Name: Manjunath Angadi

Indian Food EDA

```
# This Python 3 environment comes with many helpful analytics libraries ins
talled
# It is defined by the kaggle/python Docker image: https://github.com/kaggl
e/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will
list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Imports

```
import matplotlib.pyplot as plt
import seaborn as sns
```

Initial data exploration

```
# Reading the data
df = pd.read_csv('/kaggle/input/indian-food-101/indian_food.csv')
df.head()
```

	name	ingredients	diet	prep_time	cook_time	flavor_profile	course	state	region
0	Balu shahi	Maida flour, yogurt, oil, sugar	vegetarian	45	25	sweet	dessert	West Bengal	East
1	Boondi	Gram flour, ghee, sugar	vegetarian	80	30	sweet	dessert	Rajasthan	West
2	Gajar ka halwa	Carrots, milk, sugar, ghee, cashews, raisins	vegetarian	15	60	sweet	dessert	Punjab	North
3	Ghevar	Flour, ghee, kewra, milk, clarified butter, su...	vegetarian	15	30	sweet	dessert	Rajasthan	West
4	Gulab jamun	Milk powder, plain flour, baking powder, ghee,...	vegetarian	15	40	sweet	dessert	West Bengal	East

Name: Manjunath Angadi

```
# Relevant info about all the attributes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 255 entries, 0 to 254
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   name            255 non-null    object
 1   ingredients     255 non-null    object
 2   diet            255 non-null    object
 3   prep_time       255 non-null    int64 
 4   cook_time       255 non-null    int64 
 5   flavor_profile  255 non-null    object
 6   course          255 non-null    object
 7   state           255 non-null    object
 8   region          254 non-null    object
dtypes: int64(2), object(7)
memory usage: 18.1+ KB
```

```
# Information about the numeric attributes
df.describe()
```

	prep_time	cook_time
count	255.000000	255.000000
mean	31.105882	34.529412
std	72.554409	48.265650
min	-1.000000	-1.000000
25%	10.000000	20.000000
50%	10.000000	30.000000
75%	20.000000	40.000000
max	500.000000	720.000000

```
# Number of unique values of each attribute
print(df.nunique())
```

```
name              255
ingredients       252
diet                2
prep_time          22
cook_time          19
flavor_profile      5
course              4
state              25
```

```
region             7
dtype: int64
```

```
# Unique values
print(df['diet'].unique())
print(df['flavor_profile'].unique())
print(df['course'].unique())
print(df['region'].unique())
print(df['state'].unique())
```

```
['vegetarian' 'non vegetarian']
['sweet' 'spicy' 'bitter' '-1' 'sour']
['dessert' 'main course' 'starter' 'snack']
['East' 'West' 'North' '-1' 'North East' 'South' 'Central' nan]
['West Bengal' 'Rajasthan' 'Punjab' 'Uttar Pradesh' '-1' 'Odisha'
 'Maharashtra' 'Uttarakhand' 'Assam' 'Bihar' 'Andhra Pradesh' 'Karnataka'
 'Telangana' 'Kerala' 'Tamil Nadu' 'Gujarat' 'Tripura' 'Manipur'
 'Nagaland' 'NCT of Delhi' 'Jammu & Kashmir' 'Chhattisgarh' 'Haryana'
 'Madhya Pradesh' 'Goa']
```

First Thoughts

Looking at the basic information, it seems that the data needs some formatting:

- The ingredients column in specific could use some changes, I'll be switching it from a single string to a list of strings in order to navigate it better.
- There's a null value in the 'region' column, but since there aren't any null values for state, I believe I'll be able to fill this entry out based on the other data.
- Both the flavor profile, region and state columns present a -1 value, I'll change it to 'others' for now since I don't know what might be the correct entry.
- There are also -1 values in the preparation time and cooking time columns, I'll switch them to 0s so they are easier to handle.

Name: Manjunath Angadi

Data Preparation

I'll first deal with the ingredients column.

In [8]:

```
# Turning the ingredients entry into list instead of a single string
ingredients = []

for row in range(len(df)):
    ing = df['ingredients'][row].replace(" ", "").split(',')
    ingredients.append(ing)

df['ingredients'] = ingredients
df.head()
```

	name	ingredients	diet	prep_time	cook_time	flavor_profile	course	state	region
0	Balu shahi	[Maidaflour, yogurt, oil, sugar]	vegetarian	45	25	sweet	dessert	West Bengal	East
1	Boondi	[Gramflour, ghee, sugar]	vegetarian	80	30	sweet	dessert	Rajasthan	West
2	Gajar ka halwa	[Carrots, milk, sugar, ghee, cashews, raisins]	vegetarian	15	60	sweet	dessert	Punjab	North
3	Ghevar	[Flour, ghee, kewra, milk, clarifiedbutter, su...	vegetarian	15	30	sweet	dessert	Rajasthan	West
4	Gulab jamun	[Milkpowder, plainflour, bakingpowder, ghee, m...	vegetarian	15	40	sweet	dessert	West Bengal	East

```
# Identifying the null value
df[df['region'].isnull()]
```

	name	ingredients	diet	prep_time	cook_time	flavor_profile	course	state	region
110	Panjeeri	[Wholewheatflour, muskmelonseeds, poppyseeds, ...	vegetarian	10	25	sweet	dessert	Uttar Pradesh	NaN

```
df[df['state'] == 'Uttar Pradesh']
```

Name: Manjunath Angadi

	name	ingredients	diet	prep_time	cook_time	flavor_profile	course	state	region
6	Jalebi	[Maida, cornflour, bakingsoda, vinegar, curd, ...	vegetarian	10	50	sweet	dessert	Uttar Pradesh	North
13	Petha	[Firmwhitepumpkin, sugar, kitchenlime, alumpow...	vegetarian	10	30	sweet	dessert	Uttar Pradesh	North
15	Rabri	[Condensedmilk, sugar, spices, nuts]	vegetarian	10	45	sweet	dessert	Uttar Pradesh	North
18	Sohan halwa	[Cornflour, ghee, dryfruits]	vegetarian	10	60	sweet	dessert	Uttar Pradesh	North
90	Kachori	[Moongdal, rava, garammasala, dough, fennelseeds]	vegetarian	30	60	spicy	snack	Uttar Pradesh	North
95	Kofta	[Paneer, potato, cream, cornflour, garammasala]	vegetarian	20	40	spicy	main course	Uttar Pradesh	North
97	Lauki ke kofte	[Bottlegourd, garammasalapowder, gramflour, gi...	vegetarian	20	40	spicy	main course	Uttar Pradesh	North
105	Navrattan korma	[Greenbeans, potatoes, khuskhus, lowfat, garam...	vegetarian	25	40	spicy	main course	Uttar Pradesh	North
110	Panjeeri	[Wholewheatflour, muskmelonseeds, poppyseeds, ...	vegetarian	10	25	sweet	dessert	Uttar Pradesh	NaN

All the meals from Uttar Pradesh are from the North region, I'll fill the null entry with North.

```
# # Correcting the entry and checking if it worked
df['region'] = df['region'].replace(np.nan, 'North')
df[df['state'].isnull()]
```

	name	ingredients	diet	prep_time	cook_time	flavor_profile	course	state	region

```
# Replacing weird entries with values that makes more sense
df['prep_time'] = df['prep_time'].replace(-1, 0)
df['cook_time'] = df['cook_time'].replace(-1, 0)
df['flavor_profile'] = df['flavor_profile'].replace('-1', 'others')
df['region'] = df['region'].replace('-1', 'Others')
df['state'] = df['state'].replace('-1', 'Others')

print(df['flavor_profile'].unique())
print(df['region'].unique())
print(df['state'].unique())
print(df.describe())
```

```
['sweet' 'spicy' 'bitter' 'others' 'sour']
['East' 'West' 'North' 'Others' 'North East' 'South' 'Central']
['West Bengal' 'Rajasthan' 'Punjab' 'Uttar Pradesh' 'Others' 'Odisha'
 'Maharashtra' 'Uttarakhand' 'Assam' 'Bihar' 'Andhra Pradesh' 'Karnataka'
 'Telangana' 'Kerala' 'Tamil Nadu' 'Gujarat' 'Tripura' 'Manipur'
 'Nagaland' 'NCT of Delhi' 'Jammu & Kashmir' 'Chhattisgarh' 'Haryana'
```

```
 'Madhya Pradesh' 'Goa']
        prep_time   cook_time
count  255.000000  255.000000
mean    31.223529   34.639216
std     72.502844   48.185452
min      0.000000    0.000000
25%     10.000000   20.000000
50%     10.000000   30.000000
75%     20.000000   40.000000
max    500.000000  720.000000
```

Ingredients per Region

For this analysis I'll create a new dataset allowing me to check the frequency with witch the ingredien ts appears in each region.

```
# Lists that will be filled with each ingredient value, repeated or not
East = []
West = []
North = []
Others = []
NorthEast = []
South = []
Central = []

# Filling the lists
for row in range(len(df)):
    if df['region'][row] == 'East':
        East.extend(df['ingredients'][row])
    if df['region'][row] == 'West':
        West.extend(df['ingredients'][row])
    if df['region'][row] == 'North':
        North.extend(df['ingredients'][row])
    if df['region'][row] == 'North East':
        NorthEast.extend(df['ingredients'][row])
    if df['region'][row] == 'South':
        South.extend(df['ingredients'][row])
    if df['region'][row] == 'Central':
        Central.extend(df['ingredients'][row])
    if df['region'][row] == 'Others':
        Others.extend(df['ingredients'][row])

# Series witch include each unique ingredient value
all_ing = pd.Series(East+West+North+Others+NorthEast+South+Central).unique
()
```

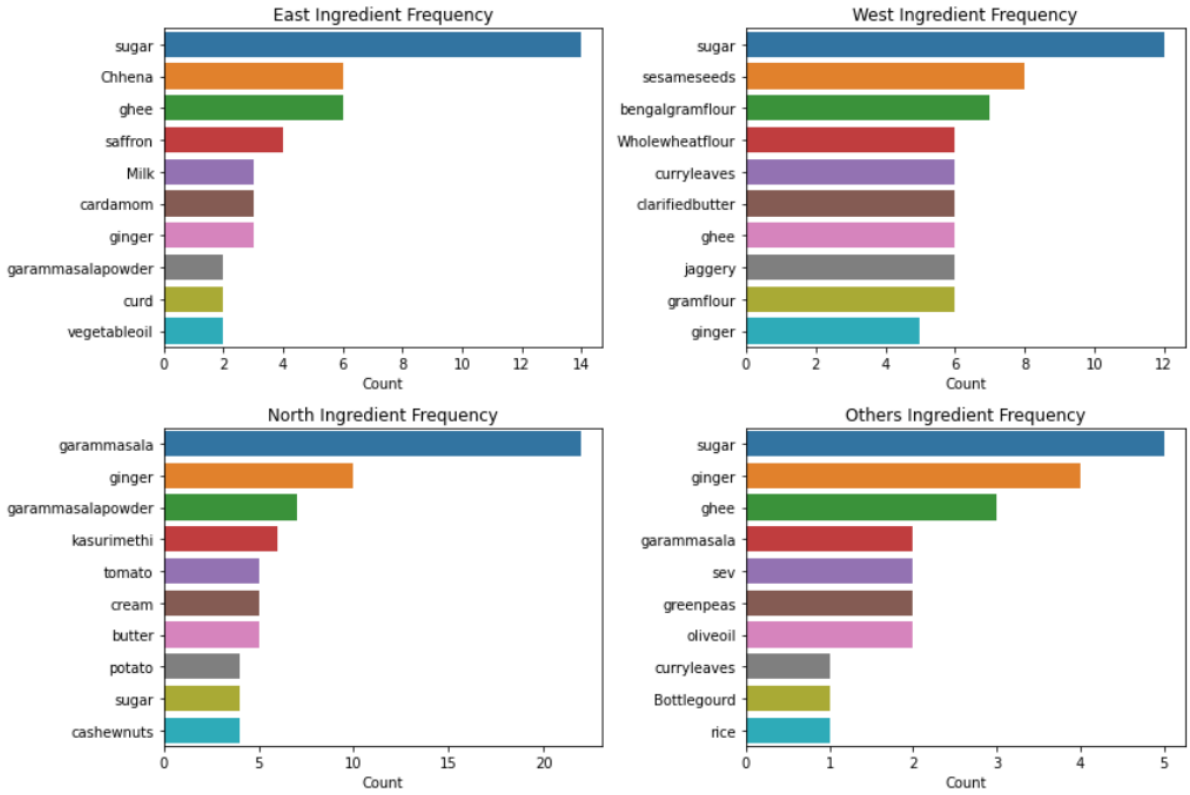
Name: Manjunath Angadi

```
# Lists that will store the number of times each ingredient appears
EastC = []
WestC = []
NorthC = []
OthersC = []
NorthEC = []
SouthC = []
CentralC = []

# Filling the lists
for ingredient in all_ing:
    EastC.append(East.count(ingredient))
    WestC.append(West.count(ingredient))
    NorthC.append(North.count(ingredient))
    OthersC.append(Others.count(ingredient))
    NorthEC.append(NorthEast.count(ingredient))
    SouthC.append(South.count(ingredient))
    CentralC.append(Central.count(ingredient))

# Creating the dataset with the values collected
ing_per_region = pd.DataFrame({'East': EastC, 'West': WestC, 'North': Nort
hC, 'Others': OthersC, 'North East': NorthEC,
                               'South': SouthC, 'Central': CentralC}, inde
x=all_ing)

# Adding a column with the sum of all the other columns
ing_per_region['Sum'] = (ing_per_region['East'] + ing_per_region['West'] +
ing_per_region['North'] + ing_per_region['Others']
                         + ing_per_region['North East'] + ing_per_region['S
outh'] + ing_per_region['Central'])

ing_per_region.head()
```

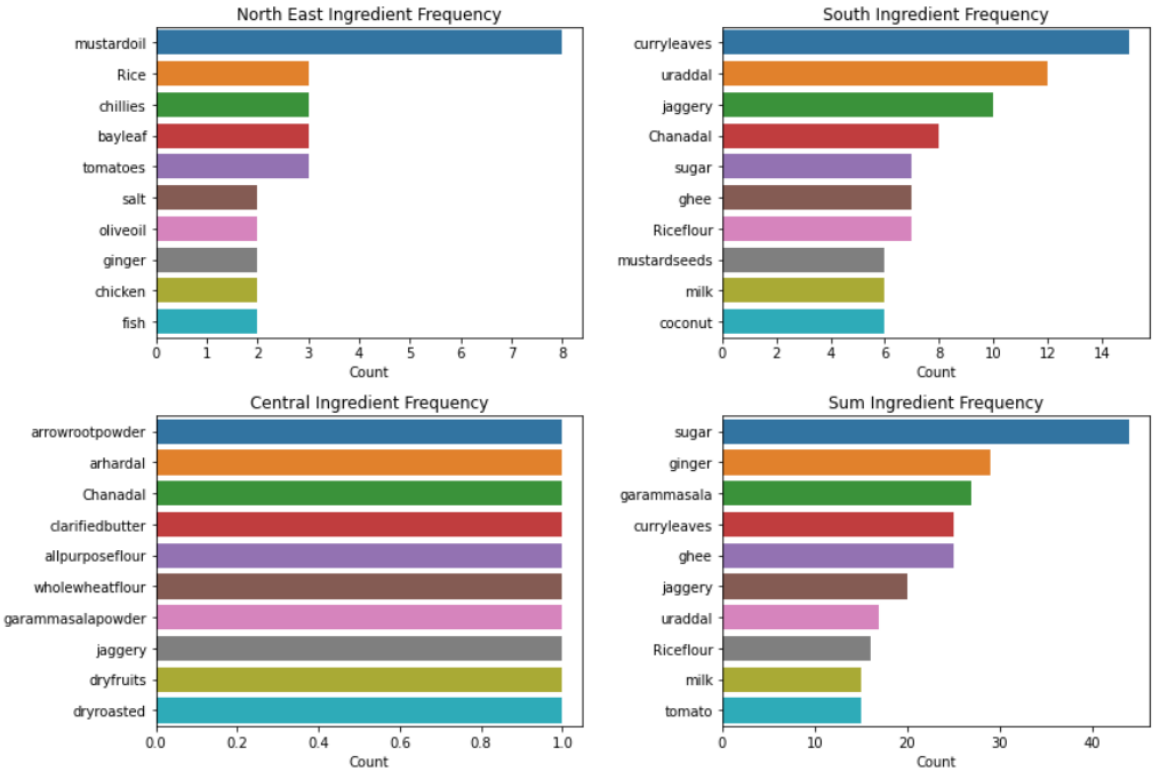
	East	West	North	Others	North East	South	Central	Sum
Maidaflour	1	0	0	0	0	1	0	2
yogurt	1	1	2	0	0	0	0	4
oil	2	2	1	0	0	0	0	5
sugar	14	12	4	5	2	7	0	44
Milkpowder	1	0	0	0	0	0	1	2

Name: Manjunath Angadi

```
# Plotting the frequency of appearence graphs
plt.figure(figsize=(12,16))

for n, region in enumerate(ing_per_region.columns):
    ordered = ing_per_region.sort_values([region], ascending=False)[0:10]
    plt.subplot(4, 2, n+1)
    sns.barplot(x=ordered[region], y=ordered.index, orient='h')
    plt.xlabel('Count')
    plt.title(f'{region} Ingredient Frequency')

plt.tight_layout()
plt.show()
```



Name: Manjunath Angadi

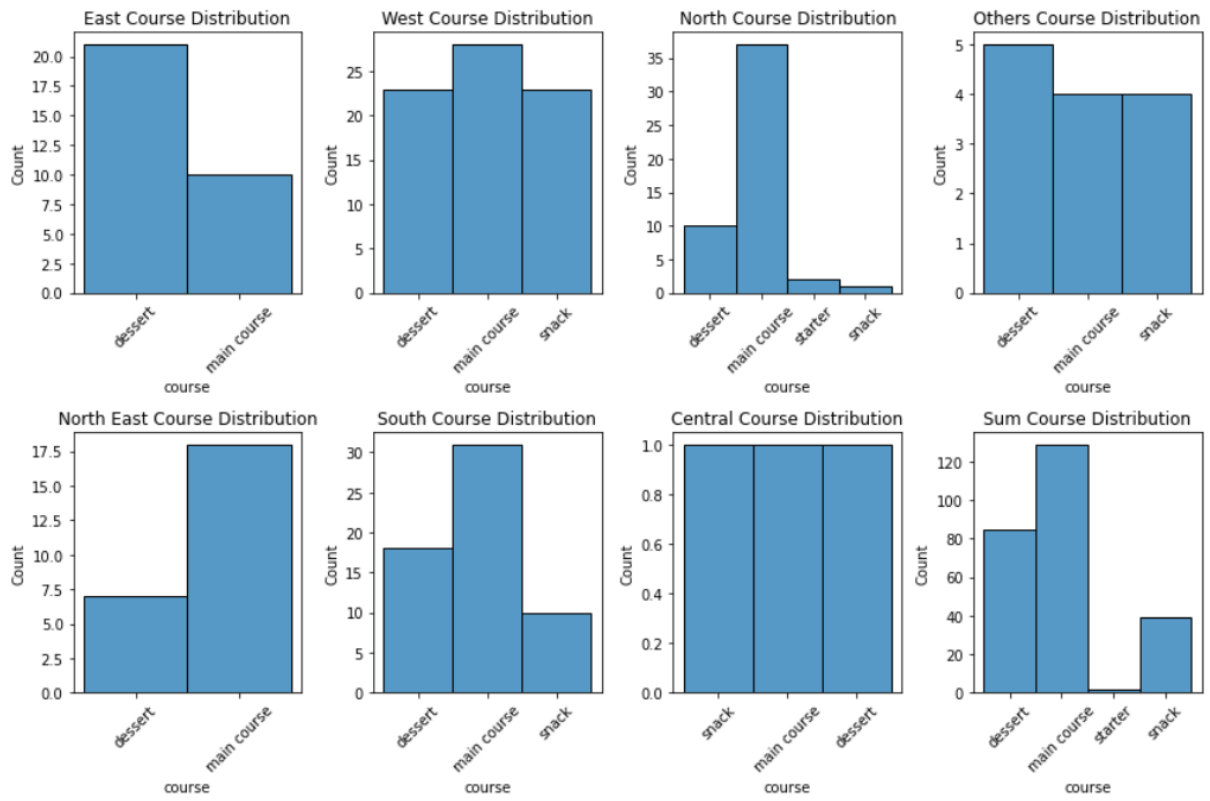


First of all it's important to remember that this dataset doesn't include information about the frequency each of these dishes are consumed in India. Therefore, this analysis will be based solely in the frequency that the ingredients show throughout the recipes for each region, and may be biased depending on how well this dataset represent the indian diet.

With that in mind, our graphs can be pretty telling, the East, West and Other regions show a high count of sugar. This may be related to the amount of desserts in there. Let's check it out by observing the course distributions through each region.

```
plt.figure(figsize=(12,8))

for n, region in enumerate(ing_per_region.columns):
    if region == 'Sum':
        plt.subplot(2, 4, n+1)
        sns.histplot(df['course'])
        plt.title(f'{region} Course Distribution')
        plt.xticks(rotation=45)
    else:
        plt.subplot(2, 4, n+1)
        sns.histplot(df[df['region'] == region]['course'])
        plt.title(f'{region} Course Distribution')
        plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```

Name: Manjunath Angadi



As I thought these regions have a lot of dessert recipes related to it, causing the count of sugar in it to be really high. This doesn't necessarily mean these regions use more sugar in their recipes, since of the course distribution. A conclusion we can take from it though is that most indian desserts have sugar in it.

Another hipothesis we can formulate from our graphs is that the ingredient, garammasala, is most used in the North region. With this same logic we could infer the North East region uses more mustardoil and the South uses a lot of curry leaves.

Unfornatelly the amount of plates in the central region was to little for the analysis to make sense...

Another important thing these graphs show us is that our dataset have a really small amount of starter courses, this will probably show up in our next analysis topic.

Top 10 Shortest Overall Time Meal per Course

For this analysis I'll be adding a new column to our first dataset that will give us the sum of preparati on time and cooking time.

```
df['Time'] = df['prep_time'] + df['cook_time']
df.head()
```

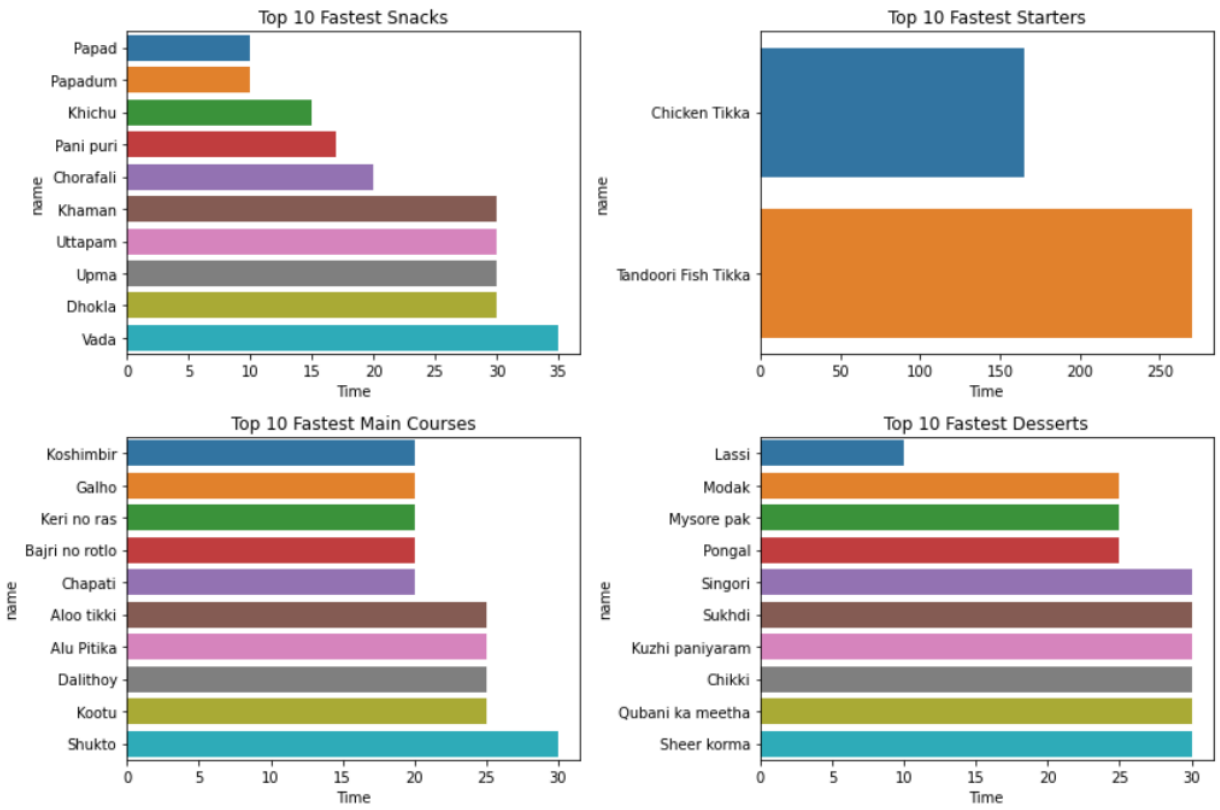
Name: Manjunath Angadi

	name	ingredients	diet	prep_time	cook_time	flavor_profile	course	state	region	Time
0	Balu shahi	[Maidaflour, yogurt, oil, sugar]	vegetarian	45	25	sweet	dessert	West Bengal	East	70
1	Boondi	[Gramflour, ghee, sugar]	vegetarian	80	30	sweet	dessert	Rajasthan	West	110
2	Gajar ka halwa	[Carrots, milk, sugar, ghee, cashews, raisins]	vegetarian	15	60	sweet	dessert	Punjab	North	75
3	Ghevar	[Flour, ghee, kewra, milk, clarifiedbutter, su...	vegetarian	15	30	sweet	dessert	Rajasthan	West	45
4	Gulab jamun	[Milkpowder, plainflour, bakingpowder, ghee, m...	vegetarian	15	40	sweet	dessert	West Bengal	East	55

Now let's check what we get in the plots.

```
snack = df.loc[(df['Time'] != 0) & (df['course'] == 'snack')]
dessert = df.loc[(df['Time'] != 0) & (df['course'] == 'dessert')]
main = df.loc[(df['Time'] != 0) & (df['course'] == 'main course')]
starter = df.loc[(df['Time'] != 0) & (df['course'] == 'starter')]

fast_snacks = snack.sort_values(['Time'], ascending=True)[0:10]
fast_desserts = dessert.sort_values(['Time'], ascending=True)[0:10]
fast_main = main.sort_values(['Time'], ascending=True)[0:10]
fast_starter = starter.sort_values(['Time'], ascending=True)[0:10]

plt.figure(figsize=(12,8))

plt.subplot(2,2,1)
sns.barplot(x=fast_snacks['Time'], y=fast_snacks['name'], orient='h')
plt.title('Top 10 Fastest Snacks')

plt.subplot(2,2,2)
sns.barplot(x=fast_starter['Time'], y=fast_starter['name'], orient='h')
plt.title('Top 10 Fastest Starters')

plt.subplot(2,2,3)
sns.barplot(x=fast_main['Time'], y=fast_main['name'], orient='h')
plt.title('Top 10 Fastest Main Courses')

plt.subplot(2,2,4)
sns.barplot(x=fast_desserts['Time'], y=fast_desserts['name'], orient='h')
plt.title('Top 10 Fastest Desserts')

plt.tight_layout()
plt.show()
```

Name: Manjunath Angadi



As mentioned before the amount of starter courses in the dataset is really low, it seems that there are only two starters in our whole dataset...

Besides that, all the other course categories presented a nice range of short timed options!

I did some research and here is a little list of what I plan to be cooking myself:

- Snacks: Papadum (similar to tortillas) and Pani puri (fried bun filled with vegetables).
- Main Courses: Koshimbir (salad), Galho (found it as Naga Galho, a juicy rice vegetable mix) and Shukto (vegetable curry).
- Desserts: Modak (coconut filled bun), Sukhdi (biscuit).

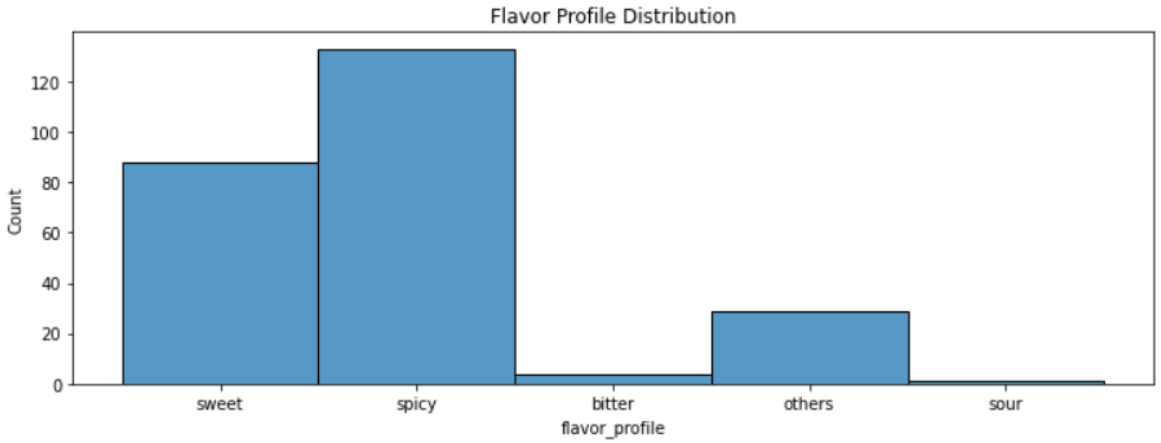
My choices were based on how easy it would be to get the ingredients and how tasty it looked to me, so you may want to check the options by yourself and choose what suits you best.

What Ingredients Define the Flavor Profile

In this last part of the analysis I'll be using the same coding structure as for the ingredients per region section. The only thing I'll change is that I'll be looking at the distribution of flavor profiles before I create our new dataset, to see what we can extract from it prior to our main focus in the analysis.

```
plt.figure(figsize=(12,4))
sns.histplot(df['flavor_profile'])
plt.title('Flavor Profile Distribution')
```

```
plt.show()
```



By looking at the distribution of dishes through the flavor profiles, it gets evident that our analysis will be more fruitful if it's focused on the sweet and spicy flavors, since those are the ones that appear the most in the dataset.

This information by itself indicates that the indian cuisine is mostly filled with spicy and sweet dishes, or that our dataset misrepresents the amount of bitter and sour dishes. Indian food is commonly known for how hot it can get, so my bets are in the former proposition.

Since it would be really easy to, I'll be including the rest of the flavors in the analysis just in case there are any insights I may miss.

Now let's create our ingredients per flavor dataset.

```
Sweet = []
Spicy = []
Bitter = []
Others = []
Sour = []

for row in range(len(df)):
    if df['flavor_profile'][row] == 'sweet':
        Sweet.extend(df['ingredients'][row])
    if df['flavor_profile'][row] == 'spicy':
        Spicy.extend(df['ingredients'][row])
    if df['flavor_profile'][row] == 'bitter':
        Bitter.extend(df['ingredients'][row])
    if df['flavor_profile'][row] == 'others':
        Others.extend(df['ingredients'][row])
    if df['flavor_profile'][row] == 'sour':
        Sour.extend(df['ingredients'][row])
```

Name: Manjunath Angadi

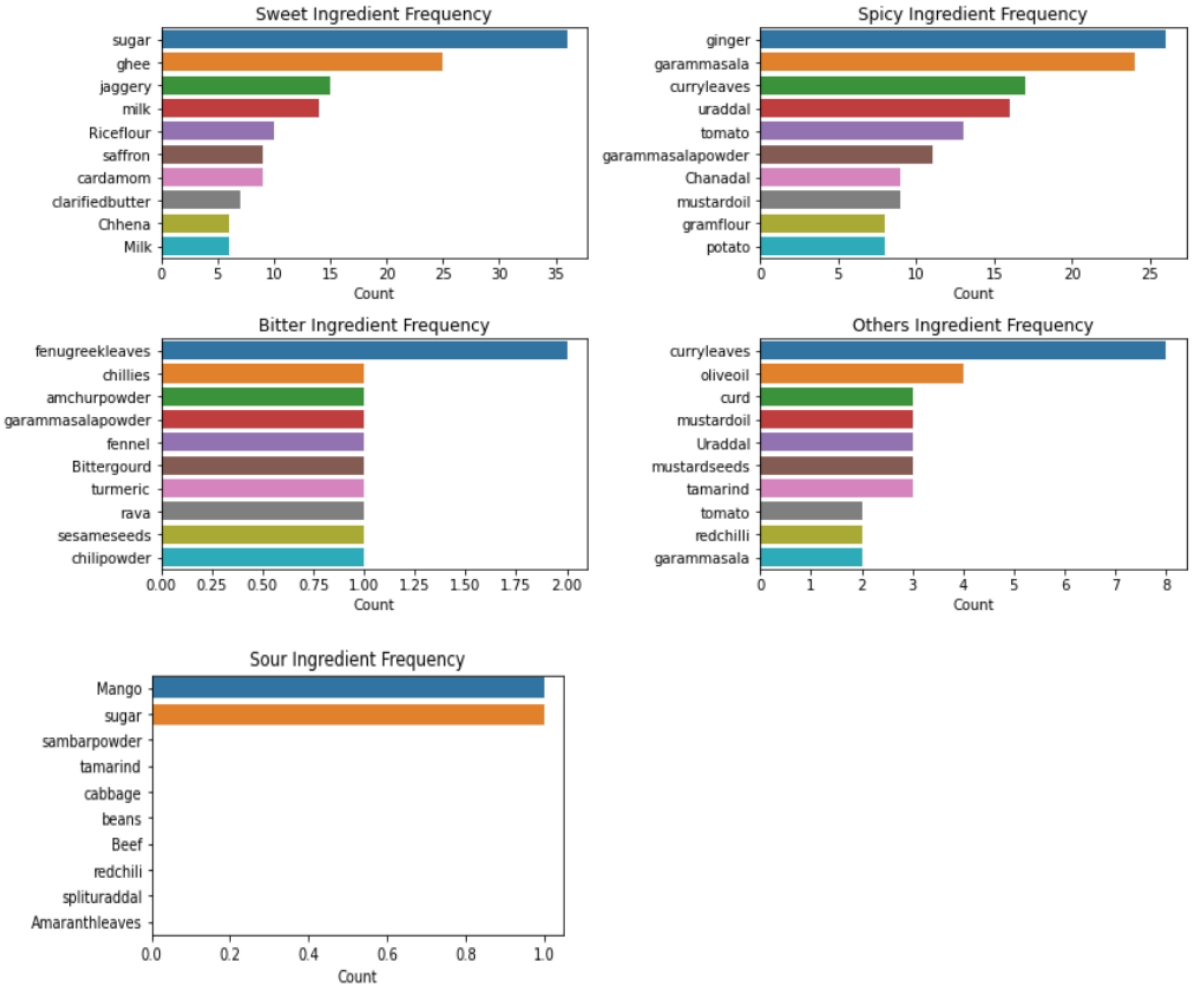
```
all_ing = pd.Series(Sweet+Spicy+Bitter+Others+Sour).unique()

SweetC = []
SpicyC = []
BitterC = []
OthersC = []
SourC = []

for ingredient in all_ing:
    SweetC.append(Sweet.count(ingredient))
    SpicyC.append(Spicy.count(ingredient))
    BitterC.append(Bitter.count(ingredient))
    OthersC.append(Others.count(ingredient))
    SourC.append(Sour.count(ingredient))

ing_per_flavor = pd.DataFrame({'Sweet': SweetC, 'Spicy': SpicyC, 'Bitter':
BitterC, 'Others': OthersC, 'Sour': SourC},
                              index=all_ing)

ing_per_flavor.head()
```

	Sweet	Spicy	Bitter	Others	Sour
Maidaflour	2	0	0	0	0
yogurt	1	3	0	0	0
oil	2	1	1	1	0
sugar	36	6	0	1	1
Gramflour	3	1	0	0	0

With the dataset created it's time for the plots.

```
plt.figure(figsize=(12,12))

for n, flavor in enumerate(ing_per_flavor.columns):
    ordered = ing_per_flavor.sort_values([flavor], ascending=False)[0:10]
    plt.subplot(4, 2, n+1)
    sns.barplot(x=ordered[flavor], y=ordered.index, orient='h')
    plt.xlabel('Count')
    plt.title(f'{flavor} Ingredient Frequency')

plt.tight_layout()
plt.show()
```

Name: Manjunath Angadi



As I thought, the **bitter** and **sour** flavor profiles don't tell us much, mostly because of the frequency that they appear in the dataset.

In the other hand the **sweet** and **spicy** flavors show us some interesting patterns!

The **sweet** profile is, *not surprisingly*, lead by sugar and the third place is a sugar concentrate. In the second position there is ghee, a clarified butter. This three ingredients on the top of our list allow us to imply that if we try an indian dessert it have a high chance of being a buttery sweet dish.

The **spicy** profile is, *surprisingly*, lead by ginger, an ingredient that is not spicy on it's own. Just like the ghee in the sweet profile, that is not sweet, these ingredients tell us about the composition of the dishes instead of what actually grant them that flavor profile. The second and third positions though, are held by garam masala and curry leaves, ingredients that will surely make the dish hotter. I will also mention urad dal, beans, and tomato as they are the first two non-spice ingredients. From this information, we could expect our average spicy indian dish to be a hot tomato sauce filled with flavor and beans.

There are plenty other information that could be extracted from this dataset, but I selected only what I thought would be most interesting to know. So for this analysis this is it!