## horizontal line



Create your own dataset

26/03/2018

**─ Part 2**

Βούλγαρη Ελένη

Καρύδης Αθανάσιος

Κωνσταντοπούλου Βασιλική

Μυλωνάκη Αγγελική

.

# Introduction

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

Below are the tables included in our database schema in addition to the proposed ways of filling their record values to produce the resulting Dataset.

# Database Schema

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

As a result:

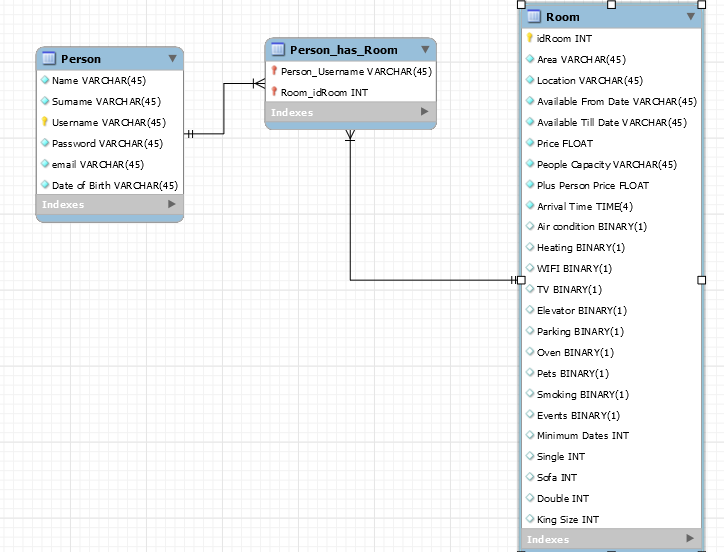
* Table **Person** contains 88000 entries
* Table **Room** contains 6000 entries
* Table **Reservations** contains 1.000.000 entries, combining the above two tables

|  |  |
| --- | --- |
| **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** | Following Gaussian distribution |
| **Area** | Import from CSV file |
| **People Capacity** | Random selection from list of values in Python |
| **Plus Person Price** | Following Gaussian distribution |
| **Arrival Time** | Auto Generation using python |
| **Single** | Random selection in Python |
| **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 that 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 to be followed to introduce the faulty behavior mentioned above will be:

* 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”
* Introduce missing values in different features
* Mix attribute values, which means filling an attribute value with an unexpected value, while expecting a different one
* Randomly introduce outliers
* Use Standard Distributions to set the value of our Attributes (e.g Gaussian, 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
* Create controlled errors in the Rooms dataset like missing or wrong values with python.

|  |  |  |
| --- | --- | --- |
| a/a | Wrongs / 6000 | Missing\_Values / 6000 |
| Area | 61 | 5 |
| Price | 48 | 31 |
| Capacity | 37 | 6 |
| Plus\_Person | 24 | 38 |
| Beds | 15 | 230 |
| Rules | 0 | 12 |
| Facilities | 3 | 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 that exist in the tables.

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 that a specific person has stayed before
7. Find all the rooms a specific person has stayed in the past and counting the total days he stayed there
8. Find all the rooms that allow Events and have Plus person size less than 20 euros. Order them counting the amount of visitors that they have had and select the more popular one
9. Group the rooms by their bed facilitation (single,sofa,double,king size), count them and find the most frequent pattern (eg 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 actually present in the database, but can be found 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. In this way, someone that wants to construct a new accomodation facility could use this to decide what type of room is more likely to have the most gains
3. Examine the features “beds”,”people capacity” and “events” to find the most social people.
4. When creating a new accomodation facility, predict the price that should be assigned to it, given the characteristics it has, compared to similar groups of rooms.