

Dexter’s Phone Bill

Structure of a database management system (DBMS)

Creating DBMS queries

**Dexter’s problem**

I am angry. Lately, my phone bill has increased and I am not sure why. I have asked my phone provider to check my bill but they have told me that it is correct - but I want to check!

So I need you to do me a favour.

I want you to build a database management system (DBMS) that will calculate and display the cost of my phone bill.

I will then compare my phone bill to the amount calculated by the DBMS.

**Note:**

* Only voice calls are made using the phone
* Some calls are included for free within my phone plan
* The cost of other calls are charged at $0.03 per second

My phone bill for March 2018 was:

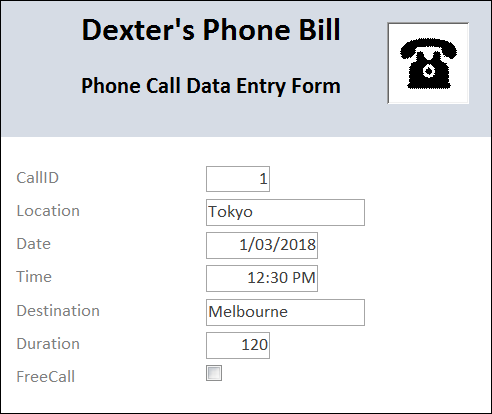
|  |
| --- |
| The Phone Company  Related image  $35.85  Please pay by 28 April 2018 |

**Activity – creating database queries.**

So far, we have created a form and a table.

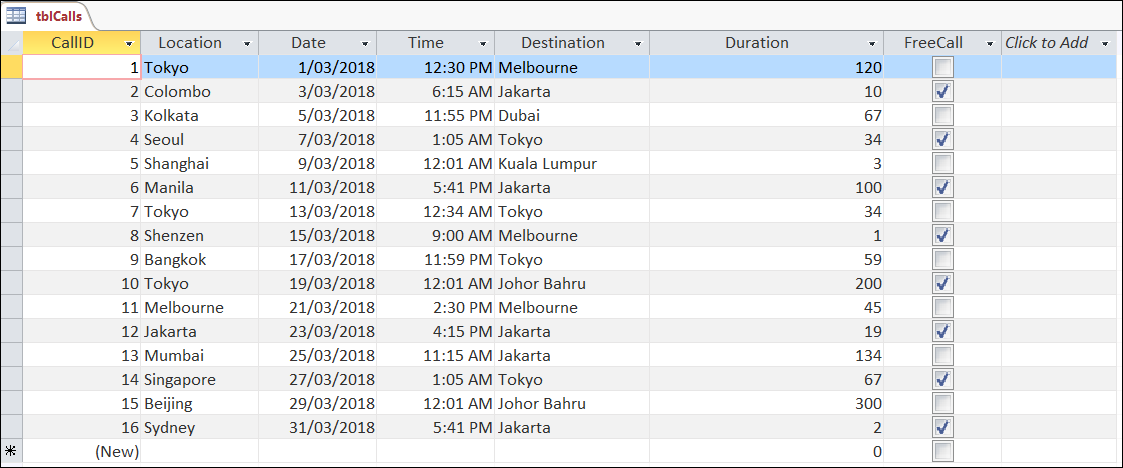
Forms are used to add and view data, tables are used to store the data.

**Database form**



The Calls table now contains 16 records (rows of data).

**Database table**



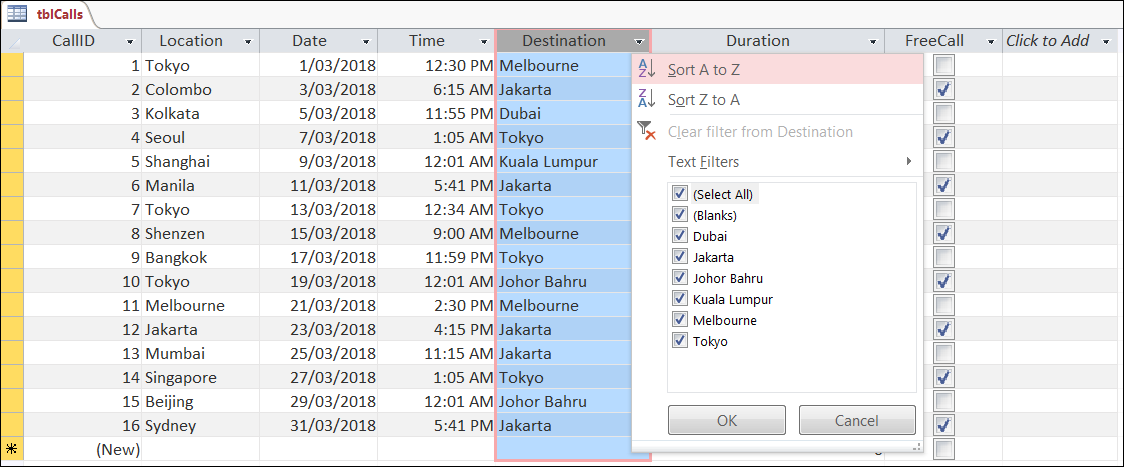
**Filtering and sorting data**

Data contained in a table can be filtered and sorted whilst in Datasheet View.

**Sorting data**

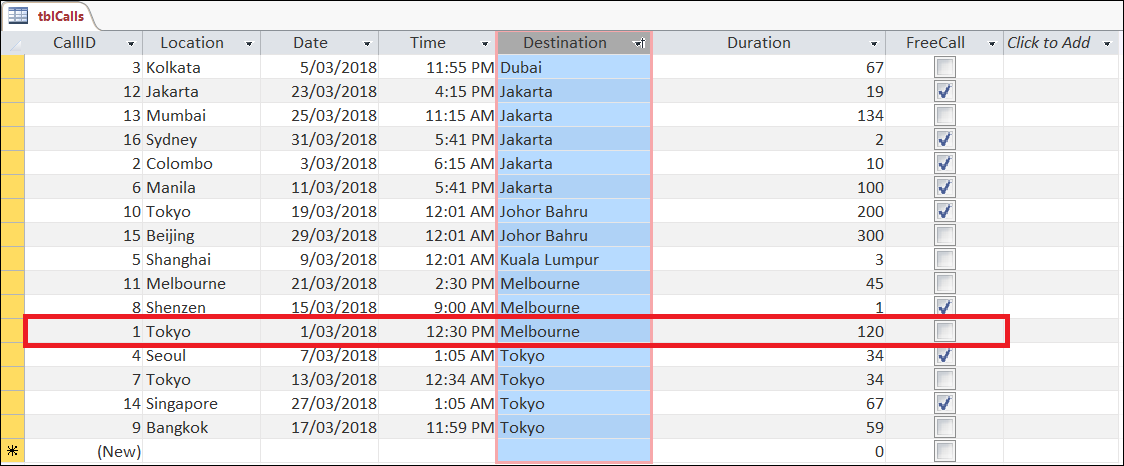
Sorting data involves placing each record (or row) in order based upon the values in one field (or column).

**1. Click** on the down arrow (circled red) next to the label Destination.



**2. Sort** the data from A to Z.

The image below shows the data after the sorted was executed.



All the destinations have been sorted in to alphabetical (or ascending) order, from A to Z.

In addition, each field (column) has also been sorted, based on the value in the Destination column, so the data in each record (row) is still the same, although records (rows) are shown in a different order.

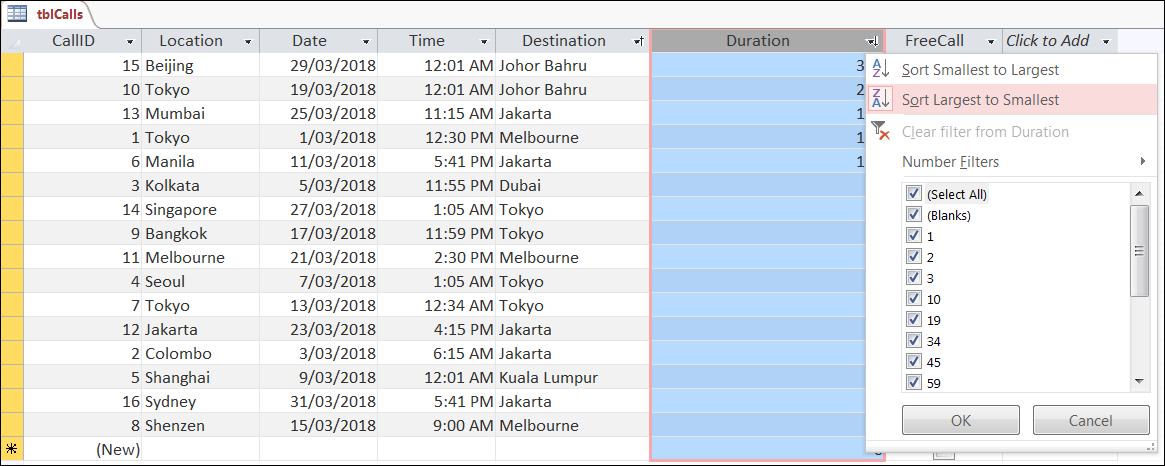
In the example highlighted, all of the data related to CallID 1 is still together in the same record (row), but the record (row) has been placed in alphabetical order based upon the value in the Destination column.

**Data types and sorting**

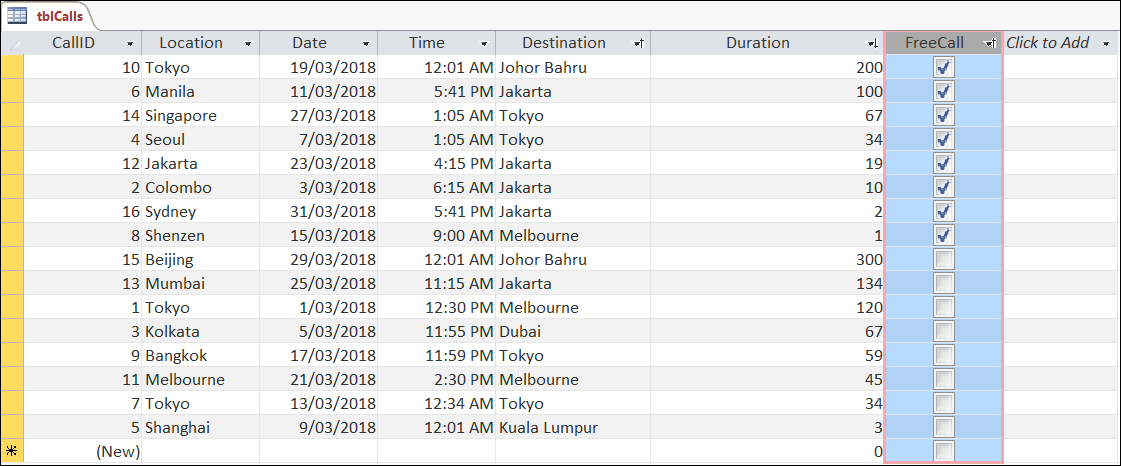
The method used to sort data is based on data type of that field (or column) e.g. short text, number, date/time, etc.

* Short text: A to Z (or Z to A)
* Number: lowest to highest (or highest to lowest)
* Date/Time: oldest to newest (or newest to oldest)
* Yes/No: selected to cleared (or cleared and selected)

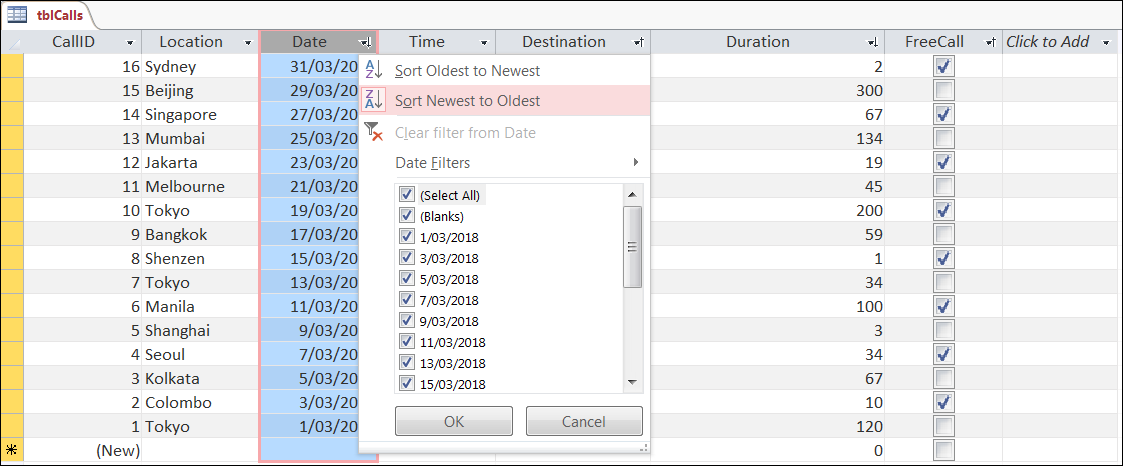
**3. Sort** the data in the table by Duration from largest to smallest.



**4. Sort** the data in the table by FreeCall? from selected to cleared.



**5. Sort** the data in the table by Date from newest to oldest.

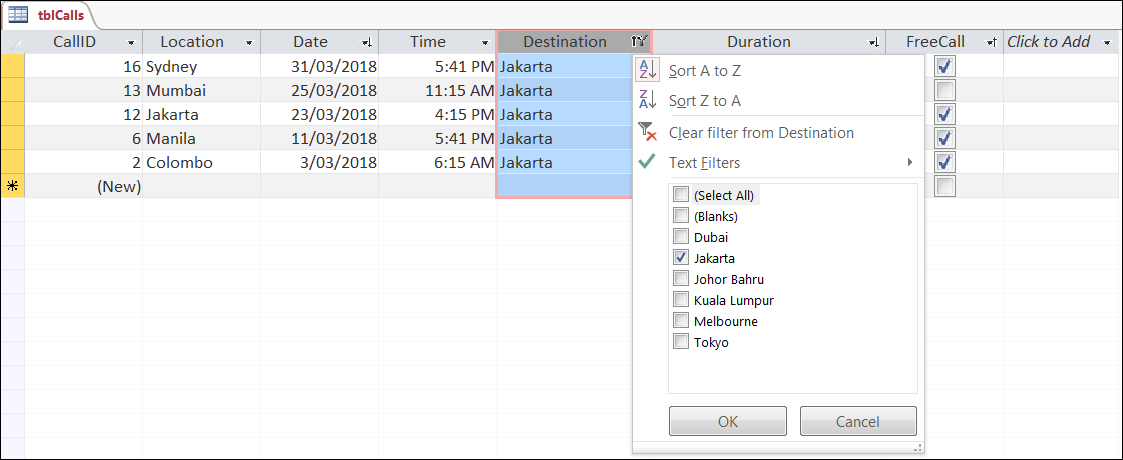


**Filtering**

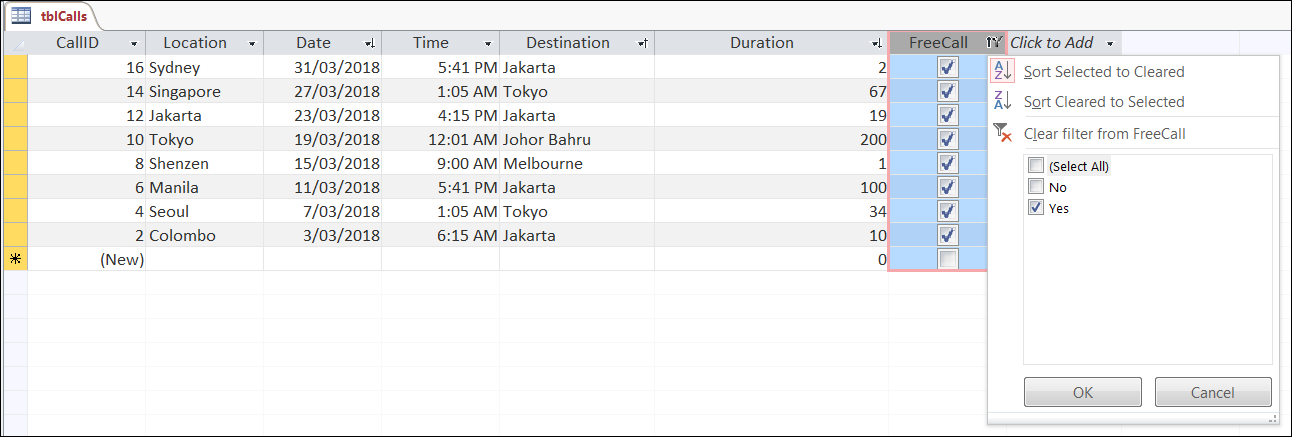
Filters are used to display only data that meets certain criteria or rules.

Only records (rows) that meet the criteria are displayed in the table.

**6. Filter** the data in the table to show only records where the destination was Jakarta.



**7. Filter** the data in the table to show only records where FreeCall is yes.



**Queries**

A query is a question.

A database query allows a user to ask a question that will retrieve particular data from a DBMS.

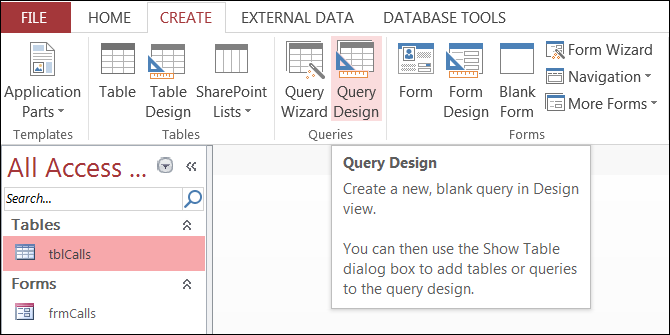
There are a few methods that can be used to create queries.

One method uses structured query language (or SQL for short) to retrieve data from a table.

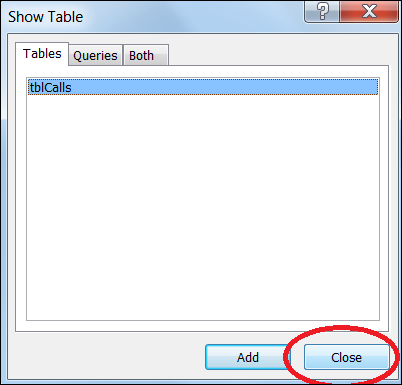
Let’s make a query using SQL.

**Note:** Close the Calls table first.

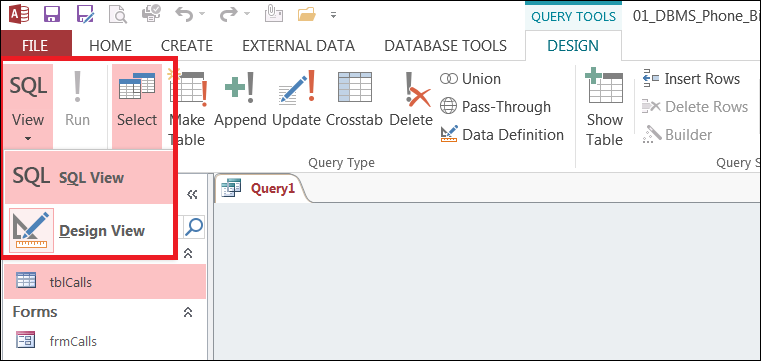
**8. Click** Create > Query Design.



**9. Close** the Show Table window.

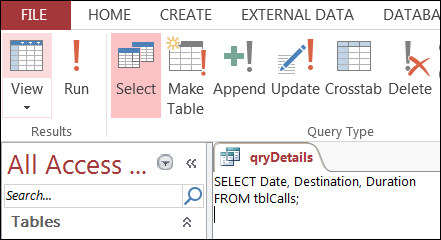


**10. Click** View > SQL View.

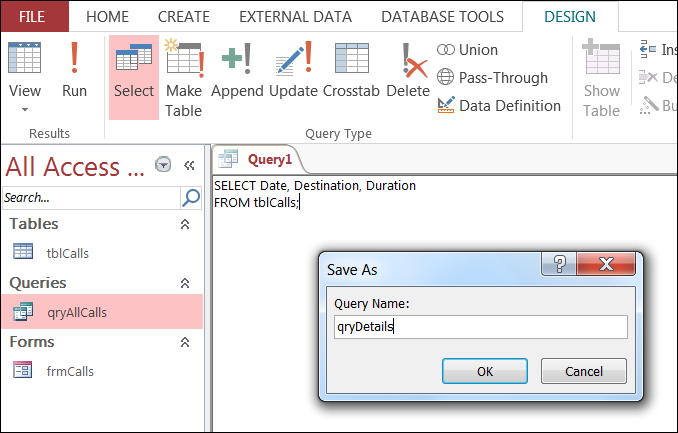


**11. Type** the query below in to the SQL window:

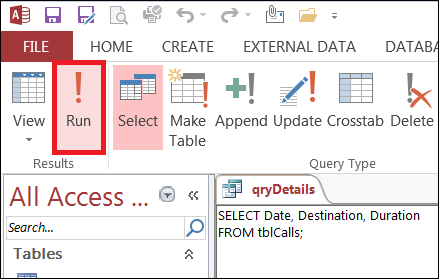
**SELECT** Date, Destination, Duration  
 **FROM** tblCalls;



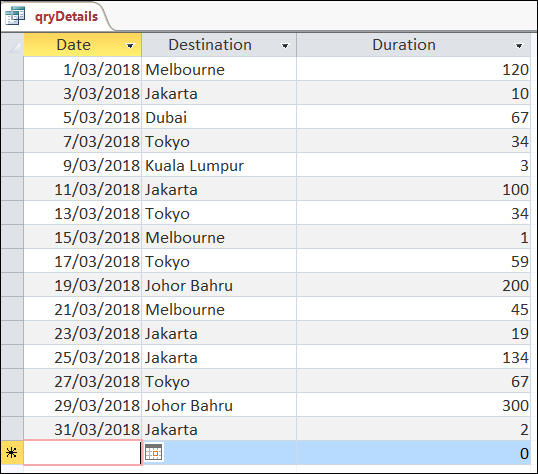
**12. Save** the query as qryDetails.



**13. Click** Run (highlighted red).



The results of the query should now be visible.



The SQL query contained two commands:

**SELECT**  The name of the fields (or columns) that need to be retrieved.

**FROM**  The name of the table where the fields are located.

In the qryDetails query Date, Destination and Duration were retrieved from the database.

These fields were located in the table tblCalls.

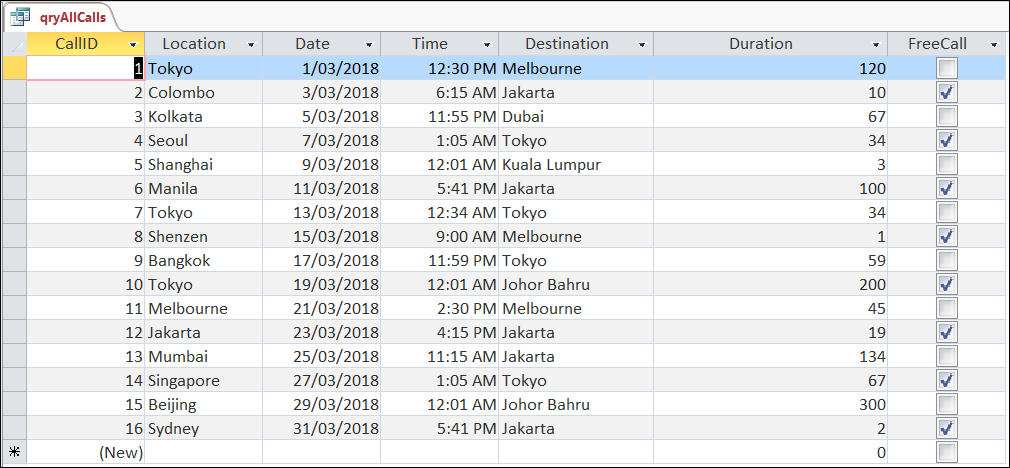
**14. Create** a new SQL query and type the following:

**SELECT** \*  
**FROM** tblCalls

**Note:** The \* symbol is used to retrieve ALL of the fields from the table.

**15. Save** the query as qryAllCalls, then **run** the query.

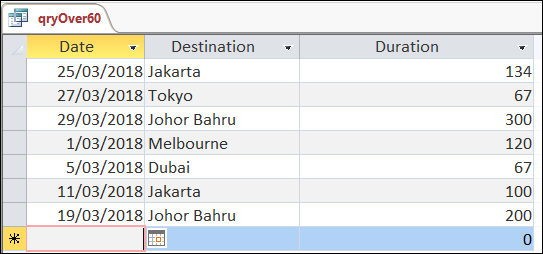
The expected results are shown below.



**16. Create** a new SQL query and **type** in the following:

**SELECT** Date, Destination, Duration  
 **FROM** tblCalls  
 **WHERE** Duration >60

**17. Save** the query as qryOver60, then **run** the query.

The expected results are shown below:  
  


The **WHERE** command sets a condition or criteria for the data that is required to be retrieved.

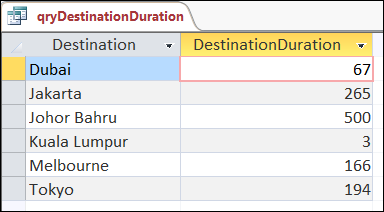
In this query only records (rows) where the destination is greater than 60 are displayed.

**18. Create** a new SQL query and type in the following:

**SELECT** Destination, SUM(Duration) As DestinationDuration  
**FROM** tblCalls  
**GROUP BY** Destination;

**19. Save** the query as qryDestinationDuration and **run** the query.

The expected results are shown below:



The query is retrieving data from the Destination and Duration fields (or columns).

A function, or mathematical equation, (SUM) is being applied to the data in the Duration field. The value of each duration is being added (SUM) together to calculate the total duration.

This calculated value is then stored in a new column called DestinationDuration.

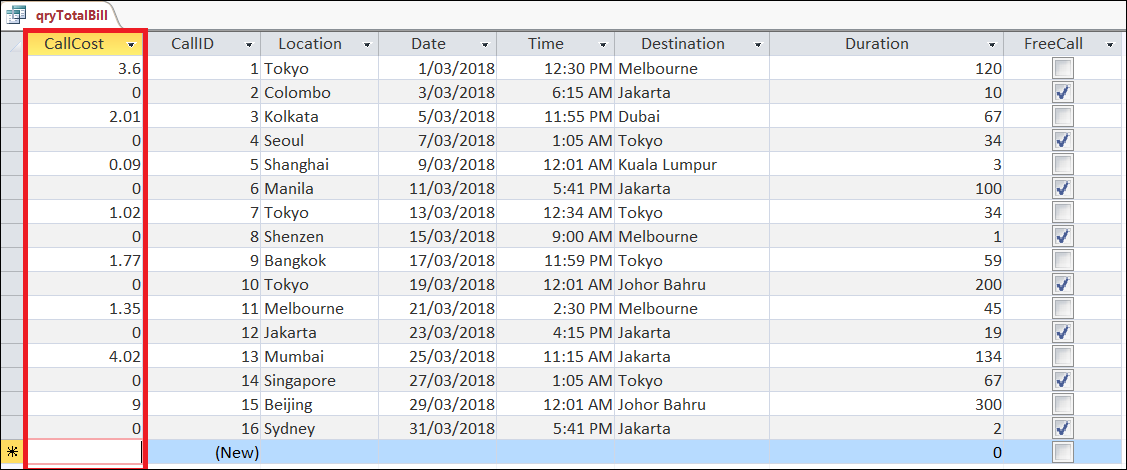
The GROUP BY command is then used to display the DestinationDuration for each destination.

**20. Create** a new SQL query and type in the following:

**SELECT** \*, IIf([FreeCall] = Yes, 0, Duration\* 0.03) AS CallCost  
**FROM** tblCalls;

**21. Save** the query as TotalBill and **run** the query.

The expected results are shown below.



The query has retrieved all of the fields (columns) from the tblCalls table.

A calculation has also been performed to create a new column CallCost.

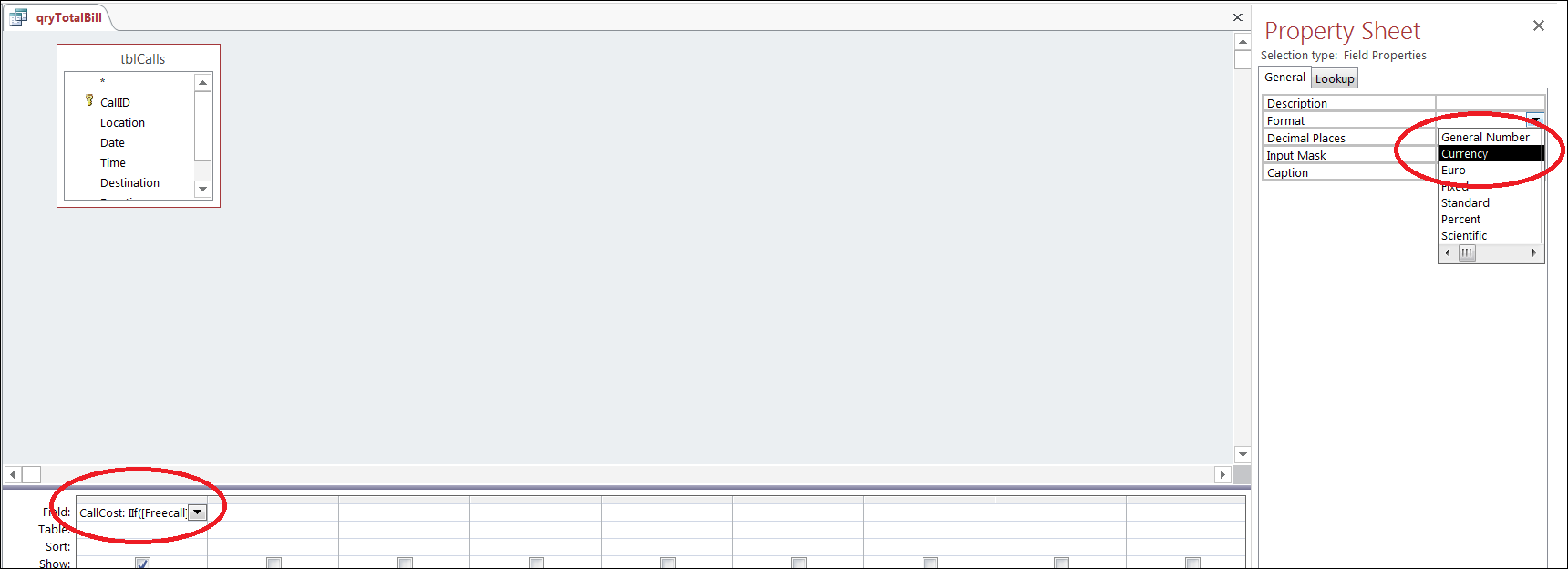
The calculation involved the use of an IIF function. An IIF function is used to perform one of two tasks.

In this situation if FreeCall = Yes, then the CallCost = 0, if FreeCall was not Yes (e.g. it was No) then the CallCost = Duration \* $0.03.

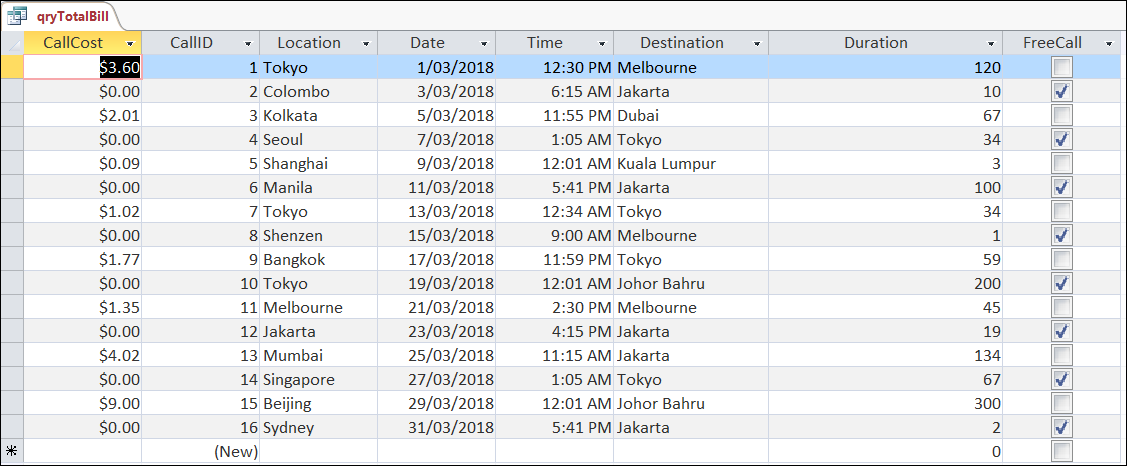
The CallCost column needs to be formatted so it displays the data as a currency data type.

Also the order of the columns needs to be rearranged.

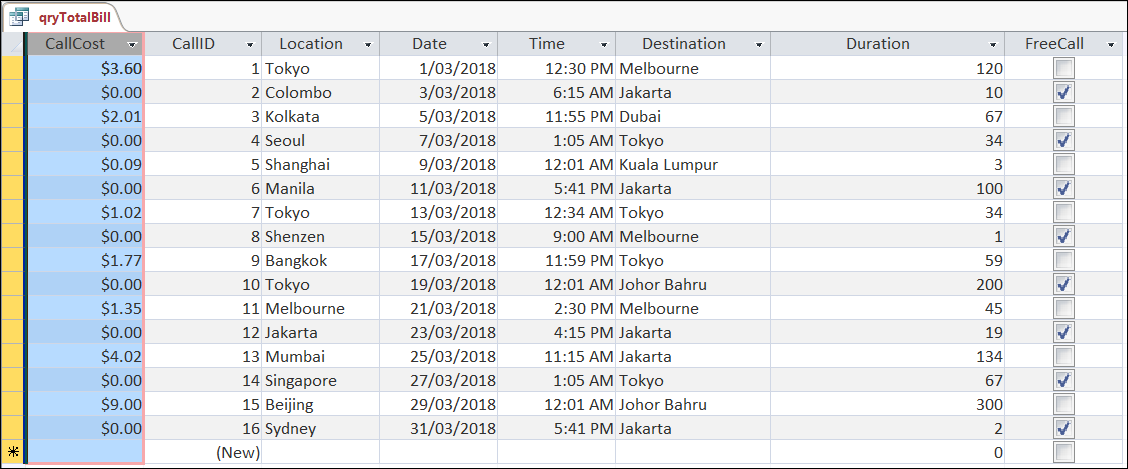
**22. Click** Design View (of the query) > **Select** the CallCost field then **select** Currency from the Format drop down list.



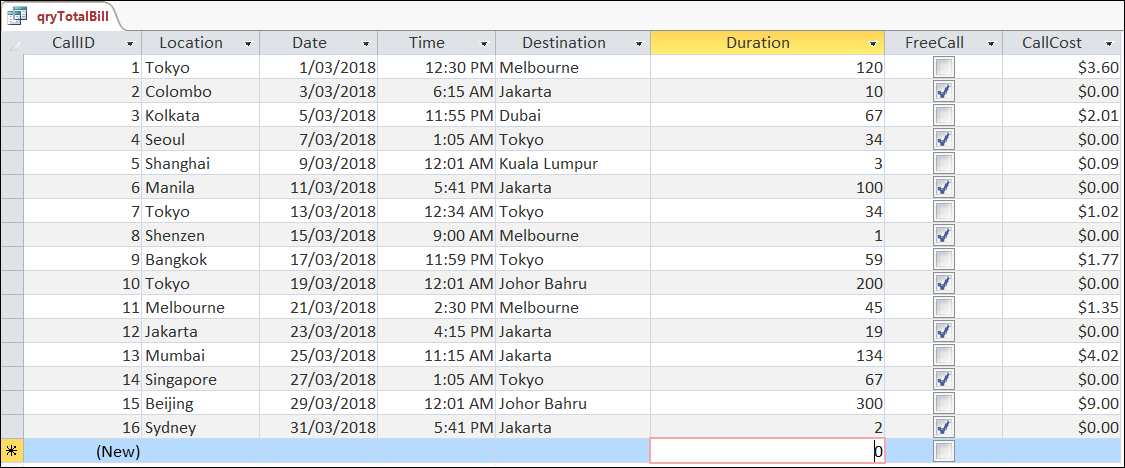
The CallCost column then appears with a $ sign and each value has two decimal places.



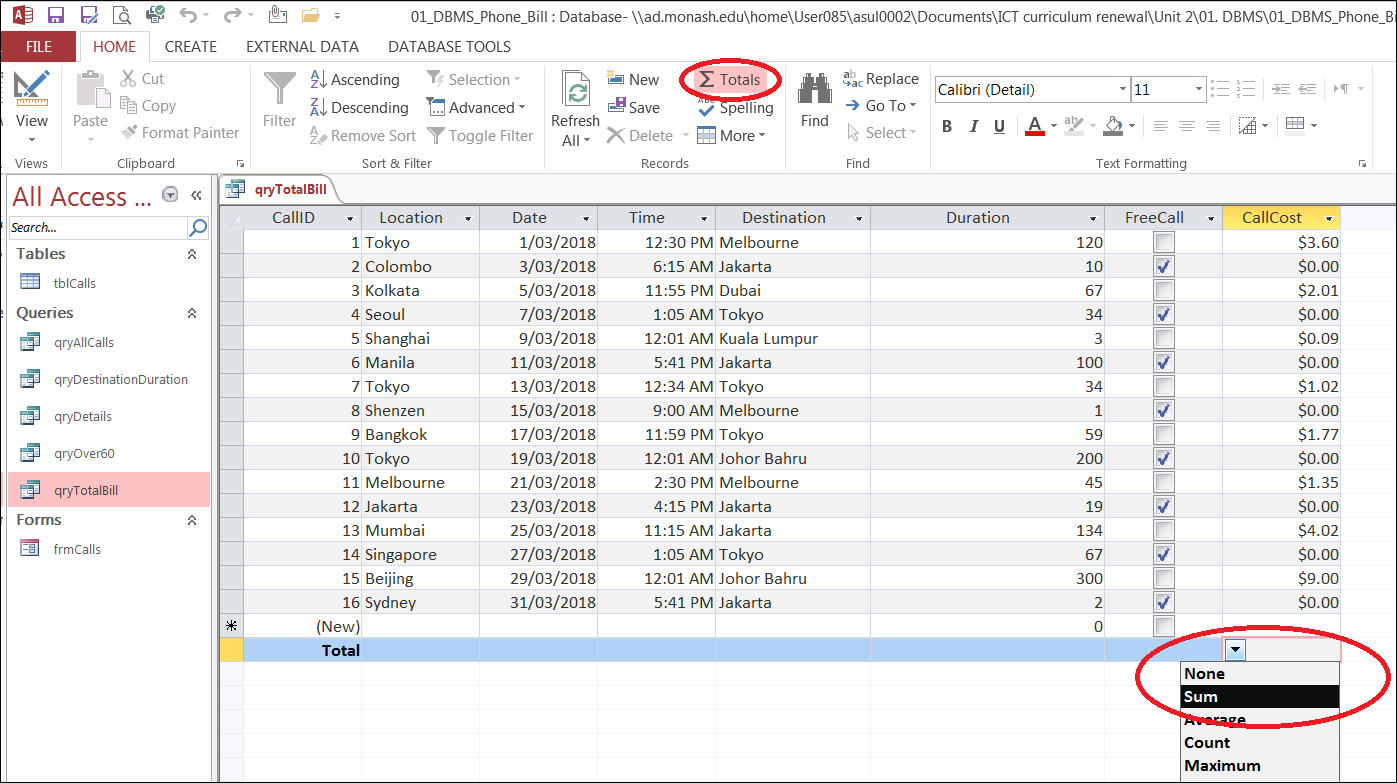
**23. Select** and **drag** the CallCost column so it appears last in the query results.



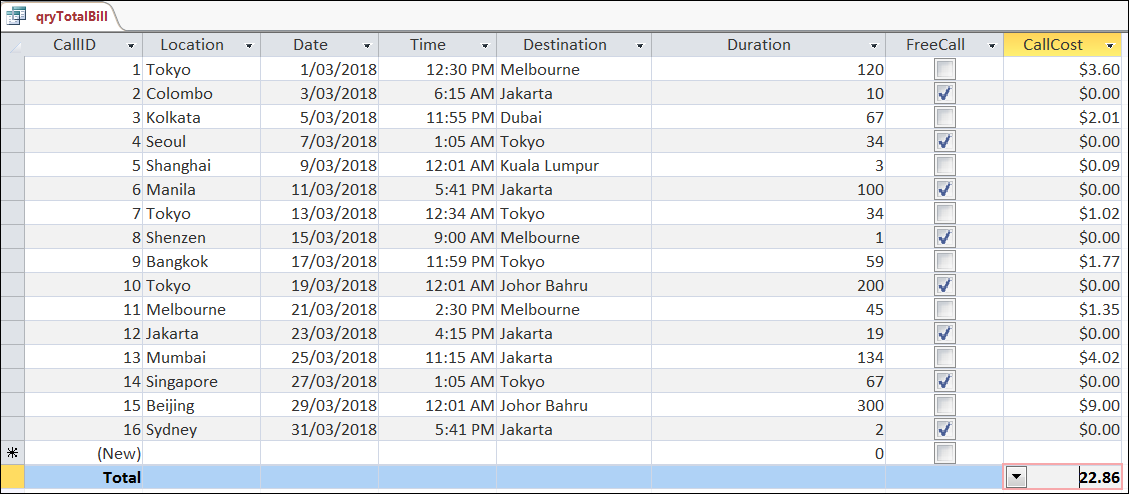
The order of the columns in the query should now appear in the following order.



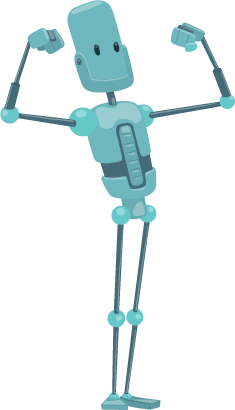
**24. Click** the Totals icon on the Home tag, then **select** SUM from the drop down list.



The total of all of the CallCost is now displayed.



**25. Save** the query.

You have now finished this activity.

In this activity you have:

* Sorted data in data grid view
* Filtered data in data grid view.
* Retrieved data using SQL queries
* Formatted the appearance of data within a query

Well done!

**Note:**

You will need the PhoneBill DBMS file for the next activity. Save it in a safe place.