SQL Assignment

In [53]:
```
import pandas as pd
import sqlite3

from IPython.display import display, HTML
```

In [54]:
```
# Note that this is not the same db we have used in course videos, please download from
# https://drive.google.com/file/d/1O-1-L1DdNxEK6O6nG2jS31MbrMh-OnXM/view?usp=sharing
```

In [55]:
```
conn = sqlite3.connect("Db-IMDB-Assignment.db")
```

Overview of all tables

In [56]:
```
tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE type='t
tables = tables["Table_Name"].values.tolist()
```

In [57]:
```
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
```

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 [58]:
```
'%%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 Person.Name as Director_name,Movie.title as Movie_name,CAST(SUBSTR(TRIM(M
        FROM Person
        JOIN M_Director mc ON Person.PID = mc.PID
        JOIN Movie ON mc.MID = Movie.MID
        JOIN M_Genre mg ON Movie.MID = mg.MID
        JOIN Genre g ON mg.GID = g.GID
        WHERE g.Name LIKE '%Comedy%'
        AND
        (CAST(SUBSTR(TRIM(Movie.year),-4) AS INTEGER)%4 = 0
        AND
        CAST(SUBSTR(TRIM(Movie.year),-4) AS INTEGER)%100 <> 0
        OR
        CAST(SUBSTR(TRIM(Movie.year),-4) AS INTEGER)%400 = 0)
        '''
grader_1(query1)
```

```
       Director_name                        Movie_name  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
```

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

In [59]:
```
%%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 Name FROM Person WHERE TRIM(PID) IN (
             SELECT TRIM(PID) FROM M_Cast WHERE TRIM(MID) = (
             SELECT TRIM(MID) FROM Movie M WHERE TRIM(M.title) = 'Anand' AND CAST(SUBSTR
            """)
grader_2(query2)
```

```
                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
Wall time: 40 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 [60]:

```
%%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
inner join
(
    select trim(mc.PID) PD, mc.MID 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
inner join
(
    select trim(mc.PID) PD, mc.MID 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
Wall time: 381 ms
```

In [9]:
```
%%time
def grader_3(q3):
    q3_results  = pd.read_sql_query(q3,conn)
    print(q3_results.head(10))
    assert (q3_results.shape == (300,1))

query3 = ("""
          SELECT DISTINCT Name from PERSON P WHERE TRIM(PID) IN (
          SELECT DISTINCT TRIM(PID) FROM M_Cast WHERE (MID) IN (
          SELECT (MID) FROM Movie M WHERE CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)<1970)
          INTERSECT
          SELECT DISTINCT TRIM(PID) FROM M_Cast WHERE (MID) IN (
          SELECT (MID) FROM Movie m WHERE CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)>1990))

""")
grader_3(query3)
```

```
<bound method NDFrame.head of                   Name
0         Rishi Kapoor
1     Amitabh Bachchan
2               Asrani
3         Zohra Sehgal
4      Parikshat Sahni
..                 ...
295             Poonam
296      Jamila Massey
297        K.R. Vijaya
298              Sethi
299       Suryakantham

[300 rows x 1 columns]>
Wall time: 176 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 [61]:
```
%%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))

## *** Write a query, which will return all the directors(id's) along with the number of
query_4a ="""
          SELECT MD.PID as Director_ID ,Count(MD.PID) as Movie_Count FROM M_Director MD
          JOIN Movie M ON MD.MID = M.MID
          GROUP BY MD.PID
          ORDER BY Movie_count DESC
"""
print(grader_4a(query_4a))

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

```
  Director_ID  Movie_Count
0   nm0223522           39
1   nm0080315           35
2   nm0698184           30
3   nm0890060           30
4   nm0080333           29
5   nm0611531           27
6   nm0007181           21
7   nm0154113           19
8   nm0759662           19
9   nm0007131           18
True
Wall time: 25 ms
```

In [62]:
```
%%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 DISTINCT P.name Director_Name,Count(*) Movie_Count FROM Person P
         JOIN M_Director MD ON P.PID = MD.PID
         GROUP BY MD.PID HAVING COUNT(*) >= 10
         ORDER BY Movie_Count DESC

"""
grader_4(query4)
```

```
          Director_Name  Movie_Count
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
Wall time: 42 ms
```

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

In [63]:
```
%%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))
'*** Write your query that will get movie id, and number of people for each gender ***'
query_5aa ="""
            SELECT m.MID,p.Gender,Count(*) FROM Person p
            JOIN M_Cast mc ON p.PID = TRIM(mc.PID)
            JOIN Movie m ON m.MID = TRIM(mc.MID)
            GROUP BY m.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))
'*** Write your query that will have at least one male actor try to use query that you h
query_5ab ="""
    SELECT m.MID,p.Gender,Count(*) FROM Person p
    JOIN M_Cast mc ON p.PID = TRIM(mc.PID)
    JOIN Movie m ON m.MID = TRIM(mc.MID)
    GROUP BY m.MID,p.Gender
    HAVING p.Gender = 'Male'

"""

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 Gender  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
Wall time: 549 ms
```

In [64]:
```
%%time
def grader_5a(q5a):
    q5a_results  = pd.read_sql_query(q5a,conn)
    print(q5a_results.head(10))
    assert (q5a_results.shape == (4,2))
# *** Write your query for the question 5a ***
query5a = """
        SELECT CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) as year ,Count(DISTINCT m.Title)
        Movie m JOIN M_Cast mc ON mc.MID = m.MID
        JOIN Person p ON p.PID = TRIM(mc.PID)
        WHERE m.Title
        NOT IN

        (SELECT m.Title FROM
        Movie m JOIN M_Cast mc ON m.MID = mc.MID
        JOIN Person p ON TRIM(mc.PID) = p.PID
        WHERE p.Gender = 'Male')

        GROUP BY year
"""
grader_5a(query5a)
```

```
   year  Female_Cast_Only_Movies
0  1939                        1
1  1999                        1
2  2000                        1
3  2018                        1
Wall time: 603 ms
```

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 [65]:
```
%%time
def grader_5b(q5b):
    q5b_results  = pd.read_sql_query(q5b,conn)
    print(q5b_results.head(10))
    assert (q5b_results.shape == (4,3))

    #*** Write your query for the question 5b***
query5b = """

    SELECT Female_movies.year, (CAST(Female_movies.Female_Cast_Only_Movies AS FLOAT)*100
    TM.Total_movies FROM

    (SELECT year,Count(m.Title) as Total_movies
    From Movie m
    GROUP BY CAST(SUBSTR(m.year,-4) AS Integer)) as TM

    JOIN

    (SELECT CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) as year ,Count(DISTINCT m.Title) as
    Movie m JOIN M_Cast mc ON mc.MID = m.MID
    JOIN Person p ON p.PID = TRIM(mc.PID)
    WHERE m.Title
    NOT IN

    (SELECT m.Title FROM
    Movie m JOIN M_Cast mc ON m.MID = mc.MID
    JOIN Person p ON TRIM(mc.PID) = p.PID
    WHERE p.Gender = 'Male')

    GROUP BY year) as Female_movies

    ON Female_movies.year = TM.year


"""
grader_5b(query5b)
```

```
   year  Percentage_Female_Only_Movie  Total_movies
0  1939                     50.000000             2
1  1999                      1.515152            66
2  2000                      1.562500            64
3  2018                      0.961538           104
Wall time: 656 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 [66]:
```
%%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 = """
         SELECT m.title Movie,Count(Distinct(mc.PID)) count FROM MOVIE m
         JOIN M_Cast mc ON mc.MID=m.MID
         GROUP BY m.MID ORDER BY count DESC

"""
grader_6(query6)
```

```
                       Movie  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
Wall time: 310 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 [67]:
```
%%time
def grader_7a(q7a):
    q7a_results  = pd.read_sql_query(q7a,conn)
    print(q7a_results.head(10))
    assert (q7a_results.shape == (78, 2))

# *** Write a query that computes number of movies in each year ***

query7a = """
        SELECT CAST(SUBSTR(m.year,-4) AS Integer) as Movie_year , Count(m.Title) as Tota
        Movie m
        GROUP BY Movie_year


"""
grader_7a(query7a)

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

```
   Movie_year  Total_Movies
0        1931             1
1        1936             3
2        1939             2
3        1941             1
4        1943             1
5        1946             2
6        1947             2
7        1948             3
8        1949             3
9        1950             2
Wall time: 14 ms
```

In [68]:

```
%%time
def grader_7b(q7b):
    q7b_results  = pd.read_sql_query(q7b,conn)
    print(q7b_results.head(10))
    assert (q7b_results.shape == (713, 4))
''' ***
    Write a query that will do joining of the above table(7a) with itself
    such that you will join with only rows if the second tables year is <= current_year+
    *** '''
query7b = """
        SELECT * FROM
        (SELECT CAST(SUBSTR(m.year,-4) AS Integer) as Movie_year , Count(m.Title) as Tot
        Movie m
        GROUP BY Movie_year) as table1

        JOIN

        (SELECT CAST(SUBSTR(m.year,-4) AS Integer) as Movie_year , Count(m.Title) as Tot
        Movie m
        GROUP BY Movie_year) as table2

        WHERE table2.Movie_year <= table1.Movie_year + 9 AND table2.Movie_year >= table1


          """
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
Wall time: 22 ms
```

In [69]:
```
%%time
def grader_7(q7):
    q7_results  = pd.read_sql_query(q7,conn)
    print(q7_results.head(10))
    assert (q7_results.shape == (1, 2))
'''
*** Write a query that will return the decade that has maximum number of movies ***
'''
query7 = """
        SELECT MAX(sum_of_movies) as Decade_Movie_count, Movie_year as Decade FROM

        (SELECT table1.Movie_year,SUM(table2.Total_Movies) as sum_of_movies FROM

        (SELECT CAST(SUBSTR(m.year,-4) AS Integer) as Movie_year , Count(m.Title) as Tot
        Movie m
        GROUP BY Movie_year) as table1

        JOIN

        (SELECT CAST(SUBSTR(m.year,-4) AS Integer) as Movie_year , Count(m.Title) as Tot
        Movie m
        GROUP BY Movie_year) as table2

        WHERE table2.Movie_year <= table1.Movie_year + 9 AND table2.Movie_year >= table1

        GROUP BY table1.Movie_year)



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

```
   Decade_Movie_count  Decade
0                1203    2008
Wall time: 20 ms
```

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

In [70]:

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

'''
*** Write a query that will results in number of movies actor-director worked together *
'''
query8a = """
            SELECT DISTINCT mc.PID as Actor, md.PID as director,Count(*) as Movies from
            JOIN M_Director md ON m.MID = md.MID
            JOIN M_Cast mc ON mc.MID = m.MID
            GROUP BY mc.ID
            ORDER BY Actor

"""
grader_8a(query8a)

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

```
        Actor   director  Movies
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  nm0149446       1
Wall time: 694 ms
```

In [71]:
```
%%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))
'''
*** Write a query that answers the 8th question ***
'''
query8 = """

        SELECT Name , Movies_with_yc FROM
        (
            SELECT p.Name,p.PID,Count(m.MID) as Movies_with_yc FROM Movie m
            JOIN M_Cast mc ON mc.MID = m.MID
            JOIN M_Director md ON md.MID = m.MID
            JOIN Person p ON p.PID = TRIM(mc.PID)
            JOIN Person p1 ON p1.PID = md.PID
            WHERE TRIM(p1.Name) = 'Yash Chopra'
            GROUP BY p.PID
        )with_yc

        Left OUTER JOIN

        (
            SELECT PID, MAX(Movies_without_yc) as Max_movies_without_yc FROM
            (
                SELECT p.PID,Count(m.MID) as Movies_without_yc FROM Movie m
                JOIN M_Cast mc ON mc.MID = m.MID
                JOIN M_Director md ON md.MID = m.MID
                JOIN Person p ON p.PID = TRIM(mc.PID)
                JOIN Person p1 ON p1.PID = md.PID
                WHERE TRIM(p1.Name) != 'Yash Chopra'
                GROUP BY p.PID,p1.Name
            )GROUP BY PID
        )without_yc

        ON with_yc.PID = without_yc.PID

        WHERE with_yc.Movies_with_yc >= without_yc.Max_movies_without_yc
        OR without_yc.PID IS NULL
        ORDER BY with_yc.Movies_with_yc DESC



"""
grader_8(query8)
```

```
                Name  Movies_with_yc
0        Jagdish Raj              11
1   Manmohan Krishna              10
2           Iftekhar               9
3      Shashi Kapoor               7
4      Rakhee Gulzar               5
5     Waheeda Rehman               5
6           Ravikant               4
7     Achala Sachdev               4
8        Neetu Singh               4
9      Leela Chitnis               3
(245, 2)
Wall time: 845 ms
```

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 [72]:
```
%%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))
'''
 *** Write a query that answers the 9th question ***
'''
query9a = """


        SELECT DISTINCT p2.PID as s1_PID FROM Person p2
           JOIN M_Cast mc2 ON TRIM(mc2.PID) = p2.PID
           JOIN Movie m2 ON m2.MID = TRIM(mc2.MID)
           Where TRIM(p2.Name) != 'Shah Rukh Khan'
           AND m2.MID IN
           (
                SELECT DISTINCT m1.MID as Movies_srk FROM Person p1
                JOIN M_Cast mc1 ON TRIM(mc1.PID) = p1.PID
                JOIN Movie m1 ON m1.MID = TRIM(mc1.MID)
                Where TRIM(p1.Name) = 'Shah Rukh Khan'  )

"""
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 act
# removing S1 actors from the combined list of S1 & S2 actors, so that we get only S2 ac
```

```
      s1_PID
0  nm0004418
1  nm1995953
2  nm2778261
3  nm0631373
4  nm0241935
5  nm0792116
6  nm1300111
7  nm0196375
8  nm1464837
9  nm2868019
(2382, 1)
Wall time: 441 ms
```

In [78]:
```
%%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))
'''
*** Write a query that answers the 9th question ***
'''
query9 = """

            SELECT p5.s1_s2_actors as Actor_name FROM
            (
                SELECT DISTINCT (p4.PID) as s1_s2_PID,TRIM(p4.Name) as s1_s2_actors FROM
                JOIN M_Cast mc4 ON TRIM(mc4.PID) = p4.PID
                JOIN Movie m4 ON m4.MID = TRIM(mc4.MID)
                Where TRIM(p4.Name) != 'Shah Rukh Khan'
                AND m4.MID IN
                (
                    SELECT DISTINCT m3.MID as srk2_movies FROM Person p3
                    JOIN M_Cast mc3 ON TRIM(mc3.PID) = p3.PID
                    JOIN Movie m3 ON m3.MID = TRIM(mc3.MID)
                    Where p3.PID IN
                    (
                        SELECT DISTINCT (p2.PID) as srk1_actors FROM Person p2
                        JOIN M_Cast mc2 ON TRIM(mc2.PID) = p2.PID
                        JOIN Movie m2 ON m2.MID = TRIM(mc2.MID)
                        Where TRIM(p2.Name) != 'Shah Rukh Khan'
                        AND m2.MID IN
                        (
                                SELECT m1.MID as Movies_srk FROM Person p1
                                JOIN M_Cast mc1 ON TRIM(mc1.PID) = p1.PID
                                JOIN Movie m1 ON m1.MID = TRIM(mc1.MID)
                                Where TRIM(p1.Name) = 'Shah Rukh Khan'  ))))p5

                                WHERE p5.s1_s2_PID NOT IN

                                (
                                SELECT DISTINCT (p6.PID) as S1_PID FROM Person p6
                                JOIN M_Cast mc6 ON TRIM(mc6.PID) = p6.PID
                                JOIN Movie m6 ON m6.MID = TRIM(mc6.MID)
                                Where TRIM(p6.Name) != 'Shah Rukh Khan'
                                AND m6.MID IN
                                (
                                SELECT DISTINCT m7.MID as Movies_srk FROM Person p7
                                JOIN M_Cast mc7 ON TRIM(mc7.PID) = p7.PID
                                JOIN Movie m7 ON m7.MID = TRIM(mc7.MID)
                                Where TRIM(p7.Name) = 'Shah Rukh Khan'))
"""
grader_9(query9)
```

```
            Actor_name
0       Alicia Vikander
1          Dominic West
2        Walton Goggins
3             Daniel Wu
4  Kristin Scott Thomas
5          Derek Jacobi
6    Alexandre Willaume
7          Tamer Burjaq
8        Adrian Collins
9        Keenan Arrison
(25698, 1)
Wall time: 1.54 s
```

In [ ]:

In [ ]: