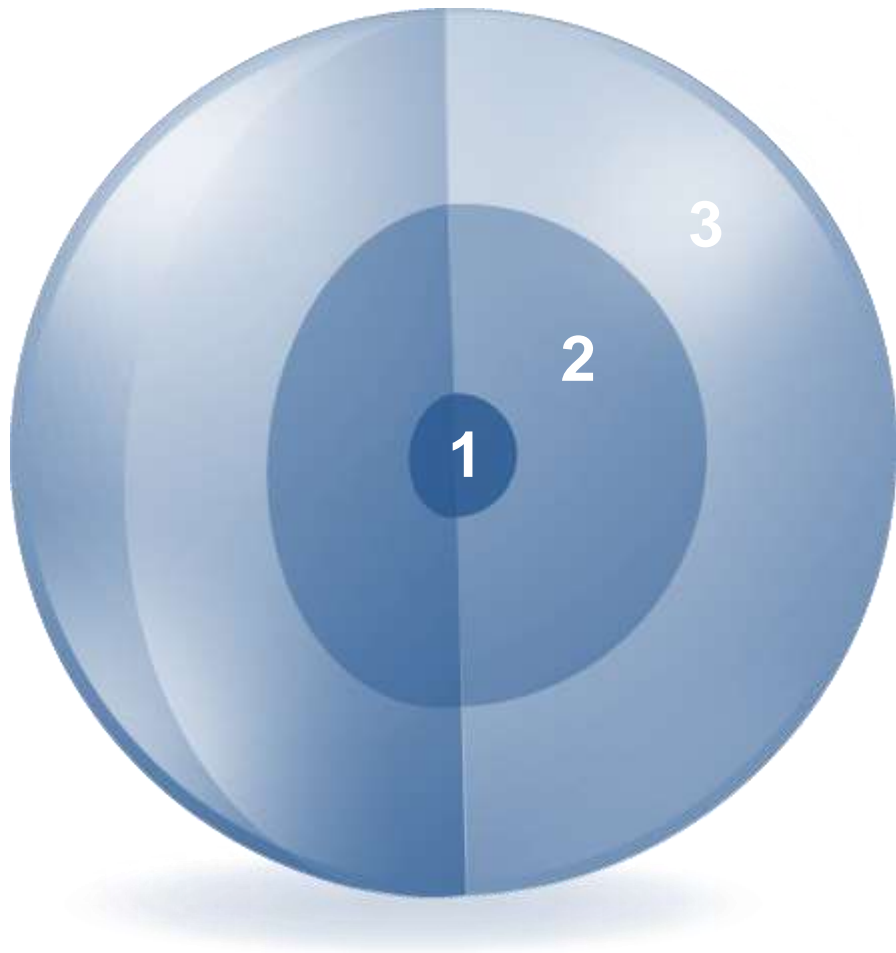


Restricting and Sorting Data





What You will learn at the end of this Session?



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

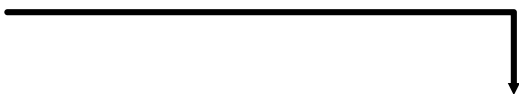
Limiting Rows Using a Selection





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_PRES		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 order_id, order_date, order_status  
FROM orders  
WHERE order_status = 1 ;
```

	 ORDER_ID	 ORDER_DATE	 ORDER_STATUS
1	2397	20-NOV-99 04.11.54.696211000 AM	1
2	2454	03-OCT-99 05.19.34.678340000 AM	1
3	2421	13-MAR-99 09.23.54.562432000 AM	1
4	2431	14-SEP-98 06.33.04.763452000 PM	1
5	2439	31-AUG-99 09.49.37.811132000 PM	1
6	2444	28-JUL-99 01.52.27.462632000 AM	1



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 order_id, order_date, order_mode
FROM orders
WHERE order_mode = 'direct' ;
```

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

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 order_id, order_date
FROM orders
WHERE order_id <= 2400 ;
```

	ORDER_ID	ORDER_DATE
1	2354	15-JUL-00 05.48.23.234567000 AM
2	2355	26-JAN-98 10.52.51.962632000 PM
3	2356	26-JAN-00 10.52.41.934562000 PM
4	2357	09-JAN-98 09.49.44.123456000 AM
5	2358	09-JAN-00 06.33.12.654278000 AM
6	2359	09-JAN-98 11.04.13.112233000 AM

■ ■ ■

Range Conditions Using the BETWEEN Operator

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

```
SELECT product_id, quantity_on_hand  
FROM inventories  
WHERE product_id BETWEEN 3100 AND 3108;
```

Lower limit Upper limit

	PRODUCT_ID	QUANTITY_ON_HAND
1	3108	122
2	3108	110
3	3108	194
4	3108	170
5	3108	146

Membership Condition Using the IN Operator

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

```
SELECT order_id, order_mode, order_status  
FROM orders  
WHERE order_id IN (2458, 2397, 2454);
```

	ORDER_ID	ORDER_MODE	ORDER_STATUS
1	2397	direct	1
2	2454	direct	1
3	2458	direct	0

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 order_ID, order_status, sales_rep_id
FROM orders
WHERE sales_rep_id IS NULL;
```

	ORDER_ID	ORDER_STATUS	SALES_REP_ID
1	2355	8	(null)
2	2356	5	(null)
3	2359	9	(null)
4	2361	8	(null)
5	2362	4	(null)
6	2363	0	(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

- AND requires both the component conditions to be true:

```
SELECT order_mode, order_status, customer_id
FROM orders
WHERE order_mode = 'direct'
AND customer_id = 103;
```

	ORDER_MODE	ORDER_STATUS	CUSTOMER_ID
1	direct	1	103
2	direct	4	103

Using the OR Operator

- OR requires either component condition to be true:

```
SELECT order_id, order_status, order_total
FROM orders
WHERE order_status = 0
      OR order_total >= 100000 ;
```

	ORDER_ID	ORDER_STATUS	ORDER_TOTAL
1	2458	0	70647.34
2	2354	0	46257
3	2434	8	242458.25
4	2361	8	120131.3
5	2363	0	10082.3
6	2367	10	144054.8
7	2369	0	11097.4
8	2375	2	103834.4
9	2385	4	295892
10	2388	4	282694.3
11	2399	0	25270.3


```
SELECT order_id, order_status, order_total
FROM orders
WHERE order_status
      NOT IN (0,1,2,3);
```

	ORDER_ID	ORDER_STATUS	ORDER_TOTAL
1	2357	5	59872.4
2	2394	5	21863
3	2435	6	62303
4	2455	7	14087.5
5	2379	8	17848.2
6	2396	8	34930
7	2434	8	242458.25
8	2436	8	6394.8
9	2446	8	93570.57
10	2447	8	33893.6
11	2432	10	10523

■ ■ ■

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.

```
SELECT *  
FROM inventories  
WHERE warehouse_id = 9  
OR quantity_on_hand = 150  
AND product_id = 3139 ;
```

1

	PRODUCT_ID	WAREHOUSE_ID	QUANTITY_ON_HAND
1	3139	8	150
2	1729	9	23
3	1733	9	35

■ ■ ■

```
SELECT *  
FROM inventories  
WHERE ( warehouse_id = 9  
OR quantity_on_hand = 150 )  
AND product_id = 3139 ;
```

2

	PRODUCT_ID	WAREHOUSE_ID	QUANTITY_ON_HAND
1	3139	8	150
2	3139	9	135

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 order_id, order_date, order_status  
FROM orders  
ORDER BY order_date ;
```

	ORDER_ID	ORDER_DATE	ORDER_STATUS
1	2442	27-JUL-90 11.52.59.662632000 PM	9
2	2445	28-JUL-90 03.04.38.362632000 AM	8
3	2418	21-MAR-96 05.48.21.862632000 AM	4
4	2357	09-JAN-98 09.49.44.123456000 AM	5

...

Sorting in descending order:

```
SELECT order_id, round(order_date), order_status  
FROM orders  
ORDER BY order_date desc;
```

1


Sorting by column alias:

```
SELECT order_id, round(order_date), order_status "Order Status"  
FROM orders  
ORDER BY order_date desc ;
```

2


Sorting by using the column's numeric position:

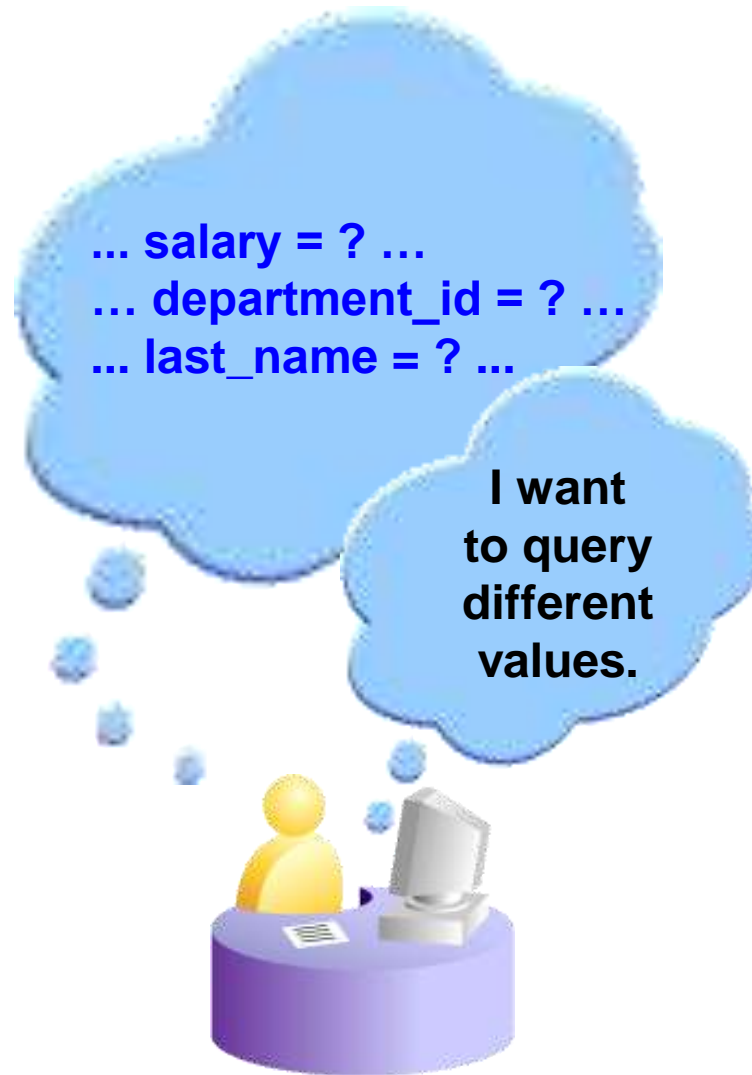
```
SELECT  last_name, job_id, department_id, hire_date
FROM    employees
ORDER BY 3;
```

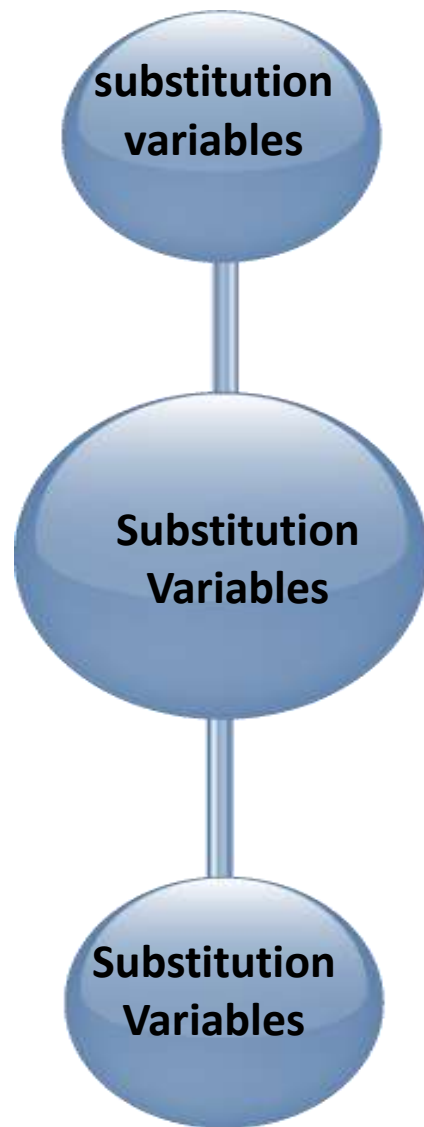


Sorting by multiple columns:

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







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

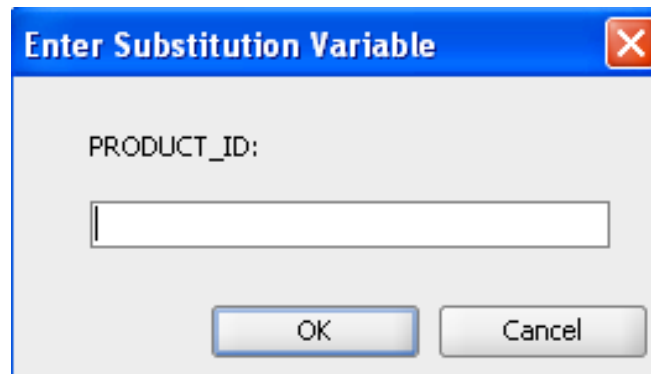
Table names

Entire SELECT statements

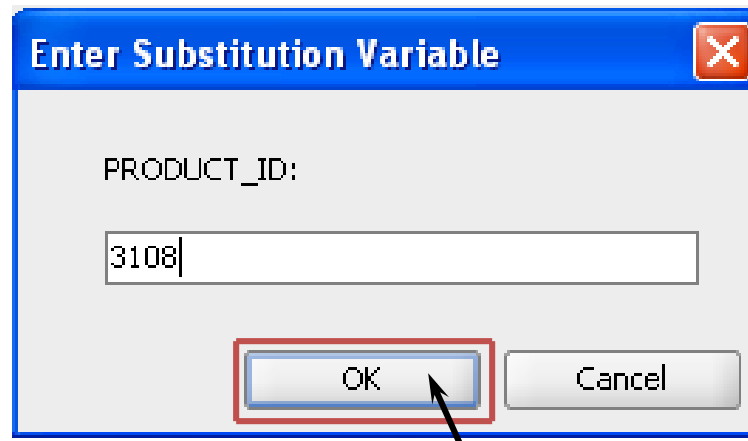
Using the Single-Ampersand Substitution Variable


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

```
SELECT product_id, warehouse_id, quantity_on_hand  
FROM inventories  
WHERE product_id = &product_id ;
```



Using the Single-Ampersand Substitution Variable

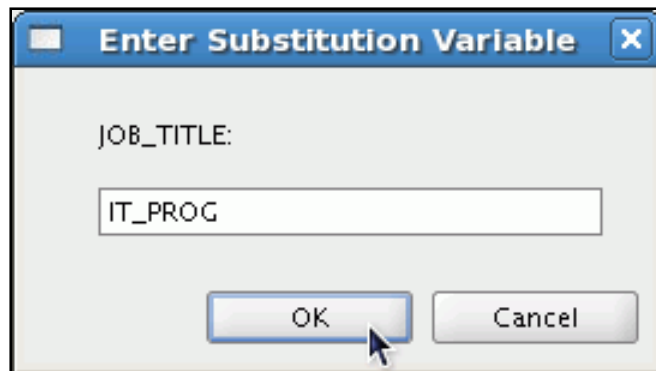


	 PRODUCT_ID	 WAREHOUSE_ID	 QUANTITY_ON_HAND
1	3108	8	122
2	3108	9	110
3	3108	2	194
4	3108	4	170
5	3108	6	146

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' ;
```



	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 &column_name  
FROM inventories  
WHERE &Condition  
ORDER BY &Order_column ;
```

Enter Substitution Variable

COLUMN_NAME:

quantity_on_hand

OK Cancel

Enter Substitution Variable

CONDITION:

quantity_on_hand = 150

OK Cancel

Enter Substitution Variable

ORDER_COLUMN:

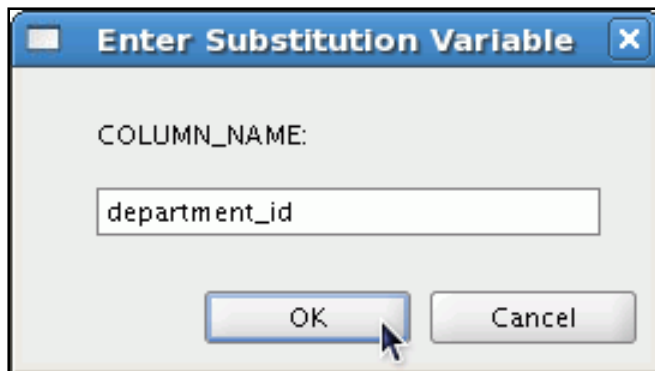
product_id

OK Cancel

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) in the top right corner. It contains a label "COLUMN_NAME:" followed by a text input field containing the text "department_id". At the bottom, there are two buttons: "OK" and "Cancel". A mouse cursor is pointing at the "OK" button.

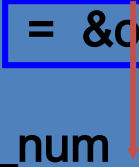
	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

Using the DEFINE Command

Use the DEFINE command to create and assign a value to a variable.

```
DEFINE order_num = 2458  
  
SELECT order_id, order_date, order_mode, order_total  
FROM orders  
WHERE order_id = &order_num;  
  
UNDEFINE order_num
```

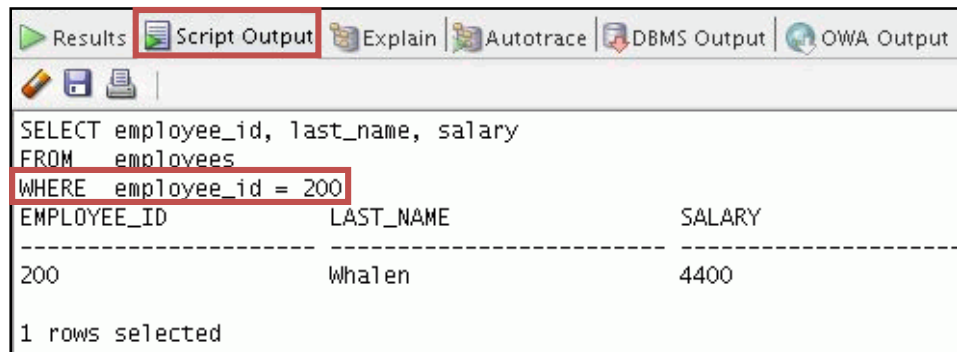
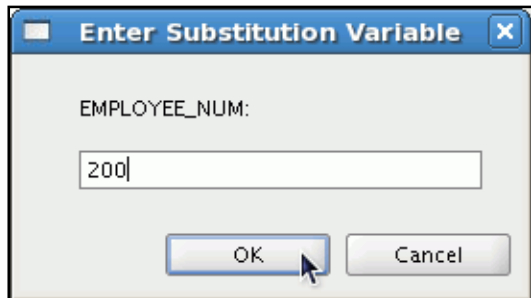


Using the VERIFY Command

- 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

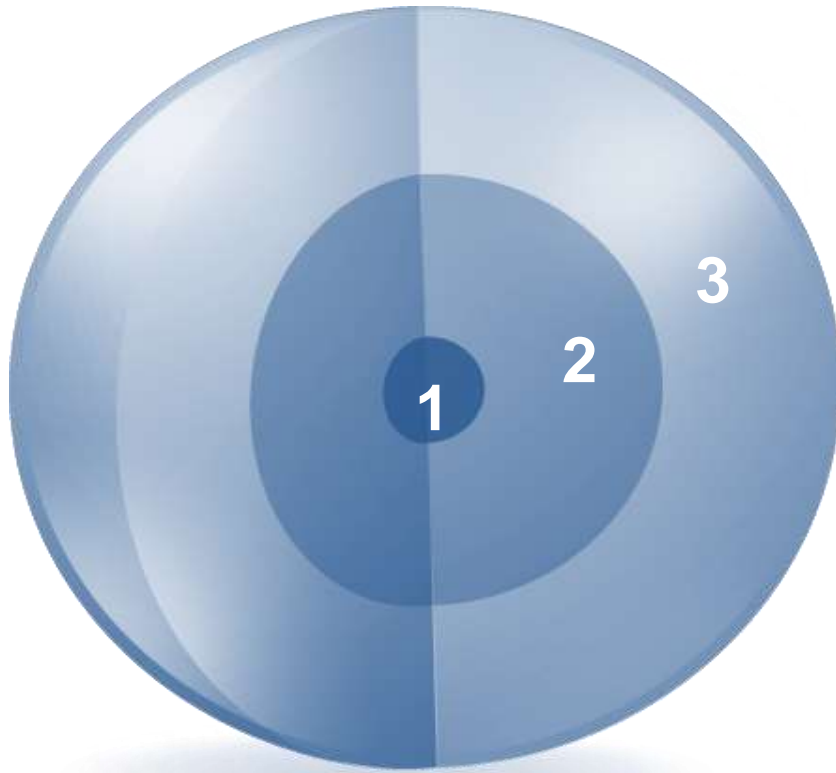
```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  employee_id = &employee_num;
```



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

1. `>=`
2. `IS NULL`
3. `!=`
4. `IS LIKE`
5. `IN BETWEEN`
6. `<>`

Session Summary



1. 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
2. Use ampersand substitution to restrict and sort output at run time
3. 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]] ;
```

This practice covers the following topics:

