Writing Basic SQL Statements

<table>
<thead>
<tr>
<th>Schedule:</th>
<th>Timing</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80 minutes</td>
<td>Lecture</td>
</tr>
<tr>
<td></td>
<td>40 minutes</td>
<td>Practice</td>
</tr>
<tr>
<td></td>
<td>120 minutes</td>
<td>Total</td>
</tr>
</tbody>
</table>
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 SQL*Plus commands

Lesson Aim

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 you need to perform these actions. You may want to create SELECT statements that can be used time and time again. This lesson also covers the use of SQL*Plus commands to execute SQL statements.
Capabilities of SQL SELECT Statements

A SELECT statement retrieves information from the database. Using a SELECT statement, you can do the following:

- Selection: You can use the selection capability in SQL to choose the rows in a table that you want returned by a query. You can use various criteria to selectively restrict the rows that you see.
- Projection: You can use the projection capability in SQL to choose the columns in a table that you want returned by your query. You can choose as few or as many columns of the table as you require.
- Join: You can use the join capability in SQL to bring together data that is stored in different tables by creating a link between them. You will learn more about joins in a later lesson.
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 specifies the table containing the columns listed in the SELECT clause

In the syntax:

\[
\text{SELECT } [\text{DISTINCT}] \{*, \text{column [alias]}\},... \text{ FROM table;}
\]

- SELECT identifies *what* columns.
- FROM identifies *which* table.

**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 specifies the table containing the columns listed in the SELECT clause

In the syntax:

- `SELECT` is a list of one or more columns.
- `DISTINCT` suppresses duplicates.
- `*` selects all columns.
- `column` selects the named column.
- `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.

- 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 empno, ename, ...` is a clause.
- A *statement* is a combination of two or more clauses. For example, `SELECT * FROM emp` is a SQL statement.
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.
• Tabs and indents are used to enhance readability.

Executing SQL Statements

• Place a semicolon (;) at the end of the last clause.
• Place a slash on the last line in the buffer.
• Place a slash at the SQL prompt.
• Issue a SQL*Plus RUN command at the SQL prompt.

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 abbreviated or split across lines.
• Clauses are usually placed on separate lines for readability and ease of editing.
• Tabs and indents can 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.
• Within SQL*Plus, a SQL statement is entered at the SQL prompt, and the subsequent lines are numbered. This is called the SQL buffer. Only one statement can be current at any time within the buffer.
Selecting All Columns, All Rows

You can display all columns of data in a table by following the SELECT keyword with an asterisk (*). In the example on the slide, the department table contains three columns: DEPTNO, DNAME, and LOC. The table contains four 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 on the slide, displays all columns and all rows of the DEPT table:

```
SQL> SELECT deptno, dname, loc
2  FROM dept;
```
Selecting Specific Columns, 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 on the slide displays all the department numbers and locations from the DEPT table.

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

```sql
SQL> SELECT loc, deptno
2  FROM dept;
```

<table>
<thead>
<tr>
<th>LOC</th>
<th>DEPTNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW YORK</td>
<td>10</td>
</tr>
<tr>
<td>DALLAS</td>
<td>20</td>
</tr>
<tr>
<td>CHICAGO</td>
<td>30</td>
</tr>
<tr>
<td>BOSTON</td>
<td>40</td>
</tr>
</tbody>
</table>
Column Heading Defaults

- Default justification
  - Left: Date and character data
  - Right: Numeric data
- Default display: Uppercase

Character and date column headings can be truncated, but number headings cannot be truncated. The column headings appear in uppercase by default. You can override the column heading display with an alias. Column aliases are covered later in this lesson.
Arithmetic Expressions

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

Arithmetic Operators

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

<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>
Using Arithmetic Operators

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

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

**Note:** SQL*Plus ignores blank spaces before and after the arithmetic operator.
Operator Precedence

- Multiplication and division take priority over addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to force prioritized evaluation and to clarify statements.

Operator Precedence

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

You can use parentheses to force the expression within parentheses to be evaluated first.
### Operator Precedence (continued)

The example on the slide displays the name, salary, and annual compensation of employees. It calculates the annual compensation as 12 multiplied by the monthly salary, plus a one-time bonus of $100. Notice that multiplication is performed before addition.

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

```sql
SQL> SELECT ename, sal, 12*sal+100
  2  FROM   emp;

<table>
<thead>
<tr>
<th>ENAME</th>
<th>SAL</th>
<th>12*SAL+100</th>
</tr>
</thead>
<tbody>
<tr>
<td>KING</td>
<td>5000</td>
<td>60100</td>
</tr>
<tr>
<td>BLAKE</td>
<td>2850</td>
<td>34300</td>
</tr>
<tr>
<td>CLARK</td>
<td>2450</td>
<td>29500</td>
</tr>
<tr>
<td>JONES</td>
<td>2975</td>
<td>35800</td>
</tr>
<tr>
<td>MARTIN</td>
<td>1250</td>
<td>15100</td>
</tr>
<tr>
<td>ALLEN</td>
<td>1600</td>
<td>19300</td>
</tr>
</tbody>
</table>
```

14 rows selected.
Using Parentheses

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

The example on the slide displays the name, salary, and annual compensation of employees. It calculates the annual compensation as monthly salary plus a monthly bonus of $100, multiplied by 12. Because of the parentheses, addition takes priority over multiplication.
Null Values

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

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

Columns of any datatype can contain null values, unless the column was defined as NOT NULL or as PRIMARY KEY when the column was created.

In the COMM column in the EMP table, you notice that only a SALESMAN can earn commission. Other employees are not entitled to earn commission. A null value represents that fact. Turner, who is a salesman, does not earn any commission. Notice that his commission is zero and not null.
Null Values in Arithmetic Expressions

Arithmetic expressions containing a null value evaluate to null.

SQL> select ename, 12*sal+comm 
2  from emp 
3  WHERE ename='KING';

<table>
<thead>
<tr>
<th>ENAME</th>
<th>12*SAL+COMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>KING</td>
<td></td>
</tr>
</tbody>
</table>

Null Values (continued)

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

In the example on the slide, employee KING is not in SALESMAN and does not get any commission. Because the COMM column in the arithmetic expression is null, the result is null.

For more information, see Oracle Server SQL Reference, Release 8, “Elements of SQL.”
Defining a Column Alias

• Renames a column heading
• Is useful with calculations
• Immediately follows column name; optional AS keyword between column name and alias
• Requires double quotation marks if it contains spaces or special characters or is case sensitive

Column Aliases

When displaying the result of a query, SQL*Plus normally uses the name of the selected column as the column heading. In many cases, this heading may not be descriptive and hence is 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, special characters (such as # or $), or is case sensitive, enclose the alias in double quotation marks (" ").
Column Aliases (continued)

The first example displays the name and the monthly salary of all the employees. Notice that the optional AS keyword has been used before the column alias name. The result of the query would be the same whether the AS keyword is used or not. Also notice that the SQL statement has the column aliases, name and salary, in lowercase, whereas the result of the query displays the column headings in uppercase. As mentioned in the last slide, column headings appear in uppercase by default.

The second example displays the name and annual salary of all the employees. Because Annual Salary contains spaces, 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

- Concatenates columns or character strings to other columns
- Is represented by two vertical bars (||)
- Creates a resultant column that is a character expression

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.
Using the Concatenation Operator

SQL> SELECT ename||job AS "Employees"
2  FROM emp;

Employees
------------
KINGPRESIDENT
BLAKEMANAGER
CLARKMANAGER
JONESMANAGER
MARTINSALESMAN
ALLEN SALESMAN
...
14 rows selected.

Concatenation Operator (continued)

In the example, ENAME and JOB are concatenated, and they are given the alias Employees. Notice that the employee number and job 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 included in the SELECT list.
- Date and character literal values must be enclosed within single quotation marks.
- Each character string is output once for each row returned.

Literal Character Strings

A literal is character, a number, or a date included in the SELECT list 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 within single quotation marks (‘ ’); number literals must not.
Literal Character Strings (continued)

The example on the slide displays names and jobs 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 name and salary for each employee is concatenated with a literal to give the returned rows more meaning.

```sql
SQL> SELECT ename ||" is a" ||job
2   AS "Employee Details"
3  FROM emp;

Employee Details
-------------------------
KING is a PRESIDENT
BLAKE is a MANAGER
CLARK is a MANAGER
JONES is a MANAGER
MARTIN is a SALESMAN
...
14 rows selected.
```

```sql
SQL> SELECT ename ||': '||'1' ||' Month salary = '||sal Monthly
2  FROM emp;

MONTHLY
---------------------------------------------------------------
KING: 1 Month salary = 5000
BLAKE: 1 Month salary = 2850
CLARK: 1 Month salary = 2450
JONES: 1 Month salary = 2975
MARTIN: 1 Month salary = 1250
ALLEN: 1 Month salary = 1600
TURNER: 1 Month salary = 1500
...
14 rows selected.
```
Duplicate Rows

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

SQL> SELECT deptno
2       FROM emp;

DEPTNO
-------
  10  
  30  
  10  
  20  

...  
14 rows selected.

Duplicate Rows

Unless you indicate otherwise, SQL*Plus displays the results of a query without eliminating duplicate rows. The example on the slide displays all the department numbers from the EMP table. Notice that the department numbers are repeated.
Eliminating Duplicate Rows

Eliminate duplicate rows by using the DISTINCT keyword in the SELECT clause.

```sql
SQL> SELECT DISTINCT deptno
2  FROM emp;
```

```
DEPTNO
-------
10
20
30
```

Duplicate Rows (continued)

To eliminate duplicate rows in the result, include the DISTINCT keyword in the SELECT clause immediately after the SELECT keyword. In the example on the slide, the EMP table actually contains fourteen rows but there are only three 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 represents a distinct combination of the columns.

```sql
SQL> SELECT deptno, job
2  FROM emp;
```

```
DEPTNO JOB
------- -------
10  CLERK
10  MANAGER
10  PRESIDENT
20  ANALYST
... 
9 rows selected.
```
SQL and SQL*Plus

SQL is a command language for communication with the Oracle Server from any tool or application. Oracle SQL contains many extensions. When you enter a SQL statement, it is stored in a part of memory called the SQL buffer and remains there until you enter a new statement.

SQL*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
- Reduces the amount of time required for creating and maintaining systems
- Is an English-like language

Features of SQL*Plus
- Accepts ad hoc entry of statements
- Accepts SQL input from files
- Provides a line editor for modifying SQL statements
- Controls environmental settings
- Formats query results into a basic report
- Accesses local and remote databases
SQL Statements Versus SQL*Plus Commands

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

**SQL*Plus**
- An environment
- Oracle proprietary
- Keywords can be abbreviated
- Commands do not allow manipulation of values in the database

---

**SQL and SQL*Plus (continued)**

The following table compares SQL and SQL*Plus:

<table>
<thead>
<tr>
<th>SQL</th>
<th>SQL*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>Manipulates data and table definitions in the database</td>
<td>Does not allow manipulation of values in the database</td>
</tr>
<tr>
<td>Is entered into the SQL buffer on one or more lines</td>
<td>Is entered one line at a time; not stored in the SQL buffer</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 a termination character to execute command immediately</td>
<td>Does not require termination characters; commands are executed immediately</td>
</tr>
<tr>
<td>Uses functions to perform some formatting</td>
<td>Uses commands to format data</td>
</tr>
</tbody>
</table>
Overview of SQL*Plus

- Log in to SQL*Plus.
- Describe the table structure.
- Edit your SQL statement.
- Execute SQL from SQL*Plus.
- Save SQL statements to files and append SQL statements to files.
- Execute saved files.
- Load commands from file to buffer to edit.

SQL*Plus

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 repetitive use in the future

SQL*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, loads, and runs script files</td>
</tr>
<tr>
<td>Execution</td>
<td>Sends SQL statements from SQL buffer to Oracle8 Server</td>
</tr>
<tr>
<td>Edit</td>
<td>Modifies SQL statements in the buffer</td>
</tr>
<tr>
<td>Interaction</td>
<td>Allows 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 SQL*Plus environment, and display column definitions</td>
</tr>
</tbody>
</table>

Instructor Note (for page 1-27)

Snippet: “Establishing a Database Session”

Logging in to SQL*Plus: Release number may vary, depending on the version installed.
Logging In to SQL*Plus

• From Windows environment:

  How you invoke SQL*Plus depends on which type of operating system or Windows environment you are running.
  To log in through a Windows environment:
  1. Click Start—>Programs—>Oracle for Windows NT—>SQL*Plus 8.0.
  2. Fill in username, password, and database.

• From command line:

  sqlplus [username[/password[@database]]]

Logging In to SQL*Plus

How you invoke SQL*Plus depends on which type of operating system or Windows environment you are running.

To log in through a Windows environment:
1. Click Start—>Programs—>Oracle for Windows NT—>SQL*Plus 8.0.
2. Fill in username, password, and database.

To log in through a command-line environment:
1. Log on to your machine.
2. Enter the SQL*Plus command as shown in the slide.

In the command:

  *username* is your database username
  *password* is your database password (if you enter your password here, it is visible)
  @database is the database connect string

**Note:** To ensure the integrity of your password, do not enter it at the operating system prompt. Instead, enter only your username. Enter your password at the Password prompt.

Once you are successfully logged in to SQL*Plus, you see the following message:

SQL*Plus : Release 8.0.3.0.0 - Production on Tue Jun 22 16:03:43 1999 (c) Copyright 1999 Oracle Corporation. All rights reserved.

By : Dr. Waqar Ahmad
Displaying Table Structure

Use the SQL*Plus DESCRIBE command to display the structure of a table.

DESC[RIBE] tablename

Displaying Table Structure

In SQL*Plus, you can display the structure of a table using the DESCRIBE command. The result of the command is to see the column names and datatypes as well as whether a column must contain data.

In the syntax:

   tablename    is the name of any existing table, view, or synonym accessible to the user
Displaying Table Structure

The example on the slide displays the information about the structure of the DEPT table.

In the result:

- **Null?** indicates whether a column must contain data; NOT NULL indicates that a column must contain data.
- **Type** displays the datatype for a column.

The datatypes are described in the following table:

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER((p,s))</td>
<td>Number value having a maximum number of digits (p), the number of digits to the right of the decimal point (s)</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, 9999 A.D.</td>
</tr>
<tr>
<td>CHAR((s))</td>
<td>Fixed-length character value of size (s)</td>
</tr>
</tbody>
</table>

Displaying Table Structure (continued)
SQL*Plus Editing Commands

SQL*Plus commands are entered one line at a time and are not stored in the SQL buffer.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[PPEND] <em>text</em></td>
<td>Adds text to the end of the current line</td>
</tr>
<tr>
<td>C[HANGE] / <em>old</em> / <em>new</em></td>
<td>Changes <em>old</em> text to <em>new</em> in the current line</td>
</tr>
<tr>
<td>C[HANGE] / <em>text</em> /</td>
<td>Deletes <em>text</em> from the current line</td>
</tr>
<tr>
<td>CL[EAR] BUFF[ER]</td>
<td>Deletes all lines from the SQL buffer</td>
</tr>
<tr>
<td>DEL</td>
<td>Deletes current line</td>
</tr>
</tbody>
</table>

Guidelines

- If you press [Return] before completing a command, SQL*Plus prompts you with a line number.
- You terminate the SQL buffer by either entering one of the terminator characters (semicolon or slash) or pressing [Return] twice. You then see the SQL prompt.
SQL*Plus Editing Commands

- I[NPUT]
- I[NPUT] text
- L[IST]
- L[IST] n
- L[IST] m n
- R[UN]
- n
- n text
- 0 text

SQL*Plus Editing Commands (continued)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I[NPUT]</td>
<td>Inserts an indefinite number of lines</td>
</tr>
<tr>
<td>I[NPUT] text</td>
<td>Inserts a line consisting of text</td>
</tr>
<tr>
<td>L[IST]</td>
<td>Lists all lines in the SQL buffer</td>
</tr>
<tr>
<td>L[IST] n</td>
<td>Lists one line (specified by n)</td>
</tr>
<tr>
<td>L[IST] m n</td>
<td>Lists a range of lines (m to n)</td>
</tr>
<tr>
<td>R[UN]</td>
<td>Displays and runs the current SQL statement in the buffer</td>
</tr>
<tr>
<td>n</td>
<td>Specifies the line to make the current line</td>
</tr>
<tr>
<td>n text</td>
<td>Replaces line n with text</td>
</tr>
<tr>
<td>0 text</td>
<td>Inserts a line before line 1</td>
</tr>
</tbody>
</table>

You can enter only one SQL*Plus command per SQL prompt. SQL*Plus commands are not stored in the buffer. To continue a SQL*Plus command on the next line, end the current line with a hyphen (-).

**Instructor Note**

Show students the use of the commonly used editing commands, such as A[PPEND], C[HANGE], DEL, L[IST], and R[UN].
SQL*Plus File Commands

SQL statements communicate with the Oracle Server. SQL*Plus commands control the environment, format query results, and manage files. You can use the commands identified in the following table:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVE <code>filename</code></td>
<td>Saves current contents of SQL buffer to a file. Use APPEND to add to an existing file; use REPLACE to overwrite an existing file. The default extension is <code>.sql</code>.</td>
</tr>
<tr>
<td>GET <code>filename</code></td>
<td>Writes the contents of a previously saved file to the SQL buffer. The default extension for the filename is <code>.sql</code>.</td>
</tr>
<tr>
<td>START <code>filename</code></td>
<td>Runs a previously saved command file.</td>
</tr>
<tr>
<td><code>@ filename</code></td>
<td>Runs a previously saved command file (same as START).</td>
</tr>
<tr>
<td>EDIT</td>
<td>Invokes the editor and saves the buffer contents to a file named afiedt.buf.</td>
</tr>
<tr>
<td>EDIT <code>[filename,ext]</code></td>
<td>Invokes editor to edit contents of a saved file.</td>
</tr>
<tr>
<td>SPOOL <code>[filename,ext]</code></td>
<td>Stores query results in a file. OFF closes the spool file. OUT closes the spool file and sends the file results to the system printer.</td>
</tr>
<tr>
<td>EXIT</td>
<td>Leaves SQL*Plus.</td>
</tr>
</tbody>
</table>
Summary

Use SQL*Plus as an environment to:
• Execute SQL statements
• Edit SQL statements

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

SELECT Statement
In this lesson, you have learned about retrieving data from a database table with the SELECT statement.

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

<table>
<thead>
<tr>
<th>where:</th>
<th>SELECT</th>
<th>is a list of at least one column</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISTINCT</td>
<td>suppresses duplicates</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>selects all columns</td>
<td></td>
</tr>
<tr>
<td>column</td>
<td>selects the named column</td>
<td></td>
</tr>
<tr>
<td>alias</td>
<td>gives selected column a different heading</td>
<td></td>
</tr>
<tr>
<td>FROM table</td>
<td>specifies the table containing the columns</td>
<td></td>
</tr>
</tbody>
</table>

SQL*Plus
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 Overview

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

Practice Overview
This is the first of many practices. The solutions (if you require them) can be found in Appendix A. Practices are intended to introduce all topics covered in the lesson. Questions 2–4 are paper-based.

In any practice, there may be “if you have time” or “if you want extra challenge” questions. Do these only if you have completed all other questions within the allocated time and would like a further challenge to your skills.

Take the practice slowly and precisely. You can experiment with saving and running command files. If you have any questions at any time, attract the instructor’s attention.

Paper-Based Questions
For questions 2–4, circle either True or False.

Instructor Note
Let the students know that to get a listing of the tables they can access during the course, the command is:

```
SQL> SELECT * FROM TAB;
```