## horizontal line



Create your own dataset

31/05/2018

**─ Part 2**

Βούλγαρη Ελένη

Καρύδης Αθανάσιος

Κωνσταντοπούλου Βασιλική

Μυλωνάκη Αγγελική

.

# Introduction

For this assignment we have created a database schema which stores information regarding rooms for rent in the context of a mobile/Web application.

In the first section, instructions for the reproduction of the dataset can be found. In the second section, the database schema is presented, along with the tables it contains and the methods of filling their record values to produce the resulting dataset. In the third section, the database schema in UML is shown. The fourth section describes the ways the erroneous values were inserted in the dataset and the fifth and last section lists a number of query and machine learning tasks that can be applied to the produced dataset.

The related repository that contains our python implementation in addition to the produced CSVs can be accessed via [this link](https://github.com/geloumil/BigDataMining2).

# Reproduction instructions

1. Click on the link above
2. Click on the Clone or Download button
3. Copy the Clone with HTTPS link
4. Open terminal, write git clone and paste the link you have copied on step 4
5. Go to the folder which has been created
6. Write “ls” and you will see the “DataMining2.py” file
7. Type the command “python DataMining2.py” to run the python script
8. Three files will be created (Person.csv, Room.csv, reservations.csv)
9. These files combined are the complete dataset

Important: Python and PostgreSQL must be supported to run the .py executable. In case of rerunning the executable, the user must DROP tables from postgreSQL.

*It is advised to run the python executable in the Virtual Machine given in the start of the MSc program.*

# Database Schema

We have enhanced our tables so as to better apply to the exercise’s requirements provided in class.

As a result:

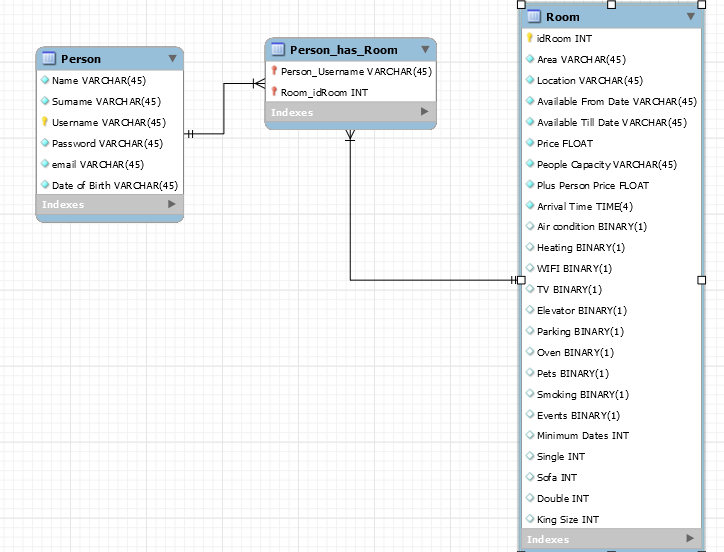
* Table **Person** contains 88000 entries
* Table **Room** contains 6000 entries
* Table **Reservations** contains 1.000.000 entries, combining the above two tables

|  |  |
| --- | --- |
| **Person** | |
| *Attribute* | *Data Generation Method* |
| **Name** | Import from Csv file |
| **Surname** |
| **Username** |
| **Password** | Random password generation via Website |
| **e-mail** | Combining first five characters from Name and Surname columns |
| **Date of Birth** | Auto Generation using python  (\*\*Only allowing adults) |

|  |  |
| --- | --- |
| **Room** | |
| *Attribute* | *Data Generation Method* |
| **IdRoom** | Serial Number Sequence |
| **Location** | Import from CSV file |
| **Available from Date** | Auto Generation using python |
| **Available till Date** | Auto Generation using python |
| **Price** | Uniform sampling with respect to Area |
| **Area** | Uniform sampling |
| **People Capacity** | Uniform sampling with respect to Area |
| **Plus Person Price** | Uniform sampling |
| **Arrival Time** | Random selection from a list using Python |
| **Single** | Uniform sampling with respect to People Capacity |
| **Sofa** |
| **Double** |
| **King Size** |
| **Pets** | Using Python Bernoulli function |
| **Smoking** |
| **Events** |
| **Minimum Dates** | Random selection in Python |
| **Air Condition** | Using Python Bernoulli function |
| **Heating** |
| **Wi-Fi** |
| **TV** |
| **Elevator** |
| **Parking** |
| **Oven** |

|  |  |
| --- | --- |
| **Reservations** | |
| *Attribute* | *Data Generation Method* |
| **Person\_id** | Randomly connecting elements from Room and Person |
| **Room\_id** |

# Database Schema (UML)

­

# Methods for erroneous record generation

In order to explore the ways ‘noise’ can appear in a given Dataset, we are using multiple ways of introducing the following errors in our records:

* noise, for example beyond bounds values of records
* inconsistent entries
* missing values
* spelling mistakes
* duplicate entries

The methods we followed to introduce the faulty behavior mentioned above are:

* Introduce spelling mistakes in CSV or in the database using python in a random subset of records.
* Duplicate email addresses, since they are constructed using the first characters of attributes “name” and “surname”.
* Randomly introduce missing values in different features following the relevant pattern:
  + in “facilities” every 2000 records
  + In “capacity” every 1000 records
  + In “price”, every 190 entries
  + In area, every 1002 entries
* Randomly introduce outliers
  + in “beds”, we introduce an outlier (20 beds) almost every 400 records
  + in “pets” section, although it can only take two values (bernoulli), we set a third value every almost 500 entries
  + In “plus\_prize”, every 250 entries put an outlier value
  + In “capacity”, every 160 entries we put an outlier value
  + In “price”, every 125 entries we put an outlier value
  + In “area” every 98 entries, we put a zero value
* Use Standard Distributions to set the value of our Attributes (e.g Uniform, Binomial,Bernoulli)
* Database Design not following a Normal Form
* Table “reservations” can contain duplicate entries since we randomly correlate items from Person and Room tables

|  |  |  |
| --- | --- | --- |
| a/a | Wrongs % | Missing\_Values % |
| Area | 1.01 | 0.08 |
| Price | 0.8 | 0.51 |
| Capacity | 0.61 | 0.1 |
| Plus\_Person | 0.4 | 0.63 |
| Beds | 0.25 | 3.83 |
| Rules | 0.23 | 0 |
| Facilities | 0.05 | 0 |

# Tasks

The constructed DataSet can be explored in two ways. To begin with, since we have created a Database schema, we could execute queries to retrieve valuable information.

The following set of queries apply:

1. Name the people who have stayed in only one room.
2. Find the rooms that have Wi-Fi and don’t have an oven.
3. Find the people over 40 years old, that have stayed in a Room in Thessaloniki.
4. Find the rooms that can be booked for at least a month and are located in Athens, having price less than 30 euro a day.
5. Find all available rooms from January to March, that are located in Katerini or Lamia.
6. Count the rooms that have a King Size bed, allow pets and a specific person has stayed in them before.
7. Find all the rooms a specific person has stayed in the past and count the total days of his stay.
8. Find all the rooms that allow Events and have a Plus person price less than 20 euros. Order them according to the amount of visitors they have had in the past and select the most popular.
9. Group the rooms by their bed facilitation (single,sofa,double,king size), count them and find the most frequent pattern (e.g. single and a sofa or two singles).
10. Find the people that only stay in hotels for smokers.

In addition to the queries above, we could use the Dataset to train and test machine learning algorithms. By this approach, we can retrieve information that is not apparent from the dataset, but can be ‘exported’ by following a smarter path.

The following Machine Learning tasks apply:

1. Create and find groups of people that have the similar preferences on the way they spend their vacation.
2. Find the most common type of room that people prefer in each place, given their family status. This way someone, who wants to construct a new accomodation facility, can decide what type of room he is more likely to have the most gain from.
3. Examine the features “beds”, ”people capacity” and “events” to find the most social people.
4. Compare similar groups of rooms, given their characteristics, to predict the price that should be assigned to a newly created accomodation facility.