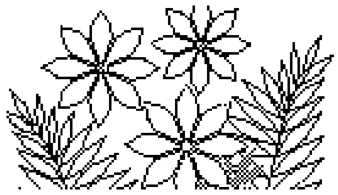


Chapter 8
Relational Tables in Microsoft Access

Objectives
This chapter continues exploration of Microsoft Access. You will learn how to use data from multiple tables and queries by defining how to join records from those separate tables and queries. You will also learn how to construct a graph or chart in Access. An important aspect of the chapter is to recognise features of the SQL statements underlying the queries that you create.

This chapter will cover:
- Importing data into a database table
- Creating relationships between tables
- Creating relationships between a table and a query
- Creating a chart
- Recognising features of SQL statements

Preparation
Read the whole of the chapter before beginning the exercises.

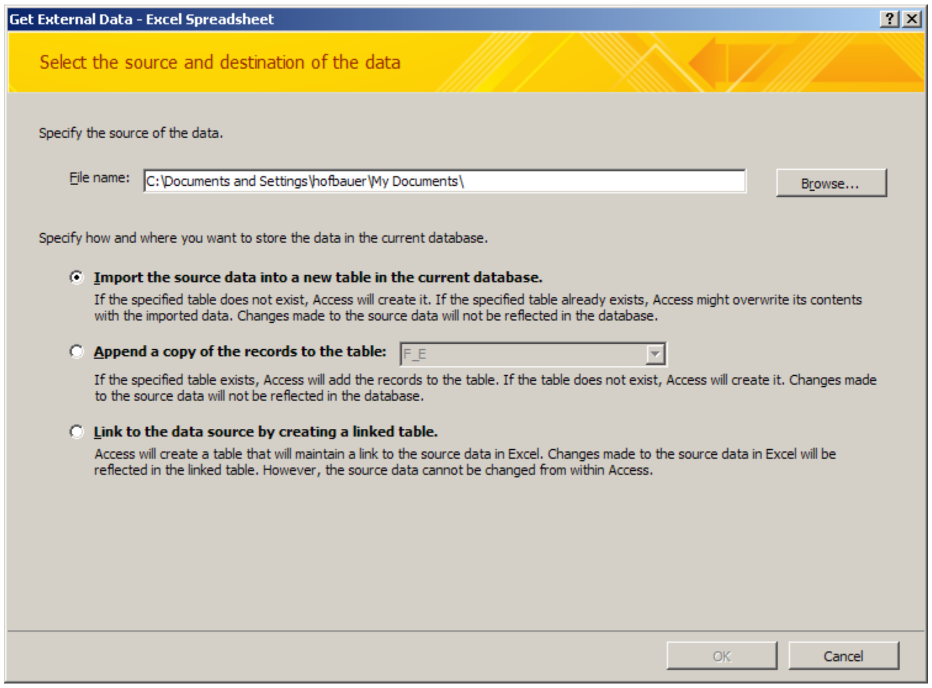
Exercise 1 – Importing Data

You must first copy the database file Ch8_Ex.accdb and the Excel file AllQuakes.xlsx, found in Support Files (Chapter 8) under Resources, to a folder you create called, say, Ch8, in your own account or on your own computer.

Open the file Ch8_Ex and you'll see a database with two tables – one called F_E and the other called Mercalli. Open these two tables in order to see what they contain. The F_E table contains a long list of numbers and associated names of countries or geographic regions of the world. The number is simply a code for the geographic region. The codes are unique, one for each region. These are called the Flynn_Engdahl regions, used by scientists who study earthquakes to specify the location of an earthquake. The Mercalli table also contains a code, this time named Intensity, and a text field that describes the effects of an earthquake with the given intensity code. This is called the Mercalli scale of earthquake effects. Your first task is to import data consisting of a large list of earthquakes.

To import the data select the External Data tab, and in the Import & Link group click on the Excel icon. A window called Get External Data – Excel Spreadsheet will appear and you should navigate to the file called AllQuakes.xlsx file that you previously copied.

Figure 8.1a – the Get External Data window of Microsoft Access


Get External Data - Excel Spreadsheet

Select the source and destination of the data

Specify the source of the data.

File name: C:\Documents and Settings\hofbauer\My Documents\

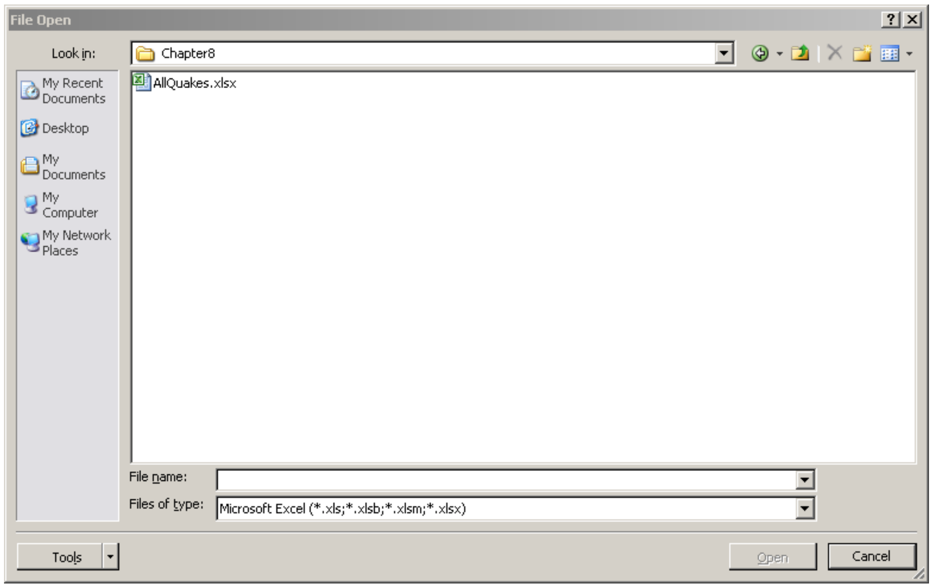
Specify how and where you want to store the data in the current database.

(●) **Import the source data into a new table in the current database.**
If the specified table does not exist, Access will create it. If the specified table already exists, Access might overwrite its contents with the imported data. Changes made to the source data will not be reflected in the database.

( ) **Append a copy of the records to the table:** F_E
If the specified table exists, Access will add the records to the table. If the table does not exist, Access will create it. Changes made to the source data will not be reflected in the database.

( ) **Link to the data source by creating a linked table.**
Access will create a table that will maintain a link to the source data in Excel. Changes made to the source data in Excel will be reflected in the linked table. However, the source data cannot be changed from within Access.

Figure 8.1b – the Get External Data – File Open window of Microsoft Access


File Open

Look in: Chapter8

AllQuakes.xlsx

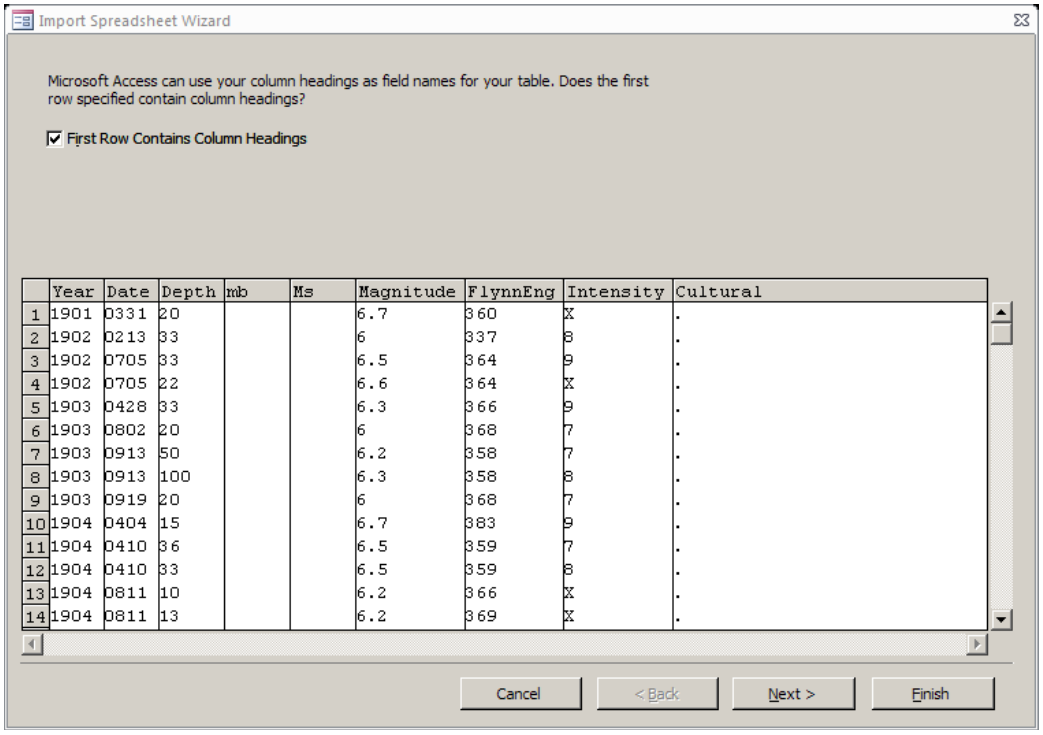
File name:

Files of type: Microsoft Excel (*.xls;*.xlsb;*.xlsm;*.xlsx)

The file AllQuakes(.xlsx) is an Excel file that has columns of data labeled Year, Date, Depth, mb, Ms, Magnitude, FlynnEng, Intensity and Cultural. Click on the Open button once you have selected the AllQuakes file and then click on the OK button. The window shown in Figure 8.2 will appear. Before proceeding, verify the First Row Contains Column Headings check box has a check mark.

Notice first that the date data, although numbers, is not meant to be interpreted as numbers (you would never add or multiply such values). The data is in fact representing a month (the first two digits) and a day of the month (the second two digits). The year data might reasonably be interpreted as numeric, although perhaps only if you anticipate doing arithmetic with it. The depth data is certainly numeric (the numbers represent how many kilometres the epicentre of the earthquake was below the surface of the earth). The mb and Ms values are also numeric. The mb value represents the magnitude of the body wave associated with the earthquake and the Ms value represents the magnitude of the surface wave. Note that many of these values are missing – simply because scientists were not present to make the measurements for many earthquakes. The magnitude figure can be taken to represent the familiar Richter scale and it might be treated as a number. The FlynnEng data, although digits, should not be treated as numbers (you would never add or multiply such values) and neither should the Intensity or Cultural data. Each of these are codes rather than numbers to be used in arithmetic.

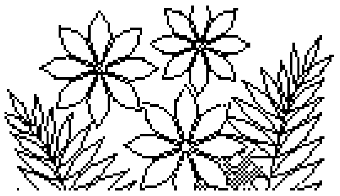
<u>Figure 8.2 – step 1 of the Import Wizard</u>



Microsoft Access can use your column headings as field names for your table. Does the first row specified contain column headings?

☑ First Row Contains Column Headings

	Year	Date	Depth	mb	Ms	Magnitude	FlynnEng	Intensity	Cultural
1	1901	0331	20			6.7	360	X	.
2	1902	0213	33			6	337	8	.
3	1902	0705	33			6.5	364	9	.
4	1902	0705	22			6.6	364	X	.
5	1903	0428	33			6.3	366	9	.
6	1903	0802	20			6	368	7	.
7	1903	0913	50			6.2	358	7	.
8	1903	0913	100			6.3	358	8	.
9	1903	0919	20			6	368	7	.
10	1904	0404	15			6.7	383	9	.
11	1904	0410	36			6.5	359	7	.
12	1904	0410	33			6.5	359	8	.
13	1904	0811	10			6.2	366	X	.
14	1904	0811	13			6.2	369	X	.

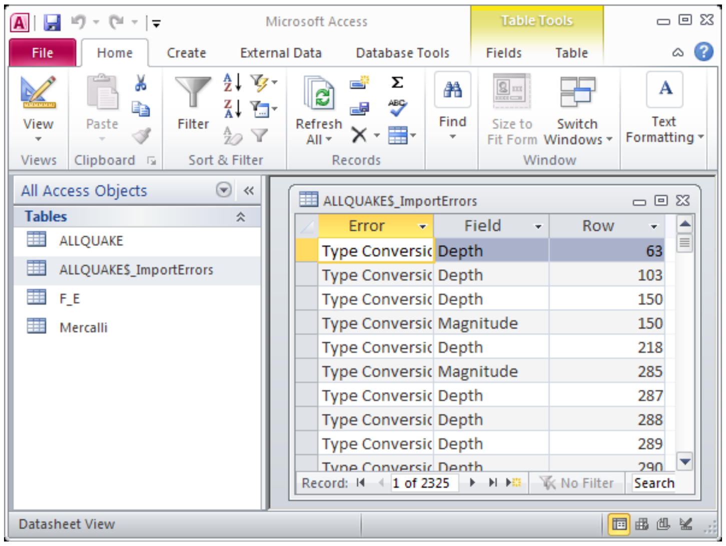


Click on the Next> button and make sure that you choose to store the data in a new table rather than an existing one. The next window allows you to change the names of the fields if necessary. In this case there is no need to change any names. The window also allows you to skip (not import) a field, to change the data type in some situations (you can't do it with this data) and to specify if a field is indexed. It is useful to index a field if it is going to be used as the basis for a search, for example. Don't index any fields just yet – we can return to do it later if necessary. Notice that the data types of mb and Ms are set as text – and that you cannot change it. We would like the values in these fields to be treated as numbers. The import wizard sets them to text because in the first record these fields are blank. You'll see that you can reset the data type in a different way later on. The data types of Date, Intensity, and Cultural are text – as they should be.

The next window of the Import Wizard allows you to specify a primary key. A primary key is usually very important because it uniquely identifies each record and is often used as the basis for joining records from one table with records from another table. Often the primary key would be some kind of identification code. For example, your social insurance number uniquely identifies you and is usually used as a primary key in databases that the government maintains. Another example is your student number, which also uniquely identifies you, for the purposes of York University. You should give some thought to other things that might be used as primary keys by organisations that maintain databases. In this case there is no field in the data that uniquely identifies each record and you should simply let Access add a primary key by checking the appropriate radio button, which Access has already done for you. Access will simply add a field, which numbers the records.

The final window allows you to specify the name for the table. The default name ALLQUAKE is adequate, unless you really don't like it. Click the Finish button and wait as the data is imported – it will take a short while because there are close to 5000 records (4686 records, to be exact) in the database. Access will tell you that there were errors importing the data and that a table containing these errors has been added. Simply click the Close button to proceed. You'll find that there are four tables in the database as shown in Figure 8.3. Select and open the ALLQUAKE$_ImportErrors table and you should see the data shown in Figure 8.3. If you examine the data in the ALLQUAKE table itself you'll see that the type conversion errors arise because of blank fields in the data. None of them are actually errors in which records have been lost. You can safely delete the ALLQUAKE$_ImportErrors table (close the table window and in the Home tab select Delete in the Records group).

Figure 8.3 – the tables and import error table after importing the allQuakes data



Error	Field	Row
Type Conversic	Depth	63
Type Conversic	Depth	103
Type Conversic	Depth	150
Type Conversic	Magnitude	150
Type Conversic	Depth	218
Type Conversic	Magnitude	285
Type Conversic	Depth	287
Type Conversic	Depth	288
Type Conversic	Depth	289
Type Conversic	Depth	290

Setting Relationships between the Tables

We have the three tables – ALLQUAKE, F_E and Mercalli – because this is the most efficient way of storing the data. If instead of having the F_E table the ALLQUAKE table had contained the name of the Flynn-Engdahl region, rather than just the code number, those names would have been duplicated many times, thus wasting space. Moreover, if we had changed a particular region name we would have to go through the whole 4686 records changing the name wherever it occurred.

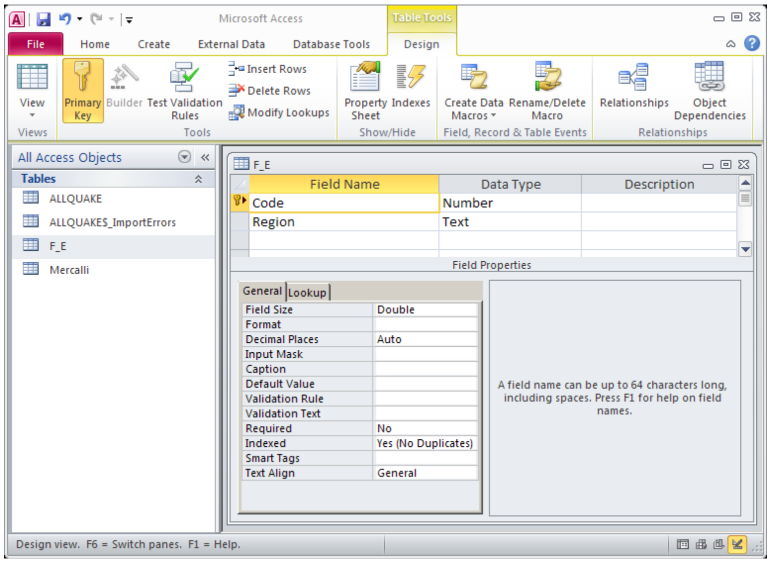
With this organisation into tables we hope to be able to link or join the ALLQUAKE and F_E tables so that the region name is available instead of the code number whenever we want to use it. The reason for the Mercalli table is similar. In that case we want to substitute the effect description for the intensity code at times.

It is for the reason of joining tables that we want to specify a primary key if possible and to specify relationships between tables.

Specifying a Primary Key

Open the F_E table. You'll see that the values in the Code column are unique and could therefore be used as a primary key. Enter the Design View mode (from the Home tab, View group) and, making sure the Code field is selected, choose the Primary Key command from the Tools group inside the Table Tools Design tab. Figure 8.4 shows how the design view window should appear after this step. The little key icon to the left of the Code field row indicates that the field has been made the primary key.

Figure 8.4 – specifying a primary key for the F_E table



Field Name	Data Type	Description
Code	Number	
Region	Text	

Field Size	Double
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)
Smart Tags	
Text Align	General

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

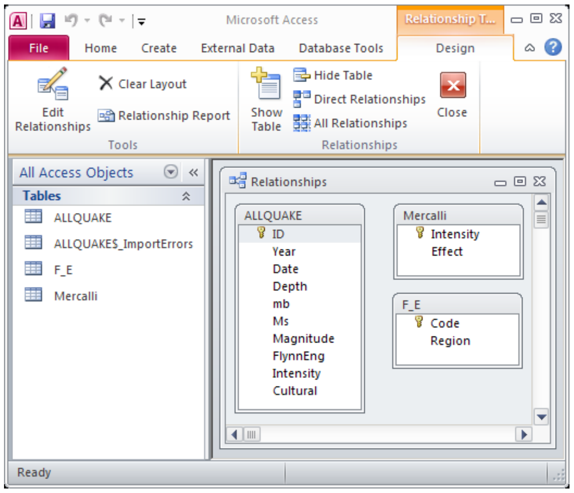
Design view. F6 = Switch panes. F1 = Help.

Repeat this process to specify that the Intensity field is the primary key in the Mercalli table. When you close the windows for the F_E and Mercalli tables you'll be asked if you want to save the changes save the changes – answer Yes of course.

Relationships Between Tables

In the Database Tools tab you will find a command called Relationships. This command results in the window shown in Figure 8.5. The Relationships window displays all of the tables in the database, along with the fields in each table. The idea is now to specify which fields in the different tables should be used to join the records of those tables.

Figure 8.5 – the Relationships window before specifying any relationships



ALLQUAKE
ID
Year
Date
Depth
mb
Ms
Magnitude
FlynnEng
Intensity
Cultural

Mercalli
Intensity
Effect

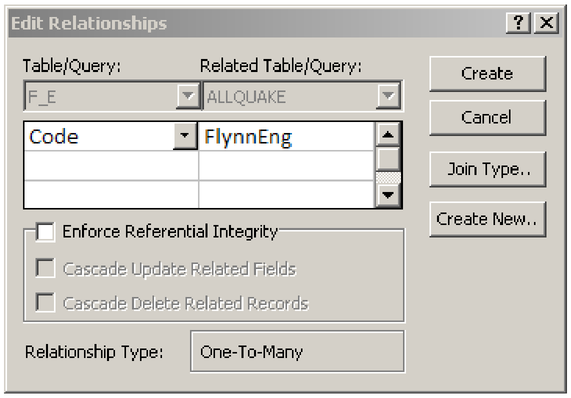
F_E
Code
Region

Ready

It is fairly obvious from the descriptions on previous pages that the Flynn-Engdahl code should be used to join the ALLQUAKE and the F_E tables. This would mean that each record in the ALLQUAKE table with a particular Flynn-Engdahl code would be joined with the same record from the F_E table. For example, records in the ALLQUAKE table with Flynn-Engdahl code 23 (region name British Columbia, Canada in this case) would each be joined with the one record from the F_E table with that code. This is called a one-to-many relationship.

Select the FlynnEng field in the ALLQUAKE table, drag it so that it is positioned over the Code field in the F_E table and release the mouse button. The window shown in Figure 8.6 will appear. Click the Join Type… button and select the radio button labeled 1, which says "Only include rows where the joined fields from both tables are equal." This means the records in ALLQUAKE will be matched (joined) with the appropriate record from the F_E table. Click OK to close this Join Properties window and then click the Enforce referential integrity check box (Figure 8.6). This is not strictly necessary but it results in some symbols appearing in the Relationships window which clearly indicate the one-to-many nature of the relationship.

<u>Figure 8.6 – specifying the Join Type and Referential Integrity</u>



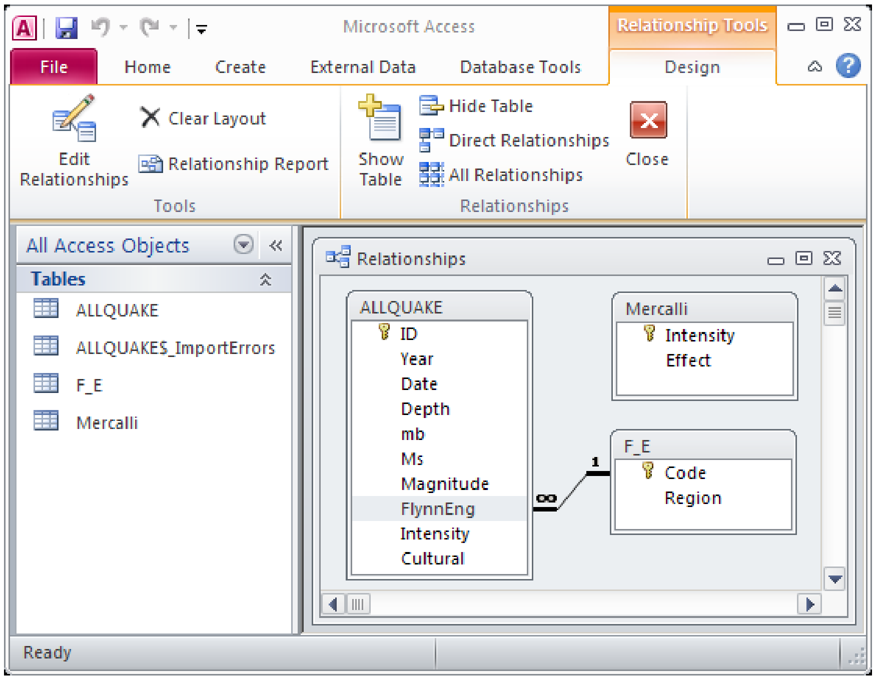
Table/Query:	Related Table/Query:
F_E	ALLQUAKE
Code	FlynnEng

☐ Enforce Referential Integrity
☐ Cascade Update Related Fields
☐ Cascade Delete Related Records

Relationship Type: One-To-Many

Click the Create button in the Relationships window and you should see the result shown in Figure 8.7. This shows a line joining the FlynnEng field to the Code field. There is an infinity symbol at the ALLQUAKE table end and a 1 at the F_E table end indicating the many-to-one (or one-to-many) relationship between the join fields.

Figure 8.7 – the Relationship between the ALLQUAKE and F_E tables



Repeat this process, except this time join the Intensity fields in the ALLQUAKE and Mercalli tables in a many-to-one relationship. You should end up with a line linking the two fields, with the infinity symbol at one end and a 1 at the other.

A Note on Referential Integrity

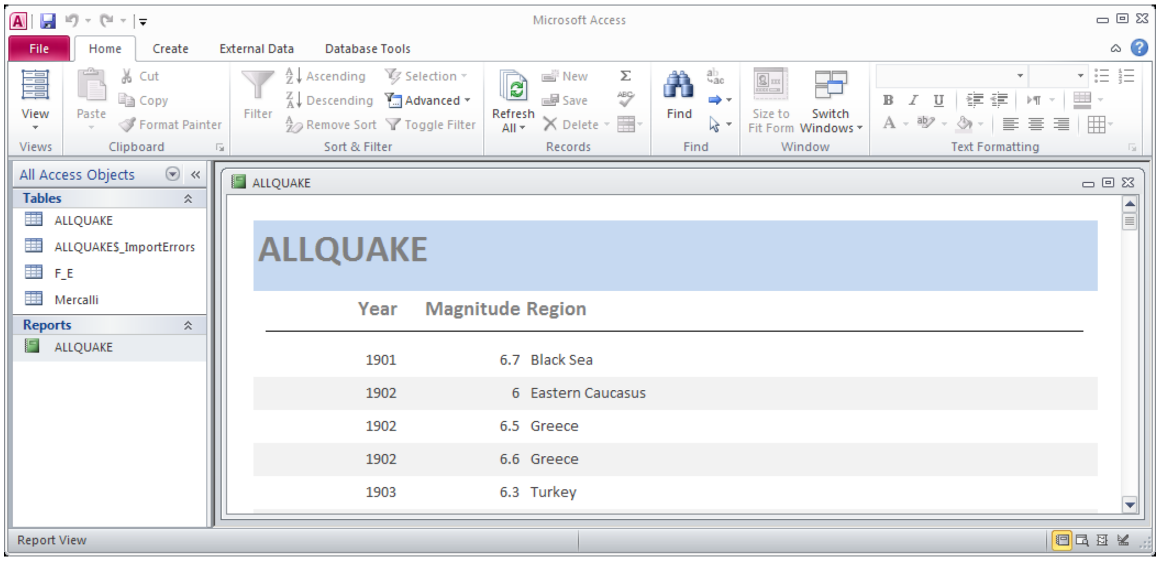
Referential integrity prevents you from modifying the database tables in ways that violate the join relationships. For example, it would make no sense to add a record to the ALLQUAKE table where the FlynnEng field had a value that does not exist in the F_E table. Referential integrity prevents this. It would also be no good to delete a record from the F_E table when there were records in the ALLQUAKE table that had FlynnEng field values corresponding to this record. By deleting the record from the F_E table you would be creating "orphans" in the ALLQUAKE table, i.e. records for which there was no matching code and region name in the F_E table.

A Simple Report Demonstrating a Table Join

Before we move on let's see the effect of establishing relationships between tables, as you have just done. Click on the Report Wizard in the Reports group inside the Create tab. Select the fields Year and Magnitude from the ALLQUAKE table, then select the F_E table and choose the Region field. You have just selected data from different tables to be included in the one report. For this test example we won't bother with any of the other features of creating a report – just click the Finish button at this point.

The report that you get (the first few records are shown in Figure 8.8) lists all of the records in the ALLQUAKE table and for each one the appropriate region name from the F_E table is joined to it. This is a simple demonstration of the reason for establishing the join relationships on key fields. (There are over 100 pages in this report **–** ***DO NOT PRINT IT.***) **Note**: your version might not look exactly like Figure 8.8. Year is formatted as a General Number.

Figure 8.8 – the Allquake report with region names from the F_E table joined



ALLQUAKE

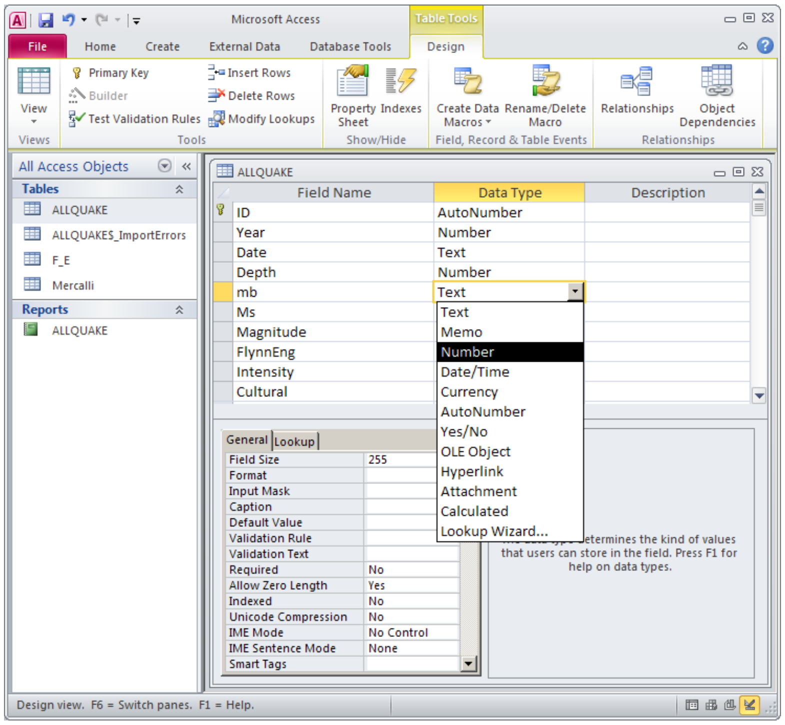
Year	Magnitude	Region
1901	6.7	Black Sea
1902	6	Eastern Caucasus
1902	6.5	Greece
1902	6.6	Greece
1903	6.3	Turkey

Modifying the Database Design

Some of the data in the ALLQUAKE table is not in a very useful form at the moment. The mb and Ms fields have been imported as text data (because the first record happened to have a blank in these fields) when they should be numeric. Changing this will illustrate that we can return to the table design to make modifications whenever we want.

Close the Report window if it is still open and return to the Tables tab. Select the ALLQUAKE table, open it and from the View group inside the Home tab select Design View. The window shown in Figure 8.9 should appear. Click in the Data Type column of the mb field row and then click on the arrow that appears in order to see the drop down list of data type choices available to you. You want to change this from Text to Number, so simply make the selection. Do the same for the Ms field.

<u>**Figure 8.9 – the Design View for the ALLQUAKE table**</u>



ALLQUAKE

Field Name	Data Type	Description
ID	AutoNumber	
Year	Number	
Date	Text	
Depth	Number	
mb	Text	
Ms		
Magnitude		
FlynnEng		
Intensity		
Cultural		

Field Size	255
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	Yes
Indexed	No
Unicode Compression	No
IME Mode	No Control
IME Sentence Mode	None
Smart Tags	

…etermines the kind of values that users can store in the field. Press F1 for help on data types.

As you can see modifying the table design is very simple. You could also add new fields or delete existing fields if necessary. In the case of a large database you should be very careful to design it correctly in the first place, however. Adding a new field might require you to manually go through the entire table entering values in each record for that field – an unenviable task even for the few thousand records in this table let alone a commercial size database.

Some Calculated Fields

It will be interesting as we analyse the data in this database to be able to display the date and the day of the week on which the earthquake event occurred. At the moment the date is contained in the two fields Year and Date where the latter is a code in which the first 2 digits represent the month and the second two represent the day of the month. We need to combine these two fields to make an actual date value, which we could then use to determine the day of the week.

Click on the Queries tab and then click on Query Design in the Queries group inside the Create tab (this should select Design View rather than one of the query wizards). You only need the ALLQUAKE table in this case, so make sure it is selected in the Show Tables window, click the Add button and close the window. You should have the window show in Figure 8.10.

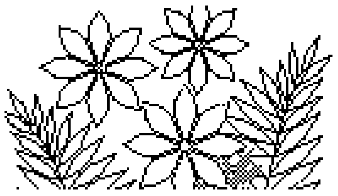
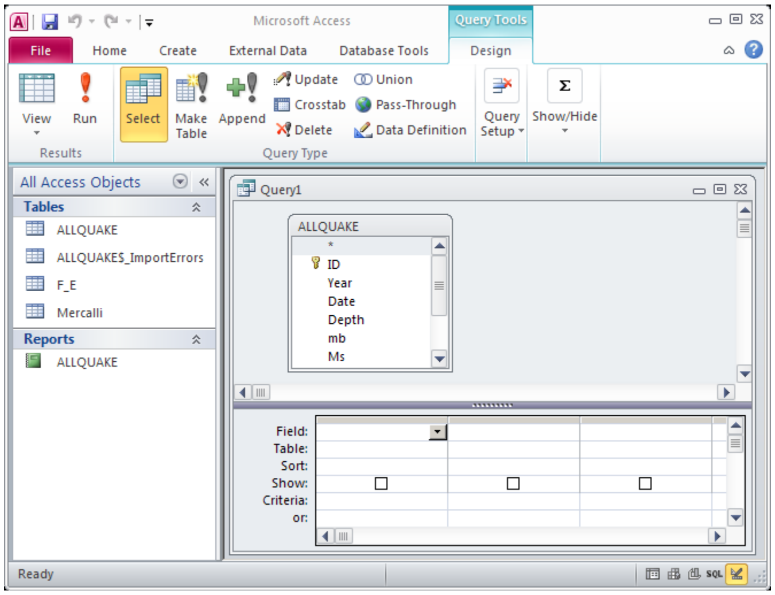


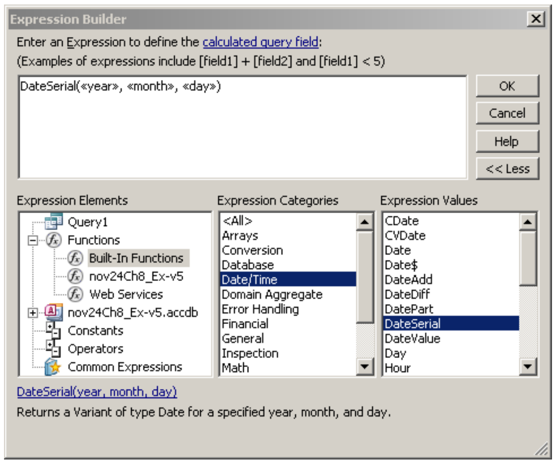
Figure 8.10 – constructing a query using the ALLQUAKE table



To create a date value you need to compute the values of the arguments for the DateSerial function, which are integer values of year, month and day of month. The first argument, year, is easy because that is a field in the ALLQUAKE table. The other two arguments need to be extracted from the date code.

Click in the Field row of the first column so that the vertical blinking cursor is positioned there and click on the Builder icon in the toolbar. The Expression Builder window will appear and you should paste – by double-clicking on it -- the DateSerial function from the Date/Time group of Built-In Functions, as shown in Figure 8.11.

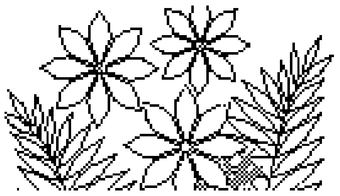
Figure 8.11 – the Expression Builder window



Enter an Expression to define the calculated query field:
(Examples of expressions include [field1] + [field2] and [field1] < 5)

DateSerial(«year», «month», «day»)

DateSerial(year, month, day)
Returns a Variant of type Date for a specified year, month, and day.

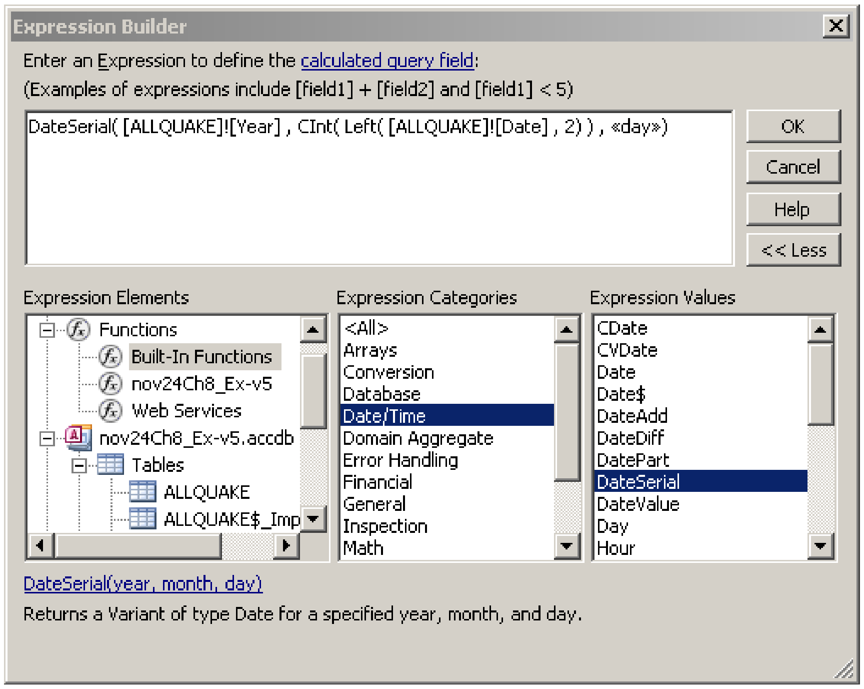


Select the <<year>> argument and paste the Year field from the ALLQUAKE table, by double-clicking on it. You'll have to switch to the Tables selection in the left-hand panel of the Expression Builder window (Figure 8.11). Next select the <<month>> argument, return to the Built-In Functions and paste in the Left function from the Text group of functions. This function will extract the <<length>> leftmost characters from a text value. The text argument, called <<string>>, should be the Date field from the ALLQUAKE table, so paste it in now, and then replace the <<length>> argument with the number 2 (since it is the two leftmost characters that we want).

However, this month argument for the DateSerial function must be an integer, and at the moment the two leftmost characters from the Date field are still text rather than numeric. To convert from text to numeric we need yet another function – this time the CInt function from the Conversion group of functions. You could type it in, making sure to use upper case C and I (upper case I, not L) and also taking care to put parentheses in the right place. Figure 8.12 shows what you should have so far.

You need to do exactly the same thing to replace the <<day>> argument except that you'll use the Right function rather than the Left function. So complete this expression and click the OK button. Close the query design window (saving the query and specifying a name for it, e.g. Calculated would do) and open the query so that you can see the date values produced by this expression.

Figure 8.12 – the partially completed DateSerial expression


Expression Builder

Enter an Expression to define the calculated query field:
(Examples of expressions include [field1] + [field2] and [field1] < 5)

```
DateSerial( [ALLQUAKE]![Year] , CInt( Left( [ALLQUAKE]![Date] , 2) ) , «day»)
```

DateSerial(year, month, day)
Returns a Variant of type Date for a specified year, month, and day.

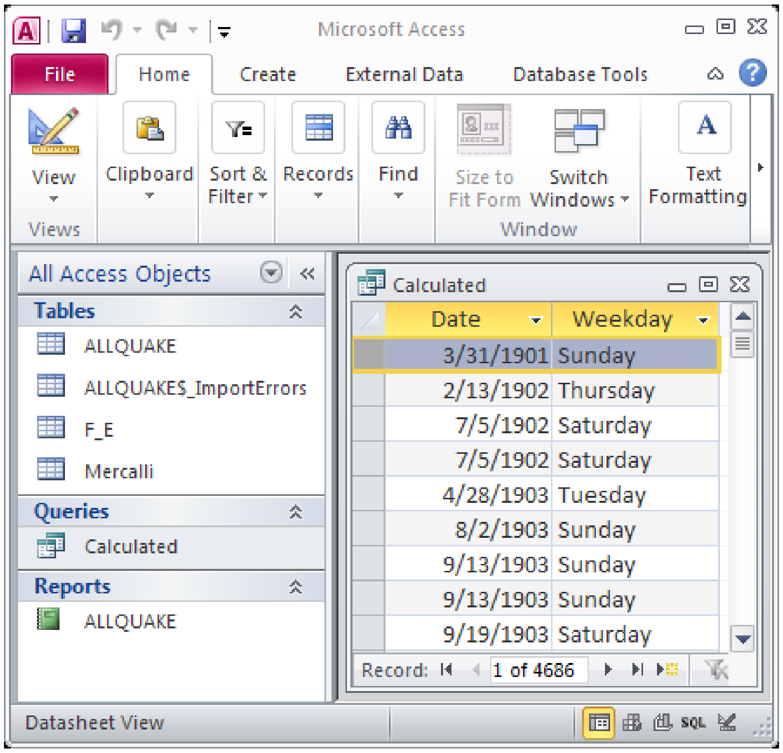
The name for the calculated field is by default Expr1. You can change this by re-entering design mode and replacing Expr1 with a more suitable name such as Date. It is all right to use this name even though a field named Date already exists in the ALLQUAKE table. That is because the query is separate from the table and there is thus no confusion over what the name refers to.

A calculated field containing the day of the week can be obtained using the function Weekday. This function takes a DateSerial value and returns a number between 1 and 7 (1 is for Sunday, 2 for Monday etc to 7 for Saturday). To have the words Sunday, Monday, etc. appear you'll have to use the WeekdayName function. The expression will look like this:

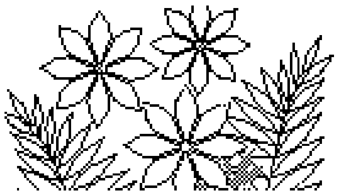
```
WeekdayName(Weekday([Date]))
```

Name the new expression Weekday rather than leaving the default name as Expr2. The first few records of the query as seen in the data sheet view, should be similar to those shown in Figure 8.13.

Figure 8.13 – the data sheet view of the query



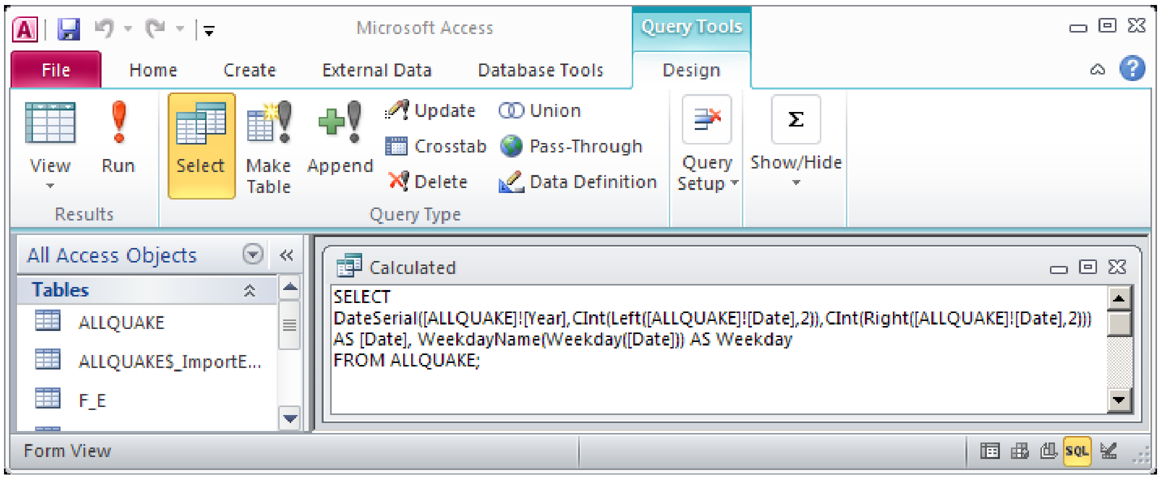
Date	Weekday
3/31/1901	Sunday
2/13/1902	Thursday
7/5/1902	Saturday
7/5/1902	Saturday
4/28/1903	Tuesday
8/2/1903	Sunday
9/13/1903	Sunday
9/13/1903	Sunday
9/19/1903	Saturday



SQL: Structured Query Language

Take a moment to also look at this query using the SQL View option from the Views group in the Home tab. You can see this in Figure 8.14. Observe the syntax carefully.

Figure 8.14 – the SQL view of the query



```
SELECT
DateSerial([ALLQUAKE]![Year],CInt(Left([ALLQUAKE]![Date],2)),CInt(Right([ALLQUAKE]![Date],2)))
AS [Date], WeekdayName(Weekday([Date])) AS Weekday
FROM ALLQUAKE;
```

You can see that this is an SQL select statement. The expressions that you built are also part of the SQL statement. The structure of the statement is:

```
SELECT   [expression] AS [exprName1],
         [expression] AS [exprName2] FROM tableName
```

where the [expression]s are the formulas you just built, exprName1 and exprName2 are the names you just gave to the new calculated fields, and tableName is the ALLQUAKE table.

If you knew the SQL syntax well you could have just typed this statement. You'll not be asked to type an SQL statement in this lab but you are expected to recognise and understand the basics of SQL. The text for the course also discusses basic SQL syntax. As we create additional queries you'll be reminded to examine the SQL view.

Some Queries and Reports

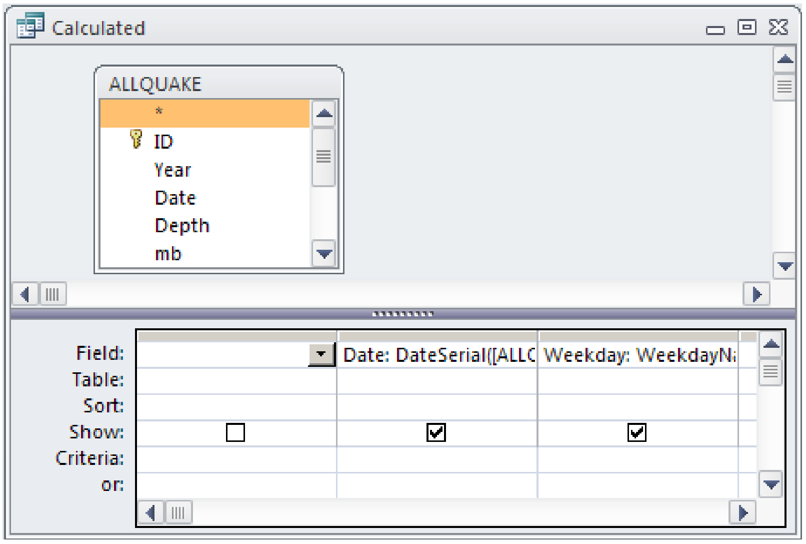
Large Earthquakes

You might want to create a report listing the large earthquakes – meaning those with Richter scale magnitude 7.5 or greater – showing the day, date, region, magnitude, and Mercalli scale effect. To do this you will first have to create a query that obtains just this data from the tables and queries that already exist in the database and then create a report from the query.

This new query is going to require that the ALLQUAKE table and the existing query (that contains the calculated Weekday and Date) be joined. The records will have to be joined one for one. However, there is currently no field in the existing query that can be used to define the join relationship. The ALLQUAKE table has a primary key called ID that could be used if there was a similar field in the query. So your first step is to add a field to the query which simply numbers the records so that the records can be matched one for one with the records in the ALLQUAKE table.

Open the existing query, enter the design view and select Insert Column from the Query Setup group. You should have a window similar to Figure 8.15. Add the ID field from the ALLQUAKE table to this query. In general when you first create a query and table you should always make sure it contains a key field. It was a mistake not to have done so with this particular query. We can now go ahead with creating the new query.

<u>Figure 8.15 – adding a key field to the existing query</u>



Field:		Date: DateSerial([ALLC	Weekday: WeekdayN
Table:			
Sort:			
Show:	☐	☑	☑
Criteria:			
or:			

Click on Query Design in the Queries group inside the Create tab. Select the ALLQUAKE, Mercalli, and F_E tables and also the query containing the calculated values for day and date. The ALLQUAKE, Mercalli and F_E tables should already have relationships defined amongst them. The query will probably also have a join relationship displayed that links the ID field from ALLQUAKE to the ID field in the query (Figure 8.16). If it does not, simply drag the ID field from the ALLQUAKE table to the ID field in the query and release the mouse button.

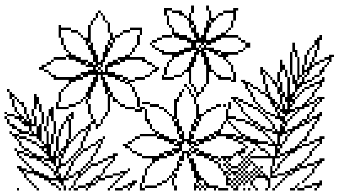
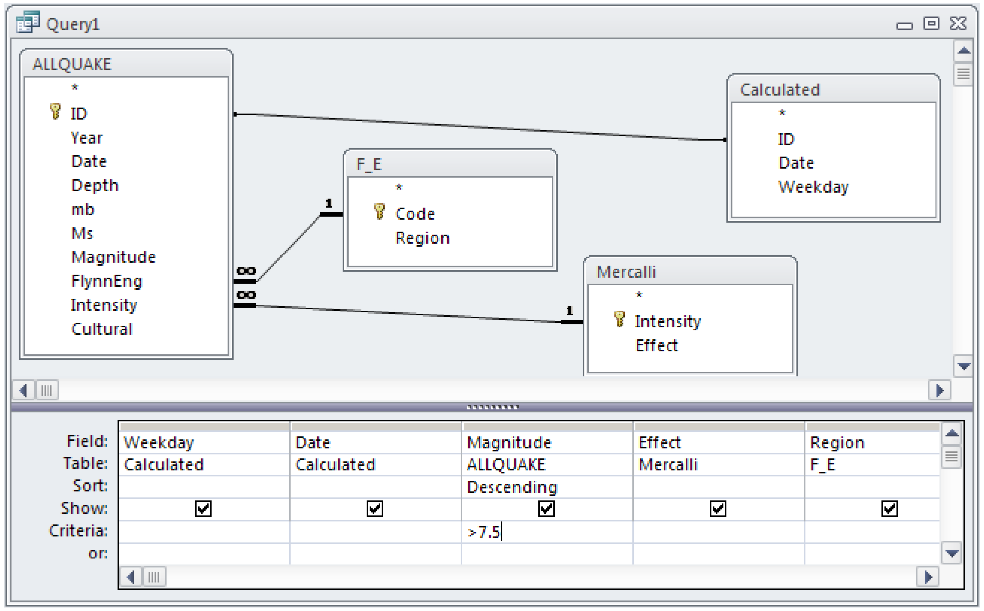


Figure 8.16 – the large earthquakes query design view



Field:	Weekday	Date	Magnitude	Effect	Region
Table:	Calculated	Calculated	ALLQUAKE	Mercalli	F_E
Sort:			Descending		
Show:	✓	✓	✓	✓	✓
Criteria:			>7.5		
or:					

Examine the join properties for this ALLQUAKE/query relationship by double clicking on the line that joins the two objects). You should find that the join property option 1 - the radio button labeled Only include rows where the joined fields from both tables are equal - should be selected.

Now select the fields Weekday and Date from the query, Magnitude from the ALLQUAKE table, Effect from the Mercalli table and Region from the F_E table. Also include the FlynnEng field from the ALLQUAKE table but uncheck the Show box.

Since we're interested in large earthquakes let's sort the records by descending magnitude (i.e. the largest magnitude values will appear first) and also specify the criteria that the magnitude field be greater than 7.5. You can see this in Figure 8.16.

Return to the datasheet view to see the actual data selected. You should have 27 records in the query. Close the query window and save the query, changing the name to LargeQuakes perhaps.

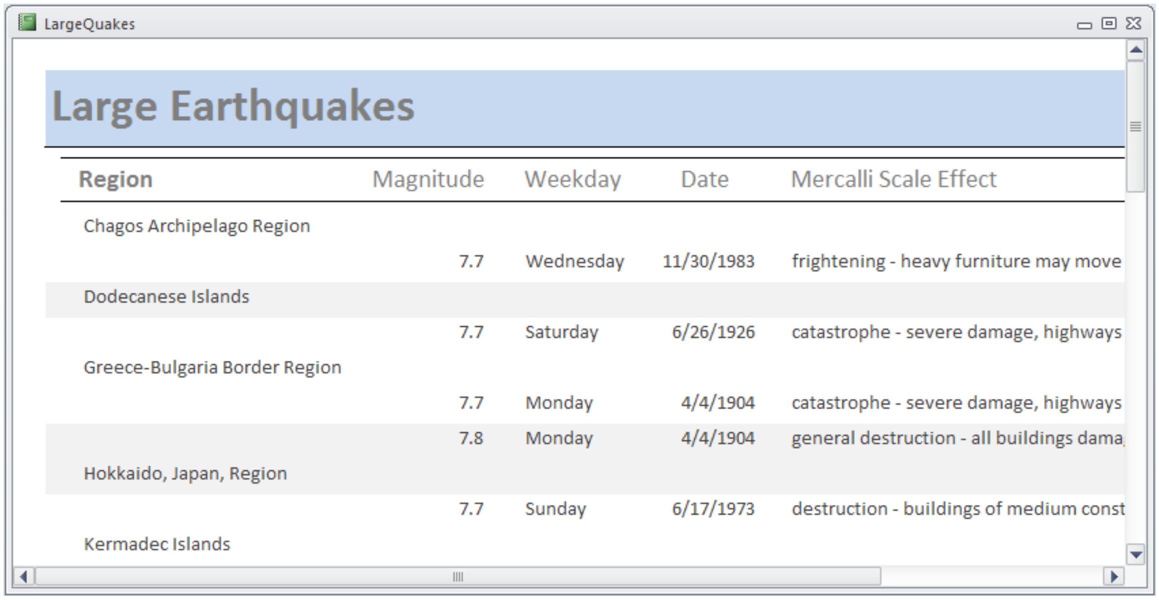
The SQL statement that creates this query can be seen by selecting SQL view from the View menu. There are four parts to this SQL statement. There is the SELECT clause which specifies which fields to include; a FROM clause which specifies where to obtain the data from and includes the joining of tables and query; a WHERE clause which defines the criteria magnitude > 7.5; and an ORDER BY clause which specifies the sorting. (The SQL statement is reproduced on the next page.)

```
SELECT Calculated.Weekday, Calculated.Date, ALLQUAKE.Magnitude,
Mercalli.Effect, F_E.Region
FROM (Mercalli INNER JOIN (F_E INNER JOIN ALLQUAKE ON F_E.Code
= ALLQUAKE.FlynnEng) ON Mercalli.Intensity = ALLQUAKE.Intensity)
INNER JOIN Calculated ON ALLQUAKE.ID = Calculated.ID
WHERE (((ALLQUAKE.Magnitude)>7.5))
ORDER BY ALLQUAKE.Magnitude DESC;
```

The Large Earthquake Report

Figure 8.17 shows part of a report derived from this large earthquake query that you should create.

Figure 8.17 – the Large Earthquake report



Large Earthquakes

Region	Magnitude	Weekday	Date	Mercalli Scale Effect
Chagos Archipelago Region				
	7.7	Wednesday	11/30/1983	frightening - heavy furniture may move
Dodecanese Islands				
	7.7	Saturday	6/26/1926	catastrophe - severe damage, highways
Greece-Bulgaria Border Region				
	7.7	Monday	4/4/1904	catastrophe - severe damage, highways
	7.8	Monday	4/4/1904	general destruction - all buildings dama
Hokkaido, Japan, Region				
	7.7	Sunday	6/17/1973	destruction - buildings of medium const
Kermadec Islands				

Earthquake Magnitude Distribution over Depths

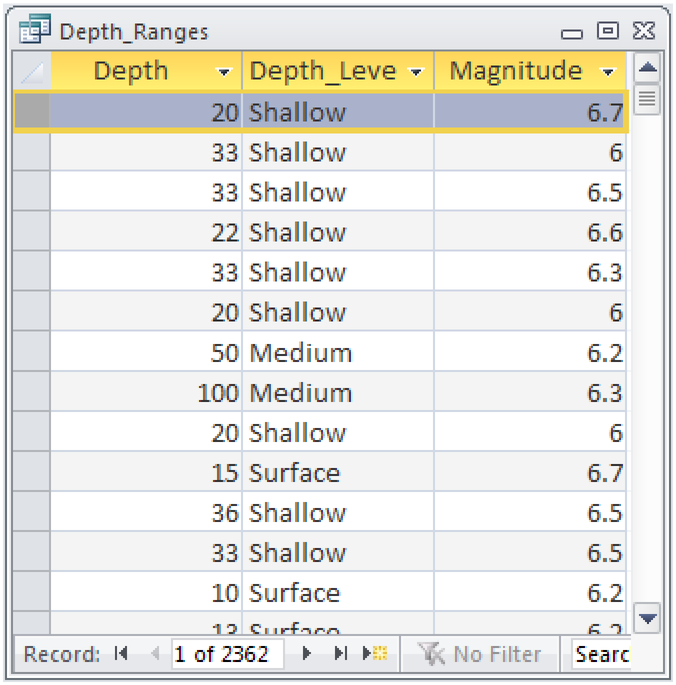
You are also interested in whether large magnitude earthquakes occur at predominantly shallow depths (near the surface of the earth) or if they occur much deeper in the earth's crust. More generally, you would like to see how many earthquakes in each magnitude range (say 8.0 to 7.0; 7.0 to 6.0; 6.0 to 5.0, etc.) occur at depth ranges of surface, shallow, medium or deep. We will define **surface** to be at depth 0 to 15km; **shallow** to be 15 to 40km; **medium** to be 40 to 100km; and **deep** to be greater than 100km.

This is a new task, so start a new Query in Design view. We'll need some data from the ALLQUAKE table. Select that table in the list, click the Add button in the Show Table dialog box, and close it. Click on the Depth field and drag it to the Field: pane. You must be careful to make sure that records in the ALLQUAKE table that have no value for the Depth field are not included. Set the Criteria: for the field to Is Not Null.

The depth values surface, shallow, medium, and deep can be calculated using a sequence of nested IIF functions. Build an expression in the query to assign one of these ranges to each record of the query.

Now add the Magnitude field from ALLQUAKE to the query. You should also set the Criteria: for this field to avoid importing records that have no value for Magnitude. Save the query as Depth_Ranges. The first few records are shown in Figure 8.18.

Figure 8.18 – the query containing just the magnitude and depth code fields


Depth_Ranges

Depth	Depth_Leve	Magnitude
20	Shallow	6.7
33	Shallow	6
33	Shallow	6.5
22	Shallow	6.6
33	Shallow	6.3
20	Shallow	6
50	Medium	6.2
100	Medium	6.3
20	Shallow	6
15	Surface	6.7
36	Shallow	6.5
33	Shallow	6.5
10	Surface	6.2

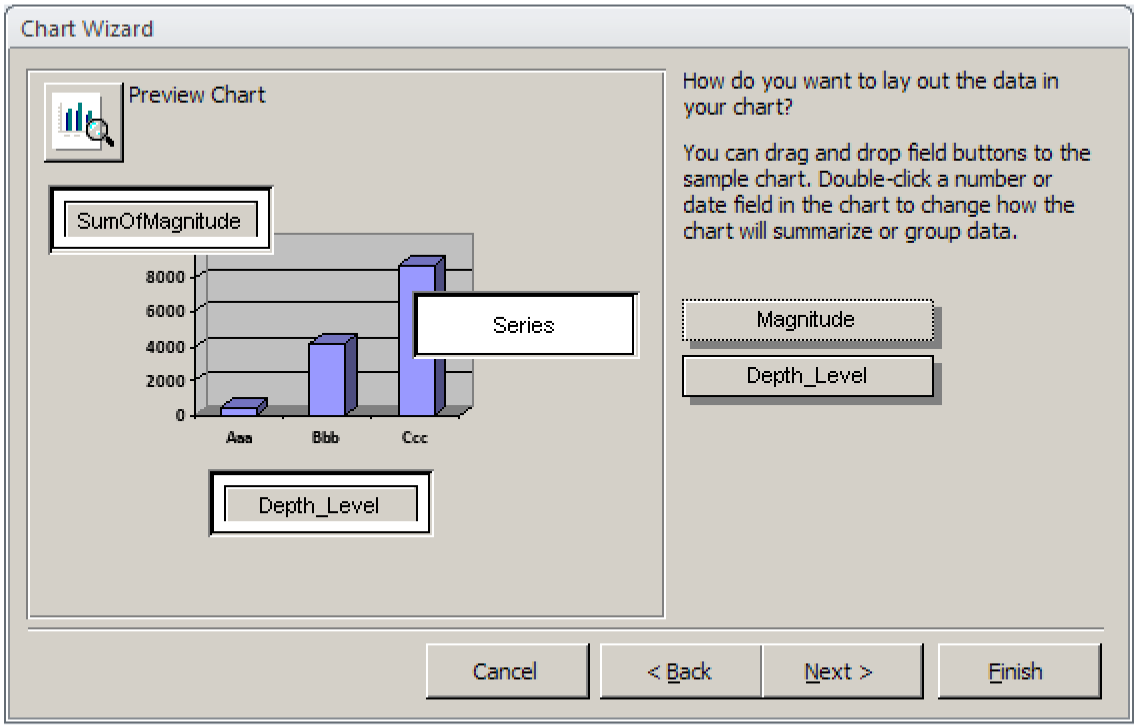
Record: 1 of 2362 No Filter Searc

Next, start to create a new report and choose Chart Wizard as the method of creating the report. Select Report Design in the Reports group inside the Create tab. Next, click on the Chart icon in the Controls group and drag it to Detail. This activates the Chart Wizard. In the first panel of the chart wizard choose the Depth_Ranges query as the source of data for the chart. Make sure that only the fields Magnitude and Depth_Levels, in that order, are used as the fields for the chart and choose the 3-D column chart as the type.

You'll be faced next with the window shown in Figure 8.19. Observe that the horizontal axis of the chart is labeled Depth_Levels by default, and this is probably what you want. You're aiming to create a chart that shows, for each depth category, the distribution of earthquakes by magnitude (i.e. how many earthquakes of each magnitude occurred at each depth).

The empty area labeled Series should be used for the Magnitude field – so drag the Magnitude tile on the right into that area. You'll see the graph representation change into a 3-D format.

<u>Figure 8.19 – specifying the chart layout</u>



How do you want to lay out the data in your chart?

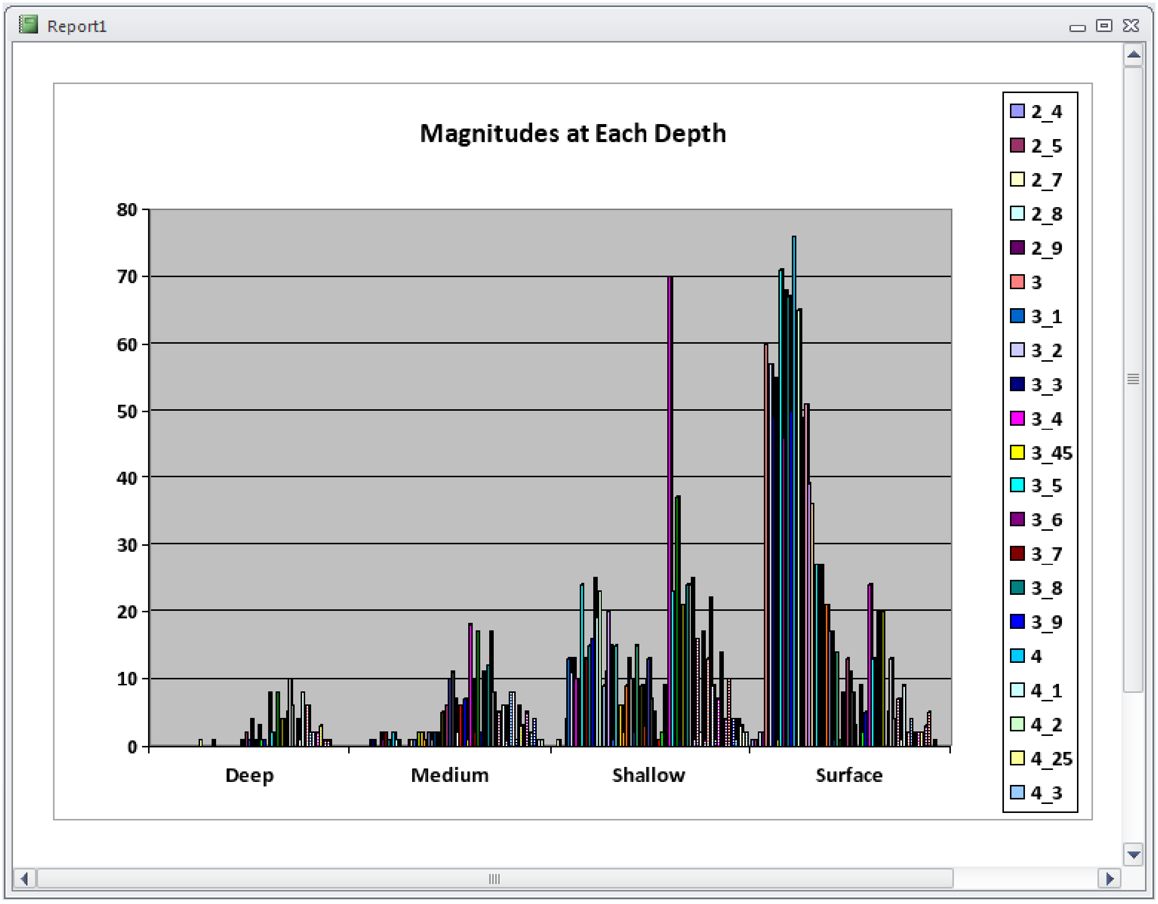
You can drag and drop field buttons to the sample chart. Double-click a number or date field in the chart to change how the chart will summarize or group data.

You don't want a sum of the magnitude values, however, so double click that tile and select Count from the list that appears.

Finish the chart wizard by clicking on the Finish button. It will be quite small and clearly you must change its size to approximately fill a page. To do this switch to, or stay in, the Design View and drag the edges of the page to about 16cm horizontally and the expand the Details section to about 20cm (vertically). Next, select the chart itself and drag the handles so that the chart region approximately fills the page. Notice that the axis labels and the chart itself are not representing the actual data from the database. They are simply "dummy" values that give you a rough idea of what your graph will look like.

After you have expanded the window containing the graph to an appropriate size close the windows, saving the report under an appropriate name if asked, and then switch to Layout View or, if necessary, reopen the report to view the graph. You should have something resembling Figure 8.20.

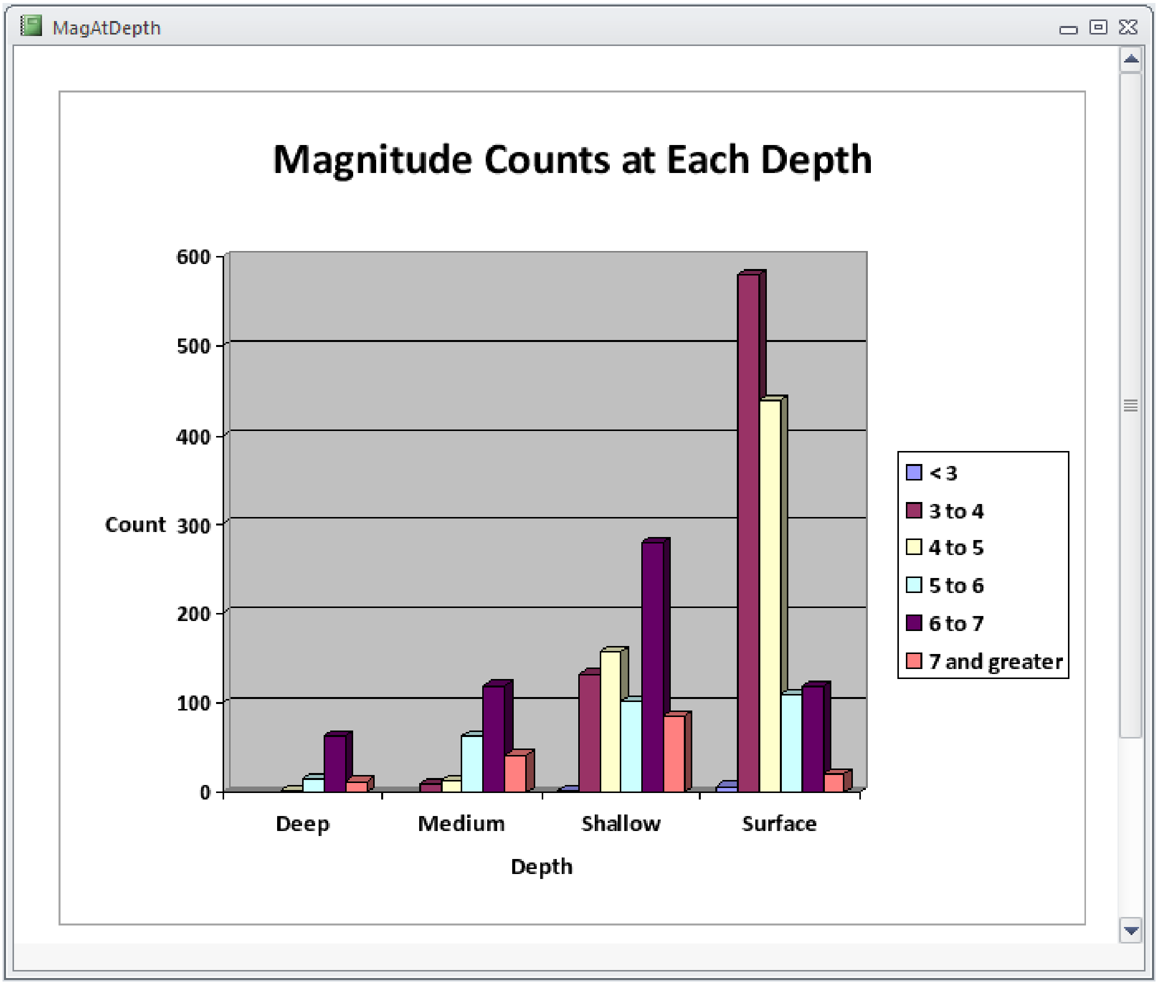
Figure 8.20 – the graph of counts of each magnitude at each depth range



This is not particularly useful because of the large number of depth values. There are so many, in fact, that the legend on the right is too large to be completely displayed. It would be more useful if you created ranges for the magnitude values and then graphed counts for those ranges rather than for each and every magnitude value.

Do this yourself. Add a field to the Depth_Ranges query that will assign bins to the values of Magnitude – use the labels displayed in Figure 8.21.

Figure 8.21 – a better chart of magnitudes at each depth range


MagAtDepth
Magnitude Counts at Each Depth
Count
Deep
Medium
Shallow
Surface
Depth
< 3
3 to 4
4 to 5
5 to 6
6 to 7
7 and greater