```
#                    -------------------- SQL on IMBD --------------------


from PIL import Image

# image_file = 'drive/My Drive/IMDB SQL/db_schema.jpeg'

image_file = 'db_schema.jpeg'

with Image.open(image_file) as image:
    width, height = image.size


from IPython.display import Image
Image('db_schema.jpeg')
print()
print('*'*61)
print("Below is the image which displays the tables and it's columns")
print('*'*61)
```

```

*************************************************************
Below is the image which displays the tables and it's columns
*************************************************************
```

Source Links

- https://www.appliedaicourse.com/
- https://www.w3schools.com/sql/default.asp
- https://www.w3resource.com/sql-exercises/movie-database-exercise/joins-exercises-on-movie

```
import numpy as np
import pandas as pd
import sqlite3


# Create a connection with database
con = sqlite3.connect('Db-IMDB.db')

# Load all the table
table = pd.read_sql_query("SELECT * FROM sqlite_master WHERE type = 'table'", con)


# Create a connection with database
con = sqlite3.connect('Db-IMDB.db')

# Load all the table
table = pd.read_sql_query("SELECT * FROM sqlite_master WHERE type = 'table'", con)


table
```

	type	name	tbl_name	rootpage	
0	table	Movie	Movie	2	CREATE TABLE "Movie" (\n"index" INTEGER,\n "I
1	table	Genre	Genre	4	CREATE TABLE "Genre" (\n"index" INTEGER,\n "
2	table	Language	Language	5	CREATE TABLE "Language" (\n"index" INTEGER,\
3	table	Country	Country	6	CREATE TABLE "Country" (\n"index" INTEGER,\
4	table	Location	Location	7	CREATE TABLE "Location" (\n"index" INTEGER,\
5	table	M_Location	M_Location	11	CREATE TABLE "M_Location" (\n"index" INTEGER
6	table	M_Country	M_Country	10	CREATE TABLE "M_Country" (\n"index" INTEGER,\
7	table	M_Language	M_Language	9	CREATE TABLE "M_Language" (\n"index" INTEGER
8	table	M_Genre	M_Genre	8	CREATE TABLE "M_Genre" (\n"index" INTEGER,\
9	table	Person	Person	12	CREATE TABLE "Person" (\n"index" INTEGER,\n
10	table	M_Producer	M_Producer	14	CREATE TABLE "M_Producer" (\n"index" INTEGER
11	table	M_Director	M_Director	15	CREATE TABLE "M_Director" (\n"index" INTEGER
12	table	M_Cast	M_Cast	16	CREATE TABLE "M_Cast" (\n"index" INTEGER,\n

```
print("Type: table\n")
print("Number of unique names:", table['name'].nunique(), '\n')
print("Number of unique table names:", table['tbl_name'].nunique(), '\n')
print("Unique names:\n")
print(table['name'].unique())
```

```
Type: table

Number of unique names: 13

Number of unique table names: 13

Unique names:

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

```
print("\nDistinct tables are\n")

pd.read_sql_query("SELECT DISTINCT tbl_name FROM sqlite_master WHERE type = 'table'", con)
```

```
Distinct tables are
```

	tbl_name
0	Movie
1	Genre
2	Language
3	Country
4	Location
5	M_Location
6	M_Country
7	M_Language
8	M_Genre
9	Person
10	M_Producer
11	M_Director
12	M_Cast

```
# We will check if the year feature has no odd values.

pd.read_sql_query("SELECT DISTINCT YEAR FROM MOVIE", con)
```

```
        year
  0     2018
  1     2012
  2     2016
  3     2017
  4     2008
...      ...
120  IV 2017
121     1943
122     1950
123   I 1969
124  II 2009
125 rows × 1 columns
```

Removing Roman Numerals from year column

```
rom_year = pd.read_sql_query("select year from Movie where year LIKE '%I%'", con)
rom_year.head(5)
```

```
        year
0     I 2009
1     I 2018
2  XVII 2016
3     I 2017
4    II 2018
```

Observations:

1.We can see few odd values such as 'IV 2011' instead of 2011, 'II 1983' instead of 1983, etc. 2.We wi

Assignment

1.List all the directors who directed a 'Comedy' movie in a leap year. (You need to check that the genr
query should return director name, the movie name, and the year.

```
rows = pd.read_sql_query('''select Name,title,year from Person p join M_Director md on p.PID
                            (select MID from Movie where (year%4=0 and year%100!=0) and (year
                            (select MID from M_Genre where GID in(select GID from Genre where
print(rows.head(10))
```

```
             Name                              title  year
0    Milap Zaveri                         Mastizaade  2016
1    Milap Zaveri                         Mastizaade  2016
2    Danny Leiner  Harold & Kumar Go to White Castle  2004
3    Danny Leiner  Harold & Kumar Go to White Castle  2004
4  Anurag Kashyap                 Gangs of Wasseypur  2012
5  Anurag Kashyap                 Gangs of Wasseypur  2012
6    Frank Coraci        Around the World in 80 Days  2004
7    Frank Coraci        Around the World in 80 Days  2004
8   Griffin Dunne             The Accidental Husband  2008
9   Griffin Dunne             The Accidental Husband  2008
```

2. List the names of all the actors who played in the movie 'Anand' (19

```
#''
Movie_Anand_Actors = pd.read_sql_query("""

                                            WITH

                                            MOVIE_ANAND AS
                                            (SELECT M.title Movie, MC.PID
                                            FROM MOVIE M
                                            JOIN M_CAST MC ON M.MID = MC.MID
                                            WHERE M.title = 'Anand')

                                            SELECT DISTINCT TRIM(P.NAME) Actors
                                            FROM MOVIE_ANAND MA
                                            JOIN PERSON P ON TRIM(P.PID) = TRIM(MA.PID)
                                          """, con)


print("\nList of all actors who played in the movie Ananda (1971):")
print('-'*58, '\n')
print(Movie_Anand_Actors)
```

```
List of all actors who played in the movie Ananda (1971):
----------------------------------------------------------

              Actors
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
10    Brahm Bhardwaj
11      Gurnam Singh
12      Lalita Pawar
13       Durga Khote
14        Dara Singh
15     Johnny Walker
16         Moolchand
```

```
## 3) List all the actors who acted in a film before 1970 and in a film after 1990. (That is:

Actors_1970_1990 = pd.read_sql_query("""

                                    WITH

                                    Actors_Before_1970 AS
                                    (SELECT DISTINCT TRIM(MC.PID) PID
                                    FROM MOVIE M
                                    JOIN M_CAST MC ON M.MID = MC.MID
                                    WHERE CAST(SUBSTR(M.year,-4) AS UNASSIGNED) < 1970),

                                    Actors_After_1990 AS
                                    (SELECT DISTINCT TRIM(MC.PID) PID
                                    FROM MOVIE M
                                    JOIN M_CAST MC ON M.MID = MC.MID
                                    WHERE CAST(SUBSTR(M.year,-4) AS UNASSIGNED) > 1990),

                                    ACTORS AS
                                    (SELECT A_1970.PID PID
                                    FROM Actors_Before_1970 A_1970
                                    JOIN Actors_After_1990 A_1990 ON A_1970.PID = A_1990.PID)

                                    SELECT DISTINCT TRIM(P.NAME) Actors_Before_1970_After_199
                                    FROM PERSON P
                                    JOIN ACTORS A ON A.PID = TRIM(P.PID)

                                    """, con)
```

```
print("\nActors who worked in movies before 1970 and after 1990:")
print('-'*55, '\n')
print(Actors_1970_1990)
```

```
Actors who worked in movies before 1970 and after 1990:
-------------------------------------------------------

    Actors_Before_1970_After_1990
0                     Rishi Kapoor
1                 Amitabh Bachchan
2                           Asrani
3                     Zohra Sehgal
4                  Parikshat Sahni
..                             ...
295                         Poonam
296                  Jamila Massey
297                    K.R. Vijaya
298                          Sethi
299                   Suryakantham

[300 rows x 1 columns]
```

```
## 4. List all directors who directed 10 movies or more, in descending order of the number of


Directors_Movies_More_Than_10 = pd.read_sql_query("""

                                                    SELECT DISTINCT TRIM(P.Name) Directors, C
                                                    FROM PERSON P
                                                    JOIN M_DIRECTOR MD ON TRIM(P.PID) = TRIM(
                                                    GROUP BY Directors
                                                    HAVING Movie_Count >= 10
                                                    ORDER BY Movie_Count DESC


                                                """, con)


print("\nList of all directors who directed 10 or more movies:")
print('-'*69, '\n')
print(Directors_Movies_More_Than_10)
```

```
List of all directors who directed 10 or more movies:
---------------------------------------------------------------------

                   Directors  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            Basu Chatterjee           19
8             Shakti Samanta           19
9               Subhash Ghai           18
10  Abbas Alibhai Burmawalla           17
11          Rama Rao Tatineni          17
12              Shyam Benegal          17
13                     Gulzar          16
14             Manmohan Desai          16
15               Raj N. Sippy          16
16           Mahesh Manjrekar          15
17                 Raj Kanwar          15
18                Indra Kumar          14
19               Rahul Rawail          14
20                 Raj Khosla          14
21          Rajkumar Santoshi          14
22  Ananth Narayan Mahadevan           13
23             Anurag Kashyap          13
24                  Dev Anand          13
25               Harry Baweja          13
26        K. Raghavendra Rao           13
27              Rakesh Roshan          13
28                Vijay Anand          13
29               Anees Bazmee          12
30                Anil Sharma          12
31               Guddu Dhanoa          12
32          Madhur Bhandarkar          12
33            Nagesh Kukunoor          12
34                Prakash Jha          12
35              Prakash Mehra          12
36               Rohit Shetty          12
37             Satish Kaushik          12
38                Umesh Mehra          12
39            Govind Nihalani          11
40                Ketan Mehta          11
41                 Mohit Suri          11
42             Nasir Hussain           11
43         Pramod Chakravorty          11
44               Sanjay Gupta          11
45                  Bimal Roy          10
46               Hansal Mehta          10
47              J. Om Prakash          10
48                J.P. Dutta           10
49                 K. Bapaiah          10
50        K. Muralimohana Rao          10
51                Mehul Kumar          10
```

```
     52                N. Chandra           10
     53           Pankaj Parashar           10
     54                Raj Kapoor           10
     55             Sudhir Mishra           10
     56          Tigmanshu Dhulia           10
     57           Vishal Bhardwaj           10
```

```
## 5.a) For each year, count the number of movies in that year that had only female actors.

Female_Actors = pd.read_sql_query("""

                                    WITH

                                    MOVIE_MALE_NONE AS
                                    (SELECT MC.MID MID_F
                                    FROM M_CAST MC
                                    JOIN PERSON P ON TRIM(P.PID) = TRIM(MC.PID)
                                    WHERE TRIM(P.GENDER) IN ('Male', 'None'))

                                    SELECT CAST(SUBSTR(M.year,-4) AS UNASSIGNED) Year, COUNT(
                                    FROM MOVIE M
                                    WHERE TRIM(MID) NOT IN (SELECT MID_F FROM MOVIE_MALE_NONE
                                    GROUP BY CAST(SUBSTR(M.year,-4) AS UNASSIGNED)
                                    ORDER BY Year

                                   """, con)

print("\nYearly count of movies which has only female actors:")
print('-'*69, '\n')
print(Female_Actors)
```

```
5.
b) Now include a small change: report for each year the percentage of movies in that year wit

Female_Movie_Percentage = pd.read_sql_query("""

                                            WITH

                                            Movie_Non_Females AS
```

```
                                    (SELECT DISTINCT TRIM(MC.MID) MID
                                    FROM M_CAST MC
                                    JOIN PERSON P ON TRIM(MC.PID) = TRIM(P.PID)
                                    WHERE TRIM(P.GENDER) IN ('Male','None')),

                                    MOVIE_FEMALE_Year AS
                                    (SELECT CAST(SUBSTR(M.YEAR,-4) AS UNASSIGNED) YEA
                                    COUNT(DISTINCT TRIM(MID)) Female_Movie_Only
                                    FROM Movie M
                                    WHERE TRIM(MID) NOT IN (SELECT MID FROM Movie_Non
                                    GROUP BY CAST(SUBSTR(M.year,-4) AS UNASSIGNED)),

                                    MOVIES_YEAR AS
                                    (SELECT CAST(SUBSTR(M.YEAR,-4) AS UNASSIGNED) Yea
                                    COUNT(DISTINCT TRIM(MID)) Total_Movies
                                    FROM MOVIE M
                                    GROUP BY CAST(SUBSTR(M.YEAR,-4) AS UNASSIGNED))

                                    SELECT MY.YEAR, MY.Total_Movies,
                                    ROUND((IFNULL(MF.Female_Movie_Only, 0) * 100)/MY.
                                    FROM MOVIES_YEAR MY
                                    LEFT OUTER JOIN MOVIE_FEMALE_Year MF ON
                                    TRIM(MY.YEAR) = TRIM(MF.YEAR)
                                    ORDER BY Female_Movie_Percentage DESC

                                    """, con)


print("\nPercentage of movies with only females")
print('-'*38, '\n')
print(Female_Movie_Percentage)
```

```
Percentage of movies with only females
--------------------------------------

    Year  Total_Movies  Female_Movie_Percentage
0   1939             2                     50.0
1   1999            66                      1.0
2   2000            64                      1.0
3   2018           104                      1.0
4   1931             1                      0.0
..   ...           ...                      ...
73  2013           136                      0.0
74  2014           126                      0.0
75  2015           119                      0.0
76  2016           129                      0.0
77  2017           126                      0.0

[78 rows x 3 columns]
```

6) Find the film(s) with the largest cast. Return the movie title and the size of the cast. B

```
Largest_Cast = pd.read_sql_query("""


                                    WITH

                                    CAST AS
                                    (SELECT COUNT(DISTINCT TRIM(MC.PID)) Cast_Count, MC.MID M
                                    FROM M_CAST MC
                                    GROUP BY MC.MID)

                                    SELECT M.title Movie, C.Cast_Count
                                    FROM MOVIE M
                                    JOIN CAST C ON C.MID = M.MID
                                    ORDER BY Cast_Count DESC
                                    LIMIT 1

                                  """, con)


print("\nLargest cast movie and count:")
print('-'*29, '\n')
print(Largest_Cast)
```

```
 7) A decade is a sequence of 10 consecutive years. For example, say in your database you hav
```

```
Decade_More_Movies = pd.read_sql_query("""


                                        WITH

                                        YEARS_UNIQUE AS
                                       (SELECT DISTINCT
                                        CAST(SUBSTR(year,-4) AS UNSIGNED) YEAR,
                                        CAST(SUBSTR(year,-4) AS UNSIGNED) DECADE_START,
                                        CAST(SUBSTR(year,-4) AS UNSIGNED) + 9 DECADE_END,
                                        'Decade of : ' || SUBSTR(year,-4) DECADE
                                        FROM MOVIE),

                                        MOVIE_COUNT_YEARS AS
                                       (SELECT COUNT(DISTINCT MID) Movie_Count, CAST(SUBSTR(y
                                        FROM MOVIE
                                        GROUP BY CAST(SUBSTR(year,-4) AS UNSIGNED)),

                                        MOVIE_COUNT_DECADE AS
                                       (SELECT SUM(Movie_Count) Total_Movies, YU.DECADE
                                        FROM MOVIE_COUNT_YEARS MCY, YEARS_UNIQUE YU
                                        WHERE MCY.YEAR BETWEEN YU.DECADE_START AND YU.DECADE_
                                        GROUP BY YU.DECADE)

                                        SELECT Decade, Total_Movies
                                        FROM MOVIE_COUNT_DECADE
                                        WHERE Total_Movies = (SELECT MAX(Total_Movies)
                                        FROM MOVIE_COUNT_DECADE)

                                        """, con)


print("\nMore movies in decade:")
print('-'*22, '\n')
print(Decade_More_Movies)
```

```
More movies in decade:
----------------------

             DECADE  Total_Movies
0  Decade of : 2008          1205
```

8) Find the actors that were never unemployed for more than 3 years at a stretch. (Assume that the a
consecutive movies).

```
Actors_Never_Unemployed_More_3 = pd.read_sql_query("""

                        WITH

                        ACTORS_MOVIE_YEAR AS
                        (SELECT TRIM(MC.PID) PID, CAST(SUBSTR(year,-4) AS UNASSIGNED) Year,
                        COUNT(DISTINCT TRIM(M.MID)) Number_of_Mov
                        FROM M_CAST MC, MOVIE M
                        WHERE TRIM(MC.MID) = TRIM(M.MID)
                        GROUP BY TRIM(MC.PID), CAST(SUBSTR(year,-4) AS UNASSIGNED)
                        ORDER BY NUMBER_OF_MOV DESC),

                        ACTORS_MORE_THAN_YEAR AS
                        (SELECT AMY.PID, COUNT(AMY.YEAR) AS Number_of_Years, MIN(AMY.YEAR) AS
                        MAX(AMY.YEAR) AS Max_Year
                        FROM ACTORS_MOVIE_YEAR AMY
                        GROUP BY AMY.PID
                        HAVING COUNT(AMY.YEAR) > 1),

                        ACTORS_NUMBER_MORE_THAN_YEAR AS
                        (SELECT AMY.PID, AMY.YEAR, AMY.YEAR+4 AS Year_4, AMY.NUMBER_OF_MOV,
                        ATY.MIN_YEAR, ATY.MAX_YEAR
                        FROM ACTORS_MOVIE_YEAR AMY, ACTORS_MORE_THAN_YEAR ATY
                        WHERE AMY.PID = ATY.PID),

                        NUMBER_MOVIE_PRESENT AS
                        NUMBER_MOVIE_PRESENT AS
                        (SELECT AM.PID, NY.YEAR, SUM(AM.NUMBER_OF_MOV) AS NUMBER_OF_MOVIE_PRE
                        FROM ACTORS_NUMBER_MORE_THAN_YEAR AM, ACTORS_NUMBER_MORE_THAN_YEAR NY
                        WHERE AM.PID = NY.PID AND
                        AM.YEAR BETWEEN NY.MIN_YEAR AND NY.YEAR
                        GROUP BY AM.PID, NY.YEAR),

                        ACTOR_MOVIE_4_YEAR AS
                        (SELECT AM.PID, NY.YEAR, SUM(AM.NUMBER_OF_MOV) AS ACTOR_MOVIE_4_YEARS
                        FROM ACTORS_NUMBER_MORE_THAN_YEAR AM, ACTORS_NUMBER_MORE_THAN_YEAR NY
                        WHERE AM.PID = NY.PID AND
                        AM.YEAR BETWEEN NY.MIN_YEAR AND NY.YEAR_4 AND
                        NY.YEAR_4 <= NY.MAX_YEAR
                        GROUP BY AM.PID, NY.YEAR)

                        SELECT DISTINCT TRIM(P.NAME) AS ACTORS_NEVER_UNEMPLOYED_MORE_THAN_3_Y
                        FROM PERSON P
                        WHERE TRIM(P.PID) NOT IN (SELECT DISTINCT NMP.PID
                        FROM NUMBER_MOVIE_PRESENT NMP, ACTOR_MOVIE_4_YEAR AM_4
                        WHERE NMP.PID = AM_4.PID AND
                        NMP.YEAR = AM_4.YEAR AND
                        NMP.NUMBER_OF_MOVIE_PRESENT = AM_4.ACTOR_MOVIE_4_YEARS_PRESENT)


                        """, con)

print("\nActors who were never unemployed for more than 3 years at stretch:")
print('-'*70, '\n')
print(Actors_Never_Unemployed_More_3)
```

```
Actors who were never unemployed for more than 3 years at stretch:
----------------------------------------------------------------------

      ACTORS_NEVER_UNEMPLOYED_MORE_THAN_3_YEARS
0                                Christian Bale
1                                Cate Blanchett
2                           Benedict Cumberbatch
3                                 Naomie Harris
4                                   Andy Serkis
...                                         ...
32580                            Deepak Ramteke
32581                              Kamika Verma
32582                         Dhorairaj Bhagavan
32583                              Nasir Shaikh
32584                              Adrian Fulle

[32585 rows x 1 columns]
```

9) Find all the actors that made more movies with Yash Chopra than ar

```
Actors_more_movie_Yash = pd.read_sql_query("""

                                WITH

                                YASH_CHOPRA AS
                                (SELECT TRIM(P.PID) PID
                                FROM PERSON P
                                WHERE Trim(P.NAME) = 'Yash Chopra'),

                                MOVIES_COUNT AS
                                (SELECT TRIM(MC.PID) ACTORS, TRIM(MD.PID) DIRECTORS,
                                COUNT(DISTINCT TRIM(MD.MID)) MOVIE_COUNT
                                FROM M_CAST MC, M_DIRECTOR MD
                                WHERE TRIM(MC.MID) = TRIM(MD.MID)
                                GROUP BY ACTORS, DIRECTORS),

                                YASH_MOVIE_COUNT AS
                                (SELECT CM.ACTORS, CM.DIRECTORS,
                                CM.MOVIE_COUNT MOVIE_COUNT_YASH
                                FROM MOVIES_COUNT CM, YASH_CHOPRA YC
                                WHERE CM.DIRECTORS = YC.PID),

                                OTHER_DIRECTORS AS
                                (SELECT ACTORS, MAX(MOVIE_COUNT) MAX_MOVIE_COUNT
                                FROM MOVIES_COUNT CM, YASH_CHOPRA YC
                                WHERE CM.DIRECTORS <> YC.PID
                                GROUP BY ACTORS),

                                ACTORS_MOVIE AS
                                (SELECT YM.ACTORS,
                                CASE WHEN YM.MOVIE_COUNT_YASH > IFNULL(OD.MAX_MOVIE_COUNT, 0)
                                FROM YASH_MOVIE_COUNT YM
                                LEFT OUTER JOIN OTHER_DIRECTORS OD ON YM.ACTORS = OD.ACTORS)

                                SELECT DISTINCT TRIM(P.NAME) NAME_OF_ACTORS
                                FROM PERSON P
                                WHERE TRIM(P.PID) IN (SELECT DISTINCT ACTORS
                                FROM ACTORS_MOVIE
                                WHERE MAX_YASH_MOVIE = 'YES')

                                """, con)


print("\nActors who worked more in movies directed by Yash:")
print('-'*50, '\n')
print(Actors_more_movie_Yash)
```

```
Actors who worked more in movies directed by Yash:
--------------------------------------------------

         NAME_OF_ACTORS
0       Waheeda Rehman
1       Achala Sachdev
2          Yash Chopra
3           Vinod Negi
4    Chandni Jas Keerat
..                 ...
100         Yasin Khan
101     Sandow S. Sethi
102              Naval
103          Prem Sood
104     Ramlal Shyamlal

[105 rows x 1 columns]
```

10) The Shahrukh number of an actor is the length of the shortest path between the actor and Shahru
Shahrukh Khan has Shahrukh number 0; all actors who acted in the same film as Shahrukh have Shah
same film as some actor with Shahrukh number 1 have Shahrukh number 2, etc. Return all actors who

```
Shahruk_2_Actors = pd.read_sql_query("""

                        WITH

                        SHAHRUK_0 AS
                        (SELECT TRIM(P.PID) PID
                        FROM PERSON P
                        WHERE TRIM(P.NAME) like '%Shahrukh%'),

                        SHAHRUK_1_MOV AS
                        (SELECT DISTINCT TRIM(MC.MID) MID, S_0.PID
                        FROM SHAHRUK_0 S_0, M_CAST MC
                        WHERE TRIM(MC.PID) = S_0.PID),

                        SHAHRUK_1_ACTS AS
                        (SELECT DISTINCT TRIM(MC.PID) PID
                        FROM M_CAST MC, SHAHRUK_1_MOV SM_1
                        WHERE TRIM(MC.MID) = SM_1.MID AND
                        TRIM(MC.PID) <> SM_1.PID),

                        SHAHRUK_2_MOV AS
                        (SELECT DISTINCT TRIM(MC.MID) MID, SA_1.PID
                        FROM SHAHRUK_1_ACTS SA_1, M_CAST MC
                        WHERE TRIM(MC.PID) = SA_1.PID)
                        SELECT DISTINCT TRIM(P.NAME) Shahruk_2_Actors
                        FROM PERSON P, M_CAST MC, SHAHRUK_2_MOV SM_2
                        WHERE TRIM(MC.PID) = TRIM(P.PID) AND
                        MC.MID = SM_2.MID AND
                        MC.PID <> SM_2.PID


                        """, con)


print("\nActors with Shahruk number 2:")
print('-'*50, '\n')
print(Shahruk_2_Actors)
```

```
Actors with Shahruk number 2:
--------------------------------------------------

            Shahruk_2_Actors
0               Freida Pinto
1       Caroline Christl Long
2              Rajeev Pahuja
3          Michelle Santiago
4             Jandre le Roux
...                      ...
15283           Dhruv Shetty
15284        Hayley Cleghorn
15285        Nirvasha Jithoo
15286         Kamal Maharshi
15287           Mohini Manik

[15288 rows x 1 columns]
```

Source Links

- https://www.appliedaicourse.com/
- https://www.w3schools.com/sql/default.asp
- https://www.w3resource.com/sql-exercises/movie-database-exercise/joins-exercises-on-movie