**SQL**

Oracle apex -> getting started -> request a free workspace.

Workspace : hrb\_workspace

UserName : [harshalborgude933@gmail.com](mailto:harshalborgude933@gmail.com)

Password : V

( <https://apex.oracle.com/pls/apex/> )

SQL Workshop -> Utilities -> Sample Datasets -> EMP / DEPT install

( using this option we will get readymade EMP or DEPT tables )

2) <https://livesql.oracle.com/apex/f?p=590:1000> - to study and practice sql – analytical functions

Code library -> analytic functions -> Analytic Functions: Databases for Developers(chris saxon)

URL : https://livesql.oracle.com/apex/f?p=590:1:1827647202351::NO:::

User : [harshalborgude933@gmail.com](mailto:harshalborgude933@gmail.com)

Pass : V

* **Select :**

1. Select col1, col2, col3 from table1;

* will return col1,col2,col3 from table1.

1. select distinct job from emp;

* this will return unique values from job colm.
* **Select + Where :**

1. select empno,ename from emp where job='MANAGER';

* return empno and ename colm’s where job clmn have “MANAGER” word from emp table.
* If data is text then it should be in single quote.

1. select empno from emp where sal='5000';

* return empno from emp table which have sal equal to 5000.
* AND :

1. select \* from emp where sal=2850 and empno=7698 and mgr=7839;

* return all clmn’s where sal , emp and mgr matches same.
* **Operator :**

1. **Not ( ! ) :**

* select \* from emp where job !='SALESMAN';
* will return data where job is not equals to SALESMAN.

1. **Less Than (**  small **<** large **) / greater Than (** large **>** small**) / GTequalsTo ( >= ) / LTequalsTo ( <= ) :**

* select \* from emp where job !='SALESMAN' and sal<5000 and deptno>=10;
* will return data where job is not SALESMAN and sal is less than 5000 and deptno is greater that 10.
* select \* from emp where comm > sal;
* will return data where comm colm is greather than sal colm.

**Q . Get all employees that are not managers and have a salary greater than 2500 and also work in department no =20.**

* select \* from emp where job!='MANAGER' and sal>2500 and deptno=20;
* **Where + And + Or :**

1. **select \* from emp where job='MANAGER' or job='ANALYST';**

* return emp who’s job is manager or analyst.

Q . The name of those employees that are not managers nor salesmans and have salary greater than or equals to 2000.

* select ename from emp where job!='MANAGER' and job!='SALESMAN' and sal>=2000;

Q . WAQ ( write a query ) to Return ENAME and HIREDATE where deptNo is 20 and 30.

* select ename,hiredate from emp where deptno=20 or deptno=30;
* **Between , IN and Null :**
* **In** ( num , strings etc ) - in ( a , b )
* Q . WAQ ( write a query ) to Return ENAME and HIREDATE where deptNo is 20 and 30.
* Select ename,hiredate from emp where deptno in (20,30)
* select ename,hiredate from emp where deptno=20 or deptno=30;
* Q. waq to get details of ford,smith,allen , ward , martin
* select \* from emp where ename in ('FORD' ,'SMITH' ,'ALLEN' ,'MARTIN');
* select \* from emp where ename not in ('FORD' ,'SMITH' ,'ALLEN' ,'MARTIN');
* select \* from emp where ename='FORD' or ename='SMITH' or ename='ALLEN' or ename='MARTIN';
* **between**  ( numbers , texts , dates ) - ( between a and b ) :
* range of a and b is included
* WAQ to get list of hired employees between '06/09/1981' and '12/17/1981'.
* select \* from emp where hiredate between '06/09/1981' and '12/17/1981';
* WAQ to get list of hired employees between 2000 and 5000.
* select \* from emp where sal between 2000 and 5000;
* select \* from emp where sal not between 2000 and 5000;
* Null : ( is null )
* Waq to return data where comm is not null.
* select \* from emp where comm is not null;

Q. write a query that returns those employees that don’t make any commission and have a salary greater than 1100 but less than 5000. Exclude those employees that have a salary equals to 3000.

-> select \* from emp where ( comm is null or comm=0 ) and ( sal > 1100 and sal < 5000 and sal !=1500);

-> select \* from emp where ( comm is null or comm=0 and sal > 1100 and sal < 5000 and sal <>1500);

-> select \* from emp where comm is null or comm=0 and sal between 1100 and 5000 and sal !=3000;

Q. return those employees that are salesman and that makes either 300 dollars in commission or greater that 1000 dollars in commission.

-> select \* from emp where JOB='SALESMAN' and (comm=300 or comm>1000);

* Like ( pattern matching ) :
* select \* from emp where JOB like '%LES%';
* Aliasing :
* select ename as "employee name", hiredate as "hire date", sal as salery from emp ;
* Cancatenating ( || ) :
* select 'hello my name is '|| ename || ' and salery is '|| sal as salery from emp;

ans : hello my name is KING and salery is 5000

* ORDER BY ( ASC /\ , DESC \/ ) :
* select \* from emp order by sal desc;
* select deptno,sal,ename from emp order by deptno,sal;

First sorting with dept no and then will sort by saleries for particular dept.

===================================================================

**Assignments :** 

1. Write a query that retrieves suppliers that work in either Georgia or California.

* **Select \* from suppliers where state=’Georgia’ or state=’California’;**

1. Write a query that retrieves suppliers with the characters "wo" and the character "I" or "i" in their name.

* **Select \* from suppliers where supplier\_name like %wo% and (supplier\_name like %I% or %i%);**

1. Write a query that retrieves suppliers on which a minimum of 37,000 and a maximum of 80,000 was spent.

* **Select \* from supplier where total\_spent >=37,000 and total\_spent<=80000;**
* **Select \* from supplier where total\_spent between 37000 and 80000;**

===================================================================

**Single Row Function**

1. **sumThese(x,y,z) :**
2. **Concat() :**

* **Select ‘my name is ’ || ename from emp;**
* **select concat('my name is ',ename) from emp;**
* my name is KING

1. **upper(‘string’) / lower (‘STRING’): converts string to upper case**

* **select upper('myname') from emp;**

MYNAME

* **select concat(upper('my name is '),lower(ename)) from emp;**

MY NAME IS king

* **select concat(lower(ename),' IS THE NAME')||' and their job is: '||job from emp;**

king IS THE NAME and their job is: PRESIDENT

* **select concat(concat(lower(ename),' IS THE NAME'),concat(' and their job is: ',job)) as sentence from emp;**
* **select lower(ename) as lower\_ename from emp;**
* **select \* from emp where lower(job)='manager';**
* **select \* from emp where job=upper('manager');**

1. **INITCAP(‘stringone’) :**

* **select initcap('my name is harshal') as line from dual;**

My Name Is Harshal

1. **length(‘String’) :**

* **select length('my name is harshal') as line from dual;**

18

* **select ename , length(ename) as length from emp;**

king 4

* **select ename , length(ename) as length from emp where length(ename)=4;**

1. **substr(‘String’,2,3) :**

* **select substr('hello',2,2) from dual;**

**: el**

**: ( string , starting position of index to be extracted , num of character to extract )**

* **select substr('hello',2) from dual;**

**:** ello

* select 'Hello my name is' ,substr('hello my name is',10,5) from dual;

: Hello my name is -> name

1. **LPAD(‘String’,6,’$’) / RPAD ( ‘string’,6,’\*’) :**

( string , total width of string , char to be appended left/right )

* **select lpad('harshal',10,'\*') from dual;**

**:** \*\*\*harshal

* **select rpad('harshal',10,'\*') from dual;**

: harshal\*\*\*

1. **Ltrim / Rtrim :**

* **select ltrim('harshal','h') from dual;**

**:** arshal

* **select rtrim('harshal','al') as name from dual;**

**:** harsh

* **select ltrim('hhhhhharshal','h') as name from dual;**

: arshal

1. **ROUND ( x , y) :**

* **select round(100.3456,2) from dual;**

**: 100.35**

1. **Trunc (n , d ) :**

* **select trunc(100.3456,2) from dual;**

**: 100.34**

1. **SysDate : return todays date**

* **Select sysdate from dual;**

**:** 02/19/2022

1. **Systimestamp :**

* **select systimestamp from dual;**

**:** 19-FEB-22 12.38.59.411952 PM +00:00

1. **ADD\_MONTHS ( date , num )** :

* **select add\_months('02/19/2022',2) from dual;**

: 04/19/2022

* **select add\_months('02/19/2022', -3) from dual;**
* **select add\_months(sysdate,-1) from dual;**

**:** 01/19/2022

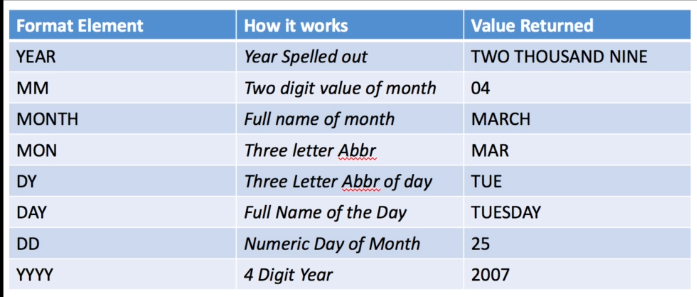
1. **Months\_between (date1, date2 ) :**

* **select MONTHS\_BETWEEN('06/19/2022', '01/19/2022') from dual;**

**:** 5

1. **to\_char(str , format) :**

* **select sysdate,to\_char(sysdate,'mm-dd-yyyy') from dual;**
* **select sysdate,to\_char(sysdate,'dd-mm-yyyy') from dual;**

****

* **select sysdate,to\_char(sysdate,'ddth "of" month , yyyy') from dual;**

**:** 19th of february , 2022

* **select sal,to\_char(sal,'$99,999.99') from emp;**

**:** $5,000.00

1. **To\_date(date, format ) :**

* **select to\_date('2012-08-27','yyyy-mm-dd') from dual;**

**:** 08/27/2012

**:** converting given date and its format into actual format , which is dd/mm/yyyy

1. **LAST\_DAY(*d) :***

last\_day is a date function that requires a date as an argument. It returns the last day of the month in which the given date falls. The argument is required for this function to work properly.

1. **NEXT\_DAY(d, c) :**

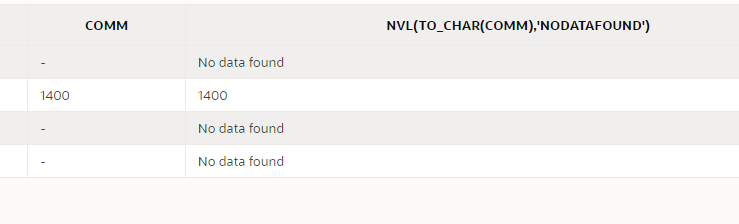
The first argument is the date and the second argument is a text reference to a day of the week. Both arguments are required for this function to work properly. This function returns a valid date representing the first occurrence of the *c* day following the date represented in *d*.

1. **NVL ( clm , string ) :**  fill empty cell with string

* select ename,job,sal,nvl(comm,0) from emp where empno in (7839,7698,7566,7654);

: empty cell from clm will filled with 0.

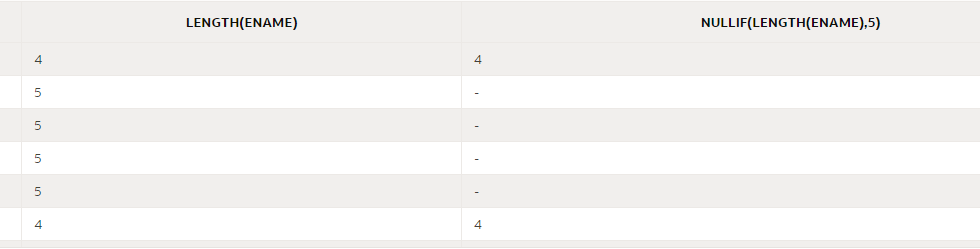
* **select ename,job,sal,comm ,nvl(to\_char(comm),'No data found') from emp where empno in (7839,7698,7566,7654);**



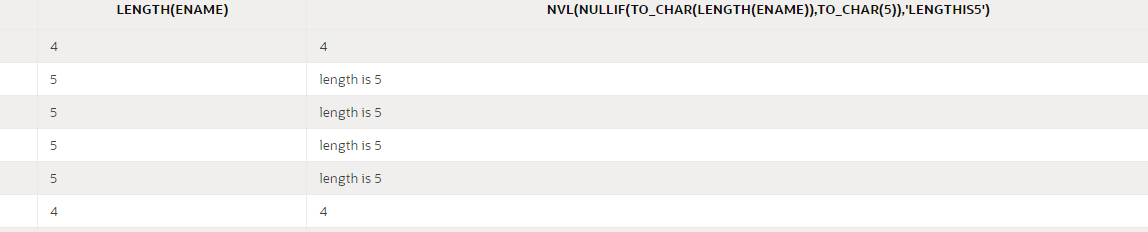
* to\_char will convert comm clm into char format and then nvl will fill empty rows with given string

1. **NULLIF( arg1 , arg2 ) :** null the row if both arg are matching

* **select ename , length(ename) , nullif(length(ename),5) from emp;**



* **select ename , length(ename) , nvl(nullif(to\_char(length(ename)),to\_char(5)),'length is 5') from emp;**



* nulling the row If length is 5 and converting them into char so

**Grouping Functions**

1. **MAX ( ):** return max amnt from clm

* **select max(sal) from emp;**

**: 5000**

1. **Min ( ) :** return min amnt from clm

* **select min(sal) from emp;**

: 800

1. **Sum ( ) :** return sum of the clm

* **select sum(sal) from emp;**

**:** 29025

1. **Avg( ) :**

* **select avg(sal) as avg\_sal from emp;**

**:** 2073.21428571428571428571428571428571429

* **select avg(sal) as avg\_sal from emp where job=’MANAGER’**

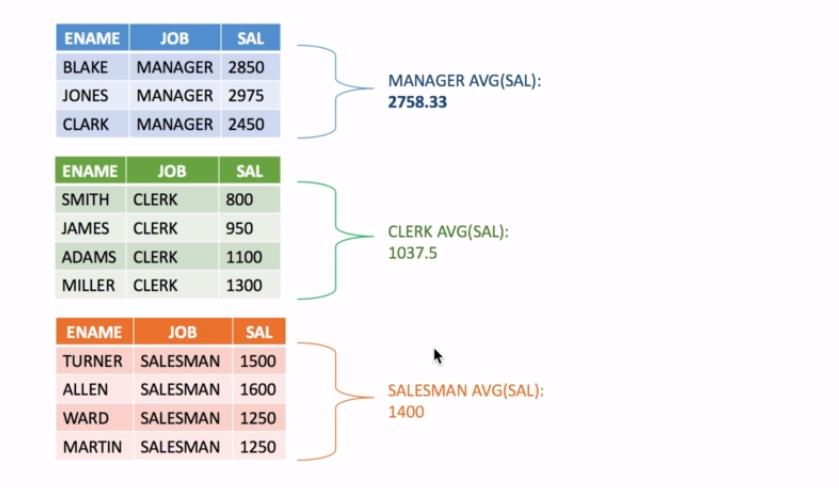
1. **Count (\*) :**

* **select count(ename) as count from emp; ->** 14
* **select Count(\*) from emp;**

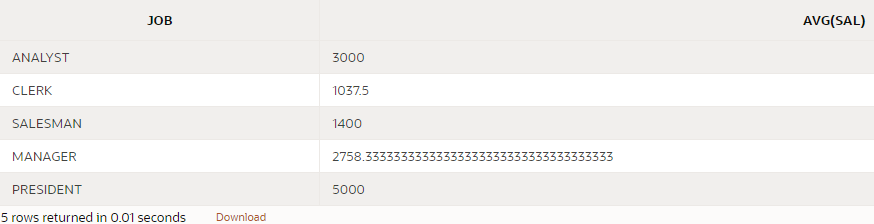
: 14

* **select sum(sal) as sum , avg(sal) as avg, max(sal) as max , min(sal) as min , count(\*) as count from emp;**

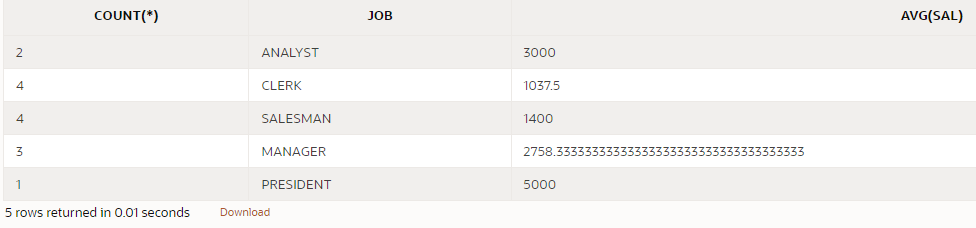
1. **GroupBy() :**



* **select job,avg(sal) from emp group by (job);**



* **select count(\*),job,avg(sal) from emp group by (job)**



* **No Group functions in the where clause is allowed if we are using group by. So We have to use “Having”**

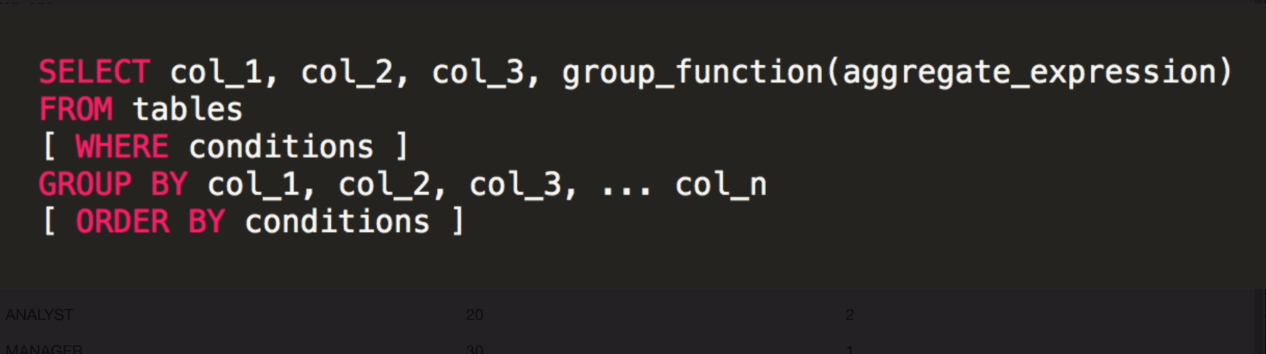
1. **Having :**

* select count(\*),job from emp group by job having count(\*)=2;

: 2 , analyst

**Q. Write a query which returns dept number having more than 3 emp;**

**-> select deptno,count(\*) from emp group by deptno having count(\*)>3;**



**Nested Queries**

* **select \* from dept where deptno=(select deptno from dept where deptno=30);**
* first nested query will get executed and then its result will be provided to first query and then based on that it will return the result.
* **select \* from emp where deptno=(select deptno from dept where loc='CHICAGO');**
* return all result from emp table where deptno is equals to location ‘CHICAGO’ from dept table.
* **select \* from emp where deptno in (select deptno from dept where deptno in (10,20));**
* returns all clms from emp where dept number is 10,20 from dept table.

**Select Within Select**

* **select \* from emp where deptno=(select deptno from dept where deptno in (10,20));**
* first query will return dept no and for only that dept numberst it will return all the columns from emp table.

**Joins – Relating Tables Together**

* We can use joins instead of subqueries

1. **SELECT \* from emp, dept WHERE emp.deptno = dept.deptno;**

* Will return all clms from emp where both emp and dept having same deptno.

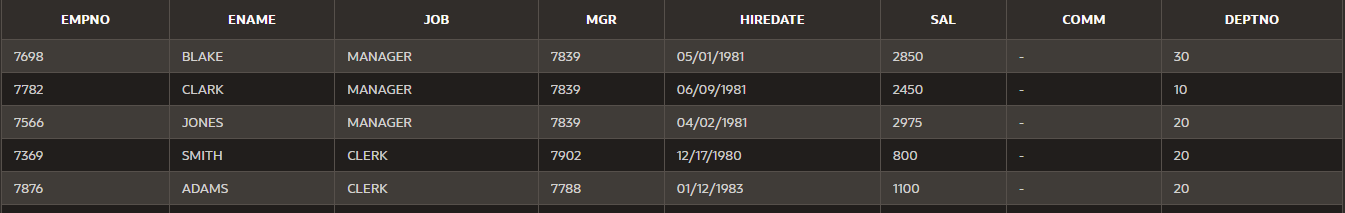
1. **SELECT e.ename as name, e.job as job, e.sal as sal from emp e, dept d WHERE e.deptno = d.deptno and d.loc = 'DALLAS';**

* Will return ename,job,sal from emp and dept table when dept no of emp and dept table is same and aslo loc is ‘DALLAS’.

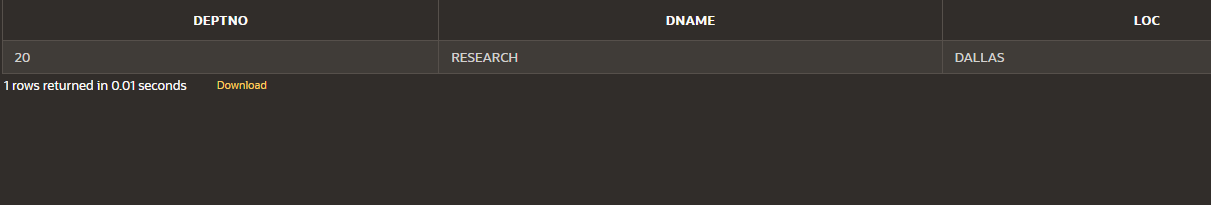
**Joins - SubQueries in from clause**

1. **SELECT e.ename as first\_name, job, sal, e.deptno from (select \* from emp where job in ('MANAGER', 'CLERK')) e, (SELECT \* from dept where loc = 'DALLAS') d WHERE e.deptno = d.deptno;**

* Basically it contains 3 queries , 2 subqueries written in from clause , first will return all clms who have job in manager and clerk ,second will return all clms who have location of DALLAS , and from them main query will return name , job , dept no ,sal of persons who are in same dept no from both tables.
* **select \* from emp where job in ('MANAGER', 'CLERK');**



* **SELECT \* from dept where loc = 'DALLAS';**



* **Final :**

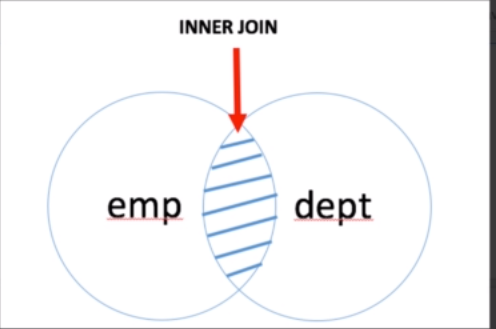


**JOINS – Inner Join**

INNER JOIN

1. **SELECT \* FROM emp INNER JOIN dept ON emp.deptno=dept.deptno;**

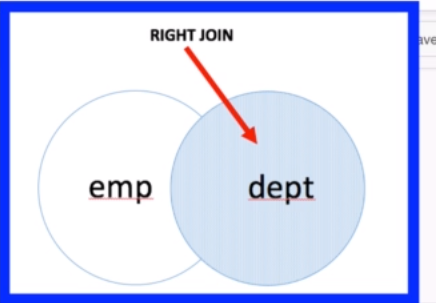
* Will return all the rows from both table who have same dept number.



RIGHT JOIN / RIGHT OUTER JOIN / (+) -> JOIN symbol

1. **SELECT \* FROM emp RIGHT JOIN dept ON emp.deptno=dept.deptno;**

* Will return all the data which matches dept no on both table also will return all the data from right table.



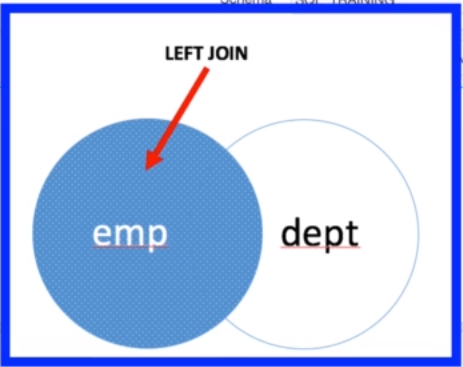
1. SELECT \* from emp, dept WHERE emp.deptno (+) = dept.deptno ;

* Here we are having inner join on dept no but also have (+) sign on right table so tells that right join is there and will return all clms rows from right table with inner join.

LEFT JOIN / LEFT OUTER JOIN

1. **SELECT \* FROM emp LEFT JOIN dept ON emp.deptno=dept.deptno;**

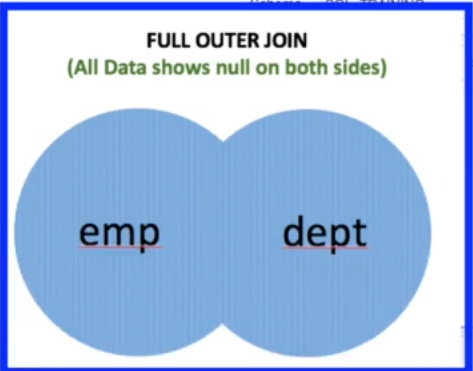
* Will return all the data from left table and also whichever matches to the right table with dept no.



**FULL OUTER JOIN**

1. **SELECT \* FROM emp FULL OUTER JOIN dept ON (emp.deptno = dept.deptno);**

* Whatever data is possible to join it will join on deptno but remaining it will return with blank clms.



1. **SELECT \* FROM (select \* from emp where job=’SALESMAN’) e FULL OUTER JOIN (select \* from dept) d ON e.deptno = d.deptno;**

* Will return clms from subqueries and that will be considered for joins.
* First subquery will return only those records which have job as salesman.
* It will consider all the clms from dept table.

**Exists and Not Exists Condition**

* **Exists :** It Works like if condition in query, if exist query return something then only mail query will return its data , if exists query didn’t return anything then main query will not return anything.
* **Not Exists :** It is exact reverse to the Exists statement result.

1. **SELECT \* FROM emp where exists (select \* from emp where job='MANAGER');**

* Here if exists query return anything then only main query will return its result , if exists query is not returning anything then main query will not return anything.
* If(existsQuery){

Return (Select \* from emp);

}

1. **SELECT d.\* FROM dept d WHERE NOT EXISTS (SELECT \* FROM emp WHERE d.deptno = emp.deptno) OR LOC = 'CHICAGO';**

* Return only those records which have deptno matches from emp and dept table , here uses alias d of outer table , to use in exists query , after getting result by comparing each row , again it will check loc and return the row which matches given string.

**SELF JOIN**

Using single table as 2 tables and correlating different clms with each other. In emp table , empno is employee number and MGR is managers employee number , managers are also present in employee no clms , so by self joining them , we can get employee names and there manager names.

1. **select e.ename as employee,m.ename as mangr from emp e,emp m where e.mgr=m.empno;**

* here emp table is having 2 aliases e and m , e.ename returns employee names and m.ename returns manager names , from same table . and employee tables manager num is compared with manager tables employee number , so in this way we will get employee names and manager names from same table.

**CROSS JOIN / NO JOIN**

* Cross join means no join.

1. **select \* from emp,dept;**
2. **select \* from emp cross join dept;**

* Here want to get return from 2 tables , 1 table have 10 records and 2nd have 4 records , so total is 14 , but this query will return 56 records. As each clm of first table sql will try to match with another table so result is multiplication of both . Here no join is there , This is also called as **Cross JOIN.**

**NATURAL JOIN**

* Natural Join is basically a inner join without specifying a column name.

1. **select deptno,ename,mgr from emp natural join dept;**

* return deptno , ename,mgr from emp and dept table who has same deptno as both table have deptno clm and Natural join compares that common clm.Same like inner join by providing clm names to compare.

**JOINs by “USING” clause**

* It works with join keywors , here we are providing clm names to be compared inside using clause. Can be used with inner , left,right,natural joins.

1. **select deptno,ename,mgr from emp join dept using (deptno);**

* return given clms by comparing deptno clm of both table.

1. **select deptno,ename,mgr from emp right join dept using (deptno);**

* will return clms matches with deptno value and also will return all the rows from dept table.

**EQUI JOIN**

* Equal symbol means equijoin.

1. **select emp.deptno,emp.ename,emp.mgr from emp ,dept where emp.deptno = dept.deptno;**

* This “=” symbol means it’s a EQUIJOIN.works same as inner join.

**NON EQUI JOIN**

* Whenever we don’t have any specific values in columns to join , and we cant be able to use any other kind of join then we can use “non equijoin”
* In non equijoin we can provide a range using between.

1. **select e.ename,e.sal , j.grade\_level from emp e JOIN job\_grade j ON e.sal BETWEEN j.lowest\_sal AND j.highest\_sal;**

* return ename , sal from emp and grade level from job\_grade table with gardes depending sal having range of lowest sal and highest sal from job\_grade table.

**CASE Statement**

* Syntax :

CASE [ expression ]

When condition1 then result1

When condition2 then result2

…

When condition then result\_n

Else result

END

1. **select ename,job, (CASE job**

**WHEN 'PRESIDENT' then 'big shot'**

**WHEN 'MANAGER' then 'decides the pay'**

**WHEN 'ANALYST' then 'good at maths'**

**WHEN 'CLERK' then 'hard working'**

**ELSE 'not comment'**

**END) as "COMMENT"**

**FROM emp;**

* return ename and job clm with that , depending upon job clm 3rd clm “COMMENT” is created depending upon the given conditions.
* Like when job having value of PRESIDENT then COMMENT clm will have value of big shot.
* If nothing there matching then will print whatever there in ELSE.
* Any kind of conditions we can place in when.
* Inside when condtion we can provide multiple conditions using “AND”,”Or” logic.



1. **select ename,job,sal, (CASE**

**WHEN sal >= 3000 then 'Earn Enouh'**

**ELSE 'Can earn more'**

**END) as "SUGGESTION"**

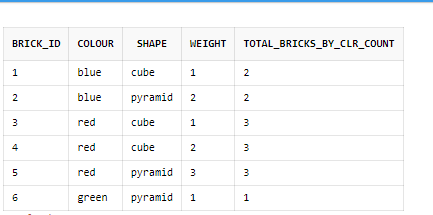
**FROM emp;**

* return ename,job,sal,SUGGESTION clms
* suggestion clm will be form based on sal , when sal is greater than 3000 then will return given string otherwise will show 2nd string.
* Writing else is optional.
* Can write n number of conditions.

**OVER clause with PARTITION BY**

1. **select b.\*,(select count(\*) from bricks where colour=b.colour) total\_bricks\_by\_clr\_count from bricks b;**

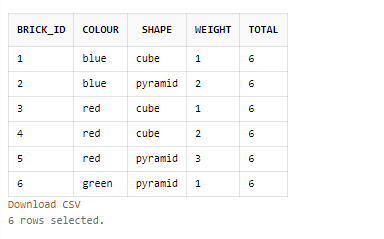
* will return all from from b table (b.\*) and subquery will return count from bricks table where colour = colour of outer query.
* This way we can manage to dummy over and partition by.



**OVER :**

1. **select b.\* ,count(\*) over() as total from bricks;**

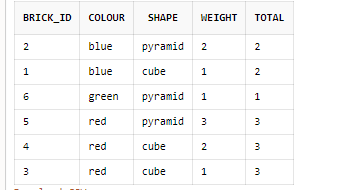
* will return all clms as well new one clm total which will have count of total items.
* It will run count operation over whole table and for every row will return total count.



**OVER (PARTITION BY) :**

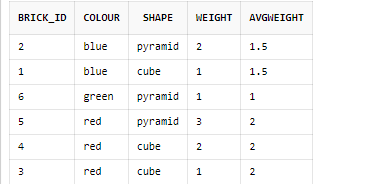
1. **select b.\* ,count(\*) over(partition by colour) as total from bricks b;**

* will return all clms with extra total clms , that have count and partitioned by colour clm.
* Count(\*) will apply on all the records having group of clr.



1. **select b.\* ,avg(weight) over(partition by colour) as avgWeight from bricks b;**

* will return all clms from bricks and also will return avgWeight clm which averages of weight on partion of colours.



1. **select b.\* ,**

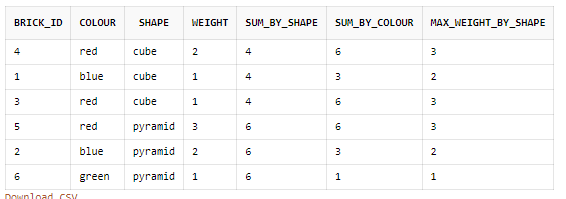
**sum(weight) over(partition by shape) sum\_by\_shape,**

**sum(weight) over(partition by colour) sum\_by\_colour,**

**max(weight) over(partition by colour) max\_weight\_by\_colour**

**from bricks b;**

* will return all clms with that other 3 clms for sum of weight by shape,colour and max weight by colour.
* Sum\_by\_shape will partition by shape and will calculate sum of weight for that partition.

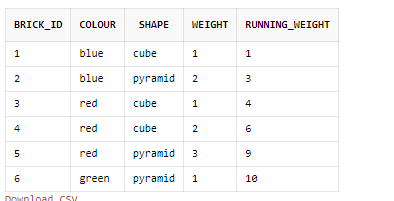


1. **select b.\* ,**

**sum(weight) over(order by brick\_id) running\_weight**

**from bricks b;**

* will return sum of weight , will keep adding weight of each row and at last row will show total weight.
* Bcz of ordering by brick\_id , it is keep adding each weight of row ,means as order is there so weights are added by order.
* If partition would be there then it would have added by partition.



1. Range between unbounded preceding and current row : This statement will get added by oracle by default when order by is there in over statement.

**select b.\* ,**

**sum(weight) over(**

**order by brick\_id** Range between unbounded preceding and current row **) running\_weight**

**from bricks b;**

**Unbounded preceding order in over clause**

1. **select b.\* ,**

**sum(weight) over(order by weight** range between unbounded preceding and current row**) running\_weight**

**from bricks b order by weight;**

1. **select b.\* ,**

**sum(weight) over(order by weight** rows between unbounded preceding and current row**) running\_weight**

**from bricks b order by weight;**



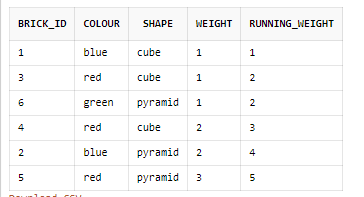
**Sliding Window**

1. **select b.\* ,**

**sum(weight) over(order by weight** rows between 1 preceding and current row**) running\_weight**

**from bricks b order by weight;**

* will sum for 1 preceding row only , so last 1 row and current row.
* We can change number of preceding rows.
* That’s why it is called as sliding window.
* Unbounded means , unbounded preceding rows.

****

**Rank functions**

row\_number() , rank() , dense\_rank()

1. **select brick\_id,weight ,**

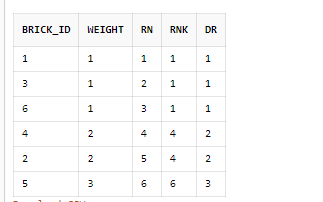
**row\_number() over(order by weight) rn,**

**rank() over(order by weight) rnk,**

**dense\_rank() over(order by weight) dr**

**from bricks;**

* row\_number will give numbers to rows.
* Rank will give **rank to row according to row number** , for same value it will give same previous rank.
* Dense rank will **assign rank considering given clm order or grouping** , but it follows proper ranking sequence.it give rank to the group of items in clm. it don’t rank based on row number like rank function.

****

* **Lead , Lag :**

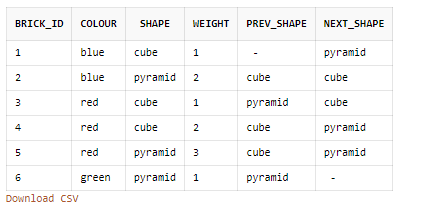
1. **select b.\*,**

**lag(shape) over(order by brick\_id) prev\_shape,**

**lead(shape) over(order by brick\_id) next\_shape**

**from bricks b;**

* lag(clm) will return previous name of same clm after ordering according to given clm.
* Lead(clm) will return next name of same clm after ordering according to given clm.
* If no previous item is there then it will be “–“



* **first\_value() , last\_value() :**

1. **select b.\*,**

**first\_value(weight) over(order by brick\_id) first\_weight\_by\_id,**

**last\_value(weight) over(order by brick\_id) last\_weight\_by\_id**

**from bricks b;**

* first value will give first value of weight of the coloumn after ordering by clm.
* Last value will give last value according to the window.
* To get perfect last value , can add string after order by – “range between current row and unbounded following”
* Here after “order be brick\_id” , that string can be added.
* “unbounded preceding” is for unbounding values of clm in previous places.
* “unbounded following” is for unbounding values of clm in next places.

