SQL Assignment

In [17]:
```
import pandas as pd
import sqlite3
from IPython.display import display, HTML
```

In [18]:
```
# 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 [19]:
```
conn = sqlite3.connect("Db-IMDB-Assignment.db")
```

Overview of all tables

In [20]:
```
tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE type='table'",conn)
tables = tables["Table_Name"].values.tolist()
```

In [21]:
```
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")
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	Name	TEXT	0	None	0
2	2	GID	INTEGER	0	None	0

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


Schema of Language
```

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	Name	TEXT	0	None	0
2	2	LAID	INTEGER	0	None	0

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 [22]:
```
%%time
def grader_1(q1):
    q1_results  = pd.read_sql_query(q1,conn)
    print(q1_results.head(10))
    print(q1_results.shape)
    assert (q1_results.shape == (232,3))

# 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



query1 = """
SELECT p.Name as Director_Name,b.title as Title,b.year as Year
FROM (((M_Genre mg INNER JOIN (SELECT GID FROM Genre WHERE Name LIKE '%Comedy%') g ON g.GID = mg.GID) com INNER JOIN Movie m ON m
WHERE ((CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)  % 4 = 0 and CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)  % 100 != 0) or (CAST(SUBS

"""
grader_1(query1)
```

```
          Director_Name                         Title  Year
0          Prakash Mehra                    Hera Pheri  1976
1             Raj Kanwar                       Deewana  1992
2           Priyadarshan                    Hera Pheri  2000
3           Naresh Kumar                 Gora Aur Kala  1972
4          Eeshwar Nivas  My Name Is Anthony Gonsalves  2008
5           Anees Bazmee                Singh Is Kinng  2008
6   Deepak S. Shivdasani           Mr. White Mr. Black  2008
7          Anurag Kashyap            Gangs of Wasseypur  2012
8           Rajat Kapoor                        Mithya  2008
9          Ravindra Dave                  Dulha Dulhan  1964
(232, 3)
Wall time: 65.3 ms
```

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

In [23]:
```
%%time
def grader_2(q2):
    q2_results  = pd.read_sql_query(q2,conn)
    print(q2_results.head(10))
    print(q2_results.shape)
    assert (q2_results.shape == (17,1))

query2 = """
Select p.Name
From Person p,(Select mc.PID
               from (select MID
                     from Movie m
                     where title = 'Anand' and 1971 = CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)
                     ) A, M_Cast mc
                    where  mc.MID = A.MID) p_pid
Where Trim(p.PID) = Trim(p_pid.PID)"""


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
(17, 1)
Wall time: 281 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 [24]:
```
%%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: 233 ms
```

In [25]:
```
%%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 p.Name
FROM
(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) p1,Person p
WHERE p1.PID IN
(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) AND p.PID = p1.PID
"""
grader_3(query3)
```

```
              Name
0   Waheeda Rehman
1    Johnny Walker
2          Mehmood
3            Ratna
4   Rajendra Kumar
5         Iftekhar
6        Raj Mehra
7     Lalita Pawar
8   Achala Sachdev
9       Sunil Dutt
Wall time: 283 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 [26]:
```
%%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 trim(PID) as Director_ID,count(MID) as Movie_Count
From M_Director
group by trim(PID)
order by Movie_Count desc
"""
print(grader_4a(query_4a))
```

```
  Director_ID  Movie_Count
0   nm0223522           39
1   nm0080315           35
2   nm0890060           30
3   nm0698184           30
4   nm0080333           29
5   nm0611531           27
6   nm0007181           21
7   nm0759662           19
8   nm0154113           19
9   nm0007131           18
True
Wall time: 6.83 ms
```

In [27]:
```
%%time
def grader_4(q4):
    q4_results  = pd.read_sql_query(q4,conn)
    print(q4_results.head(10))
    print(q4_results.shape)
    assert (q4_results.shape == (58,2))

query4 = """
select p.Name as Director_Name,Movie_Count
from (Select trim(PID) as ID,count(MID) as Movie_Count
      From M_Director
      group by trim(PID)
      Having count(MID) > 9
      order by Movie_Count desc) d_pid , Person p
Where p.PID = d_pid.ID
"""
grader_4(query4)
```

```
           Director_Name  Movie_Count
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
(58, 2)
Wall time: 54.2 ms
```

Type *Markdown* and LaTeX: α^2

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

In [28]:
```
%%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(*)
From M_Cast mc, Person p
Where trim(mc.PID) = p.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,Count(*)From M_Cast mc, Person p
Where trim(mc.PID) = p.PID
group by mc.MID,p.Gender
HAVING (p.Gender= 'Male' AND Count(*)!=0)
"""

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: 281 ms
```

In [29]:
```
%%time
def grader_5a(q5a):
    q5a_results  = pd.read_sql_query(q5a,conn)
    print(q5a_results.head(10))
    assert (q5a_results.shape == (4,2))

query5a = """
SELECT CAST(SUBSTR(m.year,-4) AS Integer) as YEAR,count(m.MID) as Female_Cast_Only_Movies
FROM
(Select mc.MID,p.Gender,Count(*)
From M_Cast mc, Person p
Where trim(mc.PID) = p.PID
group by mc.MID,p.Gender) as m_all,Movie m
WHERE m_all.MID not in (Select mc.MID
                            From M_Cast mc, Person p
                            Where trim(mc.PID) = p.PID
                            group by mc.MID,p.Gender
                            HAVING (p.Gender= 'Male' AND Count(*)!=0))
and m_all.MID = m.MID
GROUP BY CAST(SUBSTR(m.year,-4) AS Integer)
"""
grader_5a(query5a)
```

```
   YEAR  Female_Cast_Only_Movies
0  1939                        1
1  1999                        1
2  2000                        1
3  2018                        1
Wall time: 287 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 [ ]:

In [30]:
```
%%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 = """
SELECT m.year,(fm.Female_Cast_Only_Movies+0.0)/count(MID) as Percentage_Female_Only_Movie,count(MID) as Total_Movies
FROM (SELECT CAST(SUBSTR(m.year,-4) AS Integer) as YEAR,count(m.MID) as Female_Cast_Only_Movies
        FROM
        (Select mc.MID,p.Gender,Count(*)
        From M_Cast mc, Person p
        Where trim(mc.PID) = p.PID
        group by mc.MID,p.Gender) as m_all,Movie m
        WHERE m_all.MID not in (Select mc.MID
                                    From M_Cast mc, Person p
                                    Where trim(mc.PID) = p.PID
                                    group by mc.MID,p.Gender
                                    HAVING (p.Gender= 'Male' AND Count(*)!=0))
        and m_all.MID = m.MID
        GROUP BY CAST(SUBSTR(m.year,-4) AS Integer) ) fm
INNER JOIN Movie m ON fm.YEAR = CAST(SUBSTR(m.year,-4) AS Integer)
GROUP BY CAST(SUBSTR(m.year,-4) AS Integer)
"""
grader_5b(query5b)
```

```
   year  Percentage_Female_Only_Movie  Total_Movies
0  1939                      0.500000             2
1  1999                      0.015152            66
2  2000                      0.015625            64
3  2018                      0.009615           104
Wall time: 288 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 [31]:
```
%%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,count(mc.PID) as count
From M_Cast mc INNER JOIN Movie m ON mc.MID = m.MID
GROUP BY mc.MID
ORDER BY count DESC
"""
grader_6(query6)
```

```
                       title  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: 88.7 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 [32]:
```
%%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(TRIM(m.year),-4) AS INTEGER) as Movie_Year,Count(*) as Total_Movies
From Movie m
Group by CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)
"""
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: 6 ms
```

In [33]:
```
%%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+9 and more than or equal current_year
    # ***
query7b ="""
Select ym1.Movie_Year,ym1.Total_Movies,ym2.Movie_Year,ym2.Total_Movies
From (Select CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) as Movie_Year,Count(*) as Total_Movies
From Movie m
Group by CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)) ym1,
(Select CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) as Movie_Year,Count(*) as Total_Movies
From Movie m
Group by CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)) ym2
Where ym1.Movie_Year <= ym2.Movie_Year and ym2.Movie_Year <= ym1.Movie_Year + 9
"""
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: 10 ms
```

In [ ]:

In [34]:
```
%%time
def grader_7(q7):
    q7_results  = pd.read_sql_query(q7,conn)
    print(q7_results.head(10))
    assert (q7_results.shape == (1, 2))

# CONCAT(Cast(ym1.Movie_Year as varchar(10)),'-',Cast(max(ym2.Movie_Year) as varchar(10)))
query7 = """
SELECT S_YEAR,max(count)
FROM   ( Select ym1_Movie_Year as S_YEAR ,max(ym2_Movie_Year) as E_YEAR , SUM(ym2_Total_Movies) as count
        From(Select ym1.Movie_Year as ym1_Movie_Year,ym1.Total_Movies as ym1_Total_Movies,ym2.Movie_Year as ym2_Movie_Year,ym2.To
                From (Select CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) as Movie_Year,Count(*) as Total_Movies
                From Movie m
                Group by CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)) ym1,
                (Select CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) as Movie_Year,Count(*) as Total_Movies
                From Movie m
                Group by CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER)) ym2
                Where ym1.Movie_Year <= ym2.Movie_Year and ym2.Movie_Year <= ym1.Movie_Year + 9 )
        GROUP BY ym1_Movie_Year)
"""
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
```

```
   S_YEAR  max(count)
0    2008        1203
Wall time: 8.16 ms
```

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

In [35]:
```
%%time
def grader_8a(q8a):
    q8a_results  = pd.read_sql_query(q8a,conn)
    print(q8a_results.head(10))
    assert (q8a_results.shape == (73408, 3))

query8a = """
SELECT mc.PID as Actor,md.PID as Director,count(*)
FROM M_Cast mc, M_Director md
WHERE mc.MID = md.MID
GROUP BY mc.PID,md.PID
"""
grader_8a(query8a)

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

```
       Actor   Director  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
Wall time: 229 ms
```

In [36]:

```
%%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 = """
SELECT p.Name,yash.count
From Person p INNER JOIN
(SELECT with_yash.Actor,
(CASE
WHEN with_out_yash.Actor IS NULL THEN with_yash.count
WHEN (with_yash.count - with_out_yash.count) >= 0 THEN with_yash.count
END) as Count
FROM
(SELECT mc.PID as Actor,count(*) as count
FROM M_Cast mc, M_Director md
WHERE mc.MID = md.MID AND md.PID IN (SELECT PID
                                    FROM Person
                                    WHERE trim(Name) LIKE '%Yash %Chopra%')
GROUP BY mc.PID) with_yash

Left OUTER JOIN

(SELECT Actor,max(count) as count
FROM (SELECT mc.PID as Actor,md.PID,count(*) as count
FROM M_Cast mc, M_Director md
WHERE mc.MID = md.MID AND md.PID != (SELECT PID
                                    FROM Person
                                    WHERE trim(Name) = 'Yash Chopra')
GROUP BY mc.PID,md.PID)
GROUP BY Actor)with_out_yash ON with_yash.Actor = with_out_yash.Actor
ORDER BY count DESC) yash ON trim(p.PID) = trim(yash.Actor)
WHERE yash.count IS NOT NULL
ORDER BY yash.count DESC;
"""
grader_8(query8)
```

```
                Name  Count
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: 1.78 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 [37]:
```
%%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 INNER JOIN (SELECT MID
                            FROM M_Cast
                            WHERE trim(PID) = (SELECT trim(p.PID)
                                                FROM Person p
                                                WHERE trim(p.Name) LIKE '%Shah%rukh%Khan%')) m ON mc.MID = m.MID
WHERE trim(mc.PID) != (SELECT trim(p.PID) FROM Person p WHERE trim(p.Name) LIKE '%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 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)
Wall time: 655 ms
```

In [38]:
```
%%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 = """
SELECT p.Name as Actor_Name
FROM Person p
WHERE trim(p.PID) IN
(SELECT DISTINCT trim(mc.PID)
FROM M_Cast mc INNER JOIN (SELECT DISTINCT mc.MID
                           FROM M_Cast mc INNER JOIN (SELECT DISTINCT mc.PID
                                                    FROM M_Cast mc INNER JOIN (SELECT MID
                                                                                FROM M_Cast
                                                                                WHERE trim(PID) = (SELECT trim(p.PID)
                                                                                                FROM Person p
                                                                                                WHERE trim(p.Name) LIKE '%Shah%ru
                                                    WHERE trim(mc.PID) != (SELECT trim(p.PID) FROM Person p WHERE trim(p.Name) LI
WHERE mc.PID not in (SELECT DISTINCT mc.PID as S1_PID
                     FROM M_Cast mc INNER JOIN (SELECT MID
                                                    FROM M_Cast
                                                    WHERE trim(PID) = (SELECT trim(p.PID)
                                                                    FROM Person p
                                                                    WHERE trim(p.Name) LIKE '%Shah%rukh%Khan%')) m ON mc.MID = m.
"""
grader_9(query9)
```

```
               Actor_Name
0            Freida Pinto
1             Rohan Chand
2            Damian Young
3         Waris Ahluwalia
4   Caroline Christl Long
5           Rajeev Pahuja
6       Michelle Santiago
7         Alicia Vikander
8            Dominic West
9          Walton Goggins
(25698, 1)
Wall time: 1.01 s
```

In [ ]: