2nd Laboratory Exercise

Software Development Tools
& Systems Programming

Eleftheriadis Dimitrios 2015030067

computeSales.py

In the laboratory exercise, the aim is the implementation of a program, which will accept as input evidence files by checking the correctness and printing statistics, in python.

SQLite Option Explanation

After much searching, we ended up using the sqlite library provided by python. The reason was because sqlite was written in C, offering much faster speeds in searches, and even in storing information and even huge files.

Also, sqlite provides the ability to temporarily store the entire database in memory, which makes it even faster than read / write to disk.

For example for a text file containing 13,217,667 lines, our program takes just 46 seconds * to read, check the accuracy and store the information in the database. Recovering and displaying data from the Database takes only fractions of a second.

The program should run with the sudo command to list the necessary permissions for the database, however depending on the rights of each user may not be necessary.

* This result came as an average of repeated code tests on a Dell laptop with an Intel (R) Core (TM) i7-6500U CPU @ 2.50GHz and 7860MB of total memory.

Program Functionality

Initially, the menu was implemented with 4 options: 1 for reading a receipt file, 2 for the statistics of a product, 3 for the statistics of a VAT (ΑΦΜ) number and 4 for exit, in a while loop.
Then we implemented the 3 functions for the 3 options.

To read the receipt file:

We ask for the name of the file in the function and with the command with we open the file for reading. By reading line-by-line we check each time, its correctness and consequently of the respective proof. When we find the line that refers to the VAT number, we take the VAT number and pass it through the control (it will be analyzed below). So we continue for the whole receipt and find for each product the name, the quantity, the price, the total and finally the bill total price. Then we check if we meet the receiver of the receipt and we know if we changed the receipt. We pass all these data to a temporary table and then if all the data are correct we pass them to the main table where we store all the correct receipts.

We created these tables, as the implementation of the other 2 options was done through sqlite and we created a local database. So we created a function for the connection to the local database, two functions that create the tables (one for the temp table and one for the bill table), two functions to insert the tables and a function to delete the elements from the temp table.

To print product statistics:

As mentioned above this query was implemented with sqlite. Below is the code:

```
85  def second_question(conn, pr):
86      cur = conn.cursor()
87      cur.execute("SELECT AFM, SUM(total) FROM bills WHERE product_name=? GROUP BY AFM, product_name",(pr,) )
88      pn=cur.fetchall()
89      for row in pn:
90              t=round(row[1],2)
91              afm=str(row[0])
92              print(afm,t)
```

To print the statistics of a VAT number:
In the same logic as the previous one, this option was implemented. The code is given:

```
94  def third_question(conn, afm):
95      cur = conn.cursor()
96      cur.execute("SELECT product_name, SUM(total) FROM bills WHERE AFM=? GROUP BY product_name",(afm,) )
97      pn=cur.fetchall()
98      for row in pn:
99              t=round(row[1],2)
100             print(row[0],t)
101
102
```

Checks

For the implementation of the program several checks had to be made. The first test is the VAT number, ie if it exists and if it is 10 digits. It was also checked whether the total price of a product in proportion to the quantity, but also the final price of the receipt is correct.
We also check if a new receipt starts, where we take the necessary measures, resetting some auxiliary variables and emptying the temp table. It is worth noting that if an error is found at any point in the receipt, then the program omits all rows

of this receipt until it finds the new one.

Our program is not case-sensitive using the upper () function when we read the user input for the product they want to search for.

The statistics are also printed in ascending order via group by.

We have also used exceptions where needed. An example is in the case that the price of the product is missing in a line of proof, then we catch this exception, and set the proof as wrong.

Conclusions

In conclusion, python is one of the most user-friendly languages for file handling and in combination with sqlite's built-in library the query results were fast and consistent.