### Questions on Part 1: Turn in only this final sheet with all answers and SQL code and results for question 11. Do not turn in the previous pages.

1. Explain the difference between an explicit and implicit transaction?

Implicit transactions allow a user to manipulate their database using DML with the convenience of an undo button. In implicit transactions our commands are stored temporarily until we either a commit or rollback command is issued, during this time our database will be locked to other changes. Finally implicit transactions imply the start and end of a transaction. Explicit transactions are defined by a begin transaction statement followed by our desired DML commands and finally a commit or rollback statement. An explicit transaction allows for multiple DML commands to be executed in a single batch.

1. What is the default transactional behavior of SQL server? IMPLICIT TRANSACTION OR EXPLICIT TRANSATION how does one override that behavior?

If we open a new query and enter select @@OPTIONS & 2 we can see that the default mode is 0 or Explicit Transaction mode. To override this behavior we must use the command set implicit\_transactions on this allows us to enable or disable implicit transaction mode (2).

1. How do you start an implicit transaction in SQL?

First you must issue the command set implicit\_transactions on to override the default explicit transaction mode. Then you may enter your desired DML commands. Finally you can issue either the commit or rollback command to modify the database permanently.

1. How do you start an explicit transaction in SQL?

Because our default mode is explicit we can begin by entering the command begin transaction. Following the begin transaction statement we can issue multiple DML commands to modify our database to our liking. Finally we must either commit or rollback our changes.

1. Describe a scenario for which one would use IMPLICIT TRANSACTIONS)?

Implicit transactions are useful when tinkering with a non production database where you are the sole user. The ability to quickly rollback changes that have already been executed is useful when we want to experiment with new or unfamiliar code.

1. Describe a scenario for explicit transactions in the real-world.

As we learned in our lab explicit transactions are how we do things in the professional world. An example of this would be a production server with multiple users where changes either need to be committed or rolled back all at once. We can use some internal logic to add safeguards to our code, these safeguards can automatically roll back transactions if just a single portion is unsuccessful.

1. If you have a stored procedure and the body of that stored procedure executes more than one update statement, should you use a transaction? Why or why not?

It would be very sensible to use a transaction (preferably explicit) in the body of this procedure. The purpose of this transaction would be to rollback all the updates if any one of the update statements throws an error; otherwise we would be left with a half-baked database with some updates successful and others not. This leaves us with a bigger problem than if we just had to go back and fix the procedure and execute it again.

1. As demonstrated from the two colors example, a key benefit of transactions is that they do not allow for intermediate states (for example if one insert works, but the other fails, then they both fail). Explain the importance of not allowing intermediate states.

The importance of not allowing intermediate states can be explained by a necessity for databases to be consistent and follow rules. Intermediate states could commit partial changes to a database making it difficult to fix mistakes once code is executed. We could lose track of the changes made to our database if we had to track each piece of code that ran and each that didn’t.

1. If you try to use the two colors procedure you created in the lab to add the colors blue and purple what will happen and why? (What gets inserted, does the transaction succeed?)

We get a primary key violation and our procedure is aborted. This is because one of our values (blue) is already in our colors table. Consequentially purple and blue are both prevented from being added to the table.

1. If you try to use the two colors procedure you created in the lab to add the colors pink and teal what will happen and why? (What gets inserted, does the transaction succeed?)

The transaction succeeds in adding both teal and pink to the database table colors. The transaction is successful because neither value is already found in our table. Therefor we commit the colors to our table.

1. Create and execute the SQL that shows how many colors are in the Colors table. Show SQL and Result set.

select COUNT(color\_name) as color\_count from colors

