Producing Readable Output with iSQL*Plus
Objectives

After completing this lesson, you should be able to do the following:

• Produce queries that require a substitution variable
• Customize the iSQL*Plus environment
• Produce more readable output
• Create and execute script files
Substitution Variables

I want to query different values.

... salary = ? ...
... department_id = ? ...
... last_name = ? ...
Substitution Variables

Use iSQL*Plus substitution variables to:

• Temporarily store values
  – Single ampersand (&)
  – Double ampersand (&&)
  – DEFINE command

• Pass variable values between SQL statements

• Dynamically alter headers and footers
Using the & Substitution Variable

Use a variable prefixed with an ampersand (&) to prompt the user for a value.

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = &employee_num;
```
Using the & Substitution Variable

Define Substitution Variables

"employee_num" 101

old 3: WHERE employee_id = &employee_num
new 3: WHERE employee_id = 101

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>17000</td>
<td>90</td>
</tr>
</tbody>
</table>
Character and Date Values with Substitution Variables

Use single quotation marks for date and character values.

```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title';
```

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>DEPARTMENT_ID</th>
<th>SALARY*12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hunold</td>
<td>60</td>
<td>108000</td>
</tr>
<tr>
<td>Ernst</td>
<td>60</td>
<td>72000</td>
</tr>
<tr>
<td>Lorentz</td>
<td>60</td>
<td>50400</td>
</tr>
</tbody>
</table>
Specifying Column Names, Expressions, and Text

Use substitution variables to supplement the following:

- **WHERE** conditions
- **ORDER BY** clauses
- Column expressions
- Table names
- Entire **SELECT** statements
Specifying Column Names, Expressions, and Text

```
SELECT employee_id, last_name, job_id, &column_name
FROM employees
WHERE &condition
ORDER BY &order_column;
```

Define Substitution Variables

- "column_name": salary
- "condition": salary > 15000
- "order_column": last_name

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>De Haan</td>
<td>AD_VP</td>
<td>17000</td>
</tr>
<tr>
<td>100</td>
<td>King</td>
<td>AD_PRES</td>
<td>24000</td>
</tr>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>AD_VP</td>
<td>17000</td>
</tr>
</tbody>
</table>
Defining Substitution Variables

- You can predefine variables using the \texttt{SQL*Plus DEFINE} command.
  
  \texttt{DEFINE variable = value} creates a user variable with the CHAR data type.

- If you need to predefine a variable that includes spaces, you must enclose the value within single quotation marks when using the \texttt{DEFINE} command.

- A defined variable is available for the session
DEFINE and UNDEFINE Commands

• A variable remains defined until you either:
  – Use the UNDEFINE command to clear it
  – Exit SQL*Plus

• You can verify your changes with the DEFINE command.

```
DEFINE job_title = IT_PROG
DEFINE job_title
DEFINE JOB_TITLE = "IT_PROG" (CHAR)
```

```
UNDEFINE job_title
DEFINE job_title
SP2-0135: symbol job_title is UNDEFINED
```
Using the DEFINE Command with & Substitution Variable

- Create the substitution variable using the DEFINE command.

```sql
DEFINE employee_num = 200
```

- Use a variable prefixed with an ampersand (&) to substitute the value in the SQL statement.

```sql
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE employee_id = &employee_num ;
```
Using the && Substitution Variable

Use the double-ampersand (&&) if you want to reuse the variable value without prompting the user each time.

```
SELECT employee_id, last_name, job_id, &&column_name
FROM employees
ORDER BY &&column_name;
```

Define Substitution Variables

```
"column_name" department_id
```

20 rows selected.
Using the \texttt{VERIFY} Command

Use the \texttt{VERIFY} command to toggle the display of the substitution variable, before and after \texttt{iSQL*Plus} replaces substitution variables with values.

\begin{verbatim}
SET VERIFY ON
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num;
\end{verbatim}

```
"employee_num" [200]
```

```
\begin{tabular}{ll}
old & 3: WHERE employee_id = &employee_num \\
new & 3: WHERE employee_id = 200 \\
\end{tabular}
```

\textcopyright~Oracle Corporation, 2001. All rights reserved.
Customizing the *i*SQL*Plus* Environment

- Use **SET** commands to control current session.

  ```
  SET system_variable value
  ```

- Verify what you have set by using the **SHOW** command.

  ```
  SET ECHO ON
  SHOW ECHO
  echo ON
  ```
SET Command Variables

- ARRAYSIZE \{20 | n\}
- FEEDBACK \{6 | n | OFF | ON\}
- HEADING \{OFF | ON\}
- LONG \{80 | n | ON | text\}

SET HEADING OFF
SHOW HEADING
HEADING OFF
iSQL*Plus Format Commands

- COLUMN [column option]
- TTITLE [text | OFF | ON]
- BTITLE [text | OFF | ON]
- BREAK [ON report_element]
The COLUMN Command

Controls display of a column:

```
COL[UMN] [{column|alias} [option]]
```

- **CLE [AR]**: Clears any column formats
- **HEA[DING] text**: Sets the column heading
- **FOR[MAT] format**: Changes the display of the column using a format model
- **NOPRINT | PRINT**
- **NULL**
Using the \texttt{COLUMN} Command

- Create column headings.

\begin{verbatim}
COLUMN last_name HEADING 'Employee\textbar Name'
COLUMN salary JUSTIFY LEFT FORMAT $99,990.00
COLUMN manager FORMAT 999999999 NULL 'No manager'
\end{verbatim}

- Display the current setting for the \texttt{LAST\_NAME} column.

\begin{verbatim}
COLUMN last_name
\end{verbatim}

- Clear settings for the \texttt{LAST\_NAME} column.

\begin{verbatim}
COLUMN last_name CLEAR
\end{verbatim}
# COLUMN Format Models

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Single zero-suppression digit</td>
<td>999999</td>
<td>1234</td>
</tr>
<tr>
<td>0</td>
<td>Enforces leading zero</td>
<td>099999</td>
<td>001234</td>
</tr>
<tr>
<td>$</td>
<td>Floating dollar sign</td>
<td>$9999</td>
<td>$1234</td>
</tr>
<tr>
<td>L</td>
<td>Local currency</td>
<td>L9999</td>
<td>L1234</td>
</tr>
<tr>
<td>.</td>
<td>Position of decimal point</td>
<td>9999.99</td>
<td>1234.00</td>
</tr>
<tr>
<td>,</td>
<td>Thousand separator</td>
<td>9,999</td>
<td>1,234</td>
</tr>
</tbody>
</table>
Using the **BREAK** Command

Use the **BREAK** command to suppress duplicates.

```
BREAK ON job_id
```
Using the TTITLE and BTITLE Commands

- Display headers and footers.
  \[
  \text{TTITLE} \ [ \text{text}\text{OFF}\text{ON}] \\
  \text{TTITLE} \ 'Salary|Report'
  \]

- Set the report header.
  \[
  \text{BTITLE} \ 'Confidential'
  \]
Using the \texttt{TTITLE} and \texttt{BTITLE} Commands

- Display headers and footers.
  \begin{verbatim}
  \texttt{TTITLE [text|OFF|ON]}
  \end{verbatim}

- Set the report header.
  \begin{verbatim}
  \texttt{TTITLE 'Salary|Report'}
  \end{verbatim}

- Set the report footer.
  \begin{verbatim}
  \texttt{BTITLE 'Confidential'}
  \end{verbatim}
Creating a Script File to Run a Report

1. Create and test the SQL SELECT statement.
2. Save the SELECT statement into a script file.
3. Load the script file into an editor.
4. Add formatting commands before the SELECT statement.
5. Verify that the termination character follows the SELECT statement.
Creating a Script File to Run a Report

6. Clear formatting commands after the `SELECT` statement.
7. Save the script file.
8. Load the script file into the `iSQL*Plus text window`, and click the Execute button.
## Sample Report

**Fri Sep 28**

<table>
<thead>
<tr>
<th>Job Category</th>
<th>Employee</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC_ACCOUNT</td>
<td>Gietz</td>
<td>$8,300.00</td>
</tr>
<tr>
<td>AC_MGR</td>
<td>Higgins</td>
<td>$12,000.00</td>
</tr>
<tr>
<td>AD_ASST</td>
<td>Whalen</td>
<td>$4,400.00</td>
</tr>
<tr>
<td>IT_PROG</td>
<td>Ernst</td>
<td>$6,000.00</td>
</tr>
<tr>
<td></td>
<td>Hunold</td>
<td>$9,000.00</td>
</tr>
<tr>
<td></td>
<td>Lorentz</td>
<td>$4,200.00</td>
</tr>
<tr>
<td>MK_MAN</td>
<td>Hartstein</td>
<td>$13,000.00</td>
</tr>
<tr>
<td>MK_REP</td>
<td>Fay</td>
<td>$6,000.00</td>
</tr>
<tr>
<td>SA_MAN</td>
<td>Zlotkey</td>
<td>$10,500.00</td>
</tr>
<tr>
<td>SA_REP</td>
<td>Abel</td>
<td>$11,000.00</td>
</tr>
<tr>
<td></td>
<td>Grant</td>
<td>$7,000.00</td>
</tr>
<tr>
<td></td>
<td>Taylor</td>
<td>$8,600.00</td>
</tr>
</tbody>
</table>

Confidential
## Sample Report

<table>
<thead>
<tr>
<th>Job Category</th>
<th>Employee</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC_ACCOUNT</td>
<td>Gietz</td>
<td>$8,300.00</td>
</tr>
<tr>
<td>AC_MGR</td>
<td>Higgins</td>
<td>$12,000.00</td>
</tr>
<tr>
<td>AD_ASST</td>
<td>Whalen</td>
<td>$4,400.00</td>
</tr>
<tr>
<td>IT_PROG</td>
<td>Ernst</td>
<td>$6,000.00</td>
</tr>
<tr>
<td></td>
<td>Hunsold</td>
<td>$9,000.00</td>
</tr>
<tr>
<td></td>
<td>Lorentz</td>
<td>$4,200.00</td>
</tr>
<tr>
<td>MK_MAN</td>
<td>Hartstein</td>
<td>$13,000.00</td>
</tr>
<tr>
<td>MK_REP</td>
<td>Fay</td>
<td>$6,000.00</td>
</tr>
<tr>
<td>SA_MAN</td>
<td>Zlotkey</td>
<td>$10,500.00</td>
</tr>
<tr>
<td>SA_REP</td>
<td>Abel</td>
<td>$11,000.00</td>
</tr>
<tr>
<td></td>
<td>Grant</td>
<td>$7,000.00</td>
</tr>
<tr>
<td></td>
<td>Taylor</td>
<td>$8,600.00</td>
</tr>
</tbody>
</table>
Summary

In this lesson, you should have learned how to:

• Use *SQL*Plus substitution variables to store values temporarily

• Use SET commands to control the current *SQL*Plus environment

• Use the COLUMN command to control the display of a column

• Use the BREAK command to suppress duplicates and divide rows into sections

• Use the TTITLE and BTITLE commands to display headers and footers
Practice 7 Overview

This practice covers the following topics:

• Creating a query to display values using substitution variables
• Starting a command file containing variables