Logical Operators

NEW TERM *Logical operators* are those operators that use SQL keywords to make compar-isons instead of symbols. The logical operators covered in the following subsections are

- IS NULL
- BETWEEN
- IN
- LIKE
- EXISTS
- UNIQUE
- ALL and ANY

IS NULL

The NULL operator is used to compare a value with a NULL value. For example, you might look for employees who do not have a pager by searching for NULL values in the PAGER column of the EMPLOYEE_TBL table.

The following example shows comparing a value to a NULL value:

Example	*Meaning*
WHERE SALARY IS NULL	Salary has no value

The following example does not find a NULL value:

Example	*Meaning*
WHERE SALARY = NULL	Salary has a value containing the letters N-U-L-L

INPUT

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME, PAGER

FROM EMPLOYEE_TBL

WHERE PAGER IS NULL;
```

OUTPUT

```
EMP_ID    LAST_NAM FIRST_NA PAGER
--------- -------- -------- -----
311549902 STEPHENS TINA
442346889 PLEW     LINDA
220984332 WALLACE  MARIAH
443679012 SPURGEON TIFFANY

4 rows selected.
```

Understand that the literal word "null" is different than a `NULL` value. Examine the following example:

```
SELECT EMP_ID, LAST_NAME, FIRST_NAME, PAGER

FROM EMPLOYEE_TBL

WHERE PAGER = NULL;
```

```
no rows selected.
```

BETWEEN

The `BETWEEN` operator is used to search for values that are within a set of values, given the minimum value and the maximum value. The minimum and maximum values are included as part of the conditional set.

Example	*Meaning*
`WHERE SALARY BETWEEN '20000' AND '30000'`	The salary must fall between 20000 and 30000, including the values `20000` and `30000`

```
SELECT *

FROM PRODUCTS_TBL

WHERE COST BETWEEN 5.95 AND 14.5;
```

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------------ ------
222        PLASTIC PUMPKIN 18 INCH         7.75
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
1234       KEY CHAIN                       5.95

4 rows selected.
```

Notice that the values `5.95` and `14.5` are included in the output.

Note

`BETWEEN` is inclusive and therefore includes the minimum and maximum values in the query results.

IN

The `IN` operator is used to compare a value to a list of literal values that have been specified. For `TRUE` to be returned, the compared value must match at least one of the values in the list.

Examples	*Meaning*
`WHERE SALARY IN('20000', '30000', '40000')`	The salary must match one of the values `20000`, `30000`, or `40000`

INPUT

```
SELECT *

FROM PRODUCTS_TBL

WHERE PROD_ID IN ('13','9','87','119');
```

OUTPUT

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------------ ------
119        ASSORTED MASKS                  4.95
87         PLASTIC SPIDERS                 1.05
9          CANDY CORN                      1.35
13         FALSE PARAFFIN TEETH            1.1

4 rows selected.
```

Using the `IN` operator can achieve the same results as using the `OR` operator and can return the results more quickly.

LIKE

The `LIKE` operator is used to compare a value to similar values using wildcard operators. There are two wildcards used in conjunction with the `LIKE` operator:

- The percent sign (`%`)
- The underscore (_)

The percent sign represents zero, one, or multiple characters. The underscore represents a single number or character. The symbols can be used in combinations.

Examples are

`WHERE SALARY LIKE '200%'` Finds any values that start with `200`

`WHERE SALARY LIKE '%200%'` Finds any values that have `200` in any position

`WHERE SALARY LIKE '_00%'` Finds any values that have `00` in the second and third positions

`WHERE SALARY LIKE '2_%_%'` Finds any values that start with `2` and are at least three characters in length

`WHERE SALARY LIKE '%2'` Finds any values that end with `2`

`WHERE SALARY LIKE '_2%3'` Finds any values that have a `2` in the second position and end with a `3`

`WHERE SALARY LIKE '2___3'` Finds any values in a five-digit number that start with `2` and end with `3`

The following example shows all product descriptions that end with the letter *S* in uppercase:

INPUT

```
SELECT PROD_DESC

FROM PRODUCTS_TBL

WHERE PROD_DESC LIKE '%S';
```

OUTPUT

```
PROD_DESC
------------------
LIGHTED LANTERNS
ASSORTED COSTUMES
PLASTIC SPIDERS
ASSORTED MASKS

4 rows selected.
```

The following example shows all product descriptions whose second character is the letter *S* in uppercase:

INPUT

```
SELECT PROD_DESC

FROM PRODUCTS_TBL

WHERE PROD_DESC LIKE '_S%';
```


```
PROD_DESC
------------------
ASSORTED COSTUMES
ASSORTED MASKS

2 rows selected.
```

EXISTS

The `EXISTS` operator is used to search for the presence of a row in a specified table that meets certain criteria.

Example	*Meaning*
`WHERE EXISTS (SELECT EMP_ID FROM EMPLOYEE_TBL WHERE EMPLOYEE_ID = '333333333')`	Searching to see whether the `EMP_ID` `3333333333` is in the `EMPLOYEE_TBL`

The following example is a form of a subquery, which is further discussed during Hour 14, "Using Subqueries to Define Unknown Data."


```
SELECT COST

FROM PRODUCTS_TBL

WHERE EXISTS ( SELECT COST

FROM PRODUCTS_TBL

WHERE COST > 100 );
```


```
No rows selected.

----------
```

There were no rows selected because no records existed where the cost was greater than 100.

Consider the following example:

INPUT

```
SELECT COST

FROM PRODUCTS_TBL

WHERE EXISTS ( SELECT COST

FROM PRODUCTS_TBL

WHERE COST < 100 );
```

OUTPUT

```
COST
----------
     29.99
      7.75
       1.1
      14.5
        10
      1.35
      1.45
      1.05
      4.95
      5.95
     59.99

11 rows selected.
```

The cost was displayed for records in the table because records existed where the product cost was less than 100.

UNIQUE

The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).

Example	*Meaning*
`WHERE UNIQUE (SELECT SALARY FROM EMPLOYEE_TBL WHERE EMPLOYEE_ID = '333333333')`	Testing SALARY to see whether there are duplicates

ALL and ANY Operators

The ALL operator is used to compare a value to all values in another value set.

Example	*Meaning*
`WHERE SALARY > ALL SALARY (SELECT FROM EMPLOYEE_TBL WHERE CITY = 'INDIANAPOLIS')`	Testing SALARY to see whether it is greater than all salaries of the employees living in Indianapolis


```
SELECT *

FROM PRODUCTS_TBL

WHERE COST > ALL ( SELECT COST

FROM PRODUCTS_TBL

WHERE COST < 10 );
```


```
PROD_ID    PROD_DESC                      COST
---------- ------------------------------ ------
11235      WITCHES COSTUME                29.99
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
2345       OAK BOOKSHELF                  59.99

4 rows selected.
```

In this output, there were five records that had a cost greater than the cost of all records having a cost less than 10.

The `ANY` operator is used to compare a value to any applicable value in the list according to the condition.

Example

```
WHERE SALARY > ANY (SELECT SALARY FROM
EMPLOYEE_TBL WHERE CITY = 'INDIANAPOLIS')
```

Meaning

Testing `SALARY` to see whether it is greater than any of the salaries of employees living in Indianapolis


```
SELECT *

FROM PRODUCTS_TBL

WHERE COST > ANY ( SELECT COST

FROM PRODUCTS_TBL

WHERE COST < 10 );
```

OUTPUT

```
PROD_ID    PROD_DESC                      COST
---------- ------------------------------ ------
11235      WITCHES COSTUME                29.99
222        PLASTIC PUMPKIN 18 INCH         7.75
13         FALSE PARAFFIN TEETH            1.1
90         LIGHTED LANTERNS               14.5
15         ASSORTED COSTUMES              10
9          CANDY CORN                      1.35
6          PUMPKIN CANDY                   1.45
119        ASSORTED MASKS                  4.95
1234       KEY CHAIN                       5.95
2345       OAK BOOKSHELF                  59.99

10 rows selected.
```

In this output, more records were returned than when using `ALL`, because the cost only had to be greater than any of the costs that were less than 10. The one record that was not displayed had a cost of 1.05, which was not greater than any of the values less than `10` (which was, in fact, 1.05) .