# CS4400: Database Systems
# Homework 2
# SQL queries

Due February 2, 2017, 11:59pm

Instructions: Upload your submission on ICON under Assignments > Homework 2.

You must submit **two plain text files** (`create-tables.sql` and `hw2-queries.sql`) containing your answers. They **must** execute correctly when you run the following commands on the command line (assuming hw2.db does not exist yet).

```
sqlite3 –init create-tables.sql hw2.db
sqlite3 –init hw2-queries.sql hw2.db
```

**or** when you run the following commands in the sqlite shell

```
.read create-tables.sql
.read hw2-queries.sql
```

## Goals for this assignment

- Create a database of multiple tables and load data into it
- Translate a query from an English description to a SQL query involving filters, projections, joins, and aggregates

In this homework, you will write several SQL queries on relational database of airline flights (it is ***not*** the same airline data from class). The data is from the Bureau of Transportation Statistics http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time. The database consists of four tables:

**Flights** (year, month_id, day_of_month, day_of_week_id, carrier_id, flight_num, origin_city, origin_state, dest_city, dest_state, departure_delay, taxi_out, arrival_delay, cancelled, actual_time, distance)
**Carriers** (cid, name)
**Months** (mid, month)
**Weekdays** (did, day_of_week)

You will decide on the type for each field when you create the tables.

You must impose the following constraints on your database:

- The primary key of Flights is (year, month_id, day_of_month, carrier_id, flight_num, origin_city)
- The primary keys of the other tables are cid, mid, did, respectively.
- Flights.carrier_id references Carrier.cid
- Flights.month_id references Months.mid
- Flights.day_of_week_id references Weekdays.did

We provide the flights database as a set of plain text files in Flights dataset (.tar.gz archive) http://homepage.cs.uiowa.edu/~bdmyers/cs4400_sp17/public/flights-small.tar.gz . Extract the contents with the command `tar xzvf flights-small.tar.gz`. Each file in the archive contains all the rows for the named table, one row per line.

In this homework, you need to do two things: (1) import the flights dataset into SQLite and (2) run SQL queries to answer a set of questions about the data.

## 1. Importing the flights database (5 of the 20 points)

To make sure the database you will create is persistent, run sqlite3 with a new database file as the argument. For example, `sqlite3 hw2.db`. Then you can use the `create table` statement to create the tables, choosing appropriate types for each column and specifying all key constraints as described above.

```
create table table_name ( … );
```

Recall that SQLite does not enforce foreign keys by default. To enable foreign keys use the following command. This checking is useful for making sure you imported the tables correctly. The command will have no effect if your version of SQLite was not compiled with foreign keys enabled.

```
PRAGMA foreign_keys=ON;
```

Then, you can use the SQLite .import command to read data from each text file into its table:

```
.import filename table_name
```

See examples of `.import` statements in the airline example used in class, SQLite documentation, or the `.help` command.

Put all the code for part A (four `create table` statements and four `.import` statements) into a file called `create-tables.sql`.

Tip: keep in mind that creating the database takes awhile (up to a minute), so once you get the table creation right, keep your database in the database file for use in part 2.

## 2. SQL queries (15 of the 20 points)

For each question below, write a single SQL query to answer that question. Put your queries in a file called `hw2-queries.sql`. Add a comment to each query indicating the question it answers and the number of rows in the query result. So that you have some indication that your answer is right, we have provided the expected size of the result for each query.

**Important:**

- The predicates in your queries should correspond to the English descriptions. For example, if a question asks you to find flights by Alaska Airlines Inc., the query should include a predicate that checks for that specific name as opposed to checking for the matching `carrier_id`. Same for predicates over months, weekdays, etc.
- You should be able to answer all the questions below with SQL queries that do NOT contain any subqueries!
- If a query uses a `group by` clause, make sure that all attributes in your `select` clause for that query are either grouping keys or aggregate values. SQLite will let you select other attributes, but that is wrong (recall in class we discussed how SQLite will just pick a row to display in this case). Other database systems would reject such queries because their meaning is poorly specified.

1. List the distinct flight numbers of all flights from Cedar Rapids to Chicago by SkyWest Airlines Inc. on Mondays. Also notice that, in the database, the city names include the state. So Cedar Rapids appears as Cedar Rapids/Iowa City IA. *[output relation cardinality: 7 rows]*
2. Find all itineraries from Los Angeles to Boston on July 15th, 2015. Search only for itineraries that have exactly one stop. Both legs of the flight must have occurred in the same day and must be with the same carrier. The total flight time (actual_time) of the entire itinerary should be less than 6 hours. For each itinerary, the query should return the name of the carrier, the first flight number, the origin, the first destination, the flight time, the second flight number, the second flight origin, the second destination, the second flight time, and the total flight time. *[output relation cardinality: 117]*
3. Find the day of the week with the longest average arrival delay. Return the name of the day and the average delay. *[output relation cardinality: 1 row]*
4. Find the names of all airlines that ever flew more than 1000 flights in a single day. Return only the names and no duplicates. *[output relation cardinality: 11 rows]*
5. Find all airlines that had more than 0.5% of their flights out of Cedar Rapids cancelled. Return the name of the airline, the percentage of cancelled flights out

of Cedar Rapids. The results should be listed from highest cancellation rate to lowest. *[output relation cardinality: 4]*

Put the code for all 5 questions in order in a file named `hw2-queries.sql`

## Bonus (optional, 1 extra credit point)

Come up with a non-trivial query that would be useful to a booking agent or an airline industry analyst. Keep in mind that an excessively large output size would be difficult for a human to understand unless it is able to be visualized as a plot or otherwise. Write the query in SQL along with a comment that describes the query in English. You may use subqueries and quantifiers if you wish, but it is not required.

submit your query in an additional file named `hw2-bonus.sql`

## Acknowledgements

derived from UW CSE344 fall 2015