**Overview of System**

This system is quite straightforward. I want to focus on the scheduling system. In some previous deliverable talks, I mentioned how I had to rework the original plan when it came to scheduling. I gave a brief overview of the change to the original plan, but here I want to give it a longer overview of the system. The system starts with one large overarching procedure, which inserts data into the month, week, day, and jobs tables. This is what is used when you are entering either data for a new volunteer’s work history, or data for a new month for a volunteer. The second one enters data into the week, day, and job tables, this is used when a new week is being entered in a particular month. The third one enters data in the day and job tables. This one is used if someone volunteers for more than one day a week. This fulfills the requirement that a volunteer is allowed to work more than once a week, but not be required to work every week.

I thought about adding a year table, but I decided against it for now, in the future if we need to start tracking things more closely adding that would be possible. For what is required at the moment, this is sufficient. One thought that might be a note for the future, you might be able to condense some stuff down in this schema. At the moment, I like how it looks, but if you want to take some complexity away, you could relegate the job table to a look-up table for the day table. However, if for some reason someone worked in multiple jobs in one day, this current system would allow for multiple inputs for one day in the job table. I also have a stored procedure that calls up the complete work history of a particular volunteer. You simply enter their volunteerpersonID and it gives you a list of every month, week, day, and the job they did that particular day.

**Testing and Limitations/Bugs**

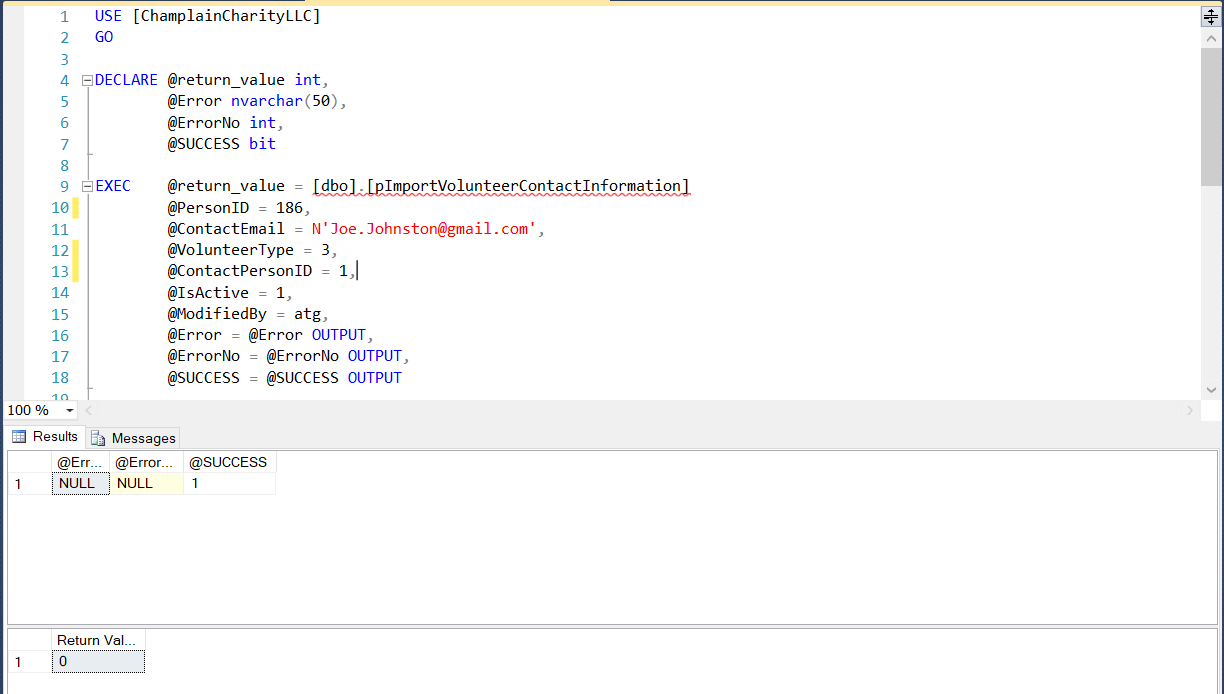
Every single insert procedure has been tested that I created, my previous deliverables have the picture evidence. All of my reports have been tested as well, making sure it does what it is supposed to do. Before I fully had a system for scheduling I had to manually enter in a row to test everything. Once I fixed the pInsertVolunteerContactInformation procedure I was able to enter in some information with that procedure to give me more data to test. I do not know of any big bugs in the system, I am aware that the current procedures that govern the scheduling logic only work with one set of inputs, however as most of this work is done on the front end, this should not be a problem. I have created a procedure that takes some inputs and uses a function to get some information from various tables in the Person Schema and insert it into VolunteerContactInformation, along with some other inputs at the start of the procedure. I was getting a double input bug, but I discovered the source and narrowed it down to only insert one row per individual.

**Patch Notes/Database Changes**

Changed VolunteerPersonID to be identity (Thought it was the source of a bug, it was not, but these changes are much nicer)

Altered insert statements on the table to not accept input for VolunteerPersonID, as a consequence of this being the identity column and primary key I deleted the ContactInformationID as it was redundant and not useful for my purposes.

Figured out the bug, and altered the function to narrow the data pulled from all the tables in the Person schema. I was having two inputs happen as the person had two addresses, I narrowed it to only importing the home address.



This shows that the import procedure works. I technically could import the email address from the person.person table, but no one had any entered so I decided to have it be user input. If email addresses were added to the table I could easily alter the function and procedure to match.

**After Action Report**

This class stretched me a lot, I learned a lot about how SQL works on a larger scale. I learned more about error handling and I learned so much about how everything works together. If I was to alter how I approached this class, I would probably have taken things a little slower. I tended to aim high with what I wanted to do, get overwhelmed, and then dial things back to what is reasonable. I often found things that might have worked for what I wanted to do, but I was way out of my knowledge level. Seeing those things, and seeing what was possible has given me a hunger to learn more and get better at SQL. Working out how to get the last piece of the import procedure to work was a fun challenge, I was struggling and just let myself think on it for a day or two, and then it worked. It helped me to see how much you can do with a user-defined table. I still barely understand error handling, but I now know enough to figure out how to make it work.

My favorite challenge though was cleaning up the database those first two weeks, I definitely went a little overboard on it. However, the challenge of switching all of those columns and all of their values and making all the links was fun. As I said earlier, when I got to the end of the scheduling logic, I realized that if the customer wanted me to, I could have combined the daytime table and the jobs table. I wanted to leave them separate, but it would be a slightly time-consuming job, but not too difficult. As the class went on I got a better feel for the layout of the database, and if I would have known what I know now, I probably would have done more alterations from the start. However I am happy with where things ended up.