Branch: master

Find file Copy path

appliedai / Assignment_18_SQL_Assignment / sh.shankar1@gmail.com_ma_1.ipynb

shshankar1 initial commit

8961a7c on Oct 30

1 contributor

Raw Blame History

774 lines (774 sloc) 27.1 KB

In [1]:
```
import sqlite3
import logging

def fetch_connection(db_file):
    conn = None
    try:
        conn = sqlite3.connect(db_file)
    except:
        logging.error('Unable to create connection')

    return conn
```

In [2]:
```
!python --version
```

```
Python 3.6.8 :: Anaconda, Inc.
```

In [3]:
```
conn = fetch_connection('Db-IMDB.db')
```

1. 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.

In [6]:
```
import pandas as pd
result_1 = pd.read_sql_query('''select m.title, m.year, p.name
as \'Director\' from Person p
                                join M_Director md on md.pid =
p.pid
                                join Movie m on m.mid = md.mid
                                join M_Genre mg on mg.mid = m.
mid
                                join Genre g on g.gid = mg.gid
                                where g.name like '%Comedy%'
                                and m.year % 4 = 0
                                ''', conn)
```

In [7]:
```
print(result_1)
```

```
                                 title  year              Dir
ector
0                           Mastizaade  2016           Milap Z
averi
1                           Mastizaade  2016           Milap Z
averi
2     Harold & Kumar Go to White Castle  2004           Danny L
einer
3     Harold & Kumar Go to White Castle  2004           Danny L
einer
4                    Gangs of Wasseypur  2012          Anurag Ka
shyap
..                                 ...   ...
...
383                        Let's Enjoy  2004  Siddharth Anand
Kumar
384                            Sathyam  2008         Amma Rajas
```

```
ekhar
385                        Tandoori Love  2008          Oliver P
aulus
386                          Le Halua Le  2012             Raja C
handa
387                   Raja Aur Rangeeli  1996       K.S. Prakas
h Rao

[388 rows x 3 columns]
```

2. List the names of all the actors who played in the movie 'Anand' (1971)

In [8]:
```
result_2 = pd.read_sql_query('''
                            select p.name from Person p
                            join M_Cast mc on p.pid = trim(mc.
pid)
                            join Movie m on m.mid = mc.mid
                            where m.title=\'Anand\' and m.year
= 1971
                            ''', conn)
```

In [9]:
```
print(result_2)
```

```
                 Name
0    Amitabh Bachchan
1       Rajesh Khanna
2      Brahm Bhardwaj
3          Ramesh Deo
4           Seema Deo
5          Dev Kishan
6         Durga Khote
7       Lalita Kumari
8        Lalita Pawar
9        Atam Prakash
10      Sumita Sanyal
11     Asit Kumar Sen
12         Dara Singh
13      Johnny Walker
14          Moolchand
15        Gurnam Singh
16             Savita
```

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

In [10]:
```
import pandas as pd

result_3 = pd.read_sql_query('''
                            select p.name from Person p
                            join M_Cast mc on p.pid = trim(mc.
pid)
                            join Movie m on m.mid = mc.mid
                            where m.year < 1970 or m.year > 19
90
                            ''', conn)
```

In [11]:
```
print(result_3)
```

```
                   Name
0       Christian Bale
1       Cate Blanchett
2        John Benfield
3          Lorna Brown
4      Patrick Godfrey
...                ...
70523        Alok Nath
70524      Yunus Parvez
70525       Asha Sharma
70526      Ajay Nagrath
70527        Arun Govil

[70528 rows x 1 columns]
```

4. 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 [12]:
```
result_4 = pd.read_sql_query('''
                                select p.name, vw.movie_count
                                from Person p
                                join
                                (
                                select md.pid, count(*) as mov
ie_count
                                from M_Director md
                                group by md.pid
                                having count(*) > 10
                                )vw on p.pid = vw.pid
                            ''', conn)
```

In [13]:
```
print(result_4)
```

```
                       Name  movie_count
0          Mahesh Manjrekar           15
1            Satish Kaushik           12
2            Anurag Kashyap           13
3               Yash Chopra           21
4              Subhash Ghai           18
..                      ...          ...
83              Umesh Mehra           12
84  Ananth Narayan Mahadevan           13
85        K. Raghavendra Rao           13
86          Govind Nihalani           11
87            Nasir Hussain           11

[88 rows x 2 columns]
```

5.

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

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 [5]:
```
import pandas as pd

result_5a = pd.read_sql_query('''
                                select m.year, count(*) as mov
ie_count from Movie m
                                join
                                (
                                    select distinct mid from M
_Cast
                                    where mid not in
                                    (
                                        select mc.mid from M_C
ast mc
                                        join Person p on p.pid
= trim(mc.pid)
                                        where p.gender = \'Mal
e\'
                                    )
                                )vw on vw.mid = m.mid
                                group by m.year

                            ''', conn)
```

In [6]:
```
print(result_5a)
```

```
   year  movie_count
0  1939            1
1  1999            1
2  2000            1
3  2009            1
4  2012            1
5  2018            2
```

In [7]:
```
import pandas as pd

result_5b = pd.read_sql_query(
                            '''
                            select m.year, count(m.mid) as tot
al_movie_count,
                            i_vw.female_only_cast_movie_count,
                            (i_vw.female_only_cast_movie_count
*100/(count(m.mid)*1.0)) as percentage_female_only_cast_movie
                            from Movie m
                            left join
                            (
                                select m.year, count(*) as fem
ale_only_cast_movie_count from Movie m
                                join
                                (
                                    select distinct mid from M
_Cast
                                    where mid not in
                                    (
                                        select mc.mid from M_C
ast mc
                                        join Person p on p.pid
= trim(mc.pid)
                                        where p.gender = \'Mal
e\'
                                    )
                                )vw on vw.mid = m.mid
                                group by m.year
                            )i_vw on m.year = i_vw.year
                            group by m.year
                            ''', conn)
```

In [10]:
```
result_5b.head(100)
```

Out[10]:

	year	total_movie_count	female_only_cast_movie_count	percentag
0	1931	1	NaN	NaN
1	1936	3	NaN	NaN
2	1939	2	1.0	50.000000
3	1941	1	NaN	NaN
4	1943	1	NaN	NaN
...	...	...	...	...
73	2014	126	NaN	NaN
74	2015	119	NaN	NaN
75	2016	129	NaN	NaN
76	2017	126	NaN	NaN
77	2018	104	2.0	1.923077

78 rows × 4 columns

6. 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 [18]:
```
result_6 = pd.read_sql_query('''
                            select vw.mid, m.title, max(vw.cas
t_count) as cast_size
                            from
                            (
                                select count(*) as cast_count,
mid
                                from M_Cast group by mid
```

```
                              )vw
                              join Movie m on m.mid = vw.mid

                              ''', conn)
print(result_6)
```

```
         mid         title  cast_size
0  tt5164214  Ocean's Eight        238
```

7. A decade is a sequence of 10 consecutive years. For example, say in your database you have movie information starting from 1965. Then the first decade is 1965, 1966, ..., 1974; the second one is 1967, 1968, ..., 1976 and so on. Find the decade D with the largest number of films and the total number of films in D.

In [38]:
```
result_7 = pd.read_sql_query('''
                                select Decade, max(movie_count
s)
                                from
                                (
                                    select Decade, count(*) as
movie_counts
                                    from
                                    (
                                        select m.year, vw.min_
year, (((m.year-vw.min_year)/10)+1) as Decade from Movie m
                                        join
                                        (
                                            select min(year) a
s min_year from Movie
                                        )vw on 1=1
                                    )i_vw
                                    group by Decade
                                )o_vw
                            ''', conn)
print(result_7)

#verifying results
movie_decade_1991_2000 = pd.read_sql_query('''
                                            select * from Movi
e where year >= \'2001\' and year <= \'2010\'
                                        ''', conn)
print(len(movie_decade_1991_2000))
```

```
   Decade  max(movie_counts)
0       8               1047
1047
```

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

In [39]:
```
import pandas as pd

result_8 = pd.read_sql_query('''
                              select vw.*, (vw.next_year - vw.ye
ar) as gap
                            from
                            (
                                select i_vw.pid, i_vw.name, i_
vw.title, i_vw.year,
                                    LEAD(i_vw.year, 1, 0) OVER
(PARTITION BY i_vw.name ORDER BY i_vw.year ASC) AS next_year
                                    from
                                    (
                                        (
                                            select distinct tr
im(pid) as pid, trim(name) as name from Person
                                        )p
                                        join
                                        (
                                            select distinct tr
im(mid) as mid, trim(pid) as pid from M_Cast
                                        )mc on p.pid = mc.pid
                                        join
                                        (
                                            select trim(mid) a
s mid, trim(title) as title, trim(year) as year from Movie
                                        )m on m.mid = mc.mid
                                    )i_vw
                            )vw
                            where vw.next_year > 0 and (vw.nex
t_year - vw.year) < 3
                        ''', conn)

print(result_8)
```

```
             pid             name                   title  y
ear next_year  \
0      nm1869655  A. Abdul Hameed              Prem Nagar  1
974      1975
1      nm0359845      A.K. Hangal            Teesri Kasam  1
966      1967
2      nm0359845      A.K. Hangal                 Shagird  1
967      1969
3      nm0359845      A.K. Hangal         Saat Hindustani  1
969      1971
4      nm0359845      A.K. Hangal                   Guddi  1
971      1971
...          ...              ...                     ...
...       ...
42638  nm0892606      Zul Vellani    Charas: A Joint Effort  2
004      2005
42639  nm1302631      Zulfi Sayed                  Pyaasa  2
002      2003
42640  nm1302631      Zulfi Sayed              Chupke Se  2
003      2004
42641  nm1302631      Zulfi Sayed  Wajahh: A Reason to Kill  2
004      2005
42642  nm1302631      Zulfi Sayed             Desh Drohi  2
```