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 iSQL*Plus commands

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

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

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>DEPARTMENT_NAME</th>
<th>MANAGER_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Administration</td>
<td>200</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>Marketing</td>
<td>201</td>
<td>1800</td>
</tr>
<tr>
<td>30</td>
<td>Shipping</td>
<td>124</td>
<td>1500</td>
</tr>
<tr>
<td>40</td>
<td>IT</td>
<td>103</td>
<td>1400</td>
</tr>
<tr>
<td>50</td>
<td>Sales</td>
<td>149</td>
<td>2500</td>
</tr>
<tr>
<td>60</td>
<td>Executive</td>
<td>100</td>
<td>1700</td>
</tr>
<tr>
<td>70</td>
<td>Accounting</td>
<td>205</td>
<td>1700</td>
</tr>
<tr>
<td>80</td>
<td>Contracting</td>
<td></td>
<td>1700</td>
</tr>
</tbody>
</table>

7 rows selected.

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;
```

<table>
<thead>
<tr>
<th>DEPARTMENT_ID</th>
<th>LOCATION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1700</td>
</tr>
<tr>
<td>20</td>
<td>1800</td>
</tr>
<tr>
<td>60</td>
<td>1500</td>
</tr>
<tr>
<td>62</td>
<td>1400</td>
</tr>
<tr>
<td>60</td>
<td>2500</td>
</tr>
<tr>
<td>90</td>
<td>1700</td>
</tr>
<tr>
<td>110</td>
<td>1700</td>
</tr>
<tr>
<td>190</td>
<td>1700</td>
</tr>
</tbody>
</table>

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;
```

<table>
<thead>
<tr>
<th>LOCATION_ID</th>
<th>DEPARTMENT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1700</td>
<td>10</td>
</tr>
<tr>
<td>1800</td>
<td>20</td>
</tr>
<tr>
<td>1500</td>
<td>50</td>
</tr>
</tbody>
</table>

... 

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 iSQL*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 (;).

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 iSQL*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

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

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

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>HIRE_DATE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>17-JUN-87</td>
<td>24000</td>
</tr>
<tr>
<td>Kochhar</td>
<td>21-SEP-99</td>
<td>17000</td>
</tr>
<tr>
<td>De Haan</td>
<td>13-JAN-93</td>
<td>17000</td>
</tr>
<tr>
<td>Hunold</td>
<td>03-JAN-90</td>
<td>9000</td>
</tr>
<tr>
<td>Ernst</td>
<td>21-MAY-91</td>
<td>6000</td>
</tr>
<tr>
<td>Higgins</td>
<td>07-JUN-94</td>
<td>12000</td>
</tr>
<tr>
<td>Getz</td>
<td>07-JUN-94</td>
<td>8000</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Add</td>
</tr>
<tr>
<td>-</td>
<td>Subtract</td>
</tr>
<tr>
<td>*</td>
<td>Multiply</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>12*salary+100</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>34000</td>
<td>209000</td>
</tr>
<tr>
<td>Kochhar</td>
<td>17000</td>
<td>204100</td>
</tr>
<tr>
<td>De Haan</td>
<td>17000</td>
<td>204100</td>
</tr>
<tr>
<td>***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 rows selected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>12*(salary+100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>34000</td>
<td>209000</td>
</tr>
<tr>
<td>Kochhar</td>
<td>17000</td>
<td>205300</td>
</tr>
<tr>
<td>De Haan</td>
<td>17000</td>
<td>205300</td>
</tr>
<tr>
<td>***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 rows selected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 \((12 \times \text{salary}) + 100\) 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.

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

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

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

Column Aliases
When displaying the result of a query, iSQL*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;

<table>
<thead>
<tr>
<th>NAME</th>
<th>COMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td></td>
</tr>
<tr>
<td>Kochhar</td>
<td></td>
</tr>
<tr>
<td>De Haan</td>
<td></td>
</tr>
</tbody>
</table>

20 rows selected.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Annual Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>King</td>
<td>288000</td>
</tr>
<tr>
<td>Kochhar</td>
<td>204000</td>
</tr>
<tr>
<td>De Haan</td>
<td>204000</td>
</tr>
</tbody>
</table>

20 rows selected.

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

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

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

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.
```

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.
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.

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.
Using Literal Character Strings

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

<table>
<thead>
<tr>
<th>Employee Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>King is a AD_PRES</td>
</tr>
<tr>
<td>Kochhar is a AD_VP</td>
</tr>
<tr>
<td>De Haan is a AD_VP</td>
</tr>
<tr>
<td>Hunold is a IT_PROG</td>
</tr>
<tr>
<td>Ernst is a IT_PROG</td>
</tr>
<tr>
<td>Lorentz is a IT_PROG</td>
</tr>
<tr>
<td>Mourgos is a ST_MAN</td>
</tr>
<tr>
<td>Raj is a ST_CLERK</td>
</tr>
</tbody>
</table>

20 rows selected.

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;
```

<table>
<thead>
<tr>
<th>MONTHLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>King: 1 Month salary = 24000</td>
</tr>
<tr>
<td>Kochhar: 1 Month salary = 17000</td>
</tr>
<tr>
<td>De Haan: 1 Month salary = 17000</td>
</tr>
<tr>
<td>Hunold: 1 Month salary = 9000</td>
</tr>
<tr>
<td>Ernst: 1 Month salary = 6000</td>
</tr>
<tr>
<td>Lorentz: 1 Month salary = 4200</td>
</tr>
<tr>
<td>Mourgos: 1 Month salary = 5800</td>
</tr>
<tr>
<td>Raj: 1 Month salary = 3500</td>
</tr>
</tbody>
</table>

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;
```

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

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

SELECT department_id
FROM employees;

SELECT DISTINCT department_id
FROM employees;

Duplicate Rows

Unless you indicate otherwise, 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 iSQL*Plus Interaction

SQL and iSQL*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 iSQL*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 Statements Versus iSQL*Plus Commands

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

iSQL*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

SQL and iSQL*Plus (continued)
The following table compares SQL and iSQL*Plus:

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

After you log in to iSQL*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

iSQL*Plus

iSQL*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

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

<table>
<thead>
<tr>
<th>Category</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment</td>
<td>Affects the general behavior of SQL statements for the session</td>
</tr>
<tr>
<td>Format</td>
<td>Formats query results</td>
</tr>
<tr>
<td>File manipulation</td>
<td>Saves statements in text script files and runs statements from text script files</td>
</tr>
<tr>
<td>Execution</td>
<td>Sends SQL statements from the browser to the Oracle server</td>
</tr>
<tr>
<td>Edit</td>
<td>Modifies SQL statements in the Edit window</td>
</tr>
<tr>
<td>Interaction</td>
<td>Enables you to create and pass variables to SQL statements, print variable values, and print messages to the screen</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Has various commands to connect to the database, manipulate the iSQL*Plus environment, and display column definitions</td>
</tr>
</tbody>
</table>
Logging In to *iSQL*\(^{+}\)Plus

From your browser environment:

1. Start the browser.
2. Enter the URL address of the *iSQL*\(^{+}\)Plus environment.
3. On the Login page, enter appropriate values in the Username, Password, and Connect Identifier fields.

Logging In to *iSQL*\(^{+}\)Plus

To log in from a browser environment:
1. Start the browser.
2. Enter the URL address of the *iSQL*\(^{+}\)Plus environment.
3. On the Login page, enter appropriate values in the Username, Password, and Connect Identifier fields.
iSQL*Plus Environment

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

1. **Text box**: Area where you type the SQL statements and iSQL*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 iSQL*Plus session and return to the iSQL*Plus Login page
8. **Preferences icon**: Click to change your interface configuration, system configuration, or password
9. **Help icon**: Provides access to iSQL*Plus help documentation
Displaying Table Structure

Use the **iSQL*Plus DESCRIBE** command to display the structure of a table:

```
DESC[RIBE]  tablename
```

Displaying the Table Structure

In **iSQL*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 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:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER((p, s))</td>
<td>Number value having a maximum number of digits (p), with (s) digits to the right of the decimal point</td>
</tr>
<tr>
<td>VARCHAR2((s))</td>
<td>Variable-length character value of maximum size (s)</td>
</tr>
<tr>
<td>DATE</td>
<td>Date and time value between January 1, 4712 B.C., and December 31, A.D 9999.</td>
</tr>
<tr>
<td>CHAR((s))</td>
<td>Fixed-length character value of size (s)</td>
</tr>
</tbody>
</table>
Interacting with Script Files

Placing Statements and Commands into a Text Script File

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

1. Type the SQL statements in the text box in 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.

Example:

```
SELECT last_name, hire_date, salary
FROM employees;
```
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 (continued)

Using Statements and Commands from a Script File in iSQL*Plus

You can use previously saved commands and statements from a script file in iSQL*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

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.
Running Previous Statements

The History page in iSQL*Plus lets you execute previously run statements in your session. The History page shows your most recently run SQL statements and iSQL*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.
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 iSQL*Plus Preferences

1. You can set preferences for your iSQL*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.
Changing the Output Location

You can send the results that are generated by a SQL statement or iSQL*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 iSQL*Plus environment to write, save, and execute SQL statements and iSQL*Plus commands

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

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

iSQL*Plus

iSQL*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 iSQL*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 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

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 iSQL*Plus session using the user ID and password that are provided by the instructor.

2. iSQL*Plus commands access the database.

   True/False

3. The following SELECT statement executes successfully:

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

   True/False

4. The following SELECT statement executes successfully:

   ```sql
   SELECT *
   FROM job_grades;
   ```

   True/False

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

   ```sql
   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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT_ID</td>
<td>NOT NULL</td>
<td>NUMBER(4)</td>
</tr>
<tr>
<td>DEPARTMENT_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>MANAGER_ID</td>
<td></td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>LOCATION_ID</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
</tbody>
</table>

Oracle Database 10g: SQL Fundamentals I  1 - 38
Practice 1 (continued)

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE_ID</td>
<td>NOT NULL</td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(20)</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(25)</td>
</tr>
<tr>
<td>EMAIL</td>
<td>NOT NULL</td>
<td>VARCHAR2(25)</td>
</tr>
<tr>
<td>PHONE_NUMBER</td>
<td>NOT NULL</td>
<td>VARCHAR2(20)</td>
</tr>
<tr>
<td>HIRE_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
<tr>
<td>JOB_ID</td>
<td>NOT NULL</td>
<td>VARCHAR2(10)</td>
</tr>
<tr>
<td>SALARY</td>
<td></td>
<td>NUMBER(8,2)</td>
</tr>
<tr>
<td>COMMISSION_PCT</td>
<td></td>
<td>NUMBER(2,2)</td>
</tr>
<tr>
<td>MANAGER_ID</td>
<td></td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>DEPARTMENT_ID</td>
<td></td>
<td>NUMBER(4)</td>
</tr>
</tbody>
</table>

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.

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

```sql
SELECT JOB_ID FROM EMPLOYEES GROUP BY JOB_ID;
```

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 iSQL*Plus text box. Name the column headings Emp #, Employee, Job, and Hire Date, respectively. Then run your query again.

<table>
<thead>
<tr>
<th>Emp #</th>
<th>Employee</th>
<th>Job</th>
<th>Hire Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>King</td>
<td>AD_PRES</td>
<td>17-JUN-87</td>
</tr>
<tr>
<td>101</td>
<td>Kochhar</td>
<td>AD_VP</td>
<td>21-SEP-99</td>
</tr>
<tr>
<td>102</td>
<td>De Haan</td>
<td>AD_VP</td>
<td>13-JAN-93</td>
</tr>
<tr>
<td>103</td>
<td>Hunold</td>
<td>IT_PROG</td>
<td>03-JAN-90</td>
</tr>
<tr>
<td>104</td>
<td>Ernst</td>
<td>IT_PROG</td>
<td>21-MAY-91</td>
</tr>
<tr>
<td>107</td>
<td>Lorentz</td>
<td>IT_PROG</td>
<td>07-FEB-99</td>
</tr>
<tr>
<td>124</td>
<td>Mourgos</td>
<td>ST_MAN</td>
<td>16-NOV-99</td>
</tr>
<tr>
<td>141</td>
<td>Rajs</td>
<td>ST_CLERK</td>
<td>17-OCT-95</td>
</tr>
<tr>
<td>142</td>
<td>Davies</td>
<td>ST_CLERK</td>
<td>29-JAN-97</td>
</tr>
<tr>
<td>143</td>
<td>Mates</td>
<td>ST_CLERK</td>
<td>15-MAR-98</td>
</tr>
<tr>
<td>144</td>
<td>Vargas</td>
<td>ST_CLERK</td>
<td>09-JUL-98</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Employee and Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>King, AD_PRES</td>
</tr>
<tr>
<td>Kochhar, AD_VP</td>
</tr>
<tr>
<td>De Haan, AD_VP</td>
</tr>
<tr>
<td>Hunold, IT_PROG</td>
</tr>
<tr>
<td>Ernst, IT_PROG</td>
</tr>
<tr>
<td>Lorentz, IT_PROG</td>
</tr>
<tr>
<td>Mourgos, ST_MAN</td>
</tr>
<tr>
<td>Rajs, ST_CLERK</td>
</tr>
<tr>
<td>Davies, ST_CLERK</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>THE_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,Steven,King,SKING,515.123.4567,AD_PRES.,17-JUN-87,24000.,90</td>
</tr>
<tr>
<td>101,Neena,Kochhar,NKOCHHAR,515.123.4568,AD_VP,100,21-SEP-89,17000.,90</td>
</tr>
<tr>
<td>102,Lex,De Haan,LDEHAAN,515.123.4569,AD_VP,100,13-JAN-93,17000.,90</td>
</tr>
<tr>
<td>103,Alexander,Hunold,AHUNOLD,590.423.4567,IT_PROG,102,03-JAN-90,3000.,50</td>
</tr>
<tr>
<td>104,Bruce,Enst,BERNST,590.423.4568,IT_PROG,103,21-MAY-91,6000.,60</td>
</tr>
<tr>
<td>107,Diana,Lorentz,DLORENTZ,590.423.5567,IT_PROG,103,07-FEB-99,4200.,30</td>
</tr>
<tr>
<td>124, Kevin, Mourgos, KMOURGOS, 650.123.5234, ST_MAN, 100,16-NOV-99,5500.,50</td>
</tr>
<tr>
<td>141,Trenna,Rajs,TRAJS,650.121.8009,ST_CLERK,124,17-OCT-95,3500.,50</td>
</tr>
<tr>
<td>142,Curtis,Davies,CDAVIES,650.121.2994,ST_CLERK,124,29-JAN-97,3100.,50</td>
</tr>
<tr>
<td>143,Randall,Matos,RMATOS,650.121.2674,ST_CLERK,124,15-MAR-98,2600.,50</td>
</tr>
<tr>
<td>144,Peter,Vargas, PVARGAS,650.121.2004,ST_CLERK,124,09-JUL-98,2500.,50</td>
</tr>
</tbody>
</table>

20 rows selected.