**Institute of Technology Tralee**
Institiúid Teicneolaíochta Trá Lí

# Database Concepts
## Lab 01 – The Relational Database

### In this lab you will become familiar with:

- The basic features of the Relational Data Model (Using MS Access)
- Primary Key / Foreign Key Constraints
- Querying the Database

### Before you start:

All material relating to this module (notes, lab sheets, database files, scripts, etc) should be organised such that you can readily access it.

- Create a folder called DBConcepts (No spaces!) on your X:\Student drive.

- Within this folder, create the following sub-folders
  - Lectures
  - Labs
  - Databases
  - Scripts

  You may add more folders, if and when required, during the semester.

### Getting Started:

- A sample MS Access database called ***StockSys*** has been created for you to use during this lab. The file extension is not shown (StockSys.accdb)

- You can find this database at the following location:

  **X:\Lab\C Woods\DBConcepts_CP\Databases\StockSys**

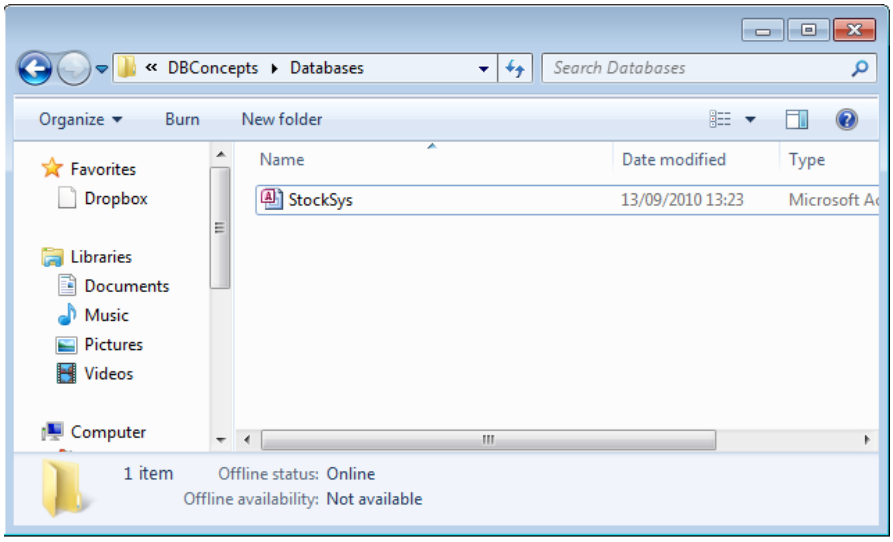
- Copy this file to the appropriate folder on your X: drive

  **X:\.....\DBConcepts\Databases**

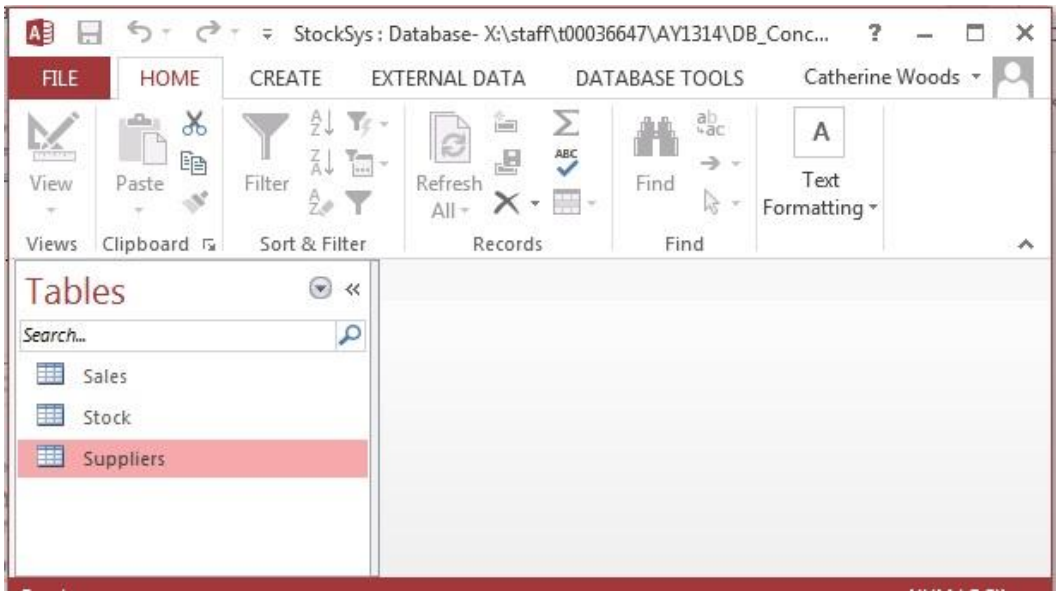
## Open the sample Database

Using ***Computer Folder***, locate the sample database on your X\Student drive.

Double click on the file ***StockSys***(.accdb)




The following window will then be displayed.

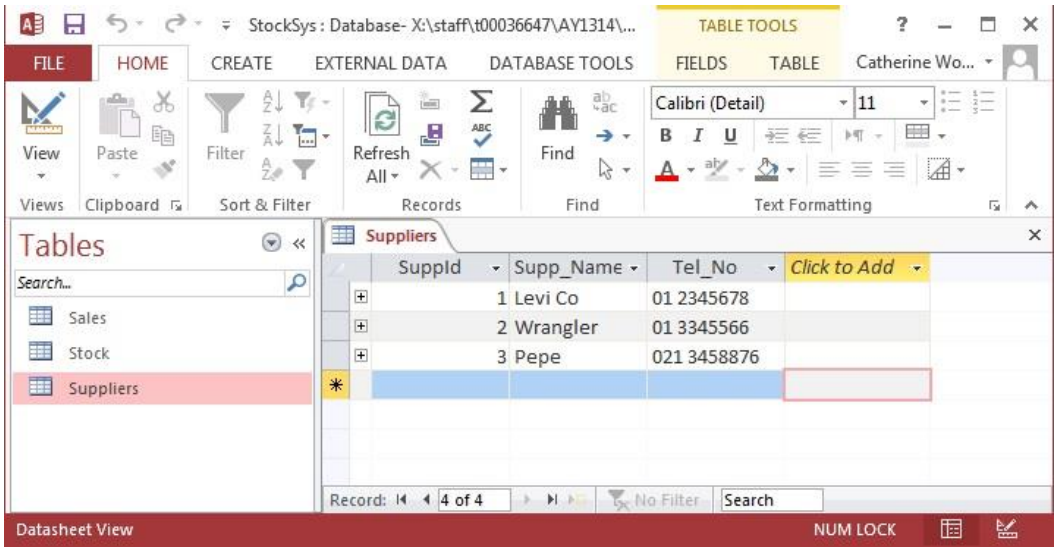


## Relational Database Tables

You can see that the ***StockSys*** database contains three relations(tables): *Suppliers*, *Stock* and *Sales*.
To open a table and view the data in the table, simply double-click on the table name. The contents of the table will be shown in ***Datasheet*** format.

Open the ***Suppliers*** table.

A table consists of rows (tuples) and columns (attributes) of data.




The value of an attribute is taken from the ***domain*** of values allowed for that attribute. For example, a supplier name may be in the domain of alphabetical strings of maximum length 50. A supplierId is all numeric values in the range 1 – 999.

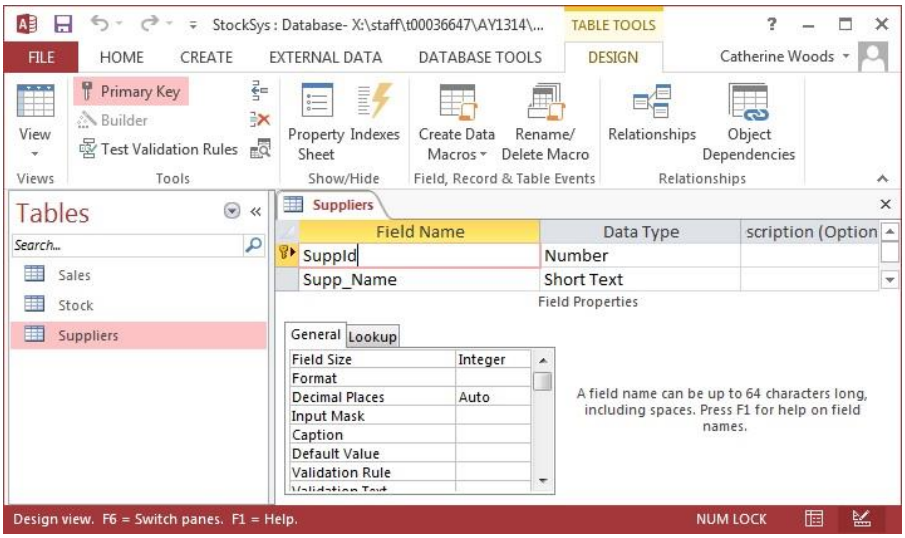
The value of an attribute for a given value of the primary key must be ***atomic*** (one and only one value).

A table may be viewed in one of two ways: ***datasheet view*** or ***design view***.

Select **Home → View** to toggle between the two views.


The datasheet view allows you to see the actual data in the table (shown above).

The design view allows you to see the data type and size of each of the attributes (columns) in the table (shown below).



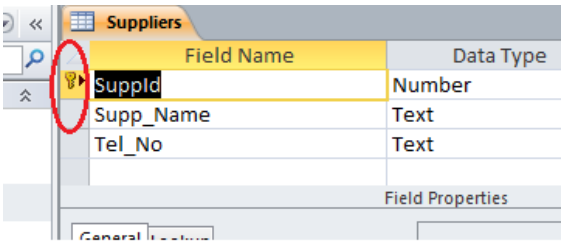

The design view also indicates the ***primary key*** attribute(s) for the table. Notice the 'key' icon on the left hand side of the field name.

## Data Integrity

The relational data model enforces two integrity rules – entity integrity and referential integrity.

### Entity Integrity

***Entity integrity*** requires that each entity (record, row) in a table (relation) is unique. ***Primary keys*** are used to enforce entity integrity. A primary key is defined as ***one or more attributes of a table which uniquely identifies each record in the table***. A primary key may not be NULL and must be unique for each entity (record, row). In the table ***Suppliers*** shown below, the single attribute **SuppId** is the primary key (see symbol).




Some tables may have a ***composite primary key*** i.e. a primary key composed of more than one attribute.

StudentSubjects

| Field Name | Data Type |
|---|---|
| StudentId | Short Text |
| SubjectId | Short Text |
| Grade | Number |

Primary key attributes should, as a rule of thumb, be listed first. Every relation (table) in a relational database **must** have a primary key.

## Referential Integrity

Relationships may exist between tables in a database. Such a relationship is referred to as a ***parent-child*** relationship. The *primary key* of the parent table exists as a *foreign key* in the child (or related) table. These relationships are used to enforce ***referential Integrity***.

***Referential integrity*** defines a relationship between two (or more) tables. ***Foreign keys*** are used to implement referential integrity. A foreign key is defined as ***an attribute of a table which exists as a primary key in a related table***. A foreign key may be NULL and does not have to be unique.

Consider the table ***Stock*** shown below. Stock is supplied by a supplier.

Stock

| Stock_No | Description | Cost_Price | Sale_Price | Qty | Supp_Id |
|---|---|---|---|---|---|
| 1 | Levis 501 | €50.00 | €75.00 | 100 | 1 |
| 2 | Wrangler Regula | €40.00 | €65.00 | 50 | 2 |
| 3 | Levis 901 | €80.00 | €105.00 | 35 | 1 |
| 4 | Pepe Skinny | €45.00 | €70.00 | 20 | 3 |

The ***Supp_Id*** attribute tells us which supplier supplies each stock item. Supp_Id is therefore a ***foreign key*** in the table *Stock*. The value of the Supp_Id attribute must exist as a primary key value in the related table ***Suppliers***. Foreign keys must be explicitly defined.

Suppliers

| SuppId | Supp_Name | Tel_No |
|---|---|---|
| 1 | Levi Co | 01 2345678 |
| 2 | Wrangler | 01 3345566 |
| 3 | Pepe | 021 3458876 |

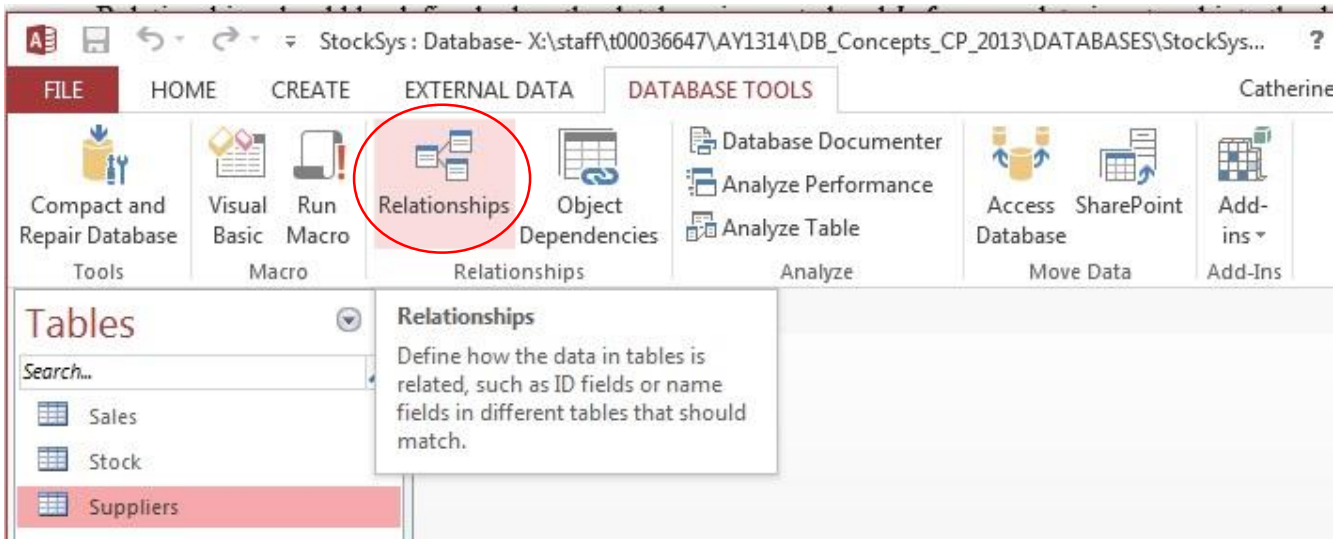
Inserting a row in the table Stock with a Supp_Id other than 1, 2 or 3 will cause referential integrity to be violated and will not be allowed by the DBMS.

## Table Relationships

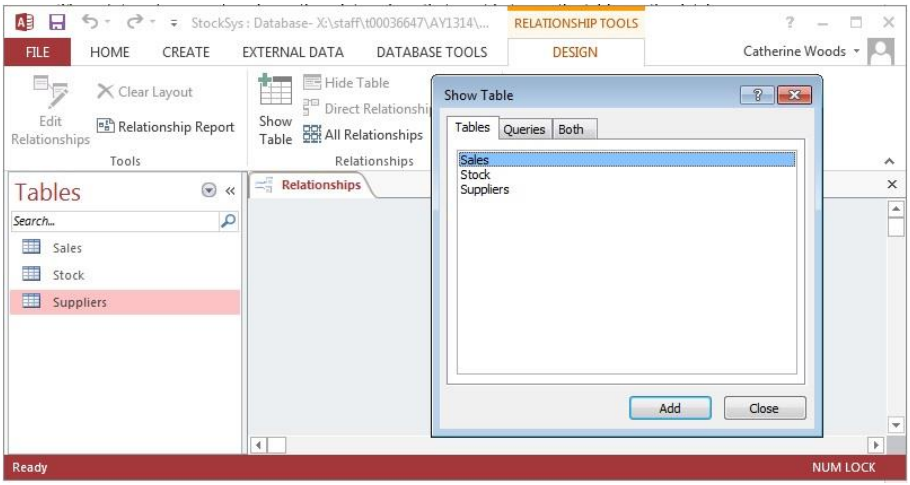
Take a few minutes to familiarise yourself with the data and the structure of the tables in StockSYS by using the ***Design***/***Datasheet*** views of the tables.

Relationships should be defined when the database is created and ***before*** any data is entered into the database tables.

Ensure that all of the tables in the ***StockSys*** database are closed.
Click on the **Database Tools** tab and then the **Relationships** icon.



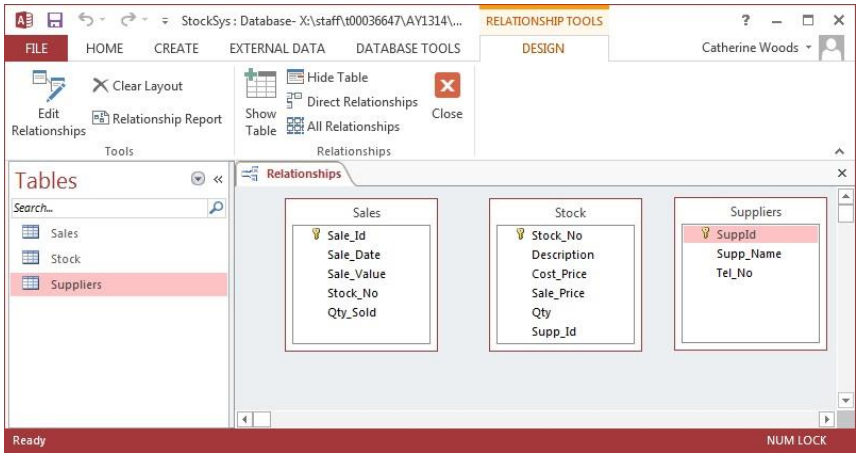

The *relationships window* shows the relationships that exist between the tables in the database.
If no relationships have yet been defined, click the ***Show Table*** icon.



We can easily identify two obvious relationships that should exist in our example:

- *Stock* is supplied by a *supplier*
- *Stock* is sold in a *sale*

Add the tables between which a relationship exists. In our case, all three tables must be added.




Consider the relationship ***Stock is supplied by a Supplier***.

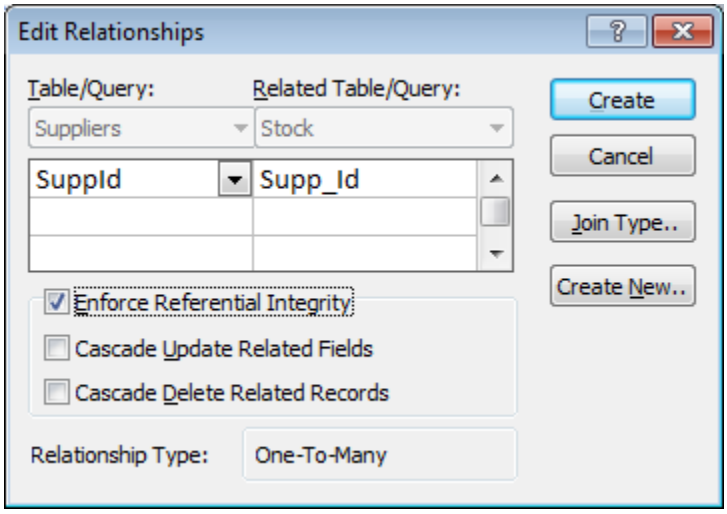
- A Supplier is identified by *SuppId* {PK} in the *Suppliers* table (Parent table)

- For a relationship to exist between the Suppliers table and the Stock table, a ***common attribute*** must exist in each table. A common attribute requires that the attribute be the same data type and size in both tables but does not require the same attribute name.

- The attribute *Supp_Id* exists in the ***Stock*** table and the attribute SuppId exists in the ***Suppliers*** table. These attributes are of the same data type and therefore a common attribute exists. The table *Suppliers* is the parent table (PK is SuppId) and the table *Stock* is the child table (FK is Supp_Id). Note the difference in the name of the common attribute.

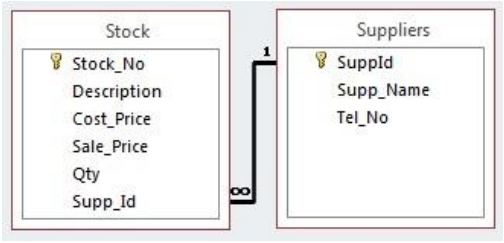
- To create the relationship between the two tables simply click on *SuppId* in the parent table (Suppliers) and drag-and-drop it on *Supp_Id* in the child table (Stock).

- The *Edit Relationships* dialogue box is then displayed (as shown on the following page).




Check the ***Enforce Referential Integrity*** check box and then click *Create*.

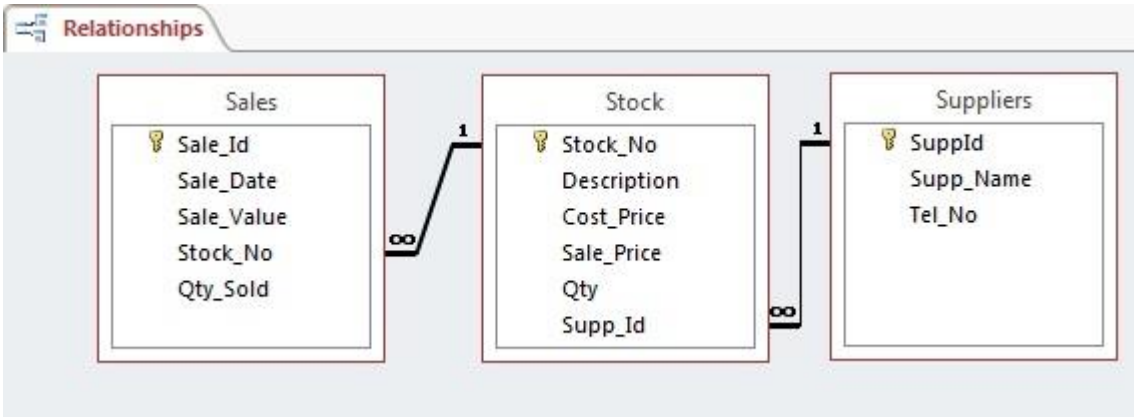
- The relationship is shown in the relationships window as a 'Join' line between the two tables.




The ***cardinality*** of the relationship is also shown. In this instance the relationship is a one-to-many (1:M) relationship. A SuppId can exist only once in the parent table (Suppliers) but may exist many times in the child table (Stock). Relationship cardinality may be 1:1; 1:M; M:N.

- Create the relationship between the *Sales* and the *Stock* tables.

The relationship window should then look like the following:



- Close the Relationship window and save the changes you have made.

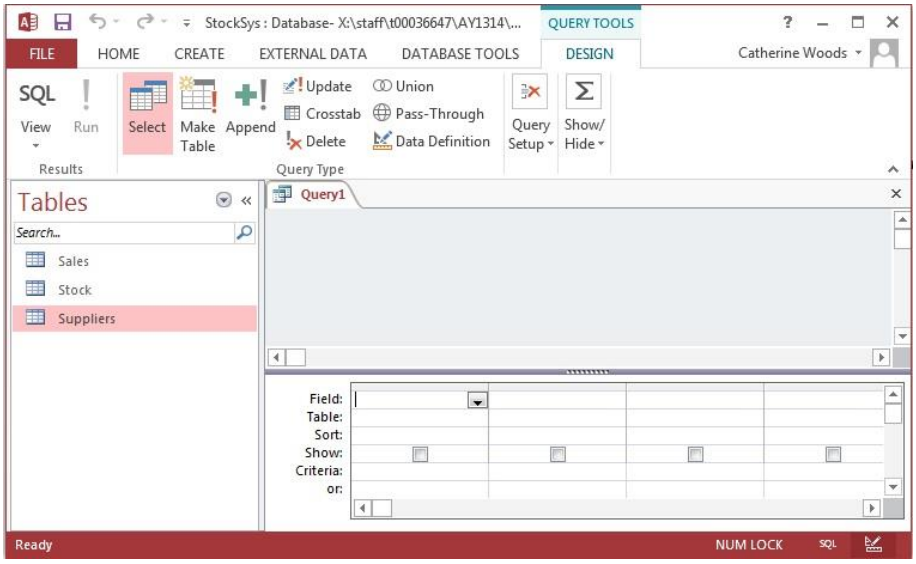
## Querying the Database

The data in a database provides us with information. Using our sample, we may wish to know:

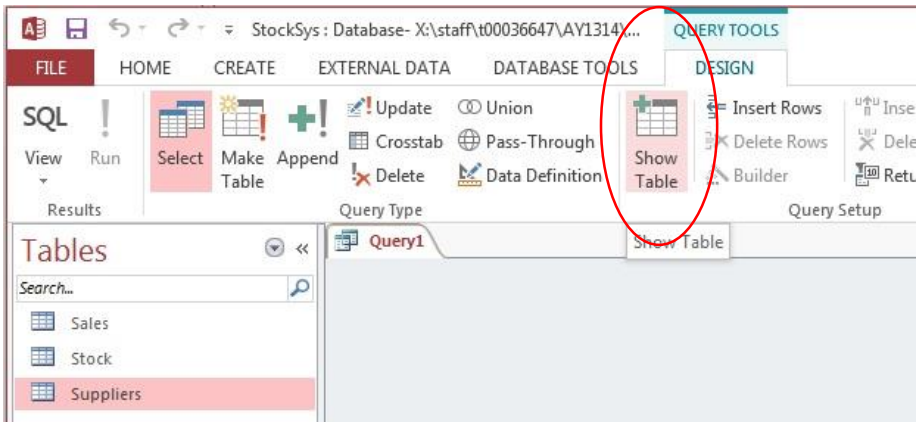
- What stock the company sells
- What suppliers the company deals with
- The cost of a pair of levis 501 jeans
- The sales figures for a given month

To find the answers to these questions we need to ***query*** the database.

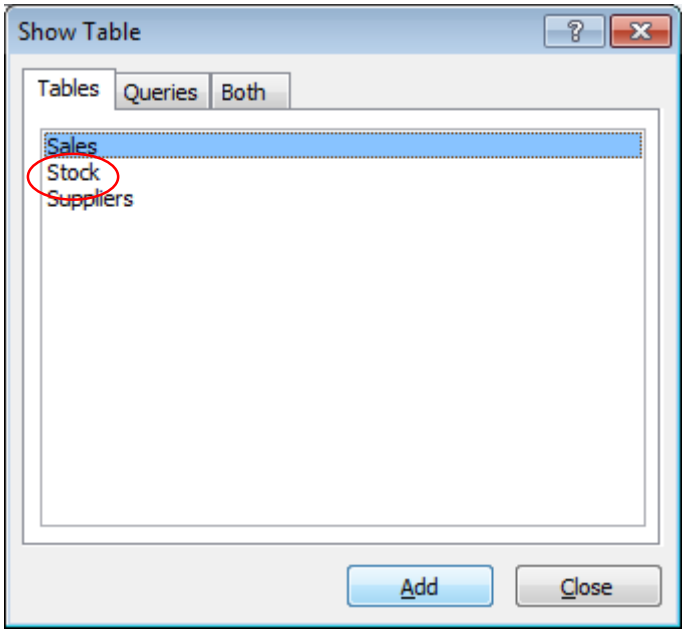
Suppose we wish to see all stock which the company sells. Select the ***Create*** tab and then ***Query Design***. The Query By Example (QBE) grid is presented.



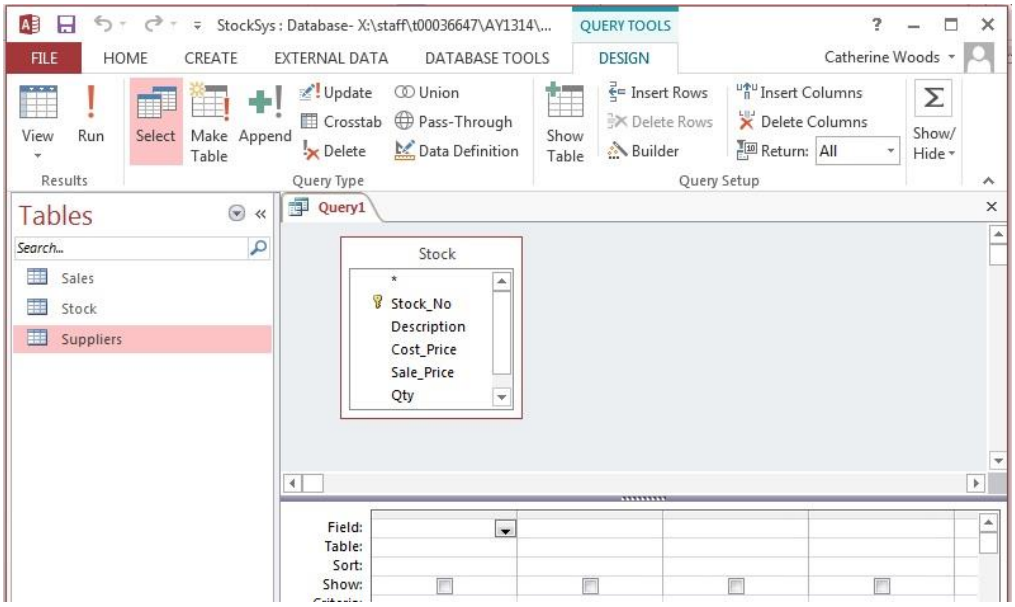

Next, you must add the table(s) which you wish to query. To do this, click on the ***Show Table*** icon.



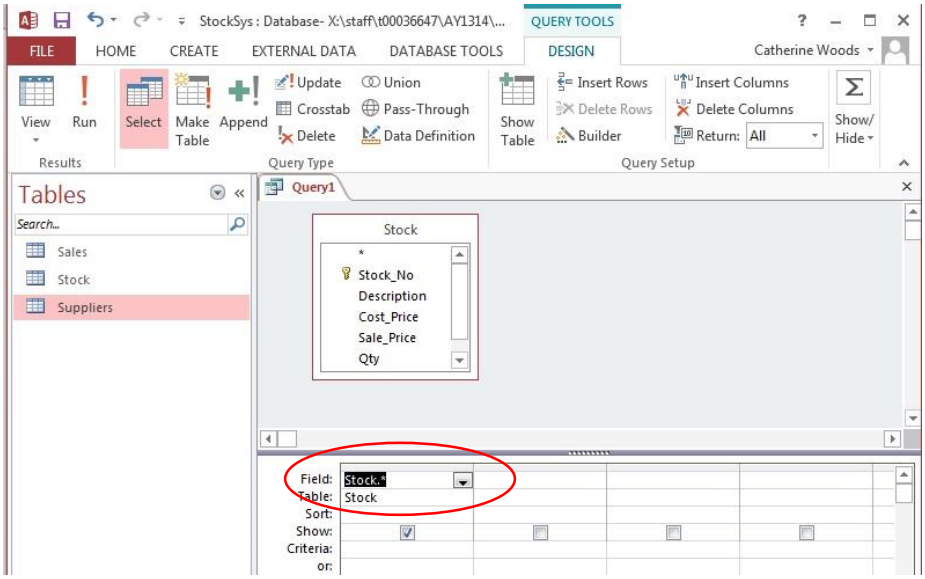
Select ***Stock*** in the Show Table window and add it to the query design.




Close the Show Table window. The query design now looks as follows:

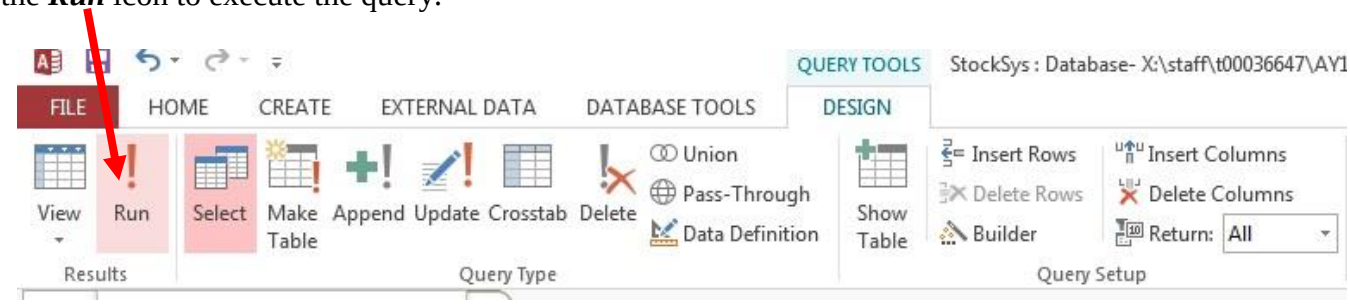



Click on the arrow in the ***Field*** element of the QBE grid. Select **Stock.*** (or double click the * item in the table)

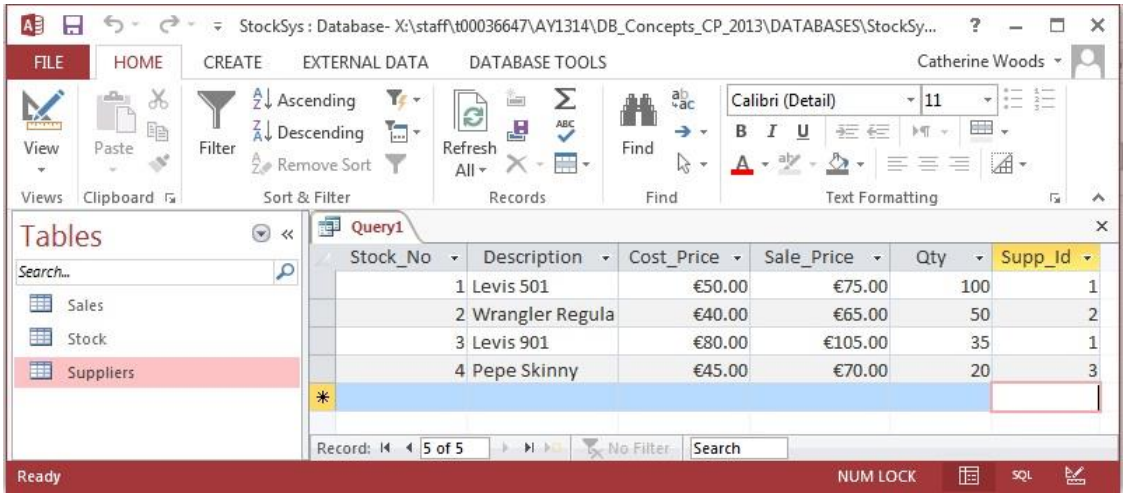



You can also drag-and-drop the attributes from the table(s) into the ***Field*** element of the QBE grid.

Click the ***Run*** icon to execute the query.

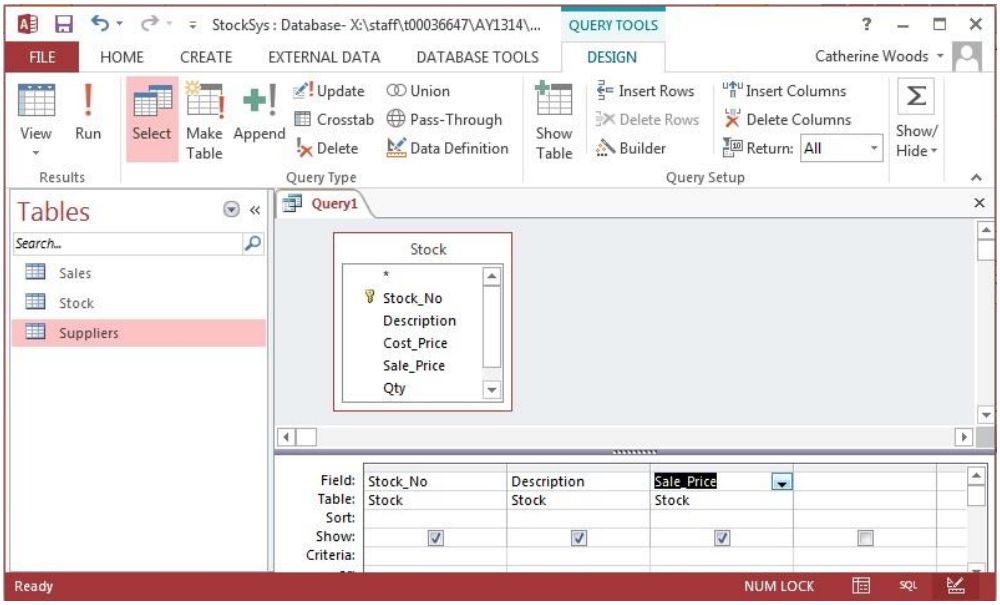



Notice that the results of the query (result set) are displayed in the very same way as the datasheet view of a table.

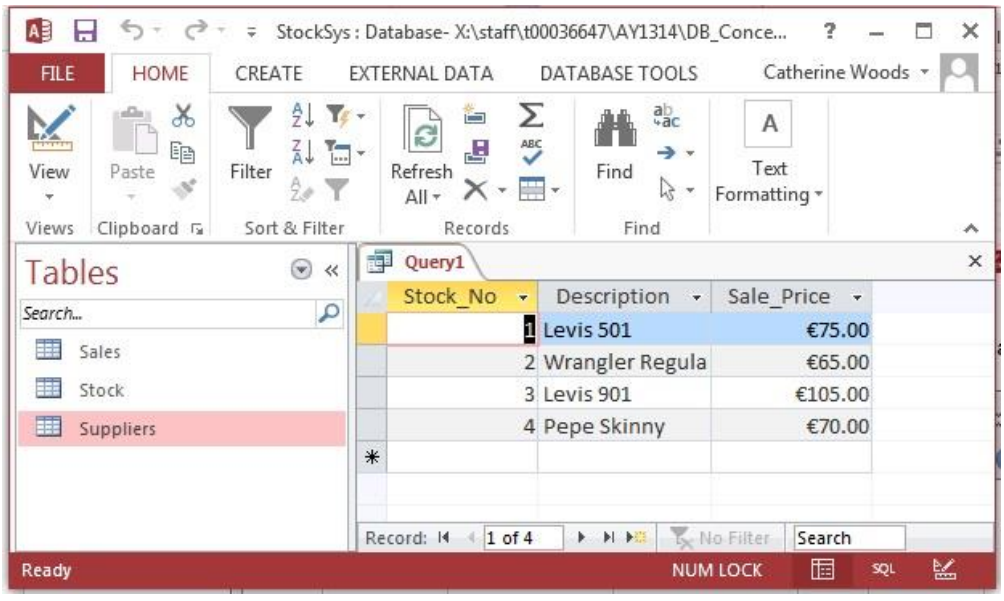


Click on the ***Design*** icon to return to design view.

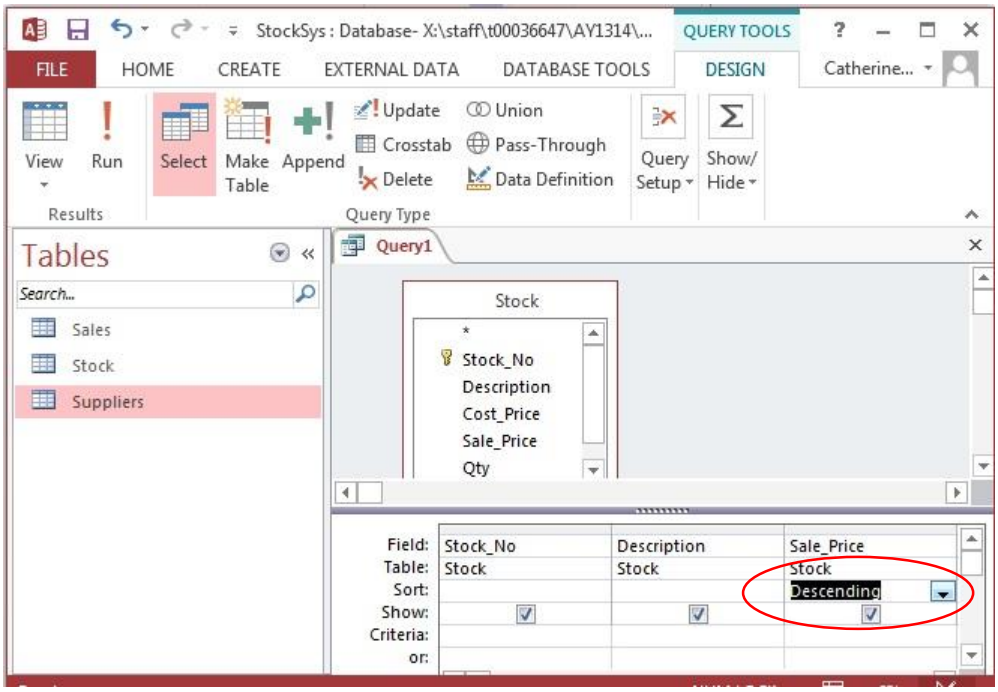
Suppose we wish to see only the Stock_No, Description and Sale_Price of all the stock. To do this, we would place each of the required attributes in the ***Field*** section of the QBE grid and check the ***show*** checkbox:



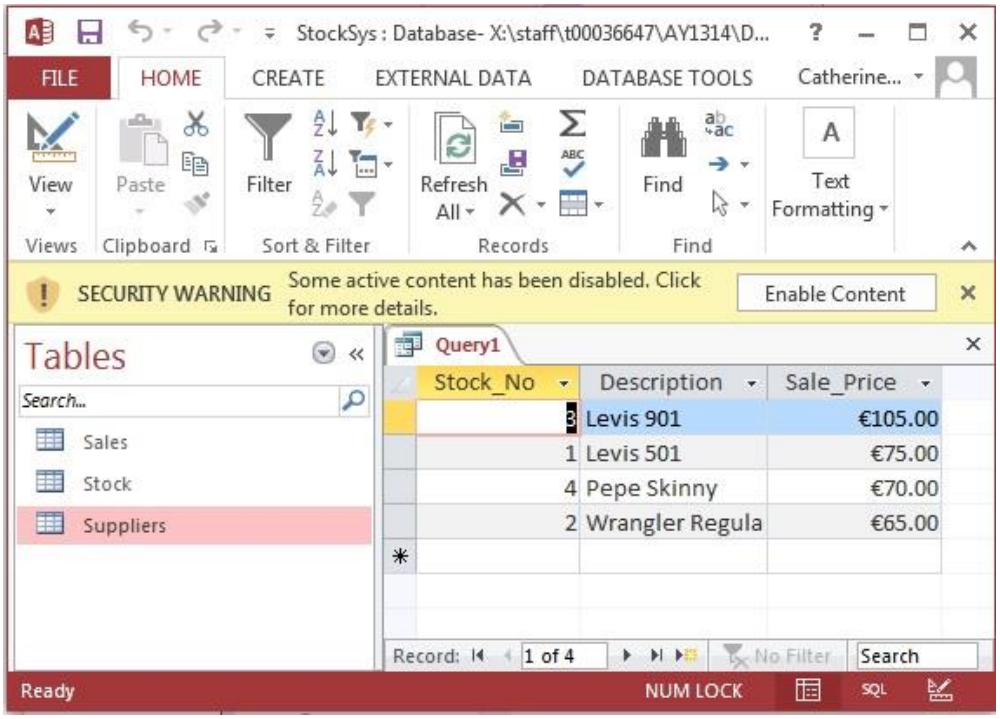

When we run the query, a different result is obtained. Only the selected attributes appear in the result.



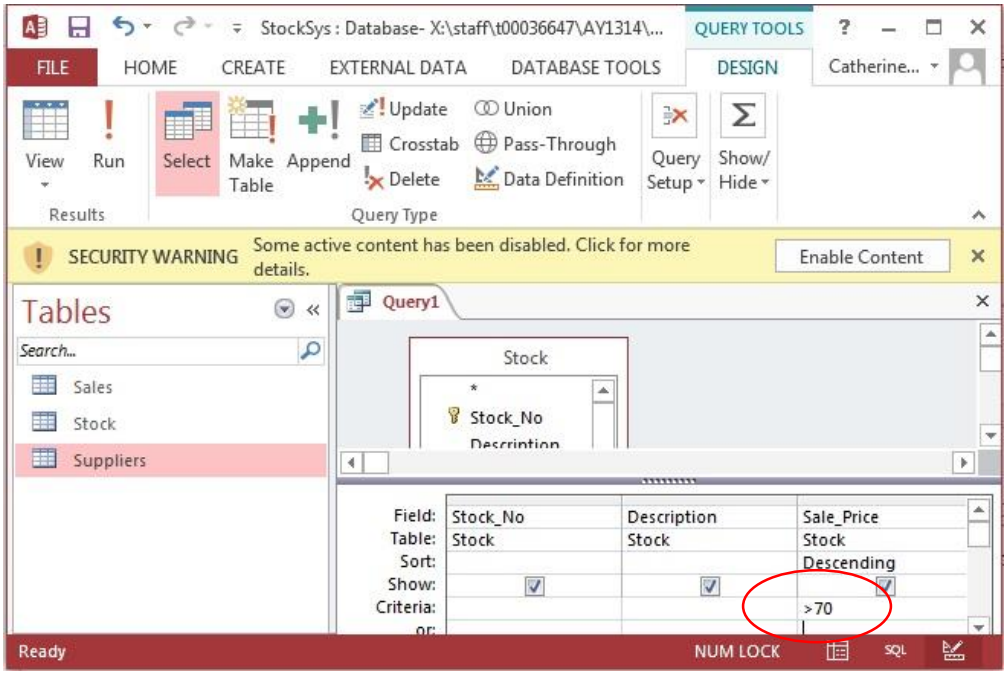
Suppose we wish to view the stock in descending order of sale price i.e. the most expensive stock first. Simply set the ***Sort*** element for the attribute Sale_Price to ***Descending***



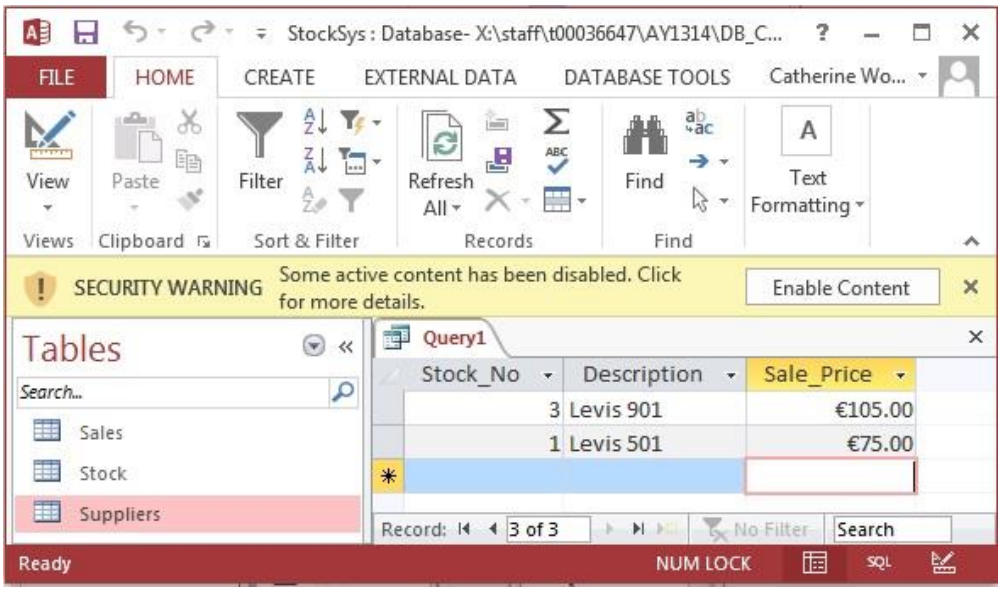

When we run the query, a different result is obtained:



Other filters can be applied when querying the database. For example, suppose we wish to see only the stock whose Sale_Price exceeds €70. This ***criteria*** must be specified in the query design.




The result of this query looks as follows:

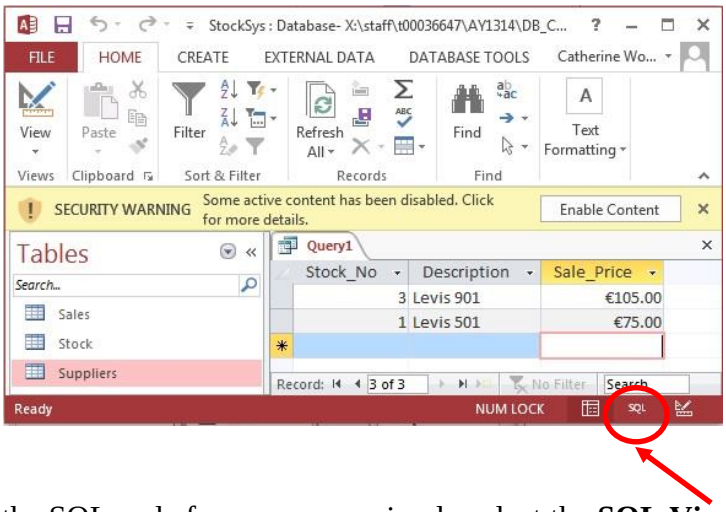


## Querying the Database using SQL

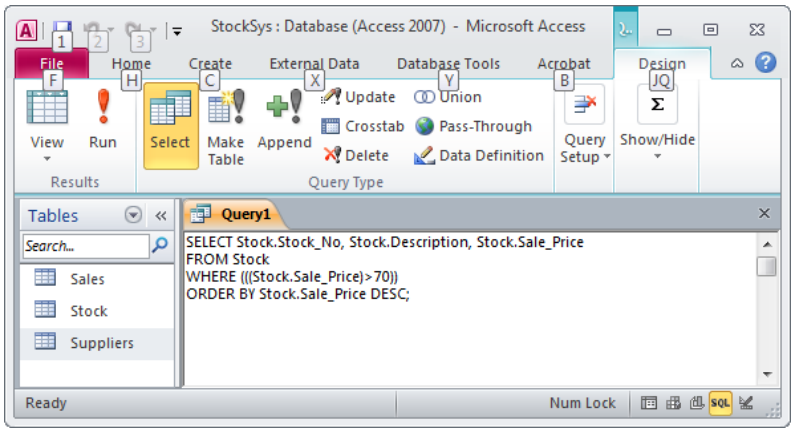
The QBE grid is useful for MS Access users or for users who may not have technical or programming skills. Since the advent of the relational database in 1970, many relational database management systems (RDBMS) have been developed and are used extensively in industry. These include:

- Oracle
- Ingres
- dBase
- MS SQL Server
- MySQL

Since 1986, **data manipulation** and **data definition** has been standardised through the use of Structured Query Language (SQL). SQL acts as an interface between application software and the underlying relational database, regardless of the database management system being used. Experienced programmers or database users use SQL to define and/or query a database. Queries created using the MS Access QBE grid are translated into SQL by the DBMS.




To see the SQL code for any query, simply select the **SQL View** option at the bottom of the window. The SQL view for the last query looks as follows:




For the remainder of this module, you will learn the syntax of the SQL programming language and will write SQL queries to create a relational database, to modify a relational database definition and to manipulate the data in a relational database. You will use both MS Access and Oracle sqldeveloper/SQL*Plus to do this.

## Examinable Material

The material presented in this lab is examinable in the final examination (written).
Students should be able to discuss/describe the features of the relational data model.

This requires students to define, explain and illustrate using examples the following terms:

- Relation, tuple, attribute (Table, row, column)
- Atomic values
- Attribute domain
- Integrity Rules
    - Entity Integrity
    - Referential Integrity
- Primary Key
- Foreign Key
- Relationship
- Relationship Cardinality (one-to-one, one-to-many, etc)
- Database Query/Result set