Table of Contents

1. Compound criteria

For more interesting queries, we can use compound criteria. These are criteria that contain multiple conditions joined with the logical operators AND, OR, and NOT.

1.1. The AND logical operator

With this operator, the compound test has a true value if both conditions are true.

Demo 01: We want to see employees who have a salary greater than 15000

```
select emp_id, name_last as "Employee",  salary, dept_id
from emp_employees
where salary > 15000;
```

```
    EMP_ID Employee                     SALARY    DEPT_ID
---------- ------------------------ ---------- ----------
       100 King                         100000         10
       101 Koch                          98005         30
       102 D'Haa                         60300        215
       145 Russ                          59000         80
       146 Partne                        88954        215
       108 Green                         62000         30
       205 Higgs                         75000         30
       162 Holme                         98000         35
       200 Whale                         65000         35
. . .
```

Demo 02: We want to see employees in dept 30 who have a salary greater than 15000. A row has to pass both tests to be included in the result set.

```
select emp_id, name_last as "Employee",  salary, dept_id
from emp_employees
where salary > 60000
AND dept_id =30;
```

```
    EMP_ID Employee                     SALARY    DEPT_ID
---------- ------------------------ ---------- ----------
       101 Koch                          98005         30
       108 Green                         62000         30
       205 Higgs                         75000         30
       203 Mays                          64450         30
       109 Fiet                          65000         30
       110 Chen                          60300         30
       206 Geitz                         88954         30
       204 King                          99090         30
```

When we AND in another filter we will generally reduce the number of rows returned by the query.

Demo 03: We want to see jobs that do not seem to be in Sales with a minimum salary more than 40000. We cannot be certain that these are all of the non-sales jobs- just that they are jobs which do not have Sales in the job title. The Like filter is discussed in another document in this unit.

```
select job_id, min_salary, max_salary
from emp_jobs
where job_title NOT LIKE '%Sales%'
AND min_salary > 40000
;
```

```
    JOB_ID MIN_SALARY MAX_SALARY
---------- ---------- ----------
         1     100000     100000
        16      60000     120000
        32      60000
        64      60000
       128      60000
```

Demo 04: This shows employees with a salary between 20000 and 60000 . The Between operator tests True for the end points.

```
select emp_id,name_last as "Employee", salary
from emp_employees
where  salary between 20000 and 60000
order by salary, emp_id;
```

```
    EMP_ID Employee                      SALARY
---------- ------------------------- ----------
       150 Tuck                           20000
       155 Hiller                         29000
       207 Russ                           30000
       145 Russ                           59000
```

Demo 05: If you need to exclude the end point, then use expression > x and expression < y for a strictly greater than test.

```
select emp_id,name_last as "Employee", salary
from emp_employees
where  salary > 20000
AND    salary < 60000
order by salary;
```

```
    EMP_ID Employee                      SALARY
---------- ------------------------- ----------
       155 Hiller                         29000
       207 Russ                           30000
       145 Russ                           59000
```

Demo 06: Avoid writing tests that logically can never have a True value. What value for salary could pass both of these tests?

```
select emp_id,name_last as "Employee", salary
from emp_employees
where  salary < 12000
AND    salary > 25000
order by salary;
```

```
no rows selected
```

Demo 07: You are not limited to combining two tests.

```
select emp_id, name_last as "Employee"
, dept_id, salary, job_id
from emp_employees
where  dept_id = 35
AND salary > 50000
AND job_id in (8, 16)
;
```

```
    EMP_ID Employee                  DEPT_ID     SALARY     JOB_ID
---------- ------------------------- ---------- ---------- ----------
       162 Holme                             35      98000         16
       200 Whale                             35      65000         16
```

1.2. The OR logical operator

With this operator, the compound test has a true value if either one or both conditions are true.

Demo 08: Find employees who work in either dept 20 or 30. It would be better to use an IN operator for this test. Notice that you have to repeat the full test for each OR clause.

```
select emp_id, name_last as "Employee", dept_id
from emp_employees
where  dept_id = 30
OR     dept_id = 20
order by "Employee"
;
```

```
EMP_ID               Employee                 DEPT_ID
-------------------- ------------------------ ----------------------
110                  Chen                     30
109                  Fiet                     30
206                  Geitz                    30
108                  Green                    30
201                  Harts                    20
205                  Higgs                    30
204                  King                     30
101                  Koch                     30
203                  Mays                     30
```

Demo 09: Here we want employees who earn more than 70000

```
select emp_id
, hire_date, salary, job_id
from emp_employees
where salary > 70000
order by emp_id;
```

```
    EMP_ID HIRE_DATE     SALARY     JOB_ID
---------- --------- ---------- ----------
       100 17-JUN-89     100000          1
       101 17-JUN-08      98005         16
       146 29-FEB-12      88954         64
       161 15-JUN-11     120000         16
       162 17-MAR-11      98000         16
       204 15-JUN-13      99090         32
       205 01-JUN-08      75000         16
       206 15-JUN-13      88954         32

8 rows selected.
```

Demo 10: Here we want employees who earn more than 70000 OR are in dept 30.

```
select emp_id
, dept_id, salary, job_id
from emp_employees
where  dept_id = 30
OR     salary > 70000
order by emp_id
;
```

```
    EMP_ID    DEPT_ID     SALARY     JOB_ID
---------- ---------- ---------- ----------
       100         10     100000          1
       101         30      98005         16
       108         30      62000         16
       109         30      65000         32
       110         30      60300         32
       146        215      88954         64
       161        215     120000         16
       162         35      98000         16
       203         30      64450         16
       204         30      99090         32
       205         30      75000         16
       206         30      88954         32

12 rows selected.
```

Demo 11: Now we add another possibility - that the employee's job id is 8 or 16

```
select  emp_id
, hire_date, salary, job_id
from emp_employees
where  dept_id = 30
OR     salary > 70000
OR     job_id in (8, 16)
order by emp_id
;
```

```
    EMP_ID HIRE_DATE     SALARY     JOB_ID
---------- --------- ---------- ----------
       100 17-JUN-89     100000          1
       101 17-JUN-08      98005         16
       108 14-APR-95      62000         16
       109 29-FEB-12      65000         32
       110 31-DEC-12      60300         32
       146 29-FEB-12      88954         64
       150 28-OCT-01      20000          8
       155 05-MAR-04      29000          8
       161 15-JUN-11     120000         16
       162 17-MAR-11      98000         16
       200 17-JUN-11      65000         16
       203 30-JUN-10      64450         16
       204 15-JUN-13      99090         32
       205 01-JUN-08      75000         16
       206 15-JUN-13      88954         32
       207 17-JUN-11      30000          8

16 rows selected.
```

With each additional OR clause we add, we have the potential of having more rows match.

Demo 12: We have query for max_salary >= 20000
Here we are also including the nulls with an IS NULL test

```
select job_id, job_title, min_salary, max_salary
from emp_jobs
where max_salary >= 20000
OR    max_salary is null;
```

```
    JOB_ID JOB_TITLE       MIN_SALARY MAX_SALARY
---------- --------------- ---------- ----------
         1 President           100000     100000
         2 Marketing             5000      75000
         4 Sales Manager        15000      60000
         8 Sales Rep            10000      30000
        16 Programmer           60000     120000
        32 Code Debugger        60000
        64 DBA                  60000
       128 RD                   60000

8 rows selected.
```

1.3. The NOT logical operator

The NOT operator works on a single test and reverses the value of that test. The NOT test is often used in combination with AND or OR tests.

Demo 13: We want employees who are **not** in department 20 or 30.

```
select emp_id, name_last as "Employee", dept_id
from emp_employees
where NOT dept_id IN ( 30, 20)
order by "Employee"
;
```

```
    EMP_ID Employee                    DEPT_ID
---------- ------------------------ ----------
       102 D'Haa                           215
       161 Dewal                           215
       160 Dorna                           215
       104 Ernst                           210
       155 Hiller                           80
       162 Holme                            35
       103 Hunol                           210
       100 King                             10
       146 Partne                          215
       207 Russ                             35
       145 Russ                             80
       150 Tuck                             80
       200 Whale                            35

13 rows selected.
```

The above test could also be written as `Where dept_id NOT IN ( 30, 20)` and I think that is easier to read. Note that NOT IN is closer to the way the task is described. I would also encourage you to use `Where salary not between 10000 and 20000` instead of `Where NOT salary between 10000 and 20000.`

Using the not operator before the tests means that your mind has to keep track of the `NOT` while it reads the rest of the test.

Take extra care when using two NOT words in the same test- often people get the logic of double negatives wrong.

2. Hierarchy of evaluation of the logical operators

If you write a criterion that includes more than one logical operator, you need to be concerned about the hierarchy of evaluation. The order of operations is: first the NOT operators are evaluated then the ANDs and then the ORs. Parentheses are used to change the order of operations.

Suppose we want to see products that are either pet supplies or sporting goods that cost less than 100. This is an ambiguous statement. Assume this essentially means we want the cheaper sporting good and the cheaper pet supplies items.

Demo 14: This query following the wording of the task description but does not do the job. We have two Pet items that cost more than $100.

```
select prod_id, prod_list_price, catg_id
from prd_products
where  catg_id = 'PET' OR catg_id = 'SPG'
AND    prod_list_price < 100;
```

```
   PROD_ID PROD_LIST_PRICE CATG_I
---------- --------------- ------
      1020           12.95 SPG
      1030           29.95 SPG
      1140           14.99 PET
      1141           99.99 PET
      1142             2.5 PET
      1143             2.5 PET
      1150            4.99 PET
      1151           14.99 PET
      1152              55 PET
      4567          549.99 PET
      4568          549.99 PET
      4576           29.95 PET
      4577           29.95 PET
13 rows selected
```

Demo 15: If we reverse the testing of the two categories, we get sporting goods items that cost more than $100. That is not right.

```
select prod_id, prod_list_price, catg_id
from prd_products
where  catg_id = 'SPG' OR catg_id = 'PET'
AND    prod_list_price < 100;
```

```
   PROD_ID PROD_LIST_PRICE CATG_I
---------- --------------- ------
      1010             150 SPG
      1020           12.95 SPG
      1030           29.95 SPG
      1140           14.99 PET
      1141           99.99 PET
      1142             2.5 PET
      1143             2.5 PET
      1150            4.99 PET
      1151           14.99 PET
      1152              55 PET
      1040          349.95 SPG
      1050          269.95 SPG
      1060          255.95 SPG
      4576           29.95 PET
      4577           29.95 PET
15 rows selected.
```

What is happening here is that we have an AND operator and an OR operator. The rules of precedence is that the AND operator is evaluated first. So the second of these where clauses

```
where catg_id = 'SPG' or catg_id = 'PET' and prod_list_price < 100;
```

is evaluated as shown here and all of the sporting goods items are returned and Pet supplies that cost more than $100 are returned.

```
where catg_id = 'SPG' or (catg_id = 'PET' and prod_list_price < 100);
```

We can use parentheses to change the order of evaluation

Demo 16: Adding the parentheses gives us the correct result.

```
select prod_id, prod_list_price, catg_id
from prd_products
where  (catg_id = 'SPG' OR catg_id = 'PET')
AND    prod_list_price < 100;
```

```
   PROD_ID PROD_LIST_PRICE CATG_I
---------- --------------- ------
      1020           12.95 SPG
      1030           29.95 SPG
      1140           14.99 PET
      1141           99.99 PET
      1142             2.5 PET
      1143             2.5 PET
      1150            4.99 PET
      1151           14.99 PET
      1152              55 PET
      4576           29.95 PET
      4577           29.95 PET

11 rows selected.
```

Demo 17: It is better to use the IN operator, avoiding the AND/OR Issue.

```
select  prod_id, prod_list_price, catg_id
from prd_products
where catg_id IN ( 'SPG', 'PET')
AND   prod_list_price < 100;
```

3. DeMorgan's laws

Often, there is more than one way to write a complex logical expression. The following equivalencies are known as DeMorgan's Laws.

Where expP and expQ represent logical expressions

NOT (expP AND expQ)	is equivalent to	NOT expP OR NOT expQ
NOT (expP OR expQ)	is equivalent to	NOT expP AND NOT expQ

Demo 18: These are equivalent queries. Prod_list_price and catg_id are not null in the products table.

```
select prod_id
, prod_desc
, prod_list_price, catg_id
from prd_products
where NOT( prod_list_price < 300 OR catg_id = 'APL');
```

```
select prod_id
, prod_desc
, prod_list_price, catg_id
from prd_products
where NOT( prod_list_price < 300)  AND NOT( catg_id = 'APL');

select prod_id
, prod_desc
, prod_list_price, catg_id
from prd_products
where  prod_list_price >= 300  AND  catg_id != 'APL';
```

4. Three-way logic and truth tables

Generally we think of logical expressions having two possible values — True and False. Because database systems allow the use of Null, we have to be concerned with three logical values —True, False, and Unknown. Suppose we have a row in the jobs table with no value for the attribute max_salary , and we evaluate the logical expression: max_salary > 25000 the value of the expression is Unknown for that row. If you are executing a query with a Where clause, if the value of the test is Unknown, the row is not returned.

Remember, NULL is a data value, UNKNOWN is a logical value.

These are the truth tables for the operators NOT, AND, and OR.

The evaluation of the True and False cases are straight forward. With the NOT operator, if I do not know the value of an expression is True or False then I do not know if the negation of that expression is True or False.

NOT	
True	False
Unknown	Unknown
False	True

For the AND operator to Return True both of the operators must have a True value. So if one of the operands is True and the other is unknown, then I cannot know if the ANDed expression is true- so the value is unknown. But if one of the operands is False, then the ANDed expression cannot be true and we know its value is False.

AND	True	Unknown	False
True	True	Unknown	False
Unknown	Unknown	Unknown	False
False	False	False	False

For the OR operator to Return True at least one of the operators must have a True value. So if one of the operands is True and the other is unknown, then the ORed expression is TRUE. If one of the operands is False and the other is unknown then I cannot know the value of the Ored expression and its value is Unknown.

OR	True	Unknown	False
True	True	True	True
Unknown	True	Unknown	Unknown
False	True	Unknown	False

Demo 19: Cust id 402500 has a null for the credit limit column

```
select customer_id, customer_name_last
from cust_customers
where customer_id = 402500;
```

```
   CUSTOMER_ID CUSTOMER_NAME_LAST
-------------- -------------------------
        402500 Jones

1 row selected.
```

Demo 20: Cust id 402500 passes the first of these tests; the second test for that row has a value of unknown and therefore Cust id 402500 is not returned by this query since the tests are ANDed

```
select *
from cust_customers
where customer_id < 403050 AND customer_credit_limit < 1000;
```

```
   CUSTOMER_ID CUSTOMER_NAME_LAST     CUSTOMER_NAME_LAST    CUSTOMER_CREDIT_LIMIT
-------------- ---------------------  --------------------- ---------------------
        400801 Washington             Geo                                   750
        401250 Morse                  Samuel                                750
        402100 Morise                 William                               750
        402110 Coltrane               John                                  750
        402120 McCoy                  Tyner                                 750

5 rows selected.
```

Demo 21: Cust id 402500 passes the first of these tests; the second test for that row has a value of unknown and Cust id 402500 is returned by this query since the tests are ORed and Cust id passed at least one of the tests

```
select *
from cust_customers
where customer_id < 403050 OR customer_credit_limit < 1000;
```

```
   CUSTOMER_ID CUSTOMER_NAME_LAST     CUSTOMER_NAME_LAST    CUSTOMER_CREDIT_LIMIT
-------------- ---------------------  --------------------- ---------------------
        401250 Morse                  Samuel                                750
        401890 Northrep               William                              1750
        402100 Morise                 William                               750
        402110 Coltrane               John                                  750
        402120 McCoy                  Tyner                                 750
        402500 Jones                  Elton John
        403000 Williams               Sally                                6000
        403010 Otis                   Elisha                               6000
        400801 Washington             Geo                                   750
        400300 McGold                 Arnold                               6000

10 rows selected.
```