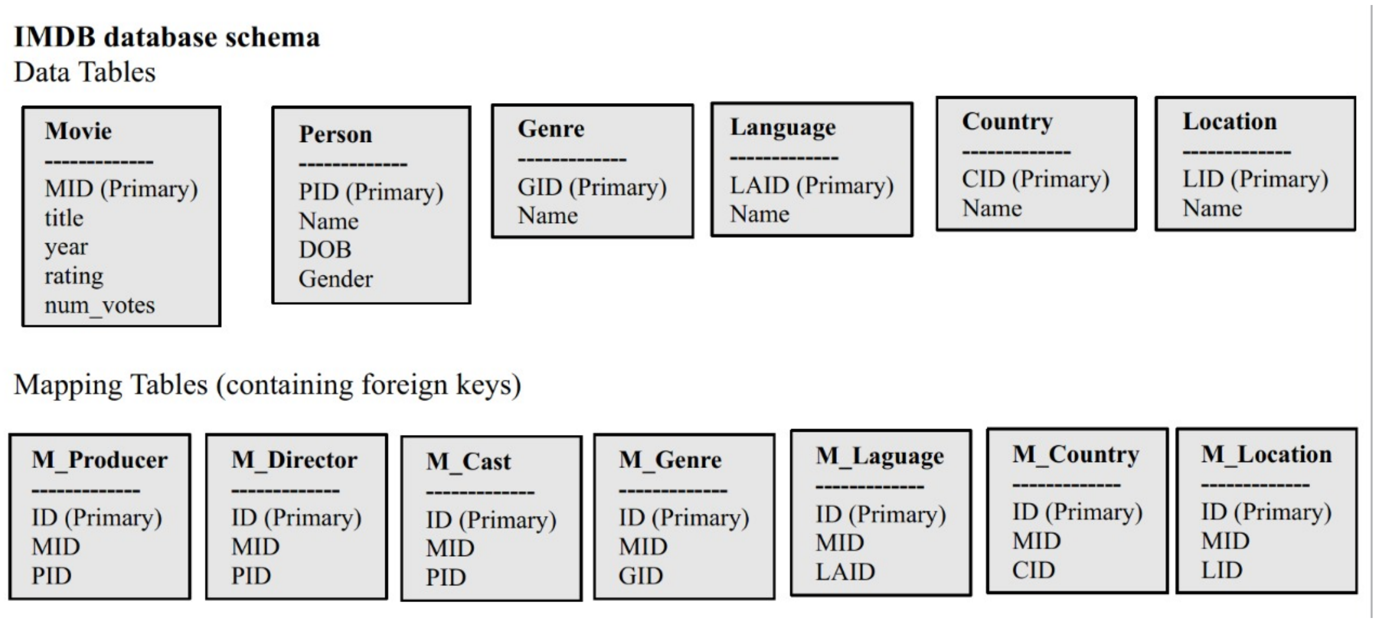
Import libraries and creating connection to the database

In [5]:

```
import sqlite3
import pandas as pd
conn = sqlite3.connect('db-imdb.db')
conn
```

Out[5]:

```
<sqlite3.Connection at 0x20387dc7b90>
```



1. List all the directors who directed a 'Comedy' movie in a leap year. (You need to check that the genre is 'Comedy’ and year is a leap year) Your query should return director name, the movie name, and the year.

In [4]:

```
r = pd.read_sql_query("""select Person.Name as directors, Genre.Name as genre, title
                         as movie_title, year from Person join M_director using(PID)
                         join movie using(MID) join M_Genre using (MID) join genre using
                         (GID) where Genre.Name = 'Comedy' and year % 4 = 0;""",conn)
r
```

Out[4]:

	directors	genre	movie_title	year
0	Bhagyaraj	Comedy	Mr. Bechara	1996
1	Pankaj Parashar	Comedy	Ab Ayega Mazaa	1984
2	Mahesh Bhatt	Comedy	Papa Kahte Hain	1996
3	Jabbar Patel	Comedy	Ek Hota Vidushak	1992
4	Kawal Sharma	Comedy	Maalamaal	1988
5	Srinivas Bhashyam	Comedy	Paisa Vasool	2004
6	Raj Kaushal	Comedy	Shaadi Ka Laddoo	2004
7	Siddharth Anand Kumar	Comedy	Let's Enjoy	2004
8	Govind Menon	Comedy	Kis Kis Ki Kismat	2004
9	Sachin	Comedy	Navra Mazha Navsacha	2004
10	Karan Razdan	Comedy	Mr Bhatti on Chutti	2012
11	Sachin Yardi	Comedy	Kyaa Super Kool Hain Hum	2012
12	Sameer Sharma	Comedy	Luv Shuv Tey Chicken Khurana	2012
13	Anand Balraj	Comedy	Daal Mein Kuch Kaala Hai	2012
14	Rajnish Thakur	Comedy	Mere Dost Picture Abhi Baaki Hai	2012
15	Vickrant Mahajan	Comedy	Challo Driver	2012
16	Jagdish Rajpurohit	Comedy	Bumboo	2012
17	Rakesh Mehta	Comedy	Life Ki Toh Lag Gayi	2012
18	Nitin Kakkar	Comedy	Filmistaan	2012
19	Aditya Datt	Comedy	Will You Marry Me	2012
20	Milap Zaveri	Comedy	Mastizaade	2016
21	Umesh Ghadge	Comedy	Kyaa Kool Hain Hum 3	2016
22	Abhishek Sharma	Comedy	Tere Bin Laden: Dead Or Alive	2016
23	Sanjeev Sharma	Comedy	Saat Uchakkey	2016
24	Krishnadev Yagnik	Comedy	Days of Tafree	2016
25	Suhas Kadav	Comedy	Motu Patlu: King of Kings	2016

2. List the names of all the actors who played in the movie 'Anand' (1971)

In [5]:

```
r = pd.read_sql_query("""select  title, Name, year from movie join m_cast using(MID)
                         INNER join Person on TRIM(Person.PID)=TRIM(M_cast.PID)
                         where title = 'Anand' and year = '1971'""",conn)
r
```

Out[5]:

```
    title              Name  year
0   Anand  Amitabh Bachchan  1971
1   Anand     Rajesh Khanna  1971
2   Anand     Sumita Sanyal  1971
3   Anand        Ramesh Deo  1971
4   Anand         Seema Deo  1971
5   Anand    Asit Kumar Sen  1971
6   Anand        Dev Kishan  1971
7   Anand      Atam Prakash  1971
8   Anand      Lalita Kumari  1971
9   Anand            Savita  1971
10  Anand    Brahm Bhardwaj  1971
11  Anand      Gurnam Singh  1971
12  Anand      Lalita Pawar  1971
13  Anand       Durga Khote  1971
14  Anand        Dara Singh  1971
15  Anand     Johnny Walker  1971
16  Anand         Moolchand  1971
```

3. List all the actors who acted in a film before 1970 and in a film after 1990. (That is: < 1970 and > 1990.)

In [4]:

```
r = pd.read_sql_query("""select distinct Person.Name, movie.year from Person inner
                         join M_Cast on TRIM(Person.PID)=TRIM(M_cast.PID)
                         join Movie using(MID) where year BETWEEN 1970 and 1990""",conn)
r
```

Out[4]:

	Name	year
0	Richard Dreyfuss	1977
1	François Truffaut	1977
2	Teri Garr	1977
3	Melinda Dillon	1977
4	Bob Balaban	1977
...	...	...
10026	Chandrashekhar	1986
10027	Prem Sagar	1986
10028	Kamal Kapoor	1986
10029	Biswajeet	1986
10030	Asif	1986

10031 rows × 2 columns

4. List all directors who directed 10 movies or more, in descending order of the number of movies they directed. Return the directors' names and the number of movies each of them directed.

In [5]:

```
r = pd.read_sql_query("""select Name, count(MID) from Person join M_Director using(PID)
                          group by PID having count(MID) >= 10 order by count(MID) desc""",
r
```

Out[5]:

	Name	count(MID)
0	David Dhawan	39
1	Mahesh Bhatt	35
2	Ram Gopal Varma	30
3	Priyadarshan	30
4	Vikram Bhatt	29
5	Hrishikesh Mukherjee	27
6	Yash Chopra	21
7	Shakti Samanta	19
8	Basu Chatterjee	19
9	Subhash Ghai	18
10	Rama Rao Tatineni	17
11	Abbas Alibhai Burmawalla	17
12	Shyam Benegal	17
13	Raj N. Sippy	16
14	Gulzar	16
15	Manmohan Desai	16
16	Mahesh Manjrekar	15
17	Raj Kanwar	15
18	Rajkumar Santoshi	14
19	Rahul Rawail	14
20	Raj Khosla	14
21	Indra Kumar	14
22	K. Raghavendra Rao	13
23	Ananth Narayan Mahadevan	13
24	Anurag Kashyap	13
25	Harry Baweja	13
26	Vijay Anand	13
27	Dev Anand	13
28	Rakesh Roshan	13
29	Rohit Shetty	12
30	Madhur Bhandarkar	12
31	Anil Sharma	12
32	Umesh Mehra	12
33	Prakash Mehra	12

	Name	count(MID)
34	Nagesh Kukunoor	12
35	Satish Kaushik	12
36	Prakash Jha	12
37	Guddu Dhanoa	12
38	Anees Bazmee	12
39	Mohit Suri	11
40	Govind Nihalani	11
41	Ketan Mehta	11
42	Nasir Hussain	11
43	Sanjay Gupta	11
44	Pramod Chakravorty	11
45	Bimal Roy	10
46	J. Om Prakash	10
47	Pankaj Parashar	10
48	K. Muralimohana Rao	10
49	Sudhir Mishra	10
50	Hansal Mehta	10
51	Mehul Kumar	10
52	J.P. Dutta	10
53	Tigmanshu Dhulia	10
54	N. Chandra	10
55	Vishal Bhardwaj	10
56	K. Bapaiah	10
57	Raj Kapoor	10

5a. For each year, count the number of movies in that year that had only female actors.

5b. Now include a small change: report for each year the percentage of movies in that year with only female actors, and the total number of movies made that year.

For example, one answer will be: 1990 31.81 13522 meaning that in 1990 there were 13,522 movies, and 31.81% had only female actors. You do not need to round your answer.

In [2]:

```
r = pd.read_sql_query('''SELECT
                        first_table.id,
                        second_table.Female_Count,
                        first_table.title,
                        first_table.year
                    FROM
                        (SELECT
                            movie.mid AS id,
                            person.gender AS gender,
                            movie.year AS year,
                            person.name AS name,
                            movie.title as Title,
                            COUNT(person.pid) as total_count
                        FROM
                            movie
                            JOIN m_cast ON movie.mid=TRIM(m_cast.mid)
                            JOIN person ON person.pid=TRIM(m_cast.pid)
                        GROUP BY Title) AS first_table
                        JOIN
                        (SELECT
                            movie.mid AS id,
                            person.gender AS gender,
                            person.name AS name,
                            movie.title as Title,
                            COUNT(person.pid) as Female_Count
                        FROM
                            movie
                            JOIN m_cast ON movie.mid=TRIM(m_cast.mid)
                            JOIN person ON person.pid=TRIM(m_cast.pid)
                            WHERE gender='Female'
                        GROUP BY Title) AS second_table
                        ON first_table.id = second_table.id
                    WHERE first_table.total_count = second_table.Female_Count
                    GROUP BY first_table.year''',conn)
r
```

Out[2]:

	id	Female_Count	Title	year
0	tt0375882	1	Kala Jigar	1939
1	tt0272001	11	Bindhaast	1999
2	tt0354922	10	Snegithiye	2000
3	tt8458202	2	Pihu	2018

In [4]:

```
r = pd.read_sql_query("""
                     SELECT
                         firstTable.Year AS YEAR,
                         secondTable.total_movie_count AS total_movie_count,
                         firstTable.total_movie_count_with_female_cast_only,
                         (CAST(firstTable.total_movie_count_with_female_cast_only AS FLOAT)
                     FROM (
                         SELECT
                             m.Year,
                             COUNT(*)  AS total_movie_count_with_female_cast_only
                         FROM Movie m WHERE m.mid IN
                            (SELECT DISTINCT trim(m_c.mid) FROM m_cast m_c WHERE TRIM(m_c.m
                                (SELECT DISTINCT trim(m_c.mid) FROM m_cast m_c WHERE TRIM(m
                                    (SELECT p.pid FROM Person p WHERE LOWER(p.gender) != 'f
                                )
                            ) GROUP BY m.year) AS firstTable
                            JOIN
                            (SELECT m.year AS year,
                                    COUNT(*) total_movie_count
                            FROM Movie m GROUP BY m.year) AS secondTable
                            ON firstTable.year = secondTable.year
""",conn)
r
```

Out[4]:

	YEAR	total_movie_count	total_movie_count_with_female_cast_only	female_only_cast_percent
0	1939	2	1	50.000
1	1999	66	1	1.515
2	2000	64	1	1.562
3	2009	110	1	0.909
4	2012	111	1	0.900
5	2018	104	2	1.923

6. Find the film(s) with the largest cast. Return the movie title and the size of the cast. By "cast size" we mean the number of distinct actors that played in that movie: if an actor played multiple roles, or if it simply occurs multiple times in casts, we still count her/him only once.

In [6]:

```
r = pd.read_sql_query("select title, count(distinct(PID)) as cast_size from Movie join M_Ca
r
```

Out[6]:

	title	cast_size
0	Ocean's Eight	238

7. A decade is a sequence of 10 consecutive years. For example, say in your database you have movie information starting from 1965. Then the first decade is 1965, 1966, ..., 1974; the second one is 1967, 1968, ..., 1976 and so on. Find the decade D with the largest number of films and the total number of films in D.

In [5]:

```
r = pd.read_sql_query('''SELECT m_y.year AS Decade_Start,
                              m_y.year + 9 as Decade_End,
                              COUNT(*) AS Number_Of_Movies
                              FROM (SELECT DISTINCT(year) FROM movie) m_y
                              JOIN movie m
                              ON m.year >= m_y.year AND m.year < m_y.year + 10
                              GROUP BY m_y.year
                              ORDER BY count(*) desc LIMIT 1
                              ''', conn)
r
```

Out[5]:

	Decade_Start	Decade_End	Number_Of_Movies
0	2008	2017	1205

8. Find the actors that were never unemployed for more than 3 years at a stretch. (Assume that the actors remain unemployed between two consecutive movies).

In [8]:

```
r = pd.read_sql_query("select PID, Name from Person where PID not in(select distinct(PID) f
r
```

Out[8]:

	PID	Name
0	nm0000288	Christian Bale
1	nm0000949	Cate Blanchett
2	nm1212722	Benedict Cumberbatch
3	nm0365140	Naomie Harris
4	nm0785227	Andy Serkis
...	...	...
37561	nm2182643	Kamika Verma
37562	nm1029114	Dhorairaj Bhagavan
37563	nm3769883	Nasir Shaikh
37564	nm1470989	Kannan
37565	nm0298158	Adrian Fulle

37566 rows × 2 columns

9. Find all the actors that made more movies with Yash Chopra than any other director.

In [30]:

```
ans = pd.read_sql_query("""SELECT DISTINCT  Actor, Count(*) Movies_with_YashChopra
                        FROM(SELECT DISTINCT p1.Name as Director, m1.title as Movie
                        FROM Person p1 Inner Join M_Director md on TRIM(md.PID)=p1.PID
                        Inner Join Movie m1 on TRIM(md.MID)=m1.MID and
                        p1.Name LIKE 'Yash%' Group By p1.Name, m1.title) t1
                        Inner Join (SELECT DISTINCT p2.Name as Actor,m2.title as Movie fro
                        Inner Join M_Cast mc on TRIM(mc.PID)=p2.PID
                        Inner Join Movie m2 on TRIM(mc.MID)=m2.MID Group By p2.Name, m2.ti
                        Group By t2.Actor
                        Order By Movies_with_YashChopra DESC""",conn)
ans
```

Out[30]:

	Actor	Movies_with_YashChopra
0	Jagdish Raj	11
1	Manmohan Krishna	10
2	Manmohan Krishna	10
3	Iftekhar	9
4	Madan Puri	8
...	...	...
509	Romesh Sharma	1
510	Sachin	1
511	Sajid Khan	1
512	Sunny Deol	1
513	Tinnu Verma	1

514 rows × 2 columns

10. The Shahrukh number of an actor is the length of the shortest path between the actor and Shahrukh Khan in the "co-acting" graph. That is, Shahrukh Khan has Shahrukh number 0; all actors who acted in the same film as Shahrukh have Shahrukh number 1; all actors who acted in the same film as some actor with Shahrukh number 1 have Shahrukh number 2, etc. Return all actors whose Shahrukh number is 2.

In [31]:

```
r = pd.read_sql_query("""SELECT DISTINCT TRIM(name) Name
                         FROM Person p INNER JOIN M_Cast c on TRIM(p.PID) = TRIM(c.PID)
                         INNER JOIN Movie m ON m.MID = c.MID AND TRIM(p.Name)!='Shah Rukh K
                         and m.title in (SELECT DISTINCT title FROM Person p3
                         INNER JOIN M_Cast c3 on p3.PID = TRIM(c3.PID) AND TRIM(p3.Name) =
                         INNER JOIN Movie m3 ON TRIM(m3.MID) = TRIM(c3.MID) AND p3.Name IN
                         (SELECT DISTINCT Name FROM Person p2
                         INNER JOIN M_Cast c2 ON TRIM(p2.PID) = TRIM(c2.PID)
                         INNER JOIN Movie m2 ON m2.MID = c2.MID AND TRIM(p2.Name)!='Shah Ru
                         AND m2.title IN
                         (SELECT DISTINCT title FROM Person p3
                         INNER JOIN M_Cast c3 ON p3.PID = TRIM(c3.PID) AND TRIM(p3.Name) =
                         INNER JOIN Movie m3 ON m3.MID = c3.MID))) ORDER BY Name""",conn)

r
```

Out[31]:

	Name
0	'Musafir' Radio Performing
1	A'Ali de Sousa
2	A. Abdul Hameed
3	A. Darpan
4	A. Gabibi
...	...
16160	Zulfi Sayed
16161	Zulkhumor Muminova
16162	Zurab Kapianidze
16163	Zuri Echea
16164	Zuzanna Zajac

16165 rows × 1 columns