Übung 4

Aufgabe 4.1

In die Tabelle person haben sich Einträge mit identischem Feld name eingeschlichen:

```
CREATE TABLE person (
  nr INT PRIMARY KEY,
  name VARCHAR(30),
  geburt DATE,
  adresse1 VARCHAR(30),
  adresse2 VARCHAR(30)
);
```

Eine Testtabelle mit Inhalt können Sie über das Script ueb04-create.sql einspielen.

Realisieren Sie die folgenden Fragestellungen in SQL:

a) Ermitteln Sie alle Namen die mehrfach vorkommen.

```
-- Variante 1
SELECT name
FROM person
GROUP BY name
HAVING count(*) > 1;

-- Variante 2
SELECT DISTINCT *
from
 (select name
 from person

 EXCEPT ALL

 select distinct name
 from person) tmp
;

-- Variante 3
select distinct p1.name
from person p1 join person p2 on p1.name = p2.name
where p1.nr <> p2.nr;
```

b) Zeigen Sie zusätzlich bei den Namen an, wie oft sie vorkommen.

```
SELECT name, count(name)
FROM person
GROUP BY name
HAVING count(*) > 1;
```

c) Löschen Sie alle Namen die mehrfach vorkommen bis auf einen Namen. Hinweise: Wodurch unterscheiden sich Tupel mit gleichem Namen? Wie können Sie eines dieser Tupel auswählen?

Tupel mit gleichem Namen unterscheiden sich nur in nr. Lösche pro Namen alle Tupel außer das mit der jeweils höchsten nr.

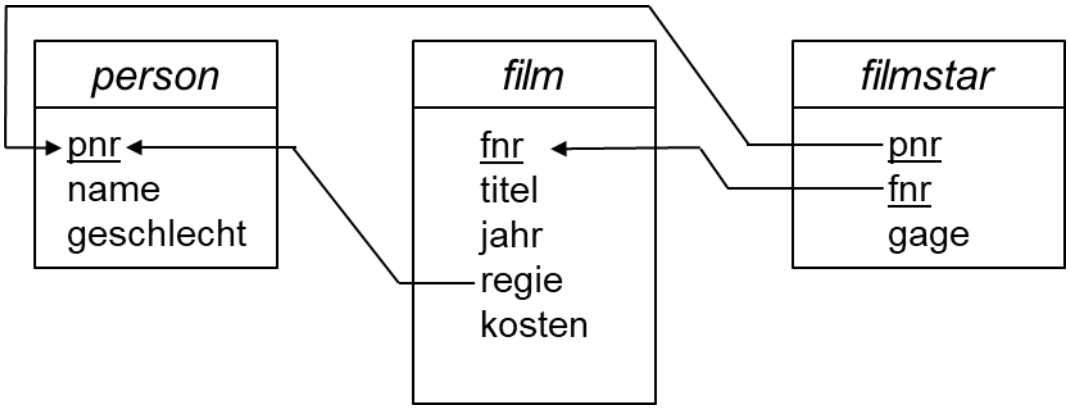
```
delete from person
where nr not in (
 select max(nr)
 from person
 group by name);
```

d) Können Sie Ihr "Doublettenlöschverfahren" verallgemeinern auf mehrere Spalten, z.B. auf doppelte Kombinationen name + geburt?

Ja, group by auf Attributkombination anwenden, also z.B: group by name, geburt

Aufgabe 4.2

Betrachten Sie das folgende Datenbankschema:



Geben Sie SQL-DDL Statements zum Anlegen folgender Views an:

a) Einen View `BilligFilm` mit Filmnr, Titel, Jahr, Regisseur, Kosten und Personnr der beteiligten Schauspieler aller Filme mit Kosten unter 100.000 Euro.

```
create view BilligFilm (FilmNr, Titel, Jahr, Regisseur, Kosten, PersonNr) as
select film.fnr, film.Titel, film.jahr, film.regie, film.kosten, filmstar.pnr
from film join filmstar on film.fnr = filmstar.fnr
where film.kosten < 100000;
```

b) Einen View `Regisseur` mit Personnr, Name und Geschlecht der Personen, die bei mindestens einem Film Regisseur waren.

```
create view regisseur as
select *
from person
where pnr in (select regie from film);
```

c) Einen View SuperStar mit Personnr, Name, Geschlecht und maximaler Gage der Personen, die für mindestens einen Film mehr als 500.000 Euro Gage bekommen haben.

```
create view SuperStar (pnr, name, geschlecht, maxgage) as
select person.*, max(filmstar.gage)
from person, filmstar
where person.pnr = filmstar.pnr
group by person.pnr
having max(filmstar.gage) > 500000;
```

Aufgabe 4.3

Formulieren Sie die folgenden Abfragen mithilfe der in Übung 4.2 definierten Views, ohne die Basistabellen zu verwenden:

a) Welche Superstars haben schon mal an einem Film mit Kosten unter 100.000 Euro mitgewirkt?

```
select * from superstar
where prn in ( select personnr from billigfilm);
```

b) Welche Person hat schon mal eine Gage über 1 Mio Euro erhalten?

```
select * from superstar where maxgage > 1000000;
```

c) Welche weiblichen Superstars waren auch als Regisseur tätig?

```
select * from superstar
where geschlecht = 'weiblich' and prn in (select pnr from regisseur);
```

Aufgabe 4.4

Welche der Views aus Übung 4.2 sind gemäß SQL2 änderbar? Wenn nicht: warum nicht?

a) Nicht updatable wegen Join.
b) Updatable, wenn über Subquery wie oben definiert, weil kein Selbstbezug in Subquery. Wenn aber über Join definiert, nicht updatable wegen Join.
c) Nicht updatable wegen Gruppierung und Aggregatfunktion.