**Statements or Languages**

We have five types of statements

**1. Data Definition Language (DDL):**

* Create
* Alter
* Rename
* Drop
* Truncate

**2. Date Retrieval Language:**

* Select

**3. Data Manipulation Language (DML):**

* Insert
* Delete
* Update
* Merge

**4. Transaction Control Language (TCL):**

* Commit
* Rollback
* Savepoint

**5. Data Control Language (DCL):**

* Grant
* Revoke

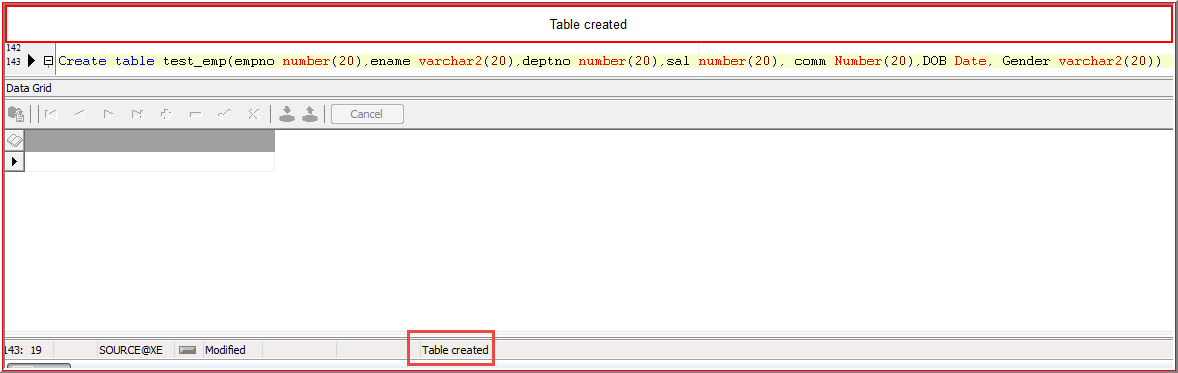
**Select, from => Keywords**

**Select empno, ename => select clause**

**From emp => from clause**

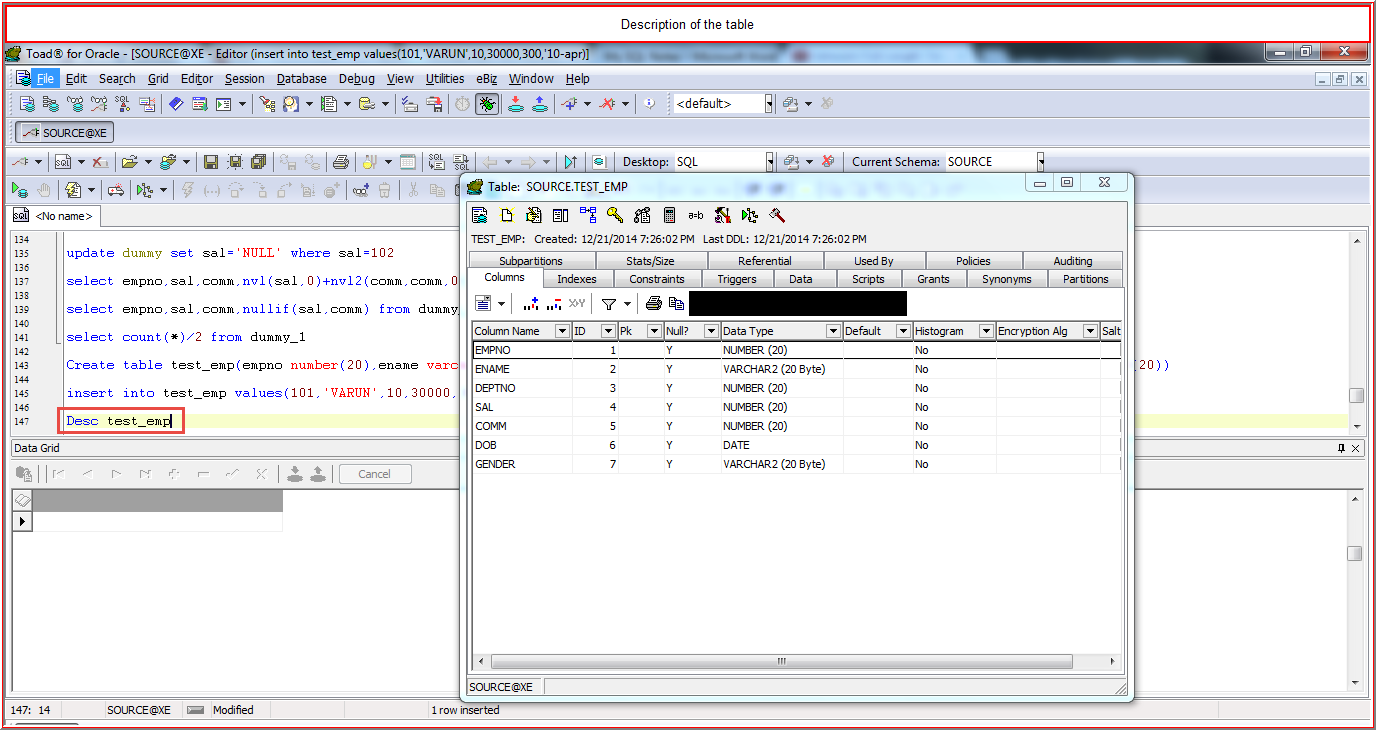
**Create:**

Create table emp(empno number(20),ename varchar2(20),deptno number(20),sal number(20), comm Number(20),DOB Date, Gender varchar2(20))

****

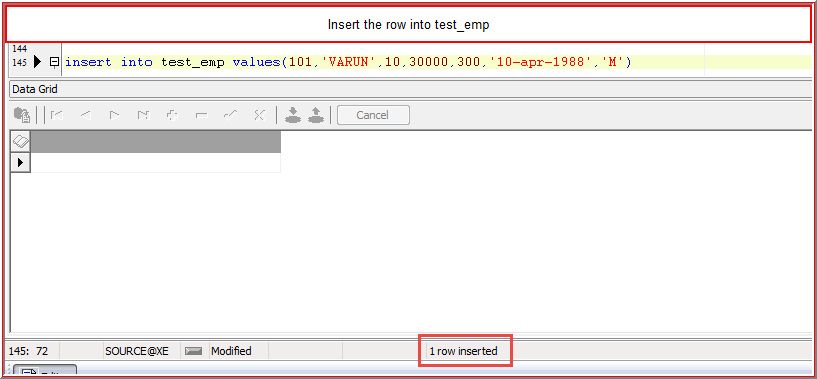
**Desc test\_emp or describe test\_emp**

Desc test\_emp

****

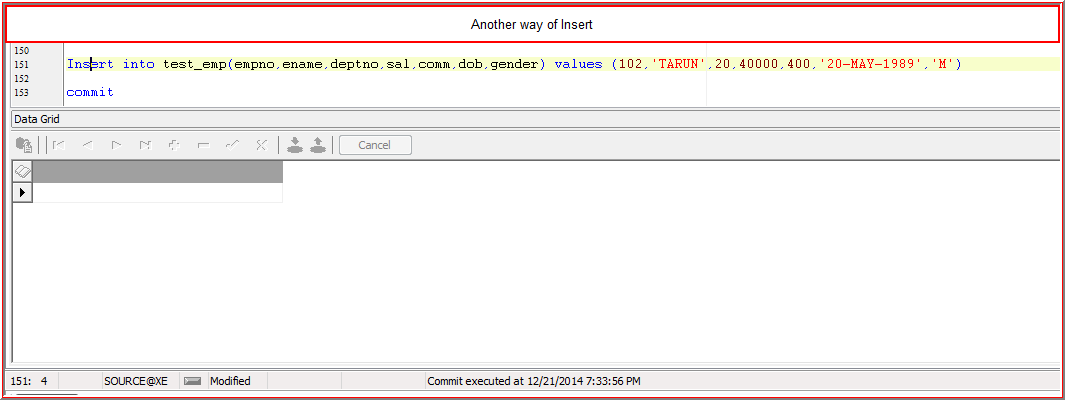
**Insert:**

**insert into test\_emp values(101,’VARUN’,10,30000,300,’10-apr-1988’,’M’)**



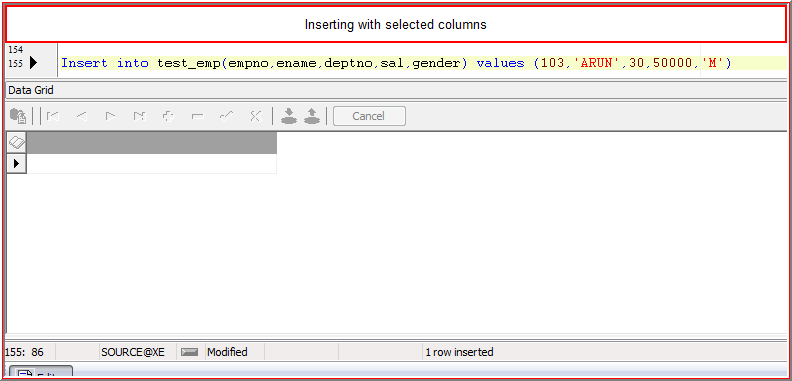
OR

**Insert into test\_emp(empno,ename,deptno,sal,comm,dob,gender) values (102,’TARUN’,20,40000,400,’20-MAY-1989’,’M’)**

****

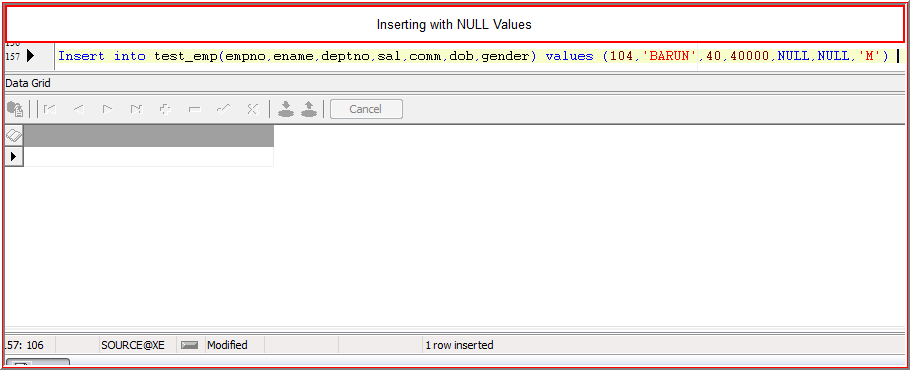
**OR**

**Insert into test\_emp(empno,ename,deptno,sal,gender) values (103,’ARUN’,30,50000,’M’)**

****

**OR**

**Insert into test\_emp(empno,ename,deptno,sal,comm,dob,gender) values (104,’BARUN’,40,40000,NULL,NULL,’M’)**

****

**Make Sure while Inserting Date:**

**21-DEC-2014 - Valid**

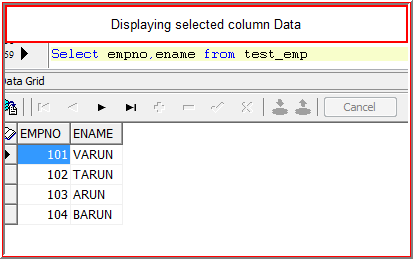
**21/DEC/2014 - Invalid**

**21-DECEMBER-2014 -Invalid**

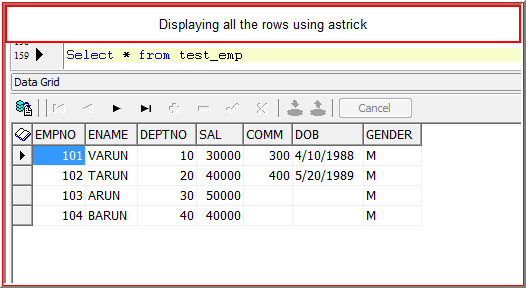
**Select**

Retrieving the data

Select empno,ename from test\_emp



Displaying the all rows using astrick(\*)



**Inserting multiple rows at a time in a single table**

insert all

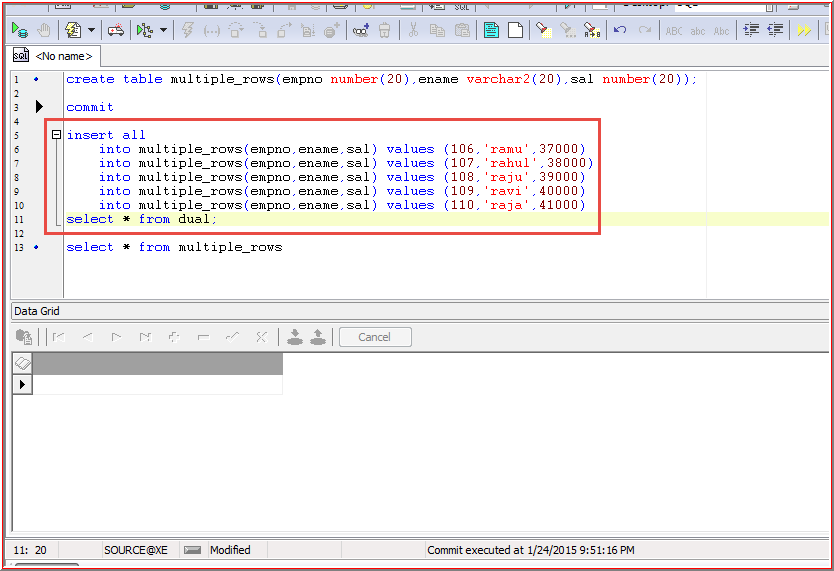
into multiple\_rows(empno,ename,sal) values (106,'ramu',37000)

into multiple\_rows(empno,ename,sal) values (107,'rahul',38000)

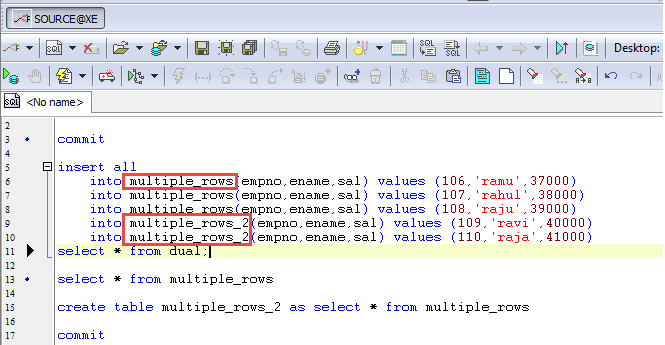
into multiple\_rows(empno,ename,sal) values (108,'raju',39000)

into multiple\_rows(empno,ename,sal) values (109,'ravi',40000)

into multiple\_rows(empno,ename,sal) values (110,'raja',41000)

select \* from dual;****

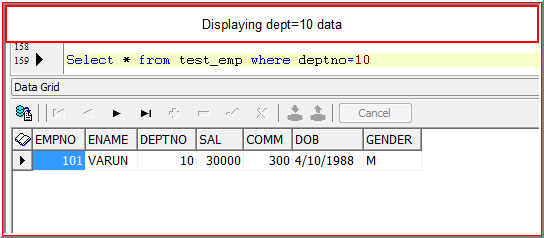
**Inserting multiple rows at a time in more than one table**

****

**Where clause:**

Used to restrict the data

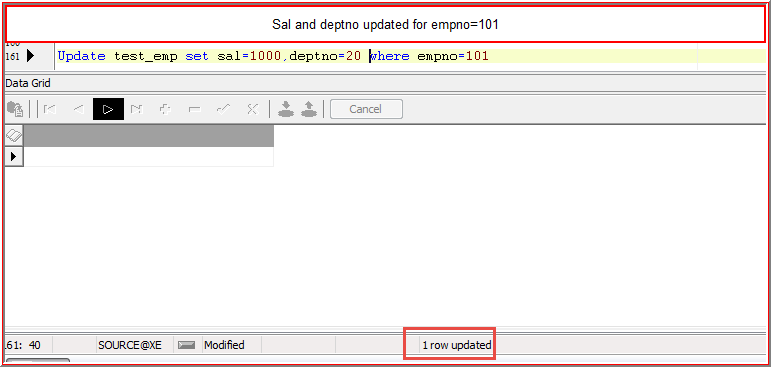
Select \* from test\_emp where deptno=10



**Update**

Using this to update the rows

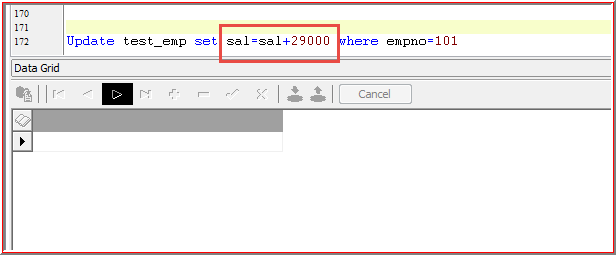
Update test\_emp set sal=1000,deptno=20 where empno=101



//Sal=30000 overrides to 1000

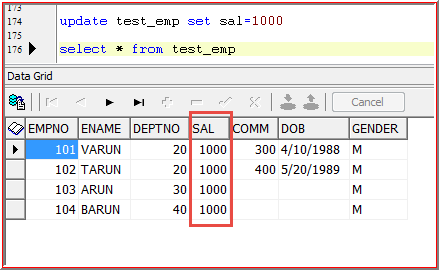
**Adding sal 29000 to existing salary empno=101**

Update test\_emp set sal=sal+29000 where empno=101



**Update all rows at a time**

update test\_emp set sal=1000



Or

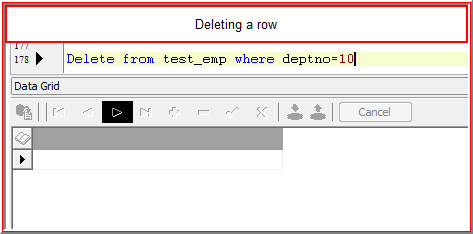
update test\_emp set sal=1000 where 1=1(All rows updated)

update test\_emp set sal=1000 where 1=2(No rows updated)

**Delete statement**

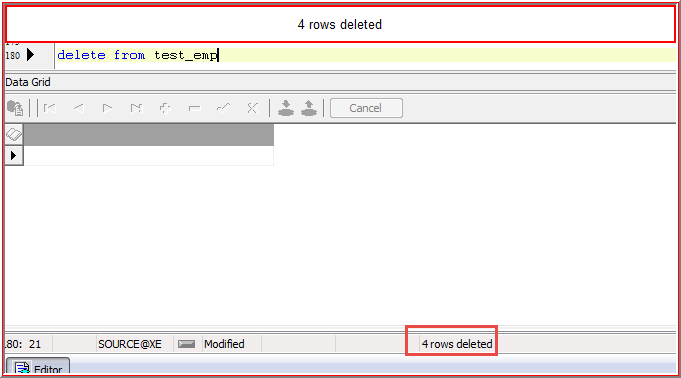
**Deleting a row**

Delete from test\_emp where deptno=10



**Deleting all rows**

Delete from test\_emp



Delete from test\_emp where 1=1(All rows deleted)

Delete from test\_emp where 1=2

**Transaction Control Language**

If we perform any DML Statements (insert, delete, update) primarily it will affect the **BUFFER** instead of **Database**

**Commit:**

All the changes from the **Buffer** will be updated into the **Database** and **Buffer** will be closed

**Roll back:**

Without making the changes to **Database, buffer** will be closed

Whenever we start the DML statement, buffer will be created

**Rollback works only for DML statements**

* When we execute the roll back statement after the **insert or delete or update**, the buffer will be closed
* Whenever we use **commit** statement after the DML statements, the **Buffer** Changes will be effected to **Database**
* **Rollback** should use before **commit** statement, otherwise there is no use
* Once we use **rollback** after the **commit**, then there is no use of **rollback**

**Savepoint**

If we use commit or rollback, it will affect all the transactions whatever we have done so far. But I want to **commit or rollback some statements;** we can achieve this using **savepoint**

**Savepoint marks the position in the buffer**

**Savepoint A;**

Update test\_emp set sal=7000 where empno=101;

Update test\_emp set sal=5000 where empno=102;

**Savepoint B;**

Delete from test\_emp where empno=103;

**Savepoint C;**

Insert into test\_emp values(104,’varun’,32000)

**Execute the below statement**

**Rollback to B;**

**From B all save points will be rollbacked (i.e savepoint B and C)**

**DATA TYPES**

**Number**

Create table test\_emp2(empno number(4,3),empid integer,empsal decimal,ename char(20),efirst varchar2(20),doj date)

insert into test\_emp2(empno,empid) values (9,1000)

Number will allow only upto 9 and after precision will allow ony 3 digits, if we enter more than 3 it will round off

Integer 🡨-- 1000 <-🡪1000

Decimal 🡨--- 559.95 🡨--🡪559.95

Sal number(7,3)

**Note:** 7 indicates that maximum inserted value and 3 represents maximum number after decimal point

586 🡪 586

98578 🡪 It won’t insert

985.7689 🡪985.769(It will roundoff the value to next if > 5)

985.7682 🡪 985.768(It will roundoff the value to that if <5)

**Char and Varchar2**

|  |  |
| --- | --- |
| EMP1 | EMP2 |
| ename char(10) | ename varchar2(10) |
| Ramu | Ramu |
| Ramana | Ramana |

Select length(ename) from emp1

10

10

Select length(ename) from emp2

4

6

**Difference between char and varchar2**

|  |  |
| --- | --- |
| Char | Varchar2 |
| Accepts alphanumeric | Accepts alphanumeric |
| MAX Size is 2000 bytes | MAX Size is 4000 bytes |
| FIXED Length | Variable Length |
| Data Size depends upon declared | Data Size depends upon data |

**Date**

Input : 22-DEC-2014

Output : 12/22/2014

Input should be **22-DEC-2014**, but not **22/DEC/2014,22/DECEMBER/2014**

Date will occupy ‘9’ bytes

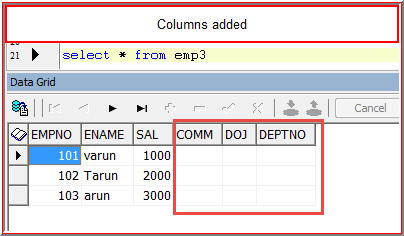
Characters (‘- -‘are used for formatting purpose only. It won’t occupy memory)

**Alter**

|  |  |  |
| --- | --- | --- |
| emp3 |  |  |
| empno | ename | sal |
| 101 | varun | 1000 |
| 102 | tarun | 2000 |
| 103 | arun | 3000 |

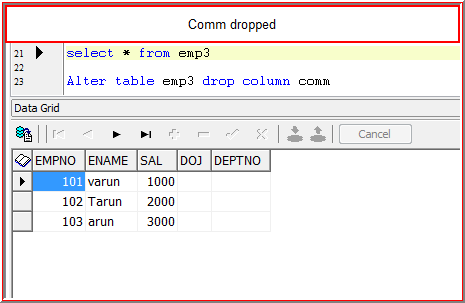
**Adding the column**

**Alter table emp3 add (comm number(20),DOJ date, deptno number(20))**

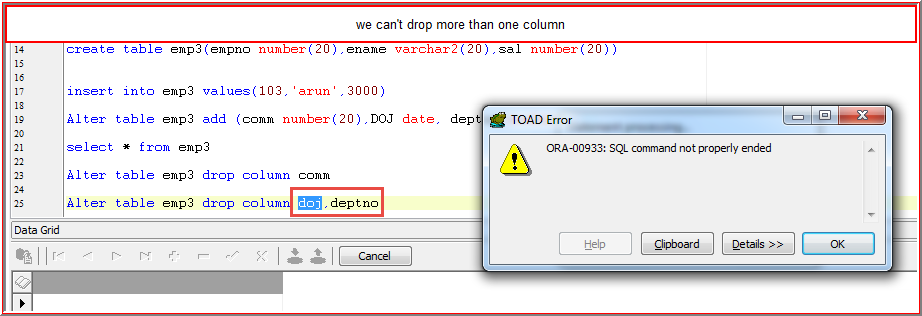


**Dropping the column:**

Alter table emp3 drop column comm



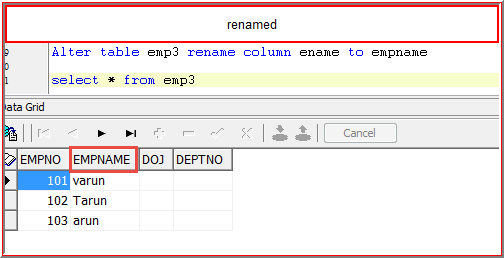
Alter table emp3 drop column doj,deptno



We can’t drop more than one column at a time

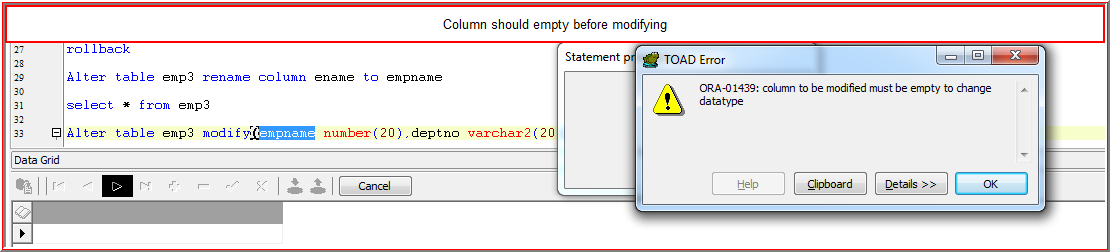
**Renaming the column name**

Alter table emp3 rename column ename to empname

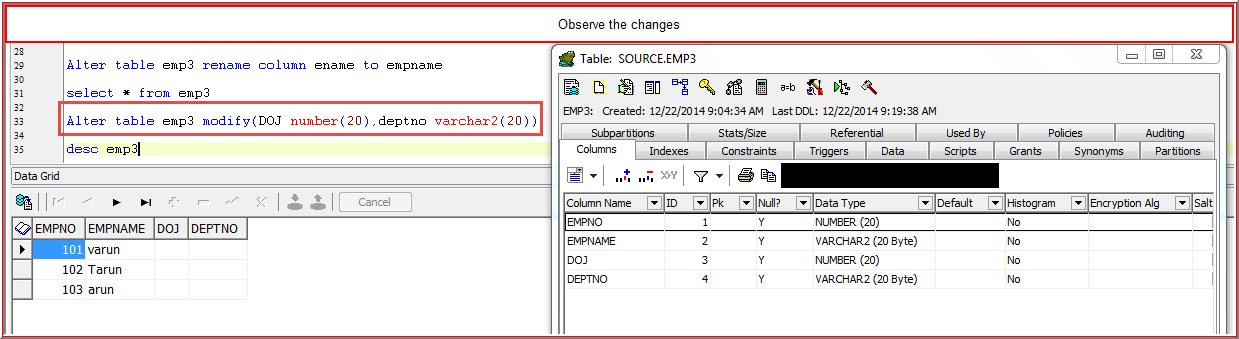


**Changing the datatype**

Alter table emp3 modify(empname number(20),deptno varchar2(20))



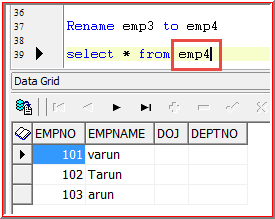
Alter table emp3 modify(DOJ number(20),deptno varchar2(20))





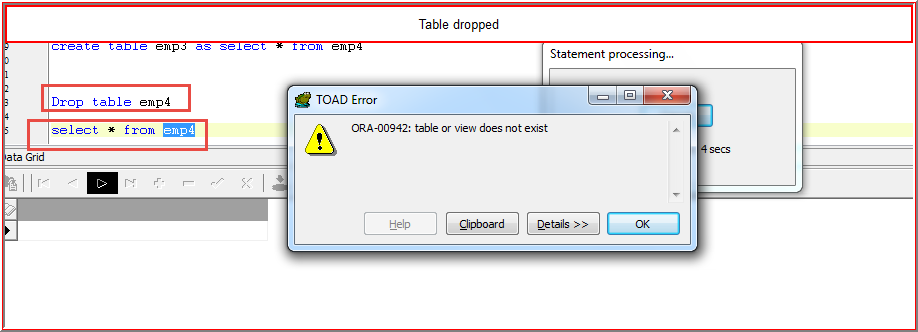
**Renaming Table Name**

Rename emp3 to emp4



**Dropping the table**

Drop table emp4



**Truncate**

Truncate table emp3

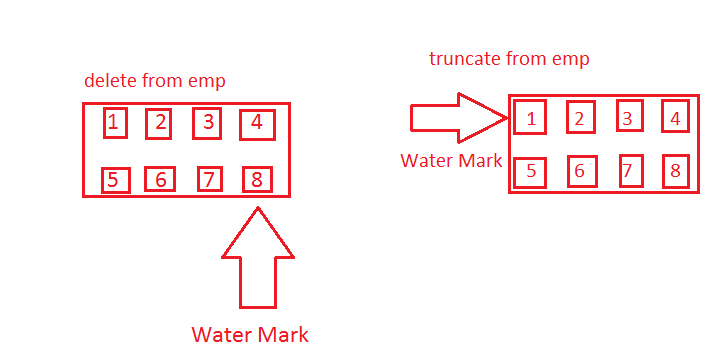
It will delete all the rows from the table

**Difference between Delete and Truncate**

|  |  |
| --- | --- |
| Delete | Truncate |
| It is a DML statement(Buffer concept is there | DDL(No Buffer concept) |
| we can use where clause to delete particular rows | We cannot use where clause, It will delete all the rows |
| Once we delete the data we can get back using rollback | Once we delete the data we can not get back it |
| It will raise the triggers | It won't raise the Triggers |

**High Water Mark**

It marks the position of last row in Database



For Delete Data will be deleted from buffer

For Truncate don’t have Buffer. So, it will be getting the data from Database.

Truncate is faster than Delete. Because delete reads both **Buffer and Database (Copy from Database)**

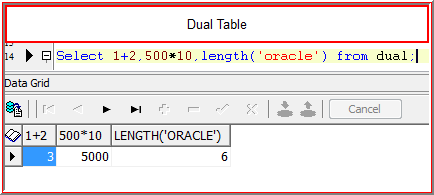
**Dual Table**

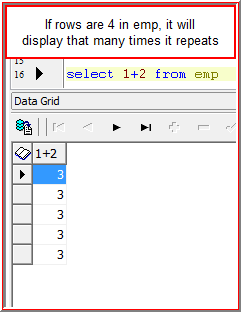
If we are writing any statement which is no way related to any table at that time. We can use the dual table

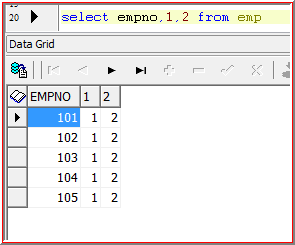
Eg: Calculating 1+2,5\*2, length of prakash etc

Dual Table contains the column **Dummy,** If the datatype varchar2 size 1 Nobody can modify the dual table

Select 1+2,500\*10,length(‘oracle’) from dual;

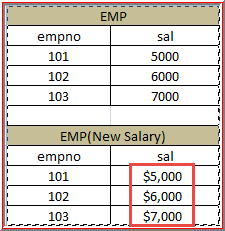






**Functions:**

It will take data from **Database** and manipulates the data as per the requirement and display the output



**Two Types of Functions**

1. **Single Row Function:**

It operates on individual rows and result will be per row

Eg: select empno,length(ename) from emp;

In order to calculate length of ename, no need to consider other rows

2. **Multiple row functions or Group functions or Aggregate functions:**

Operates on **group** of the rows and result will be per group

E.g.: select max(sal) from emp;

In order to calculate max(sal). We need to consider all the rows.

**1.Character Functions**

**Case Manipulations functions**

|  |  |
| --- | --- |
| Upper | It converts at any case to uppercase |
| Lower | It converts at any case to lowercase |
| Initcap | Starting letter Cap and remaining all small letters |

**Character Manipulation functions**

|  |  |  |
| --- | --- | --- |
| Concat | Rpad | Translate |
| length | Lpad | Replace |
| substr | Rtrim | Decode |
| instr | Ltrim | Case |
|  | Trim |  |

**2. Number Functions**

|  |  |
| --- | --- |
| Abs | ceil |
| Sign | floor |
| Mod | round |
|  | trunc |

**3. Conversion functions**

**To\_date: -** Varchar2 to Date

**To\_char: -** Date to varchar2

**To\_number:-** varchar2 to number

4. **Date Functions:**

|  |
| --- |
| add\_months |
| months\_between |
| next\_day |
| last\_day |
| round |
| trunc |

**5. General Functions:**

|  |
| --- |
| NVL |
| NULL |
| NULLif |
| coalesce |

**Multi Row Functions**

|  |
| --- |
| MAX |
| MIN |
| SUM |
| COUNT |
| AVG |

|  |  |  |
| --- | --- | --- |
| empno | ename | sal |
| 101 | Ramu | 1000 |
| 102 | Ramana | 2000 |
| 103 | Ramu | 3000 |

2. Extract the employ details whose name is Ramu

**Select \* from where ename=’Ramu’;**

|  |  |  |
| --- | --- | --- |
| 101 | Ramu | 1000 |

**Select empno,upper(ename),sal from emp**

|  |  |  |
| --- | --- | --- |
| empno | ename | sal |
| 101 | RAMU | 1000 |
| 102 | RAMANA | 2000 |
| 103 | RAMU | 3000 |

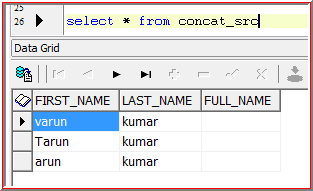
**Select empno,lower(ename),sal from emp**

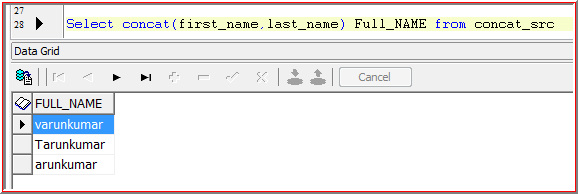
|  |  |  |
| --- | --- | --- |
| empno | ename | sal |
| 101 | ramu | 1000 |
| 102 | ramana | 2000 |
| 103 | ramu | 3000 |

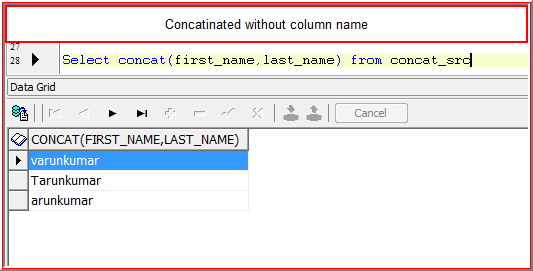
**Select empno,initcap(ename),sal from emp**

|  |  |  |
| --- | --- | --- |
| empno | ename | sal |
| 101 | Ramu | 1000 |
| 102 | Ramana | 2000 |
| 103 | Ramu | 3000 |

**2. Concat():**

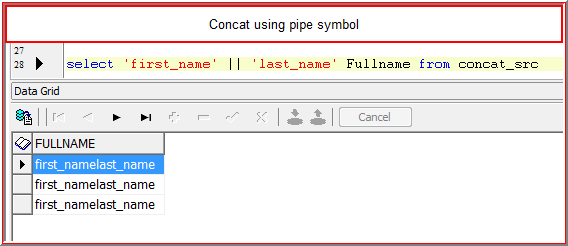


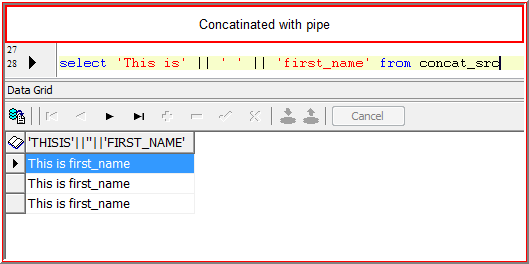
Select concat(first\_name,last\_name) Full\_NAME from concat\_src 

Select concat(first\_name,last\_name) from concat\_src

**Concat using pipe symbol(||)**

Select ‘first\_name’ || ‘last\_name’ from concat\_src



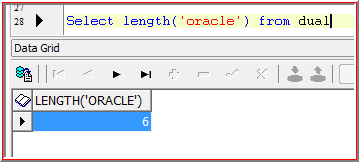
select 'This is' || ' ' || 'first\_name' from concat\_src

2. **Length ()**

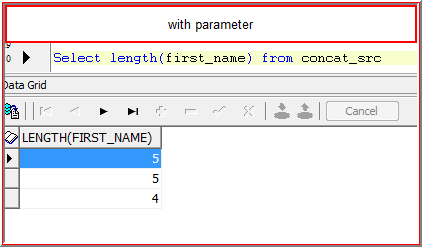
It will give the size of the string

Finding length without parameter

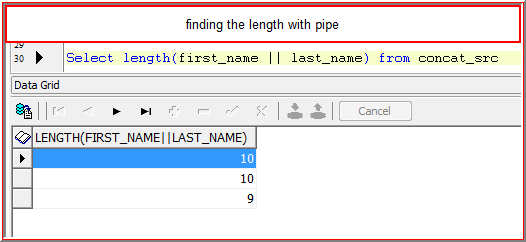
Select length(‘oracle’) from dual



Select length(first\_name) from concat\_src

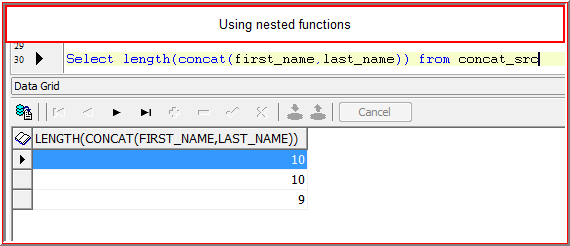


**With pipe**

Select length(first\_name || last\_name) from concat\_src 

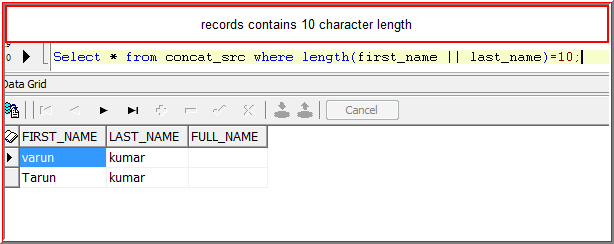
**Nested functions**

Select length(concat(first\_name,last\_name)) from concat\_src



**Extract the employee details whose full name contains 10 characters**

Select \* from concat\_src where length(first\_name || last\_name)=10;



**2. Substr ():**

It extracts number of characters from particular position

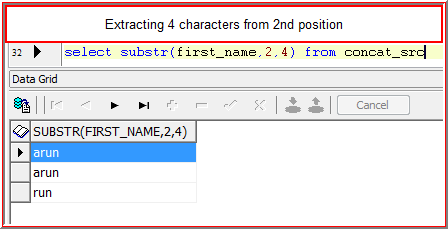
Substr(‘oracleimages’,2,4)=racl

1st parameter = string name, using either parameter or direct string with in Quotes

2nd parameter = Starting position of extract

3rd parameter = ending position of extract

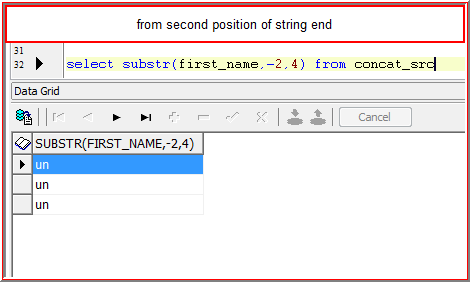
**Extract 4 characters from 2nd position**

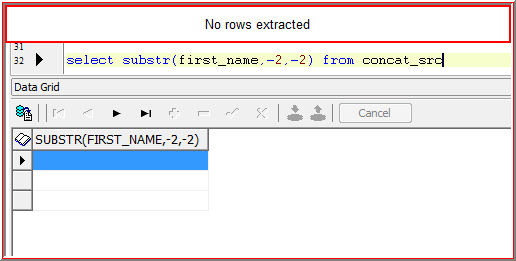
select substr(first\_name,2,4) from concat\_src

select substr(first\_name,-2,4) from concat\_src

**varun**

from u to next letters

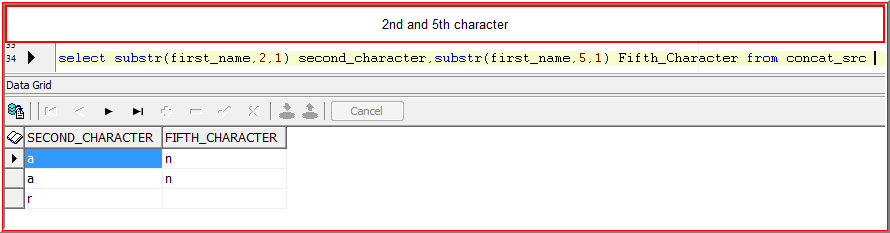


select substr(first\_name,-2,-2) from concat\_src 

**\*Extraction should be always from left to right**

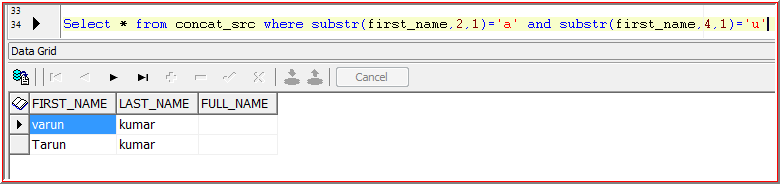
**Find out 2th and 5th character from first\_name**

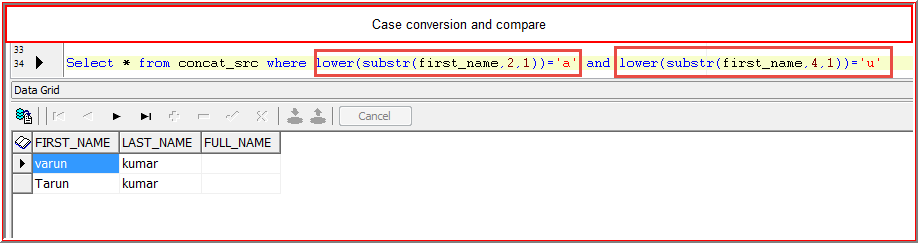
select substr(first\_name,2,1) second\_character,substr(first\_name,5,1) Fifth\_Character from concat\_src

****

**Extract the details whose first\_name contain 2nd character as ‘a’ and 4th character as ‘u’**

**Select \* from concat\_src where** substr(first\_name,2,1)=’a’ and substr(first\_name,4,1)=’u’

****

**Select \* from concat\_src where (**substr(first\_name,2,1)=’a’ and substr(first\_name,4,1)=’u’ 

**2. Instr() :**

It will give the position of the character for particular occurrence

Instr(‘oracle images’,’a’,1,2) =>9

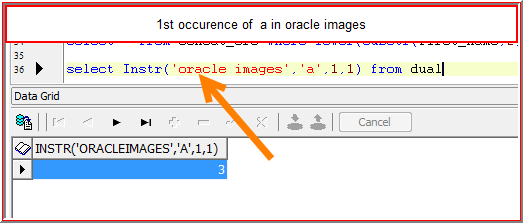
1st parameter = string name, using either parameter or direct string with in Quotes

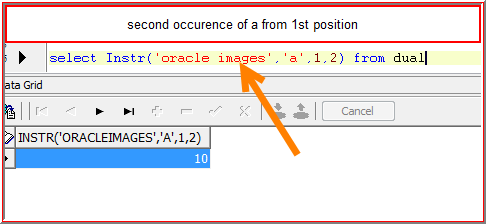
2nd parameter = Character

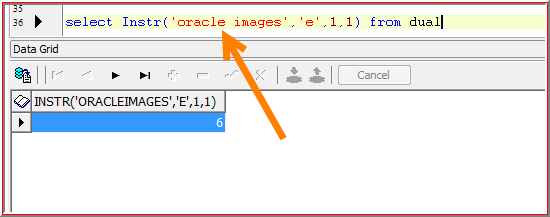
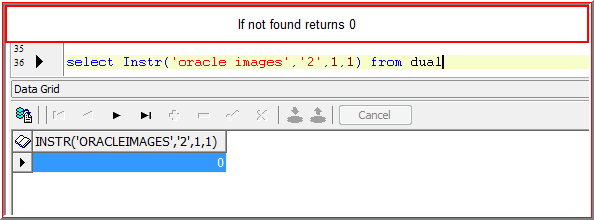
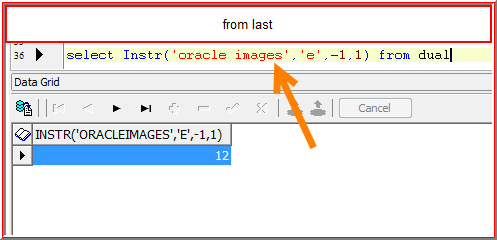
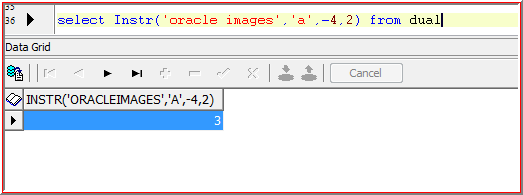
3rd parameter = staring position of string

4th parameter=Number occurrence

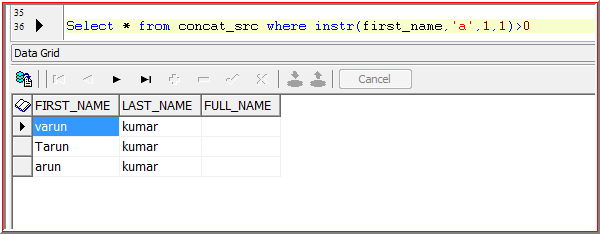
At which position ‘a’ has occurred 2nd time from 1st position





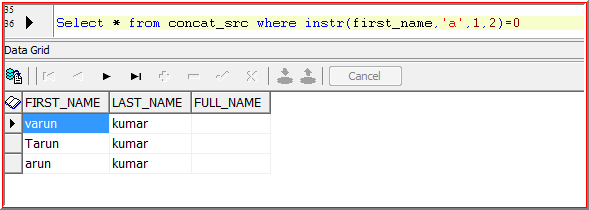
   

**Extract the details whose first\_name contains minimum one ‘a’**

Select \* from concat\_src where instr(first\_name,'a',1,1)>0

**Extract the details whose first\_name contains 1 ‘a’ only**

**Select \* from concat\_src where instr(first\_name,’a’,1,2)=0**

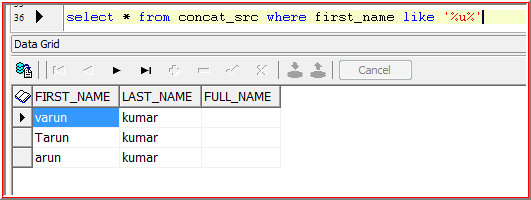
****

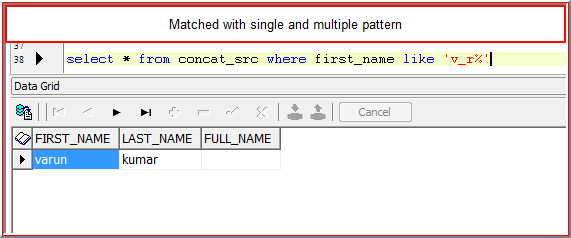
**Like**

It will search the records based on the give patter using the symbols percentile (%), underscore(\_)

% = Multiple characters

\_ = Single character

select \* from concat\_src where first\_name like '%u%' 

select \* from concat\_src where first\_name like 'v\_r%' 

**Extract the details whose first\_name minimum 3 a’s**

Select \* from emp where instr(ename,’a’,1,3)>0

Select \* from emp where ename like ‘%a%a%a’

**Extract the emp details whose name contains only 2 ‘a’ s**

Select \* from emp where instr(ename,’a’,1,2)>0 and instr(ename,’a’,1,3)=0

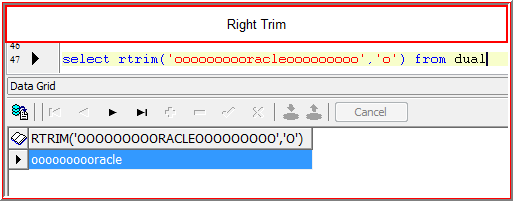
**Extract the emp details whose name contains only 4 ‘a’ s**

Select \* from emp where instr(ename,’a’,1,4)>0 and instr(ename,’a’,1,5)=0

**We cannot achieve this using like operator**

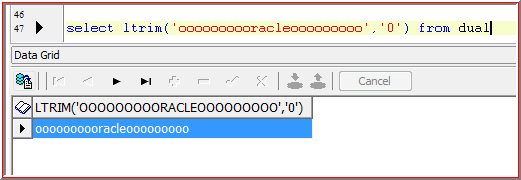
**RTRIM():** Trims the characters from right to left

select rtrim('oooooooooracleooooooooo','0') from dual

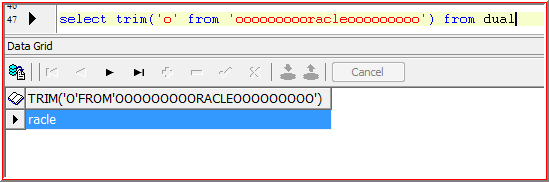


**LTRIM():** Trims the characters from left to right

select ltrim('oooooooooracleooooooooo','0') from dual

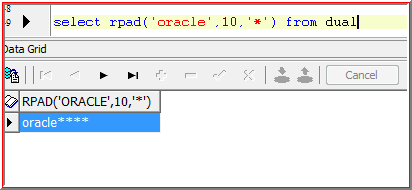
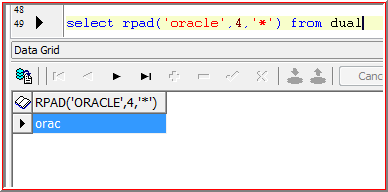


**TRIM():** Trims the characters from left to right and right to left

select trim('o' from 'oooooooooracleooooooooo') from dual 

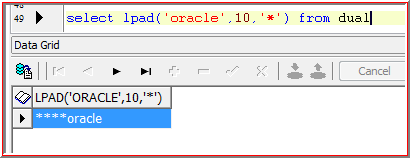
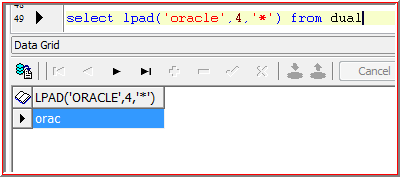
**RPAD():**

Add the specified string on right side

**LPAD():**

Add the specified string on left side

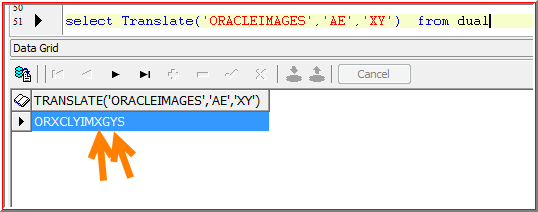
 

**2.Translate() :**

It replaces one set of sequence of characters with another set of sequence characters

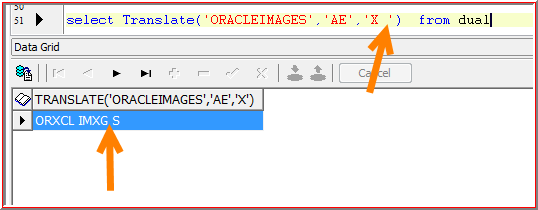
Translate(‘ORACLEIMAGES’,’AE’,’XY’)

select Translate('ORACLEIMAGES','AE','XY') from dual



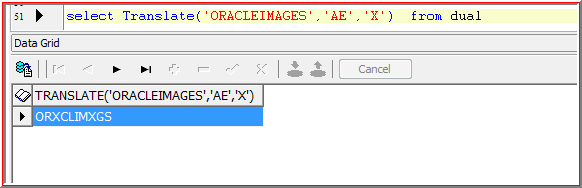
A will be replaced with X

E will be replaced with Y



A will be replaced with X

E will be replaced with Space

select Translate('ORACLEIMAGES','AE','X') from dual 

A will be replaced with X

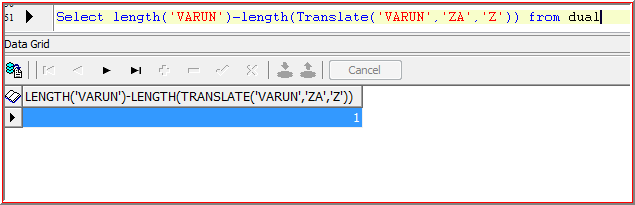
There is no replacement for ‘E’ so it will be deleted

**Find out how many ‘a’’s in your name**

Select length(‘VARUN’)-length(Translate(‘VARUN’,’ZA’,’Z’))

5-4=1

Select length('VARUN')-length(Translate('VARUN','ZA','Z')) from dual

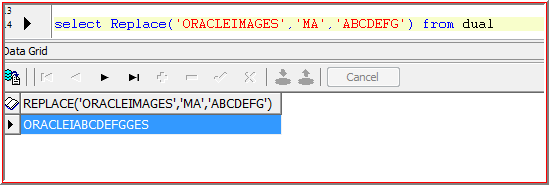


**Replace()**

It will replace one string with another

Eg:

Replace(‘ORACLEIMAGES’,’MA’,’ABCDEFG’)

select Replace('ORACLEIMAGES','MA','ABCDEFG') from dual 

**Decode:**

It uses the If then else logic

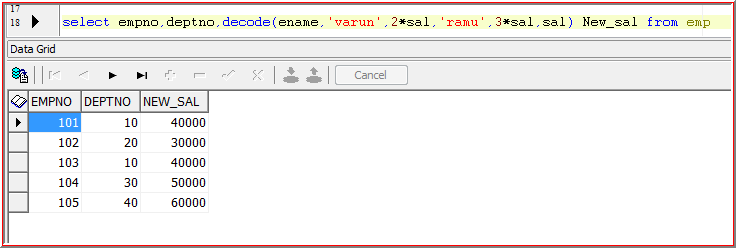
If job=’Manager’ display sal as 2\*sal

If job=’Salesman’ display sal as 3\*sal

Else

Display original salary

select empno,deptno,decode(ename,'varun',2\*sal,'ramu',3\*sal,sal) New\_sal from emp

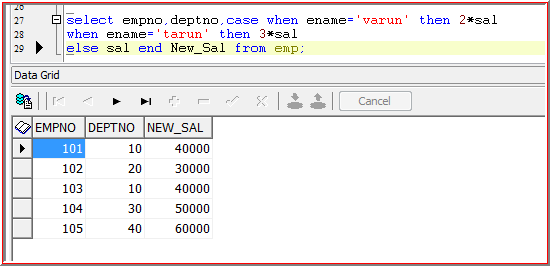


**Case**

select empno,deptno,case when ename='varun' then 2\*sal

when ename='tarun' then 3\*sal

else sal end New\_Sal from emp;



* If we are performing multiple operations on the same row then first operation will be consider and remaining operations will be ignored
* We can’t achieve above requirement using decode as in decode we can perform operations on single column only

If sal between 0-1999 then display as low\_sal

If sal between 2000-4999 then display as avg\_sal

If sal>5000 display as high\_sal

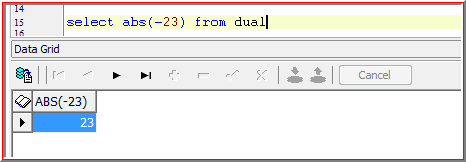
* We can’t achieve the above requirement using decode, we can’t use relational operators(In Decode we can use only column values, we can’t use any other conditions

**Disadvantages of decode:**

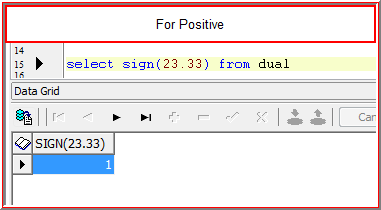
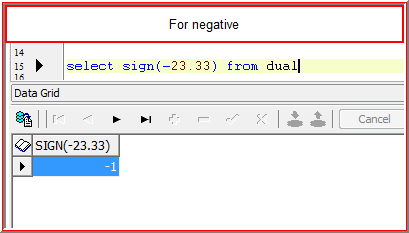
* We can’t perform operations on multiple columns
* We can’t use any conditions, performance wise decode is the best as first itself, we are mentioning on which column we are going to do the operations. But in case for every condition column can be changed. So, it will be take some time to check the column

**Number functions**

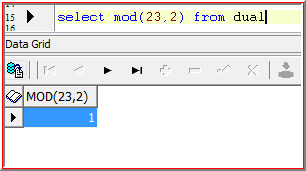
**Abs()** It returns absolute value(It ignore the sign)



**Sign() :** It returns the sign of the number(It ignores the value)

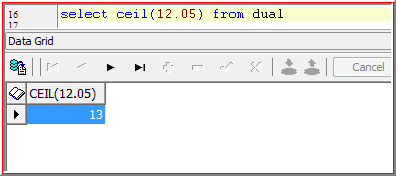
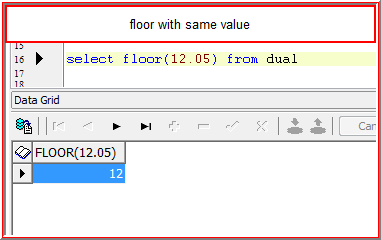
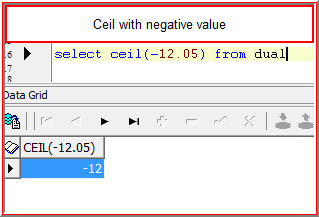
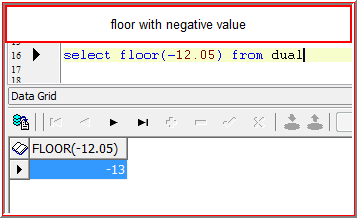
****

**Mod()** : It returns the remainder



**Ceil()** : It returns the highest integer value

**floor()** : It returns the lowest integer value

**Round() :** It rounds the number to ‘n’ decimal forms, If ‘n+1>=5 it will add ‘1’ to nth position number(If ‘n’ is +ve)

If nth position is >=5 then it add 1 to nth position(If ‘n’ is negative)

Round(12586.58**4**32,2)

12586.58

‘4’ <=5. So no need to add +1 to 2nd position(8)

Round(12586.58**7**32,2)

12586.59

‘7’ >5. So we need to add +1 to 2nd position(8)

Round(12**5**86.58**7**32,-3)

13000

Round(12**3**86.58**7**32,-3)

12000

**Trunc()**

It truncates the number to ‘n’ decimal points

Trunc(12586.58454,2)

12586.58

**Conversion Functions:**

These are used to convert one **datatype to another datatype**

**1.TO\_DATE (): Converts varchar2 to Date**

|  |  |  |
| --- | --- | --- |
| Emp | | |
|  | | |
| empno | sal | DOJ(Varchar2) |
| 101 | 1000 | 15-Jan-10 |
| 102 | 2000 | 17-Apr-11 |

If we want to add ‘2’ month’s**.** It is not possible because it declares as varchar2. So, we need to convert it into Date, then only we can add.

To\_date will achieve the above requirement

**To\_date(’15-Jan-2010’,’DD-MM-YY’) => 01/15/10**

or

**To\_date(’15-Jan-2016’,’DD-MM-YYYY’) => 01/15/2016**

**2. To\_Char (): Converts Date to Varchar2**

select to\_char('1/10/2010') from dual

**SYSDATE: 21-Dec-2014**

select To\_char(sysdate,'cc') from dual => 21

Note: cc means current centuary

select To\_char(sysdate,'yyyy') from dual => 2014

select To\_char(sysdate,'year') from dual => twenty fourteen

select To\_char(sysdate,'mm') from dual => 12

select To\_char(sysdate,'mon') from dual => dec

select To\_char(sysdate,'w') from dual => 3(Current week of this month)

select To\_char(sysdate,'ww') from dual => 51(Current week of this year)

select To\_char(sysdate,'d') from dual => 1(Day of the week, 21 is Sunday)

select To\_char(sysdate,'dd') from dual => 21(Day of current month)

select To\_char(sysdate,'dy') from dual => sun(Day in alphabets)

select To\_char(sysdate,'day') from dual => Sunday

select To\_char(sysdate,'HH:MI:SS') from dual=> 09:32:30

select To\_char(sysdate,'HH:MI:SS am') from dual => 09:33:06 am

select To\_char(sysdate,'HH:MI:SS pm') from dual => 09:33:40 am

select To\_char(sysdate,'HH24:MI:SS') from dual => 09:34:23(Time in 24 hours format)

select To\_char(sysdate,'wsp') from dual => three

with Spellings

select To\_char(sysdate,'wsp') from dual => three(Week of this month in spelling)

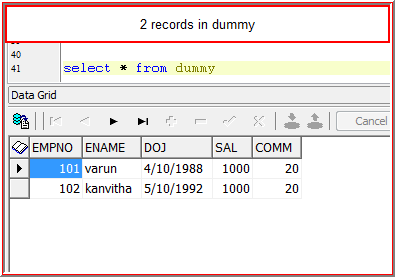
Note: sp means spelling

select To\_char(sysdate,'wth') from dual => 3rd

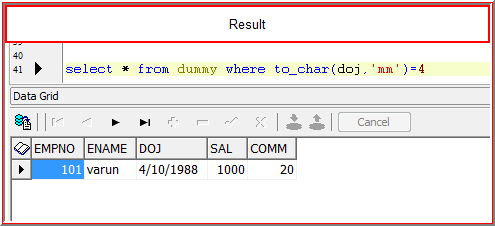
Note: week with extension

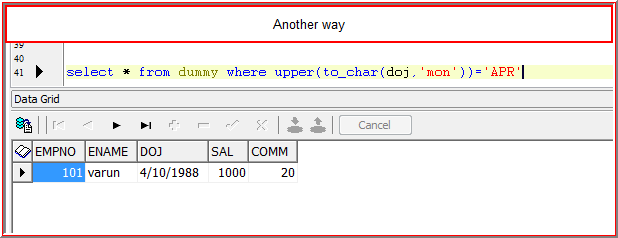
select To\_char(sysdate,'wspth') from dual => third

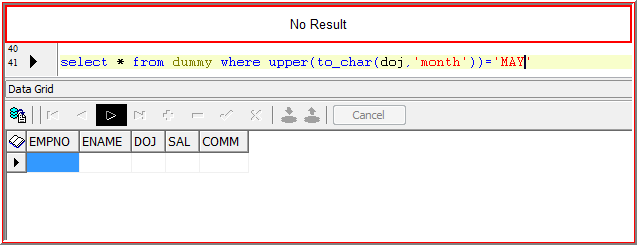
**Extract the emp details who has joined in the month of march**



select \* from dummy where to\_char(doj,'mm')=4

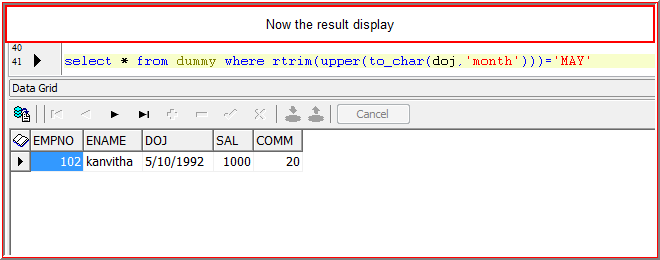
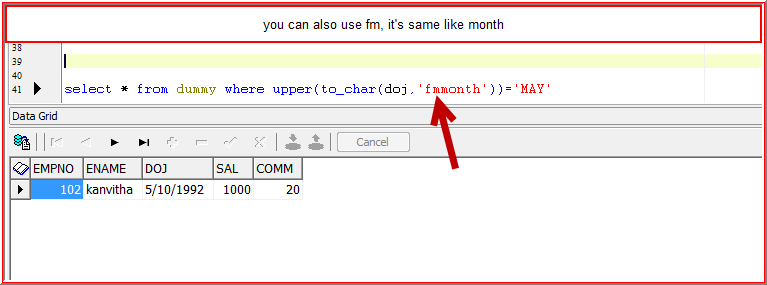




Month returns 9 character lengths and remaining spaces, that’s why it doesn’t match with **May**

It will take maximum month length SEPTEMBER=9

If 9 characters are not there, it will fill with spaces. so, we need to trim the characters

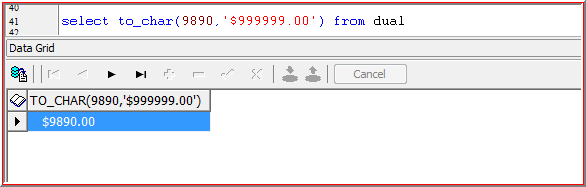
**Fm means fill mode**

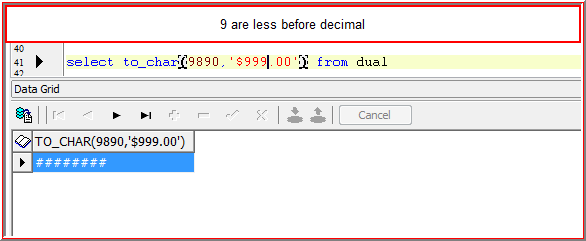
Embedded spaces can be removed by placing the 'fm' prefix .

It’s same like Days Suday,Monday,Tuesday

It will take maximum day length wednesday=9

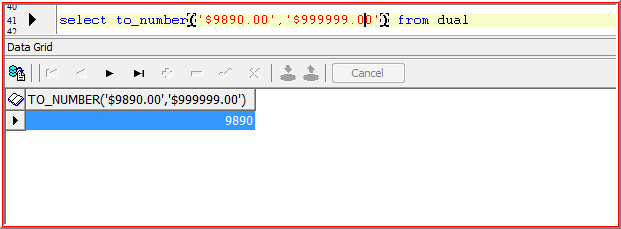
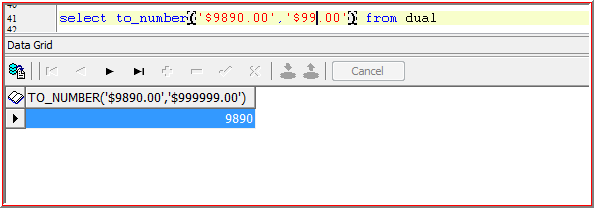
**To\_char : It converts Number to character**

****

****

**There is no enough 9’s increase the number of 9’s**

**To\_number : It converts character to Number**

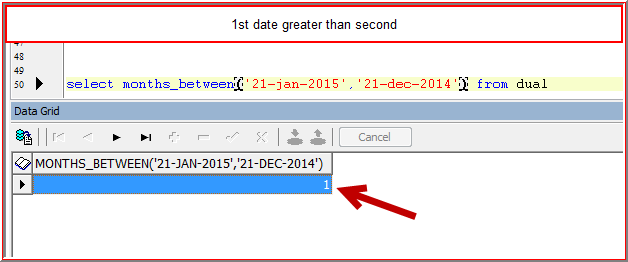
 

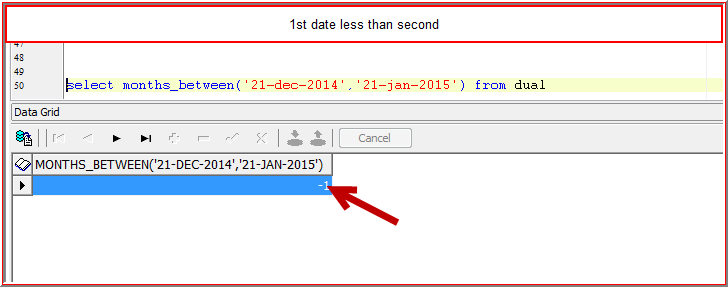
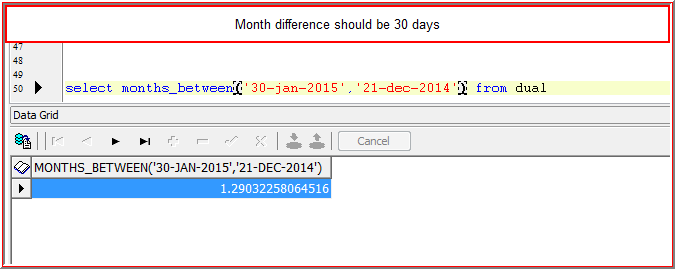
**There is no enough 9’s increase the number of 9’s**

**Date Functions**

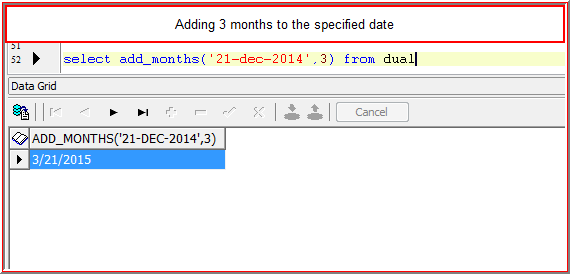
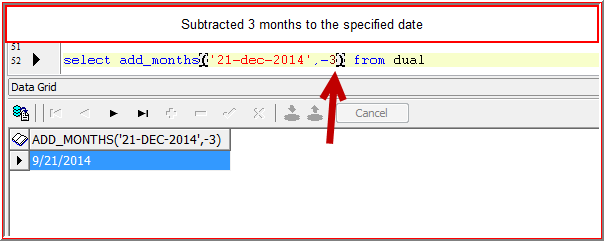
Months\_Between (): It will give number of months between specified Dates

If 1st date is greater than second date then the result will be **positive else negative**

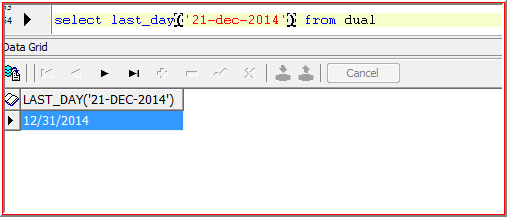


**Add\_months () : It will add number months to the specified date**

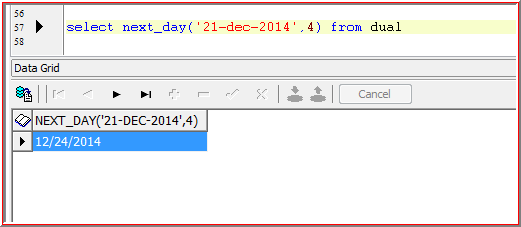
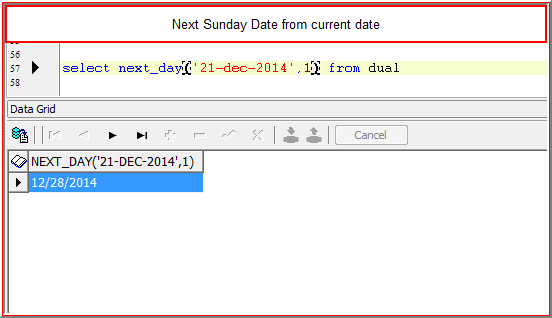
**Last\_date() : It will give the last day date of the month**



**Next\_day()** : It will gives the date of the specified day which occurs next to specified date

|  |  |
| --- | --- |
| Sunday | 1 |
| Monday | 2 |
| Tuesday | 3 |
| Wednesday | 4 |
| Thursday | 5 |
| Friday | 6 |
| Saturday | 7 |

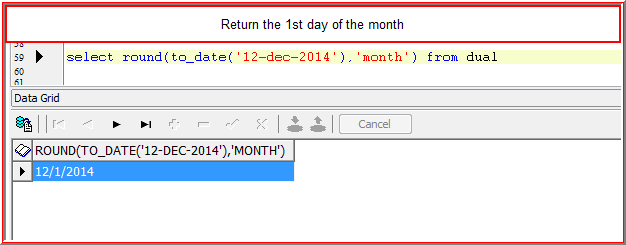
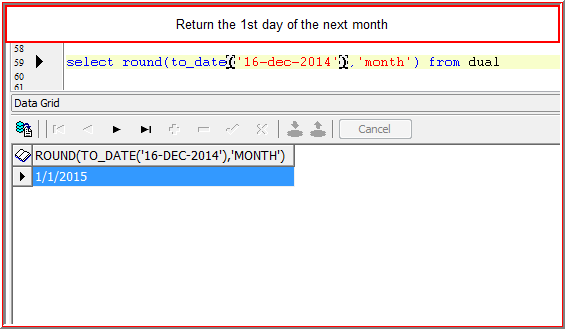
Next Wednesday date from current date

In the above date functions we have given strings instead of the date

But above functions have the capacity to convert the string to date

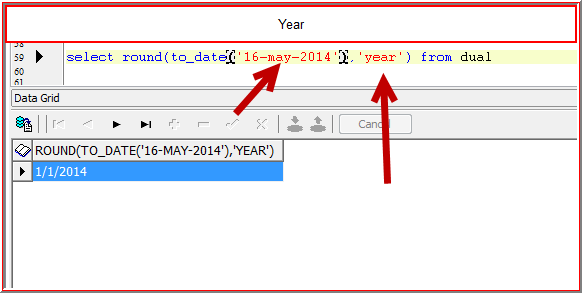
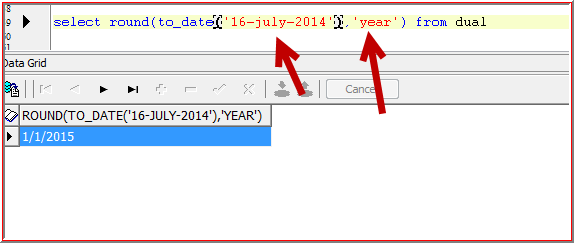
**Round() : It rounds the day to nearest day**

** **

If second argument is **Month** then consider the specified **Date Day**

**If Day>15** then return **next month starting date** else return **specified month starting date**

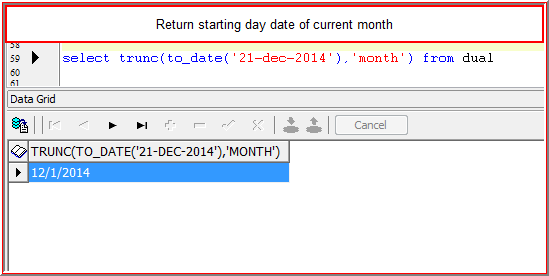
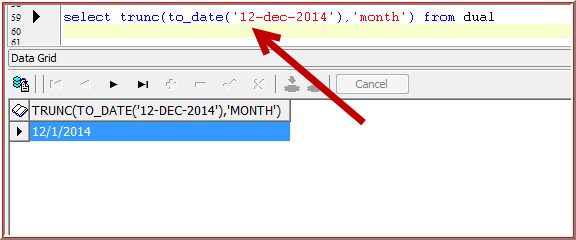
**If second argument is year**

** **

If second argument is **Year** then consider the specified **Date Month**

**If Month>6** then return **next year starting date** else return **specified year starting date**

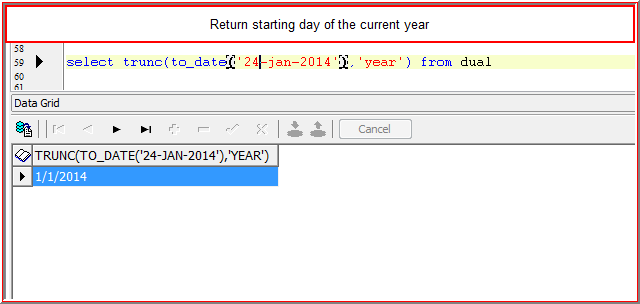
**Trunc () : It truncates the day to nearest day**

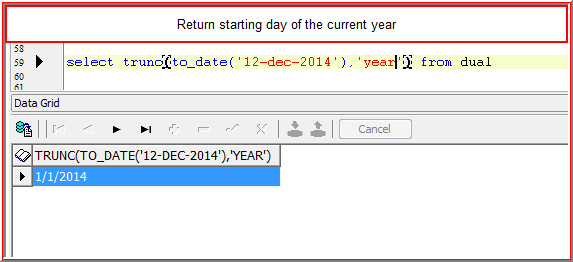
** **

If second argument is **Month** then consider the specified **Date Day**

It returns **specified month starting date**

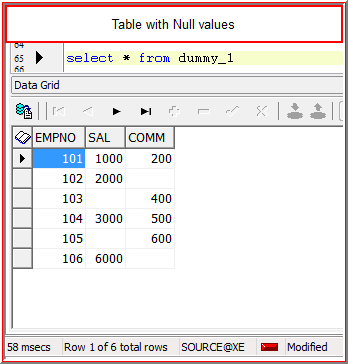
**If second argument is year**

****



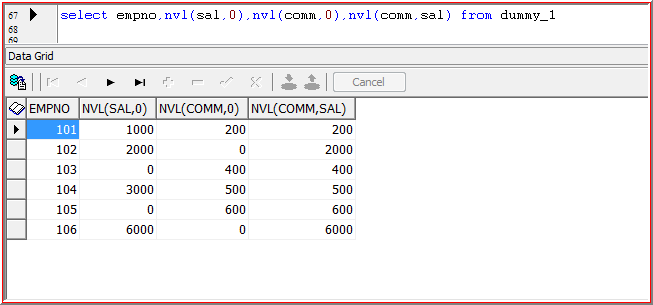
If second argument is year then it returns the current year starting date

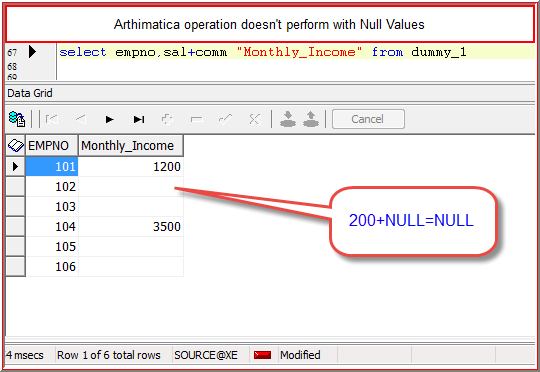
**General Functions:**



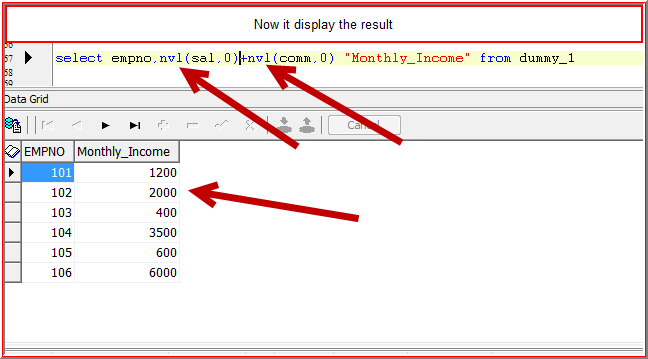
**1.NVL ():**

It converts NULL values to actual value (Actual value should be ‘0’) or other column value





**To overcome the above problem we use NVL()**



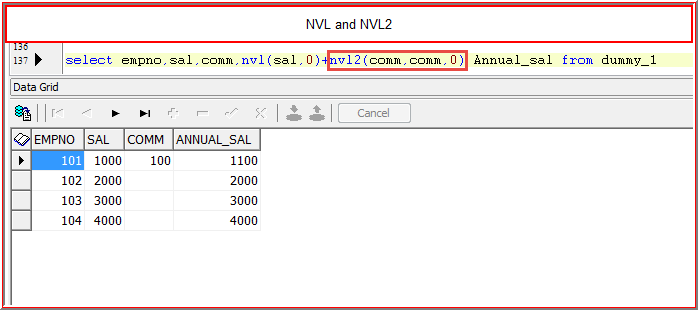
If comm. Is null then it will take 0

**2.NVL2() :**

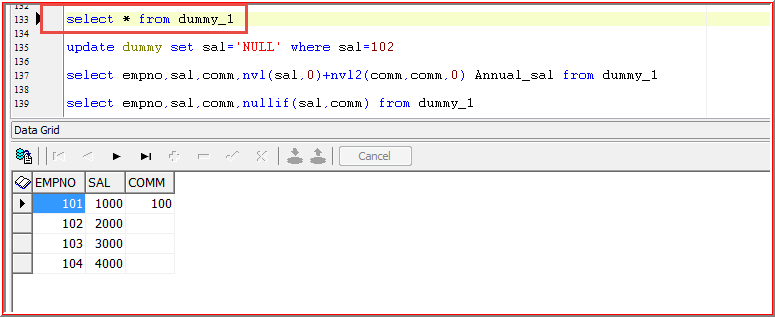
It is a function used to convert null values into actual values

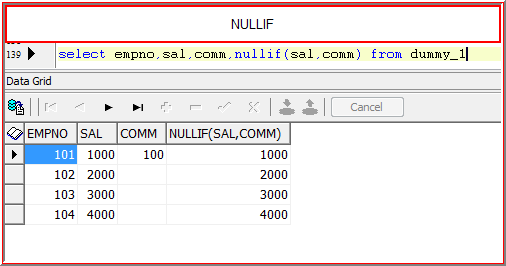
->It will take three expressions or parameters, If expression one is **NULL** then it will take **exp3** else it will take **exp2**

**eg**



**NULLIF () :**





It will take two expressions if exp1 = exp2, then it will return ‘NULL’ value, else it will return ‘exp1’

**Null (exp1) =Null (exp2 ) => NULL**

**1000(exp1) =Null(exp2) => NULL**

**NULL (exp1)=1000(exp2) => NULL**

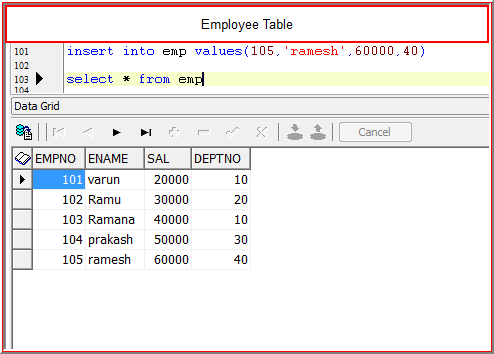
**Null is not equal to any other Null Value. So, it returns null value**

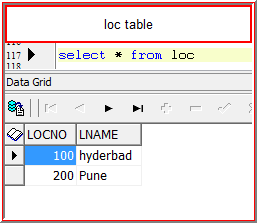
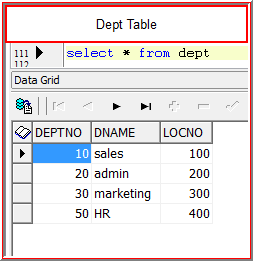
**4. Coalesce(exp1,exp2,exp3,………….expn)**

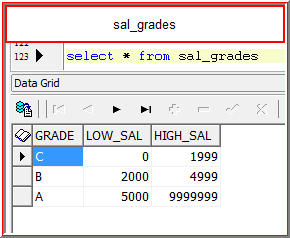
It will take ‘n’ number of expressions and will return first “not Null” Expression

Eg: coalesce(sal,com,pf,td,0)

**JOINS:**







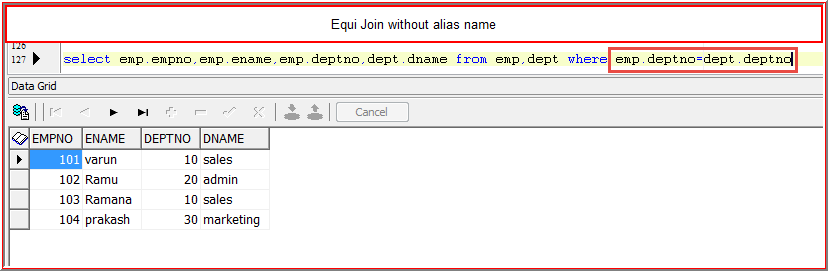
**Extract the empno, ename, deptno and dname details**

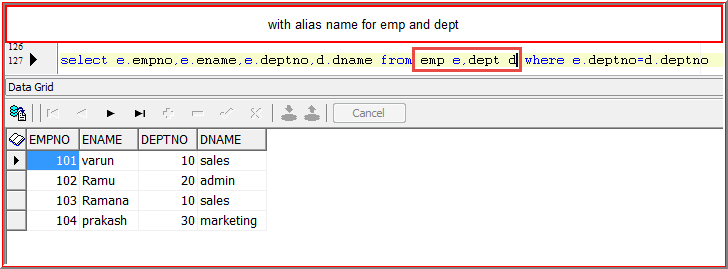
Empno, ename, deptno existed in EMP table dname existed DEPT table

So, needs to use the joins to get the desired information

**Equi-Join :**

A Join condition uses equivalent operator(=) is called Equi-join



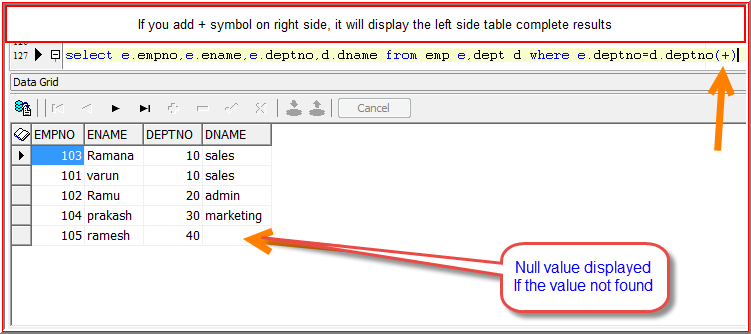


In the above result empno=105 not displayed, because 105 respective deptno=40 is not matched with dept table

Still I want 105 details, to achieve this we use outer join

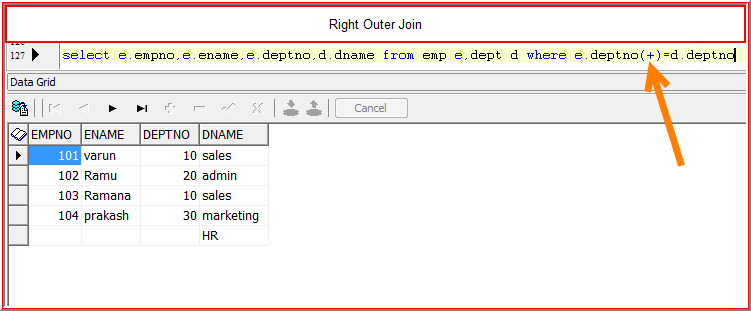
**Outer Join** = Equijoin result + Extra Rows

**Left outer Join**

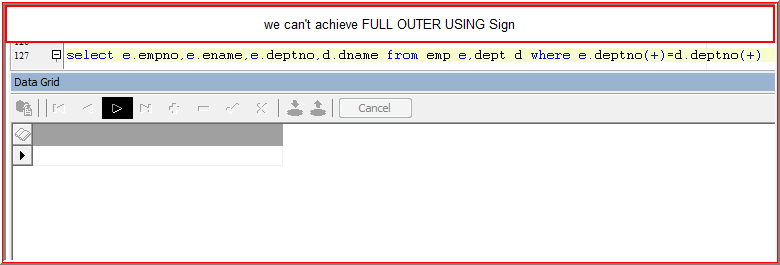


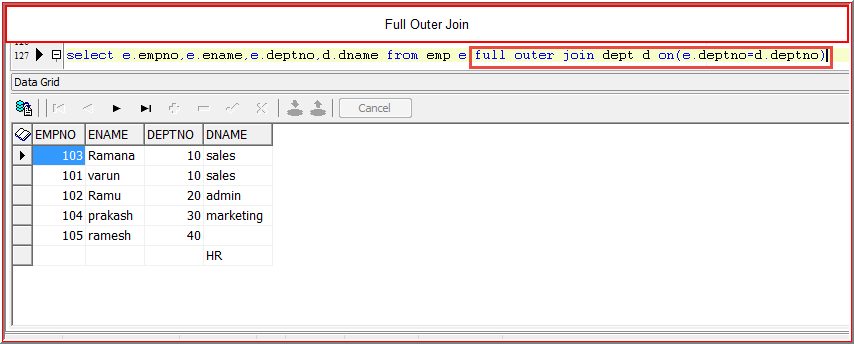
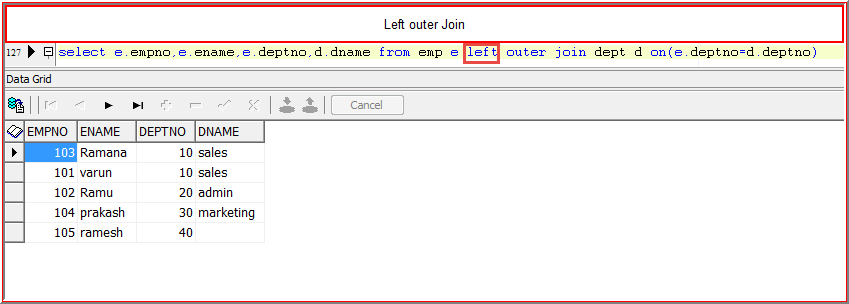
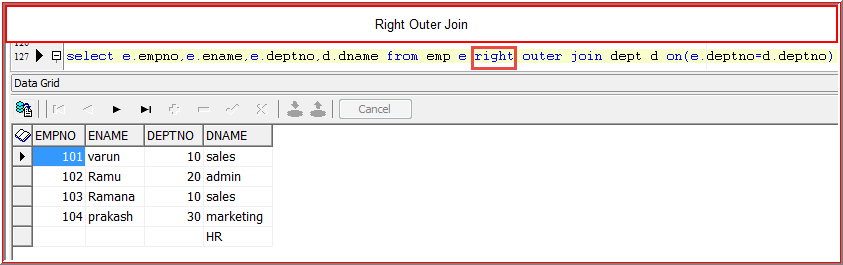
Now 105 displayed

**Right Outer Join**

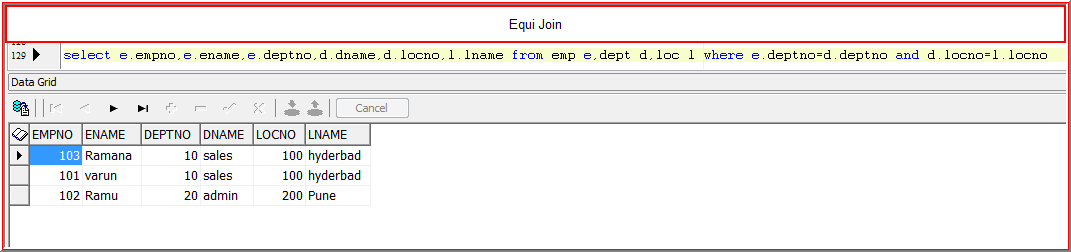
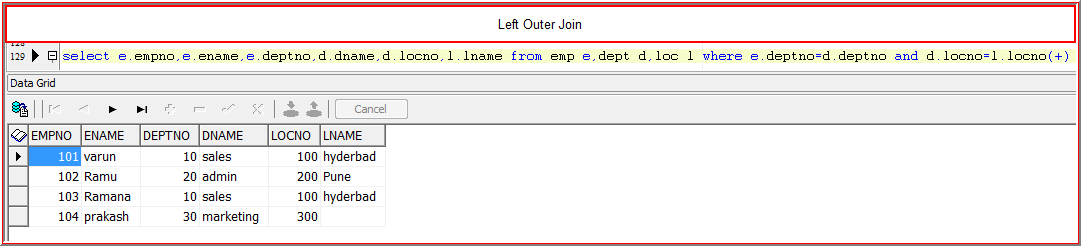
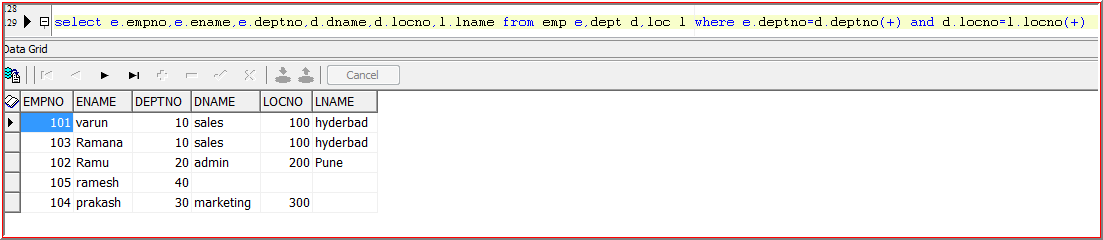
****

**Full Outer Join**

****

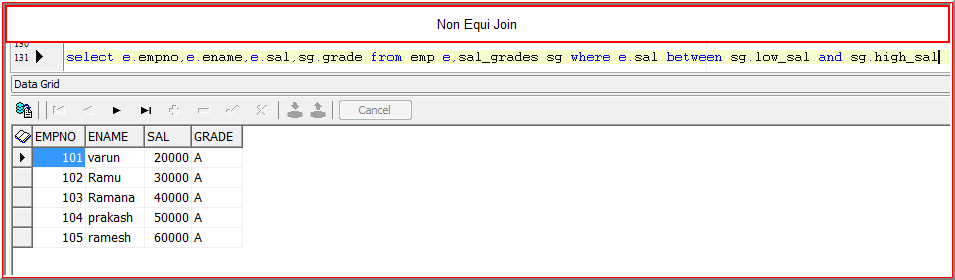
**  **

**Extract empno, ename, deptno, dname, loc, locname**

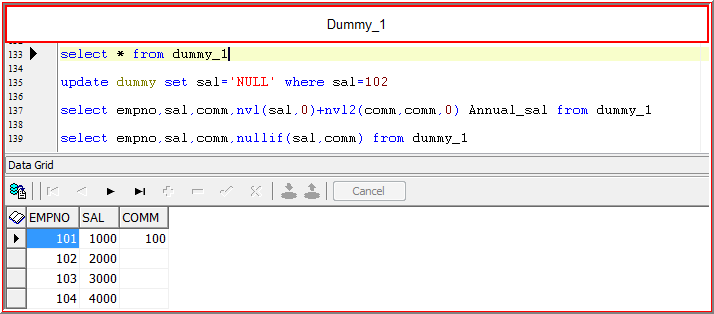
**  **

**Non Equi Join**

A join Condition which uses other than equivalent Operator

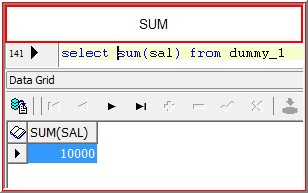


**Multirow functions or Group Functions**

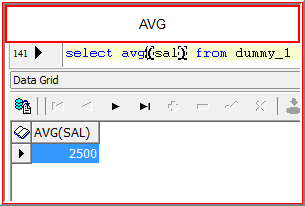
****

**Sum()**

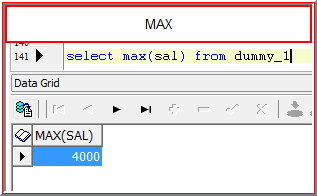
It will returns summation value



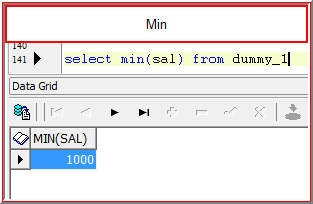
**Avg():** It will returns avg value

****

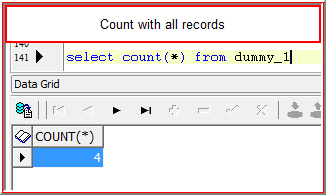
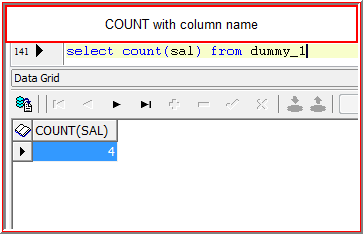
**Max() : It will return max value**

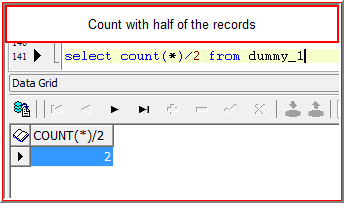
****

**Min() : It will return Min value**

****

**Count() : It will return number of records**

****

****

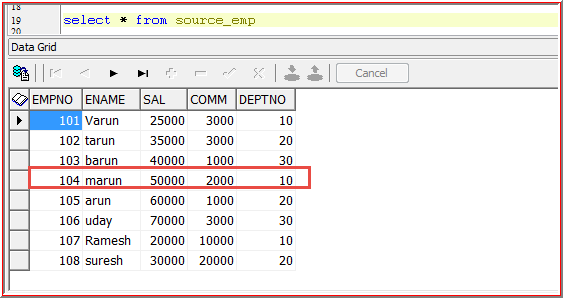
**SUB Queries**

**It is a select statement which is embedded in another ‘Clause of statement’.**

**or**

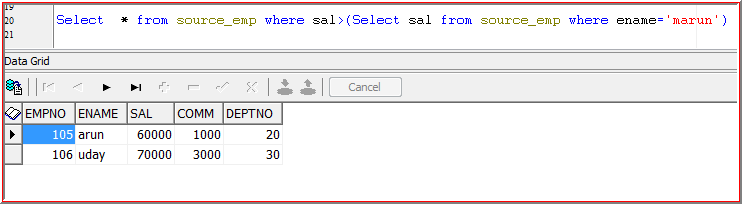
**Query within another Query is called sub query**

**We can write sub queries in “select” clause “from” clause and “where” clause**

****

**Extract all the employee details who are earning the salary more than the employee name “marun” Salary**

Select \* from source\_emp where sal>(Select sal from source\_emp where ename=’marun’)

****

**Types of sub queries:**

1. **Single row sub query :**  In these queries first the inner query will be executed and return single record as output, then this output will be passed to outer query as single input. So, that the outer query will be executed and return the final output

**Operators used for single row sub query**

**Equal ‘=’**

**Not Equal ‘!=’**

**Greater than ‘>’**

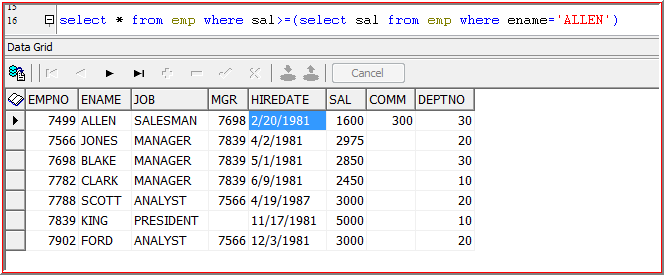
**Greater than or equal to ‘>=’**

**Less than ‘<’**

**Less than or equal to ‘<=’**

**Extract all the employees whose salaries are greater than ALLEN**

select \* from emp where sal>=(select sal from emp where ename='ALLEN')

****

1. **Multi row sub query :**

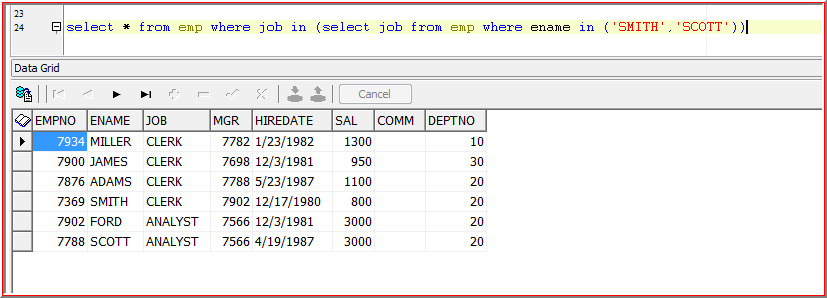
In these sub queries first the inner query will be executed and will return ‘’More than one record” as output and this output will be passed to outer query as input. So, that the outer query will executed and it will return the final output

**Multirow sub Query operators:**

**In, >any, <any, >all, <all**

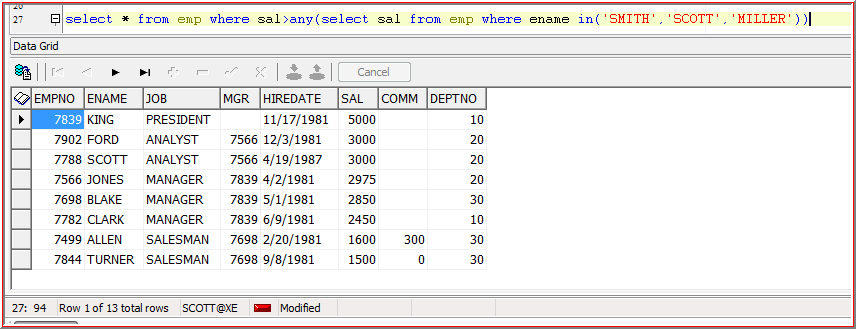
**Extract all the details who are colleagues of SMITH or SCOTT**

select \* from emp where job in (select job from emp where ename in ('SMITH','SCOTT'))

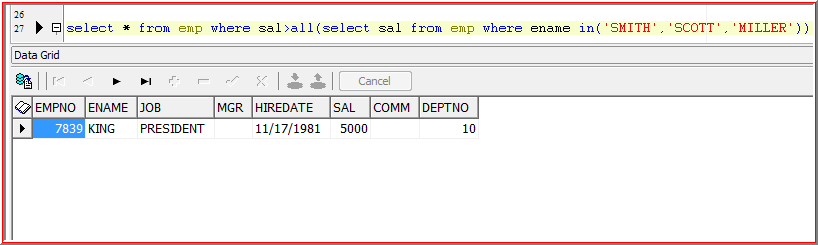
****

**>any**

select \* from emp where sal>any(select sal from emp where ename in('SMITH','SCOTT','MILLER'))

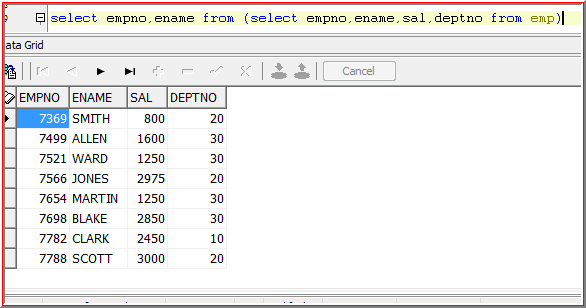


**>all**

select \* from emp where sal>all(select sal from emp where ename in('SMITH','SCOTT','MILLER')) 

**Writing Sub Query in ‘from’ clause**

**select \* from (select empno,ename,sal,deptno from emp)**

****

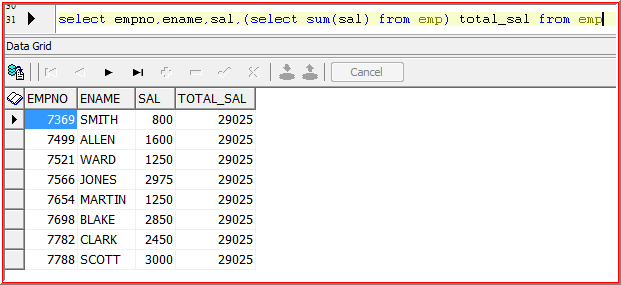
* **If we write the sub query in ‘from’ clause then it is called as ‘Inline View’**

**Writing sub query in ‘select’ clause**

If we write the sub query in the select clause then it is called **‘Scalar Sub Query’**

**select empno,ename,sal,(select sum(sal) from emp) total\_sal from emp**

select empno,ename,sal,(select sum(sal) from emp) total\_sal from emp



**CO-Related Sub Query :**

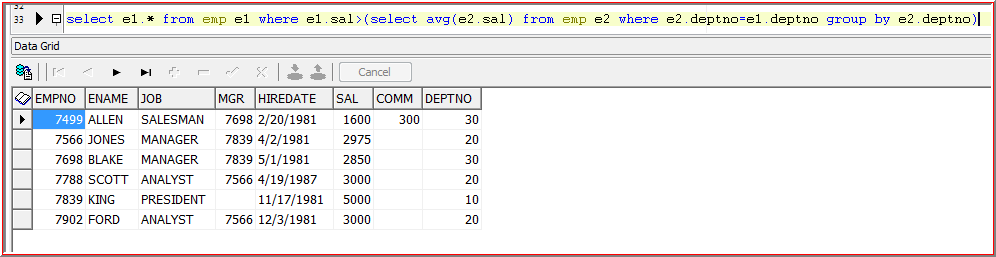
In these queries first the outer query will be executed and this output will be passed to inner query as input. So, that the inner query will be executed and will return the output and this output will be passed to outer query as input. So, that the outer query will be executed and will return the final output

In Normal Sub Queries the sub query will be executed only once i.e a statement, where as in co related sub queries the sub query will executed **once per a record**

**Example:**

**Extract all the employee details who are earning the salary more than their dept average salary**

select e1.\* from emp e1 where e1.sal>(select avg(e2.sal) from emp e2 where e2.deptno=e1.deptno group by e2.deptno)



**Views**

It is a “Database Object” which is created based on a **table or another view**

It’s like a window through which can be data viewed or changed

**Syntax**

Create [or replace] [force] view view\_name

Is/as

{sub query}

[with read only]

[with check option ]

**Create view from table**

**create view emp\_v1**

**as**

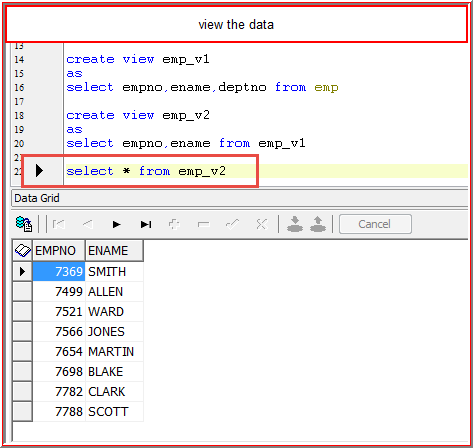
**select empno,ename,deptno from emp**

**Create view from view**

**create view emp\_v2**

**as**

**select empno,ename from emp\_v1**



**Data Change**

**F4 is the shortcut to see the view**

Whatever the manipulations we are performing on a view, all those will manipulations effect all the base tables or base views of that view

**[or Replace]:**

By using this option we can replace an existing view

Create or replace view emp\_v1

As

Select empno,ename,deptno from emp

**[Force]**

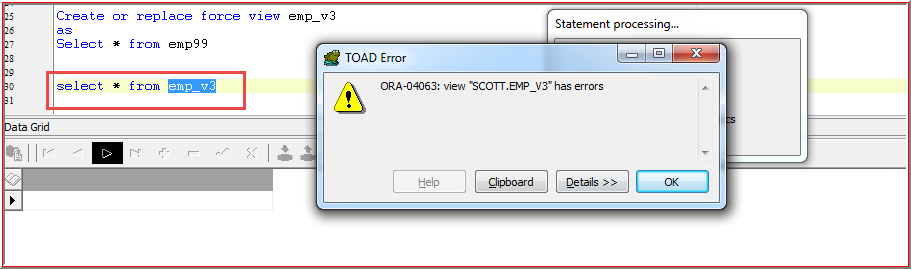
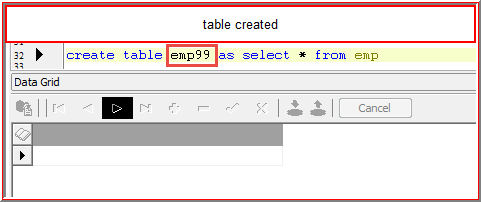
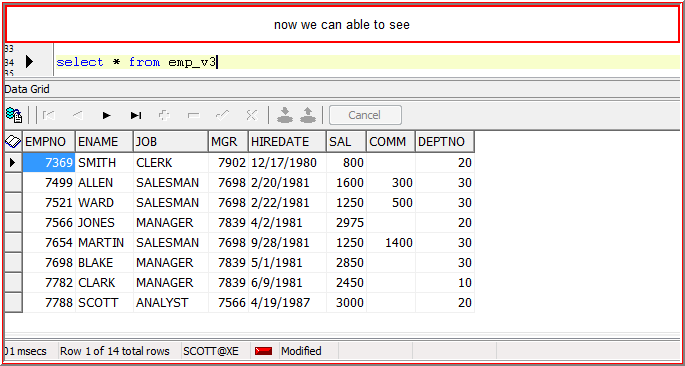
If we can use this option we can create a view even the base table does not existed also. But we cannot access that view until and unless the base table or base view is created

Create or replace force view emp\_v3

as

Select \* from emp99

(table not exist, but it will create a view)

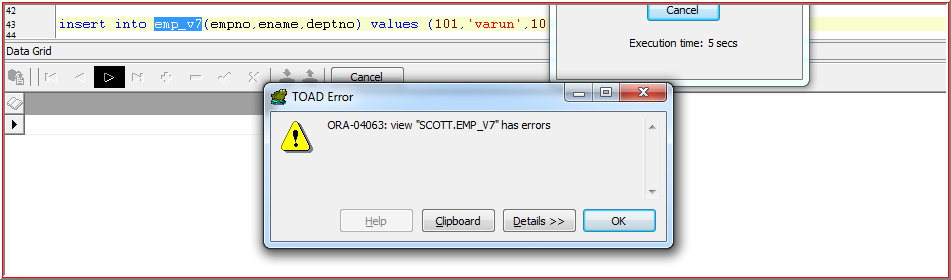
**[With read only]**

If we use this option then it won’t allow you to write (DML Operations) the data i.e It will allow you to read the data

Create or replace force view emp\_v7

As

Select \* from emp87 with read only



[With check option]

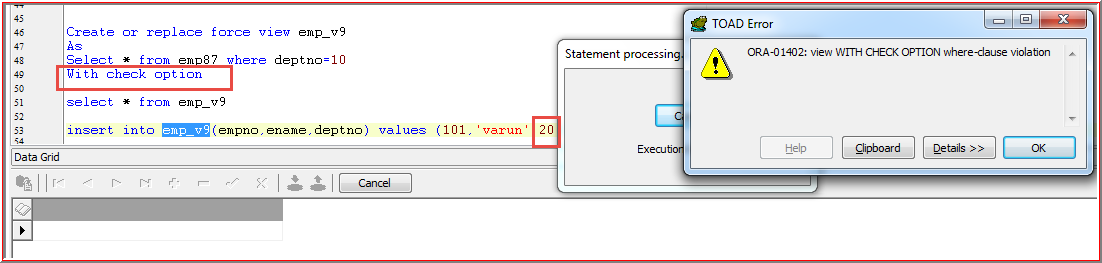
If we use this option then it will check for the conditions which we are writing in where clause of a sub query while creating a view

Create or replace force view emp\_v9

As

Select \* from emp87 where deptno=10

With check option



View with check option where clause violation

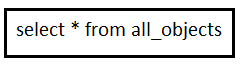
**Types of Views**

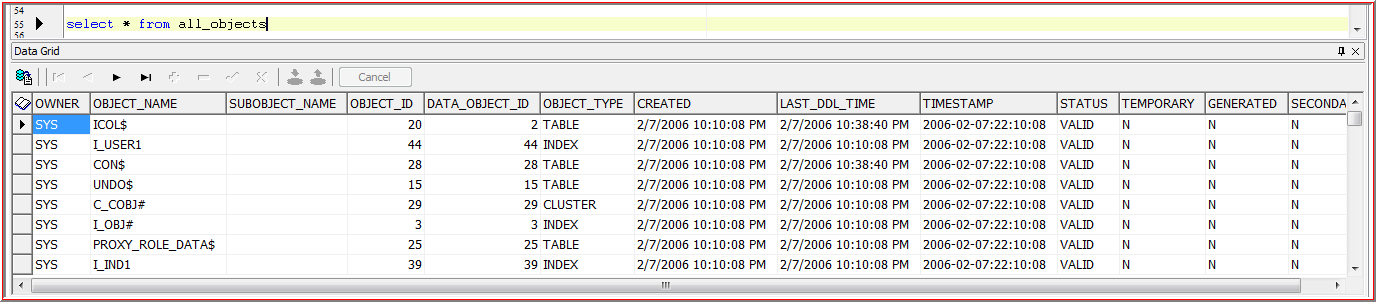
1. Simple View
2. Complex View

**Data Dictionary Tables or Meta Data Tables**

**Meta Data:**

Data about data is nothing but called as meta data

****



In this table all the objects will be stored

Table stores data, table data stores in another table (like table\_name, attributes etc)

**Database Objects:**

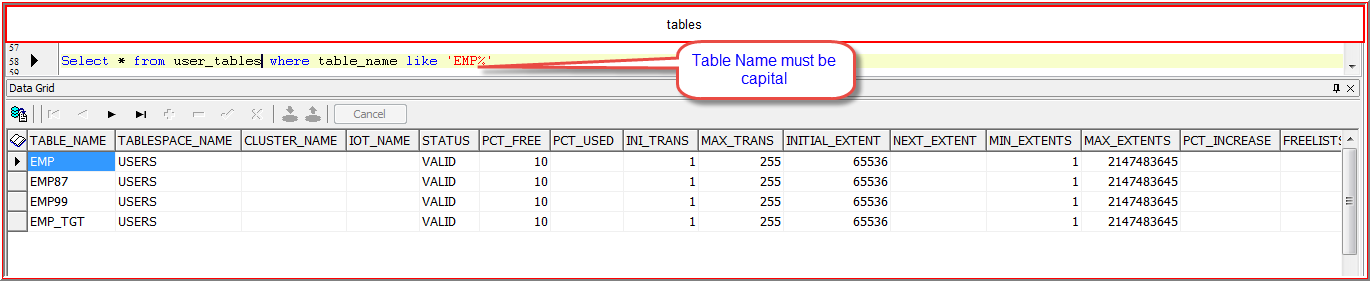
Whatever the objects we are storing in the database all those objects are called as Database objects

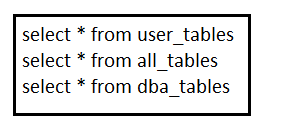
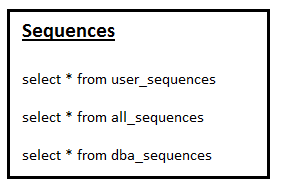
**Types of Database Objects:**

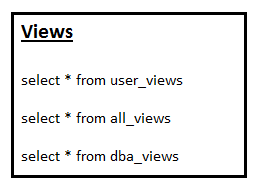
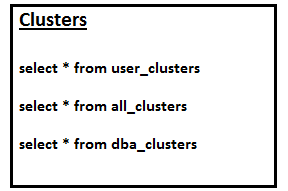
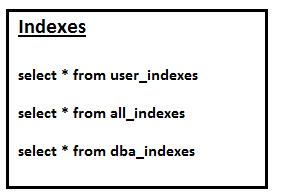
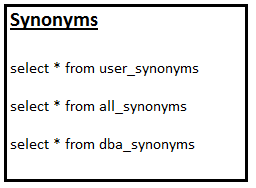
* Tables
* Views
* Sequences
* Synonyms
* Indexes
* Clusters

**Tables:**

Select \* from user\_table where table\_name like ‘EMP%’



|  |  |
| --- | --- |
| **Types of Views** |  |
|  |  |
| **Simple View** | **Complex View** |
| It is created based on single Table or view v1-->e1 | It is created based on more than one table v1--> e1,e2 or v1,v2 |
| we cannot use group by clause | we can use group by clause |
| we cannot use group functions | we can use group functions |
| we can perform DML operations on a 'View' | we cannot perform DML operations on a 'View' |
| eg: emp\_v1 (Enter F4) Copy the code by click on second tab of view and paste into Toad |  |

**Materialized View:**

It is a snapshot of a table and will copy the memory in the Database

**Normalized View (Simple or Complex)**

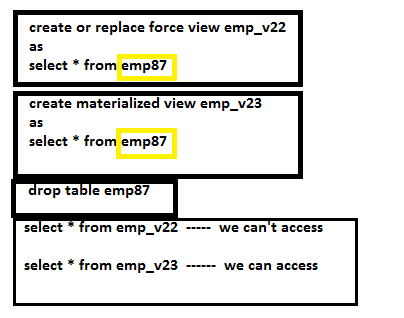
If we drop a base table or base views, then we can’t access that view (view will be existed in the DB, but we can’t access until and unless the base table or base view created) where as in **Materialized View,** If we drop Base Table or Base View also we can access that view

Eg :

**Materialized view** is like photocopy, if the person will die, photocopy present

**Normal View** is like human being, if the human die nothing will be present

Creating Normal and Materialized views on same Table



**SEQUENCES**

It is a Database Object used to generate primary key integer values

**Syntax:**

**Create sequence sequence\_name**

**[Increment by]**

**[Start with]**

**[Max value]**

**[Min value]**

**[Cycle]**

**[Cache]**

**Create sequence emp\_seq1**

* If we are not specifying any incremented by value or start with value the default by, it will take incremented value as ‘1’ and start with value also as ‘1’

**Accessing the sequence**

We can access the sequence by using ‘**pseudo columns’** **Nextval and currval**

**Pseudo columns** are columns which behaves like actual table columns but we can’t access or modify these columns

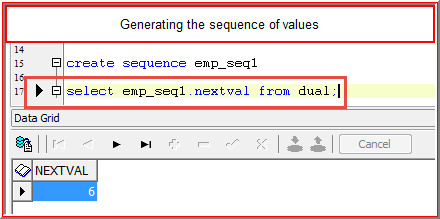
**Types of pseudo columns**

* **Rowid**
* **Rownum**
* **Nextval**
* **Currval**
* **Level**

**Nextval**

It will generate next available value in sequence

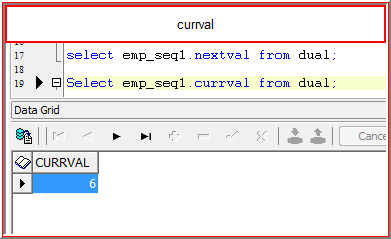
Eg: select emp\_seq1.nextval from dual;

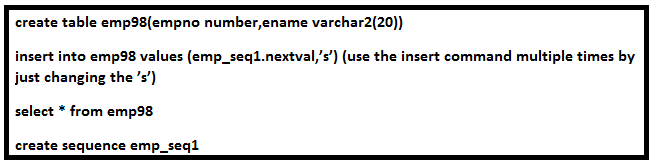


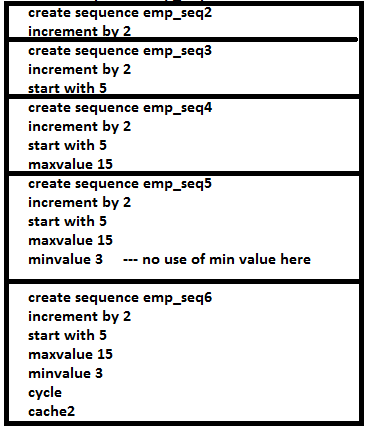
**Currval**

It will return current available value in the sequence

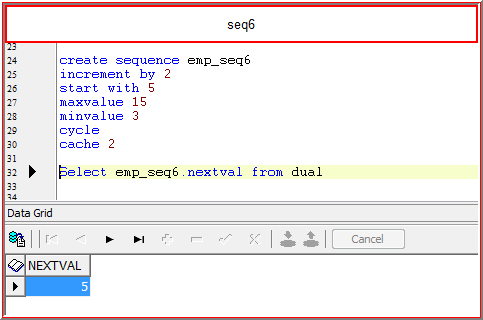
**Select emp\_seq1.currval from dual;**



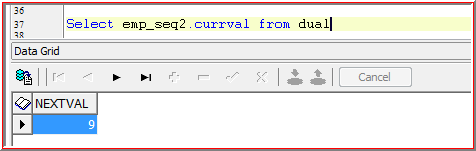




**Select emp\_seq6.nextval from dual**



Select emp\_seq2.currval from dual



**Synonyms**

Synonym is the **permanent alias name** for the object

Synonym will be stored in the Database

|  |  |  |
| --- | --- | --- |
|  | **Login with user name Scott** | **Login with user name Source** |
|  |  |  |
| 1 | create table emp87(empno number(20)) |  |
| 2 |  | Accesing  select \* from emp87 Data not accessed "Data Table or view does't exist" |
| 3 | Giving permission to Source to access Data |  |
| 4 | Grant all on emp87 to source |  |
| 5 |  | select \* from emp87 Data not accessed |
| 6 |  | select \* from **scott**.emp87 Data accessed(Need to add prefix before table name |
| 7 |  | To avoid every time using scott, we will create permanent name |
| 8 | create synonym emp87 for scott.emp87 |  |
| 9 |  | select \* from emp87 we can access without prefix |

**Index**

It is a database object used to improve the performance

It will merge the **same values into bunch of rows**

**Types of Index:**

* **Normal Index**
* **Composite Index**
* **Unique Index**

**Normal Index**

If we create the index on only one column then it is called as Normal Index

**Create index emp8\_deptno\_ind6 on emp8(deptno)**

**Select \* from emp8 where deptno=20**

**Select \* from emp8 where job=’manager’ ---no use of index**

**Composite Index**

If we create the index on more than one column then it is called as Composite Index

**Create index emp8\_deptno\_ind6 on emp8(deptno,job)**

**Select \* from emp8 where deptno=20 and job=’manager’**

**Select \* from emp8 where deptno=20 --It won’t work, Need to apply on both columns**

**Unique Index**

If we create a index on a column which doesn’t contain any duplicate values (It may contains NULL values) is called Unique Index

**Create unique index emp8\_uni\_empno on emp8(empno)**

**Constraints:**

Constraints are nothing but business rules, which are used to prevent invalid data while manipulating the Database objects

**Types of Constraints:**

* Not Null
* Unique
* Primary key
* Foreign key
* Check
* Default

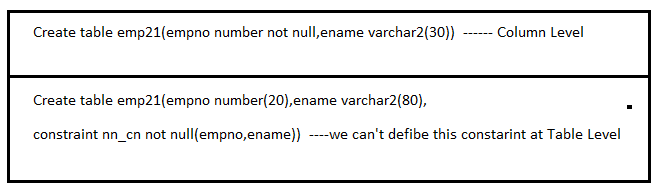
We can define all these constraints at two levels

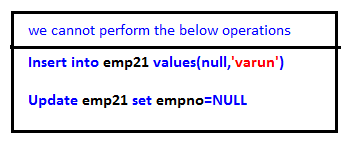
1. **Column Level** : we can define all constraints at column level
2. **Table Level** : we can define all constraints at table level except “Not Null” Constraints

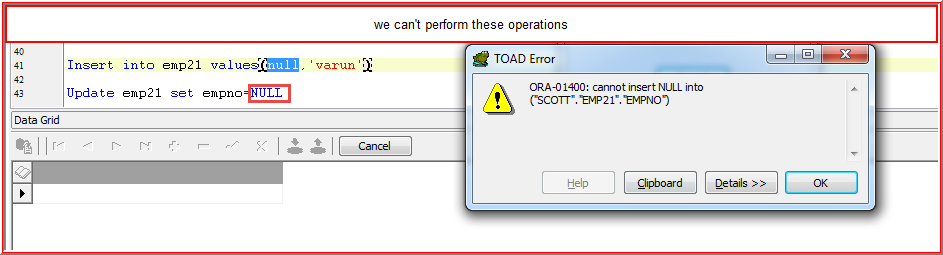
**1.Not Null**:

If we use this constraint then it won’t allow you to insert or update **NULL** Values

* We can apply this constraint only at column level
* We can use ‘n’ number of ‘NOT NULL’ constraints at column level



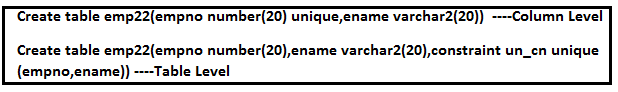




**2. Unique**

If we apply this constraint then it won’t allow you insert or update **duplicate values**

Allows **Null Values**

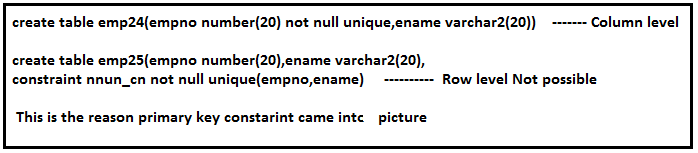
****

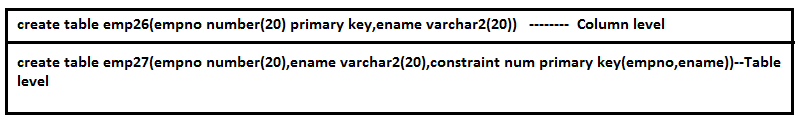
|  |  |  |
| --- | --- | --- |
| emp23 | | |
|  |  |  |
| **empno** | **ename** |  |
| 101 | a | allow |
| 101 | b | allow |
| 102 | b | allow |
| 102 | b | Not Allowed, violent row level |

**3. Primary Key**

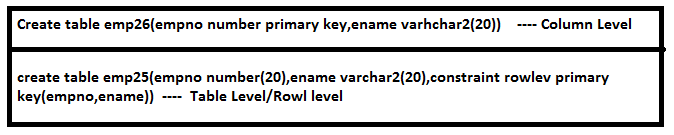
It is a combination of **“NOT NULL and UNIQUE CONSTRAINTS”**

i.e. it won’t allow you to insert or update **NULL VALUES** as well as **DUPLICATE VALUES**



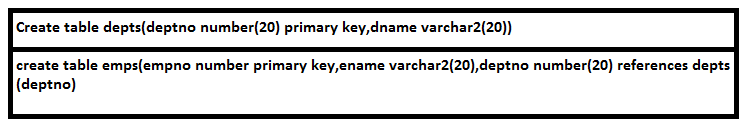


Not Null Unique combination we can’t apply at table level. So this is the reason primary key came.

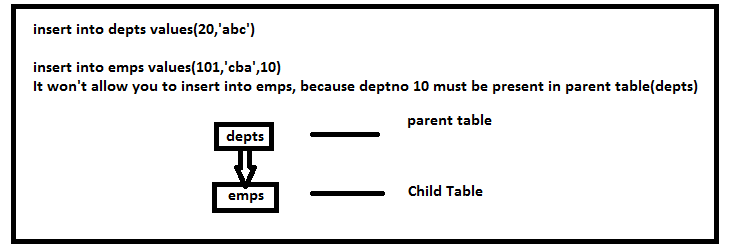


4. **Foreign Key:**

It will create relationship between primary key and foreign key within the same table or different table



If we want to insert any record in **emps table**, that need to be present in **depts table**



Delete from depts. where deptno=20

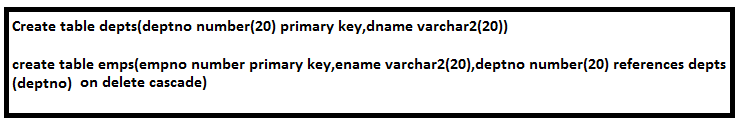
If we want to delete records in **parent table,** first delete in child’s, then in parent. So, it is difficult to search all childs to delete.

To avoid the above problem we comes with the concepts called

**1. ON DELETE CASCADE**

If we use this option in child table at the time of creating the table whenever we are deleting primary key values (i.e parent table—>dept’s) automatically foreign key values will be **Deleted** 🡪 deleted entire ‘row’ in child and parent table

Eg: drop table tablename



**Delete from depts where deptno=20**

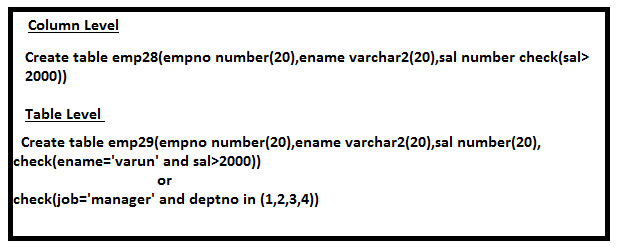
**Parent and child records both deleted**

**2. ON DELETE SET NULL**

If we use this option in child table at the time of creating the table whenever we are deleting primary key values (i.e parent table—>dept’s) automatically foreign key values will be **Set to Null** 🡪 Deleted only value and set to null in child table, but in parent table delete entire row

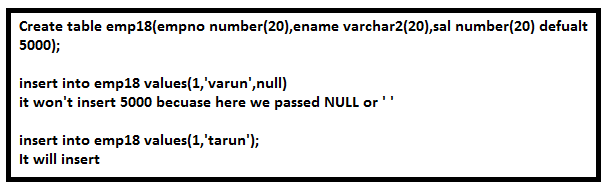
**5. Check Constraint:-**

It will check for the conditions while manipulating the data of the **Database Objects**



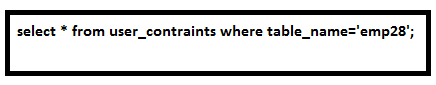
**6. Defualt (It’s not constraint)**

Whenever we are not passing any values (even Null) to the objects then it will take default value.



**7. Dropping Constraints:**

To find constraint Name



To drop the constraint using constraint\_name



**Cascade:**

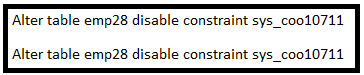
If we use this option, whenever we are deleting primary key constraints then it automatically foreign key constraints will be dropped



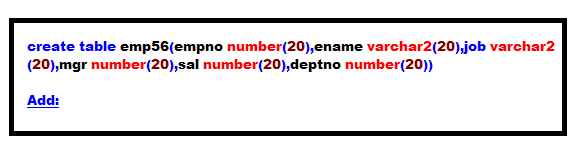
It will successfully execute

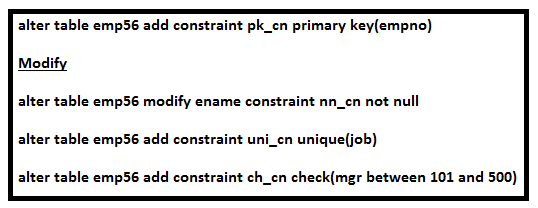
Use without cascade, It won’t allow you

**Enabling and Disabling a Constraint:**



**Adding Constraints to the existing table**

****

****

**alter table emp56 add constraint fk\_cn foreign key(deptno) references depts.(deptno)**

**Concepts in pseudo column**

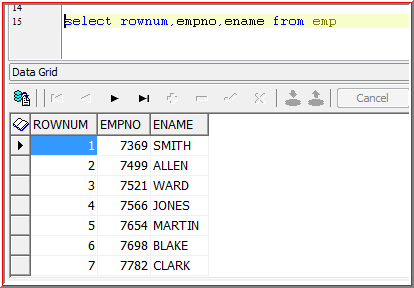
**Rownum:**

Rownum is nothing but a serial number or it’s like a sequence number and it will generated at the time of execution of a Query

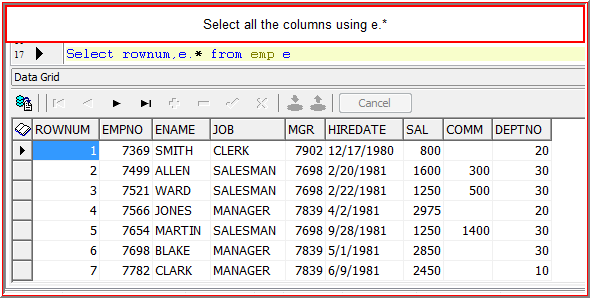
It is not constant for every record

Eg:

select rownum,empno,ename from emp



Select rownum,e.\* from emp e



**Rowid:** It is nothing but physical memory location of a record

It will be generated at the time of **insertion of a record**

**It is constant for every record**

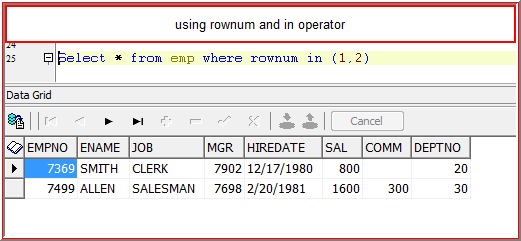
**Top N and Bottom N Analysis**

Extract top ‘2’ records

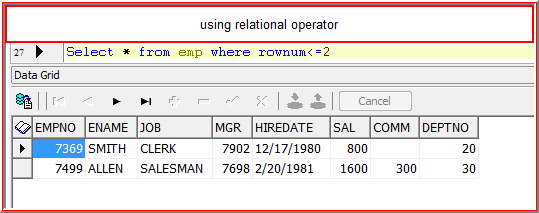
Select \* from emp

Select \* from emp where rownum=1

Select \* from emp where rownum=2



Select \* from emp where rownum<=2

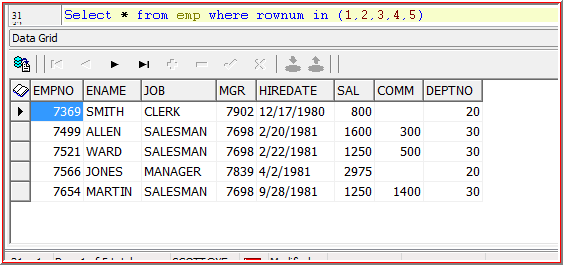


The condition should satisfy rownum 1 and 2 otherwise it won’t display

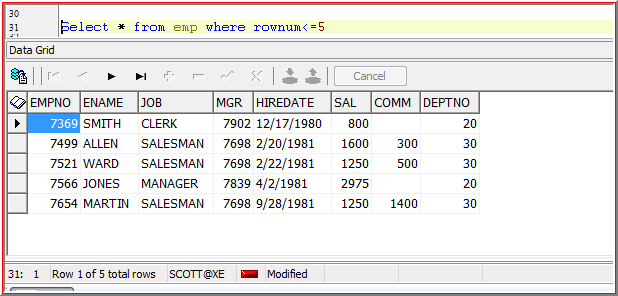
Extract Top ‘5’ records

**Select \* from emp**

**Select \* from emp where rownum in (1,2,3,4,5)**

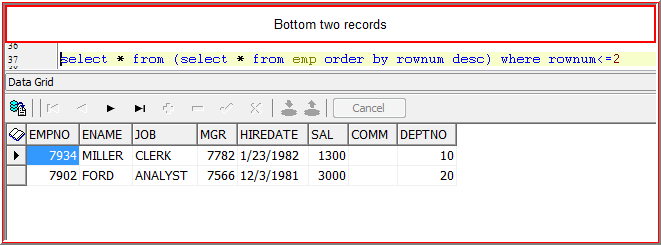
****

Select \* from emp where rownum<=5



Extract bottom 2 records

select \* from (select \* from emp order by rownum desc) where rownum<=2



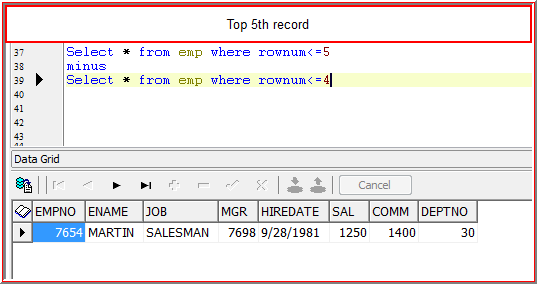
**Extract top 5th record**

Select \* from emp where rownum=5 -It wont work out

Select \* from emp where rownum<=5

minus

Select \* from emp where rownum<=4

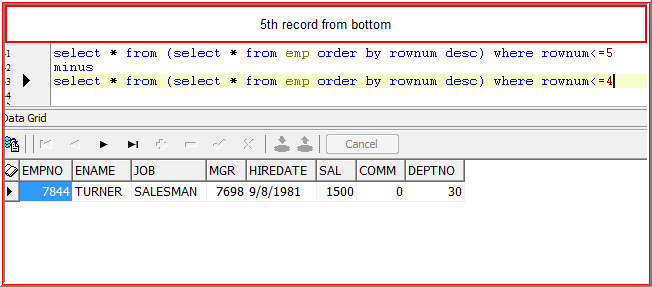
****

**Extract bottom 5th record**

select \* from (select \* from emp order by rownum desc) where rownum<=5

minus

select \* from (select \* from emp order by rownum desc) where rownum<=4

****

**Extract bottom 3rd and 7th record**

(Select \* from emp where rownum<=3

minus

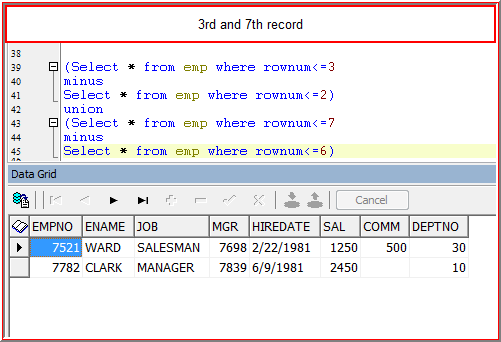
Select \* from emp where rownum<=2)

union

(Select \* from emp where rownum<=7

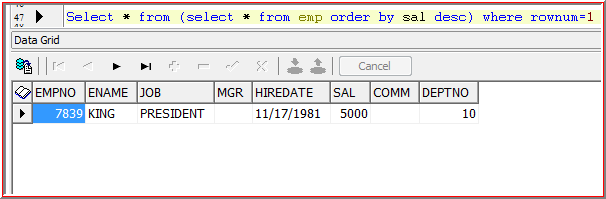
minus

Select \* from emp where rownum<=6)

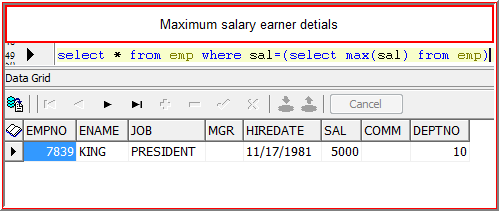
****

**Extract highest salary earner details of the organization**

**Select \* from (select \* from emp order by sal desc) where rownum=1**

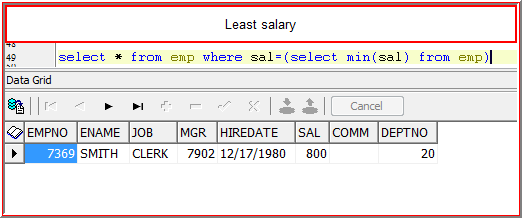
****

select \* from emp where sal=(select max(sal) from emp)

****

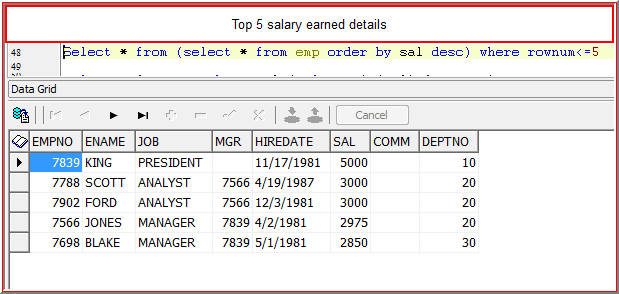
**Extract least salary earner details of the organization**

Select \* from (select \* from emp order by sal asc) where rownum=1

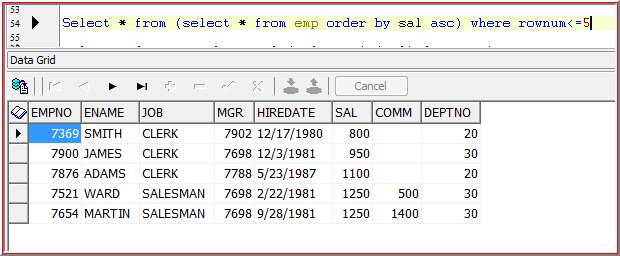
select \* from emp where sal=(select min(sal) from emp) 

**Extract Top 5 salary earner details of the organization**

Select \* from (select \* from emp order by sal desc) where rownum<=5

****

Select \* from (select \* from emp order by sal asc) where rownum<=5

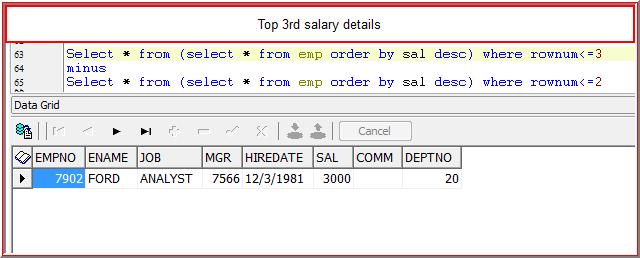
****

**Extract Top 3rd salary earner details of the organization**

Select \* from (select \* from emp order by sal asc) where rownum<=3

minus

Select \* from (select \* from emp order by sal asc) where rownum<=2

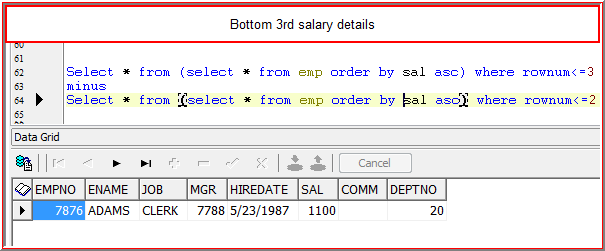
****

**Extract bottom 3rd salary earner details of the organization**

Select \* from (select \* from emp order by sal asc) where rownum<=3

minus

Select \* from (select \* from emp order by sal asc) where rownum<=2

****

**Extract top 5th and 7th salary earner details of the organization**

(Select \* from (select \* from emp order by sal desc) where rownum<=5

minus

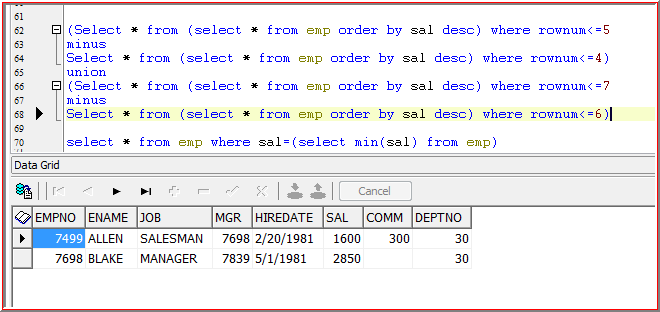
Select \* from (select \* from emp order by sal desc) where rownum<=4)

union

(Select \* from (select \* from emp order by sal desc) where rownum<=7

minus

Select \* from (select \* from emp order by sal desc) where rownum<=6)

****