In [1]:

```
import pandas as pd
import os
import sqlite3
import time
```

In [2]:

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

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 [3]:

```
start = time.time()
Q1 = pd.read_sql_query("""
SELECT NAME DIRECTOR_NAME FROM PERSON
WHERE PID IN (SELECT TRIM(PID) FROM M_DIRECTOR
WHERE MID IN( SELECT MID FROM MOVIE WHERE
(YEAR%4=0 and YEAR%100!=0) and (YEAR%4=0 or YEAR%100=0 and YEAR%400=0)
AND MID IN (SELECT trim(MID) FROM M_GENRE
WHERE GID IN (SELECT GID FROM GENRE WHERE trim(NAME) like '%comedy%'))))
""", con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q1.shape)
Q1
```

```
Time taken to run the query 0.5546820163726807
Shape of the output is (181, 1)
```

Out[3]:

	DIRECTOR_NAME
0	Griffin Dunne
1	Madonna
2	Gurinder Chadha
3	Frank Coraci
4	Tarun Mansukhani
5	Lekh Tandon
6	S.S. Rajamouli
7	Jugal Hansraj
8	Mike Judge
9	Rajat Kapoor
10	Karan Johar
11	Anurag Kashyap
12	Rajpal Yadav
13	Farah Khan
14	Subhash Ghai
15	Arbaaz Khan
16	Vaibhav Misra
17	Rakesh Roshan
18	Nikkhil Advani
19	Sohail Khan

	DIRECTOR_NAME
19	Sohail Khan
20	Sachin
21	Abbas Tyrewala
22	Umesh Shukla
23	Ganesh Acharya
24	Shakun Batra
25	Abhishek Sharma
26	Sajid Khan
27	Ketan Mehta
28	Mahesh Bhatt
29	Jagdish Rajpurohit
...	...
151	Jyoti Swaroop
152	Jaideep Sen
153	Ram Mukherjee
154	Rabi Kinagi
155	Tarun Majumdar
156	Jeeva
157	Bobby Kolli
158	J.K. Bihari
159	Ajai Sinha
160	K.S. Prakash Rao
161	Kalpataru
162	K.S. Ravi
163	Anil Senior
164	Mandeep Kumar
165	Sourabh Shrivastava
166	Rajesh Bajaj
167	Suhas Kadav
168	Jaideep Varma
169	Srinivas Bhashyam
170	Chandrakant Kulkarni
171	Aspi Irani
172	Ghorban Mohammadpour
173	Ramanjit Juneja
174	Salim Raza
175	Sachin Kamlakar Khot
176	Debu Sen
177	Shankaraiya
178	Amma Rajasekhar
179	Oliver Paulus
180	Raja Chanda

181 rows × 1 columns

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

In [4]:

```
start = time.time()
Q2 = pd.read_sql_query("""SELECT NAME ACTOR_NAME FROM PERSON
WHERE PID IN (SELECT TRIM(PID) FROM M_CAST
WHERE MID IN(SELECT MID FROM MOVIE WHERE TITLE='Anand'))""", con)
WHERE MID IN(SELECT MID FROM MOVIE WHERE TITLE='Anand'))""", con)
#Always remember to close the database
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q1.shape)
Q2
```

```
Time taken to run the query 0.09194707870483398
Shape of the output is (181, 1)
```

Out[4]:

	ACTOR_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
16	Moolchand

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

In [5]:

```
start = time.time()
Q3 = pd.read_sql_query("""SELECT * FROM
(SELECT DISTINCT NAME ACTOR_NAME
FROM PERSON WHERE PID IN
(SELECT TRIM(PID)FROM M_CAST WHERE MID IN(SELECT MID FROM MOVIE WHERE YEAR<1970))

INTERSECT

SELECT DISTINCT NAME FROM PERSON
WHERE PID IN
(SELECT TRIM(PID) FROM M_CAST WHERE MID IN(SELECT MID FROM MOVIE WHERE YEAR>1990)))
""", con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q3.shape)
#Always remember to close the database
Q3
```

```
Time taken to run the query 0.36578893661499023
Shape of the output is (453, 1)
```

Out[5]:

	ACTOR_NAME

	ACTOR_NAME
0	A.K. Hangal
1	Aachi Manorama
2	Abbas
3	Abdul
4	Abhi Bhattacharya
5	Abhimanyu Sharma
6	Achala Sachdev
7	Adil
8	Ajay
9	Ajit
10	Akashdeep
11	Akbar Bakshi
12	Alka
13	Allu Ramalingaiah
14	Altaf
15	Amar
16	Amarnath
17	Ameer
18	Amitabh Bachchan
19	Amjad Khan
20	Amol Sen
21	Amrit
22	Anand
23	Anand Kumar
24	Anand Tiwari
25	Anil
26	Anil Kumar
27	Anil Nagrath
28	Anjali Kadam
29	Anju Mahendru
...	...
423	Tulsi
424	Tun Tun
425	Uma
426	Umesh Sharma
427	Unni Mary
428	Urmila Bhatt
429	Usha Kiran
430	Utpal Dutt
431	Veena
432	Veera
433	Vijay
434	Vijayalalitha
435	Vijayalaxmi
436	Viju Khote
437	Vikram Makandar
438	Vineet Kumar
439	Vinod Mehra
440	Vinod Sharma
441	Vishnu

	ACTOR_NAME
442	Vishwa Mehra
443	Vyjayanthimala
444	Waheeda Rehman
445	Wasi Khan
446	Yash Kumar
447	Yasmin
448	Yunus Parvez
449	Yusuf
450	Zia
451	Zohra Sehgal
452	Zul Vellani

453 rows × 1 columns

1. 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 [6]:

```
start = time.time()
Q4 = pd.read_sql_query("""
SELECT P.NAME,COUNT(MD.PID) NUMBER_OF_MOVIES FROM PERSON P JOIN M_DIRECTOR MD ON P.PID=MD.PID  GRO
UP BY MD.PID HAVING NUMBER_OF_MOVIES>=10
ORDER BY NUMBER_OF_MOVIES DESC""", con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q4.shape)
Q4
```

```
Time taken to run the query 0.06096529960632324
Shape of the output is (58, 2)
```

Out[6]:

	Name	NUMBER_OF_MOVIES
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

	Name	NUMBER_OF_MOVIES
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
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. b. Now include a small change: report for each year the percentage of movies in that

In [7]:

```
Q5=pd.read_sql_query("""SELECT CAST(SUBSTR(year,-4) AS UNSIGNED) as year,count(*)
Female_Number_of_Movie_Count
FROM (SELECT MV.MID,CAST(SUBSTR(MV.YEAR,-4) AS UNSIGNED) YEAR FROM PERSON as PS JOIN M_CAST as MC
on PS.PID=TRIM(MC.PID), MOVIE as MV on MV.MID=MC.MID
EXCEPT
SELECT MV1.MID,CAST(SUBSTR(MV1.YEAR,-4) AS UNSIGNED) YEAR FROM PERSON as PS1 JOIN M_CAST as MC1 on
PS1.PID=TRIM(MC1.PID), MOVIE MV1 on MV1.MID=MC1.MID
WHERE PS1.GENDER='Male') group by year  """,con)
#CAST(SUBSTR(year,-4) AS UNSIGNED)
Q5
```

Out[7]:

	year	Female_Number_of_Movie_Count
0	1939	1
1	1999	1
2	2000	1
3	2018	2

In [8]:

```
start = time.time()
Q5_a=pd.read_sql_query("""SELECT CAST(SUBSTR(M.year,-4) AS UNASSIGNED) Year, COUNT(DISTINCT TRIM(M
ID)) Female_Number_of_Movie_Count
FROM MOVIE M
where MID NOT IN(
SELECT MC.MID
FROM M_CAST MC
JOIN PERSON P ON P.PID = trim(MC.PID)
WHERE TRIM(P.GENDER) IN ('Male', 'None'))
GROUP BY CAST(SUBSTR(M.year,-4) AS UNASSIGNED)""",con) #SUBSTR(M.year,-4) USED TO REMOVE UNWANTED
CHARACTER IN YEAR COLUMN
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q5_a.shape)
Q5_a
```

```
Time taken to run the query 0.2568511962890625
Shape of the output is (6, 2)
```

Out[8]:

	Year	Female_Number_of_Movie_Count
0	1939	1
1	1999	1
2	2000	1
3	2009	1
4	2012	1
5	2018	2

5b. Now include a small change: report for each year the percentage of movies in that

In [9]:

```
start = time.time()
Q5=pd.read_sql_query("""

SELECT MY.YEAR, MY.Total_Movies,(IFNULL(MF.Female_Number_of_Movie_Count, 0) * 100)/MY.Total_Movies
Female_Movie_Percentage
FROM (SELECT CAST(SUBSTR(M.YEAR,-4) AS UNASSIGNED) Year,
COUNT(DISTINCT TRIM(MID)) Total_Movies
FROM MOVIE M
GROUP BY CAST(SUBSTR(M.YEAR,-4) AS UNASSIGNED)) MY
LEFT OUTER JOIN (SELECT CAST(SUBSTR(M.year,-4) AS UNASSIGNED) Year, COUNT(DISTINCT TRIM(MID)) Fema
le_Number_of_Movie_Count
FROM MOVIE M
where MID NOT IN(
SELECT MC.MID
FROM M_CAST MC
JOIN PERSON P ON P.PID = trim(MC.PID)
WHERE TRIM(P.GENDER) IN ('Male', 'None'))
GROUP BY CAST(SUBSTR(M.year,-4) AS UNASSIGNED)) MF ON
TRIM(MY.YEAR) = TRIM(MF.YEAR)
where Female_Movie_Percentage>0
ORDER BY Female_Movie_Percentage DESC
```

```
 """,con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q5.shape)

Q5
```

```
Time taken to run the query 0.3088076114654541
Shape of the output is (4, 3)
```

Out[9]:

	Year	Total_Movies	Female_Movie_Percentage
0	1939	2	50
1	1999	66	1
2	2000	64	1
3	2018	104	1

1. 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 [10]:

```
start = time.time()
Q6 = pd.read_sql_query(""" SELECT O.MID, M.TITLE, MAX(O.CAST_COUNT) AS CAST_SIZE
FROM (SELECT COUNT(*) AS CAST_COUNT,MID
FROM M_CAST GROUP BY MID) O JOIN MOVIE M ON M.MID=O.MID """, con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q6.shape)
#Always remember to close the database

Q6
```

```
Time taken to run the query 0.14591693878173828
Shape of the output is (1, 3)
```

Out[10]:

	MID	title	CAST_SIZE
0	tt5164214	Ocean's Eight	238

1. 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 [11]:

```
Q7 = pd.read_sql_query("""
SELECT D as DECADE ,YEAR, MAX(MOVIE_COUNTS) LARGE_NUMBER_OF_MOVIE
FROM (SELECT D, COUNT(*) AS MOVIE_COUNTS,YEAR
FROM ( SELECT CAST(SUBSTR(M.YEAR,-4) AS UNSIGNED) as YEAR, CAST(SUBSTR(YW.MIN_YEAR,-4) AS UNSIGNED
), (((CAST(SUBSTR(M.YEAR,-4) AS UNSIGNED)-CAST(SUBSTR(YW.MIN_YEAR,-4) AS UNSIGNED))/10)+1) AS D FR
OM MOVIE M JOIN ( SELECT MIN(CAST(SUBSTR(year,-4) AS UNSIGNED)) AS MIN_YEAR FROM  MOVIE)YW ON 1=1)
I_YW GROUP BY D)O_YW """,  con)


#CAST(SUBSTR(year,-4) AS UNSIGNED) YEAR
Q7
```

```
Out[11]:
```

	DECADE	YEAR	LARGE_NUMBER_OF_MOVIE
0	8	2008	1047

1. 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 [12]:
```

```
con = sqlite3.connect('Db-IMDB-Assignment.db')
start = time.time()
Q8 = pd.read_sql_query("""with
NUM_MORE_THAN_YEAR AS
(SELECT AY.PID, AY.YEAR, AY.YEAR+4 AS Year_4, AY.NUMBER_OF_MOV,
ATY.MIN_YEAR, ATY.MAX_YEAR
FROM (SELECT TRIM(MC.PID) PID, CAST(SUBSTR(year,-4) AS UNASSIGNED) Year,
COUNT(DISTINCT TRIM(M.MID)) Number_of_Mov
FROM M_CAST MC, MOVIE M
WHERE MC.MID = TRIM(M.MID)
GROUP BY TRIM(MC.PID), CAST(SUBSTR(year,-4) AS UNASSIGNED)
ORDER BY NUMBER_OF_MOV DESC) AY, (SELECT AY.PID, COUNT(AY.YEAR) AS Number_of_Years, MIN(AY.YEAR) A
S Min_Year,
MAX(AY.YEAR) AS Max_Year
FROM (SELECT TRIM(MC.PID) PID, CAST(SUBSTR(year,-4) AS UNASSIGNED) Year,
COUNT(DISTINCT TRIM(M.MID)) Number_of_Mov
FROM M_CAST MC, MOVIE M
WHERE MC.MID = TRIM(M.MID)
GROUP BY TRIM(MC.PID), CAST(SUBSTR(year,-4) AS UNASSIGNED)
ORDER BY NUMBER_OF_MOV DESC) AY
GROUP BY AY.PID
HAVING COUNT(AY.YEAR) > 1) ATY
WHERE AY.PID = ATY.PID),
MORE_THAN_YEAR AS
(SELECT AM.PID, NY.YEAR, SUM(AM.NUMBER_OF_MOV) AS MORE_THAN_YEARS_PRESENT
FROM NUM_MORE_THAN_YEAR AM, NUM_MORE_THAN_YEAR NY
WHERE AM.PID = NY.PID AND
AM.YEAR BETWEEN NY.MIN_YEAR AND NY.YEAR_4 AND
NY.YEAR_4 <= NY.MAX_YEAR
GROUP BY AM.PID, NY.YEAR)
SELECT DISTINCT TRIM(P.NAME) AS ACTORS_NEVER_UNEMPLOYED
FROM PERSON P
WHERE TRIM(P.PID) NOT IN (SELECT DISTINCT NMP.PID
FROM (SELECT AM.PID, NY.YEAR, SUM(AM.NUMBER_OF_MOV) AS NUMBER_OF_MOVIE_PRESENT
FROM NUM_MORE_THAN_YEAR AM, NUM_MORE_THAN_YEAR NY
WHERE AM.PID = NY.PID AND
AM.YEAR BETWEEN NY.MIN_YEAR AND NY.YEAR
GROUP BY AM.PID, NY.YEAR) NMP, MORE_THAN_YEAR AM_4
WHERE NMP.PID = AM_4.PID AND
NMP.YEAR = AM_4.YEAR AND
NMP.NUMBER_OF_MOVIE_PRESENT = AM_4.MORE_THAN_YEARS_PRESENT)""", con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q8.shape)
#Always remember to close the database
con.close()
Q8
```

```
Time taken to run the query 8.591050863265991
Shape of the output is (32585, 1)
```

```
Out[12]:
```

	ACTORS_NEVER_UNEMPLOYED
0	Christian Bale
1	Cate Blanchett
2	Benedict Cumberbatch

	ACTORS_NEVER_UNEMPLOYED
3	Naomie Harris
4	Andy Serkis
5	Peter Mullan
6	Jack Reynor
7	Eddie Marsan
8	Tom Hollander
9	Matthew Rhys
10	Rohan Chand
11	Keveshan Pillay
12	Louis Ashbourne Serkis
13	Moonsamy Narasigadu
14	Soobrie Govender
15	Gopal Singh
16	Kista Munsami
17	Mahomed Araf Cassim
18	Riaz Mansoor
19	Roshan Jayesh Patel
20	T'khai Phillips
21	Sachin Soni
22	Hridhay Somera
23	Ethaniel Jaden Moonsamy
24	Gareth Ryan Benjamin
25	Nirvayesh Chakravorty Thanendra
26	Adiyan Ahmed Choudhury
27	Amara Motala
28	Diyara Prakash
29	Diyajal Prakash
...	...
32555	Rakesh Chaturvedi
32556	Swapna Joshi
32557	Shukla Barnali Ray
32558	Pavithran
32559	Vara Mullapoodi
32560	D. Sumana Kittur
32561	Abhishek Chhadha
32562	Arup Dutta
32563	Illangkannan
32564	Visakh G S
32565	Sandip Ray
32566	S.V. Krishna Reddy
32567	R.K. Selvamani
32568	Amma Rajasekhar
32569	Sanjay Talreja
32570	Rajatesh Nayyar
32571	Murali Nair
32572	Pryas Gupta
32573	Shivamani
32574	Oliver Paulus
32575	Vishal Inamdar
32576	Kumar Shahani
32577	Avtandil Varsimashvili

	ACTORS_NEVER_UNEMPLOYED
32578	G. Ram Prasad
32579	Raja Chanda
32580	Deepak Ramteke
32581	Kamika Verma
32582	Dhorairaj Bhagavan
32583	Nasir Shaikh
32584	Adrian Fulle

32585 rows × 1 columns

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

In [13]:

```
con = sqlite3.connect('Db-IMDB-Assignment.db')
start = time.time()
Q9 = pd.read_sql_query("""WITH
YASH_PID AS
(SELECT TRIM(P.PID) PID
FROM PERSON P
WHERE Trim(P.NAME) = 'Yash Chopra'),
MOVIE_COUNT_OF_YASH AS
(SELECT CM.ACTORS, CM.DIRECTORS,
CM.MOVIE_COUNT MOVIE_COUNT_YASH
FROM (SELECT TRIM(MC.PID) ACTORS, TRIM(MD.PID) DIRECTORS,
COUNT(DISTINCT TRIM(MD.MID)) MOVIE_COUNT
FROM M_CAST MC, M_DIRECTOR MD
WHERE MC.MID = TRIM(MD.MID)
GROUP BY ACTORS, DIRECTORS) CM, YASH_PID YC
WHERE CM.DIRECTORS = YC.PID),
COUNT_OF_OTHER_DIRECTORS_MV AS
(SELECT ACTORS, MAX(MOVIE_COUNT) MAX_MOVIE_COUNT
FROM (SELECT TRIM(MC.PID) ACTORS, TRIM(MD.PID) DIRECTORS,
COUNT(DISTINCT TRIM(MD.MID)) MOVIE_COUNT
FROM M_CAST MC, M_DIRECTOR MD
WHERE MC.MID = TRIM(MD.MID)
GROUP BY ACTORS, DIRECTORS) CM, YASH_PID YC
WHERE CM.DIRECTORS <> YC.PID
GROUP BY ACTORS),
ACTORS_MOVIE AS
(SELECT YM.ACTORS,
CASE WHEN YM.MOVIE_COUNT_YASH >=IFNULL(OD.MAX_MOVIE_COUNT, 0) THEN
'YES' ELSE 'NO' END MAX_YASH_MOVIE
FROM MOVIE_COUNT_OF_YASH YM
LEFT OUTER JOIN COUNT_OF_OTHER_DIRECTORS_MV OD ON YM.ACTORS = OD.ACTORS)
SELECT DISTINCT TRIM(P.NAME) ACTORS_NAME
FROM PERSON P
WHERE TRIM(P.PID) IN (SELECT DISTINCT ACTORS
FROM ACTORS_MOVIE
WHERE MAX_YASH_MOVIE = 'YES')""", con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q9.shape)
#Always remember to close the database
con.close()
Q9
```

```
Time taken to run the query 1.5161430835723877
Shape of the output is (243, 1)
```

Out[13]:

	ACTORS_NAME
0	Sharib Hashmi
1	Kulbir Badesron
2	Gurdas Maan

	ACTORS_NAME
3	Parikshat Sahni
4	Claire Ashton
5	Waheeda Rehman
6	Taj Gill
7	Kumud Pant
8	Gerald Tomkinson
9	Dev K. Kantawall
10	Harish Chandra
11	Saira Banu
12	Achala Sachdev
13	Darshan Aulakh
14	Kanwar Jagdish
15	Sharan Hunjan
16	Dolly Jagdeo
17	Vinita Sharma
18	Steven Baker
19	Andrew Bicknell
20	Banwarhlal Jhol
21	Kimti Anand
22	Damyanti Puri
23	Hemlata Deepak
24	Surendra Rahi
25	Yash Chopra
26	Vinod Negi
27	Balwant Bansal
28	Rajesh Jolly
29	Anup Kanwal Singh
...	...
213	Nazir
214	Renu Arya
215	Manju Maini
216	Ram Maini
217	Prince Shakeel
218	Ismail
219	Sushil Kumar
220	Sarla
221	Jago
222	Aziz Mirza
223	Aruna
224	Mahendra Sandhu
225	Mahan Swadesh
226	Om Sahni
227	Chandu
228	Bhola
229	Ramanand
230	Kuldeep Chand
231	Gopal
232	Kishan
233	Nasir
234	Ashok Chadda

	ACTORS_NAME
234	Ashok Chadda
235	Rajesh
236	Master Kelly
237	Yasin Khan
238	Ramchandra
239	Sandow S. Sethi
240	Naval
241	Prem Sood
242	Ramlal Shyamlal

243 rows × 1 columns

1. 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 [15]:

```
con = sqlite3.connect('Db-IMDB-Assignment.db')
start = time.time()
Q10 = pd.read_sql_query("""
WITH
SRK_PID as
(SELECT PID FROM PERSON where name like '%Shah Rukh Khan%'),

SRK_MOVIES AS
(SELECT distinct(MID) as MID ,YEAR
FROM MOVIE
WHERE MID IN
(SELECT TRIM(MID)
FROM M_CAST
WHERE trim(PID) IN
(SELECT TRIM(PID)
FROM PERSON
WHERE TRIM(PID) IN (SELECT * FROM SRK_PID)))),

ACTORS_FROM_MOVIES AS
(SELECT NAME,PID FROM PERSON
WHERE TRIM(PID) IN
(SELECT TRIM(PID) FROM M_CAST WHERE TRIM(MID) IN
(SELECT TRIM(MID) FROM SRK_MOVIES ))),

MOVIES_OF_THAT_ACTOR AS
(SELECT PID FROM PERSON WHERE PID IN (SELECT TRIM(PID) FROM M_CAST WHERE TRIM(MID) IN
(SELECT TRIM(MID) AS MID
FROM MOVIE
WHERE TRIM(MID) IN
(SELECT TRIM(MID)
FROM M_CAST
WHERE trim(PID) IN
(SELECT TRIM(PID)
FROM PERSON
WHERE TRIM(PID) IN (SELECT TRIM(PID) FROM ACTORS_FROM_MOVIES))))))

SELECT PID,TRIM(NAME) FROM PERSON WHERE PID IN (SELECT PID FROM MOVIES_OF_THAT_ACTOR where pid not
in (select pid from ACTORS_FROM_MOVIES))
ORDER BY pid

""", con)
end = time.time()
time_taken=(end - start)
print('Time taken to run the query',time_taken)
print('Shape of the output is',Q10.shape)
#Always remember to close the database
con.close()
Q10
# where actors not in (select pid from ACTORS_FROM_MOVIES) and actors not in (select pid from SRK_
PID)
```

```
Time taken to run the query 0.7365806102752686
Shape of the output is (25698, 2)
```

Out[15]:

	PID	TRIM(NAME)
0	nm0000027	Alec Guinness
1	nm0000047	Sophia Loren
2	nm0000093	Brad Pitt
3	nm0000096	Gillian Anderson
4	nm0000112	Pierce Brosnan
5	nm0000137	Bo Derek
6	nm0000140	Michael Douglas
7	nm0000144	Cary Elwes
8	nm0000147	Colin Firth
9	nm0000155	Whoopi Goldberg
10	nm0000173	Nicole Kidman
11	nm0000174	Val Kilmer
12	nm0000193	Demi Moore
13	nm0000195	Bill Murray
14	nm0000200	Bill Paxton
15	nm0000204	Natalie Portman
16	nm0000218	Kristin Scott Thomas
17	nm0000230	Sylvester Stallone
18	nm0000235	Uma Thurman
19	nm0000254	Isabelle Adjani
20	nm0000273	Alan Arkin
21	nm0000274	David Arquette
22	nm0000277	Richard Attenborough
23	nm0000332	Don Cheadle
24	nm0000367	Gérard Depardieu
25	nm0000375	Robert Downey Jr.
26	nm0000439	Neil Patrick Harris
27	nm0000444	Glenne Headly
28	nm0000458	William Hurt
29	nm0000502	Christopher Lloyd
...	...	...
25668	nm9972257	Kierra
25669	nm9973266	Thiagarajan
25670	nm9977801	Rajeev
25671	nm9977802	Saraswati
25672	nm9977803	Mahi Sharma
25673	nm9977805	Sachin Arya
25674	nm9977806	Pushpendra
25675	nm9977807	Sanju Ram
25676	nm9979161	Adil Lokhandwala
25677	nm9980716	Zara Khan
25678	nm9984753	Mannan Handa
25679	nm9984754	Peter Wong
25680	nm9984755	Maloslavskii
25681	nm9984756	Amit Singh

	PID	TRIM(NAME)
25682	nm9984757	Nitansh Shrivastava
25683	nm9984758	Munish Dev
25684	nm9984759	Harsh Parnami
25685	nm9984760	Krishna Banshak
25686	nm9984761	Sanjeev Sharma
25687	nm9984763	Divya Bhatia
25688	nm9984764	Radhicka Kc
25689	nm9984765	Elena Tseluiko
25690	nm9984766	Rajesh Babu
25691	nm9984767	Roman Khan
25692	nm9984768	Imran Khan
25693	nm9984769	Fernando Cruz
25694	nm9984770	Sohail Mirza
25695	nm9985086	Shreyas Sanghavi
25696	nm9988016	Ashiqa Salvan
25697	nm9988018	Ravindra Vijay

25698 rows × 2 columns