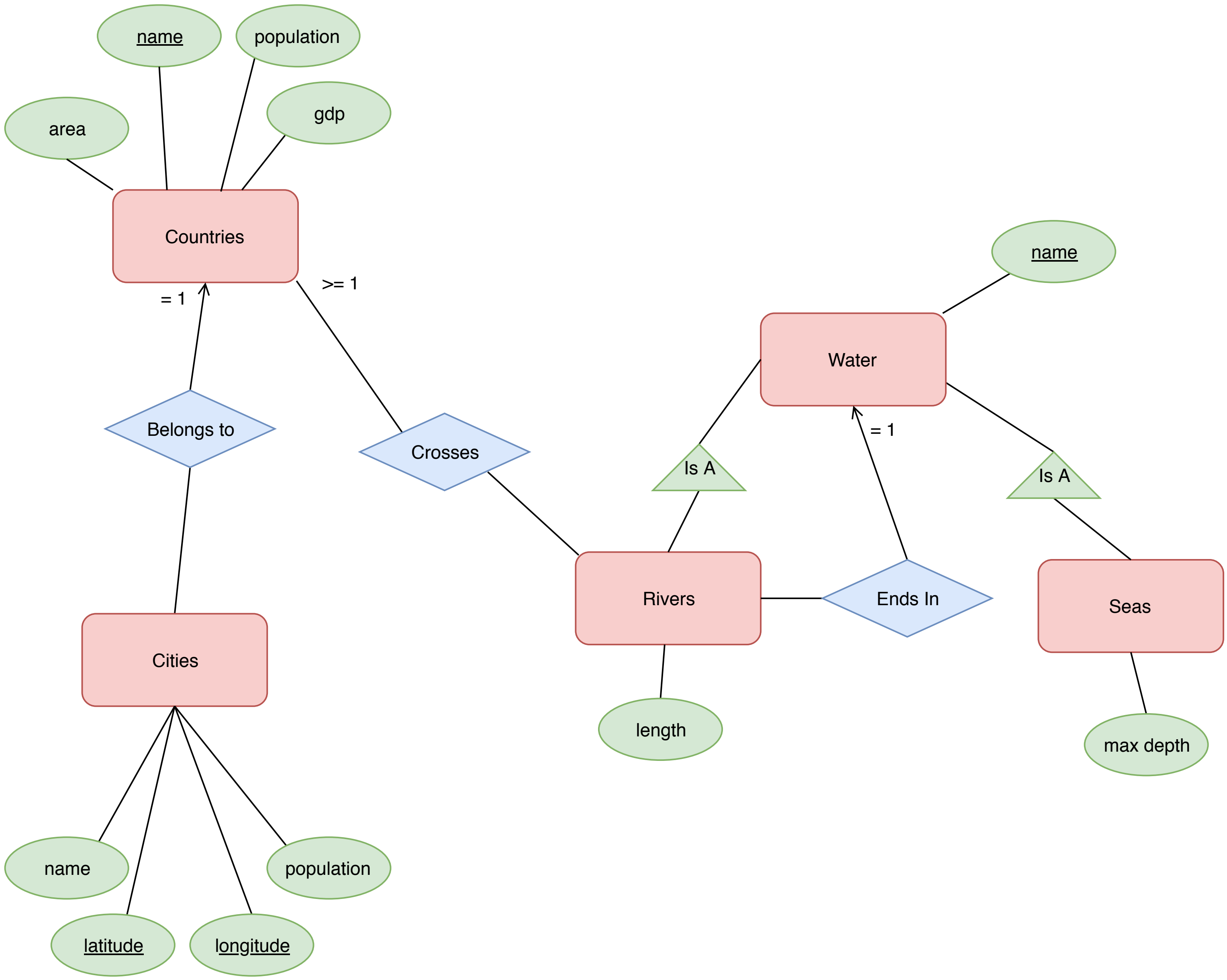
Part 1.1: Design an E/R diagram for geography that contains the following kinds of objects or entities together with the listed attributes.



Part 1.2a: Translate the diagram above by writing the SQL CREATE TABLE statements to represent this E/R diagram.

(I have included another file — part1_textversion.txt— which is a text file with these CREATE TABLE statements in my submission as well in case you need to copy/paste)

```
-- part 1.2

CREATE TABLE InsuranceCo(
        name TEXT,
        phone int,

        PRIMARY KEY (name)
);

CREATE TABLE ProfessionalDriver(
        ssn int,
        medicalHistory TEXT,

        PRIMARY KEY (ssn)
);

CREATE TABLE NonProfessionalDriver(
        ssn int,
        PRIMARY KEY (ssn)
);

CREATE TABLE Driver(
        ssn int,
        driverID int,
        FOREIGN KEY (ssn) references NonProfessionalDriver(ssn),
        FOREIGN KEY (ssn) references ProfessionalDriver(ssn),

        PRIMARY KEY (ssn)
);

CREATE TABLE Person(
        ssn int,
        name TEXT,
        FOREIGN KEY (ssn) references Driver(ssn),

        PRIMARY KEY (ssn)
);
```

1.2a continued

```
CREATE TABLE Truck(
        licensePlate varchar(10),
        capacity int,
        ssn int,
        FOREIGN KEY (ssn) references ProfessionalDriver(ssn),

        PRIMARY KEY (licensePlate)
);

CREATE TABLE Car(
        licensePlate varchar(10),
        make TEXT,

        PRIMARY KEY (licensePlate)
);

CREATE TABLE Vehicle(
        licensePlate varchar(10),
        year int,
        maxLiability float,
        insuranceCompany varchar(50),
        ssn int,
        FOREIGN KEY (licensePlate) references Car(licensePlate),
        FOREIGN KEY (licensePlate) references Truck(licensePlate),
        FOREIGN KEY (insuranceCompany) references InsuranceCo(name),
        FOREIGN KEY (ssn) references Person(ssn),

        PRIMARY KEY (licensePlate)
);

CREATE TABLE Drives(
        licensePlate varchar(10),
        ssn int,
        FOREIGN KEY (ssn) references NonProfessionalDriver(ssn),
        FOREIGN KEY (licensePlate) references Car(licensePlate),

        PRIMARY KEY (licensePlate, ssn)
);
```

Part 1.2b:
Which relation in your relational schema represents the relationship "insures" in the E/R diagram and why is that your representation?

Part 1.2c:
Compare the representation of the relationships "drives" and "operates" in your schema, and explain why they are different.

The Vehicle table represents the relationship "insures" by containing the attributes maxLiability and the name of the insurance company. Because the "insures" relation is one to many, each entry in the vehicle table is already uniquely associated to a maxLiability and an insurance company (or none at all, in which case NULL would probably be stored in these fields), so creating another table to store this information would be redundant.

Drives is a many to many relationship, so each car is not necessarily unqiuely associated to a NonProfessional Driver. In this case, we can create another table so that the data associated for each car does not have to be duplicated for every associated NonProfessionalDriver. By creating another table, we can duplicate only the information necessary to link each driver to their associated car, storing the bulk of the data in the Car and NonProfessionalDriver tables and cutting down on data redundancy.
Operates on the other hand is a many to one relationship. As discussed in the previous question, we can reduce data redundancy by collecting our foreign keys within the Truck table itself rather than creating a whole other table, as each Truck is already unqiuely associated to a given Professional Driver.

Part 1.3: Consider the following two relational schemas and sets of functional dependencies:

Part 1.3a: R(A,B,C,D,E) with functional dependencies D → B, CE → A.

```
---- part 1.3

R(A,B,C,D,E)
Arbitrarily choose nontrivial dependency D -> B
D+ = {D,B} is NOT = ABCDE; D not a superkey of R even though D -> B is a
nontrivial functional dependency. Thus, we decompose R on D -> B.
Y = {D,B} \ D = {B}
Z = {ABCDE} \ {DB} = {ACE}

decompose R into Y U {D} = P(D,B) and Z U {D} = Q(D,A,C,E)
P is in BCNF as D+ = {D,B} so D is a superkey and B+ = {B} so B->B is trivial
For relation Q, Choose nontrivial dependency CE -> A
{CE}+ = {A,C,E} is NOT = DACE (the entire set);
CE  is not superkey for P despite the fact that CE has a nontrivial dependency.
Split Q on CE -> A.
Y = {ACE} \ {CE} = {A}
Z = {DACE} \ {ACE} = {D}


decompose Q into Y U {CE} = S(A,C,E) and Z U {CE} = T(D,C,E)
P is in BCNF (all closure sets trivial or represent superkeys)
There exists no nontrivial dependencies on T, so T is naturally in BCNF
The only dependency on S is CE -> A, but we can see {CE}+ = ACE, so
CE is a superkey. Thus, S is in BCNF as well. Therefore R has been
completely decomposed.

R(ABCDE) -> P(D,B), S(A,C,E), T(D,C,E)
---
```

Part 1.3b: S(A,B,C,D,E) with functional dependencies A → E, BC → A, DE → B.

```
S(A,B,C,D,E)

consider the nontrivial dependency A -> E
A+ = {AE}, so A is not a superkey for S despite having a nontrivial dependency.
Thus, we will decompose S on A.
Y = {AE} \ {A} = {E}
Z = {ABCDE} \ {AE} = {BCD}


Decompose S into Y U {A} = P(A,E) and Z U {A} = Q(A,B,C,D)
P is in BCNF as the only dependency on P is A -> E, but this clearly implies
A is a superkey.
For the relation Q, consider the dependency BC -> A
{BC}+ = {A,B,C} is not Q; thus BC is not a superkey, and is a nontrivial dependency
Thus, we will decompose Q on BC
Y = {ABC} \ {BC} = {A}
Z = {ABCD} \ {ABC} = {D}

Decompose Q into Y U {BC} = R(A,B,C) and Z U {BC} = T(B,C, D)
R is in BCNF as the only nontrivial dependency on R is BC -> A, which clearly
implies that BC is a superkey to R.
T is in BCNF as there exists no nontrivial functional dependencies on BCD.
Therefore, we can decompose S into:

S(ABCDE) -> P(A,E), R(A,B,C), T(B,C,D)
```

Part 1.4:

A set of attributes X is called closed (with respect to a given set of functional dependencies) if X+=X. Consider a relation with schema R(A,B,C,D) and an unknown set of functional dependencies. For each closed attribute set below, give a set of functional dependencies that is consistent with it.

```
---- part 1.4

a)

Let there be no nontrivial functional dependencies on R(A,B,C,D).
Thus, our set of functional dependencies is simply:

all trivial dependencies
A -> A
B -> B
C -> C
D -> D
etc


Thus, the closure set for any set of attributes is itself (the set is closed)


b)

Functional Dependencies

A -> B
B -> C
C -> D
D -> A

In this case, as all attributes imply each other in a cyclical fashion, any
set of attributes X that is not empty contains all attributes in its closure set.
Naturally, this implies the only closed sets can be {} or all attributes {ABCD}

c)

Functional Dependencies

A -> B
B -> A
C -> D
D -> AC

In this case, any set that includes either C or D will contain all attributes, and
thus cannot be closed unless the set is the set of all attributes {ABCD}.
{A}+ = {AB} and {B}+ = {AB} so neither of these sets are closed, but {AB}+ = {AB}
so this set is closed. Again, the empty set {} is always closed and so is {ABCD}
```