Functional Specification

Team Name: Cloud Nine
Team Members: Jeremy Maas, Matt Burton, McHale Trotter, Kevin Sampson, Justin Chen, Ryan Hirscher

Table of Contents

Introduction

This document serves as a detailed introduction to the codebase highlighting the purpose and use of files, functions, schemas, and conditions involved in each. The overall codebase is written in Python, and the database in SQL.

Naming Conventions

Type	Example
Files	`this_file_name.py`
Functions/Methods	`this_function_name`
Classes	`ThisClassName`
@Properties	`property_name`
Local variables	`variableNames`
Private variables	`_underscorePrefixed`
Constants	`CONSTANT_NAME`
	(Note: constants must go at the top of the file)
Attributes	`attribute_name_convention`

Module Information

populate-aircraft-table.sql

Location: comfort-airlines/docker/sql-files/populate-aircraft-table.sql
Purpose: Replaces the entities in the aircraft table with the hardcoded list of aircraft in this file.
Execution: Once inside the cloudnine database, ensure that the schema has been initialized, then run:

```
source /docker-entrypoint-initdb.d/populate-aircraft-table.sql
```

populate-airports-table.sql

Location: comfort-airlines/docker/sql-files/populate-airports-table.sql
Purpose: Replaces the entities in the airports table with the hardcoded list of airports in this file.
Execution: Once inside the cloudnine database, ensure that the schema has been initialized, then run:

```
source /docker-entrypoint-initdb.d/populate-airports-table.sql
```

populate-flights-table.sql

Location: comfort-airlines/docker/sql-files/populate-flights-table.sql
Purpose: Replaces the entities in the flights table with the hardcoded list of flights in this file.
Execution: Once inside the cloudnine database, ensure that the schema has been initialized, then run:

```
source /docker-entrypoint-initdb.d/populate-flights-table.sql
```

schema.sql

Location: comfort-airlines/docker/sql-files/schema.sql
Purpose: Empties and replaces the existing tables with the database schema as defined in this file.
Execution: Once inside the cloudnine database, initialize the schema by running:

```
source /docker-entrypoint-initdb.d/schema.sql
```

maintenance_exception.py

Location: comfort-airlines/exceptions/maintenance_exception.py
Purpose: Handles error logging for invalid maintenance calls

aircraft_menu.py

Location: comfort-airlines/menus/aircraft_menu.py
Purpose: Responsible for implementing the menu options under the aircraft menu option

Method Name	Purpose	Parameters	Return Values
`view_aircraft()`	Prints the aircraft entities from the aircraft table	None	None
`print_aircrafts_header()`	Formats and prints the column headers for aircraft attributes	None	None
`print_aircraft(aircraft)`	Formats and prints the information of the aircraft object passed in	`aircraft`: dataframe row	None
`edit_aircraft()`	Displays the list of editing options	None	None
`add_aircraft()`	Adds an aircraft to the database from user input	None	None
`remove_aircraft()`	Prints the aircraft table and then allows user to remove an aircraft by tail_number	None	None
`get_valid_tail_number()`	Returns valid tail_number from user input or 'quit' if the user cancels	None	`str` or `'quit'`
`get_valid_name_or_model(input_type)`	Returns valid string from user input or 'quit' if the user cancels	`input_type`: str	`str` or `'quit'`
`get_int_value(input_type)`	Returns valid integer value from user input or 'quit' if the user cancels	`input_type`: str	`int` or `'quit'`

airport_menu.py

Location: comfort-airlines/menus/airport_menu.py
Purpose: Responsible for implementing the menu options under the airport menu option

Method Name	Purpose	Parameters	Return Values
`view_airports()`	Queries the database for the airports table and prints the entities	None	None
`print_airports_header()`	Formats and prints the column headers for airport attributes	None	None
`print_airport(airport)`	Formats and prints the information of the airport object passed in	`airport`: dataframe row	None
`edit_airport()`	Displays the list of editing options	None	None
`add_airport()`	Adds an airport to the database from user input	None	None
`remove_airport()`	Prints the airport table and then allows user to remove an airport by abbreviation	None	None
`get_valid_name()`	Returns valid airport name from user input or 'quit' if the user cancels	None	`str` or `'quit'`
`get_valid_abbreviation(removingAirport)`	Returns valid airport abbreviation from user input or 'quit' if the user cancels. If removingAirport is True, checks if abbreviation exists in the database	`removingAirport`: bool	`str` or `'quit'`
`get_valid_latitude()`	Returns valid latitude from user input or 'quit' if the user cancels	None	`float` or `'quit'`

Method Name	Purpose	Parameters	Return Values
`get_valid_longitude()`	Returns valid longitude from user input or 'quit' if the user cancels	None	`float` or `'quit'`
`get_valid_timezone_offset()`	Returns valid timezone offset from user input or 'quit' if the user cancels	None	`int` or `'quit'`
`get_valid_metro_population()`	Returns valid metro population from user input or 'quit' if the user cancels	None	`int` or `'quit'`
`get_valid_is_hub()`	Returns valid is_hub value from user input or 'quit' if the user cancels	None	`int` or `'quit'`
`get_valid_total_gates(metroPopulation, isHub)`	Returns valid total gates value from user input or 'quit' if the user cancels. Calculates maximum possible gates based on metroPopulation and isHub values	`metroPopulation`: int, `isHub`: int	`int` or `'quit'`

configuration_menu.py

Location: comfort-airlines/menus/configuration_menu.py
Purpose: Responsible for implementing the menu options under the configure simulation menu option

Method Name	Purpose	Parameters	Return Values
`read_config()`	Returns the simulation configuration from the JSON file	None	`dict`
`write_config(config)`	Writes the simulation configuration to the JSON file	`config`: dict	None
`configure_start_date()`	Configures the start date of the simulation	None	None
`configure_duration()`	Configures the duration of the simulation	None	None
`configure_report_frequency()`	Configures the report frequency of the simulation	None	None
`get_report_frequency_options(duration)`	Determines the available report frequency options based on the duration of the simulation	`duration`: int	`list`
`format_options_for_print(options)`	Formats the report frequency options for display	`options`: list	`str`
`ensure_valid_report_frequency(config, duration)`	Ensures that the selected report frequency is valid based on the duration of the simulation	`config`: dict, `duration`: int	`str`
`configure_costs()`	Displays the costs submenu	None	None

cost_menu.py

Location: comfort-airlines/menus/cost_menu.py
Purpose: Responsible for implementing the menu options under the configure costs menu option

Method Name	Purpose	Parameters	Return Values
`read_config()`	Returns the simulation configuration from the JSON file	None	`dict`
`write_config(config)`	Writes the simulation configuration to the JSON file	`config`: dict	None
`configure_fuel_cost()`	Configures the fuel cost for the simulation	None	None
`configure_takeoff_cost()`	Configures the takeoff cost for the simulation	None	None
`configure_landing_cost()`	Configures the landing cost for the simulation	None	None
`configure_leasing_costs()`	Configures the leasing costs for different aircraft models	None	None

Method Name	Purpose	Parameters	Return Values
`retrieve_aircraft_models_and_costs()`	Retrieves the aircraft models and their leasing costs from the database	None	`dataframe`
`display_aircraft_models_and_costs(dataframe)`	Displays the current leasing costs for each aircraft model	`dataframe` : dataframe	None
`handle_model_input(leasingCosts)`	Handles user input for selecting an aircraft model	`leasingCosts` : dict	`str`
`handle_leasing_cost_input(model, leasingCosts, config)`	Handles user input for updating the leasing cost of an aircraft model	`model` : str, `leasingCosts` : dict, `config` : dict	None
`update_leasing_costs_in_database(leasingCosts)`	Updates the leasing costs of aircraft models in the database	`leasingCosts` : dict	None
`is_valid_dollar_value(inputString)`	Checks if the input string represents a valid dollar value	`inputString` : str	`bool`

simulation_menu.py

Location: comfort-airlines/menus/simulation_menu.py
Purpose: Responsible for implementing the menu options under the simulation menu option

Method Name	Purpose	Parameters	Return Values
`run_simulation()`	Runs the simulation	None	None
`configure_simulation()`	Displays the menu for configuring simulation options	None	None
`analyze_simulation()`	Displays the analyze simulation submenu	None	None
`analyze_follow_aircraft()`	Prints 'Executing analyze_follow_aircraft()'	None	None
`analyze_download_reports()`	Prints 'Executing analyze_download_reports()'	None	None

timetable_menu.py

Location: comfort-airlines/menus/timetable_menu.py
Purpose: Responsible for implementing the menu options under the timetable main menu option

Method Name	Purpose	Parameters	Return Values
`view_timetable()`	Displays the timetable	None	None
`search_routes()`	Prints 'Executing search_routes()'	None	None
`edit_timetable()`	Displays the edit timetable submenu	None	None
`download_timetable()`	Prints 'Executing download_timetable()'	None	None
`sort_by_cost()`	Prints 'Executing sort_by_cost()'	None	None
`sort_by_number_of_stops()`	Prints 'Executing sort_by_number_of_stops()'	None	None
`sort_by_departure_time()`	Prints 'Executing sort_by_departure_time()'	None	None
`add_flight()`	Prints 'Executing add_flight()'	None	None
`remove_flight()`	Prints 'Executing remove_flight()'	None	None
`upload_timetable()`	Prints 'Executing upload_timetable()'	None	None

aircraft.py

Location: comfort-airlines/objects/aircraft.py
Purpose: Represents aircraft objects in the simulation. Important for tracking the dynamic information of each aircraft

Attribute Name	Type	Unit
`_id`	int	
`_tailNumber`	string	
`_name`	string	
`_model`	string	
`_maximumSpeed`	int	mph
`_maximumCapacity`	int	passengers
`_maximumFuel`	int	gallons
`_currentFuel`	int	gallons
`_cargoVolume`	int	cubic feet
`_leasingCost`	int	USD
`_timeSinceLastMaintenance`	int	minutes
`_requiresMaintenance`	bool	

Method Name	Purpose	Parameters	Return Values
`timeSinceLastMaintenance(self, durationOfLastFlight)`	Set the time since last maintenance and updates the requires maintenance value	`self` : aircraft, `durationOfLastFlight` : int	None

airport.py

Location: comfort-airlines/objects/airport.py
Purpose: Represents airport objects in the simulation. Important for tracking the dynamic information of each airport

Attribute Name	Type	Unit
_id	int	
_name	string	
_abbreviation	string	
_latitude	float	degrees
_longitude	float	degrees
_timezoneOffset	int	hours
_metroPopulation	int	people
_totalGates	int	
_availableGates	int	
_isHub	int	binary

Method Name	Purpose	Parameters	Return Values
`remove_gate(self)`	Attempts to increase the avaialable gates by one	`self` : airport	None

Method Name	Purpose	Parameters	Return Values
`add_gate(self)`	Attempts to decrease the available gates by one	`self` : airport	None

flight.py

Location: comfort-airlines/objects/flight.py
Purpose: Used to mirror the flights in the database and update their values during the simulation

Attribute Name	Type	Unit
_id	int	
_number	string	
_aircraftID	int	
_departureAirportID	int	
_destinationAirportID	int	
_angleOfFlight	float	degrees
_duration	int	minutes
_departureTime	int	minutes
_arrivalTime	int	minutes
_onTimeBin	int	binary

Method Name	Purpose	Parameters	Return Values
duration(self, duration)	Sets the flight duration and the arrival time	`self` : flight, `duration` : int	None
arrivalTime(self, newArrivalTime)	Sets the arrival time and updates the on time value	`self` : flight, `duration` : int	None

aircraft_objects.py

Location: comfort-airlines/simulation/aircraft_objects.py
Purpose: Create and store a dictionary of aircraft objects used in the simulation to manage their dynamic information
Global Variable: `aircrafts` : dictionary containing all of the aircraft objects in the simulation

Method Name	Purpose	Parameters	Return Values
create_aircrafts_from_database()	Returns the aircraft entities from the database in a dictionary	None	`dict`

airport_objects.py

Location: comfort-airlines/simulation/airport_objects.py
Purpose: Create and store a dictionary of airport objects used in the simulation to manage their dynamic information
Global Variable: `airports` : dictionary containing all of the airport objects in the simulation

Method Name	Purpose	Parameters	Return Values
create_airports_from_database()	Returns the airport entities from the database in a dictionary	None	`dict`

report.py

Location: comfort-airlines/simulation/report.py
Purpose: Generates reports about the simulation

Method Name	Purpose	Parameters	Return Values
`handle_report(config, minutes)`	Handles report errors and calls the generate_report() function	`config` : dict, `minutes` : int	None
`should_generate_report(config, minutes)`	Returns true if the simulation time and report frequency align to generate a report	`config` : dict, `minutes` : int	`bool`
`generate_report()`	Prints 'Executing generate_report()'	None	None
`is_start_of_day(minutes)`	Returns true if minutes is the start of a day	`minutes` : int	`bool`
`is_start_of_week(minutes)`	Returns true if minutes is the start of a week	`minutes` : int	`bool`
`is_start_of_month(minutes)`	Returns true if minutes is the start of a month	`minutes` : int	`bool`
`is_start_of_year(minutes)`	Returns true if minutes is the start of a year	`minutes` : int	`bool`
`is_end_of_simulation(config, minutes)`	Returns true if minutes is the end of the simulation	`config` : dict, `minutes` : int	`bool`

schedule.py

Location: comfort-airlines/simulation/schedule.py
Purpose: Stores a singleton list of all simulation events and at what times they occur

Method Name	Purpose	Parameters	Return Values
`get_instance(cls)`	Used for retrieving the singleton instance of the schedule	`cls` : Schedule	`Schedule`
`clear_schedule(self)`	Removes all events from the schedule	`self` : Schedule	None
`add_event(self, event)`	Appends an event to the schedule at the time of the event	`self` : Schedule, `event` : ScheduledEvent	None
`get_events_for_minute(self, minute)`	Returns the list of events at a given time	`self` : Schedule, `minute` : int	`list`

scheduled_event.py

Location: comfort-airlines/simulation/scheduled_event.py
Purpose: Handle various event types, storing references to the associated information

Scheduled Event Class

Method Name	Purpose	Parameters	Return Values
`__init__(self, eventType, time)`	Event constructor	`self` : ScheduledEvent, `eventType` : string, `time` : int	None
`execute(self)`	Does nothing by default. Override for each event	`self` : ScheduledEvent	None

Departure Event Class

Method Name	Purpose	Parameters	Return Values
`__init__(self, flight)`	Creates the ScheduledEvent, and stores references to aircraft and airport involved in departure	`self` : DepartureEvent, `flight` : Flight	None
`execute(self)`	Returns an event string stating which aircraft departs from which airport	`self` : DepartureEvent	`string`

Arrival Event Class

Method Name	Purpose	Parameters	Return Values
`__init__(self, flight)`	Creates the ScheduledEvent, and stores references to aircraft and airport involved in arrival	`self` : ArrivalEvent, `flight` : Flight	None
`execute(self)`	Returns an event string stating which aircraft arrives at which airport	`self` : ArrivalEvent	`string`

simulation_config.json

Location: comfort-airlines/simulation/simulation_config.json
Purpose: JSON Dictionary to store the data relating to the simulation configuration options accessible to the user

Value Name	Use	Default Value
`startDate`	Sets the day that the simulation begins	`0`
`reportFrequency`	Sets the interval that reports are generated	`final`
`fuelCost`	Sets the price of fuel per gallon	`6.19`
`takeoffCost`	Sets the price per each aircraft takeoff	`2000`
`landingCost`	Sets the price for each aircraft landing	`2000`
`leasingCost`	Sets the leasing cost of each aircraft model type per month	`737-600` : 245000, `737-800` : 270000, `A200-100` : 192000, `A220-300` : 228000

simulation.py

Location: comfort-airlines/simulation/simulation.py
Purpose: Implements the funcitonality of the user options from the Simulation section of the main menu

Method Name	Purpose	Parameters	Return Values
`run_simulation()`	Runs the main simulation loop and executes events for each minute of the simulation's duration	None	None
`populate_schedule_from_timetable(schedule)`	Resets the event schedule with departure and arrival events based on the flights in the timetable	`schedule` : Schedule	`Schedule`
`create_timetable_from_database()`	Returns a dictionary containing every flight from the flights table in the database	None	`dict`
`get_simulation_configuration()`	Returns the simulation configuration from the JSON file	None	`dict`

generate-timetable.py

Location: comfort-airlines/timetable/generate-timetable.py
Purpose: Produce a list of flights abiding by gate constraints which fly through at least one hub each day

Method Name	Purpose	Parameters	Return Values
`place_aircrafts()`	Allocates all aircraft to an airport	None	None
`generate()`	Generates an aircraft's flight path for the entire day	None	None
`nearest_home()`	Finds the nearest airport that requires more aircraft of this model	None	`Airport`
`choose_random_airport(startAirport, CountToHub, aircraft, CurrentTime)`	Returns a random suitable airport	`startAirport` : Airport, `CountToHub` : int, `aircraft` : Aircraft, `CurrentTIme` : int	`Airport`

timetable.py

Location: comfort-airlines/timetable/timetable.py
Purpose: Implements the funcitonality of the user options from the Timetable section of the main menu

Method Name	Purpose	Parameters	Return Values
`view_timetable()`	Prints the table of all flights from the database	None	None
`print_timetable_header()`	Formats and prints the column headers for flight attributes	None	None
`print_flight(flight)`	Formats and prints the information of the flight object passed in	`flight` : dataframe row	None

clock.py

Location: comfort-airlines/utilities/clock.py
Purpose: Print the simulation time in various formats

Method Name	Purpose	Parameters	Return Values
`get_time(minutes)`	Converts minutes to days, hours, and minutes	`minutes` : int	`days` : int, `hours` : int, `minutes` : int
`print_time(minutes)`	Returns minutes in a human readable format of day and time	`minutes` : int	`string`
`get_flight_time(minutes)`	Used in view_timetable() and returns just the hours and minutes in human readable format	`minutes` : int	`string`

database.py

Location: comfort-airlines/utilities/database.py
Purpose: API for interfacing with the database

Method Name	Purpose	Parameters	Return Values
`connect(self)`	Establishes a connection to the database	`self` : Database	None
`disconnect(self)`	Closes the database connection	`self` : Database	None
`execute_query(self, query, params=None)`	Executes the given SQL query	`self` : Database, `query` : string, `params` : sequence	`cursor`
`execute_query_to_dataframe(self, query, params=None)`	Executes the given SQL query and returns results as a pandas DataFrame	`self` : Database, `query` : string, `params` : sequence	`dataframe`
`execute_insert_update_delete_query(self, query, params=None)`	Executes INSERT, UPDATE, or DELETE SQL query	`self` : Database, `query` : string, `params` : sequence	None

display_menu.py

Location: comfort-airlines/utilities/display_menu.py
Purpose: Displays lists of options and handles user input for selecting an option

Method Name	Purpose	Parameters	Return Values
`display_menu(menu, is_submenu=False)`	Displays an enumerated list of menu options and handles user input for option selection	`menu` : dict, `is_submenu` : bool	None

flight_angle.py

Location: comfort-airlines/utilities/flight_angle.py
Purpose: Returns % of base flight time that a flight will take based on wind and bearing angle

Method Name	Purpose	Parameters	Return Values
`calculate_percentage(startAirport, endAirport, wind = .045)`	Returns a float to multiply with flight time to find the actual time the flight will take	`startAirport` : Airport, `endAirport` : Airport, `wind` : float	`float`

flight_demand.py

Location: comfort-airlines/utilities/flight_demand.py
Purpose: Returns number of people flying from Airport A to Airport B based on metro population

Method Name	Purpose	Parameters	Return Values
`individual_demand(startingAirport, endingAirport)`	Returns number of people flying from Airport A to Airport B based on metro population	`startingAirport` : Airport, `endingAirport` : Airport	`int`

flight_duration.py

Location: comfort-airlines/utilities/flight_duration.py
Purpose: Calculates the duration of flights based on aircraft models, airport locations, and flight angles

Method Name	Purpose	Parameters	Return Values
`calculate_flight_duration(aircraft, departureAirport, destinationAirport)`	Returns the time in minutes for an aircraft to get from its departure airport to its destination	`aircraft` : Aircraft, `departureAirport` : Airport, `destinationAirport` : Airport	`int`
`calculate_total_flight_duration(aircraft, departureAirport, destinationAirport, refueling = False)`	Returns the time in minutes for an aircraft to get from its departure airport to its destination and turn around for its next flight	`aircraft` : Aircraft, `departureAirport` : Airport, `destinationAirport` : Airport, `refueling` : bool	`int`

flight_takeoff.py

Location: comfort-airlines/utilities/flight_takeoff.py
Purpose: Calculates the time for an aircraft to reach cruising altitude or descend form cruising altitude

Method Name	Purpose	Parameters	Return Values
`calculate_acceleration_time(cruisingAltitude, maxSpeed)`	Calculates the time it takes to fly and get into cruising altitude then to reach max speed	`cruisingAltitude` : int, `maxSpeed` : int	`int`
`calculate_descent_time(distance)`	Calculates the time to land using the aircraft's distance away from it's destination	`distance` : float	`int`

great_circle.py

Location: comfort-airlines/utilities/great_circle.py
Purpose: Returns the distance in miles between two airports

Method Name	Purpose	Parameters	Return Values
`great_circle(airportOne, airportTwo)`	Returns the distance in miles between two airports	`airportOne` : Airport, `airportTwo` : Airport	`float`

turn_around_time.py

Location: comfort-airlines/utilities/turn_around_time.py
Purpose: Returns the minimum amount of time in minutes that an aircraft must wait before taking off for its next flight

Method Name	Purpose	Parameters	Return Values

Method Name	Purpose	Parameters	Return Values
`turn_around_time(aircraftNeedsRefuel)`	Returns the minimum amount of time in minutes that an aircraft must wait before taking off for its next flight	`aircraftNeedsRefuel: bool`	`int`

comfort_airlines.py

Location: comfort-airlines/comfort_airlines.py
Purpose: The main executable program for the user interface. Responsible for displaying the main menu and calling the corresponding methods. **Execution:** In the comfort-airlines directory, run the following command in the terminal to run the main executable:

```
python3 comfort_airlines.py
```

Database

Tables

Tables_in_cloudnine
aircraft
airports
flights

Aircraft Table

| Field | Type | Null | Key | Default | Extra | +----------------+-------------+------+-----+---------+----------------+ | aircraft_id | int(11) | NO | PRI | NULL | auto_increment | | tail_number | varchar(20) | YES | | NULL | | | name | varchar(255) | YES | | NULL | | | model | varchar(255) | YES | | NULL | | | maximum_speed | int(11) | YES | | NULL | | | maximum_capacity | int(11) | YES | | NULL | | | maximum_fuel | int(11) | YES | | NULL | | | cargo_volume | int(11) | YES | | NULL | | | leasing_cost | int(11) | YES | | NULL | |

Airports Table

| Field | Type | Null | Key | Default | Extra | +----------------+-------------+------+-----+---------+----------------+ | airport_id | int(11) | NO | PRI | NULL | auto_increment | | name | varchar(255) YES | | NULL | | | abbreviation | varchar(3) | YES | | NULL | | | latitude | float | YES | | NULL | | | longitude | float | YES | | NULL | | | timezone_offset | int(11) | YES | | NULL | | | metro_population | int(11) | YES | | NULL | | | total_gates | int(11) | YES | | NULL | | | is_hub | binary(1) | YES | | NULL | |

Flights Table

| Field | Type | Null | Key | Default | Extra | +---------------------+-------------+------+-----+---------+----------------+ | flight_id | int(11) | NO | PRI | NULL | auto_increment | | flight_number | varchar(20) | YES | | NULL | | | aircraft_id | int(11) | YES | MUL | NULL | | | departure_airport_id | int(11) | YES | MUL | NULL | | | destination_airport_id | int(11) | YES | MUL | NULL | | | angle_of_flight | float | YES | | NULL | | | duration | int(11) | YES | | NULL | | | departure_time | int(11) | YES | | NULL | | | arrival_time | int(11) | YES | | NULL | |

Docker

Location: comfort-airlines/docker/
Purpose: Containerizes the database for cross platform compatibility
Connection: To connect to the running docker container execute the following command in the terminal:

```
docker exec -it mariadb-container mariadb -u admin -p cloudnine
```

docker-compose.yaml

Location: comfort-airlines/docker/docker-compose.yaml
Purpose: Configurations for docker are listed in here which are parsed when to composing up an instance with key value pairs. This specifies the volumes to mount for SQL files and MariaDB data, environment values like database name and password prompt, and GUI for database.
Execution: With the docker daemon running, this container not running, and the working directory being the comfort-airlines/docker/ directory, execute the following command in the terminal to start the docker container:

```
docker-compose up -d
```

Volumes

Volume directories are folders that are linked to the docker container for access within the container.

MariaDB Data

Location: comfort-airlines/docker/mariadb-data/
Purpose: This directory is a volume connected to the docker used to store the database contents.

SQL Files

Location: comfort-airlines/docker/sql-files/
Purpose: This directory is a volume connected to the docker so users can execute these files in the cloudnine database. These files are copied in into the *docker-entrypoint-initdb.d/* directory within the docker.
Execution: Execut the files in this folder by first connecting to the cloudnine database then executing the following command:

```
source docker-entrypoint-initdb.d/<file_name.sql>
```