SQL Assignment

In [34]:

```
import pandas as pd
import sqlite3

from IPython.display import display, HTML
```

In [35]:

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

In [36]:

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

Overview of all tables

In [37]:

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

```
['Movie', 'Genre', 'Language', 'Country', 'Location', 'M_Location', 'M_Country', 'M_Langu
age', 'M_Genre', 'Person', 'M_Producer', 'M_Director', 'M_Cast']
```

In [38]:

```
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

	cid	name	type	notnull	dflt_value	pk
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

	cid	name	type	notnull	dflt_value	pk
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

	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 [39]:

```
%%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 m.Title,p.Name,m.year
                  FROM Movie m JOIN M_director d ON m.MID = d.MID
                  JOIN Person p ON d.PID = p.PID
                  JOIN M_Genre mg ON m.MID = mg.MID
```

```
                  JOIN Genre g ON g.GID = mg.GID
                  WHERE g.Name LIKE '%Comedy%'
                  AND(CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) %4 = 0 AND CAST(SUBSTR(TR
IM(m.year),-4) AS INTEGER) % 100 <> 0  OR  CAST(SUBSTR(TRIM(m.year),-4) AS INTEGER) % 400
= 0 ) """
grader_1(query1)
```

```
                              title              Name  year
0                         Mastizaade       Milap Zaveri  2016
1  Harold & Kumar Go to White Castle       Danny Leiner  2004
2                  Gangs of Wasseypur     Anurag Kashyap  2012
3        Around the World in 80 Days       Frank Coraci  2004
4             The Accidental Husband      Griffin Dunne  2008
5                             Barfi!        Anurag Basu  2012
6                  Bride & Prejudice    Gurinder Chadha  2004
7    Beavis and Butt-Head Do America         Mike Judge  1996
8                            Dostana   Tarun Mansukhani  2008
9                     Kapoor & Sons       Shakun Batra  2016
CPU times: total: 234 ms
Wall time: 226 ms
```

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

In [40]:

```
%%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 p
             WHERE trim(p.PID, ' ') IN
             (
             SELECT trim(mc.PID, ' ')
             FROM
             Movie m JOIN M_Cast mc ON m.MID = mc.MID
             WHERE m.title = 'Anand'
             AND m.year = 1971 )"""
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
CPU times: total: 469 ms
Wall time: 471 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 [41]:

```
%%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
CPU times: total: 844 ms
Wall time: 869 ms
```

In [42]:

```
%%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 = """
WITH
   ACTORS_BEFORE_1970 AS (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 ) ,


    ACTORS_AFTER_1990 AS ( 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 I
nteger)>1990
                                                  )) r2
```

```
                            ON r2.PD=p.PID )


        SELECT Name FROM Person WHERE PID IN (SELECT DISTINCT(PID)
                                                FROM ACTORS_BEFORE_1970
                                                 WHERE PID IN (SELECT PID FROM ACTOR
S_AFTER_1990))


"""

grader_3(query3)
```

```
                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
CPU times: total: 1 s
Wall time: 1.01 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 [43]:

```
%%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
movies they directed *** """
query_4a =""" SELECT
              PID director_id,
              COUNT(MID) number_of_movies_directed
              FROM M_Director
              GROUP BY director_id
              """

print(grader_4a(query_4a))

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

```
  director_id  number_of_movies_directed
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: 46.9 ms
Wall time: 24 ms
```

In [44]:

```
%%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 TRIM(P.NAME) director_name,
              nm.num_of_movies_directed
              FROM
                  (SELECT
                   PID, COUNT(MID) num_of_movies_directed
                   FROM M_Director
                   GROUP BY PID
                   HAVING num_of_movies_directed >= 10) nm
                   JOIN Person p
                   ON TRIM(nm.PID) = TRIM(p.PID)
                   ORDER BY nm.num_of_movies_directed DESC  """
grader_4(query4)
```

```
          director_name  num_of_movies_directed
0           David Dhawan                      39
1           Mahesh Bhatt                      35
2           Priyadarshan                      30
3       Ram Gopal Varma                       30
4           Vikram Bhatt                      29
5  Hrishikesh Mukherjee                       27
6            Yash Chopra                      21
7       Basu Chatterjee                       19
8        Shakti Samanta                       19
9           Subhash Ghai                      18
CPU times: total: 1.62 s
Wall time: 1.69 s
```

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

In [45]:

```
%%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 geneder *
** """
query_5aa = ''' SELECT
                mc.MID movie_id,
                p.Gender gender,
                count(p.Gender) gender_count
                FROM M_Cast mc
                JOIN Person p
                ON TRIM(mc.PID) = p.PID
                GROUP BY movie_id,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 yo
u have written above *** """
query_5ab = ''' SELECT
                mc.MID movie_id,
```

```
                p.Gender gender,
                count(p.Gender) gender_count
                FROM M_Cast mc
                JOIN Person p
                ON TRIM(mc.PID) = p.PID
                GROUP BY movie_id,gender
                HAVING (gender='Male')>=1
            '''

print(grader_5ab(query_5ab))


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

```
    movie_id  gender  gender_count
0  tt0021594    None             0
1  tt0021594  Female             3
2  tt0021594    Male             5
3  tt0026274    None             0
4  tt0026274  Female            11
5  tt0026274    Male             9
6  tt0027256    None             0
7  tt0027256  Female             5
8  tt0027256    Male             8
9  tt0028217  Female             3
True
    movie_id gender  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
CPU times: total: 1.36 s
Wall time: 1.42 s
```

In [46]:

```
%%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 = """ WITH gender_count AS (select TRIM(mc.MID) movie_id,p.Gender
                                        FROM M_Cast mc JOIN Person p ON TRIM(mc.PID)
= p.PID
                                        GROUP BY MID,Gender
                                    )

             SELECT
             CAST(SUBSTR(TRIM(year),-4) AS INTEGER) YEAR,
             COUNT(DISTINCT(MID)) FEMALE_ACTORS_COUNT
             FROM Movie WHERE TRIM(MID) NOT IN (
                                                SELECT movie_id
                                                FROM gender_count
                                                WHERE Gender NOT IN ("Female")
                                                )
             GROUP BY YEAR
             ORDER BY YEAR
                      """
grader_5a(query5a)
```

```
   YEAR  FEMALE_ACTORS_COUNT
0  1939                    1
1  1999                    1
2  2000                    1
```

```
3  2018                     1
CPU times: total: 656 ms
Wall time: 661 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 [47]:

```
%%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
                  gender_count AS (SELECT TRIM(mc.MID) movie_id,
                                          p.Gender
                                          FROM M_Cast mc
                                          JOIN
                                          Person p
                                          ON TRIM(mc.PID) = p.PID
                                          GROUP BY MID,Gender
                                            ),


                ONLY_FEMALE_ACTED_MOVIES AS
                                              ( SELECT
                                                CAST(SUBSTR(TRIM(year),-4) AS INTEGER) yea
r,
                                                COUNT(DISTINCT(MID)) female_actors_count
                                                FROM Movie
                                                WHERE TRIM(MID) NOT IN (
                                                                        SELECT movie_id
                                                                         FROM gender_cou
nt
                                                                         WHERE Gender NO
T IN ("Female")
                                                                        )
                                                GROUP BY YEAR
                                                ORDER BY YEAR ),

                    TOTAL_MOVIE_COUNT AS
                                          ( SELECT CAST(SUBSTR(year,-4) AS INTEGER) year,
                                            COUNT( TRIM(MID) ) total_movie_count
                                            FROM Movie
                                            GROUP BY CAST(SUBSTR(year,-4) AS INTEGER)
                                            )

                       SELECT
                       ofm.YEAR,
                       ((ofm.female_actors_count*1.0)/tmc.total_movie_count ) Percentage
_Female_Only_Movie ,
                         tmc.total_movie_count
                         FROM ONLY_FEMALE_ACTED_MOVIES ofm
                         JOIN TOTAL_MOVIE_COUNT tmc
                         ON ofm.year = tmc.year

      """
grader_5b(query5b)
```

```
   year  Percentage_Female_Only_Movie  total_movie_count
0  1939                      0.500000                  2
1  1999                      0.015152                 66
2  2000                      0.015625                 64
3  2018                      0.009615                104
```

```
CPU times: total: 641 ms
Wall time: 654 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 [48]:

```
%%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 count_people AS
                                  (SELECT TRIM(MID) movieid , COUNT(DISTINCT(PID)) cast_co
unt
                                   FROM M_cast GROUP BY TRIM(MID)
                                   )
                        SELECT m.title,cp.cast_count
                        FROM count_people cp JOIN Movie m
                        ON cp.movieid = TRIM(m.MID)
                        ORDER BY cp.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: 281 ms
Wall time: 299 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 [49]:

```
%%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(year,-4) AS INTEGER) year,
                                      COUNT( TRIM(MID) ) total_movie_count
                                      FROM Movie
                                      GROUP BY CAST(SUBSTR(year,-4) AS INTEGER) """
```

```
grader_7a(query7a)

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

```
   year  total_movie_count
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
CPU times: total: 15.6 ms
Wall time: 16 ms
```

In [50]:

```
%%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 = """   WITH MOVIE_COUNT AS (SELECT CAST(SUBSTR(year,-4) AS INTEGER) year,
                                    COUNT( TRIM(MID) ) total_movie_count
                                    FROM Movie
                                    GROUP BY CAST(SUBSTR(year,-4) AS INTEGER)
                                    ),


                        TOTAL_YEARS AS(  SELECT mc.year curr_year,
                                               mc.total_movie_count AS curr_mc,
                                               mc1.year second_year,
                                               mc1.total_movie_count AS second_mc
                                        FROM
                                        MOVIE_COUNT mc
                                        CROSS JOIN
                                        MOVIE_COUNT mc1

                                        )

                         SELECT curr_year AS Movie_Year,
                                curr_mc   AS Total_Movies,
                                second_year AS    Movie_Year ,
                                second_mc AS Total_Movies
                         FROM TOTAL_YEARS
                         WHERE  second_year <= curr_year+9 AND second_year >=curr_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: 31.2 ms
Wall time: 31.2 ms
```

In [51]:

```
%%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
                            DECADE_START_END AS
                                                ( SELECT DISTINCT
                                                  CAST(SUBSTR(year,-4) AS INTEGER) as dec
_start,
                                                  CAST(SUBSTR(year,-4) AS INTEGER)+9 as d
ec_end
                                                  FROM Movie
                                                  GROUP BY dec_end
                                                  ORDER BY dec_end
                                                ) ,

                        MOVIE_COUNT_EACH_YEAR AS
                                                ( SELECT CAST(SUBSTR(year,-4) AS INTEGER
) year,
                                                  COUNT( TRIM(MID) ) movie_count
                                                  FROM Movie
                                                  GROUP BY CAST(SUBSTR(year,-4) AS INTEG
ER)
                                                ),



                    DECADE_MOVIE_COUNT AS (SELECT dse.dec_start,
                                                  dse.dec_end,
                                                  SUM(mcy.movie_count) as dec_movie_coun
t
                                                  FROM DECADE_START_END dse,
                                                  MOVIE_COUNT_EACH_YEAR mcy
                                                  WHERE mcy.year BETWEEN dse.dec_start A
ND dse.dec_end
                                                  GROUP BY dse.dec_start
                                           )
                    SELECT
                          CAST(dec_start AS TEXT) ||' - ' ||  CAST(dec_end AS TEXT) AS D
ECADE,
                          MAX(dec_movie_count) AS total_movies
                          FROM DECADE_MOVIE_COUNT

                            """
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
```

```
        DECADE  total_movies
0  2008 - 2017          1203
CPU times: total: 31.2 ms
Wall time: 24 ms
```

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

In [52]:

```
%%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 mc.PID AS actor_id,
                            md.PID AS director_id,
                            COUNT(md.MID) AS no_of_movies
              FROM M_Director md,M_Cast mc
              WHERE md.MID = mc.MID
              GROUP BY mc.PID,md.PID
              ORDER BY COUNT(md.MID)"""
grader_8a(query8a)

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

```
     actor_id director_id  no_of_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
CPU times: total: 1.17 s
Wall time: 1.23 s
```

In [53]:

```
%%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
                    ACTOR_DIRECTOR_MOVIE_COUNT AS (SELECT mc.PID AS actor_id,
                                                          md.PID AS director_id,
                                                          COUNT( md.MID) AS no_of_mov
ies
                                                   FROM M_Director md,M_Cast mc
                                                   WHERE md.MID = mc.MID
                                                   GROUP BY mc.PID,md.PID

                                                   ),

                    YASH_CHPORA_ID AS (SELECT PID as yash_id
                                       FROM PERSON
                                       WHERE TRIM(Name) ='Yash Chopra' ),

                    ACTORS_IN_YASH_MOVIES AS (SELECT admc.actor_id AS A_ID,
                                                     admc.director_id AS Yash_ID,
                                                     admc.no_of_movies AS Movie_Count
                                              FROM
                                              ACTOR_DIRECTOR_MOVIE_COUNT admc
                                              JOIN
                                              YASH_CHPORA_ID yci
                                              ON TRIM(admc.director_id) = yci.yash_id
                                              ORDER BY admc.no_of_movies DESC
                                              ),

                      ACTORS_IN_OTHER_MOVIES AS ( SELECT *
                                                  FROM   ACTOR_DIRECTOR_MOVIE_COUNT
                                                  EXCEPT
```

```
                                        SELECT *
                                        FROM ACTORS_IN_YASH_MOVIES),

                      ACTORS_IN_OTHER_MOVIES_COUNT AS (SELECT actor_id AS A_ID,
                                                              MAX(no_of_movies) AS Max
_Movie_Count
                                                    FROM  ACTORS_IN_OTHER_MOVIES
                                                    GROUP BY actor_id
                                                    ),
                       ACTED_MORE_IN_YASH AS(SELECT aym.A_ID Actor_ID,
                                                    aym.Movie_Count Movie_Count,
                                                    aom.Max_Movie_Count other_movie_count
                                             FROM ACTORS_IN_YASH_MOVIES aym
                                             JOIN
                                             ACTORS_IN_OTHER_MOVIES_COUNT aom
                                             ON aym.A_ID = aom.A_ID

                                             ),

                       ACTED_ONLY_IN_YASH AS(SELECT A_ID,
                                                    Movie_Count
                                             FROM ACTORS_IN_YASH_MOVIES
                                             WHERE A_ID IN (SELECT A_ID FROM ACTORS_IN_YAS
H_MOVIES
                                                            EXCEPT
                                                            SELECT Actor_ID FROM ACTED_MO
RE_IN_YASH)
                                             ),

                      FINAL_ACTORS AS( SELECT Actor_ID,
                                              Movie_Count
                                      FROM ACTED_MORE_IN_YASH
                                      WHERE Movie_Count >=other_movie_count
                                      UNION ALL
                                      SELECT A_ID,
                                             Movie_Count
                                      FROM ACTED_ONLY_IN_YASH

                                      ORDER BY Movie_Count DESC
                                      )
                       SELECT p.Name,fa.Movie_Count
                       FROM FINAL_ACTORS fa
                       JOIN Person p
                       ON TRIM(Actor_ID) = p.PID
                       ORDER BY fa.Movie_Count DESC


                     """
grader_8(query8)
```

```
                Name  Movie_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)
CPU times: total: 2.11 s
Wall time: 2.1 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

the same film as some actor with Shahrukh number 1 have Shahrukh number 2, etc. Return all actors whose Shahrukh number is 2.

In [54]:

```
%%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 = """WITH SHAHRUKH_ID AS (SELECT PID AS shah_id
                                  FROM Person
                                  WHERE Name LIKE '%Shah Rukh Khan%'),

                  SHAHRUKH_MOVIES AS (SELECT DISTINCT(mc.MID) AS shah_movies,
                                             sid.shah_id AS shah_id
                                      FROM M_Cast mc
                                      JOIN SHAHRUKH_ID sid
                                      ON TRIM(mc.PID) = sid.shah_id
                                      ),

                  S1_ACTORS AS (SELECT DISTINCT(TRIM(mc.PID)) AS S1_PID
                                FROM M_Cast mc
                                JOIN
                                SHAHRUKH_MOVIES sm
                                ON mc.MID = sm.shah_movies
                                AND TRIM(mc.PID) <> sm.shah_id )

                  SELECT * FROM S1_ACTORS
                  """
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 acto
rs
# removing S1 actors from the combined list of S1 & S2 actors, so that we get only S2 act
ors
```

```
      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: 188 ms
Wall time: 197 ms
```

In [55]:

```
%%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 SHAHRUKH_ID AS (SELECT PID AS shah_id
                                  FROM Person
                                  WHERE Name LIKE '%Shah Rukh Khan%'),

                  SHAHRUKH_MOVIES AS (SELECT DISTINCT(mc.MID) AS shah_movies,
```

```
                                        sid.shah_id AS shah_id
                                    FROM M_Cast mc
                                    JOIN SHAHRUKH_ID sid
                                    ON TRIM(mc.PID) = sid.shah_id
                                    ),

                        S1_ACTORS AS (SELECT DISTINCT(TRIM(mc.PID)) AS S1_PID
                                        FROM M_Cast mc
                                        JOIN
                                        SHAHRUKH_MOVIES sm
                                        ON mc.MID = sm.shah_movies
                                        AND TRIM(mc.PID) <> sm.shah_id ),

                        S2_MOVIES AS (SELECT DISTINCT(mc.MID) AS s2_movie_id
                                      FROM M_Cast mc
                                      JOIN
                                      S1_ACTORS sa
                                      ON TRIM(mc.PID) = S1_PID),

                        S1_S2_ACTORS AS (SELECT DISTINCT(TRIM(mc.PID)) AS s1_s2_actor_id
                                         FROM M_Cast mc
                                         JOIN
                                         S2_MOVIES s2
                                         WHERE mc.MID = s2_movie_id
                                         ),

                        S2_ACTORS AS ( SELECT s1_s2_actor_id AS s2_actor_id
                                         FROM S1_S2_ACTORS

                                         EXCEPT

                                         SELECT S1_PID
                                         FROM S1_ACTORS
                                         )

                         SELECT Name AS Actor_Name FROM Person WHERE PID IN (SELECT s2_acto
r_id
                                                                 FROM S2_ACTORS
                                                                 WHERE s2_actor_id <> (SELEC
T shah_id FROM SHAHRUKH_ID))

                        """
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)
CPU times: total: 1.08 s
Wall time: 1.05 s
```

In [ ]: