interfacesql

April 16, 2021

1 Notebook para comandos SQL

geralmente utilizo o package psycopg2 para me comunicar com o banco de dados postgresql, às vezes utilizo pgadmin4 para melhor visualização dos db e tabelas, mas como utilizo apenas para puxar tabelas do banco de dados para análise de dados, costumo só utilizar python com notebook.

[43]:
```
import psycopg2 #package python de interação com banco de dados, referências em:
 ↪ https://www.psycopg.org/docs/ e https://pypi.org/project/psycopg2/
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT
import pandas as pd #mais famoso package de manipulação de dados em python
```

[44]:
```
# informações do banco de dados
host, port, db, user, password = "localhost", 5432, "dbbft", "postgres", "1234"
```

[5]:
```
#por enquanto, a conexão deve somente acontecer no server
conn = psycopg2.connect(user=user, password=password)
```

Na execução do objeto psycopg2.connect().cursor().execute() por padrão ele não escreve direto no banco de dados, é necessário chamar o objeto da classe connect(), connect().commit(), para realizar um commit no server postgresql, mas usando ISOLATION_LEVEL_AUTOCOMMIT, o objeto execute() já realiza os comandos dentro do server

[30]:
```
conn.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT)
```

[31]:
```
# a classe cursor permite realizar queries com o banco de dados
qry = conn.cursor()
```

[13]:
```
qry.execute("CREATE DATABASE dbbft")
```

[32]:
```
#se conectando à database
conn = psycopg2.connect(host=host, port=port, database=db, user=user,␣
 ↪password=password)
```

[33]:
```
create_table = "CREATE TABLE usuarios (id INT GENERATED ALWAYS AS IDENTITY␣
 ↪PRIMARY KEY, email TEXT UNIQUE NOT NULL, senha TEXT UNIQUE NOT NULL);"
```

[34]:
```
qry.execute(create_table)
```

[35]:
```
# adicionando um email, senha na tabela usuários para afins de teste
qry.execute("INSERT INTO usuarios (email, senha) VALUES ('teste@mail.com',␣
 ↪'abc123');")
```

Para análise de dados é interessante puxar a tabela utilizando pandas para a manipulação

[36]:
```
select = "SELECT * FROM usuarios;"
```

[37]:
```
data = pd.read_sql(select, conn)
data
```

[37]:

	id	email	senha
0	1	teste@mail.com	abc123

DB, tabela criadas com sucesso!

1.1 Verificando se os dados estão sendo gravados no DB

[45]:
```
host, port, db, user, password = "localhost", 5432, "dbbft", "postgres", "1234"
conn = psycopg2.connect(host=host, port=port, database=db, user=user,␣
 ↪password=password)
qry = conn.cursor()
select = "SELECT * FROM usuarios;"
data = pd.read_sql(select, conn)
```

[46]:
```
data
```

[46]:

	id	email	senha
0	1	teste@mail.com	abc123
1	3	qwerty@email.com	81dc9bdb52d04dc20036dbd8313ed055

email cadastrado já com senha criptografada

1.2 Percebi que escrevi UNIQUE na coluna senha

[47]:
```
alter = "ALTER TABLE usuarios DROP CONSTRAINT usuarios_senha_key"
```

[48]:
```
qry.execute(alter)
```

[49]:
```
conn.commit()
```

1.3 Verificando a existência de emails e senhas no DB

[50]:
```
data = pd.read_sql(select, conn)
```

[51]:
```
data
```

[51]:

	id	email	senha
0	1	teste@mail.com	abc123

```
1   3      qwerty@email.com  81dc9bdb52d04dc20036dbd8313ed055
2  10       qwerty@asdf.com  81dc9bdb52d04dc20036dbd8313ed055
3  13  tamoquasela@mail.com  81dc9bdb52d04dc20036dbd8313ed055
4  14          ft@teste.com  81dc9bdb52d04dc20036dbd8313ed055
5  15      teste59@mail.com  81dc9bdb52d04dc20036dbd8313ed055
6  16         1234@1234.com  81dc9bdb52d04dc20036dbd8313ed055
7  17       asdf@qwerty.com  81dc9bdb52d04dc20036dbd8313ed055
```

[53]:
```
data['email'].value_counts() #Conta quantas vezes um observável da variável␣
↪aparece, basicamente, mostra a sua frequência
```

[53]:
```
qwerty@asdf.com         1
tamoquasela@mail.com    1
asdf@qwerty.com         1
qwerty@email.com        1
teste@mail.com          1
1234@1234.com           1
teste59@mail.com        1
ft@teste.com            1
Name: email, dtype: int64
```

cada email é único

[55]:
```
data['senha'].value_counts()
```

[55]:
```
81dc9bdb52d04dc20036dbd8313ed055    7
abc123                              1
Name: senha, dtype: int64
```

senhas podem ser repetidas

[ ]:
```

```