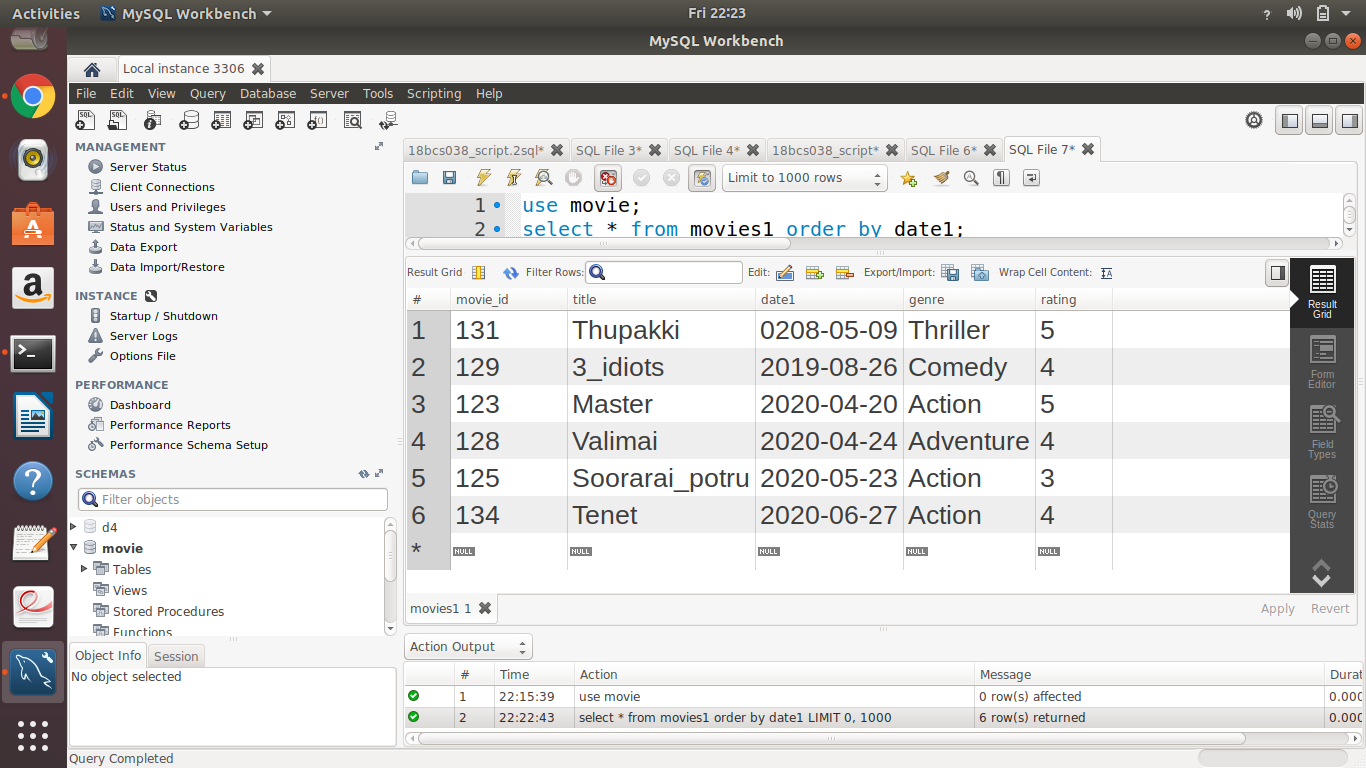
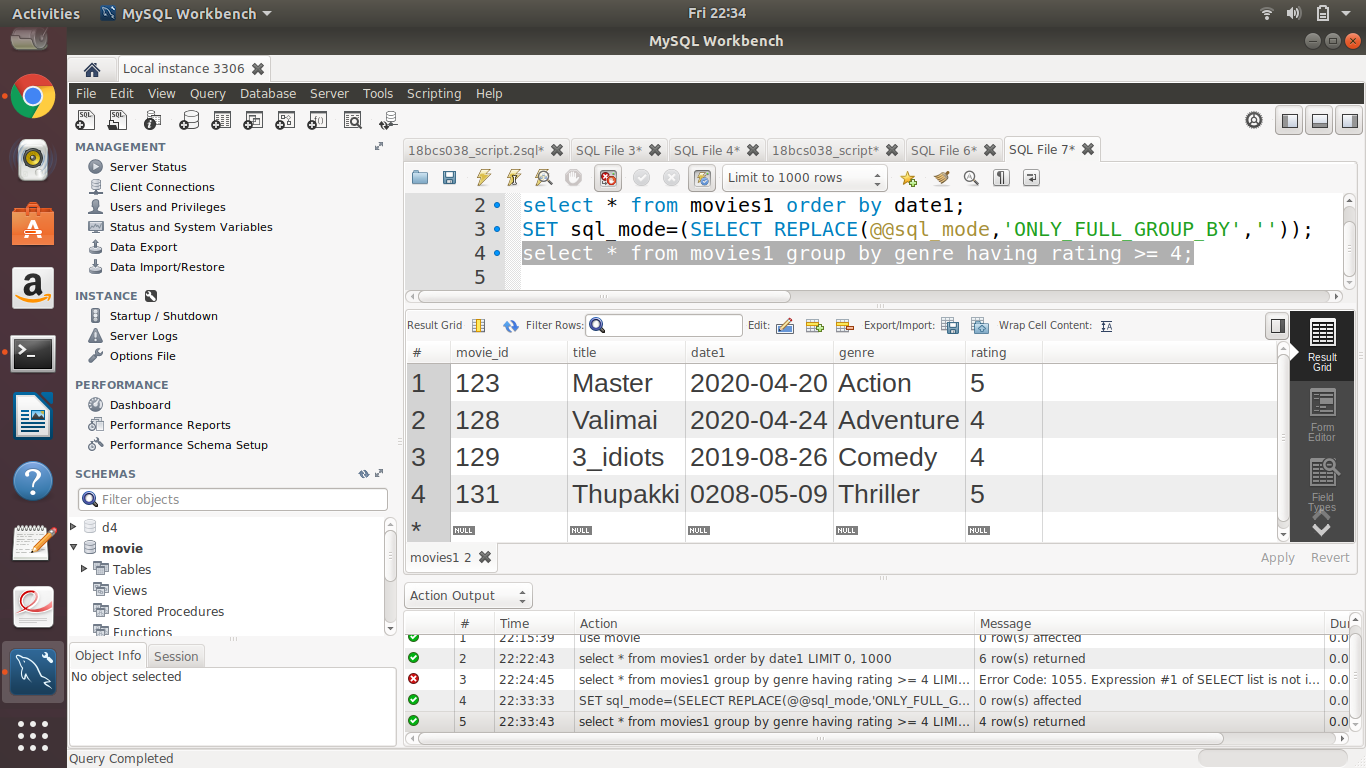
1. Order by clause.

Ans. select \* from movies1 order by date1;



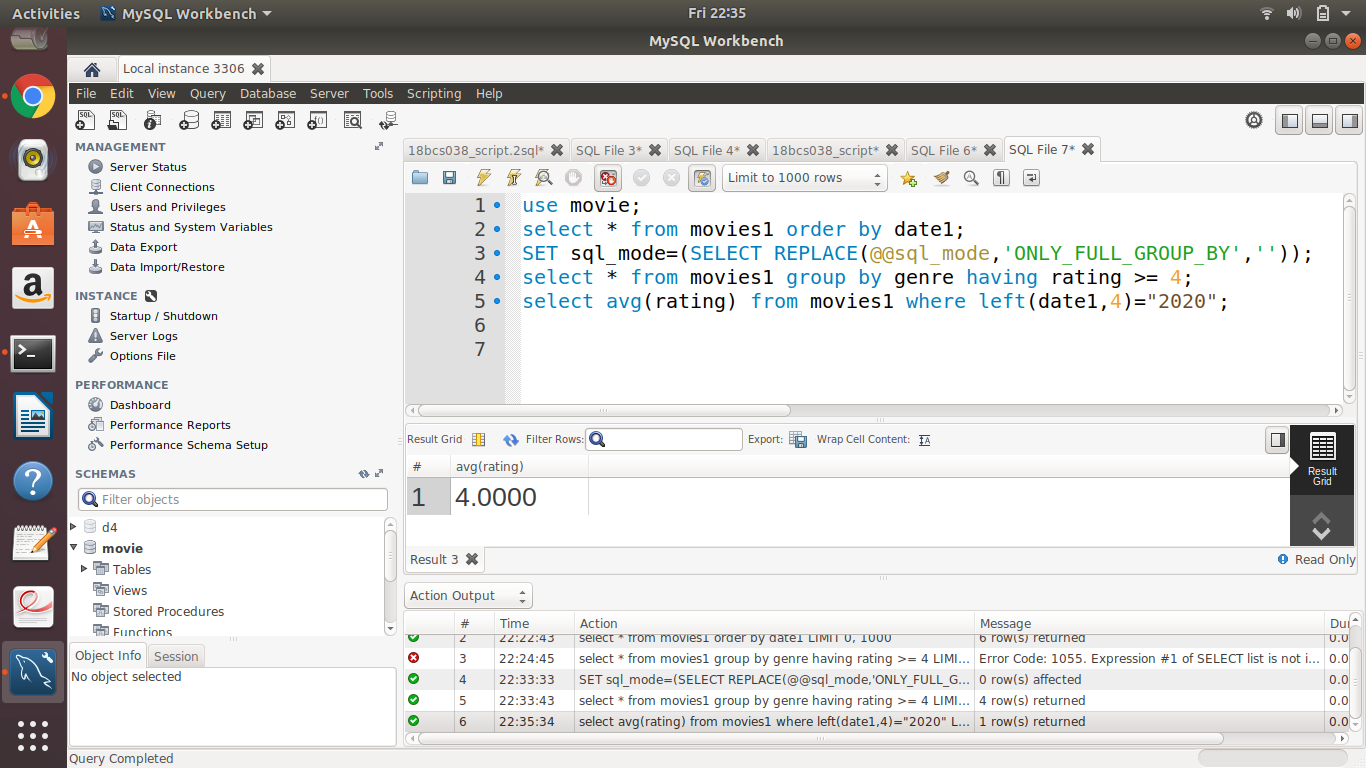
1. Group by and having.

Ans. select \* from movies1 group by genre having rating >= 4;



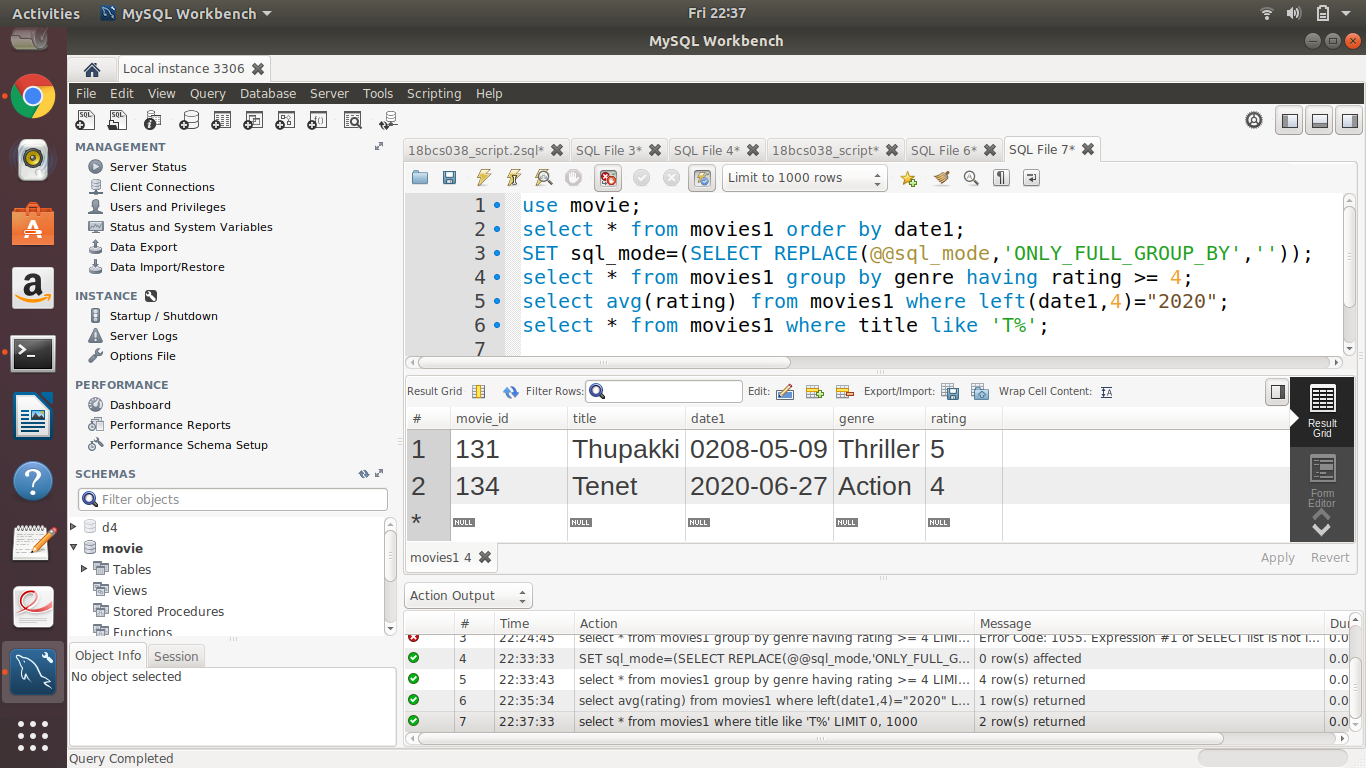
1. Aggregate functions.

Ans. select avg(rating) from movies1 where left(date1,4)="2020";



1. Logical operators especially with LIKE.

Ans. select \* from movies1 where title like 'T%';

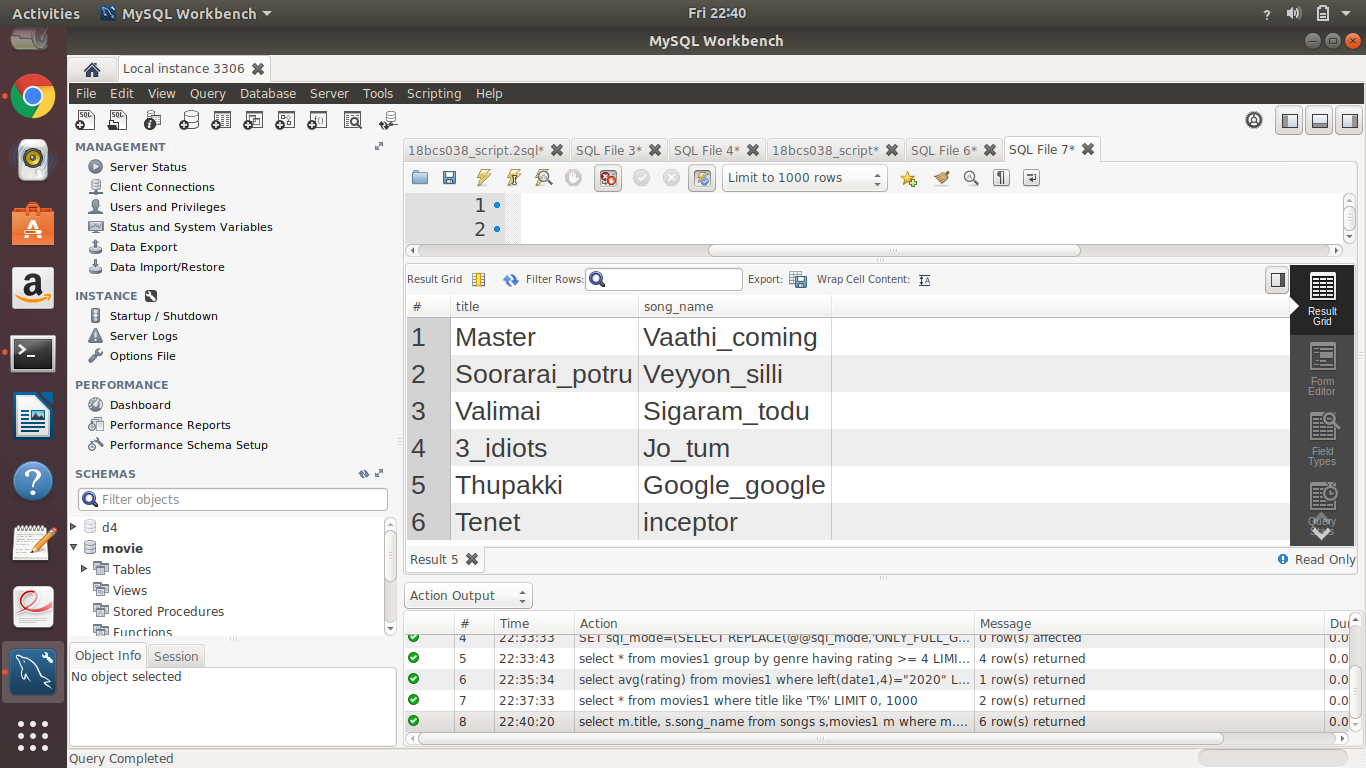


1. At least 4 nested queries specific to your Database, out of which at least 2 should have multiple subquery.

Ans.

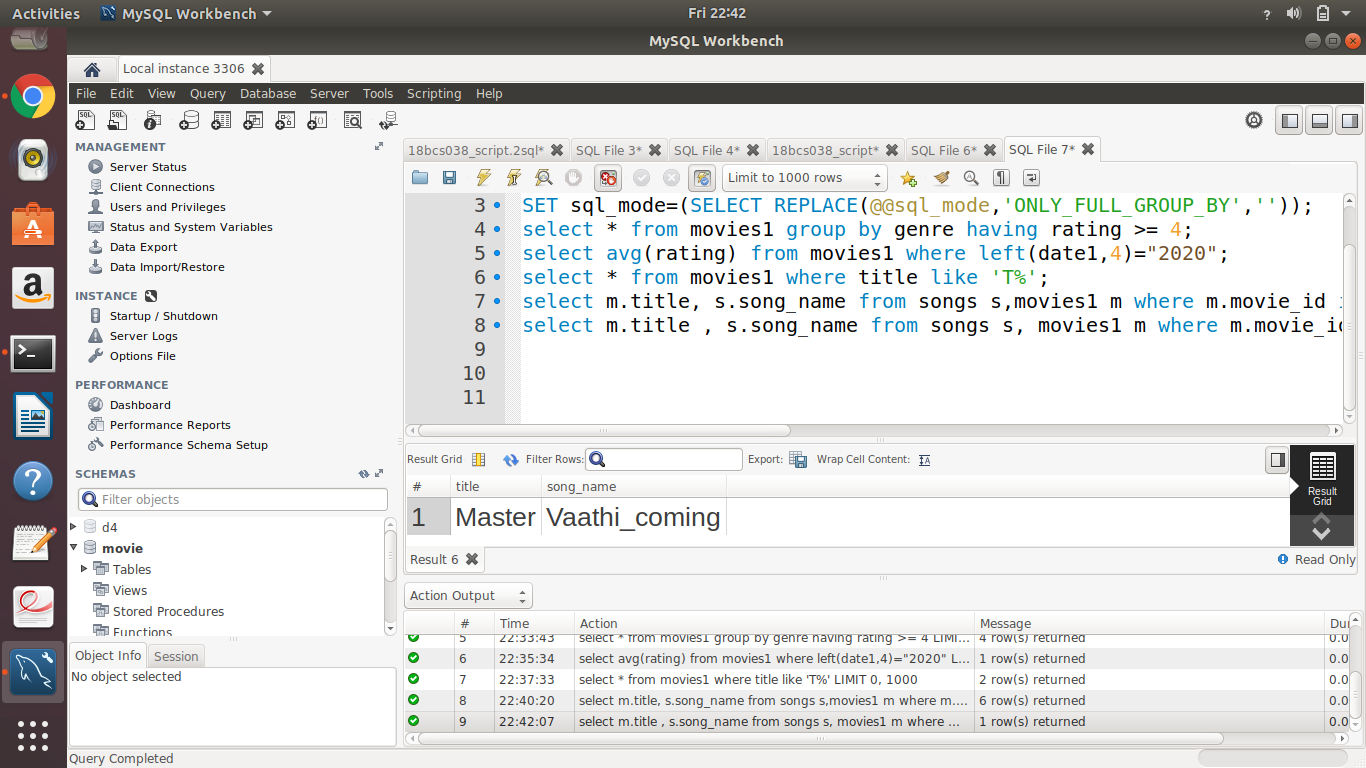
A.

select m.title, s.song\_name from songs s,movies1 m where m.movie\_id in (select m.movie\_id where m.movie\_id=s.movie\_id);



B.

select m.title , s.song\_name from songs s, movies1 m where m.movie\_id=(select m.movie\_id where m.movie\_id=s.movie\_id and m.title="Master");



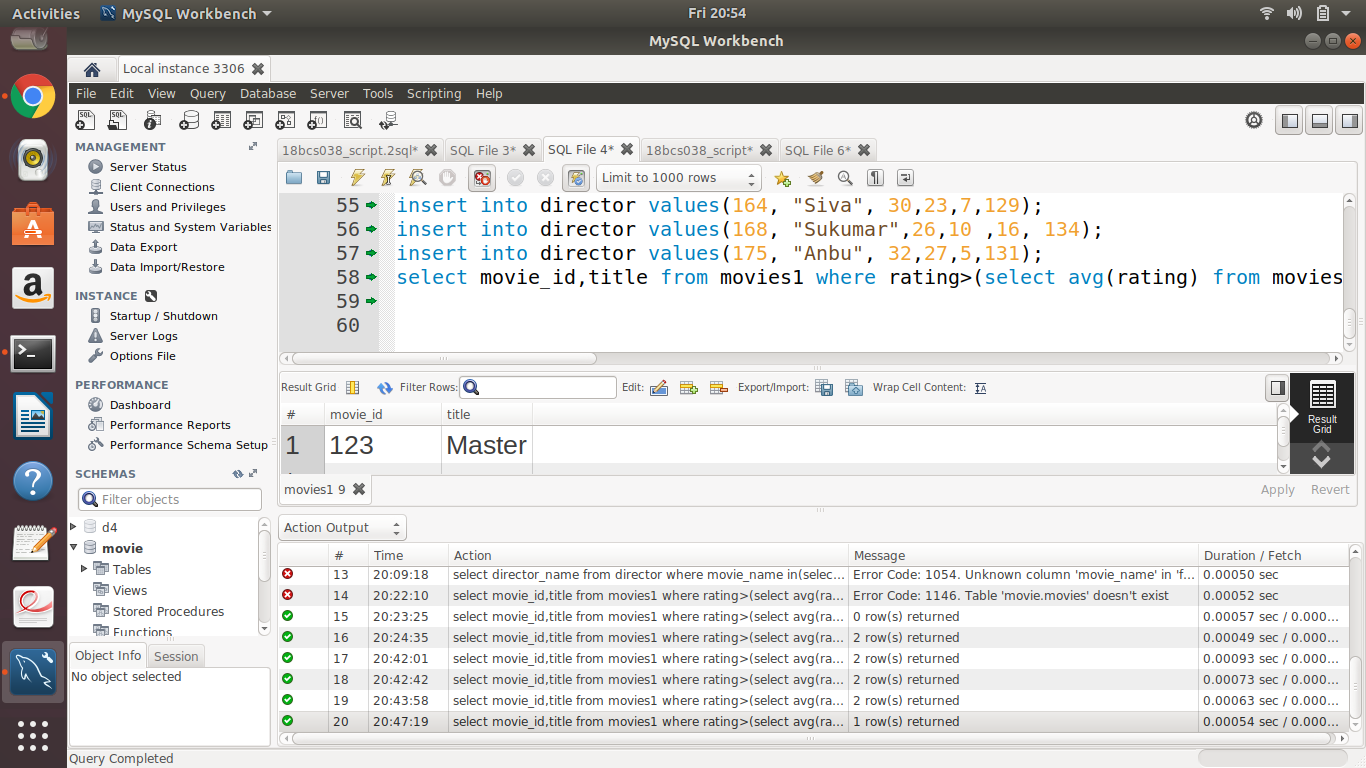
C.

**Query:**

select movie\_id,title from movies1 where rating>(select avg(rating) from movies1 where genre=(select genre from movies1 where movies1.movie\_id=134))and genre in(select genre from movies1 where movies1.movie\_id=134);

**Explanation:**

/\*The inner query returns the average of the ratings of those movies with genre same as the genre of movies with movie\_id=134. The outer query displays the movie details with rating > the value returned by the inner query and with genre same as that with movie\_id=134.\*/



D.

**Query:**

select songs.song\_name,songs.singer from songs where songs.movie\_id in(select movie\_id from director where no\_of\_flops >(select min(no\_of\_flops) from director));

**Explanation:**

/\*Displays the song name and singer from the songs of movies directed by directors with no\_of\_flops greater than the minimum number of flops of any director from the director table\*/

