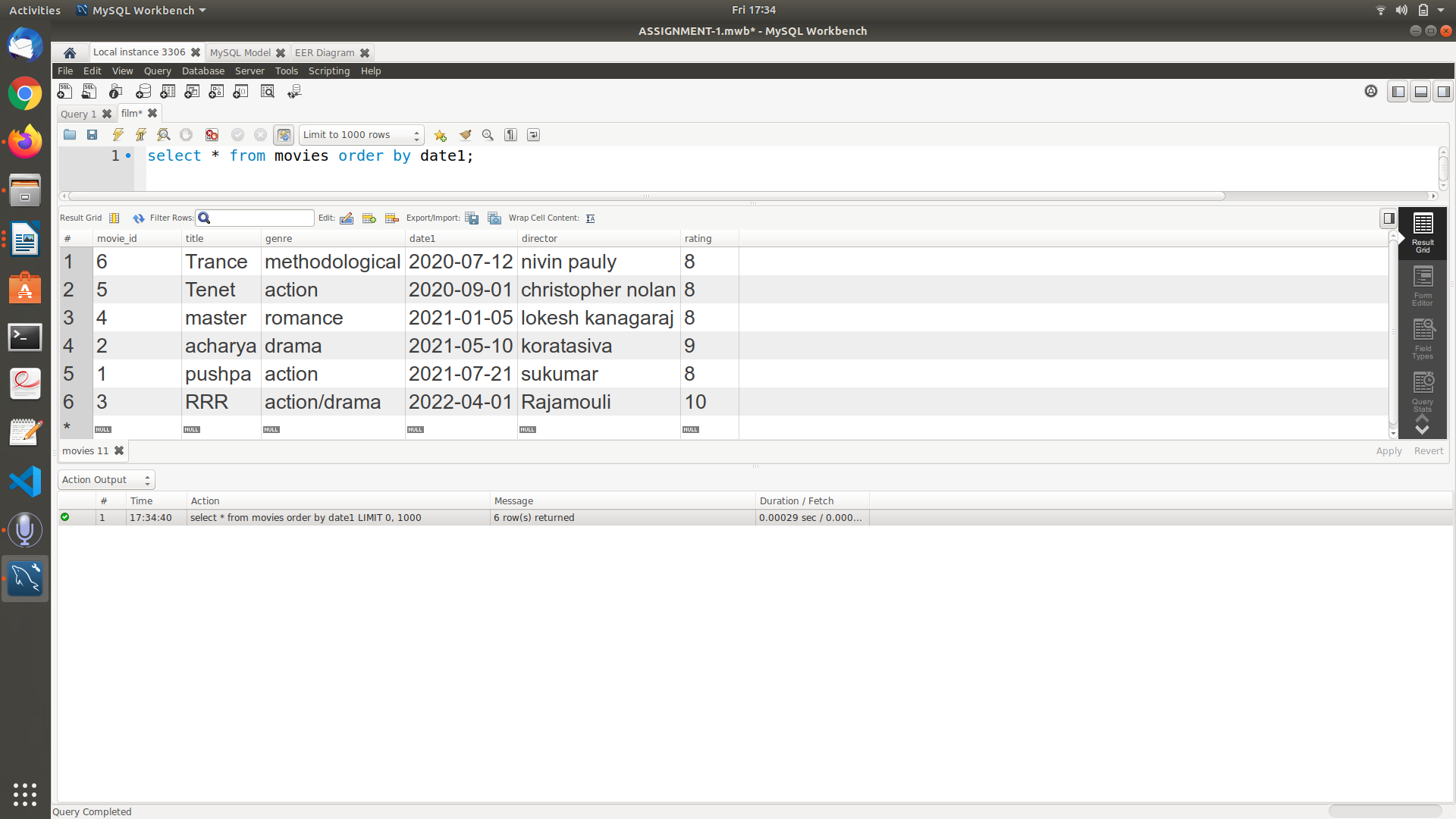
**ASSIGNMENT-4**

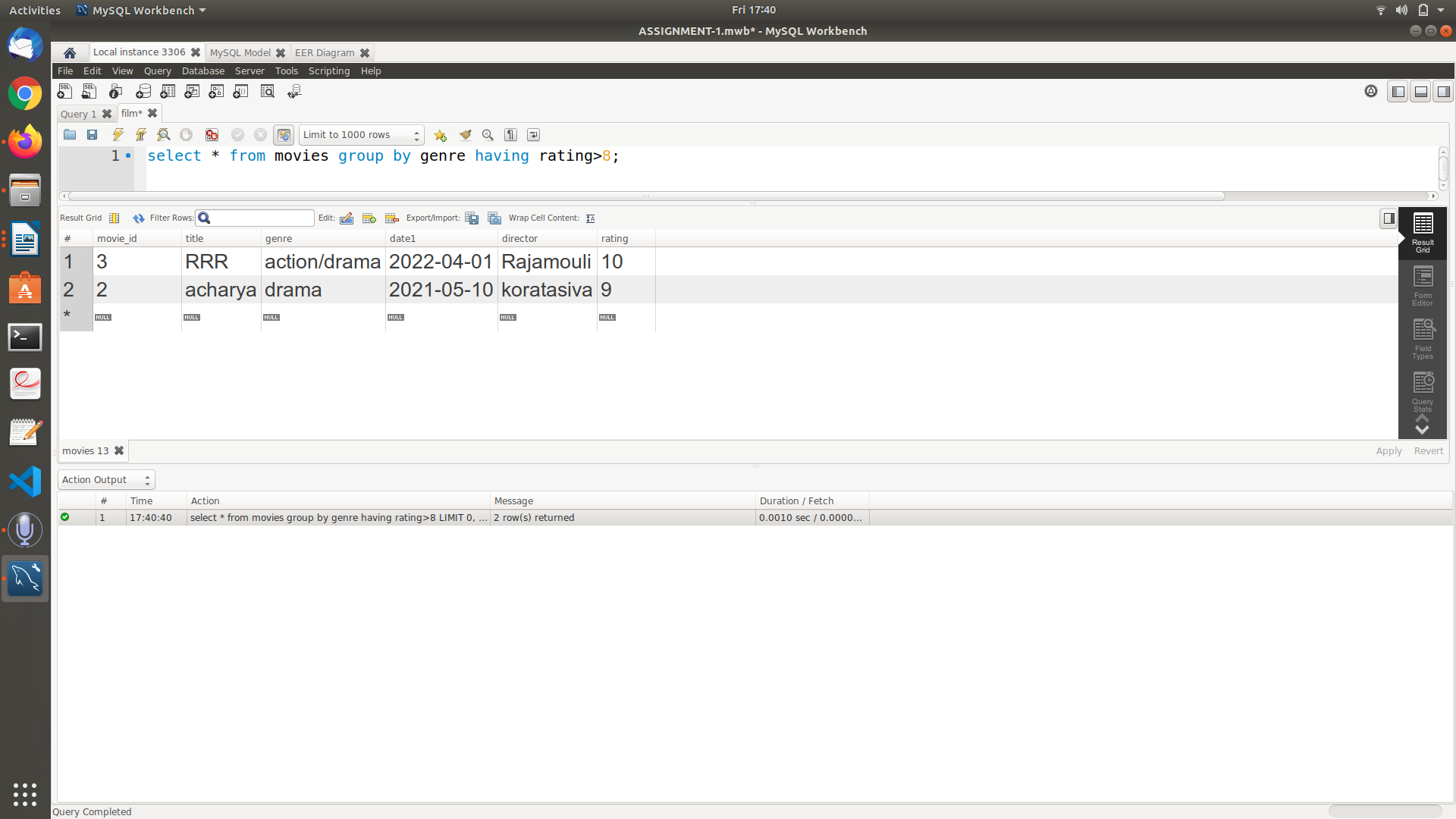
**1. Order by Clause**

**SQL Query** : select \* from movies order by date1;

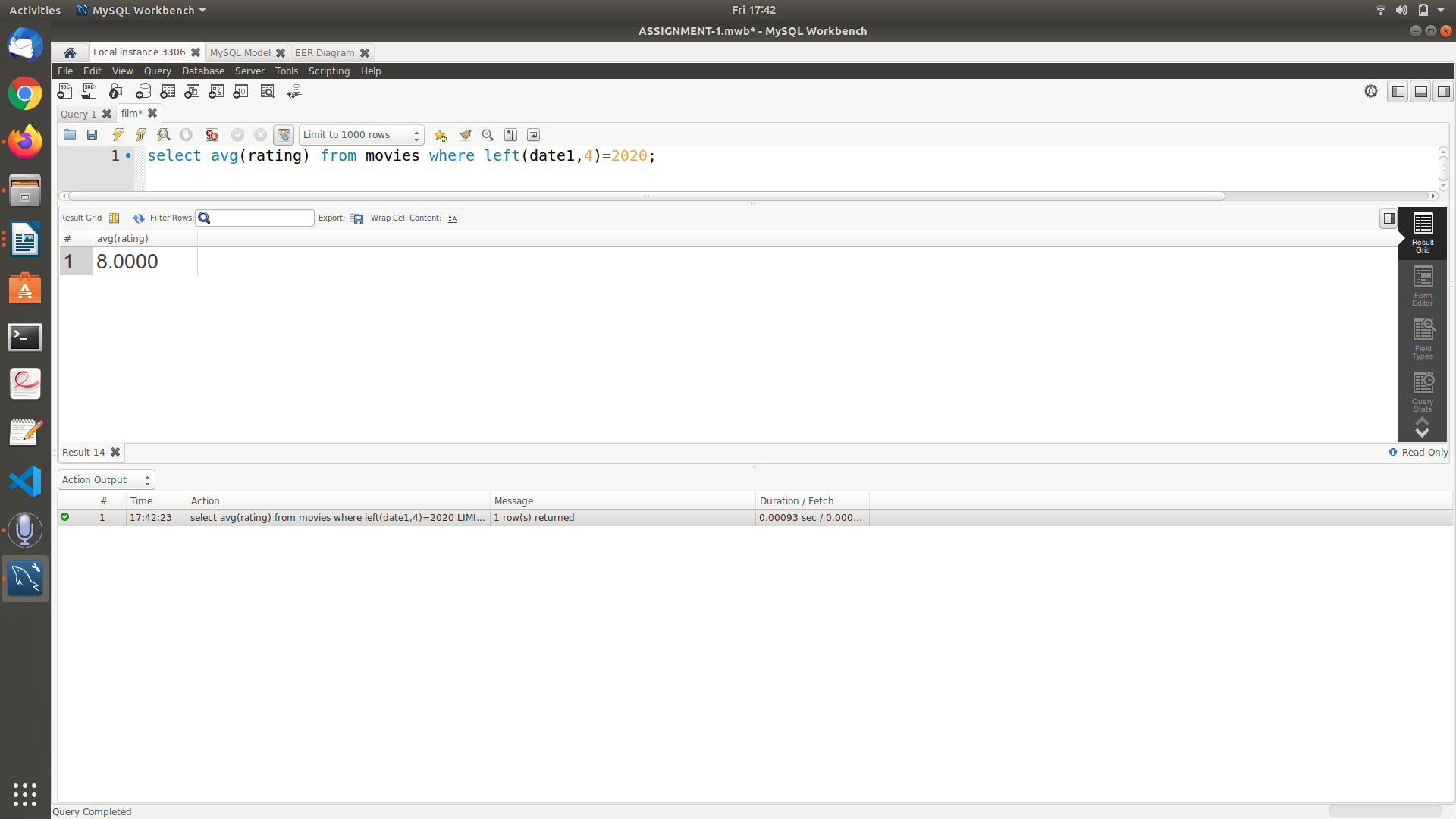
**Output** : 

**2. Group by and having**

**SQL Query** : select \* from movies group by genre having rating>8;

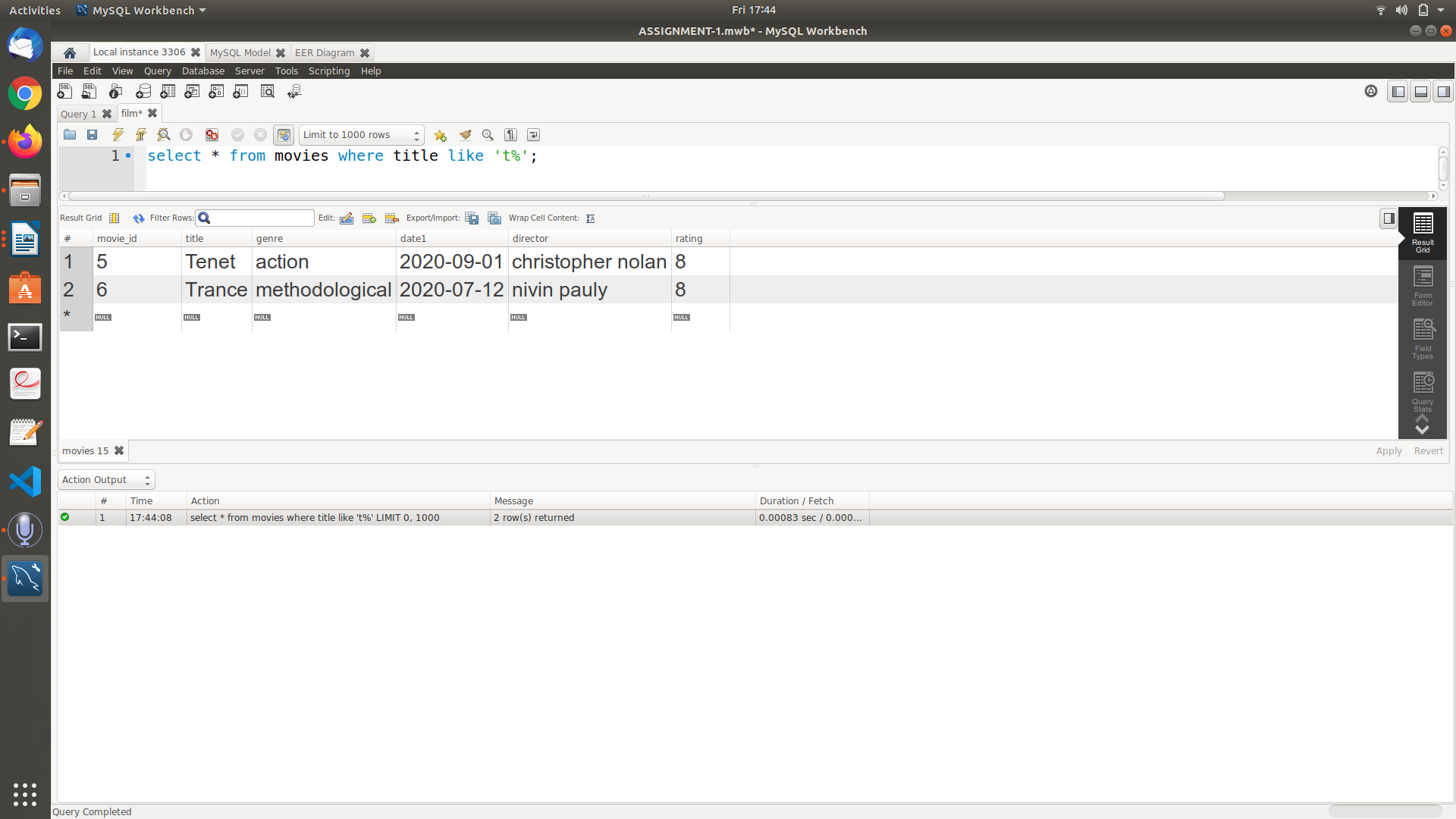
**Output** :

**3. Aggregate functions  
 SQL Query** : select avg(rating) from movies where left(date1,4)=2020;

**Output** :

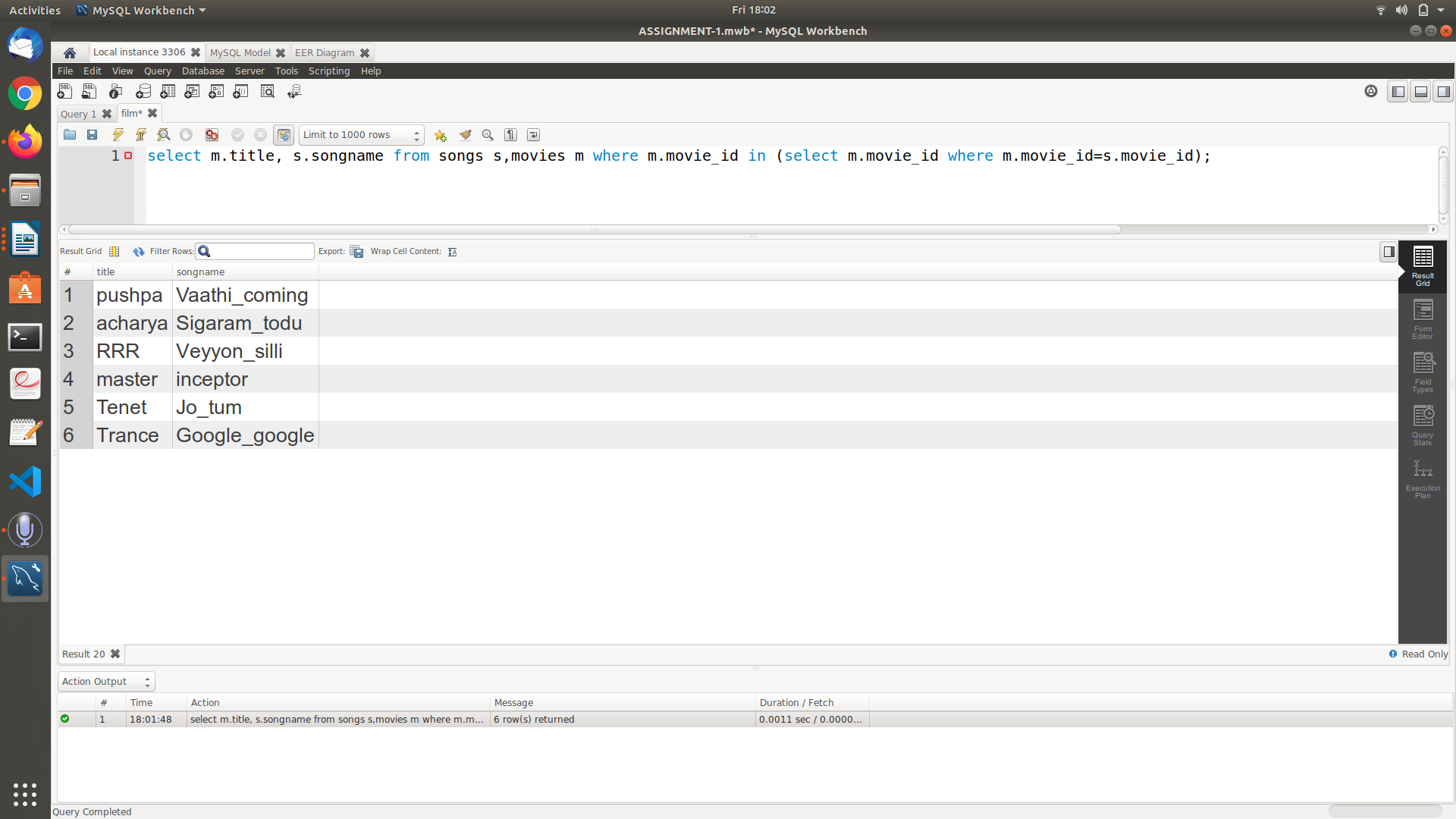
**4. Logical operators especially with LIKE**

**SQL Query** : select \* from movies where title like 't%';

**Output** :

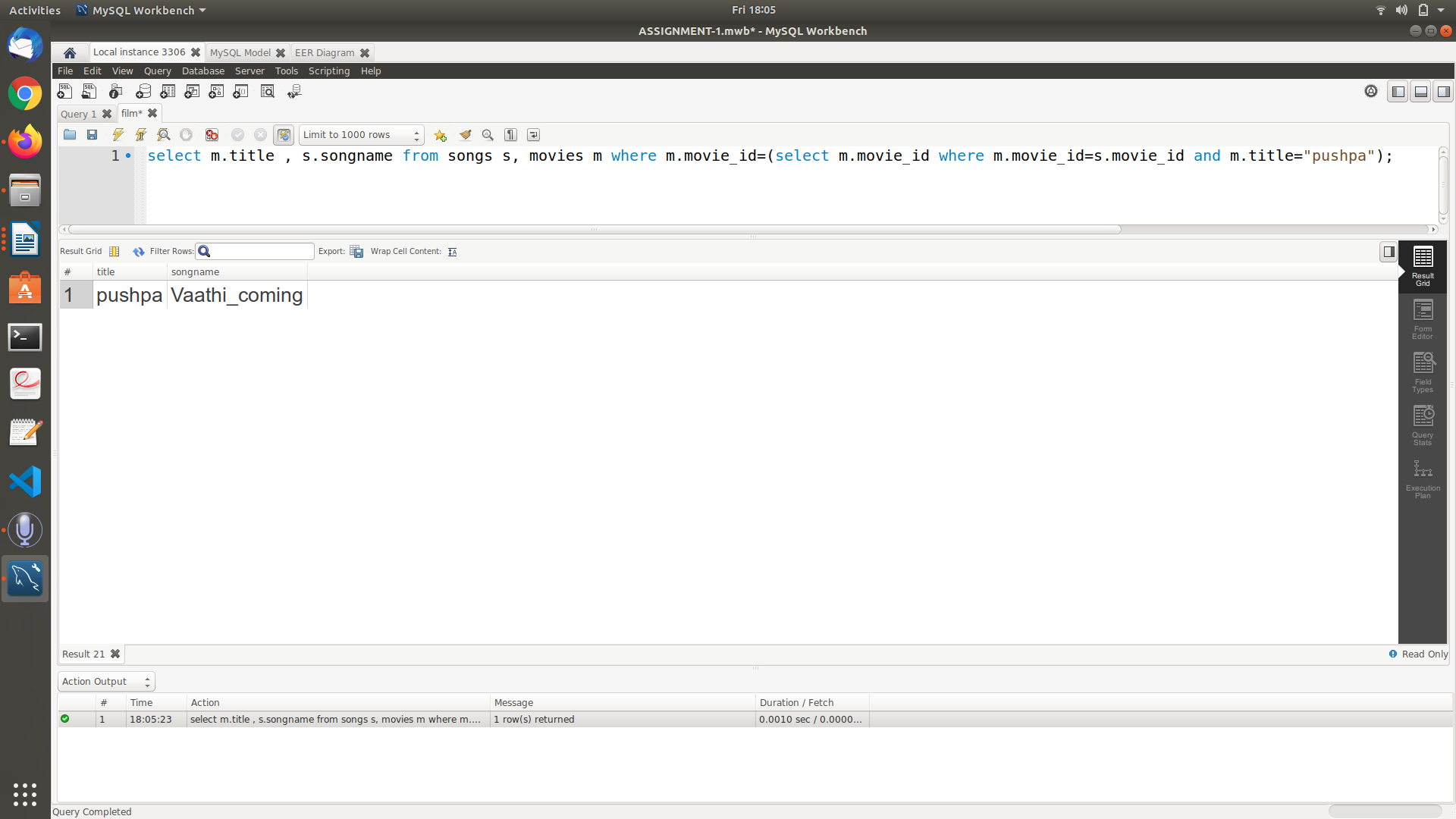
**5. At least 4 Nested queries specific to your Database, out of which at least 2 should have multiple subquery.**

**i)** **SQL Query** :select m.title, s.songname from songs s,movies m where m.movie\_id in (select m.movie\_id where m.movie\_id=s.movie\_id);

**Output**: 

**ii)SQL Query** :

select m.title , s.songname from songs s, movies m where m.movie\_id=(select m.movie\_id where m.movie\_id=s.movie\_id and m.title="pushpa");

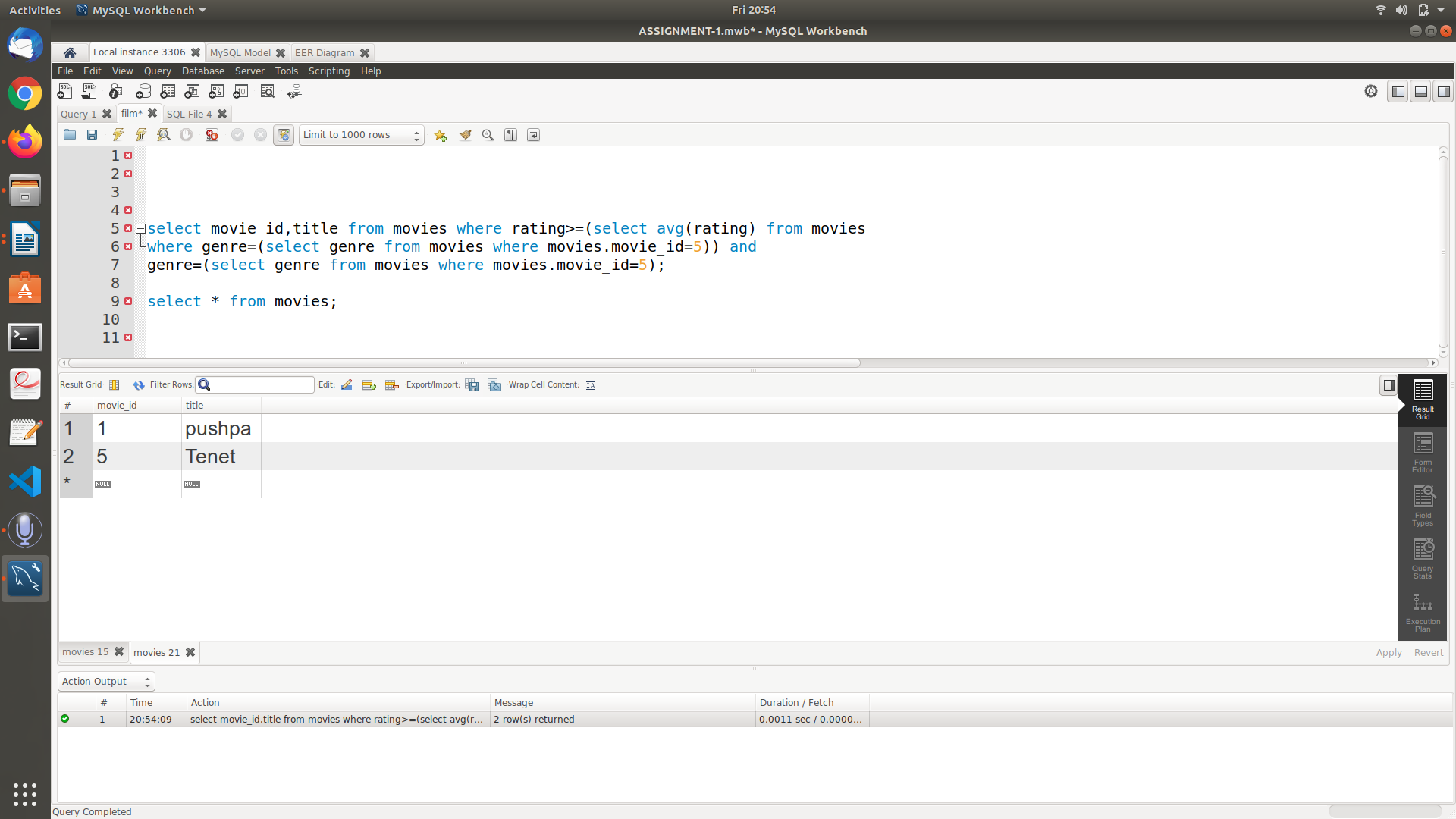
**Output:**

**iii)SQL Query :**

select movie\_id,title from movies where rating>=(select avg(rating) from movies where genre=(select genre from movies where movies.movie\_id=5)) and genre=(select genre from movies where movies.movie\_id=5);

**Explanation :**

It selects the genre of movie\_id=5 and then finds the average of ratings corresponding to that particular genre ,now the outer queries displays the details of movies having rating greater than equal to the value returned by inner query and genre same as the one with movie\_id=5

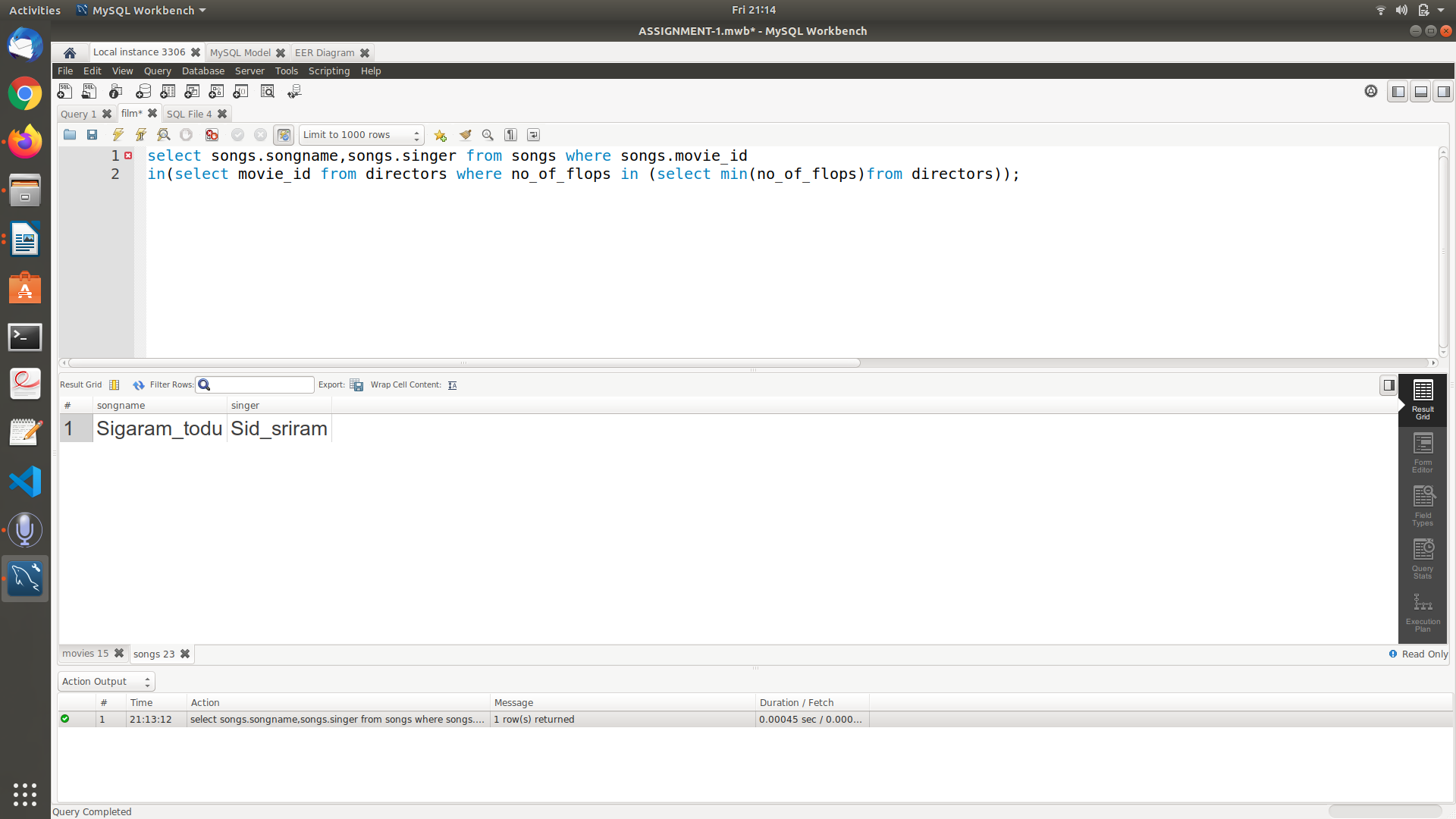
**Output**:

‘**iv)SQL Query**

select songs.songname,songs.singer from songs where songs.movie\_id in(select movie\_id from directors where no\_of\_flops in (select min(no\_of\_flops)from directors));

**Explanation :**

Displays song name and singer from songs table where number\_of\_flops directed by particular director is minimum

**Output :**