Managing SQL Database Using Python

July 24, 2024

0.1 Importing Necessary Libraries

[1]:
```
import pandas as pd
import configparser
import psycopg2

# magic function that will allow us to connect to postgres SQL database
%load_ext sql
```

0.2 Creating the Configuration File

[2]:
```
import configparser

# Define the configuration
config = configparser.ConfigParser()
config['SQL'] = {
    'DB_NAME_DEFAULT': 'postgres',
    'DB_USER': 'postgres',
    'DB_PASSWORD': 'outfitters'
}

# Specify the file path where you want to save the configuration file
config_file_path = 'config.ini'

# Write the configuration to the file
with open(config_file_path, 'w') as config_file:
    config.write(config_file)

print(f"Configuration file saved to {config_file_path}")
```

```
Configuration file saved to config.ini
```

0.3 Read the Paramaters from the Config File

[3]:
```
config = configparser.ConfigParser()
config.read('config.ini')
DB_NAME_DEFAULT = config.get('SQL', 'DB_NAME_DEFAULT')
DB_USER = config.get('SQL', 'DB_USER')
```

```
DB_PASSWORD = config.get('SQL', 'DB_PASSWORD')
```

0.4 Connect to the Default Database

[4]:
```
conn= psycopg2.connect("host=127.0.0.1 dbname={} user={} password={}".
 ↪format(DB_NAME_DEFAULT, DB_USER, DB_PASSWORD))
conn.set_session(autocommit= True)
cur= conn.cursor()
```

0.5 Create Coffee Shops Database

[ ]:
```
cur.execute("DROP DATABASE IF EXISTS coffeeshops")
cur.execute("CREATE DATABASE coffeeshops WITH ENCODING 'utf8' TEMPLATE␣
 ↪template0")
conn.close()
```

0.6 Connect to Coffee Shops Database

[17]:
```
conn= psycopg2.connect("host=127.0.0.1 dbname=coffeeshops user={} password={}".
 ↪format(DB_USER, DB_PASSWORD))
cur= conn.cursor()
```

0.7 Drop Employees Table if Exists and Create a New One

[18]:
```
cur.execute(
'''
DROP TABLE IF EXISTS employees
''')
conn.commit()
```

[15]:
```
conn.close()
```

[19]:
```
cur.execute(
'''
CREATE TABLE IF NOT EXISTS employees (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(50),
    hire_date DATE,
    shop_name VARCHAR(50),
    salary INT
)
'''
)
conn.commit()
```

0.8 Read the csv File

[20]:
```
df = pd.read_csv(r"C:\Users\HH\Downloads\sql-with-python-main\coffeeshop.csv")
```

[21]:
```
df['hire_date'] = pd.to_datetime(df['hire_date'], format='%d/%m/%Y').dt.
 ↪strftime('%Y-%m-%d')
```

[36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   employee_id  100 non-null    int64 
 1   first_name   100 non-null    object
 2   last_name    100 non-null    object
 3   email        74 non-null     object
 4   hire_date    100 non-null    object
 5   shop_name    100 non-null    object
 6   salary       100 non-null    int64 
dtypes: int64(2), object(5)
memory usage: 5.6+ KB
```

0.9 Insert the values in the employees table of coffee shop database

[22]:
```
for i, row in df.iterrows():
    cur.execute(
    '''
    INSERT INTO employees
    (employee_id, first_name, last_name, email, hire_date, shop_name, salary)
    VALUES(%s, %s, %s, %s, %s, %s, %s)
    ''', row.tolist())

conn.commit()
```

[23]:
```
conn.close()
```

0.10 Running queries to make sure everything works as expected

[24]:
```
conn_string= "postgresql://{}:{}@127.0.0.1/coffeeshops".format(DB_USER,␣
 ↪DB_PASSWORD)
%sql $conn_string
```

[25]:
```
%sql SELECT COUNT(*) FROM employees;
```

```
 * postgresql://postgres:***@127.0.0.1/coffeeshops
1 rows affected.
```

[25]:
```
[(100,)]
```

[27]:
```
%sql SELECT * FROM employees LIMIT 5;
```

```
 * postgresql://postgres:***@127.0.0.1/coffeeshops
5 rows affected.
```

[27]:
```
[(54733, 'Vonni', 'Davsley', 'vdavsley0@joomla.org', datetime.date(2016, 3, 21),
'Urban Grind', 9253),
 (49009, 'Cary', 'Brauninger', 'cbrauninger1@independent.co.uk',
datetime.date(2014, 8, 26), 'Common Grounds', 45036),
 (59278, 'Gerianna', 'Tolcharde', 'NaN', datetime.date(2019, 9, 6), 'Common
Grounds', 37007),
 (92214, 'Helene', 'Bealing', 'NaN', datetime.date(2022, 11, 12), 'Urban Grind',
13118),
 (32890, 'Vivien', 'McCrackem', 'NaN', datetime.date(2019, 2, 12), 'Common
Grounds', 59034)]
```

[30]:
```
%sql SELECT COUNT(*) FROM employees WHERE email= 'NaN';
```

```
 * postgresql://postgres:***@127.0.0.1/coffeeshops
1 rows affected.
```

[30]:
```
[(26,)]
```

[41]:
```
%sql SELECT ROUND(AVG(salary), 2), MAX(salary), MIN(salary) FROM employees;
```

```
 * postgresql://postgres:***@127.0.0.1/coffeeshops
1 rows affected.
```

[41]:
```
[(Decimal('40492.38'), 67797, 9253)]
```

[43]:
```
# Order by the number of employees per shop descending

%sql SELECT shop_name, COUNT(*) FROM employees GROUP BY shop_name ORDER BY 2␣
↪DESC;
```

```
 * postgresql://postgres:***@127.0.0.1/coffeeshops
3 rows affected.
```

[43]:
```
[('Urban Grind', 35), ('Early Rise', 33), ('Common Grounds', 32)]
```

[46]:
```
# Order by the average salary per shop ascending

%sql SELECT shop_name, ROUND(AVG(salary), 1) FROM employees GROUP BY shop_name␣
↪ORDER BY 2 ASC;
```

```
 * postgresql://postgres:***@127.0.0.1/coffeeshops
3 rows affected.
```

[46]:
```
[('Urban Grind', Decimal('39313.9')),
 ('Early Rise', Decimal('40240.1')),
 ('Common Grounds', Decimal('42041.5'))]
```

[ ]: