SQL Assignment

In [5]:
```
import pandas as pd
import sqlite3

from IPython.display import display, HTML
```

In [6]:
```
# 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 [7]:
```
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='table'",conn)
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

	cid	name	type	notnull	dflt_value	pk
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.

```
-> We need to list down the director name, movie name, and the year. Director who directed comedy movies in a leap year.

-> Tables :
    Person     => Director Name
    Movie      => Movie Name, Year
    Genre      => Genre Name
    M_Genre    => MID, GID
    M_Director => MID, PID
```

In [ ]:

```
In [12]: %%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(per.Name) as Director_Name,
                TRIM(mov.title) as Movie_Name,
                TRIM(mov.year) as Movie_Year

        FROM
                Movie mov JOIN
                M_Genre mg
                ON mov.MID=mg.MID JOIN

                M_Director m_dir
                ON m_dir.MID=mg.MID JOIN

                Person per
                ON per.PID=m_dir.PID JOIN

                Genre g
                ON g.GID=mg.GID

        WHERE
                TRIM(g.Name) LIKE '%Comedy%'
                AND
                    (
                    CAST(SUBSTR(TRIM(mov.year),-4) AS INTEGER)%4 = 0
                    AND CAST(SUBSTR(TRIM(mov.year),-4) AS INTEGER)%100 <> 0
                    OR CAST(SUBSTR(TRIM(mov.year),-4) AS INTEGER)%400 =0
                    )


    '''
grader_1(query1)
```

```
      Director_Name                          Movie_Name Movie_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: 247 ms
```

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

```
Table required:
    => Person : For actor names
    => Movie  : For movie title and year
    => M_Cast : For all the actors who played in the Movie Anand (MID, PID)
```

In [17]:
```
%%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
            p.Name as Actor_Names
        FROM
            Movie m JOIN
            M_Cast mc
            ON TRIM(m.MID)=TRIM(mc.MID) JOIN

            Person p
            ON TRIM(mc.PID)=TRIM(p.PID)
        WHERE
            TRIM(m.title)='Anand'
            """
grader_2(query2)
```

```
         Actor_Names
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: 522 ms
```

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

```
step 1: List all the actors who acted in a film before 1970
step 2: List all the actors who acted in a film after  1990
NOTE: actors should not have worked in between 1970 and 1990.
step 3: JOIN all of that.



step 1:
```

```
table required:
Movie   => Movie year (MID)
Person  => Actor Name (PID)
M_Cast  => joining    (MID, PID)
```

In [18]:
```
%%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))

# INNER JOIN => Intersection part

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: 1.03 s
```

```
In [19]: %%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_EARLY_1970 AS
        (
            SELECT DISTINCT p.PID FROM Person p
            INNER JOIN
            (
                SELECT TRIM(mc.PID) PD FROM M_Cast mc
                WHERE mc.MID IN
                    (
                     SELECT mv.MID FROM Movie mv
                     WHERE
                         CAST(SUBSTR(mv.year, -4) AS INTEGER) < 1970
                    )
            )r1
            ON r1.PD=p.PID
        ),
        ACTORS_LATER_1990 AS
        (
            SELECT DISTINCT p.PID FROM Person p
            INNER JOIN
            (
                SELECT TRIM(mc.PID) PD FROM M_Cast mc
                WHERE mc.MID IN
                    (
                     SELECT mv.MID FROM Movie mv
                     WHERE
                         CAST(SUBSTR(mv.year, -4) AS INTEGER) > 1990
                    )
            )r1
            ON r1.PD=p.PID
        )
        SELECT
            DISTINCT
            TRIM(p.Name) Actor_Name
        FROM
            ACTORS_EARLY_1970 AE1970 JOIN

            ACTORS_LATER_1990 AL1990
            ON AE1970.PID = AL1990.PID JOIN

            Person p
            ON AE1970.PID = TRIM(p.PID)"""
grader_3(query3)
```

```
         Actor_Name
0      Rishi Kapoor
1  Amitabh Bachchan
2            Asrani
3      Zohra Sehgal
4    Parikshat Sahni
```

```
5     Rakesh Sharma
6       Sanjay Dutt
7         Ric Young
8             Yusuf
9    Suhasini Mulay
Wall time: 1.29 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.

```
Query 4a:
Write a query which will return all the directors'ID's along the number of movies they directed.

Solution:
    Tables required :
        Movie       => for MID
        M_Director  => for MID,PID
```

```
In [21]: %%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 m_dir.PID, COUNT(m_dir.MID) as NO_OF_MOVIES_DIRECTED

    FROM Movie m JOIN
    M_Director m_dir

    ON m.MID=m_dir.MID

    GROUP BY m_dir.PID
    '''
print(grader_4a(query_4a))

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

```
         PID  NO_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
Wall time: 44 ms
```

```
In [23]: %%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 p.Name Director_Name, COUNT(m_dir.MID) as NO_OF_MOVIES_DIRECTED

    FROM Movie m JOIN
    M_Director m_dir

    ON m.MID=m_dir.MID JOIN
    Person p

    ON p.PID=m_dir.PID

    GROUP BY m_dir.PID

    HAVING NO_OF_MOVIES_DIRECTED>=10

    ORDER BY NO_OF_MOVIES_DIRECTED
    '''
grader_4(query4)
```

```
         Director_Name  NO_OF_MOVIES_DIRECTED
0           Raj Kapoor                     10
1           K. Bapaiah                     10
2      Vishal Bhardwaj                     10
3           N. Chandra                     10
4     Tigmanshu Dhulia                     10
5          J.P. Dutta                     10
6          Mehul Kumar                     10
7         Hansal Mehta                     10
8        Sudhir Mishra                     10
9  K. Muralimohana Rao                     10
Wall time: 221 ms
```

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

```
Query 5aa: Write a Query that will get movie id, and number of people for each gender.
Solution:
    Tables required:
        Movie : MID
        Person: PID, Gender
        M_Cast: MID, PID
```

```
In [27]: %%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(p.Gender)
             FROM

                     M_Cast mc JOIN
                     Person p
                     ON p.PID=TRIM(mc.PID)

             GROUP BY
                     mc.MID,p.Gender

             ORDER BY
                     mc.MID
             '''

         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(p.Gender)
                                 FROM
                                         M_Cast mc JOIN
                                         Person p
                                         ON p.PID=TRIM(mc.PID)
                                 GROUP BY
                                         mc.MID,p.Gender
                                 HAVING
                                         Gender='Male' AND COUNT(p.Gender)>=1
                     """

         print(grader_5ab(query_5ab))


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

```
         MID  Gender  COUNT(p.Gender)
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
         MID Gender  COUNT(p.Gender)
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: 1.06 s
```

```
In [28]: %%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(year,-4) AS UNSIGNED) year,
            COUNT(DISTINCT TRIM(MID) ) NUM_OF_MOV_WITH_FEMALES_ONLY
    FROM
            Movie
    WHERE
            TRIM(MID) NOT IN (
                    SELECT
                            DISTINCT
                            TRIM(mc.MID) MID
                    FROM
                                M_Cast mc JOIN
                                Person p

                                ON TRIM(mc.PID) = p.PID
                    WHERE
                                p.Gender IN ('Male','None')
                )
    GROUP BY
            CAST(SUBSTR(year,-4) AS UNSIGNED)
    ORDER BY
            year"""
grader_5a(query5a)
```

```
   year  NUM_OF_MOV_WITH_FEMALES_ONLY
0  1939                             1
1  1999                             1
2  2000                             1
3  2018                             1
Wall time: 519 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 [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
            CAST(SUBSTR(mv.year,-4) AS UNSIGNED) year,
            (ofcm.NUM_OF_MOV_WITH_FEMALES_ONLY/(COUNT(mv.MID)*1.0)) as
                Percentage_Female_Only_Movie,
            COUNT(mv.MID) as Total_Movies
    FROM
            Movie mv JOIN
            (
                SELECT
                        CAST(SUBSTR(m.year,-4) AS UNSIGNED) year,
                        COUNT(DISTINCT TRIM(MID) ) NUM_OF_MOV_WITH_FEMALES_ONLY
                FROM
                        Movie m
                WHERE
                        TRIM(MID) NOT IN (
                                SELECT
                                        DISTINCT
                                        TRIM(mc.MID) MID
                                FROM
                                            M_Cast mc JOIN
                                            Person p

                                            ON TRIM(mc.PID) = p.PID
                                WHERE
                                            p.Gender IN ('Male','None')
                            )
                GROUP BY
                        CAST(SUBSTR(m.year,-4) AS UNSIGNED)
                ORDER BY
                        m.year
            )ofcm

            ON ofcm.year=CAST(SUBSTR(mv.year,-4) AS UNSIGNED)
    GROUP BY
            CAST(SUBSTR(mv.year,-4) AS UNSIGNED)
            """
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: 501 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.

```
Tables required:
    Movie : (MID,title) movie title
    M_Cast: (MID,PID)
    Person: (PID)
```

find no.of person per movies

In [32]:
```
%%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 title,
            actor.NUMBER_OF_ACTOR count
    FROM
            Movie m JOIN
            (
                SELECT
                        TRIM(MID) MID,
                        COUNT(DISTINCT TRIM(PID)) NUMBER_OF_ACTOR
                FROM
                        M_Cast
                GROUP BY
                        TRIM(MID)
                ORDER BY NUMBER_OF_ACTOR DESC
            )
            actor
            ON
            actor.MID=m.MID
        """
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: 308 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 [34]:
```
%%time
def grader_7a(q7a):
    q7a_results  = pd.read_sql_query(q7a,conn)
    print(q7a_results.head(10))
    assert (q7a_results.shape == (78, 2))

query7a = """
    SELECT
            CAST(SUBSTR(m.year,-4) AS UNSIGNED) Movie_year,
            COUNT(m.MID) Total_Movies

    FROM
            Movie m
    GROUP BY
            CAST(SUBSTR(m.year,-4) AS UNSIGNED)
    """
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: 11 ms
```

```
In [35]: %%time
def grader_7b(q7b):
    q7b_results  = pd.read_sql_query(q7b,conn)
    print(q7b_results.head(10))
    assert (q7b_results.shape == (713, 4))

query7b = """
    SELECT
            *
    FROM
            (
            SELECT
                    CAST(SUBSTR(m.year,-4) AS UNSIGNED) Movie_year,
                    COUNT(m.MID) Total_Movies

            FROM
                    Movie m
            GROUP BY
                    CAST(SUBSTR(m.year,-4) AS UNSIGNED)
            )table1
            JOIN
            (
            SELECT
                    CAST(SUBSTR(m.year,-4) AS UNSIGNED) Movie_year,
                    COUNT(m.MID) Total_Movies

            FROM
                    Movie m
            GROUP BY
                    CAST(SUBSTR(m.year,-4) AS UNSIGNED)
            )table2
            ON table2.Movie_year <= table1.Movie_year+9
               AND
               table2.Movie_year >=table1.Movie_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
Wall time: 12.1 ms
```

```
In [8]: %%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
                NUMBER_OF_MOVIES_PER_YEAR AS
                    (
                    SELECT
                            COUNT(DISTINCT m.MID) num_of_movies,
                            CAST(SUBSTR(m.year,-4) as UNSIGNED) year
                    FROM
                            Movie m
                    GROUP BY
                            CAST(SUBSTR(m.year,-4) as UNSIGNED)
                    ),
                DECADE_START_END AS
                    (
                    SELECT
                            DISTINCT
                            CAST(SUBSTR(m.year,-4) as UNSIGNED) year,
                            CAST(SUBSTR(m.year,-4) as UNSIGNED) decade_start,
                            CAST(SUBSTR(m.year,-4) as UNSIGNED)+9 decade_end,
                            SUBSTR(m.year,-4) Decade
                    From
                            Movie m
                    ),
                Number_Of_Movies_in_decade AS
                    (
                    SELECT
                            SUM(num_of_Movies) Total_Number_of_Movies,
                            dse.decade
                    FROM
                            NUMBER_OF_MOVIES_PER_YEAR mpy,
                            DECADE_START_END dse
                    WHERE
                            mpy.year BETWEEN dse.decade_start AND dse.decade_end
                    GROUP BY
                            dse.decade
                    )
            SELECT
                    decade,
                    Total_Number_of_Movies as Decade_Movie_Count
            FROM
                    Number_of_Movies_in_decade
            WHERE
                    Total_Number_of_Movies = (
                                SELECT
                                        MAX(Total_Number_of_Movies)
                                FROM
                                        Number_of_Movies_in_decade
                )
            """
        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  Decade_Movie_Count
0   2008                1203
Wall time: 61 ms
```

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

```
Required Tables:

M_Cast: MID, PID
M_Director: MID, PID
Person    : PID, Name, Gender
```

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
                m_cast.PID Actor_PID,
                m_dir.PID Director_PID,
                COUNT(DISTINCT TRIM(m_dir.MID)) AS num_of_movies
        FROM
                M_Director m_dir JOIN
                M_Cast m_cast
                ON
                TRIM(m_dir.MID)=TRIM(m_cast.MID)
        GROUP BY
                Actor_PID, Director_PID
    """
grader_8a(query8a)

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

```
    Actor_PID Director_PID  num_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
Wall time: 3min 2s
Parser   : 221 ms
```

```
In [ ]:
```

In [14]:
```
%%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
        distinct TRIM(p.Name) Actor_Name, res.movie_count
    from
        Person p JOIN
        (
            SELECT
                distinct Actor_PID,Director_PID, movie_count
            from
                (
                SELECT
                        TRIM(m_cast.PID) Actor_PID,
                        TRIM(m_dir.PID) Director_PID,
                        COUNT(DISTINCT TRIM(m_dir.MID)) movie_count
                FROM
                        M_Director m_dir,
                        M_Cast m_cast
                WHERE
                        TRIM(m_dir.MID)=TRIM(m_cast.MID)
                GROUP BY
                        Actor_PID, Director_PID
                )
            where
                    (Actor_PID,movie_count)
                    IN(
                        select
                            Actor_PID,
                            MAX(ifnull(movies,0)) as movie_count
                        from
                                (
                                SELECT
                                        TRIM(m_cast.PID) Actor_PID,
                                        TRIM(m_dir.PID) Director_PID,
                                        COUNT(DISTINCT TRIM(m_dir.MID))  movies
                                FROM
                                        M_Director m_dir,
                                        M_Cast m_cast
                                WHERE
                                        TRIM(m_dir.MID)=TRIM(m_cast.MID)

                                GROUP BY
                                        Actor_PID, Director_PID
                                )
                        group by
                            Actor_PID
                    )
                    AND
                    Director_PID=(select TRIM(PID) from Person where TRIM(Name)='Yash Chopra')
        )res
        ON TRIM(p.PID)=res.Actor_PID
    order by
        movie_count desc"""
grader_8(query8)
```

```
         Actor_Name  movie_count
0       Jagdish Raj           11
1  Manmohan Krishna           10
2         Iftekhar             9
3     Shashi Kapoor            7
4    Waheeda Rehman            5
5     Rakhee Gulzar            5
6    Achala Sachdev            4
7       Neetu Singh            4
8          Ravikant            4
9   Parikshat Sahni            3
(243, 2)

---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_8(q8)

AssertionError:
```

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.

Query(Made Easy): Here you have to print all the actors who have acted with the co-actors of Shahrukh Khan and not the co-actors of Shahrukh Khan. For example, Kajol is a co-actor of Shahrukh khan then you have to include all the co-actors of Kajol who didn't act with Shahrukh Khan.

In [ ]:

In [ ]:

```
In [34]: %%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(mc.PID) PID
        from
                M_Cast mc
        where
                TRIM(mc.MID) IN (
                    select
                            distinct
                            TRIM(MID) MID
                    from
                            M_Cast
                    where
                            TRIM(PID)=(
                                select
                                        TRIM(PID)
                                from
                                        Person
                                where
                                        Name like '%Shah Rukh Khan%')
                    )
                AND
                TRIM(mc.PID)<>(select TRIM(PID) from Person 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 actors
# removing S1 actors from the combined list of S1 & S2 actors, so that we get only S2 actors
```

```
         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: 172 ms
```

In [28]:
```
%%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 Actor_Name
    FROM
        Person p
    WHERE
        trim(p.PID) IN
        (
            select
                distinct
                TRIM(PID) PID
            from
                M_Cast
            where
                TRIM(MID) IN
                (
                select
                    distinct
                    TRIM(m_cast.MID) MID
                from
                    M_Cast m_cast
                where
                    TRIM(m_cast.PID) IN
                    (
                    select
                        distinct
                        TRIM(mc.PID) PID
                    from
                        M_Cast mc
                    where
                        TRIM(mc.MID) IN (
                            select
                                distinct
                                TRIM(MID) MID
                            from
                                M_Cast
                            where
                                TRIM(PID)=(
                                    select
                                        TRIM(PID)
                                    from
                                        Person
                                    where
                                        Name like '%Shah Rukh Khan%'
                                    )
                        )
                        AND
                        TRIM(mc.PID)<>(select TRIM(PID) from Person where Name like '%Shah Rukh Khan%')
                    )

                )AND
                TRIM(PID) NOT IN
                (
                    select
                        distinct
                        TRIM(mc.PID) PID
                    from
                        M_Cast mc
                    where
                        TRIM(mc.MID) IN (
                            select
                                distinct
                                TRIM(MID) MID
                            from
                                M_Cast
                            where
                                TRIM(PID)=(
                                    select
                                        TRIM(PID)
                                    from
                                        Person
                                    where
                                        Name like '%Shah Rukh Khan%'
                                    )
                        )

                )
        )"""
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: 939 ms
```

In [ ]:

In [ ]: