**Prepared by Abhishek Kumar**

[https://www.linkedin.com/in/abhishekkumar-0311/ (https://www.linkedin.com/in/abhishekkumar-0311/)](https://www.linkedin.com/in/abhishekkumar-0311/)

# SQLite Python

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

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

%matplotlib inline
```

In [2]:
```
import pandas as pd
import numpy as np
import sqlite3 as sql
# conn = sql.connect('default.db')
import matplotlib.pyplot as plt
```

## Working with SQLite3 db

- Import the sqlite3 package
  - `import sqlite3 as sql`
- Creata a database or connecct to the existing one
  - `conn = sql.connect('default.db')`
  - this creates a new db if it does not already exist
- Now we need to create tables in this database
  - create a table by providing a schema and ddl
  - Export an existing dataframe to create a table in the database. This newly created table can be used for querying
  - `existing_df.to_sql('dftosql',conn)`
- Queryin the above created sql table

- `query = 'select * from dftosql`
- `dfinit = pd.read_sql(sql_query,conn)`
- The dataframe dfinit will have the query result

In [3]:
```
airport = pd.read_csv('./data/airports.csv')
run = pd.read_csv('./data/runways.csv')
```

In [4]:
```
airport.shape
run.shape
```

Out[4]: (69063, 18)

Out[4]: (42895, 20)

In [9]:
```
# Demonstration of SQL connect and querying though python

def cleanup(table):
    cursor = conn.cursor()
    cursor.execute('DROP TABLE IF EXISTS '+ table )
    return

# sample test before creating a function
conn = sql.connect('default.db')
# airport.to_sql('airport',conn)

# Incase you are Re-creating the table, the above codeline would fail.
# In such scenarios, call the cleanup function.

cleanup('airport')
airport.to_sql('airport',conn)
```

## Extension - with user defined function

In [25]:

```python
# df_return = sqldb(airport,'airport',query,conn=, direct_run=)
defdf = pd.DataFrame()
sql_tbl = 'defdf'
def sqldb(sql_query, conn = sql.connect('default.db'), direct_run = 1, df = defdf, sql_tbl = sql_tbl ):
    ''' df -> pandas dataframe
        sql_tbl -> equivalent table in Db
        sql_query -> query to be executed in sql env
        conn -> connection to database: deafult db is set to default.db
        direct_run -> indicates whether the query needs to be executed on existing table or needs to be re-created '
    if direct_run == 0:
        cursor = conn.cursor()
        drop_query = "DROP TABLE IF EXISTS " + sql_tbl
        cursor.execute(drop_query)
        df.to_sql(sql_tbl,conn)
    dfinit = pd.read_sql(sql_query,conn)
    return dfinit
```

In [26]:

```
conn = sql.connect('default.db')
direct_run = 1
# df_arg = airport
# sql_tbl = 'airport'
query = 'select * from airport limit 10'

df_return = sqldb(query,conn,direct_run=1)
df_return
```

Out[26]:

| | level_0 | index | id | ident | type | name | latitude_deg | longitude_deg | elevation_ft | continent | iso_country | iso_region | municipality |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 6523 | 00A | heliport | Total Rf Heliport | 40.070801 | -74.933601 | 11.0 | None | US | US-PA | Bensalem |
| **1** | 1 | 1 | 323361 | 00AA | small_airport | Aero B Ranch Airport | 38.704022 | -101.473911 | 3435.0 | None | US | US-KS | Leoti |
| **2** | 2 | 2 | 6524 | 00AK | small_airport | Lowell Field | 59.947733 | -151.692524 | 450.0 | None | US | US-AK | Anchor Point |
| **3** | 3 | 3 | 6525 | 00AL | small_airport | Epps Airpark | 34.864799 | -86.770302 | 820.0 | None | US | US-AL | Harvest |
| **4** | 4 | 4 | 6526 | 00AR | closed | Newport Hospital & Clinic Heliport | 35.608700 | -91.254898 | 237.0 | None | US | US-AR | Newport |
| **5** | 5 | 5 | 322127 | 00AS | small_airport | Fulton Airport | 34.942803 | -97.818019 | 1100.0 | None | US | US-OK | Alex |
| **6** | 6 | 6 | 6527 | 00AZ | small_airport | Cordes Airport | 34.305599 | -112.165001 | 3810.0 | None | US | US-AZ | Cordes |
| **7** | 7 | 7 | 6528 | 00CA | small_airport | Goldstone (GTS) Airport | 35.354740 | -116.885329 | 3038.0 | None | US | US-CA | Barstow |
| **8** | 8 | 8 | 324424 | 00CL | small_airport | Williams Ag Airport | 39.427188 | -121.763427 | 87.0 | None | US | US-CA | Biggs |
| **9** | 9 | 9 | 322658 | 00CN | heliport | Kitchen Creek Helibase Heliport | 32.727374 | -116.459742 | 3350.0 | None | US | US-CA | Pine Valley |

## SQLAlchemy

https://towardsdatascience.com/heres-how-to-run-sql-in-jupyter-notebooks-f26eb90f3259 (https://towardsdatascience.com/heres-how-to-run-sql-in-jupyter-notebooks-f26eb90f3259)

In [27]:
```
import sqlalchemy
```

In [28]:
```
sqlalchemy.create_engine('sqlite:///default.db')
```

Out[28]:
```
Engine(sqlite:///default.db)
```

In [29]:
```
# !pip install ipython-sql
```

In [30]:
```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [31]:
```
%sql sqlite:///default.db
```

In [32]:

```
res = %sql select * from airport limit 5
res
```

```
 * sqlite:///default.db
Done.
```

Out[32]:

| level_0 | index | id | ident | type | name | latitude_deg | longitude_deg | elevation_ft | continent | iso_country | iso_region | munic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 6523 | 00A | heliport | Total Rf Heliport | 40.07080078125 | -74.93360137939453 | 11.0 | None | US | US-PA | Be |
| 1 | 1 | 323361 | 00AA | small_airport | Aero B Ranch Airport | 38.704021999999995 | -101.473911 | 3435.0 | None | US | US-KS | |
| 2 | 2 | 6524 | 00AK | small_airport | Lowell Field | 59.94773299999999 | -151.692524 | 450.0 | None | US | US-AK | Anchc |
| 3 | 3 | 6525 | 00AL | small_airport | Epps Airpark | 34.86479949951172 | -86.77030181884766 | 820.0 | None | US | US-AL | H |
| 4 | 4 | 6526 | 00AR | closed | Newport Hospital & Clinic Heliport | 35.6087 | -91.254898 | 237.0 | None | US | US-AR | N |

In [33]:

```
type(res)
```

Out[33]: sql.run.ResultSet

In [34]:

```
dfagn = res.DataFrame()
dfagn
```

Out[34]:

| | level_0 | index | id | ident | type | name | latitude_deg | longitude_deg | elevation_ft | continent | iso_country | iso_region | municipality | sc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 6523 | 00A | heliport | Total Rf Heliport | 40.070801 | -74.933601 | 11.0 | None | US | US-PA | Bensalem | |
| **1** | 1 | 1 | 323361 | 00AA | small_airport | Aero B Ranch Airport | 38.704022 | -101.473911 | 3435.0 | None | US | US-KS | Leoti | |
| **2** | 2 | 2 | 6524 | 00AK | small_airport | Lowell Field | 59.947733 | -151.692524 | 450.0 | None | US | US-AK | Anchor Point | |
| **3** | 3 | 3 | 6525 | 00AL | small_airport | Epps Airpark | 34.864799 | -86.770302 | 820.0 | None | US | US-AL | Harvest | |
| **4** | 4 | 4 | 6526 | 00AR | closed | Newport Hospital & Clinic Heliport | 35.608700 | -91.254898 | 237.0 | None | US | US-AR | Newport | |

In [ ]:

In [ ]:

In [ ]:

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

In [35]:

```python
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 [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)
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
<ipython-input-35-0fd3ab21beb3> in <module>
      5
      6 # Create table - CLIENTS
----> 7 c.execute('''CREATE TABLE CLIENTS
      8              ([generated_id] INTEGER PRIMARY KEY,[Client_Name] text, [Country_ID] integer, [Date] date)''')
      9

OperationalError: table CLIENTS already exists
```

In [36]:

```python
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

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

# 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 tab
# - 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(

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

# export_csv = df.to_csv (r'C:\Users\Ron\Desktop\Client\export_list.csv', index = None, header=True) # Uncomment thi
# 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-36-6285767eb693> in <module>
      9 movie.to_sql('MOVIE', conn, if_exists='append', index = False) # Insert the values from the csv file into the t
able 'CLIENTS'
     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 table 'COUNTRY'
     13

~\anaconda3\lib\site-packages\pandas\io\parsers.py in read_csv(filepath_or_buffer, sep, delimiter, header, names, index
_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspac
e, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_da
tetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal,
lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, dialect, error_bad_lines, warn_bad_line
s, delim_whitespace, low_memory, memory_map, float_precision)
    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 [ ]:

```
1
```

## https://www.sqlitetutorial.net/sqlite-python/create-tables/ (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 [37]:

```python
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")
```

```
2.6.0
```

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 [ ]:

```

```

# Writing SQL query on a dataframe using pandassql

In [38]:

```
# To get multiple outputs in the same cell

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

%matplotlib inline
```

In [39]:

```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[39]:

| | color | director_name | num_critic_for_reviews | duration | director_facebook_likes | actor_3_facebook_likes | actor_2_name | actor_1_facebook_likes | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Color | James Cameron | 723.0 | 178.0 | 0.0 | 855.0 | Joel David Moore | 1000.0 | 760 |
| **1** | Color | Gore Verbinski | 302.0 | 169.0 | 563.0 | 1000.0 | Orlando Bloom | 40000.0 | 309 |
| **2** | Color | Sam Mendes | 602.0 | 148.0 | 0.0 | 161.0 | Rory Kinnear | 11000.0 | 200 |
| **3** | Color | Christopher Nolan | 813.0 | 164.0 | 22000.0 | 23000.0 | Christian Bale | 27000.0 | 448 |
| **4** | NaN | Doug Walker | NaN | NaN | 131.0 | NaN | Rob Walker | 131.0 | |

5 rows × 28 columns

In [40]:

```
#%%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_c
pysqldf(q2)#.sort_values(by=)
pysqldf(q2).sort_values(by='tot_critic', ascending=True)
```

Out[40]:

| | color | director_name | num_critic_for_reviews | duration | director_facebook_likes | actor_3_facebook_likes | actor_2_name | actor_1_facebook_likes | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Color | James Cameron | 723.0 | 178.0 | 0.0 | 855.0 | Joel David Moore | 1000.0 | 76( |
| **1** | Color | James Cameron | 315.0 | 194.0 | 0.0 | 794.0 | Kate Winslet | 29000.0 | 65{ |
| **2** | Color | James Cameron | 210.0 | 153.0 | 0.0 | 539.0 | Jenette Goldstein | 780.0 | 204 |
| **3** | Color | James Cameron | 94.0 | 141.0 | 0.0 | 618.0 | Tia Carrere | 2000.0 | 14( |
| **4** | Color | James Cameron | 82.0 | 171.0 | 0.0 | 638.0 | Todd Graff | 2000.0 | 54 |
| **5** | Color | James Cameron | 250.0 | 154.0 | 0.0 | 604.0 | Carrie Henn | 2000.0 | 85 |
| **6** | Color | James Cameron | 204.0 | 107.0 | 0.0 | 255.0 | Brian Thompson | 2000.0 | 3{ |

7 rows × 28 columns

Out[40]:

| | director_name | tot_critic |
|---|---|---|
| **0** | Steven Spielberg | 6582.0 |
| **1** | Ridley Scott | 4616.0 |
| **2** | Martin Scorsese | 4285.0 |

| | director_name | tot_critic |
|---|---|---|
| **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[40]:

| | 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 (https://www.kdnuggets.com/2017/02/python-speak-sql-pandasql.html)**

In [ ]:

```
1
```

In [ ]:

```
1
```