```
/**********************************************************************
 * NAME: Samuel Sovi
 * CLASS: CPSC 321 Section 1
 * DATE: 10/11/22
 * HOMEWORK: hw2
 * DESCRIPTION: Implementing Tables and Schemas to practice creating
 * relationships between tables using foreign keys and using JOINs
 *
 * ADDITIONAL NOTE:
 *      Useful commands
 *      1) mysql -p -h cps-database
 *      2) use ssovi_DB;
 *      3) show tables;
 *      4) exit
 **********************************************************************/


-- TODO: add drop table statements
DROP TABLE IF EXISTS border;
DROP TABLE IF EXISTS city;
DROP TABLE IF EXISTS province;
DROP TABLE IF EXISTS country;


-- TODO: add create table statements

-- country table used to store countries
CREATE TABLE country (
    -- country code is a unique small code for each country
    country_code VARCHAR(5) NOT NULL,
    -- country name
    country_name VARCHAR(60) NOT NULL,
    -- gdp of a given country
    gdp INT NOT NULL,
    -- inflation value of a country
    inflation FLOAT(8,2) NOT NULL,
    PRIMARY KEY (country_code)
);

CREATE TABLE province (
    -- province name
    province_name VARCHAR(60) NOT NULL,
    -- country code is a unique small code for each country
    country_code VARCHAR(5) NOT NULL,
    -- area of the province in km^2
    area INT NOT NULL,
    PRIMARY KEY (province_name, country_code),
    FOREIGN KEY (country_code) REFERENCES country (country_code)
);

CREATE TABLE city (
    -- name of the city
    city_name VARCHAR(60) NOT NULL,
    -- name of the province/state the city is in
    province_name VARCHAR(60) NOT NULL,
    -- country code of the country that the city is in
    country_code VARCHAR(5) NOT NULL,
    -- population of the city
    population INT NOT NULL,
```

```
    PRIMARY KEY (city_name, province_name, country_code),
    FOREIGN KEY (country_code) REFERENCES country (country_code),
    FOREIGN KEY (province_name) REFERENCES province (province_name)
);

CREATE TABLE border (
    -- first country of the two sharing a border with each other
    country_code_1 VARCHAR(5) NOT NULL,
    -- second country of the two sharing a border with each other
    country_code_2 VARCHAR(5) NOT NULL,
    -- length of the shared border between both countries
    border_length INT NOT NULL,
    PRIMARY KEY (country_code_1, country_code_2),
    FOREIGN KEY (country_code_1) REFERENCES country (country_code),
    FOREIGN KEY (country_code_1) REFERENCES country (country_code),
    -- note, country 1 has to be > country 2 (to preserve unique borders)
    -- this makes it so example: US, CN and CN, US cannot be different borders
    CONSTRAINT unique_code CHECK (country_code_1 > country_code_2)
);


-- TODO: add insert statements

-- Countries

INSERT INTO country VALUES ("US", "Fake United States", 46900, 3.8);
INSERT INTO country VALUES ("JP", "Fake Japan", 20456, 4.5);
INSERT INTO country VALUES ("CN", "Fake China", 60800, 2.4);

-- Provinces:

INSERT INTO province VALUES ("Californio", "US", 40000);
INSERT INTO province VALUES ("New Pork", "US", 2000);
INSERT INTO province VALUES ("Areazona", "US", 27000);

INSERT INTO province VALUES ("Brokyo", "JP", 7000);
INSERT INTO province VALUES ("Keeoto", "JP", 3000);
INSERT INTO province VALUES ("Saporofessor", "JP", 13000);

INSERT INTO province VALUES ("Hongey Kong", "CN", 17000);
INSERT INTO province VALUES ("Beifeng", "CN", 26000);
INSERT INTO province VALUES ("Shangfry", "CN", 62000);

-- US cities:

INSERT INTO city VALUES ("Sant Jose", "Californio", "US", 5);
INSERT INTO city VALUES ("Sant Franciscbro", "Californio", "US", 25000);
INSERT INTO city VALUES ("Los Pantalones", "Californio", "US", 150);

INSERT INTO city VALUES ("Judon", "New Pork", "US", 210293);
INSERT INTO city VALUES ("Rilas", "New Pork", "US", 29183);
INSERT INTO city VALUES ("Pagos", "New Pork", "US", 203);

INSERT INTO city VALUES ("Poit", "Areazona", "US", 293);
INSERT INTO city VALUES ("Canta", "Areazona", "US", 283);
INSERT INTO city VALUES ("Crey", "Areazona", "US", 22103);

-- JP cities

INSERT INTO city VALUES ("Yashiori", "Brokyo", "JP", 1500);
```

```
INSERT INTO city VALUES ("Kuro", "Brokyo", "JP", 17203);
INSERT INTO city VALUES ("Shinipachi", "Brokyo", "JP", 132);

INSERT INTO city VALUES ("Chungye", "Keeoto", "JP", 128);
INSERT INTO city VALUES ("Yami", "Keeoto", "JP", 123092);
INSERT INTO city VALUES ("Hirigata", "Keeoto", "JP", 2183);

INSERT INTO city VALUES ("Tachiban", "Saporofessor", "JP", 2);
INSERT INTO city VALUES ("Hanamata", "Saporofessor", "JP", 1230);
INSERT INTO city VALUES ("Akihama", "Saporofessor", "JP", 12302);

-- CN cities

INSERT INTO city VALUES ("Qingcheng", "Hongey Kong", "CN", 1500);
INSERT INTO city VALUES ("Chungye", "Hongey Kong", "CN", 128);
INSERT INTO city VALUES ("Seopo", "Hongey Kong", "CN", 12392);

INSERT INTO city VALUES ("Irelia", "Beifeng", "CN", 21382);
INSERT INTO city VALUES ("Qiyana", "Beifeng", "CN", 493);
INSERT INTO city VALUES ("Yuumi", "Beifeng", "CN", 10802);

INSERT INTO city VALUES ("Qiling", "Shangfry", "CN", 1284);
INSERT INTO city VALUES ("Ningguang", "Shangfry", "CN", 9432);
INSERT INTO city VALUES ("Wangsheng", "Shangfry", "CN", 502834);

-- borders

INSERT INTO border VALUES ("US", "CN", 12303);
INSERT INTO border VALUES ("JP", "CN", 1234);
-- Note that the CN after JP was intentionally made this way to test Problems 9
and 10 in part 2
-- The opposite order will also work if the CONSTRAINT in border is switched :)
```

```
/**************************************************************************
 * NAME: Samuel Sovi
 * CLASS: CPSC 321 Section 1
 * DATE: 10/11/22
 * HOMEWORK: hw2
 * DESCRIPTION: Implementing Tables and Schemas to practice creating
 * relationships between tables using foreign keys and using JOINs
 *
 * ADDITIONAL NOTE:
 *      Useful commands
 *      1) mysql -p -h cps-database
 *      2) use ssovi_DB;
 *      3) show tables;
 *      4) exit
 **************************************************************************/


-- Tables:
\! echo 'Country:';
SELECT *
FROM country;

\! echo 'Province:';
SELECT *
FROM province;

\! echo 'City:';
SELECT *
FROM city;

\! echo 'Border:';
SELECT *
FROM border;

-- TODO: Implement the queries in part 2 below. For each be sure to
--       copy each of the problem statements.



-- Quesiton 1:
\! echo 'Question 1:';
SELECT *
FROM country
WHERE inflation < 3 and gdp > 40000;

-- Question 2:
\! echo 'Question 2:';
SELECT c.country_code, c.country_name, c.inflation, p.province_name, p.area
FROM country c, province p
WHERE c.country_code = p.country_code AND p.area < 10000
ORDER BY c.inflation DESC, c.country_code ASC, p.area ASC;



-- Question 3:
\! echo 'Question 3:';
SELECT c.country_code, c.country_name, c.inflation, p.province_name, p.area
FROM country c JOIN province p ON (c.country_code = p.country_code)
WHERE p.area < 10000
```

```
ORDER BY c.inflation DESC, c.country_code ASC, p.area ASC;

-- Question 4:
\! echo 'Question 4:';
SELECT DISTINCT p.province_name
FROM country c, province p, city c1
WHERE c.country_code = p.country_code AND p.province_name = c1.province_name AND
 c1.population > 30000;

-- Question 5:
\! echo 'Question 5:';
SELECT DISTINCT p.province_name
FROM country c JOIN province p ON (c.country_code = p.country_code) JOIN city c1
 ON (p.province_name = c1.province_name)
WHERE c1.population > 30000;

-- Question 6:
\! echo 'Question 6:';
SELECT p.province_name
FROM country c, province p, city c1, city c2
WHERE c.country_code = p.country_code AND p.province_name = c1.province_name AND
 c1.province_name = c2.province_name
    AND c1.city_name > c2.city_name AND c1.population > 10000 AND c2.population
> 10000;

-- Question 7:
\! echo 'Question 7:';
SELECT p.province_name
FROM country c JOIN province p ON (c.country_code = p.country_code) JOIN city c1
 ON (p.province_name = c1.province_name) JOIN city c2 ON (c1.province_name = c2.
province_name)
WHERE c1.city_name > c2.city_name AND c1.population > 10000 AND c2.population >
10000;

-- Question 8:
\! echo 'Question 8:';
SELECT c1.city_name, c1.province_name, c1.country_code, c2.city_name, c2.provinc
e_name, c2.country_code, c1.population
FROM city c1 JOIN city c2 ON (c1.population = c2.population)
WHERE c1.city_name > c2.city_name OR (c1.province_name > c2.province_name AND c1
.city_name = c2.city_name);

-- Question 9:
\! echo 'Question 9:';
SELECT c1.country_name, c1.country_code
FROM country c1, country c2, border b
WHERE c1.gdp > 50000 AND c1.inflation < 3 AND c2.gdp < 30000 AND c2. inflation >
 4 AND ((c1.country_code = b.country_code_1
    AND c2.country_code = b.country_code_2) OR (c2.country_code = b.country_code
_1 AND c1.country_code = b.country_code_2));



-- Question 10:
\! echo 'Question 10:';
SELECT c1.country_name, c1.country_code, c2.country_code
FROM country c1 CROSS JOIN country c2 JOIN border b ON ((c1.country_code = b.cou
ntry_code_1
    AND c2.country_code = b.country_code_2) OR (c1.country_code = b.country_code
_2 AND c2.country_code = b.country_code_1))
```

```
WHERE (c1.gdp > 50000 AND c1.inflation < 3 AND c2.gdp < 30000 AND c2. inflation
> 4);
```

Tables:

Country:

country_code	country_name	gdp	inflation
CN	Fake China	60800	2.40
JP	Fake Japan	20456	4.50
US	Fake United States	46900	3.80

3 rows in set (0.000 sec)

Province:

province_name	country_code	area
Areazona	US	27000
Beifeng	CN	26000
Brokyo	JP	7000
Californio	US	40000
Hongey Kong	CN	17000
Keeoto	JP	3000
New Pork	US	2000
Saporofessor	JP	13000
Shangfry	CN	62000

9 rows in set (0.000 sec)

City:

city_name	province_name	country_code	population
Akihama	Saporofessor	JP	12302
Canta	Areazona	US	283
Chungye	Hongey Kong	CN	128
Chungye	Keeoto	JP	128
Crey	Areazona	US	22103
Hanamata	Saporofessor	JP	1230
Hirigata	Keeoto	JP	2183
Irelia	Beifeng	CN	21382
Judon	New Pork	US	210293
Kuro	Brokyo	JP	17203
Los Pantalones	Californio	US	150
Ningguang	Shangfry	CN	9432
Pagos	New Pork	US	203
Poit	Areazona	US	293
Qiling	Shangfry	CN	1284
Qingcheng	Hongey Kong	CN	1500
Qiyana	Beifeng	CN	493
Rilas	New Pork	US	29183
Sant Franciscbro	Californio	US	25000
Sant Jose	Californio	US	5
Seopo	Hongey Kong	CN	12392
Shinipachi	Brokyo	JP	132
Tachiban	Saporofessor	JP	2
Wangsheng	Shangfry	CN	502834
Yami	Keeoto	JP	123092
Yashiori	Brokyo	JP	1500
Yuumi	Beifeng	CN	10802

27 rows in set (0.000 sec)

Border:

country_code_1	country_code_2	border_length
JP	CN	1234
US	CN	12303

2 rows in set (0.000 sec)

Queries

```
Question 1:
+--------------+--------------+-------+-----------+
| country_code | country_name | gdp   | inflation |
+--------------+--------------+-------+-----------+
| CN           | Fake China   | 60800 |      2.40 |
+--------------+--------------+-------+-----------+
1 row in set (0.000 sec)

Question 2:
+--------------+--------------------+-----------+----------------+------+
| country_code | country_name       | inflation | province_name  | area |
+--------------+--------------------+-----------+----------------+------+
| JP           | Fake Japan         |      4.50 | Keeoto         | 3000 |
| JP           | Fake Japan         |      4.50 | Brokyo         | 7000 |
| US           | Fake United States |      3.80 | New Pork       | 2000 |
+--------------+--------------------+-----------+----------------+------+
3 rows in set (0.000 sec)

Question 3:
+--------------+--------------------+-----------+----------------+------+
| country_code | country_name       | inflation | province_name  | area |
+--------------+--------------------+-----------+----------------+------+
| JP           | Fake Japan         |      4.50 | Keeoto         | 3000 |
| JP           | Fake Japan         |      4.50 | Brokyo         | 7000 |
| US           | Fake United States |      3.80 | New Pork       | 2000 |
+--------------+--------------------+-----------+----------------+------+
3 rows in set (0.000 sec)

Question 4:
+----------------+
| province_name  |
+----------------+
| Shangfry       |
| Keeoto         |
| New Pork       |
+----------------+
3 rows in set (0.000 sec)

Question 5:
+----------------+
| province_name  |
+----------------+
| Shangfry       |
| Keeoto         |
| New Pork       |
+----------------+
3 rows in set (0.000 sec)

Question 6:
+----------------+
| province_name  |
+----------------+
| Beifeng        |
| New Pork       |
+----------------+
2 rows in set (0.000 sec)

Question 7:
+----------------+
| province_name  |
+----------------+
| Beifeng        |
| New Pork       |
+----------------+
2 rows in set (0.000 sec)

Question 8:
+-----------+----------------+--------------+-----------+----------------+--------------+------------+
| city_name | province_name  | country_code | city_name | province_name  | country_code | population |
+-----------+----------------+--------------+-----------+----------------+--------------+------------+
| Chungye   | Keeoto         | JP           | Chungye   | Hongey Kong    | CN           |        128 |
| Yashiori  | Brokyo         | JP           | Qingcheng | Hongey Kong    | CN           |       1500 |
+-----------+----------------+--------------+-----------+----------------+--------------+------------+
2 rows in set (0.000 sec)

Question 9:
+--------------+--------------+
| country_name | country_code |
+--------------+--------------+
| Fake China   | CN           |
+--------------+--------------+
1 row in set (0.000 sec)

Question 10:
+--------------+--------------+--------------+
| country_name | country_code | country_code |
+--------------+--------------+--------------+
| Fake China   | CN           | JP           |
+--------------+--------------+--------------+
1 row in set (0.000 sec)
```