SQL:

Q: How to remove duplicate rows from table in Oracle?

Ans:

DELETE FROM EMP\_TEMP\_DUPS

WHERE rowid not in

(SELECT MIN (rowid)

FROM EMP\_TEMP\_DUPS

GROUP BY id,firstname,lastname,age,salary);

Q: How can you display top two employee salaries group by department in oracle?

select

d.Salary

,d.Department

from

(

select

r.Salary

,r.Department

,row\_number() over(

partition by r.Department

order by r.Salary desc) as RowNumber

from HumanResources as r

) as d

where d.RowNumber < 3

# [**Finding duplicate values in a SQL table**](https://stackoverflow.com/questions/2594829/finding-duplicate-values-in-a-sql-table)

ID NAME EMAIL

1 John asd@asd.com

2 Sam asd@asd.com

3 Tom asd@asd.com

4 Bob bob@asd.com

5 Tom asd@asd.com

Ans:

SELECT

name, email, COUNT(\*)

FROM

users

GROUP BY

name, email

HAVING

COUNT(\*) > 1

Q: How to find top three highest salary in emp table in oracle?

SELECT \*FROM

(

SELECT \*FROM emp

ORDER BY Salary desc

)

WHERE rownum <= 3

ORDER BY Salary ;

SELECT \* FROM

(

SELECT EMPLOYEE, LAST\_NAME, SALARY,

RANK() OVER (ORDER BY SALARY DESC) EMPRANK

FROM emp

)

WHERE emprank <= 3;

# Q: [Highest Salary in each department](https://stackoverflow.com/questions/8477040/highest-salary-in-each-department)

Table Name EmpDetails

DeptID EmpName Salary

Engg Sam 1000

Engg Smith 2000

HR Denis 1500

HR Danny 3000

IT David 2000

IT John 3000

Ans:

SELECT DeptID, MAX(Salary) FROM EmpDetails GROUP BY Dept

Q: SQL User Defined Functions?

CREATE FUNCTION <FUNCTION\_NAME> (@InputParameterName type ….)

RETURNS VARCHAR2(50)

AS

BEGIN

DECLARE @OutParameter

SET @OutParameter=(Select <COLUMN\_Name> from <TABLE\_NAME>

Where <Condition=@InputParameterName>)

RETURN @OutParameter

END;

Invoke User defined Function:

Select <User\_Defined\_Function>(Parameters);

/\*

CREATE TABLE testDATA (EmployeeID int, LoginID nvarchar(50))

insert into testDATA

select 1, 'database\userid1' union

select 2, 'database\userid2' union

select 3, 'database\userid3'

CREATE FUNCTION dbo.fx\_IDtest

(@EmployeeID int)

RETURNS nvarchar(50)

AS

BEGIN

DECLARE @UserID nvarchar(50)

SET @UserID = (SELECT SUBSTRING(LoginID,CHARINDEX('\',LoginID)+1, LEN(LoginID))

FROM testDATA

WHERE EmployeeID = @EmployeeID

)

RETURN @UserID

END

\*/

SELECT \*, dbo.fx\_IDtest(EmployeeID)

FROM testDATA

WHERE EmployeeID = 1

SELECT dbo.fx\_IDtest(1)

Both select statements, give me userid1 as a return.

Notice, that you must provide the schema for UDF's in your SELECT, this is often overlooked. You cannot do SELECT fx\_IDtest(1) instead you must do SELECT dbo.fx\_IDtest(1)

Q: Finding duplicate values in a SQL table?

Question:

t's easy to find duplicates with one field:

SELECT name, COUNT(email)

FROM users

GROUP BY email

HAVING COUNT(email) > 1

So if we have a table

ID NAME EMAIL

1 John asd@asd.com

2 Sam asd@asd.com

3 Tom asd@asd.com

4 Bob bob@asd.com

5 Tom asd@asd.com

This query will give us John, Sam, Tom, Tom because they all have the same email.

However, what I want is to get duplicates with the same email and name.

That is, I want to get "Tom", "Tom".

Ans:

The reason I need this: I made a mistake, and allowed to insert duplicate name and email values. Now I need to **remove/change** the duplicates, so I need to find them first.

SELECT

name, email, COUNT(\*)

FROM

users

GROUP BY

name, email

HAVING

COUNT(\*) > 1

Simply group on both of the columns.

Note: the ANSI standard is to have all non aggregated columns in the GROUP BY. MySQL allows you to avoid this, but results are unpredictable:

Q: What is a stored procedure? How do they work? What is the make-up of a stored procedure (things each must have to be a stored procedure)?

Ans1:

Stored procedures are a batch of SQL statements that can be executed in a couple of ways. Most major DBMs support stored procedures; however, not all do. You will need to verify with your particular DBMS help documentation for specifics. As I am most familiar with SQL Server I will use that as my samples.

To create a stored procedure the syntax is fairly simple:

CREATE PROCEDURE <owner>.<procedure name>

<Param> <datatype>

AS

<Body>

So for example:

CREATE PROCEDURE Users\_GetUserInfo

@login nvarchar(30)=null

AS

SELECT \* from [Users]

WHERE ISNULL(@login,login)=login

A benefit of stored procedures is that you can centralize data access logic into a single place that is then easy for DBA's to optimize. Stored procedures also have a security benefit in that you can grant execute rights to a stored procedure but the user will not need to have read/write permissions on the underlying tables. This is a good first step against SQL injection.

Stored procedures do come with downsides, basically the maintenance associated with your basic [CRUD](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) operation. Let's say for each table you have an Insert, Update, Delete and at least one select based on the primary key, that means each table will have 4 procedures. Now take a decent size database of 400 tables, and you have 1600 procedures! And that's assuming you don't have duplicates which you probably will.

This is where using an [ORM](https://stackoverflow.com/questions/1279613/what-is-an-orm-and-where-can-i-learn-more-about-it) or some other method to auto generate your basic CRUD operations has a ton of merit.

Execute Stored Procedure:

EXEC <Procedure\_Name>;

Ans2:

A stored procedure is a set of precompiled SQL statements that are used to perform a special task.

Example: If I have an Employee table

Employee ID Name Age Mobile

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

001 Sidheswar 25 9938885469

002 Pritish 32 9178542436

First I am retrieving the Employee table:

Create Procedure Employee details

As

Begin

Select \* from Employee

End

To run the procedure on SQL Server:

Execute Employee details

--- (Employee details is a user defined name, give a name as you want)

Then second, I am inserting the value into the Employee Table

Create Procedure employee\_insert

(@EmployeeID int, @Name Varchar(30), @Age int, @Mobile int)

As

Begin

Insert Into Employee

Values (@EmployeeID, @Name, @Age, @Mobile)

End

To run the parametrized procedure on SQL Server:

Execute employee\_insert 003,’xyz’,27,1234567890

--(Parameter size must be same as declared column size)

Example: @Name Varchar(30)

In the Employee table the Name column's size must be varchar(30).

Q: How to Execute SQL Server Stored Procedure in SQL Developer?

EXEC proc\_name 'paramValue1' 'paramValue2'

When I run this as either a statement or a script, I get this error:

Error starting at line 1 in command:

EXEC proc\_name 'paramValue1' 'paramValue2'

Error report:

Incorrect syntax near the keyword 'BEGIN'.

Ans:

You don't need EXEC clause. Simply use

proc\_name paramValue1, paramValue2

(and you need commas as Misnomer mentioned)

CREATE or replace PROCEDURE EMP\_TEMP\_PROC\_New

IS

No\_Of\_Records Number

BEGIN

SELECT count(\*) into No\_Of\_Records from EMP\_TEMP ;

--dbms\_output.enable();

dbms\_output.put\_line(No\_Of\_Records);

END EMP\_TEMP\_PROC\_New;

set serveroutput on;

exec EMP\_TEMP\_PROC\_New;

Q: Creating a trigger on Oracle 11g?

You can use :new in your trigger to reference the values being inserted, for example

create or replace trigger <trigger\_name>

after insert on <table\_name>

for each row

declare

l\_id number;

begin

select :new.id into l\_id from dual;

-- now l\_id contains the id of the inserted row, do what you want with it

end;

Don't take the example to literally; you don't have to first select :new.id into a variable, you can use it directly in SQL inside the trigger. I did it here just for illustration.

Take a look at the Oracle docs: [Coding Triggers](http://docs.oracle.com/cd/B19306_01/appdev.102/b14251/adfns_triggers.htm)

However, you might also want to take a look at some arguments why you should think twice if you really need to put your logic into triggers: [The Trouble with Triggers](http://www.oracle.com/technetwork/issue-archive/2008/08-sep/o58asktom-101055.html)

Ans2:

CREATE or REPLACE TRIGGER myTrigger

AFTER UPDATE OR INSERT ON product

REFERENCING NEW AS NEW

FOR EACH ROW

BEGIN

INSERT INTO h\_product

(

H\_PRODUCT\_ID,

PRODUCT\_ID

)

VALUES

(

seq\_h\_product.nextval,

:new.product\_id

);

END;

/

Q: Functions, Procedure & Triggers - how are they different, and when to use?

Ans:1

Procedures doesn't return any values their just get parameters and do something with them, functions does the same by their also can return you a value based on their work. Triggers are kind of event handlers that react on any action you want and start procedure when this action happens. For example, you can create trigger on select on some table, and when someone does select from this table you can write about this action in some log table.

Ans2:

* Functions and Procedures: basically just a piece of code that you run at will. In any language Function will return a value (eg' number of rows updated, a string etc') and a Procedure will not return any value.
* Triggers: Are pieces of code that run because of an event; For example you have a table and you would like that after every insert to this table, you will get an email - Then you define an AFTER INSERT trigger ON myImportant table and tell it to send you an email with the contents of the recent Insert.

A trigger can and probably will use functions and procedures.

->Functions can be used in SELECT statement, whereas Procedures cannot.

You already found the main difference. You create a function if you want to use it in SQL. You create a procedure, when you want to use it only in PL/SQL.

Q: Functions vs procedures in Oracle?

can anybody explain what is the main difference between functions and procedures in Oracle? Why must I use procedures if I can do everything with functions?

If I cannot call procedure in sql statement, ok, I'll write a function to do the same work.

Procedures don't return values, ok, I'll return only sql%rowcount or 1(success), 0(exception) after any dml operation

Both procedures and functions can pass variables to calling environment via OUT/IN OUT parameters

I heard that the main difference is in performance, 'procedures are faster than functions'. But without any detail.

Ans:

The difference is- A function must return a value (of any type) by default definition of it, whereas in case of a procedure you need to use parameters like OUT or IN OUT parameters to get the results. You can use a function in a normal SQL where as you cannot use a procedure in SQL statements.

Some Differences between Functions and Procedures

1. A function always returns a value using the return statement while a procedure may return one or more values through parameters or may not return at all.Although, OUT parameters can still be used in functions, they are not advisable neither are there cases where one might find a need to do so. Using OUT parameter restricts a function from being used in a SQL Statement.
2. Functions can be used in typical SQL statements like SELECT, INSERT, UPDATE, DELETE, MERGE, while procedures can't.
3. Functions are normally used for computations where as procedures are normally used for executing business logic.
4. Oracle provides the provision of creating "[Function Based Indexes](http://oracle-base.com/articles/8i/function-based-indexes.php)" to improve the performance of the subsequent SQL statement. This applies when performing the function on an indexed column in where clause of a query.

More Information on Functions Vs. Procedures [here](http://careerride.com/Oracle-function-vs-procedure.aspx) and [here](http://docs.oracle.com/cd/B25329_01/doc/appdev.102/b25108/xedev_programs.htm).