# SQL Tutorial 2 – Advance Select Statements

For this tutorial you will need to import the file ‘**CREATE MOVIE RATIN GENRE.sql**’

If you make any mistakes and want to reimport the file you will need to drop the MOVIE table first before you drop either RATING or GENRE tables. Ask you teacher for my information if you want to know why.

If you have forgotten how to import data or how to drop tables see the instructions in SQL Tutorial 1.

This tutorial also assumes that your teacher has gone over relationships and Primary Keys and Foreign Keys (or whatever terminology is used in your class room).

## Advanced Select Statements

In our previous tutorial we had only ever been selecting data from a single table, great if we are using a single flat file database but we want the power that comes with a relational database. To be able to work properly with a relational database we need to be able to select data across multiple tables.

To do this we need to use the INNER JOIN key words. INNER JOIN allows us to connect two or more tables together using their primary and foreign keys.

Take the very incomplete and totally not normalised ERD below as an example. We have three tables, **student**, **subject** **and teacher**. A student can be enrolled in one subject only, a subject can have many students, the student table has a foreign key that refers to the subject table primary key. The same relations can be said between the teacher and the subject table.

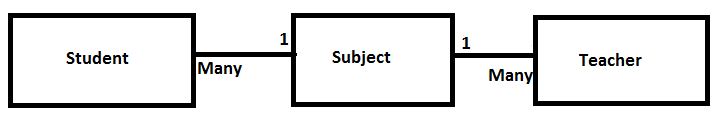
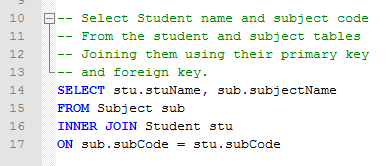


Figure 1 - How can we vastly improve this design?

If we wanted to select the names of all the students and the name of the class they are enrolled in we have to use the following SQL statement.



Primary Key

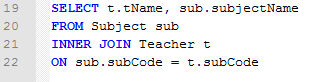
Foreign Key

Alias

Now there seems to be a lot going on in that statement but there is only two new things we need to know about. Firstly the little prefix before our attributes, **stu** and **sub**. These are called aliases, as you may have guessed when we are selecting from more than one table we need to indicate which table the attribute is coming from. The aliases are being declared on lines 15 and 16.

Secondly the INNER JOIN key word. This allows us to join two tables together. The order in which you join the tables together does not matter (for the time being.) But by habit I have always joined the table with the primary key onto the table with the foreign key. As long as you are consistent. The INNER JOIN is a continuation of the FROM Select word. So we state the first table, join it to the second table, and then join them on their primary and foreign keys. (Your teacher will explain it better.)

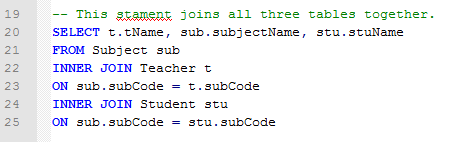
An example for joining the Subject and Teacher table is below.



Questions:

1. What is the alias for the Subject table?
2. What is the alias for the Teacher table?
3. Which table contains the primary key and which table contains the foreign key?
4. Can you tell which table contains the primary key just be looking at the inner join statement?

Below we have an example of joining all three tables together.



Note that, the inner joins don’t flow on to each other, just because we have the student inner join straight after the teacher inner join means we are joining these two tables together. It is the primary key and foreign key statement in the ON key word that joins the tables.

So on line 23 we are joining the Subject and Teacher table together.

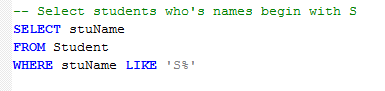
On line 25 we are joining the Subject and Student table together.

## Where statement extended

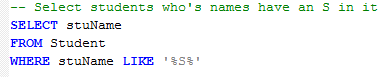
In the last tutorial we used a basic example for the where statement, these example limit the amount of criteria we can use to select data below is a list of useful features that can be used to refine our selections.

**Wildcards and Like**

Wildcards can be used to replace characters on numbers when we want to find data that contains certain values. For example we can use a wild card to select all the students that have a name beginning with S

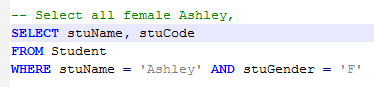


We use the ‘%’ sign to represent any values. We use the LIKE key word for partial matches. You can use as many % signs as you want for example:

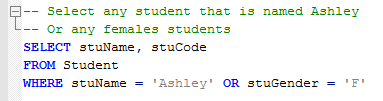


**AND OR NOT**

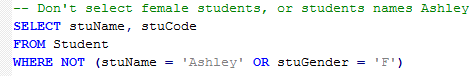
Like programming languages SQL had AND, OR and NOT operations. When using AND both conditions on each side of the AND operations must be true for data to be selected.



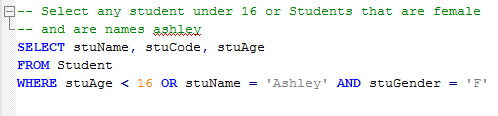
When using the OR operation only one condition on each side of the OR operations must be true for data to be selected.



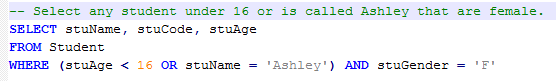
The NOT operations selects any data that makes the conditions it is grouped with false.



You can have multiple AND and OR and NOT operations in the one WHERE clause. But SQL will decided which operation gets precedence. AND always takes precedence over OR. That means the AND is evaluated first before we evaluate the OR.

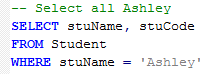


But we can use () to change what is evaluated first.

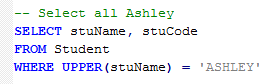


**UPPER and LOWER function**

SQL is case sensitive then comparing strings. If we wanted to find all students named Ashley but someone had missed typed an entry as aShley, the following example would not select that student.



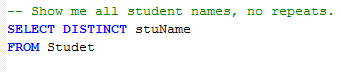
If we wanted to make sure that aShley was also selected we can use the UPPER and LOW functions to temporarily convert all characters to upper or lower case and compare that way.



Note that Ashely is now all in upper case letters for the comparison to work.

**DISTINCT**

Say we wanted a list of all the names we have for our students, but didn’t want to see any duplicates. We can use the **DISTINCT** key word.



## Tutorial Questions

Remember you will need to import the file ‘**CREATE MOVIE RATIN GENRE.sql**’ for this tutorial.

Note: That the movie table has two foreign keys:

* Rating code which refers to the rating table.
* Genre code which refers to the genre table.

Write and execute the SQL statement to solve the following problems.

1. List all rows for
2. The Movie table
3. The Rating table
4. The Genre table.
5. List all columns of movies that have a title that begin with the word **‘the’**. Make the query work for any combination of upper or lowercase characters.
6. Same as above but allow the word **‘the’** to appear anywhere in the title.
7. List all rating codes in the movie table. Each rating should only appear once in the list.
8. Write a single SQL statement that list the title of the movies have any of the following MovieNo: 103, 241, 288
9. Using the Movie tables, list the title, year, runtime and rating of rows that **meet all** of the following criteria:

* Rating code is ‘M’
* Length of the film is 120 minutes or longer

1. List all movies that have the letter ‘A’ anywhere in the title (ant case).
2. List all movies that where not listed in question 7.
3. Write a single SQL statements to list the title, rating code, imdb score and genre code of all movies where either of the following criteria is met:

* Rating ‘PG’ and the imdb score is less than 6
* Rating is ‘M’ and the imdb score is less than 5.5

1. Modify the query from question 9 to list only the movies that meet the original criteria but also have a genre code of ‘C’
2. List each movie’s title, year and genre code where the following criteria is met:

* Year is one the following: 1999 or 2006
* Genre code is D

1. This task will require you to join two tables. For each row in the movie table, display the title, year and the matching shortdesc form the rating table.
2. Same as above, but only for movies where the genre code is ‘A’
3. You must join three tables together for this task. For each row in the movie table, display the title, short description and the matching genre name form the Genre table.
4. Same as above, but only for movies where the rating is ‘PG’ and the genre code is ‘T’