# Simple SQL Queries

### Submission details:

Please submit this HW as a Jupyter Notebook and a PDF of your results (both should show output).

For the submision create a local database with `sqlite3` or `sqlalchemy` in a Jupyter notebook and make the queries either with a cursor object (and then print the results) or by using pandas `pd.read_sql_query()`.

---

When completing this homework you can experiment with SQL commands by utilizing this great online editor:

### https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

There are already some tables in the online Database, namely:

```
Categories, Employees, OrderDetails, Orders, Products, Shippers, and Suppliers.
```

We are not going to use them, and if you want you can drop them by running `DROP TABLE [table-name];` (or just keep them).

# Homework

## Let's play with Dogs (& SQL)

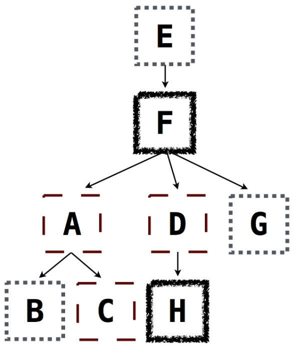
First create a table called parents. It has two columns: 'parent' and 'child'. The first column indicates the parent of the child in the second column. We will use a new form of `CREATE TABLE` expression to produce this table.

```
CREATE TABLE parents (
    parent VARCHAR(20),
    child VARCHAR(20));

INSERT INTO parents (parent, child)
  VALUES ("abraham", "barack") UNION
  VALUES ("abraham", "clinton") UNION
  VALUES ("delano", "herbert") UNION
  VALUES ("fillmore", "abraham") UNION
  VALUES ("fillmore", "delano") UNION
  VALUES ("fillmore", "grover") UNION
  VALUES ("eisenhower", "fillmore");
```

### Picture of the Dog Family Tree (illustration of parents table)

(A = abrham, B = barack, etc.)



## Q1 Simple SELECTS (on the parents table)

1. SELECT all records in the table.
2. SELECT child and parent, where abraham is the parent.
3. SELECT all children that have an 'e' in their name (hint: use LIKE and '%e%').
4. SELECT all unique parents (use SELECT DISTINCT) and order them by name, descending order (i.e. fillmore first)
5. SELECT all dogs that are siblings (one-to-one relations). Only show a sibling pair once. To do this you need to select two times from the parents table.

## Q2 Joins

Create a new table called dogs, which indicates the fur type of every dog. In the image above:

- long haired dogs = red dashed box
- curly haired dogs = black fluffy box
- short haired dogs = grey dotted box

Create the table by running (this is an alternative way to create a table that is a little bit more concise than the method shown during the lecture. Here SQL will figure out the datatypes of the columns):

```
CREATE TABLE dogs AS
  SELECT "abraham" AS name, "long" AS fur UNION
  SELECT "barack", "short" UNION
  SELECT "clinton", "long" UNION
  SELECT "delano", "long" UNION
  SELECT "eisenhower", "short" UNION
  SELECT "fillmore", "curly" UNION
  SELECT "grover", "short" UNION
  SELECT "herbert", "curly";
```

1. COUNT the number of short haired dogs
2. JOIN tables parents and dogs and SELECT the parents of curly dogs.
3. JOIN tables parents and dogs, and SELECT the parents and children that have the same fur type. Only show them once.

## Q3 Aggregate functions, numerical logic and grouping

Create a new table with many different animals. The table includes the animal's kind, number of legs and weight. Create it by running:

```
CREATE table animals AS
 SELECT "dog" AS kind, 4 AS legs, 20 AS weight UNION
 SELECT "cat" , 4 , 10 UNION
 SELECT "ferret" , 4 , 10 UNION
 SELECT "parrot" , 2 , 6 UNION
 SELECT "penguin" , 2 , 10 UNION
 SELECT "t-rex" , 2 , 12000;
```

1. SELECT the animal with the minimum weight. Display kind and min_weight.
2. Use the aggregate function AVG to display a table with the average number of legs and the average weight.
3. SELECT the animal kind(s) that have more than two legs, but weighs less than 20. Display kind, weight, legs.
4. SELECT the average weight for all the animals with 2 legs and the animals with 4 legs (by using GROUP BY).

.

.

.

.

*The HW is inspired by material from CS61A.*