Querying and Updating an Airline Database

As a simple example of using database tables, in this exercise you will run queries and updates on some small tables from an 'airline' database. The tables describe aircraft, cities, countries and specific flights. To do this exercise you will need to have installed a browser and editor for SQLite databases. See the instructions on Blackboard for installing either the *DB Browser for SQLite* standalone application or the *SQLite Manager* add-on for Firefox.

A. Importing the airline tables

Accompanying these instructions you will find an SQLite script `airline.sql`. Create a database file `airline.db` and populate it by running the script. In the *DB Browser for SQLite* application this can be done in two separate steps, but an easier way is to choose the 'Import database from SQL file' option and follow the prompts. This script will create four tables called `aicraft`, `cities`, `countries` and `flights` in your database file.

(There are minor differences in different versions of the SQLite language. Just in case you have any problems running the SQLite script provided we have also supplied a copy of the `airline.db` file, but it would be better for you to run the script yourself to see what it contains.)

B. Browsing the airline tables

Once you have successfully imported the database you will be able to browse the contents of the individual tables. In the *DB Browser for SQLite* application this is done easily via the 'Browse Data' tab. Examine each of the tables to familiarise yourself with their contents.

The airline's tables as provided contain the data shown below.

Cities:

CityCode	CityName	CountryCode
ADL	Adelaide	AUS
AKL	Auckland	NZL
BNE	Brisbane	AUS
CBR	Canberra	AUS
CGK	Jakarta	IND
HNL	Honolulu	USA
LAX	Los Angeles	USA
MEL	Melbourne	AUS
SFO	San Francisco	USA
SYD	Sydney	AUS
BRI	Brisbane	USA

Countries:

CountryCode	CountryName
AUS	Australia
IND	Indonesia
MLY	Malaysia
NZL	New Zealand
USA	United States of America

Aircraft:

AircraftType	AircraftDescription	SeatingCapacity
AB3	Airbus A300	250
D10	McDonnel Douglas DC10	150
727	Boeing 727	150
737	Boeing 737	120
74L	Boeing 747SP	260
743	Boeing 747-338	420
744	Boeing 747-438	420
757	Boeing 757	150
767	Boeing 767	260

Flights:

FlightNum	FromCityCode	ToCityCode	SeatsRemaining	AircraftType
1	BNE	SYD	10	AB3
2	SYD	CBR	20	727
3	SYD	MEL	30	757
4	SYD	AKL	40	D10
5	BNE	CGK	50	757
6	BNE	LAX	60	74L
7	SYD	HNL	70	767
8	HNL	SFO	80	767
9	SYD	LAX	90	744
10	SYD	BNE	100	AB3

C. <u>Querying the airline tables</u>

Next you should write some `select` queries to retrieve specific data from the database. In the *DB Browser for SQLite* application this is done via the 'Execute SQL' tab. It gives you the ability to edit and execute an SQLite script to either query or update the database. You can also import and execute SQLite scripts from files.

Answer the following questions by writing and executing appropriate **`select`** queries to display the required information.

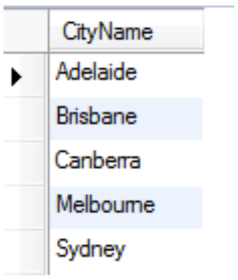
Question 1: How many rows does the following query return?

```
SELECT * FROM cities
```

Question 2: How many rows does the following query return?

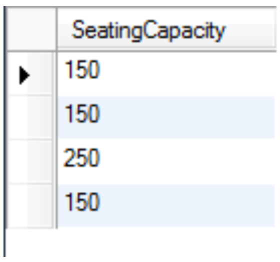
```
SELECT * FROM cities
WHERE CountryCode = 'AUS'
```

Question 3: Write a query that displays the name of all Australian cities in the `cities` table as shown below:



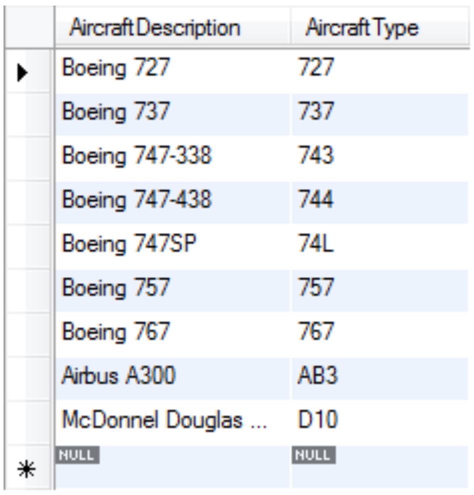
CityName
Adelaide
Brisbane
Canberra
Melbourne
Sydney

Question 4: Write a query that displays the seating capacity for all aircraft with at least 150 seats, but not more than 250 seats. Your query should produce the following output:



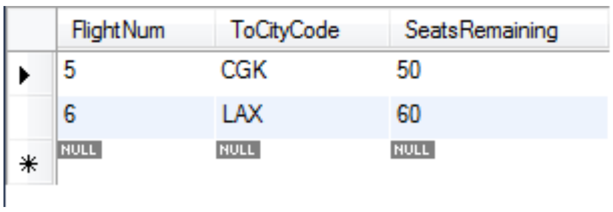
SeatingCapacity
150
150
250
150

Question 5: Write a query that displays just the aircraft description and aircraft type columns for all aircraft. Your query should produce the following output:



AircraftDescription	AircraftType
Boeing 727	727
Boeing 737	737
Boeing 747-338	743
Boeing 747-438	744
Boeing 747SP	74L
Boeing 757	757
Boeing 767	767
Airbus A300	AB3
McDonnel Douglas ...	D10
NULL	NULL

Question 6: Write a query that displays the flight number, destination city, and the number of remaining seats of all the flights that depart from Brisbane and have more than 20 seats left. Your query should produce the following output:



FlightNum	ToCityCode	SeatsRemaining
5	CGK	50
6	LAX	60
NULL	NULL	NULL

D. Updating the airline tables manually

The database interface also gives you the ability to directly edit the tables. In the *DB Browser for SQLite* application this is done via the 'Browse Data' tab. Existing cells can be edited just by clicking on them and new rows can be created by pressing the 'New Record' button. Use these features to make the following changes to the airline tables.

7. Reduce the number of seats remaining on Flight 10 from Sydney to Brisbane to 25.
8. Change Flight 3's destination from Melbourne to Adelaide.
9. Add the following information to the `countries` table: The country code for Canada is 'CA' and the code for Spain is 'ES'.
10. None of our airline's flights involve a Boeing 747-338. Assuming the airline no longer uses this type of aircraft, delete it from the `aircraft` table entirely.

In the *DB Browser for SQLite* application note that none of the changes you have made will be stored in the database file itself until you press the 'Write Changes' button. Until you do this you have the ability to undo all your changes by pressing 'Revert Changes'.

E. Updating the airline tables using a script

Although it's easy to manipulate the database's tables manually via the *DB Browser for SQLite* application's user interface, we also need to know how to do so via a written script if we are to automate the process. In this case you must make the following changes to the tables by editing and running appropriate SQLite statements in the 'Execute SQL' tab. After running your script, check that the tables were updated correctly by inspecting them in the 'Browse Data' tab. Note that if you make a mistake and corrupt one of the tables you can undo the change by pressing the 'Revert Changes' button (provided you haven't already committed the changes by pressing the 'Write Changes' button).

11. The largest passenger airliner in the world today is the Airbus A380. Assuming our airline has acquired one, write a script to add this plane into the `aircraft` table. When configured for economy class passengers only, the A380 has a massive seating capacity of 853.
12. A group of ten people have booked seats to travel from Brisbane to Canberra via Sydney. Write a script to update the number of seats remaining in the `flights` table accordingly.
13. Due to a massive cyclone off the California coast, all flights to Los Angeles have been cancelled! Write a single SQLite statement to delete all such flights from the `flights` table.

F. Exporting the airline tables

Finally, having modified your database, you can use the *DB Browser for SQLite* application to export the results into a text file. This will produce an SQLite script, like the one you imported to start this exercise, that you can use to recreate the tables later. (Obviously your tables will persist in the database itself, but this allows you to make a backup of your tables or to copy them to a different computer.) You can do this for either your whole database of tables or only a selected table. Finish the exercise by saving your tables via the 'Export database to SQL file' menu item, and inspect the resulting file's contents in a text editor to confirm that it is a human-readable SQLite script.