-6 missing screenshots & output...

34/40

CPSC 321: Assignment 5 *Due: Tuesday, Oct. 19*

Part 1: SQL Table Creation.

1. (14 pts) Implement the following schema using `CREATE TABLE` statements for your MariaDB database. The schema is loosely based on the CIA World Factbook.[1] Your statements must go into a script file, with appropriate comments, called `hw5-a.sql`. You must pick suitable attribute data types and include primary and foreign keys as specified below.

   `Country(country_code, country_name, gdp, inflation)`

   - A country has a country codes (e.g., "US"), a full name (e.g., "United States of America"), a gross domestic product per capita (e.g., 46,900 dollars per person), and inflation rates (e.g., 3.8 percent), where `country_code` is the primary key. Assume that two countries with different country codes can have the same country name.

   `Province(province_name, country_code, area)`

   - A province (which is a state in the US) consists of a name (e.g., "Washington"), the country code the province is located in, and the total province area in km^2. The `province_name` and `country_coode` together form the primary key, with `country_code` a foreign key to the `Country` table. Assume it is possible for two countries to have a province with the same name (e.g., Montana exists in both the US and Bulgaria).

   `City(city_name, province_name, country_code, population)`

   - A city is identified by its name, province, and country, and has a total population. The `province_name` and `country_code` together define a foreign key to the `Province` table. Assume it is possible for two provinces to have a city with the same name (e.g., Portland is a city in both Oregon and Maine).

   `Border(country_code_1, country_code_2, border_length)`

   - A border defines a connection between two countries, with a corresponding border length in km. Both *country_code_1* and *country_code_2* are separately foreign keys to the `Country` table. Assume there is only one row in the table for a given border between two countries (i.e., the table does not store a symmetric closure over the border relation).

2. (6 pts) Populate your tables in Question 1 using `INSERT INTO` statements with enough data to test your table constraints. At a minimum, you must include at least three

[1] https://www.cia.gov/the-world-factbook/

different countries, three different provinces per country, and three different cities per province. You must also include two borders. Include your insert statements in your `hw5-a.sql` file. Note you do not have to use "real" data when populating your tables. If you use real data, it does not have to be "comprehensive", e.g., you do not need to include all provinces within a country, and you do not need to include all cities within a province.

3. (14 pts) Implement the tables from the schema you obtained by translating your Lego ER Diagram in HW-4 (Question 4). Put all of your `CREATE TABLE` statements, with appropriate comments, in a script file named `hw5-b.sql`. Select suitable attribute data types and include all key, foreign key, and non null constraints as appropriate.

4. (6 pts) Populate your tables in Question 3 using `INSERT INTO` statements with enough data to test your implementation. You are free to use real or fake data. However, if you use real data, you do not need to be comprehensive (e.g., you do not need to include all bricks in a set, etc.).

Part 2: Project. There are no additional tasks that you need to turn in for your project for this week.

Submission. Submit your answers to the above questions as follows.

- Submit both script files `hw5-a.sql` and `hw5-b.sql`. Be sure your files are commented, including a file header with your name, the class (CPSC 321), the semester, the homework number, and a brief description. Each table should have a comment describing its purpose and any additional information that may be needed to understand the table. Comment attributes as appropriate. Each attribute does not need a comment. However, comments should be provided for attributes if they may be unclear to someone else looking at your schema.

- For each table you create and populate, provide a printout of the table contents in a separate file named `hw4-results.pdf`. Include your name, the class number, the semester, and the homework number at the top of the file. You can print the contents of the table using the simple SQL query `SELECT * FROM` *table*`;` by replacing *table* with the name of the table you want to print out. Note you can add these select statements to the end of your script files.

- Submit your assignment files using GitHub classroom (see instructions on piazza). An initial version of `hw5-a.sql` and `hw5-b.sql` will be provided as starter code.

```
/*****************************************************************
* NAME: Zachary Foteff
* CLASS: CPSC321
* DATE: 10/18/2021
* HOMEWORK: 5
* DESCRIPTION:  Create and insert statements for creating a database of
*               information regarding countries, provinces, and cities
*****************************************************************/
--  Drop table statements
DROP TABLE IF EXISTS Province;
DROP TABLE IF EXISTS City;
DROP TABLE IF EXISTS Border;
DROP TABLE IF EXISTS Country;
--  Create table statements
CREATE TABLE Country (
    --  3 character code for a country
    country_code    VARCHAR(3) NOT NULL,
    country_name    TINYTEXT NOT NULL,
    gdp             DECIMAL(12, 2) NOT NULL,
    inflation       DECIMAL(4, 2) NOT NULL,
    PRIMARY KEY (country_code)
);

CREATE TABLE Province (
    province_name   VARCHAR(100) NOT NULL,
    country_code    VARCHAR(4) NOT NULL,
    area            DECIMAL(11,3) UNSIGNED NOT NULL,
    PRIMARY KEY (province_name, country_code),
    FOREIGN KEY (country_code) REFERENCES Country (country_code)
);

CREATE TABLE City (
    --  City/municipality name
    city_name       VARCHAR(100) NOT NULL,
    province_name   VARCHAR(100) NOT NULL,
    country_code    VARCHAR(4) NOT NULL,
    city_population INT UNSIGNED NOT NULL,
    PRIMARY KEY (city_name, province_name, country_code),
    FOREIGN KEY (country_code) REFERENCES Country (country_code)
);

CREATE TABLE Border (
    country_code_1  VARCHAR(4) NOT NULL,
    country_code_2  VARCHAR(4) NOT NULL,
    border_length   INT UNSIGNED NOT NULL,
    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 statements
--  Country insertions
INSERT INTO Country VALUES ('USA', 'UNITED STATES', 20936600.00, 5.4);
INSERT INTO Country VALUES ('MEX', 'MEXICO', 1076163000.32, 6.0);
INSERT INTO Country VALUES ('CAN', 'CANADA', 1643407000.98, 4.1);
INSERT INTO Country VALUES ('GUA', 'GUATEMALA', 77604632.17, 3.67);

--  US province insertions
INSERT INTO Province VALUES ('Oregon', 'USA', 248607.80);
INSERT INTO Province VALUES ('Washington', 'USA', 278479.97);
```

good

```
INSERT INTO Province VALUES ('Delaware', 'USA', 5133.36);

--  US city insertions
INSERT INTO City VALUE ('Happy Valley', 'Oregon', 'USA', 90000);
INSERT INTO City VALUE ('Eugene', 'Oregon', 'USA', 115000);
INSERT INTO City VALUE ('Portland', 'Oregon', 'USA', 150000);
INSERT INTO City VALUE ('Walla Wall', 'Washington', 'USA', 90000);
INSERT INTO City VALUE ('Spokane', 'Washington', 'USA', 115000);
INSERT INTO City VALUE ('Seattle', 'Washington', 'USA', 150000);
INSERT INTO City VALUE ('Pike Creek', 'Delaware', 'USA', 90000);
INSERT INTO City VALUE ('Arden', 'Delaware', 'USA', 115000);
INSERT INTO City VALUE ('Wilmington', 'Delaware', 'USA', 150000);

--  MEX province insertions
INSERT INTO Province VALUES ('Sinaloa', 'MEX', 2407.80);
INSERT INTO Province VALUES ('Jalisco', 'MEX', 2479.97);
INSERT INTO Province VALUES ('Tobasco', 'MEX', 5133.36);

--  MEX city insertions
INSERT INTO City VALUE ('Elota', 'Sinaloa', 'MEX', 90000);
INSERT INTO City VALUE ('Choix', 'Sinaloa', 'MEX', 115000);
INSERT INTO City VALUE ('Mocorito', 'Sinaloa', 'MEX', 150000);
INSERT INTO City VALUE ('Pihuamo', 'Jalisco', 'MEX', 90000);
INSERT INTO City VALUE ('Acatic', 'Jalisco', 'MEX', 115000);
INSERT INTO City VALUE ('San Marcos', 'Jalisco', 'MEX', 150000);
INSERT INTO City VALUE ('Villahermosa', 'Tobasco', 'MEX', 90000);
INSERT INTO City VALUE ('Cardenas', 'Tobasco', 'MEX', 115000);
INSERT INTO City VALUE ('Comalcalco', 'Tobasco', 'MEX', 150000);

--  CAN province insertions
INSERT INTO Province VALUES ('Quebec', 'CAN', 24860.80);
INSERT INTO Province VALUES ('British-Columbia', 'CAN', 2879.97);
INSERT INTO Province VALUES ('Manitoba', 'CAN', 51333.36);

--  CAN city insertions
INSERT INTO City VALUE ('Amos', 'Quebec', 'CAN', 90000);
INSERT INTO City VALUE ('Blainville', 'Quebec', 'CAN', 115000);
INSERT INTO City VALUE ('Lorraine', 'Quebec', 'CAN', 150000);
INSERT INTO City VALUE ('Delta', 'British-Columbia', 'CAN', 90000);
INSERT INTO City VALUE ('Vacouver', 'British-Columbia', 'CAN', 115000);
INSERT INTO City VALUE ('Rossland', 'British-Columbia', 'CAN', 150000);
INSERT INTO City VALUE ('Steinbach', 'Manitoba', 'CAN', 90000);
INSERT INTO City VALUE ('Thompson', 'Manitoba', 'CAN', 115000);
INSERT INTO City VALUE ('Winnipeg', 'Manitoba', 'CAN', 150000);

--  GUA province insertions
INSERT INTO Province VALUES ('Quetzaltenango', 'GUA', 24607.80);
INSERT INTO Province VALUES ('Escuintla', 'GUA', 2784.97);
INSERT INTO Province VALUES ('Zacapa', 'GUA', 5133.36);

--  GUA City insertions
INSERT INTO City VALUE ('Ostuncalco', 'Quetzaltenango', 'GUA', 90000);
INSERT INTO City VALUE ('Cantel', 'Quetzaltenango', 'GUA', 115000);
INSERT INTO City VALUE ('Zunil', 'Quetzaltenango', 'GUA', 150000);
INSERT INTO City VALUE ('Brito', 'Escuintla', 'GUA', 90000);
INSERT INTO City VALUE ('Iztapa', 'Escuintla', 'GUA', 115000);
INSERT INTO City VALUE ('Baul', 'Escuintla', 'GUA', 150000);
INSERT INTO City VALUE ('Capucal', 'Zacapa', 'GUA', 90000);
INSERT INTO City VALUE ('Caulotes', 'Zacapa', 'GUA', 115000);
INSERT INTO City VALUE ('Arenal', 'Zacapa', 'GUA', 150000);
```

```
--  Border table insertion
INSERT INTO Border VALUE ('USA', 'MEX', 3145);
INSERT INTO Border VALUE ('USA', 'CAN', 8891);
INSERT INTO Border VALUE ('MEX', 'GUA', 871);

-- Select statements (to print tables)
SELECT * FROM Country;
SELECT * FROM Province;
SELECT * FROM City;
SELECT * FROM Border;
```

```
/*****************************************************************
* NAME: Zachary Foteff
* CLASS: CPSC321
* DATE: 10/18/2021
* HOMEWORK: 5
* DESCRIPTION: Create and insert stemenets for constructing a database
*              containing information about Legos and Lego sets
*****************************************************************/
--  Drop table statements
DROP TABLE IF EXISTS SetCategories;
DROP TABLE IF EXISTS PartsList;
DROP TABLE IF EXISTS LegoSets;
DROP TABLE IF EXISTS Themes;
DROP TABLE IF EXISTS Categories;
DROP TABLE IF EXISTS Bricks;

--  Create table statements
CREATE TABLE Bricks (
    elem_id         INT(50) UNSIGNED NOT NULL,
    des_id          INT(50) UNSIGNED NOT NULL,
    brick_name      VARCHAR(50) NOT NULL,
    brick_color     VARCHAR(25) NOT NULL,
    price           DECIMAL(10, 2) UNSIGNED NOT NULL,
    PRIMARY KEY (elem_id, des_id)
);

CREATE TABLE Themes (
    --  Themes for a Lego set
    theme_name      VARCHAR(100) NOT NULL,
    year_start      YEAR,
    year_end        YEAR,
    license         VARCHAR(100),
    PRIMARY KEY (theme_name)
);

CREATE TABLE Categories (
    --  Categories of Lego sets
    category_name   VARCHAR(100) NOT NULL,
    age_range       VARCHAR(10),
    PRIMARY KEY (category_name)
);

CREATE TABLE LegoSets(
    --  Data representing a Lego set
    item_num        INT(50) UNSIGNED NOT NULL UNIQUE,
    set_name        VARCHAR(100) NOT NULL,
    theme_name      VARCHAR(100) NOT NULL,
    price           DECIMAL(10, 2) UNSIGNED NOT NULL,
    minifig_count   SMALLINT NOT NULL,
    vip_points      SMALLINT NOT NULL,
    width           SMALLINT,
    heigth          SMALLINT,
    depth           SMALLINT,
    prod_start_year YEAR,
    prod_end_year   YEAR,
    PRIMARY KEY (item_num),
    FOREIGN KEY (theme_name) REFERENCES Themes (theme_name)
);

CREATE TABLE SetCategories (
```

```
    --  Table to categories mapped with set ids
    item_num        INT(50) UNSIGNED NOT NULL,
    category_name   VARCHAR(100) NOT NULL,
    PRIMARY KEY (item_num, category_name),
    FOREIGN KEY (item_num) REFERENCES LegoSets (item_num),
    FOREIGN KEY (category_name) REFERENCES Categories (category_name)
);

CREATE TABLE PartsList(
    --  A list of parts for a Lego set
    elem_id         INT(50) UNSIGNED NOT NULL,
    des_id          INT(50) UNSIGNED NOT NULL,
    item_num        INT(50) UNSIGNED NOT NULL,
    num_bricks      INT UNSIGNED,
    PRIMARY KEY (elem_id, des_id, item_num),
    FOREIGN KEY (elem_id, des_id) REFERENCES Bricks (elem_id, des_id),
    FOREIGN KEY (item_num) REFERENCES LegoSets (item_num)
);

--  Insert statements
--  Brick insertion statements
INSERT INTO Bricks VALUES (101, 11, '2x4', 'Red', "0.22");
INSERT INTO Bricks VALUES (102, 11, '2x2', 'Green', "0.20");
INSERT INTO Bricks VALUES (101, 12, '2x4', 'Blue', "0.22");
INSERT INTO Bricks VALUES (103, 11, '4x16 Flat', 'Grey', "0.82");
INSERT INTO Bricks VALUES (104, 11, '1x4', 'Yellow', "0.32");
INSERT INTO Bricks VALUES (105, 11, '8x8 Flat', 'Black', "1.95");

--  Themes insertion statements
INSERT INTO Themes VALUES ("Star Wars", 2000, 2025, "Walt Disney Inc.");
INSERT INTO Themes VALUES ("Pirates of the Caribbean", 2004, 2025, "Walt Disney
Inc.");
INSERT INTO Themes VALUES ("Lego City", NULL, NULL, NULL);
INSERT INTO Themes VALUES ("Real-World", NULL, NULL, "Various");

--  Categories insertion statements
INSERT INTO Categories VALUES ("Pirates", "8-18+");
INSERT INTO Categories VALUES ("Outer Space", "4-18+");
INSERT INTO Categories VALUES ("Movies", "4-18+");
INSERT INTO Categories VALUES ("City", "4-18+");
INSERT INTO Categories VALUES ("Real-world", "4-18+");

--  Lego Set insertion statements
INSERT INTO LegoSets VALUES (1001, "Jabba's Palace", "Star Wars", 120.00, 8, 350
, 12, 6, 8, 2002, 2004);
INSERT INTO LegoSets VALUES (1002, "Space Shuttle", "Real-World", 85.50, 4, 400,
 8, 3, 3, 1998, 2002);
INSERT INTO LegoSets VALUES (1003, "Captain Jack's Ship", "Pirates of the Caribb
ean", 75.00, 6, 350, 8, 16, 6, 2004, 2008);
INSERT INTO LegoSets VALUES (1004, "Gas Station", "Lego City", 60.00, 4, 300, 14
, 12, 12, 2008, 2012);
INSERT INTO LegoSets VALUES (1005, "Lego City Minifig Pack", "Lego City", 15.00,
 2, 75, NULL, NULL, NULL, 2008, 2012);

--  Set categories insertion statements
INSERT INTO SetCategories VALUES (1001, "Movies");
INSERT INTO SetCategories VALUES (1001, "Outer Space");
INSERT INTO SetCategories VALUES (1002, "Outer Space");
INSERT INTO SetCategories VALUES (1002, "Real-world");
INSERT INTO SetCategories VALUES (1003, "Pirates");
```

```
INSERT INTO SetCategories VALUES (1003, "Movies");
INSERT INTO SetCategories VALUES (1004, "City");
INSERT INTO SetCategories VALUES (1004, "Real-world");
INSERT INTO SetCategories VALUES (1005, "City");
INSERT INTO SetCategories VALUES (1005, "Real-world");

--  Part list insertion statements
INSERT INTO PartsList VALUES (101, 11, 1001, 75);
INSERT INTO PartsList VALUES (101, 12, 1001, 45);
INSERT INTO PartsList VALUES (102, 11, 1001, 905);
INSERT INTO PartsList VALUES (103, 11, 1001, 75);
INSERT INTO PartsList VALUES (104, 11, 1001, 100);
INSERT INTO PartsList VALUES (105, 11, 1001, 185);

--  Select statements (to print tables)
SELECT * FROM Bricks;
SELECT * FROM LegoSets;
SELECT * FROM Categories;
SELECT * FROM SetCategories;
SELECT * FROM Themes;
SELECT * FROM PartsList;
```