Chapter 2

Restricting and Sorting Data

Objectives

After completing this lesson, you should be able to do the following:

•Limit the rows retrieved by a query

•Sort the rows retrieved by a query

Lesson Aim

While retrieving data from the database, you may need to restrict the rows of data that are display
or specify the order in which the rows are displayed. This lesson explains the SQL statements that
you will use to perform these actions.

Limiting Rows Using a Selection

```
SELECT *
FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/1980	800		20
7499	ALLEN	SALESMAN	7698	20/02/1981	1600	300	30
7521	WARD	SALESMAN	7698	22/02/1981	1250	500	30
7566	JONES	MANAGER	7839	02/04/1981	2975		20
7654	MARTIN	SALESMAN	7698	28/09/1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/1981	2850		30
7782	CLARK	MANAGER	7839	09/06/1981	2450		10
7788	SCOTT	ANALYST	7566	09/12/1982	3000		20
7839	KING	PRESIDENT		17/11/1981	5000		10
7844	TURNER	SALESMAN	7698	08/09/1981	1500	0	30
7876	ADAMS	CLERK	7788	12/01/1983	1100		20
7900	JAMES	CLERK	7698	03/12/1981	950		30
7902	FORD	ANALYST	7566	03/12/1981	3000		20
7934	MILLER	CLERK	7782	23/01/1982	1300		10

14 rows selected.

```
SELECT *
FROM emp
WHERE deptno=10;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	09/06/1981	2450		10
7839	KING	PRESIDENT		17/11/1981	5000		10
7934	MILLER	CLERK	7782	23/01/1982	1300		10

Limiting Rows Using a Selection

In the example on the slide, assume that you want to display all the employees in department 10. The highlighted set of rows with a value of 10 in DEPTNO column are the only ones returned of restriction is the basis of the WHERE clause in SQL.

Limiting Rows Selected

Restrict the rows returned by using the WHERE clause.

```
SELECT    [DISTINCT] {*| column [alias], ...}
FROM      table
WHERE     condition ;
```

- **The WHERE clause follows the FROM clause**

Limiting Rows Selected

You can restrict the rows returned from the query by using the WHERE clause. A WHERE clause contains a condition that must be met, and it directly follows the FROM clause.

In the syntax:

WHERE	restricts the query to rows that meet a condition
condition	is composed of column names, expressions, constants, and a comparison operator

The WHERE clause can compare values in columns, literal values, arithmetic expressions, or functions. The WHERE clause consists of three elements:

•Column name

•Comparison operator

•Column name, constant, or list of values

Using the WHERE Clause

```
SQL> SELECT  ename ,   job, deptno
     FROM      emp
     WHERE     job = 'CLERK'
```

```
ENAME        JOB          DEPTNO
JAMES        CLERK            30
SMITH        CLERK            20
ADAMS        CLERK            20
MILLER       CLERK            10
```

Using the WHERE clause

In the example, the SELECT statement retrieves the name, job title, and department number of all employees whose job title is CLERK.

Note that the job title CLERK has been specified in uppercase to ensure that the match is made with the job column in the EMP table. Character strings are case sensitive.

Character Strings and Dates

•Character strings and date values are enclosed in single quotation marks.

•Character values are case sensitive and date values are format sensitive.

•The default date format is DD-MON-YY.

```
SQL> SELECT  ename, job, deptno
FROMemp
WHERE     ename  =  'JAMES';
```

Character Strings and Dates

Character strings and dates in the WHERE clause must be enclosed in single quotation marks (' '). Number constants, however, should not.

All character searches are case sensitive. In the following example, no rows are returned because the EMP table stores all the data in uppercase:

```
SQL> SELECT ename,  empno,  job,  deptno
FROM     emp
WHERE    job='clerk';
```

Oracle stores dates in an internal numeric format, representing the century, year, month, day, hours, minutes, and seconds. The default date display is DD-MON-YY.

Note: Changing default date format will be covered in Lesson 3. Number values are not enclosed within quotation marks.

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Comparison Operators

Comparison operators are used in conditions that compare one expression to another.

They are used in the WHERE clause in the following format:

Syntax

```
... WHERE expr operator value
```

Examples

```
... WHERE hiredate='01-JAN-95'
... WHERE sal>=1500
... WHERE ename='SMITH'
```

Using the Comparison Operators

```
SQL> SELECT ename, sal, comm
  FROM   emp
  WHERE  sal<=comm;
```

ENAME	SAL	COMM
MARTIN	1250	1400

Using the Comparison Operators

In the example, the SELECT statement retrieves name, salary, and commission from the EMP table, where the employee salary is less than or equal to the commission amount. Note that there is no explicit value supplied to the WHERE clause. The two values being compared are taken from the SAL and COMM columns in the EMP table.

Other Comparison Operators

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive)
IN(list)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Using the BETWEEN Operator

Use the BETWEEN operator to display rows based on a range of values.

```
SELECT  ename, sal, comm
  FROM     emp
  WHERE  sal BETWEEN  1000 AND 1500 ;
```

ENAME	SAL	COMM
WARD	1250	500
MARTIN	1250	1400
TURNER	1500	0
ADAMS	1100	
MILLER	1300	

The BETWEEN Operator

You can display rows based on a range of values using the BETWEEN operator. The range that you specify contains a lower range and an upper range.

The SELECT statement on the slide returns rows from the EMP table for any employee whose salary is between $1000 and $1500.

Values specified with the BETWEEN operator are inclusive. You must specify the lower limit first.

Using the IN Operator

Use the IN operator to test for values in a list

```
SQL> SELECT  empno, ename, sal, mgr
         FROM     emp
         WHERE    mgr  IN (7902, 7566, 7788);
```

EMPNO	ENAME	SAL	MGR
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788
7902	FORD	3000	7566

The IN Operator

To test for values in a specified list, use the IN operator.

The slide example displays employee number, name, salary, and manager's employee number of all the employees whose manager's employee number is 7902. 7566. or 7788,

The IN operator can be used with any datatype. The following example returns a row from the EMP table for any employee whose name is included in the list of names in the WHERE clause:

```
SQL>  SELECT  empno, ename, mgr, deptno
2FROM     emp
3WHERE    ename IN ( 'FORD' , 'ALLEN' ) ;
```

If characters or dates are used in the list, they must be enclosed in single quotation marks (' ')-

Using the LIKE Operator

Use the LIKE operator to perform wildcard searches of valid search string values.

Search conditions can contain either literal characters or numbers.

1. **% denotes zero or many characters, denotes one character.**
2. **_ denotes one character.**

```
SQL> SELECT     ename
2FROM       emp
3WHERE     ename LIKE '_C%';
```

The LIKE Operator

You may not always know the exact value to search for. You can select rows that match a character pattern by using the LIKE operator. The character pattern-matching operation is referred to as a *wildcard search.* Two svmbols can be used to construct the search string.

Symbol	Description
%	Represents any sequence of zero or more characters
_	Represents any single character

The SELECT statement above returns the employee name from the EMP table for any employee whose name begins with an "S". Note the uppercase "S" . Names beginning with an "s" will not be returned.

The LIKE operator can be used as a shortcut for some BETWEEN comparisons. The following example displays names and hire dates of all employees who joined between January 1981 and December 1981:

```
SQL> SELECT     hiredate
2FROM     emp
3WHERE    hiredate LIKE '%81' ;
```

Using the LIKE Operator

You can combine pattern-matching characters.

```
SQL> SELECT     ename
4FROM     emp
5WHERE    ename LIKE 'A%' ;
```

```
ENAME

MARTIN
JAMES
WARD
```

You can use the ESCAPE identifier to search for "%" or "_".

Combining Wildcard Characters

The % and _ symbols can be used in any combination with literal characters. The example on the slide displays the names of all employees whose name has an "A'" as the second character.

The ESCAPE Option

When you need to have an exact match for the actual '%' and '_' characters, use the ESCAPE option. This option specifies what the ESCAPE character is. To display the names of employees whose name contains ˋA_B\’ use the following SQL statement:

```
SQL> SELECT     ename
2FROM       emp
3WHERE     ename  LIKE  '%A\_B%'  ESCAPE  '\' ;
```

The ESCAPE option identifies the backslash (\) as the escape character. In the pattern, the escape character precedes the underscore (_). This causes the Oracle Server to interpret the underscore literally.

Using the IS NULL Operator

Test for null values with the IS NULL operator.

```
SQL> SELECT    ename,  mgr
2FROM     emp
3WHERE    mgr IS NULL;
```

```
ENAME
              MGR
KING
```

The IS NULL Operator

The IS NULL operator tests for values that are null. A null value means the value is unavailable, unassigned, unknown, or inapplicable. Therefore, you cannot test with (=) because a null value cannot be equal or unequal to any value. The slide example retrieves the name and manager of all employees who do not have a manager.

For example, to display name Job title, and commission for all employees who are not entitled to get a commission, use the following SQL statement.

```
SELECT      ename,    job
     FROM     emp
     WHERE    comm IS NULL;
```

ENAME	JOB
SMITH	CLERK
JONES	MANAGER
BLAKE	MANAGER
CLARK	MANAGER
SCOTT	ANALYST
KING	PRESIDENT
ADAMS	CLERK
JAMES	CLERK
FORD	ANALYST
MILLER	CLERK

10 rows selected.

Logical Operators

Operator	Meaning
AND	Returns TRUE if *both* component conditions are TRUE
OR	Returns TRUE if *either* component condition is TRUE
NOT	Returns TRUE if the following condition is FALSE

Logical Operators

A logical operator combines the result of two component conditions to produce a single result based on them or to invert the result of a single condition. Three logical operators are available in SQL:

- AND
- OR
- NOT

All the examples so far have specified only one condition in the WHERE clause. You can use several conditions in one WHERE clause using the AND and OR operators.

Using the AND Operator

AND requires both conditions to be TRUE

```
SQL> SELECT empno, ename,
job, sal
     FROM   emp
     WHERE  sal >= 1100
     AND    job= 'CLERK' ;
```

EMPNO	ENAME	JOB	SAL
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

The AND Operator

In the example, both conditions must be true for any record to be selected. Therefore, an employee who has a job title of CLERK *and* earns more than $1100 will be selected.

All character searches are case sensitive. No rows are returned if CLERK is not in uppercase. I Character strings must be enclosed in quotation marks.

AND Truth Table

The following table shows the results of combining two expressions with AND:

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOW N	FALSE	UNKNOWN

Using the OR Operator

OR requires either condition to be TRUE

```
SQL> SELECT empno, ename, job, sal
  FROM   emp
  WHERE  sal >= 1100
  OR     job = 'CLERK' ;
```

EMPNO	ENAME	JOB	SAL
7369	SMITH	CLERK	800
7499	ALLEN	SALESMAN	1600
7521	WARD	SALESMAN	1250
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000
7844	TURNER	SALESMAN	1500
7876	ADAMS	CLERK	1100
7900	JAMES	CLERK	950
7902	FORD	ANALYST	3000
7934	MILLER	CLERK	1300

14 rows selected.

The OR Operator

In the example, either condition can be true for any record to be selected. Therefore, an employee who has a job title of CLERK *or* earns more than $1100 will be selected.

The OR Truth Table

The following table shows the results of combining two expressions with OR:

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Using the NOT Operator

```
SQL>  SELECT  ename,   job
            FROM       emp
            WHERE  job NOT IN  ( 'CLERK',  'MANAGER' , 'ANALYST' ) ;
```

ENAME	JOB
ALLEN	SALESMAN
WARD	SALESMAN
MARTIN	SALESMAN
KING	PRESIDENT
TURNER	SALESMAN

The NOT Operator

The slide example displays name and job title of all the employees whose job title *is not* CLERK. MANAGER, or ANALYST.

The NOT Truth Table

The following table shows the result of applying the NOT operator to a condition:

NOT	TRUE	FALSE	UNKNOWN
	FALSE	TRUE	UNKNOWN

Note: The NOT operator can also be used with other SQL operators, such as BETWEEN, LIKE, and
NULL.

```
... WHERE  NOT job IN (' CLERK', 'ANALYST')
... WHERE  sal  NOT  BETWEEN  1000 AND  1500
... WHERE  ename NOT LIKE '%A%'
```

Rules of Precedence

Order Evaluated	Operator
1	All comparison operators
2	NOT
3	AND
4	**OR**

Override rules of precedence by using parentheses.

Rules of Precedence

```
SELECT ename,   job,   sal
  FROM emp
  WHERE  job = 'SALESMAN'  OR  job =  'PRESIDENT ' ;
```

ENAME	JOB	SAL
ALLEN	SALESMAN	1600
WARD	SALESMAN	1250
MARTIN	SALESMAN	1250
TURNER	SALESMAN	1500

Example of Precedence of AND Operator

In the slide example, there are two conditions:

The first condition is that job is PRESIDENT *and* salary is greater than 1500.

The second condition is that job is SALESMAN. Therefore, the SELECT statement reads as follows:

"Select the row if an employee is a PRESIDENT *and* earns more than $1500 *or* if the employee is a SALESMAN"

Rules of Precedence

The parentheses to force priority

```
SELECT  ename, job, sal

FROM   emp

WHERE  (job = 'SALESMAN' OR  job= ' PRESIDENT ' )

AND sal >1500;
```

ENAME	JOB	SAL
ALLEN	SALESMAN	1600

Using Parentheses

In the example, there are two conditions: •   The first condition is that job is PRESIDENT *or* SALESMAN.

The second condition is that salay is greater than 1500. Therefore, the SELECT statement reads as follows:

"Select the row if an employee is a PRESIDENT or a SALESMAN and if the employee earns more than $1500."

ORDER BY Clause

- Sort rows with the ORDER BY clause
  - ASC: ascending order, default
  - DESC: descending order
- The ORDER BY clause comes last in the SELECT statement.

```
SQL> SELECT    ename ,  job,  deptno,  hiredate
     FROM      emp
     ORDER BY hiredate;
```

ENAME	JOB	DEPTNO	HIREDATE
SMITH	CLERK	20	17/12/1980
ALLEN	SALESMAN	30	20/02/1981
WARD	SALESMAN	30	22/02/1981
JONES	MANAGER	20	02/04/1981
BLAKE	MANAGER	30	01/05/1981
CLARK	MANAGER	10	09/06/1981
TURNER	SALESMAN	30	08/09/1981
MARTIN	SALESMAN	30	28/09/1981
KING	PRESIDENT	10	17/11/1981
JAMES	CLERK	30	03/12/1981
FORD	ANALYST	20	03/12/1981
MILLER	CLERK	10	23/01/1982
SCOTT	ANALYST	20	09/12/1982
ADAMS	CLERK	20	12/01/1983

14 rows selected.

The ORDER BY Clause

The order of rows returned in a query result is undefined. The ORDER BY clause can be used to sort the rows. If you use the ORDER BY clause, you must place last. You can specify an expression or an alias to sort.

Syntax

```
SELECT expr
FROM  table
[WHERE  condition(s) ]
[ORDER BY    {column, expr} [ASC | DESC]];
```

where:

ORDER BY	specifies the order in which the retrieved rows are displayed
ASC	orders the rows in ascending order (this is the default order)
DESC	orders the rows in descending order

If the ORDER BY clause isnot used, the sort order is undefined, and the oracle server may not fetch rows in the same order for the same query twice. Use ORDER BY clause to display the rows in a specified order.

Sorting in Descending Order

```
SELECT  ename, job,   deptno,   hiredate
    FROM    emp
    ORDER  BY  hiredate  DESC;
```

ENAME	JOB	DEPTNO	HIREDATE
ADAMS	CLERK	20	12/01/1983
SCOTT	ANALYST	20	09/12/1982
MILLER	CLERK	10	23/01/1982
FORD	ANALYST	20	03/12/1981
JAMES	CLERK	30	03/12/1981
KING	PRESIDENT	10	17/11/1981
MARTIN	SALESMAN	30	28/09/1981
TURNER	SALESMAN	30	08/09/1981
CLARK	MANAGER	10	09/06/1981
BLAKE	MANAGER	30	01/05/1981
JONES	MANAGER	20	02/04/1981
WARD	SALESMAN	30	22/02/1981
ALLEN	SALESMAN	30	20/02/1981
SMITH	CLERK	20	17/12/1980

14 rows selected.

Default Ordering of Data

The default sort order is ascending:

- Numeric values are displayed with the lowest values first
    - —for example, 1-999.
- Date values are displayed with the earliest value first
    - —for example: 0l-JAN-92 before 0l-JAN-95.
- Character values are displayed in alphabetical order
    - —for example: A first and Z last.
- Null values are displayed last for ascending sequences and first for descending sequences.

Reversing the Default Order

To reverse the order in which rows are displayed, specify the keyword **DESC** after the column name in the ORDER BY clause. The slide example sorts the result by the most recently hired employee.

Sorting by Column Alias

```
SQL> SELECT  empno,  ename,  sal*12  "annsal"
     FROM  emp
     ORDER BY  "annsal";
```

EMPNO	ENAME	annsal
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7521	WARD	15000
7654	MARTIN	15000
7934	MILLER	15600
7844	TURNER	18000
7499	ALLEN	19200
7782	CLARK	29400
7698	BLAKE	34200
7566	JONES	35700
7788	SCOTT	36000
7902	FORD	36000
7839	KING	60000

14 rows selected.

Sorting By Column Aliases

You can use a column alias in the ORDER BY clause. The slide example sorts the data by annual salary.

Sorting by Multiple Columns

• The order of ORDER BY list is the order of sort

```
SELECT ename,  deptno,  sal
    FROM          emp
    ORDER BY  deptno,  sal  DESC;
```

ENAME	DEPTNO	SAL
KING	10	5000
CLARK	10	2450
MILLER	10	1300
SCOTT	20	3000
FORD	20	3000
JONES	20	2975
ADAMS	20	1100
SMITH	20	800
BLAKE	30	2850
ALLEN	30	1600
TURNER	30	1500
MARTIN	30	1250
WARD	30	1250
JAMES	30	950

14 rows selected.

You can sort by a column that is not in the SELECT list.

Sorting by Multiple Columns

You can sort query results by more than one column. The sort limit is the number of columns in the given table.

In the ORDER BY clause, specify the columns, and separate the column names using commas. If you want to reverse the order of a column, specify DESC after its name. You can order by columns that are not included in the SELECT clause.

Example

Display name and salary of all employees. Order the result by department number and then descending order by salary.

```
SQL> SELECT   ename,   sal
       FROM   emp
       ORDER BY  deptno, sal DESC;
```

Summary

```
SELECT  [DISTINCT] {*| column [alias], ...}
FROM    table
[WHERE  condition(s) ]
[ORDER BY    {column,  expr,  alias}  [ASC|DESC]];
```

Summary

In this lesson, you have learned about restricting and sorting rows returned by the SELECT statement. You have also learned how to implement various operators.

Practice Overview

Selecting data and changing the order of rows displayed
Restricting rows by using the WHERE clause
Using the double quotation marks in column aliases

Practice Overview

This practice gives you a variety of exercises using the WHERE clause and the ORDER BY clause.

Practice 2

1. Create a query to display the name and salary of employees earning more than $2850. Save your SQL statement to a file named *p2ql.sql.* Run your query.

ENAME	SAL
JONES	2975
SCOTT	3000
KING	5000
FORD	3000

2. Create a query to display the employee name and department number for employee number 7566.

ENAME	DEPTNO
JONES	20

3. Modify *p2ql.sql* to display the name and salary for all employees whose salary is not in the range of $1500 and $2850. Resave your SQL statement to a file named *p2q3.sql.* Rerun your query.

ENAME	SAL
SMITH	800
WARD	1250
JONES	2975
MARTIN	1250
SCOTT	3000
KING	5000
ADAMS	1100
JAMES	950
FORD	3000
MILLER	1300

10 rows selected.

4. Display the employee name. job. and start date of employees hired between February 20. 1981, and May 1. 1981. Order the query in ascending order by start date.

ENAME	JOB	HIREDATE
ALLEN	SALESMAN	20/02/1981
WARD	SALESMAN	22/02/1981
JONES	MANAGER	02/04/1981
BLAKE	MANAGER	01/05/1981

5. Display the employee name and department number of all employees in departments 10 and 30 in alphabetical order by name.

```
ENAME          DEPTNO
ALLEN          30
BLAKE          30
CLARK          10
JAMES          30
KING           10
MARTIN         30
MILLER         10
TURNER         30
WARD           30
9 rows         selected
```

6. *Modify p2q3.scj/to* list the name and salary of employees who earn more than $1500 and are in department 10 or 30. Label the columns Employee and Monthly Salary, respectively. Resave your SQL statement to a file named *p2q6.sql* Rerun your query.

```
Employee       Monthly Salary
KING           5000
BLAKE          2850
CLARK          2450
ALLEN          1600
```

Practice 2 (continued)

7. Display the name and hire date of even- employee who was hired in 1982.

```
ENAME               HIREDATE
SCOTT               09-DEC-82
MILLER              23-JAN-82
```

8. Display the name and job title of all employees who do not have a manager.

ENAME	JOB
KING	PRESIDENT

9. Display the name, salary, and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

```
ENAME          SAL           COMM
ALLEN          1600          300"
TURNER         1500          0
MARTIN         1250          1400
WARD           1250          500
```

If you have time, complete the following exercises:

10. Display the names of all employees where the third letter of their name is an *A*.

ENAME

```
BLAKE
CLARK
ADAMS
```

11. Display the name of all employees who have two *Ls* in their name and are in department 30 or their manager is 7782.

ENAME

ALLEN

MILLER

Practice 2 (continued)

If you want extra challenge, complete the following exercises:

12. Display the name, job, and salary for all employees whose job is Clerk or Analyst and their salary is not equal to $1000, $3000. or $5000.

```
ENAME        JOB          SAL

JAMES        CLERK        950
SMITH        CLERK        800
ADAMS        CLERK       1100
MILLER       CLERK       1300
```

13. Modify *p2q6.sql* to display the name, salary, and commission for all employees whose commission amount is greater than their salary increased by 10%. Rerun your query. Resave your query as *p2q!3.sql.*

```
Employee       Monthly Salary      COMM
MARTIN         1250                1400
```