Introduction to databases with MySQL and PHPMyAdmin

In this section

In this section 1
Databases 1
MySQL 2
Why are we using it 2
phpMyAdmin 2
How can you access it 2
Accessing the data 5
The SELECT statement 5
Activity 1. 6
Activity 2. 6
Ordering the data (ORDER BY) 6
Activity 3. 6
The WHERE clause 7
Activity 4. 7
BETWEEN Operator 7
Activity 5. 7
IN Operator 7
Activity 6. 8
The LIKE Operator 8
Activity 7. 8
Suppressing duplicates (DISTINCT) 8
Activity 8. 8
Performing Math 8
Counting Records 9
Activity 9. 9
Using GROUP BY 9
Activity 10. 9
Multiple tables (table joins) 10
Activity 11. 11
Joining three or more tables 11
Activity 12. 11
The INSERT statement 12
The UPDATE statement 12
The DELETE statement 12
Activity 13. 12

Databases

There are many SQL databases around.

Oracle (www.oracle.com)

Informix (www.informix.com)

Sybase (www.sybase.com)

MySQL (www.mysql.com)

Postgres (www.ca.postgresql.org/index.html)

Access

Updated 6/2/2012

MySQL

The MySQL database server is the world's most popular open source database. Over six million installations use MySQL to power high-volume Web sites and other critical business systems — including industry-leaders like **The Associated Press, Yahoo, NASA, Sabre Holdings** and **Suzuki**.

MySQL is an attractive alternative to higher-cost, more complex database technology. Its **award-winning speed, scalability and reliability** make it the right choice for corporate IT departments, Web developers and packaged software vendors.

(Source: http://www.mysql.com/)

Why are we using it

PHP is often used in conjunction with MySQL.

phpMyAdmin

phpMyAdmin is a tool written in **PHP** intended to handle the administration of MySQL over the Web.

How can you access it

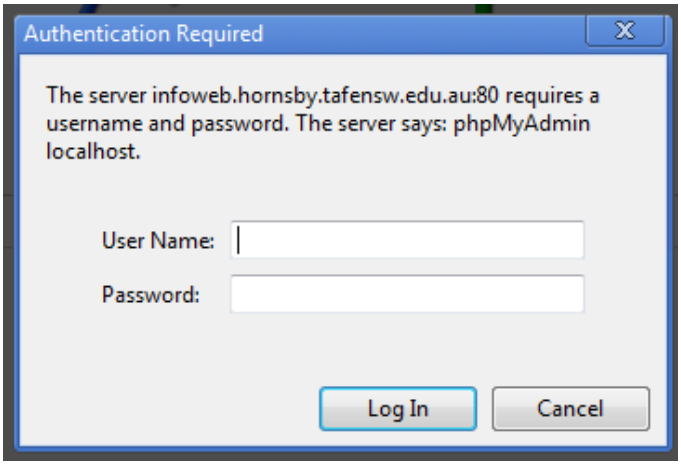
Please start your hands on activity here:

1.

Open up a browser and type the following URL in the address bar:

http://infoweb.hornsby.tafensw.edu.au/phpMyAdmin/

The following login screen will be displayed:

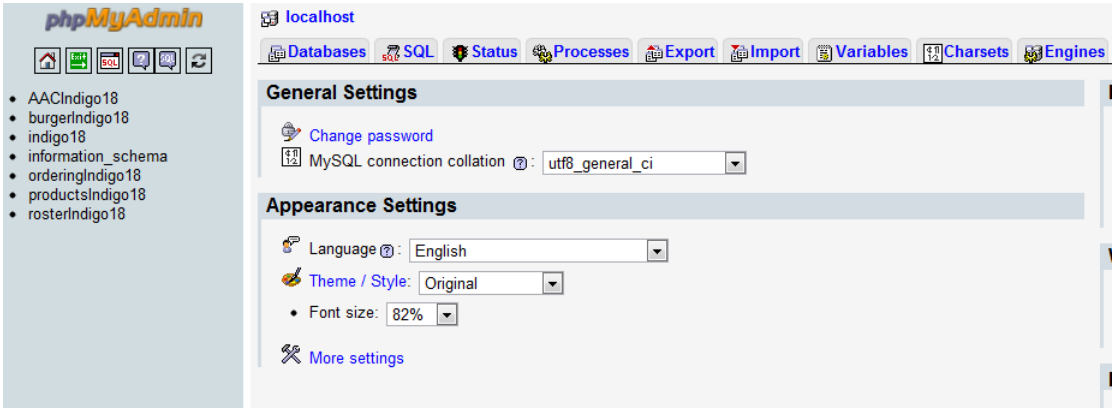


2.

Type in your username and password and click on OK.

3.

You will have the following screen displayed:



Your databases are empty, that is contain no tables at present. A dbName(x) shows how many tables it has in the brackets. So, dbName means '0' tables in that database.

4. **IMPORTING a DataBase file**

You will import the tables together with their records.

On the left hand pane select the **orderingXX** database

In the right hand pane select the **'Import'** tab

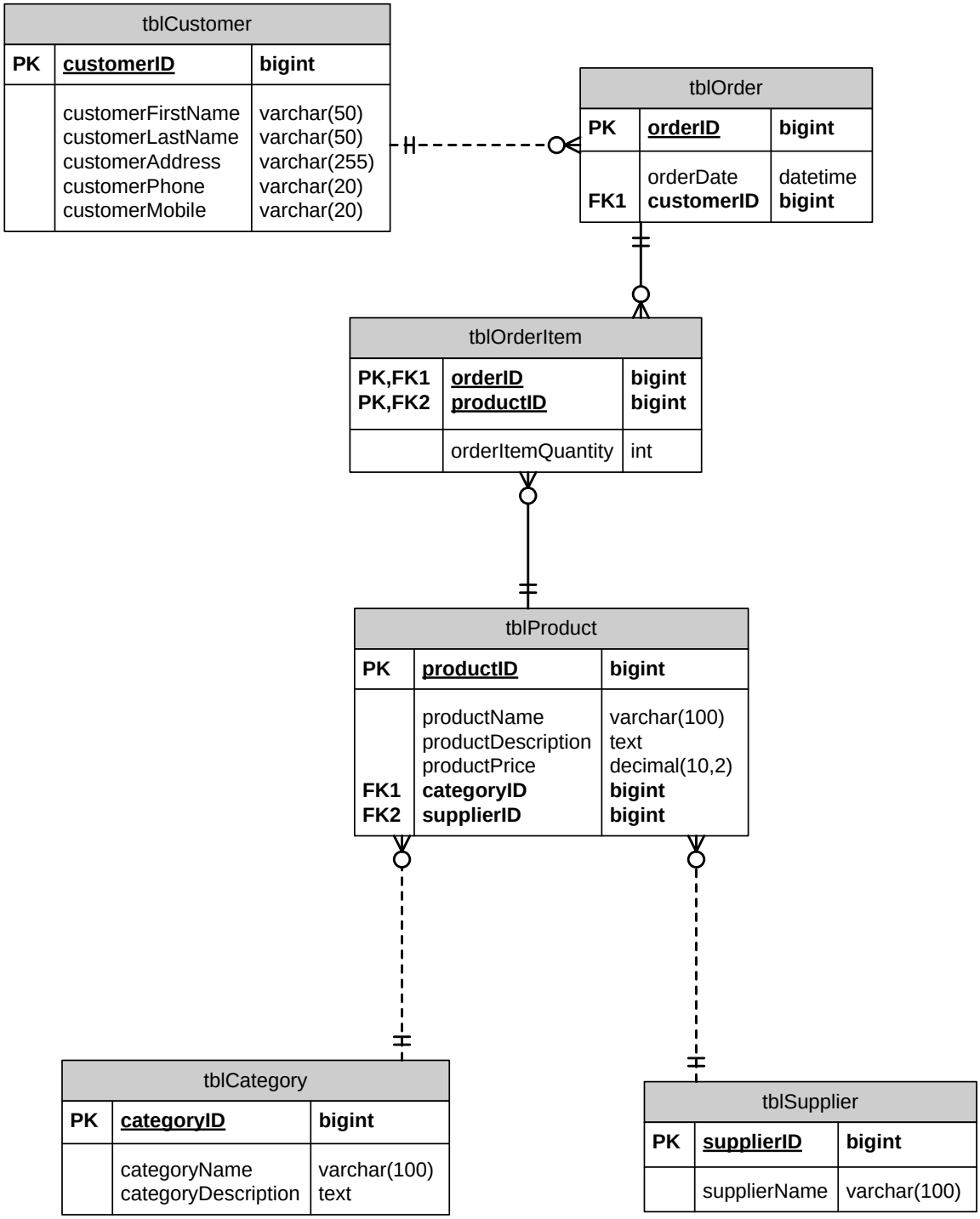
click the **'Browse'** button

find the file (**orderingDB.sql**) and select it

then click the **'Go'** button

You should see 6 tables in the left hand pane, and a message confirming success should be displayed on the right hand pane.

The ER diagram for the ordering database is on the following page.

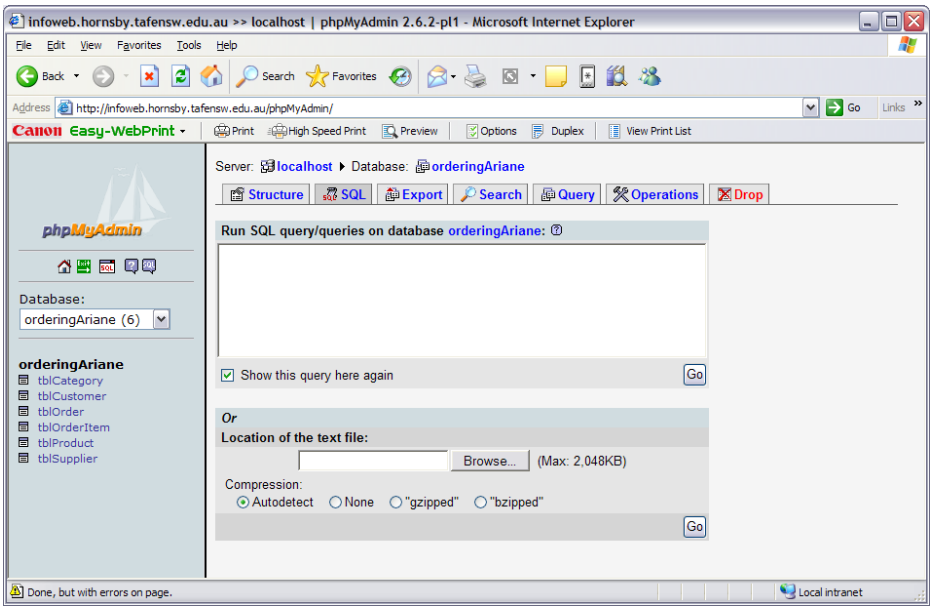


SQL

Structured Query Language (SQL) is the language spoken by many relational database management systems. It is the most popular and widely used database query language in the world.

Accessing the data

phpMyAdmin allows you to write SQL statements by selecting the SQL tab or icon on the left hand pane. The following screen will be displayed.



Inside the box you can type the SQL query then press the "Go" button to execute the code.

The SELECT statement

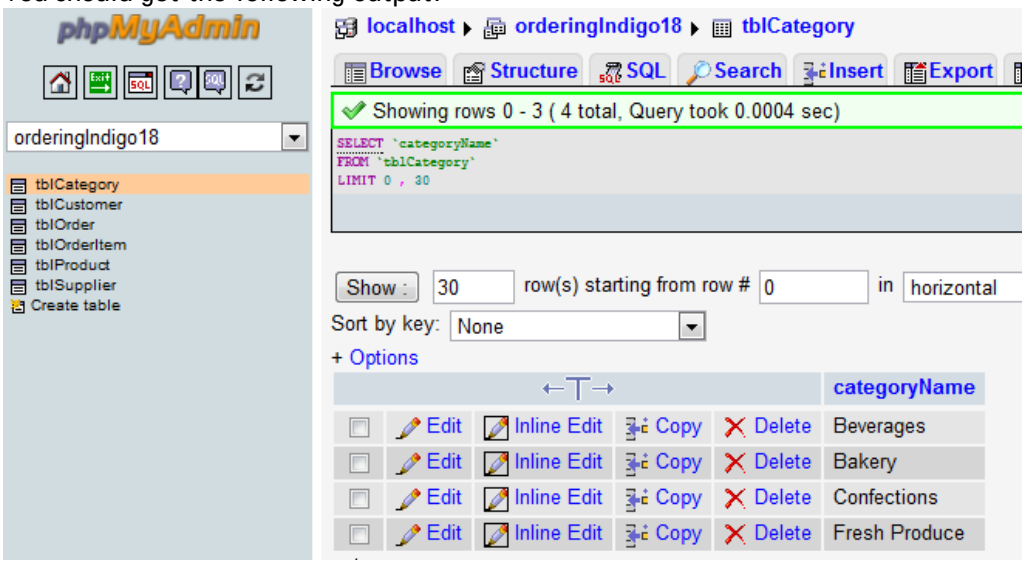
The syntax for the select statement is:

```
SELECT column_name
FROM table_name;
```

Example

```
SELECT categoryName
FROM tblCategory
```

You should get the following output:



Activity 1.

1. Select the first name from the tblCustomer table
2. Select the product name from the tblProduct table

If we want to get all the columns from a table, we can name all of them:

```
SELECT orderID, ProductID, orderItemQuantity
FROM tblOrderItem
```

Or you can use the * shortcut

```
SELECT *
FROM tblOrderItem
```

Activity 2.

1. Select the first and last name from the customer table
2. Select everything from the customer table
3. Select everything from the orderItem table
4. Select everything from the product table
5. Select everything from the supplier table

Ordering the data (ORDER BY)

The data we have been viewing is brought back to us in no particular order. It is often necessary to return the data in a particular order.

The syntax is as follows:

```
SELECT column_name1, column_name2, …..column_nameX
FROM table_name
ORDER BY column_name1
```

Example

```
SELECT customerFirstName, customerLastName
FROM tblCustomer
ORDER BY customerLastName
```

The default order for sorting is ASC (ascending). The above statement will sort the output by alphabetical order of the last name.

If we wanted the sorting to go the other way then the SQL statement could be changed to:

```
SELECT customerFirstName, customerLastName
FROM tblCustomer
ORDER BY customerLastName DESC
```

If you would like to sort by more than one column the columns can be separated by a comma:

```
SELECT customerFirstName, customerLastName
FROM tblCustomer
ORDER BY customerLastName  DESC, customerFirstName  DESC
```

Activity 3.

1. Select everything from the customer table order by first name
2. Select everything from the product table order by productID
3. Select everything from the product table order by product price

Updated 6/2/2012

The WHERE clause

So far every select statement has returned all the rows in the table. It is possible to only retrieve the rows that match particular criteria. For example you may want to see all the products supplied by a particular supplier.

```
SELECT *
FROM tblProduct
WHERE supplierID =1
```

Note: Quotes are used for strings

The WHERE clause can be used in conjunction with various testing operators besides the "=" sign. Specifically, you can use the ">", "<", "<=", ">=", "<>" (means not equal to) operators to select ranges. You can also use "!=" for not equal to.

Example

```
SELECT *
FROM tblProduct
WHERE productID >5
```

You can have multiple conditions combined using AND, OR and NOT

```
SELECT *
FROM tblProduct
WHERE productID >5
AND productID <15
```

Activity 4.

1. Select all products with a unit price less than $3.00
2. Select all products with a unit price greater than $1.00 and less than $3.00
3. Select all products in the beverages category (category 1)
4. Select all products that are not in the beverages category (category 1)

BETWEEN Operator

BETWEEN operator is used in a range situation and replaces the <= and >=. For example

```
SELECT productName, productPrice
FROM tblProduct
WHERE productPrice>=2 AND productPrice<=3
```

Can be replaced by

```
SELECT productName, productPrice
FROM tblProduct
WHERE productPrice BETWEEN 2 AND 3
```

Activity 5.

1. Select all products with a unit price $3 or more and less than or equal to $6
2. Select all products that belong to category 1, 2 or 3

IN Operator

The IN operator is used to replace many OR conditions. For example

```
SELECT productDescription, supplierID
FROM tblProduct
WHERE supplierID=2 OR supplierID=4 OR supplierID=5
```

Can be replaced by

```
SELECT productDescription, supplierID
FROM tblProduct
WHERE supplierID IN (2,4,5)
```

Activity 6.

1. Display products whose price is either $1.50 or $2.50 or $3.50
2. Display the orders that occurred on the even days, that is 24 or 26 or 28-6-2005

The LIKE Operator

This operator uses two (2) characters as wildcards in pattern matching: "**%**" to match 0 to multiple characters and "_" to match one character.

Example

```
    SELECT productName
    FROM tblProduct
    WHERE productName LIKE "P%"
Or  WHERE productName LIKE "%Bread%"
Or  WHERE productName LIKE "Broc_li"
```

The 3 different results would look like

productName
Peppermint tea
Pk 6 Rolls
Pepsi 250ml
Pepsi Max 250ml
Pumpkin

productName
White Bread Toast
Wholemeal Bread Toast
White Bread Sandwich

productName
Brocoli

Activity 7.

1. Select all customers whose surname starts with "S"
2. Find the products that have "100" in their description for grams or mls

Suppressing duplicates (DISTINCT)

If we would like to get a list of the different prices you could write the following SQL statement.

```
    SELECT productPrice
    FROM tblProduct
```

The query brings back 22 rows. 22 rows are brought back because we have 22 products in our database. Some prices are repeated as some products sell for the same price. To display all the product prices without duplicates you need to include the word distinct.

```
    SELECT DISTINCT productPrice
    FROM tblProduct
```

This brings back 8 rows only with no duplications.

Activity 8.

1. Select the distinct supplierID from the product table.

Performing Math

A useful arithmetic tool is the SUM operator that is used to total a column. The basic format looks like:

```
    SELECT SUM(column_name)
    FROM table_name;
    WHERE where_clause [optional];
```

Another couple of useful tools are the MAX and MIN operators that allow you to grab the boundary values in a column (alphanumeric). The most generic syntax follows something like:

```
SELECT MAX(column_name)
FROM table_name
WHERE where_clause [optional];
```

To get the highest order id type in the following:

```
SELECT MAX(orderId )
FROM tblOrder
```

Counting Records

It is also very easy to count the number of records that meet a certain criteria. This function is performed with the COUNT operator and follows the syntax:

```
SELECT COUNT(column_name)
FROM table_name
WHERE where_clause [optional];
```

To count the number of customers type in the following:

```
SELECT count(customerID )
FROM tblCustomer
```

To count the number of items:

```
SELECT count(productID )
FROM tblProduct
```

Sum, count, max and min all return one row containing the result.

Activity 9.

1. What is the highest price in the products table?
2. What is the product name of the highest priced item?
3. What is the average price (use AVG)?
4. What is the difference between COUNT(a field) and COUNT(*)?
5. Use COUNT to find out how many items in the product table belong to category 1 (which is beverages).

Using GROUP BY

You can use GROUP BY with some Math functions to group results. Try this

```
SELECT categoryID, SUM(productPrice)
FROM tblProduct
GROUP BY categoryID
```

Activity 10.

1. Find the sum of product prices grouped by supplier id
2. What is the sum of item quantities for each order number

Multiple tables (table joins)

Primary keys

A *primary key* is a column or set of columns that uniquely identifies the rest of the data in any given row.

Foreign keys

A *foreign key* is a column in a table where that column is a primary key of another table, which means that any data in a foreign key column must have corresponding data in the other table where that column is the primary key. In DBMS-speak, this correspondence is known as *referential integrity*.

The purpose of these *keys* is so that data can be related across tables, without having to repeat data in every table. This is the power of relational databases.

In our database so far we have 2 tables, customer and order, which both contain the customerID field. This field is the primary key of the customer table and a foreign key in the order table.

To retrieve the orders for the customer with a first name of Phillip and last name Cramer we would need to know his customerID (which is 3) and write the following statement:

```
SELECT orderID, orderDate, customerID
FROM tblOrder
WHERE customerID =3
```

orderID	orderDate	customerID
3	2005-06-29	3
4	2005-06-29	3

This output identifies the customer as customer 3. Often full names are more helpful than numbers. To display the customer name instead of their customer id we would need to get the customer name from the customer table and the order information from the orders table:

```
SELECT customerFirstName,
customerLastName
FROM tblCustomer
WHERE customerID =3
```

```
SELECT orderID, customerID,
orderDate
FROM tblOrder
WHERE customerID =3
```

The above statements would give us 2 separate results. If we would like to combine the 2 into one result we need to write the following statement:

```
SELECT orderID, tblCustomer.customerID, customerFirstName,
customerLastName, orderDate
FROM tblOrder, tblCustomer
WHERE tblOrder.customerID = tblCustomer.customerID
AND tblCustomer.customerID =3
```

This would give use the following result:

ordered	customerID	customerFirstName	customerLastName	orderDate
3	3	Philip	Cramer	2005-06-29
4	3	Philip	Cramer	2005-06-29

Notice that both tables involved in the relation are listed in the FROM clause of the statement. The link between the tables is made by the customerID field. This field exists in both the customer and the order table.

The line “where tblCustomer.customerId = tblOrder.customerId” is really saying get all the rows from the customer table which match the customerId in the order table. The dot

notation is needed to avoid ambiguity. When a field has the same name in two tables or more you need to specify the table name (use the dot notation).

Otherwise the following message will be displayed:

```
#1052 - Column: 'customerID' in where clause is ambiguous
```

If you don't join the tables the query will retrieve all possible combinations.

```
SELECT orderID, tblCustomer.customerID, customerFirstName,
customerLastName, orderDate
FROM tblOrder, tblCustomer
WHERE tblCustomer.customerID =3
```

This will bring back 16 rows.

This statement has matched customer 3 with every row in the order table even if these orders were not placed by that customer.

Activity 11.

1. Write the SQL statement to retrieve the first name, last name, customer Id, Order Id, order date, for customer with an ID of 4.

2. Write the SQL statement to retrieve the product Name, product price and the Supplier Name

3. Modify the above SQL statement so that the output is sorted by price.

4. Modify the above code so that the supplier ID is also displayed.

Joining three or more tables

If we want to find out the customer ID, first and last name of the customer, order ID, Order Date, Product ID, orderItemQuantity of the products ordered by the customer 5 we will need to join 3 tables as these values come from 3 separate tables

tblCustomer	customerID	tblOrder	orderID	tblOrderItem	productID
	customerFirstName		orderDate		orderID
	customerLastName		customerID		orderItemQuantity

The customer table can be linked to the order table by the customerID. The order table can be linked to the orderItem table by the orderID. The resulting SQL statement would be:

```
SELECT tblCustomer.customerID, customerFirstName, customerLastName,
tblOrder.orderID, orderDate, productID, orderItemQuantity
FROM tblCustomer, tblOrder, tblOrderItem
WHERE tblCustomer.customerID = tblOrder.customerID
AND tblOrder.orderID = tblOrderItem.orderID
AND tblCustomer.customerID =5
```

Activity 12.

1. Write the SQL to find out the orderID, CustomerID, order date, product id and quantity for customer with a name of "Jane Smith".

Expected output:

orderID	orderDate	customerID	productID	orderItemQuantity
9	2005-06-26	9	19	1

2. Modify the above code so that the product name is displayed instead of the product ID. You will need to add another table the "Product" table.

orderID	orderDate	customerID	productID	orderItemQuantity	productName

9	2005-06-26	9	19	1	Orange

The INSERT statement

To insert a new row of data into a table the syntax is:

```
INSERT INTO table_Name (field1, field2, field3,…)
VALUES (data1, data2, data3, …);
```

NOTE: the second format is preferred.

Example

```
INSERT INTO tblCustomer(customerFirstName, customerLastName,
customerAddress, customerPhone, customerMobile)
VALUES ("Tom", "Jones", "205 Pacific Hwy Hornsby", "1237896",
"0765 567 876")
```

NOTE: the customerID is an auto increment field and should not be inserted because the auto increment datatype is automatically generated for you by the database.

The UPDATE statement

Once the data is in the database it can be modified by using the UPDATE statement.

The syntax for this statement is:

```
UPDATE table_Name SET field1 = newvalue1, field2 = newvalue2, …
WHERE condition
```

Example

```
UPDATE tblCustomer
SET customerPhone = "94725678",
customerMobile = "0657 111 111"
WHERE customerID =11
```

NOTE: If there is no “where” clause all the rows in the table will be updated.

The DELETE statement

It is possible to delete rows from the database the syntax for this is:

```
DELETE FROM table_Name
WHERE condition
```

Example

```
DELETE FROM tblCustomer
WHERE customerID =11
```

NOTE: If there is no “where” clause all the rows in the table will be deleted.

Activity 13.

1. Add yourself to the customer table
2. Modify your first and last name to something else
3. Delete yourself from the customer table