Team 11 Report

Adarsh Cavale (aac90), Lucas Leiby (lwl10), Avery Peiffer (aep65)

## Users’ Manual

In order to use Team 11’s driver program, please follow the steps below:

1. Download or clone the Git repository.
2. Run the team11.sql script.
3. Run the team11.java driver program to begin interfacing with our application.
4. You will be presented with a command-line program that will prompt you for keyboard input to make certain choices.
5. You are first presented with the choice for performing customer or administrative functionality.
6. After you’ve selected customer or administrative functionality, you will be presented with a list of tasks that can be performed. Type the appropriate number of the task you wish to perform.
7. Once the task has completed, you can select to perform another operation or exit the program.

## Limitations

### Missing or Incomplete Functionality

Customer:

Task 7: We can grab the info from the user that they wish to add to the database, but there is a conflict with the reservation number when actually performing the insert statement.

### Improvements to be Made

Administrative:

Task 1:

Checking if database is already empty: It would be a nice improvement on efficiency if we could check if the database was already empty before performing all of those operations.

Tasks 2-5:

Verifying File Input Correctness: When processing files, the current functionality assumes that there are N number of values per line but does not check for valid input (e.g.: the first value of a line is an integer. The program assumes it is).

Task 6-7:

Input validation: There could be additional checks to ensure that the dates and times are inputted in the correct format. The tasks give an example of the formatting to be used, but there is nothing to check whether the specific format was used.

Customer:

Task 1:

Input Validation: When gathering the user’s information, there can be checks to ensure the input is valid and in the correct format. The current functionality assumes the user’s input is precisely the exact format needed to be inserted into the database.

Task 4:

Input Validation: There could be additional checks to ensure that the cities are inputted in the correct format. The driver says that the cities should be inputted in uppercase, using the three letter abbreviations found in the flight data. But there are no checks to enforce this explicitly.

Task 9:

Information Output and Readability: Dates and timestamps that appear in the resulting output tables can be displayed in a more user-friendly fashion (e.g.: 2020-08-23 16:30:0.0 may be easier to read as November 23, 2020 4:30PM). Whether a ticket has been purchased or not could be shown as “Yes” or “No” instead of “true” or “false.”

Task 10:

Input Validation: There are no checks to enforce that a number was entered. The system will fail gracefully if a number is entered, but it does not exist in the database, but there are no checks in the driver to specify that it must be an integer.

Tasks 11-12:

Input Validation: There are no checks to make sure that k is an actual number.

Better Handling of Ties: It might be worthwhile to provide the user with an option as to how to handle ties. We approached it by just including all customers who have an equal value. If the user asks for the top 2 customers by number of legs traveled, and three customers had 9, 8, and 8 legs traveled, respectively, the system will return all three customers since there is a tie. This makes intuitive sense but should be left up to the user’s discretion.

Moving Queries to Functions: These two tasks had long queries that were tested using functions that created views. The function that was used for Task 11 can be found around Line 520 of team11.sql. I (Avery) could not find out what was wrong with it, but when I executed all of the statements outside of the function it worked correctly. Since this was low on our priority list, I just moved the query to the driver program instead. The same is true for Task 12.