

SELECT (\* | <col\_list>)
FROM stable name !> [, (....)]
[WHERE <condition> = true]
[GROVP BY <cri 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.

Dutabase Concepts: DB04 SQL SELECT 3

FROM [, (....)]
[WHERE <condition> true]
[GROUP BY <col name ! [HAVINE <condition> = true]]
[ORDER BY (...)]];

[] 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:

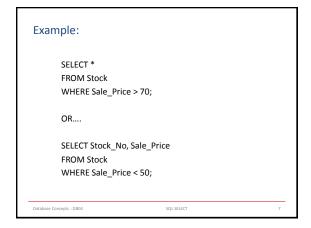
Brackets not included!

Semi-colon at end of statement.

SELECT {\* | <col\_list>}
FROM [, (....)]
[WHERE <condition> = true]
[GROUP BY <col name> [AVING <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.



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

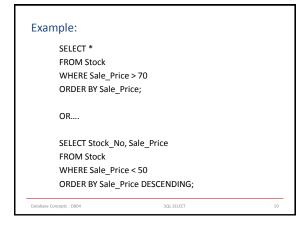
GROUP BY and HAVING are optional clauses used to write more complex queries.

More about these later......
```

```
SELECT (* | <col_list>)
FROM  [,  (....)]
[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.
```



Some Rules for Writing SQL Commands

Database Concepts: DB04 SQL SELECT 11

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; 

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; 

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; 

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; 

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; 

SELECT \*

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;

Dutublese Concepts: 28004

SQL SELECT 13

 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';

Database Concepts: DB04 SQL SELECT 14

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

SELECT \*
FROM Stock;

Database Concepts : DB04

SQL SELECT

6. For clarity, each <u>clause</u> 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.

Database Concepts: DB04

SQL SELECT

If a clause has several parts, they should each appear on a separate line and should be <u>indented</u> under the start of the clause

SELECT \*
FROM Stock
WHERE Sale\_Price > 50 AND
Sale\_Price < 80
ORDER BY Description;

Database Concepts : DB04

SQL SELECT

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'

Database Concepts : DB04

SQL SELECT

## 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.

Database Concepts : DB04

SQL SELECT

## Examples:

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

Database Concepts : DB04

SQL SELECT

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

SELECT \*

FROM Stock

WHERE Description LIKE "\*Levis\*";

Database Concepts : DB04

QL SELECT

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

**SELECT** \*

**FROM** Stock

WHERE Description LIKE 'Levis\*';

Database Concepts : DB04

SQL SELECT

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

**SELECT** \*

**FROM** Stock

WHERE Description LIKE '\*01';

Database Concepts : DB04

SQL SELECT

## 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.

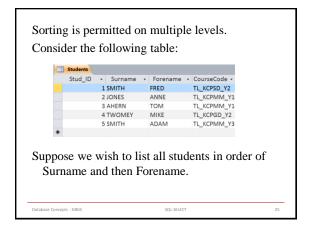
SELECT \*

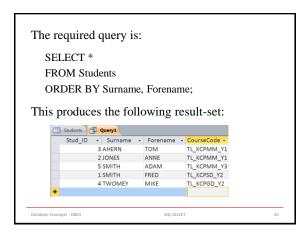
FROM Stock

ORDER BY Sale\_Price DESC;

Database Concepts: DB04

SQL SELECT





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;