Kevin Hance

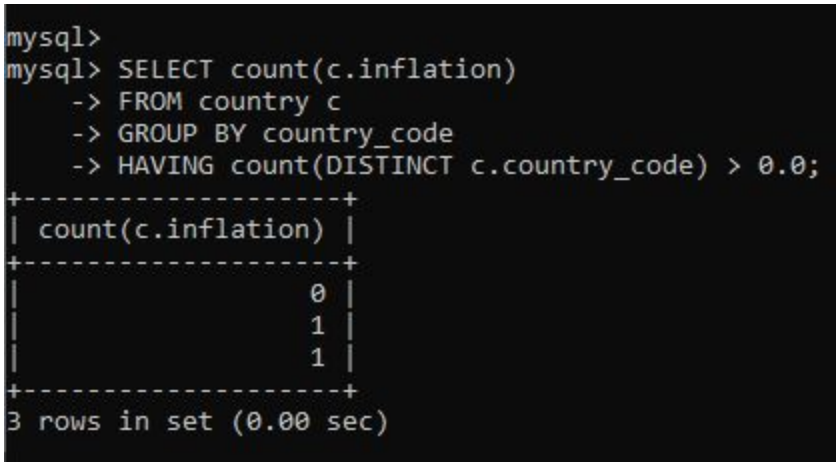
DBMS (CPSC 321)

HW7

READING ASSIGNMENT:

1. The main difference between SQL WHERE and HAVING is that WHERE operates on each individual row, whereas HAVING filters based on groups after a GROUP BY statement has been made. This impacts performance, as using a WHERE statement with a GROUP BY will attempt to filter all rows, some of which would not otherwise be included in the query result.
2. All aggregate functions except COUNT(*) will ignore null values. COUNT(*) is the exception in this case because, if there is no CASE statement that the row satisfies the criteria of, the row will be treated as a null value, despite the fact that it may have some non-null values that are worth counting.
3.
```
DROP TABLE IF EXISTS border;
DROP TABLE IF EXISTS city;
DROP TABLE IF EXISTS province;
DROP TABLE IF EXISTS country;

CREATE TABLE country(
    country_code VARCHAR(10),
    country_name VARCHAR(50),
    gdp INT UNSIGNED,
    inflation FLOAT,
    PRIMARY KEY (country_code)
);

CREATE TABLE province(
    province_name VARCHAR(50),
    country_code VARCHAR(10),
    area FLOAT UNSIGNED,
    PRIMARY KEY (province_name, country_code),
    FOREIGN KEY (country_code) REFERENCES country (country_code)
);

CREATE TABLE city(
    city_name VARCHAR(50),
    province_name VARCHAR(50),
    country_code VARCHAR(10),
    population INT UNSIGNED,
    PRIMARY KEY (city_name, province_name, country_code),
```

```
    FOREIGN KEY (province_name, country_code) REFERENCES province
(province_name, country_code)
);

CREATE TABLE border(
    country_code_1 VARCHAR(10),
    country_code_2 VARCHAR(10),
    border_length FLOAT UNSIGNED,
    PRIMARY KEY (country_code_1, country_code_2),
    FOREIGN KEY (country_code_1) REFERENCES country (country_code),
    FOREIGN KEY (country_code_2) REFERENCES country (country_code)
);

INSERT INTO country VALUES ('OS', 'Oswaldo', 78000, 6.7);
INSERT INTO country VALUES ('RL', 'Renlandia', 54000, 7.7);
INSERT INTO country VALUES ('GN', 'Geneva', 65000, null);

SELECT count(c.inflation)
FROM country c
GROUP BY country_code
HAVING count(DISTINCT c.country_code) > 0.0;
```



4. A view is similar to a table in that you can query the view for information in the form of tables. A prepared statement is a pre-written query on a table or view that returns a specific result.

TECHNICAL WORK:

1.

```
mysql> -- question 1
mysql> SELECT c.country_name, c.country_code, c.GDP, c.inflation, sum(cit.population)
    -> FROM country c JOIN city cit ON cit.country_code = c.country_code
    -> GROUP BY c.country_code;
+--------------+--------------+-------+-----------+---------------------+
| country_name | country_code | GDP   | inflation | sum(cit.population) |
+--------------+--------------+-------+-----------+---------------------+
| Geneva       | GN           | 65000 |       2.1 |            10834320 |
| Oswaldo      | OS           | 78000 |       6.7 |              728142 |
| Renlandia    | RL           | 54000 |       7.7 |              202026 |
+--------------+--------------+-------+-----------+---------------------+
3 rows in set (0.01 sec)
```

```
SELECT c.country_name, c.country_code, c.GDP, c.inflation,
sum(cit.population)
FROM country c JOIN city cit ON cit.country_code = c.country_code
GROUP BY c.country_code;
```

2.

```
mysql> SELECT p.country_code, p.province_name, p.area, sum(c.population)
    -> FROM province p JOIN city c ON p.province_name = c.province_name AND p.country_code = c.country_cod
    -> GROUP BY p.province_name
    -> HAVING sum(c.population) > @population;
+--------------+---------------+-------+-------------------+
| country_code | province_name | area  | sum(c.population) |
+--------------+---------------+-------+-------------------+
| GN           | Antalens      | 72003 |           5400120 |
| GN           | Huport        | 54041 |           5530210 |
+--------------+---------------+-------+-------------------+
2 rows in set (0.00 sec)
```

```
set @population = 5000000;
SELECT p.country_code, p.province_name, p.area, sum(c.population)
FROM province p JOIN city c ON p.province_name = c.province_name
ANDp.country_code = c.country_code
GROUP BY p.province_name
HAVING sum(c.population) > @population;
```

3.

```
mysql> SELECT c.country_code, c.country_name, count(DISTINCT cit.city_name)
    -> FROM country c JOIN city cit ON c.country_code = cit.country_code
    -> GROUP BY c.country_code
    -> ORDER BY count(DISTINCT cit.city_name) DESC;
+--------------+--------------+------------------------------+
| country_code | country_name | count(DISTINCT cit.city_name) |
+--------------+--------------+------------------------------+
| RL           | Renlandia    |                            5 |
| OS           | Oswaldo      |                            5 |
| GN           | Geneva       |                            4 |
+--------------+--------------+------------------------------+
3 rows in set (0.00 sec)
```

```
SELECT c.country_code, c.country_name, count(DISTINCT cit.city_name)
FROM country c JOIN city cit ON c.country_code = cit.country_code
GROUP BY c.country_code
ORDER BY count(DISTINCT cit.city_name) DESC;
```

4.

```
mysql> SELECT c.country_code, c.country_name, sum(p.area)
    -> FROM country c JOIN province p ON c.country_code = p.country_code
    -> GROUP BY c.country_code
    -> ORDER BY sum(p.area) DESC;
+--------------+--------------+-------------+
| country_code | country_name | sum(p.area) |
+--------------+--------------+-------------+
| GN           | Geneva       |      126044 |
| RL           | Renlandia    |       96720 |
| OS           | Oswaldo      |       85362 |
+--------------+--------------+-------------+
3 rows in set (0.00 sec)
```

```
SELECT c.country_code, c.country_name, sum(p.area)
FROM country c JOIN province p ON c.country_code = p.country_code
GROUP BY c.country_code
ORDER BY sum(p.area) DESC;
```

5.

```
mysql> SELECT country_name
    -> FROM country c JOIN province p JOIN city cit ON c.country_code = p.country_code AND p.country_code = cit.country
code
    -> GROUP BY c.country_code
    -> HAVING count(DISTINCT cit.city_name) >= @min_cities AND count(DISTINCT p.province_name) >= @min_provinces;
+--------------+
| country_name |
+--------------+
| Oswaldo      |
| Renlandia    |
+--------------+
2 rows in set (0.00 sec)
```

```
set @min_cities = 5;
set @min_provinces = 1;
SELECT country_name
FROM country c JOIN province p JOIN city cit ON c.country_code =
p.country_code AND p.country_code = cit.country_code
GROUP BY c.country_code
HAVING count(DISTINCT cit.city_name) >= @min_cities AND count(DISTINCT
p.province_name) >= @min_provinces;
```

6.

```
mysql> set @gdp = 60000;
SELEQuery OK, 0 rows affected (0.00 sec)

mysql> SELECT c.country_code, c.gdp, sum(p.area)
    -> FROM country c JOIN province p ON c.country_code = p.country_code
    -> GROUP BY c.country_code
    -> HAVING c.gdp >= @gdp
    -> ORDER BY
    -> CASE WHEN sum(p.area) <> 0 THEN sum(p.area) END DESC,
    -> CASE WHEN c.gdp <> 0 THEN c.gdp END DESC;
+--------------+-------+-------------+
| country_code | gdp   | sum(p.area) |
+--------------+-------+-------------+
| GN           | 65000 |      126044 |
| OS           | 78000 |       85362 |
+--------------+-------+-------------+
2 rows in set (0.00 sec)
```

```
set @gdp = 60000;
SELECT c.country_code, c.gdp, sum(p.area)
FROM country c JOIN province p ON c.country_code = p.country_code
GROUP BY c.country_code
HAVING c.gdp >= @gdp
ORDER BY
CASE WHEN sum(p.area) <> 0 THEN sum(p.area) END DESC,
CASE WHEN c.gdp <> 0 THEN c.gdp END DESC;
```

7.

```
mysql> DROP VIEW IF EXISTS sym_borders;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE VIEW sym_borders AS
    -> SELECT *
    -> FROM border;
Query OK, 0 rows affected (0.01 sec)ERT INTO sym_

mysql> INSERT INTO sym_borders
    -> SELECT country_code_2, country_code_1, border_length
    -> FROM border;
FROM sym_Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
mysql> SELECT * FROM sym_borders;
+----------------+----------------+---------------+
| country_code_1 | country_code_2 | border_length |
+----------------+----------------+---------------+
| GN             | OS             |          4800 |
| GN             | RL             |          4500 |
| OS             | GN             |          4800 |
| OS             | RL             |          1200 |
| RL             | GN             |          4500 |
| RL             | OS             |          1200 |
+----------------+----------------+---------------+
6 rows in set (0.00 sec)
```

```
DROP VIEW IF EXISTS sym_borders;
CREATE VIEW sym_borders AS
SELECT *
FROM border;
INSERT INTO sym_borders
SELECT country_code_2, country_code_1, border_length
FROM border;

SELECT * FROM sym_borders;
```

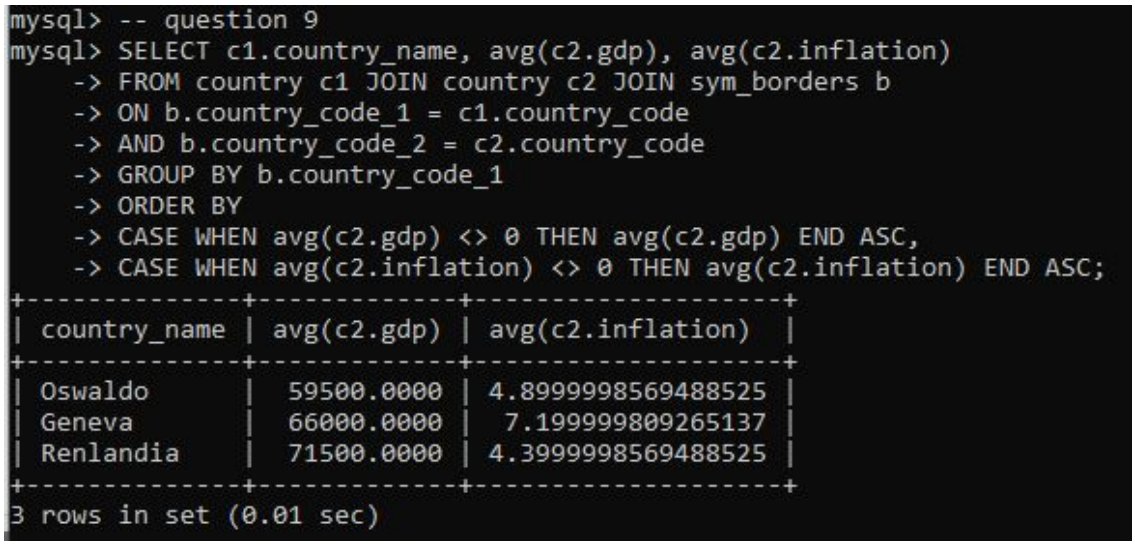
8.

```
mysql> -- question 8
mysql> -- original
mysql> (SELECT c1.country_name, c2.country_name as c_gdp_high
    -> FROM country c1 JOIN country c2 JOIN border bord
    -> ON (bord.country_code_1 = c1.country_code AND bord.country_code_2 = c2.country_code)
    -> WHERE c1.gdp < c2.gdp
    -> AND c1.inflation > c2.inflation)
    -> UNION
    -> (SELECT c2.country_name, c1.country_name as c_gdp_high
    -> FROM country c1 JOIN country c2 JOIN border bord
    -> ON (bord.country_code_1 = c1.country_code AND bord.country_code_2 = c2.country_code)
    -> WHERE c2.gdp < c1.gdp
    -> AND c2.inflation > c1.inflation);
LECT +--------------+------------+
| country_name | c_gdp_high |
+--------------+------------+
| Renlandia    | Geneva     |
| Renlandia    | Oswaldo    |
+--------------+------------+
2 rows in set (0.00 sec)

mysql> -- new query
mysql> SELECT c1.country_name, c2.country_name as c_gdp_high
    -> FROM country c1 JOIN country c2 JOIN sym_borders bord
    -> ON (bord.country_code_1 = c1.country_code AND bord.country_code_2 = c2.country_code)
    -> WHERE (c1.gdp < c2.gdp
    -> AND c1.inflation > c2.inflation)
    -> OR (c2.gdp < c1.gdp
    -> AND c2.inflation > c1.inflation);
+--------------+------------+
| country_name | c_gdp_high |
+--------------+------------+
| Renlandia    | Geneva     |
| Renlandia    | Oswaldo    |
| Geneva       | Renlandia  |
| Oswaldo      | Renlandia  |
+--------------+------------+
4 rows in set (0.00 sec)
```

```
-- question 8
-- original
(SELECT c1.country_name, c2.country_name as c_gdp_high
FROM country c1 JOIN country c2 JOIN border bord
ON (bord.country_code_1 = c1.country_code AND bord.country_code_2 =
c2.country_code)
WHERE c1.gdp < c2.gdp
AND c1.inflation > c2.inflation)
UNION
(SELECT c2.country_name, c1.country_name as c_gdp_high
FROM country c1 JOIN country c2 JOIN border bord
ON (bord.country_code_1 = c1.country_code AND bord.country_code_2 =
c2.country_code)
WHERE c2.gdp < c1.gdp
AND c2.inflation > c1.inflation);
-- new query
SELECT c1.country_name, c2.country_name as c_gdp_high
FROM country c1 JOIN country c2 JOIN sym_borders bord
```

```
ON (bord.country_code_1 = c1.country_code AND bord.country_code_2 =
c2.country_code)
WHERE (c1.gdp < c2.gdp
AND c1.inflation > c2.inflation)
OR (c2.gdp < c1.gdp
AND c2.inflation > c1.inflation);
```

9.



```
mysql> -- question 9
mysql> SELECT c1.country_name, avg(c2.gdp), avg(c2.inflation)
    -> FROM country c1 JOIN country c2 JOIN sym_borders b
    -> ON b.country_code_1 = c1.country_code
    -> AND b.country_code_2 = c2.country_code
    -> GROUP BY b.country_code_1
    -> ORDER BY
    -> CASE WHEN avg(c2.gdp) <> 0 THEN avg(c2.gdp) END ASC,
    -> CASE WHEN avg(c2.inflation) <> 0 THEN avg(c2.inflation) END ASC;
+--------------+-------------+--------------------+
| country_name | avg(c2.gdp) | avg(c2.inflation)  |
+--------------+-------------+--------------------+
| Oswaldo      |  59500.0000 | 4.8999998569488525 |
| Geneva       |  66000.0000 |  7.199999809265137 |
| Renlandia    |  71500.0000 | 4.3999998569488525 |
+--------------+-------------+--------------------+
3 rows in set (0.01 sec)
```

```
-- question 9
SELECT c1.country_name, avg(c2.gdp), avg(c2.inflation)
FROM country c1 JOIN country c2 JOIN sym_borders b
ON b.country_code_1 = c1.country_code
AND b.country_code_2 = c2.country_code
GROUP BY b.country_code_1
ORDER BY
CASE WHEN avg(c2.gdp) <> 0 THEN avg(c2.gdp) END ASC,
CASE WHEN avg(c2.inflation) <> 0 THEN avg(c2.inflation) END ASC;
```

10.

```
mysql> -- question 10
mysql> -- part 1:  Show all cities which are in a country with a designated relationship between
mysql> --          the country's inflation and gdp, and those of a bordering country. Should return
mysql> --          the name of the city, name of the province the city is in, and the population of the
mysql> --          city, ordered by population of the city going from highest to lowest.
mysql> -- first argument: country 1 has a higher gdp, higher inflation
mysql> SELECT c.city_name, c.province_name, c.population
    -> FROM country c1 JOIN country c2 JOIN sym_borders b JOIN city c
    -> ON b.country_code_1 = c1.country_code
    -> AND b.country_code_2 = c2.country_code
    -> AND c.country_code = c1.country_code
    -> WHERE c1.gdp > c2.gdp
    -> AND c1.inflation > c2.inflation
    -> ORDER BY c.population DESC;
a+-----------+---------------+------------+
| city_name | province_name | population |
+-----------+---------------+------------+
| Slaren    | Oslodo        |     590190 |
| Belogonia | St. Janice    |      97635 |
| Britano   | St. Janice    |      33434 |
| Sluurgan  | Oslodo        |       5919 |
| Tesa      | Oslodo        |        964 |
+-----------+---------------+------------+
5 rows in set (0.01 sec)
```

```
mysql> -- second argument: country 1 has a higher gdp, lower inflation
mysql> SELECT c.city_name, c.province_name, c.population
    -> FROM country c1 JOIN country c2 JOIN sym_borders b JOIN city c
    -> ON b.country_code_1 = c1.country_code
    -> AND b.country_code_2 = c2.country_code
    -> AND c.country_code = c1.country_code
    -> WHERE c1.gdp < c2.gdp
    -> AND c1.inflation > c2.inflation
    -> ORDER BY c.population DESC;
+-----------+---------------+------------+
| city_name | province_name | population |
+-----------+---------------+------------+
| Gonba     | Flaubury      |      53146 |
| Gonba     | Flaubury      |      53146 |
| Whita     | Flaubury      |      52743 |
| Whita     | Flaubury      |      52743 |
| Stombus   | Huport        |      49384 |
| Stombus   | Huport        |      49384 |
| Piolas    | Huport        |      46626 |
| Piolas    | Huport        |      46626 |
| Outling   | Flaubury      |        127 |
| Outling   | Flaubury      |        127 |
+-----------+---------------+------------+
10 rows in set (0.00 sec)

mysql>
mysql> -- part 2:  Show all provinces with a designated relationshop between the area of that province
mysql> --          and that of a province in a bordering country. Should return the name of each province,
mysql> --          the area of each province, and the names of the countries that border each other.
mysql> -- first argument: province 1 has a higher area
mysql> SELECT p1.province_name, p2.province_name, p1.area, p2.area, c1.country_code, c2.country_code
    -> FROM country c1 JOIN country c2 JOIN sym_borders b
    -> JOIN province p1 JOIN province p2
    -> ON b.country_code_1 = c1.country_code
    -> AND b.country_code_2 = c2.country_code
    -> AND c1.country_code = p1.country_code
    -> AND c2.country_code = p2.country_code
    -> WHERE p1.area > p2.area
    -> ORDER BY p1.area DESC;
 sec+---------------+---------------+-------+-------+--------------+--------------+
| province_name | province_name | area  | area  | country_code | country_code |
+---------------+---------------+-------+-------+--------------+--------------+
| Huport        | Antalens      | 91030 | 72003 | RL           | GN           |
| Huport        | Huport        | 91030 | 54041 | RL           | GN           |
| Huport        | Oslodo        | 91030 |  8712 | RL           | OS           |
| Huport        | St. Janice    | 91030 | 76650 | RL           | OS           |
| St. Janice    | Antalens      | 76650 | 72003 | OS           | GN           |
| St. Janice    | Flaubury      | 76650 |  5690 | OS           | RL           |
| St. Janice    | Huport        | 76650 | 54041 | OS           | GN           |
| Antalens      | Flaubury      | 72003 |  5690 | GN           | RL           |
| Antalens      | Oslodo        | 72003 |  8712 | GN           | OS           |
| Huport        | Flaubury      | 54041 |  5690 | GN           | RL           |
| Huport        | Oslodo        | 54041 |  8712 | GN           | OS           |
| Oslodo        | Flaubury      |  8712 |  5690 | OS           | RL           |
+---------------+---------------+-------+-------+--------------+--------------+
```

```
mysql> -- second argument: province 1 has a lower area
mysql> SELECT p1.province_name, p2.province_name, p1.area, p2.area, c1.country_code, c2.country_code
    -> FROM country c1 JOIN country c2 JOIN sym_borders b
    -> JOIN province p1 JOIN province p2
    -> ON b.country_code_1 = c1.country_code
    -> AND b.country_code_2 = c2.country_code
    -> AND c1.country_code = p1.country_code
    -> AND c2.country_code = p2.country_code
    -> WHERE p1.area < p2.area
    -> ORDER BY p1.area DESC;
+---------------+---------------+-------+-------+--------------+--------------+
| province_name | province_name | area  | area  | country_code | country_code |
+---------------+---------------+-------+-------+--------------+--------------+
| St. Janice    | Huport        | 76650 | 91030 | OS           | RL           |
| Antalens      | Huport        | 72003 | 91030 | GN           | RL           |
| Antalens      | St. Janice    | 72003 | 76650 | GN           | OS           |
| Huport        | Huport        | 54041 | 91030 | GN           | RL           |
| Huport        | St. Janice    | 54041 | 76650 | GN           | OS           |
| Oslodo        | Huport        |  8712 | 91030 | OS           | RL           |
| Oslodo        | Antalens      |  8712 | 72003 | OS           | GN           |
| Oslodo        | Huport        |  8712 | 54041 | OS           | GN           |
| Flaubury      | Antalens      |  5690 | 72003 | RL           | GN           |
| Flaubury      | Huport        |  5690 | 54041 | RL           | GN           |
| Flaubury      | Oslodo        |  5690 |  8712 | RL           | OS           |
| Flaubury      | St. Janice    |  5690 | 76650 | RL           | OS           |
+---------------+---------------+-------+-------+--------------+--------------+
12 rows in set (0.00 sec)

mysql>
```

```
-- question 10
-- part 1:  Show all cities which are in a country with a designated relationship between
--          the country's inflation and gdp, and those of a bordering country. Should return
--          the name of the city, name of the province the city is in, and the population of the
--          city, ordered by population of the city going from highest to lowest.
-- first argument: country 1 has a higher gdp, higher inflation
SELECT c.city_name, c.province_name, c.population
FROM country c1 JOIN country c2 JOIN sym_borders b JOIN city c
ON b.country_code_1 = c1.country_code
AND b.country_code_2 = c2.country_code
AND c.country_code = c1.country_code
WHERE c1.gdp > c2.gdp
AND c1.inflation > c2.inflation
ORDER BY c.population DESC;
-- second argument: country 1 has a higher gdp, lower inflation
SELECT c.city_name, c.province_name, c.population
FROM country c1 JOIN country c2 JOIN sym_borders b JOIN city c
ON b.country_code_1 = c1.country_code
AND b.country_code_2 = c2.country_code
AND c.country_code = c1.country_code
WHERE c1.gdp < c2.gdp
AND c1.inflation > c2.inflation
```

```
ORDER BY c.population DESC;

-- part 2:  Show all provinces with a designated relationshop between
the area of that province
--          and that of a province in a bordering country. Should
return the name of each province,
--          the area of each province, and the names of the countries
that border each other.
-- first argument: province 1 has a higher area
SELECT p1.province_name, p2.province_name, p1.area, p2.area,
c1.country_code, c2.country_code
FROM country c1 JOIN country c2 JOIN sym_borders b
JOIN province p1 JOIN province p2
ON b.country_code_1 = c1.country_code
AND b.country_code_2 = c2.country_code
AND c1.country_code = p1.country_code
AND c2.country_code = p2.country_code
WHERE p1.area > p2.area
ORDER BY p1.area DESC;
-- second argument: province 1 has a lower area
SELECT p1.province_name, p2.province_name, p1.area, p2.area,
c1.country_code, c2.country_code
FROM country c1 JOIN country c2 JOIN sym_borders b
JOIN province p1 JOIN province p2
ON b.country_code_1 = c1.country_code
AND b.country_code_2 = c2.country_code
AND c1.country_code = p1.country_code
AND c2.country_code = p2.country_code
WHERE p1.area < p2.area
ORDER BY p1.area DESC;
```