



Restricting and Sorting Data

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 `VERIFY` commands


Limiting Rows Using a Selection

EMPLOYEES

	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	205	Higgins	AC_MGR	110
5	206	Gietz	AC_ACCOUNT	110

...

**“retrieve all
employees in
department 90”**



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

Limiting the Rows That Are Selected

- 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.

Using the WHERE Clause

```
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

Character Strings and Dates

- Character strings and date 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.

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

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-FEB-96' ;
```

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

Using Comparison Operators

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

	A Z	LAST_NAME	A Z	SALARY
1		Matos		2600
2		Vargas		2500

Range Conditions Using the BETWEEN Operator

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

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

Lower limit

Upper limit

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

Membership Condition Using the IN Operator

Use the IN operator to test for values in a list:

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

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 one character.

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

Combining Wildcard Characters

- You can combine the two wildcard characters (% , _) with literal characters for pattern matching:

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

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

- You can use the `ESCAPE` identifier to search for the actual % and _ symbols.

Using the NULL Conditions

Test for nulls with the IS NULL operator.

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

	LAST_NAME	MANAGER_ID
1	King	(null)

Defining Conditions Using the Logical Operators

Operator	Meaning
AND	Returns TRUE if <i>both</i> component conditions are true
OR	Returns TRUE if <i>either</i> 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:

```
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	201	Hartstein	MK_MAN	13000
2	149	Zlotkey	SA_MAN	10500

Using the OR Operator

OR requires either component condition to be true:

```
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	201	Hartstein	MK_MAN	13000
2	205	Higgins	AC_MGR	12000
3	100	King	AD_PRES	24000
4	101	Kochhar	AD_VP	17000
5	102	De Haan	AD_VP	17000
6	124	Mourgos	ST_MAN	5800
7	149	Zlotkey	SA_MAN	10500
8	174	Abel	SA_REP	11000

Using the NOT Operator

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

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

Lesson Agenda

- Limiting rows with:
 - The `WHERE` clause
 - The comparison conditions using `=`, `<=`, `BETWEEN`, `IN`, `LIKE`, and `NULL` operators
 - 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 `VERIFY` commands

Rules of Precedence

Operator	Meaning
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
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

You can use parentheses to override rules of precedence.

Rules of Precedence

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

1

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	Abel	SA_REP	11000
3	Taylor	SA_REP	8600
4	Grant	SA_REP	7000

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

2

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000

Lesson Agenda

- Limiting rows with:
 - The `WHERE` clause
 - The comparison conditions using `=`, `<=`, `BETWEEN`, `IN`, `LIKE`, and `NULL` operators
 - 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 `VERIFY` commands

Using the ORDER BY Clause

- Sort the 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    last_name, job_id, department_id, hire_date
FROM      employees
ORDER BY  hire_date ;
```


	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

...

Sorting


- Sorting in descending order:

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



- Sorting by column alias:


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



Sorting


- Sorting by using the column's numeric position:

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



- Sorting by multiple columns:

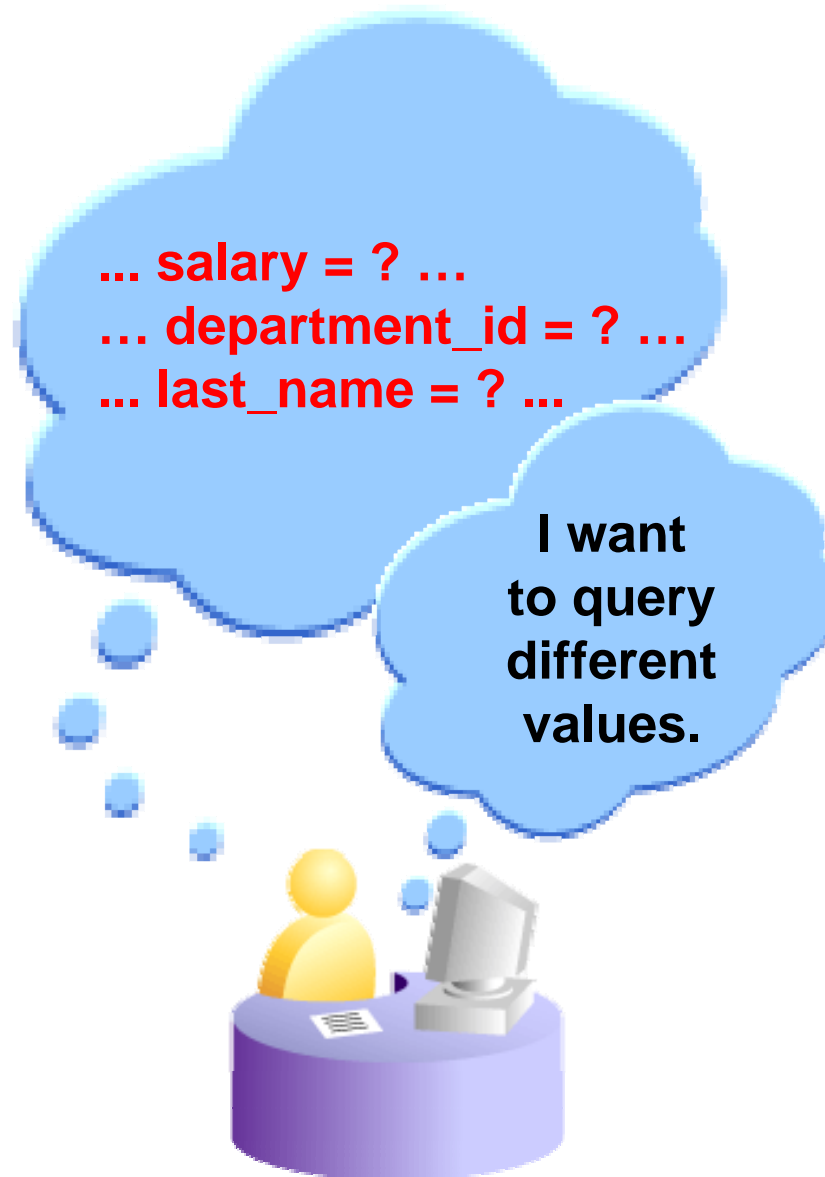
```
SELECT last_name, department_id, salary
FROM    employees
ORDER BY department_id, salary DESC;
```



Lesson Agenda

- Limiting rows with:
 - The `WHERE` clause
 - The comparison conditions using `=`, `<=`, `BETWEEN`, `IN`, `LIKE`, and `NULL` operators
 - 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 `VERIFY` commands

Substitution Variables



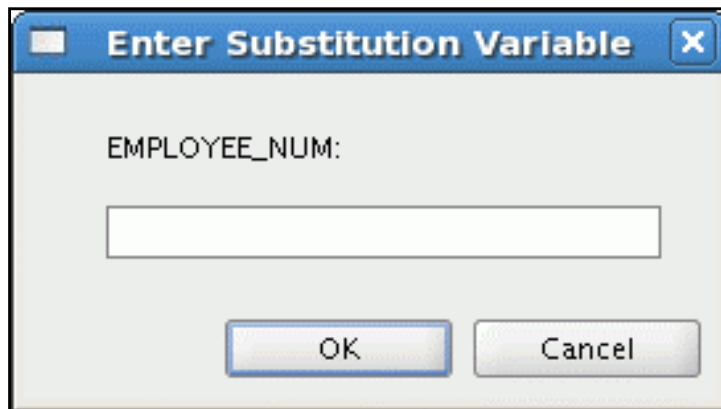
Substitution Variables

- 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 clauses
 - Column expressions
 - Table names
 - Entire SELECT statements

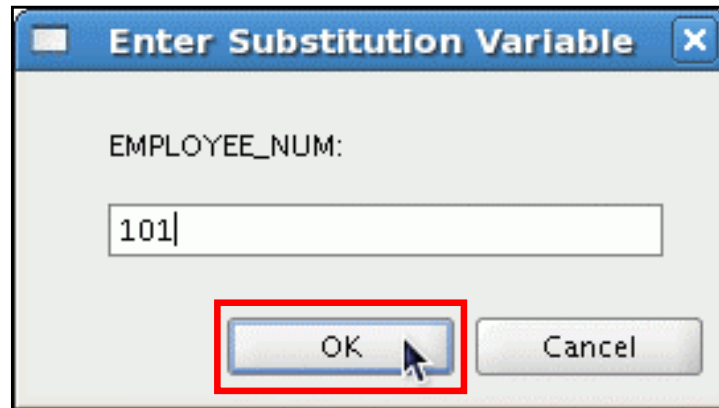
Using the Single-Ampersand Substitution Variable

Use a variable prefixed with an ampersand (&) to prompt the user for a value:





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



Using the Single-Ampersand Substitution Variable



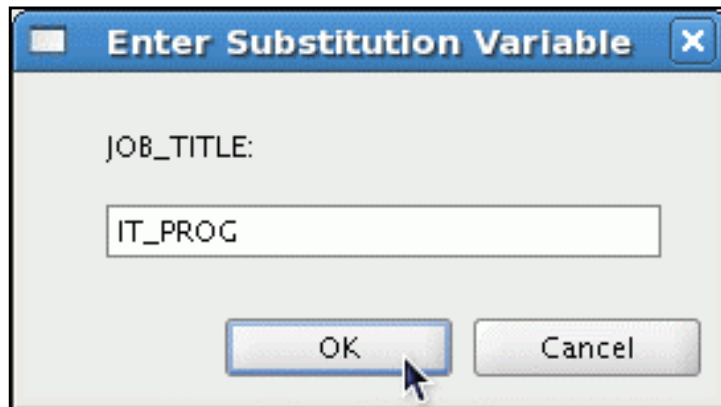
A dialog box titled "Enter Substitution Variable" with a close button (X) in the top right corner. The text "EMPLOYEE_NUM:" is displayed above a text input field containing the value "101". Below the input field, there are two buttons: "OK" and "Cancel". The "OK" button is highlighted with a red rectangular border, and a mouse cursor is pointing at it.

	 EMPLOYEE_ID	 LAST_NAME	 SALARY	 DEPARTMENT_ID
1	101	Kochhar	17000	90

Character and Date Values with Substitution Variables

Use single quotation marks for date and character values:

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

A dialog box titled "Enter Substitution Variable" with a close button (X) in the top right corner. Inside the dialog, the label "JOB_TITLE:" is followed by a text input field containing the value "IT_PROG". At the bottom of the dialog, there are two buttons: "OK" and "Cancel". A mouse cursor is pointing at the "OK" button.

JOB_TITLE:

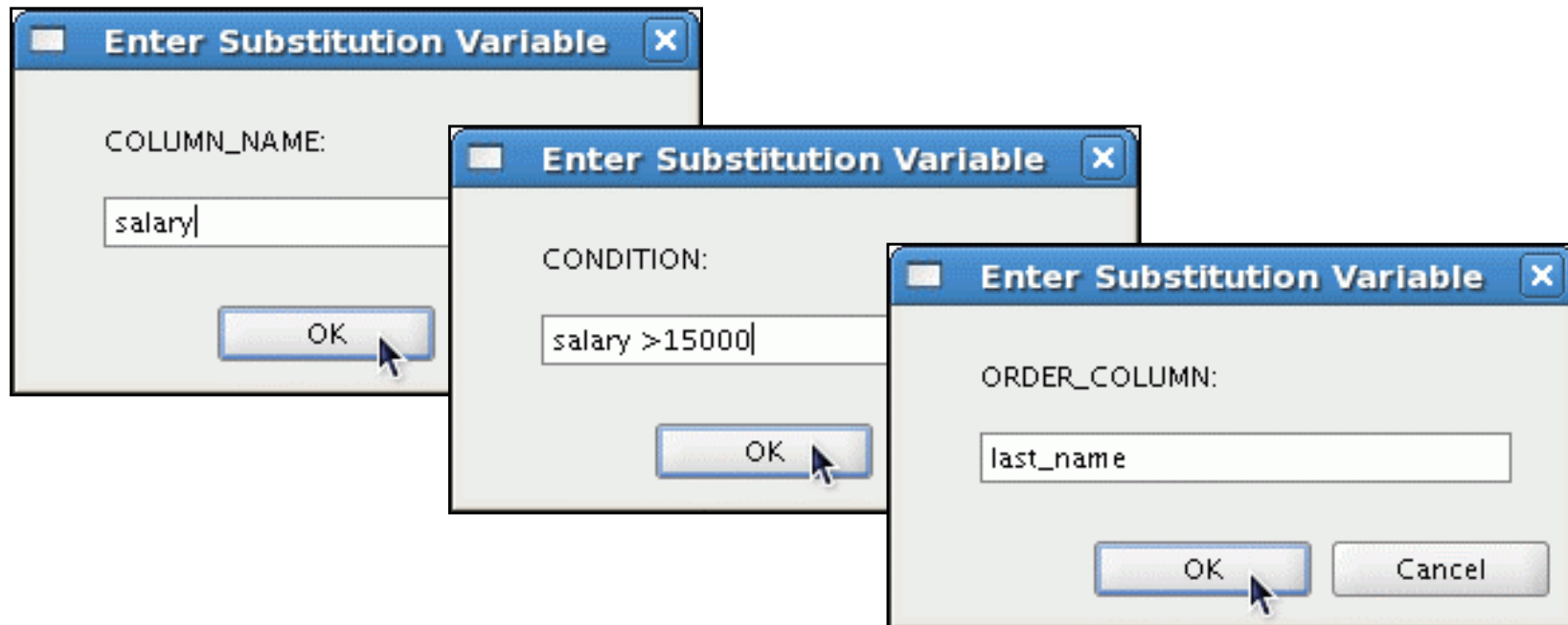
IT_PROG

OK Cancel

	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

Specifying Column Names, Expressions, and Text

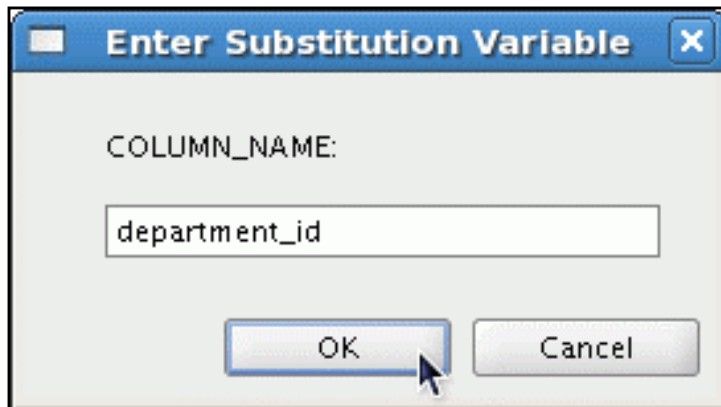
```
SELECT employee_id, last_name, job_id, &column_name  
FROM employees  
WHERE &condition  
ORDER BY &order_column ;
```



Using the Double-Ampersand Substitution Variable

Use double ampersand (&&) if you want to reuse the variable value without prompting the user each time:

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



A dialog box titled "Enter Substitution Variable" with a close button (X). It contains a label "COLUMN_NAME:" and a text input field containing "department_id". At the bottom are "OK" and "Cancel" buttons. A mouse cursor is pointing at the "OK" button.

	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

...

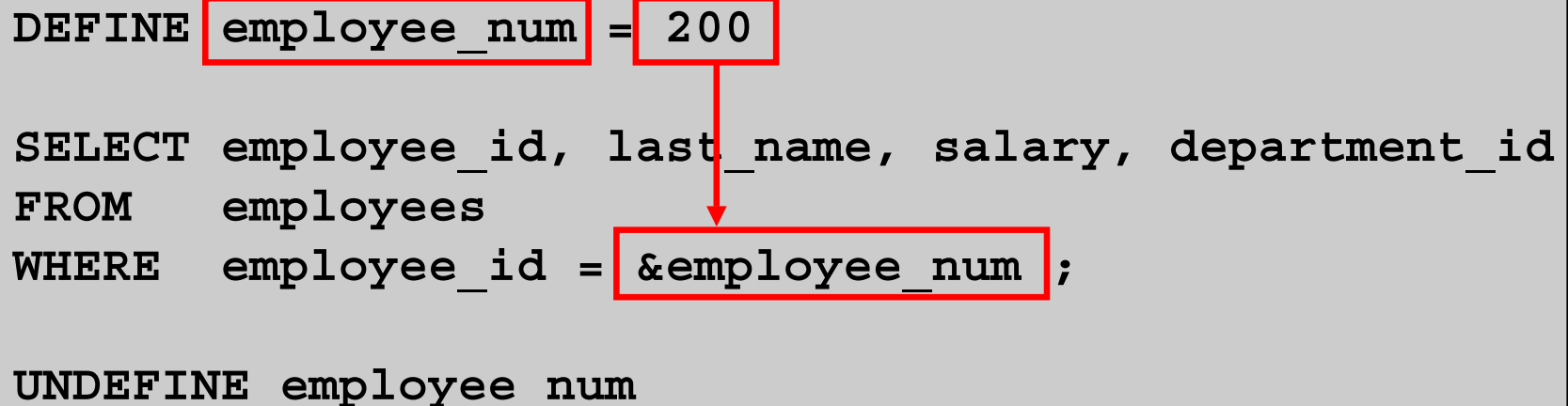
Lesson Agenda

- Limiting rows with:
 - The `WHERE` clause
 - The comparison conditions using `=`, `<=`, `BETWEEN`, `IN`, `LIKE`, and `NULL` operators
 - 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 VERIFY commands**

Using the DEFINE Command

- Use the 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;  
  
UNDEFINE employee_num
```

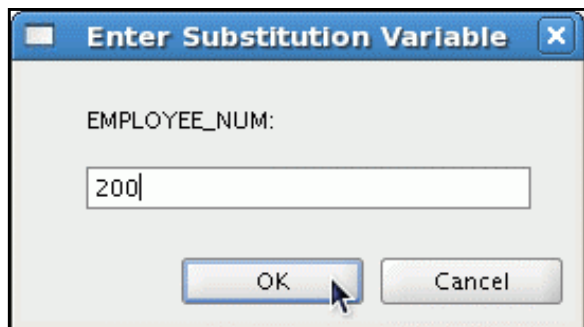
A red arrow points from the value '200' in the DEFINE statement to the '&employee_num' placeholder in the WHERE clause of the SQL query, illustrating how the variable's value is substituted.

Using the VERIFY Command

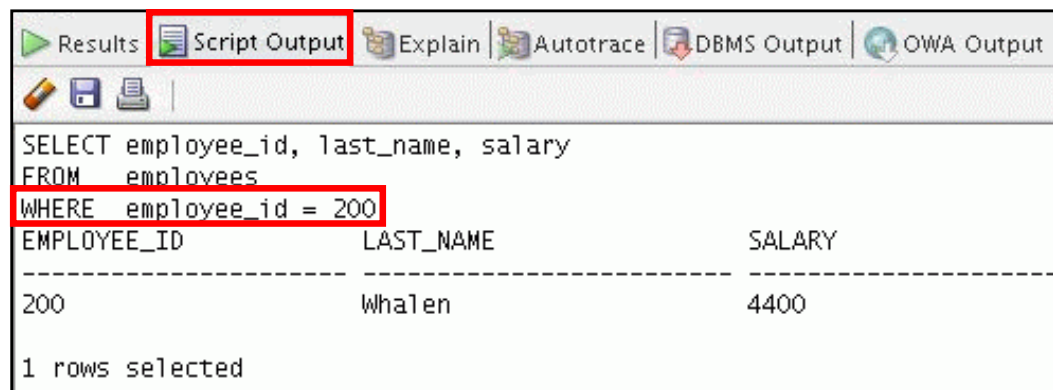
Use the VERIFY command to toggle the display of the substitution variable, both before and after SQL Developer replaces substitution variables with values:

```
SET VERIFY ON
```

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



A dialog box titled "Enter Substitution Variable" with a close button (X). It contains a label "EMPLOYEE_NUM:" followed by a text input field containing the value "200". At the bottom, there are "OK" and "Cancel" buttons. A mouse cursor is pointing at the "OK" button.



The "Script Output" window in SQL Developer shows the execution of the SQL query. The query text is displayed, with the substitution variable replaced by its value. Below the query, the results are shown in a table format.

EMPLOYEE_ID	LAST_NAME	SALARY
200	Whalen	4400

1 rows selected

Quiz

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

1. >=

2. IS NULL

3. !=

4. IS LIKE

5. IN BETWEEN

6. <>

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` operators
 - Apply the logical `AND`, `OR`, and `NOT` operators
- Use the `ORDER BY` clause to sort rows of output:

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

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

Practice 2: Overview

This practice covers the following topics:

- Selecting data and changing the order of the rows that are displayed
- Restricting rows by using the `WHERE` clause
- Sorting rows by using the `ORDER BY` clause
- Using substitution variables to add flexibility to your SQL `SELECT` statements