

E-R Diagram

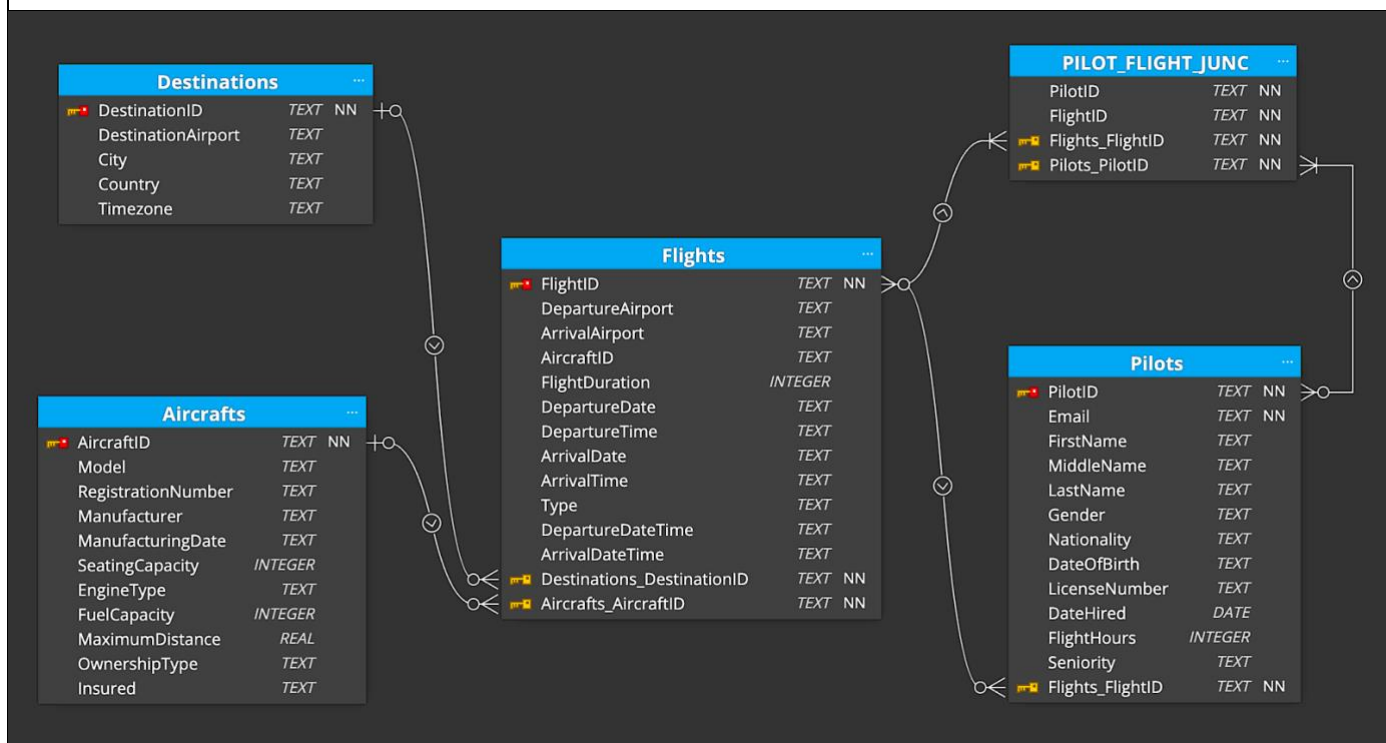

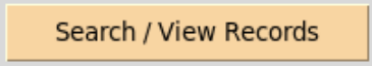





Diagram built with Luna Modeller Mac version 7.4.2 available to download at datasen.com

Notes:

1. An aircraft is assigned to zero to several flights while a flight is assigned to zero to one aircraft – the zero option is allowed in cases of incomplete information, such as when a new flight record is inserted but not yet assigned to any aircrafts.
2. A pilot can conduct zero to many flights and a flight can be assigned to zero to many pilots – the zero option is allowed in cases of incomplete information, such as when a new flight record is inserted but not yet assigned to any pilots.
3. A flight can have zero to one departure airport and zero to one destination airport – the zero option is allowed in cases of incomplete information, such as when a new flight record is inserted but not yet confirmed the departure or arrival airport.
4. The DestinationID, FlightID, AircraftID and PilotID columns are unique identifiers (Primary Key) respectively to the Flights, Aircrafts and Pilots tables.
5. The DATE, TIME and DATETIME datatypes are not supported in SQLite, hence date and time attributes (DateOfBirth, DepartureDate, DepartureTime, DepartureDateTime, ArrivalDate, ArrivalTime, ArrivalDateTime, ManufacturingDate) are set to TEXT data type at table creation. Values are then manipulated using the Date(), Time(), Datetime() functions.
6. DepartureDateTime (DepartureDate + DepartureTime) and ArrivalDateTime (ArrivalDate + ArrivalTime) are both composite attributes, stored in the Flights table as virtual, generated fields.
7. The many-to-many relationship between Pilots and Flights are represented in dependent table Pilot_Flight_Junc. Each record stores an associated pair of PilotID and FlightID.
8. Although the Pilot_Flight_Junc table technically contains two columns that are Primary Keys in another table (PilotID as PK to Pilots table and FlightID as PK to Flights table), the foreign key constraint is only imposed on PilotID, such that association records can be deleted from the Pilot_Flight_Junc table when flight records are deleted from the Flights table.

User Guide	
Starting the App	<p>If starting on original code link:</p> <p>(1) Make sure code on lines 19-26 are uncommented like so:</p> <pre>16 ##### RUN ONLY ON FORKED COPY ##### 17 ##### Otherwise please comment out !!!!! ##### 18 ##### 19 ## create tables 20 [c.execute(command) for command in create_table_commands] 21 ##### 22 #####</pre> <p>(2) Run program.</p> <p>(3) If program terminated, comment out lines 19-26 when running again.</p> <p>If running on forked Replit copy:</p> <p>(1) Make sure code on lines 19-26 are commented out Like so:</p> <pre>##### RUN ONLY ON FORKED COPY ##### ##### Otherwise please comment out !!!!! ##### ##### ## create tables # [c.execute(command) for command in create_table_commands] ##### #####</pre> <p>(2) Run program.</p>
Insert	
Search and View	
Update	
Delete	
Summary Statistics (Flight Finder and Count)	<div></div> <p>The user can get how many flights fulling one or more of the following conditions</p> <ul style="list-style-type: none">• in any given period• from any airport/city/country• to any airport/city/country• departing before/after any time of the day• arriving before/after any time of the day• associated with any pilot(s).

	<ul style="list-style-type: none"> • associated with any aircraft
--	--

Additional Functionalities	
Type	Description
General	<p>Dedicated graphical User Interface using <i>Tkinter</i> Python package for user-friendliness.</p> <ul style="list-style-type: none"> • Real-time views of all tables available on every core functionality page (excl. Flight statistics), using “View Table” button • Real-time views of the E-R diagram available on every core functionality page, using “View Schema” button • Real-time views of Junction table recording all Pilot-Flight relationships
Summary Statistics	<ul style="list-style-type: none"> • Flexibility to the user to input various combination of search criteria, mirroring what one would expect to find in a real-life flight search engine. • Smart algorithm that automatically ignores a search field if the user leaves the input box blank. • Some input boxes are locked for editing until their ‘parent’ inputs are made, to prevent the user from making invalid inputs. • Several of the search fields involve subqueries running on the back-end for pulling information in the relevant tables. • Listboxes and Comboboxes for user-friendliness and input validity check. • Instructions and examples of valid expressions shown next to all input boxes for user friendliness.
Insert	<ul style="list-style-type: none"> • Instructions and examples of valid expressions shown next to all input boxes for user friendliness. • Drop-down lists and List boxes built in for data validation. • All field values are checked and validated against data requirements, before allowing insertion. • Colour-coded highlights for special fields, i.e., Primary key, relationships to be recorded in junction table (for tables Flights and Pilots). • Autofill airport names for the Destination table, to avoid typos in long string inputs.
Search and View	<ul style="list-style-type: none"> • Allows the user to input virtually any combination of filter conditions, with the choice to switch between field-wise conditions (option to join with OR or AND), and customised condition (where the user can input any complex SQL WHERE statement, including subqueries). • Table view presented in table form using the TreeView widget in Tkinter. • The title in table view window shows how many records are found. • Allow user to search Flight table and Pilot table using foreign keys (e.g., search flights based on which pilots are associated with the flight), where the code performs a subquery using the pilot_flight_junc table.
Update	<ul style="list-style-type: none"> • Autofills all field values for the retrieved record (if exists) by clicking “Autofill” button, upon which the user can make edits to update. If record doesn’t exist shows error message. • Instructions and examples of valid expressions shown next to all input boxes for user friendliness. • Drop-down lists and List boxes built in for data validation. • All field values are checked and validated against data requirements, before allowing insertion. • Colour-coded highlights for special fields, i.e., Primary key, relationships to be recorded in junction table (for tables Flights and Pilots).
Delete	<ul style="list-style-type: none"> • Allows the user to input virtually any combination of filter conditions, with the choice to switch between field-wise conditions (option to join with OR or AND), and customised conditions (where the user can input any complex SQL WHERE statement, including subqueries).

	<ul style="list-style-type: none"> • After the user selects delete, a table view presented in table form using the TreeView widget in Tkinter, to allow user to view what items are about to be deleted. • Allow user to confirm delete before deleting, in case of accidental deletes. • The title in table view window shows how many records are found. • Allow user to search Flight table and Pilot table using foreign keys (e.g., search flights based on which pilots are associated with the flight), where the code performs a subquery using the <i>pilot_flight_junc</i> table.
--	---

Resources used:

1. **Course on Tkinter Python module:** [100 days of Python by Angela Yu](#), Sections “Day 27” and “Day 28”
2. **YouTube tutorial on SQLite3:** [SQLite Databases with Python – Full Course, by FreeCodeCamp.Org](#)
3. **YouTube tutorial on** [Working with Dates in SQLite with Python by Sean MacKenzie](#).
4. **Values generated by ChatGPT:** Aircraft models, Aircraft manufacturers, Aircraft type, Python list of country names and nationalities, Python list of 16 timezones.

Note on code structure and efficiency:

Since object-oriented programming is not yet covered in the course, this programme is built to the best of the candidate’s procedural programming abilities. This inevitably results in horrendously long script as shown. Plans are made to restructure the code into OOP modules in the near future.