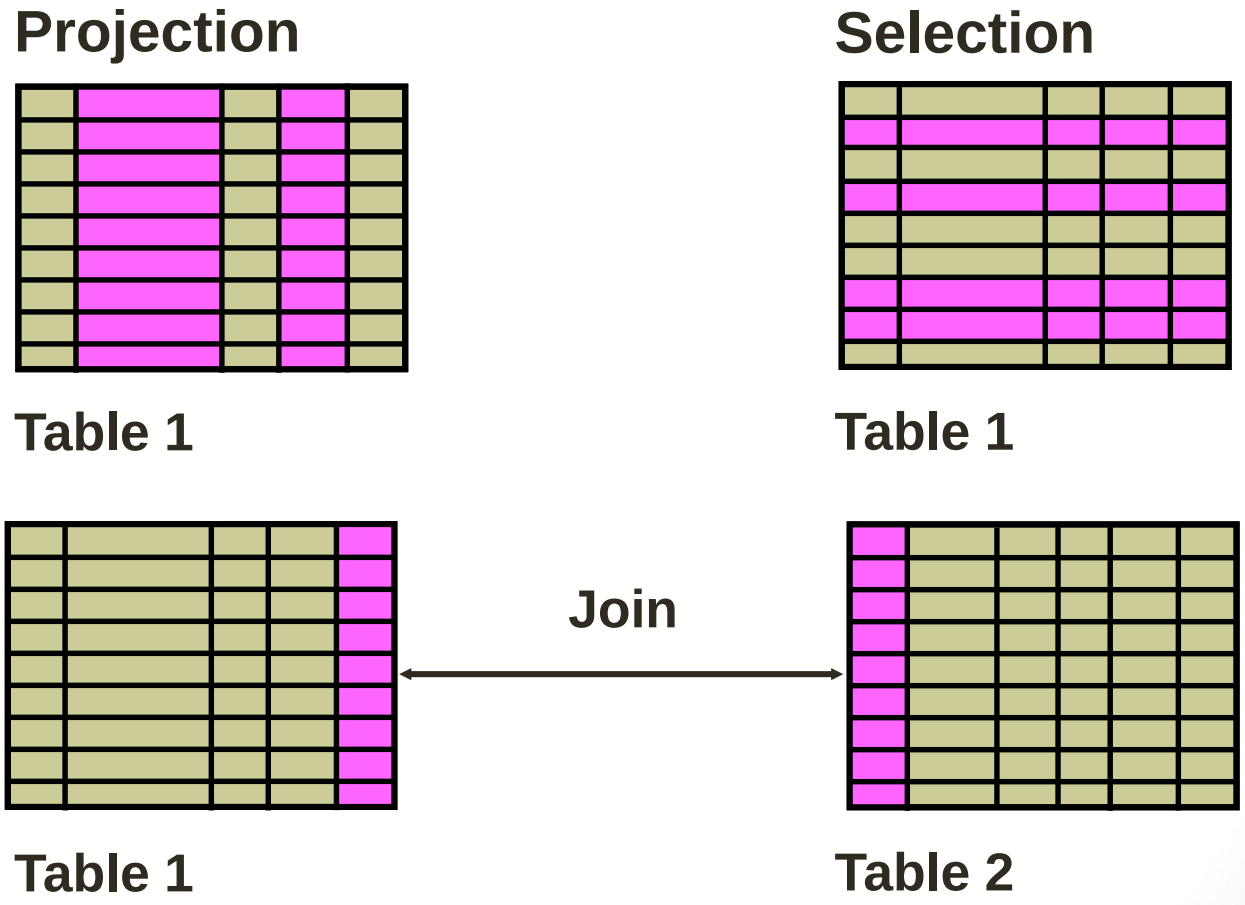
Retrieving Data Using the SQL SELECT Statement

by ABHISHEK SHARMA

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

Capabilities of SQL SELECT Statements



Basic SELECT Statement

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

- SELECT identifies the columns to be displayed
- FROM identifies the table containing those columns

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.

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.

Writing SQL Statements

- SQL statements are not case-sensitive.
- SQL statements can be on one or more lines.
- Keywords cannot be abbreviated or split across lines.
- Clauses are usually placed on separate lines.
- Indents are used to enhance readability.
- In *i*SQL*Plus, SQL statements can optionally be terminated by a semicolon (;). Semicolons are required if you execute multiple SQL statements.
- In SQL*plus, you are required to end each SQL statement with a semicolon (;).

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

Arithmetic Expressions

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

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

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.

Operator Precedence

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

1

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100

...

20 rows selected.

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

2

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200

...

20 rows selected.

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.

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
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

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

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.

Concatenation Operator

- A concatenation operator:
  - Links columns or character strings to other columns
  - Is represented by two vertical bars (||)
  - Creates a resultant column that is a character expression

```
SELECT   last_name||job_id AS "Employees"
FROM     employees;
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
...

20 rows selected.

Literal Character Strings

- A literal is a character, a number, or a date that is included in the `SELECT` statement.
- Date and character literal values must be enclosed by single quotation marks.
- Each character string is output once for each row returned.

Using Literal Character Strings

```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...

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.

Duplicate Rows

- The default display of queries is all rows, including duplicate rows.

```
SELECT department_id
FROM   employees;
```

1

DEPARTMENT_ID
90
90
90

...

20 rows selected.

```
SELECT DISTINCT department_id
FROM   employees;
```

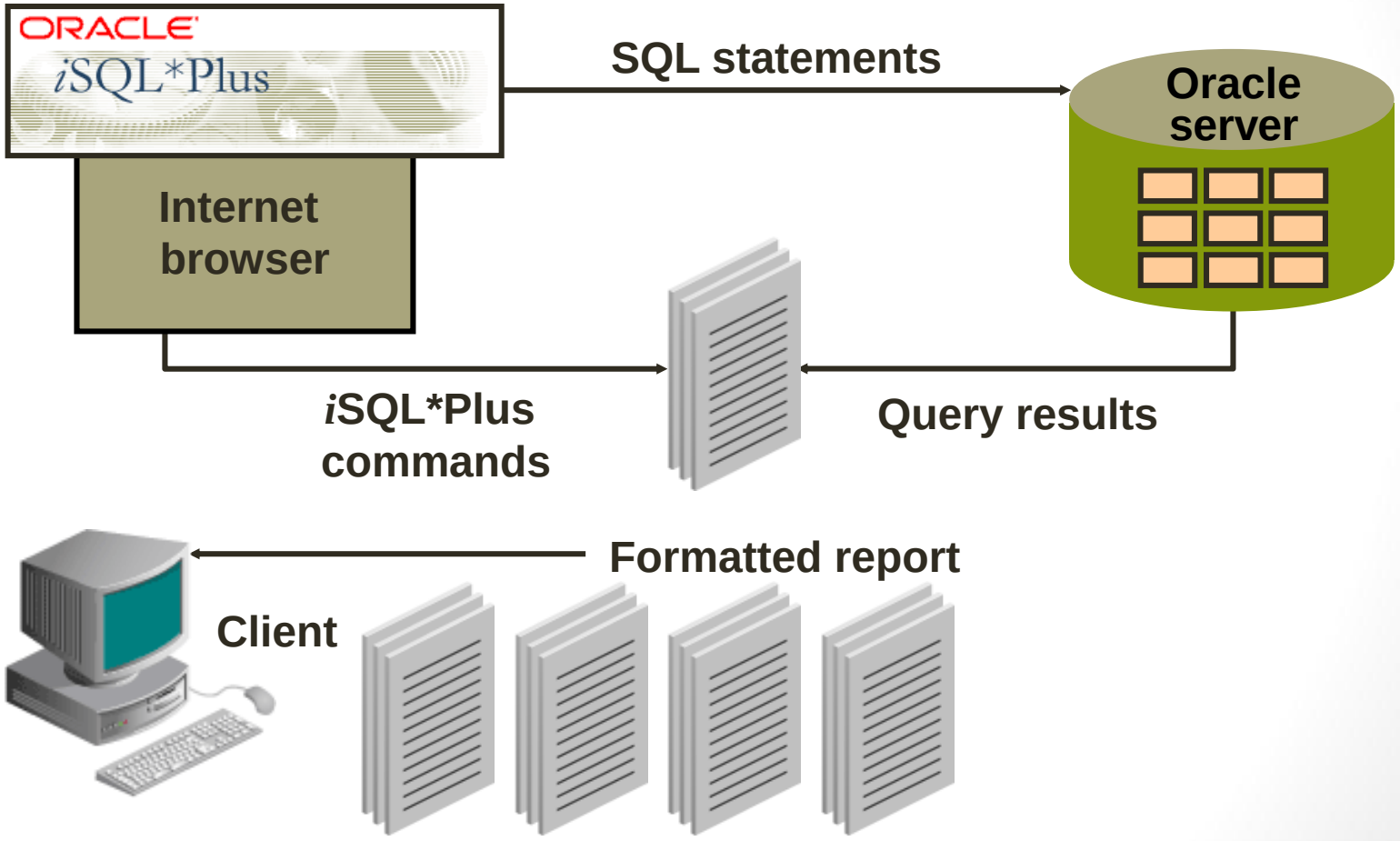
2

DEPARTMENT_ID
10
20
50

...

8 rows selected.

SQL and *i*SQL*Plus Interaction



SQL Statements Versus *i*SQL*Plus Commands

SQL

- A language
- ANSI standard
- Keyword cannot be abbreviated
- Statements manipulate data and table definitions in the database

SQL statements

***i*SQL*Plus**

- An environment
- Oracle-proprietary
- Keywords can be abbreviated
- Commands do not allow manipulation of values in the database
- Runs on a browser
- Centrally loaded; does not have to be implemented on each machine

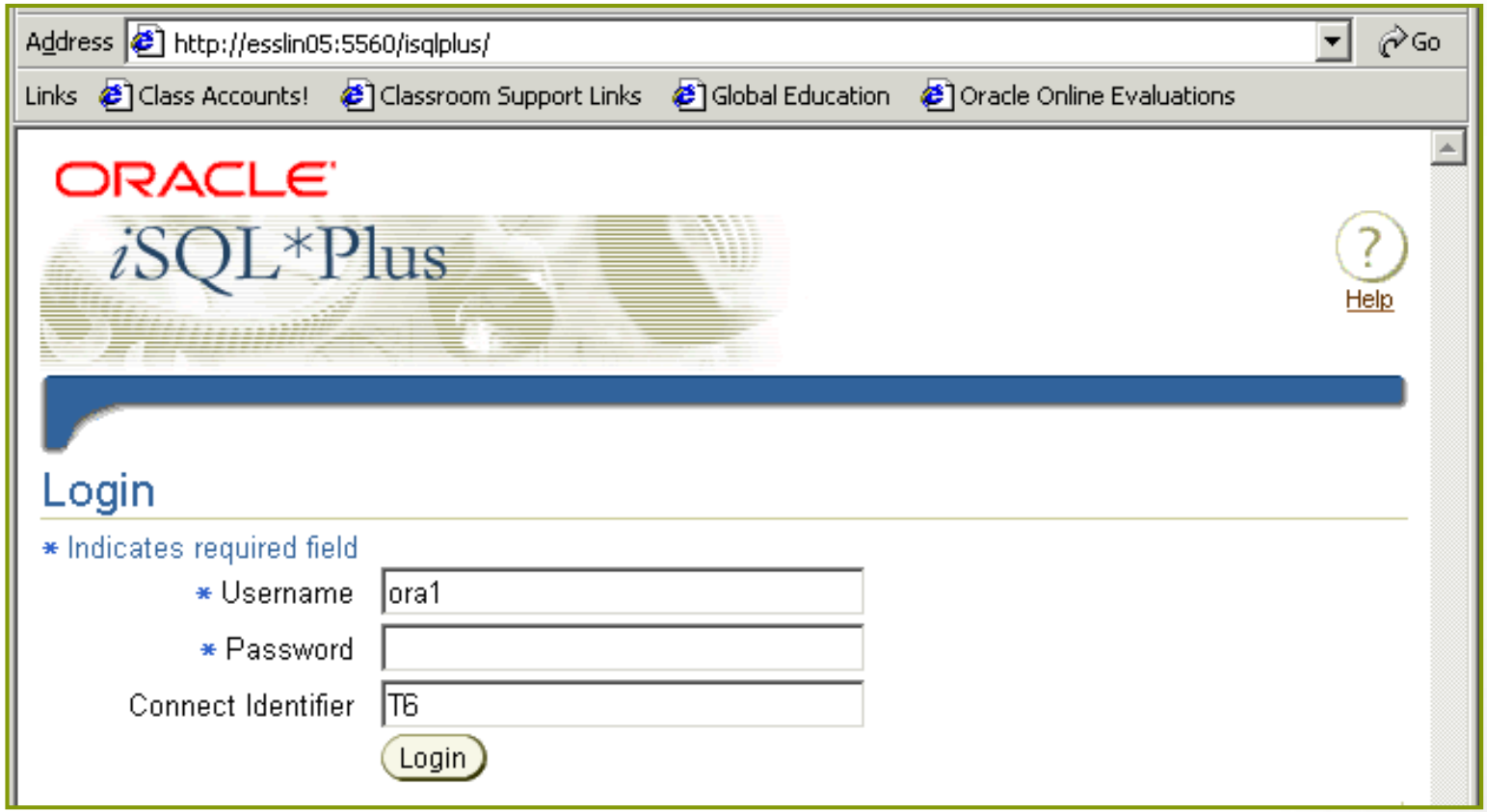
***i*SQL*Plus commands**

Overview of *i*SQL*Plus

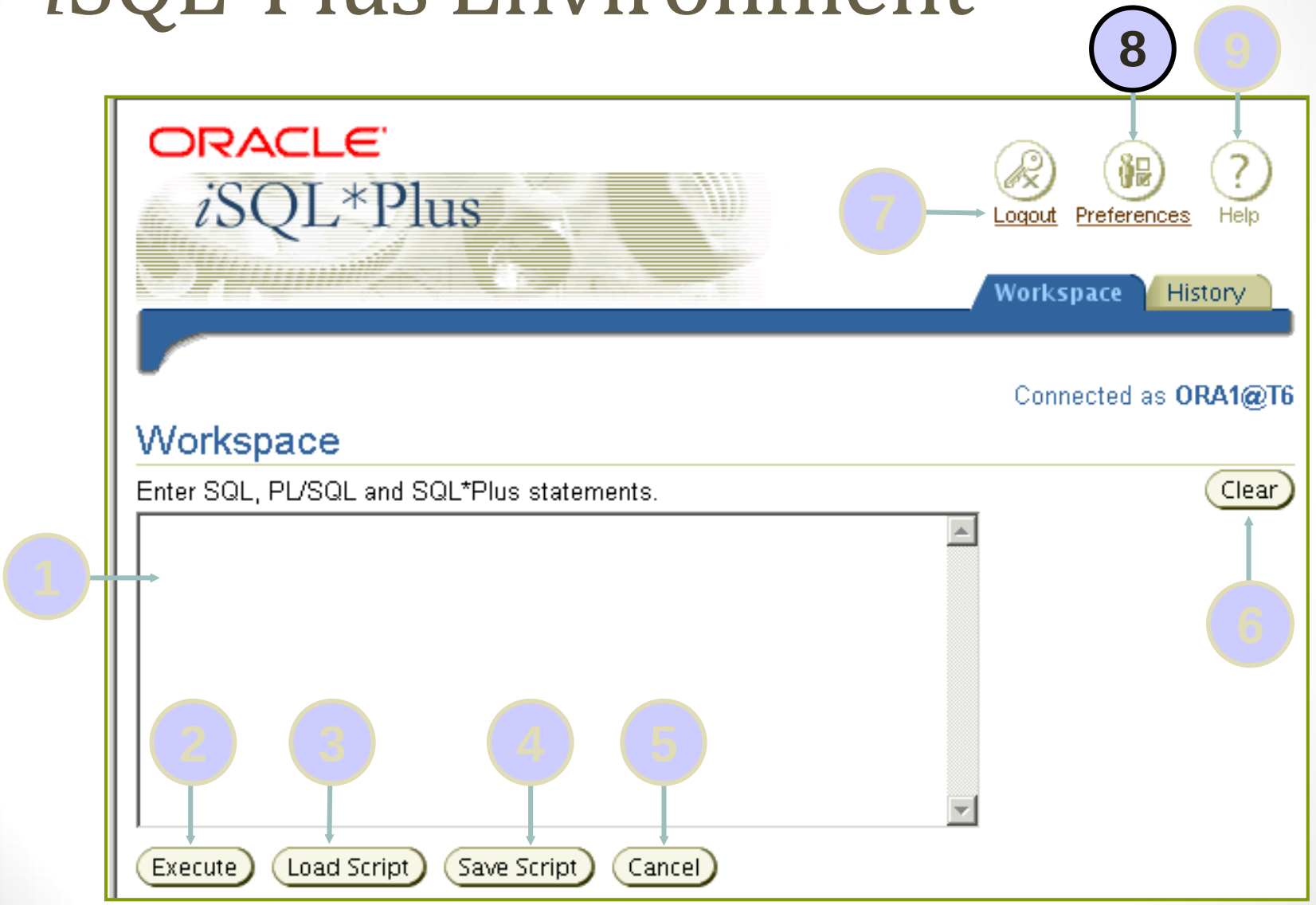
- After you log in to *i*SQL*Plus, you can:
  - Describe table structures
  - Enter, execute, and edit SQL statements
  - Save or append SQL statements to files
  - Execute or edit statements that are stored in saved script files

Logging In to *i*SQL*Plus

From your browser environment:



*i*SQL*Plus Environment



Displaying Table Structure

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

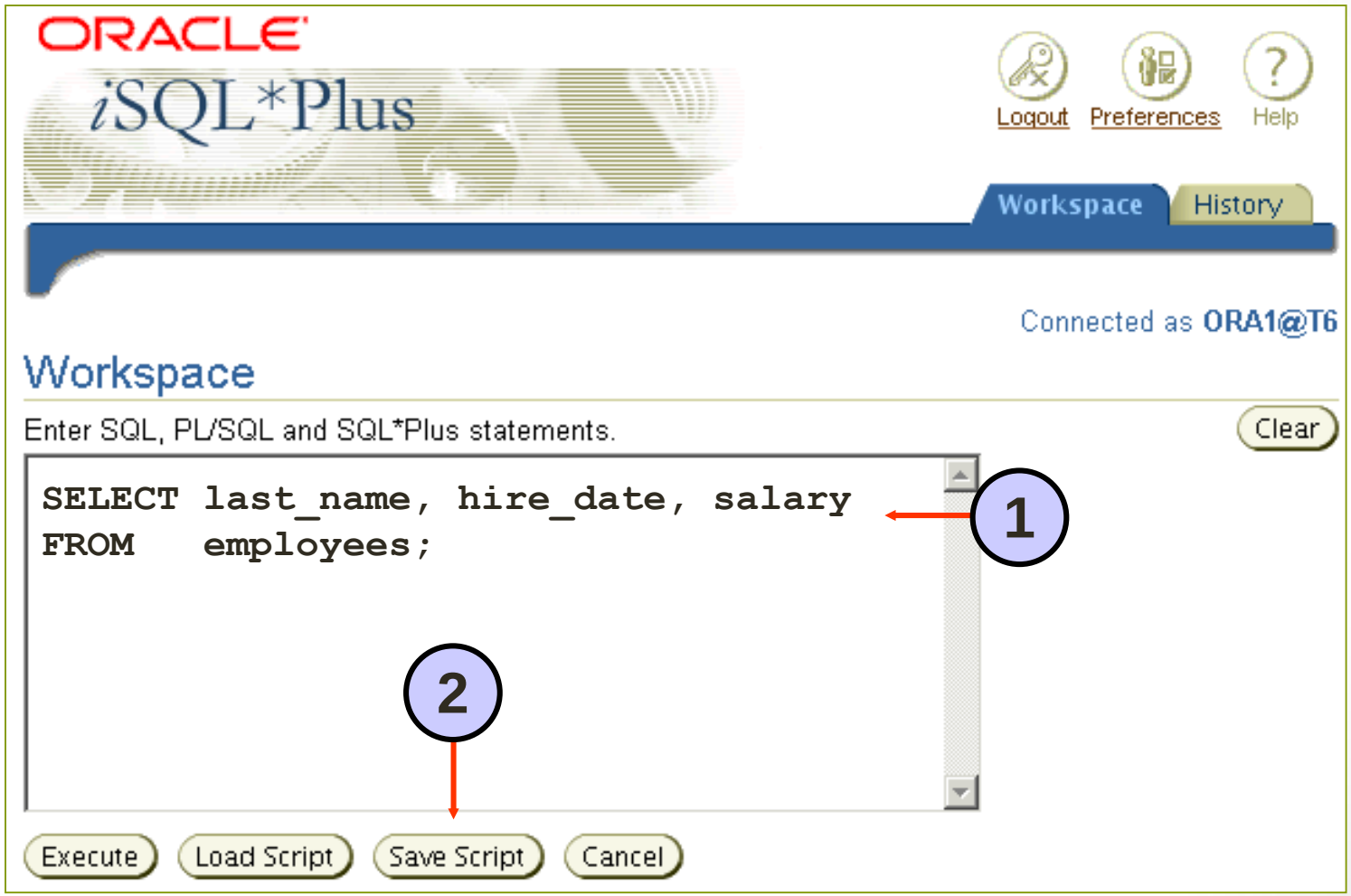
```
DESC[RIBE] tablename
```

Displaying Table Structure

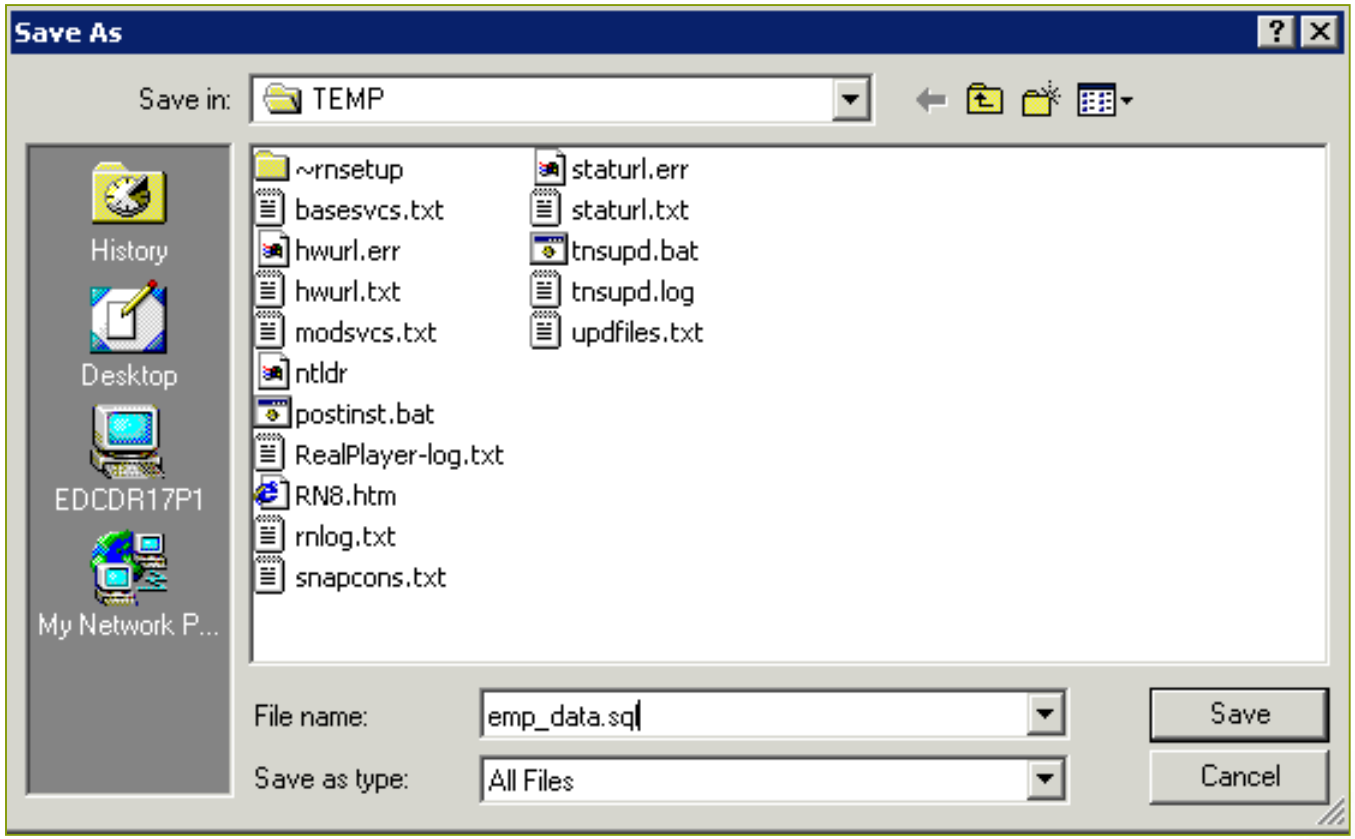
```
DESCRIBE employees
```

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)

Interacting with Script Files

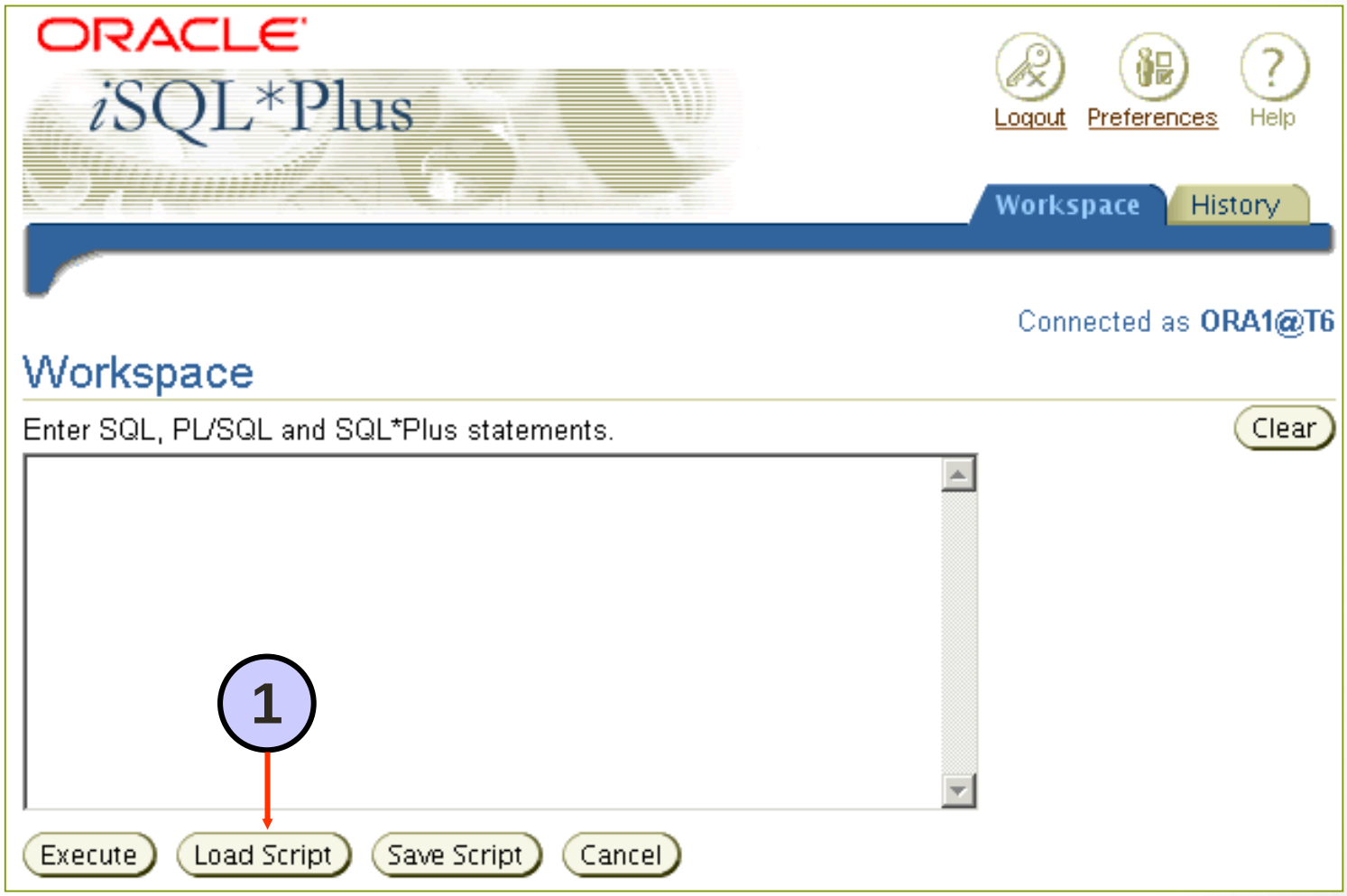


Interacting with Script Files

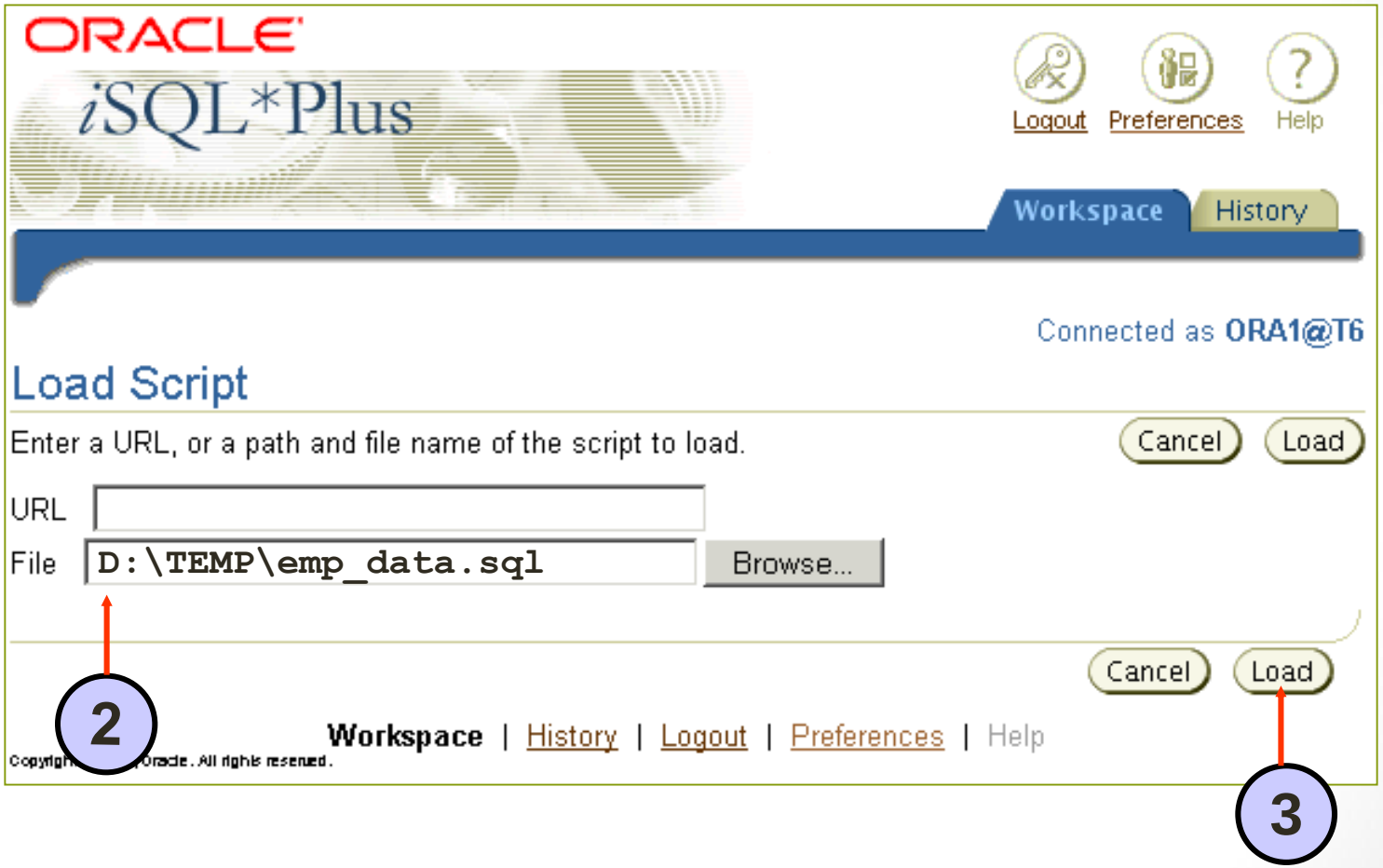


by ABHISHEK SHARMA

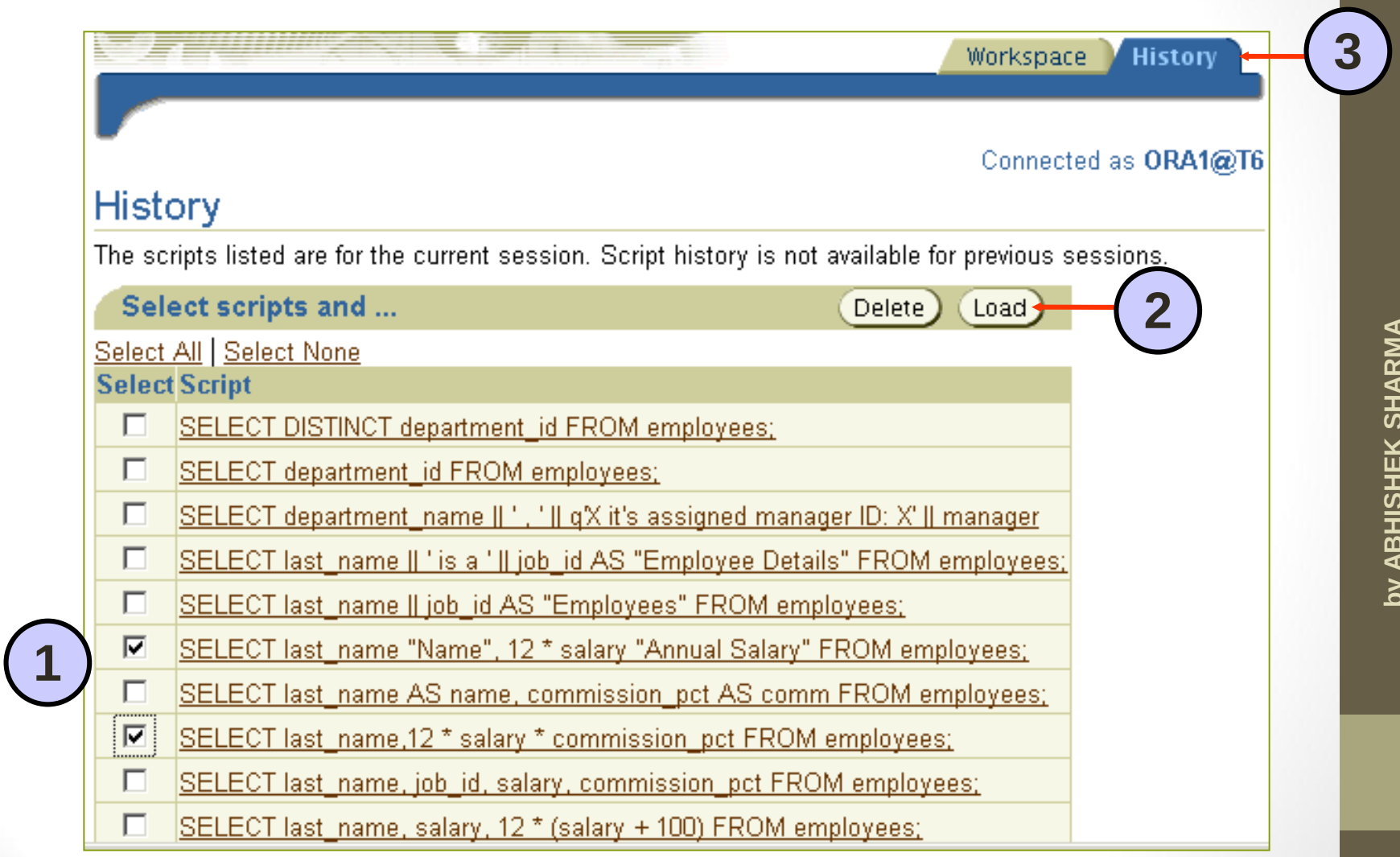
Interacting with Script Files



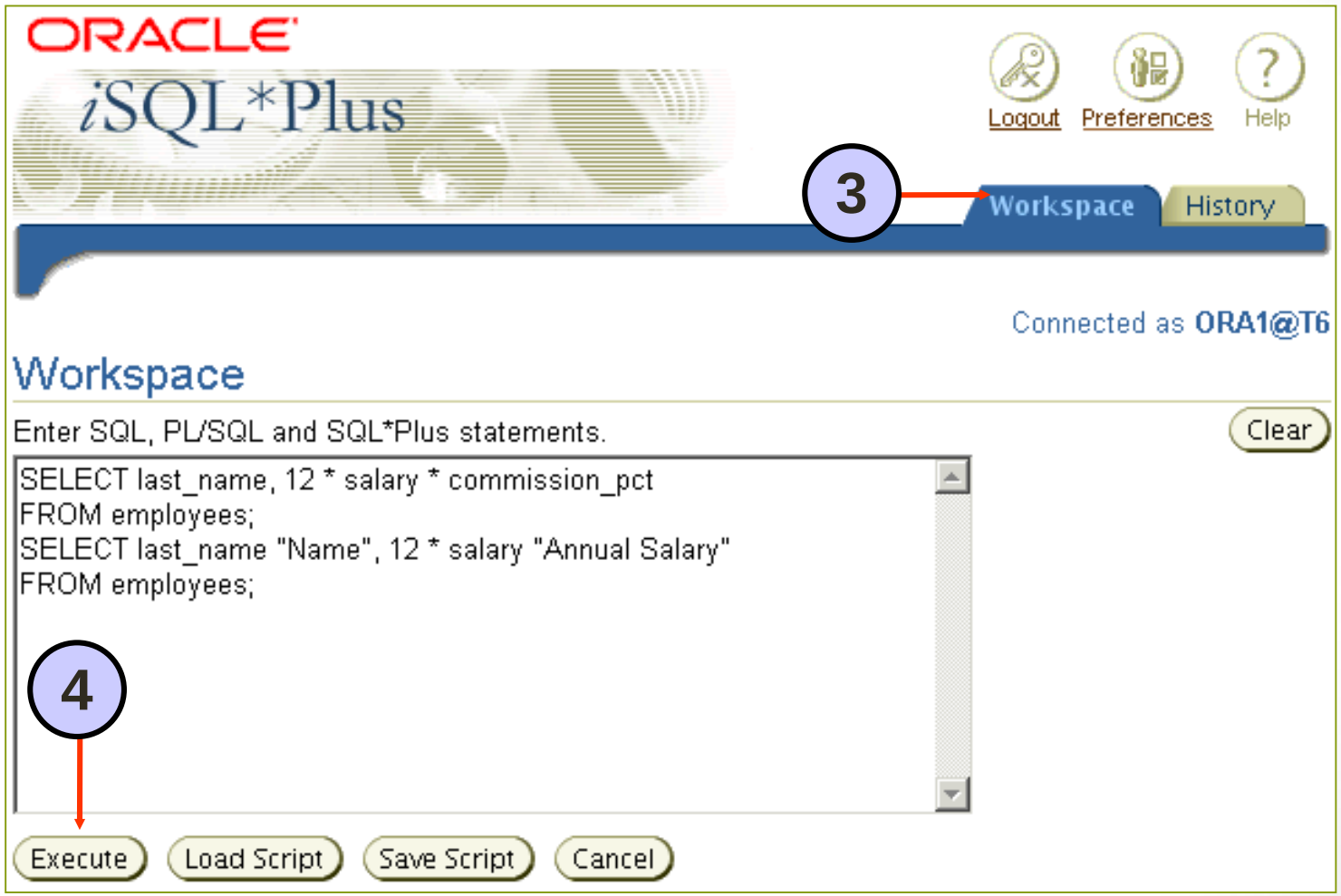
Interacting with Script Files



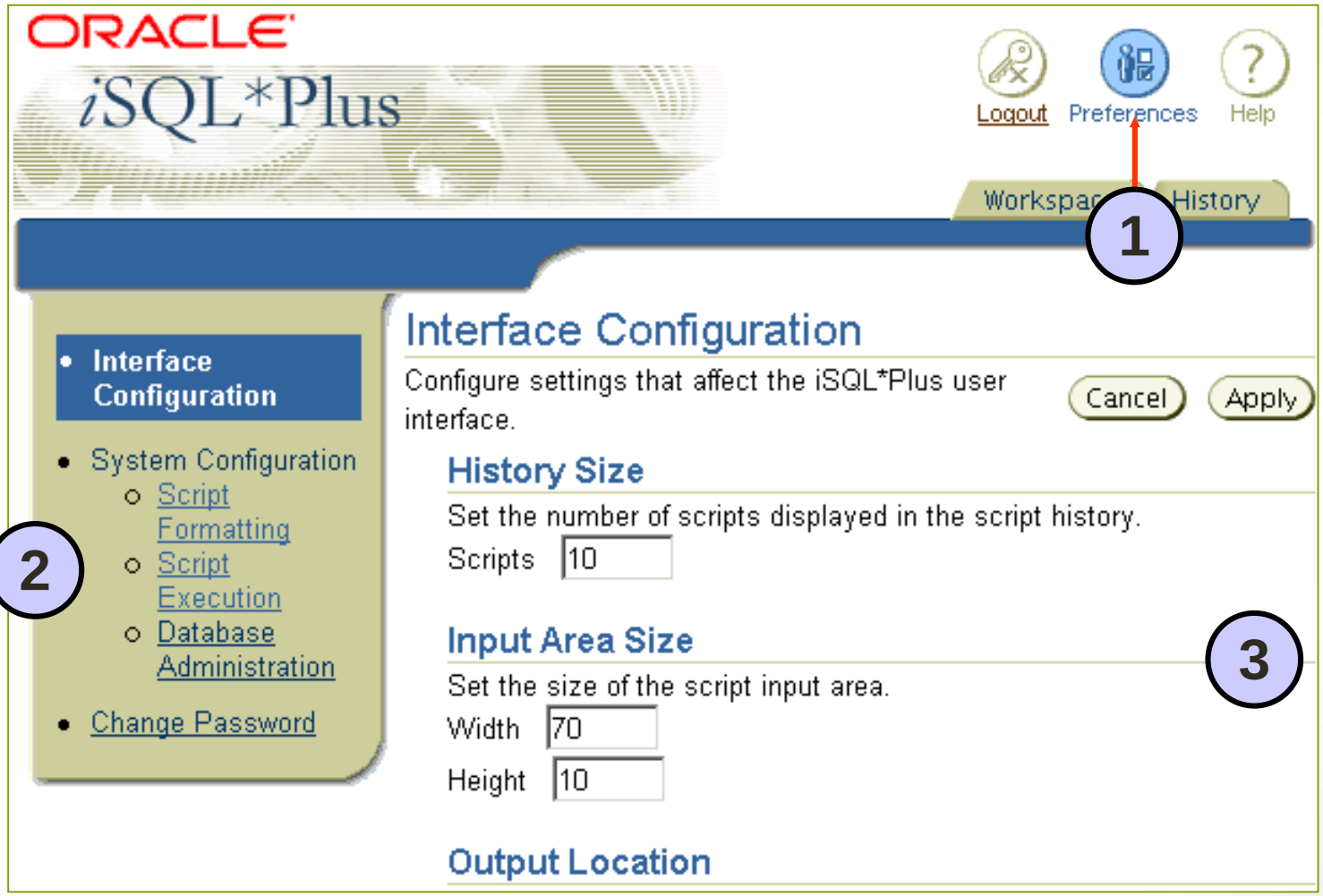
*i*SQL*Plus History Page



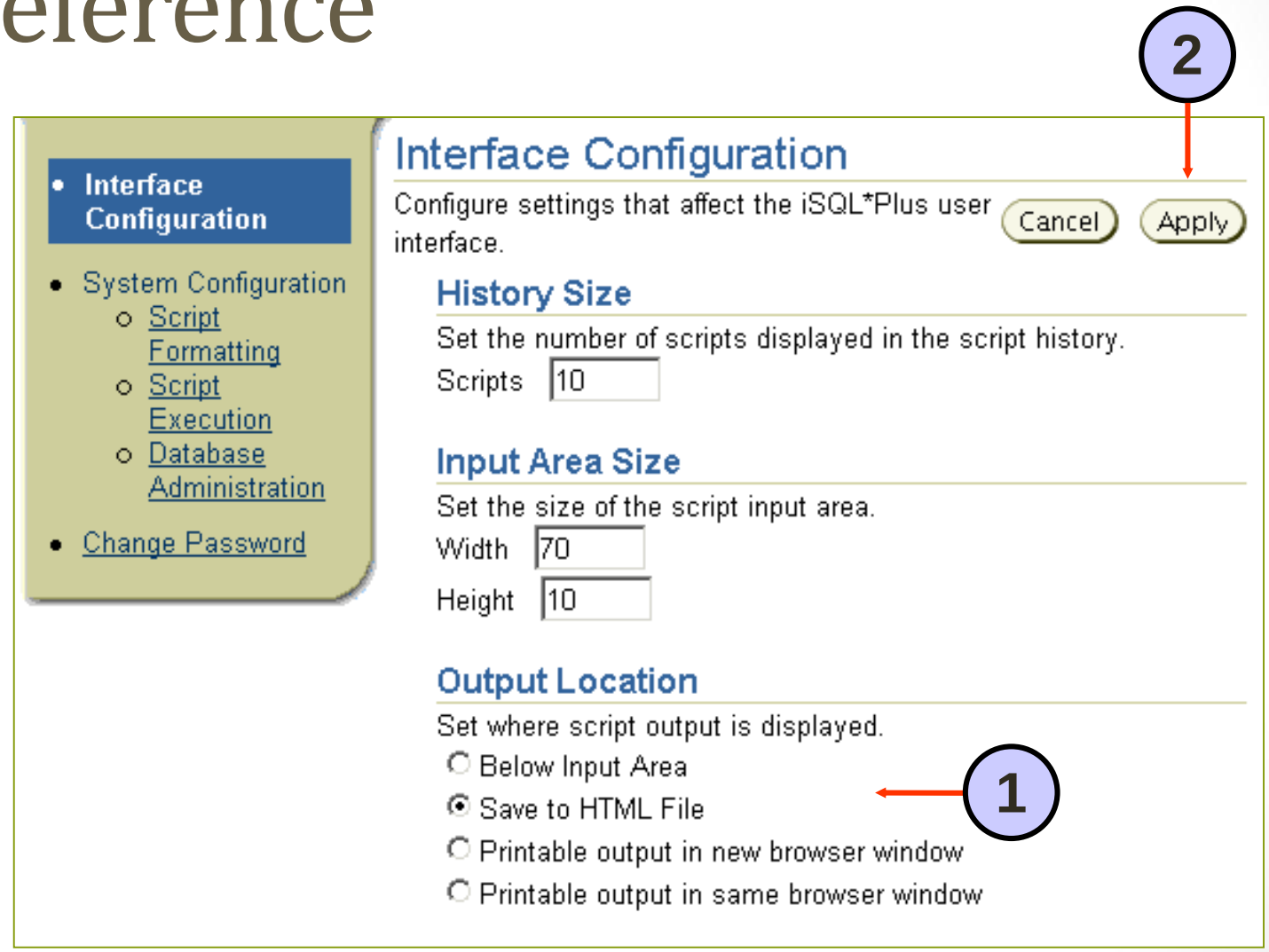
*i*SQL*Plus History Page



Setting *i*SQL*Plus Preferences



Setting the Output Location Preference



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;
```

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