

# THE JOY of QUERYing

A recipe book for creating Queries, and reports in C/DEMS

Washington State Dept of Health

7/31/03

Since its birth in 1999, the Diabetes Electronic Management system (DEMS) and its successor the Chronic Disease Electronic Management System (CDEMS) have transformed how care is provided in clinics across the nation. In Washington State alone, currently tens of thousands of patients are being managed in one of these registries and hundreds of providers are using the program for more efficient, focused visits and to help with case management.



Because all clinics are different, how clinics use the program varies greatly, but it's clear that once someone experiences the power of a registry to change the way care is managed – doing things the 'old' way is unthinkable.

Finding new ways to use the data in a DEMS or CDEMS registry (collectively termed C/DEMS here) is what this course is all about.

We will cover:

- The internal structure of C/DEMS, and how you can change it to meet your needs,
- Simple queries that can change your life forever,
- Discovering what queries you already have at your fingertips, but didn't know about.
- Making queries provide just what you want them to – to meet your needs.
- Surviving in the computer jungle.

## The internal structure of C/DEMS,

### Make it work for you!

C/DEMS is written in Microsoft Access 97 (Access), which is a relational database, and we all know that relationships are what's important. Relational databases are very efficient for storing lots of different types of data in one place. A dude named Codd started all this fuss about databases in the 70s. He observed that you could store data efficiently by creating master keys that tie many different data items (stored in tables) together and using the keys you can piece together these items to give the total picture.



C/DEMS stores patient contact information, the vital signs taken and dates of each clinic visit, lab results, specialty service dates, types of meds each patient is on and the patient's health conditions. Each of these items is stored in tables and to piece together a picture across these categories requires using the key that identifies a patient. In C/DEMS the key that holds the data together is the combination of the `chart_number` and the `clinic_code` fields.



This means that you'll find these two identifiers (`chart_number` and `clinic_code`) in every table storing patient data. And there's lots of patient data in these registries. To see for yourself the real deal, open any C/DEMS file (e.g., `Qlib.mdb`; `RPT.mdb`, `DEMS.mdb`, `cdems.mdb`, `dems_dta.mdb`, `cdems_dta.mdb`) and press the F11 key. Pick the tab that says "Tables" and you'll see several items starting with "tbl" these are the tables holding patient data.

If the file has a little arrow next to it – you are not in the primary data file (`dem_dta.mdb` or `cdem_dta.mdb`), but instead are looking at a "linked" file. Linked tables function like other tables, but are physically located in another database file - - on a server or maybe even in a different city. The advantage of using "linked" tables is that many sites can have access, enter and edit data in the master file, while only one file needs maintenance. The master patient data file in C/DEMS,



How to determine where the patient data (`cdem_dta.mdb` or `dem_dta.mdb`) file is located:

Method 1: In Access 2002 you find where the linked tables are physically located by hovering your mouse over a table name for a short period. The path to the source file will display.

Method 2: You can use the linked table manager. (Access2000 or 2002) Tools on menu bar->Database Utility->Linked table manager. (Access97) Tools-.Add-ins-Linked Table manager. The path to each linked file is displayed.

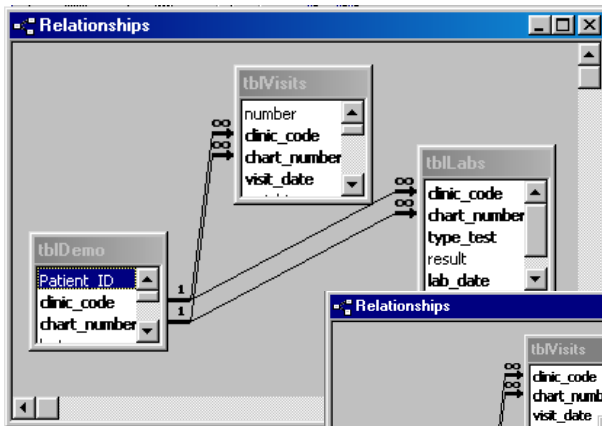
Method 3: Tools->Analyzer->Documenter->Tables tab->pick a table-> Select Properties checkbox In "include for table section" ->Ok->Ok. The path ( if the table is linked) is on first line after Attributes,'database=xxxx"

With this much emphasis - - Do you get the idea that knowing where the data is is important for backing up and maintenance

where the “tbl” tables reside, always has a name that includes: “dta” (e.g., “cdem\_dta.mdb” or “dems\_dta.mdb”).

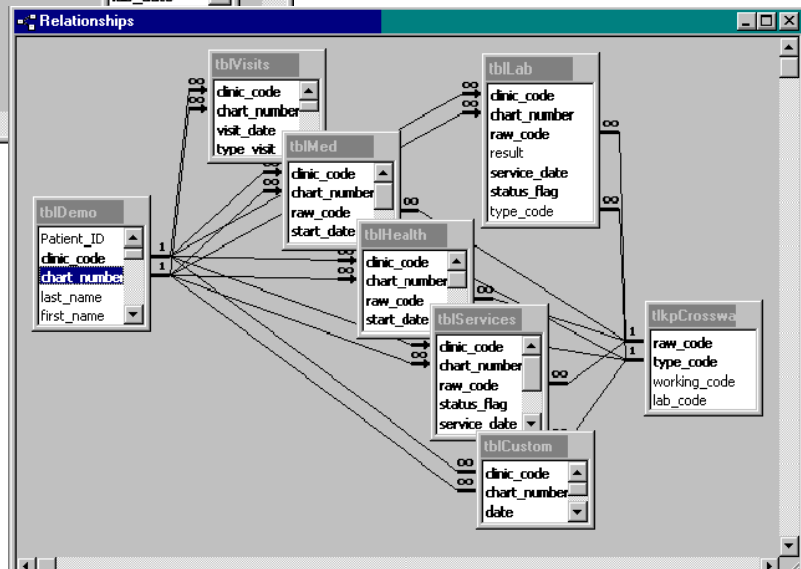
In DEMS you’ll see patient data stored in the “tblDemo”, “tblVisits”, or “tblLab” table. Patient contact information like name and address and provider’s name is stored in the tblDemo table and lab results in the tblLabs table and everything else in the tblVisits table. This is efficient because you don’t need to store all the names and contact information with every lab or visit record. If you looked at a lab record in tblLabs, you’d see the clinic\_code and chart\_number fields as well as the lab results. You’d have to look in the tblDemo record to find the patient’s name associated with that chart\_number/clinic code key or in tblVisits to find out that patient’s last BP and the date it was taken. The clinic\_code and chart\_number combination operates kind of like a magic decoder ring.

At every visit DEMS takes a snapshot of the patient’s meds, service history, current health conditions, and current self-management note. These are all stored all in the tblVisits table and dated with the visit date.



All this is different in CDEMS. Each of these items has a separate table. So instead of the DEMS single tblVisits table, in CDEMS you’ll find a tblMed, a tblHealth, tblVisits, tblServices and tblCustom table (tblCustom holds the self management note). In CDEMS

there are more tables than in DEMS, but the advantage is that you can add information about a patient at any time – not just in snapshots dated with a visit date.



Here you can see the table structure for DEMS and CDEMS. To be in C/DEMS, a patient has to have a record in tblDemo, and the chart\_number/clinic\_code combination key must be unique. Notice the “1” in the relationships: standing for ‘one unique patient’. The lines indicate the relationships between the tables – remember ... relationships are the core of relational databases. You’ll see that there are always two lines – one for chart\_number and one for clinic\_code between tblDemo and the other tables. The effect of this is you typically need to specify both to identify records related to any individual patient. Once a patient is in table

“tblDemo” they can have many labs, visit, service, med and health condition records stored in the database (the infinity sign).

The relationships in DEMS appear much simpler than in CDEMS, but the differences are not so great when you look at them closely. “tblVisits” is expanded into five separate tables and there is a new table added:”tlpkCrosswalk”.

Remember that we said all patient data is stored in tables with names starting with “tbl”? Well, support tables in C/DEMS all start with “tlkp”. “tlkpCrosswalk” is a support table that functions in part to limit the addition of new entries into any of the tables to items already stored in the tlpkCrosswalk table. The outcome is that you cannot remove entries in tlpkCrosswalk unless there are no records using that raw code and you cannot add any records to any of the tables unless the raw code is already in “tlkpCrosswalk”.

So, if you opened “tblHealth” and tried to manually add a health condition:

- Condition=“PinkToe”
- Chart\_number=“9zzzx”
- Clinic\_code= “utopia”

You would probably be blocked by CDEMS, because it’s unlikely you’d have a health condition of “PinkToe in tlpkCrosswalk or a patient in the “Utopia” clinic with chart number “9zzzx”.

The C/DEMS developers have frozen the core table structure of C/DEMS. The functionality of CDEMS requires only the current elements. This means that any additions you make to any of the tables will not conflict with future features released for CDEMS. You can add anything you wish to the table structure without fear that you will break it! Always add – never change anything!!



## **Training Exercise One**

**A preliminary set of activities**

**Overview of some simple techniques:**

We will

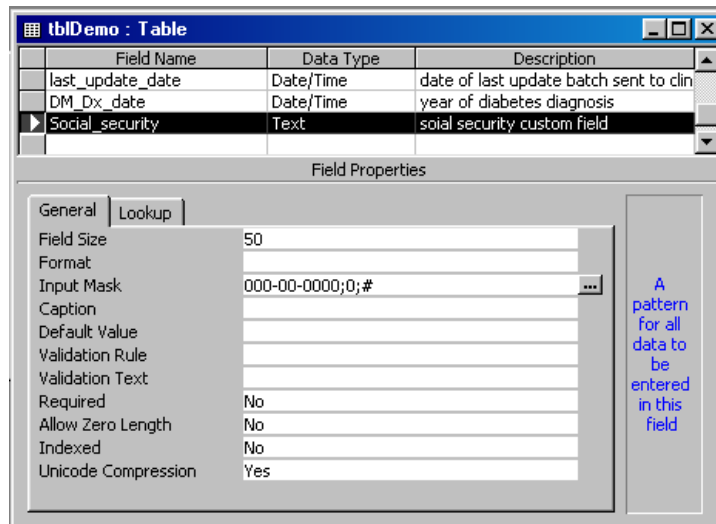
- Add a new field to tblDemo
- Use an input mask on the new field
- Create a validation rule
- Create and run a one table query
- Create an update query with complex criteria
- Filter the results using an “and/either-or” criteria
- Use the switch function (optional)
- Zooming the design window for a cell (optional)
- Add a wildcard to the criteria
- Create a mailing label using the titles.

### Create a social security number field in tblDemo

It's relatively simple to add another field in one of the tables. For instance, maybe you want to collect and store every patient's social security number. All you'd need to do is open tblDemo in design view and add the new field into the table design (you'll need to be in the 'dta' table to do this)

Be sure you're in either "dem\_dta.mdb" or "cdems\_dta.mdb"

From Database view->Go to Tables->tblDemo->Design. Scroll to the bottom of the field list and make the following entries:



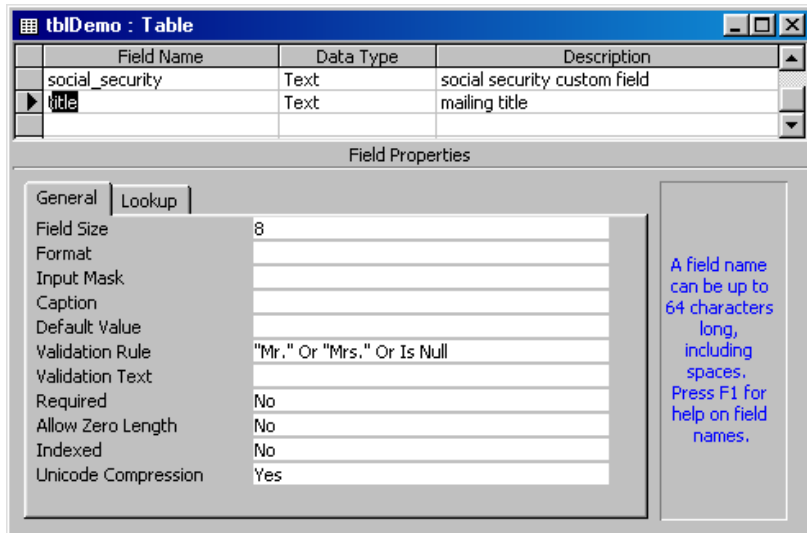
The patient data is available to all components of CDEMS, but is physically located in the "dem\_dta.mdb" or "cdems\_dta.mdb". Design changes to the tables can only be made in these files.

It may seem to you that you could do this in any of the patient "tbl" tables, and you're right. Providence in Olympia, created a quality improvement program around improving self-management goals. They stipulated each goal should have a frequency, duration and likelihood of success measure.

All these fields were added to the design of tblCustom (where text goals are typically stored). Providence then added an additional field to store an administrative rating of how well the goal measured against an 'ideal' goal. They used this rating field to track if providers over time improved in creating 'quality' goals.

### Create a “title” field in tblDemo

Add another field “title” to tblDemo. This will hold the gender-based title used in addresses. Notice the “validation rule” entry.



### Listing names for all patients in tblDemo.

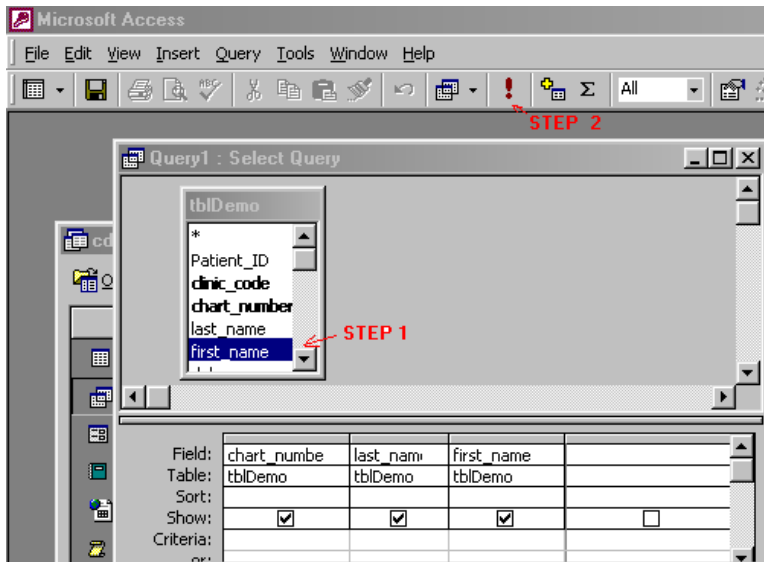
This is the simplest query of all time.

Database view-> Queries->New->OK

From “Show Table” window select tblDemo by double clicking on it -> Close

Double click on the following fields: “chart\_number”, “last\_name”, “first\_name”

Click on the red asterisk in the toolbar->a list of all patients will appear.

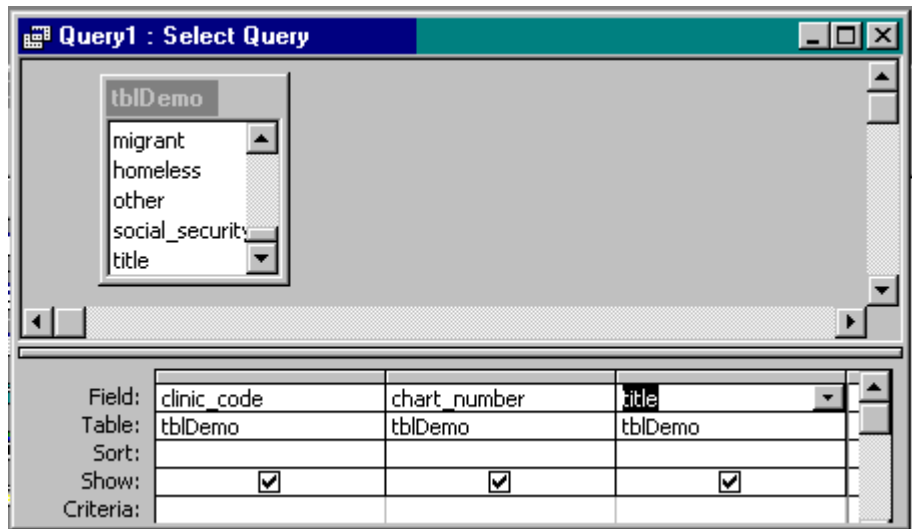


The three columns you selected will appear and each patient in table

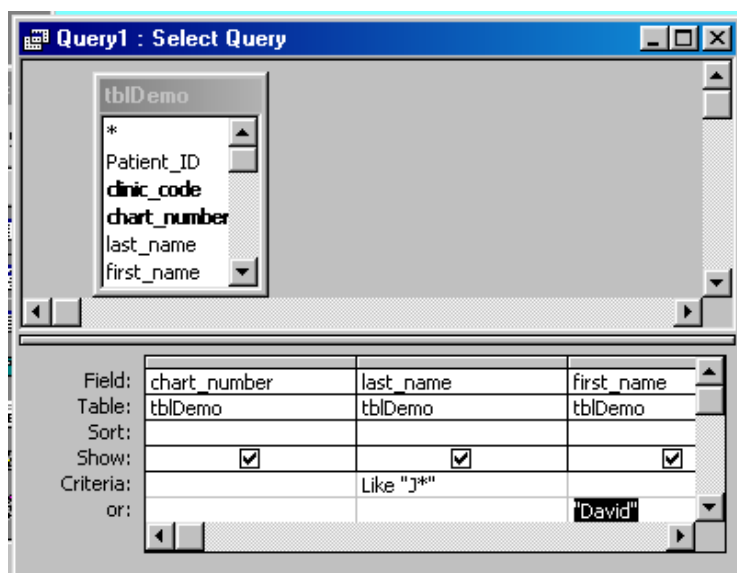
tblDemo will be displayed.

Do NOT close the list window but instead go to “View” in the menu bar and then Design View. This will return you to the original window you used to create your first query.

Try adding the new fields to this queries design: “social security” and “title”. They are available but probably have no values stored in them. To store a social security number for a patient, just open tblDemo and type in the social security field for a patient (remember we have a input mask established – so you cannot enter anything you wish!)

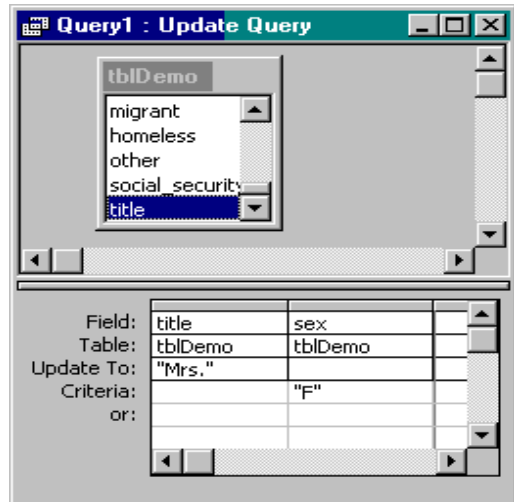


### Adding simple criteria to a query



column. Criteria can be entered using normal search syntax or using special database features.

So, wildcards are fine (e.g., “F\*”) returns all patients with last name having first letter “F”). Criteria on the same line are additive – the listed patients must meet all the criteria. Criteria on different lines are “and/or”. A patient



having any of the criteria are included. This example would return all patients with first name of David and first letter of last name of “J”. Criteria are not case sensitive.

For the mailer example you’d probably restrict on the provider field (pcp) or on some other more complex criteria. You may be interested in giving the mailer only to Spanish language patients or had those with a recent education class.

**Update Query to fill “title” field with “Ms.” Or “Mr.”**

You can fill all or some of the cells in a field using an “Update query”: Queries->New->Ok->Select tblDemo->Close-> Select Queries in menu bar->Update Query

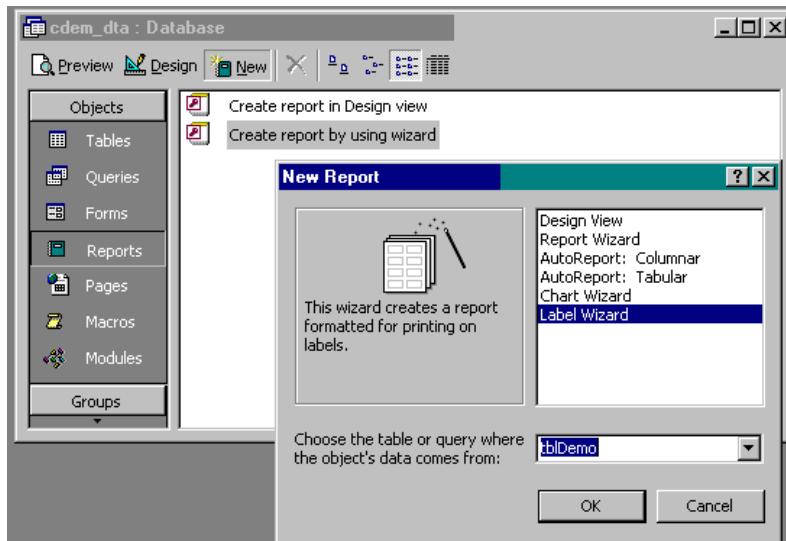
Now to fill in “Ms.” in the “title” for all the patients with “F” stored in the tblDemo “sex” field you’d simply create the following query. When you run this by clicking on the red asterisk icon, you’ll find that all the patients recorded as females in C/DEMS will have the “title” field filled with “Ms.”. Delete queries work just the same way! These are called “Action Queries” because they modify the underlying tables

We have used the criteria line to specify that only records with “F” in the sex field will be updated. You could repeat this query for patients recorded as males by clicking on View->Design->change the “F” to “M” and “Ms. to “Mr.”-> rerun the query by hitting the red asterisk.

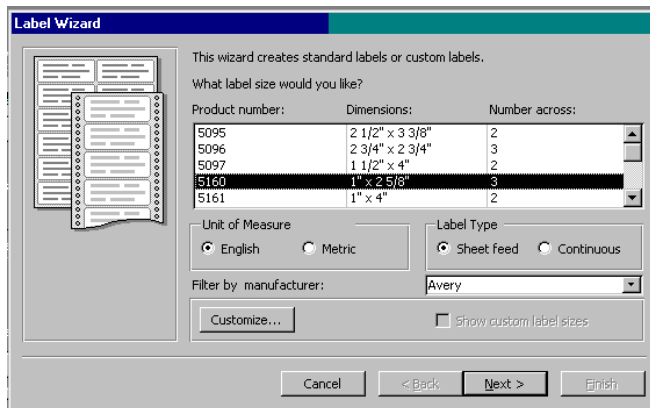
**Creating a label using the Report wizard**

Once you have a title field in tblDemo – you can make a label simply by using tblDemo as your database for the labels, and creating the labels in word or use the MS Access label wizard.

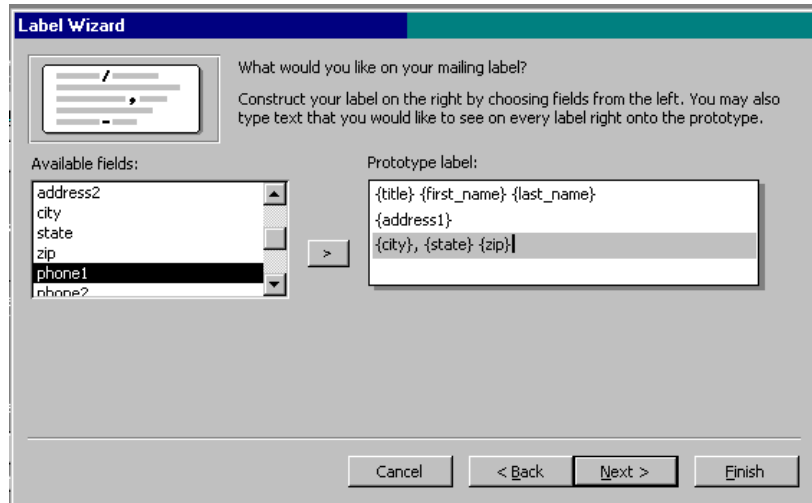
### Step 1: Reports->New->Label Wizard



### Step 2: On next screen pick Avery 5160 labels.



Step 3: Use the default fonts by clicking on "Next". Advance to the layout screen. You can select items to place on a label from the available fields by double clicking on them. Insert commas, spaces and new lines as you normally would in a text editor.

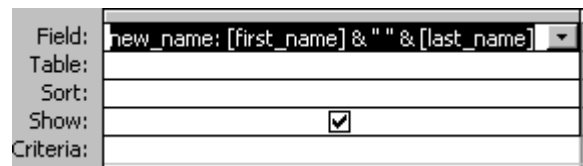


### (Optional) Alternate advanced method of updating “title” field

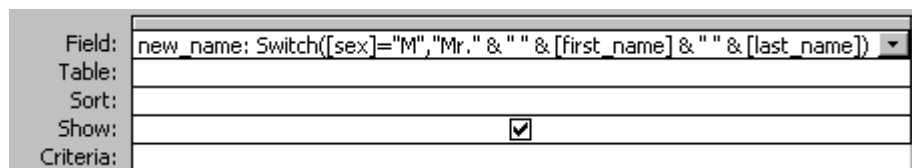
This requires pulling the last\_name and first\_name field together and assigning the title according to the entry stored in the “Sex” field using VBA functions.

Step 1: create the following column in a new query:

This creates a new alias field called “new\_name” made up of three parts: the first name, a blank space (quotes with a space between) and the last name.



Step 2: We want to add a “Mr.” In front of the name, if the patient is a male and a “Ms.” if a female. Modify the cell to look like this:



This creates the name in format; “Mr. + space + First name + space + last name” for males, with the appropriate spaces, but not for females.

The query works because of the ‘switch’ routine - - an example of

Switch routine has this format:

Switch(test1, return if test1 true, test2, return if test2 true,...). The items are tested from left to right and if more than one test is true, the routine returns the first item that is true. If any of the tests yield an error the query will not run.

the advanced syntax possible in queries.

Step 3: Adding to the code the test for if the patient is a female creates this rather ungainly cell.

Field:	new_name: Switch([sex]="M","Mr." & " " & [first_name] & " " & [last_name],[sex]="F","Ms." & " " & [first_name] & " " & [last_name])
Table:	
Sort:	
Show:	<input checked="" type="checkbox"/>
Criteria:	
or:	

Makes you hope to have a way to better see the results – like a ‘zoom’ option:

Zooming for editing: If you click in the new cell anywhere, and press the Shift+F2 keys the cell editing is much easier

You can use the code we just created to create a select query which has a new “alias” field along with the other information needed to do a mailing label. Just add the address, city, state and zip code fields to the current select query. Save this select query and use it as the source database in creating labels..

### (Optional) Advanced query to return names in

#### “Mr. Fred Finkelstonless” format

You can return values that have been transformed according to your rules. Maybe you want to do a list with the names in a new format. To create a mailer, you want the names in the list to include a title in address format like: “Mr. David Jones” or “Ms. Ginny Heart” instead of last\_name and first\_name in separate columns. This requires pulling the last\_name and first\_name field together and assigning the title according to the entry stored in the “Sex” field.

Step 1: create the following column in a new query:

This creates a new alias field called “new\_name” made up of three parts: the first name, a blank space (quotes with a space between) and the last name.

Field:	new_name: [first_name] & " " & [last_name]
Table:	
Sort:	
Show:	<input checked="" type="checkbox"/>
Criteria:	

Step 2: We want to add a “Mr.” In front of the name, if the patient is a male and a “Ms.” if a female. Modify the cell to look like this:

Field:	new_name: Switch([sex]="M","Mr." & " " & [first_name] & " " & [last_name])
Table:	
Sort:	
Show:	<input checked="" type="checkbox"/>
Criteria:	

This creates the name in format; “Mr. + space + First name + space + last name” for males, with the appropriate spaces, but not for females. It works because of the switch routine.

Switch routine has this format:  
Switch(test1, return if test1 true, test2, return if test2 true,...). The items are tested from left to right and if more than one test is true, the routine returns the first item that is true. If any of the tests yield an error the query will not run.

Step 3: Adding to the code the test for if the patient is a female creates this rather ungainly cell:

Field:	new_name: Switch([sex]="M", "Mr." & " " & [first_name] & " " & [last_name], [sex]="F", "Ms." & " " & [first_name] & " " & [last_name])
Table:	
Sort:	
Show:	<input checked="" type="checkbox"/>
Criteria:	
or:	





## Training Exercise Two:

### Conducting a cohort analysis

#### Using “start\_date” in CDEMS

All clinics want to know how they are doing with their improvement efforts, but it is not easy to monitor.

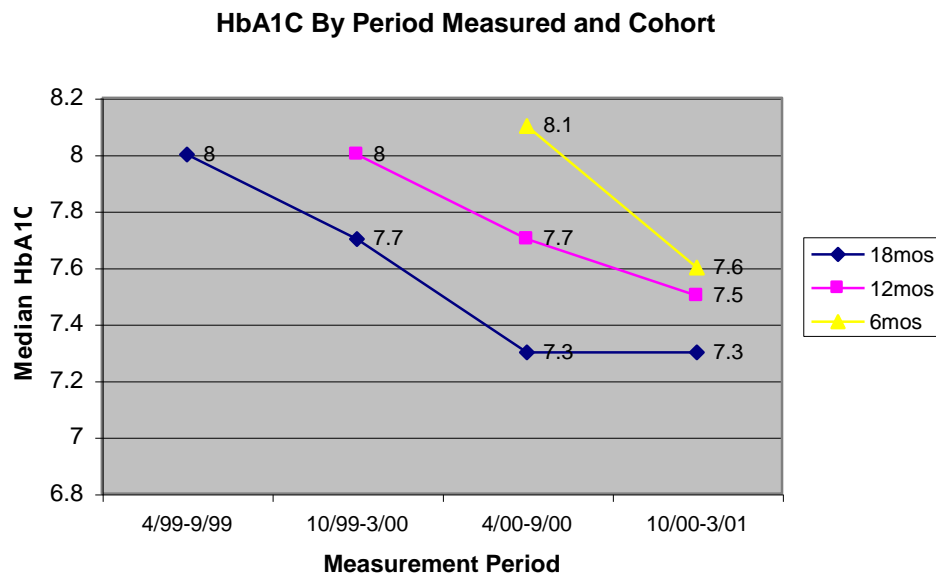
One difficulty is that clinics are always adding new patients and the new patients typically are not as ‘under control’ with their conditions as those patients who have had intensive case management supported by a registry.



So it is not uncommon for clinic-wide run charts to actually show worsening or flat results over time after installing C/DEMS! Pretty discouraging.

One solution is to periodically run reports which display how a group of patients who started together are doing over time. In CDEMS when patients entered the registry is a required field, but not in DEMS – so this work is much easier in CDEMS.

When I did this for the 14 Seattle Community Diabetes Intervention clinics I got the following graph, that graph displays cohorts which began using DEMS during the same 6 month periods, and tracks each cohort forward at six month intervals.



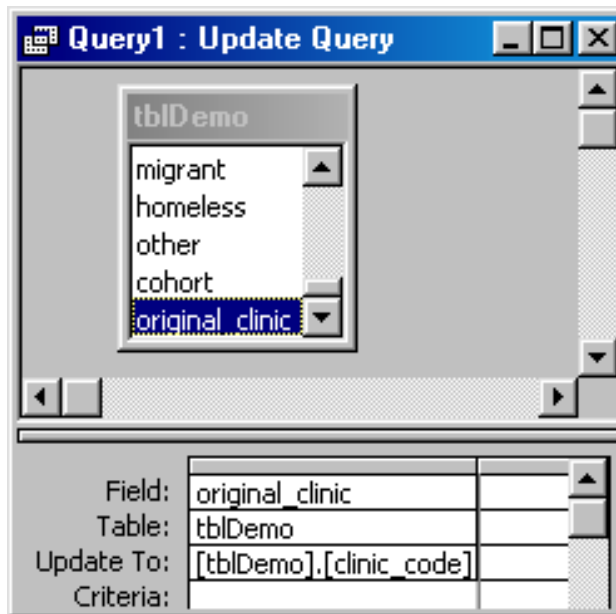
We will walk you through the steps to do this analysis at your clinic.

- Create new fields to store the cohort information
- Use update and make table action queries
- Use joins in advanced queries
- Finding Median lab value (your intro to undocumented mysteries of C/DEMS!).

### Create new fields to store cohort information

Step 1: We will add a two new fields to tblDemo to store the date the patient: Add two new fields to tblDemo: “cohort” and “original\_clinic”. Use the default “text” data type.

Step 2: Create and run the following “action query”. Action queries actually modify data.



To create an action query: Queries->New->OK->double click on “tblDemo” ->Close->Select “Queries” in menu bar->Update Query. Running this query will copy the clinic\_code for each patient into the “original\_clinic” field you just created. Notice the entry in the “Update to cell”, “[tblDemo].[clinic\_code]”. This is an example of qualifying a field, that is, to remove ambiguity about where the data is to come from. (Microsoft calls this “disambiguation ☺”)

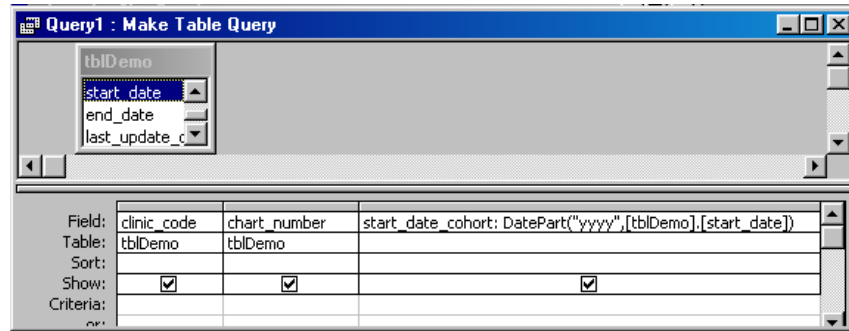
Fully identifying a field: Fields in Access sometimes need to be fully identified. The syntax is to enclose the field with square brackets and precede it with the host table name also enclosed in brackets and followed by a period (e.g., [tblDemo].[last\_name]).

### Making a new table.

Step 3. Identify the start\_date of the patient and store the information in the “cohort” field for the patient. We will store the calendar year they started in CDEMS, but it takes two steps because of a quirk in Access. We’ll first create a new table and then pass the data back to tblDemo:

Queries->New->OK->Close->Select tblDemo->Queries (from menu bar)->Make Table->name new table “Cohort” in “current database”->double click on chart\_number, clinic\_code, start\_date.

Edit the entries so they look like;



Step 4. Run by hitting the red asterisk, and you'll have a new table named "Cohort"

DatePart is one of the built-in functions of Access. To get more information on using it go into Access Help. (Visual Basic Help is an option during Access install and is indispensable.)

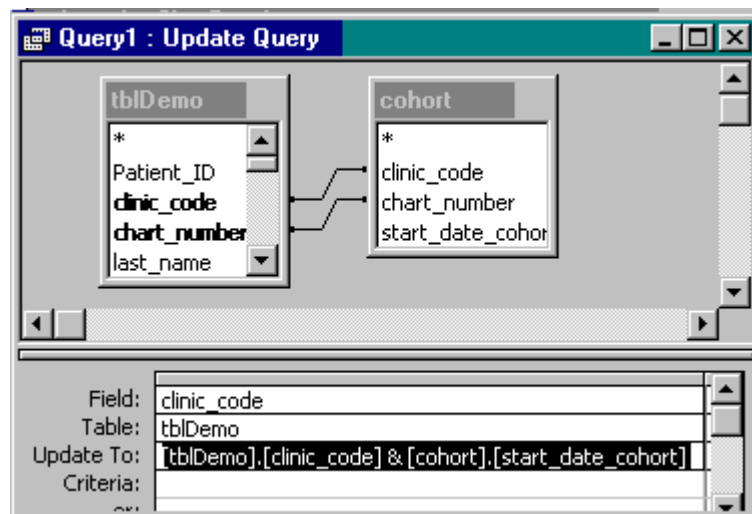
DatePart works by returning what you specify from fields with a data type of "date" in table design view. You can even specify the routine to tell you the day of week a date falls on.

Typical data types are Text (stores any characters you enter), integers (store whole numbers like 1, 2, 8) and single (stores numbers with a decimal component like "1.2") You cannot do DatePart on a non-date entry. Functions working on fields with incorrect data types are one of your biggest challenges.

Step 5. We will edit the current clinic codes so that they are modified by the year started in the registry. Clinic\_code "Pilot" will become "Pilot1999" if the patient started in 1999.

New Query: Query->New->OK->Click on tblDemo->Add -> Click on Cohort->Close->Query->Update

Make the Query look like this:



Two new ideas to take in here. The first is that you can add new tables or even the results of queries to the design of a new query. Here we added the 'cohort' table we created in the last step. The second is that if there are two data sources in a query – you need to

tell C/DEMS what the keys are that tie the two tables together – in C/DEMS these will almost always be, the at a minimum, the clinic\_code/chart\_number keys. You tell Access how the “cohort” and “tblDemo” tables are related (“joined” in database terminology) by clicking on the chart\_number field in the tblDemo table and

Joins in databases are of three basic kinds: inner, left or right. They are the basic building block for all but the simplest one-table queries. A join like we just created is term a straight or inner-join. To be included in the results of the query a patient would have to be present in both data sources. Inner joins are represented as straight lines.

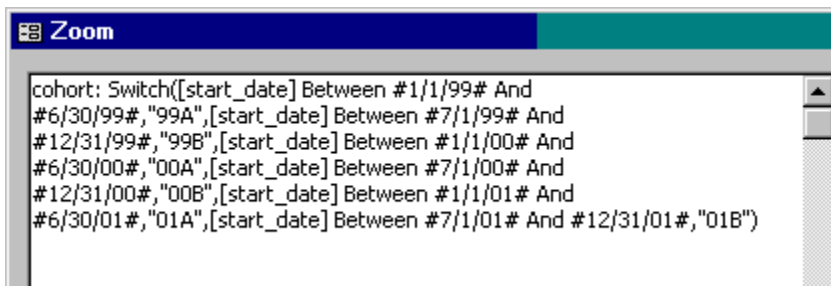
More often we will create left joins from one to many additional tables or queries. You create a left join by dragging a straight join then right clicking the mouse on the line. The pop-up window allows you to specify the type of join to create.

dragging over the chart\_number field in the “cohort” table.

Step 6. Add the new cohorts to your tlkpClinic table as new clinics. Now you can run the Diabetes Summary Reports for each cohort at 6 month intervals, or use the next step to calculate medians.

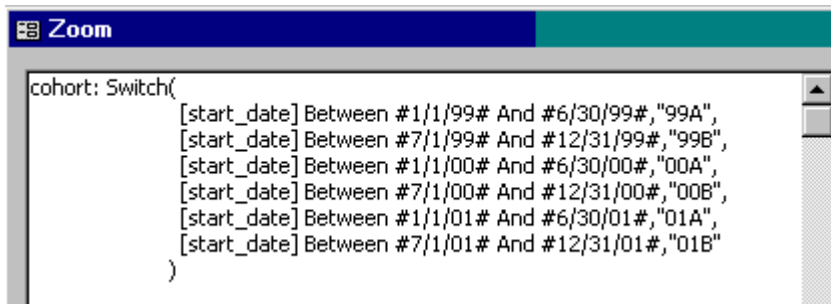
### 6 month cohorts instead of calendar year.

To make your cohorts for 6 months time periods you’d substitute the steps below for step 3 above:



```
cohort: Switch([start_date] Between #1/1/99# And #6/30/99#,"99A",[start_date] Between #7/1/99# And #12/31/99#,"99B",[start_date] Between #1/1/00# And #6/30/00#,"00A",[start_date] Between #7/1/00# And #12/31/00#,"00B",[start_date] Between #1/1/01# And #6/30/01#,"01A",[start_date] Between #7/1/01# And #12/31/01#,"01B")
```

This may look intimidating, but what is happening is very simple minded. Look at the same query cell in a different format:



```
cohort: Switch(
    [start_date] Between #1/1/99# And #6/30/99#,"99A",
    [start_date] Between #7/1/99# And #12/31/99#,"99B",
    [start_date] Between #1/1/00# And #6/30/00#,"00A",
    [start_date] Between #7/1/00# And #12/31/00#,"00B",
    [start_date] Between #1/1/01# And #6/30/01#,"01A",
    [start_date] Between #7/1/01# And #12/31/01#,"01B"
)
```

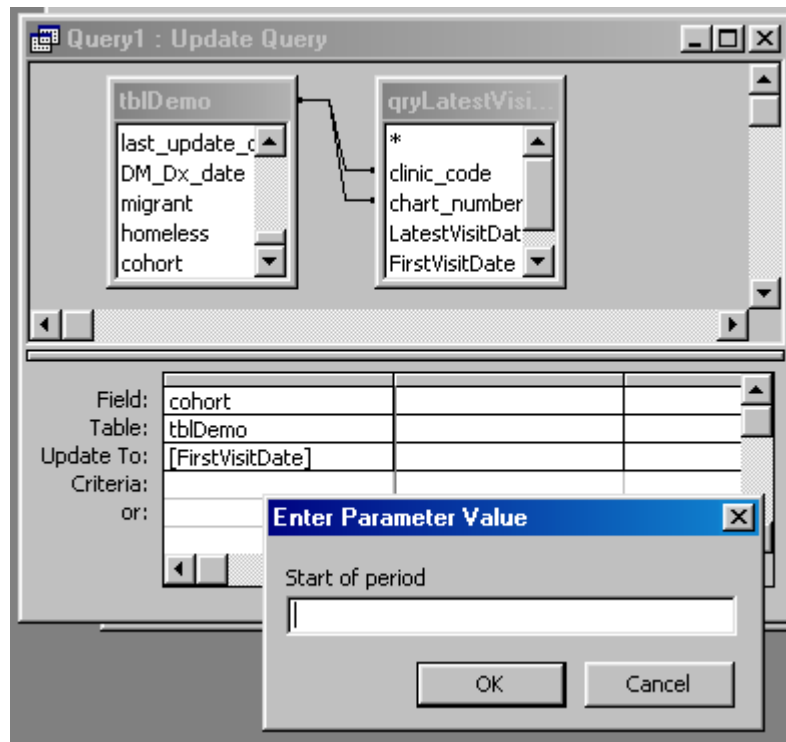
The switch command is looking from left to right for the first true statement and then returning the appropriate string. For instance a start date of 1/2/99 would be assigned to cohort “99A”. The syntax is critical. The “#” signs tell access the field is stored as a date and the double quotes marks indicate a string is being returned.

## Finding a Median

In CDEMS you can go into “qryMaxLabAndResultInPeriod” filter the results so that only half results are displayed-> sort ->and scroll down to find the record half way down the list. The result displayed is the median! You have to have the reports Splash form open and to select the clinic and time period before you run the query. It gets these values from the main Reports screen.

## In DEMS

Step 1 is finding the equivalent of the start\_date field in CDEMS. This would be the earliest date a patient had a lab, office visit or was recorded as starting in DEMS. One available query is “qryLatestVisitDate”. It is listed in the Key DEMS Queries at back of this booklet.



Joining it to tblDemo in a new update query will give the equivalent step 1 described above for CDEMS.

To calculate the median in DEMS you could  
Run “qryDemographics\_Visits\_Tests(Part1)”  
Sort on the column labeled “HbA1c”  
Count down half way through the nonblank rows.





## Surviving in Query Land Jungle

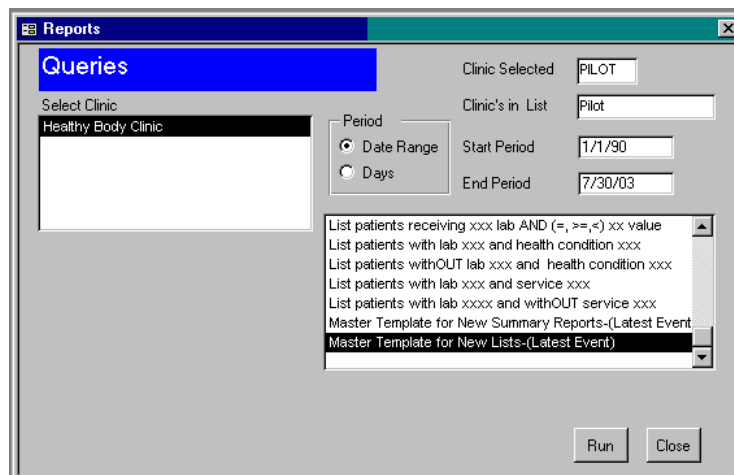
### Using Help, Key Skills

C/DEMS is written in Visual Basic for Applications, run inside Access 97, 2k or 2002, which runs inside Windows on your computer which is possibly connected to a Novell, Banyan or Microsoft network and printing on any of a variety of printers. We have marvelous opportunities because of the rich Windows/Access Environment but sometimes have trouble. When you install Access, install all the VBA help files – this will give you access to a valuable help resource.

In C/DEMS there are over two hundred queries already written and stored in the respective report writing packages (QLIB\_mdb, DEMS; or RPT\_mdb, CDEMS). I've attached a summary for some of these queries.

### Using the Custom List Wizard in CDEMS

For CDEMS users there is a wonderful wizard that can be used as a starting point for custom queries. From RPT choose the custom list creator:



When you craft a query using this tool it is stored as “temp” in the queries collection. It stays in the collection till you write a new query with the wizard. If you like a query – you can simply rename temp to another name and it will be available for your use.

You rename the “temp” query by right clicking on it->Rename and edit the name to something new. This query can then be the start of your own custom queries.

If you write custom queries name them so that they stay separate from the core built-in queries. ie; preface them with sqry or something unique.s allows you to import them to any new RPT or QLIB program which come come out.

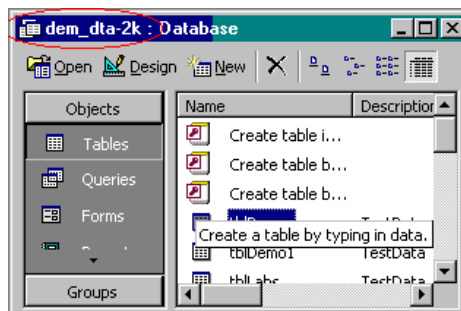
### Adding complex string criteria to a query

Simple string criteria are familiar to Windows users, but many more options are available inside Access. Using “Jones” as a criteria is an example of a “string” criteria (Sometimes called “text” – these are any printed characters not just numbers or dates). You can do much more with strings, dates and numeric criteria. Some of the possibilities you may not even be able to imagine a use for.

Example query functions:

Function	Example use	Comment
Len(field)	new1: Len([last_name])	This is the key function for dealing with strings. Entering this line in a query will give the length of each name in C/DEMS. (“Johnson” = 7)
Left(field)	new1: Left(last_name, 5 )	Returns the left “x” most characters of the field. (x=5; “Johnson”=”Johns”)
Right(field)	new1: Right(last_name,5)	Returns the right x characters of the field. (x=5; “Johnson”=”hnsn”)
Mid(field)	new1: Mid(last_name,2,3)	Returns the middle y characters starting at position x. (x=2,y=3; “Johnson”=”ohn”)
Instr(“x”,field)	new1: Instr(“oh”,[last_name])	Returns the position where the “oh” is located in the second field. (“Johnson”=2)
between (criteria)	between 1 and 2	used as criteria, Returns values that fall in the range specified.
IsNull	IsNull([Last_name])	select only records where value is absent for variable

### How to determine which file you have open:



You can tell by pressing the “F11” key and looking in the blue title bar for current file name.

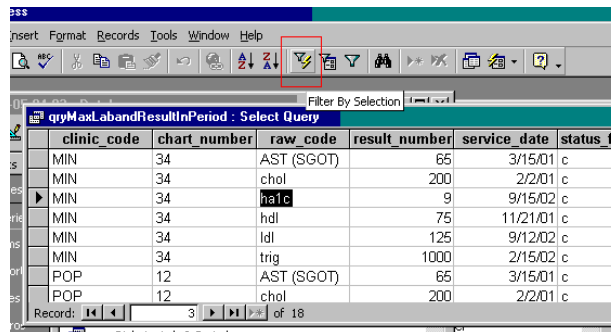
### How to get into Database view.

The F11 key will take you to the native database view in MSAccess. The database view is where you can create new queries, look at the tables directly or edit any forms, reports, queries in C/DEMS.

Alternatively Window->Unhide->OK takes you to the Database view.

### How to filter a table

To display only certain records you only need to click on any like cell and then the filter icon. This example will reduce the listed items to only HbA1C labs. removing the filter is easy also: Records->Remove Filter.



## KEY DEMS Queries in QLIB

Query	Function	Input	Comment
qryLatestVisitDate	Returns the first and last visit date in reporting period	tblVisits	Part of query set that drives the Diabetes Summary Report
QRY_40YearsOld_Aspirin	Returns patients with med_aspirin field at last visit=yes and age >39.	tblDemo QRY_Last_Visit_Date tblVisits	The particular age calculation used calculates age using difference between current year and year born so may identify patients as 40 yrs old as much as 11 months early (e.g., dob in Dec and today's date is following January=age would be 1 yr!
QRY_40YearsOld_No_Aspirin	All patients other than those with med_aspirin at last visit and age >39.	tblDemo QRY_40YearsOld_Aspirin	(Same age as QRY_40YearsOld_Aspirin note.)
QRY_Last_Visit_Date	Returns last visit date	tblDemo tblVisits	Only includes patients with at least one visit.
QRY_Last_Visit_xx_Days_AndOver	Returns last visit date that was before today's date less user entered number of days	Prompts for an integer ([days]). QRY_Last_Visit_xx_Days_AndUnder tblDemo	Label says on or before, but really patients with last visit more than xx days and patients which never had a visit are listed!
QRY_Last_Visit_xx_Days_AndUnder	Returns last visit date that was on or after today's date less user entered number of days	Prompts for an integer ([days]). QRY_Last_Visit_Date	For example entering "1" gets you everyone with visit yesterday or today.
QRY_Latest_HA1C_Date&Result	Last HbA1c lab date and result for each patient	QRY_Latest_HbA1C_Date qryLabs	
QRY_Latest_HbA1C_Date	Last HbA1c lab date for each patient.	qryLabs tlkpLabTest	tlkpLabTest only needed so that lab name can be used in lab type select. If no numeric HbA1c tests for patients they are not in list.
QRY_WSDC_LatestLabValues	In period, returns for specified Diabetes labs the last result for each lab in a separate column – one row for each patient, one column for each lab (HbA1C, Microalbuminuria, Creatinine, Cholesterol, HDL, LDL, Triglyceride, 24HrUrineprotein, Microalbumin/Creatinine Ratio, Liver Function test, Clinic Lab)	User provides start and end date of period qryLatestTestLabDate qryLabs tlkpLabTest	If patient has no labs in period they are not in list.
QRY_WSDCCollaborative_1			
QRY_WSDCLatest_BP_Under140/90			
QRY1_Latest_BP_Date			
QRY1_LatestTestLabDate	Returns the last lab date for each lab and patient	qryLabs	Similar to qryLatestTestLabDate , but no start and end date. Has useless criteria to select only records with non blank results (qryLabs records cannot have such a value anyhow).
QRY1_LatestVisitDate	Returns the first and last visit date for each patient in the previous 730 days from date query ran.	tblVisits	If no visit in those 730 days – patient not listed
qryLabs	Contains all lab results converted to numbers. If leading "<" or ">" it is stripped. So ">30" becomes "30".	tblLabs	This table - - not tblLabs - -should be used if you are comparing results against any numeric criteria. Non-numeric values and results of "9999" are dropped from table.
qryLatestHA1CDate	In user provided period returns the first and last HbA1c lab date.	User provides start and end dates qryLabs	Has useless criteria selection on results having nonblank value
qryLatestTestLabDate	In user provided period returns the last lab date for each lab and patient	User provides start and end dates QryLabs. If patient had nonnumeric or blank result this qry would not report it.)	There is a criteria to include only non-blank results which is useless because to be in qryLabs you must have numeric result
qryLatestVisitDate	In user specified report time period returns last and first visit date and number of visits in period	tblVisits	Used in the DM Summary Report

## CDEMS Queries

QRY_LatestVisitDate	For reporting period, returns 1 <sup>st</sup> and last visit dates, count of office visits	Office visits are typed as "o" s in tblVisits. Users can track several types of visits, but only O's are used for many of the reports.
qryLabs	Returns only lab results in numeric format. Strips leading "<" and ">" characters. (e.g., <30 = 30)	Useful for queries that have numeric criteria. Any lab conducted that returns non-numeric or '9999" result is discarded
qryLabs2	Strips leading "<" and ">" from numeric results. Returns all results, new T/F field indicates if numeric result	Couple with criteria selecting only "True" records from the "Is numeric" field. Can do calculations and count total labs done – even those with non-numeric results like "abnl"
qryMaxHealthDateInPeriod		
qryMaxLabandResultInPeriod		
qryMaxLabandResultInPeriod2	Returns the last date and result for every lab	Returns numeric and non-numeric results, and a is numeric field which can be true/false. This allows users to seect only numeric records results for math
qryMaxServiceDateAtEndOfPeriod	Returns for each patient last date they ever were recorded to have a service as of the end of the reporting period	Useful for determining if Pnevax given and other similar long- acting services.
qryMaxServiceDateInPeriod	Returns for each patient the last date each service was completed in period.	If patient had no services they'd not be in list.
temp	Stores the custom designed query from the RPT main screen	This is a great starting spot for a custom query