I didn’t make any changes to the schema of the database for the implementation.

My program files are fairly straightforward. I have a single driver cpp file that handles all of the implementation. I didn’t add any additional classes or structs but rather handled the data as it was retrieved. At the beginning of my program a Session object is constructed to connect to MySQL. From there I query the session to retrieve or create the database ‘Experiment’. Once I switch to using the Experiment database I create all of the tables if they don’t already exist. This concludes the setup for the application. From there I prompt the user with a menu and read in the user’s input. I have a switch statement to determine what option the user selected. Each of the tasks that the user can do has been broken into a separate function. One of the main things about my application is that when a user goes to insert an experiment or a run, it is all or nothing. They cannot break the pieces apart and insert just a single result or parameter into the database. For inserting an experiment, I ask the user for the experimentId first, making sure there isn’t already an experiment in the table with that id. From there I ask the user for the rest of the meta data. There isn’t any checking that needs to be done when inserting an experiment, so I ask for and enter the experiment meta data along with the parameters and results and insert it all into the database. For inserting a run, it becomes more complicated since there are quite a few more checks that need to happen. First, after locating a valid experiment, you must enter all the required parameters and then may optionally enter any parameters that aren’t required. The same goes for results of the experiment. When inserting either a run parameter or run result, I use regex to make sure the value the user enters, matches with the type of the parameter. To look up information about the experiment, I ask the user for the experimentId and making sure that it exists in the database, I query for the meta data and print it, then I query the ParameterType table and the ResultType table matching to the experimentId to print out all of the meta data about the parameters and results. To look up a run it is mostly the same except first I print out all possible runs that belong to the experimentId that the user provides. Then I let the user select which run they’d like to display the information about. From there printing out the meta data about the run, run parameters, and run results is similar to printing the experiment meta data. For the experiment report it is nearly the same as fetching data about the experiment except I open an html file and print the data along with html tags for creating tables. The aggregate report option prompts the user to pick a list of parameters for the experiment they provided that may be aggregated, which are any that are an int or float. Once they’ve selected the parameter I ask for a min and max date and then print the aggregated values which are calculated using SQL queries. I make sure the user doesn’t enter the same date for the start and end date. Lastly for the parameter search I ask the user for a parameterName and type and use a SQL query to retrieve the meta data of each experiment with that parameter, by joining the Experiment and ParameterType table.