**Backend**

**Developer Exercise**

****

Thank you in advance for the time and efforts you will invest in this task.

**We are TeamScale** - TeamScale founded in 2011, as an offshore service MNC company, with offices in Gurgaon, Israel and Canada. We are operating for the last 1.5 years our Indian studio for international high prestigious projects with our Partner GeoSim.

**Few Words about Ayyeka**:

Ayyeka, an Industrial Internet of Things (IIoT) technology company, was founded in 2011 to simplify the process of delivering data from remote infrastructure and dispersed assets to decision makers. Setting up smart infrastructure networks is streamlined and secure using Ayyeka's hardware-enabled Software-as-a-Service (SaaS) solutions, which collect data from any sensor, over any communication network, using any server, and integrate into any SCADA or software platform. As the future is increasingly connected and data driven, Ayyeka is at the cutting-edge of IIoT leading in tactical edge analytics, low-power connectivity, and advanced sensing.

<https://www.ayyeka.com/>

We will be happy to have you with us.

**About the Task**

Test would need around **3-4 hour to be completed however we are giving you 6 hours** so that you can complete the test and get a nice look and feel of the same.

This exercise provides a small SQLite database with some data derived from the 1996 US Census and a few questions related to working with SQL and open source analysis packages.

Some guidance

1. Use open source languages and tools, such as C# or Python.
2. Clone this repository to your computer. Work in your clone of it, and when you're done, send us a tarball, zip file, or link to your repo online.
3. Use the Internet as a resource to help you complete your work. We do it all the time.
4. Don't comment your code! The code has to be self-explaining.
5. There are many ways to approach and solve the problems presented in this exercise.
6. Have fun!
7. Don't spend more than about 3 hours max on this. You have to complete all the tasks listed below.
8. Add basic logging and error handling to your project.

Google will point you to popular libraries for connecting to SQLite databases from Python and C#.

**The Task**

There are many things you can do with this dataset. Here are a few structured tasks to attempt:

1. Read the section below about The Data.
2. Write a SQL query that creates a consolidated dataset from the normalized tables in the database. In other words, write a SQL query that "flattens" the database to a single table. Use LEFT, RIGHT JOIN if needed.
3. Export the "flattened" table to a CSV file.
4. Create a simple REST API server to provide the "flattened" data to the client.
5. The server has to read the data from the CSV file on start and put it into data structures for data providing.
6. The server has to support the GET and SET operations on the data objects. Use your imagination to parameterize the REST API. For example, you can add parameter to limit the maximal number of the returned records, or add some filtering.

**Note:** Add a few words in a text file, about the project. What you did, how you came up with a solution to the problem.

**The Data**

This repository contains a file called exercise01.sqlite. It is a normalized relational [SQLite database](http://www.sqlite.org/).

It contains a table, named records, that has 48842 US Census records with the following fields:

* id: a unique id number for each record
* age: a continuous variable representing an individual's age
* work class id: foreign key to the work classes table, representing the broad class of occupation of an individual
* education level id: foreign key to the education levels table, representing the highest level of education an individual received
* education num: a continuous variable representing an individual's current education level
* marital status id: foreign key to the marital statuses table, representing an individual's marital status
* occupation id: foreign key to the occupations table, representing an individual's occupation
* race id: foreign key to the races table, representing an individual's race
* sex id: foreign key to the sexes table, representing an individual's sex
* Capital gain: a continuous variable representing post-social insurance income, in the form of capital gains.
* Capital loss: a continuous variable representing post-social insurance losses, in the form of capital losses.
* Hours week: a continuous variable representing the number of hours per week an individual worked.
* Country id: foreign key to the countries table, representing an individual's native country
* over\_50k: a Boolean variable representing whether the individual makes over $50,000/year. A value of 1 means that the person makes greater than $50,000/year and a value of 0 means that the person makes less than or equal to $50,000/year.

Inspection of the database will reveal the reference tables and the values that they contain, referenced by the foreign keys in the categorical fields of the records table. Basically, anywhere you see a field name above that ends with \_id, there is a corresponding table in the database that contains the values associated with that categorical variable. Fields that contain continuous values, such as age, do not join to other tables.

Some of the reference tables have an entry for a question mark ? that represents missing data in records.

Good Luck