SPL211: Assignment 4

Python and SQL

7/01/2021

Responsible TA: Or Dinari
Due Date: 17/1/2021 23:59

1 Introduction

In this assignment you are required to implement the database of a *SARS-CoV-2* vaccine distribution center. The assignment is composed of three parts:

1. Create and populate the database according to a configuration file.
2. Execute a list of orders, according to a second file.
3. Print a summary in a third file.

Note that you **must** use a Persistence Layer, with DTO and DAO, as seen in PS13. Failing to use it will results in a 20 points deduction.

For this assignment you will use *Python 3.6.9* and *sqlite3*, which are included in the *VM*.
For configuration file *config.txt*, orders file *orders.txt* and output file *output.txt*, you will run the code (on the VM) with the following line:

```
> python3 main.py config.txt orders.txt output.txt
```

 Before writing any code, make sure you read the **whole** assignment.

2 Building the Database

You will store the database the file *database.db*.
The database will contain the following tables:

2.1 Structure

- vaccines: Hold the information on the vaccines currently in the inventory.
  - id INTEGER PRIMARY KEY
  - date DATE NOT NULL
  - supplier INTEGER REFERENCES Supplier(id)
  - quantity INTEGER NOT NULL
- suppliers: Holds the suppliers data.

- id INTEGER PRIMARY KEY
- name STRING NOT NULL
- logistic INTEGER REFERENCES Logistic(id)

- clinics: Holds the information on the different clinics.
  - id INTEGER PRIMARY KEY
  - location STRING NOT NULL
  - demand INTEGER NOT NULL
  - logistic INTEGER REFERENCES Logistic(id)
- logistics: Holds the information on the different delivery services.
  - id INTEGER PRIMARY KEY
  - name STRING NOT NULL
  - count_sent INTEGER NOT NULL
  - count_received INTEGER NOT NULL

2.2 Configuration file

In order to build the database, you will parse a configuration file, the file will have the following structure:

```
<#1>,<#2>,<#3>,<#4>
<vaccines>
<suppliers>
<clinics>
<logistics>
```

Where each of the numbers in the first line stands for the number of entries of that type, and each entry has the relevant table details, separated by a comma. For example:

```
3,1,2,2
1,2021-01-10,1,10
2,2021-01-11,1,20
3,2021-01-12,1,20
1,Pfizer,1
1,Beer-Sheva,50,1
2,Tel-Aviv,150,2
1,DHL,0,0
2,UPS,0,0
```

In the above example we have 3 bulks of vaccines in the inventory, 1 supplier, 2 clinics, and 2 logistics services. Note that there are no spaces, just comas and new lines.

3 Orders

You will support two type of orders:

Receive Shipment This order will receive a shipment from one of the available suppliers:

```
<name>,<amount>,<date>
```

This order will add $< amount >$ vaccines from $< name >$ on $< date >$. Executing the order will add the relevant item to the Vaccines table, and update the *count_received* on the relevant supplier. For example, using the previous example tables:

```
Pfizer,20,2021-01-02
```

Will add an entry in the *Vaccines* table, with *4,2021-01-02,1,20*, and increase the *count_received* by 20 on the *DHL* entry.

Send Shipment This order will send a shipment from the distribution center to one of the clinics:

```
<location>,<amount>
```

This order will remove $< amount >$ from the demand of $< location >$, in addition, it will remove the sum of $< amount >$ from the inventory, if the quantity of an entry in the Vaccines table reduce to zero, that entry should be removed from the table. Note that if $< amount >$ is larger than a single entry, the order will affect multiple entires. Older vaccines will be shipped prior to newer ones. In a similar way to the previous order, we will update the relevant logistic service *count_sent* with the added $< amount >$. For example, using the previous example tables:

```
Tel-Aviv,50
```

Will reduce the demand in Tel-Aviv by 50, remove all three entries from the *Vaccines* table, and increase the *count_sent* by 50 on the ~~DHL~~UPS entry.

3.1 Executing the Orders

The orders will be read from the specified file, and executed in the order they appear in the file, each line will be a single order. For example:

```
Pfizer,20,2021-01-02
Pfizer,100,2021-01-10
Tel-Aviv,40
```

Will first receive the two shipments from Pfizer, then send a single shipment to Tel-Aviv.

4 Summary File

After each order a line will be added to the summary, the line should include:

```
<total_inventory>,<total_demand>,<total_received>,<total_sent>
```

For executing the 3 orders above, using the previous table, you will have an output file of 3 lines:

```
70,200,20,0
170,200,120,0
130,160,120,40
```

5 Submission And Testing

5.1 Testing

The assignment will be tested automatically **on the VM**, we will check **both** the output file, and the database file after finishing the execution. You **MUST** use exactly the specification above, for both the output file and the database. Failing to do so will result in the automatic tests failing, we will not allow appeals on such case.

You are supplied with an example input, and expected outputs (database and output file).

5.2 Submission

Your submission should include a *main.py* file which we will use to run the assignment (according to 1). You may add additional *.py* files, everything should be zipped to ID1_ID2.zip

 Make sure you **commit** the changes to the database at the end of the code execution! We will check both the database and the output file.