《Introduction to Database System A*》 Experiment report 3

contents：Full query in SQL View	lab：10-409 Instructor:liuli
Student ID:2023337621159	Student Name:代翔

Software Environment ：

Windows/linux/macOS + Postgresql/Mysql + GUI(pgadmin 4)

- Purpose：

(1)The exercise is to have a taste on full query

To understand group,having clause and aggregation functions.

To order the result

(2)To understand database view; to practice on constructing/use on the view

- Guidance for the laboratory exercise：

Notation1： full query(with group by/having/order by clauses and aggregators)

SELECT [DISTINCT] A_1[AS ALIAS1], A_2[as ALIAS2]…
FROM R
[WHERE condtion]
[GROUP BY A_i,A_j]
[HAVING condition]
[ORDER BY attribute list ASC|DESC];

- Tips for SQL statement:

(1).A_1,A_n could be reasonable expressions involving attribute, constant or function

(2).Condition is to restrict the tuples in the working relations.

(3).SELECT could be followed by aggregation such as avg(An)、count(An)、sum(An)、max(An)、min(An) to do calculation within the table or group(if there is" group by" clause).

(4).DISTINCT: eliminate the duplicated tuples

(5).GROUP BY clause is used to group the table，producing one tuple for each group. Usually, a HAVING clause will follow the GROUP BY clause to restrict the group satisfying the condition within Having clause.

e.g.

```
SELECT DISTINCT sp.pid
FROM sp
GROUP BY pid
HAVING count(sid)>1;
```

(6).If there is an ORDER BY clause in the statement， reorder the tuples according to the attribute list in the ORDER BY clause in ascending or descending order.

Notation2：view

- to define a view

CREATE VIEW <VIEW_name> AS

Select statement

- Eliminate view

DROP VIEW <VIEW_name>

contents ():

(1)Write SQL statements and execute each statement for the following queries(questions);

(2) After you have successfully execute the statement, copy this statement to the document ; paste the snapshot picture for each execution result of the SQL statement in the document; Handing your answer file using the file name as exp3 _ID(ID should be replaced by your own student ID);

Experiment background:

we have set up a database “my_movies_database” with the relations schema below:

Movies(title,year,length,movietype, studioname, producerC)

movieStar(name,address,gender,birthdate)

starsIn(movietitle,movieyear,starname)

Studio(name,address,presC)

Movieexec(name,address,cert,networ

Example tasks and answers:

Using relation movies, Find each studio who has the given name and whose movie numbers no less than two,return their name and number of movie for the studio,then sort the result in a descending order on attribute studioname.

SQL ANSWER:

```
Select studioname as name,count(*) as movienum
from movies
where studioname is not null
group by studioname
having count(*)>=2
order by studioname desc
```

RESULTING PICTURE:

Experiment tasks:

Query to database in SQL statement:

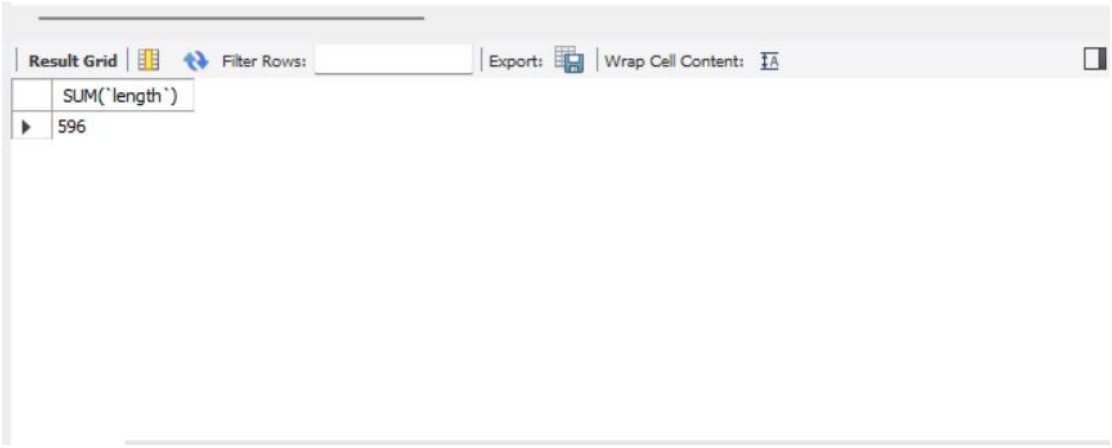
write select statement in SQL to answer the following questions

Part 1. Aggregate:

1) Find the total movie length by 'MGM';

SQL ANSWER:
```
SELECT SUM(`length`)
FROM `movies`
WHERE `studioname` = 'MGM';
```

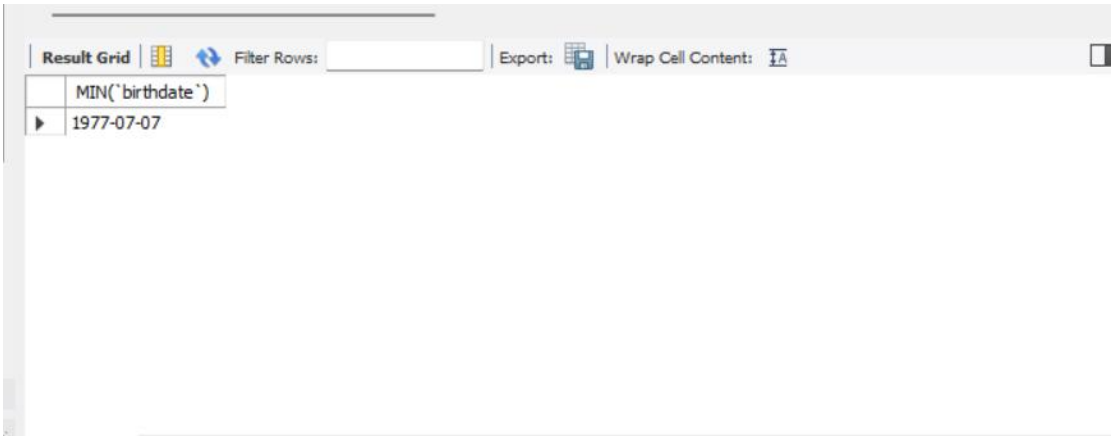
RESULTING PICTURE:



2) Find the youngest female star, return her birthdate value;

SQL ANSWER:
```
SELECT MIN(`birthdate`)
FROM `moviestar`
WHERE `gender` = 'F';
```

RESULTING PICTURE:

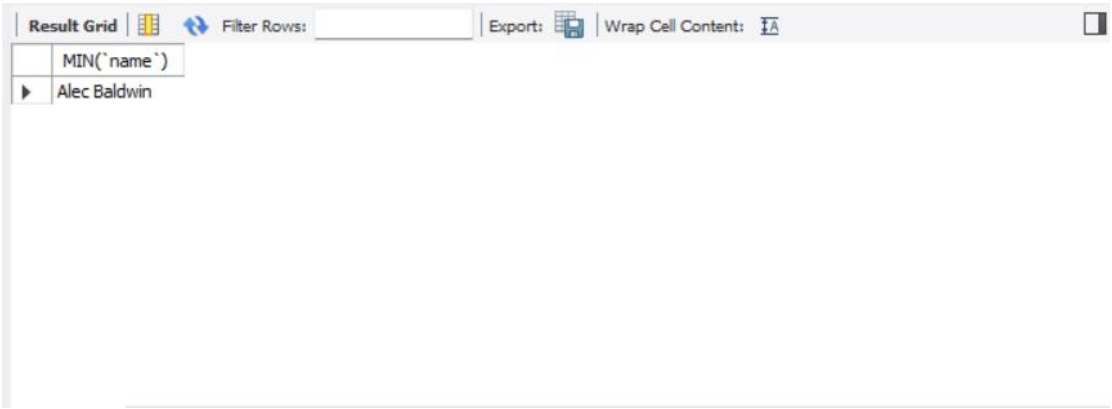


3) Find the minmum name value for the male star;

SQL ANSWER:
```
SELECT MIN(`name`)
```

FROM `moviestar`
WHERE `gender` = 'M';

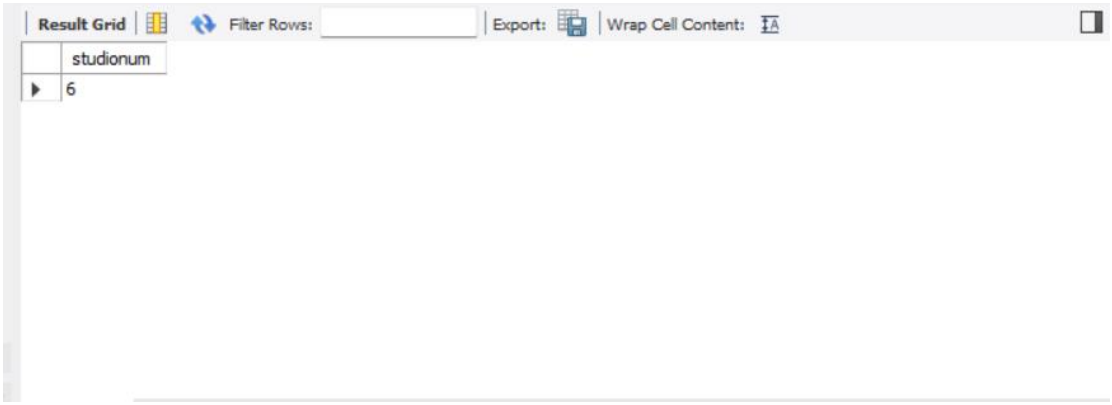
RESULTING PICTURE:



4) on relation studio, Return the number of studios, with the returning expressional title as studionum;

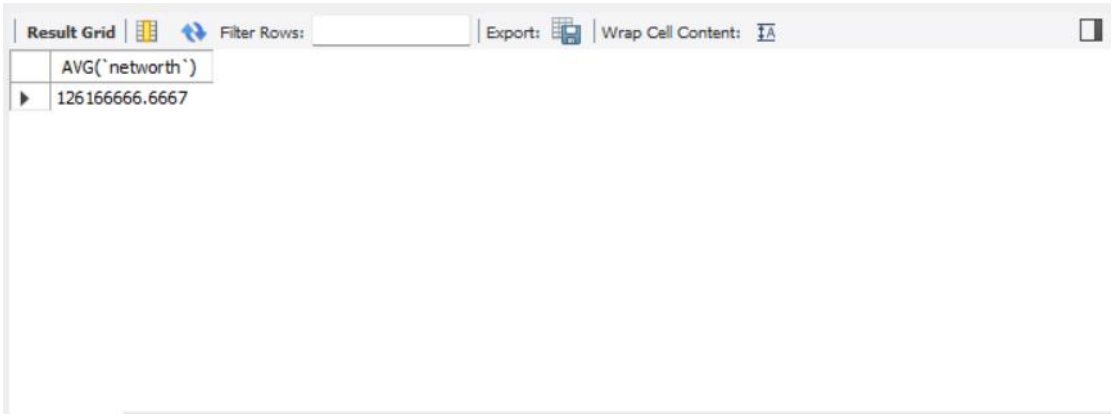
SQL ANSWER: SELECT COUNT(*) AS `studionum`
FROM `studio`;

RESULTING PICTURE:



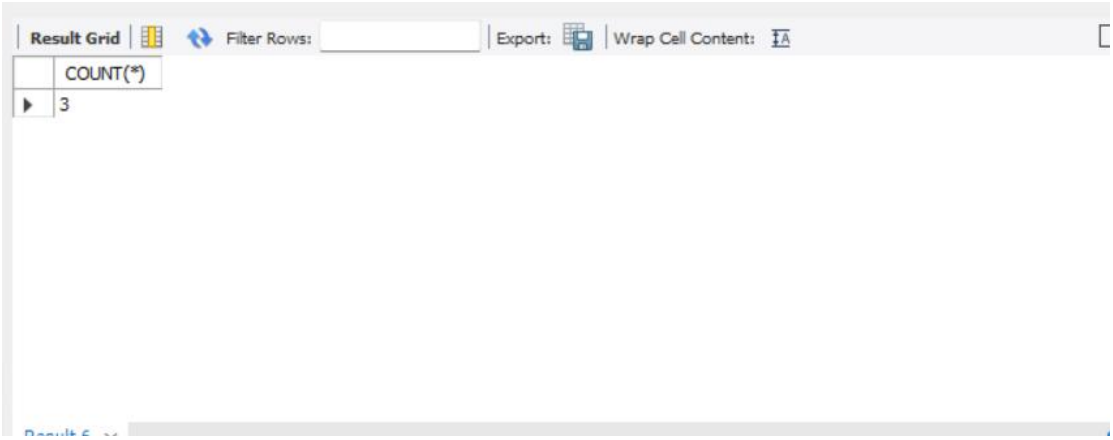
5) Find the average networth value for movie executives;

SQL ANSWER:SELECT AVG(`networth`)
FROM `movieexec`;



6) Find the number of specific movie stars for movie Star Wars in 1977;

SQL ANSWER:SELECT COUNT(*)
FROM `starsin`
WHERE `movietitle`='Star Wars';


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

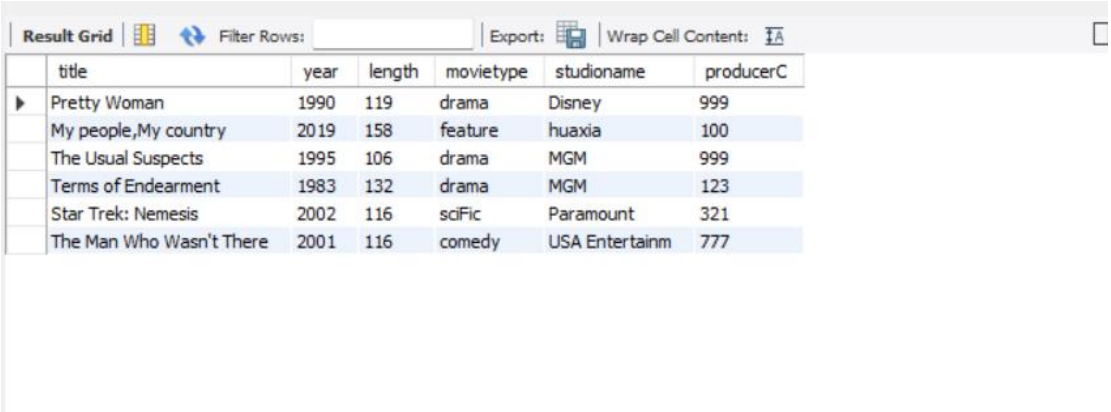
COUNT(*)
3

Part 2. Order clause,Group and having clause:

1. Return movies after 1980, with the ascending order by studioname, and producerC in a reverse order for the same studioname;

SQL ANSWER: SELECT *
FROM `movies`
WHERE `year`>1980
ORDER BY `studioname` ASC,`producerC` DESC;

RESULTING PICTURE:

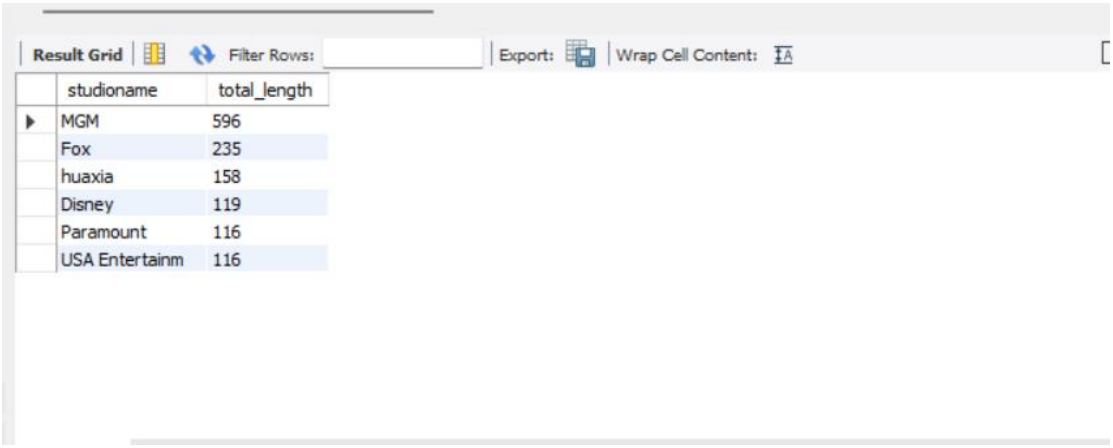

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	year	length	movietype	studioname	producerC
Pretty Woman	1990	119	drama	Disney	999
My people,My country	2019	158	feature	huaxia	100
The Usual Suspects	1995	106	drama	MGM	999
Terms of Endearment	1983	132	drama	MGM	123
Star Trek: Nemesis	2002	116	sciFic	Paramount	321
The Man Who Wasn't There	2001	116	comedy	USA Entertainm	777

2. Find the total length of movies for each studio(not include unknown studio) , return studio(name) and total_length,reorder the result by total length in descending order;

SQL ANSWER: SELECT `studioname`,SUM(`length`) AS `total_length`
FROM `movies`
WHERE `studioname` IS NOT NULL
GROUP BY `studioname`
ORDER BY `total_length` DESC;

RESULTING PICTURE:

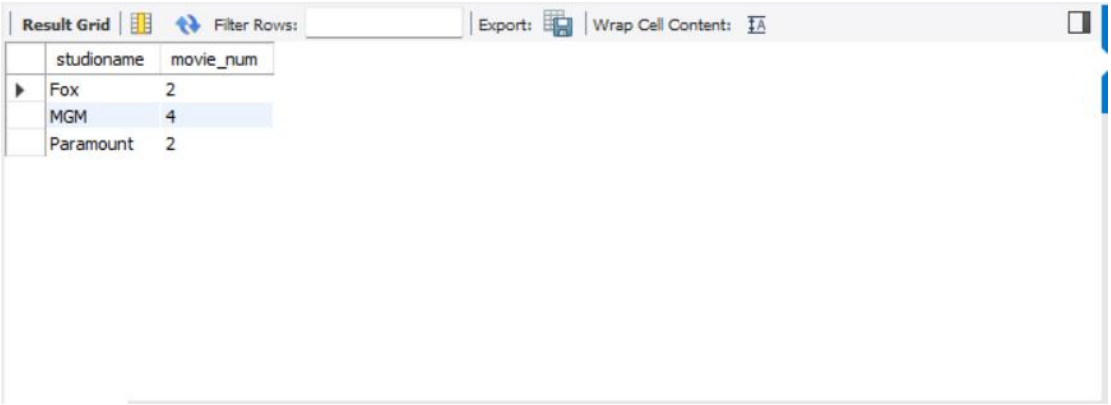


studioname	total_length
MGM	596
Fox	235
huaxia	158
Disney	119
Paramount	116
USA Entertainm	116

3. Use relation movies, find those studios with total movie number no less than 2.return studioname and movie number;

SQL ANSWER:
```
SELECT `studioname`,COUNT(*) AS `movie_num`
FROM `movies`
GROUP BY `studioname`
HAVING COUNT(*) >= 2;
```

RESULTING PICTURE:

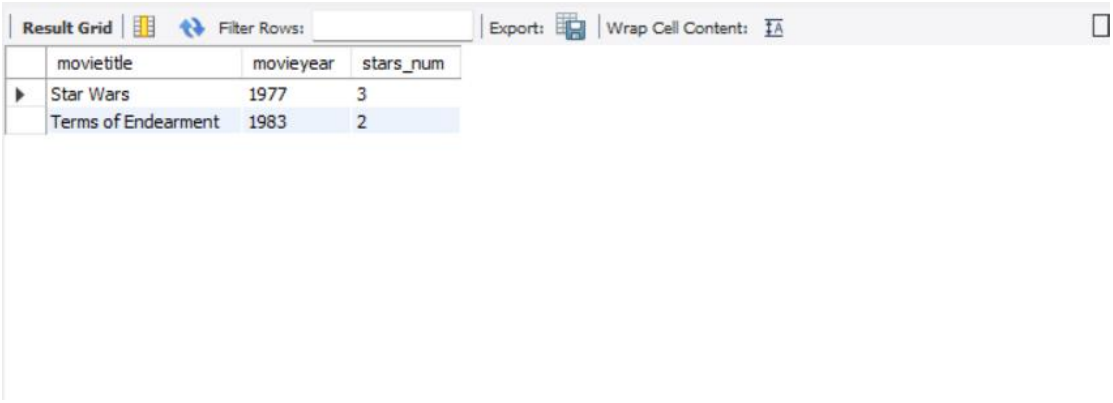


studioname	movie_num
Fox	2
MGM	4
Paramount	2

4. For each movie that has **more than one** star, calculate the number of stars for each movie,and return the title ,year and the number of stars in each movie;

SQL ANSWER:
```
SELECT `movietitle`,`movieyear`,COUNT(`starname`) AS `stars_num`
FROM `starsin`
GROUP BY `movietitle`,`movieyear`
HAVING COUNT(`starname`) > 1;
```

RESULTING PICTURE:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

movietitle	movieyear	stars_num
Star Wars	1977	3
Terms of Endearment	1983	2

5. For the stars in relation moviestar, devide movie stars by gender, then for each gender, return gender and number of member for each gender, order by number in descending order;

SQL ANSWER:
```
SELECT `gender`,COUNT(`name`) AS `gender_num`
FROM `moviestar`
GROUP BY `gender`
ORDER BY `gender_num` DESC;
```

RESULTING PICTURE:

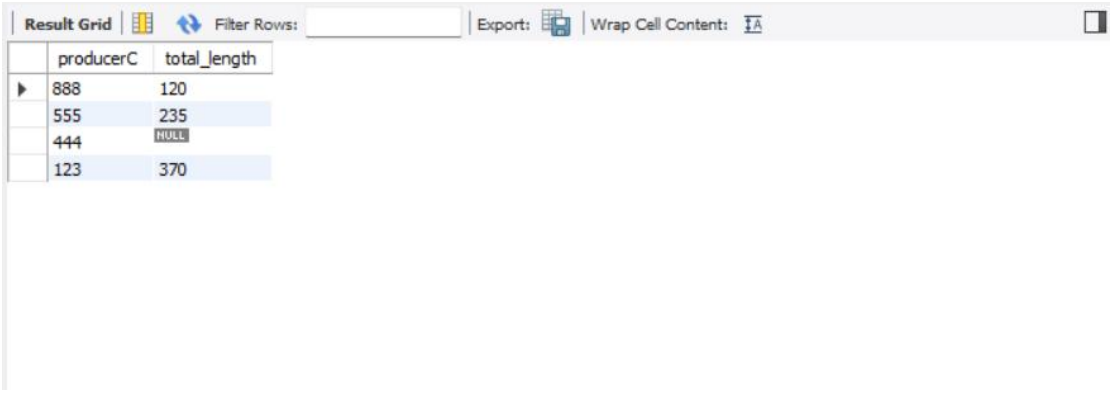
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

gender	gender_num
M	3
F	3

6. print the producers' number and producers' total film length for only those producers who has the given producer number in table movies and made at least 1 film prior to 1980;sort by producer number in the result ;

SQL ANSWER:
```
SELECT `producerC`,SUM(`length`) AS `total_length`
FROM `movies`
WHERE `producerC` IS NOT NULL
GROUP BY `producerC`
HAVING COUNT(CASE WHEN `year`<1980 THEN 1 END) >= 1
ORDER BY `producerC` DESC;
```

RESULTING PICTURE:



Part 3. Define and use of View

Write SQL command to define the view based on the tables in the database ; then query or modify on view. (After you have successfully execute each statement, copy this statement to this document ; paste the snapshot picture for the **query result througn foxview** in the document if necessary);

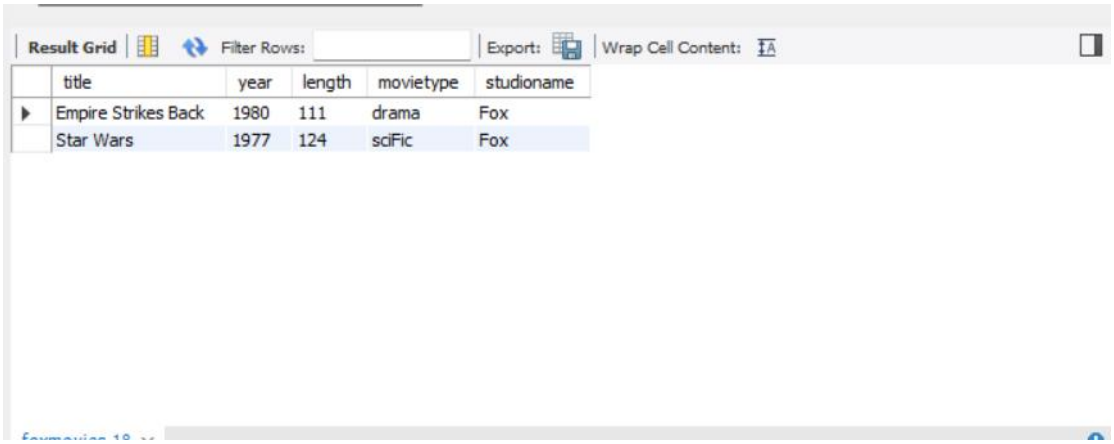
1. Create a view foxmovies giving the title,year,length,movietype,studioname((the view involve the movies produced by Fox studio));

   SQL ANSWER:

```
CREATE VIEW `foxmovies` AS
SELECT `title`,`year`,`length`,`movietype`,`studioname`
FROM `movies`
WHERE `studioname`='Fox';

SELECT * FROM `foxmovies`;
```

   RESULTING PICTURE



2. Find movies of sciFic type by FOX studio from foxmovies view;

SQL ANSWER:

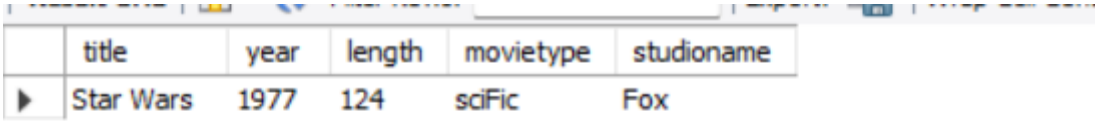
SELECT *

FROM `foxmovies`

WHERE `movietype`='sciFic';

RESULTING

PICTURE



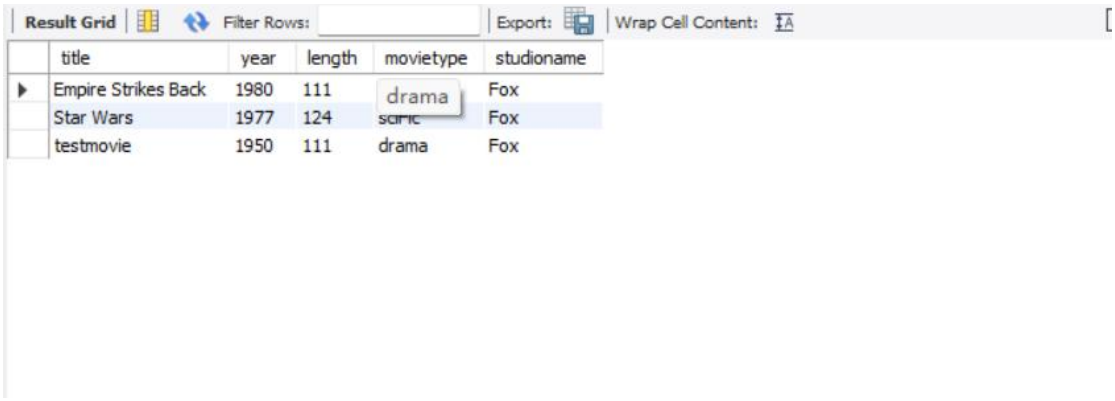
title	year	length	movietype	studioname
Star Wars	1977	124	sciFic	Fox

3. Add a new tuple(“testmovie”,1950,111,”drama”,”Fox”) through the view foxmovies;

SQL ANSWER: INSERT `foxmovies` VALUES ('testmovie',1950,111,'drama','Fox');

RESULTING

PICTURE



4. Update the year value of the movie which was produced in 1950 to 1951 through the view Foxmovies;

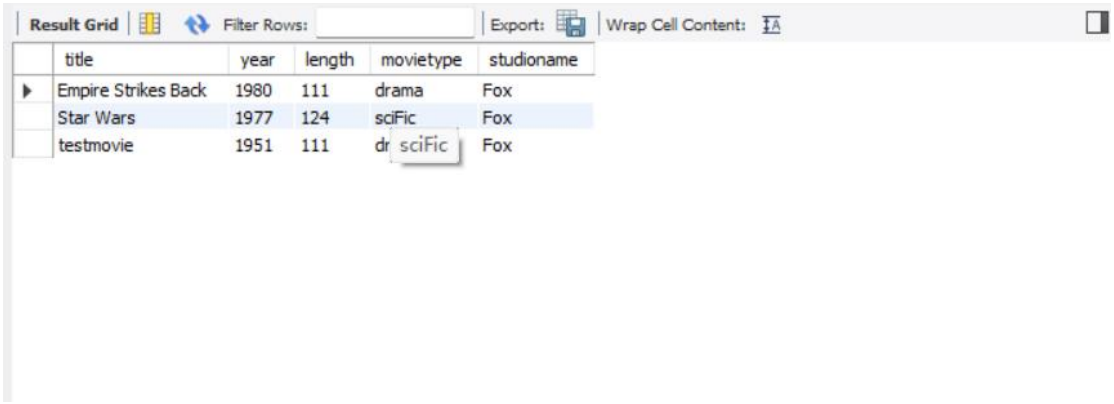
SQL ANSWER: **UPDATE `foxmovies`**

SET `year`=1951

WHERE `year`=1950;

RESULTING

PICTURE



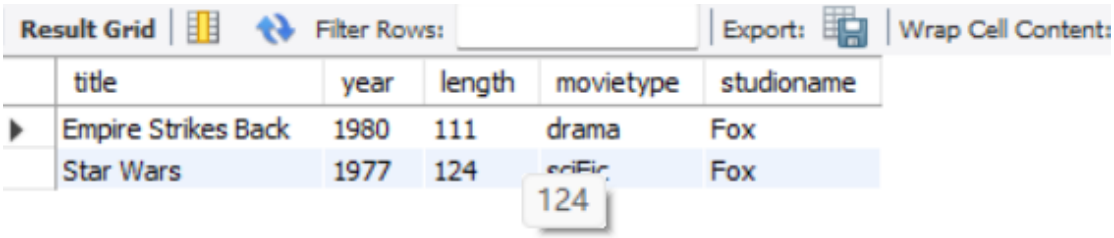
5. delete a new tuple which was produced in 1951 through the view foxmovies;

SQL ANSWER: DELETE

FROM `foxmovies`

WHERE `year`=1951;

RESULTING PICTURE



notice:

.Handing your answer **PDF** file using the file name as exp3_ID(ID should be replaced by your own student ID);

The electronic version of your document should be handed in **before deadline(within 1 week).**