EDA ON REFUGE AND GLOBAL FOOD PRICE DATA BY CHUKWUEMEKA

July 5, 2021

1 GLOBAL FOOD PRICE PROJECT PROPOSAL

1.1 INTRODUCTION

Global Food Price is described as the average price of food commodities across countries, regions, and the global level. The level of food price usually depends on the food production process which includes food marketing and distribution. Fluctuations in the price of commodities can be multifactorial ranging from availability of natural resources for agriculture, market demand, energy cost, cost of production, exchange rate, government policy, and weather events amongst all. These factors have both positive and negative effects.

Starting from 2007-08 there was a surge in food prices, particularly in developing countries. This led to a global crisis causing political and economic instability as well as social unrest between poor and developed countries. The trend dropped and increased again in 2009 and 2010, reaching new heights in 2011 & 2012. Over the years prices dropped significantly reaching a lower point in March 2016 with a reduced Food and Agricultural Organization(FAO) food price index. The FAO Food Price Index (FFPI) is a measure of the monthly change in international prices of a basket of food commodities. In recent times, global food prices rose in March 2021, which marked the 10th consecutive monthly increase with products like vegetable oil and dairy products leading the rise.

It is needful to say that at this point, the impact of food prices not only provides an indicator of the balance of agricultural produce and market demands but also has an impact on the cost of living, food policies, and migration. While the producers benefit from the high food prices, consumers only benefit when the food prices are low. By implication, food prices now have an impact on food affordability, quality of a diet, undernourishment, and hunger.

In line with the United Nations Development Programme's (UNDP) sustainable development goal 2, i.e., zero hunger we will be taking a closer look at the trends in global food prices, possible causes and effects of increased global food prices and offer our solution.

1.2 PROBLEM STATEMENT

It can be observed from the previous discussion that global food price fluctuations can cause famine and large population shift. Hence, Identifying the drivers of global food prices and predicting future changes in global food prices, could help in understanding food prices and its causal effects.

1.3 AIM

Our research aims to understand and analyze fluctuations in global food prices and pair the outcome with currency fluctuations, weather patterns, and refugee movements. This will help us to build an end-to-end analysis and a food price prediction engine that will help the Government make better decisions on food policy adjustments, International bodies with planning of food aid programmes, Individuals with planning and productivity in the advent of a potential food price crisis...

1.4 OBJECTIVES

To achieve the above aim, we will: Analyze available datasets to observe and make inferences about changing food prices, fluctuations, and the trend they follow. Attempt to compare their correlation with factors such as currency fluctuation, weather patterns, and refugee movements. Investigate which food item controls the trends of the majority of the food markets. Use the best-performed model in predicting food prices and deploying it in a web application that can predict food prices.

1.5 REVIEW OF PAST LITERATURE

Most of the recent research on Global Food Prices has centered around policy-making across nations and countries in addressing the issue. An article by ALNAP, it cited IFPRI/CGIAR, 2008 where it was stated that factors that have contributed to the global food price crisis are either cyclical, structural, or unique. Various World Organizations like WFP, UNOCHA/CERF, UNICEF, IMF, WORLD BANK, NEPAD, ADB, AU, WTO, etc have championed different policies towards mitigating the menace of the Global Food Prices crisis, especially through financial aids. Notable among them are FAO's Procurement and distribution of seeds, fertilizers, and other inputs which have been carried out in 54 countries under the Food and Agriculture Organisation (FAO) Initiative on Soaring Food Prices (ISFP). FAO is also urging governments and the International community to implement measures in support of poor countries hard hit by food price increases, specifically to provide small farmers with improved access to inputs like seeds and fertilizers to increase local crop production (RHVP/Wahenga brief, 2008).

From a micro perspective, Nigeria as a country has had several policies both in present and in the past regarding mitigation of the food prices crisis. Policies like Operation Feed the Nation, Green Revolution, and presently FADAMA programs. These policies and programs have contributed little or none to solving the challenge of the food price crisis.

With regards to predictive modeling technique, Artificial Neural Network(ANN) algorithm and Time Series Forecasting algorithms like ARIMA have been used recently by researchers in this Global Food prices crisis domain. A Machine Learning Approach to Forecasting Consumer Food Prices, J. Jay Harris(2017), applied ANN in modeling Global Food Prices, which was significantly insightful. In this project, we shall also be exploring predictive modeling techniques as well as time series models in forecasting food prices.

1.6 DATA COLLECTION

Data related to global food prices will be collected from the Open source database compiled by the World Food Programme and distributed by the Humanitarian Data Exchange. Data on currency fluctuations will be gotten from the World Bank's open-source database on official exchange rates. Data on Refugee movements will be extracted from the Refugee statistics of the United Nations

High Commissioner for Refugees. Data on Weather patterns will be excerpts from the World Meteorological Organization.

1.7 MACHINE LEARNING WORKFLOW

Data Volumes ↓ Data Ingestion ↓ Data Wrangling ↓ Data Cleaning ↓ Data preprocessing→ Stationarity check→ Time series modeling→forecasting ↓ Predictive modeling

1.8 WEB APPLICATION DEVELOPMENT FOR THE MODEL

The end product of this Global food prediction engine will be in the form of a web app that can be accessed from anywhere as long as there is an Internet connection, It will have a drop-down list to select the food categories, and a graph showing the trend of the price fluctuations over the years and the prediction over the next couple of months. The web app will be built using the streamlit service which makes deploying models quick and easy. The model which would have been worked on and perfected is saved as a pickle file and a python script is created for the usage of the model, then using streamlit, the interface stated above is created in python, then connected and deployed for use.

1.9 References:

"World Food Situation". FAO. Archived from the original on 29 April 2011. Retrieved 24 April 2011. How do Food Prices Affect Producers and Consumers in Developing Countries?, ICTSD, Information Note Number 10, September 2009 UN Food and Agriculture Organization (2009). The State of Food Insecurity in the World 2009. Rome. Rahman, M. Mizanur (11 August 2011). "Food price inflation: Global and national problem". The Daily Star. "FAO Food Price Index". FAO. Retrieved 2 May 2017.

1.10 Group Trailblazers:

- Abiona Oluwafemi
- Roqeebat Olanrewaju
- Omeh Chukwuemeka
- Habeebullah Agbaje

1.11 Terms

1.11.1 Who is a refugee?

Refugees are people who have fled war, violence, conflict or persecution and have crossed an international border to find safety in another country.They often have had to flee with little more than the clothes on their back, leaving behind homes, possessions, jobs and loved ones. Refugees are defined and protected in international law. The 1951 Refugee Convention is a key legal document and defines a refugee as:"someone who is unable or unwilling to return to their country of origin owing to a well-founded fear of being persecuted for reasons of race, religion, nationality, membership of a particular social group, or political opinion." link

A refugee is a person who has fled their own country because they are at risk of serious human rights violations and persecution there. The risks to their safety and life were so great that they felt they had no choice but to leave and seek safety outside their country because their own government cannot or will not protect them from those dangers. Refugees have a right to international protection.

1.11.2 Who is an asylum-seeker?

An asylum-seeker is a person who has left their country and is seeking protection from persecution and serious human rights violations in another country, but who hasn't yet been legally recognized as a refugee and is waiting to receive a decision on their asylum claim. Seeking asylum is a human right. This means everyone should be allowed to enter another country to seek asylum.

1.11.3 Who is a migrant?

There is no internationally accepted legal definition of a migrant. Like most agencies and organizations, we at Amnesty International understand migrants to be people staying outside their country of origin, who are not asylum-seekers or refugees.

Some migrants leave their country because they want to work, study or join family, for example. Others feel they must leave because of poverty, political unrest, gang violence, natural disasters or other serious circumstances that exist there.

2 Import Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
```

3 Setting display layout

[2]:
```
pd.set_option("display.max_column", None)
pd.set_option("display.max_colwidth", None)
pd.set_option("display.max_row", None)
pd.set_option("display.float_format", lambda x: "%.2f" %x)
plt.style.use('ggplot')
plt.rcParams['font.size'] = 10
```

[3]:
```
ref = pd.read_csv("populations_countries.csv",delimiter = ',')
print(f"The number of rows is: {ref.shape[0]} and numbers of columns is: {ref.
 ↪shape[1]}")
```

```
The number of rows is: 90004 and numbers of columns is: 11
```

[4]:
```
gfp = pd.read_csv("world_food_price.csv", sep = "\t", index_col = 'Unnamed: 0')
print(f"The number of rows is: {gfp.shape[0]} and numbers of columns is: {gfp.
 ↪shape[1]}")
```

```
/home/chuxian/anaconda3/lib/python3.8/site-
packages/numpy/lib/arraysetops.py:569: FutureWarning: elementwise comparison
failed; returning scalar instead, but in the future will perform elementwise
comparison
  mask |= (ar1 == a)
```

```
The number of rows is: 1859290 and numbers of columns is: 12
```

[5]:
```
gfp.head()
```

[5]:
```
         date      country         city    market currency    type unit  \
0  2014-01-01  Afghanistan  Badakhshan  Fayzabad       AFN  Retail   KG
1  2014-02-01  Afghanistan  Badakhshan  Fayzabad       AFN  Retail   KG
2  2014-03-01  Afghanistan  Badakhshan  Fayzabad       AFN  Retail   KG
3  2014-04-01  Afghanistan  Badakhshan  Fayzabad       AFN  Retail   KG
4  2014-05-01  Afghanistan  Badakhshan  Fayzabad       AFN  Retail   KG

      month  Year  price product continet
0   January  2014  50.00   Bread     Asia
1  February  2014  50.00   Bread     Asia
2     March  2014  50.00   Bread     Asia
3     April  2014  50.00   Bread     Asia
4       May  2014  50.00   Bread     Asia
```

[6]:
```
gfp.columns
```

[6]:
```
Index(['date', 'country', 'city', 'market', 'currency', 'type', 'unit',
       'month', 'Year', 'price', 'product', 'continet'],
      dtype='object')
```

[7]:
```
gfp.shape
```

[7]:
```
(1859290, 12)
```

[8]:
```
# gfp.rename(columns = {"mp_year": "Year", "mp_price":"price", "mp_month":
 ↪"month"}, inplace = True)
# gfp.head()
```

[9]:
```
# gfp.month = gfp["month"].replace({1:"January", 2:"February",3:"March",4:
 ↪"April",5:"May",
#                    6:"June",7:"July",8:"August",9:"September",10:"October",
#                    11:"November",12:"December"}).reset_index(drop = True)
# gfp.head()
```

[10]:
```
# gfp["continet"] = gfp["country"].map({'Afghanistan': "Asia",
#  'Algeria': "Africa",
#  'Angola': "Africa",
#  'Argentina':"Americas",
#  'Armenia': "Asia",
#  'Azerbaijan': "Asia",
#  'Bangladesh': "Asia",
#  'Bassas da India': "Asia",
#  'Belarus': "Europe",
#  'Benin': "Africa",
#  'Bhutan': "Asia",
#  'Bolivia':"Americas",
#  'Burkina Faso': "Africa",
#  'Burundi': "Africa",
#  'Cambodia': "Asia",
#  'Cameroon': "Africa",
#  'Cape Verde': "Africa",
#  'Central African Republic': "Africa",
#  'Chad': "Africa",
#  'China': "Asia",
#  'Colombia': "Americas",
#  'Congo': "Africa",
#  'Costa Rica':"Americas",
#  "Cote d'Ivoire": "Africa",
#  'Democratic Republic of the Congo': "Africa",
#  'Djibouti': "Africa",
#  'Dominican Republic':"Americas",
#  'Ecuador':"Americas",
#  'Egypt': "Africa",
#  'El Salvador': "Americas",
#  'Eritrea': "Africa",
#  'Ethiopia':"Africa",
#  'Gabon': "Africa",
#  'Gambia': "Africa",
#  'Georgia':"Europe",
#  'Ghana': "Africa",
#  'Guatemala':"Americas",
#  'Guinea': "Africa",
#  'Guinea-Bissau': "Africa",
#  'Haiti': "Americas",
#  'Honduras':"Americas",
#  'Indonesia': "Asia",
#  'Iran  (Islamic Republic of)': "Asia",
#  'Iraq': "Asia",
#  'Japan': "Asia",
#  'Jordan': "Asia",
#  'Kazakhstan': "Asia",
#  'Kenya': "Africa",
#  'Kyrgyzstan': "Asia",
#  "Lao People's Democratic Republic":"Asia",
#  'Lebanon': "Asia",
#  'Lesotho': "Africa",
#  'Liberia': "Africa",
#  'Libya': "Africa",
#  'Madagascar': "Africa",
#  'Malawi': "Africa",
#  'Mali': "Africa",
#  'Mauritania': "Africa",
#  'Mexico': "Americas",
#  'Moldova Republic of':"Europe",
#  'Mongolia': "Asia",
#  'Mozambique': "Africa",
#  'Myanmar': "Asia",
#  'Namibia': "Africa",
#  'Nepal': "Asia",
#  'Nicaragua':"Americas",
#  'Niger': "Africa",
#  'Nigeria': "Africa",
#  'Pakistan': "Asia",
#  'Panama':"Americas",
#  'Paraguay': "Americas",
#  'Peru':"Americas",
#  'Philippines': "Asia",
#  'Russian Federation':"Europe",
#  'Rwanda': "Africa",
#  'Senegal': "Africa",
#  'Sierra Leone': "Africa",
#  'Somalia': "Africa",
#  'South Africa': "Africa",
#  'South Sudan': "Africa",
#  'Sri Lanka': "Asia",
#  'State of Palestine': "Asia",
#  'Sudan': "Africa",
#  'Swaziland': "Africa",
#  'Syrian Arab Republic': "Asia",
#  'Tajikistan': "Asia",
#  'Thailand': "Asia",
#  'Timor-Leste': "Asia",
#  'Togo': "Africa",
#  'Turkey': "Asia",
#  'Uganda': "Africa",
#  'Ukraine': "Europe",
#  'United Republic of Tanzania': "Africa",
#  'Venezuela': "Americas",
#  'Viet Nam': "Asia",
#  'Yemen': "Asia",
#  'Zambia': "Africa",
#  'Zimbabwe': "Africa"})
# gfp.head()
```

[11]:
```
#gfp.to_csv("world_food_price.csv", sep = "\t")
```

[12]:
```
gfp.describe(include = 'all')
```

[12]:
```
              date  country                city           market currency  \
count      1859290  1859290             1859290          1859290  1859290
unique         257       98                 616             3193       84
top     2020-10-01   Rwanda  North/Amajyaruguru National Average      XOF
freq         35222   136993              590998            18005   244565
mean           NaN      NaN                 NaN              NaN      NaN
std            NaN      NaN                 NaN              NaN      NaN
min            NaN      NaN                 NaN              NaN      NaN
25%            NaN      NaN                 NaN              NaN      NaN
50%            NaN      NaN                 NaN              NaN      NaN
75%            NaN      NaN                 NaN              NaN      NaN
max            NaN      NaN                 NaN              NaN      NaN

           type     unit    month        Year       price  product continet
count   1859290  1859290  1859290  1859290.00  1859290.00  1859290  1859290
unique        4      113       12         NaN         NaN      601        4
top      Retail       KG    March         NaN         NaN   Millet   Africa
freq    1692723  1446536   169139         NaN         NaN    58243  1013505
mean        NaN      NaN      NaN     2015.95     6654.93      NaN      NaN
std         NaN      NaN      NaN        4.23   112034.74      NaN      NaN
min         NaN      NaN      NaN     2000.00        0.00      NaN      NaN
25%         NaN      NaN      NaN     2013.00       42.86      NaN      NaN
50%         NaN      NaN      NaN     2017.00      235.50      NaN      NaN
75%         NaN      NaN      NaN     2020.00     1100.00      NaN      NaN
max         NaN      NaN      NaN     2021.00 21777780.00      NaN      NaN
```

[13]:
```
"""
=====No duplicate Entries====
"""
dp_r = ref.duplicated()
print(f"The number of duplicated row and column are: {ref[dp_r].shape[0]} and␣
↪{ref[dp_r].shape[1]} respectively")
ref.duplicated().sum()
```

```
The number of duplicated row and column are: 0 and 11 respectively
```

[13]:
```
0
```

[14]:
```
gfp_d = gfp.duplicated()
print(f"The number of duplicated row and column are: {gfp[gfp_d].shape[0]} and␣
↪{gfp[gfp_d].shape[1]} respectively")
```

```
gfp.duplicated().sum()
```

```
The number of duplicated row and column are: 0 and 12 respectively
```

```
[14]: 0
```

```
[15]: ref.columns, gfp.columns
```

```
[15]: (Index(['Year', 'Country of origin', 'Country of origin (ISO)',
              'Country of asylum', 'Country of asylum (ISO)',
              'Refugees under UNHCR's mandate', 'Asylum-seekers',
              'IDPs of concern to UNHCR', 'Venezuelans displaced abroad',
              'Stateless persons', 'Others of concern'],
             dtype='object'),
       Index(['date', 'country', 'city', 'market', 'currency', 'type', 'unit',
              'month', 'Year', 'price', 'product', 'continet'],
             dtype='object'))
```

```
[16]: ref.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90004 entries, 0 to 90003
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Year                            90004 non-null  int64
 1   Country of origin               90004 non-null  object
 2   Country of origin (ISO)         88989 non-null  object
 3   Country of asylum               90004 non-null  object
 4   Country of asylum (ISO)         90004 non-null  object
 5   Refugees under UNHCR's mandate  90004 non-null  int64
 6   Asylum-seekers                  90004 non-null  int64
 7   IDPs of concern to UNHCR        90004 non-null  int64
 8   Venezuelans displaced abroad    59 non-null     float64
 9   Stateless persons               90004 non-null  int64
 10  Others of concern               90004 non-null  int64
dtypes: float64(1), int64(6), object(4)
memory usage: 7.6+ MB
```

```
[17]: gfp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1859290 entries, 0 to 1859289
Data columns (total 12 columns):
 #   Column    Dtype
---  ------    -----
 0   date      object
 1   country   object
 2   city      object
 3   market    object
 4   currency  object
 5   type      object
 6   unit      object
 7   month     object
 8   Year      int64
 9   price     float64
 10  product   object
 11  continet  object
dtypes: float64(1), int64(1), object(10)
memory usage: 184.4+ MB
```

[18]:

```
ref.dtypes
```

[18]:

```
Year                                   int64
Country of origin                     object
Country of origin (ISO)               object
Country of asylum                     object
Country of asylum (ISO)               object
Refugees under UNHCR's mandate         int64
Asylum-seekers                         int64
IDPs of concern to UNHCR               int64
Venezuelans displaced abroad         float64
Stateless persons                      int64
Others of concern                      int64
dtype: object
```

[19]:

```
gfp.dtypes
```

[19]:

```
date         object
country      object
city         object
market       object
currency     object
type         object
unit         object
month        object
Year          int64
price       float64
product      object
continet     object
dtype: object
```

[20]:

```
ref_obj = ref.select_dtypes(include = 'object')
ref_int = ref.select_dtypes(include = 'int')
ref_float = ref.select_dtypes(include = 'float')
```

```
ref_obj.shape, ref_int.shape, ref_float.shape
```

[20]:
```
((90004, 4), (90004, 6), (90004, 1))
```

[21]:
```
gfp_obj = gfp.select_dtypes(include = 'object')
gfp_int = gfp.select_dtypes(include = 'int')
gfp_float = gfp.select_dtypes(include = 'float')
gfp_obj.shape, gfp_int.shape, gfp_float.shape
```

[21]:
```
((1859290, 10), (1859290, 1), (1859290, 1))
```

[22]:
```
ref.nunique().sort_values(ascending = False)
```

[22]:
```
Refugees under UNHCR's mandate    6996
Asylum-seekers                    3915
Stateless persons                  675
Others of concern                  640
IDPs of concern to UNHCR           454
Country of origin                  212
Country of origin (ISO)            211
Country of asylum                  189
Country of asylum (ISO)            189
Venezuelans displaced abroad        59
Year                                21
dtype: int64
```

[23]:
```
gfp.nunique().sort_values(ascending = False)
```

[23]:
```
price       227112
market        3193
city           616
product        601
date           257
unit           113
country         98
currency        84
Year            22
month           12
type             4
continet         4
dtype: int64
```

[24]:
```
ref.describe(include = 'all')
```

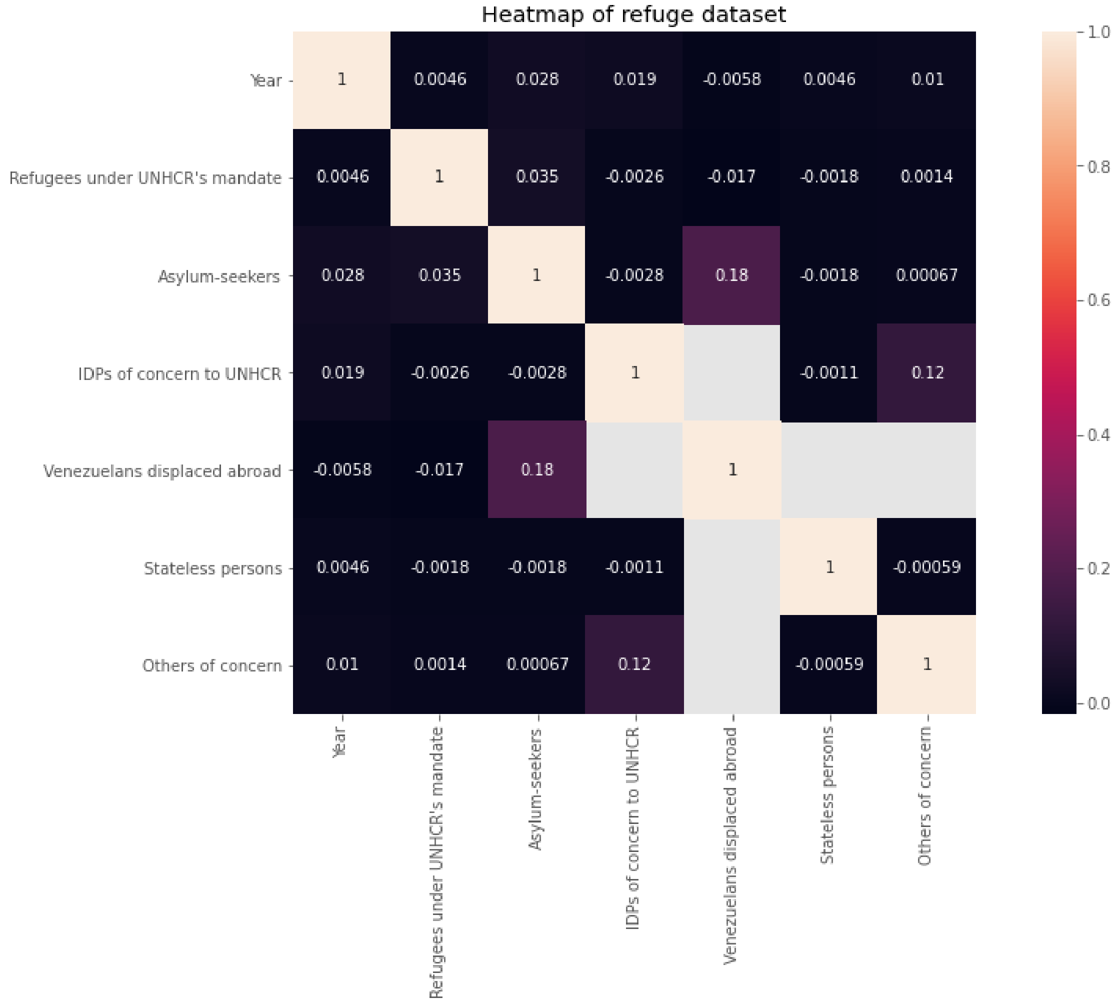
[24]:
```
           Year Country of origin Country of origin (ISO)  \
count  90004.00             90004                   88989
unique      NaN               212                     211
```

```
top         NaN           Somalia                     SOM
freq        NaN              1990                    1990
mean    2011.07               NaN                     NaN
std        6.02               NaN                     NaN
min     2000.00               NaN                     NaN
25%     2006.00               NaN                     NaN
50%     2012.00               NaN                     NaN
75%     2016.00               NaN                     NaN
max     2020.00               NaN                     NaN

               Country of asylum Country of asylum (ISO)  \
count                      90004                   90004
unique                       189                     189
top     United States of America                     USA
freq                        3572                    3572
mean                         NaN                     NaN
std                          NaN                     NaN
min                          NaN                     NaN
25%                          NaN                     NaN
50%                          NaN                     NaN
75%                          NaN                     NaN
max                          NaN                     NaN

        Refugees under UNHCR's mandate  Asylum-seekers  \
count                         90004.00        90004.00
unique                             NaN             NaN
top                                NaN             NaN
freq                               NaN             NaN
mean                           3077.74          393.77
std                           46313.22         5586.88
min                               0.00            0.00
25%                               5.00            0.00
50%                              14.00            6.00
75%                             103.00           41.00
max                         3641370.00       940668.00

        IDPs of concern to UNHCR  Venezuelans displaced abroad  \
count                   90004.00                         59.00
unique                       NaN                           NaN
top                          NaN                           NaN
freq                         NaN                           NaN
mean                     4881.33                     170025.02
std                    124509.91                     358009.88
min                         0.00                         11.00
25%                         0.00                       6568.00
50%                         0.00                      25686.00
75%                         0.00                     142565.50
```

```
max                   8252788.00                    1771237.00

        Stateless persons  Others of concern
count            90004.00           90004.00
unique                NaN                NaN
top                   NaN                NaN
freq                  NaN                NaN
mean               732.52             362.10
std              26468.49           16814.69
min                  0.00               0.00
25%                  0.00               0.00
50%                  0.00               0.00
75%                  0.00               0.00
max            3500000.00         2351313.00
```

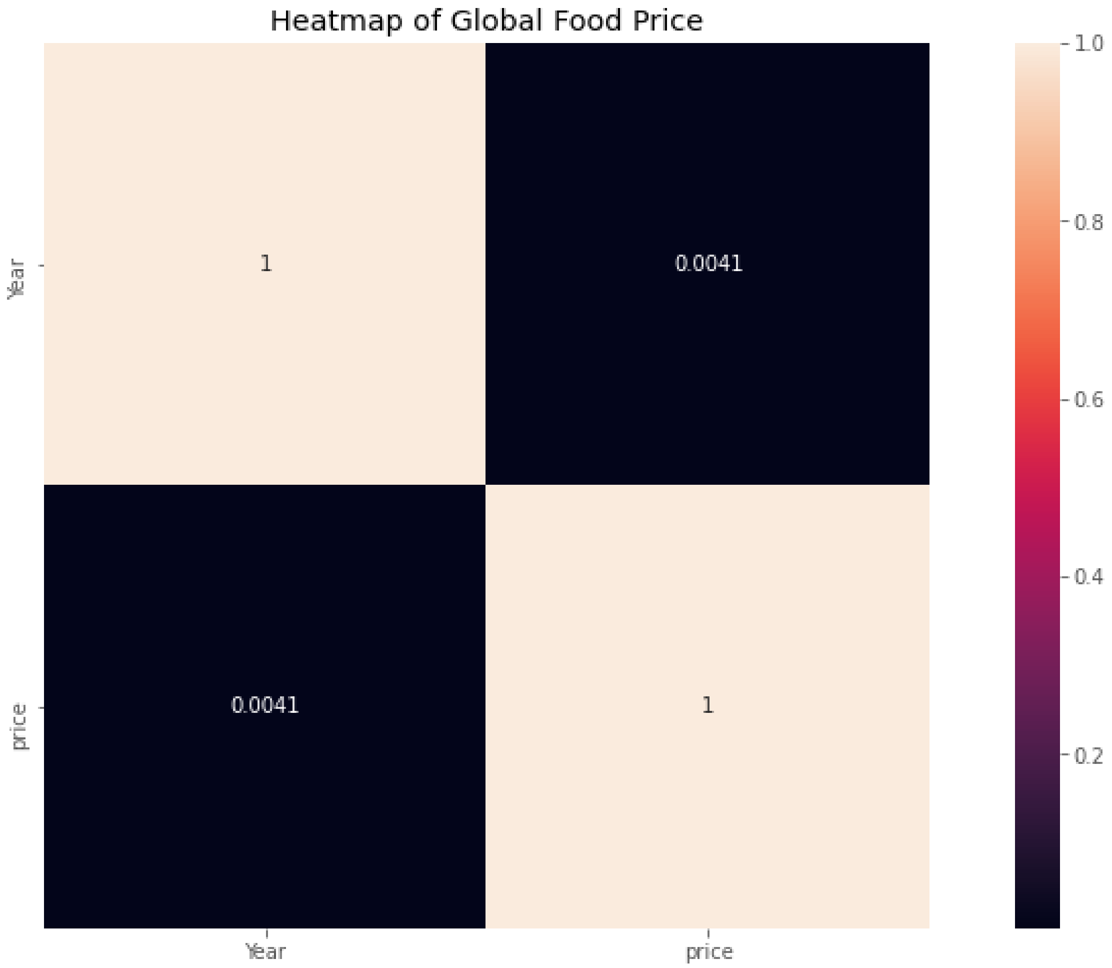
[25]:

```
plt.figure(figsize = (15,8))
ref_corr = ref.corr()
sns.heatmap(ref_corr, annot = True, square = True)
plt.title("Heatmap of refuge dataset")
plt.show()
```



[26]:

```
plt.figure(figsize = (15,8))
gfp_corr = gfp.corr()
sns.heatmap(gfp_corr, annot = True, square = True)
plt.title("Heatmap of Global Food Price")
plt.show()
```



4 Feature Engineering

- To create a continent and sub-region features of the country's of origin or country of asyluumns

[27]:
```
cont = pd.read_csv("countryContinent.csv")
cont.shape
```

[27]:
```
(249, 9)
```

[28]:
```
cont.head()
```

[28]:
```
             country code_2 code_3  country_code     iso_3166_2 continent  \
0        Afghanistan     AF    AFG             4  ISO 3166-2:AF      Asia
1       land Islands     AX    ALA           248  ISO 3166-2:AX    Europe
2            Albania     AL    ALB             8  ISO 3166-2:AL    Europe
3            Algeria     DZ    DZA            12  ISO 3166-2:DZ    Africa
```

```
4  American Samoa     AS    ASM            16  ISO 3166-2:AS   Oceania

        sub_region  region_code  sub_region_code
0    Southern Asia       142.00            34.00
1  Northern Europe       150.00           154.00
2  Southern Europe       150.00            39.00
3  Northern Africa         2.00            15.00
4        Polynesia         9.00            61.00
```

```
[29]: cont.nunique()
```

```
[29]: country            249
      code_2             248
      code_3             249
      country_code       249
      iso_3166_2         249
      continent            5
      sub_region          22
      region_code          5
      sub_region_code     22
      dtype: int64
```

```
[30]: cont.continent.value_counts()
```

```
[30]: Africa      58
      Americas    55
      Asia        51
      Europe      51
      Oceania     25
      Name: continent, dtype: int64
```

```
[31]: cont.sub_region.value_counts()
```

```
[31]: Caribbean                    28
      Eastern Africa               20
      Western Asia                 18
      Western Africa               17
      Northern Europe              16
      Southern Europe              16
      South America                14
      South-Eastern Asia           11
      Eastern Europe               10
      Polynesia                    10
      Middle Africa                 9
      Western Europe                9
      Southern Asia                 9
      Eastern Asia                  8
```

```
Central America              8
Northern Africa              7
Micronesia                   7
Melanesia                    5
Southern Africa              5
Northern America             5
Central Asia                 5
Australia and New Zealand    3
Name: sub_region, dtype: int64
```

[32]:

```
ref["continent"] = ref["Country of origin"].map({'Afghanistan': 'Asia',
 'Iraq': 'Asia',
 'Serbia and Kosovo: S/RES/1244 (1999)': "Europe",
 'Turkey': 'Asia',
 'Chad': 'Africa',
 'Cameroon': 'Africa',
 'Congo': 'Africa',
 'Dem. Rep. of the Congo': 'Africa',
 'Palestinian': 'Asia',
 'Guinea': 'Africa',
 'Liberia': 'Africa',
 'Libya': 'Africa',
 'Mali': 'Africa',
 'Morocco': 'Africa',
 'Nigeria': 'Africa',
 'Rwanda': 'Africa',
 'Sierra Leone': 'Africa',
 'Somalia': 'Africa',
 'Sudan': 'Africa',
 'Syrian Arab Rep.': "Asia",
 'Western Sahara': 'Africa',
 'Unknown ':"Unknown Continent",
 'Angola': 'Africa',
 'Burundi': 'Africa',
 'Comoros': 'Africa',
 'Guinea-Bissau': 'Africa',
 'United Rep. of Tanzania': 'Africa',
 'Zambia': 'Africa',
 'Djibouti': 'Africa',
 'Eritrea': 'Africa',
 'Ethiopia': 'Africa',
 'Russian Federation': "Europe",
 'Yemen': "Asia",
 'Stateless': "Asia",
 'Albania': "Europe",
 'Algeria': 'Africa',
 'Armenia': 'Asia',
 'Benin': 'Africa',
 'Bangladesh': 'Asia',
 'Bosnia and Herzegovina': "Europe",
 'Bulgaria': "Europe",
 'Chile': "Americas",
 'Colombia': "Americas",
 'Cuba': "Americas",
 'Dominican Rep.': "Americas",
 'Ecuador': "Americas",
 'Estonia': "Europe",
 'Georgia':'Europe',
 'Ghana': 'Africa',
 'Haiti':'Asia',
 'India':'Asia',
 'Iran (Islamic Rep. of)': 'Asia',
 'Kazakhstan':"Asia",
 'Kyrgyzstan': "Asia",
 "Lao People's Dem. Rep.": "Asia",
 'Lebanon': "Asia",
 'Sri Lanka': 'Asia',
 'Nicaragua': "Americas",
 'Pakistan': 'Asia',
 'Paraguay': "Americas",
 'Peru': "Americas",
 'Romania': "Europe",
 'Senegal': 'Africa',
 'Viet Nam': "Asia",
 'Tunisia': 'Africa',
 'Ukraine': "Europe",
 'Azerbaijan': 'Asia',
 'Egypt': 'Africa',
 'Argentina': "Americas",
 'Austria': "Europe",
 'Bahrain': 'Asia',
 'Belarus': "Europe",
 'Bolivia (Plurinational State of)': "Americas",
 'Brazil': "Americas",
 'Cambodia': 'Asia',
 'China': 'Asia',
 'Cyprus': 'Asia',
 'Czechia':"Europe",
 'Fiji': "Oceania",
 'France': "Europe",
 'United Kingdom of Great Britain and Northern Ireland': "Europe",
 'Germany': "Europe",
 'Guatemala': "Americas",
 'China': 'Asia',
 'Hong Kong SAR': 'Asia',
 'Croatia': "Europe",
 'Hungary': "Europe",
 'Indonesia': 'Asia',
 'Israel': 'Asia',
 'Italy': "Europe",
 'Jordan': 'Asia',
 'Kenya': 'Africa',
 'Rep. of Korea': 'Asia',
 "Dem. People's Rep. of Korea": 'Asia',
 'Kuwait': "Asia",
 'Lithuania':"Europe",
 'Latvia':"Europe",
 'North Macedonia':"Europe",
 'Rep. of Moldova':"Europe",
 'Mexico': "Americas",
 'Malaysia': 'Asia',
 'Mongolia': 'Asia',
 'Mauritius':"Africa",
 'Myanmar': 'Asia',
 'Nepal': 'Asia',
 'Niger': 'Africa',
 'Philippines': 'Asia',
 'Poland': "Europe",
 'Portugal': "Europe",
 'South Africa': 'Africa',
 'El Salvador': "Americas",
 'Saudi Arabia': 'Asia',
 'Singapore': 'Asia',
 'Solomon Islands':"Oceania",
 'Slovenia':"Europe",
 'Thailand': "Asia",
 'Timor-Leste': "Asia",
 'Tonga':"Oceania",
 'United Arab Emirates': 'Asia',
 'Uganda': 'Africa',
 'Uruguay': "Americas",
 'Uzbekistan':"Asia",
 'Zimbabwe': 'Africa',
 "Cote d'Ivoire": 'Africa',
 'Tajikistan':"Asia",
 'Togo': 'Africa',
 'Bhutan':"Asia",
 'Burkina Faso': 'Africa',
 'Mauritania': "Africa",
 'Central African Rep.': 'Africa',
 'Equatorial Guinea': 'Africa',
 'Madagascar': 'Africa',
 'Namibia': 'Africa',
 'China, Macao SAR': "Asia",
 'Honduras':"Americas",
 'Antigua and Barbuda':"Americas",
 'Barbados':"Americas",
 'Belgium': "Europe",
 'Botswana': 'Africa',
 'Costa Rica':"Americas",
 'Denmark': "Europe",
 'Dominica':"Americas",
 'Gabon': 'Africa',
 'Gambia': 'Africa',
 'Greece':"Europe",
 'Grenada':"Americas",
 'Guyana':"Americas",
 'Iceland': "Europe",
 'Ireland': "Europe",
 'Jamaica':"Americas",
 'Japan': 'Asia',
 'Saint Lucia':"Americas",
 'Malawi': "Africa",
 'Mozambique': 'Africa',
 'Malta':"Europe",
 'Netherlands':"Europe",
 'Oman': "Asia",
 'Panama':"Americas",
 'Qatar': 'Asia',
 'Spain': "Europe",
 'Slovakia':"Europe",
 'Eswatini':"Africa",
 'Sweden': "Europe",
 'Switzerland': "Europe",
 'Trinidad and Tobago': "Americas",
 'United States of America': "Americas",
 'Saint Vincent and the Grenadines': "Americas",
 'Venezuela (Bolivarian Republic of)': "Americas",
 'Tibetan': "Asia",
 'Turkmenistan': "Asia",
 'Seychelles': 'Africa',
 'Sao Tome and Principe': 'Africa',
 'Papua New Guinea': "Oceania",
 'Suriname': "Americas",
 'Tuvalu':"Oceania",
 'Canada':"Americas",
 'Belize':"Americas",
 'Australia':"Oceania",
 'Bahamas':"Americas",
 'Cabo Verde': "Africa",
 'Finland': "Europe",
 'Nauru':"Oceania",
 'San Marino':"Europe",
 'Saint Kitts and Nevis':"Americas",
 'Samoa':"Oceania",
 'Lesotho': 'Africa',
 'Andorra':"Europe",
 'New Zealand': "Europe",
 'Norway': "Europe",
 'Micronesia (Federated States of)':"Oceania",
 'Gibraltar':"Europe",
 'Turks and Caicos Islands':"Americas",
 'Kiribati':"Oceania",
 'Maldives': "Asia",
 'Bermuda':"Americas",
 'Brunei Darussalam': "Asia",
 'New Caledonia':"Oceania",
 'Monaco':"Europe",
 'Montenegro':"Europe",
 'Holy See':"Europe",
 'South Sudan': "Africa",
 'Niue':"Oceania",
 'Palau':"Oceania",
 'Cayman Islands':"Americas",
 'Marshall Islands':"Oceania",
 'Curacao ':"Americas",
 'Guadeloupe':"Americas",
 'Vanuatu':"Oceania",
 'French Guiana':"Americas",
 'Luxembourg':"Europe",
 'Liechtenstein':"Europe",
 'Anguilla':"Americas",
 'Martinique':"Americas"})
ref["continent"].head()
```

[32]:

```
0      Asia
1      Asia
2    Europe
3      Asia
4    Africa
Name: continent, dtype: object
```

[33]:

```
ref.head()
```

[33]:

```
   Year                     Country of origin Country of origin (ISO)  \
0  2000                           Afghanistan                     AFG
1  2000                                  Iraq                     IRQ
2  2000  Serbia and Kosovo: S/RES/1244 (1999)                     SRB
3  2000                                Turkey                     TUR
4  2000                                  Chad                     TCD

  Country of asylum Country of asylum (ISO)  Refugees under UNHCR's mandate  \
0       Afghanistan                     AFG                               0
1           Albania                     ALB                               9
2           Albania                     ALB                             507
3           Albania                     ALB                               5
4           Algeria                     DZA                              20

   Asylum-seekers  IDPs of concern to UNHCR  Venezuelans displaced abroad  \
0               0                    758625                           NaN
1               0                         0                           NaN
2               5                         0                           NaN
3               0                         0                           NaN
4              19                         0                           NaN

   Stateless persons  Others of concern continent
0                  0                  0      Asia
1                  0                  0      Asia
2                  0                  0    Europe
3                  0                  0      Asia
4                  0                  0    Africa
```

5 Handling Missing Values

[34]:

```python
print(ref.isnull().sum(), "\t", gfp.isnull().sum())
```

```
Year                                  0
Country of origin                     0
Country of origin (ISO)            1015
Country of asylum                     0
Country of asylum (ISO)               0
Refugees under UNHCR's mandate        0
Asylum-seekers                        0
IDPs of concern to UNHCR              0
Venezuelans displaced abroad      89945
Stateless persons                     0
Others of concern                     0
continent                            61
dtype: int64     date        0
country     0
```

```
city        0
market      0
currency    0
type        0
unit        0
month       0
Year        0
price       0
product     0
continet    0
dtype: int64
```

[35]:
```
print(ref.notnull().sum(), "\t", gfp.notnull().sum())
```

```
Year                                 90004
Country of origin                    90004
Country of origin (ISO)              88989
Country of asylum                    90004
Country of asylum (ISO)              90004
Refugees under UNHCR's mandate       90004
Asylum-seekers                       90004
IDPs of concern to UNHCR             90004
Venezuelans displaced abroad            59
Stateless persons                    90004
Others of concern                    90004
continent                            89943
dtype: int64     date        1859290
country     1859290
city        1859290
market      1859290
currency    1859290
type        1859290
unit        1859290
month       1859290
Year        1859290
price       1859290
product     1859290
continet    1859290
dtype: int64
```

[36]:
```
ref["Country of origin"].fillna(value = ref["Country of origin"].mode(),␣
 ↪inplace = True)
ref.isnull().sum()
```

[36]:
```
Year                                 0
Country of origin                    0
Country of origin (ISO)           1015
Country of asylum                    0
Country of asylum (ISO)              0
Refugees under UNHCR's mandate       0
Asylum-seekers                       0
IDPs of concern to UNHCR             0
Venezuelans displaced abroad     89945
Stateless persons                    0
Others of concern                    0
continent                           61
dtype: int64
```

```
[37]: ref["continent"].value_counts()
```

```
[37]: Africa               38918
      Asia                 30089
      Europe               11467
      Americas              8063
      Unknown Continent     1015
      Oceania                391
      Name: continent, dtype: int64
```

```
[38]: ref["continent"].value_counts(normalize = True)*100
```

```
[38]: Africa              43.27
      Asia                33.45
      Europe              12.75
      Americas             8.96
      Unknown Continent    1.13
      Oceania              0.43
      Name: continent, dtype: float64
```

```
[39]: ref.groupby("continent")["Country of origin"].value_counts().head()
```

```
[39]: continent  Country of origin
      Africa     Somalia                      1990
                 Dem. Rep. of the Congo       1892
                 Sudan                        1813
                 Ethiopia                     1587
                 Nigeria                      1455
      Name: Country of origin, dtype: int64
```

```
[40]: ref.continent.unique()
```

```
[40]: array(['Asia', 'Europe', 'Africa', 'Unknown Continent', 'Americas',
             'Oceania', nan], dtype=object)
```

6 Exploring column by column

6.1 Year

- The years range from 2000 - 2020
- The year 2020 has highest frequency
- The year 2000 has the least frequency

```
[41]: ref.Year.describe(include = 'all')
```

```
[41]: count    90004.00
      mean      2011.07
      std          6.02
      min       2000.00
      25%       2006.00
      50%       2012.00
      75%       2016.00
      max       2020.00
      Name: Year, dtype: float64
```

```
[42]: gfp.Year.describe(include = 'all')
```

```
[42]: count    1859290.00
      mean        2015.95
      std            4.23
      min         2000.00
      25%         2013.00
      50%         2017.00
      75%         2020.00
      max         2021.00
      Name: Year, dtype: float64
```

```
[43]: ref.Year.unique()
```

```
[43]: array([2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
             2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020])
```

```
[44]: gfp.Year.unique()
```

```
[44]: array([2014, 2015, 2016, 2017, 2018, 2019, 2020, 2003, 2004, 2005, 2006,
             2007, 2008, 2009, 2010, 2011, 2012, 2013, 2000, 2001, 2002, 2021])
```

```
[45]: ref.Year.value_counts()
```

```
[45]: 2020    6024
      2019    5438
      2018    5263
      2017    5120
```

```
2015    4933
2016    4923
2014    4744
2013    4561
2012    4356
2010    4212
2011    4208
2008    4001
2009    4000
2007    3937
2006    3789
2005    3640
2004    3586
2003    3570
2002    3403
2001    3259
2000    3037
Name: Year, dtype: int64
```

```
[46]: gfp.Year.value_counts()
```

```
[46]: 2020    368153
      2019    190030
      2018    172422
      2017    162502
      2016    139648
      2015    129724
      2014    115283
      2021    111779
      2013    104448
      2012     85009
      2011     64175
      2010     47032
      2009     42971
      2008     35891
      2007     26017
      2006     19004
      2005     13545
      2004      9484
      2003      8520
      2002      5967
      2001      4087
      2000      3599
      Name: Year, dtype: int64
```

7 Exploring the Global Food Price Data

7.1 Categorical Variable

7.1.1 Country

- There are 98 different countries captured
- In Africa continent, Rwanda has the highest frequency and it has 7.37% of whole countries counts
- In Asia Continent, Bassas da India has the highest frequency
- In Europe continent, Ukraine has the highest frequency
- In Americas continent, Columbia has the highest frequency

7.1.2 City

- The city represent the various cities in the countries that are captured in this data
- There are a total of 616 different cities captured in this dataset
- The city of North/Amajyaruguru in Bassas da India is highest occured city
- The city of North/Amajyaruguru in Bassas da India has 31.79% of whole cities counts in the dataset

7.1.3 Market

- The market is markets in the cities captured in this dataset
- There are a total of 3193 differnt markets captured in this dataset
- Various countries National Average Market occured most in the dataset
- National Average market

7.1.4 Currency

- The currency represents the currency used in the markets or that serves as medium of exchange
- There are a total of 84 unique/different currencies in this dataset
- Countries using XOF(Frac) as their currency are most occured
- XOF is 13.15% of all currency counts
- Countries using XOF are mainly African countries of Benin, Burkina Faso, Cote d'Ivoire, Guinea-Bissau, Mali, Niger, Senegal and Togo

7.1.5 Type

- The type represents the type of markets or shop or even the kind of purchases customers make in these market.
- There are 4 unique market types in this dataset
- Retail market type are most occured with a frequency of 1692723
- Retail market type has 91.07% of all type counts

7.1.6 Unit

- The unit represent the quantity that are sold in these various market types. The units corresponds to the price. For instance if am to buy 100KG of Rice, its price won't be same if i'm buying only 50KG. Also, the type(Retail, Wholesale, Producer or Farm-market) will also determine how the product will be priced. It will be relatively cheap if bought directly from the farm-market or producer compared to going to a retail or wholesale shop.

- There are 114 unique units captured and products that are measured in just KG are the most ocuured.
- KG has 77.8% of whole units counts in the dataset.

7.1.7 Months

- There are 12 unique months captured here.
- Month of March is the highest.

7.1.8 Product

- There 601 unique products in this dataset
- Millet is the product with highest frequency

7.1.9 Continent

- There 4 continents captured in this data
- Africa continent has the highest frequency

```
[47]: gfp.head()
```

```
[47]:          date      country        city    market currency    type unit  \
      0  2014-01-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      1  2014-02-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      2  2014-03-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      3  2014-04-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      4  2014-05-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG

            month  Year  price product continet
      0   January  2014  50.00   Bread     Asia
      1  February  2014  50.00   Bread     Asia
      2     March  2014  50.00   Bread     Asia
      3     April  2014  50.00   Bread     Asia
      4       May  2014  50.00   Bread     Asia
```

```
[48]: gfp_obj.columns
```

```
[48]: Index(['date', 'country', 'city', 'market', 'currency', 'type', 'unit',
             'month', 'product', 'continet'],
            dtype='object')
```

```
[49]: gfp_obj.groupby("continet").country.value_counts()
```

```
[49]: continet  country
      Africa    Rwanda                                  136993
                Mali                                     67018
```

	Burundi	55591
	Gambia	51309
	Niger	48475
	Nigeria	47551
	Democratic Republic of the Congo	43726
	Zambia	41845
	United Republic of Tanzania	41590
	Mozambique	40464
	Libya	36765
	Benin	34802
	Burkina Faso	33213
	Senegal	33044
	Ghana	23961
	Malawi	22682
	Ethiopia	22634
	Cameroon	21669
	Chad	18318
	Somalia	16193
	Guinea-Bissau	14739
	South Sudan	13676
	Central African Republic	12378
	Sierra Leone	11138
	Guinea	11098
	Mauritania	10530
	Sudan	9758
	Liberia	9528
	Lesotho	9124
	Madagascar	7792
	Kenya	7760
	Cote d'Ivoire	7737
	Uganda	7517
	Namibia	7050
	Zimbabwe	6796
	Togo	5537
	Djibouti	5374
	Congo	5257
	Swaziland	4247
	Egypt	2247
	Cape Verde	2102
	Algeria	1633
	Angola	1272
	South Africa	768
	Gabon	504
	Eritrea	100
Americas	Colombia	23411
	Bolivia	17064
	Haiti	12540

	Nicaragua	8382
	El Salvador	7648
	Mexico	3895
	Guatemala	3741
	Ecuador	3662
	Peru	3553
	Dominican Republic	2367
	Panama	2176
	Honduras	1617
	Argentina	972
	Paraguay	723
	Costa Rica	297
	Venezuela	6
Asia	Bassas da India	125815
	Syrian Arab Republic	87445
	Philippines	77251
	Indonesia	72353
	Kyrgyzstan	55250
	Lebanon	38000
	Yemen	28551
	Lao People's Democratic Republic	27013
	Tajikistan	25648
	Myanmar	22093
	State of Palestine	21384
	Iraq	19890
	Jordan	19869
	Cambodia	19122
	Armenia	18263
	Nepal	16572
	Afghanistan	10521
	Turkey	9012
	Pakistan	7809
	Bangladesh	7166
	Sri Lanka	4522
	Mongolia	3640
	Kazakhstan	3365
	Timor-Leste	1639
	Japan	1372
	China	1312
	Thailand	849
	Iran (Islamic Republic of)	470
	Bhutan	344
	Viet Nam	257
	Azerbaijan	125
Europe	Ukraine	24431
	Russian Federation	1080
	Moldova Republic of	777

```
          Belarus                                  441
          Georgia                                   80
Name: country, dtype: int64
```

[50]:
```
gfp_obj.describe(include = "all")
```

[50]:
```
               date  country                 city            market  currency  \
count       1859290  1859290              1859290           1859290   1859290
unique          257       98                  616              3193        84
top      2020-10-01   Rwanda  North/Amajyaruguru  National Average       XOF
freq          35222   136993               590998             18005    244565

           type     unit    month  product  continet
count   1859290  1859290  1859290  1859290   1859290
unique        4      113       12      601         4
top      Retail       KG    March   Millet    Africa
freq    1692723  1446536   169139    58243   1013505
```

7.2 Numerical Variables

Year and Price are the only numerical features in the dataset

7.2.1 Year

- There are 22 unique years in the dataset
- The year 2020 is most occured with a frequency of 368153 which amount to 19.8% percent of whole year counts

7.2.2 Price

- The average price of products is 6654.93 irrespective of currency

[51]:
```
gfp_int.columns, gfp_float.columns
```

[51]:
```
(Index(['Year'], dtype='object'), Index(['price'], dtype='object'))
```

[52]:
```
gfp_int.describe(include = "all")
```

[52]:
```
            Year
count 1859290.00
mean     2015.95
std         4.23
min      2000.00
25%      2013.00
50%      2017.00
75%      2020.00
max      2021.00
```

[53]:
```
gfp_int.Year.unique()
```

[53]:
```
array([2014, 2015, 2016, 2017, 2018, 2019, 2020, 2003, 2004, 2005, 2006,
       2007, 2008, 2009, 2010, 2011, 2012, 2013, 2000, 2001, 2002, 2021])
```

[54]:
```
gfp_int.Year.value_counts(normalize = True)*100
```

[54]:
```
2020   19.80
2019   10.22
2018    9.27
2017    8.74
2016    7.51
2015    6.98
2014    6.20
2021    6.01
2013    5.62
2012    4.57
2011    3.45
2010    2.53
2009    2.31
2008    1.93
2007    1.40
2006    1.02
2005    0.73
2004    0.51
2003    0.46
2002    0.32
2001    0.22
2000    0.19
Name: Year, dtype: float64
```

[55]:
```
gfp.groupby("currency").price.describe(include = "all")
```

[55]:
```
                        count      mean        std    min      25%      50%  \
currency
AFN                  10521.00  31799.60  268814.20   2.77    23.00    35.00
AMD                  18263.00    853.26     813.69  38.20   300.00   499.60
AOA                   1272.00   1038.96    1876.20  35.20   231.46   403.25
ARS                    972.00     24.30      42.35   0.17     0.73     6.00
AZN                    125.00      0.53       0.17   0.24     0.40     0.49
BDT                   7166.00    338.98     848.78  14.00    32.91    52.80
BIF                  55591.00   2462.96    4301.46   1.00   679.25  1077.67
BOB                  17064.00     88.22     169.52   0.33     5.11     9.90
BTN                    344.00     51.25      19.93  12.55    34.55    53.44
BYR                    441.00      1.02       0.80   0.10     0.39     0.82
CDF                  43726.00   2574.49    3533.06  33.30   667.00  1225.00
CNY                   1312.00      3.34       1.03   1.30     2.41     3.40
```

COP	23411.00	54610.77	372093.81	104.00	1216.29	2080.00
CVE	2102.00	87.17	50.52	27.41	57.97	74.83
DJF	5374.00	876.76	2261.38	22.50	120.00	140.00
DOP	2367.00	980.65	1274.98	6.30	24.35	49.19
DZD	1633.00	177.81	144.62	4.00	83.00	140.00
EGP	2247.00	19.07	26.60	0.66	4.80	10.06
ERN	100.00	2272.00	864.22	900.00	1587.50	2200.00
ETB	22634.00	1130.71	2319.36	0.77	10.00	304.00
GEL	80.00	25.60	20.81	1.30	1.50	33.50
GHS	23961.00	105.79	140.12	0.32	7.00	55.08
GMD	51309.00	55.43	60.39	0.00	17.00	32.00
GNF	11098.00	10691.28	8441.18	500.00	5333.00	8000.00
GTQ	3741.00	87.29	113.78	0.31	5.44	17.18
HNL	962.00	703.00	355.85	210.00	388.06	720.62
HTG	12540.00	133.37	136.60	6.75	30.00	82.70
IDR	72353.00	34494.19	31266.45	1630.65	13750.00	23948.28
INR	125815.00	95.02	273.69	2.50	24.00	45.00
IQD	19890.00	2601.57	3013.99	75.00	900.00	1500.00
IRR	470.00	78136.56	65879.74	2990.00	41926.75	58555.00
JOD	19869.00	2.06	2.36	0.00	0.70	1.20
JPY	1372.00	637.52	772.87	111.00	201.00	249.00
KES	7760.00	1348.36	2048.02	5.00	45.00	87.00
KGS	55250.00	94.03	98.44	1.00	28.25	55.20
KHR	19122.00	6788.44	8022.53	400.00	1892.98	3416.50
KZT	3365.00	379.61	458.83	34.00	97.00	160.00
LAK	27013.00	24979.33	22103.06	0.00	6917.00	16000.00
LBP	38000.00	3937.48	5328.40	21.66	1516.67	2250.00
LKR	4522.00	119.37	149.40	11.50	66.23	84.94
LRD	9528.00	1496.57	1865.64	5.00	200.00	550.00
LSL	9124.00	33.03	30.30	5.00	9.54	14.24
LYD	36765.00	5.34	7.31	0.01	2.00	3.00
MDL	777.00	5.32	2.12	1.65	3.88	5.00
MGA	7792.00	2449.32	1166.05	220.00	1520.00	2200.00
MMK	22093.00	1046.36	801.55	50.00	424.24	772.00
MNT	3640.00	3115.61	2755.69	200.00	974.00	1538.00
MRO	10530.00	2068.33	14425.67	8.50	178.00	250.00
MWK	22682.00	271.99	264.85	6.45	61.32	181.08
MXN	3895.00	11.08	7.07	1.17	5.13	10.00
MZN	40464.00	42.87	28.83	1.01	24.20	37.50
NAD	7050.00	10.58	7.53	0.70	6.00	10.00
NGN	47551.00	5207.50	8999.00	5.00	200.00	438.00
NIO	3358.00	680.62	439.98	83.08	342.25	534.72
NIS	21384.00	22.73	34.24	0.65	4.00	7.83
NPR	16572.00	102.07	109.36	6.00	35.00	60.00
PAB	2002.00	25.04	31.65	0.17	0.72	1.12
PEN	3553.00	3.18	1.91	0.30	1.77	2.64
PHP	77251.00	96.53	90.46	1.60	35.84	65.00

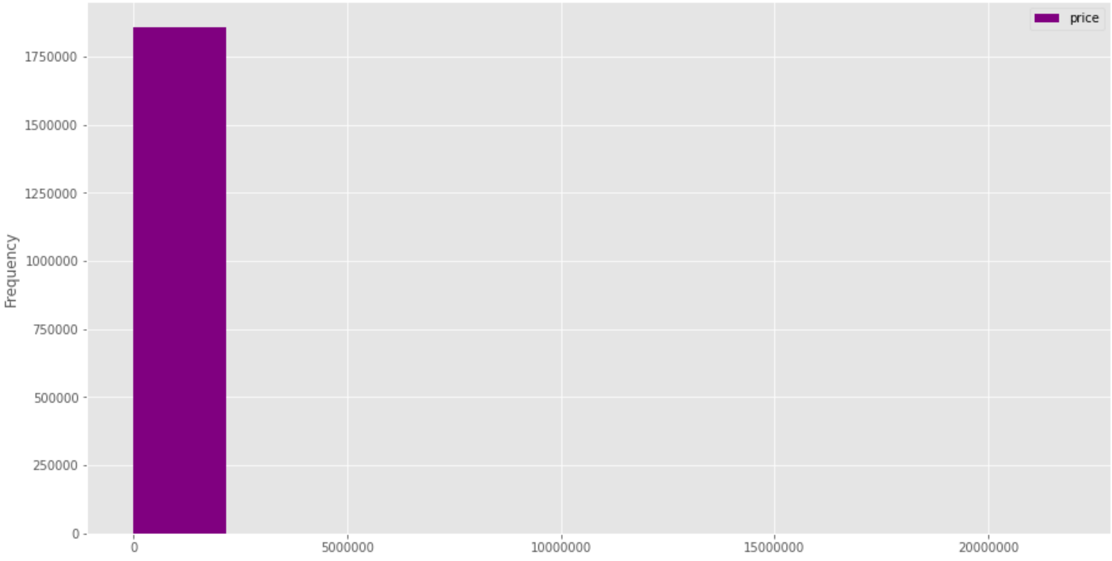
```
PKR                   7809.00     93.72      70.06    9.00    37.00    70.00
PYG                    723.00   8112.41    7950.50  502.00  3167.00  4000.00
RUB                   1080.00     13.83       3.75    6.30    10.47    14.00
RWF                 136993.00   1727.65    9833.43   11.00   238.50   400.00
SDG                   9758.00    343.85    1230.71    0.50     5.00    32.50
SLL                  11138.00  12216.67   13626.65  100.00  4166.67  7500.00
SOS                  16193.00 126360.52 1052608.24  500.00  5700.00 13000.00
SSP                  13676.00   3402.81   14924.25    1.80    28.25   350.00
SYP                  87445.00  20522.82   87116.28    0.00   241.67   528.17
SZL                   4247.00     21.14      34.81    3.84     8.99    11.60
THB                    849.00     15.82      15.06    4.00     7.64    10.24
TJS                  25648.00      7.28       9.20    0.10     1.74     3.50
TRY                   9012.00     16.52      22.47    0.25     3.65     6.61
TZS                  41590.00  30072.61   55294.34  183.80  1325.02  2500.00
UAH                  24431.00     26.46      28.93    1.39     8.15    12.95
UGX                   7517.00   1889.55    1393.44  203.75  1000.00  1500.00
USD                  23388.00     53.29     194.03    0.09     0.49     1.52
VEF                      6.00   5471.00    2094.61 3139.10  3843.68  5289.23
VND                    257.00   7334.50    1021.12 4514.29  6770.00  7322.20
XAF                  58126.00   7003.31   14563.20    5.00   250.00   620.00
XOF                 244565.00   1541.86   10135.87    1.03   188.00   268.00
YER                  28551.00   2774.27   11444.89   24.00   220.00   367.00
ZAR                    768.00      2.20       1.17    0.50     1.21     1.96
ZMW                  41845.00      7.97       7.64    0.01     2.15     4.25
ZWL                   2507.00    220.35     350.39   12.54    62.92   107.50

                         75%         max
currency
AFN                    58.00  5833333.00
AMD                  1049.70     4000.00
AOA                   855.30    14280.95
ARS                    31.25      355.55
AZN                     0.68        1.00
BDT                    83.41     4925.00
BIF                  1880.00    73604.00
BOB                   110.00     2938.00
BTN                    65.94      100.00
BYR                     1.26        3.15
CDF                  3259.00   267366.67
CNY                     4.20        5.80
COP                  4408.50 21777780.00
CVE                    95.49      346.67
DJF                   200.00    14500.00
DOP                  1782.07     5518.18
DZD                   230.00     1200.00
EGP                    20.72      150.46
ERN                  3000.00     5000.00
```

ETB	1400.00	35000.00
GEL	46.50	55.00
GHS	155.00	1748.54
GMD	65.22	1000.00
GNF	12833.33	90000.00
GTQ	140.25	501.79
HNL	884.38	2178.29
HTG	180.00	875.00
IDR	38225.81	201708.33
INR	91.60	4910.00
IQD	3000.00	21250.00
IRR	93907.50	445000.00
JOD	2.17	12.25
JPY	578.00	2627.00
KES	2390.31	10350.00
KGS	107.25	700.00
KHR	7833.50	85000.00
KZT	436.00	2196.00
LAK	40000.00	85000.00
LBP	4000.00	102112.64
LKR	99.60	1203.75
LRD	2517.19	11000.00
LSL	63.97	124.48
LYD	5.09	333.92
MDL	7.00	12.75
MGA	3080.00	9800.00
MMK	1515.15	6989.00
MNT	5423.00	12758.00
MRO	387.50	300000.00
MWK	408.99	10325.78
MXN	16.38	39.25
MZN	57.50	1000.00
NAD	14.38	68.57
NGN	7752.50	90000.00
NIO	928.28	2916.67
NIS	22.17	160.00
NPR	140.00	1000.00
PAB	47.00	122.67
PEN	4.15	8.98
PHP	137.08	960.83
PKR	147.50	322.46
PYG	16182.00	30500.00
RUB	16.70	23.00
RWF	744.00	332333.33
SDG	171.00	20000.00
SLL	13157.84	166667.00
SOS	22300.00	17250000.00

```
SSP                    900.00    260000.00
SYP                   1475.00   1283333.33
SZL                     15.98       820.00
THB                     13.65        67.33
TJS                      9.03       142.90
TRY                     23.94       167.86
TZS                  31457.61   3000000.00
UAH                     34.72       181.72
UGX                   2200.00     10000.00
USD                     28.14      2062.12
VEF                   6661.32      8627.91
VND                   8025.00     10288.89
XAF                   6500.00    214833.00
XOF                    425.00    474500.00
YER                    600.00    147500.00
ZAR                      2.93         5.76
ZMW                     12.67       100.00
ZWL                    171.16      4225.00
```

[56]:

```
gfp_float.plot.hist(bins = 10, figsize = (15,8), color = "purple");
plt.ticklabel_format(style = "plain", useOffset = False);
```



8 Feature Engineering 2

The dataset shows a lot of food products totaling 601 unique products. A deep look into it shows that majority are products that are just repeated based on how they are packaged, processed or quanity sold. In this second phase of feature engineering, few unique food products that fall into one of the 6 classes of food were painstakenly slected. This will help in narrowing down the analysis and *making good visualization*

[57]:
```
"""
=====SELECTING MAJOR UNIQUE PRODUCT UNIT MEASURE=======
"""
gfpu = gfp.loc[gfp.unit.isin(['KG','Unit','Packet','Pounds','10 pcs','L','Cubic␣
 ↪meter','Dozen','Cuartilla',
 'Libra','Sack', 'Package', 'Head', 'MT','Bunch','Marmite', 'Gallon','200␣
 ↪ML','Loaf',
'Pile', 'Heap','Bundle', 'LCU/3.5kg','100 Tubers'])]

gfpu.unit.describe(include = 'all')
```

[57]:
```
count     1626459
unique         23
top            KG
freq      1446536
Name: unit, dtype: object
```

[58]:
```
gfp.groupby("unit").product.unique()
```

[58]:
```
unit
0.5 KG
[Tomatoes, Onions]
0.8 KG
[Milk (powder)]
1 piece
[Eggs]
1.1 KG
[Groundnuts, Bread (bakery), Bread (shop), Bread (bakery, parallel market)]
1.2 KG
[Beans (white), Beans (red), Fish, Groundnuts, Cowpeas]
1.3 KG
[Tomatoes (paste), Beans (white), Maize flour, Beans (red), Sugar, Cowpeas,
Bananas, Sorghum]
1.4 KG
[Rice (imported), Sorghum, Rice (local), Millet]
1.5 KG
[Wheat flour, Fish]
1.5 L
[Oil (palm), Water (drinking)]
1.6 KG
[Chickpeas, Oil (groundnut)]
1.8 KG
[Eggs, Milk (powder)]
1.8 L
[Oil (vegetable)]
```

```
10 KG
[Meat (chicken), Meat (chicken, local), Maize meal (imported), Rice (Emata),
Potatoes, Maize meal]
10 pcs
[Eggs (duck), Eggs (duck, fermented), Eggs, Eggs (local), Eggs (imported)]
100 KG
[Wheat flour, Wheat, Rice (medium grain), Rice (coarse, BR-8/ 11/, Guti Sharna),
Rice (coarse, Guti Sharna), Rice, Sugar, Rice (imported), Millet, Sorghum
(local), Maize (white), Groundnuts (shelled), Sorghum (white), Onions, Sorghum
(red), Cowpeas (Red), Beans (niebe), Sorghum, Chickpeas, Barley, Beans (fava),
Lentils, Teff, Wheat (white), Teff (white), Barley (white), Teff (red), Lin
seed, Sorghum (mixed), Niger seed, Rape seed, Teff (Sergegna), Peas (mixed),
Teff (mixed), Peas, Maize (yellow), Beans (haricot, white), Wheat (mixed), Peas
(split, dry), Barley (mixed), Beans (mung), Groundnuts, Sesame, Millet (finger),
Beans (haricot), Beans, Beans (haricot, red), Wheat (food aid), Maize (food
aid), Sorghum (food aid), Maize, Rice (local), Beans (red), Yam, Cowpeas
(white), Cowpeas (brown), Sorghum (brown), Cassava meal (gari, yellow), Gari
(white), Rice (milled, local), Yam (Abuja), Oil (palm), Potatoes (Irish), Millet
(bulrush)]
100 L
[Oil (palm)]
100 Pounds
[Maize (yellow), Wheat (imported), Maize (white), Potatoes, Beans (black), Rice
(ordinary, first quality), Rice (ordinary, second quality), Bananas]
100 Tubers
[Yam, Yam (Abuja)]
100 pcs
[Plantains]
109 KG
[Sorghum, Soybeans, Cowpeas (white)]
11.5 KG
[Potatoes (Dutch), Potatoes (Irish, imilla)]
115 G
[Salt]
12 KG
[Plantains, Bananas]
12.5 KG
[Maize meal, Wheat flour, Wheat meal]
120 KG
[Rice (local)]
125 G
[Fish (canned), Fish (sardine, canned), Yogurt]
15 KG
[Maize (white), Maize (yellow)]
150 G
[Bread]
16 KG
```

```
[Peppers (dried)]
160 G
[Cheese (picon), Fish (tuna, canned)]
160 KG
[Cassava (fresh)]
168 G
[Cheese (picon)]
170 G
[Fish (tuna, canned)]
18 KG
[Maize (white), Maize (yellow)]
185 G
[Fish (tuna, canned)]
2 KG
[Eggs, Rice]
2.1 KG
[Watermelons]
2.25 KG
[Bitterball, Peppers (fresh)]
2.5 KG
[Maize meal, Wheat flour]
20 G
[Milk]
20 KG
[Cocoyam (macabo), Beans (red), Peppers (fresh), Fish (mackerel, fresh)]
20 L
[Oil (palm)]
200 G
[Cocoa (powder), Salt, Meat (beef, canned), Fish (tuna, canned)]
200 ML
[Milk (UHT), Milk (condensed)]
25 KG
[Rice (small grain, imported)]
250 G
[Tea (black), Tea (green), Salt, Handwash soap]
250 KG
[Yam, Yam (puna)]
250 ML
[Shampoo]
27 KG
[Eggplants]
28 pcs
[Diapers]
3 KG
[Sorghum, Wheat, Sorghum (food aid)]
3 L
[Oil (maize)]
```

```
3.1 KG
[Yam]
3.4 KG
[Yam]
3.5 KG
[Maize (white), Cassava, Millet (white), Sesame, Cowpeas, Sorghum (food aid),
Cassava (dry), Sorghum (brown), Sorghum (white, imported), Sorghum (local),
Maize (food aid), Sorghum (red, local), Milling cost (sorghum), Maize meal,
Millet]
30 pcs
[Eggs, Diapers]
300 G
[Pasta, Spinach]
350 G
[Pasta]
360 pcs
[Eggs]
380 G
[Milk (pasteurized)]
385 G
[Milk (condensed)]
400 G
[Milk (powder), Bread, Beans, Chickpeas, Tomatoes (paste), Oranges, Onions]
45 KG
[Rice, Maize (white), Beans (red), Beans (silk red), Beans (black), Rice
(ordinary, first quality), Rice (ordinary, second quality)]
46 KG
[Meat (chicken, whole), Potatoes (Irish, imilla), Maize (yellow), Wheat flour
(local), Rice (ordinary, first quality), Rice (ordinary, second quality), Wheat
flour (imported), Quinoa, Maize, Beans (white), Beans (red), Beans (black),
Rice, Sorghum, Maize (white), Beans (silk red), Sorghum (white), Rice (low
quality), Beans (kidney, pinto), Lentils, Beans (cranberry)]
5 KG
[Groundnuts (shelled), Cassava (fresh), Rice, Maize meal]
5 L
[Oil (vegetable), Oil (sunflower)]
5 pcs
[Bread]
50 KG
[Rice (local), Rice (long grain, imported), Rice, Wheat flour, Rice (basmati,
broken), Rice (Belem), Rice (imported), Potatoes (Irish), Rice (small grain,
imported), Cassava (fresh), Rice (white, imported), Charcoal, Cassava, Feed
(flour), Feed (wheat bran), Feed (rakhel), Sugar (white), Rice (milled, local),
Maize (yellow), Maize (white), Sorghum (brown), Sorghum (white, imported),
Sorghum (local), Sorghum (red, local), Millet (white), Wheat flour (locally
processed), Beans (sugar), Milling cost (wheat)]
50 Pounds
```

```
[Milk (powder), Wheat flour, Tomatoes]
50 pcs
[Onions (white)]
500 G
[Pasta, Peas (split, dry), Beans (sugar-red), Rice, Milk (powder), Pasta
(spaghetti), Pasta (macaroni), Bread, Labaneh, Yogurt, Wheat flour]
500 ML
[Milk (pasteurized), Milk (UHT), Milk (cow, pasteurized)]
52 KG
[Tomatoes (local), Tomatoes (navrongo)]
60 KG
[Wheat flour]
650 G
[Eggs, Tomatoes (paste)]
68 KG
[Gari]
70 G
[Tomatoes (paste)]
700 G
[Bread, Bread (brown)]
73 KG
[Onions]
750 G
[Salt (iodised), Maize flour (imported)]
750 ML
[Oil (vegetable), Oil (sunflower), Oil (palm), Oil (cooking)]
800 G
[Bread, Bread (brown)]
84 KG
[Rice (paddy)]
85 G
[Coffee (instant)]
90 KG
[Maize, Cassava, Groundnuts (shelled), Beans (niebe), Onions, Cassava
(cossette), Sesame, Soybeans, Sorghum (red), Rice (local), Groundnuts
(unshelled), Sorghum (white), Peas (yellow), Sorghum, Maize (white), Beans
(dry), Millet, Wheat]
900 G
[Cassava flour, Maize flour, Rice (regular, milled), Sugar (white), Milk
(powder)]
900 ML
[Dishwashing liquid]
91 KG
[Cassava]
93 KG
[Millet]
Bar
```

```
[Handwash soap]
Bunch
[Plantains (apentu), Plantains (apem), Kale]
Bundle
[Fish (dry), Fish (fresh)]
Cuartilla
[Rice (carolina 2da), Rice (estaquilla), Rice (good quality), Potatoes (Dutch),
Potatoes (Irish, imilla), Noodles (short)]
Cubic meter
[Water (drinking)]
Day
[Salt]
Dozen
[Bananas, Eggs]
Gallon
[Oil (vegetable, imported), Oil (palm), Cane juice (strong), Cane juice (light)]
Head
[Livestock (goat, medium-sized castrated male), Livestock (cattle), Livestock
(sheep, medium-sized castrated male), Livestock (Goat), Livestock (Sheep),
Livestock (donkey), Livestock (camel), Livestock (ox), Livestock (bull),
Cabbage, Lettuce, Livestock (sheep, medium-sized male), Livestock (goat, medium-
sized male), Chicken, Livestock (sheep, two-year-old male)]
Heap
[Straw]
KG             [Bread, Wheat, Rice (low quality), Oil (cooking), Sugar, Pulses,
Wheat flour (high quality), Salt, Rice (high quality), Wheat flour (low
quality), Wheat flour, Rice, Beans (white), Potatoes, Meat (chicken), Lentils,
Tomatoes, Meat (beef), Carrots, Onions, Bananas, Tea, Apples, Oranges, Meat
(camel), Cassava flour, Salt (iodised), Sugar (white), Rice (white, imported),
Maize meal (yellow), Fish (mackerel, dry), Beans (kidney, pinto), Rice (white),
Maize (yellow), Beans, Pasta, Meat (pork), Cheese (dry), Peas (split, dry),
Cabbage, Apples (red), Cucumbers (greenhouse), Tomatoes (paste), Beetroots, Fish
(fresh), Bread (high grade flour), Bread (first grade flour), Buckwheat, Rice
(coarse), Lentils (masur), Rice (medium grain), Rice (coarse, BR-8/ 11/, Guti
Sharna), Rice (coarse, Guti Sharna), Oil (palm), Tea (black), Oil (sunflower),
Oil (mustard), Oil (groundnut), Oil (soybean), Lentils (moong), Sugar
(jaggery/gur), Lentils (urad), Ghee (vanaspati), Chickpeas, Maize, Rice
(imported), Sorghum, Maize (white), Rice (local), Millet, Sweet potatoes, Yam,
Plantains, Peas (green, dry), Soybeans, Sorghum (red), Cassava (cossette),
Lemons, Wheat flour (imported), Cassava meal (gari), Fish (tilapia), Papaya,
Okra (fresh), Groundnuts (Bambara), Peppers (red, dry), Groundnuts (small,
unshelled), Cassava meal (tapioca), Coconut (dried), Yam (white), Fish (fresh,
silvi), Oil (palm nut), Cassava meal (gari, fine), Yam (dry), Yam (flour), Beans
(red), Beans (black), Rice (paddy), Cassava (fresh), Yam (yellow), Leafy
vegetables, …]
L
[Milk, Oil, Milk (camel), Oil (palm), Oil (soybean), Oil (vegetable), Milk (non-
```

pasteurized), Milk (pasteurized), Oil (groundnut), Oil (cotton), Oil (vegetable, imported), Oil (sunflower), Oil (maize), Oil (mixed), Shampoo, Disinfecting solution, Oil (olive), Milk (cow, fresh), Milk (camel, fresh), Kefir, Water (drinking), Dishwashing liquid, Laundry detergent, Oil (vegetable, local), Oil (mixed, imported), Oil (mustard), Oil (cooking), Oil (vegetable, fortified, food aid), Milk (fresh), Groundnuts (shelled), Yogurt, Oil (vegetable, fortified)]
LCU/3.5kg
[Milling cost (sorghum), Milling cost (maize)]
Libra
[Sugar, Potatoes (Irish, imilla), Cocoa, Coffee, Fish (tilapia), Maize, Rice, Beans (red), Rice (high quality), Beans (white), Beans (black), Meat (chicken), Rice (ordinary, first quality), Rice (ordinary, second quality), Sorghum, Maize (white), Beans (silk red), Lentils, Beans (cranberry)]
Loaf
[Bread (brown), Bread (rye), Bread (wheat)]
MT
[Rice (white), Rice (paddy), Maize (white), Beans (black), Rice (milled 80-20), Beans (red), Maize flour (white), Sorghum (white), Beans (silk red), Maize (yellow), Rice, Wheat, Bulgur, Maize, Beans]
Marmite
[Rice (tchako), Rice (imported), Sorghum, Rice (local), Beans (red), Beans (black), Wheat flour (imported), Sugar (white), Maize meal (local)]
Package
[Laundry detergent, Tea (herbal)]
Packet
[Tea (sahm), Pasta (spaghetti), Spinach, Parsley, Noodles (instant, indomie)]
Pile
[Lettuce, Potato Leaves, Cassava leaves]
Pound
[Noodles (short), Sugar, Rice (carolina 2da), Rice (estaquilla), Potatoes (Dutch), Potatoes (Irish, imilla), Beans (red), Beans (black), Eggs, Maize (yellow), Rice (good quality), Bread, Potatoes, Meat (chicken), Pasta, Tomatoes, Meat (beef, chops with bones), Cheese (dry), Plantains, Coffee, Onions, Rice (ordinary, second quality), Milk (powder), Bananas, Tortilla (maize), Meat (chicken, whole), Rice (imported), Sorghum, Rice (local), Maize meal (imported), Rice, Maize (white), Rice (milled 80-20), Onions (white), Meat (pork), Meat (beef), Cabbage, Salt, Coffee (instant), Squashes, Oranges, Fish (fresh), Peppers (sweet)]
Sack
[Charcoal]
USD/LCU
[Exchange rate, Exchange rate (unofficial)]
Unit
[Livestock (sheep, one-year-old alive female), Bread, Cheese (dry), Eggs, Bread (wheat), Fish (sardine, canned), Laundry soap, Eggs (white, AA), Eggs (local), Tea (green), Coffee (instant), Milk (powder), Tea, Meat (beef, canned), Pasta (spaghetti), Handwash soap, Meat (chicken, local), Candles (small) , Candles

```
(big) , Batteries (small), Batteries (big), Bread (khoboz), Bananas, Bread
(vetkoek), Bread (brotchen), Bread (traditional), Eggs (duck), Livestock (Goat),
Livestock (hen), Avocados, Livestock (cattle), Livestock (Sheep), Livestock
(pig), Meat (camel), Coconut]
Name: product, dtype: object
```

[59]:

```
"""
======SELECTING MAJOR GLOBALLY USED PRODUCTS========
"""
gfpu = gfp.loc[gfp["product"].isin(["Eggs","Yam","Milk (UHT)", "Plantains␣
 ↪(apentu)", "Fish (fresh)","Water (drinking)","Bananas",
"Oil (vegetable, imported)", "Oil (palm)","Livestock (Goat)", "Livestock␣
 ↪(Sheep)", "Livestock (donkey)",
 "Livestock (camel)", "Livestock (ox)", "Livestock (bull)", "Cabbage",␣
 ↪"Lettuce","Livestock (cattle)","Coconut",
"Bread", "Wheat","Sugar","Salt","Rice","Potatoes"," Tomatoes", "Meat (beef)",␣
 ↪"Carrots", "Onions","Tea",
 "Apples","Oranges","Beans","Milk","Oil","Oil (palm)","Bulgur","Bread (brown)",␣
 ↪"Bread (rye)", "Bread (wheat)",
"Lentils","Straw","Milling cost (sorghum)", "Milling cost␣
 ↪(maize)","Sorghum","Laundry detergent","Avocados",
"Spinach", "Parsley", "Noodles (instant, indomie)","Lettuce", "Potato Leaves",␣
 ↪"Cassava leaves", "Charcoal"])]
```

[60]:

```
gfpu["product"].describe(include = 'all')
```

[60]:

```
count      503181
unique         51
top       Sorghum
freq        48608
Name: product, dtype: object
```

[61]:

```
gfpu.describe(include = 'all')
```

[61]:

```
              date         country                city           market  \
count       503181          503181              503181           503181
unique         257              89                 546             2829
top     2021-03-01  Bassas da India  North/Amajyaruguru  National Average
freq          8142           51772              183652             5745
mean           NaN             NaN                 NaN              NaN
std            NaN             NaN                 NaN              NaN
min            NaN             NaN                 NaN              NaN
25%            NaN             NaN                 NaN              NaN
50%            NaN             NaN                 NaN              NaN
75%            NaN             NaN                 NaN              NaN
max            NaN             NaN                 NaN              NaN
```

```
       currency    type    unit   month      Year       price  product continet
count    503181  503181  503181  503181 503181.00   503181.00   503181   503181
unique       75       2      65      12       NaN         NaN       51        4
top         INR  Retail      KG   March       NaN         NaN  Sorghum   Africa
freq      51772  466990  362890   45878       NaN         NaN    48608   246128
mean        NaN     NaN     NaN     NaN   2015.87     7174.09      NaN      NaN
std         NaN     NaN     NaN     NaN      4.17    46613.87      NaN      NaN
min         NaN     NaN     NaN     NaN   2000.00        0.00      NaN      NaN
25%         NaN     NaN     NaN     NaN   2013.00       26.45      NaN      NaN
50%         NaN     NaN     NaN     NaN   2017.00      190.00      NaN      NaN
75%         NaN     NaN     NaN     NaN   2019.00      900.00      NaN      NaN
max         NaN     NaN     NaN     NaN   2021.00  2620000.00      NaN      NaN
```

```
[62]: gfpu.head()
```

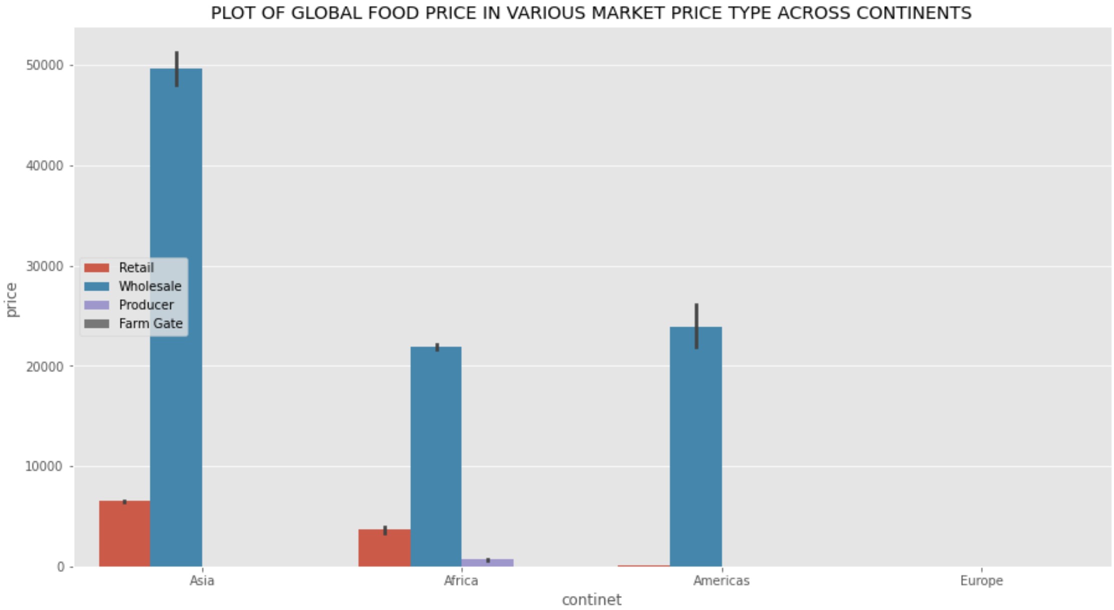
```
[62]:          date      country         city    market currency    type unit  \
      0  2014-01-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      1  2014-02-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      2  2014-03-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      3  2014-04-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG
      4  2014-05-01  Afghanistan  Badakhshan  Fayzabad      AFN  Retail   KG

            month  Year  price product continet
      0   January  2014  50.00   Bread     Asia
      1  February  2014  50.00   Bread     Asia
      2     March  2014  50.00   Bread     Asia
      3     April  2014  50.00   Bread     Asia
      4       May  2014  50.00   Bread     Asia
```

9 Visualization

```
[66]: plt.figure(figsize=(15,8));
      sns.barplot('continet','price',hue='type',data=gfp, );      #kind= 'point',␣
       ↪height=5.2, width = 10.2                                                  ␣
       ↪                                                                          ␣
       ↪                                                                          ␣
       ↪                                                                          ␣
       ↪
      plt.title('PLOT OF GLOBAL FOOD PRICE IN VARIOUS MARKET PRICE TYPE ACROSS␣
       ↪CONTINENTS');
      plt.legend()
      gfp["price"].describe().T
```

```
[66]: count    1859290.00
      mean        6654.93
      std       112034.74
```

```
min             0.00
25%            42.86
50%           235.50
75%          1100.00
max      21777780.00
Name: price, dtype: float64
```



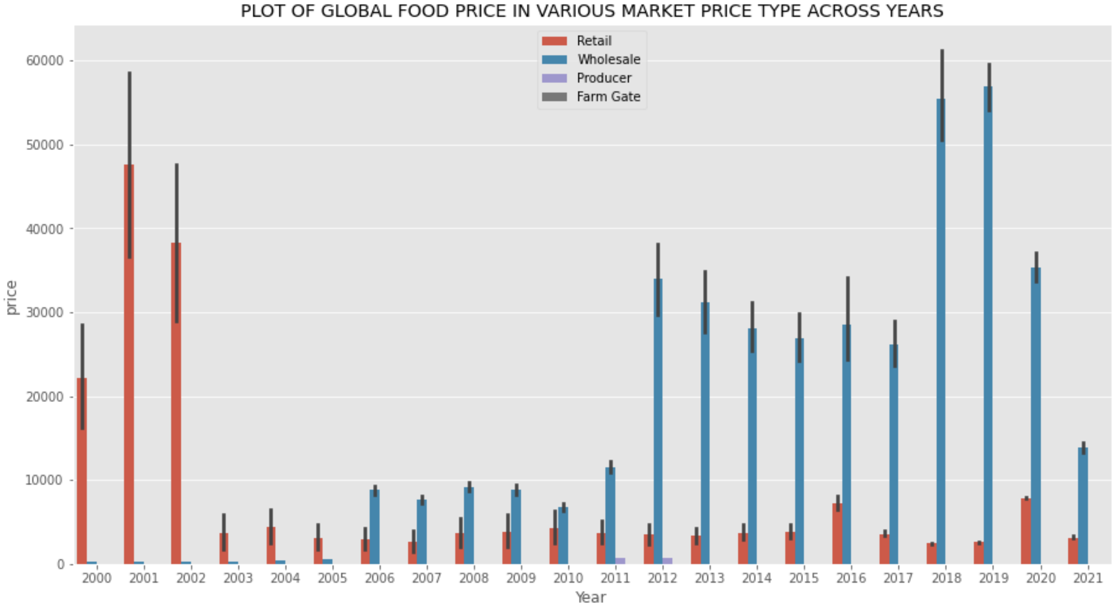
[67]:

```
plt.figure(figsize=(15,8));
sns.barplot('Year','price',hue='type',data=gfp, );      #kind= 'point',␣
 ↪height=5.2, width = 10.2                                                ␣
 ↪                                                                        ␣
 ↪                                                                        ␣
 ↪                                                                        ␣
 ↪
plt.title('PLOT OF GLOBAL FOOD PRICE IN VARIOUS MARKET PRICE TYPE ACROSS␣
 ↪YEARS');
plt.legend()
gfp["price"].describe().T
```

[67]:

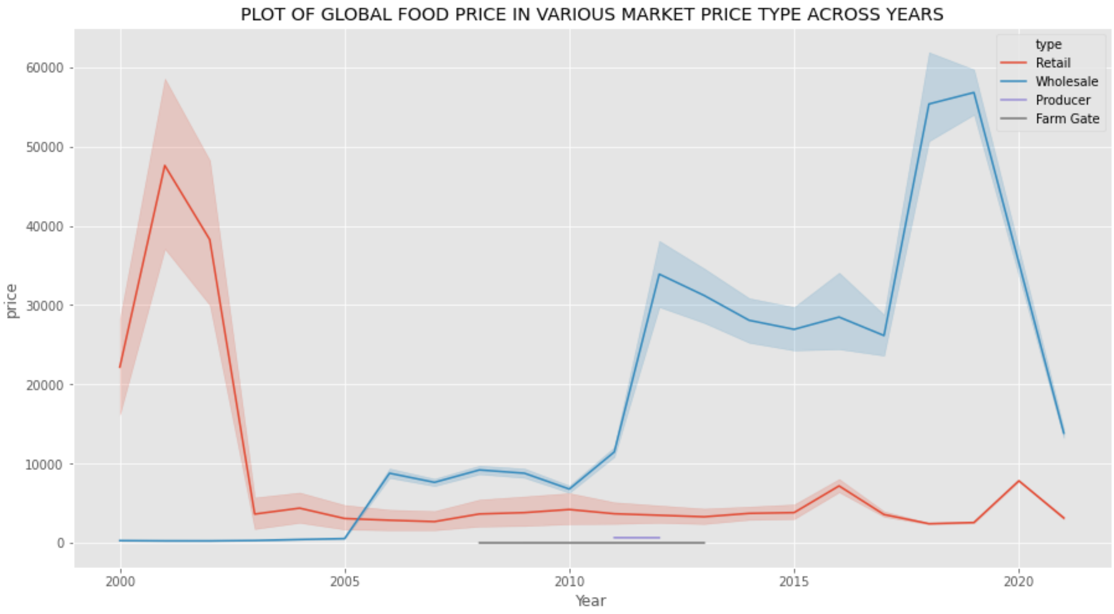
```
count     1859290.00
mean         6654.93
std        112034.74
min             0.00
25%            42.86
50%           235.50
75%          1100.00
max      21777780.00
```

```
Name: price, dtype: float64
```



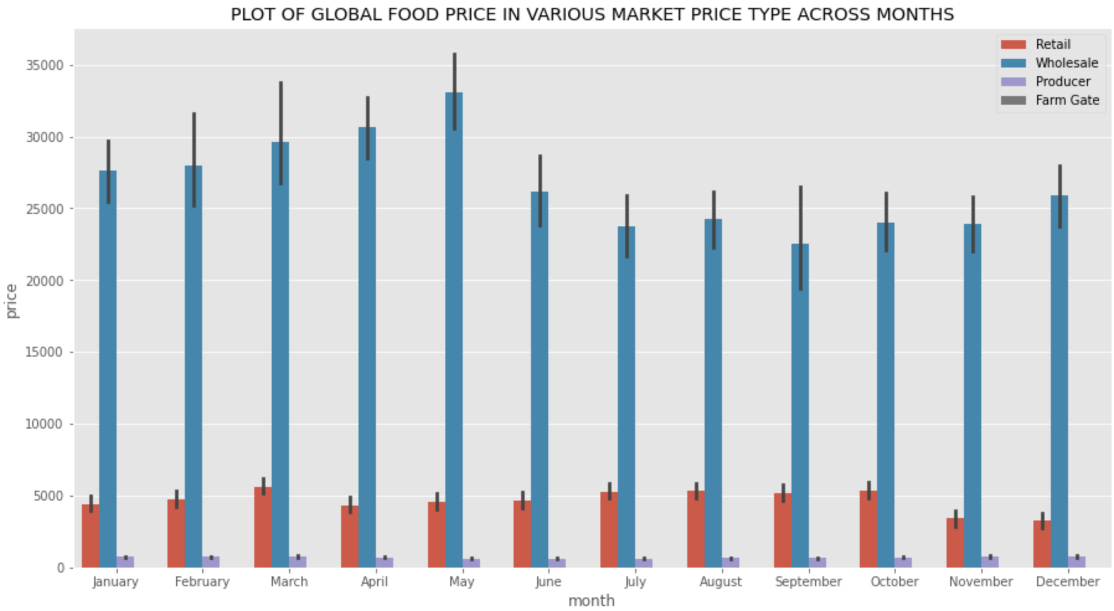
```
[69]: plt.figure(figsize=(15,8));
      sns.lineplot('Year','price',hue='type',data=gfp, );       #kind= 'point',␣
       ↪height=5.2, width = 10.2                                                  ␣
       ↪                                                                          ␣
       ↪                                                                          ␣
       ↪                                                                          ␣
       ↪
      plt.title('PLOT OF GLOBAL FOOD PRICE IN VARIOUS MARKET PRICE TYPE ACROSS␣
       ↪YEARS');
      plt.legend()
      gfp["price"].describe().T
```

```
[69]: count     1859290.00
      mean         6654.93
      std        112034.74
      min             0.00
      25%            42.86
      50%           235.50
      75%          1100.00
      max      21777780.00
      Name: price, dtype: float64
```

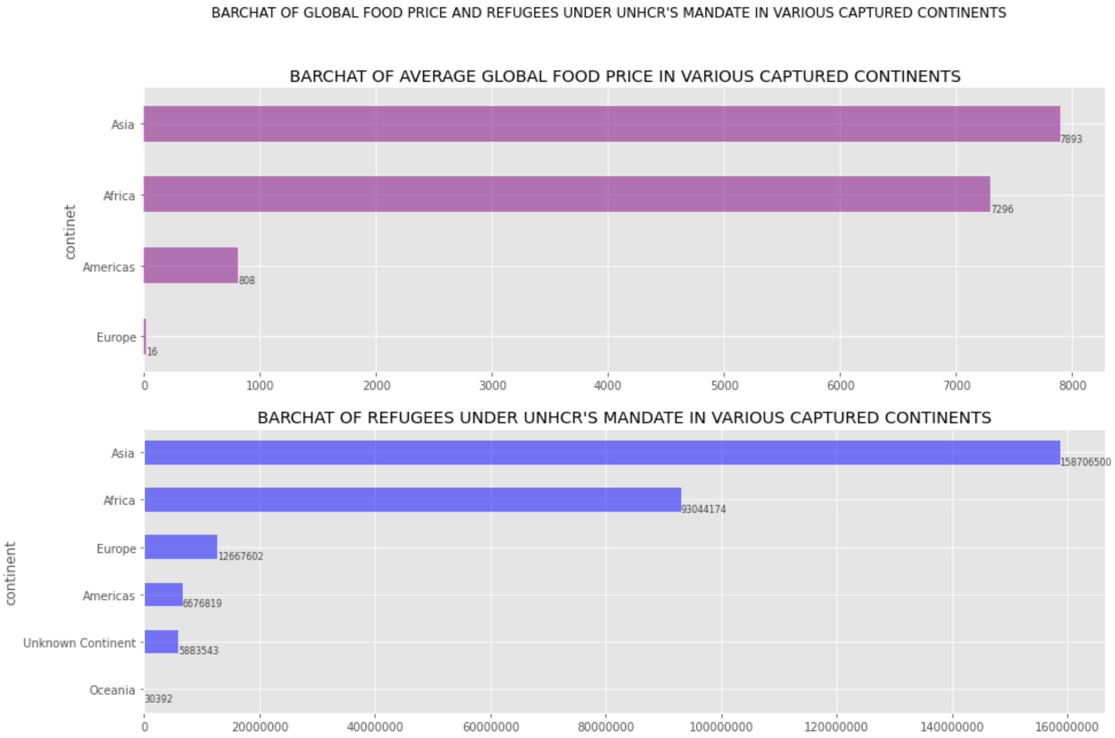


```
[70]: plt.figure(figsize=(15,8));
      sns.barplot('month','price',hue='type',data=gfp, );       #kind= 'point',␣
       ↪height=5.2, width = 10.2                                                   ␣
       ↪                                                                           ␣
       ↪                                                                           ␣
       ↪                                                                           ␣
       ↪
      plt.title('PLOT OF GLOBAL FOOD PRICE IN VARIOUS MARKET PRICE TYPE ACROSS␣
       ↪MONTHS');
      plt.legend()
      gfp["price"].describe().T
```

```
[70]: count    1859290.00
      mean        6654.93
      std       112034.74
      min            0.00
      25%           42.86
      50%          235.50
      75%         1100.00
      max     21777780.00
      Name: price, dtype: float64
```



[72]:
```
plt.suptitle("BARCHAT OF GLOBAL FOOD PRICE AND REFUGEES UNDER UNHCR'S MANDATE␣
 ↪IN VARIOUS CAPTURED CONTINENTS")
plt.subplots_adjust(wspace=0.3)
plt.grid(True)
plt.subplot(2,1,1)
plt.ticklabel_format(style='plain',useOffset=False)
#condition1 = (gfp.type == "Retail")# & (gfp.product == "Millet")
axc = gfpu.groupby("continet")["price"].mean().sort_values(ascending = True).
 ↪plot.barh(figsize = (15,10), alpha = 0.5,
                            color = 'purple', ylabel = "Price",
                                                    title="BARCHAT OF AVERAGE␣
 ↪GLOBAL FOOD PRICE IN VARIOUS CAPTURED CONTINENTS");
for i in axc.patches:
    axc.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
 ↪fontsize=8, color='black', alpha=0.7);
sns.despine(left=True)
plt.subplot(2,1,2)
plt.ticklabel_format(style = 'plain', useOffset=False)
axr = ref.groupby('continent')["Refugees under UNHCR's mandate"].sum().
 ↪sort_values(ascending = True).plot.barh(figsize = (15,10),
                                alpha = 0.5,color = 'blue', ylabel = "Refugees␣
 ↪under UNHCR's mandate",
                                title="BARCHAT OF REFUGEES UNDER UNHCR'S␣
 ↪MANDATE IN VARIOUS CAPTURED CONTINENTS");
for i in axr.patches:
    axr.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
↪fontsize=8, color='black', alpha=0.7);
plt.show()
```

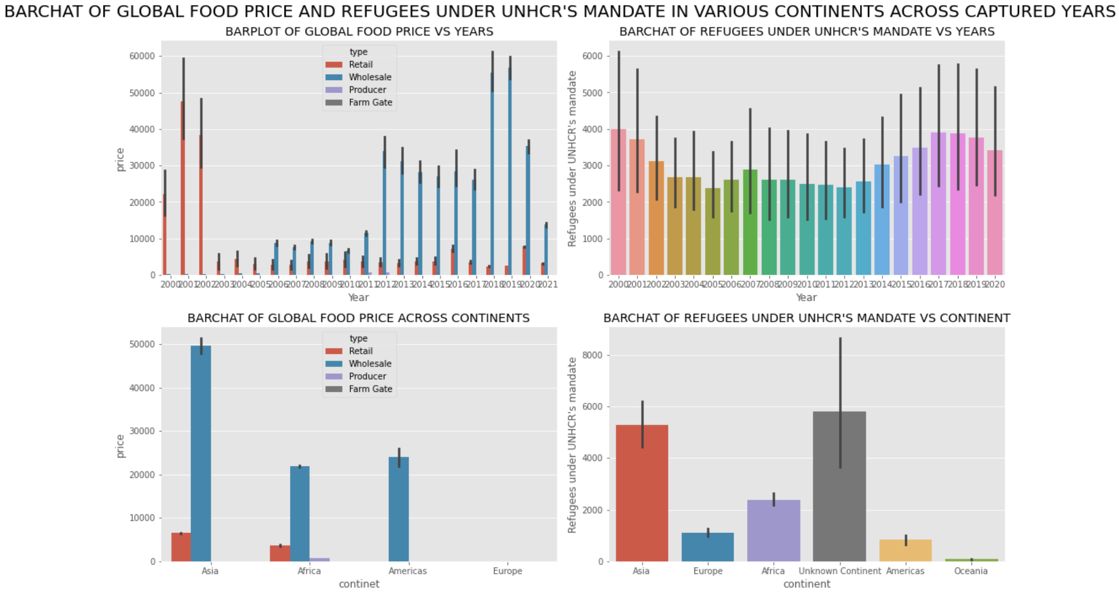


BARCHAT OF GLOBAL FOOD PRICE AND REFUGEES UNDER UNHCR'S MANDATE IN VARIOUS CAPTURED CONTINENTS

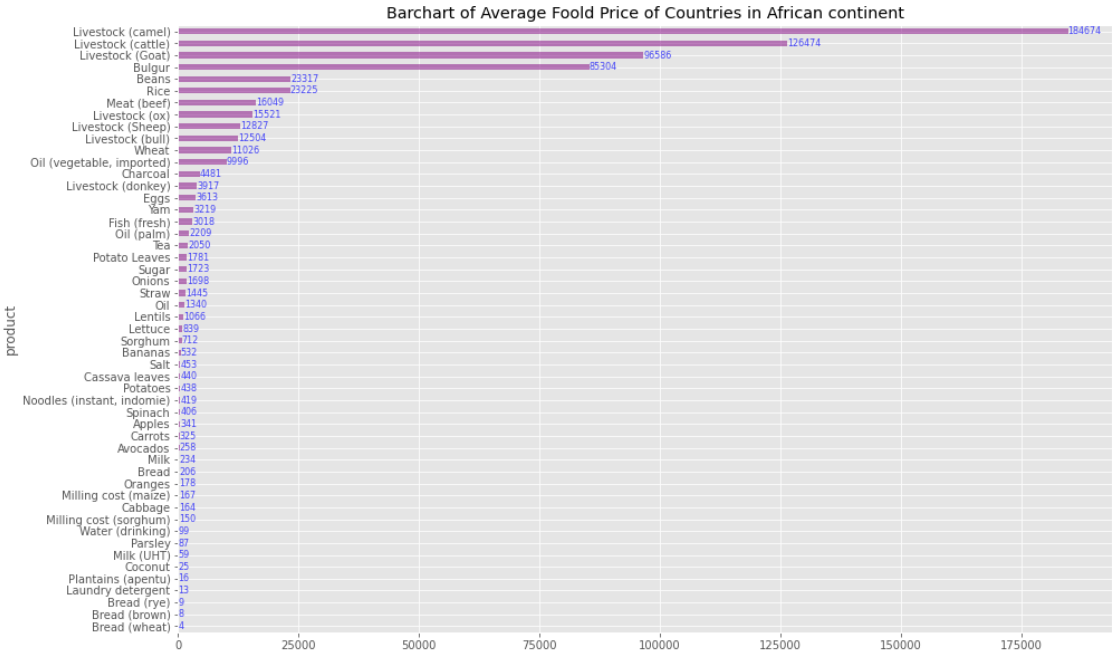
BARCHAT OF AVERAGE GLOBAL FOOD PRICE IN VARIOUS CAPTURED CONTINENTS

BARCHAT OF REFUGEES UNDER UNHCR'S MANDATE IN VARIOUS CAPTURED CONTINENTS

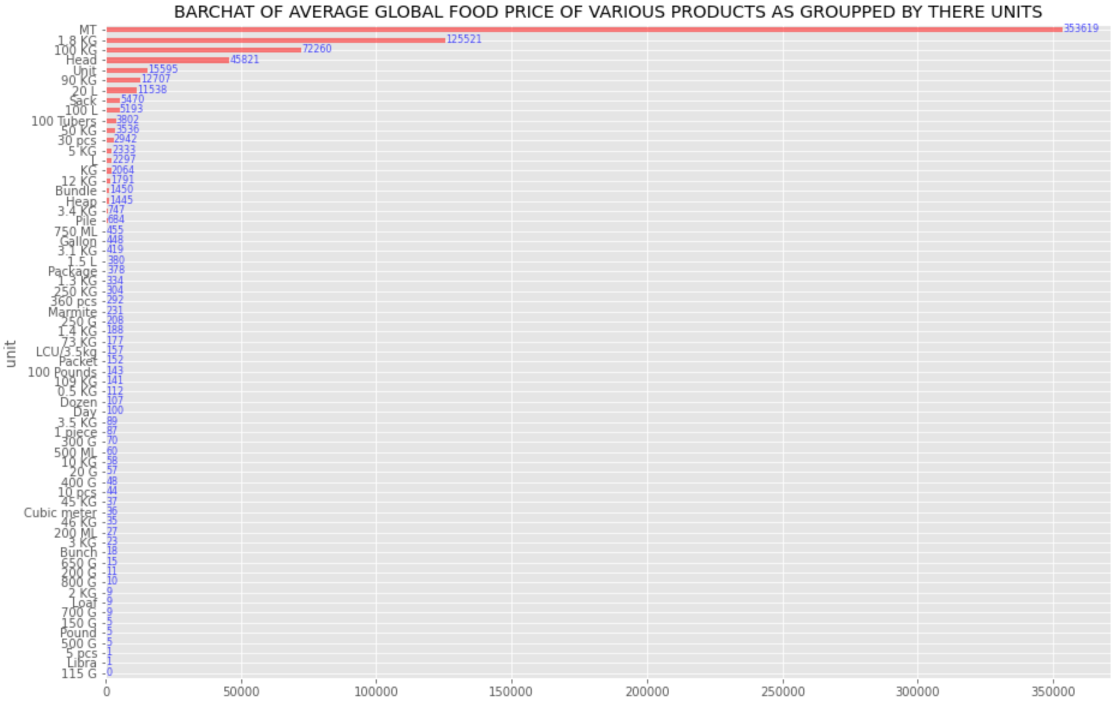
```
[73]: #Boxplots
plt.figure(figsize = (15,10))
plt.suptitle("BARCHAT OF GLOBAL FOOD PRICE AND REFUGEES UNDER UNHCR'S MANDATE␣
↪IN VARIOUS CONTINENTS ACROSS CAPTURED YEARS", fontsize = 20)
plt.subplots_adjust(wspace=0.3)
plt.grid(True)

plt.subplot(2,2,1)
sns.barplot(data = gfp,x = 'Year', y = 'price', hue = 'type');
plt.title('BARPLOT OF GLOBAL FOOD PRICE VS YEARS')
sns.despine(left= True)
plt.tight_layout()

plt.subplot(2,2,2)
sns.barplot(data = ref, x = 'Year', y = "Refugees under UNHCR's mandate");
plt.title("BARCHAT OF REFUGEES UNDER UNHCR'S MANDATE VS YEARS")
sns.despine(left=True)
plt.tight_layout()
plt.subplot(2,2,3)
sns.barplot(data=gfp, x = 'continet', y = 'price', hue = 'type');
plt.title("BARCHAT OF GLOBAL FOOD PRICE ACROSS CONTINENTS")
sns.despine(left=True)
plt.tight_layout()

plt.subplot(2,2,4)
sns.barplot(data=ref, x = 'continent', y = "Refugees under UNHCR's mandate");
plt.title("BARCHAT OF REFUGEES UNDER UNHCR'S MANDATE VS CONTINENT")
sns.despine(left=True)
plt.tight_layout()
plt.show()
```



[85]:
```
plt.ticklabel_format(style='plain',useOffset=False)
ax4 = gfpu.groupby("product").price.mean().sort_values(ascending = True).plot.
↪barh(figsize = (15,10), alpha = 0.5,
color = 'purple',title = "BARCHAT OF AVERAGE GLOBAL FOOD PRICE OF VARIOUS␣
↪PRODUCTS");
for i in ax4.patches:
    ax4.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
↪fontsize=8, color='b', alpha=0.7);
```



```
[87]: plt.ticklabel_format(style='plain',useOffset=False)
      ax4 = gfpu.groupby("unit").price.mean().sort_values(ascending = True).plot.
       ↪barh(figsize = (15,10), alpha = 0.5,
      color = 'red',title = "BARCHAT OF AVERAGE GLOBAL FOOD PRICE OF VARIOUS PRODUCTS␣
       ↪AS GROUPPED BY THERE UNITS");
      for i in ax4.patches:
          ax4.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
       ↪fontsize=8, color='b', alpha=0.7);
```



[88]:

```
gfpu.groupby("unit").product.unique()
```

[88]:

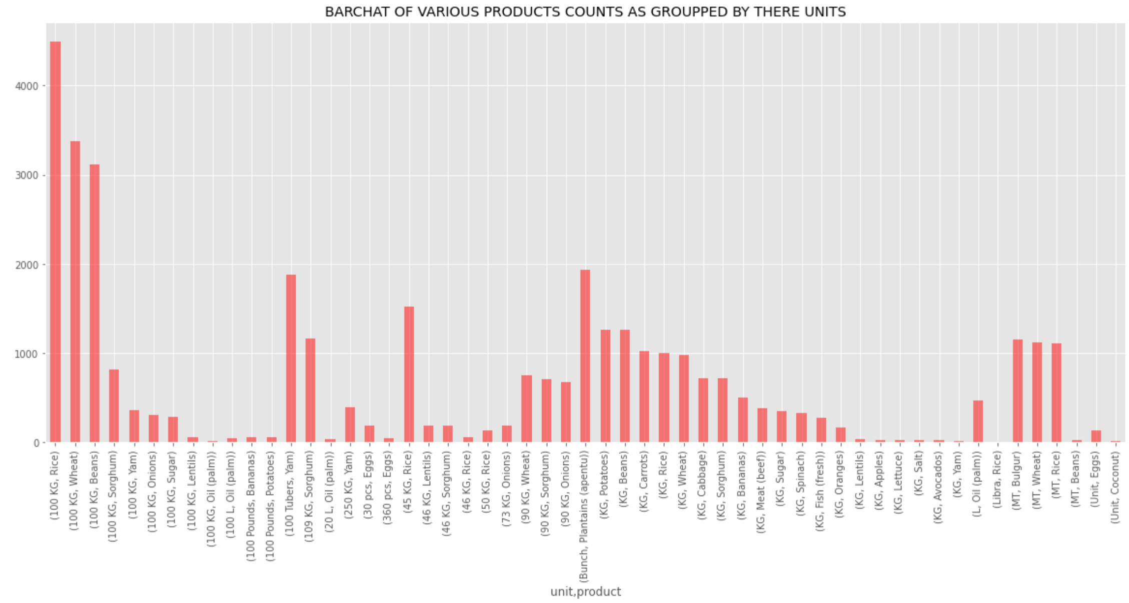
```
unit
0.5 KG
[Onions]
1 piece
[Eggs]
1.3 KG
[Sugar, Bananas, Sorghum]
1.4 KG
[Sorghum]
1.5 L
[Oil (palm), Water (drinking)]
1.8 KG
[Eggs]
10 KG
[Potatoes]
10 pcs
[Eggs]
100 KG
[Wheat, Rice, Sugar, Onions, Sorghum, Lentils, Beans, Yam, Oil (palm)]
100 L
[Oil (palm)]
100 Pounds
```

```
[Potatoes, Bananas]
100 Tubers
[Yam]
109 KG
[Sorghum]
115 G
[Salt]
12 KG
[Bananas]
150 G
[Bread]
2 KG
[Eggs, Rice]
20 G
[Milk]
20 L
[Oil (palm)]
200 G
[Salt]
200 ML
[Milk (UHT)]
250 G
[Salt]
250 KG
[Yam]
3 KG
[Sorghum, Wheat]
3.1 KG
[Yam]
3.4 KG
[Yam]
3.5 KG
[Milling cost (sorghum)]
30 pcs
[Eggs]
300 G
[Spinach]
360 pcs
[Eggs]
400 G
[Bread, Beans, Oranges, Onions]
45 KG
[Rice]
46 KG
[Rice, Sorghum, Lentils]
5 KG
[Rice]
```

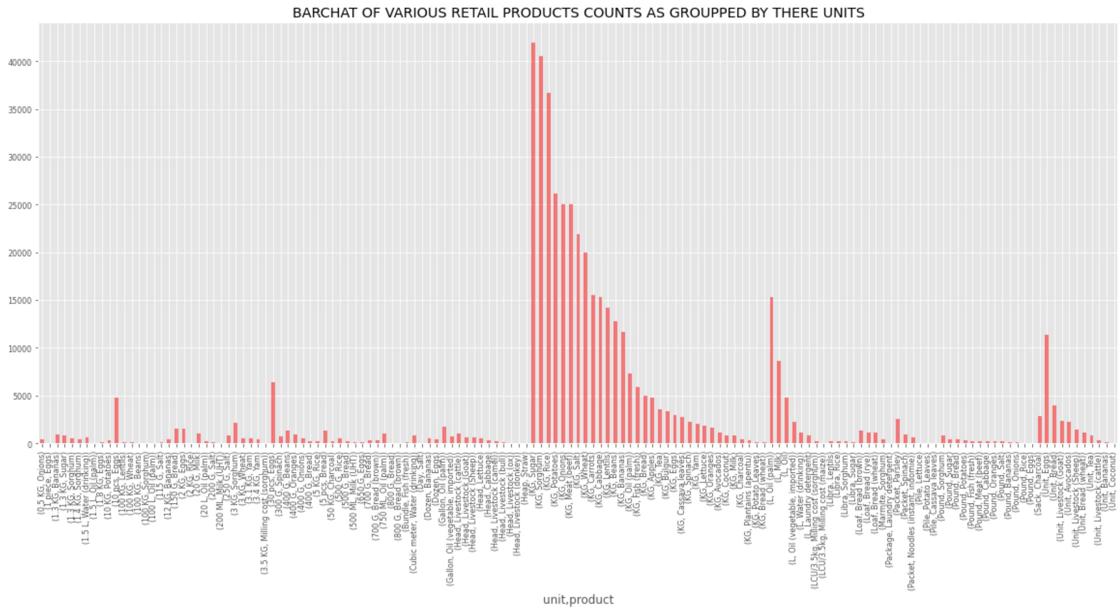
```
5 pcs
[Bread]
50 KG
[Rice, Charcoal]
500 G
[Rice, Bread]
500 ML
[Milk (UHT)]
650 G
[Eggs]
700 G
[Bread, Bread (brown)]
73 KG
[Onions]
750 ML
[Oil (palm)]
800 G
[Bread, Bread (brown)]
90 KG
[Onions, Sorghum, Wheat]
Bunch
[Plantains (apentu)]
Bundle
[Fish (fresh)]
Cubic meter
[Water (drinking)]
Day
[Salt]
Dozen
[Bananas, Eggs]
Gallon
[Oil (vegetable, imported), Oil (palm)]
Head
[Livestock (cattle), Livestock (Goat), Livestock (Sheep), Livestock (donkey),
Livestock (camel), Livestock (ox), Livestock (bull), Cabbage, Lettuce]
Heap
[Straw]
KG             [Bread, Wheat, Sugar, Salt, Rice, Potatoes, Lentils, Meat (beef),
Carrots, Onions, Bananas, Tea, Apples, Oranges, Beans, Cabbage, Fish (fresh),
Oil (palm), Sorghum, Yam, Bread (wheat), Avocados, Spinach, Lettuce, Milk,
Cassava leaves, Plantains (apentu), Charcoal, Eggs, Bulgur, Coconut, Potato
Leaves]
L
[Milk, Oil, Oil (palm), Oil (vegetable, imported), Water (drinking), Laundry
detergent]
LCU/3.5kg
[Milling cost (sorghum), Milling cost (maize)]
```

```
Libra
[Sugar, Rice, Sorghum, Lentils]
Loaf
[Bread (brown), Bread (rye), Bread (wheat)]
MT
[Rice, Wheat, Bulgur, Beans]
Marmite
[Sorghum]
Package
[Laundry detergent]
Packet
[Spinach, Parsley, Noodles (instant, indomie)]
Pile
[Lettuce, Potato Leaves, Cassava leaves]
Pound
[Sugar, Eggs, Bread, Potatoes, Onions, Bananas, Sorghum, Rice, Meat (beef),
Cabbage, Salt, Oranges, Fish (fresh)]
Sack
[Charcoal]
Unit
[Bread, Eggs, Bread (wheat), Tea, Bananas, Livestock (Goat), Avocados, Livestock
(cattle), Livestock (Sheep), Coconut]
Name: product, dtype: object
```

[102]:
```
condition_u = (gfpu.type == "Wholesale")
gfpu[condition_u].groupby("unit").product.value_counts().plot.bar(figsize =␣
 ↪(20, 8), alpha = 0.5,
color = 'red',title = "BARCHAT OF VARIOUS WHOLESALE PRODUCTS COUNTS AS GROUPPED␣
 ↪BY THERE UNITS", fontsize = 10);
```



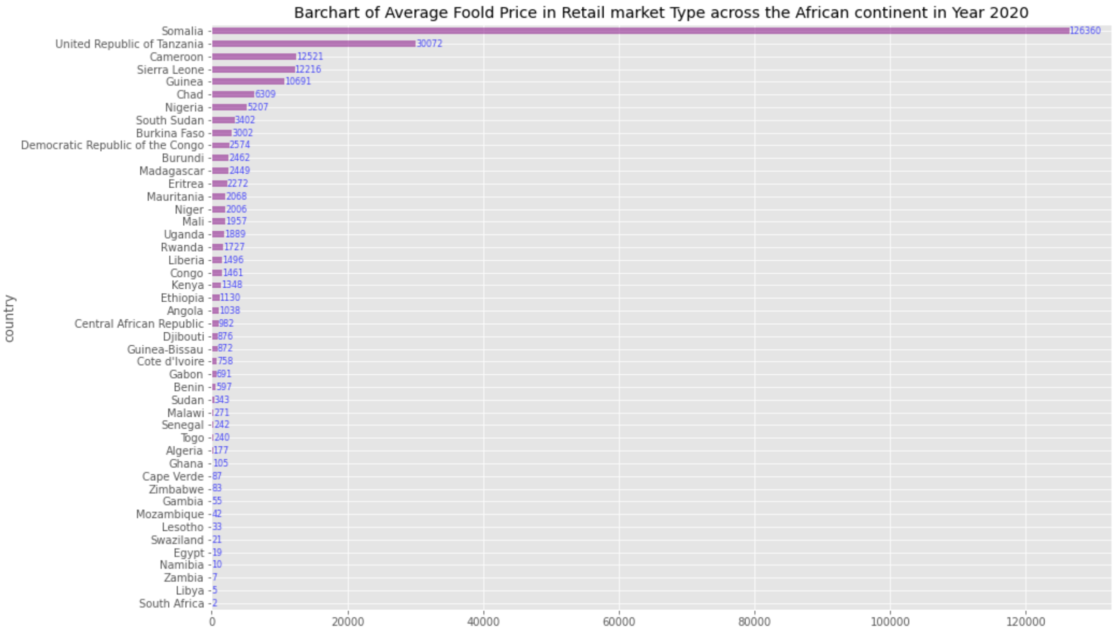
```
[105]: condition_R = (gfpu.type == "Retail")
       gfpu[condition_R].groupby("unit").product.value_counts().plot.bar(figsize =␣
        ↪(20, 8), alpha = 0.5,
       color = 'red',title = "BARCHAT OF VARIOUS RETAIL PRODUCTS COUNTS AS GROUPPED BY␣
        ↪THERE UNITS", fontsize = 8);
```



10 Deeper Visualization into Countries and Their Continent

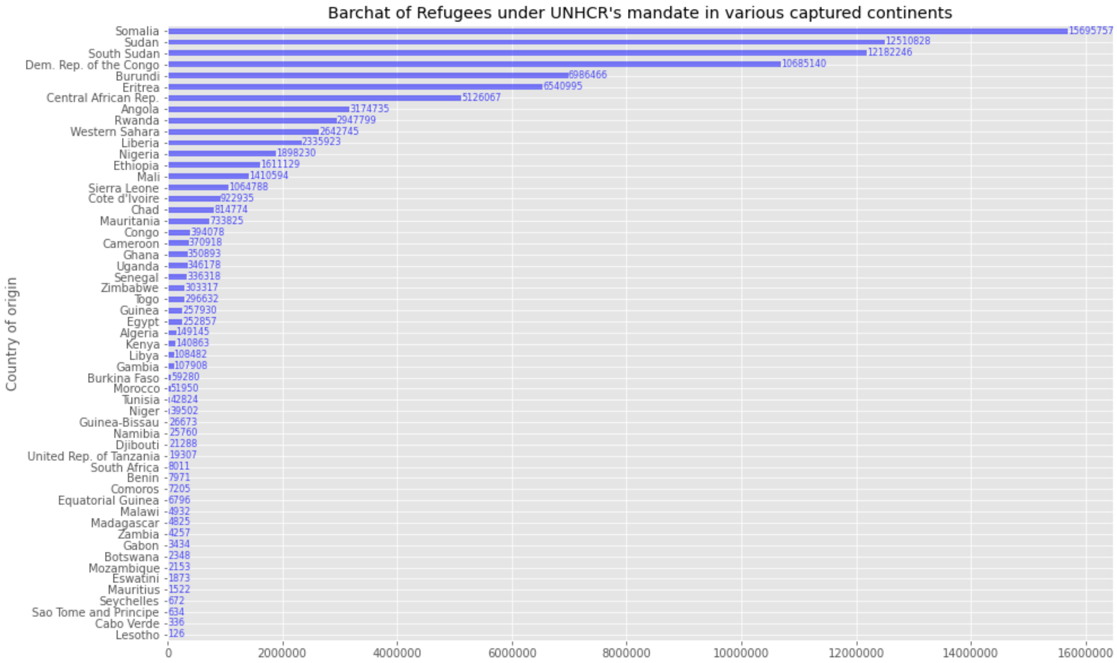
10.1 AFRICA

```
[76]: cond2 = (gfp.continet == "Africa")# & (gfp.type == "Retail") & (gfp.Year ==␣
       ↪2020))
      plt.ticklabel_format(style='plain',useOffset=False)
      ax4 = gfp[cond2].groupby("country").price.mean().sort_values(ascending = True).
       ↪plot.barh(figsize = (15,10), alpha = 0.5,
      color = 'purple',title = "Barchart of Average Foold Price of Countries in␣
       ↪African continent");
      for i in ax4.patches:
          ax4.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
       ↪fontsize=8, color='b', alpha=0.7);
```



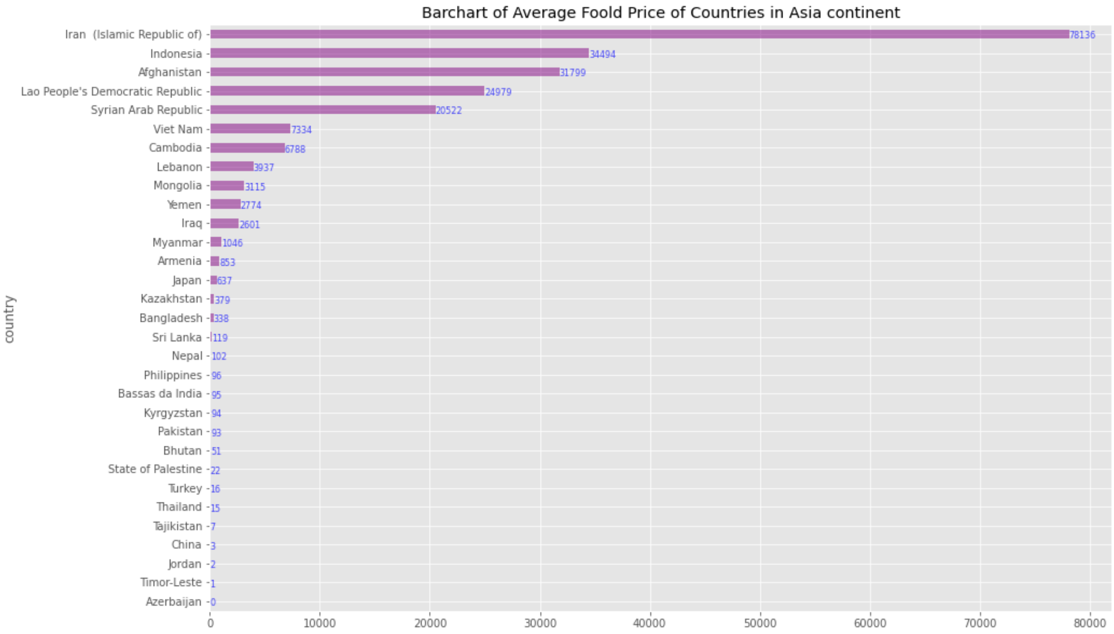
Barchart of Average Foold Price in Retail market Type across the African continent in Year 2020

[77]:
```
condition2 = (ref.continent == "Africa") #& (ref.Year == 2020)
plt.ticklabel_format(style = 'plain', useOffset=False)
axr = ref[condition2].groupby("Country of origin")["Refugees under UNHCR's␣
 ↪mandate"].sum().sort_values(ascending = True).plot.barh(figsize = (15,10),
                                alpha = 0.5,color = 'blue', ylabel = "Refugees␣
 ↪under UNHCR's mandate",
                                title="Barchat of Refugees under UNHCR's␣
 ↪mandate of Countries in African continents");
for i in axr.patches:
    axr.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
 ↪fontsize=8, color='black', alpha=0.7);
plt.show()
```



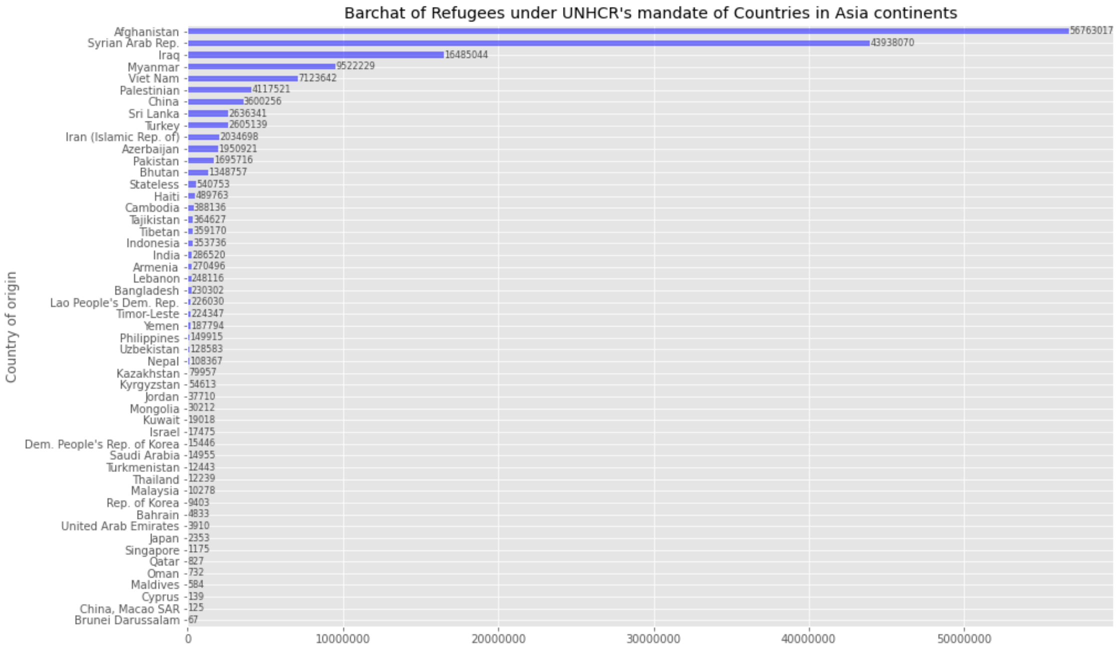
10.2 ASIA

[78]:
```
cond3 = (gfp.continet == "Asia")# & (gfp.type == "Retail") & (gfp.Year == 2020))
plt.ticklabel_format(style='plain',useOffset=False)
ax4 = gfp[cond3].groupby("country").price.mean().sort_values(ascending = True).
 ↪plot.barh(figsize = (15,10), alpha = 0.5,
color = 'purple',title = "Barchart of Average Foold Price of Countries in Asia␣
 ↪continent");
for i in ax4.patches:
    ax4.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
 ↪fontsize=8, color='b', alpha=0.7);
```



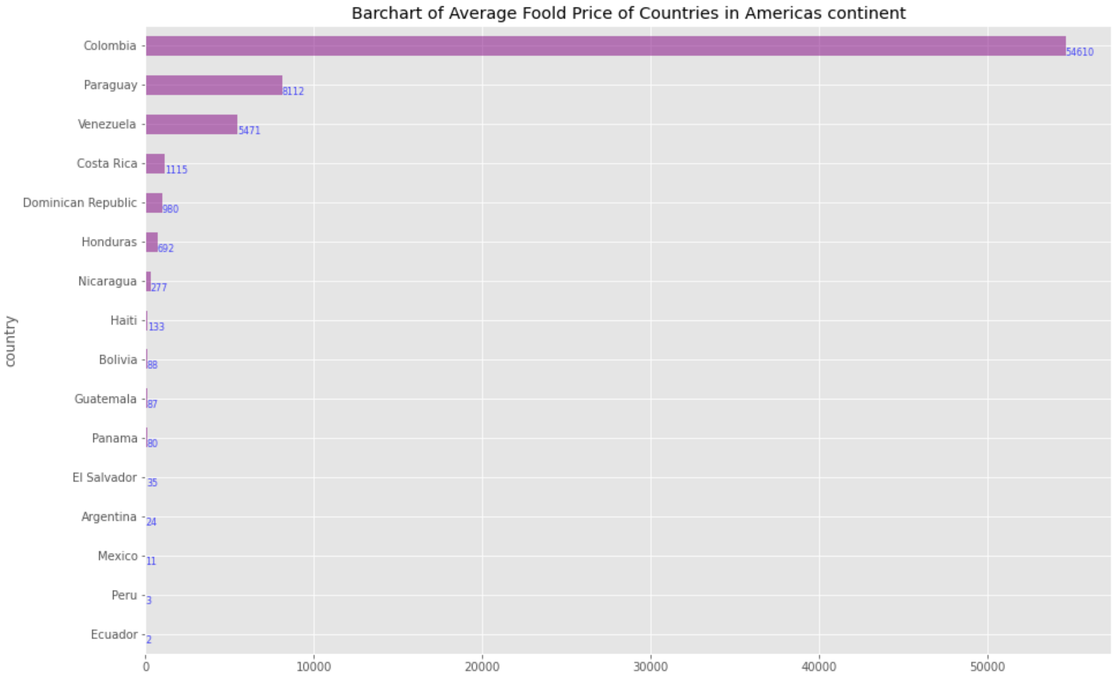
[79]:

```
condition3 = (ref.continent == "Asia") #& (ref.Year == 2020)
plt.ticklabel_format(style = 'plain', useOffset=False)
axr = ref[condition3].groupby("Country of origin")["Refugees under UNHCR's␣
 ↪mandate"].sum().sort_values(ascending = True).plot.barh(figsize = (15,10),
                                alpha = 0.5,color = 'blue', ylabel = "Refugees␣
 ↪under UNHCR's mandate",
                                title="Barchat of Refugees under UNHCR's␣
 ↪mandate of Countries in Asia continents");
for i in axr.patches:
    axr.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
 ↪fontsize=8, color='black', alpha=0.7);
plt.show()
```

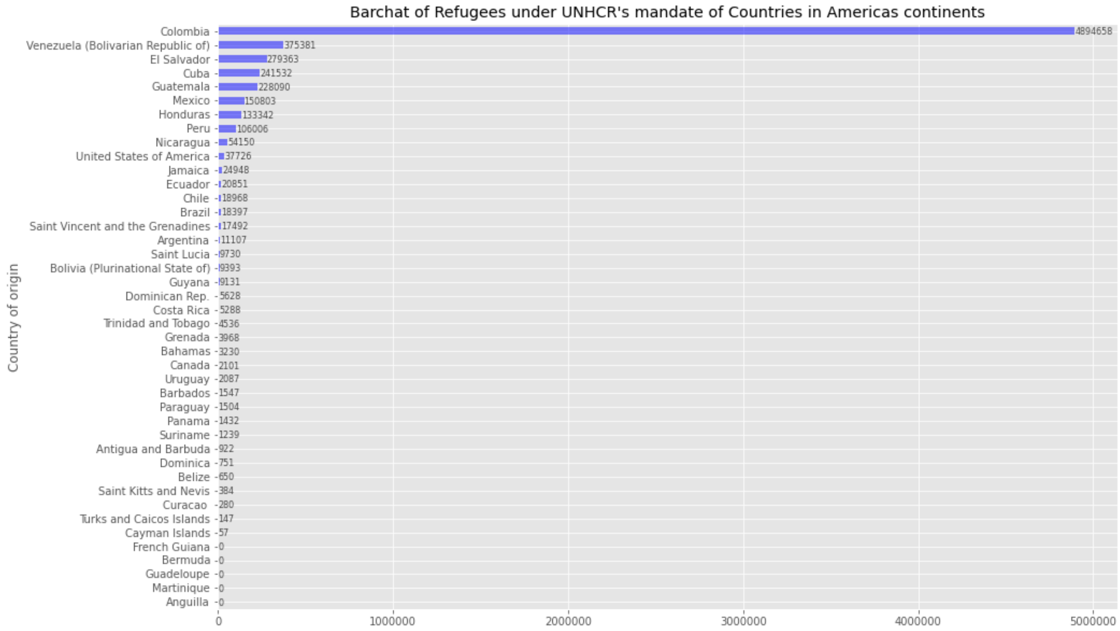


10.3 AMERICAS

```
[80]: cond4 = (gfp.continet == "Americas")# & (gfp.type == "Retail") & (gfp.Year ==␣
      ↪2020))
      plt.ticklabel_format(style='plain',useOffset=False)
      ax4 = gfp[cond4].groupby("country").price.mean().sort_values(ascending = True).
      ↪plot.barh(figsize = (15,10), alpha = 0.5,
      color = 'purple',title = "Barchart of Average Foold Price of Countries in␣
      ↪Americas continent");
      for i in ax4.patches:
          ax4.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
      ↪fontsize=8, color='b', alpha=0.7);
```

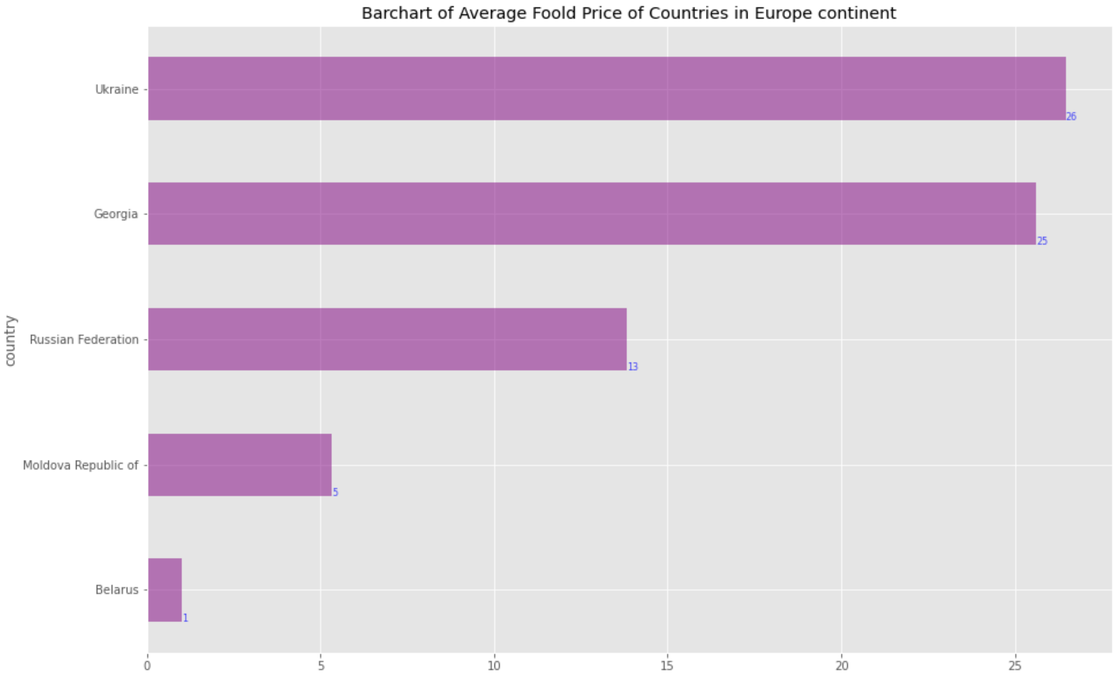


[81]:
```
condition4 = (ref.continent == "Americas") #& (ref.Year == 2020)
plt.ticklabel_format(style = 'plain', useOffset=False)
axr = ref[condition4].groupby("Country of origin")["Refugees under UNHCR's␣
 ↪mandate"].sum().sort_values(ascending = True).plot.barh(figsize = (15,10),
                                alpha = 0.5,color = 'blue', ylabel = "Refugees␣
 ↪under UNHCR's mandate",
                                title="Barchat of Refugees under UNHCR's␣
 ↪mandate of Countries in Americas continents");
for i in axr.patches:
    axr.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
 ↪fontsize=8, color='black', alpha=0.7);
plt.show()
```

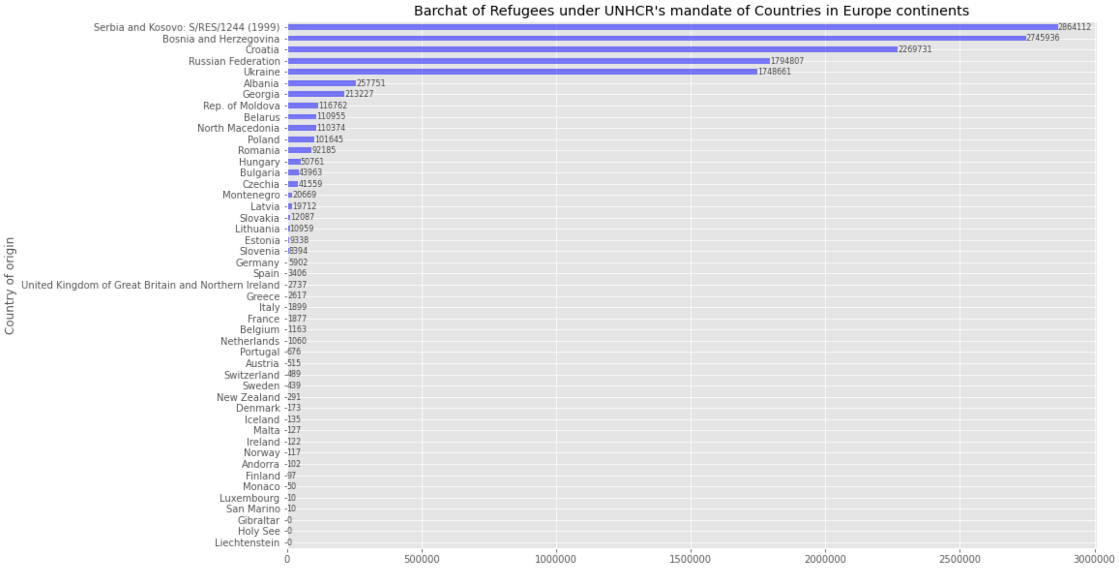


10.4 EUROPE

[82]:
```
cond5 = (gfp.continet == "Europe")# & (gfp.type == "Retail") & (gfp.Year == 2020))
plt.ticklabel_format(style='plain',useOffset=False)
ax4 = gfp[cond5].groupby("country").price.mean().sort_values(ascending = True).plot.barh(figsize = (15,10), alpha = 0.5,
color = 'purple',title = "Barchart of Average Foold Price of Countries in Europe continent");
for i in ax4.patches:
    ax4.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))), fontsize=8, color='b', alpha=0.7);
```



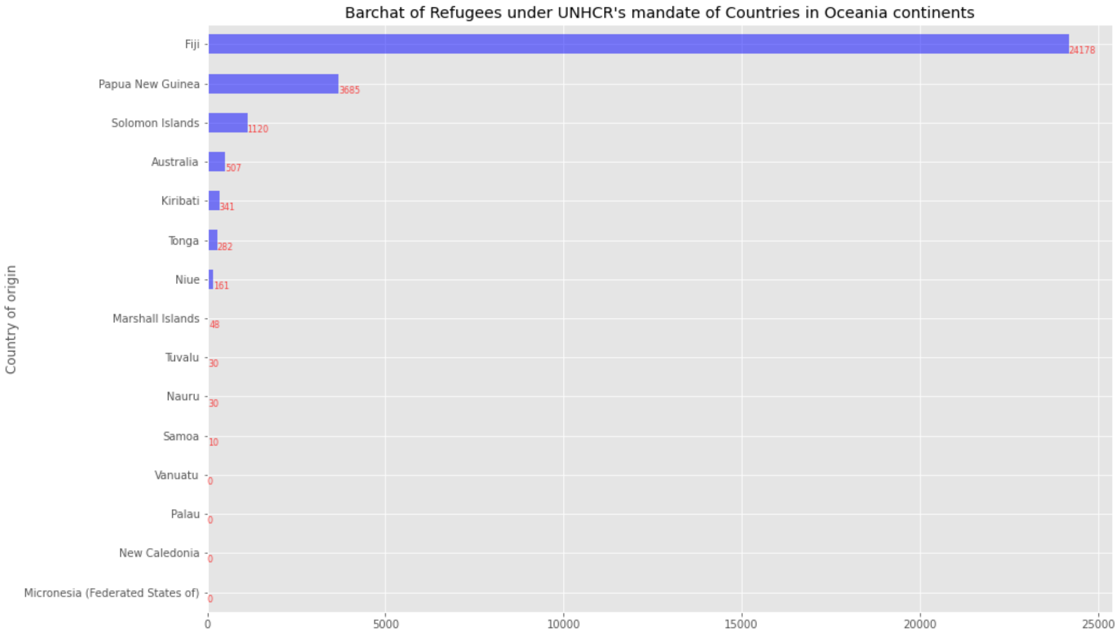
```
[83]: condition5 = (ref.continent == "Europe") #& (ref.Year == 2020)
      plt.ticklabel_format(style = 'plain', useOffset=False)
      axr = ref[condition5].groupby("Country of origin")["Refugees under UNHCR's␣
       ↪mandate"].sum().sort_values(ascending = True).plot.barh(figsize = (15,10),
                                      alpha = 0.5,color = 'blue', ylabel = "Refugees␣
       ↪under UNHCR's mandate",
                                      title="Barchat of Refugees under UNHCR's␣
       ↪mandate of Countries in Europe continents");
      for i in axr.patches:
          axr.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
       ↪fontsize=8, color='black', alpha=0.7);
      plt.show()
```



10.5 OCEANIA

- GLOBAL FOOD PRICE DATASET DIDN'T CAPTURE OCEANIA COUNTRIES

[84]:
```
condition6 = (ref.continent == "Oceania") #& (ref.Year == 2020)
plt.ticklabel_format(style = 'plain', useOffset=False)
axr = ref[condition6].groupby("Country of origin")["Refugees under UNHCR's␣
↪mandate"].sum().sort_values(ascending = True).plot.barh(figsize = (15,10),
                                                alpha = 0.5,color = 'blue', ylabel = "Refugees␣
↪under UNHCR's mandate",
                                                title="Barchat of Refugees under UNHCR's␣
↪mandate of Countries in Oceania continents");
for i in axr.patches:
    axr.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
↪fontsize=8, color='red', alpha=0.7);
plt.show()
```

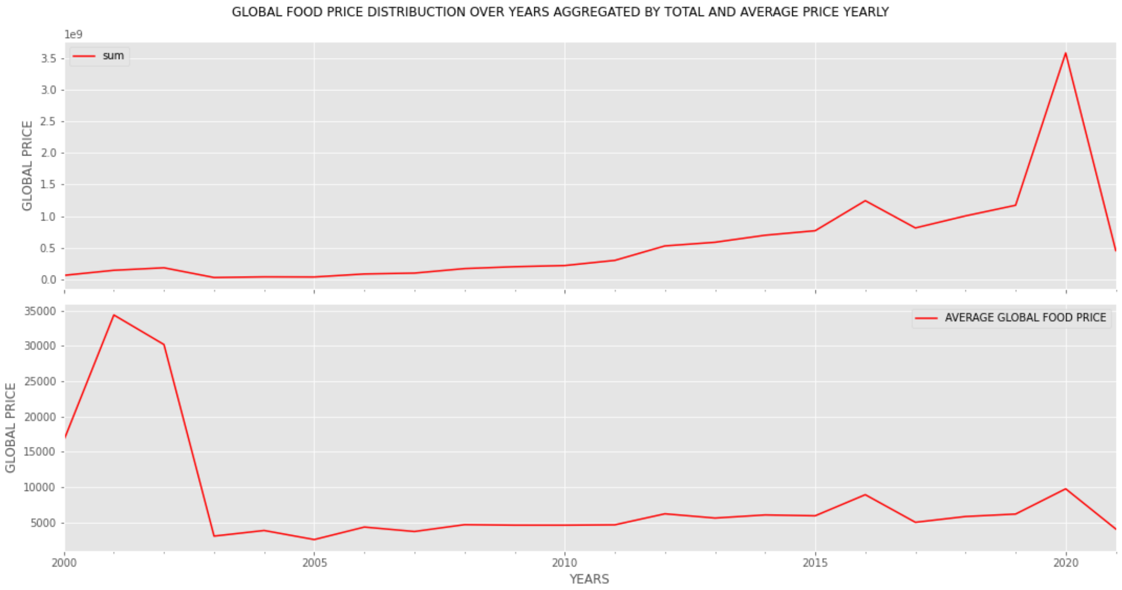


```
[106]: from datetime import datetime
       gfp['date'] = pd.to_datetime(gfp['date'])
       gfp_date = gfp.set_index("date")
       gfp_date.sort_values(by = ["date"], inplace = True)
       gfp_date.head(2)
```

```
[106]:                country       city     market currency       type    unit  \
       date
       2000-01-01  Bangladesh      Dhaka      Dhaka      BDT  Wholesale  100 KG
       2000-01-01    Colombia  Antioquia   Medellin      COP  Wholesale      KG

                     month  Year    price                              product  \
       date
       2000-01-01  January  2000  1138.80  Rice (coarse, BR-8/ 11/, Guti Sharna)
       2000-01-01  January  2000   430.63                       Maize (yellow)

                   continet
       date
       2000-01-01      Asia
       2000-01-01  Americas
```

```
[107]: gfp_date['price'].resample('Y', origin = 0).agg(['sum','mean']).plot(title =␣
        ↪'GLOBAL FOOD PRICE DISTRIBUCTION OVER YEARS AGGREGATED BY TOTAL AND AVERAGE␣
        ↪PRICE YEARLY',
               subplots = True, figsize = (15,8), ylabel = 'GLOBAL PRICE', color =␣
        ↪'red', xlabel = "YEARS");
```

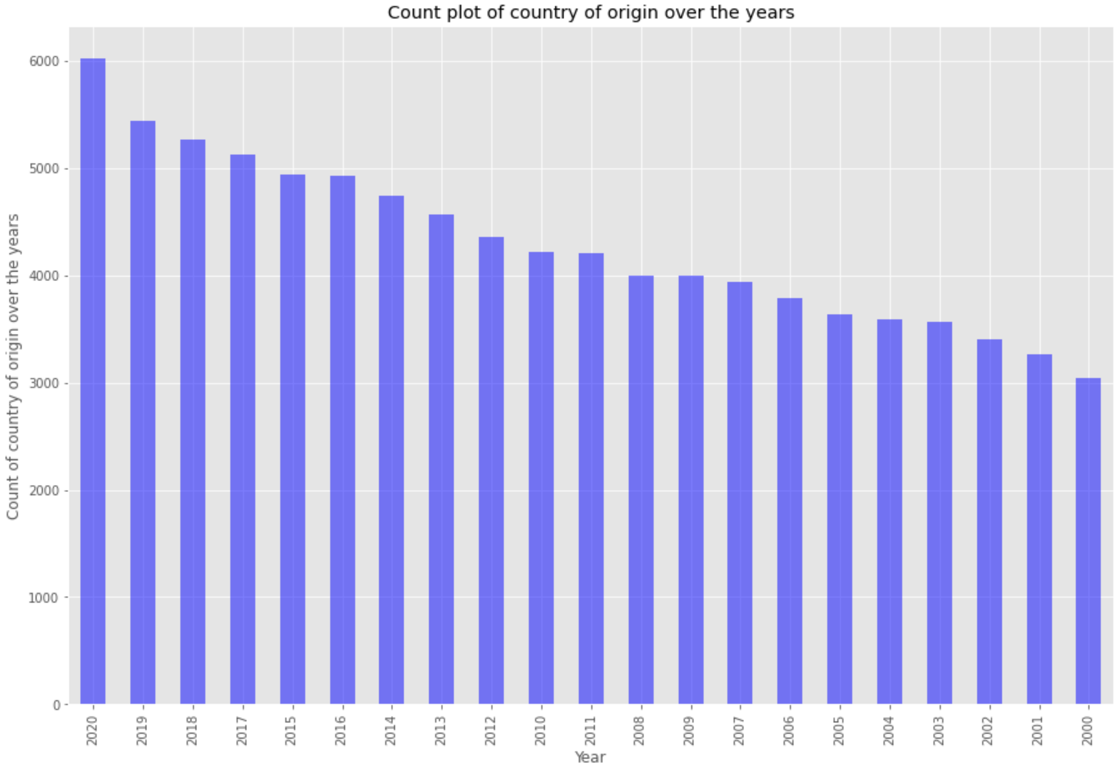
```
plt.legend(["AVERAGE GLOBAL FOOD PRICE"])
plt.tight_layout()
plt.show()
```



[108]:
```
# ["Eggs","Yam","Milk (UHT)", "Plantains (apentu)", "Fish (fresh)","Water␣
↪(drinking)","Bananas",
# "Oil (vegetable, imported)", "Oil (palm)","Livestock (Goat)", "Livestock␣
↪(Sheep)", "Livestock (donkey)",
#  "Livestock (camel)", "Livestock (ox)", "Livestock (bull)", "Cabbage",␣
↪"Lettuce","Livestock (cattle)","Coconut",
# "Bread", "Wheat","Sugar","Salt","Rice","Potatoes"," Tomatoes", "Meat (beef)",␣
↪"Carrots", "Onions","Tea",
#  "Apples","Oranges","Beans","Milk","Oil","Oil (palm)","Bulgur","Bread␣
↪(brown)", "Bread (rye)", "Bread (wheat)",
# "Lentils","Straw","Milling cost (sorghum)", "Milling cost␣
↪(maize)","Sorghum","Laundry detergent","Avocados",
# "Spinach", "Parsley", "Noodles (instant, indomie)","Lettuce", "Potato␣
↪Leaves", "Cassava leaves", "Charcoal"]
```

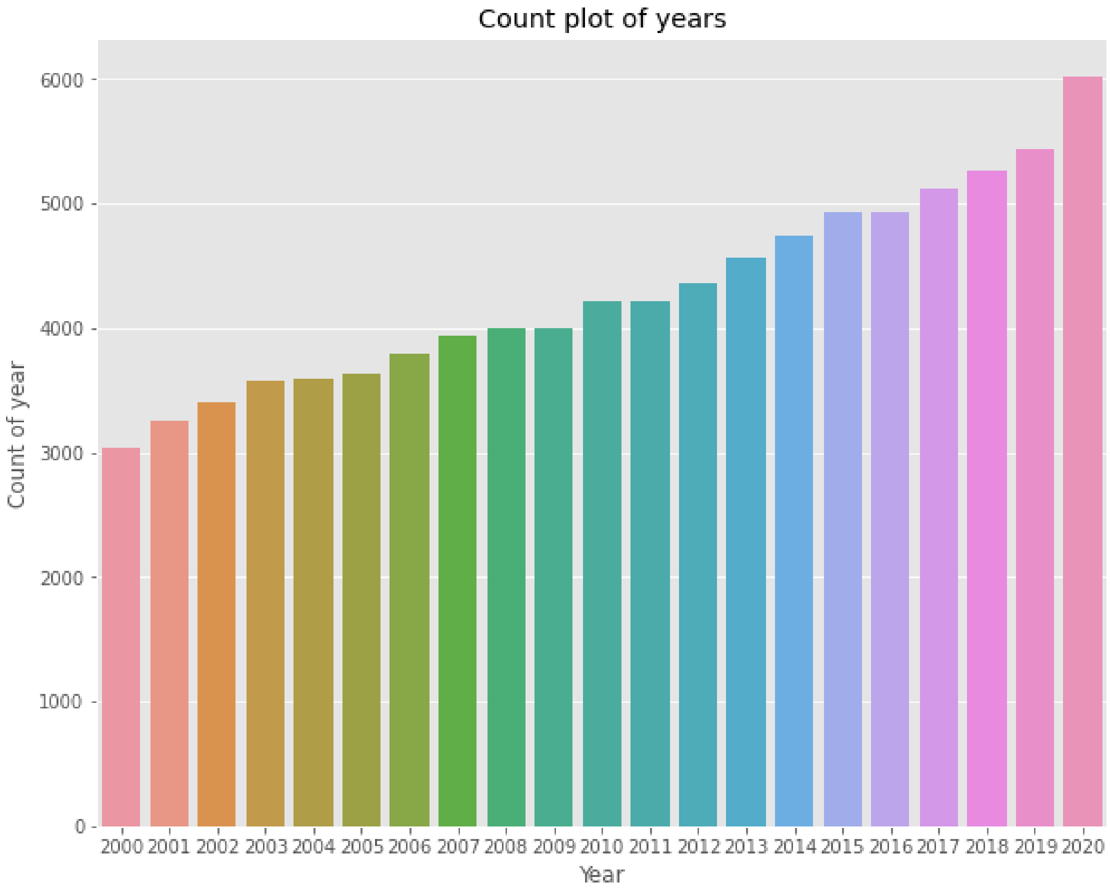
[109]:
```
# ['KG','Unit','Packet','Pounds','10 pcs','L','Cubic meter','Dozen','Cuartilla',
#  'Libra','Sack', 'Package', 'Head', 'MT','Bunch','Marmite', 'Gallon','200␣
↪ML','Loaf',
# 'Pile', 'Heap','Bundle', 'LCU/3.5kg','100 Tubers']
```

[110]:
```
ref.groupby("Year")["Country of origin"].count().sort_values(ascending =False).
↪plot.bar(figsize = (15,10), alpha = 0.5,
                                         color = 'blue', ylabel = "Count of␣
↪country of origin over the years",
                                         title = "Count plot of country of␣
↪origin over the years");
```



[111]:

```
plt.figure(figsize = (10,8))
sns.countplot(x = "Year", data = ref)
plt.ylabel("Count of year")
plt.title("Count plot of years")
plt.show()
```



10.6 Country of origin

*The country of origin feature represents the country where those seeking refugee are coming from.

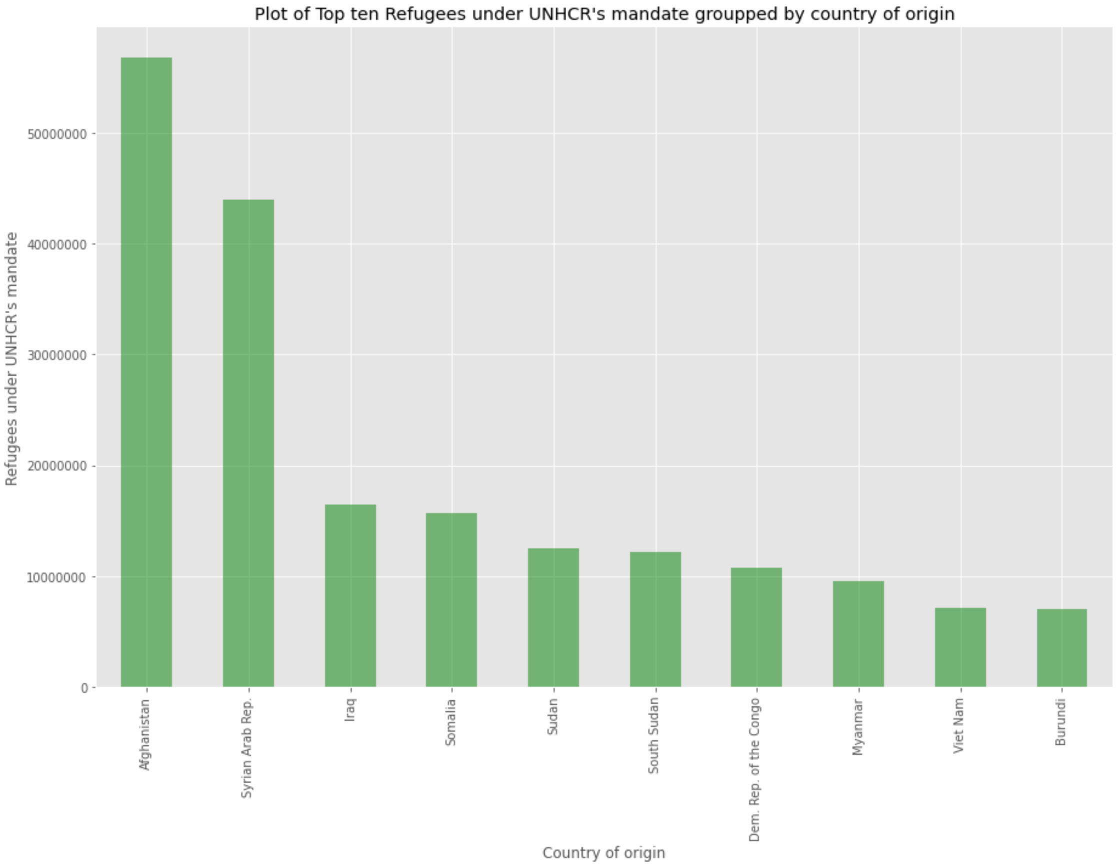
- Somalia is country with the highest number in terms of refugees country's of origin.
- There are 212 countries where refugees originated.
- The data didn't give any information why people from these countries are seeking refugee and asyluum in other countries, but we know that most of these countries are in war. Countries like Somalia, Afghanistan, Syria etc.
- Refugees from Afghanistan as country of origin has the highest number of Refugees under UNHCR's mandate

```
[112]: ref["Country of origin"].describe(include = 'all')
```

```
[112]: count        90004
       unique         212
       top        Somalia
       freq          1990
       Name: Country of origin, dtype: object
```

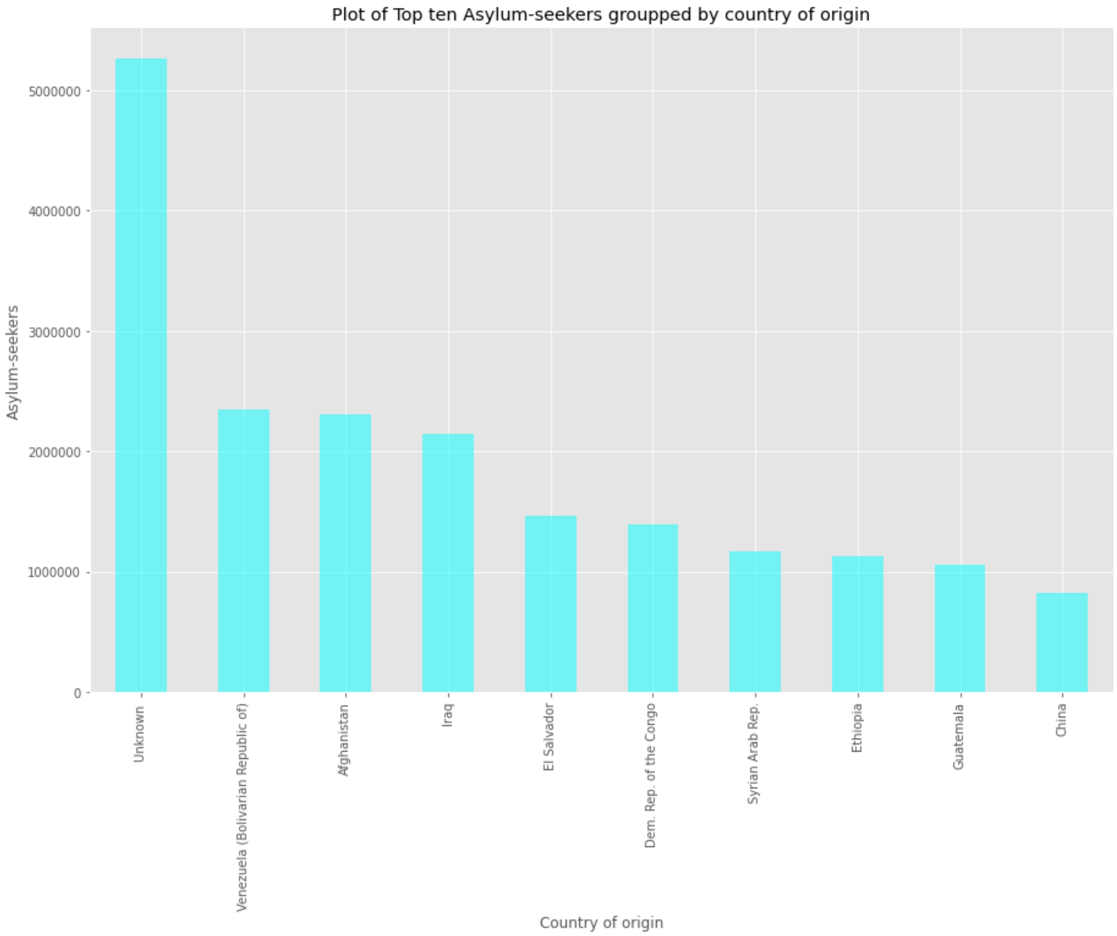
[113]:

```
"""
Afghanistan has the number of Refugees under UNHCR's mandate
"""
plt.ticklabel_format(style='plain',useOffset=False)
ref.groupby("Country of origin")["Refugees under UNHCR's mandate"].sum().
 ↪sort_values(ascending =False).head(10).plot.bar(figsize = (15,10), alpha = 0.
 ↪5,
                            color = 'green', ylabel = "Refugees under UNHCR's␣
 ↪mandate",
                                                           title="Plot of Top ten␣
 ↪Refugees under UNHCR's mandate groupped by country of origin");
```



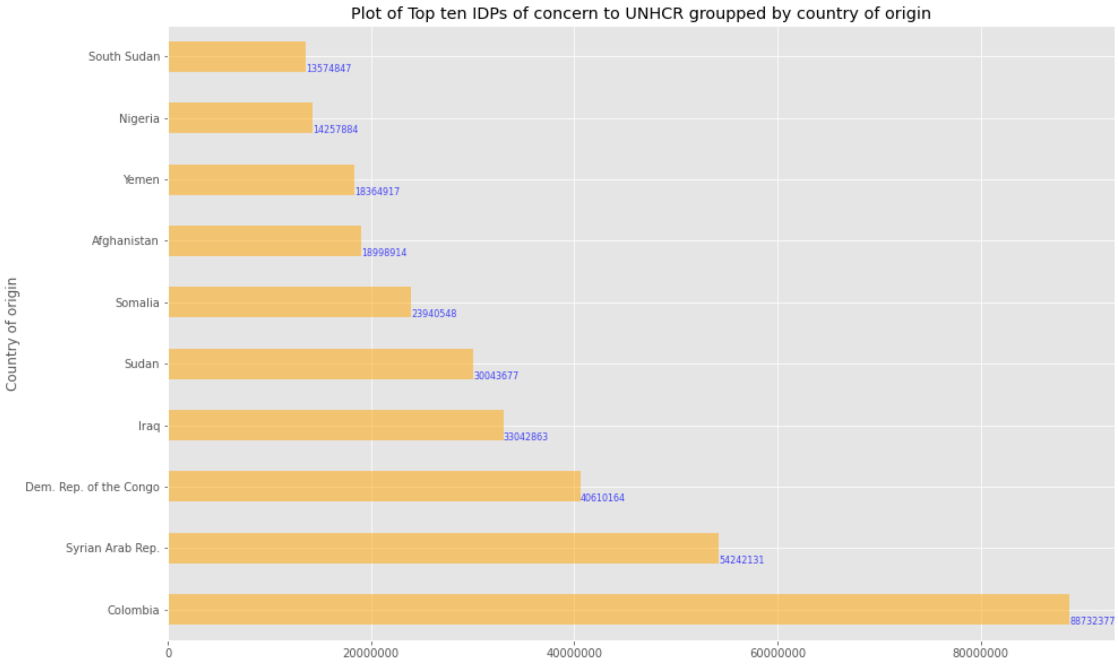
[114]:

```
"""
Asyluum seekers from unknown countries are highest followed by Venezuela
"""
plt.ticklabel_format(style='plain',useOffset=False)
ref.groupby("Country of origin")["Asylum-seekers"].sum().sort_values(ascending␣
 ↪= False).head(10).plot.bar(figsize = (15,10), alpha = 0.5,
                            color = 'cyan', ylabel = "Asylum-seekers",
                                    title="Plot of Top ten Asylum-seekers␣
↪groupped by country of origin");
```



[115]:

```
"""
Colombia has the highest number of IDPs of concern to UNHCR based on country of␣
↪origin
"""
plt.ticklabel_format(style='plain',useOffset=False)
ax2 = ref.groupby("Country of origin")["IDPs of concern to UNHCR"].sum().
↪sort_values(ascending = False).head(10).plot.barh(figsize = (15,10), alpha =␣
↪0.5,
                            color = 'orange', ylabel = "IDPs of concern to␣
↪UNHCR",
                                        title="Plot of Top ten IDPs of␣
↪concern to UNHCR groupped by country of origin");
for i in ax2.patches:
    ax2.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
↪fontsize=8, color='b', alpha=0.7);
```



```
[116]: ref.columns
```

```
[116]: Index(['Year', 'Country of origin', 'Country of origin (ISO)',
              'Country of asylum', 'Country of asylum (ISO)',
              'Refugees under UNHCR's mandate', 'Asylum-seekers',
              'IDPs of concern to UNHCR', 'Venezuelans displaced abroad',
              'Stateless persons', 'Others of concern', 'continent'],
             dtype='object')
```

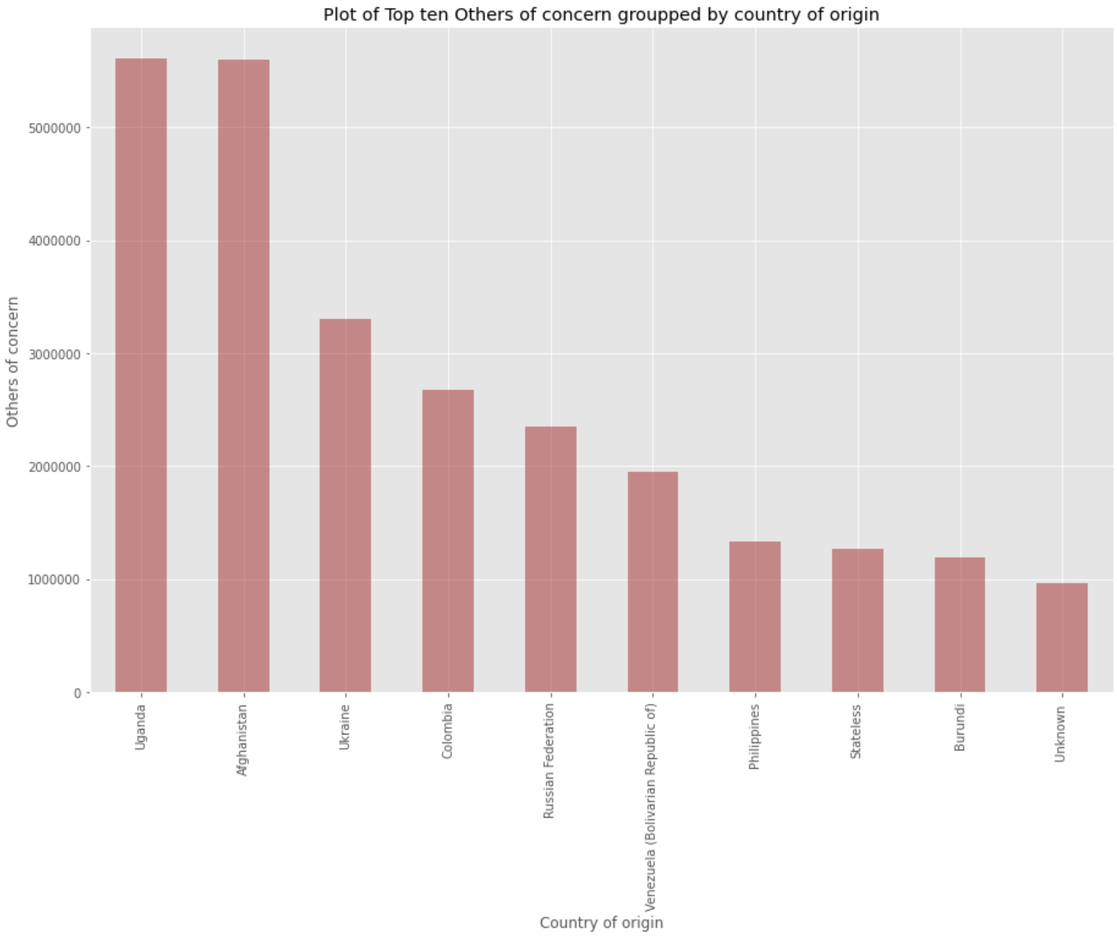
```
[117]: ref.groupby("Country of origin")["Venezuelans displaced abroad"].sum().
        ↪sort_values(ascending = False).head(10)
```

```
[117]: Country of origin
       Venezuela (Bolivarian Republic of)    10031476.00
       Afghanistan                                  0.00
       Panama                                       0.00
       New Zealand                                  0.00
       Nicaragua                                    0.00
       Niger                                        0.00
       Nigeria                                      0.00
       Niue                                         0.00
       North Macedonia                              0.00
       Norway                                       0.00
       Name: Venezuelans displaced abroad, dtype: float64
```

```
[118]: ref.groupby("Country of origin")["Stateless persons"].sum().
        ↪sort_values(ascending = False).head(10)
```

```
[118]: Country of origin
       Stateless          65929313
       Afghanistan               0
       New Caledonia             0
       Nicaragua                 0
       Niger                     0
       Nigeria                   0
       Niue                      0
       North Macedonia           0
       Norway                    0
       Oman                      0
       Name: Stateless persons, dtype: int64
```

```
[119]: plt.ticklabel_format(style='plain',useOffset=False)
       ref.groupby("Country of origin")["Others of concern"].sum().
        ↪sort_values(ascending = False).head(10).plot.bar(figsize = (15,10), alpha =␣
        ↪0.5,
                                    color = 'brown', ylabel = "Others of concern",
                                    title="Plot of Top ten Others of concern groupped␣
        ↪by country of origin");
```



10.7 Country of asylum & Asylum-seekers

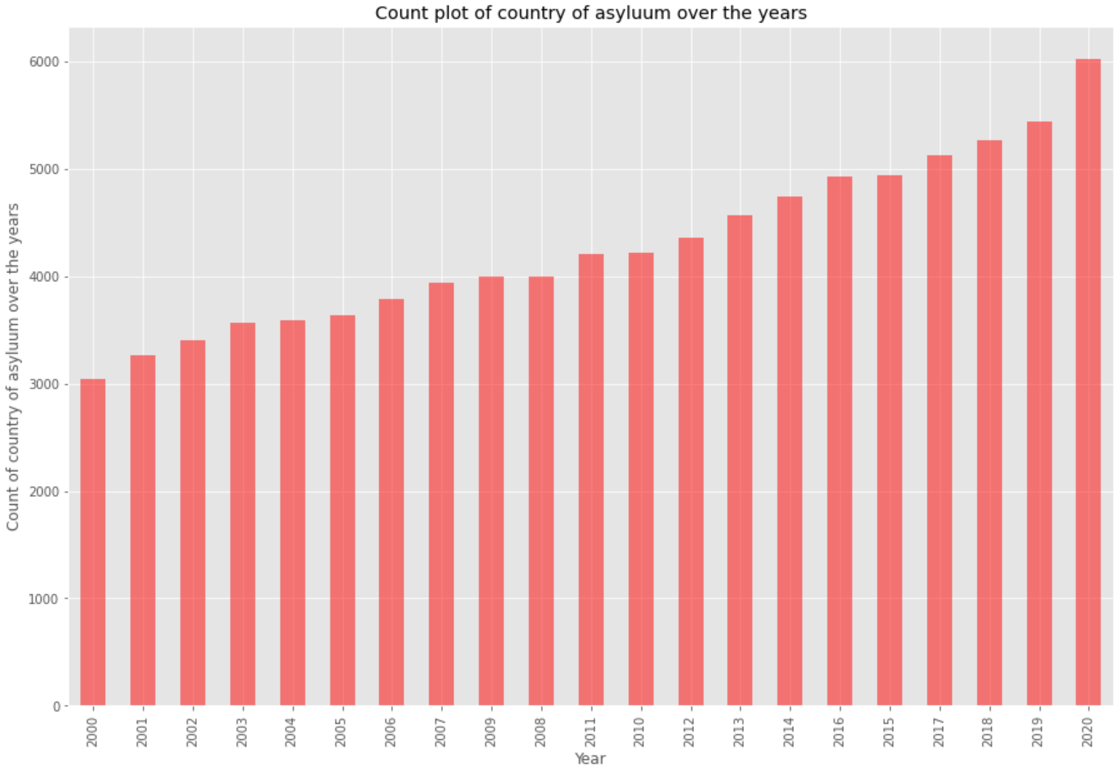
The Country of asylum represent the country where people from various countries of origin are seeking for asyluum outside their home country. There are 189 country of asyluum seekers * United States of America has the highest number of asyluum seekers with 3572 frequency.

```
[120]: ref[["Country of asylum", "Asylum-seekers"]].describe(include = 'all')
```

```
[120]:             Country of asylum  Asylum-seekers
       count                   90004        90004.00
       unique                    189             NaN
       top      United States of America             NaN
       freq                     3572             NaN
       mean                      NaN          393.77
       std                       NaN         5586.88
       min                       NaN            0.00
       25%                       NaN            0.00
       50%                       NaN            6.00
```

```
75%                              NaN          41.00
max                              NaN      940668.00
```

[121]:
```
"""
Year 2020 has the highest number of people seeking asyluum in various countries␣
↪represented
"""
ref.groupby("Year")["Country of asylum"].count().sort_values(ascending =True).
↪plot.bar(figsize = (15,10), alpha = 0.5,
                                        color = 'red', ylabel = "Count of␣
↪country of asyluum over the years",
                                            title="Count plot of country of␣
↪asyluum over the years");
```

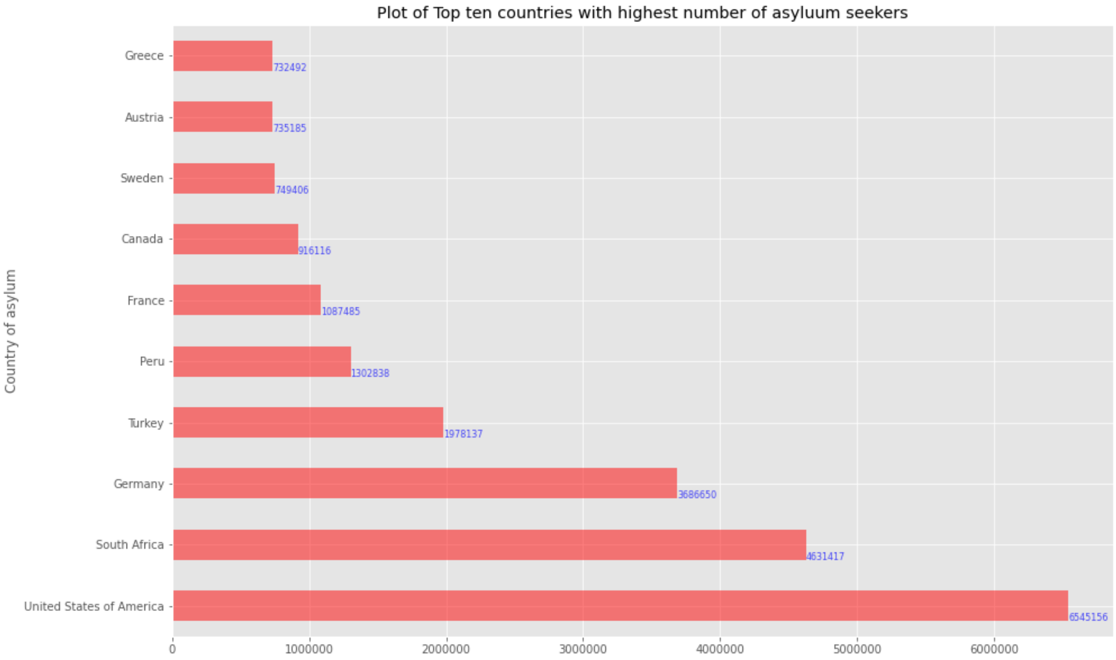


[122]:
```
ref.groupby("Country of asylum")["Asylum-seekers"].sum().sort_values(ascending␣
↪= False).head(10)
```

[122]:
```
Country of asylum
United States of America    6545156
South Africa                4631417
Germany                     3686650
Turkey                      1978137
```

```
Peru                        1302838
France                      1087485
Canada                       916116
Sweden                       749406
Austria                      735185
Greece                       732492
Name: Asylum-seekers, dtype: int64
```

[123]:
```
plt.ticklabel_format(style='plain',useOffset=False)
ax1 = ref.groupby("Country of asylum")["Asylum-seekers"].sum().
 ↪sort_values(ascending = False).head(10).plot.barh(figsize = (15,10), alpha =␣
 ↪0.5,
                        color = 'red', ylabel = "Asyluum seekers",
                                        title="Plot of Top ten countries with␣
 ↪highest number of asyluum seekers");
for i in ax1.patches:
    ax1.text(i.get_width()+0.005, i.get_y(), str(int(round(i.get_width(),2))),␣
 ↪fontsize=8, color='b', alpha=0.7);
```



10.8 Refugees under UNHCR's mandate

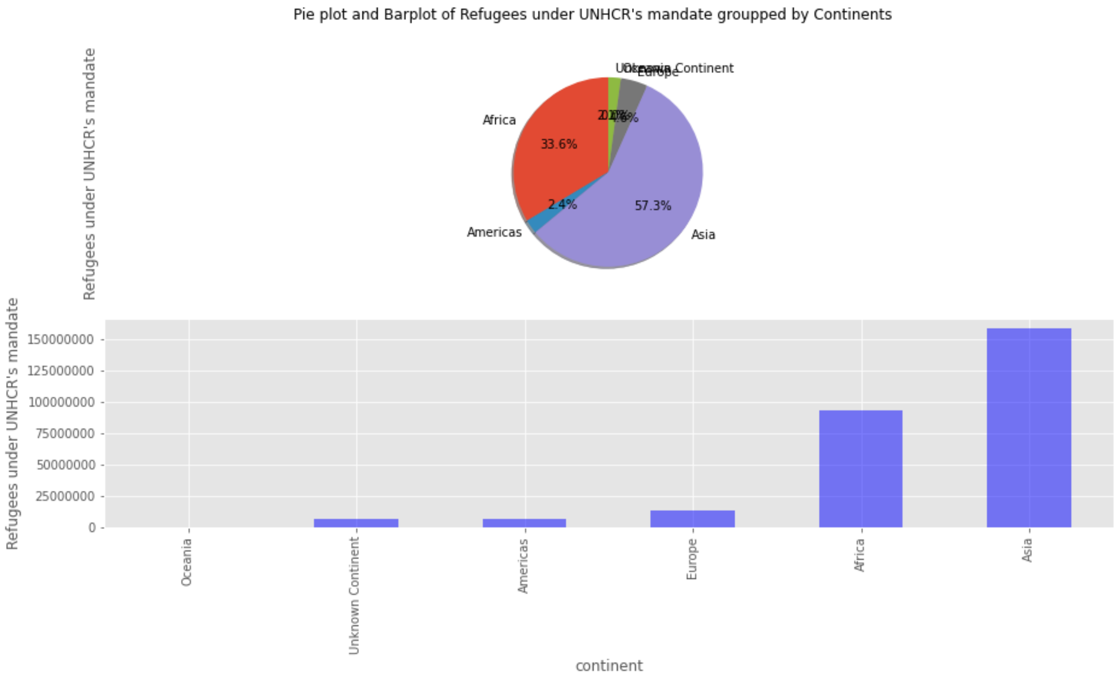
[124]:
```
ref["Refugees under UNHCR's mandate"].describe(include = 'all')
```

[124]:
```
count     90004.00
mean       3077.74
```

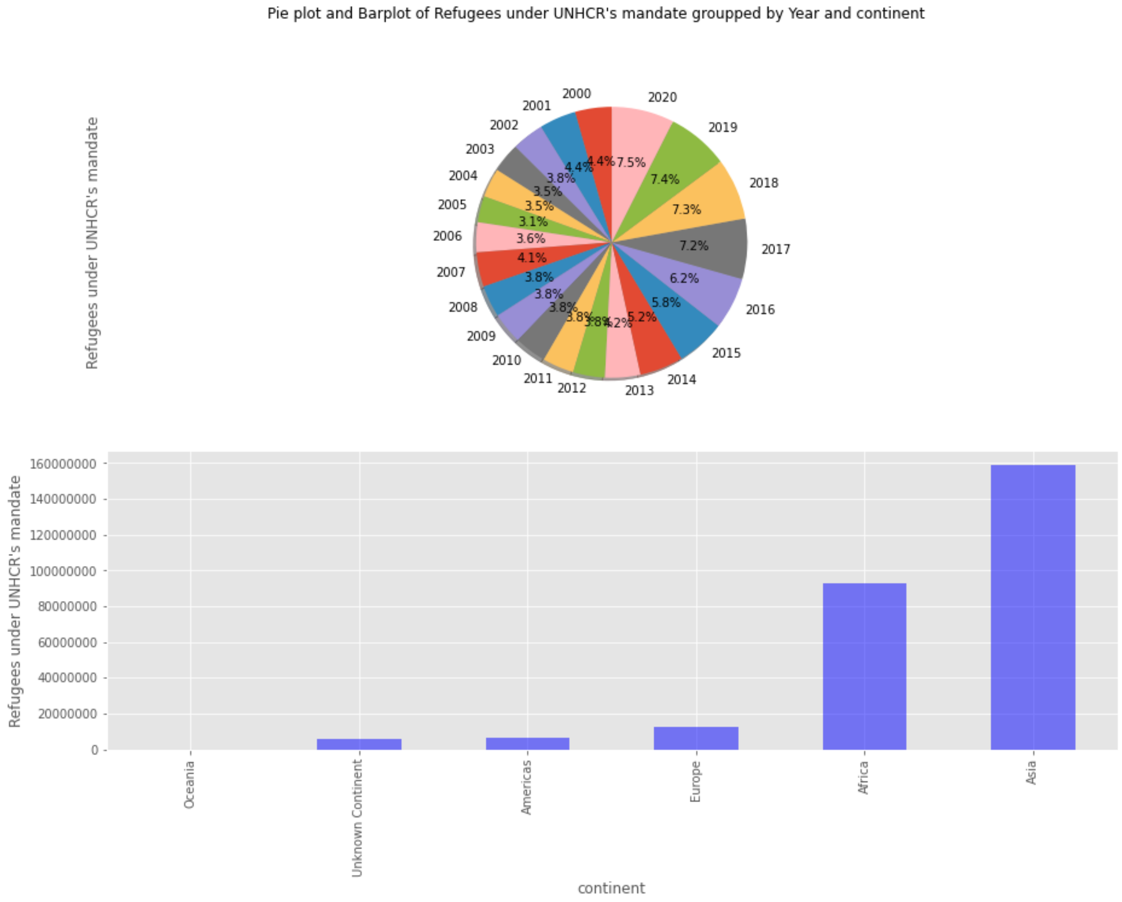
```
std        46313.22
min            0.00
25%            5.00
50%           14.00
75%          103.00
max      3641370.00
Name: Refugees under UNHCR's mandate, dtype: float64
```

[125]:

```
plt.suptitle("Pie plot and Barplot of Refugees under UNHCR's mandate groupped␣
 ↪by Continents")
plt.subplot(2,1,1)
ax = ref.groupby('continent')["Refugees under UNHCR's mandate"].sum().plot.
 ↪pie(autopct = "%1.1f%%",
                                                    shadow = True, startangle =␣
 ↪90, figsize = (15,7))
ax.axis("equal")
plt.subplot(2,1,2)
plt.ticklabel_format(style = 'plain', useOffset=False)
ax1 = ref.groupby('continent')["Refugees under UNHCR's mandate"].sum().
 ↪sort_values(ascending = True).plot.bar(figsize = (15,7),
                                alpha = 0.5,color = 'blue', ylabel = "Refugees␣
 ↪under UNHCR's mandate");
plt.show()
```



[126]:
```
plt.suptitle("Pie plot and Barplot of Refugees under UNHCR's mandate groupped␣
 ↪by Year and continent")
plt.subplot(2,1,1)
ax = ref.groupby('Year')["Refugees under UNHCR's mandate"].sum().plot.
 ↪pie(autopct = "%1.1f%%", shadow = True, startangle = 90, figsize = (15,10))
ax.axis("equal")
plt.subplot(2,1,2)
plt.ticklabel_format(style = 'plain', useOffset=False)
ax1 = ref.groupby('continent')["Refugees under UNHCR's mandate"].sum().
 ↪sort_values(ascending = True).plot.bar(figsize = (15,10), alpha = 0.5,
                                        color = 'blue', ylabel = "Refugees␣
 ↪under UNHCR's mandate");
plt.show()
```



[127]:
```
ref.columns
```

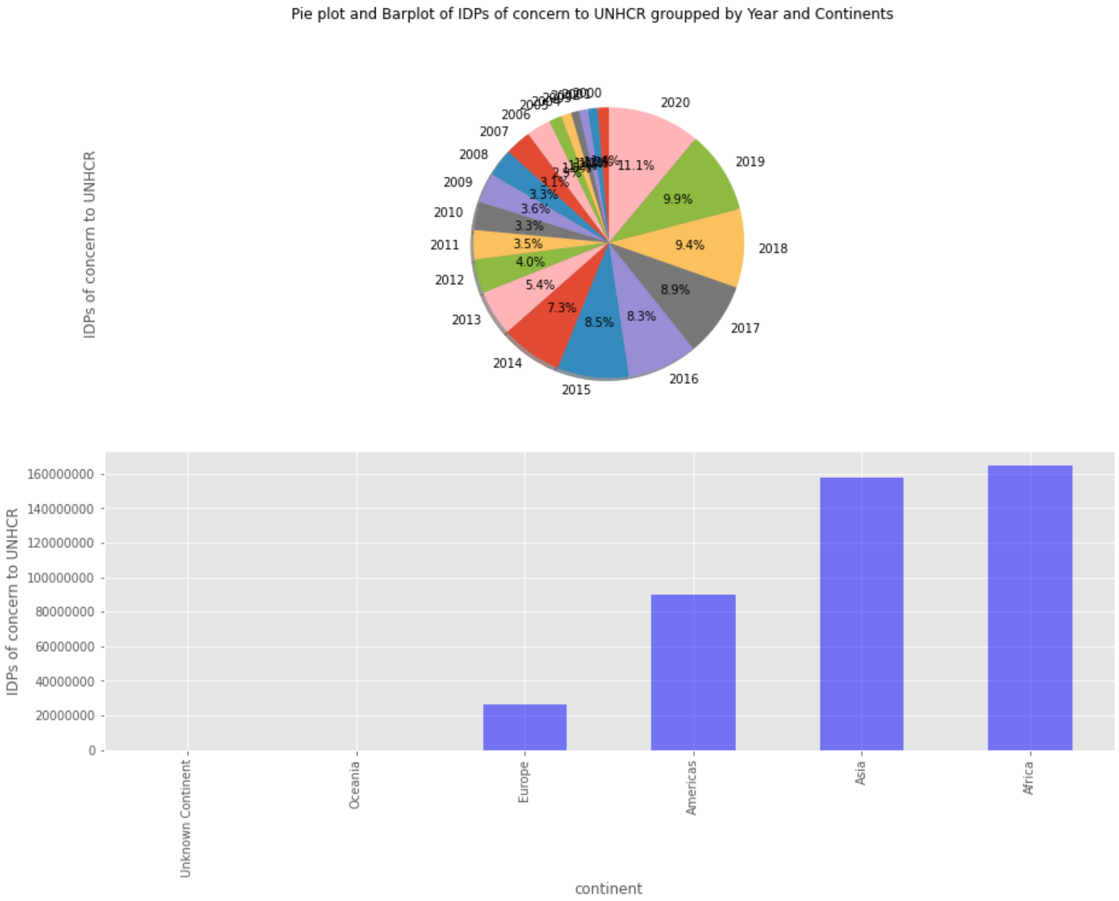
[127]:
```
Index(['Year', 'Country of origin', 'Country of origin (ISO)',
       'Country of asylum', 'Country of asylum (ISO)',
       'Refugees under UNHCR's mandate', 'Asylum-seekers',
       'IDPs of concern to UNHCR', 'Venezuelans displaced abroad',
       'Stateless persons', 'Others of concern', 'continent'],
      dtype='object')
```

10.9 IDPs of concern to UNHCR

```
[128]: ref["IDPs of concern to UNHCR"].describe(include = 'all')
```

```
[128]: count      90004.00
       mean        4881.33
       std       124509.91
       min            0.00
       25%            0.00
       50%            0.00
       75%            0.00
       max      8252788.00
       Name: IDPs of concern to UNHCR, dtype: float64
```

```
[129]: plt.suptitle("Pie plot and Barplot of IDPs of concern to UNHCR groupped by Year␣
        ↪and Continents")
       plt.subplot(2,1,1)
       ax = ref.groupby('Year')["IDPs of concern to UNHCR"].sum().plot.pie(autopct =␣
        ↪"%1.1f%%",
                                                                   shadow = True, startangle =␣
        ↪90, figsize = (15,10))
       ax.axis("equal")
       plt.subplot(2,1,2)
       plt.ticklabel_format(style = 'plain', useOffset=False)
       ax1 = ref.groupby('continent')["IDPs of concern to UNHCR"].sum().
        ↪sort_values(ascending = True).plot.bar(figsize = (15,10),
                                               alpha = 0.5,color = 'blue', ylabel = "IDPs of␣
        ↪concern to UNHCR");
       plt.show()
```

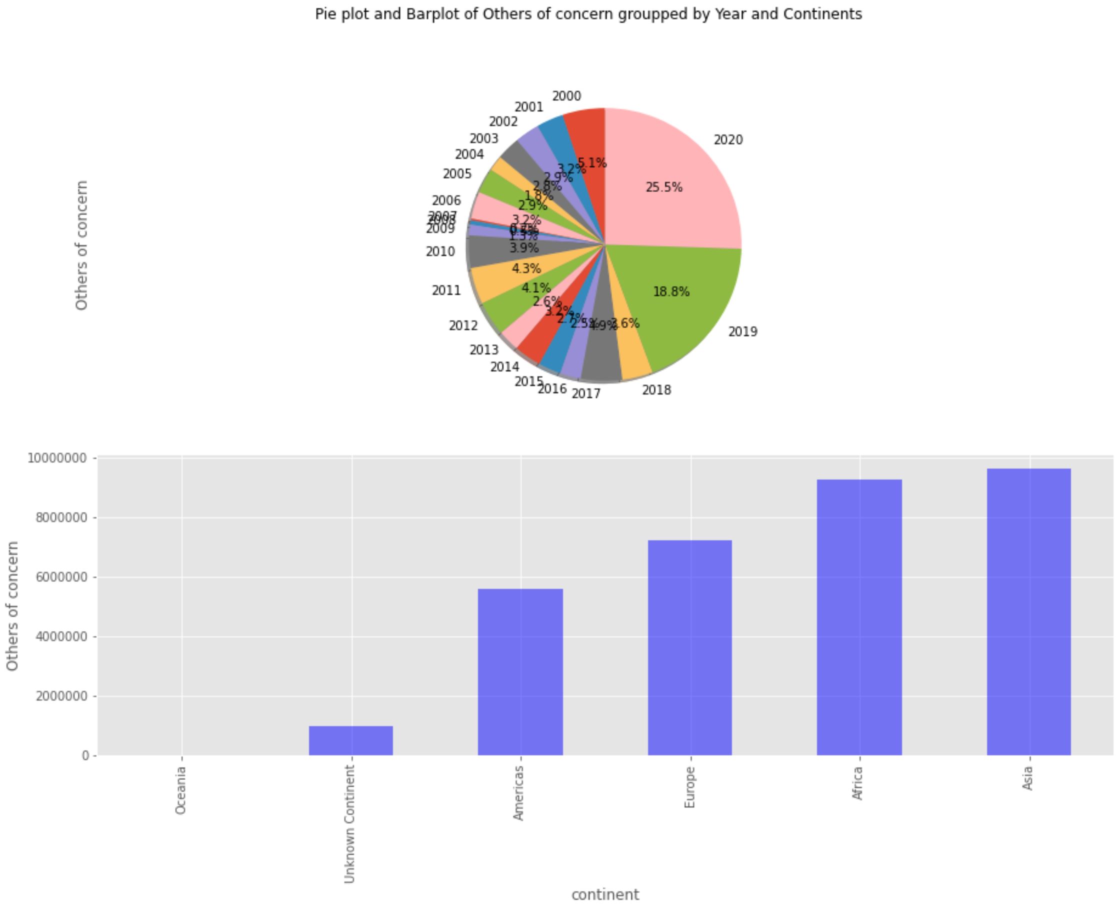


10.10 Others of concern

```
[130]: ref["Others of concern"].describe(include = 'all')
```

```
[130]: count      90004.00
       mean         362.10
       std        16814.69
       min            0.00
       25%            0.00
       50%            0.00
       75%            0.00
       max      2351313.00
       Name: Others of concern, dtype: float64
```

```
[131]: plt.suptitle("Pie plot and Barplot of Others of concern groupped by Year and␣
        ↪Continents")
       plt.subplot(2,1,1)
       ax = ref.groupby('Year')["Others of concern"].sum().plot.pie(autopct = "%1.
        ↪1f%%",
                                                              shadow = True, startangle =␣
        ↪90, figsize = (15,10))
       ax.axis("equal")
       plt.subplot(2,1,2)
       plt.ticklabel_format(style = 'plain', useOffset=False)
       ax1 = ref.groupby('continent')["Others of concern"].sum().sort_values(ascending␣
        ↪= True).plot.bar(figsize = (15,10),
                                                  alpha = 0.5,color = 'blue', ylabel = "Others of␣
        ↪concern");
       plt.show()
```



11 Conclusion

The EDA shows a true picture of what we see in our world. The Asia world battling a lot of wars comes out top in term of refugees and asyluum seekers. Though, there was no exact data information on relation of global food prices to refuee movement, but war, civil unrest comes with hunger. This makes people leave their country of origin to another country as a refugee or asylum seeker.Exploration of global food price data shows there is strong high food prices in ASIA AND

AFRICA. The countries that comes top are war countries of Afganistan in Asia and Somalia in Africa. This shows there is a relationship between refugee movements and food prices. Therefore, I can categorically say there is a positive correlation with refugee movement to food prices. If there is conflicts or war, there will be lack of food and even if there is food, it will scarce and costly. This will encourage migration of people to avoid dying of starvation.