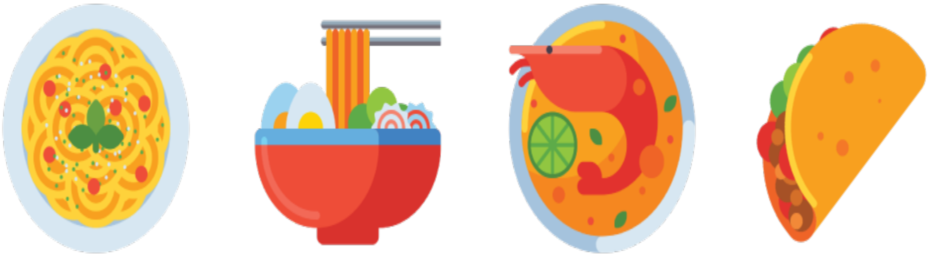
# International Cuisines Project:

### Examining The Variations and Parallels Throughout Cuisines from Different Cultures



**Project Outline**

## - Goals, purposes, and history

### - Introduction

Since the dawn of time, food has been one of humanity's most basic needs. From primitive hunters and gatherers, mankind have evolved unique cooking customs, methods, and a wide range of ingredients, spices, and culinary combinations across time.
Since every country has a different cuisine with distinctive flavors, I'm interested to know if the ingredients (raw materials used in cooking, such potatoes, carrots, etc.) are the same. Comparing the nutritional profiles of different cuisines and determining which countries place a greater emphasis on fats or proteins, as well as which have lower calorie counts, also fascinates me.

### - Objectives and goals

I am aiming to explore the differences and the similarities of traditional international cuisines.
I will focus on the following objectives:

**Differences between types of cuisines:**

- What is the average ingredient number per dish?
- What are the main ingredients of the cuisine?
- What is the average nutritional profile of the cuisine? (calories, fats, proteins, etc)

**Similarities between types of cuisines:**

- What are the universal dishes that all cuisines have? Do they differ from one another?
- What are the main ingredients for all cuisines?
- What is the average nutritional profile for all cuisines?

**With this project proposal, my objectives are:**

1. locating a trustworthy source to gather information about the recipes, ingredients, and nutritional value of different ethnic cuisines.
2. Given the impossibility of gathering data on every cuisine on the planet, I would have to concentrate on those that most accurately reflect well-known geographic regions.
3. Make sure that each cuisine has a comparable amount of samples collected.
4. Store the gathered data in an organized database or dataframe after preprocessing it.
5. Conduct a basic analysis on the information.

## - Data

### - Data requirements

The main components of my research are recipes and cuisines, together with a list of ingredients and nutritional information.
I would have to concentrate on material that is already divided into cuisine sorts or world regions because it can be challenging to determine the type of cuisine simply by reading the name of a recipe.
For that reason, I've chosen a web database called RedipeDB that contains tens of thousands of recipe samples from over 22 world regions.
This database is assembled by a team of professors and students of *The Center for Computational Biology, Indraprastha Institute of Information Technology Delhi*. It contains all the information required to reach the objectives of my project and therefore it would be the only source of data I will refer to.

### - Choices of cuisines

I've decided to select 5 popular cuisines to explore:

- Middle Eastern
- South Asian
- Indian
- French
- Italian

The choice was made based on world-wide popularity and personal curiosity. The aspect of popularity was not measured in terms of population size, but in terms of adaptivity to world-wide cultures. For example: French and Italian cuisines are widley used around the world even in remote places from these countries of origin. The population of Italy is approx 60 million (source: Wikipedia Demographics of Italy), and of France is approx 68 million (source: Wikipedia Demographics of France) yet Pizzas and Coruscants can be found in almost every city in the world and therefore they were selected for this project alongside cuisines that serve a much larger number of population.

### - Limitations and constraints of the data

- **Recipe authenticity concerns:**
  Since my project aims to research cultural traditions of food, it's unclear which recipes fully comply with the cuisine and which are interporated by individual cooks. My hypothesis is that some recipes may not use traditional materials and may be a fusion between different cuisines.

- **Database size and time concerns:**
  As of Jan 2024, RecipeDB contains 5908 pages of recipes, with 20 recipes per page.
  This means that the database contains 118,160 different recipes which would take a substantial amount of time to web scrape.
  Due to time constraints - I decided to focus on the first 500 recipe samples from each cuisine, which translates to 25 pages. 500 samples should be sufficient to perform the first part of the research, and more samples can be obtained using the same scraper at a later point in time.

In [6]:

```
import pandas as pd
import matplotlib
```

## - Loading data from CSV into DataFrames

we will start alanysing the data by loading it into DataFrames and viewing the data

In [7]:

```
# load recipes22.csv
recipe_df = pd.read_csv("recipes22.csv")
recipe_df.head(10)
```

Out[7]:

| | Recipe Title | Region | Country | Servings | Calories (KCal) | Protein (g) | Fat (g) | More Info | primary_key |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Egyptian Lentil Soup | Middle Eastern | Egyptian | 4 | 805.69 | 49.55 | 5.65 | more info | 0 |
| **1** | Egyptian Green Beans with Carrots | Middle Eastern | Egyptian | 4 | 386.76 | 15.56 | 19.72 | more info | 1 |
| **2** | Egyptian Bamia | Middle Eastern | Egyptian | 4 | 1762.76 | 82.9 | 146.89 | more info | 2 |
| **3** | Magpie's Easy Falafel Cakes | Middle Eastern | Egyptian | 3 | 1686.82 | 67.85 | 62.48 | more info | 3 |

|   | Recipe Title | Region | Country | Servings | Calories (KCal) | Protein (g) | Fat (g) | More Info | primary_key |
|---|---|---|---|---|---|---|---|---|---|
| **4** | Dukkah | Middle Eastern | Egyptian | 24 | 1006.23 | 26.18 | 73.17 | more info | 4 |
| **5** | Om Ali | Middle Eastern | Egyptian | 8 | 4117.46 | 46.59 | 212.15 | more info | 5 |
| **6** | Rice Pudding with Lemon Juice and Caramelized ... | Middle Eastern | Egyptian | 4 | 1043.98 | 68.58 | 52.6 | more info | 6 |
| **7** | Couscous with Olives and Sun-Dried Tomato | Middle Eastern | Egyptian | 4 | 1469.86 | 23.61 | 120.24 | more info | 7 |
| **8** | Fava Bean Breakfast Spread | Middle Eastern | Egyptian | 6 | 1065.33 | 63.02 | 24.93 | more info | 8 |
| **9** | Koshary | Middle Eastern | Egyptian | 8 | 1814.17 | 87.48 | 58.29 | more info | 9 |

In [8]:

```
# load ingredients22.csv
ingredients_df = pd.read_csv("ingredients22.csv")
ingredients_df.head(10)
```

Out[8]:

|   | Ingredient Name | Quantity | Unit | State | Energy (kcal) | Carbohydrates | Protein (g) | Total Lipid (Fat) (g) | recipe_key |
|---|---|---|---|---|---|---|---|---|---|
| **0** | water | 3 | cups | NaN | 0 | 0 | 0 | 0 | 0 |
| **1** | red lentil | 1 | cup | NaN | 687.36 | 121.15 | 45.9 | 4.16 | 0 |
| **2** | rom tomato | 1 | NaN | quartered | - | - | - | - | 0 |
| **3** | carrot | 1 | NaN | quartered | 52.48 | 12.26 | 1.19 | 0.3 | 0 |
| **4** | onion | 1 | NaN | quartered | 28 | 6.53 | 0.77 | 0.07 | 0 |

|  | Ingredient Name | Quantity | Unit | State | Energy (kcal) | Carbohydrates | Protein (g) | Total Lipid (Fat) (g) | recipe_key |
|---|---|---|---|---|---|---|---|---|---|
| **5** | garlic | 4 | cloves | quartered | 17.88 | 3.96 | 0.76 | 0.06 | 0 |
| **6** | water | 1 | cup | NaN | 0 | 0 | 0 | 0 | 0 |
| **7** | cumin | 2 | teaspoons | ground | 15.75 | 1.85 | 0.74 | 0.93 | 0 |
| **8** | sea salt | 1/2 | teaspoon | NaN | - | - | - | - | 0 |
| **9** | black pepper | 1/2 | teaspoon | cracked | 2.88 | 0.73 | 0.11 | 0.03 | 0 |

## - Disposing of Unnescesary Columns

In both file, we can find columns containing data that doesnt relate to this project
In recipe.csv we do not need the *More Info* column as it contains a repeated statement with no significance to this project.
Additionally, in ingredients.csv we do not need the *State* column as weather the egg is boiled or poached or if the beef is ground or shreded - will not affect the outcome of this project and therefore the *State* variable of ingredients is unnescesary. Also, since many of its values are Nan (based on a fast glimpse) I prefer to remove it altogether.
We will then remove both of those columns with the .drop() method.

In [9]:

```
recipe_df = recipe_df.drop(columns=["More Info"])
ingredients_df = ingredients_df.drop(columns=["State"])
```

## - Verify Sample Size

In [10]:

```
# fetch the unique values of the region column and count occurances
cuisine_types = recipe_df["Region"].value_counts()[:8]

print("Fetched", len(cuisine_types), "Cuisine Types:")

cuisine_types
```

```
Fetched 5 Cuisine Types:
```

Out[10]:

```
Region
Middle Eastern         500
Southeast Asian        500
Indian Subcontinent    500
```

```
French                  500
Italian                 500
Name: count, dtype: int64
```

we see that the number of recipes is a perfect match from one cuisine to another. This ensures a proportional representation of each cuisine in terms of sample size.

**- All Cuisine:**

I will obtain all the required data and print and plot the results:

In [11]:

```
#All 5 Cuisine Types:
universal_ingr = ingredients_df['Ingredient Name'].value_counts()[:10]
print("***** Universal Ingrediants: *****")
print(universal_ingr)

# clear data from records that may trigger errors
recipe_df = recipe_df.apply(pd.to_numeric, errors='coerce')

# fetch the average calories values
calories_universal = recipe_df['Calories (KCal)'].mean()
proteins_universal = recipe_df['Protein (g)'].mean()
fat_universal = recipe_df['Fat (g)'].mean()

print("\n***** Universal Nutritional Average Stats: *****")
print("Calories:", calories_universal)
print("Proteins:", proteins_universal)
print("Fats:", fat_universal)

print("\n***** Visuallise Universal Nutritional Data: *****")

df_toplot = pd.DataFrame({'Nutritional Profile':['Proteins', 'Fats'], 'Value (g)':[proteins_universal, fat_universal]})
# plot graph
df_toplot.plot.bar(x='Nutritional Profile', y='Value (g)', rot=0, title='Universal Average Nutritional Values')
```

```
***** Universal Ingrediants: *****
Ingredient Name
salt             1194
onion             890
water             884
garlic            849
olive oil         650
butter            541
vegetable oil     485
tomato            457
black pepper      440
```

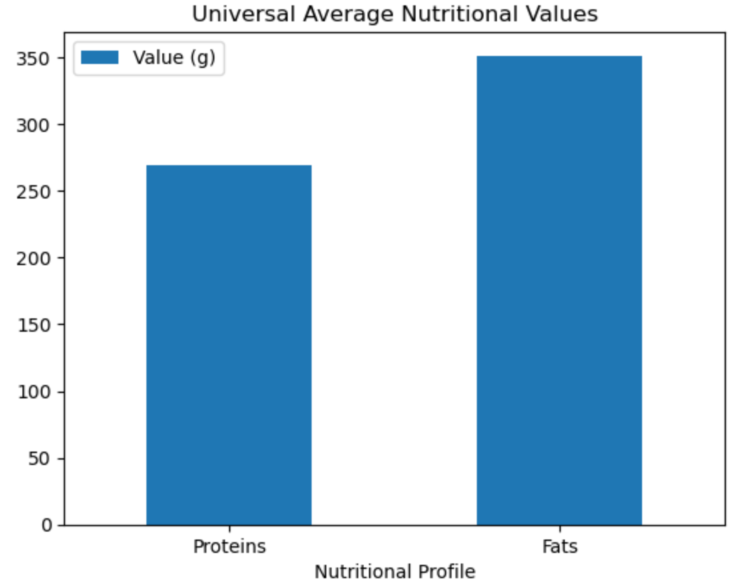
```
white sugar       401
Name: count, dtype: int64

***** Universal Nutritional Average Stats: *****
Calories: 5599.323789642714
Proteins: 269.4782135688478
Fats: 351.4684785226817

***** Visuallise Universal Nutritional Data: *****
```

Out[11]:

```
<Axes: title={'center': 'Universal Average Nutritional Values'}, xlabel='Nutr
itional Profile'>
```



## - Cusine-Based Analytics

In this section we will find out the differences between cuisines and evaluate them one by one

In [12]:

```
# Middle eastern Cuisine:
recipe_df = pd.read_csv("recipes22.csv")

# fetch middle eastern recipes only
middle_eastern = recipe_df[recipe_df["Region"]=="Middle Eastern"]

# fetch middle eastern ingredients only
middle_ingredients = ingredients_df[ingredients_df["recipe_key"].isin(middle_eastern["primary_key"])]

# fetch top 10 ingredients
top_middle_ing = middle_ingredients['Ingredient Name'].value_counts()[:10]
print("***** Top Middle Eastern Ingrediants: *****")
```

```python
print(top_middle_ing)

# clear data from records that may trigger errors
middle_eastern = middle_eastern.apply(pd.to_numeric, errors='coerce')

# fetch the average calories values
calories_middle = middle_eastern['Calories (KCal)'].mean()
proteins_middle = middle_eastern['Protein (g)'].mean()
fat_middle = middle_eastern['Fat (g)'].mean()

print("\n***** Middle Eastern Nutritional Average Stats: *****")
print("Calories:", calories_middle)
print("Proteins:", proteins_middle)
print("Fats:", fat_middle)

print("\n***** Visualise middle Eastern Nutritional Data: *****")

df_toplot = pd.DataFrame({'Nutritional Profile':['Proteins', 'Fats'], 'Value (g)':[proteins_middle, fat_middle]})
# plot graph
df_toplot.plot.bar(x='Nutritional Profile', y='Value (g)', rot=0, title='Middle Eastern Average Nutritional Values')
```

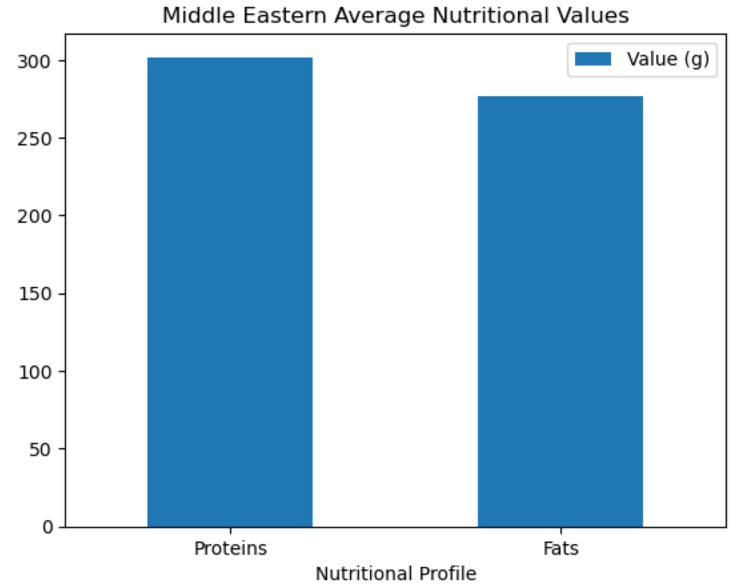
```
***** Top Middle Eastern Ingrediants: *****
Ingredient Name
salt            259
onion           214
olive oil       213
water           207
garlic          187
lemon juice     137
cumin           133
black pepper    116
tomato          114
parsley          90
Name: count, dtype: int64

***** Middle Eastern Nutritional Average Stats: *****
Calories: 4725.916412825652
Proteins: 301.81839679358717
Fats: 276.3223647294589

***** Visualise middle Eastern Nutritional Data: *****
```

Out[12]:

```
<Axes: title={'center': 'Middle Eastern Average Nutritional Values'}, xlabel='Nutritional Profile'>
```



In [13]:

```
#South Asian Cuisine:
recipe_df = pd.read_csv("recipes22.csv")

# fetch south asian recipes only
south_asian = recipe_df[recipe_df["Region"]=="Southeast Asian"]

# fetch south asian ingredients only
south_asian_ingredients = ingredients_df[ingredients_df["recipe_key"].isin(south_asian["primary_key"])]

# fetch top 10 ingredients
top_asian_ing = south_asian_ingredients['Ingredient Name'].value_counts()[:10]
print("***** Top South Asian Ingredients: *****")
print(top_asian_ing)

# clear data from records that may trigger errors
south_asian = south_asian.apply(pd.to_numeric, errors='coerce')

# fetch the average calories values
calories_south_asian = south_asian['Calories (KCal)'].mean()
proteins_south_asian = south_asian['Protein (g)'].mean()
fat_south_asian = south_asian['Fat (g)'].mean()

print("\n***** South Asian Nutritional Average Stats: *****")
print("Calories:", calories_south_asian)
print("Proteins:", proteins_south_asian)
print("Fats:", fat_south_asian)
```

```python
print("\n***** Visuallise South Asian Nutritional Data: *****")

df_toplot = pd.DataFrame({'Nutritional Profile':['Proteins', 'Fats'], 'Value (g)':[proteins_south_asian, fat_south_asian]})
# plot graph
df_toplot.plot.bar(x='Nutritional Profile', y='Value (g)', rot=0, title='Southeast Asian Average Nutritional Values')
```

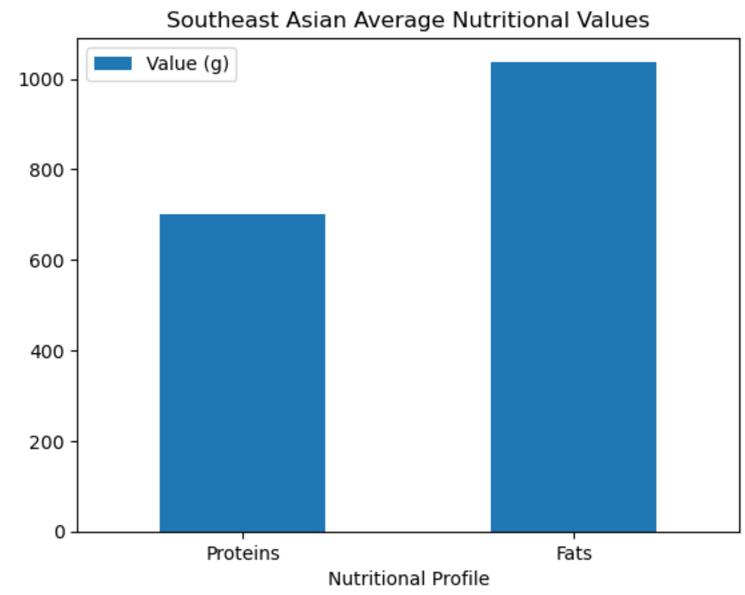
```
***** Top South Asian Ingredients: *****
Ingredient Name
salt                 218
water                213
garlic               185
onion                183
soy sauce            148
vegetable oil        119
garlic clove         116
white sugar          108
ginger                89
coconut milk          85
Name: count, dtype: int64

***** South Asian Nutritional Average Stats: *****
Calories: 15091.77847082495
Proteins: 702.0106841046277
Fats: 1038.2386519114689

***** Visuallise South Asian Nutritional Data: *****
```

Out[13]:

```
<Axes: title={'center': 'Southeast Asian Average Nutritional Values'}, xlabel='Nutritional Profile'>
```



In [14]:

```
#Indian Cuisine:
recipe_df = pd.read_csv("recipes22.csv")

# fetch Indian recipes only
indian = recipe_df[recipe_df["Region"]=="Indian Subcontinent"]

# fetch Indian ingredients only
indian_ingredients = ingredients_df[ingredients_df["recipe_key"].isin(indian["primary_key"])]

# fetch top 10 ingredients
top_indian_ing = indian_ingredients['Ingredient Name'].value_counts()[:10]
print("***** Top Indian Ingredients: *****")
print(top_indian_ing)

# clear data from records that may trigger errors
indian = indian.apply(pd.to_numeric, errors='coerce')

# fetch the average calories values
calories_indian = indian['Calories (KCal)'].mean()
proteins_indian = indian['Protein (g)'].mean()
fat_indian = indian['Fat (g)'].mean()

print("\n***** Indian Nutritional Average Stats: *****")
print("Calories:", calories_indian)
print("Proteins:", proteins_indian)
print("Fats:", fat_indian)
print("\n***** Visuallise Indian Nutritional Data: *****")

df_toplot = pd.DataFrame({'Nutritional Profile':['Proteins', 'Fats'], 'Value (g)':[proteins_indian, fat_indian]})
# plot graph
df_toplot.plot.bar(x='Nutritional Profile', y='Value (g)', rot=0, title='Indian Average Nutritional Values')
```

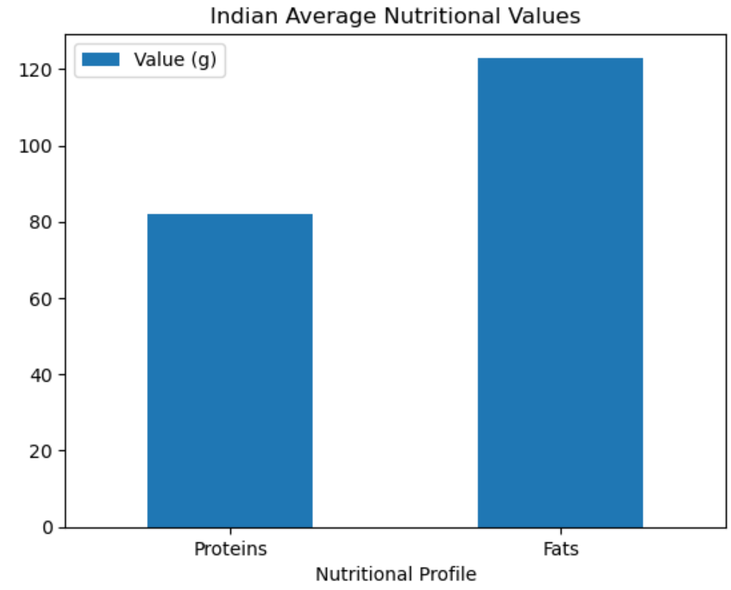
```
***** Top Indian Ingredients: *****
Ingredient Name
salt                 361
onion                292
water                256
vegetable oil        227
turmeric             223
garlic               217
cilantro             146
garam masala         144
cumin                144
tomato               140
Name: count, dtype: int64

***** Indian Nutritional Average Stats: *****
Calories: 1848.82854
Proteins: 81.91157999999999
Fats: 122.98782000000003

***** Visuallise Indian Nutritional Data: *****
```

Out[14]:

```
<Axes: title={'center': 'Indian Average Nutritional Values'}, xlabel='Nutriti
onal Profile'>
```



In [15]:

```
#French Cuisine:
recipe_df = pd.read_csv("recipes22.csv")

# fetch French recipes only
french = recipe_df[recipe_df["Region"]=="French"]

# fetch French ingredients only
french_ingredients = ingredients_df[ingredients_df["recipe_key"].isin(french["primary_key"])]

# fetch top 10 ingredients
top_french_ing = french_ingredients['Ingredient Name'].value_counts()[:10]
print("***** Top French Ingredients: *****")
print(top_french_ing)

# clear data from records that may trigger errors
french = french.apply(pd.to_numeric, errors='coerce')

# fetch the average calories values
calories_french = french['Calories (KCal)'].mean()
proteins_french = french['Protein (g)'].mean()
fat_french = french['Fat (g)'].mean()

print("\n***** French Nutritional Average Stats: *****")
print("Calories:", calories_french)
print("Proteins:", proteins_french)
print("Fats:", fat_french)

print("\n***** Visuallise French Nutritional Data: *****")

df_toplot = pd.DataFrame({'Nutritional Profile':['Proteins', 'Fats'], 'Value (g)':[proteins_french, fat_french]})
# plot graph
df_toplot.plot.bar(x='Nutritional Profile', y='Value (g)', rot=0, title='French Average Nutritional Values')
```

```
***** Top French Ingredients: *****
Ingredient Name
salt             206
butter           205
egg              137
water            123
sugar            123
milk             109
olive oil        108
onion             80
garlic clove      78
```

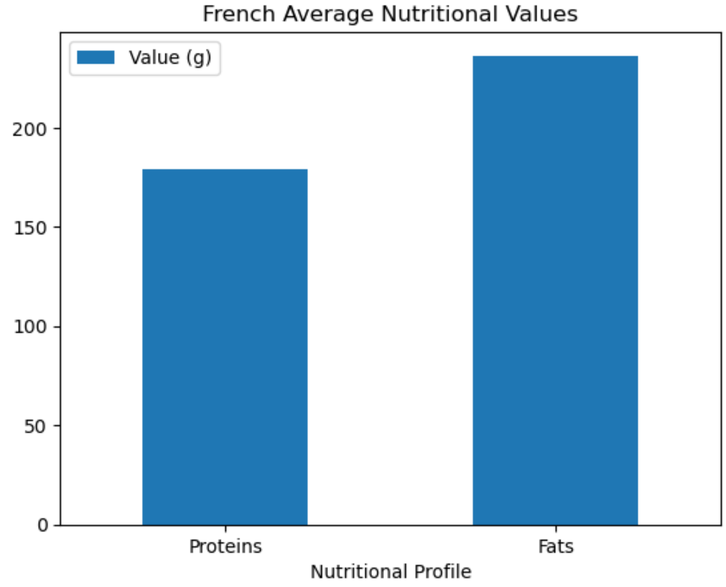
```
purpose flour     78
Name: count, dtype: int64

***** French Nutritional Average Stats: *****
Calories: 4820.510885311872
Proteins: 179.25158953722334
Fats: 236.29722334004023

***** Visuallise French Nutritional Data: *****
```

Out[15]:

```
<Axes: title={'center': 'French Average Nutritional Values'}, xlabel='Nutriti
onal Profile'>
```



In [16]:

```
#Italian Cuisine:
recipe_df = pd.read_csv("recipes22.csv")

# fetch Italian recipes only
italian = recipe_df[recipe_df["Region"]=="Italian"]

# fetch Italian ingredients only
italian_ingredients = ingredients_df[ingredients_df["recipe_key"].isin(italian["primary_key"])]

# fetch top 10 ingredients
top_italian_ing = italian_ingredients['Ingredient Name'].value_counts()[:10]
print("***** Top French Ingredients: *****")
print(top_italian_ing)

# clear data from records that may trigger errors
```

```python
italian = italian.apply(pd.to_numeric, errors='coerce')

# fetch the average calories values
calories_italian = italian['Calories (KCal)'].mean()
proteins_italian = italian['Protein (g)'].mean()
fat_italian = italian['Fat (g)'].mean()

print("\n***** Italian Nutritional Average Stats: *****")
print("Calories:", calories_italian)
print("Proteins:", proteins_italian)
print("Fats:", fat_italian)

print("\n***** Visuallise Italian Nutritional Data: *****")

df_toplot = pd.DataFrame({'Nutritional Profile':['Proteins', 'Fats'], 'Value (g)':[proteins_italian, fat_italian]})
# plot graph
df_toplot.plot.bar(x='Nutritional Profile', y='Value (g)', rot=0, title='Italian Average Nutritional Values')
```

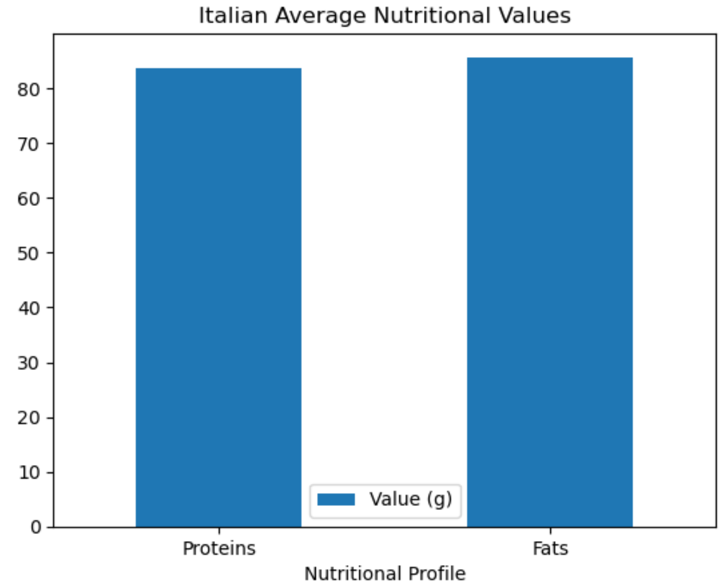
```
***** Top French Ingredients: *****
Ingredient Name
garlic             218
olive oil          201
salt               150
butter             132
parmesan cheese    127
onion              121
tomato             117
black pepper       116
white sugar        114
basil              103
Name: count, dtype: int64

***** Italian Nutritional Average Stats: *****
Calories: 1543.89797188755
Proteins: 83.77451807228917
Fats: 85.71261044176708

***** Visuallise Italian Nutritional Data: *****
```

Out[16]:

```
<Axes: title={'center': 'Italian Average Nutritional Values'}, xlabel='Nutrit
ional Profile'>
```



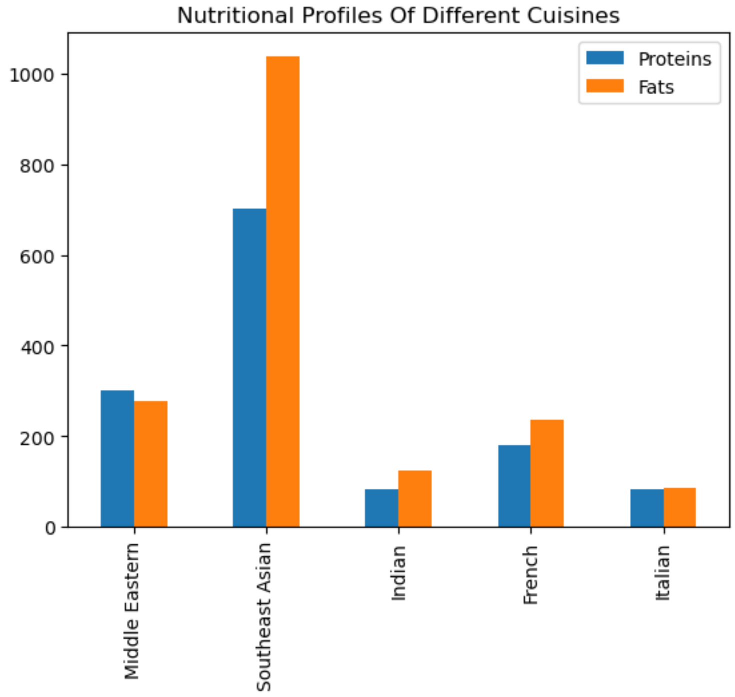
# - Conclusions

## - Total Visualised Outcome

I will combine all the data we obtained above into one visual plot:

In [17]:

```
proteins = [proteins_middle, proteins_south_asian, proteins_indian, proteins_french, proteins_italian]
fats = [fat_middle, fat_south_asian, fat_indian, fat_french, fat_italian]
index = ['Middle Eastern', 'Southeast Asian', 'Indian', 'French', 'Italian']
df = pd.DataFrame({'Proteins': proteins,'Fats': fats}, index=index)
ax = df.plot.bar(rot=90, title="Nutritional Profiles Of Different Cuisines")
```



## - Nutritional Profile Conclusions

In the above plot, we can see that each cuisine has a different balance between fats and proteins.

- it seems that the French and the Indian cuisins have similar protein to fat ratios.
- it seems that the Itallian cuisine has almost an even balance between proteins and fats.
- it seems that the Middle Eastern cuisine is the only one that includes more proteins than fats.
- it seems that the Southeast Asien cuisine has the biggest gap between proteins and fats.

## - Gaps in my approach to nutritional profiling

The evaluation of the above nutritional values is not as accurate as it could be. As each recipe has a different number and convention of "*Servings*" - it is difficult to determine the exact nutritional values.
In some recipes, "*Servings*" refers to a number of cookies or chocolate balls, in other recipes it relates to the number of main dishes or loafs. The servings are not measured universaly, which makes the results approximate rather than obsolute.

## - Insights

- My analysis shows that each cuisine has a unique balance of nutritional values and it can be expended further introducing a larger number of samples.
- In addition, even though many ingredients are universaly popular - among the top 10 most used ingredients in each cuisine we can find unique values.
  For example: butter is the top universal ingredient, however - it doesn't appear in the top 10 of the Middle Eastern, Southeast Asian and Indian cuisines.

- Some cuisines relate to one another.
  For example: 6 of the top 10 ingredients of the French and Itallian cuisines are a perfect match! In particular: olive oil, salt, butter, onion, black pepper, sugar

## - Credits and Resources

- The icons in the very top of the notebook were taken from the free stock image website: www.flaticon.com
- I refered to the Pandas documentation to generate the bar plots: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.bar.html
- I also refered to the Pandas documentation to generate word counts: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value_counts.html