CostaExpress TLDR Guide

Emma(Qinnuo) Li, Anastasia Mokhon, Samantha Lavrinc

Getting Started:

** Please view the extended guide if any errors occur.

To begin the database setup, run the following sql files in datagrip:

- createDB.sql
- getpricefunction.sql
- updatePhase3_v2.sql
- updatePhase3_v3.sql
- updatePhase3_v4.sql

Ensure that all views have been created within the functions of the sql files.

Note that the dbpack file is a package. If running the file returns an error stating it cannot find these classes, that means that your local system is not recognizing dbpack as a package file.

Launch the java file from the root (Demo) directory using the command:

With Windows using chocolatey:

```
Make
java -cp "postgresql-42.2.18.jar;." dbpack/CostaExpress
```

or with Mac/Linux:

```
Make
java -cp postgresql-42.2.18.jar:. dbpack/CostaExpress
```

Select Admin > Login with password > AllData.txt, with no file name, and select ‘Import’

The import statement should have generated the demonstration data provided by the project outline which can be verified using select statements in datagrip.

Then, run the following sql file in datagrip:

- updatePhase3_v1.sql

Verify that the most recent sql statement has updates the stop_seatcount table with the command SELECT * from stop_seatcount;

The demo program should now be fully functional, no need for a restart.

Launch Screen:

By clicking 'Agent,' the program is directed to the Agent Login Screen.
By clicking 'Admin,' the program is directed to the Admin Login Screen.

Login Screen:

Note: Username postgres is default and does not need to be entered to obtain access, however a user password must still be added.

Agent Access:

Select one of the menu options.

- Add Customer gives the option to add a Customer to the database.
- Edit/View Customer gives the option to edit and view database Customers.
- Search Database provides a menu of search options including:
    - Search by Single Route
    - Search by Combination Route
    - Advanced Search Options A – I
- Add Reservation gives the option to add a new reservation.
- Update Reservation gives the option to update an existing reservation.
- Logout returns the user to the login menu.

Add Customer:

Enter the requested information and select the 'Add' button.
On successful creation, a customer ID is generated and returned into the Customer ID field.

Edit/View Customer:

Enter either the (Customer ID) or (First Name, Last Name and Email Address).
Then select 'Get Customer.'
The Customer's information is displayed in the relative fields and can now be updated.
To update, select the 'Save' button.

Search Options:

The Search Option Menu corresponds to the original assignment pdf for search parameters.
Additionally, we added a Search Customer Reservation feature.

The Search Selections are as follows:

- Single Route -
- Combo Route -
- Advanced Searches

    - A – Find all trains that pass through a specific station at a specific day/time combination
    - B – Find the routes that travel more than one rail line
    - C – Rank the trains that are scheduled for more than one route
    - D – Find routes that pass through the same stations but don't have the same stops
    - E – Find any stations through which all trains pass
    - F – Find all the trains that do not stop at a specific station
    - G – Find routes that stop at least at XX% of the stations they visit
    - H – Display the schedule of a route
    - I – Find the availability of a route at every stop on a specific day and time
- Back Button – Select to return to the previous screen.

Single Route Search:

Enter the ID number for the Start and End station, as well as the day and a sorting option.
Clicking the 'Search' button populates the result table.

Combination Route Search:

Enter the ID number for the Start and End station, as well as the day and a sorting option.
Clicking the 'Search' button populates the result table.
The First route is the first 'leg' of the journey, with the end station that functions as a layover station.
The Second route is the second 'leg' of the journey, with the start station that functions as a layover station.

Advanced Search (A):

Finds all trains that pass through a specific station during a specific day/time combination.

- Enter the Station ID, day and time.

Advanced Search (B):

Finds the Routes that travel more than one rail line.

Advanced Search (C):

Ranks the trains that are scheduled for more than one route.

Advanced Search (D):

Finds the routes that pass through the same stations but don't share the same stops.

Advanced Search (E):

Finds any stations through which all trains pass.

Advanced Search (F):

Finds all trains that do not stop at a specific station.

- Enter the Station ID and select 'Search.'

Advanced Search (G):

Finds routes that stop at least at XX% of the stations they visit, where XX is a number from 10 to 90.

- Enter the Percentage as (XX) or 50 for 50% and select 'Search.'

Advanced Search (H):

For a specified route, list the days of departure, departure hours and trains that run it.

- Enter the Route ID and select 'Search.'

Advanced Search (I):

Finds the availability of a route at every stop on a specific day and time.

- Enter the desired Route ID, Day and Time as 00:00:00 and select 'Search.'

Reservation Entry:

Obtain Trip Price:

Begin by entering the Start Station ID, End Station ID, Time, Day and Route which can be obtained from one of the route search options. Select the 'Get Price' button to obtain the price of the trip.

Add Reservation:

After obtaining the trip price, select whether the Customer would like to utilize a service to adjust their reservation based on a downed line. Then, enter the Customer ID and Amount Paid. By clicking 'Add Reservation,' the Agent is saving the customer's current balance in the system and if the balance is $0, will generate a ticket number.

Edit Reservation:

To update a reservation, it is assumed that the only thing that would need edited about a reservation is the balance field. If a customer wishes to book a reservation with different stops, at a different time or on a different day, an agent may then instead delete the current reservation and create a new reservation with the updates.

Begin by entering the reservation number and selecting 'Get Reservation' to generate the reservation's corresponding data.

- To update the Customer's reservation balance, enter the current payment amount and select the 'Save' option. If the reservation balance is now $0.00, a new ticket is generated.
- To delete the Customer's reservation, instead select the 'delete' option.

Admin Screen:

Import:

AllData.txt – No file name should be entered. This option uses the AllData.txt file in the root directory to generate demo data.

All other options – These options can import data that was generated *using the program's export option.* Thus, this option should not be used unless Export has generated export .txt files from the database.

If data has been exported using the Export button, select the desired table data to import and add a direct file name to the location of the export data, be sure to fill in the entire path.

EX: C:\... all this stuff too...\CS1555_CostaExpress\export\tickets.txt

Export:

Before using the export function:

> Open updatePhase3_v4.sql and update the export file location to your system specific export file location. Run updatePhase3_v4.sql to save these updates.

You may now safely export the database data to the chosen export file. This can be verified with new .txt documents reflecting the name of the table.

Delete Database:

This brings up an option to either confirm the deletion or cancel. Select 'Yes' to delete the data within the database.

Final Thoughts:

Throughout this project, our group was gradually exposed to additional functionality provided by PostgreSQL and JDBC. For most of us, it was our first exposure to SQL and for all of us, it was our first experience writing a fully functioning interactive program from scratch. Approaching this opportunity as a learning experience, many of our early design choices were made with minimal awareness of features like triggers, procedures, views and functions. For example, looking back with the knowledge we obtained from class, it would have been beneficial to create smaller functions to traverse lines and obtain the distance or station numbers from a start station to an end station much earlier in development. It would have also been interesting to add additional functionality and features to the resulting program, perhaps creating a more cohesive display that combined batch results of multiple input parameters. As it currently functions, we have multiple classes for each one of our displays that serve to demonstrate the project requirements, but many of these could have easily been combined, like Add and Edit Customer or Add and Edit Reservation. Or with consideration to use, a program that displays each entity as a searchable/editable object, where each attribute that represents a relationship also functions as a button that reveals a display of the corresponding attributes for the entity.

The next major steps in development would be to implement additional security to the password feature, fully roll out line_disruption trigger implementation, and work on program access to demonstrate the functionality of the reservation_cancel trigger. Additionally, while the basic logic of the sorting functionality within single route search produces results that are sorted based on price, time, stations and stops, because we didn't implement the smaller functions to obtain the distance, or station numbers based on an entry and exit station, neither the single or combination search options display the resulting price, time, stops or stations. The pricing functionality based on distance is implemented on the add reservation screen, so there is a way to obtain the information, but it is not conveniently displayed in the search screen.