

CSPro2sql & Dashboard

USER GUIDE

Document Information

Settings	Value
Document Title:	CSPPro2sql & Dashboard USER GUIDE
Project Component/Activity	C3 – Setting up a system for data collection monitoring
Document Authors:	Mauro BRUNO, Guido DROVANDI, Paolo GIACOMI, Milena GRASSIA
Project Manager:	Milena GRASSIA
Doc. Version:	1.0
Approval Status	Final
Sensitivity:	PUBLIC
Date:	16/06/2018

Configuration Management: Document Location

The document is part of the Final Project Report submitted to the Italian Agency for Development Cooperation (AICS) for approval.

The final version of this document will be stored on the Istat web page.

Index

1. INTRODUCTION	4
1.1. General framework	4
1.2. Purpose of the document	5
2. SYSTEM ARCHITECTURE	6
2.1. Data collection process	6
2.2. Data collection architecture	7
3. INSTALLING AND USING THE SYSTEM	8
3.1. CSPro2sql	8
3.2. CSPro Dashboard	11
3.3. CSPro dictionary metadata	14
3.4. Administrative territorial structure	15
LIST OF ACRONYMS	17
ANNEXES	18
Annex 1 CSPro Dashboard: step by step installation guide	19
Annex 2 CSPro Dashboard: scheduler configuration	29
Annex 3 CSPro Dashboard: report implementation guide	41
Annex 4 CSPro Dashboard: technologies and frameworks	45

1. INTRODUCTION

1.1. General framework

This document relies on the work carried out under Component 2 “Setting up a system for data collection monitoring” of the project in Ethiopia “Capacity building in statistics – Population Census” funded by the Italian Agency for Development Cooperation (AICS) and implemented by the Italian National Institute of Statistics (Istat), with the aim to reinforce the statistical capacity of the Central Statistical Agency of Ethiopia (CSA), responsible for conducting the 4th Population and Housing Census (EPHC).

Project overview

Title	Capacity building in statistics – Population Census		
Financing Agency	AICS, the Italian Agency for Development Cooperation	Implementing Agency	Istat, the Italian National Institute of Statistics
Partner Institution	CSA, the Central Statistical Agency of Ethiopia		
Starting data	7 June 2016	Duration	24 months
Overall objective	To contribute to the improvement of the statistical information available in Ethiopia through the implementation of the 4th Population and Housing Census that will make available updated information on the structure and composition of the Ethiopian population		
Main expected results/ Components	<ul style="list-style-type: none"> - To reinforce the statistical capacity of CSA through the increase of the average professional level and qualifications of the staff involved in the PHC preparation (Technical assistance and training for improving the Census Methodology, setting up a System for data collection monitoring, Data center configuration) - To improve the capacity of the CSA staff involved in the census data quality analysis (Technical assistance and training for designing the Post Enumeration Survey – PES) 		
Coordination with other Development partners/ Implementing agencies	<ul style="list-style-type: none"> - UNFPA - DIFID/ONS - USAID/US Census Bureau 		

4th EPHC - Key features and main innovations introduced

Mapping	Fully digital maps using GPS-enabled handheld devices (GIS technology)
	~ 150.000 Enumeration Areas ~ 36,700 Supervision Areas
Primary data collection	Paperless, first time using CAPI technique
	Designed and managed in CPro, the software package developed by the US Census Bureau - USCB
	CPro software on Android Tablets
	~ 180.000 Android Tablets
Field work quality	Web based monitoring system (CPro Dashboard, designed and implemented with the Italian support) fully integrated (via CPro2sql software) with the primary data collection system
	Real time reporting

1.2. Purpose of the document

This document is to be used as a guide to understand the architecture designed and implemented for the census data collection process, further the document describes the steps to install the software developed to enable the real-time monitoring of the survey progress (CPro2sql and CPro Dashboard).

The document also includes written and visual information to assist CSA staff (and other potential users) in completing tasks associated with the implementation of new reports in the CPro Dashboard.

This Guide summarizes and integrates all technical documents and recommendations issued during the project implementation period as output of the single missions of Istat IT experts to CSA.

2. SYSTEM ARCHITECTURE

As part of the project, a generalized metadata driven monitoring system was implemented, integrated with the CSPro data collection system, developed by the US Census Bureau.

2.1. Data collection process

In order to implement a generalized system to support census fieldwork activities, it is necessary to split the data collection process into the following phases (shown in Figure 1):

- **Primary data collection:** the enumerators collect data on tablets or smartphones using CSPro. Data are stored on the devices in a *plain text* format. Later, using CSPro sync functionality, the enumerators transfer the plain text files over the internet to a central database (CSWeb). This system allows data collection in offline mode in areas where the network is not available or is unreliable.
- **Data transformation:** CSWeb stores each *plain text* file, containing questionnaire microdata, in a table field. This means that, as the number of returned questionnaires increases, it becomes difficult to extract and elaborate real-time information from the database. In order to increase efficiency, it is necessary to store microdata in a more structured way (i.e. each variable in a separate column). The data transformation (from plain text to microdata in separate columns) is performed by the software component CSPro2Sql (described in the following section).
- **Field work monitoring:** data stored in the central database can be used to monitor the progress of the enumeration activities, identify which areas have already been covered and produce real time reports on population structure. In order to display reports, a web application (Dashboard) was developed. Moreover, the reports have been integrated with GIS maps. The link with the GIS data allows display of the status of the data collection on maps, so that the fieldwork process can be visually monitored.

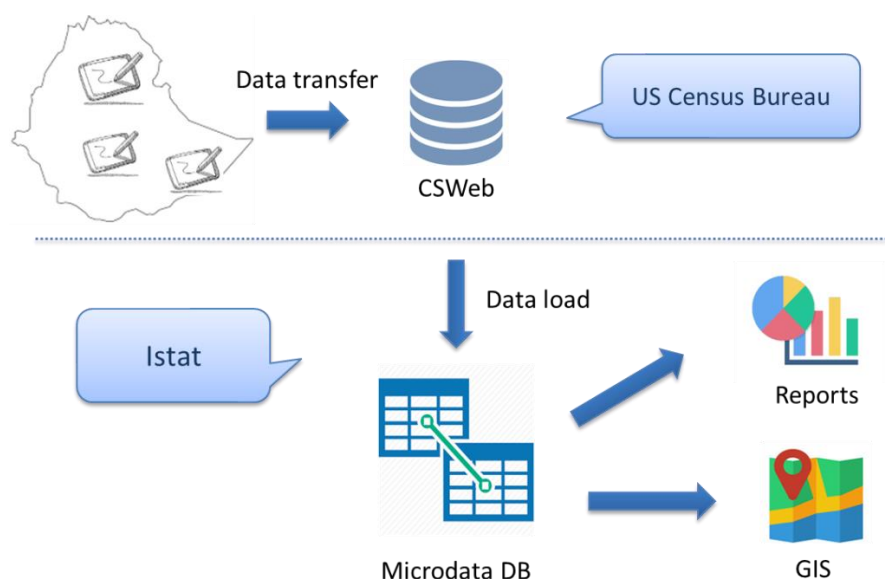


Figure 1: Data collection process

2.2. Data collection architecture

In order to implement the data collection process described above, it is necessary to design a metadata driven architecture, which allows generation of the microdata database and the dashboard reports, parsing the questionnaire metadata (CSPRO Dictionary). The software components needed, are described below:

- **CSPRO data dictionary:** the data dictionary contains all the questionnaire metadata (e.g. variables, classifications, relations between variables). CSPRO generates a data dictionary file for each questionnaire.
- **CSPRO2sql:** this software component has a key role in the integrated architecture. More specifically CSPRO2sql, parsing the content of the data dictionary, generates the scripts to create the microdata database and to load microdata collected using CSPRO. Further, CSPRO2sql offers functionalities to generate the several types of reports (e.g. fieldwork monitoring, age distribution, sex distribution), displayed by the Dashboard web application.
- **Dashboard:** this component is a web application implemented using open source Java frameworks. The current Dashboard displays the reports generated by CSPRO2sql. Moreover, the dashboard offers the possibility to integrate GIS maps.

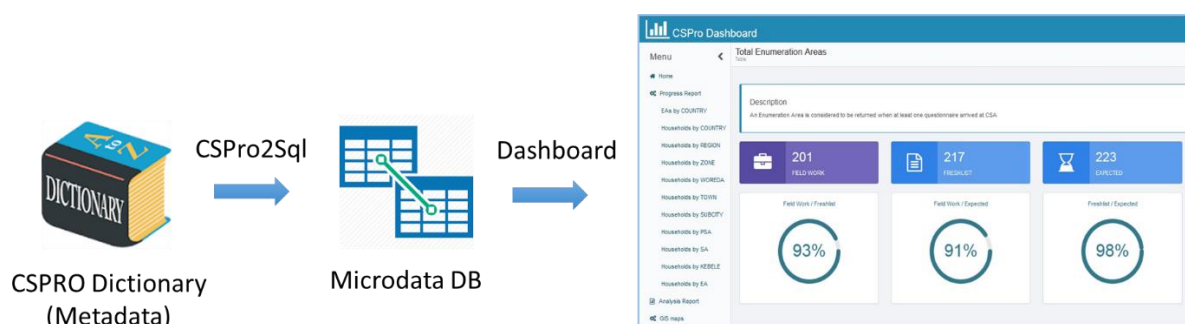


Figure 2 - Data collection architecture

3. INSTALLING AND USING THE SYSTEM

3.1. CPro2Sql

CPro2Sql is a Java application to migrate questionnaires from CPro 7.0 database (CWeb) to a MySQL database (MicrodataDB). The MicrodataDB will contain the microdata, i.e. a column per each variable (Item) defined in the CPro-Dictionary, and the report tables.

A. REQUIREMENTS

In order to install and run CPro2Sql the following environment should be set up:

- ✓ JAVA 1.7+
- ✓ MYSQL 5.7

Further, you need the following libraries:

- ✓ Apache Commons CLI (commons-cli-1.3.1.jar)
- ✓ MySQL Connector/J 5.1.40 (mysql-connector-java-5.1.40-bin.jar)

B. INSTALLATION

CPro2Sql is simple to install. All you need is to download and unzip the CPro2Sql.zip, available on github (<https://github.com/mauroIstat/CPro2Sql>). Depending on your system, execute from the command line CPro2Sql.bat or CPro2Sql.sh.

C. USAGE

CPro2Sql is composed of several engines (run CPro2Sql to get usage info):

```
CPro2Sql -e schema -p PROPERTIES_FILE [-fk] [-o OUTPUT_FILE]
CPro2Sql -e loader -p PROPERTIES_FILE [-a] [-cc] [-co] [-f|-r]
CPro2Sql -e monitor -p PROPERTIES_FILE [-o OUTPUT_FILE]
CPro2Sql -e update -p PROPERTIES_FILE
CPro2Sql -e status -p PROPERTIES_FILE
```

Each engine has a specific purpose, more precisely:

- **schema**: to create the microdata MySQL script (MicrodataDB);
- **loader**: to transfer data from CWeb to MicrodataDB;
- **monitor**: to create the dashboard MySQL script;
- **update**: to update the dashboard report data;
- **status**: to check the loader engine status.

A set of parameters can be provided to CPro2sql engines:

-a,--all	transfer all the questionnaires
-cc,--check-constraints	perform constraints check
-co,--check-only	perform only constraints check (no data transfer)
-e,--engine <arg>	select engine: [loader schema monitor update status]
-f,--force	skip check of loader multiple running instances
-fk,--foreign-keys	create foreign keys to value sets
-h,--help	display this help
-o,--output <arg>	name of the output file
-p,--properties <arg>	properties file
-r,--recovery	recover a broken session of the loader
-v,--version	print the version of the program

D. CONFIGURATION

In order to run CPro2Sql engines it is necessary to configure a properties file. Such file must contain the following properties:

```
db.source.uri: CPro 7.0 database connection string
db.source.schema: CPro 7.0 database schema
db.source.username: CPro 7.0 database username
db.source.password: CPro 7.0 database password
db.source.data.table: CPro 7.0 table containing questionnaires plain data

db.dest.uri: microdata MySQL connection string;
db.dest.schema: microdata MySQL schema;
db.dest.username: microdata MySQL username;
db.dest.password: microdata MySQL password;
db.dest.table.prefix: microdata MySQL table prefix
```

Within this configuration, CPro2Sql reads the CPro dictionaries from CPro 7.0 database. It is also possible to specify a CPro dictionary files:

```
dictionary.filename: the path to the CPro-Dictionary file
```

Note: the source CPro 7.0 database and the microdata MySQL could be the same

E. EXAMPLE

We assume that a user generated a Household questionnaire with CSPro 7.0. and that the questionnaire metadata are stored in the dictionary household_dict. The purpose of the example is to describe the properties file and the steps needed to move the microdata from CSWeb to the MicrodataDB.

Example of properties file (eg. Household.properties):

```
# Source CSPro database
db.source.uri=jdbc:mysql://localhost:3306
db.source.schema=CSPro
db.source.username=srcUsername
db.source.password=srcPassword
db.source.data.table=household_dict

# Destination microdata MySQL
db.dest.uri=jdbc:mysql://localhost:3306
db.dest.schema=CSPro_microdata
db.dest.username=dstUsername
db.dest.password=dstPassword
db.dest.table.prefix=h
```

Execution steps:

We assume that your working path is WORKING_PATH and that the properties file is stored in your WORKING_PATH.

In order to create a microdata MySQL database (CSPro_microdata) and load in this database the data stored into the CSPro 7.0 database (CSPro) execute the following steps:

```
> CSPro2Sql -e schema -p Household.properties -o WORKING_PATH\microdata.sql
> mysql -u dstUsername -p < WORKING_PATH\microdata.sql
> CSPro2Sql -e loader -p Household.properties -cc
```

To monitor the loader activity run:

```
> CSPro2Sql -e status -p Household.properties
```

3.2. CSPro Dashboard

CSPro Dashboard is an open-source Java Spring-based web application built on the database created using CSPro2Sql. The web application can be easily configured in order to provide out-of-the-box:

- ✓ Authentication & authorization;
- ✓ Responsive graphical interface (html, css, js):
 - Report tables with enhanced interaction controls (search, export, sorting, etc.);
 - Report charts (bar, pie, doughnut);
- ✓ CSPro2Sql reports:
 - Status of the data transfer process (RUNNING/COMPLETED, number of records transferred)
 - Errors in the data transfer process (number of errors, details on each error, etc.)
- ✓ Server side components:
 - CRUD (insert, delete, update);
 - REST web services

A. REQUIREMENTS

CSPro Dashboard displays the content of report tables generated using CSPro2Sql and provides information on the data transfer process. Therefore, in order to run the CSPro Dashboard you need:

- **CsWeb:** the Mysql database containing CSPro 7.0 plain text files.
- **CSPro2Sql:** you need to perform the execution steps described in the chapter CSPro2Sql. Briefly speaking you should transfer the data from the CSPro 7.0 database to the microdata MySQL database.
- **Dashboard metadata:** CSPro dictionary should contain dashboard metadata. Such metadata are needed to specify which item in the dictionary corresponds to report variables (e.g. age, sex, religion, territory). A detailed description of such metadata is provided in section **3.3 CSPro dictionary metadata**.
- **Administrative territorial structure:** in order to generate reports based on the territorial structure, it is necessary to create a *territory* table (at the moment this step is not automated). A detailed description of the structure and content of the table is provided in section **3.4 Administrative territorial structure**

Further, in order to build the CSPro Dashboard application, your environment should fulfill the following requirements:

- ✓ A favorite text editor or IDE
- ✓ JDK 1.8 or later
- ✓ Maven 3.0+

B. INSTALLATION

B.1 DATABASE

In order to describe the build steps, we assume that your CSPro2Sql property file is the following (e.g. *Household.properties*) and that you have performed the execution steps described above (the example subsection in previous paragraph).

If you have successfully completed these steps, you have a microdata MySQL database *CSPro_microdata* containing the data from the CSPro 7.0 database. Now you are ready to execute the commands that generate the tables used by the CSPro Dashboard. First, you need to specify the following information:

```
table.individual=name of the table containing individual data [#individual]
column.individual.sex=sex column [#sex]
column.individual.age=age column [#age]
column.individual.sex.value.male=male sex value [#male]
column.individual.sex.value.female= male sex value [#female]
column.individual.religion=religion column [#religion]
range.individual.age=age groups #age[0,5,10,15,20,25,30,35,40,45,50,55,60,65,70]
```

In the list at the end of each row, in square brackets, are listed the fields that should be added to the household dictionary, using the 'Note' field. An example is provided below.

```
[Record]
Label=Individual
Name=INDIVIDUAL
RecordTypeValue='I'
Required=No
MaxRecords=999
RecordLen=215
Note=#individual
```

Further, if you need reports based on the administrative territorial structure, metadata on the territorial structure should be provided. For a more detailed list of dictionary metadata check the following section.

The dictionary metadata are used by CSPro2Sql monitor engine in order to generate the report tables. Now you can execute the following commands:

```
> CSPro2Sql -e monitor -p Household.properties -o WORKING_PATH\dashboard.sql
> mysql -u dstUsername -p < WORKING_PATH\dashboard.sql
> CSPro2Sql -e update -p Household.properties -cc
```

The script will populate the USER/ROLES table with two users:

```
Username: admin@dashboard.it
Password: dashboard
Role: ADMIN
Username: guest@dashboard.it
Password: dashboard
Role: GUEST
```

B.2 WEB APPLICATION

Download the source code and open it with your favorite IDE. As a first step check the content of the application.properties file, located in the path Other Sources > src/main/resources:

```
spring.datasource.url = jdbc:mysql://localhost:3306/CSPro_microdata?useSSL=false
spring.datasource.username = dstUsername
spring.datasource.password = dstPassword
```

The properties should match the Destination microdata MySQL properties specified in the Household.properties.

Now you can perform your first build of the application. If the build process ends successfully, you are ready to run the application. The application is built using the open source framework Spring Boot, which generates an executable jar (that can be run from the command line):

```
java -jar CSProdashboard.jar
```

It is also possible to create a war file that can be deployed on a servlet container such as Tomcat. Simply modify the maven build profile to DashboardWar.

3.3. CSPro dictionary metadata

As described in previous section, in order to generate reports (displayed by the dashboard) a set of metadata should be provided. The list of metadata is the following:

```
table.individual=name of the table containing individual data [#individual]
column.individual.sex=sex column [#sex]
column.individual.age=age column [#age]
column.individual.religion=religion column [#religion]
range.individual.age=age groups [#age]
```

Such metadata should be manually added to the household dictionary, using the note field. As an example, a list of editings is provided below:

[Record] Label=Individual Name=INDIVIDUAL RecordTypeValue='I' Required=No MaxRecords=999 RecordLen=215 Note=#individual	[Item] Label=307 Sex Name=P307 Start=100 Len=1 ZeroFill=Yes Note=#sex	[ValueSet] Label=307 Sex Name=P307_VS1 Value=1;Male Note=#male Value=2;Female Note=#female
[Item] Label=309 Religion Name=P309 Start=103 Len=2 ZeroFill=Yes Note=#religion	[Item] Label=308 Age 308 Name=P308 Start=101 Len=2 ZeroFill=Yes Note=#age[0,5,10,15,20,25,30,35,40,45,50,55,60,65,70,80]	

3.4. Administrative territorial structure

The territorial structure should be provided as an external input (territory management will be improved in the next release of the software). Currently the following steps should be performed:

Step 1: add territory metadata to the household dictionary

As described in the following section, metadata are added using the 'Note' field. In order to clarify the structure of the metadata, we assume that the territorial structure is the following:

Region -> Province -> City

This means that at the top level of the territory structure we have *region*, while at the lowest level we have *city*. This structure can be modelled in the dictionary as follows:

[Item] Label=101Region Name=ID101 Start=2 Len=2 ZeroFill=Yes Note=#territory[Region]	[Item] Label=102 Province Name=ID102 Start=4 Len=2 ZeroFill=Yes Note=#territory[Province, ID101]	[Item] Label=103 City Name=ID103 Start=6 Len=2 ZeroFill=Yes Note=#territory[City, ID102]
--	--	--

Step 2: create territory table

The second step is the creation of a territory table. In our example, the structure of the table should be the following:

ID101_NAME	ID101	ID102_NAME	ID102	ID103_NAME	ID103	TERRITORY_CODE
Region 1	01	Province 1	01	City 1	01	010101
Region 1	01	Province 1	01	City 2	02	010102
Region 1	01	Province 2	02	City 1	01	010201
Region 1	01	Province 2	02	City 2	02	010202
Region 2	02	Province 1	01	City 1	01	020101
Region 2	02	Province 1	01	City 2	02	020102
Region 2	02	Province 2	02	City 1	01	020201
Region 2	02	Province 2	02	City 2	02	020202

The corresponding insert table script would be:

```
INSERT INTO `territory`  
(`ID101_NAME`,`ID101`,`ID102_NAME`,`ID102`,`ID103_NAME`,`ID103`,`TERRITORY_CODE`)  
VALUES ('Region 1','01','Province 1','01','City 1','01','010101');  
  
INSERT INTO `territory`  
(`ID101_NAME`,`ID101`,`ID102_NAME`,`ID102`,`ID103_NAME`,`ID103`,`TERRITORY_CODE`)  
VALUES ('Region 1','01','Province 1','01','City 2','02','010102');  
  
INSERT INTO `territory`  
(`ID101_NAME`,`ID101`,`ID102_NAME`,`ID102`,`ID103_NAME`,`ID103`,`TERRITORY_CODE`)  
VALUES ('Region 1','01','Province 2','02','City 1','01','010201');
```

Step 3: update report tables

The last step is quite simple. All you need is to run CSPro2sql to update the reports:

```
F:\Program Files\CsPro2Sql>CsPro2Sql -e update -p pilot3\pilot3.properties
```


LIST OF ACRONYMS

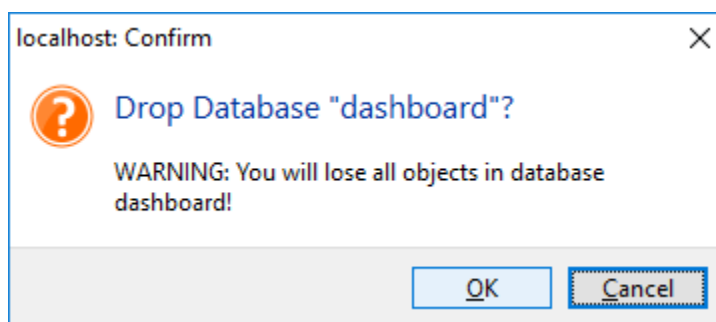
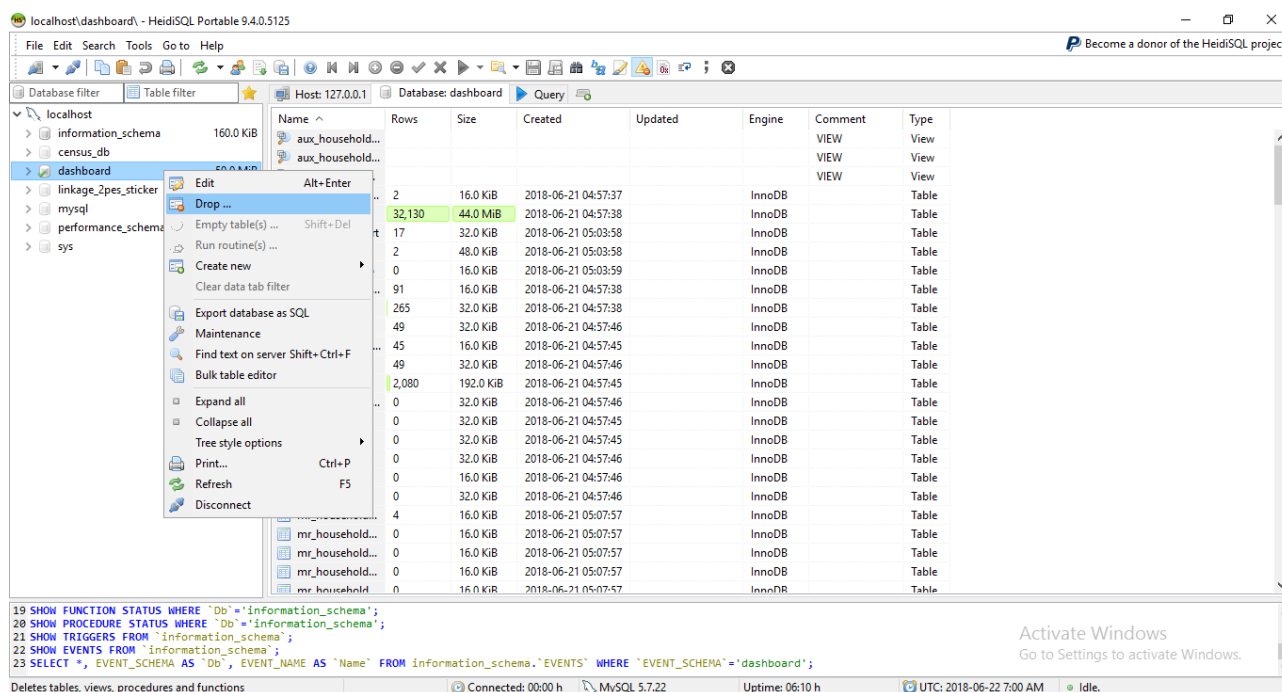
AICS	Italian Agency for Development Cooperation
CAPI	Computer-assisted personal interviewing
CSA	(Ethiopian) Central Statistical Agency
CSPro	Census and Survey Processing
CSPro2sql	Software managing the data transformation phase (from the CSPro Database to the Microdata DB)
DB	Database
DFID	(UK) Department for International Development
EPHC	Ethiopian Population and Housing Census
GIS	Geographic Information System
GPS	Global Positioning System
Istat	Italian National Institute of Statistics
IT	Information technology
ONS	(UK) Office for National Statistics
PES	Post Enumeration Survey
SQL	Structured Query Language
UNFPA	United Nations Population Fund
USAID	United States Agency for International Development
USCB	United States Census Bureau

ANNEXES

Annex 1 CPro Dashboard: step by step installation guide

Step 1: create the dashboard database

As first step, you need to drop the old version of the dashboard database (if you have one).



Now you are ready to create a new instance of the dashboard database.

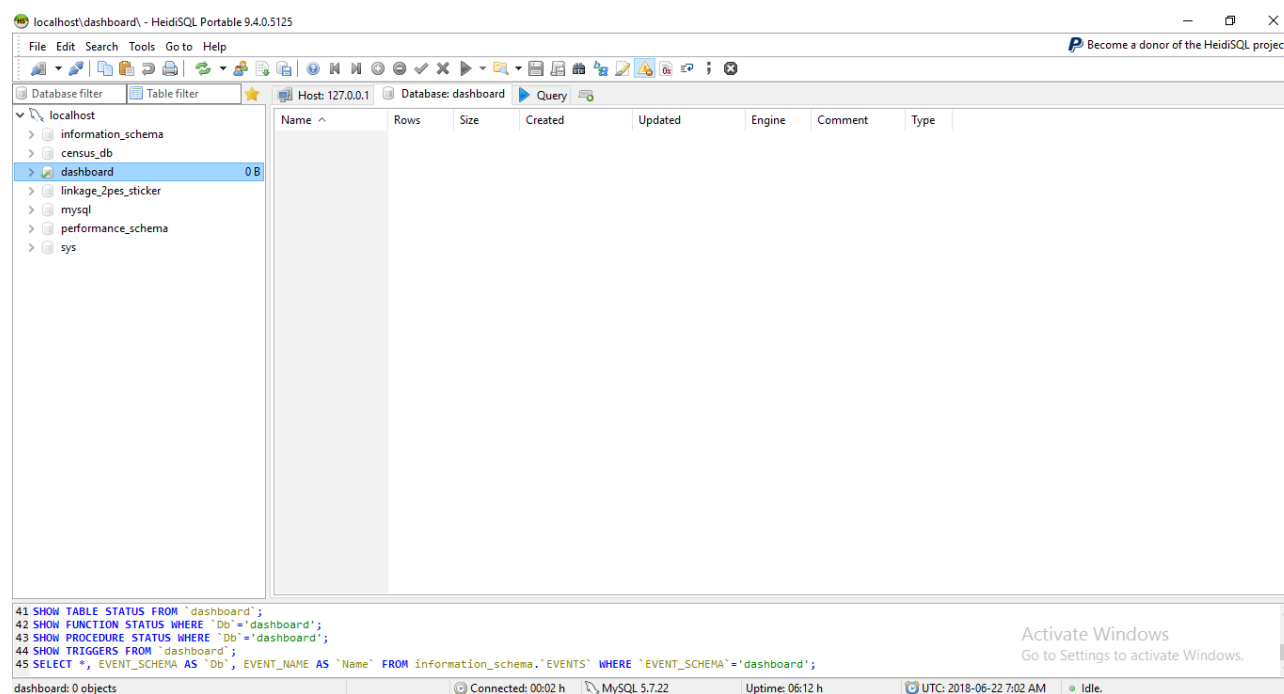
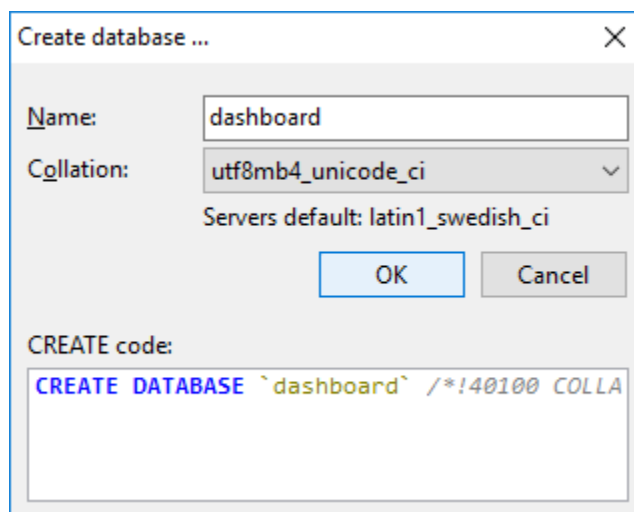


Figure 3 - At the end of this phase you will have an empty dashboard database

```

Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\dashboard>f:

F:\>cd "Program Files"

F:\Program Files>cd CsPro2Sql

F:\Program Files\CsPro2Sql>CsPro2Sql -e schema -p pilot3\pilot3.properties -o pilot3\schema.sql_
  
```

Figure 4 - Use CSPro2sql to generate the dashboard schema

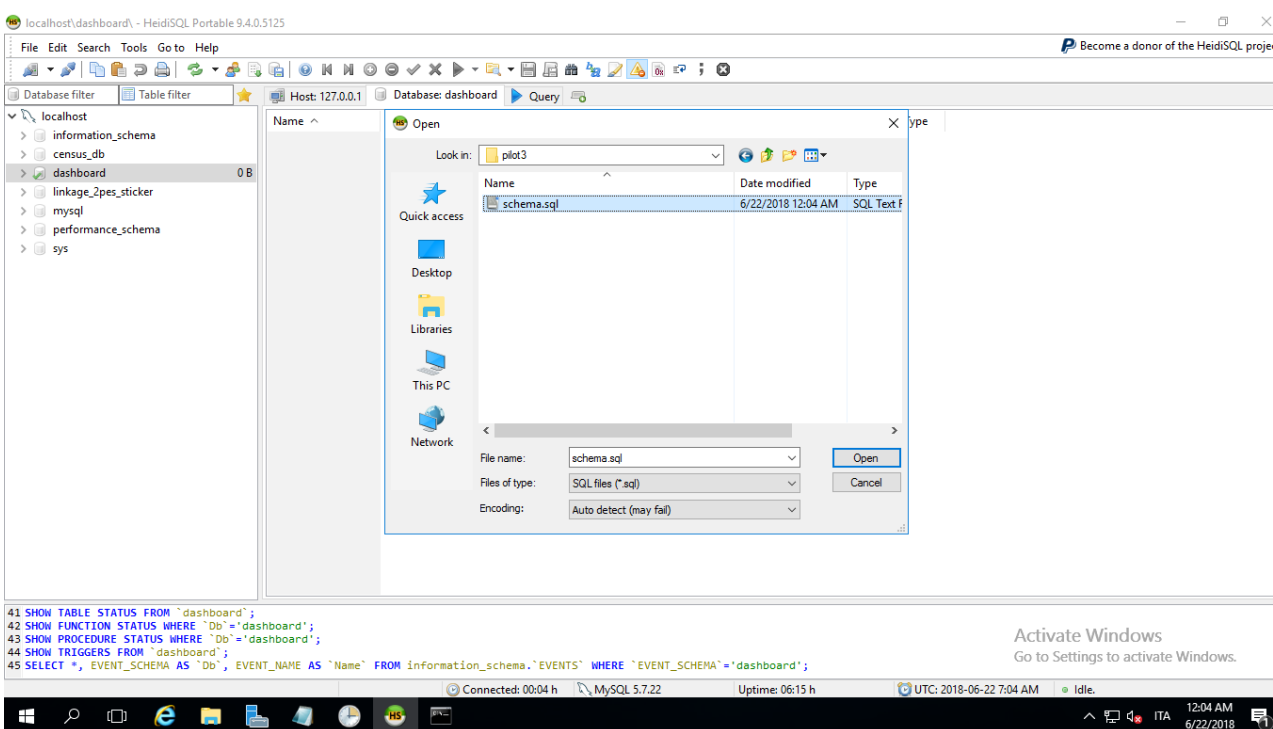


Figure 5 - Open the sql file, generated by CSPro2sql, in your favorite dbms client.

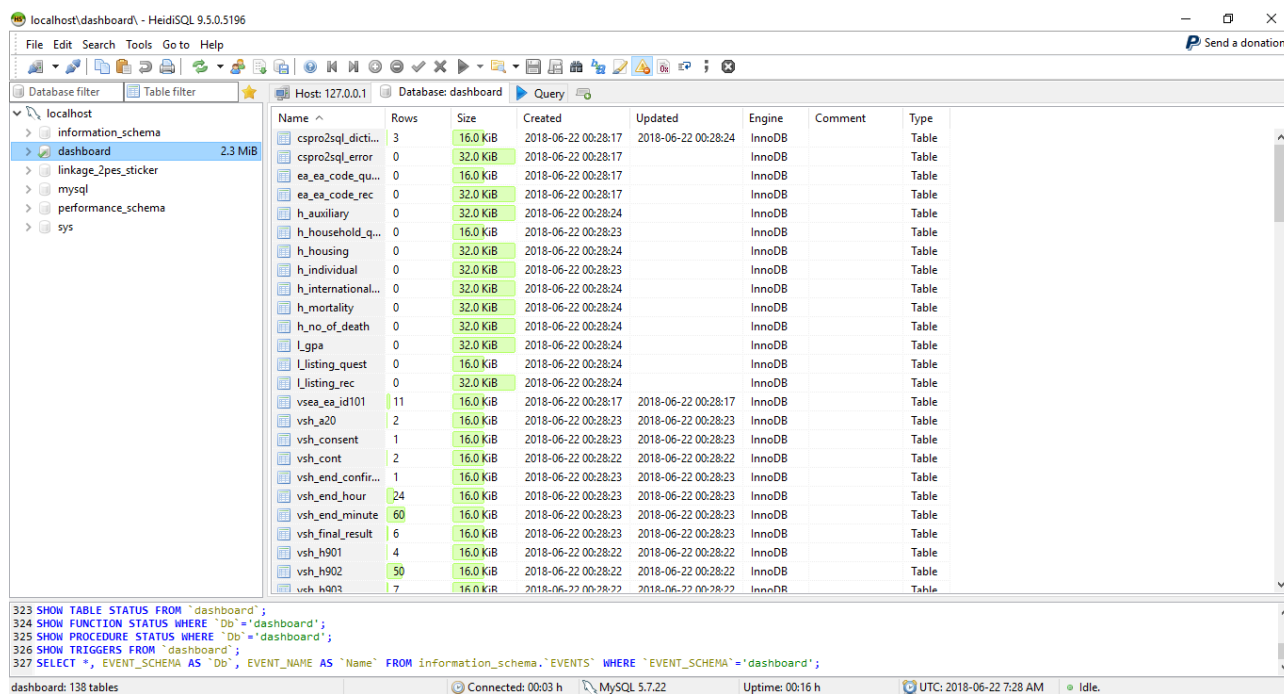
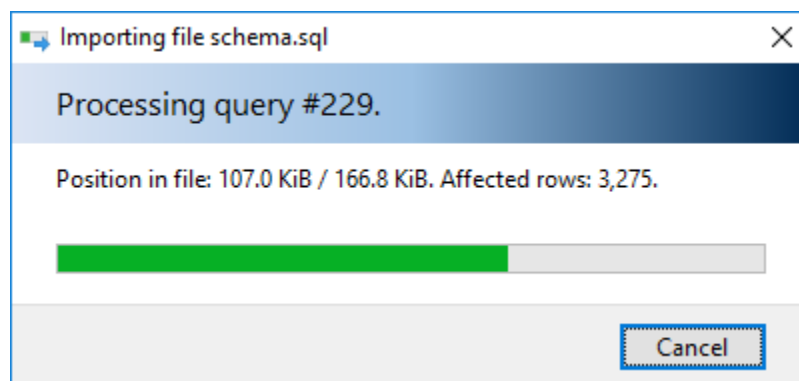


Figure 6 - At the end of this phase, the database will contain all the tables needed to store microdata

Step 2: load CSPro data in the dashboard database

It is time to start moving data from CSWeb to the dashboard database.

```
F:\Program Files\CsPro2Sql>CsPro2Sql -e loader -p pilot3\pilot3.properties -cc _
```

The output of CSPro2sql is the following:

```
F:\Program Files\CsPro2Sql>CsPro2Sql -e loader -p pilot3\pilot3.properties -cc
22/06/2018 00:05:52 Starting data transfer from CsPro/EA_CODE_DICT to MySql... [0 -> 4089]
+
22/06/2018 00:05:52 Data transfer completed!
Loaded: 91
Deleted: 0
Errors: 0
Total: 91
22/06/2018 00:05:52 Starting data transfer from CsPro/HOUSEHOLD_DICT to MySql... [0 -> 6569]
+++++++ Load: 1987 Err: 0 Tot: 2000
+++++++ Load: 3975 Err: 0 Tot: 4000
+++++++ _
```

The engine will display several information concerning the data transfer process.

```
22/06/2018 00:06:40 Data transfer completed!
Loaded: 8663
Deleted: 56
Errors: 0
Total: 8719
22/06/2018 00:06:40 Starting data transfer from CsPro/LISTING_DICT to MySql... [0 -> 6567]
+++++++ Load: 1974 Err: 0 Tot: 2000
+++++++ Load: 3940 Err: 0 Tot: 4000
+++++++ Load: 5907 Err: 0 Tot: 6000
+++++++ Load: 7882 Err: 0 Tot: 8000
+++++++ Load: 9859 Err: 0 Tot: 10000
+++++++ Load: 11828 Err: 0 Tot: 12000
+++++++ Load: 13790 Err: 0 Tot: 14000
+++++++ Load: 15760 Err: 0 Tot: 16000
+++++++ Load: 17733 Err: 0 Tot: 18000
+++++++ Load: 19700 Err: 0 Tot: 20000
+++++++ Load: 21662 Err: 0 Tot: 22000
+++++++ Load: 23620 Err: 0 Tot: 24000
+++++++ Load: 25598 Err: 0 Tot: 26000
+++++++
22/06/2018 00:10:52 Data transfer completed!
Loaded: 27111
Deleted: 424
Errors: 0
Total: 27535
```

Step 3: create the reporting tables

Once microdata is in the dashboard database, it is possible to generate reports to monitor fieldwork activities and to analyze retrieved data.

As a first step, it is necessary to create the monitor tables:

```
F:\Program Files\CsPro2Sql>CsPro2Sql -e monitor -p pilot3\pilot3.properties -o pilot3\monitor.sql
```

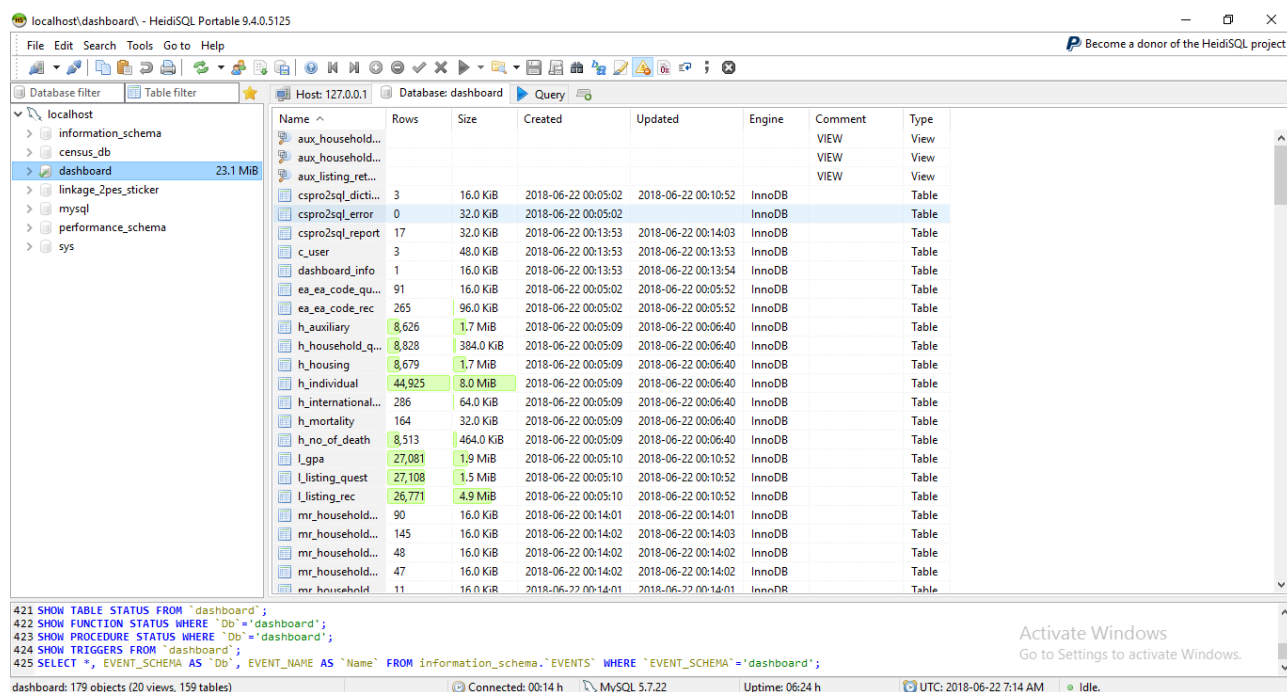
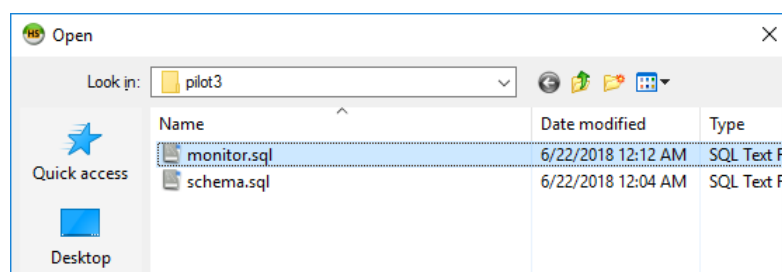


Figure 7 - Open the monitor sql file, generated by CSPro2sql, in your favorite dbms client.

Step 4: create the territory table and update the database

In order to generate reports at different territorial levels, it is necessary to create a territory table (as described in section 3.4 *Administrative territorial structure* (at the moment this procedure has to be performed manually).

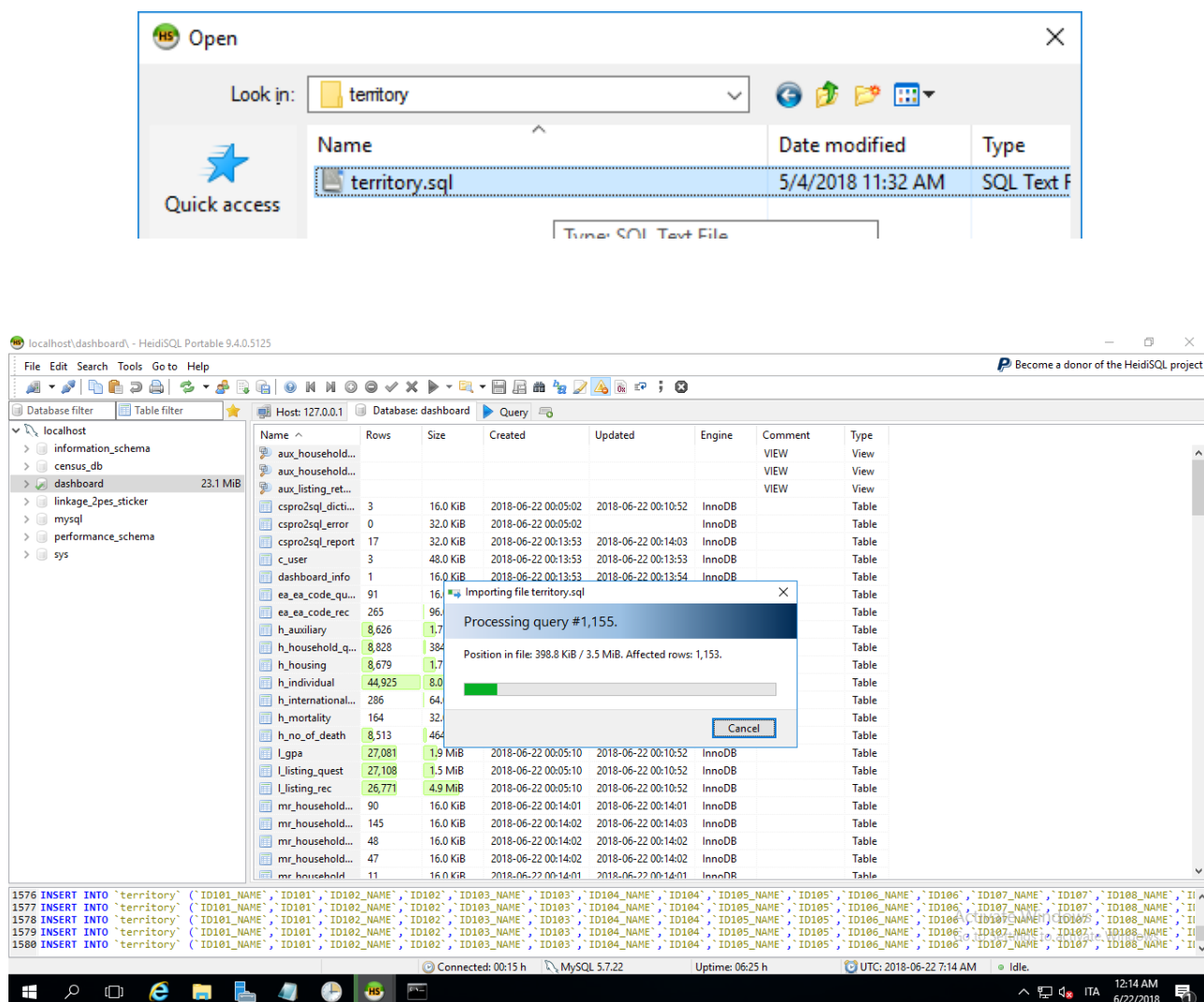


Figure 8 - Execute the territory script

Now it is time to update content the dashboard database. Run the following command:

```
F:\Program Files\CsPro2Sql\CsPro2Sql -e update -p pilot3\pilot3.properties
```

The output of the CSPro2sql script will be the following:

```
F:\Program Files\CsPro2Sql>CsPro2Sql -e update -p pilot3\pilot3.properties
Updating r_household_by_ea... done
Updating r_household_expected_by_ea... done
Updating r_household_expected_by_kebele... done
Updating r_household_expected_by_psa... done
Updating r_household_expected_by_region... done
Updating r_household_expected_by_sa... done
Updating r_household_expected_by_subcity... done
Updating r_household_expected_by_town... done
Updating r_household_expected_by_woreda... done
Updating r_household_expected_by_zone... done
Updating r_individual_info... done
Updating r_questionnaire_info... done
Updating r_regional_area... done
Updating r_religion... done
Updating r_sex_by_age... done
Updating r_sex_by_age_group... done
Updating r_total... done
```

Step 5: access the dashboard web application

Assuming that you have already installed the dashboard and that the application is running at:

<http://localhost:8080/dashboard>

Access the page and login with the default credentials:

Username: admin@dashboard.it

Password: dashboard

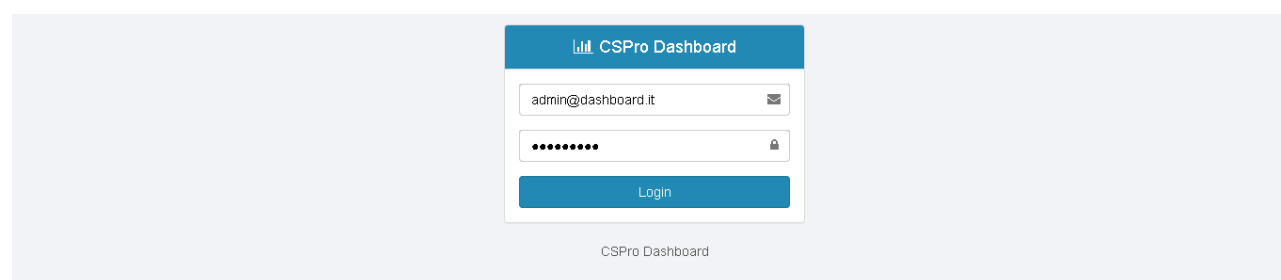


Figure 9 - Dashboard login page

In the following figures, some reports provided by the dashboard are displayed.

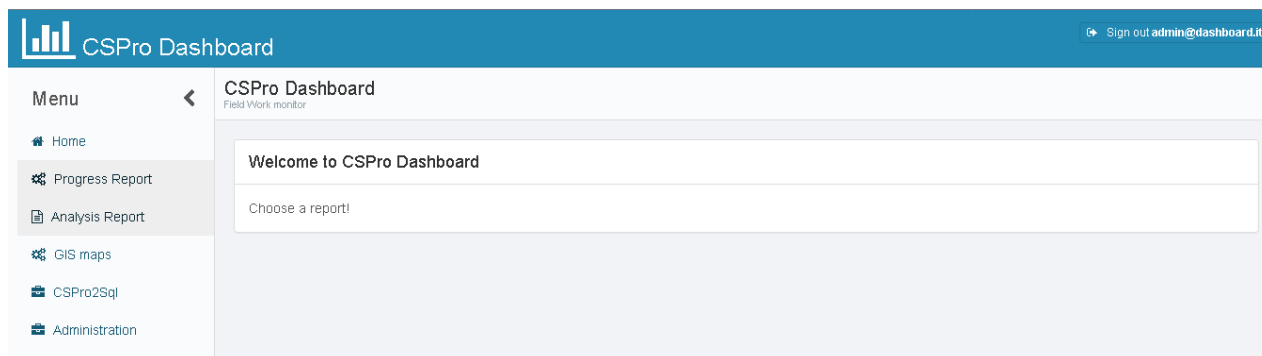


Figure 10 - Welcome page

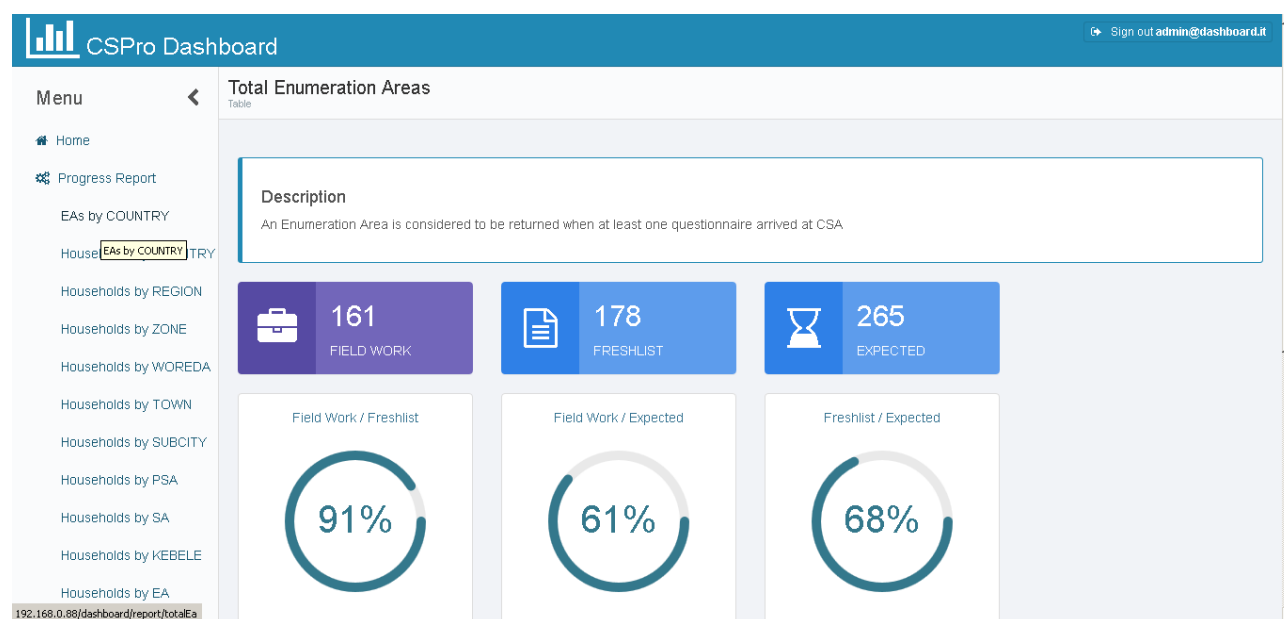
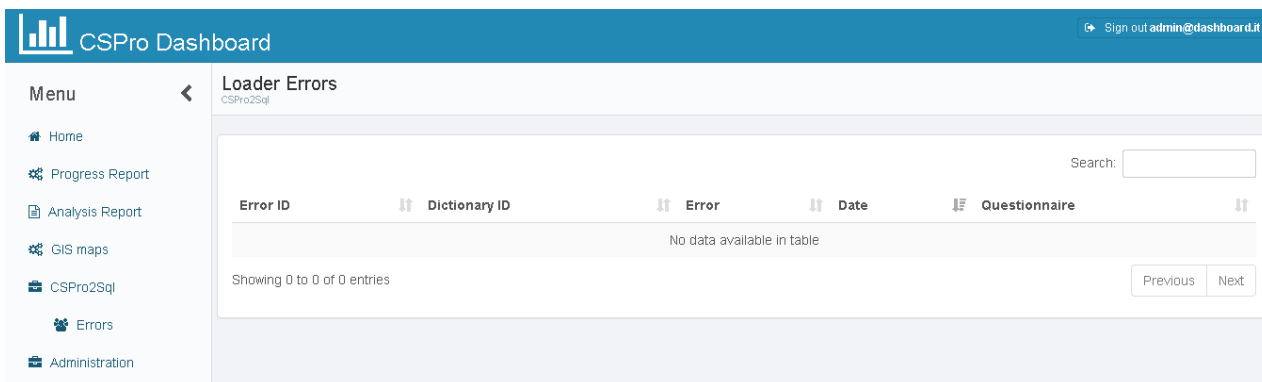


Figure 11 - Progress report at national level



CSPro Dashboard Sign out admin@dashboard.it

Menu

- Home
- Progress Report
- Analysis Report
- GIS maps
- CSPro2Sql
- Errors**
- Administration

Loader Errors
CSPro2Sql

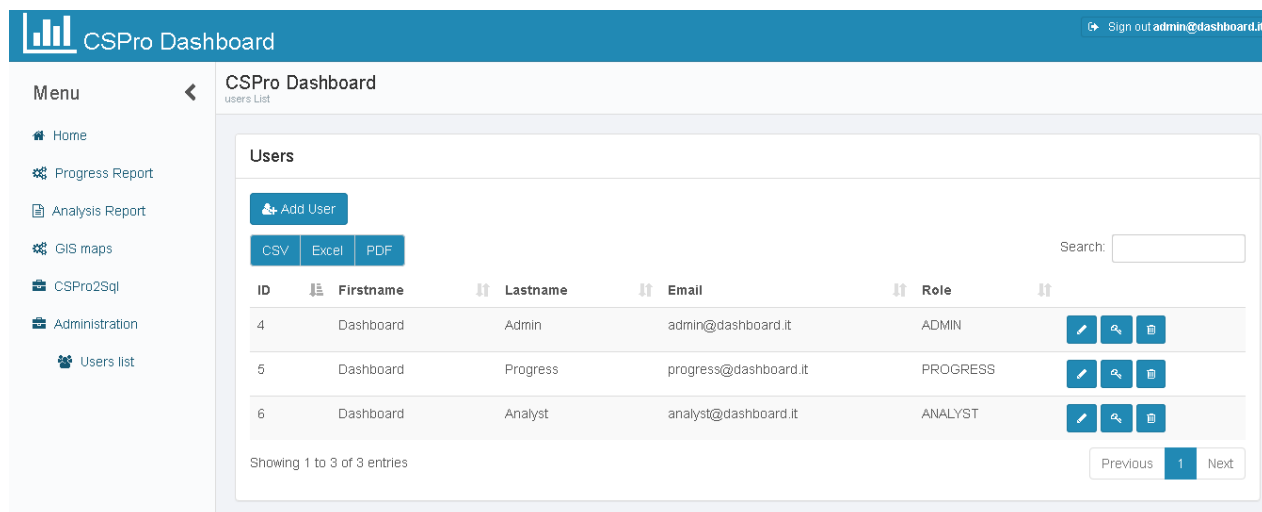
Search:

Error ID	Dictionary ID	Error	Date	Questionnaire
No data available in table				

Showing 0 to 0 of 0 entries

Previous Next

Figure 12 - Data transfer errors



CSPro Dashboard Sign out admin@dashboard.it

Menu

- Home
- Progress Report
- Analysis Report
- GIS maps
- CSPro2Sql
- Administration
- Users list**

CSPro Dashboard
users List

Users

[Add User](#)

[CSV](#) [Excel](#) [PDF](#)

Search:

ID	Firstname	Lastname	Email	Role
4	Dashboard	Admin	admin@dashboard.it	ADMIN
5	Dashboard	Progress	progress@dashboard.it	PROGRESS
6	Dashboard	Analyst	analyst@dashboard.it	ANALYST

Showing 1 to 3 of 3 entries

Previous 1 Next

Figure 13 - Administration page

Annex 2 CSPro Dashboard: scheduler configuration

In order to update the reports provided by the dashboard it is necessary to configure a batch script that invokes CSPro2sql engines. In this section we analyze the sequence of operations needed to create a task scheduler on a windows server.

Script 1: loader

We assume that you created a batch script *script1_H_loader.bat* to move data from CSWeb to the MicrodataDB:

```
@ECHO OFF

echo %date%-%time%

ECHO Starting Script1 execution at %date% - %time% >> log\script1_H_loader.log

java -cp lib\CSPro2sql.jar;lib\commons-cli-1.3.1.jar;lib\mysql-connector-java-5.1.40-bin.jar CSPro2sql.Main %* -e loader -p pilot3\pilot3.properties -cc -f >> log\script1_H_loader.log 2>>&1
```

We will describe how to generate a task that everyday executes the script.

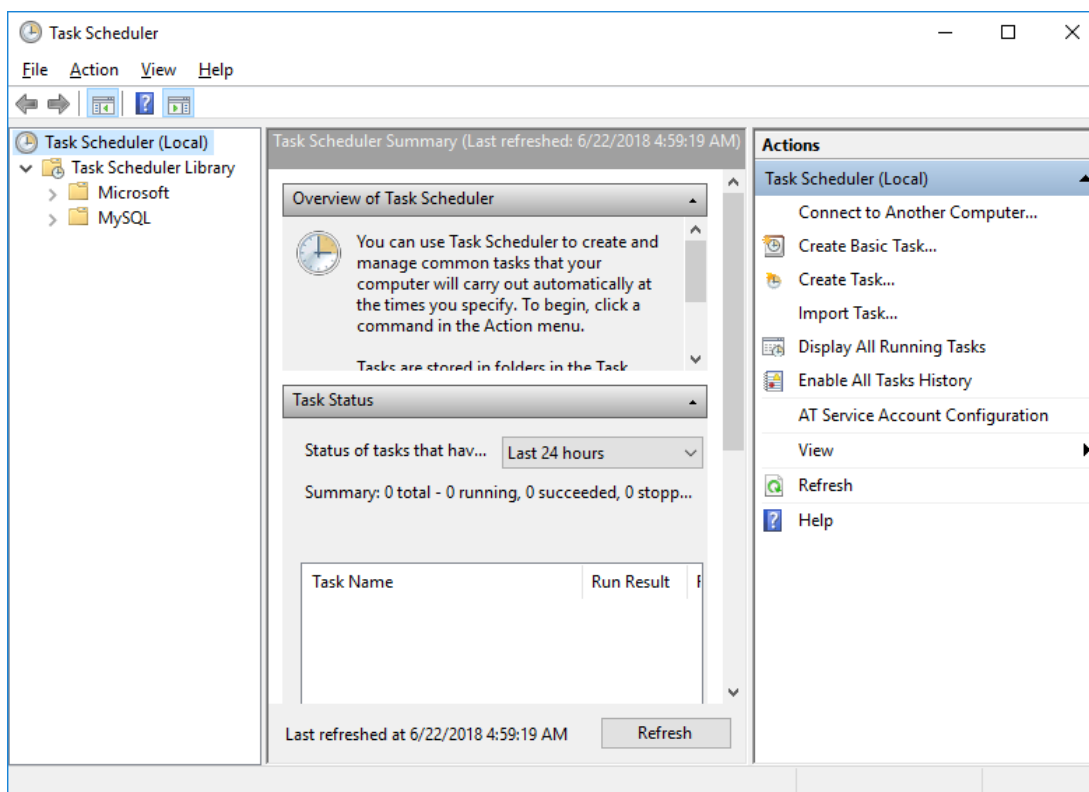


Figure 14 - Access the task scheduler

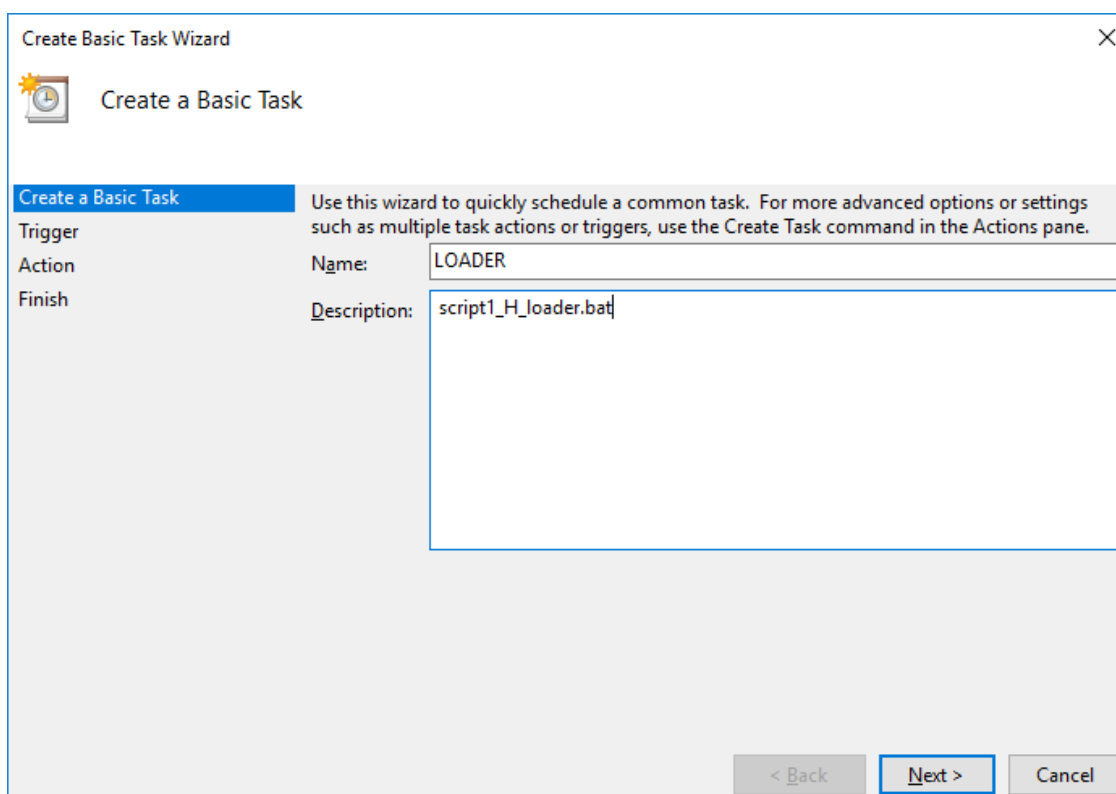
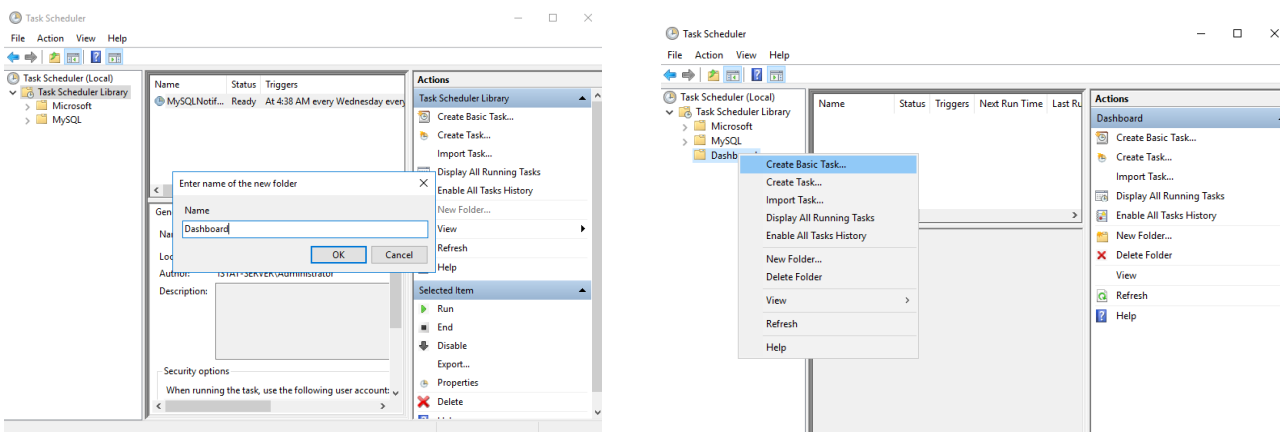



Figure 15 - Create a new Basic Task


Task Trigger

Create a Basic Task

Trigger

Action

Finish

When do you want the task to start?

☒ Daily

☐ Weekly

☐ Monthly

☐ One time

☐ When the computer starts


☐ When I log on

☐ When a specific event is logged

< Back

Next >

Cancel


Daily



Create a Basic Task

Trigger

Daily

Action

Finish

Start: 6/22/2018  8:00:00 PM  ☐ Synchronize across time zones


Recur every: 1 days

< Back

Next >

Cancel

Create Basic Task Wizard

 **Action**

Create a Basic Task

Trigger
Daily

Action

Finish

What action do you want the task to perform?


☒ Start a program

☐ Send an e-mail (deprecated)

☐ Display a message (deprecated)

< Back Next > Cancel

Create Basic Task Wizard

 **Start a Program**

Create a Basic Task

Trigger
Daily

Action

Finish


Program/script:
 Browse...

Add arguments (optional):

Start in (optional):

< Back Next > Cancel

Create Basic Task Wizard

 Summary

Create a Basic Task

Trigger

Daily

Action

Start a Program

Finish

Name:

Description:

Trigger:

Action:

☒ Open the Properties dialog for this task when I click Finish

When you click Finish, the new task will be created and added to your Windows schedule.

< Back Finish Cancel

LOADER Properties (Local Computer)

General Triggers Actions Conditions Settings History (disabled)

Name:

Location:

Author:

Description:

Security options

When running the task, use the following user account:

☐ Run only when user is logged on

☒ Run whether user is logged on or not

☐ Do not store password. The task will only have access to local computer resources.

☐ Run with highest privileges

☒ Hidden

Configure for:

OK Cancel

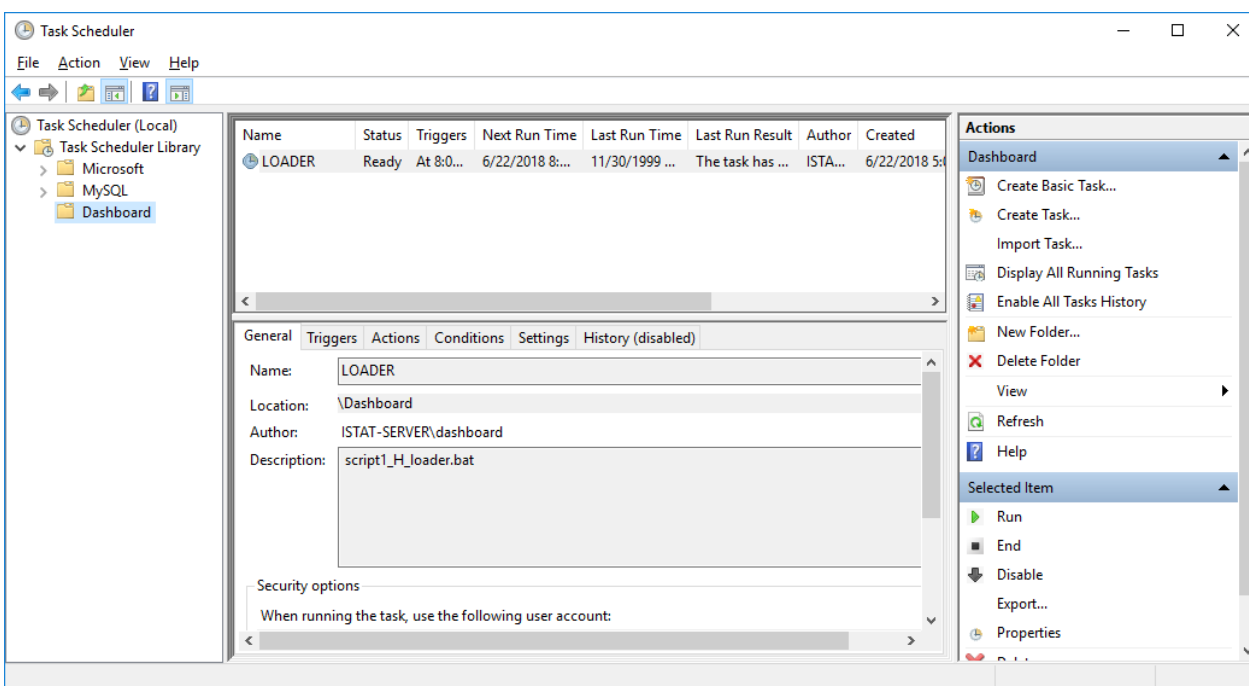


Figure 16 - The loader script will be executed every day

Script 2: update

We assume that you created a batch script *script2_H_update.bat* to update the reports displayed by the dashboard:

```
@ECHO OFF

@echo %date%-%time%

ECHO Starting Script2 execution at %date% - %time% >> log\script2_H_update.log

java -cp lib\CSPPro2sql.jar;lib\commons-cli-1.3.1.jar;lib\mysql-connector-java-5.1.40-
bin.jar CSPPro2sql.Main %* -e update -p pilot3\pilot3.properties
>>log\script2_H_update.log 2>>&1
```

We will describe how to generate a task that everyday executes the script.

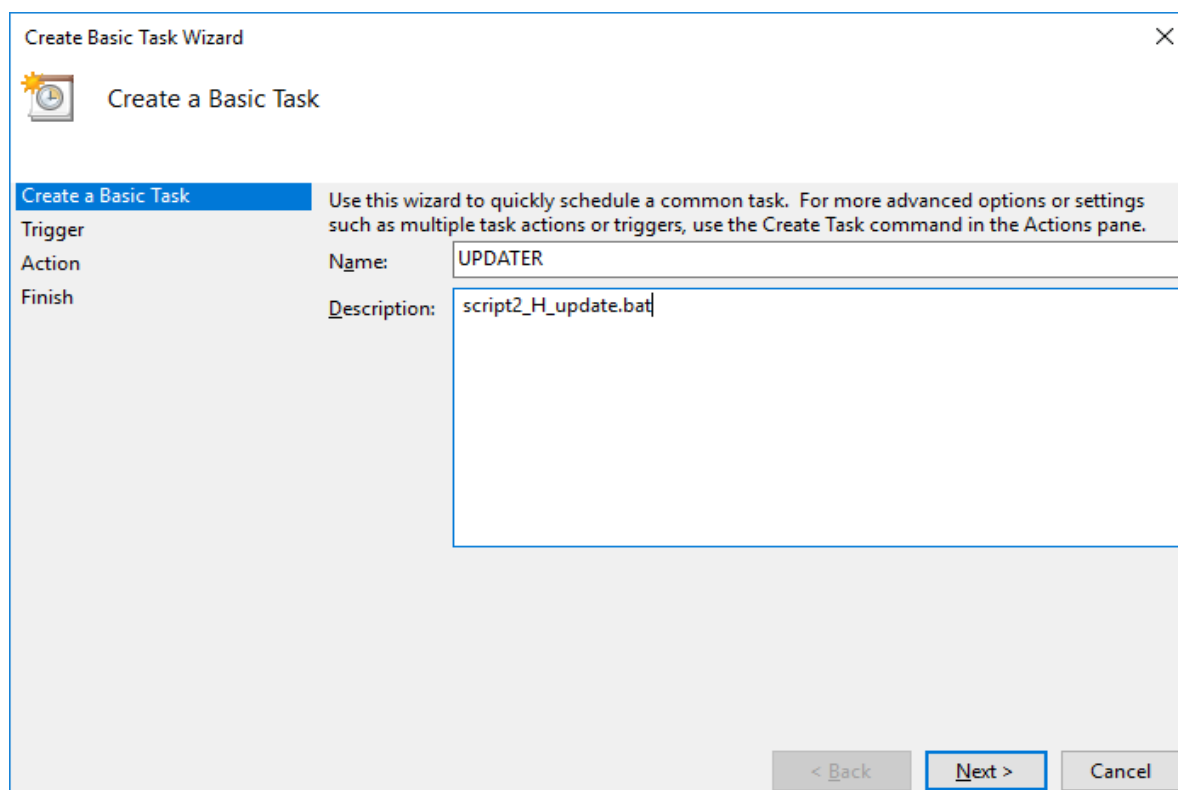



Figure 17 - Create a new Basic Task to run the updater script

Create Basic Task Wizard

 Task Trigger

Create a Basic Task

Trigger

Action

Finish

When do you want the task to start?

☒ Daily

☐ Weekly

☐ Monthly

☐ One time


☐ When the computer starts

☐ When I log on

☐ When a specific event is logged

< Back Next > Cancel

Create Basic Task Wizard

 Daily

Create a Basic Task

Trigger

Daily

Action


Finish

Start: 6/22/2018 11:00:00 PM ☐ Synchronize across time zones

Recur every: 1 days

< Back Next > Cancel

Create Basic Task Wizard

 **Action**

Create a Basic Task


Trigger
 Daily
Action
 Finish

What action do you want the task to perform?

☒ Start a program
☐ Send an e-mail (deprecated)
☐ Display a message (deprecated)

< Back Next > Cancel

Create Basic Task Wizard

 **Start a Program**

Create a Basic Task

Trigger
 Daily
Action
 Start a Program
 Finish


Program/script:
 Browse...

Add arguments (optional):

Start in (optional):

< Back Next > Cancel

Create Basic Task Wizard [X]

 **Summary**

Create a Basic Task

Trigger: **Daily**
 Action: **Start a Program**
Finish

Name:
 Description:

Trigger:
 Action:

☒ Open the Properties dialog for this task when I click Finish
 When you click Finish, the new task will be created and added to your Windows schedule.

UPDATER Properties (Local Computer) [X]

General | Triggers | Actions | Conditions | Settings | History (disabled)

Name:
 Location:
 Author:
 Description:

Security options

When running the task, use the following user account:

☐ Run only when user is logged on
☒ Run whether user is logged on or not
☐ Do not store password. The task will only have access to local computer resources.
☐ Run with highest privileges

☒ Hidden Configure for:

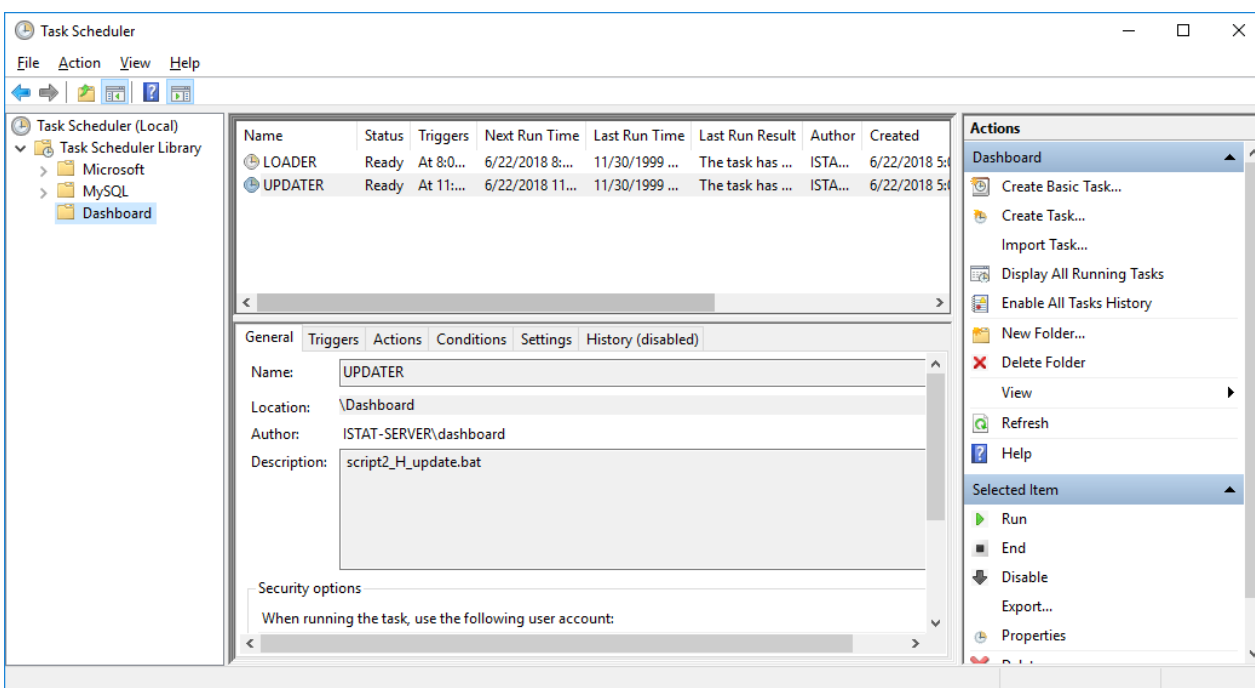
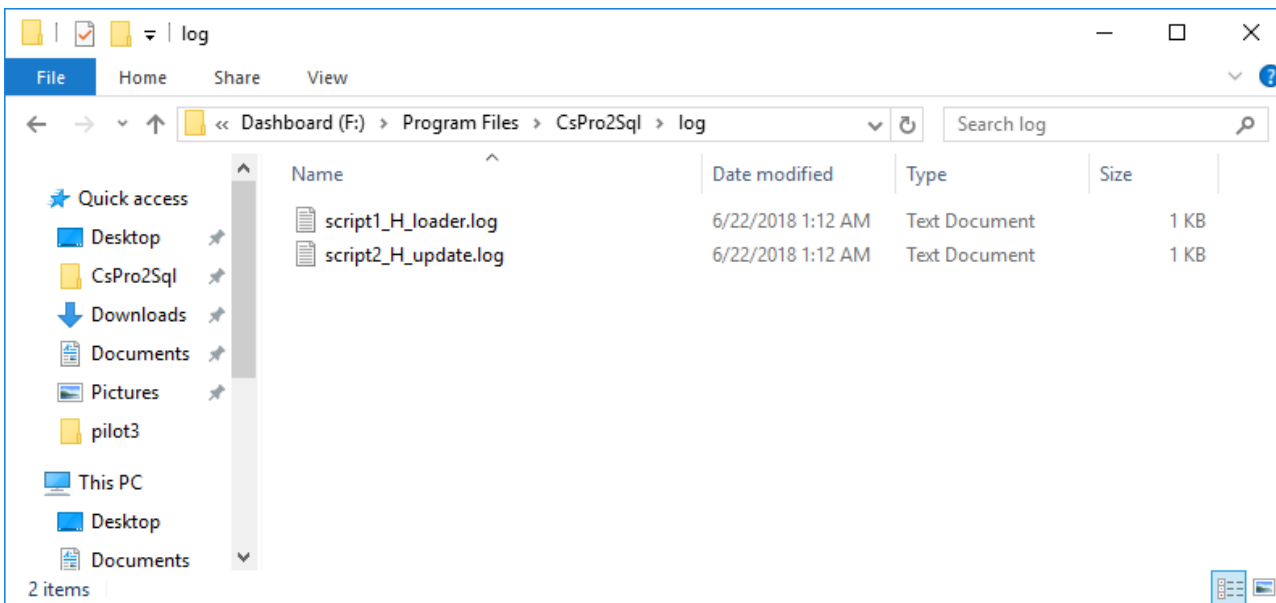


Figure 18 - The updater script will be executed every day

Log files

Log files will be available in the log folder, as shown in the above figure



Annex 3 CPro Dashboard: report implementation guide

In the following paragraph is provided a step by step guide for adding a new report on CPro Dashboard.

Step 1: Create the new report in MySQL Database

Select the correct DB schema

```
USE dashboard;
```

SQL instruction to create a new view *r_religion_new*

```
CREATE OR REPLACE VIEW `r_religion_new` AS
SELECT
  `vs`.`VALUE` AS `RELIGION`, COUNT(0) AS `INDIVIDUALS`
FROM
  (`h_individual` `i`
  JOIN `vsh_p309` `vs` ON ((`i`.`P309` = `vs`.`ID`)))
GROUP BY `vs`.`VALUE`;
```

SQL instruction to manage the materialized view

```
DROP TABLE IF EXISTS dashboard.mr_religion_new;
SELECT 0 INTO @ID;
CREATE TABLE dashboard.mr_religion_new (PRIMARY KEY (ID)) AS
SELECT @ID := @ID + 1 ID, r_religion_new.* FROM dashboard.r_religion_new;
```

Insert a record into the reports list table

```
DELETE FROM dashboard.`CPro2sql_report` WHERE NAME = 'r_religion_new';
INSERT INTO dashboard.`CPro2sql_report` (NAME, LIST_ORDER) VALUES
('r_religion_new', 7);
```

The table *reports* is used by the Dashboard Java code to generate dynamically the navigation menu.

Step 2: Edit html pages in order to display the report

In order to display the new report in the navigation menu, it is necessary to modify the **layout.html** file. More precisely, the following element should be added:

```
<li id="report-religion-new" th:if="${reports.contains('r_religion_new')}">
    <a title="Religion" th:href="@{/report/religionNew}">
        <span>Religion New</span>
    </a>
</li>
```

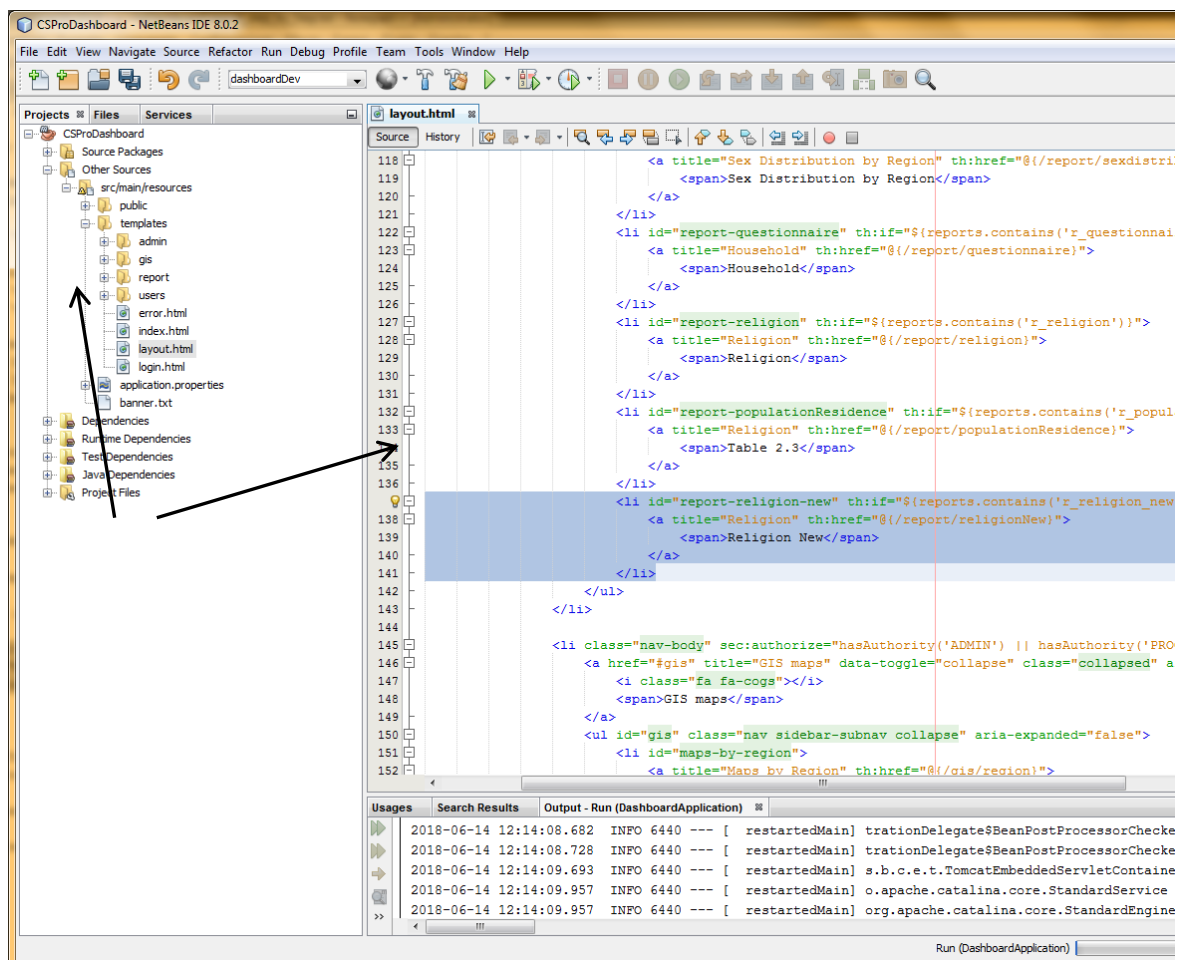


Figure 19 - Dashboard layout page

Create a new file named **religionNew.html** in the template/report folder, it can be helpful to copy an existing file and edit it (religion.html in this case).

Modify the tag `<script>` in the new file:

```
<script th:src="@{/js/report/religionNewList.js}" type="text/javascript"></script>
```

by changing the javascript file name from religion.js in religionNew.js

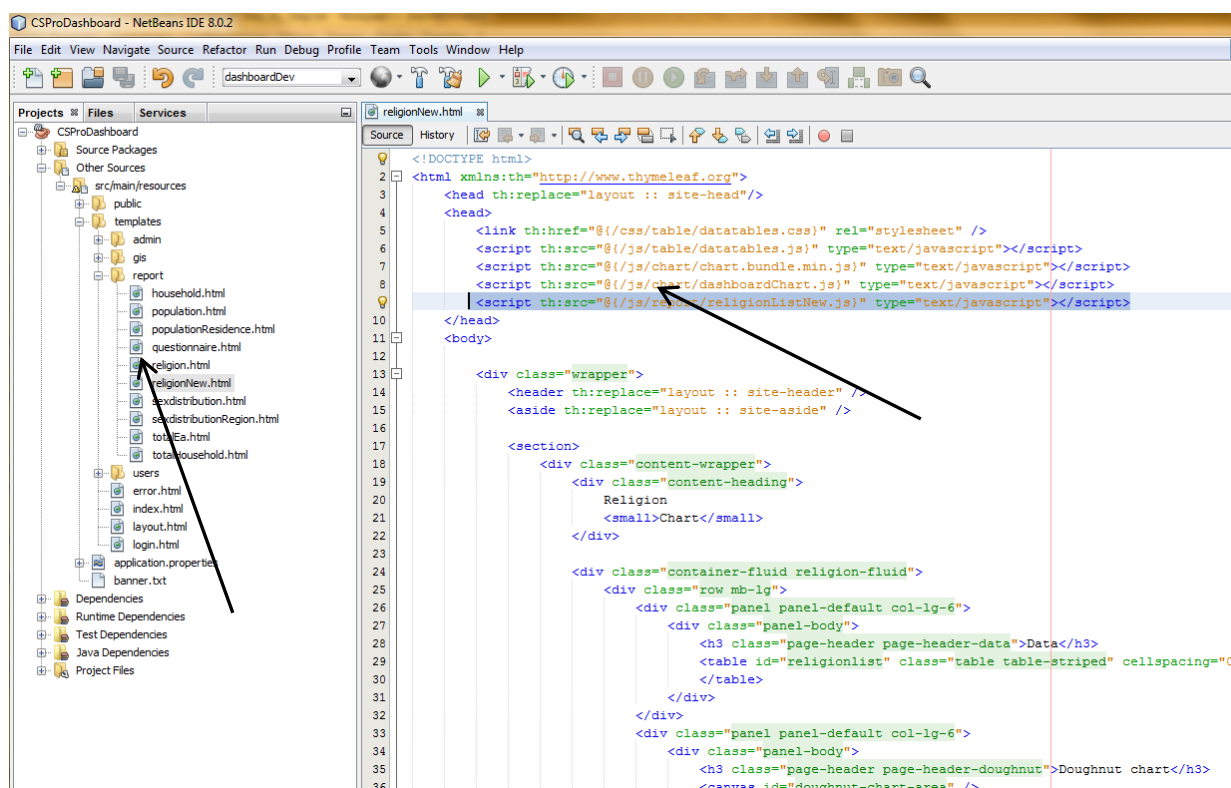
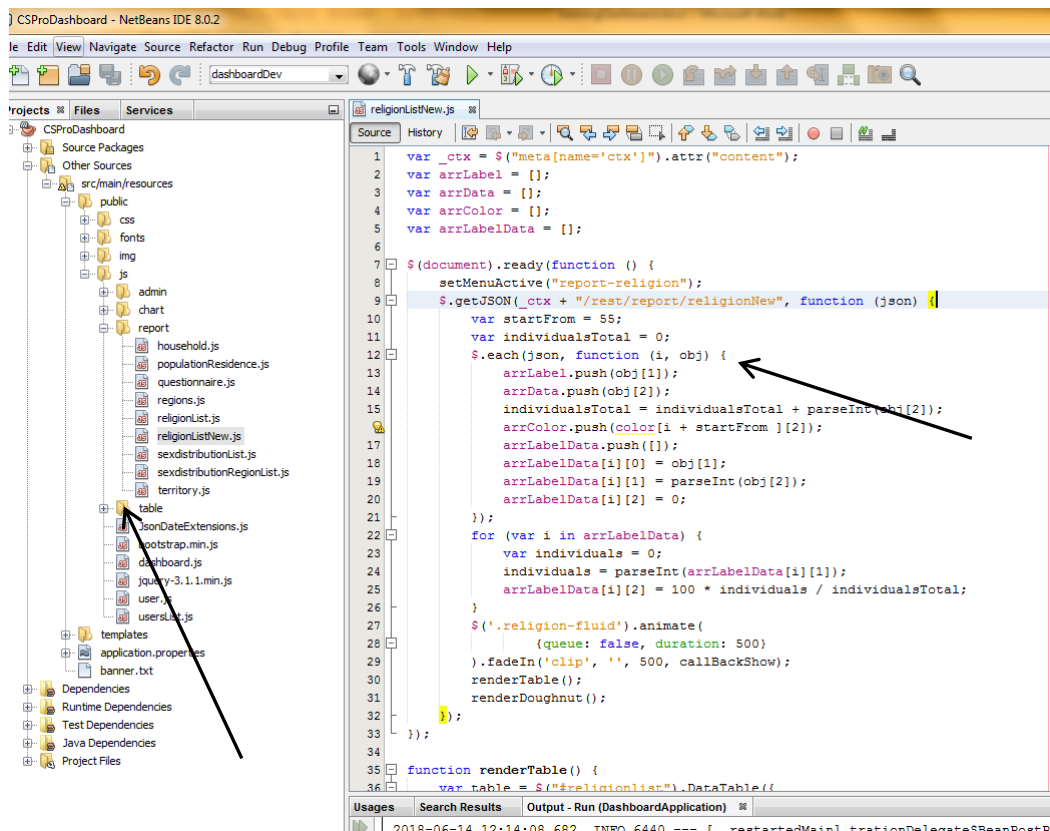


Figure 20 - Create a new html file and set the correct javascript input

Create a new javascript file named **religionNewList.js** in public/js/report, to access to data get the json call at the address:

<http://localhost:8080/dashboard/rest/report/religionNew>

In the above url, **religionNew** is the materialized view name (**religion_new**) in camel case.



```




1  var _ctx = $("meta[name='ctx']").attr("content");
2  var arrLabel = [];
3  var arrData = [];
4  var arrColor = [];
5  var arrLabelData = [];
6
7  $(document).ready(function () {
8      setMenuActive("report-religion");
9      $.getJSON(_ctx + "/rest/report/religionNew", function (json) {
10         var startFrom = 55;
11         var individualsTotal = 0;
12         $.each(json, function (i, obj) {
13             arrLabel.push(obj[1]);
14             arrData.push(obj[2]);
15             individualsTotal = individualsTotal + parseInt(obj[2]);
16             arrColor.push(color[i + startFrom][2]);
17             arrLabelData.push([i]);
18             arrLabelData[i][0] = obj[1];
19             arrLabelData[i][1] = parseInt(obj[2]);
20             arrLabelData[i][2] = 0;
21         });
22         for (var i in arrLabelData) {
23             var individuals = 0;
24             individuals = parseInt(arrLabelData[i][1]);
25             arrLabelData[i][2] = 100 * individuals / individualsTotal;
26         }
27         $('.religion-fluid').animate(
28             {queue: false, duration: 500}
29         ).fadeIn('clip', '', 500, callBackShow);
30         renderTable();
31         renderDoughnut();
32     });
33 });
34
35 function renderTable() {
36     var table = $("#religionList").DataTable({

```


Figure 21 - Example of js code to invoke the rest service


Annex 4 CPro Dashboard: technologies and frameworks


Client side frameworks:

-  **jQuery**
write less, do more.
 - <https://jquery.com/>
 - JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.
-  **DataTables**
 - <https://datatables.net/>
 - Plug-in for the jQuery Javascript library. It is a highly flexible tool, build upon the foundations of progressive enhancement, that adds all of these advanced features to any HTML table.
-  **AJAX**
Asynchronous Javascript And XML.
 - <http://api.jquery.com/jquery.ajax/>
 - Set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications.
-  **Bootstrap**
 - <https://getbootstrap.com/>
 - Open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.
-  **HTML5**
 - <https://www.w3.org/TR/html52/>
 - Markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current major version of the HTML standard.
-  **CSS3**
 - <https://www.w3.org/Style/CSS/>
 - Cascading Style Sheets (CSS) style sheet language used for describing the presentation of a document written in a markup language like HTML.
-  **Font Awesome**
 - <https://fontawesome.com/>
 - Icon set and toolkit.


Server Side frameworks:

-  **Thymeleaf**
 - <https://www.thymeleaf.org/>
 - Server-side Java template engine for both web and standalone environments.

-  **spring**
by Pivotal
 - <https://spring.io/>
 - Application framework and inversion of control container for the Java platform.
 - Main projects:
 - Spring Boot
 - Spring MVC
 - Spring Security
 - Spring Data JPA

-  **HIBERNATE**
 - <http://hibernate.org/>
 - Object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.

- **Maven™**
 - <https://maven.apache.org/>
 - Software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

-  **MySQL™**
 - <https://www.mysql.com/>
 - Open-source relational database management system.