Øving 3

Oppgave 1: Lage SQL-tabeller og legge inn data Gitt følgende relasjonsskjema og ER-diagram:

Film (FilmID, Tittel, Produksjonsår, RegissørID)
SjangerForFilm (FilmID, SjangerID)
Sjanger (SjangerID, Navn, Beskrivelse)
Skuespiller (SkuespillerID, Navn, Fødselsår)
SkuespillerIFilm (FilmID, SkuespillerID, Rolle)
Regissør (RegissørID, Navn)

a) Vi ønsker at dersom man sletter eller oppdaterer en rad i Film-tabellen, skal tilsvarende referanser til denne raden også slettes eller oppdateres. Dette betyr at hvis man for eksempel sletter en film, skal SkuespillerIFilm- og SjangerForFilm-tabellene også slette de radene der referanser til denne filmen inngår. Hvordan kan man spesifisere en slik restriksjon i SQL?

FROM Film WHERE row in SELECT(DISTINCT row in SkuespillerIFilm);
FROM FIlm WHERE row in SELECT(DISTINCT row in SjangerForFilm);

b) Skriv nå kode som konstruerer disse tabellene i SQL. Velg selv passende datatyper for attributtene og husk primær- og fremmednøkler. Ta også med kravet presentert i a).

CREATE TABLE Film (FilmID INT, Produksjonsår INT, RegissørID INT);
CREATE TABLE SjangerForFilm (FilmID INT, SjangerID INT);
CREATE TABLE SkuespillerIFilm (FilmID INT, SkuespillerID INT, Rolle VARCHAR(20));

c) Skriv SQL-setninger som legger inn følgende data i databasen: Regissør(1, “Peyton Reed”) Regissør(2, "Tom Shadyac") Film(1, “Yes Man ”, 2008, 1) Skuespiller(“1 , “Jim Carrey”, 1962) SkuespillerIFilm(1,1, “Carl”)

INSERT INTO Regissør VALUES (*1, “Peyton Reed”)(2, "Tom Shadyac");*
INSERT INTO Film VALUES *(1, “Yes Man ”, 2008, 1);*
INSERT INTO Skuespiller VALUES *(“1 , “Jim Carrey”, 1962);*
INSERT INTO SkuespillerIFilm VALUES *(1,1, “Carl”);*

d) Skriv en SQL-setning som oppdaterer navnet til Jim Carrey til James Eugene Carrey.
UPDATE Skuespiller

```
SET Navn = ‘James Eugene Carrey’
WHERE Navn = ‘Jim Carrey’;
```

e) Skrive en SQL-setning som sletter Tom Shadyac fra databasen.

```
DELETE FROM Regissør WHERE Navn = ’Tom Shadyac’;
```

Oppgave 2: Spørringer i SQL

Vi skal nå fortsette med filmdatabasen fra oppgave 1. Dokumentasjonen for spørringer i MySQL kan være nyttig om du står fast på denne oppgaven . 1 Vi anbefaler at du går på Blackboard og henter inn kode fra et script, ov3_filmdata, som setter inn en del mer data i tabellene for deg. Alternativt kan du koble deg opp til en database som kjører på en server på IDI. Se eget skriv på Blackboard for mer informasjon: mysql_filmbase.pdf. 1 https://dev.mysql.com/doc/refman/5.7/en/sql-syntax-data-manipulation.html 2 Skriv spørringer i SQL som….

a) Henter ut filmID, tittel, produksjonsår og regissørID på alle filmer.

```
SELECT * FROM film;
```

b) Finner navn på alle skuespillere født senere enn 1960.

```
SELECT Navn FROM skuespiller WHERE Fødselsår > 1960;
```

c) Finner navn på alle skuespillere født på 80-tallet, sortert i alfabetisk rekkefølge.

```
SELECT Navn FROM skuespiller
WHERE Fødselsår BETWEEN 1980 and 1989
ORDER BY Navn ASC;
```

d) Finner titlene på alle filmene og de tilhørende rollene som “Morgan Freeman” har spilt.

```
SELECT film.Tittel, skuespillerifilm.Rolle
FROM skuespillerifilm
INNER JOIN film ON film.FilmId = skuespillerifilm.FilmID
INNER JOIN skuespiller ON skuespiller.SkuespillerID = skuespillerifilm.SkuespillerID
WHERE skuespiller.Navn = "Morgan Freeman"
```

e) Henter ut de distinkte titlene på filmene hvor regissøren og en av skuespillerne i filmen har likt navn.

```
select distinct film.Tittel, skuespiller.Navn, regissør.Navn
from ((regissør
inner join film on film.RegissørID = regissør.RegissørID )
inner join skuespiller on skuespiller.Navn = regissør.Navn)
```

f) Finner antallet skuespillere som har et navn som starter på "C".

```
SELECT SUM(case when Navn like 'C%' then 1 else 0 end) FROM skuespiller;
```

g) For hver sjanger finner sjangernavnet og antallet filmer av den sjangeren.

```
select sjanger.Navn, count(sjanger.SjangerID)
from sjangerforfilm
inner join film on film.FilmId=sjangerforfilm.FilmID
inner join sjanger on sjanger.SjangerID=sjangerforfilm.SjangerID
group by sjanger.SjangerID;
```

h) Finner navnet på skuespillere som har spilt i filmen "Ace Ventura: Pet Detective", men aldri i filmen "Ace Ventura: When Nature Calls".

```
select skuespiller.Navn
from ((skuespillerifilm
inner join skuespiller on skuespiller.SkuespillerID = skuespillerifilm.SkuespillerID )
inner join film on film.FilmId = skuespillerifilm.FilmId )
where film.Tittel = 'Ace Ventura: Pet Detective' and not film.Tittel = 'Ace Ventura:
When Nature Calls'
```

i) For hver film finner navnet på filmen, filmID og gjennomsnittlig fødselsår på skuespillerne i filmen. Vi ønsker kun å få med de filmene som har gjennomsnittlig fødselsår større enn gjennomsnittlig fødselsår for alle skuespillerne i databasen. (Hint: Her kan det være lurt med en underspørring i en HAVING-del etter aggregeringen).

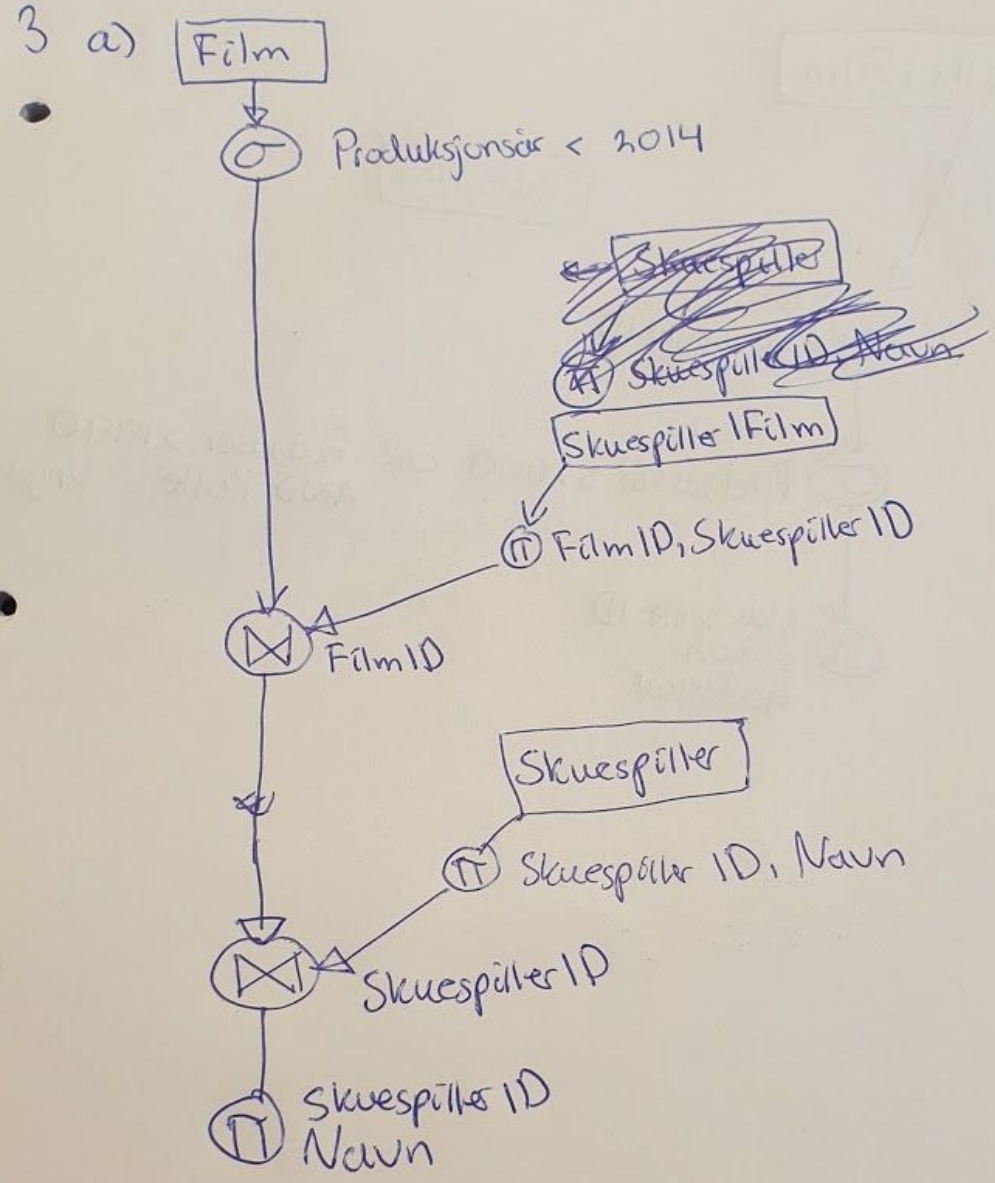
```
select film.Tittel, film.FilmId, avg(skuespiller.Fødselsår) as gjsnittFødselsÅr
from film
inner join skuespillerifilm on (film.FilmId = skuespillerifilm.FilmId)
inner join skuespiller on (skuespiller.SkuespillerID = skuespillerifilm.SkuespillerID)
group by film.FilmId
having avg(skuespiller.Fødselsår) > (
select avg(skuespiller.Fødselsår)
from skuespiller);
```

Oppgave 3: Flere spørringer i relasjonsalgebra

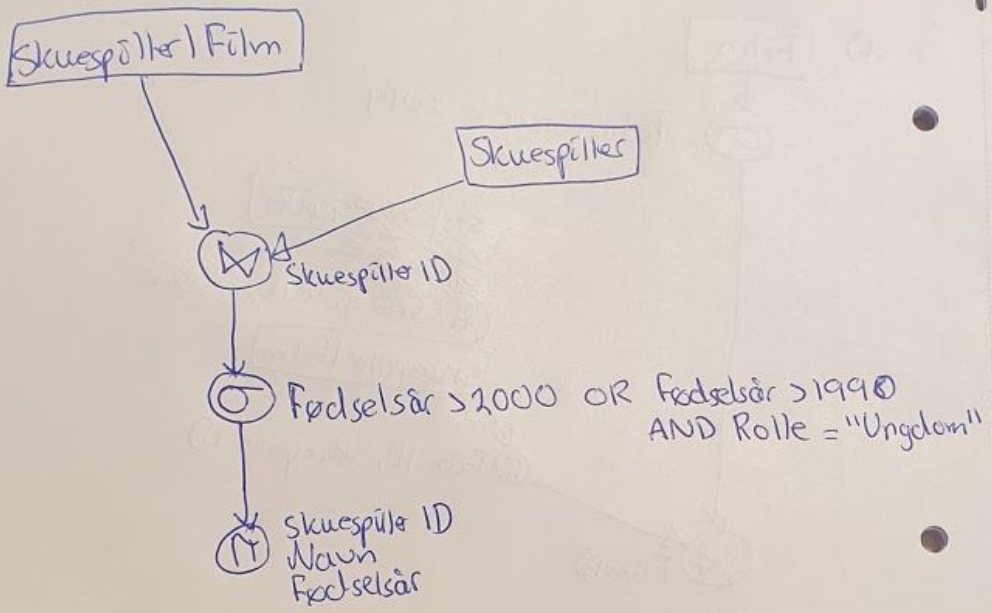
Lag spørringer i relasjonsalgebra som…

a) Finner skuespillerID og navn på de skuespillere som ikke har spilt i en film produsert etter 2014.
b) Henter skuespillerID, navn og fødselsår på skuespillerne som er født senere enn 2000, og de skuespillerne som er født senere enn 1990 og har spilt i en film som en ungdom (Rolle = “Ungdom”).
c) For hver skuespiller finner skuespillerID, navnet og antall filmer vedkommende har spilt i.

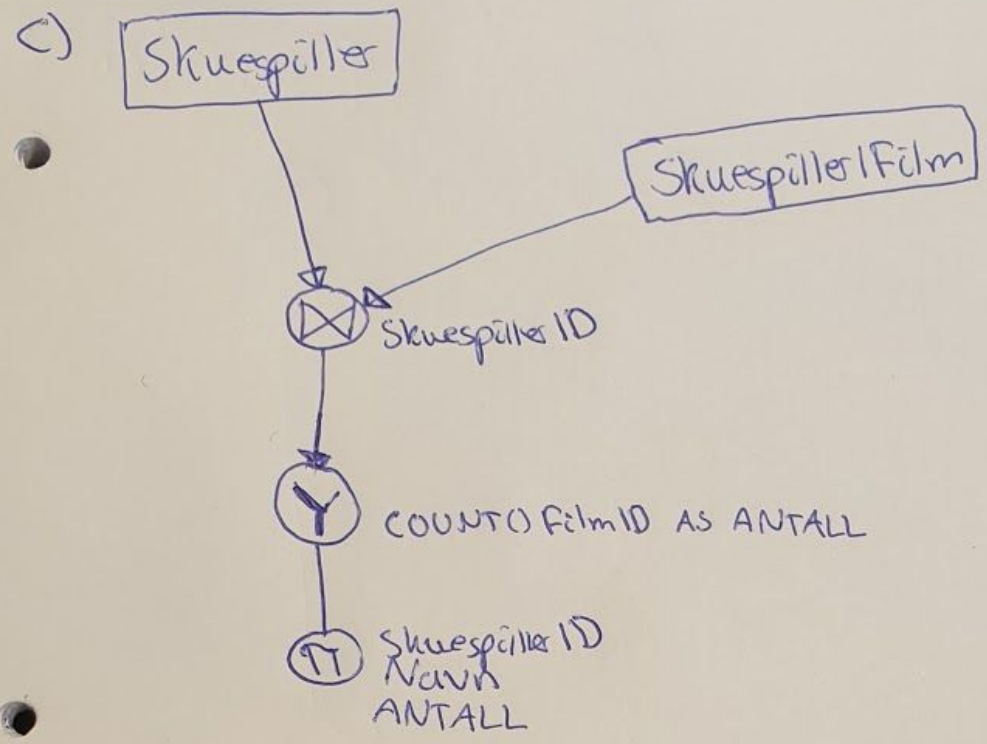
3 a)


Film
Produksjonsår < 2014
Skuespiller
Skuespiller IFilm
FilmID, SkuespillerID
FilmID
Skuespiller
Skuespiller ID, Navn
SkuespillerID
Skuespiller ID
Navn

b)


Skuespiller i Film
Skuespiller
Skuespiller ID
Fødselsår >2000 OR Fødselsår >1990
AND Rolle = "Ungdom"
Skuespiller ID
Navn
Fødselsår

c)


Skuespiller
Skuespiller1Film
Skuespiller ID
COUNT() FilmID AS ANTALL
Skuespiller ID
Navn
ANTALL

Oppgave 4: Introduksjon til normaliseringsteori
Betrakt følgende tabell med data:

a) Dette er et eksempel på dårlig design. Det er fordi lagrer vi mye av den samme informasjonen mange ganger (redundans), når vi sannsynligvis kan strukturere tabellen på en smartere måte for å unngå dette. Vi får også problemer med innsetting, oppdatering og sletting av data. Hvis vi for eksempel sletter en person, kan vi risikere å helt miste informasjon om et fakultet. Hvis fakultetet EDI endrer navn til DI (institutt for datainnovasjon), slik at både fakultetskoden og fakultetsnavnet må endres, hvor mange felter trenger å oppdateres?

10 felt må oppdateres

b) Kan du foreslå et alternativt design som gjør at vi trenger å oppdatere færre felter hvis EDI endrer navn? (Tips: Normaliseringsteorien presenterer som regel løsninger som innebærer å splitte en tabell opp i flere deltabeller. I dette tilfellet blander vi personinformasjon med fakultetsinformasjon.) Du kan anta at informasjon om fakultetsnavn og fakultetsbygg vil være like for samme verdier av Fakultetskode. Dvs. at vi har en funksjonell avhengighet Fakultetskode -> Fakultetsnavn, FakultetsBygg

PersonID	Navn	Telefon	Fakultetskode
270393	Ola Johansen	73735667	EDI
313874	Kari Vintermo	73739023	EDI
241257	Bernt Nilsen	73731234	EDI
935784	Olav Foss	73738471	EDI
345481	Per Høyder	73739021	EDI
134876	Åse Bekkerud	73746617	IØA

Fakultetskode	Fakultetsnavn	Fakultetsbygg
EDI	Institutt for energi og datainnovasjon	Oasen
IØA	Institutt for økonomiske arkitekturer	Solklossen

Fakultetskoden er fremmednøkkel i den andre tabellen.

Oppgave 5: Funksjonelle avhengigheter, nøkler og tillukning

a) Anta følgende tabell. Hvilke påstander kan umulig stemme? Begrunn svarene dine.

1) A -> A +
2) A -> B +
3) A -> C false fordi a1 går til c1 og c2
4) AB -> C false fordi a1 går til c1 og c2
5) C -> D +
6) D -> C false fordi d2 går til d1 og d4
7) ABCD er en supernøkkel for tabellen ABC er en minimal supernøkkel
8) ABC er en supernøkkel for tabellen +
9) A er en kandidatnøkkel for tabellen c er også en kandidatnøkkel
10) AC er en kandidatnøkkel for tabellen +

b) X+ kalles tillukningen av X og er mengden av alle attributter som er funksjonelt avhengig av X (basert på en mengde funksjonelle avhengigheter F).

Gitt tabellen R = {A, B, C, D} og F = {A -> C, B-> D, ABC -> D}. Finn A+, D+, BC+ og AB+.

A+ = AC
D+ = D
BC+ = BCD
AB + = ABCD