

SQL Assignment

```
In [1]: import pandas as pd
import sqlite3

from IPython.display import display, HTML
```

```
In [2]: # Note that this is not the same db we have used in course videos, please download
# https://drive.google.com/file/d/10-1-L1DdNxEK606nG2jS31MbrMh-OnXM/view?usp=share_link
```

```
In [3]: cd "D:\AppliedAI\Applied_ML_Course_Assignments\18_SQL\"

D:\AppliedAI\Applied_ML_Course_Assignments\18_SQL
```

```
In [4]: conn = sqlite3.connect("./Db-IMDB-Assignment.db")
```

Overview of all tables

```
In [5]: tables = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM sqlite_master WHERE
tables = tables["Table_Name"].values.tolist()
print(tables)

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

```
In [6]: 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

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(*)

```
In [33]: from IPython.display import Image # displaying image
Image(filename = "./db_schema.jpeg") # https://stackoverflow
```

Out[33]:

IMDB database schema

Data Tables

Movie	Person	Genre	Language	Country	Location
MID (Primary)	PID (Primary)	GID (Primary)	LAID (Primary)	CID (Primary)	LID (Primary)
title	Name	Name	Name	Name	Name
year	DOB				
rating	Gender				
num_votes					

Mapping Tables (containing foreign keys)

M_Producer	M_Director	M_Cast	M_Genre	M_Language	M_Country	M_Location
ID (Primary)	ID (Primary)	ID (Primary)	ID (Primary)	ID (Primary)	ID (Primary)	ID (Primary)
MID	MID	MID	MID	MID	MID	MID
PID	PID	PID	GID	LAID	CID	LID

In []:

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 [7]: disp=pd.read_sql_query("SELECT * FROM Movie",conn)
disp.describe()
print(disp.shape)
print(disp)
```

```
(3473, 6)
   index  MID      title  year  rating  num_votes
0      0  tt2388771    Mowgli  2018     6.6     21967
1      1  tt5164214  Ocean's Eight  2018     6.2     110861
2      2  tt1365519   Tomb Raider  2018     6.4     142585
3      3  tt0848228   The Avengers  2012     8.1    1137529
4      4  tt8239946    Tumbbad  2018     8.5       7483
...     ...     ...     ...     ...     ...
3468  3470  tt0090611  Allah-Rakha  1986     6.2        96
3469  3471  tt0106270      Anari  1993     4.7       301
3470  3472  tt0852989  Come December  2006     5.7        57
3471  3473  tt0375882    Kala Jigar  1939     3.3       174
3472  3474  tt0375890     Kanoon  1994     3.2       103
```

[3473 rows x 6 columns]

```
In [8]: # Lets look at genre table
disp_Genre=pd.read_sql_query("SELECT * FROM Genre",conn)
disp_Genre.describe()
print(disp_Genre.shape)
print(disp_Genre)
```

```
(328, 3)
   index  Name  GID
0      0  Adventure, Drama, Fantasy  0
1      1  Action, Comedy, Crime  1
2      2  Action, Adventure, Fantasy  2
3      3  Action, Adventure, Sci-Fi  3
4      4  Drama, Horror, Thriller  4
..     ..
323  323  Animation, Adventure, Fantasy  323
324  324  Biography, Drama, War  324
325  325  Animation, Drama, Adventure  325
326  326  Drama, Action  326
327  327  Drama, Mystery, Sci-Fi  327
```

[328 rows x 3 columns]

```
In [9]: # Since .describe methods displays only the columns with integers, we can say the
# Lets explore year column
```

```
In [10]: movie_year=disp["year"]
print(type(movie_year))
print(movie_year.describe())
print(set(movie_year))
```

```
<class 'pandas.core.series.Series'>
count      3473
unique      125
top         2005
freq        128
Name: year, dtype: object
{'2001', '1970', '2018', 'I 1992', 'I 1969', 'I 2015', '1936', '1971', '2002',
'XVII 2016', 'IV 2010', '1991', '1992', '1977', 'III 2007', '1982', 'I 1968',
'III 2017', '1941', '1966', 'III 2015', '1987', 'I 2017', 'II 2009', '1974', '1
989', '1931', '2005', '1953', '2008', 'II 2013', '1950', 'I 1986', '1959', 'I 2
009', '2014', 'II 1983', '1983', '1986', '1946', 'I 1989', 'I 2007', '1963', '1
957', '1969', '1956', 'I 2005', '1980', '1993', '1978', 'VI 2015', '1973', 'I 1
996', '1948', '2012', '1943', '1939', 'II 2008', '2015', 'I 2011', '2011', '197
2', 'I 2003', '2007', '1985', '1964', 'I 2013', '1997', 'II 2018', 'V 2015', 'I
II 2016', 'I 2006', 'II 1998', 'I 2008', '1968', '1952', '1976', 'IV 2017', 'I
1980', '1995', '2009', '2017', '2000', '1965', '2016', 'I 2012', '1951', '195
4', '1958', 'IV 2011', '1955', '1960', '1996', '1975', 'I 2002', 'II 2011', 'I
1983', '1947', '1990', '1998', '2003', 'I 2018', 'I 1964', '1988', '1979', '201
3', 'II 2017', '1981', '2010', '1961', '1994', '2004', 'I 1997', '2006', '194
9', 'II 2010', '1962', 'II 2012', 'I 2010', 'I 2014', 'I 2016', '1967', 'I 200
1', '1999', '1984'}
```

```

In [11]: cursor = conn.cursor()
cursor.execute('UPDATE Movie SET year = REPLACE(year, "I", "");')
cursor.execute('UPDATE Movie SET year = REPLACE(year, "V", "");')
cursor.execute('UPDATE Movie SET year = REPLACE(year, "X ", "");')
cursor.execute('UPDATE Movie SET title = LTRIM(title);')
cursor.execute('UPDATE Movie SET year = RTRIM(LTRIM(year));')
cursor.execute('UPDATE Movie SET rating = RTRIM(LTRIM(rating));')
cursor.execute('UPDATE Movie SET num_votes = RTRIM(LTRIM(num_votes));')

cursor.execute('UPDATE M_Producer SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE M_Producer SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE M_Director SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE M_Director SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE M_Cast SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE M_Cast SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE M_Genre SET gid = RTRIM(LTRIM(gid));')
cursor.execute('UPDATE M_Genre SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE Genre SET gid = RTRIM(LTRIM(gid));')
cursor.execute('UPDATE Genre SET name = RTRIM(LTRIM(name));')

cursor.execute('UPDATE Person SET name = RTRIM(LTRIM(name));')
cursor.execute('UPDATE Person SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE Person SET gender = RTRIM(LTRIM(gender));')

#####
X=pd.read_sql_query(""" SELECT * from movie ORDER BY year DESC limit 5""", conn)

```

```

%%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 * from Movie """ grader_1(query1)

```

```
In [12]: # Main answer # T2.title as movie, T2.year as year, T3.name as Genre
Q1=pd.read_sql_query("SELECT T1.name, T5.year, T5.title as Director FROM Person
INNER JOIN M_Director AS T2 on T1.PID = T2.PID \
INNER JOIN M_Genre AS T3 ON T2.ID=T3.ID \
INNER JOIN Genre As T4 on T3.GID=T4.GID \
INNER JOIN Movie AS T5 on T2.MID = T5.MID \
WHERE T4.name LIKE '%comedy%' AND \
(T5.year%4=0 AND T5.year%100 != 0) OR (T5.year%4=0 AND T5.ye
", conn)

print(Q1)
##Q1.sort_values("year")
##Q1[~Q1.duplicated()]
```

	Name	year	Director
0	Griffin Dunne	2008	The Accidental Husband
1	Mahesh Manjrekar	2000	Kurukshetra
2	Mahesh Manjrekar	2000	Astitva
3	Mahesh Manjrekar	2000	Jis Desh Mein Ganga Rehta Hain
4	Madonna	2008	Filth and Wisdom
..
275	Tirupati Swamy	2000	Azaad
276	Shankaraiya	2012	Khokababu
277	Amma Rajasekhar	2008	Sathyam
278	Oliver Paulus	2008	Tandoori Love
279	Raja Chanda	2012	Le Halua Le

[280 rows x 3 columns]

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

```
In [13]: %%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 T3.name FROM Movie as T1 \
              INNER JOIN M_Cast AS T2 ON T2.MID = T1.MID \
              INNER JOIN Person AS T3 ON T2.PID = T3.PID \
              WHERE T1.title LIKE '%Anand%' AND T1.year = 1971"""

grader_2(query2)
```

```
      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
Wall time: 301 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 [14]: %%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 T1.PID FROM Person AS T1 \
INNER JOIN M_Cast AS T2 ON T1.PID = T2.PID \
INNER JOIN Movie AS T3 ON T3.MID = T2.MID \
WHERE T3.year < 1970 """

query_more_1990 = """
SELECT T1.PID FROM Person AS T1 \
INNER JOIN M_Cast AS T2 ON T1.PID = T2.PID \
INNER JOIN Movie AS T3 ON T3.MID = T2.MID \
WHERE T3.year > 1990 """

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: 841 ms
```

```
%%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 = """ * Write your query for the question 3 * """ grader_3(query3)
```

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.

```
%%time
```

```
def grader_4a(query_4a): query_4a = pd.read_sql_query(query_4a, conn)
print(query_4a.head(10)) print(query_4a.shape) return (query_4a.shape == (1462,2))
```

```
query_4a = """SELECT Distinct T1.Name AS Directors, count(T3.MID) AS Films FROM Person T1
Inner Join M_director AS T2
ON T1.PID = T2.PID
Inner Join Movie AS T3
ON T2.MID = T3.MID
```

```
GROUP BY T1.Name
HAVING COUNT(T3.MID) >= 10
ORDER BY Films DESC""" print(grader_4a(query_4a))
```

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

```
In [15]: %%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 T1.Name AS Directors, count(T3.MID) AS Films FROM Per
            Inner Join M_director AS T2 \
            ON T1.PID =T2.PID \
            Inner Join Movie AS T3 \
            ON T2.MID = T3.MID \
            GROUP BY T1.Name \
            HAVING COUNT(T3.MID) >=10 \
            ORDER BY Films DESC"""

grader_4(query4)
```

	Directors	Films
0	David Dhawan	39
1	Mahesh Bhatt	36
2	Ram Gopal Varma	30
3	Priyadarshan	30
4	Vikram Bhatt	29
5	Hrishikesh Mukherjee	27
6	Yash Chopra	21
7	Shakti Samanta	19
8	Basu Chatterjee	19
9	Subhash Ghai	18

Wall time: 107 ms

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

In [50]: %%time

```
#####
cursor = conn.cursor()
cursor.execute('UPDATE Movie SET year = REPLACE(year, "I", "");')
cursor.execute('UPDATE Movie SET year = REPLACE(year, "V", "");')
cursor.execute('UPDATE Movie SET year = REPLACE(year, "X ", "");')
cursor.execute('UPDATE Movie SET title = LTRIM(title);')
cursor.execute('UPDATE Movie SET year = RTRIM(LTRIM(year));')
cursor.execute('UPDATE Movie SET rating = RTRIM(LTRIM(rating));')
cursor.execute('UPDATE Movie SET num_votes = RTRIM(LTRIM(num_votes));')

cursor.execute('UPDATE M_Producer SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE M_Producer SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE M_Director SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE M_Director SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE M_Cast SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE M_Cast SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE M_Genre SET gid = RTRIM(LTRIM(gid));')
cursor.execute('UPDATE M_Genre SET mid = RTRIM(LTRIM(mid));')

cursor.execute('UPDATE Genre SET gid = RTRIM(LTRIM(gid));')
cursor.execute('UPDATE Genre SET name = RTRIM(LTRIM(name));')

cursor.execute('UPDATE Person SET name = RTRIM(LTRIM(name));')
cursor.execute('UPDATE Person SET pid = RTRIM(LTRIM(pid));')
cursor.execute('UPDATE Person SET gender = RTRIM(LTRIM(gender));')
#####

# 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))
    print(query_5aa.shape)
    return (query_5aa.shape == (8846, 3))

query_5aa = """ SELECT T1.year, count(T1.Title) FROM Movie as T1 \
                INNER JOIN M_Cast AS T2 ON T2.MID = T1.MID \
                INNER JOIN Person AS T3 ON T3.PID = T2.PID \
                WHERE T3.Gender = 'Female' \
                Group BY T1.year \
                """

print(grader_5aa(query_5aa))

def grader_5ab(query_5ab):
    query_5ab = pd.read_sql_query(query_5ab, conn)
    print(query_5ab.head(10))
    print(query_5ab.shape)
    return (query_5ab.shape == (3469, 3))

#query_5ab = """ *** Write your query that will have at least one male actor try t
```

```
query_5ab = """ SELECT T1.Title, count(T1.Title) FROM Movie as T1 \
                INNER JOIN M_Cast AS T2 ON T2.MID = T1.MID \
                INNER JOIN Person AS T3 ON T3.PID = T2.PID \
                WHERE T3.Gender LIKE 'M%' \
                Group BY T1.year \
                """
```

```
print(grader_5ab(query_5ab))
```

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

```

    year  count(T1.Title)
0  1931              3
1  1936             19
2  1939             18
3  1941              7
4  1943              3
5  1946              6
6  1947             11
7  1948             10
8  1949             15
9  1950             13
(78, 2)
False

          title  count(T1.Title)
0          Alam Ara              5
1          Devdas             24
2      The Little Princess             27
3      Footsteps in the Dark             46
4          Kismet              11
5  Dr. Kotnis Ki Amar Kahani             12
6          Jugnu              9
7          Mela              25
8          Andaz              27
9          Jogan              28
(78, 2)
False
Wall time: 977 ms
```

```
In [37]: X = pd.read_sql_query("SELECT * FROM Person LIMIT 5;", conn)
X
```

Out[37]:

	index	PID	Name	Gender
0	0	nm0000288	Christian Bale	Male
1	1	nm0000949	Cate Blanchett	Female
2	2	nm1212722	Benedict Cumberbatch	Male
3	3	nm0365140	Naomie Harris	Female
4	4	nm0785227	Andy Serkis	Male

```
In [17]: %%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 = """ *** Write your query for the question 5a *** """
grader_5a(query5a)
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_5a(q5a)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=args[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceback(exc, traceback)
    44     if traceback == Ellipsis:
    45         _, _, traceback = sys.exc_info()
----> 46     raise exc.with_traceback(traceback)
```

47
48

```
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
1593         cur.execute(*args, **kwargs)
1594     else:
-> 1595         cur.execute(*args)
1596     return cur
1597 except Exception as exc:
```

DatabaseError: Execution failed on sql ' *** Write your query for the question
5a *** ': near "": syntax error

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 [18]: %%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 = """ *** Write your query for the question 5b *** """
grader_5b(query5b)
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwar
gs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_5b(q5b)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, in
dex_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index
_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwar
gs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=a
rgs[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceba
ck(exc, traceback)
    44         if traceback == Ellipsis:
    45             _, _, traceback = sys.exc_info()
---> 46         raise exc.with_traceback(traceback)
```

```
47
48

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
1593         cur.execute(*args, **kwargs)
1594     else:
-> 1595         cur.execute(*args)
1596     return cur
1597 except Exception as exc:
```

DatabaseError: Execution failed on sql ' *** Write your query for the question 5b *** ': near "(": syntax error

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 [19]: %%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 = """ *** Write your query for the question 5b *** """
grader_6(query6)
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594         else:
-> 1595             cur.execute(*args)
    1596         return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_6(q6)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, in
dex_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index
_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwar
gs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=args[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceba
ck(exc, traceback)
    44         if traceback == Ellipsis:
    45             _, _, traceback = sys.exc_info()
----> 46         raise exc.with_traceback(traceback)
```

47
48

```
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
1593         cur.execute(*args, **kwargs)
1594     else:
-> 1595         cur.execute(*args)
1596     return cur
1597 except Exception as exc:
```

DatabaseError: Execution failed on sql ' *** Write your query for the question 5b *** ': near "***": syntax error

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 [20]: %%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 = """ *** Write a query that computes number of movies in each year ***
grader_7a(query7a)

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

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_7a(q7a)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=args[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceback(exc, traceback)
    44     if traceback == Ellipsis:
```

```
45         _, _, traceback = sys.exc_info()
---> 46     raise exc.with_traceback(traceback)
47
48

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
1593         cur.execute(*args, **kwargs)
1594     else:
-> 1595         cur.execute(*args)
1596     return cur
1597 except Exception as exc:
```

DatabaseError: Execution failed on sql ' *** Write a query that computes number of movies in each year *** ': near "***": syntax error

```
In [21]: %%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 = """
***
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
***
"""

grader_7b(query7b)
# if you see the below results the first movie year is less than 2nd movie year
# 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
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594         else:
-> 1595             cur.execute(*args)
    1596         return cur
```

OperationalError: near "(": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_7b(q7b)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, in
dex_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index
_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwa
rgs)
    1608         "Execution failed on sql '{sql}': {exc}".format(sql=a
rgs[0], exc=exc)
    1609         )
```

```

-> 1610             raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceback(exc, traceback)
    44         if traceback == Ellipsis:
    45             _, _ , traceback = sys.exc_info()
--> 46         raise exc.with_traceback(traceback)
    47
    48

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1593             cur.execute(*args, **kwargs)
    1594         else:
-> 1595             cur.execute(*args)
    1596         return cur
    1597     except Exception as exc:

DatabaseError: Execution failed on sql '
***
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
***
': near "*": syntax error

```

```
In [22]: %%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 = """ *** Write a query that will return the decade that has maximum number
grader_7(query7)
# if you check the output we are printinng all the year in that decade, its fine
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_7(q7)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=args[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceback(exc, traceback)
    44     if traceback == Ellipsis:
    45         _, _, traceback = sys.exc_info()
```

```
---> 46         raise exc.with_traceback(traceback)
      47
      48

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
      1593             cur.execute(*args, **kwargs)
      1594         else:
-> 1595             cur.execute(*args)
      1596         return cur
      1597     except Exception as exc:
```

DatabaseError: Execution failed on sql ' *** Write a query that will return the decade that has maximum number of movies ***': near "": syntax error

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


```
In [23]: %%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 = """ *** Write a query that will results in number of movies actor-directed
grader_8a(query8a)

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

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_8a(q8a)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=args[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceback(exc, traceback)
    44     if traceback == Ellipsis:
```

```
45         _, _, traceback = sys.exc_info()
---> 46     raise exc.with_traceback(traceback)
47
48

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
1593         cur.execute(*args, **kwargs)
1594     else:
-> 1595         cur.execute(*args)
1596     return cur
1597 except Exception as exc:
```

DatabaseError: Execution failed on sql ' *** Write a query that will results in number of movies actor-director worked together ***': near "***": syntax error

In [24]: %%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 = """ *** Write a query that answers the 8th question *** """
grader_8(query8)
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_8(q8)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=args[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceback(exc, traceback)
    44     if traceback == Ellipsis:
```

```
45         _, _, traceback = sys.exc_info()
--> 46     raise exc.with_traceback(traceback)
47
48

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
1593         cur.execute(*args, **kwargs)
1594     else:
-> 1595         cur.execute(*args)
1596     return cur
1597 except Exception as exc:

DatabaseError: Execution failed on sql ' *** Write a query that answers the 8
th question ***': near "": syntax error
```

Q9 --- The Shahrukh number of an actor is the length of the shortest path between the actor and Shahrukh Khan in the "co-acting" graph. That is, Shahrukh Khan has Shahrukh number 0; all actors who acted in the same film as Shahrukh have Shahrukh number 1; all actors who acted in the same film as some actor with Shahrukh number 1 have Shahrukh number 2, etc. Return all actors whose Shahrukh number is 2.

```
In [25]: %%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 = """ *** Write a query that answers the 9th question ***"""
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
# removing S1 actors from the combined list of S1 & S2 actors, so that we get only
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_9a(q9a)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, index_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
-> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=args[0], exc=exc)
-> 1609         )
    1610         raise_with_traceback(ex)
    1611
```

```
1612     @staticmethod
```

```
~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceback  
ck(exc, traceback)
```

```
    44     if traceback == Ellipsis:  
    45         _, _, traceback = sys.exc_info()  
--> 46     raise exc.with_traceback(traceback)  
    47  
    48
```

```
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)  
gs)
```

```
    1593         cur.execute(*args, **kwargs)  
    1594     else:  
-> 1595         cur.execute(*args)  
    1596     return cur  
    1597     except Exception as exc:
```

DatabaseError: Execution failed on sql ' *** Write a query that answers the 9
th question ***': near "***": syntax error

```
In [26]: %%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 = """ *** Write a query that answers the 9th question ***"""
grader_9(query9)
```

```
-----
OperationalError                                Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwar
gs)
    1594             else:
-> 1595                 cur.execute(*args)
    1596             return cur
```

OperationalError: near "***": syntax error

During handling of the above exception, another exception occurred:

```
DatabaseError                                Traceback (most recent call last)
<timed exec> in <module>

<timed exec> in grader_9(q9)

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_sql_query(sql, con, in
dex_col, coerce_float, params, parse_dates, chunksize)
    330         coerce_float=coerce_float,
    331         parse_dates=parse_dates,
--> 332         chunksize=chunksize,
    333     )
    334

~\Anaconda3\lib\site-packages\pandas\io\sql.py in read_query(self, sql, index
_col, coerce_float, params, parse_dates, chunksize)
    1643
    1644         args = _convert_params(sql, params)
-> 1645         cursor = self.execute(*args)
    1646         columns = [col_desc[0] for col_desc in cursor.description]
    1647

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwar
gs)
    1608             "Execution failed on sql '{sql}': {exc}".format(sql=a
rgs[0], exc=exc)
    1609         )
-> 1610         raise_with_traceback(ex)
    1611
    1612     @staticmethod

~\Anaconda3\lib\site-packages\pandas\compat\__init__.py in raise_with_traceba
ck(exc, traceback)
    44         if traceback == Ellipsis:
    45             _, _, traceback = sys.exc_info()
```

```

---> 46         raise exc.with_traceback(traceback)
      47
      48

~\Anaconda3\lib\site-packages\pandas\io\sql.py in execute(self, *args, **kwargs)
      1593             cur.execute(*args, **kwargs)
      1594         else:
-> 1595             cur.execute(*args)
      1596         return cur
      1597     except Exception as exc:

```

DatabaseError: Execution failed on sql ' *** Write a query that answers the 9th question ***': near "***": syntax error

```
In [27]: QJ= pd.read_sql_query("SELECT COUNT(*) FROM Movie",conn)
        QJ
```

Out[27]:

	COUNT(*)
0	3473

In []:

In []: