UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA UNIVERSIDAD DE MÁLAGA

Graduados en Ingeniería de la Salud

Biological Databases

Nutritional Database

Made by
Susana Rocío Fernández Giaccomassi
Pablo Moreno García-Espina

Supervised by
Ismael Navas Delgado

Department
Lenguajes y Ciencias de la Computación

MÁLAGA, June 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DEL SOFTWARE

Bases de Datos Biológicas

Nutritional Database

Made by
Susana Rocío Fernández Giaccomassi,
Pablo Moreno García-Espina

Supervised by
Ismael Navas Delgado

Department
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNE 2023

Contents

1 Introduction

1.1 Motivation

As a task for the Biological Database subject we currently are studying, we were told to develop a Biological Database during the whole semester, applying the knowledge seen in class to enhance the quality of our database.

At first we thought about creating a database with medicines in order to imitate a pharmacy storage system. However, we thought that maybe this idea was already quite ”overused”. Then, we came up with the idea of developing a database related with food.

Nowadays having healthy eating habits and a healthy lifestyle is becoming more demanding by a great part of the population. Over a 60 percent of the people surveyed in 2020 about the eating habits admitted that they were rather concerned about eating in a healthy way. We found an opportunity in this fact, and we thought it was a perfect idea to continue to develop this kind of database.

1.2 Objectives

We have created this database mainly in order to make our patient's lifestyle easier. Depending on the objective of our patients, their current state of health and other aspects such as age, sex, etc, we will assign them a diet so as to reach their goals.

Therefore, we will need a group of nutritionists. They are the ones who will be in charge of the development of the diets. We will storage data for aliments mainly, and we will use this data in order to create new diets for our patients in the nutritional center.

1.3 Structure of the document

As we wanted to create this database in different languages, we will divide this document in three distinguishable parts: MySQL, MongoDB and XML. In every section of the document we will discuss about the structure of the database and we will illustrate its use with some queries.

1.4 Technologies used

We have mainly used MySQLWorkBench, MongoDB Compass and ExistsDB for MySQL, MongoDB and XML database respectively. These are the programs we have used in order to write the queries and to see how our database works.

We will use a relational database, a type of database that stores and provides access to data points which are related to one another.

In order to generate data for our database, we will use CSV files.

We have also created a repository in GitHub in order to organise all of the files we have been created during the process of developing the database (including this report)

2
MySQLWorkbench

2.1 RDBMS Deployment

In the schema of our database, we will have the following tables: Food, Patient, Nutritionist, Diet and List Of Food. One Diet will consist of different kinds of Food, a Diet will be created by a Nutritionist (although a Nutrionist can create more than one) and a Nutritionist will take care of one or more Patients. The table List Of Food is created as a consequence of the relation between Diet and Food (a relation M:N creates a new table including the primary keys of both tables included in the relation).

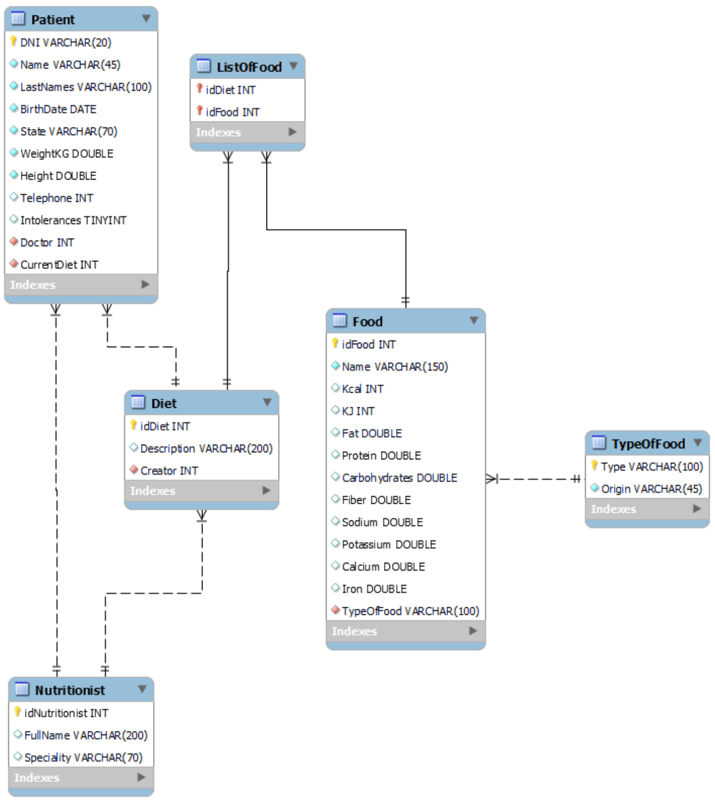


Figure 1: EER model

Firstly, we have introduced the table Food, which has the food ID as the primary key. The rest of columns of the table are related with information such as the energy in kcal and KJ that provides the food, the quantity of Protein and Fiber in food in every 100 grams... Then the table to register the information about the Patient has been created and fulfilled with relevant information about it in order to organize the new diet. It also stores the current diet that the person is following and the intolerances, information that can be quite useful in order to design a new diet.

The table of the Nutritionist is created also with an ID as the primary key. It also contains information about the name of the nutritionist and the specialty that he or she has (obesity, for example)

For the diets we have another table. As an identifier, we have and ID for the diet (primary key) Also, a description and a Creator are columns for this table. The Description will contain the different steps to follow in order to do the diet. The data which has been introduced in the table Food has been obtained by a csv file in https://www.fao.org/infoods/infoods/tablas-y-bases-de-datos/es/

2.2 Query Design

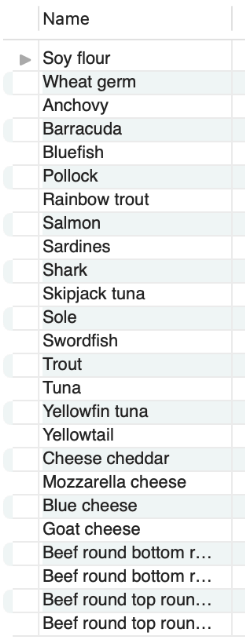
· Food with more than 20 grams of protein (Fig. 2):

```
SELECT NutritionalDB.Food.Name
FROM NutritionalDB.Food WHERE Food.Protein > 20;
```

· All the names of patients treated by the doctor Pablo Moreno Garcia-Espina (Fig. 3):

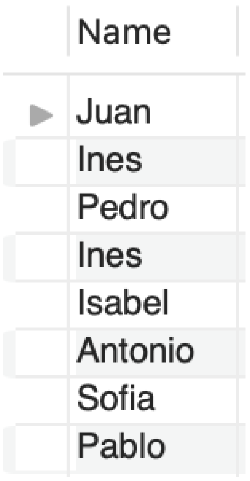
```
SELECT NutritionalDB.Patient.Name
FROM NutritionalDB.Patient JOIN NutritionalDB.Nutritionist
ON Patient.Doctor = Nutritionist.idNutritionist
WHERE Nutritionist.FullName = 'Pablo Moreno Garcia-Espina';
```

· Description of the diets created by the doctor Susana Rocio Fernandez Giaccomassi (Fig. 4)



Name
Soy flour
Wheat germ
Anchovy
Barracuda
Bluefish
Pollock
Rainbow trout
Salmon
Sardines
Shark
Skipjack tuna
Sole
Swordfish
Trout
Tuna
Yellowfin tuna
Yellowtail
Cheese cheddar
Mozzarella cheese
Blue cheese
Goat cheese
Beef round bottom r…
Beef round bottom r…
Beef round top roun…
Beef round top roun…

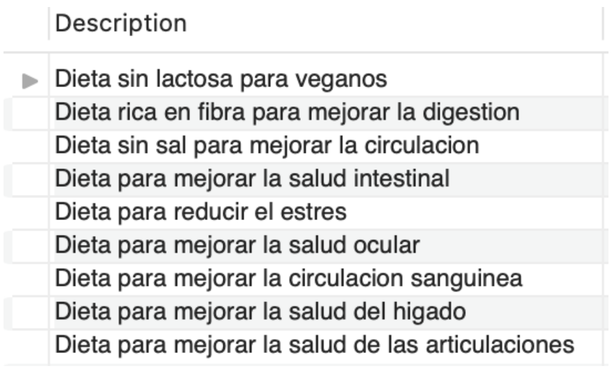
Figure 2: First query output



Name
Juan
Ines
Pedro
Ines
Isabel
Antonio
Sofia
Pablo

Figure 3: Second query output

```
SELECT NutritionalDB.Diet.Description
FROM NutritionalDB.Diet JOIN NutritionalDB.Nutritionist
ON Diet.Creator = Nutritionist.idNutritionist
WHERE Nutritionist.FullName = 'Susana Rocio Fernandez
   Giaccomassi';
```



Description
Dieta sin lactosa para veganos
Dieta rica en fibra para mejorar la digestion
Dieta sin sal para mejorar la circulacion
Dieta para mejorar la salud intestinal
Dieta para reducir el estres
Dieta para mejorar la salud ocular
Dieta para mejorar la circulacion sanguinea
Dieta para mejorar la salud del higado
Dieta para mejorar la salud de las articulaciones

Figure 4: Third query output

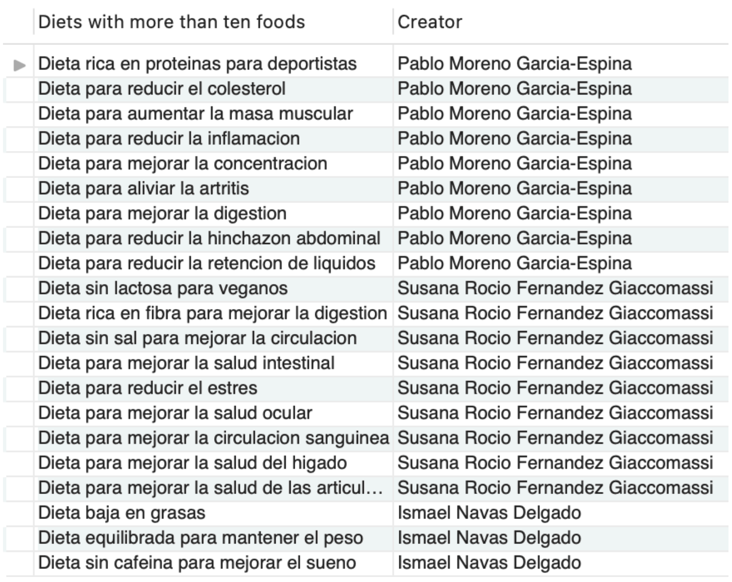
· Obtain the description and the name of their creator of all diets with more than 10 foods (Fig. 5):

```
SELECT NutritionalDB.Diet.Description as 'Diets with more than
   ten foods', NutritionalDB.Nutritionist.FullName as 'Creator'
FROM NutritionalDB.Diet JOIN NutritionalDB.Nutritionist
ON Diet.Creator = Nutritionist.idNutritionist
WHERE NutritionalDB.Diet.idDiet IN (
SELECT NutritionalDB.Diet.idDiet
FROM NutritionalDB.ListOfFood
GROUP BY idDiet HAVING COUNT(*) > 10
);
```

· Get the names and protein content of foods that have more protein than the average protein of all foods (Fig. 6):

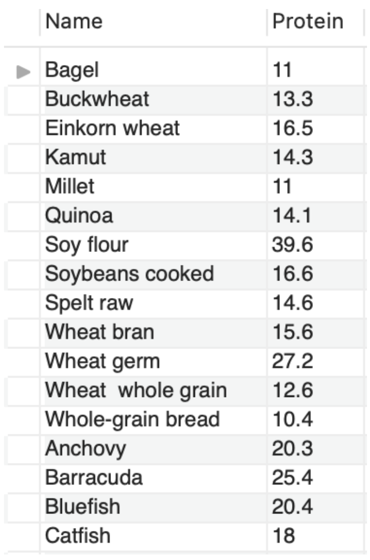
```
SELECT NutritionalDB.Food.Name, NutritionalDB.Food.Protein
FROM NutritionalDB.Food
```

```
WHERE NutritionalDB.Food.Protein > (
SELECT AVG(NutritionalDB.Food.Protein)
FROM NutritionalDB.Food
);
```



Diets with more than ten foods	Creator
Dieta rica en proteinas para deportistas	Pablo Moreno Garcia-Espina
Dieta para reducir el colesterol	Pablo Moreno Garcia-Espina
Dieta para aumentar la masa muscular	Pablo Moreno Garcia-Espina
Dieta para reducir la inflamacion	Pablo Moreno Garcia-Espina
Dieta para mejorar la concentracion	Pablo Moreno Garcia-Espina
Dieta para aliviar la artritis	Pablo Moreno Garcia-Espina
Dieta para mejorar la digestion	Pablo Moreno Garcia-Espina
Dieta para reducir la hinchazon abdominal	Pablo Moreno Garcia-Espina
Dieta para reducir la retencion de liquidos	Pablo Moreno Garcia-Espina
Dieta sin lactosa para veganos	Susana Rocio Fernandez Giaccomassi
Dieta rica en fibra para mejorar la digestion	Susana Rocio Fernandez Giaccomassi
Dieta sin sal para mejorar la circulacion	Susana Rocio Fernandez Giaccomassi
Dieta para mejorar la salud intestinal	Susana Rocio Fernandez Giaccomassi
Dieta para reducir el estres	Susana Rocio Fernandez Giaccomassi
Dieta para mejorar la salud ocular	Susana Rocio Fernandez Giaccomassi
Dieta para mejorar la circulacion sanguinea	Susana Rocio Fernandez Giaccomassi
Dieta para mejorar la salud del higado	Susana Rocio Fernandez Giaccomassi
Dieta para mejorar la salud de las articul...	Susana Rocio Fernandez Giaccomassi
Dieta baja en grasas	Ismael Navas Delgado
Dieta equilibrada para mantener el peso	Ismael Navas Delgado
Dieta sin cafeina para mejorar el sueno	Ismael Navas Delgado

Figure 5: Fourth query ouput



Name	Protein
Bagel	11
Buckwheat	13.3
Einkorn wheat	16.5
Kamut	14.3
Millet	11
Quinoa	14.1
Soy flour	39.6
Soybeans cooked	16.6
Spelt raw	14.6
Wheat bran	15.6
Wheat germ	27.2
Wheat whole grain	12.6
Whole-grain bread	10.4
Anchovy	20.3
Barracuda	25.4
Bluefish	20.4
Catfish	18

Figure 6: Fifth query output

2.3 RDBMS Optimization

Firstly, we have changed the first query in order to make a deeper search. We can also search in the database according to the type of food, as we made a join between the table of Food and the table of Type of Food.

In order to make the search easier when it comes to diets related with sporty people, we have decided to design two indexes: one with Name and Kcal and another with Protein and Name. This index becomes useful with our first query, as we are looking firstly at the food with protein's value of more than 20 grams.

For the previous index, when we try to execute the first query apparently the index is not used. That could be because the improvement when the index is used is not considerably high when you compare it with the performance without the index.

This picture (Fig. 7) shows the performance of the first query without the index.

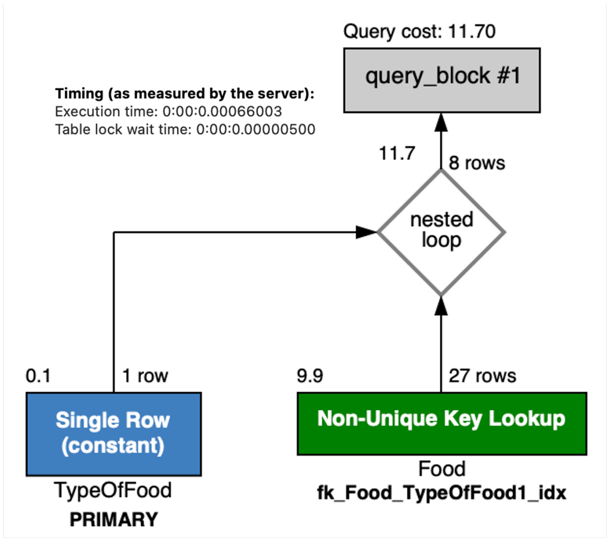
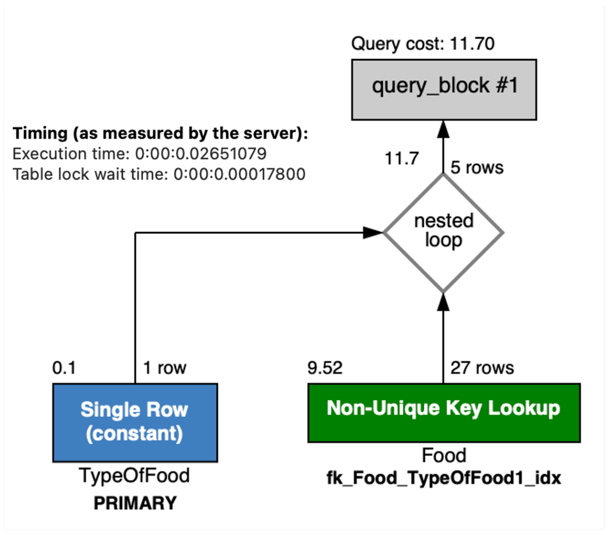


Figure 7: First query without index

This picture (Fig. 8) shows the performance of the first query with the index.

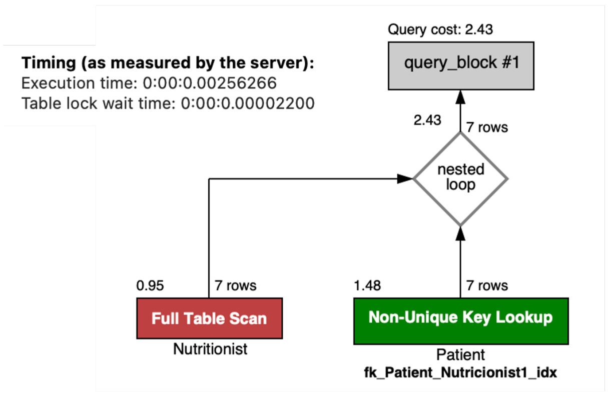
Furthermore, as we will frequently look for the different diets that one nutritionist has created, we will create an index for the column "Full Name" in the table Nutritionist. We already have an index for the id of a nutritionist, but we will typically get a diet of a nutritionist by his concrete name. With this index we will get an advantage in the second and third query, as they use the full name of the nutritionist in order to make the query.



Timing (as measured by the server):
Execution time: 0:00:0.02651079
Table lock wait time: 0:00:0.00017800

Figure 8: First query with index

This picture (Fig. 9) shows the time that lasts the second query without the index.



Timing (as measured by the server):
Execution time: 0:00:0.00256266
Table lock wait time: 0:00:0.00002200

Figure 9: Second query without index

This picture (Fig. 10) shows the time that lasts the second query with the index.

This picture (Fig. 11) shows the performance of the first query without the index.

This picture (Fig. 12) shows the performance of the third query with the index.

We see how the time elapsed in the second time is lower than in the first time. The existence of the index makes the performance of the query higher.

We will also do queries which involve the name of the different patients we have in our database. Therefore, it would be suitable too to make an index with the name of the patients. Every time that we do a query in which the name of the patient is involved, we will get an advantage in terms of time, as we could see before.

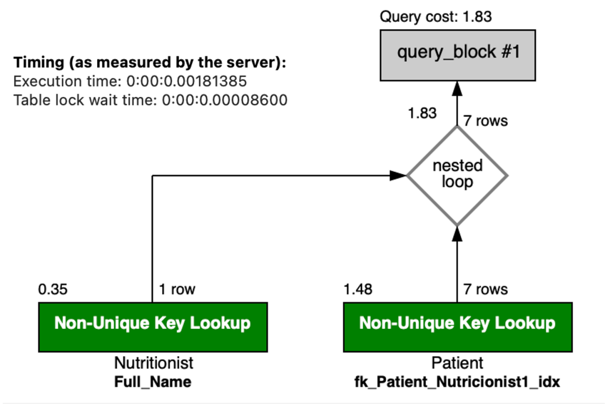


Figure 10: Second query with index

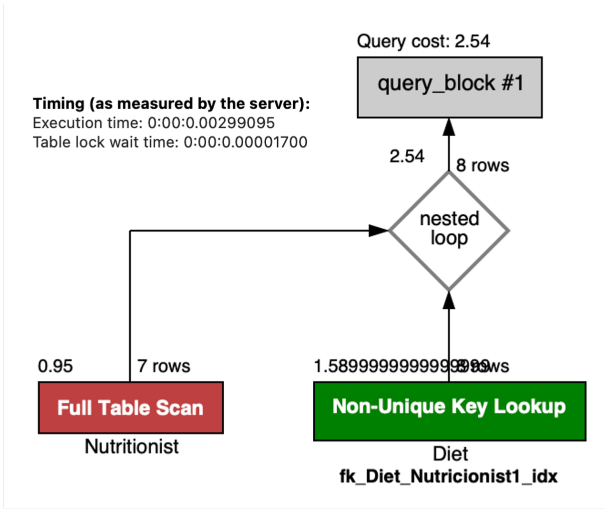


Figure 11: Third query without index

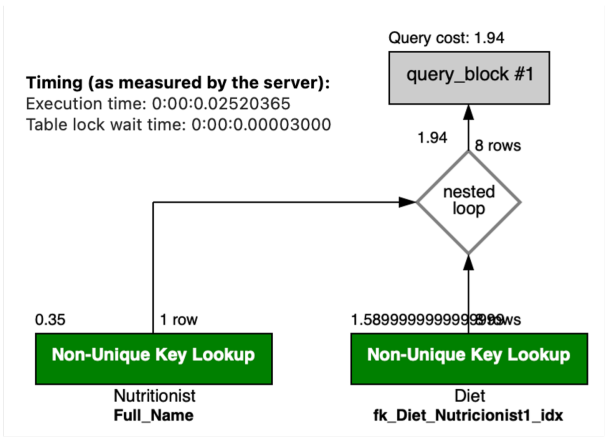


Figure 12: Third query with index

We think that nutritionists may want to filter the aliments by the nutritional information and origin, so the query would be:

```
SELECT f.Name
FROM Food f
JOIN TypeOfFood t ON f.TypeOfFood = t.Type
WHERE t.Origin = 'animal' AND (f.Protein > 15 OR f.Iron > 5);
```

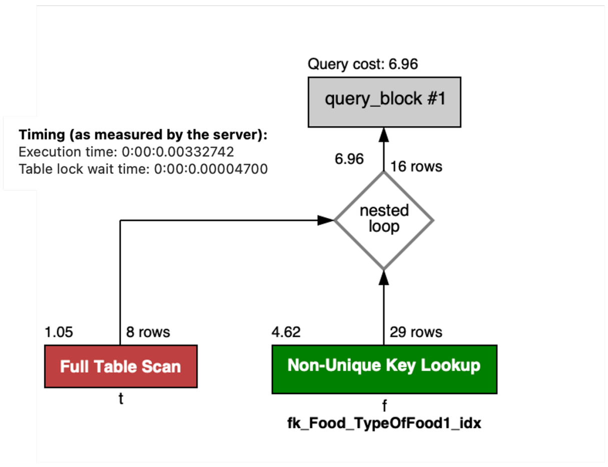


Figure 13: Extra query with index

In this case, we have created an index for the table Food including the columns Protein and Iron, but it did not used it, neither the protein index created before. We also tried with two indexes, one for the column Protein and the other one for Iron, but it goes through the entire table. The execution plan is on Fig.13.

2.4 Python

In order to develop the Python application to test the queries from an user interface, we have created a new project in PyCharm named NutritionalDB.

Specifically in the class NutritionalDB.py we have created three definitions: one to initiate the connection to the database in SQL, one close the connection and another one to print the result of the queries we will implement next.

```python
import pymysql
import pymysql.cursors
```

```
from pymysql import DatabaseError
class NutritionalQuerys(object):

    def __init__(self, host, user, password, database):
        self.connection = pymysql.connect(host=host,
                                          user=user,
                                          password=password,
                                          db=database,
                                          charset='utf8mb4',
                                          cursorclass=pymysql.cursors
    .DictCursor)
        print("Successfully Connected!")

    def close(self):
        self.connection.close()
        print("Connection Closed!")

    def print_query(self, query):
        cursor = self.exec_query(query)
        print()
        for row in cursor:
            print(row)
```

However, if we analyze the code for the "print query" definition we see that another definition called "exec query" is called. This new definition is used in order to send the query to the cursor created with the connection.

```
        def exec_query(self, query):
        try:
            with self.connection.cursor() as cursor:
                cursor.execute(query)
                return cursor
        except DatabaseError as e:
            if e.args[1] == '#42000Unknown Database':
                print(e)
            else:
                print(e)
```

After these first definitions, we have the definitions for each of the queries. When we made the queries in SQL we asked for concrete values of the columns (for example 20 grams of protein), however here we have parameterized the functions in order to make the query according to the user's goal.

When the user in Python calls the definition in order to ask for a food whose protein content is greater than X, he has to specify this X value in the call to the method.

```
        def gtProtein(self, protein):
        sql = 'SELECT NutritionalDB.Food.Name FROM Food WHERE Food.
    Protein > ' + str(protein) + ';'
        print()
        print('Food with more than ' + str(protein) + ' grams of
    protein:')
        self.print_query(sql)
```

As we can see, all of the definitions we have implemented in Python have the same structure. Firstly we create a string called sql in which we write the proper query. We have to insert here the different parameters we have in the header of the definition. Secondly, we print an empty line in order to make a separation in the terminal between queries. Then we print a text in order to make the user understand about the meaning of the query, and finally we execute the method "print query" in order to show in the terminal the results of the query.

We have created a method that receives a doctor as input and returns the patients treated by the input doctor.

```
        def treatedBy(self, doctor):
        sql = "SELECT NutritionalDB.Patient.Name FROM NutritionalDB.
    Patient JOIN Nutritionist " \
              "ON Patient.Doctor = Nutritionist.idNutritionist " \
              "WHERE Nutritionist.FullName = '" + doctor + "';"
        print()
        print('Patients treated by ' + str(doctor) + ':')
        self.print_query(sql)
```

The following definition receives an integer value and returns the diets with more than that number of food.

```

    def dietsWithMoreFood(self, quantity):
        sql = "SELECT NutritionalDB.Diet.Description as " \
              "'Diets with more than " + str(quantity) + " foods', " \
                "NutritionalDB.Nutritionist.FullName as 'Creator' FROM " \
                "NutritionalDB.Diet JOIN NutritionalDB.Nutritionist ON " \
                "Diet.Creator = Nutritionist.idNutritionist " \
                "WHERE NutritionalDB.Diet.idDiet IN (SELECT " \
                "NutritionalDB.Diet.idDiet FROM NutritionalDB.ListOfFood " \
                "GROUP BY idDiet HAVING COUNT(*) > " \
                 "" + str(quantity) + ");"
        print()
        print('Diets with more than ' + str(quantity) + ' foods:')
        self.print_query(sql)
```

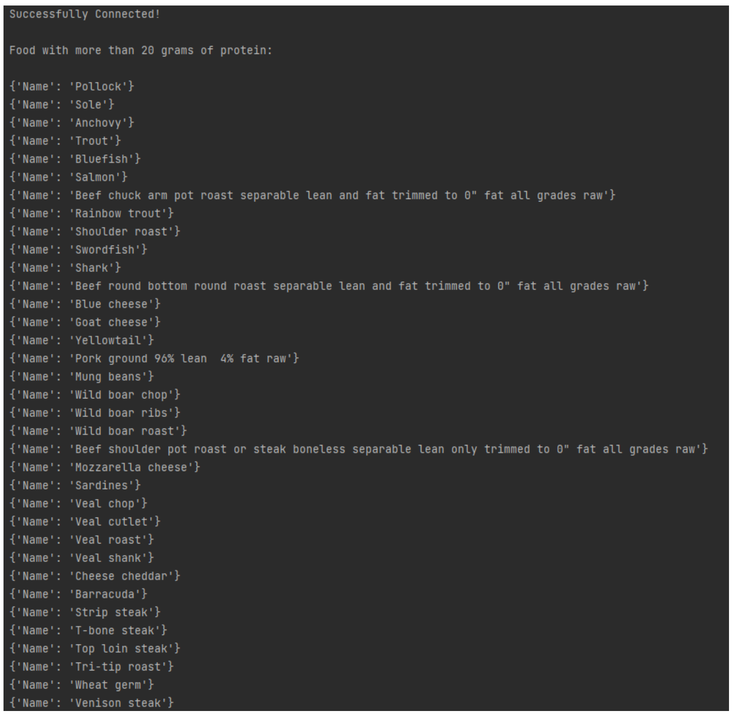
We can also get the food with more than the average of an input column. And the last definition receives two columns and two values, so we can filter by columns and grams.

```

    def moreThanAvg(self, column):
        sql = "SELECT NutritionalDB.Food.Name, NutritionalDB.Food." + column + \
              " FROM NutritionalDB.Food" \
              " WHERE NutritionalDB.Food." + column + " > (SELECT AVG(NutritionalDB.Food." \
              + column + ") FROM NutritionalDB.Food);"
        print()
        print('Food with more ' + str(column) + ' than the average:')
        self.print_query(sql)

    def filterByInfoAndOrigin(self, info1, info2, value1, value2,
  origin):
        sql = "SELECT NutritionalDB.Food.Name FROM NutritionalDB.Food
  " \
              " JOIN NutritionalDB.TypeOfFood ON Food.TypeOfFood =
  TypeOfFood.Type" \
              " WHERE TypeOfFood.Origin = '" + origin + \
              "' AND (Food." + info1 + " > " + str(value1) + " OR
  Food." + info2 + \
              " > " + str(value2) + ");"
        print()
        print('Food with more than ' + str(value1) + ' grams of ' +
  str(info1) + ' or more than '
              + str(value2) + ' grams of ' + str(info2) + ' from ' +
  str(origin) + ' origin :')
        self.print_query(sql)
```

Lastly, we have created a main class to test these definitions:

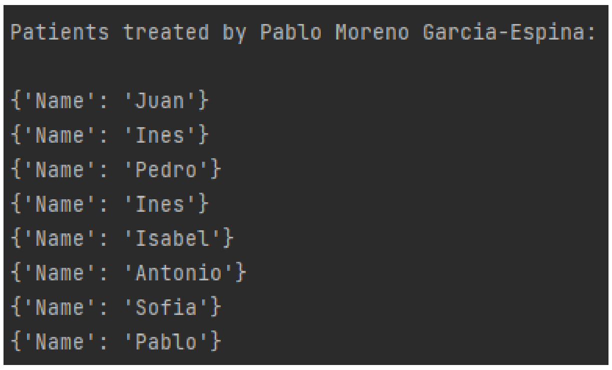
```
if __name__ == '__main__':
    password = 'Your_Password'
    ndb = NutritionalQuerys('localhost', 'root', password, '
  NutritionalDB')
    ndb.gtProtein(20)
    ndb.treatedBy('Pablo Moreno Garcia-Espina')
    ndb.dietsWithMoreFood(10)
    ndb.moreThanAvg('Protein')
    ndb.filterByInfoAndOrigin('Protein', 'Iron', 15, 5, 'animal')
```

A preview of the outputs can be seen below in figures 14-18:



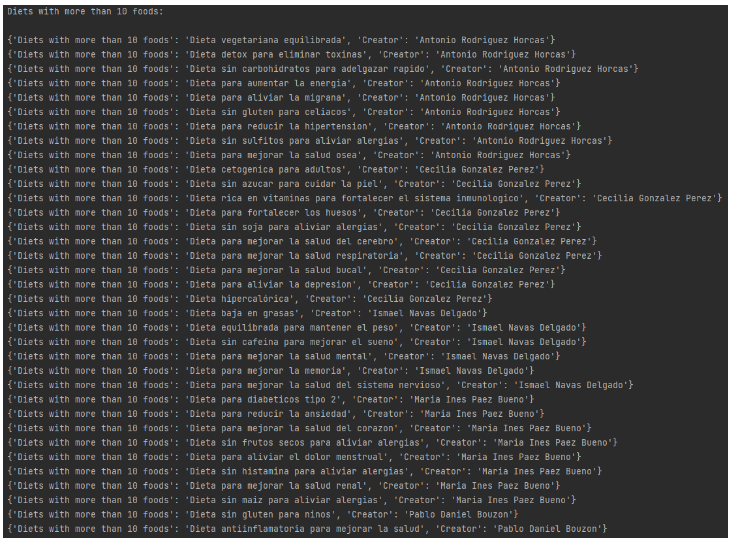
```
Successfully Connected!

Food with more than 20 grams of protein:

{'Name': 'Pollock'}
{'Name': 'Sole'}
{'Name': 'Anchovy'}
{'Name': 'Trout'}
{'Name': 'Bluefish'}
{'Name': 'Salmon'}
{'Name': 'Beef chuck arm pot roast separable lean and fat trimmed to 0" fat all grades raw'}
{'Name': 'Rainbow trout'}
{'Name': 'Shoulder roast'}
{'Name': 'Swordfish'}
{'Name': 'Shark'}
{'Name': 'Beef round bottom round roast separable lean and fat trimmed to 0" fat all grades raw'}
{'Name': 'Blue cheese'}
{'Name': 'Goat cheese'}
{'Name': 'Yellowtail'}
{'Name': 'Pork ground 96% lean  4% fat raw'}
{'Name': 'Mung beans'}
{'Name': 'Wild boar chop'}
{'Name': 'Wild boar ribs'}
{'Name': 'Wild boar roast'}
{'Name': 'Beef shoulder pot roast or steak boneless separable lean only trimmed to 0" fat all grades raw'}
{'Name': 'Mozzarella cheese'}
{'Name': 'Sardines'}
{'Name': 'Veal chop'}
{'Name': 'Veal cutlet'}
{'Name': 'Veal roast'}
{'Name': 'Veal shank'}
{'Name': 'Cheese cheddar'}
{'Name': 'Barracuda'}
{'Name': 'Strip steak'}
{'Name': 'T-bone steak'}
{'Name': 'Top loin steak'}
{'Name': 'Tri-tip roast'}
{'Name': 'Wheat germ'}
{'Name': 'Venison steak'}
```

Figure 14: gtProtein() output



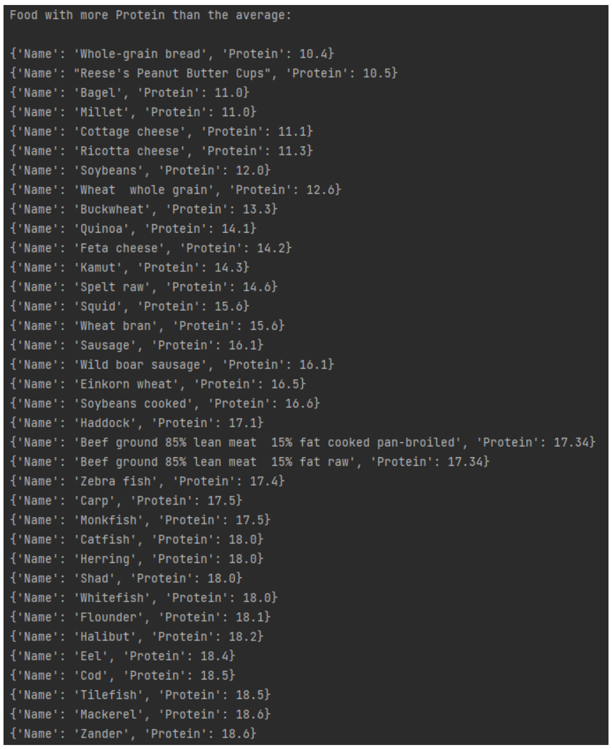
```
Patients treated by Pablo Moreno Garcia-Espina:

{'Name': 'Juan'}
{'Name': 'Ines'}
{'Name': 'Pedro'}
{'Name': 'Ines'}
{'Name': 'Isabel'}
{'Name': 'Antonio'}
{'Name': 'Sofia'}
{'Name': 'Pablo'}
```

Figure 15: treatedBy() output



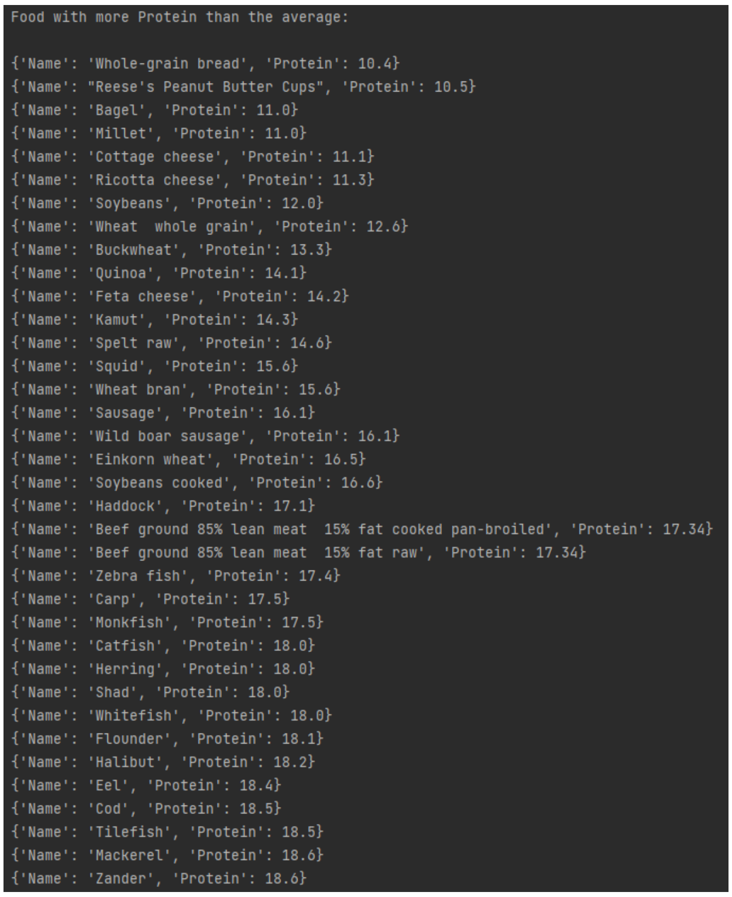
```
Diets with more than 10 foods:

{'Diets with more than 10 foods': 'Dieta vegetariana equilibrada', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta detox para eliminar toxinas', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta sin carbohidratos para adelgazar rapido', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta para aumentar la energia', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta para aliviar la migrana', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta sin gluten para celiacos', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta para reducir la hipertension', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta sin sulfitos para aliviar alergias', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud osea', 'Creator': 'Antonio Rodriguez Horcas'}
{'Diets with more than 10 foods': 'Dieta cetogenica para adultos', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta sin azucar para cuidar la piel', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta rica en vitaminas para fortalecer el sistema inmunologico', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta para fortalecer los huesos', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta sin soja para aliviar alergias', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud del cerebro', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud respiratoria', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud bucal', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta para aliviar la depresion', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta hipercalórica', 'Creator': 'Cecilia Gonzalez Perez'}
{'Diets with more than 10 foods': 'Dieta baja en grasas', 'Creator': 'Ismael Navas Delgado'}
{'Diets with more than 10 foods': 'Dieta equilibrada para mantener el peso', 'Creator': 'Ismael Navas Delgado'}
{'Diets with more than 10 foods': 'Dieta sin cafeina para mejorar el sueno', 'Creator': 'Ismael Navas Delgado'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud mental', 'Creator': 'Ismael Navas Delgado'}
{'Diets with more than 10 foods': 'Dieta para mejorar la memoria', 'Creator': 'Ismael Navas Delgado'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud del sistema nervioso', 'Creator': 'Ismael Navas Delgado'}
{'Diets with more than 10 foods': 'Dieta para diabeticos tipo 2', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta para reducir la ansiedad', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud del corazon', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta sin frutos secos para aliviar alergias', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta para aliviar el dolor menstrual', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta sin histamina para aliviar alergias', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta para mejorar la salud renal', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta sin maiz para aliviar alergias', 'Creator': 'Maria Ines Paez Bueno'}
{'Diets with more than 10 foods': 'Dieta sin gluten para ninos', 'Creator': 'Pablo Daniel Bouzon'}
{'Diets with more than 10 foods': 'Dieta antiinflamatoria para mejorar la salud', 'Creator': 'Pablo Daniel Bouzon'}
```

Figure 16: dietsWithMoreFood() output



```
Food with more Protein than the average:

{'Name': 'Whole-grain bread', 'Protein': 10.4}
{'Name': "Reese's Peanut Butter Cups", 'Protein': 10.5}
{'Name': 'Bagel', 'Protein': 11.0}
{'Name': 'Millet', 'Protein': 11.0}
{'Name': 'Cottage cheese', 'Protein': 11.1}
{'Name': 'Ricotta cheese', 'Protein': 11.3}
{'Name': 'Soybeans', 'Protein': 12.0}
{'Name': 'Wheat  whole grain', 'Protein': 12.6}
{'Name': 'Buckwheat', 'Protein': 13.3}
{'Name': 'Quinoa', 'Protein': 14.1}
{'Name': 'Feta cheese', 'Protein': 14.2}
{'Name': 'Kamut', 'Protein': 14.3}
{'Name': 'Spelt raw', 'Protein': 14.6}
{'Name': 'Squid', 'Protein': 15.6}
{'Name': 'Wheat bran', 'Protein': 15.6}
{'Name': 'Sausage', 'Protein': 16.1}
{'Name': 'Wild boar sausage', 'Protein': 16.1}
{'Name': 'Einkorn wheat', 'Protein': 16.5}
{'Name': 'Soybeans cooked', 'Protein': 16.6}
{'Name': 'Haddock', 'Protein': 17.1}
{'Name': 'Beef ground 85% lean meat  15% fat cooked pan-broiled', 'Protein': 17.34}
{'Name': 'Beef ground 85% lean meat  15% fat raw', 'Protein': 17.34}
{'Name': 'Zebra fish', 'Protein': 17.4}
{'Name': 'Carp', 'Protein': 17.5}
{'Name': 'Monkfish', 'Protein': 17.5}
{'Name': 'Catfish', 'Protein': 18.0}
{'Name': 'Herring', 'Protein': 18.0}
{'Name': 'Shad', 'Protein': 18.0}
{'Name': 'Whitefish', 'Protein': 18.0}
{'Name': 'Flounder', 'Protein': 18.1}
{'Name': 'Halibut', 'Protein': 18.2}
{'Name': 'Eel', 'Protein': 18.4}
{'Name': 'Cod', 'Protein': 18.5}
{'Name': 'Tilefish', 'Protein': 18.5}
{'Name': 'Mackerel', 'Protein': 18.6}
{'Name': 'Zander', 'Protein': 18.6}
```

Figure 17: moreThanAvg() output



```
Food with more Protein than the average:

{'Name': 'Whole-grain bread', 'Protein': 10.4}
{'Name': "Reese's Peanut Butter Cups", 'Protein': 10.5}
{'Name': 'Bagel', 'Protein': 11.0}
{'Name': 'Millet', 'Protein': 11.0}
{'Name': 'Cottage cheese', 'Protein': 11.1}
{'Name': 'Ricotta cheese', 'Protein': 11.3}
{'Name': 'Soybeans', 'Protein': 12.0}
{'Name': 'Wheat  whole grain', 'Protein': 12.6}
{'Name': 'Buckwheat', 'Protein': 13.3}
{'Name': 'Quinoa', 'Protein': 14.1}
{'Name': 'Feta cheese', 'Protein': 14.2}
{'Name': 'Kamut', 'Protein': 14.3}
{'Name': 'Spelt raw', 'Protein': 14.6}
{'Name': 'Squid', 'Protein': 15.6}
{'Name': 'Wheat bran', 'Protein': 15.6}
{'Name': 'Sausage', 'Protein': 16.1}
{'Name': 'Wild boar sausage', 'Protein': 16.1}
{'Name': 'Einkorn wheat', 'Protein': 16.5}
{'Name': 'Soybeans cooked', 'Protein': 16.6}
{'Name': 'Haddock', 'Protein': 17.1}
{'Name': 'Beef ground 85% lean meat  15% fat cooked pan-broiled', 'Protein': 17.34}
{'Name': 'Beef ground 85% lean meat  15% fat raw', 'Protein': 17.34}
{'Name': 'Zebra fish', 'Protein': 17.4}
{'Name': 'Carp', 'Protein': 17.5}
{'Name': 'Monkfish', 'Protein': 17.5}
{'Name': 'Catfish', 'Protein': 18.0}
{'Name': 'Herring', 'Protein': 18.0}
{'Name': 'Shad', 'Protein': 18.0}
{'Name': 'Whitefish', 'Protein': 18.0}
{'Name': 'Flounder', 'Protein': 18.1}
{'Name': 'Halibut', 'Protein': 18.2}
{'Name': 'Eel', 'Protein': 18.4}
{'Name': 'Cod', 'Protein': 18.5}
{'Name': 'Tilefish', 'Protein': 18.5}
{'Name': 'Mackerel', 'Protein': 18.6}
{'Name': 'Zander', 'Protein': 18.6}
```

Figure 18: filterByInfoAndOrigin() output

3 MongoDB

3.1 NoSQL Database Design

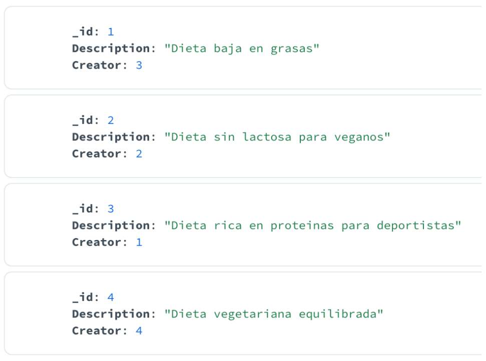
In this section we are going to create an alternative to our SQL database. This is going to be done in MongoDB, using JSON documents to create the collections. The structure that our NoSQL database is going to have is simple, having one collection for each table on the relational SQL database.

To export our tables in the JSON format, we used a powerful tool of MySQL Workbench. We selected each table and chose the option ”export wizard table”. Then, we selected the csv format, and changed it for JSON file when saving it into the repository folder.

In order to create the different collections in MongoDB, we opened a new connection in MongoDB Compass and started to create new collections in a folder. We created the collections with the same name as the tables in SQL and import the data from the JSON files we have just generated.

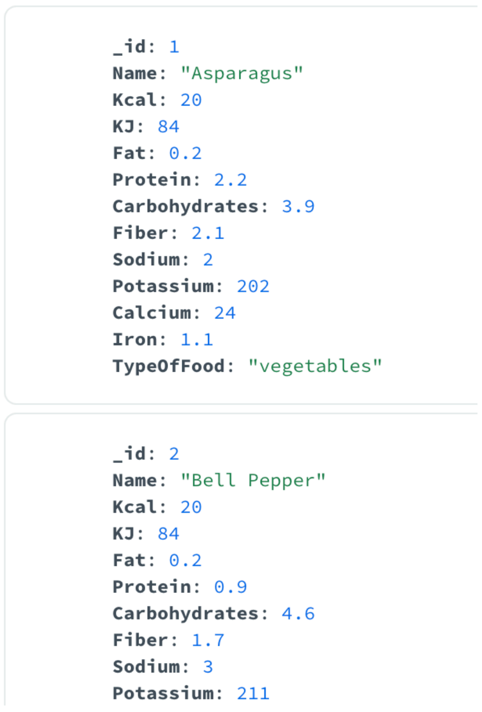
Before we add the data to the collections we have to do a change in the JSON files. As MongoDB always renames the primary key with ”_id”, in order to relate the primary key in SQL with the primary key in MongoDB we have renamed the primary key in the JSON files as ”_id”.

These is the appearance of the different collections in MongoDB: (Fig. 19-24)



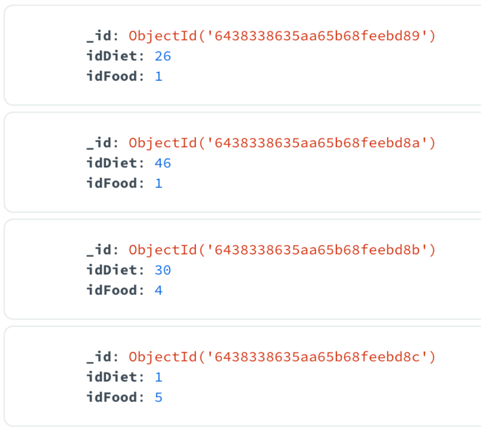
```
_id: 1
Description: "Dieta baja en grasas"
Creator: 3

_id: 2
Description: "Dieta sin lactosa para veganos"
Creator: 2

_id: 3
Description: "Dieta rica en proteinas para deportistas"
Creator: 1

_id: 4
Description: "Dieta vegetariana equilibrada"
Creator: 4
```

Figure 19: Diet Collection in MongoDB



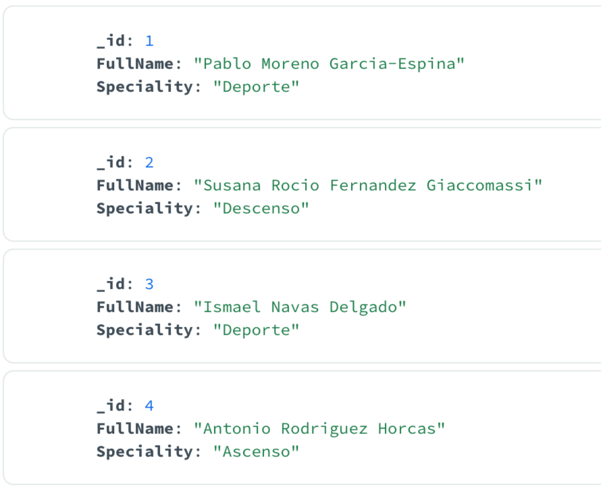
```
_id: 1
Name: "Asparagus"
Kcal: 20
KJ: 84
Fat: 0.2
Protein: 2.2
Carbohydrates: 3.9
Fiber: 2.1
Sodium: 2
Potassium: 202
Calcium: 24
Iron: 1.1
TypeOfFood: "vegetables"

_id: 2
Name: "Bell Pepper"
Kcal: 20
KJ: 84
Fat: 0.2
Protein: 0.9
Carbohydrates: 4.6
Fiber: 1.7
Sodium: 3
Potassium: 211
```

Figure 20: Food Collection in MongoDB



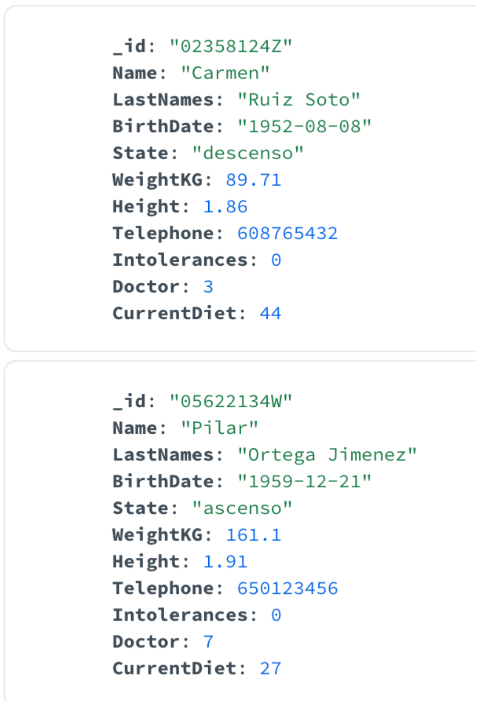
```
_id: ObjectId('6438338635aa65b68feebd89')
idDiet: 26
idFood: 1

_id: ObjectId('6438338635aa65b68feebd8a')
idDiet: 46
idFood: 1

_id: ObjectId('6438338635aa65b68feebd8b')
idDiet: 30
idFood: 4

_id: ObjectId('6438338635aa65b68feebd8c')
idDiet: 1
idFood: 5
```

Figure 21: ListOfFood Collection in MongoDB



```
_id: 1
FullName: "Pablo Moreno Garcia-Espina"
Speciality: "Deporte"

_id: 2
FullName: "Susana Rocio Fernandez Giaccomassi"
Speciality: "Descenso"

_id: 3
FullName: "Ismael Navas Delgado"
Speciality: "Deporte"

_id: 4
FullName: "Antonio Rodriguez Horcas"
Speciality: "Ascenso"
```

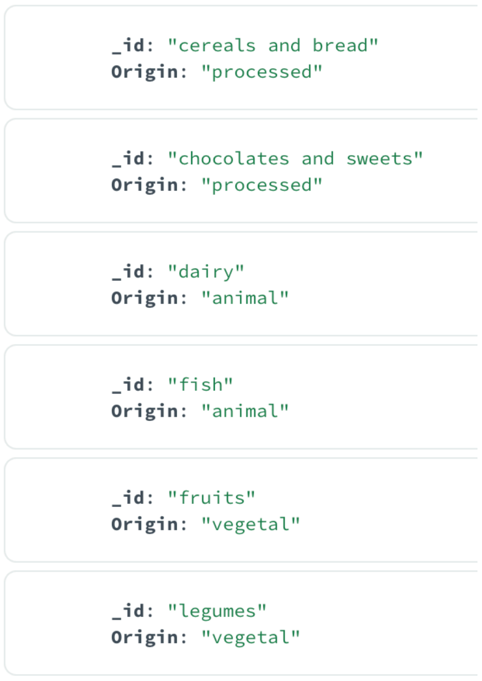
Figure 22: Nutritionist Collection in MongoDB



```
_id: "02358124Z"
Name: "Carmen"
LastNames: "Ruiz Soto"
BirthDate: "1952-08-08"
State: "descenso"
WeightKG: 89.71
Height: 1.86
Telephone: 608765432
Intolerances: 0
Doctor: 3
CurrentDiet: 44
```

```
_id: "05622134W"
Name: "Pilar"
LastNames: "Ortega Jimenez"
BirthDate: "1959-12-21"
State: "ascenso"
WeightKG: 161.1
Height: 1.91
Telephone: 650123456
Intolerances: 0
Doctor: 7
CurrentDiet: 27
```

Figure 23: Patient Collection in MongoDB



```
_id: "cereals and bread"
Origin: "processed"
```

```
_id: "chocolates and sweets"
Origin: "processed"
```

```
_id: "dairy"
Origin: "animal"
```

```
_id: "fish"
Origin: "animal"
```

```
_id: "fruits"
Origin: "vegetal"
```

```
_id: "legumes"
Origin: "vegetal"
```

Figure 24: TypeOfFood Collection in MongoDB

3.2 NoSQL Database Queries and Pipelines

In this part of the project we are going to translate the different SQL queries we designed for our database into MongoDB language.

In the first MongoDB query we see the following:

```
[
  {
    $match: {
      Protein: {
        $gt: 20,
      },
    },
  },
  {
    $match: {
      TypeOfFood: "meat",
    },
  },
  {
    $project: {
      _id: 0,
      Name: 1,
      Protein: 1,
    },
  },
]
```

Firstly we have used match to extract all of the food with protein levels higher than twenty grams, and then we have extracted out of them the ”meat” food. In order to obtain the same query as in SQL, we have made a project to stay just with the fields name and protein.

In the second query we have:

```
[
  {
    $match: {
      FullName: "Pablo Moreno Garcia-Espina",
    },
  },
  {
    $lookup: {
      from: "Patient",
      localField: "_id",
      foreignField: "Doctor",
      as: "patients",
    },
  },
  {
    $project: {
      _id: 0,
      "patients.Name": 1,
    },
  },
]
```

We filter the FullName avoiding all excepting ”Pablo Moreno Garcia-Espina” and then we make a join from the table Nutritionist to the table Patient relating the identifier field and the Doctor one. Finally we project the patient’s name.

The third query in MongoDB is:

```
[
  {
    $lookup: {
      from: "Nutritionist",
      localField: "Creator",
      foreignField: "_id",
      as: "nutritionist",
    },
  },
  {
    $match: {
      "nutritionist.FullName":
        "Susana Rocio Fernandez Giaccomassi",
    },
  },
  {
    $project: {
      _id: 0,
      Description: 1,
    },
  },
]
```

We join the table Diet with Nutritionist by relating the fields Creator in Diet and identifier in Nutritionist. Then we remain only with the diets made by the nutritionist ”Susana Rocio Fernandez Giacomassi” and we project the diet’s description.

The fourth query is:

```
[
    {
        '$lookup': {
            'from': 'Nutritionist',
            'localField': 'Creator',
            'foreignField': '_id',
            'as': 'nutritionistDoctor'
        }
    }, {
        '$lookup': {
            'from': 'ListOfFood',
            'localField': '_id',
            'foreignField': 'idDiet',
            'as': 'foods'
        }
    }, {
        '$match': {
            'foods': {
                '$gt': {
                    '$size': 10
                }
            }
        }
    }, {
        '$project': {
            'Diets with more than ten foods': '$Description',
            'Creator': '$nutritionistDoctor.FullName'
        }
    }
])
```

Here we are doing an aggregation from the Diet collection, first we use a lookup to create an array with the nutritionists of each diet. Then we do another lookup with ListOfFood to have all the foods of a diet in the foods array. Then we use match to filter the arrays food with size greater than 10. Lastly, we project the description of the diets and their creator. Basically, we are searching the diets with more than ten foods.

The last query in MongoDB is the following one:

```
[
    {
        '$group': {
            '_id': '$Null',
            'avgProtein': {
                '$avg': '$Protein'
            },
            'foods': {
                '$push': '$$ROOT'
            }
        }
    }, {
        '$unwind': {
            'path': '$foods'
        }
    }, {
        '$addFields': {
            'existe': {
                '$gt': [
                    '$foods.Protein', '$avgProtein'
                ]
            }
        }
    }, {
        '$match': {
            'existe': True
        }
    }
]
```

We start using the stage "group" to calculate the average value of Protein of the collection Food, with the "avg" operator. Additionally, we create an array of all the documents in each group using "push". This operator is used to append a value or document to an array field, in this case we add the value ROOT to the "foods" array. ROOT represents the current document being processed. So, by using ROOT, the entire document with all the food items is added to the array. As result of this stage we obtain one document with an array containing all the food items and the average protein value.

The second stage is an unwind, that decontructs the “foods” array created and outputs a new document for each element in the array and the average Protein value.

Then we use addFields to add a boolean field called “existe” that indicates whether the Protein field of the document is greater than the average protein value. Here we use the “gt” operator.

The last stage is a match that filters the documents based on the “existe” field, selecting only those documents where existe is true. Finally, we obtain the documents where “existe” is true, i.e. those with a Protein value greater than the average.

4 XML

4.1 Python Export to XML from Queries

In this section we are going to use a python script to convert the JSON files obtained for MongoDB to XML files. The code run for this is the following one:

```
if __name__ == '__main__':
    # get the xml from an URL that return json
    print("Reading ...")
    data = readfromjson("TypeOfFood.json")
    print("Translating ...")
    f = open("TypeOfFood.xml", "a")
    f.write(json2xml.Json2xml(data).to_xml())
    f.close()
```

In this program, we use the function readfromjson() that reads the .json file and returns an object that represents the data.Then we open the new file .xml with open and we use 'a' as second parameter, to append the data. The library json2xml is used to convert the JSON object into a XML, and we write the XML object in the new .xml file.

Once we have the six tables converted to .xml files, we can upload them into ExistDB. We have a docker container with ExistDB, so we access using localhost:8080. We have chosen exide to create a collection. The steps performed to create the collection and upload the files are:

- Select directory

- File > Manage > Log in > Create collection

- We named it ”NutritionalDB”

- Double click in the collection

- Upload Files

Finally, we uploaded the files into the collection.

The format of Diet.xml is:

```
  <all>
<item type="dict">
  <_id type="int">1</_id>
  <Description type="str">Dieta baja en grasas</Description>
  <Creator type="int">3</Creator>
</item>
      ...
      ...
  <\all>
```

Food.xml structure:

```
  <foods>
<food type="dict">
  <_id type="int">1</_id>
  <Name type="str">Asparagus</Name>
  <Kcal type="int">20</Kcal>
  <KJ type="int">84</KJ>
  <Fat type="float">0.2</Fat>
  <Protein type="float">2.2</Protein>
  <Carbohydrates type="float">3.9</Carbohydrates>
  <Fiber type="float">2.1</Fiber>
  <Sodium type="float">2.0</Sodium>
  <Potassium type="float">202.0</Potassium>
  <Calcium type="float">24.0</Calcium>
  <Iron type="float">1.1</Iron>
  <TypeOfFood type="str">vegetables</TypeOfFood>
</food>
      ...
      ...
  <\foods>
```

ListOfFood.xml is structured as follows:

```
  <lists>
<list type="dict">
  <idDiet type="int">26</idDiet>
  <idFood type="int">1</idFood>
</list>
      ...
      ...
  <\lists>
```

Nutritionist.xml lists the doctors in the following structure:

```
  <nutritionists>
<nutritionist type="dict">
  <_id type="int">1</_id>
  <FullName type="str">Pablo Moreno Garcia-Espina</FullName>
  <Speciality type="str">Deporte</Speciality>
</nutritionist>
      ...
      ...
  <\nutritionists>
```

For the table Patient, we have the Patient.xml file:

```
  <patients>
<patient type="dict">
  <_id type="str">02358124Z</_id>
  <Name type="str">Carmen</Name>
  <LastNames type="str">Ruiz Soto</LastNames>
  <BirthDate type="str">1952-08-08</BirthDate>
  <State type="str">descenso</State>
```

```
    <WeightKG type="float">89.71</WeightKG>
    <Height type="float">1.86</Height>
    <Telephone type="int">608765432</Telephone>
    <Intolerances type="int">0</Intolerances>
    <Doctor type="int">3</Doctor>
    <CurrentDiet type="int">44</CurrentDiet>
  </patient>
        ...
        ...
    <\patients>
```

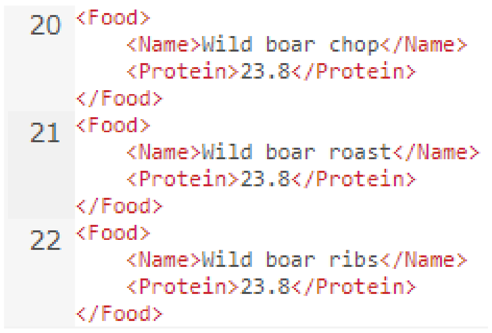
And lastly, the TypeOfFood.xml file has the different types of food:

```
    <types>
  <type type="dict">
    <_id type="str">cereals and bread</_id>
    <Origin type="str">processed</Origin>
  </type>
        ...
        ...
    <\types>
```

4.2 XML Database

Now that we have the tables in XML, we can translate our queries into XQuery language. The first query would be:

```
for $food in  doc("/db/NutritionalDB/Food.xml")/foods/food[
   TypeOfFood = 'meat']
where $food/Protein > 20
return <Food>
            <Name>{data($food/Name)}</Name>
            <Protein>{data($food/Protein)}</Protein>
        </Food>
```



```
<Food>
    <Name>Wild boar chop</Name>
    <Protein>23.8</Protein>
</Food>
<Food>
    <Name>Wild boar roast</Name>
    <Protein>23.8</Protein>
</Food>
<Food>
    <Name>Wild boar ribs</Name>
    <Protein>23.8</Protein>
</Food>
```

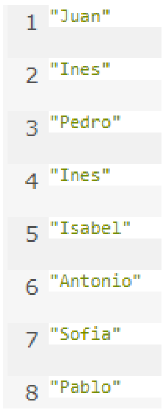
Figure 25: First XQuery output

We are returning the result in the xml structure, and we obtain 22 Food items, as it was expected (FIgure 25).

The following query asks for the patients treated by a specific doctor:

```
for $patient in doc("/db/NutritionalDB/Patient.xml")/patients/
   patient
for $nutritionist in doc("/db/NutritionalDB/Nutritionist.xml")/
   nutritionists/nutritionist
where $patient/Doctor = $nutritionist/_id and $nutritionist/
   FullName = 'Pablo Moreno Garcia-Espina'
return data($patient/Name)
```

And it gives us the 8 patients treated by the doctor (FIgure 26).



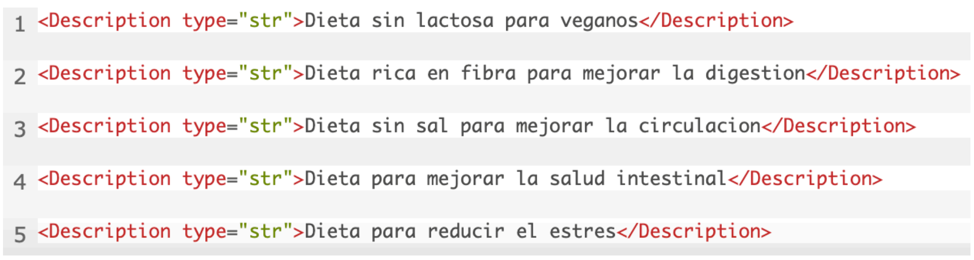
```
"Juan"
"Ines"
"Pedro"
"Ines"
"Isabel"
"Antonio"
"Sofia"
"Pablo"
```

Figure 26: Second XQuery output

The third query is used in order to determine the diets which have been created by one nutritionist:

```
for $diet in doc("/db/NutritionalDB/Diet.xml")/diets/diet
for $nutritionist in doc("/db/NutritionalDB/Nutritionist.xml")/
   nutritionists/nutritionist
where $diet/Creator = $nutritionist/_id and $nutritionist/
   FullName = "Susana Rocio Fernandez Giaccomassi"
return data($diet/Description)
```

It return 27 different diet descriptions (Figure 27)



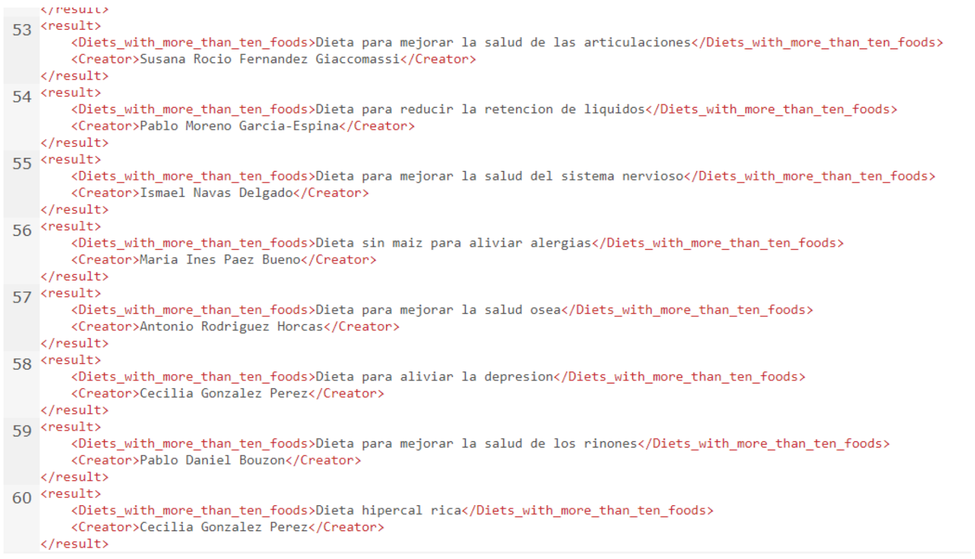
```
<Description type="str">Dieta sin lactosa para veganos</Description>
<Description type="str">Dieta rica en fibra para mejorar la digestion</Description>
<Description type="str">Dieta sin sal para mejorar la circulacion</Description>
<Description type="str">Dieta para mejorar la salud intestinal</Description>
<Description type="str">Dieta para reducir el estres</Description>
```

Figure 27: Third XQuery output

The fourth query returns the diet descriptions and the creator of the diet which has more than ten food in their diet.

```
for $diet in doc("/db/NutritionalDB/Diet.xml")/diets/diet
let $dietId := $diet/_id
where $dietId = doc("/db/NutritionalDB/ListOfFood.xml")/lists/
   list[idDiet = $dietId][count(../list) > 10]/idDiet
let $creator := doc("/db/NutritionalDB/Nutritionist.xml")/
   nutritionists/nutritionist[_id = $diet/Creator]/FullName
return <result>
    <Diets_with_more_than_ten_foods> {data($diet/Description)
   }</Diets_with_more_than_ten_foods>
    <Creator>{data($creator)}</Creator>
</result>
```

The values returned by this query are the following ones:



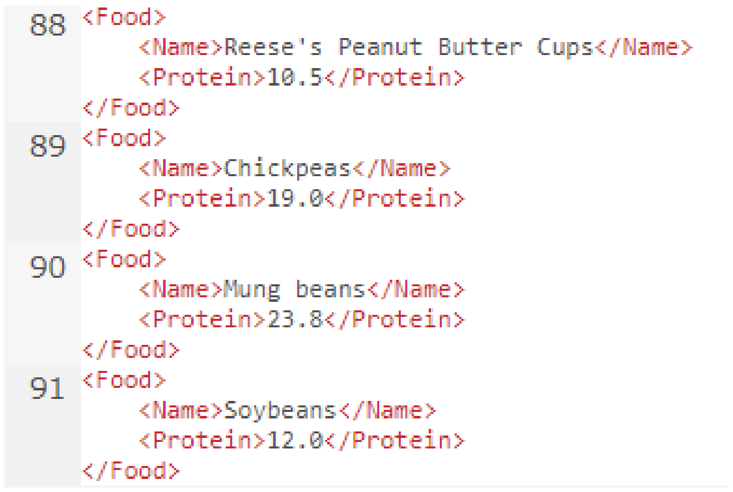
```
53 <result>
       <Diets_with_more_than_ten_foods>Dieta para mejorar la salud de las articulaciones</Diets_with_more_than_ten_foods>
       <Creator>Susana Rocio Fernandez Giaccomassi</Creator>
   </result>
54 <result>
       <Diets_with_more_than_ten_foods>Dieta para reducir la retencion de liquidos</Diets_with_more_than_ten_foods>
       <Creator>Pablo Moreno Garcia-Espina</Creator>
   </result>
55 <result>
       <Diets_with_more_than_ten_foods>Dieta para mejorar la salud del sistema nervioso</Diets_with_more_than_ten_foods>
       <Creator>Ismael Navas Delgado</Creator>
   </result>
56 <result>
       <Diets_with_more_than_ten_foods>Dieta sin maiz para aliviar alergias</Diets_with_more_than_ten_foods>
       <Creator>Maria Ines Paez Bueno</Creator>
   </result>
57 <result>
       <Diets_with_more_than_ten_foods>Dieta para mejorar la salud osea</Diets_with_more_than_ten_foods>
       <Creator>Antonio Rodriguez Horcas</Creator>
   </result>
58 <result>
       <Diets_with_more_than_ten_foods>Dieta para aliviar la depresion</Diets_with_more_than_ten_foods>
       <Creator>Cecilia Gonzalez Perez</Creator>
   </result>
59 <result>
       <Diets_with_more_than_ten_foods>Dieta para mejorar la salud de los rinones</Diets_with_more_than_ten_foods>
       <Creator>Pablo Daniel Bouzon</Creator>
   </result>
60 <result>
       <Diets_with_more_than_ten_foods>Dieta hipercal rica</Diets_with_more_than_ten_foods>
       <Creator>Cecilia Gonzalez Perez</Creator>
   </result>
```

Figure 28: Fourth XQuery output

The last query searches the food with more protein than the average:

```
let $avg-protein := avg(doc("/db/NutritionalDB/Food.xml")/foods
   /food/Protein)
for $food in  doc("/db/NutritionalDB/Food.xml")/all/item
where $food/Protein > $avg-protein
return <Food>
            <Name>{data($food/Name)}</Name>
            <Protein>{data($food/Protein)}</Protein>
       </Food>
```

We calculate the average value of protein for the entire file and then we search the items with more than the average, returning an xml structure (FIgure 29).



```
88 <Food>
       <Name>Reese's Peanut Butter Cups</Name>
       <Protein>10.5</Protein>
   </Food>
89 <Food>
       <Name>Chickpeas</Name>
       <Protein>19.0</Protein>
   </Food>
90 <Food>
       <Name>Mung beans</Name>
       <Protein>23.8</Protein>
   </Food>
91 <Food>
       <Name>Soybeans</Name>
       <Protein>12.0</Protein>
   </Food>
```

Figure 29: Last XQuery output

5 Conclusions and Futures Lines of Research

5.1 Conclusions

After creating a database for a nutrition center that includes information on nutritionists, patients, diets, and nutritional information on foods, we have optimized the database by creating indexes and transferring it to MongoDB and XML formats. This project provides an efficient and organized system for storing and accessing nutrition-related data, making it easier for nutritionists to manage their patients' dietary needs. By utilizing multiple formats, the database can be easily adapted to different platforms and programs, making it a valuable tool for the nutrition industry. Overall, this project showcases the importance of well-structured and optimized databases in the field of biological databases.

5.2 Future lines of Research

For future research, it could be valuable to explore additional optimization strategies that can enhance the database's performance and efficiency. One potential approach could involve implementing the InnoDB storage engine for data loading and utilizing a second database that leverages MyISAM and some Memory tables for query processing. This could help to streamline the system and optimize it further for both data loading and querying tasks.

In addition to exploring different storage engines and table types, it may also be beneficial to investigate other database management systems and technologies that could potentially improve the functionality and usability of the database.

As the field of biological databases continues to evolve, there are likely to be many exciting opportunities for future research and development in this area.