Chapter 1

Writing Basic SQL Statements

Objectives

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

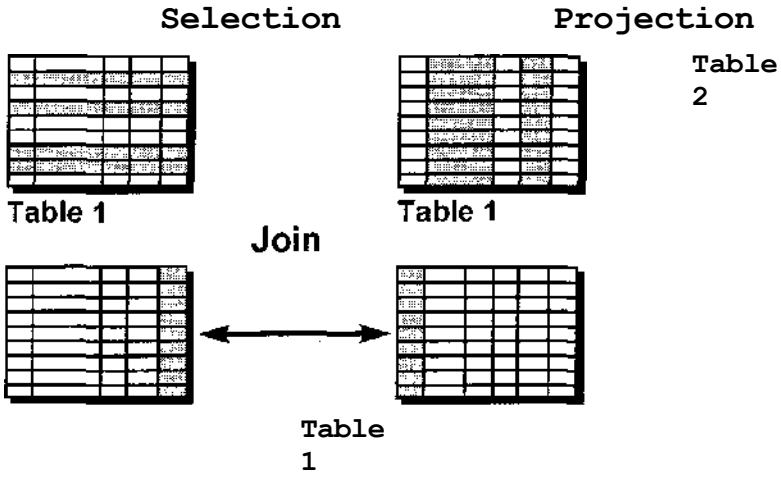
- **List the capabilities of SQL SELECT statements**
- **Execute a basic SELECT statement**
- **Differentiate between SQL statements and SQL*Plus commands**

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 Ihe SQL statements that you need to perform these actions.

You may want to create SELECT statements that can be used time and time again. Tins lesson also covers the use of SQL*Plus commands to execute SQL statements

Capabilities of SQL SELECT 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 through a column that both the tables share. You will learn more about joins in a later lesson.

Basic SELECT Statement

```
SELECT    [DISTINCT]   {*,   column   [alias],..,}
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 an 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.**

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.
- 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.

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 RUK command at the SQL prompt.

Selecting All Columns

```
SELECT  *
FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/1980	800		20
7499	ALLEN	SALESMAN	7698	20/02/1981	1600	300	30
7521	WARD	SALESMAN	7698	22/02/1981	1250	500	30
7566	JONES	MANAGER	7839	02/04/1981	2975		20
7654	MARTIN	SALESMAN	7698	28/09/1981	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/1981	2850		30
7782	CLARK	MANAGER	7839	09/06/1981	2450		10
7788	SCOTT	ANALYST	7566	09/12/1982	3000		20
7839	KING	PRESIDENT		17/11/1981	5000		10
7844	TURNER	SALESMAN	7698	08/09/1981	1500	0	30
7876	ADAMS	CLERK	7788	12/01/1983	1100		20
7900	JAMES	CLERK	7698	03/12/1981	950		30
7902	FORD	ANALYST	7566	03/12/1981	3000		20
7934	MILLER	CLERK	7782	23/01/1982	1300		10

14 rows selected.

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:

```
SELECT   deptno, dname, loc
FROM dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Selecting Specific Columns

```
SELECT  ename, deptno, hiredate
FROM emp;
```

ENAME	DEPTNO	HIREDATE
SMITH	20	17/12/1980
ALLEN	30	20/02/1981
WARD	30	22/02/1981
JONES	20	02/04/1981
MARTIN	30	28/09/1981
BLAKE	30	01/05/1981
CLARK	10	09/06/1981
SCOTT	20	09/12/1982
KING	10	17/11/1981
TURNER	30	08/09/1981
ADAMS	20	12/01/1983
JAMES	30	03/12/1981
FORD	20	03/12/1981
MILLER	10	23/01/1982

14 rows selected.

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 names, department numbers and hiredates from the DEPT table.

In the SELECT clause, specifiy the columns that you want to see, in the order in which you want them to appear in the output. For example, to display deptno before ename, you use the following statement.

```
SELECT    deptno, ename

FROM emp;
```

DEPTNO	ENAME
20	SMITH
30	ALLEN

Column Heading Defaults

Default justification

-**Left:** **Date and character data**

-**Right:** **Numeric data**

Default display: **Uppercase**

-

Column Heading Defaults

Character column heading and data as well as date column heading and data are left justified within a column width. Number headings and data are right-justified.

```
SELECT ename, hiredate, sal
FROM emp
```

ENAME	HIREDATE	SAL
SMITH	17/12/1980	800
ALLEN	20/02/1981	1600
WARD	22/02/1981	1250
JONES	02/04/1981	2975
MARTIN	28/09/1981	1250
BLAKE	01/05/1981	2850
CLARK	09/06/1981	2450
SCOTT	09/12/1982	3000
KING	17/11/1981	5000
TURNER	08/09/1981	1500
ADAMS	12/01/1983	1100
JAMES	03/12/1981	950
FORD	03/12/1981	3000
MILLER	23/01/1982	1300

14 rows selected.

Character and date column headings can be truncated, but number headings can not 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

Create expressions on NUMBER and DATE data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

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 am clause of a SQL statement except the FROM clause

Using Arithmetic Operators

```
SELECT ename, sal, sal+300
FROM emp;
```

ENAME	SAL	SAL+300
SMITH	800	1100
ALLEN	1600	1900
WARD	1250	1550
JONES	2975	3275
MARTIN	1250	1550
BLAKE	2850	3150
CLARK	2450	2750
SCOTT	3000	3300
KING	5000	5300
TURNER	1500	1800
ADAMS	1100	1400
JAMES	950	1250
FORD	3000	3300
MILLER	1300	1600

14 rows selected.

Using Arithmetic Operators

The example in tin: 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

```
SELECT ename,  sal,   12 * sal + 100
FROM emp ;
```

ENAME	SAL	12*SAL+100
SMITH	800	9700
ALLEN	1600	19300
WARD	1250	15100
JONES	2975	35800
MARTIN	1250	15100
BLAKE	2850	34300
CLARK	2450	29500
SCOTT	3000	36100
KING	5000	60100
TURNER	1500	18100
ADAMS	1100	13300
JAMES	950	11500
FORD	3000	36100
MILLER	1300	15700

14 rows selected.

Operator Precedence (continued)

The example on the slide displays the name, salary, and annual compensation of employs 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.

Using Paranthesis

```
SELECT ename,  sal,    12 * (sal + 100)
FROM emp;
```

ENAME	SAL	12*(SAL+100)
SMITH	800	10800
ALLEN	1600	20400
WARD	1250	16200
JONES	2975	36900
MARTIN	1250	16200
BLAKE	2850	35400
CLARK	2450	30600
SCOTT	3000	37200
KING	5000	61200
TURNER	1500	19200
ADAMS	1100	14400
JAMES	950	12600
FORD	3000	37200
MILLER	1300	16800

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.

Defining a Null Value

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

```
SELECT ename, job, comm
FROM emp
```

ENAME	JOB	COMM
SMITH	CLERK	
ALLEN	SALESMAN	300
WARD	SALESMAN	500
JONES	MANAGER	
MARTIN	SALESMAN	1400
BLAKE	MANAGER	
CLARK	MANAGER	
SCOTT	ANALYST	
KING	PRESIDENT	
TURNER	SALESMAN	0
ADAMS	CLERK	
JAMES	CLERK	
FORD	ANALYST	
MILLER	CLERK	

14 rows selected.

Null Values If a row lacks the data value for a particular column, that value is said to be *mill,* 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 NOTNULL 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.

```
SELECT ename,  12 * sal + comm
FROM emp
WHERE ename ='KING'
```

ENAME	12*SAL+COMM
KING	

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 docs not get any commission. Because the COMM column in the arithmetic expression is null, the result is null.

For more information, see *Oracle Server SOL 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 normal!} 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 (“ “).

Using Column Aliases

SELECT ename AS Ad , sal Maaş
FROM emp;

AD	MAAŞ
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	1250
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950
FORD	3000
MILLER	1300

14 rows selected.

Column Aliases (continued)

The first example displays the name and the monthly salary of all the employees. Notice mat 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 die 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.

Column Aliases (continued)

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.

```
SELECT ename AS "Ad" , sal "Maaş "
FROM emp;
```

Ad	Maaş
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	1250
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950
FORD	3000
MILLER	1300

14 rows selected.

Column Aliases (continued)

```
SELECT ename "Adı", sal * 12 "Yıllık Ücret"
  FROM emp ;
```

Adı	Yıllık Üc
SMITH	9600
ALLEN	19200
WARD	15000
JONES	35700
MARTIN	15000
BLAKE	34200
CLARK	29400
SCOTT	36000
KING	60000
TURNER	18000
ADAMS	13200
JAMES	11400
FORD	36000
MILLER	15600

14 rows selected.

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

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 cither side of the operator are combined to make a single output column.

Using the Concatenation Operator

```
SELECT    ename || job  AS  "Employees"
   FROM   emp ;
```

Employees
SMITHCLERK
ALLENSALESMAN
WARDSALESMAN
JONESMANAGER
MARTINSALESMAN
BLAKEMANAGER
CLARKMANAGER
SCOTTANALYST
KINGPRESIDENT
TURNERSALESMAN
ADAMSCLERK
JAMESCLERK
FORDANALYST
MILLERCLERK

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, expression, or number 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 any character, expression, or number included in die 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.

Using Literal Character Strings

```
SELECT  ename || 'is  a'   ||  ' ' || job
        AS "Employee Details"
        FROM   emp ;
```

Employee Details
SMITHis a CLERK
ALLENis a SALESMAN
WARDis a SALESMAN
JONESis a MANAGER
MARTINis a SALESMAN
BLAKEis a MANAGER
CLARKis a MANAGER
SCOTTis a ANALYST
KINGis a PRESIDENT
TURNERis a SALESMAN
ADAMSis a CLERK
JAMESis a CLERK
FORDis a ANALYST
MILLERis a CLERK

14 rows selected.

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 m the SELECT statement. The spaces improve the readability of the output

Literal Character Strings (continued)

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

```
SELECT ename || ' : ' || '1' || ' Aylık Ücret = ' || sal Aylık
FROM   emp ;
```

AYLIK
SMITH : 1 Aylık Ücret = 800
ALLEN : 1 Aylık Ücret = 1600
WARD : 1 Aylık Ücret = 1250
JONES : 1 Aylık Ücret = 2975
MARTIN : 1 Aylık Ücret = 1250
BLAKE : 1 Aylık Ücret = 2850
CLARK : 1 Aylık Ücret = 2450
SCOTT : 1 Aylık Ücret = 3000
KING : 1 Aylık Ücret = 5000
TURNER : 1 Aylık Ücret = 1500
ADAMS : 1 Aylık Ücret = 1100
JAMES : 1 Aylık Ücret = 950
FORD : 1 Aylık Ücret = 3000
MILLER : 1 Aylık Ücret = 1300

14 rows selected.

Duplicate Rows

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

```
SELECT deptno
FROM   emp;
```

```
DEPTNO
10
30
10
20
…
14 rows selected.
```

Duplicate Rows

Unless you indicate otherwise. SQL*Plus displays die 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,

```
SELECT DISTINCT deptno

FROM   emp;
```

DEPTNO
30
20
10

Duplicate Rows (continued)

To eliminate duplicate rows in the result, include the DISTINCT keyword m 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

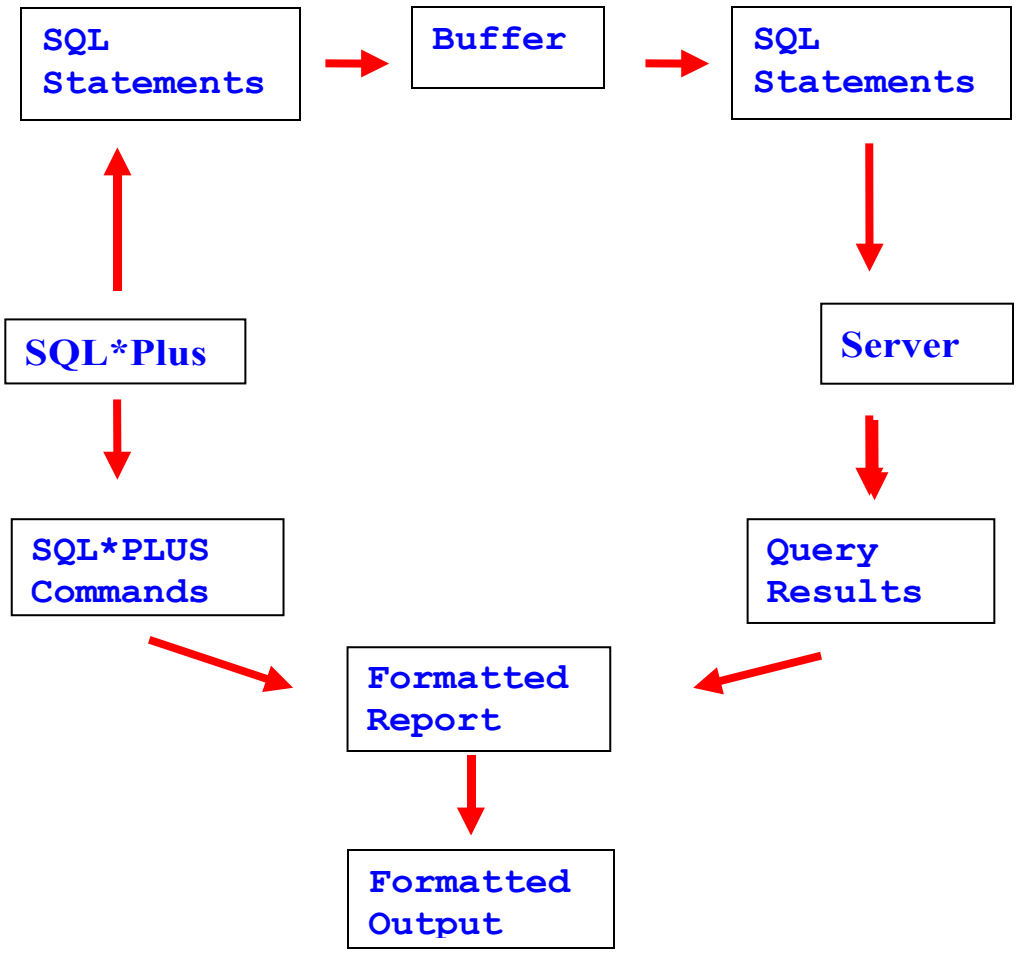
```
SELECT DISTINCT deptno, job

FROM   emp;
```

DEPTNO	JOB
20	CLERK
30	SALESMAN
20	MANAGER
30	CLERK
10	PRESIDENT
30	MANAGER
10	CLERK
10	MANAGER
20	ANALYST

9 rows selected.

SQL and SQL*Plus

Interaction



SQL and SQL*Plus

SOL 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 *SOL buffer* and remains there until you enter a new statement.

*SOL*Plus* is an Oracle tool that recognizes and submits SQL statements to the Oracle Server or 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.

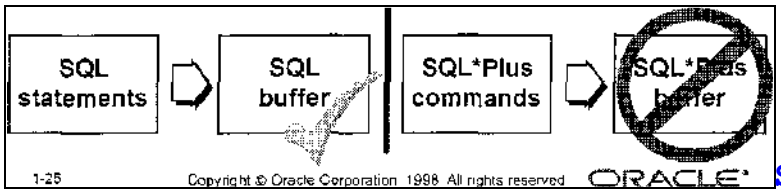
Features of SQL'Plus

- 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	SQL*Plus
A language	An environment
ANSI standard	Oracle proprietary
Keyword cannot be abbreviated	Keywords can be abbreviated
Statements manipulate data and table definitions in the database	Commands do not allow manipulation of values in the databasese

SQL and SQL*Plus (continued)



The following table compares SQL and SQL*Plus:

SQL	SQL*Plus
Is a language for communicating Server to access data	Recognizes the server
	Is the Oracle
	Is entered one line at a time: not stored in the SQL buffer

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 ail environment tn \\hich *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 repetitue use in the future

SQL*Plus commands can be divided into the follouing main categories.

Category	Purpose
Environment	Affects the general behavior of SQL statements for the session
Format	Formats query results
File manipulation	Saves, loads, and runs script files
Execution	Sends SQL statements from SQL buffer to OracieS Server
Edit	Modifies SQL statements in the buffer
Interaction	Alkms 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 SQL*Plus environment, and display column definitions

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.

```
DESC emp ;
```

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

```
DESC dept;
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

```
DESC salgrade ;
```

Name	Null?	Type
GRADE		NUMBER
LOSAL		NUMBER
HISAL		NUMBER

Displaying Table Structure

Displaying **Table** Structure (continued)

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 m the following table:

Datatype	Description
NUMBER(p,s)	Number value having a maximum number of digits p, the number of digits to the right of the decimal point s
VARCHAR2(s)	Variable-length character value of maximum size s
DATE	Date and time value between January 1. 4712 B.C. and December 31.9999 A.D
CHAR(s)	Fixed-length character value of size s

SQL*Plus Editing Commands

A[PPEND] *text*

C[HANGE] /old /new

C[HANGE] text

CL[EAR] BUFF[ER]

DEL

DEL n

DEL m n

SQL'Plus Editing Commands

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

Command	Description	
A[PPEND] *text*	Adds text to the end of the current line	
C[HANGE]/ci/e//>wM-	Changes *old* text to m'w in the current line	
C[HANGE]/text /	Deletes *text* from the current line	
CL[EAR] BUFF[ER]	Deletes all lines from the SQL buffer	
DEL	Deletes current line	

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)

Command	Description
I[NPU T]	Inserts an indefinite number of lines
I[NPUT] t*ext*	Inserts a line consisting of rf.vr
I.[IST]	Lists all lines in the SOL buffer
L[IST] *n*	Lists one line (specified bv *n)*
L[IST] *m n*	Lists a ranae of lines *(m* to *n)*
R[UNl	Displavs and runs the current SQL statement m the buffer
n	Specifies the line to make the current line
n text	Replaces line *n* with *text*
0 *text*	Inserts a line before line 1

You can enter onh one SQL*PIus 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 (-).

SQL*Plus File Commands

SAVE *filename*

GET *filename*

START *filename*

@*filename*

EDIT *filename*

SPOOL *filename*

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

Command	Description
SAVfEJ *filename* [.ext] [REP[LACLJAPP[END]]	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 - sql.
GET filename.ext]	Writes the contents of a previously sa\ed file to the SQL buffer. The default extension for the filename is - sql.
STA[RT] *filename[ext]*	Runs a previously saved command file.
@ filename	Runs a previously sa\ed command file (same as START).
ED [IT]	Invokes the editor and saves the buffer contents to a file named
ED[IT] *[filename[ext]]*	Invokes the editor to edit contents to a saved file
SPO[OL] [filename[ext]] | OFF|OUT	Stores query results in a file. OFF closes the spool file and sends the file results to the system printer
EXIT	Leaves SQL*PLUS

Summary

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

Use SQL*Plus as an environment to:

- **Execute SQL statements**
- **Edit SQL statements**

SELECT Statement

In this lesson, you have learned about retrieving data from a database table with the SELECT statement. The syntax is as follows:

SELECT [DISTINCT] { * , column_name [alias], …}

FROM table_name ;

In this syntax,the roles of key words are:

SELECT	is a list of at least one column
DISTINCT	suppresses duplicates
*	selects all columns
Column_name	selects the named column
Alias	gives selected column a different heading
FROM	specifies the table containing the columns

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.

Practice 1

1.Initiate a SQL*Plus session using the user ID and password provided by the instructor.

2.SQL*Plus commands access the database.

True/False

3.Will the SELECT statement execute successfully?

True/False

```
SQL> SELECT ename, job, sal Salary
     FROM emp ;
```

4.Will the SELECT statement execute successfully?

True/False

```
SQL> SELECT *
     FROM salgrade
```

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

```
SQL>  SELECT empno, ename
      Salary x 12 ANNUAL SALARY
      FROM emp;
```

6. Show the structure of the DEPT table. Select all data from the DEPT table.

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Practice 1 (continued)

7. Show the structure of the EMP table. Create a query to display the name, job. hire date, and employee number for each employee, with employee number appearing first. Save your SQL statement to a file named *plq7.sql.*

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		VARCHAR2(10)
JOB		VARCHAR2(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

8. Run your query in the file plq7.sql.

EMPNO	ENAME	JOB	HIREDATE
7369	SMITH	CLERK	17/12/1980
7499	ALLEN	SALESMAN	20/02/1981
7521	WARD	SALESMAN	22/02/1981
7566	JONES	MANAGER	02/04/1981
7654	MARTIN	SALESMAN	28/09/1981
7698	BLAKE	MANAGER	01/05/1981
7782	CLARK	MANAGER	09/06/1981
7788	SCOTT	ANALYST	09/12/1982
7839	KING	PRESIDENT	17/11/1981
7844	TURNER	SALESMAN	08/09/1981
7876	ADAMS	CLERK	12/01/1983
7900	JAMES	CLERK	03/12/1981
7902	FORD	ANALYST	03/12/1981
7934	MILLER	CLERK	23/01/1982

14 rows selected.

Practice 1 (continued)

9. Create a query to display unique jobs from the EMP table.

JOB
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

If you have time, complete the following exercises:

10. *Loadplc/7.sql* into the SQL buffer. Name the column headings Emp #, Employee, Job, and Hire Date, respectively. Rerun your query.

Emp # Employee	Job	Hire Date
7339 KING	PRESIDENT	17-NOV-81
7698 BLAKE	MANAGER	01 -MAY- 81
7782 CLARK	MANAGER	09-JUN-81
7566 JONES	MANAGER	02 -APR- 81
7654 MARTIN	SALESMAN	28-SEP-81
7499 ALLEN	SALESMAN	20-FEB-81
7844 TURNER	SALESMAN	08-SEP-81
7900 JAMES	CLERK	03-DEC-81
7521 WARD	SALESMAN	22-FEB-81
7902 FORD	ANALYST	03-DEC-81
7369 SMITH	CLERK	17-DEC-80
7788 SCOTT	ANALYST	09-DEC-82
7876 ADAMS	CLERK	12 -JAN- 8 3
7934 MILLER	CLERK	2 3- JAN- 8 2

14 rows selected

Practice 1 (continued)

11. Display the name concatenated with the job. separated by a comma and space, and name the column Employee and Title.

```
Employee and Title

KING, PRESIDENT BLAKE, MANAGER
CLARK, MANAGER JONES, MANAGER
MARTIN, SALESMAN ALLEN,
SALESMAN TURNER, SALESMAN
JAMES, CLERK WARD, SALESMAN
FORD, ANALYST SMITH, CLERK
SCOTT, ANALYST ADAMS, CLERK-
MILLER, CLERK 14 rows selected.
```

If you want extra challenge, complete the following exercise:

12. Create a query to display all the data from the EMP table. Separate each column *by* a comma. Name the column THE OUTPUT.

```
THE  OUTPUT

7339,KING,PRESIDENT, ,17-NOV-81,5000, , 10
7693,BLAKE,MANAGER,7839,01-MAY-31, 2350, ,30
7732,CLARK,MANAGER,7339,09-JUN-31, 2450, , 10
7566,JONES, MANAGER,7339,02-APR-31, 2975, , 20
7654,MARTIN,SALESMAN,7698,28-SEP-81, 1250, 1400,30
7499,ALLEN,SALESMAN,7698,20-FEB-81, 1600,300,30
7344,TURNER, SALESMAN,7698,08-SEP-81, 1500, 0,30
7900,JAMES,CLERK,7698,03-DEC-81,950,,30
7521,WARD,SALESMAN,7698,22-FEB-81,1250, 500,30
7902,FORD,ANALYST,7566,03-DEC-81, 3000, , 20
7369,SMITH,CLERK,7902,17-DEC-80,800,,20
7788,SCOTT,ANALYST,7566,09-DEC-82,3000,,20
```