



# Using Microsoft Access

<b>USING MICROSOFT ACCESS</b>	<b>1</b>
<b>Relational Databases</b>	<b>2</b>
Problems with Un-Normalised Data	4
<b>Normalisation</b>	<b>5</b>
First Normal Form (1NF)	5
Second Normal Form (2NF)	6
Third Normal Form (3NF)	7
Exercise 1. Creating a Relational Database	8
Exercise 2. Creating the Tables	9
Exercise 3. Creating Relationships	11
Exercise 4. Entering Records in Related Tables	14
Exercise 5. Entering Information for Lookup Fields	16
Exercise 6. Creating Lookup Lists	17
Exercise 7. Checking Lookup Relationships	21
Exercise 8. Testing the Lookup Fields	22

## Relational Databases

In all of the previous exercises you have worked on a simple database that only had one table. In most cases, a database will be too complex to be able to work well with only one table. Consider the following invoice document.

<b>EGBST INVOICE</b>	<b>NO I393</b>		
<b>24 Invisible Road</b>			
<b>Kalgoorlie 6430</b>			
<p>Joondalup Jewellers          402 Walter Road          Morley WA 6059          93752845</p>			
<p>Date 9<sup>th</sup> April 2004</p>			
QTY	Description	Price	Cost
2 ounces 20	Gold Rings	\$800 \$110	\$1600 \$2200
<b>Total</b>		\$3800	

Suppose a business manually completes an invoice like this every time one of their customers makes a purchase. If the business decides that they want all of this information to be recorded in a database so a neat invoice can be printed, would one table be sufficient to store all of that information?

If we listed all of the fields that might be required to store this information, the fields might include:

- Invoice number
- Customer name
- Customer address
- Customer state
- Customer postcode
- Customer phone number
- Date
- Quantity
- Description
- Price
- Cost
- Total

If one table was used to store this information, it might look something like the example on the following page.

Invoice	Customer Name	Customer Address	Customer state	Customer postcode	Customer phone	Date	Quantity	Description	Price	Cost	Total
I393	Joondalup Jewellers	402 Walter Rd Morley	WA	6059	93752845	9 Apr	2	Gold	\$800	\$1600	\$3800
I393	Joondalup Jewellers	402 Walter Rd Morley	WA	6059	93752845	9 Apr	20	Rings	\$110	\$3800	\$3800
I394	Hourglass Jewellers	230 Light St Morley	WA	6059	93751920	11 Apr	4	Gold	\$800	\$2400	\$2710
I394	Hourglass Jewellers	230 Light St Morley	WA	6059	93751920	11 Apr	5	Watches	\$50	\$250	\$2710
I394	Hourglass Jewellers	230 Light St Morley	WA	6059	93751920	11 Apr	3	Bracelets	\$20	\$60	\$2710
I395	Balcatta Bracelets	45 Russel St Morley	WA	6059	92769385	14 Apr	12	Rings	\$110	\$1320	\$1480
I395	Balcatta Bracelets	45 Russel St Morley	WA	6059	92769385	14 Apr	8	Bracelets	\$20	\$160	\$1480

## Problems with Un-Normalised Data

In this example we can see some problems:

- Every time a sales transaction is being entered for an invoice, the entire invoice details (such as invoice number, date and customer details) are being repeated.
- Every time an invoice is prepared for a customer, all of that customer's details are being repeated.
- Some information, such as totals, may not need to be entered if it can be generated by the database.

This duplication shows evidence of **redundancy** in the design and can lead to several problems:

- The database will take up more room on the computer because the same information is being stored several times. This takes up more disk space and makes the database run slower.
- The more times information is added, the more chance there is of making an error in entry.
- If the same information was entered differently, it may be treated as different information. E.g. *Hourglass Jewellers* and *Hourglass-Jewellers* might not be recognised by the database as the same customer if they were entered differently at different times.

This redundancy can be eliminated by designing the database more efficiently using more than one related table. A database designed using multiple tables that are related to each other is referred to as a **Relational Database**.

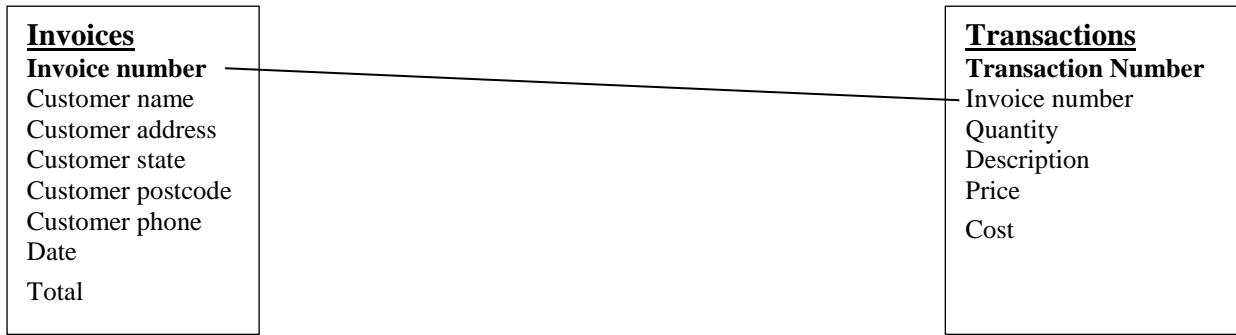
Planning what tables will be needed in a database is usually done using a process known as **Normalisation**. This is a step-by-step process for identifying and eliminating redundancies in a database. The final result is a plan for an efficient database using multiple related tables. This plan is often prepared with the aid of an **Entity Relationship Diagram (ERD)**. These exercises will only provide a brief explanation of Normalisation. If you need more help on normalising databases, there are plenty of resources which cover the topic in detail.

## Normalisation

Normalisation usually involves three main stages (additional stages are used by some database developers). Each of these stages is referred to as a **Normal Form**.

### First Normal Form (1NF)

First normal form involves eliminating repeating groups. In the previous example, every time transactions from an invoice are entered, the details for the invoice would also need to be repeated. This problem can be eliminated by having a table for *Invoice* details and another table for *Transaction* details. The *Transaction* details table could include the *Invoice number* to identify which invoice the transaction belongs to. This would mean that *Invoice number* would be the link, or **Relationship**, between the two tables as illustrated below.



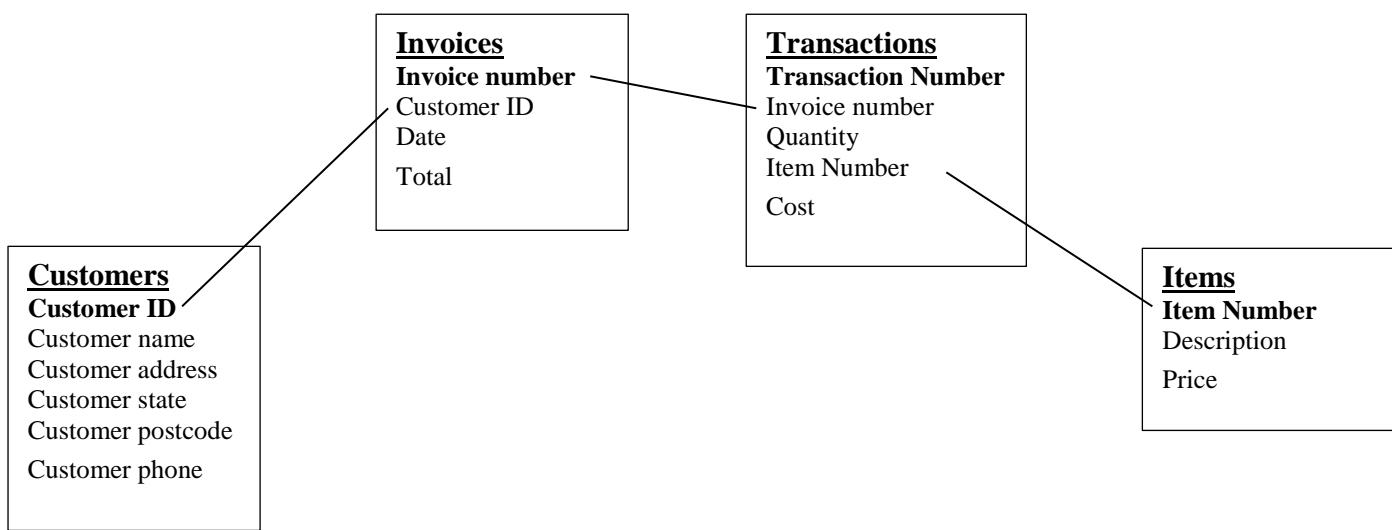
In the above example, the **Primary Key** in each table is indicated using bold formatting. *Transaction number* has been added to the *Transactions* table as a primary key.

Now the invoice details are entered in the invoices table, whenever a transaction is entered in the Transactions table only the Invoice Number will need to be entered. The invoice number will be related to the rest of the invoice details in the *Invoice* Table. This database is now in **First Normal Form** or **1NF**.

## Second Normal Form (2NF)

When a database is in 2NF, all of the fields in each table will depend directly on the primary key. If we look at the *Invoices* table in its current form, we can see that this is not the case. The invoice date and invoice total are dependent on the invoice number, but the rest of the fields are not. Since there may be more than one invoice for the same customer, the fields that provide information about the customer will most likely appear on many invoices. At the moment, they will be repeated for each invoice. This problem can be eliminated by creating a separate table to store the customer details.

The same problem is evident in the *Transactions* table. The quantity and cost is unique to each transaction and therefore dependent on the transaction number. The price and description, however, are not directly dependant on the primary key. Every time there is a transaction for a certain item, the price and description for that item are being repeated. This can be fixed by creating a separate table for the Items that can be sold. The following diagram illustrates the new relational design.

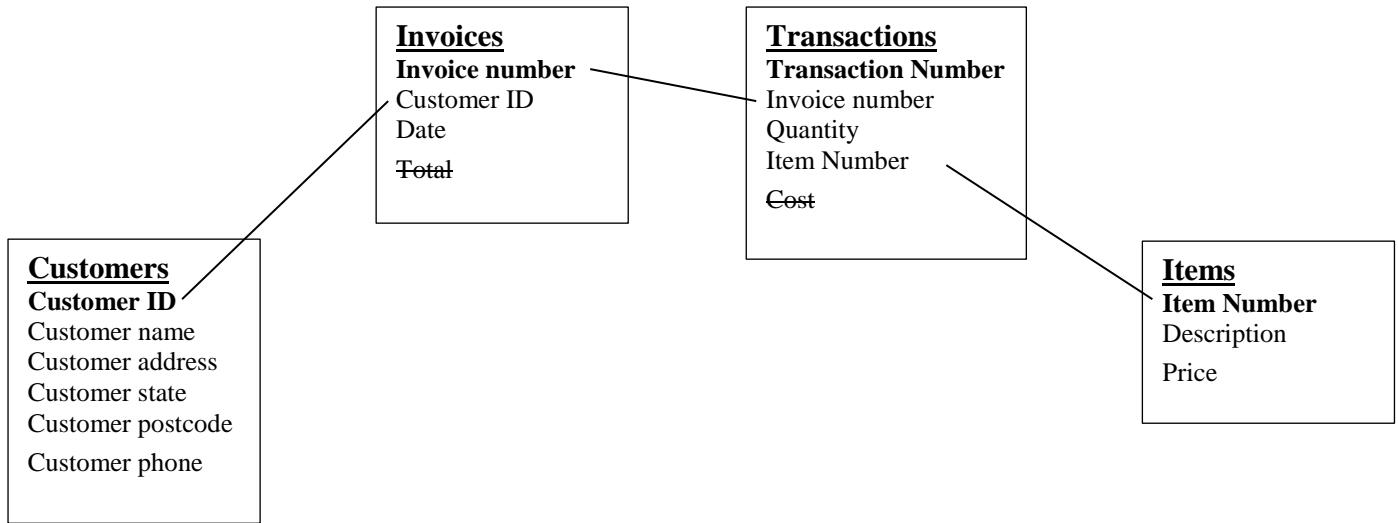


**Primary Keys** have been created for both the *Customers* table and the *Items* table. The lines between each table indicate which fields will be linked. In database terminology these lines are the **Relationships**. There are different types of relationships in databases (one to many, one to one, many to many) but for the purpose of these exercises, it will be enough to know that the tables need to be related to each other via linked fields.

This database design is now in **Second Normal Form** or **2NF**.

### Third Normal Form (3NF)

In 3NF, the database design will not include any redundant fields, such as fields that can be automatically calculated by the database. In the example below, we have eliminated *Cost* from the *Transactions* table since that can be calculated automatically by multiplying *Quantity* sold by the *Price* of the item. We have also eliminated the invoice *Total*, since that can be calculated by adding up the transactions on the invoice.

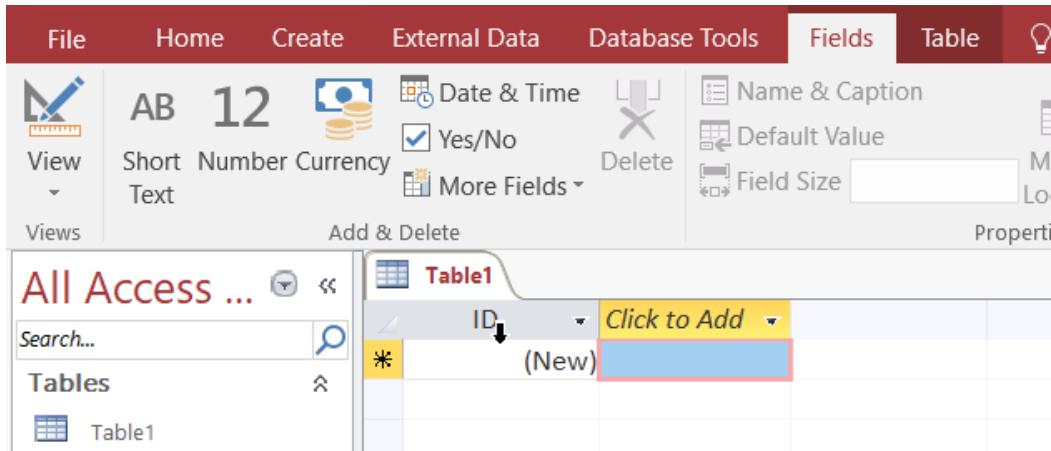


This database design is now in **Third Normal Form** or **3NF**. It can be said to be **Normalised**.

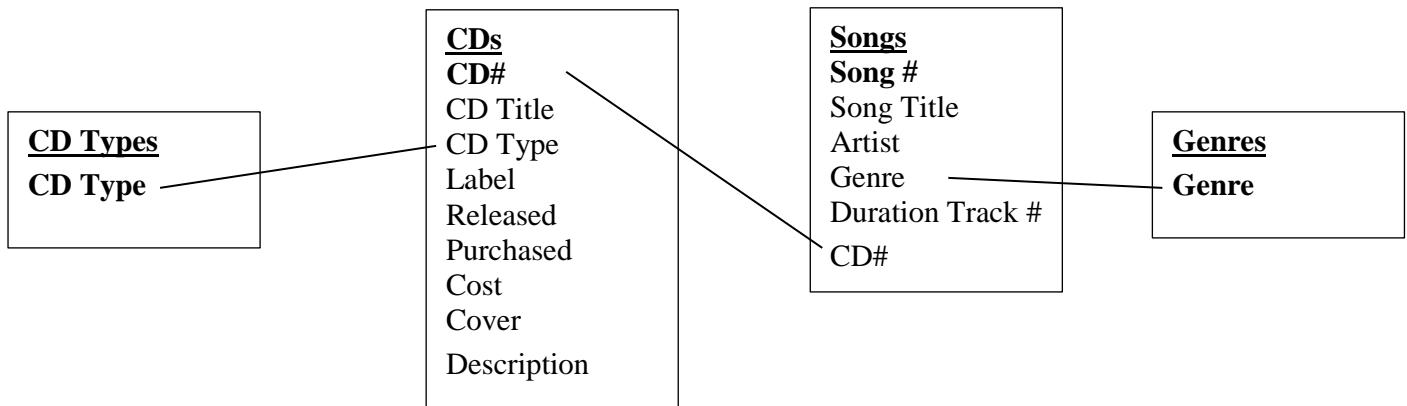
### Exercise 1. Creating a Relational Database

In the remaining exercises we will create a relational database that will be used to keep track of a music CD collection.

1. Open Microsoft Access.
2. Create a new blank Database file with the filename *Music Collection.accdb*.



We will assume that our database has already been planned and normalised, to come up with the following table layout.



The CD Types table and Genres tables have been added to assist in the creation of lookup fields as you will see later.

## Exercise 2. Creating the Tables

1. Create and save each of the four tables for the database. Refer to the previous exercises if you can't remember how this is done. The field names, data types and relevant properties for each table are shown below. Remember to specify the primary key before you save a table. You can add additional field properties if appropriate.

Table name – CD TYPES

Field Name	Data Type	Description	Properties
CD Type	Short Text	Type of CD (Album, Soundtrack etc)	Primary Key

Table name – CDS

Field Name	Data Type	Description	Properties
CD #	AutoNumber	Identification number for the CD	Primary Key
CD Title	Short Text	Title of the CD	
CD Type	Short Text	Type of CD (Album, Soundtrack etc)	Default Value – Album
Label	Short Text	Label the CD was released by	
Released	Short Text	Year the CD was released	Field Size – 4
Purchased	Date/Time	Date the CD was purchased	Format – dd/mm/yy Input Mask – 99/00/00 Default Value – =Date() Validation Rule – <=Date() Validation Text
Cost	Currency	Amount paid to buy the CD	
Cover	OLE Object	Picture of the CD cover	
Description	Long Text	Description of the CD	

Table name – SONGS

Field Name	Data Type	Description	Properties
Song #	AutoNumber	Identification number for the CD	Primary Key
Song Title	Short Text	Title of the song	
Artist	Short Text	Artist the song is performed by	
Genre	Short Text	Type of song (rock, rap, techno etc.)	
Duration	Short Text	Length of the song (minutes:seconds)	Field Size – 5 Input Mask – 99\::00;0;_
Track #	Short Text	Position of the song on the CD	Field Size – 2
CD #	Number	Number of the CD that the song is on	

Table Name - GENRES

Field Name	Data Type	Description	Properties
Genre	Short Text	Type of song (rock, rap, techno etc.)	Primary Key

The table designs should look similar to the ones shown below.

**CD TYPES**

Field Name	Data Type	
CD Type	Short Text	Type of CD (Album, Soundtrack etc)

**CDS**

Field Name	Data Type	
CD #	AutoNumber	Identification number for the CD
CD Title	Short Text	Title of the CD
CD Type	Short Text	Type of CD (Album, Soundtrack etc)
Label	Short Text	Label the CD was released by
Released	Short Text	Year the CD was released
Purchased	Date/Time	Date the CD was purchased
Cost	Currency	Amount paid to buy the CD
Cover	OLE Object	Picture of the CD cover
Description	Long Text	Description of the CD

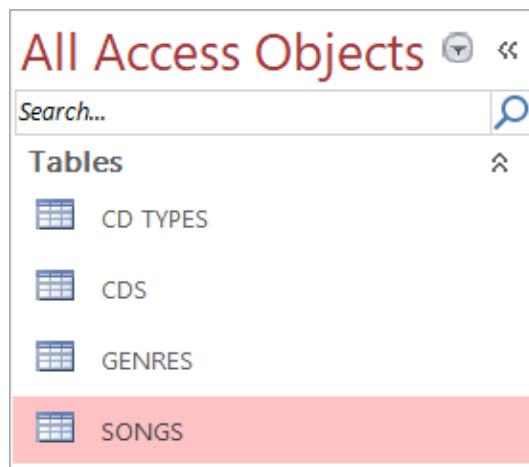
**GENRES**

Field Name	Data Type	
Genre	Short Text	Type of song (rock, rap, techno etc.)

**SONGS**

Field Name	Data Type	
Song #	AutoNumber	Identification number for the CD
Song Title	Short Text	Title of the song
Artist	Short Text	Artist the song is performed by
Genre	Short Text	Type of song (rock, rap, techno etc.)
Duration	Short Text	Length of the song (minutes:seconds)
Track #	Short Text	Position of the song on the CD
CD #	Number	Number of the CD that the song is on

The **Navigation Pane** should show all four tables.



### Exercise 3. Creating Relationships

Relationships between tables can be created and managed manually using the Relationships window. Relationships can also be created automatically in some instances, such as when the lookup wizard is used. In this exercise we will manually create a relationship between the *CDS* table and the *SONGS* table.

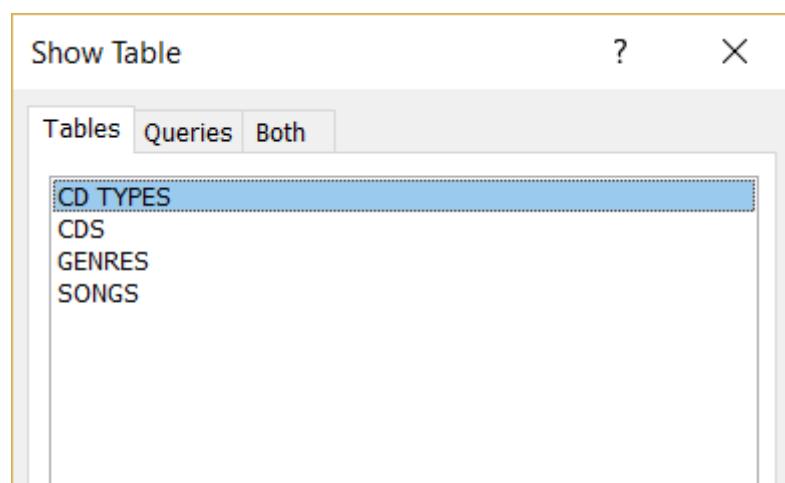
1. Click the **Database Tools** tab on the Ribbon.



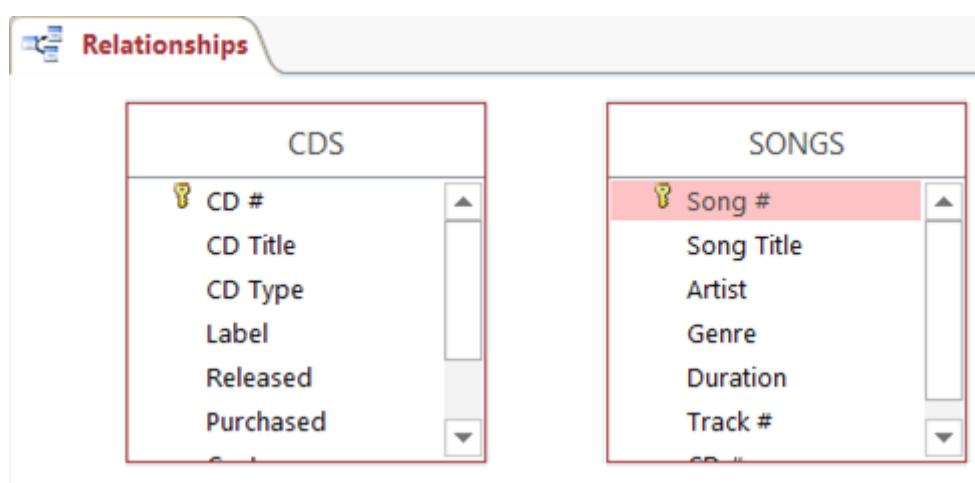
2. Click on **Relationships**.



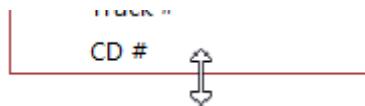
A **Show Table** dialog similar to the one for designing queries will appear.



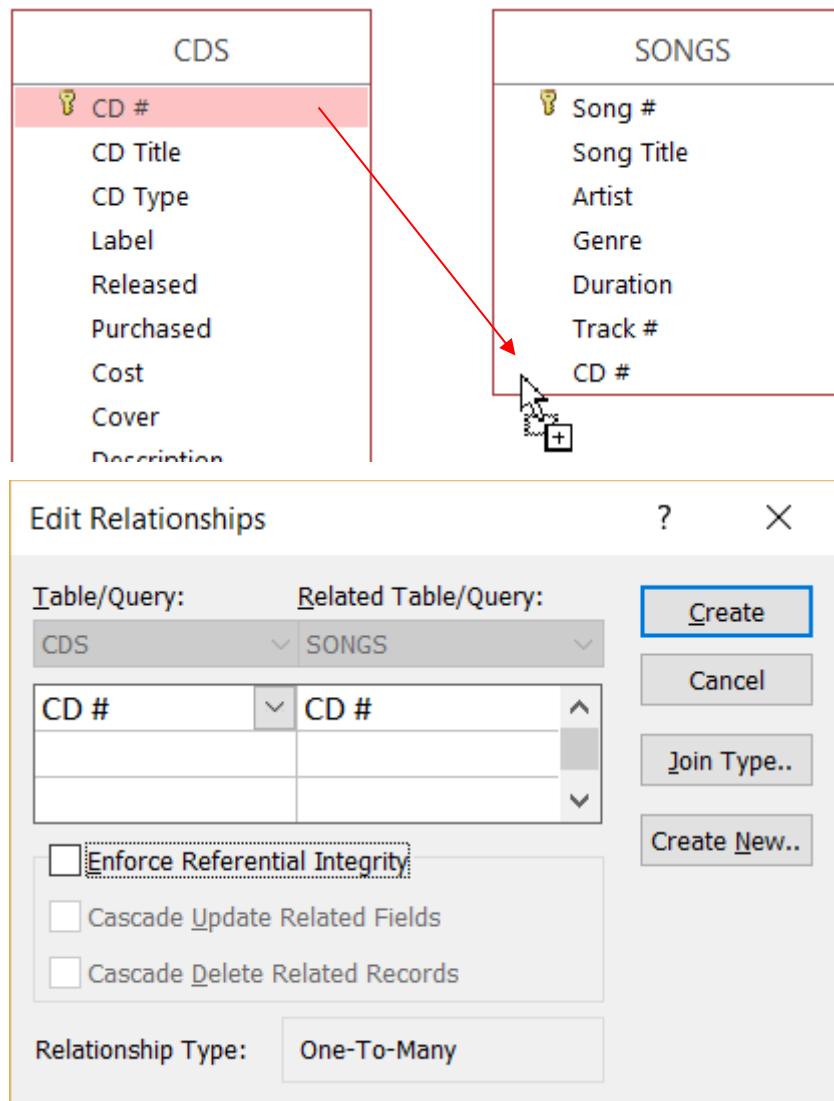
3. **Double-click** on the *CDS* table and **Double-click** on the *SONGS* table to add them both to the relationships window. Close the Show Table window when they are both added to the **Relationships** window.



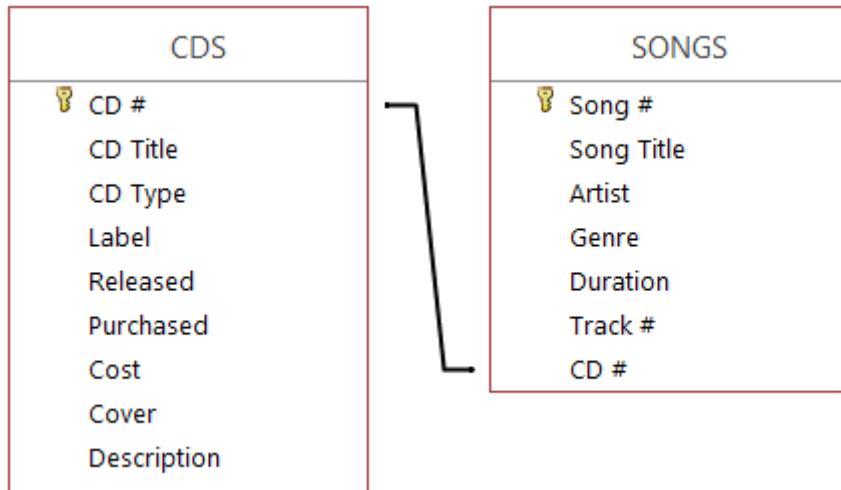
4. Move your mouse to the bottom edge of the *SONGS* table until it changes to a re-sizing arrow as shown above. Drag downwards until you can see all of the fields in the table listed. Do the same with the *CDS* table.



5. Click on *CD#* in the *CDS* table and drag it on to *CD#* in the *SONGS* table. The **Edit Relationships** dialog will appear.



6. Make sure *CD#* is selected in both tables as shown above.  
 7. Click the **Create** button to create the relationship.



A line will appear to indicate the relationship. Your database now knows that each song is related to the CD the song is from. The CD# is used as the common field that links the two tables.

8. Close the relationships window. When you are prompted to save the changes, click **Yes**.

Later on you will see several ways that this relationship can be used in the database.

**Note** Fields that are related to each other need to be a similar data type, otherwise there may be problems. For example, linking an auto number to a number is fine, but either one of those linked to a text field could cause problems later on.

#### Exercise 4. Entering Records in Related Tables

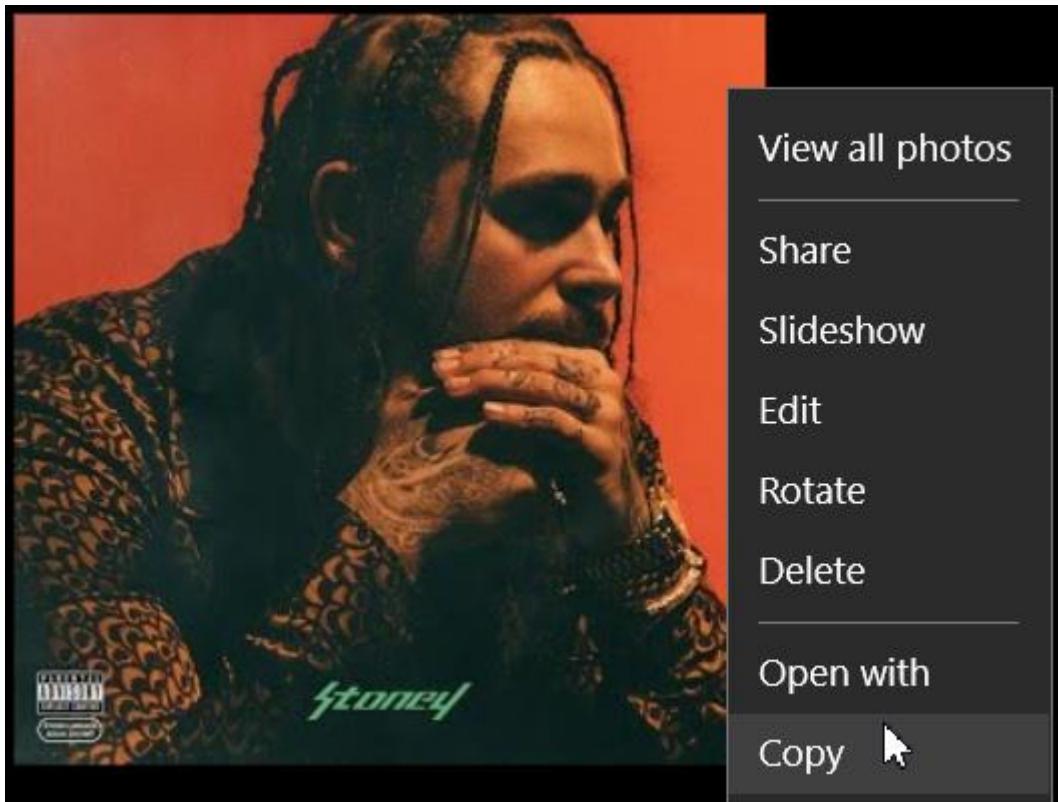
1. Double-click the *CDS* table to open it.
2. Enter the following record in the *CDS* table (The *CD#* will be filled in by the AutoNumber).

CD #	CD Title	CD Type	Label	Released	Purchased	Cost	Cover	Description
Stoney	Album	UMG Recordings	2016	2016	13/04/17	\$19.99		

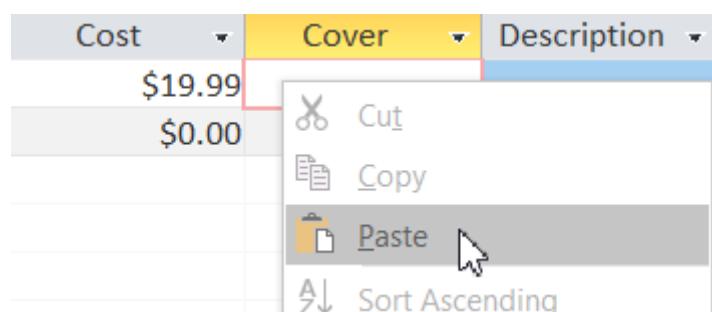
To add an image in the cover art field you will first need to make sure you have a copy of all the Access Exercise files. If you don't already, you can get them from

<http://oneil.com.au/pc/access.html>

3. Go to where you have saved the image file **Post Malone - Stoney.jpg** and open it in an image viewing application (such as the standard Windows photo viewer). Right click the open image and copy it.



4. Now go back to Microsoft Access with the CDs table still open.
5. Right-click inside the **Cover** field and select **Paste**.



While you are in datasheet view the word Picture will appear to show you that a picture is stored in the Cover field for that record. When you are viewing the data in a form or a table as we will do later, the picture itself can be displayed.

When you finish entering the record, a + sign will appear to the left of the record. This occurs when there is a related table. In this case, the relationship may be used to enter records for songs that are on the album.

	CD #	CD Title	CD Type	Label	Released	Purchased	Cost	Cover	De
+	1 Stoney	Album	UMG Recordings	2016		13/04/17	\$19.99	Picture	
*	(New)	Album				10/02/19	\$0.00		

- Click on the + sign to display a blank record from the related *SONGS* table.

	CD #	CD Title	CD Type	Label	Released	Purchased	Cost	Cover	De
+	1 Stoney	Album	UMG Recordings	2016		13/04/17	\$19.99	Picture	
*	Song #	Song Title	Artist	Genre	Duration	Track #	Click to Add		
*	(New)								

- Use that space to add the song records shown below. In fields like the *Artist* and *Genre* field where the information is the same for each song, remember that you can use the **Ctrl** ' (single quotation mark) shortcut to repeat information from the previous record.

SONGS						
Song #	Song Title		Artist	Genre	Duration	Track #
1	Broken Whiskey Glass		Post Malone	Hip-Hop	03:53	1
2	Big Lie		Post Malone	Hip-Hop	03:27	2
3	Deja Vu (feat. Justin Bieber)		Post Malone	Hip-Hop	03:54	3
4	No Option		Post Malone	Hip-Hop	02:59	4
5	Cold		Post Malone	Hip-Hop	04:28	5
6	White Iverson		Post Malone	Hip-Hop	04:16	6
7	I Fall Apart		Post Malone	Hip-Hop	03:43	7
8	Patient		Post Malone	Hip-Hop	03:14	8
9	Go Flex		Post Malone	Hip-Hop	02:59	9
10	Feel (feat. Kehlani)		Post Malone	Hip-Hop	03:17	10
11	Too Young		Post Malone	Hip-Hop	03:57	11
12	Congratulations (feat. Quavo)		Post Malone	Hip-Hop	03:40	12
13	Up There		Post Malone	Hip-Hop	03:14	13
14	Yours Truly, Austin Post		Post Malone	Hip-Hop	03:39	14

- Close the table when complete. If you have made any changes to the table design (such as adjusting column widths to make the information fit) you may be prompted to save the changes.

**Exercise 5. Entering Information for Lookup Fields**

The *CD TYPES* table and the *GENRES* table will both be used for lookup fields soon, so we will enter some data in to those fields.

1. Open the *CD TYPES* field.
2. Enter the following types.

- Album
- Soundtrack
- Compilation
- CD Single
- Other

CD TYPES	
	CD Type
	Album
	Soundtrack
	Compilation
	CD Single
	Other

3. Click the **Sort Ascending** icon to sort them in alphabetical order. 
4. Close the table and save the changes when prompted.
5. Open the *GENRES* table and enter the following genres.

- Hip-hop
- Rock
- Pop
- Kids
- Dance
- Funk
- Rock
- Club

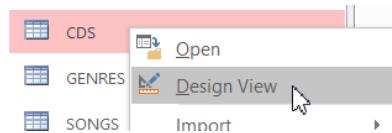
GENRES	
	Genre
	Club
	Dance
	Funk
	Hip-hop
	Kids
	Pop
	Rock

6. Sort the records and close the table, saving changes when prompted.

## Exercise 6. Creating Lookup Lists

A lookup field is a field with a combo box (sometimes referred to as a drop-down list). This allows the user of the database to select information from a list rather than having to type information in a field. The *CD TYPES* table and the *GENRES* table used in the previous exercise will now be used to create lookup fields for the other two tables.

1. Open the *CDS* table in design view.

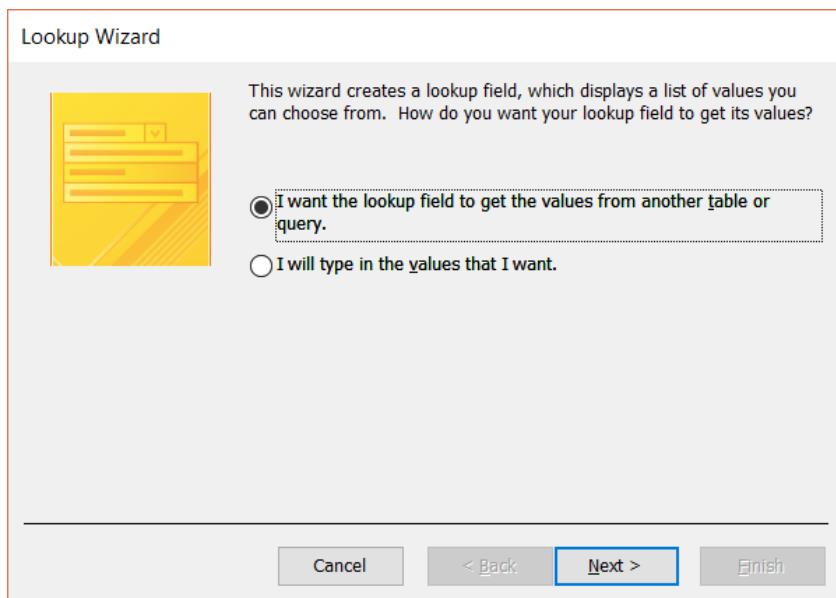


CD Title	Short Text
CD Type	Short Text
Label	Short Text
Released	Long Text
Purchased	Number
Cost	Date/Time
Cover	Currency
Description	AutoNumber
	Yes/No
	OLE Object
	Hyperlink
	Attachment
	Calculated
	Lookup Wizard...

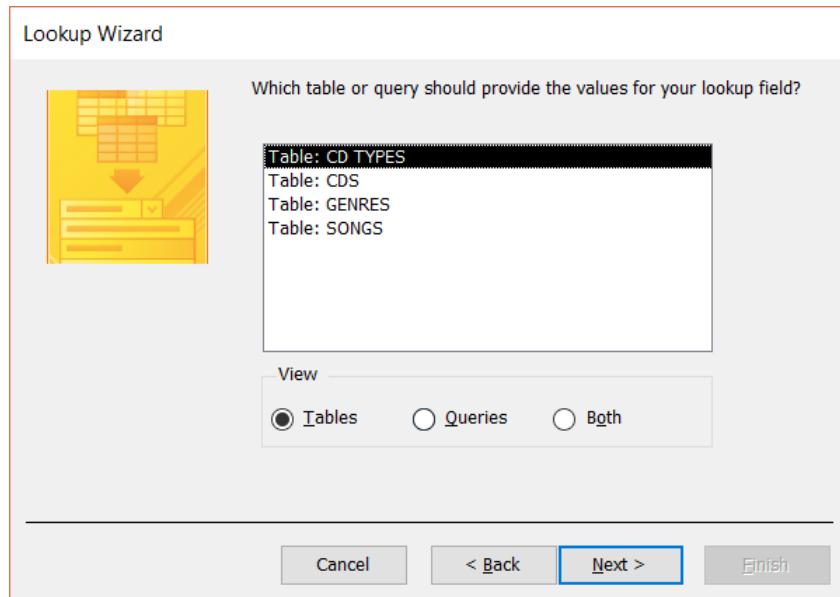
2. Select the *CD Type* field.
3. From the list of field data types, choose **Lookup Wizard** as shown to the right. The Lookup Wizard will start.

**Tip** Whenever a menu option is followed by dots... that usually indicates that selecting that option will open another window like a wizard or options window.

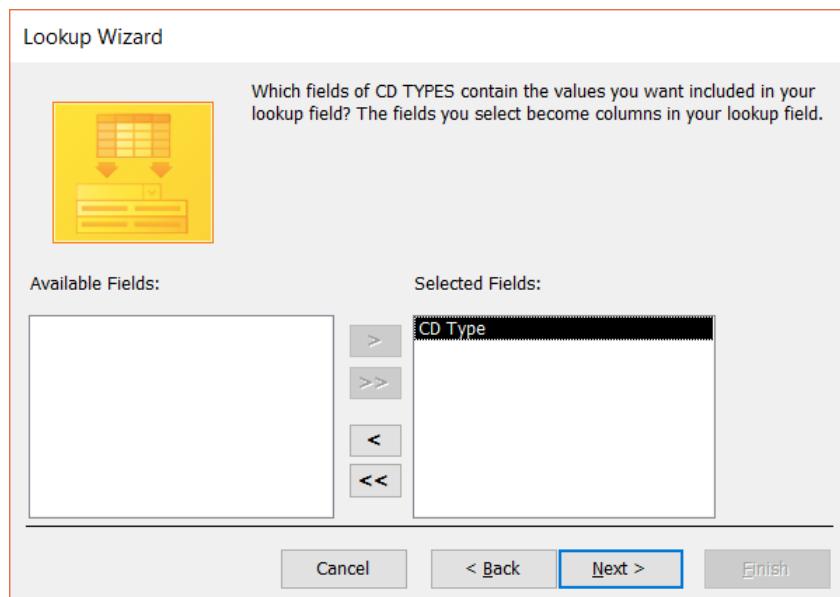
We want the options in the drop-down list to come from the *CD TYPES* table we have created.



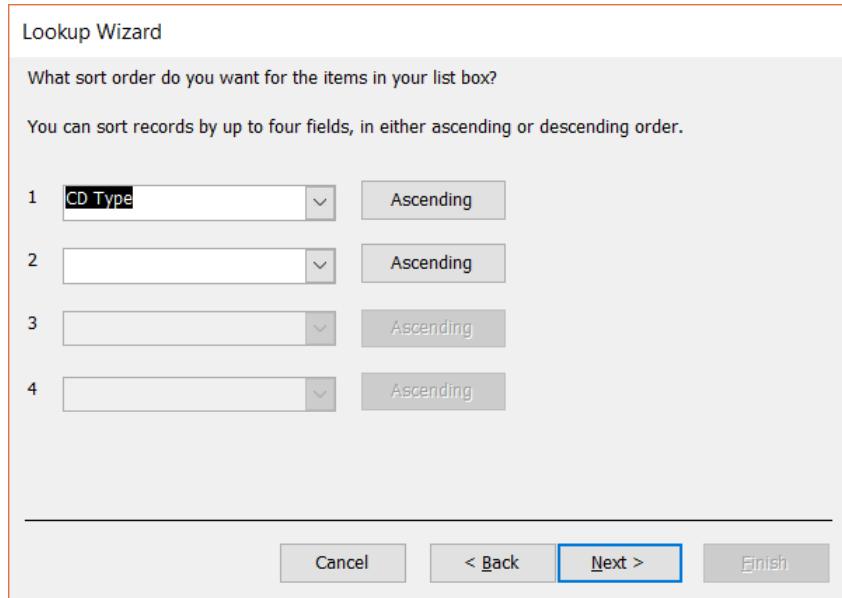
4. In the first step of the wizard, leave the first option selected and click **Next**.



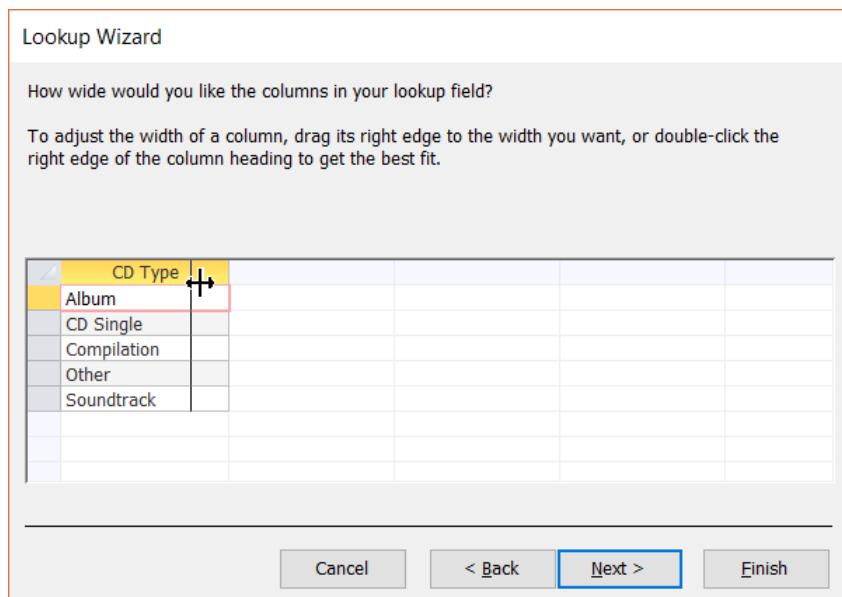
5. In the next step, make sure *Table: CD TYPES* is selected and click **Next**.



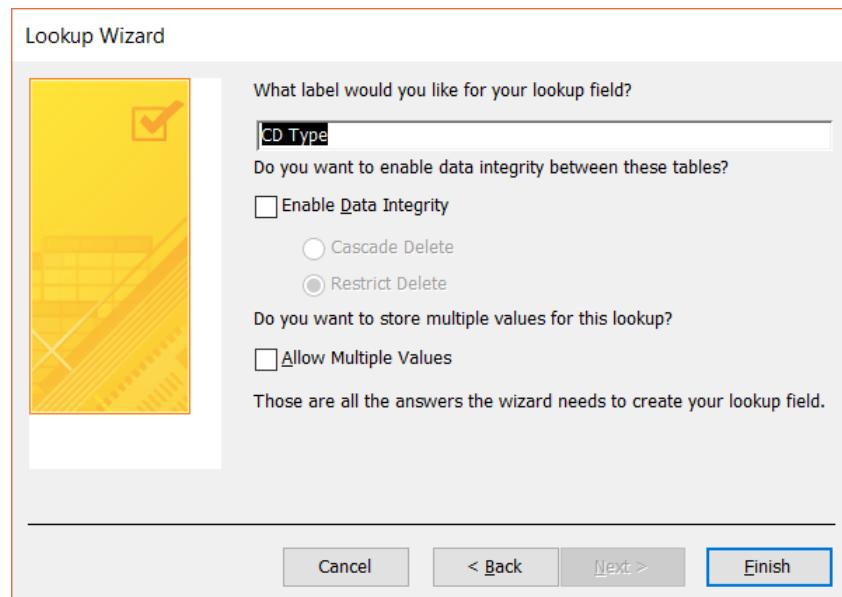
6. Select the *CD Type* field and either **double-click** it or click on the **>>** button to move it in to the **Selected Fields:** list. Click **Next** when it appears as the example above.



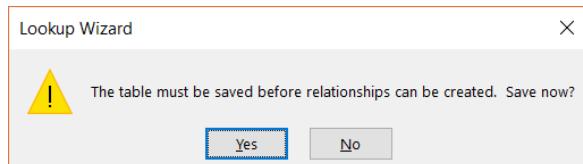
7. Here you can choose to have the *CD Types* sorted in alphabetical order as shown above. Click **Next** when ready.



8. Adjust the width of the column if necessary. This will determine how wide the drop-down list will be. Click **Next**.



9. Leave the field name as *CD Type* and click **Finish**.



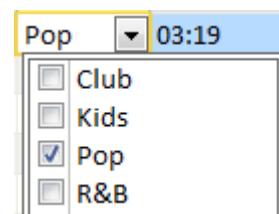
10. Click **Yes** to save the changes to the table design and to create a relationship between the *CDS* & *CD TYPES* tables.
11. The wizard has made several changes to the field properties. To see these changes, make sure the **CD Types** field is still selected and click on the **Lookup** tab in the properties section below.

General	Lookup
Display Control	Combo Box
Row Source Type	Table/Query
Row Source	SELECT [CD TYPES].[CD Type] FROM [CD TYPES] ORDER BY [CD Type];
Bound Column	1
Column Count	1
Column Heads	No
Column Widths	1.588cm
List Rows	16
List Width	1.587cm

When you become familiar with the properties shown here, you can skip the wizard and quickly create lookup fields by modifying these properties directly.

12. Close the table. The wizard should have already saved any necessary changes.
13. Open the *SONGS* table in design view.
14. Select the *Genre* field.
15. Follow the previous steps to create a lookup field for Genres (based on information from the *GENRES* table).

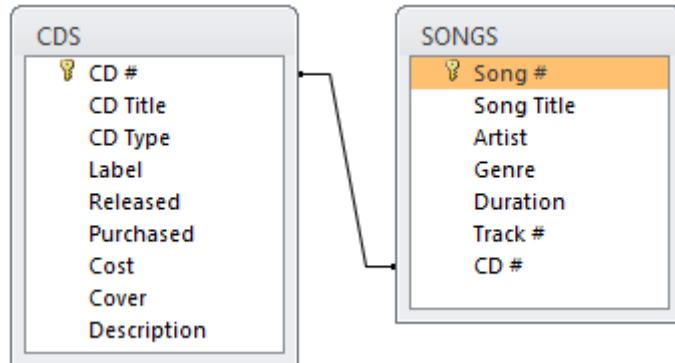
**Note** In the last step of the wizard there is an option to allow multiple values. This will result in a combo box where more than one option can be ticked. We won't do that for now though.  **Allow Multiple Values**



### Exercise 7. Checking Lookup Relationships

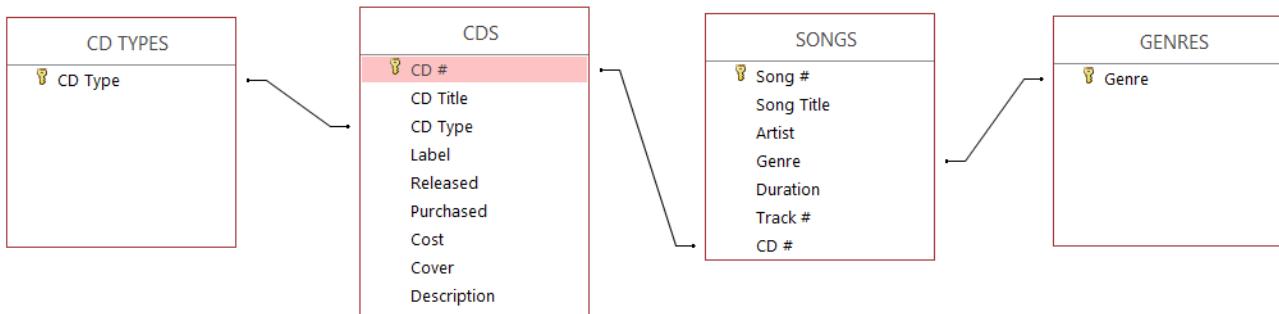
The lookup wizard would have created table relationships for both lookup fields. We can check this by looking in the relationships window.

1. Select **Database Tools** from the **Ribbon** and click the **Relationships** icon.



You will see the two tables that we created a relationship for earlier. The other relationships that have been created aren't currently visible.

2. Click the All Relationships icon.
3. Move the tables around so that it is easier to see the relationships between each table. A suggested layout is shown below.



4. Press **[Ctrl] [S]** or click the **Save** icon to save the relationship layout changes.
5. Close the relationships window.

### Exercise 8. Testing the Lookup Fields

1. Double-click the *CDS* table to open it.
2. Click in the *CD Title* field below *Stoney*.
3. Enter *Guardians of the Galaxy: Awesome Mix Vol. 2* for the new **CD Title**.
4. In the **CD Type** field there will now be a combo box (drop-down list). Use the list to select *Soundtrack* for the **CD type**.
5. Complete the rest of the information as follows:

CD Title	CD Type	Label	Released	Purchased	Cost
Stoney	Album	UMG Recordings	2016	13/04/17	\$19.99
Guardians of the Galaxy: Awesome Mix Vol. 2	Soundtrack	Marvel Music Inc	2017	17/01/19	\$15.99



6. Open the image named **Awesome Mix 2.jpg** in your photo viewer application and copy it in to your Cover field.
7. Click the + to the left of the new record to view the fields from the *SONGS* table.
8. Enter the song information shown below. You can now use the combo box to select the genre of each song. Note that since each genre starts with a different letter, you can press the first letter to select a genre from the list without using your mouse.

SONGS					
Song Title	Artist	Genre	Duration	Track #	
Mr. Blue Sky	Electric Light Orchestra	Rock	05:03	1	
Fox On The Run	Sweet	Rock	03:25	2	
Lake Shore Drive	Aliotta Haynes Jeremiah	Rock	03:49	3	
The Chain	Fleetwood Mac	Rock	04:27	4	
Bring It On Home To Me	Sam Cooke	Rock	02:43	5	
Southern Nights	Glen Campbell	Rock	02:57	6	
My Sweet Lord	George Harrison	Rock	04:37	7	
Brandy (You're A Fine Girl)	Looking Glass	Rock	03:03	8	
Come A Little Bit Closer	Jay & The Americans	Pop	02:46	9	
Wham Bam Shang-a-Lang	Silver	Rock	03:32	10	
Surrender	Cheap Trick	Rock	04:14	11	
Father And Son	Cat Stevens	Rock	03:39	12	
Flash Light	Parliament	Funk	04:28	13	
Guardians Inferno (feat. David Hasselhoff)	The Sneepers	Pop	03:16	14	

9. Close the table when the CD and Song information is entered as shown.

**Note** If you want to add or edit the information that appears in the lookup lists, all you need to do is open the tables and make the necessary changes/additions to the records in those tables.