Lab #1

CSCI 3287

Modify this cell and put your Name and email

Name: Nicolas Mavromatis

Email: nima6629@colorado.edu

Instructions / Notes:

Read these carefully

- Download `flights.db.bz2` from Moodle and make sure it is in the same directory as this Jupyter notebook. You should uncompress it using the terminal command `bunzip2 flights.db.bz2` . The resulting `flights.db` file should be about 84MB.
- Run the cell below to load the database `flights.db` .
- You **may** create new Jupyter notebook cells to use for e.g. testing, debugging, exploring, etc.- this is encouraged in fact!
- However, you you must clearly mark the solution to each sub-problem by having your solution in the cell immediately after the cells marked `### BEGIN SOLUTION`
- Remember:
  - `%sql [SQL]` is for *single line* SQL queries
  - `%%sql`
    `[SQL]` is for *multi line* SQL queries

Submission Instructions:

- Do *NOT* submit your iPython notebook -- instead, you should print the notebook as a PDF document and submit that. To do that, use `File -> Export Notebook As.. -> HTML` , then open the HTML document and print it to a PDF file.

If you run into problems with a query taking a very very long time, first try `Kernel -> Restart All and Run All Cells..` and then ask on Piazza

Have fun!

In [1]:

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

Out[1]:

```
'Connected: @flights.db'
```

Introduction: Travel Delays

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 homework, we'll use SQL to explore airline travel delays that occurred in July 2007. To start, let's look at the primary relation in the database we've prepared for you:

In [2]:

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

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

Out[2]:

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	Actua
2007	7	1	7	2052	2050	2153	2155	WN	575	N304SW	

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

In [3]:

```
%%sql
SELECT COUNT(*) AS num_rows
FROM ontime;
```

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

Out[3]:

num_rows
648900

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. We actually downloaded the data from http://stat-computing.org/dataexpo/2009/supplemental-data.html and focused on just July 2007 to reduce the data size.

We've pre-loaded a number of additional tables that will help you decode important fields like

- `ontime` - the ontime flight data described at http://stat-computing.org/dataexpo/2009/sqlite.html
- `carriers` - airlines described at http://stat-computing.org/dataexpo/2009/supplemental-data.html
- `airports` - airports described at http://stat-computing.org/dataexpo/2009/supplemental-data.html
- `planes` - individual plane information described at http://stat-computing.org/dataexpo/2009/supplemental-data.html
- `weekdays` - hand-made database of weekdays with Sunday as day #1

Please use the following cell to explore these the `carriers`, `airports`, `planes` and `weekdays` tables. Try a few different select statements to see what each table contains.

In [4]:

```
%%sql
select * from ontime limit 2;
```

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

Out[4]:

Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	FlightNum	TailNum	Actua
2007	7	1	7	2052	2050	2153	2155	WN	575	N304SW	

2007	7	1	7	1309	1250	1408	1355	WN	692	N680AA

In [5]:

```
%%sql
select * from carriers limit 2;
```

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

Out[5]:

code	carrier
02Q	Titan Airways
04Q	Tradewind Aviation

In [6]:

```
%%sql
select * from weekdays limit 2;
```

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

Out[6]:

DayOfWeek	Name	Abbrev	Short
1	Sunday	Sun	Su
2	Monday	Mon	Mo

Alright -- let's get started!

Part I: SQL Queries

Query 1: How long are flights delayed on average? (10 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 2007 (i.e., the whole dataset).

In [7]:

```
### BEGIN SOLUTION
```

In [8]:

```
%%sql
select avg(arrDelay) from ontime where (month==7 and year==2007);
```

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

Out[8]:

avg(arrDelay)
14.107679837700504

Query 2: What was the worst flight delay? (10 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 2007 (i.e., the whole dataset).

In [9]:

```
### BEGIN SOLUTION
```

In [10]:

```
%%sql
select max(arrDelay) from ontime where (month==7 and year==2007);
```

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

Out[10]:

max(arrDelay)
1386

Query 3: What flight am I happiest I didn't take? (10 points)

Yikes! What flight was so late?

In the cell below, write a SQL query that returns the airline code (i.e., `UniqueCarrier`), origin city name, destination city name, flight number, and arrival delay for the flight(s) with the maximum arrival delay for the entire month of July 2007. Do not hard-code the arrival delay you found above. Hint: use a subquery.

Your query should have the following columns (using "...as.." to change attribute labels into table labels)

Airline Code	Origin	Destination	Flight Number	Arrival Delay
value...	value	value	value	value

If future problems, we would describe this as a query returning (`Airline Code` , `Origin` , `Destination` , `Flight Number` , `Arrival Delay`).

In [11]:

```
### BEGIN SOLUTION
```

In [12]:

```
%%sql
--Use arrDelay as matching key for nested select
SELECT uniqueCarrier as AirlineCode, origin as Origin, dest as Dest, flightnum as
FlightNumber, arrDelay as ArrivalDelay
from ONTIME where arrdelay in (Select max(arrdelay) from ontime where (month==7 and
year==2007));
```

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

Out[12]:

AirlineCode	Origin	Dest	FlightNumber	ArrivalDelay
AA	LAS	DFW	1004	1386

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

Since class just started, I don't have time to head to travel 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 (`Day of Week` , `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 [13]:

```
#SELECT column_name(s)
#FROM table1
#INNER
#JOIN table2
#ON table1.column_name = table2.column_name;

#NOTE: You can order by Alias defined!!

### BEGIN SOLUTION
```

In [14]:

```
%%sql
select weekdays.Name as DayofWeek, avg(Ontime.arrdelay) as AverageDelay
from onTime
INNER JOIN weekdays ON ontime.dayofweek == weekdays.dayofweek
group by weekdays.name order by AverageDelay DESC;
```

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

Out[14]:

DayofWeek	AverageDelay
Wednesday	18.104848250718305
Saturday	17.03286664090445
Sunday	15.889976224441275
Thursday	13.359122962962964
Monday	12.97324491855269
Tuesday	12.716138992293423
Friday	7.1869733939345615

Query 5: Which airlines that fly out of DEN are delayed least?

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

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 DEN at least once in July 2007 (i.e., in the current dataset), in descending order.

The schema of your relation should be of the form (`Airline Name` , `Average Delay`).

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

In [15]:

```
#NOTE: You can first do one nested select, followed by a second, where each has different
```

```
group by clauses!
### BEGIN SOLUTION
```

In [16]:

```
%%sql
--Use origin as matching key for nested query
--Finish with conditions that apply to first clause

select carriers.carrier as AirlineName, avg(Ontime.arrdelay) as AverageDelay
from onTime INNER JOIN carriers ON ontime.uniqueCarrier == carriers.code
where origin in
    (select origin from ontime where(origin=='DEN' and month==7 and year==2007) group by
uniqueCarrier)
group by AirlineName order by AverageDelay desc;
```

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

Out[16]:

AirlineName	AverageDelay
JetBlue Airways	30.157894736842106
Comair Inc.	26.0
Expressjet Airlines Inc.	23.3046875
American Airlines Inc.	19.608169440242058
United Air Lines Inc.	16.398711524695777
Northwest Airlines Inc.	16.311720698254366
Alaska Airlines Inc.	15.78139534883721
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)	13.11697247706422
Delta Air Lines Inc.	12.268361581920903
Continental Air Lines Inc.	12.004878048780487
Frontier Airlines Inc.	11.90063233965673
Skywest Airlines Inc.	10.33487618606804
Mesa Airlines Inc.	9.656940063091483
Southwest Airlines Co.	8.14156378600823
Atlantic Southeast Airlines	6.045454545454546
AirTran Airways Corporation	4.193548387096774

Query 6: What proportion of airlines are regularly late?

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 `ontime`) whose flights are on average at least 10 minutes late to arrive. For example, if 4 of 8 airlines have average arrival delays of at least 10 minutes, you would report 0.5

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 [17]:

```
#Note the nested select in ()
### BEGIN SOLUTION
```

In [18]:

```
%%sql
--Nested select!
--First select has no from...doesn't need it. Just used to set up (A)/(B)

select
    --count only applies to the nested selection
    (select cast(count(*) as float)
        from
        (
          select avg(arrDelay) as x from ontime group by uniqueCarrier having x>=10
        )
    )
/
(select count(distinct uniqueCarrier)*1.0 from ontime) as Proportion;
```

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

Out[18]:

Proportion
0.7

Query 7: How do late departures affect late arrivals?

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 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: $$ Cov(X,Y) = \frac{1}{n-1} \sum_{i=1}^n (x_i-\bar{x})(y_i-\bar{y}) $$ where x_i denotes the ith sample of X, y_i the ith 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. You should explicitly exclude entries where either the arrival or departure delay is NULL.

Note: we could also compute a statistic like the 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.

In [19]:

```
### BEGIN SOLUTION
```

In [20]:

```
%%sql
--AVG: Must use group by (unique attribute) or else returns 1 result
--I assume this covariance divides by N-1

select
```

```
(
    (
        select sum
            (

                (
                    select (depDelay-avg(depDelay))*(arrdelay-avg(arrdelay))
                    from ontime where ((depDelay is not NULL) and (arrdelay is not NULL))
                )
            )

        from ontime
    )
    * (select (1.0/(count(*)-1)) from ontime)

)as Covariance;
```

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

Out[20]:

Covariance
205.71852095880595

Query 8: It was a bad week...

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 just*** 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, 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.

In [21]:

```
### BEGIN SOLUTION
```

In [22]:

```
%%sql
--delete all tables if necessary
drop table L;
drop table R;
drop table fin;
```

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

Out[22]:

```
[]
```

In [23]:

```
%%sql
--Create multiple tables to make calcs easier
create table L
(uniqueCarrier varchar[50],
 left float,
 right float,
 diff float
);

create table R
(uniqueCarrier varchar[50],
 left float,
 right float,
 diff float
);

create table Fin
(uniqueCarrier varchar[50],
 left float,
 right float,
 diff float
);
```

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

Out[23]:

```
[]
```

In [24]:

```
%%sql
--Insert into L all averages of start of month. UniqueCarrier will be future key
INSERT INTO L(left, uniqueCarrier)
SELECT avg(arrDelay) as a, uniqueCarrier
FROM ONTIME
WHERE (month==7 and dayofMonth<24) group by uniqueCarrier;

--insert into R all averages of last week of month
INSERT INTO R(right, uniqueCarrier)
SELECT avg(arrDelay) as a, uniqueCarrier
FROM ONTIME
WHERE (month==7 and dayofMonth>=24) group by uniqueCarrier;

--Do a join on L and R into fin. HAPPENS first even though it is later in syntax.
--NOTES: Aliases are defined as L A, R B AFTER they are referenced. A-OK
--ALiases are needed to differentiate cols with same name
--from tbl1 inner join tbl2 acts like one big "from group" that so you don't need to list
from a,b with a comma
--Note B.right-A.left ez calculation. Happens after join so they are matched on
"uniqueCarrier"

insert into Fin select distinct A.uniqueCarrier, A.left, B.right, B.right-A.left from L A,
R B
INNER JOIN L on A.uniqueCarrier == B.uniqueCarrier;
```

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

Out[24]:

```
[]
```

In [25]:

```
%%sql
--Finally, just calculate max diff.
--Note: select "carrier" means use those values (names), after matching col attrs are
joined.
select carrier as AirlineName, max(Fin.diff) as MaxIncrease
from Fin
INNER JOIN carriers
ON Fin.uniqueCarrier == carriers.code limit 1;
```

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

Out[25]:

AirlineName	MaxIncrease
US Airways Inc. (Merged with America West 9/05. Reporting for both starting 10/07.)	8.125207088689457

Query 9: Of Hipsters and Technologists

I'm keen to visit both Portland (PDX) and San Francisco (SFO), but I can't fit both into the same trip. To maximize my frequent flier mileage, I'd like to use the same airline for each. Which airlines fly both DEN -> PDX and DEN -> SFO?

In the cell below, write a single SQL query that returns the distinct airline names (*not* ID, and with no duplicates) that flew both DEN -> PDX and DEN -> SFO in July 2007.

In [26]:

```
#Note it is easiest just to nest sometimes, matching keys from queries.
### BEGIN SOLUTION
```

In [27]:

```
%%sql

select distinct carrier as AirlineName from carriers INNER JOIN ontime ON
ontime.uniqueCarrier == carriers.code where uniqueCarrier in
(

    select uniqueCarrier from onTime where
        ((origin=='DEN' and dest=='PDX') and (month==7 and year==2007))
        and uniquecarrier in
             (select uniqueCarrier from onTime where ((origin=='DEN' and dest=='SFO')
                and (month==7 and year==2007)) group by uniqueCarrier)
        group by uniqueCarrier
);
```

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

Out[27]:

AirlineName
United Air Lines Inc.
Frontier Airlines Inc.

Query 10: Decision Fatigue and Equidistance

I'm flying back to Denver from LA later this month, and I can fly out of either LA (LAX), Ontario (ONT) or San Diego (SAN) and can fly into either Denver (DEN) or Colorado Spring (COS). If this month is like July, which flight will have the shortest arrival delay for flights leaving after 2PM local time?

In the cell below, write a single SQL query that returns the average arrival delay of flights departing either LAX, ONT or SAN after 2PM local time (`CrsDepTime`) and arriving at one of DEN or COS. Group by departure and arrival airport and return results descending by arrival delay.

Note: the `CrsDepTime` field is an integer formatted as hhmm (e.g. 4:15pm is 1615)

In [28]:

```
### BEGIN SOLUTION
```

In [29]:

```
%%sql
--Note: I included CRSDepTime even though it did not explicity say to.
select avg(ArrDelay) as Avg, origin, dest, CrsDepTime from onTime
where((origin=='LAX' or origin=='SAN' or origin=='ONT') and (dest=='DEN' or dest=='COS')
      and (crsdepTime>1400)  and arrDelay is not Null)
group by dest, origin order by avg Desc;
```

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

Out[29]:

Avg	Origin	Dest	CRSDepTime
25.548387096774192	SAN	COS	1510
23.285714285714285	ONT	DEN	1450
22.167192429022084	LAX	DEN	1415
18.78409090909091	SAN	DEN	1525
13.366666666666667	LAX	COS	2038
8.0	ONT	COS	1410

You're done! Now submit!

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