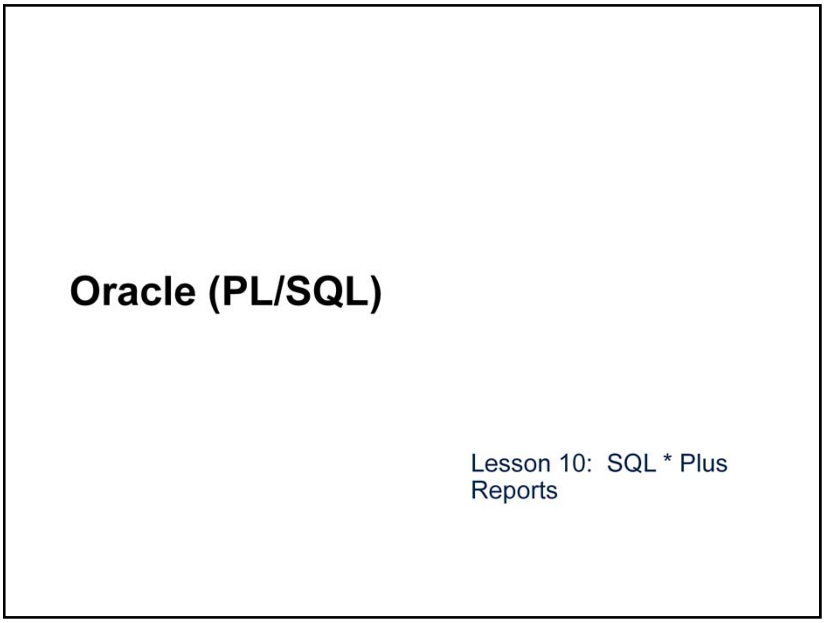
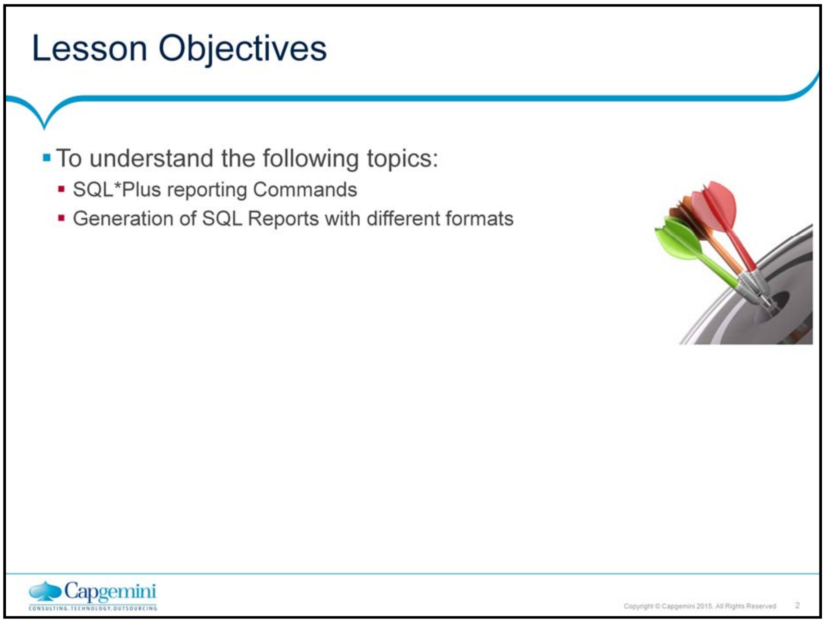


# Oracle (PL/SQL)

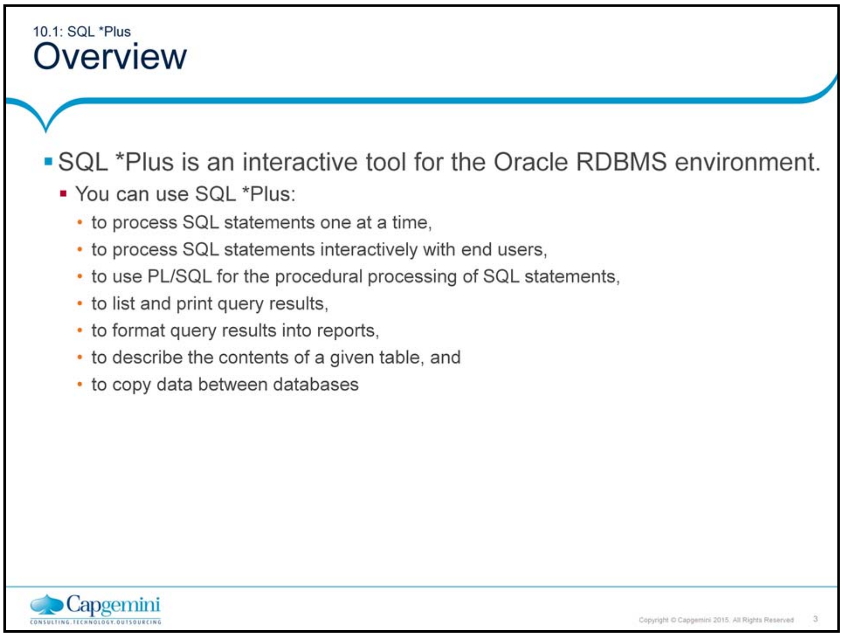
Lesson 10: SQL * Plus Reports



## Lesson Objectives

- To understand the following topics:
  - SQL*Plus reporting Commands
  - Generation of SQL Reports with different formats

Copyright © Capgemini 2015. All Rights Reserved 2



10.1: SQL *Plus

## Overview

- SQL *Plus is an interactive tool for the Oracle RDBMS environment.
  - You can use SQL *Plus:
    - to process SQL statements one at a time,
    - to process SQL statements interactively with end users,
    - to use PL/SQL for the procedural processing of SQL statements,
    - to list and print query results,
    - to format query results into reports,
    - to describe the contents of a given table, and
    - to copy data between databases

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

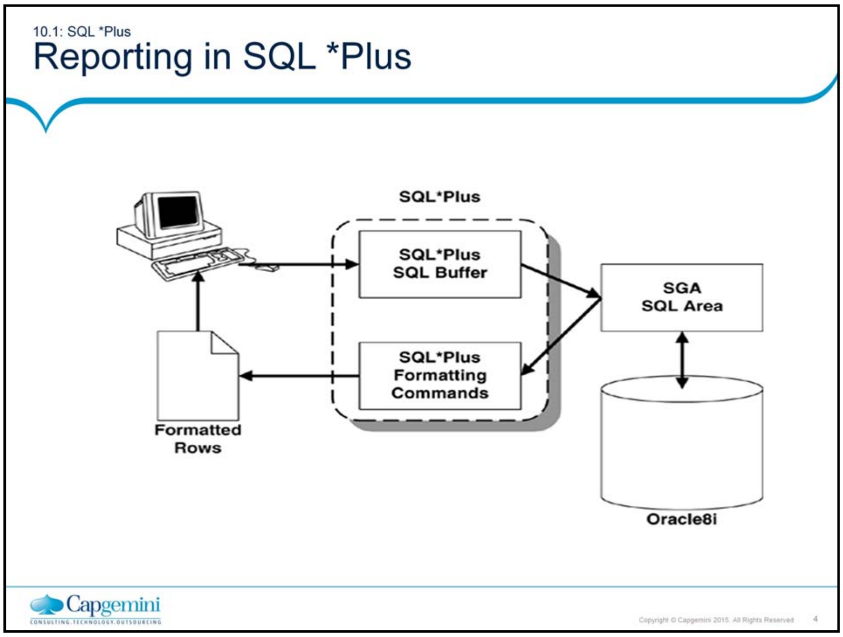
Copyright © Capgemini 2015. All Rights Reserved 3

**SQL *Plus**:

- This chapter concentrates on using SQL *Plus to format output into a variety of reports.
- It introduces methods of using SQL*Plus to create dynamic, data-driven SQL *Plus programs, and operating-system specific command language programs.

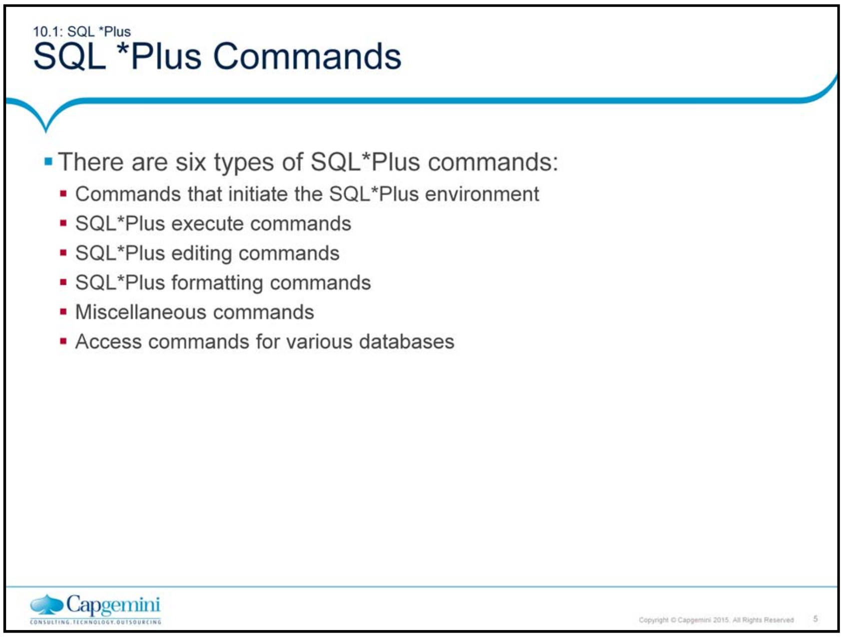
**Brief History of SQL *Plus**

- SQL *Plus originated from the beginning of the Oracle RDBMS days as a product called User Friendly Interface (UFI).
- Before version 4 of Oracle RDBMS, UFI was used primarily to administer the Oracle environment.
- UFI later was renamed to SQL*Plus with the UFI product.
- There have been additions to several of the command capabilities, additional ways of starting SQL *Plus, and a changed role for SQL *Plus through the major releases of the Oracle RDBMS kernel.


10.1: SQL *Plus

## Reporting in SQL *Plus

Copyright © Capgemini 2015. All Rights Reserved 4

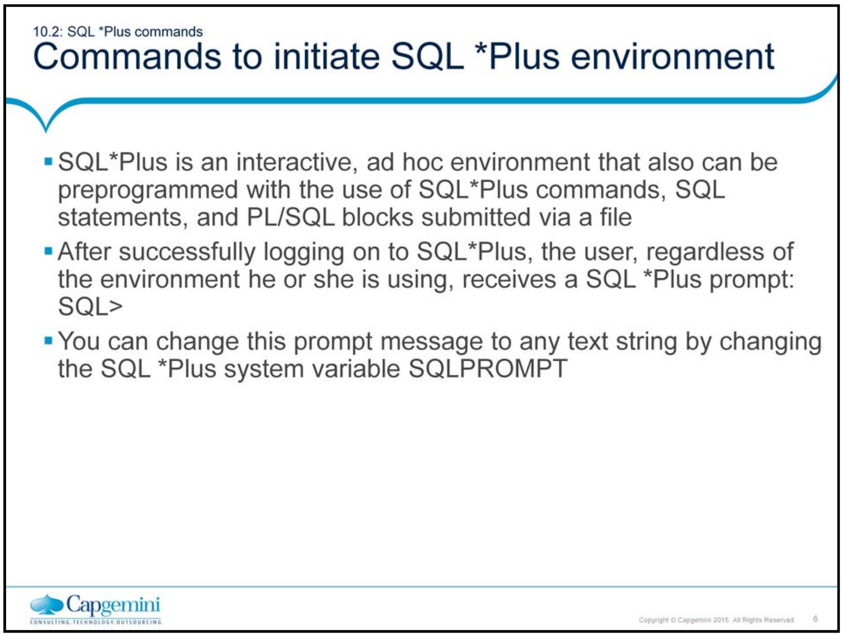


10.1: SQL *Plus

## SQL *Plus Commands

- There are six types of SQL*Plus commands:
  - Commands that initiate the SQL*Plus environment
  - SQL*Plus execute commands
  - SQL*Plus editing commands
  - SQL*Plus formatting commands
  - Miscellaneous commands
  - Access commands for various databases

Copyright © Capgemini 2015. All Rights Reserved 5



10.2: SQL *Plus commands

## Commands to initiate SQL *Plus environment

- SQL*Plus is an interactive, ad hoc environment that also can be preprogrammed with the use of SQL*Plus commands, SQL statements, and PL/SQL blocks submitted via a file
- After successfully logging on to SQL*Plus, the user, regardless of the environment he or she is using, receives a SQL *Plus prompt: SQL>
- You can change this prompt message to any text string by changing the SQL *Plus system variable SQLPROMPT

Copyright © Capgemini 2015. All Rights Reserved 6

**<u>Commands that initiate the SQL *Plus environment</u>:**

- You can enhance the basic SQL *Plus environment for each user or group of users by using a file named LOGIN.SQL. This file should be located in the directory or home environment from which SQL*Plus is initiated.
- Oracle and SQL *Plus is run in a variety of computer environments. The method used to create files and the definition of the home environment vary greatly among types of computer operating systems. Typical contents of this file are various SET commands that alter the SQL *Plus default settings for the particular user.
- The PRODUCT_USER_PROFILE table, owned by SYSTEM, is one way to provide product-level security that enhances the security provided by the SQL GRANT and REVOKE commands. This level of security is used to disable certain SQL and SQL *Plus commands for individual users.
- There are various ways to initiate the SQL *Plus environment, depending on the type of computer platform being used.
- To leave the SQL *Plus environment, simply type EXIT at the SQL> prompt and press RETURN or ENTER key.
- To terminate a SQL *Plus command file, make EXIT the last line of the file.

**Character-Mode Environments**

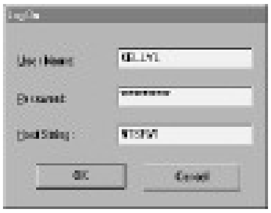
- SQL *Plus is a character-based tool that runs in both character-mode and graphical environments. The way in which the SQL *Plus environment is initiated varies greatly between the two types of environments. The following syntax initiates SQL *Plus and prompts the user for a valid username and password at the command prompt:
    - C:\Sqlplus
- After SQL *Plus is started, key in a valid password and press ENTER. SQL*Plus then prompts you for a password. The password is not displayed on-screen.
- After starting SQL*Plus in character mode, the output should look similar to the following:

```
SQL*Plus: Release 9.2.0.1.0 - Production on Tue Jul 8
    21:25:8 2008
(c) Copyright 19982, 2002 Oracle Corporation.  All rights
    reserved.
Enter username:
```

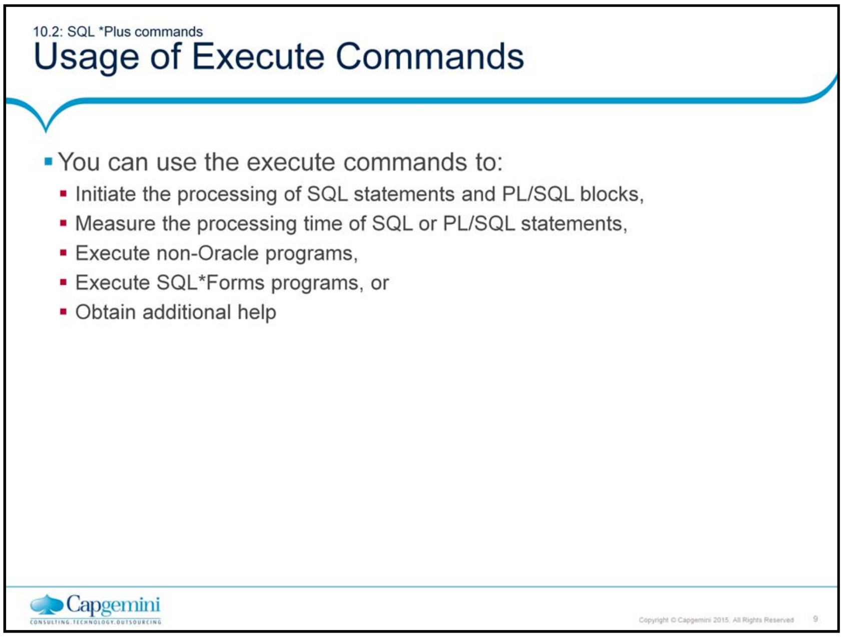
- The following syntax initiates SQL*Plus but does not prompt for the user ID or password:
    - C:\Sqlplus \nolog
- Using the option _S or _SILENT (_S spelled out) does not display the SQL*Plus version and copyright information:
    - C:\Sqlplus  -S userid/password
    - This code is handy when you are initiating reports written in SQL*Plus from a menu system and you want the appearance of a seamless application.
- The following syntax initiates the SQL*Plus environment and connects the user to the remote database identified by the database name:
    - C:\Sqlplus  userid/password@database
- This syntax initiates the SQL*Plus environment and executes the SQL*Plus commands and the SQL (or PL/SQL blocks) contained within the file (SQL*Plus command file):
    - C:\sqlplus userid/password @filename
- Always use an operating system-dependent, fully qualified filename when specifying a filename to execute.
- This syntax initiates the SQL*Plus environment and expects the first line of the file to contain a valid user ID and password, in this format:
    - C:\sqlplus @filename
- If the user ID and password are valid, SQL*Plus processes the SQL*Plus commands and the SQL (or PL/SQL blocks) contained in the file.
    - C:\sqlplus userid/password @filename param1 param2 ...
- The above command-line parameters are passed to variables inside the SQL*Plus command file and are identified inside this file by &1, &2, and so on.

**Graphical-Mode Environments:**

- This section discusses the syntax required to initiate the SQL*Plus environment from the Windows graphical environments. When started in graphical mode, SQL*Plus displays a Log On window that requests a valid username, a password, and a connect string, as shown in Figure.



User Name: [illegible]
Password: *********
Host String: [illegible]

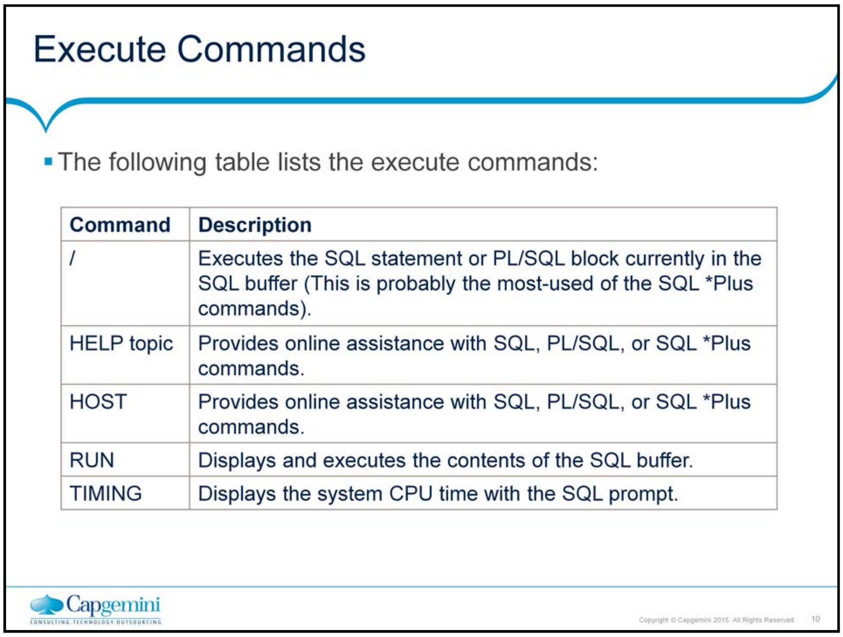


10.2: SQL *Plus commands

## Usage of Execute Commands

- You can use the execute commands to:
  - Initiate the processing of SQL statements and PL/SQL blocks,
  - Measure the processing time of SQL or PL/SQL statements,
  - Execute non-Oracle programs,
  - Execute SQL*Forms programs, or
  - Obtain additional help

Copyright © Capgemini 2015. All Rights Reserved 9

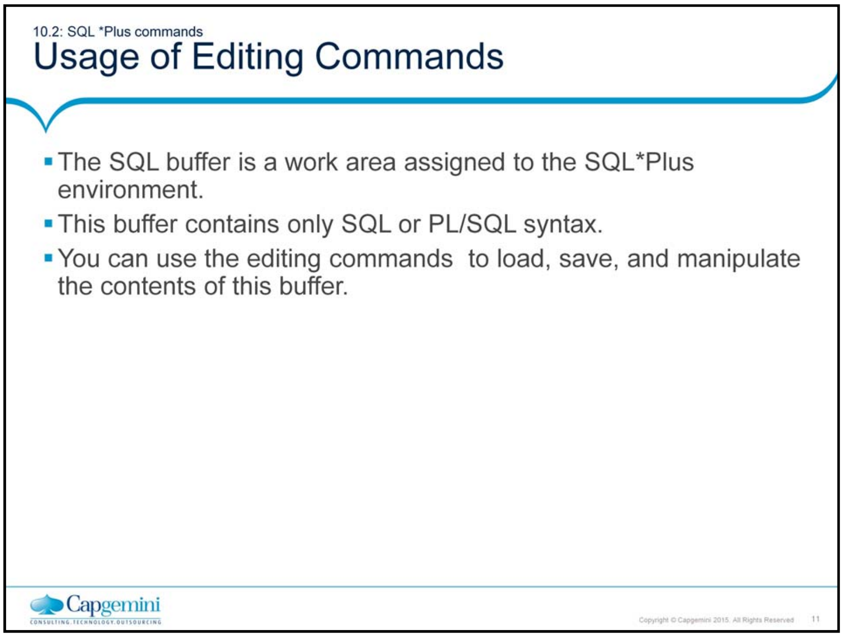


## Execute Commands

- The following table lists the execute commands:

| Command | Description |
|---|---|
| / | Executes the SQL statement or PL/SQL block currently in the SQL buffer (This is probably the most-used of the SQL *Plus commands). |
| HELP topic | Provides online assistance with SQL, PL/SQL, or SQL *Plus commands. |
| HOST | Provides online assistance with SQL, PL/SQL, or SQL *Plus commands. |
| RUN | Displays and executes the contents of the SQL buffer. |
| TIMING | Displays the system CPU time with the SQL prompt. |

Copyright © Capgemini 2015. All Rights Reserved 10



10.2: SQL *Plus commands

## Usage of Editing Commands

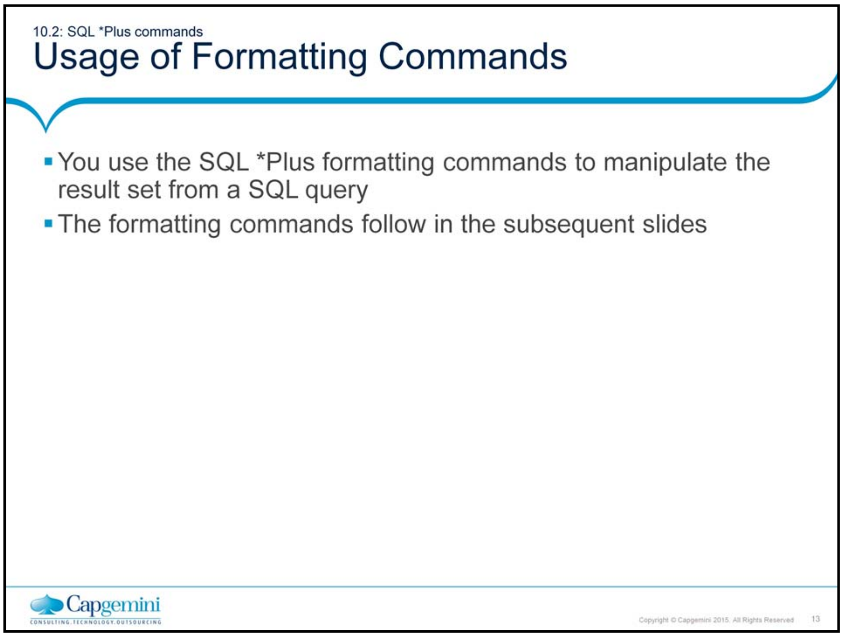
- The SQL buffer is a work area assigned to the SQL*Plus environment.
- This buffer contains only SQL or PL/SQL syntax.
- You can use the editing commands  to load, save, and manipulate the contents of this buffer.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 11

## SQL*Plus Editing Commands (contd.):

| Command | Function |
|---|---|
| A new text or APPEND new text | Appends text to the end of the current line of the SQL buffer |
| C/target text/new text/ or | Changes the target text to the CHANGE/target text/new text/ new text on the current line in the SQL buffer. |
| CLEAR BUFFER or CL BUFF | Deletes all lines in the SQL buffer. |
| DEL | Deletes the current line in the SQL buffer. |
| DEL y | Deletes line y from the SQL buffer. |
| DEL y z | Deletes from line y to line z in the SQL buffer. |
| DEL * | Deletes the current line from the SQL buffer. |
| DEL LAST | Deletes the last line in the SQL buffer. |
| EDIT filename | Uses an operating-system_dependent text editor. To edit the SQL buffer with an operating system_ dependent text editor, simply leave off the filename. |
| GET filename | Reads an operating-system_dependent file into the SQL buffer. |
| I text or INPUT text | Adds the text after the current line in the SQL buffer. |
| L number or LIST number | Displays the contents of the SQL buffer. When the number syntax is used, LIST will display the line number and make that line the current line in the SQL buffer. |
| LIST y z | Displays lines y to z from the SQL buffer. |
| LIST LAST | Lists the last line in the SQL buffer. |
| SAVE filename | Saves the contents of the SQL buffer to an operating-system_dependent file. |
| START filename param1 param2 ... | START executes the contents of the SQL*Plus command file named in filename and passes any input parameters to the SQL*Plus command file. |

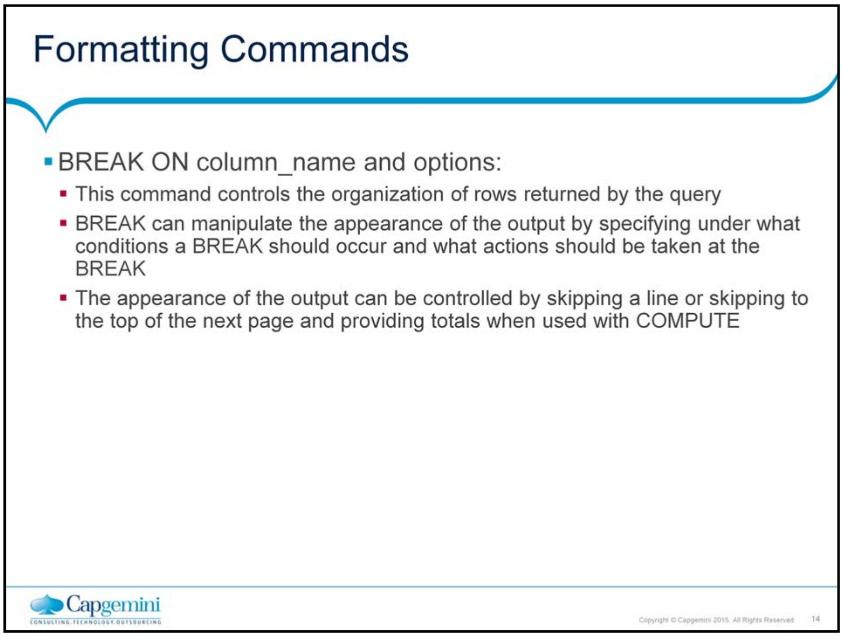


10.2: SQL *Plus commands

## Usage of Formatting Commands

- You use the SQL *Plus formatting commands to manipulate the result set from a SQL query
- The formatting commands follow in the subsequent slides

Copyright © Capgemini 2015. All Rights Reserved 13



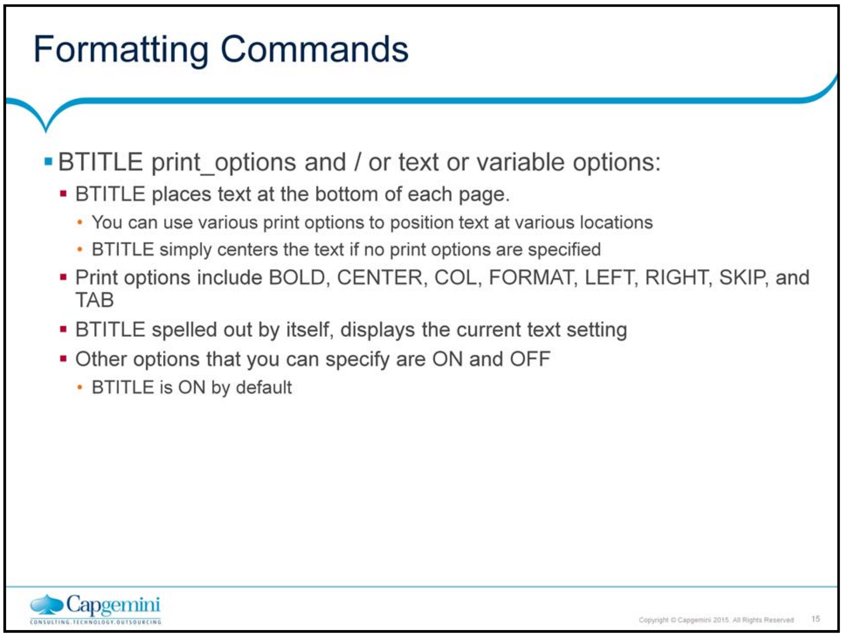
## Formatting Commands

- BREAK ON column_name and options:
  - This command controls the organization of rows returned by the query
  - BREAK can manipulate the appearance of the output by specifying under what conditions a BREAK should occur and what actions should be taken at the BREAK
  - The appearance of the output can be controlled by skipping a line or skipping to the top of the next page and providing totals when used with COMPUTE

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 14

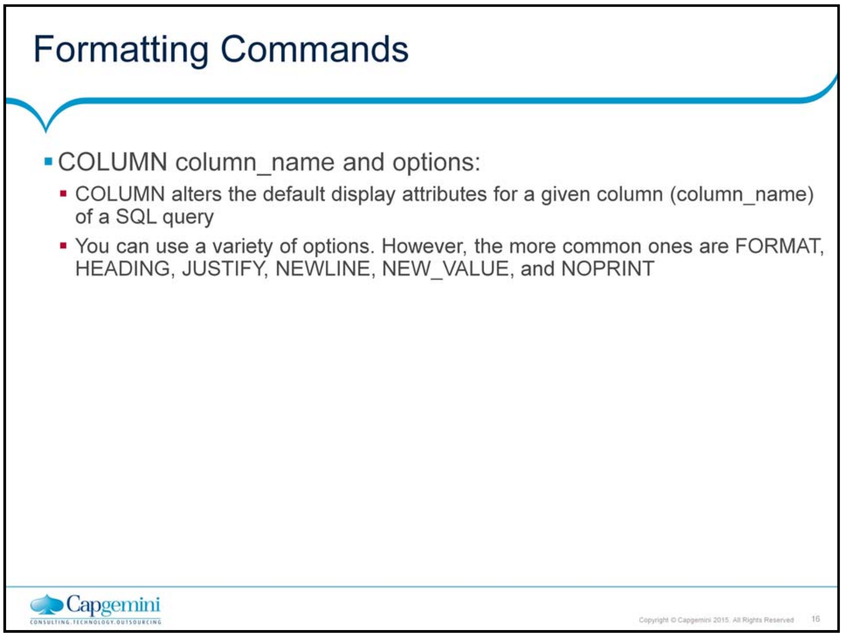
**Note:**
- Any number of lines can be skipped at a BREAK point.
- BREAK points can be defined at the column level, for multiple columns, on a row, on a page, or on a report.
- Refer the COMPUTE command for BREAK examples.
- Keying the BREAK by itself at the SQL prompt displays the current BREAK settings.



## Formatting Commands

- BTITLE print_options and / or text or variable options:
  - BTITLE places text at the bottom of each page.
    - You can use various print options to position text at various locations
    - BTITLE simply centers the text if no print options are specified
  - Print options include BOLD, CENTER, COL, FORMAT, LEFT, RIGHT, SKIP, and TAB
  - BTITLE spelled out by itself, displays the current text setting
  - Other options that you can specify are ON and OFF
    - BTITLE is ON by default

Copyright © Capgemini 2015. All Rights Reserved 15



## Formatting Commands

- COLUMN column_name and options:
  - COLUMN alters the default display attributes for a given column (column_name) of a SQL query
  - You can use a variety of options. However, the more common ones are FORMAT, HEADING, JUSTIFY, NEWLINE, NEW_VALUE, and NOPRINT

Copyright © Capgemini 2015. All Rights Reserved 16

**<u>Formatting Commands</u>:**

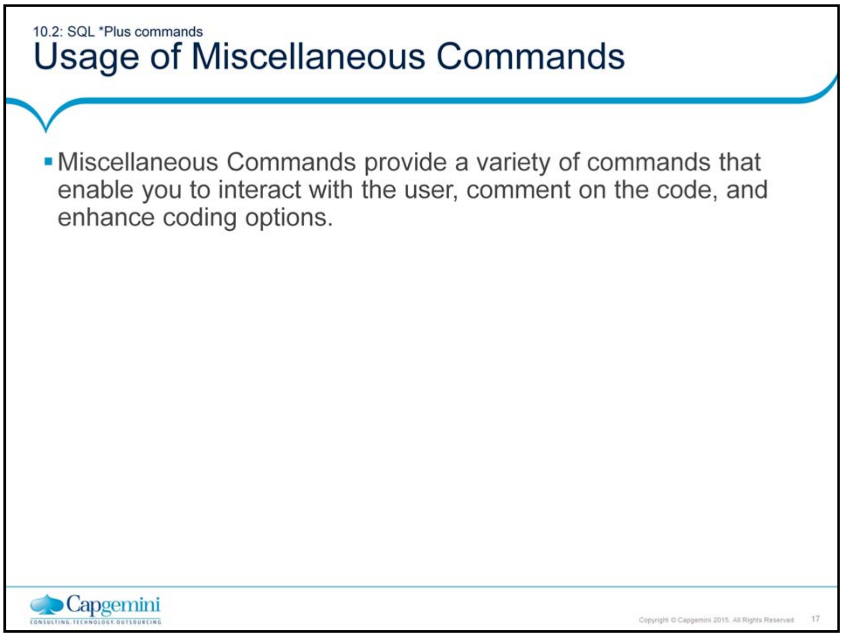
- **FORMAT:** It is useful for applying editing to numeric fields, date masks to date fields, and specific lengths to variable-length character fields.

- **HEADING:** It overrides the SQL*Plus default heading for the particular column.

- **JUSTIFY:** It overrides the SQL*Plus column alignment to the heading default.

- **NEWLINE:** It prints the column on the beginning of the next line.

- **NEW_VALUE:** It assigns the contents of the column to a SQL*Plus variable (see DEFINE, later in this section).
  - ➢ You can then use this value with TTITLE or to store intermediate results for master / detail-type reports.
  - ➢ It is useful to store and pass information between two or more separate SQL statements.

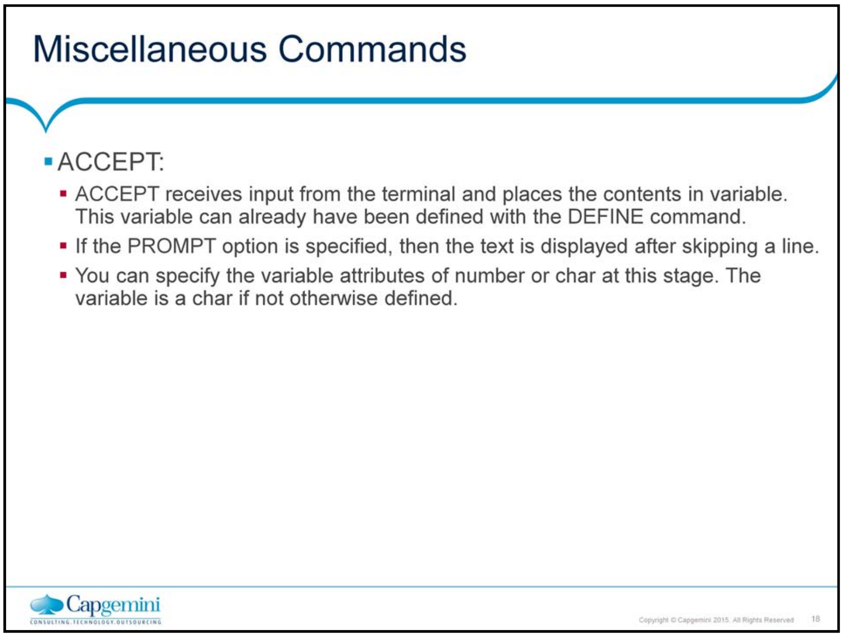


10.2: SQL *Plus commands

## Usage of Miscellaneous Commands

- Miscellaneous Commands provide a variety of commands that enable you to interact with the user, comment on the code, and enhance coding options.

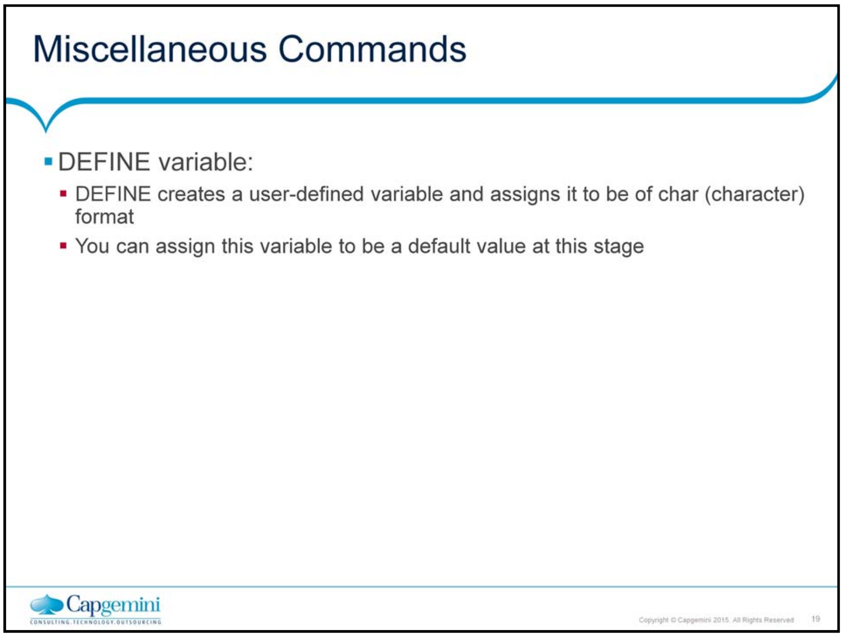
Copyright © Capgemini 2015. All Rights Reserved 17



## Miscellaneous Commands

- ACCEPT:
  - ACCEPT receives input from the terminal and places the contents in variable. This variable can already have been defined with the DEFINE command.
  - If the PROMPT option is specified, then the text is displayed after skipping a line.
  - You can specify the variable attributes of number or char at this stage. The variable is a char if not otherwise defined.

Copyright © Capgemini 2015. All Rights Reserved 18



## Miscellaneous Commands

- DEFINE variable:
  - DEFINE creates a user-defined variable and assigns it to be of char (character) format
  - You can assign this variable to be a default value at this stage

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 19

**Note:**

- DEFINE statements are handy for assigning a variable name to the input parameters coming into the SQL*Plus command file.
- **For example:**
  DEFINE SYSTEM_NAME = &1
- This line creates a character variable SYSTEM_NAME and assigns it the text associated with the first input parameter.
- The DEFINE statement makes the SQL*Plus command file's code easier to follow.

**Miscellaneous Commands (contd.):**

- **DESC or DESCRIBE database object:**
  DESCRIBE displays the columns associated with a table, view, or synonym.
- **PAUSE text:**
  PAUSE prints the contents of text after skipping a line, and then waits for you to press the RETURN (or ENTER) key.
- **PROMPT text:**
  PROMPT simply skips a line and prints the contents of text.
- **REM or REMARK:**
  SQL*Plus ignores the contents of this line when it is used in SQL*Plus command files. REMARK enables documentation or other comments to be contained in these SQL*Plus command files.
- **SET SQL*Plus System Variable:**
  The SET command controls the default settings for the SQL*Plus environment. You can automatically alter these settings for each SQL*Plus session by including them in the LOGIN.SQL file, discussed earlier in this chapter in the "SQL*Plus Commands" section.
  Refer Chapter 6 of Oracle's SQL*Plus User's Guide and Reference for a complete listing of the SET options.
  Following are some common SET options used for reporting:
  - **SET LINESIZE 80:** Controls the width of the output report line.
  - **SET PAGESIZE 55:** Controls the number of lines per page.

  Following are some common SET options that suppress various SQL*Plus output:
  - **SET FEEDBACK OFF:** Suppresses the number of query rows returned.
  - **SET VERIFY OFF:** Suppresses the substitution text when using &variables, including command-line variables.
  - **SET TERMOUT OFF:** Suppresses all terminal output; this is particularly useful with the SPOOL command.
  - **SET ECHO OFF:** Suppresses the display of SQL*Plus commands.
- **SPOOL filename or options:**
  - SPOOL opens, closes, or prints an operating-system_dependent file. Specifying SPOOL filename creates an operating-system-dependent file. Filename can contain the full pathname of the file and extension.
  - If no file extension is given, the file suffix, LST, is appended (filename.LST). Options include OFF and OUT.
  - If OFF is specified, the operating-system_dependent file simply is closed.
  - If OUT is specified, the operating-system_dependent file is closed and sent to the operating-system_dependent printer assigned as the default printer to the user's operating-system environment.
    **NOTE**
    If you issue SPOOL filename without issuing a SPOOL OFF or SPOOL OUT, the current operating system_dependent file is closed and the new one, as specified by the SPOOL command, is opened.
- **UNDEFINE variable:**
  UNDEFINE removes the previously DEFINEd variable from the SQL*Plus environment.

**Access Commands for Various Databases:**

- The database-access commands CONNECT, DISCONNECT, and COPY are used to connect to and share data with other Oracle databases. (A discussion of these commands is beyond the scope of this chapter.)

**Reporting Techniques:**

- This section covers some common SQL*Plus report-formatting features. It also covers techniques for controlling the resulting output. You will see examples of both simple and advanced reporting techniques in this section.
- Listing 4.1 formats the results of a SQL query. It defines a report title and formats, assigns column headings, and applies some control breaks for intermediate and report totaling.
- Listing 4.1. Simple SQL*Plus report code:

```
define ASSIGNED_ANALYST = &1
set FEEDBACK OFF
set VERIFY OFF
set TERMOUT OFF
set ECHO OFF
column APPLICATION_NAME    format a12    heading `Application'
column PROGRAM_NAME        format a12    heading `Program'
column PROGRAM_SIZE        format 999999    heading `Program|Size'
break on APPLICATION_NAME skip 2
break on report skip 2
compute sum of PROGRAM_SIZE on APPLICATION_NAME
compute sum of PROGRAM_SIZE on report
ttitle `Programs by Application | Assigned to: &&ASSIGNED_ANALYST'
spool ANALYST.OUT
select APPLICATION_NAME,PROGRAM_NAME,nvl(PROGRAM_SIZE,0)
  from APPLICATION_PROGRAMS
 where ASSIGNED_NAME = `&&ASSIGNED_ANALYST'
 order by APPLICATION_NAME,PROGRAM_NAME
/
spool off
exit
```

contd.

**Reporting Techniques (contd.):**

- Following is the output report from the code in Listing 13.1:

```
Tue Jul 13                                          page    1
                  Programs by Application
                      Assigned to: BILLK
Program
Application Program              Size
----------- ------------         ---------
COBOL       CLAIMS               10156
            HOMEOWN              22124
            PREMIUMS             10345
                                 ---------
            sum                42625

FORTRAN     ALGEBRA              6892
            MATH1                7210
            SCIENCE1             10240
                                 ---------
            sum                  24342

            sum                  66967
```

- Listing 4.1 is a simple but common form of SQL*Plus formatting.
- This report passes a command-line parameter (&1 on line 1) and assigns it to the variable name ASSIGNED_ANALYST.
- The ASSIGNED_ANALYST variable is then used in the headings (line 13) and again as part of the SQL query (line 17).
- Lines 2 through 5 suspend all terminal output from the SQL*Plus environment.
- The && is used to denote substitution of an already defined variable.
- This report contains two breaks:
  - one when the column APPLICATION_NAME changes (line 9), and
  - one at the end of the report (line 10)
- Totals are calculated, as well, for each of these breaks (lines 11 and 12).
- The pipe character (|) in the TTITLE command (line 13) moves the following text onto its own line.
- Line 13 opens an operating-system_dependent file named ANALYST.OUT in the current operating-system_dependent directory. The order by clause of the query on line 18 ensures that the breaks occur in an orderly manner.

**Note:**

- Always order the query output by the breaks expected by the program. The only way to guarantee the order of the rows is to use an ORDER BY clause on the query.

**Advanced Reporting Techniques**

- Listing 4.2 creates a cross-tabular report with a spreadsheet appearance.
- Listing 4.2. Cross-tabular SQL*Plus report code.

```
define RPT_DATE = &1
set FEEDBACK OFF
set VERIFY OFF
set TERMOUT OFF
set ECHO OFF
column SALES_REP        format a12    heading `Sales|Person'
column NISSAN           format 999999 heading `Nissan'
column TOYOTA           format 999999 heading `Toyota'
column GM               format 999999 heading `GM'
column FORD             format 999999 heading `Ford'
column CHRYSLER         format 999999 heading `Chrysler'
column TOTALS           format 999999 heading `Totals'
break on report skip 2
compute sum of NISSAN on report
compute sum of TOYOTA on report
compute sum of GM on report
compute sum of FORD on report
compute sum of CHRYSLER on report
compute sum of TOTALS on report
ttitle left `&&IN_DATE' center `Auto Sales' RIGHT `Page: ` format 999 -
        SQL.PNO skip CENTER ` by Sales Person `
spool SALES.OUT
select SALES_REP,
     sum(decode(CAR_TYPE,'N',TOTAL_SALES,0)) NISSAN,
     sum(decode(CAR_TYPE,'T',TOTAL_SALES,0)) TOYOTA,
     sum(decode(CAR_TYPE,'G',TOTAL_SALES,0)) GM,
     sum(decode(CAR_TYPE,'F',TOTAL_SALES,0)) FORD,
     sum(decode(CAR_TYPE,'C',TOTAL_SALES,0)) CHRYSLER ,
     sum(TOTAL_SALES) TOTALS
from CAR_SALES
where SALES_DATE <= to_date(`&&RPT_DATE')
group by SALES_REP
/
spool off
exit
```

contd.

**Advanced Reporting Techniques (contd.):**

- The following code shows the output report from Listing 4.2:

```
31-AUG-95          Auto Sales
Page: 1
by Sales Person

Sales
Sales Person     Nissan     Toyota     GM         Ford
    Chrysler  Totals
---------------      --------   -------- --------      --------      --------
    ------
Elizabeth     5500     2500     0        0        4500
    12500
Emily              4000     6000     4400     2000     0
    16400
Thomas             2000     1000     6000     4000     1500
    14500
              --------      -------  -------- -------  --------      -----
              11500    9500     10400  6000     6000
    43400
```

- Listing 4.2 is a cross-tabular SQL*Plus command file. This report passes a command-line parameter (&1 on line 1) and assigns it to the variable name RPT_DATE.
- The RPT_DATE variable is then used in the headings (line 20) and again as part of the SQL query (line 31). Lines 2 through 5 suspend all terminal output from the SQL*Plus environment. The report is created in the operating system_dependent file SALES.OUT.
- Column-formatting commands control the appearance of the columns (lines 6 through 12). The combination of compute commands (lines 14 through 19), the sum statements in the query (lines 24 through 29), and the group by clause in the query (line 32) give the report output the appearance of a cross-tabular report.

**Note:**

- TTITLE technique in Listing 4.2 (lines 20 and 21) different from that of Listing 4.1 (line 13).

contd.

**<u>Advanced Reporting Techniques (contd.)</u>:**

- Listing 4.3 displays a major break field with the supporting data immediately following.
- Listing 4.3. Master/detail SQL*Plus report code.

```
ttitle `Sales Detail | by Sales Rep'
set HEADINGS OFF
column DUMMY NOPRINT
select 1 DUMMY, SALES_REP_NO,'Sales Person: ` || SALES_REP
from sales
UNION
select 2 DUMMY,SALES_REP_NO,'--------------------'
from sales
UNION
select 3 DUMMY,SALES_REP_NO, rpad(CAR_MAKE,4) || `    ` ||
    to_char(SALE_AMT,'$999,999.99')
from sales_detail
UNION
select 4 DUMMY,SALES_REP_NO,'        ----------'
from sales
UNION
select 5 DUMMY,SALES_REP_NO,'Total:  ` ||
    to_char(sum(TOTAL_SALES),'$999,999.99')
from sales
UNION
select 6 DUMMY,SALES_REP_NO,'         `
from sales
order by 2,1,3
/
```

contd.

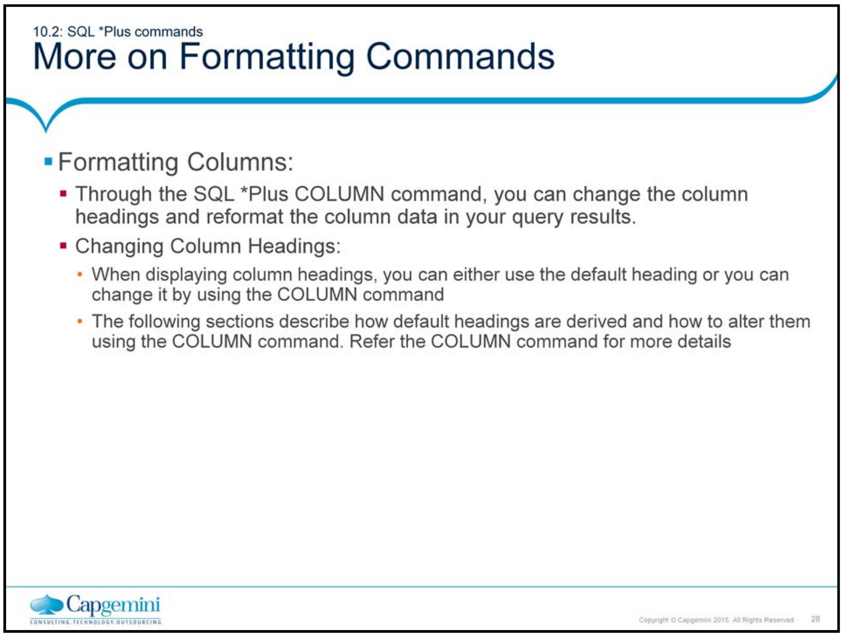
**Advanced Reporting Techniques (contd.)**:

- The following code shows the output report from Listing 4.3:

```
Thur Aug 31                                page   1
                   Sales Detail
                   by Sales Rep
Sales Person:  Elizabeth
---------------------------
Chrysler        $3,000
Chrysler        $1,500
Nissan          $2,000
Nissan          $2,000
Nissan          $1,500
Toyota          $2,500
                ----------
Total:       $12,500
Sales Person:   Emily
---------------------------
Ford         $1,000
Ford         $1,000
GM              $2,000
GM              $2,400
Nissan          $2,000
Nissan          $2,000
Toyota          $1,000
Toyota          $2,500
Toyota          $2,500
             ----------
Total:       $16,400
Sales Person:   Thomas
---------------------------
Chrysler        $1,500
Ford         $1,000
Ford         $3,000
GM              $1,400
GM              $1,600
GM              $3,000
Nissan          $2,000
Toyota          $1,000
                ----------
Total:       $16,400
```

## Advanced Reporting Techniques (contd.):

- Listing 4.3 creates a master/detail SQL*Plus report by using the SQL UNION command.
- In this example, there are six distinct, separate types of lines to be printed:
    - the sales person (line 4),
    - a line of dashes under the sales person (line 7),
    - the detail line (line 10),
    - a line of dashes under the detail total (line 14),
    - a total line (line 17), and
    - a blank line (line 21)
- There are six separate queries that have their output merged and sorted together by the SQL JOIN statement (lines 6, 9, 13, 16, 19, and 23).
- When you use JOIN to merge the output of two or more queries, the output result set must have the same number of columns.
- The headings are turned off (line 2) because regular SQL*Plus column headings are not desired for this type of report.
- The first column of each query has an alias column name of DUMMY. This DUMMY column is used to sort the order of the six types of lines (denoted by each of the six queries). The DUMMY column's only role is to maintain the order of the lines within the major sort field (SALES_REP_NO, in this example). Therefore, the NOPRINT option is specified in line 3.



10.2: SQL *Plus commands

## More on Formatting Commands

- Formatting Columns:
  - Through the SQL *Plus COLUMN command, you can change the column headings and reformat the column data in your query results.
  - Changing Column Headings:
    - When displaying column headings, you can either use the default heading or you can change it by using the COLUMN command
    - The following sections describe how default headings are derived and how to alter them using the COLUMN command. Refer the COLUMN command for more details

Copyright © Capgemini 2015. All Rights Reserved 28

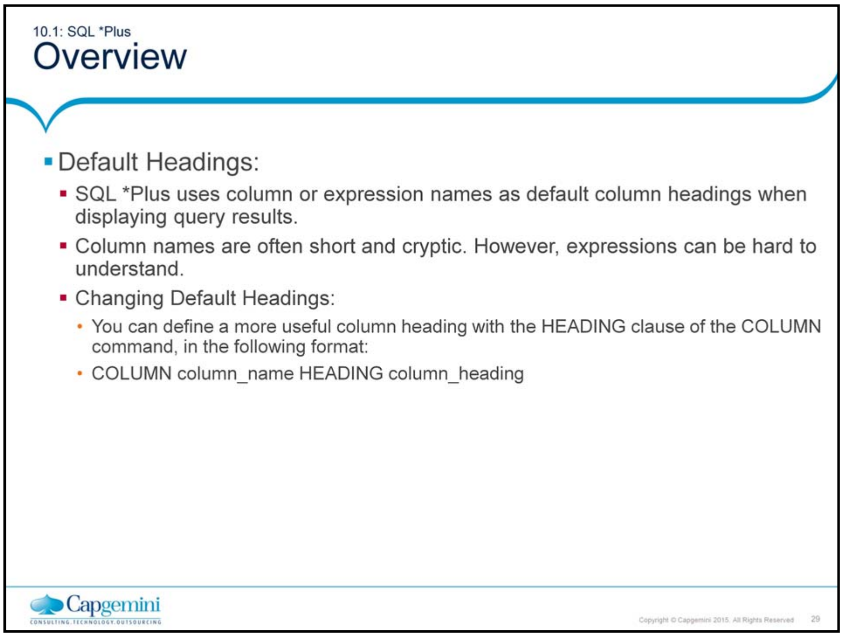
**<u>More on Formatting Commands</u>:**
**Formatting Columns:**
Examples:

- To format the output of a currency value, you can use the following code:

```
COLUMN sal FORMAT $99,999.00 HEADING
Salary
```

```
COLUMN home_dir NEW_VALUE home_path
NOPRINT
```

- To assign a value to home_dir you can use the COLUMN command. The first SQL query can reference the home_dir. All other SQL queries can then reference the home_path for the information returned by the first SQL query:



10.1: SQL *Plus

## Overview

- Default Headings:
  - SQL *Plus uses column or expression names as default column headings when displaying query results.
  - Column names are often short and cryptic. However, expressions can be hard to understand.
  - Changing Default Headings:
    - You can define a more useful column heading with the HEADING clause of the COLUMN command, in the following format:
    - COLUMN column_name HEADING column_heading

Copyright © Capgemini 2015. All Rights Reserved 29

**<u>More on Formatting Commands</u>:**

**COMPUTE function: OF options and ON break options**

- COMPUTE calculates and prints totals for groupings of rows defined by the BREAK command. You can use a variety of standard functions. The most common option is the name of the column in the query on which the total has to be calculated.
- The break option determines where the totals are to be printed and reset, as defined by the BREAK command.

  **For example:**

  - ➢ The following list produces a report with totals of monthly_sales and commissions when the sales_rep column value changes:

```
BREAK ON sales_rep SKIP 2
BREAK ON REPORT
COMPUTE SUM OF monthly_sales ON sales_rep
COMPUTE SUM OF commissions ON sales_rep
COMPUTE SUM OF monthly_sales ON REPORT
COMPUTE SUM OF commissions ON REPORT
```

  - ➢ It then skips two lines and produces monthly_sales and commissions totals at the end of the report.

**Note:**

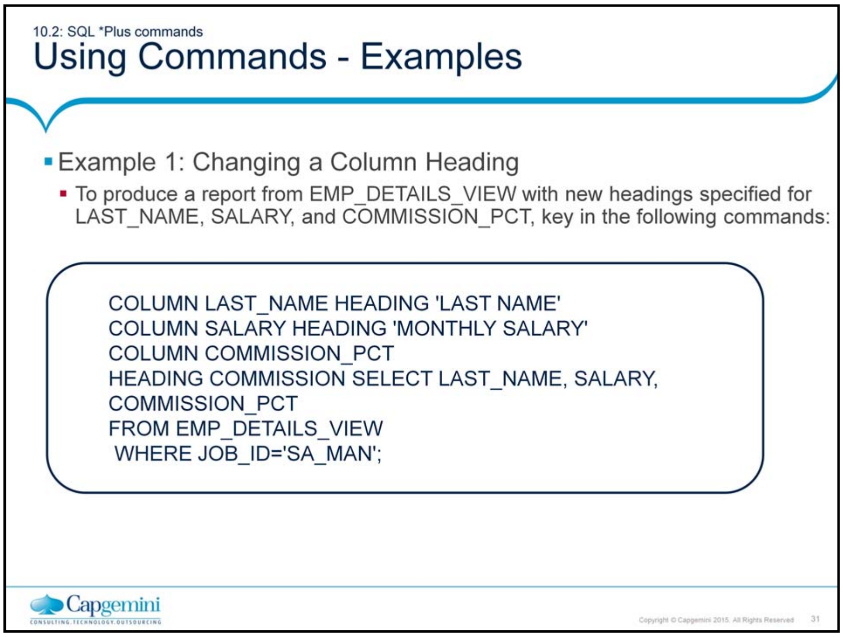
- The COMPUTE command resets the accumulator fields back to zero after printing.

**TTITLE print_options and / or text or variable options**

- TTITLE places text at the top of each page.
- You can use various print options that position text at various locations.
- TTITLE centers the text and adds date and page numbers if no print options are specified.
- Print options include BOLD, CENTER, COL, FORMAT, LEFT, RIGHT, SKIP, and TAB.
- TTITLE with no options at all, displays the current text setting.
  - ➢ Other options you can specify are ON and OFF.
  - ➢ TTITLE is ON by default.

**CLEAR and options**

- CLEAR resets any of the SQL*Plus formatting commands. You can also use it to clear the screen. The options include BREAKS, BUFFER, COLUMNS, COMPUTES, SCREEN, SQL, and TIMING.



10.2: SQL *Plus commands

## Using Commands - Examples

- Example 1: Changing a Column Heading
  - To produce a report from EMP_DETAILS_VIEW with new headings specified for LAST_NAME, SALARY, and COMMISSION_PCT, key in the following commands:

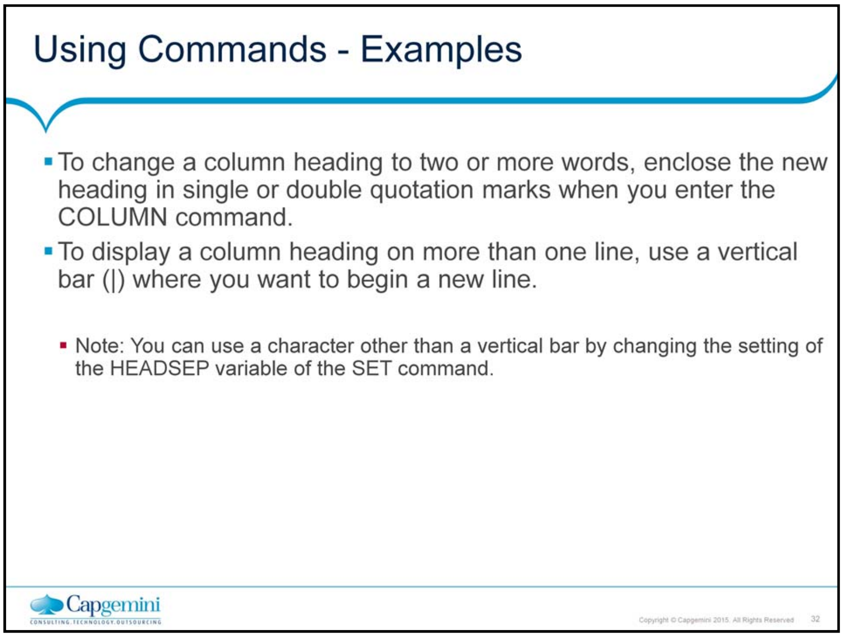
```
COLUMN LAST_NAME HEADING 'LAST NAME'
COLUMN SALARY HEADING 'MONTHLY SALARY'
COLUMN COMMISSION_PCT
HEADING COMMISSION SELECT LAST_NAME, SALARY,
COMMISSION_PCT
FROM EMP_DETAILS_VIEW
 WHERE JOB_ID='SA_MAN';
```

Copyright © Capgemini 2015. All Rights Reserved 31

**Output:**

```
LAST NAME      MONTHLY SALARY
               COMMISSION
-------------  ----------------------            ----------------
Russell        14000                             .4
Partners       13500                             .3
Errazuriz      12000                             .3
```

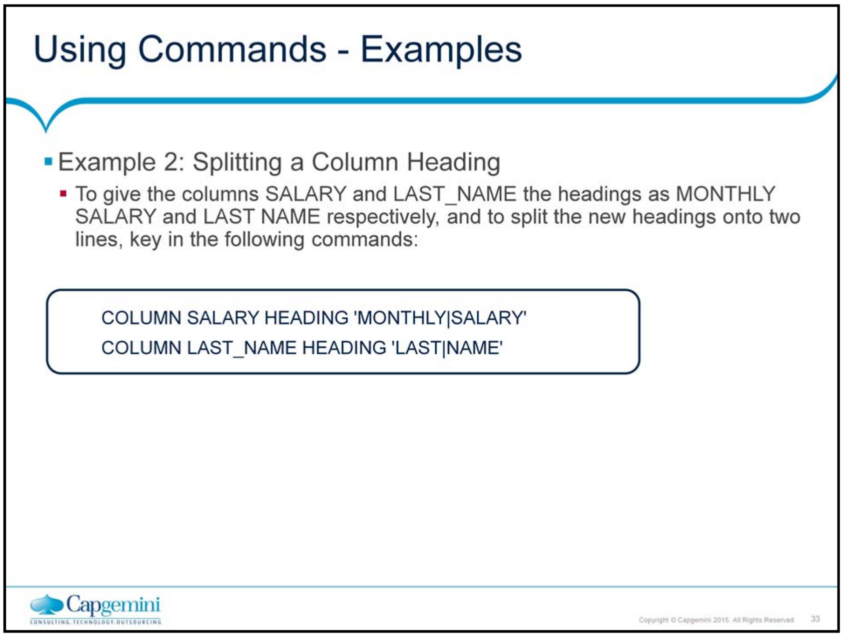
**Note:** The new headings will remain in effect until you enter different headings, reset each column's format, or exit from SQL *Plus.



## Using Commands - Examples

- To change a column heading to two or more words, enclose the new heading in single or double quotation marks when you enter the COLUMN command.
- To display a column heading on more than one line, use a vertical bar (|) where you want to begin a new line.
  - Note: You can use a character other than a vertical bar by changing the setting of the HEADSEP variable of the SET command.

Copyright © Capgemini 2015. All Rights Reserved 32



## Using Commands - Examples

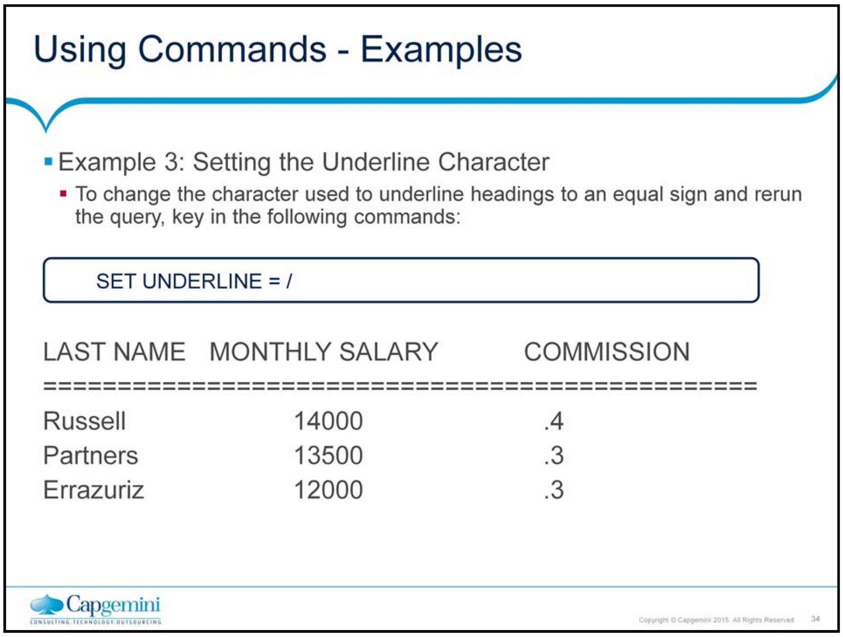
- Example 2: Splitting a Column Heading
  - To give the columns SALARY and LAST_NAME the headings as MONTHLY SALARY and LAST NAME respectively, and to split the new headings onto two lines, key in the following commands:

```
COLUMN SALARY HEADING 'MONTHLY|SALARY'
COLUMN LAST_NAME HEADING 'LAST|NAME'
```

Copyright © Capgemini 2015. All Rights Reserved 33

Output:

```
LAST NAME      MONTHLY
               COMMISSION
NAME           SALARY
---------------  ----------------------              -----------------
Russell        14000                                 .4
Partners       13500                                 .3
Errazuriz      12000                                 .3
```



## Using Commands - Examples

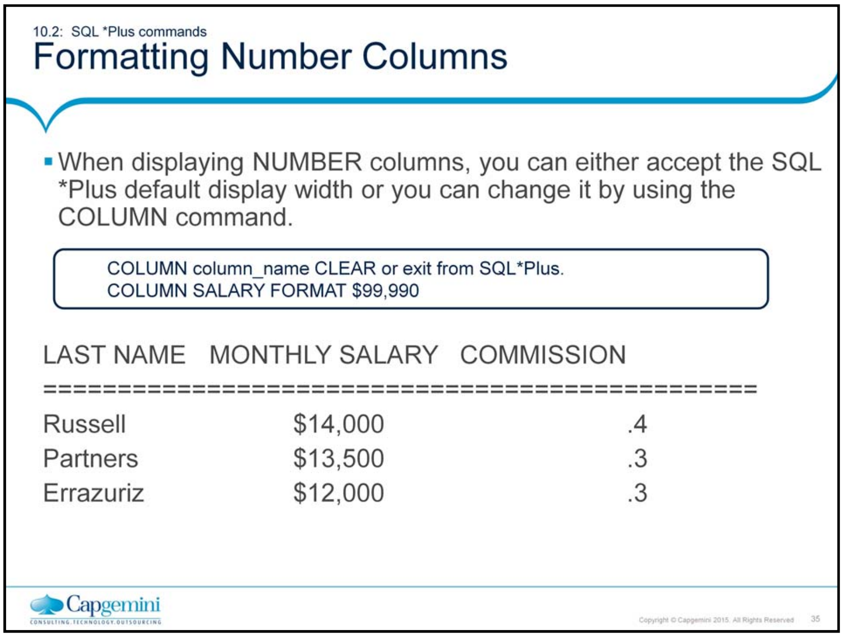
- Example 3: Setting the Underline Character
  - To change the character used to underline headings to an equal sign and rerun the query, key in the following commands:

```
SET UNDERLINE = /
```

```
LAST NAME   MONTHLY SALARY        COMMISSION
============================================
Russell              14000           .4
Partners             13500           .3
Errazuriz            12000           .3
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 34



10.2: SQL *Plus commands

## Formatting Number Columns

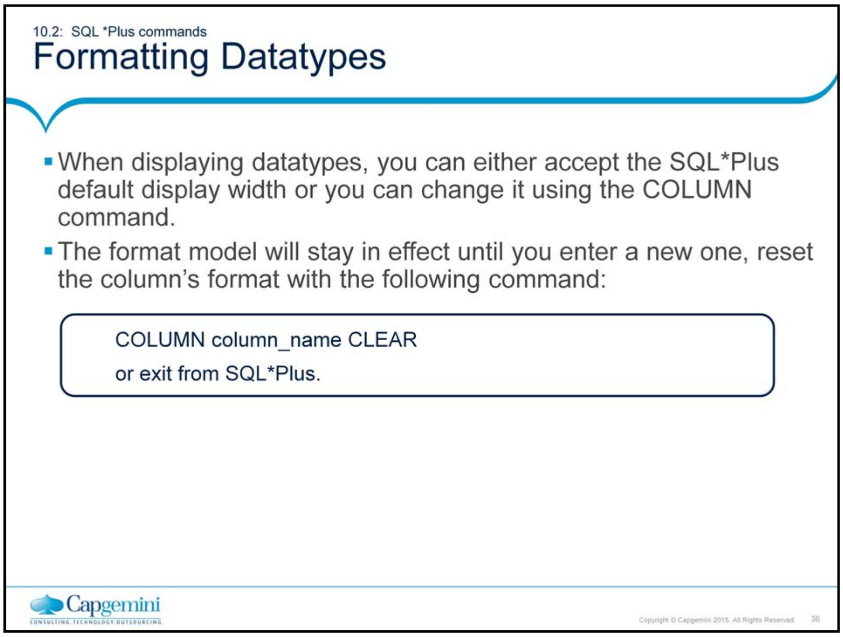
- When displaying NUMBER columns, you can either accept the SQL *Plus default display width or you can change it by using the COLUMN command.

```
COLUMN column_name CLEAR or exit from SQL*Plus.
COLUMN SALARY FORMAT $99,990
```

```
LAST NAME    MONTHLY SALARY   COMMISSION
==============================================
Russell             $14,000              .4
Partners            $13,500              .3
Errazuriz           $12,000              .3
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 35

**Note:**
The above command describe the default display and how you can alter it with the COLUMN command.



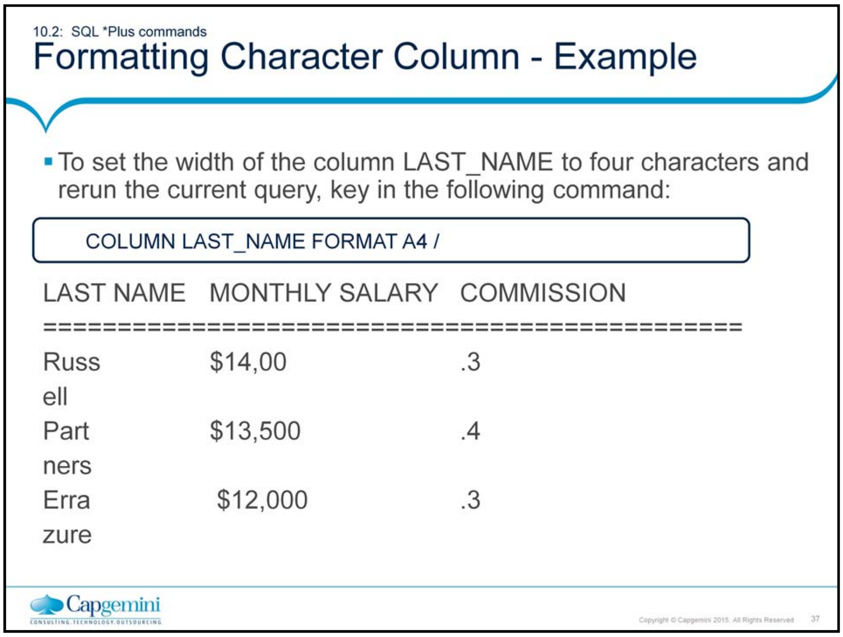
10.2: SQL *Plus commands

## Formatting Datatypes

- When displaying datatypes, you can either accept the SQL*Plus default display width or you can change it using the COLUMN command.
- The format model will stay in effect until you enter a new one, reset the column's format with the following command:

```
COLUMN column_name CLEAR
or exit from SQL*Plus.
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved 36



10.2: SQL *Plus commands

## Formatting Character Column - Example

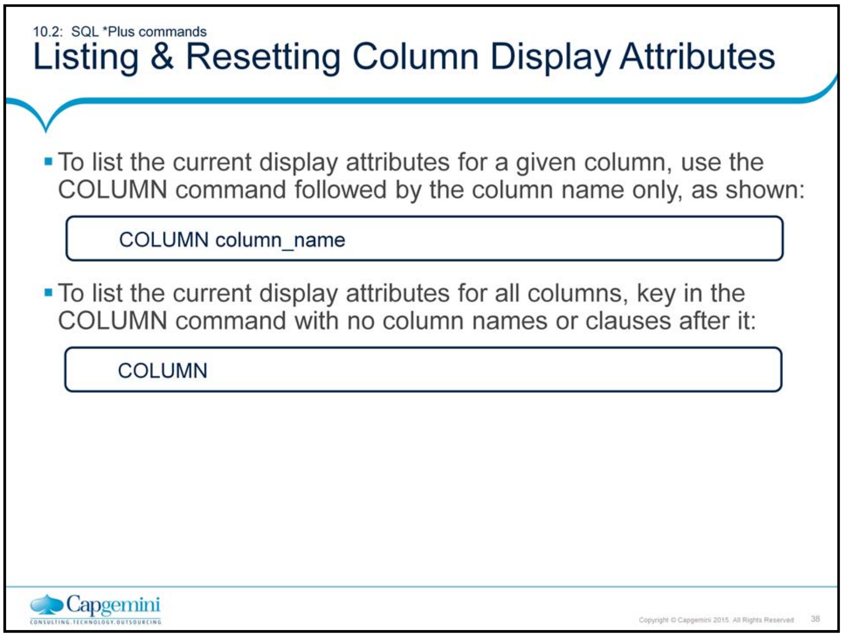
- To set the width of the column LAST_NAME to four characters and rerun the current query, key in the following command:

```
COLUMN LAST_NAME FORMAT A4 /
```

```
LAST NAME   MONTHLY SALARY   COMMISSION
==============================================
Russ         $14,00            .3
ell
Part         $13,500           .4
ners
Erra          $12,000          .3
zure
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 37



10.2: SQL *Plus commands

## Listing & Resetting Column Display Attributes

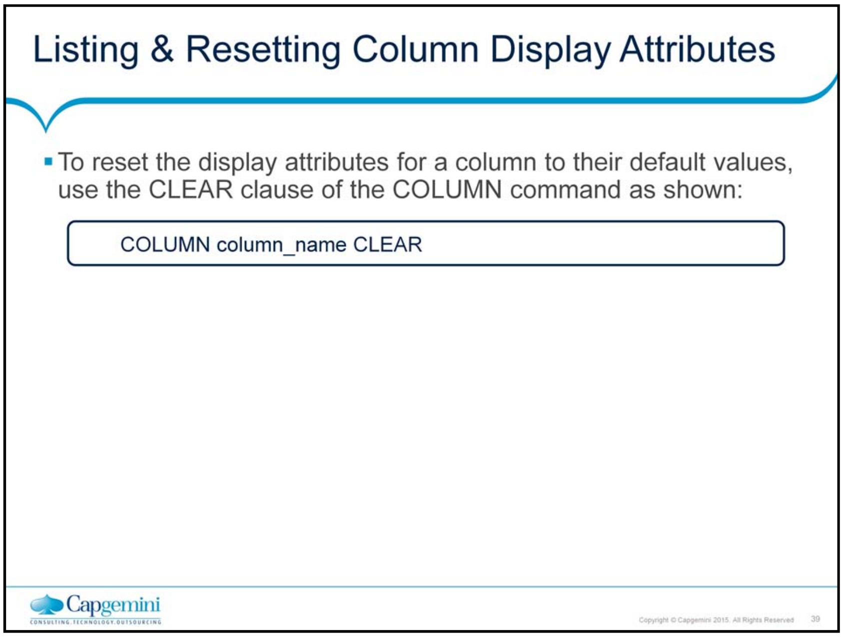
- To list the current display attributes for a given column, use the COLUMN command followed by the column name only, as shown:

```
COLUMN column_name
```

- To list the current display attributes for all columns, key in the COLUMN command with no column names or clauses after it:

```
COLUMN
```

Copyright © Capgemini 2015. All Rights Reserved 38

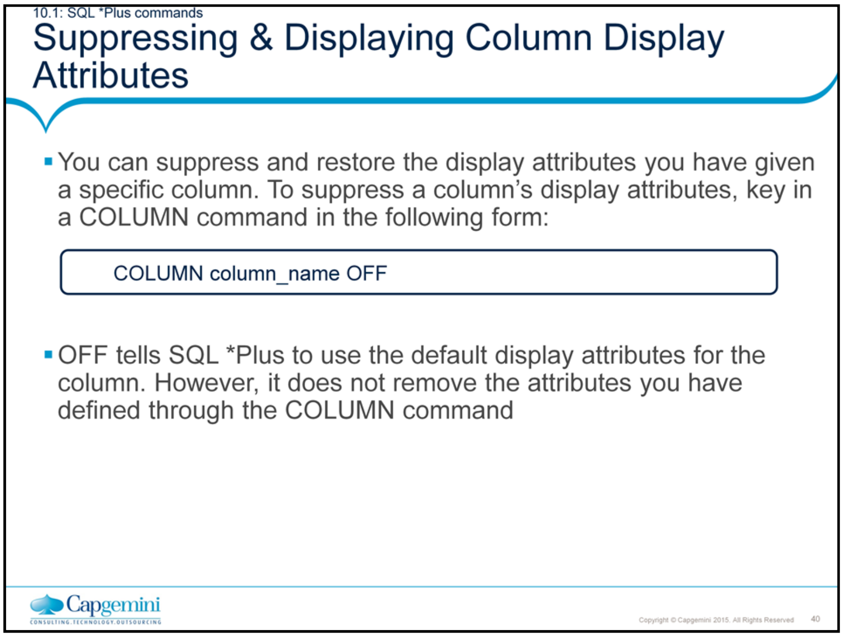


## Listing & Resetting Column Display Attributes

- To reset the display attributes for a column to their default values, use the CLEAR clause of the COLUMN command as shown:

```
COLUMN column_name CLEAR
```

Copyright © Capgemini 2015. All Rights Reserved 39



10.1: SQL *Plus commands

## Suppressing & Displaying Column Display Attributes

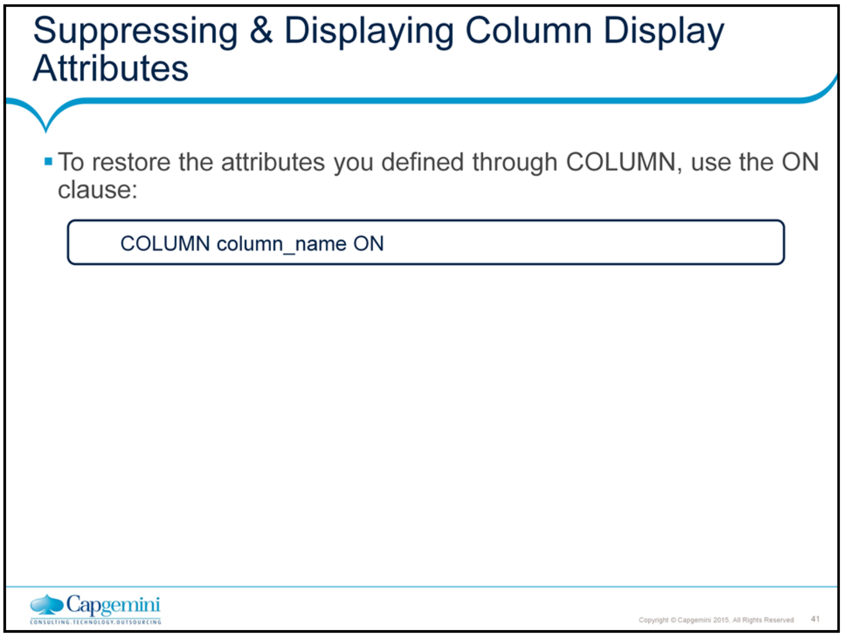
- You can suppress and restore the display attributes you have given a specific column. To suppress a column's display attributes, key in a COLUMN command in the following form:

```
COLUMN column_name OFF
```

- OFF tells SQL *Plus to use the default display attributes for the column. However, it does not remove the attributes you have defined through the COLUMN command

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 40

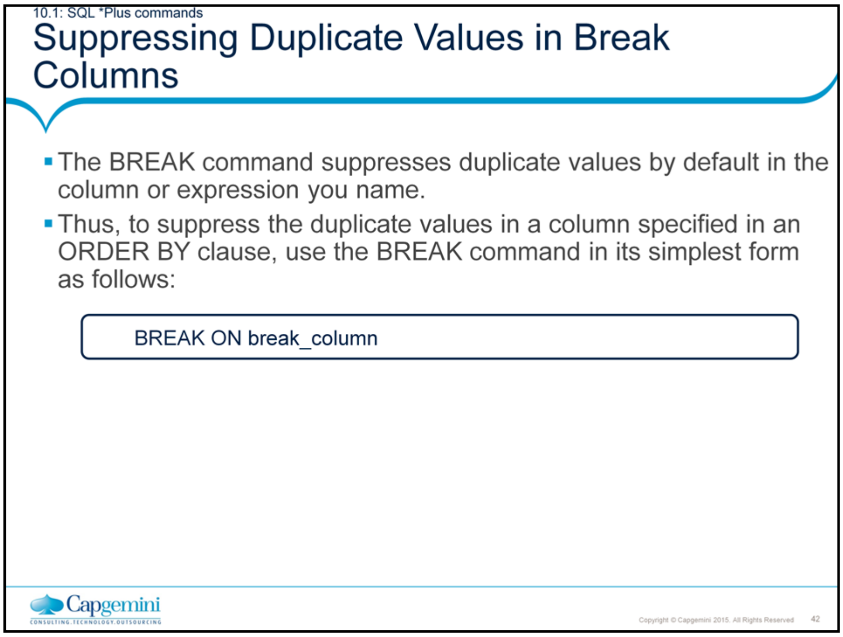


## Suppressing & Displaying Column Display Attributes

- To restore the attributes you defined through COLUMN, use the ON clause:

```
COLUMN column_name ON
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING Copyright © Capgemini 2015. All Rights Reserved 41



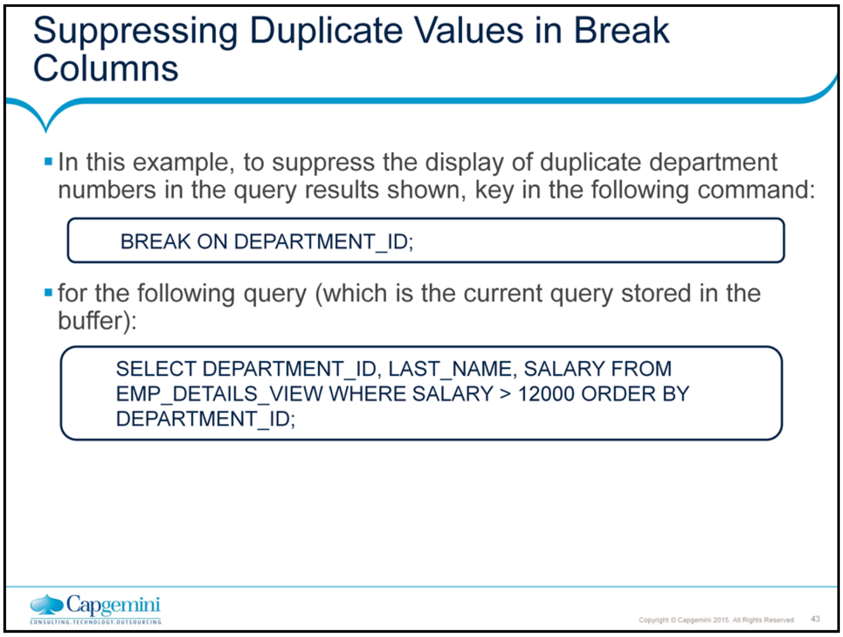
10.1: SQL *Plus commands

## Suppressing Duplicate Values in Break Columns

- The BREAK command suppresses duplicate values by default in the column or expression you name.
- Thus, to suppress the duplicate values in a column specified in an ORDER BY clause, use the BREAK command in its simplest form as follows:

```
BREAK ON break_column
```

Copyright © Capgemini 2015. All Rights Reserved 42



## Suppressing Duplicate Values in Break Columns

- In this example, to suppress the display of duplicate department numbers in the query results shown, key in the following command:

```
BREAK ON DEPARTMENT_ID;
```

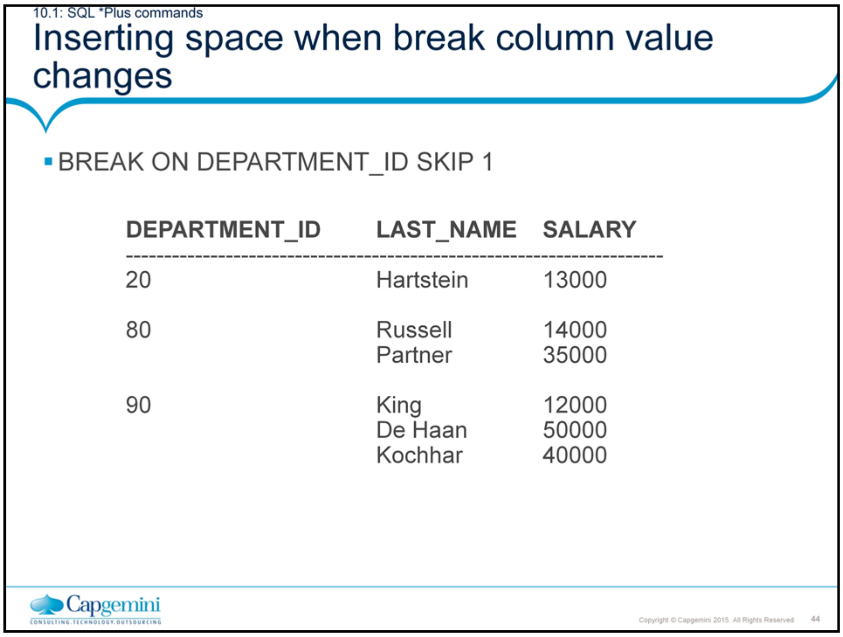
- for the following query (which is the current query stored in the buffer):

```
SELECT DEPARTMENT_ID, LAST_NAME, SALARY FROM
EMP_DETAILS_VIEW WHERE SALARY > 12000 ORDER BY
DEPARTMENT_ID;
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING Copyright © Capgemini 2015. All Rights Reserved 43

Output:

```
DEPARTMENT_ID                 LAST_NAME    SALARY
-------------------------------------------------------------
20                            Hartstein    13000
80                            Russell      14000
                              Partner      35000
90                            King         12000
                              De Haan      50000
                              Kochhar      40000
```



10.1: SQL *Plus commands

## Inserting space when break column value changes

- BREAK ON DEPARTMENT_ID SKIP 1

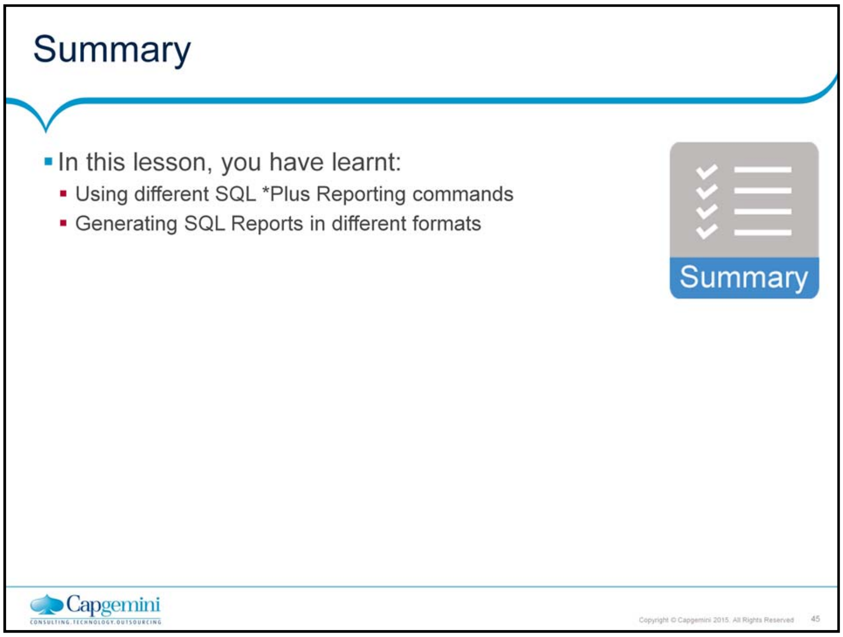
```
DEPARTMENT_ID     LAST_NAME   SALARY
---------------------------------------------------------------------
20                Hartstein    13000

80                Russell      14000
                  Partner      35000

90                King         12000
                  De Haan      50000
                  Kochhar      40000
```

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

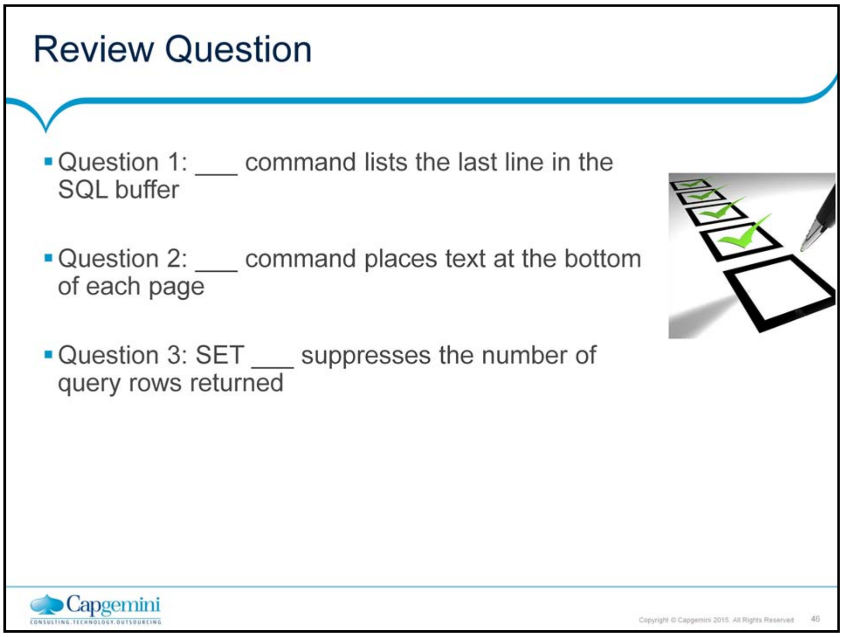
Copyright © Capgemini 2015. All Rights Reserved 44



## Summary

- In this lesson, you have learnt:
  - Using different SQL *Plus Reporting commands
  - Generating SQL Reports in different formats

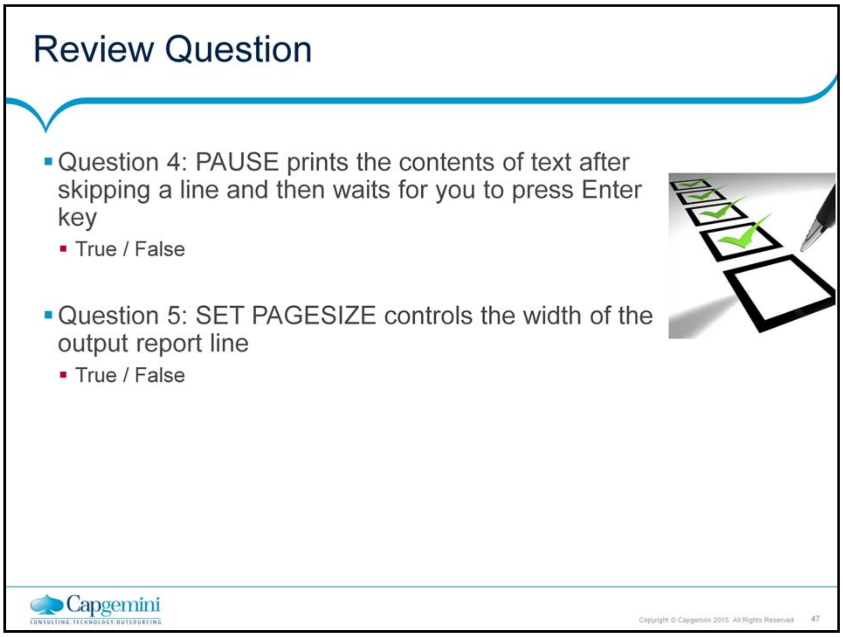
Copyright © Capgemini 2015. All Rights Reserved 45



## Review Question

- Question 1: ___ command lists the last line in the SQL buffer
- Question 2: ___ command places text at the bottom of each page
- Question 3: SET ___ suppresses the number of query rows returned

Copyright © Capgemini 2015. All Rights Reserved 46



## Review Question

- Question 4: PAUSE prints the contents of text after skipping a line and then waits for you to press Enter key
  - True / False
- Question 5: SET PAGESIZE controls the width of the output report line
  - True / False

Copyright © Capgemini 2015. All Rights Reserved 47