C. Relational algebra and SQL

1. **[5 pts] Construct the Cartesian product of the following three tables:**

A1	A2
1	**red**
2	**blue**

B1	B2	B3	B4
red	**17**	**1972**	**VW**
blue	**8**	**1968**	**Honda**

C1	C2	C3
Bob	**NYC**	**5000**
Elaine	**DC**	**18000**

A1	A2	B1	B2	B3	B4	C1	C2	C3
1	red	red	17	1972	VW	Bob	NYC	5000
1	red	red	17	1972	VW	Elaine	DC	18000
1	red	blue	8	1968	Honda	Bob	NYC	5000
1	red	blue	8	1968	Honda	Elaine	DC	18000
2	blue	red	17	1972	VW	Bob	NYC	5000
2	blue	red	17	1972	VW	Elaine	DC	18000
2	blue	blue	8	1968	Honda	Bob	NYC	5000
2	blue	blue	8	1968	Honda	Elaine	DC	18000

2. **[5 pts] Construct the natural join, left outer join and right outer join tables for the following tables:**

name	street	city
Coyote	**Toon**	**Hollywood**
Rabbit	**Tunnel**	**Carrotville**
Smith	**Revolver**	**Seattle**
Williams	**Seaview**	**Seattle**

name	branch_name	salary
Coyote	**Mesa**	**1500**
Rabbit	**Mesa**	**1300**
Gates	**Redmond**	**1500**
Brin	**Mountainview**	**3000**

natural(inner) join:

name	street	city	branch_name	salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300

left outer join:

name	street	city	branch_name	salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Smith	Revolver	Seattle	null	null
Williams	Seaview	Seattle	null	null

right outer join:

name	street	city	branch_name	salary
Coyote	Toon	Hollywood	Mesa	1500
Rabbit	Tunnel	Carrotville	Mesa	1300
Gates	null	null	Redmond	1500
Brin	null	null	Mountainview	3000

3. **[15 pts] Consider the following relational schema with the keys underlined:**

country(name, code, capital, province)
city(name, country, province, population)
borders(country1, country2, length)
encompasses(country, continent, percentage)

   a) **[5 pts] Does this schema allow you to represent the different spellings of a country's capital (e.g. the capital of China can be Beijing or Peking)? Why?**

   NO. "code" is a key in the country table, thus only one tuple may exist in this table for each country code, hence only one value in the "capital" column is allowed for each country.

   b) **[10 pts] Write the following queries in the relational algebra:**
   - find the names of all countries in Europe that do not border Switzerland.

   countries in Europe: $CIE=\pi_{\text{name}}(\sigma_{\text{country=code}\wedge\text{continent="Europe"}}(\text{country}\times\text{encompasses}))$
   countries bordering Switzerland:

$$CBS=\pi_{\text{c٢.name}}(\sigma_{\text{country١=c١.code}\wedge\text{country٢=c٢.code}\wedge\text{c١.name="Switzerland"}}(\rho_{\text{c١}}(\text{country})\times\rho_{\text{c٢}}(\text{country})\times\text{borders}))$$

   countries in Europe not bordering Switzerland: $CIE-CBS$

   - find the names of all cities in the world with population higher than every city in Peru.

   cities in the world with population lower than some city in Peru:

$$LTP=\pi_{\text{c١.name}}(\sigma_{\text{c١.population}<\text{c٢.population}\wedge\text{c٢.country=co.code}\wedge\text{co.name="Peru"}}(\rho_{\text{c١}}(\text{city})\times\rho_{\text{c٢}}(\text{city})\times\rho_{\text{co}}(\text{country})))$$

   all cities in the world: $ALL=\pi_{\text{name}}(\text{city})$
   cities with population larger than all cities in Peru (population not smaller than any city in Peru): $ALL-LTP$

4. **[15 pts] In the relational schema from point 3, write the following queries in SQL:**

   a) **[5 pts] Find all cities located in the same province as the capital city in France**

```
select c1.name
from city c1, city c2, country co
where
     co.name = 'France'   and
     c2.name = co.capital  and
     c1.province = c2.province
```

   or (since province is an attribute in the country table - the capital province)

```
select ci.name
from city ci, country co
where
     co.name = 'France' and ci.province = co.province
```

b) [5 pts] Find all countries not completely contained in Europe that border a country in Europe.

```
select co1.name // first part are countries that border a country in Europe
from country co1, country co2, borders b, encompasses e
where
    co1.code = b.country1 and
    co2.code = b.country2 and
    co2.code = e.country   and
    e.continent = 'Europe' and
    e.percentage = 100    // note the question is a bit vague here - this requires a
                          // country to border a country entirely contained in Europe
minus
select co.name   // here we get rid of all countries completely contained in Europe
from country co, encompasses e
where
    co.code = e.country       and
    e.continent = 'Europe'   and
    e.percentage = 100
```

c) [5 pts] Increase the population of all cities in France by 10%.

```
update city
set population = population * 1.1
where
    country in
       ( select country.code
         from country
         where
           country.name = 'France'
       )
```

5. [20 pts] Using the relational schema from point 3, answer the following questions:

a) [5 pts] Write an SQL query that returns the average length of borders shared by Venezuela with its neighbors.

```
select avg(length)
from country co, borders bo
where
    co.name = 'Venezuela' and
    co.code = bo.country1
group by co.name
```

b) **[10 pts] Rewrite the following query so it doesn't use sub-queries:**

```
select distinct co.name
from country co
where exists (select ci.name
              from city ci
              where ci.country = co.name and ci.population > 1000000)
```

```
select distinct co.name
from country co, city ci
where
    ci.country = co.code and    // Note I had a typo city.country refers to the country code
    ci.population > 1000000
```

c) **[5 pts] Is the following query correct? If yes, explain what it does. If no, explain what you think it is intended to do and how you would fix it. What assumption does this query make about the "borders" table?**

```
select co.name, sum(co2.population)
from country co, country co2, borders bo
where co.code = bo.country1
  and co2.code = bo.country2
```

The query is not correct because you cannot use aggregate functions (such as sum) without a group by statement.

The assumption made by this query (as by all the solutions I showed above) is that the "borders" relation is symmetric, i.e. both ("United States", "Canada"), and ("Canada", "United States") occur in this table. Otherwise this query would be incomplete.