Joins Part 1

1. Films directed by John Woo?

```
select m.*
from movies m
    join credits c
      on c.movieid = m.movieid
    join people p
      on p.peopleid = c.peopleid
where c.credited_as = 'D'
  and p.first_name = 'John'
  and p.surname = 'Woo'
```

2. List the cast of the American "Treasure Island" 1950 film (first name, surname, and display "Director" instead of "D" and "Actor" instead of "A"

```
select p.first_name, p.surname,
       case c.credited_as
         when 'A' then 'Actor'
         when 'D' then 'Director'
       end credited_as
from movies m
     join credits c
       on c.movieid = m.movieid
     join people p
       on p.peopleid = c.peopleid
where m.title = 'Treasure Island' -- Beware, for strict comparison
SQLite
  and m.country = 'us'            -- IS case sensitive
  and m.year_released = 1950;
```

3. Marilyn Monroe appeared in how many 1952 films?

```
select count(*)
from (select m.movieid
      from people p
           join credits c
             on c.peopleid = p.peopleid
           join movies m
             on m.movieid = c.movieid
      where c.credited_as = 'A'
        and p.first_name = 'Marilyn'
        and p.surname = 'Monroe'
        and m.year_released = 1952) a
```

4. Directors who have directed films in more than one country?

```
select p.first_name, p.surname
from (select c.peopleid
      from credits c
           join movies m
           on m.movieid = c.movieid
      where c.credited_as = 'D'
      group by c.peopleid
      having count(distinct m.country) > 1) x
    join people p
      on p.peopleid = x.peopleid;
```

5. Display the title, year and country code for films from 2010 or later for which the name of the director is missing from the database.

```
select m.title, m.year_released, m.country
from movies m
     left join credits c
       on c.movieid = m.movieid
      and c.credited_as = 'D'
where m.year_released >= 2010
  and c.movieid is null;   -- Any column from credits works
```

6. What are the title, country and year of release of remakes in the database ?
(films for which an earlier film with the same title exists)

```
select distinct r.title, r.country, r.year_released
from movies m
     join movies r -- remakes
       on r.title = m.title
     and r.year_released > m.year_released
```

distinct is required when there is more than one remake (otherwise the third film would appear as remake of the first one and as remake of the second one).

7. **List the names of the known directors of 2015 films (no need to display anything about the film). If the film is Chinese, Korean or Japanese, the name should be displayed as surname followed by first name, otherwise it must be first name followed by surname.**

Simplest version (concatenates to return a single name column)

```
select distinct   -- required, there might be two films for one
director
       case m.country
         when 'cn' then surname || ' ' || coalesce(first_name, ' ')
         when 'tw' then surname || ' ' || coalesce(first_name, ' ')
         when 'hk' then surname || ' ' || coalesce(first_name, ' ')
         when 'jp' then surname || ' ' || coalesce(first_name, ' ')
         when 'kr' then surname || ' ' || coalesce(first_name, ' ')
         else coalesce(first_name, ' ') || ' ' || surname
       end as director
from movies m
     join credits c
       on c.movieid = m.movieid
     join people p
       on p.peopleid = c.peopleid
where c.credited_as = 'D'
  and m.year_released = 2015;
```

Note that **`coalesce()`** is required in this version, otherwise all the directors who are only known by one name have a null row returned for them.

It's also possible to have two separate case ... end returning surname in one case, and first_name in the other, then the reverse ... but then naming the columns becomes a bit difficult.

8. **Longest film directed by a woman?**

```
select distinct
       m.title,
       m.country,
       m.year_released,
       m.runtime
from movies m
     join credits c
       on c.movieid = m.movieid
     join people p
       on p.peopleid = c.peopleid
where p.gender = 'F'
```

```
  and c.credited_as = 'D'
  and m.runtime =
     (select max(m.runtime) -- NULLs will be ignored
      from movies m
           join credits c
             on c.movieid = m.movieid
           join people p
             on p.peopleid = c.peopleid
      where p.gender = 'F'
        and c.credited_as = 'D')

or

with dfdw as  -- Detail of Films Directed by Women
(select m.title,
        m.country,
        m.year_released,
        m.runtime
 from (select distinct c.movieid
           -- DISTINCT because a film can be directed by several
women.
           -- Note that this DISTINCT is better than the one in query
           -- q2b, where it was applied to all the information returned
           -- for the films. Here we are sorting only identifiers, which
           -- means fewer bytes, less work, and faster - it could make
a
           -- significant difference on hundreds of millions of rows.
       from credits c
            join people p
              on p.peopleid = c.peopleid
       where p.gender = 'F'
         and c.credited_as = 'D') fdw -- Films Directed by Women
    join movies m
      on m.movieid = fdw.movieid
    where coalesce(runtime,0) > 0 -- Ignore anything for which we
have
                                  -- no data or zero
   )
select dfdw.*
from dfdw
where dfdw.runtime =
      (select max(runtime)
       from dfdw)
;
```

Longest film directed by a woman, take 3. This uses the "with" construct (unavailable in MySQL before MySQL 8, available everywhere else), often known as "common table expression" or CTE. The technique is also known as "query factorization". You basically give a name to a query, that you can use then at multiple places as if it were a table. Less typing for you, and the optimizer can make the choice of reinserting the text of the query everywhere you name it or get the result set once and reinject it wherever it is needed. The second option is particularly interesting if the query returns few rows after having scanned many, which can be the case with an aggregate, but it's the optimizer's decision, not yours. Naming a query isn't very useful if you use it only once, but it's reasonably frequent that you see the same bits of SQL several times in a query, like here or in UNION queries (similar subquery in several parts of the UNION)