In [1]:

```
import sqlite3
import pandas as pd

conn = sqlite3.connect('Db-IMDB.db')
```

In [2]:

```
sql = """ UPDATE genre SET name=trim(name) """
cur = conn.cursor()
cur.execute(sql)
conn.commit()
```

In [3]:

```
#Questions for the assignment
#https://drive.google.com/file/d/1b1Wd29cjNSszHP8OeK__GBohxyuBSt0r/view?usp=sharing
```

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]:

```
query = pd.read_sql_query('''select distinct p.name, m.title, m.year from person p, movie m join
    (select md.pid as pid, mg.mid as mid from m_director md join m_genre mg on mg.mid=md.mid
    where mg.gid IN
    (select g.gid from genre g where g.name like "%Comedy%")) as temp on temp.pid=p.pid and
temp.mid=m.mid group by m.title
    having m.year%4==0 and m.year%100!=0 or m.year%400==0''', conn)
query
```

Out[4]:

	Name	title	year
0	Rabi Kinagi	100% Love	2012
1	Ajai Sinha	3 Bachelors	2012
2	Remo D'Souza	A Flying Jatt	2016
3	Pankaj Parashar	Ab Ayega Mazaa	1984
4	Sachin Kundalkar	Aiyyaa	2012
5	Saeed Akhtar Mirza	Albert Pinto Ko Gussa Kyon Ata Hai	1980
6	Basu Chatterjee	Apne Paraye	1980
7	Frank Coraci	Around the World in 80 Days	2004
8	Rajpal Yadav	Ata Pata Lapatta	2012
9	Nitya Mehra	Baar Baar Dekho	2016
10	Salim Raza	Bach ke Zara	2008
11	Siddharth Anand	Bachna Ae Haseeno	2008
12	Tarun Majumdar	Balika Badhu	1976
13	Anurag Basu	Barfi!	2012
14	Hrishikesh Mukherjee	Bawarchi	1972
15	Mike Judge	Beavis and Butt-Head Do America	1996
16	Aditya Chopra	Befikre	2016
17	Rama Rao Tatineni	Beti No. 1	2000
18	Vivek Sharma	Bhoothnath	2008

	Name	title	year
19	J.K. Bimani	Biwi Ho Toh Aisi	1988
20	Rohit Shetty	Bol Bachchan	2012
21	David Dhawan	Bol Radha Bol	1992
22	Brij	Bombay 405 Miles	1980
23	Nagesh Kukunoor	Bombay to Bangkok	2008
24	S. Ramanathan	Bombay to Goa	1972
25	Bhappi Sonie	Brahmachari	1968
26	Gurinder Chadha	Bride & Prejudice	2004
27	Jagdish Rajpurohit	Bumboo	2012
28	Sachin Yardi	C Kkompany	2008
29	Hriday Shetty	Chaalis Chauraasi	2012
...	...	...	...
199	Ramesh Sippy	Seeta Aur Geeta	1972
200	Raj Kaushal	Shaadi Ka Laddoo	2004
201	Prakash Mehra	Sharaabi	1984
202	Anees Bazmee	Singh Is Kinng	2008
203	Ashwani Dhir	Son of Sardaar	2012
204	Onir	Sorry Bhai!	2008
205	Karan Johar	Student of the Year	2012
206	Rohit Shetty	Sunday	2008
207	Vimal Kumar	Suno Sasurjee	2004
208	P. Sambasiva Rao	Swayamvar	1980
209	Oliver Paulus	Tandoori Love	2008
210	T.L.V. Prasad	Tauba Tauba	2004
211	Abhishek Sharma	Tere Bin Laden: Dead Or Alive	2016
212	Joy Augustine	Tere Mere Sapne	1996
213	Mandeep Kumar	Tere Naal Love Ho Gaya	2012
214	Kunal Kohli	Teri Meri Kahaani	2012
215	Griffin Dunne	The Accidental Husband	2008
216	James Dodson	The Other End of the Line	2008
217	Sunil K. Reddy	Thikka	2016
218	Kunal Kohli	Thoda Pyaar Thoda Magic	2008
219	Guddu Dhanoa	Tu Chor Main Sipahi	1996
220	Milind Dhaimade	Tu Hai Mera Sunday	2016
221	Vijay	Tutak Tutak Tutiya	2016
222	Sachin Kamlakar Khot	Ugly Aur Pagli	2008
223	Shoojit Sircar	Vicky Donor	2012
224	Brij	Victoria No. 203	1972
225	Shyam Benegal	Welcome to Sajjanpur	2008
226	Aditya Datt	Will You Marry Me	2012
227	Deepak Anand	Yaad Rakhegi Duniya	1992
228	Subhash Ghai	Yuvvraaj	2008

229 rows × 3 columns

In [5]:

```
act = pd.read_sql_query('''select distinct p.name from person p''',conn)
act
```

Out[5]:

	Name

	Name
0	Christian Bale
1	Cate Blanchett
2	Benedict Cumberbatch
3	Naomie Harris
4	Andy Serkis
5	Peter Mullan
6	Jack Reynor
7	Eddie Marsan
8	Tom Hollander
9	Matthew Rhys
10	Freida Pinto
11	Rohan Chand
12	Keveshan Pillay
13	Louis Ashbourne Serkis
14	Moonsamy Narasigadu
15	Soobrie Govender
16	Gopal Singh
17	Kista Munsami
18	Mahomed Araf Cassim
19	Riaz Mansoor
20	Roshan Jayesh Patel
21	T'khai Phillips
22	Sachin Soni
23	Hridhay Somera
24	Ethaniel Jaden Moonsamy
25	Gareth Ryan Benjamin
26	Nirvayesh Chakravorty Thanendra
27	Adiyan Ahmed Choudhury
28	Amara Motala
29	Diyara Prakash
...	...
36283	Visakh G S
36284	Sandip Ray
36285	S.V. Krishna Reddy
36286	R.K. Selvamani
36287	Amma Rajasekhar
36288	Rohit Gupta
36289	Bela Negi
36290	Sanjay Talreja
36291	Rajatesh Nayyar
36292	Murali Nair
36293	Pryas Gupta
36294	Shivamani
36295	Oliver Paulus
36296	Vishal Inamdar
36297	Kumar Shahani
36298	Ka-Fai Wai
36299	Avtandil Varsimashvili
36300	G. Ram Prasad
36301	Raja Chanda

	Name
36302	Deepak Ramteke
36303	Srinivas Sunderrajan
36304	Kamika Verma
36305	Dhorairaj Bhagavan
36306	Nasir Shaikh
36307	Abbas
36308	Kannan
36309	Adrian Fulle
36310	Gulshan Kumar
36311	Iqbal
36312	Sushma Shiromani

36313 rows × 1 columns

In [6]:

```
sql = """ UPDATE m_cast SET pid=trim(pid) """
cur = conn.cursor()
cur.execute(sql)
conn.commit()
```

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

In [7]:

```
df = pd.read_sql_query('''SELECT name from Person where pid IN
                          (SELECT pid FROM m_cast where mid IN
                          (SELECT mid from movie where title = 'Anand'));''', conn)
```

In [8]:

```
print("all the actors who played in the movie 'Anand' (1971)")
df
```

```
all the actors who played in the movie 'Anand' (1971)
```

Out[8]:

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

	Name
16	Moonsamy

In [9]:

```
sql = '''UPDATE Movie SET year=SUBSTR(year,-4) WHERE year LIKE '% %' '''
cur = conn.cursor()
cur.execute(sql)
conn.commit()
```

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

In [10]:

```
movie = pd.read_sql_query('''SELECT name from person where pid IN
                          (SELECT pid FROM m_cast where mid IN
                          (SELECT mid FROM movie WHERE year NOT BETWEEN 1970 AND 1990))''', conn)
movie
```

Out[10]:

	Name
0	Christian Bale
1	Cate Blanchett
2	Benedict Cumberbatch
3	Naomie Harris
4	Andy Serkis
5	Peter Mullan
6	Jack Reynor
7	Eddie Marsan
8	Tom Hollander
9	Matthew Rhys
10	Freida Pinto
11	Rohan Chand
12	Keveshan Pillay
13	Louis Ashbourne Serkis
14	Moonsamy Narasigadu
15	Soobrie Govender
16	Gopal Singh
17	Kista Munsami
18	Mahomed Araf Cassim
19	Riaz Mansoor
20	Roshan Jayesh Patel
21	T'khai Phillips
22	Sachin Soni
23	Hridhay Somera
24	Ethaniel Jaden Moonsamy
25	Gareth Ryan Benjamin
26	Nirvayesh Chakravorty Thanendra
27	Adiyan Ahmed Choudhury
28	Amara Motala
29	Diyara Prakash
...	...
29965	Anand Balraj

	Name
29966	Muazzam Beg
29967	Asad Shan
29968	Viji Thampi
29969	A.M.R. Ramesh
29970	Joy Mukherjee
29971	Punarvasu Naik
29972	Ashok Kheny
29973	Nagarjuna Akkineni
29974	Sartaj Singh Pannu
29975	Ranjit Kapoor
29976	Sanjay Amar
29977	Anjan Dutt
29978	Manmohan Krishna
29979	V. Ravichandran
29980	Pawan Gill
29981	Bharat Dabholkar
29982	Jyothika
29983	Marion Rodrigues
29984	Yograj Bhat
29985	Sadhu Kokila
29986	Parvin Dabas
29987	Vineet Khetrapal
29988	C. Jenner Jose
29989	Mukesh Asopa
29990	Rahat Kazmi
29991	Srinivas Sunderrajan
29992	Abbas
29993	Iqbal
29994	Sushma Shiromani

29995 rows × 1 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 [11]:

```
query = '''
    select a.name, count(b.mid) num_movies from person a left join m_director b
    on a.pid=b.pid group by a.pid having num_movies>10 order by num_movies desc
'''
result = pd.read_sql_query(query, conn)
result
```

Out[11]:

	Name	num_movies
0	David Dhawan	78
1	Mahesh Bhatt	70
2	Ram Gopal Varma	60
3	Vikram Bhatt	58
4	Hrishikesh Mukherjee	54
5	Yash Chopra	42

	Name	num_movies
6	Basu Chatterjee	38
7	Shakti Samanta	38
8	Subhash Ghai	36
9	Shyam Benegal	34
10	Abbas Alibhai Burmawalla	34
11	Manmohan Desai	32
12	Gulzar	32
13	Raj N. Sippy	32
14	Raj Kanwar	30
15	Mahesh Manjrekar	30
16	Priyadarshan	30
17	Indra Kumar	28
18	Raj Khosla	28
19	Rahul Rawail	28
20	Rajkumar Santoshi	28
21	Rakesh Roshan	26
22	Dev Anand	26
23	Vijay Anand	26
24	Harry Baweja	26
25	Anurag Kashyap	26
26	Ananth Narayan Mahadevan	26
27	K. Raghavendra Rao	26
28	Anees Bazmee	24
29	Guddu Dhanoa	24
...	...	...
82	Puri Jagannadh	14
83	Sangeeth Sivan	14
84	K. Viswanath	14
85	Ashu Trikha	14
86	Imtiaz Ali	14
87	Saawan Kumar Tak	14
88	Manoj Kumar	12
89	Deepak Anand	12
90	Aparna Sen	12
91	Sooraj R. Barjatya	12
92	Brij	12
93	Ravi Chopra	12
94	Rituparno Ghosh	12
95	Karan Johar	12
96	Sohail Khan	12
97	Sandesh Kohli	12
98	Mohan Kumar	12
99	Jag Mundhra	12
100	K. Ravi Shankar	12
101	Shantaram Rajaram Vankudre	12
102	Babbar Subhash	12
103	Deepak Tijori	12
104	Onir	12
105	Vipul Amrutlal Shah	12
106	Kabir Khan	12

	Name	num_movies
107	Shaad Ali	12
108	Samir Karnik	12
109	A.R. Murugadoss	12
110	Vivek Agnihotri	12
111	Nila Madhab Panda	12

112 rows × 2 columns

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

In [12]:

```
fr = pd.read_sql_query('''select z.year, count(*)
from Movie z
where not exists (select *
                  from Person x, M_Cast xy
                  where x.PID = xy.PID and xy.MID = z.MID and x.Gender!='Female')
group by z.year''', conn)
fr
```

Out[12]:

	year	count(*)
0	1939	1
1	1999	1
2	2000	1
3	2009	1
4	2012	1
5	2018	2

5. b. 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 [13]:

```
perc_query = pd.read_sql_query('''select a.year, a.c*100.00/b.c as percentage, b.c as
total_overall
from (select z.year, count(*) as c
      from Movie z
      where not exists (select *
                        from Person x, M_Cast xy
                        where x.PID = xy.PID and xy.MID = z.MID and x.Gender!='Female')
      group by z.year) a,
     (select z.year, count(*) as c from Movie z group by z.year) b
where a.year=b.year
order by a.year''', conn)
perc_query
```

Out[13]:

	year	percentage	total_overall
0	1939	50.000000	2
1	1999	1.515152	66
2	2000	1.562500	64
3	2009	0.909091	110

	year	percentage	total_overall
4	2012	0.909091	110
5	2018	1.923077	104

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 [14]:

```
l_cast_query = pd.read_sql_query('''select x.title, count(distinct xy.PID) as c
from Movie x, M_Cast xy
where x.MID = xy.MID
group by x.MID, x.title
having not exists (select uv.MID
                   from M_Cast uv
                   group by uv.MID
                   having count(distinct uv.PID) > count(distinct xy.PID))''', conn)
l_cast_query
```

Out[14]:

	title	c
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 [15]:

```
decade_query = pd.read_sql_query('''select y.year, count(*)
from (select distinct x.year from Movie x) y,
     Movie z
where y.year <= z.year and z.year < y.year+10
group by y.year
having not exists (select y1.year
                   from (select distinct x1.year from Movie x1) y1, Movie z1
                   where y1.year <= z1.year and z1.year < y1.year+10
                   group by y1.year
                   having count(z1.MID) > count(z.MID))''', conn)
decade_query
```

Out[15]:

	year	count(*)
0	2008	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 [16]:

```
unemployed_query = pd.read_sql_query('''select Name from Person where PID
    not in(select distinct(PID)from M_Cast as C1 natural join Movie as
    M1 where exists(select MID from M_Cast as C2 natural join Movie as M2 where
    C1.PID = C2.PID and (M2.year - 3) > M1.year and
    not exists(select MID from M_Cast as C3 natural join
```

```
    Movie as M3 where C1.PID = C3.PID and M1.year < M3.year and M3.year < M2.year)))''', conn)
unemployed_query
```

Out[16]:

	Name
0	Christian Bale
1	Cate Blanchett
2	Benedict Cumberbatch
3	Naomie Harris
4	Andy Serkis
5	Peter Mullan
6	Jack Reynor
7	Eddie Marsan
8	Tom Hollander
9	Matthew Rhys
10	Freida Pinto
11	Rohan Chand
12	Keveshan Pillay
13	Louis Ashbourne Serkis
14	Moonsamy Narasigadu
15	Soobrie Govender
16	Gopal Singh
17	Kista Munsami
18	Mahomed Araf Cassim
19	Riaz Mansoor
20	Roshan Jayesh Patel
21	T'khai Phillips
22	Sachin Soni
23	Hridhay Somera
24	Ethaniel Jaden Moonsamy
25	Gareth Ryan Benjamin
26	Nirvayesh Chakravorty Thanendra
27	Adiyan Ahmed Choudhury
28	Amara Motala
29	Diyara Prakash
...	...
38255	Sandip Ray
38256	S.V. Krishna Reddy
38257	R.K. Selvamani
38258	Amma Rajasekhar
38259	Rahat Kazmi
38260	Rohit Gupta
38261	Bela Negi
38262	Sanjay Talreja
38263	Rajatesh Nayyar
38264	Murali Nair
38265	Pryas Gupta
38266	Shivamani
38267	Oliver Paulus
38268	Vishal Inamdar

	Name
38269	Kumar Shahani
38270	Ka-Fai Wai
38271	Avtandil Varsimashvili
38272	G. Ram Prasad
38273	Raja Chanda
38274	Deepak Ramteke
38275	Srinivas Sunderrajan
38276	Kamika Verma
38277	Dhorairaj Bhagavan
38278	Nasir Shaikh
38279	Abbas
38280	Kannan
38281	Adrian Fulle
38282	Gulshan Kumar
38283	Iqbal
38284	Sushma Shiromani

38285 rows × 1 columns

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

In [32]:

```
# #number of movies between an actor and director
# act =  pd.read_sql_query( '''SELECT pd.name as Director, pa.name as actor, COUNT(*)
# FROM M_director d JOIN
#       Person pd
#       ON pd.pid = d.pid JOIN
#       M_cast c
#       ON c.MID = d.MID JOIN
#       Person pa
#       ON pa.PID = c.PID
# GROUP BY pd.name, pa.name''',conn)
# act.head()
```

In [19]:

```
actors = pd.read_sql_query('''SELECT da.*
    FROM (SELECT pd.pid, pd.name as Director, pa.pid, pa.name as actor, COUNT(*) as cnt,
             RANK() OVER (PARTITION BY pa.pid ORDER BY COUNT(*) DESC) as seqnum,
             COUNT(*) OVER (PARTITION BY pa.pid, COUNT(*)) as num_with_cnt
    FROM M_director d JOIN
           Person pd
           ON pd.pid = d.pid JOIN
           M_cast c
           ON c.MID = d.MID JOIN
           Person pa
           ON pa.PID = c.PID
      GROUP BY pd.pid, pd.name, pa.pid, pa.name
     ) da
    WHERE director = 'Yash Chopra' AND
      seqnum = 1 AND
      num_with_cnt = 1;''',conn)
actors
```

Out[19]:

	pid	Director	pid:1	actor	cnt	seqnum	num_with_cnt

10. The Shahrukh number of an actor is the length of the shortest path

between the actor andShahrukh Khan in the "co-acting" graph. That is, Shahrukh Khan has Shahrukh number 0; allactors 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 [33]:

```
sql = """ UPDATE Person SET Name=trim(Name) """
cur = conn.cursor()
cur.execute(sql)
conn.commit()
```

In [34]:

```
srk_query = pd.read_sql_query('''select  count(distinct c2.PID)
from Person a0, M_Cast c0, M_Cast c1a, M_Cast c1b, M_Cast c2
where  a0.Name = 'Shah Rukh Khan'
   AND a0.PID   = c0.PID
   AND c0.MID  = c1a.MID
   AND c1b.PID = c1a.PID
   AND c1b.MID = c2.MID
   AND c2.PID NOT IN
        (select d1.PID
         from  Person b0, M_Cast d0, M_Cast d1
         where b0.Name = 'Shah Rukh Khan'
           AND b0.PID  = d0.PID
           AND d0.MID = d1.MID
        )''', conn)
srk_query
```

Out[34]:

	count(distinct c2.PID)
0	25698