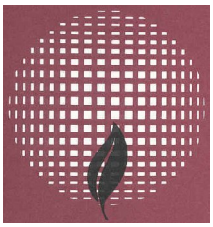


The Institute for Economic Development

Boston University

Dataset Management with Microsoft Access

Introduction

The purpose of this document is to familiarize users with ways to import datasets and construct data queries in Microsoft Access, the database software packaged with Microsoft Office, with the view that Access can be a useful tool for storing and manipulating large datasets.

Why use databases?

Whereas many formats and programs commonly used for data management, for example Microsoft Excel, store data as individual pieces or cells, relational databases created by programs such as Access use records as their fundamental units of account. This makes databases particularly useful for storing datasets that have a large number of observations or variables. A large dataset containing say 100 variables would span columns "A" through "CV" in an Excel spreadsheet. One well-placed accidental deletion of a cell not only corrupts the entire dataset but could in this case be nearly impossible to detect, particularly if the dataset already has missing values. By storing data at the record or observation level, relational databases provide a higher level of protection against such errors and, when errors are made, ensure that they are more often made uniformly so that they become more obvious to the user. Moreover, database queries allow users to perform targeted searches of their datasets, check for systematic errors, update data, and construct new variables and even new datasets in ways vastly more efficient than those offered by spreadsheet programs or statistical packages. Used in conjunction with these programs, relational databases such as those created by Access protect the integrity of data and offer users much more flexibility in the manipulation of their datasets.

This document…

…is not intended to be a user's guide to Microsoft Access, nor a fully comprehensive listing of all the potentially useful functions of Access, but rather to serve as an introduction to basic data importation and querying techniques that can help users work more efficiently with their datasets. Users unfamiliar with Access will probably find it necessary to refer to an Access user's manual or the "Help" index as they familiarize themselves with the techniques presented here. This is a working document and new techniques may be added in the future.

Suggestions

If you have suggestions for how to improve this document, please send them to pkarner@bu.edu.

Disclaimer

Finally, use the techniques described by this document at your own risk. Neither the author nor the Institute for Economic Development bear any responsibility for data lost or damaged as a result of following the techniques suggested in this document. *Always make a backup copy of your data before using any of the techniques described below.*

Table of Contents

1. Glossary

Primary key: A set (possibly singleton) of variables that uniquely identifies each record – i.e. whose values are different for each record. Usually the minimum such set. For example, in a panel dataset, the primary key may be composed of the "country" and "year" variables because each country-year combination uniquely identifies an observation. See also http://en.wikipedia.org/wiki/Primary_key.

Query: A filter designed by a database user to search data and highlight records that match particular criteria. At their core, queries are strings of commands written in a database query language such as SQL. Typically database programs such as Access have a graphical interface that assists users in creating queries much as graphically-oriented programs such as Macromedia Dreamweaver allow users to design web pages.

Record: The basic unit of account of a database or, more specifically, of a table in a relational database. If a table consists of a dataset, a record is an observation.

Relational database: A data storehouse distinguished from other types of databases (e.g. flat-file databases) by its ability to create and store relationships across tables, e.g. between a "Books" table and an "Authors" table, which generally allow users to store data more efficiently and design a richer set of queries. See also http://en.wikipedia.org/wiki/Relational_database.

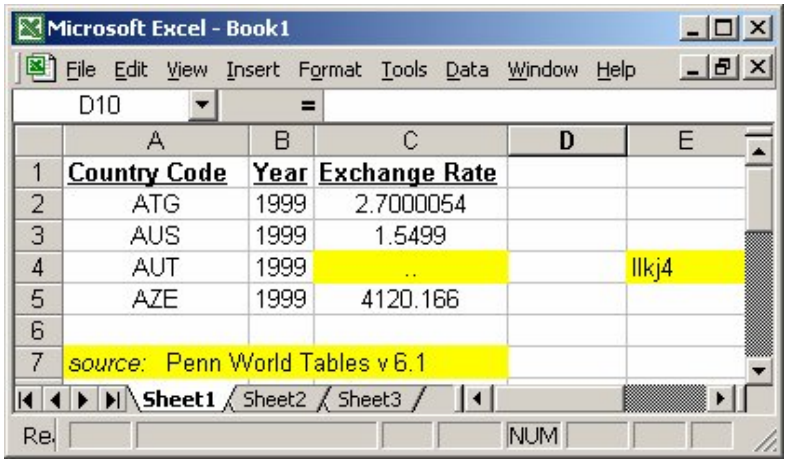
2. Importing Data from Excel

A. Prepare the data – anticipate formatting issues

1. Variable (column) names should be in first row

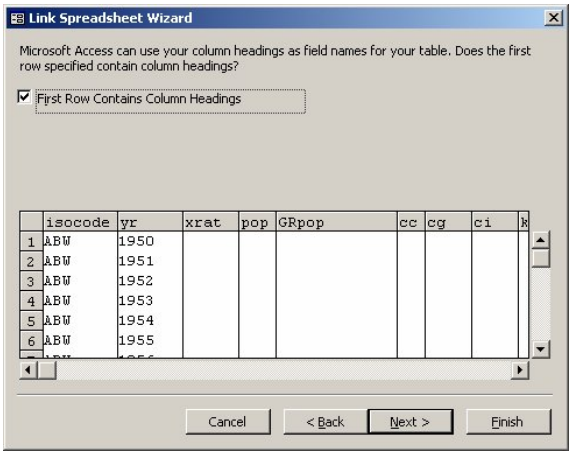
2. Delete any gaps between records or below/to the right of the dataset – this will cause extra records (sometimes A LOT of them!) to be imported

3. Missing values should be blank, not ".." for example – this allows Access to classify the data with the proper data type (e.g. integer, text, etc.):



B. In Access, choose "File…Get External Data…Import"

1. Check "First Row Contains Column Headings"



2. Change column names if you wish

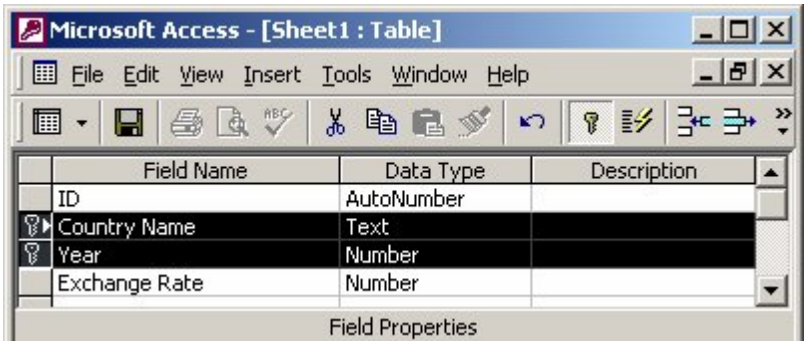
3. Let Access create a primary key unless the data have only a single key and you are sure that you deleted any possibly erroneous records.

4. Import to new table unless the variable names and data types of the incoming data are identical to those in the table to which you want to append the incoming data.

5. Format the new table
    a. Look at the data – scroll all the way to the right and to the bottom to make sure there are no extra variables / observations

    b. Design View:
        i. Make sure the data types are correct – especially that no "integers" should be "doubles," as this could cause Access to erroneously truncate the decimal portion of data in a variable.

        ii. Define a primary key – Access has probably created a unique number for each record. If you wish, you can highlight the set of variables that uniquely identify each observation, right-click and make them the primary key, then delete the AutoNumber "ID" primary key that Access created for you.

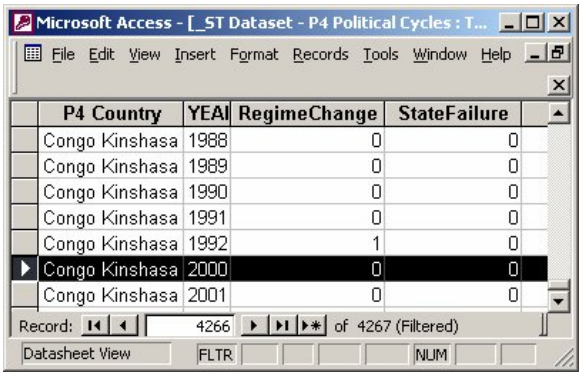


3. Using Access to Sort and Filter Data

A. Sorting
- This is an easy way to get a feel for the highs and lows of the data, missing data, and bogus entries.

B. Filtering (by selection or by excluding selection)
- Extremely handy for testing the impact of certain observations or groups of observations on your results – e.g. "what would happen if I excluded China from the model?"

- To define groups of observations, create a dummy variable which takes value "1" if the observation is a member of the group. Then, you can filter out all 1's to exclude the group, or you can filter out all 0's to see the effect of just that group – e.g. "does this result hold without Sub-Saharan Africa in the sample?" or "does this result hold for Latin America only?"



- Advanced Filter/Sort – allows you to apply the same screening process every time you look at the data – essentially a query.

C. Hiding Columns: Format…Hide (Unhide) Columns

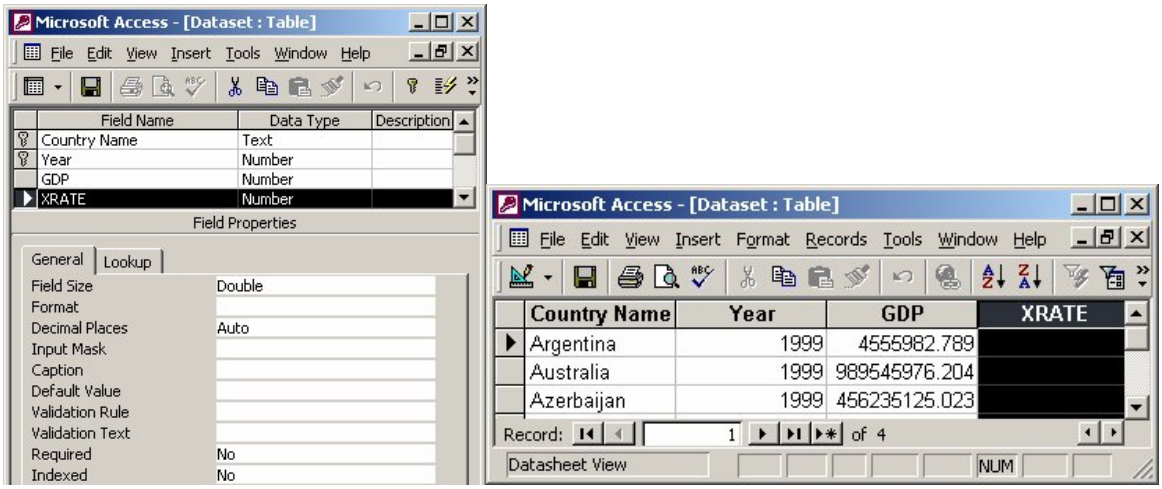
- Allows you to retain data that could be useful in the future without it getting in the way now.

- Can help prevent improperly specified models – e.g. I have a variable called "GDP" and another called "LNGDP"…hiding the one that is not being used prevents confusing the two in regressions.

4. Compiling Data from Multiple Sources Using Queries

Designing a query to compile data

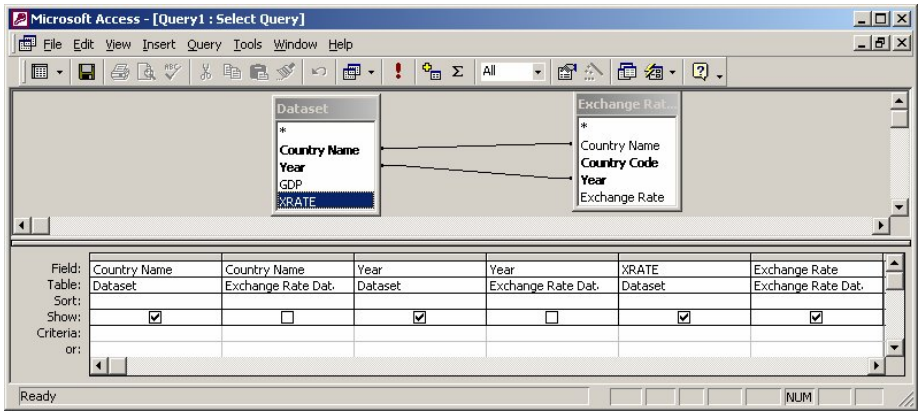
A. Create empty columns in your dataset to prepare for the incoming data:



B. Create New Query in Design View

C. Create *relationships* between variables with the same meaning in two different tables by dragging one variable on top of the other in the query's Design View.

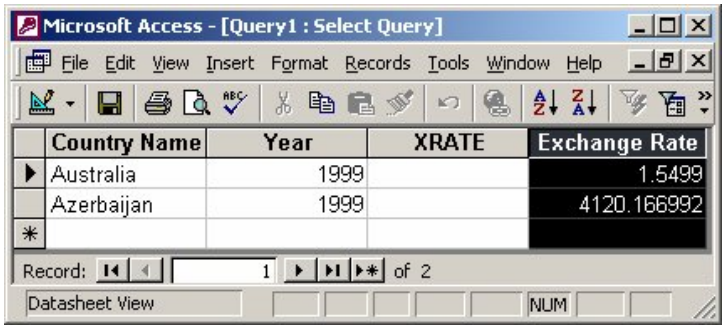
D. Drag the variables that you wish to display from the two tables down into the query design – this creates the SQL statements that will run your query



E. Run the Query – click the upper-left button in design view to display the results.

F. Check number of records – If smaller than your dataset, or smaller than you wish as a result of different names for the same observation, see below.

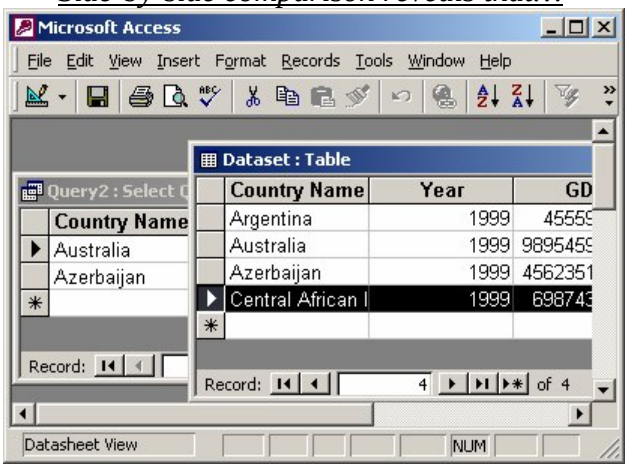
G. Copy/Paste data from the other table into the empty column of your dataset:

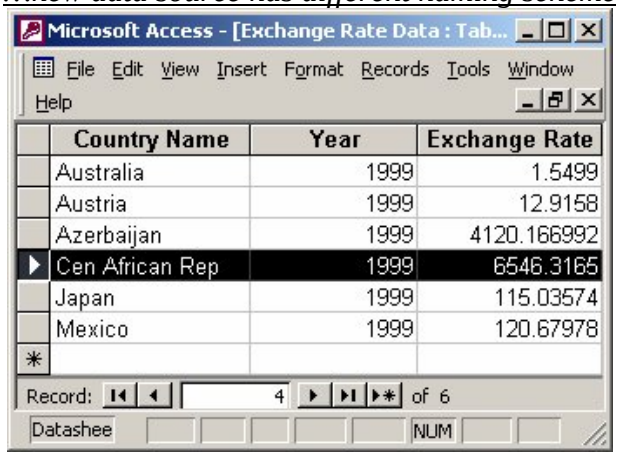


5. Querying Datasets with Different Names for the Same Observations

A. Generally, a side-by-side comparison between the query and the dataset is the best way to see which observations are missing that perhaps should not be:

Side-by-side comparison reveals that… *…new data source has different naming scheme*





B. Working with several tables/variables/years, etc. from the same source - it is useful either to create a 'translator' table or to create an additional column in your dataset containing the names of the records in the new data source. This saves you from having to re-format the record names for each table of the new source and also preserves the data so that it can be more easily traced back to its original source.

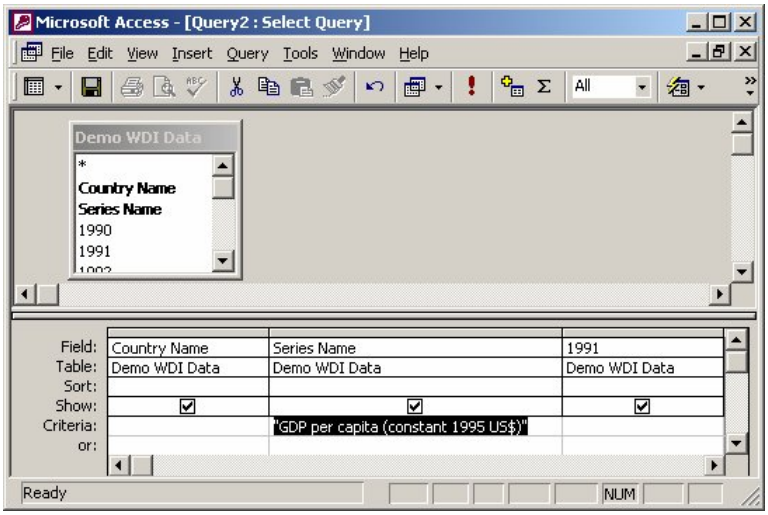
- Translator Table: Make a table with two columns – one containing the record names from your dataset and the other containing the record names from the new source. When you design the query, use this table to match records from the other table to your dataset. This technique may prevent the query from being "updatable" – in this case, refer to the suggestions below.

6. Suggestions for Making Queries Easier:

- Try to avoid pulling together data from more than two tables at a time – if you wish to consolidate data from three tables, it is generally easier to do two separate queries.

- Use the *Criteria* field in query design to help with querying more complex datasets.

  - e.g. The *Criteria* field is especially useful when working with a dataset containing multiple series imported from large datasets such as the World Bank's *World Development Indicators* (WDI). The incoming WDI data has a variable for the series name and a variable for each year. If you only wish to display certain series, specify their names (in quotation marks, separated by commas) in the Series variable's *Criteria* field:



- Queries don't take long to make, so don't hesitate to construct them again. If you believe that your query looks grossly different than it should (e.g. the number of records is hundreds greater/fewer than expected, etc.), you're probably right – it is likely that the SQL code is no longer perfectly synchronized to the design view's graphical interface. In this case, it's best to start the query over from scratch.

- For un-updatable queries, you cannot copy-and-paste within the query. You have a couple of options:

    1. If you are sure that your query results look exactly like your dataset (in length, etc.) then copy from the query results and paste directly into your dataset – A WORD OF WARNING: It is very easy to match data with the wrong observations doing this. Luckily, the errors will be systematic and will begin occurring at the record in your dataset that does not match up with the query, so visually inspecting the query and dataset side-by-side can help you identify errors.

    2. Copying and pasting between Access and Excel is easy, so you can always copy both columns into Excel, make sure they line up (and adjust them if they don't by deleting/adding rows), and then paste the formatted data into your dataset.

7. General Suggestions

- Keep a backup copy of your database whenever you are updating it. Unfortunately, many aspects of the data importing and cutting/pasting processes described above are irreversible for large datasets because a single operation can consume lots of memory.

- Keep databases as self-contained as possible. If you are going through the trouble of importing a new dataset, and especially if it could be useful to others, give it its own database and link it to your own database (see below), then break the link when you're done. This keeps your database as small and efficient as possible.

- Use the "Link Tables" feature – "File…Get External Data…Link Tables" allows you to use a table from a different database in your own without duplicating it. This can save a

lot of disk space. When you're done copying the needed data into your dataset, simply break the link by deleting the linked table: