SQL Assignment

In [2]:
```
import pandas as pd
import sqlite3

from IPython.display import display, HTML
```

In [ ]:
```
# 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 [3]:
```
conn = sqlite3.connect("Db-IMDB-Assignment.db")
```

Overview of all tables

In [4]:
```
tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE type='
tables = tables["Table_Name"].values.tolist()
```

In [5]:
```
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

	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

Schema of Country

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

Schema of Location

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

Schema of M_Location

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	LID	REAL	0	None	0
3	3	ID	INTEGER	0	None	0

Schema of M_Country

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	CID	REAL	0	None	0
3	3	ID	INTEGER	0	None	0

Schema of M_Language

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	LAID	INTEGER	0	None	0
3	3	ID	INTEGER	0	None	0

Schema of M_Genre

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	GID	INTEGER	0	None	0
3	3	ID	INTEGER	0	None	0

Schema of Person

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	PID	TEXT	0	None	0
2	2	Name	TEXT	0	None	0
3	3	Gender	TEXT	0	None	0

Schema of M_Producer

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	PID	TEXT	0	None	0
3	3	ID	INTEGER	0	None	0

Schema of M_Director

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	PID	TEXT	0	None	0
3	3	ID	INTEGER	0	None	0

Schema of M_Cast

	cid	name	type	notnull	dflt_value	pk
0	0	index	INTEGER	0	None	0
1	1	MID	TEXT	0	None	0
2	2	PID	TEXT	0	None	0
3	3	ID	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 [124…
```
%%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(a.Name) AS DirectorName , Trim(c.title) as MovieName ,  CAST(SU
            from Person a
            Inner join M_Director b on a.PID = b.PID
            Inner join Movie c on b.MID = c.MID
            Inner join M_Genre d on c.MID = d.MID
            Inner join Genre e on e.GID = d.GID
            Where e.Name like '%Comedy%'
            and (
            IIF(CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) % 4 = 0,True,False) or
            IIF(CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) % 4 = 0 and CAST(SUBSTR(TRIM(c
            IIF(CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) % 4 = 0 and CAST(SUBSTR(TRIM(c
            )

            """
grader_1(query1)
```

```
      DirectorName                          MovieName  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
Wall time: 444 ms
```

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

In [126…
```
%%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 a.Name
            from  Person a
            Inner join M_cast b on a.PID = trim(b.PID)
            Inner join Movie c on c.MID = trim(b.MID)
            Where c.title = "Anand"
            and CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) = 1971

"""
grader_2(query2)
```

```
                Name
0      Rajesh Khanna
1   Amitabh Bachchan
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: 313 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 [127…

```
%%time

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

query_less_1970 =""" Select a.PID
                     from  Person a
                     Inner join M_cast b on a.PID = trim(b.PID)
                     Inner join Movie c on c.MID = trim(b.MID)
                     Where  CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) < 1970 """
query_more_1990 =""" Select a.PID
                     from  Person a
                     Inner join M_cast b on a.PID = trim(b.PID)
                     Inner join Movie c on c.MID = trim(b.MID)
                     Where CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) > 1990 """
print(grader_3a(query_less_1970, query_more_1990))

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

```
True
Wall time: 971 ms
```

In [128…

```
%%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 Trim(a.Name) as Name
             from  Person a
             Inner join M_cast b on a.PID = trim(b.PID)
             Inner join Movie c on c.MID = trim(b.MID)
             Where  CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) < 1970
             and a.PID in (
                 Select distinct a.PID
                 from  Person a
                 Inner join M_cast b on a.PID = trim(b.PID)
                 Inner join Movie c on c.MID = trim(b.MID)
                 Where  CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) > 1990
```

```
                            )

              """

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: 1.17 s
```

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 [129…

```
%%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 a.PID as ID , Count(*) as MoviesDirected
              from Person a
              Inner join M_director b on a.PID = b.PID
              Inner join Movie c on c.MID = b.MID
              Group by a.PID
            """
print(grader_4a(query_4a))

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

```
          ID  MoviesDirected
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
Wall time: 79.8 ms
```

In [131…

```
%%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 Name As Director_Name, Count(title) as Movie_Count
              from Person a
              Inner join M_director b on a.PID = b.PID
              Inner join Movie c on c.MID = b.MID
              Group by Name
              Having Count(Title) >=10
              Order by Count(Title) desc
              """
grader_4(query4)
```

```
          Director_Name  Movie_Count
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
Wall time: 78 ms
```

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

In [132…

```
%%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 c.MID , Gender, count(a.PID) as CountOfActors
               from Person a
               Inner join M_cast b on a.PID = Trim(b.PID)
               Inner join Movie c on b.MID = c.MID
               Group by c.MID , 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 c.MID , Gender, count(a.PID) as CountOfActors
               from Person a
               Inner join M_cast b on a.PID = Trim(b.PID)
               Inner join Movie c on b.MID = c.MID
               Where Gender = "Male"
               Group by c.MID , Gender
               Having count(a.PID) > 0"""
```

```
print(grader_5ab(query_5ab))


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

```
         MID  Gender  CountOfActors
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  CountOfActors
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: 952 ms
```

In [65]:
```
%%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(TRIM(c.year),-4) AS INTEGER) as Year ,  count(distinct
                      from Person a
                      Inner join M_cast b on a.PID = Trim(b.PID)
                      Inner join Movie c on b.MID = c.MID
                      Where c.MID not in (
                          Select c.MID
                          from Person a
                          Inner join M_cast b on a.PID = Trim(b.PID)
                          Inner join Movie c on b.MID = c.MID
                          Where Gender in ("Male" , "None")
                      )
                      Group by CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER)
                      """
grader_5a(query5a)
```

```
   Year  Female_only_Movies
0  1939                   1
1  1999                   1
2  2000                   1
3  2018                   1
Wall time: 787 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 [133…

```
%%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 a.Year , Cast(Female_only_movies as float)/cast(Total_movies as float) as Percen
from (
    Select CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER) as Year ,  count(Distinct c.MID) as
    from Person a
       Inner join M_cast b on a.PID = Trim(b.PID)
       Inner join Movie c on b.MID = c.MID
       Where c.MID not in (
           Select c.MID
           from Person a
           Inner join M_cast b on a.PID = Trim(b.PID)
           Inner join Movie c on b.MID = c.MID
           Where Gender in ("Male" , "None")
       )
    Group by CAST(SUBSTR(TRIM(c.year),-4) AS INTEGER)) a
Inner join (
    Select CAST(SUBSTR(TRIM(year),-4) AS INTEGER) as Year ,  count(MID) as Total_Movies
    From Movie
    Group by CAST(SUBSTR(TRIM(year),-4) AS INTEGER)) b
    on a.year = b.year



           """
grader_5b(query5b)
```

```
   Year  Percentage_Female_Only  Total_Movies
0  1939                0.500000             2
1  1999                0.015152            66
2  2000                0.015625            64
3  2018                0.009615           104
Wall time: 762 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 [134…

```
%%time
```

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

query6 = """ Select c.Title , count(Distinct a.PID) as CountofCast
             from  Person a
             Inner join M_cast b on a.PID = trim(b.PID)
             Inner join Movie c on c.MID = trim(b.MID)
             Group by c.MID
             Order by CountofCast desc"""
grader_6(query6)
```

```
                       title  CountofCast
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
(3473, 2)
Wall time: 522 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 [135…

```
%%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(year),-4) AS INTEGER) as Year ,  count(MID) as To
              From Movie
              Group by CAST(SUBSTR(TRIM(year),-4) AS INTEGER) """
grader_7a(query7a)

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

```
   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: 17 ms
```

In [26]:

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

query7b = """
    Select * from (
    Select CAST(SUBSTR(TRIM(year),-4) AS INTEGER) as Year1 ,  count(MID) as Total_Movie
    From Movie
    Group by CAST(SUBSTR(TRIM(year),-4) AS INTEGER)
    )a
    join
    (
    Select CAST(SUBSTR(TRIM(year),-4) AS INTEGER) as Year2 ,  count(MID) as Total_Movie
    From Movie
    Group by CAST(SUBSTR(TRIM(year),-4) AS INTEGER)
    )b
    on a.year1 <= b.year2 and a.year1+9 >= b.year2


          """
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
```

```
   Year1  Total_Movies1  Year2  Total_Movies2
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
(713, 4)
Wall time: 18 ms
```

In [137…

```
%%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 = """

Select Total as Total_Movies , year1 as Decade
from (
    Select year1  , sum(total_movies2) as Total, row_number() over (order by sum(total_
```

```
    from (
       Select CAST(SUBSTR(TRIM(year),-4) AS INTEGER) as Year1 ,  count(MID) as Total_Mo
       From Movie
       Group by CAST(SUBSTR(TRIM(year),-4) AS INTEGER)
       )a
       join
       (
       Select CAST(SUBSTR(TRIM(year),-4) AS INTEGER) as Year2 ,  count(MID) as Total_Mo
       Group by CAST(SUBSTR(TRIM(year),-4) AS INTEGER)
       )b
       on a.year1 <= b.year2 and a.year1+9 >= b.year2
       Group by year1
       ) where row = 1


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

```
   Total_Movies  Decade
0          1203    2008
Wall time: 17.9 ms
```

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

In [172…

```
%%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 b.PID as Actor , a.PID as Director , count(a.MID) as Movies
              from M_Director a
              Inner join M_Cast b on a.MID = Trim(b.MID)
              Group by Actor , Director

            """
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  nm0113819       1
Wall time: 341 ms
```

In [31]:

```
%%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 b.Name , a.Movies from
       (
       Select Actor , Director , Movies from
              (

              Select Trim(b.PID) as Actor , a.PID as Director , count(a.MID) as Movies
              from M_Director a
              Inner join M_Cast b on a.MID = Trim(b.MID)
              Group by Actor , Director
              )
              where (Actor , Movies) in
                  (
                  Select Actor , max(movies) from
                      (
                      Select Trim(b.PID) as Actor , a.PID as Director , count(a.MID) as
                      from M_Director a
                      Inner join M_Cast b on a.MID = Trim(b.MID)
                      Group by Actor ,Director
                      )
                  Group by Actor
                  )
                  and Trim(Director) in (Select distinct Trim(PID) from Person where Tr
       )a inner join
       (
       Select PID , Name from Person
       )b
        on a.Actor = b.PID
        Order by Movies desc




"""

grader_8(query8)
```

```
                Name  Movies
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: 1.23 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 [139…

```
%%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 Trim(PID) as PID from M_cast where Trim(MID) in (
                      Select Trim(b.MID) as MID
                      from Person a
                      Inner join M_cast b on a.PID = Trim(b.PID)
                      Where Name like "%Shah Rukh Khan%"
                    ) and Trim(PID) not in (
                      Select a.PID
                      from Person a
                      Inner join M_cast b on a.PID = Trim(b.PID)
                      Where 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 ac
# removing S1 actors from the combined list of S1 & S2 actors, so that we get only S2 a
```

```
        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: 656 ms
```

In [140…

```
%%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 Name from (
        Select distinct Trim(PID) as PID from M_cast where Trim(MID) in (

                    Select Distinct Trim(MID)
```

```
                    from M_cast
                    where Trim(PID)
                    in (
                        Select Distinct Trim(PID) as PID from M_cast
                        where Trim(MID)
                        in (
                              Select Trim(b.MID) as MID
                              from Person a
                              Inner join M_cast b on a.PID = Trim(b.PID)
                              Where Name like "%Shah Rukh Khan%"
                            ) and Trim(PID) not in (
                              Select a.PID
                              from Person a
                              Inner join M_cast b on a.PID = Trim(b.PID)
                              Where Name like "%Shah Rukh Khan%"
                            )
                        )
        ) and Trim(PID) not in (
         Select Distinct Trim(PID) as PID from M_cast
                        where Trim(MID)
                        in (
                              Select Trim(b.MID) as MID
                              from Person a
                              Inner join M_cast b on a.PID = Trim(b.PID)
                              Where Name like "%Shah Rukh Khan%"
                            ) and Trim(PID) not in (
                              Select a.PID
                              from Person a
                              Inner join M_cast b on a.PID = Trim(b.PID)
                              Where Name like "%Shah Rukh Khan%"
                            )
        ) and Trim(PID) not in(Select a.PID
                              from Person a
                              Inner join M_cast b on a.PID = Trim(b.PID)
                              Where Name like "%Shah Rukh Khan%")
                    ) a Inner join
                    Person b on a.PID = b.PID
"""
grader_9(query9)
```

```
                   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: 2.17 s
```