Connect databses in Python

Masaya Kameyama

2021-07-29

- toc: false
- branch: master
- badges: true
- comments: true
- categories:
- hide: false
- search_exclude: true

Summary

There are many mathods to connect db in python. We introduce two packages sqlalchemy and psycopg2. We assume .env file in which connection settings are are defined.

RDB

sqlalchemy

```python
import os
import sys
import sqlalchemy
from os.path import join, dirname
from dotenv import load_dotenv
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, create_engine
from sqlalchemy.orm import sessionmaker
```

```
import pandas as pd
import time
dotenv_path = join(dirname("$home"), '.env')
load_dotenv(dotenv_path)
conn_aurora = '{}://{}:{}@{}:{}/{}'.format('postgresql', os.environ['WRITE_RDB_USERNAME'],
conn_redshift = '{}://{}:{}@{}:{}/{}'.format('postgresql', os.environ['DWH_USERNAME'], os.
engine = create_engine(conn_redshift, echo=True)
```

```
def sql(query):
    session = sessionmaker(bind=engine)()
    df = pd.read_sql_query(sql=query, con=engine)
    time.sleep(1)
    session.close()
    return df
```

example query

```
q = "select id from companies limit 10"
```

```
sql(q)
```

```
2021-07-29 19:56:34,134 INFO sqlalchemy.engine.Engine select version()
2021-07-29 19:56:34,136 INFO sqlalchemy.engine.Engine [raw sql] {}
2021-07-29 19:56:34,165 INFO sqlalchemy.engine.Engine select current_schema()
2021-07-29 19:56:34,165 INFO sqlalchemy.engine.Engine [raw sql] {}
2021-07-29 19:56:34,206 INFO sqlalchemy.engine.Engine select id from companies limit 10
2021-07-29 19:56:34,208 INFO sqlalchemy.engine.Engine [raw sql] {}
```

	id
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9

	id
9	10

psycopg2

```
import psycopg2
import pandas as pd
import time
from sshtunnel import SSHTunnelForwarder


def queryRedshift(sql):
    conn = psycopg2.connect(
        host=os.environ['DWH_HOST'],
        port=os.environ['DWH_PORT'],
        dbname=os.environ['DWH_DATABASE'],
        user=os.environ['DWH_USERNAME'],
        password=os.environ['DWH_PASSWORD'])
    cur = conn.cursor()
    cur.execute(sql)
    result = cur.fetchall()
    colnames = [col.name for col in cur.description]
    # pandas.DataFrame
    new_result = [[one for one in one_result]  for one_result in result]
    result = pd.DataFrame(new_result,columns=colnames)
    cur.close()
    conn.close()
    #        1
    time.sleep(1)
    return result
```

```
queryRedshift(q)
```

	id
0	1
1	2
2	3
3	4
4	5
5	6
6	7

	id
7	8
8	9
9	10

Bigquery

It is known that the bq performance in python is depends on a connection method. https://medium.com/@davide.sarra/slow-bigquery-results-no-more-8aa4dde92613

Lets campare short and long time queris ## short time query

```
query_short_time="""
select id from {}.{}.companies
""".format(os.environ['BQ_PROJECT_NAME'], os.environ['BQ_DATASET_NAME'])
```

```
start=time.perf_counter()
pd.read_gbq(query_short_time, os.environ['BQ_PROJECT_NAME'])
print(time.perf_counter()-start)
```

```
1.206214640999974
```

```
start=time.perf_counter()
pd.read_gbq(query_short_time, os.environ['BQ_PROJECT_NAME'], use_bqstorage_api=True)
print(time.perf_counter()-start)
```

```
2.5740085850000014
```

Long time query

```
query_long_time="""
omit
""";
```

```
start=time.perf_counter()
pd.read_gbq(query_long_time, os.environ['BQ_PROJECT_NAME'])
print(time.perf_counter()-start)
```

```
209.955542535
```

```
start=time.perf_counter()
pd.read_gbq(query_long_time, os.environ['BQ_PROJECT_NAME'], use_bqstorage_api=True)
print(time.perf_counter()-start)
```

```
10.184412958999985
```

cf) Long time query in Redshift

```
query="""
omit
""";
```

```
start=time.perf_counter()
sql(query)
print(time.perf_counter()-start)
```

```
49.52570845600002
```