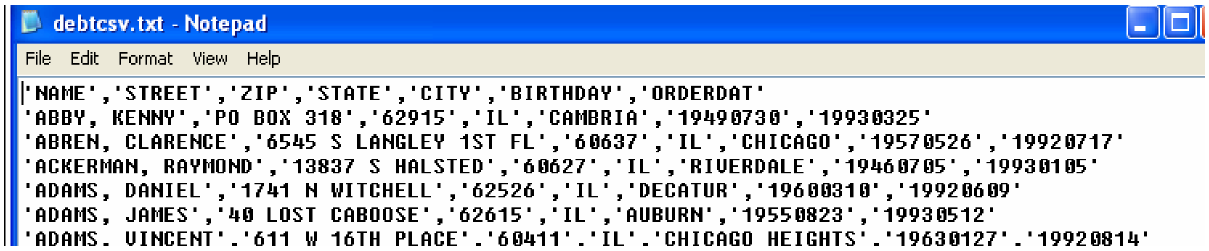
Welcome to the Real World:

Importing, rearranging and cleaning data

Jeff Porter
IRE and NICAR
jeff@ire.org

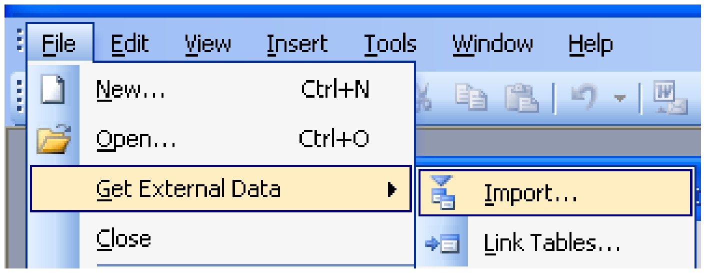
In most Access exercises, the data is nicely arranged already in tables, ready to use. When you obtain government data, though, the chances are it's a bit more complicated. Here's an exercise that should help.

The file in question is called debtcsv.txt. Open it up on notepad and you'll see:

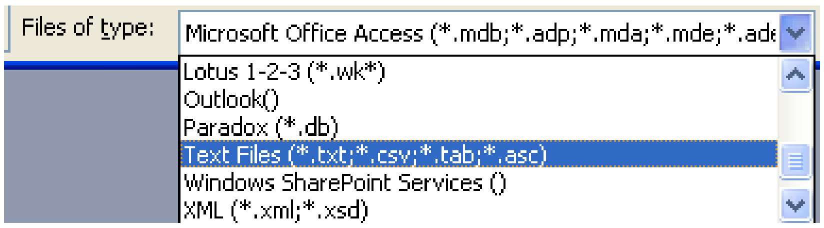


```
'NAME','STREET','ZIP','STATE','CITY','BIRTHDAY','ORDERDAT'
'ABBY, KENNY','PO BOX 318','62915','IL','CAMBRIA','19490730','19930325'
'ABREN, CLARENCE','6545 S LANGLEY 1ST FL','60637','IL','CHICAGO','19570526','19920717'
'ACKERMAN, RAYMOND','13837 S HALSTED','60627','IL','RIVERDALE','19460705','19930105'
'ADAMS, DANIEL','1741 N WITCHELL','62526','IL','DECATUR','19600310','19920609'
'ADAMS, JAMES','40 LOST CABOOSE','62615','IL','AUBURN','19550823','19930512'
'ADAMS, VINCENT','611 W 16TH PLACE','60411','IL','CHICAGO HEIGHTS','19630127','19920814'
```

Looks ugly, doesn't it? But Access can handle it with ease. First, start a new and blank Access database. Use File > New … in the menu. Navigate to the correct folder to save the database file, name the file Strings, and hit the Create button. Choose the button for Tables, then use this menu item: File > Get External Data > Import…

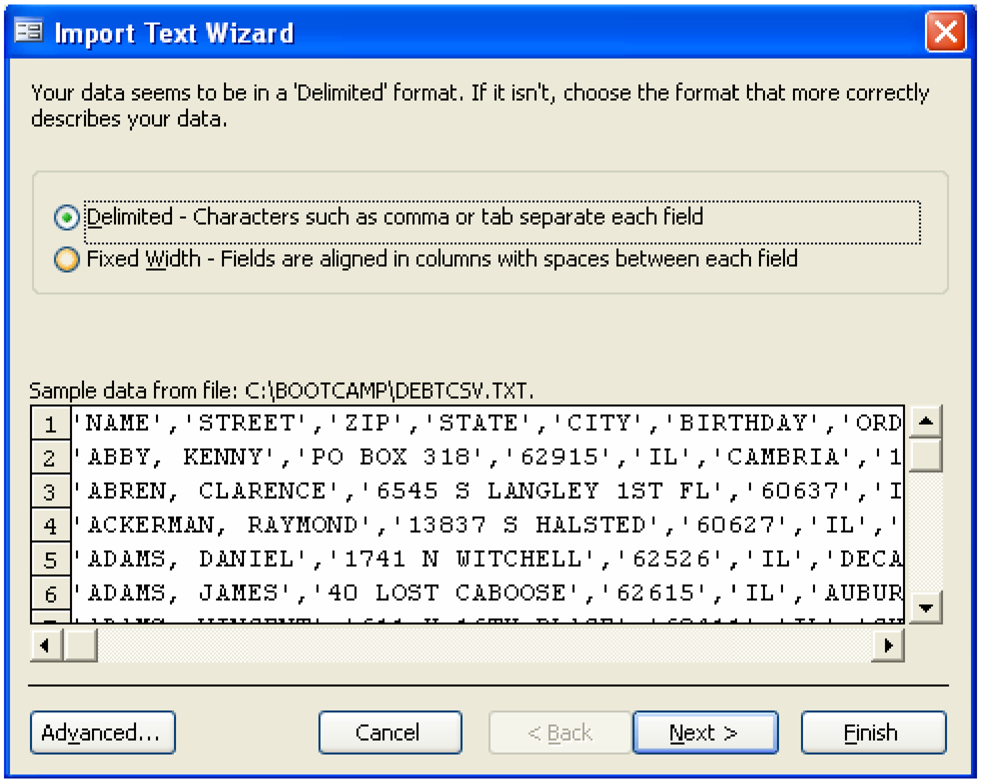


You'll be presented a dialog box. At the top, navigate to the appropriate folder, where the file debtcsv.txt is stored. Next, at the bottom of the dialog box, click next to "File of Type" and choose Text:


Files of type:
Microsoft Office Access (*.mdb;*.adp;*.mda;*.mde;*.ade
Lotus 1-2-3 (*.wk*)
Outlook()
Paradox (*.db)
Text Files (*.txt;*.csv;*.tab;*.asc)
Windows SharePoint Services ()
XML (*.xml;*.xsd)

With that, the file debtcsv.txt should appear. **EXTRA TIP:** Sometimes, government agencies give files odd suffixes, such as .lst. If it's not .txt, .csv, .tab or .asc, Access won't recognize it as a text file. So just change the suffix to .txt.

Once the file is found, highlight and click on the Import button. That will bring you to another dialog box:


Import Text Wizard

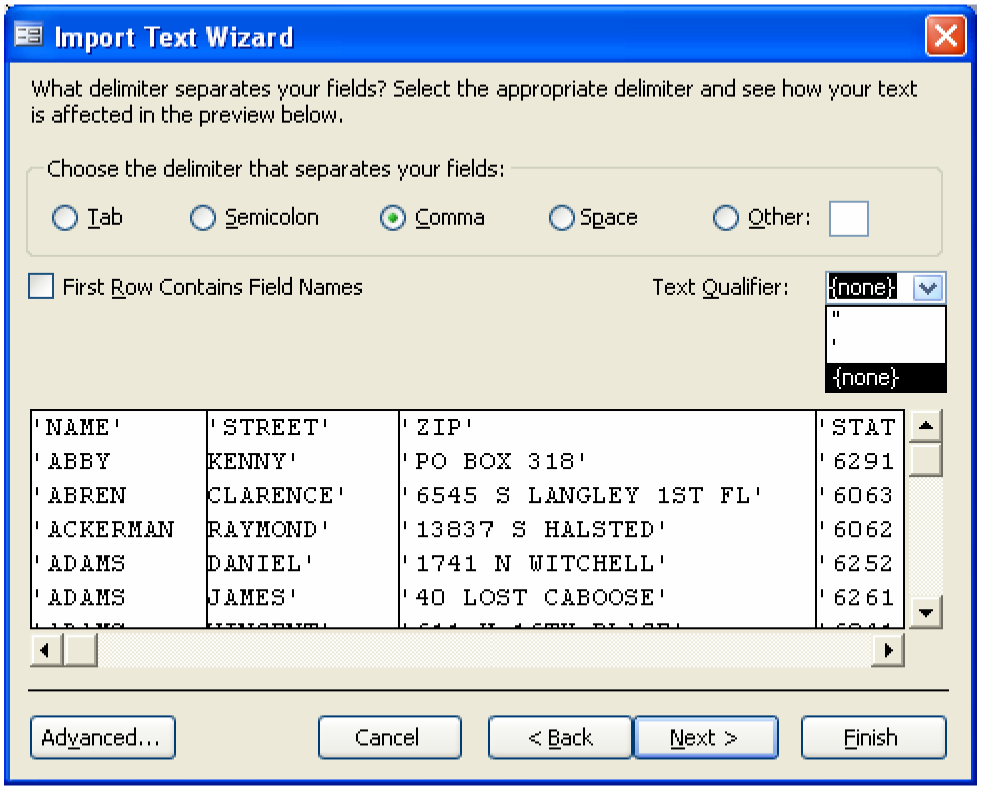
Your data seems to be in a 'Delimited' format. If it isn't, choose the format that more correctly describes your data.

[✓ Delimited - Characters such as comma or tab separate each field]
Fixed Width - Fields are aligned in columns with spaces between each field

Sample data from file: C:\BOOTCAMP\DEBTCSV.TXT.

```
1 'NAME','STREET','ZIP','STATE','CITY','BIRTHDAY','ORD
2 'ABBY, KENNY','PO BOX 318','62915','IL','CAMBRIA','1
3 'ABREN, CLARENCE','6545 S LANGLEY 1ST FL','60637','I
4 'ACKERMAN, RAYMOND','13837 S HALSTED','60627','IL','
5 'ADAMS, DANIEL','1741 N WITCHELL','62526','IL','DECA
6 'ADAMS, JAMES','40 LOST CABOOSE','62615','IL','AUBUR
```

It should guess, correctly, that the file is Delimited – that is, each field is separated by a comma. Some times, government agencies provide Fixed Width files, which is nicely arranged in columns, with just spaces to indicate. If that's the case, you should receive a record layout that lists each field; whether it's text, numbers, dates or other format; and the width of the field.



What delimiter separates your fields? Select the appropriate delimiter and see how your text is affected in the preview below.

Choose the delimiter that separates your fields:

Tab Semicolon [✓ Comma] Space Other:

First Row Contains Field Names

Text Qualifier: {none}

```
'NAME'     'STREET'    'ZIP'                       'STAT
'ABBY      KENNY'      'PO BOX 318'                '6291
'ABREN     CLARENCE'   '6545 S LANGLEY 1ST FL'     '6063
'ACKERMAN  RAYMOND'    '13837 S HALSTED'           '6062
'ADAMS     DANIEL'     '1741 N WITCHELL'           '6252
'ADAMS     JAMES'      '40 LOST CABOOSE'           '6261
```

That takes you to step 2. Notice that it guesses, likely correctly, that the delimiter are commas. If it doesn't guess correctly, you can easily change it. Watch the little window showing a slice of the data carefully. Something is not right, is it? A vital piece of importing is at the far right: Text Qualifier. What's a Text Qualifier? Notice in the sample how items are encased in single quotes. Those are there for a reason, allowing the possibility that, even if the data fields are separated by commas, a comma might appear as text inside one of the fields. For example, the name Abby, Kenny. But Access didn't guess it correctly, so that name is split apart, with the last name of Abby showing up under the column head Name but Kenny's first name is under Street. Not good. So choose the single quotes as Text Qualifier, and Kenny is lined up quite nicely. One more task before you leave this box: Check the box that says First Row Contains Field Names. Sometimes, those first rows don't contain names, but in this case, they do. That being taken care of, hit Next.

The last dialog box question: should this be saved in new table or an already-existing table. Chances are, you'll want to choose the first open. But, for example, if you get a series of text files that should be part of the same table and follow the same structure, choose the latter and identify the table in your database, after you first create the table following the procedure of this exercise. Hit Next.

That takes you to the final step, which is also, potentially, the most tedious. But don't hurry through this. It will as you the name, length and format of each field. To move from one field to another, just click under the field name. A few format notes: Most of the first are easy: names and addresses are text, but what about ZIP codes, court order dates and dates of birth? As the data presented, it doesn't look like a date most of us recognize. Instead of 07/30/1949, it's 19490730. So if you choose Date as the format from the drop-down menu in this dialog box, you won't get good results. In fact, the data will be erased. For now, keep those dates as text. Do the same with ZIP codes. After all, do you every do math on ZIP codes? As much as they appear to be numbers, they're really text creatures. **EXTRA HINT:** When in doubt, leave data elements as text. Keeping it in text gives you a high degree of control, and you can always change it later.

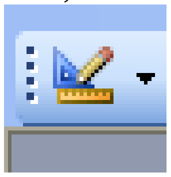
Peruse the data elements, make sure, in this case, they're all text, and hit Next.

Just a couple more steps: It asks you if you want to create a Primary Key. Basically, it's a sequential number, record by record. In this example, it doesn't harm anything, so keep it. If you've already got a Primary Key in a later database, you can tell it no thanks. Again, hit Next. And finally, it asks you what name you want for the file. Name it what you will in the future, but for now, keep it named ORDERS. Hit Finish, wait for a little box telling you the import is successful, and click OK. Then your table should appear in the Access database.

Now, time to divide and conquer

Open the table and take a look. A few problems are obvious. Names are all in one field, and sometimes we want to split out last names. The two date fields aren't real dates, and true, actual dates are quite valuable at times. City spellings are inconsistent. For example, there's East St Louis and an E St Louis. So we'll see how to fix those, too. We'll do this in the long way, no short cuts yet. We'll carve up the data, then put it back together in the correct order, one small step at a time.

First, when the table open, hit the little icon at the top left that looks like a little triangle:



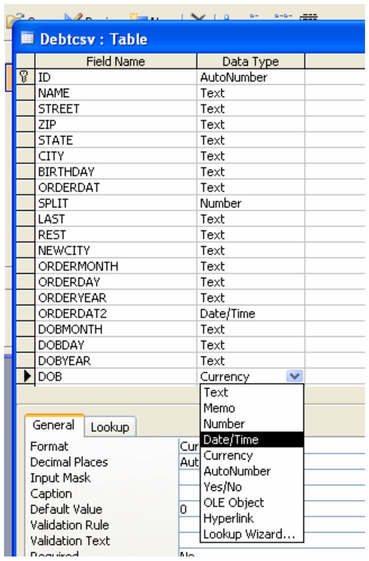
That takes you to the "design" of the table, a listing of all the fields and type. We're going to add a few.

Click below the last field, or ORDERDAT. Name the first new field SPLIT. Click over to Data Type and choose Number. Look below, and you'll see it's calling it a Long Integer. For now, that's fine. Name the next one LAST. Click over to Data Type and make sure it stays as Text. Look below, and you'll see that it defaults to 50 characters in length. For this example, that's fine. Add another one called REST. Again make sure it's Text and 50 characters in length. Those are the fields needed to fix the names.

Now, to fix the city names: After REST, create a new field and name it NEWCITY. Fifty characters should work fine for that one, too.

Below NEWCITY, create a field called ORDERMONTH and make sure it's Text. Instead of keeping it 50, change it to 2. Do the same for a new field called ORDERDAY. One more field: ORDERYEAR. Make it text, four in length. Next, create one called ORDERDAT2 for Data Type, choose Date/Time. Prepare to fix the dates of birth in the same way: Three new text fields, naming them DOBMONTH, DOBDAY, DOBYEAR, making sure the last one is four characters in length. Then put one called DOB, in Date/Time format.

Your screen should look like this:



Debtcsv : Table

Field Name	Data Type
ID	AutoNumber
NAME	Text
STREET	Text
ZIP	Text
STATE	Text
CITY	Text
BIRTHDAY	Text
ORDERDAT	Text
SPLIT	Number
LAST	Text
REST	Text
NEWCITY	Text
ORDERMONTH	Text
ORDERDAY	Text
ORDERYEAR	Text
ORDERDAT2	Date/Time
DOBMONTH	Text
DOBDAY	Text
DOBYEAR	Text
DOB	Currency

Format	Cur
Decimal Places	Aut
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	

At the same place where you found the little triangle, now you'll what looks like a little table or spreadsheet. Click on it. It'll ask you if you want to save, and tell it Yes.

There are your new fields. Now it's time to fill in the blanks.

First a little intro on what we'll call string functions. String is just another word for text. Because of the nature of text, it's easy to manipulate, move around and rearrange. There are three string functions we'll want to play with. These definitions are courtesy of MaryJo Sylwester of the St. Paul Pioneer Press:

LEFT: This tells the computer to start at the first byte on the left side of the field. Then we have the option to tell it how many bytes to take.
Syntax: LEFT(fieldname, number of bytes)

MID: This starts someplace in the middle of the field, which we will designate, and goes either to the end of the field (if we don't specify) or to a certain point that we can set.
Syntax: MID(fieldname, byte number to start at, number of bytes to take)

INSTR: This works as a sort of search tool to count until it finds the character you seek. It returns a number, which can tell the computer to either start or stop to split, say, a name field at the comma. For this type of work, it is used in conjunction with the MID function (more on that syntax later). It can also be used to determine at what position a certain character is located. The character you what to find should be enclosed in quotes.
Syntax: INSTR(fieldname, “character we want to find”)

There is also a RIGHT function, which starts at the first byte on the right side of the field and then you can tell it how many bytes to take. (It isn’t as useful as the others, however.)

Thanks, MaryJo.

Now, the first fix will be for names. Note that commas are important because those characters are a dividing line between the last name and the rest of the name. So using the INSTR function, you can find out, in each record, where the comma starts. But one small problem: When it’s all said and done, you won’t want the comma included in the last name, do you? So you want to memorize, for each field, the number of characters where the comma starts, then subtract one.

So start up a new query. Go to View > SQL View and erase everything. For this set of queries, the magic word is not SELECT. Instead, the magic word is UPDATE. Type this language:

UPDATE ORDERS SET split = INSTR(name,',')-1

Let’s parse that out. The first words note that you’re going to change, or update, your table named ORDERS. The word SET flags Access to let it know you’re about to name a field to update. The word SPLIT is a reference to the field you created. The equal sign tells it to fill in the field SPLIT with the following material. INSTR is the command to start looking for a particular character. NAME is the field and the ‘,’ is, well, a comma. Hit the exclamation point and you’ll be told you’re about to fill in all those records. Tell it Yes and then check out the content of the field SPLIT. It should be a series of numbers that match exactly the number of letters in each last name. So now, we’ve got a reference number to split up those names.

Do a File > Save and give your query a name. Without closing it,, let’s take the next step with another UPDATE query, erasing your previous language after the word SET:

UPDATE ORDERS SET last = LEFT(name,split)

Notice that in this function, you’re not typing in a particular number, but the word SPLIT, a field name. The update queries allow Access to scan the entire table, record by record, and use a field as a number. So in other words, in the first record, the value SPLIT would be 4. That’s how many characters are in the name ABBY. Second record, the value is 5, the same number of characters in the name ABREN. And so on.

Hit the exclamation and run the query. Go back to the table and check the field LAST. It should be filled in with those last names.

Do a File > Save As …. and give your query a new name. Don't close it.

Now, for the rest of the names. We'll use the MID function:

UPDATE ORDERS SET rest = MID(name,split+3)

Why +3? Well, take a look at the field SPLIT and the name of ABBY, KENNY. The value is 4. but if you start at character 4, you'll start with the Y. You want to start with the K. So 4+3=7, and the letter K is the seventh character in the field. We don't have to list a third "argument" for this one, just the field name and one number, because we want the function to pick up the rest of the field. So the command is telling Access to start two positions over past the SPLIT value and take everything else.

So run that query, and you'll have the rest of the name (mostly first names, with a smattering of other material, such as suffixes, middle names or middle initials. Do another File > Save As… and let's move on.

With the names being split, let's go to our next task of fixing dates. Dates are quite valuable if you want to measure days, months, years. For example, in this example, one could find the age of the person of the date the order was filed. That is, if the field is an actual date.

Check out the field ORDERDAT. For the first record, it's: 19930325. So the first four characters show the year, the next two the month, and the last two the day. So Kenny's order came across on March 25, 1993. We, of course, already have four fields to handle the conversion, covering month, day and year. Here's how to fill them. We follow the same parsing in the functions, except that unlike names, the date fields are extremely consistent. So we don't need to calculate a place to split them apart and won't need to use the INSTR function:

UPDATE ORDERS SET ordermonth = MID(orderdat,5,2)

In other words, we fill in the field ORDERMONTH by looking into the field ORDERDAT, count over 5 spaces and take 2 characters. In Kenny's case, that's 03. Do another File > Save As…

Next:

UPDATE ORDERS SET orderday = MID(orderdat,7,2)

That'll take the day. Another File > Save As…

And the year is easier, since it starts at the beginning:

UPDATE ORDERS SET orderyear = left(orderdat,4)

Now, to put stitch those together in the field ORDERDAT2:

UPDATE ORDERS SET orderdat2 = ordermonth + "/" + orderday + "/" + orderyear

In other words, we put in the month, a slash, the day, another slash, then the year. Run it and check your results. And do another File > Save As…

Now it's your turn: Use the same tools to fill in the DOB field.

Dirty data and one way to clean it

Peruse your ORDERS table and you'll see some inconsistency in city names, as mentioned above. Here's the methodology on fixing this. First, run this query to check out city name problems:

SELECT city, count(*)
FROM ORDERS
GROUP BY city
ORDER BY 1

The ORDER BY part means the cities will be listed in alphabetical order. Run the query and you can eyeball the results:

For the purpose of this exercise, let's focus on a common inconsistency for CHICAGO. Sometimes, it's spelled CHGO. And then we can work on a slightly more complicated name: CHICAGO HEIGHTS. It's inconsistent in full spelling, but it often starts with CHICAGO H… . First, let's populate the field NEWCITY like this:

UPDATE ORDERS SET newcity = city

That just throws all the content of field CITY to the field NEWCITY. We created a new field so if, by accident, we make a mess of this, we can easily start anew with our untouched original field, CITY.

To fix CHGO:

UPDATE ORDERS SET newcity = 'CHICAGO'
WHERE city = 'CHGO'

Here, we tell the update query to make the new city CHICAGO, but only in instances, using the WHERE line, in which CHGO is in the CITY field.
A bit more complicated, using the wild card:

```
UPDATE ORDERS SET newcity = 'CHICAGO HEIGHTS'
WHERE city LIKE  'CHICAGO H*'
```

Now, you take a shot at E ST LOUIS vs. EAST ST LOUIS.

With this type of technique, you’ve got the power to allow more consistency, and applied correctly, more accuracy for you r data analysis.

Other data imports

Not every data file you’ll get will be a comma-delimited file or arranged in this way. Access has a similar Wizard dealing with fixed-width text files – similar to the Excel interface. The only large worry is that you still want to make sure you’ve formatted your fields correctly.

Other imports are even easier. NICAR provides data in DBF format, for example, You can use the same starting point: File > Get External Data > Import … and choose File of Type as dBase IV, or DBF. You can do the same task for Excel files and a host of others. Peruse the list of File of Type.