use sakila;

-- task 1--

select concat(first\_name, ' ' ,last\_name) as name, avg(amount) as avg\_amount, actor.last\_update from film

join film\_actor using(film\_id)

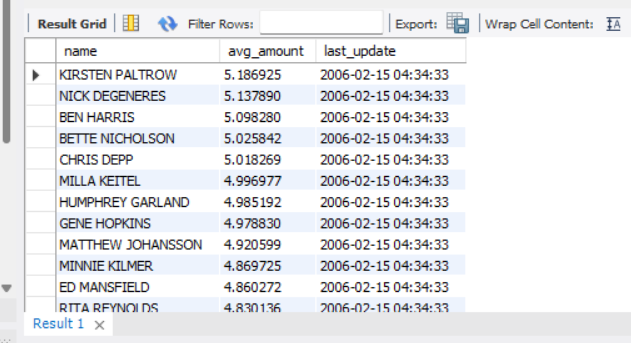
join actor using(actor\_id)

join inventory using(film\_id)

join rental using(inventory\_id)

join payment using(rental\_id)

group by name, actor.last\_update

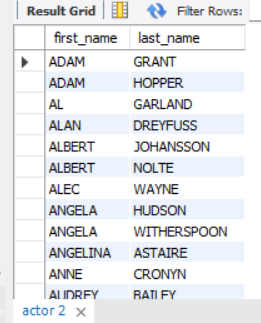
order by 2 desc;

Conclusion:-The board member want to see the list of actor available on profit and last update.

-- task 2.1--

select first\_name, last\_name from actor

order by 1,2;



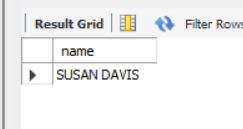
Conclusion:-Change in first\_name and last\_name.

-- task 2.2--

select concat(first\_name, ' ' ,last\_name)as name from actor

group by name

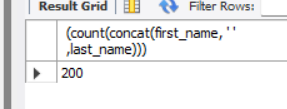
having count(name)>1;



Conclusion:-same first\_name and last\_name of actors.

-- task 2.3--

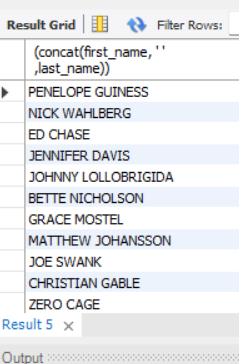
select distinct(count(concat(first\_name, ' ' ,last\_name))) from actor;



Conclusion:-Unique names of actors.

-- task 3--

select distinct(concat(first\_name, ' ' ,last\_name)) from actor;



Conclusion:-The board want to know the list of actors whose names are repeated and who names are not repeated.

-- task 4--

select count(name), first\_name as "actor\_name", name "film\_category" from

film join film\_category using(film\_id)

join category using(category\_id)

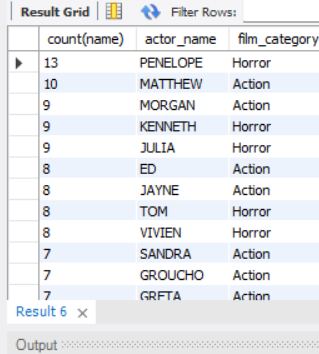
join film\_actor using(film\_id)

join actor using(actor\_id)

where name in ('Action', 'Romance', 'Horror', 'Mystery')

group by name, actor\_name

order by 1 desc;



Conclusion:-Acotrs playing identify roles such as ‘action’, horror’, ‘romance’, and ‘mystery’ the board member want to have a detailed overview of film.

-- task 5--

select movie\_name,

rating\_category,

rating\_description from

(select title movie\_name,

rating rating\_category,

case

when rating like 'G' or rating like 'pg' then 'suitable for kids'

when rating like 'R' then 'restricted for all under 16 unless accompanied by a parent'

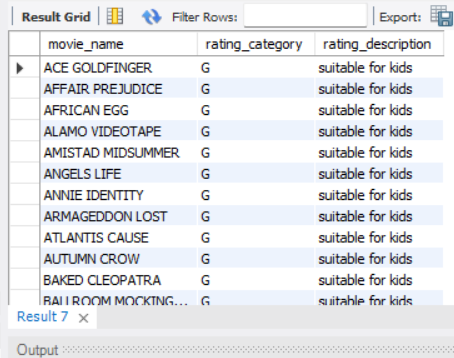
when rating like 'nc-17' then 'restricted for all audiences under 18'

end rating\_description

from film ) exp

where rating\_description is not null

order by rating\_category;



Conclusion:-The board want to know various rating categories with description.

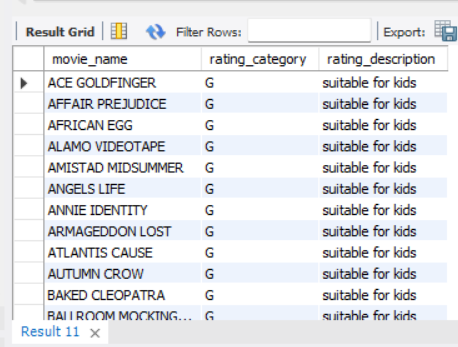
-- task 6.1--

select title movie\_name,

replacement\_cost

from film

where replacement\_cost <=9;

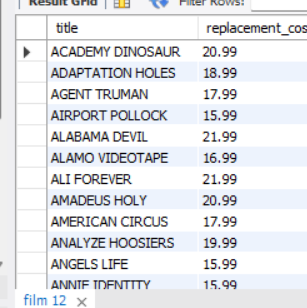


Conclusion:-The board members want to know the replacement cost <=9.

-- task 6.2--

select title, replacement\_cost from film

where replacement\_cost between 15 and 22;



Conclusion:- The board members want to know the replacement cost between 15 and 20.

-- task 6.3--

select title ,

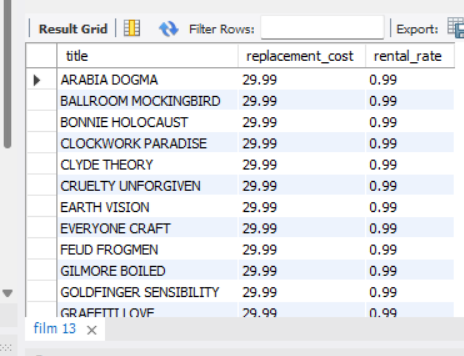
replacement\_cost,

rental\_rate

from film

where rental\_rate =(select min(rental\_rate) from film)

and replacement\_cost=(select max(replacement\_cost) from film);



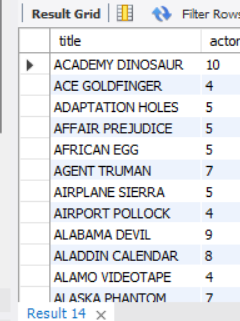
Conclusion:-The movie with the highest replacement cost and lowest rental cost.

-- task 7--

select title, count(\*) as "actor" from film join film\_actor

using(film\_id)

group by title;



Conclusion:-The board member want to know the list all the films and no.of actors.

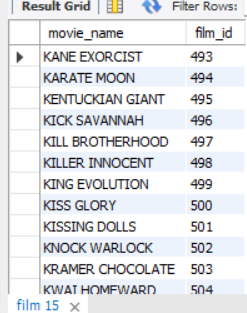
-- task 8--

select title movie\_name, film\_id

from film

where title like 'K%'

or title like 'q%';



Conclusion:-The film starting with the letters ‘K’ and ‘Q’ have also soared in popularity.

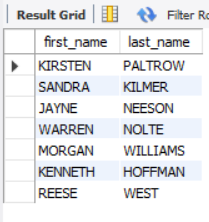
-- task 9--

select first\_name, last\_name

from film join film\_actor on film.film\_id= film\_actor.film\_id

join actor on actor.actor\_id=film\_actor.actor\_id

where title = 'agent truman';



Conclusion:- AGENT TRUMAN is a great success and all actors who appeared in the films.

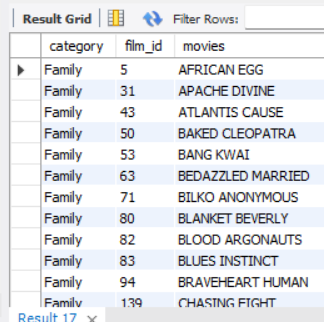
-- task 10--

select name as category, film.film\_id, title as movies

from category join film\_category on category.category\_id=film\_category.category\_id

join film on film.film\_id=film\_category.film\_id

where name = 'family';



Conclusion:-The board member want to promote all family movies.

-- task 11--

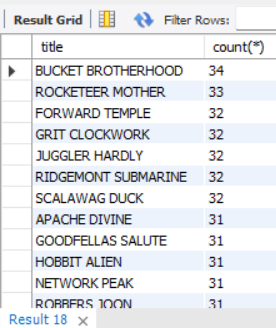
select title, count(\*) from film

join inventory using(film\_id)

join rental using(inventory\_id)

group by title

order by 2 desc;



Conclusion:-The most frequently rented movie in descending order and more copies of movies.

-- task 12--

select category\_name,

difference from

(select name category\_name,

avg(replacement\_cost) - avg(rental\_rate) difference

from category a

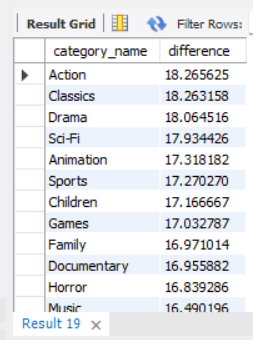
inner join film\_category b on a.category\_id=b.category\_id

inner join film c on b.film\_id=c.film\_id

group by category\_name)exp

where difference >15

order by difference desc;



Conclusion:-The average difference btw the film replacement cost and rental rate is greater than 15.

-- task 13--

select category\_name,

number\_of\_films from

(select name category\_name,

count(title) number\_of\_films

from category a

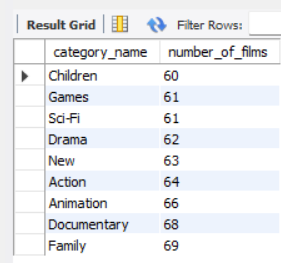
inner join film\_category b on a.category\_id=b.category\_id

inner join film c on c.film\_id=b.film\_id

group by name) exp

where number\_of\_films between 60 and 71

order by number\_of\_films;



Conclusion:-Board member want to identify the genre s having 60-70 films.