Prepared by Abhishek Kumar

https://www.linkedin.com/in/abhishekkumar-0311/

Writing SQL query on a dataframe using pandassql

In [1]:
```python
# To get multiple outputs in the same cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

%matplotlib inline
```

In [2]:
```python
#!pip install pandasql

import pandas as pd
import numpy as np
import pandasql as ps
from pandasql import sqldf
import sqlite3
from sqlite3 import Error

df = pd.read_csv('E:\VCS\GitHub\Machine-Learning-with-Python\data\movie.csv')

df.head()
```

Out[2]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gros
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	Na

5 rows × 28 columns

```
In [3]: #%%timeit
        pysqldf = lambda q: sqldf(q, globals())

        q1 = "Select * from df where director_name = 'James Cameron'"
        pysqldf(q1)

        q2 = "Select director_name , sum(num_critic_for_reviews) as tot_critic from df group by director_name order by tot_critic desc"
        pysqldf(q2)#.sort_values(by=)
        pysqldf(q2).sort_values(by='tot_critic', ascending=True)
```

Out[3]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gros
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.(
1	Color	James Cameron	315.0	194.0	0.0	794.0	Kate Winslet	29000.0	658672302.(
2	Color	James Cameron	210.0	153.0	0.0	539.0	Jenette Goldstein	780.0	204843350.(
3	Color	James Cameron	94.0	141.0	0.0	618.0	Tia Carrere	2000.0	146282411.(
4	Color	James Cameron	82.0	171.0	0.0	638.0	Todd Graff	2000.0	54222000.(
5	Color	James Cameron	250.0	154.0	0.0	604.0	Carrie Henn	2000.0	85200000.(
6	Color	James Cameron	204.0	107.0	0.0	255.0	Brian Thompson	2000.0	38400000.(

7 rows × 28 columns

Out[3]:

	director_name	tot_critic
0	Steven Spielberg	6582.0
1	Ridley Scott	4616.0

	director_name	tot_critic
2	Martin Scorsese	4285.0
3	Clint Eastwood	4244.0
4	Christopher Nolan	4090.0
...	...	...
2393	Cary Bell	NaN
2394	Brandon Landers	NaN
2395	Anthony Vallone	NaN
2396	Amal Al-Agroobi	NaN
2397	Al Franklin	NaN

2398 rows × 2 columns

Out[3]:

	director_name	tot_critic
2357	Alan Jacobs	1.0
2330	Tom Sanchez	1.0
2331	Timothy Hines	1.0
2332	Shekar	1.0
2333	Scott Smith	1.0
...	...	...
2393	Cary Bell	NaN
2394	Brandon Landers	NaN
2395	Anthony Vallone	NaN
2396	Amal Al-Agroobi	NaN
2397	Al Franklin	NaN

2398 rows × 2 columns

https://www.kdnuggets.com/2017/02/python-speak-sql-pandasql.html

SQLite Python

https://datatofish.com/create-database-python-using-sqlite3/#:~:text=Import%20the%20CSV%20files%20using,file%20using%20the%20to_csv%20command

In [4]:
```
import sqlite3

conn = sqlite3.connect('TestDB.db')  # You can create a new database by changing the name within the quotes
c = conn.cursor() # The database will be saved in the location where your 'py' file is saved

# Create table - CLIENTS
c.execute('''CREATE TABLE CLIENTS
             ([generated_id] INTEGER PRIMARY KEY,[Client_Name] text, [Country_ID] integer, [Date] date)''')

# Create table - COUNTRY
c.execute('''CREATE TABLE COUNTRY
             ([generated_id] INTEGER PRIMARY KEY,[Country_ID] integer, [Country_Name] text)''')

# Create table - DAILY_STATUS
c.execute('''CREATE TABLE DAILY_STATUS
             ([Client_Name] text, [Country_Name] text, [Date] date)''')

conn.commit()

# Note that the syntax to create new tables should only be used once in the code (unless you dropped the table/s at the end of the
# The [generated_id] column is used to set an auto-increment ID for each record
# When creating a new table, you can add both the field names as well as the field formats (e.g., Text)
```

Out[4]:
```
<sqlite3.Cursor at 0x1739d8df3b0>
```

Out[4]:
```
<sqlite3.Cursor at 0x1739d8df3b0>
```

Out[4]:
```
<sqlite3.Cursor at 0x1739d8df3b0>
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:

```

```

In [5]:

```
import sqlite3
import pandas as pd
from pandas import DataFrame

conn = sqlite3.connect('movie.db')
c = conn.cursor()

movie = pd.read_csv (r'E:\VCS\GitHub\Machine-Learning-with-Python\data\movie.csv')
movie.to_sql('MOVIE', conn, if_exists='append', index = False) # Insert the values from the csv file into the table 'CLIENTS'

read_country = pd.read_csv (r'C:\Users\Ron\Desktop\Client\Country_14-JAN-2019.csv')
read_country.to_sql('COUNTRY', conn, if_exists='replace', index = False) # Replace the values from the csv file into the table 'CO

# When reading the csv:
# - Place 'r' before the path string to read any special characters, such as '\'
# - Don't forget to put the file name at the end of the path + '.csv'
# - Before running the code, make sure that the column names in the CSV files match with the column names in the tables created an
# - If needed make sure that all the columns are in a TEXT format

c.execute('''
INSERT INTO DAILY_STATUS (Client_Name,Country_Name,Date)
SELECT DISTINCT clt.Client_Name, ctr.Country_Name, clt.Date
FROM CLIENTS clt
LEFT JOIN COUNTRY ctr ON clt.Country_ID = ctr.Country_ID
          ''')

c.execute('''
SELECT DISTINCT *
FROM DAILY_STATUS
WHERE Date = (SELECT max(Date) FROM DAILY_STATUS)
          ''')

#print(c.fetchall())

df = DataFrame(c.fetchall(), columns=['Client_Name','Country_Name','Date'])
print (df) # To display the results after an insert query, you'll need to add this type of syntax above: 'c.execute(''' SELECT * f

df.to_sql('DAILY_STATUS', conn, if_exists='append', index = False) # Insert the values from the INSERT QUERY into the table 'DAILY

# export_csv = df.to_csv (r'C:\Users\Ron\Desktop\Client\export_list.csv', index = None, header=True) # Uncomment this syntax if yo
# Don't forget to add '.csv' at the end of the path (as well as r at the beg to address special characters)
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-5-6285767eb693> in <module>
      9 movie.to_sql('MOVIE', conn, if_exists='append', index = False) # Insert the values from the csv file into the table 'CLIEN
TS'
     10 
---> 11 read_country = pd.read_csv (r'C:\Users\Ron\Desktop\Client\Country_14-JAN-2019.csv')
     12 read_country.to_sql('COUNTRY', conn, if_exists='replace', index = False) # Replace the values from the csv file into the t
able 'COUNTRY'
     13 

~\anaconda3\lib\site-packages\pandas\io\parsers.py in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, useco
ls, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfoote
r, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, dat
e_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doubleq
uote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad_lines, delim_whitespace, low_memory, memory_map, float_pre
cision)
    684     )
    685 
--> 686     return _read(filepath_or_buffer, kwds)
    687 
    688 

~\anaconda3\lib\site-packages\pandas\io\parsers.py in _read(filepath_or_buffer, kwds)
    450 
    451     # Create the parser.
--> 452     parser = TextFileReader(fp_or_buf, **kwds)
    453 
    454     if chunksize or iterator:

~\anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(self, f, engine, **kwds)
    944             self.options["has_index_names"] = kwds["has_index_names"]
    945 
--> 946         self._make_engine(self.engine)
    947 
    948     def close(self):

~\anaconda3\lib\site-packages\pandas\io\parsers.py in _make_engine(self, engine)
   1176     def _make_engine(self, engine="c"):
   1177         if engine == "c":
-> 1178             self._engine = CParserWrapper(self.f, **self.options)
   1179         else:
   1180             if engine == "python":

~\anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(self, src, **kwds)
   2006         kwds["usecols"] = self.usecols
   2007 
```

```
-> 2008         self._reader = parsers.TextReader(src, **kwds)
   2009         self.unnamed_cols = self._reader.unnamed_cols
   2010

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas\_libs\parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\Ron\\Desktop\\Client\\Country_14-JAN-2019.csv'
```

In [ ]:

https://www.sqlitetutorial.net/sqlite-python/create-tables/

When you connect to an SQLite database file that does not exist, SQLite automatically creates the new database for you.

To create a database, first, you have to create a Connection object that represents the database using the connect() function of the sqlite3 module.

For example, the following Python program creates a new database file pythonsqlite.db in the c:\sqlite\db folder.

Note that you must create the c:\sqlite\db folder first before you execute the program. Or you can place the database file a folder of your choice.

In [ ]:

```
import sqlite3
from sqlite3 import Error


def create_connection(db_file):
    """ create a database connection to a SQLite database """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print(sqlite3.version)
    except Error as e:
        print(e)
    finally:
        if conn:
            conn.close()


if __name__ == '__main__':
    create_connection(r"E:\VCS\GitHub\Machine-Learning-with-Python\data\movie.db")
```

In this code:

First, we define a function called create_connection() that connects to an SQLite database specified by the database file db_file. Inside the function, we call the connect() function of the sqlite3 module.

The connect() function opens a connection to an SQLite database. It returns a Connection object that represents the database. By using the Connection object, you can perform various database operations.

In case an error occurs, we catch it within the try except block and display the error message. If everything is fine, we display the SQLite database version.

It is a good programming practice that you should always close the database connection when you complete with it.

Second, we pass the path of the database file to the create_connection() function to create the database. Note that the prefix r in the r"E:\VCS\GitHub\DataScienceAtWork\data\movie.db" instructs Python that we are passing a raw string.

Let's run the program and check the E:\VCS\GitHub\DataScienceAtWork\data folder.

python sqlite create database If you skip the folder path E:\VCS\GitHub\DataScienceAtWork\data, the program will create the database file in the current working directory (CWD).

If you pass the file name as :memory: to the connect() function of the sqlite3 module, it will create a new database that resides in the memory (RAM) instead of a database file on disk.

In [ ]:

```

```

In [ ]:

```

```

In [7]:

```
import pymysql
import pandas as pd

# Create dataframe
data = pd.DataFrame({
    'Capital':["Kolkata", "Hyderabad", "Bengaluru"],
    'Founded':['1596', '1561', '1537'],
    'Address':['WB','TS','KA']
})


# Connect to the database
connection = pymysql.connect(host='localhost',
                         user='root',
                         password='',
                         db='mydb')


# create cursor
cursor=connection.cursor()

# creating column list for insertion
cols = "`,`".join([str(i) for i in data.columns.tolist()])

# Insert DataFrame recrds one by one.
for i,row in data.iterrows():
    sql = "INSERT INTO `city` (`" +cols + "`) VALUES (" + "%s,"*(len(row)-1) + "%s)"
    cursor.execute(sql, tuple(row))

    # the connection is not autocommitted by default, so we must commit to save our changes
    connection.commit()

# Execute query
sql = "SELECT * FROM `city`"
cursor.execute(sql)

# Fetch all the records
result = cursor.fetchall()
for i in result:
    print(i)

connection.close()



# Note :-

# My table description

# describe city;

# +---------+-------------+------+-----+---------+----------------+

# | Field   | Type          | Null | Key | Default | Extra          |

# +---------+-------------+------+-----+---------+----------------+
# | ID      | int          | NO   | PRI | NULL    | auto_increment |
# | Capital | varchar(255) | YES  |     | NULL    |                |
# | Founded | varchar(255) | YES  |     | NULL    |                |
# | Address | varchar(255) | YES  |     | NULL    |                |
# +---------+--------------+------+-----+---------+----------------+
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-7-e767729de2b9> in <module>
----> 1 import pymysql
      2 import pandas as pd
      3 
      4 # Create dataframe
      5 data = pd.DataFrame({

ModuleNotFoundError: No module named 'pymysql'
```

```
In [ ]:
```