BBM473 Database Laboratory (Fall 2018)

Introduction to SQL

Before we actually get into basic SQL queries (**asking questions *of* data in tables**), we'll look at some of the basics about how to **create** tables.

NOTE: Make sure to have a copy of the database file, "dataset_1.db", from the last lecture downloaded and in this directory for the below to work!

```
In [ ]: %load_ext sql
        %sql sqlite:///dataset_1.db
```

Activity 2-1:

Schemas & table creation

Recall that the database we just loaded has one table, `precipitation_full`, having the following schema:

> - `state_code`
> - `station_id`
> - `year`
> - `month`
> - `day`
> - `hour`
> - `precipitation`
> - `flag_1`
> - `flag_2`

Each tuple in this table describes one hour of rainfall (`precipitation`- in hundredths of an inch) at one station (`station_id`) in one state (`state_code`). Note that tuples with `hour=25` record the total rainfall for that day, and that we can ignore the values of attributes `flag_1` and `flag_2` for now.

Now, however, let's see how to view the **schema** of existing tables on your own; there are several ways, including but not limited to:

- DESCRIBE tablename
- SHOW CREATE TABLE tablename
- SHOW COLUMNS tablename

Unfortunately, support for these varies widely between DBMSs, and is also limited by our IPython interface (for example sqlite, which we are using, does not support the above; it does have a `.schema tablename` command, however this doesn't work in IPython notebooks...)

One that does work for us here though is:

```
In [ ]: %sql PRAGMA table_info(precipitation_full);
```

A bit verbose, but gets the job done!

And, we can get the exact statement used to create the table as follows (**a great way to find guidance here!!**):

```
In [ ]: %sql SELECT sql FROM sqlite_master WHERE name = 'precipitation_full';
```

Without going into full detail (yet), the above table contains one record for each hour at each station, and contains the amount of precipitation that was measured during that hour.

Suppose that this lecture has been repurposed as a rain measurement corps to assist with the department that collected this data! Based on what we've covered so far, the above example, and the internet, create a table for storing the staff assignments. Table requirements:

- Everyone in the class will be holding a cup in the rain for a specific several-hour shift at a specific station; this assignment will remain the same every day
- Each person will have one off-day per week
- Each person's cup might be of a different size, measured as a float value
- The Dept. of Interior data servers can't handle the full dataset we would generate, and require a random subsample- so some people will be randomly chosen to stand in the rain without a cup. These assignments need to be recorded somehow in the table too.
- Some people in the class have Welsh names (https://www.youtube.com/watch?v=fHxO0UdpoxM)

Type your create table statement here:

NB: Remember to start with `%sql` for single line sql or `%%sql`

Execute the code below:

```
In [ ]: %%sql drop table if exists product;
        create table product(
               pname        varchar primary key, -- name of the product
               price        money,               -- price of the product
               category     varchar,             -- category
               manufacturer varchar NOT NULL     -- manufacturer
        );
        insert into product values('Gizmo', 19.99, 'Gadgets', 'GizmoWorks');
        insert into product values('PowerGizmo', 29.99, 'Gadgets', 'GizmoWorks');
        insert into product values('MultiTouch', 203.99, 'Household', 'Hitachi');
        insert into product values('SingleTouch', 149.99, 'Photography', 'Canon');
```

Activity 2-2:

Single table queries

Task #1

Try writing a query to get an output table of all the products with "Touch" in the name, showing just their name and price, and sorted alphabetically by manufacturer.

Let's look at the products first:

```
In [ ]: %sql select * from product;
```

Write your query here:

```
In [ ]: %sql Select pname, price from product where pname LIKE '%Touch%'
```

Next, write a query that returns the *distinct* names of manufacturers that make products with "Gizmo" in the name:

```
In [ ]: %sql Select distinct manufacturer from product where pname like '%Gizmo%'
```

Task #2:

Try some of these queries but first guess what they return.

```
In [ ]: %sql SELECT DISTINCT category FROM product ORDER BY category;
```

```
In [ ]: %sql SELECT category FROM product ORDER BY pname;
```

```
In [ ]: %sql SELECT DISTINCT category FROM product ORDER BY pname;
```

Execute the code below:

```
In [ ]: # Create tables & insert some random numbers
        # Note: in Postgresql, try the generate_series function...
        %sql DROP TABLE IF EXISTS R; DROP TABLE IF EXISTS S; DROP TABLE IF EXISTS T;
        %sql CREATE TABLE R (A int); CREATE TABLE S (A int); CREATE TABLE T (A int);
        for i in range(1,6):
            %sql INSERT INTO R VALUES (:i)
        for i in range(1,10,2):
            %sql INSERT INTO S VALUES (:i)
        for i in range(1,11,3):
            %sql INSERT INTO T VALUES (:i)
```

```
In [ ]: %%sql
        drop table if exists product; -- This needs to be dropped if exists, see why fu
        rther down!
        drop table if exists company;
        pragma foreign_keys = ON; -- WARNING by default off in sqlite
        create table company (
            cname varchar primary key, -- company name uniquely identifies the company.
            stockprice money, -- stock price is in money
            country varchar); -- country is just a string
        insert into company values ('ToyWorks', 25.0, 'USA');
        insert into company values ('ToyFriends', 65.0, 'China');
        insert into company values ('ToyCo', 15.0, 'China');

        create table product(
               pname varchar, -- name of the product
               price money, -- price of the product
               category varchar, -- category
               manufacturer varchar, -- manufacturer
               primary key (pname, manufacturer),
               foreign key (manufacturer) references company(cname));
        insert into product values('Pikachu', 19.99, 'Toy', 'ToyWorks');
        insert into product values('Pikachu', 19.99, 'Toy', 'ToyFriends');
        insert into product values('Pokeball', 29.99, 'Electronic', 'ToyCo');
        insert into product values('Bulbasaur', 149.99, 'Toy', 'ToyFriends');
        insert into product values('Charizard', 203.99, 'Toy', 'ToyCo');
        insert into product values('PokeCamera', 19.99, 'Electronic', 'ToyWorks');
```

Activity 2-3:

Multi-table queries

Task #1:

For three tables R, S, T that only have one attribute A:

- R = {1,2,3,4,5}
- S = {1,3,5,7,9}
- T = {1,4,7,10}

Can you write a query to select $R \cap (S \cup T)$- in other words elements that are in R and either S or T?

Write your query here:

Now test your query above for the case where $S = \emptyset$- what happens and why?

Execute the below, then re-run your query above

```
In [ ]: %%sql
        delete from S;
```

Task #2

- Schema is same as before

> Product (pname, price, category, manufacturer)
> Company (cname, stockPrice, country)

- Our goal is to answer the following question:

> Find all categories of products that are made by Chinese companies

Write your query here: