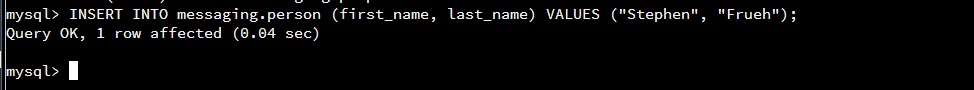
Stephen Frueh

DAD 220

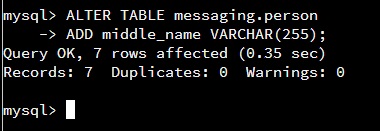
Final Project Screenshots, Results, and Captions

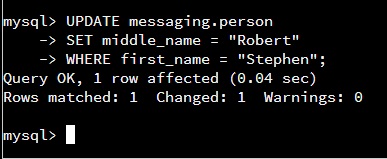
4/9/2018

When beginning the final project, I left out the lines which initialize MySQL and USE the messaging database. After that, the command prompt is ready to start the tasks.

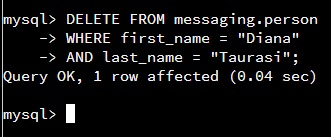


The first task was to insert a record into the PERSON database using our own name. This was fairly straightforward using a simple INSERT INTO command and providing the values that would insert me into the table.

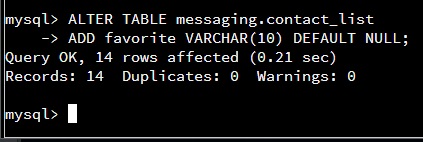


The second task was to alter the PERSON table to add a custom field of our own choosing. Going with the theme of the table as it is already, I chose to add a column for the person’s middle name, which can be a string 255 characters long.

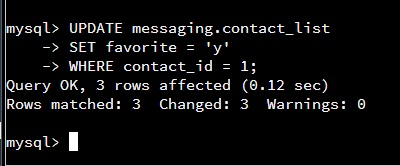
The third task was to update the record we inserted into the PERSON table to add the value in the new column that was added in task 2. Since the column I added was the person’s middle name, this just required using the UPDATE command to set the middle\_name column anywhere the first and last name of the record was equal to my name.



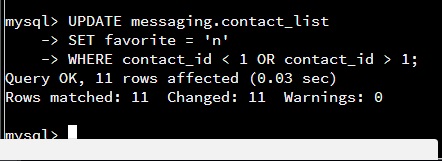
The fourth task was then to delete a specific record out of the PEOPLE table. This is easily done by using the DELETE FROM command, and in a WHERE clause specifying the first and last name of the person record we want to delete.



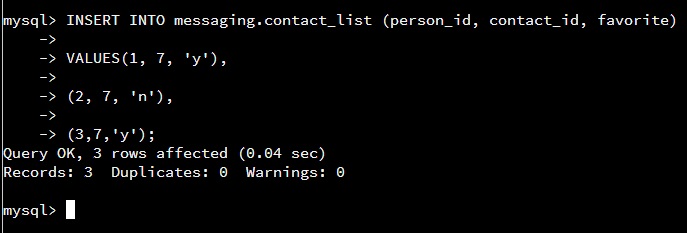
Next I was asked to alter the contact\_list table to add a specific column to the table. This column could be NULL and can be a string of 10 characters. Similar to the previous ALTER TABLE command with specific specifications for the new column.



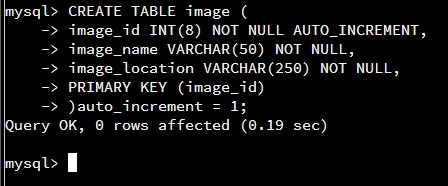
Now I was asked to update the new column I just added for any contact\_id that was equal to Michael Phelps’ contact, setting the new column to ‘y’. Since there were multiple rows that might have to be updated, and which ones had to be specific, I just used a where clause checking the contact\_id if it was equal to Michael Phelps. If so, change the FAVORITE column to a ‘y’. Otherwise leave it alone.

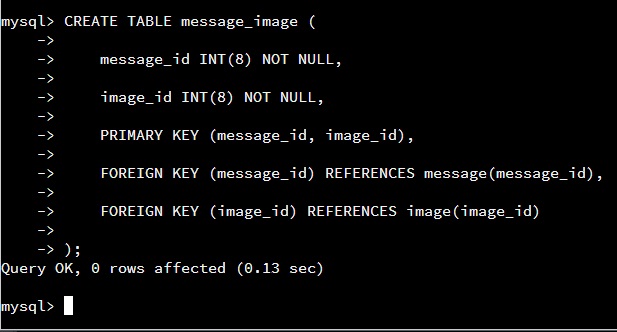


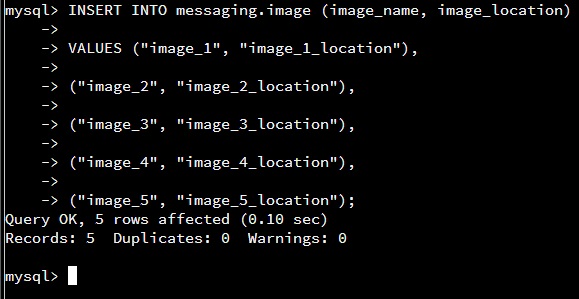
Similar to the previous tasks, the next task asked me to go through each record in the contact\_list table again, and change any rows’ favorite column that IS NOT Michael Phelps to a ‘n’. Almost exactly the same as the previous task, but instead of checking if the contact\_id = 1, we check for a contact\_id that IS NOT equal to 1. If it isn’t 1, change the FAVORITE column to a ‘n’.



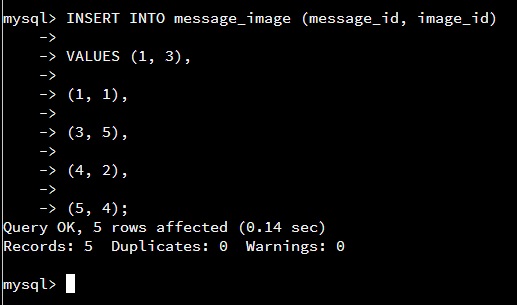
Next I was asked to insert three rows of data, or records, into the contact\_list table with ourselves as the contact\_id field. Since I was the last contact added to the PEOPLE table, my ID was 7. Above I’m setting the values for the three new records with values for people\_IDs 1, 2, and 3 with me as the contact and whether or not I’m a favorite for that person.



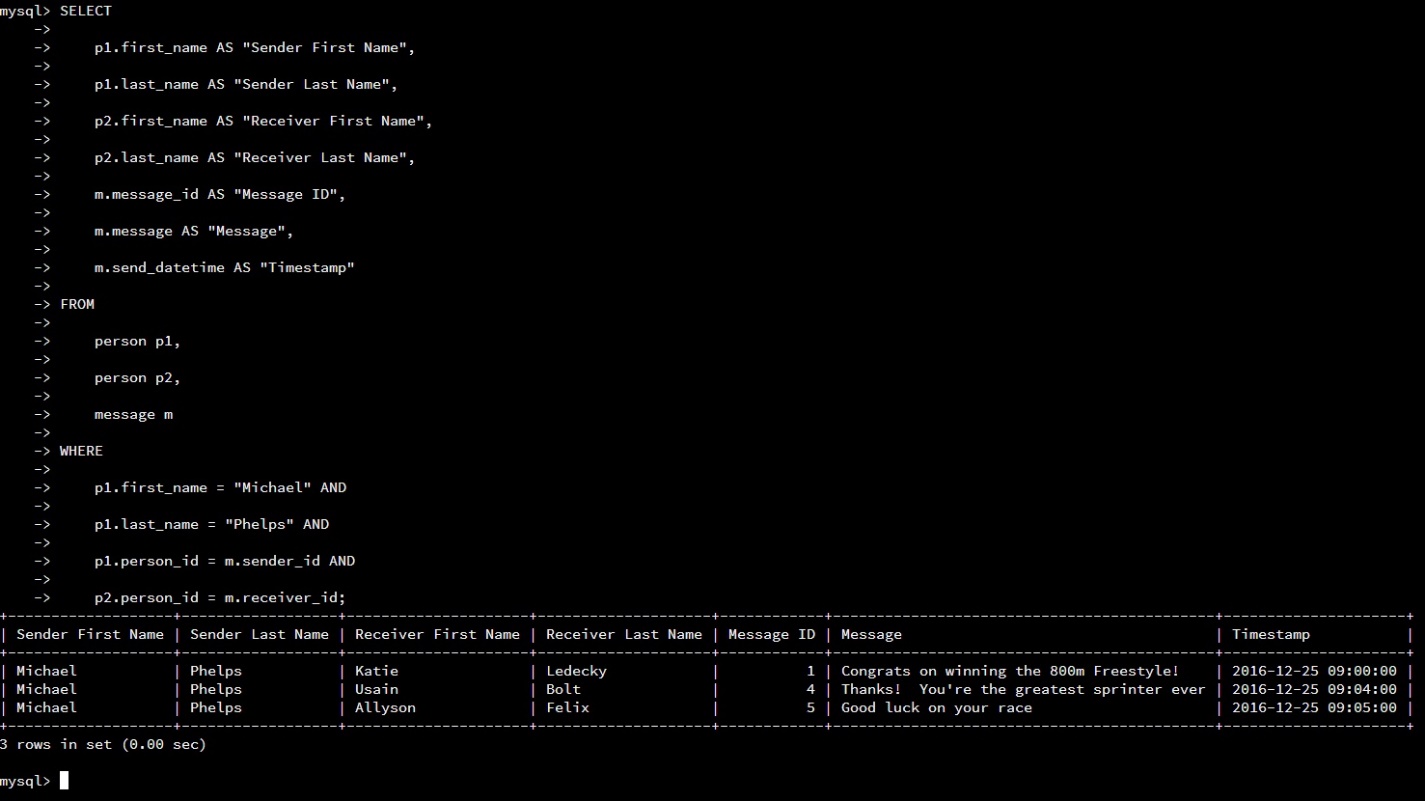
The next task was to create a new table called IMAGE that is used to store data related to images sent in messages. I used the CREATE TABLE command, with arguments detailing each column in the table and setting the primary key of the table to be the IMAGE\_ID field.

Next, I created another new table which was an intersection table between the messages and the images contained in them. This table is only made up of two FOREIGN KEYS, one from the message table and one from the image table, with both of them being primary keys for this new table. When an image is sent in a message, the message ID and the image ID are both stored in this table, keeping the messages and the images together.

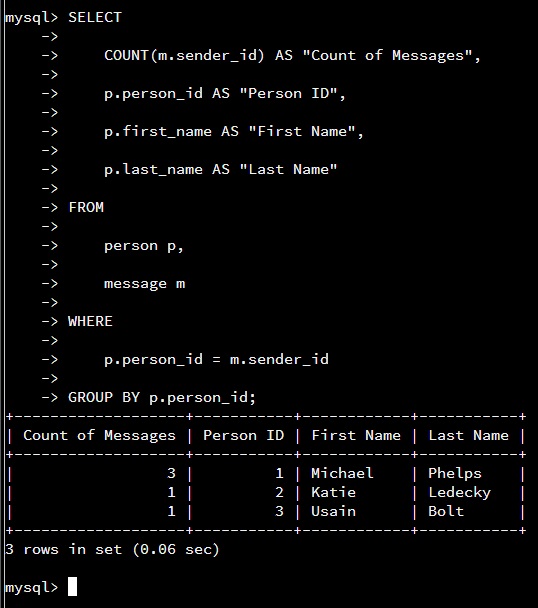
After creating the message\_image table, next I was required to populate the IMAGE table created a few tasks back with some records. Here the values are generically named along with the locations of the images.



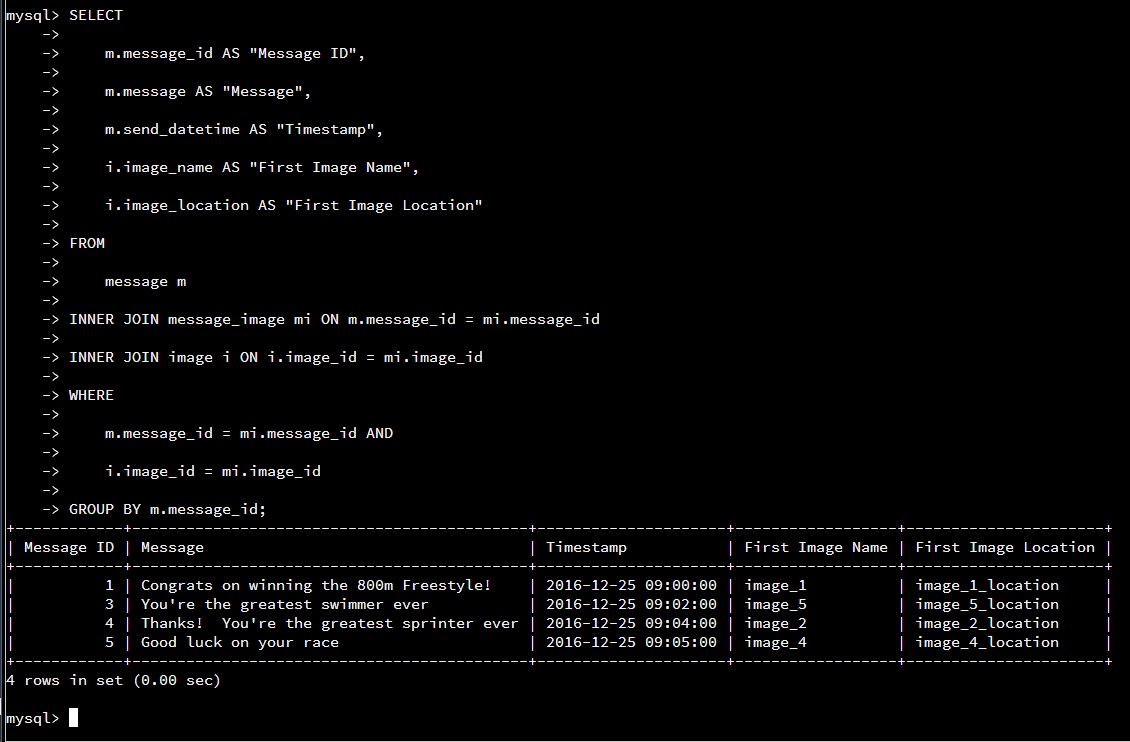
Now that both, the MESSAGE and IMAGE tables have records they are holding, we can add them to the MESSAGE\_IMAGE table and link the images to the messages they were sent in. This is simply done by adding records to the MESSAGE\_IMAGE table with the ID of the message which included the image, and the ID of the image itself. The first two records are the one message from Michael Phelps which included two separate images. Message\_ID 1 has both, Image\_ID 1 and Image\_ID 3 included in the messages, and both are linked to the message ID.



The next task was to select all messages sent from Michael Phelps to another contact. This is simply selecting all the messages, and then only the ones where Michael Phelps’ person\_ID is the same as the messages’ SENDER\_ID, since it’s the ID of the person who sent each message. The first and last name of the sender of the message has to equal “Michael Phelps”, and his ID must be equal to the messages’ SENDER\_ID for it to be selected.



The next task was to count the total number of messages sent by each person who sent messages. This was done using the COUNT function, and counting each number of messages with the same sender\_ID. At the end of the command, I group the results by the person’s ID, or the results would display the count of all the messages for just the first person, which would be incorrect. The GROUP BY separates the rows of the table displayed by the argument used with the command, in this case, each person’s ID.



The last task was to display only the first image each person has sent in a message. In the case of Michael Phelps, he has sent two images in one message, but we only want to include the first one. To not show any images after the first one, I used the GROUP BY command with the message\_id argument, to only include one row for each message. The rest of the command is handling linking each user to their messages with images included using two INNER JOIN statements. The first one joins the message ID with the message\_image table’s foreign key message\_id, and the second INNER JOIN does the same thing with the Image ID. Since the MESSAGE\_IMAGE table links the messages and their images, the IDs are compared to check that each image is linked to it’s correct message, and the messages’ data displayed such as the sender and image.