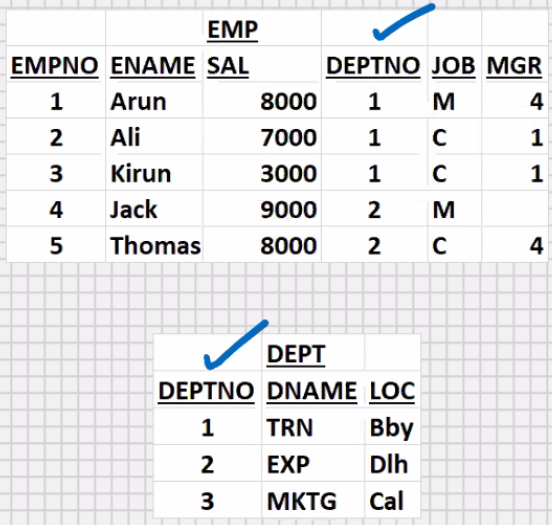
**MySQL – SQL – Correlated Sub-query (Very Important)**

* Using the exists operator (**EXISTS** is a special operator)



Display the dnames that contains employees:-

Display the dnames that do not contains employees:-

**Solution 1:-**

Select deptno from emp;

1

1

1

2

2

Select distinct deptno from emp;

1

2

Select distinct deptno from dept

Where deptno = any (1,2)

(Select distinct deptno from emp);

TRN

EXP

Select distinct deptno from dept

Where deptno in (1,2)

(Select distinct deptno from emp);

TRN

EXP

Select distinct deptno from dept

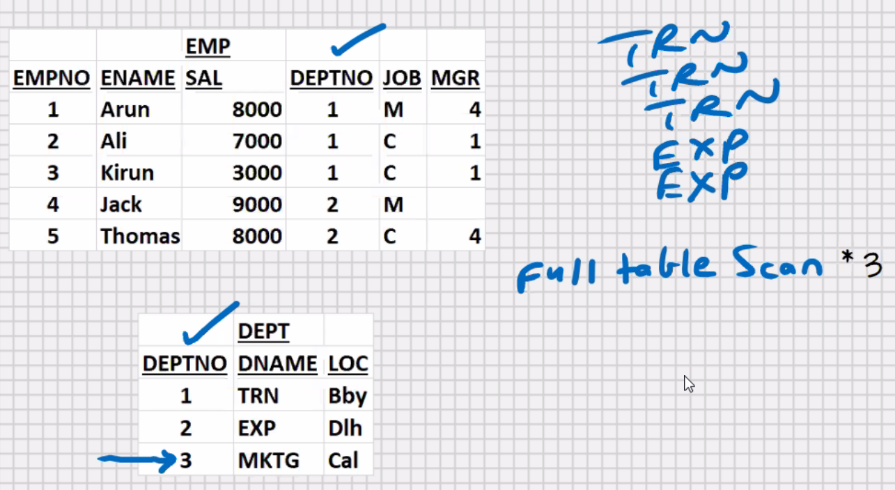
Where deptno not in (1,2)

(Select distinct deptno from emp);

MKTG

**Solution 2:-**

Select dname from emp, dept

Where dept.deptno = emp.deptno;

TRN

TRN

TRN

EXP

EXP

Select distinct dname from emp, dept

Where dept.deptno = emp.deptno;

TRN

EXP

**Solution 3:-**

* If you have a join alongwith subquery to make it work faster, use correlated subquery(use the EXISTS operator)
* This is the exception when sub query is faster than join

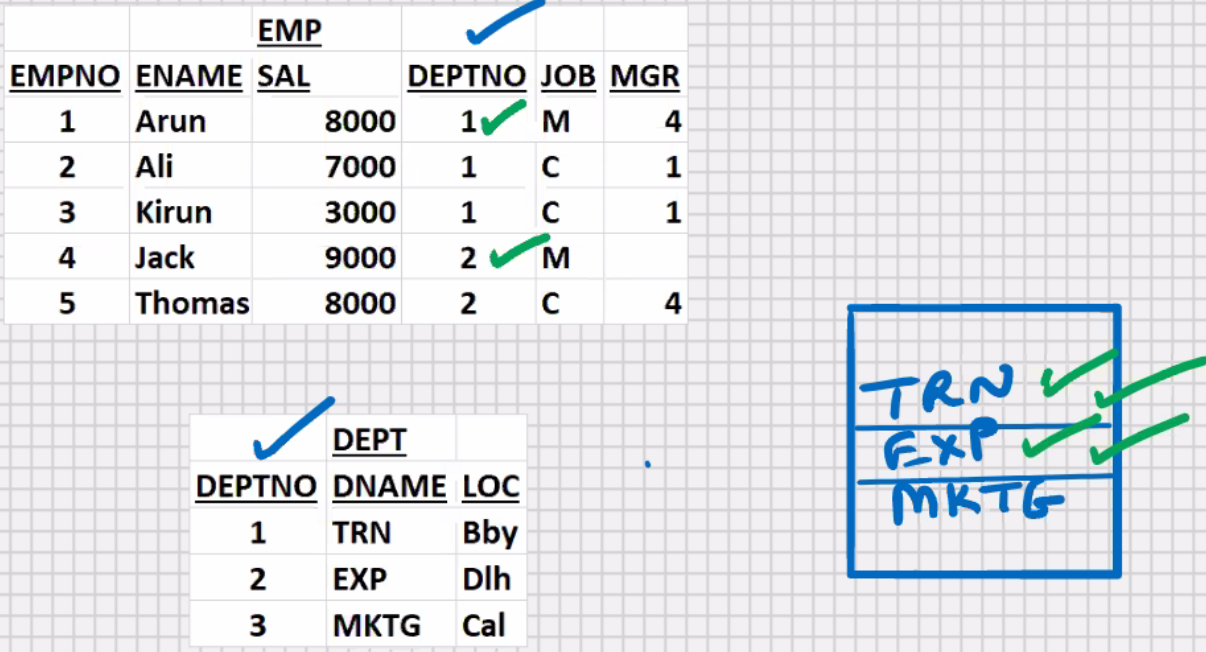
Select dname from dept where **exists**

(select deptno from emp

Where dept.deptno = emp.deptno);

TRN

EXP



* First the main query is executed
* For every row returned by main query, it will run the sub-query once the sub-query returns a boolean TRUE or FALSE value
* if subquery returns TRUE value, then main query is executed for that row
* if subquery returns FALSE value, the main query is not executed for that row
* unlike earlier we don’t use **DISTINCT** here, hence no sorting takes place in server RAM; this speeds it up
* unlike earlier, the number of full table scans is reduced; this further speeds it up

Select dname from dept where **not** **exists**

(select deptno from emp

Where dept.deptno = emp.deptno);

MKTG

**MySQL – SQL – Set Operators**

* based on set theory
* foundation of RDBMS is based on set theory(founder of RDBMS (1968)-> Dr. Codd)
* data is intersection of row and column

EMP1

|  |  |
| --- | --- |
| EMPNO | ENAME |
| 1 | A |
| 2 | B |
| 3 | C |

EMP2

|  |  |
| --- | --- |
| EMPNO | ENAME |
| 1 | A |
| 2 | B |
| 4 | D |
| 5 | E |

Select empno, ename from 1

Union

Select empno, ename from 2;

EMPNO ENAME

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

1 A

2 B

3 C

4 D

5 E

**Union**-> will combine the output of both the select statements, and it will suppress the duplicates

* upper limit is 255 for joining select statement in union

Select **empno1**, ename from 1

Union

Select empno2, ename from 2;

**EMPNO1** ENAME

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

1 A

2 B

3 C

4 D

5 E

Select empno1, ename from 1 where………..

Union

Select empno2, ename from 2 group by ……….;(structure must match)

Select empno1, ename from 1

Union

Select empno2, ename from 2

Order by 1; // used order by clause after second select

* Order by clause is applicable to the final output

EMPNO1 ENAME

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

1 A

2 B

3 C

4 D

5 E

**Union all** -> will combine the output of both the select statements , and the duplicates are not suppressed

Select empno, ename from 1

**Union all**

Select empno, ename from 2;

Order by 1;

Empno ename

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

1 A

1 A

2 B

2 B

3 C

4 D

5 E

Select empno, ename from 1

intersect

Select empno, ename from 2;

Empno ename

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

1 A

2 B

**Intersect** :- will return what is common in both the select statements, and the duplicates are suppressed

Select empno1, ename from 1

minus

Select empno2, ename from 2;

Empno1 ename

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

3 C

**minus** :- will return what is present in the select statements, and not present in the select statement, and the duplicates are suppressed

select job from emp where deptno = 10

minus

select job from emp where deptno = 20;

JOB

-----

PRESIDENT

select job from emp where deptno = 10

intersect

select job from emp where deptno = 20;

JOB

-----

MANAGER

CLERK

Union, union all, intersect , minus

Select …………….

Union

Select …………….

Union all

Select …………….

intersect

Select …………….

minus

Select …………….

Union

Select …………….

Order by 1;

* Max upto 255 select statements
* (this limit of SQL can be exceeded with the help of views)

Union,union all -> available in all RDBMS(including MySQL)

Intersect, minus -> available in Oracle , **not available in MySQL**

* Multiple select statements; brackets for changing the precedence, **not supported in MySQL**

**Pseudocolumns**

* Pseudocolumns are fake columns (virtual columns)

e.g

a. computed columns (ANNUAL = sal\*12)

b. Expressions (NET\_EARNINGS = sal+comm)

c. Functions-based columns (e.g. TOTAL = sum(sal))

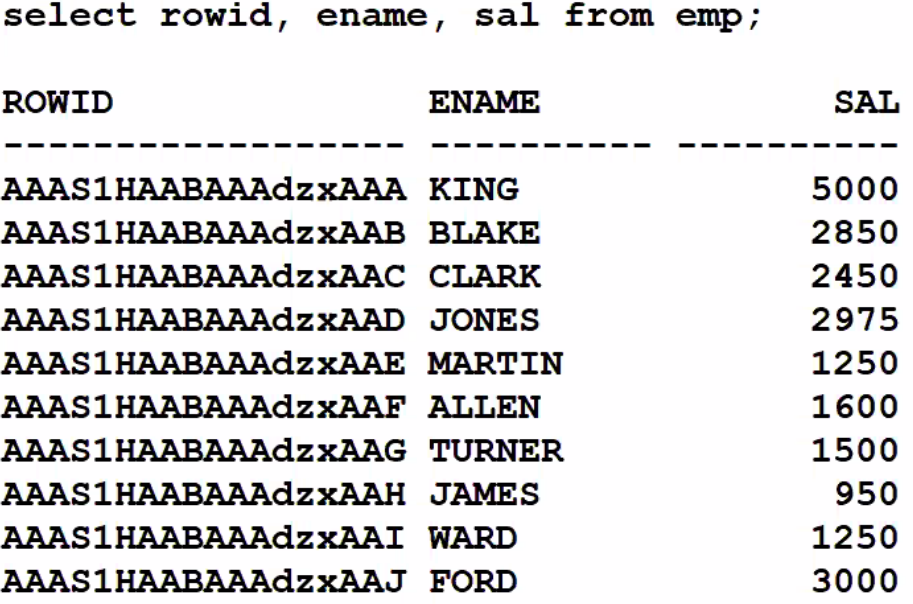
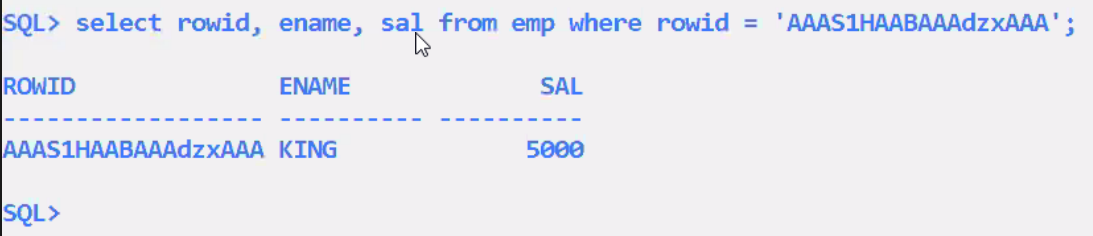
**RDBMS supplied Pseudocolumns**

Select ename, sal from emp;

Row indentifier

Select **rowid**, ename, sal from emp;

* Rowid is the row address
* Rowid is the address of the row in the DB server HD
* Rowid is the actual physical memory location where that row is stored in the DB Server HD
* Rowid is fixed length encrypted string of 18 characters
* When you select from a table , the order of rows in the ouptput will always be in the ascending order of rowid.
* When you update a row if the row length is increasing if the free space is not available at the current address, then the rowid may change
* No two rows of any table in the DB can have the same rowid
* Rowid works as an unique identifier for every row in the DB server HD



* **You can use rowid to update or delete duplicate rows in the table**

Uses: Rowid is used internally by MySQL and Oracle:-

1. To Distinguish between 2 rows in the DB server HD
2. Rowid works as an unique indentifier for every row in the DB server HD
3. For Row locking
4. To manage the indexes
5. To manage the cursors
6. Row management etc.

* Rowid is present in oracle and you can view it
* Rowid is present in MySQL but you cannot view it

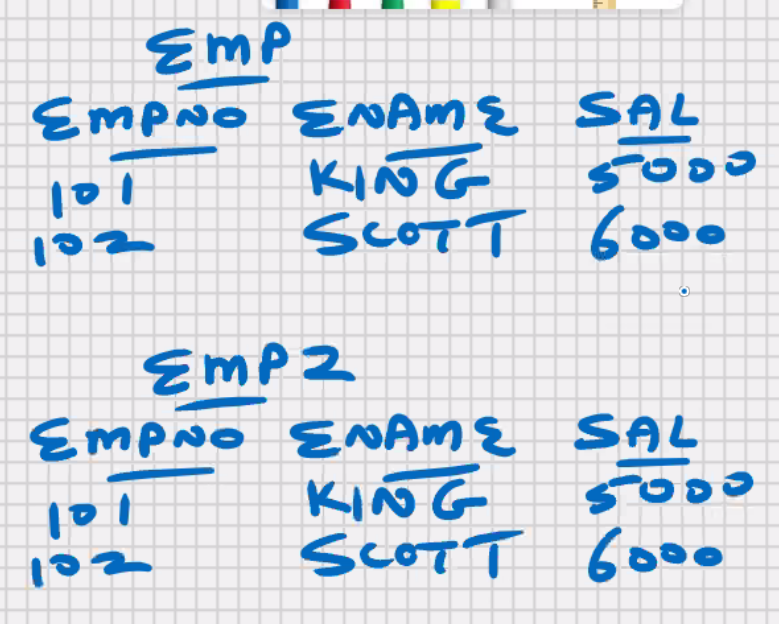
**MySQL – SQL – ALTER TABLE (DDL COMMAND)**

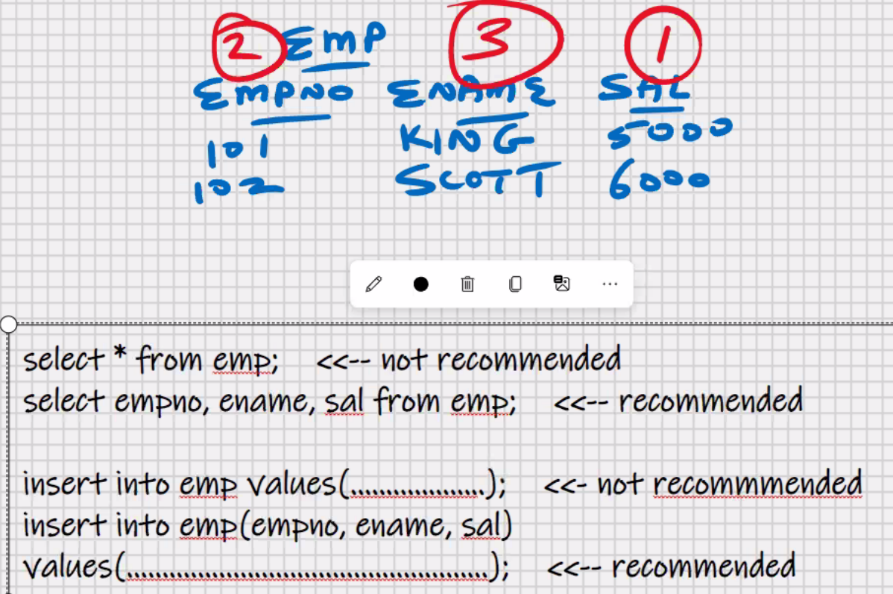
* Rename a table
* Add a column to the table
* Drop a column
* Increase width of column

Indirectly :

* Reduce the width of column
* Change the datatypes of column
* Copy the rows from one table into another table
* Copy a table (for testing purposes)
* Copy only the structure of table
* Rename a column
* Change the position of columns in table structure

(because of null values) (for storage considerations)





* Rename a table

In oracle:

Rename emp to employees;

* Rename is a DDL command (extra in MySQl and oracle)

In MySQL:-

Rename table emp to employees;

* Add a column to the table

Alter table emp add gst float;

Rename table emp to employees;

* Drop a column

Alter table emp drop column gst;

* Increase width of column

Ename varchar (20) -> ename varchar (30)

Alter table emp modify ename varchar(30);

Indirectly :

* Reduce the width of column

In MySQL:

Alter table emp modify ename varchar(20);

* Data will get truncated (*that’s dangerous yaar*)

In Oracle:

Alter table emp add x varchar(25);

Update emp set x = ename , ename = null;

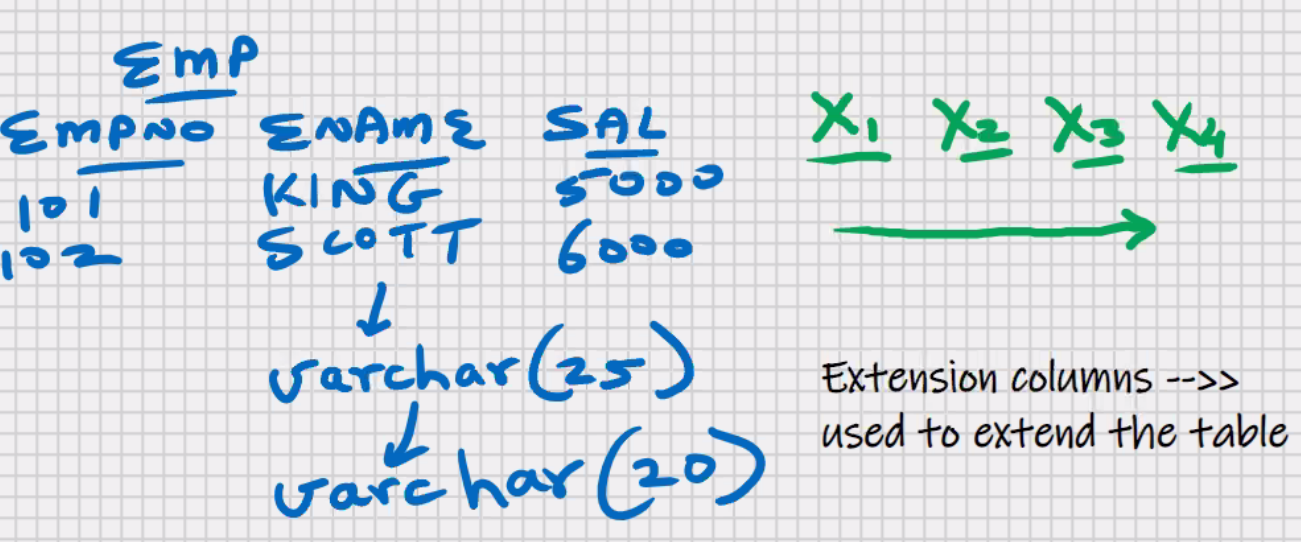
Alter table emp modify ename varchar(20);

/\*data testing with x column\*/

Update emp set ename = x;

Alter table emp drop column x;

* In oracle, You can reduce the width provided the contents are null
* Above procedure is recommended for MySQL also
* You must have few extra columns in every tables(**Implement in projects**)



* Change the datatypes of column

Change data type of empno from int to char(4)

Alter table emp add x int;

Update emp set x = empno, empno = null

Update emp set empno = null;

Alter table emp modify empno char (4);

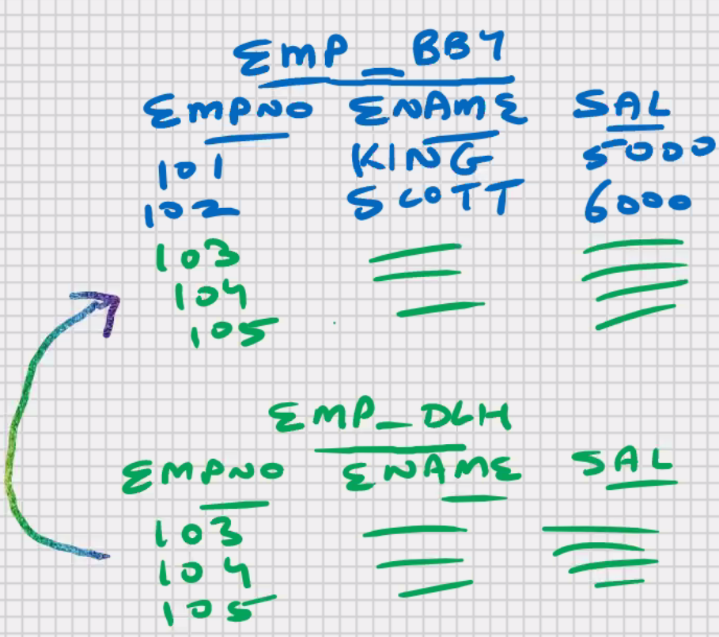
/\*data testing with x column\*/

Update emp set empno = null;

Alter table emp drop column x;

* In Oracle, you can change the datatype provided the contents are null
* Used above method for MySQL also
* Copy the rows from one table into another table

Insert into emp\_bby select \* from emp\_dlh;

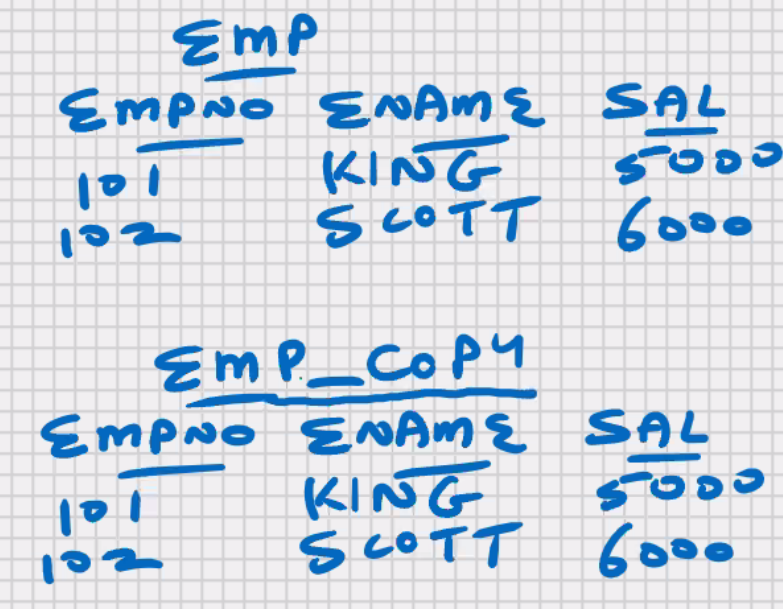


**To copy certain rows only:-**

Insert into emp\_bby select \* from

Emp\_dlh where ……………;

* Copy a table (for testing purposes)

create table emp\_copy

as

select \* from emp;

**To copy certain columns only:**

Create table emp\_copy

As

Select emp, ename from emp;

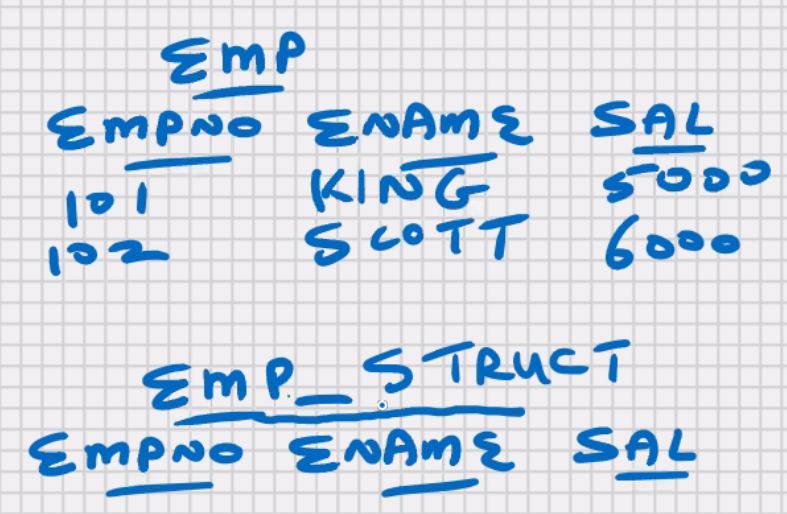
**To copy certain rows only:**

create table emp\_copy

as

select \* from emp where …………..;

* When you create a table using sub-query, the indexes on original table are not copy into the new table
* If you want the same indexes on the new table, you will have to create them manually
* When you create a table using subquery the constraints on original table are not copied into the new table, except for the not null constraint, because nullability is a feature of the datatype
* Copy only the structure of table



Method 1:-

Create table emp\_struct

As

Select \* from emp;

Delete from emp\_struct;

Commit;

Method 2:-

Create table emp\_struct

As

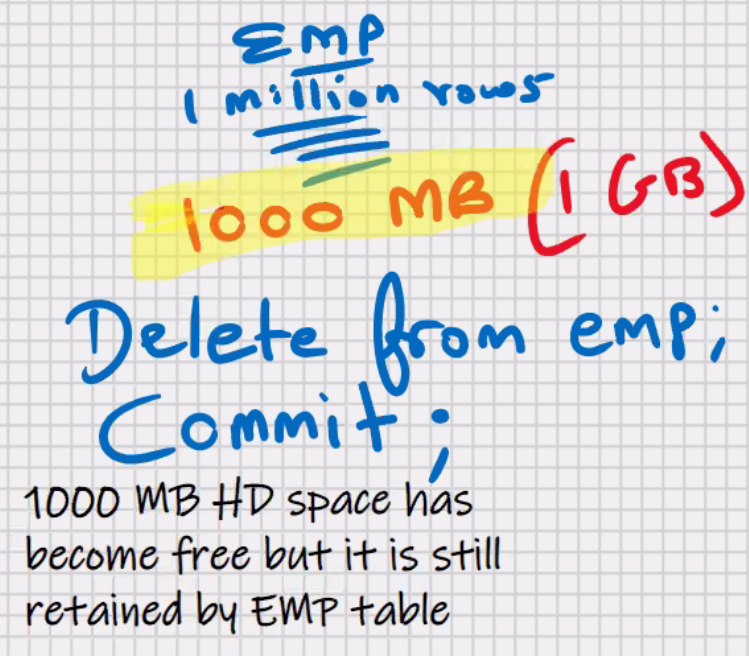
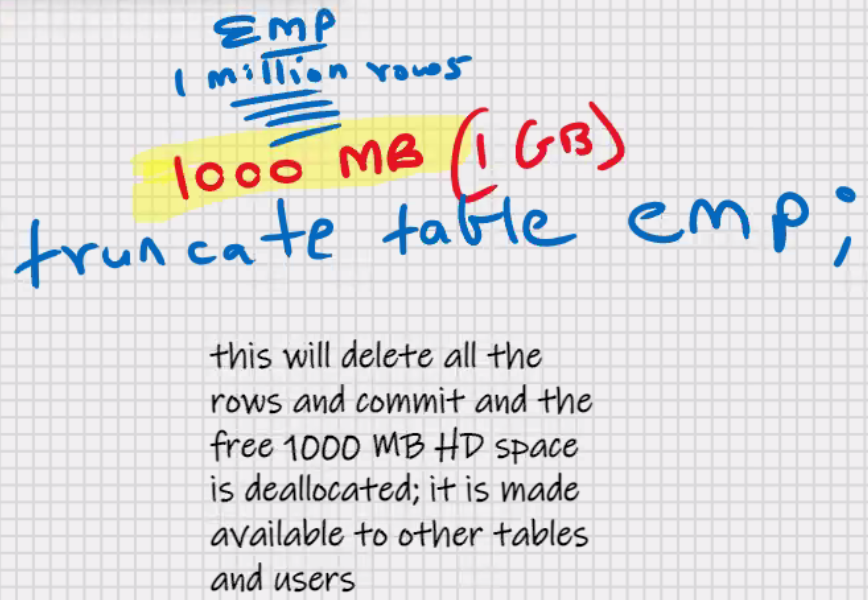
Select \* from emp;

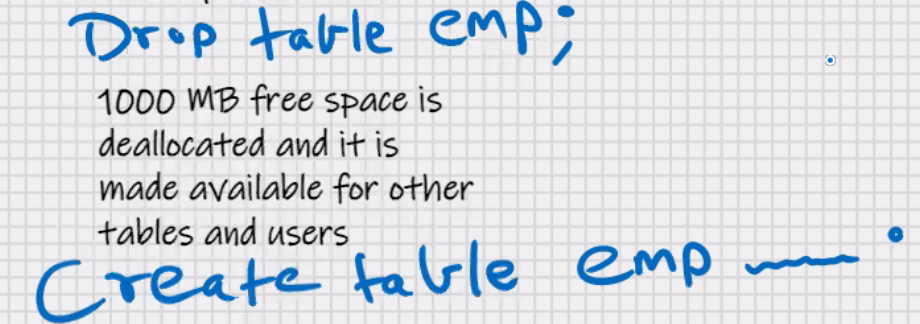
Truncate table emp\_struct; -> will delete all the rows and commit

Also

**Difference between DELETE and TRUNCATE**

|  |  |
| --- | --- |
| **DELETE** | **TRUNCATE** |
| DML commad | DDL command |
| Requires commit | Auto commit |
| Rollback is possible | Rollback is not possible |
| ANSI SQL and common for all RDBMS | Extra in MySQL and Oracle RDBMS |
| You can specify WHERE clause with delete | WHERE clause is not supported |
| Delete triggers on table will execute | Delete triggers on table will not execute |

****

****

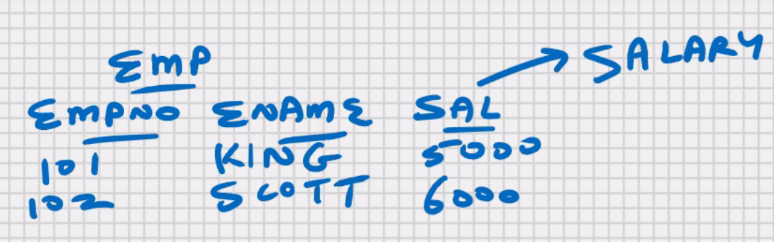
Method 3:-

Create table emp\_struct

As

Select \* from emp where 1 = 2; //give imposible condition

* Rename a column

create table emp\_copy

as

select emp, ename,sal salary

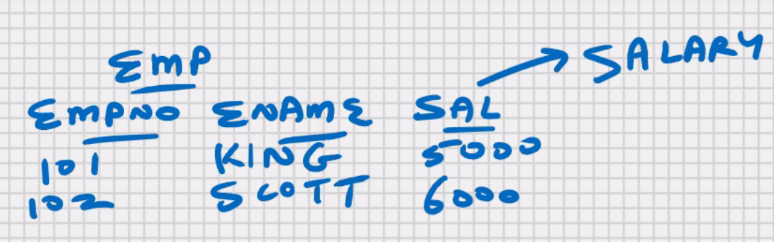
from emp;

drop table emp;

rename table emp\_copy to emp;

* Change the position of columns in table structure

(because of null values) (for storage considerations)

create table emp\_copy



as



select sal, empno ,ename from emp;

drop table emp;

rename table emp\_copy to emp;