Oracle SQL Help

Connecting to Oracle via SQL*Plus

From an Athena machine:

`setup 11.521`	Run script from Athena prompt in any xterm in order to open new xterm window and set path and environment so they can find Oracle applications
`sqlplus xxxxx`	Start the SQL*PLUS client and connect to the Oracle server where xxxxx is your Oracle userid
*Then, within SQL*Plus...*	
`Enter password:` ***yyyyy***	When prompted, enter your oracle password, *yyyyy*
`quit`	When you are finished running SQL queries, exit SQL*Plus via 'quit'

From a PC (with Oracle 9i Client or equivalent running):

Click on *Start/Programs/Oracle/SQL*Plus* to start the SQL*Plus program and enter your Oracle userid and password when prompted as indicated above. Exit SQL*Plus via the command: 'quit'

Syntax of Principal SQL Statements

The basic SELECT statement to query one or more tables:

```
    SELECT [DISTINCT] column_name1[, column_name2, ...]
      FROM table_name1[, table_name2, ...]
     WHERE search_condition1
      [AND search_condition2 ...]
      [OR search_condition3...]
[GROUP BY column_names]
[ORDER BY column_names];
```

Notes about SELECT (see examples at the bottom):

- You can specify an alternative column heading to be used on tabular output as in the following SELECT statement:

```
SELECT medianhinc income, whitepop+blackpop people
  FROM census
```

- Comparisons for search condition:

=	equality
<>	inequality
!=	inequality
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
IN (*a, b, c, ...*)	one of a, b, c, etc.
BETWEEN *a* AND *b*	greater than or equal to *a* and less than or equal to *b*

Creating a temporary table:

```
CREATE TABLE tempname AS
    SELECT [..same as above..]
```

Creating a table from scratch:

```
CREATE TABLE table_name
    (columname1 datatype1[, columname2 datatype2, ...]);
```

The primary data types are: **number**, **date**, and **varchar2(*n*)** where '*n*' is an integer

Nested queries:

```
SELECT [..same as above..]
  FROM table_name
 WHERE column_name IN
   (SELECT ...);
```

Multiple table joins (simple example):

```
SELECT e.name, d.floor
 FROM employee e, department d
WHERE e.dname = d.dname and e.name = 'joe';
```

Update operations (change values in existing rows):

```
UPDATE table_name
SET column_name1 = expression1 [, ...]
[WHERE search_condition];
```

Views (like a table but the SQL command, and not the data, is saved):

```
CREATE VIEW view_name [(column_name1, column_name2, ...)]
    AS SELECT column_names
         FROM table_name
        WHERE search_condition
```

Selecting a portion of a character string:

```
SELECT SUBSTR(lastname,1,10) lastname10
  FROM table_name;
```

A Few Examples from the Sample PARCELS Database

Note that comments that will be ignored by SQL*Plus can be introduced at any point on a line by using two dashes (--). Everything after the two dashes on the line will be ignored.

```
-- 1. The simplest query on the TAX table
SELECT * FROM tax;

-- 2. A query with an expression and a column alias
SELECT parcelid, (landval + bldval) tot_val, tax
  FROM tax;

-- 3. A simple query that groups over a whole table
SELECT COUNT(*), MIN(tax), MAX(tax), AVG(tax)
  FROM tax;

-- 4. This example uses a literal string in the SELECT list
SELECT parcelid, 'Total prop value is', (landval + bldval), tax
  FROM tax;

-- 5. The simplest query from the PARCELS table
SELECT * FROM parcels;

-- The COLUMN command is not a SQL command but rather a SQL*Plus
-- command that tells the program how to format column output.
-- This is very useful for producing readable output.
-- The first example formats a numeric column to display 3 digits
('990').
-- The second formats a character column to display 8 characters
('A8').
-- See the SQL*Plus Reference information on the COLUMN command
-- for details on specifying the formats.
COLUMN parcelid FORMAT 990
COLUMN add2 FORMAT A8
```

```
-- 6. The simplest query on the PARCELS table
SELECT * FROM parcels;

-- 7. A simple query using the "IS NULL" syntax
SELECT *
  FROM parcels
 WHERE sqft >= 20000 OR sqft IS NULL;

-- 8. Compare the results of the query above with the one below.
-- Why are they different?
SELECT *
  FROM parcels
 WHERE sqft >= 20000 OR sqft = NULL;

-- 9. A query to find values within a certain range
SELECT *
  FROM parcels
 WHERE sqft >= 10000 AND sqft <= 100000;

-- 10. Another way of writing the query above using the
-- "BETWEEN" keyword
SELECT *
  FROM parcels
 WHERE sqft BETWEEN 10000 AND 100000;

-- 11. A simple join between PARCELS and FIRES
SELECT p.*, f.estloss
  FROM parcels p, fires f
 WHERE p.parcelid = f.parcelid;

-- 12. A slight variation
  SELECT p.parcelid, p.sqft, p.landuse, f.estloss
    FROM parcels p, fires f
 WHERE p.parcelid = f.parcelid
ORDER BY f.estloss;

-- 13. A simple GROUP BY query
  SELECT zip, COUNT(DISTINCT pid) parcels, AVG(sqft) avg_sqft
    FROM parcels
GROUP BY zip
ORDER BY zip;

-- 14. Show all the fires
  SELECT f.*
    FROM fires f
ORDER BY f.parcelid, f.fdate;

-- 15. Show all the fires that have a matching record
-- in the ZONING table.
  SELECT f.*
    FROM fires f, zoning z
   WHERE f.parcelid = z.parcelid
ORDER BY f.parcelid, f.fdate;

-- 16. This example uses the "NOT IN" syntax
```

```
-- with a subquery to find the fires that
-- do NOT have a matching record in the
-- ZONING table (i.e., the records from query
-- 14 that did NOT appear in query 15).
SELECT f.*
  FROM fires f
 WHERE f.parcelid NOT IN
   (SELECT z.parcelid
      FROM zoning z);


-- 17. A three-table join to show the owners of parcels
-- with fires
SELECT p.parcelid, landuse, oname, estloss
  FROM parcels p, owners o, fires f
 WHERE p.parcelid = f.parcelid and
       p.onum = o.ownernum
 ORDER BY oname;

-- 18. Find the properties owned by "GERALD RAPPAPORT"
-- that had fires and permits.
SELECT distinct o.oname, p.add1, p.add2, p.parcelid
  FROM owners o, parcels p, fires f, permits e
 WHERE o.oname = 'GERALD RAPPAPORT'
   AND o.ownernum = p.onum
   AND p.parcelid = f.parcelid
   AND p.pid = e.pid and p.wpb = e.wpb;
```

Using the SQL*Plus Interface to Oracle's SQL

Here are some useful tips and tricks to customize your SQL*Plus environment and to save and retrieve SQL queries and output tables from UNIX files:

`SELECT * FROM cat;`

This SQL query lists most of the tables (and synonyms and views) from which you have permission to make selections including all those that you have created (but not some that others have created and given you permission to use).

SQL*Plus stores your most recent SQL statement in a buffer. You place a command in the buffer either by typing (or cutting-and- pasting) one in at the '**SQL>**' prompt or loading one in from a file. The following commands make use of the SQL buffer:

`GET` ***`file`***

Loads *file* into the SQL buffer. By default, SQL*Plus assumes that the file name ends with '.sql'. For example, the commands

`GET query1.sql`

and

`GET query1`

load the same file. The file should contain only one SQL statement.

`SAVE` ***`file`***
`SAV` ***`file`***

Saves the statement in the SQL buffer to *file*. The file extension '.sql' is assumed. Hence, the commands **`SAVE query2.sql`** and **`SAVE query2`** will write to the same file.

LIST
L

Lists the contents of the SQL buffer

RUN
R

Lists and runs the current command in the SQL buffer
/ *(forward slash)*
Runs the current command in the SQL buffer without listing it first

START ***file***
STA ***file***
@*file*

Loads and runs the commands in *file*. The extension '.sql' is assumed. For example, the command **START command1.sql** and **START command1** execute the same file. Unlike **GET**, files run with **START** may contain many SQL statements or SQL*Plus commands.

SPOOL ***file***
SPO ***file***

Begins saving all output from the SQL*Plus session (including what you type) to file. The file extension '.lst' is assumed. Hence, the commands **SPOOL log.lst** and **SPOOL log** would both create a file called 'log.lst'.

SPOOL OFF
SPO OFF

Stops writing SQL*Plus output to a spool file opened using the 'SPOOL file' command.

SET ECHO ON

After this command is issued, the **START** command will display each statement in the command file just before it is executed.

SET ECHO OFF

After this command is issued, the **START** command will not display statements in the command file as they are executed.

SET LINESIZE *n*

Sets the number of characters printed on a line to *n*. For example,
SET LINESIZE 150
will instruct SQL*Plus to print 150 characters on a line. This command is useful once you have increased the width of an xterm window beyond the standard 80 characters to see more columns. The default setting is **'SET LINESIZE 80'**.

SET PAGESIZE *n*

Sets the number of lines that SQL*Plus displays before repeating the column headings to n. The default setting is '**SET PAGESIZE 14**', which is generally too small. '**SET PAGESIZE 24**' is recommended.

```
SET PAUSE '--Hit return to continue--'
SET PAUSE ON
```

Use these two commands to cause SQL*Plus to pause before each page of output. Note that a pause will also occur before the FIRST page, so that you will need to press the return key to see any output.

HELP
HELP *topic*

Accesses the on-line help system. 'HELP' alone displays an introductory message; 'HELP topic' provides information about a specific topic. For example, HELP SELECT provides information about the SELECT statement.

COLUMN *column_name* FORMAT *format_type*

Sets the display format of *columnname* to *formattype*. Using this command can help improve the appearance of your queries. Some examples:

```
COLUMN FTYPE FORMAT A5
COLUMN HHINDEX FORMAT 999990
COLUMN TENURE FORMAT 9990.0
```

Note that each SQL statement you enter should be terminated with a semicolon (;). If you forget it, you can enter it on a line by itself or use the forward slash (/) to run your statement. SQL statements (e.g., SELECT, INSERT, UPDATE, DELETE) may be spread

over any number of lines; just press return when you want to start a new line. Using multiple lines for each clause in the statement significantly improves their readability, so this technique is highly recommended. SQL*Plus commands that are NOT SQL statements (e.g., SET, SPOOL, HELP, COLUMN) must appear on one line and do NOT require semicolons at the end.

SQL statements and SQL*Plus commands are NOT case sensitive. Case DOES matter, however, for any text enclosed in either single (') or double (") quotation marks.

Creating Unique Temporary Tables

Doing the homework exercises requires setting up temporary tables. At least initially, the class setup for Oracle has everyone using the same Oracle userid for some of the class datasets. Hence, several people could end up trying to create a table with the same name at the same time. To avoid this problem, you should pre-append a unique id to the name of every temporary table that you create. Suppose, for example, the 7th person in the class list wanted to save all parcels with more than 10,000 square feet into a temporary table called BIGPARCELS. Then this person would append 't7' to the front of any temporary table name. For example:

```
CREATE TABLE t7bigparcels as
  SELECT *
    FROM parcels
    WHERE sqft > 10000;
```

When done with the table, please 'drop' it via:

```
DROP TABLE t7bigparcels;
```

so that the database is not cluttered with lots of old temporary tables.

Oracle References

Complete Oracle 8i documentation is available online at

http://technet.oracle.com/docs/products/oracle8i/doc_index.htm.

The documentation is generally available both in HTML format (good for viewing online in a browser) and PDF format (better for printing).

Particularly useful references:

- *Oracle 8i SQL Reference*. This is the authoritative reference for Oracle's flavor of SQL.
- *SQL*Plus Quick Reference*. A concise guide to SQL*Plus commands.
- *SQL*Plus Reference*. The full SQL*Plus manual.

- *Oracle 8*i *Designing and Tuning for Performance*. Suggests ways to tune queries and database designs to improve performance. Offers insights into how Oracle operates.
- *Oracle 8*i *Error Messages.* Provides extra information about Oracle's sometimes cryptic error messages.

*Note: To access this documentation you will need to register for a **free** membership to the Oracle Technology Network. To register, click the 'My Profile' button in the upper right portion of the page and then complete and submit the registration form.*

A recommended Oracle reference book:

Kevin Loney and George Koch
Oracle8i: The Complete Reference
Berkeley: Osborne McGraw-Hill, 2000