1. Write a query to return a table of genres with the oldest and newest movie in that genre.
E.g.
Genre, Oldest, Newest
Action, The Road Warrior, Casino Royale

Schema (PostgreSQL v13)

CREATE TABLE movies (genre TEXT, title TEXT, year INT);

INSERT INTO movies VALUES

('Action', 'The Road Warrior', 1982),
('Action', 'Aliens', 1986),
('Action', 'Die Hard', 1988),
('Action', 'Terminator 2: Judgment Day', 1991),
('Action', 'Casino Royale', 2006),
('Adventure', 'King Kong', 1933),
('Adventure', 'The Adventures of Robin Hood', 1938),
('Adventure', 'The Treasure of the Sierra Madre', 1948),
('Adventure', 'Raiders of the Lost Ark', 1981),
('Adventure', 'Jurassic Park', 1993),
('Animated', 'Snow White and the Seven Dwarfs', 1937),
('Animated', 'Fantasia', 1940),
('Animated', 'Beauty and the Beast', 1991),
('Animated', 'Toy Story', 1995),
('Animated', 'Spirited Away', 2001),
('Biopic', 'Lawrence of Arabia', 1962),
('Biopic', 'Patton', 1970),
('Biopic', 'Amadeus', 1984),
('Biopic', 'Malcolm X', 1992),
('Biopic', 'Lincoln', 2012);

Query #1

```
select tab.genre, "Oldest", "Newest"
from
(Select genre, title "Oldest" from movies
where year in (select min(year) from movies group by genre))
tab Inner Join
(Select genre, title "Newest" from movies
where year in (select Max(year) from movies group by genre))
newtab
On tab.genre = newtab.genre;
```

genre	Oldest	Newest
Action	The Road Warrior	Casino Royale
Adventure	King Kong	Jurassic Park
Animated	Snow White and the Seven Dwarfs	Spirited Away
Biopic	Lawrence of Arabia	Lincoln

[View on DB Fiddle](https://www.db-fiddle.com/f/82o6KQDLfeU4iFkfMqu2LC/5)

2. Write a query to get the number of movies broken down by decade.

Schema (PostgreSQL v13)

CREATE TABLE movies (genre TEXT, title TEXT, year INT);

INSERT INTO movies VALUES

('Action', 'The Road Warrior', 1982),
('Action', 'Aliens', 1986),
('Action', 'Die Hard', 1988),
('Action', 'Terminator 2: Judgment Day', 1991),
('Action', 'Casino Royale', 2006),
('Adventure', 'King Kong', 1933),
('Adventure', 'The Adventures of Robin Hood', 1938),
('Adventure', 'The Treasure of the Sierra Madre', 1948),
('Adventure', 'Raiders of the Lost Ark', 1981),
('Adventure', 'Jurassic Park', 1993),
('Animated', 'Snow White and the Seven Dwarfs', 1937),
('Animated', 'Fantasia', 1940),
('Animated', 'Beauty and the Beast', 1991),
('Animated', 'Toy Story', 1995),
('Animated', 'Spirited Away', 2001),
('Biopic', 'Lawrence of Arabia', 1962),
('Biopic', 'Patton', 1970),
('Biopic', 'Amadeus', 1984),
('Biopic', 'Malcolm X', 1992),
('Biopic', 'Lincoln', 2012);

Query #2

Select year/10* 10 as Decade, count (title)
From movies
Group By year/ 10* 10
Order by decade Desc;

decade	count
2010	1
2000	2
1990	5
1980	5
1970	1
1960	1
1940	2
1930	3

[View on DB Fiddle](https://www.db-fiddle.com/f/82o6KQDLfeU4iFkfMqu2LC/0)

3. That data came from https://www.filmsite.org/top100filmsbygenre.html.
How would you extract the rest of the 100 movies from that page and transform into the SQL DML like the above?
No need to implement anything; a detailed description is enough.

Answer: I can use web scraping although this is not legal for all website. For the web scraping, I will use Python libraries requests, HTML parser library and Beautiful Soup, following the below steps:

1. I will use Python libary request to send an HTTP resuest to the URL of the webpage, the server will respond to the request returning HTML content of the webpage.
2. I will use a HTML parser e.g. html5lib to create a nested/tree structure of the returned HTML data

3. Next, I will use Beautiful soup to navigate, search and extract the required data from the HTML content

4. Lastly, I will save the data in csv file where I can then use SQL to import the data and query them in any database.

Another method which is usually Legal is using API issued from the website providers. This allow access to the data in a predefined manner; With APIs, I avoid parsing HTML and then get access and extract the data directly using formats like JSON. Afterwards, I can load the data in a database and then query it using SQL.