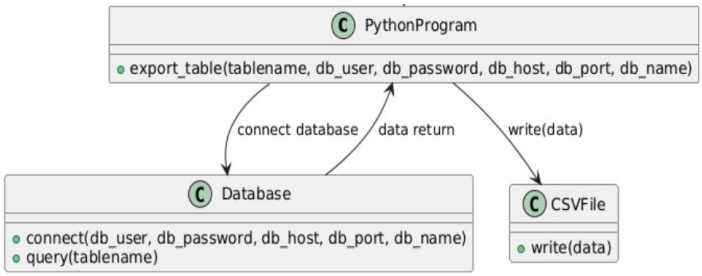
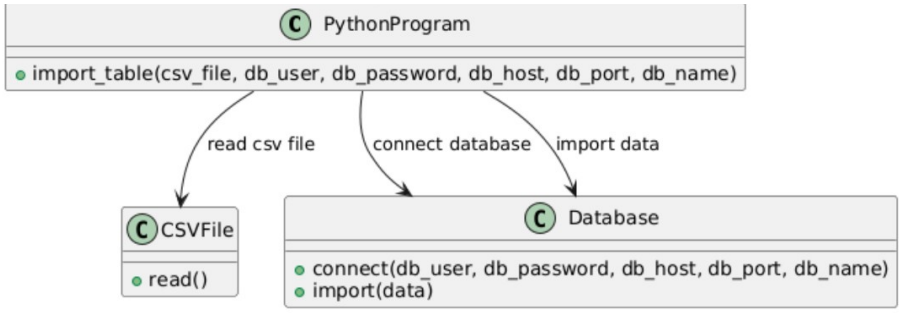
# Student Number: 21
# Author: Davy Hui

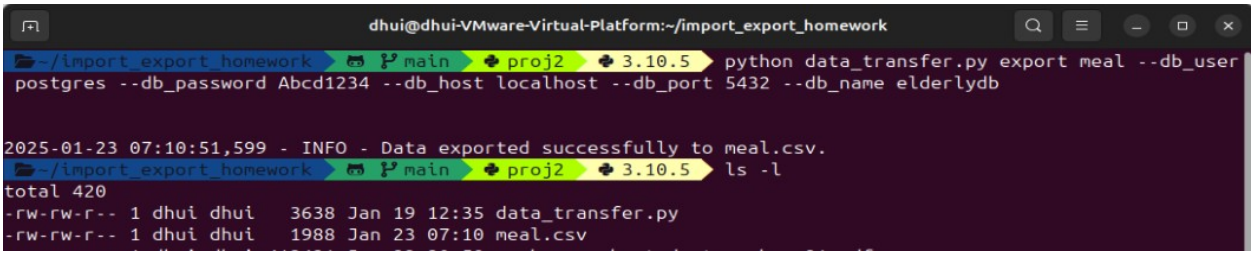
This homework assignment demonstrates how my program, written in Python, exports a Postgres table to a CSV file and then uses the same tool to import it back

Program Name: **data_transfer.py**
Programming Language: **Python**

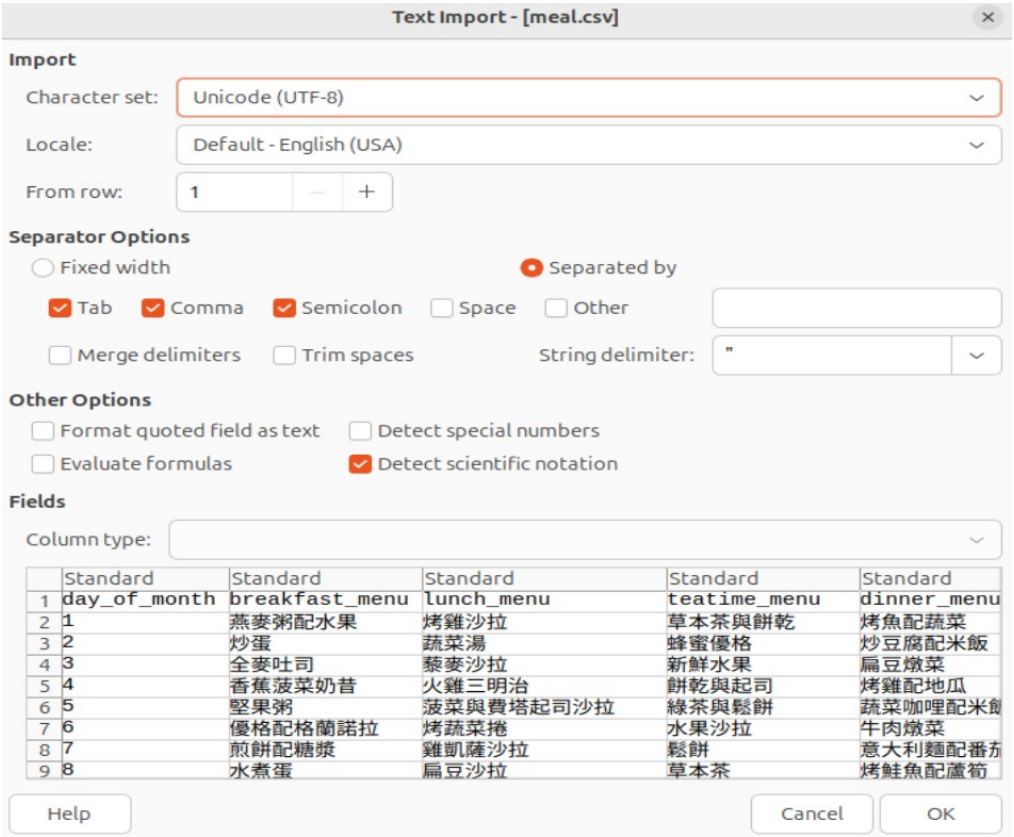
| | |
|---|---|
| Export Table to CSV | Usage:<br><br>**python data_transfer.py export <tablename> --db_user <userid> --db_password <password> --db_host <servername> --db_port <serverport> --db_name <databasename>** |
|  | |
| Import Table from CSV | Usage:<br><br>**python data_transfer.py import <tablename> --csv_file <csv filename> --db_user <userid> --db_password <password> --db_host <servername> --db_port <serverport> --db_name <databasename>** |
|  | |

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).




Step 2. Open CSV file for editing



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

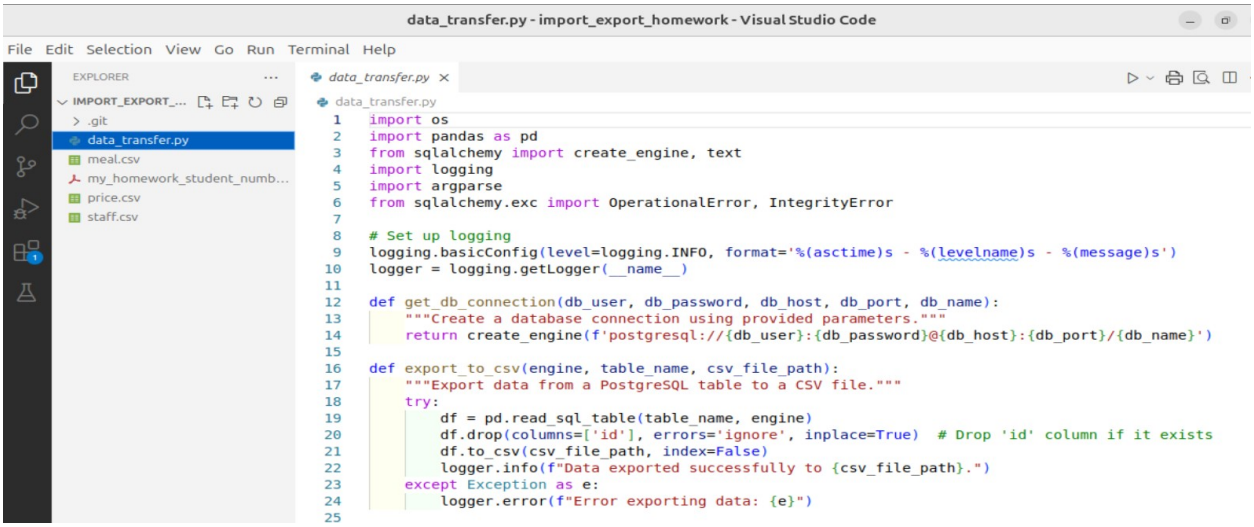
```
dhui@dhui-VMware-Virtual-Platform:~/import_export_homework
~/import_export_homework  main  proj2  3.10.5  python data_transfer.py import meal --csv_file meal.csv --db_user postgres --db_password Abcd1234 --db_host localhost --db_port 5432 --db_name elderlydb

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.


| 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()




```python
import os
import pandas as pd
from sqlalchemy import create_engine, text
import logging
import argparse
from sqlalchemy.exc import OperationalError, IntegrityError

# Set up logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

def get_db_connection(db_user, db_password, db_host, db_port, db_name):
    """Create a database connection using provided parameters."""
    return create_engine(f'postgresql://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}')

def export_to_csv(engine, table_name, csv_file_path):
    """Export data from a PostgreSQL table to a CSV file."""
    try:
        df = pd.read_sql_table(table_name, engine)
        df.drop(columns=['id'], errors='ignore', inplace=True)  # Drop 'id' column if it exists
        df.to_csv(csv_file_path, index=False)
        logger.info(f"Data exported successfully to {csv_file_path}.")
    except Exception as e:
        logger.error(f"Error exporting data: {e}")
```

import_from_csv()

```python
def import_from_csv(engine, table_name, csv_file_path):
    """Import data from a CSV file into a PostgreSQL table."""
    try:
        with engine.connect() as connection:
            connection.execute(text(f"DELETE FROM {table_name};"))  # Clear the table
            connection.commit()
            logger.info("Table cleared successfully.")

        df = pd.read_csv(csv_file_path)
        df.drop(columns=['id'], errors='ignore', inplace=True)  # Drop 'id' column if it exists
        df.to_sql(table_name, engine, if_exists='append', index=False)
        logger.info(f"Data uploaded successfully! {len(df)} records imported.")
    except FileNotFoundError:
        logger.error(f"The file {csv_file_path} was not found.")
    except pd.errors.EmptyDataError:
        logger.error("The file is empty.")
    except pd.errors.ParserError:
        logger.error("The file could not be parsed.")
    except IntegrityError as e:
        logger.error(f"Integrity error during import: {e}")
    except OperationalError as e:
        logger.error(f"Database connection error: {e}")
    except Exception as e:
        logger.error(f"Error importing data: {e}")
```

Required parameters imported into main() for import/export

```python
def main():
    parser = argparse.ArgumentParser(description='Export or import data to/from a PostgreSQL table.')
    parser.add_argument('operation', choices=['export', 'import'], help='Specify whether to export or import d
    parser.add_argument('table_name', help='The name of the PostgreSQL table.')
    parser.add_argument('--csv_file', help='The path to the CSV file (required for import).', required=False)
    parser.add_argument('--db_user', required=True, help='Database user name.')
    parser.add_argument('--db_password', required=True, help='Database password.')
    parser.add_argument('--db_host', default='localhost', help='Database host (default: localhost).')
    parser.add_argument('--db_port', default='5432', help='Database port (default: 5432).')
    parser.add_argument('--db_name', required=True, help='Database name.')

    args = parser.parse_args()

    # Create a database connection
    engine = get_db_connection(args.db_user, args.db_password, args.db_host, args.db_port, args.db_name)

    if args.operation == 'export':
        export_to_csv(engine, args.table_name, f"{args.table_name}.csv")
    elif args.operation == 'import':
        if not args.csv_file:
            logger.error("CSV file path is required for import.")
            return
        import_from_csv(engine, args.table_name, args.csv_file)

if __name__ == '__main__':
    main()
```

**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 ---