**Q=Difference betrween procedure and function?**

Basic Difference

1. Function must return a value but in Stored Procedure it is optional( Procedure can return zero or n values).
2. Functions can have only input parameters for it whereas Procedures can have input/output parameters.
3. Functions can be called from Procedure whereas Procedures cannot be called from Function.

Advance Difference

1. Procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
2. Procedures can not be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
3. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.
4. Functions that return tables can be treated as another rowset. This can be used in JOINs with other tables.
5. Inline Function can be though of as views that take parameters and can be used in JOINs and other Rowset operations.
6. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
7. We can go for Transaction Management in Procedure whereas we can't go in Function.

**Q=** What is cursor?

**Ans=** 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. The set of rows the cursor holds is called the *active*set.

There are two types of cursors in PL/SQL:

## Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the **SQL cursor**, which always has attributes such as **%FOUND, %ISOPEN, %NOTFOUND**, and **%ROWCOUNT**. The SQL cursor has additional attributes, **%BULK\_ROWCOUNT** and **%BULK\_EXCEPTIONS**, designed for use with the **FORALL** statement.

Explicit Cursors

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;

Working with an explicit cursor includes the following steps −

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

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example −

CURSOR c\_customers IS

SELECT id, name, address FROM customers;

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows −

OPEN c\_customers;

## Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example, we will fetch rows from the above-opened cursor as follows −

FETCH c\_customers INTO c\_id, c\_name, c\_addr;

## Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close the above-opened cursor as follows −

CLOSE c\_customers;

Question 1: SQL Query to find second highest salary of Employee?

select MAX(Salary) from Employee WHERE Salary NOT IN (select MAX(Salary) from Employee );

Q= nth height salary?

SELECT \* /\*This is the outer query part \*/

FROM Employee Emp1

WHERE (N-1) = ( /\* Subquery starts here \*/

SELECT COUNT(DISTINCT(Emp2.Salary))

FROM Employee Emp2

WHERE Emp2.Salary > Emp1.Salary)

# **PROCEDURE**

A **procedure** is a group of PL/SQL statements that you can call by name. The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks.

The procedure contains a header and a body.

* **Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.
* **Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

## How to pass parameters in procedure:

When you want to create a procedure or function, you have to define parameters .There is three ways to pass parameters in procedure:

1. **IN parameters:**The IN parameter can be referenced by the procedure or function. The value of the parameter cannot be overwritten by the procedure or the function.
2. **OUT parameters:**The OUT parameter cannot be referenced by the procedure or function, but the value of the parameter can be overwritten by the procedure or function.
3. **INOUT parameters:**The INOUT parameter can be referenced by the procedure or function and the value of the parameter can be overwritten by the procedure or function.

# **PL/SQL Function**

The PL/SQL Function is very similar to PL/SQL Procedure. The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value. Except this, all the other things of PL/SQL procedure are true for PL/SQL function too.

Question 2: SQL Query to find Max Salary from each department.

SELECT DeptID, MAX(Salary) FROM Employee  GROUP BY DeptID.

Question 3: Write SQL Query to display the current date.

SELECT GetDate();

**Question 4: Write an SQL Query to check whether date passed to Query is the date of given format or not**.

SELECT  ISDATE('1/08/13') AS "MM/DD/YY";

Question 5: Write an SQL Query to print the name of the distinct employee whose DOB is between 01/01/1960 to 31/12/1975.

SELECT DISTINCT EmpName FROM Employees WHERE DOB  BETWEEN ‘01/01/1960’ AND ‘31/12/1975’;

**Question 11: Write SQL Query to find duplicate rows in a database? and then write SQL query to delete them?**

SELECT \* FROM emp a WHERE rowid = (SELECT MAX(rowid) FROM EMP b WHERE a.empno=b.empno)

to Delete:

DELETE FROM emp a WHERE rowid != (SELECT MAX(rowid) FROM emp b WHERE a.empno=b.empno);

**Question 13: How do you find all employees which are also manager? .**

SELECT e.name, m.name FROM Employee e, Employee m WHERE e.mgr\_id = m.emp\_id;

Read more: <http://www.java67.com/2013/04/10-frequently-asked-sql-query-interview-questions-answers-database.html#ixzz4x2SWezGL>

Read more: <http://www.java67.com/2013/04/10-frequently-asked-sql-query-interview-questions-answers-database.html#ixzz4x2SThjck>  
  
Read more: <http://www.java67.com/2013/04/10-frequently-asked-sql-query-interview-questions-answers-database.html#ixzz4x2QXs7hN>  
  
Read more: <http://www.java67.com/2013/04/10-frequently-asked-sql-query-interview-questions-answers-database.html#ixzz4x2QUm8P1>

**5. When are we going to use truncate and delete?**

1. TRUNCATE is a DDL command, whereas DELETE is a DML command.
2. We can’t execute a trigger in case of TRUNCATE whilst with DELETE, we can accomplish a trigger.
3. TRUNCATE is quicker than DELETE, for the reason that when we use DELETE to delete the data, at that time it store the whole statistics in the rollback gap on or after where we can get the data back after removal. In case of TRUNCATE, it will not store data in rollback gap and will unswervingly rub it out. TRUNCATE do not recover the deleted data.
4. We can use any condition in WHERE clause using DELETE but it is not possible with TRUNCATE.5.If a table is referenced by any foreign key constraints, then TRUNCATE won’t work.

Q=2 Ways to find Nth highest salary in SQL - Oracle, MSSQL and MySQL

Ans=SELECT name, salary FROM #Employee e1 WHERE N-1 = (SELECT COUNT(DISTINCT salary) FROM #Employee e2 WHERE e2.salary > e1.salary)

SELECT name, salary FROM #Employee e1 WHERE N-1 = (SELECT COUNT(DISTINCT salary) FROM #Employee e2 WHERE e2.salary > e1.salary)SELECT name, salary FROM #Employee e1 WHERE 2-1 = (SELECT COUNT(DISTINCT salary) FROM #Employee e2 WHERE e2.salary > e1.salary)

Q=

create table Table100 (id integer);

insert into Table100 values(1);

insert into Table100 values(1);

insert into Table100 values(1);

create table Table200 (id integer);

insert into Table200 values(1);

insert into Table200 values(1);

insert into Table200 values(1);

select \* from Table100;

select \* from Table200;

equi join:

select t1.\*,t2.\* from Table100 t1,Table200 t2 where t1.id = t2.id; 🡪 9 records

cotison product :

select t1.\*,t2.\* from Table100 t1,Table200 t2 --🡪 9 records

**Q:Difference between union and union all?**

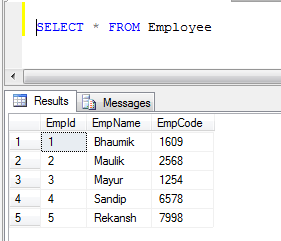
Both UNION and UNION ALL concatenate the result of two different SQLs. They differ in the way they handle duplicates.

* UNION performs a DISTINCT on the result set, eliminating any duplicate rows.
* UNION ALL does not remove duplicates, and it therefore faster than UNION.

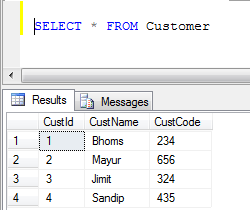
**Note:** While using this commands all selected columns need to be of the same data type.

Example: If we have two tables, 1) Employee and 2) Customer

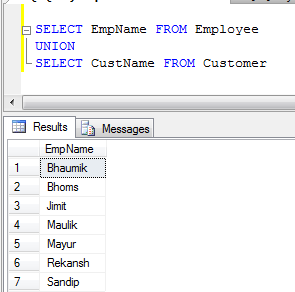
1. Employee table data:



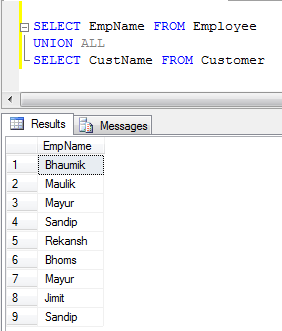
1. Customer table data:



1. UNION Example (It removes all duplicate records):



1. UNION ALL Example (It just concatenate records, not eliminate duplicates, so it is faster than UNION):



What is joins how many types of join is available in sql?