!pip install ipython-sql
conda install -c conda-forge ipython-sql

# CS150A Homework1 - Coding

## Instructions / Notes:

***Read these carefully***

- You will need to install the `ipython-sql` module to run the scripts. (eg. `pip install --user ipython-sql`)
- Run the cell below to load the database `hw1.db`.
- You **may** create new Jupyter notebook cells to use for e.g. testing, debugging, exploring, etc.- this is encouraged in fact!
- When you see `In [*]:` to the left of the cell you are executing, this means that the code / query is *running*.
  - **If the cell is hanging- i.e. running for too long: to restart the SQL connection, you must restart the entire python kernel**
  - To restart the kernel using the menu bar: "Kernel >> Restart & Clear Output"), then re-execute the sql connection cell at the top
  - You will also need to restart the connection if you want to load a different version of the database file
- Remember:
  - `%sql [SQL]` is for *single line* SQL queries
  - `%%sql [SQL]` is for *multi line* SQL queries
- We have provided correct output from our solution in `correct_output.txt`.
  - **Your** `submit.py` **should match this output exactly.** This means:
    - the columns should have the **same names** as `correct_output.txt`
    - the columns should be in the **same order** as `correct_output.txt`

## Submission Instructions:

- Do *NOT* submit your iPython notebook directly.
- Instead, upload your answers in PDF version of the HW1.ipynb with your outputs to Gradescope.

**If you hava any confusion, please ask TA team in Piazza.**

*Have fun!*

In [206]:

```
%load_ext sql
%sql sqlite:///hw1.db
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

Out[206]:

'Connected: @hw1.db'

# Part 1: Travel Delays (75 points)

**In this part, you should answer query 1 to 8.**

There's nothing I dislike more than travel delays -- how about you?

In fact, I'm always scheming new ways to avoid travel delays, and I just found an amazing dataset that will help me understand some of the causes and trade-offs when traveling.

I wonder if you can use SQL to help me!

---

Not surprisingly... you can!

In this part, we'll use SQL to explore airline travel delays that occurred in July 2017.

To start, let's look at the primary relation in the database we've prepared for you:

In [207]:

```
%%sql
SELECT *
FROM flight_delays
LIMIT 1;
```

```
 * sqlite:///hw1.db
Done.
```

Out[207]:

| year | quarter | month | day_of_month | day_of_week | fl_date | unique_carrier | airline_id | carrier | t |
|---|---|---|---|---|---|---|---|---|---|
| 2017 | 3 | 7 | 1 | 6 | 2017-07-01 | AS | 19930 | AS | |

Cool, there are so many columns! How many rows are there?

In [208]:

```
%%sql
SELECT COUNT(*) AS num_rows
FROM flight_delays
```

```
 * sqlite:///hw1.db
Done.
```

Out[208]:

| num_rows |
|---|
| 509070 |

Wow, that's a lot of data! Good thing you don't have to answer all of my questions by hand...

You don't need to import more data into the database. However, you can find a description of each field online at https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236 (https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236).

We've pre-loaded a number of additional tables that will help you decode important fields like `airline_id`, `airport_id`, and `day_of_week` into human-readable form for the queries below.

Please use the following cell to explore these the `airlines` and `weekdays` tables:

In [209]:

```
%%sql

```

```
UsageError: %%sql is a cell magic, but the cell body is empty. Did you mean the line
magic %sql (single %)?
```

Alright -- let's get started!

# Query 1: How long are flights delayed on average? (5 points)

Just to get a sense of the data, let's start with a simple query.

In the cell below, write a SQL query that returns the average arrival delay for the entire month of July 2017 (i.e., the whole dataset).

In [210]:

```
%%sql
SELECT AVG(f.arr_delay) AS avg_delay
FROM flight_delays AS f
```

```
 * sqlite:///hw1.db
Done.
```

Out[210]:

| avg_delay |
|---|
| 8.295147670495197 |

# Query 2: What was the worst flight delay? (6 points)

Hmm, the average doesn't look too bad! What about the *worst* delay?

In the cell below, write a SQL query that returns the maximum arrival delay for the entire month of July 2017 (i.e., the whole dataset).

In [211]:

```
%%sql
SELECT MAX(ABS(f.arr_delay)) AS max_delay
FROM flight_delays AS f
```

```
 * sqlite:///hw1.db
Done.
```

Out[211]:

| max_delay |
|---|
| 1895.0 |

# Query 3: What flight am I happiest I didn't take? (8 points)

Yikes! What flight was so late?

In the cell below, write a SQL query that returns the carrier (i.e., `carrier`), flight number, origin city name, arrival city name, and flight date for the flight with the maximum arrival delay for the entire month of July 2017. Do not hard-code the arrival delay you found above. Hint: use a subquery.

In [212]:

```
%%sql
SELECT f.carrier, f.fl_num, f.origin_city_name, f.dest_city_name, f.fl_date
FROM flight_delays AS f
WHERE ABS(f.arr_delay) = (SELECT MAX(ABS(f.arr_delay)) FROM flight_delays AS f)
-- max() should be subquery here
```

```
 * sqlite:///hw1.db
Done.
```

Out[212]:

| carrier | fl_num | origin_city_name | dest_city_name | fl_date |
|---|---|---|---|---|
| AA | 58 | Kona, HI | Los Angeles, CA | 2017-07-26 |

# Query 4: Which are the worst days to travel? (10 points)

Since CS145 just started, I don't have time to head to Kona anytime soon. However, I'm headed out of town for a trip next week! What day is worst for booking my flight?

In the cell below, write a SQL query that returns the average arrival delay time for each day of the week, in descending order. The schema of your relation should be of the form ( `weekday_name` , `average_delay` ).

**Note: do *not* report the weekday ID.** (Hint: look at the `weekdays` table and perform a join to obtain the weekday name.)

In [213]:

```
%%sql
SELECT *
FROM weekdays
LIMIT 2
-- sample of weekdays
```

```
 * sqlite:///hw1.db
Done.
```

Out[213]:

| weekday_id | weekday_name |
|---|---|
| 1 | Monday |
| 2 | Tuesday |

q4 answer

In [214]:

```
%%sql
SELECT w.weekday_name, average_delay
FROM weekdays AS w,
(SELECT f.day_of_week AS day_id, AVG(f.arr_delay) AS average_delay
FROM flight_delays AS f
GROUP BY f.day_of_week)
WHERE w.weekday_id = day_id
ORDER BY average_delay DESC
```

```
 * sqlite:///hw1.db
Done.
```

Out[214]:

| weekday_name | average_delay |
|---|---|
| Friday | 14.452012705575017 |
| Monday | 10.537501524948151 |
| Thursday | 8.47985564692658 |
| Wednesday | 8.4561902339015 |
| Saturday | 7.544554592337578 |
| Tuesday | 4.6315245398299725 |
| Sunday | 4.211659780807007 |

# Query 5: Which airlines that fly out of SFO are delayed least? (10 points)

Now that I know which days to avoid, I'm curious which airline I should fly out of SFO. Since I haven't been told where I'm flying, please just compute the average for the airlines that fly from SFO.

In the cell below, write a SQL query that returns the average arrival delay time (across *all* flights) for each carrier that flew out of SFO at least once in July 2017 (i.e., in the current dataset), in descending order.

**Note: do *not* report the airlines ID.** (Hint: a subquery is helpful here; also, look at the `airlines` table and perform a join.)

In [215]:

```
%%sql
SELECT *
FROM airlines
LIMIT 2
-- sample of weekdays
```

```
 * sqlite:///hw1.db
Done.
```

Out[215]:

| airline_id | airline_name |
|---|---|
| 19031 | Mackey International Inc.: MAC |
| 19032 | Munz Northern Airlines Inc.: XY |

In [216]:

```
%%sql
SELECT f.airline_id AS aid, AVG(f.arr_delay) AS average_delay
FROM flight_delays AS f
WHERE f.origin = 'SFO'
GROUP BY f.airline_id
```

```
 * sqlite:///hw1.db
Done.
```

Out[216]:

| aid | average_delay |
|---|---|
| 19393 | 13.792032410533423 |
| 19690 | -3.129032258064516 |
| 19790 | 1.7166535122336228 |
| 19805 | 12.36739864864865 |
| 19930 | 8.09014675052411 |
| 19977 | 6.870443073471677 |
| 20304 | 10.977406375735066 |
| 20409 | 14.575539568345324 |
| 20436 | 19.879781420765028 |
| 21171 | 11.772509323388386 |

q5 answer
tip: arr_delay (including not SFO origined) of all airlines fly out of SFO once, not the arr_delay of their flights from SFO

In [217]:

```
%%sql
SELECT a.airline_name, average_delay
FROM airlines AS a,
-- aid-arr_delay
(SELECT aid, AVG(f.arr_delay) AS average_delay
FROM
-- SFO_origin_airline_id
(SELECT f.airline_id AS aid
FROM flight_delays AS f
WHERE f.origin = 'SFO'
GROUP BY f.airline_id),
flight_delays AS f
WHERE aid = f.airline_id
GROUP BY aid
)
WHERE a.airline_id = aid
ORDER BY average_delay DESC
```

```
 * sqlite:///hw1.db
Done.
```

Out[217]:

| airline_name | average_delay |
|---|---|
| JetBlue Airways: B6 | 21.459498403302437 |
| American Airlines Inc.: AA | 11.186316345868732 |
| Frontier Airlines Inc.: F9 | 10.106046316671906 |
| SkyWest Airlines Inc.: OO | 9.099801905480614 |
| Southwest Airlines Co.: WN | 8.777705490592203 |
| United Air Lines Inc.: UA | 6.730333043511133 |
| Virgin America: VX | 6.066317340756509 |
| Delta Air Lines Inc.: DL | 2.783035510698813 |
| Hawaiian Airlines Inc.: HA | -0.2698805109188298 |
| Alaska Airlines Inc.: AS | -1.1098173650924996 |

# Query 6: What proportion of airlines are regularly late? (12 points)

Yeesh, there are a lot of late flights! How many airlines are regularly late?

In the cell below, write a SQL query that returns the proportion of airlines (appearing in `flight_delays`) whose flights are on average at least 10 minutes late to arrive. Do not hard-code the total number of airlines, and make sure to use at least one `HAVING` clause in your SQL query.

Note: sqlite `COUNT(*)` returns integer types. Therefore, your query should likely contain at least one `SELECT CAST (COUNT(*) AS float)` or a clause like `COUNT(*)*1.0`.

In [218]:

```
%%sql
SELECT f.airline_id,COUNT(*)
FROM flight_delays AS f
GROUP BY f.airline_id
HAVING AVG(f.arr_delay) >= 10
```

```
 * sqlite:///hw1.db
Done.
```

Out[218]:

| airline_id | COUNT(*) |
|---|---|
| 19805 | 79403 |
| 20366 | 29231 |
| 20409 | 26436 |
| 20436 | 9611 |

(not q6 answer, just a note) proportion is type count not times count

In [219]:

```
%%sql
SELECT SUM(regular_delay_count)/count_all AS proportion
FROM
(SELECT f.airline_id,COUNT(*) AS regular_delay_count
FROM flight_delays AS f
GROUP BY f.airline_id
HAVING AVG(f.arr_delay) >= 10),
(SELECT COUNT(*)*1.0 AS count_all
FROM flight_delays AS f)
```

```
 * sqlite:///hw1.db
Done.
```

Out[219]:

| proportion |
|---|
| 0.2842064941952973 |

In [220]:

```
%%sql
SELECT COUNT(regular_delay_flight_count)*1.0 AS regular_delay_types
FROM (
SELECT f.airline_id AS regular_delay_flight_count
FROM flight_delays AS f
GROUP BY f.airline_id
HAVING AVG(f.arr_delay) >= 10)
```

```
 * sqlite:///hw1.db
Done.
```

Out[220]:

| regular_delay_types |
|---|
| 4.0 |

q6 answer
COUNT(*) for inner view has default DISTINCT
Remember to add it when not embedded view

In [221]:

```
%%sql
SELECT (COUNT(DISTINCT regular_delay_flight_count)*1.0)/(COUNT(DISTINCT airline_types)*1.0) AS propo
FROM
(SELECT f.airline_id AS regular_delay_flight_count
FROM flight_delays AS f
GROUP BY f.airline_id
HAVING AVG(f.arr_delay) >= 10)
,
(SELECT f.airline_id AS airline_types
FROM flight_delays AS f
GROUP BY airline_id)
```

```
 * sqlite:///hw1.db
Done.
```

Out[221]:

| proportion |
|---|
| 0.3333333333333333 |

# Query 7: How do late departures affect late arrivals? (12 points)

It sure looks like my plane is likely to be delayed. I'd like to know: if my plane is delayed in taking off, how will it affect my arrival time?

The sample covariance (https://en.wikipedia.org/wiki/Covariance) provides a measure of the joint variability of two variables. The higher the covariance, the more the two variables behave similarly, and negative covariance indicates the variables indicate the variables tend to be inversely related. We can compute the sample covariance as:

$$Cov(X,Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{x})(y_i - \hat{y})$$

where $x_i$ denotes the $i$th sample of $X$, $y_i$ the $i$th sample of $Y$, and the mean of $X$ and $Y$ are denoted by $\bar{x}$ and $\bar{y}$.

In the cell below, write a single SQL query that computes the covariance between the departure delay time and the arrival delay time.

*Note: we could also compute a statistic like the Pearson correlation coefficient (https://en.wikipedia.org/wiki/Pearson_correlation_coefficient) here, which provides a normalized measure (i.e., on a scale from -1 to 1) of how strongly two variables are related. However, sqlite doesn't natively support square roots (unlike commonly-used relational databases like PostgreSQL and MySQL!), so we're asking you to compute covariance instead.*

q7 answer

In [222]:

```
%%sql
SELECT SUM((f.dep_delay-avg_dept)*(f.arr_delay-avg_arr))/num_rows AS cov
FROM flight_delays AS f,
(SELECT AVG(f.dep_delay) AS avg_dept, AVG(f.arr_delay) AS avg_arr
FROM flight_delays AS f),
(SELECT COUNT(*) AS num_rows
FROM flight_delays)
```

```
 * sqlite:///hw1.db
Done.
```

Out[222]:

| cov |
|---|
| 2367.266118774892 |

# Query 8: It was a bad week... (12 points)

Which airlines had the largest absolute increase in average arrival delay in the last week of July (i.e., flights on or after July 24th) compared to the previous days (i.e. flights before July 24th)?

In the cell below, write a single SQL query that returns the airline name (*not* ID) with the maximum absolute increase in average arrival delay between the first 23 days of the month and days 24-31. Report both the airline name and the absolute increase.

Note: due to sqlite's handling of dates (http://www.sqlite.org/lang_datefunc.html), it may be easier to query using `day_of_month`.

Note 2: This is probably the hardest query of the assignment; break it down into subqueries that you can run one-by-one and build up your answer subquery by subquery.

Hint: You can compute two subqueries, one to compute the average arrival delay for flights on or after July 24th, and one to compute the average arrival delay for flights before July 24th, and then join the two to calculate the increase in delay.

q8 answer
tip: absolute increase

In [223]:

```
%%sql
SELECT a.airline_name, avg_delay_afon_24-avg_delay_bf_24 AS delay_increase
FROM airlines AS a,
(SELECT airline_id AS aid1, AVG(f.arr_delay) AS avg_delay_bf_24
FROM flight_delays AS f
WHERE f.day_of_month < 24
GROUP BY f.airline_id),
(SELECT airline_id AS aid2, AVG(f.arr_delay) AS avg_delay_afon_24
FROM flight_delays AS f
WHERE f.day_of_month >= 24
GROUP BY f.airline_id)
WHERE aid1 = aid2 AND a.airline_id = aid1
ORDER BY delay_increase DESC
LIMIT 1
```

```
 * sqlite:///hw1.db
Done.
```

Out[223]:

| airline_name | delay_increase |
|---|---|
| Southwest Airlines Co.: WN | 1.2365259414370655 |

# Part 2: Uber Orders (25 points)

**In this part, you should answer query 9.**

To help Uber improve the performance of their taxi-hailing APP, you are required to calculate the cancellation rate of orders.

Let's start with the following table `Orders` and `Users`.

In [224]:

```
%%sql
SELECT *
FROM Orders
Limit 1;
```

```
 * sqlite:///hw1.db
Done.
```

Out[224]:

| id | client_id | driver_id | city_id | status | day |
|---|---|---|---|---|---|
| 1 | 1 | 10 | 1 | completed | 2022-10-01 |

In [225]:

```
%%sql
SELECT o.id, o.day
FROM Orders AS o
WHERE o.day = '2022-10-01' OR '2022-10-02' OR '2022-10-03'
```

```
 * sqlite:///hw1.db
Done.
```

Out[225]:

| id | day |
|---|---|
| 1 | 2022-10-01 |
| 2 | 2022-10-01 |
| 3 | 2022-10-01 |
| 4 | 2022-10-01 |
| 5 | 2022-10-02 |
| 6 | 2022-10-02 |
| 7 | 2022-10-02 |
| 8 | 2022-10-03 |
| 9 | 2022-10-03 |
| 10 | 2022-10-03 |

tip: check the schema, day is `date` type here

In [226]:

```
%%sql
PRAGMA table_info(Orders);
```

```
 * sqlite:///hw1.db
Done.
```

Out[226]:

| cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|
| 0 | id | INT | 1 | None | 1 |
| 1 | client_id | INT | 1 | None | 0 |
| 2 | driver_id | INT | 1 | None | 0 |
| 3 | city_id | INT | 1 | None | 0 |
| 4 | status | ENUM | 0 | None | 0 |
| 5 | day | date | 0 | None | 0 |

In `Orders`:

The table holds all Uber taxi orders, `id` is the primary key for this table. Each order has a unique `id`, while `client_id` and `driver_id` are foreign keys to the `users_id` at the `Users` table. `status` is an ENUM type of ('completed', 'cancelled_by_driver', 'cancelled_by_client').

An ENUM is a string object with a value chosen from a list of permitted values that are enumerated explicitly in the column specification at table creation time.

In [227]:

```
%%sql
SELECT *
FROM Users
Limit 1;
```

```
 * sqlite:///hw1.db
Done.
```

Out[227]:

| users_id | banned | role |
|---|---|---|
| 1 | No | client |

schema: data type

In [228]:

```
%%sql
PRAGMA table_info(Users);
```

```
 * sqlite:///hw1.db
Done.
```

Out[228]:

| cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|
| 0 | users_id | INT | 1 | None | 1 |
| 1 | banned | ENUM | 0 | None | 0 |
| 2 | role | ENUM | 0 | None | 0 |

In `Users`:

The table holds all users, `users_id` is the primary key for this table. Each user has a unique `users_id`, and `role` is an ENUM type of ('client', 'driver', 'partner'). `banned` is an ENUM type of ('Yes', 'No').

# Query 9: What is the cancellation rate? (25 points)

The definition of Cancellation Rate as follows, where unbanned users means `banned` =='No':

**cancellation rate = the number of canceled (by client or driver) requests with unbanned users / the total number of requests with unbanned users on that day.**

In the cell below, write a single SQL query that returns the cancellation rate of requests with unbanned users (both client and driver must not be banned) each day between "2022-10-01" and "2022-10-03". Round Cancellation Rate to two decimal points.

Hint: you may refer the `CASE` expression.

Tip:

The CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    WHEN conditionN THEN resultN
    ELSE result
END;
```

Tips:
sql opt order
1 ~（位非）
2 *（乘）、/（除）、%（取模）
3 +（正）、-（负）、+（加）、+（串联）、-（减）、&（位与）、^（位异或）、|（位或）
4 =、><>=、<=、<>!=、!>!<（比较运算符）
5 NOT
6 AND
7 ALL、ANY、BETWEEN、IN、LIKE、OR、SOME
8 =（赋值）

In [229]:

```
%%sql
SELECT o.day AS day1,
CASE
    WHEN COUNT(*)*1.0 THEN COUNT(*)*1.0
    ELSE 0.0
END AS count1
FROM Orders AS o, Users AS u
WHERE (o.day = '2022-10-01' OR o.day = '2022-10-02' OR o.day = '2022-10-03')
AND o.status != 'completed'
AND o.client_id = u.users_id
AND u.banned = 'No'
GROUP BY o.day
```

```
 * sqlite:///hw1.db
Done.
```

Out[229]:

| day1 | count1 |
|---|---|
| 2022-10-01 | 1.0 |
| 2022-10-03 | 1.0 |

In [230]:

```
%%sql
SELECT  o.day, COUNT(*)
FROM Orders AS o, Users AS u
WHERE (o.day = '2022-10-01' OR o.day = '2022-10-02' OR o.day = '2022-10-03')
AND u.banned = 'No'
AND o.client_id = u.users_id
GROUP BY o.day
```

```
 * sqlite:///hw1.db
Done.
```

Out[230]:

| day | COUNT(*) |
|---|---|
| 2022-10-01 | 3 |
| 2022-10-02 | 2 |
| 2022-10-03 | 2 |

q9 answer
LEFT JOIN: preserve the left's not matched rows
CASE None is bool no need to EXIST()

In [231]:

```
%%sql
SELECT day2 AS Day,
CASE
    WHEN ROUND(count1/count2,2) THEN ROUND(count1/count2,2)
    ELSE 0.0
END
AS Cancellation_Rate
FROM
(SELECT  o.day AS day2, COUNT(*)*1.0 AS count2
FROM Orders AS o, Users AS u
WHERE (o.day = '2022-10-01' OR o.day = '2022-10-02' OR o.day = '2022-10-03')
AND u.banned = 'No'
AND o.client_id = u.users_id
GROUP BY o.day)
LEFT JOIN
(SELECT  o.day AS day1, COUNT(*)*1.0 AS count1
FROM Orders AS o, Users AS u
WHERE (o.day = '2022-10-01' OR o.day = '2022-10-02' OR o.day = '2022-10-03')
AND o.status != 'completed'
AND o.client_id = u.users_id
AND u.banned = 'No'
GROUP BY o.day)
ON day1 = day2
```

```
 * sqlite:///hw1.db
Done.
```

Out[231]:

| Day | Cancellation_Rate |
|---|---|
| 2022-10-01 | 0.33 |
| 2022-10-02 | 0.0 |
| 2022-10-03 | 0.5 |

# You're done! Now submit!

- Refer to the top of this notebook for submission instructions.