Unit XVIII Assignment II **Resubmission**

By Nathan Windisch

PIII: Implementing a Database

The following screenshots are the tables that I have made within my database. They all have no redundant data within them, and they all have 10 rows of data that follow the data dictionary, which can be seen on the next page.

The following image is the Customers table.

tblCustomer | tblGardener | tblJob

CustomerI	CustomerFirstNan	CustomerLastName	CustomerEmail
0	Ryan	Krage	ryankrage@email.
1	Nathan	Windisch	nathanwindisch@
2	Jamie	Wagstaff	jamiewagstaff@er
3	Megan	Clarke	meganclarke@em
4	Ruben	Small	rubensmall@email
5	James	Phillips	jamesphillips@emc
6	Charlie	Dodd	charliedodd@emc
7	Sachin	Dougall	sachindougall@er
8	Rebekah	Kinchin	rebekahkinchin@e
9	Conor	Mehring	conormehring@er
10	Thomas	Widdis	thomaswiddis@em
0			

The following image is the Employees table.

tblCustomer | tblGardener | tblJob

Gardener	GardenerFirstName	GardenerLastNai
0	Raymond	Gonzales
1	Betty	Lewis
2	Frances	Perry
3	Steven	Sanders
4	Brian	Bennett
5	Keith	Edwards
6	Lori	Parker
7	Norma	Flores
8	Arthur	Thompson
9	Jesse	Murphy

The following image is the Jobs table.

tblCustomer | tblGardener | tblJob

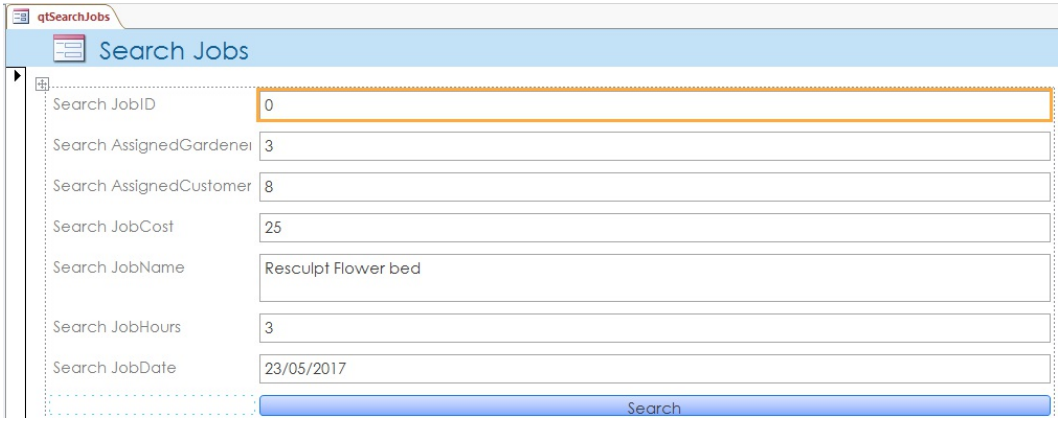
JobID	AssignedGardener	AssignedCustomer	JobCost	JobName	JobHours	JobDate
0	3	8	25	Resculpt Flower bed	3	23/05/2017
1	7	9	250	Remove Tree from garden	2	22/05/2017
2	2	6	10	Mow Lawn	4	01/06/2017
3	8	1	50	Plant new flowerbeds	1	31/05/2017
4	5	5	15	Remove Weeds	1	27/05/2017
5	4	7	25	Mow lawn and remove weeds	12	28/05/2017
6	1	8	70	Gather flowers from suppliers	1	30/05/2017
7	0	3	100	Plant new trees	3	24/05/2017
8	7	6	50	Add rocks to rockery	0	24/05/2017
9	5	0	130	Replace patio	0	21/05/2017

The following is the aforementioned data dictionary.

Field Name	Data Type	Maximum Field Length	Description
EmployeeID	UUID	16 Characters	Used as a unique identifier for each employee
EmployeeName	String	64 Characters	Used as a human readable format for identifying customers
EmployeeRole	String	16 Character	Used for the permission system
RoleDescription	String	128 Character	A brief description of their role
RoleWageMin	Double	10 Digits	The minimum amount that a person in that role can be paid
RoleWageMax	Double	10 Digits	The maximum amount that a person in that role can be paid
CustomerID	UUID	16 Characters	used as a unique identifier for each customer
CustomerName	String	64 Characters	Used as a human readable format for identifying customers
CustomerBalance	Double	10 Digits	The amount of credits that the customer has
CustomerAddress	String	64 Characters	The address of the customer
CustomerPostcode	String	8 Characters	The postcode of the customer
CustomerPhoneNumber	Integer	15 Characters	Either the mobile or the landline number of the customer
ItemID	UUID	16 Characters	Used as a unique identifier for each item
ItemName	String	64 Characters	Used as a human readable format for identifying items
ItemPrice	Double	10 Digits	The price of the item
ItemDescription	String	128 Characters	A brief description of the product
ItemTags	ArrayList	64 Characters Per Entry	A list of tags that can be applied to the item

PIV: Forms

The following form is a form which allows users to search though the Jobs table. The following image is taken from that search query:



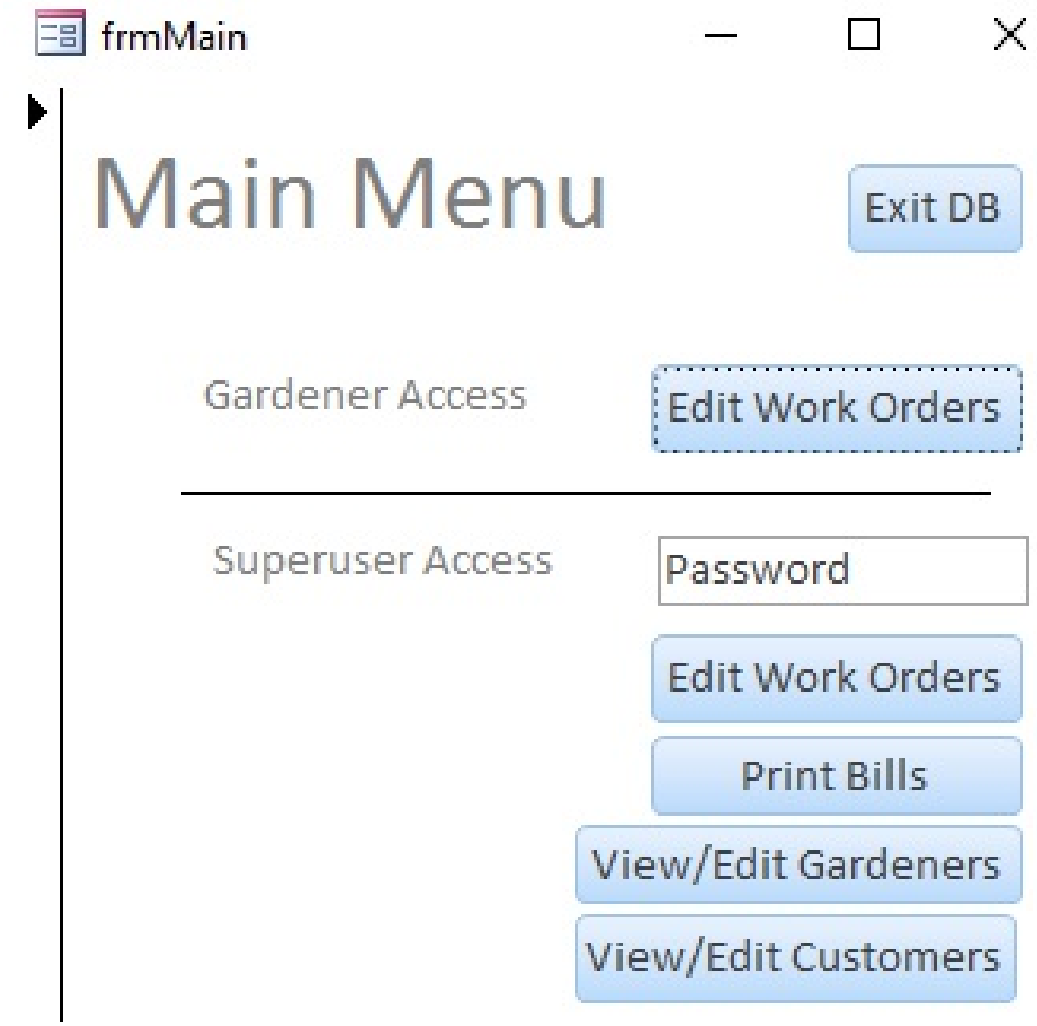
This is the output that it will give if the values in the query are set to be "ProductID == 1":

JobID	JobName	JobPrice	AssignedGardener	AssignedCustomer
1	Clearance	250	7	3

Other queries can be made with this form, and this is the output that it will give if the values in the query are set to "AssignedCustomer == 3":

JobID	JobName	JobPrice	AssignedGardener	AssignedCustomer
1	Clearance	250	7	3
2	Maintenance	2000	3	3
3	Tidying	300	5	3
4	Redesign	5000	9	3

The following is a secondary form that is used for adding data to the database. It requires a password so that the user can verify that they have the correct access:



And this is a hypothetical outcome that the form being filled out could create:

GardenerID	GardenerFirstName	GardenerLastName
10	David	Tennant

PV: Querying the Database

I shall now create some reports that show the results of the database being queried. I shall show the total sales of the business, the top five customers and the total amount of customers to gardeners.

The following image is the output of the total sales:

qrTotalJobs

Total Jobs	Total Income
10	725

The following image is the output of the top five customers:

qrTop5

AssignedCustomer	Services Bought
6	2
5	1
3	1
1	1
0	1

The following image is the output of the amount of customers per gardener:

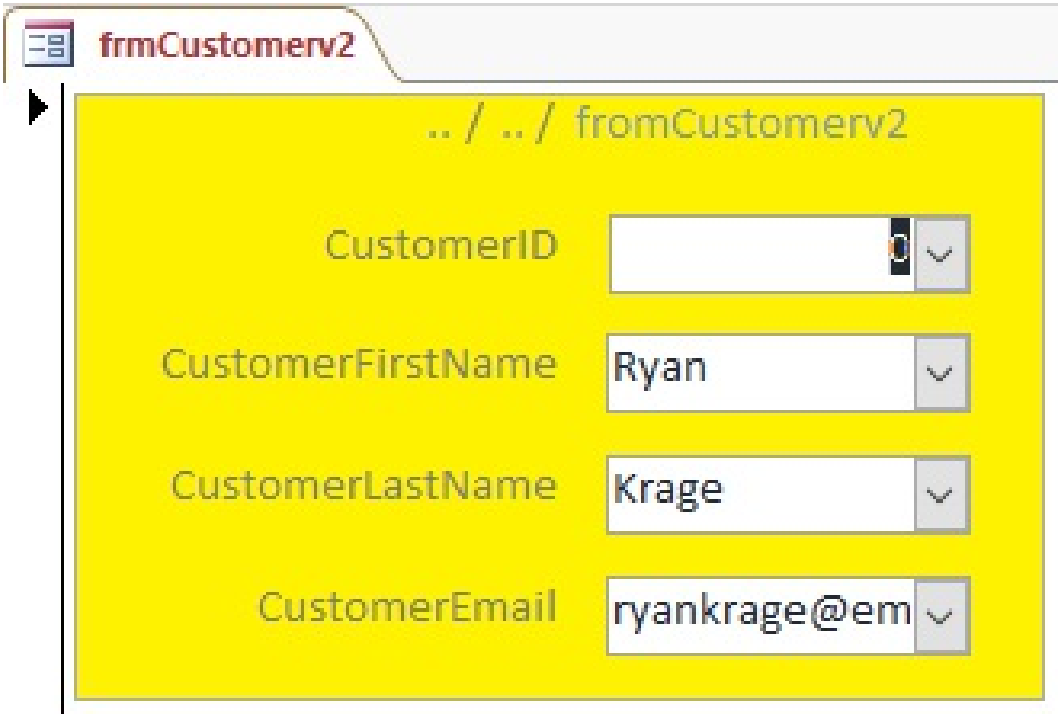
qrCustomersPerGardener

AssignedGardener	CountOfAssignedGardener
7	2
5	2
8	1
4	1
3	1
2	1
1	1
0	1

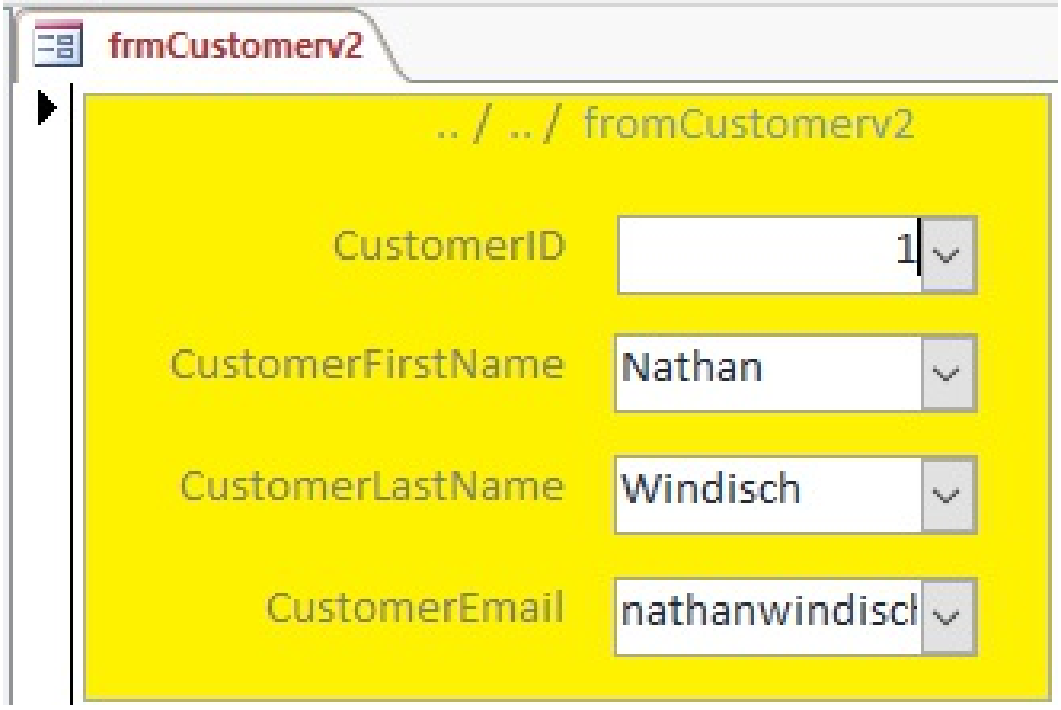
PVI: Form Themes and Normalization

Form Themes

The following image is a form that has been stylised to be more appealing to the user with a bright yellow background, navigation at the top in the form of breadcrumbs so that the user can go to previous sections, and a dropdown box to allow easier data addition, which will auto update the entire form when any of the boxes are changed.



The following image is the same form but with different data that has been auto-updated.



Normalization

Within databases there are three different forms of normalization.

- **UNF**, or **Unnormalized** is when there is lots of data redundancy and can contain many data structures within a single hallmark.
- **1NF**, or **First Normal Form** is when each field in a table does not contain the same type of information. An example of this would be in a customer list where each table would only contain one phone number.
- **2NF**, or **Second Normal Form** is when each field in a table must be a function of the other fields in the table if it is not a determiner of the contents of that field
- **3NF**, or **Third Normal Form** is when there is absolutely no duplicate information within the table. For example, if two tables both require a phone number field, that information would be placed into a separate table, and the two other tables would then information that they want such as the phone number data, via an index field in the newly created phone number table. Any and all change that are made to a phone number will now automatically update and be reflected to all tables that use the phone number table.

Examples

The following are some examples of different relational databases that I have mocked up with items based around gardening:

UNF

ID	Type	Price
01	daisy	07.49
02	iris	02.24
03	lavender	06.87
04	pansy	21.05

1NF

Unnormalized

ID	Price
01	07.49
02	02.24
03	06.87
04	21.05

1NFified

ID	Type
01	daisy
02	iris
03	lavender
04	pansy

2NF

Unnormalized

ID	Client ID	Balance
01	110	96.03
02	420	05.24
03	911	12.62

2NFified

ID	Client ID
01	110
02	420
03	911

Client ID	Balance
110	96.03
420	05.24
911	12.62

3NF

Unnormalized

Flower ID	Bloom Season ID	Literal Season Name	Price
221	001	Spring	05.88
486	002	Summer	09.15

683	001	Spring	35.73

3NFified

Flower ID	Bloom Season ID	Price
221	001	05.88
486	002	09.15
683	001	35.73

Bloom Season ID	Literal Season Name
001	Spring
002	Summer

PVII: Testing and Logging

Test Log

ID	Date	Summary	Details	Expected Outcome	Passed?	Problem ID
1	18/3/17	Form Gardener First Name Entry Check	Check if input is text and one word only	Only one word with no numbers/symbols can be inputted	Yes	N/A
2	18/3/17	Form Gardener Last Name Entry Check	Check if input is text and one word only	Only one word with no numbers/symbols can be inputted	Yes	N/A
3	18/3/17	Form Customer First Name Entry Check	Check if input is text and one word only	Only one word with no numbers/symbols can be inputted	Yes	N/A
4	18/3/17	Form Customer Last Name Entry Check	Check if input is text and one word only	Only one word with no numbers/symbols can be inputted	Yes	N/A
5	18/3/17	Form Customer First Name Entry Check	Check if input is text and one word only	Only one word with no numbers/symbols can be inputted	Yes	N/A
6	18/3/17	Generate Bill Report	Check if the bill will be generated	A bill will be generated with dynamically changed with the new data	Yes	N/A
7	18/3/17	Generate Top5 Report	Check if the top five list will be generated	A list of the top five customers will be generated with dynamically changed with the new data	Yes	N/A
8	18/3/17	Generate Total Sales Report	Check if the sales list will be generated	A list of all the sales will be generated with dynamically changed with the new data	Yes	N/A
9	18/3/17	Print Bill via Linux	Check if the bill can be printed on Linux	The bill will be sent to the printer	Yes	N/A
9	18/3/17	Print Bill via Windows	Check if the bill can be printed on Windows	The bill will be sent to the printer	No	1
10	18/3/17	Referential Integrity	Check if the database has referential integrity	The database will have referential integrity	Yes	N/A

Fault Log

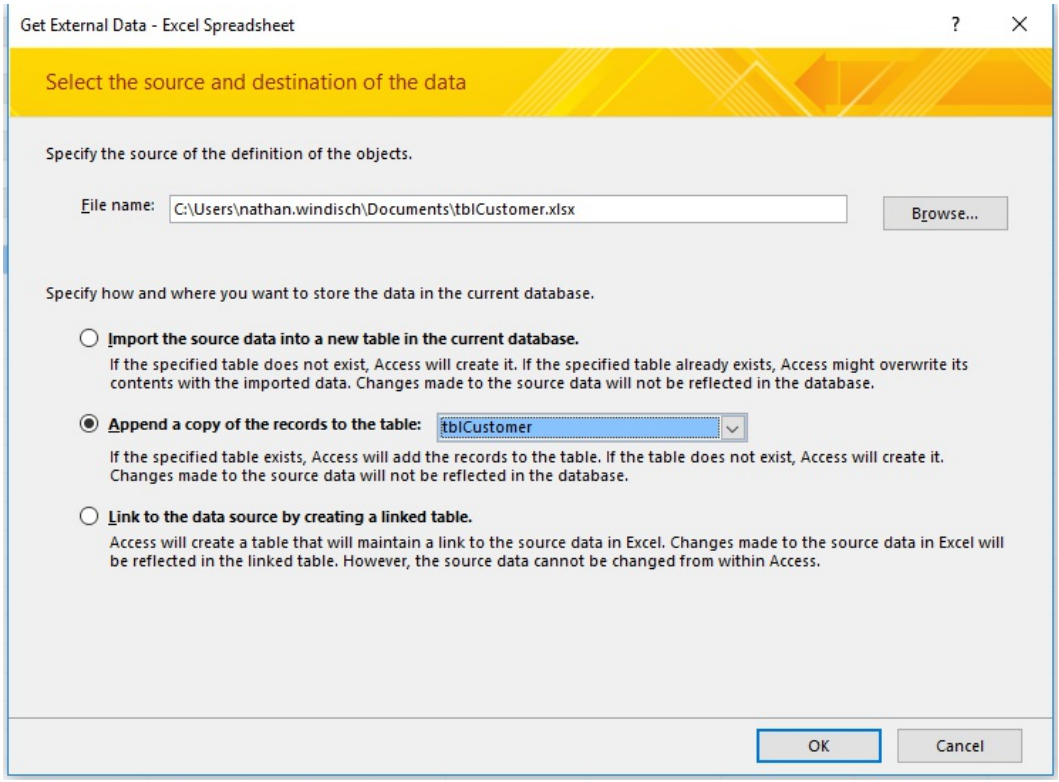
Problem ID	Test ID	Problem	Solution
1	9	The bill wouldn’t print on Linux	Create a new method for printing via Linux, see D2

MII: Importing Data

The following image is the file that I was going to use to import:

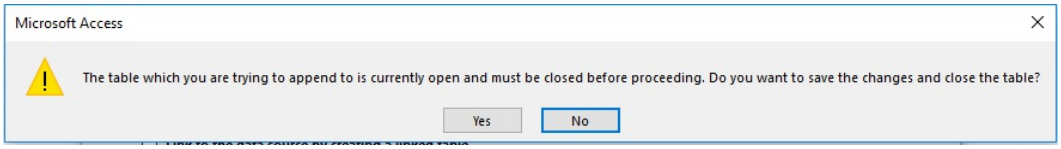
CustomerID	CustomerFirstName	CustomerLastName	CustomerEmail
11	Silvester	McCoy	silvestermccoy@email.net
12	Thomas	Baker	thomasbaker@email.net
13	Jon	Pertwee	jonpertwee@email.net
14	Paul	McGann	paulmcgann@email.net
15	Christopher	Eccleston	christophereccleston@email.net

I used a wizard within Access to help me import the data, as follows:

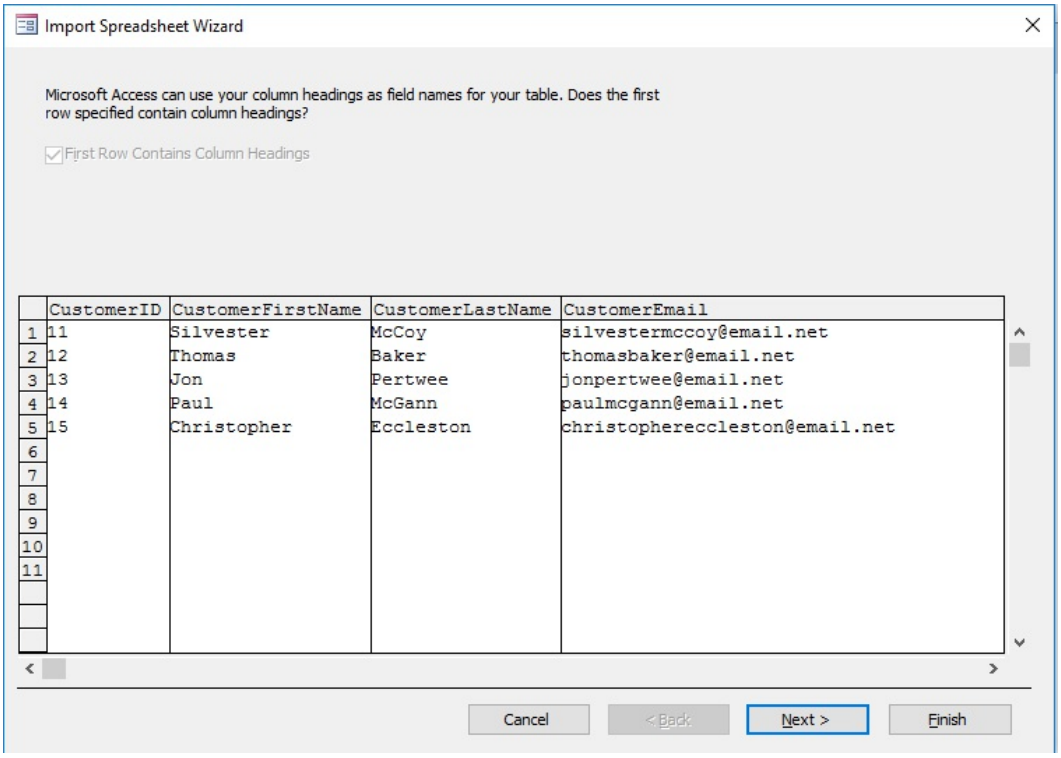


In the above image I set the data to append to the end of the table.

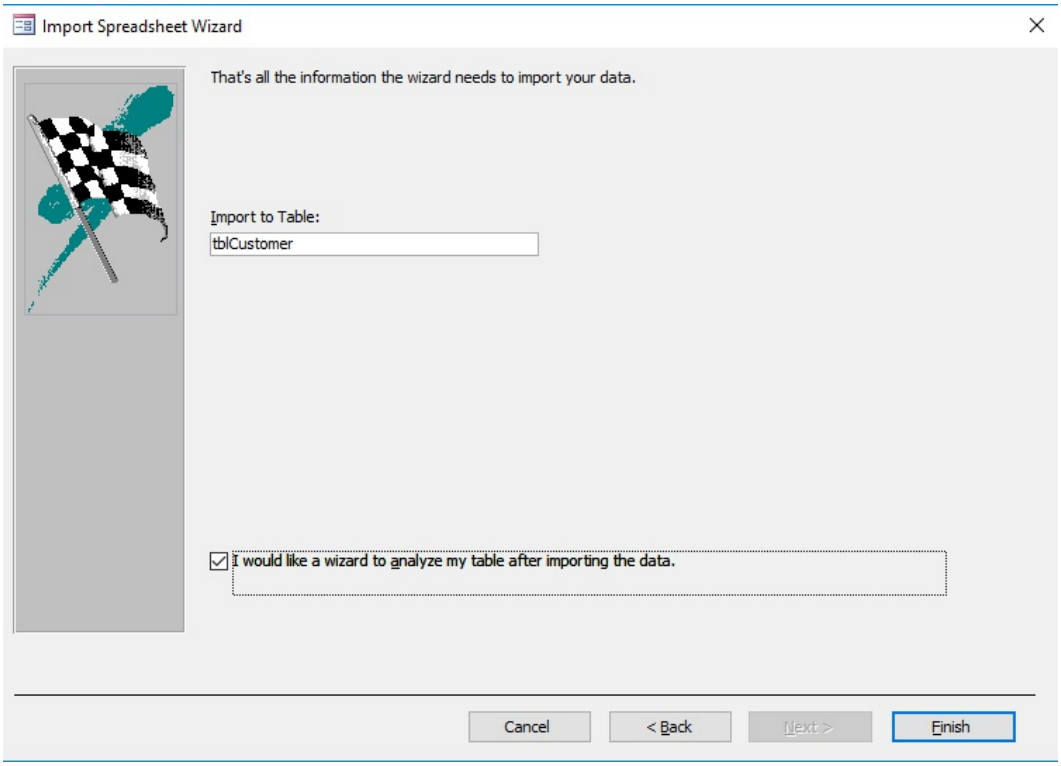
I came across an error which was caused by the fact that I already had the table open, so I had to close it:



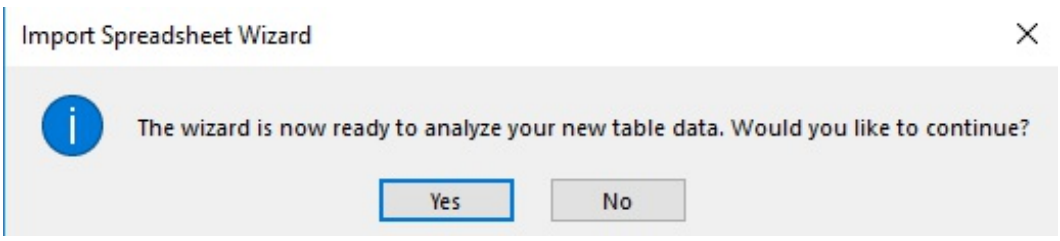
In the image below I had to confirm that all of the data was in the right column, which it was:



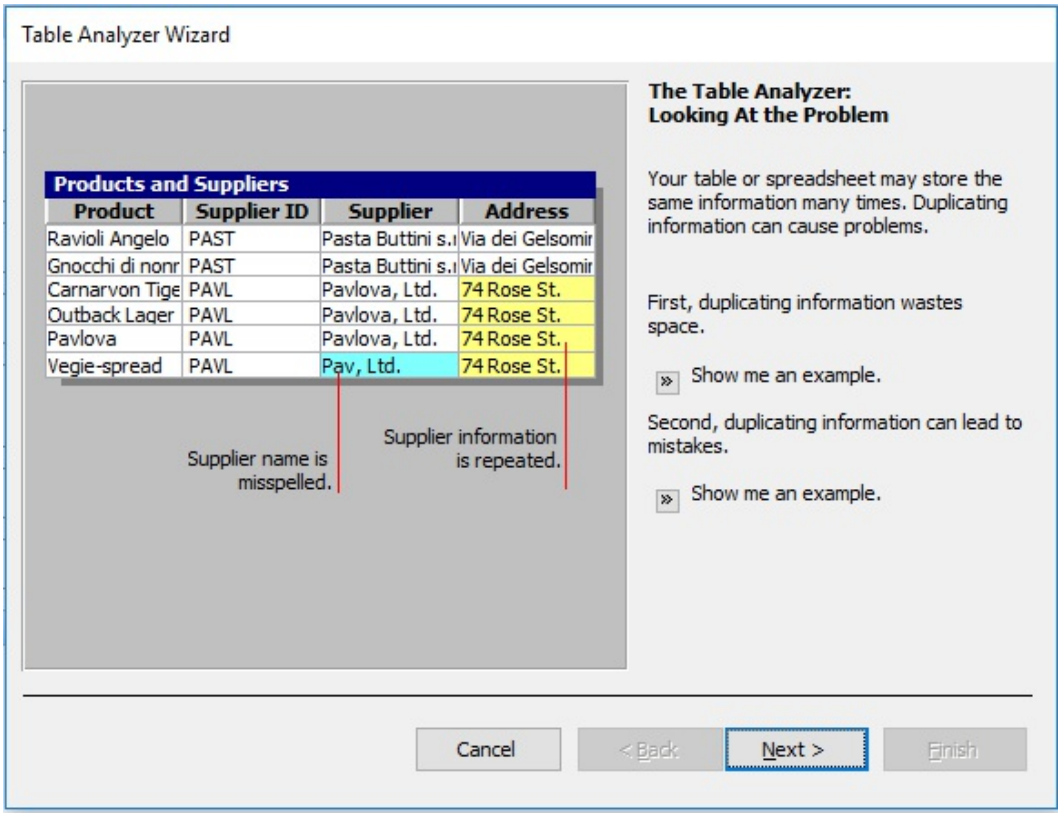
In the next image, I had to confirm what table I wanted to import the data to, and confirm if I wanted the wizard to analyse my data:



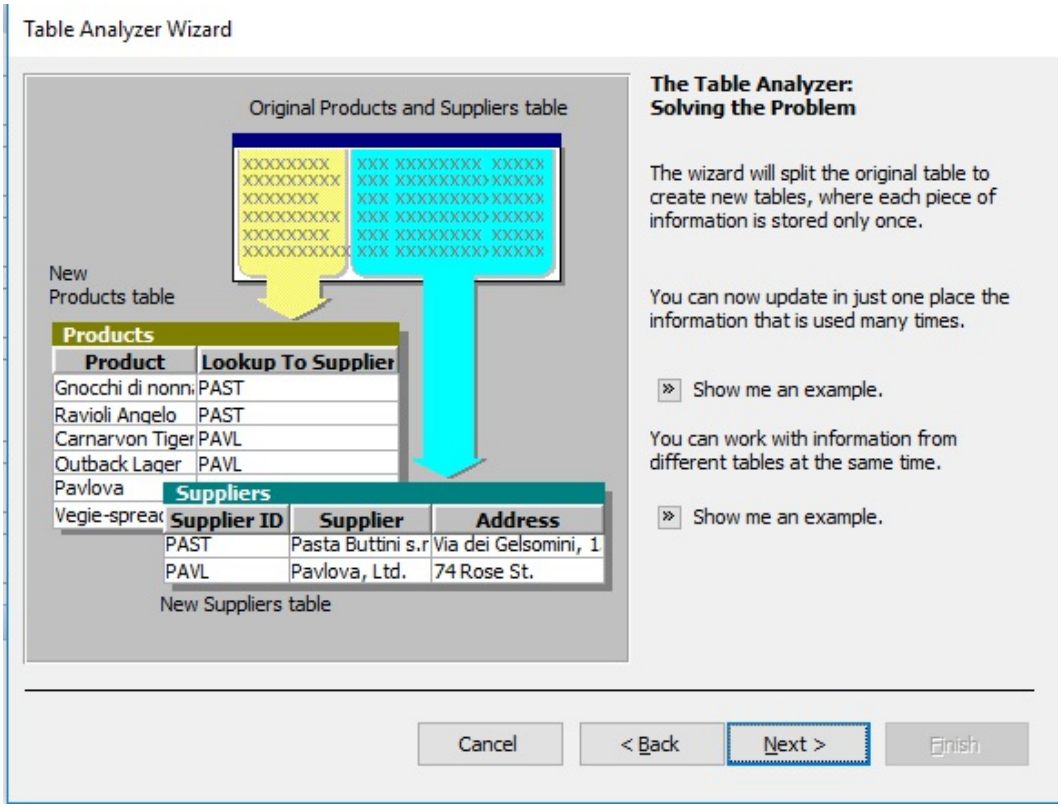
Here, I had to start the analysis:



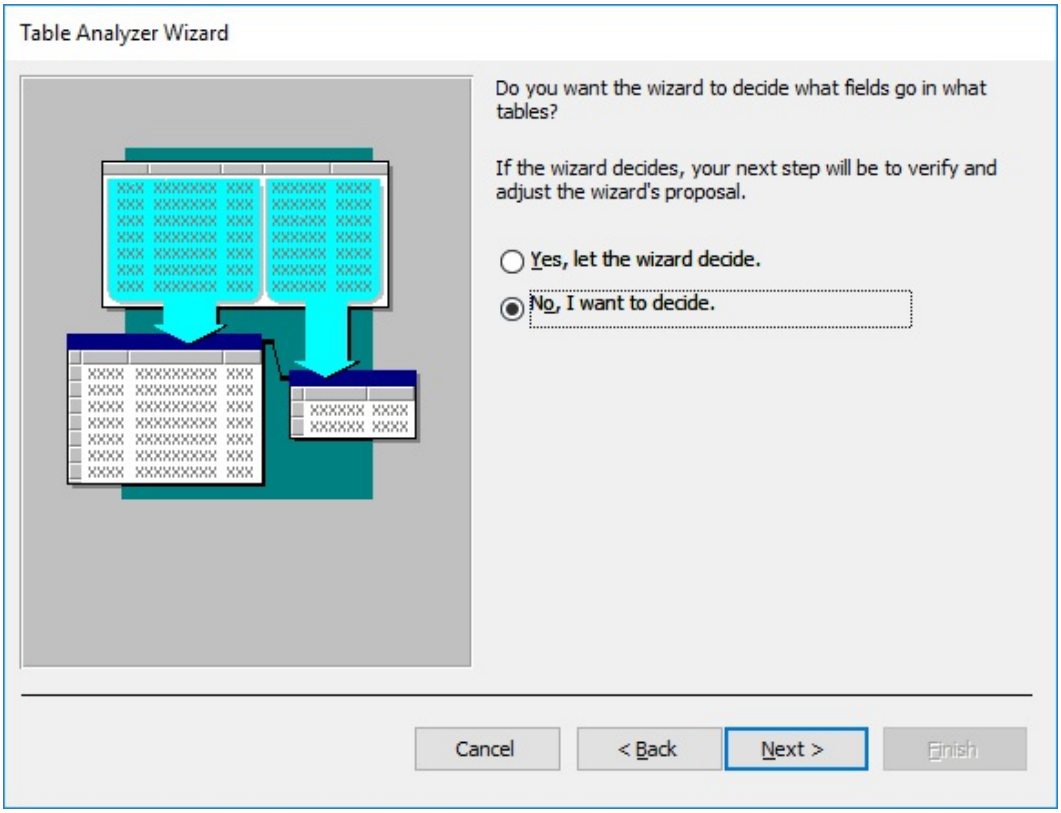
This was the first part of the analysis wizard, it started by explaining what one of the issues might be:



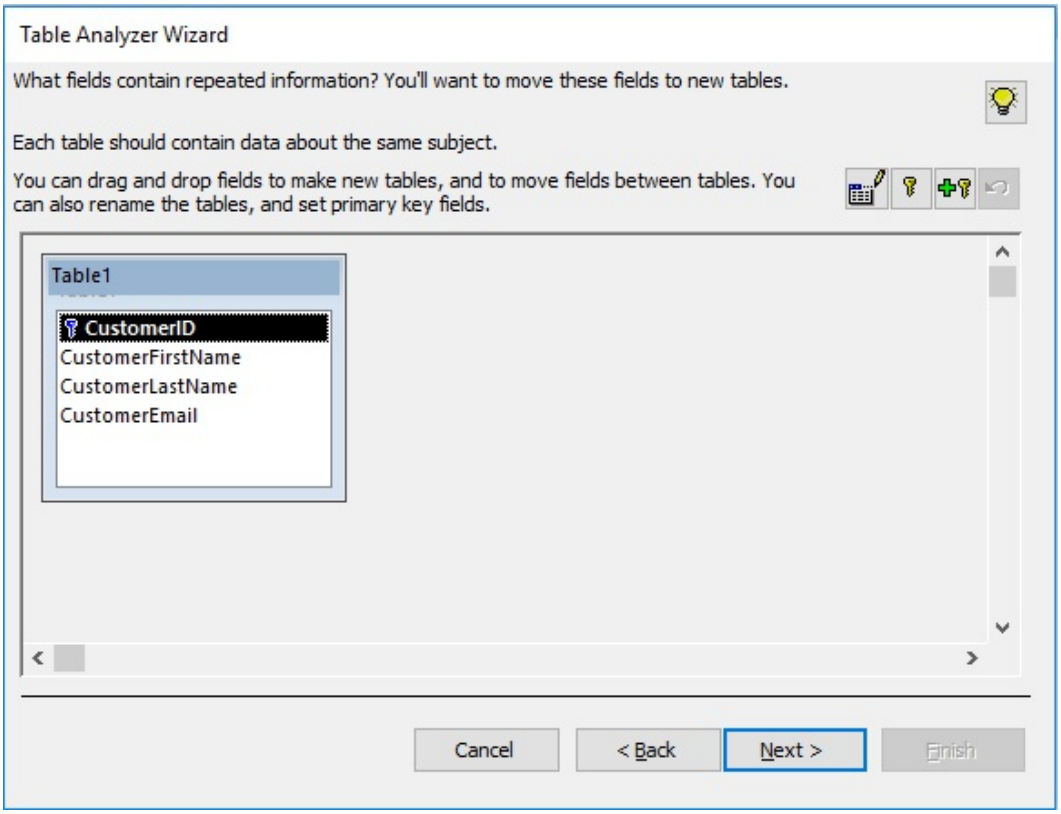
Next, it showed me how it was going to fix the problem:



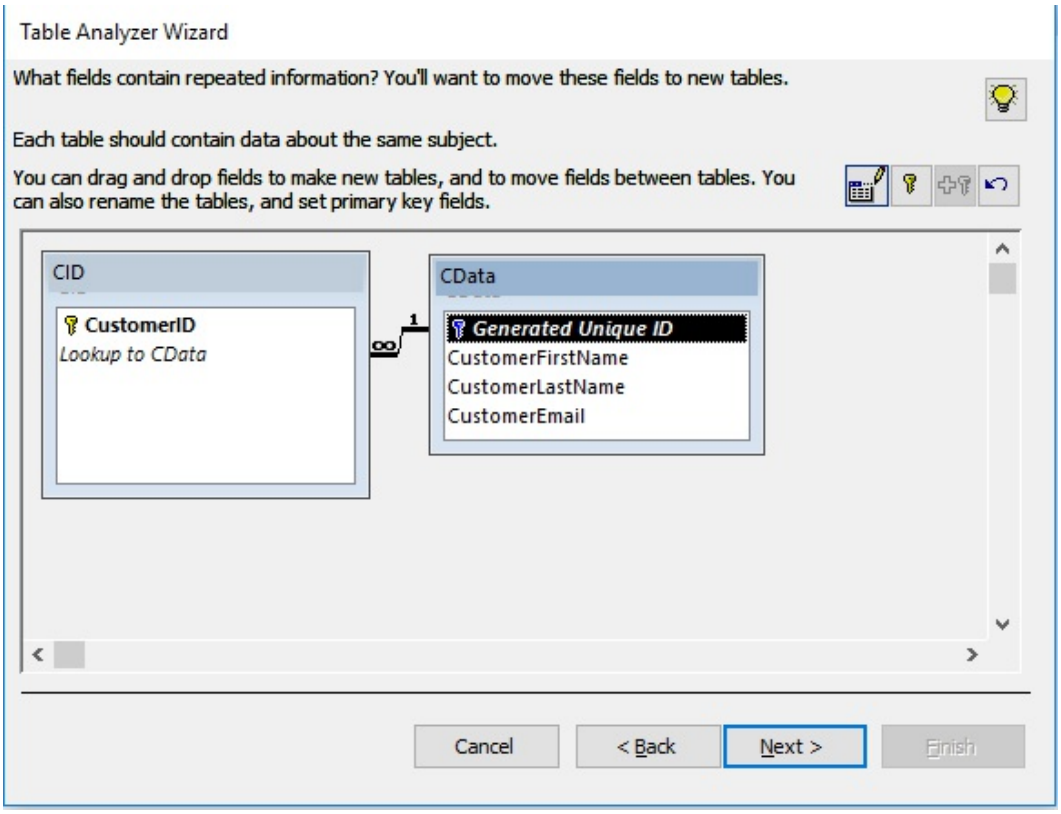
I wanted to decide what I was going to do, so that I could have full control over the data:



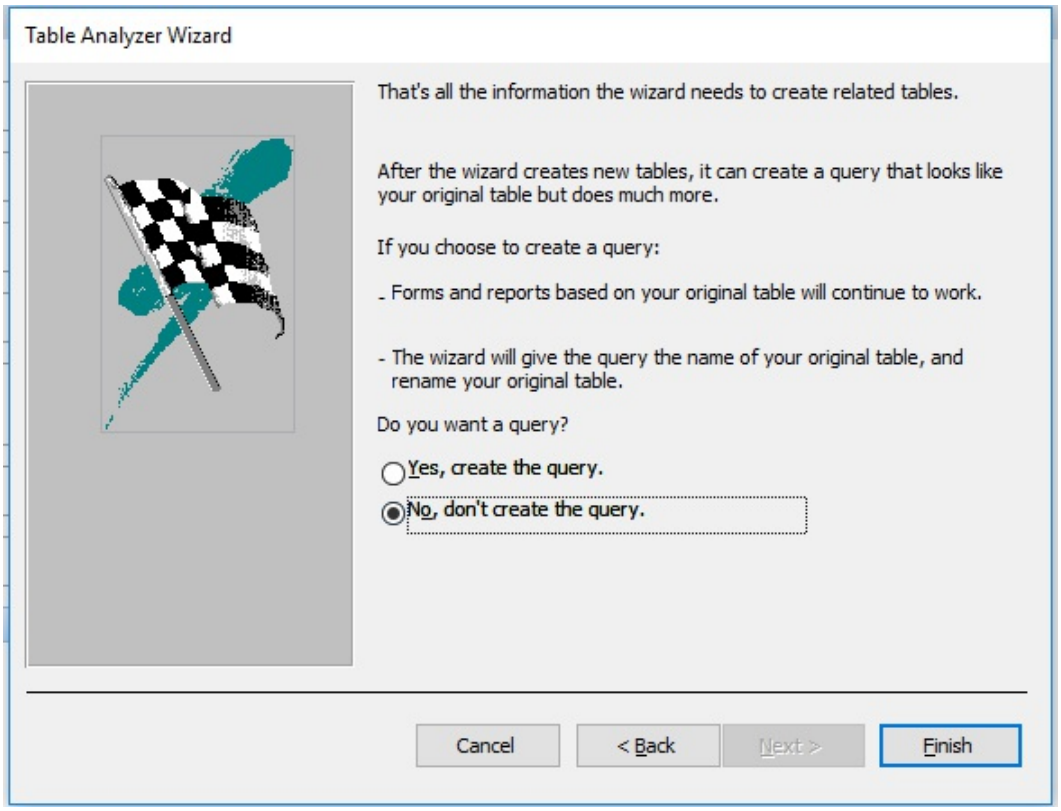
I confirmed that all of the data is correct, as seen below:



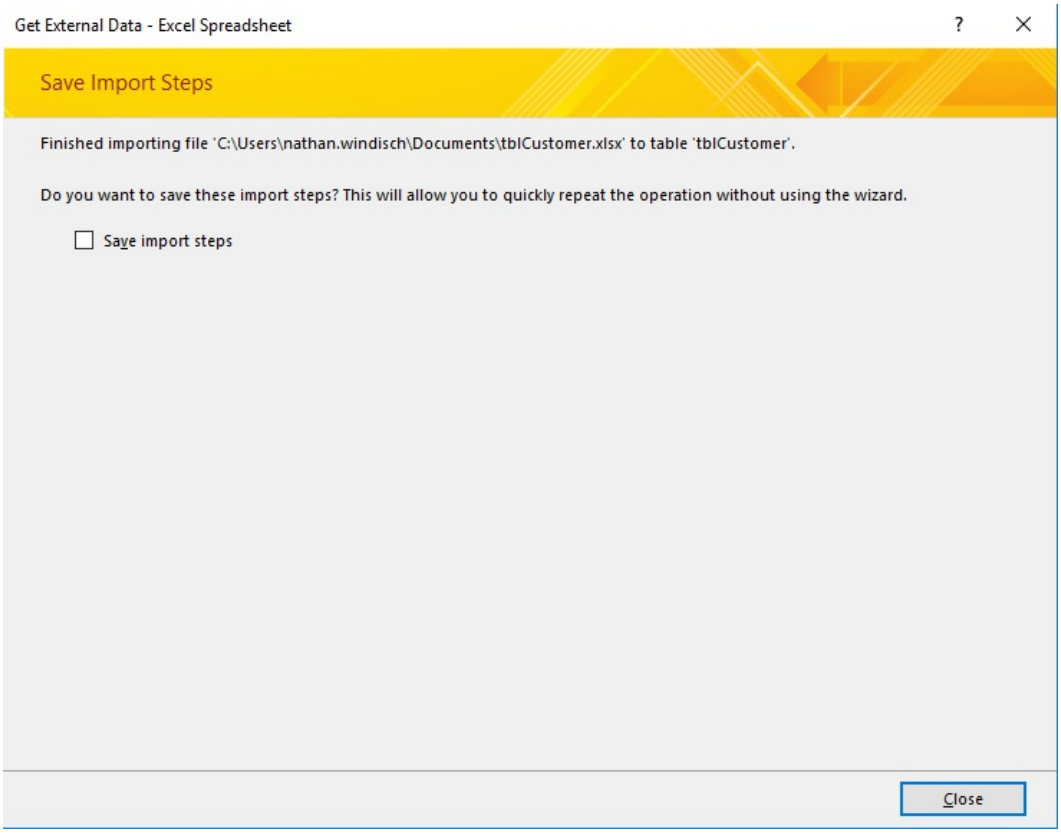
I split the table in two, so that the main lookup could be done in the "primary" table, `CID`, and the secondary data which might not be unique was in the "secondary" table, `CData`.



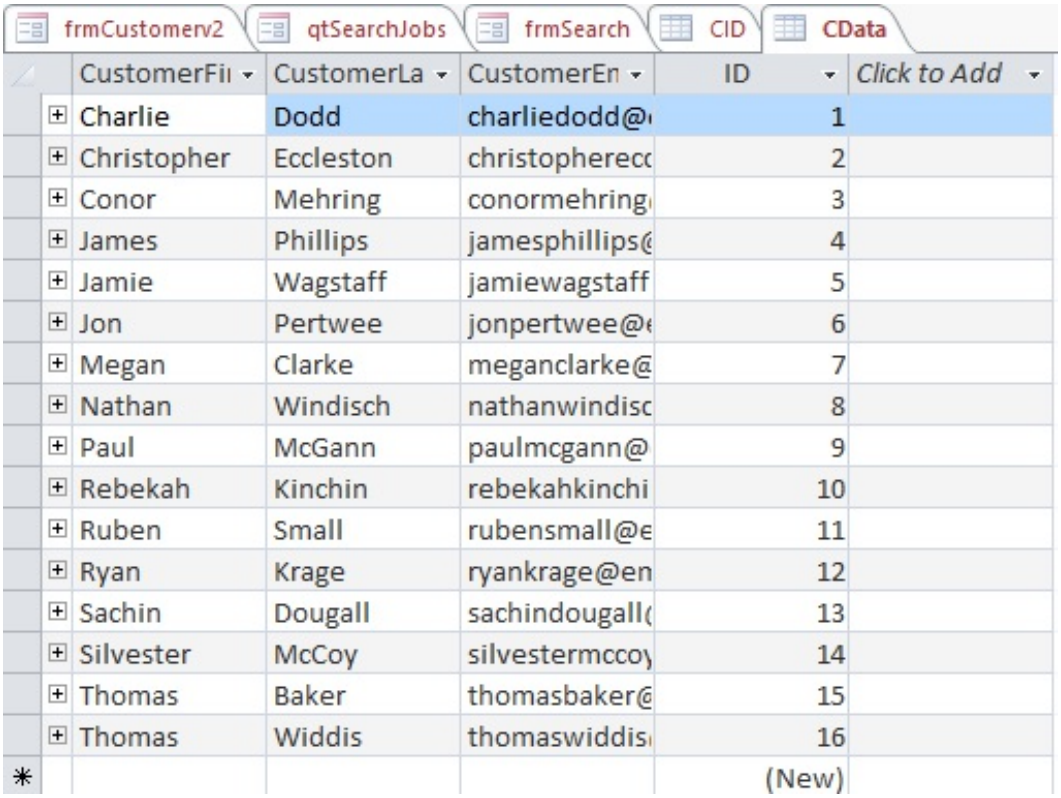
I did not want to create a new query:



The data was successfully imported:



Albeit, the columns were a tad unordered:



frmCustomerv2 | qtSearchJobs | frmSearch | CID | CData

CustomerFi	CustomerLa	CustomerEn	ID	Click to Add
Charlie	Dodd	charliedodd@	1	
Christopher	Eccleston	christopherecc	2	
Conor	Mehring	conormehring	3	
James	Phillips	jamesphillips@	4	
Jamie	Wagstaff	jamiewagstaff	5	
Jon	Pertwee	jonpertwee@	6	
Megan	Clarke	meganclarke@	7	
Nathan	Windisch	nathanwindisc	8	
Paul	McGann	paulmcgann@	9	
Rebekah	Kinchin	rebekahkinchi	10	
Ruben	Small	rubensmall@e	11	
Ryan	Krage	ryankrage@en	12	
Sachin	Dougall	sachindougall(	13	
Silvester	McCoy	silvestermccoy	14	
Thomas	Baker	thomasbaker@	15	
Thomas	Widdis	thomaswiddis	16	
			(New)	

Fixed, but the IDs were a bit odd...

frmCustomerv2 | qtSearchJobs | frmSearch | CID | CData

ID	CustomerFirstName	CustomerLastName	CustomerEmail
1	Charlie	Dodd	charliedodd@email.net
2	Christopher	Eccleston	christophereccleston@email.net
3	Conor	Mehring	conormehring@email.net
4	James	Phillips	jamesphillips@email.net
5	Jamie	Wagstaff	jamiewagstaff@email.net
6	Jon	Pertwee	jonpertwee@email.net
7	Megan	Clarke	meganclarke@email.net
8	Nathan	Windisch	nathanwindisch@email.net
9	Paul	McGann	paulmcgann@email.net
10	Rebekah	Kinchin	rebekahkinchin@email.net
11	Ruben	Small	rubensmall@email.net
12	Ryan	Krage	ryankrage@email.net
13	Sachin	Dougall	sachindougall@email.net
14	Silvester	McCoy	silvestermccoy@email.net
15	Thomas	Baker	thomasbaker@email.net
16	Thomas	Widdis	thomaswiddis@email.net

Because I was looking at the `CData` table:

frmCustomerv2 | qtSearchJobs | frmSearch | CID | CData

ID	CustomerFirstName	CustomerLastName	CustomerEmail
1	Charlie	Dodd	charliedodd@email.net
CustomerID	Click to Add		
6			
2	Christopher	Eccleston	christophereccleston@email.net
CustomerID	Click to Add		
15			
3	Conor	Mehring	conormehring@email.net
4	James	Phillips	jamesphillips@email.net
5	Jamie	Wagstaff	jamiewagstaff@email.net
6	Jon	Pertwee	jonpertwee@email.net
7	Megan	Clarke	meganclarke@email.net
8	Nathan	Windisch	nathanwindisch@email.net
9	Paul	McGann	paulmcgann@email.net
10	Rebekah	Kinchin	rebekahkinchin@email.net
11	Ruben	Small	rubensmall@email.net
12	Ryan	Krage	ryankrage@email.net
13	Sachin	Dougall	sachindougall@email.net
14	Silvester	McCoy	silvestermccoy@email.net
15	Thomas	Baker	thomasbaker@email.net
16	Thomas	Widdis	thomaswiddis@email.net
(New)			

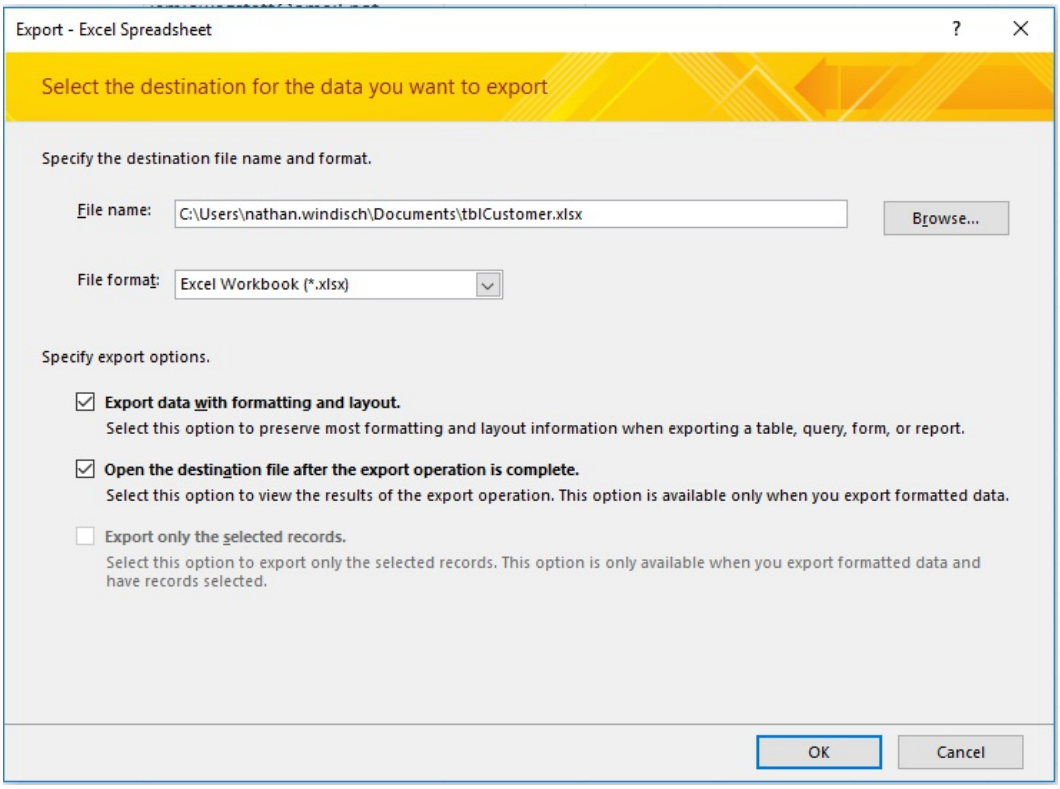
This meant that the IDs were not the same as the Customer IDs, which were in a separate table. The data is much more compact there:

frmCustomerv2 | qtSearchJobs | frmSearch | CID | CData

CustomerID	Lookup to CData
0	Ryan, Krage, ryankrage@email.net
1	Nathan, Windisch, nathanwindisch@email.net
2	Jamie, Wagstaff, jamiewagstaff@email.net
3	Megan, Clarke, meganclarke@email.net
4	Ruben, Small, rubensmall@email.net
5	James, Phillips, jamesphillips@email.net
6	Charlie, Dodd, charliedodd@email.net
7	Sachin, Dougall, sachindougall@email.net
8	Rebekah, Kinchin, rebekahkinchin@email.net
9	Conor, Mehring, conormehring@email.net
10	Thomas, Widdis, thomaswiddis@email.net
11	Silvester, McCoy, silvestermccoy@email.net
12	Thomas, Baker, thomasbaker@email.net
13	Jon, Pertwee, jonpertwee@email.net
14	Paul, McGann, paulmcgann@email.net
15	Christopher, Eccleston, christophereccleston@email.net

MIII: Exporting Data

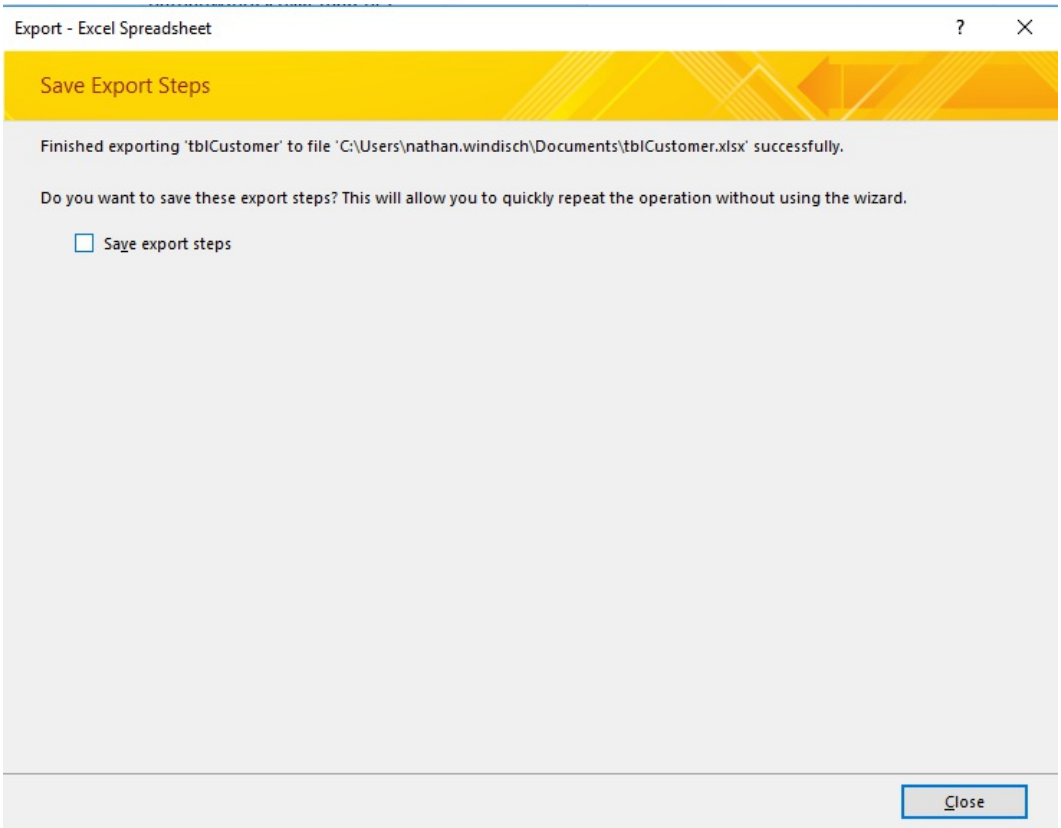
I used a wizard within Access to help me export the data, as follows:



The following image is the outcome that was generated, an Excel file with all of the data:

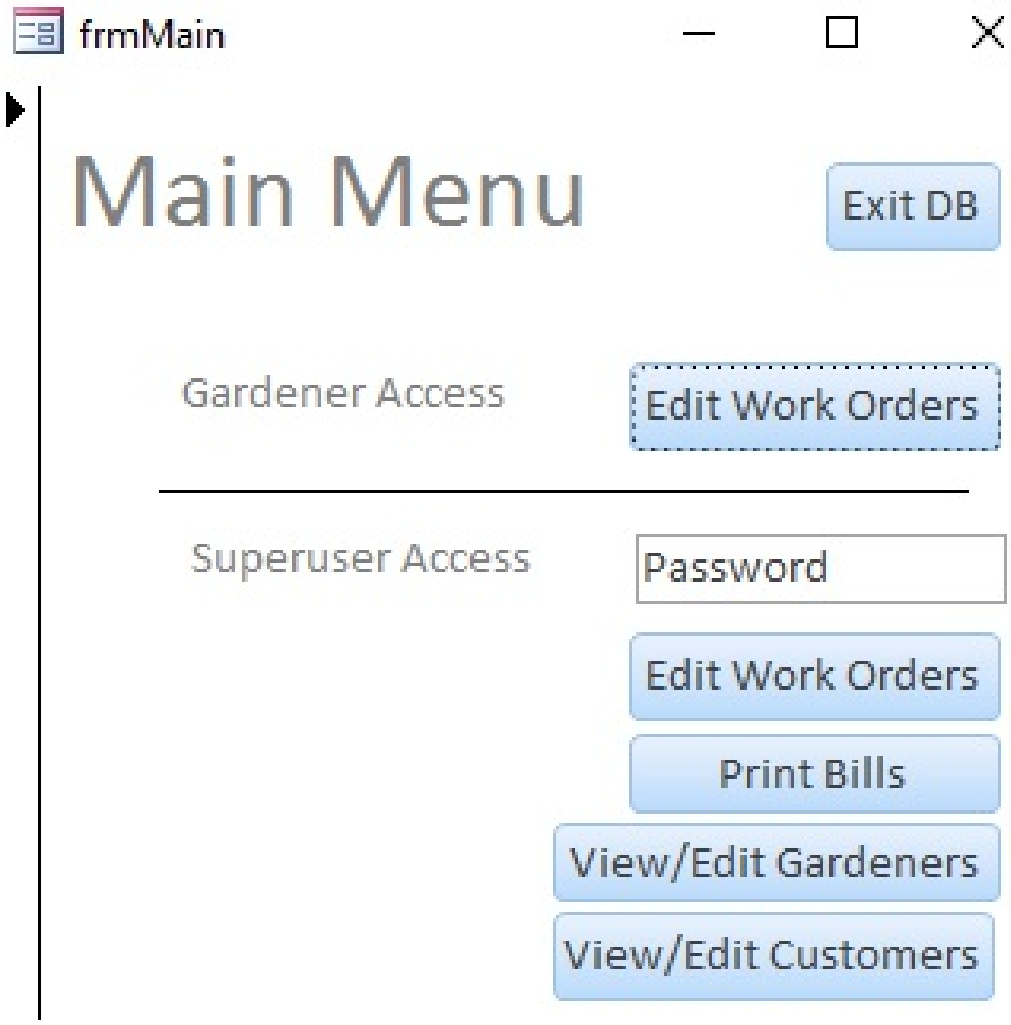
CustomerID	CustomerFirstName	CustomerLastName	CustomerEmail
0	Ryan	Krage	ryankrage@email.net
1	Nathan	Windisch	nathanwindisch@email.net
2	Jamie	Wagstaff	jamiewagstaff@email.net
3	Megan	Clarke	meganclarke@email.net
4	Ruben	Small	rubensmall@email.net
5	James	Phillips	jamesphillips@email.net
6	Charlie	Dodd	charliedodd@email.net
7	Sachin	Dougall	sachindougall@email.net
8	Rebekah	Kinchin	rebekahkinchin@email.net
9	Conor	Mehring	conormehring@email.net
10	Thomas	Widdis	thomaswiddis@email.net

This is the final step to show that everything was successful:



MIV: Advanced Features

The following image is of the main form which has a button on it that will print the bills when pressed:



The following is the code, written in Python

```
import subprocess

def print():
    printfile = open("print.txt", r)

    if platform.system() == "Linux" or platform.system() == "linux":
        lpr =  subprocess.Popen("/usr/bin/lpr", stdin=subprocess.PIPE)
        print("starting to print the file")
        lpr.stdin.write(printfile)
        print("file printing finished")
    else:
        print("your operating system is not currently supported")

print()
```

The code will also be executed whenever an order has been completed.

DII: Feedback

When I decided to start to look for feedback, I created a questionnaire with the following questions and format:

> Please rate the following questions that are prepended with an `*` out of 5, where 1 is terrible and 5 is excellent.
>
> > *Appearance:
> >
> > *Legibility:
> >
> > *Ease of Access:
> >
> > *Extensive Features:
> >
> > *Intuitive Design:
> >
> > *Overall Satisfaction:
> >
> > Improvement Suggestions:

I gave the questionnaire to one of my customers, Ryan Krage. The following are his answers.

> Please rate the following questions that are prepended with an `*` out of 5, where 1 is terrible and 5 is excellent.
>
> > *Appearance: 4
> >
> > *Legibility: 5
> >
> > *Ease of Access: 5
> >
> > *Extensive Features: 1
> >
> > *Intuitive Design: 5
> >
> > *Overall Satisfaction: 4
> >
> > Improvement Suggestions: The code to print the bill didn't work.

I took the criticisms given to me on-board and rewrote the code to work with the client's Windows setup, as the previous iteration of the code only worked with Linux. I added an option feature at the start of the program for the user to select what operating system they were running.

```
import platform
import subprocess

def print():
    printfile = open("print.txt", r)

    if platform.system() == "Windows" or platform.system() == "windows":
        print("starting to print the file")
        win32api.ShellExecute(
            0,
            "print",
            printfile,
            '/d:"%s"' % win32print.GetDefaultPrinter (),
            ".",
            0
        )
        print("file printing finished")
    elif platform.system() == "Linux" or platform.system() == "linux":
        lpr =  subprocess.Popen("/usr/bin/lpr", stdin=subprocess.PIPE)
        lpr.stdin.write(printfile)
    else:
        print("your operating system is not currently supported")

print()
```