Exercise 1: Create and manipulate identified tables/relations for the University application.

AIM : To identify the scheme for the university application and CREATE, ALTER and DROP and INSERTING rows into a table. (use constraints while Creating tables) in oracle database using SQL and to manipulate the tables using the 'ALTER' command.

Students (sid: interger , name: string, branch:string,dob:date, cgpa: real (Cgpa min:4 to Max:10)) Give the appropriate integrity constraints.

Faculty (fid: integer, fname: string , fdesignation: string , sal: real salary(Min:2000,Max:20,000) Give the appropriate integrity constraints.

Courses (cid: string, cname: string,dept:string) Give the appropriate integrity constraints.

Dept (did:integer,dname:string,dloc:string,fid:integer) Give the appropriate integrity constraints.

Insert the appropriate values into the student Table: your roll num, name,branch,dob,cgpa

Insert the appropriate values into the faculty Table (e.g 20,'smith','professor')

Insert the appropriate values into the courses Table (e.g 320 Data Structures CSE)

Insert the appropriate values into the dept Table (e.g 20, CSE, HYD)

Description:

Create a Schema

Creating a schema in Oracle, can at first, appear to be a little confusing. You might think that the CREATE SCHEMA statement would create your schema, but that is not the case. The CREATE SCHEMA statement is used only to create objects (ie: tables, views, etc) in your schema in a single SQL statement, but does not actually create the schema itself.

- Step 1 - Create a new user in Oracle
- Step 2 - Assign SYSTEM privileges to new user in Oracle
- Step 3 - Create objects in the schema
- Step 4 - Grant Object Privileges
- Step 1 - Create a new user in Oracle

CREATE USER statement

The CREATE USER statement creates a database account that allows you to log into the Oracle database.

Syntax

The syntax for the CREATE USER statement in Oracle/PLSQL is:

CREATE USER <user_name> IDENTIFIED BY <password>

Parameters or Arguments:

<user_name> : The name of the database account that you wish to create.

<password> : To assign a password user can log into the Oracle database.

Step 2 - Assign SYSTEM privileges to new user in Oracle

GRANT ALL PRIVILEGES TO<user name/schema name>;

Parameters or Arguments:

ALL

Example:

If you wanted to execute a simple CREATE USER statement that creates a new user and assigns a password, you could do the following:

SQL> CREATE USER cseb IDENTIFIED BY lendi;

User created.

SQL> GRANT ALL PRIVILEGES TO cseb;

Grant succeeded.

<u>DDL -- CREATE, ALTER, DROP, TRUNCATE, RENAME</u>

CREATE TABLE Statement

The Oracle CREATE TABLE statement allows you to create and define a table.

Syntax

The syntax for the CREATE TABLE statement in Oracle/PLSQL is:

```
CREATE TABLE table_name (   column1 datatype [ NULL | NOT NULL ],
                            column2 datatype [ NULL | NOT NULL ],
                              ...
                              ...
                            column_n datatype [ NULL | NOT NULL ]
                    );
```

(OR)

SQL create table with constraint syntax:

CREATE TABLE *table_name* (*column_name1* *data_type*(*size*) *constraint_name*, *column_name2* *data_type*(*size*) *constraint_name*, *column_name3* *data_type*(*size*) *constraint_name*,....);

Parameters or Arguments:

<*Table_name*>: The name of the table that you wish to create.

<column1, column2, ... column_n>: The columns that you wish to create in the table.

Each column must have a datatype.

The column should either be defined as "null" or "not null" and if this value is left blank, the database assumes "null" as the default.

ALTER TABLE Statement

The Oracle ALTER TABLE statement is used to add, modify, or drop/delete columns in a table. The Oracle ALTER TABLE statement is also used to rename a table.

Add column in table

Syntax

To ADD A COLUMN in a table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table_name ADD column_name column-definition;

Add multiple columns in table

Syntax

To ADD MULTIPLE COLUMNS to an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table_name ADD (column_1 column-definition, column_2 column-definition, ... column_n column_definition);

Modify column in table

Syntax

To MODIFY A COLUMN in an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table_name MODIFY column_name column_type;

Modify Multiple columns in table

Syntax

To MODIFY MULTIPLE COLUMNS in an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table_name MODIFY(column_1 column_type, column_2 column_type, ... column_n column_type);

Drop column in table

Syntax

To DROP A COLUMN in an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table_name DROP COLUMN column_name;

TRUNCATE TABLE Statement:

The TRUNCATE TABLE statement is used to remove all records from a table in Oracle. It performs the same function as a DELETE statement without a WHERE clause.

Warning: If you truncate a table, the TRUNCATE TABLE statement cannot be rolled back.

Syntax

The syntax for the TRUNCATE TABLE statement in Oracle/PLSQL is:

```
TRUNCATE TABLE [schema_name.]table_name
  [ PRESERVE MATERIALIZED VIEW LOG | PURGE MATERIALIZED VIEW LOG ]
  [ DROP STORAGE | REUSE STORAGE ] ;
```

Parameters or Arguments

schema_name

Optional. If specified, it is the name of the schema that the table belongs to.

table_name

The table that you wish to truncate.

preserve materialized view log

Optional. If specified, the materialized view log will be preserved when the table is truncated. This is the default behavior.

purge materialized view log

Optional. If specified, the materialized view log will be purged when the table is truncated.

Drop Storage

Optional. If specified, all storage for the truncated rows will be deallocated, except the space that has been allocated by MINEXTENTS. This is the default behavior.

Reuse Storage

Optional. If specified, all storage for the truncated rows will remain allocated to the table.

Rename column in table:

Syntax

Starting in Oracle 9i Release 2, you can now rename a column.

To RENAME A COLUMN in an existing table, the Oracle ALTER TABLE syntax is:

ALTER TABLE table_name   RENAME COLUMN old_name to new_name;

Rename table

Syntax

To RENAME A TABLE, the Oracle ALTER TABLE syntax is:

ALTER TABLE table_name   RENAME TO new_table_name;

DML :INSERT, UPDATE, DELETE

INSERT Statement: INSERT INTO/VALUES

The Oracle INSERT statement is used to insert a single record or multiple records into a table in Oracle. Use to Add Rows to existing table.INSERT This will be used to insert the records into table. We have two methods to insert.

- By value method
- By address method

Using Value Method

Syntax

INSERT INTO <*table_name*> VALUES (*value1, value2, value3 …. Value n*);

Note: To insert a new record again you have to type entire insert command, if there are lot of records this will be difficult. This will be avoided by using address method.

Using Address Method

The syntax for the Oracle INSERT statement when inserting a single record using the VALUES keyword is:

Syntax: INSERT INTO <table_name> VALUES *(&col1, &col2, &col3 …. &coln*);

This will prompt you for the values but for every insert you have to use forward slash.

INSERTING DATA INTO SPECIFIED COLUMNS USING VALUE METHOD

Syntax:

insert into <*table_name*)(*col1, col2, col3 … Coln*) values (*value1*, *value2*, *value3 …. Valuen*);

INSERTING DATA INTO SPECIFIED COLUMNS USING ADDRESS METHOD

Syntax: insert into <*table_name*)(*col1, col2, col3 … coln*) values *(&col1, &col2 ….&coln);* This will prompt you for the values but for every insert you have to use forward slash.

UPDATE Statement: UPDATE/SET/WHERE

The Oracle UPDATE statement is used to update existing records in a table in an Oracle database. There are 2 syntaxes for an update query in Oracle depending on whether you are performing a traditional update or updating one table with data from another table.Use to Edit Existing Rows in tables.

Syntax

The syntax for the UPDATE statement when updating one table in Oracle/PLSQL is:

UPDATE table SET column1 = expression1,column2 = expression2, ... column_n = expression_n WHERE conditions;

DELETE Statement: DELETE FROM/WHERE

The Oracle DELETE statement is used to delete a single record or multiple records from a table in Oracle. Use the DELETE statement to delete the rows from existing tables which are in your schema or if you have DELETE privilege on them.This can be used to delete the table data temporarily.

Syntax

The syntax for the DELETE statement in Oracle/PLSQL is:

DELETE FROM table WHERE conditions;

Parameters or Arguments

Table:The table that you wish to delete records from.

Conditions: The conditions that must be me t for the records to be deleted.

Note: You do not need to list fields in the Oracle DELETE statement since you are deleting the entire row from the table.

DQL : Data Query Language: SELECT/FROM/WHERE

SELECT Statement

The Oracle SELECT statement is used to retrieve records from one or more tables in an Oracle database.

SELECT: retrieve records from one or more table

Syntax

The syntax for the SELECT statement in Oracle/PLSQL is:

SELECT *expressions* FROM *tables* WHERE *conditions;*

Parameters or Arguments

Expressions: The columns or calculations that you wish to retrieve.

Tables: The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

Conditions: The conditions that must be met for the records to be selected.

SELECT − This is one of the fundamental query command of SQL. It selects the attributes based on the condition described by WHERE clause.

FROM − This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.

WHERE − This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.

There are three ways we can retrieve data from a table:

- Retrieve one column
- Retrieve multiple columns
- Retrieve all columns

<u>Sample Queries:-</u>

1. Display the contents of the faculty relation. Ans. (SQl query): Select * from faculty;
2. Write a SQL query describes the structures of all tables.
3. Display the contents of the dept relation.
4. Display all the faculty names.
5. Display all the department names.
6. Display all the faculty names along with their designations.

7. Display the details of faculty names whose salaries are greater than 5000.
8. To add new column CITY to the student relation and enter the city values
9. Modify column in table (increase) the data types and size of course name in course relation
10. Delete student details whose cgpa is less than 5
11. Change the name as sname in student relation
12. Drop branch in student relation
13. Display the salaries of employees without repetitions.
14. Display the names and designation of faculty whose designation is associate professor and whose salary is greater than 10000.

Viva Questions:

1. What is the difference between SQL and PL/SQL?
2. What are various DDL commands in SQL? Give brief description of their purposes.
3. What are various DML commands in SQL? Give brief description of their purposes.
4. What are various DCL commands in SQL? Give brief description of their purposes.
5. Difference between DBMS and RDBMS.
6. What is a key difference between Drop, Truncate and Delete?
7. What is a PRIMARY KEY?
8. What is a FOREIGN KEY?
9. What is a UNIQUE KEY?
10. What is the difference between UNIQUE and PRIMARY KEY?

****

Exercise 2: SET OPERATORS AND COMPARASION OPERATORS

AIM : Queries (along with sub Queries) using UNION,UION ALL,INTERSECT,ANY, ALL, IN,NOT IN , EXISTS, NOT EXISTS.

Description: SQL - Sub Queries

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

There are a few rules that subqueries must follow:

Subqueries must be enclosed within parentheses.

A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.

An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a subquery.

Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.

The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.

A subquery cannot be immediately enclosed in a set function.

The BETWEEN operator cannot be used with a subquery; however, the BETWEEN operator can be used within the subquery.

Set Operation in SQL

SQL supports few Set operations to be performed on table data. These are used to get meaningful results from data, under different special conditions

UNION,UNION ALL ,INTERSET,IN ,NOT IN,EXISTS, NOT EXISTS.

UNION clause/operator

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows. Each SELECT must have the same number of columns selected, the same number of column expressions, the same data type, and have them in the same order, but they do not have to be the same length.

Syntax

The syntax for the UNION operator in Oracle/PLSQL is:

SELECT expression1, expression2, ... expression_n FROM tables WHERE conditions **UNION** *SELECT expression1, expression2, ... expression_n FROM tables*
WHERE conditions;

Parameters or Arguments

expression1, expression2, expression_n

The columns or calculations that you wish to retrieve.

Tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

Conditions

The conditions that must be met for the records to be selected.

UNION ALL OPERATOR:

The SQL UNION ALL operator is used to combine the result sets of 2 or more SELECT statements. It returns all rows from the query (even if the row exists in more than one of the SELECT statements). The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows.Each SELECT statement within the UNION ALL must have the same number of fields in the result sets with similar data types.

SYNTAX

The syntax for the SQL UNION ALL operator is:

SELECT expression1, expression2, ... expression_n FROM tables WHERE conditions
UNION ALL
SELECT expression1, expression2, ... expression_n FROM tables WHERE conditions;

Parameters or Arguments

expression1, expression2, expression_n

The columns or calculations that you wish to retrieve.

Tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

Conditions

The conditions that must be met for the records to be selected.

INTERSECT OPERATOR

The Oracle INTERSECT operator is used to return the results of 2 or more SELECT statements. However, it only returns the rows selected by all queries or data sets. If a record exists in one query and not in the other, it will be omitted from the INTERSECT results. Each SELECT statement within the INTERSECT must have the same number of fields in the result sets with similar data types.

SYNTAX

The syntax for the INTERSECT operator in Oracle/PLSQL is:

SELECT expression1, expression2, ... expression_n   FROM tables WHERE conditions
INTERSECT *SELECT expression1, expression2, ... expression_n FROM tables*
WHERE conditions;

Parameters or Arguments

expression1, expression2, ... expression_n

The columns or calculations that you wish to retrieve.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

conditions

The conditions that must be met for the records to be selected.

IN Operator/ Condition

The in operator is used to compare a value to a list of literal values that have been specified**.**The IN operator allows you to specify multiple values in a WHERE clause. The Oracle IN condition is used to help reduce the need to use multiple OR conditions in a SELECT, INSERT, UPDATE, or DELETE statement. The Oracle IN condition is also called the Oracle IN operator.

SQL SYNTAX

SELECT column_name(s) FROM table_name WHERE olumn_name IN (value1,..,value n.);

parameters or arguments

value1, value2, ... value_n

The values to test against expression.

USING NOT OPERATOR

it complements IN operator

SELECT column_name (s) FROM table_name WHERE column_name NOT IN (value1,value2,..value n.);

EXISTS Condition:

The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.exists simply tests whether the inner query returns any row. if it does, then the outer query proceeds. if not, the outer query does not execute, and the entire sql statement returns nothing.

The syntax for exists is:

SELECT "column_name1" FROM "table_name1"

WHERE EXISTS (SELECT * FROM "table_name2"WHERE "condition");

The syntax for the SQL EXISTS condition is: WHERE EXISTS (subquery);

The subquery is a SELECT statement. If the subquery returns at least one record in its result set, the EXISTS clause will evaluate to true and the EXISTS condition will be met. If the subquery does not return any records, the EXISTS clause will evaluate to false and the EXISTS condition will not be met EXISTS Condition

NOT EXISTS

NOT EXISTS works like EXISTS, except the WHERE clause in which it is used is satisfied if no rows are returned by the subquery.The NOT operator reverses the meaning of the logical operator with which it is used. Eg: NOT EXISTS, NOT BETWEEN, NOT IN, etc. This is a negate operator.

COMPARISON OPERATORS MODIFIED BY ANY, SOME, OR ALL

Comparison operators that introduce a subquery can be modified by the keywords ALL or ANY. SOME is an ISO standard equivalent for ANY.

Subqueries introduced with a modified comparison operator return a list of zero or more values and can include a GROUP BY or HAVING clause. These subqueries can be restated with EXISTS.

Using the > comparison operator as an example, >ALL means greater than every value. In other words, it means greater than the maximum value.

For example, >ALL (1, 2, 3) means greater than 3.

>ANY means greater than at least one value, that is, greater than the minimum.

So >ANY (1, 2, 3) means greater than 1.

For a row in a subquery with >ALL to satisfy the condition specified in the outer query, the value in the column introducing the subquery must be greater than each value in the list of values returned by the subquery.

Similarly, >ANY means that for a row to satisfy the condition specified in the outer query, the value in the column that introduces the subquery must be greater than at least one of the values in the list of values returned by the subquery.

ALL

The ALL operator is used to compare a value to all values in another value set.

scalar_expression { = | <> | != | > | >= | !> | < | <= | !< } ALL (subquery)

scalar_expression

Is any valid expression.

{ = | <> | != | > | >= | !> | < | <= | !< }

Is a comparison operator.

subquery

Is a subquery that returns a result set of one column. The data type of the returned column must be the same data type as the data type of *scalar_expression*.

ANY

scalar_expression { = | < > | ! = | > | > = | ! > | < | < = | ! < }   { SOME | ANY } (subquery)

scalar_expression is any valid expression.{ = | <> | != | > | >= | !> | < | <= | !< }   is any valid comparison operator.

SOME | ANY: Specifies that a comparison should be made sub query

Is a subquery that has a result set of one column. The data type of the column returned must be the same data type as scalar_expression.

Sample Queries:

Q1: Getting the *cnames* and *cities* of companies' ACC' and 'TATA'.(union)

Q2: Display the names of employees who are living in 'NAGPUR' or 'BOMBAY'(union)

Q3: Display the employees names using UNION and UNION ALL

Q4: Display the employees names using UNION ALL and UNION

Q5: Retrieving the name of the city that is common to companies 'ACC' and 'TATA'(intersect)

Q6: Display the names of employees living in 'Nagpur' and working with company 'ACC'(intersect)

Q7: Find employee names and cities they live in and working IN company ' ACC' using IN Operator

Q8: Display names of employees who living IN 'NAGPUR','BOMBAY'   using in list (in)

Q9: Display the ename and salary of employees who salary NOT IN 3000,4000, 5000(NOT IN)

Q10:Returns names of employee of ACC having a salary greater than **ANY** other employee of 'TATA'.

Q11: Returns names of employee of ACC having a salary greater than **EVERY** employee of 'TATA'.

Q12: Returns names of employee, salary having a salary greater than/less than ANY of {2000, 3000, 4000}.

Q13: Returns names of employee, salary having a salary greater than/less than ALL of {2000, 3000, 4000}.

Q14.find the names of employee using exist living in the same city where sunil is living

Q15.find the names of employee having living city same as ajay(using exits)

Q16.find the names of employee having living city nagpur and company acc (using exists)

Sample Tables

```
SQL> select * from manager;
          ENAME                 MNAME
          ----------            ----------
          ANIL                  AJAY
          SHANKAR               VIJAY
          JAYA                  NULL
          SUNIL                 JAYA
          VIJAY                 NULL
          PRAKASH               SHANKAR
          AJAY                  NULL
          7 rows selected.
          SQL> select * from company;
          CNAME         CITY
          ----------    ----------
          ACC           MADRAS
          TATA          BOMBAY
          ACC           NAGPUR
          CMC           BOMBAY
          CMC           MADRAS
          TATA          DELHI
          6 rows selected.
          SQL> select * from emp;
          ENAME      CNAME         SALARY        JDATE
          ----------  ------------     ----------     ----------
          ANIL          ACC          1500          01-MAY-89
          SHANKAR    TATA           2000          10-JULY-90
          JAYA          CMC          1700          01-JAN-88
          SUNIL         CMC          1800          07-JAN-91
          VIJAY         TATA          500          03-JAN-88
```

```
PRAKASH     TATA        3000        27-MAY-89
AJAY        ACC         800         17-MAR-95
AMOL        ACC         1000        17-MAR-95
8 rows selected.
SQL> select * from emp1;
ENAME               CITY
----------          ----------
ANIL                NAGPUR
SHANKAR             BOMBAY
JAYA                MADRAS
SUNIL               BOMBAY
VIJAY               DELHI
PRAKASH             CALCUTTA
AJAY                MADRAS
7 rows selected.
```

VIVA QUESTIONS:

1. Difference Between SQL In/SQL Exists?
2. Difference Between SQL Not In/SQL Not Exists?
3. Difference Between SQL Union/SQL Union All?
4. How do we display rows from the table without duplicates?
5. Define DDL Interpreter.
6. What is the difference between rename and alias?
7. Which is more faster - IN or EXISTS.

Exercise 3: Queries using Aggregate functions and GROUP BY, HAVING and Creation and dropping of Views for a table.

AIM : To use select statement and demonstrate different Aggregate functions and group by ,having. To create, alter, and drop view for given tables.

DESCRIPTION

SQL GROUP Functions

Group functions are built-in SQL functions that operate on groups of rows and return one value for the entire group.

These functions are: **COUNT, MAX, MIN, AVG, SUM, DISTINCT**

Aggregate functions perform a variety of actions such as counting all the rows in a table, averaging a column's data, and summing numeric data. As with other types of queries, you can restrict, or filter out the rows these functions act on with the WHERE clause.

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two catagories,

- Aggregrate Functions
- Scalar Functions

Aggregrate Functions

These functions return a single value after calculating from a group of values. Following are some frequently used Aggregrate functions.

Some of the commonly used aggregate functions are as below -

- SUM([ALL | DISTINCT] expression)
- AVG([ALL | DISTINCT] expression)
- COUNT([ALL | DISTINCT] expression)
- COUNT(*)
- MAX(expression)
- MIN(expression)

The ALL and DISTINCT keywords are optional, and perform as they do with the SELECT clauses.The ALL keyword is the default where the option is allowed.The expression listed in the syntax can be a constant, a function, or any combination of column names, constants, and functions connected by arithmetic operators. However, aggregate functions are most often used with a column name. Except COUNT function,all the aggregate functions consider NULL values.

There are two rules that you must understand and follow when using aggregates:

Aggregate functions can be used in both the SELECT and HAVING clauses

Aggregate functions cannot be used in a WHERE clause. Its violation will produce the

Oracle ORA-00934 group function is not allowed here error message.

COUNT (): This function returns the number of rows in the table that satisfies the condition specified in the WHERE condition. If the WHERE condition is not specified, then the query returns the total number of rows in the table.

The COUNT function is used to count the number of rows returned in a SELECT statement. Count returns the number of rows present in the table either based on some condition or without condition.

Syntax:

SELECT COUNT(column_name) from table-name

OR

SELECT COUNT(aggregate_expression) FROM tables WHERE conditions;

OR

The syntax for the COUNT function when grouping the results by one or more columns is:

SELECT expression1, expression2, ... expression_n,   COUNT(aggregate_expression) FROM tables WHERE conditions   GROUP BY expression1, expression2, ... expression_n;

Parameters or Arguments

expression1, expression2, ... expression_n

Expressions that are not encapsulated within the COUNT function and must be included in the GROUP BY clause at the end of the SQL statement.

aggregate_expression

This is the column or expression whose non-null values will be counted.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

conditions

These are conditions that must be met for the records to be selected.

DISTINCT(): This function is used to select the distinct rows.

To get the count of employees with unique name, the query would be:

SELECT COUNT (DISTINCT name) FROM employee;

MAX ():The MAX function is used to return the maximum value of an expression in a SELECT statement.

Syntax

SELECT MAX(column_name) from table-name;

OR

SELECT MAX(aggregate_expression)FROM tablesWHERE conditions;

OR

The syntax for the MAX function when grouping the results by one or more columns is:

SELECT expression1, expression2, ... expression_n, MAX(aggregate_expression)

FROM tables WHERE conditions GROUP BY expression1, expression2, ... expression_n;

Parameters or Arguments

expression1, expression2, ... expression_n

Expressions that are not encapsulated within the MAX function and must be included in the GROUP BY clause at the end of the SQL statement.

aggregate_expression

This is the column or expression from which the maximum value will be returned.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

conditions

These are conditions that must be met for the records to be selected.

MIN (): The MIN function is used to return the minimum value of an expression in a SELECT statement. MIN function returns minimum value from a selected column of the table.

Syntax

SELECT MIN(column_name) from table-name;

SELECT MIN (salary) FROM employee;

SELECT MIN(aggregate_expression)FROM tablesWHERE conditions;

OR

The syntax for the MIN function when grouping the results by one or more columns is:

SELECT expression1, expression2, ... expression_n,MIN(aggregate_expression)

FROM tables WHERE conditions GROUP BY expression1, expression2, ... expression_n;

Parameters or Arguments

expression1, expression2, ... expression_n

Expressions that are not encapsulated within the MIN function and must be included in the GROUP BY clause at the end of the SQL statement.

aggregate_expression

This is the column or expression from which the minimum value will be returned.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

conditions

These are conditions that must be met for the records to be selected.

SUM ():The SQL SUM function is used to return the sum of an expression in a SELECT statement.

SUM function returns total sum of a selected columns numeric values.

Syntax

SELECT SUM(column_name) from table-name;

SELECT SUM(aggregate_expression)FROM tablesWHERE conditions;

OR

The syntax for the SUM function when grouping the results by one or more columns is:

SELECT expression1, expression2, ... expression_n,SUM(aggregate_expression)

FROM tables WHERE conditionsGROUP BY expression1, expression2, ... expression_n;

Parameters or Arguments

expression1, expression2, ... expression_n

Expressions that are not encapsulated within the SUM function and must be included in the GROUP BY clause at the end of the SQL statement.

aggregate_expression

This is the column or expression that will be summed.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

conditions

These are conditions that must be met for the records to be selected.

AVG ():The SQL AVG function is used to return the average of an expression in a SELECT statement.This function is used to get the average value of a numeric column.Average returns average value after calculating from values in a numeric column.

Syntax is

SELECT AVG(column_name) from table_name

OR

SELECT AVG("column_name")FROM "table_name";

SELECT AVG(aggregate_expression)FROM tablesWHERE conditions;

OR

the syntax for the AVG function when grouping the results by one or more columns is:

SELECT expression1, expression2, expression, AVG(aggregate expression)

FROM tables WHERE conditions GROUP BY expression1, expression2, expression;

Parameters or Arguments

expression1, expression2, ... expression_n

Expressions that are not encapsulated within the AVG function and must be included in the GROUP BY clause at the end of the SQL statement.

aggregate_expression

This is the column or expression that will be averaged.

tables

The tables that you wish to retrieve records from. There must be at least one table listed in the FROM clause.

conditions

These are conditions that must be met for the records to be selected.

VIEW

To create, update, and drop views.

WHAT IS A VIEW IN ORACLE?

An Oracle VIEW, in essence, is a virtual table that does not physically exist. Rather, it is created by a query joining one or more tables. In SQL, a view is a virtual table based on the result-set of an SQL statement.A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. A view in SQL is a logical subset of data from one or more tables. View is used to restrict data access.

Types of View:There are two types of view,

- Simple View
- Complex View

Simple View	Complex View
Created from one table	Created from one or more table
Does not contain functions	Contain functions
Does not contain groups of data	Contains groups of data

Create View Syntax :

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW *view* [(*alias*[, *alias*]...)] AS *subquery* [WITH CHECK OPTION [CONSTRAINT *constraint*]] [WITH READ ONLY [CONSTRAINT *constraint*]];

UPDATE VIEW

You can modify the definition of an Oracle VIEW without dropping it by using the Oracle CREATE OR REPLACE VIEW Statement.

Update a View

Update command for view is same as for tables.

Syntax to Update a View is,

UPDATE view-name set value WHERE condition;

If we update a view it also updates base table data automatically.

[OR]

Syntax

The syntax for the CREATE OR REPLACE VIEW Statement in Oracle/PLSQL is:

CREATE OR REPLACE VIEW view_name AS

SELECT columns FROM table WHERE conditions;

view_name

The name of the Oracle VIEW that you wish to create or replace.

Read-Only View

We can create a view with read-only option to restrict access to the view.

Syntax to create a view with Read-Only Access

CREATE or REPLACE force view view_name AS SELECT column_name(s)

FROM table_name WHERE condition with read-only

The above syntax will create view for read-only purpose, we cannot Update or Insert data into read-only view. It will throw an error.

DROP VIEW

Once an Oracle VIEW has been created, you can drop it with the Oracle DROP VIEW Statement.

Syntax

The syntax for the DROP VIEW Statement in Oracle/PLSQL is:

*DROP VIEW view_name;*view_name

The name of the view that you wish to drop.

<u>Sample Queries</u>

Q1.Compute the total salary, average salary, max salary, min salary of all the employees of the company

Q2.Find the minimum jdate and maximum jdate of employees

Q3. Find total number of employees in the company

Q4. Find total number of employees in the company acc

Q5. Find the average salary of each department

Q6. display the names of companies and maximum salary in that company.

Q7. Calculate the no of employees in each company.

Q8. find the company having maximum salary greater than 3000

Q9.find the first, second and third highest salary employee details

Q10. Display the companies having more than one employee

Q11. Find the 3rd rank student details

Q12.create simple view (read only and update) and complex views and drop the views

viva Questions

1. What is USING Clause and give example?
2. What restrictions can you apply when you are creating views?
3. What is the use of Aggregate functions in Oracle?
4. What do you mean by GROUP BY Clause?
5. What is a View?
6. What is the data type of DUAL table?
7. When you use WHERE clause and when you use HAVING clause

****

Exercise 4: Demonstrate various Queries on single row functions like DATE, STRING (or CHARACTER) and CONVERSION functions for a table.

AIM :To use date, character, conversion function in select statement and demonstrate and Queries on functions.

DESCRIPTION

- Single line functions:

Date Functions: SYSDATE, MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, TRUNC, ROUND, LEAST, GREATEST

Character Functions: LOWER, UPPER, INITCAP

Character Manipulation Functions: CONCATENATION, LPAD, RPAD, LTRIM, RTRIM, LENGTH, SUBSTR AND INSTR, REPLACE

Conversion function: TO_CHAR, TO_NUMBER, TO_DATE, NAVL, NAVL2, NULLIF

- Date Functions:

SYSDATE FUNCTION: SYSDATE returns the current date and time set for the operating system on which the database resides. The data type of the returned value is DATE The function requires no arguments

SYNTAX

The syntax for the SYSDATE function in Oracle/PLSQL is:

SYSDATE

Parameters or Arguments

There are no parameters or arguments for the SYSDATE function.

MONTHS_BETWEEN:The Oracle/PLSQL MONTHS_BETWEEN function returns the number of months between date1 and date2.

SYNTAX: The syntax for the MONTHS_BETWEEN function in Oracle/PLSQL is:

MONTHS_BETWEEN (date1, date2)

Parameters or Arguments

date1

The first date used to calculate the number of months between.

date2

The second date used to calculate the number of months between.

Note: If a fractional month is calculated, the MONTHS_BETWEEN function calculates the fraction based on a 31-day month.

ADD_MONTHS: The Oracle/PLSQL ADD_MONTHS function returns a date with a specified number of months added.

SYNTAX: The syntax for the ADD_MONTHS function in Oracle/PLSQL is:

ADD_MONTHS (date1, number_months)

Parameters or Arguments

date1

The starting date (before the n months have been added).

number_months

The number of months to add to date1.

NEXT_DAY:The Oracle/PLSQL NEXT_DAY function returns the first weekday that is greater than a date.

SYNTAX: The syntax for the NEXT_DAY function in Oracle/PLSQL is:

NEXT_DAY(date, weekday)

Parameters or Arguments

date

A date value used to find the next weekday.

weekday

The day of the week that you wish to return. It can be one of the following values:

Value	Description
SUNDAY	First Sunday greater than date
MONDAY	First Monday greater than date
TUESDAY	First Tuesday greater than date
WEDNESDAY	First Wednesday greater than date
THURSDAY	First Thursday greater than date
FRIDAY	First Friday greater than date
SATURDAY	First Saturday greater than date

LAST_DAY:The Oracle/PLSQL LAST_DAY function returns the last day of the month based on a date value.

SYNTAX: The syntax for the LAST_DAY function in Oracle/PLSQL is:

LAST_DAY(date)

Parameters or Arguments

date

The date value to use to calculate the last day of the month.

TRUNC FUNCTION (WITH DATES)

DESCRIPTION

The Oracle/PLSQL TRUNC function returns a date truncated to a specific unit of measure.

SYNTAX (WITH DATES)

The syntax for the TRUNC function in Oracle/PLSQL is:

TRUNC (date [, format])

Parameters or Arguments

date

The date to truncate.

format

Optional. The unit of measure to apply for truncating. If the format parameter is omitted, the TRUNC function will truncate the date to the day value, so that any hours, minutes, or seconds will be truncated off. It can be one of the following values:

Unit	Valid format parameters
Year	SYYYY, YYYY, YEAR, SYEAR, YYY, YY, Y
ISO Year	IYYY, IY, I
Quarter	Q
Month	MONTH, MON, MM, RM
Week	WWIWIWWW
Day	DDD, DD, J
Start day of the week	DAY, DY, D
Hour	HH, HH12, HH24
Minute	MI

ROUND:The Oracle/PLSQL ROUND function returns a date rounded to a specific unit of measure.

ROUND FUNCTION SYNTAX (WITH DATES)

The syntax for the ROUND function in Oracle/PLSQL is:

ROUND(date [, format])

Parameters or Arguments

date

The date to round.

format

Optional. The unit of measure to apply for rounding. If the format parameter is omitted, the ROUND function will round to the nearest day.

LEAST:The Oracle/PLSQL LEAST function returns the smallest value in a list of expressions.

SYNTAX

The syntax for the LEAST function in Oracle/PLSQL is:

LEAST(expr1 [, expr2, ... expr_n])

Parameters or Arguments

expr1

The first expression to evaluate whether it is the smallest.

expr2, ... expr_n

Optional. Additional expressions that are to be evaluated.

Note:

If the datatypes of the expressions are different, all expressions will be converted to whatever datatype expr1 is.

If the comparison is based on a character comparison, one character is considered smaller than another if it has a lower character set value.

Having a NULL value in one of the expressions will return NULL as the least value.

GREATEST:The Oracle/PLSQL GREATEST function returns the greatest value in a list of expressions.

SYNTAX:The syntax for the GREATEST function in Oracle/PLSQL is:

GREATEST(expr1 [, expr2, ... expr_n])

Parameters or Arguments

expr1

The first expression to be evaluated whether it is the greatest.

expr2, ... expr_n

Optional. Additional expressions that are to be evaluated.

Sample Queries:

Character Functions & Character Manipulation Functions

Q1. write sql query to convert given sting in to lowercase.

Q2. write sql query to convert given sting in to uppercase

Q3. write sql query to convert given sting to initial uppercase character

Q4. write sql query to concatenate two strings.

Q5. write sql query to display the substring of a given string

Q6. write sql query to display the length of the given string

Q7. Write a sql query to find the first position of the character in the given string.

Q8. Write a sql query to demonstrate the padding of fields.

Q9. write a sql query to demonstrate trim the character from given string.

Q10. Write sql query to replace string with another sting

Date Functions :

Q11. Write a sql query to find the current date.

Q12. write sql query to find the number of months between two given dates

Q13. write sql query to display the date after addition of given number of month

Q14. write a sql query to find the next week day from given date.

Q15. write a sql query to find the last day from given date.

Q16. write a sql query to find the least, greatest date from given date.

Q17. write a sql query to round the given date.

Q18. write a sql query to truncate the given date.

Conversion function : TO_CHAR ,TO_NUMBER,TO_DATE

Viva questions

1.Whether any commands are used for Months calculation? If so, What are they?

2.What is difference between SUBSTR and INSTR?

3.What are character functions?

4.How does ROWID help in running a query faster?

5.How would you convert date into Julian date format?We can use the J format string :

SQL > select to_char(to_date(‘29-Mar-2013’,’dd-mon-yyyy’),’J’) as julian from dual;

****

INTRODUCTION TO PL/SQL

What is PL/SQL?

PL/SQL is a Procedural Language extension of Structured Query Language (SQL).PL/SQL is specially designed for Database oriented activities. Oracle PL/SQL allows you to perform data manipulation operation those are safe and flexible.PL/SQL is a very secure functionality tool for manipulating, controlling, validating, and restricting unauthorized access data from the SQL database. Using PL/SQL we can improve application performance. It also allows to deal with errors so we can provide user friendly error messages.PL/SQL have a great functionality to display multiple records from the multiple tables at the same time.PL/SQL is capable to send entire block of statements and execute it in the Oracle engine at once.

Advantages PL/SQL

Procedural language support: PL/SQL is a development tools not only for data manipulation futures but also provide the conditional checking, looping or branching operations same as like other programming language.

Reduces network traffic : This one is great advantages of PL/SQL. Because PL/SQL nature is entire block of SQL statements execute into oracle engine all at once so it's main benefit is reducing the network traffic.

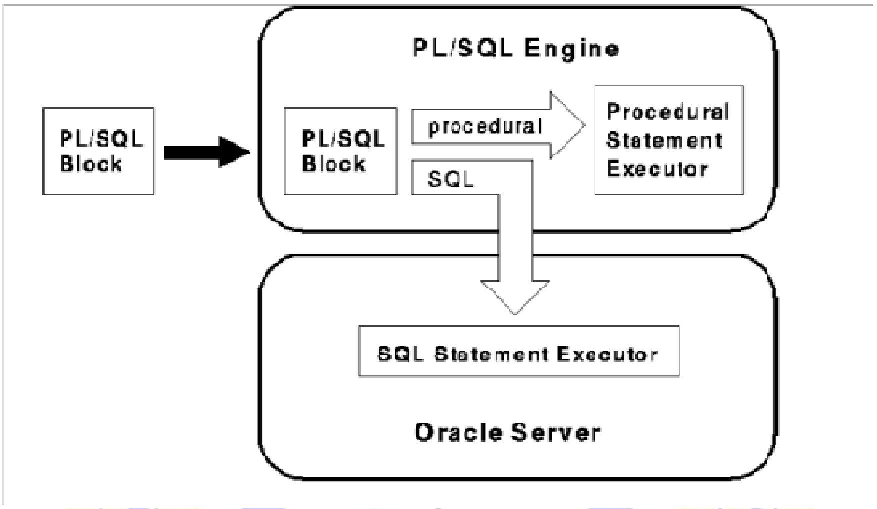
Error handling : PL/SQL is dealing with error handling, It's permits the smart way handling the errors and giving user friendly error messages, when the errors are encountered.

Declare variable : PL/SQL gives you control to declare variables and access them within the block. The declared variables can be used at the time of query processing.

Intermediate Calculation : Calculations in PL/SQL done quickly and efficiently without using Oracle engines. This improves the transaction performance.

Portable application : Applications are written in PL/SQL are portable in any Operating system. PL/SQL applications are independence program to run any computer.

PL/SQL ARCHITECTURE



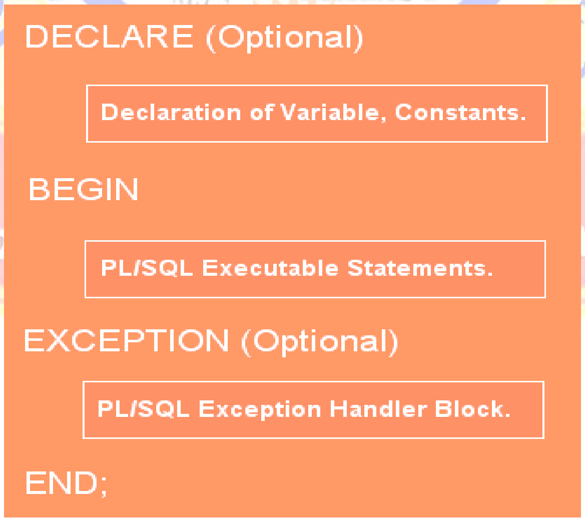
PL/SQL block structure follows divide-and-conquer approach to solve the problem stepwise.

PL/SQL program units are generally categorized as follows:

 Anonymous blocks

 Stored procedures

PL/SQL block



DECLARE (Optional)

Declaration of Variable, Constants.

BEGIN

PL/SQL Executable Statements.

EXCEPTION (Optional)

PL/SQL Exception Handler Block.

END;

Declarations

This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.

Executable Commands

This section is enclosed between the keywords **BEGIN** and **END** and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.

Exception Handling

This section starts with the keyword **EXCEPTION**. This section is again optional and contains exception(s) that handle errors in the program.

Unknown Column types

PL/SQL this data type is used when column type is not known.

Data types	Description
%Type	This data type is used to store value unknown data type column in a table. Column is identified by %type data type.
%RowType	This data type is used to store values unknown data type in all columns in a table. All columns are identified by %RowType datatype.

Variable Declaration Syntax

The general syntax to declare a variable is:

variable_name   Datatype [Size] [NOT NULL] := [value];

Exercise 5_a: PL/SQL BLOCKS

AIM: Creation of simple PL/SQL program which includes declaration section, executable section and exception –Handling section

Description:

```
        DECLARE
                        <declarations section>
        BEGIN
                        <executable command(s)>
        EXCEPTION
                        <exception handling>
        END;
```

Exercise 5_b: TCL command

AIM: Insert data into student table and use COMMIT, ROLLBACK and SAVEPOINT in PL/SQL Block.

Description:Transaction Control Language (TCL) commands are used to manage transactions in database. These are used to manage the changes made by DML statements. There are following commands used to control transactions

- Commit
- Rollback
- Savepoint

The COMMIT Command:make changes permanent save to a database during the current transaction. The COMMIT command is the transactional command used to save changes invoked by a transaction to the database. The COMMIT command saves all transactions to the database since the last COMMIT or ROLLBACK command. Commit command is used to permanently save any transaction into database.

The syntax for COMMIT command is as follows:

COMMIT;

The ROLLBACK Command:ROLLBACK command execute at the end of current transaction and undo/undone any changes made since the begin transaction. The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. The ROLLBACK command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued. This command restores the database to last committed state. It is also use with save point command to jump to a save point in a transaction.

The syntax for ROLLBACK command is as follows:

ROLLBACK TO SAVEPOINT-NAME;

The SAVEPOINT Command: SAVEPOINT command save the current point with the unique name in the processing of a transaction. A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction. This command serves only in the creation of a SAVEPOINT among transactional statements. The ROLLBACK command is used to undo a group of transactions. Save point command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

The syntax for SAVEPOINT command is as follows:

SAVEPOINT   SAVEPOINT_NAME;

Viva questions:

1. What is the basic structure of PL/SQL?
3. Define Commit, Rollback and Save point.
4. What is the difference between grant, commit, rollback and save point.
5. What is the maximum buffer size that can be specified using the DBMS_OUTPUT.ENABLE function? Answer 1,000,000

****

Exercise 6: NESTED IF, CASE and CASE expression. NULLIF and COALESCE functions.

AIM : Develop a program that includes the features NESTED IF, CASE and CASE expression. The Program can be extended using the NULLIF and COALESCE functions.

CONDITIONAL SELECTION STATEMENTS

Description:

The conditional selection statements, IF and CASE, run different statements for different data values. The IF statement either runs or skips a sequence of one or more statements, depending on a condition. The IF statement has these forms:

IF…. THEN

IF… THEN… ELSE

IF….. THEN.. ELSIF

The CASE statement chooses from a sequence of conditions, and runs the corresponding statement. The CASE statement has these forms:Simple, which evaluates a single expression and compares it to several potential values. Searched, which evaluates multiple conditions and chooses the first one that is true. The CASE statement is appropriate when a different action is to be taken for each alternative.

IF…. THEN Statement

The IF… THEN statement has this structure:

```
IF condition THEN
    statements
END IF;
```

If the condition is true, the statements run; otherwise, the IF statement does nothing.

IF THEN ELSE Statement

The IF THEN ELSE statement has this structure:

```
IF condition THEN
        statements
  ELSE
        else_statements
  END IF;
```

If the value of condition is true, the statements run; otherwise, the else statements run.

IF.. THEN…. ELSIF Statement

The IF THEN ELSIF statement has this structure:

```
      IF condition_1 THEN  statements_1
          ELSIF condition_2 THEN  statements_2
               [ ELSIF condition_3 THEN   statements_3  ]...
                   [ ELSE
                      else_statements     ]
     END IF;
```

The IF THEN ELSIF statement runs the first statements for which condition is true. Remaining conditions are not evaluated. If no condition is true, the else_statements run, if they exist; otherwise, the IF THEN ELSIF statement does nothing.

CASE STATEMENT:The CASE statement chooses from a sequence of conditions, and executes a corresponding statement. The CASE statement evaluates a single expression and compares it against several potential values, or evaluates multiple Boolean expressions and chooses the first one that is TRUE. Simple case statement, Searched case statement

The Simple case statement has this structure:

```
case selector
when selector_value_1 then statements_1
when selector_value_2 then statements_2
...
when selector_value_n then statements_n
[ else
  else_statements ]
end case;]
```

the selector is an expression (typically a single variable). Each selector_value can be either a literal or an expression.

Searched case statement

The searched case statement has this structure:

```
CASE [ expression ]
   WHEN condition_1 THEN result_1  WHEN condition_2 THEN result_2
```

... WHEN condition_n THEN result_n ELSE result

END

The searched case statement runs the first statements for which condition is true. Remaining conditions are not evaluated. if no condition is true, the case statement runs else statements if they exist and raises the predefined exception case_not_found otherwise.

Parameters or Arguments

Expression

Optional. It is the value that you are comparing to the list of conditions. (ie: condition_1, condition_2, ... condition_n)

condition_1, condition_2, ... condition_n

The conditions that must all be the same datatype. The conditions are evaluated in the order listed. Once a condition is found to be true, the CASE statement will return the result and not evaluate the conditions any further.

result_1, result_2, ... result_n

Results that must all be the same datatype. This is the value returned once a condition is found to be true.

COALESCE FUNCTION:The Oracle/PLSQL COALESCE function returns the first non-null expression in the list. If all expressions evaluate to null, then the COALESCE function will return null.

The syntax for the COALESCE function in Oracle/PLSQL is:

COALESCE(expr1, expr2, ... expr_n)

Parameters or Arguments

expr1, expr2, ... expr_n

The expressions to test for non-null values.The COALESCE function will compare each value, one by one.

NULLIF FUNCTION:The Oracle/PLSQL NULLIF function compares expr1 and expr2. If expr1 and expr2 are equal, the NULLIF function returns NULL. Otherwise, it returns expr1.

The syntax for the NULLIF function in Oracle/PLSQL is:

NULLIF(expr1, expr2)

Parameters or Arguments

expr1:First value to compare. Must be either a numeric value or a value that is the same datatype as expr2.

expr2:Second value to compare. Must be either a numeric value or a value that is the same datatype as expr1.

Note: expr1 can be an expression that evaluates to NULL, but it cannot be the literal NULL.

Viva questions

1. What special operators does Oracle provide for dealing with NULLs?
2. Explain three different rules that apply to NULLs when doing comparisons?
3. What is COALESCE function?
4. What is NULL value in oracle?
5. Differ between DECODE and CASE.

****

Exercise_7: PL/SQL LOOP Statement, Exceptions

AIM : Program development using WHILE LOOPS, numeric FOR LOOPS, nested loops using ERROR Handling, BUILT –IN Exceptions, User defined Exceptions, RAISE- APPLICATION ERROR.

Description

PL/SQL LOOP statement is an iterative control statement that allows you to execute a sequence of statements repeatedly like WHILE and FOR loop.The simplest form of the LOOP statement consists of the LOOP keyword, a sequence of statements and the END LOOP keywords as shown below:

```
LOOP
        sequence_of_statements;
END LOOP;
```

Iterative control statements

- FOR LOOP
- WHILE LOOP

WHILE-LOOP

A WHILE LOOP is used when a set of statements has to be executed as long as a condition is true. The condition is evaluated at the beginning of each iteration. The iteration continues until the condition becomes false.

Before each iteration of the loop, the condition is evaluated. If the condition is true, the sequence of statements is executed, then control resumes at the top of the loop. If the condition is false or null, the loop is bypassed and control passes to the next statement

The General Syntax WHILE LOOP is:

```
WHILE condition LOOP
        sequence_of_statements
END LOOP;
```

Parameters or Arguments

condition

The condition is test each pass through the loop. If condition evaluates to TRUE, the loop body is executed. If condition evaluates to FALSE, the loop is terminated.

statements

The statements of code to execute each pass through the loop.

FOR-LOOP:

There are two kinds of PL/SQL FOR loops: the numeric FOR loop and the cursor FOR loop. The numeric FOR loop is the traditional and familiar "counted" loop A FOR LOOP is used to execute a set of statements for a predetermined number of times. Iteration occurs between the start and end integer values given. The counter is always incremented by 1. The loop exits when the counter reaches the value of the end integer.

The syntax for the FOR Loop is:

```
FOR counter IN [REVERSE] lower_bound..higher_bound LOOP
            sequence_of_statements
 END LOOP;
```

Parameters or Arguments

loop_counter: The loop counter variable.

REVERSE:Optional. If specified, the loop counter will count in reverse.

lower_bound: The starting value for loop_counter.

higher_bound: The ending value for loop_counter.

Statements:The statements of code to execute each pass through the loop.

A double dot (..) serves as the range operator.The range is evaluated when the `FOR` loop is first entered and is never re-evaluated.the FOR LOOP allows you to execute code repeatedly for a fixed number of times. Whereas the number of iterations through a `WHILE` loop is unknown until the loop completes, the number of iterations through a `FOR` loop is known before the loop is entered.

PL/SQL – Exceptions

An error condition during a program execution is called an exception in PL/SQL. PL/SQL supports programmers to catch such conditions using EXCEPTION block in the program and an appropriate action is taken against the error condition.

PL/SQL catches and handles exceptions by using exception handler architecture. Whenever an exception occurs, it is raised. The current PL/SQL block execution halts and control is passed to a separated section called exception section.

There are two types of exceptions:

- System-defined exceptions
- User-defined exceptions

Pre-defined Exceptions

An internal exception is raised implicitly whenever PL/SQL program violates an Oracle rule or exceeds a system-dependent limit. Every Oracle error has a number, but exceptions must be handled by name. So, PL/SQL predefines some common Oracle errors as exceptions

For example, the predefined exception NO_DATA_FOUND is raised when a SELECT INTO statement returns no rows.

Exception	Oracle Error	SQLCODE Value
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
TOO_MANY_ROWS	ORA-01422	-1422
ZERO_DIVIDE	ORA-01476	-1476

User-defined Exceptions

PL/SQL allows you to define your own exceptions according to the need of your program. A user-defined exception must be declared and then raised explicitly, using either a RAISE statement or the procedure DBMS_STANDARD.RAISE_APPLICATION_ERROR.

Defining Your Own Error Messages: Procedure RAISE_APPLICATION_ERROR

The procedure RAISE_APPLICATION_ERROR issue user-defined ORA- error messages from stored subprograms. Report errors to your application and avoid returning unhandled exceptions.

To call RAISE_APPLICATION_ERROR, use the syntax

RAISE_APPLICATION_ERROR(ERROR_NUMBER, MESSAGE[, {TRUE | FALSE}]);

where error_number is a negative integer in the range -20000 .. -20999 and message is a character string up to 2048 bytes long. If the optional third parameter is TRUE, the error is placed on the stack of previous errors. If the parameter is FALSE (the default), the error replaces all previous errors. RAISE_APPLICATION_ERROR is part of package DBMS_STANDARD, and as with package STANDARD

Viva questions:

1. How exception is different from error?

2. What is exception? What are the types of exceptions?

3. Explain Raise_application_error.

4. How does a syntax error differ from a runtime error?

5. What is Exception Handling?

6. Types of Exception.

7. What does the PL/SQL raise_application_error code do?

****

Exercise _8: PL/SQL - Procedures

Aim : Programs development using creation of procedures, passing parameters IN and OUT of PROCEDURES.

Description:

A stored procedure is a PL/SQL block that is stored in the database with a name. It is invoked using the name. Each procedure is meant for a specific purpose. A stored procedure is stored in the database as an object. It is also called as database procedure as it is stored in the database. A procedure may take one or more parameters. If a procedure takes parameters then these parameters are to be supplied at the time of calling the procedure. these subprograms do not return a value directly, mainly used to perform an action.

Creating a Procedure

A procedure is created with the CREATE OR REPLACE PROCEDURE statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows:

CREATE [OR REPLACE] PROCEDURE procedure_name [(parameter_name [IN | OUT | IN OUT] type [, ...])] {IS | AS} BEGIN

< procedure_body > END procedure_name;

Where,

procedure-name specifies the name of the procedure.

[OR REPLACE] option allows modifying an existing procedure.

The optional parameter list contains name, mode and types of the parameters.

There are three types of parameters that can be declared:

IN represents that value will be passed from outside and

IN - The parameter can be referenced by the procedure or function. The value of the parameter cannot be overwritten by the procedure or function.

OUT represents that this parameter will be used to return a value outside of the procedure.

OUT - The parameter cannot be referenced by the procedure or function, but the value of the parameter can be overwritten by the procedure or function.

IN OUT - The parameter can be referenced by the procedure or function and the value of the parameter can be overwritten by the procedure or function.

Procedure-body contains the executable part.

The AS keyword is used instead of the IS keyword for creating a standalone procedure.

DROP PROCEDURE

Once you have created your procedure in Oracle, you might find that you need to remove it from the database.

Syntax

DROP PROCEDURE procedure_name;

procedure_name

The name of the procedure that you wish to drop.

SQL> Drop procedure emp;

Viva questions

1. What is the parameter mode that can be passed to a procedure?
2. What are the different Oracle Database objects?
3. What is stored Procedure?
4. What are the advantages of stored procedure?

****

Exercise _9: PL/SQL - function

AIM : Program development using creation of stored functions, invoke functions in SQL Statements and write complex functions.

Description:

CREATE FUNCTION

Use the CREATE FUNCTION statement to create a standalone stored function or a call specification. A PL/SQL function is same as a procedure except that it returns a value. Functions: these subprograms return a single value, mainly used to compute and return avalue.A stored function (also called a user function or user-defined function) is a set of PL/SQL statements you can call by name. Stored functions are very similar to procedures, except that a function returns a value to the environment in which it is called. User functions can be used as part of a SQL expression.

Creating a Function

A standalone function is created using the CREATE FUNCTION statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows:

```
CREATE [OR REPLACE] FUNCTION function_name
    [(parameter_name [IN | OUT | IN OUT] type [, ...])]
   RETURN return_datatype      {IS | AS}
BEGIN
    < function_body >
END [function_name];
```

Where,

Function-name specifies the name of the function.

[OR REPLACE] option allows modifying an existing function.

The optional parameter list contains name, mode and types of the parameters. IN represents that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.

The function must contain a return statement.

RETURN clause specifies that data type you are going to return from the function.

Function-body contains the executable part.

The AS keyword is used instead of the IS keyword for creating a standalone function.

DROP FUNCTION

Once you have created your function in Oracle, you might find that you need to remove it from the database.

Syntax

The syntax to a drop a function in Oracle is:

DROP FUNCTION function_name;

function_name

The name of the function that you wish to drop.

Sql> Drop function emp;

Viva questions

1. Mention what PL/SQL package consists of?
2. Mention what are the benefits of PL/SQL packages?
3. What are different modes of parameters used in functions and procedures?
4. What is difference between a formal and an actual parameter?
5. Can a function take OUT parameters? If not why.
6. No.A function has to return a value; an OUT parameter cannot return a value.

****

Exercise _10: PL/SQL - package

AIM : Program development using creation of package specification, package bodies, package variables and cursors and calling stored packages

Description:Packages

A package is a schema object that groups logically related PL/SQL types, items, and subprograms. Packages usually have two parts, a specification and a body, although sometimes the body is unnecessary. The specification (spec for short) is the interface to your applications; it declares the types, variables, constants, exceptions, cursors, and subprograms available for use. The body fully defines cursors and subprograms, and so implements the spec.A package is a collection of related procedures, functions, variables and data types. A
package typically contains two parts – specification and body.

Specification

Package specification contains declarations for items, procedure and functions that are to be made public. All public objects of package are visible outside the package. In other words, public objects are callable from outside of the package.

Private items of the package can be used only in the package and not outside the package.

The following is the syntax to create package specification.

```
CREATE PACKAGE package_name AS
/* declare public objects of package */
END;
```

Body

Body of the package defines all the objects of the package. It includes public objects that are declared in package specification and objects that are to be used only within the package – private members.

```
CREATE PACKAGE BODY package_name AS
/* define objects of package */
END;
```

procedures declared in the package specification
functions declared in the package specification
definition of cursors declared in the package specification

local procedures and functions, not declared in the package specification

local variables

Calling a procedure of package

In order to access a public object of a package use the following syntax:

package_name.object_name

package_name is the name of the package whose object you want to access.

object_name is the name of a public object in the package.

VIVA Questions:

1. Explain the difference between a FUNCTION, PROCEDURE and PACKAGE.
2. What packages (if any) has Oracle provided for use by developers?
3. What are % TYPE and % ROWTYPE? What are the advantages of using these over datatypes?
4. What is a stored procedure?
5. What is difference between a PROCEDURE & FUNCTION?
6. What are the modes of parameters that can be passed to a procedure?
7. What is a package? What are the advantages of packages?
8. What are two parts of package?
9. What are the rules of writing package?
10. Explain IN, OUT and INOUT in procedures

****

Exercise _11: PL/SQL Cursors

AIM : Develop programs using features parameters in a CURSOR, AND CURSOR variables.

PL/SQL – Cursors

Description:

Cursor is the work area which Oracle reserves for internal processing of SQL statements. This work area is private for oracles reserved are called cursor.

A cursor is a temporary work area created in the system memory when a SQL statement is executed. A cursor contains information on a select statement and the rows of data accessed by it.This temporary work area is used to store the data retrieved from the database, and manipulate this data. A cursor can hold more than one row, but can process only one row at a time.

Oracle creates a memory area, known as context area, for processing an SQL statement, which contains all information needed for processing the statement, for example, number of rows processed, etc.A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

How to Use Cursor

In PL/SQL block SELECT statement cannot return more than one row at a time. So Cursor use to some group of rows (more than one row) for implementing certain logic to get all the group of records.

Implicit Cursor

Even if we execute a SELECT statement or DML statement Oracle reserves a private SQL area in memory called cursor.

Implicit cursor scope you can get information from cursor by using session attributes until another SELECT statement or DML statement execute.

There are two types of cursors in PL/SQL:

- Implicit Cursor or Internal Cursor: Manage for Oracle itself or internal process itself.
- Explicit Cursor or User-defined Cursor: Manage for user/programmer or external processing

Implicit cursors

Oracle uses implicit cursors for its internal processing.These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed. They are also created when a SELECT statement that returns just one row is executed.

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed Programmers cannot control the implicit cursors and the information in it.

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has the attributes like %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The SQL cursor has additional attributes, %BULK_ROWCOUNT and %BULK_EXCEPTIONS, designed for use with the FORALL statement.

The following table provides the description of the most used attributes:

Attribute	**Description**
%FOUND	Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one Or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
%NOTFOUND	The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE or DELETE statement affected no rows, or a SELECT INTO Statement returned no rows. Otherwise, it returns FALSE.
%ISOPEN	Always returns FALSE for implicit cursors, because Oracle closes the SQL Cursor automatically after executing its associated SQL statement.
%ROWCOUNT	Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Explicit Cursor

Explicit Cursor which is construct/manage by user itself calls explicit cursor. User itself to declare the cursor, open cursor to reserve the memory and populate data, fetch the records from the active data set one at a time, apply logic and last close the cursor. You can not directly assign value to an explicit cursor variable you have to use expression or create subprogram for assign value to explicit cursor variable.

Explicit cursors are programmer defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is :

CURSOR cursor_name IS select_statement;

Step for Using Explicit Cursor::

- Declaring the cursor for initializing in the memory
- Opening the cursor for allocating memory
- Fetching the cursor for retrieving data
- Closing the cursor to release allocated memory

Oracle provides few attributes called as implicit cursor attributes to check the status of DML operations. The cursor attributes available are %FOUND, %NOTFOUND, %ROWCOUNT, and %ISOPEN.

The status of the cursor for each of these attributes are defined in the below table.

Declaring the Cursor:Declaring the cursor defines the cursor with a name and the associated SELECT statement.

Opening the Cursor:Opening the cursor allocates memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it.

Fetching the Cursor:Fetching the cursor involves accessing one row at a time.

Closing the Cursor:Closing the cursor means releasing the allocated memory

Viva questions

1. What is a cursor variable?
2. What are cursor attributes?
3. What are the attributes of Cursor?
4. What is an integrity constraint?
5. Difference between numaric for loop and cursor for loop
6. Explain Implicit and Explicit cursors
7. What are PL/SQL Cursor Exceptions?
8. Difference between NO DATA FOUND and %NOTFOUND

****

Exercise _12: **PL/SQL - Triggers**

AIM : Develop Programs using BEFORE and AFTER Triggers, Row and Statement Level Triggers

Description:

Oracle engine invokes automatically whenever a specified event occurs.Trigger is stored into database and invoked repeatedly, when specific condition match.

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events:

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- A database definition (DDL) statement (CREATE, ALTER, or DROP).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

Component of Trigger

A trigger has three basic parts:

- A triggering event or statement
- A trigger restriction
- A trigger action

Triggering SQL statement : SQL DML (INSERT, UPDATE and DELETE) statement that execute and implicitly called trigger to execute.

Trigger Action : When the triggering SQL statement is execute, trigger automatically call and PL/SQL trigger block execute.

Trigger Restriction : We can specify the condition inside trigger to when trigger is fire.

Type of Triggers

BEFORE Trigger : BEFORE trigger execute before the triggering DML statement (INSERT, UPDATE, DELETE) execute. Triggering SQL statement is may or may not execute, depending on the BEFORE trigger conditions block.

AFTER Trigger : AFTER trigger execute after the triggering DML statement (INSERT, UPDATE, DELETE) executed. Triggering SQL statement is execute as soon as followed by the code of trigger before performing Database operation.

ROW Trigger : ROW trigger fire for each and every record which are performing INSERT, UPDATE, DELETE from the database table. If row deleting is define as trigger event, when trigger file, deletes the five rows each times from the table.

Statement Trigger : Statement trigger fire only once for each statement. If row deleting is define as trigger event, when trigger file, deletes the five rows at once from the table.

Combination Trigger : Combination trigger are combination of two trigger type,

Before Statement Trigger : Trigger fire only once for each statement before the triggering DML statement.

Before Row Trigger : Trigger fire for each and every record before the triggering DML statement.

After Statement Trigger : Trigger fire only once for each statement after the triggering DML statement executing.

After Row Trigger : Trigger fire for each and every record after the triggering DML statement executing.

PL/SQL Triggers Syntax

```
CREATE [OR REPLACE] TRIGGER trigger_name
        BEFORE | AFTER [INSERT, UPDATE, DELETE [COLUMN NAME..]
        ON table_name          Referencing [ OLD AS OLD | NEW AS NEW ]
        FOR EACH ROW | FOR EACH STATEMENT [ WHEN Condition ]
DECLARE
   [declaration_section
            variable declarations;
            constant declarations;    ]
BEGIN
        [executable_section
                PL/SQL execute/subprogram body    ]
EXCEPTION
   [exception_section
                PL/SQL Exception block    ]
END;
```

Syntax Description

CREATE [OR REPLACE] TRIGGER trigger_name : Create a trigger with the given name. If already have overwrite the existing trigger with defined same name.

BEFORE | AFTER : Indicates when the trigger get fire. BEFORE trigger execute before when statement execute before. AFTER trigger execute after the statement execute.

[INSERT, UPDATE, DELETE [COLUMN NAME..] : Determines the performing trigger event. define more then one triggering event separated by OR keyword.

ON table_name : Define the table name to performing trigger event.

Referencing [OLD AS OLD | NEW AS NEW] : Give referencing to a old and new values of the data. :old means use existing row to perform event and :new means use executing new row to perform event. You can set referencing names user define name from old (or new).

FOR EACH ROW | FOR EACH STATEMENT : Trigger must fire when each row gets Affected (ROW Trigger). and fire only once when the entire sql statement is execute (STATEMENT Trigger).

WHEN Condition : Optional. Use only for row level trigger. Trigger fire when specified condition is satisfy.

Viva questions:

1. Explain 3 basic parts of a trigger.
2. What is the maximum number of triggers that can be applied to a single table?
3. what is trigger?what are the different types of triggers.
4. what is instead of trigger.
5. Can you use a commit statement within a database trigger? No

****

Exercise _13: SQL Sequence - Auto Increment

AIM: Use the CREATE SEQUENCE statement to create a sequence, which is a database object from which multiple users may generate unique integers.

Description:

Auto-increment allows a unique number to be generated when a new record is inserted into a table.A sequence is a database item that generates a sequence of integers.the integers generated by a sequence to populate a numeric primary key column.

CREATE SEQUENCE syntax:

```
CREATE SEQUENCE sequence_name
        [START WITH start_num]
        [INCREMENT BY increment_num]
        [ { MAXVALUE maximum_num } ]
        [ { MINVALUE minimum_num } ]
        [ { CYCLE | NOCYCLE } ]
        [ { CACHE cache_num | NOCACHE } ];
```

Where

The default start_num is 1.

The default increment number is 1.

minimum_num must be less than or equal to start_num, and minimum_num must be less than maximum_num.

maximum_num must be greater than or equal to start_num, and maximum_num must be greater than minimum_num.

CYCLE specifies the sequence generates integers even after reaching its maximum or minimum value.

When an ascending sequence reaches its maximum value, the next value generated is the minimum.

When a descending sequence reaches its minimum value, the next value generated is the maximum.

NOCYCLE specifies the sequence cannot generate any more integers after reaching its maximum or minimum value.

NOCYCLE is the default.

CACHE cache_num specifies the number of integers to keep in memory.

The default number of integers to cache is 20.

The minimum number of integers that may be cached is 2.

initial-value specifies the starting value of the Sequence, increment-value is the value by which sequence will be incremented and maxvalue specifies the maximum value until which sequence will increment itself. cycle specifies that if the maximum value exceeds the set limit, sequence will restart its cycle from the begining. No cycle specifies that if sequence exceeds maxvalue an error will be thrown.

Example to create Sequence

```
SQL> CREATE SEQUENCE student_seq
        START WITH     501
                INCREMENT BY   1
        NOCACHE
        NOCYCLE;
        Sequence created.
```

Viva questions:

1. What is a sequence?
2. Trying to create a sequence in Oracle that starts with the max value from a Specific table. Why does this not work?
3. While creating a sequence, what does cache and nocache options mean?
4. With respect to a sequence, the cache option specifies how many sequence Values will be stored in memory for faster access.
5. How to generate Sequence Numbers.

****

Exercise _14: SQL - JOINS

AIM : JOINS IN ORACLE-different joins in oracle with examples

Description:

1. The purpose of a join is to combine the data across tables.
2. A join is actually performed by the where clause which combines the specified rows of tables.
3. If a join involves in more than two tables then Oracle joins first two tables based on the joins condition and then compares the result with the next table and so on.

TYPES

1 Equi join
2 Non-equi join
3 Self join
4 Natural join
5 Cross join
6 Outer join
  - Left outer
  - Right outer
  - Full outer
7 Inner join
8 Using clause
9 On clause

Assume that we have the following tables.

```
SQL> select * from dept;
DEPTNO     DNAME           LOC
10         INVENTORY       HYBD
20         FINANCE         BGLR
30         HR              MUMBAI
SQL> select * from emp;
EMPNO      ENAME           JOB        MGR        DEPTNO
111        SAKETH          ANALYST    444        10
222        SUDHA           CLERK      333        20
```

333	JAGAN	MANAGER	111	10
444	MADHU	ENGINEER	222	40

1. EQUI JOIN:A join which contains an equal to '=' operator in the joins condition.

An equijoin is a join with a join condition containing an equality operator. An equijoin combines rows that have equivalent values for the specified columns

2. NON-EQUI JOIN: A join which contains an operator other than equal to '=' in the joins condition.

3. SELF JOIN:Joining the table itself is called self join.A self join is a join of a table to itself. This table appears twice in the FROM clause and is followed by table aliases that qualify column names in the join condition. To perform a self join, Oracle Database combines and returns rows of the table that satisfy the join condition.

4.NATURAL JOIN:Natural join compares all the common columns.

5. CROSS JOIN:This will gives the cross product.

6. OUTER JOIN: Outer join gives the non-matching records along with matching records.

An outer join extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

•To write a query that performs an outer join of tables A and B and returns all rows from A (a left outer join), use the LEFT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of B in the join condition in the WHERE clause. For all rows in A that have no matching rows in B, Oracle Database returns null for any select list expressions containing columns of B.

•To write a query that performs an outer join of tables A and B and returns all rows from B (a right outer join), use the RIGHT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of A in the join condition in the WHERE clause. For all rows in B that have no matching rows in A, Oracle returns null for any select list expressions containing columns of A.

•To write a query that performs an outer join and returns all rows from A and B, extended with nulls if they do not satisfy the join condition (a full outer join), use the FULL [OUTER] JOIN syntax in theFROM clause.

You can use outer joins to fill gaps in sparse data. Such a join is called a partitioned outer join and is formed using the query_partition_clause of the join_clause syntax. Sparse data is data that does not have rows for all possible values of a dimension such as time or department. For example, tables of sales data typically do not have rows for products that had no sales on a given date. Filling data gaps is useful in situations where data sparsity complicates analytic computation or where some data might be missed if the sparse data is queried directly.

LEFT OUTER JOIN

This will display the all matching records and the records which are in left hand side table those that are not in right hand side table.

RIGHT OUTER JOIN:This will display the all matching records and the records which are in right hand side table those that are not in left hand side table.

FULL OUTER JOIN:This will display the all matching records and the non-matching records from both tables.

```
SQL> select empno,ename,job,dname,loc from emp e full outer join dept d
on(e.deptno=d.deptno);
EMPNO     ENAME      JOB        DNAME         LOC
333       JAGAN      MANAGER    INVENTORY     HYBD
111       SAKETH     ANALYST    INVENTORY     HYBD
222       SUDHA      CLERK      FINANCE       BGLR
444       MADHU      ENGINEER   HR            MUMBAI
```

7. INNER JOIN: This will display all the records that have matched. An inner join (sometimes called a simple join) is a join of two or more tables that returns only those rows that satisfy the join condition.

Viva questions:

1. What is join ?what are different types of joins.

2. What is cross join?

3. What is difference between Cartesian Join and Cross Join?

4. What is a OUTER JOIN

5. What are the guidelines for joins?

6. What is a JOIN? Explain types of JOIN in Oracle.

****