

1. 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.

2. Films where you can find Humphrey Bogart and Lauren Bacall playing together?

```
select m.title, m.country, m.year_released
from (select c.movieid
      from (select peopleid
            from people
            where (first_name = 'Humphrey'
                  and surname = 'Bogart')
               or (first_name = 'Lauren'
                  and surname = 'Bacall')) famous_couple
      join credits c
        on c.peopleid = famous_couple.peopleid
```

```

        and c.credited_as = 'A'
    group by c.movieid
    having count(*) = 2) bogart_plus_bacall
join movies m
on m.movieid = bogart_plus_bacall.movieid;

```

3. How many times did John Wayne play in a John Ford film in the database?

```

select count(*)
from (select movieid
      from (select peopleid,
                    case surname
                     when 'Ford' then 'D'
                     else 'A'
                    end credited_as
            from people
            where first_name = 'John'
              and surname in ('Wayne', 'Ford')) wayne_ford
      join credits c
        on c.peopleid = wayne_ford.peopleid
        and c.credited_as = wayne_ford.credited_as
      group by movieid
      having count(distinct c.peopleid) = 2) by_ford_with_wayne
-- distinct because Ford might have
-- played AND directed and he might appear twice
-- count(c.peopleid) >= 2 could also work
;

```

4. Confusion between Western and Asian names. Display the peopleids and one surname and the matching surname as well as year of birth and year of death for rows in table people where birth year and death year (if set) are identical, and first_name and surname are swapped. They may be the same person entered twice by mistake.

```

select p1.peopleid,
       p2.peopleid,
       p1.first_name,
       p1.surname,
       p1.born,

```

```

    p1.died
from people p1
  join people p2
  on p2.first_name = p1.surname
  and p2.surname = p1.first_name
  and p2.born = p1.born
  and coalesce(p2.died, 0) = coalesce(p1.died, 0)
  and p2.peopleid > p1.peopleid -- to avoid duplicates

```

- 5. Display first name, surname, year of death and year of their last film for actors who died more than 20 years after the last film we have with them in the database.**

```

select p.first_name, p.surname, a.last_film, p.died
from (select c.peopleid, max(m.year_released) last_film
      from movies m
      join credits c
      on c.movieid = m.movieid
      where credited_as = 'A'
      group by c.peopleid) a
join people p
  on p.peopleid = a.peopleid
where p.died > 20 + a.last_film

```

- 6. What is in the database the first film in which Jackie Chan starred?**

```

select m.title, m.year_released, m.country
from (select c.peopleid, min(m.year_released) first_film_year
      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 = 'Jackie'
      and p.surname = 'Chan') a
join credits c
  on c.peopleid = a.peopleid
  and c.credited_as = 'A'
join movies m
  on m.movieid = c.movieid

```

```
and m.year_released = a.first_film_year
```

- 7. List the first name and surname, as well as the number of films by Orson Welles where they appear, of all actors, other than Orson Welles himself, who played in an Orson Welles film.**

```
select p.first_name, p.surname, count(*) films
from (select p.peopleid ow, c.movieid
      from people p
        join credits c
          on c.peopleid = p.peopleid
      where c.credited_as = 'D'
        and p.first_name = 'Orson'
        and p.surname = 'Welles') ow_films
join credits c
  on c.movieid = ow_films.movieid
 and c.credited_as = 'A'
 and c.peopleid <> ow_films.ow
join people p
  on p.peopleid = c.peopleid
group by p.first_name, p.surname
```

- 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)

Set operators

9. List all year and "Events" (films released time, people births time, people deaths time) that occurred between 1930 and 1935

```
SELECT m.year_released AS year,  
       m.title || ' (' || c.country_name || ') was released' AS event  
FROM movies m  
JOIN  
     countries c ON c.country_code = m.country  
WHERE m.year_released BETWEEN 1930 AND 1935  
UNION ALL  
SELECT born,  
       trim(coalesce(first_name, '') || ' ' || surname || ' was born')  
FROM people  
WHERE born BETWEEN 1930 AND 1935  
UNION ALL  
SELECT died,  
       trim(coalesce(first_name, '') || ' ' || surname || ' died')  
FROM people  
WHERE died BETWEEN 1930 AND 1935  
ORDER BY year;
```

10. Same as question1, pushed into a subquery to add a sort key

```
SELECT year, event  
FROM (  
    SELECT m.year_released AS year,  
           m.title || ' (' || c.country_name || ') was released' AS event,  
           m.title AS sort_key
```

```

        FROM movies m
        JOIN
        countries c ON c.country_code = m.country
        WHERE m.year_released BETWEEN 1930 AND 1935
    UNION ALL
    SELECT born,
        trim(coalesce(first_name, "") || ' ' || surname || ' was born'),
        surname AS sort_key
    FROM people
    WHERE born BETWEEN 1930 AND 1935
    UNION ALL
    SELECT died,
        trim(coalesce(first_name, "") || ' ' || surname || ' died'),
        surname AS sort_key
    FROM people
    WHERE died BETWEEN 1930 AND 1935
    )
    x
ORDER BY year,sort_key;

```

11. Same as before, more sophisticated sort_key

```

SELECT year,
    event
FROM (
    SELECT m.year_released AS year,
        m.title || ' (' || c.country_name || ') was released' AS event,
        trim([replace](m.title, 'The', '')) AS sort_key
    FROM movies m
    JOIN
    countries c ON c.country_code = m.country
    WHERE m.year_released BETWEEN 1930 AND 1935
    UNION ALL
    SELECT born,
        trim(coalesce(first_name, "") || ' ' || surname || ' was born'),
        surname AS sort_key
    FROM people
    WHERE born BETWEEN 1930 AND 1935
    UNION ALL

```

```

        SELECT died,
               trim(coalesce(first_name, '') || ' ' || surname || ' died'),
               surname AS sort_key
        FROM people
        WHERE died BETWEEN 1930 AND 1935
    )
    x
ORDER BY year,sort_key;

```

12. Events that happened the year when the earliest "Devdas" was released

```

WITH earliest_devdas AS (
    SELECT min(year_released) AS year
    FROM movies
    WHERE title = 'Devdas'
)
SELECT m.year_released AS year,
       m.title || ' (' || c.country_name || ') was released' AS event
FROM movies m
JOIN
    countries c ON c.country_code = m.country
WHERE m.year_released = (
                        SELECT year
                        FROM earliest_devdas
                    )
UNION ALL
SELECT born,
       trim(coalesce(first_name, '') || ' ' || surname || ' was born')
FROM people
WHERE born = (
            SELECT year
            FROM earliest_devdas
        )
UNION ALL
SELECT died,
       trim(coalesce(first_name, '') || ' ' || surname || ' died')

```



```
FROM people
WHERE died = (
    SELECT year
    FROM earliest_devdas
);
```

13. Films where Qi Shu played without Ge You. Illustrates that "except" isn't really necessary

```
SELECT m.title,
       m.country,
       m.year_released
FROM (
    SELECT c.movieid
    FROM credits c
    JOIN
        people p ON p.peopleid = c.peopleid
    WHERE p.first_name = 'Shu' AND
          p.surname = 'Qi' AND
          c.credited_as = 'A'
    EXCEPT
    SELECT c.movieid
    FROM credits c
    JOIN
        people p ON p.peopleid = c.peopleid
    WHERE p.first_name = 'You' AND
          p.surname = 'Ge' AND
          c.credited_as = 'A'
)
x
JOIN
    movies m ON m.movieid = x.movieid
ORDER BY m.year_released;

-- or

SELECT m.title, m.country, m.year_released
```

```

FROM (
    SELECT c.movieid
    FROM credits c
    JOIN
        people p ON p.peopleid = c.peopleid
    WHERE p.first_name = 'Shu' AND
        p.surname = 'Qi' AND
        c.credited_as = 'A' AND
        c.movieid NOT IN (
            SELECT c.movieid
            FROM credits c
            JOIN
                people p ON p.peopleid = c.peopleid
            WHERE p.first_name = 'You' AND
                p.surname = 'Ge' AND
                c.credited_as = 'A'
        )
    )x
JOIN
    movies m ON m.movieid = x.movieid
ORDER BY m.year_released;
-- or

SELECT m.title, m.country, m.year_released
FROM (
    SELECT c.movieid
    FROM credits c
    JOIN
        people p ON p.peopleid = c.peopleid
    WHERE p.first_name = 'Shu' AND
        p.surname = 'Qi' AND
        c.credited_as = 'A' AND
        NOT EXISTS (
            SELECT NULL
            FROM credits c2
            JOIN
                people p2 ON p2.peopleid = c2.peopleid
            WHERE p2.first_name = 'You' AND

```

```

        p2.surname = 'Ge' AND
        c2.credited_as = 'A' AND
        p2.peopleid = c2.peopleid AND
        c2.movieid = c.movieid
    )
)x
JOIN
    movies m ON m.movieid = x.movieid
ORDER BY m.year_released;

-- or

SELECT m.title,m.country, m.year_released
FROM (
    SELECT c.movieid
    FROM credits c
    JOIN
        people p ON p.peopleid = c.peopleid
    LEFT OUTER JOIN
        (
            SELECT c2.movieid
            FROM credits c2
            JOIN
                people p2 ON p2.peopleid = c2.peopleid
            WHERE p2.first_name = 'You' AND
                p2.surname = 'Ge' AND
                c2.credited_as = 'A' AND
                p2.peopleid = c2.peopleid
        )
        y ON y.movieid = c.movieid
    WHERE p.first_name = 'Shu' AND
        p.surname = 'Qi' AND
        c.credited_as = 'A' AND
        y.movieid IS NULL
)
x
JOIN
    movies m ON m.movieid = x.movieid

```

```

ORDER BY m.year_released;
-- or

SELECT m.title, m.country, m.year_released
FROM (
    SELECT c.movieid
    FROM credits c
    JOIN
        people p ON p.peopleid = c.peopleid
    WHERE (p.first_name = 'Shu' AND
        p.surname = 'Qi') OR
        (p.first_name = 'You' AND
        p.surname = 'Ge') AND
        c.credited_as = 'A'
    GROUP BY c.movieid
    HAVING count( * ) = 1 AND
        min(surname) = 'Qi'
)x
-- If the min is Qi, there is no Ge
JOIN
    movies m ON m.movieid = x.movieid
ORDER BY m.year_released;

```

[Recursive](#)

The chain of life ...

```

WITH q (surname,first_name,born,died)
AS (
    SELECT surname,first_name,born,died
    FROM people
    WHERE surname = 'Qi' AND
        first_name = 'Qiqiu'
    UNION ALL
    SELECT p.surname,p.first_name,p.born,p.died
    FROM people p
    JOIN
        q ON p.born = q.died)
-- Note that we get duplicates without distinct,
/* as several people have exactly the same lifespan */

```

```
SELECT DISTINCT *  
FROM q  
ORDER BY born,surname;
```