PH_Food_Prices

April 1, 2024

1 PH Food Prices (2000-2023)

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

1.0.1 Read the file and analyze its data

[2]:
```
# Dataset from https://data.humdata.org/dataset/wfp-food-prices-for-philippines
food_prices = pd.read_csv('ph_food_prices.csv')
food_prices.head()
```

[2]:

	date	region	city/province	market	latitude	longitude	category	commodity	unit	priceflag	pricetype	currency	price	usdprice
0	15/01/2000	National Capital region	Metropolitan Manila	Metro Manila	14.604167	120.982222	cereals and tubers	Maize flour (yellow)	KG	actual	Retail	PHP	15.00	0.3717
1	15/01/2000	National Capital region	Metropolitan Manila	Metro Manila	14.604167	120.982222	cereals and tubers	Rice (milled, superior)	KG	actual	Retail	PHP	20.00	0.4957
2	15/01/2000	National Capital region	Metropolitan Manila	Metro Manila	14.604167	120.982222	cereals and tubers	Rice (milled, superior)	KG	actual	Wholesale	PHP	18.35	0.4548
3	15/01/2000	National Capital region	Metropolitan Manila	Metro Manila	14.604167	120.982222	cereals and tubers	Rice (regular, milled)	KG	actual	Retail	PHP	18.00	0.4461
4	15/01/2000	National Capital region	Metropolitan Manila	Metro Manila	14.604167	120.982222	cereals and tubers	Rice (regular, milled)	KG	actual	Wholesale	PHP	16.35	0.4052

[3]:
```
food_prices.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 141765 entries, 0 to 141764
Data columns (total 14 columns):
```

```
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   date           141765 non-null  object
 1   region         141765 non-null  object
 2   city/province  141765 non-null  object
 3   market         141765 non-null  object
 4   latitude       141765 non-null  float64
 5   longitude      141765 non-null  float64
 6   category       141765 non-null  object
 7   commodity      141765 non-null  object
 8   unit           141765 non-null  object
 9   priceflag      141765 non-null  object
 10  pricetype      141765 non-null  object
 11  currency       141765 non-null  object
 12  price          141765 non-null  float64
 13  usdprice       141765 non-null  float64
dtypes: float64(4), object(10)
memory usage: 15.1+ MB
```

[4]:
```
food_prices.describe(include='all')
```

[4]:

	date	region	city/province	market	latitude	\
count	141765	141765	141765	141765	141765.000000	
unique	281	17	78	97	NaN	
top	15/07/2020	Region III	Davao del Sur	Davao City	NaN	
freq	3906	12110	5967	4316	NaN	
mean	NaN	NaN	NaN	NaN	11.671230	
std	NaN	NaN	NaN	NaN	3.616373	
min	NaN	NaN	NaN	NaN	5.029099	
25%	NaN	NaN	NaN	NaN	8.477437	
50%	NaN	NaN	NaN	NaN	11.560250	
75%	NaN	NaN	NaN	NaN	14.608900	
max	NaN	NaN	NaN	NaN	18.194082	

	longitude	category	commodity	unit	\
count	141765.000000	141765	141765	141765	
unique	NaN	4	67	2	
top	NaN	vegetables and fruits	Rice (regular, milled)	KG	
freq	NaN	60870	6065	136845	
mean	122.881505	NaN	NaN	NaN	
std	1.924088	NaN	NaN	NaN	
min	118.735278	NaN	NaN	NaN	
25%	121.086142	NaN	NaN	NaN	
50%	122.755280	NaN	NaN	NaN	
75%	124.716536	NaN	NaN	NaN	
max	126.207645	NaN	NaN	NaN	

```
       priceflag pricetype currency          price       usdprice
count     141765    141765   141765  141765.000000  141765.000000
unique         1         3        1            NaN            NaN
top       actual    Retail      PHP            NaN            NaN
freq      141765    135111   141765            NaN            NaN
mean         NaN       NaN      NaN     116.467966       2.287652
std          NaN       NaN      NaN     109.621618       2.117082
min          NaN       NaN      NaN       1.600000       0.034300
25%          NaN       NaN      NaN      40.710000       0.810200
50%          NaN       NaN      NaN      80.000000       1.586300
75%          NaN       NaN      NaN     158.920000       3.166500
max          NaN       NaN      NaN    1166.000000      21.062900
```

[5]:
```
# Check if there is null data
food_prices.isnull().sum()
```

[5]:
```
date             0
region           0
city/province    0
market           0
latitude         0
longitude        0
category         0
commodity        0
unit             0
priceflag        0
pricetype        0
currency         0
price            0
usdprice         0
dtype: int64
```

[6]:
```
# Check if there is duplication of data
food_prices.duplicated().sum()
```

[6]:
```
0
```

1.0.2 Convert column into correct data type

[56]:
```
# Convert date column into datetime
food_prices['date'] = pd.to_datetime(food_prices['date'], dayfirst=False)
```

1.0.3 Augement data with additional columns

[8]:
```
# Add Year and Month Column
food_prices['year'] = food_prices['date'].dt.year
food_prices['month'] = food_prices['date'].dt.month
```

1.0.4 Drop the unnecessary column

[9]:
```
food_prices = food_prices.
  ↪drop(food_prices[['latitude','longitude','currency','priceflag']], axis=1)
```

[10]:
```
# Rename the column price into price_php
food_prices.rename(columns={'price': 'price_php'}, inplace=True)
```

[11]:
```
food_prices.head()
```

[11]:
```
        date                   region        city/province        market  \
0 2000-01-15  National Capital region  Metropolitan Manila  Metro Manila
1 2000-01-15  National Capital region  Metropolitan Manila  Metro Manila
2 2000-01-15  National Capital region  Metropolitan Manila  Metro Manila
3 2000-01-15  National Capital region  Metropolitan Manila  Metro Manila
4 2000-01-15  National Capital region  Metropolitan Manila  Metro Manila

             category                commodity unit  pricetype  price_php  \
0  cereals and tubers     Maize flour (yellow)   KG     Retail      15.00
1  cereals and tubers  Rice (milled, superior)   KG     Retail      20.00
2  cereals and tubers  Rice (milled, superior)   KG  Wholesale      18.35
3  cereals and tubers   Rice (regular, milled)   KG     Retail      18.00
4  cereals and tubers   Rice (regular, milled)   KG  Wholesale      16.35

   usdprice  year  month
0    0.3717  2000      1
1    0.4957  2000      1
2    0.4548  2000      1
3    0.4461  2000      1
4    0.4052  2000      1
```

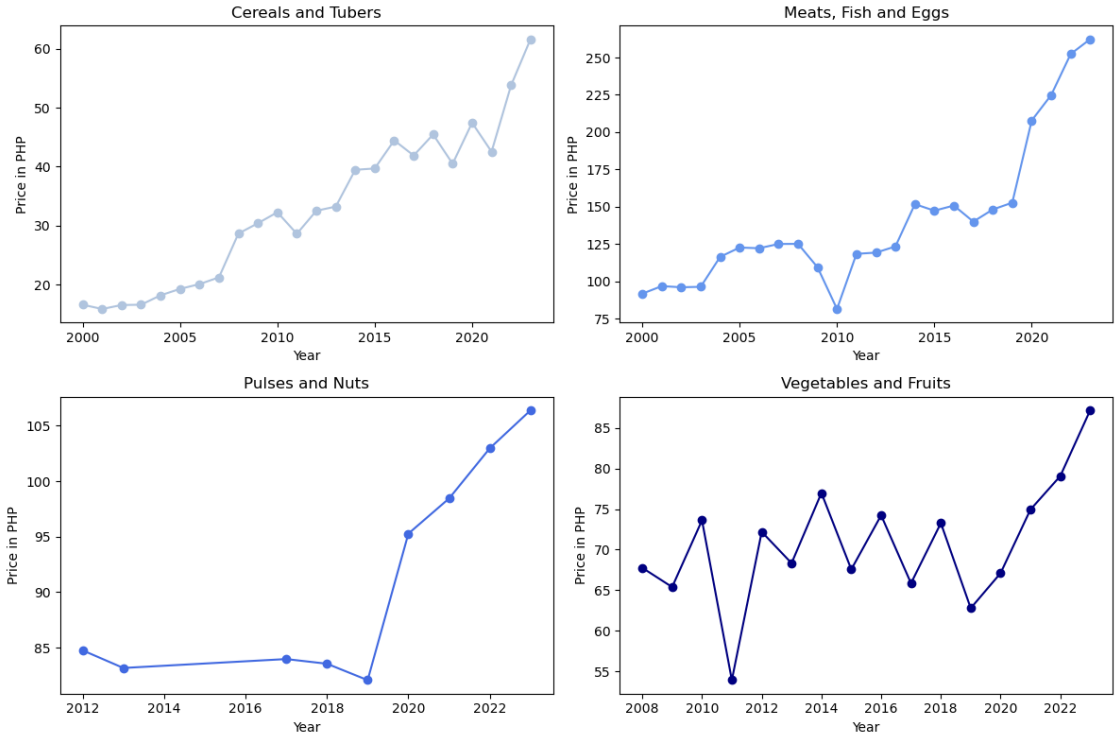
1.0.5 1. What are the prices of foods based on their categories over the years?

[12]:
```
# Extract the foods based on their category
food_cereals = food_prices[food_prices['category'] == 'cereals and tubers']
food_meats = food_prices[food_prices['category'] == 'meat, fish and eggs']
food_nuts = food_prices[food_prices['category'] == 'pulses and nuts']
food_veg = food_prices[food_prices['category'] == 'vegetables and fruits']
```

[13]:
```
# Group it according to their years and get the average prices of it
food_cereals_grp = food_cereals.groupby('year').mean(numeric_only = True)
food_meats_grp = food_meats.groupby('year').mean(numeric_only = True)
food_nuts_grp = food_nuts.groupby('year').mean(numeric_only = True)
food_veg_grp = food_veg.groupby('year').mean(numeric_only = True)
```

[14]:
```
# Create subplots
fig, axs = plt.subplots(2, 2, figsize=(12, 8))
colors = ['lightsteelblue','cornflowerblue','royalblue','navy']
```

```
# Plot each group on separate subplots
axs[0, 0].plot(food_cereals_grp.index, food_cereals_grp['price_php'],␣
 ↪marker='o', color=colors[0])
axs[0, 0].set_title('Cereals and Tubers')
axs[0, 0].set_xlabel('Year')
axs[0, 0].set_ylabel('Price in PHP')

axs[0, 1].plot(food_meats_grp.index, food_meats_grp['price_php'], marker='o',␣
 ↪color=colors[1])
axs[0, 1].set_title('Meats, Fish and Eggs')
axs[0, 1].set_xlabel('Year')
axs[0, 1].set_ylabel('Price in PHP')

axs[1, 0].plot(food_nuts_grp.index, food_nuts_grp['price_php'], marker='o',␣
 ↪color=colors[2])
axs[1, 0].set_title('Pulses and Nuts')
axs[1, 0].set_xlabel('Year')
axs[1, 0].set_ylabel('Price in PHP')

axs[1, 1].plot(food_veg_grp.index, food_veg_grp['price_php'], marker='o',␣
 ↪color=colors[3])
axs[1, 1].set_title('Vegetables and Fruits')
axs[1, 1].set_xlabel('Year')
axs[1, 1].set_ylabel('Price in PHP')

# Adjust layout
plt.tight_layout()

# Display the plot
plt.show()
```



1.0.6 2. What are the average prices of different foods based on cereal and tuber commodities?

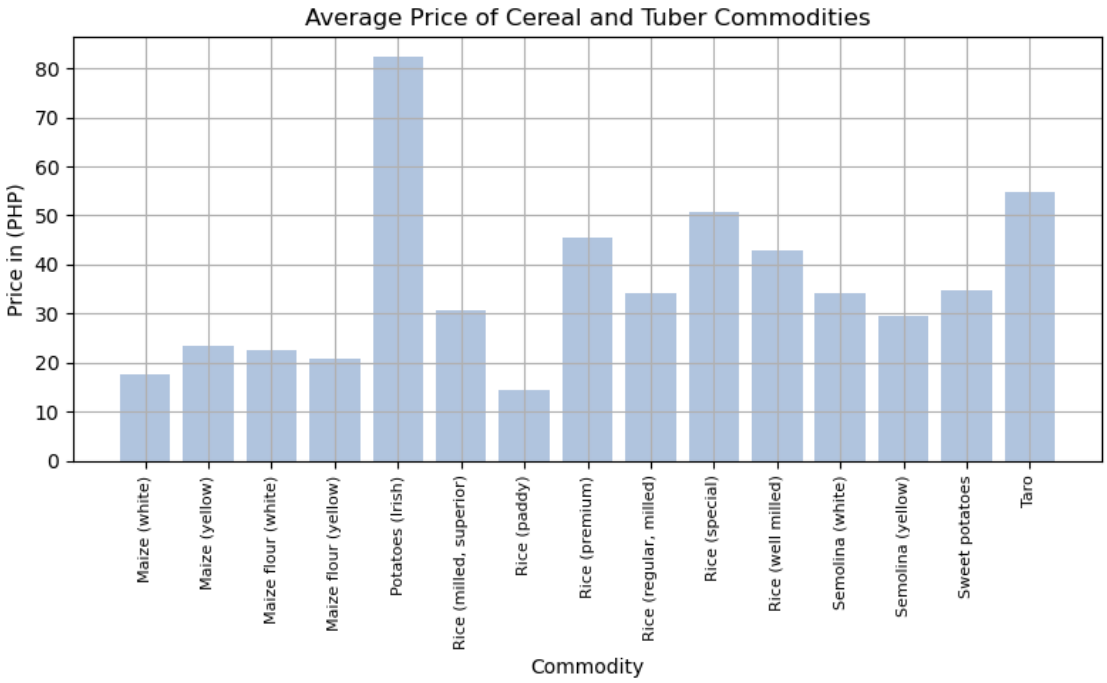
[15]:
```
# Check its value in tabular format
food_cereals_commodity = food_cereals.groupby('commodity').mean(numeric_only =␣
↪True)
food_cereals_commodity
```

[15]:

commodity	price_php	usdprice	year	month
Maize (white)	17.602381	0.365075	2012.584127	6.701587
Maize (yellow)	23.310014	0.455083	2017.526093	6.516220
Maize flour (white)	22.502729	0.476735	2009.375267	6.445629
Maize flour (yellow)	20.912271	0.441249	2009.353712	6.448326
Potatoes (Irish)	82.216344	1.621449	2017.794979	6.409780
Rice (milled, superior)	30.554382	0.644484	2009.447018	6.457544
Rice (paddy)	14.577440	0.328681	2009.855422	6.362952
Rice (premium)	45.453482	0.926445	2020.000000	8.473461
Rice (regular, milled)	34.111007	0.683148	2016.154658	6.483924
Rice (special)	50.802469	0.977202	2021.590666	6.295333
Rice (well milled)	42.761923	0.825025	2021.494423	6.429134
Semolina (white)	34.043766	0.665719	2021.062500	6.961538
Semolina (yellow)	29.489088	0.581289	2020.838235	7.302941

```
Sweet potatoes          34.854404  0.715550  2016.251445  6.748194
Taro                    54.781201  1.100345  2020.569024  7.627385
```

[16]:
```
# Values
cereals_group = food_cereals.groupby('commodity')
average_prices = cereals_group['price_php'].mean()

# Plotting
plt.figure(figsize=(8, 5))
plt.bar(average_prices.index, average_prices.values, color= colors[0])
plt.xticks(rotation='vertical', size=8)
plt.title('Average Price of Cereal and Tuber Commodities')
plt.xlabel('Commodity')
plt.ylabel('Price in (PHP)')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



1.0.7 3. What are the average prices of different foods based on meat, fish and eggs commodities?

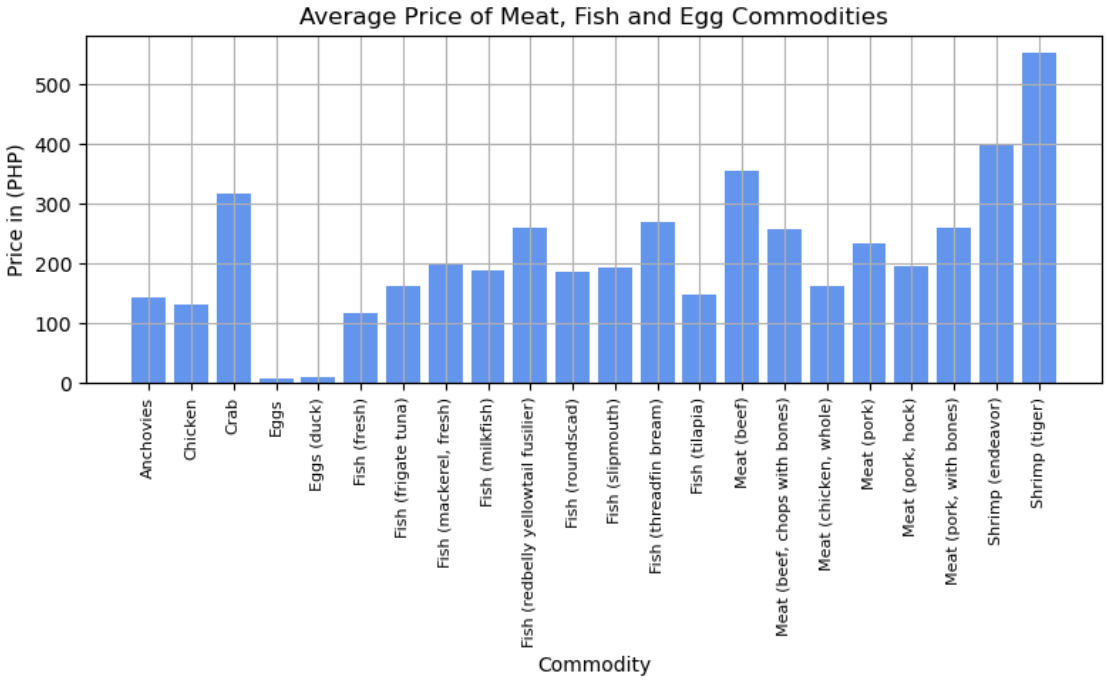
[17]:
```
# Check its value in tabular format
food_meats_commodity = food_meats.groupby('commodity').mean(numeric_only = True)
food_meats_commodity
```

[17]:

commodity	price_php	usdprice	year	month
Anchovies	143.070315	2.760217	2021.483254	6.418262
Chicken	130.802500	2.669101	2020.000000	8.588235
Crab	318.336297	6.172280	2021.381669	6.518924
Eggs	6.588257	0.131344	2018.529018	6.434440
Eggs (duck)	8.895130	0.174999	2021.211460	6.654843
Fish (fresh)	117.450311	2.485786	2014.060656	6.316393
Fish (frigate tuna)	161.713752	3.219353	2020.517241	7.772414
Fish (mackerel, fresh)	199.179325	3.916124	2020.708589	7.616564
Fish (milkfish)	189.262851	3.645045	2021.473290	6.450493
Fish (redbelly yellowtail fusilier)	261.012299	5.091756	2021.156561	6.857919
Fish (roundscad)	185.614170	3.581907	2021.444405	6.470795
Fish (slipmouth)	192.945194	3.785784	2021.144876	6.825972
Fish (threadfin bream)	268.889487	5.282732	2020.910769	7.236923
Fish (tilapia)	148.326045	2.859232	2021.476083	6.449341
Meat (beef)	354.339791	6.853757	2021.386139	
Meat (beef, chops with bones)	258.495050	5.122917	2018.820733	
Meat (chicken, whole)	163.257027	3.271382	2018.354890	
Meat (pork)	233.560284	4.636326	2016.856418	
Meat (pork, hock)	195.083050	3.916536	2020.322302	
Meat (pork, with bones)	260.820101	5.028031	2021.436502	
Shrimp (endeavor)	398.397759	7.787631	2020.930425	
Shrimp (tiger)	553.253425	10.684839	2021.505576	

```
Meat (beef)                          6.550647
Meat (beef, chops with bones)        6.386827
Meat (chicken, whole)                6.447243
Meat (pork)                          6.420598
Meat (pork, hock)                    8.043165
Meat (pork, with bones)              6.487162
Shrimp (endeavor)                    7.344340
Shrimp (tiger)                       6.380112
```

[18]:
```
# Values
meats_group = food_meats.groupby('commodity')
average_prices = meats_group['price_php'].mean()

# Plotting
plt.figure(figsize=(8, 5))
plt.bar(average_prices.index, average_prices.values, color= colors[1])
plt.xticks(rotation='vertical', size=8)
plt.title('Average Price of Meat, Fish and Egg Commodities')
plt.xlabel('Commodity')
plt.ylabel('Price in (PHP)')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



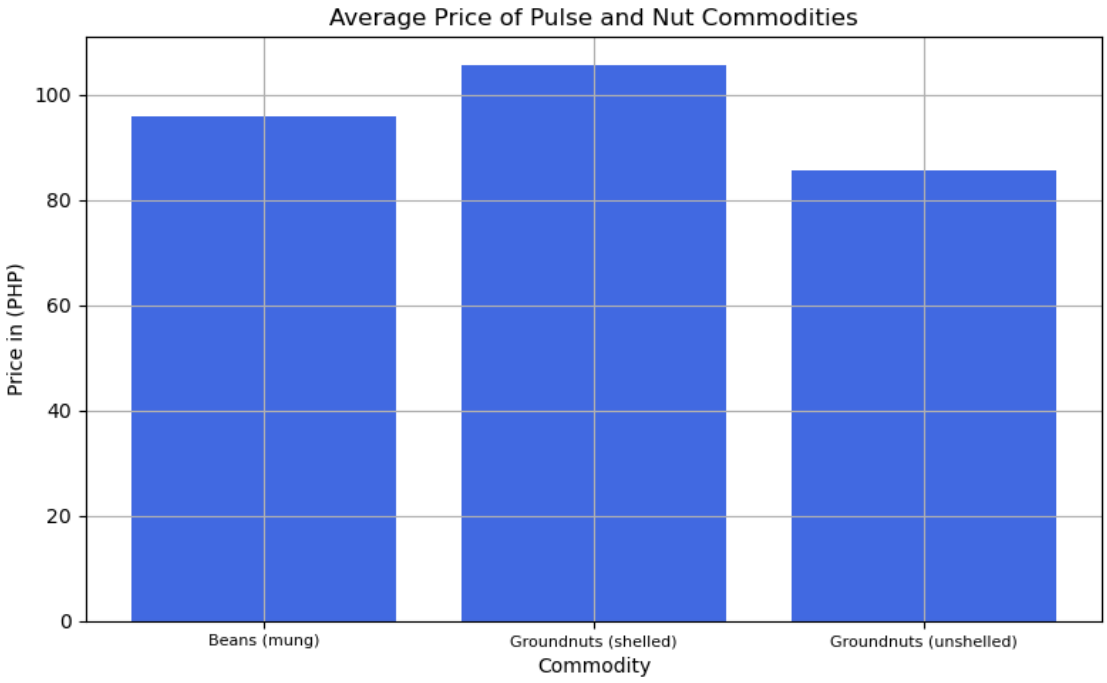
1.0.8 4. What are the average prices of different foods based on pulse and nut commodities?

[19]:
```
# Check its value in tabular format
food_nuts_commodity = food_nuts.groupby('commodity').mean(numeric_only = True)
food_nuts_commodity
```

[19]:

commodity	price_php	usdprice	year	month
Beans (mung)	95.892711	1.882528	2020.310321	6.573266
Groundnuts (shelled)	105.656252	2.055130	2021.303342	6.655013
Groundnuts (unshelled)	85.606886	1.756495	2019.329829	6.725361

[20]:
```
# Values
nuts_group = food_nuts.groupby('commodity')
average_prices = nuts_group['price_php'].mean()

# Plotting
plt.figure(figsize=(8, 5))
plt.bar(average_prices.index, average_prices.values, color= colors[2])
plt.xticks(size=8)
plt.title('Average Price of Pulse and Nut Commodities')
plt.xlabel('Commodity')
plt.ylabel('Price in (PHP)')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



1.0.9 5. What are the average prices of different foods based on vegetable and fruit commodities?

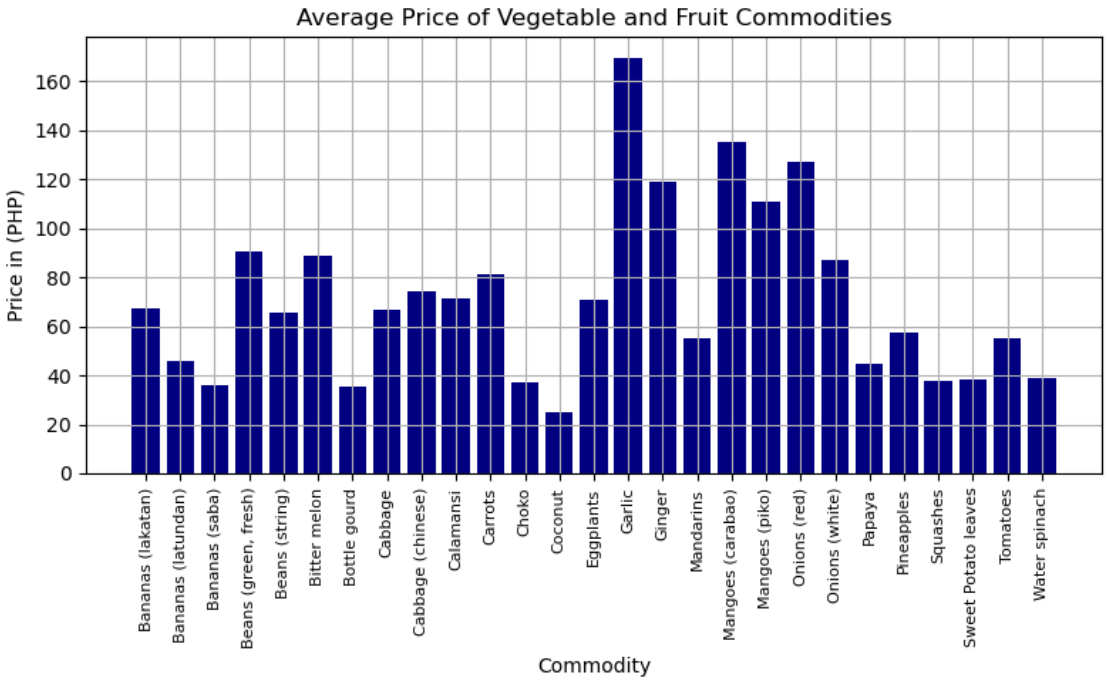
[21]:
```
# Check its value in tabular format
food_veg_commodity = food_veg.groupby('commodity').mean(numeric_only = True)
food_veg_commodity
```

[21]:

commodity	price_php	usdprice	year	month
Bananas (lakatan)	67.554881	1.307257	2021.450020	6.512343
Bananas (latundan)	45.934332	0.891649	2021.352273	6.648674
Bananas (saba)	36.200947	0.710288	2021.015909	7.120455
Beans (green, fresh)	90.223809	1.759541	2021.164058	6.860290
Beans (string)	65.528576	1.273776	2021.233986	6.773425
Bitter melon	88.812625	1.713834	2021.453802	6.467780
Bottle gourd	35.528016	0.701354	2021.061053	6.882632
Cabbage	66.813808	1.328457	2018.378677	6.432714
Cabbage (chinese)	74.526775	1.453204	2021.124661	6.963415
Calamansi	71.227346	1.371292	2021.330853	6.687004
Carrots	81.285737	1.614221	2018.375282	6.400902
Choko	37.254118	0.724328	2021.287304	6.650041
Coconut	24.782074	0.479415	2021.432712	6.492409
Eggplants	71.049634	1.366791	2021.480732	6.445787
Garlic	169.452478	3.456442	2018.054362	6.266330
Ginger	118.710807	2.325668	2021.354541	6.627478

```
Mandarins              55.246019  1.088677  2021.245370  6.731481
Mangoes (carabao)     135.348905  2.624672  2021.352941  6.582449
Mangoes (piko)        110.994143  2.174957  2020.956175  6.848606
Onions (red)          126.908194  2.506834  2018.080074  6.465486
Onions (white)         86.750117  1.795826  2016.180712  6.802346
Papaya                 44.405408  0.897056  2020.304000  8.132800
Pineapples             57.184409  1.128158  2020.845304  7.217680
Squashes               37.580402  0.726678  2021.380880  6.682218
Sweet Potato leaves    38.031376  0.756837  2020.603468  7.678613
Tomatoes               55.348039  1.099578  2018.325331  6.412614
Water spinach          38.885556  0.774976  2020.581633  7.692744
```

[22]:
```python
# Values
veg_group = food_veg.groupby('commodity')
average_prices = veg_group['price_php'].mean()

# Plotting
plt.figure(figsize=(8, 5))
plt.bar(average_prices.index, average_prices.values, color= colors[3])
plt.xticks(rotation='vertical', size=8)
plt.title('Average Price of Vegetable and Fruit Commodities')
plt.xlabel('Commodity')
plt.ylabel('Price in (PHP)')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



1.0.10 6. What are the prices of Rice (milled,superior) and Rice(milled, regular) over the available years?

[23]:
```
# Extract the Rice (milled, superior) from the dataset
rice_superior = food_prices[food_prices['commodity'] == 'Rice (milled,␣
 ↪superior)']
rice_superior = rice_superior.groupby('year').mean(numeric_only = True)
rice_superior
```

[23]:

year	price_php	usdprice	month
2000	18.746528	0.425686	6.500000
2001	18.475000	0.362428	6.500000
2002	19.415070	0.376337	6.549296
2003	19.443403	0.359081	6.500000
2004	20.315211	0.363980	6.563380
2005	21.979306	0.400479	6.500000
2006	22.475625	0.439017	6.500000
2007	23.843542	0.518858	6.500000
2008	31.388056	0.708152	6.500000
2009	32.684028	0.688494	6.500000
2010	32.586111	0.724411	6.500000
2011	33.084722	0.763632	6.500000
2012	33.992431	0.805844	6.500000

```
2013  35.766806  0.844035  6.500000
2014  41.254514  0.934321  6.500000
2015  40.826087  0.899724  6.391304
2016  40.620709  0.859824  6.517730
2017  41.307483  0.820481  6.482517
2018  44.389008  0.842687  6.442748
2019  40.793697  0.787555  6.470588
2020  38.031818  0.749909  1.454545
```

[24]:
```
# Extract the Rice (regular, milled) from the dataset
rice_regular = food_prices[food_prices['commodity'] == 'Rice (regular, milled)']
rice_regular = rice_regular.groupby('year').mean(numeric_only = True)
rice_regular
```

[24]:
```
      price_php  usdprice     month
year
2000  17.152869  0.388108  6.680328
2001  16.551230  0.324330  6.696721
2002  17.312353  0.335582  6.655462
2003  17.283984  0.319020  6.650407
2004  18.438049  0.330301  6.634146
2005  20.309024  0.369985  6.674797
2006  20.657280  0.403727  6.640000
2007  22.212742  0.484293  6.661290
2008  28.654956  0.646273  6.654867
2009  29.604602  0.623711  6.654867
2010  30.369454  0.675630  6.584699
2011  30.665398  0.707819  6.654867
2012  31.907514  0.756701  6.594595
2013  33.031226  0.782694  6.012903
2014  38.674899  0.876980  7.342282
2015  36.864205  0.811941  6.505682
2016  36.373757  0.769623  6.574586
2017  36.950323  0.733184  6.875576
2018  40.755750  0.772930  6.600000
2019  36.455789  0.701989  5.368421
2020  37.362736  0.760898  8.300166
2021  37.577252  0.764284  6.510571
2022  39.088476  0.719223  6.500000
2023  40.309948  0.728883  4.000000
```

[25]:
```
# Extract the Rice (special) from the dataset
rice_paddy = food_prices[food_prices['commodity'] == 'Rice (paddy)']
rice_paddy = rice_paddy.groupby('year').mean(numeric_only = True)
rice_paddy
```

[25]:

year	price_php	usdprice	month
2008	13.714146	0.310783	6.548780
2009	14.262883	0.300466	6.582822
2010	14.516981	0.322806	6.559748
2012	15.814101	0.375481	6.870504
2013	15.361282	0.379018	2.051282

[26]:
```
# Extract the Rice (special) from the dataset
rice_special = food_prices[food_prices['commodity'] == 'Rice (special)']
rice_special = rice_special.groupby('year').mean(numeric_only = True)
rice_special
```

[26]:

year	price_php	usdprice	month
2020	50.889495	1.037444	8.515152
2021	49.886670	1.014688	6.510989
2022	50.918259	0.937073	6.500000
2023	52.048662	0.941171	4.000000

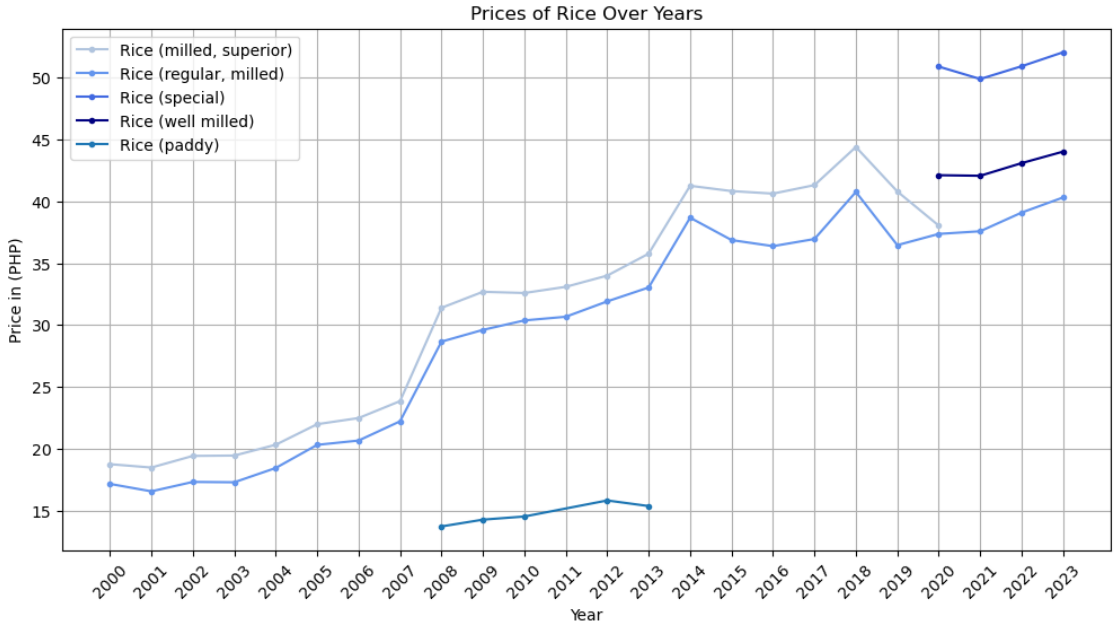
[27]:
```
# Extract the Rice (well milled) from the dataset
rice_wmilled = food_prices[food_prices['commodity'] == 'Rice (well milled)']
rice_wmilled = rice_wmilled.groupby('year').mean(numeric_only = True)
rice_wmilled
```

[27]:

year	price_php	usdprice	month
2020	42.109800	0.858351	8.489149
2021	42.063011	0.855511	6.510989
2022	43.086183	0.792922	6.500000
2023	44.016684	0.795924	4.000000

[28]:
```
# X value from 2000 - 2023
years = range(2000,2024)

# Create and Display the plot lin line chart
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(rice_superior.index, rice_superior['price_php'], label='Rice (milled,␣
 ↪superior)',marker='.',color = colors[0])
ax.plot(rice_regular.index, rice_regular['price_php'], label='Rice (regular,␣
 ↪milled)',marker='.',color = colors[1])
ax.plot(rice_special.index, rice_special['price_php'], label='Rice (special)',␣
 ↪marker='.',color = colors[2])
ax.plot(rice_wmilled.index, rice_wmilled['price_php'], label='Rice (well␣
 ↪milled)',marker='.',color = colors[3])
ax.plot(rice_paddy.index, rice_paddy['price_php'], label='Rice␣
 ↪(paddy)',marker='.')
```

```
# Set labels and title
ax.set_xlabel('Year')
ax.set_ylabel('Price in (PHP)')
ax.set_title('Prices of Rice Over Years')
plt.xticks(years, rotation=45)

# Add a legend
ax.legend()

# Display the plot
plt.grid(True)
plt.show()
```



1.0.11 7. What are the prices of Meat(beef), Meat(pork), Meat(chicken, whole) over the years?

[29]:
```
# Extract the Meat (beef, chops with bones)from the dataset
beef = food_prices[food_prices['commodity'] == 'Meat (beef, chops with bones)']
beef = beef.groupby('year').mean(numeric_only = True)
beef
```

[29]:

year	price_php	usdprice	month
2008	182.587500	4.131308	6.500000
2009	166.799531	3.514164	6.500000

```
2010  202.575833  4.505183  6.500000
2011  179.705260  4.147907  6.500000
2012  181.058087  4.280183  5.934426
2013  181.822805  4.418485  3.329268
2014  186.031111  4.224127  9.765432
2015  186.914857  4.128223  6.068571
2016  192.299737  4.070317  6.521053
2017  203.858895  4.047926  6.531579
2018  225.726919  4.286261  6.474747
2019  233.946988  4.483959  3.024096
2020  244.354164  4.984138  8.520067
2021  284.136624  5.776902  6.511601
2022  319.971470  5.883811  6.500000
2023  333.674504  6.033685  4.000000
```

[30]:
```
# Extract the Meat (pork) from the dataset
pork = food_prices[food_prices['commodity'] == 'Meat (pork)']
pork = beef.groupby('year').mean(numeric_only = True)
pork
```

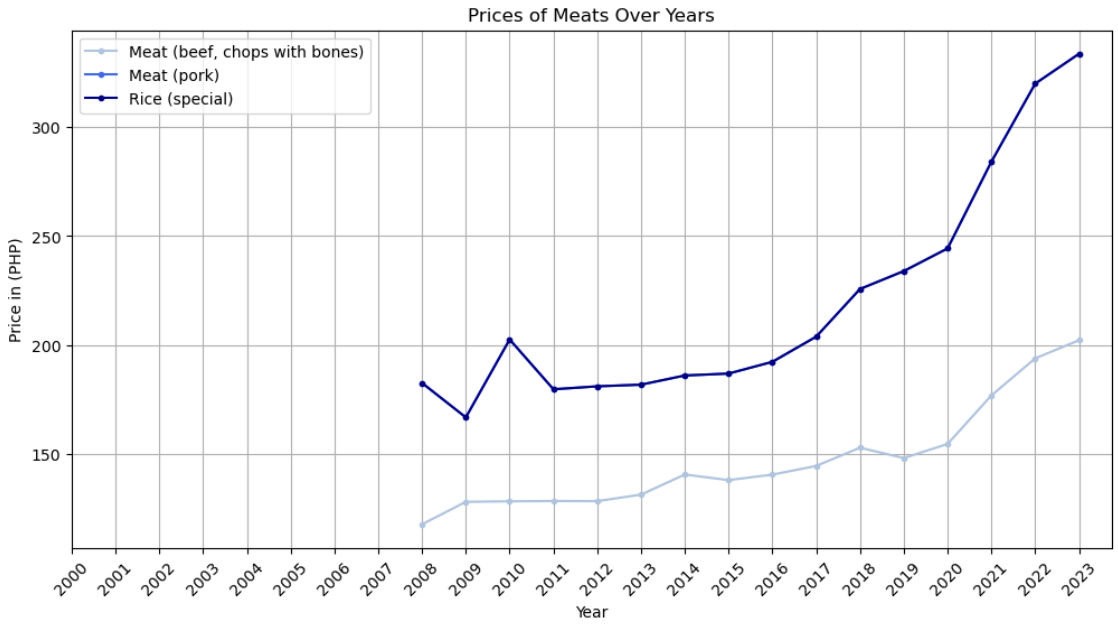
[30]:
```
       price_php  usdprice     month
year
2008  182.587500  4.131308  6.500000
2009  166.799531  3.514164  6.500000
2010  202.575833  4.505183  6.500000
2011  179.705260  4.147907  6.500000
2012  181.058087  4.280183  5.934426
2013  181.822805  4.418485  3.329268
2014  186.031111  4.224127  9.765432
2015  186.914857  4.128223  6.068571
2016  192.299737  4.070317  6.521053
2017  203.858895  4.047926  6.531579
2018  225.726919  4.286261  6.474747
2019  233.946988  4.483959  3.024096
2020  244.354164  4.984138  8.520067
2021  284.136624  5.776902  6.511601
2022  319.971470  5.883811  6.500000
2023  333.674504  6.033685  4.000000
```

[31]:
```
# Extract the Meat (chicken, whole) from the dataset
chicken = food_prices[food_prices['commodity'] == 'Meat (chicken, whole)']
chicken = chicken.groupby('year').mean(numeric_only = True)
chicken
```

[31]:
```
       price_php  usdprice     month
year
2008  117.743333  2.664225  6.500000
```

```
2009  128.041615  2.698476  6.500000
2010  128.275276  2.851928  6.607362
2011  128.417917  2.964054  6.500000
2012  128.355568  3.034390  5.945946
2013  131.351235  3.186914  3.456790
2014  140.588765  3.192156  9.765432
2015  138.006492  3.039098  6.481675
2016  140.535579  2.974685  6.521053
2017  144.518550  2.870191  6.580000
2018  152.879698  2.903512  6.472362
2019  148.120357  2.838644  3.023810
2020  154.602669  3.153125  8.518946
2021  176.874398  3.600055  6.512285
2022  193.885586  3.563077  6.500000
2023  202.199085  3.656467  4.000000
```

[32]:
```
# X value from 2000 - 2023
years = range(2000,2024)

# Create and Display the plot lin line chart
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(chicken.index, chicken['price_php'], label= 'Meat (beef, chops with␣
 ↪bones)',marker='.',color = colors[0])
ax.plot(beef.index, beef['price_php'], label= 'Meat (pork)',marker='.',color =␣
 ↪colors[2])
ax.plot(pork.index, pork['price_php'], label='Rice (special)', marker='.',color␣
 ↪= colors[3])

# Set labels and title
ax.set_xlabel('Year')
ax.set_ylabel('Price in (PHP)')
ax.set_title('Prices of Meats Over Years')
plt.xticks(years, rotation=45)

# Add a legend
ax.legend()

# Display the plot
plt.grid(True)
plt.show()
```



1.0.12 8. What are the prices of Beans (mung), Groundnuts (shelled), and Groundnuts (unshelled) over the years?

[33]:
```
# Extract the Beans (mung) from the dataset
mung = food_prices[food_prices['commodity'] == 'Beans (mung)']
mung = mung.groupby('year').mean(numeric_only = True)
mung
```

[33]:

year	price_php	usdprice	month
2012	76.064153	1.818100	8.457627
2013	72.879667	1.790422	2.500000
2017	83.968346	1.660407	8.526316
2018	83.548090	1.587099	6.472362
2019	82.069643	1.572851	3.023810
2020	94.301675	1.922791	8.487562
2021	99.496284	2.023396	6.500000
2022	102.471574	1.886036	6.500000
2023	103.783413	1.876724	4.000000

[34]:
```
# Extract the Groundnuts (shelled) from the dataset
shelled = food_prices[food_prices['commodity'] == 'Groundnuts (shelled)']
shelled = shelled.groupby('year').mean(numeric_only = True)
shelled
```

[34]:

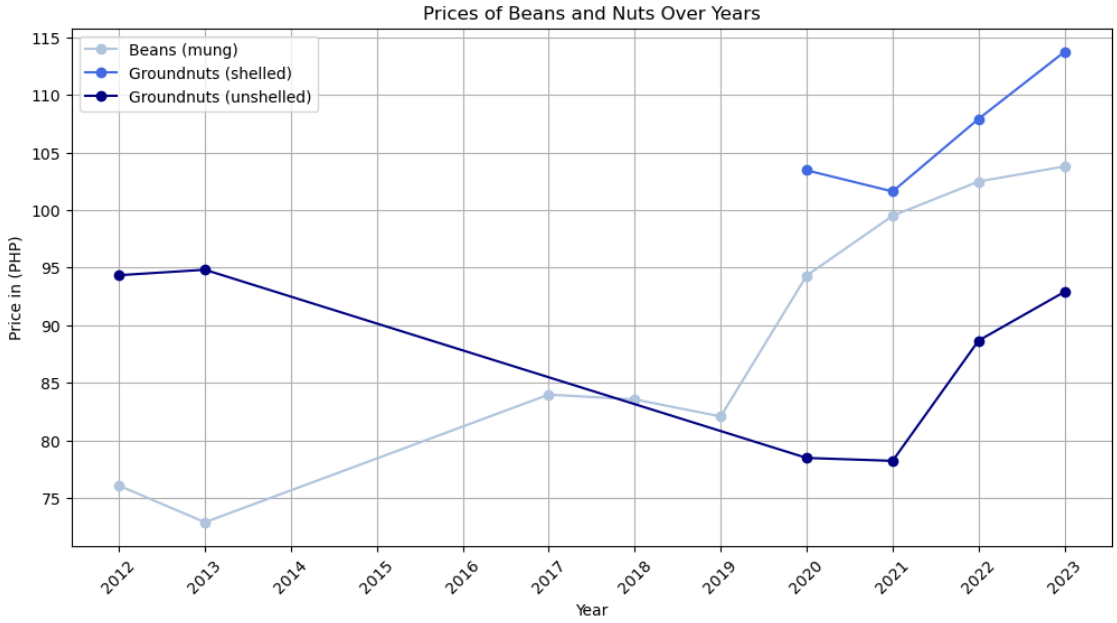
year	price_php	usdprice	month
2020	103.445852	2.109492	8.496212
2021	101.608100	2.066870	6.500000
2022	107.912287	1.984959	6.500000
2023	113.735548	2.056660	4.000000

[35]:
```
# Extract the Groundnuts (unshelled) from the dataset
unshelled = food_prices[food_prices['commodity'] == 'Groundnuts (unshelled)']
unshelled = unshelled.groupby('year').mean(numeric_only = True)
unshelled
```

[35]:

year	price_php	usdprice	month
2012	94.336636	2.254625	8.364486
2013	94.810377	2.328892	2.528302
2020	78.466622	1.597845	8.346847
2021	78.213636	1.590662	6.500000
2022	88.659423	1.631251	6.500000
2023	92.892418	1.679697	4.000000

[36]:
```
# X value from 2012 - 2023
years = range(2012,2024)

# Create and Display the plot lin line chart
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(mung.index, mung['price_php'], label= 'Beans (mung)',marker='o',color =␣
 ↪colors[0])
ax.plot(shelled.index, shelled['price_php'], label= 'Groundnuts␣
 ↪(shelled)',marker='o',color = colors[2])
ax.plot(unshelled.index, unshelled['price_php'], label='Groundnuts␣
 ↪(unshelled)', marker='o',color = colors[3])

# Set labels and title
ax.set_xlabel('Year')
ax.set_ylabel('Price in (PHP)')
ax.set_title('Prices of Beans and Nuts Over Years')
plt.xticks(years, rotation=45)

# Add a legend
ax.legend()

# Display the plot
plt.grid(True)
plt.show()
```



1.0.13 9. What are the prices of Onions (red) and Onions (white) over the years?

[37]:
```
# Extract the Onions (red) from the dataset
red = food_prices[food_prices['commodity'] == 'Onions (red)']
red = red.groupby('year').mean(numeric_only = True)
red
```

[37]:

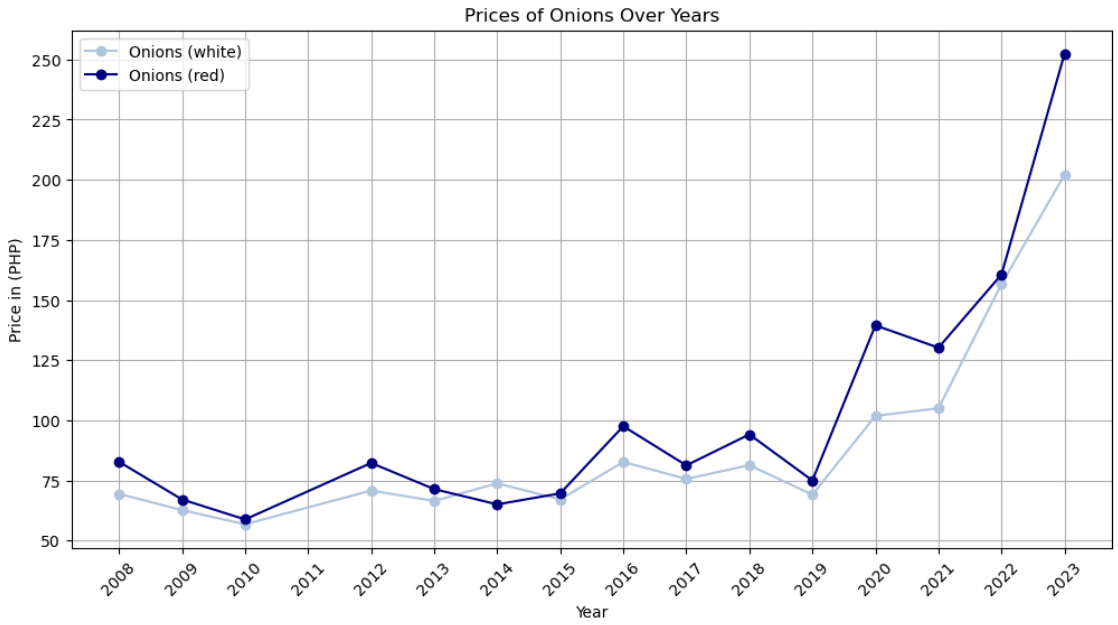
year	price_php	usdprice	month
2008	82.767839	1.871806	6.608040
2009	67.103529	1.415738	6.500000
2010	58.789902	1.306737	6.500000
2012	82.263114	1.948488	6.161677
2013	71.406190	1.735879	3.357143
2014	65.050000	1.476048	9.835443
2015	69.633403	1.526684	6.481675
2016	97.526684	2.064194	6.521053
2017	81.257000	1.612912	6.580000
2018	94.127136	1.788502	6.472362
2019	74.984643	1.436796	3.023810
2020	139.471250	2.852604	8.513158
2021	130.180913	2.646285	6.512853
2022	160.615765	2.941762	6.500000
2023	252.350047	4.565492	4.000000

[38]:
```
# Extract the Onions (white) from the dataset
white = food_prices[food_prices['commodity'] == 'Onions (white)']
white = white.groupby('year').mean(numeric_only = True)
white
```

[38]:

year	price_php	usdprice	month
2008	69.424534	1.569771	6.503106
2009	62.652532	1.321978	6.664557
2010	56.782788	1.265775	6.581818
2012	70.799448	1.675895	6.006897
2013	66.500270	1.615334	3.351351
2014	73.814833	1.676572	9.566667
2015	67.112632	1.473055	6.548872
2016	82.689462	1.751570	6.530769
2017	75.627986	1.499781	6.597122
2018	81.330500	1.543430	6.471429
2019	69.072973	1.323678	3.027027
2020	101.835925	2.079624	8.508079
2021	105.012183	2.136987	6.500000
2022	156.622639	2.831719	6.500000
2023	202.125000	3.655286	4.000000

[39]:
```
# X value from 2008 - 2023
years = range(2008,2024)

# Create and Display the plot lin line chart
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(white.index, white['price_php'], label= 'Onions␣
 ↪(white)',marker='o',color = colors[0])
ax.plot(red.index, red['price_php'], label= 'Onions (red)',marker='o',color =␣
 ↪colors[3])

# Set labels and title
ax.set_xlabel('Year')
ax.set_ylabel('Price in (PHP)')
ax.set_title('Prices of Onions Over Years')
plt.xticks(years, rotation=45)

# Add a legend
ax.legend()

# Display the plot
plt.grid(True)
plt.show()
```



1.0.14 10. What are the average price of cereals and tubers per region?

[40]:
```
# Extract the foods based on their category
food_cereals = food_prices[food_prices['category'] == 'cereals and tubers']
food_meats = food_prices[food_prices['category'] == 'meat, fish and eggs']
food_nuts = food_prices[food_prices['category'] == 'pulses and nuts']
food_veg = food_prices[food_prices['category'] == 'vegetables and fruits']
```

[41]:
```
# Group by Region
reg_cereals = food_cereals.groupby('region').mean(numeric_only = True)
reg_cereals
```

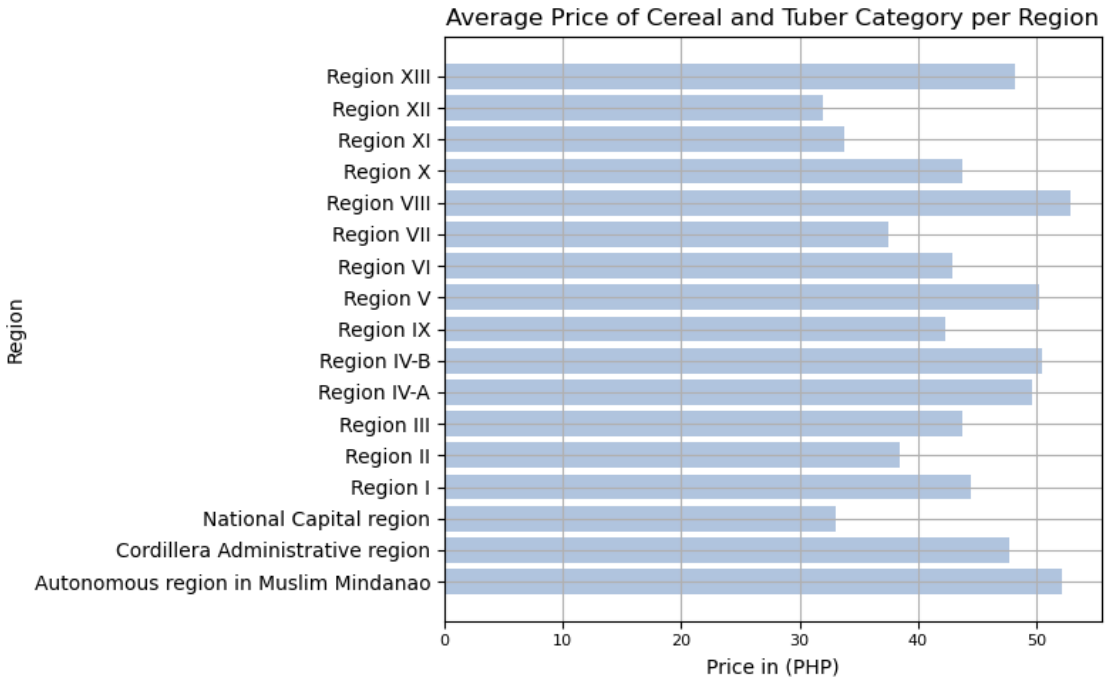
[41]:

region	price_php	usdprice	year
Autonomous region in Muslim Mindanao	52.142586	1.030300	2018.975732
Cordillera Administrative region	47.764341	0.940433	2019.896739
National Capital region	33.078471	0.684074	2011.175381
Region I	44.431037	0.880022	2018.863372
Region II	38.414020	0.758985	2019.290958
Region III	43.782958	0.872051	2017.108964
Region IV-A	49.672782	0.983454	2019.652439
Region IV-B	50.533045	0.994282	2019.539267
Region IX	42.296324	0.835698	2017.796157
Region V	50.201824	0.982982	2019.782866
Region VI	42.941176	0.858614	2015.969369
Region VII	37.547868	0.761098	2013.594051

```
Region VIII                              52.895866  1.039900  2019.476761
Region X                                 43.746881  0.864336  2019.142753
Region XI                                33.741377  0.684489  2013.314845
Region XII                               31.953721  0.649065  2013.350021
Region XIII                              48.225120  0.953937  2019.229215
```

```
                                            month
region
Autonomous region in Muslim Mindanao  6.499582
Cordillera Administrative region      6.519701
National Capital region               6.426208
Region I                              6.650194
Region II                             6.544315
Region III                            6.597424
Region IV-A                           6.637195
Region IV-B                           6.666667
Region IX                             6.521303
Region V                              6.596750
Region VI                             6.762613
Region VII                            6.514872
Region VIII                           6.582394
Region X                              6.630307
Region XI                             6.523980
Region XII                            6.558198
Region XIII                           6.532246
```

[42]:
```
# Plotting
plt.figure(figsize=(8, 5))
plt.barh(reg_cereals.index, reg_cereals['price_php'], color= colors[0])
plt.xticks(size=8)
plt.title('Average Price of Cereal and Tuber Category per Region')
plt.xlabel('Price in (PHP)')
plt.ylabel('Region')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



[43]:
```
food_prices['category'].unique()
```

[43]:
```
array(['cereals and tubers', 'meat, fish and eggs',
       'vegetables and fruits', 'pulses and nuts'], dtype=object)
```

1.0.15 11. What are the average price of meat, fish and eggs per region?

[44]:
```
# Group by Region
reg_meats = food_meats.groupby('region').mean(numeric_only = True)
reg_meats
```

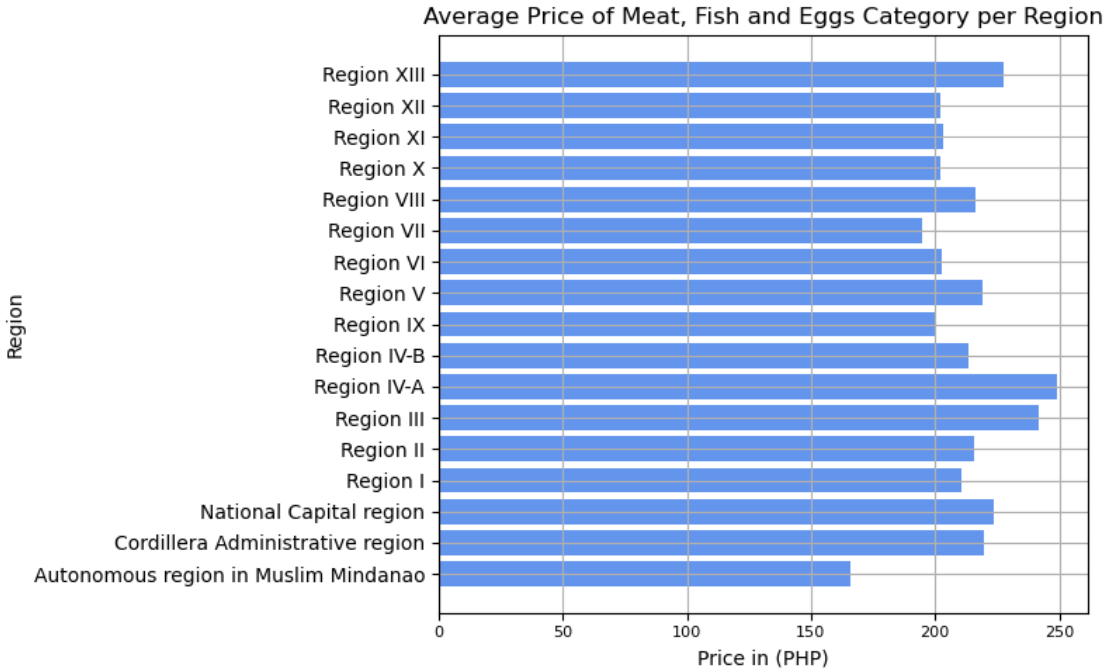
[44]:

region	price_php	usdprice	year
Autonomous region in Muslim Mindanao	165.968492	3.292692	2019.686805
Cordillera Administrative region	219.801779	4.314243	2020.086177
National Capital region	223.438706	4.410858	2016.643617
Region I	210.335606	4.130096	2019.770362
Region II	215.990342	4.240193	2019.827483
Region III	241.496093	4.700136	2020.118820
Region IV-A	249.102604	4.874288	2020.111526
Region IV-B	213.209391	4.168453	2020.253727
Region IX	199.896201	3.892709	2019.857804
Region V	219.124264	4.296793	2020.164779
Region VI	202.529022	3.946357	2019.864920
Region VII	194.895731	3.858347	2018.612301

```
Region VIII                              216.491233  4.228475  2020.335129
Region X                                 201.993398  3.962428  2019.931435
Region XI                                202.983528  3.959116  2019.245335
Region XII                               202.269130  3.967594  2018.999580
Region XIII                              227.578172  4.439933  2020.181296
```

```
                                           month
region
Autonomous region in Muslim Mindanao  6.632189
Cordillera Administrative region      6.563943
National Capital region               6.354610
Region I                              6.567739
Region II                             6.476884
Region III                            6.516479
Region IV-A                           6.600000
Region IV-B                           6.534161
Region IX                             6.517459
Region V                              6.632300
Region VI                             6.525681
Region VII                            6.609112
Region VIII                           6.586901
Region X                              6.625703
Region XI                             6.568417
Region XII                            6.521227
Region XIII                           6.534304
```

[45]:
```
# Plotting
plt.figure(figsize=(8, 5))
plt.barh(reg_meats.index, reg_meats['price_php'], color= colors[1])
plt.xticks(size=8)
plt.title('Average Price of Meat, Fish and Eggs Category per Region')
plt.xlabel('Price in (PHP)')
plt.ylabel('Region')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



1.0.16 12. What are the average price of pulses and nuts per region?

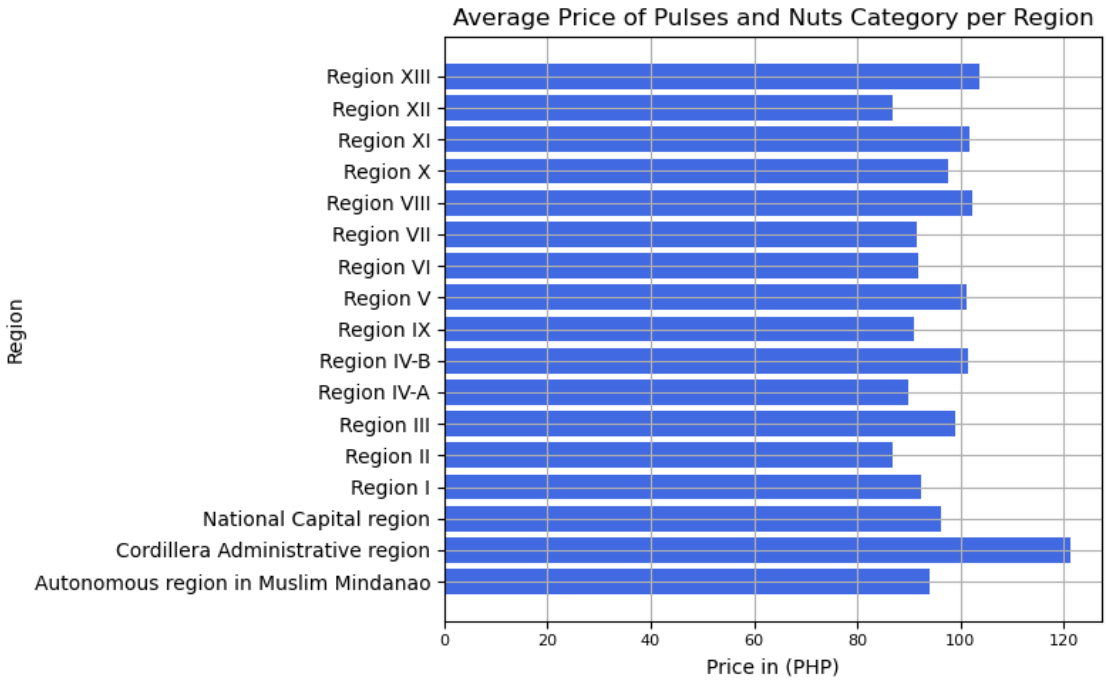
[46]:
```
# Group by Region
reg_nuts = food_nuts.groupby('region').mean(numeric_only = True)
reg_nuts
```

[46]:

region	price_php	usdprice	year
Autonomous region in Muslim Mindanao	94.111208	1.824522	2020.771812
Cordillera Administrative region	121.336272	2.410127	2020.346154
National Capital region	96.374444	1.933576	2019.088889
Region I	92.277158	1.830992	2020.246575
Region II	86.974496	1.712061	2020.596730
Region III	99.008195	1.938123	2020.943205
Region IV-A	89.877202	1.798361	2019.697248
Region IV-B	101.619321	1.974994	2021.000000
Region IX	90.886507	1.783762	2020.514706
Region V	101.297960	1.972852	2020.956284
Region VI	91.945664	1.859257	2019.517699
Region VII	91.557914	1.802797	2020.582781
Region VIII	102.226060	1.998560	2020.812000
Region X	97.520246	1.926580	2020.252632
Region XI	101.836838	1.989803	2020.654412
Region XII	86.803645	1.706634	2020.404682
Region XIII	103.609643	2.064267	2020.011905

```
                                          month
region
Autonomous region in Muslim Mindanao  6.557047
Cordillera Administrative region      6.713018
National Capital region               6.585185
Region I                              6.633562
Region II                             6.482289
Region III                            6.651116
Region IV-A                           6.894495
Region IV-B                           6.501171
Region IX                             6.761029
Region V                              6.398907
Region VI                             7.247788
Region VII                            6.665563
Region VIII                           6.536000
Region X                              6.666667
Region XI                             6.563725
Region XII                            6.548495
Region XIII                           6.718254
```

[47]:
```
# Plotting
plt.figure(figsize=(8, 5))
plt.barh(reg_nuts.index, reg_nuts['price_php'], color= colors[2])
plt.xticks(size=8)
plt.title('Average Price of Pulses and Nuts Category per Region')
plt.xlabel('Price in (PHP)')
plt.ylabel('Region')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



1.0.17 13. What are the average price of vegetables and fruits per region?

[48]:
```
# Group by Region
reg_veg = food_veg.groupby('region').mean(numeric_only = True)
reg_veg
```

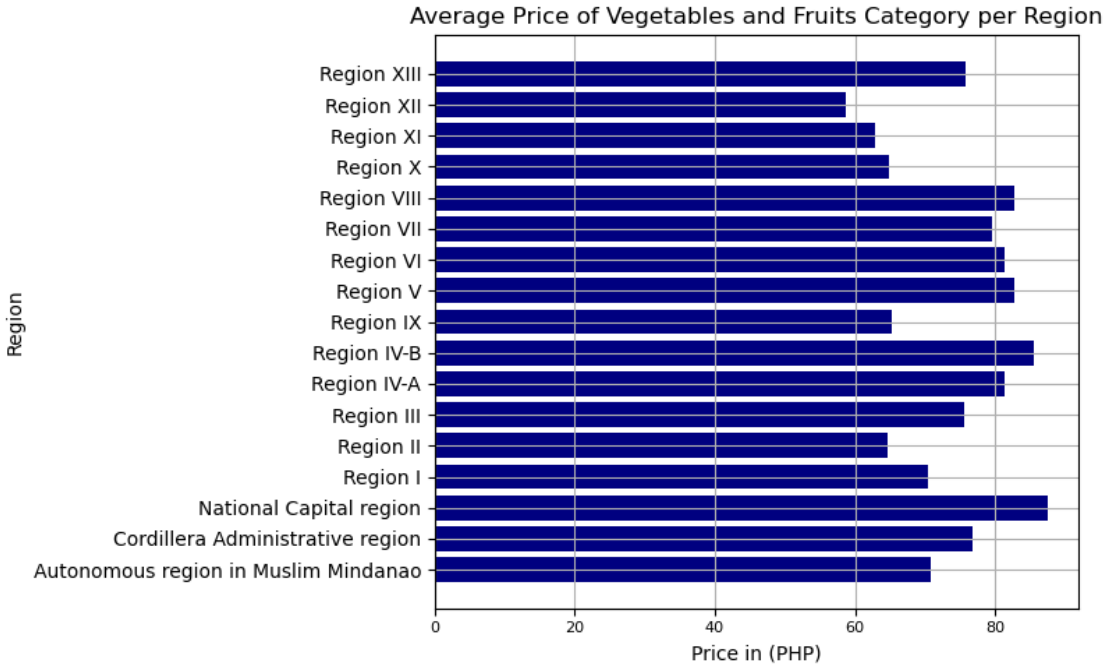
[48]:

region	price_php	usdprice	year
Autonomous region in Muslim Mindanao	70.741863	1.390529	2019.845730
Cordillera Administrative region	76.795134	1.517722	2020.146930
National Capital region	87.531794	1.739244	2017.904973
Region I	70.414719	1.393061	2019.706256
Region II	64.591358	1.288261	2019.608331
Region III	75.564632	1.474074	2020.495916
Region IV-A	81.430604	1.615207	2019.828786
Region IV-B	85.484509	1.692565	2020.064854
Region IX	65.232967	1.275358	2019.991204
Region V	82.787676	1.627502	2020.266034
Region VI	81.280037	1.601860	2020.069282
Region VII	79.636059	1.577670	2019.731325
Region VIII	82.734235	1.622589	2020.310377
Region X	64.788797	1.276744	2020.165195
Region XI	62.852949	1.238168	2019.981588
Region XII	58.673521	1.161302	2019.927937

```
Region XIII                            75.840019  1.492568  2020.286447
```

```
                                          month
region
Autonomous region in Muslim Mindanao  6.661846
Cordillera Administrative region      6.648246
National Capital region               6.373002
Region I                              6.658304
Region II                             6.642213
Region III                            6.649164
Region IV-A                           6.721882
Region IV-B                           6.720193
Region IX                             6.527740
Region V                              6.581769
Region VI                             6.739508
Region VII                            6.607439
Region VIII                           6.657765
Region X                              6.620429
Region XI                             6.601841
Region XII                            6.540952
Region XIII                           6.527808
```

[49]:
```
# Plotting
plt.figure(figsize=(8, 5))
plt.barh(reg_veg.index, reg_veg['price_php'], color= colors[3])
plt.xticks(size=8)
plt.title('Average Price of Vegetables and Fruits Category per Region')
plt.xlabel('Price in (PHP)')
plt.ylabel('Region')
plt.grid(True)

# Adjust the layout
plt.tight_layout()

# Display the plot
plt.show()
```



1.0.18 14. What are the average price of cereals and tubers in Region IV-A and Region IV-B over the years?

[50]:
```
food_prices['region'].unique()
```

[50]:
```
array(['National Capital region', 'Region III', 'Region IX', 'Region VI',
       'Region VII', 'Region XI', 'Region XII',
       'Autonomous region in Muslim Mindanao',
       'Cordillera Administrative region', 'Region I', 'Region II',
       'Region IV-A', 'Region IV-B', 'Region V', 'Region VIII',
       'Region X', 'Region XIII'], dtype=object)
```

[51]:
```
# Select the date wherein category is equals to cereals and region is equals to␣
 ↪reg IV-A
four_a_cereals = food_prices[(food_prices['category'] == 'cereals and tubers') &
                             (food_prices['region'] == 'Region IV-A')]

# Select the date wherein category is equals to cereals and region is equals to␣
 ↪reg IV-B
four_b_cereals = food_prices[(food_prices['category'] == 'cereals and tubers') &
                             (food_prices['region'] == 'Region IV-B')]
```

[52]:
```
four_a_cereals = four_a_cereals.groupby('year').mean(numeric_only = True)
four_a_cereals
```

[52]:

year	price_php	usdprice	month
2008	27.812500	0.627597	6.500000
2009	30.314412	0.638200	6.205882
2010	36.423333	0.809633	6.500000
2012	34.638387	0.821187	6.387097
2013	38.268462	0.929577	3.076923
2014	46.918000	1.064620	9.800000
2015	44.301250	0.973000	6.500000
2016	56.212500	1.185646	6.500000
2017	52.528750	1.042817	6.500000
2018	58.894167	1.117112	6.500000
2019	58.817000	1.127270	3.000000
2020	51.171308	1.043214	8.501558
2021	51.443467	1.046735	6.500000
2022	50.458514	0.927832	6.500000
2023	54.434161	0.984402	4.000000

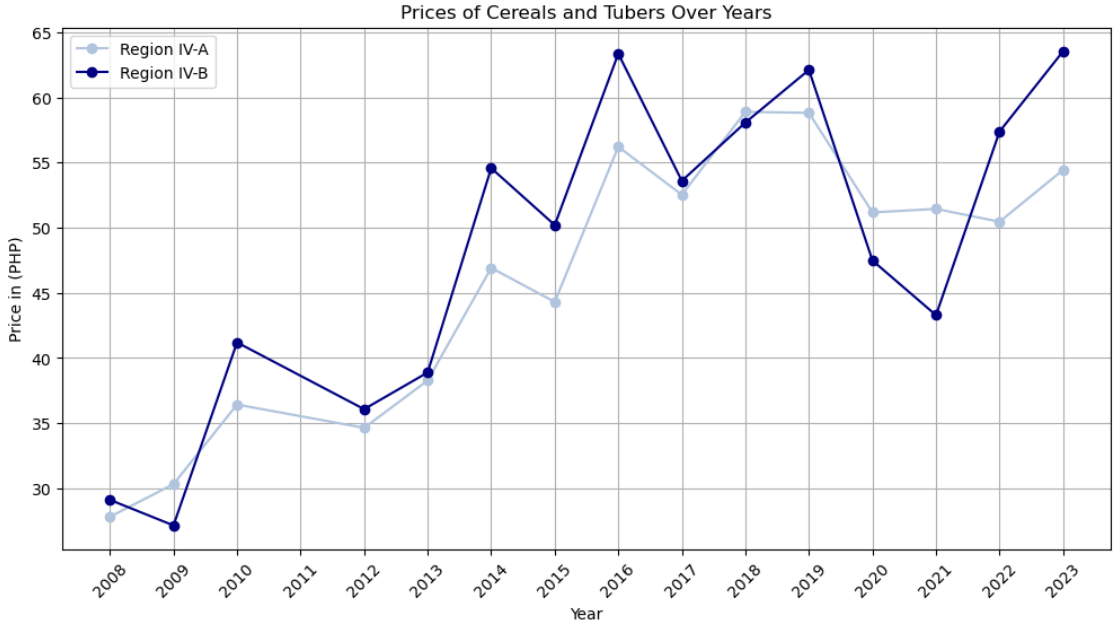
[53]:
```
four_b_cereals = four_b_cereals.groupby('year').mean(numeric_only = True)
four_b_cereals
```

[53]:

year	price_php	usdprice	month
2008	29.109444	0.655825	6.500000
2009	27.146667	0.577185	7.000000
2010	41.194167	0.915471	6.500000
2012	36.071290	0.855239	6.387097
2013	38.886923	0.944154	3.076923
2014	54.575000	1.238810	9.800000
2015	50.202500	1.103888	6.500000
2016	63.353750	1.339850	6.500000
2017	53.595417	1.064225	6.500000
2018	58.070455	1.103486	6.545455
2019	62.113000	1.190740	3.000000
2020	47.481134	0.969117	8.601375
2021	43.284740	0.880582	6.500000
2022	57.379432	1.055638	6.500000
2023	63.543182	1.149082	4.000000

[54]:
```
# X value from 2009-2023
years = range(2008,2024)

# Create and Display the plot lin line chart
fig, ax = plt.subplots(figsize=(12, 6))
ax.plot(four_a_cereals.index, four_a_cereals['price_php'], label= 'Region␣
 ↪IV-A',marker='o',color = colors[0])
```

```
ax.plot(four_b_cereals.index, four_b_cereals['price_php'], label= 'Region␣
 ↪IV-B',marker='o',color = colors[3])

# Set labels and title
ax.set_xlabel('Year')
ax.set_ylabel('Price in (PHP)')
ax.set_title('Prices of Cereals and Tubers Over Years')
plt.xticks(years, rotation=45)

# Add a legend
ax.legend()

# Display the plot
plt.grid(True)
plt.show()
```



1.0.19 15. What will be the price of Rice (regular, milled) in next five years?

[55]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

X = rice_regular.index.values.reshape(-1, 1)  # Years
y = rice_regular['price_php'].values  # Prices
```

```
# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict prices for the actual years and the next 5 years
all_years = np.concatenate((X, np.arange(2024, 2029).reshape(-1, 1)))
predicted_prices = model.predict(all_years)

# Plot the actual data, linear regression line, and predicted data
plt.figure(figsize=(12, 6))
plt.plot(X, y, label='Actual Data', marker='o')
plt.plot(all_years, predicted_prices, label='Predicted Data',␣
↪linestyle='dashed', color='red')
plt.xlabel('Year')
plt.ylabel('Price in (PHP)')
plt.title('Regular Rice Future Prices')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

[ ]: