

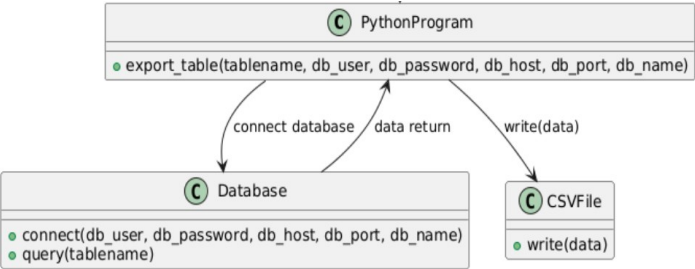
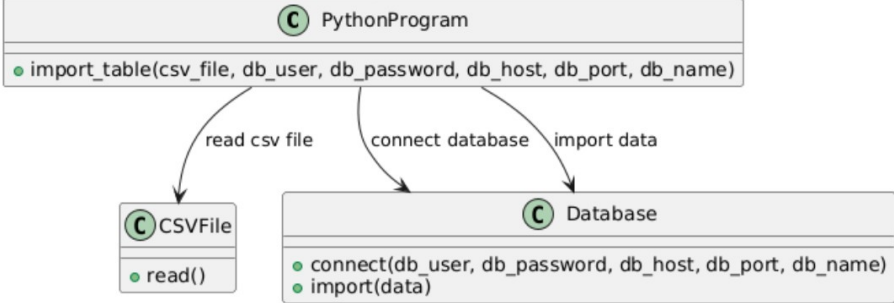
# Student Number: 21

## Author: Davy Hui

This homework assignment demonstrates how my program, written in Python, exports a Postgres table to a CSV file and then uses the same tool to import it back

Program Name: **data\_transfer.py**

Programming Language: **Python**

Export Table to CSV	<p>Usage:</p> <pre>python data_transfer.py export &lt;tablename&gt; --db_user &lt;userid&gt; --db_password &lt;password&gt; --db_host &lt;servername&gt; --db_port &lt;serverport&gt; --db_name &lt;databasename&gt;</pre>
	 <pre>classDiagram     class PythonProgram {         +export_table(tablename, db_user, db_password, db_host, db_port, db_name)     }     class Database {         +connect(db_user, db_password, db_host, db_port, db_name)         +query(tablename)     }     class CSVFile {         +write(data)     }     PythonProgram --&gt; Database : connect database     Database --&gt; PythonProgram : data return     PythonProgram --&gt; CSVFile : write(data)</pre> <p>The diagram illustrates the export process. The <b>PythonProgram</b> class has a method <code>export_table(tablename, db_user, db_password, db_host, db_port, db_name)</code>. It interacts with the <b>Database</b> class via a <code>connect database</code> message and receives <code>data return</code>. The <b>Database</b> class has methods <code>connect(db_user, db_password, db_host, db_port, db_name)</code> and <code>query(tablename)</code>. The <b>CSVFile</b> class has a <code>write(data)</code> method, which is called by <b>PythonProgram</b> via a <code>write(data)</code> message.</p>
Import Table from CSV	<p>Usage:</p> <pre>python data_transfer.py import &lt;tablename&gt; --csv_file &lt;csv filename&gt; --db_user &lt;userid&gt; --db_password &lt;password&gt; --db_host &lt;servername&gt; --db_port &lt;serverport&gt; --db_name &lt;databasename&gt;</pre>
	 <pre>classDiagram     class PythonProgram {         +import_table(csv_file, db_user, db_password, db_host, db_port, db_name)     }     class CSVFile {         +read()     }     class Database {         +connect(db_user, db_password, db_host, db_port, db_name)         +import(data)     }     PythonProgram --&gt; CSVFile : read csv file     PythonProgram --&gt; Database : connect database     PythonProgram --&gt; Database : import data</pre> <p>The diagram illustrates the import process. The <b>PythonProgram</b> class has a method <code>import_table(csv_file, db_user, db_password, db_host, db_port, db_name)</code>. It interacts with the <b>CSVFile</b> class via a <code>read csv file</code> message. It also interacts with the <b>Database</b> class via <code>connect database</code> and <code>import data</code> messages. The <b>CSVFile</b> class has a <code>read()</code> method. The <b>Database</b> class has methods <code>connect(db_user, db_password, db_host, db_port, db_name)</code> and <code>import(data)</code>.</p>

Here is an example demonstrating how to export the 'meal' table to a CSV file named 'meal.csv'. After the administrator makes changes to the file, it can be imported back into the database and displayed on the meal screen. The same procedure applies to the Price and Staff tables as well.

Step 1. In the import\_export\_homework folder, execute the following export command to export the 'meal' table. The program will default to exporting it as 'meal.csv' (i.e., the table name followed by .csv).

```
dhui@dhui-VMware-Virtual-Platform:~/import_export_homework
~/import_export_homework main proj2 3.10.5 python data_transfer.py export meal --db_user postgres --db_password Abcd1234 --db_host localhost --db_port 5432 --db_name elderlydb

2025-01-23 07:10:51,599 - INFO - Data exported successfully to meal.csv.
~/import_export_homework main proj2 3.10.5 ls -l
total 420
-rw-rw-r-- 1 dhui dhui 3638 Jan 19 12:35 data_transfer.py
-rw-rw-r-- 1 dhui dhui 1988 Jan 23 07:10 meal.csv
```

Step 2. Open CSV file for editing

**Text Import - [meal.csv]**

**Import**

Character set: Unicode (UTF-8)

Locale: Default - English (USA)

From row: 1

**Separator Options**

☐ Fixed width ☒ Separated by

☒ Tab ☒ Comma ☒ Semicolon ☐ Space ☐ Other

☐ Merge delimiters ☐ Trim spaces String delimiter: "

**Other Options**

☐ Format quoted field as text ☐ Detect special numbers

☐ Evaluate formulas ☒ Detect scientific notation

**Fields**

Column type:

	Standard	breakfast_menu	lunch_menu	teatime_menu	dinner_menu
1	day_of_month				
2	1	燕麥粥配水果	烤雞沙拉	草本茶與餅乾	烤魚配蔬菜
3	2	炒蛋	蔬菜湯	蜂蜜優格	炒豆腐配米飯
4	3	全麥吐司	藜麥沙拉	新鮮水果	扁豆燉菜
5	4	香蕉菠菜奶昔	火雞三明治	餅乾與起司	烤雞配地瓜
6	5	堅果粥	菠菜與費塔起司沙拉	綠茶與鬆餅	蔬菜咖哩配米飯
7	6	優格配格蘭諾拉	烤蔬菜捲	水果沙拉	牛肉燉菜
8	7	煎餅配糖漿	雞凱薩沙拉	鬆餅	意大利麵配番茄
9	8	水煮蛋	扁豆沙拉	草本茶	烤鮭魚配蘆筍

Help Cancel OK

Step 3. After completing the changes, run the import command to import the data back into the 'meal' table.

```
dhui@dhui-VMware-Virtual-Platform:~/import_export_homework
~/import_export_homework main proj2 3.10.5 python data_transfer.py import meal --csv_file meal.csv --db_user postgres --db_password Abcd1234 --db_host localhost --db_port 5432 --db_name elderlydb
2025-01-23 07:30:41,125 - INFO - Table cleared successfully.
2025-01-23 07:30:41,179 - INFO - Data uploaded successfully! 31 records imported.
```

Step 4. The updated meal data will then be displayed on the meal page.

聯絡電話  
2343 2255

關注我們

主頁

員工簡介

申請資助

服務

預約

院舍設施

地點資訊

關於我們

聯絡我們

當月餐單

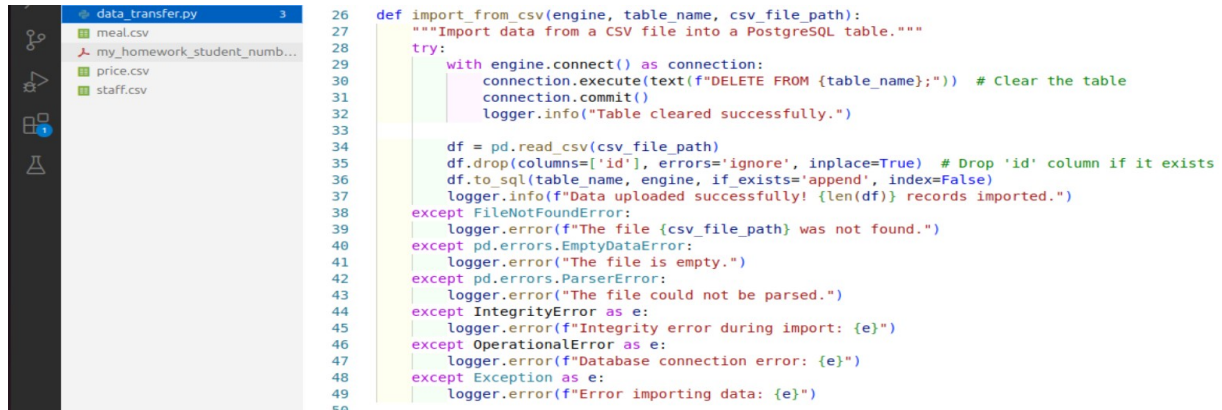
日期	星期	早餐	午餐	下午茶	晚餐
2025年01月1日	星期三	燕麥粥配水果	烤雞沙拉	草本茶與餅乾	烤魚配蔬菜
2025年01月2日	星期四	炒蛋	蔬菜湯	蜂蜜優格	炒豆腐配米飯
2025年01月3日	星期五	全麥吐司	藜麥沙拉	新鮮水果	扁豆燉菜
2025年01月4日	星期六	香蕉菠菜奶昔	火雞三明治	餅乾與起司	烤雞配地瓜
2025年01月5日	星期日	堅果粥	菠菜與費塔起司沙拉	綠茶與鬆餅	蔬菜咖哩配米飯
2025年01月6日	星期一	優格配格蘭諾拉	烤蔬菜捲	水果沙拉	牛肉燉菜

Source Code:

get\_db\_connection() export\_to\_csv()

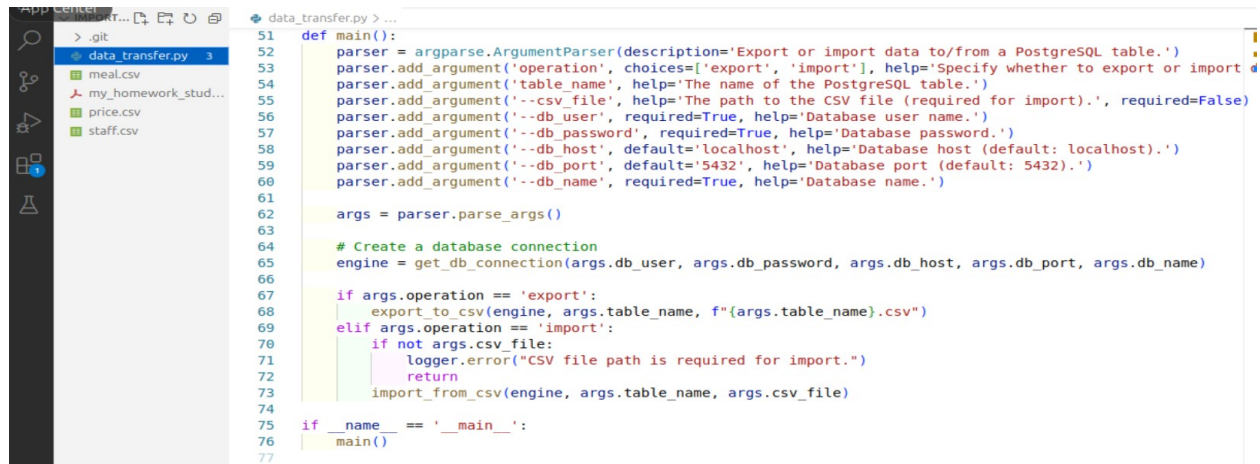
```
data_transfer.py - import_export_homework - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  IMPORT_EXPORT_...
    .git
    data_transfer.py
    meal.csv
    my_homework_student_numb...
    price.csv
    staff.csv
  data_transfer.py
1  import os
2  import pandas as pd
3  from sqlalchemy import create_engine, text
4  import logging
5  import argparse
6  from sqlalchemy.exc import OperationalError, IntegrityError
7
8  # Set up logging
9  logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
10 logger = logging.getLogger(__name__)
11
12 def get_db_connection(db_user, db_password, db_host, db_port, db_name):
13     """Create a database connection using provided parameters."""
14     return create_engine(f'postgresql://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}')
15
16 def export_to_csv(engine, table_name, csv_file_path):
17     """Export data from a PostgreSQL table to a CSV file."""
18     try:
19         df = pd.read_sql_table(table_name, engine)
20         df.drop(columns=['id'], errors='ignore', inplace=True) # Drop 'id' column if it exists
21         df.to_csv(csv_file_path, index=False)
22         logger.info(f"Data exported successfully to {csv_file_path}.")
23     except Exception as e:
24         logger.error(f"Error exporting data: {e}")
25
```

import\_from\_csv()



```
26 def import_from_csv(engine, table_name, csv_file_path):
27     """Import data from a CSV file into a PostgreSQL table."""
28     try:
29         with engine.connect() as connection:
30             connection.execute(text(f"DELETE FROM {table_name};")) # Clear the table
31             connection.commit()
32             logger.info("Table cleared successfully.")
33
34         df = pd.read_csv(csv_file_path)
35         df.drop(columns=['id'], errors='ignore', inplace=True) # Drop 'id' column if it exists
36         df.to_sql(table_name, engine, if_exists='append', index=False)
37         logger.info(f"Data uploaded successfully! {len(df)} records imported.")
38     except FileNotFoundError:
39         logger.error(f"The file {csv_file_path} was not found.")
40     except pd.errors.EmptyDataError:
41         logger.error("The file is empty.")
42     except pd.errors.ParserError:
43         logger.error("The file could not be parsed.")
44     except IntegrityError as e:
45         logger.error(f"Integrity error during import: {e}")
46     except OperationalError as e:
47         logger.error(f"Database connection error: {e}")
48     except Exception as e:
49         logger.error(f"Error importing data: {e}")
50
```

Required parameters imported into main() for import/export



```
51 def main():
52     parser = argparse.ArgumentParser(description='Export or import data to/from a PostgreSQL table.')
53     parser.add_argument('operation', choices=['export', 'import'], help='Specify whether to export or import')
54     parser.add_argument('table_name', help='The name of the PostgreSQL table.')
55     parser.add_argument('--csv_file', help='The path to the CSV file (required for import).', required=False)
56     parser.add_argument('--db_user', required=True, help='Database user name.')
57     parser.add_argument('--db_password', required=True, help='Database password.')
58     parser.add_argument('--db_host', default='localhost', help='Database host (default: localhost).')
59     parser.add_argument('--db_port', default='5432', help='Database port (default: 5432).')
60     parser.add_argument('--db_name', required=True, help='Database name.')
61
62     args = parser.parse_args()
63
64     # Create a database connection
65     engine = get_db_connection(args.db_user, args.db_password, args.db_host, args.db_port, args.db_name)
66
67     if args.operation == 'export':
68         export_to_csv(engine, args.table_name, f"{args.table_name}.csv")
69     elif args.operation == 'import':
70         if not args.csv_file:
71             logger.error("CSV file path is required for import.")
72             return
73         import_from_csv(engine, args.table_name, args.csv_file)
74
75 if __name__ == '__main__':
76     main()
77
```

The program is designed to accept the required parameters without hardcoding the table name and database information. This generic tool can be used to export and import other tables across different PostgreSQL databases.

--- END ---