

Oblig2_4 - spørringer mot databasen Movies -Yahye Abdi Ahmed

Table of Contents

Oppgave 1: Hent ut alle filmene sortert på 'year'.....	2
Oppgave 2: Hent ut alle filmene som har en metascore.....	3
Oppgave 3: Hent ut filmer og hvem som har regissert filmen (director).....	4
Oppgave 4: Hent navnet og antallet filmer en regissør har regissert, sortert synkende på antallet.....	5
Oppgave 5: Hent navnet og antallet filmer en regissør har regissert, samt totalinntekten (gross) for disse filmene, sortert synkende på inntekten.....	6
Oppgave 6: Hent ut alle filmer, samt tilhørende regissør og sjanger (genre).....	7
Oppgave 7: Hent ut film og regissør for den filmen som har gitt høyest inntekt.....	8
Oppgave 8: Hent ut filmer med høyest inntekt for hver sjanger.....	9
.....	11
Oppgave 9: Hent ut hvilke filmer Christopher Nolan har regissert, som også Christian Bale har spilt i.....	11
Oppgave 10: Hent ut skuespillerne og antallet filmer de har spilt i, sortert synkende på antallet.....	12

Oppgave 1: Hent ut alle filmene sortert på 'year'

SQL-spørring:

1: Hent ut alle filmene sortert på 'year'

```
SELECT *  
FROM movie  
ORDER BY year DESC;
```

I spørringen her, henter vi alt fra tabellen movie og sorterer data i tabellen med kolonnen "year".

SELECT * henter alt. FROM movie fra tabellen movie. ORDER BY year DESC er gruppering seksjon. Year kolonnen er data type, så den blir sortert på år, her med DESC blir det synkende.

Filmen som nyest i lista er fra 2021.

Spørringen har LIMIT på 300 rad her.

Result Grid										
Filter Rows:										
#	id	title	year	runtime	imdb_rating	metascore	votes	gross	director	ic
1	228	Zack Snyder's Justice League	2021	242	8.1	88	262704	157		
2	620	The Trial of the Chicago 7	2020	129	7.8	76	130453	383		
3	20	Soorai Potru	2020	153	8.6	90	58433	16		
4	33	Hamilton	2020	160	8.5	90	62258	25		
5	91	Dara iz Jasenovca	2020	130	8.3	88	34230	63		
6	92	The Father	2020	97	8.3	88	31071	64		
7	209	Soul	2020	100	8.1	83	222918	144		
8	621	Druk	2020	117	7.8	80	60039	60		
9	576	Sound of Metal	2019	120	7.8	82	62758	355		
10	19	Gisaengchung	2019	132	8.6	96	591108	53.37	15	
11	602	Toy Story 4	2019	100	7.8	84	209892	43...	374	
12	886	Once Upon a Time... in Holl...	2019	161	7.6	83	572662	142.5	6	

movie 5

Action Output

#	Time	Action	Message
5	08:38:17	SELECT * FROM movie ORDER BY @year DESC LIM...	300 row(s) returned
6	08:38:24	SELECT * FROM movie ORDER BY year LIMIT 0, 300	300 row(s) returned
7	08:38:33	SELECT * FROM movie ORDER BY year ASC LIMIT 0, ...	300 row(s) returned
8	08:38:42	SELECT * FROM movie ORDER BY year DESC LIMIT ...	300 row(s) returned

Oppgave 2: Hent ut alle filmene som *har* en metascore

SQL-spørring:

```
SELECT *  
FROM movie  
HAVING metascore  
ORDER BY year DESC;
```

SELECT * FROM movie er SELECT seksjonen som henter alt fra tabellen movie. Under det har vi HAVING seksjon som sjekker om kolonnen metascore har en verdi og ikke er NULL. Det betyr at rader der kolonnen metascore er NULL blir ikke tatt med. Tilslutt har vi ORDER BY clause eller gruppering seksjon. Oppgaven spør ikke om å gruppere, så det er litt ekstra.

Sånn ser resultat av spørringen ut. Vi ser ikke noe film som ikke har metascore.

Result Grid										
Filter Rows: <input type="text"/>										
#	id	title	year	runtime	imdb_rating	metascore	votes	gross	director_id	
1	33	Hamilton	2020	160	8.5	90	62258	NA	25	
2	92	The Father	2020	97	8.3	88	31071	NA	64	
3	620	The Trial of the Chicago 7	2020	129	7.8	76	130453	NA	383	
4	209	Soul	2020	100	8.1	83	222918	NA	144	
5	621	Druk	2020	117	7.8	80	60039	NA	60	
6	19	Gisaengchung	2019	132	8.6	96	591108	53.37	15	
7	61	Avengers: Endgame	2019	181	8.4	78	856425	858.37	48	
8	884	Dark Waters	2019	126	7.6	73	66100	NA	512	
9	85	1917	2019	119	8.3	78	451579	159.23	57	
10	135	Klaus	2019	96	8.2	65	110478	NA	91	
11	576	Sound of Metal	2019	120	7.8	82	62758	NA	355	
12	198	Portrait de la jeune fille ...	2019	122	8.1	95	69081	3.76	134	

movie 8

Action Output

#	Time	Action	Message
8	08:38:42	SELECT * FROM movie ORDER BY year DESC LIMIT ...	300 row(s) returned
9	08:56:52	SELECT * FROM movie HAVING metascore ORDER ...	300 row(s) returned
10	09:06:42	SELECT * FROM movie ORDER BY year DESC LIMIT ...	300 row(s) returned
11	09:06:52	SELECT * FROM movie HAVING metascore ORDER ...	300 row(s) returned

Oppgave 3: Hent ut filmer og hvem som har regissert filmen (director)

SQL-spørring:

```
SELECT m.*, d.name
FROM movie AS m INNER JOIN director AS d
ON m.director_id = d.id;
```

Denne spørringen er litt mer komplisert med INNER JOIN. To tabeller kobles sammen. Det gjøres ved bruk av primary key og foreign key. Forholdet mellom tabellen **movie** og **director** er "one to many". En film kan ha kun 1 director, men 1 director kan ha flere filmer assosiert med seg. Movie tabell har foreign key av tabellen director i kolonnen som heter director_id som representerer primary key for id kolonnen i **director** tabellen. På denne måten blir de to tabellene kunne bli koblet sammen med nøklene.

SELECT m.*, d.name - henter alt movie med "m.*". M er alias for tabellen **movie**, .* henter alle kolonner i movie tabellen. d.name, d er alias for **director** tabellen.

FROM movie **AS** m **INNER JOIN** director **AS** d. Dette bestemmer hvilke kolonne det blir hentet data fra. Movie AS m henter fra tabellen **movie** og samtidig gir tabellnavnet alias som kan bruke i spørringen for enklere referere til tabellen movie. Med INNER JOIN, kobler vi sammen tabellene.

ON bestemmer når hvor eller kolonne de to tabellene kobles sammen på. Her med movie.director_id og director.id.

INNER JOIN henter som har felles verdi i tabellene. Også er det mulig å velge i SELECT seksjonen hvilke tabeller vi ønsker å hente. Her i resultat ser

#	id	title	year	runtime	imdb_rating	metascore	votes	gross	director_id	name
1	1	The Shawshank Redemption	1994	142	9.3	80	2377480	28.34	1	Frank Darabont
2	25	The Green Mile	1999	189	8.6	61	1166235	136.8	1	Frank Darabont
3	2	The Godfather	1972	175	9.2	100	1646818	134.97	2	Francis Ford Coppola
4	4	The Godfather: Part II	1974	202	9	90	1146666	57.3	2	Francis Ford Coppola
5	76	Apocalypse Now	1979	147	8.4	94	614397	83.47	2	Francis Ford Coppola
6	700	The Conversation	1974	113	7.8	85	100624	4.42	2	Francis Ford Coppola
7	977	The Godfather: Part III	1990	162	7.6	100	365307	66.67	2	Francis Ford Coppola
8	3	The Dark Knight	2008	152	9	84	2341261	534.86	3	Christopher Nolan
9	9	Inception	2010	148	8.8	74	2099854	292.58	3	Christopher Nolan
10	21	Interstellar	2014	169	8.6	74	1544204	188.02	3	Christopher Nolan
11	37	The Prestige	2006	130	8.5	66	1208183	53.09	3	Christopher Nolan
12	65	The Dark Knight Rises	2012	164	8.4	78	1536594	448.14	3	Christopher Nolan

Result 13

Action Output

#	Time	Action	Message	Duration / Fe
16	10:16:49	SELECT * FROM movie_has_actor LIMIT 0, 300	300 row(s) returned	0,0023 sec / 0,
17	10:59:21	SELECT m, d.name FROM movie AS m INNER JOIN d...	Error Code: 1054. Unknown column 'm' in 'fiel...	0,00061 sec
18	10:59:32	SELECT *, d.name FROM movie AS m INNER JOIN di...	300 row(s) returned	0,0021 sec / 0,
19	11:00:06	SELECT m.*, d.name FROM movie AS m INNER JOIN...	300 row(s) returned	0,0024 sec / 0,

Oppgave 4: Hent navnet og antallet filmer en regissør har regissert, sortert synkende på antallet

SQL-spørringen:

4: Hent navnet og antallet filmer en regissør har regissert, sortert synkende på antallet

```
SELECT d.name AS regissor, COUNT(d.name) AS antall_filmer
FROM movie AS m
INNER JOIN director AS d
ON m.director_id = d.id
GROUP BY regissor
ORDER BY antall_filmer DESC
;
```

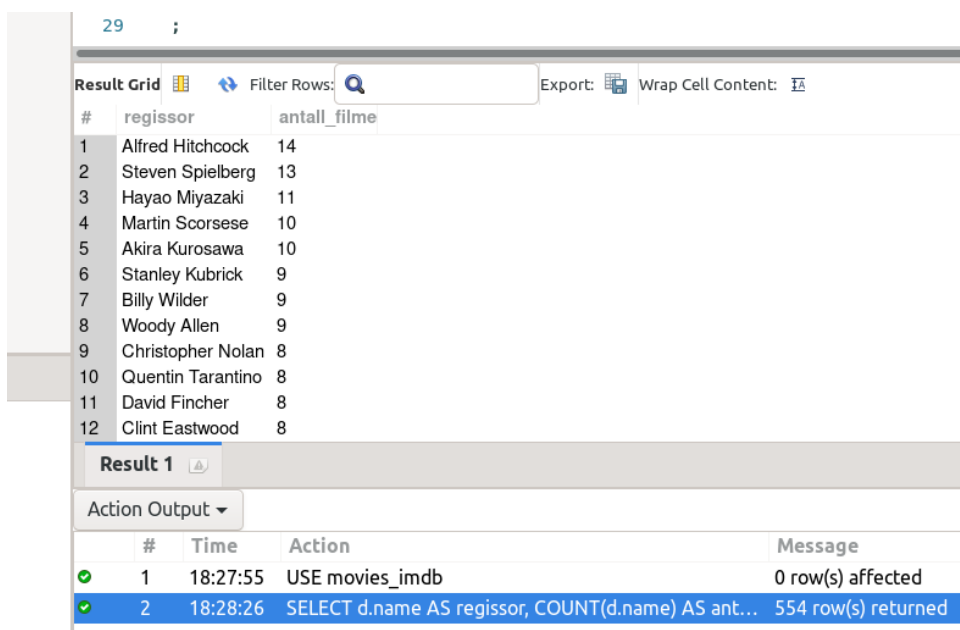
For å kunne hente antall filmer en regissør har regissert, slår vi sammen movie og director tabellen. Så grupperer resultatet med regissører. Til slutt sorterer vi resultatet synkende med antall filmer per regissør. Fra regissør med flest antall filmer til den med minst antall filmer regissert.

SELECT d.name **AS** regissor, **COUNT**(d.name) **AS** antall_filmer. I **SELECT** seksjon henter og viser kolonnen name fra director med alias regissor. **COUNT**(d.name) aggregering funksjon som teller opp alle rader per director sin navn.

FROM movie **AS** m klønnene blir fra movie tabellen, alias blir m for movie. Vi joiner movie tabellen med director tabellen med **INNER JOIN** director **AS** d. Med **ON** m.director_id = d.id blir de to tabellene koblet sammen ved primary key og foreign key. Det er ganske lik **INNER JOIN** som i oppgave3.

Vi grupperer **GROUP BY** regissor med director name kolonne. Sånn at vi får resultat og ikke bare 1. Tilslutt sorteres data høyest til laveste antall_filmer, som returnerer antall film per director har regissert.

Spørringen viser at Alfred Hitchcock har regissert flest filmer i denne databasen.



29 ;

#	regissor	antall_filme
1	Alfred Hitchcock	14
2	Steven Spielberg	13
3	Hayao Miyazaki	11
4	Martin Scorsese	10
5	Akira Kurosawa	10
6	Stanley Kubrick	9
7	Billy Wilder	9
8	Woody Allen	9
9	Christopher Nolan	8
10	Quentin Tarantino	8
11	David Fincher	8
12	Clint Eastwood	8

Result 1

#	Time	Action	Message
1	18:27:55	USE movies_imdb	0 row(s) affected
2	18:28:26	SELECT d.name AS regissor, COUNT(d.name) AS ant...	554 row(s) returned

Oppgave 5: Hent navnet og antallet filmer en regissør har regissert, samt totalinntekten (gross) for disse filmene, sortert synkende på inntekten

SQL-spørringen:

```
SELECT d.name AS regissor, COUNT(d.name) AS antall_filmer, SUM(m.gross) AS
totalinntekt
FROM movie AS m
INNER JOIN director AS d
ON m.director_id = d.id
GROUP BY regissor
ORDER BY totalinntekt DESC
;
```

Denne spørringen bygger videre på fra den i oppgave4 over. Det som er nytt er at vi hentet kolonne sum inneholder SUM() av gross og at data blir sortert med totalinntekt per regissør.

I SELECT seksjon henter 3 kolonner, director name, antall_filmer som er lik den i oppgave4 og SUM() movie.gross med alias totalinntekt.

INNER JOIN director **AS** d **ON** m.director_id = d.id slår sammen movie og director tabellen med primary og foreign key.

GROUP BY regissor, data blir gruppert med regissor igjen her også og tilslutt sortert med totalinntekt synkende. Så vi får se synkende liste hvem som har tjent mest inntekt av filmene sine sammenlagt.

Steven Spielberg med 13 filmer har størst inntekt, men på andre plass Joe Russo med bare 4 tjent ganske mye.

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
#	regissor	antall_filme	totalinntekt			
1	Steven Spielberg	13	2478.129991531372			
2	Joe Russo	4	2205.0399780273438			
3	Christopher Nolan	8	1937.449993133545			
4	James Cameron	5	1748.2400283813477			
5	Peter Jackson	5	1597.3099975585938			
6	Abrams	3	1423.1699829101562			
7	Lee Unkrich	4	1331.6100158691406			
8	Robert Zemeckis	5	1049.4400024414062			
9	David Yates	3	978.9500122070312			
10	Brad Bird	3	893.1800193786621			
11	Ridley Scott	6	766.7100028991699			
12	Quentin Tarantino	8	727.0399951934814			
Result 2						
Action Output						
#	Time	Action	Message			
✓ 1	18:27:55	USE movies_imdb	0 row(s) affected			
✓ 2	18:28:26	SELECT d.name AS regissor, COUNT(d.name) AS ant...	554 row(s) return			
✓ 3	18:49:55	SELECT d.name AS regissor, COUNT(d.name) AS ant...	554 row(s) return			

Oppgave 6: Hent ut alle filmer, samt tilhørende regissør og sjanger (genre)

SQL-spørring:

6: Hent ut alle filmer, samt tilhørende regissør og sjanger (genre)
ressurs: fikk god forklaring fra studentassistenter på hvordan tabeller kobles i spørring i oppgave 4 og 5.

```
SELECT m.*, d.name AS regissør, g.name AS sjanger
FROM movie AS m
INNER JOIN director AS d
ON m.director_id = d.id
INNER JOIN movie_has_genre AS mg
ON mg.movie_id = m.id
INNER JOIN genre AS g
ON mg.genre_id = g.id;
```

I denne spørringen kobler vi sammen 4 tabeller. Først kobler vi **movie** og **director** tabellene sammen. Etter det kobler vi **movie** tabellen til mellom tabellen **movie_has_genre**. Til slutt kobler vi **movie_has_genre** tabellen til **genre** tabellen.

SELECT m.*, d.name **AS** regissør, g.name **AS** sjanger. Select seksjon henter alle kolonner fra tabellen **movie** med m.*. d.name er name kolonnen fra **director** tabellen. Den henter navn på director. Alias for denne kolonnen er regissør som blir vist i sluttresultat. Siste kolonnen g.name med alias sjanger er name kolonnen fra tabellen **genre**.

FROM movie **AS** m. Vi henter data fra tabellen movie.

INNER JOIN director **AS** d **ON** m.director_id = d.id. Første **INNER JOIN** kobling mellom movie tabellen og director tabellen. Det blir gjort ved å bruke primary key og foreign key. Movie tabellen har foreign til tabellen director som referer til primary key id i director tabellen. Så **INNER JOIN** skjer ved å sammenligne foreign key og primary key med **ON**.

INNER JOIN movie_has_genre **AS** mg **ON** mg.movie_id = m.id. Andre **INNER JOIN** samme syntax som den. Movie tabellen og genre tabellen many to many forhold. movie_has_genre tabellen er mellom tabellen danner many to many forholdet mellom movie og genre tabellen.

Så for å koble **movie** og **genre** tabellene, gjør vi koblingen ved å koble **movie_has_genre** til **movie** tabellen på den ene siden og koble **movie_has_genre** til **genre** tabellen på andre siden. Det er samme som 3 person som står ved siden av hverandre og holder hender. Den personen som er i midten blir som mellom tabellen movie_has_genre.

movie_has_genre inneholder foreign key til både **movie** og **genre** tabellen. Her kobler vi movie tabellen til movie_has_genre tabellen ved å sammenligne med **ON** primary key i movie tabellen med foreign i movie_has_genre tabellen.

INNER JOIN genre **AS** g **ON** mg.genre_id = g.id;. Til slutt kobler **movie_has_genre** tabellen til **genre** tabellen. Koblingen er lik den mellom movie og movie_has_genre.

I **INNER JOIN** sammeligner med primary key i **genre** tabellen med foreign key i **movies_has_genre** tabellen.

I resultat ser vi at alle kolonne fra **movie** tabellen hentet. Det 10 kolonner i movie tabellen. Fra **director** tabellen henter vi kun regissør kolonnen og **genre** henter vi kun en kolonne, sjanger kolonnen.

#	id	title	year	runtime	imdb_rating	metascore	votes	gross	director_id	regissør	sjanger
1	1	The Shawshank Redemption	1994	142	9.3	80	2377480	28.34	1	Frank Darabont	Drama
2	25	The Green Mile	1999	189	8.6	61	1166235	136.8	1	Frank Darabont	Drama
3	25	The Green Mile	1999	189	8.6	61	1166235	136.8	1	Frank Darabont	Crime
4	25	The Green Mile	1999	189	8.6	61	1166235	136.8	1	Frank Darabont	Fantasy
5	2	The Godfather	1972	175	9.2	100	1646818	134.97	2	Francis Ford Coppola	Drama
6	2	The Godfather	1972	175	9.2	100	1646818	134.97	2	Francis Ford Coppola	Crime
7	4	The Godfather: Part II	1974	202	9	90	1146666	57.3	2	Francis Ford Coppola	Drama
8	4	The Godfather: Part II	1974	202	9	90	1146666	57.3	2	Francis Ford Coppola	Crime
9	76	Apocalypse Now	1979	147	8.4	94	614397	83.47	2	Francis Ford Coppola	Drama
10	76	Apocalypse Now	1979	147	8.4	94	614397	83.47	2	Francis Ford Coppola	Mystery
11	76	Apocalypse Now	1979	147	8.4	94	614397	83.47	2	Francis Ford Coppola	War
12	700	The Conversation	1974	113	7.8	85	100624	4.42	2	Francis Ford Coppola	Drama

#	Time	Action	Message	Durati
1	21:57:40	SELECT m.*, d.name AS regissør, g.name AS sjanger ...	1000 row(s) returned	0,0082
2	22:38:21	SELECT m.*, d.name AS regissør, g.name AS sjanger ...	1000 row(s) returned	0,0026

Oppgave 7: Hent ut film og regissør for den filmen som har gitt høyest inntekt

SQL-spørring:

7: Hent ut film og regissør for den filmen som har gitt høyest inntekt
 # gruppering når MAX(gross) er i øverst SELECT seksjon fungerte dårlig. Da må vi gruppere data og vi får ikke 1 resultat i gruppering.

```
SELECT m.title AS film, d.name AS regissør, m.gross AS inntekt
FROM movie AS m
INNER JOIN director AS d
ON m.director_id = d.id
WHERE m.gross = (SELECT MAX(gross) FROM movie)
;
```

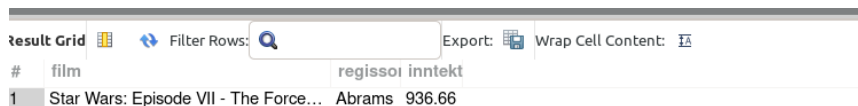
I denne kobler vi sammen movie tabellen med director tabellen.

Select seksjon henter movie.title med alias film. director.name med alias regissør. Tredje og siste kolonnen er movie.gross med inntekt. FROM movie, henter fra tabellen movie med alias m.

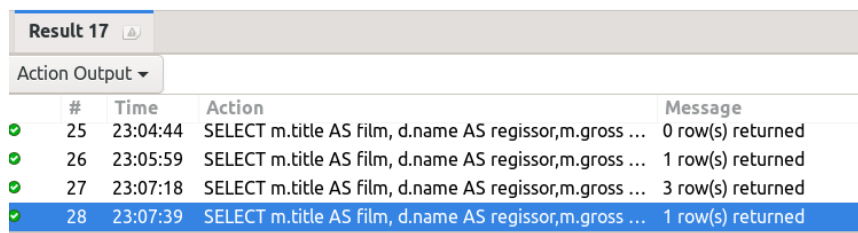
INNER JOIN director **AS** d **ON** m.director_id = d.id. **INNER JOIN** i one to many relationship blir samme som vi gjorde oppgavene før. Movie tabellen og genre tabellen kobles sammen med primary og foreign. Movie tabellen har foreign key til director tabellen.

WHERE m.gross = (**SELECT MAX**(gross) **FROM** movie);. Tilslutt kjører **MAX**(gross) i **WHERE** seksjon. Dette gjør at vi ikke trenger å gruppere og vi hentet kun 1 rad som inneholder filmen størst gross. Dette gjør ved movie.gross filmen er lik den største i select parentes.

I resultat henter vi kun 1 rad. Star Wars Episode VII er filmen med størst gross i denne databasen.



#	film	regissør	inntekt
1	Star Wars: Episode VII - The Force...	Abrams	936.66



#	Time	Action	Message
25	23:04:44	SELECT m.title AS film, d.name AS regissør,m.gross ...	0 row(s) returned
26	23:05:59	SELECT m.title AS film, d.name AS regissør,m.gross ...	1 row(s) returned
27	23:07:18	SELECT m.title AS film, d.name AS regissør,m.gross ...	3 row(s) returned
28	23:07:39	SELECT m.title AS film, d.name AS regissør,m.gross ...	1 row(s) returned

Oppgave 8: Hent ut filmer med høyest inntekt for hver sjanger

SQL-spørring:

8: Hent ut filmer med høyest inntekt for hver sjanger

source: <https://www.youtube.com/watch?v=cW2MLIZ5CaU>

jeg satt fast lenge på den, men fant løsning på video som forklarte at jeg kunne ikke ha irrelevante kolonner i select som ikke er en del GROUP BY clausen
fikk hjelp med studass

```
SELECT movie.title AS film, x.sjanger AS sjanger, x.inntekt
FROM
(
SELECT g.name AS sjanger, MAX(m.gross) AS inntekt
FROM movie AS m
INNER JOIN movie_has_genre AS mg
ON mg.movie_id = m.id
INNER JOIN genre AS g
ON mg.genre_id = g.id
GROUP BY g.name
) AS x, movie
```

```
WHERE x.inntekt = movie.gross;
```

Denne spørringen består av SELECT seksjoner.

SELECT movie.title **AS** film, x.sjanger **AS** sjanger, x.inntekt. I første SELECT seksjon velger vi ut 3 kolonner. Første kolonne fra movie tabellen og henter navn på filmer med movie.title, alias filmer. Andre kolonnen er tabellen x og henter sjanger med x.sjanger, alias sjanger. X-tabellen er resultat fra andre SELECT seksjon som blir sett som egen kolonne, der x er alias for den tabellen. Tredje kolonne fra movie tabellen og henter gross klonnen med movie.gross, alias inntekt.

```
FROM  
(  
SELECT g.name AS sjanger, MAX(m.gross) AS inntekt  
FROM movie AS m  
INNER JOIN movie_has_genre AS mg  
ON mg.movie_id = m.id  
INNER JOIN genre AS g  
ON mg.genre_id = g.id  
GROUP BY g.name
```

) **AS** x, movie Vi henter de 3 kolonnene i SELECT fra subquery SELECT her som INNER joiner movie tabellen med genre tabellen gjennom movie_has_genre mellom tabellen. SELECT i subquery henter 2 kolonnen, først kolonne fra genre tabellen, genre.name med alias sjanger. Andre kolonnen er fra movie og henter movie.gross kolonnen. Her bruker vi aggregert funksjon MAX() for å hente kun filmer med høyest gross, alias for kolonnen er inntekt. Resultat i subquery blir gruppert med genre.name, altså sjanger og vi kun hver sjanger kun 1 gang. Tilsammen 21 sjanger. Subquery her alias x. Vi henter også movie tabellen.

WHERE x.inntekt = movie.gross; Til slutt, I WHERE sammenligner i inntekt fra subquery med movie.gross som gjør at vi kan se navnet på filmen som har høyest gross i hver sjanger.

Vi får 21 treff i resultat siden antall sjanger er 21.

Result Grid				
Filter Rows:		Export:		
Wrap Cell Content:				
#	film	sjanger	inntekt	
1	The Dark Knight	Crime	534.86	
2	Saving Private Ryan	War	216.54	
3	Joker	Thriller	335.45	
4	Avengers: Endgame	Drama	858.37	
5	The Sixth Sense	Mystery	293.51	
6	Gone with the Wind	History	198.68	
7	Bohemian Rhapsody	Music	216.43	
8	Dances with Wolves	Western	184.21	
9	The Exorcist	Horror	232.91	
10	Fiddler on the Roof	Musical	80.5	
11	Star Wars: Episod...	Action	936.66	
12	Star Wars: Episod...	Adventure	936.66	

Result 3

Action Output

#	Time	Action	Message
1	20:05:58	SELECT movie.title AS film, x.sjanger AS sjanger, x.in...	21 row(s) returned
2	20:06:38	SELECT movie.title AS film, x.sjanger AS sjanger, x.in...	70 row(s) returned
3	20:07:10	SELECT movie.title AS film, x.sjanger AS sjanger, x.in...	21 row(s) returned

Oppgave 9: Hent ut hvilke filmer Christopher Nolan har regissert, som også Christian Bale har spilt i

SQL-spørring:

```
# 9: Hent ut hvilke filmer Christopher Nolan har regissert, som også Christian
Bale har spilt i
SELECT m.title AS film, d.name AS regissor, a.name AS skuespiller
FROM movie AS m
INNER JOIN director AS d
ON m.director_id = d.id
INNER JOIN movie_has_actor AS ma
ON ma.movie_id = m.id
INNER JOIN actor AS a
ON ma.actor_id = a.id
WHERE d.name = "Christopher Nolan" AND a.name = "Christian Bale";
```

For å finne filmer der regissør er ”Christoffer Nolan” og skuespiller er ”Christian Bale”, må vi koble **movie** tabellen til **director** tabellen for å finne regissøren. Vi må også koble movie tabellen til actor tabellen for å finne skuespilleren.

```
SELECT m.title AS film, d.name AS regissor, a.name AS skuespiller
FROM movie AS m. I select seksjon henter vi 3 kolonner. Første kolonnen er
navnet på filmene. Det henter vi fra movie tabellen, movie.title og gir den
alias film. Andre kolonnen er regissor hentet fra director tabellen,
director.name og gir den alias regissor. Tredje kolonnen er skuespillernavn vi
henter fra actor tabellen, actor.name og gir den alias skuespiller. Vi henter
fra movie tabellen med alias m og joiner den videre med andre tabellene.
```

```
INNER JOIN director AS d
ON m.director_id = d.id. Vi inner joiner movie med director tabell samme som vi
gjorde i forrige oppgaver. Movie tabell har foreign key til director tabell. Så
vi joiner tabellene sammen ved å sammen foreign key i movie tabell til primary
key i director tabell. Director tabell gir vi her alias d.
```

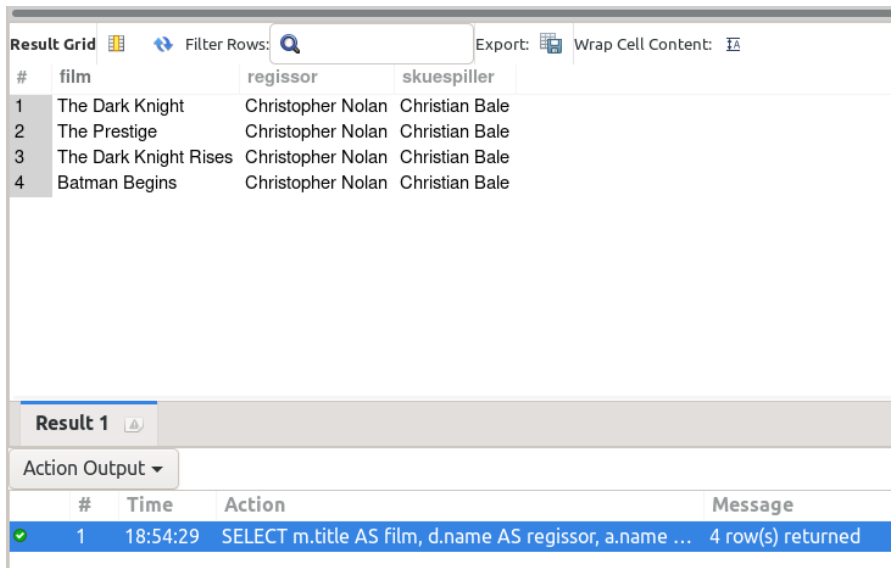
```
INNER JOIN movie_has_actor AS ma
ON ma.movie_id = m.id
INNER JOIN actor AS a
ON ma.actor_id = a.id. For å joine sammen movie tabell og actor tabell, joiner
vi via mellom tabellen movie_has_actor tabellen. Dette er 2 inner join.
```

I den første inner join kobler **movie_has_actor** tabellen til **movie** ved å sammenligne foreign key i movie_has_actor for movie tabell, movie_id med primorary i movie tabellen, movie.id.

I den andre inner join, kobler vi **actor** tabellen til **movie_has_actor** tabellen ved å sammenligne foreign key i movie_has_actor for actor, actor_id til primary key i actor tabellen, actor.id.

WHERE d.name = "Christopher Nolan" **AND** a.name = "Christian Bale";. Til slutt i **WHERE** clause begrenser vi data som blir hentet der director sin navn er Christoffer Nolan og der skuespiller sin navn er Christian Bale.

Vi får 4 treff i resultat. Dette er de 4 filmene som har Nolan som regissør og Bale som skuespiller.



The screenshot shows a database interface with a 'Result Grid' and an 'Action Output' section. The 'Result Grid' displays the following data:

#	film	regissør	skuespiller
1	The Dark Knight	Christopher Nolan	Christian Bale
2	The Prestige	Christopher Nolan	Christian Bale
3	The Dark Knight Rises	Christopher Nolan	Christian Bale
4	Batman Begins	Christopher Nolan	Christian Bale

The 'Action Output' section shows a single message:

#	Time	Action	Message
1	18:54:29	SELECT m.title AS film, d.name AS regissør, a.name ...	4 row(s) returned

Oppgave 10: Hent ut skuespillerne og antallet filmer de har spilt i, sortert synkende på antallet

SQL-spørring:

10: Hent ut skuespillerne og antallet filmer de har spilt i, sortert synkende på antallet

```
SELECT a.name AS skuespiller, COUNT(m.title) AS filmer
FROM movie AS m
INNER JOIN movie_has_actor AS ma
ON ma.movie_id = m.id
INNER JOIN actor AS a
ON ma.actor_id = a.id
GROUP BY a.name
ORDER BY filmer DESC
;
```

Denne spørringen ligner på den fra oppgave 4 der vi skulle hente regissører og antall filmer de regisserte og sortere resultatet synkende fra regissør med fleste filmer til den med minste. Her skal vi hente skuespillere og antall filmer de har vært med på og sortere resultatet synkende.

```
SELECT a.name AS skuespiller, COUNT(m.title) AS filmer
FROM movie AS m. I SELECT clause henter vi 2 kolonner. Den første er fra actor
tabellen og vi henter actor.title med alias skuespiller. I andre kolonnen
COUNT() movie.title fra movie tabellen med alias filmer. COUNT() teller opp
antall ganger skuespiller er assosiert med en film. Det gir oss tilbake numerisk
tall. FROM movie, fra tabellen movie med alias m.
```

```
INNER JOIN movie_has_actor AS ma
ON ma.movie_id = m.id
INNER JOIN actor AS a
ON ma.actor_id = a.id.
```

For å joine movie tabellen med actor tabellen, bruker vi mellom tabllen movie_has_actor. Mellom tabllen danner many to many forholdet mellom movie og actor tabellene. Først INNER JOIN kobler movie_has_actor til movie tabellen ved å foreign key i movie_has_actor, movie_id til primary i movie, movie.id. I andre INNER JOIN blir actor tabellen koblet til movie_has_actor ved på samme måte, men denne til actor tabellen. Foreign key i movie_has_actor, actor_id blir sammenlignet med primary key i actor, actor.id.

```
GROUP BY a.name
ORDER BY filmer DESC
;
```

Til slutt grupperer vi data med GROUP BY clause med actor.name og sorterer også med filmer som er alias for COUNT(m.title). DESC gjør at det blir synkende, skuespillere som har spilt i flest filmer kommer høyst på resultat tabellen.

De niro, Hanks og Pacino er top 3 skuespillere med filmer spilt i denne databasen.

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
#	skuespiller	filmer			
1	Robert De Niro	17			
2	Tom Hanks	14			
3	Al Pacino	13			
4	Brad Pitt	12			
5	Clint Eastwood	12			
6	Christian Bale	11			
7	Leonardo DiCaprio	11			
8	Matt Damon	11			
9	James Stewart	10			
10	Michael Caine	9			
11	Scarlett Johansson	9			
12	Humphrey Bogart	9			

Result 6				
Action Output ▼				
	#	Time	Action	Message
✓	3	18:59:23	SELECT g.name AS sjanger, MAX(m.gross) AS inntek...	1000 row(s) returned
✓	4	19:00:03	SELECT g.name AS sjanger, MAX(m.gross) AS inntek...	21 row(s) returned
✓	5	19:00:30	SELECT a.name AS skuespiller, COUNT(m.title) AS fil...	1000 row(s) returned
✓	6	19:45:41	SELECT a.name AS skuespiller, COUNT(m.title) AS fil...	2688 row(s) returned