The SELECT Statement is used to retrieve rows, columns and/or derived values from one or more tables in a database.

SELECT CLAUSE…

The SELECT clause is used to specify the list of items (i.e. columns and/or derived values) of information that a user wants to retrieve from the database. All desired columns are separated by commas. An asterisk (*) is used as the shorthand to retrieve all the columns in every table shown in the FROM clause.

When a SELECT statement is executed the title used for the column is the same as the defined name of the attribute in the table. Sometimes this is not desirable. To make the output more readable it would be desirable to be able to specify the title of a column that should be displayed. This is accomplished with aliases. To specify an alias, use the keyword AS to replace the default column names that are displayed when the results are returned. For example,

```
SELECT pub_name AS Publisher, pub_id

FROM publishers;
```

FROM CLAUSE…

The FROM clause of the SELECT statement specifies the tables from which the columns are selected. You can specify a qualifier for the table. For example,

```
SELECT p.pub_id, p.pub_name

FROM publishers p;
```

In this example, p represents a shorthand for publishers. When working with multiple tables there is a possibility that the same column name may exist in more than one table. In that case, you MUST use the qualifier to indicate which table should be used to retrieve the value for the specified column. Therefore, get in the habit of using table qualifiers.

WHERE CLAUSE…

The WHERE clause is used for two things: one use is to specify the search condition for the selection and the second use is to specify the join condition when multiple tables have been specified in the FROM clause.

Examples of the first use (search conditions):

```
SELECT title FROM titles WHERE advance * 2 > ytd_sales * price;

SELECT title FROM titles WHERE advance < 5000 or ytd_sales;

SELECT title FROM titles WHERE ytd_sales BETWEEN 4095 AND 12000;
```

Additional example:

```
SELECT p.pnumber, p.pname, w.hours

FROM project p, work_on w

WHERE p.pnumber=w.pno AND w.hours>10;
```

ORDER BY CLAUSE…

Use the ORDER BY clause to specify which columns should be used for sorting the results. It is possible to specify the direction of the sort by using the keyword ASC (for ascending) or DESC (for descending). The default sort order is ascending order, so you don't need to specify ASC. For example,

```
SELECT p.pnumber, p.pname, w.hours

FROM project p, work_on w

WHERE p.pnumber=w.pno AND w.hours>10

ORDER BY p.pnumber;
```

GROUP BY CLAUSE…

Every item in a GROUP BY list must appear in the SELECT list because you can make groups only out of the things that are selected. It divides the tables into sets of rows. The GROUP BY clause is typically used in conjunction with aggregate functions (which will be discussed shortly). Aggregate functions are built in functions that act upon a group of records (e.g. SUM, COUNT, et.). Aggregate functions will be presented in further detail on a separate slides and notes.

HAVING CLAUSE…

The HAVING clause is used to provide a filtering condition that each of the groups created by the GROUP BY clause must meet in order to be returned in the result set.

The basic syntax for the WHERE clause is as follows: WHERE expression comparison_operator expression. The comparison operator can be any of the items specified on this slide.

You should note that LIKE is a way to match character strings. Unlike testing for equality or inequality, LIKE actually does pattern matching. You can use wildcard characters with the LIKE comparison_operator. % means any string of zero or more characters. _ (an underscore) means any single character.

When testing whether or not a field contains a NULL value, use IS NULL (or IS NOT NULL). Do NOT use WHERE field = NULL, which would be INCORRECT.

Examples:

SELECT pub_name FROM publishers WHERE state IN ('CA', 'IN', 'MD');

SELECT title FROM titles WHERE advance IS NULL;

SELECT title FROM titles WHERE advance IS NOT NULL;

SELECT au_lname FROM authors WHERE phone NOT LIKE '415%';

More Examples for LIKE:

SELECT au_lname, city FROM authors WHERE au_lname LIKE 'Mc%' OR au_lname LIKE 'Mac%';

SELECT au_lname, phone FROM authors WHERE phone LIKE '415%';

SELECT title_id, notes FROM titles WHERE notes LIKE '%xercise%';

SELECT au_lname, city FROM authors WHERE au_lname LIKE '_ars_n';

Aggregate functions (i.e. column functions) can be used to obtain summary values. Aggregates are applied to sets of rows such as –

-All rows in a table,

-Rows specified by a WHERE clause, or

-Groups of rows set up by the GROUP BY clause

Aggregates yield one value for each set of rows.

IMPORTANT: Any field specified in the SELECT clause that is not directly used in an aggregate function MUST be specified in the GROUP BY clause in the order listed in the SELECT clause; otherwise, the SQL statement will fail!