University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: February 25 2025

Group Number: 44

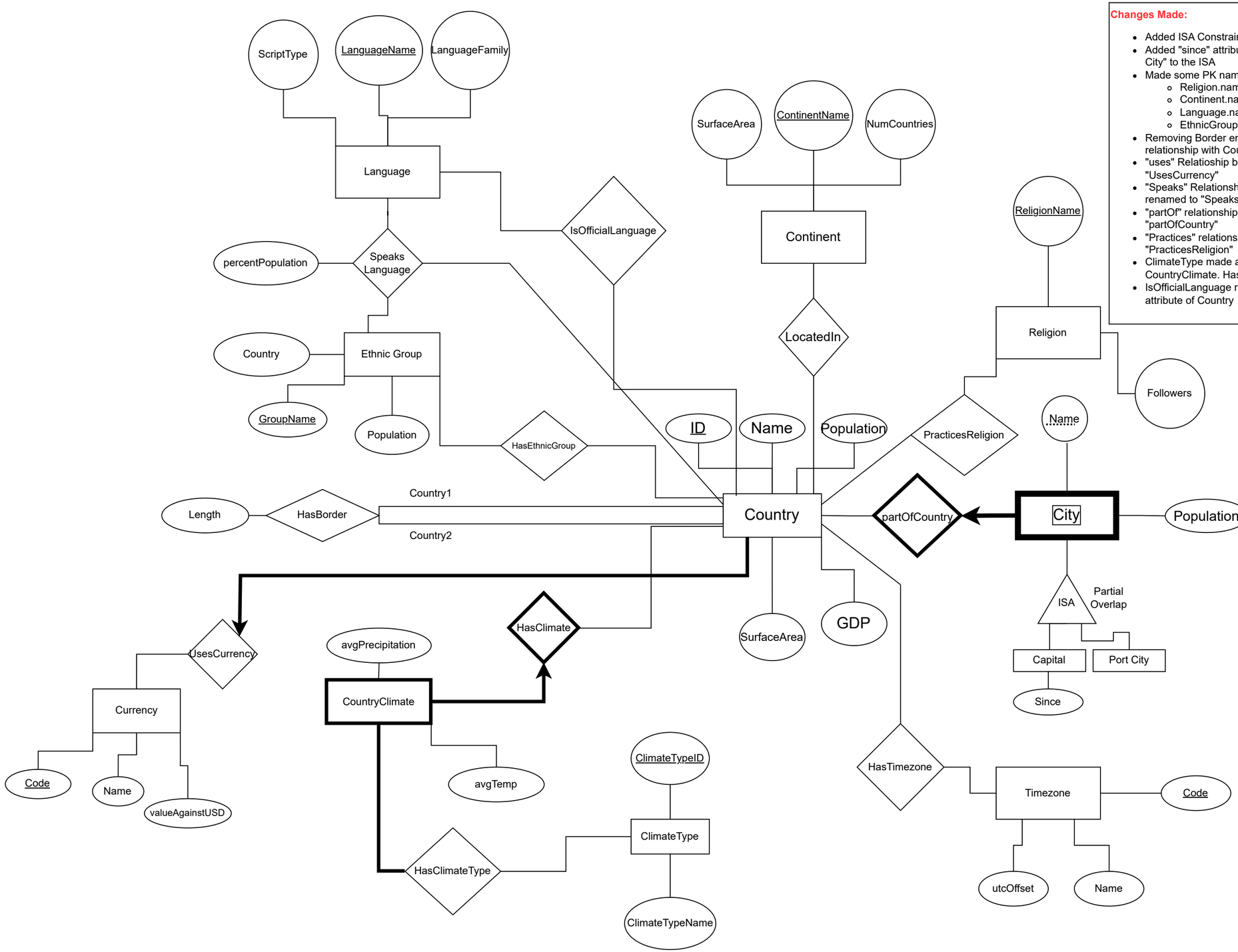
Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Danesh Rahmani	99189482	e2m2w	drahmani97@gmail.com
Vansh Chanana	59845008	w0f1n	vanshchanana2004@gmail.com
Hardit Singh	31250616	e7r5s	hardits925@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Changes Made:

- Added ISA Constraint for City ISA
- Added "since" attribute to Capital, and added a second entity "Port City" to the ISA
- Made some PK names more specific:
  - Religion.name --> Religion.religionName
  - Continent.name --> Continent.continentName
  - Language.name --> Language.languageName
  - EthnicGroup.name --> EthnicGroup.groupName
- Removing Border entity set, instead making HasBorder a recursive relationship with Country. Adding relationship attributes to HasBorder
- "uses" Relatioship between currency and country renamed to "UsesCurrency"
- "Speaks" Relationship between Language, EthnicGroup and Country renamed to "SpeaksLanguage"
- "partOf" relationship between city and country renamed to "partOfCountry"
- "Practices" relationship between Religion and country renamed to "PracticesReligion"
- ClimateType made an entity set rather than attribute of CountryClimate. HasClimateType relationship added.
- IsOfficialLanguage relationship added, removing OfficialLanguage as attribute of Country



Project Summary:

Our project involves modeling country data and demographics to allow users to interactively explore and understand the relationships between countries and their various attributes. We model many different aspects of this domain, including countries, cities, continents, borders, languages, ethnic groups, religion and currency. Users will eventually be able to view comprehensive details about any given country, as well as being able to query on any given criteria, via an interactive user interface.

Schema:

EntitySet or Relationship	**Corresponding Schema**	**Candidate Keys (Non PK are listed)**	**Other Constraints**
Continent	Continent(ContinentName: char[13], NumCountries: int, SurfaceArea: int)	{NumCountries, SurfaceArea}	CHECK NumCountries and SurfaceArea are positive values
Religion	Religion(ReligionName: char[20], Followers: int)	N/A	Followers' DEFAULT value is 0 if it is unknown. CHECK that Followers is a positive value.
Language	Language(LanguageName: char[20], ScriptType: char[20], LanguageFamily: char[20])	N/A	
Ethnic Group	EthnicGroup(GroupName: char[20], Country: char[31], Population: int)	N/A	Population DEFAULT value is 0 if it is unknown
Currency	Currency(Code: char[3], Name: char[20], valueAgainstUSD: int)	Name	Name is Unique
Timezone	Timezone(Code: char[3], Name: char[20], utcOffset: char[6])	{Name, utcOffset}	utcOffset is NOT NULL

Country Climate	CountryClimate(**CountryID:** int, avgTemp: int, avgPrecipitation: int)	N/A	CHECK that avgPrecipitation is not negative
Climate Type	ClimateType(ClimateTypeID: int, ClimateTypeName: char[20])	ClimateTypeName	ClimateTypeName is NOT NULL
City	City(CityName**:** char[168], **CountryID:** int, population: int)	N/A	CountryID is NOT NULL. CHECK that population is not negative
City ISA	Capital(**CountryID:** int, **CityName**: char[168], since: int) PortCity(**CountryID:** int, **CityName: char[168]**)	N/A	N/A
Country	Country(ID: int, Name: char[20], Population: int, GDP: int, SurfaceArea: int, **CurrencyCode:** char[3]) (CurrencyCode cannot be null)	Name	Name is UNIQUE, CurrencyCode is NOT NULL. *Population >= 0, GDP >= 0, SurfaceArea >= 0*
IsOfficial Language	IsOfficialLanguage(**CountryID:** int, **LanguageName**: char[20])	N/A	N/A
Has Timezone	HasTimezone(**CountryID**: int, **TimezoneCode:** char[5])	N/A	N/A
HasBorder	HasBorder(**Country1ID:** int, **Country2ID**: int, Length: int)	N/A	CHECK Country1ID !== Country2ID
HasEthnic Group	HasEthnicGroup(**CountryID**: int, **EthnicGroupName:** char[20])	N/A	N/A
Has Climate Type	HasClimateType(**ClimateTypeID**: int, **CountryID**: int)	N/A	N/A
Speaks Language	SpeaksLanguage(**LanguageName**: char[20], **CountryID:** int,	N/A	Percent Population

	<u>EthnicGroupName</u>: char[20], percentPopulation: int)		DEFAULT value is 0 if unknown
LocatedIn	LocatedIn(**<u>CountryID:</u>** int, **<u>ContinentName:</u>** char[13])	N/A	N/A
Practices Religion	PracticesReligion(**<u>CountryID:</u>** int, **<u>ReligionName:</u>** char[20])	N/A	N/A

Functional Dependencies:

Continent	● ContinentName → NumCountries, SurfaceArea ● NumCountries, SurfaceArea → ContinentName
Religion	● ReligionName → Followers
Language	● LanguageName → ScriptType, LanguageFamily
EthnicGroup	● GroupName → Country, Population
Currency	● Code → Name, valueAgainstUSD ● Name → Code, valueAgainstUSD
Timezone	● Code → Name, utcOffset ● Name, utcOffset → Code
CountryClimate	● CountryID → avgTemp, avgPrecipitation
ClimateType	● ClimateTypeID → ClimateTypeName ● ClimateTypeName → ClimateTypeID
City	● CityName, CountryID → Population
Capital (City ISA)	● CountryID, CityName → Since
Country	● ID → Name, Population, GDP, SurfaceArea, CurrencyCode ● CurrencyCode → CurrencyName, valueAgainstUSD ● Name → ID, Population, GDP, SurfaceArea, CurrencyCode
HasBorder	● Country1ID, Country2ID → Length
SpeaksLanguage	● LanguageName, CountryID, EthnicGroupName → percentPopulation
LocatedIn	● CountryID → ContinentName

Normalization:

After analyzing the functional dependencies for each relation, we determined that all of our relations are already in BCNF (Boyce-Codd Normal Form), which is the normal form we're using.

As such for every non-trivial functional dependency X → Y in each relation, X is a superkey.

Since all relations are already in BCNF, they eliminate all data redundancy issues that normal forms address, no decomposition is necessary for any relation, and all functional dependencies are preserved.

For each relation, we've verified that all non-trivial functional dependencies are based solely on candidate keys, and have shown this documentation below.

1. Continent:

- **Schema:** Continent(ContinentName: char[13], NumCountries: int, SurfaceArea: int)
- **Primary Key:** ContinentName
- **Candidate Key:** {NumCountries, SurfaceArea}
- **Functional Dependencies:**
    - ContinentName → NumCountries, SurfaceArea
    - NumCountries, SurfaceArea → ContinentName

The table is in BCNF because all functional dependencies have a superkey on the left-hand side. ContinentName is the primary key, and NumCountries, SurfaceArea is a candidate key.

---

2. Religion:

- **Schema:** Religion(ReligionName: char[20], Followers: int)
- **Primary Key:** ReligionName
- **Candidate Key:** None
- **Functional Dependencies:**
    - ReligionName → Followers

The table is in BCNF because the functional dependency has a superkey on the left-hand side. ReligionName is the primary key.

---

3. Language:

- **Schema:** Language(LanguageName: char[20], ScriptType: char[20], LanguageFamily: char[20])
- **Primary Key:** LanguageName
- **Candidate Key:** None
- **Functional Dependencies:**
    - LanguageName → ScriptType, LanguageFamily

The table is in BCNF because the functional dependency has a superkey on the left-hand side. LanguageName is the primary key.

---

4. EthnicGroup:

- **Schema:** EthnicGroup(GroupName: char[20], Country: char[31], Population: int)
- **Primary Key:** GroupName
- **Candidate Key:** None
- **Functional Dependencies:**
    - GroupName → Country, Population

The table is in BCNF because the functional dependency has a superkey on the left-hand side. GroupName is the primary key.

---

5. Currency:

- **Schema:** Currency(Code: char[3], Name: char[20], valueAgainstUSD: int)
- **Primary Key:** Code
- **Candidate Key:** Name
- **Functional Dependencies:**
    - Code → Name, valueAgainstUSD
    - Name → Code, valueAgainstUSD

The table is in BCNF because all functional dependencies have a superkey on the left-hand side. Code is the primary key, and Name is a candidate key.

---

6. Timezone:

- **Schema:** Timezone(Code: char[3], Name: char[20], utcOffset: char[6])
- **Primary Key:** Code
- **Candidate Key:** {Name, utcOffset}
- **Functional Dependencies:**
    - Code → Name, utcOffset
    - Name, utcOffset → Code

The table is in BCNF because all functional dependencies have a superkey on the left-hand side. Code is the primary key, and Name, utcOffset is a candidate key.

---

7. CountryClimate:

- **Schema:** CountryClimate(**CountryID:** int, avgTemp: int, avgPrecipitation: int)
- **Primary Key:** CountryID
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID → avgTemp, avgPrecipitation

The table is in BCNF because the FD has a superkey on the LHS. CountryID is the PK

---

8. ClimateType:

- **Schema:** ClimateType(ClimateTypeID: int, ClimateTypeName: char[20])
- **Primary Key:** ClimateTypeID
- **Candidate Key:** ClimateTypeName
- **Functional Dependencies:**
    - ClimateTypeID → ClimateTypeName
    - ClimateTypeName → ClimateTypeID

The table is in BCNF because all FDs have a superkey on the LHS.

---

9. City:

- **Schema:** City(CityName: char[168], **CountryID:** int, population: int)
- **Primary Key:** {CityName, CountryID}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CityName, CountryID → Population

The table is in BCNF because the functional dependency has a superkey on the left-hand side. {CityName, CountryID} is the primary key.

---

10. Capital

- **Schema:** Capital(**CountryID:** int, **CityName: char[168]**, since: int)
- **Primary Key:** {CountryID, CityName}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID, CityName → since

The table is in BCNF because all FDs have a superkey on the LHS.

---

11. PortCity

- **Schema:** PortCity(**CountryID:** int, **CityName: char[168]**)
- **Primary Key:** {CountryID, CityName}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID, CityName → {}

The table is in BCNF because there are no other non-trivial FDs other than the PK.

---

12. Country:

- **Schema:** Country(ID: int, Name: char[20], Population: int, GDP: int, SurfaceArea: int, **CurrencyCode:** char[3])
- **Primary Key:** ID
- **Candidate Key:** Name
- **Functional Dependencies:**
    - ID → Name, Population, GDP, SurfaceArea, CurrencyCode
    - Name → ID, Population, GDP, SurfaceArea, CurrencyCode

The table is in BCNF because all functional dependencies have a superkey on the left-hand side. ID is the primary key, and Name is a candidate key.

---

13. IsOfficialLanguage:

- **Schema:** IsOfficialLanguage(**CountryID:** int, **LanguageName**: char[20])
- **Primary Key:** {CountryID, LanguageName}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID, LanguageName → {}

The table is in BCNF because there are no other non-trivial FDs other than the PK.

---

14. HasTimezone:

- **Schema:** HasTimezone(**CountryID**: int, **TimezoneCode:** char[5])
- **Primary Key:** {CountryID, TimeZoneCode}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID, TimeZoneCode→ {}

The table is in BCNF because there are no other non-trivial FDs other than the PK.

---

15. HasBorder:

- **Schema:** HasBorder(**Country1ID:** int, **Country2ID**: int, Length: int)
- **Primary Key:** {Country1ID, Country2ID}
- **Candidate Key:** None
- **Functional Dependencies:**
    - Country1ID, Country2ID → Length

The table is in **BCNF** because the functional dependency has a superkey on the left-hand side. Country1ID, Country2ID is the primary key.

---

16. HasEthnicGroup:

- **Schema:** HasEthnicGroup(**CountryID**: int, **EthnicGroupName:** char[20])
- **Primary Key:** {CountryID, EthnicGroupName}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID, EthnicGroupName → {}

The table is in BCNF because there are no other non-trivial FDs other than the PK.

---

17. HasClimateType:

- **Schema:** HasClimateType(**ClimateTypeID**: int, **CountryID**: int)
- **Primary Key:** {ClimateTypeID, CountryID}
- **Candidate Key:** None
- **Functional Dependencies:**
    - ClimateTypeID, CountryID → {}

The table is in BCNF because there are no other non-trivial FDs other than the PK.

---

18. SpeaksLanguage:

- **Schema:** SpeaksLanguage(**LanguageName**: char[20], **CountryID:** int, **EthnicGroupName**: char[20], percentPopulation: int)
- **Primary Key:** {LanguageName, CountryID, EthnicGroupName}
- **Candidate Key:** None
- **Functional Dependencies:**
    - LanguageName, CountryID, EthnicGroupName → percentPopulation

The table is in BCNF because the functional dependency has a superkey on the left-hand side. LanguageName, CountryID, EthnicGroupName is the primary key.

---

19. LocatedIn:

- **Schema:** LocatedIn(**CountryID:** int, **ContinentName:** char[13])
- **Primary Key:** {CountryID, ContinentName}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID, ContinentName → {}

The table is in BCNF because there are no other non-trivial FDs other than the PK.

---

20. PracticesReligion:

- **Schema:** PracticesReligion(**CountryID:** int, **ReligionName:** char[20])
- **Primary Key:** {CountryID, ReligionName}
- **Candidate Key:** None
- **Functional Dependencies:**
    - CountryID, ReligionName→ {}

The table is in BCNF because there are no other non-trivial FDs other than the PK.

SQL DDL Statements:

***For all the places we mention "ON UPDATE CASCADE", we know Oracle won't support it.**

```
CREATE TABLE Continent (
        ContinentName VARCHAR PRIMARY KEY,
        numCountries INT,
        surfaceArea INT,
        CHECK (numCountries > 0 AND surfaceArea > 0)
);

CREATE TABLE Religion (
        ReligionName VARCHAR PRIMARY KEY,
        Followers INT DEFAULT 0,
        CHECK (Followers ≥ 0)
);

CREATE TABLE Language (
        LanguageName VARCHAR PRIMARY KEY,
        ScriptType VARCHAR.
        LanguageFamily VARCHAR,
);


CREATE TABLE EthnicGroup (
        GroupName VARCHAR PRIMARY KEY,
        Country VARCHAR,
        Population INT DEFAULT 0,
        CHECK (Population ≥ 0)
);

CREATE TABLE Currency (
        Code CHAR(3) PRIMARY KEY,
        Name VARCHAR UNIQUE,
        ValueAgainstUSD INT
);

CREATE TABLE TimeZone (
```

```
    Code VARCHAR PRIMARY KEY,
    Name INT,
    UTCOffset VARCHAR NOT NULL
);


CREATE TABLE CountryClimate (
    CountryID INT PRIMARY KEY,
    avgTemp INT,
    avgPrecipitation INT,
    CHECK (avgPrecipitation ≥ 0)
);

CREATE TABLE ClimateType (
    ClimateTypeID INT PRIMARY KEY,
    ClimateTypeName VARCHAR(20) UNIQUE
);

CREATE TABLE City (
    CityName VARCHAR,
    CountryID INT,
    Population INT,
    CHECK (Population ≥ 0)
    PRIMARY KEY (CityName, CountryID),
    FOREIGN KEY (CountryID) REFERENCES Country(ID)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE
);
```

If a CountryID is deleted, all cities associated with that country are also deleted.
If a CountryID is updated, the corresponding City entries are updated accordingly.

```
CREATE TABLE Country (
    ID INT PRIMARY KEY,
    Name VARCHAR UNIQUE,
    Population INT,
    GDP INT,
    SurfaceArea INT,
    CurrencyCode CHAR(3) NOT NULL DEFAULT 'USD',
    CHECK (Population >= 0 AND GDP >= 0 AND SurfaceArea >= 0),
    FOREIGN KEY (CurrencyCode) REFERENCES Currency (Code)
```

ON DELETE SET DEFAULT
ON UPDATE CASCADE
);

If a CurrencyCode is deleted, the Country's CurrencyCode is set to the default value (USD).

If a CurrencyCode is updated, the corresponding Country entries are updated accordingly.

CREATE TABLE *SpeaksLanguage* (
LanguageName VARCHAR,
CountryID INT,
EthnicGroupName VARCHAR,
percentPopulation INT DEFAULT 0*,*
PRIMARY KEY (LanguageName, CountryID, EthnicGroupName),
FOREIGN KEY (LanguageName) REFERENCES Language(LanguageName)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (EthnicGroupName) REFERENCES EthnicGroup*(GroupName)*
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (CountryID) REFERENCES Country(ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
);

If LanguageName, CountryID, or EthnicGroupName is deleted, the corresponding SpeaksLanguage entry is deleted.

If LanguageName, CountryID, or EthnicGroupName is updated, the corresponding SpeaksLanguage entry is updated.

CREATE TABLE *HasBorder* (
Country1ID INT,
Country2ID INT,
Length INT,
PRIMARY KEY (Country1ID, Country2ID),
FOREIGN KEY (Country1ID) REFERENCES Country(ID)

```
                                    ON DELETE CASCADE
                                    ON UPDATE CASCADE,
    FOREIGN KEY (Country2ID) REFERENCES Country(ID)
                                    ON DELETE CASCADE
                                    ON UPDATE CASCADE,
    CHECK (Country1ID != Country2ID)
);
```

If either Country1ID or Country2ID is deleted, the corresponding border entry is deleted.
If either Country1ID or Country2ID is updated, the corresponding border entry is updated.
A country cannot have a border with itself (Country1ID != Country2ID).

```
CREATE TABLE Capital (
    CountryID INT PRIMARY KEY,
    CityName VARCHAR,
    Since INT,
    FOREIGN KEY (CountryID, CityName) REFERENCES City(CountryID, CityName)
                                        ON DELETE CASCADE
                                        ON UPDATE CASCADE
);
```

If CountryID or CityName is deleted, the corresponding Capital entry is deleted.
If CountryID or CityName is updated, the corresponding Capital entry is updated.

```
CREATE TABLE PortCity (
    CountryID INT,
    CityName VARCHAR,
    PRIMARY KEY (CountryID, CityName),
    FOREIGN KEY (CountryID, CityName) REFERENCES City(CountryID, CityName)
                                        ON DELETE CASCADE
                                        ON UPDATE CASCADE
);
```

If CountryID or CityName is deleted, the corresponding PortCity entry is deleted.
If CountryID or CityName is updated, the corresponding PortCity entry is updated.

CREATE TABLE *IsOfficialLanguage (*
CountryID INT,
LanguageName VARCHAR,
PRIMARY KEY (CountryID, LanguageName),
FOREIGN KEY (CountryID) REFERENCES Country(ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (LanguageName) REFERENCES Language(LanguageName)
ON DELETE CASCADE
ON UPDATE CASCADE
);

If CountryID or LanguageName is deleted, the corresponding IsOfficialLanguage entry is deleted.
If CountryID or LanguageName is updated, the corresponding IsOfficialLanguage entry is updated.

CREATE TABLE *HasTimezone (*
CountryID INT,
TimezoneCode CHAR(5),
PRIMARY KEY (CountryID, TimezoneCode),
FOREIGN KEY (CountryID) REFERENCES Country(ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (TimezoneCode) REFERENCES Timezone(Code)
ON DELETE CASCADE
ON UPDATE CASCADE
);

If CountryID or TimezoneCode is deleted, the corresponding HasTimezone entry is deleted.
If CountryID or TimezoneCode is updated, the corresponding HasTimezone entry is updated.

CREATE TABLE *HasEthnicGroup (*
CountryID INT,
EthnicGroupName VARCHAR,
PRIMARY KEY (CountryID, EthnicGroupName),
FOREIGN KEY (CountryID) REFERENCES Country(ID)
ON DELETE CASCADE
ON UPDATE CASCADE,

FOREIGN KEY (EthnicGroupName) REFERENCES EthnicGroup(GroupName)
ON DELETE CASCADE
ON UPDATE CASCADE

);

If CountryID or EthnicGroupName is deleted, the corresponding HasEthnicGroup entry is deleted.
If CountryID or EthnicGroupName is updated, the corresponding HasEthnicGroup entry is updated.

CREATE TABLE *HasClimateType (*
ClimateTypeID INT,
CountryID INT,
PRIMARY KEY (ClimateTypeID, CountryID),
FOREIGN KEY (ClimateTypeID) REFERENCES ClimateType(ClimateTypeID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (CountryID) REFERENCES Country(ID)
ON DELETE CASCADE
ON UPDATE CASCADE

);

If ClimateTypeID or CountryID is deleted, the corresponding HasClimateType entry is deleted.
If ClimateTypeID or CountryID is updated, the corresponding HasClimateType entry is updated.

CREATE TABLE *LocatedIn (*
CountryID INT,
ContinentName VARCHAR,
PRIMARY KEY (ContinentName, CountryID),
FOREIGN KEY (CountryID) REFERENCES Country(ID)
ON DELETE CASCADE
ON UPDATE CASCADE,
FOREIGN KEY (ContinentName) REFERENCES Continent(ContinentName)
ON DELETE RESTRICT
ON UPDATE CASCADE

);

If CountryID is deleted, the corresponding LocatedIn entry is deleted.
If ContinentName is deleted, the operation is restricted to prevent orphaned countries.
If CountryID or ContinentName is updated, the corresponding LocatedIn entry is updated.

```
CREATE TABLE PracticesReligion (
        CountryID INT,
        ReligionName VARCHAR,
        PRIMARY KEY (CountryID, ReligionName),
        FOREIGN KEY (CountryID) REFERENCES Country(ID)
                                        ON DELETE CASCADE
                                        ON UPDATE CASCADE,
        FOREIGN KEY (ReligionName) REFERENCES Religion(ReligionName)
                                        ON DELETE CASCADE
                                        ON UPDATE CASCADE
);
```

If CountryID or ReligionName is deleted, the corresponding PracticesReligion entry is deleted.
If CountryID or ReligionName is updated, the corresponding PracticesReligion entry is updated.

INSERT Statements:

```
INSERT INTO Continent VALUES (‘Asia’, 47, 44614000);
INSERT INTO Continent VALUES (‘North America’, 23, 24230000);
INSERT INTO Continent VALUES (‘South America’, 12, 17814000);
INSERT INTO Continent VALUES (‘Europe’, 43, 10000000);
INSERT INTO Continent VALUES (‘Australia’, 14, 7688287);
INSERT INTO Continent VALUES (‘Africa’, 54, 30365000);
INSERT INTO Continent VALUES (‘Antarctica’, 0, 14200000);

INSERT INTO Religion VALUES (‘Christianity’, 2400000000);
INSERT INTO Religion VALUES (‘Islam’, 1900000000);
INSERT INTO Religion VALUES (‘Hinduism’, 1200000000);
INSERT INTO Religion VALUES (‘Buddhism’, 500000000);
INSERT INTO Religion VALUES (‘Judaism’, 15000000);

INSERT INTO Language VALUES ('English', 'Latin', 'Indo-European');
INSERT INTO Language VALUES ('Spanish', 'Latin', 'Indo-European');
INSERT INTO Language VALUES ('French', 'Latin', 'Indo-European');
INSERT INTO Language VALUES ('Mandarin', 'Chinese', 'Sino-Tibetan');
INSERT INTO Language VALUES ('Hindi', 'Devanagari', 'Indo-Aryan');

INSERT INTO EthnicGroup VALUES ('Caucasian', 'Canada', 25000000);
INSERT INTO EthnicGroup VALUES ('Hispanic', 'USA', 60000000);
INSERT INTO EthnicGroup VALUES ('Han Chinese', 'China', 1200000000);
INSERT INTO EthnicGroup VALUES ('Indigenous', 'Australia', 800000);
INSERT INTO EthnicGroup VALUES ('Punjabi', 'India', 30000000);

INSERT INTO Currency VALUES ('CAD', 'Canadian Dollar', 0.69);
INSERT INTO Currency VALUES ('USD', 'US Dollar', 1);
INSERT INTO Currency VALUES ('EUR', 'Euro', 1.04);
INSERT INTO Currency VALUES ('INR', 'Indian Rupee', 0.011);
INSERT INTO Currency VALUES ('JPY', 'Japanese Yen', 0.0066);
INSERT INTO Currency VALUES ('IRR', 'Iranian Rial', 0.000024);

INSERT INTO Timezone VALUES ('PST', 'Pacific Standard Time', '-08:00');
INSERT INTO Timezone VALUES ('EST', 'Eastern Standard Time', '-05:00');
INSERT INTO Timezone VALUES ('GMT', 'Greenwich Mean Time', '+00:00');
INSERT INTO Timezone VALUES ('CET', 'Central European Time', '+01:00');
INSERT INTO Timezone VALUES ('IST', 'Indian Standard Time', '+05:30');
```

```
INSERT INTO CountryClimate VALUES (1, -2.89, 524.57);
INSERT INTO CountryClimate VALUES (2, 11.5 , 762);
INSERT INTO CountryClimate VALUES (3, 10.88, 870);
INSERT INTO CountryClimate VALUES (4, 25.02, 1161.45);
INSERT INTO CountryClimate VALUES (5, 12.99, 1500);


INSERT INTO ClimateType VALUES (6, 'Tropical');
INSERT INTO ClimateType VALUES (7, 'Arid');
INSERT INTO ClimateType VALUES (8, 'Temperate');
INSERT INTO ClimateType VALUES (9, 'Continental');
INSERT INTO ClimateType VALUES (10, 'Polar');

INSERT INTO City VALUES ('Vancouver', 1, 706012);
INSERT INTO City VALUES ('Toronto', 1, 3000000);
INSERT INTO City VALUES ('New York', 2, 8258000);
INSERT INTO City VALUES ('Berlin', 3, 3432000);
INSERT INTO City VALUES ('Tokyo', 5, 14180000);

INSERT INTO Capital VALUES (1, 'Ottawa', 1867);
INSERT INTO Capital VALUES (2, 'Washington D.C.', 1790);
INSERT INTO Capital VALUES (3, 'Berlin', 1871);
INSERT INTO Capital VALUES (4, 'New Delhi', 1911);
INSERT INTO Capital VALUES (5, 'Tokyo', 1869);

INSERT INTO PortCity VALUES (1, 'Vancouver');
INSERT INTO PortCity VALUES (2, 'Toronto');
INSERT INTO PortCity VALUES (2, 'New York');
INSERT INTO PortCity VALUES (3, 'Hamburg');
INSERT INTO PortCity VALUES (5, 'Yokohama');

INSERT INTO Country VALUES (1, 'Canada', 38000000, 1800000, 9985000, 'CAD');
INSERT INTO Country VALUES (2, 'USA', 331000000, 22000000, 9834000, 'USD');
INSERT INTO Country VALUES (3, 'Germany', 83000000, 4000000, 357000, 'EUR');
INSERT INTO Country VALUES (4, 'India', 1380000000, 2900000, 3287000, 'INR');
INSERT INTO Country VALUES (5, 'Japan', 126000000, 5000000, 377900, 'JPY');

INSERT INTO IsOfficialLanguage VALUES (1, 'English');
INSERT INTO IsOfficialLanguage VALUES (2, 'Spanish');
INSERT INTO IsOfficialLanguage VALUES (3, 'German');
```

```
INSERT INTO IsOfficialLanguage VALUES (4, 'Hindi');
INSERT INTO IsOfficialLanguage VALUES (5, 'Japanese');

INSERT INTO HasTimezone VALUES (1, 'PST');
INSERT INTO HasTimezone VALUES (2, 'EST');
INSERT INTO HasTimezone VALUES (3, 'CET');
INSERT INTO HasTimezone VALUES (4, 'IST');
INSERT INTO HasTimezone VALUES (5, 'GMT');

INSERT INTO HasBorder VALUES (1, 2, 8893);
INSERT INTO HasBorder VALUES (2, 3, 1900);
INSERT INTO HasBorder VALUES (3, 4, 3300);
INSERT INTO HasBorder VALUES (4, 5, 7000);
INSERT INTO HasBorder VALUES (5, 1, 1500);

INSERT INTO HasEthnicGroup VALUES (1, 'Caucasian');
INSERT INTO HasEthnicGroup VALUES (2, 'Hispanic');
INSERT INTO HasEthnicGroup VALUES (3, 'Han Chinese');
INSERT INTO HasEthnicGroup VALUES (4, 'Punjabi');
INSERT INTO HasEthnicGroup VALUES (5, 'Indigenous');

INSERT INTO HasClimateType VALUES (6, 1);
INSERT INTO HasClimateType VALUES (7, 2);
INSERT INTO HasClimateType VALUES (8, 3);
INSERT INTO HasClimateType VALUES (9, 4);
INSERT INTO HasClimateType VALUES (10, 5);

INSERT INTO SpeaksLanguage VALUES ('English', 1, 'Caucasian', 80);
INSERT INTO SpeaksLanguage VALUES ('Spanish', 2, 'Hispanic', 75);
INSERT INTO SpeaksLanguage VALUES ('Mandarin', 3, 'Han Chinese', 90);
INSERT INTO SpeaksLanguage VALUES ('Hindi', 4, 'Punjabi', 60);
INSERT INTO SpeaksLanguage VALUES ('Japanese', 5, 'Indigenous', 50);

INSERT INTO LocatedIn VALUES (1, 'North America');
INSERT INTO LocatedIn VALUES (2, 'North America');
INSERT INTO LocatedIn VALUES (3, 'Europe');
INSERT INTO LocatedIn VALUES (4, 'Asia');
INSERT INTO LocatedIn VALUES (5, 'Asia');

INSERT INTO PracticesReligion VALUES (1, 'Christianity');
```

```
INSERT INTO PracticesReligion VALUES (2, 'Christianity');
INSERT INTO PracticesReligion VALUES (3, 'Christianity');
INSERT INTO PracticesReligion VALUES (4, 'Hinduism');
INSERT INTO PracticesReligion VALUES (5, 'Shintoism');
```

AI Acknowledgment:

Used AI to confirm that existing relations were already in BCNF for Normalization. Also used to make the language for ON DELETE and ON UPDATE explanations more uniform and consistent.