Assignment:
Open Food Facts

Data Warehouse

Course 2023/24

Víctor Morcuende Castell

Guillermo Nájera Lavid

Table of Contents

1. Introduction

In the realm of data-driven decision making, the effective organization and analysis of large datasets stand as a cornerstone. This report presents a comprehensive overview of our project, undertaken as part of the Data Warehouse subject, which revolves around the intriguing domain of data warehousing with a specific focus on the Open Food Facts dataset. The primary objective of this project is to design and develop a robust data warehouse that enables intricate analysis and provides insightful observations about food products and their various attributes.

The Open Food Facts dataset offers a rich source of information about food products. It includes diverse data points ranging from basic product details to intricate nutritional information. Our project leverages this dataset to build a multidimensional model that facilitates complex queries and analytical operations, ultimately aiming to extract meaningful patterns and trends.

To achieve this, we have employed Pentaho Data Integration (Kettle) for the Extract, Transform, Load (ETL) processes, meticulously extracting data from the source, transforming it to suit our analytical needs, and loading it into a well-structured data mart. This data mart is designed with a focus on scalability and efficiency, ensuring that it not only addresses our current analytical queries but is also robust enough to handle future expansions.

Two pivotal components of our project are the Mondrian cubes. These OLAP (Online Analytical Processing) cubes are tailored to dissect and analyze the data through multiple dimensions, providing a versatile tool for our analytical queries.

This report will delve into the intricate details of our project, discussing the design choices, methodologies, and technologies employed. It will also showcase the results obtained from our analytical queries, illustrating the practical applications and insights drawn from our data warehouse. Through this endeavor, we aim to demonstrate not only the technical prowess in handling and analyzing large datasets but also the potential of data warehousing in uncovering hidden trends and informing decision-making processes in the food industry.

2. Design and implementation

2.1. UML Class Diagram

2.1.1. Initial Design

To tackle a project of this size, we first started by understanding and getting familiar with the Open Food Facts (from this point called "OFF") dataset, so we could later achieve a proper design and creation of not only our Mondrian cubes (final task), but also our data mart and MDX queries.

After familiarizing profoundly with the OFF dataset, we began our project by designing a first version (shown below) of our class diagram representing the multidimensional model.

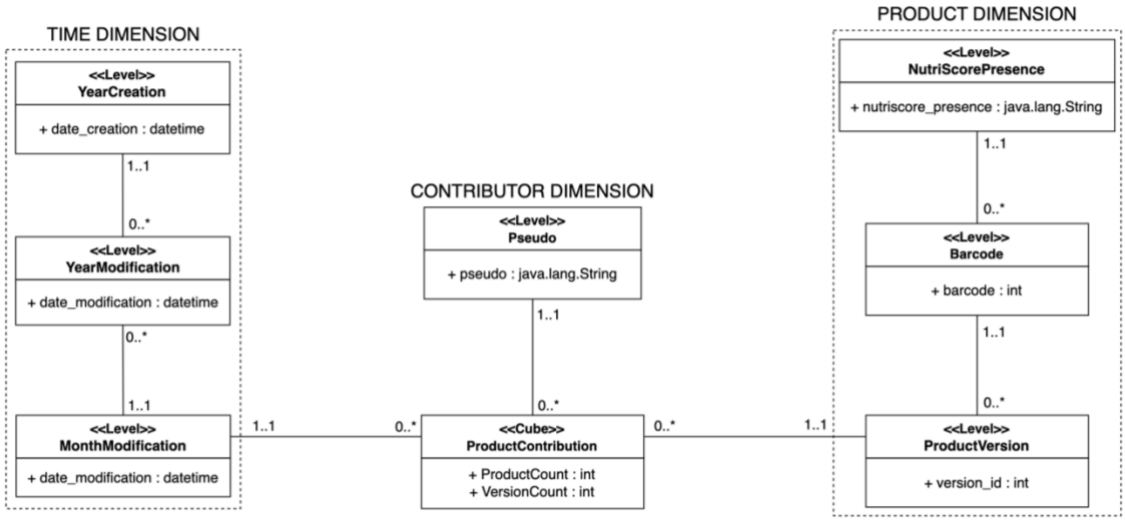

Figure 1: Initial Class Diagram

As it can be seen, we designed a first cube/fact table named "ProductContribution" to perform the 2 mandatory queries. We can appreciate here 2 new variables that did not existed at first in the professor's dataset: "version_id" and "nutriscore_presence". The reason behind the creation of "version_id" was to enable us to handle the different product's data versioning that this project showcases. Consequently, "version_id" is made up by the concatenation of the "barcode" column and the "date_modification_int" column (since it is easier to manipulate than "date_modification"). Additionally, the decision of creating "nutriscore_presence" was made to achieve the desired results on query 2 (the "having vs. not having Nutri-Score" part). As a result, "nutriscore_presence" is a mere boolean which tells if a product has "nutriscore_lettre" (that is, it has any A, B, C, D, or E values) or not.

Finally, the fact table of this first draft was made up by 2 measures, "ProductCount" and "VersionCount" (apart from all the dimensions degraded, as it can be observed), aiming each one to resolve the first mandatory queries. Being precise, "ProductCount" is a mere "distinct-count" of the "barcode" column, while "VersionCount" is a normal "count" of the new variable "version_id".

However, as we continued to develop the project (creating the Mondrian schema, data mart and MDX queries that we will later introduce) we realized that this implementation was far from optimal. Firstly, "barcode" and "version_id" should have different dimensions (Product and Version dimensions respectively), since they are both different measures and by using a hierarchy with both we were restricting ourselves from making the most out of them. Moreover, if we continued with this approach, "barcode" would never appear in the fact table, hence we could not aggregate it for performing its related queries. Nevertheless, thanks to the new implementation (final version), we also consider future cases where we would like to add new information about attributes that will not change regardless of the product's version.

On the other hand, we noticed that having a unique time dimension with both creation and modification times was incorrect, as this would suppose a duplication of the information regarding modification dates. As a result, we decided to split creation and modification times in different dimensions.

2.1.2. Final Design

After all these considerations, and considering that there were more versions between the initial draft and the final one, we ended up with the following class diagram representing our multidimensional model:

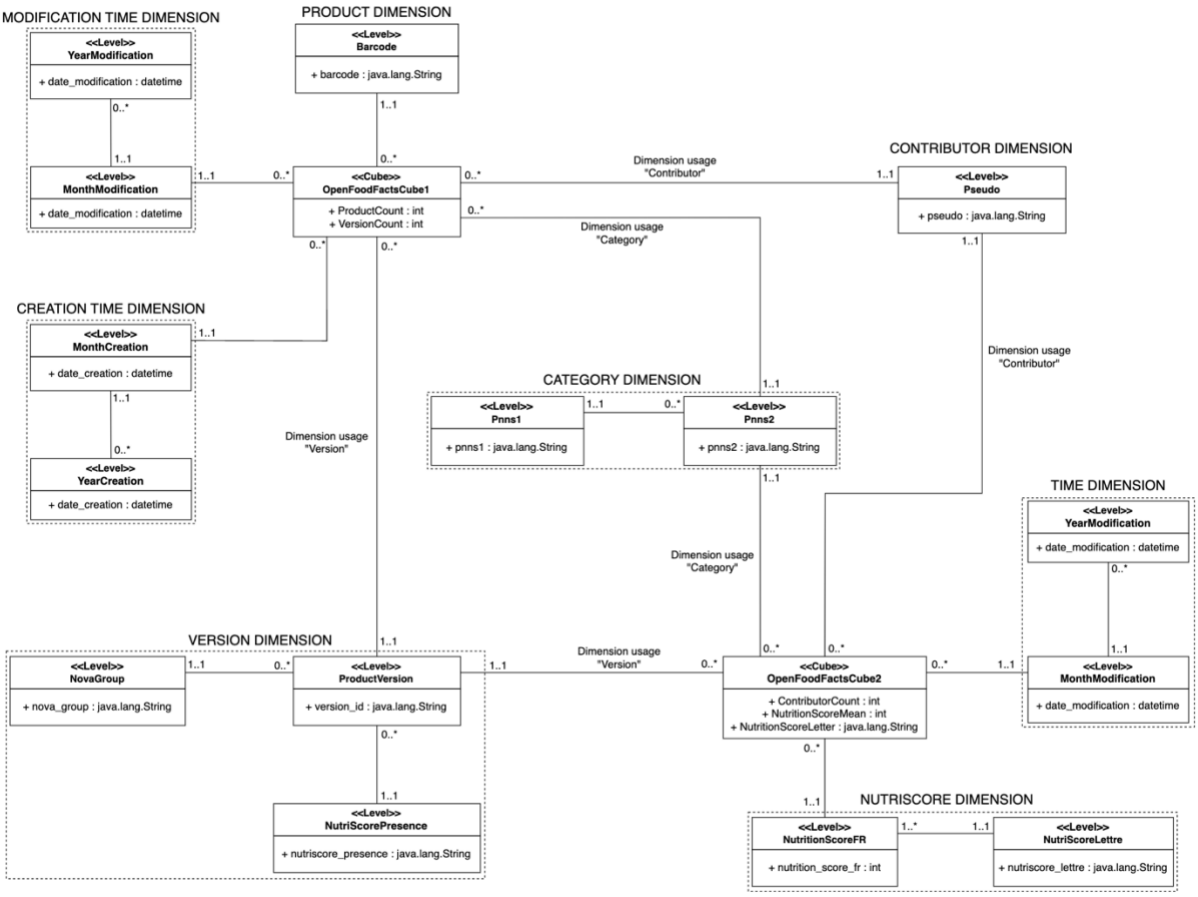

Figure 2: Final Class Diagram

As it can be seen, the previously mentioned modifications have been applied. Moreover, we created another cube/fact table, corresponding to the additional queries that we had to create. In this case, the measures used are “ContributorCount”, which we use to count contributors (depending on different dimensions), “NutritionScoreMean”, that averages the “nutrition_score_fr” column and “NutriScoreLetter”, which is a calculated measure obtain from the previous measure to explain the data in a more visual manner.

Additionally, it can be appreciated that there are duplicated variables (“date_creation” and “date_modification”), however, in the Pentaho Data Integration (Kettle) phase, we handle this data so that at the end, the duplicated columns contain different information (in this case years and months, as the Level names indicate). Also, we decided to not include the “foodgroup1” and “foodgroup2” columns inside the Category dimension, since they have the same information as “pnns1” and “pnns2” but using dashes instead of spaces, therefore we believe it was redundant and unnecessary to include them.

As a result of hours of analyzing, remodeling, and improving our design, we arrived at the above layout, where you can see 3 additional dimensions from the already mentioned ones: Category, Time (related to modification dates) and Nutriscore dimensions. The reasoning behind these decisions was based on an aim of extracting valuable insights, patterns and trends about food products, their categories, their labels (nutriscore and nova group) and how their relate and interact with each other. Furthermore, optimization and integration are fundamental pillars in our project, hence the use of several shared dimensions (like Version, Contributor and Category) for both cubes to avoid redundancy and duplication. Moreover, thanks to this we are also able to create more sophisticated and complex queries, enhancing a more robust data warehouse.

2.2. Data Mart

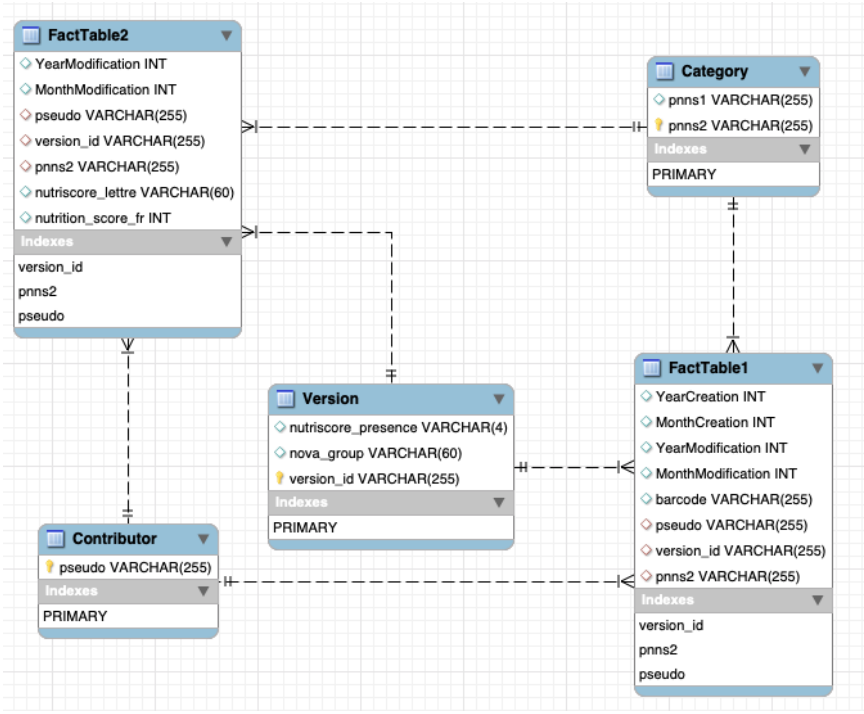

Figure 3: Relational Data Mart model

For the data mart, we made several design choices that would not only simplify the outgoing relational model, but also guarantee scalability and performance at the time of both scaling up the data warehouse (in case we wanted to add information regarding products or contributors, for instance) and perform several queries one after the other. For this reason, we decided to **degrade both Time dimensions in each fact table**. Afterwards, we decided the best way to go regarding the **Product** dimension, was to **also degrade it** in the first fact table, since we were not contemplating including it in the logic of the second cube, improving performance by not having to access its own table, which, for the moment, would not be necessary since we do not have more product linked information, but only linked to different versions.

The second fact table, representing our second cube, also contains degraded dimensions within. These dimensions are **Time** and **Nutriscore**, both only used by this second cube. However, as it can be seen in the data mart diagram and in the UML, we also have dedicated tables for the shared dimensions we decided to implement in our schema. It was imperative that we contained them in their own tables, which would guarantee data integrity and reconciliation. For this reason, we have the **Category**, **Version** and **Contributor** tables, which are referenced by the respective fact tables by their foreign keys: **pseudo**, **version_id** and **pnns2** respectively.

2.3. Kettle Transformations

For this section, we first tried to implement the less kettle as possible, trying to implement most of the transformations in SQL at the table extraction phase. However, after some tinkering and trial and error, we decided the wisest decision would be to implement more kettle nodes that could perform all these transformations and rely less in SQL. The reasoning behind this decision lies in basic readability and legibility. While it could be easier for us to just go with SQL to perform as many operations as possible, it would be substantially more pertinent for third parties to read a workflow in kettle steps, given that this way it appears to be much simpler to understand what the transformations are doing and in which order, and that is why we ended up with the implementations we will be seeing below.

2.3.1. Fact Tables

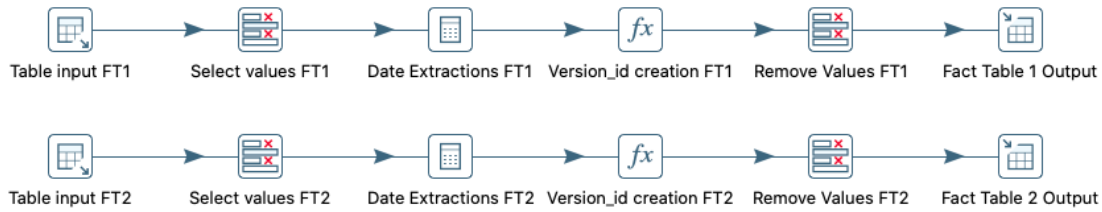

Figure 4: Fact Tables Transformation

Above we can see the Kettle transformations applied to extract transform and populate the data of the different fact tables. First, we start with a simple SQL extraction, to then select the values we are going to need for the different transformations. The first transformations consist simply of extracting the year and the month from both the modification and creation dates, which is done by the calculator node. The second set of transformations consists of refining and creating new fields, like the concatenation of "date_modification_int" and "barcode" to create the new "version_id" identifier, and the preprocessing of the different string attributes such as "**pnns2**" and "**nutriscore_lettre**", introducing "**unknown**" strings for each null or empty value we find, so that the queries in MDX throw legible results, instead of showing null.

2.3.2. Version Table

Figure 5: Version Table Transformation

For the version table, we went with a simpler transformation, given that we only needed to populate a single table with less transformations. We first start with the simple SQL extraction of the data, to then select the fields that will be of interest to us to perform the pertinent transformations, which happen to be the same as in the fact tables, creating the "version_id" field, and preprocessing string attributes like "**nova_group**" to fill null values with the string "**unknown**". However, here we performed one more transformation which was vital for the consecution of the project, which was the creation of the "**nutriscore_presence**" field. We do this by just checking with an if statement whether there is a null or empty value in the "**nutriscore_lettre**" field, in which case we will add the new field with a "**Yes**", showing there is a nutriscore value, and "**No**", showing the opposite.

2.3.3. Category Table

Figure 6: Category Table Transformation

For this transformation, we limited ourselves to just select the data from the category table and just keep the values "**pnns2**" and "**pnns1**" for the data mart.

2.3.4. Contributor Table

Figure 7: Contributor Table Transformation

For the contributor table, we can say the same thing as for the category one, a simple select with no further transformations needed.

2.3.5. Main Job

Figure 8: Workflow of Main Job

Above we can see the "**Main Job**". This job is simply a way of sequencing all our transformations in a single execution step. We decided to go this approach due to the way our relational model was designed, since running the transformations and populating all the tables at the same time would result in a catastrophic failure of the system. This was due to our foreign keys constraints, meaning that we should not be able to populate the fact tables before completely populating the other tables that are referenced, if this was the case, there would be a situation in which we are adding a version_id to the fact table, or a pseudo, or a pnns2 that would not exist in their corresponding tables, something completely forbidden in SQL. In this way, we made sure at every moment that the fact tables would not be populated before the execution of the previous transformations had finished.

2.4. MDX Queries

Apart from the 2 mandatory queries that needed to be achieved, we decided to create (as we previously explained) another cube to enhance our data warehouse. Therefore, we ended up with 4 more queries revolving around the second cube. In this section, we will present and explain the results of each of them.

2.4.1. Query 1

Query 1: Number of products per contributor type (on rows) and creation year (on columns)

Contributor	TimeCreation									
Pseudo	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
kiliweb					4	1.382	1.388	233	15	
openfoodfacts-contributors	27	136	167	274	441	247	182	75	9	
carrefour								554		
sebleouf		40	39	294	133	29	7			
miles67off		165	69	9	1					
elcoco								179		
date-limite-app			2	74	52	27	17	4		
javichu	57	69	10	14	11					
tacite			6	68	29	25	2			
phoenix			21	61	41	2				
agamitsudo		34	11	7	46	13				
jeanbono		23	15	25	8					
jacob80			51	12	5	1				
andre	24	44								
stephane	10	26		6	1					
malikele	25	1		13						
segundo				12	21	5				

Figure 9: Screenshot of Query 1

As we can observe, this query perfectly illustrates the desired output required from the mandatory queries, using only the first cube and the Contributor and TimeCreation dimensions. We can see that there is a higher number of products per contributor in the latest years in comparison with the earliest ones.

2.4.2. Query 2

Query 2: Number of versions (i.e., updates) per year and month (on rows) having vs. not having Nutri-Score (on columns)

TimeModification		By NutriScorePresence	
YearModification	MonthModification	No	Yes
2012	6	1	
	7		1
	8	2	
	9	1	1
	11	2	1
	12		1
2013	2	1	
	3	1	1
	4	1	1
	5		1
	6	3	2
	7		4
	8	3	3
	9	4	7
	10	3	3
	11		2
	12	1	5
2014	1	5	3
	2	4	28
	3	1	6
	4	3	7
	5	1	4
	6	7	4

Figure 10: Screenshot of Query 2

As it can be appreciated, this query perfectly illustrates the desired output required from the mandatory queries, using only the first cube and the TimeModification and Version (using the NutriScorePresence hierarchy) dimensions. Thanks to this query, we are able to distinguish products with different versions as well as if these versions have or do not have nutriscore labels.

2.4.3. Query 3

Query 3: Number of products per category (on rows) and creation year (on columns)

Category		TimeCreation									
Pnns1	Pnns2	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
unknown	unknown	27	55	70	143	188	1.012	821	321	11	1
Sugary snacks	Biscuits and cakes	11	59	44	79	83	119	149	59	3	
	Sweets	7	23	19	55	58	59	59	39	1	
	Chocolate products	8	24	3	20	21	23	33	15	3	
	Pastries	1	7	7	10	16	20	24	8	2	
Fish Meat Eggs	Processed meat	10	38	30	61	75	115	81	26	2	
	Fish and seafood	7	19	21	27	44	66	67	64	2	
	Meat	2	7	9	42	34	55	38	31	3	
	Eggs	2	3	6	3	1	1	4	2		
	Offals	1		2	9	3	4	1	2		
	Fatty fish										
	Lean fish										
	Meat other than poultry										
	Poultry										
Milk and dairy products	Cheese	11	35	30	44	65	91	69	26	3	
	Milk and yogurt	10	25	27	29	28	70	47	77	3	
	Dairy desserts		7	11	26	27	36	20	17		
	Ice cream		4	5	7	4	17	15	14		
Cereals and potatoes	Cereals	13	81	28	52	35	33	58	56		
	Bread	4	25	20	46	33	43	48	24	1	
	Breakfast cereals	1	17	12	36	15	25	19	8	1	
	Potatoes	3	4	10	10	6	21	20	5		
	Legumes	2	5	3	8	7	8	5	7		

Figure 11: Screenshot of Query 3

For this query, we wanted to take advantage of the new shared dimension we had developed thanks to the second cube. Now, we had access to a new Category dimension in our first cube, giving us the possibility to check for instance the number of products existing for each year and each food category, as we can see above. In order to see meaningful results, we sorted the output, like in the first query, to give us first the ones with the highest product count, and we were able to appreciate how the ones with the most contributions are the ones related to sugary snacks, something that should make sense, since these are the most popular products among people, next to first need basic products, that we can see right after. As a remark, we can see how the closer we are to present time, the more contributions are for each category, something that is completely understandable if we see the explosion of internet access and democratization, with each year having more and more mobile devices, more IT education, and a higher interest in statistical models that can benefit from the monetization of information.

2.4.4. Query 4

Query 4: Calculate the mean nutrition score in each modification year (on columns) for each category (on rows)

	Time									
Category	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
Alcoholic beverages	E	E	E	E	E	E	C	E	C	E
Beverages	E	C	C	C	C	C	C	C	C	C
Cereals and potatoes	E	A	C	B	B	C	E	E	E	C
Composite foods	E	A	D	B	C	C	E	C	C	E
Fat and sauces	E	C	D	D	D	D	D	E	E	D
Fish Meat Eggs	D	B	C	E	D	E	E	E	E	D
Fruits and vegetables	E	A	A	A	A	A	A	A	A	A
Milk and dairy products	B	C	C	C	C	E	C	C	C	E
Salty snacks	D	E	D	D	E	D	E	D	E	C
Sugary snacks	E	D	E	D	D	D	D	D	D	D
unknown	E	D	C	C	C	E	C	C	C	E

Figure 12: Screenshot of Query 4

In this query, we wanted to take advantage of the powerful possibilities given by the second cube. Here, we wanted to study not only the mean nutrition score for each food category, but also how this score would evolve through time as new versions were added to the database.

As seen above in our results, we can see how the worst graded categories are the ones related to alcoholic beverages, swiftly followed by sugary snacks and salty snacks. These are completely understanding results, since alcohol, sugars and salt are one of the key metrics that contribute to giving the nutriscore to each product, and it would make sense that those categories of products fall behind so much, while fruits and vegetables are completely healthy, according to this analysis, with a score of "A" every year except 2012. This abnormality could be explained by possible outliers in the data of 2012, that, added to the possibility that there were not too many contributions during that year, might have biased the results.

2.4.5. Query 5

Query 5: Calculate the mean nutrition score for each category (on rows) and nova group (on columns)

	By NovaGroup				
Category	1	2	3	4	unknown
Alcoholic beverages	E	E	E	C	C
Beverages	C	E	C	C	C
Cereals and potatoes	A	E	C	C	B
Composite foods	E	E	B	C	E
Fat and sauces	A	D	C	E	D
Fish Meat Eggs	A	E	C	D	E
Fruits and vegetables	A	E	A	A	A
Milk and dairy products	B	E	E	C	C
Salty snacks	B	D	E	D	D
Sugary snacks	E	D	D	D	D
unknown	B	E	C	C	C

Figure 13: Screenshot of Query 5

For this query, we wanted to see how the NovaGroup and NutriScore levels scale and compare to each other. If we look at the definition of NovaGroup, we can see that, in very simple words, is a labeling system that labels food by the degree of processing they have undergone in their manufacturing process, being 1 a very low preprocessed food, and 4 a heavily preprocessed one. What we wanted, or expected to see here, was a correlation in NutriScore evolution as we changed the NovaGroup level, and that is what we found. For instance, if we look at the Fat and Sauces category, we can see a clear trend in the way its NutriScore grade changes. When we look at Fat and Sauces that fall under the level 1 in NovaGroup, we can see that their average NutriScore is “A”, meaning that the Fat and Sauces products that are less processed are by far the healthiest ones, while, in the other hand, we can see that those products falling under the NovaGroup 4 have a NutriScore grading of “E”, the worst one. This means that those products that are most processed, are more likely to present high contents in sugar and salt, something that unfortunately happens to be true when we go and check in the supermarket by ourselves.

2.4.6. Query 6

Query 6: Calculate the nº of distinct contributors for each nova group (on columns) and category (on rows)

	By NovaGroup				
Category	1	2	3	4	unknown
Alcoholic beverages			10	8	14
Beverages	36	1	13	37	72
Cereals and potatoes	35		36	63	81
Composite foods			23	83	76
Fat and sauces	4	28	22	40	58
Fish Meat Eggs	25		37	53	78
Fruits and vegetables	41		46	30	67
Milk and dairy products	15		51	63	77
Salty snacks	11	1	29	29	35
Sugary snacks	2	15	9	120	111
unknown	32	2	16	67	135

Figure 14: Screenshot of Query 6

Regarding this query, we can see a tendency in the number of distinct contributors aggregating data for nova group with worse labels (especially nova group 4) than healthier ones. This makes total sense since products with this label (alcohol, sugary products, pastries…) have much more entries than other healthier foods (as observed with the results of query 3).

2.4.7. Query 7

Query 7: Calculate the nº of distinct contributors for each nova group (on rows) and nutriscore (on columns)

	Nutriscore					
By NovaGroup	a	b	c	d	e	unknown
1	53	34	29	5	1	46
2			14	7	15	17
3	52	33	43	59	19	35
4	75	76	106	117	80	78
unknown	126	113	123	125	91	158

Figure 15: Screenshot of Query 7

For this query, we can see and confirm similar results as in the query 6, but from a different perspective. This means that here we can also confirm that those categories, or those products with the lowest rankings in both NovaGroup and NutriScore, are the ones that contribute the most to the database (or put in other words, contributors aggregate more products of these categories), again, probably because these are the most popular ones amongst society.