Restricting and Sorting Data

Lesson Objectives

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

- Limit the rows that are retrieved by a query.
- Sort the rows that are retrieved by a query.
- Use ampersand substitution to restrict and sort output at run time.

Lesson Agenda

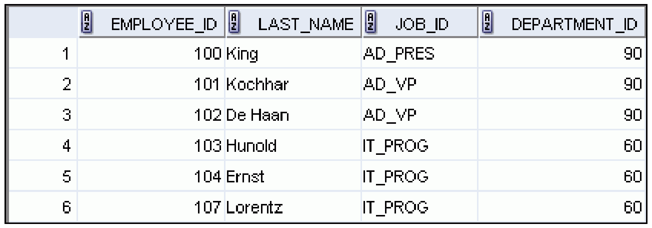
- Limiting rows with:
  - The `WHERE` clause
  - The comparison conditions using `=`, `<=`, `BETWEEN`, `IN`, `LIKE` and `NULL` conditions
  - Logical conditions using `AND`, `OR` and `NOT` operators
- Rules of precedence for operators in an expression
- Sorting rows using the `ORDER BY` clause
- Substitution variables
- `DEFINE` and `UNDEFINE` commands

Lesson Agenda

- **Limiting rows with:**
  - **The `WHERE` clause**
  - The comparison conditions using `=`, `<=`, `BETWEEN`, `IN`, `LIKE` and `NULL` conditions
  - Logical conditions using `AND`, `OR` and `NOT` operators
- Rules of precedence for operators in an expression
- Sorting rows using the `ORDER BY` clause
- Substitution variables
- `DEFINE` and `UNDEFINE` commands

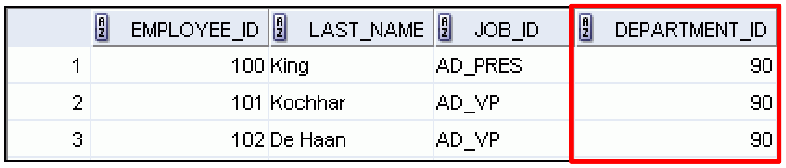
Limiting Rows Using a Selection

Table `EMPLOYEES`



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90
4	103	Hunold	IT_PROG	60
5	104	Ernst	IT_PROG	60
6	107	Lorentz	IT_PROG	60

Retrieve all employees in department id 90



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Limiting Rows Using a Selection

- Restrict the rows that are returned by using the `WHERE` clause:

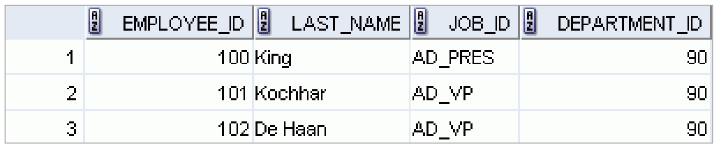
```
SELECT   *|{[DISTINCT]  column|expression [alias],...}
FROM     table
[WHERE   condition(s)];
```

- The `WHERE` clause follows the `FROM` clause.
- `WHERE` clause can compare values in columns, literal, arithmetic expression or functions. If consists of three elements:
  - Column name
  - Comparison condition
  - Column name, constant or list of values

Using the **`WHERE`** Clause

In the example, the SELECT statement retrieves the employee ID, last name, job ID and department id of all employees who are in department id 90.

```
SELECT  employee_id, last_name, job_id, department_id
FROM    employees
WHERE   department_id = 90;
```



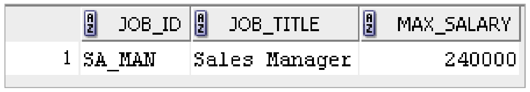
	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

NOTE: You cannot use column alias in the `WHERE` clause.

Using the **`WHERE`** Clause

PRACTICE:

Display the job ID, job title, and maximum salary (per year) from the JOBS table for a job which maximum salary (per year) is equal to $240000. Note that label of columns follow the output:



	JOB_ID	JOB_TITLE	MAX_SALARY
1	SA_MAN	Sales Manager	240000

Character Strings and Dates

- Character strings and data values are enclosed with single quotation marks.
- Character values are case-sensitive and date values are format-sensitive.
- The default date display format is `DD-MON-RR`.

Character Strings and Dates

In the example, the SELECT statement retrieves the last name, job ID and department id for employee who last name is `Whalen`.

```
SELECT  last_name, job_id, department_id
FROM    employees
WHERE   last_name = 'Whalen';
```

LAST_NAME	JOB_ID	DEPARTMENT_ID
Whalen	AD_ASST	10

Character Strings and Dates

PRACTICE:

Display the last name for all employees who were hired in February 17,1996.

***** NOTE: The default date display is in `DD-MON-RR` format.*

LAST_NAME
Hartstein

Lesson Agenda

- **Limiting rows with:**
  - The `WHERE` clause
  - **The comparison conditions using =, <=, BETWEEN, IN, LIKE and NULL conditions**
  - Logical conditions using `AND`, `OR` and `NOT` operators
- Rules of precedence for operators in an expression
- Sorting rows using the `ORDER BY` clause
- Substitution variables
- `DEFINE` and `UNDEFINE` commands

Comparison Operators

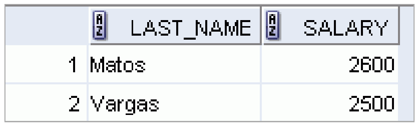
Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Use in WHERE clause.

Using Comparison Operators

In the example, the SELECT statement retrieves the last name from the EMPLOYEES table for any employee whose salary is less than or equal to 3,000.

```
SELECT  last_name, salary
FROM    employees
WHERE   salary <= 3000;
```

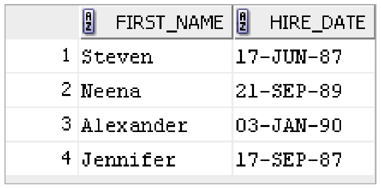


	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

Using Comparison Operators

PRACTICE:

Display the first name, hire date for all employees who were hired before June 1, 1990.

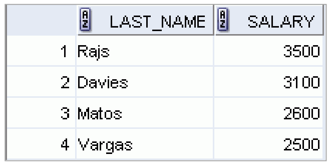


	FIRST_NAME	HIRE_DATE
1	Steven	17-JUN-87
2	Neena	21-SEP-89
3	Alexander	03-JAN-90
4	Jennifer	17-SEP-87

Range Conditions Using the **BETWEEN** Operators

- Use the BETWEEN operator to display rows based on a range of values:
- The SELECT statement in the slide returns rows from the EMPLOYEE table for any employee whose salary is between 2,500 and 3,500.

```
SELECT  last_name, salary
FROM    employees
WHERE   salary BETWEEN 2500 AND 3500;
```



	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

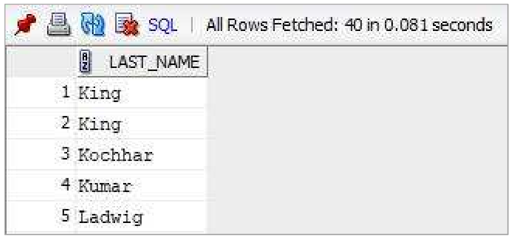
Syntax:

BETWEEN *lower_limit* **AND** *upper_limit*

Range Conditions Using the `BETWEEN` Operators

PRACTICE:

Display the last name for all employees whose last name are between King and Smith.



SQL | All Rows Fetched: 40 in 0.081 seconds

	LAST_NAME
1	King
2	King
3	Kochhar
4	Kumar
5	Ladwig

Display 40 rows.

Membership Condition Using the `IN` Operator

- To test for values in a specified set of values, use the `IN` operator. The condition defined using the `IN` operator is also known as the membership condition.
- If characters or dates are used in the list, they must be enclosed with single quotation marks ('').

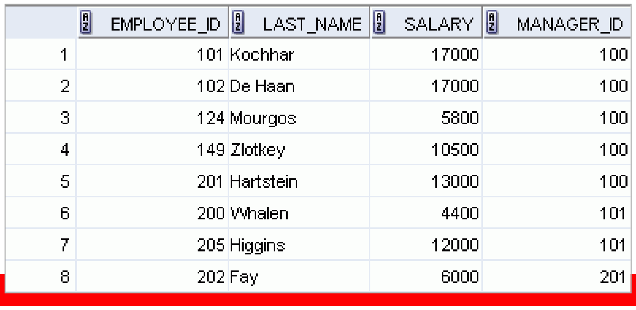
NOTE:

- The `IN` operator is internally evaluated by the Oracle server as a set of `OR` conditions, such as `a=value1` or `a=value2` or `a=value3`.
- Therefore, using the `IN` operator has no performance benefits and is used only for logical simplicity.

Membership Condition Using the `IN` Operator

In the example displays employee id, last names, salaries, and manager's employee id for all the employees whose manager's employee id is 100, 101 or 201.

```
SELECT  employee_id, last_name, salary, manager_id
FROM    employees
WHERE   manager_id IN (100,101,201);
```

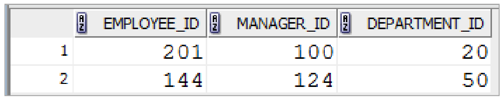


	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

Membership Condition Using the `IN` Operator

PRACTICE:

Display the employee id, manager's employee id and department id for all employees whose last name is `Hartstein` and `Vargas`.

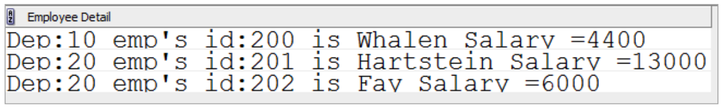


	EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
1	201	100	20
2	144	124	50

Membership Condition Using the `IN` Operator

PRACTICE:

Display follow the output with employees whose department id equal 10 or 20.



Employee Detail

```
Dep:10 emp's id:200 is Whalen Salary =4400
Dep:20 emp's id:201 is Hartstein Salary =13000
Dep:20 emp's id:202 is Fay Salary =6000
```

ORACLE

Pattern Matching 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:
  - % denotes zero or many characters.
  - _ denotes on character.

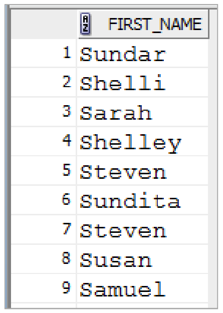
Symbol	Description
%	Represents any sequence of zero or more characters
_	Represents any single character

ORACLE

Pattern Matching Using the `LIKE` Operator

In the example displays the first name for all employees whose first name begins with the letter 'S'.

```
SELECT  first_name
FROM    employees
WHERE   first_name LIKE  'S%';
```



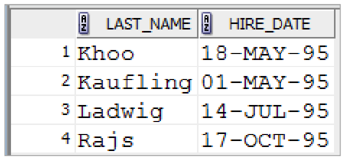
	FIRST_NAME
1	Sundar
2	Shelli
3	Sarah
4	Shelley
5	Steven
6	Sundita
7	Steven
8	Susan
9	Samuel

ORACLE

Pattern Matching Using the `LIKE` Operator

PRACTICE:

Display the last name and hire dates for all employees whose joined between January, 1995 and December, 1995.



	LAST_NAME	HIRE_DATE
1	Khoo	18-MAY-95
2	Kaufling	01-MAY-95
3	Ladwig	14-JUL-95
4	Rajs	17-OCT-95

ORACLE

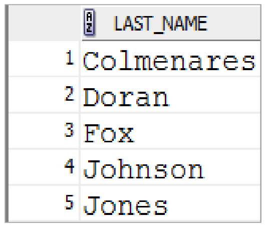
Combining Wildcard Characters

- You can combine the two wildcard characters (%, _) with literal characters for pattern matching.
- You can use the ESCAPE identifier to search for the actual % and _ symbols.

Combining Wildcard Characters

In the example displays the last name with the first letter of last name is anything but the second letter must be letter 'o' and the other letter is anything.

```
SELECT  last_name
FROM    employees
WHERE   last_name LIKE '_o%';
```

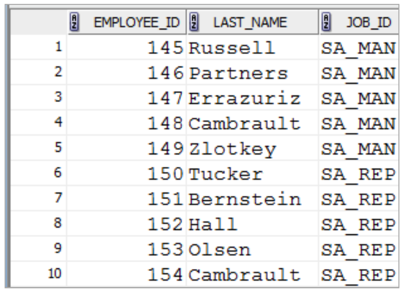


	LAST_NAME
1	Colmenares
2	Doran
3	Fox
4	Johnson
5	Jones

Combining Wildcard Characters

Use the ESCAPE identifier to search for the actual % and _ symbols.

```
SELECT  employee_id, last_name, job_id
FROM    employees
WHERE   job_id LIKE 'SA\_%' ESCAPE '\';
```



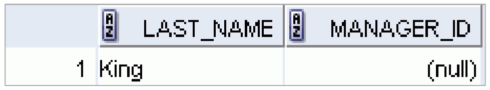
	EMPLOYEE_ID	LAST_NAME	JOB_ID
1	145	Russell	SA_MAN
2	146	Partners	SA_MAN
3	147	Errazuriz	SA_MAN
4	148	Cambrault	SA_MAN
5	149	Zlotkey	SA_MAN
6	150	Tucker	SA_REP
7	151	Bernstein	SA_REP
8	152	Hall	SA_REP
9	153	Olsen	SA_REP
10	154	Cambrault	SA_REP

In the example, want to search for strings that contain SA_

Using the NULL Conditions

- Test for nulls with the IS NULL operator.
- In the example retrieves the last names and managers of all employees who do not have a manger.

```
SELECT  last_name, manager_id
FROM    employees
WHERE   manager_id IS NULL;
```

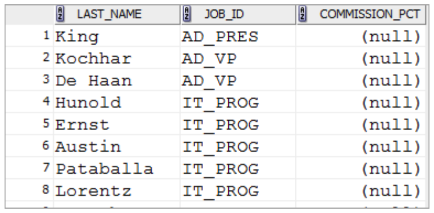


	LAST_NAME	MANAGER_ID
1	King	(null)

Using the NULL Conditions

PRACTICE:

Display the last name, job Id and commission for all employees are not entitled to receive a commission.



	LAST_NAME	JOB_ID	COMMISSION_PCT
1	King	AD_PRES	(null)
2	Kochhar	AD_VP	(null)
3	De Haan	AD_VP	(null)
4	Hunold	IT_PROG	(null)
5	Ernst	IT_PROG	(null)
6	Austin	IT_PROG	(null)
7	Pataballa	IT_PROG	(null)
8	Lorentz	IT_PROG	(null)

Display 72 rows.

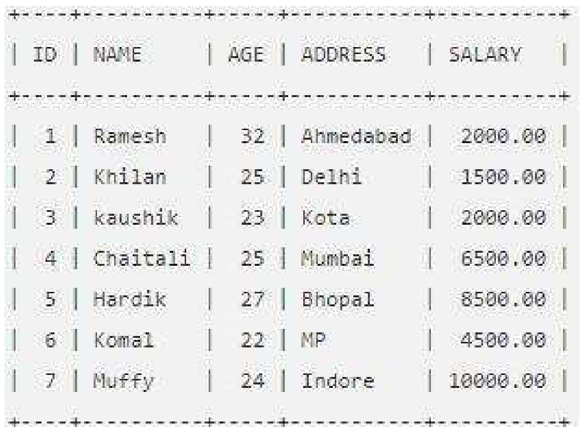
SQL TOP, LIMIT or ROWNUM Clause

- The SQL TOP clause is used to fetch a TOP N number or X percent records from a table.
- The basic syntax of the TOP clause with a SELECT statement would be as follows.

```
SELECT   TOP  number|percent column_name(s)
FROM     table
[WHERE   condition(s)];
```

Example TOP, LIMIT or ROWNUM Clause

Consider the CUSTOMERS table having the following records –

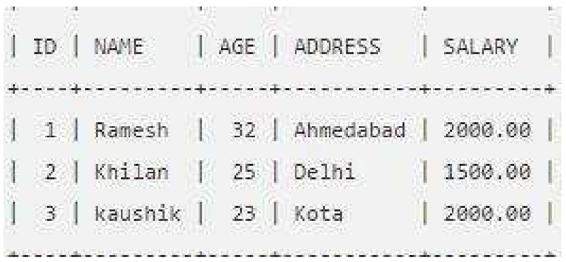


```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Example TOP, LIMIT or ROWNUM Clause

The following query is an example on the SQL server, which would fetch the top 3 records from the CUSTOMERS table.

```
SELECT  TOP 3 *
FROM    customers;
```

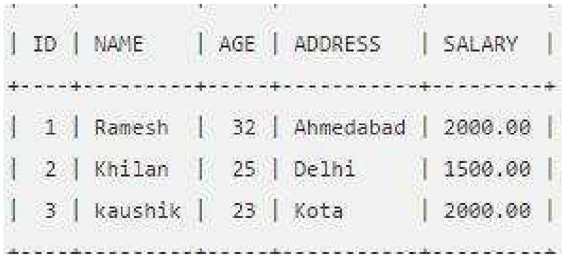


```
| ID | NAME    | AGE | ADDRESS   | SALARY  |
+----+---------+-----+-----------+---------+
|  1 | Ramesh  |  32 | Ahmedabad | 2000.00 |
|  2 | Khilan  |  25 | Delhi     | 1500.00 |
|  3 | kaushik |  23 | Kota      | 2000.00 |
+----+---------+-----+-----------+---------+
```

Example TOP, LIMIT or ROWNUM Clause

If you are using an Oracle server, then the following code block has an equivalent example.

```
SELECT  *
FROM    customers
WHERE   ROWNUM <= 3;
```



```
+----+---------+-----+-----------+---------+
| ID | NAME    | AGE | ADDRESS   | SALARY  |
+----+---------+-----+-----------+---------+
|  1 | Ramesh  |  32 | Ahmedabad | 2000.00 |
|  2 | Khilan  |  25 | Delhi     | 1500.00 |
|  3 | kaushik |  23 | Kota      | 2000.00 |
+----+---------+-----+-----------+---------+
```

Lesson Agenda

- **Limiting rows with:**
  - The WHERE clause
  - The comparison conditions using =, <=, BETWEEN, IN, LIKE and NULL conditions
  - **Logical conditions using AND, OR and NOT operators**
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- Substitution variables
- DEFINE and UNDEFINE commands

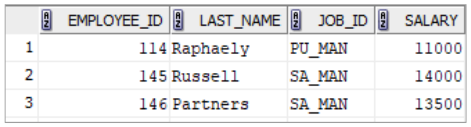
Defining Conditions Using the Logical Operators

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 condition is false

Using the AND Operator

- AND requires both the component conditions to be true:
- In the example, both the component conditions must be true for any record to be selected. Therefore, only those employees who have a job id that contains the string 'MAN' **and** earn $10,000.

```
SELECT  employee_id, last_name, job_id, salary
FROM    employees
WHERE   salary > 10000
AND     job_id LIKE '%MAN%';
```



	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	114	Raphaely	PU_MAN	11000
2	145	Russell	SA_MAN	14000
3	146	Partners	SA_MAN	13500

Display 7 rows.

Using the OR Operator

- OR requires either component condition to be true:
- In the example, either component condition can be true for any record to be selected. Therefore, any employee who has a job ID that contains the string 'MAN' **or** earn $10,000.

```
SELECT  employee_id, last_name, job_id, salary
FROM    employees
WHERE   salary > 10000
OR      job_id LIKE '%MAN%';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	100	King	AD_PRES	24000
2	101	Kochhar	AD_VP	17000
3	102	De Haan	AD_VP	17000

Display 20 rows.

Using the NOT Operator

- NOT operator can also be used with other SQL operators, such as BETWEEN, LIKE, and NULL.
- In the example displays the last name and job ID of all employees whose job ID is **not** IT_PROG, ST_CLERK or SA_REP.

```
SELECT  last_name, job_id
FROM    employees
WHERE   job_id NOT IN ('IT_PROG','ST_CLERK','SA_REP');
```

	LAST_NAME	JOB_ID
1	King	AD_PRES
2	Kochhar	AD_VP
3	De Haan	AD_VP

Display 52 rows.

Defining Conditions Using the Logical Operators

PRACTICE:

Display the last name, job ID, and commission for all employees are not entitled to receive a commission and any employee who has a job ID that contains the string 'AD_'.

	LAST_NAME	JOB_ID	COMMISSION_PCT
1	King	AD_PRES	(null)
2	Kochhar	AD_VP	(null)
3	De Haan	AD_VP	(null)

Lesson Agenda

- **Limiting rows with:**
  - The WHERE clause
  - The comparison conditions using =, <=, BETWEEN, IN, LIKE and NULL conditions
  - Logical conditions using AND, OR and NOT operators
- **Rules of precedence for operators in an expression**
- Sorting rows using the ORDER BY clause
- Substitution variables
- DEFINE and UNDEFINE commands

Rules of Precedence

Operator	Meaning
1	Arithmetic operators(นิพจน์ทางคณิตศาสตร์)
2	Concatenation operator (การเชื่อมข้อความ)
3	Comparison conditions (เครื่องหมายทางการเปรียบเทียบ)
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Not equal to
7	NOT logical condition
8	AND logical condition
9	OR logical condition

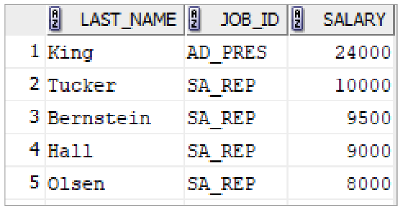
Override rules of precedence by using parentheses.

ORACLE

Rules of Precedence

Precedence of the AND operator: Example

```
SELECT  last_name, job_id, salary
FROM    employees
WHERE   job_id = 'SA_REP'
OR      job_id = 'AD_PRES'
AND     salary > 15000;
```



	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	Tucker	SA_REP	10000
3	Bernstein	SA_REP	9500
4	Hall	SA_REP	9000
5	Olsen	SA_REP	8000

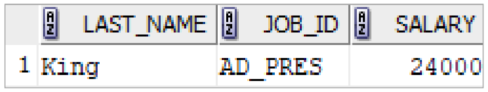
- The first condition is that the job id is AD_PRESS and the salary is greater than 15000.
- The second condition is that the job id is SA_REP

ORACLE

Rules of Precedence

Using Parentheses: Example

```
SELECT  last_name, job_id, salary
FROM    employees
WHERE   (job_id = 'SA_REP'
OR      job_id = 'AD_PRES')
AND     salary > 15000;
```



	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000

- The first condition is that the job id is AD_PRESS or SA_REP
- The second condition is that the salary is greater than 15000.

ORACLE

Lesson Agenda

- **Limiting rows with:**
  - The WHERE clause
  - The comparison conditions using =, <=, BETWEEN, IN, LIKE and NULL conditions
  - Logical conditions using AND, OR and NOT operators
- Rules of precedence for operators in an expression
- **Sorting rows using the ORDER BY clause**
- Substitution variables
- DEFINE and UNDEFINE commands

ORACLE

Using the ORDER BY Clause

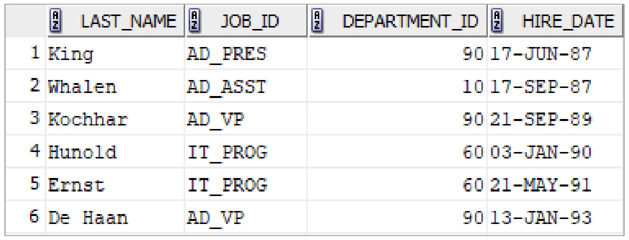
- Sort retrieved rows with the ORDER BY clause:
  - ASC Ascending order, default
  - DESC Descending order
- The ORDER BY clause comes last in the SELECT statement:

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

Sorting

- Sorting in ascending order:
- In the slide example sort the result by the hire_date.

```
SELECT    last_name, job_id, department_id, hire_date
FROM      employees
ORDER BY hire_date ASC;
```



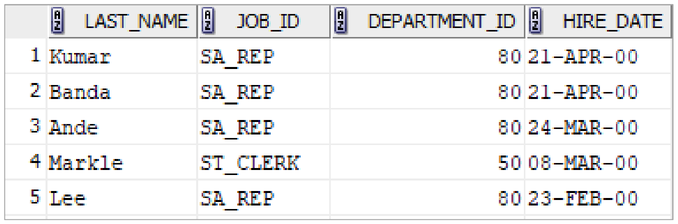
	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

Display 107 rows.

Sorting

- Sorting in descending order:
- In the slide example sort the result by the most recently hired employee.

```
SELECT    last_name, job_id, department_id, hire_date
FROM      employees
ORDER BY hire_date DESC;
```



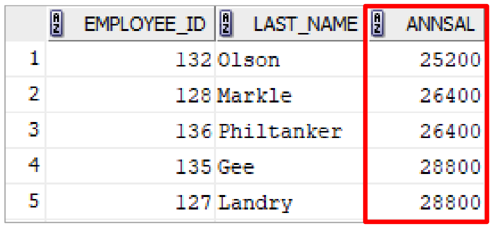
	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	Kumar	SA_REP	80	21-APR-00
2	Banda	SA_REP	80	21-APR-00
3	Ande	SA_REP	80	24-MAR-00
4	Markle	ST_CLERK	50	08-MAR-00
5	Lee	SA_REP	80	23-FEB-00

Display 107 rows.

Sorting

- Sorting by column alias:
- The slide example sorts the data by annual salary.

```
SELECT    employee_id, last_name, salary*12 annsal
FROM      employees
ORDER BY annsal;
```



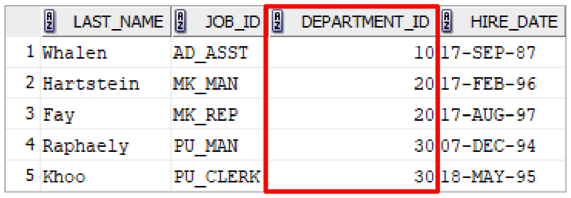
	EMPLOYEE_ID	LAST_NAME	ANNSAL
1	132	Olson	25200
2	128	Markle	26400
3	136	Philtanker	26400
4	135	Gee	28800
5	127	Landry	28800

Display 107 rows.

Sorting

- Sorting by using the column's numeric position.
- The slide example display the last names and salaries of all employees. Order the result by department number, and then in descending order by salary.

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY 3;
```



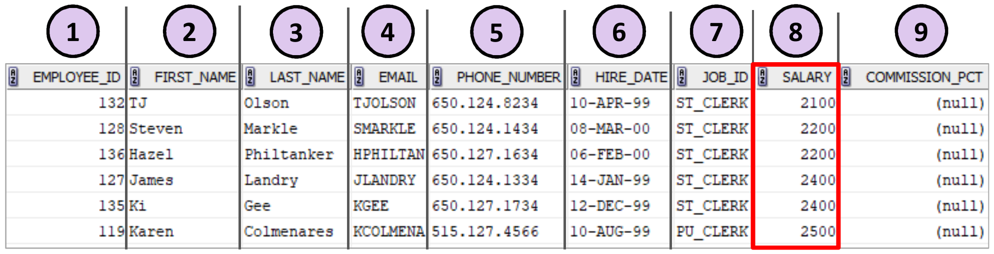
	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	Whalen	AD_ASST	10	17-SEP-87
2	Hartstein	MK_MAN	20	17-FEB-96
3	Fay	MK_REP	20	17-AUG-97
4	Raphaely	PU_MAN	30	07-DEC-94
5	Khoo	PU_CLERK	30	18-MAY-95

Display 107 rows.

Sorting

Sorting by using the column's numeric position.

```
SELECT   *
FROM     employees
ORDER BY 8;
```



1	2	3	4	5	6	7	8	9
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
132	TJ	Olson	TJOLSON	650.124.8234	10-APR-99	ST_CLERK	2100	(null)
128	Steven	Markle	SMARKLE	650.124.1434	08-MAR-00	ST_CLERK	2200	(null)
136	Hazel	Philtanker	HPHILTAN	650.127.1634	06-FEB-00	ST_CLERK	2200	(null)
127	James	Landry	JLANDRY	650.124.1334	14-JAN-99	ST_CLERK	2400	(null)
135	Ki	Gee	KGEE	650.127.1734	12-DEC-99	ST_CLERK	2400	(null)
119	Karen	Colmenares	KCOLMENA	515.127.4566	10-AUG-99	PU_CLERK	2500	(null)

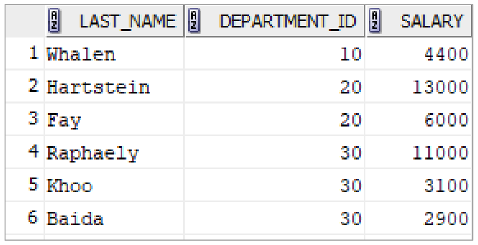
Sorting

- 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 also order by columns are not included in the SELECT clause.

Sorting

In the example display the last names and salaries of all employees. Order the result by department number, and then in descending order by salary.

```
SELECT   last_name, department_id, salary
FROM     employees
ORDER BY department_id, salary DESC;
```



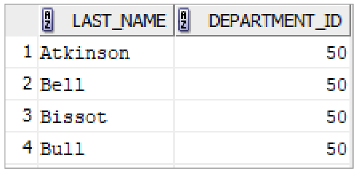
	LAST_NAME	DEPARTMENT_ID	SALARY
1	Whalen	10	4400
2	Hartstein	20	13000
3	Fay	20	6000
4	Raphaely	30	11000
5	Khoo	30	3100
6	Baida	30	2900

- First, sorting department id in ASC.
- Second, sorting salary in DESC.

Sorting

PRACTICE:

Display the last name and department ID of all employees in departments 20 or 50 in ascending alphabetical order by name.



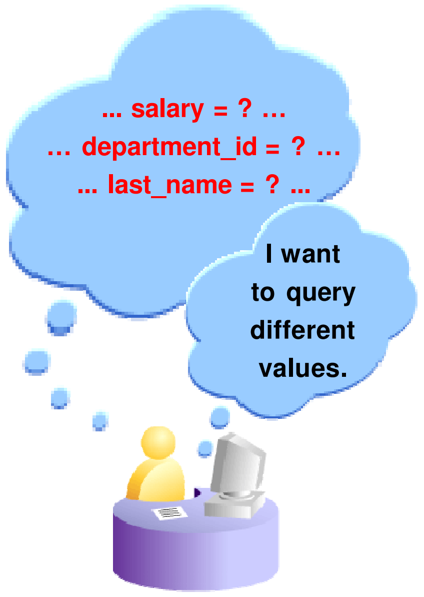
	LAST_NAME	DEPARTMENT_ID
1	Atkinson	50
2	Bell	50
3	Bissot	50
4	Bull	50

Display 47 rows.

Lesson Agenda

- **Limiting rows with:**
  - The `WHERE` clause
  - The comparison conditions using `=`, `<=`, `BETWEEN`, `IN`, `LIKE` and `NULL` conditions
  - Logical conditions using `AND`, `OR` and `NOT` operators
- Rules of precedence for operators in an expression
- Sorting rows using the `ORDER BY` clause
- **Substitution variables**
- `DEFINE` and `UNDEFINE` commands

Substitution Variables



Substitution Variables

- Use substitution variables to:
  - Temporarily store values with **single-ampersand** (**&**) and **double-ampersand** (**&&**) substitution.
- Use substitution variables to supplement the following:
  - `WHERE` conditions
  - `ORDER BY` clause
  - Column expressions
  - Table names
  - Entire `SELECT` statements

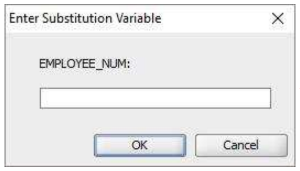
Using the Single-Ampersand Substitution Variable

- Use a variable prefixed with an ampersand (**&**) to prompt the user for a value.
- Restricted Ranges of Data: Examples
  - Reporting figures only for the current quarter of specified date range
  - Reporting on data relevant only to the user requesting the report
  - Displaying personnel only within a given department

Using the Single-Ampersand Substitution Variable

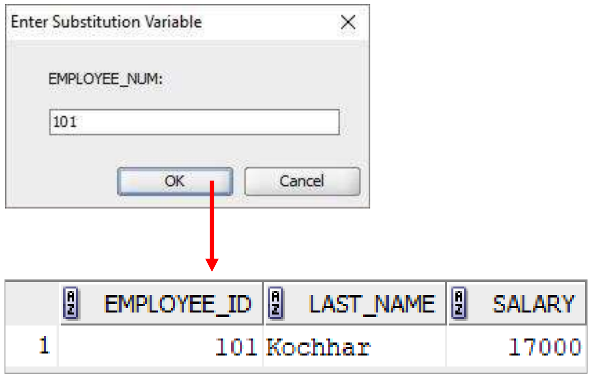
- Use a variable prefixed with an ampersand (**&**) to prompt the user for a value:
- The example in the slide creates a substitution variable for an employee number. When the statement is executed, SQL Developer prompts the user for an employee number and then displays the employee number, last name, salary, and department number for that employee.

```
SELECT    employee_id, last_name, salary
FROM      employees
WHERE     employee_id = &employee_num;
```



Using the Single-Ampersand Substitution Variable

After you enter a value and click the OK button, the result are displayed in the Results tab of your SQL Developer session.



EMPLOYEE_NUM: 101

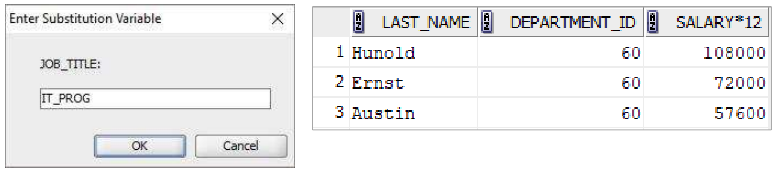
	EMPLOYEE_ID	LAST_NAME	SALARY
1	101	Kochhar	17000

Character and Date Values with Substitution Variables

Use single quotation marks for date and character values:

- In a `WHERE` clause, date and character values must be enclosed with single quotation marks.

```
SELECT    last_name, department_id, salary*12
FROM      employees
WHERE     job_id = '&job_title';
```



JOB_TITLE: IT_PROG

	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Austin	60	57600

Display 5 rows.

The example shows a query to retrieve the employee names, department numbers, and annual salaries of all employees based on the job title value of the SQL Developer substitution variable.

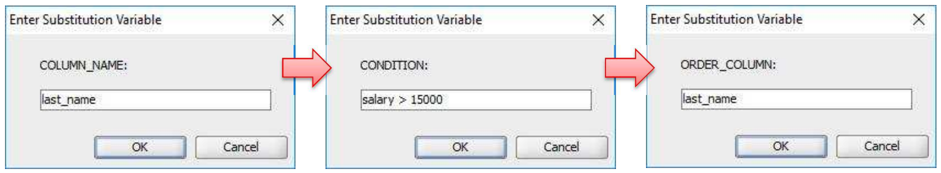
Specifying Column Names, Expressions, and Text

- You can use the substitution variables not only in the WHERE clause of a SQL statement, but also as substitution for column names, expressions, or text.
- **Example:**
  - The slide example displays the employee number, last name, job title, and any other column that is specified by the user at run time, from the `EMPLOYEES` table. For each substitution variable in the `SELECT` statement, you are prompted to enter a value, and then click OK to proceed.
  - If you do not enter a value for the substitution variable, you get an error when you execute the preceding statement.

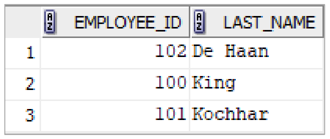
Note:
A substitution variable can be used anywhere in the `SELECT` statement, except as the first word entered at the command prompt.

Specifying Column Names, Expressions, and Text

```
SELECT    employee_id, &column_name
FROM      employees
WHERE     &condition
ORDER BY  &order_column;
```



COLUMN_NAME: last_name
CONDITION: salary > 15000
ORDER_COLUMN: last_name

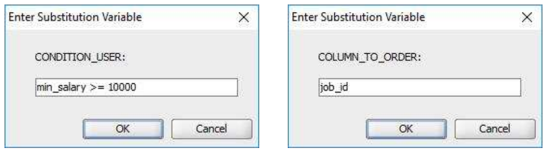


	EMPLOYEE_ID	LAST_NAME
1	102	De Haan
2	100	King
3	101	Kochhar

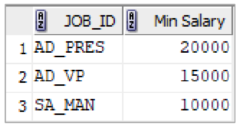
Specifying Column Names, Expressions, and Text

PRACTICE:

Display job ID, minimum salary ,and any column that is specified by the user condition, and and order column name by ascending.



CONDITION_USER: min_salary >= 10000
COLUMN_TO_ORDER: job_id



	JOB_ID	Min Salary
1	AD_PRES	20000
2	AD_VP	15000
3	SA_MAN	10000

Using the Double-Ampersand Substitution Variable

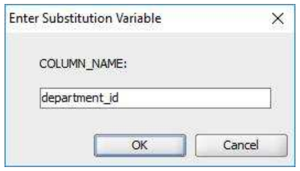
- Use double-ampersand (**&&**) if you want to reuse the variable value without prompting the user each time.
- The user sees the prompt for the vale only once.
- After a user variable is in place, you nee to sue the `UNDEFINE` command to delete it:

```
UNDEFINE column_name;
```

Using the Double-Ampersand Substitution Variable

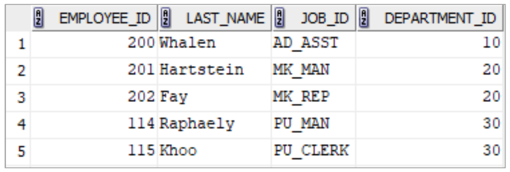
In the example in the slide, the user is asked to give the value for the variable, `column_name`, only one. The value that is supplied by the user (department_id) is used for both display and ordering of data.

```
SELECT    employee_id, last_name, job_id,
          &&column_name
FROM      employees
ORDER BY  &column_name;
```


Enter Substitution Variable

COLUMN_NAME:

department_id



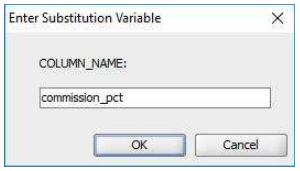
	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20
4	114	Raphaely	PU_MAN	30
5	115	Khoo	PU_CLERK	30

Display 107 rows.

Using the Double-Ampersand Substitution Variable

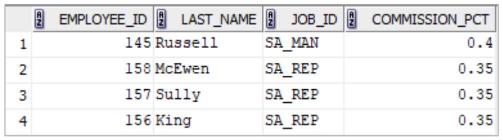
PRACTICE:

Display employee number, last name, job ID ,and the column name is not null and ordering of data by descending.


Enter Substitution Variable

COLUMN_NAME:

commission_pct



	EMPLOYEE_ID	LAST_NAME	JOB_ID	COMMISSION_PCT
1	145	Russell	SA_MAN	0.4
2	158	McEwen	SA_REP	0.35
3	157	Sully	SA_REP	0.35
4	156	King	SA_REP	0.35

Display 35 rows.

Lesson Agenda

- **Limiting rows with:**
  - The WHERE clause
  - The comparison conditions using =, <=, BETWEEN, IN, LIKE and NULL conditions
  - Logical conditions using AND, OR and NOT operators
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- Substitution variables
- **DEFINE and UNDEFINE commands**

Using the DEFINE and UNDEFINE Command

- Use **DEFINE** command to create and assign a value to a variable.
- Use the **UNDEFINE** command to remove a variable.

```
DEFINE  employee_num = 200;
SELECT  employee_id, last_name, salary, department_id
FROM    employees
WHERE   employee_id = &employee_num;
```

Press F5 to execute

```
old:SELECT    employee_id, last_name, salary, department_id
FROM      employees
WHERE   employee_id = &employee_num
new:SELECT    employee_id, last_name, salary, department_id
FROM      employees
WHERE   employee_id = 200
EMPLOYEE_ID            LAST_NAME                SALARY                DEPARTMENT_ID
---------------------- ------------------------ --------------------- ----------------------
200                    Whalen                   4400                  10
```

Quiz

Which of the following are valid operators for the WHERE clause?

- **>=**
- **IS NULL**
- **!=**
- **IS LIKE**
- **IN BETWEEN**
- **<>**

Summary

- In this lesson, you should have learned how to:
- Use the WHERE clause to restrict rows of output
  - Use the comparison conditions
  - Use the BETWEEN, IN, LIKE, and NULL conditions
  - Apply the logical AND, OR, and NOT operators
- Use the ORDER BY clause to sort rows of output

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

- Use ampersand substitution to restrict and sort output at run time.