

Data Engineering task

Kristijan Rebrović, 7.7.2022

Task:

Every week 3 csv files will be dropped at a designated file location. My task is to read files, create tables in database, and append new data every week. Next task is to transform data in a way to be used for modeling, specification is provided in DE Assignment.doc.

Technologies I'd be using are:

Python

Libraries:

- Sqlite3 – for DB
- Schedule - for repetitive tasks, schedule task every week
- time

I decided for the most simple solution, all using Python. There are also other solutions as Crontab, Bash or Airflow but for this task this will do just well.

DB Client:


Dbeaver


It's free and opensource so it's a good solution. It also supports sqlite3 db.

Solution

First step – we need to setup file location. We will need two folders:

Name

 _data_in

 Archive

In `_data_in` folder new files will be dropped weekly. Also we need to know exact time at which data will be delivered. If that is not fixed, we'll need to make a script that will check folder regularly and trigger data append script when data is delivered.

When data is delivered, we append data to db and rename files (adding timestamp to name) and move files to Archive folder. Then we write to log file (in .txt file in folder or/and table in db. Next we send confirmation mail to our users and IT support. If something fails on the way, we also write to log and send email.

Reading data and creating tables in db:

```
df_stores = pd.read_csv('C:/Users/krebrovic/Desktop/Data engineering zadatak/_data_in/stores_dataset.csv')

df_features = pd.read_csv('C:/Users/krebrovic/Desktop/Data engineering zadatak/_data_in/Features_dataset.csv',
parse_dates = ['Date'])

df_sales = pd.read_csv('C:/Users/krebrovic/Desktop/Data engineering zadatak/_data_in/sales_dataset.csv', parse_dates =
['Date'])

con = sl.connect('myDB_new.db') -- this line creates connection to db

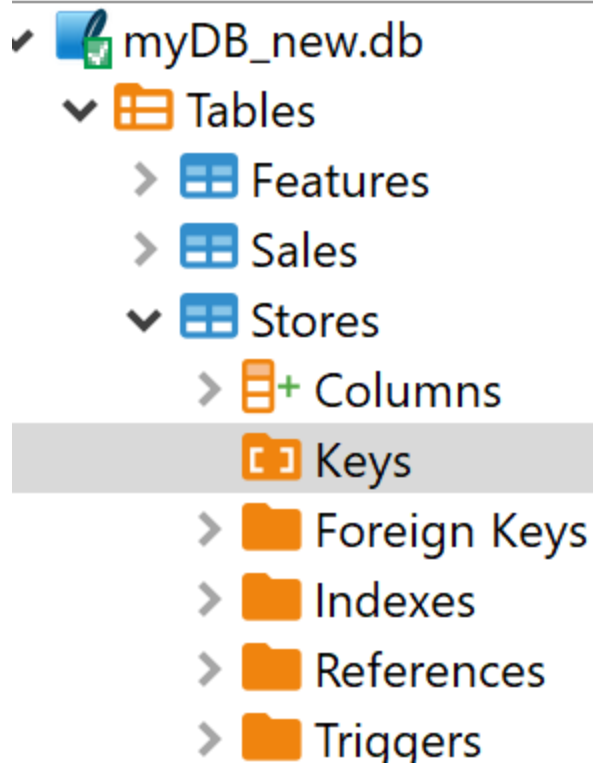
df_stores.append('Stores', con) -- this line creates table 'Stores' in db

df_features.to_sql('Features', con) -- this line creates table 'Features' in db

df_sales.to_sql('Sales', con) -- this line creates table 'Sales' in db
```

Now we have the data and we use Dbeaver to connect to db:

Enter a part of object name here



Now for the data analysis part:

```
select Store, DATE, Weekly_Sales ,  
sum(weekly_Sales) over(partition by Store) total_sale,  
lag(weekly_Sales) over(partition by Store order by DATE) sales_last_week,  
sum_last_4w,  
min_last_4w,  
max_last_4w,  
strftime('%m', date) AS num_Month,  
(sum(MarkDown1) over(partition by Store) / sum(weekly_Sales) over(partition by Store) ) * 100 as  
percentM1,  
(sum(MarkDown2) over(partition by Store) / sum(weekly_Sales) over(partition by Store) ) * 100 as  
percentM2,  
(sum(MarkDown3) over(partition by Store) / sum(weekly_Sales) over(partition by Store) ) * 100 as  
percentM3,
```

```

(sum(MarkDown4) over(partition by Store) / sum(weekly_Sales) over(partition by Store) ) * 100 as
percentM4,

(sum(MarkDown5) over(partition by Store) / sum(weekly_Sales) over(partition by Store) ) * 100 as
percentM5

FROM

(

SELECT

ST.Store,

SA.DATE,

sum(sa.Weekly_Sales) Weekly_Sales,

sum(f.MarkDown1) MarkDown1,

sum(f.MarkDown2) MarkDown2,

sum(f.MarkDown3) MarkDown3,

sum(f.MarkDown4) MarkDown4,

sum(f.MarkDown5) MarkDown5,

case when sa.date >= datetime(sa.date,'-28 days','localtime') then sum(weekly_Sales) else null END
sum_last_4w,

case when sa.date >= datetime(sa.date,'-28 days','localtime') then min(weekly_Sales) else null END
min_last_4w,

case when sa.date >= datetime(sa.date,'-28 days','localtime') then max(weekly_Sales) else null END
max_last_4w

from Stores st

join Sales sa

on st.Store = sa.Store

join Features f

on f.Store = st.Store

and f.Date = sa.Date

where st.store = 1

group by st.Store , sa.Date

) a

```

ORDER BY DATE DESC

;

We could also add Y2Y(volume and %), Q2Q, M2M sales, Monthly sales

The initial part is over, we now have set up File location, created db and tables and set up Dbeaver to query our data.

Next we need to:

- Make possible to append new data weekly
- Automatize solution

Next step

files DE.Py and Start_import.py

First we start Start_import.py - it checks if is time to start import(weekly), we need to define when the file will be delivered and define schedule. In this case it is monday 10 am. At that time script DE.Py triggers and import starts. Data is appended, log created and email sent.

Conclusion

It is ok solution for a small business, we are using free tools that can handle small amounts of data. In large systems we need to have more robust solutions. Other solutions could include AWS(Redshift), Local DWH developed in ETL tools, local scripts using Airflot, etc..

Security is issue too, we need to define permissions on folder.

Problem is that it is local and script needs to be running all the time, we could move it to server for example.

Also, someone can accidentally delete files. We need to replicate.