Final Report

Members: Bryan Garza <bgarza1@csustan.edu>, Brett Martin <bmartin6@csustan.edu>, and Mario Muniz <mmuniz4@csustan.edu>.

World Countries Statistics
Group Name: Make Before Break

Domain: The domain of our database project will be various statistics relating to geopolitical entities. This will include statistics relating to murders, economic data such as GDP, population, executions, and similar data. However, we will not be including explicitly geographical data such as location, coordinates, elevation, and other physical aspects. We are focusing on politics, economics, and domestic data.

User-Group: Our intended user group is anyone that's interested in comparing and contrasting different country's metrics, including population, GPP, happiness index, size of territory, etcetera. They don't have to be experts in geography to understand our data, we want to make it accessible for anyone to be able to browse, sort, and query to be able to discover interesting relationships between different countries. This means that our application will be useful to a wide variety of people, such as economists, historians, political scientists, politicians, students, and the curious minded.

Application Specification: Our completed application will be a web application, with an interface that allows people to compose specific queries, perhaps with natural language processing. An example of a query would be "countries with a population greater than 40M with a GDP in the top 25%, and an area smaller than 800,000 km." We envision it akin to Facebook's Graph Search, which allows users to type in queries in basic english and return relevant information. (such as, friends that are male that like pizza that also live in Modesto) We'll be able to display a list of all countries that match the criteria and sort them based on the different parameters.

What will be modeled and what will not: As stated before, we are going to be modelling data and statistics that relate to politics, economics, and domestic affairs. We will not be modeling geographic data about countries / states / nations, such as elevation, land mass, or water mass.

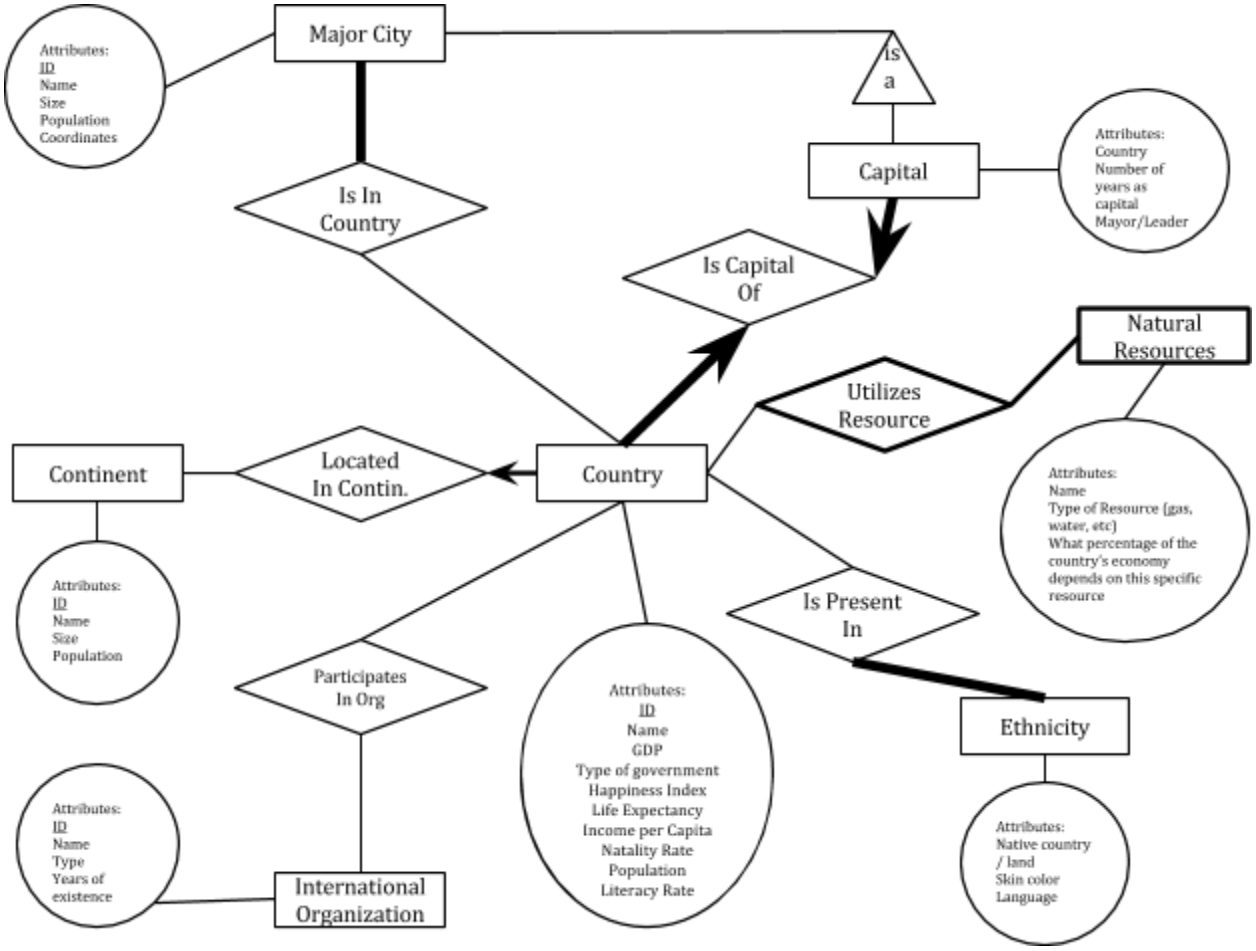
Ground rules:

1. Everyone will do an equal amount of work.

2. If someone doesn't have time to meet, they must work on the project independently that day and still contribute.
3. Stay on the same page; all members must agree in the spec / direction of the feature before implementing it
4. Give constructive criticism, if a team member doesn't agree with something, say why in a way that helps instead of discourages.
5. We will not put off the project until the end but work on it in manageable pieces, allocating time on a weekly basis to ensure quality.
6. Stay realistic and don't try to take on too much or add features that we don't have the ability to implement with our current programming knowledge.
7. Come prepared to our meetings, so that we aren't slowed down or rendered useless by unpreparedness.
8. Mario will keep the team focused (he's good at this)
9. Bryan will try not to get distracted or go crazy with ideas, while still brainstorming for interesting directions the project will take.
10. Brett will manage the project at the macro-level, keeping the team working in-sync with the larger project goals.

All three of us have experience writing web applications, using a variety of web-based technologies, and we expect the project to be very doable considering our collective knowledge.

Value added facilities: The database has the potential to be used as a valuable resource for many different individuals. For example, political science professors like to cite data and explore relationships between different traits of nations.The database that we will construct has the potential to be used to quickly compare countries, and query relationships between different traits, which would be an incredibly useful tool for anyone exploring politics or studying political science in a university classroom. It would also be a useful tool for a professor to use during a lecture.



Explanations:

Each of the categories we will be implementing in our database are represented by rectangles above. Every single one of the categories is related to something else:

Countries have the most relationships. Countries are each in a specific **continent**, and these continents can have any number of countries. Some countries, but not all, may participate in **organizations**, such as the United Nations, NATO, or ISIS. All countries must have a **capital city**. Capital cities are also a type of city, so they inherit attributes of a regular city. We will only be including major cities in this database. Cities must also be in *one specific* country. **Natural resources** are utilized by countries, and any number of countries can make use of any number of resources. These resources can be of many different types, including gas deposit, a specific element, gold, etcetera. Different countries can have the same type of resource.

Constraints:

There are only seven continents, but this limitation cannot be illustrated by the database itself. Countries can also only have one capital.

Make Before Break: Project Part 3

World Countries Stats Domain

Domain: The domain of our database project will be various statistics relating to geopolitical entities. This will include statistics relating to murders, economic data such as GDP, population, executions, and similar data. However, we will not be including explicitly geographical data such as location, coordinates, elevation, and other physical aspects. We are focusing on politics, economics, and domestic data.

SQL

```
CREATE TABLE country
( ID INTEGER,
GDP INTEGER,
name CHAR(40),
hapIndx INTEGER,
typeGovt CHAR(40),
incomePerCapita INTEGER,
population INTEGER
PRIMARY KEY (ID) )

CREATE TABLE isInCountry
( cityID INTEGER,
countryID INTEGER,
FOREIGN KEY (cityID) REFERENCES
majorCity (ID),
FOREIGN KEY (countryID)
REFERENCES country (ID),
PRIMARY KEY (cityID, countryID)
)

CREATE TABLE isCapitalOf
( countryID INTEGER NOT NULL,
capitalID INTEGER NOT NULL,
FOREIGN KEY (capitalID)
REFERENCES capital (ID),
FOREIGN KEY (countryID)
REFERENCES country (ID),
PRIMARY KEY (cityID, countryID)
)

CREATE TABLE majorCity
( ID INTEGER,
Size INTEGER,
name CHAR(40),
population INTEGER,
PRIMARY KEY (ID) )
```

```
CREATE TABLE continent
( ID INTEGER,
Size INTEGER,
name CHAR(40),
population INTEGER,
PRIMARY KEY (id) )

CREATE TABLE
utilizesNaturalResources
( ID INTEGER NOT NULL,
typeOfResource CHAR(40),
name CHAR(40),
PRIMARY KEY (id, name)
FOREIGN KEY (ID) REFERENCES
country,
     ON DELETE CASCADE)

CREATE TABLE isPresentIn
( name CHAR(40),
countryID INTEGER,
FOREIGN KEY (name) REFERENCES
ethnicGroup,
FOREIGN KEY (countryID)
REFERENCES country (ID),
PRIMARY KEY (name, countryID) )

CREATE TABLE ethnicGroup
( skinColor CHAR(40),
nativeCountry CHAR(40),
language CHAR(40),
name CHAR(40),
PRIMARY KEY (name) )
```

```
CREATE TABLE capital
( ID INTEGER,
yearsAsCapital INTEGER,
PRIMARY KEY (ID)
FOREIGN KEY (ID) REFERENCES
majorCity (ID),
ON DELETE CASCADE )

CREATE TABLE locatedInContinent
( countryID INTEGER NOT NULL,
continentID INTEGER NOT NULL,
FOREIGN KEY (continentID)
REFERENCES continent (ID),
FOREIGN KEY (countryID)
REFERENCES country (ID),
PRIMARY KEY (countryID)
)
```

```
CREATE TABLE participatesIn
( countryID INTEGER,
organizationID INTEGER,
FOREIGN KEY (organizationID)
REFERENCES organization (ID),
FOREIGN KEY (countryID)
REFERENCES country (ID),
PRIMARY KEY (name, countryID) )

CREATE TABLE organization
( name CHAR(40),
type CHAR(40),
yearsInExistence INTEGER,
ID INTEGER,
PRIMARY KEY (ID) )
```

Part 4 Revised:

Section 1

```
CREATE TABLE country
( countryID INTEGER,
GDP INTEGER,
name CHAR(40),
hapIndx INTEGER,
typeGovt CHAR(40),
incomePerCapita INTEGER,
population INTEGER
PRIMARY KEY (countryID) )

CREATE TABLE isInCountry
( cityID INTEGER,
countryID INTEGER,
FOREIGN KEY (cityID) REFERENCES majorCity (majorCityID),
FOREIGN KEY (countryID) REFERENCES country (countryID),
PRIMARY KEY (cityID, countryID))

CREATE TABLE isCapitalOf
```

```
( countryID INTEGER NOT NULL,
capitalID INTEGER NOT NULL,
FOREIGN KEY (capitalID) REFERENCES capital (ID),
FOREIGN KEY (countryID) REFERENCES country (ID),
PRIMARY KEY (capitalID, countryID))

CREATE TABLE majorCity
( majorCityID INTEGER NOT NULL,
size INTEGER,
name CHAR(40),
population INTEGER,
PRIMARY KEY (majorCityID))

CREATE TABLE capital
( capitalID INTEGER,
yearMadeCapital INTEGER,
FOREIGN KEY (capitalID) REFERENCES majorCity (majorCityID)
ON DELETE CASCADE
PRIMARY KEY (capitalID))




CREATE TABLE locatedInContinent
( countryID INTEGER NOT NULL,
continentName CHAR(40),
FOREIGN KEY (continentName) REFERENCES continent
(continentName),
FOREIGN KEY (countryID) REFERENCES country (continentName),
PRIMARY KEY (countryID))


CREATE TABLE continent
( Size INTEGER,
continentName CHAR(40),
population INTEGER,
PRIMARY KEY (continentName) )

CREATE TABLE utilizesNaturalResources
(
utilizesNaturalResourcesID INTEGER NOT NULL,
typeOfResource CHAR(40),
name CHAR(40),
```

```
countryID INTEGER,
FOREIGN KEY (countryID) REFERENCES country (countryID)
ON DELETE CASCADE,
PRIMARY KEY (utilizesNaturalResourcesID, name)
)

CREATE TABLE isPresentIn
( name CHAR(40),
countryID INTEGER,
FOREIGN KEY (name) REFERENCES ethnicGroup (name),
FOREIGN KEY (countryID) REFERENCES country (countryID),
PRIMARY KEY (name, countryID) )

CREATE TABLE ethnicGroup
( majorReligion CHAR(40),
nativeCountry CHAR(40),
language CHAR(40),
name CHAR(40),
PRIMARY KEY (name) )

CREATE TABLE participatesIn
( countryID INTEGER,
organizationID INTEGER,
FOREIGN KEY (organizationID) REFERENCES organization
(organizationID),
FOREIGN KEY (countryID) REFERENCES country (countryID),
PRIMARY KEY (organizationID, countryID) )



CREATE TABLE organization
( name CHAR(40),
type CHAR(40),
yearFounded INTEGER,
organizationID INTEGER,
PRIMARY KEY (organizationID) )
```

Section 2

```
CREATE TABLE country
( countryID INTEGER,
GDP INTEGER,
name CHAR(40),
hapIndx INTEGER,
typeGovt CHAR(40),
```

```
incomePerCapita INTEGER,
population INTEGER
PRIMARY KEY (countryID) )

CREATE TABLE isInCountry
( cityID INTEGER,
countryID INTEGER,
FOREIGN KEY (cityID) REFERENCES majorCity (majorCityID),
FOREIGN KEY (countryID) REFERENCES country (countryID),
PRIMARY KEY (cityID, countryID))

CREATE TABLE isCapitalOf
( countryID INTEGER NOT NULL,
capitalID INTEGER NOT NULL,
FOREIGN KEY (capitalID) REFERENCES capital (ID),
FOREIGN KEY (countryID) REFERENCES country (ID),
PRIMARY KEY (capitalID, countryID))

CREATE TABLE majorCity
( majorCityID INTEGER NOT NULL,
size INTEGER,
name CHAR(40),
population INTEGER,
PRIMARY KEY (majorCityID))

CREATE TABLE capital
( capitalID INTEGER,
yearMadeCapital INTEGER,
FOREIGN KEY (capitalID) REFERENCES majorCity (majorCityID)
ON DELETE CASCADE
PRIMARY KEY (capitalID))

CREATE TABLE locatedInContinent
( countryID INTEGER NOT NULL,
continentName CHAR(40),
FOREIGN KEY (continentName) REFERENCES continent
(continentName),
FOREIGN KEY (countryID) REFERENCES country (continentName),
PRIMARY KEY (countryID))

CREATE TABLE continent
( Size INTEGER,
continentName CHAR(40),
population INTEGER,
PRIMARY KEY (continentName) )
```

```
CREATE TABLE utilizesNaturalResources
(
utilizesNaturalResourcesID INTEGER NOT NULL,
typeOfResource CHAR(40),
name CHAR(40),
countryID INTEGER,
FOREIGN KEY (countryID) REFERENCES country (countryID)
ON DELETE CASCADE,
PRIMARY KEY (utilizesNaturalResourcesID, name)
)

CREATE TABLE isPresentIn
( name CHAR(40),
countryID INTEGER,
FOREIGN KEY (name) REFERENCES ethnicGroup (name),
FOREIGN KEY (countryID) REFERENCES country (countryID),
PRIMARY KEY (name, countryID) )

CREATE TABLE ethnicGroup
( majorReligion CHAR(40),
nativeCountry CHAR(40),
language CHAR(40),
name CHAR(40),
PRIMARY KEY (name) )

CREATE TABLE participatesIn
( countryID INTEGER,
organizationID INTEGER,
FOREIGN KEY (organizationID) REFERENCES organization
(organizationID),
FOREIGN KEY (countryID) REFERENCES country (countryID),
PRIMARY KEY (organizationID, countryID) )



CREATE TABLE organization
( name CHAR(40),
type CHAR(40),
yearFounded INTEGER,
organizationID INTEGER,
PRIMARY KEY (organizationID) )
```

Section 2

---

country table
countryID -> GDP
countryID -> name
countryID -> hapIndx
countryID -> typeGovt
countryID -> population

isInCountry table
(all trivial)

isCapitalOf table
(all trivial)

majorCity table
majorCityID -> Size
majorCityID -> Name
majorCityID -> population

capital table
capitalID -> yearMadeCapital
capitalID -> majorCityID

locatedInContinent table
countryID -> continentName

continent table
continentName -> Size
continentName -> population [from continent]

utilizesNaturalResources table
utilizesNaturalResourcesID, name -> typeOfResource
utilizesNaturalResourcesID, typeOfResource -> name

isPresentIn table
(all trivial)

ethnicGroup table
name -> majorReligion
name -> language
name -> nativeCountry

participatesIn table
(all trivial)

organization table
organizationID -> name
organizationID -> type
organizationID -> yearFounded

Section 3
---
country table
countryID -> GDP
countryID -> name
countryID -> hapIndx
countryID -> typeGovt
countryID -> incomePerCapita
countryID -> population

The country table is in BCNF because name, population, and typeGovt are all trivial FDs, and countryID is a superkey.

The country table is in 3NF because every BCNF relation is also in 3NF.

isInCountry table
(all trivial)

The isInCountry table is in BCNF because there are no non-trivial functional dependencies.

The isInCountry table is in 3NF because every BCNF relation is also in 3NF.

isCapitalOf table
(all trivial)

The isCapitalOf table is in BCNF because there are no non-trivial functional dependencies.

The isCapitalOf table is in 3NF because every BCNF relation is also in 3NF.

majorCity table
majorCityID -> Size
majorCityID -> Name
majorCityID -> population

majorCity is in BCNF because majorCityID is a superkey for the relation.

Major City table is in 3NF because every BCNF relation is also in 3NF.

capital table
capitalID -> yearMadeCapital
capitalID -> majorCityID

Capital table is in BCNF because majorCityID and yearMadeCapital are both trivial FDs, and capitalID is a superkey.

Capital table is in 3NF because every BCNF relation is also in 3NF.

locatedInContinent table
(all trivial)

The locatedInContinent table is in BCNF because there are no non-trivial functional dependencies.

The locatedInContinent table is in 3NF because every BCNF relation is also in 3NF.

continent table
continentName -> Size
continentName -> population [from continent]

It is in BCNF because continentName is a superkey for the relation.

It is in 3NF because every BCNF relation is also in 3NF.

utilizesNaturalResources table
utilizesNaturalResourcesID, name -> typeOfResource
utilizesNaturalResourcesID, typeOfResource -> name

It is in BCNF because utilizesNaturalResourcesID is a superkey for the relation.

It is in 3NF because every BCNF relation is also in 3NF.

isPresentIn table
(all trivial)

The isPresentIn table is in BCNF because there are no non-trivial functional dependencies.

The isPresentIn table is in 3NF because every BCNF relation is also in 3NF.

ethnicGroup table
name -> majorReligion
name -> language
name -> nativeCountry

The ethnicGroup table is in BCNF because name is a superkey for the relation.

The ethnicGroup table is in 3NF because every BCNF relation is also in 3NF.

participatesIn table
(all trivial)

The participatesIn table is in BCNF because there are no non-trivial functional dependencies.

The participatesIn table is in 3NF because every BCNF relation is also in 3NF.

organization table
organizationID -> name
organizationID -> type
organizationID -> yearFounded

The organization table is in BCNF because organizationID is a superkey for the relation.

The organization table is in 3NF because every BCNF relation is also in 3NF.

We did not change any relations. We considered changing the relationship between country and continent for countries that belong to more than one continent (for example, Egypt and Turkey), but after inspecting our tables and their relationships, we concluded that it already worked out fine.

Section 3

Number of tuples for each relation:

Relation	**Number of Tuples**
capital	106
continent	7
country	106
ethnic group	60
isCapitalOf	106
isInCountry	106
isPresentIn	60
locatedInContinent	106
majorCity	106

organization	66
participatesIn	66
utilizesNaturalResource	66

Section 4

Sample queries:

```
INSERT INTO `bgarza`.`isCapitalOf` (`countryID`, `capitalID`) VALUES ('53',
'30');

DELETE FROM `bgarza`.`locatedInContinent` WHERE
`locatedincontinent`.`countryID` = 53;

UPDATE `bgarza`.`utilizesNaturalResources` SET `typeOfResource` = 'Aliquam',
`name` = 'hymenaeos' WHERE
`utilizesnaturalresources`.`utilizesNaturalResourcesID` = 23 AND
`utilizesnaturalresources`.`name` = 'hymenaeos.';
```

Section 5

Sources for our data:

The vast majority of our factually accurate data came from the CIA World Factbook. (https://www.cia.gov/library/publications/the-world-factbook/)

Some of the real data concerning international organizations came from Wikipedia. (http://en.wikipedia.org/wiki/List_of_intergovernmental_organizations)

The rest of the fake data that we cooked up was generated by Generate Data. (http://www.generatedata.com/)

No special script was written by us to produce data for this database.

Regarding the factually accurate data:
Special care was taken to make sure that data was input into the database that corresponded with the other tuples that the data was related to. For example, if the city London was imported into the Major Cities table, care was taken to reference that in the Capitals table, as well as making sure that the isCapitalOf table referenced the correct capital city and country. Careful care was taken to ensure that each country had its own unique capital.

Regarding the randomly generated fake data:
Care was taken to ensure that the data did not violate certain constraints. An example of this would be the fact that a capital city cannot be the capital of two different countries. In some cases, such as ethnic groups being present in countries, duplication did not matter (an ethnic group can be present in more than one country), so the data was directly inputted. Certain other real-world constraints were considered. For example, no country is present in Antarctica.

To prevent the violation of key constraint violations with the generated data, each column was incremented numerically, and then input into the database in order. So, for example, randomly generated major cities were given incremental number IDs. These same IDs were then used when generating capital cities. Therefore, when the data for isInCountry was input, there were no conflicts. Each capital city was located in the country that it was a capital of.

Continent only has 7 tuples because only 7 continents exist.

Section 6

Example tuples:
capital:

capitalID	yearMadeCapital
1	1790
2	1000
3	1325
4	1949
5	1866
6	1990

continent:

Size	continentName	population
30000000	Africa	1100000000
14000000	Antarctica	0
44500000	Asia	4000000000
7600000	Australia	23000000
10000000	Europe	742000000
24000000	North America	528000000
17000000	South America	387000000

country:

countryID	GDP	name	hapIndx	typeGovt	incomePerCapita	population
1	17000000	United States	37	Parliamentary democracy	54800	318892103
2	2435000	United Kingdom	48	Constitutional Monarchy	37700	63742977
3	2143000	Mexico	53	Federal Republic	17900	120286655
4	17630000	China	45	Communist state	12900	1355692576
5	1579000	Canada	44	Parliamentary democracy	44500	34834841
6	3621000	Germany	47	Federal Republic	44700	80996685

ethnicGroup:

majorReligion	nativeCountry	language	name ▲ 1
magna.	Phasellus	scelerisque	a
felis,	auctor	Praesent	a,
tellus	gravida	leo.	adipiscing
orci.	ipsum	mauris	aliquet
vel	et	erat.	ante
lobortis	augue	magnis	blandit

isCapitalOf:

countryID	capitalID
1	1
2	2
3	3
4	4
5	5
6	6

isInCountry:

cityID	countryID
1	1
2	2
3	3
4	4
5	5
6	6

isPresentIn:

name	countryID
English	1
Germans	1
English	2
enim,	2
dapibus	3
Spaniards	3

locatedInContinent:

countryID ▲ 1	continentName
1	North America
2	Europe
3	North America
4	Asia
5	North America
6	Europe
7	Asia

majorCity:

majorCityID	size	name	population
1	68	Washington, D.C	658893
2	607	London	8308000
3	573	Mexico City	8851000
4	6487	Beijing	11510000
5	1073	Ottawa	883391
6	344	Berlin	3502000

Organization:

name	type	yearFounded	organizationID
North Atlantic Treaty Organization	Military Alliance	1949	1
World Trade Organization	International Trade Organization	1995	2
OPEC	Cartel	1961	3
Asia-Pacific Economic Cooperation	Economic forum	1989	4
European Investment Bank	International financial institution	1959	5
Council of Europe	Advisory international organization	1949	6

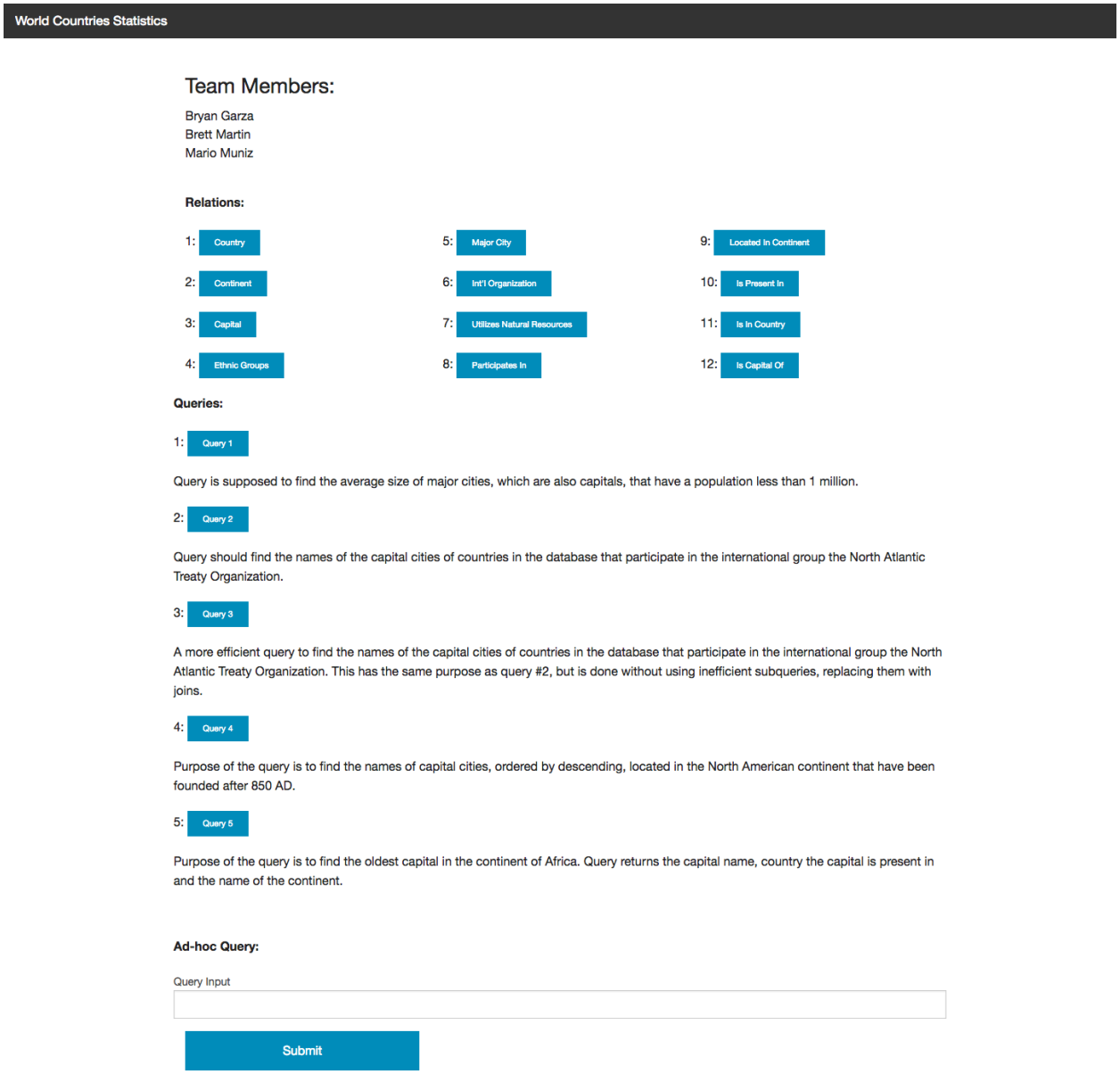
ParticipatesIn:

countryID	organizationID
1	1
1	4
1	67
2	1
2	9
3	94
4	4

utilizesNaturalResource:

utilizesNaturalResourcesID	typeOfResource	name	countryID
1	Industrial supply	Oil	1
2	Industrial supply	Oil	6
3	Industrial supply	Oil	5
4	Organic supply	Foodstuffs	2
5	Industrial supply	Oil	2
6	Element supply	Metal ores	4
7	tellus. Nunc	iaculis	55

Screenshots of our website
Part 7



This is what the home page of our web app looks like.

World Countries Statistics

ID (int)	GDP (int)	Name (text)	Happiness Index (int)	Type of Government (char)	Income per Capita (int)	Population (int)
1	17000000	United States	37	Parliamentary democracy	54800	318892103
2	2435000	United Kingdom	48	Constitutional Monarchy	37700	63742977
3	2143000	Mexico	53	Federal Republic	17900	120286655
4	17630000	China	45	Communist state	12900	1355692576
5	1579000	Canada	44	Parliamentary democracy	44500	34834841
6	3621000	Germany	47	Federal Republic	44700	80996685
7	5616561	Serbia	71	ut quam	41535	5111241
8	46452529	Bhutan	19	morbi tristique	16335	20406473
9	43281371	Guernsey	81	magna nec	93545	12697769
10	19122081	South Sudan	56	Aliquam rutrum	78518	13788061
11	48939715	Cape Verde	77	amet luctus	49261	20232426
12	42770243	Montserrat	0	ad litora	75087	4691838
13	14520774	Ethiopia	45	auctor, nunc	21192	20587794
14	32561803	Sri Lanka	29	In at	53709	9933667
15	36539195	Italy	60	urna. Nullam	39164	10234829

This is a screenshot of one of the relations, the first one (countries). There's more lines, this is only the first screenful.

World Countries Statistics

Original SQL Query

```
SELECT majorCity.name
FROM majorCity
JOIN capital
ON majorCity.majorCityID = capital.capitalID
JOIN isCapitalOf
ON capital.capitalID = isCapitalOf.capitalID
JOIN participatesIn
ON isCapitalOf.countryID = participatesIn.countryID
JOIN organization
ON participatesIn.organizationID = organization.organizationID
WHERE organization.name = 'North Atlantic Treaty Organization'
GROUP BY name
ORDER BY name ASC
```

QUERY RESULTS:

Name
Berlin
London
Ottawa
Washington, D.C

This is one of the queries, the 3rd one ("Query should find the names of the capital cities of countries in the database that participate in the international group the North Atlantic Treaty Organization").

Trying a simple ad-hoc query.

Member contributions:

Bryan (bgarza1@csustan.edu): Creating the report from all the previous sections of the project, and as well as revising Part 4.
Mario (bigmariobro@gmail.com): Worked on the PHP programming, and the website.
Brett (bmart95@gmail.com) : Brett also worked on the PHP and the website, as well as the world map.

There were no problems working together, on some parts of the project, one person would lead and work the most in, but in other sections, someone else would be the one to focus on and make sure got done.

Someone that deals with countries and their relationships to other world entities could definitely make us of our web app to get information about the world. The relationships between countries, continents, capitals, etc. are easy to understand. The ad-hoc query field is a good way of finding out answers to queries we have not already written. If there were a way to create an ad-hoc query and permanently add to the list of queries so it could be run again at a later time, that would be even more useful. Looking at http://datacatalog.worldbank.org/, another world database, we can see that it doesn't show any of the database details. If the SQL statements were abstracted out in our web app, as they are on this other site, then it would be less powerful because only what the creators of the site intended to query is shown. Users of our page can create many different types of queries just by understanding the relationships between tables of our database. The caveat that comes along with exposing the SQL to the user is that they need to be familiar with SQL for it to be useful to them, otherwise it will just be confusing. If they do understand it though, they can make use of the sample SQL statements that we have already created and base their new ad-hoc queries on those.