Queries on a single table without any aggregate

NOTE: joins not seen yet. It's OK to check table countries to find the codes to use in a query on movies.

Table alt_titles contains some titles in Chinese (in a column called 'title')

1. **List all films from Portugal or Brazil**

(Same language spoken in both countries)

```
select * from movies
where country = 'br'
   or country = 'pt'
```

or

```
select * from movies
where country in ('br', 'pt')
```

2. **American films that were released the year you were born**

```
select * from movies
where country = 'us'
  and year_released = ...
```

3. **Spanish films that contain neither 'a' (in any case) nor 'o' (in any case) in their title**

```
select * from movies
where country = 'sp'
  and lower(title) not like '%o%'
  and lower(title) not like '%a%'
```

or

```
select * from movies
where country = 'sp'
  and not (lower(title) like '%o%' or lower(title) like '%a%')
```

(note that SQLite isn't case sensitive with **like**, but Oracle or PostgreSQL would require forcing the case)

4. **List all Chinese films from the 1940s**

```
select * from movies
where (country = 'cn' or country = 'hk' or country = 'tw' or country = 'mo')
  and year_released between 1940 and 1949
```

or

```
select * from movies
where country in ('cn', 'hk', 'tw', 'mo')
  and year_released between 1940 and 1949
```

(no Macau (mo) film in the database but there might be one)

Things to point out:

- Parentheses with **or**, otherwise wrong result (already shown during the lecture, worth repeating)
- **between** includes the boundaries. `between 1940 and 1950` is wrong because there are a few films from 1950.
- Alternatively for the year:

  year_released >= 1940 and year_released < 1950 (or **<= 1949**)

  Fancier way of writing it:

  cast(year_released as varchar) like '194_'

  (add that changing the type of data on the fly can be very bad for performance; why will be explained later in the course)

5. **List of all people who where born in 1920 or earlier and are still (according to the database) alive**

```
select * from people where born <= 1920 and died is null;
```

6. **List all Chinese titles (in table alt_titles) that contain the character for "mountain"**

```
select * from alt_titles where title like '%山%';
```

7. **List all titles that contain the word "man".**

Naive answer:

```
select * from movies where title like '%man%'
```

This will return about everything except what we want - for instance films containing "Woman", "Batman", etc.

Point out:

* Case sensitivity: uppercase not the same as lowercase (beware, depends on the DBMS). Normally, all words are capitalized but you can never be sure.

* Some products provide regular expressions for searches but it's not standard

* Isolated word is either preceded by a space, or followed by a space, or preceded by nothing, or followed by nothing. It can also be followed by a quote (for instance "Dead Man's Chest")!

So:

A little better:

```
select * from movies where title like '%Man%'
```

Complete answer:

```
select * from movies
where upper(title) like 'MAN %'
   or upper(title) like 'MAN''%'
   or upper(title) like '% MAN %'
   or upper(title) like '% MAN''%'
   or upper(title) like '% MAN';
```

8. **Display the names of the people in the database who died aged 100 or more**

```
select * from people where died >= born + 100;
```

9. **Same question as the previous one but include people who are (according to the database) currently one hundred or more**

```
select * from people
where died >= born + 100
 or (died is null and born <= 1918);
```

(point out that coding "1918" in the query isn't too good; there are functions for finding the current year, functions will be discussed later)

10. **Who are the people with a surname that contains a quote**

```
select * from people where surname like '%''%';
```

11. What are the European countries with a code that starts with a c like China?

```
select * from countries where continent = 'EUROPE' and country_code like
'c%'
```

'c%' or **'c_'**

12. List the people in the database who have the same first character in their first and last names. (For instance, Charlie Chaplin.)

```
select * from people where surname like substr(first_name,1,1)||'%'
```

13. List the title, year and runtime for Indian films that have a runtime of 2 hours or less.

```
select title, year_released, runtime
from movies
where country='in' and runtime <= 120;
```

Queries on a single table - aggregate/distinct

14. Number of countries per continent

```
select continent, count(*) from countries group by continent;
```

15. Age of the youngest lady in the people table

```
select 2019 - max(born) from people where gender='F';
```

Opportunity for explaining that in a program you cannot change the date every year, and that you can use functions:

```
select cast(substr(current_date, 1, 4) as int) - max(born) from people;
```

16. Average age at death (rounded) per gender

```
select gender, round(avg(died - born))
from people
where died is not null
group by gender
```

17. Number of films per country code for countries that have a code that starts with 'm'.

```
select country, count(*)
from movies
```

```
where country like 'm%'
group by country;
```

18. There are films from how many countries/territories in the database?

```
select count(distinct country) from movies;
```

19. Year of release of the oldest Chinese film in the database?

```
select min(year_released)
from movies
where country in ('cn', 'tw', 'hk');
```

20. How many films in the database for 2010?

```
select count(*) from movies where year_released = 2010;
```

21. What is the average number of films that we have per year, since 1960 (included)?

```
select year_released, count(*)
from movies
where year_released >= 1960
group by year_released;
```

22. How many British films in 1965?

```
select count(*) from movies where country='gb' and year_released=1965;
```

23. Average number of actors per film

```
select avg(cnt)
from (select movieid, count(*) cnt
      from credits
      where credited_as='A'
      group by movieid) x;
```

Can be improved with **`round(avg(cnt))`**

24. Number of films per number of directors (how many films have one director, how many have 2, etc. - in a single query)

```
select directors, count(movieid) films
from (select movieid, count(*) directors
      from credits
      where credited_as='D'
      group by movieid) dir_cnt
group by directors;
```

Note that films for which no director is known don't appear. They could be added but using techniques not seen yet (**union**)

25. **On the same row, return how many people are recorded in the database, how many are alive and how many are dead.**

```
select count(*),
       count(died) dead, -- null not counted
       sum(case
           when died is null then 1
           else 0
       end) alive
from people;
```

There are sometimes functions other than **case ... end** that can be used (**ifnull()**, **decode()**), but **case ... end** is standard, other functions aren't.

26. **The most common surname in the database can be found how many times?**

```
select max(cnt)
from (select surname, count(*) cnt
      from people
      group by surname) name_count;
```

27. **How many people have played in a film that they have directed?**

```
select count(*)
from (select c.peopleid
      from credits c
      where c.credited_as in ('A', 'D')
      group by movieid, peopleid
      having count(*) = 2) played_and_directed;
```

28. **What is (rounded to the first decimal) the percentage of women in the database?**

```
select round(100 * sum(case gender
                       when 'F' then 1
                       else 0 end) / count(*), 1) percentage_of_women
from people;
```

29. **Display country code and number of films with a length of 3 hours or more for all countries (if there is no 3+ hours film, the country doesn't need to be shown)**

```
select country, count(*)
from movies
where runtime >= 180
group by country
```