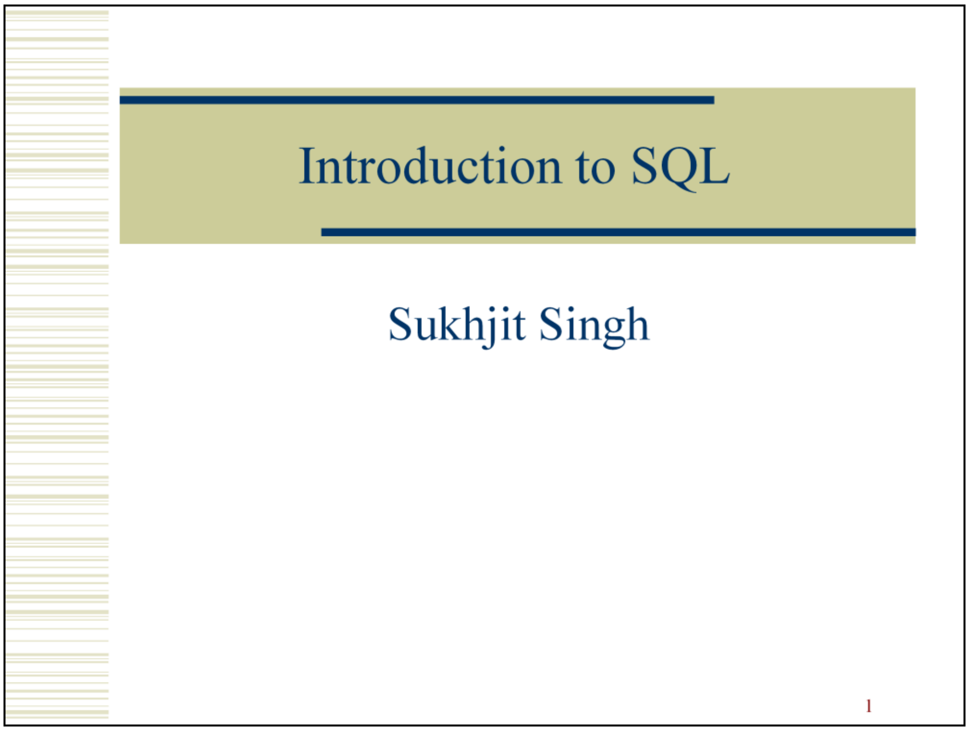
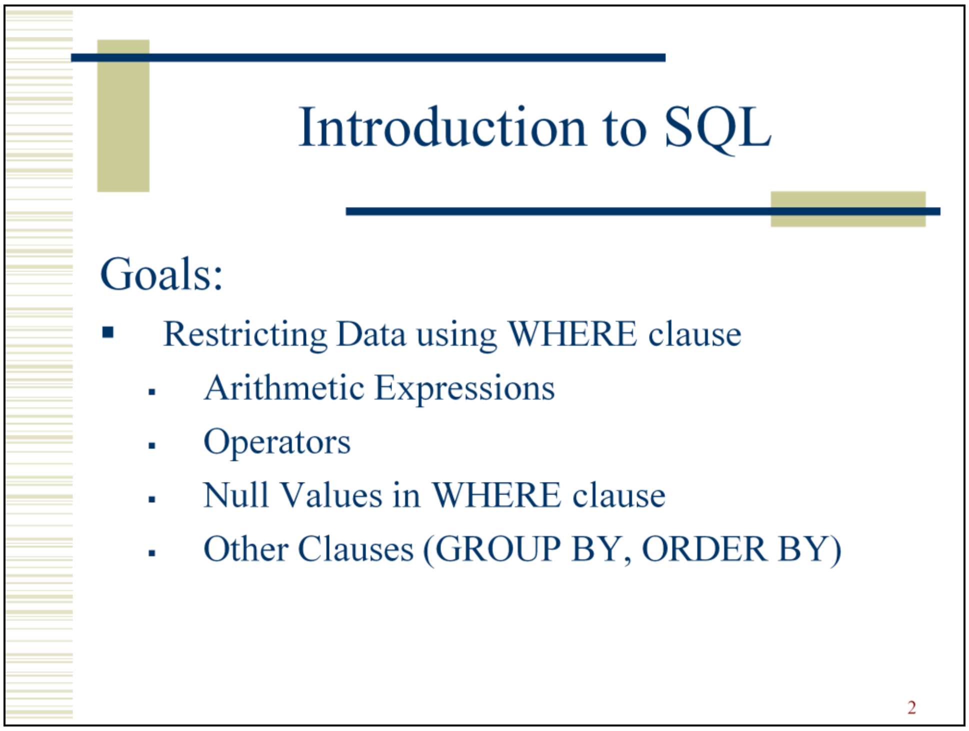


Introduction to SQL

Sukhjit Singh

1

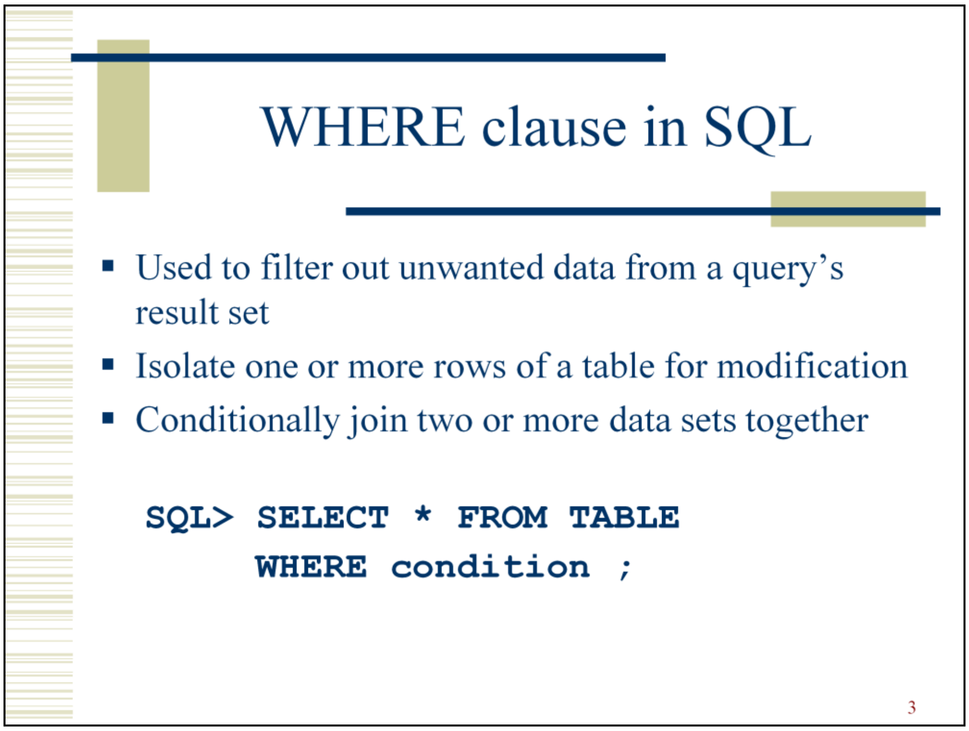


Introduction to SQL

Goals:

- Restricting Data using WHERE clause
  - Arithmetic Expressions
  - Operators
  - Null Values in WHERE clause
  - Other Clauses (GROUP BY, ORDER BY)

2

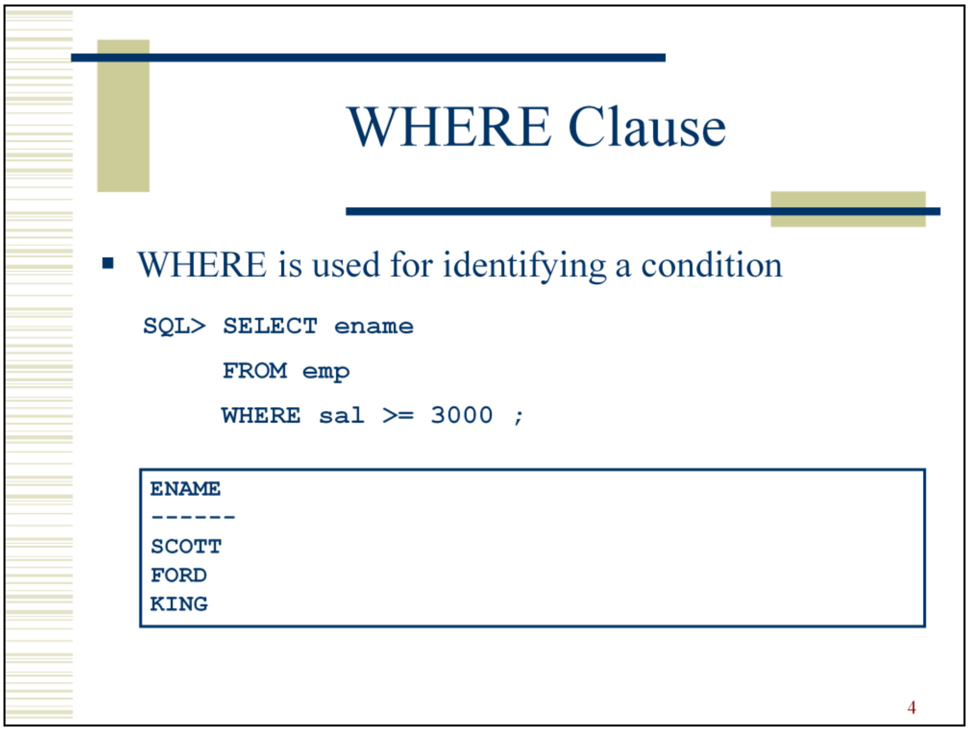


WHERE clause in SQL

- Used to filter out unwanted data from a query's result set
- Isolate one or more rows of a table for modification
- Conditionally join two or more data sets together

```
SQL> SELECT * FROM TABLE
     WHERE condition ;
```

3



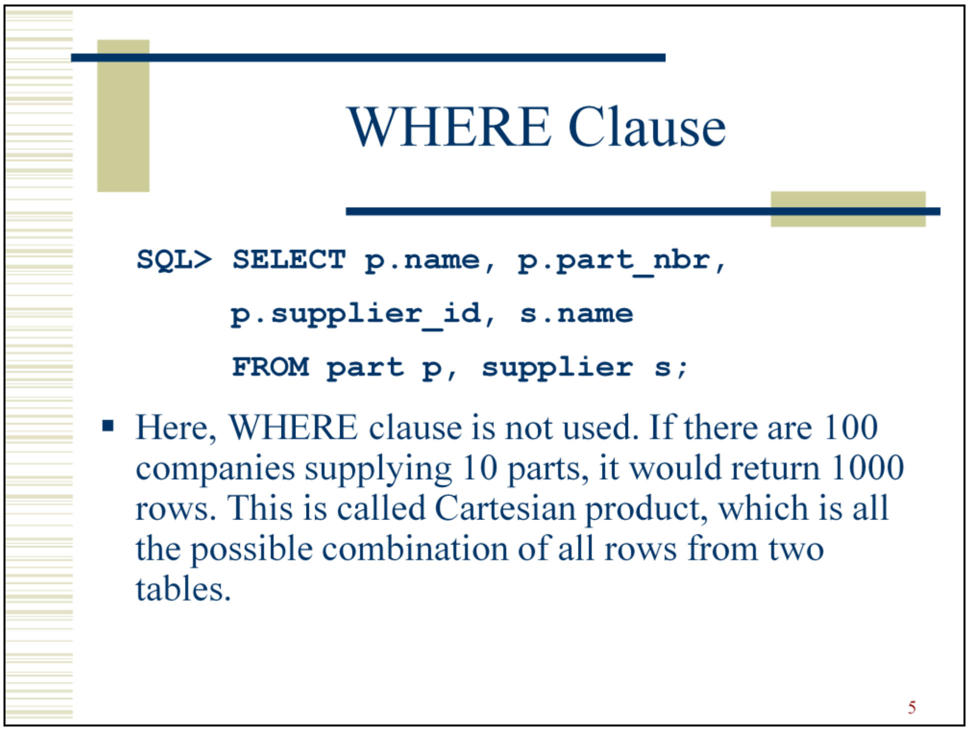
WHERE Clause

- WHERE is used for identifying a condition

```
SQL> SELECT ename
     FROM emp
     WHERE sal >= 3000 ;
```

```
ENAME
------
SCOTT
FORD
KING
```

4

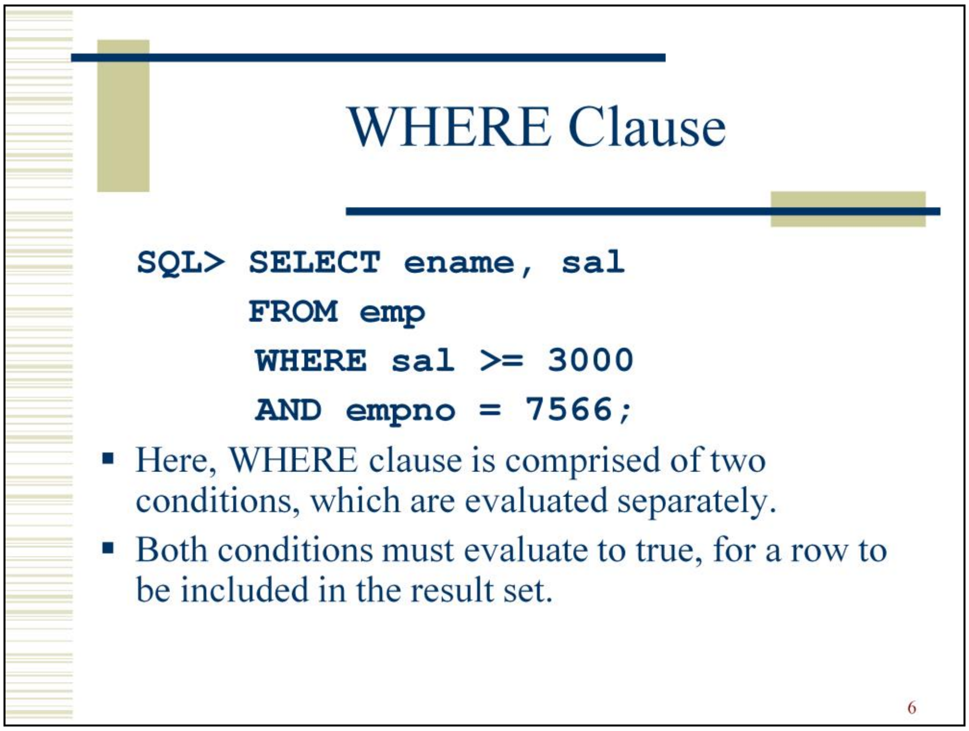


WHERE Clause

```
SQL> SELECT p.name, p.part_nbr,
     p.supplier_id, s.name
     FROM part p, supplier s;
```

- Here, WHERE clause is not used. If there are 100 companies supplying 10 parts, it would return 1000 rows. This is called Cartesian product, which is all the possible combination of all rows from two tables.

5



WHERE Clause

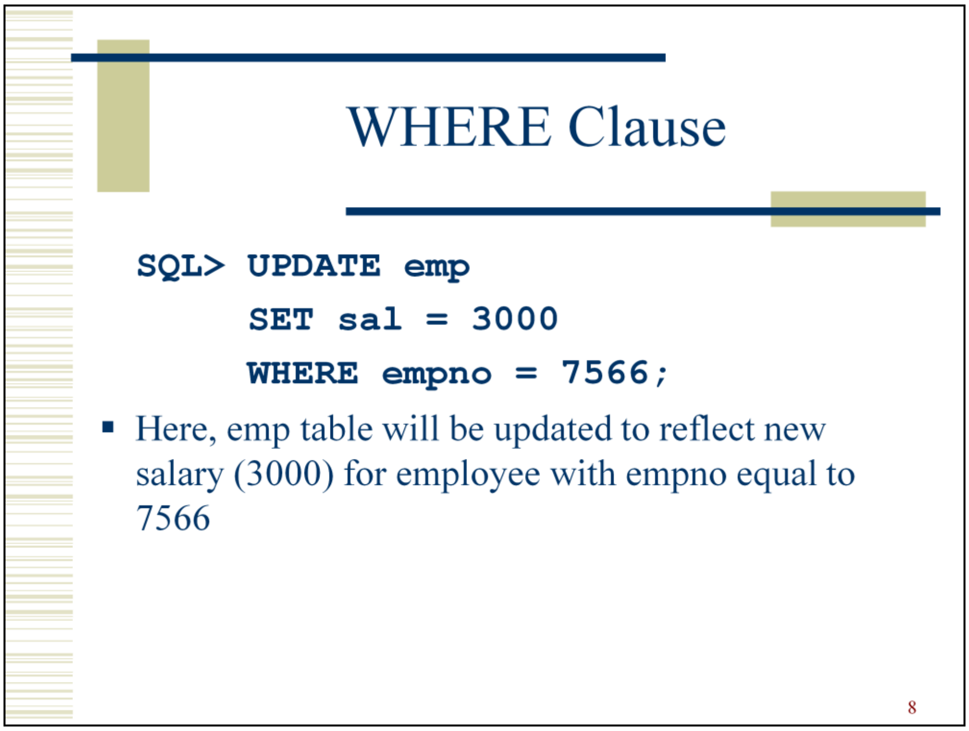
```
SQL> SELECT ename, sal
     FROM emp
      WHERE sal >= 3000
      AND empno = 7566;
```

- Here, WHERE clause is comprised of two conditions, which are evaluated separately.
- Both conditions must evaluate to true, for a row to be included in the result set.

6

Condition in WHERE clause

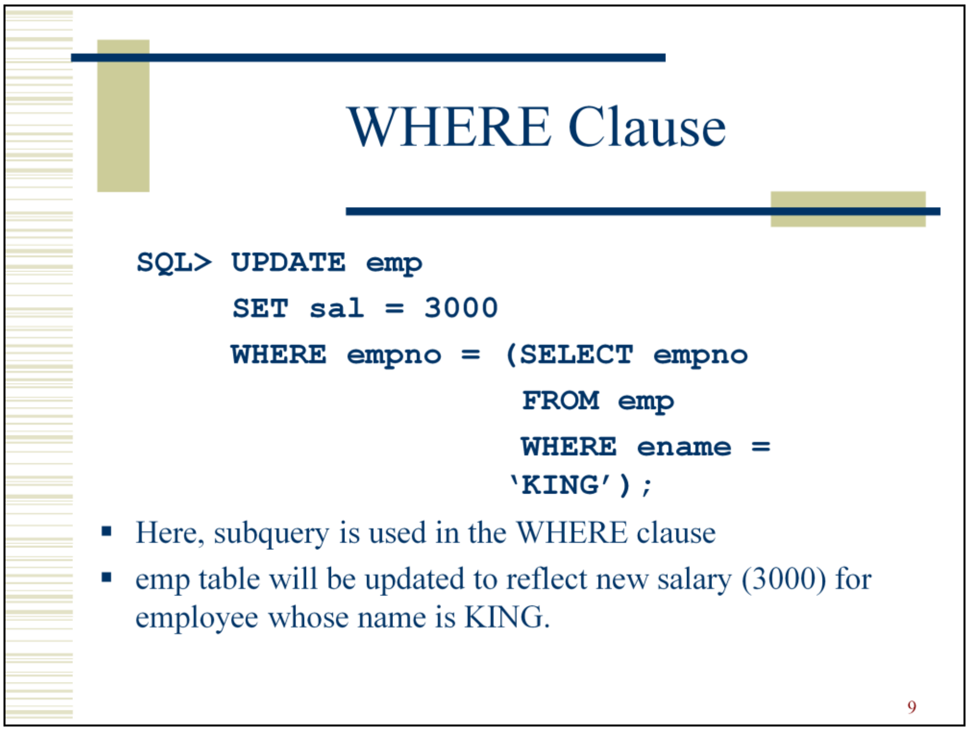
- A condition is comprised of one or more expressions with one or more operators.
- Expressions include:
  - Numbers, Columns, Literals, Functions ,Subqueries
- Operators include:
  - Arithmetic Operators, Comparison Operators, Character Operators, Logical Operators, IN and BETWEEN operators



WHERE Clause

```
SQL> UPDATE emp
     SET sal = 3000
     WHERE empno = 7566;
```

- Here, emp table will be updated to reflect new salary (3000) for employee with empno equal to 7566

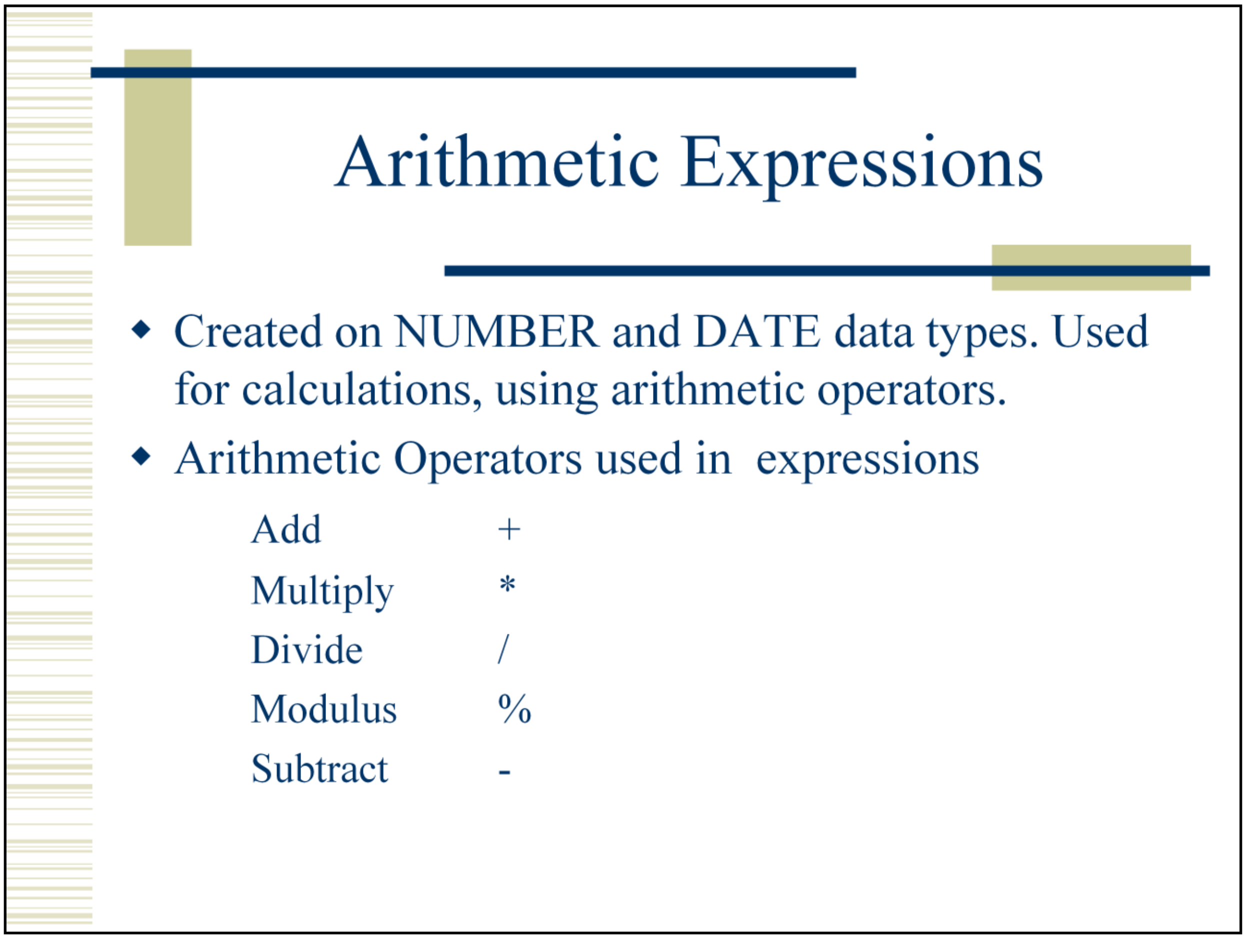


WHERE Clause

```
SQL> UPDATE emp
     SET sal = 3000
     WHERE empno = (SELECT empno
                    FROM emp
                    WHERE ename =
                   'KING');
```

- Here, subquery is used in the WHERE clause
- emp table will be updated to reflect new salary (3000) for employee whose name is KING.

9



Arithmetic Expressions

- Created on NUMBER and DATE data types. Used for calculations, using arithmetic operators.
- Arithmetic Operators used in expressions

Add	+
Multiply	*
Divide	/
Modulus	%
Subtract	-

Arithmetic Expressions

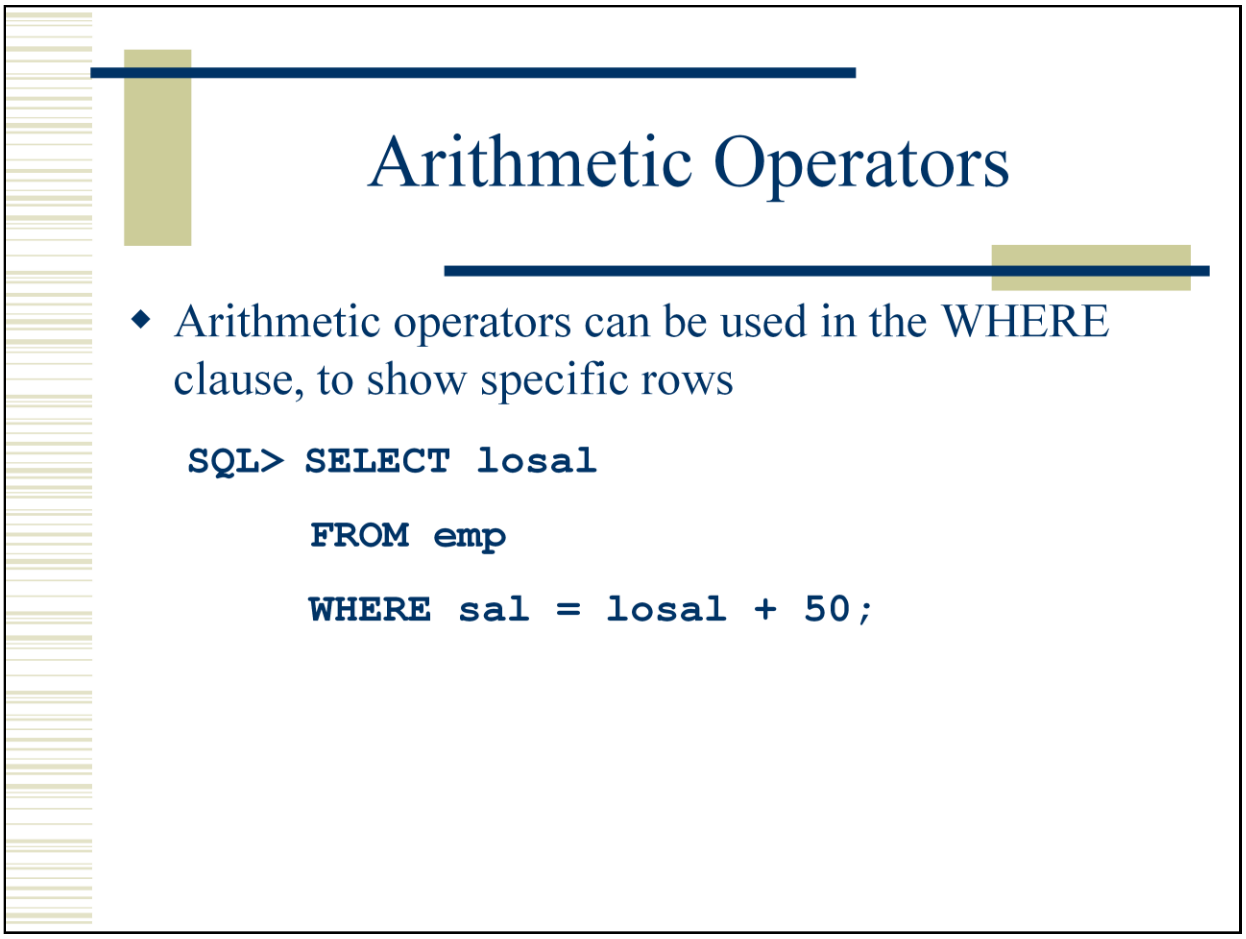
You may need to modify the way in which data is displayed, perform calculations, or look at what-if scenarios. This is possible using arithmetic expressions. An arithmetic expression may contain column names, constant numeric values, and the arithmetic operators.

Arithmetic Operators

The slide lists the arithmetic operators available in SQL. You can use arithmetic operators in any clause of a SQL statement except the FROM clause.

Instructor Note

You can use only the addition and subtraction operators with DATE datatypes.



Arithmetic Operators

- Arithmetic operators can be used in the WHERE clause, to show specific rows

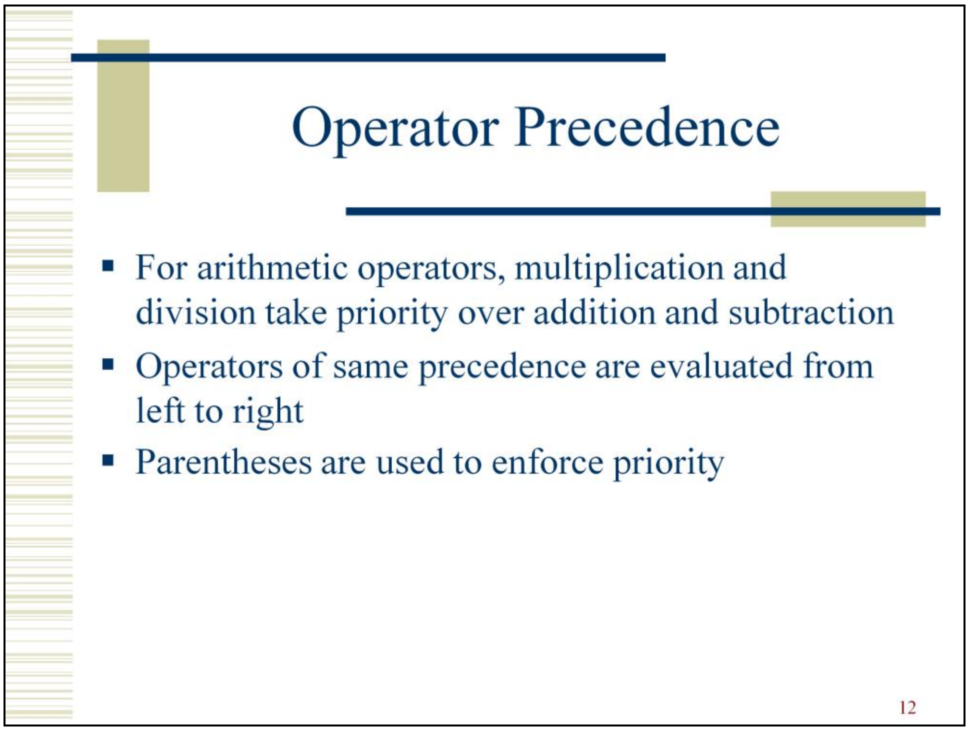
```
SQL> SELECT losal
     FROM emp
     WHERE sal = losal + 50;
```

Using Arithmetic Operators

The example in the slide uses the addition operator to calculate a salary increase of $300 for all employees and displays a new SAL+300 column in the output.

Note that the resultant calculated column SAL+300 is not a new column in the EMP table; it is for display only. By default, the name of a new column comes from the calculation that generated it—in this case, sal+300.

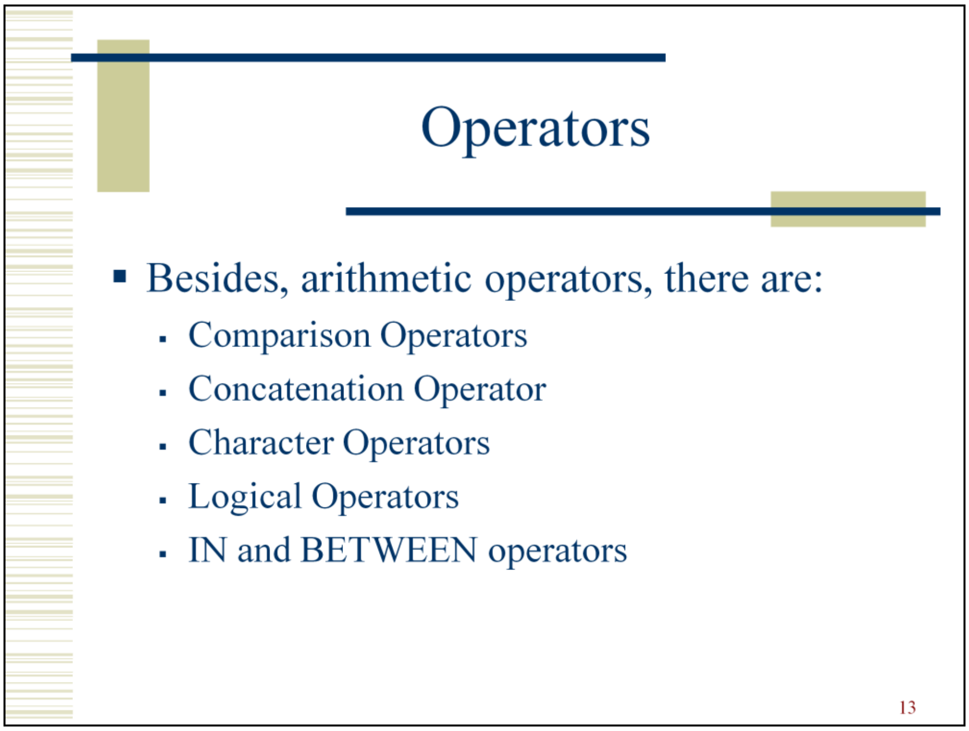
Note: SQL*Plus ignores blank spaces before and after the arithmetic operator.



Operator Precedence

- For arithmetic operators, multiplication and division take priority over addition and subtraction
- Operators of same precedence are evaluated from left to right
- Parentheses are used to enforce priority

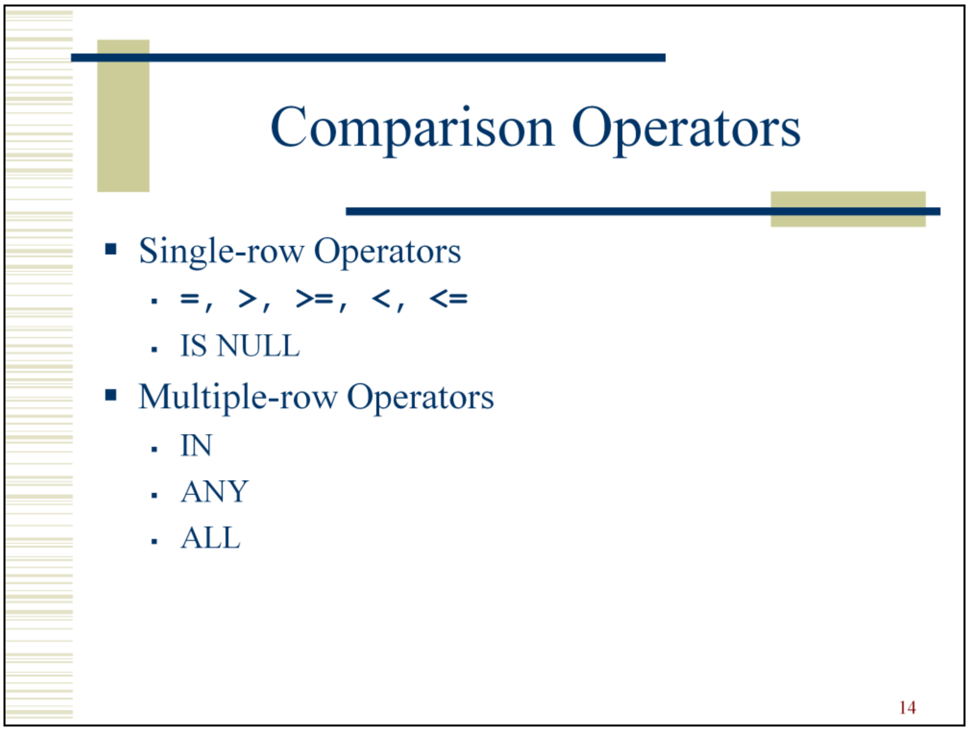
12



Operators

- Besides, arithmetic operators, there are:
  - Comparison Operators
  - Concatenation Operator
  - Character Operators
  - Logical Operators
  - IN and BETWEEN operators

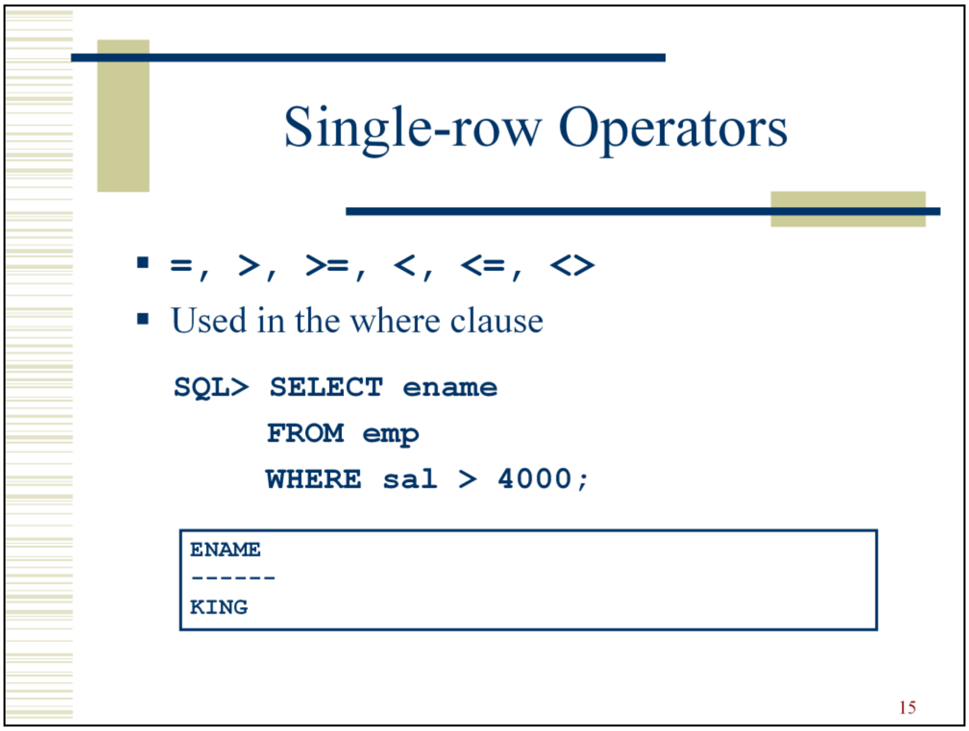
13



Comparison Operators

- Single-row Operators
  - `=, >, >=, <, <=`
  - IS NULL
- Multiple-row Operators
  - IN
  - ANY
  - ALL

14



Single-row Operators

- `=, >, >=, <, <=, <>`
- Used in the where clause

```
SQL> SELECT ename
     FROM emp
     WHERE sal > 4000;
```

```
ENAME
------
KING
```

15

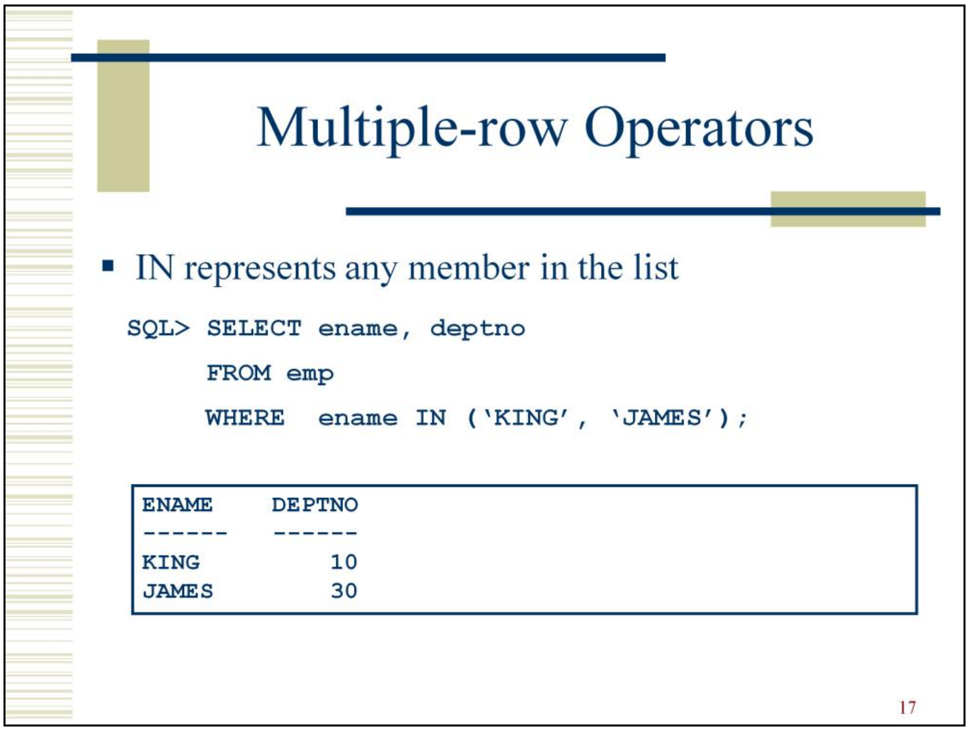
Single-row Operators

- IS NULL operator
  - Null – unavailable, unassigned value
  - Cannot compare null values using =

```
SQL> SELECT distinct deptno
     FROM emp
     WHERE comm  IS NULL;
```

```
DEPTNO
------
    10
    20
    30
```

16



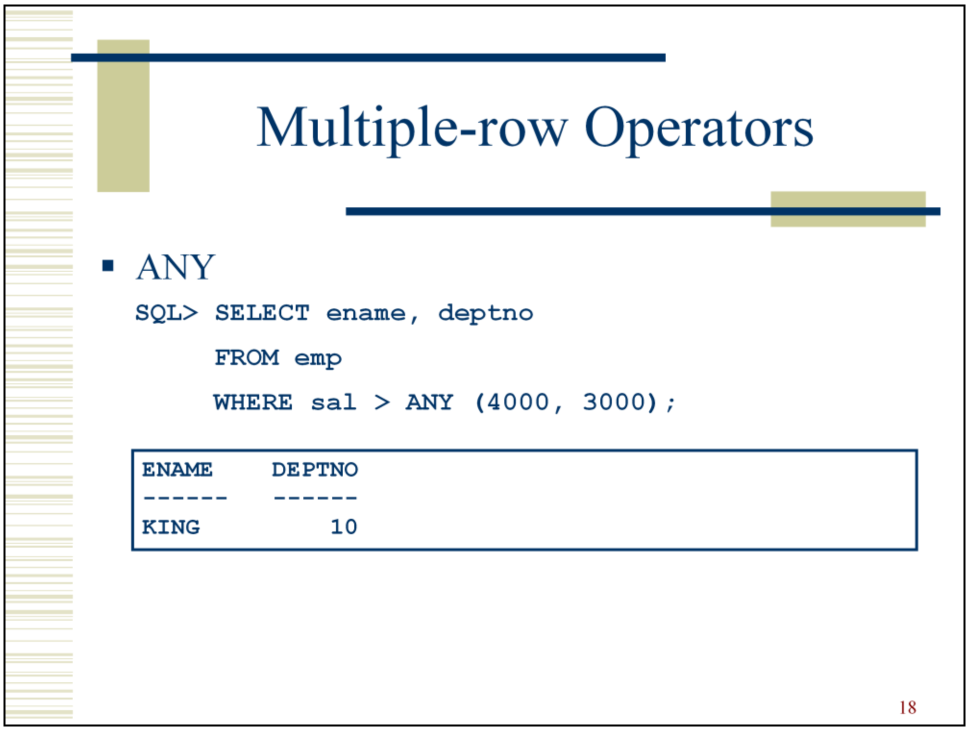
Multiple-row Operators

- IN represents any member in the list

```
SQL> SELECT ename, deptno
     FROM emp
     WHERE  ename IN ('KING', 'JAMES');
```

```
ENAME     DEPTNO
------    ------
KING          10
JAMES         30
```

17

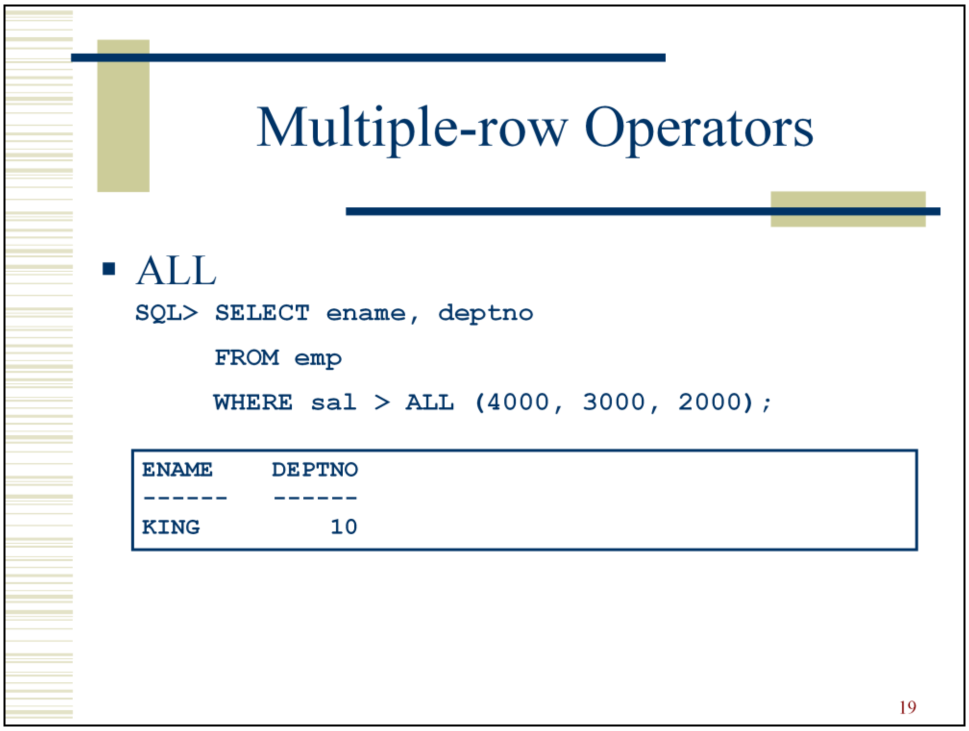

Multiple-row Operators

- ANY

```
SQL> SELECT ename, deptno
     FROM emp
     WHERE sal > ANY (4000, 3000);
```

```
ENAME     DEPTNO
------    ------
KING          10
```

18



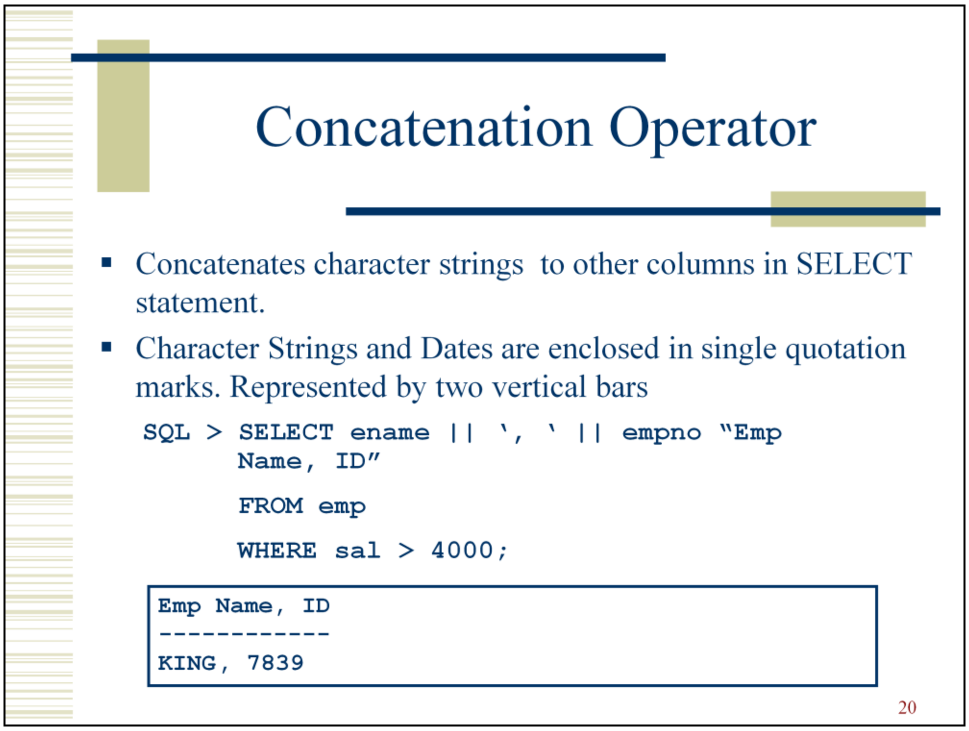
Multiple-row Operators

- ALL

```
SQL> SELECT ename, deptno
     FROM emp
     WHERE sal > ALL (4000, 3000, 2000);
```

```
ENAME    DEPTNO
------   ------
KING         10
```

19



Concatenation Operator

- Concatenates character strings to other columns in SELECT statement.
- Character Strings and Dates are enclosed in single quotation marks. Represented by two vertical bars

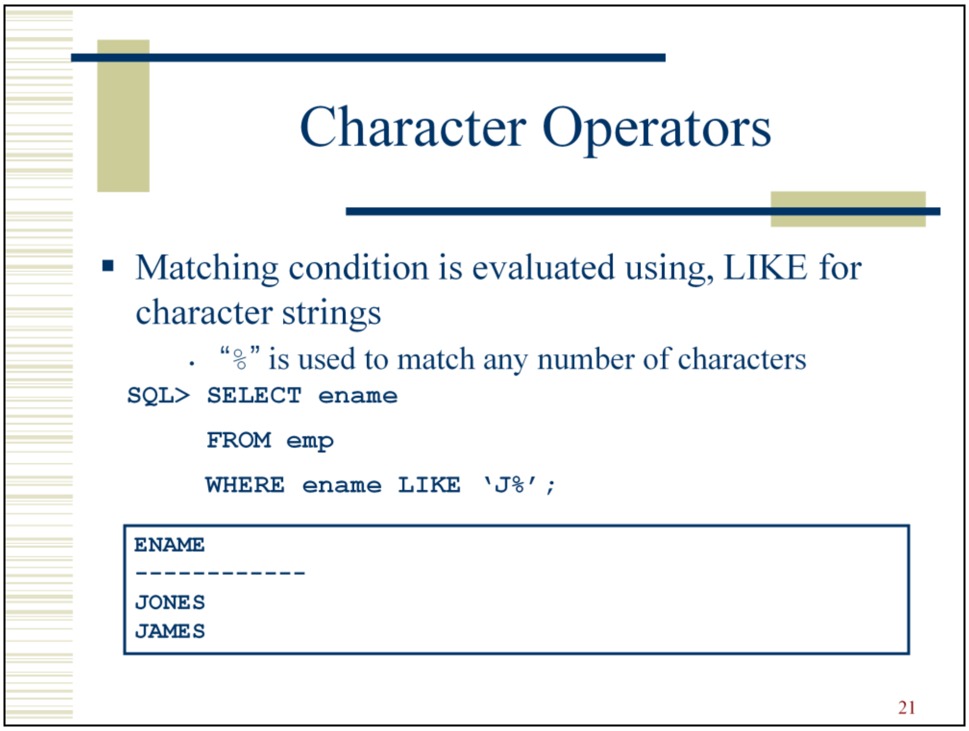
```
SQL > SELECT ename || ‘, ‘ || empno “Emp
      Name, ID”

      FROM emp

      WHERE sal > 4000;
```

```
Emp Name, ID
------------
KING, 7839
```

20



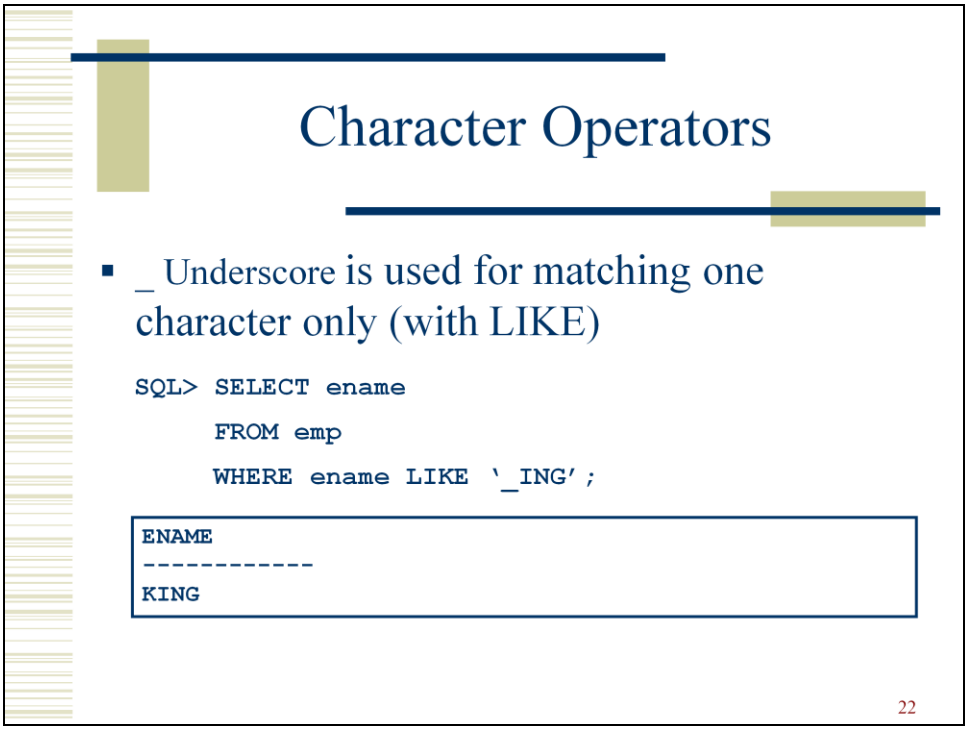
Character Operators

- Matching condition is evaluated using, LIKE for character strings
  - "%" is used to match any number of characters

```
SQL> SELECT ename
     FROM emp
     WHERE ename LIKE 'J%';
```

```
ENAME
------------
JONES
JAMES
```

21



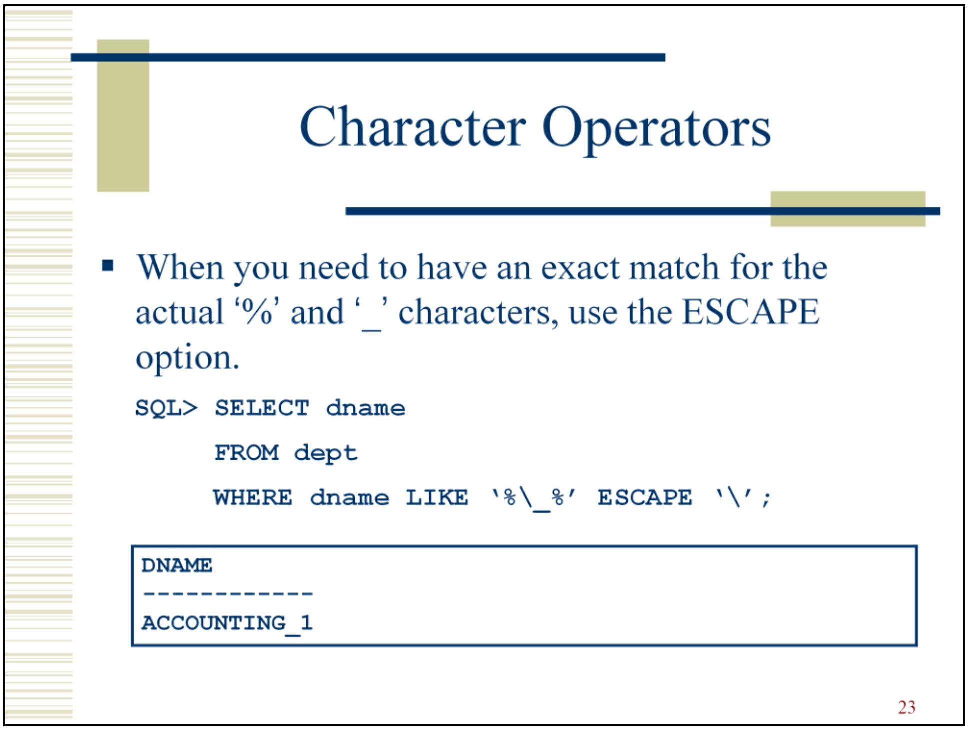
Character Operators

- _ Underscore is used for matching one character only (with LIKE)

```
SQL> SELECT ename
     FROM emp
     WHERE ename LIKE '_ING';
```

```
ENAME
------------
KING
```

22

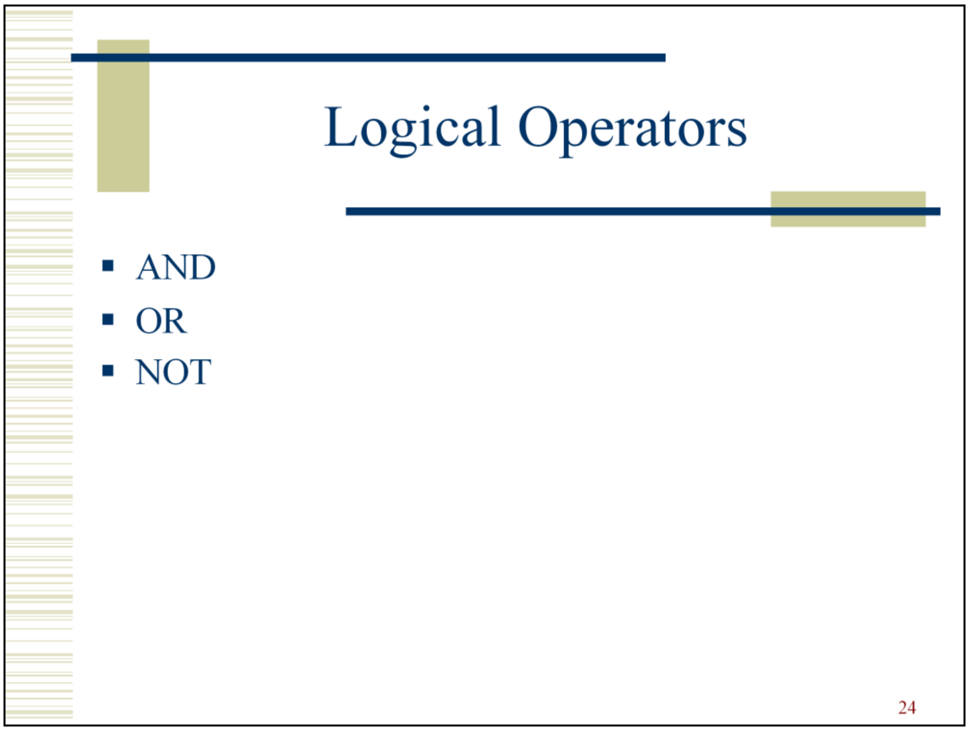


Character Operators

- When you need to have an exact match for the actual '%' and '_' characters, use the ESCAPE option.

```
SQL> SELECT dname
     FROM dept
     WHERE dname LIKE '%\_%' ESCAPE '\';
```

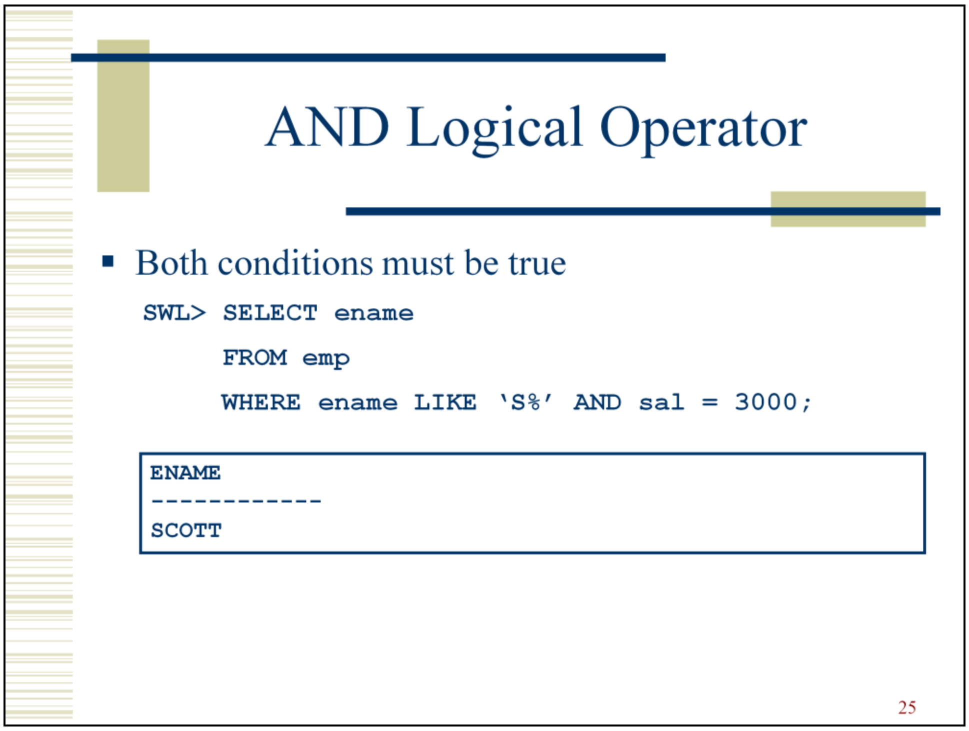
```
DNAME
------------
ACCOUNTING_1
```



Logical Operators

- AND
- OR
- NOT

24



AND Logical Operator

- Both conditions must be true

```
SWL> SELECT ename
     FROM emp
     WHERE ename LIKE 'S%' AND sal = 3000;
```

```
ENAME
------------
SCOTT
```

25

OR Logical Operator

- One of the conditions can be true

```
SQL> SELECT ename
     FROM emp
     WHERE ename LIKE ‘S%’ OR sal = 3000;
```

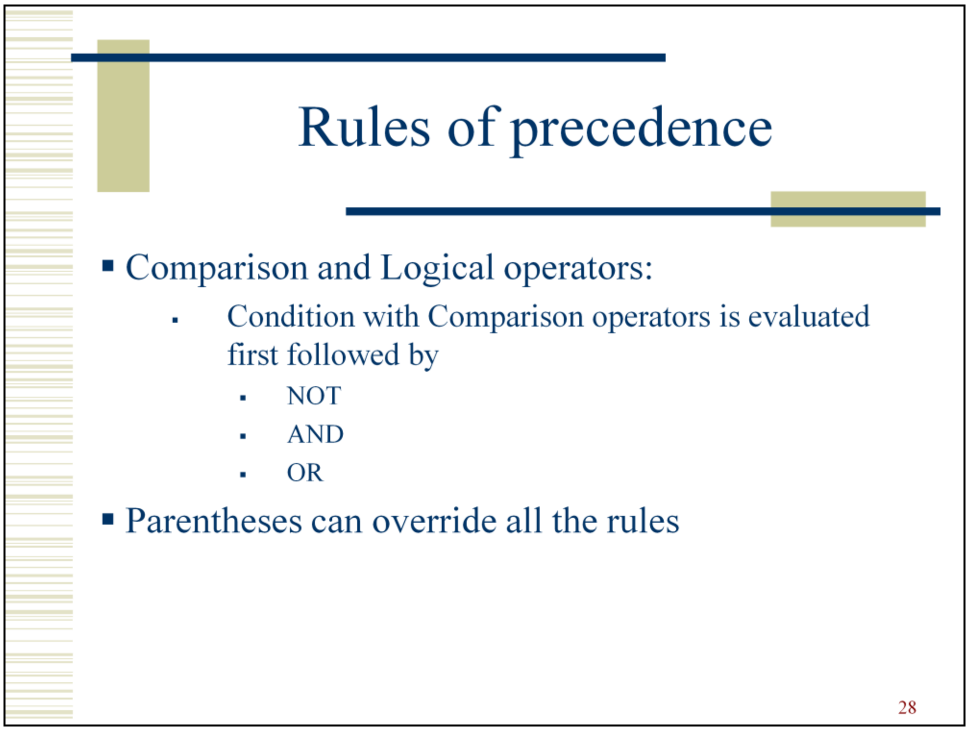
```
ENAME
------------
SCOTT
FORD
```

NOT Logical Operator

- Displays rows not matching the criteria given with NOT

```
SQL> SELECT *
     FROM dept
     WHERE deptno NOT IN (20, 40);
```

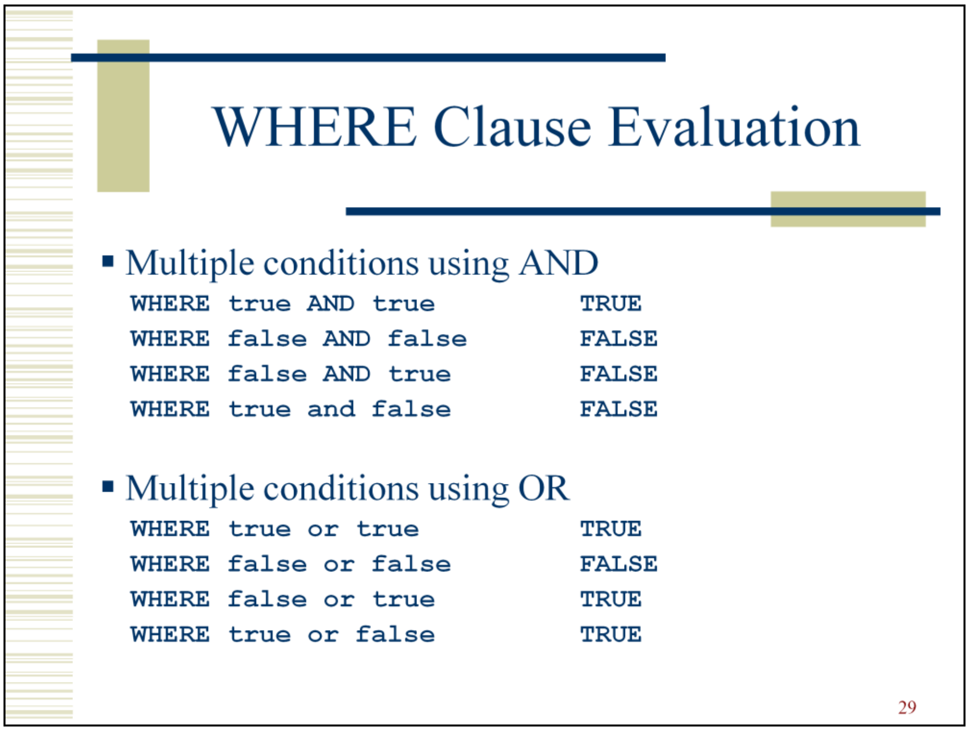
```
DEPTNO  DNAME           LOC
------  --------------  -----------
    10  ACCOUNTING      NEW YORK
    30  SALES           CHICAGO
```



Rules of precedence

- Comparison and Logical operators:
  - Condition with Comparison operators is evaluated first followed by
    - NOT
    - AND
    - OR
- Parentheses can override all the rules

28



WHERE Clause Evaluation

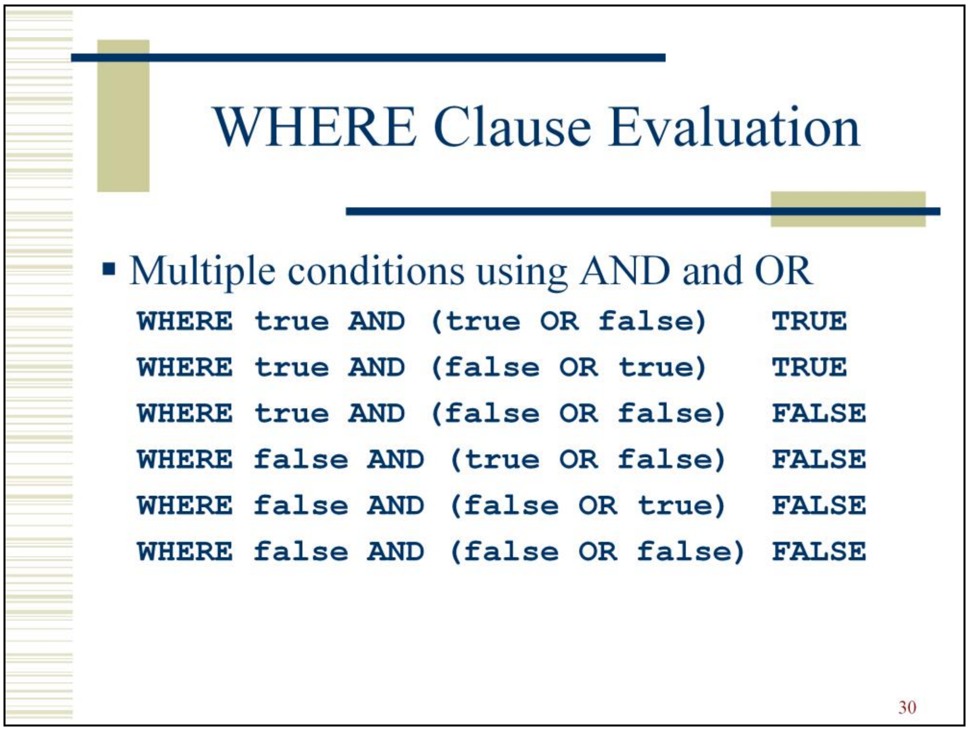
- Multiple conditions using AND

```
WHERE true AND true             TRUE
WHERE false AND false           FALSE
WHERE false AND true            FALSE
WHERE true and false            FALSE
```

- Multiple conditions using OR

```
WHERE true or true              TRUE
WHERE false or false            FALSE
WHERE false or true             TRUE
WHERE true or false             TRUE
```

29

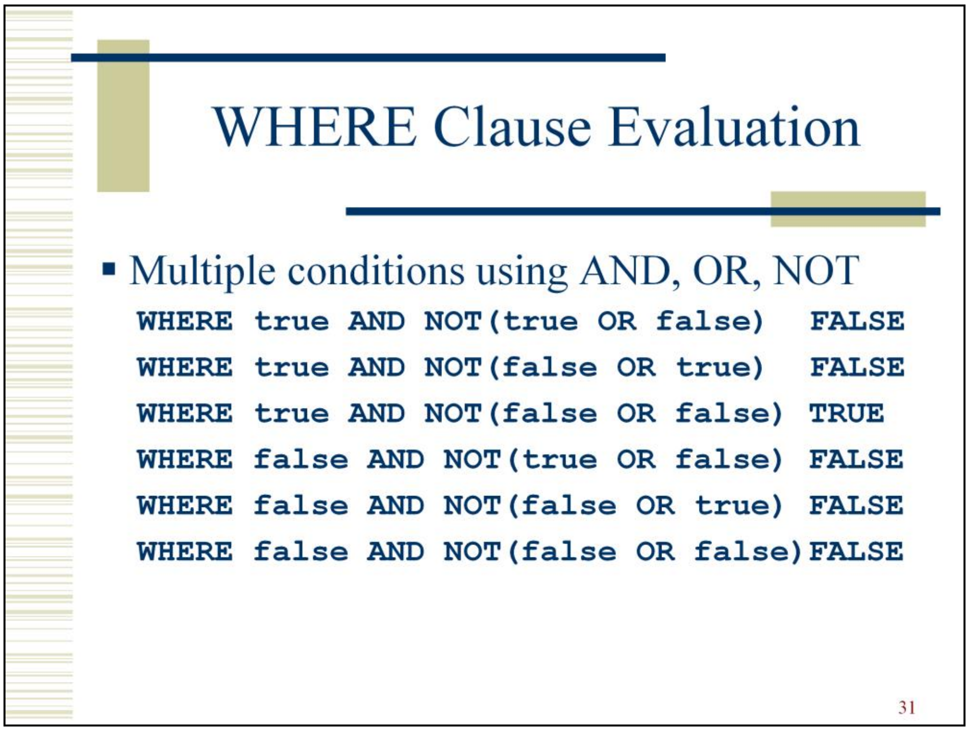


WHERE Clause Evaluation

- Multiple conditions using AND and OR

```
WHERE true AND (true OR false)    TRUE
WHERE true AND (false OR true)    TRUE
WHERE true AND (false OR false)   FALSE
WHERE false AND (true OR false)   FALSE
WHERE false AND (false OR true)   FALSE
WHERE false AND (false OR false)  FALSE
```

30

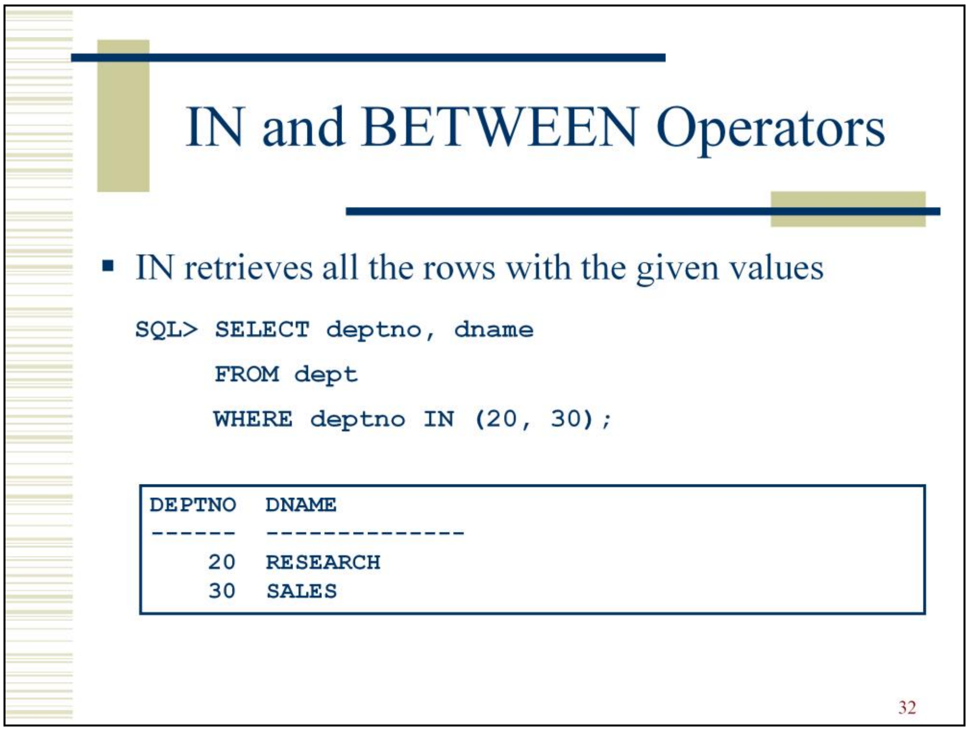


WHERE Clause Evaluation

- Multiple conditions using AND, OR, NOT

```
WHERE true AND NOT(true OR false)   FALSE
WHERE true AND NOT(false OR true)   FALSE
WHERE true AND NOT(false OR false)  TRUE
WHERE false AND NOT(true OR false)  FALSE
WHERE false AND NOT(false OR true)  FALSE
WHERE false AND NOT(false OR false) FALSE
```

31



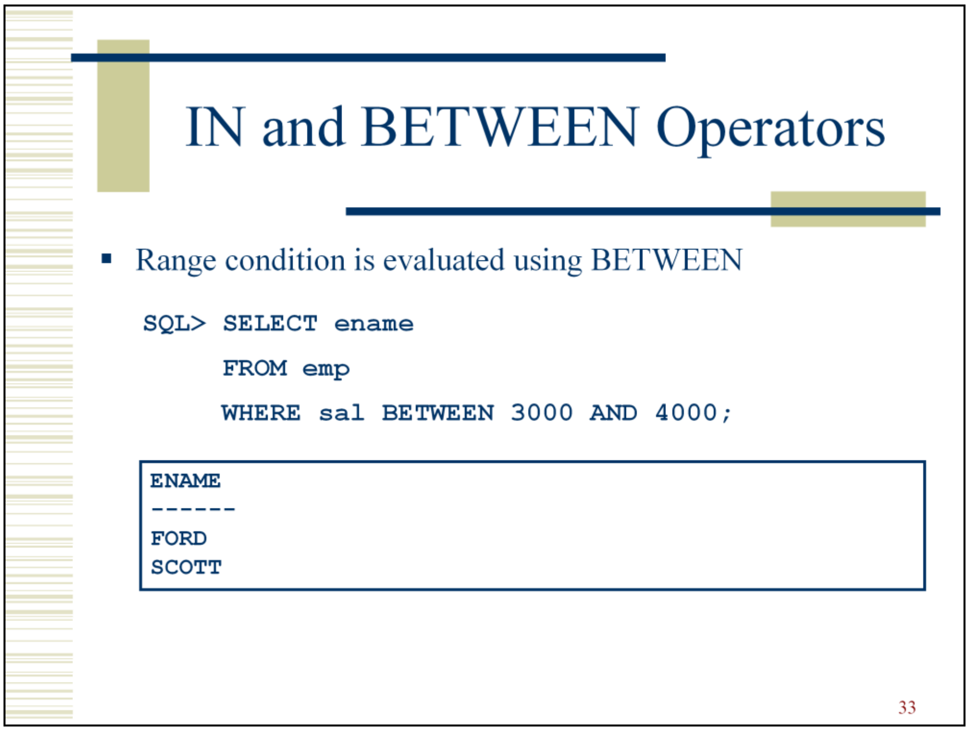
IN and BETWEEN Operators

- IN retrieves all the rows with the given values

```
SQL> SELECT deptno, dname
     FROM dept
     WHERE deptno IN (20, 30);
```

```
DEPTNO  DNAME
------  --------------
    20  RESEARCH
    30  SALES
```

32



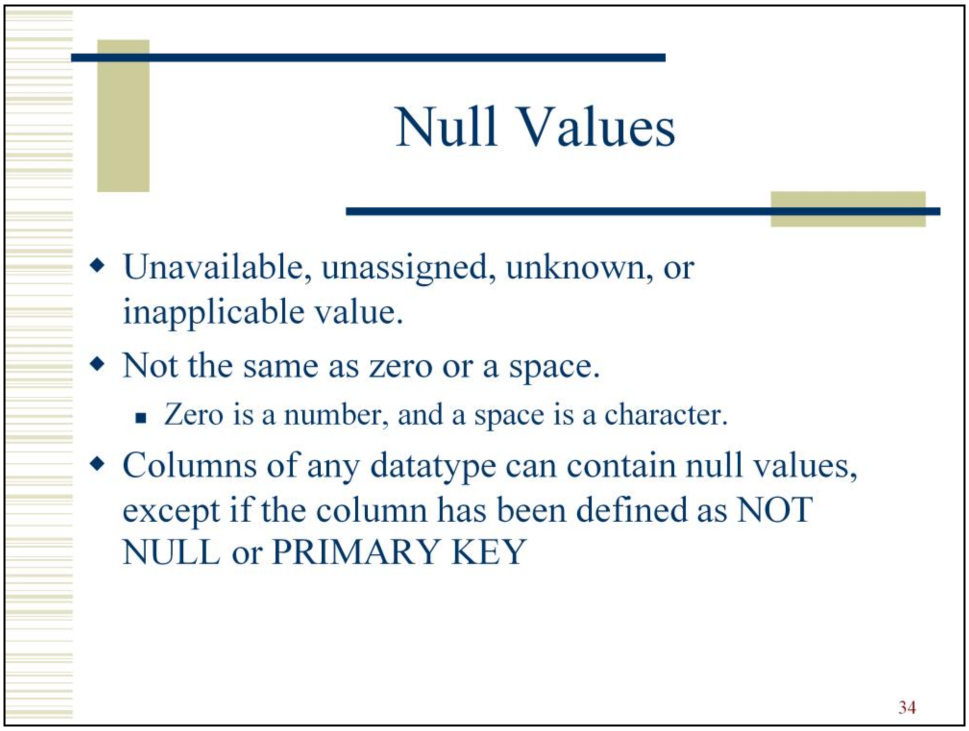
IN and BETWEEN Operators

- Range condition is evaluated using BETWEEN

```
SQL> SELECT ename
     FROM emp
     WHERE sal BETWEEN 3000 AND 4000;
```

```
ENAME
------
FORD
SCOTT
```

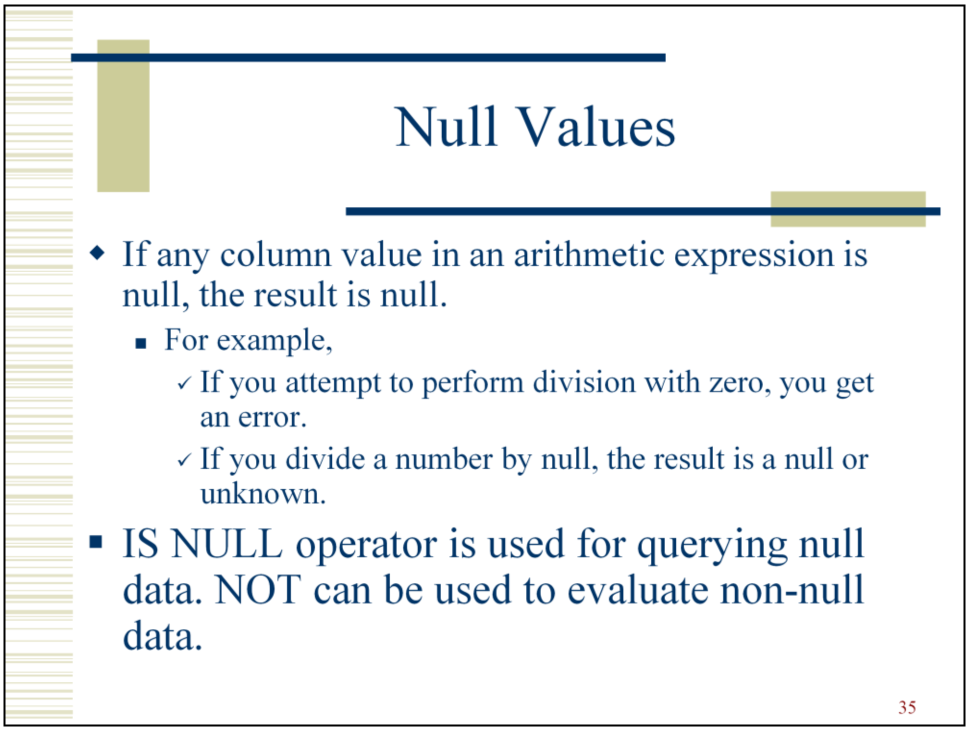
33



Null Values

- Unavailable, unassigned, unknown, or inapplicable value.
- Not the same as zero or a space.
  - Zero is a number, and a space is a character.
- Columns of any datatype can contain null values, except if the column has been defined as NOT NULL or PRIMARY KEY

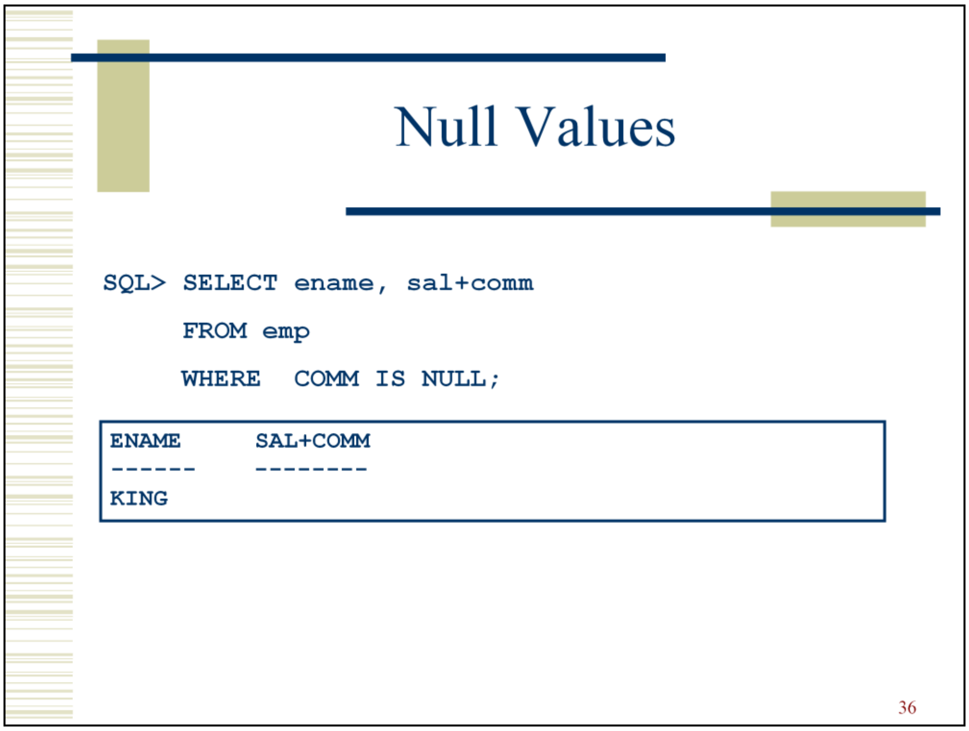
34



Null Values

- If any column value in an arithmetic expression is null, the result is null.
  - For example,
    - ✓ If you attempt to perform division with zero, you get an error.
    - ✓ If you divide a number by null, the result is a null or unknown.
- IS NULL operator is used for querying null data. NOT can be used to evaluate non-null data.

35

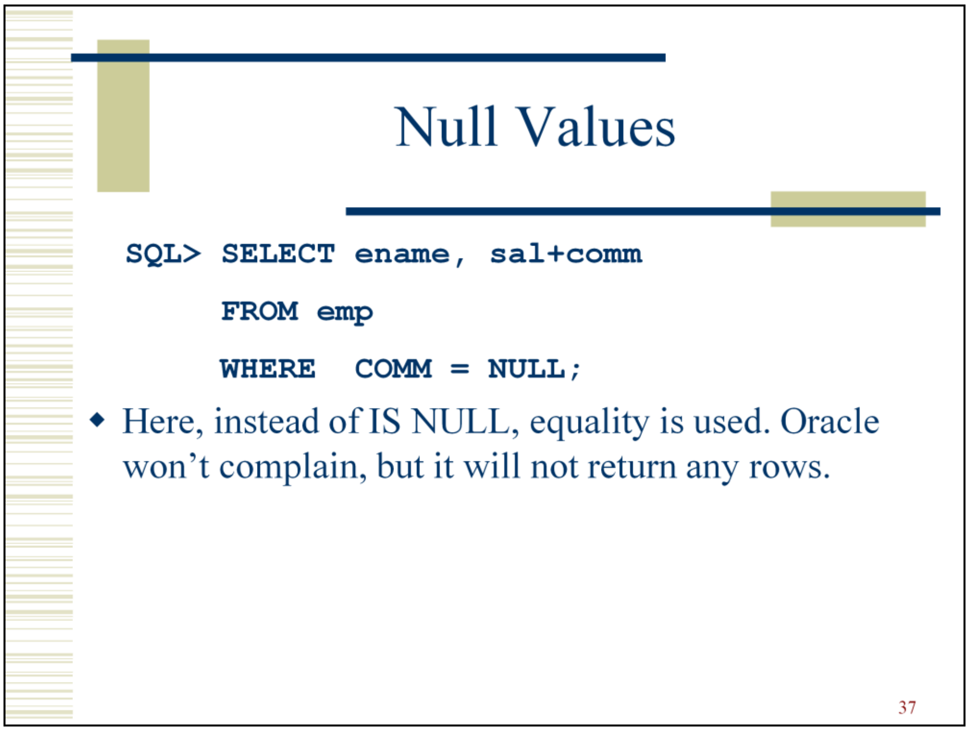


Null Values

```
SQL> SELECT ename, sal+comm
     FROM emp
     WHERE  COMM IS NULL;
```

```
ENAME      SAL+COMM
------     --------
KING
```

36



Null Values

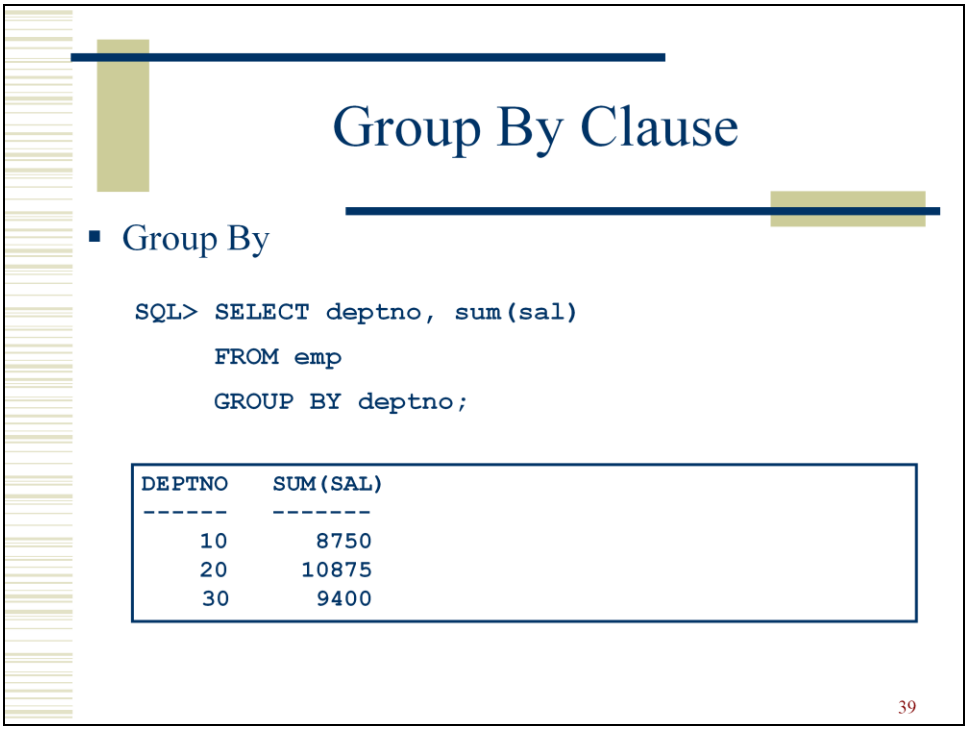
```
SQL> SELECT ename, sal+comm
     FROM emp
     WHERE  COMM = NULL;
```

- Here, instead of IS NULL, equality is used. Oracle won’t complain, but it will not return any rows.

Other Clauses

```
SELECT column,
FROM table
[WHERE  condition]
[GROUP BY group_by_expression]
[HAVING group_condition]
[ORDER BY column];
```

38



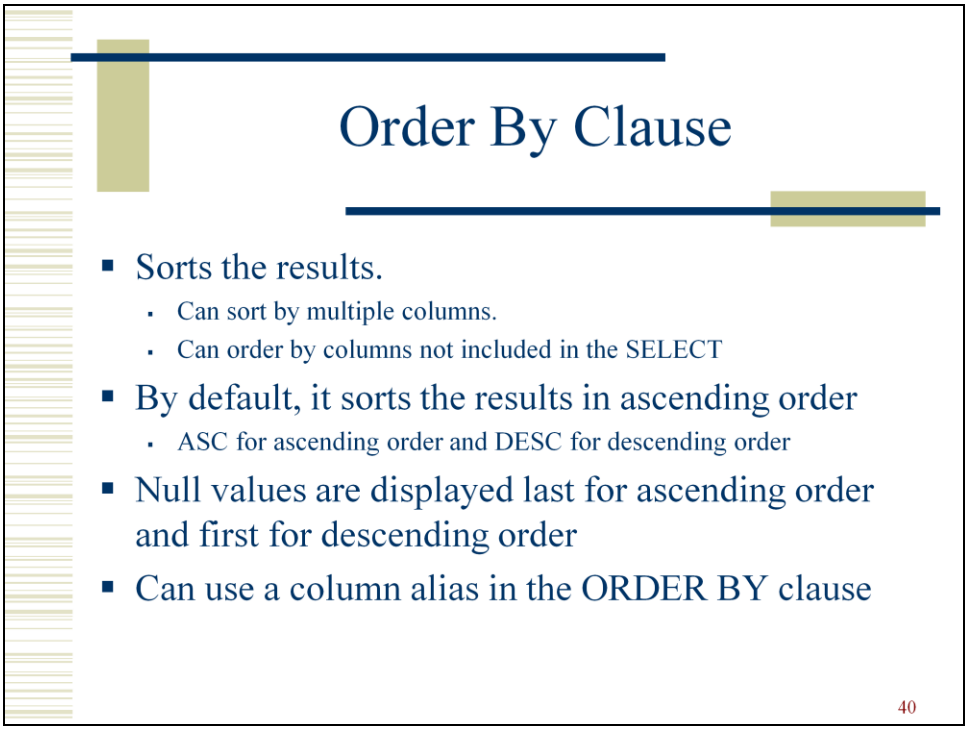
Group By Clause

- Group By

```
SQL> SELECT deptno, sum(sal)
     FROM emp
     GROUP BY deptno;
```

```
DEPTNO    SUM(SAL)
------    -------
    10       8750
    20      10875
    30       9400
```

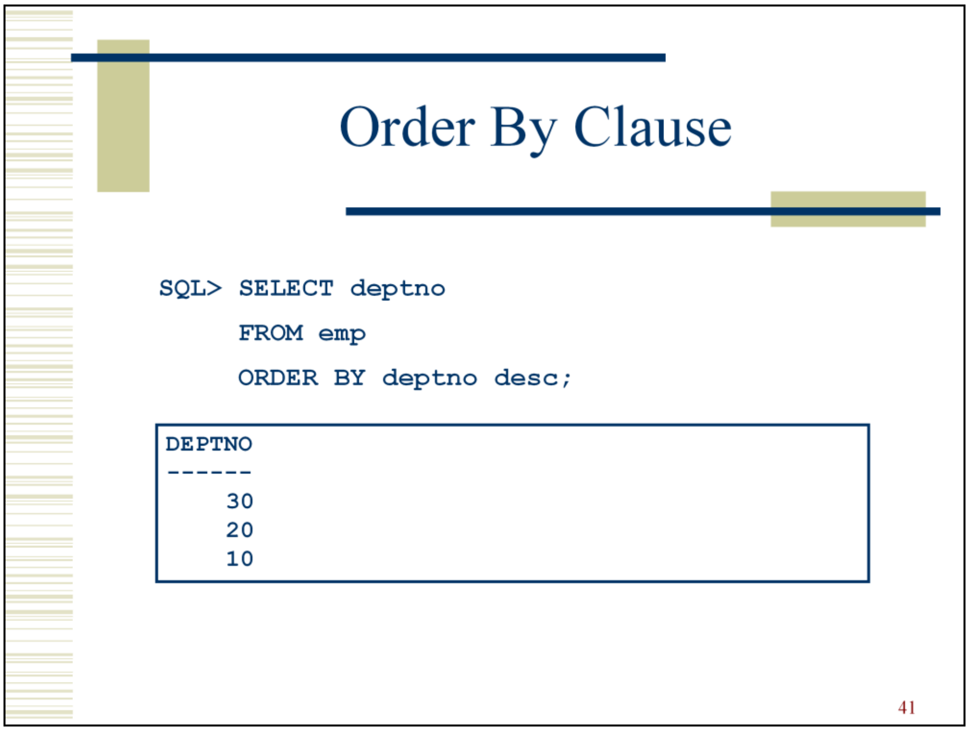
39



Order By Clause

- Sorts the results.
  - Can sort by multiple columns.
  - Can order by columns not included in the SELECT
- By default, it sorts the results in ascending order
  - ASC for ascending order and DESC for descending order
- Null values are displayed last for ascending order and first for descending order
- Can use a column alias in the ORDER BY clause

40



Order By Clause

```
SQL> SELECT deptno
     FROM emp
     ORDER BY deptno desc;
```

```
DEPTNO
------
    30
    20
    10
```

41