Section 5 Worksheet

Use the Mondial dataset in hw5 to solve the following problems.

1. Return the **set** of all mountains.

```
SELECT W.mondial.mountain
  FROM world AS W;
```

Just like in the example code that is given in the spec, you are returning the value that is associated to the name “mountain” in the mondial object. We know that in our dataset “world” we have a single object with a single field called “monial.” And inside mondial, we have the fields “mountain,” “country,” “sea,” “desert,” …. When we SELECT x.mondial.mountain, what we are doing is literally extracting the value when you traverse down the JSON document tree (take a look at the mondial.adm file!).

2. Return each mountain one by one. Compare it to Problem 1.

```
SELECT MT as mountain
  FROM world AS W, W.mondial.mountain AS MT;
```

What is different about this query is that we have now specified W.mondial.mountain as a “table”. Really though the actual name of it is called a “iteration variable.” And that makes sense! When we stick W.mondial.mountain into the FROM clause, we iterate over the associated values just as we would do with tuples in a table. That being said we need to make sure that things we stick into the FROM clause are indeed iterable. With W.mondial.mountain, if you take a look inside you’ll notice that it is a list of objects that describe mountains!

3. Return name and type for each mountain, in descending order of the height. If the mountain does not specify a type, make the type “normal”.

```
SELECT MT.name, (CASE WHEN MT.`-type` IS MISSING THEN "normal"
                      ELSE MT.`-type` END),
       int(MT.height)
  FROM world AS W, W.mondial.mountain AS MT
 ORDER BY int(MT.height) DESC;
```

If we wish to deal with projecting out the values we want, we can be more specific about what we want in our SELECT clause. From the previous question, if we just SELECT MT we will return each mountain specifying object. If we perform the above SELECT, we will return the fields that in each of our objects. Note how the possibly missing field of “-type” is resolved via the CASE statement.

4. Find mountains located in more than 1 country. Your query should return mountain name and the count of how many countries each mountain is in.

```
SELECT MT.name AS mountainName, Count(*) AS numCountries
  FROM world AS W, W.mondial.mountain AS MT,
       split(MT.`-country`, ' ') AS CC
 GROUP BY MT.name
HAVING Count(*) > 1;
```

Just like SQL, SQL++ supports grouping. All the same semantics apply. Notice though now that we are getting multiple tuples per country for each mountain. We just did "joining" without having to specify predicates! This is the beautiful part of document stores and dealing with aggregate data types. We don't need to deal with joining unless we are doing something that steps outside of our predicate bounds. To reiterate, we did not have to specify joining conditions for the country codes with the mountains because the country codes that match to each mountain are already in the respective object.

5. For each country, return the country name and a list of all the mountain names in that country.

```
SELECT MT.name AS countryName, ML AS mountainList
  FROM world AS W, W.mondial.country AS MT
   LET ML = (SELECT MT2.name AS mountain
               FROM world AS W2, W2.mondial.mountain AS MT2,
                    split(MT2.`-country`, ' ') AS CC
              WHERE MT.`-car_code` = CC);
```

And again just like SQL, SQL++ supports subquerying. You can specify subqueries just like in SQL. Here we use a new keyword that is introduced into SQL++. The LET statement is to a SELECT subquery as the WITH clause is to a FROM subquery. Technically we could write the above query like this:

```
SELECT MT.name AS countryName,
       (SELECT MT2.name AS mountain
          FROM world AS W2, W2.mondial.mountain AS MT2,
               split(MT2.`-country`, ' ') AS CC
         WHERE MT.`-car_code` = CC) AS mountainList
  FROM world AS W, W.mondial.country AS MT;
```

But that's ugly! Makes you think about the ways SQL could be improved… :P