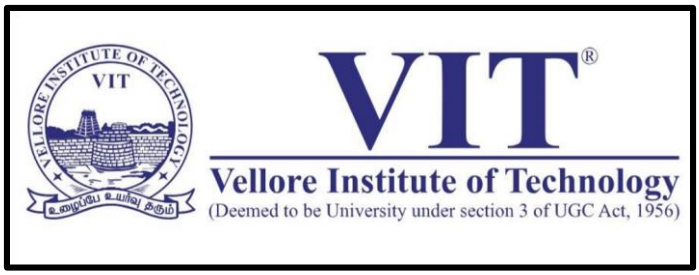


School of Computer Science and Engineering

J Component report

Programme : B.Tech

Course Title : Data Visualization

Course Code : CSE3020

Slot : D2

Title: Calcheck: A diet planner system for your BMI

GithHub: https://github.com/akash-r34/CalCheck

Team Members:

Akash R | 20BCE1501

Abrar Ahamed | 20BCE1437

Abstract

The purpose of this project is to build the CalCheck diet planning system, which will help users achieve their fitness and health goals by suggesting default diets depending upon their BMI and letting them create personalised diets. In order to assist users in making educated judgements about their dietary choices, the system also offers users insights on the nutritional content of various meals through visualisations. The system used Tableau to provide engaging visualisations of food and nutritional data in order to do this. People of different ages and backgrounds may use the system because it was made to be simple to use and open to everyone. The system was created with the goal of assisting users in making healthier decisions and enhancing their general health and well-being by offering personalised food advice and useful nutritional information. The project has a scope to create a mobile app using flutter and link the app with Tableau utilising the Tableau JavaScript API, as well as a web application and database to hold user data.

Keywords: Tableau, BMI, Diet recommendation.

Faculty: Dr. Parvathi. R **Sign:**

Date: 10/04/2023

Index

Note: Each of these sections contains subsections in the format 3.1, 3.2, etc.

1. Introduction

In today's fast-paced world, maintaining a healthy diet has become a challenge for many people. The lack of time, the abundance of unhealthy food options, and the ever-increasing stress levels have made it difficult for individuals to prioritize their nutritional needs. The importance of maintaining a healthy diet cannot be overstated. The food we eat plays a crucial role in our overall health and well-being. Poor dietary choices can lead to a variety of health problems such as obesity, heart disease, and diabetes. With the rise of sedentary lifestyles and fast-food culture, there has been a significant increase in the number of people struggling with weight issues and related health problems.

A diet planner app can be an effective tool for people looking to make positive changes in their eating habits. It provides a platform for users to track their daily food intake, set goals, and receive personalized recommendations for healthy food choices. The app can help users stay on track with their dietary goals by providing them with real-time feedback on their progress and suggesting adjustments to their diet as needed.

The purpose of this project is to design and develop a diet planner system that is user-friendly, effective, and tailored to the needs of its users. The system will utilize advanced algorithms to provide users with accurate and personalized recommendations based on their dietary preferences, fitness goals, and health conditions.

The visualization feature of this system will leverage Tableau to create interactive and informative visualizations of food and nutrient data. This will enable users to easily understand the nutrient content of different foods and make informed decisions about their dietary choices. The visualizations will be designed to be user-friendly and accessible to people of all ages and backgrounds. Thus, it provides users with valuable nutritional information in an easily understandable format.

In this report, we will discuss the features of the app, explain how they work, and discuss their benefits. We will also present the results of user testing and feedback and discuss areas for improvement. Finally, we will conclude with a discussion of the potential impact of the system on public health and future directions for the project**.**

2. Literature Review

Recent studies on humans show that calorie-restricted diets can delay the onset of ageing and lengthen life. These eating plans could boost mood and perhaps protect the body from agerelated ailments that wreak havoc. The diets function by lowering levels of free radicals, which are connected to chronic ailments, and by slowing the metabolic rate. ***Tsai and Wadden (2006)*** offers a summary of the background and development of very-low calorie diets (VLCDs) in addition to a thorough meta-analysis of the effects of VLCD therapies. With a mean weight loss of 1.2 kg per week and a total weight loss of 16.5 kg at the end of the intervention period, the authors discovered that VLCDs significantly reduced weight. Less lean body mass was lost than fat mass, which is what was responsible for the majority of this weight loss. In addition to weight loss, the authors found that patients who followed VLCD therapies had improved lipid profiles, blood pressure, and glycemic management. They also emphasised the possible dangers of VLCDs, such as the development of gallstones, electrolyte imbalances, and cardiac arrhythmias.

With the advent of technology and smartphones, ***Kratzke and Cox (2012)*** investigates how smartphone technology affects initiatives to promote health. The use of smartphones and mobile applications to enhance health outcomes and encourage healthy behaviours was examined. They point out that applications can be used to offer individualised feedback and assistance, simplify patient-provider contact, and encourage self-monitoring of health behaviours. Additionally, smartphone technology can assist people in underdeveloped or rural places in overcoming obstacles to accessing health information and resources. The usage of health-related applications may present several difficulties, including the possibility of erroneous or misleading information, privacy issues, and problems with user engagement and retention. They contend that additional study is required to properly comprehend how well these apps work to encourage behaviour change and enhance health outcomes.

Standen and Rothman's (2023) mainly focuses on two categories of mobile health apps: activity and calorie tracking apps. The writers give a summary of the present status of study on mobile apps while stressing both their advantages and disadvantages. In order to maximise the effectiveness of these applications as behavioural therapies, they also lay out a research

agenda for upcoming studies on them. The authors contend that calorie-tracking applications have the potential to be successful weight control and weight loss therapies. They point out that these apps can give users immediate feedback on their nutritional decisions and guide them towards making better decisions all day long. The authors are aware of these apps' potential drawbacks, such as the time-consuming nature of tracking food consumption and the possibility of erroneous data. The authors assert that activity-tracking applications have the potential to be successful interventions for raising physical activity levels. They point out that these apps can give users immediate feedback on their level of exercise and assist them in setting and achieving objectives. The authors are aware of these apps' potential drawbacks, such as the risk that users will engage in "gaming" behaviours to meet their activity objectives rather than participating in real physical exercise. The authors suggest a number of topics for additional investigation in order to maximise the efficacy of these apps as behavioural therapies. These include researching the effects of app design elements on user engagement and behaviour change, testing the efficacy of individualised support and feedback, and investigating the potential of social networks and gamification to increase app effectiveness.

Solbrig et al (2017) examines people's opinions on calorie tracking applications and their preferences for assistance in reaching weight loss objectives. The authors contend that although technology-based weight loss programmes are gaining popularity, little is known about how consumers view these interventions and their efficacy. 215 people who were trying to lose weight and had previously used a calorie counting app were polled by the authors. Participants' attitudes regarding calorie-counting applications were evaluated, along with their preferences for different kinds of support and incentive techniques. The survey's findings showed that respondents' opinions on calorie counting apps were divided. While many people enjoyed how easy it was to use an app to track their food consumption, the majority found the procedure to be time-consuming and tedious. As a result of exceeding their daily calorie limit, several users also mentioned feeling guilty or defeated, which had a detrimental effect on their enthusiasm to use the programme. Participants showed a great desire for individualised criticism and inspiration from a human coach or mentor in terms of support and motivation. Participants also mentioned the value of rewardbased systems, positive reinforcement, and social support from friends and family. They provided insight into how users felt about calorie tracking applications and what kind of assistance they preferred to use to reach their weight loss objectives. According to their findings, technology-based interventions have the

potential to be successful, but in order to have the greatest possible impact, they must be created with the requirements and preferences of the users in mind.

Adewumi et al (2018) contend that there is a market for mobile apps that can assist users in tracking their calorie intake and physical activity because of the rising popularity of smartphones and the interest in fitness and weight control. The authors provide details about how their calorie counter fitness app was created, including how agile software development methodology was used and how software developers, nutritionists, and fitness professionals worked together. The software gives users personalised recommendations for reaching their weight control objectives while allowing them to track their food intake and physical activity. The authors contend that the findings have significant ramifications for the creation of weight management mobile apps. They contend that the creation of efficient and user-friendly mobile apps can be achieved by a user-centered design strategy and cooperation between software developers, dietitians, and fitness professionals. In order to make sure that apps are customised to users' requirements and tastes, the authors also advise developers to undertake user testing and incorporate feedback from users throughout the design process.

Lieffers et al (2016) looks into how individuals access these apps and what they think about them. Although mobile apps have gained popularity as tools for weight management, the authors contend that little is known about how users see and interact with these programmes. The authors employed semi-structured interviews with 21 persons who had used freely downloadable dietary behaviour-change smartphone apps for weight control as part of a qualitative study. The study's objective was to learn more about how participants used these apps, as well as how they regarded both their advantages and disadvantages.

The study's findings showed that participants thought the mobile applications were useful for managing their weight, especially for keeping track of their caloric intake and physical activity. The fact that the apps may be used anywhere and at any time was also valued by the participants. Participants also said that the applications inspired them to make healthier food choices and helped them become more aware of their eating patterns. The authors assert that their findings have significant ramifications for the development and use of mobile apps that aim to influence dietary behaviour. They contend that in order to ensure that apps are suited to users' requirements and preferences, designers should give user-centered design top priority and incorporate user feedback at every stage of the design process. In order to allay

user concerns over data security and privacy, the authors also advise app developers to provide clear and transparent privacy policies.

The creation of a smartphone application that gives users personalised meal recommendations based on their dietary needs and preferences is described by ***Rehman et al* (2017)**. The authors contend that it is crucial to create tools that can assist people in making informed food decisions due to the rising prevalence of obesity and related health issues. The authors outline the creation of their smart food recommendation system, which analyses users' eating patterns and offers tailored food recommendations using a combination of rule-based and machine learning algorithms. The software also enables users to monitor their food intake and receive feedback on how they are doing with regard to reaching their dietary objectives. They contributed insightful information to the creation of dietary management smartphone apps. Their findings demonstrate the potential for intelligent meal recommendation systems to support people in making better food decisions and achieving their nutritional objectives.

A diet monitoring and recommendation system is described by ***Agapito et al. (2016)***. It seeks to offer consumers personalised and adaptable food suggestions based on their dietary objectives, tastes, and nutritional needs. By choosing healthier dietary choices, the system is intended to help people achieve their goals for health and wellness. The system employs a mobile application to gather information on a user's eating choices, routines, and health. The data is then examined by machine learning algorithms to produce individualised dietary recommendations that are catered to the needs and objectives of the user. The system also gives the user feedback on their eating patterns so they may monitor their advancement towards their objectives.

The authors present a compelling argument for the potential of such systems to assist people in reaching their health and wellness goals and enhancing their eating behaviours. The study's conclusions imply that mobile applications can be an efficient method for gathering and analysing dietary data, and that machine learning algorithms can be utilised to provide personalised meal choices that are pertinent and helpful to individuals.

In order to encourage healthy eating practices, ***Ojokoh and Babalola (2016)*** explore the creation of a personalised diet recommender system. The system generates a customised diet

plan for the user by taking into account a number of variables, including age, gender, weight, height, and level of physical activity. The authors contend that as they can take into account individual characteristics and offer customised counsel, personalised food suggestions can be a useful remedy for this issue. The several methods that have been employed to create diet recommender systems, such as rule-based systems and machine learning-based systems, are then discussed. The technology is built on a machine learning algorithm that creates customised diet programmes by using a database of food products and their nutritional data. The authors explain how the algorithm creates a plan that is both healthy and pleasurable by taking into account a number of variables, including the user's dietary choices, food allergies, and medical issues.

3. Materials and methods

3.1 Information about models

We don’t use any machine learning or statistical models in this project. The project focusses more on data collection, visualization, and app development. So, there is no pseudocode or algorithm.

However, the methodology used in this project combines data collection, preprocessing, analysis, visualization, application development, and integration to create a comprehensive tool for improving health and fitness.

3.2 Dataset

We consider the dataset to be the biggest novelty of this project as none of them were available in the web for use and we had to build them from scratch using APIs and web mining.

The Indian food dataset used in this project was created by extracting data from the USDA FoodData Central (FDC) API using a python code. The code specifies a list of FDCIDs of Indian food items and extracts information such as the name of the food item and its nutrient composition in a JSON format. The extracted information is stored in a list of dictionaries, where each dictionary corresponds to a single food item.

The Default diet plan dataset is created by assigning a dietID to every food item in the Indian food dataset. The User and recommended minimum macros intake dataset is generated using another python code that generates data for 100 users in the form of height and weight, calculates their BMI, and categorizes them into four categories based on BMI. The code then mines a website to find the minimum macro intake data for each user based on their height and weight, merges the BMI and macro intake data into a single pandas dataframe, and exports it to an excel file.

The User diet recommendation dataset is created using a python code that assigns recommended diet IDs to all users based on their minimum macro levels intake. The code reads in two datasets, the user macros dataset, and the diet plan dataset, aggregates the diet plans by their macro levels, creates a heatmap for better interpretation, and assigns recommended diet IDs to all users based on their minimum macro levels intake. The resulting dataset is exported as an Excel file.

3.3 Architecture and explanation

The project "CalCheck" aims to develop a system that assists users in achieving their health and fitness goals by recommending default diets based on their BMI and enabling them to build customized diets. The system will also provide users with insights into the nutrient content of various foods through visualizations, helping them make informed decisions about their dietary choices.

The architecture of the project involves several components:

Data Collection:

The first step in the architecture involves collecting data on various food items and their nutrient compositions. This is done using a python code that makes HTTP requests to the USDA FoodData Central (FDC) API to extract data on various food items. The code creates a dataset of Indian food items by specifying a list of FDCIDs of those items available in the FDC database. It extracts information such as the name of the food item and its nutrient composition in a JSON format.

Data Processing:

Next, another python code generates data for 100 users in the form of height and weight. It then calculates the BMI of each user and categorizes them into four categories based on BMI - "Underweight", "Normal", "Overweight", "Obesity". The code then mines the website "https://www.calculator.net/macro-calculator.html" to find the minimum macro intake data for each user based on their height and weight. It merges the BMI and macro intake data into a single pandas data frame and exports it to an excel file to create a User and recommended minimum macros intake dataset.

Data Visualization:

Tableau software is used to create interactive and informative visualizations of food and nutrient data. The app will leverage Tableau to provide users with insights into the nutrient content of various foods through visualizations, helping them make informed decisions about their dietary choices. All the visualizations are stored as Tableau workbooks.

In summary, the CalCheck project involves data collection, processing, and visualization to provide users with personalized diet recommendations and valuable nutritional information. The project will also be further proceeded into app development and database management to store user information and provide users with a user-friendly and accessible interface.

4. Proposed Works

4.1 Novelty

The project's novelty lies in the use of Tableau to create interactive and informative visualizations of food and nutrient data. The system leverages this data to provide users with personalized diet recommendations and valuable nutritional information, helping them make healthier choices and improve their overall health and well-being.

The project also generates data for 100 users in the form of height and weight, calculates the BMI of each user, and categorizes them into four categories based on BMI - "Underweight," "Normal," "Overweight," "Obesity." It then assigns recommended diet IDs to all users based on their minimum macro levels intake, helping them achieve their fitness goals.

The system provides a dashboard for users to view their progress towards their goals and visualize their calorie intake through charts and graphs

4.2 Project contributions

CalCheck's contributions lie in its ability to provide users with personalized diet recommendations and valuable nutritional information, helping them make healthier choices and achieve their fitness goals. The use of Tableau to create interactive and informative visualizations of food and nutrient data is a novel approach that sets this project apart from other similar applications.

5. Results and discussion

5.1 Results

The project has several positive outcomes. The system provides useful information to users about their diets, helps them make informed decisions, and educates them about the macro levels present in various diets. The use of Tableau to visualize data and information is also a positive aspect of the system, as it can make the information more accessible and understandable for users. Additionally, the ability to visualize macro details of each food item available in the database using interactive Tableau designs can be helpful for users in planning their meals and tracking their nutrient intake. Overall, these features can potentially contribute to improved dietary habits and health outcomes for users.

5.2 Figures and Tables

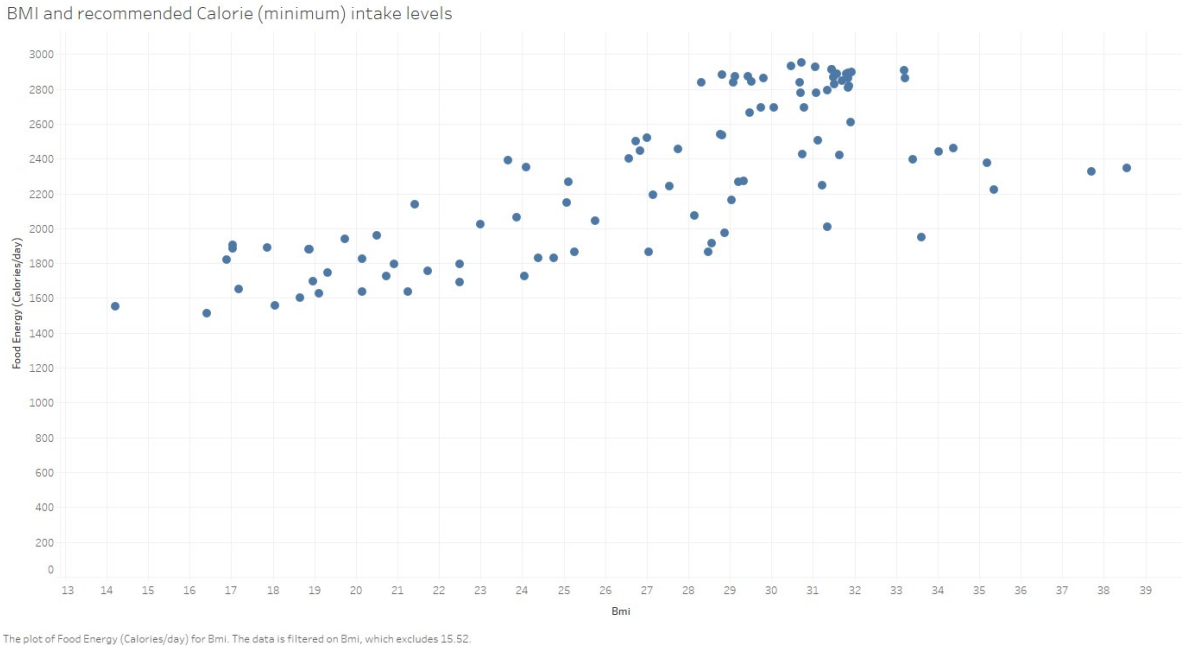


Figure 1: BMI and recommended Calorie (minimum) intake levels

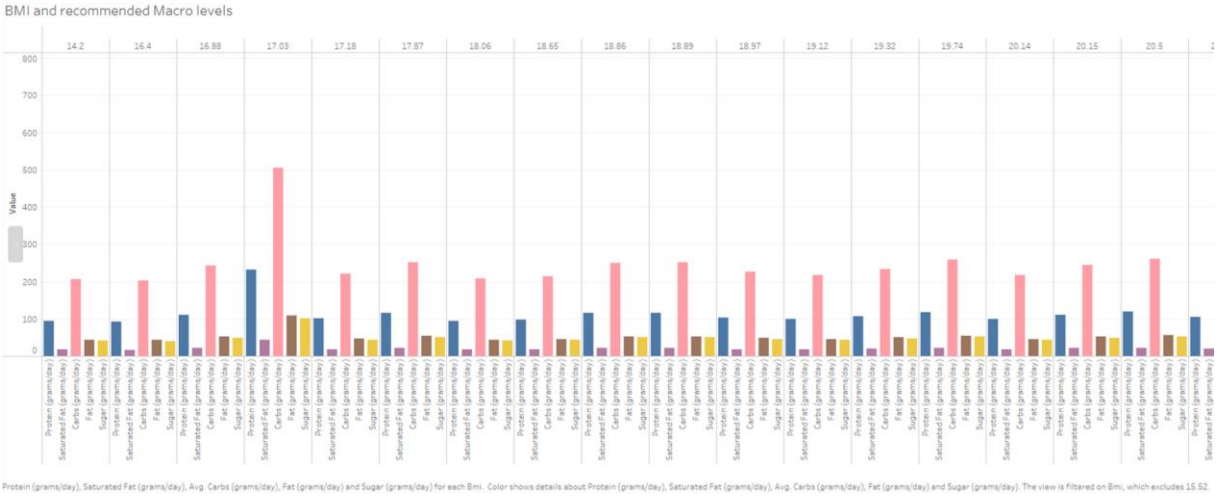


Figure 2: BMI and recommended Macro levels

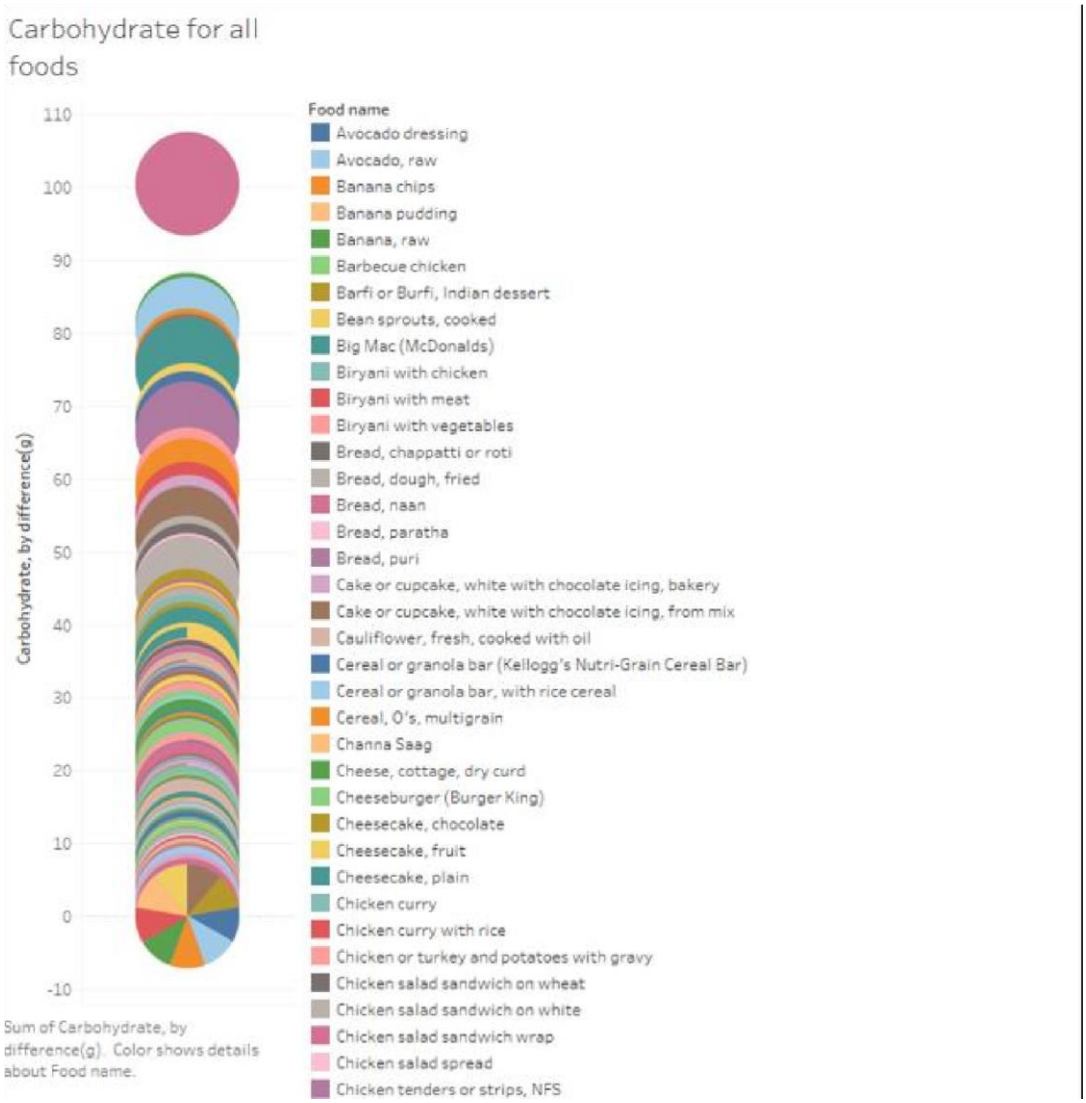


Figure 3: Carbohydrate for all foods in the database

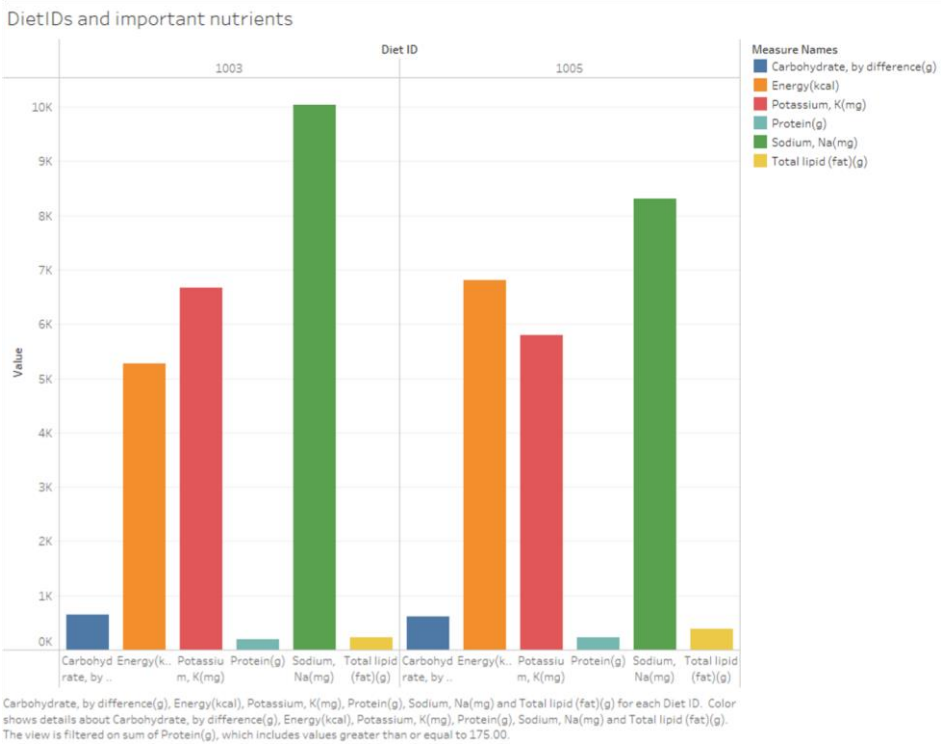
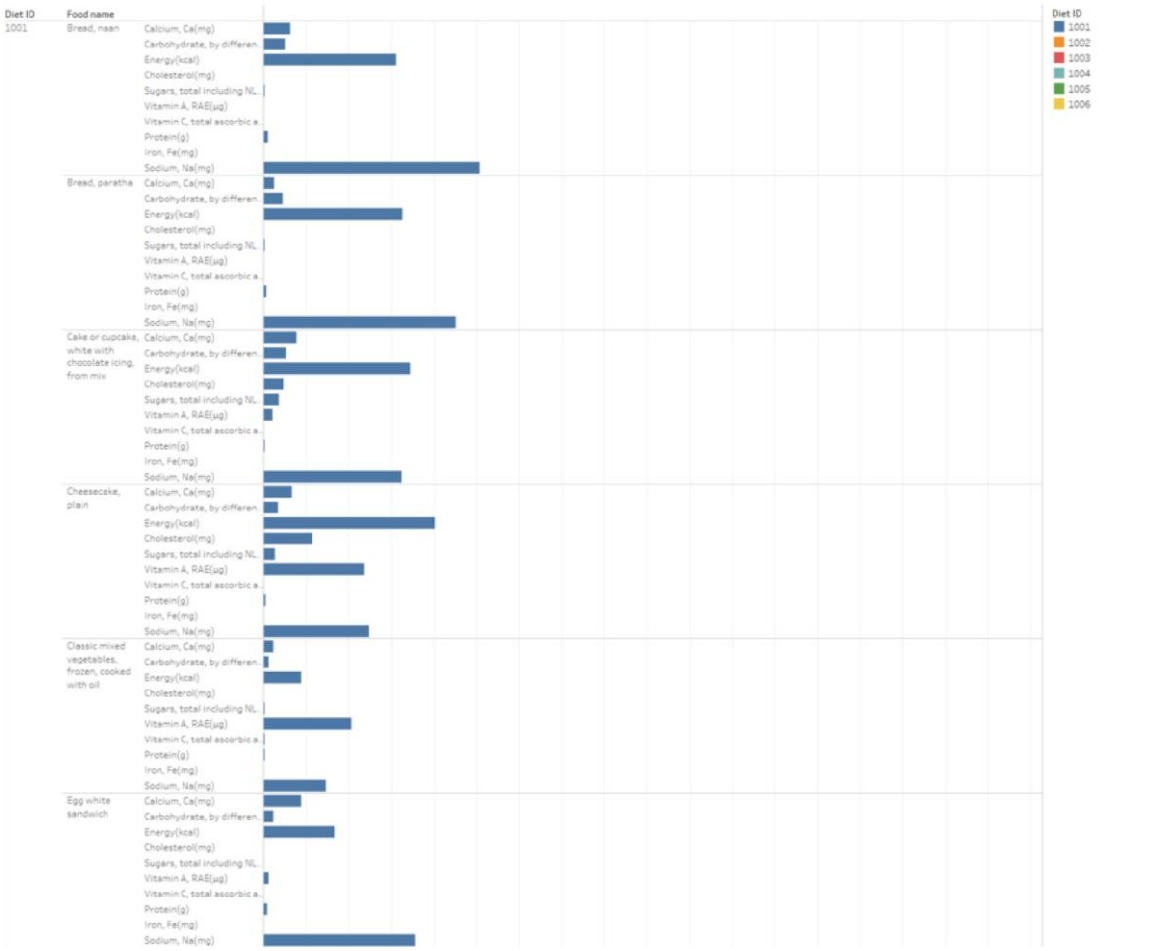


Figure 4: Specific DietIDs and important nutrients

DietIDS, food and Nutrients



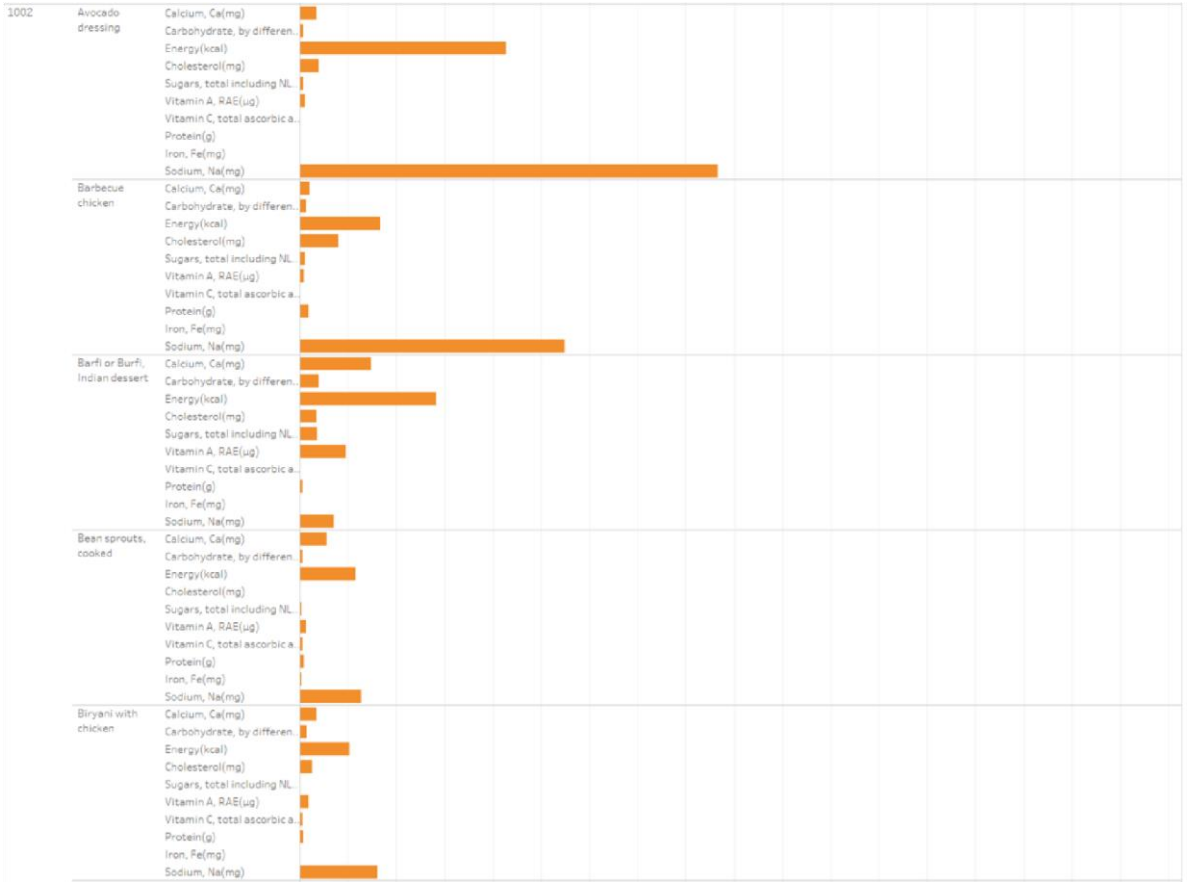


Figure 5: Specific DietIDs, food and Nutrients

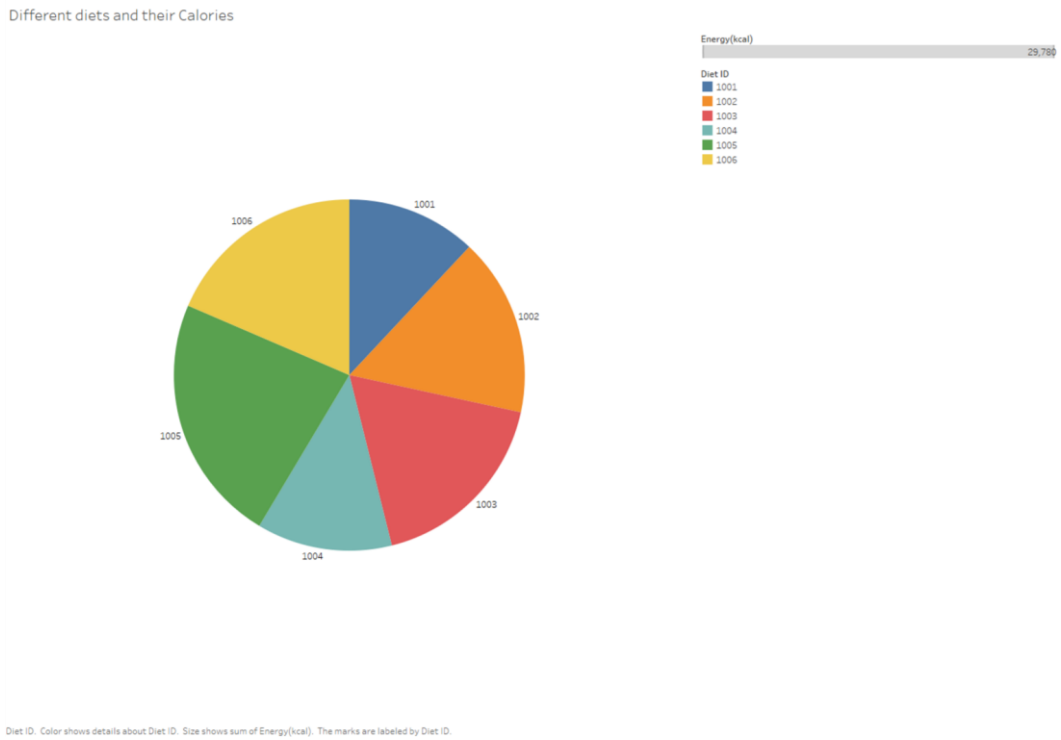


Figure 6: Different diets and their Calories

Figure 7: Food and all measure variables(nutrients)

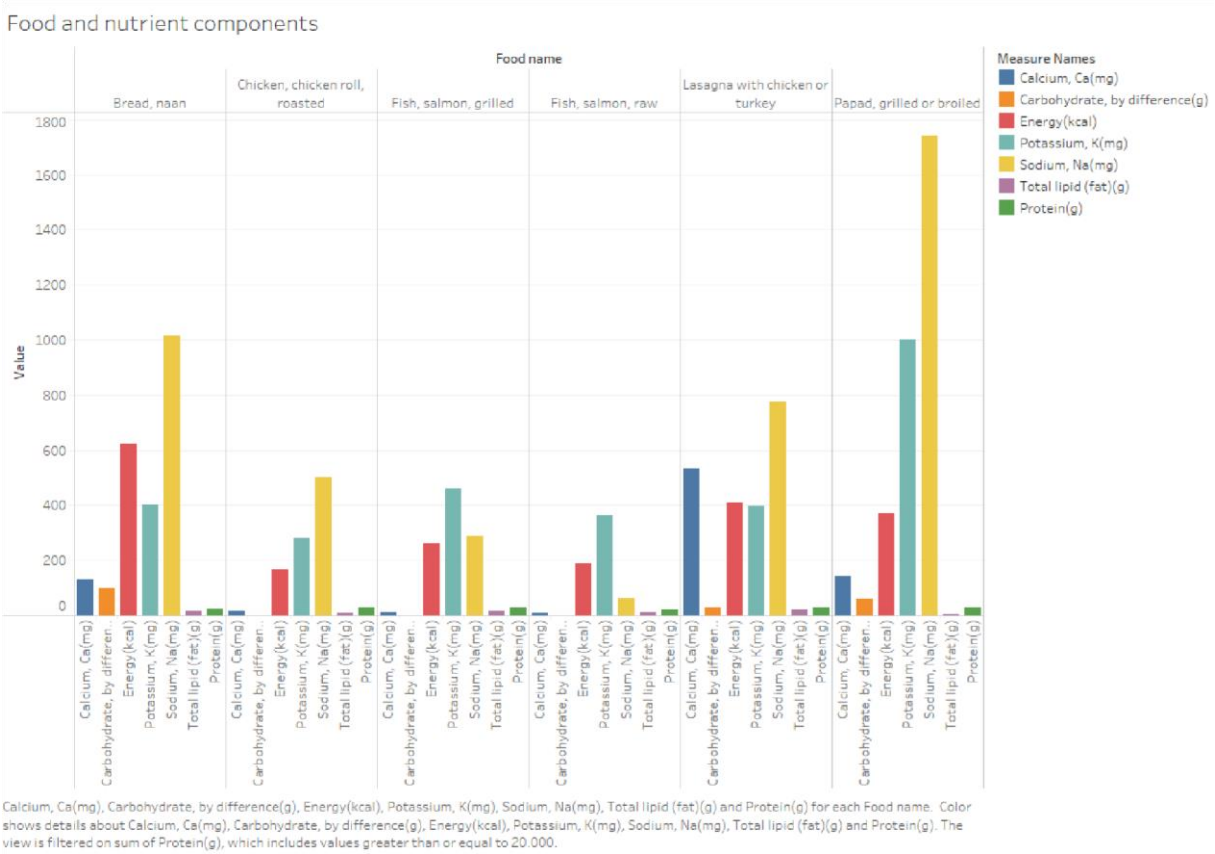


Figure 8: Food and nutrient components

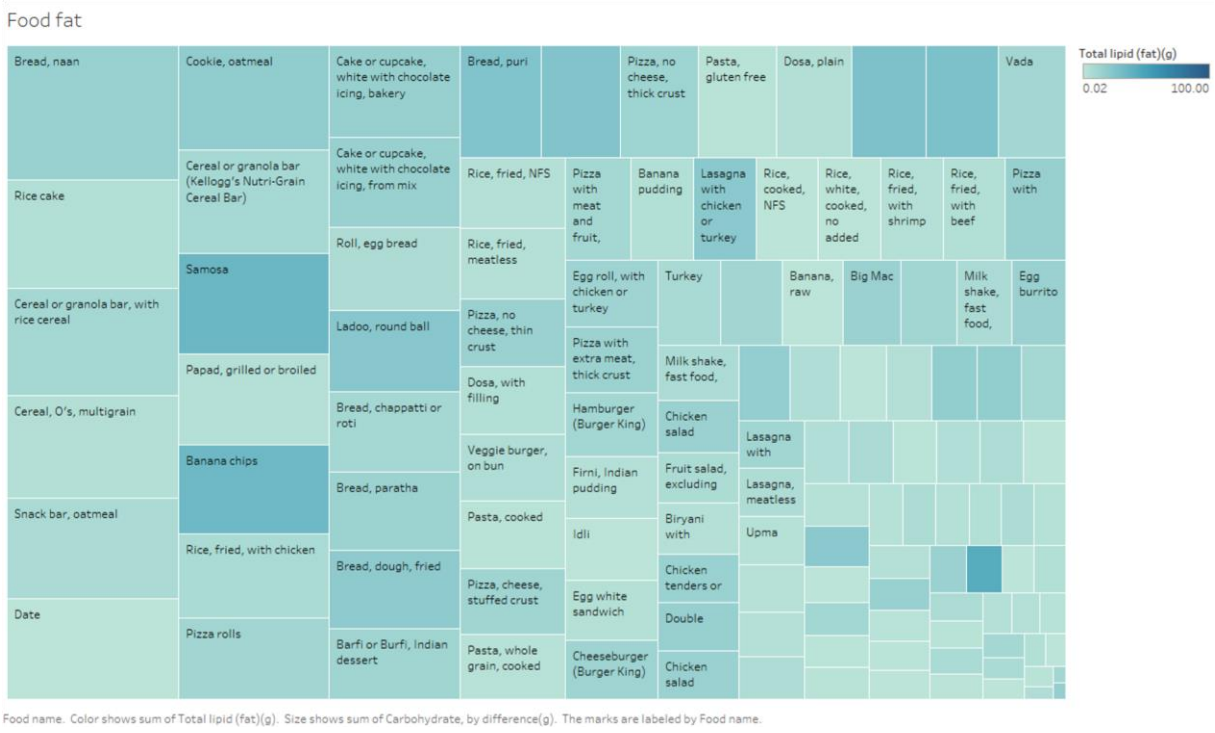


Figure 9: Food and their fat levels

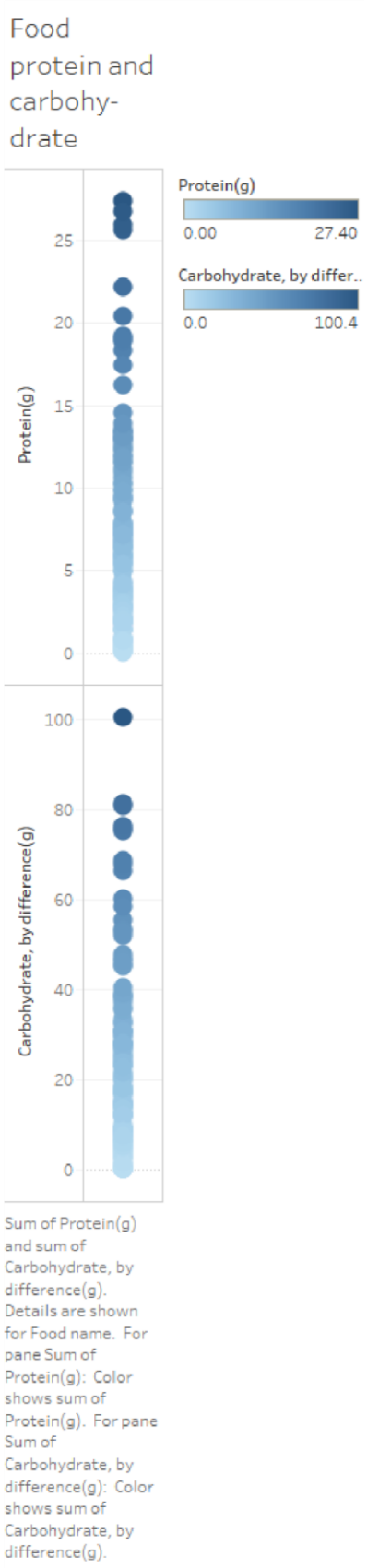


Figure 10: Food protein and carbohydrate levels

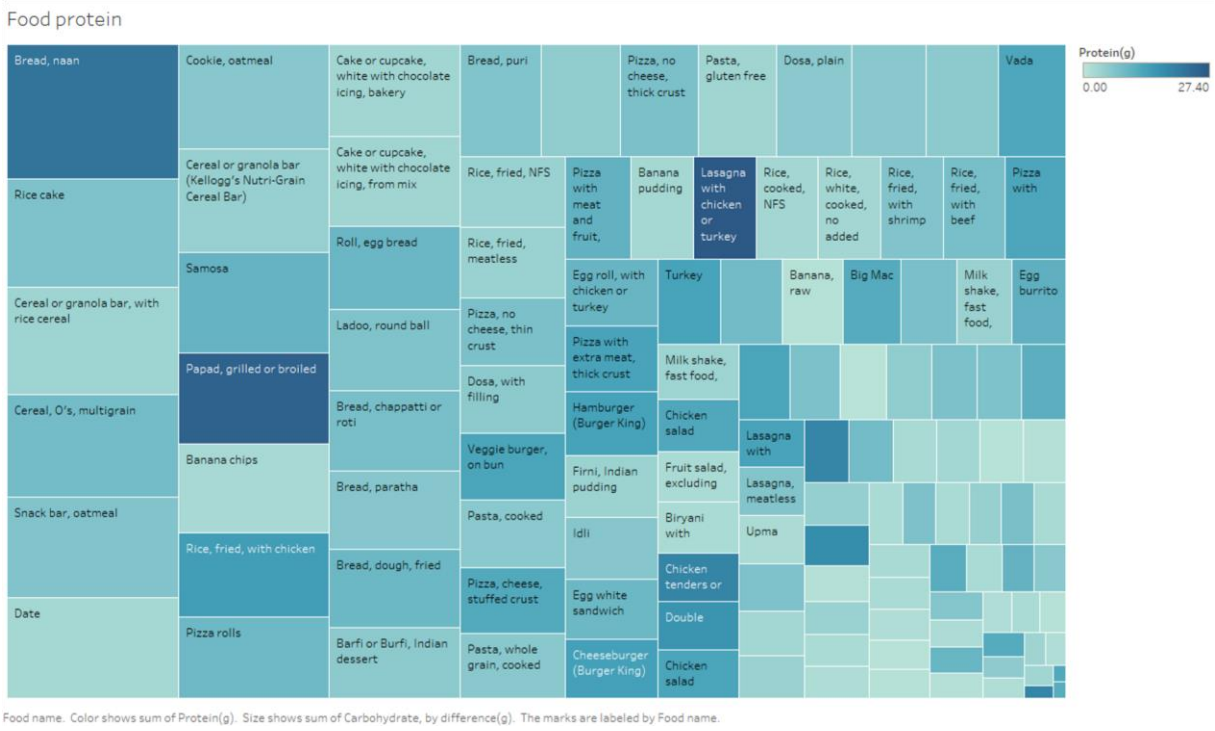


Figure 11: Food and their protein levels

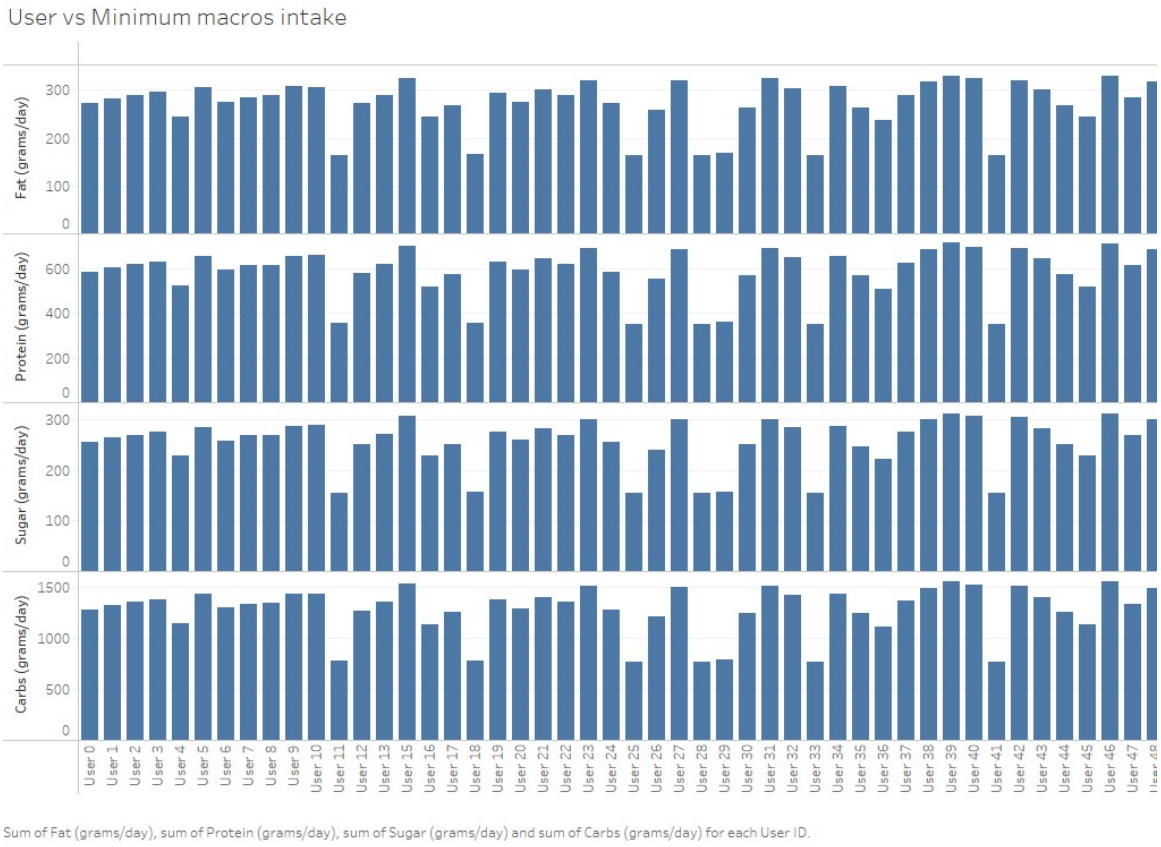


Figure 12: User vs Minimum macros intake

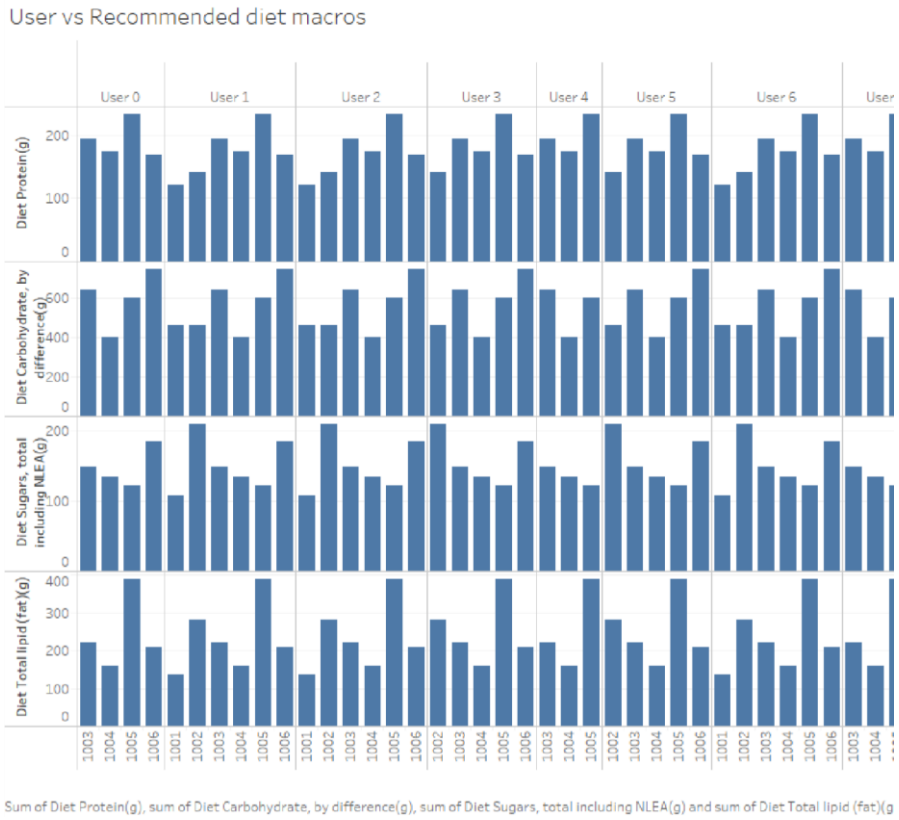
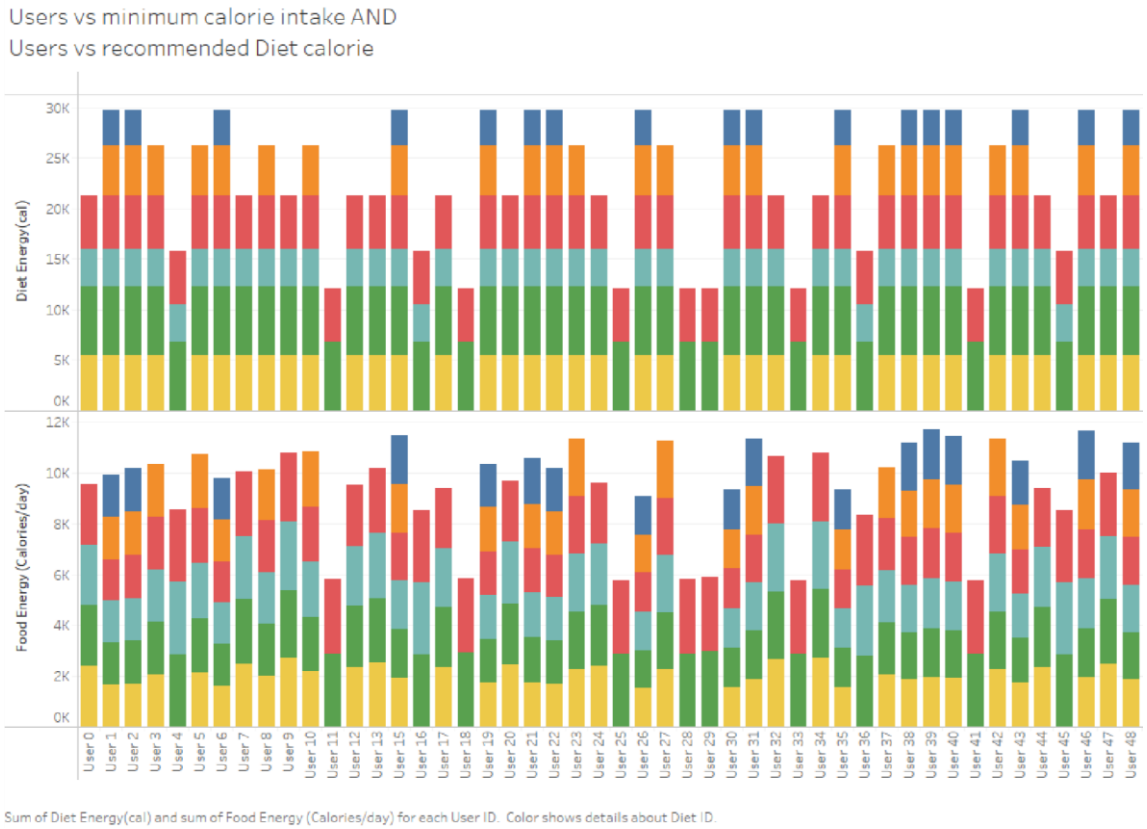


Figure 13: User vs Recommended diet macros

Figure 14: User's current health status



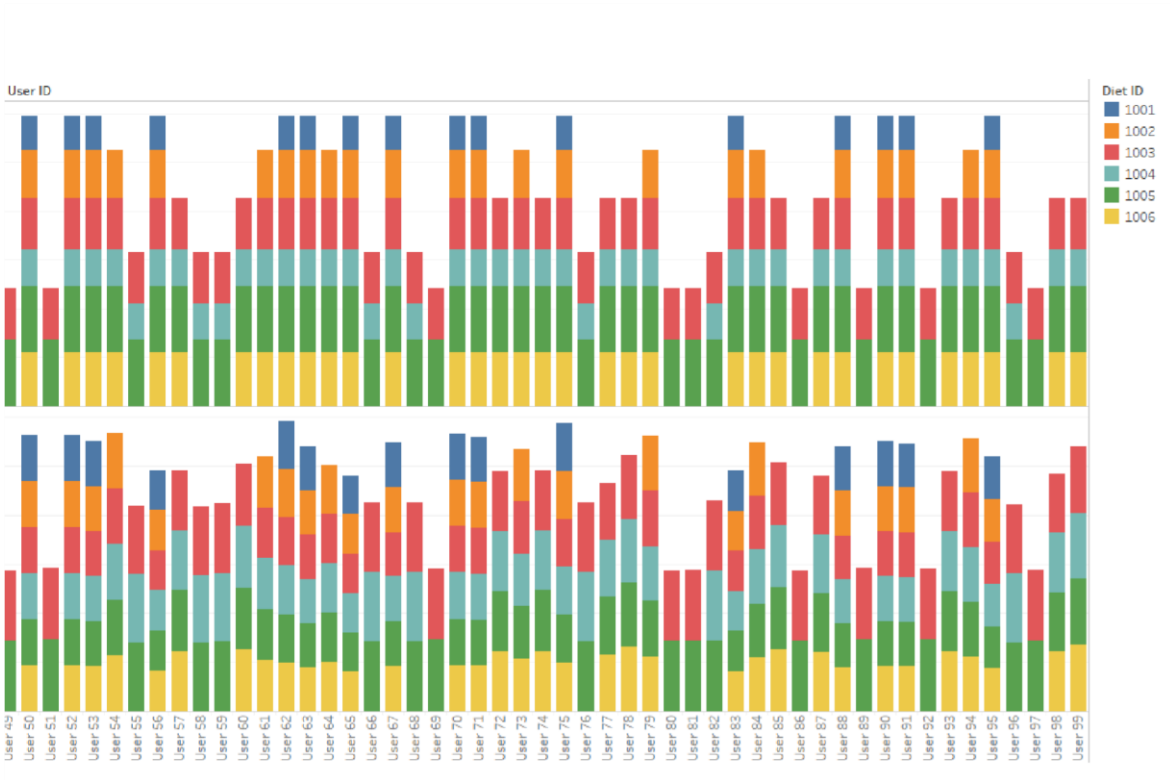


Figure 14: Users vs minimum calorie intake AND

dietID	Protein(g)	Carbohydrate, by difference(g)	Total lipid (fat)(g)	Sugars, total including NLEA(g)	Energy(kcal)
1001	121.94	462.87	137.07	107.18	3565
1002	141.36	464.99	282.09	210.02	4898
1003	194.94	638.27	219.39	149.12	5271
1004	174.13	399.36	158.96	134.79	3714
1005	233.33	603.83	390.02	121.57	6812
1006	168.29	748.45	209.06	185.22	5520

Table 1: Default diet plans and the macro levels associated with them

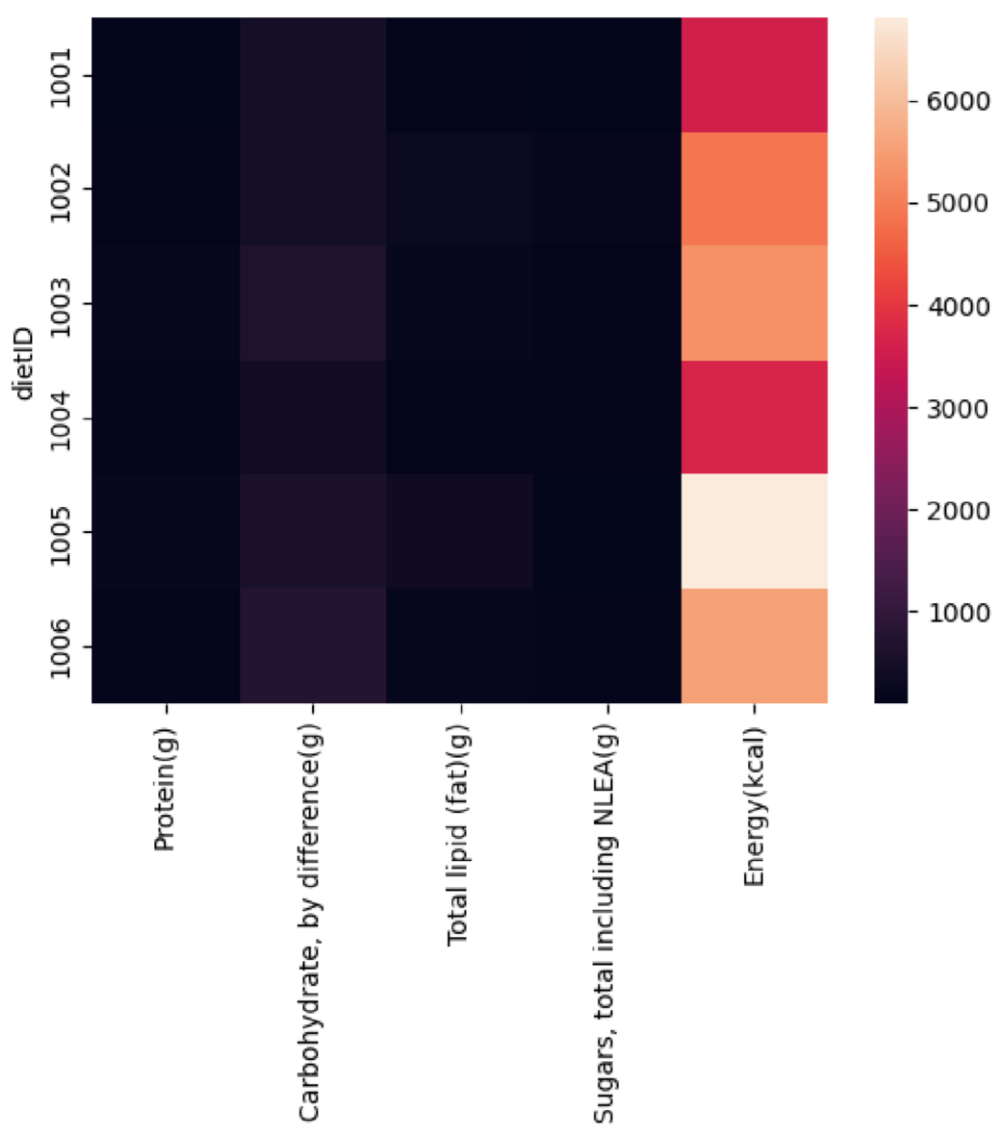


Figure 15: Heatmap of data in Table 1 using seaborn library in python

Most of the visualizations given in this report are not in their accurate format. Some Visualization images are cropped and displayed for better visibility in the report. You can view them in their proper format through our GitHub link. You can also visualize them for yourself with the twb files provided in our repository.

5.3 Explanation

The system with the aid of Tableau, a powerful data visualization tool, to create visually engaging and informative charts and graphs to help users make informed decisions about their diet. The visualization aspect of the system is designed to provide users with an easy-to-understand overview of their daily macro-nutrient intake, which includes carbohydrates, protein, and fat.

One of the key features of the system is that it provides default diet plans based on the user's BMI. This means that the system recommends a specific macro-nutrient breakdown for the user based on their body mass index. The system also educates the user about the macro-nutrient levels present in these diets using visually engaging and easy-to-understand visualizations.

For example, the app might recommend a diet with a macro-nutrient breakdown of 40% carbohydrates, 30% protein, and 30% fat for a user with a BMI of 25. The system would then display a pie chart or bar graph that shows the user the breakdown of their daily macronutrient intake in an easy-to-understand way.

Additionally, the system allows the user to visualize the macro details of each food item available in the database using interactive Tableau designs. For example, the user could search for a particular food item, such as "chicken breast," and the system would display a detailed visualization of the macro-nutrient breakdown of that food item, including the amount of carbohydrates, protein, and fat present in a serving size.

Overall, the visualization aspect of the system is designed to provide users with an easy-to understand and visually engaging way to make informed decisions about their diet as you can see in the Figures mentioned in the section Figures and Tables section.

6. Conclusion

We can see from the visualisation results that the application is well-designed and easy to use, making it a useful resource for anybody trying to improve their fitness and health. The CalCheck project's visualisations perform a good job of clearly showing the user's calorie intake and macronutrient breakdown, and the interactive elements enable a more in-depth investigation of the data. To encourage people to share their success with friends and family, social sharing options might be added in the future, along with more sophisticated features like machine learning algorithms for personalized suggestions.

7. References

1. Özer O, Phillips R, eds. (2012) The Oxford Handbook of Pricing Management (Oxford University Press, Oxford, UK).
2. Tsai, A. G., & Wadden, T. A. (2006). The evolution of very-low-calorie diets: an update and meta-analysis. Obesity, 14(8), 1283-1293.
3. Kratzke, C., & Cox, C. (2012). Smartphone technology and apps: rapidly changing health promotion. Global Journal of Health Education and Promotion, 15(1).
4. Solbrig, L., Jones, R., Kavanagh, D., May, J., Parkin, T., & Andrade, J. (2017). People trying to lose weight dislike calorie counting apps and want motivational support to help them achieve their goals. Internet interventions, 7, 23-31.
5. Standen, E. C., & Rothman, A. J. (2023). Capitalizing on the potential of mobile health applications as behavioral interventions: A research agenda for calorie-tracking and activity-tracking applications. *Social and Personality Psychology Compass*, e12731.
6. Ojokoh, B., & Babalola, A. (2016). A Personalized Healthy Diet Recommender System. *Organization for Women in Science for the Developing World (OWSD)*, *388*, 393.
7. G. Agapito *et al*., "DIETOS: A recommender system for adaptive diet monitoring and personalized food suggestion," *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, New York, NY, USA, 2016, pp. 1-8, doi: 10.1109/WiMOB.2016.7763190.

8. Adewumi, A., Olatunde, G., Misra, S., Maskeliūnas, R., & Damaševičius, R. (2018). Developing a calorie counter fitness app for smartphones. In Information Technology Science (pp. 23-33). Springer International Publishing.
9. Lieffers, J. R., Arocha, J. F., Grindrod, K., & Hanning, R. M. (2018). Experiences and perceptions of adults accessing publicly available nutrition behavior-change mobile apps for weight management. Journal of the Academy of Nutrition and Dietetics, 118(2), 229-239.
10. Rehman, F., Khalid, O., Bilal, K., & Madani, S. A. (2017). Diet-right: A smart food recommendation system. *KSII Transactions on Internet and Information Systems (TIIS)*, *11*(6), 2910-2925.

Appendix

GithHub: https://github.com/akash-r34/CalCheck

Dataset: https://github.com/akash-r34/CalCheck/tree/main/Dataset

Each dataset was created by us through api and web scraping, so the link provided will lead you to our GitHub.

Code and output

Creating Default diet plan dataset:

https://github.com/akash-r34/CalCheck/blob/main/Dataset/Mining%20food%20data/Mining_food_and_diet_data.ipynb

#Step 1: Creating Dataset Mining only the data of indian food items(using their FDCIDs) available in: https://fdc.nal.usda.gov/

```
import json import
requests
 fdcid_list = [2343829, 2343830, 2344617, 2343194, 2344618, 2344616,
2343195, 2345137, 2345138, 2342818,
              2342675, 2342680, 2342809, 2343139, 2341089, 2341103,
2341967, 2343193, 2343090, 2345911,
              2342675, 2342680, 2342809, 2343139, 2341089, 2341103,
2341967, 2343193, 2343090, 2345911,
              2343133, 2344673, 2342904, 2344473, 2341916, 2346076,
2345935, 2345988, 2342183, 2341493,
              2341858, 2341861, 2341852, 2341862, 2344240, 2344438,
2341504, 2342436, 2342435, 2342437,
              2343024, 2343886, 2343645, 2344437, 2345343, 2341636,
2345170, 2344634, 2345272, 2345652,
              2344234, 2344240, 2344242, 2344243, 2341156, 2342878,
2342877, 2342958, 2342335, 2342367, 2340907, 2340908, 2342361, 23423
58,
              2344170, 2344126, 2344102, 2344159, 2344158, 2344143,
2344155, 2343843, 2343842, 2343841, 2345258, 2341894,
              2342427, 2344187, 2342905, 2341957, 2344436, 2344438,
2344440, 2344441, 2343090, 2344214, 2341702, 2341704, 2341699, 23417
07, 2345425, 2345124, 2344239, 2345394,
              2345547, 2345539, 2345545, 2345651, 2345548, 2345551,
2345552, 2345549, 2344296, 2344431,
              2344795, 2344804, 2343374, 2343373, 2342026, 2343341,
2343343, 2343342, 2343892, 2344729,
              2344720, 2344695, 2341100, 2344665, 2344682, 2344719,
2345344, 2345765, 2345743, 2343882,
              2343427, 2343583, 2343596, 2343952, 2343566, 2344214,
2342906, 2345725, 2344698, 2345779, 2345094] ind=0
headers = {'accept': 'application/json'} data_arr = [] for
i, fdcid in enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}
   response = requests.get(url, params=params, headers=headers)

  output_json = response.json()   ind+=1   data_arr.append({'Food
name': output_json['description']})   for nutrient in
output_json['foodNutrients']:
data_arr[i][nutrient['nutrient']['name'] + '(' + nutrient['nutri
ent']['unitName'] +')'] = nutrient['amount']
```

##Step 2: Mining 10 items per code block to identify faulty api request(if any) with ease Note: This step can be skipped by just running the step 1 successfully

Each of these code block contains the same FDCIDs as the first block. The only difference is that the requests are sent 10 at a time.

```
fdcid_list = [2342675, 2342680, 2342809, 2343139, 2341089, 2341103,
2341967, 2343193, 2343090, 2345911]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}

  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
  data_arr.append({'Food name': output_json['description']})   for
nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1

fdcid_list = [2343133, 2344673, 2342904, 2344473, 2341916, 2346076,
2345935, 2345988, 2342183, 2341493]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}

  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
  data_arr.append({'Food name': output_json['description']})   for
nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1

fdcid_list = [2341858, 2341861, 2341852, 2341862, 2344240, 2344438,
2341504, 2342436, 2342435, 2342437]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}
```

```
  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
  data_arr.append({'Food name': output_json['description']})
  for nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1

fdcid_list = [2343024, 2343886, 2343645, 2344437, 2345343, 2341636,
2345170, 2344634, 2345272, 2345652]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}

  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
  data_arr.append({'Food name': output_json['description']})   for
nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1

fdcid_list = [2344234, 2344240, 2344242, 2344243, 2341156, 2342878,
2342877, 2342958, 2342335, 2342367, 2340907, 2340908, 2342361, 23423
58]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}

  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
  data_arr.append({'Food name': output_json['description']})   for
nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1

fdcid_list = [2344170, 2344126, 2344102, 2344159, 2344158, 2344143,
2344155, 2343843, 2343842, 2343841, 2345258, 2341894]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
```

```
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}

  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
  data_arr.append({'Food name': output_json['description']})   for
nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1

fdcid_list = [2342427, 2344187, 2342905, 2341957, 2344436, 2344438,
2344440, 2344441, 2343090, 2344214, 2341702, 2341704, 2341699, 23417
07, 2345425, 2345124, 2344239, 2345394]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}

  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
  data_arr.append({'Food name': output_json['description']})   for
nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1
```

Note: This block is an exception(contains more than 10 FDCIDs) as all of these FDCIDs requests were already checked for errors

```
fdcid_list = [2345547, 2345539, 2345545, 2345651, 2345548, 2345551,
2345552, 2345549, 2344296, 2344431,
              2344795, 2344804, 2343374, 2343373, 2342026, 2343341,
2343343, 2343342, 2343892, 2344729,
              2344720, 2344695, 2341100, 2344665, 2344682, 2344719,
2345344, 2345765, 2345743, 2343882,
              2343427, 2343583, 2343596, 2343952, 2343566, 2344214,
2342906, 2345725, 2344698, 2345779, 2345094]

headers = {'accept': 'application/json'} for i, fdcid in
enumerate(fdcid_list):   url =
'https://api.nal.usda.gov/fdc/v1/food/'+str(fdcid)   params
= {'format': 'full',
          'api_key': '3EOzG6fTmkUOQbhYpK0d0FWocwyAvKo4ty3RewDa'}

  response = requests.get(url, params=params, headers=headers)
output_json = response.json()
```

```
  data_arr.append({'Food name': output_json['description']})   for
nutrient in output_json['foodNutrients']:
data_arr[ind][nutrient['nutrient']['name'] + '(' + nutrient['nut
rient']['unitName'] +')'] = nutrient['amount']   ind+=1
```

#Step 3: Generating default Diets for the app We create 6 different diets, each identified by 6 different unique dietIDs.

##Randomly generating dietIDs Generate dietIDs in range from 1001 to 1006 for n(size of our food arrayy) times.

```
import random
 dietID = {'dietID': []}
 for i in range(len(data_arr)):
dietID['dietID'].append(random.randint(1001, 1006))
print(len(data_arr)) print(dietID)
```

```
145
{ 'dietID': [1003, 1006, 1006, 1002, 1003, 1006, 1005, 1002, 1002,  10
03 , 1004, 1001, 1003, 1003, 1002, 1002, 1006, 1005, 1003, 1001,  1005
, 1003, 1003, 1005, 1001, 1003, 1004, 1002, 1006, 1005, 1003, 1001,
1005, 1001, 1001, 1005, 1006, 1006, 1003, 1003, 1003, 1004, 1002, 10 04
, 1004, 1005, 1006, 1002, 1003, 1005, 1004, 1001, 1005, 1005,  1005
, 1006, 1003, 1004, 1006, 1004, 1002, 1001, 1003, 1005, 1002, 1003,
1001 , 1004, 1001, 1002, 1001, 1002, 1002, 1002, 1002, 1006, 1001,
10 03, 1006, 1002, 1004, 1004, 1001, 1002 , 1006, 1002, 1003, 1003,
1005 , 1003, 1006, 1003, 1006, 1001, 1003, 1002, 1003, 1003, 1001,
1003,
1003 , 1004, 1006, 1003, 1002, 1003, 1004, 1005, 1002, 1001, 1004,  10
01 , 1003, 1004, 1005, 1006, 1005, 1001, 1003, 1006, 1003, 1004,
1005 , 1003, 1004, 1006, 1006, 1005, 1006, 1002, 1005, 1001, 1001,
1001, 1003, 1001, 1006, 1001, 1001, 1005, 1004, 1002, 1001, 1005,
1005]}
```

##Assigning dietIDs Assign the generated dietIDs to each food item and store them in a panda dataframe.

```
import pandas as pd
from google.colab import data_table
data_table.enable_dataframe_formatter()
   df1 = pd.DataFrame.from_dict(data_arr)
dietIDs = pd.DataFrame.from_dict(dietID)
 result = pd.concat([dietIDs, df1], axis=1, join='inner')
   result
```

```
Warning: Total number of columns (67) exceeds max_columns (20). Fall
ing back to pandas display.
```

```
     dietID                                  Food name  Protein(g)  \ 0
1003                                        Idli        6.36
1      1006                                 Dosa, plain        5.70
2      1006                       Dosa, with filling        5.46    3
       1002                              Bread, puri        6.84
4      1003                                    Vada       12.80    ..
...                                        ...         ...
140    1004                             Lentil curry        3.67
141    1002                  Ghee, clarified butter        0.28
142    1001                                    Date        2.45
143    1005  Mayonnaise, reduced fat,  with olive oil        0.37    144
       1005                   Romaine lettuce, raw        1.36

     Total lipid (fat)(g)  Carbohydrate, by difference(g)  Energy(kc
al)  \
0                    0.35                          25.00          12
8.0
1                    4.05                          37.00          21
0.0
2                    4.27                          30.80          18
4.0
3                   24.90                           39.20          40 9.0
4                    9.73                          33.20          26 6.0
..                    ...                            ...           ...
140                  5.61                          12.30          11
                     0.0
141                 99.50                           0.00           87
                     6.0
142                  0.39                          75.00          28
2.0
143                 40.00                           0.00           36
1.0
144                  0.15                           2.87           1
                     5.0

     Alcohol, ethyl(g)  Water(g)  Caffeine(mg)  Theobromine(mg)  ...
\
0                  0.0     67.20           0.0              0.0  ...
1                  0.0     51.80           0.0               0.0
                   ...
2                  0.0     57.90           0.0              0.0  ...

3                  0.0     28.00           0.0              0.0  ...

4                  0.0     42.00           0.0              0.0  ...
..                 ...       ...           ...              ...  ...
```

```
140                  0.0     76.70           0.0               0.0  ... 141
0.0      0.24           0.0              0.0  ...
142                0.0     20.50           0.0               0.0
                   ...
143                0.0     57.60           0.0              0.0  ...

144                0.0     95.00           0.0              0.0  ...

     PUFA 20:4(g)  PUFA 22:6 n-3  (DHA)(g)  MUFA 16:1(g)  PUFA 18:4(g
)  \
0           0.000                       0.0         0.001
            0. 0
1           0.000                     0.0         0.014            0.
0
2           0.000                     0.0         0.014            0.
            0
3           0.001                     0.0         0.083
            0. 0
4           0.000                     0.0         0.034            0.
0
..            ...                     ...           ...           ..
.
140         0.000                     0.0         0.018            0.
            0
141         0.000                      0.0         2.230
            0. 0
142         0.000                     0.0         0.001            0.
0
143         0.000                     0.0         0.464            0.
            0
144         0.000                     0.0         0.002
            0. 0

     MUFA 20:1(g)  PUFA 20:5 n-3 (EPA)(g)  MUFA 22:1(g)  \
0           0.002                     0.0         0.000
1           0.027                     0.0         0.003
2           0.026                     0.0         0.002
3           0.167                     0.0         0.000
4           0.066                     0.0         0.000    ..
            ...                     ...           ...
140         0.030                     0.0         0.000
141         0.000                     0.0         0.000
142         0.000                     0.0         0.000
143         0.125                     0.0         0.000
144         0.000                     0.0         0.000

     PUFA 22:5 n-3 (DPA)(g)  Fatty acids, total monounsaturated(g)  \
```

```
0                       0.0                                    0.090  1
0.0                                 1.590

2                       0.0                                    1.610
3                       0.0                                    9.710
4                       0.0                                    3.810

..                      ...                                      ...
140                     0.0                                    1.890
141                     0.0                                   28.700

142                     0.0                                    0.036

143                     0.0                                   29.500

144                     0.0                                    0.006

     Fatty acids, total polyunsaturated(g)
0                                    0.142
1                                    1.610
2                                    1.720
3                                   10.000
4                                    3.950   ..
                                     ...
140                                  1.990
141                                  3.690
142                                  0.019
143                                  4.010
144                                  0.082

[145 rows x 67 columns]
```

#Step 4: Export the datasets

##Export the generated food dataset without dietIDs associated with them

```
file_name = 'Food_data_generated.xlsx'

# saving the excel df1.to_excel(file_name)
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```

##Export the generated food dataset with dietIDs associated with them

```
file_name = 'Food_data_generated_with_dietIDs.xlsx'

# saving the excel result.to_excel(file_name)
```

```
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```

Creating User and recommended minimum macros intake dataset

https://github.com/akashr34/CalCheck/blob/main/Dataset/Mining%20min.%20macros%20for%20users%20and%20recommend%20diet%20plans/Mining_user_macros_based_on_BMI.ipynb

#Task 1: Generate macros based on BMI for different users

#Step 1: Generating random height and weight for 100 users This process may generate some abnormal height and weight, which will be removed in the upcoming code blocks.

```
import random from random
import uniform
 height = [] weight = [] bmi = [] for i in range(100):
h = round(random.uniform(140.0,190.0),2)     if h >= 140.0
and h <= 150.0:
bmi.append([h,round(random.uniform(30.0,70.0),2)])
#         height += [h]
#         weight += [round(random.uniform(40.0,70.0),2)]
elif h >= 150.01 and h <= 170.0:
bmi.append([h,round(random.uniform(40.01,90.0),2)])
#         height += [h]
#         weight += [round(random.uniform(70.1,140.0),2)]
elif h >= 170.01 and  h <= 180.0:
bmi.append([h,round(random.uniform(70.01,100.0),2)])     else:
bmi.append([h,round(random.uniform(100.01,110.0),2)])
#     height += [round(random.uniform(120.0,200.0),2)]
```

#Step 2: Calculate BMI with the generated height and weight Additionaly we also give the BMI range based on the calculated BMI. This is in the form of categorical data with four types - "Underweight", "Normal", "Overweight", "Obesity".

```
bmi_list = [] for i in
bmi:     data = {}
data['Height'] = i[0]
data['Weight'] = i[1]
    data['bmi'] = round((i[1]/(i[0]**2))*10000,2)
if data['bmi'] < 18.5:         data["bmi_range"] =
```

```
"Underweight"     elif data['bmi'] >= 18.5  and
data['bmi'] < 24.9:
        data["bmi_range"] = "Normal"     elif
data['bmi'] >= 24.9 and data['bmi'] < 30.0:
data['bmi_range'] = "Overweight"     else:
        data['bmi_range'] = "Obesity"

    bmi_list.append(data) bmi_list
```

```
[{ 'Height': 177.74, 'Weight': 74.74, 'bmi': 23.66, 'bmi_range': 'Nor
mal'},
 { 'Height': 149.72, 'Weight': 38.52, 'bmi': 17.18, 'bmi_range': 'Und
erweight'},
 { 'Height': 142.78, 'Weight': 45.87, 'bmi': 22.5, 'bmi_range': 'Norm
al'},
 {'Height': 160.85, 'Weight': 61.73, 'bmi': 23.86, 'bmi_range': 'Nor
mal'},
 { 'Height': 182.25, 'Weight': 105.28, 'bmi': 31.7, 'bmi_range': 'Obe
sity'},
 { 'Height': 162.7, 'Weight': 66.38, 'bmi': 25.08, 'bmi_range': 'Over
weight'},
 {'Height': 144.64, 'Weight': 39.99, 'bmi': 19.12, 'bmi_range': 'Nor
mal'},
 { 'Height': 168.9, 'Weight': 88.78, 'bmi': 31.12, 'bmi_range': 'Obes
ity'},
 { 'Height': 160.26, 'Weight': 59.08, 'bmi': 23.0, 'bmi_range': 'Norm
al'},
 { 'Height': 178.83, 'Weight': 96.13, 'bmi': 30.06, 'bmi_range': 'Obe
sity'},
 {'Height':  156.77, 'Weight': 71.36, 'bmi': 29.04, 'bmi_range': 'Ove
rweight'},
 { 'Height': 183.84, 'Weight': 107.91, 'bmi': 31.93, 'bmi_range': 'Ob
esity'},
 { 'Height': 157.06, 'Weight': 86.79, 'bmi': 35.18, 'bmi_range': 'Obe
sity'},
 {'Height': 174.6, 'Weight': 87.67, 'bmi': 28.76, 'bmi_range': 'Over
weight'},
 { 'Height': 140.88, 'Weight': 30.81, 'bmi': 15.52, 'bmi_range': 'Und
erweight'},
 { 'Height': 145.31, 'Weight': 60.3, 'bmi': 28.56, 'bmi_range': 'Over
weight'},
 {'Height': 188.38, 'Weight': 100.5, 'bmi': 28.32, 'bmi_range': 'Ove
rweight'},
 { 'Height': 151.3, 'Weight': 88.26, 'bmi': 38.56, 'bmi_range': 'Obes
ity'},
 { 'Height': 186.64, 'Weight': 108.21, 'bmi': 31.06, 'bmi_range': 'Ob
esity'},
```

```
 { 'Height': 142.15, 'Weight': 48.59, 'bmi': 24.05, 'bmi_range': 'Nor
mal'},
 {'Height': 165.95, 'Weight': 84.69, 'bmi': 30.75, 'bmi_range': 'Obe
sity'},
 { 'Height': 147.85, 'Weight': 47.47, 'bmi': 21.72, 'bmi_range': 'Nor
mal'},
 { 'Height': 149.1, 'Weight': 42.17, 'bmi': 18.97, 'bmi_range': 'Norm
al'},
 {'Height': 161.28, 'Weight': 75.96, 'bmi': 29.2, 'bmi_range': 'Over
weight'},
 { 'Height': 172.25, 'Weight': 78.79, 'bmi': 26.56, 'bmi_range': 'Ove
rweight'},
 { 'Height': 188.33, 'Weight': 103.32, 'bmi': 29.13, 'bmi_range': 'Ov
erweight'},
 {'Height': 142.15, 'Weight': 33.14, 'bmi': 16.4, 'bmi_range': 'Unde
rweight'},
 { 'Height': 157.32, 'Weight': 77.27, 'bmi': 31.22, 'bmi_range': 'Obe
sity'},
 { 'Height': 183.72, 'Weight': 107.5, 'bmi': 31.85, 'bmi_range': 'Obe
sity'},
 { 'Height': 188.37, 'Weight': 109.03, 'bmi': 30.73, 'bmi_range': 'Ob
esity'},
 {'Height': 142.04, 'Weight': 36.43, 'bmi': 18.06, 'bmi_range': 'Und
erweight'},
 { 'Height': 163.11, 'Weight': 47.55, 'bmi': 17.87, 'bmi_range': 'Und
erweight'},
 { 'Height': 178.58, 'Weight': 94.03, 'bmi': 29.48, 'bmi_range': 'Ove
rweight'},
 {'Height': 187.69, 'Weight': 103.71, 'bmi': 29.44, 'bmi_range': 'Ov
erweight'},
 { 'Height': 177.59, 'Weight': 97.07, 'bmi': 30.78, 'bmi_range': 'Obe
sity'},
 { 'Height': 149.3, 'Weight': 31.66, 'bmi': 14.2, 'bmi_range': 'Under
weight'},
 { 'Height': 181.16, 'Weight': 100.76, 'bmi': 30.7, 'bmi_range': 'Obe
sity'},
 { 'Height': 156.44, 'Weight': 63.03, 'bmi': 25.75, 'bmi_range': 'Ove
rweight'},
 { 'Height': 147.74, 'Weight': 55.13, 'bmi': 25.26, 'bmi_range': 'Ove
rweight'},
 {'Height': 140.18, 'Weight': 66.05, 'bmi': 33.61, 'bmi_range': 'Obe
sity'},
 { 'Height': 165.88, 'Weight': 46.86, 'bmi': 17.03, 'bmi_range': 'Und
erweight'},
 { 'Height': 189.28, 'Weight': 103.25, 'bmi': 28.82, 'bmi_range': 'Ov
erweight'},
 {'Height': 161.28, 'Weight': 76.26, 'bmi': 29.32 , 'bmi_range': 'Ove
rweight'},
```

```
 { 'Height': 151.5, 'Weight': 44.34, 'bmi': 19.32, 'bmi_range': 'Norm
al'},
 { 'Height': 174.82, 'Weight': 73.65, 'bmi': 24.1, 'bmi_range': 'Norm
al'},
 { 'Height': 186.16, 'Weight': 102.26, 'bmi': 29.51, 'bmi_range': 'Ov
erweight'},
 { 'Height': 162.18, 'Weight': 51.91, 'bmi': 19.74, 'bmi_range': 'Nor
mal'},
 { 'Height': 176.68, 'Weight': 83.43, 'bmi': 26.73, 'bmi_range': 'Ove
rweight'},
 { 'Height': 142.95, 'Weight': 58.19, 'bmi': 28.48, 'bmi_range': 'Ove
rweight'},
 {'Height': 182.6 , 'Weight': 106.21, 'bmi': 31.85, 'bmi_range': 'Obe
sity'},
 { 'Height': 160.23, 'Weight': 48.43, 'bmi': 18.86, 'bmi_range': 'Nor
mal'},
 { 'Height': 185.34, 'Weight': 108.05, 'bmi': 31.45, 'bmi_range': 'Ob
esity'},
 {'Height': 160.32, 'Weight': 48.56, 'bmi': 18.89 , 'bmi_range': 'Nor
mal'},
 { 'Height': 146.95, 'Weight': 53.47, 'bmi': 24.76, 'bmi_range': 'Nor
mal'},
 { 'Height': 168.73, 'Weight': 71.53, 'bmi': 25.12, 'bmi_range': 'Ove
rweight'},
 { 'Height': 180.56, 'Weight': 102.16, 'bmi': 31.34, 'bmi_range': 'Ob
esity'},
 { 'Height': 141.6, 'Weight': 42.59, 'bmi': 21.24, 'bmi_range': 'Norm
al'},
 { 'Height': 172.69, 'Weight': 82.8, 'bmi': 27.76, 'bmi_range': 'Over
weight'},
 { 'Height': 180.51, 'Weight': 101.28, 'bmi': 31.08, 'bmi_range': 'Ob
esity'},
 {'Height': 180.67, 'Weight': 104.01, 'bmi': 31.86, 'bmi_range': 'Ob
esity'},
 { 'Height': 176.95, 'Weight': 84.56, 'bmi': 27.01, 'bmi_range': 'Ove
rweight'},
 { 'Height': 153.83, 'Weight': 66.62, 'bmi': 28.15, 'bmi_range': 'Ove
rweight'},
 {'Height': 147.89, 'Weight': 63.14, 'bmi': 28.87, 'bmi_range': 'Ove
rweight'},
 { 'Height': 148.66, 'Weight': 49.7, 'bmi': 22.49, 'bmi_range': 'Norm
al'},
 { 'Height': 146.0, 'Weight': 66.82, 'bmi': 31.35, 'bmi_range': 'Obes
ity'},
 { 'Height': 143.83, 'Weight': 38.59, 'bmi': 18.65, 'bmi_range': 'Nor
mal'},
 { 'Height': 186.93, 'Weight': 101.66, 'bmi': 29.09, 'bmi_range': 'Ov
erweight'},
```

```
 { 'Height': 154.68, 'Weight': 48.21, 'bmi': 20.15, 'bmi_range': 'Nor
mal'},
 { 'Height': 183.74, 'Weight': 103.53, 'bmi': 30.67, 'bmi_range': 'Ob
esity'},
 {'Height': 181.8, 'Weight': 109.73, 'bmi': 33.2, 'bmi_range': 'Obes
ity'},
 { 'Height': 164.63, 'Weight': 46.16, 'bmi': 17.03, 'bmi_range': 'Und
erweight'},
 { 'Height': 145.08, 'Weight': 56.93, 'bmi': 27.05, 'bmi_range': 'Ove
rweight'},
 {'Height': 161.32, 'Weight': 88.54, 'bmi': 34.02, 'bmi_range': 'Obe
sity'},
 { 'Height': 169.56, 'Weight': 61.58, 'bmi': 21.42, 'bmi_range': 'Nor
mal'},
 { 'Height': 161.63, 'Weight': 89.82, 'bmi': 34.38, 'bmi_range': 'Obe
sity'},
 { 'Height': 161.48, 'Weight': 53.45, 'bmi': 20.5, 'bmi_range': 'Norm
al'},
 { 'Height': 181.83, 'Weight': 104.17, 'bmi': 31.51, 'bmi_range': 'Ob
esity'},
 { 'Height': 151.4, 'Weight': 86.45, 'bmi': 37.71, 'bmi_range': 'Obes
ity'},
 { 'Height': 171.89, 'Weight': 94.29, 'bmi': 31.91, 'bmi_range': 'Obe
sity'},
 {'Height': 163.1, 'Weight': 73.27, 'bmi': 27.54, 'bmi_range': 'Over
weight'},
 { 'Height': 180.09, 'Weight': 107.72, 'bmi': 33.21, 'bmi_range': 'Ob
esity'},
 { 'Height': 184.01, 'Weight': 106.94, 'bmi': 31.58, 'bmi_range': 'Ob
esity'},
 {'Height': 186.41, 'Weight': 103.6, 'bmi': 29.81 , 'bmi_range': 'Ove
rweight'},
 { 'Height': 143.29, 'Weight': 41.35, 'bmi': 20.14, 'bmi_range': 'Nor
mal'},
 { 'Height': 161.27, 'Weight': 70.62, 'bmi': 27.15, 'bmi_range': 'Ove
rweight'},
 { 'Height': 174.15, 'Weight': 87.37, 'bmi': 28.81, 'bmi_range': 'Ove
rweight'},
 { 'Height': 183.4, 'Weight': 105.93, 'bmi': 31.49, 'bmi_range': 'Obe
sity'},
 { 'Height': 160.48, 'Weight': 86.05, 'bmi': 33.41, 'bmi_range': 'Obe
sity'},
 {'Height': 151.34, 'Weight': 47.89, 'bmi': 20.91, 'bmi_range': 'Nor
mal'},
 { 'Height': 188.05, 'Weight': 107.77, 'bmi': 30.48, 'bmi_range': 'Ob
esity'},
 { 'Height': 147.5, 'Weight': 53.04, 'bmi': 24.38, 'bmi_range': 'Norm
al'},
```

```
 {'Height': 161.01, 'Weight': 43.75, 'bmi': 16.88, 'bmi_range': 'Und
erweight'},
 { 'Height': 185.3, 'Weight': 107.99, 'bmi': 31.45, 'bmi_range': 'Obe
sity'},
 { 'Height': 174.06, 'Weight': 81.28, 'bmi': 26.83, 'bmi_range': 'Ove
rweight'},
 { 'Height': 150.22, 'Weight': 79.78, 'bmi': 35.35, 'bmi_range': 'Obe
sity'},
 { 'Height': 147.82, 'Weight': 45.29, 'bmi': 20.73, 'bmi_range': 'Nor
mal'},
 { 'Height': 180.27, 'Weight': 103.48, 'bmi': 31.84, 'bmi_range': 'Ob
esity'},
 { 'Height': 183.51, 'Weight': 107.12, 'bmi': 31.81, 'bmi_range': 'Ob
esity'},
 {'Height': 164.39, 'Weight':  85.51, 'bmi': 31.64, 'bmi_range': 'Obe
sity'},
 { 'Height': 179.51, 'Weight': 95.86, 'bmi': 29.75, 'bmi_range': 'Ove
rweight'}]
```

#Step 3: Create a panda dataframe with generated data

```
import pandas as pd

df = pd.DataFrame(bmi_list) df
```

```
    Height  Weight    bmi    bmi_range
0   177.74   74.74  23.66       Normal
1   149.72   38.52  17.18  Underweight
2   142.78   45.87  22.50       Normal
3   160.85   61.73  23.86       Normal
4   182.25  105.28  31.70      Obesity ..     ...     ...    ...
    ...
95  147.82   45.29  20.73       Normal
96  180.27  103.48  31.84      Obesity
97  183.51  107.12  31.81      Obesity
98  164.39   85.51  31.64      Obesity
99  179.51   95.86  29.75   Overweight

[100 rows x 4 columns]
```

#Step 4: Calculate minimum Macro intake data for all users We use a online macro calculator https://www.calculator.net/macro-calculator.html for this task. We mine the website for the minimum macro intake data of different users by giving their height and weight as parameters. We use a python package called BeautifulSoup for accomplishing this task.

```
import requests from bs4
import BeautifulSoup
 macros_data = []
```

```
 for height, weight in zip(df['Height'], df['Weight']):
  # Input variables
  #height = 170 # height in cm
  #weight = 70 # weight in kg
   url = "https://www.calculator.net/macro-calculator.html"
   params = {
      "ctype": "metric",
      "cage": 25,
      "csex": "m",
      "cheightmeter": height,
      "ckg": weight,
      "cactivity": 1.375,
      "printit": 0,
      "x": 121,
      "y": 28
  }
   response = requests.get(url, params=params)

  # Create a Beautiful Soup object
  soup = BeautifulSoup(response.content, 'html.parser')

  # Find the recommended daily calorie intake
macros = dict()

  labels = soup.find_all('td', {'class': 'arrow_box'})
values = soup.find_all('td', {'class': 'result_box'})   for
label, value in zip(labels, values):
    temp_lab = label.div.text.strip()     temp_val =
value.text.replace('<', '').replace(':', '') .strip (). split(" ")
    macros[temp_lab+" ("+temp_val[1]+")"] = temp_val[0]
print(macros)   macros_data.append(macros)
```

```
{ 'Protein (grams/day)': '146', 'Carbs (grams/day)': '319', 'Fat (gra
ms/day)': '68', 'Sugar (grams/day)': '64', 'Saturated Fat (grams/day
)': '27', 'Food Energy (Calories/dayor)': '2,390'}
{ 'Protein (grams/day)': '101', 'Carbs (grams/day)': '220', 'Fat (gra
ms/day)': '47', 'Sugar (grams/day)': '44', 'Saturated Fat (grams/day )':
'19', 'Food Energy (Calories/dayor)': '1,651'}
{ 'Protein (grams/day)': '103', 'Carbs (grams/day)': '226', 'Fat (gra
ms/day)': '48', 'Sugar (grams/day)': '45', 'Saturated Fat (grams/day
)': '19', 'Food Energy (Calories/dayor)': '1,693'}
{ 'Protein (grams/day)': '126', 'Carbs (grams/day)': '275', 'Fat (gra
ms/day)': '59', 'Sugar (grams/day)': '55', 'Saturated Fat (grams/day
)': '23', 'Food Energy (Calories/dayor)': '2,066'}
{'Protein (grams/day) ': '174', 'Carbs (grams/day)': '380', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
```

)': '32', 'Food Energy (Calories/dayor)': '2,849'}
{ 'Protein (grams/day)': '131', 'Carbs (grams/day)': '286', 'Fat (gra
ms/day)': '61', 'Sugar (grams/day)': '57', 'Saturated Fat (grams/day
)': '24', 'Food Energy (Calories/dayor)': '2,146'}
{ 'Protein (grams/day)': '99', 'Carbs (grams/day)': '217', 'Fat (gram
s/day)': '46', 'Sugar (grams/day)': '43', 'Saturated Fat (grams/day)
': '19', 'Food Energy (Calories/dayor)': '1,628'}
{ 'Protein (grams/day)': '153', 'Carbs (grams/day)': '334', 'Fat (gra
ms/day)': '71', 'Sugar (grams/day)': '67', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,507'}
{'Protein (grams/day)': '123', 'Carbs (grams/day)': '270', 'Fat (gra
ms/day)': '58', 'Sugar (grams/day)': '54', 'Saturated Fat (grams/day
)': '23', 'Food Energy (Calories/dayor)': '2,025'}
{ 'Protein (grams/day)': '164', 'Carbs (grams/day)': '359', 'Fat (gra
ms/day)': '77', 'Sugar (grams/day)': '72', 'Saturated Fat (grams/day
)': '31', 'Food Energy (Calories/dayor)': '2,694'}
{ 'Protein (grams/day)': '132', 'Carbs (grams/day)': '288', 'Fat (gra
ms/day)': '61', 'Sugar (grams/day)': '58', 'Saturated Fat (grams/day
)': '25', 'Food Energy (Calories/dayor)': '2,163'}
{ 'Protein (grams/day)': '177', 'Carbs (grams/day)': '387', 'Fat (gra
ms/day)': '82', 'Sugar (grams/day)': '77', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,899'}
{ 'Protein (grams/day)': '145', 'Carbs (grams/day)': '317', 'Fat (gra
ms/day)': '68', 'Sugar (grams/day)': '63', 'Saturated Fat (grams/day
)': '27', 'Food Energy (Calories/dayor)': '2,378'}
{'Protein (grams/day)': '155', 'Carbs (grams/day)': '339', 'Fat (gra
ms/day)': '72', 'Sugar (grams/day)': '68', 'Saturated Fat (grams/day
)': '29', 'Food Energy (Calories/dayor)': '2,541'}
{}
{ 'Protein (grams/day)': '117', 'Carbs (grams/day)': '255', 'Fat (gra
ms/day)': '54', 'Sugar (grams/day) ': '51', 'Saturated Fat (grams/day
)': '22', 'Food Energy (Calories/dayor)': '1,913'}
{ 'Protein (grams/day)': '173', 'Carbs (grams/day)': '378', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,836'}
{ 'Protein (grams/day)': '143', 'Carbs (grams/day)': '313', 'Fat (gra
ms/day)': '67', 'Sugar (grams/day)': '63', 'Saturated Fat (grams/day
)': '27', 'Food Energy (Calories/dayor)': '2,349'}
{'Protein (grams/day)': '178', 'Carbs (grams/day)': '390', 'Fat (gra
ms/day)': '83', 'Sugar (grams/day)': '78', 'Saturated Fat (grams/day)':
'33', 'Food Energy (Calories/dayor)': '2,927'}
{ 'Protein (grams/day)': '105', 'Carbs (grams/day)': '230', 'Fat (gra
ms/day)': '49', 'Sugar (grams/day)': '46', 'Saturated Fat (grams/day
)': '20', 'Food Energy (Calories/dayor)': '1,725'}
{ 'Protein (grams/day)': '148', 'Carbs (grams/day)': '323', 'Fat (gra
ms/day)': '69', 'Sugar (grams/day)': '65', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,426'}

{'Protein (grams/day)': '107', 'Carbs (grams/day)': '234', 'Fat (gra
ms/day)': '50', 'Sugar (grams/day)': '47', 'Saturated Fat (grams/day
)': '20', 'Food Energy (Calories/dayor)': '1,758'}
{ 'Protein (grams/day)': '103', 'Carbs (grams/day)': '226', 'Fat (gra
ms/day)': '48', 'Sugar (grams/day)': '45', 'Saturated Fat (grams/day
)': '19', 'Food Energy (Calories/dayor)': '1,696'}
{ 'Protein (grams/day)': '138', 'Carbs (grams/day)': '302', 'Fat (gra
ms/day)': '64', 'Sugar (grams/day)': '60', 'Saturated Fat (grams/day
)': '26', 'Food Energy (Calories/dayor)': '2,265'}
{ 'Protein (grams/day)': '146', 'Carbs (grams/day)': '320', 'Fat (gra
ms/day)': '68', 'Sugar (grams/day)': '64', 'Saturated Fat (grams/day
)': '27', 'Food Energy (Calories/dayor)': '2,399'}
{'Protein (grams/day)': '175', 'Carbs (grams/day)': '383', 'Fat (gra
ms/day)': '82', 'Sugar (grams/day)': '77', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,874'}
{ 'Protein (grams/day)': '92', 'Carbs (grams/day)': '202', 'Fat (gram
s/day)': '43', 'Sugar (grams/day)': '40', 'Saturated Fat (grams/day)
': '17', 'Food Energy (Calories/dayor)': '1,512'}
{ 'Protein (grams/day)': '137', 'Carbs (grams/day)': '300', 'Fat (gra
ms/day)': '64', 'Sugar (grams/day)': '60', 'Saturated Fat (grams/day
)': '26', 'Food Energy (Calories/dayor)': '2,249'}
{ 'Protein (grams/day)': '176', 'Carbs (grams/day)': '386', 'Fat (gra
ms/day)': '82', 'Sugar (grams/day)': '77', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,892'}
{'Protein (grams/day)': '180', 'Carbs (grams/day)': '394', 'Fat (gra
ms/day)': '84', 'Sugar (grams/day)': '79', 'Saturated Fat (grams/day
)': '34', 'Food Energy (Calories/dayor)': '2,953'}
{ 'Protein (grams/day)': '95', 'Carbs (grams/day)': '208', 'Fat (gram
s/day)': '44', 'Sugar (grams/day)': '42', 'Saturated Fat (grams/day)
': '18', 'Food Energy (Calories/dayor)': '1,557'}
{ 'Protein (grams/day)': '115', 'Carbs (grams/day)': '252', 'Fat (gra
ms/day)': '54', 'Sugar (grams/day)': '50', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,891'}
{ 'Protein (grams/day)': '162', 'Carbs (grams/day)': '355', 'Fat (gra
ms/day)': '76', 'Sugar (grams/day)': '71', 'Saturated Fat (grams/day
)': '30', 'Food Energy (Calories/dayor)': '2,663'}
{'Protein (grams/day)': '175', 'Carbs (grams/day)': '383', 'Fat (gra
ms/day)': '82', 'Sugar (grams/day)': '77', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,874'}
{ 'Protein (grams/day)': '164', 'Carbs (grams/day)': '359', 'Fat (gra
ms/day)': '77', 'Sugar (grams/day)': '72', 'Saturated Fat (grams/day
)': '31', 'Food Energy (Calories/dayor)': '2,696'}
{ 'Protein (grams/day)': '95', 'Carbs (grams/day)': '207', 'Fat (gram
s/day)': '44', 'Sugar (grams/day)': '41', 'Saturated Fat (grams/day)
': '18', 'Food Energy (Calories/dayor)': '1,553'}
{'Protein (grams/day)': '169', 'Carbs (grams/day)': '370', 'Fat (gra
ms/day)': '79', 'Sugar (grams/day)': '74', 'Saturated Fat (grams/day

)': '32', 'Food Energy (Calories/dayor)': '2,777'}
{ 'Protein (grams/day)': '125', 'Carbs (grams/day)': '273', 'Fat (gra
ms/day)': '58', 'Sugar (grams/day)': '55', 'Saturated Fat (grams/day
)': '23', 'Food Energy (Calories/dayor)': '2,046'}
{ 'Protein (grams/day)': '114', 'Carbs (grams/day)': '248', 'Fat (gra
ms/day)': '53', 'Sugar (grams/day)': '50', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,863'}
{ 'Protein (grams/day)': '119', 'Carbs (grams/day)': '260', 'Fat (gra
ms/day)': '55', 'Sugar (grams/day)': '52', 'Saturated Fat (grams/day
)': '22', 'Food Energy (Calories/dayor)': '1,948'}
{'Protein (grams/day)': '116', 'Carbs (grams/day)': '254', 'Fat (gra
ms/day)': '54', 'Sugar (grams/day)': '51', 'Saturated Fat (grams/day
)': '22', 'Food Energy (Calories/dayor)': '1,905'}
{ 'Protein (grams/day)': '176', 'Carbs (grams/day)': '384', 'Fat (gra
ms/day)': '82', 'Sugar (grams/day)': '77 ', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,881'}
{ 'Protein (grams/day)': '138', 'Carbs (grams/day)': '303', 'Fat (gra
ms/day)': '64', 'Sugar (grams/day)': '61', 'Saturated Fat (grams/day
)': '26', 'Food Energy (Calories/dayor)': '2,270'}
{ 'Protein (grams/day)': '107', 'Carbs (grams/day)': '233', 'Fat (gra
ms/day)': '50', 'Sugar (grams/day)': '47', 'Saturated Fat (grams/day
)': '20', 'Food Energy (Calories/dayor)': '1,747'}
{'Protein (grams/day)': '143', 'Carbs (grams/day)': '313', 'Fat (gra
ms/day)': '67', 'Sugar (grams/day)': '63', 'Saturated Fat (grams/day
)': '27', 'Food Energy (Calories/dayor)': '2,350'}
{ 'Protein (grams/day)': '173', 'Carbs (grams/day)': '379', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,841'}
{ 'Protein (grams/day)': '118', 'Carbs (grams/day)': '259', 'Fat (gra
ms/day)': '55', 'Sugar (grams/day)': '52', 'Saturated Fat (grams/day
)': '22', 'Food Energy (Calories/dayor)': '1,942'}
{'Protein (grams/day)': '153', 'Carbs (grams/day)': '333', 'Fat (gra
ms/day)': '71', 'Sugar (grams/day)': '67', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,501'}
{ 'Protein (grams/day)': '114', 'Carbs (grams/day)': '249', 'Fat (gra
ms/day)': '53', 'Sugar (grams/day)': '50', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,864'}
{ 'Protein (grams/day)': '175', 'Carbs (grams/day)': '382', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,865'}
{ 'Protein (grams/day)': '115', 'Carbs (grams/day)': '250', 'Fat (gra
ms/day)': '53', 'Sugar (grams/day)': '50', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,878'}
{'Protein (grams/day)': '178', 'Carbs (grams/day)': '388', 'Fat (gra
ms/day)': '83', 'Sugar (grams/day)': '78', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,913'}

{ 'Protein (grams/day)': '115', 'Carbs (grams/day)': '251', 'Fat (gra
ms/day)': '53', 'Sugar (grams/day)': '50', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,880'}
{ 'Protein (grams/day)': '112', 'Carbs (grams/day)': '244', 'Fat (gra
ms/day)': '52', 'Sugar (grams/day)': '49', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,833'}
{ 'Protein (grams/day)': '138', 'Carbs (grams/day)': '303', 'Fat (gra
ms/day)': '64', 'Sugar (grams/day)': '61', 'Saturated Fat (grams/day
)': '26', 'Food Energy (Calories/dayor)': '2,269'}
{'Protein (grams/day)': '170', 'Carbs (grams/day) ': '372', 'Fat (gra
ms/day)': '79', 'Sugar (grams/day)': '74', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,791'}
{ 'Protein (grams/day)': '100', 'Carbs (grams/day)': '218', 'Fat (gra
ms/day)': '47', 'Sugar (grams/day)': '44', 'Saturated Fat (grams/day
)': '19', 'Food Energy (Calories/dayor)': '1,637'}
{ 'Protein (grams/day)': '150', 'Carbs (grams/day)': '328', 'Fat (gra
ms/day)': '70', 'Sugar (grams/day)': '66', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,458'}
{ 'Protein (grams/day)': '169', 'Carbs (grams/day)': '371', 'Fat (gra
ms/day)': '79', 'Sugar (grams/day)': '74', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,779'}
{ 'Protein (grams/day)': '172', 'Carbs (grams/day)': '376', 'Fat (gra
ms/day)': '80', 'Sugar (grams/day)': '75', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,818'}
{'Protein (grams/day)': '154', 'Carbs (grams/day)': '336', 'Fat (gra
ms/day)': '72', 'Sugar (grams/day)': '67', 'Saturated Fat (grams/day
)': '29', 'Food Energy (Calories/dayor)': '2,518'}
{ 'Protein (grams/day)': '126', 'Carbs (grams/day)': '276', 'Fat (gra
ms/day)': '59 ', 'Sugar (grams/day)': '55', 'Saturated Fat (grams/day
)': '24', 'Food Energy (Calories/dayor)': '2,073'}
{ 'Protein (grams/day)': '120', 'Carbs (grams/day)': '263', 'Fat (gra
ms/day)': '56', 'Sugar (grams/day)': '53', 'Saturated Fat (grams/day
)': '22', 'Food Energy (Calories/dayor)': '1,974'}
{ 'Protein (grams/day)': '110', 'Carbs (grams/day)': '239', 'Fat (gra
ms/day)': '51', 'Sugar (grams/day)': '48', 'Saturated Fat (grams/day
)': '20', 'Food Energy (Calories/dayor)': '1,796'}
{'Protein (grams/day)': '122', 'Carbs (grams/day)': '268', 'Fat (gra
ms/day)': '57', 'Sugar (grams/day)': '54', 'Saturated Fat (grams/day
)': '23', 'Food Energy (Calories/dayor)': '2,008'}
{ 'Protein (grams/day)': '98', 'Carbs (grams/day)': '214', 'Fat (gram
s/day)': '46', 'Sugar (grams/day)': '43', 'Saturated Fat (grams/day)
': '18', 'Food Energy (Calories/dayor)': '1,602'}
{ 'Protein (grams/day)': '173', 'Carbs (grams/day)': '379', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,839'}

{ 'Protein (grams/day)': '111', 'Carbs (grams/day)': '244', 'Fat (gra
ms/day)': '52', 'Sugar (grams/day)': '49', 'Saturated Fat (grams/day)':
'21', 'Food Energy (Calories/dayor)': '1,827'}
{'Protein (grams/day)': '173', 'Carbs (grams/day)': '378', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,838'}
{ 'Protein (grams/day)': '177', 'Carbs (grams/day)': '387', 'Fat (gra
ms/day)': '83', 'Sugar (grams/day)': '77', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,906'}
{ 'Protein (grams/day)': '115', 'Carbs (grams/day)': '251', 'Fat (gra
ms/day)': '54', 'Sugar (grams/day)': '50', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,884'}
{'Protein (grams/day)': '114', 'Carbs (grams/day)': '249', 'Fat (gra
ms/day)': '53', 'Sugar (grams/day)': '50', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,865'}
{ 'Protein (grams/day)': '149', 'Carbs (grams/day)': '325', 'Fat (gra
ms/day)': '69', 'Sugar (grams/day)': '65', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,439'}
{ 'Protein (grams/day)': '130', 'Carbs (grams/day)': '285', 'Fat (gra
ms/day)': '61', 'Sugar (grams/day)': '57', 'Saturated Fat (grams/day
)': '24', 'Food Energy (Calories/dayor)': '2,139'}
{ 'Protein (grams/day)': '150', 'Carbs (grams/day)': '328', 'Fat (gra
ms/day)': '70', 'Sugar (grams/day)': '66', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,459'}
{'Protein (grams/day)': '119', 'Carbs (grams/day)': '261', 'Fat (gra
ms/day)': '56', 'Sugar (grams/day)': '52', 'Saturated Fat (grams/day
)': '22', 'Food Energy (Calories/dayor)': '1,958'}
{ 'Protein (grams/day)': '173', 'Carbs (grams/day)': '377', 'Fat (gra
ms/day)': '80', 'Sugar (grams/day)': '75', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,830'}
{ 'Protein (grams/day)': '142', 'Carbs (grams/day)': '310', 'Fat (gra
ms/day)': '66', 'Sugar (grams/day)': '62', 'Saturated Fat (grams/day
)': '26', 'Food Energy (Calories/dayor)': '2,325'}
{ 'Protein (grams/day)': '159', 'Carbs (grams/day)': '348', 'Fat (gra
ms/day)': '74', 'Sugar (grams/day)': '70', 'Saturated Fat (grams/day
)': '30', 'Food Energy (Calories/dayor)': '2,609'}
{'Protein (grams/day)': '137', 'Carbs (grams/day)': '299', 'Fat (gra
ms/day)': '64', 'Sugar (grams/day)': '60', 'Saturated Fat (grams/day
)': '26', 'Food Energy (Calories/dayor)': '2,244'}
{ 'Protein (grams/day)': '175', 'Carbs (grams/day)': '382', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,864'}
{ 'Protein (grams/day)': '176', 'Carbs (grams/day)': '385', 'Fat (gra
ms/day)': '82', 'Sugar (grams/day)': '77', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,887'}
{ 'Protein (grams/day)': '174', 'Carbs (grams/day)': '381', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day

)': '33', 'Food Energy (Calories/dayor)': '2,861'}
{'Protein (grams/day)': '100', 'Carbs (grams/day)': '218', 'Fat (gra
ms/day)': '46', 'Sugar (grams/day)': '44', 'Saturated Fat (grams/day
)': '19', 'Food Energy (Calories/dayor)': '1,635'}
{ 'Protein (grams/day)': '134', 'Carbs (grams/day)': '292', 'Fat (gra
ms/day)': '62', 'Sugar (grams/day)': '58', 'Saturated Fat (grams/day
)': '25', 'Food Energy (Calories/dayor)': '2,192'}
{ 'Protein (grams/day)': '154', 'Carbs (grams/day)': '338', 'Fat (gra
ms/day)': '72', 'Sugar (grams/day)': '68', 'Saturated Fat (grams/day
)': '29', 'Food Energy (Calories/dayor)': '2,533'}
{'Protein (grams/day)': '175', 'Carbs (grams/day)': '382', 'Fat (gra
ms/day)': '81', 'Sugar (grams/day)': '76', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,868'}
{ 'Protein (grams/day)': '146', 'Carbs (grams/day)': '320', 'Fat (gra
ms/day)': '68', 'Sugar (grams/day)': '64', 'Saturated Fat (grams/day
)': '27', 'Food Energy (Calories/dayor)': '2,397'}
{ 'Protein (grams/day)': '109', 'Carbs (grams/day)': '239', 'Fat (gra
ms/day)': '51', 'Sugar (grams/day)': '48', 'Saturated Fat (grams/day
)': '20', 'Food Energy (Calories/dayor)': '1,794'}
{ 'Protein (grams/day)': '179', 'Carbs (grams/day)': '391', 'Fat (gra
ms/day)': '83', 'Sugar (grams/day)': '78', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,933'}
{'Protein (grams/day)': '112', 'Carbs (grams/day)': '244', 'Fat (gra
ms/day)': '52', 'Sugar (grams/day)': '49', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,832'}
{ 'Protein (grams/day)': '111', 'Carbs (grams/day)': '243', 'Fat (gra
ms/day)': '52', 'Sugar (grams/day)': '49', 'Saturated Fat (grams/day
)': '21', 'Food Energy (Calories/dayor)': '1,820'}
{ 'Protein (grams/day)': '178', 'Carbs (grams/day)': '388', 'Fat (gra
ms/day)': '83', 'Sugar (grams/day)': '78', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,912'}
{ 'Protein (grams/day)': '149', 'Carbs (grams/day)': '326', 'Fat (gra
ms/day)': '70', 'Sugar (grams/day)': '65', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,448'}
{'Protein (grams/day)': '136', 'Carbs (grams/day)': '296 ', 'Fat (gra
ms/day)': '63', 'Sugar (grams/day)': '59', 'Saturated Fat (grams/day
)': '25', 'Food Energy (Calories/dayor)': '2,223'}
{ 'Protein (grams/day)': '105', 'Carbs (grams/day)': '230', 'Fat (gra
ms/day)': '49', 'Sugar (grams/day)': '46', 'Saturated Fat (grams/day
)': '20', 'Food Energy (Calories/dayor)': '1,728'}
{ 'Protein (grams/day)': '171', 'Carbs (grams/day)': '374', 'Fat (gra
ms/day)': '80', 'Sugar (grams/day)': '75', 'Saturated Fat (grams/day
)': '32', 'Food Energy (Calories/dayor)': '2,807'}
{'Protein (grams/day)': '176', 'Carbs (grams/day)': '385', 'Fat (gra
ms/day)': '82', 'Sugar (grams/day)': '77', 'Saturated Fat (grams/day
)': '33', 'Food Energy (Calories/dayor)': '2,885'}

```
{ 'Protein (grams/day)': '148', 'Carbs (grams/day)': '323', 'Fat (gra
ms/day)': '69', 'Sugar (grams/day)': '65', 'Saturated Fat (grams/day
)': '28', 'Food Energy (Calories/dayor)': '2,423'}
{ 'Protein (grams/day)': '164', 'Carbs (grams/day)': '359', 'Fat (gra
ms/day)': '77', 'Sugar (grams/day)': '72', 'Saturated Fat (grams/day
)': '31', 'Food Energy (Calories/dayor)': '2,696'}
```

##Create a dataframe from the mined data

```
macros_df = pd.DataFrame(macros_data) macros_df
```

```
   Protein (grams/day) Carbs (grams/day) Fat (grams/day) Sugar (gram
s/day)  \
0                  146              319              68
                   64
1                  101              220              47
    44
2                  103              226              48
                   45
3                  126              275              59
                   55
4                  174              380              81
    76
..                 ...              ...             ...
...
95                 105              230              49
                   46
96                 171              374              80
                   75
97                 176              385              82
    77
98                 148              323              69
                   65
99                 164              359              77
                   72

   Saturated Fat (grams/day) Food Energy (Calories/dayor)
0                         27                       2,390
1                         19                       1,651
2                         19                       1,693
3                         23                       2,066
4                         32                       2,849   ..
                          ...                         ...
95                        20                       1,728
96                        32                       2,807
97                        33                       2,885
98                        28                       2,423
99                        31                       2,696
```

```
[100 rows x 6 columns]
```

#Step 5: Merge both the dataframes We join both the dataframes and drop all the NaN rows.

Note: We drop all the NaN rows to remove any abnormal height and weight that may have been generated before.

```
final_df = df.join(macros_df)
final_df = final_df.dropna() final_df
```

```
    Height  Weight    bmi    bmi_range Protein (grams/day) Carbs (gr
ams/day)  \
0   177.74   74.74  23.66       Normal                146
    319
1   149.72   38.52  17.18  Underweight                101
    220
2   142.78   45.87  22.50       Normal                103
     226
3   160.85   61.73  23.86       Normal                126
     275
4   182.25  105.28  31.70      Obesity                174
     380
..     ...     ...    ...          ...                ...
...
95  147.82   45.29  20.73       Normal                105
     230
96  180.27  103.48  31.84      Obesity                171
    374
97  183.51  107.12  31.81      Obesity                176
    385
98  164.39   85.51  31.64      Obesity                148
     323
99  179.51   95.86  29.75   Overweight                164
    359

   Fat (grams/day) Sugar (grams/day) Saturated Fat (grams/day)  \
0               68                64                        27
1               47                44                        19
2               48                45                        19
3               59                55                        23
4               81                76                        32    ..
                ...               ...                       ...
95              49                46                        20
96              80                75                        32
97              82                77                        33
98              69                65                        28    99
                77                72                        31
```

```
   Food Energy (Calories/dayor)
0                         2,390
1                         1,651
2                         1,693
3                         2,066
4                         2,849   ..                        ...
95                        1,728
96                        2,807
97                        2,885
98                        2,423
99                        2,696

[100 rows x 10 columns]
```

#Step 6: Export dataset

```
# storing into the excel file
final_df.to_excel("Min_macros_for_height_and_weight.xlsx")
```

Diet recommendation https://github.com/akash-r34/CalCheck/blob/main/Diet_recommendation.ipynb

#Step 1: Import User macros dataset and diet plan dataset

```
user_macros_df = pd.read_excel("/content/Min_macros_for_height_and_weight.xlsx")
diet_id_df = pd.read_excel("/content/Food_data_generated_with_dietIDs.xlsx", index_col=0) diet_id_df
```

```
     dietID                                  Food name  Protein(g)  \ 0
1005                                        Idli        6.36
1      1003                                 Dosa, plain        5.70
2      1005                       Dosa, with filling        5.46
3      1005                              Bread, puri        6.84
4      1001                                    Vada       12.80    ..
       ...                                      ...         ...
140    1004                             Lentil curry        3.67
141    1005                Ghee, clarified butter        0.28
142    1002                                    Date        2.45
143    1006  Mayonnaise, reduced fat,  with olive oil        0.37    144
       1006                  Romaine lettuce, raw        1.36
```

```
     Total lipid (fat)(g)  Carbohydrate, by difference(g)  Energy(kc
al)  \
0                    0.35                           25.00
128
1                    4.05                           37.00
210
2                    4.27                           30.80           184
3                    24.90                           39.20           409
4                    9.73                           33.20           266
..                    ...                             ...           ...
140                  5.61                          12.30           110
141                  99.50                           0.00
876
142                  0.39                          75.00           282
143                  40.00                           0.00           361
144                  0.15                           2.87            15

     Alcohol, ethyl(g)  Water(g)  Caffeine(mg)  Theobromine(mg)  ...
\
0                     0     67.20             0                0  ...
1                     0     51.80              0                0
                     ...
2                     0     57.90             0                0  ...

3                     0     28.00             0                0  ...
4                     0     42.00             0                0  ...
..                  ...       ...           ...              ...  ...

140                   0     76.70             0                0  ...

141                   0      0.24             0                0  ...
142                   0     20.50             0                0
                     ...
143                   0     57.60             0                0  ...

144                   0     95.00             0                0  ...

     PUFA 20:4(g)  PUFA 22:6 n-3  (DHA)(g)  MUFA 16:1(g)  PUFA 18:4(g
)  \
0           0.000                      0.0         0.001            0.
0
1           0.000                      0.0         0.014            0.
0
2           0.000                      0.0         0.014            0. 0
3           0.001                      0.0          0.083            0. 0
```

```
4          0.000                    0.0         0.034            0. 0
..           ...                    ...           ...           ..
.
140        0.000                    0.0         0.018            0.
0
141        0.000                    0.0         2.230            0.
0
142        0.000                    0.0         0.001            0.
0
143        0.000                    0.0         0.464            0. 0
144        0.000                    0.0          0.002            0. 0

     MUFA 20:1(g)  PUFA 20:5 n-3 (EPA)(g)  MUFA 22:1(g)  \
0          0.002                    0.0         0.000
1          0.027                    0.0         0.003
2          0.026                    0.0         0.002
3          0.167                    0.0         0.000
4          0.066                    0.0         0.000    ..
           ...                    ...           ...
140        0.030                    0.0         0.000
141        0.000                    0.0         0.000
142        0.000                    0.0         0.000
143        0.125                    0.0         0.000
144        0.000                    0.0         0.000

     PUFA 22:5 n-3 (DPA)(g)  Fatty acids, total monounsaturated(g)  \
0                      0.0                                  0.090

1                      0.0                                  1.590
2                      0.0                                  1.610
3                      0.0                                  9.710

4                      0.0                                  3.810

..                     ...                                    ...
140                    0.0                                  1.890
141                    0.0                                 28.700

142                    0.0                                  0.036

143                    0.0                                 29.500
144                    0.0                                  0.006
                       Fatty acids, total polyunsaturated(g)
0                                  0.142
1                                  1.610
2                                  1.720
```

```
3                                       10.000
4                                       3.950   ..
                                        ...
140                                     1.990
141                                     3.690
142                                     0.019
143                                     4.010
144                                     0.082

[145 rows x 67 columns]
```

##Converting the string string values to integer values

```
user_macros_df = user_macros_df.dropna()
user_macros_df["Food Energy (Calories/day)"] = user_macros_df["Food
Energy (Calories/day)"].str.replace(',','') user_macros_df
```

```
<ipython-input-56-26743a48dbcc>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  user_macros_df["Food Energy (Calories/day)"] = user_macros_df["Foo
d Energy (Calories/day)"].str.replace(',','')

    Unnamed: 0  Height  Weight    bmi    bmi_range  Protein (grams/d
ay)  \
0            0  177.74   74.74  23.66       Normal                14
6.0
1            1  149.72   38.52  17.18  Underweight                10
             1.0
2            2  142.78   45.87   22.50       Normal                 10
             3.0
3            3  160.85   61.73  23.86       Normal                12
6.0
4            4  182.25  105.28  31.70      Obesity                17
             4.0
..         ...     ...     ...    ...          ...                 ...
95          95  147.82   45.29  20.73       Normal                10
5.0
96          96  180.27  103.48  31.84      Obesity                17
1.0
97          97  183.51  107.12  31.81      Obesity                17
             6.0
98          98   164.39   85.51  31.64      Obesity                 14
             8.0
```

```
99          99  179.51   95.86  29.75   Overweight                  16
            4.0

    Carbs (grams/day)  Fat (grams/day)  Sugar (grams/day)  \
0               319.0             68.0               64.0
1               220.0             47.0               44.0
2               226.0             48.0               45.0
3               275.0             59.0               55.0
4               380.0             81.0               76.0    ..
                ...              ...                ...
95              230.0             49.0               46.0
96              374.0             80.0               75.0
97              385.0             82.0               77.0
98              323.0             69.0               65.0
99              359.0             77.0               72.0

    Saturated Fat (grams/day) Food Energy (Calories/day)
0                        27.0                       2390
1                        19.0                       1651
2                        19.0                       1693
3                        23.0                       2066
4                        32.0                       2849   ..
                         ...                        ...
95                       20.0                       1728
96                       32.0                       2807
97                       33.0                       2885
98                       28.0                       2423
99                       31.0                       2696

[99 rows x 11 columns]
```

#Step 2: Aggregate each default diet plans by their macro levels

```
grouped_diet_id = diet_id_df.groupby(by="dietID")
macro_names = ["Protein(g)", "Carbohydrate, by difference(g)", "Tota
l lipid (fat)(g)", "Sugars, total including NLEA(g)", "Energy(kcal)"
]
grouped_macros = grouped_diet_id[macro_names].sum() grouped_macros

        Protein(g)  Carbohydrate, by difference(g)  Total lipid (fat
)(g)  \
dietID
1001       121.94                          462.87                13
7.07
1002       141.36                          464.99                28
2.09
1003       194.94                          638.27                21
```

```
9.39
1004        174.13                          399.36                  15
8.96
1005        233.33                          603.83                  39
0.02
1006        168.29                          748.45                  20 9.06
```

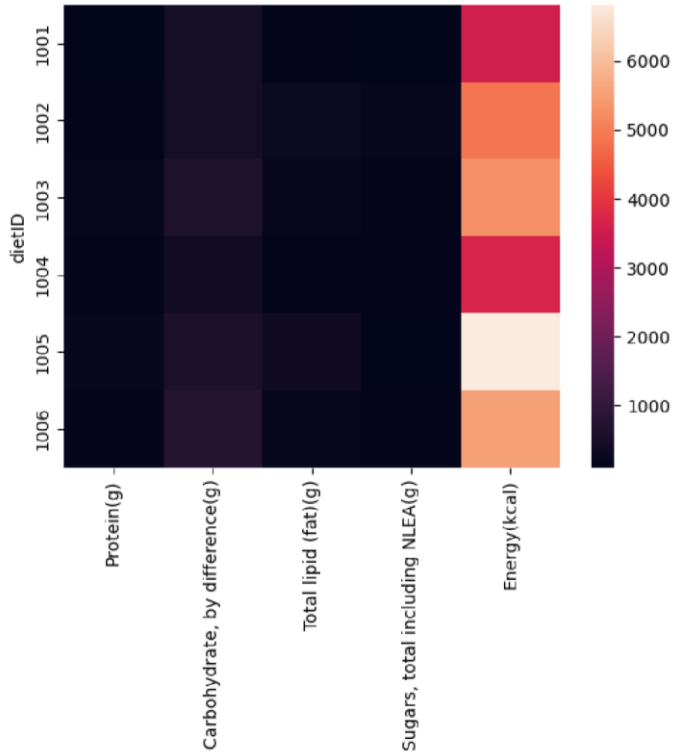
```
        Sugars, total including NLEA(g)  Energy(kcal)   dietID
1001                             107.18          3565
1002                             210.02          4898
1003                             149.12          5271
1004                             134.79          3714
1005                             121.57          6812
1006                             185.22          5520
```

Create a heatmap for the aggregated data

Just for better interpretation

```
import seaborn as sns sns.heatmap(grouped_macros)
```

```
<Axes: ylabel='dietID'>
```



Convert the aggregated data into pandas dataframe

```
macros_dietID_aggr = pd.DataFrame(grouped_macros) macros_dietID_aggr
```

```
        Protein(g)  Carbohydrate, by difference(g)  Total lipid (fat
)(g)  \
dietID
1001        121.94                          462.87                 13
7.07
1002        141.36                          464.99                 28
            2.09
1003        194.94                          638.27                  21
            9.39
1004        174.13                          399.36                 15
8.96
1005        233.33                          603.83                 39
0.02
1006        168.29                          748.45                 20
            9.06

        Sugars, total including NLEA(g)  Energy(kcal)   dietID
1001                            107.18          3565
1002                            210.02          4898
1003                            149.12          5271
1004                            134.79          3714
1005                            121.57          6812
1006                            185.22          5520
```

#Step 3: Assign recommended dietIDs to all the users If the minimum macro levels intake of the user is less than or equal to a diet plan's macro level, then recommend that plan to the user.

```
user_diet_recom=[] user_diet = dict() for uid, uprotein, ucarbs,
ufat, usugar, uenergy in zip( user_macros_ df["Unnamed: 0"],
user_macros_df["Protein (grams/day)"] , user_macros _df["Carbs
(grams/day)"], user_macros_df["Fat (grams/day)"] , user_ma
cros_df["Sugar (grams/day)"], user_macros_df["Food Energy (Calories/
day)"]):   for id, protein, carbs, fat, sugar, energy in
zip(macros_dietID_ag gr.index, macros_dietID_aggr["Protein(g)"],
macros_dietID_aggr["Carb ohydrate, by difference(g)"],
macros_dietID_aggr["Total lipid (fat)( g)"],
macros_dietID_aggr["Sugars, total including NLEA(g)"] , macros_
dietID_aggr["Energy(kcal)"]):
    if(float(uprotein)<=float(protein) and float(ucarbs)<=float( carb
s) and float(ufat)<=float(fat) and float(usugar)<=float(sugar) and f
loat(uenergy)<=float(energy)):
      user_diet ["UserID"] = "User " + str(uid)       user_diet
["Protein (grams/day)"] = float(uprotein)        user_diet
["Carbs (grams/day)"] = float(ucarbs)       user_diet ["Fat
(grams/day)"] = float(ufat)       user_diet ["Sugar
(grams/day)"] = float(usugar)       user_diet ["Food Energy
(Calories/day)"] = float(uenergy)       user_diet["Diet
Protein(g)"] = float(protein)
```

```
      user_diet["Diet Carbohydrate, by difference(g)"] = float( carbs
)
      user_diet["Diet Total lipid (fat)(g)"] = float(fat)
user_diet["Diet Sugars, total including NLEA(g)"] = float( suga r)
user_diet["Diet Energy(cal)"] = float(energy)       user_diet
["DietID"] = id       user_diet_recom.append(user_diet)
user_diet = dict()

user_diet_recom_df = pd.DataFrame(user_diet_recom) user_diet_recom_df

      UserID  Protein (grams/day)  Carbs (grams/day)  Fat (grams/day
)  \
0     User 0                146.0              319.0              68.
0
1     User 0                146.0              319.0              68.
0
2     User 0                146.0              319.0              68. 0
3     User 0                146.0               319.0              68. 0
4     User 1                101.0              220.0              47.
0
..       ...                  ...                ...               ..
.
428  User 98                148.0              323.0             69.
     0
429  User 99                164.0              359.0              77.
0
430  User 99                164.0              359.0              77.
0
431  User 99                164.0              359.0              77.
     0
432  User 99                164.0               359.0
     77. 0

     Sugar (grams/day)  Food Energy (Calories/day)  Diet Protein(g)
\
0     64.0                      2390.0           194.94
1     64.0                      2390.0           174.13 2
64.0                     2390.0           233.33

3                64.0                       2390.0           168.29

4                44.0                       1651.0           121.94
..                ...                          ...              ...
428              65.0                       2423.0           168.29

429              72.0                       2696.0           194.94
```

```
430                72.0                          2696.0            174.13

431                72.0                          2696.0            233.33

432                72.0                          2696.0            168.29

     Diet Carbohydrate, by difference(g)  Diet Total lipid (fat)(g)
\
0                                  638.27                      219.39
1                                  399.36                      158.96

2                                  603.83                      390.02

3                                  748.45                      209.06
4                                  462.87                      137.07
..                                    ...                         ...

428                                748.45                      209.06
429                                638.27                      219.39
430                                399.36                      158.96

431                                603.83                      390.02

432                                748.45                      209.06

     Diet Sugars, total including NLEA(g)  Diet Energy(cal)  DietID
0                                  149.12            5271.0    1003

1                                  134.79            3714.0    1004
                                   2
                                   121.57            6812.0    1005
3                                  185.22            5520.0    1006
4                                  107.18            3565.0    1001

..                                    ...               ...     ...
428                                185.22            5520.0    1006
429                                149.12            5271.0    1003

430                                134.79            3714.0    1004

431                                121.57            6812.0    1005
432                                185.22            5520.0    1006

[433 rows x 12 columns]
```

#Step 4: Export the resulting dataset

```
user_diet_recom_df.to_excel("Diet_recommendation_based_on_user.xlsx"
)
```