
# UNIVERSITÀ DI PISA

**Laboratory on Data Science**

Project Report

## Answers Data Warehouse

Group_24:

Davide Chen - 544795

Andrea Ribellini - 554043

---

**Academic Year 2022/2023**

# Part 1 - Datawarehouse building

# 1 Introduction

In this report we will describe the organization and the steps performed in order to build and populate the Data Warehouse required by the project assignment.

## 1.1 Project organization

Here below are described the organization of the project, the structure of the folders and the contents of the files:

- ***/dataset/:*** in this folder there are data csv file as give to us by the client
  - ***answerdatacorrect.csv*:** contains the set of columns to be extracted and distributed in the various tables
  - ***subject_ metadata.csv:*** contains the subject metadata such as *Ids*, *Name* and *Level*
- ***/tables/:*** in this folder contains the splitted tables saved into csv files ready to be loaded to the SQL Server
  - ***answer.csv***, ***organization.csv***, ***date.csv***, ***subject.csv***, ***user.csv***, ***geography.csv***
- ***preprocess.py:*** for data understanding and prepocessing
- ***splitTable.py:*** for split the columns into different files
- ***load.py:*** for load the data to the server

# 2 Data Understading and Pre-processing

It was performed some basic data understanding and pre-processing tasks to discover the presence of missing values/redundant data and correct some incorrect values, in particular:

- The *answerdatacorrect.csv* dataframe has 17 columns and 0 missing value was found
- The same dataframe has a total of 538835 rows:
  - AnswerId has 538835 unique values and 0 duplicates
  - UserId has 13630 unique values and 525205 duplicates
  - SubjectId has 412 unique values and 538423 duplicates
  - Organization table will have a total 24640 unique rows (given by [GroupId, QuizId, SchemeOfWorkId] unique rows)
  - Geography table will have a total 76 unique rows (given by Region unique rows)
- The CountyCode column has the following unique values:

$$[de, us, ie, it, nz, es, uk, fr, ca, be, au]$$

  In order to match [ISO 3166-1 alpha-2] standard of country code, it was applied a transformation for the code 'uk' to 'gb'. This will facilitate the conversion into country name and obtain the relative continent of belonging.

# 3 Splitting Table

As required by the project assignment we start with a "big table" answercorrect.csv file and we need to partition it in several tables according to the given structure. As output of this section, we will have as many csv files as tables we will have in the server: *six*.

### 3.1 Organization table

**Primay key:** OrganizationId **Secondary key:** None

For this table, the attributes required are *OrganizationId*, *GroupId*, *QuizId* and *SchemeOfWorkId*. The last 3 features are provided by the input file, but we had to generate the primary key by ourselves from the other attributes. Since each of them has unique and duplicate values in different numbers and positions, it was decided to assign the ids by the uniqueness of the entire row. In other word, the row is considered unique and assigned with a new id, if the three values of the row combined together is unique.

### 3.2 Date table

**Primay key:** DateId **Secondary key:** GeoId

Also here the primary keys DateId are not provided with the rest of the data, but was easily solved by giving a progressive enumeration for each unique date. These indexes were then saved in a dictionary with key the corresponding *AnswerId* and *UserId* so that at the time of building the answer and user table, for each row it can retrieve the correct *DateId*. Since the table should store two type of date: *DateAnswered* and *DateOfBirth*, for eliminate the redundancy of data, it was decided to save only a single value for each unique date. That means if we had a common date for DateOfBirth or DateAnswered (improbable, but never say never!) it would refer to a common record of this table .

### 3.3 Subject table

**Primay key:** SubjectId **Secondary key:** None

For the subject table we also needed to open the subject_metadata file in order to get the correct subject names for each list of index contained in *SubjectId*. In fact, each SubjectId is a list of indexes, for each of them we had to iterate over the metadata file' row, find the corrisponding value and save the name into a dictionary indexed by its level. The dictionary is then sorted by its key, to have the correct order to been written in the Description feature.

### 3.4 Geography table

**Primay key:** GeoId **Secondary key:** None

The greography table is built by converting the CountryCode values to Country Name e Continent by using a python library called *pycountry_convert*. The geoId key was generated in a progressive way from 1 to n according to the regions present in the data, after eliminating the duplicates. These indexes were saved in a dictionary with key the corresponding *UserId*s so that at the time of building the user table, for each row it can retrieve the correct *GeoId* key.

### 3.5 User table

**Primay key:** UserId **Secondary key:** DateOfBirthId

One of the easiest table: all the needed value are already provided or computed. It was enough to retrieve them and write them to the new file, after eliminated all the duplicates.

### 3.6 Answers table

**Primay key:** AnswerId **Secondary key:** OrganizationId, UserId, DateId, SubjectId

Same for the user table. The *IsCorrect* attirbute were computed by comparing the value of *AsnwerValue* and *CorrectValue*.

# 4 Load

We read tables ready to be loaded from the various csv files we've built so far. Then we load them by executing a standard insert query, row by row, and executing the commit every 100 rows and again, once, at the end for any "leftover" rows. We execute a commit every 100 rows because there's a lot of data in some of the table such as *Answers* or *Date* and we want to ensure that even if the connection fails for any reason (i.e. timeout or others) we have only lost at the most the loading of the last 100 rows.

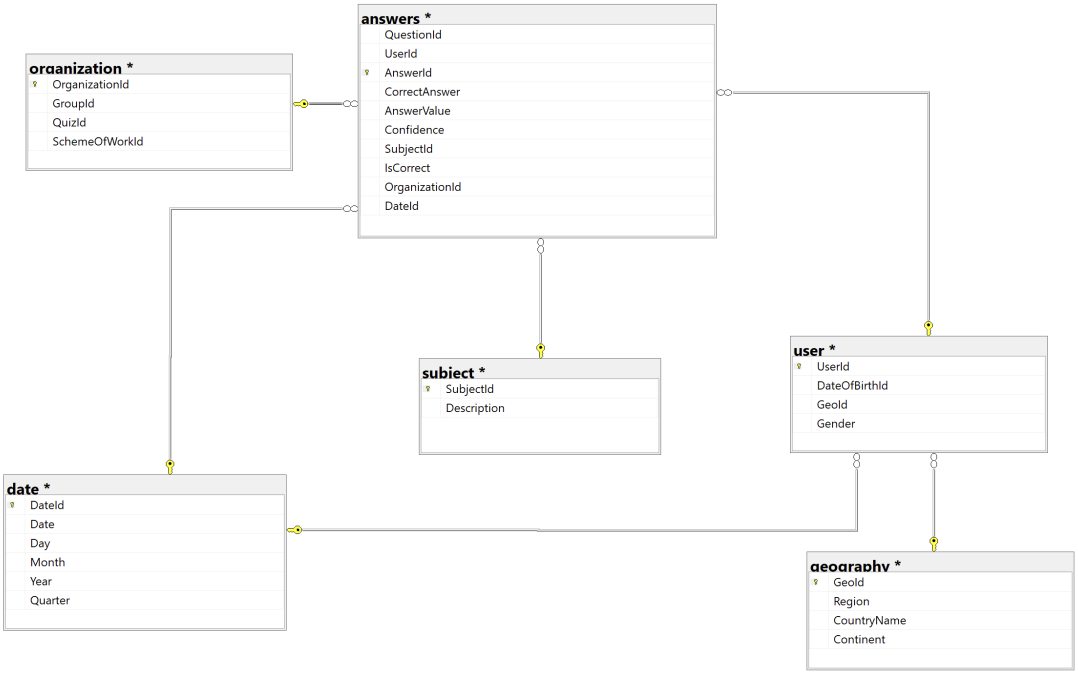



Figure 1: DB Schema obtained in Microsoft SQL Server Management Studio

# Part 2 - SSIS Solution

# 5 SSIS Data Flows

In order to build the data flows required for the seccond assignment, we first opened Visual Studio 2019 and created an Integration Service Project named LDS_Group24_Part2. Afterwards, we've initiated three packages: assignment0.dtsx, assignment1.dtsx and assignment2.dtsx.

## 5.1 Assignment 0

**The assignment require us to return for every country, the number of total answers.**

The proposed solution requires first the reading of the Answers table through an **"Origin OLE DB" node**. From this table, we selected the column *AnswerId* and *UserId*, which plays the role of a foreign key to the table User. We then did a lookup on the User table using the join condition on *UserId*. This allowed us to extract the columns *GeoId* and then retrieve the *CountryName* values from the geography table. We then performed an **Aggregation node** to count the number of answers for each country. To do that, we simply needed to group by the *CountryName* and count the numbers of *AnswerId*. As final, we saved the result in a csv file.

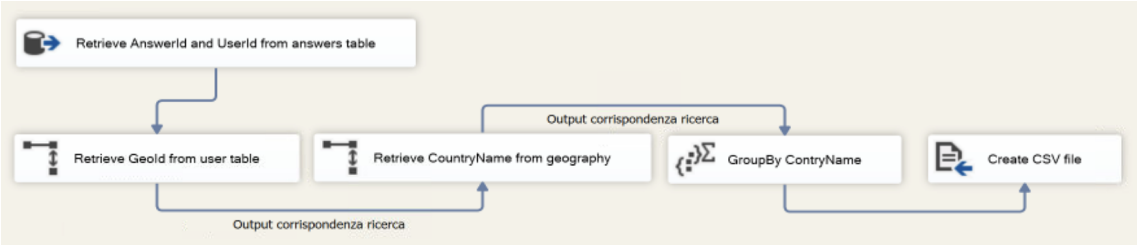



Figure 2: Assignment 0 - Data Flow

## 5.2 Assignment 1

**The assignment requires us to list all the groups (identified by GroupId) with an age mismatch, which means the difference of date of birth betwwen the yougest and the oldest student of the group is more than 365 days.**

Firstly, we get the *OrganizationId* and *GroupId* attributes from the Organization table. Then, we did a first **lookup** on Answers, imposing a mapping on *OrganizationId* and retrived *userId* from it. We did the same on the next **lookup** on User, and then Date table, getting the matching condition between *DateOfBirthId* and *DateId* and obtain the Date column. Since the program automatically parsed the values retrived as String data type, we had to perfom a data conversion to timpestamp in order to proceed for the next step. Now, with the correct data type we were able to calculate the *Min_Date* and the Max_Date for each *GroupId*. With those values, we performed a **conditional split** node, on one hand we have all the group meet the mismatch condition, on the other the othewise. We only keep the first bramch and saved it into a csv file.

## 5.3 Assignment 2

**The assignment requires us to return for each continent the ratio between correct answers of males and correct answers of females.**

We first used the **Origin OLE BD** node to read Answers and retrieve the *UserId* column (used for match with the User table) and *IsCorrect* (used later for computing the number of total correct answers). Then we perfomed a lookup for the User and Geography table in order to reach the Continent values. Afterwards we used the **Conditional Split** node to the seprate female

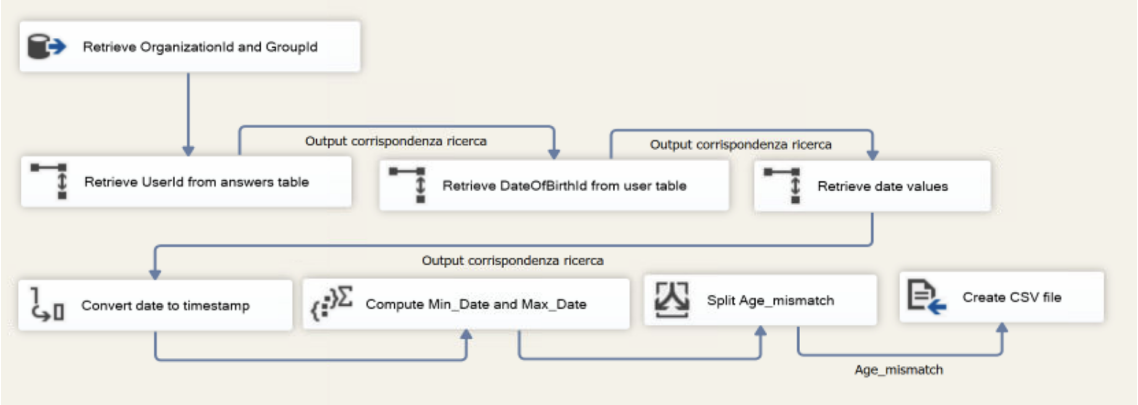



Figure 3: Assignment 1 - Data Flow

from male answers, next with the **Aggregation** nodes we grouped the data by *Continent* and summed all the *IsCorrect* answers for both male and female branch and performed a **sorting** task before **merge-join** everythings we obtained on the values of *Continent*, compute the *ratio* (Male_correct_answers/Female_correct_answers) and saved the values into a csv file.

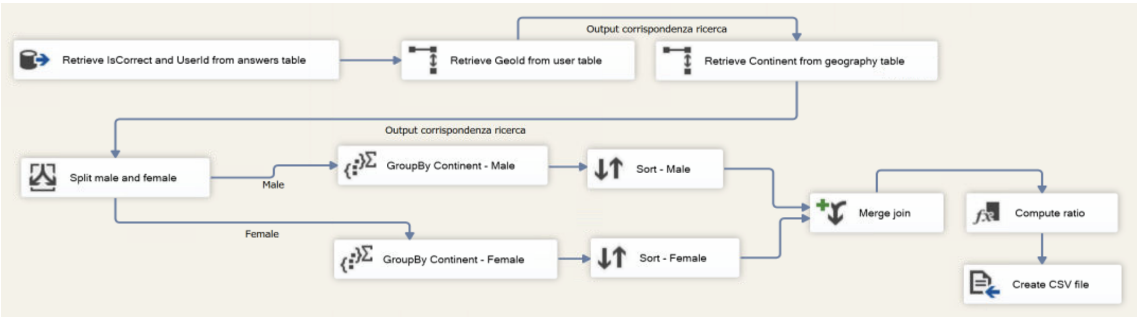



Figure 4: Assignment 2 - Data Flow

# Part 3 - Multidimensional Data Analysis

# 6 Creation of the OLAP Cube

## 6.1 Connecting with Database and creation of View

For the creation of the OLAP Cube, we've first connected to our database Group24_DB as data source. After having established the connection with the server side, we created a View of the whole database.

As next step, we decided to insert a calculated field into the Answers table, to facilitate some calculations and analysis later:

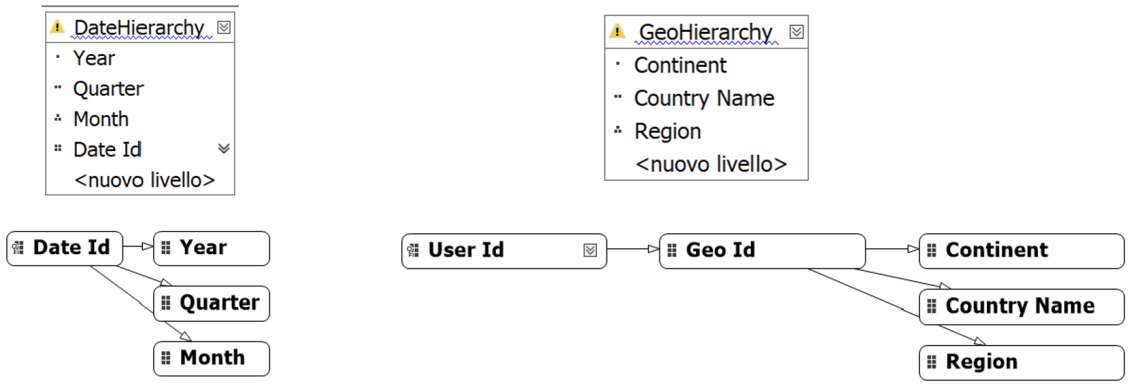
- **NotCorrect:** column with binary values 0 - 1, which is the opposite of the IsCorrect attribute:

$$\begin{cases} 1, & \text{if the attribute is wrong} \\ 0, & \text{otherwise} \end{cases}$$

## 6.2 Creation of Dimentions

For this step, we created the dimensions Organization, Subject, Date and UserGeo. All the dimensions contain the flat hierarchies of the attributes initially present in their respective tables. Within the Date dimension we also included the hierarchy renamed "DateHierarchy" having the following relations: $Year \rightarrow Quarter \rightarrow Month \rightarrow DateId$ [See Figure 5a].

A few more words should be spent on the UserGeo dimension. Due to some relational issues between the Geography and Answers tables, the geography dimension was ignored, where costructing the cube. To solve this problem, it was decided to put all the geography information in the dimension of User and call it UserGeo. In addition, the GeoHierarchy was created with the following relations: $Continent \rightarrow CountryName \rightarrow Region$ [See Figure 5b].




(a) Date

(b) Geography

Figure 5: Dimension hierarchies

After this step, we generated the OLAP cube, including some meassures needed for the analysis: *IsCorrect*, *NotCorrect*, *Answers count* and *Users count*. For the first two we set the AggregateFunction to "Sum" and the FormatingString to "Standard", while the last two was automatically set to "Count" by the system.

# 7 MDX Query

The following MDX queries are saved in the file named "MDXQueries_Group24.mdx".

## 7.1 Query 1

**Show the percentage increase or decrease in correct answers with respect to the previous year for each student**

For this query it was thought to obtain as a result a table like the one show in Table 1, with as row the userid of the students and as a column three values: the number of correct answer of 2019, 2020 and the variation in percentage of the two values.

| | is_correct_2019 | is_correct_2020 | variation |
|---|---|---|---|
| userID | | | |
| userID | | | |
| userID | | | |

Table 1: Idea of output

To accomplish this, we had to calculate a new member called variation with the following formula:

$$variation = \frac{correct\_answer\_of\_2020 - correct\_answer\_of\_2019}{correct\_answer\_of\_2019}$$

But since not always the same student took the exam both years, we had to handle null cases and cases with 0 correct answers, to avoid divisions by 0. To solve this, we used the case when statement [Listing 1: row 3-10]. After that we proceeded with the printing of the data as shown in Figure 6.

```
1  -- Assignment 1
2  WITH MEMBER difference AS
3  CASE
4  WHEN ([Measures].[Is Correct],[Date].[Year].&[2020]) = 0 then null
5  WHEN ([Measures].[Is Correct],[Date].[Year].&[2020]) = null then null
6  WHEN ([Measures].[Is Correct],[Date].[Year].&[2019]) = 0 then null
7  WHEN ([Measures].[Is Correct],[Date].[Year].&[2019]) = null then null
8  ELSE (([Measures].[Is correct],[Date].[Year].&[2020])-([Measures].[Is correct],[
       Date].[Year].&[2019]))
9                /([Measures].[Is correct],[Date].[Year].&[2019])
10 END,
11 format_string="percent"
12
13 MEMBER is_correct_2019 AS
14 ([Date].[Year].&[2019],[Measures].[Is Correct])
15
16 MEMBER is_correct_2020 AS
17 [Measures].[Is Correct] - is_correct_2019
18
19 SELECT filter([UserGeo].[User Id].[User Id], difference <> null) on rows,
20 {is_correct_2019, is_correct_2020, difference} on columns
21 FROM[Group 24 DB]
```

Listing 1: Query 1

| | is_correct_2019 | is_correct_2020 | difference |
|---|---|---|---|
| 38 | 75.00 | 45.00 | -40.00% |
| 99 | 4.00 | 63.00 | 1475.00% |
| 132 | 21.00 | 7.00 | -66.67% |
| 164 | 4.00 | 1.00 | -75.00% |
| 182 | 27.00 | 1.00 | -96.30% |
| 199 | 6.00 | 11.00 | 83.33% |
| 218 | 185.00 | 56.00 | -69.73% |
| 274 | 5.00 | 32.00 | 540.00% |
| 323 | 112.00 | 87.00 | -22.32% |

Figure 6: Example of output

## 7.2 Query 2

**For each subject show the total correct answers in percentage with respect to the total answers of that subject.**

For this part, we had to compute the correct answer percentage of each subject, to obtain this we did a simple division:

$$correct\_percentage = \frac{Number\_of\_correct\_answer\_of\_given\_subject}{Number\_of\_answer\_of\_that\_subject}$$

and then printed the table by put the calculated value on the column and the Subject on the row.

```
-- Assignment 2
WITH MEMBER correct_percentage AS
([Measures].[Is Correct]/[Measures].[Conteggio di Answers]),
format_string = "Percent"

SELECT correct_percentage on columns,
[Subject].[Subject Id].[Subject Id] on rows
FROM [Group 24 DB]
```

Listing 2: Query 2

| | correct_percentage |
|---|---|
| Maths, Advanced Pure, Functions, Composite Functions | 37.64% |
| Maths, Advanced Pure, Functions, Function Notation | 42.21% |
| Maths, Advanced Pure, Functions, Inverse Functions | 42.36% |
| Maths, Algebra, Algebraic Fractions, Adding and Subtracting Algebra... | 57.86% |
| Maths, Algebra, Algebraic Fractions, Multiplying and Dividing Algebra... | 39.59% |
| Maths, Algebra, Algebraic Fractions, Simplifying Algebraic Fractions | 55.85% |
| Maths, Algebra, Algebraic Fractions, Simplifying Algebraic Fractions | 55.13% |

Figure 7: Example of output

## 7.3 Query 3

**Show the students having a total incorrect answers greater or equal than the average incorrect answers in each continent.**

For this part, we first computed the percentage of incorrect answers of each continent by of each subject by calculate the following formula:

$$avg\_incorrect = \frac{sum\_of\_incorret\_answer\_of\_students\_by\_continent}{Number\_of\_student\_in\_that\_continent}$$

and then printed the table by putting the the number of incorrect answers and the calculated value on the column and on the rows the UserId with respect of the continent. The showed data are filtered by avg_incorrect.

| | | Not Correct | avg_incorrect |
|---|---|---|---|
| EU | 35 | 26.00 | 14.31 |
| EU | 38 | 69.00 | 14.31 |
| EU | 99 | 91.00 | 14.31 |
| EU | 218 | 24.00 | 14.31 |
| EU | 257 | 61.00 | 14.31 |

Figure 8: Example of output

```
-- Assignment 3
WITH MEMBER avg_incorrect AS
(([UserGeo].[Continent], [UserGeo].[User Id].[ALL],
[Measures].[Not Correct])/([UserGeo].[Continent], [UserGeo].[User Id].[ALL], [
    Measures].[Conteggio di User])),
format_string = "standard"

SELECT {[Measures].[Not Correct], avg_incorrect} on columns,
filter (([UserGeo].[Continent].[Continent], [UserGeo].[User Id].[User Id]),
[Measures].[Not Correct] >= avg_incorrect) on rows

FROM [Group 24 DB];
```

Listing 3: Query 3

# 8 Dashboard - Power BI Solution

We first established a connection to the server and choosing Analysis Services option as data source. After selecting our Group 24 DB Cube, Power BI showed us the Cube's structure on the right side. This allowed us to draw the plots of the following sections.

## 8.1 Assignment 4

As asked by the assignment, we implemented a dashboard to show the geographical distribution of the correct answers and incorrect answers for each continent. Thus, we chose to use the "map" graph, as it meets all the need required. The plot in Figure 9 shows a geographyc map with a series of circle, each circle represent a region in the map. The size of the circle is directly proportional to the number of answers. The larger the circle, the more answers there are in that region. The balls were then grouped by countries, to add additional level of signicativity.


Figure 9: Map of distribution of answers (Correct and Incorrect ones)

In the Figure 10 we have a zoom-in on Europe, if we hover on a bubble it will show additional details, such the numer of Correct and Incorrect answers, the total answers and the name of the region and country those date belong to.

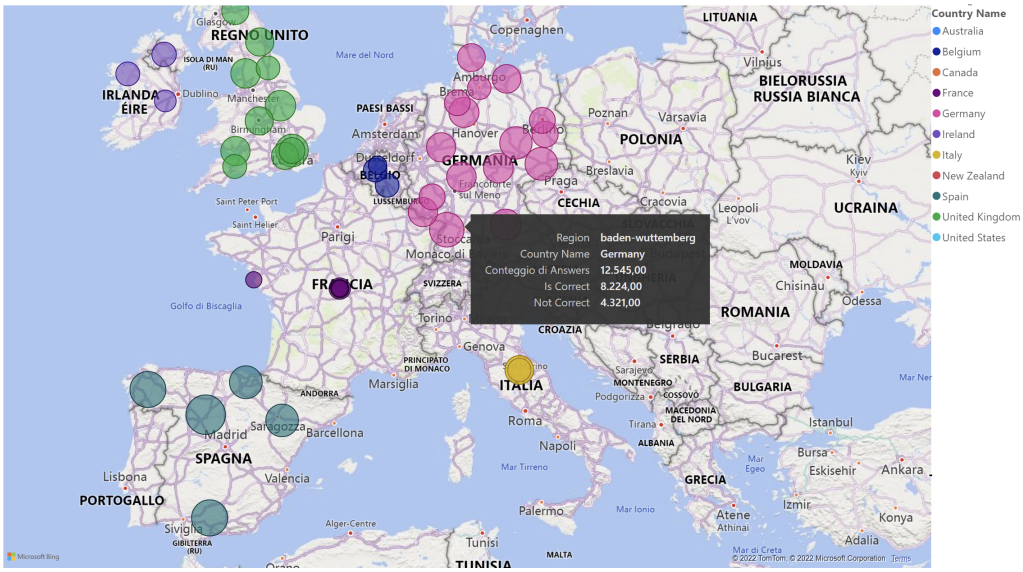



Figure 10: Zoom of the map to Europe

## 8.2 Assignment 5

For this last assignment, we decided to use a bar plot to show the distribution of correct answer among the subjects and continent. Each bar indicates the number of correct answers for that subject and the colors correspond to the continents shown in the legend above.

We think that this kind of plot could be useful for statistical purposes for institutions and governments, to understand where and which are the subjects with the most errors, to have a first insight into the possible causes that have produced these gaps. In our case in particular, the data were hardly significant by a geographical point of view, as it is highly unbalanced, with the majority of data centered on Europe.

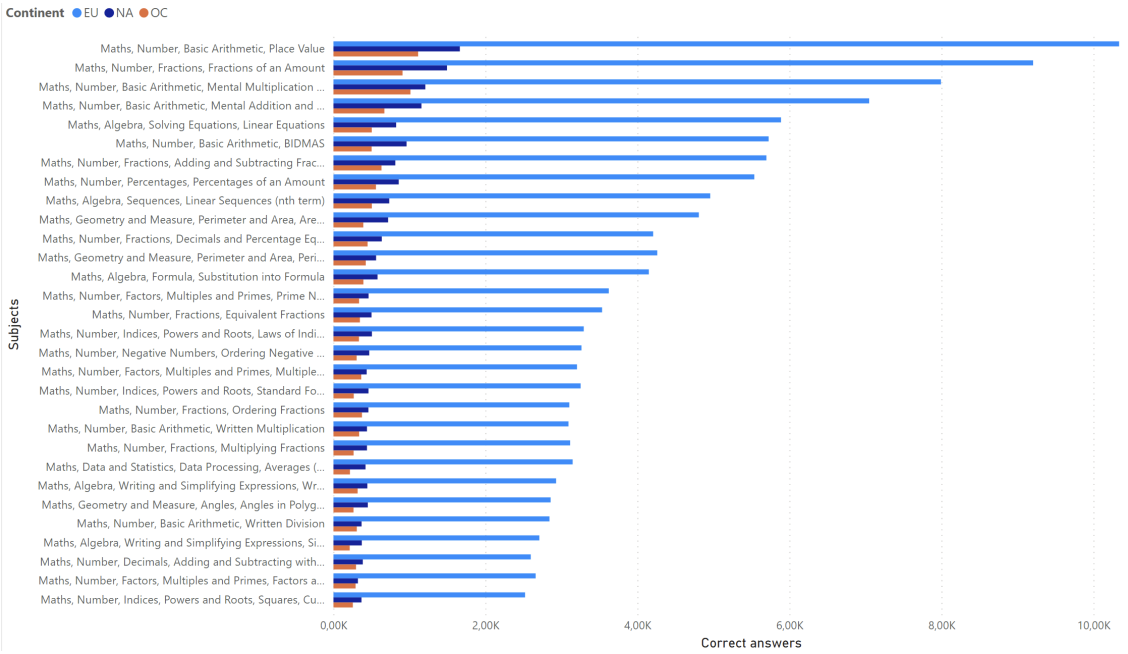



Figure 11: Correct answers by Subjects and Continent