Project Name: Reading Large Files with SQL Using Chunk Size

Table of Content

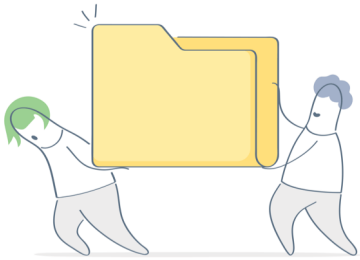
Demo

```
In [29]: df.head()
```

Out[29]:

	index	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	1	80117401	Movie	Jandino: Whatever it Takes	None	Jandino Asporaat	United Kingdom	September 9, 2016	2016	TV-MA	94 min	Stand-Up Comedy	Jandino Asporaat riffs on the challenges of ra...
1	7	80164077	Movie	Fabrizio Copano: Solo pienso en mi	Rodrigo Toro, Francisco Schultz	Fabrizio Copano	Chile	September 8, 2017	2017	TV-MA	60 min	Stand-Up Comedy	Fabrizio Copano takes audience participation t...
2	10	80169755	Movie	Joaquín Reyes: Una y no más	José Miguel Contreras	Joaquín Reyes	None	September 8, 2017	2017	TV-MA	78 min	Stand-Up Comedy	Comedian and celebrity impersonator Joaquín Re...
3	53	80177405	Movie	Marc Maron: Too Real	Lynn Shelton	Marc Maron	United States	September 5, 2017	2017	TV-MA	70 min	Stand-Up Comedy	Battle-scarred stand-up comedian Marc Maron un...
4	62	81054495	Movie	Mo Gilligan: Momentum	Chris Howe	Mo Gilligan	United Kingdom	September 30, 2019	2019	TV-MA	64 min	Stand-Up Comedy	Comedian Mo Gilligan blends smooth moves and s...

Overview



This is about reading large files with SQL through splitting it into chunk size.
There is a stark difference between large and big data.

This repository contains the code for reading large files with SQL through splitting it up into smaller chunks.
It used Pandas, Numpy and sqlite3 libraries.
These libraries help to perform individually one particular functionality.
Data is unavoidably messy in real world.
And Pandas, is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data. Pandas objects rely heavily on Numpy objects.
Numpy is used for working with arrays. It stands for Numerical Python.
Sqlite3 is a single file relational database bundled with most standard python installs.
Data Science professionals often encounter very large data sets with hundreds of dimensions and millions of observations. So, it is one of the important skills that I am learning here. There are multiple ways to handle large data sets. It supplies precisely what we need.
Parameter essentially means the number of rows to be read into a data frame at any single time in order to fit into the local memory.
Here, we are loading only some of the lines into memory at any given time. By doing this, basically we have reduced memory usage and still receive same results.
The screenshot will help you to understand flow of output.

Motivation

One of the major motive to create this, is I wanted to know working of SQL along with Large Files of Python. The reason behind making is, I was baffled when I encountered an error and I couldn't read the data from csv file as my local machine has 8GB of RAM.
Therefore, thought to create this one. The purpose of creating this repository is I wanted to dig deeper into Pandas, thatswhen I realized that pandas.read_csv has a parameter called chunksize. When we use argument to pandas, we get back an iterator over DataFrames rather than one single DataFrame. Though here, I have not even used that big dataset file. As I was more interested in concept rather than anything else. By building such mini project helped me to gain knowledge about other functionalities of Pandas library, which is most popular, common and even I have used almost everytime. Thatswhy Pandas is powerful.
Steps to handle large files in SQL database:
1) Create a connector to a database.
2) Building the database.
3) Construct the Pandas Dataframe by calling SQL query.

Technical Aspect

Numpy contains a multi-dimensional array and matrix data structures. It works with the numerical data. Numpy is faster because is densely packed in memory due to its homogeneous type. It also frees the memory faster.
Pandas module mainly works with the tabular data. It contains DataFrame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.

Pandas is very efficient with small data(usually from 100MB upto 1GB) and performance is rarely a concern. Pandas has its own limitation when it comes to big data due to its algorithm and local memory constraints.
Sqlite3 needs to be installed then can be integrated with python. It is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of SQL. It is used to perform operation related to database.
Sqlalchemy is python SQL toolkit. It is object relational mapper (orm) that gives application developers the full power and flexibility of SQL. Major benefit is varied databases support that it provides and has mature, high performing architecture and function-based query construction. ORM is a code library that automates the transfer of data stored in relational database tables into objects that more commonly used in application code.

Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed then please install Anaconda Navigator from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter*.

Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.
Open Anaconda Prompt, Perform the following steps:
cd <PATH>
pip install numpy
pip install pandas
pip install sqlite3
You can also create requirement.txt file as, pip freeze > requirements.txt
run files.
If you want you can directly install packages into Jupyter Notebook Cell with ! in front.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.

eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\5)Reading_Large_Data\Reading_Large_File_With_SQL

In Anconda Prompt, pip install -r requirements.txt to install all packages.
Open in Jupyter Notebook, <filename>.ipynb
That is,
Open in Jupyter Notebook, Reading_Large_File_With_SQL_Using_Chunksize.ipynb
It first creates and loads a csv_database.db then load netflix_titles.csv file as input to it.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder] [cd means change directory]

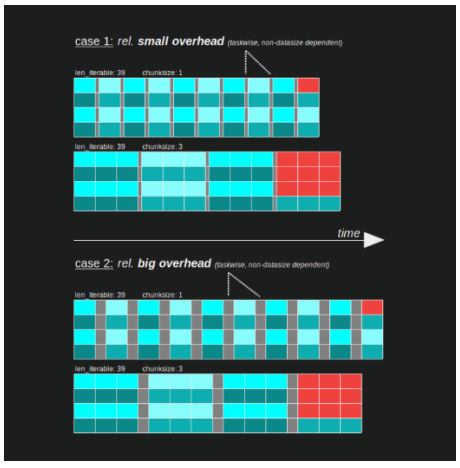
Directory Tree/Structure of Project

Folder: 5)Reading_Large_Data>Reading_Large_File_With_SQL
Reading_Large_File_With_SQL_Using_Chunksize.ipynb

To Do/Future Scope

Can try another technique.

Technologies Used/System Requirement/Tech Stack



Credits

Pradeep Sir