1. Find first 3 topmost employees

```
select * from emp m
  -> where 3 >(select count(*) from (select distinct sal from emp) e
  -> where e.sal > m.sal)
  -> order by sal desc;
```

1. Find 3rd topmost employees

```
select * from emp m
  -> where 2=(select count(*) from (select distinct sal from emp) e
  -> where e.sal > m.sal)
  -> order by sal desc;
```

window functions

row_number()	assign unique value to every row within window
rank()	assign number to distinct values, if the values same then same rank will be assigned to both rows. but when more than one row gets same rankl then it will skip numbers and then the next rank will be assigned
dense_rank()	assign number to distinct values, if the values same then same dense rank will be assigned to both rows. but when more than one row gets same dense rank, then it will not skip in between numbers and then the next rank will be assigned
lag(val, n)	it will find nth previous value
lead(val,n)	it will find nth next value
first_value(sal)	it will find first value of the given column within window
last_value(sal)	it will find last value of the given column within window

In the following example partition by will divide the data into window and order by will arrange data within window

```
select deptno,ename,sum(sal) over (partition by deptno)
from emp
```

```
select empno,ename,sal,deptno,row_number() over (),rank() over (order by sal desc)
from emp;
```

1. to find highly paid employee

```
select *  from (
select empno,ename,sal,deptno,row_number() over () rownum,rank() over (order by sal desc) rn,dense_rank() over (order by sal desc) drn from emp) e
where e.drn=1;
```

2. to find highly paid employee in each department

```
select *  from (
select empno,ename,sal,deptno,row_number() over () rownum,rank() over (order by sal desc) rn,dense_rank() over (partition by deptno order by sal desc) drn from emp) e
where e.drn=1;
```

3. to find first sal in each window
   select empno,ename,sal,deptno,first_value(sal) over (partition by deptno order by sal) fv
   from emp;
4. add next salary into current sal
   select empno,ename,sal,deptno,sal+lead(sal,1) over () lagval
   -> from emp;

indexes
2 types of indexes

1. clustered index
   a. there will be only one clustered index
   b. it does not require extra space because it is stored along with data in the table
2. non clustered index
   a. there can be many non clustered index
   b. these are stored outside table, and hence need extra space

indexes speed up the search action , but reduces the speed of DML statements, so donot create unnecessary indexes.

Types of indexes

1. primary key index-----indexes on primary key gets create automatically
2. unique index
   a. indexes on column with unique constraint gets created automatically
   b. the fields on which unique index is created, then duplicate values are not allowed in that column.

   create unique index idx_passport

   on emp(password desc)
3. regular index

   create index idx_passport

   on emp(sal)
4. full text indexes
   a. These are usually used on text type column,
   b. it stores phrases or words and their position

   create full text index idx_passport

   on emp(sal)

5. geospatial index – It is used when the field store geographical location.

to drop the index

drop index idx_passport on emp

show indexes from emp;

Views in mysql

There are 2 types views in database

1. view
2. materialized view---- when you are working on static data, then it is good to create materialized view.
   Once you fire the query 1 st time data will be retrieved,it will get save in RAM for the current session.
   It speeds up the job of retrieval of the data.

for every view, separate table will not get created, only base query gets stored for view.

if we fire select statement on view, then the base query will get executed

while creating view, if we use with check option, if view based on single table, and if view contains all not null columns, then one can perform DML operations on the view. and only valid data for which where condition is satisfied, can be added or removed or updated

To stop all DML operations on the view, use with read only option

why to use views

1. to give limited access to the table
2. hide complex queries under the view
3. we may hide table names, which increases security.

to create view

create <materialized> view < name of the view>

as

<base query>

to create a view for manager of dept 10 to give access to all the records of dept 10

create view mgr10

as

(select * from emp

where deptno=10

with read only option)

create view testview1

-> as

-> select empno,ename,e.deptno edeptno,d.deptno ddeptno,d.dname

-> from emp e, dept d

```
   -> where e.deptno=d.deptno;



create view testview
   -> as
   -> select deptno,job,sum(sal+ifnull(comm,0)),count(*)
   -> from emp
   -> group by deptno;
```