SQL

SELECT Syntax

The SQL SELECT Command

Syntax:

SELECT {* | <col_list>}
FROM <table name 1> [, <table name 2> (….)]
[WHERE <condition> = true]
[GROUP BY <col name> [HAVING <condition> = true]]
[ORDER BY <col name 1> [, <col name 2> (….)]];

Note:

SELECT {* | <col_list>}
FROM <table name 1> [, <table name 2> (….)]
[WHERE <condition> = true]
[GROUP BY <col name> [HAVING <condition> = true]]
[ORDER BY <col name 1> [, <col name 2> (….)]];

Curly braces imply that a value is required here.

Value specified must be either an asterisk OR a list of column names

SELECT is a **required** clause.

SELECT {* | <col_list>}
FROM <table name 1> [, <table name 2> (….)]
[WHERE <condition> = true]
[GROUP BY <col name> [HAVING <condition> = true]]
[ORDER BY <col name 1> [, <col name 2> (….)]];

[ ] brackets denote optional additional table names
(…) denotes possibly more than one additional table name

At least one table name must be specified in the FROM clause

FROM is a **required** clause.

Example:

```
SELECT *
FROM Stock;
```

OR….

```
SELECT Stock_No, Description
FROM Stock;
```

Note:

- Brackets not included!
- Semi-colon at end of statement.

SELECT {* | <col_list>}
FROM <table name 1> [, <table name 2> (….)]
[WHERE <condition> = true]
[GROUP BY <col name> [HAVING <condition> = true]]
[ORDER BY <col name 1> [, <col name 2> (….)]];

[ ] brackets imply that the WHERE clause is optional.

A WHERE clause filters the records in the query result. Only the records that satisfy the WHERE condition are included in the result.

Example:

```
SELECT *
FROM Stock
WHERE Sale_Price > 70;
```

OR….

```
SELECT Stock_No, Sale_Price
FROM Stock
WHERE Sale_Price < 50;
```



```
SELECT {* | <col_list>}
FROM <table name 1> [, <table name 2> (….)]
[WHERE <condition> = true]
[GROUP BY <col name> [HAVING <condition> = true]]
[ORDER BY <col name 1> [, <col name 2> (….)]];
```

GROUP BY and HAVING are optional clauses used to write more complex queries.

More about these later……


```
SELECT {* | <col_list>}
FROM <table name 1> [, <table name 2> (….)]
[WHERE <condition> = true]
[GROUP BY <col name> [HAVING <condition> = true]]
[ORDER BY <col name 1> [, <col name 2> (….)]];
```

[ ] brackets imply that the ORDER BY clause is optional.

An ORDER BY clause imposes an order on the way the resulting records are displayed.
Default sort order is ASCENDING; Must specify DESCENDING if required.

Example:

```
SELECT *
FROM Stock
WHERE Sale_Price > 70
ORDER BY Sale_Price;
```

OR….

```
SELECT Stock_No, Sale_Price
FROM Stock
WHERE Sale_Price < 50
ORDER BY Sale_Price DESCENDING;
```


Some Rules for Writing SQL Commands

1. SQL ***clauses*** cannot be split across lines:

```
SELECT * FROM Stock WHERE Sale_Price > 70;
```

```
SELECT *
FROM Stock
WHERE Sale_Price > 70;
```

```
SELECT *
FROM Stock WHERE Sale_Price > 70;
```

```
SELECT *
FROM Stock WHERE
Sale_Price > 70;
```

2. The building of words in a statement must follow syntax rules.

```
SEL *
FROM Stock
WHERE Stock_No = 2;
```

```
SELECT *
FROM Stock
WHERE Stock_No = 2;
```

3. Most components of the SQL statement are case ***insensitive***. Literal character data is case sensitive. The following queries produce different results

```
SELECT *
FROM Students
WHERE Surname = 'SMITH';

SELECT *
FROM Students
WHERE Surname = 'smith';
```


5. A statement terminator must be used to indicate the end of each SQL statement **(;)**

```
SELECT *
FROM Stock;
```


6. For clarity, each ***clause*** in an SQL statement ***should*** begin on a new line.

```
SELECT *
FROM Stock
WHERE Sale_Price > 70
ORDER BY Description;
```

The beginning of each clause in the statement should line up with the beginning of the other clauses.

7. If a clause has several parts, they should each appear on a separate line and should be *indented* under the start of the clause

```
SELECT *
FROM Stock
WHERE Sale_Price > 50 AND
        Sale_Price < 80
ORDER BY Description;
```


Wild Cards

Wildcards can be used in string literals when the exact value of the data is unknown.

For example, you may wish to retrieve

- All students whose surname begins with the letter 'F'
- All students whose surname begins with the letters 'FITZ'
- All students whose surname contains the string 'ITZ'

Microsoft Access wildcards

- an **asterisk** (*) character represents any sequence of zero or more characters
- an **question mark** (?) character represents any single character

Wildcard symbols differ from product to product.

In ORACLE, the percentage (%) symbol represents zero or more characters.

We will use the (*) character in our examples.

Examples:

The following examples are based on the tables in the StockSYS database.

List all Stock ***containing*** the string 'Levis' in its description.

```
SELECT *
FROM Stock
WHERE Description LIKE '*Levis*';
```


List all Stock whose description ***begins*** with the string 'Levis'.

```
SELECT *
FROM Stock
WHERE Description LIKE 'Levis*';
```


List all Stock whose description ***ends*** with the string '01'.

```
SELECT *
FROM Stock
WHERE Description LIKE '*01';
```


ORDER BY Clause

The ***ORDER BY*** clause is used to display the records in the result of a query in a specific order, either ***ascending*** or ***descending***.

The default sort order (if no order is specified) is **ascending**. Include the key word DESC or ASC after the sort column.

```
SELECT *
FROM Stock
ORDER BY Sale_Price DESC;
```

Sorting is permitted on multiple levels.

Consider the following table:

Students

Stud_ID	Surname	Forename	CourseCode
1	SMITH	FRED	TL_KCPSD_Y2
2	JONES	ANNE	TL_KCPMM_Y1
3	AHERN	TOM	TL_KCPMM_Y1
4	TWOMEY	MIKE	TL_KCPGD_Y2
5	SMITH	ADAM	TL_KCPMM_Y3

Suppose we wish to list all students in order of Surname and then Forename.

The required query is:

```
SELECT *
FROM Students
ORDER BY Surname, Forename;
```

This produces the following result-set:

Students | Query1

Stud_ID	Surname	Forename	CourseCode
3	AHERN	TOM	TL_KCPMM_Y1
2	JONES	ANNE	TL_KCPMM_Y1
5	SMITH	ADAM	TL_KCPMM_Y3
1	SMITH	FRED	TL_KCPSD_Y2
4	TWOMEY	MIKE	TL_KCPGD_Y2

List Stock_No, Description and Cost_Price for all stock , shown in descending order of CostPrice.

SELECT Stock_No, Description, Cost_Price
FROM Stock
ORDER BY Cost_Price **DESC**;

List Stock_No, Description and Cost_Price for all stock , shown in ascending order of CostPrice.

SELECT Stock_No, Description, Cost_Price
FROM Stock
ORDER BY Cost_Price **ASC**;

OR

SELECT Stock_No, Description, Cost_Price
FROM Stock
ORDER BY Cost_Price;

List Stock_No, Description, Cost_Price and Sale_Price for all stock , shown in ***descending*** order of Sale_Price and ***ascending*** order of Cost_Price.

SELECT Stock_No, Description, Cost_Price, Sale_Price
FROM Stock
ORDER BY Sale_Price DESC, Cost_Price;

The columns you are sorting by do not have to appear in the SELECT clause.

SELECT Stock_No, Description, Cost_Price
FROM Stock
ORDER BY Sale_Price DESC;