Università di Pisa

Dipartimento di Informatica

Laurea Magistrale in Data Science and
Business Informatics

Laboratory of Data Science

Answer Data Warehouse

Authors

Lia Trapanese 628153
Giovanni Battista D'Orsi 639186

2022/2023

Contents

1 Data Warehouse Creation

The project was carried out on two datasets, one with data on students' responses to various multiple-choice questions and the other with information about the subjects. In the first dataset there was also all information about the students, the questions, answers and references to question subject.

1.1 Assignment 0 - Database Schema Creation

The first assignment of the project requested us to create the schema for our specific database. First, we explored the dataset, checking missing values and duplicates, and we analyzed each feature in detail. We notice that there were none of them and the dataset was very cleaned. Then we created all the tables (fact table and his dimensions) in SQL Server Management Studio, matching attributes types, specifying the relation between primary keys and foreign keys and imposing the NOT NULL constraint to the keys, while the other attributes are allowed to have NULL values.

The Figure 1 illustrates the entire schema, with the relationships between fact table and dimensions.

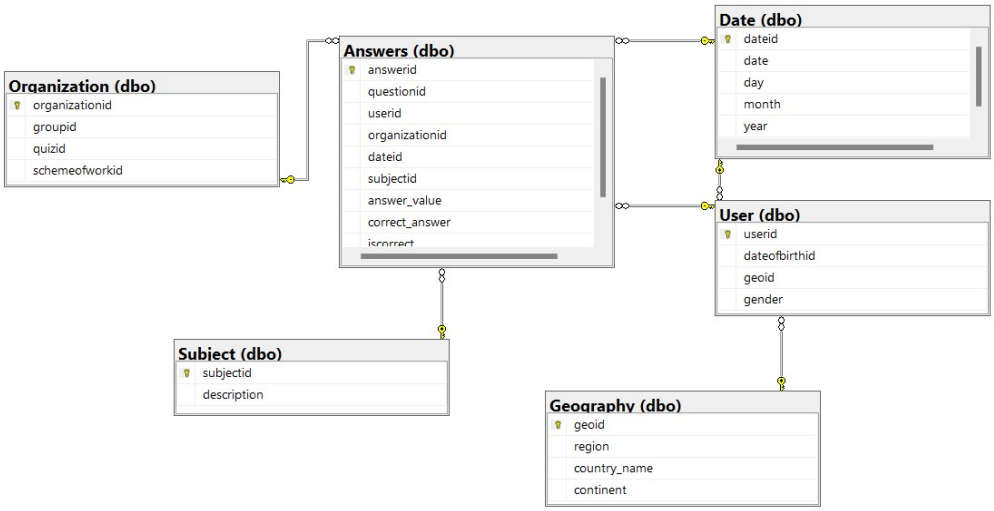


Figure 1: Star Schema

1.2 Assignment 1 - Tables' Creation

The second task was to create the six tables to populate the schema created in the previous section. We divided the dataset contained in the files answerdatasetnew.csv and subject_metadata.csv. The six tables were Answers (fact table), Date, Organization, Subject, Users and Geography and they have been created without using the pandas library. But before this, we created the "funzioni.py" file, in which we defined some functions, used in this specific assignment. First, we created the tables of the outermost dimensions of the star schema, and then generated the fact table. In the

following subsections, we will explain in particular how we created each table and how the funzioni.py file has been used.

1.2.1 Geography

The table "Geography" has the columns "geoid" as primary key, "region," "country_name," and "continent" as attributes. The second refers to the geographical information of the user answering a question and was taken from the file answerdataset-new.csv, while we had to generate or retrieve the geoid, country_name and continent. As for the "continent," the data was taken from the following source: https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes/blob/master/all/all.csv. Using this support file, we created a dictionary that has country_code as its key and a list containing country and continent as its value. It was necessary to manually enter the continent for "United Kingdom" because of a mismatched value ("Great Britain"). The dictionary key was used to match the "CountryCode" of the table answerdataset-new.csv with that of the external file to get the continent and country name in the final table. As for the "geoid," which represents the various regions, we chose to create a unique key by concatenating the "CountryCode" value with the "RegionId." In addition, we used a function called "codificaRegione" from the funzioni.py file that allowed us to add to this concatenation an incremental value uniquely associated with each region. The resulting table consists of 76 rows and 4 columns.

1.2.2 Date

The "Date" table consists of six simple attributes: "dateid" (the primary key), date, day, month, year, and quarter, and contains data on the user's date of birth and the date on which a given response was given. We read the values of the "DateOfBirth" and "DateAnswered" columns and created a single "date" column that collects them both so that we could get the information from the other tables. These dates subsequently were converted to the GGMMAAAA format and used as the dateid. We obtained the year, month and day values by taking the necessary characters from the "date" string. To extract the "quarter" attribute, we used a function that reads the month from the "date" and assigns the corresponding quarter. We have only 4 unique values(Q1,Q2,Q3,Q4). The resulting table consists of 596 rows and 6 columns.

1.2.3 Organization

The "Organization" table consists of four attributes: "organizationid" (primary key), "groupid," "quizid," and "schemeofworkid." Since we assumed that an organization is identified by the latter three attributes, we generated "organizationid" by concatenating them. Combining the three attributes could have created the same key multiple times, making it non-unique, this would have led to the loss of information. To solve this problem, we included the underscore in the primary key. The resulting table consists of 24640 rows and 4 columns.

1.2.4 Subject

The table "Subject" has only two attributes: "subjectid" as the primary key and "description." We had to create both attributes, so first we read from answerdatasetnew.csv the "SubjectId," which consists of a list of numbers, indicating the various topics on which the questions are about. We used a dictionary "diziosubj" as a data structure, entering all the unique values (lists of numbers) of SubjectId considering incremental values as keys. After that we created a dictionary which maps to each subjectid a list with the respective subject in string format and the level to which this belongs; both obtained from the subject_metadata.csv file. Finally, to obtain the final table, we chose as id the key of the first dictionary (incremental) and then modified its values by going to replace the list of subjects(with integers) with the list of subjects, in string format sorted according to the level contained in the second dictionary. Thus we got the description with inside all the subjects sorted by level. The resulting table has 412 rows and 2 columns.

1.2.5 Users

The penultimate dimension we created is "Users" because it is composed of the values from other tables: we took the unique values "userid" and "gender" from answerdatasetnew.csv, from the latter we also took the column "DateOfBirth." Firstly, we changed the gender in string type where 0 correspond to the "Neutral" gender, 1 to "Female" and 2 to "Male". In the table "Dates" we created a dictionary that has "date" as the key and "dateid" as the value in the format DDMMYYYY. So as to match the user's year of birth with the "dateid" in the table to extract the "dateid". Another column in the user table is "geoid" obtained by matching with the Geography table. Again we used a dictionary that has "region" as a key and "geoid" as a value so as to match the "region" in the answerdatasetnew.csv table referring to the user, with the one in the Geography table, and thus be able to trace through geoid the country and continent of the user. We obtained a table composed of 13630 rows and 4 columns.

1.2.6 Answers

The very last table was "Answers", the fact table. We took most of the attributes from the answerdatasetnew.csv file, like "answerid" (primary key), "questionid", "userid", "answer_value", "correct_answer" and "confidence", but we needed also to create some of them, like foreign keys (organizationid, dateid, subjectid) or the "iscorrect" attribute. We used the "createDic" function from the funzioni.py file to take all the dateid's and subjectid's from the Date.csv and Subject.csv files, while we just concatenated "groupid", "quizid" and "schemeofworkid" to create "organizationid". We obtained "iscorrect" simply using an if statement, which checked if the "answer_value" and "correct_answer" values were the same (iscorrect == 1) or not (iscorrect == 0). The resulting table consists of 538835 rows (same as the original dataset) and 9 columns.

1.3 Assignment 2 - Loading Data

Once we had created all the tables, we loaded them into SQL Management Studio by running a function ("funzione_load") to load all of them. We established a connection

to the remote database and then executed the query INSERT INTO Table VALUES using a cursor that was closed at the end of the loading process.

2 SSIS

The second part of the project consisted in solving three problems using Sequel Server Integration Services (SSIS).

2.1 Assignment 0

For every subject, the number of correct answers of male and female students.

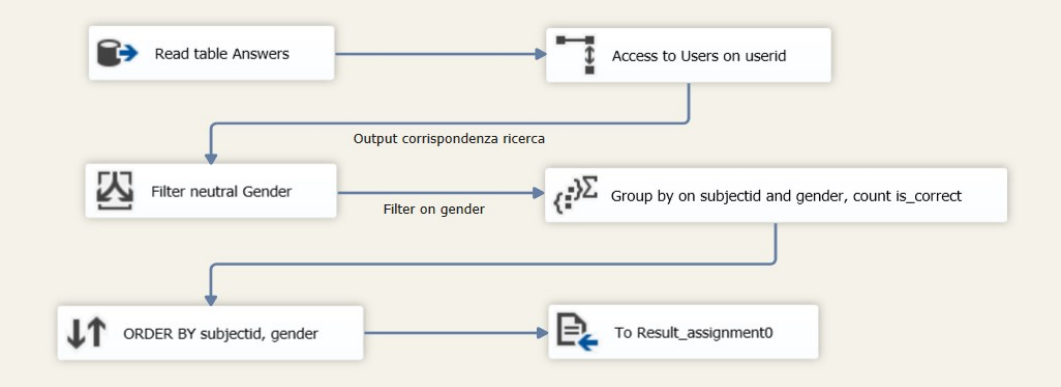


Figure 2: SSIS Assignment 0

In order to obtain the number of correct answers of male and female students for each subject, we first accessed the Answers table to select the **userid**, **iscorrect** and **subjectid** columns. We did a lookup on the **userid** of the Users table to access the genre. Wanting only males and females, we filtered out the gender by not considering "Neutral". We used the group_by operator to calculate the total number of correct answers for each gender and **subjectid**. To return the result in an ordered manner, we applied an order_by on the **subjectid** and gender fields. Finally, the results were written to a "Destination flat file".

2.2 Assignment 1

A subject is said to be easy if it has more than 90% correct answers, while it is said to be hard if it has less than 20% correct answers. List every easy and hard subject, considering only subjects with more than 10 total answers.

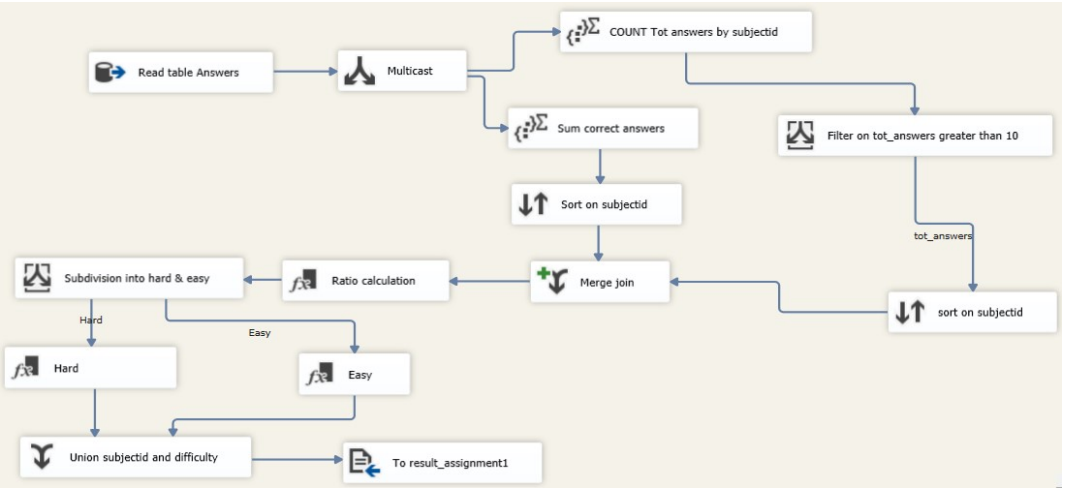


Figure 3: SSIS Assignment 1

We started by reading the Answer table from the database using an "Origin OLE DB" node and then selected the columns **subjectid** and **iscorrect**. Then, we used a "Multicast" node in order to calculate in parallel two parameters: on one hand, we computed the **tot_corr_ans** by grouping over **subjectid**, in order to get the number of correct answers given by subject. On the other hand, we computed **tot_answers** by grouping over **subjectid** and counting the number of answers given. On the latter we applied a conditional split to remove those subjects having less than 10 answers. We then merged the two resulting output in a single table with a "Merge Join" operator. The previously computed variables were used to compute a variable called "ratio" (which is the ratio between tot_corr_ans and tot_answers). Lastly we applied another conditional split in order to identify "Hard" (ratio > 0.9) and "Easy" (ratio < 0.2) subject. The labeled results were then merged in a new variable called **difficulty**. Finally we saved the table in a .txt file, using a "Flat File Destination".

2.3 Assignment 2

For each country, the student or students that answered the most questions correctly for that country.

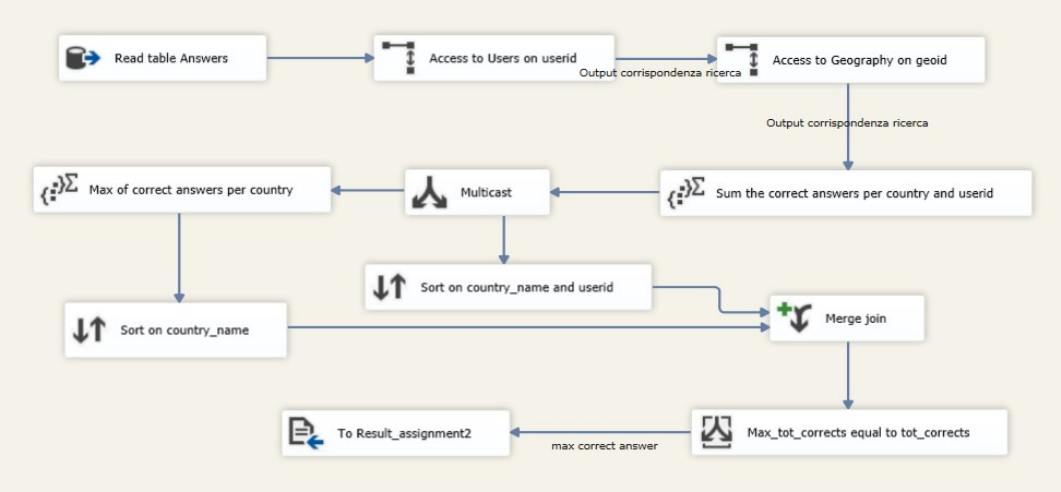


Figure 4: SSIS Assignment 2

In order to solve the last assignment, we started with a "Origin OLE DB" node, selecting the **userid** and the **iscorrect** column. Then, we used two different "Lookup" on Users and Geography tables, to extract information about **country_name**. In order to calculate the total number of correct answers, we used an "Aggregate" node, creating the variable **Tot_corrects**, with a SUM operation on iscorrect and grouping on country_name and userid. Then, we used a "Multicast" node: from one side, we just sorted country_name and userid through a "Sort" node, while on the other side we first used an "Aggregate" node to take, for each country, the maximum values of the variable Tot_corrects and saving them in a new variable called **Max Tot_corrects**, and then we sorted it ascending on country_name. After this, we merged the two tables with a "Merge join" node on country_name and we created a new column called **max correct answer**, containing the max values. In the end, we saved the table in a .txt file, using a "Flat File Destination".

3 Multidimensional Data Analysis

3.1 OLAP Cube

3.1.1 Assignment 0

For the creation of the OLAP cube we created a new project named 'LDS_Cube_Group_11'. We built the connection with the server 'lds.di.unipi.it' and then we accessed to the Group_11_DB. After having created a view on the data selecting all the tables, we developed the dimensions. We created a derivative column named wrong_answer which is obtained by applying the following SQL query:
"**CASE WHEN** iscorrect = 0 **THEN** 1 **ELSE** 0 **END**"

We created the Date and Users dimensions, but we decided to add the needed dimension (Subject) for answering the business questions. Within the dimension Users, we created two hierarchies: UserGeographyHier which is a geographical hierarchy and, DateofBirth which builds a user date of birth hierarchy. The first has the following relation: Userid → Region → Country name → Continent

For the second, each user is associated to a specific date, composed by Day, Month, Quarter and Year.

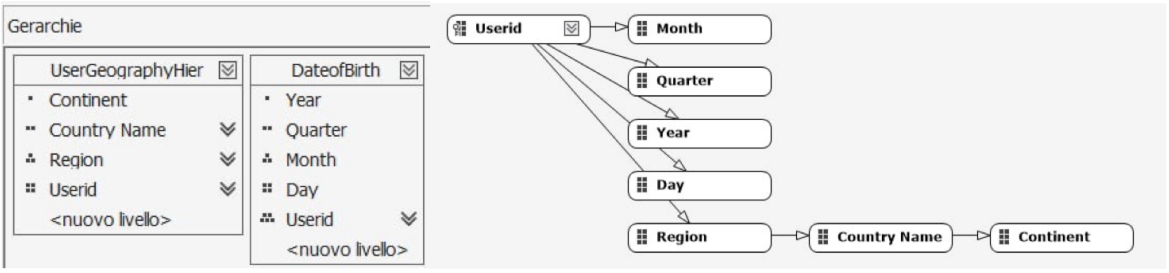


Figure 5: UserGeographyHier and DateofBirth within Users dimension

Instead, within the Date dimension, we created the DateofAnswer hierarchy, which contains temporal information about the time when the answer was given. For all the temporal hierarchies, to guarantee the order between numerical attributes such as year, month, and days, we set in Property the variable 'OrderBy' to the value 'Key'. Below we show the created hierarchy:

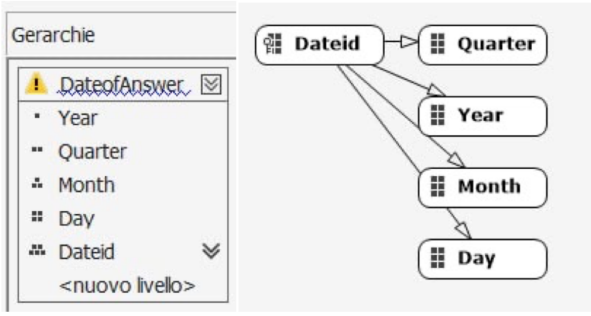


Figure 6: DateofAnswered hierarchy within Date dimension

After these steps we built the OLAP cube named Group_11_DB, making sure to include in the fact table Answer measures needed for the analysis: **sum_correctAnswer, sum_wrongAnswer, Conteggio di Answer**. The first two were obtained by setting the aggregate function to 'SUM' while the last one corresponds to a row count on the fact table. Once the cube was ready we performed the deployment.

3.2 MDX Query

3.2.1 Assignment 1

Show the student that made the most mistakes for each country

```
select [Measures].[sum_wrongAnswer] on columns,
nonempty (generate(
[Users].[Country Name].[Country Name],
topcount (([Users].[Country Name].currentmember, [Users].[Userid].[Userid]), 1 ,[Measures].[sum_wrongAnswer])

)) on rows
from [Group_11_DB]
```

Figure 7: MDXQuery1

In order to obtain the total number of mistakes made from all the students, we created a new measure in the OLAP Cube, called **sum_wrongAnswer**, which made the sum of all the incorrect answers (this measure derived from an other created column, which set as "1" all the incorrect answers). Then we selected this measure on columns, while on the rows we used a "generate" function which takes as input two expressions and each one returns a set. In the first expression we specified the country name, while in the second we found the user with the highest mistakes for each country by using the topcount function applied on the **sum_wrongAnswer** measure. Query solution can be seen in the following figure:

		sum_wrongAnswer
Australia	23352	331
Belgium	86016	256
Canada	21412	254
France	53200	242
Germany	113435	254
Ireland	79098	273
Italy	67433	392
New Zealand	96080	216
Spain	105010	330
United Kingdom	54492	502
USA	72434	278

Figure 8: Solution of MDXQuery1

3.2.2 Assignment 2

For each subject, show the student with the highest total correct answers.

```
with member max_answer as
MAX(([Subject].[Description].currentMember, [Users].[Userid].[Userid]),[Measures].[sum_correctAnswer])

select max_answer on columns,
nonempty(
    filter(
        ([Subject].[Description].[Description], [Users].[Userid].[Userid]),
        max_answer=[Measures].[sum_correctAnswer])
        ) on rows
from [Group_11_DB]
```

Figure 9: MDXQuery2

For this business question, we decided to use the command 'with member' to calculate the maximum of correct answers by users for each subject. We considered the field description to better visualize the results. Then, we selected the measure max_answer on columns and filtered to identify the user with the highest correct answer for each

subject. Obviously, we found more users for a subject because the scores were equal. Query solution can be seen in the following figure:

		max_answer
[Maths - Advanced Pure - Functions - Composite Functions]	97793	3
[Maths - Advanced Pure - Functions - Function Notation]	77990	8
[Maths - Advanced Pure - Functions - Inverse Functions]	85945	3
[Maths - Algebra - Algebraic Fractions - Adding and Subtracting Algebra...	18052	9
[Maths - Algebra - Algebraic Fractions - Adding and Subtracting Algebra...	109348	9
[Maths - Algebra - Algebraic Fractions - Multiplying and Dividing Algebr...	18052	4
[Maths - Algebra - Algebraic Fractions - Simplifying Algebraic Fractions]	24738	16
[Maths - Algebra - Algebraic Fractions - Simplifying Algebraic Fractions]	61809	16
[Maths - Algebra - Algebraic Fractions - Solving Equations with Algebrai...	107293	5
[Maths - Algebra - Co-ordinates - Co-ordinates-Others]	83476	8

Figure 10: Solution of MDXQuery2

3.2.3 Assignment 3

For each continent, show the student with the highest ratio between his total correct answers and the average correct answers of that continent.

```
with member avg_corr_ans_by_cont as
    ([Users].[UserGeographyHier].currentmember.parent.parent.parent, [Measures].[sum_correctAnswer])/
    ([Users].[UserGeographyHier].currentmember.parent.parent.parent, [Measures].[Conteggio di Answers])


member user_ratio as
([Users].[UserGeographyHier].currentmember, [Measures].[sum_correctAnswer])/avg_corr_ans_by_cont

member max_user_ratio as
    max(([Users].[UserGeographyHier].currentmember.parent.parent.parent,
        [Users].[Userid].[Userid]),
        user_ratio
        )

select {avg_corr_ans_by_cont, user_ratio, max_user_ratio, [Measures].[sum_correctAnswer]} on columns,
nonempty(
    filter(
        ([Users].[Continent].[Continent], [Users].[UserGeographyHier].[Userid] ),
        user_ratio = max_user_ratio)
    ) on rows
from [Group_11_DB]
```

Figure 11: MDXQuery3

In order to return the output required by the query, we created three members: user_ratio, avg_corr_ans_by_cont and max_user_ratio. The first calculates the average correct answer per continent (which corresponds to the sum of correct answers per continent divided by the total number of answers given for each continent) through the UserGeography hierarchy. The continent level is accessed starting from the current member (in this case corresponding to userid) and going up to the continent level using the 'parent' command. The second member calculates, for each user, the ratio of the user's total correct answers to the avg_corr_ans_by_cont of the continent associated with the user. Through the max function, the last member returns for each continent the

highest user_ratio. In the select, we used the nonempty function to avoid null fields in the result, and the filter to identify the users whose ratio is the highest in their continent through the condition (user_ratio = max_user_ratio). Query solution can be seen in the following figure:

		avg_corr_ans_by_cont	user_ratio	max_user_ratio	sum_correctAnswer
Europe	99333	0.634866662980778	1083.69212012134	1083.69212012134	688
North America	39367	0.625782488272682	938.025609537698	938.025609537698	587
Oceania	24738	0.651799650002365	794.72273419926	794.72273419926	518

Figure 12: Results of MDXQuery3

3.3 Power BI Dashboards

As final tasks the data available in the cube have been explored in a report produced from a multidimensional view. The reporting tool chosen is Power BI.

3.3.1 Assignment 4

In figure 13 is possible to visualize the geographical distribution of correct answers and wrong answers for each country. We used the two measures created in the Cube: **sum_correctAnswer** and **sum_wrongAnswer**.

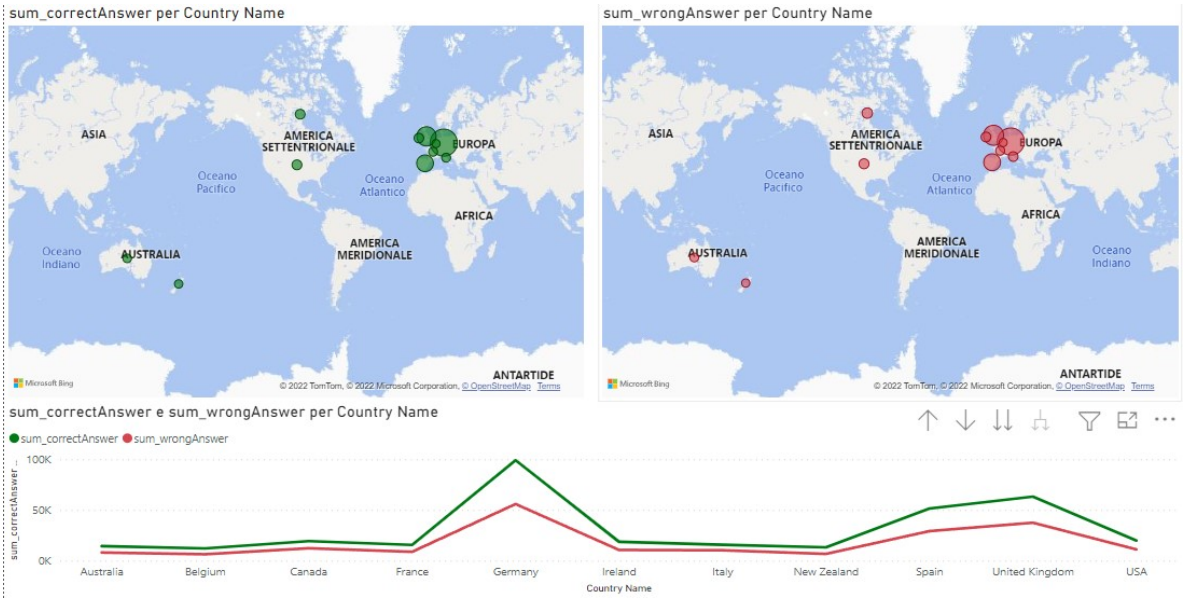
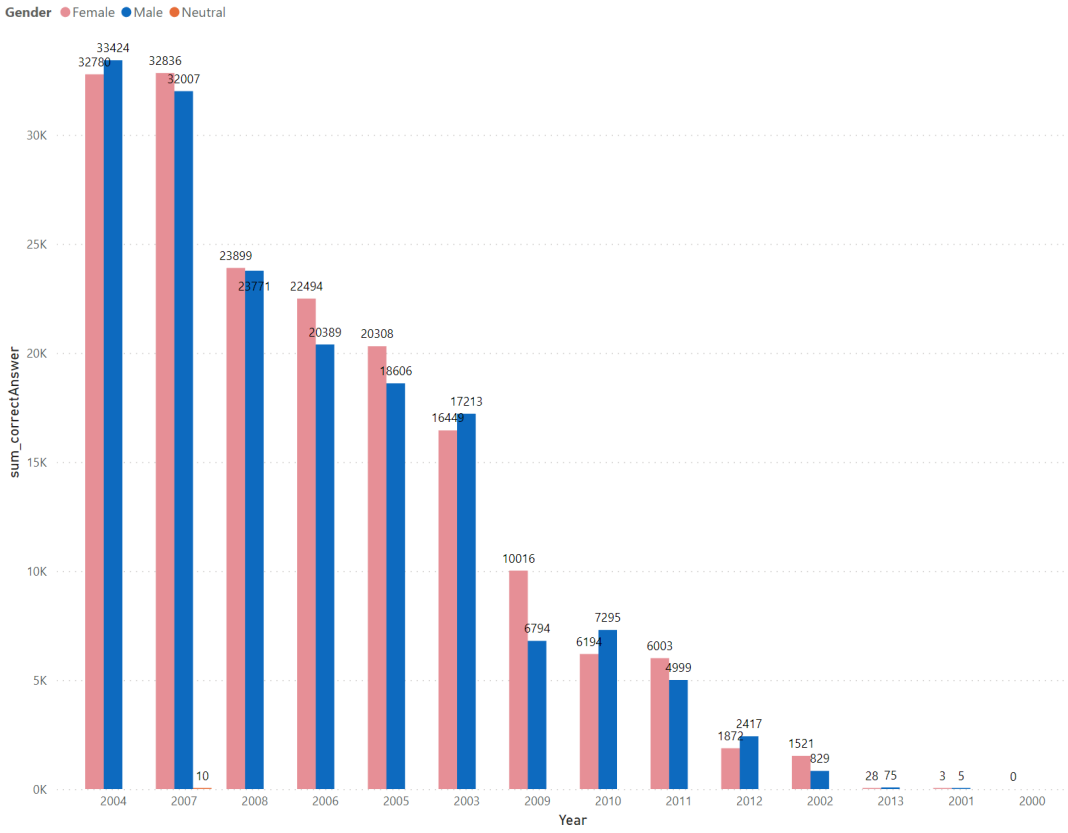


Figure 13: Geographical distribution of correct answer and wrong answers.

The bubbles are located in Europe, Australia, and United States. Comparing correct answers with incorrect answers, as also seen from the line plot, the trends of the straight lines are equivalent in contrast to the higher peak of correct answers in Europe. In particular, we can visualize it in Germany, Spain, and the United Kingdom where there are many more correct answers than in the other countries, but especially in the other continents. It is also a good result to display fewer wrong answers than correct ones. Since the bubbles may look similar in size, we decided to add a line plot in order to visualize the difference in quantity.

3.3.2 Assignment 5

For the last assignment, we chose to analyze the correct answers and incorrect answers distribution by gender, using first the temporal attribute users' year of birth and then the attribute subject. As we can see in the upper part of Figure 14, regarding the distribution of correct answers, we noticed that the results are almost equivalent for the two genders (male and female); the female users born in 2006, 2005 and 2009 gave more correct answer than the male users born in the same years. Furthermore, the users born in 2004 and 2007 are those that gave the highest number of correct answers. Focusing on the distribution of wrong answers, shown in figure Figure 14, we can see that male users born in 2004, 2007, 2003, and 2010 gave more incorrect answers than female users; by both the distribution it is also noticeable that very few users born in 2013,2001, and 2000 took a test.



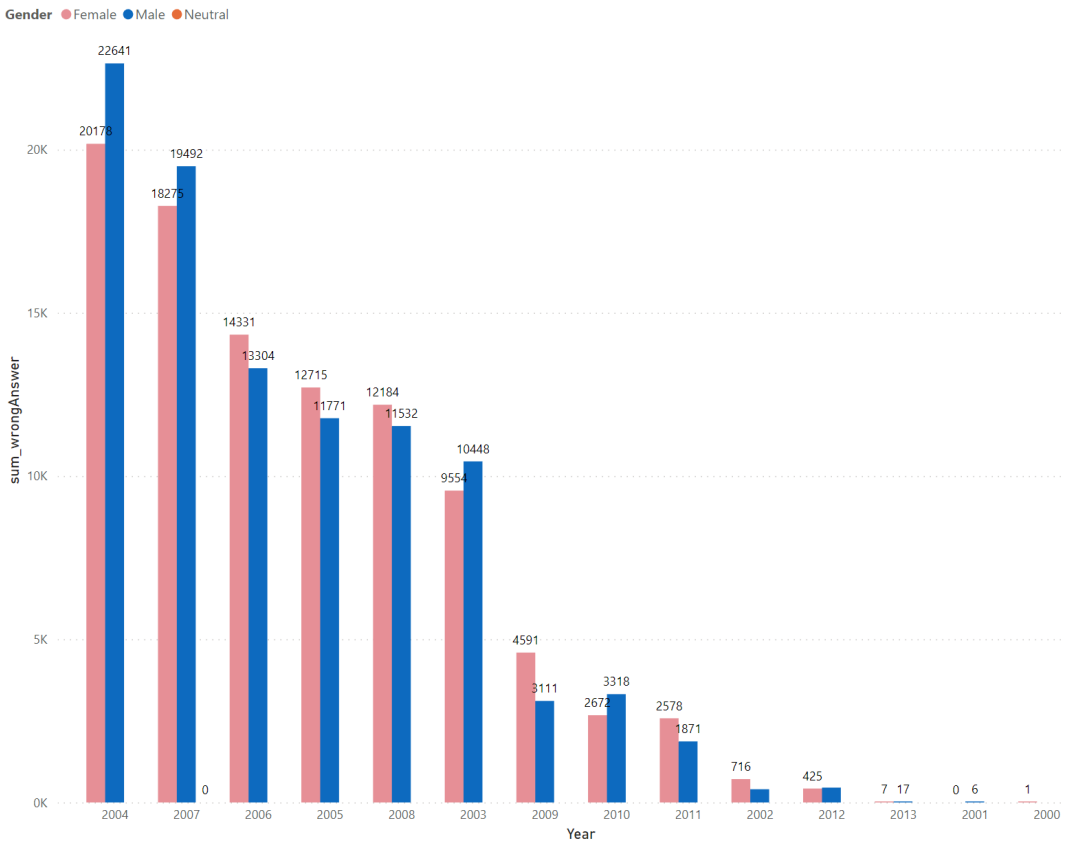


Figure 14: Year of birth and Gender distribution of correct and incorrect answers.

Analyzing the subject distribution, we created the plots in Figure 15 and in Figure 16. Looking at the first plot, as we saw in the previous analysis, in some cases female users gave more correct answers than male users; we also noticed that the subject in which we had the majority of correct answers was the "Number" one, including "Basic Arithmetic", "Fractions" and "Solving Equation". On the other side, in the second plot, we could see that male users had more wrong answers than female users, except for "Geometry" and some "Number" questions. In terms of Subjects, in this case we had a more distributed plot: we had again some "Number" questions (the majority part of the answers are about this subject), but we also had "Algebra", "Geometry, and Measure" (which was not in the correct answer plot) and "Data Statistics".

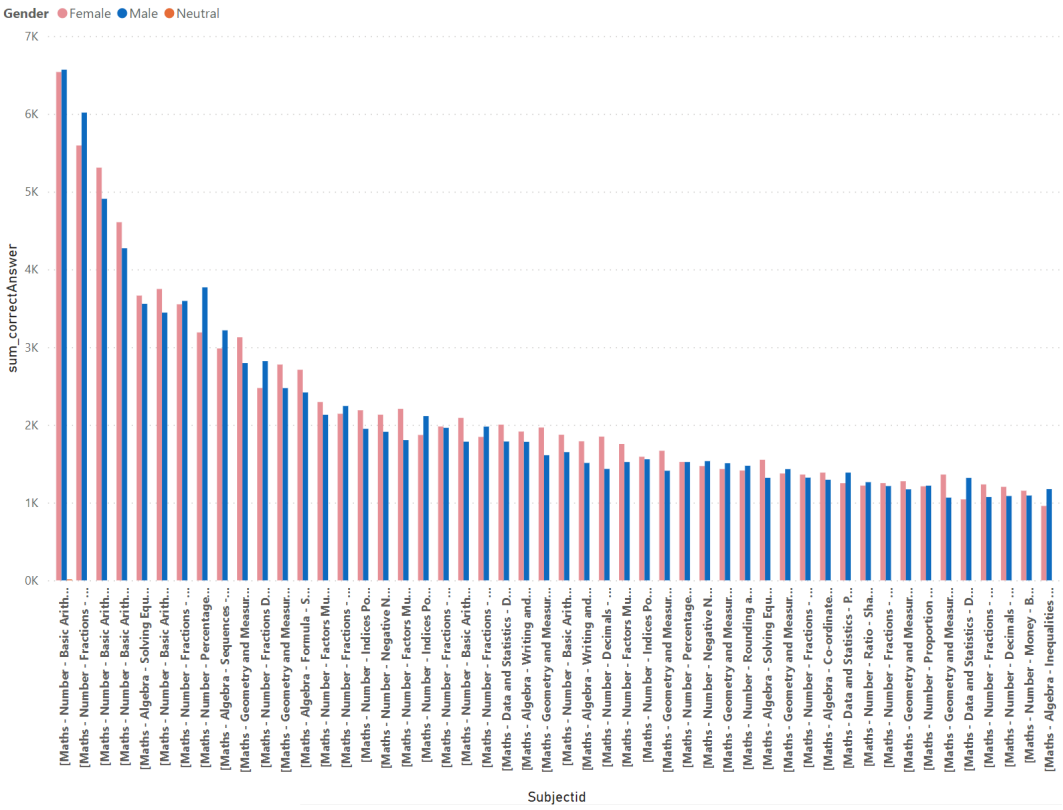


Figure 15: Subject and Gender distribution of correct answers.

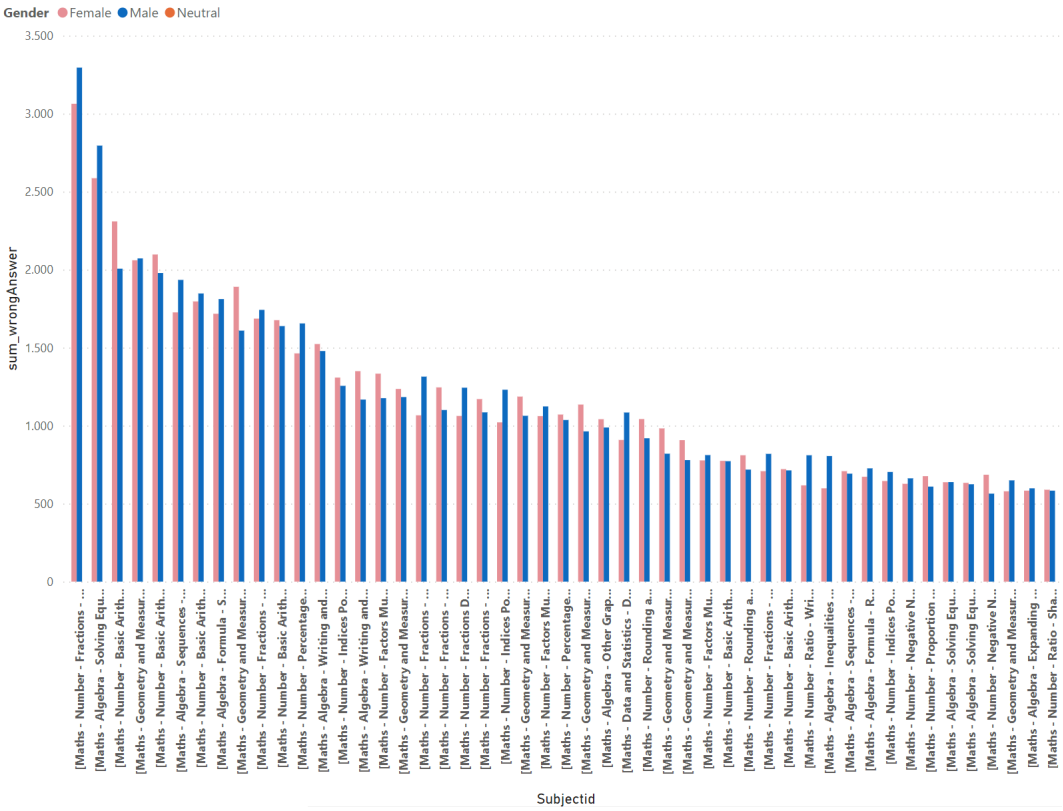


Figure 16: Subject and Gender distribution of incorrect answers.