# Week 2: Restricting and Sorting Data

**Background**

Last week we discussed about the simple SQL Select Statement, which is used to retrieve information from a database table. The simple Select statement displays us all of the data entries into that table. But sometimes need arises and we want only some specific data to be displayed in some particular manner. In that situation a simple Select statement will not work. To handle such requirement we need some clauses along with the Select statement to restrict and sometime sort the data being displayed.

Let’s take a look again on the Syntax of Select statement this time with restriction and sorting clauses need by the simple query:

|  |
| --- |
| SELECT columnname1 [, columnname2]  FROM tablename1 [, tablename2]  [WHERE condition] [And | Or condition...]  [ORDER BY "column-list" [ASC | DESC] ] |

As we can see that we have added to clause to our simple Select statement which will help us to perform some specific actions on the data in the table.

The WEHRE clause is used to restrict the data to be displayed and ORDER BY as the name suggest sort the data into a specific order required by the user. Both these clauses are discussed in detail later in the notes with proper examples.

**Limiting Rows**

As we have already discussed that every time when we want to display some data it is not necessary that we want the complete table to be display. According to the requirements we can restrict the selection and can only display whatever is required at that point in time.

To achieve this we need to add a clause to our simple Select statement which is known as WHERE clause. It is an optional clause, possibilities are that we won’t require this clause at times but it is used more frequently with the Select statement for the better representation of data.

A WEHRE clause is always placed after the FROM keyword. It contains a condition which must be met in order to limit the rows. The syntax of a Select statement with the WHERE clause is as follows:

|  |
| --- |
| **Syntax:**  SELECT columnname1 [, columnname2]  FROM tablename1 [, tablename2]  [WHERE condition] [And | Or condition...] |

In the syntax:

**WHERE** restricts the query to display specific rows

**Condition** is composed of column names, expressions,

Constants and comparison operators etc.

A WHERE clause can compare values in columns, literal values, arithmetic expressions or functions, but we cannot use any column alias name

|  |  |  |
| --- | --- | --- |
| **Example 2.1: Using WHERE clause in Select statement**   |  | | --- | | **SELECT B\_Code, Title, P\_Code, Type**  **FROM Books**  **WHERE Price = 14;** | |  | |
|  |

In example 2.1 the Select statement fetches the book code, title, Publisher code and type for all the books whose price is £ 14. It is not required to include the column name in the SELECT clause to use them in the WHERE clause.

Character strings and Date values can also be used to limit the row selection. The character strings and date values in a WHERE clause are always enclosed in single quotation mark (‘‘).

However the numeric constants should avoid using this. The value enclosed within the single quotation mark is case sensitive; therefore a special care should be taken while searching for any value.

Also it is not required to specify the column names in the SELECT clause to use them in the WHERE clause.

|  |  |  |
| --- | --- | --- |
| **Example 2.2: Using WHERE clause with character value in Select statement without specifying column names**   |  | | --- | | **SELECT \***  **FROM Publisher**  **WHERE City = ‘New York’;** | |  | |
|  |

The example 2.2 shows the use of character string in the WHERE clause. As we can see that the string enclosed in the single quotation mark ‘New York’, if we change the case of the string and write ‘new york’ instead then we will not receive any output because oracle takes the case of the string into consideration and checks the string with the exact case.

**WHERE Clause with Comparison Conditions**

A comparison condition is one which uses comparison operator to compare one expression to another expression or value. There is a wide variety of operator used for comparison in WHERE clause all of them are discussed below in details.

1. **Simple Comparison Operator:** The simple comparison operators are those who perform simple tests of equality, inequality, less than, greater than and value comparison. The table below shows all the simple operator used with WHERE clause:

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **=** | Equal to |
| **>** | Greater than |
| **>=** | Greater than or equal to |
| **<** | Less than |
| **<=** | Less than or equal to |
| **<>** | Not equal to |
| **ANY / SOME** | Compares two a value to each value in List or subquery. |
| **ALL** | Compares a value to every value in a list or subquery. |

These simple comparison operators are quite straight forward. But still we have discussed some operators which are new to us in the following examples:

|  |  |  |
| --- | --- | --- |
| **Example 2.3: Using WHERE clause with simple comparison operator Less than or equal to (<=)**   |  | | --- | | **SELECT Title, P\_Code AS ‘Publisher’, Paperback, Price**  **FROM Books**  **WHERE Price <= 11;** | |  | |
|  |

Example 2.3 uses the simple comparison operator *Less than or equals to* (<=) this operator fetches all the records which are either less than or equals to the value specified on the right hand side of the expression, in this case it is 11. Oracle fetches all the values from the books table which are either less than 11 or equals to 11.

|  |  |  |
| --- | --- | --- |
| **Example 2.4: Using WHERE clause with simple comparison operator Less than or equal to (<=) and ANY operator**   |  | | --- | | **SELECT Title, P\_Code, Paperback, Price**  **FROM Books**  **WHERE Price <=ANY (8, 11, 12);** | |  | |
|  |

Example 2.4 demonstrate the use of ANY or SOME operators; these operators are always preceded with the simple comparison operators (i.e. =, <, <=,>,>=, <>). As we have seen in example 12 the select statement displays the record where the price is less than 11 where as in example 13 the where condition is comparing the prices with the range of prices. It compares a value to each of the values given in the list.

|  |  |  |
| --- | --- | --- |
| **Example 2.5: Using WHERE clause with simple comparison operator Less than or equal to (>=) and ALL operator**   |  | | --- | | **SELECT Title, P\_Code, Paperback, Price**  **FROM Books**  **WHERE Price >=ALL (8, 11, 12);** | |  | |
|  |

Example 2.5 demonstrate the use of ALL operator, it compares a value to every value in the list; it is also preceded by a simple operator (i.e. =, <>, <,>, <=,>=).

1. **Logical comparison Operator:**  The Logical operators are used to combine the results of two comparison condition or to invert the result of a single comparison condition. The operator which falls under this category are listed below and explained in detailed.

* **AND –** The AND operator is used when we want to compare two or more condition in the Select statement and at the same time we want all the conditions to be true. An AND operator is TRUE if both the conditions are TRUE and it will result in FALSE if any of the condition is FALSE. The truth table for AND operator make the picture clearer.

**AND Truth Table**

|  |  |  |
| --- | --- | --- |
| Condition 1 | Condition 2 | Result |
| *TRUE* | *TRUE* | ***TRUE*** |
| *TRUE* | *FALSE* | ***FALSE*** |
| *FALSE* | *TRUE* | ***FALSE*** |
| *FALSE* | *FALSE* | ***FALSE*** |

|  |  |  |
| --- | --- | --- |
| **Example 2.6: Using WHERE clause with Logical AND operator**   |  | | --- | | **SELECT \***  **FROM Books**  **WHERE Price <= 8 AND Type = ‘HOR’;** | |  | |
|  |

In example 2.6 we are using the AND comparison operator, here in this example we have specified two condition and the output is the record which matches both the conditions. If there is no record matching the condition the Oracle will simply return a NULL.

* **OR –** The OR operator is also used for the similar purpose as AND operator is used but the major difference between the two is that OR operator returns all the records which matches any of the two conditions specified in the WHERE clause. Therefor an OR operator is TRUE if any of the condition is TRUE and is FALSE if both the conditions are FALSE. The truth table for the OR operator is given below to make the concept more understandable.

**OR Truth Table**

|  |  |  |
| --- | --- | --- |
| Condition 1 | Condition 2 | Result |
| *TRUE* | *TRUE* | ***TRUE*** |
| *TRUE* | *FALSE* | ***TRUE*** |
| *FALSE* | *TRUE* | ***TRUE*** |
| *FALSE* | *FALSE* | ***FALSE*** |

|  |  |  |
| --- | --- | --- |
| **Example 2.7: Using WHERE clause with Logical OR operator**   |  | | --- | | **SELECT \***  **FROM Books**  **WHERE Price <= 8 OR Type = ‘HOR’;** | |  | |
|  |

Example 2.7 shows us the use of the OR operator in the select statement with the WHERE clause**.** It is similar to the example 15 but the only difference is it uses OR operator and the output which we got are the records which matches with any of the condition specified in the WHERE clause.

* **NOT –** The NOT operator is used to reverse the result. It returns FALSE if the operand is TRUE and TRUE if it is FALSE. Therefore NOT operator basically negates the output whatever it is. The truth table for NOT operator is given below:

**NOT Truth Table**

|  |  |
| --- | --- |
| Condition 1 | Result |
| *TRUE* | ***FALSE*** |
| *FALSE* | ***TRUE*** |
| *NULL* | ***NULL*** |

|  |  |  |
| --- | --- | --- |
| **Example 2.8: Using WHERE clause with Logical NOT operator**   |  | | --- | | **SELECT \***  **FROM Books**  **WHERE NOT( Type = ‘HOR’);** | |  | |
|  |

Example 2.8 above demonstrates the use of NOT operator; it shows all the Books records except the ones which have TYPE as HOR. If we remove the NOT from the WHERE clause it will only display the books with TYPE as HOR.

1. **Other Comparison Operator:** Apart from the simple and Logical operators there are few more operators which can be used in the WHERE clause for comparison of a range of values or to matching pattern while comparing. The different comparison operators associated with this category are discussed in detail below:
2. **BETWEEN:** The BETWEEN operator is used to test a range. Let say there are two values A and B; the BETWEEN evaluates the of values falling under A AND B, it is TRUE if the value is greater than or equals to A or less than or equals to B. the BETWEEN operator always uses the logical AND operator to create a range.

|  |  |  |
| --- | --- | --- |
| **Example 2.9: Using WHERE clause with BETWEEN operator**   |  | | --- | | **SELECT B\_Code, Title, Price**  **FROM Books**  **WHERE Price BETWEEN 11 AND 14;** | |  | |
|  |

Example 2.9 demonstrates the use of the BETWEEN operator, here in this example only the book code, title and price of those books whose price falls under the range of 11 and 14. If NOT is used with the BETWEEN operator the output will be the reverse of the actual output.

1. **IN:** The IN operator allow use to specify the discrete value in the WHERE clause as search criteria. It is also known as the *membership condition.* The IN operator is considered equivalent to the = ANY operator, which evaluates to TRUE is the value exists in the list or the result set from subquery (discussed later). IN operator can be used with any data type i.e. character string, numeric, date etc.

|  |  |  |
| --- | --- | --- |
| **Example 2.10: Using WHERE clause with IN operator**   |  | | --- | | **SELECT B\_Code, Title, Price**  **FROM Books**  **WHERE Price IN (11, 12, 14);** | |  | |
|  |

In example 2.10 only those records are displayed which matches with the values in the list or in other words are *‘IN the list of values’.* If NOT is used with the IN operator it will reverse the output and also it is considered equivalent to! = ALL.

1. **EXISTS:** The EXISTS operator is always followed by a subquery (*discussed later in the notes)*. EXISTS evaluates to TRUE if the subquery returns at least one row.

The example 2.11 below demonstrate the use of EXISTS operator with the subquery *(the concept of subquery are discussed in Week 6)*. Here in this example book code and title of those books are displayed whose quantity in Inventory is equal to 1. The output of the main SELECT statement is dependent on the output provided by the subquery mentioned after the EXISTS operator within the parenthesis.

|  |  |  |
| --- | --- | --- |
| **Example 2.11: Using WHERE clause with EXISTS operator**   |  | | --- | | **SELECT B\_Code, Title**  **FROM Books B**  **WHERE EXISTS**  ***(SELECT Quantity***  ***FROM Inventory I***  ***WHERE I.B\_Code = B.B\_Code AND Quantity = 1)*;** | |  | |
|  |

1. **IS NULL:** The IS NULL operator tests for the NULL values. A NULL value is the one which is unavailable, unassigned, unknown or inapplicable; for such kind of values = or! = will not work. IS NULL evaluates to TRUE is the value is NULL.If the values which are not NULL are to be searched IS NOT NULL keyword is used it reverse the output of IS NULL operator. The example 2.12 below explains the use of IS NULL operator.

|  |  |  |
| --- | --- | --- |
| **Example 2.12: Using WHERE clause with IS NULL operator**   |  | | --- | | **SELECT \***  **FROM Books**  **WHERE Paperback IS NULL;** | |  | |
|  |

1. **LIKE:** The LIKE clause is used for searching character pattern. Sometime we just want to search the names starting with some character or want to search the names or titles with some kind of a pattern in them.

This character pattern matching is known as the ***wildcard*** search. It uses to symbols to characterise our search criteria for the LIKE operator. These symbols are explained below:

|  |  |
| --- | --- |
| **SYMBOL** | **DESCRIPTION** |
| **%** | It is used to match any sequence of zero or more characters |
| **\_ (underscore)** | It is used to match any single character. |

These symbols can be used in combination with literal characters. There is also an ESCAPE option associated with the LIKE operator which is always used to differentiate between literals and the keywords. For example we may require to search a string which is comprised of an underscore (\_), now to interpret as underscore and not as the oracle keyword we use backslash (\) as the escape character any such keyword.

The example 2.13 demonstrates the use of LIKE operator using bot the symbols in combination. Like clause never uses equals to sign. Here in the WHERE clause we are searching for all those books name which starts with ‘T’, then have some character in between, then the third character as ‘e’ and then followed by any number of characters. But it should match this character pattern.

|  |  |  |
| --- | --- | --- |
| **Example 2.13: Using WHERE clause with LIKE operator**   |  | | --- | | **SELECT title**  **FROM Books**  **WHERE title LIKE ‘T\_e%’ ;** | |  | |
|  |

**Precedence Rule**

The Priority of an operator helps to determine the order in which an expression is to be evaluated and calculated. After covering all the operators in week one as well as in week two we are ready to understand their priority or the precedence over each other and what to expect if two or more operators appears together in an expression. The table below displays the order of precedence/priority. Not to forget we can always override this order by enclosing the expression into parenthesis we want to evaluate first.

|  |  |
| --- | --- |
| **Priority** | **Operator** |
| 1 | Arithmetic Operator **(+ -, \*, /, +, -)** |
| 2 | Concatenation Operator **(||)** |
| 3 | Simple Comparison Operator **(>, >=, <, <=, <>, ANY/SOME, ALL)** |
| 4 | **IS (NOT) NULL, LIKE, (NOT) IN** |
| 5 | **(NOT) BETWEEN** |
| 6 | Logical **NOT** Operator |
| 7 | Logical **AND** Operator |
| 8 | Logical **OR** Operator |

**Sorting Rows**

The order of the rows retrieved by the select query is always undefined and it’s not always the case that data stored in the table is entered in a defined order. But sometimes it becomes necessary to view the data in a particular order for better understanding and readability. Although we cannot change the order of data in the table but we can arrange the output in a specific order. This can be achieved by using the OREDER BY clause with the SELECT statement. Below is the syntax of SELECT statement with ORDER BY clause:

|  |
| --- |
| **Syntax**  SELECT columnname1 [, columnname2]  FROM tablename1 [, tablename2]  [ORDER BY "column-list" [ASC | DESC] ] |

In the syntax:

**ORDER BY** Specify the order in which the rows are to be displayed.

**ASC** Arrange the rows in ascending order (default order).

**DESC** Arrange the rows in descending order.

The ORDER BY clause is always written after the FROM clause and WHERE clause in the SELECT statement. It can also be used in the absence of the WHERE clause.

Example 2.14 demonstrates the use ORDER BY clause, here in this example we want to display the first name and last names of the authors in an ascending order, As you must have noticed we haven’t used ASC in our Select statement, because it is not necessary whenever we use ORDER BY clause to sort our data is by default sort them in ascending. But to display the data in descending order we need to specify the keyword DESC.

Example 2.15 displays the same select statement in a descending order.

|  |  |  |
| --- | --- | --- |
| **Example 2.14: Using ORDER BY clause to sort the data**   |  | | --- | | **SELECT Fname, Lname**  **FROM Author**  **ORDER BY Fname ;** | |  | |
|  |
|  |
|  |

|  |  |  |
| --- | --- | --- |
| **Example 2.15: Using ORDER BY clause to sort the data in descending order**   |  | | --- | | **SELECT Fname, Lname**  **FROM Author**  **ORDER BY Fname DESC;** | |  | |

**ORDER BY with Column Alias**

We can use the column alias name in the OREDER BY clause to sort our data. For instance example 2.16 displays the publisher’s name in descending order by using the Alias column name.

|  |  |  |
| --- | --- | --- |
| **Example 2.16: Using ORDER BY clause with column alias name**   |  | | --- | | **SELECT P\_Name AS “Publisher Name”**  **FROM Publisher**  **ORDER BY “Publisher Name” DESC;** | |  | |
|  |

**ORDER BY with Multiple Columns**

ORDER BY clause can use more than one column for sorting. The column specified after the ORDER BY clause are separated by commas. We can also mention the columns which are not specified in the SELECT clause, but if the DISTINT clause is used in the Select statement then the sorting can only be done on the columns specified in the SELECT clause.

Example 26 demonstrates the use ORDER BY clause with multiple column names. If we want to reverse the output then we are required to add DESC keyword after the column name.

|  |  |  |
| --- | --- | --- |
| **Example 2.17: Using ORDER BY clause with column alias name**   |  | | --- | | **SELECT B\_Code, title, P\_Code, Price**  **FROM Books**  **ORDER BY Price, title;** | |  | |
|  |

## SOLVED PRACTICE QUESTIONS

**Practice Set – 2.1**

1. What is the default display length of the DATE datatype column?
2. 8
3. 9
4. 19
5. 6

**Solution:**

(B) The default display format of the DATE column is DD-MON-YY, hence its length is 9.

1. You issue the following query:

SELECT B\_Code, Title

FROM Books

WHERE Price = 12 or Price = 14;

Which other operator can replace the OR condition in the WHERE clause?

1. BETWEEN
2. LIKE
3. <=
4. IN
5. >=

**Solution:**

(D) IN operator can be used in WHERE clause in the following way

**WHERE Price IN (12, 14)**

1. Which special character is used to query all the columns from the table without

Listing each column by name?

1. %
2. &
3. \*
4. @

**Solution:**

(C) An asterisk (\*) is used to denote all columns in a table.

1. Display the name of all the authors who have an *a* and *e* in their last name.

**Solution:**

SELECT Lname AS “Last Name” FROM Author

WHERE Lname LIKE ‘%a%’ AND Lname LIKE ‘%e%’;

|  |
| --- |
|  |

**Output:**

1. Display Book code, Book Title Price and type of the all the books whose type is

FIC in alphabetical order of Book title.

**Solution:**

SELECT B\_Code, Title, Price, Type FROM Books

WHERE TYPE=’FIC’

ORDER BY Title;

**Output:**

|  |
| --- |
|  |

1. Display the book title and Price for all the books whose price is not in the range of

£10 and £20.

**Solution:**

SELECT Title, Price FROM Books

WHERE Price NOT BETWEEN 10 AND 20;

**Output:**

|  |
| --- |
|  |

1. Display the books name and price for all the books whose price is in the range of

£ 10 and £ 20, also arrange the data in descending order of Price

**Solution:**

SELECT Title, Price FROM Books

WHERE Price BETWEEN 10 AND 20

ORDER BY Price DESC;

**Output:**

|  |
| --- |
|  |

1. List all the details of all the publishers. Order the list in descending order

by Publisher code.

**Solution:**

SELECT \* FROM Publisher

ORDER BY P\_Code DESC;

**Output:**

|  |
| --- |
|  |

1. When doing pattern matching using the LIKE operator, which of the following

Character is used as the default escape character by Oracle?

1. |
2. /
3. \
4. There is no default escape character in Oracle9i

**Solution:**

(D) There is no default escape character in Oracle9i.

1. Column alias names cannot be used in which clause?
2. SELECT clause
3. WHERE clause
4. ORDER BY clause
5. None of the above

**Solution:**

(B) Column names cannot be used in the WHERE clause of SQL statement.

1. The following listing shows the records of the Books table.

|  |
| --- |
|  |

When you issue the following query, which value will be displayed in the first row?

SELECT B\_Code FROM Books

WHERE Type = ‘HOR’

ORDERBY Title DESC;

1. Magic Terror
2. 9611
3. 189
4. No rows will be returned because Title cannot be used in the ORDER BY clause.

**Solution:**

(C) There are three records falling under the Type HOR; B\_Code 189

(Magic Terror), 1351 (Dreamcatcher) and 9611 (Black House). When the book titles are sorted in descending order, Magic Terror is the first row to be displayed.

1. Refer to the listing of records in Books table in question 11 how many rows the

Following query return?

SELECT \* FROM Books WHERE Title BETWEEN ‘A’ AND ‘D’;

1. 4
2. 2
3. A character column cannot be used in the between operator.
4. 6

**Solution:**

(D) Here, a character column is compared against a string using BETWEEN operator, which is equivalent to Title >= ‘A’ AND Title < ‘D’. Therefor the titles starting with letter D are not included in the result.

1. When you issue the following query, will you encounter any error? If YES then how

Many and why.

SELECT B\_Code AS “Book Number”, Title AS “Book Name”

FROM Books

WHERE Type =’HOR’ AND “Book Number” = 180

ORDER BY “Book Number”

1. 1
2. 3
3. 4
4. No error, the statement will finish successfully.

**Solution:**

(C) Because column alias names cannot be used in the WHERE clause but they can be used in ORDER BY clause.

1. Write a query to display the all the books name whose third character is ‘*e*’.

**Solution:**

SELECT Title FROM Books WHERE Title LIKE ‘\_\_e%’;

**Output:**

|  |
| --- |
|  |

1. Display all the book titles, type and price whose Publisher is not BY, VB, PB and

whose price equal to £5, £14 or £24.

**Solution:**

SELECT Title, Type, Price FROM Books

WHERE P\_Code NOT IN (‘BY’,’VB’,’PB’) AND Price IN (5,14,24);

**Output:**

|  |
| --- |
|  |

## UNSOLVED PRACTICE QUESTIONS

**Note: For the unsolved practice question PRODUCT database is used which available in Annexure A.**

**Practice Set – 2.2**

1. Display all products where the second character of the product name is *e*.
2. Display the names and codes of all the suppliers who supplies *Glasses*.
3. Display the product name and Price of the product delivered to department ‘Toy’.
4. List the name of all the products whose name starts with *B*  and arrange the result in the

descending order of their colours.

1. Display the products supplied by S4.
2. Display the list of products to be delivered in the descending order of their ordered quantity.
3. Display the product’s detail whose colour is Brown.
4. Display the list of products to be delivered in department ‘Clothes’.
5. Display the list of products to be delivered where quantity is greater than 10 and price is greater than 250.
6. Arrange the above question 9 in the order of their suppliers.

\*\*\* Chapter Ends \*\*\*