User Guide:

On entering the program, a prompt appears where the name of the database to be used should be entered. Be sure to enter the correct database name, as if the entered name is not an existing database then a new database with that name will be created. After the name of the database has been entered, an interface showing several options will appear. Each option is on its own line and is assigned a number that can be entered to select an option. From this menu, there is also the option to quit the program by entering ‘q’ and then pressing the ENTER key. When a function is selected by typing its number then pressing the ENTER key, there is no longer an option to quit the program until the function is completed, at which point the main menu will return. Each function has its own interface, which depends on the data it requires to complete its task:

show_current_reviewers:

The purpose of this function is to first show all the papers allowing only one to be selected using the display_pages functionality. After quitting this function, a prompt appears to either enter the index of a paper which will be accepted as long as it is valid, or quit the function and go back to the list of functions to choose. If the user inputs a valid index, the email of all reviewers who have reviewed that paper will be shown.

show_potential_reviewers:

The purpose of this function is to first show all the papers allowing only one to be selected using the display_pages functionality. The user needs to press ‘q’ to quit this function after which a prompt appears for the user to select the index of the name of a paper. Once a paper is selected a list of potential reviewers is displayed for that paper. This list does not include the those reviewers who have already reviewed the selected paper. The user is prompted to choose the index of a reviewer. If the user chooses a reviewer who is not on the list or if the reviewer is the same as the author, the program warns the user of invalid input and asks for input again. This loop continues until the user enters a valid input or presses ‘q’ to quit this function. Once the user has entered a valid input the program asks for scores in each category separately and calculates the overall score. The program then inserts a new row in the reviews table of the database to reflect the new review on the selected paper.

get_reviews_in_range:

The purpose of this function is to show all reviewers who have reviewed an amount of papers between two bounds, inclusive of the bounds. When this function is

selected, a prompt appears to enter a number to be used as one of the bounds of the range to display reviews within. Once an integer is entered, another prompt appears asking for another bound. The upper and lower bounds can be entered in either order, but invalid input in either prompt will cause an error message, then both bounds must be re-entered. Negative integers may be entered, but only reviewers who have reviewed zero or more papers and have a number of reviews within the bounds will be shown. After the two bounds have been entered correctly, reviewers that have reviewed an amount of papers between the two bounds are output, and the main menu will reappear.

show_author_participation:

This function shows in how many sessions the authors participate in the format that the user chooses. For this, a prompt appears to select two options: 1 to show a bar plot of all individual authors and the number of sessions they participate in regardless if they participate in that session more than once; and 2 to enter any of the authors who participate in the sessions. To make it more readable, after the user presses 2, a list of all the authors who participate in the sessions appears. The user is then prompted to choose the name of the author. If the user chooses an author who is not in the list presented, or inputs an invalid answer, the program will warn the user about the invalid input and ask for input again or to press q to quit the function. If the user chooses a valid author, the number of sessions in which the chosen author participates will be shown.

most_popular_areas:

This function finds the five most popular areas, as well as areas tied for being in the top five. When this function is selected, it will query the database previously input to the program and display the top 5 areas with ties in a labelled pie chart.

show_avg_review_scores:

The purpose of this function is to show the average scores given by each reviewer in each category. When the user selects this function, a single grouped bar chart representing the information is displayed.

Software Design:

First, the **main** function was designed to query the user for the database then go into a menu displaying the possible functions (each of which takes in the database connection and the cursor as arguments) and giving the option to quit the program. After asking the user for the name of the database, the function **connect** sets up the database and gets a

connection/cursor. For code simplicity and cleanliness, the names of the user-accessible functions are stored in an array in the **main** function, then function names and their indices in the array are shown to the user. For convenience, an index is prompted for instead of a function name. There is error handling implemented during this process to ensure that the input is an integer that is a valid index in the function array for the purposes of selecting a function. The first two user-accessible functions have similar functionality in that they both need to page through all the papers and select one, so the function **display_pages** was created to allow the user to look through the titles of the papers in batches of 5. These functions also both need to select a paper, so another function **get_valid_input** was made to call **display_pages** and prompt for a single paper to be selected by its index. This function returns the dataframe containing the names of all the papers and the index of the selected paper to its caller, in case the caller needs access to the whole list of papers.

The function **show_current_reviewers** calls **get_valid_input** and then queries the database to find all reviewers of the selected paper. The function **show_potential_reviewers** also calls **get_valid_input**, then prompts for a potential reviewer to be chosen until a reviewer that is allowed to review the paper is entered. It is assumed that authors are not allowed to review their own paper. Once a valid reviewer is entered, the user is prompted to make a new entry for the reviewer's review scores for the paper.

The function **get_reviewers_in_range** prompts the user for two bounds. Both are placed in the same try/except block which means an error in entering the second bound will make the user re-enter the first bound as well, but the code is nicer this way. However, the bounds can be entered in either order for user convenience and to reduce the amount of error handling needed. The query used in this function assumes that anybody in the users table can review papers. Then the dataframe containing the reviewers is converted to a list to be printed because the user does not need to know the dataframe indices.

The function **show_author_participation** prompts for an integer input (either 1 or 2) to either view a barplot of the amount of sessions for all individual authors or view the number of sessions a specific author participates in. If the first option is selected, then a bar plot is generated and the function is exited, otherwise the user is shown the potential authors and their indices and prompts the user to select the index of an author. Error handling is in place to make sure the author exists. After the author is selected, the number of sessions they participate in is shown to the user.

The functions **most_popular_areas** and **show_avg_review_scores** do not prompt the user for input. They only output graphs. The function **most_popular_areas** queries the database to find the five most popular areas (as well as all other areas tied for being in the top five) ignoring all the areas with no papers related to them. The function

show_avg_review_scores just uses an SQL query to find the average score in each category given by each reviewer and displays them in a bar chart.

Testing Strategy:

We first checked the main, display_pages and get_valid_input functions as they are used by multiple functions. We tested for valid inputs to see if the functions were responding in the expected way. Then we entered invalid inputs to ensure that the code handles errors appropriately. After the common functions we checked the main function again to make sure that it transitions between the different functions without errors. We then checked each of the following six functions and repeating a similar procedure as above, testing valid and invalid inputs. To make the interface consistent we checked the output of all the functions to see if common errors are handled similarly. Finally, all the group members tried to break different parts of the code in order to ensure that all errors were handled appropriately. The vast majority of bugs found related to either recovering in the case of invalid input or needing to change the code as forum posts clarified the implementation details of functions.

Work Breakdown:

We broke the work down by giving each person one function worth 10 marks and one function worth 15-20 marks to implement, where one person doing a question worth 15 marks also made the main interface. Each person was responsible for writing, commenting, testing, and writing the user guide for their own code. We coordinated through email to meet up and work on the project, as well as discuss implementation details and figure out what needed to be done. During our meetings we discussed the ways each person was implementing his/her part of the code so that the code as a whole was consistent. All the group members together checked the error handling of the entire code. Each person completed their portion of work in good time and contributed relatively equal amounts of work to the project. A rough estimate of the time spent on this project would be over 12 hours per person.