1

Retrieving Data Using the SQL SELECT Statement

Objectives

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

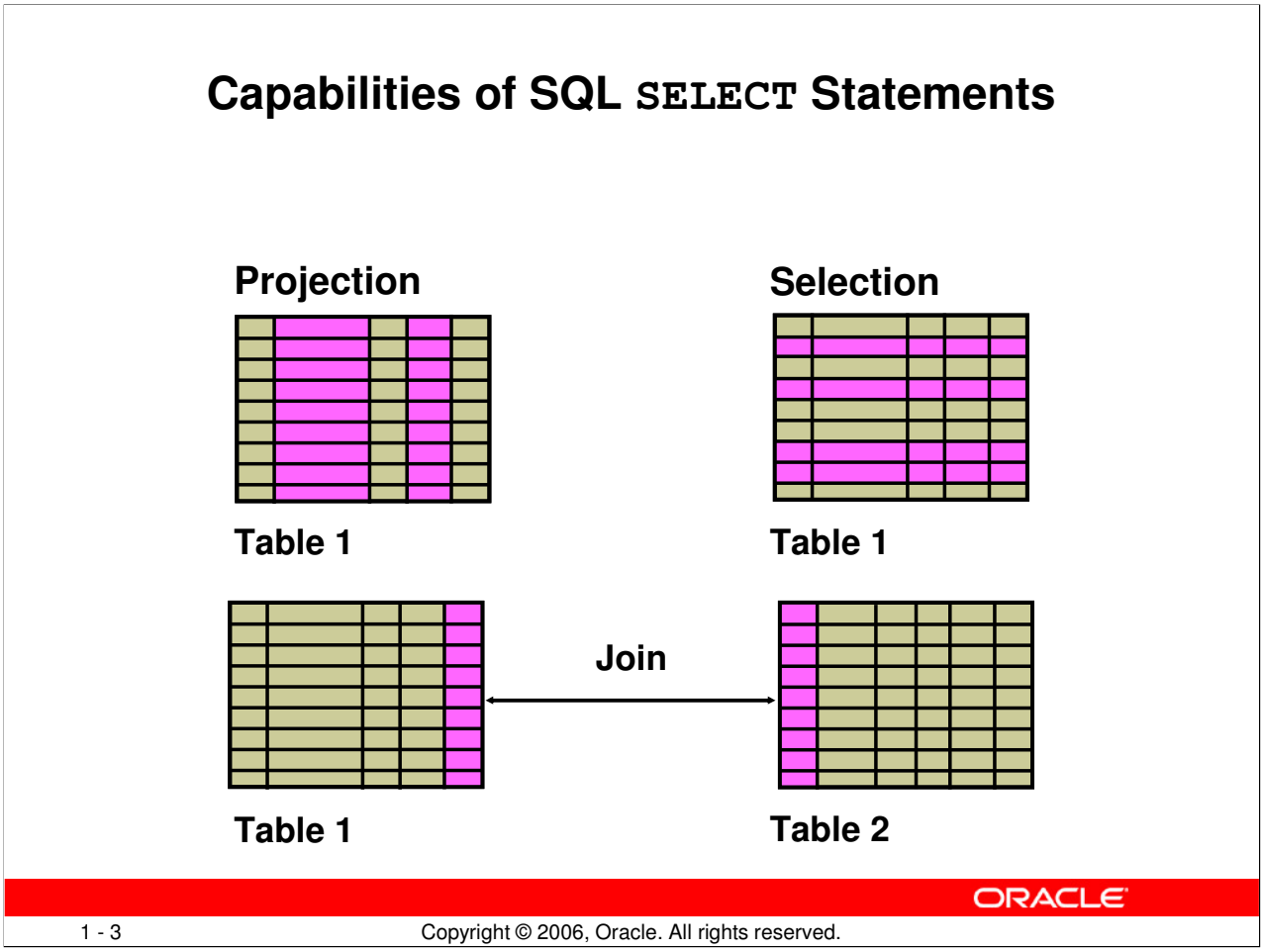
- **List the capabilities of SQL `SELECT` statements**
- **Execute a basic `SELECT` statement**
- **Differentiate between SQL statements and *i*SQL*Plus commands**

ORACLE

Objectives

To extract data from the database, you need to use the structured query language (SQL) `SELECT` statement. You may need to restrict the columns that are displayed. This lesson describes all the SQL statements that are needed to perform these actions. You may want to create `SELECT` statements that can be used more than once.

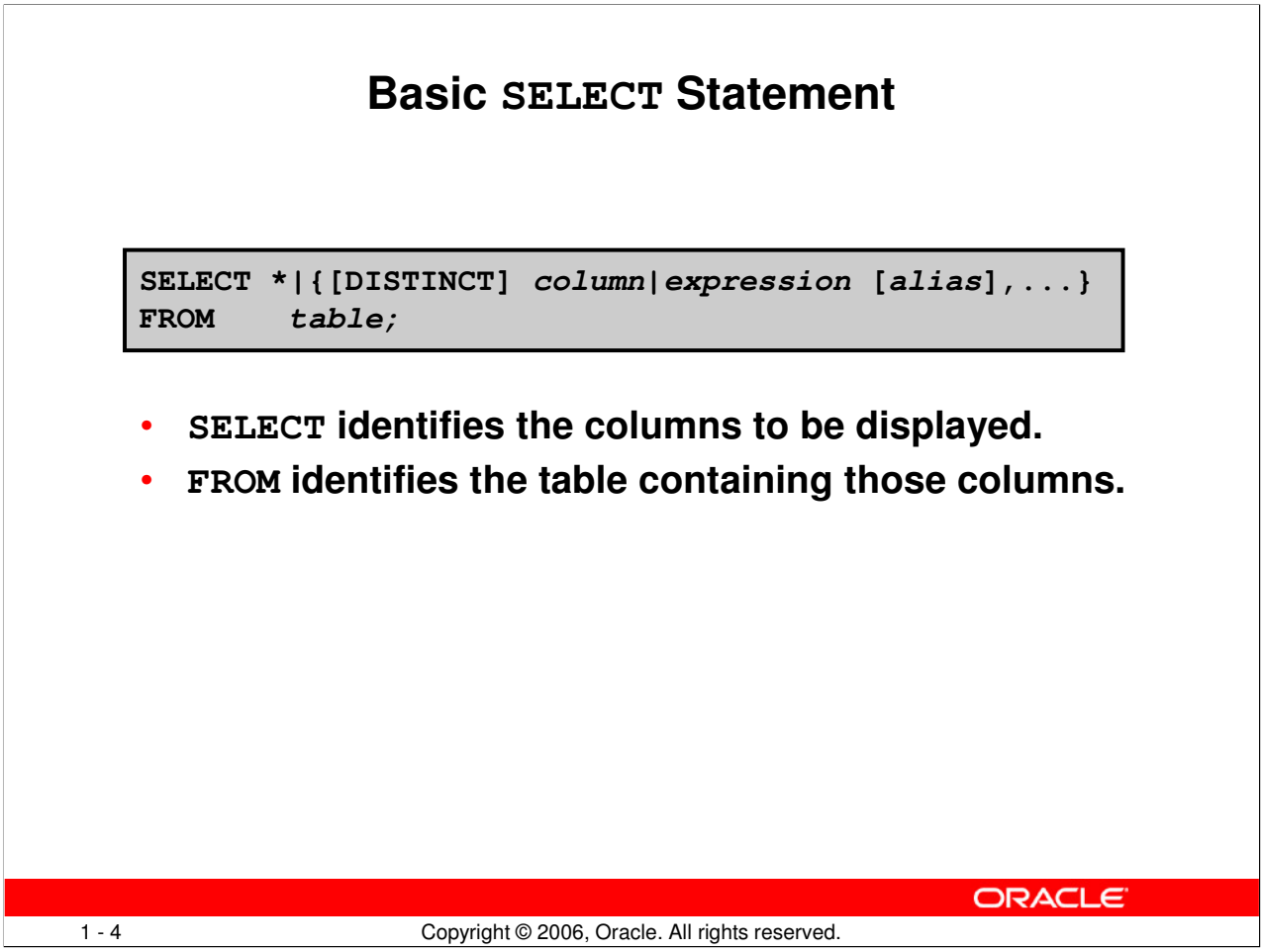
This lesson also covers the *i*SQL*Plus environment in which you execute SQL statements.



Capabilities of SQL SELECT Statements

A SELECT statement retrieves information from the database. With a SELECT statement, you can use the following capabilities:

- **Projection:** Choose the columns in a table that are returned by a query. Choose as few or as many of the columns as needed.
- **Selection:** Choose the rows in a table that are returned by a query. Various criteria can be used to restrict the rows that are retrieved.
- **Joining:** Bring together data that is stored in different tables by specifying the link between them. SQL joins are covered in more detail in the lesson titled "Displaying Data from Multiple Tables."



Basic SELECT Statement

In its simplest form, a SELECT statement must include the following:

- A SELECT clause, which specifies the columns to be displayed
- A FROM clause, which identifies the table containing the columns that are listed in the SELECT clause

In the syntax:

SELECT	is a list of one or more columns
*	selects all columns
DISTINCT	suppresses duplicates
column|expression	selects the named column or the expression
alias	gives selected columns different headings
FROM *table*	specifies the table containing the columns

Note: Throughout this course, the words *keyword*, *clause*, and *statement* are used as follows:

- A *keyword* refers to an individual SQL element.
  For example, SELECT and FROM are keywords.
- A *clause* is a part of a SQL statement.
  For example, SELECT employee_id, last_name, ... is a clause.
- A *statement* is a combination of two or more clauses.
  For example, SELECT * FROM employees is a SQL statement.

Selecting All Columns

```
SELECT *
FROM   departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

ORACLE

Selecting All Columns of All Rows

You can display all columns of data in a table by following the `SELECT` keyword with an asterisk (`*`). In the example in the slide, the department table contains four columns: `DEPARTMENT_ID`, `DEPARTMENT_NAME`, `MANAGER_ID`, and `LOCATION_ID`. The table contains seven rows, one for each department.

You can also display all columns in the table by listing all the columns after the `SELECT` keyword. For example, the following SQL statement (like the example in the slide) displays all columns and all rows of the `DEPARTMENTS` table:

```
SELECT  department_id, department_name, manager_id, location_id
FROM    departments;
```

Selecting Specific Columns

```
SELECT department_id, location_id
FROM   departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Selecting Specific Columns of All Rows

You can use the `SELECT` statement to display specific columns of the table by specifying the column names, separated by commas. The example in the slide displays all the department numbers and location numbers from the `DEPARTMENTS` table.

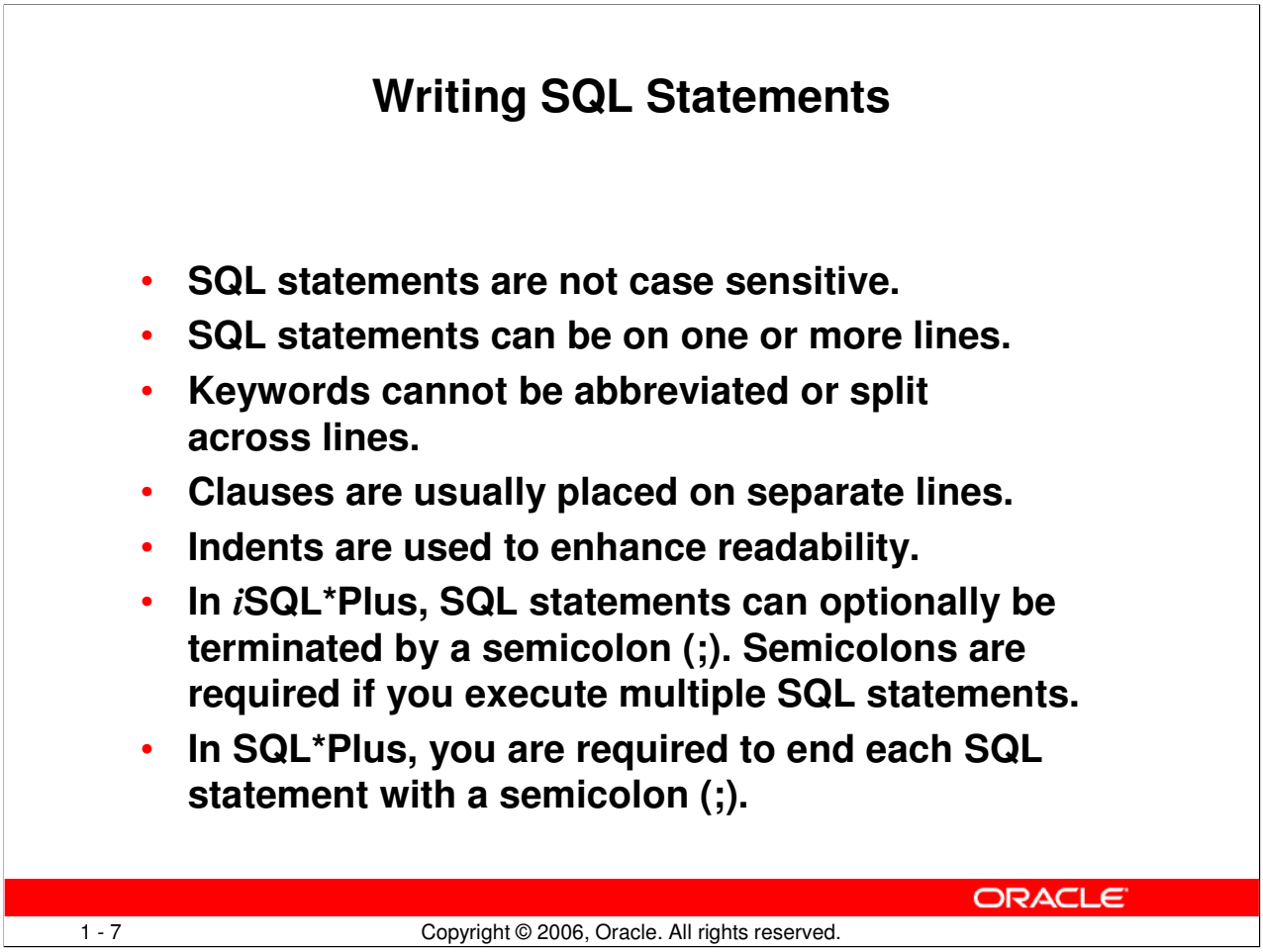
In the `SELECT` clause, specify the columns that you want, in the order in which you want them to appear in the output. For example, to display location before department number going from left to right, you use the following statement:

```
SELECT location_id, department_id
FROM   departments;
```

LOCATION_ID	DEPARTMENT_ID
1700	10
1800	20
1500	50

▪ ▪ ▪

8 rows selected.



Writing SQL Statements

Using the following simple rules and guidelines, you can construct valid statements that are both easy to read and easy to edit:

- SQL statements are not case sensitive (unless indicated).
- SQL statements can be entered on one or many lines.
- Keywords cannot be split across lines or abbreviated.
- Clauses are usually placed on separate lines for readability and ease of editing.
- Indents should be used to make code more readable.
- Keywords typically are entered in uppercase; all other words, such as table names and columns, are entered in lowercase.

Executing SQL Statements

Using *i*SQL*Plus, click the Execute button to run the command or commands in the editing window.

Using SQL*Plus, terminate the SQL statement with a semicolon and then press the Enter key to run the command.

Column Heading Defaults

- ***i*SQL*Plus:**
  - **Default heading alignment: Center**
  - **Default heading display: Uppercase**
- **SQL*Plus:**
  - **Character and Date column headings are left-aligned**
  - **Number column headings are right-aligned**
  - **Default heading display: Uppercase**

ORACLE

Column Heading Defaults

In *i*SQL*Plus, column headings are displayed in uppercase and centered.

```
SELECT last_name, hire_date, salary
FROM   employees;
```

LAST_NAME	HIRE_DATE	SALARY
King	17-JUN-87	24000
Kochhar	21-SEP-89	17000
De Haan	13-JAN-93	17000
Hunold	03-JAN-90	9000
Ernst	21-MAY-91	6000
▪ ▪ ▪		
Higgins	07-JUN-94	12000
Gietz	07-JUN-94	8300

20 rows selected.

You can override the column heading display with an alias. Column aliases are covered later in this lesson.

Arithmetic Expressions

Create expressions with number and date data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

ORACLE

Arithmetic Expressions

You may need to modify the way in which data is displayed, or you may want to perform calculations or look at what-if scenarios. These are all possible using arithmetic expressions. An arithmetic expression can contain column names, constant numeric values, and the arithmetic operators.

Arithmetic Operators

The slide lists the arithmetic operators that are available in SQL. You can use arithmetic operators in any clause of a SQL statement (except the `FROM` clause).

Note: With the `DATE` and `TIMESTAMP` data types, you can use the addition and subtraction operators only.

Using Arithmetic Operators

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...

20 rows selected.

Using Arithmetic Operators

The example in the slide uses the addition operator to calculate a salary increase of $300 for all employees. The slide also displays a `SALARY+300` column in the output.

Note that the resultant calculated column `SALARY+300` is not a new column in the `EMPLOYEES` table; it is for display only. By default, the name of a new column comes from the calculation that generated it—in this case, `salary+300`.

Note: The Oracle server ignores blank spaces before and after the arithmetic operator.

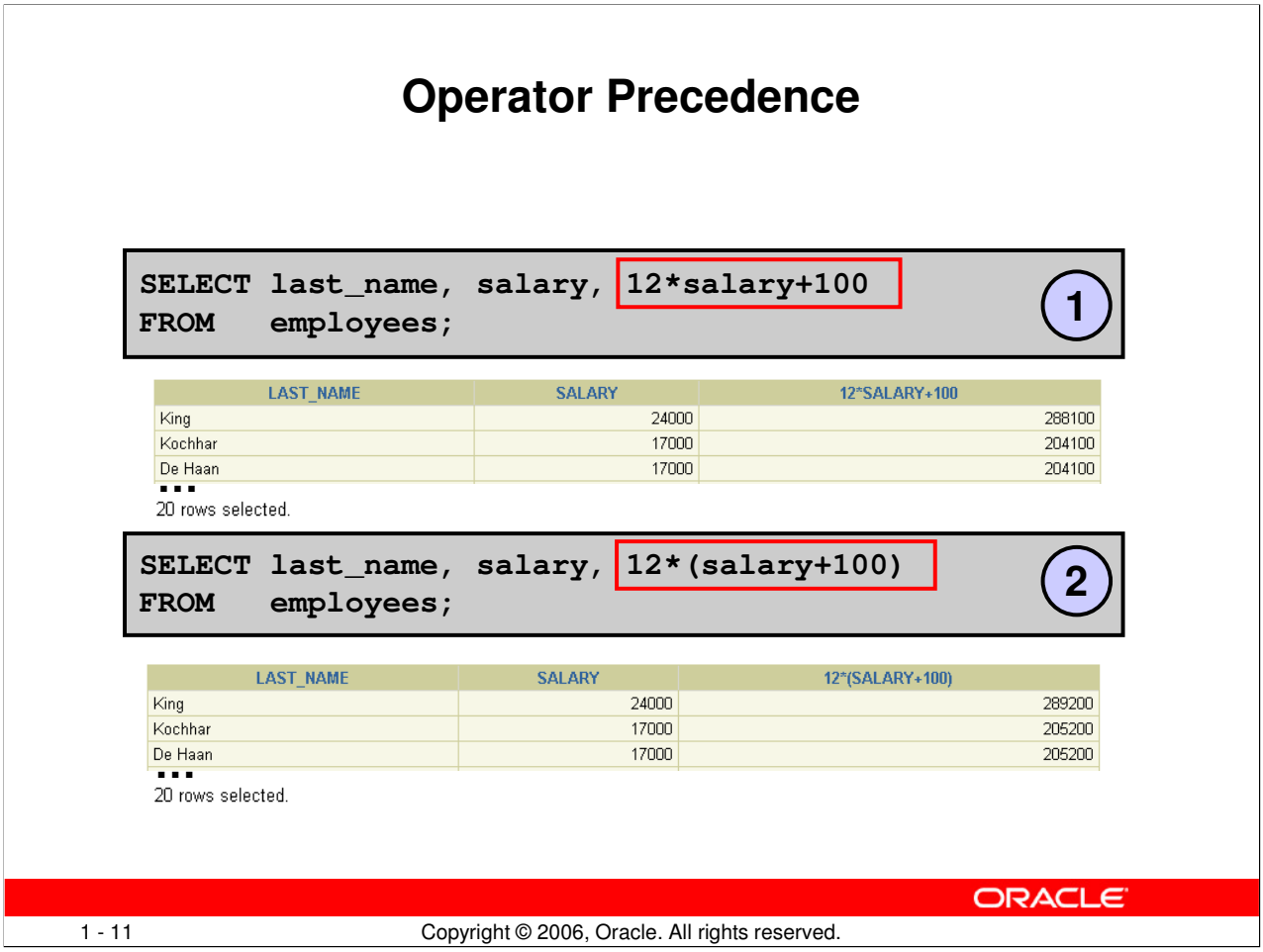
Operator Precedence

If an arithmetic expression contains more than one operator, multiplication and division are evaluated first. If operators in an expression are of the same priority, then evaluation is done from left to right.

You can use parentheses to force the expression that is enclosed by parentheses to be evaluated first.

Rules of Precedence:

- Multiplication and division occur before addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to override the default precedence or to clarify the statement.



Operator Precedence (continued)

The first example in the slide displays the last name, salary, and annual compensation of employees. It calculates the annual compensation by multiplying the monthly salary by 12, plus a one-time bonus of $100. Note that multiplication is performed before addition.

Note: Use parentheses to reinforce the standard order of precedence and to improve clarity. For example, the expression in the slide can be written as `(12*salary)+100` with no change in the result.

Using Parentheses

You can override the rules of precedence by using parentheses to specify the desired order in which operators are to be executed.

The second example in the slide displays the last name, salary, and annual compensation of employees. It calculates the annual compensation as follows: adding a monthly bonus of $100 to the monthly salary, and then multiplying that subtotal by 12. Because of the parentheses, addition takes priority over multiplication.

Defining a Null Value

- **A null is a value that is unavailable, unassigned, unknown, or inapplicable.**
- **A null is not the same as a zero or a blank space.**

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

ORACLE

Null Values

If a row lacks a data value for a particular column, that value is said to be *null* or to contain a null.

A null is a value that is unavailable, unassigned, unknown, or inapplicable. A null is not the same as a zero or a space. Zero is a number, and a space is a character.

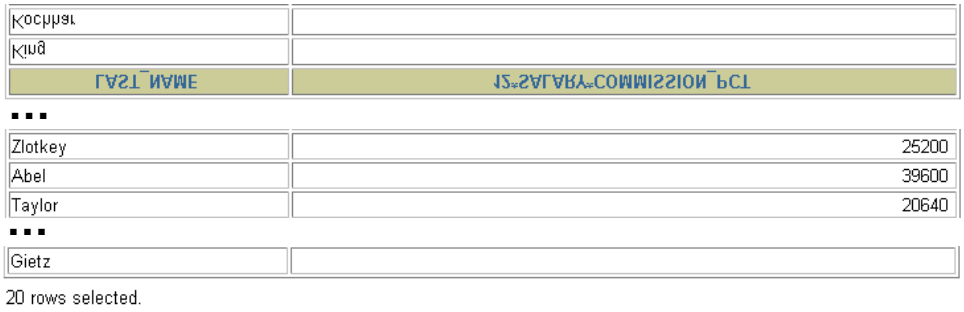
Columns of any data type can contain nulls. However, some constraints (`NOT NULL` and `PRIMARY KEY`) prevent nulls from being used in the column.

In the `COMMISSION_PCT` column in the `EMPLOYEES` table, notice that only a sales manager or sales representative can earn a commission. Other employees are not entitled to earn commissions. A null represents that fact.

Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```



LAST_NAME	12*SALARY*COMMISSION_PCT
King	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

Null Values in Arithmetic Expressions

If any column value in an arithmetic expression is null, the result is null. For example, if you attempt to perform division by zero, you get an error. However, if you divide a number by null, the result is a null or unknown.

In the example in the slide, employee King does not get any commission. Because the `COMMISSION_PCT` column in the arithmetic expression is null, the result is null.

For more information, see "Basic Elements of SQL" in *SQL Reference*.

Defining a Column Alias

A column alias:

- **Renames a column heading**
- **Is useful with calculations**
- **Immediately follows the column name (There can also be the optional `AS` keyword between the column name and alias.)**
- **Requires double quotation marks if it contains spaces or special characters or if it is case sensitive**

ORACLE

Column Aliases

When displaying the result of a query, *i*SQL*Plus normally uses the name of the selected column as the column heading. This heading may not be descriptive and, therefore, maybe difficult to understand. You can change a column heading by using a column alias.

Specify the alias after the column in the `SELECT` list using a space as a separator. By default, alias headings appear in uppercase. If the alias contains spaces or special characters (such as # or $), or if it is case sensitive, enclose the alias in double quotation marks (" ").

Using Column Aliases

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

```
SELECT last_name "Name" , salary*12 "Annual Salary"
FROM   employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

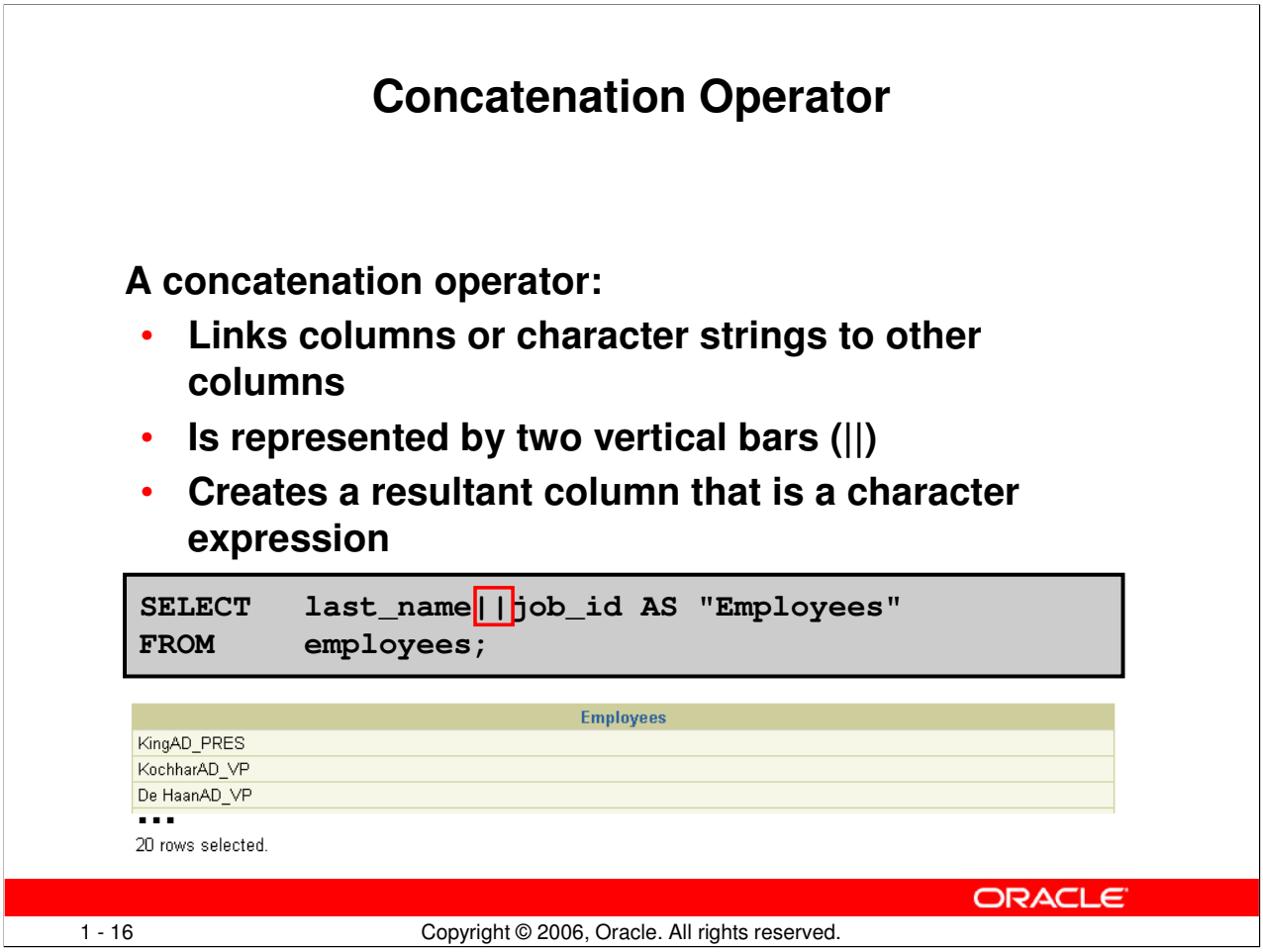
20 rows selected.

ORACLE

Column Aliases (continued)

The first example displays the names and the commission percentages of all the employees. Notice that the optional `AS` keyword has been used before the column alias name. The result of the query is the same whether the `AS` keyword is used or not. Also notice that the SQL statement has the column aliases, `name` and `comm`, in lowercase, whereas the result of the query displays the column headings in uppercase. As mentioned in a previous slide, column headings appear in uppercase by default.

The second example displays the last names and annual salaries of all the employees. Because `Annual Salary` contains a space, it has been enclosed in double quotation marks. Notice that the column heading in the output is exactly the same as the column alias.



Concatenation Operator

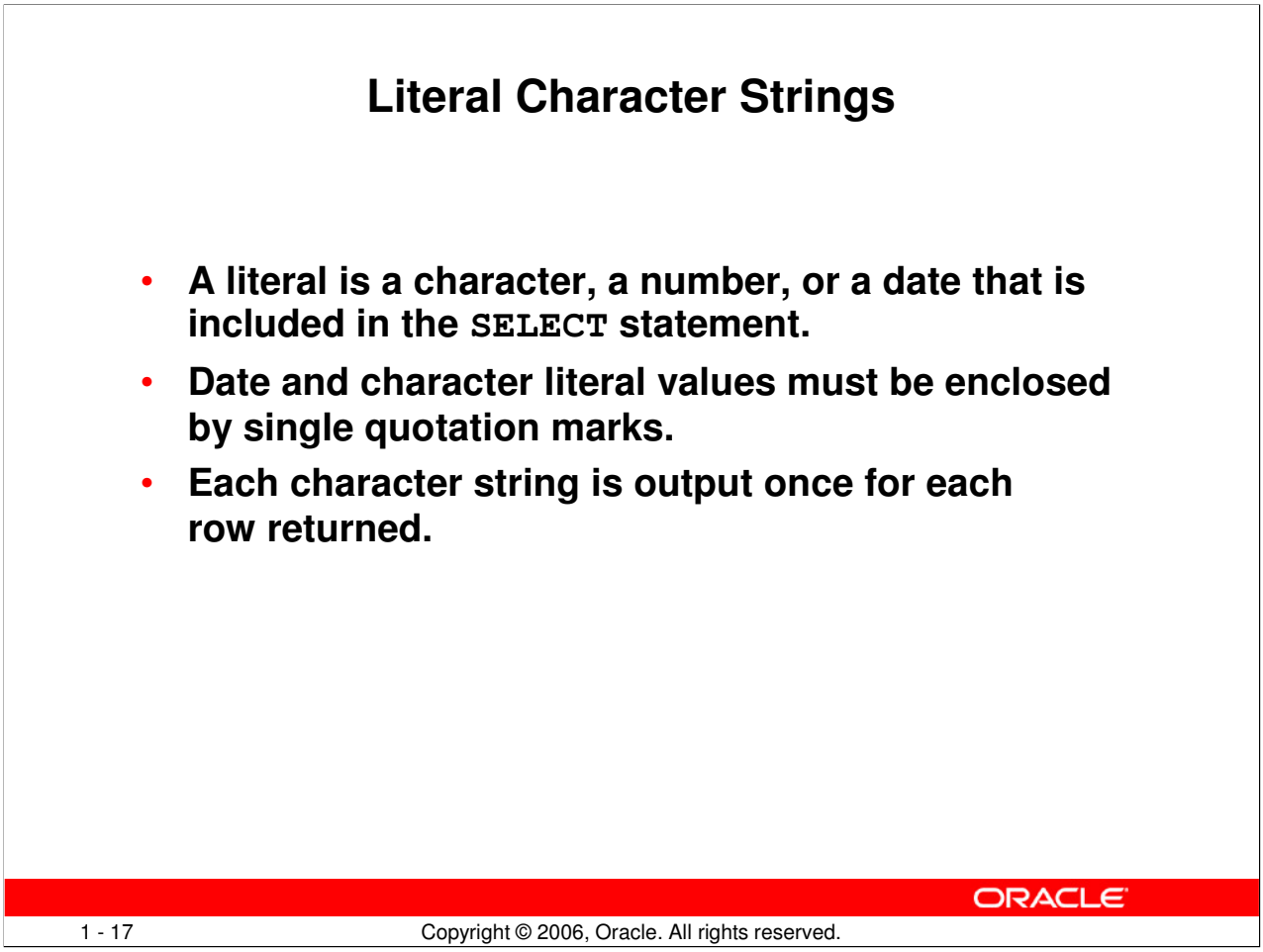
You can link columns to other columns, arithmetic expressions, or constant values to create a character expression by using the *concatenation operator* (||). Columns on either side of the operator are combined to make a single output column.

In the example, `LAST_NAME` and `JOB_ID` are concatenated, and they are given the alias `Employees`. Notice that the employee last name and job code are combined to make a single output column.

The `AS` keyword before the alias name makes the `SELECT` clause easier to read.

Null Values with the Concatenation Operator

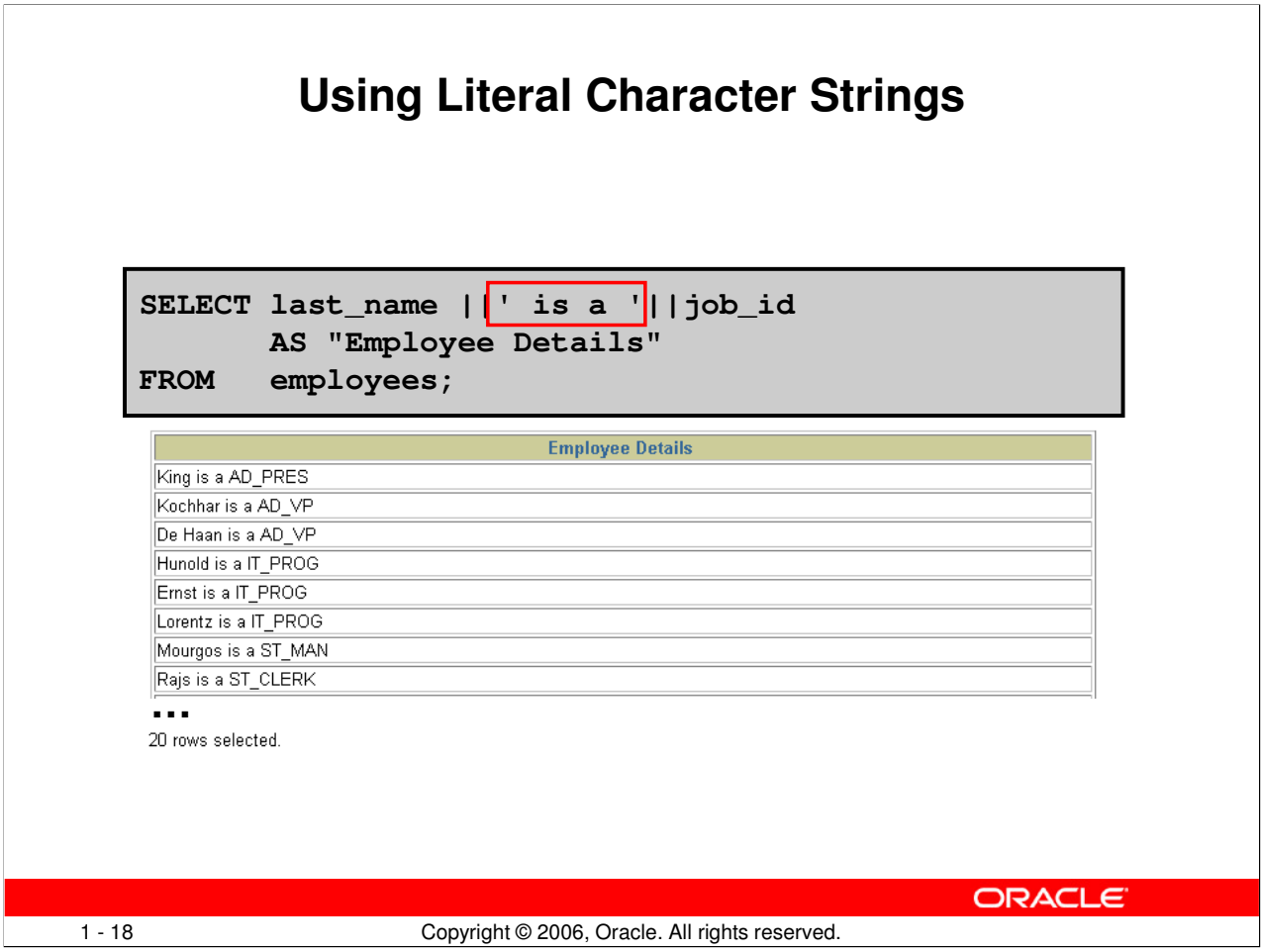
If you concatenate a null value with a character string, the result is a character string. `LAST_NAME || NULL` results in `LAST_NAME`.



Literal Character Strings

A literal is a character, a number, or a date that is included in the `SELECT` list and that is not a column name or a column alias. It is printed for each row returned. Literal strings of free-format text can be included in the query result and are treated the same as a column in the `SELECT` list.

Date and character literals *must* be enclosed by single quotation marks (`' '`); number literals need not be so enclosed.



Literal Character Strings (continued)

The example in the slide displays last names and job codes of all employees. The column has the heading Employee Details. Notice the spaces between the single quotation marks in the `SELECT` statement. The spaces improve the readability of the output.

In the following example, the last name and salary for each employee are concatenated with a literal to give the returned rows more meaning:

```
SELECT last_name ||': 1 Month salary = '||salary Monthly
FROM   employees;
```

MONTHLY
King: 1 Month salary = 24000
Kochhar: 1 Month salary = 17000
De Haan: 1 Month salary = 17000
Hunold: 1 Month salary = 9000
Ernst: 1 Month salary = 6000
Lorentz: 1 Month salary = 4200
Mourgos: 1 Month salary = 5800
Rajs: 1 Month salary = 3500

...
20 rows selected.

Alternative Quote (q) Operator

- Specify your own quotation mark delimiter
- Choose any delimiter
- Increase readability and usability

```
SELECT department_name ||
       q'[, it's assigned Manager Id: ]'
       || manager_id
       AS "Department and Manager"
FROM departments;
```

Department and Manager
Administration, it's assigned manager ID: 200
Marketing, it's assigned manager ID: 201
Shipping, it's assigned manager ID: 124
...

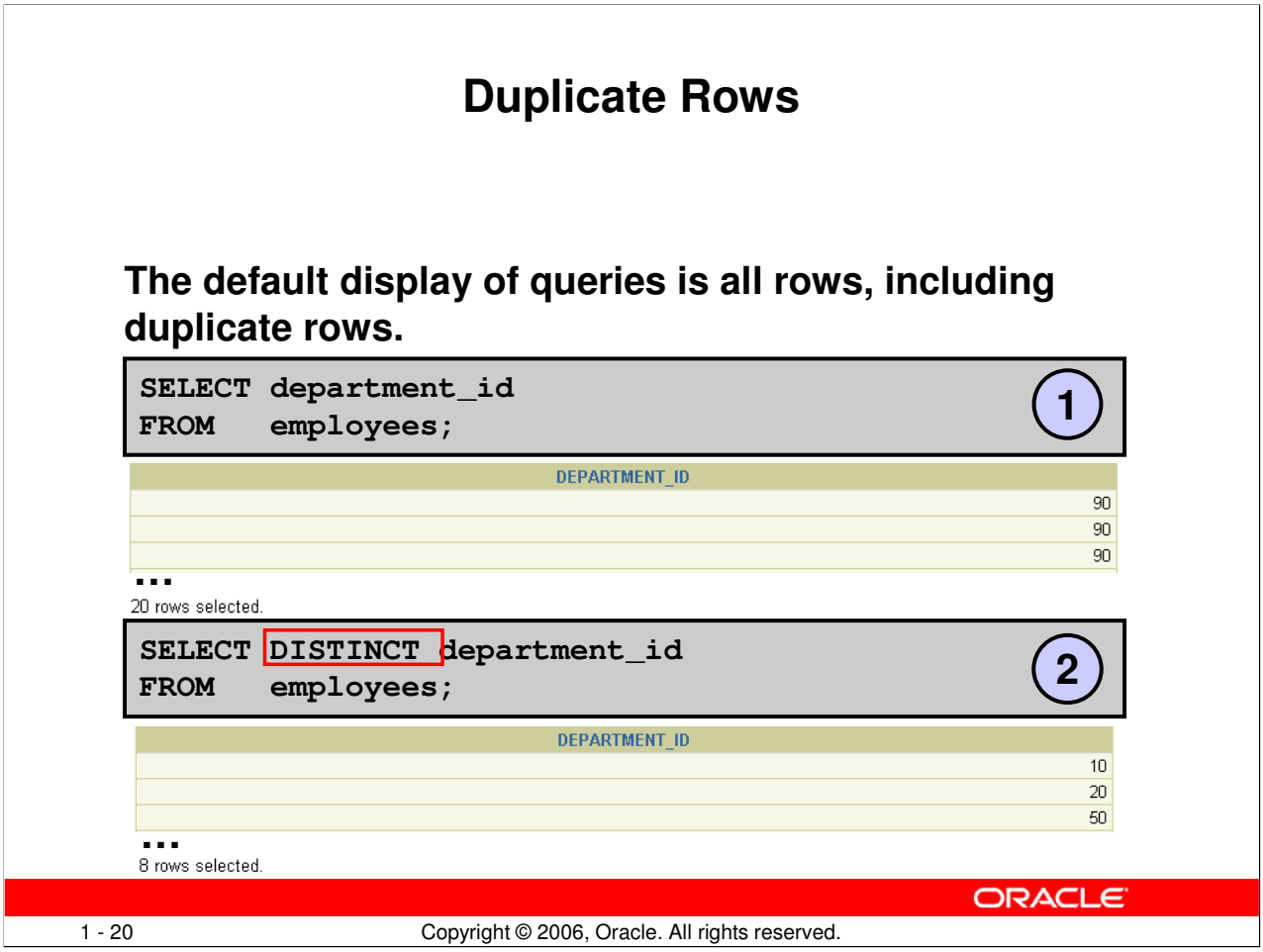
8 rows selected.

Alternative Quote (q) Operator

Many SQL statements use character literals in expressions or conditions. If the literal itself contains a single quotation mark, you can use the quote (q) operator and choose your own quotation mark delimiter.

You can choose any convenient delimiter, single-byte or multibyte, or any of the following character pairs: [ ], { }, ( ), or < >.

In the example shown, the string contains a single quotation mark, which is normally interpreted as a delimiter of a character string. By using the q operator, however, the brackets [] are used as the quotation mark delimiter. The string between the brackets delimiters is interpreted as a literal character string.



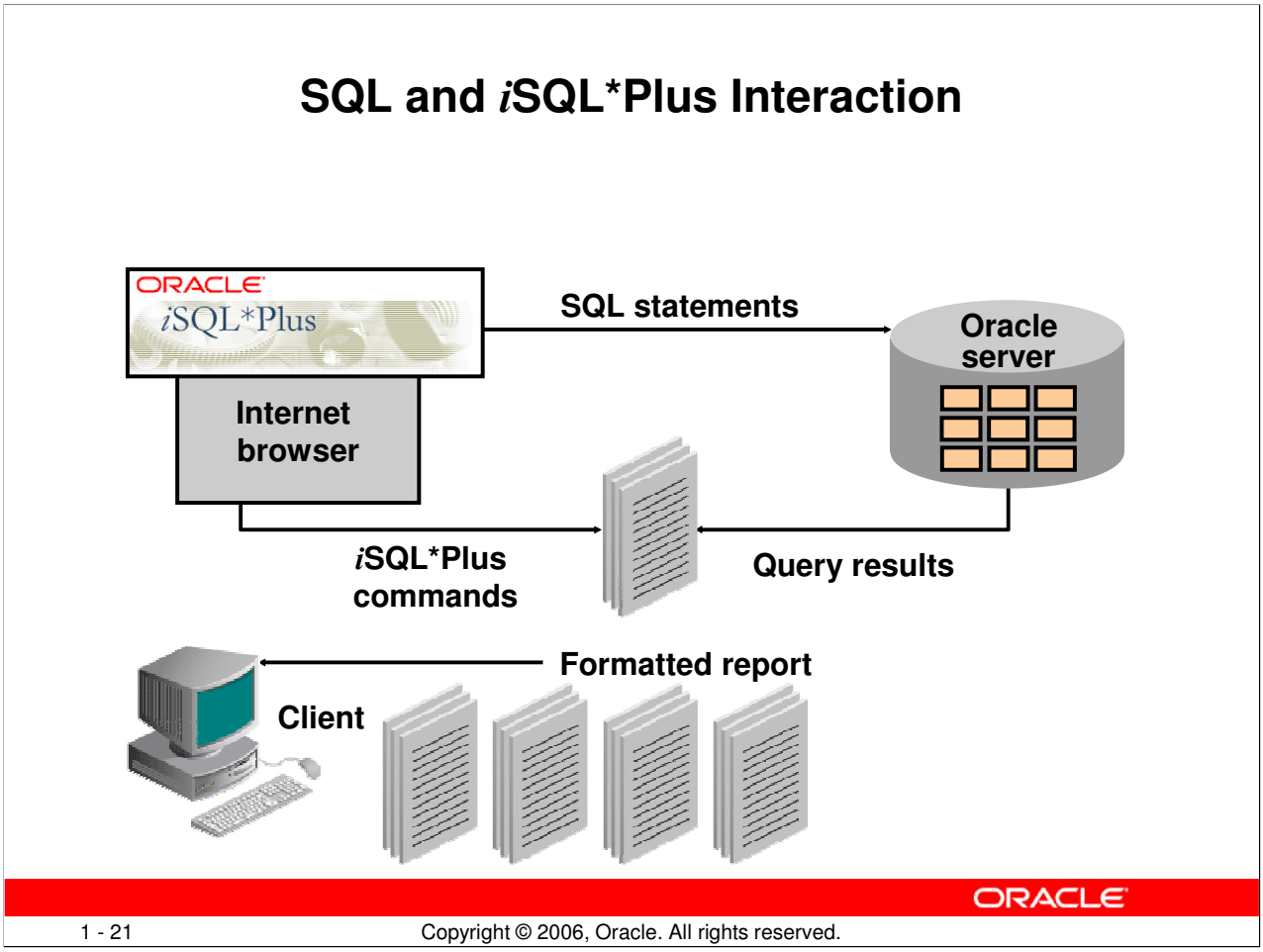
Duplicate Rows

Unless you indicate otherwise, *i*SQL*Plus displays the results of a query without eliminating duplicate rows. The first example in the slide displays all the department numbers from the EMPLOYEES table. Notice that the department numbers are repeated.

To eliminate duplicate rows in the result, include the DISTINCT keyword in the SELECT clause immediately after the SELECT keyword. In the second example in the slide, the EMPLOYEES table actually contains 20 rows, but there are only seven unique department numbers in the table.

You can specify multiple columns after the DISTINCT qualifier. The DISTINCT qualifier affects all the selected columns, and the result is every distinct combination of the columns.

```
SELECT  DISTINCT department_id, job_id
FROM    employees;
```



SQL and *i*SQL*Plus

SQL is a command language for communication with the Oracle server from any tool or application. Oracle SQL contains many extensions.

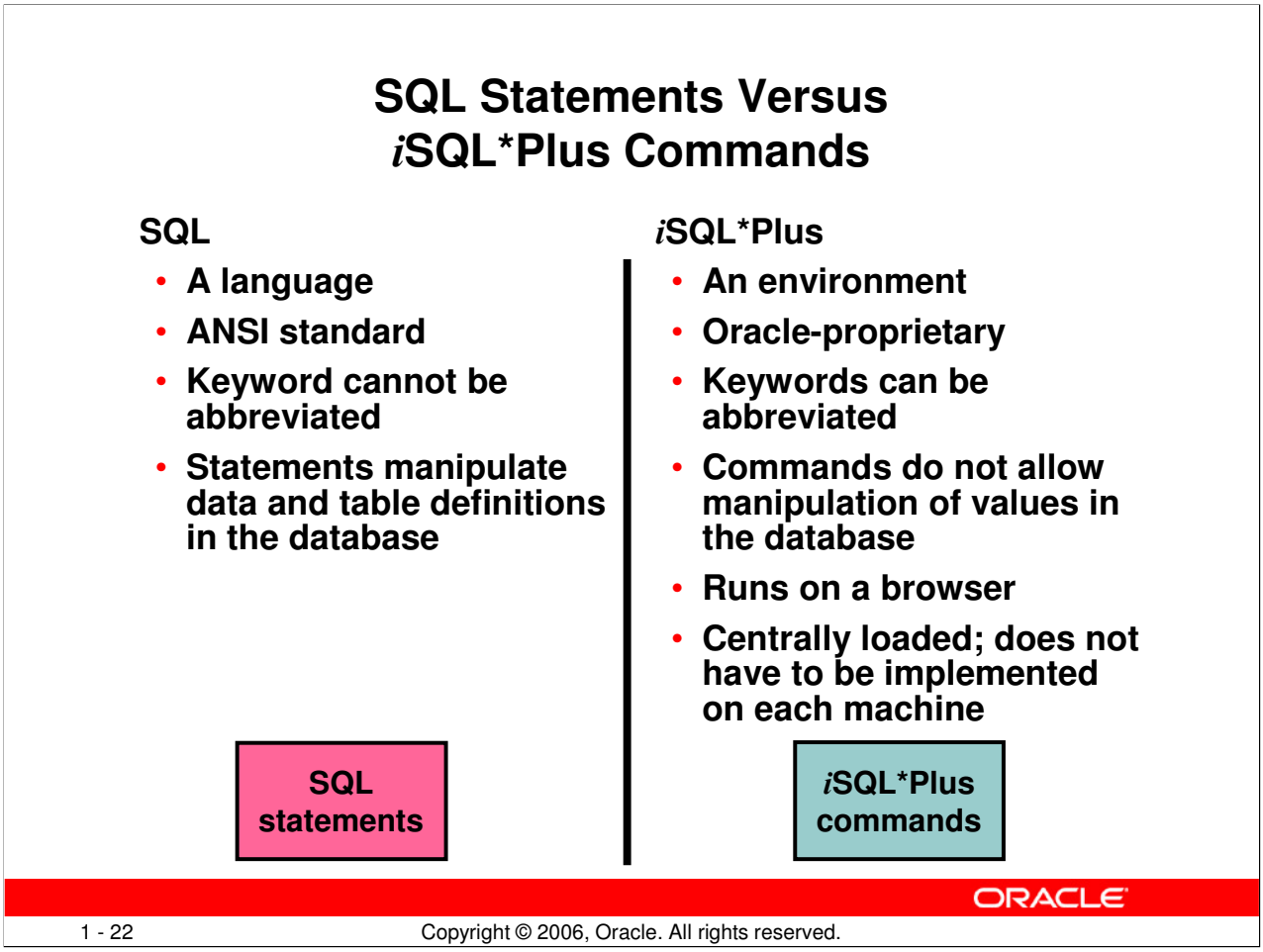
*iSQL*Plus* is an Oracle tool that recognizes and submits SQL statements to the Oracle server for execution and contains its own command language.

Features of SQL

- Can be used by a range of users, including those with little or no programming experience
- Is a nonprocedural language
- Is an English-like language

Features of *i*SQL*Plus

- Is accessed from a browser
- Accepts SQL statements
- Provides online editing for modifying SQL statements
- Controls environmental settings
- Formats query results into a basic report
- Accesses local and remote databases



SQL and *i*SQL*Plus (continued)

The following table compares SQL and *i*SQL*Plus:

SQL	*i*SQL*Plus
Is a language for communicating with the Oracle server to access data	Recognizes SQL statements and sends them to the server
Is based on American National Standards Institute (ANSI)–standard SQL	Is the Oracle-proprietary interface for executing SQL statements
Retrieves data; manipulates data and table definitions in the database	Does not allow manipulation of values in the database
Does not have a continuation character	Has a dash (–) as a continuation character if the command is longer than one line
Cannot be abbreviated	Can be abbreviated
Uses functions to perform some formatting	Uses commands to format data

Overview of *i*SQL*Plus

After you log in to *i*SQL*Plus, you can:

- **Describe table structures**
- **Enter, execute, and edit SQL statements**
- **Save or append SQL statements to files**
- **Execute or edit statements that are stored in saved script files**

ORACLE

*i*SQL*Plus

*i*SQL*Plus is an environment in which you can do the following:

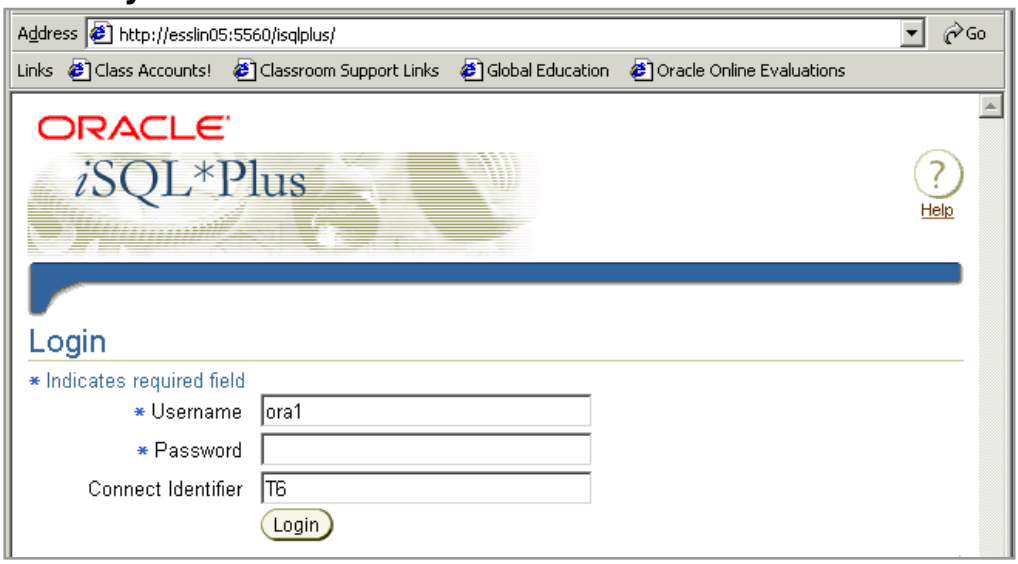
- Execute SQL statements to retrieve, modify, add, and remove data from the database
- Format, perform calculations on, store, and print query results in the form of reports
- Create script files to store SQL statements for repeated use in the future

*i*SQL*Plus commands can be divided into the following main categories:

Category	Purpose
Environment	Affects the general behavior of SQL statements for the session
Format	Formats query results
File manipulation	Saves statements in text script files and runs statements from text script files
Execution	Sends SQL statements from the browser to the Oracle server
Edit	Modifies SQL statements in the Edit window
Interaction	Enables you to create and pass variables to SQL statements, print variable values, and print messages to the screen
Miscellaneous	Has various commands to connect to the database, manipulate the *i*SQL*Plus environment, and display column definitions

Logging In to *i*SQL*Plus

From your browser environment:

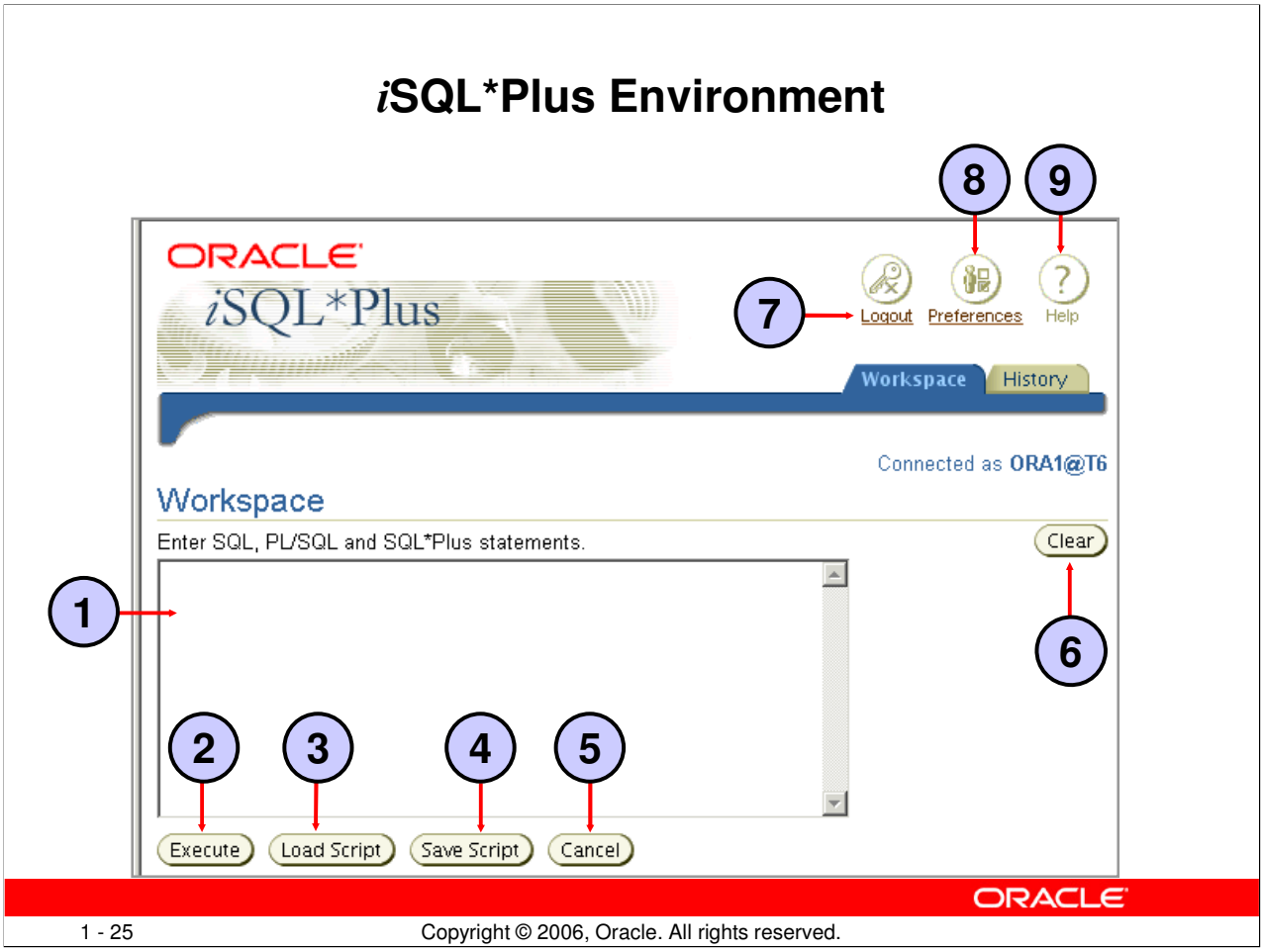


ORACLE

Logging In to *i*SQL*Plus

To log in from a browser environment:

1. Start the browser.
2. Enter the URL address of the *i*SQL*Plus environment.
3. On the Login page, enter appropriate values in the Username, Password, and Connect Identifier fields.



*i*SQL*Plus Environment

In the browser, the *i*SQL*Plus Workspace page has several key areas:

1. **Text box:** Area where you type the SQL statements and *i*SQL*Plus commands
2. **Execute button:** Click to execute the statements and commands in the text box
3. **Load Script button:** Brings up a form where you can identify a path and file name or a URL that contains SQL, PL/SQL, or SQL*Plus commands and load them into the text box
4. **Save Script button:** Saves the contents of the text box to a file
5. **Cancel button:** Stops the execution of the command in the text box
6. **Clear Screen button:** Click to clear text from the text box
7. **Logout icon:** Click to end the *i*SQL*Plus session and return to the *i*SQL*Plus Login page
8. **Preferences icon:** Click to change your interface configuration, system configuration, or password
9. **Help icon:** Provides access to *i*SQL*Plus help documentation

Displaying Table Structure

Use the *i*SQL*Plus `DESCRIBE` command to display the structure of a table:

```
DESC[RIBE] tablename
```

Displaying the Table Structure

In *i*SQL*Plus, you can display the structure of a table by using the `DESCRIBE` command. The command displays the column names and data types, and it shows you whether a column *must* contain data (that is, whether the column has a `NOT NULL` constraint).

In the syntax, *`tablename`* is the name of any existing table, view, or synonym that is accessible to the user.

Displaying Table Structure

```
DESCRIBE employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

ORACLE

Displaying the Table Structure (continued)

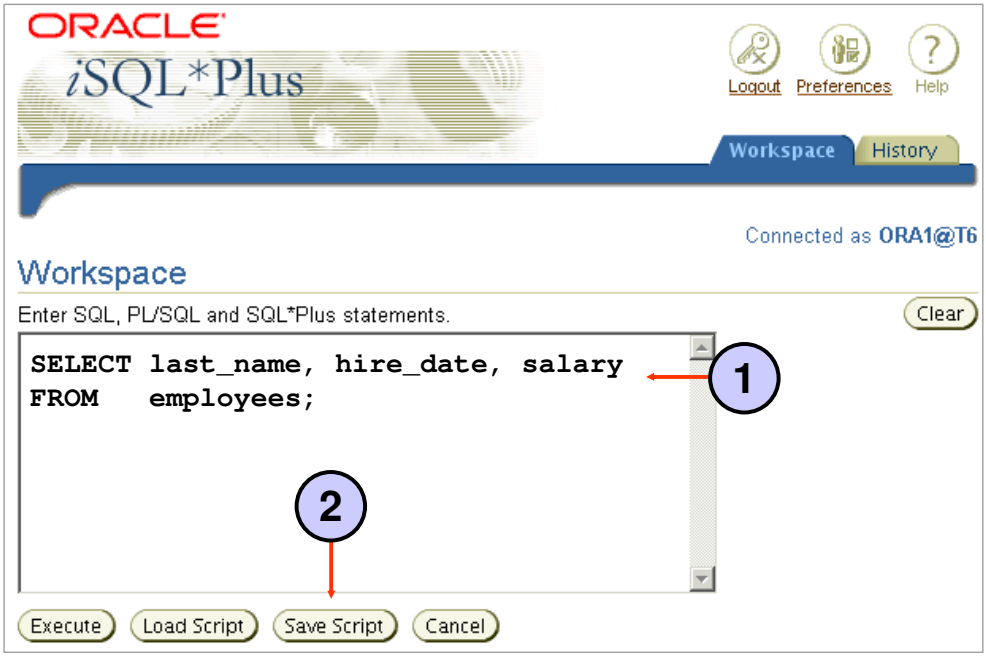
The example in the slide displays the information about the structure of the `EMPLOYEES` table.

In the resulting display, *Null?* indicates that the values for this column maybe unknown. `NOT NULL` indicates that a column must contain data. *Type* displays the data type for a column.

The data types are described in the following table:

Data Type	Description
`NUMBER(p,s)`	Number value having a maximum number of digits *p*, with *s* digits to the right of the decimal point
`VARCHAR2(s)`	Variable-length character value of maximum size *s*
`DATE`	Date and time value between January 1, 4712 B.C., and December 31, A.D 9999.
`CHAR(s)`	Fixed-length character value of size *s*

Interacting with Script Files



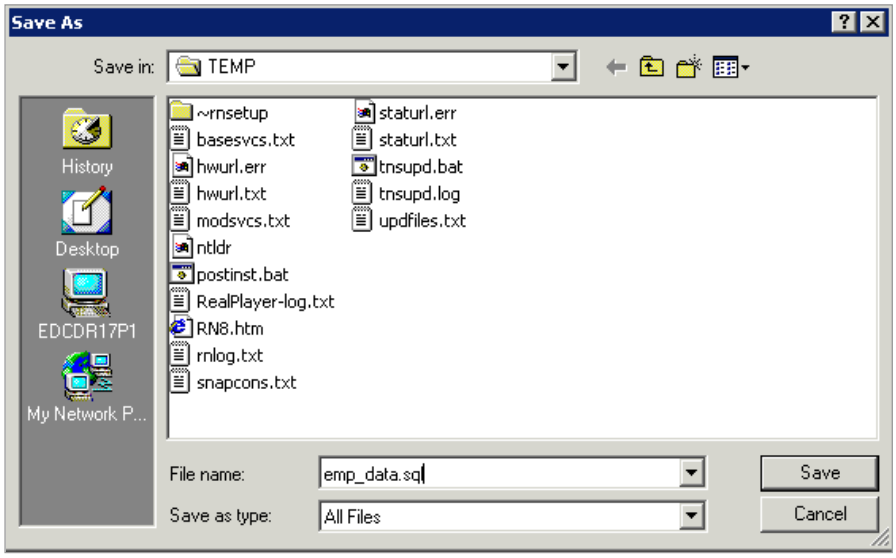
Interacting with Script Files

Placing Statements and Commands into a Text Script File

You can save commands and statements from the text box in *i*SQL*Plus to a text script file as follows:

1. Type the SQL statements in the text box in *i*SQL*Plus.
2. Click the Save Script button. This opens the Windows File Save dialog box. Identify the name of the file. Note that the file extension defaults to `.sql`. You can change the file type to a text file or save it as a `.sql` file.

Interacting with Script Files

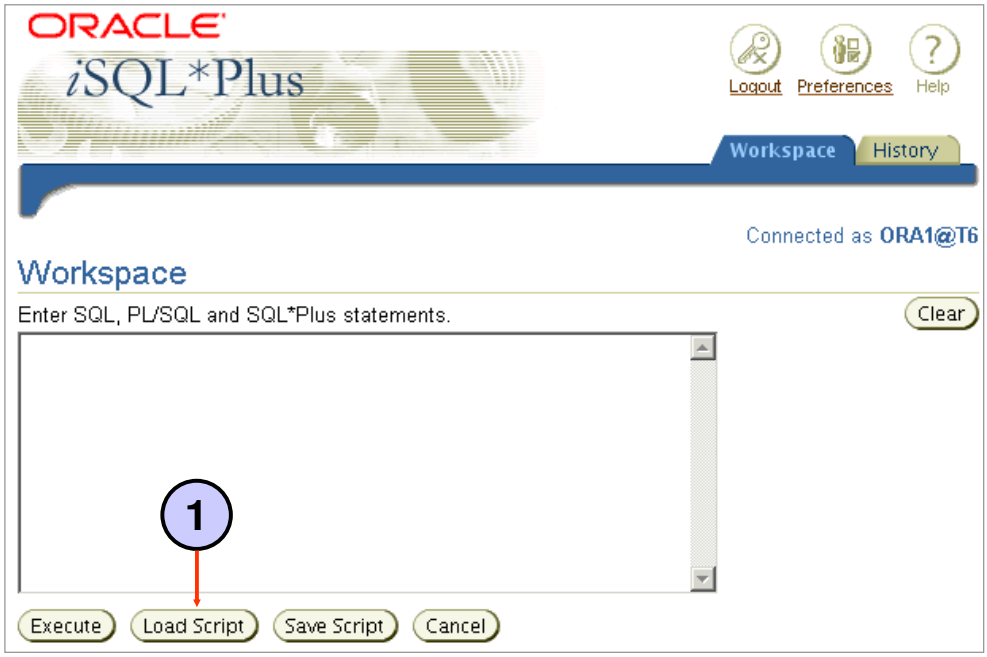


ORACLE

Interacting with Script Files (continued)

In the example shown, the SQL `SELECT` statement typed in the text box is saved to a file named `emp_data.sql`. You can choose the type of the file, name of the file, and location of where you want to save the script file.

Interacting with Script Files



Interacting with Script Files (continued)

Using Statements and Commands from a Script File in *i*SQL*Plus

You can use previously saved commands and statements from a script file in *i*SQL*Plus as follows:

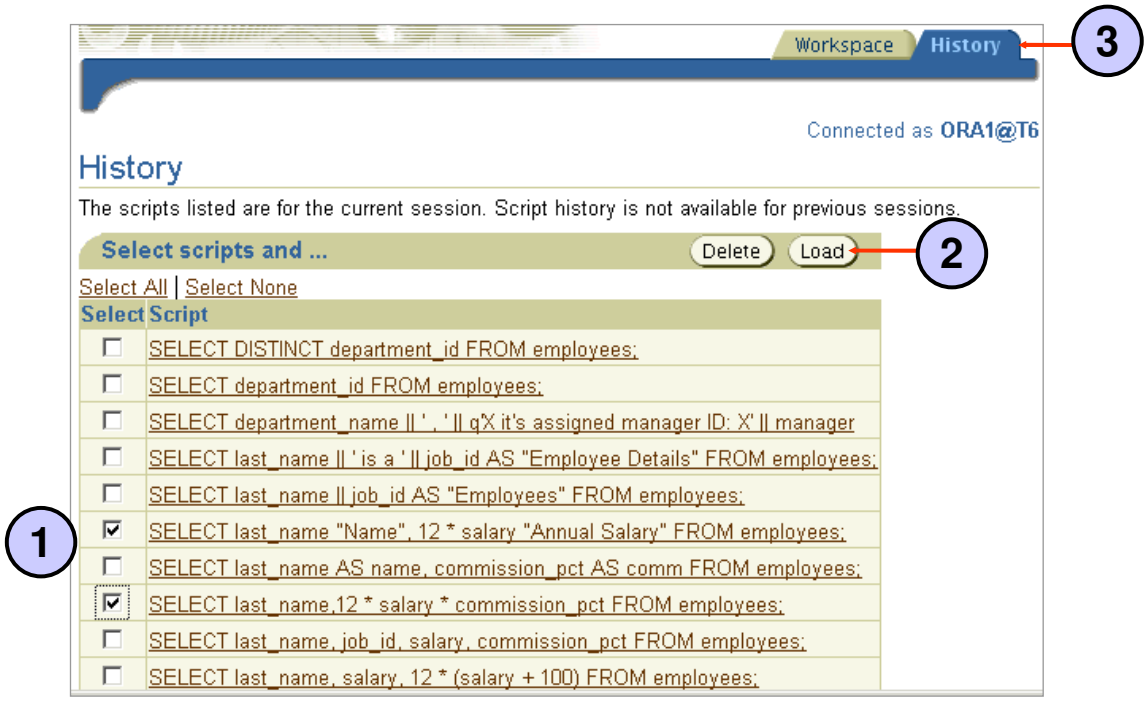
1. Click the Load Script button. This opens a form where you can enter the name of the file or a URL containing the SQL, PL/SQL, or SQL*Plus commands that you want to enter in the text box.

Interacting with Script Files

Interacting with Script Files (continued)

2. Enter the script name and path, or the URL location. Or you can click the Browse button to find the script name and location.
3. Click the Load button to bring the contents of the file or URL location into the text box.

*i*SQL*Plus History Page



1 - 32

Running Previous Statements

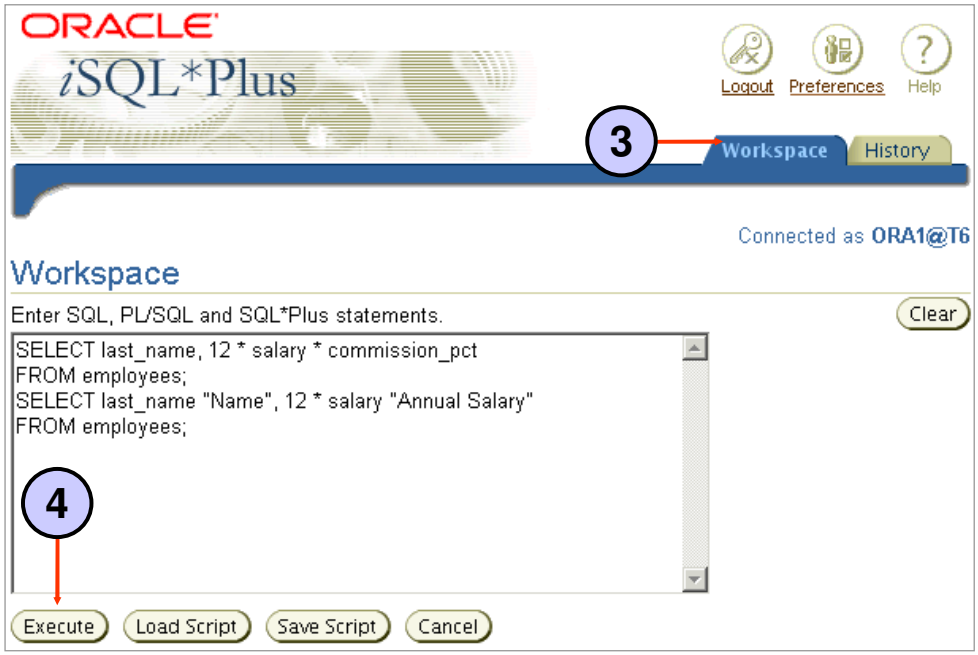
The History page in *i*SQL*Plus lets you execute previously run statements in your session. The History page shows your most recently run SQL statements and *i*SQL*Plus commands. To rerun the statements:

1. Select the statement that you want to execute.
2. Click the Load button.

Note

- You can control the number of statements that are shown on the History page with Preferences settings.
- You can choose to delete selected statements by clicking the Delete button.

*i*SQL*Plus History Page

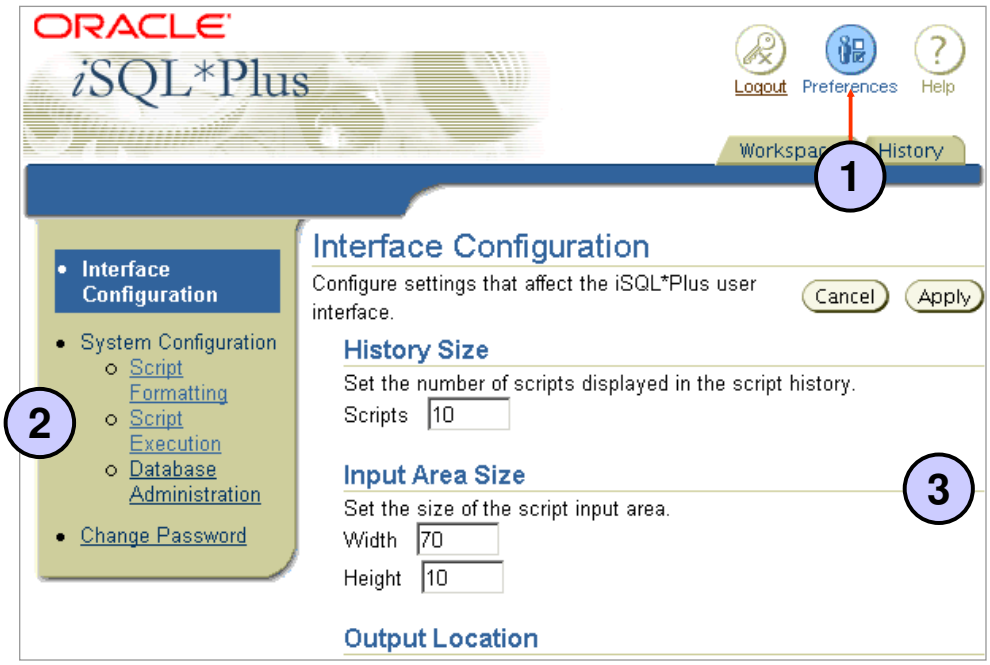


ORACLE

Running Previous Statements (continued)

3. Return to the Workspace page.
4. Click the Execute button to run the commands that have been loaded into the text box.

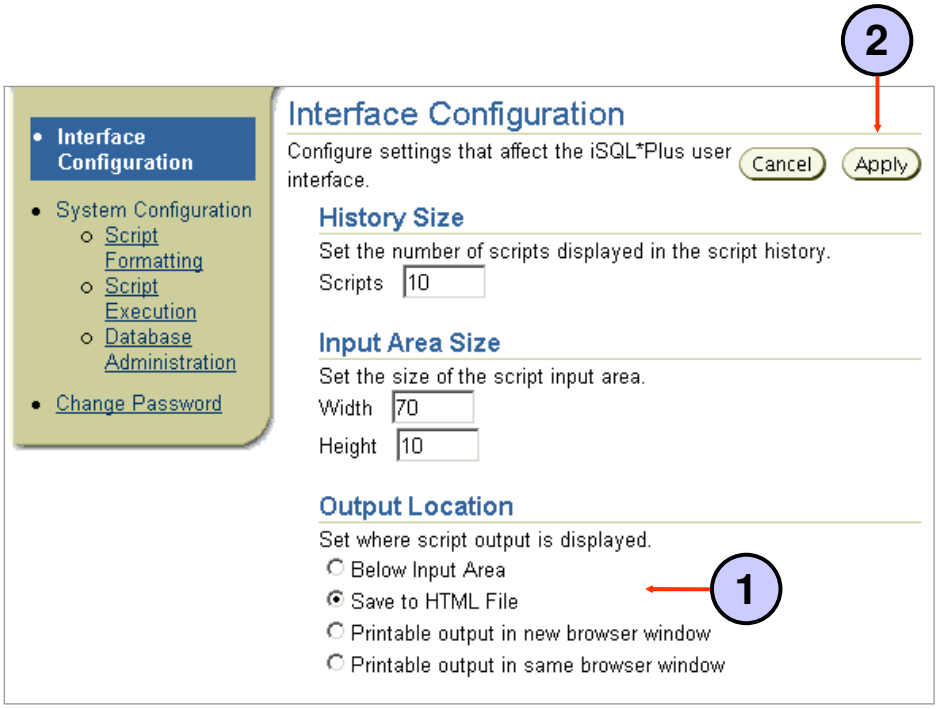
Setting *i*SQL*Plus Preferences



*i*SQL*Plus Preferences

1. You can set preferences for your *i*SQL*Plus session by clicking the Preferences icon.
2. The preferences are divided into categories. You can set preferences for script formatting, script execution, and database administration, and you can change your password.
3. When you choose a preference category, a form is displayed that lets you set the preferences for that category.

Setting the Output Location Preference



ORACLE

Changing the Output Location

You can send the results that are generated by a SQL statement or *i*SQL*Plus command to the screen (the default), a file, or another browser window.

On the Preferences page:

1. Select an Output Location option.
2. Click the Apply button.

Summary

In this lesson, you should have learned how to:

- **Write a `SELECT` statement that:**
  - **Returns all rows and columns from a table**
  - **Returns specified columns from a table**
  - **Uses column aliases to display more descriptive column headings**
- **Use the *i*SQL*Plus environment to write, save, and execute SQL statements and *i*SQL*Plus commands**

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM table;
```

ORACLE

`SELECT` Statement

In this lesson, you should have learned how to retrieve data from a database table with the `SELECT` statement.

```
SELECT  *|{[DISTINCT] column [alias],...}
FROM    table;
```

In the syntax:

`SELECT`	is a list of one or more columns
`*`	selects all columns
`DISTINCT`	suppresses duplicates
`column|expression`	selects the named column or the expression
`alias`	gives selected columns different headings
`FROM table`	specifies the table containing the columns

*i*SQL*Plus

*i*SQL*Plus is an execution environment that you can use to send SQL statements to the database server and to edit and save SQL statements. Statements can be executed from the SQL prompt or from a script file.

Practice 1: Overview

This practice covers the following topics:

- **Selecting all data from different tables**
- **Describing the structure of tables**
- **Performing arithmetic calculations and specifying column names**
- **Using *i*SQL*Plus**

Practice 1: Overview

This is the first of many practices in this course. The solutions (if you require them) can be found in Appendix A. Practices are intended to cover all topics that are presented in the corresponding lesson.

Note the following location for the lab files:

`E:\labs\SQL1\labs`

If you are asked to save any lab files, save them at this location.

To start *i*SQL*Plus, start your browser. You need to enter a URL to access *i*SQL*Plus. The URL requires the host name, which your instructor will provide. Enter the following command, replacing the host name with the value that your instructor provides:

`http://<HOSTNAME:5560>/isqlplus`

In any practice, there maybe exercises that are prefaced with the phrases “If you have time” or “If you want an extra challenge.” Work on these exercises only if you have completed all other exercises in the allocated time and would like a further challenge to your skills.

Perform the practices slowly and precisely. You can experiment with saving and running command files. If you have any questions at any time, ask your instructor.

Practice 1

Part 1

Test your knowledge:

1. Initiate an *i*SQL*Plus session using the user ID and password that are provided by the instructor.
2. *i*SQL*Plus commands access the database.

   True/False
3. The following `SELECT` statement executes successfully:

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

   True/False
4. The following `SELECT` statement executes successfully:

```
SELECT *
FROM   job_grades;
```

   True/False
5. There are four coding errors in the following statement. Can you identify them?

```
SELECT    employee_id, last_name
sal x 12  ANNUAL SALARY
FROM      employees;
```

Part 2

Note the following location for the lab files:

`E:\labs\SQL1\labs`

If you are asked to save any lab files, save them at this location.

*To start iSQL*Plus, start your browser. You need to enter a URL to access iSQL*Plus. The URL requires the host name, which your instructor will provide. Enter the following command, replacing the host name with the value that your instructor provides:*

`http://<HOSTNAME:5560>/isqlplus`

You have been hired as a SQL programmer for Acme Corporation. Your first task is to create some reports based on data from the Human Resources tables.

6. Your first task is to determine the structure of the `DEPARTMENTS` table and its contents.

Name	Null?	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Practice 1 (continued)

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

7. You need to determine the structure of the `EMPLOYEES` table.

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

The HR department wants a query to display the last name, job code, hire date, and employee number for each employee, with employee number appearing first. Provide an alias `STARTDATE` for the `HIRE_DATE` column. Save your SQL statement to a file named `lab_01_07.sql` so that you can dispatch this file to the HR department.

Practice 1 (continued)

8. Test your query in the `lab_01_07.sql` file to ensure that it runs correctly.

EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
100	King	AD_PRES	17-JUN-87
101	Kochhar	AD_VP	21-SEP-89
102	De Haan	AD_VP	13-JAN-93
103	Hunold	IT_PROG	03-JAN-90
104	Ernst	IT_PROG	21-MAY-91
107	Lorentz	IT_PROG	07-FEB-99
124	Mourgos	ST_MAN	16-NOV-99
141	Rajs	ST_CLERK	17-OCT-95
142	Davies	ST_CLERK	29-JAN-97
143	Matos	ST_CLERK	15-MAR-98
144	Vargas	ST_CLERK	09-JUL-98
149	Zlotkey	SA_MAN	29-JAN-00
174	Abel	SA_REP	11-MAY-96
176	Taylor	SA_REP	24-MAR-98
▪ ▪ ▪			
206	Gietz	AC_ACCOUNT	07-JUN-94

20 rows selected.

9. The HR department needs a query to display all unique job codes from the `EMPLOYEES` table.

JOB_ID
AC_ACCOUNT
AC_MGR
AD_ASST
AD_PRES
AD_VP
IT_PROG
MK_MAN
MK_REP
SA_MAN
SA_REP
ST_CLERK
ST_MAN

12 rows selected.

Practice 1 (continued)

Part 3

If you have time, complete the following exercises:

10. The HR department wants more descriptive column headings for its report on employees. Copy the statement from `lab_01_07.sql` to the *i*SQL*Plus text box. Name the column headings `Emp #`, `Employee`, `Job`, and `Hire Date`, respectively. Then run your query again.

Emp #	Employee	Job	Hire Date
100	King	AD_PRES	17-JUN-87
101	Kochhar	AD_VP	21-SEP-89
102	De Haan	AD_VP	13-JAN-93
103	Hunold	IT_PROG	03-JAN-90
104	Ernst	IT_PROG	21-MAY-91
107	Lorentz	IT_PROG	07-FEB-99
124	Mourgos	ST_MAN	16-NOV-99
141	Rajs	ST_CLERK	17-OCT-95
142	Davies	ST_CLERK	29-JAN-97
143	Matos	ST_CLERK	15-MAR-98
144	Vargas	ST_CLERK	09-JUL-98
▪ ▪ ▪			
206	Gietz	AC_ACCOUNT	07-JUN-94

20 rows selected.

11. The HR department has requested a report of all employees and their job IDs. Display the last name concatenated with the job ID (separated by a comma and space) and name the column `Employee and Title`.

Employee and Title
King, AD_PRES
Kochhar, AD_VP
De Haan, AD_VP
Hunold, IT_PROG
Ernst, IT_PROG
Lorentz, IT_PROG
Mourgos, ST_MAN
Rajs, ST_CLERK
Davies, ST_CLERK
▪ ▪ ▪
Gietz, AC_ACCOUNT

20 rows selected.

Practice 1 (continued)

If you want an extra challenge, complete the following exercise:

12. To familiarize yourself with the data in the `EMPLOYEES` table, create a query to display all the data from that table. Separate each column output by a comma. Name the column title `THE_OUTPUT`.

THE_OUTPUT
100,Steven,King,SKING,515.123.4567,AD_PRES,,17-JUN-87,24000,,90
101,Neena,Kochhar,NKOCHHAR,515.123.4568,AD_VP,100,21-SEP-89,17000,,90
102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-93,17000,,90
103,Alexander,Hunold,AHUNOLD,590.423.4567,IT_PROG,102,03-JAN-90,9000,,60
104,Bruce,Ernst,BERNST,590.423.4568,IT_PROG,103,21-MAY-91,6000,,60
107,Diana,Lorentz,DLORENTZ,590.423.5567,IT_PROG,103,07-FEB-99,4200,,60
124,Kevin,Mourgos,KMOURGOS,650.123.5234,ST_MAN,100,16-NOV-99,5800,,50
141,Trenna,Rajs,TRAJS,650.121.8009,ST_CLERK,124,17-OCT-95,3500,,50
142,Curtis,Davies,CDAVIES,650.121.2994,ST_CLERK,124,29-JAN-97,3100,,50
143,Randall,Matos,RMATOS,650.121.2874,ST_CLERK,124,15-MAR-98,2600,,50
144,Peter,Vargas,PVARGAS,650.121.2004,ST_CLERK,124,09-JUL-98,2500,,50
▪ ▪ ▪
206,William,Gietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110

20 rows selected.