Project 3

Data from the World (Worth 50 Points)

Goal: In this project we will use data from the CIA (yes, that's the Central Intelligence Agency of the US government, ~~http://www.cia.gov/cia/publications/factbook/~~) World Fact Book as a basis for working with spreadsheets and relational databases. The data will give us a simple domain in which to do analysis, paralleling the sort of analysis you might do if you were curious about some topic. Notice that we're practicing with IT concepts, not learning data analysis, which is best done in a statistics class.

One more thing. This is a slightly different kind of application than is described in Chapter 16 of the textbook, in which a database is used in the context of a small business.

Background: The project assumes that you are familiar with spreadsheets (Labs 9,10, Chapter 13, Lecture 22), and databases (Lab 11, Chapters 14, 15, Lectures 23-25). The earlier part of the project (up to Step 6) is focused on spreadsheets only.

Files: Although you could get the original files from the CIA, the files found under "Classwork" on the class Web page have the right data *and they have been cleaned*. Data that is imported into a spreadsheet or database from an external source, even another spreadsheet or database, often requires fixing up in some ways. You've been spared most of that trouble here. The files for the class are:

`featureData.txt` – listing places that would be labeled on a map
`sourceData.txt` – giving country names, abbreviations, domains and area
`demogData.txt` – giving demographic data such as birthrates and population

All of the data is "tab-delimited," meaning that there are tabs between the data items and each line ends with a "return." The three files should be able to be imported into either a spreadsheet application or a database application, though we will usually import spreadsheets rather than text files into our databases.

Overview of Tasks: The first part is primarily based on the demographic information in the `demogData.txt` file. We perform a few simple operations, mostly sorting, on the data. Then, we decide to incorporate data from `sourceData.txt`. To do this, we will build a database table, and then convert it back to a spreadsheet. The point is to illustrate how the matching capabilities of join work. The second part is more focused on setting up a database and working with forms. The sourceData.txt and featureData.txt are combined to enhance the presentation. We build a variety of queries to illustrate the power of relational database concepts.

Tasks: Perform the following tasks.

Step 1) *Starting out.* After grabbing the files and placing each in your file space, import each into its own Excel workbook, and save each as its own file in `.xls` format. Review each and note the column headings. Look through the data to familiarize yourself with it.

Thinking to the incorporation of this data into a database, which fields contain unique data, i.e. which could be keys for this table?

Open a new MS Word document, and give it a heading "Demographic Analysis." Save the file as `project3.doc`. Put your name, etc. in the document to identify yourself. You will be building this document as you work through the project and you will turn in the file.

Step 2) *Ranking of the Demographic Data*. The columns in the demographic data are:

`country`, the standard CIA name for the country
`population`, from the most recent census
`fertility`, births per woman over her lifetime
`infant`, mortality, deaths per 1000 live births
`life` expectancy, expected age of death

There is one "bad" row in this spreadsheet. Remove the European Union from the table because it repeats the information of the individual countries.

Rank (sort) each column (except country) in descending order, which requires 4 separate sorts. Report (partial) results in your Demographic Analysis document by pasting the headings and the first five rows of each of the tables you produce. Thus the input (which you don't report) would be included in the document as

Input File

country	population	fertility	infant	lifeExpct
Afghanistan	29928987	6.75	163	42.9
Albania	3563112	2.04	22	77.24
Algeria	32531853	1.92	31	73
American Samoa	57881	3.25	9	75.84
Andorra	70549	1.29	4	83.51

This is a simple copy-from-the-spreadsheet and paste-in-the-document operation on the first six rows of the spreadsheet.

Step 3) *High Fertility Means Large Population?* It seems to make sense that the countries with the highest fertility rates would have the largest size. One simple test of this would be to compare the Top 10 on the fertility list and the Top 10 on the population list. Do so, and (by visual inspection) list in your Demographic Analysis the countries that are on both lists. (Be accurate!)

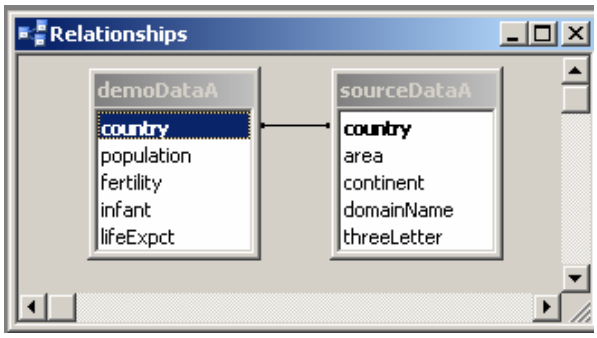
Step 4) *High Life Expectancy Implies Large Population?* Considering the disappointing overlap in Step (3), use a similar analysis to determine which countries are both in the Top 10 life expectancy and in the Top 10 on population.

Also, include in your document a bar chart of the Top 10 countries ranked by life expectancy.

Step 5) *Low Infant Mortality Implies Large Population?* So far we have not been very successful at discovering the cause of a large population. Perhaps it is low infant mortality. Try one more time, following Step (3), to use a similar analysis to determine which countries are both in the Lowest 10 Infant Mortality and the Top 10 on population. List the common elements in your document.

Step 6) *Add area data*. The demographic data doesn't tell us how large each of the countries is, so we are comparing huge countries like China with tiny countries like Macau. Also the world is a big place. We need more specific data. This is stored in the `sourceData.xls` file. We'd like to add the area and continent data, but there are slightly different items in each file, so the rows won't match even if they are sorted by country name. What to do? We will use the join feature of databases to build a new table. Proceed as follows:

- Open a blank Access database, called `project3.mdb`. Access is listed among the Microsoft Office applications.
- Import the `demogData.xls` as one table, and `sourceData.xls` as another table *having new names of your choosing*. To do this, be in the table window and click on New. You will be given a choice of wizards, and you should follow Import Data.
- Import data of `.xls` type extension, and work through the wizard steps; you may have to check "first row contains headings", because it does, and you want to choose as the primary key the `country` field for both tables.
- Once your two tables are imported, find the Relationships under Tools. You need to associate the `country` fields of the two tables so that the database system can match the items in the two tables. To do this, right click and select Show Tables, Add both tables, close that window, and then drag the `country` field of one table to the country field of the other. You will get a dialog box that you can ignore (if you click Create), and your relationships window should look like this:



- Next, move to the query window, and click on New. The wizard presents you with choices, and a Simple Query is the right pick. The wizard helps you easily build a new table using a query. Follow these steps …
- Move all of the demographic fields to the new table. Move only the `area` and the `continent` fields from the source table to the new table. Click through the rest of the wizard and notice your new table.

- Viola!
- Finally, because it's easier to do data analysis interactively with a spreadsheet, we will Export the table we just created as `project3.xls` file. (For file format, there are several choices; go for Excel 97-2002.)

In your Demographic Analysis document, include a copy of the SQL query that produced this table. From the query window, click on Design, and then under the View menu, find the SQL. Copy and paste the text into your document. Write in your own words what this query says to do to construct the table. (See Chapter 15, pp. 433-435 for more on the SQL query structure.)

Did you notice that one table had 216 rows and the other had 230 rows, but the combined table has only 214 rows. Why? That's all the rows that had a country match in each table. This is the powerful property of join that we are using.

Step 7) *Population density*. The population density is the number of people per square unit of area. To compute this value, open the spreadsheet from Step (6), and compute a new column that is `population/area`. Rank the countries from densest to sparsest, and report your Top 5 entries in the Demographic Analysis file that you're producing.

Step 8) *African Correlation*. With the continent field in the table, it is possible to restrict our attention to a single continent, and we'll pick Africa. Limiting interest only to those rows with an AF in the `continent` field, find the Top 10 highest infant mortality and the Top 10 highest fertility. Using visual inspection, find the countries in both lists and record them in your Demographic Analysis document.

Step 9) *Set up the last table.* Import the `featureData.txt` spreadsheet into the database. See Step (6), where the first two tables were imported.

The three tables, `sourceData`, `demogData` and `featureData` constitute the physical database of this project. From them we will build queries to create new tables, the logical database, containing the exact data of interest to us. We have already done that once in Step (6) above, when we combined the first two tables. Since in Step (6) you were asked to give your own names to `sourceData` and `demogData`, as well as a custom name to the query that you created to combine them, please rename the tables and query at this point (the query should be called `demography`) to make following these instructions easier.

Step 10) *Finding Oceania*. We begin with the `demography` query and modify it to a query that helps answer questions restricted to a single continent such as Oceania. Move to the query development window of the Access database. Select the `demography` query and click on Design. You will get the Query-By-Example (QBE) window that corresponds to your query. Go to View > SQL and recall how the demography query worked.

To select only those records that have an OC in the continent field (for Oceania, not Orange County), that is

> **Select** continent="OC" **From** demography

type ="OC" in the Criteria entry of the `continent` field. The Criteria constraint allows only records to be selected if their fields meet these criteria. Save the changes as `OCdemography`.

Next, inspect the SQL form of your query, and notice what has changed. Copy-paste this query into your Further Analysis document, and also include the first five data rows of the resulting table plus the column headings.

Step 11) *Home in the Tropics.* Familiarize yourself with the `featureData` table. Notice that it has latitude and longitude fields. Following what you have learned in Step (10) about SQL, make a query that lists the physical features in the Western Hemisphere between the Tropic of Cancer and the Tropic of Capricorn. These two Tropic lines are at about 23^{o} North and South. To do this, you

- Create a New query using the Simple Query wizard that selects the `place` field and all of the position fields, but not the `country`.
- Next, edit the SQL to add a `WHERE` clause (put it on its own line before the semicolon); a `WHERE` was visible in the SQL of Step (10).
- The `WHERE` should test that the latitude degrees (`latM`) are less than 23 and that the longitude (`EW`) is equal to "W" for Western Hemisphere. (Notice that it doesn't matter if the latitude is north or south). In SQL two conditions can be combined using `AND`.

- Check your work.

Paste your SQL command into your Further Analysis document and include the first five lines of the resulting table.

Extra credit. Technically, the tropics are defined to span the region from 23^o 27' N to 23^o 27' S. For extra credit, revise Step (11) so it includes this exact interval, assuming `latM` is "degrees" and `latS` is "minutes." [Hint: Considering testing that degrees are less than 24^o.)

Step 12) *Adding Capitals.* Next, noticing that the `featureData` table contains capital cities, we consider how to make a query that selects only capitals. Abstractly, the query we want is

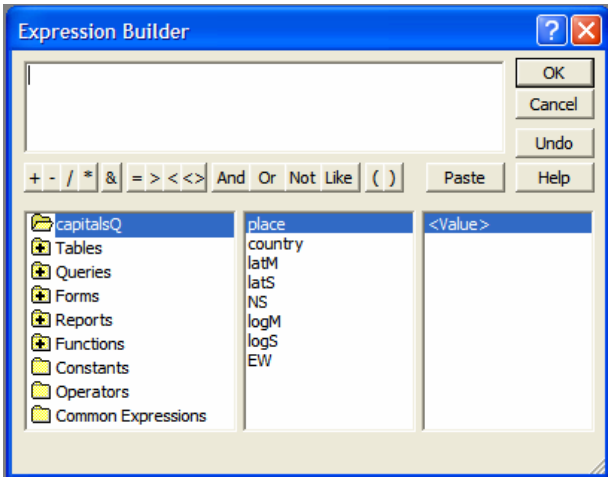
capitalsQ = **Select** place **contains** "(capital)" **From** featureData

where "contains" means that the letter string occurs in the (`place`) field. We need to program this because checking for a substring isn't part of the primitive facilities of the QBE type queries. So, begin by making a New query using the Simple Query wizard to make a query that selects all of the fields from `featureData`. Then edit the SQL to add a `WHERE` clause containing the statement

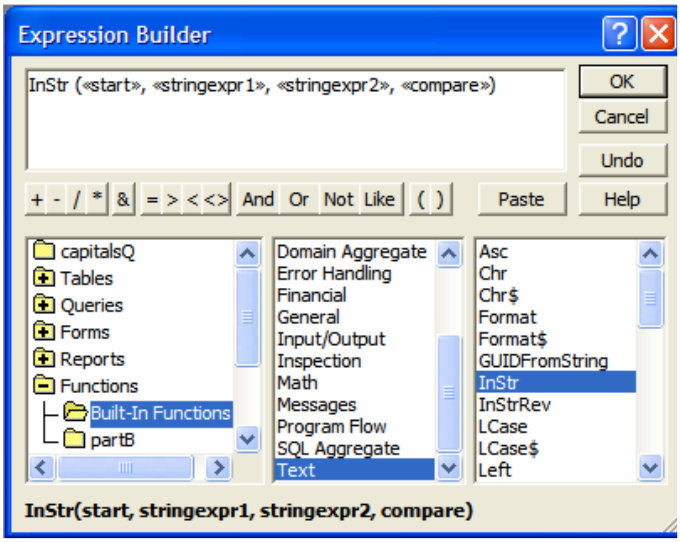
```
(InStr([place],"(capital)"))<>0
```

In English, this says, apply the "In String" function to the `place` field, looking for the substring `"(capital)"`, which if found, returns its position in the field, otherwise returns 0, and if the result is not equal to (< >) zero, then the record is selected. Call the query `capitalsQ` and check out the result. Include in your Further Analysis file the SQL query and the first five lines of the result.

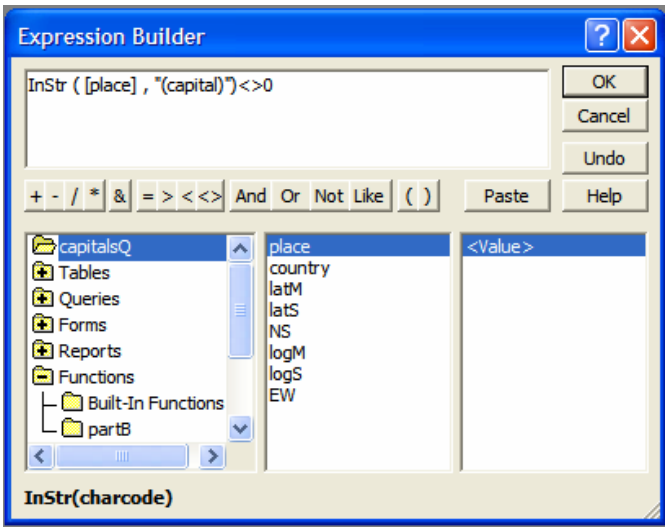
For future reference … If we don't know SQL very well, we can construct this computation by right clicking on the Criteria box of the `place` field, and selecting Build... . When the following window appears



navigate Functions > Built-In Functions > Text > InStr and click on Paste, which places the operation inn the window, and shows its structure at the bottom of the Builder.



This navigation is based entirely on guess work, because we may not know for sure that `InStr()` searches in a string, but it's a good guess that some function does, and if so, it's a built-in function. But, now we must learn what `InStr()` actually does, because we need to chose its arguments. To do that we check Help, and find out that it really needs only two arguments rather than the four shown. (The function can do other stuff, we don't want.) To reference the place field, we navigate Queries > capitalsQ > place and then Paste the <value> into position of the first two arguments. We replace the second two arguments with the constant string `"(capital)"`. We need to check to see if the outcome is not equal to zero by clicking on the <> and adding a zero. Finally, we click OK and the result is achieved.



Step 13) *Countries and capitals*. We now use join to combine countries and capitals. Begin by going to the Relationships page under Tools (see Step (6)) and connect the

`country` field of the `sourceData` table with the `country` field of the `capitalsQ` query from step (13). Save the result.

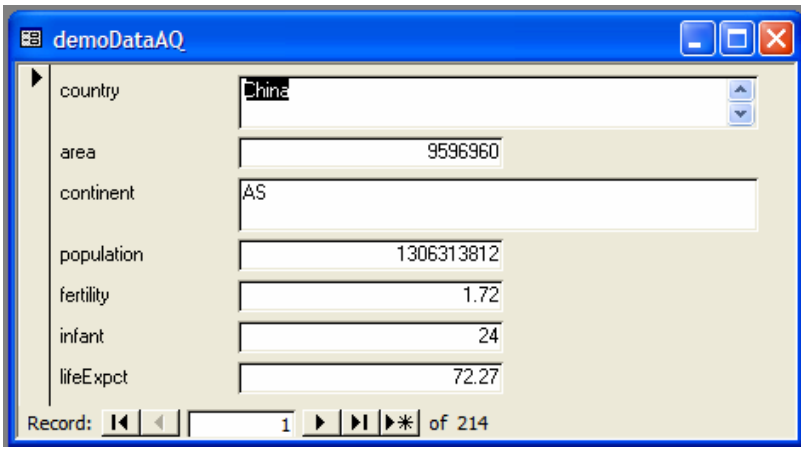
Our plan is to associate the country, its capital, its domain and its position data. The logical operation that we are performing uses a join to associate countries with capitals:

Select sourceData.country, sourceData.continent, sourceData.domain,
capitalsQ.place,
capitalsQ.latM, capitalsQ.latS, capitalsQ.NS,
capitalsQ.logM, capitalsQ.logS, capitalsQ.EW
From sourceData ▷◁ capitalsQ

Now we need to code it in SQL.

Moving to the query development window, click on Create a Query in Design View. Add the `sourceData` table and the `capitalsQ` query to the work area, and verify that there is a relationship between the two country fields. Next begin filling the fields. From `sourceData`, include the `country`, `continent` and `domain`. From `capitalsQ`, include the `place` (that's the capital) and the position data. Build the query. When it's finished, check it out and include the SQL as well as the first five rows in your Further Analysis document.

Step 14) *Displaying Data.* Next we build a form to display demography data. A form is simply a GUI that we set up to give a nice display of each record of a table. Forms are most easily constructed with a wizard, and we'll use that approach. Find the forms development display, and choose Create form by using wizard. Pick the `demography` query from Step (6) and move all of its fields to be selected, i.e. left to right. Next click through the choices (you can pick whatever background you prefer), and give the form the name `countryForm`. Here is what I produced. (Ignore its name.)

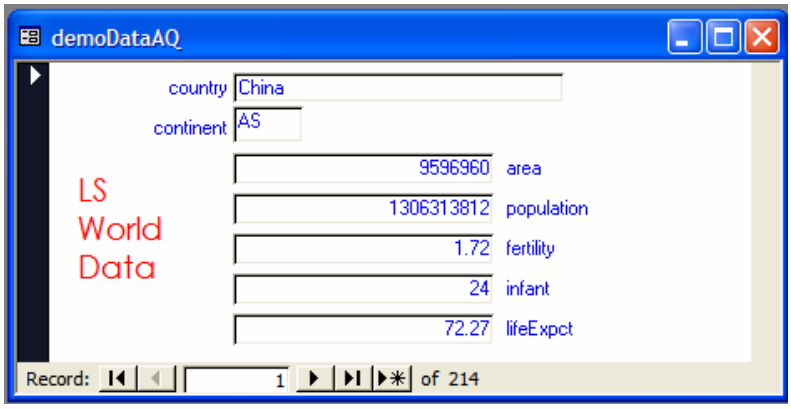


Next, you must edit the form (move into Design view) to accomplish the following:

- Resize the country box to be a single line.
- Resize the continent box to be "proper" size and put below the country name.

- Group the area with the other data values and move the labels to the right side.
- Change the background and/or font color.
- Finally, customize it with your initials and a phrase using a label from the tool menu (the ***Aa*** control).

These operations work in a pretty obvious way. Resizing requires that you select it and grab the corners. All of the boxes of the form are paired with a label. You can move them independently by grabbing their upper left corner (the cursor changes to a black hand with a pointing figure). Right clicking is the easiest way to change things like the background color; you'll have to experiment a little. My result is as follows:



Capture the result and put it in your Further Analysis document. Perhaps the easiest way to do this is to do a "screen shot," which requires you to simultaneously depress the Shift, Ctrl and Prt Sc keys; find the latter key in the upper right of your keyboard. That operation puts a picture of the display on the clipboard; this can be pasted into your document.

Step 15) *User-controlled demographic data.* Our goal is to allow the user to specify a continent and then to display the demographic data for its countries. In principle this is not tough, since

Select continent=[*user specified*] **From** demographic

is the table we want and the previous form can display it. The question is, how do we get the user data? To make this work in a civilized manner, we need to set up a form in which the user enters the data and then the display of the countries is launched. We'll do that in three steps:

- Build the initial form
- Revise the demographic query to reference the right text box
- Fix up the form so that it looks nice

Proceed as follows:

Build Initial Form. Begin in the design view, and right click to show Tools, the menu of controls to place on the form. Place a text box (the **ab** control) onto the form and make it large enough for the two letter `continent` text. (Somewhat larger is fine.) Next to it,

place a command button control. Follow the wizard to set up this button. Select Form Operations > Open Form. Make the form the `countryForm` that was just completed in the previous step. Show all records. Use the text, `Show Countries` as the button text. Call the command button, `Display`. Save the form as `WorldData`, and try it out. When you click on the button, your previously created form should display. (The continent input doesn't work yet.)

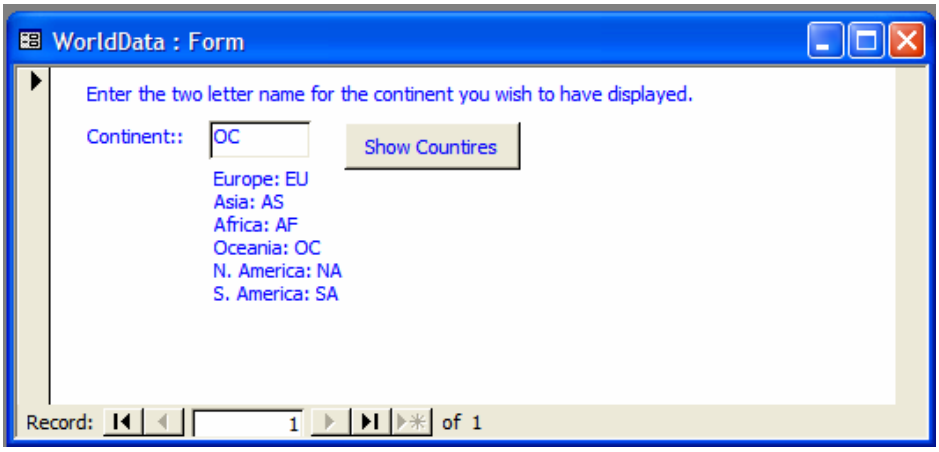
Modify the demographic Query. What we want is to type in the continent that we're interested in, click on the button, and only have those countries listed. To do this we need to have the `demographic` query select data on the value of the `WorldData` form's text box. It is an amazingly easy thing to do.

Begin by saving a copy of the `demographic` query under a new name (to help the grader), and then opening the original in the Design view. In the Criteria box for the `continent` field, right click and select Build. (We discussed this process at the end of Step (12).) The Expression Builder has a window at top showing what will be placed in the Criteria field once this definition is complete. Begin by placing an = sign in the window. What we want is to reference the value of the text window in the `WorldData` form, so we select Forms in the first column, double clicking to open the folder. Inside we double click on All Forms, and then on `WorldData`.

In the middle window are all of the objects on that form, only one of which should have a name like `text0`. (It will be exactly `text0` if you've followed these steps perfectly.) Select `text0` because this is the name of the text box you placed on the `WorldData` form. Once it is highlighted, the `<value>` field of the last column is also highlighted, though if it isn't then select it. Finally, click Paste above that column, and note what is inserted after the = sign. It should be: `Forms![WorldData]![Text0]`. This simply tells the database how to find the value that should be used for the selection operation of the `demography` table. Click OK, and notice that the continent's Criteria field has been set to select on the contents of `Text0`.

Try it out! Open `WorldData`, type some continent abbreviation, say `OC`, and notice that when the demographic GUI comes up, it only has the Oceania countries. It was long to describe, but it wasn't very difficult to do.

Primp the `WorldData` *Form*. Finally, return to the `WorldData` page in Design View, and add a label above the window and button reading: "Enter the two letter name for the continent you wish to have displayed." Also, below the window list off the continent abbreviations. Finally, customize the page so that it looks like the previous page in color and style. Mine came out looking like this.



Take a screen shot of your `WorldData` page with EU as the entry in the Continents window, and showing the `countryForm` display of the demographic data. Include the picture in the Further Analysis document.

Step 16) *Lost Islands of the World*. Our plan is to locate all of the islands mentioned in the `featureData` table, and locate which of them don't match countries. This is straightforward. We repeat the idea we used in Step (12) to create

> islandLife = **Select** place contains "sland" **From** featureData

We search for `"sland"` rather than `island`, `Island`, `islands` or `islands` so we can match them all without missing any. Following Step (12), create the `islandLife` query, keeping all fields except `latS` and `logS`. There should be 395 rows.

Next we construct a query

> claimedIslands = islandLife ▷◁ sourceData

which is the set of features with a recognizable (by our `sourceData` table) country associated with it. Follow Step (13) to create `claimedIslands`, keeping only `country` and `continent` from the `sourceData` table.

Finally, we want to find the rows that are in `islandLife` but not in `claimedIslands`. Abstractly, this is a simple difference operation

> islandLife – claimedIslands

but SQL is the *structured* query language, so it does not implement our five primitive table operations directly. To compute this difference, we use one last wizard. We click on New in the query development page and select Find Unmatched Query Wizard. We select the two relevant queries and step through the wizard. When the process is complete, we have a table of 65 rows of places where the country field didn't match.

What went wrong in the join? Nothing, really. Some of places are associated “countries” like `Antarctica` that are not countries. Others, like `Russia (de facto)` and `United States (Alaska)` simply do not match exactly, and so are thrown out. If we wanted our work to be complete, we’d have to decide how to handle all of these special cases.

Our search has been useful though. The list orphan places has some very unusual places on it, places we might like to go, such as

Passion, Ill de la (island)	Clipperton Island	10 N, 109 W

It sounds pretty interesting.

Copy and paste your SQL query for the “difference” operation, and the top five rows of the islandLife – claimedIslands into your Further Analysis document.

Turn-in.

>> Upload the files:
project3.doc
project3.mdb
To the Catalyst Turn In Area

Project 3 Point Breakdown:

50 Points Total

Step 2: [4 Points] Sorted by population, fertility, infant, and lifeExpect showing first 6 rows (5 rows and a column)

Step 3: [2 Points] Countries on both top 10 lists of fertility and population

Step 4: [2 Points] Countries in both lifeExpect and population
[2 Points] Bar chart of top 10 ranted by life expectancy

Step 5: [2 Points] Countries in both lowest 10 infant mortality and top 10 population

Step 6: [2 Points] SQL that produced table saved into word document
[2 Points] In your words, what does this query do

Step 7: [2 Points] Top 5 densest countries listed using new column

Step 8: [2 Points] African Countries in both top 10 highest infant mortality and fertility

Step 10: [2 Points] SQL that produced query results of OCdemography
[2 Points] First 5 rows of the resulting table and headings

Step 11: [2 Points] SQL that produced query results of place where latm is less than 23 & EW is equal to W
[2 Points] First 5 rows of the resulting table and headings

Extra Credit: [1 Points] SQL that produced query results of place between 23 deg 27 min N and S
[1 Points] First 5 rows of the resulting table and headings

Step 12: [2 Points] SQL that produced query results of capitalsQ
[2 Points] First 5 rows of the resulting table and headings

Step 13: [2 Points] SQL that produced query results of sourseData joined with capitalsQ
[2 Points] First 5 rows of the resulting table and headings

Step 14: [5 Points] Pasted screen capture of modified form (1 point per modification)

Step 15: [5 Points] Pasted screen capture of the WorldData page and countryForm

Step 16: [2 Points] SQL that produced query results of difference operation
[2 Points] First 5 rows of the resulting table and headings