In [2]:
```
# Connect to the IMDB Databse provided. The .db file should be in the
# same folder as the jupyter notebook.

import pandas as pd
import sqlite3 as sql # included as part of python standard library

conn  = sql.connect("Db-IMDB.db")
```

In [173]:
```
# Get list of all tables in the DB.

result = pd.read_sql_query("SELECT * FROM sqlite_master where type = 't
able';", conn)
print(result)
```

```
     type        name    tbl_name  rootpage  \
0   table       Movie       Movie         2
1   table       Genre       Genre         4
2   table    Language    Language         5
3   table     Country     Country         6
4   table    Location    Location         7
5   table  M_Location  M_Location        11
6   table   M_Country   M_Country        10
7   table  M_Language  M_Language         9
8   table     M_Genre     M_Genre         8
9   table      Person      Person        12
10  table  M_Producer  M_Producer        14
11  table  M_Director  M_Director        15
12  table      M_Cast      M_Cast        16

                                                  sql
0   CREATE TABLE "Movie" (\n"index" INTEGER,\n  "M...
1   CREATE TABLE "Genre" (\n"index" INTEGER,\n  "N...
2   CREATE TABLE "Language" (\n"index" INTEGER,\n ...
3   CREATE TABLE "Country" (\n"index" INTEGER,\n  ...
4   CREATE TABLE "Location" (\n"index" INTEGER,\n ...
5   CREATE TABLE "M_Location" (\n"index" INTEGER,\...
```

```
6   CREATE TABLE "M_Country" (\n"index" INTEGER,\n...
7   CREATE TABLE "M_Language" (\n"index" INTEGER,\...
8   CREATE TABLE "M_Genre" (\n"index" INTEGER,\n  ...
9   CREATE TABLE "Person" (\n"index" INTEGER,\n  "...
10  CREATE TABLE "M_Producer" (\n"index" INTEGER,\...
11  CREATE TABLE "M_Director" (\n"index" INTEGER,\...
12  CREATE TABLE "M_Cast" (\n"index" INTEGER,\n  "...
```

In [202]:

```
'''
    Inconsistencies in Data.

    Person table has duplicate rows.
    Same primary key has two rows with names having some whitespaces as
 we can see below.
'''
result = pd.read_sql_query("""
SELECT
   P.Name,
   LENGTH(P.Name) AS name
FROM
            Person p
WHERE
    TRIM(P.PID) = 'nm0788916'
""", conn)
print(result)
```

```
              Name  name
0   Sanjeev Sharma    15
1   Sanjeev Sharma    14
```

In [177]:

```
'''
Similarly in the Movie table, the year column doesn't look right.

As we can see below , there are I 2009, II 2009.

We can treat them all as 2009 by using substring.

CAST(SUBSTR(M.year,-4) AS UNSIGNED) year will return 'I 2009' AS numeri
c 2009.
```

```
'''
result = pd.read_sql_query("""
SELECT
   DISTINCT
   year
FROM
    Movie
""", conn)
print(result)
```

```
          year
0         2018
1         2012
2         2016
3         2017
4         2008
5        I 2009
6         1977
7         2013
8         2015
9         2007
10        2002
11        1951
12        2009
13        2014
14        2004
15        1997
16        1983
17        1994
18        2011
19        1996
20        2001
21        2010
22        2006
23        1971
24       I 2018
25    XVII 2016
26        1958
27       I 2017
```

```
28        1984
29        1987
..         ...
95      I 1968
96        1954
97      I 1980
98        1941
99     II 2008
100     I 1983
101     V 2015
102       1963
103       1931
104       1953
105     I 2001
106     I 1989
107       1948
108       1952
109    II 1998
110       1947
111     I 1992
112     I 2012
113       1936
114     I 1996
115       1946
116    IV 2011
117    II 1983
118    IV 2010
119    II 2011
120    IV 2017
121       1943
122       1950
123     I 1969
124    II 2009

[125 rows x 1 columns]
```

In [176]:
```
# Question 1 : Get list of directors who directed a 'Comedy' in Leap Ye
ar.
```

```
# Using where Genre = Comedy since only Comedy movies were asked.
# Need to use like '%Comedy%' if the question was genre consits of come
dy.

'''
Using WITH clauses makes our code more readble by splitting into smalle
r sub queries.

First we get all the movies with the genre as 'Comedy'.

Then we filter out the movies by year as leap year, we use CAST(SUBSTR
(M.year,-4) AS UNSIGNED)
to tackle the inconsistent data.

For all the movies in leap year with genre as 'Comedy' we get the direc
tors name from persons table.

We use Distinct TRIM(Name) to remove duplicate rows with extra spaces.

'''

list_of_directors = pd.read_sql_query("""
WITH
    Comedy_Movies AS
    (
        SELECT
             -- COUNT(*)  107
            MG.MID
        FROM
                    GENRE G
            JOIN    M_GENRE MG
            ON      G.GID = MG.GID
        WHERE
            TRIM(G.Name) = 'Comedy'
    ),
    Comedy_Movies_In_Leap_Yr AS
    (
        SELECT
            M.MID,
```

```
            M.title,
            CAST(SUBSTR(M.year,-4) AS UNSIGNED) year
        FROM
                    Comedy_Movies CM
            JOIN    Movie M
            ON      CM.MID = M.MID
        WHERE
            (CAST(SUBSTR(M.year,-4) AS UNSIGNED) %4 = 0 )
    )
   SELECT
        DISTINCT
        TRIM(P.Name) Director_Name,
        CM.title Movie_Name,
        CM.year
    FROM
                Comedy_Movies_In_Leap_Yr CM
        JOIN    M_Director MD
        ON      CM.MID = MD.MID
        JOIN    Person P
        ON      TRIM(MD.PID) = TRIM(P.PID)
    ORDER BY
        year
""", conn)

print(list_of_directors)
```

	Director_Name	Movie_Name	year
0	Pankaj Parashar	Ab Ayega Mazaa	1984
1	Kawal Sharma	Maalamaal	1988
2	Jabbar Patel	Ek Hota Vidushak	1992
3	Mahesh Bhatt	Papa Kahte Hain	1996
4	Bhagyaraj	Mr. Bechara	1996
5	Sachin	Navra Mazha Navsacha	2004
6	Govind Menon	Kis Kis Ki Kismat	2004
7	Siddharth Anand Kumar	Let's Enjoy	2004
8	Raj Kaushal	Shaadi Ka Laddoo	2004
9	Srinivas Bhashyam	Paisa Vasool	2004
10	Jagdish Rajpurohit	Bumboo	2012
11	Anand Balraj	Daal Mein Kuch Kaala Hai	2012
12	Karan Razdan	Mr Bhatti on Chutti	2012

```
13         Rajnish Thakur  Mere Dost Picture Abhi Baaki Hai    2012
14       Vickrant Mahajan                     Challo Driver    2012
15           Rakesh Mehta              Life Ki Toh Lag Gayi    2012
16           Nitin Kakkar                        Filmistaan    2012
17          Sameer Sharma      Luv Shuv Tey Chicken Khurana    2012
18            Aditya Datt                 Will You Marry Me    2012
19           Sachin Yardi           Kyaa Super Kool Hain Hum    2012
20         Abhishek Sharma     Tere Bin Laden: Dead Or Alive    2016
21           Umesh Ghadge              Kyaa Kool Hain Hum 3    2016
22           Milap Zaveri                        Mastizaade    2016
23         Sanjeev Sharma                     Saat Uchakkey    2016
24      Krishnadev Yagnik                    Days of Tafree    2016
25            Suhas Kadav         Motu Patlu: King of Kings    2016
26           Anees Bazmee                        Thank You  I 2011
```

In [3]:
```
# Question 2 : List all names of actors who playes in the movie 'Anand'

result = pd.read_sql_query("""
SELECT
    Distinct
   TRIM(P.Name) Actor_Name
FROM
            Movie M
    JOIN    M_Cast MC
    ON      M.MID = MC.MID
    JOIN    Person p
    ON      Trim(MC.PID) = Trim(P.PID)
WHERE
    M.title = 'Anand'
""", conn)
print(result)
```

```
          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
```

In [183]:

```
# Question 3 : List of all actors who acted oin a film before 1970 and
 in a fim after 1990.

result = pd.read_sql_query("""
WITH
    ACTORS_BEFORE_1970 AS
    (
        SELECT
            DISTINCT
            TRIM(MC.PID) PID
        FROM
                    Movie M
            JOIN    M_Cast MC
            ON      M.MID = MC.MID
        WHERE
            CAST(SUBSTR(M.year,-4) AS UNSIGNED) < 1970
    ),
    ACTORS_AFTER_1990 AS
    (
        SELECT
            DISTINCT
            TRIM(MC.PID) PID
        FROM
                    Movie M
            JOIN    M_Cast MC
            ON      M.MID = MC.MID
        WHERE
```

```
            CAST(SUBSTR(M.year,-4) AS UNSIGNED) > 1990
    )
    SELECT
        DISTINCT
        TRIM(P.Name) Actor_Name
    FROM
                ACTORS_BEFORE_1970 A_1970
        JOIN    ACTORS_AFTER_1990 A_1990
        ON      A_1970.PID = A_1990.PID
        JOIN    Person P
        ON      A_1970.PID = TRIM(P.PID)
""", conn)
print(result)
```

```
           Actor_Name
0         Rishi Kapoor
1      Amitabh Bachchan
2                Asrani
3         Zohra Sehgal
4       Parikshat Sahni
5         Rakesh Sharma
6           Sanjay Dutt
7             Ric Young
8                 Yusuf
9        Suhasini Mulay
10          A.K. Hangal
11        Jeremy Child
12        Farida Jalal
13       Waheeda Rehman
14        Rajesh Khanna
15           Ramesh Deo
16            Seema Deo
17       Asit Kumar Sen
18       Brahm Bhardwaj
19         Lalita Pawar
20           Dara Singh
21        Johnny Walker
22            Moolchand
23           Saira Banu
24          Prem Chopra
```

```
25          Dina Pathak
26       Achala Sachdev
27           Shashikala
28   Mohandas K. Gandhi
29     Jawaharlal Nehru
..                  ...
270      Gemini Ganesan
271                Aziz
272         Mohamad Ali
273         Master Amar
274               Gopal
275              Manish
276            Surendra
277           Raj Joshi
278              Nikita
279             Jaswant
280              Merlyn
281     Vikram Makandar
282     Lata Mangeshkar
283               Munni
284             Gummadi
285   Allu Ramalingaiah
286               Kaveri
287        Bharati Devi
288               Kumar
289                 Uma
290              Ismail
291         Miss Firoza
292                Dube
293               Dolly
294             Shekhar
295              Poonam
296       Jamila Massey
297         K.R. Vijaya
298               Sethi
299        Suryakantham

[300 rows x 1 columns]
```

In [59]:
```
# QUestion 4 : List all directors who directed 10 or more movies.

result = pd.read_sql_query("""
SELECT
    DISTINCT
    TRIM(P.NAME) DIRECTOR_NAME,
    NM.NUM_OF_MOVIES_DIRECTED
FROM
            (SELECT
                PID,
                COUNT(MID) NUM_OF_MOVIES_DIRECTED
            FROM
                M_Director
            GROUP BY
                PID
            HAVING
                NUM_OF_MOVIES_DIRECTED >= 10
            ) NM
    JOIN    PERSON P
    ON      TRIM(NM.PID) = TRIM(P.PID)
ORDER BY
    NM.NUM_OF_MOVIES_DIRECTED DESC
""", conn)
print(result)
```

	DIRECTOR_NAME	NUM_OF_MOVIES_DIRECTED
0	David Dhawan	39
1	Mahesh Bhatt	35
2	Priyadarshan	30
3	Ram Gopal Varma	30
4	Vikram Bhatt	29
5	Hrishikesh Mukherjee	27
6	Yash Chopra	21
7	Basu Chatterjee	19
8	Shakti Samanta	19
9	Subhash Ghai	18
10	Shyam Benegal	17
11	Abbas Alibhai Burmawalla	17
12	Rama Rao Tatineni	17
13	Manmohan Desai	16

14	Gulzar	16
15	Raj N. Sippy	16
16	Raj Kanwar	15
17	Mahesh Manjrekar	15
18	Indra Kumar	14
19	Raj Khosla	14
20	Rahul Rawail	14
21	Rajkumar Santoshi	14
22	Rakesh Roshan	13
23	Dev Anand	13
24	Vijay Anand	13
25	Harry Baweja	13
26	Anurag Kashyap	13
27	Ananth Narayan Mahadevan	13
28	K. Raghavendra Rao	13
29	Anees Bazmee	12
30	Guddu Dhanoa	12
31	Prakash Jha	12
32	Satish Kaushik	12
33	Nagesh Kukunoor	12
34	Prakash Mehra	12
35	Umesh Mehra	12
36	Anil Sharma	12
37	Madhur Bhandarkar	12
38	Rohit Shetty	12
39	Pramod Chakravorty	11
40	Sanjay Gupta	11
41	Nasir Hussain	11
42	Ketan Mehta	11
43	Govind Nihalani	11
44	Mohit Suri	11
45	Raj Kapoor	10
46	K. Bapaiah	10
47	Vishal Bhardwaj	10
48	N. Chandra	10
49	Tigmanshu Dhulia	10
50	J.P. Dutta	10
51	Mehul Kumar	10
52	Hansal Mehta	10

```
53              Sudhir Mishra                    10
54        K. Muralimohana Rao                    10
55            Pankaj Parashar                    10
56              J. Om Prakash                    10
57                  Bimal Roy                    10
```

In [187]:

```
# Question 5A) Number of movies in a year with all female actors.
# I'm considering None as not female

result = pd.read_sql_query("""
WITH
    MOVIES_WITH_NON_FEMALES AS
    (
        SELECT
            DISTINCT
            TRIM(MC.MID) MID
        FROM
                    M_Cast MC
            JOIN    Person P
            ON      TRIM(MC.PID) = TRIM(P.PID)
        WHERE
            TRIM(P.Gender) IN ('Male','None') -- Considering None as no
t female.
    )
    SELECT
        CAST(SUBSTR(M.year,-4) AS UNSIGNED) year,
        COUNT(DISTINCT TRIM(MID) ) NUM_OF_MOV_WITH_ONLY_FEMALES
    FROM
        Movie M
    WHERE
        TRIM(MID) NOT IN (SELECT MID FROM MOVIES_WITH_NON_FEMALES)
    GROUP BY
        CAST(SUBSTR(M.year,-4) AS UNSIGNED)
    ORDER BY
        year
""", conn)
print(result)
```

```
    year  NUM_OF_MOV_WITH_ONLY_FEMALES
```

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

In [197]:

```
# Question 5 B) Report for each year the % of movies in that year with
 only female actors and the
# total number of movies made that year.

result = pd.read_sql_query("""
WITH
    MOVIES_WITH_NON_FEMALES AS
    (
        SELECT
            DISTINCT
            TRIM(MC.MID) MID
        FROM
                    M_Cast MC
            JOIN    Person P
            ON      TRIM(MC.PID) = TRIM(P.PID)
        WHERE
            TRIM(P.Gender) IN ('Male','None') -- Considering None as no
t female.
    ),
    NUM_OF_MOV_WITH_ONLY_F_BY_YR AS
    (
    SELECT
        CAST(SUBSTR(M.year,-4) AS UNSIGNED) YEAR,
        COUNT(DISTINCT TRIM(MID) ) NUM_OF_MOV_WITH_ONLY_FEMALES
    FROM
        Movie M
    WHERE
        TRIM(MID) NOT IN (SELECT MID FROM MOVIES_WITH_NON_FEMALES)
    GROUP BY
        CAST(SUBSTR(M.year,-4) AS UNSIGNED)
    ),
    TOTAL_NUM_OF_MOV_BY_YR AS
```

```
    (
        SELECT
            CAST(SUBSTR(M.year,-4) AS UNSIGNED) YEAR,
            COUNT(DISTINCT TRIM(MID) ) TOTAL_NUM_OF_MOV
        FROM
            Movie M
        GROUP BY
            CAST(SUBSTR(M.year,-4) AS UNSIGNED)
    )
    SELECT
        TOT_MOV.YEAR,
        TOT_MOV.TOTAL_NUM_OF_MOV,
        ROUND((IFNULL(MOV_F.NUM_OF_MOV_WITH_ONLY_FEMALES,0) * 100 )/TOT
_MOV.TOTAL_NUM_OF_MOV,2) PERCENT_OF_MOV_WITH_ONLY_F
    FROM
                TOTAL_NUM_OF_MOV_BY_YR TOT_MOV
        LEFT OUTER JOIN
                NUM_OF_MOV_WITH_ONLY_F_BY_YR MOV_F
        ON      TRIM(TOT_MOV.YEAR) = TRIM(MOV_F.YEAR)
    ORDER BY
        PERCENT_OF_MOV_WITH_ONLY_F DESC
""", conn)
print(result)
```

	YEAR	TOTAL_NUM_OF_MOV	PERCENT_OF_MOV_WITH_ONLY_F
0	1939	2	50.0
1	1999	66	1.0
2	2000	64	1.0
3	2018	104	1.0
4	1931	1	0.0
5	1936	3	0.0
6	1941	1	0.0
7	1943	1	0.0
8	1946	2	0.0
9	1947	2	0.0
10	1948	3	0.0
11	1949	3	0.0
12	1950	2	0.0
13	1951	6	0.0
14	1952	6	0.0

15	1953	8	0.0
16	1954	6	0.0
17	1955	9	0.0
18	1956	6	0.0
19	1957	13	0.0
20	1958	9	0.0
21	1959	6	0.0
22	1960	14	0.0
23	1961	7	0.0
24	1962	12	0.0
25	1963	10	0.0
26	1964	15	0.0
27	1965	14	0.0
28	1966	18	0.0
29	1967	19	0.0
..	...	...	...
48	1986	33	0.0
49	1987	32	0.0
50	1988	37	0.0
51	1989	47	0.0
52	1990	42	0.0
53	1991	41	0.0
54	1992	58	0.0
55	1993	63	0.0
56	1994	60	0.0
57	1995	56	0.0
58	1996	60	0.0
59	1997	55	0.0
60	1998	55	0.0
61	2001	73	0.0
62	2002	87	0.0
63	2003	103	0.0
64	2004	103	0.0
65	2005	129	0.0
66	2006	101	0.0
67	2007	109	0.0
68	2008	107	0.0
69	2009	110	0.0
70	2010	125	0.0

```
71  2011              116                       0.0
72  2012              111                       0.0
73  2013              136                       0.0
74  2014              126                       0.0
75  2015              119                       0.0
76  2016              129                       0.0
77  2017              126                       0.0

[78 rows x 3 columns]
```

In [199]:

```sql
# Question 6) Return the film with the largest cast
result = pd.read_sql_query("""
WITH
    CAST_NUMBER AS
    (
        SELECT
            TRIM(MID) MID,
            COUNT(DISTINCT TRIM(PID)) NUM_OF_PEOPLE
        FROM
            M_Cast
        GROUP BY
            TRIM(MID)
    )
    SELECT
        M.MID,
        M.title,
        CM.NUM_OF_PEOPLE
    FROM
                CAST_NUMBER CM
        JOIN    Movie M
        ON      CM.MID = TRIM(M.MID)
    WHERE
        CM.NUM_OF_PEOPLE  = (
            SELECT
                MAX(NUM_OF_PEOPLE)
            FROM
                CAST_NUMBER
        )
```

```
""", conn)
print(result)
```

```
         MID          title  NUM_OF_PEOPLE
0  tt5164214  Ocean's Eight            238
```

In [205]:

```
# Question 7) Decade with the largest number of films and the total num
ber of films in that decade.
result = pd.read_sql_query("""
WITH
    DISTINCT_YEARS AS
    (
    SELECT
        DISTINCT
        CAST(SUBSTR(year,-4) AS UNSIGNED) YEAR,
        CAST(SUBSTR(year,-4) AS UNSIGNED) START_OF_DECADE,
        CAST(SUBSTR(year,-4) AS UNSIGNED)+9 END_OF_DECADE,
        'Decade of : ' || SUBSTR(year,-4)   DECADE
    FROM
        Movie
    ),
    NUMBER_OF_MOV_BY_YR AS
    (
    SELECT
    COUNT(DISTINCT MID) NUM_OF_MOV,
    CAST(SUBSTR(year,-4) AS UNSIGNED) YEAR
    FROM
        Movie
    GROUP BY
        CAST(SUBSTR(year,-4) AS UNSIGNED)
    ),
    NUM_OF_MOV_IN_DECADE AS
    (
    SELECT
        SUM(NUM_OF_MOV) TOTAL_MOVIES,
        DY.DECADE
    FROM
        NUMBER_OF_MOV_BY_YR NM,
        DISTINCT_YEARS DY
```

```
    WHERE
        NM.YEAR BETWEEN DY.START_OF_DECADE AND DY.END_OF_DECADE
    GROUP BY
        DY.DECADE
    )
    SELECT
        DECADE,
        TOTAL_MOVIES
    FROM
        NUM_OF_MOV_IN_DECADE
    WHERE
        TOTAL_MOVIES = (
            SELECT
                MAX(TOTAL_MOVIES)
            FROM
                NUM_OF_MOV_IN_DECADE
            )
""", conn)
print(result)
```

```
              DECADE  TOTAL_MOVIES
0  Decade of : 2008          1205
```

In [271]:

```
"""
* 8) Find Actors that were never unemployed for more than 3 years.
*
*   Assumtions  :
*
*   A) I'm considering only people who have worked for more than one ye
ar.
*   B) Considering the time period between min and max years of the act
or.
*      i.e if the actor has been working from 1990 to 2000 and the acto
r
*      acted only in 1990, 1991 1996, 1998 and 2000. Then he is conside
red as
*      unemployed for more than 3 yrs ( > 3 yrs => atleast 4 years).
*
*   Logic for solving :
```

```
*       Calculate the total num of movies that the actor acted from his
 min year i.e 1990 to 1991,
*       let say it's 3 and then calculate the total num of movies he ac
ted from his min year 1990
*       to 1991+4 (1995) and it comes back as 3 since he hasn't made an
y movies between 1991 and 1995,
*       this means that he has been unemployed for more than 3 years (4
 years) 1992,1993,1994,1995.
*       Therefore we don't consider him.
"""
result = pd.read_sql_query("""

WITH
    NUM_OF_MOV_FOR_AN_ACTR_BY_YR AS
    (
        SELECT
            TRIM(MC.PID) PID,
            CAST(SUBSTR(year,-4) AS UNSIGNED) YEAR,
            COUNT(DISTINCT TRIM(M.MID)) NUM_OF_MOV
        FROM
            M_Cast MC,
            Movie M
        WHERE
            TRIM(MC.MID) = TRIM(M.MID)
        GROUP BY
            TRIM(MC.PID),
            CAST(SUBSTR(year,-4) AS UNSIGNED)
        ORDER BY
            NUM_OF_MOV DESC
    ),
    ACTRS_FOR_MORE_THAN_ONE_YR AS
    (
        SELECT
            PID,
            COUNT(YEAR) AS NUM_OF_YEARS,
            MIN(YEAR) AS MIN_YEAR,
            MAX(YEAR) AS MAX_YEAR
        FROM
            NUM_OF_MOV_FOR_AN_ACTR_BY_YR
```

```
        GROUP BY
            PID
        HAVING
            COUNT(YEAR) > 1
    ),
    NUM_OF_FOR_ACTR_W_MRE_THN_1_YR AS
    (
        SELECT
            NM.PID,
            NM.YEAR,
            NM.YEAR+4 AS YEAR_PLUS_4,
            NM.NUM_OF_MOV,
            AY.MIN_YEAR,
            AY.MAX_YEAR
        FROM
            NUM_OF_MOV_FOR_AN_ACTR_BY_YR NM,
            ACTRS_FOR_MORE_THAN_ONE_YR AY
        WHERE
            NM.PID = AY.PID
    ),
    NUM_OF_MOV_TILL_DATE_BY_ACTOR AS
    (
        SELECT
            NA.PID,
            NY.YEAR,
            SUM(NA.NUM_OF_MOV) AS NUM_OF_MOV_TILL_THAT_YEAR
        FROM
            NUM_OF_FOR_ACTR_W_MRE_THN_1_YR NA,
            NUM_OF_FOR_ACTR_W_MRE_THN_1_YR NY
        WHERE
            NA.PID = NY.PID AND
            NA.YEAR BETWEEN NY.MIN_YEAR AND  NY.YEAR
        GROUP BY
            NA.PID,
            NY.YEAR
    ),
    NUM_OF_MV_BY_ACTR_BY_YR_PLS_4 AS
    (
        SELECT
```

```
            NA.PID,
            NY.YEAR,
            SUM(NA.NUM_OF_MOV) AS NUM_OF_MOV_TILL_AS_OF_YR_PLS_4
        FROM
            NUM_OF_FOR_ACTR_W_MRE_THN_1_YR NA,
            NUM_OF_FOR_ACTR_W_MRE_THN_1_YR NY
        WHERE
            NA.PID = NY.PID AND
            NA.YEAR BETWEEN NY.MIN_YEAR AND  NY.YEAR_PLUS_4 AND
            NY.YEAR_PLUS_4 <= NY.MAX_YEAR
        GROUP BY
            NA.PID,
            NY.YEAR
    )
    SELECT
        DISTINCT
        TRIM(P.Name) AS ACTORS_NEVER_UNEMPLOYED_FOR_MORE_THAN_3_YRS
    FROM
        Person P
    WHERE
        TRIM(P.PID) NOT IN
            (
            SELECT
                DISTINCT
                NMT.PID
            FROM
                NUM_OF_MOV_TILL_DATE_BY_ACTOR NMT,
                NUM_OF_MV_BY_ACTR_BY_YR_PLS_4 NMP
            WHERE
                NMT.PID = NMP.PID AND
                NMT.YEAR = NMP.YEAR AND
                NMT.NUM_OF_MOV_TILL_THAT_YEAR = NMP.NUM_OF_MOV_TILL_AS_
OF_YR_PLS_4
            )

""", conn)
print(result)
```

```
      ACTORS_NEVER_UNEMPLOYED_FOR_MORE_THAN_3_YRS
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	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
32578                                G. Ram Prasad
32579                                  Raja Chanda
32580                               Deepak Ramteke
32581                                 Kamika Verma
32582                          Dhorairaj Bhagavan
32583                                 Nasir Shaikh
32584                                 Adrian Fulle

[32585 rows x 1 columns]
```

In [11]:

```
"""
 Question 9. Find all the actors that made more movies with Yash Chopra
 than any other director.
"""
result = pd.read_sql_query("""
WITH
    YASH_CHOPRAS_PID AS
    (
        SELECT
            TRIM(P.PID) AS PID
        FROM
            Person P
        WHERE
            Trim(P.Name) = 'Yash Chopra'
    ),
```

```
    NUM_OF_MOV_BY_ACTOR_DIRECTOR AS
    (
        SELECT
            TRIM(MC.PID) ACTOR_PID,
            TRIM(MD.PID) DIRECTOR_PID,
            COUNT(DISTINCT TRIM(MD.MID)) AS NUM_OF_MOV
        FROM
            M_Cast MC,
            M_Director MD
        WHERE
            TRIM(MC.MID)= TRIM(MD.MID)
        GROUP BY
            ACTOR_PID,
            DIRECTOR_PID
    ),
    NUM_OF_MOVIES_BY_YC AS
    (
        SELECT
            NM.ACTOR_PID,
            NM.DIRECTOR_PID,
            NM.NUM_OF_MOV NUM_OF_MOV_BY_YC
        FROM
            NUM_OF_MOV_BY_ACTOR_DIRECTOR NM,
            YASH_CHOPRAS_PID YCP
        WHERE
            NM.DIRECTOR_PID = YCP.PID
    ),
    MAX_MOV_BY_OTHER_DIRECTORS AS
    (
        SELECT
            ACTOR_PID,
            MAX(NUM_OF_MOV) MAX_NUM_OF_MOV
        FROM
            NUM_OF_MOV_BY_ACTOR_DIRECTOR NM,
            YASH_CHOPRAS_PID YCP
        WHERE
            NM.DIRECTOR_PID <> YCP.PID
        GROUP BY
            ACTOR_PID
```

```
    ),
    ACTORS_MOV_COMPARISION AS
    (
    SELECT
        NMY.ACTOR_PID,
        CASE WHEN NMY.NUM_OF_MOV_BY_YC > IFNULL(NMO.MAX_NUM_OF_MOV,0) T
HEN 'Y' ELSE 'N' END MORE_MOV_BY_YC
    FROM
        NUM_OF_MOVIES_BY_YC NMY
        LEFT OUTER JOIN
        MAX_MOV_BY_OTHER_DIRECTORS NMO
        ON
        NMY.ACTOR_PID = NMO.ACTOR_PID
    )
    SELECT
        DISTINCT
        TRIM(P.Name) ACTOR_NAME
    FROM
        Person P
    WHERE
        TRIM(P.PID) IN (
            SELECT
                DISTINCT
                ACTOR_PID
            FROM
                ACTORS_MOV_COMPARISION
            WHERE
                MORE_MOV_BY_YC = 'Y'
        )
""", conn)
print(result)
```

```
                ACTOR_NAME
0           Waheeda Rehman
1           Achala Sachdev
2              Yash Chopra
3              Vinod Negi
4       Chandni Jas Keerat
5            Shivaya Singh
6          Huzefa Gadiwala
```

```
7              Manish Arora
8              Pankaj Raina
9               Neetu Singh
10            Julia St John
11            Susan Fordham
12                Steve Box
13           Jasmine Jardot
14             Abbie Murphy
15            Julie Vollono
16             Varun Thakur
17           Katy Kartwheel
18              Lucy Phelps
19   Matthew David McCarthy
20           Benjayx Murphy
21          Melissa Hollett
22      Nick Thomas-Webster
23              Anick Wiget
24               Jay Conroy
25            Gary Wronecki
26                Sean Moon
27     Rudy Valentino Grant
28          Richard Herdman
29     James Michael Rankin
..                      ...
75        Chandu Allahabadi
76              Shyam Arora
77            Leela Chitnis
78               Ravi Dubey
79              Ashok Verma
80             Pratima Puri
81          Ramola Bachchan
82             Sumati Gupte
83           Akhtar-Ul-Iman
84               Pran Mehra
85                   Nissar
86            Master Rizwan
87                   Naseem
88                 Parijat
89                   Nazir
```

```
90                Ram Maini
91                 Om Sahni
92                    Bhola
93                 Ramanand
94                    Gopal
95                   Kishan
96                    Nasir
97             Ashok Chadda
98                   Rajesh
99             Master Kelly
100              Yasin Khan
101         Sandow S. Sethi
102                   Naval
103               Prem Sood
104         Ramlal Shyamlal

[105 rows x 1 columns]
```

In [25]:

```
"""
*  10.  The Shahrukh number of an actor is the length of the shortest p
ath
*       between the actor and Shahrukh Khan in the "co-acting" graph. T
hat
*       is, Shahrukh Khan has Shahrukh number 0; all actors who acted i
n
*       the same film as Shahrukh have Shahrukh number 1; all actors wh
o
*       acted in the same film as some actor with Shahrukh number 1 hav
e
*       Shahrukh number 2, etc. Return all actors whose Shahrukh number
 is 2.
"""
result = pd.read_sql_query("""
WITH
    SHAHRUKH_0 AS
    (
        SELECT
            TRIM(P.PID) PID
        FROM
```

```
            Person P
        WHERE
            Trim(P.Name) like '%Shahrukh%'
    ),
    SHAHRUKH_1_MOVIES AS
    (
        SELECT
            DISTINCT
            TRIM(MC.MID) MID,
            S0.PID
        FROM
            M_Cast MC,
            SHAHRUKH_0 S0
        WHERE
            TRIM(MC.PID) = S0.PID
    ),
    SHAHRUKH_1_ACTORS AS
    (
        SELECT
            DISTINCT
            TRIM(MC.PID) PID
        FROM
            M_Cast MC,
            SHAHRUKH_1_MOVIES S1M
        WHERE
            TRIM(MC.MID) = S1M.MID AND
            TRIM(MC.PID) <> S1M.PID
    ),
    SHAHRUKH_2_MOVIES AS
    (
        SELECT
            DISTINCT
            TRIM(MC.MID) MID,
            S1A.PID
        FROM
            M_Cast MC,
            SHAHRUKH_1_ACTORS S1A
        WHERE
            TRIM(MC.PID) = S1A.PID
```

```
    )
    SELECT
        DISTINCT
        TRIM(MC.PID) PID,
        TRIM(P.Name) ACTOR_NAME
    FROM
        Person P,
        M_Cast MC,
        SHAHRUKH_2_MOVIES S2M
    WHERE
            TRIM(MC.PID) = TRIM(P.PID) AND
            TRIM(MC.MID) = S2M.MID AND
            TRIM(MC.PID) <> S2M.PID;

""", conn)
print(result)
```

	PID	ACTOR_NAME
0	nm2951768	Freida Pinto
1	nm6467532	Caroline Christl Long
2	nm6071249	Rajeev Pahuja
3	nm3491108	Michelle Santiago
4	nm7509518	Jandre le Roux
5	nm5951787	Raj Awasti
6	nm5525290	Michael Chapman
7	nm8232648	James Heron
8	nm7247557	Alex Jaep
9	nm6631007	Marian Lorencik
10	nm7255636	Celina Nessa
11	nm5721141	James Pimenta
12	nm4964257	M'laah Kaur Singh
13	nm0380073	Maximiliano Hernández
14	nm3630374	Sohum Shah
15	nm3708961	Deepak Damle
16	nm8334880	Piyush Kaushik
17	nm1390115	Harish Khanna
18	nm3818286	Sushant Singh Rajput
19	nm0080232	Nitish Bharadwaj
20	nm8644385	Lalu Makhija

21	nm6661769	Mir Sarwar
22	nm4731677	Ayushmann Khurrana
23	nm0007102	Tabu
24	nm2331000	Radhika Apte
25	nm0223521	Anil Dhawan
26	nm2435847	Manav Vij
27	nm1817397	Ashwini Kalsekar
28	nm3777127	Chhaya Kadam
29	nm1664541	Zakir Hussain
...	...	...
15881	nm2019990	Pradhan Manjari
15882	nm2021136	Poonam Jha
15883	nm1459053	Sunila Karambelkar
15884	nm1763457	Arup Ganguli
15885	nm1760405	Laxmi Patel
15886	nm1760399	Meena Pankaj
15887	nm1686154	Pratap
15888	nm2184586	Vidya Shenoy
15889	nm2182462	Jeetendra Khanna
15890	nm2177257	K.L. Sethi
15891	nm2084775	Malaika Shinoy
15892	nm2465676	Poonam Bajwa
15893	nm1688585	Kishin Punjabi
15894	nm1082198	Surjeet Redi
15895	nm0695939	Premji
15896	nm1693988	Kamu
15897	nm5578623	Monal
15898	nm1524755	Ushma Rathod
15899	nm1567918	Shilpi
15900	nm1946131	Zubeda Khan
15901	nm2519512	N. Sagar
15902	nm4042918	Habib Tanvar
15903	nm1881395	Mohd. Zahiruddin
15904	nm2522571	Muktha George
15905	nm0030135	Anjuman
15906	nm3099317	Dhruv Shetty
15907	nm2371614	Hayley Cleghorn
15908	nm2675737	Nirvasha Jithoo
15909	nm2370589	Kamal Maharshi

```
15910  nm1866356          Mohini Manik

[15911 rows x 2 columns]
```