Home page

pgrocer.net

Priscilla Grocer

Chair, Computer Information Systems, Bristol Community College

CIS120/17 CIS122/44 CIS150/50 CIS256/67 CIS270/54 CIS272 should see me... Need help or support?

p. grocer @ rcn . com

This is the email to use to ask questions and submit work.

Previous Semesters Advisement for CIS/CIT NSF Support Postings/Notes My Schedule

Note that last semesters courses are available under Previous Semesters

Computer Information Systems, Bristol Community College, Fall River, MA Office: K112 Phone: 508-678-2811 ext 2403 email: Priscilla.Grocer@bristolcc.edu

Type here to search

9:35 AM
9/3/2019

Notes for CIS50

www.pgrocer.net/Cis50/cis50notes.html

Notes for CIS50 - Oracle and SQL

Notes and handouts	Link to retrieve
Introduction to Relational Databases	Student Payroll
Introduction to Oracle and SQL	Introduction to SQL in Oracle Create a table in SQL SQL Conditions Working with SQL
SQL and SQL*Plus	More on maintaining an SQL table SQL functions More SQL functions Group SQL functions Introduction to multiple tables (relational) - joins Relational database examples Oracle Join statement Subqueries More on subqueries Using input variables More on variables with Oracle's SQL*Plus Table maintenance revisited (again) Continuation of table maintenance revisited (again) Introduction to views Introduction to reports Indexes in Oracle - An introduction More on Primary and Foreign keys Notes on key questions More on views Information on database objects Script for input data
Older versions	Introduction to user security Copying a Table from DEMO Exporting an Oracle table (older version)
PL/SQL	Introduction to PL/SQL More examples of inner/outer procedures Introduction to IF statements More on PL/SQL IF statements Introduction to loops Data manipulation in PL/SQL - introduction Record and table types Introduction to Cursors Example 1 using cursors Continuing example 1 using cursors Another set of cursor examples Introduction to exceptions Introduction to procedures Introduction to functions

We are going over Create a table in SQL and SQL Conditions.

Type here to search

9:48 AM
9/3/2019

CIS150/50 Course Page

www.pgrocer.net/PFGCIS150.html

Oracle and SQL CIS150/50

About This Course:
- Need help or support?
- Introduction to CIS150
- Syllabus
- Withdrawal Policy as of Fall 2017

Site Resources:
- Notes and Handouts
- Presentations
- SmartBoard Presentations
- Audio

Weekly Schedule:
- Weekly schedule - CIS150/50

Using Oracle:
- Installation and login

Links:
- Links & Tutorials

The information at this site is in draft form and is the property of Priscilla Grocer. The information may not be duplicated or distributed without her permission. ©Copyright 2019

Oracle and SQL (CIS150/50)

Weekly Schedule

Please do not change your linux or Oracle passwords.

Essentially there are three ways for you to be able to work with Oracle:
1) Download the full version of Oracle and install it on your computer and work locally - that is what I have done
2) Work in the BCC labs with the BCC server installed Oracle
3) Access the BCC server installed Oracle from home using putty - there are instructions for that by Eric at my site.

Remember, all assignments including inclass assignments must be done by everyone - that includes online students and students who missed the class.. Play computer means to figure out the results yourself by evaluating the data rather than running the query. When you are asked to write code you should also show the results that the code generated unless specifically told not to.

Some students have encountered problems in putty because something has expired. If you encounter a problem, send me an email and I will ask IT to fix it.

If you get an assignment back without a grade you need to fix it and resubmit to get a grade.

Schedule by week	Information to cover
	Note that the audio and Smartboard presentations for each class are posted within a couple of days. Please check the weekly schedule multiple times a week, I sometimes add things during the course of the week. You need to pass in the code you write and the output that is produced unless you are told otherwise. Sometimes this involves showing data that was written to a file/table. Please keep copies of all work you submit until you receive your final grade at the end of the semester. Comment on how I want the assignment: You should take the commands you do in SQL and the results and paste them into a word document. It should be similiar to the way I prepared the notes. You then send me the word document. I do not want you to show the screen itself - they are really hard to read!
	I recommend one of the text books in the syllabus, but a textbook is not required. There is enough material on the web and in my notes that I decided to make the book a recommendation. Please note that many students have found it useful to have a copy of the text. I do recommend that you get one of these books or another book dealing with Oracle SQL (we are using version 11g or 12c this semester) as a reference. We will not do chapter by chapter coverage, the text gives you additional information on topics and a reference if you have questions. ional database information **LOGIN information:** You need to decide if you want to download Oracle, use the BCC labs or login from home using Putty. Note that there is a reference to help you download and instal under Using Oracle: Installation and login. **Downloading** Be very careful when you install, Oracle leaves a lot of footprints that makes reinstalling very difficult. You should have received the login information you need for this course. If not, let me know. PuTTY is available in the labs. Two sites that can be used to download PuTTY are: **PuTTY** **PuTTY** **Using Putty (copy and paste)** Basic steps to login to putty and get to Oracle: 1) Start putty and put in the hostname as cisweb.bristolcc.edu and the port as 220 2) At the login screen enter your username and your Linux password. 3) When you see that you are at cisweb, enter sqlplus12c.sh

Type here to search

9:49 AM
9/3/2019

Title: Sep 3-9:49 AM (3 of 22)

Create a table

www.pgrocer.net/Cis50/oraclecr.html

Creating an Oracle database table:

This handout covers the basics of creating a database table. We will revisit this topic later and discuss more sophisticated elements at that time.

The CREATE command is used to create a database table. For this example, I will create a table called first_table. The table will contain an identification number, a name, a code, a date and an amount without decimals and an amount with decimals. The data types that are used for this table are described below:

Data Type	Description
VARCHAR2(size)	Variable length character data - maximum size 4000
CHAR(size)	Fixed length character data - maximum size of 2000
NUMBER(len,dec)	Numeric data with length and number of decimal digits
DATE	Stores date

The CREATE statement has the format CREATE TABLE followed by the name of the table. The fields/columns are listed inside a set of parenthesis with the column name and type. The columns are separated by commas. The example described is illustrated below.

SQL CODE:

```
SQL> CREATE TABLE first_table
  2  (idno varchar2(4),
  3  name varchar2(15),
  4  first_code char(2),
  5  first_date date,
  6  first_wholeamt number(3),
  7  first_decimal number(6,2));

Table created.
```

Creating a database table with a variety of data types and then describing the table using DESCRIBE. Note that you can use DESC.

To see the layout of the table that was created, you can use the DESCRIBE command as shown below:

SQL CODE:

```
SQL> DESCRIBE first_table;
 Name                            Null?    Type
 ------------------------------- -------- ----
 IDNO                                     VARCHAR2(4)
 NAME                                     VARCHAR2(15)
 FIRST_CODE                               CHAR(2)
 FIRST_DATE                               DATE
 FIRST_WHOLEAMT                           NUMBER(3)
 FIRST_DECIMAL                            NUMBER(6,2)
```

To put data into the table, you use the INSERT command. The format is:

INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)

SQL CODE:

Type here to search

9:59 AM
9/3/2019

Create a table

www.pgrocer.net/Cis50/oraclecr.html

SQL CODE:

```
SQL> CREATE TABLE first_table
  2  (idno varchar2(4),
  3  name varchar2(15),
  4  first_code char(2),
  5  first_date date,
  6  first_wholeamt number(3),
  7  first_decimal number(6,2));

Table created.
```

To see the layout of the table that was created, you can use the DESCRIBE command as shown below:

SQL CODE:

```
SQL> DESCRIBE first_table;
 Name                            Null?    Type
 ------------------------------- -------- ----
 IDNO                                     VARCHAR2(4)
 NAME                                     VARCHAR2(15)
 FIRST_CODE                               CHAR(2)
 FIRST_DATE                               DATE
 FIRST_WHOLEAMT                           NUMBER(3)
 FIRST_DECIMAL                            NUMBER(6,2)
```

To put data into the table, you use the INSERT command. The format is:

INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)

SQL CODE:

```
SQL> INSERT INTO first_table
  2  values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);

1 row created.

SQL> INSERT INTO first_table
  2  values('2222','John Costa','CI','15-FEB-97',654,765.23);

1 row created.

SQL> INSERT INTO first_table
  2  values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);

1 row created.

SQL> INSERT INTO first_table
  2  values('4444','Stephen Woods','CI','30-APR-98',4,4.59);

1 row created.
```

These four records have now been inserted in the table. We can now list the table.

Type here to search

10:02 AM
9/3/2019

Create a table

www.pgrocer.net/Cis50/oraclecr.html

To see the layout of the table that was created, you can use the DESCRIBE command as shown below:

SQL CODE:

```
SQL> DESCRIBE first_table;
 Name                            Null?    Type
 ------------------------------- -------- ----
 IDNO                                     VARCHAR2(4)
 NAME                                     VARCHAR2(15)
 FIRST_CODE                               CHAR(2)
 FIRST_DATE                               DATE
 FIRST_WHOLEAMT                           NUMBER(3)
 FIRST_DECIMAL                            NUMBER(6,2)
```

To put data into the table, you use the INSERT command. The format is:

INSERT INTO table
values(list the fields that are going into each column for this row - varchar2, char and date fields must be enclosed in single quotes - numeric columns are not enclosed in quotes)

SQL CODE:

```
SQL> INSERT INTO first_table
  2  values('1111','Susan Johnson','CI','12-JAN-96',123,12.45);

1 row created.

SQL> INSERT INTO first_table
  2  values('2222','John Costa','CI','15-FEB-97',654,765.23);

1 row created.

SQL> INSERT INTO first_table
  2  values('3333','Linda Higgins','CF','23-MAR-95',12,1.48);

1 row created.

SQL> INSERT INTO first_table
  2  values('4444','Stephen Woods','CI','30-APR-98',4,4.59);

1 row created.
```

These four records have now been inserted in the table. We can now list the table.

SQL CODE:

```
SQL> SELECT *
  2  FROM first_table;

IDNO NAME            FI FIRST_DAT FIRST_WHOLEAMT FIRST_DECIMAL
---- --------------- -- --------- -------------- -------------
1111 Susan Johnson   CI 12-JAN-96            123         12.45
2222 John Costa      CI 15-FEB-97            654        765.23
3333 Linda Higgins   CF 23-MAR-95             12          1.48
4444 Stephen Woods   CI 30-APR-98              4          4.59
```

Type here to search

10:07 AM
9/3/2019

Notes for CIS50

www.pgrocer.net/Cis50/cis50notes.html

Notes for CIS50 - Oracle and SQL

Notes and handouts	Link to retrieve
Introduction to Relational Databases	Student Payroll
Introduction to Oracle and SQL	Introduction to SQL in Oracle Create a table in SQL SQL Conditions Working with SQL
SQL and SQL*Plus	More on maintaining an SQL table SQL functions More SQL functions Group SQL functions Introduction to multiple tables (relational) - joins Relational database examples Oracle Join statement Subqueries More on subqueries Using input variables More on variables with Oracle's SQL*Plus Table maintenance revisited (again) Continuation of table maintenance revisited (again) Introduction to views Introduction to reports Indexes in Oracle - An introduction More on Primary and Foreign keys Notes on key questions More on views Information on database objects Script for input data
Older versions	Introduction to user security Copying a Table from DEMO Exporting an Oracle table (older version)
PL/SQL	Introduction to PL/SQL More examples of inner/outer procedures Introduction to IF statements More on PL/SQL IF statements Introduction to loops Data manipulation in PL/SQL - introduction Record and table types Introduction to Cursors Example 1 using cursors Continuing example 1 using cursors Another set of cursor examples Introduction to exceptions Introduction to procedures Introduction to functions

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

SQL and Conditions

Additional records were added to the database before doing these examples. The donor database now contains the following:

SQL CODE:

```
SQL> SELECT *
  2  FROM DONOR;

IDNO  NAME            STADR           CITY       ST ZIP   DATEFST      YRGOAL CONTACT
----- --------------- --------------- ---------- -- ----- --------- --------- ------------
11111 Stephen Daniels 123 Elm St      Seekonk    MA 02345 03-JUL-98       500 John Smith
12121 Jennifer Ames   24 Benefit St   Providence RI 02045 24-MAY-97       400 Susan Jones
22222 Carl Hersey     24 Benefit St   Providence RI 02045 03-JAN-98           Susan Jones
23456 Susan Ash       21 Main St      Fall River MA 02720 04-MAR-92       100 Amy Costa
33333 Nancy Taylor    26 Oak St       Fall River MA 02720 04-MAR-92        50 John Adams
34567 Robert Brooks   36 Pine St      Fall River MA 02720 04-APR-98        50 Amy Costa

6 rows selected.

SQL>
```

One of the major things that needs to be done in a query is setting certain criteria and only retrieving rows that meet the specified criteria. In SQL, this is accomplished with the WHERE clause.

For example, let's say I want to show only information for donors that live in Rhode Island. For this selection, I decide that I want to see the identification number, the name, the city and the state.

SELECT idno, name, city, state
FROM donor
WHERE state = 'RI';

The where clause lets me test each record in the table against the criteria state = 'RI'. If the state does not equal RI, the record will not be displayed. Notice that the string literal RI is enclosed in single quotes.

SQL CODE:

Only the specified fields will show for each record that meets the criteria of state equal to RI.

```
SQL> SELECT idno, name, city, state
  2  FROM donor
  3  WHERE state = 'RI';

IDNO  NAME            CITY       ST
----- --------------- ---------- --
12121 Jennifer Ames   Providence RI
22222 Carl Hersey     Providence RI

SQL>
```

The comparison operators that are used in SQL are:

Operator	Definition
=	Equal to
>	Greater than
<	Less than

Type here to search

10:09 AM
9/3/2019

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

The comparison operators that are used in SQL are:

Operator	Definition
=	Equal to
>	Greater than
<	Less than
<>	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

SELECT idno, name, yrgoal
FROM donor
WHERE yrgoal >= 100;

In the example above, I want to show idno, name, yrgoal for all records where the yrgoal is either = to or greater than 100. Notice that because yrgoal is a numeric field and this is numeric data, the 100 is not enclosed in quotes.

SQL CODE:

```
SQL> SELECT idno, name, yrgoal
  2  FROM donor
  3  WHERE yrgoal >= 100;

IDNO  NAME                YRGOAL
----- --------------- ---------
11111 Stephen Daniels       500
12121 Jennifer Ames         400
23456 Susan Ash             100

SQL>
```

As in all languages, comparisons should be of like data types. String values and date values are enclosed in the single quote marks and the information enclosed is case sensitive. Numeric data can be enclosed in quotes if the field be compared to is a character/string field. In other words, if I define idno as character or string data then I can put the number I am comparing in single quotes. The example below is comparing a date field to a specific date. Notice that the default format for dates is DD-MON-YY.

SQL CODE:

```
SQL> SELECT idno, name, datefst
  2  FROM donor
  3  WHERE datefst > '01-JAN-96';

IDNO  NAME            DATEFST
----- --------------- ---------
11111 Stephen Daniels 03-JUL-98
12121 Jennifer Ames   24-MAY-97
22222 Carl Hersey     03-JAN-98
34567 Robert Brooks   04-APR-98
```

In the next example, I want to get all donors who live in Fall River. My problem is that I am not sure how the data has been keyed in: Fall River, FALL RIVER, fall river. Since the data inside the single quotes is case sensitive I need to use the upper or lower case function to temporarily convert the data for a balanced comparison. In the example below, I am converting the data in the table to upper case and then comparing that against FALL RIVER. This guarantees that I am comparing like data. In the following example, I am converting the data in the table to lower case and then comparing that against fall river.

Type here to search

10:11 AM
9/3/2019

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

SELECT idno, name, yrgoal
FROM donor
WHERE yrgoal >= 100;

In the example above, I want to show idno, name, yrgoal for all records where the yrgoal is either = to or greater than 100. Notice that because yrgoal is a numeric field and this is numeric data, the 100 is not enclosed in quotes.

SQL CODE:

```
SQL> SELECT idno, name, yrgoal
  2  FROM donor
  3  WHERE yrgoal >= 100;

IDNO  NAME               YRGOAL
----- --------------- ---------
11111 Stephen Daniels       500
12121 Jennifer Ames         400
23456 Susan Ash             100

SQL>
```

As in all languages, comparisons should be of like data types. String values and date values are enclosed in the single quote marks and the information enclosed is case sensitive. Numeric data can be enclosed in quotes if the field be compared to is a character/string field. In other words, if I define idno as character or string data then I can put the number I am comparing in single quotes. The example below is comparing a date field to a specific date. Notice that the default format for dates is DD-MON-YY.

SQL CODE:

```
SQL> SELECT idno, name, datefst
  2  FROM donor
  3  WHERE datefst > '01-JAN-96';

IDNO  NAME            DATEFST
----- --------------- ---------
11111 Stephen Daniels 03-JUL-98
12121 Jennifer Ames   24-MAY-97
22222 Carl Hersey     03-JAN-98
34567 Robert Brooks   04-APR-98
```

In the next example, I want to get all donors who live in Fall River. My problem is that I am not sure how the data has been keyed in: Fall River, FALL RIVER, fall river. Since the data inside the single quotes is case sensitive I need to use the upper or lower case function to temporarily convert the data for a balanced comparison. In the example below, I am converting the data in the table to upper case and then comparing that against FALL RIVER. This guarantees that I am comparing like data. In the following example, I am converting the data in the table to lower case and then comparing that against fall river.

SQL CODE:

```
SQL> SELECT idno, name, city
  2  FROM donor
  3  WHERE upper(city) = 'FALL RIVER';

IDNO  NAME            CITY
----- --------------- ----------
23456 Susan Ash       Fall River
33333 Nancy Taylor    Fall River
34567 Robert Brooks   Fall River
```

Type here to search

10:12 AM
9/3/2019

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

SQL CODE:

```
SQL> SELECT idno, name, city
  2  FROM donor
  3  WHERE upper(city) = 'FALL RIVER';

IDNO  NAME            CITY
----- --------------- ----------
23456 Susan Ash       Fall River
33333 Nancy Taylor    Fall River
34567 Robert Brooks   Fall River

SQL> SELECT idno, name, city
  2  FROM donor
  3  WHERE lower(city) = 'fall river';

IDNO  NAME            CITY
----- --------------- ----------
23456 Susan Ash       Fall River
33333 Nancy Taylor    Fall River
34567 Robert Brooks   Fall River

SQL>
```

This converts the data in the city field to upper case no matter how it was entered and then it test against the upper case of the city in a literal. The upper is a built in Oracle function. It assures that no matter what case was used in entering Fall River into the table, it will find a match. Note that this can also be done with the function lower.

SQL allows the testing of a field for NULL. Remember NULL means that no value has been assigned to the field and therefore the field cannot be equal to anything or unequal to anything, it is just plain null. The test that will be used in the WHERE clause is the IS NULL test. In the example below, the output shows the row where there is no entry in the yrgoal field, the field is null.

SQL CODE:

```
SQL> SELECT idno, name, city, yrgoal
  2  FROM donor
  3  WHERE yrgoal IS NULL;

IDNO  NAME            CITY          YRGOAL
----- --------------- ---------- ---------
22222 Carl Hersey     Providence
```

SQL supports the logical operators AND, OR and NOT. As you know, with AND all conditions in the AND relationship must be true to select the record and with OR only one of the conditions in the OR relationship has to be true to select a record. With NOT, the condition must not be true to select the record.

In the example below, I am testing for state = 'MA' (note that because MA is a character string it must be enclosed in single quotes) and yrgoal that is either > or = to 100. The greater than or equal to has to be expressed as >=. All rows from the table where both of these conditions are true will be displayed.

SQL CODE:

```
SQL> SELECT idno, name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'MA' AND yrgoal >= 100;

IDNO  NAME            ST    YRGOAL
----- --------------- -- ---------
11111 Stephen Daniels MA       500
23456 Susan Ash       MA       100

SQL>
```

Type here to search

10:14 AM
9/3/2019

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

AND relationship

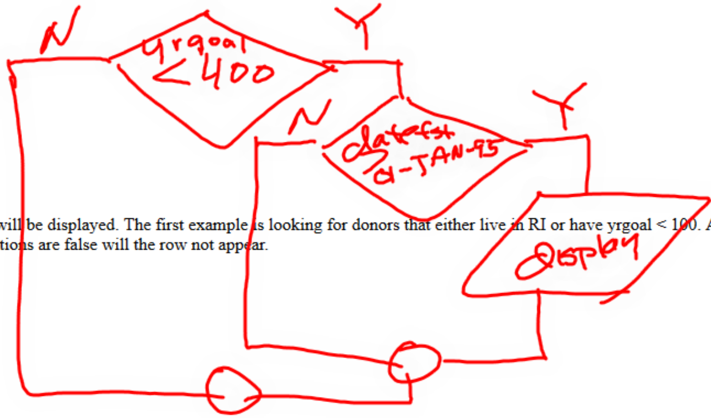
```
SQL> SELECT idno, name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'MA' AND yrgoal >= 100;

IDNO  NAME            ST    YRGOAL
----- --------------- -- ---------
11111 Stephen Daniels MA       500
23456 Susan Ash       MA       100

SQL>
```

In the next example, I am displaying all records where the yrgoal is < 400 and the datefst is > then January 1, 1995. Again notice the data is enclosed in single quotes and it is in the default format of DD-MON-YY.

YY
YN
NY
NN



SQL CODE:

```
SQL> SELECT name, state, datefst, yrgoal
  2  FROM donor
  3  WHERE yrgoal < 400 AND datefst > '01-JAN-95';

NAME            ST DATEFST      YRGOAL
--------------- -- --------- ---------
Robert Brooks   MA 04-APR-98        50
```

The next example uses the logical operator OR. In this case, if either of the two conditions are true then the row/record will be displayed. The first example is looking for donors that either live in RI or have yrgoal < 100. Again, if either condition is true the row will be displayed. If both conditions happen to be true, that is fine and the row will be displayed. Only if both conditions are false will the row not appear.

SQL CODE:

```
SQL> SELECT name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'RI' OR yrgoal < 100;


NAME            ST    YRGOAL
--------------- -- ---------
Jennifer Ames   RI       400
Carl Hersey     RI
Nancy Taylor    MA        50
Robert Brooks   MA        50
```

The second example of the logical OR tests the contact column for each row to see if the name is either Susan Jones or John Adams. If either name is in the contact column than that row will be displayed.

SQL CODE:

```
SQL> SELECT name, state, yrgoal, contact
  2  FROM donor
  3  WHERE contact = 'Susan Jones' OR contact = 'John Adams';

NAME            ST    YRGOAL CONTACT
--------------- -- --------- ------------
Jennifer Ames   RI       400 Susan Jones
Carl Hersey     RI           Susan Jones
Nancy Taylor    MA        50 John Adams
```

Type here to search

10:18 AM
9/3/2019

OR relationship

```
SQL> SELECT idno, name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'MA' AND yrgoal >= 100;

IDNO  NAME            ST    YRGOAL
----- --------------- -- ---------
11111 Stephen Daniels MA       500
23456 Susan Ash       MA       100

SQL>
```

In the next example, I am displaying all records where the yrgoal is < 400 and the datefst is > then January 1, 1995. Again notice the data is enclosed in single quotes and it is in the default format of DD-MON-YY.

SQL CODE:

```
SQL> SELECT name, state, datefst, yrgoal
  2  FROM donor
  3  WHERE yrgoal < 400 AND datefst > '01-JAN-95';

NAME            ST DATEFST      YRGOAL
--------------- -- --------- ---------
Robert Brooks   MA 04-APR-98        50
```

The next example uses the logical operator OR. In this case, if either of the two conditions are true then the row/record will be displayed. The first example is looking for donors that either live in RI or have yrgoal < 100. Again, if either condition is true the row will be displayed. If both conditions happen to be true, that is fine and the row will be displayed. Only if both conditions are false will the row not appear.

SQL CODE:

```
SQL> SELECT name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'RI' OR yrgoal < 100;


NAME            ST    YRGOAL
--------------- -- ---------
Jennifer Ames   RI       400
Carl Hersey     RI
Nancy Taylor    MA        50
Robert Brooks   MA        50
```

The second example of the logical OR tests the contact column for each row to see if the name is either Susan Jones or John Adams. If either name is in the contact column than that row will be displayed.

SQL CODE:

```
SQL> SELECT name, state, yrgoal, contact
  2  FROM donor
  3  WHERE contact = 'Susan Jones' OR contact = 'John Adams';

NAME            ST    YRGOAL CONTACT
--------------- -- --------- ------------
Jennifer Ames   RI       400 Susan Jones
Carl Hersey     RI           Susan Jones
Nancy Taylor    MA        50 John Adams
```

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

```
SQL> SELECT name, state, yrgoal
  2  FROM donor
  3  WHERE state = 'RI' OR yrgoal < 100;


NAME            ST    YRGOAL
--------------- -- ---------
Jennifer Ames   RI       400
Carl Hersey     RI
Nancy Taylor    MA        50
Robert Brooks   MA        50
```

The second example of the logical OR tests the contact column for each row to see if the name is either Susan Jones or John Adams. If either name is in the contact column than that row will be displayed.

SQL CODE:

```
SQL> SELECT name, state, yrgoal, contact
  2  FROM donor
  3  WHERE contact = 'Susan Jones' OR contact = 'John Adams';

NAME            ST    YRGOAL CONTACT
--------------- -- --------- ------------
Jennifer Ames   RI       400 Susan Jones
Carl Hersey     RI           Susan Jones
Nancy Taylor    MA        50 John Adams
```

In the third example, I am again testing the same field. I want to display all records where the yrgoal is either < 100 or null.

SQL CODE:

```
SQL> SELECT name, yrgoal, contact
  2  FROM donor
  3  WHERE yrgoal < 100 or yrgoal IS NULL;

NAME               YRGOAL CONTACT
--------------- --------- ------------
Carl Hersey               Susan Jones
Nancy Taylor           50 John Adams
Robert Brooks          50 Amy Costa
```

When working with NOT, AND, and OR, it is important to remember the order in which the processing occurs:

- NOT is resolved first
- AND is resolved next
- OR is resolved last

Parenthesis can be used to change this order because parenthesis are resolved before things that are not contained in parenthesis.

For example:

```
Condition A AND Condition B OR Condition C can be read as Condition A AND Condition B or just
Condition C and resolves as:

            Condition A AND Condition B
```

Type here to search

10:21 AM
9/3/2019

Title: Sep 3-10:21 AM (14 of 22)

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

```
  3  WHERE contact = 'Susan Jones' OR contact = 'John Adams';

NAME            ST    YRGOAL CONTACT
--------------- -- --------- ------------
Jennifer Ames   RI       400 Susan Jones
Carl Hersey     RI           Susan Jones
Nancy Taylor    MA        50 John Adams
```

In the third example, I am again testing the same field. I want to display all records where the yrgoal is either < 100 or null.

SQL CODE:

```
SQL> SELECT name, yrgoal, contact
  2  FROM donor
  3  WHERE yrgoal < 100 or yrgoal IS NULL;

NAME               YRGOAL CONTACT
--------------- --------- ------------
Carl Hersey               Susan Jones
Nancy Taylor           50 John Adams
Robert Brooks          50 Amy Costa
```

When working with NOT, AND, and OR, it is important to remember the order in which the processing occurs:

- NOT is resolved first
- AND is resolved next
- OR is resolved last

Parenthesis can be used to change this order because parenthesis are resolved before things that are not contained in parenthesis.

For example:

```
Condition A AND Condition B OR Condition C can be read as Condition A AND Condition B or just
Condition C and resolves as:

            Condition A AND Condition B
                        OR
            Condition C

Condition A AND (Condition B OR Condition C) can be read as Condition A AND either Condition B
OR Condition C and resolves as:

                                  Condition B
           Condition A AND           OR
                                  Condition C
```

A AND B OR C

1st

next

either

A AND (B OR C)

has to be true

An example of this in Oracle SQL is shown below. In this example I want yrgoal greater than 100 and state = 'RI' or just contact = 'Amy Costa', and that is what I will get. However, if what I wanted was that yrgoal always had to be greater than 100 and then either the state had to be 'RI' or the contact had to be 'Amy Costa' then the first example below would not work. Parenthesis would have to be included to group state = 'RI' and contact = 'Amy Costa'.

SQL CODE:

```
SQL> list
  1  SELECT name, city, state, datefst, yrgoal, contact
```

Type here to search

10:25 AM
9/3/2019

Title: Sep 3-10:25 AM (15 of 22)

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

```
Robert Brooks        50 Amy Costa
```

When working with NOT, AND, and OR, it is important to remember the order in which the processing occurs:

- NOT is resolved first
- AND is resolved next
- OR is resolved last

Parenthesis can be used to change this order because parenthesis are resolved before things that are not contained in parenthesis.

For example:

```
Condition A AND Condition B OR Condition C can be read as Condition A AND Condition B or just
Condition C and resolves as:

            Condition A AND Condition B
                        OR
            Condition C

Condition A AND (Condition B OR Condition C) can be read as Condition A AND either Condition B
OR Condition C and resolves as:

                                   Condition B
           Condition A AND              OR
                                   Condition C
```

An example of this in Oracle SQL is shown below. In this example I want yrgoal greater than 100 and state = 'RI' or just contact = 'Amy Costa', and that is what I will get. However, if what I wanted was that yrgoal always had to be greater than 100 and then either the state had to be 'RI' or the contact had to be 'Amy Costa' then the first example below would not work. Parenthesis would have to be included to group state = 'RI' and contact = 'Amy Costa'.

SQL CODE:

```
SQL> list
  1  SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE yrgoal >100 AND state = 'RI' OR contact = 'Amy Costa'
SQL> /

NAME            CITY       ST DATEFST    YRGOAL CONTACT
--------------- ---------- -- --------- ------- ------------
Jennifer Ames   Providence RI 24-MAY-97     400 Susan Jones
Susan Ash       Fall River MA 04-MAR-92     100 Amy Costa
Robert Brooks   Fall River MA 04-APR-98      50 Amy Costa
```

ANDs are resolved before ORs so the two things in the AND muct be true OR just the contact must be Amy Costa.

When you look at the results of the SQL query above, the first record meets the criteria yrgoal > 100 and state = 'RI', the second and third record meet the criteria contact = 'Amy Costa'.

In this second example the parenthesis are included so I will get records where yrgoal is always greater than 100 and either the state = 'RI' or the contact = 'Amy Costa'. In this case, only one record met the criteria. It had yrgoal > 100 and state = 'RI'.

SQL CODE:

```
SQL> list
  1  SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE yrgoal > 100 AND (state = 'RI' OR contact = 'Amy Costa')
SQL> /
```

Type here to search

10:30 AM
9/3/2019

```
OR Condition C and resolves as:

                                    Condition B
           Condition A AND              OR
                                    Condition C
```

An example of this in Oracle SQL is shown below. In this example I want yrgoal greater than 100 and state = 'RI' or just contact = 'Amy Costa', and that is what I will get. However, if what I wanted was that yrgoal always had to be greater than 100 and then either the state had to be 'RI' or the contact had to be 'Amy Costa' then the first example below would not work. Parenthesis would have to be included to group state = 'RI' and contact = 'Amy Costa'.

SQL CODE:

```
SQL> list
  1  SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE yrgoal >100 AND state = 'RI' OR contact = 'Amy Costa'
SQL> /

NAME            CITY       ST DATEFST      YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Jennifer Ames   Providence RI 24-MAY-97       400 Susan Jones
Susan Ash       Fall River MA 04-MAR-92       100 Amy Costa
Robert Brooks   Fall River MA 04-APR-98        50 Amy Costa
```

When you look at the results of the SQL query above, the first record meets the criteria yrgoal > 100 and state = 'RI', the second and third record meet the criteria contact = 'Amy Costa'.

In this second example the parenthesis are included so I will get records where yrgoal is always greater than 100 and either the state = 'RI' or the contact = 'Amy Costa'. In this case, only one record met the criteria. It had yrgoal > 100 and state = 'RI'.

SQL CODE:

has to be true

either

```
SQL> list
  1  SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE yrgoal > 100 AND (state = 'RI' OR contact = 'Amy Costa')
SQL> /

NAME            CITY       ST DATEFST      YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Jennifer Ames   Providence RI 24-MAY-97       400 Susan Jones

SQL>
```

Parenthesis are resolved first so it is determine if the state or the contact meet the criteria. This is then combined with the yrgoal check.

Note that if the criteria had been changed to yrgoal > 50 rather than 100, a second record would display (Susan Ash).

SQL CODE:

```
SQL> SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3  WHERE yrgoal > 50 AND (state = 'RI' OR contact = 'Amy Costa');

NAME            CITY       ST DATEFST      YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Jennifer Ames   Providence RI 24-MAY-97       400 Susan Jones
Susan Ash       Fall River MA 04-MAR-92       100 Amy Costa

SQL>
```

Type here to search
10:31 AM
9/3/2019

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

```
NAME            CITY       ST DATEFST      YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Jennifer Ames   Providence RI 24-MAY-97       400 Susan Jones

SQL>
```

Note that if the criteria had been changed to yrgoal > 50 rather than 100, a second record would display (Susan Ash).

SQL CODE:

```
SQL> SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3  WHERE yrgoal > 50 AND (state = 'RI' OR contact = 'Amy Costa');

NAME            CITY       ST DATEFST      YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Jennifer Ames   Providence RI 24-MAY-97       400 Susan Jones
Susan Ash       Fall River MA 04-MAR-92       100 Amy Costa

SQL>
```

Another example of a combined AND and OR is shown below. In this example I want state of Rhode Island with a date of first contribution greater than January 1, 1998 or the state of Massachusetts with a date for first contribution greater than January 1, 1996. Notice that no parenthesis are needed because the two ANDs get resolved first and then the results hit the OR test. Parenthesis could have been inserted around the two AND combinations for clarity.

1 3 2

SQL CODE:

```
SELECT name, city, state, datefst, yrgoal, contact
FROM donor
WHERE state = 'RI' AND datefst > '01-JAN-98' OR state = 'MA' AND datefst > '01-JAN-96';

NAME            CITY       ST DATEFST      YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Stephen Daniels Seekonk    MA 03-JUL-98       500 John Smith
Carl Hersey     Providence RI 03-JAN-98           Susan Jones
Robert Brooks   Fall River MA 04-APR-98        50 Amy Costa

SQL>
```

SQL also supports the NOT operator. Remember that in the hierarchy, the NOT is resolved before the AND which is resolved before the OR. In the example below, I want to print out information on all donors who do not have Amy Costa as their contact.

SQL CODE:

```
SQL> SELECT name, state, contact
  2  FROM donor
  3  WHERE NOT contact = 'Amy Costa';

NAME            ST CONTACT
--------------- -- ------------
Stephen Daniels MA John Smith
Jennifer Ames   RI Susan Jones
Carl Hersey     RI Susan Jones
Nancy Taylor    MA John Adams
```

Another example of the use of NOT is in this compound AND statement where each of the clauses contain a NOT. In this case I want all donors who do not have Amy Costa as their contact and do not live in the state of Rhode Island. The NOT has priority,

Type here to search

10:35 AM
9/3/2019

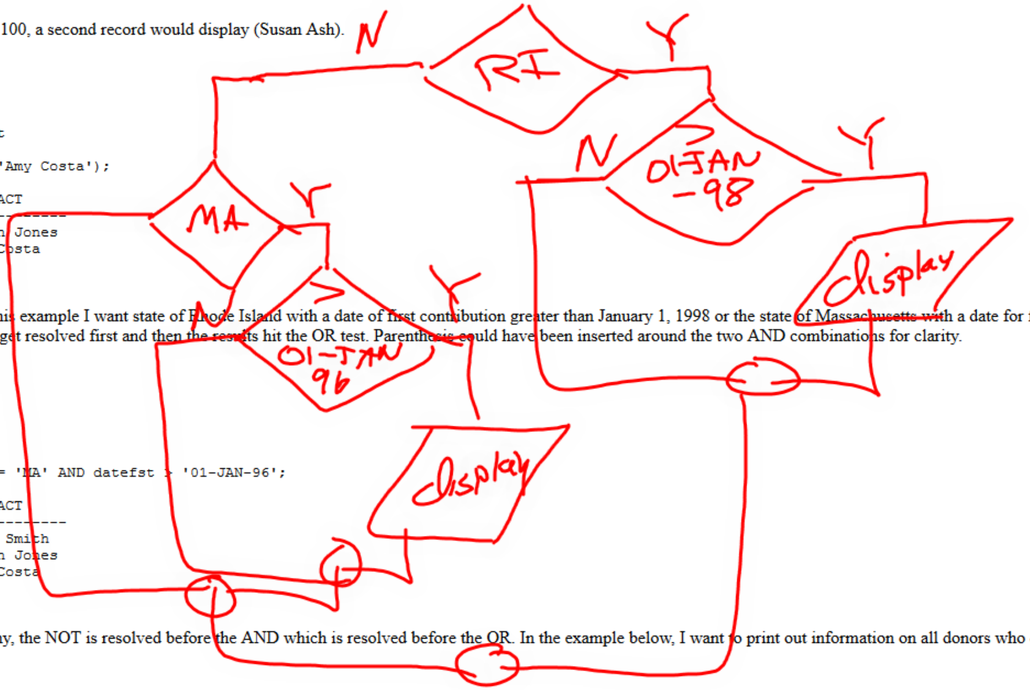
SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

```
NAME            CITY       ST DATEFST     YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Jennifer Ames   Providence RI 24-MAY-97       400 Susan Jones

SQL>
```

Note that if the criteria had been changed to yrgoal > 50 rather than 100, a second record would display (Susan Ash).



SQL CODE:

```
SQL> SELECT name, city, state, datefst, yrgoal, contact
  2  FROM donor
  3  WHERE yrgoal > 50 AND (state = 'RI' OR contact = 'Amy Costa');

NAME            CITY       ST DATEFST     YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Jennifer Ames   Providence RI 24-MAY-97       400 Susan Jones
Susan Ash       Fall River MA 04-MAR-92       100 Amy Costa

SQL>
```

Another example of a combined AND and OR is shown below. In this example I want state of Rhode Island with a date of first contribution greater than January 1, 1998 or the state of Massachusetts with a date for first contribution greater than January 1, 1996. Notice that no parenthesis are needed because the two ANDs get resolved first and then the results hit the OR test. Parenthesis could have been inserted around the two AND combinations for clarity.

SQL CODE:

```
SELECT name, city, state, datefst, yrgoal, contact
FROM donor
WHERE state = 'RI' AND datefst > '01-JAN-98' OR state = 'MA' AND datefst > '01-JAN-96';

NAME            CITY       ST DATEFST     YRGOAL CONTACT
--------------- ---------- -- --------- --------- ------------
Stephen Daniels Seekonk    MA 03-JUL-98       500 John Smith
Carl Hersey     Providence RI 03-JAN-98           Susan Jones
Robert Brooks   Fall River MA 04-APR-98        50 Amy Costa

SQL>
```

SQL also supports the NOT operator. Remember that in the hierarchy, the NOT is resolved before the AND which is resolved before the OR. In the example below, I want to print out information on all donors who do not have Amy Costa as their contact.

SQL CODE:

```
SQL> SELECT name, state, contact
  2  FROM donor
  3  WHERE NOT contact = 'Amy Costa';

NAME            ST CONTACT
--------------- -- ------------
Stephen Daniels MA John Smith
Jennifer Ames   RI Susan Jones
Carl Hersey     RI Susan Jones
Nancy Taylor    MA John Adams
```

Another example of the use of NOT is in this compound AND statement where each of the clauses contain a NOT. In this case I want all donors who do not have Amy Costa as their contact and do not live in the state of Rhode Island. The NOT has priority,

Type here to search

10:37 AM 9/3/2019

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

```
NAME            ST CONTACT
--------------- -- ------------
Stephen Daniels MA John Smith
Jennifer Ames   RI Susan Jones
Carl Hersey     RI Susan Jones
Nancy Taylor    MA John Adams
```

Another example of the use of NOT is in this compound AND statement where each of the clauses contain a NOT. In this case I want all donors who do not have Amy Costa as their contact and do not live in the state of Rhode Island. The NOT has priority, so the NOT gets resolved first and then the AND combines the two NOT conditions to say not Amy Costa and not Rhode Island. This means neither Amy Costa nor Rhode Island can appear in the row. Another way to say it is, anyone who is not Amy Costa and any state that is not Rhode Island.

SQL CODE:

```
SQL> list
  1  SELECT name, state, yrgoal, contact
  2  FROM donor
  3* WHERE NOT contact = 'Amy Costa' AND NOT state = 'RI'
SQL> /

NAME            ST    YRGOAL CONTACT
--------------- -- --------- ------------
Stephen Daniels MA       500 John Smith
Nancy Taylor    MA        50 John Adams

SQL>
```

In the next example, the AND was put inside parenthesis so it got resolved first. The NOT got applied to this resolution. The final result was all rows that did not contain Amy Costa AND Rhode Island. Since there are no records for Amy Costa and Rhode Island all the records are displayed.

SQL CODE:

```
SQL> SELECT name, state, yrgoal, contact
  2  FROM donor
  3  WHERE NOT(contact = 'Amy Costa' AND state = 'RI');

NAME            ST    YRGOAL CONTACT
--------------- -- --------- ------------
Stephen Daniels MA       500 John Smith
Jennifer Ames   RI       400 Susan Jones
Carl Hersey     RI           Susan Jones
Susan Ash       MA       100 Amy Costa
Nancy Taylor    MA        50 John Adams
Robert Brooks   MA        50 Amy Costa

6 rows selected.

SQL>
```

To help clarify this, I am now asking for all rows that did not contain Amy Costa and Massachusetts. There were two rows for Massachusetts that had Amy Costa as the contact person and those rows are not displayed.

SQL CODE:

```
SQL> SELECT name, state, yrgoal, contact
  2  FROM donor
```

Type here to search

10:38 AM
9/3/2019

Title: Sep 3-10:38 AM (20 of 22)

```
Carl Hersey     RI          Susan Jones
Nancy Taylor    MA       50 John Adams

SQL>
```

OTHER OPERATORS: BETWEEN, IN, LIKE

Oracle SQL provides several other operators that can be used to retrieve specific records. The BETWEEN and the IN simplify asking for several things. BETWEEN deals with a range and IN deals with a list.

In the example below, I want to display all rows where the datefst column contains a date between January 1, 1997 and July 3, 1998. Notice that July 2, 1998 appears on the list which tells us that the BETWEEN is inclusive meaning it includes the two dates that are the beginning and end of the range.

SQL CODE:

```
SQL> SELECT name, datefst, yrgoal
  2  FROM donor
  3  WHERE datefst BETWEEN '01-JAN-97' AND '03-JUL-98';

NAME            DATEFST      YRGOAL
--------------- --------- ---------
Stephen Daniels 03-JUL-98       500
Jennifer Ames   24-MAY-97       400
Carl Hersey     03-JAN-98
Robert Brooks   04-APR-98        50

SQL>
```

The IN clause sets up a list and checks to see if the field is included in the list. In the first example, I am looking to see if yrgoal is in the list 50, 100, 500. Only rows that where the yrgoal column has one of those values will be displayed.

SQL CODE:

```
SQL> SELECT name, datefst, yrgoal
  2  FROM donor
  3  WHERE yrgoal IN (50, 100, 500);

NAME            DATEFST      YRGOAL
--------------- --------- ---------
Stephen Daniels 03-JUL-98       500
Susan Ash       04-MAR-92       100
Nancy Taylor    04-MAR-92        50
Robert Brooks   04-APR-98        50

SQL>
```

In this example, I am checking to see if the contact is IN the list which includes John Adams and Amy Costa. Those rows where the contact column contains those names are displayed.

SQL CODE:

```
SQL> list
  1  SELECT name, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE contact IN ('John Adams', 'Amy Costa')
SQL> /
```

Type here to search
10:41 AM
9/3/2019

SQL Conditions

www.pgrocer.net/Cis50/oracle2.html

In this example, I am checking to see if the contact is IN the list which includes John Adams and Amy Costa. Those rows where the contact column contains those names are displayed.

SQL CODE:

```
SQL> list
  1  SELECT name, datefst, yrgoal, contact
  2  FROM donor
  3* WHERE contact IN ('John Adams', 'Amy Costa')
SQL> /

NAME            DATEFST      YRGOAL CONTACT
--------------- --------- --------- ------------
Susan Ash       04-MAR-92       100 Amy Costa
Nancy Taylor    04-MAR-92        50 John Adams
Robert Brooks   04-APR-98        50 Amy Costa

SQL>
```

The LIKE operator lets the user look for something when they aren't sure of the exact spelling or format. For example, I can look for all fields that start with S or all fields that contain a J in the fourth character. The % symbol is used to indicate any number of characters and the _ symbol is used to indicate one character. If either of these characters appear in the column that is being checked the \ can be used as the ESCAPE identifier to indicate that the exact character % or _ is to be looked for and that they temporarily do not have the meaning usually associated with them.

For example: WHERE name LIKE '%S_DE%' ESCAPE '\' means that you are looking in the name field for the string S_DE and there can be letters before the S and after the DE.

In the example shown below, I am looking in the contact column for all contacts with the first name of John. The % after the John means that John can be followed by any number of other letters.

SQL CODE:

```
SQL> list
  1  SELECT name, stadr, city, contact
  2  FROM donor
  3* WHERE contact LIKE 'John%';
NAME            STADR           CITY       CONTACT
--------------- --------------- ---------- ------------
Stephen Daniels 123 Elm St      Seekonk    John Smith
Nancy Taylor    26 Oak St       Fall River John Adams

SQL>
```

In this example, I am looking for a lower case a in the second character followed by any number of other characters. The _ at the beginning indicates one character then the a in the second character and the % indicates any number of characters following.

SQL CODE:

```
SQL> SELECT name, stadr, city
  2  FROM donor
  3  WHERE name LIKE '_a%';

NAME            STADR           CITY
--------------- --------------- ----------
Carl Hersey     24 Benefit St   Providence
Nancy Taylor    26 Oak St       Fall River

SQL>
```

Type here to search

10:43 AM 9/3/2019

Title: Sep 3-10:43 AM (22 of 22)