**To create table:**

Syntax:

SQL>create table <table name>

(<col1><type>, <col2><type> …);

Note: the character “;” is terminator for SQL Statement.

Ex:

SQL>create table Emp\_det

(empno number(4), ename varchar2(15),

job varchar2(15), mgr number(4),

Hiredate timestamp, sal number (6, 2),

comm Number(6,2), ephoto blob,

Deptno number(2));

SQL>create table student\_det

(StudNo Number (4), Fname Varchar2(15),

LnameVarchar2(15), DOB Date,

DOJ Date, Fees Number(6,2),

Gender Char);

**Inserting Data in Table:**

• A row is inserted into a table by using INSERT command.

• Rows can insert into a

* + Table
  + Views Base Table.
  + A partition of a Table.
  + A Sub Partition of a Composite Partition Table.
  + An Object Table.
  + An Object View's Base Table.

Inserting of data into a table can be executed in two ways.

• Conventional INSERT.

• Direct-Path INSERT.

In conventional Insert statement, Oracle reuses free space in the table into which the data in being insert and maintains Referential Integrity Constraints.

In Direct-Path Insert, Oracle append the insert data after existing data in the Table, the free space is not reused.

Syntax: Sql>INSERT INTO <table name> [list of columns]

VALUES (list of values);

Sql> INSERT Into Student\_Det Values

(

1001,

'THOMAS',

'SIEBEL',

'30-DEC-81',

'02-JAN-91',

30000,

‘M’

);

• In this case the values should be provided to all the columns that exist inside the table.

• The order of values declared in the Values clause should follow the original order of the columns in the table.

• The Character and Date type data should be declared in single Quotes.

• Number information can be applied normally.

**Inserting Data into Required Columns:**

Sql>Insert Into Student\_Det

( StudNo, Fname, Lname , DOJ ) Values

( 1002, ‘JAN’, ‘SMITH’, '12-JAN-09' );

• In this case the Order of columns declared in insert need not be the same as that of the original table order.

• The data values in the values clause should match with that of INSERT list.

• The columns not supplied with data are filled with NULL values, Until the NOT NULL constraint is declared.

**Inserting NULL values into Table:**

• Null represent information that is not available.

• Null Value is

* Unknown value
* Undefined value
* Not equal to 0 or blank space represented with 'NULL' keyword.
* Not allowed arithmetic, relational operation on Null, If performed it return Null only.
* RDBMS must support NULL Values.
* NULL values can be inserted in two ways.

• Implicit🡪Omit the columns from list oracle supplies NULL values.

• Explicit🡪Specify the NULL keyword.

Sql>Insert Into Student\_Det

(

StudNo,

Fname,

Lname,

DOB,

DOJ,

Fees,

Gender

)

Values

(

1003, ‘CRANIA’, NULL,

'12-JAN-81’, NULL, 30000, ‘M’

);

**Inserting special Values into Table:**

**SYSDATE FUNCTION**

• SYSDATE is Pseudo-column.

• This function return the current date& time.

**USER FUNCTION:**

• This function returns the user name of the user who has logged in.

**Example:**

Sql>CREATE TABLE Student

(Stuid Number(4),

Name Clob,

DOB Date,

DOJ Date,

Fees Number (8, 2),

Gender char);

**Example:**

Sql>INSERT INTO student VALUES (1001, user, '01-JAN-85', sysdate, 2000,’M’);

**Substitution Variables:**

• These variables are used to stored values temporarily

• The-values can be stored temporarily through

🡪Single Ampersand (&)

🡪Double Ampersand (&&)

🡪Prompt and ACCEPT Commands

• The Single Ampersand substitution variable applies for each instance when the SQL statement is create or execute.

• The Double Ampersand substitution variable applies for all instances until that SQL statement is existing.

**USING SINGLE AMPERSAND SUBSTITUTION VARIABLE:**

Sql> INSERT INTO dept VALUES(&Dno,'&Dname','&Loc');

Sql> INSERT INTO emp(Empno,Ename,Hiredate,Deptno) VALUES(&Eno,’&Ename’,’&Hiredate’,&dno);

Note: To run command is used for executing a previous command...

>run

or

>/

**USING DOUBLE AMPERSAND SUBSTITUTION VARIABLE:**

Sql>INSERT INTO student VALUES(&Sid,’&Name',’&Dob’,&Doj,&&fee,’&Gender’);

**DEFINING CUSTOMIED PROMPTS:**

Sql>ACCEPT Deptno prompt 'Please Enter the deptnumber:'

Sql>ACCEPT Dname prompt 'Please Enter the Deptname:'

Sql>ACCEPT loc prompt 'Please Enter the Location:'

Sql>INSERT INTO dept VALUES(&Deptno,&Dname,&loc);

**CREATING AN .LST FILE:**

• The List file is used to save all information which performed in SQL \* Plus.

**Navigation:**

1) In SQL \* Plus

File->Spool->Spool File

2) Give the file name

3) Performed all tasks.

4) Now Spool Off.

**To create spool file:**

Sql> spool D:/oraclepractice/d1;

**To off spool file :**

sql> spool off;

**TO REUSE/EDIT SAME SPOOL FILE**

SQL> SPOOL D:/SQL1 APPEND

**CREATING AN SQL SCRIPT FILE:**

• The SAVE Command is used to store the current contents of the SQL Buffer.

• Steps:

1. At SQL prompt type the full name of the path where the file has to be created.

2. Give the name of the file with .sql extension.

3. If the has to be replaced with the same existing name then use

REPLACE ALL clauses.

**Example:**

Sql>SAVE E:\nitdir\ins

Sql> SAVE E:\nitdir\ins REPLACE

**To Get Script:**

Get used to display the script file.

Sql>Get E:\nitdir\ins

**To Run Script:**

Run is used to display the script and run the file.

Sql>RUN E:\nitdir\ins

To simple the script files.

Sql>@ E:\nitdir\ins

**To Open the Script file in editor.**

Sql>ed E:\nitdir\ins

**Data Retrieval Languages(DQL/DRL):**

**QUERY**: It is an operation that retrieves data from one or more table or view.

**SELECT Statement:**

• Select used to retrieve data from one or more than table, view and object tables.

• It is used for read only purpose.

• The select is the most frequently used command as access to information is needed all the time.

**PREREQUISTIES:**

• The user must have the SELECT privileges on the specified

Object.

**CAPABILTTFS OF SQL SELECT STATEMENT:**

The different types SELECT criteria.

• SELECT:

O-> It chooses the rows in a table that are expected to return by a query.

• PROJECTION:

O->It chooses the columns in a table that are expected to return by a query.

• JOIN:

O-> It chooses the data in from one or more numbers of tables.

**Data Retrieving Language:**

Syntax:

Sql>SELECT \*| {[DISTINCT/UNIQUE] column | expression

[Alias],…..} FROM table.

• SELECT ->identifies what columns, specifies a list of

column(one/more)

• FROM ->identifies which table

• DISTINCT ->Suppress Duplicates.

• \* ->Select all Columns

• Columns/ Expression ->Select the Named Columns or the expression.

• Alias ->Gives select columns different headings

**WRITING SQL STATEMENTS TO SELECT DATA FROM TABLES.**

**The sample table.**

1. DEPT

Column Name Data Type

----------------------------------------- -------------- --------

DEPTNO NUMBER (2)

DNAME VARCHAR2 (14)

LOC VARCHAR2 (13)

1. EMP

Column Name Data Type

-------------------------------------- ------------- ------------

EMPNO NUMBER (4)

ENAME VARCHAR2 (10)

JOB VARCHAR2 (9)

MGR NUMBER (4)

HIREDATE DATE

SAL NUMBER (7, 2)

COMM NUMBER (7, 2)

DEPTNO NUMBER (2)

1. SALGRADE

Column Name Data Type

---------------------------------------- ---------- -----------

GRADE NUMBER

LOSAL NUMBER

HISAL NUMBER

**Dept:**

SQL> Select \*From Dept;

DEPTNO DNAME LOC

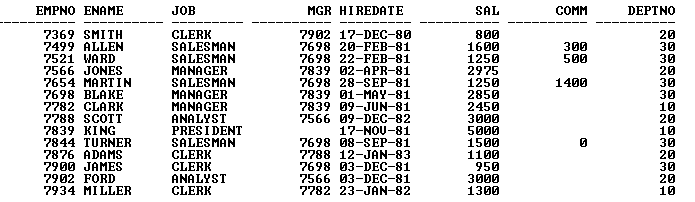
----------------- ------------------- -------------

10 ACCOUNTING NEW YORK.

20 RESEARCH DALLAS

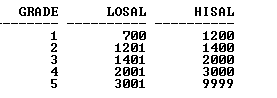
30 SALES CHICAGO

40 OPERATIONS BOSTON

SQL> SELECT \* FROM EMP;

14 rows selected.

SQL> Select \*From Salgrade;



**Retrieving Data from All Columns:**

Sql>SELECT \*FROM Emp;

Sql>SELECT \*FROM Dept;

Sql>SELECT \*FROM Salgrade;

• In this the ‘\*’ is a projection operator.

• It project data from all the columns existing in the table with all records.

• The data is displayed in a table format.

**Retrieving Data from Specific Columns:**

Sql> SELECT empno, ename, job, sal FROM Emp;

Sql>SELECT loc, dname, deptno FROM Dept;

Sql> SELECT Hisal,Losal,Grade FROM Salgrade;

• The columns names need not be in the same order as table.

• The columns should be separated using comma.

• The column names can be separated into different lines within the SQL Buffer.

• The casing of column names is not important.

**Column Heading Default:**

• The default justification of the data after it is retrieved from the table is...

LEFT ->Date and Character

RIGHTS ->Number Data

• The default Columns Heading display in UPPER case.

**Applying Arithmetic Operations in Select Statements:**

• Arithmetic Expressions can be implemented through SELECT statement.

• Arithmetic Expressions can be implemented to

o Modify the way the data is displayed.

o Perform calculations.

**An Arithmetic Expression can contain:**

• Simple columns names

• Constant numeric values

• Arithmetic operators

**ARITHMETIC OPERATORS:**

• The Arithmetic operations can be used to create expressions on NUMBER and DATE data.

• The Arithmetic operators supported are....

o + ->Addition

o - ->Subtraction

o \* -> Multiplication

o / -> division.

• The Arithmetic operators can be used in any clause of a SQL statement, except the FROM clause.

• SQL \* Plus ignores Blank Spaces before and after the Arithmetic operator.

Sql> Select empno,ename,sal,sal+500 FROM Emp;

Sql> Select empno,ename,sal,sal-1000 FROM Emp;

**Operator Precedence:**

• Multiplication and Division take priority over addition and subtractions(\*/+-)

• Operators of the same priority are evaluated from left to right

• To prioritize evaluation and to increase clarity parenthesis can be implemented.

Sql> Select empno,ename,sal,12\*sal + 100 from Emp;

Sql>Select empno,ename,sal,(12\*sal)+100 from Emp;

Sql>Select empno,ename,sal,12\*(sal+500) from Emp;

**Handling Null Values:**

NULL:

• Unknown value

• Undefined value

• Not equal to 0 or blank space.

• If a row lacks the data for a particular column than that value is said to be NULL or to contain NULL.

• It represented with NULL keyword

• No operations allowed on Null, (If performed any arithmetic operations it return null only)

• RDBMS must support NULL Values.

Sql>Select ename,job,sal,comm from emp;

Sql> Select ename,Job,sal,comm,12\*sal+comm from emp;

**General Function:**

This function work with any data type and pertain to using null values.

**NVL Function:**

• The NVL function is used to convert a NULL value to an actual value.

Syntax

NVL(Exprl,Expr2)

Expr1: is the source value or expression that may contain NULL.

Expr2: is the target value for converting NULL.

NVL Conversions for Various Data Types:

• NVL(Number\_column, 0)

• NVL(date\_column, '01-JAN-09')

• NVL(Character\_column,’Unavailable')

Note : If Expr1 is Character data then Expr2 may any Data type.

Sql> Select NVL(100,200) from dual;

Sql> Select NVL(null,200) from dual;

Sql> Select ename, sal, comm,sal+NVL(comm,0) from emp;

Sql> Select ename, sal, comm, (sal\*12)+NVL(comm,0) from emp;

Sql> Select ename, sal, comm, (sal+5OO)+NVL(comm,0) from emp;

**NVL2 Function:**

Syntax:

NVL2(exprl,expr2,expr3)

• If exprl is not null,NVL2 returns expr2.if exprl is null, NVL2 returns expr3.

• Exprl may any data type.

• The data type of the return value is always the same as the data type of expr2, unless Expr2 is character data.

Example:-

1) select nvl2(comm,0,1000) from emp;

2) select sal,comm,sal+nvl2(comm, 100,1000) from emp;

>select ename,sal,comm, nvl2(comm,’sal+comm’,’sal’) income from emp where deptno in(10,30);

NULL IF Function:

Syntax:

NULLIF(expr1,expr2)

* Compares two expressions and returns null if they are equal, or the first if there are not equal.

Example:

>SELECT NULLIF (100,200) from dual;

>SELECT ENAME,LENGTH(ENAME) "expr1", JOB,LENGTH(JOB) "expr2", NULLIF(LENGTH(ENAME),LENGTH(JOB)) result

from emp;

**Note:**

o The data type of the expr1 is same as the data type of expr2.

o The NULLIF function is logically equivalent to CASE expression.

**COALESCE:**

• It return first non-null expression in the expression list.

Sql> SELECT coalesce(100,600,200) FROM dual;

Sql> SELECT coalesce(null,600,200) FROM dual;

Sql>SELECT Ename,Deptno,COALESCE(COMM,SAL,10) COMM FROM EMP

**Defining a Column Alias:**

• An Alias is an alternate name given for any Oracle Object.

• Aliases in Oracle are of two types.

\*Column Alias \*Table Alias

• Column alias rename a Column Heading.

• Alias Headings appear in uppercase By default.

• Specify the alias after the column in the SELECT list using a space as a separator.

• AS keyword between the column name and alias is optional.

• The Alias contains spaces or special characters (such as # or $), or is case sensitive, enclose the alias in double quotation marks (“ “)

• An alias cannot be used, anywhere in the SELECT list for operation purpose.

• An alias effectively renames the SELECT list item for the duration of the Query.

Sql> SELECT Empno EmpNumber,

Ename EmpName,

Sal "EmpSalary"

Job Designation

FROM Emp;

Sql> SELECT Grade AS "SalGrade", Losal ”Lower Salary Range”,

Hisal "High Salary Range"

From Salgrade;

Sql> Select empno "EmpNumber", Sal "Basic",Sal\*0.25 HRA, Sal\*0.20 DA,Sal\*0.15 "Pf",

Sal+Sal\*0.25+Sal\*0.20-Sal\*0.15 "Gross" FROM emp;

**Concatenation Operation:**

• The Concatenates operator concatenate columns or character strings or expressions or constant to other values or columns.

• Is represented by two vertical bars (| |).

• The resultant column that is a character.

Sql> SELECT empno||ename FROM Emp;

Sql> Select 'The Basic Salary of '||Ename|| 'is Rs '||Sal Employee From Emp;

Sql>Select Empno| |Ename| |' ,Designation is '| |job "Employees Information " From Emp;

**LITERALS IN ORACLE:**

• A Literal and Constant value are synonyms to one another and refer to a fixed data value.

• The types of Literals recognized by Oracle are

• Text Literals

• Number Literals

• Interval Literals

**Text Literals:**

• It specifies a text or character literal.

• It is used to specify values whenever 'text' or CHAR appear in

\*Expression \*Condition \*SQL Function \* SQL Statements.

• It should be enclosed in single quotes.

• A text literal can have a maximum length of 4000 Bytes.

Example:

'Employee Information'

'Manager's Specification'

**Using Literal Character Strings:**

* A literal value is a character, a number, or a date that is included in the SELECT list.
* A literal value not a column name or a column alias.
* A literal is printed for each row returned, that is retrieved by the SELECT statement.
* Literal strings of free-format text can be included in the query.
* A free-format text treated the same as a column in the SELECT list.
* Date and character literal must be enclosed within the single quotation marks ‘‘.
* Literal increase the readability of the output.

Sql>Select Ename | |':'| | 'Month Salary='| | Sal As Salaries From Emp;

Sql>Select 'The Designation of '| | Ename | |' is '| | job As Designation From Emp;

Sql> Select The employee name is: '| |Ename| |’and’

| |'Designation :'| |job from Emp;

Sql> Select 'The Annual Salary of '||Ename||' is '||Sal\*12 As

Annual\_Salary From Emp;

Sql> Select Dname| |'Department is Located at'| |Loc From Dept ;

Sql> Select Ename| |' Joined the Organization on ' | | Hiredate From Emp;

Sql> Select Ename| |' Works in Department Number'||Deptno

| |‘ as ' | |job From Emp;

Sql> Select ‘The Employee Name is: ‘||Ename||', Designation is '| |Job FROM Emp;

Sql> SELECT The Gross salary of | |Ename| |'is’

| |(Sal+Sal\*.25+Sal\*.20-Sal\*.15) FROM Emp;

**Duplicate Rows:**

(By using unique or distinct, we can remove duplicate values in report)

• Unless we indicate otherwise, SQL\*Plus displays the results of a query without eliminating duplicate rows.

• To eliminate duplicate rows in the result, the DISTINCT keyword used.

• We 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.

Sql>Select DISTINCT Job From Emp;

Sql> Select DISTINCT Job,Deptno From Emp;

Sql>Select DISTINCT Deptno,Job From Emp;