SQL Problem statements

- Query a database based on given statements

In [92]:

```
import pandas as pd
import sqlite3

from IPython.display import display, HTML
```

In [93]:

```
# Note that this is not the same db we have used in course videos, please download from this link
# https://drive.google.com/file/d/1O-1-L1DdNxEK6O6nG2jS31MbrMh-OnXM/view?usp=sharing
```

In [94]:

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

Overview of all tables

In [95]:

```
tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE type='table'",conn)
tables = tables["Table_Name"].values.tolist()
```

In [96]:

```
for table in tables:
    query = "PRAGMA TABLE_INFO({})".format(table)
    schema = pd.read_sql_query(query,conn)
    print("Schema of",table)
    display(schema)
    print("-"*100)
    print("\n")
```

```
Schema of Movie
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	title	TEXT	0	None	0
3	3	year	TEXT	0	None	0
4	4	rating	REAL	0	None	0
5	5	num_votes	INTEGER	0	None	0

```
----------------------------------------------------------------------------------------------------


Schema of Genre
```

In [97]:

```
tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE type='table'",conn)
```

In [98]:

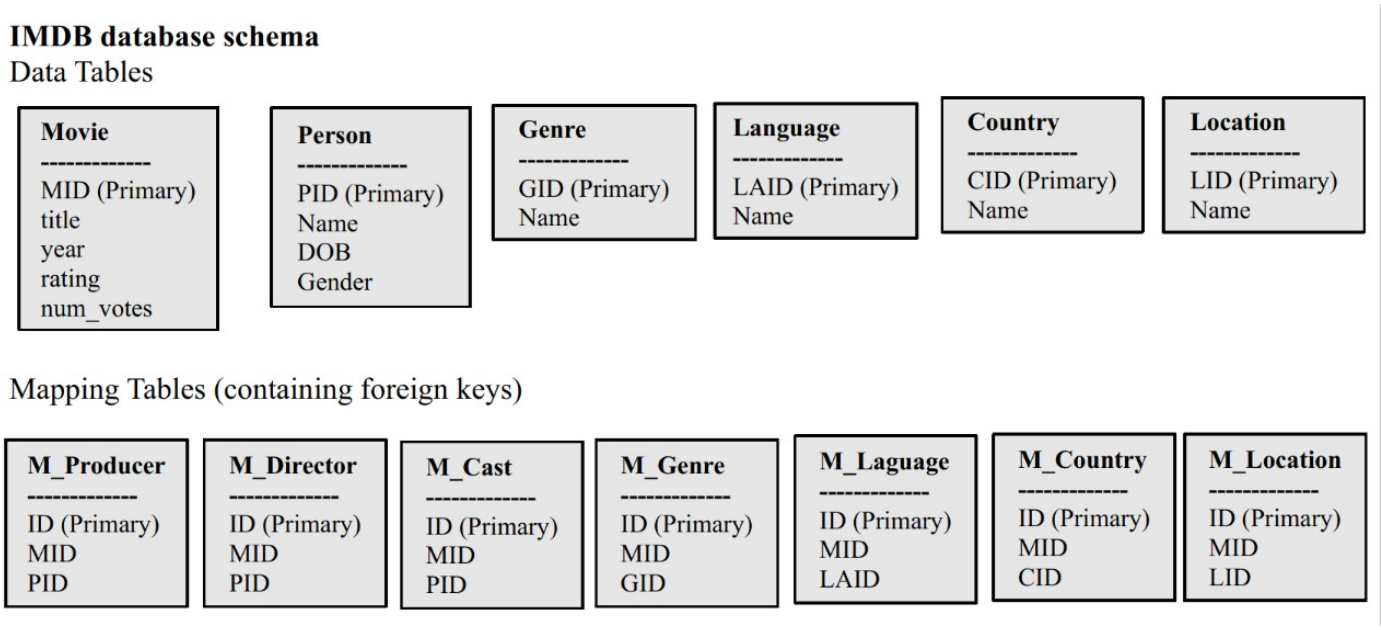
```
tables
```

Out[98]:

	Table_Name
0	Movie
1	Genre
2	Language
3	Country
4	Location
5	M_Location
6	M_Country
7	M_Language
8	M_Genre
9	Person
10	M_Producer
11	M_Director
12	M_Cast

Useful tips:

1. the year column in 'Movie' table, will have few chracters other than numbers which you need to be preprocessed, you need to get a substring of last 4 characters, its better if you convert it as int type, ex: CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)
2. For almost all the TEXT columns we have show, please try to remove trailing spaces, you need to use TRIM() function
3. When you are doing count(coulmn) it won't consider the "NULL" values, you might need to explore other alternatives like Count(*)



Q1 --- 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.

To determine whether a year is a leap year, follow these steps:

- **STEP-1:** If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
- **STEP-2:** If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
- **STEP-3:** If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
- **STEP-4:** The year is a leap year (it has 366 days).
- **STEP-5:** The year is not a leap year (it has 365 days).

Year 1900 is divisible by 4 and 100 but it is not divisible by 400, so it is not a leap year.

In [99]:

```
%%time
def grader_1(q1):
    q1_results  = pd.read_sql_query(q1,conn)
    print(q1_results.head(10))
    assert (q1_results.shape == (232,3))

query1 = """

SELECT TRIM(p.name) AS director_name,TRIM(m.title) AS title,m.year
FROM Movie AS m
JOIN m_director AS d
    ON m.mid==d.mid
JOIN person AS p
    ON p.PID==d.PID
JOIN m_genre AS g
    ON g.id==d.id
JOIN genre
    ON genre.gid==g.gid

WHERE (CAST(SUBSTR(TRIM(year),-4) AS INTEGER)) % 4 == 0
        AND ((CAST(SUBSTR(TRIM(year),-4) AS INTEGER)) % 100 != 0
        OR (CAST(SUBSTR(TRIM(year),-4) AS INTEGER)) % 400 == 0 )
        AND genre.name like "%comedy%"  """



grader_1(query1)
```

```
      director_name                            title  year
0      Milap Zaveri                        Mastizaade  2016
1      Danny Leiner  Harold & Kumar Go to White Castle  2004
2    Anurag Kashyap                Gangs of Wasseypur  2012
3      Frank Coraci        Around the World in 80 Days  2004
4     Griffin Dunne             The Accidental Husband  2008
5       Anurag Basu                           Barfi!  2012
6   Gurinder Chadha                 Bride & Prejudice  2004
7        Mike Judge    Beavis and Butt-Head Do America  1996
8  Tarun Mansukhani                           Dostana  2008
9      Shakun Batra                      Kapoor & Sons  2016
CPU times: total: 141 ms
Wall time: 153 ms
```

Q2 --- List the names of all the actors who played in the movie 'Anand' (1971)

In [100]:

```
%%time
def grader_2(q2):
    q2_results  = pd.read_sql_query(q2,conn)
    print(q2_results.head(10))
    assert (q2_results.shape == (17,1))


query2 = """

SELECT TRIM(p.name) actor_names
FROM person p
WHERE p.pid IN
    (
    SELECT TRIM(mc.pid)
    FROM m_cast mc
    WHERE mc.mid
    IN
        (
        SELECT TRIM(m.mid)
        FROM movie m
        WHERE m.title=="Anand"
        )
    )

"""



grader_2(query2)
```

```
        actor_names
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
CPU times: total: 46.9 ms
Wall time: 51.5 ms
```

Q3 --- List all the actors who acted in a film before 1970 and in a film after 1990. (That is: < 1970 and > 1990.)

In [101]:

```
%%time

def grader_3a(query_less_1970, query_more_1990):
    q3_a = pd.read_sql_query(query_less_1970,conn)
    print(q3_a.shape)
    q3_b = pd.read_sql_query(query_more_1990,conn)
    print(q3_b.shape)
    return (q3_a.shape == (4942,1)) and (q3_b.shape == (62570,1))

query_less_1970 ="""

SELECT p.PID FROM Person p
JOIN
(
    SELECT TRIM(mc.PID) PD
    FROM M_cast mc
    WHERE mc.MID in
       (
        SELECT mv.MID
        FROM Movie mv
        WHERE CAST(SUBSTR(mv.year,-4) AS Integer)<1970
       )
) r1
ON r1.PD=p.PID

"""
query_more_1990 ="""

SELECT p.PID FROM Person p
JOIN
(
    SELECT TRIM(mc.PID) PD
    FROM M_cast mc
    WHERE mc.MID in
        (
        SELECT mv.MID
        FROM Movie mv
        WHERE CAST(SUBSTR(mv.year,-4) AS Integer)>1990
        )
) r1
ON r1.PD=p.PID

"""
print(grader_3a(query_less_1970, query_more_1990))

# using the above two queries, you can find the answer to the given question
```

```
(4942, 1)
(62570, 1)
True
CPU times: total: 734 ms
Wall time: 779 ms
```

In [102]:

```
%%time
def grader_3(q3):
    q3_results  = pd.read_sql_query(q3,conn)
    print(q3_results.head(10))
    print(q3_results.shape)
    assert (q3_results.shape == (300,1))

query3 = """
WITH
first AS
    (
    SELECT p.PID, TRIM(p.name) AS actor_name FROM Person p
    JOIN
    (
        SELECT TRIM(mc.PID) PD
        FROM M_cast mc
        WHERE TRIM(mc.MID) in
       (
        SELECT TRIM(mv.MID)
        FROM Movie mv
        WHERE CAST(SUBSTR(mv.year,-4) AS Integer)<1970
       )
    ) r1
    on r1.PD=p.PID ),

second AS
    (
    SELECT p.PID,  TRIM(p.name) AS actor_name FROM Person p
    JOIN
    (
        SELECT TRIM(mc.PID) PD
        FROM M_cast mc
        WHERE TRIM(mc.MID) in
        (
        SELECT TRIM(mv.MID)
        FROM Movie mv
        WHERE CAST(SUBSTR(mv.year,-4) AS Integer)>1990
        )
    ) r1
    on r1.PD=p.PID)

SELECT name AS actor_name
FROM person p
WHERE p.pid in
    (
    SELECT first.pid
    FROM first
    INTERSECT
    SELECT second.pid
    FROM second
    )


"""
grader_3(query3)
```

```
          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
(300, 1)
CPU times: total: 625 ms
Wall time: 623 ms
```

Q4 --- 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 [103]:

```
%%time

def grader_4a(query_4a):
    query_4a = pd.read_sql_query(query_4a,conn)
    print(query_4a.head(10))
    return (query_4a.shape == (1462,2))

query_4a ="""
SELECT md.pid director_id ,count(*) movie_count
FROM m_director as md
JOIN movie m
    ON md.mid==m.mid
GROUP BY director_id
"""
print(grader_4a(query_4a))

# using the above query, you can write the answer to the given question
```

```
  director_id  movie_count
0   nm0000180            1
1   nm0000187            1
2   nm0000229            1
3   nm0000269            1
4   nm0000386            1
5   nm0000487            2
6   nm0000965            1
7   nm0001060            1
8   nm0001162            1
9   nm0001241            1
True
CPU times: total: 31.2 ms
Wall time: 27.4 ms
```

In [104]:

```
%%time
def grader_4(q4):
    q4_results  = pd.read_sql_query(q4,conn)
    print(q4_results.head(10))
    assert (q4_results.shape == (58,2))

query4 = """

SELECT trim(p.name),count(*) as number_of_movies
FROM person as p
JOIN
    (
    SELECT md.pid
    FROM m_director as md
    ) t1
    ON t1.pid==p.pid
GROUP BY p.name
HAVING number_of_movies >= 10
ORDER BY number_of_movies DESC


"""
grader_4(query4)
```

```
           trim(p.name)  number_of_movies
0          David Dhawan                39
1          Mahesh Bhatt                36
2          Priyadarshan                30
3       Ram Gopal Varma                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
CPU times: total: 62.5 ms
Wall time: 61.5 ms
```

Q5.a --- For each year, count the number of movies in that year that had only female actors.

In [105]:

```
%%time

# note that you don't need TRIM for person table

def grader_5aa(query_5aa):
    query_5aa = pd.read_sql_query(query_5aa,conn)
    print(query_5aa.head(10))
    return (query_5aa.shape == (8846,3))

query_5aa ="""
SELECT mc.mid,p.gender,count(*) count
FROM person p
JOIN m_cast mc
    ON p.pid==trim(mc.pid)
GROUP BY mc.mid,p.gender

"""
print(grader_5aa(query_5aa))

def grader_5ab(query_5ab):
    query_5ab = pd.read_sql_query(query_5ab,conn)
    print(query_5ab.head(10))
    return (query_5ab.shape == (3469, 3))

query_5ab ="""

SELECT mc.mid,p.gender g,count(*) count
FROM person p
JOIN m_cast mc
    ON p.pid==trim(mc.pid)
WHERE trim(gender) LIKE "male"
GROUP BY mc.mid,p.gender

"""

print(grader_5ab(query_5ab))


# using the above queries, you can write the answer to the given question
```

```
         MID  Gender  count
0  tt0021594    None      1
1  tt0021594  Female      3
2  tt0021594    Male      5
3  tt0026274    None      2
4  tt0026274  Female     11
5  tt0026274    Male      9
6  tt0027256    None      2
7  tt0027256  Female      5
8  tt0027256    Male      8
9  tt0028217  Female      3
True
         MID     g  count
0  tt0021594  Male      5
1  tt0026274  Male      9
2  tt0027256  Male      8
3  tt0028217  Male      7
4  tt0031580  Male     27
5  tt0033616  Male     46
6  tt0036077  Male     11
7  tt0038491  Male      7
8  tt0039654  Male      6
9  tt0040067  Male     10
True
CPU times: total: 953 ms
Wall time: 966 ms
```

In [106]:

```
%%time
def grader_5a(q5a):
    q5a_results  = pd.read_sql_query(q5a,conn)
    print(q5a_results.head(10))
    assert (q5a_results.shape == (4,2))     #reference taken from        #https://stackoverflow.com/a/67030339/17345549

query5a = """


WITH
male_present AS (
                SELECT m.mid mpmid
                    FROM Movie m
                    JOIN M_Cast mc
                        ON m.MID = mc.MID
                    JOIN Person p
                        ON trim(mc.PID) = p.PID
                    WHERE Gender = 'Male'
                )


SELECT year, COUNT(DISTINCT mid) as Female_only_Casted_count
FROM
    (
    SELECT SUBSTR(m.year,-4) as 'Year', trim(m.MID) as MID
        FROM Movie m
        JOIN M_Cast mc
            ON m.MID = mc.MID
        JOIN Person p
            ON TRIM(mc.PID) = p.PID
    ) t1

WHERE TRIM(t1.mid) NOT IN (SELECT mpmid FROM male_present )
GROUP BY year

"""
grader_5a(query5a)
```

```
   Year  Female_only_Casted_count
0  1939                         1
1  1999                         1
2  2000                         1
3  2018                         1
CPU times: total: 1 s
Wall time: 1.09 s
```

Q5.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 [107]:

```
%%time
def grader_5b(q5b):
    q5b_results  = pd.read_sql_query(q5b,conn)
    print(q5b_results.head(10))
    assert (q5b_results.shape == (4,3))


query5b= """

WITH

Movie_Non_Females AS
    (
    SELECT DISTINCT TRIM(MC.mid) mid
    FROM m_cast MC
    JOIN person P ON TRIM(MC.pid) = P.pid
    WHERE TRIM(P.gender) IN ('Male','None')
    ),

Movie_female_year AS
    (
    SELECT CAST(SUBSTR(M.year,-4) AS INTEGER) year,
    COUNT(DISTINCT TRIM(mid)) Female_Movie_Only
    FROM movie M
    WHERE TRIM(MID) NOT IN (SELECT mid FROM Movie_Non_Females)
    GROUP BY CAST(SUBSTR(M.year,-4) AS INTEGER)
    ),

movies_year AS
    (
    SELECT CAST(SUBSTR(M.year,-4) AS INTEGER) Year,
    COUNT(DISTINCT TRIM(mid)) Total_Movies
    FROM movie M
    GROUP BY CAST(SUBSTR(M.YEAR,-4) AS INTEGER)
    )

SELECT MY.YEAR, CAST(MF.Female_Movie_Only as real)/trim(MY.Total_Movies) Female_Movie_Percentage, MY.Total_Movies
FROM movies_year MY
JOIN Movie_female_year MF
    ON TRIM(MY.year) = TRIM(MF.year)
ORDER BY Female_Movie_Percentage DESC

                                        """
grader_5b(query5b)
```

```
   Year  Female_Movie_Percentage  Total_Movies
0  1939                 0.500000             2
1  2000                 0.015625            64
2  1999                 0.015152            66
3  2018                 0.009615           104
CPU times: total: 359 ms
Wall time: 343 ms
```

Q6 --- 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 [108]:

```
%%time
def grader_6(q6):
    q6_results  = pd.read_sql_query(q6,conn)
    print(q6_results.head(10))
    assert (q6_results.shape == (3473, 2))

query6 ="""

WITH
cast_p AS
        (
        SELECT MC.mid,COUNT(DISTINCT(MC.pid)) cast_count
        FROM m_cast MC
        GROUP BY MC.mid
        )

SELECT M.title, C.cast_count
FROM movie M
JOIN cast_p C
    ON C.MID=M.MID
ORDER BY cast_count DESC

"""
grader_6(query6)
```

```
                       title  cast_count
0               Ocean's Eight         238
1                    Apaharan         233
2                        Gold         215
3             My Name Is Khan         213
4  Captain America: Civil War         191
5                    Geostorm         170
6                     Striker         165
7                        2012         154
8                      Pixels         144
9       Yamla Pagla Deewana 2         140
CPU times: total: 188 ms
Wall time: 192 ms
```

Q7 --- A decade is a sequence of 10 consecutive years.

For example, say in your database you have movie information starting from 1931.

the first decade is 1931, 1932, ..., 1940,

the second decade is 1932, 1933, ..., 1941 and so on.

Find the decade D with the largest number of films and the total number of films in D

In [109]:

```
%%time
def grader_7a(q7a):
    q7a_results  = pd.read_sql_query(q7a,conn)
    print(q7a_results.head(10))
    assert (q7a_results.shape == (78, 2))

query7a = """
SELECT CAST(SUBSTR(M.YEAR,-4) AS integer) Year, COUNT(DISTINCT TRIM(mid)) Total_Movies
FROM MOVIE M
GROUP BY CAST(SUBSTR(M.YEAR,-4) AS integer)
ORDER BY Total_Movies DESC
          """
grader_7a(query7a)

# using the above query, you can write the answer to the given question
```

```
   Year  Total_Movies
0  2013           136
1  2016           129
2  2005           129
3  2017           126
4  2014           126
5  2010           125
6  2015           119
7  2011           116
8  2012           110
9  2009           109
CPU times: total: 15.6 ms
Wall time: 7.98 ms
```

In [110]:

```
%%time
def grader_7b(q7b):
    q7b_results  = pd.read_sql_query(q7b,conn)
    print(q7b_results.head(10))
    assert (q7b_results.shape == (713, 4))

query7b = """
WITH

table1 AS
    (
    SELECT DISTINCT
        CAST(SUBSTR(m.year,-4) AS INTEGER) year,
        CAST(SUBSTR(m.year,-4) AS INTEGER) decade_start,
        CAST(SUBSTR(m.year,-4) AS INTEGER) + 9 decade_end,
        COUNT(DISTINCT TRIM(MID)) Total_Movies
    FROM movie m
    GROUP BY CAST(SUBSTR(M.YEAR,-4) AS INTEGER)
    ORDER BY CAST(SUBSTR(M.YEAR,-4) AS INTEGER)
    ),

table2 as
    (
    SELECT CAST(SUBSTR(M.YEAR,-4) AS INTEGER) Year,
        COUNT(DISTINCT TRIM(MID)) Total_Movies
    FROM movie M
    GROUP BY CAST(SUBSTR(M.YEAR,-4) AS INTEGER)
    ORDER BY CAST(SUBSTR(M.YEAR,-4) AS INTEGER)
    )

SELECT t1.year movie_year,t1.Total_Movies,t2.year movie_year,t2.Total_Movies
FROM table2 t2,table1 t1
WHERE t2.year BETWEEN t1.decade_start AND t1.decade_end
ORDER BY t1.year


            """
grader_7b(query7b)
# if you see the below results the first movie year is less than 2nd movie year and
# 2nd movie year is less or equal to the first movie year+9


# using the above query, you can write the answer to the given question
```

```
   movie_year  Total_Movies  movie_year  Total_Movies
0        1931             1        1931             1
1        1931             1        1936             3
2        1931             1        1939             2
3        1936             3        1936             3
4        1936             3        1939             2
5        1936             3        1941             1
6        1936             3        1943             1
7        1939             2        1939             2
8        1939             2        1941             1
9        1939             2        1943             1
CPU times: total: 15.6 ms
Wall time: 22.2 ms
```

In [111]:

```
%%time
def grader_7(q7):
    q7_results  = pd.read_sql_query(q7,conn)
    print(q7_results.head(10))
    assert (q7_results.shape == (1, 2))

query7 = """
WITH
table1 AS
    (
    SELECT DISTINCT
        CAST(SUBSTR(m.year,-4) AS INTEGER) year,
        CAST(SUBSTR(m.year,-4) AS INTEGER) decade_start,
        CAST(SUBSTR(m.year,-4) AS INTEGER) + 9 decade_end,
        COUNT(DISTINCT TRIM(MID)) Total_Movies
    FROM movie m
    GROUP BY CAST(SUBSTR(M.year,-4) AS INTEGER)
    ORDER BY CAST(SUBSTR(M.year,-4) AS INTEGER)
    ),

table2 as
    (
    SELECT CAST(SUBSTR(M.year,-4) AS INTEGER) year,
        COUNT(DISTINCT TRIM(MID)) Total_Movies
    FROM movie m
    GROUP BY CAST(SUBSTR(M.year,-4) AS INTEGER)
    ORDER BY CAST(SUBSTR(M.year,-4) AS INTEGER)
    )


SELECT f.movie_year, max(total)
FROM
    (
    SELECT t1.year movie_year, SUM(t2.total_movies) total
    FROM table2 t2,table1 t1
    WHERE t2.year BETWEEN t1.DECADE_START AND t1.DECADE_END
    GROUP BY t1.year
    ) f







"""
grader_7(query7)
# if you check the output we are printinng all the year in that decade, its fine you can print 2008 or 2008-2017
```

```
   movie_year  max(total)
0        2008        1203
CPU times: total: 15.6 ms
Wall time: 24.3 ms
```

Q8 --- Find all the actors that made more movies with Yash Chopra than any other director.

In [112]:

```
%%time
def grader_8a(q8a):
    q8a_results  = pd.read_sql_query(q8a,conn)
    print(q8a_results.head(10))
    print(q8a_results.shape)
    assert (q8a_results.shape == (73408, 3))

query8a = """

SELECT MC.pid actors, MD.pid directors,
COUNT(DISTINCT MD.MID) movie_count
FROM m_cast MC, m_director MD
WHERE MC.mid = MD.mid
GROUP BY directors, actors
ORDER BY actors, directors




"""
grader_8a(query8a)

# using the above query, you can write the answer to the given question
# *** Write a query that will results in number of movies actor-director worked together ***
```

```
       actors  directors  movie_count
0   nm0000002  nm0496746            1
1   nm0000027  nm0000180            1
2   nm0000039  nm0896533            1
3   nm0000042  nm0896533            1
4   nm0000047  nm0004292            1
5   nm0000073  nm0485943            1
6   nm0000076  nm0000229            1
7   nm0000092  nm0178997            1
8   nm0000093  nm0000269            1
9   nm0000096  nm0113819            1
(73408, 3)
CPU times: total: 703 ms
Wall time: 729 ms
```

In [113]:

```
%%time

def grader_8(q8):
    q8_results  = pd.read_sql_query(q8,conn)
    print(q8_results.head(10))
    print(q8_results.shape)
    assert (q8_results.shape == (245, 2))

query8 = """
WITH

yash_movie AS
(
SELECT DISTINCT pactor.pid AS actor_pid, COUNT(*) AS count
FROM  m_cast mc
JOIN  movie m
    ON m.mid = mc.mid
JOIN m_director md
    ON md.mid = mc.mid
JOIN  person  pactor
    ON pactor.pid = Trim(mc.pid)
JOIN  person pdirector
    ON pdirector.pid = md.pid
WHERE  pdirector.name LIKE '%Yash%Chopra%'
GROUP  BY actor_pid
),

others_movie AS
(
SELECT actor_pid as actor_pid, max(count) AS COUNT
FROM
    (
    SELECT  pactor.pid AS Actor_pid, pdirector.name director, Count(*) AS COUNT
    FROM  m_cast mc
    JOIN  movie m
      ON m.mid = mc.mid
    JOIN  m_director  md
      ON md.mid = mc.mid
    JOIN  person  pactor
      ON pactor.pid = Trim(mc.pid)
    JOIN  person pdirector
      ON pdirector.pid = md.pid
    WHERE pdirector.name NOT LIKE '%Yash%Chopra%'
    GROUP  BY actor_pid, director
     )
GROUP  BY actor_pid
)


SELECT P.name AS ACTOR, count
FROM
    (
    SELECT y.actor_pid,y.count
    FROM yash_movie y
    LEFT JOIN others_movie o
    ON y.actor_pid=o.actor_pid
    WHERE  y.count >= o.count OR o.actor_pid IS NULL
    GROUP BY y.actor_pid
    ORDER BY y.count DESC
    ) S, person P
    WHERE P.pid=S.actor_pid




"""
grader_8(query8)
```

```
               ACTOR  count
0      Sharib Hashmi      1
1    Kulbir Badesron      1
2        Gurdas Maan      1
3    Parikshat Sahni      3
4      Claire Ashton      1
5     Waheeda Rehman      5
6           Taj Gill      1
7         Kumud Pant      1
8   Gerald Tomkinson      1
9   Dev K. Kantawall      1
(245, 2)
CPU times: total: 1.92 s
Wall time: 1.96 s
```

Q9 --- 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 [118]:

```
%%time
def grader_9a(q9a):
    q9a_results  = pd.read_sql_query(q9a,conn)
    print(q9a_results.head(10))
    print(q9a_results.shape)
    assert (q9a_results.shape == (2382, 1))

query9a = """




SELECT DISTINCT MC.pid AS S1_pid
FROM m_cast MC
JOIN
    (
    SELECT pid FROM person
    ) P
    ON P.pid=TRIM(MC.pid)
JOIN
    (
    SELECT MC.mid mid, p.pid ppid
    FROM m_cast MC
    JOIN
        (
        SELECT pid,name FROM person
        ) P
        ON P.pid=TRIM(MC.pid)
    WHERE P.name LIKE '%Shah%rukh%Khan%'
    ) S
    ON mc.mid=s.mid AND p.pid <> s.ppid




"""
grader_9a(query9a)
# using the above query, you can write the answer to the given question

# selecting actors who acted with srk (S1)
# selecting all movies where S1 actors acted, this forms S2 movies list
# selecting all actors who acted in S2 movies, this gives us S2 actors along with S1 actors
# removing S1 actors from the combined list of S1 & S2 actors, so that we get only S2 actors
```

```
       S1_pid
0   nm0004418
1   nm1995953
2   nm2778261
3   nm0631373
4   nm0241935
5   nm0792116
6   nm1300111
7   nm0196375
8   nm1464837
9   nm2868019
(2382, 1)
CPU times: total: 2.53 s
Wall time: 2.6 s
```

In [119]:

```
%%time
def grader_9(q9):
    q9_results  = pd.read_sql_query(q9,conn)
    print(q9_results.head(10))
    print(q9_results.shape)
    assert (q9_results.shape == (25698, 1))

query9 = """
WITH

shahruk_0 AS
    (
    SELECT DISTINCT MC.mid mid, p.pid pid
    FROM m_cast MC
    JOIN
        (
        SELECT pid,name FROM person
        ) P
        ON P.PID=TRIM(MC.PID)
        WHERE P.NAME LIKE '%Shah%rukh%Khan%'
    ),

S1 AS
    (
    SELECT DISTINCT MC.pid AS S1_pid, MC.mid AS mid
    FROM m_cast MC
    JOIN shahruk_0 s0
    ON mc.mid=s0.mid AND mc.pid <> s0.pid
    ),

s1_mov as
    (
    SELECT DISTINCT mc.mid, pid
    FROM m_cast MC
    WHERE mc.pid IN (SELECT S1_pid FROM s1)
    )

SELECT distinct pid
FROM m_cast mc
where mc.mid in (SELECT mid FROM s1_mov) AND mc.pid NOT IN (SELECT S1_pid FROM s1) AND mc.pid NOT IN (SELECT pid FROM shahruk_0)

"""



grader_9(query9)
```

```
          PID
0   nm2539953
1   nm0922035
2   nm0324658
3   nm0943079
4   nm0000218
5   nm0001394
6   nm0929654
7   nm3116102
8   nm3248891
9   nm2418809
(25698, 1)
CPU times: total: 688 ms
Wall time: 684 ms
```