Structured Query Language (SQL)

**Introduction**

A database is a table, or a collection of tables that store data. The software that operates on this data is called a database management system (DBMS) and all database management systems will need to be able to carry out the following actions:

* Create a database
* Add data to the database
* Delete data from the database
* Edit the data
* Search the database for specific for data

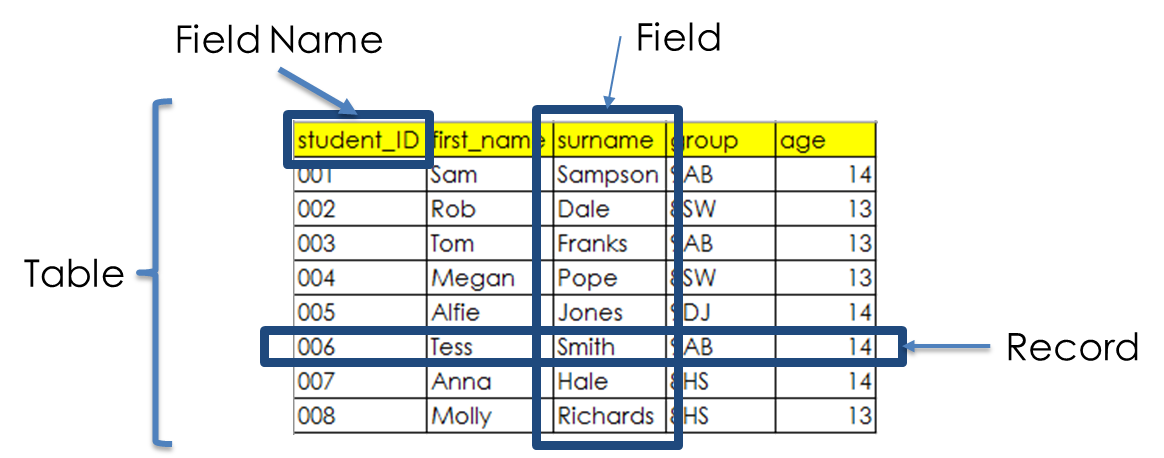
**Database Terminology**

**TABLE**: A table is simply a collection of data that relates to a person or object (often referred to as an entity (e.g. students))

**RECORD**: A collection of data about a single entity (e.g. a student)

**FIELD**: A unique piece of data about an entity (student surnames)

**FIELD NAME**: An identifier for the single piece of data (e.g. ‘surnames’).



**Queries**

A very common job carried out on a database is a query. A query is where we search for data in our database. We ask the database questions and it responds with answers. For example, we might ask our database for all students who are 14 and the database will respond with a list of all students that match the ‘age=14’ criteria.

When we write programs that have databases, SQL (structured query language) is the language we use to create, update, delete and query our program’s database.

**SQL**

The SQL language is made up of a selection of statements to carry out jobs on databases.

There are quite a few but the basic / common statements are:

CREATE TABLE – create a database table

SELECT – selects data

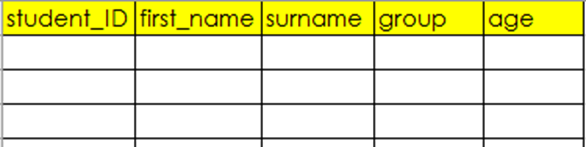
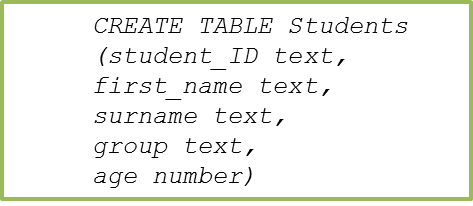
UPDATE – edits data

DELETE – removes data

INSERT INTO – inserts new data

**CREATE-ing a Table**

If we wanted to create the following empty table…



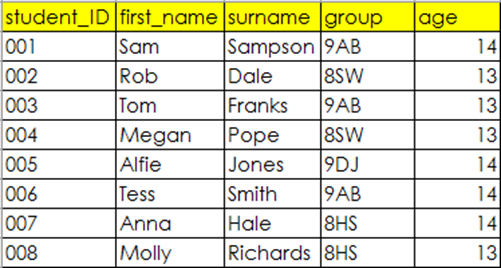
…we would use the following syntax:

The first line of SQL creates a table called ‘Students’. The following lines, separated by commas specify the name of each column and the type of data it is to hold.

**SELECT-ing Data**

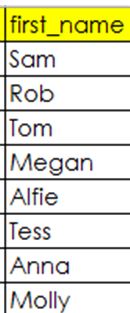
The SELECT statement is used to select particular data that we want to work on. We could select the entire table or specific data based on certain criteria.

Using this example table…

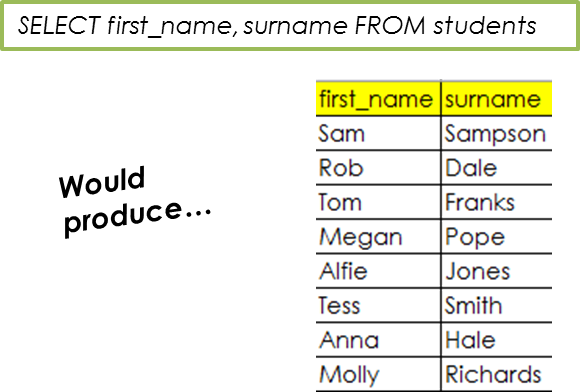


…to select data we could use the following syntax:

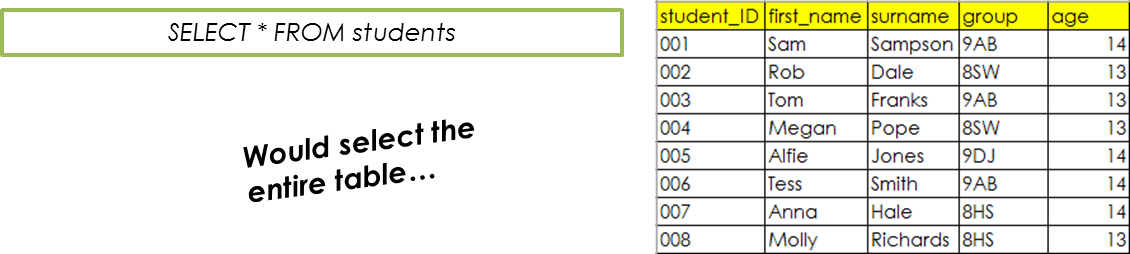
…which would produce the following:



This particular example will select all data in the column called ‘first\_name’.

As well as selecting a single column, you can also select multiple columns from a table…

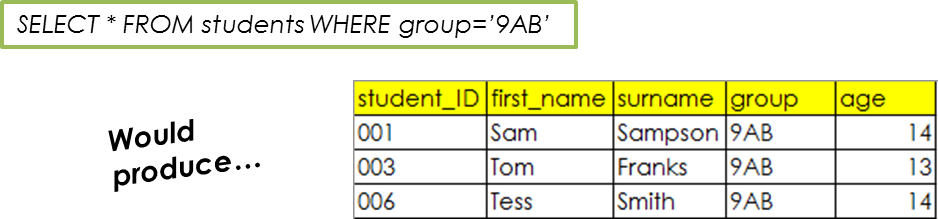
…and if you want to select everything we can use the wildcard (\*).



**The WHERE Clause**

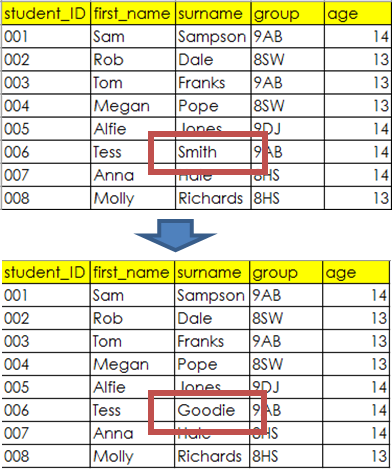
The “Where” clause is used with the ‘Select’ statement and is great if we want to drill down and retrieve specific data which match a given criteria.

The SQL in this example selects all rows in the table where the value in the field column matches ‘9AB’.



**The UPDATE Statement**

The ‘update’ statement allows us to update / alter data that is already stored in the database. The WHERE statement sets the particular record to be updated. If there is no value in the WHERE statement, then all the records will be updated.

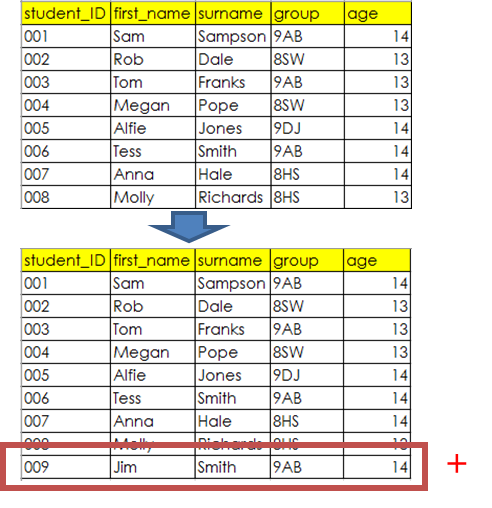
The SQL in this example will update the table called students, changing the surname to ‘Goodie’, but only in the row(s) where the first name matches the value ‘Tess’.





…notice that Tess now has the surname ‘Goodie’.

**The INSERT Statement**

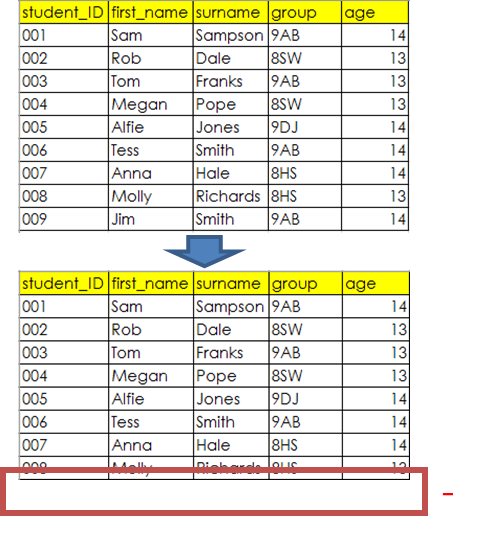
If we want to insert new data into our database we can use the INSERT INTO statement.





The SQL in this example will insert a new row into the table called students, with the values specified in the brackets. They will be entered in the order in which they are provided.

**The DELETE Statement**

Unsurprisingly this statement is used to delete a record / records from a database.

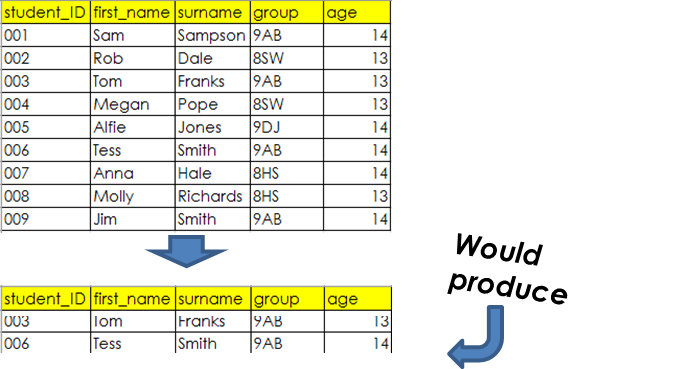


The SQL in this example will delete the row(s) in the table called students, where the value in the field called ‘first\_name’ matches ‘Jim’.

**The LIKE Statement**

The LIKE statement is used with a WHERE statement in order to find patterns within the data in a column. For example it can be used along with a wildcard ‘%’ to find all names that start with a particular letter.

**Example:**

****

SELECT \* FROM students

WHERE first\_name LIKE ‘T%’

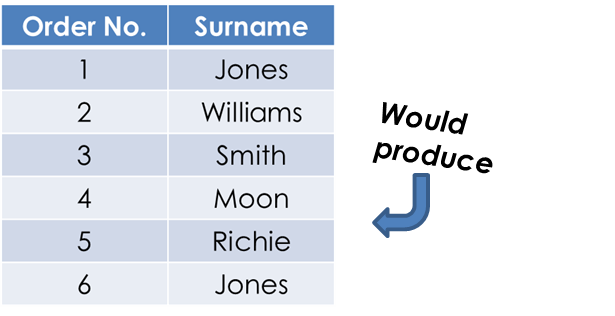
The SQL in this example will select all row(s) in the table called students, where the value in the field called ‘first\_name’ starts with the letter ‘t’.

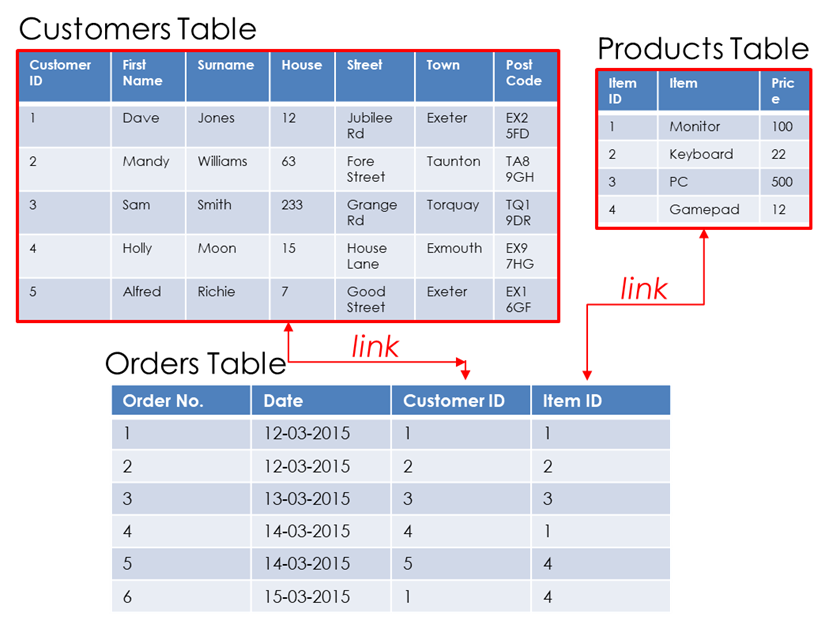
**The JOIN Statement**

The JOIN statement is used to select records in two separate tables that both have the same value.

SELECT Orders.OrderNo, Customers.Surname  
FROM Orders  
JOIN Customers ON Orders.CustomerID = Customers.CustomerID

**Example:**

****

****

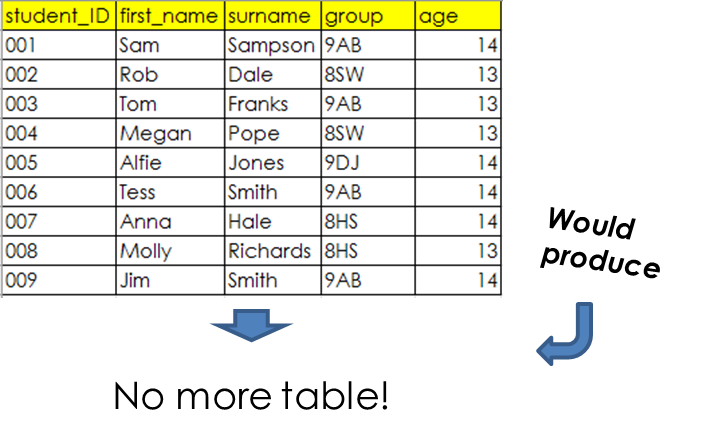
The SQL in this example will join the order numbers with the corresponding customer surnames.

**The DROP Statement**

We have already seen how the DELETE statement can be used to delete data in a table / database. What is important to recognise is that when data is deleted, the actual structure of the table/database will remain.

The DROP statement however, is used to delete either a table or a database’s data along with the table/database structures.

**Example:**

****

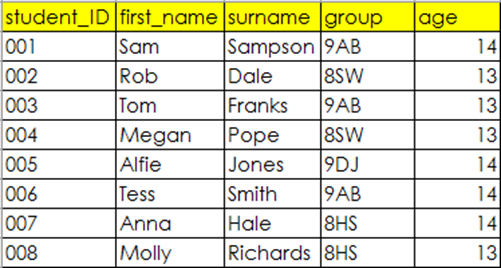
DROP TABLE students

The SQL in this example will remove the table ‘students’ from existence.

**ORDERING QUERY RESULTS**

After selecting a set of records during a query, sometimes it is useful to order the results by a particular field. This can be done using the SQL statement **ORDER BY**.

For example, for the given table of data, if we wished to select all students in a particular group and order the result of the query by surname we could use the following SQL statement:



SELECT \* FROM students

WHERE group = 8HS

ORDER BY surname ASC

DESC can be used to place the results in descending order.

**TRANSACTION PROCESSING**

A single operation that is performed on a database is known as a transaction. For example, a single operation carried out on a database booking system could be the booking of a hotel room, which cannot complete until payment has been made. This transaction therefore could involve more than one process (1. booking 2. after payment confirmed).

It is therefore vital that transactions fully complete and cannot partially complete (e.g. booking without payment).

To ensure that a database processes transactions reliably, it will follow the ACID principles.

**ACID (Atomicity, Consistency, Isolation, Durability)**

**Atomicity**

Atomicity is the requirement that a transaction must be fully processed or not processed at all, removing the possibility of a partially completed transaction.

**Consistency**

Consistency is the requirement that a transaction cannot compromise referential integrity.

For example, consistency will ensure that an order cannot be placed by a customer (who exists in the customer table) for a product that does not exist in the products table. In the same way, consistency would ensure that a product cannot be deleted from the products table if it already exists in a previous order.

**Isolation**

Isolation is the requirement that a transaction is performed in isolation so that no other process can interfere until the transaction is completed. In practical terms, to facilitate this, whilst a transaction is operating on a record, that record is locked, stopping others writing to the record until the transaction has been completed and committed to memory.

**Durability**

Durability is the requirement that once a transaction has completed, the database change is immediately written to secondary storage so that the transaction cannot be lost in the event of a system failure or power cut.

**Record Locking**

There could be a situation where a record is accessed and edited by two (or more) different users, simultaneously, which can cause issues.

*Example:*

For a given customer record, if a database user altered the customer’s surname and another altered the customer’s address and both saved their changes at the same time, what changes would persist?

The likelihood is that one of the changes would not be made. Inconsistencies would therefore be likely.

Record locking ensures that the issues outlined above do not happen.

Record locking is the process by which a record is locked the moment a user retrieves a record, so that no other user can access the record whilst it is being edited/updated. The moment that transaction has completed, the record is then unlocked for others to access.

***Keywords / Key Terms:***

**SQL:***. Structured Query Language- A standard database language used to create and manipulate databases.*

**Query:***. A request for data from a database.*

**Transaction:** *A single operation that is performed on a database.*