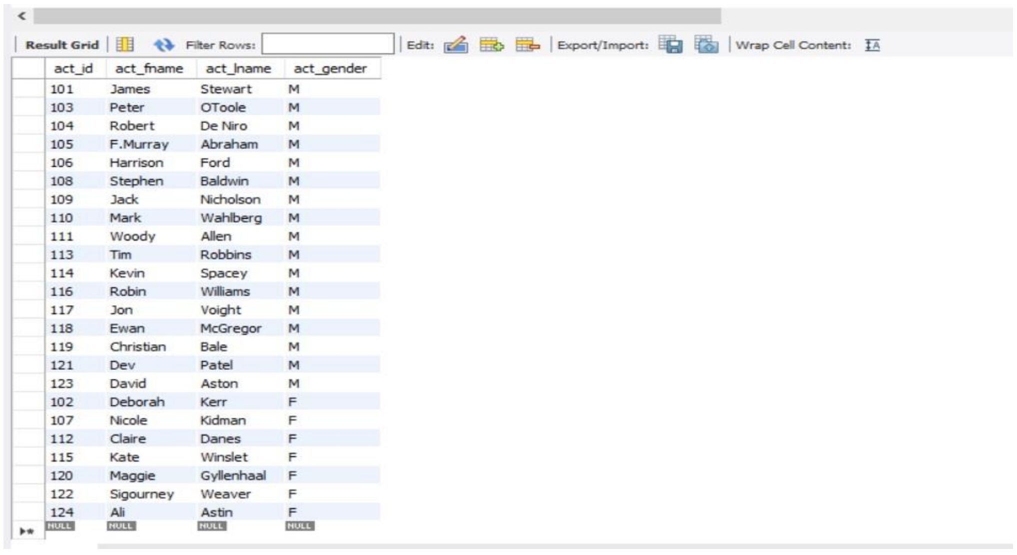
1.Create Database project_movie_data and deploy table data according to ER diagram.

```
        create database
project_movie_data; use
project_movie_data; show tables;
```

2. Create 9 tables which are presented in ER model and load the data with their foreign key and primary key values.

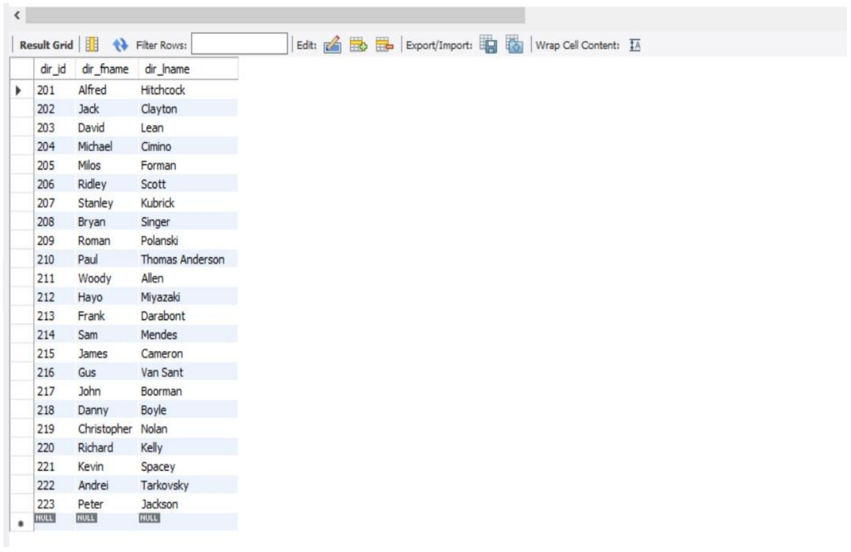
```
 create table actor( act_id
int primary key, act_fname
char(20), act_lname
char(20), act_gender
char(1));
 insert into actor(act_id,act_fname,act_lname,act_gender)
values(101,'James','Stewart','M'),(102,'Deborah','Kerr','F'),
(103,'Peter','OToole','M'),(104,'Robert','De Niro','M'),(105,'F.Murray','Abraham','M'),
(106,'Harrison','Ford','M'),(107,'Nicole','Kidman','F'),(108,'Stephen','Baldwin ','M'),
(109,'Jack','Nicholson','M'),(110,'Mark','Wahlberg','M'),(111,'Woody','Allen',' M'),
(112,'Claire','Danes','F'),(113,'Tim','Robbins','M'),(114,'Kevin','Spacey','M') ,
(115,'Kate','Winslet','F'),(116,'Robin','Williams','M'),(117,'Jon','Voight','M'
),
(118,'Ewan','McGregor','M'),(119,'Christian','Bale','M'),(120,'Maggie','Gyllenh aal','F'),
(121,'Dev','Patel','M'),(122,'Sigourney','Weaver','F'),(123,'David','Aston','M' ),
(124,'Ali','Astin','F');
 select*from actor;
```


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

act_id	act_fname	act_lname	act_gender
101	James	Stewart	M
103	Peter	OToole	M
104	Robert	De Niro	M
105	F.Murray	Abraham	M
106	Harrison	Ford	M
108	Stephen	Baldwin	M
109	Jack	Nicholson	M
110	Mark	Wahlberg	M
111	Woody	Allen	M
113	Tim	Robbins	M
114	Kevin	Spacey	M
116	Robin	Williams	M
117	Jon	Voight	M
118	Ewan	McGregor	M
119	Christian	Bale	M
121	Dev	Patel	M
123	David	Aston	M
102	Deborah	Kerr	F
107	Nicole	Kidman	F
112	Claire	Danes	F
115	Kate	Winslet	F
120	Maggie	Gyllenhaal	F
122	Sigourney	Weaver	F
124	Ali	Astin	F
NULL	NULL	NULL	NULL

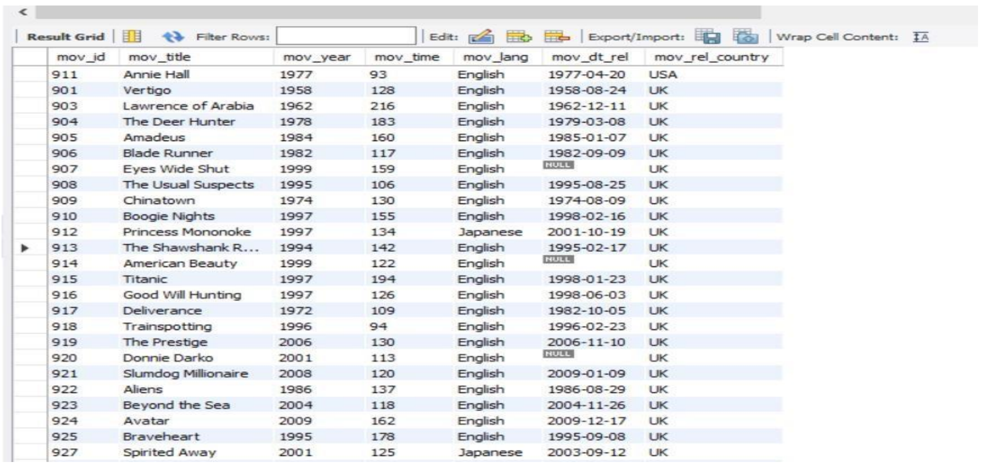
```
create table director( dir_id int primary key, dir_fname char(20), dir_lname char(20));
 insert into director(dir_id,dir_fname,dir_lname)
```

```
values(201,'Alfred','Hitchcock'),(202,'Jack','Clayton'),(203,'David','Lean'),(2
04,'Michael','Cimino'),
(205,'Milos','Forman'),(206,'Ridley','Scott'),(207,'Stanley','Kubrick'),
(208,'Bryan','Singer'),(209,'Roman','Polanski'),(210,'Paul','Thomas Anderson'),
(211,'Woody','Allen'),(212,'Hayo','Miyazaki'),(213,'Frank','Darabont'),
(214,'Sam','Mendes'),(215,'James','Cameron'),(216,'Gus','Van Sant'),
(217,'John','Boorman'),(218,'Danny','Boyle'),(219,'Christopher','Nolan'),
(220,'Richard','Kelly'),(221,'Kevin','Spacey'),(222,'Andrei','Tarkovsky'),
(223,'Peter','Jackson');
 select*from director;
```



dir_id	dir_fname	dir_lname
201	Alfred	Hitchcock
202	Jack	Clayton
203	David	Lean
204	Michael	Cimino
205	Milos	Forman
206	Ridley	Scott
207	Stanley	Kubrick
208	Bryan	Singer
209	Roman	Polanski
210	Paul	Thomas Anderson
211	Woody	Allen
212	Hayo	Miyazaki
213	Frank	Darabont
214	Sam	Mendes
215	James	Cameron
216	Gus	Van Sant
217	John	Boorman
218	Danny	Boyle
219	Christopher	Nolan
220	Richard	Kelly
221	Kevin	Spacey
222	Andrei	Tarkovsky
223	Peter	Jackson
NULL	NULL	NULL

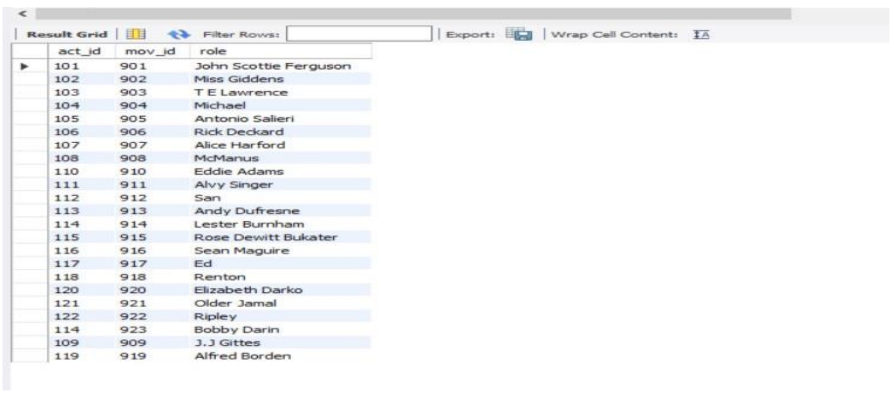
```
create table movie( mov_id int primary key, mov_title char(50), mov_year int, mov_time int,
mov_lang char(50), mov_dt_rel date, mov_rel_country char(5));


insert into movie(mov_id,mov_title,mov_year,mov_time,mov_lang,mov_dt_rel,mov_rel_country)
values(901,'Vertigo',1958,128,'English','1958-08-24','UK'), (902,'The
Innocents',1961,100,'English','1962-02-19','SW'),
(903,'Lawrence of Arabia',1962,216,'English','1962-12-11','UK'),
(904,'The Deer Hunter',1978,183,'English','1979-03-08','UK'),
(905,'Amadeus',1984,160,'English','1985-01-07','UK'),
(906,'Blade Runner',1982,117,'English','1982-09-09','UK'),
(907,'Eyes Wide Shut',1999,159,'English',null,'UK'),
(908,'The Usual Suspects',1995,106,'English','1995-08-25','UK'),
(909,'Chinatown',1974,130,'English','1974-08-09','UK'),
(910,'Boogie Nights',1997,155,'English','1998-02-16','UK'),
(911,'Annie Hall',1977,93,'English','1977-04-20','USA'),
(912,'Princess Mononoke',1997,134,'Japanese','2001-10-19','UK'),
(913,'The Shawshank Redemption',1994,142,'English','1995-02-17','UK'),
(914,'American Beauty',1999,122,'English',null,'UK'),
(915,'Titanic',1997,194,'English','1998-01-23','UK'),
(916,'Good Will Hunting',1997,126,'English','1998-06-03','UK'),
(917,'Deliverance',1972,109,'English','1982-10-05','UK'),
(918,'Trainspotting',1996,94,'English','1996-02-23','UK'),
(919,'The Prestige',2006,130,'English','2006-11-10','UK'),
(920,'Donnie Darko',2001,113,'English',null,'UK'),
(921,'Slumdog Millionaire',2008,120,'English','2009-01-09','UK'),
(922,'Aliens',1986,137,'English','1986-08-29','UK'),
(923,'Beyond the Sea',2004,118,'English','2004-11-26','UK'),
(924,'Avatar',2009,162,'English','2009-12-17','UK'),
(926,'Seven Samurai',1954,207,'Japanese','1954-04-26','JP'),
(927,'Spirited Away',2001,125,'Japanese','2003-09-12','UK'),
(928,'Back to Future',1985,116,'English','1985-12-04','UK'),
(925,'Braveheart',1995,178,'English','1995-09-08','UK');
 select*from movie;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

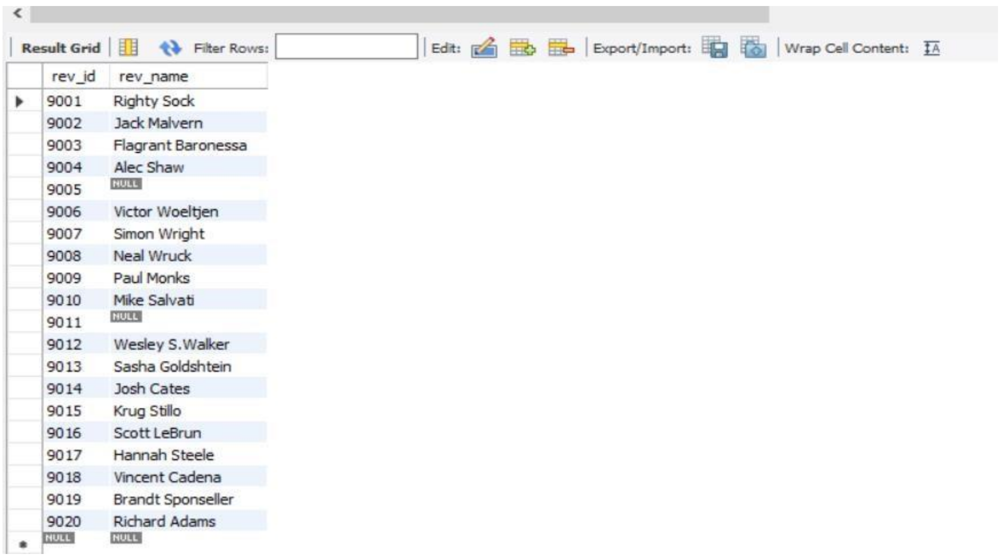
	mov_id	mov_title	mov_year	mov_time	mov_lang	mov_dt_rel	mov_rel_country
	911	Annie Hall	1977	93	English	1977-04-20	USA
	901	Vertigo	1958	128	English	1958-08-24	UK
	903	Lawrence of Arabia	1962	216	English	1962-12-11	UK
	904	The Deer Hunter	1978	183	English	1979-03-08	UK
	905	Amadeus	1984	160	English	1985-01-07	UK
	906	Blade Runner	1982	117	English	1982-09-09	UK
	907	Eyes Wide Shut	1999	159	English	NULL	UK
	908	The Usual Suspects	1995	106	English	1995-08-25	UK
	909	Chinatown	1974	130	English	1974-08-09	UK
	910	Boogie Nights	1997	155	English	1998-02-16	UK
	912	Princess Mononoke	1997	134	Japanese	2001-10-19	UK
▶	913	The Shawshank R...	1994	142	English	1995-02-17	UK
	914	American Beauty	1999	122	English	NULL	UK
	915	Titanic	1997	194	English	1998-01-23	UK
	916	Good Will Hunting	1997	126	English	1998-06-03	UK
	917	Deliverance	1972	109	English	1982-10-05	UK
	918	Trainspotting	1996	94	English	1996-02-23	UK
	919	The Prestige	2006	130	English	2006-11-10	UK
	920	Donnie Darko	2001	113	English	NULL	UK
	921	Slumdog Millionaire	2008	120	English	2009-01-09	UK
	922	Aliens	1986	137	English	1986-08-29	UK
	923	Beyond the Sea	2004	118	English	2004-11-26	UK
	924	Avatar	2009	162	English	2009-12-17	UK
	925	Braveheart	1995	178	English	1995-09-08	UK
	927	Spirited Away	2001	125	Japanese	2003-09-12	UK

```
create table movie_cast( act_id int , mov_id int, role char(30), foreign key(act_id) references
actor(act_id), foreign key(mov_id) references movie(mov_id));
 insert into movie_cast values(101,901,'John
Scottie Ferguson'),
(102,902,'Miss Giddens'),
(103,903,'T E Lawrence'),(104,904,'Michael'),(105,905,'Antonio Salieri'),
(106,906,'Rick Deckard'),(107,907,'Alice Harford'),
(108,908,'McManus'),(110,910,'Eddie Adams'),(111,911,'Alvy Singer'),
(112,912,'San'),(113,913,'Andy Dufresne'),(114,914,'Lester Burnham'),
(115,915,'Rose Dewitt Bukater'),(116,916,'Sean Maguire'),
(117,917,'Ed'),(118,918,'Renton'),(120,920,'Elizabeth Darko'),
(121,921,'Older Jamal'),(122,922,'Ripley'),(114,923,'Bobby Darin'),
(109,909,'J.J Gittes'),(119,919,'Alfred Borden');
 select*from movie_cast;
```



```
act_id  mov_id  role
101     901     John Scottie Ferguson
102     902     Miss Giddens
103     903     T E Lawrence
104     904     Michael
105     905     Antonio Salieri
106     906     Rick Deckard
107     907     Alice Harford
108     908     McManus
110     910     Eddie Adams
111     911     Alvy Singer
112     912     San
113     913     Andy Dufresne
114     914     Lester Burnham
115     915     Rose Dewitt Bukater
116     916     Sean Maguire
117     917     Ed
118     918     Renton
120     920     Elizabeth Darko
121     921     Older Jamal
122     922     Ripley
114     923     Bobby Darin
109     909     J.J Gittes
119     919     Alfred Borden
```

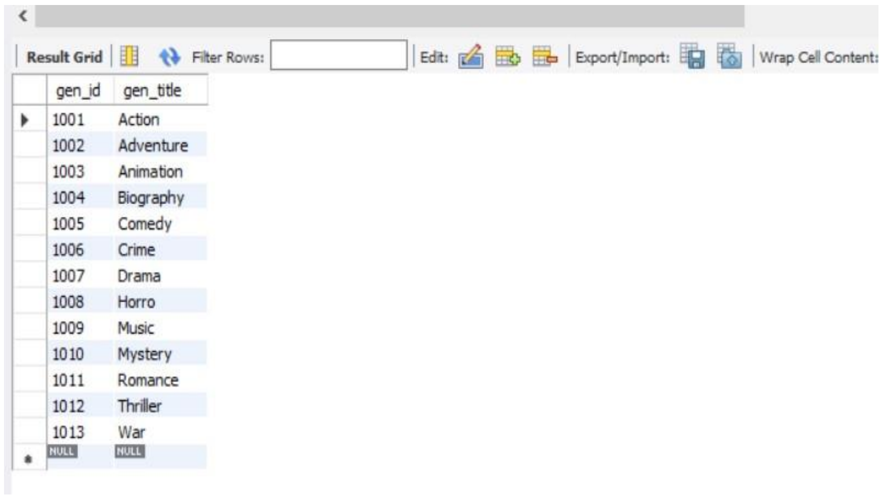
```
create table reviewer( rev_id int primary key, rev_name char(30));
 insert into reviewer(rev_id,rev_name)
values(9001,'Righty Sock'),
(9002,'Jack Malvern'),(9003,'Flagrant Baronessa'),(9004,'Alec Shaw'),
(9005,null),(9006,'Victor Woeltjen'),(9007,'Simon Wright'),(9008,'Neal Wruck'), (9009,'Paul
Monks'),(9010,'Mike Salvati'),(9011,null),(9012,'Wesley S.Walker'),
(9013,'Sasha Goldshtein'),(9014,'Josh Cates'),(9015,'Krug Stillo'),
(9016,'Scott LeBrun'),(9017,'Hannah Steele'),(9018,'Vincent Cadena'),
(9019,'Brandt Sponseller'),(9020,'Richard Adams');
 select*from reviewer;
```



```
rev_id  rev_name
9001    Righty Sock
9002    Jack Malvern
9003    Flagrant Baronessa
9004    Alec Shaw
9005    NULL
9006    Victor Woeltjen
9007    Simon Wright
9008    Neal Wruck
9009    Paul Monks
9010    Mike Salvati
9011    NULL
9012    Wesley S.Walker
9013    Sasha Goldshtein
9014    Josh Cates
9015    Krug Stillo
9016    Scott LeBrun
9017    Hannah Steele
9018    Vincent Cadena
9019    Brandt Sponseller
9020    Richard Adams
NULL    NULL
```

```
create table genres( gen_id int primary key, gen_title char(20));
```

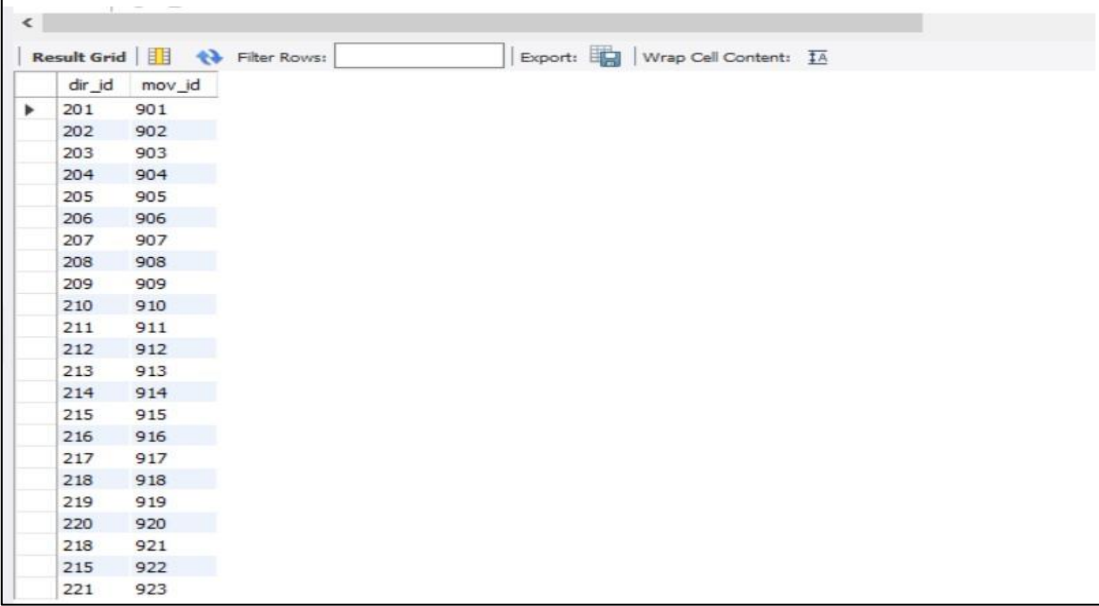
```
 insert into genres(gen_id,gen_title)
values(1001,'Action'),
(1002,'Adventure'),(1003,'Animation'),(1004,'Biography'),(1005,'Comedy'),
(1006,'Crime'),(1007,'Drama'),(1008,'Horro'),(1009,'Music'),
(1010,'Mystery'),(1011,'Romance'),(1012,'Thriller'),(1013,'War');
 select*from genres;
```



gen_id	gen_title
1001	Action
1002	Adventure
1003	Animation
1004	Biography
1005	Comedy
1006	Crime
1007	Drama
1008	Horro
1009	Music
1010	Mystery
1011	Romance
1012	Thriller
1013	War
NULL	NULL

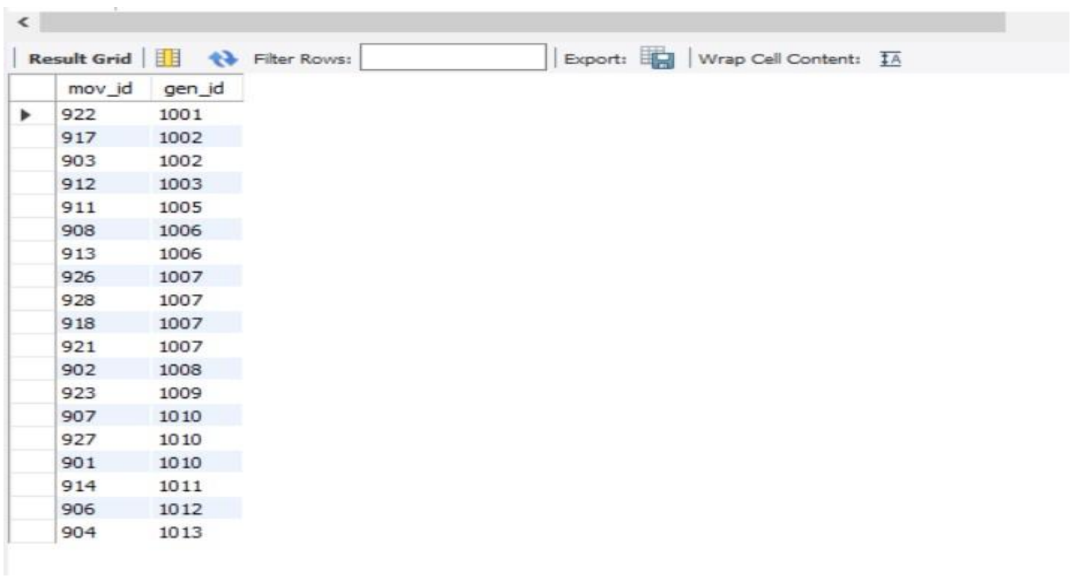
```
create table movie_direction( dir_id int, mov_id int, foreign key (dir_id)references
director(dir_id), foreign key(mov_id) references movie(mov_id));
 insert into movie_direction
values(201,901),(202,902),(203,903),
(204,904),(205,905),(206,906),(207,907),
(208,908),(209,909),(210,910),(211,911),
(212,912),(213,913),(214,914),(215,915), (216,916),(217,917),(218,918),(219,919),
(220,920),(218,921),(215,922),(221,923);
```

```
select*from movie_direction;
```



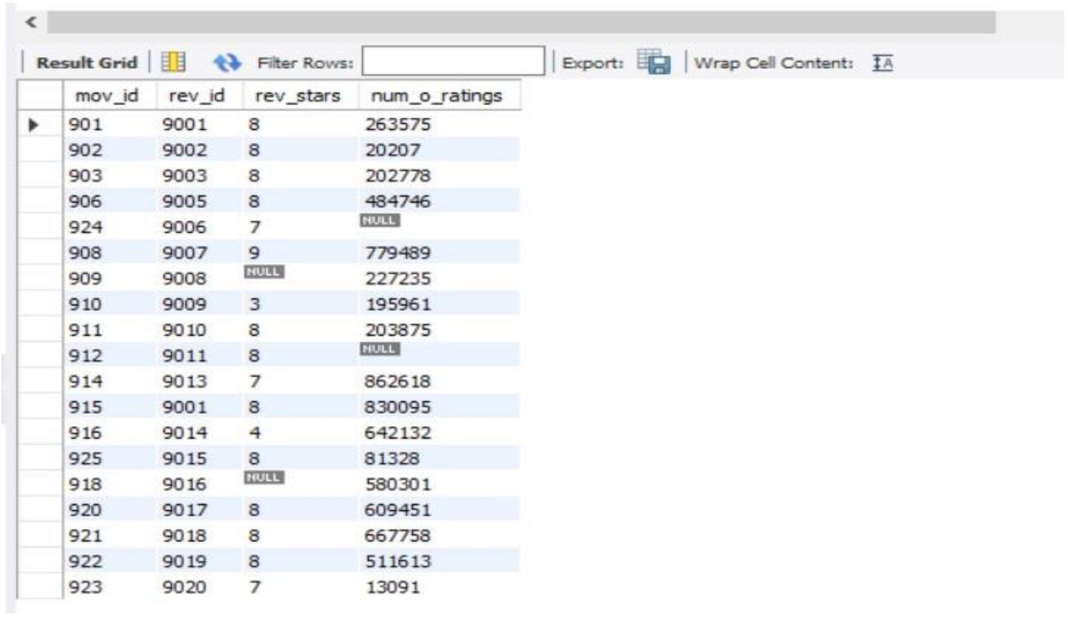
dir_id	mov_id
201	901
202	902
203	903
204	904
205	905
206	906
207	907
208	908
209	909
210	910
211	911
212	912
213	913
214	914
215	915
216	916
217	917
218	918
219	919
220	920
218	921
215	922
221	923

```
create table movie_genres( mov_id int, gen_id int, foreign key(mov_id) references
movie(mov_id), foreign key(gen_id) references genres(gen_id));
 insert into movie_genres
values(922,1001),(917,1002),(903,1002),
(912,1003),(911,1005),(908,1006),(913,1006), (926,1007),(928,1007),(918,1007),(921,1007),
(902,1008),(923,1009),(907,1010),
(927,1010),(901,1010),(914,1011),(906,1012),
(904,1013);
 select*from movie_genres;
```



mov_id	gen_id
922	1001
917	1002
903	1002
912	1003
911	1005
908	1006
913	1006
926	1007
928	1007
918	1007
921	1007
902	1008
923	1009
907	1010
927	1010
901	1010
914	1011
906	1012
904	1013

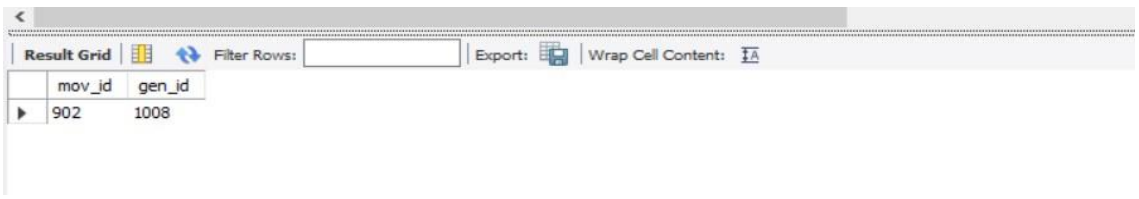
```
create table rating( mov_id int, rev_id int, rev_stars int, num_o_ratings int, foreign key
(mov_id)references movie(mov_id), foreign key(rev_id) references reviewer(rev_id));
 insert into rating
values(901,9001,8.4,263575),(902,9002,7.9,20207),
(903,9003,8.3,202778),(906,9005,8.2,484746),(924,9006,7.3,null),
(908,9007,8.6,779489),(909,9008,null,227235),
(910,9009,3,195961),(911,9010,8.1,203875),(912,9011,8.4,null),
(914,9013,7,862618),(915,9001,7.7,830095),
(916,9014,4,642132),(925,9015,7.7,81328),(918,9016,null,580301),
(920,9017,8.1,609451),(921,9018,8,667758),(922,9019,8.4,511613),
(923,9020,6.7,13091);
 select*from rating;
```



mov_id	rev_id	rev_stars	num_o_ratings
901	9001	8	263575
902	9002	8	20207
903	9003	8	202778
906	9005	8	484746
924	9006	7	NULL
908	9007	9	779489
909	9008	NULL	227235
910	9009	3	195961
911	9010	8	203875
912	9011	8	NULL
914	9013	7	862618
915	9001	8	830095
916	9014	4	642132
925	9015	8	81328
918	9016	NULL	580301
920	9017	8	609451
921	9018	8	667758
922	9019	8	511613
923	9020	7	13091

3. Write a query in SQL to list the Horror movies?

```
 select mov_id,gen_id from movie_genres
where gen_id in
(select gen_id from genres where gen_id=1008);
```

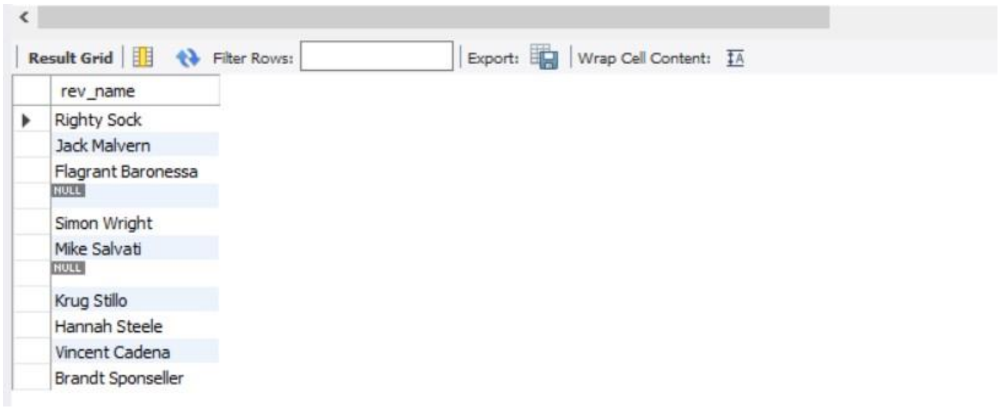


mov_id	gen_id
902	1008

4. Write a query in SQL to find the name of all reviewers who have rated 8 or more stars?

```
 select rev_name from reviewer
where rev_id in
```

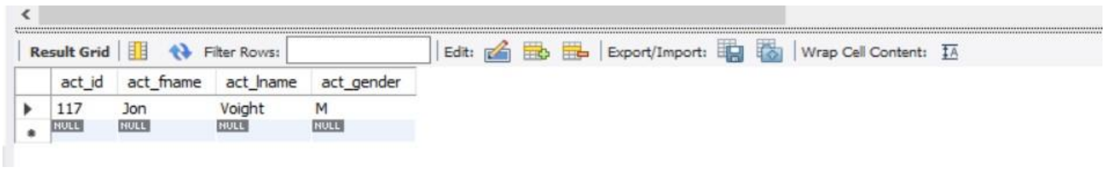
```
(select rev_id from rating where rev_stars >= 8);
```



```
rev_name
Righty Sock
Jack Malvern
Flagrant Baronessa
NULL
Simon Wright
Mike Salvati
NULL
Krug Stillo
Hannah Steele
Vincent Cadena
Brandt Sponseller
```

5. Write a query in SQL to list all the information of the actors who played a role in the movie ‘Deliverance’

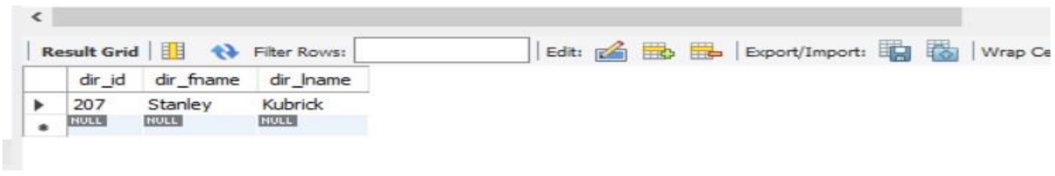
```
SELECT * FROM actor

WHERE act_id IN( SELECT

act_id

FROM movie_cast

WHERE mov_id IN (

SELECT mov_id

FROM movie  WHERE

mov_title='Deliverance'

 )).
```



```
act_id  act_fname  act_lname  act_gender
117     Jon        Voight     M
NULL    NULL       NULL       NULL
```

6. Write a query in SQL to find the name of the director (first and last names) who directed a movie that casted a role for ‘Eyes Wide Shut’. (using subquery)

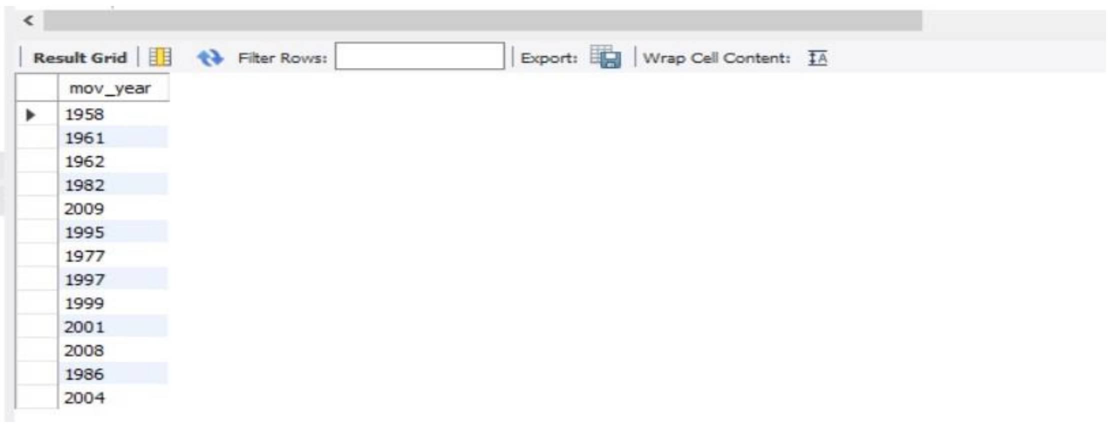
```
 select * from director

where dir_id in

(select dir_id from movie_direction where mov_id=907);
```



```
dir_id  dir_fname  dir_lname
207     Stanley    Kubrick
NULL    NULL       NULL
```

8. Write a query in SQL to find all the years which produced at least one movie and that received a rating of more than 4 stars.

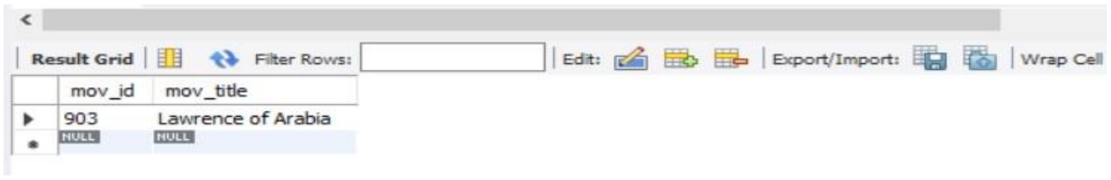
```
SELECT DISTINCT mov_year

FROM movie

WHERE mov_id IN (

SELECT mov_id

FROM rating

WHERE rev_stars>4);
```



```
mov_year
1958
1961
1962
1982
2009
1995
1977
1997
1999
2001
2008
1986
2004
```

10. Write a query in SQL to find the name of movies who were directed by 'David'

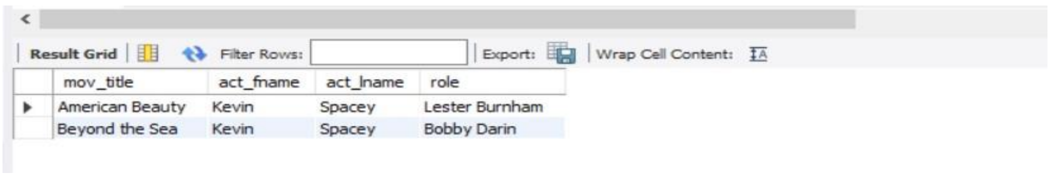
```
select  mov_id,mov_title  from  movie   where  mov_id  in  (select  mov_id  from
movie_direction where dir_id=203);
```



```
mov_id  mov_title
903     Lawrence of Arabia
NULL    NULL
```

12. Find the name of the actor who have worked in more than one movie.

```
SELECT mov_title, act_fname, act_lname, role

FROM movie

JOIN movie_cast

ON movie_cast.mov_id=movie.mov_id

JOIN actor

ON movie_cast.act_id=actor.act_id

WHERE actor.act_id IN (

SELECT act_id

FROM movie_cast

GROUP BY act_id HAVING COUNT(*)>1);
```



	mov_title	act_fname	act_lname	role
▶	American Beauty	Kevin	Spacey	Lester Burnham
	Beyond the Sea	Kevin	Spacey	Bobby Darin