table name) SET gender (column I created) = ‘M’ for male WHERE person\_id = 7. I knew my person\_id was 7 because I used “SELECT \* FROM person” before updating the record. I then used “SELECT \* FROM person” to verify my record updated.

Code:

UPDATE person SET gender = "M" WHERE person\_id = 7;

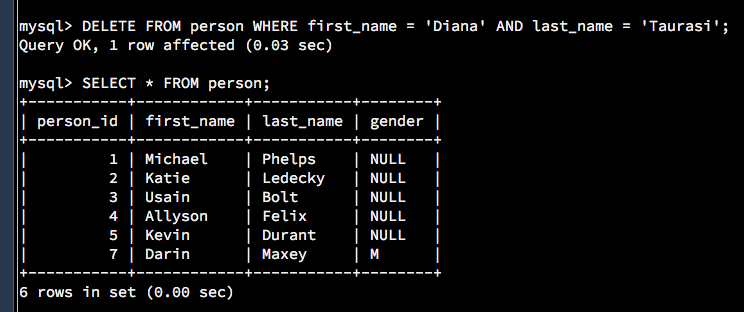


**Task 4: Delete Records from Person Table**

To delete “Diana Taurasi” from the person table I used “DELETE FROM person.” I also used WHERE to specify only delete the records with first name “Diana” and last name “Taurasi.” I also could have used “DELETE FROM person WHERE person\_id = 6” since Diana Taurasi was assigned person\_id = 6.

Code:

DELETE FROM person WHERE first\_name = 'Diana' AND last\_name = 'Taurasi';

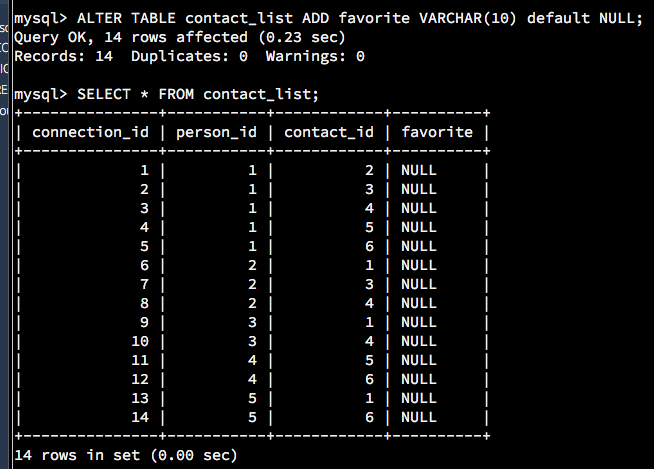


**Task 5: Alter the Contact List Table**

To add the “favorite” column I used “ALTER TABLE contact\_list” since contact\_list is the table we are trying to update. Then next part is “ADD favorite” to add a column named favorite. The “VARCHAR(10)” part limits the favorite column to 10 characters. I then set the column to “default NULL” since it is not a required field.

Code:

ALTER TABLE contact\_list ADD favorite VARCHAR(10) default NULL;

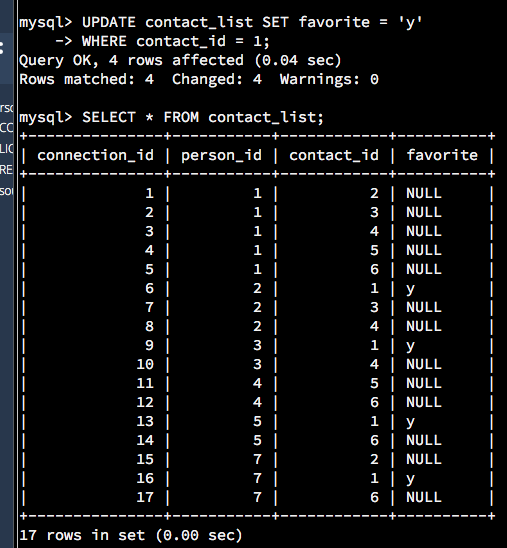


**Task 6: Update Records in the Contact List Table**

To update the contact list table I used “UPDATE contact\_list” because contact\_list is the table name and set favorite = ‘y’ “WHERE” contact\_id = 1 because that correlates to Micheal Phelps.

Code:

UPDATE contact\_list SET favorite = 'y' WHERE contact\_id = 1;

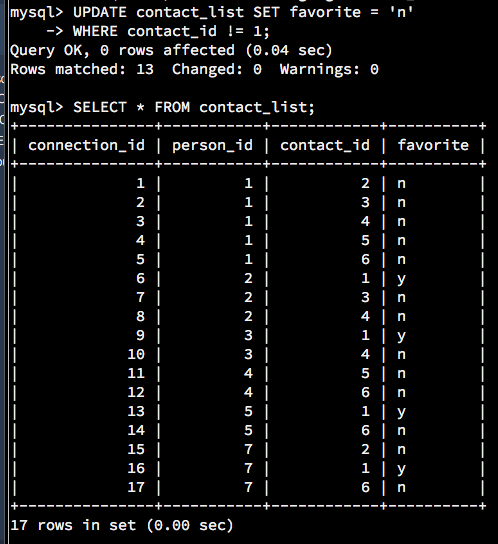


**Task 7: Update Records in the Contact List Table**

To update the favorite column to ‘n’ for any contact not equal to Micheal Phelps I used SET favorite = ‘n’ WHERE contact\_id does not equal to 1 by using “!= 1.”

Code:

UPDATE contact\_list SET favorite = 'n' WHERE contact\_id != 1;



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

I was able to insert rows into the contact\_list table I used “INSERT INTO.” I excluded the connection\_id column because it is the primary key. I updated the values in the same order that the columns are listed. I also could have used “INSERT INTO contact\_list (person\_id, contact\_id, favorite) VALUES (7,2, ‘y’), (7,1, ‘n’), (7,6,’y’);.”

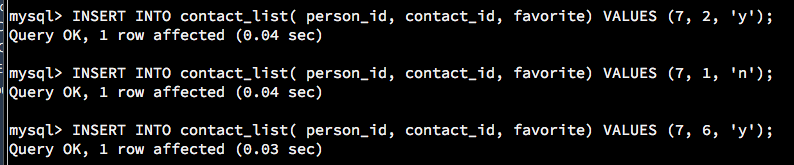
Note: I had to revise task 6 and 7 because I messed up the favorite column. I revised it but after I completed task 8. The favorite values ended up being (7, 2, ‘n’), (7,1,’y’), (7,6,’n’). See “SELECT \* FROM contact\_list;” from task 7.

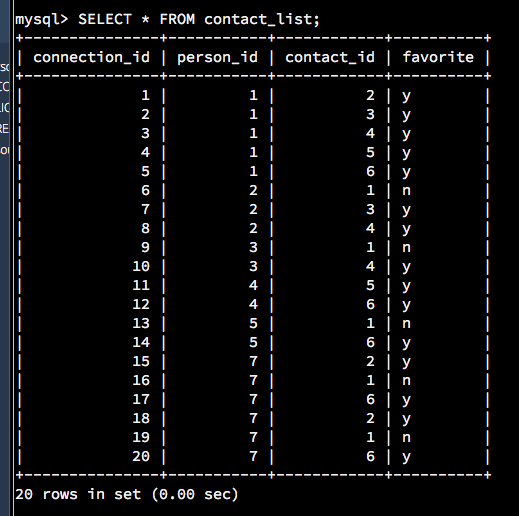
Code:

INSERT INTO contact\_list( person\_id, contact\_id, favorite) VALUES (7, 2, 'y');

INSERT INTO contact\_list( person\_id, contact\_id, favorite) VALUES (7, 1, 'n');

INSERT INTO contact\_list( person\_id, contact\_id, favorite) VALUES (7, 6, 'y');



**Task 9: Create the Image Table**

To create the image table I set up the image\_id as an INT column and set up the image\_name and image\_location columns as VARCHAR. Image\_id is set up to allow up to an eight digit number, image\_name is set up to hold up to 50 characters, and image\_location is set up to hold up to 250 characters. I then set the image\_id column as the primary key at the and to auto\_increment by 1.

Code:

CREATE TABLE image(

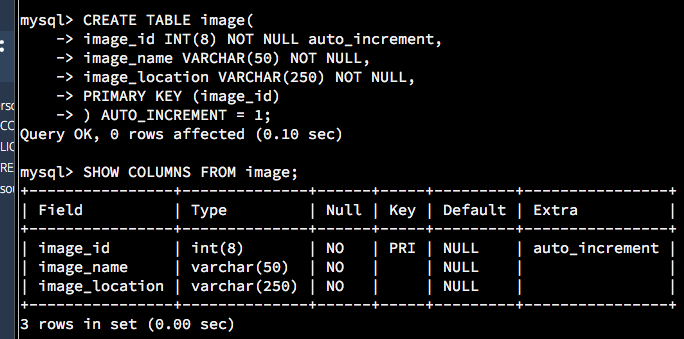
-> image\_id INT(8) NOT NULL auto\_increment,

-> image\_name VARCHAR(50) NOT NULL,

-> image\_location VARCHAR(250) NOT NULL,

-> PRIMARY KEY (image\_id)

-> ) AUTO\_INCREMENT = 1;



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

I used similar code as task 9, but for task 10 I created two primary keys message\_id and image\_id. I made them required fields by adding “NOT NULL”.

Code:

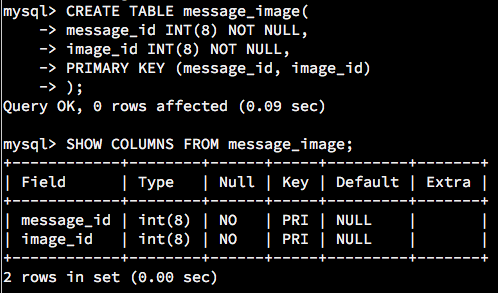
CREATE TABLE message\_image(

-> message\_id INT(8) NOT NULL,

-> image\_id INT(8) NOT NULL,

-> PRIMARY KEY (message\_id, image\_id)

-> );



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

For task 11 I used similar logic as task 8, but instead of inserting records one row at a time, I combinged them by using VALUES (name1, location1), (name2, location2),…etc. I used “SELECT \* FROM image” to make sure the table is how I wanted to set it up.

Code:

INSERT INTO image (image\_name, image\_location)

VALUES('800m freestyle', 'swimming photos'), ('gold medals', 'olympic photos'), ('greatest of all time', 'olympics photos'), ('sprint', 'swimming photos'), ('motivational picture', 'motivation photos');



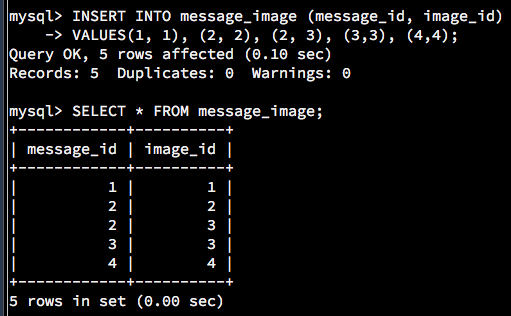
**Task 12: Insert Records to Message-Image Table**

I used the same logic as task 11 for task 12, but I made sure to add the message\_id has two images by assigning it two image\_id’s.

Code:

INSERT INTO message\_image (message\_id, image\_id)

VALUES(1, 1), (2, 2), (2, 3), (3,3), (4,4);



**Task 13: Find All of the Messages that Michael Phelps Sent**

Task 13 took me the longest to figure out. What I ended up having to do for this one is create two temporary tables (sender and receiver). I first created sender then joined it to the message table by the person\_id and sender\_id columns. I then joined that with receiver by person\_id from the reciver table and receiver\_id from the message table. I then filtered it down to WHERE sender person\_id = 1 to filter the data down to only include messages sent by Micheal Phelps. The beginning of the code is selecting all the columns I want to bring back in the data.

Code:

SELECT sender.first\_name AS sender\_first\_name, sender.last\_name AS sender\_last\_name,

receiver.first\_name AS receiver\_first\_name, receiver.last\_name AS receiver\_last\_name,

message.message\_id, message.message, message.send\_datetime

FROM message

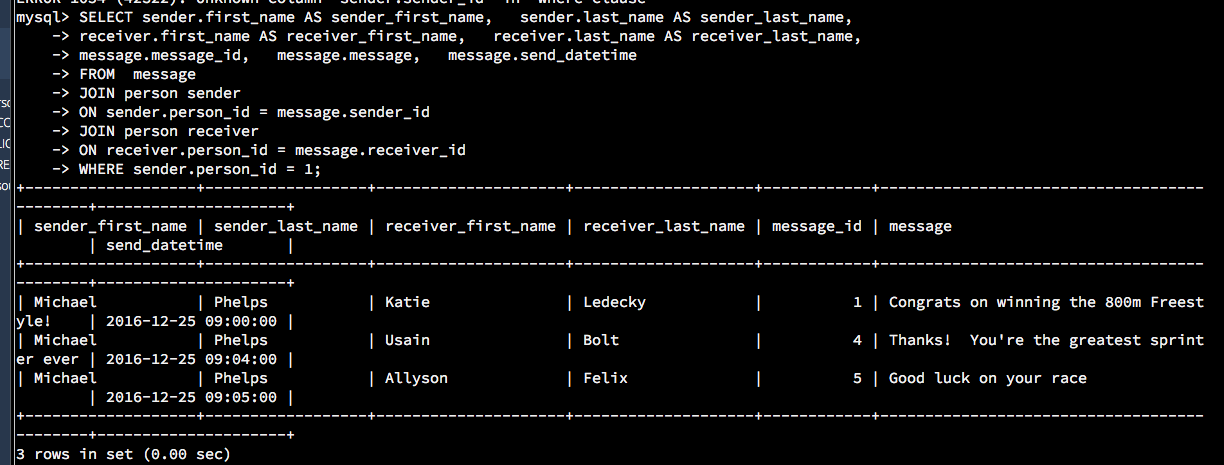
JOIN person sender

ON sender.person\_id = message.sender\_id

JOIN person receiver

ON receiver.person\_id = message.receiver\_id

WHERE sender.person\_id = 1;



**Task 14: Find the Number of Messages Sent for Every Person**

To find the number of messages sent for every person I used COUNT, JOIN, and GROUP BY. I joined the message and person tables by sender\_id and person\_id. I also found the number of messages by using COUNT of the messag\_id’s. Then I grouped the data by the person\_id to return the count of messages by person.

Code:

SELECT COUNT(message.message\_id) AS count\_of\_messages, person.person\_id,

person.first\_name AS sender\_first\_name,

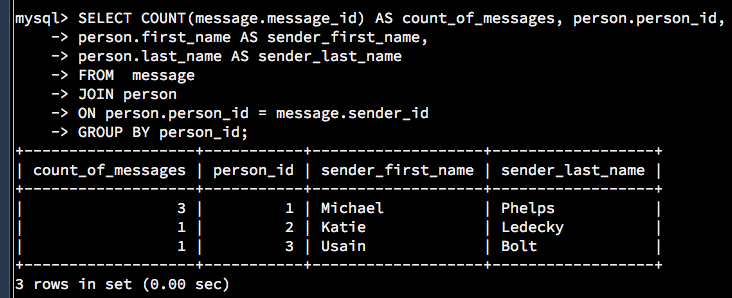
person.last\_name AS sender\_last\_name

FROM message

JOIN person

ON person.person\_id = message.sender\_id

GROUP BY person\_id;



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

To find all the messages that have at least one image I used INNER JOIN to join the message table and the message\_image table by the message\_id columns. The INNER JOIN combines the two tables using columns that appear in both tables. Once I joined the data I used GROUP BY to group by message\_id to exclude the messages that have more than one image.

Code:

SELECT message.message\_id, message.message, message.send\_datetime,

image.image\_name, image.image\_location

FROM message

INNER JOIN message\_image

ON message.message\_id = message\_image.message\_id

INNER JOIN image

ON message\_image.image\_id = image.image\_id

GROUP BY message\_id;

