Peace Is The key

Team YoungFervor, Binus University, Indonesia

Our Team

1. Laurentia Alyssa Castilani
2. Belinda Mutiara
3. Natasha Hartanti Winata
4. Gabrielle Felicia Ariyanto

Our Scope

Ukraine
Україна (Ukrainian)

Flag | Coat of arms

Anthem: Державний Гімн України
Derzhavnyi Himn Ukrainy
"State Anthem of Ukraine"

1:20

Show globe
Show map of Europe
Show all

Location of Ukraine (green)
Russian-occupied territories of Ukraine at the start of the 2022 Russian invasion (light green)

Capital and largest city	Kyiv 49°N 32°E
Official language and national language	Ukrainian[1]
Ethnic groups (2001)[2]	77.8% Ukrainians 17.3% Russians 4.9% Others
Religion (2018)[3]	87.3% Christianity 11.0% No religion 0.8% Others 0.9% Unanswered
Demonym(s)	Ukrainian
Government	Unitary semi-presidential republic
• President	Volodymyr Zelenskyy
• Prime Minister	Denys Shmyhal
• Chairman of the Verkhovna Rada	Ruslan Stefanchuk
Legislature	Verkhovna Rada

Area	
• Total	603,628[4] km² (233,062 sq mi) (45th)
• Water (%)	3.8[5]
Population	
• January 2022 estimate	▼ 41,167,336[6] (excluding Crimea) (36th)
• 2001 census	48,457,102[2]
• Density	73.8/km² (191.1/sq mi) (115th)
GDP (PPP)	2021 estimate
• Total	▲ $588 billion[7]
• Per capita	▲ $14,330[7]
GDP (nominal)	2021 estimate
• Total	▲ $198 billion[7]
• Per capita	▲ $4,830[7]

source: en.wikipedia.org

Why Ukraine Economics ?

- **Ukraine is currently the hottest topic all over the world.**
- **Food crisis happens in developing countries as they depend on Ukraine's exported goods.**

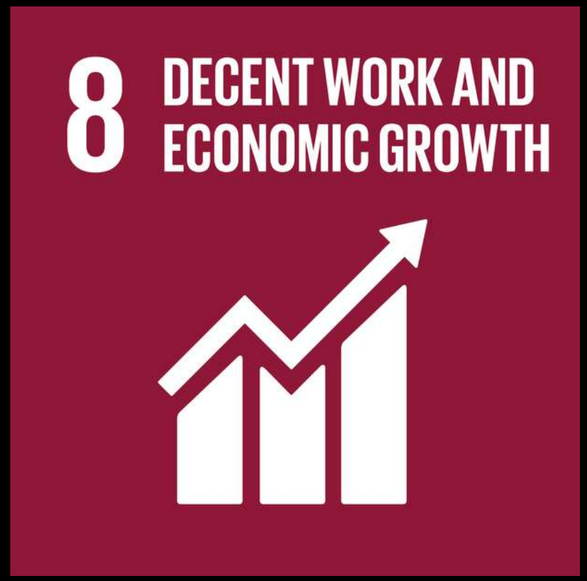
What we aim to solve today?

Situation	Under the pressure of rising food, energy, and key commodity prices, the conflict against Ukraine has been accompanied by a rapid increase in inflation.
Vision	Ukraine economic recovery as soon as possible

SDG goals

Target 8.1

Sustain per capita **economic growth** in accordance with national circumstances and, in particular, at least 7 percent gross domestic product growth per annum in the least developed countries.



source: https://sdg-tracker.org

SDG goals

Target 16.A

Strengthen relevant national institutions, including through international cooperation, for building capacity at all levels, in particular in developing countries, to prevent violence and combat terrorism and crime

source: https://sdg-tracker.org

Load Libraries

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

!pip install scikit-plot
import scikitplot as skplt
```

Load several libraries that will be used in the analysis process

Ukraine Peace Index 2022

Global Peace Index Dataset

Load Data

```
Peace_Index = pd.read_csv("https://raw.githubusercontent.com/belindamutiaraaaaa/data/main/Peace_Index.csv")
```

```
Peace_Index.head()
```

	Country	Rank	Score
0	Iceland	1	1107
1	New Zealand	2	1269
2	Ireland	3	1288
3	Denmark	4	1296
4	Austria	5	1300

```
[ ]  Peace_Index.tail()
```

	Country	Rank	Score
158	South Sudan	159	3184
159	Russia	160	3275
160	Syria	161	3356
161	Yemen	162	3394
162	Afghanistan	163	3554

import data

Data Description

Country : Country name

Rank : The order of the countries from the most peaceful to the least peaceful

Score : a country's peace index

Missing Value & Data Type

```
Peace_Index.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 163 entries, 0 to 162
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Country  163 non-null    object
 1   Rank     163 non-null    object
 2   Score    163 non-null    int64
dtypes: int64(1), object(2)
memory usage: 3.9+ KB
```

From the output above it can be seen:

1. No missing values in any column
2. There is only one data type that is string
3. The data types of the Rank still not appropriate, Rank should have an integer data type.

Checking Duplicate

```
duplicate_PI = Peace_Index[Peace_Index.duplicated()]
duplicate_PI
```

	Country	Rank	Score

There is no duplication in Peace Index dataset

Transform Data Type

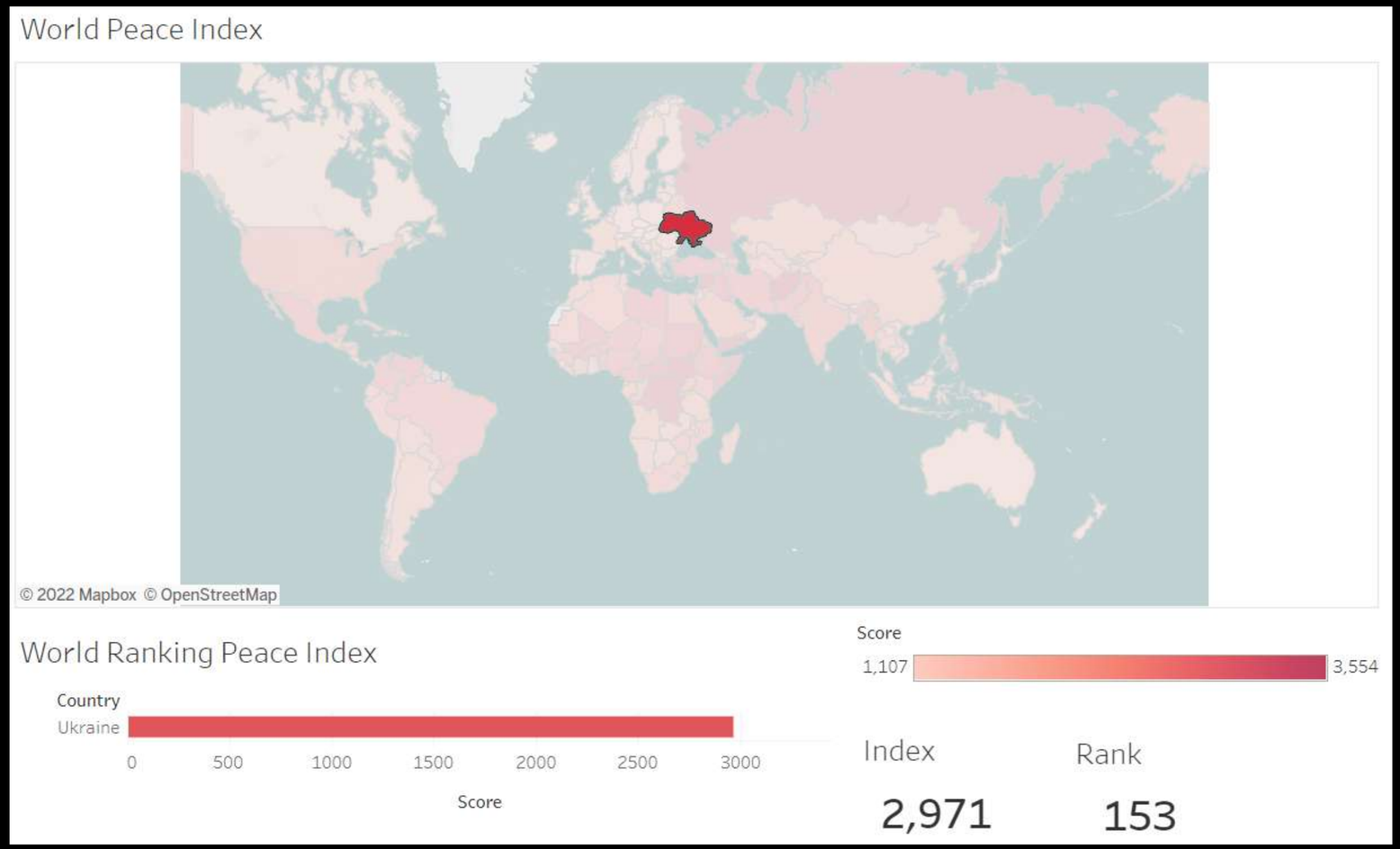
```
[ ]  Peace_Index.loc[Peace_Index["Rank"] == "48=", "Rank"] = "48"

[ ]  Peace_Index.loc[Peace_Index["Rank"] == "62=", "Rank"] = "62"

[ ]  Peace_Index.loc[Peace_Index["Rank"] == "65=", "Rank"] = "65"

[ ]  Peace_Index.loc[Peace_Index["Rank"] == "75=", "Rank"] = "75"

[ ]  Peace_Index.loc[Peace_Index["Rank"] == "81=", "Rank"] = "81"

[ ]  Peace_Index.loc[Peace_Index["Rank"] == "86=", "Rank"] = "86"
```


```
[ ]  data_types_dict = {'Rank': int}
     Peace_Index = Peace_Index.astype(data_types_dict)
```

Some values in the value column do not match the integer standard, for example there is a value "81=" we should change it to 81 and so on for the same case.

Now the Rank column is clean from input errors!

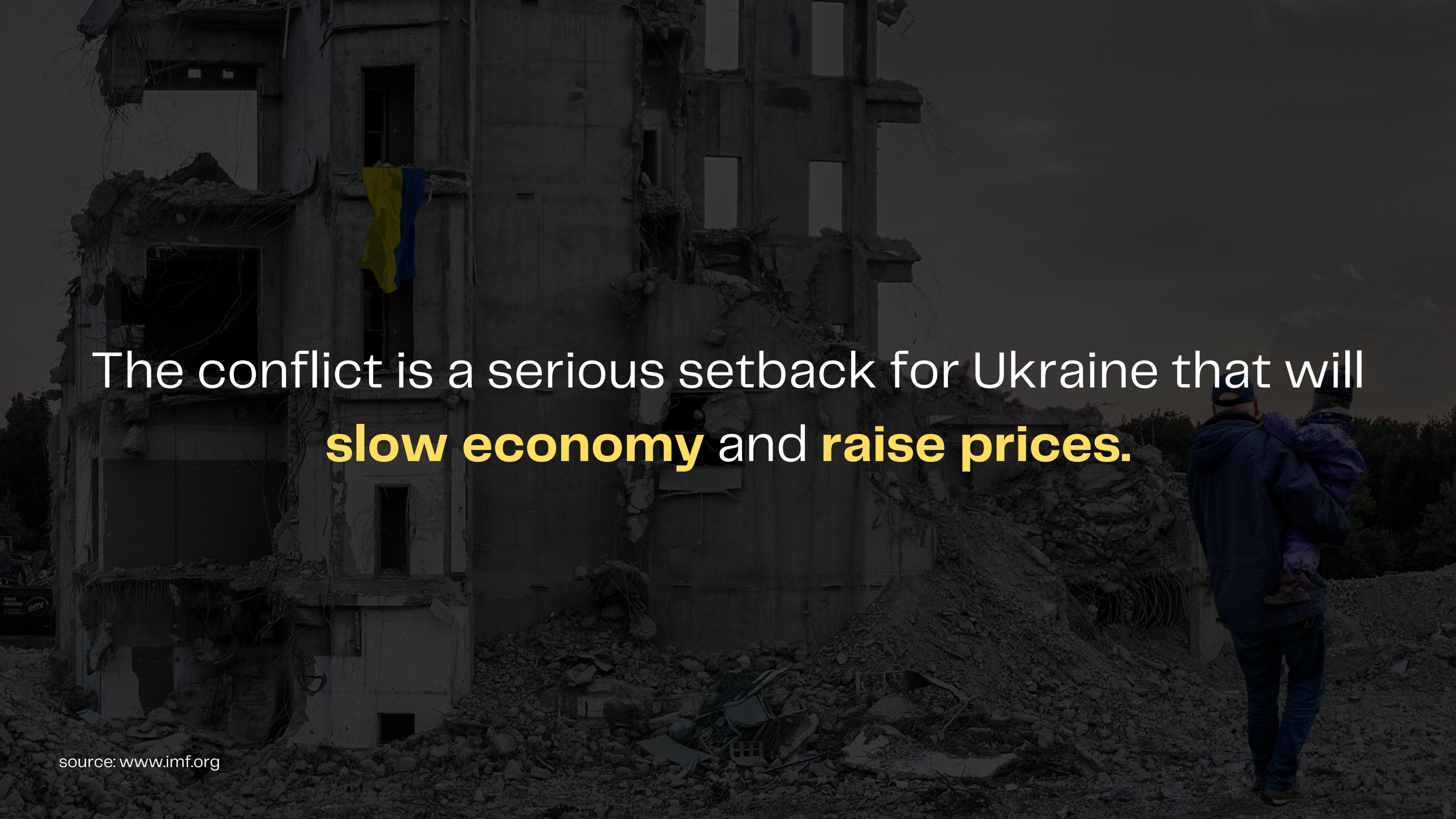
Ukraine Peace Index Map Distribution



in 2022 Ukraine ranks **153rd** most peaceful country in the world, with a score of **2,971**

source: www.visionofhumanity.org

Dashboard: https://public.tableau.com/app/profile/belinda.mutiara/viz/WorldPeaceIndex2022/Dashboard1



The conflict is a serious setback for Ukraine that will **slow economy** and **raise prices.**

source: www.imf.org

Ukraine's Producer Price Index

Ukraine Producer Price Index Dataset

Load Data

```
UKR_PPI = pd.read_csv("https://raw.githubusercontent.com/laurentiaalyssa/Data/main/Producer%20Prices%20data%20for%20Ukraine.csv")
```

```
UKR_PPI.head(5)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item Code (CPC)	Item	Element Code	Element	Year Code	Year	Months Code	Months	Unit	Value	Flag
0	#country+code	#date+start	#date+end	NaN	NaN	#country+name	#indicator+code	NaN	#indicator+name	NaN	NaN	NaN	#date+year	NaN	NaN	#indicator+type	#indicator+value+num	NaN
1	UKR	1994-01-01	1994-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1994.0	1994	7021.0	Annual value	NaN	0.790000	I
2	UKR	1995-01-01	1995-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1995.0	1995	7021.0	Annual value	NaN	4.090000	I
3	UKR	1996-01-01	1996-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1996.0	1996	7021.0	Annual value	NaN	6.800000	I
4	UKR	1997-01-01	1997-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1997.0	1997	7021.0	Annual value	NaN	8.030000	I

```
UKR_PPI.tail(5)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item Code (CPC)	Item	Element Code	Element	Year Code	Year	Months Code	Months	Unit	Value	Flag
9751	UKR	2017-01-01	2017-12-31	230.0	'804	Ukraine	1800	'F1800	Vegetables&Melons, Total	5539.0	Producer Price Index (2014-2016 = 100)	2017.0	2017	7021.0	Annual value	NaN	131.640000	I
9752	UKR	2018-01-01	2018-12-31	230.0	'804	Ukraine	1800	'F1800	Vegetables&Melons, Total	5539.0	Producer Price Index (2014-2016 = 100)	2018.0	2018	7021.0	Annual value	NaN	149.190000	I
9753	UKR	2019-01-01	2019-12-31	230.0	'804	Ukraine	1800	'F1800	Vegetables&Melons, Total	5539.0	Producer Price Index (2014-2016 = 100)	2019.0	2019	7021.0	Annual value	NaN	154.420000	I
9754	UKR	2020-01-01	2020-12-31	230.0	'804	Ukraine	1800	'F1800	Vegetables&Melons, Total	5539.0	Producer Price Index (2014-2016 = 100)	2020.0	2020	7021.0	Annual value	NaN	150.510000	I
9755	UKR	2021-01-01	2021-12-31	230.0	'804	Ukraine	1800	'F1800	Vegetables&Melons, Total	5539.0	Producer Price Index (2014-2016 = 100)	2021.0	2021	7021.0	Annual value	NaN	160.570000	I

Data Description

```
UKR_PPI.columns
```

```
Index(['Iso3', 'StartDate', 'EndDate', 'Area Code', 'Area Code (M49)', 'Area',
       'Item Code', 'Item Code (CPC)', 'Item', 'Element Code', 'Element',
       'Year Code', 'Year', 'Months Code', 'Months', 'Unit', 'Value', 'Flag'],
      dtype='object')
```

Iso3: Three-letter country code
StartDate: The planned date on when the data will be recorded (yyyy-mm-dd)
EndDate: The planned date on when the data will stop being recorded (yyyy-mm-dd)
Area Code:
Area Code (M49): Standard Country or Area Codes for Statistical Use is a standard for area codes used by the United Nations
Area: Country name
Item Code: The factor type which represent the item in "Item" column.
I**tem Code(CPC)**: The factor type which represent the item in "Item" column.
Item: Category which the value represent.

Element Code :
Element :
Year Code: The factor type which represent the item in "Year" column.
Year: The year in which the data was recorded.
Month Code: The factor type which represent the item in "Months" column.
Months: The name of the month
Unit: The unit for the value displayed in the column "Value".
Value: The value of CPI
Flag:

Missing Value & Data Type

```
UKR_PPI.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9756 entries, 0 to 9755
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Iso3             9756 non-null   object 
 1   StartDate        9756 non-null   object 
 2   EndDate          9756 non-null   object 
 3   Area Code        9755 non-null   float64
 4   Area Code (M49)  9755 non-null   object 
 5   Area             9756 non-null   object 
 6   Item Code        9756 non-null   object 
 7   Item Code (CPC)  9755 non-null   object 
 8   Item             9756 non-null   object 
 9   Element Code     9755 non-null   float64
 10  Element          9755 non-null   object 
 11  Year Code        9755 non-null   float64
 12  Year             9756 non-null   object 
 13  Months Code      9755 non-null   float64
 14  Months           9755 non-null   object 
 15  Unit             6551 non-null   object 
 16  Value            9756 non-null   object 
 17  Flag             9755 non-null   object 
dtypes: float64(4), object(14)
memory usage: 1.3+ MB
```

From the output above it can be seen:

1. There is missing value in 'Unit' column, but it won't be a problem because we will only use the LCU unit

2. There are some columns that do not match the data type, which are 'StartDate', 'EndDate', 'Year', 'Value'

Checking Duplicate

```
duplicate_UKR_PPI = UKR_PPI[UKR_PPI.duplicated()]
duplicate_UKR_PPI
```

Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item Code (CPC)	Item	Element Code	Element	Year Code	Year	Months Code	Months	Unit	Value	Flag

Every records in the dataset are unique.

There is no duplication in Producer Price Index dataset

Checking Unique Value

In row index 0, there is a data description so we're going to just ignore that while looking at the unique values

```
UKR_PPI['Item'].unique()
```

```
array(['#indicator+name', 'Almonds, in shell',
       'Anise, badian, coriander, cumin, caraway, fennel and juniper berries, raw',
       'Apples', 'Apricots', 'Barley', 'Beans, dry', 'Blueberries',
       'Broad beans and horse beans, dry',
       'Broad beans and horse beans, green', 'Buckwheat', 'Cabbages',
       'Cantaloupes and other melons', 'Carrots and turnips',
       'Castor oil seeds', 'Cauliflowers and broccoli', 'Cereals n.e.c.',
       'Cherries', 'Chestnuts, in shell', 'Chicory roots',
       'Chillies and peppers, green (Capsicum spp. and Pimenta spp.)',
       'Cranberries', 'Cucumbers and gherkins', 'Currants',
       'Eggplants (aubergines)',
       'Eggs from other birds in shell, fresh, n.e.c.',
       'Flax, processed but not spun', 'Gooseberries', 'Grapes',
       'Green garlic', 'Hazelnuts, in shell', 'Hempseed',
       'Hen eggs in shell, fresh', 'Hop cones',
       'Horse meat, fresh or chilled',
       'Horse meat, fresh or chilled (biological)', 'Lentils, dry',
       'Lettuce and chicory', 'Linseed', 'Locust beans (carobs)',
       'Lupins', 'Maize (corn)',
       'Meat of cattle with the bone, fresh or chilled',
       'Meat of cattle with the bone, fresh or chilled (biological)',
       'Meat of chickens, fresh or chilled',
       'Meat of chickens, fresh or chilled (biological)',
       'Meat of goat, fresh or chilled',
       'Meat of goat, fresh or chilled (biological)',
       'Meat of pig with the bone, fresh or chilled',
       'Meat of pig with the bone, fresh or chilled (biological)',
       'Meat of rabbits and hares, fresh or chilled',
       'Meat of rabbits and hares, fresh or chilled (biological)',
       'Meat of sheep, fresh or chilled',
       'Meat of sheep, fresh or chilled (biological)', 'Millet',
       'Mushrooms and truffles', 'Mustard seed', 'Natural honey', 'Oats',
       'Onions and shallots, dry (excluding dehydrated)',
       'Onions and shallots, green', 'Other beans, green',
       'Other berries and fruits of the genus vaccinium n.e.c.',
       'Other fruits, n.e.c.',
       'Other nuts (excluding wild edible nuts and groundnuts), in shell, n.e.c.',
       'Other oil seeds, n.e.c.', 'Other pome fruits',
       'Other pulses n.e.c.', 'Other stone fruits',
       'Other vegetables, fresh n.e.c.', 'Peaches and nectarines',
       'Pears', 'Peas, dry', 'Peas, green', 'Plums and sloes', 'Potatoes',
       'Pumpkins, squash and gourds', 'Quinces', 'Rape or colza seed',
       'Raspberries', 'Raw milk of camel', 'Raw milk of cattle',
       'Raw milk of goats', 'Raw milk of sheep', 'Rice', 'Rye',
       'Shorn wool, greasy, including fleece-washed shorn wool',
       'Sorghum', 'Sour cherries', 'Soya beans', 'Strawberries',
       'Sugar beet', 'Sunflower seed', 'Tomatoes',
       'True hemp, raw or retted', 'Unmanufactured tobacco', 'Vetches',
       'Walnuts, in shell', 'Watermelons', 'Wheat', 'Agriculture',
       'Cereals, Total', 'Coarse Grain, Total', 'Eggs Primary',
       'Fibre Crops Primary', 'Fruit excl Melons, Total', 'Fruit Primary',
       'Livestock', 'Meat, Total', 'Milk, Total',
       'Oilcrops, Oil Equivalent', 'Pulses, Total',
       'Roots and Tubers, Total', 'Treenuts, Total', 'Vegetables Primary',
       'Vegetables&Melons, Total'], dtype=object)
```

```
UKR_PPI['Year'].unique()
```

```
array(['#date+year', '1994', '1995', '1996', '1997', '1998', '1999',
       '2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007',
       '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015',
       '2016', '2017', '2018', '2019', '2020', '2021'], dtype=object)
```

```
UKR_PPI['Flag'].unique()
```

```
array([nan, 'I', 'A'], dtype=object)
```

```
UKR_PPI['Unit'].unique()
```

```
array(['#indicator+type', nan, 'LCU', 'SLC', 'USD'], dtype=object)
```

```
UKR_PPI['Element'].unique()
```

```
array([nan, 'Producer Price Index (2014-2016 = 100)',
       'Producer Price (LCU/tonne)', 'Producer Price (SLC/tonne)',
       'Producer Price (USD/tonne)'], dtype=object)
```

```
UKR_PPI['Months'].unique()
```

```
array([nan, 'Annual value', 'January', 'February', 'March', 'April',
       'May', 'June', 'July', 'August', 'September', 'October',
       'November', 'December'], dtype=object)
```

Dropping Row(s) that are/is not needed

```
UKR_PPI = UKR_PPI.drop(labels = 0, axis = 0)
UKR_PPI.head(5)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item Code (CPC)	Item	Element Code	Element	Year Code	Year	Months Code	Months	Unit	Value	Flag
1	UKR	1994-01-01	1994-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1994.0	1994	7021.0	Annual value	NaN	0.790000	I
2	UKR	1995-01-01	1995-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1995.0	1995	7021.0	Annual value	NaN	4.090000	I
3	UKR	1996-01-01	1996-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1996.0	1996	7021.0	Annual value	NaN	6.800000	I
4	UKR	1997-01-01	1997-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1997.0	1997	7021.0	Annual value	NaN	8.030000	I
5	UKR	1998-01-01	1998-12-31	230.0	'804	Ukraine	221	'01371	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1998.0	1998	7021.0	Annual value	NaN	9.000000	I

The first row (index 0) only contain description of each column.

By doing so, we removed a unique value or even a missing value from each column (Improving data quality)

Rename Variables

```
UKR_PPI.rename(columns = {'Area Code': 'AreaCode',
                          'Area Code (M49)':'AreaCode(M49)',
                          'Item Code': 'ItemCode',
                          'Item Code (CPC)' : 'ItemCode(CPC)',
                          'Element Code' : 'ElementCode',
                          'Year Code' : 'YearCode',
                          'Months Code' : 'MonthsCode'}, inplace = True)
UKR_PPI.columns
```

```
Index(['Iso3', 'StartDate', 'EndDate', 'AreaCode', 'AreaCode(M49)', 'Area',
       'ItemCode', 'ItemCode(CPC)', 'Item', 'ElementCode', 'Element',
       'YearCode', 'Year', 'MonthsCode', 'Months', 'Unit', 'Value', 'Flag'],
      dtype='object')
```

Some variable names have spaces consequently, we have to remove the spaces by renaming variables

Changing Data Type

```
UKR_PPI['StartDate'] = pd.to_datetime(UKR_PPI['StartDate'])
UKR_PPI['EndDate'] = pd.to_datetime(UKR_PPI['EndDate'])
UKR_PPI['Year'] = UKR_PPI['Year'].astype(int)
UKR_PPI['Value'] = UKR_PPI['Value'].astype(float)
UKR_PPI.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9755 entries, 1 to 9755
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Iso3           9755 non-null   object
 1   StartDate      9755 non-null   datetime64[ns]
 2   EndDate        9755 non-null   datetime64[ns]
 3   AreaCode       9755 non-null   float64
 4   AreaCode(M49)  9755 non-null   object
 5   Area           9755 non-null   object
 6   ItemCode       9755 non-null   object
 7   ItemCode(CPC)  9755 non-null   object
 8   Item           9755 non-null   object
 9   ElementCode    9755 non-null   float64
 10  Element        9755 non-null   object
 11  YearCode       9755 non-null   float64
 12  Year           9755 non-null   int64
 13  MonthsCode     9755 non-null   float64
 14  Months         9755 non-null   object
 15  Unit           6550 non-null   object
 16  Value          9755 non-null   float64
 17  Flag           9755 non-null   object
dtypes: datetime64[ns](2), float64(5), int64(1), object(10)
memory usage: 1.4+ MB
```

There are some columns that do not match the data type, which are 'StartDate', 'EndDate', 'Year', 'Value'.

Fix data type by changing it :
'StartDate' : From string to date time
'EndDate' : From string to date time
'Year' : From string to integer
'Value' : From string to float

Now the column data types are fixed!
Therefore we can proceed with our analysis

Dropping Variables (Column)

```
#Select the column to be used in the analysis
UKR_PPI = UKR_PPI.drop(columns = ['Iso3', 'AreaCode', 'AreaCode(M49)', 'ItemCode', 'ItemCode(CPC)', 'YearCode', 'MonthsCode'])
UKR_PPI.head(5)
```

	StartDate	EndDate	Area	Item	ElementCode	Element	Year	Months	Unit	Value	Flag
1	1994-01-01	1994-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1994	Annual value	NaN	0.79	I
2	1995-01-01	1995-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1995	Annual value	NaN	4.09	I
3	1996-01-01	1996-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1996	Annual value	NaN	6.80	I
4	1997-01-01	1997-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1997	Annual value	NaN	8.03	I
5	1998-01-01	1998-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	1998	Annual value	NaN	9.00	I

Reasoning for variables removal:

- **Iso3**: This dataset focuses only in Ukraine so there is no need for Ukraine's Iso3
- **Area Code, Area Code(M49), Area**: All of the data is located in Ukraine so these variables has the same values in each rows
- **Item code, ItemCode(CPC), YearCode, MonthsCode**: We decided to use the 'Item', 'Year', and 'Months' variables which already represent the whole

Filter Data

```
#Filter data from 2000 and use only Annual value
UKR_PPI = UKR_PPI.loc[(UKR_PPI['Year'] >= 2000) & (UKR_PPI['Months'] == 'Annual value')]
UKR_PPI.head(5)
```

	StartDate	EndDate	Area	Item	ElementCode	Element	Year	Months	Unit	Value	Flag
7	2000-01-01	2000-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	2000	Annual value	NaN	14.11	I
8	2001-01-01	2001-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	2001	Annual value	NaN	15.52	I
9	2002-01-01	2002-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	2002	Annual value	NaN	16.31	I
10	2003-01-01	2003-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	2003	Annual value	NaN	19.68	I
11	2004-01-01	2004-12-31	Ukraine	Almonds, in shell	5539.0	Producer Price Index (2014-2016 = 100)	2004	Annual value	NaN	20.74	I

We decided to use 2000 as the starting year for all dataset and use only Annual value to see the development of Producer Price Index from year to year.

Select data

```
#Select only LCU (Local Currency Units)

UKR_PPI_LCU = UKR_PPI.loc[(UKR_PPI['Unit'] == 'LCU')]
UKR_PPI_LCU.head(5)
```

	StartDate	EndDate	Area	Item	ElementCode	Element	Year	Months	Unit	Value	Flag
63	2000-01-01	2000-12-31	Ukraine	Apples	5530.0	Producer Price (LCU/tonne)	2000	Annual value	LCU	306.0	A
64	2001-01-01	2001-12-31	Ukraine	Apples	5530.0	Producer Price (LCU/tonne)	2001	Annual value	LCU	436.0	A
65	2002-01-01	2002-12-31	Ukraine	Apples	5530.0	Producer Price (LCU/tonne)	2002	Annual value	LCU	376.0	A
66	2003-01-01	2003-12-31	Ukraine	Apples	5530.0	Producer Price (LCU/tonne)	2003	Annual value	LCU	304.0	A
67	2004-01-01	2004-12-31	Ukraine	Apples	5530.0	Producer Price (LCU/tonne)	2004	Annual value	LCU	561.0	A

Select only LCU (Local Currency Units)

Select data

```
#Select only SLC (Standard Local Currency Units)

UKR_PPI_SLC = UKR_PPI.loc[(UKR_PPI['Unit'] == 'SLC')]
UKR_PPI_SLC.head(5)
```

	StartDate	EndDate	Area	Item	ElementCode	Element	Year	Months	Unit	Value	Flag
90	2000-01-01	2000-12-31	Ukraine	Apples	5531.0	Producer Price (SLC/tonne)	2000	Annual value	SLC	306.0	A
91	2001-01-01	2001-12-31	Ukraine	Apples	5531.0	Producer Price (SLC/tonne)	2001	Annual value	SLC	436.0	A
92	2002-01-01	2002-12-31	Ukraine	Apples	5531.0	Producer Price (SLC/tonne)	2002	Annual value	SLC	376.0	A
93	2003-01-01	2003-12-31	Ukraine	Apples	5531.0	Producer Price (SLC/tonne)	2003	Annual value	SLC	304.0	A
94	2004-01-01	2004-12-31	Ukraine	Apples	5531.0	Producer Price (SLC/tonne)	2004	Annual value	SLC	561.0	A

Select only SLC (Standard Local Currency Units)

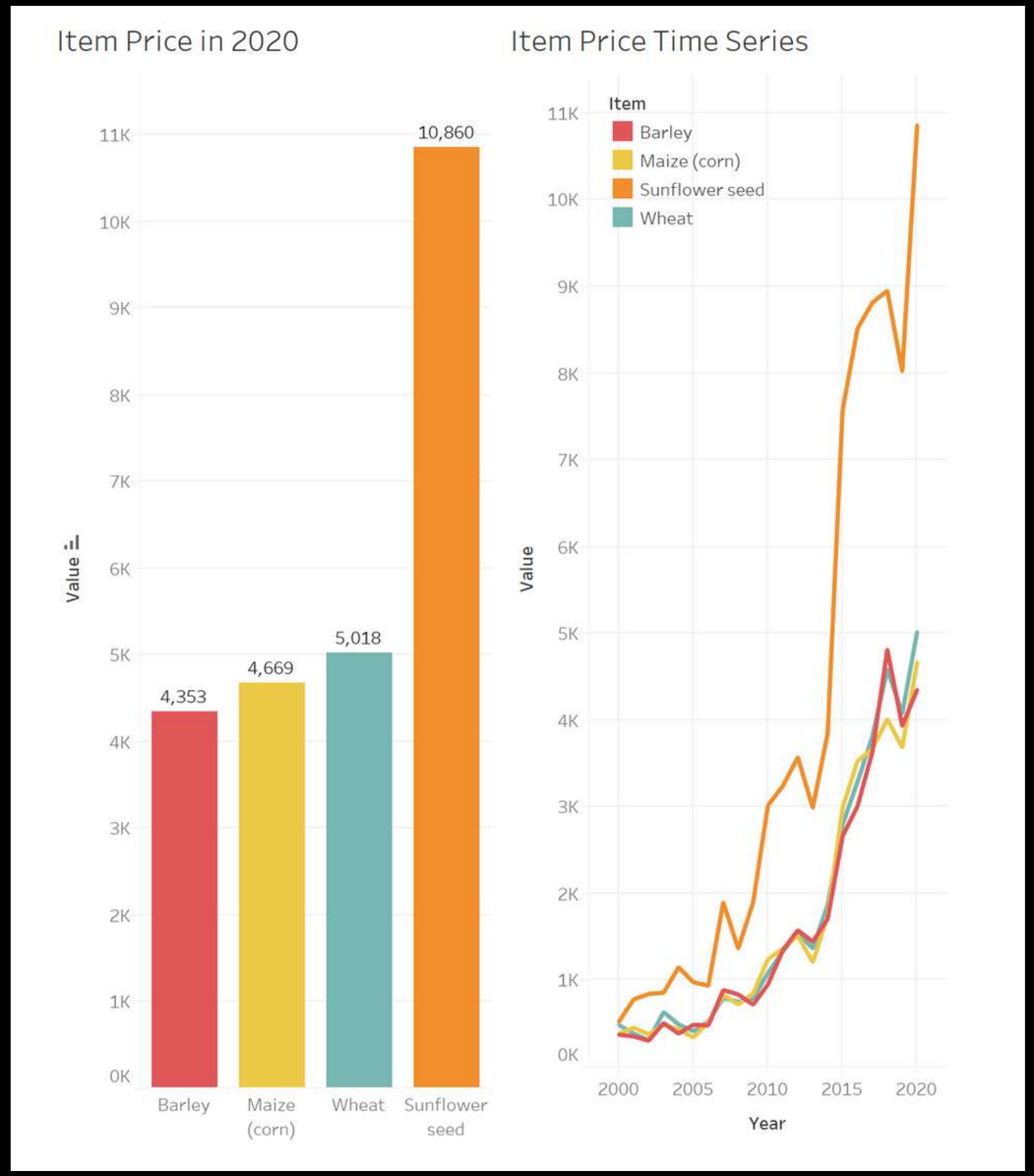
```
#Select only USD (United States Dollar)

UKR_PPI_USD = UKR_PPI.loc[(UKR_PPI['Unit'] == 'USD')]
UKR_PPI_USD.head(5)
```

	StartDate	EndDate	Area	Item	ElementCode	Element	Year	Months	Unit	Value	Flag
117	2000-01-01	2000-12-31	Ukraine	Apples	5532.0	Producer Price (USD/tonne)	2000	Annual value	USD	56.2	A
118	2001-01-01	2001-12-31	Ukraine	Apples	5532.0	Producer Price (USD/tonne)	2001	Annual value	USD	81.2	A
119	2002-01-01	2002-12-31	Ukraine	Apples	5532.0	Producer Price (USD/tonne)	2002	Annual value	USD	70.6	A
120	2003-01-01	2003-12-31	Ukraine	Apples	5532.0	Producer Price (USD/tonne)	2003	Annual value	USD	57.0	A
121	2004-01-01	2004-12-31	Ukraine	Apples	5532.0	Producer Price (USD/tonne)	2004	Annual value	USD	105.5	A

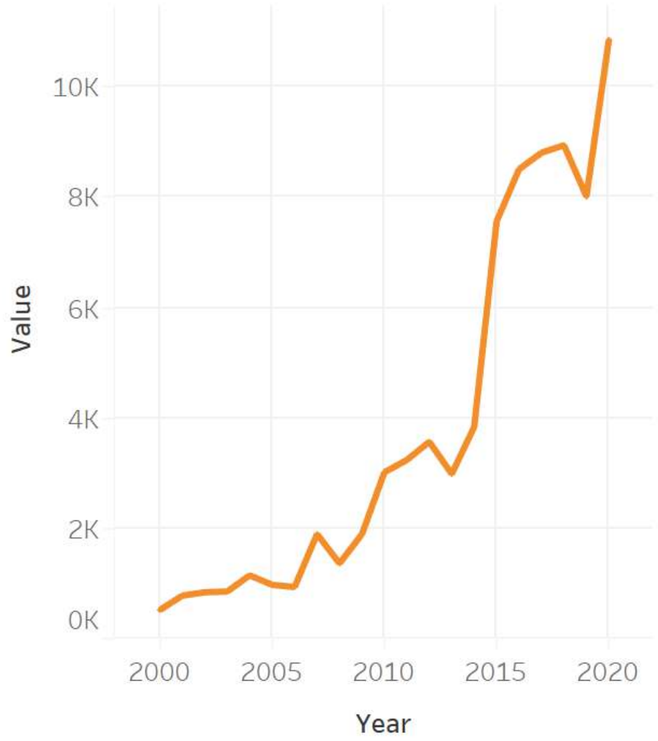
Select only USD (United States Dollar)

UKRAINE'S PPI INDEX

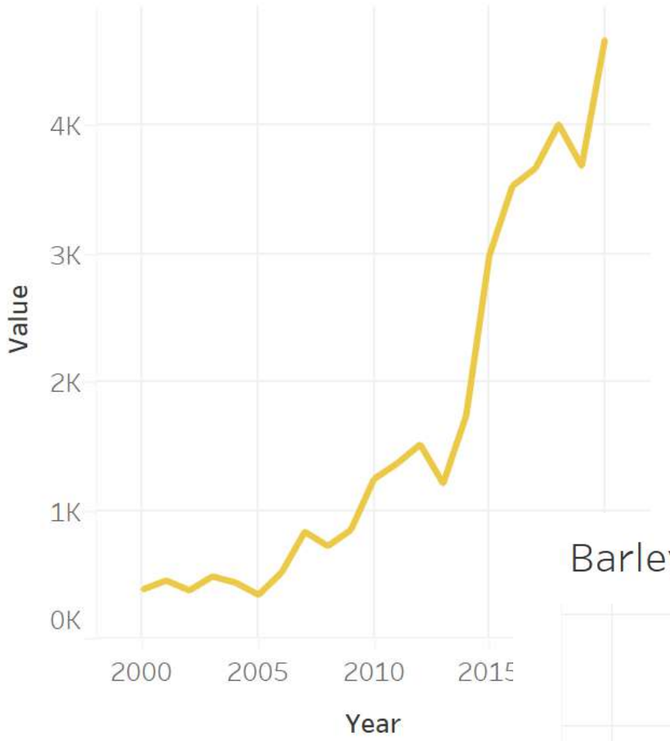
Untill 2020, Ukraine has experienced **an increase** in PPI including the products that **they are able to produce** themself



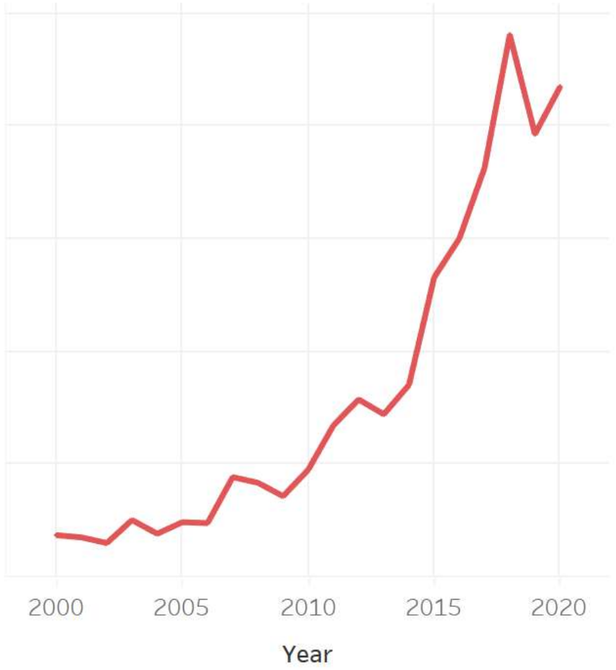
Sunflower Seed



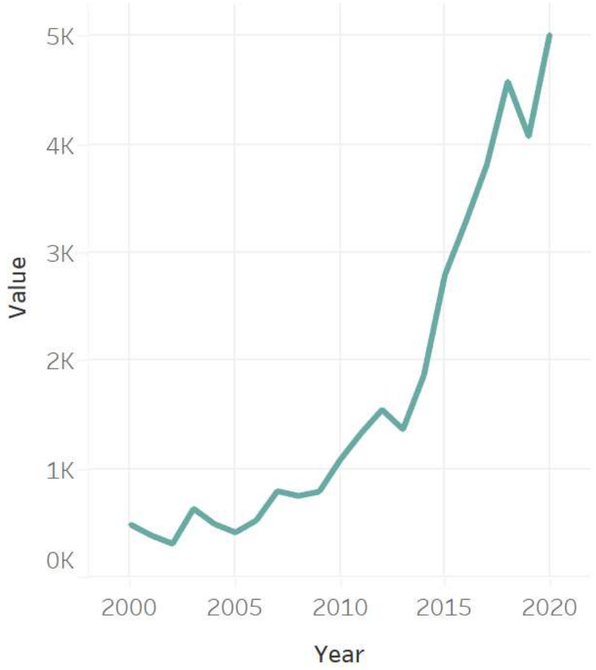
Maize (corn)



Barley



Wheat



Customer Production Rate

Customer Production Rate Dataset

Load Data

```
UKR_CPI = pd.read_csv("https://raw.githubusercontent.com/laurentiaalyssa/Data/main/Consumer%20Price%20Indices%20data%20for%20Ukraine.csv")
```

```
UKR_CPI.head(6)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item	Months Code	Months	Year Code	Year	Unit	Value	Flag	Note
0	#country+code	#date+start	#date+end	NaN	NaN	#country+name	#indicator+code	#indicator+name	NaN	NaN	NaN	#date+year	#indicator+type	#indicator+value+num	NaN	NaN
1	UKR	2000-01-01	2000-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2000.0	2000	NaN	19.586001	I	base year is 2015
2	UKR	2001-01-01	2001-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2001.0	2001	NaN	23.565986	I	base year is 2015
3	UKR	2002-01-01	2002-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2002.0	2002	NaN	24.766656	I	base year is 2015
4	UKR	2003-01-01	2003-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2003.0	2003	NaN	24.766656	I	base year is 2015
5	UKR	2004-01-01	2004-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2004.0	2004	NaN	26.644501	I	base year is 2015

```
UKR_CPI.tail(5)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item	Months Code	Months	Year Code	Year	Unit	Value	Flag	Note
794	UKR	2017-12-01	2017-12-31	230.0	'804	Ukraine	23014	Food price inflation	7012.0	December	2017.0	2017	%	17.722222	E	NaN
795	UKR	2018-12-01	2018-12-31	230.0	'804	Ukraine	23014	Food price inflation	7012.0	December	2018.0	2018	%	7.786692	E	NaN
796	UKR	2019-12-01	2019-12-31	230.0	'804	Ukraine	23014	Food price inflation	7012.0	December	2019.0	2019	%	4.772329	E	NaN
797	UKR	2020-12-01	2020-12-31	230.0	'804	Ukraine	23014	Food price inflation	7012.0	December	2020.0	2020	%	4.931049	E	NaN
798	UKR	2021-12-01	2021-12-31	230.0	'804	Ukraine	23014	Food price inflation	7012.0	December	2021.0	2021	%	12.743927	E	NaN

Data Description

```
UKR_CPI.columns
```

```
Index(['Iso3', 'StartDate', 'EndDate', 'Area Code', 'Area Code (M49)', 'Area',
       'Item Code', 'Item', 'Months Code', 'Months', 'Year Code', 'Year',
       'Unit', 'Value', 'Flag', 'Note'],
      dtype='object')
```

Iso3: Three-letter country code
StartDate: The planned date on when the data will be recorded (yyyy-mm-dd)
EndDate: The planned date on when the data will stop being recorded (yyyy-mm-dd)
Area Code:
Area Code (M49): Standard Country or Area Codes for Statistical Use is a standard for area codes used by the United Nations
Area: Country name
Item Code: The factor type which represent the item in "Item" column.
Item: Category which the value represent.
Month Code: The factor type which represent the item in "Months" column.
Months: The name of the month
Year Code: The factor type which represent the item in "Year" column.
Year: The year in which the data was recorded.
Unit: The unit for the value displayed in the column "Value".
Value: The value of CPI
Flag:
Note: Side note about the record.

Missing Values and Data Type

```
UKR_CPI.shape
```

```
(799, 16)
```

Dimension of our dataset:
799 records and 16 variables

There are some variables with some missing values

```
UKR_CPI.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 799 entries, 0 to 798
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Iso3              799 non-null    object 
 1   StartDate         799 non-null    object 
 2   EndDate           799 non-null    object 
 3   Area Code         798 non-null    float64
 4   Area Code (M49)   798 non-null    object 
 5   Area              799 non-null    object 
 6   Item Code         799 non-null    object 
 7   Item              799 non-null    object 
 8   Months Code       798 non-null    float64
 9   Months            798 non-null    object 
 10  Year Code         798 non-null    float64
 11  Year              799 non-null    object 
 12  Unit              259 non-null    object 
 13  Value             799 non-null    object 
 14  Flag              798 non-null    object 
 15  Note              540 non-null    object 
dtypes: float64(3), object(13)
memory usage: 100.0+ KB
```

Checking Duplicates

```
duplicate_UKR_CPI = UKR_CPI[UKR_CPI.duplicated()]
duplicate_UKR_CPI
```

Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item	Months Code	Months	Year Code	Year	Unit	Value	Flag	Note

Every records in the dataset are unique

Unique Values

In row index 0, there is a data description so we're going to just ignore that while looking at the unique values

```
UKR_CPI["Iso3"].unique()
```

```
array(['#country+code', 'UKR'], dtype=object)
```

```
np.sum(len(UKR_CPI["StartDate"].unique()))
```

```
271
```

```
np.sum(len(UKR_CPI["EndDate"].unique()))
```

```
271
```

```
UKR_CPI["Area Code"].unique()
```

```
array([ nan, 230.])
```

Total Unique Dates:

12 Months * 22 Years * 3 Item + 6 months of 2022

```
UKR_CPI["Area Code (M49)"].unique()
```

```
array([nan, "'804"], dtype=object)
```

```
UKR_CPI["Area"].unique()
```

```
array(['#country+name', 'Ukraine'], dtype=object)
```

```
UKR_CPI["Item Code"].unique()
```

```
array(['#indicator+code', '23013', '23012', '23014'], dtype=object)
```

```
UKR_CPI["Item"].unique()
```

```
array(['#indicator+name', 'Consumer Prices, Food Indices (2015 = 100)',
       'Consumer Prices, General Indices (2015 = 100)',
       'Food price inflation'], dtype=object)
```

```
UKR_CPI["Months Code"].unique()
```

```
array([  nan, 7001., 7002., 7003., 7004., 7005., 7006., 7007., 7008.,
       7009., 7010., 7011., 7012.])
```

```
UKR_CPI["Months"].unique()
```

```
array([nan, 'January', 'February', 'March', 'April', 'May', 'June',
       'July', 'August', 'September', 'October', 'November', 'December'],
      dtype=object)
```

```
UKR_CPI["Year Code"].unique()
```

```
array([  nan, 2000., 2001., 2002., 2003., 2004., 2005., 2006., 2007.,
       2008., 2009., 2010., 2011., 2012., 2013., 2014., 2015., 2016.,
       2017., 2018., 2019., 2020., 2021., 2022.])
```

```
UKR_CPI["Year"].unique()
```

```
array(['#date+year', '2000', '2001', '2002', '2003', '2004', '2005',
       '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021',
       '2022'], dtype=object)
```

```
UKR_CPI["Unit"].unique()
```

```
array(['#indicator+type', nan, '%'], dtype=object)
```

```
np.sum(len(UKR_CPI["Value"].unique()))
```

```
714
```

```
UKR_CPI["Flag"].unique()
```

```
array([nan, 'I', 'X', 'A', 'E'], dtype=object)
```

```
UKR_CPI["Note"].unique()
```

```
array([nan, 'base year is 2015'], dtype=object)
```

Dropping Row(s) that are/is not needed

```
UKR_CPI = UKR_CPI.drop(labels = 0, axis = 0)
UKR_CPI.head(6)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Item Code	Item	Months Code	Months	Year Code	Year	Unit	Value	Flag	Note
1	UKR	2000-01-01	2000-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2000.0	2000	NaN	19.586001	I	base year is 2015
2	UKR	2001-01-01	2001-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2001.0	2001	NaN	23.565986	I	base year is 2015
3	UKR	2002-01-01	2002-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2002.0	2002	NaN	24.766656	I	base year is 2015
4	UKR	2003-01-01	2003-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2003.0	2003	NaN	24.766656	I	base year is 2015
5	UKR	2004-01-01	2004-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2004.0	2004	NaN	26.644501	I	base year is 2015
6	UKR	2005-01-01	2005-01-31	230.0	'804	Ukraine	23013	Consumer Prices, Food Indices (2015 = 100)	7001.0	January	2005.0	2005	NaN	29.643633	X	base year is 2015

The first row (index 0) only contain description of each column.

By doing so, we removed a unique value or even a missing value from each column (Improving data quality)

Changing Data Type

```
UKR_CPI["Value"] = UKR_CPI["Value"].astype('float64')
```

```
UKR_CPI.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 798 entries, 1 to 798
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Iso3             798 non-null    object 
 1   StartDate        798 non-null    object 
 2   EndDate          798 non-null    object 
 3   Area Code        798 non-null    float64
 4   Area Code (M49)  798 non-null    object 
 5   Area             798 non-null    object 
 6   Item Code        798 non-null    object 
 7   Item             798 non-null    object 
 8   Months Code      798 non-null    float64
 9   Months           798 non-null    object 
 10  Year Code        798 non-null    float64
 11  Year             798 non-null    object 
 12  Unit             258 non-null    object 
 13  Value            798 non-null    float64
 14  Flag             798 non-null    object 
 15  Note             540 non-null    object 
dtypes: float64(4), object(12)
memory usage: 106.0+ KB
```

We need the variable "Value" in the form of float and not object
Therefore we can proceed with our analysis

Dropping Variables (Column)

```
UKR_CPI_Chosen = pd.DataFrame(UKR_CPI[['Year', 'Value', 'Item']])
UKR_CPI_Chosen
```

	Year	Value	Item
1	2000	19.586001	Consumer Prices, Food Indices (2015 = 100)
2	2001	23.565986	Consumer Prices, Food Indices (2015 = 100)
3	2002	24.766656	Consumer Prices, Food Indices (2015 = 100)
4	2003	24.766656	Consumer Prices, Food Indices (2015 = 100)
5	2004	26.644501	Consumer Prices, Food Indices (2015 = 100)
...	...	...	...
794	2017	17.722222	Food price inflation
795	2018	7.786692	Food price inflation
796	2019	4.772329	Food price inflation
797	2020	4.931049	Food price inflation
798	2021	12.743927	Food price inflation

798 rows × 3 columns

Rather than dropping the value one by one, we decided to make another data frame consisting the variables we needed.

Reasoning for variables removal:
- Iso3: This dataset focuses only in Ukraine so there is no need for Ukraine's Iso3
- StartDate, EndDate, Months Code, Months: We will focuses to the growth each year in Ukraine's CPI and not monthly.
- Area Code, Area Code(M49), Area: All of the data is located in Ukraine so these variables has the same values in each rows
- Unit, Flag, and Note: only helps us to understand the dataset while reading it.
- Item code and Year Code: It will be used if we need the data type factor of "Item" and "Year"

Reasoning for chosen variables:
- Year: We're going to group our data by years, by searching its' mean.
- Item: We need to know what is being represented by the percentage we got.
- Value: Contains the CPI data we need

Reasoning for adding new variables:
- Mean_Value: To give the ease in processing the data, by displaying the mean CPI in that year, in each categories

Adding New Variable(s)

```
UKR_CPI_Chosen['Mean_Value'] = UKR_CPI_Chosen.groupby(['Year', 'Item'], sort=False)['Value'].transform('mean')
```

In this step we are counting the mean of all records that has the same item category and year.

UKR_CPI_Chosen

	Year	Value	Item	Mean_Value
1	2000	19.586001	Consumer Prices, Food Indices (2015 = 100)	21.518124
2	2001	23.565986	Consumer Prices, Food Indices (2015 = 100)	23.946350
3	2002	24.766656	Consumer Prices, Food Indices (2015 = 100)	24.076159
4	2003	24.766656	Consumer Prices, Food Indices (2015 = 100)	25.303444
5	2004	26.644501	Consumer Prices, Food Indices (2015 = 100)	27.500946
...	...	...	...	...
794	2017	17.722222	Food price inflation	12.906168
795	2018	7.786692	Food price inflation	11.344656
796	2019	4.772329	Food price inflation	8.017908
797	2020	4.931049	Food price inflation	2.746711
798	2021	12.743927	Food price inflation	10.784551

798 rows × 4 columns

The variable **"Mean_Value"** contains the mean value of the year's CPI based on respective values in "Item".

Reasoning for adding new variables:
- Mean_Value: To give the ease in processing the data, by displaying the mean CPI in that year, in each categories

Dropping Variables (Column) [2]

This step is necessary to increase the computing speed

```
UKR_CPI_Chosen.drop(columns = ['Value'], inplace = True)
UKR_CPI_Chosen
```

	Year	Item	Mean_Value
1	2000	Consumer Prices, Food Indices (2015 = 100)	21.518124
2	2001	Consumer Prices, Food Indices (2015 = 100)	23.946350
3	2002	Consumer Prices, Food Indices (2015 = 100)	24.076159
4	2003	Consumer Prices, Food Indices (2015 = 100)	25.303444
5	2004	Consumer Prices, Food Indices (2015 = 100)	27.500946
...	...	...	...
794	2017	Food price inflation	12.906168
795	2018	Food price inflation	11.344656
796	2019	Food price inflation	8.017908
797	2020	Food price inflation	2.746711
798	2021	Food price inflation	10.784551

798 rows × 3 columns

Removing Duplicated Rows

As we have already remove the variable "Value" which contains the CPI of each month respectively and we are not using the value based on months rather based on years (using the mean), meaning that the rows that are in the same year and same item category will have duplicated attributes. Therefore, to clean the dataset, we would need to remove the duplicated rows

```
UKR_CPI_Final = UKR_CPI_Chosen.drop_duplicates()
UKR_CPI_Final
```

	Year	Item	Mean_Value
1	2000	Consumer Prices, Food Indices (2015 = 100)	21.518124
2	2001	Consumer Prices, Food Indices (2015 = 100)	23.946350
3	2002	Consumer Prices, Food Indices (2015 = 100)	24.076159
4	2003	Consumer Prices, Food Indices (2015 = 100)	25.303444
5	2004	Consumer Prices, Food Indices (2015 = 100)	27.500946
...	...	...	...
558	2018	Food price inflation	11.344656
559	2019	Food price inflation	8.017908
560	2020	Food price inflation	2.746711
561	2021	Food price inflation	10.784551
562	2022	Food price inflation	20.144415

68 rows × 3 columns

```
UKR_CPI_Final = UKR_CPI_Final.pivot(index= 'Year', columns='Item', values='Mean_Value')
UKR_CPI_Final
```

Item	Consumer Prices, Food Indices (2015 = 100)	Consumer Prices, General Indices (2015 = 100)	Food price inflation
Year			
2000	21.518124	19.420949	NaN
2001	23.946350	21.740145	11.501460
2002	24.076159	21.902489	0.546283
2003	25.303444	23.048172	5.152437
2004	27.500946	25.135448	8.682911
2005	30.977389	28.502921	12.671689
2006	32.622527	31.081867	5.354994
2007	35.767493	35.070885	9.688284
2008	48.534183	43.920938	36.072180
2009	54.331091	50.892441	12.178234
2010	60.247362	55.660709	10.899336
2011	64.103318	60.076458	6.459367
2012	62.738424	60.428976	-2.109844
2013	61.389099	60.275909	-2.145373
2014	68.789628	67.621731	12.100088
2015	100.384592	100.564985	45.839896
2016	109.378362	114.559015	9.872513
2017	123.468429	131.094884	12.906168
2018	137.231545	145.441432	11.344656
2019	148.207785	156.921453	8.017908
2020	152.281708	161.221243	2.746711
2021	168.717525	176.309933	10.784551
2022	199.035820	199.286045	20.144415

We pivoted the table, making the categories in "Item" the variables now.

Final Review of The Processed Data Frame

Final Attributes

```
UKR_CPI_Final.columns

Index(['Consumer Prices, Food Indices (2015 = 100)',
       'Consumer Prices, General Indices (2015 = 100)',
       'Food price inflation'],
      dtype='object', name='Item')
```

Fixing the display of the table, where year is now also a variable

```
UKR_CPI_Final = UKR_CPI_Final.reset_index()
UKR_CPI_Final
```

Item	Year	Consumer Prices, Food Indices (2015 = 100)	Consumer Prices, General Indices (2015 = 100)	Food price inflation
0	2000	21.518124	19.420949	NaN
1	2001	23.946350	21.740145	11.501460
2	2002	24.076159	21.902489	0.546283
3	2003	25.303444	23.048172	5.152437
4	2004	27.500946	25.135448	8.682911
5	2005	30.977389	28.502921	12.671689
6	2006	32.622527	31.081867	5.354994
7	2007	35.767493	35.070885	9.688284
8	2008	48.534183	43.920938	36.072180
9	2009	54.331091	50.892441	12.178234
10	2010	60.247362	55.660709	10.899336
11	2011	64.103318	60.076458	6.459367
12	2012	62.738424	60.428976	-2.109844
13	2013	61.389099	60.275909	-2.145373
14	2014	68.789628	67.621731	12.100088
15	2015	100.384592	100.564985	45.839896
16	2016	109.378362	114.559015	9.872513
17	2017	123.468429	131.094884	12.906168
18	2018	137.231545	145.441432	11.344656
19	2019	148.207785	156.921453	8.017908
20	2020	152.281708	161.221243	2.746711
21	2021	168.717525	176.309933	10.784551
22	2022	199.035820	199.286045	20.144415

Handling Missing Value

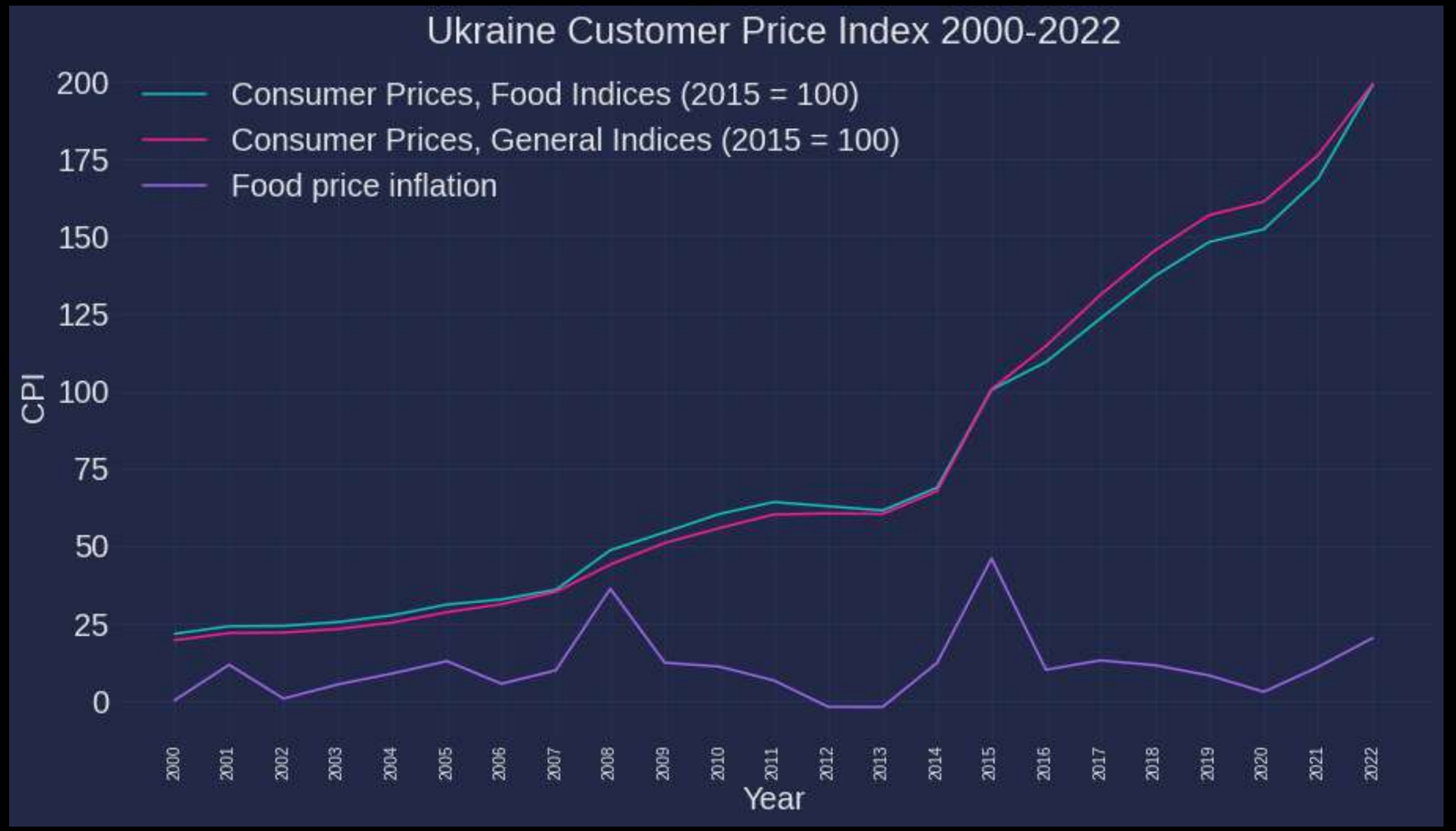
```
UKR_CPI_Final['Food price inflation'].replace([np.nan], 0, inplace=True)
UKR_CPI_Final
```

Item	Year	Consumer Prices, Food Indices (2015 = 100)	Consumer Prices, General Indices (2015 = 100)	Food price inflation
0	2000	21.518124	19.420949	0.000000
1	2001	23.946350	21.740145	11.501460
2	2002	24.076159	21.902489	0.546283
3	2003	25.303444	23.048172	5.152437
4	2004	27.500946	25.135448	8.682911
5	2005	30.977389	28.502921	12.671689
6	2006	32.622527	31.081867	5.354994
7	2007	35.767493	35.070885	9.688284
8	2008	48.534183	43.920938	36.072180
9	2009	54.331091	50.892441	12.178234
10	2010	60.247362	55.660709	10.899336
11	2011	64.103318	60.076458	6.459367
12	2012	62.738424	60.428976	-2.109844
13	2013	61.389099	60.275909	-2.145373
14	2014	68.789628	67.621731	12.100088
15	2015	100.384592	100.564985	45.839896
16	2016	109.378362	114.559015	9.872513
17	2017	123.468429	131.094884	12.906168
18	2018	137.231545	145.441432	11.344656
19	2019	148.207785	156.921453	8.017908
20	2020	152.281708	161.221243	2.746711
21	2021	168.717525	176.309933	10.784551
22	2022	199.035820	199.286045	20.144415

Visualizing

```
plt.rcParams.update({'font.size': 20})
plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothie-dark.mplstyle')

x = UKR_CPI_Final["Year"]
y1 = UKR_CPI_Final["Consumer Prices, Food Indices (2015 = 100)"]
y2 = UKR_CPI_Final["Consumer Prices, General Indices (2015 = 100)"]
y3 = UKR_CPI_Final["Food price inflation"]
plt.plot(x, y1, label= "Consumer Prices, Food Indices (2015 = 100)")
plt.plot(x, y2, label= "Consumer Prices, General Indices (2015 = 100)")
plt.plot(x, y3, label= "Food price inflation")
plt.title("Ukraine Customer Price Index 2000-2022")
plt.xlabel("Year", fontsize=20)
plt.ylabel("CPI")
plt.gcf().set_size_inches(15, 8)
plt.xticks(size = 10, rotation = 90)
plt.legend()
plt.show()
```

UKRAINE's CUSTOMER PRICE INDEX



Ukraine's Customer Price Index **rose steeply** through out the 22.5 years

Ukraine's Inflation Rate

Inflation Rate Dataset

Load Data

```
UKR_INF = pd.read_csv("https://raw.githubusercontent.com/laurentiaalyssa/Data/main/INFLATION_UKR.csv", sep=';')
```

```
UKR_INF.head(5)
```

	Year	Inflation rate
0	2000	25.8
1	2001	6.1
2	2002	-0.6
3	2003	8.2
4	2004	12.3

```
UKR_INF.tail(5)
```

	Year	Inflation rate
18	2018	9.8
19	2019	4.1
20	2020	5.0
21	2021	10.0
22	2022	30.0

Data Description

```
UKR_INF.columns
```

```
Index(['Year', 'Inflation rate'], dtype='object')
```

Year: The year in which the data was recorded.
Inflation Rate: The value of inflation rate

Data Type, Missing and Unique Values

```
UKR_INF.shape
```

```
(23, 2)
```

Dimension of our dataset:
23 records and 2 variables

```
UKR_INF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Year            23 non-null     int64  
 1   Inflation rate  23 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 496.0 bytes
```

We can see that our data has no missing value

```
UKR_INF["Year"].unique()
```

```
array([2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
       2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021,
       2022])
```

```
UKR_INF["Inflation rate"].unique()
```

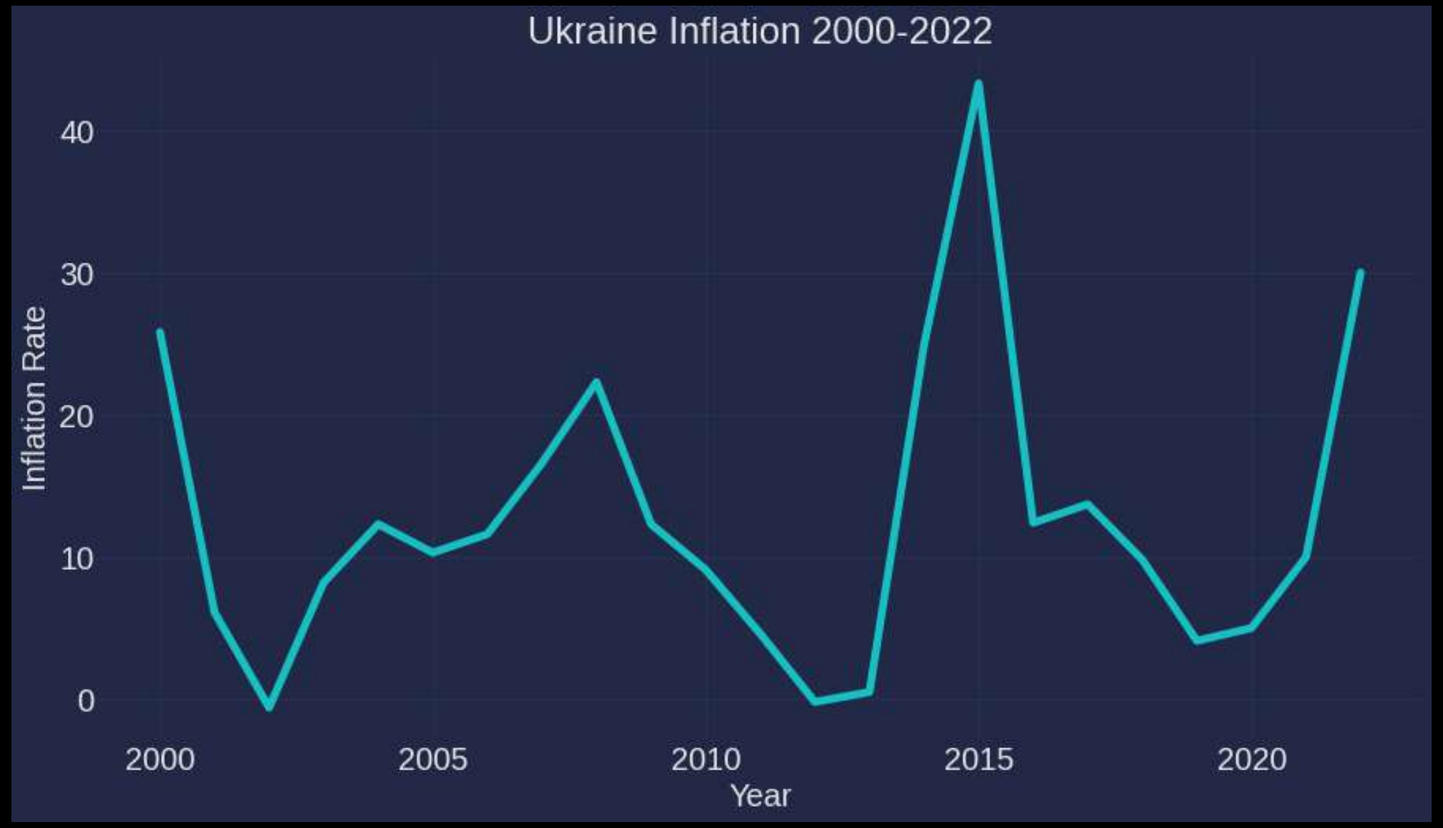
```
array([25.8,  6.1, -0.6,  8.2, 12.3, 10.3, 11.6, 16.6, 22.3,  9.1,  4.6,
       -0.2,  0.5, 24.9, 43.3, 12.4, 13.7,  9.8,  4.1,  5. , 10. , 30. ])
```

Since, it's a small dataset, it's obvious from this output, that all values in both variables are all unique

Visualization

```
plt.rcParams.update({'font.size': 20})
plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothie-dark.mplstyle')

plt.plot(UKR_INF['Year'], UKR_INF['Inflation rate'], linewidth=5)
plt.title("Ukraine Inflation 2000-2022")
plt.xlabel("Year", fontsize=20)
plt.ylabel("Inflation Rate")
plt.gcf().set_size_inches(15, 8)

plt.xticks(size = 20)
plt.yticks(size = 20)
plt.show()
```

UKRAINE's INFLATION RATE



Ukraine's inflation rate fluctuated throughout the 23-year period. Its' peak was back in 2015, but since 2020 the graph line **keeps increasing significantly**.

Increase Inflation Impact

Increase

Exchange Rate

tend to

Decrease

GDP, Import, Export

Ukraine Exchange Rate

Load Data

```
[ ]  UKR_ER = pd.read_csv( "https://raw.githubusercontent.com/laurentiaalyssa/Data/main/Exchange%20rates%20data%20for%20Ukraine.csv")
```

```
[ ]  UKR_ER.head(5)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Element Code	Element	ISO Currency Code	Currency	Year Code	Year	Months Code	Months	Unit	Value	Flag
0	#country+code	#date+start	#date+end	NaN	NaN	#country+name	NaN	NaN	NaN	NaN	NaN	#date+year	NaN	NaN	#indicator+type	#indicator+value+num	NaN
1	UKR	1996-02-01	1996-02-29	230.0	'804	Ukraine	LCU	Local currency units per USD	UAH	Hryvnia	1996.0	1996	7002.0	February	NaN	1.881034500000	X
2	UKR	1996-03-01	1996-03-31	230.0	'804	Ukraine	LCU	Local currency units per USD	UAH	Hryvnia	1996.0	1996	7003.0	March	NaN	1.891516100000	X
3	UKR	1996-04-01	1996-04-30	230.0	'804	Ukraine	LCU	Local currency units per USD	UAH	Hryvnia	1996.0	1996	7004.0	April	NaN	1.872666700000	X
4	UKR	1996-05-01	1996-05-31	230.0	'804	Ukraine	LCU	Local currency units per USD	UAH	Hryvnia	1996.0	1996	7005.0	May	NaN	1.841900000000	X

```
UKR_ER.tail(5)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Element Code	Element	ISO Currency Code	Currency	Year Code	Year	Months Code	Months	Unit	Value	Flag
374	UKR	2017-01-01	2017-12-31	230.0	'804	Ukraine	SLC	Standard local currency units per USD	UAH	Hryvnia	2017.0	2017	7021.0	Annual value	NaN	26.596606300000	X
375	UKR	2018-01-01	2018-12-31	230.0	'804	Ukraine	SLC	Standard local currency units per USD	UAH	Hryvnia	2018.0	2018	7021.0	Annual value	NaN	27.200492333300	X
376	UKR	2019-01-01	2019-12-31	230.0	'804	Ukraine	SLC	Standard local currency units per USD	UAH	Hryvnia	2019.0	2019	7021.0	Annual value	NaN	25.845589333300	X
377	UKR	2020-01-01	2020-12-31	230.0	'804	Ukraine	SLC	Standard local currency units per USD	UAH	Hryvnia	2020.0	2020	7021.0	Annual value	NaN	26.957524383300	X
378	UKR	2021-01-01	2021-12-31	230.0	'804	Ukraine	SLC	Standard local currency units per USD	UAH	Hryvnia	2021.0	2021	7021.0	Annual value	NaN	27.286189383300	X

import dataset

Data Description

- **StartDate : Date Start**
- **EndDate : Date End**
- **Area Code : Code for each area**
- **Area : Country Name**
- **Element Code : Element Code**
- **Element : Whether it is Local Currency Unit per USD or Standard Local Currency units per USD**
- **Currency : Currency name**
- **Year : Year**
- **Months : Months**
- **Unit : Indicator type**
- **Value : Indicator value num**

Missing Value & Data Type

```
[ ]  UKR_ER.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 379 entries, 0 to 378
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Iso3                379 non-null    object 
 1   StartDate           379 non-null    object 
 2   EndDate             379 non-null    object 
 3   Area Code           378 non-null    float64
 4   Area Code (M49)     378 non-null    object 
 5   Area                379 non-null    object 
 6   Element Code        378 non-null    object 
 7   Element             378 non-null    object 
 8   ISO Currency Code   378 non-null    object 
 9   Currency            378 non-null    object 
 10  Year Code           378 non-null    float64
 11  Year                379 non-null    object 
 12  Months Code         378 non-null    float64
 13  Months              378 non-null    object 
 14  Unit                1 non-null      object 
 15  Value               379 non-null    object 
 16  Flag                378 non-null    object 
dtypes: float64(3), object(14)
memory usage: 50.5+ KB
```

From the data above we can see:
1. The first row contains the data description, we don't want that so in the next step we will discard the first line.
2. The missing value in the Area Code, Area Code(M49), Element Code, Element, ISO Currency Code, Months Code, Months, and Flag columns is caused by the presence of the first row, so that after the first row is deleted, the missing value will be disappear.
3. In the unit column there is only 1 data that is filled in, the first row, so we will discard the unit column in the next step.
4. There are two types of data: string and float
5. Some attributes do not have the appropriate data type, for example start date and end date are still in the form of strings, in the next stage we will change the data type to date data type, as well as other attributes that do not have the appropriate data type.

Checking Duplicate

```
duplicate_ukrER = UKR_ER[UKR_ER.duplicated()]
duplicate_ukrER
```

Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Element Code	Element	ISO Currency Code	Currency	Year Code	Year	Months Code	Months	Unit	Value	Flag

There is no duplication in Ukraine Exchange Rate dataset

Drop First Row

```
#drop first row
UKR_ER = UKR_ER.drop(labels = 0, axis = 0)
UKR_ER.head(3)
```

	Iso3	StartDate	EndDate	Area Code	Area Code (M49)	Area	Element Code	Element	ISO Currency Code	Currency	Year Code	Year	Months Code	Months	Unit	Value	Flag
1	UKR	1996-02-01	1996-02-29	230.0	'804	Ukraine	LCU	Local currency units per USD	UAH	Hryvnia	1996.0	1996	7002.0	February	NaN	1.881034500000	X
2	UKR	1996-03-01	1996-03-31	230.0	'804	Ukraine	LCU	Local currency units per USD	UAH	Hryvnia	1996.0	1996	7003.0	March	NaN	1.891516100000	X
3	UKR	1996-04-01	1996-04-30	230.0	'804	Ukraine	LCU	Local currency units per USD	UAH	Hryvnia	1996.0	1996	7004.0	April	NaN	1.872666700000	X

Drop row(s) that contains NaN value.

Selecting Variable

```
[ ]  #Select the column to be used in the analysis
     UKR_ER = UKR_ER.drop(columns = ['Iso3', 'Area Code', 'Area Code (M49)', 'Unit', 'Months Code', 'Year Code', 'ISO Currency Code' ])
     UKR_ER.head(5)
```

	StartDate	EndDate	Area	Element Code	Element	Currency	Year	Months	Value	Flag
1	1996-02-01	1996-02-29	Ukraine	LCU	Local currency units per USD	Hryvnia	1996	February	1.881034500000	X
2	1996-03-01	1996-03-31	Ukraine	LCU	Local currency units per USD	Hryvnia	1996	March	1.891516100000	X
3	1996-04-01	1996-04-30	Ukraine	LCU	Local currency units per USD	Hryvnia	1996	April	1.872666700000	X
4	1996-05-01	1996-05-31	Ukraine	LCU	Local currency units per USD	Hryvnia	1996	May	1.841900000000	X
5	1996-06-01	1996-06-30	Ukraine	LCU	Local currency units per USD	Hryvnia	1996	June	1.819766700000	X

Select the column to be used in the analysis

Fixing Data Type

```
# Fixed data type
UKR_ER['Value'] = UKR_ER['Value'].astype(float)
UKR_ER['StartDate'] = pd.to_datetime(UKR_ER['StartDate'])
UKR_ER['EndDate'] = pd.to_datetime(UKR_ER['EndDate'])
UKR_ER['Year'] = UKR_ER['Year'].astype(int)
UKR_ER.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 378 entries, 1 to 378
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   StartDate     378 non-null    datetime64[ns]
 1   EndDate       378 non-null    datetime64[ns]
 2   Area          378 non-null    object        
 3   Element Code  378 non-null    object        
 4   Element       378 non-null    object        
 5   Currency      378 non-null    object        
 6   Year          378 non-null    int64         
 7   Months        378 non-null    object        
 8   Value         378 non-null    float64       
 9   Flag          378 non-null    object        
dtypes: datetime64[ns](2), float64(1), int64(1), object(6)
memory usage: 32.5+ KB
```

create appropriate data types

Filter Data

```
#Filter data from 2000
UKR_ER = UKR_ER.loc[(UKR_ER['Year'] >= 2000)]
UKR_ER
```

	StartDate	EndDate	Area	Element Code	Element	Currency	Year	Months	Value	Flag
51	2000-01-01	2000-12-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2000	Annual value	5.440233	X
52	2000-01-01	2000-01-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2000	January	5.381100	X
53	2000-02-01	2000-02-29	Ukraine	LCU	Local currency units per USD	Hryvnia	2000	February	5.543100	X
54	2000-03-01	2000-03-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2000	March	5.468000	X
55	2000-04-01	2000-04-30	Ukraine	LCU	Local currency units per USD	Hryvnia	2000	April	5.422800	X

We will choose 2000 as the starting year for all dataset.

Select data

```
#Select only annual value LCU (Local Currency Units)

UKR_ER_LCU = UKR_ER.loc[(UKR_ER['Months'] == 'Annual value') & (UKR_ER['Element Code'] == 'LCU')  ]
UKR_ER_LCU.head()
```

	StartDate	EndDate	Area	Element Code	Element	Currency	Year	Months	Value	Flag
51	2000-01-01	2000-12-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2000	Annual value	5.440233	X
64	2001-01-01	2001-12-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2001	Annual value	5.372158	X
77	2002-01-01	2002-12-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2002	Annual value	5.326625	X
90	2003-01-01	2003-12-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2003	Annual value	5.332688	X
103	2004-01-01	2004-12-31	Ukraine	LCU	Local currency units per USD	Hryvnia	2004	Annual value	5.319181	X

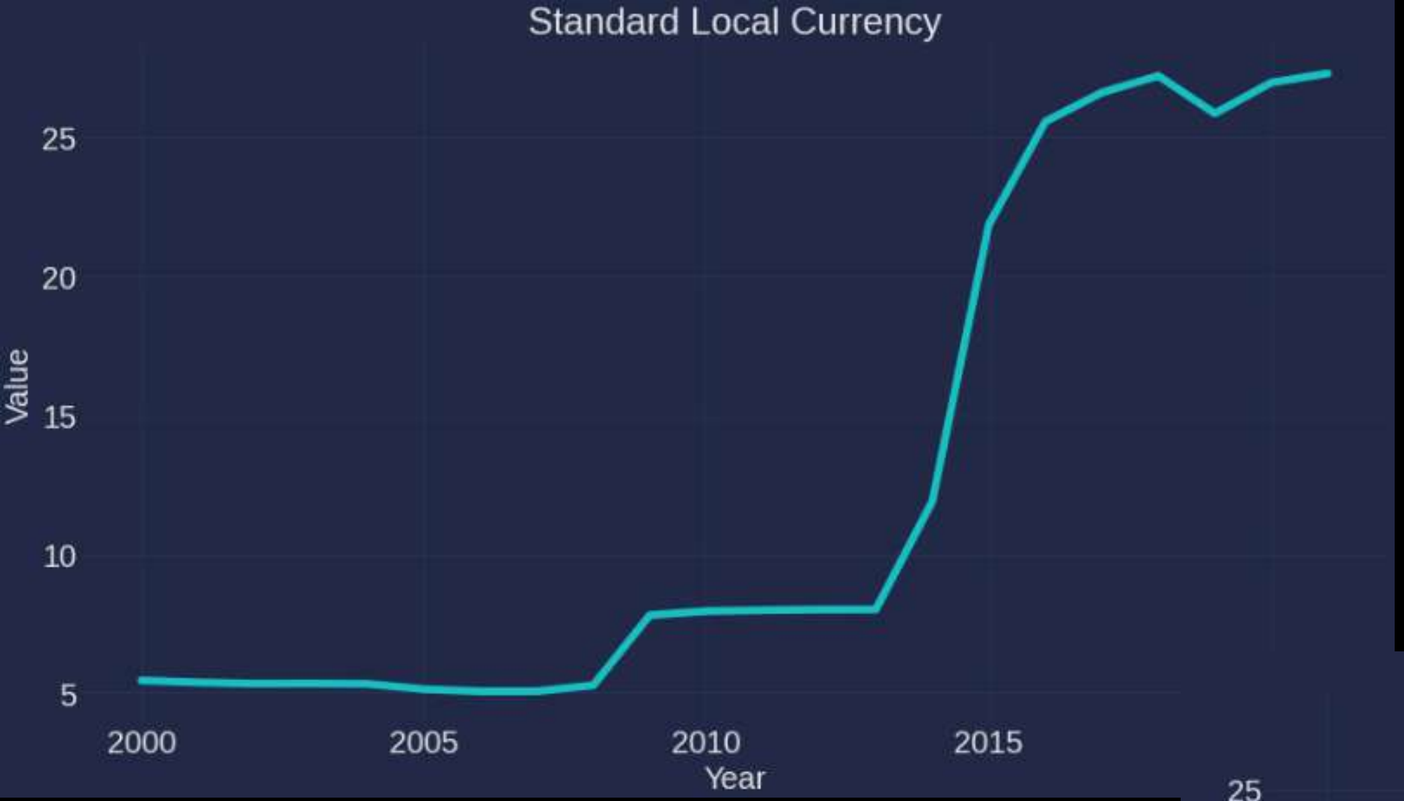
Select only annual value LCU (Local Currency Units)

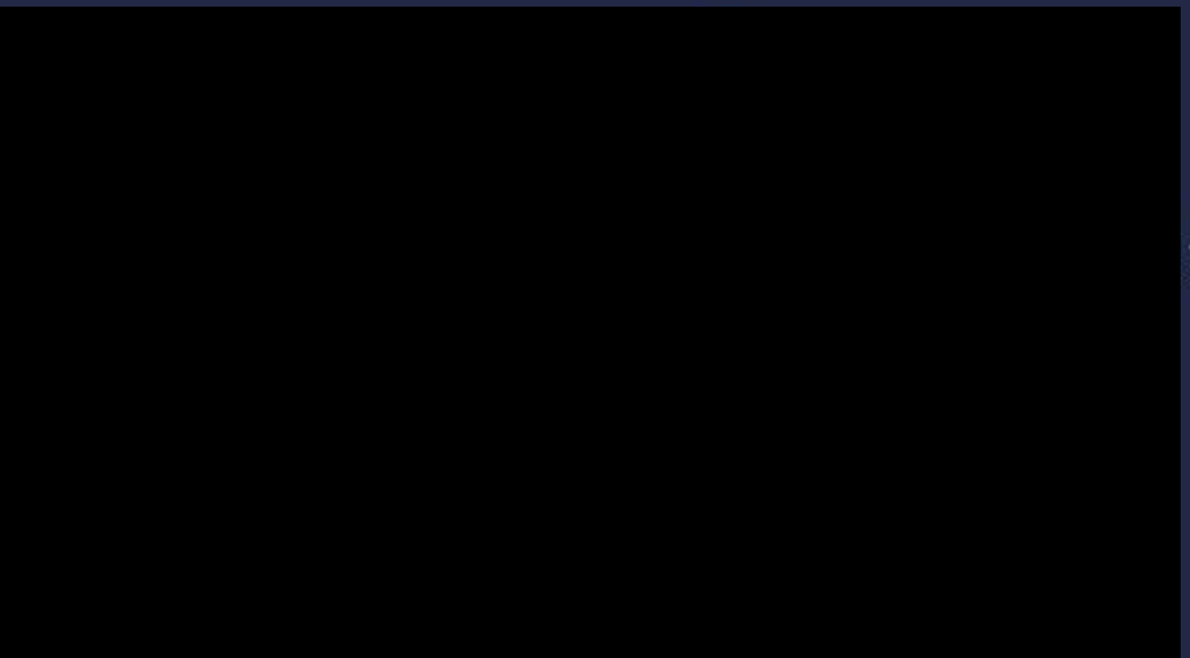
Select data

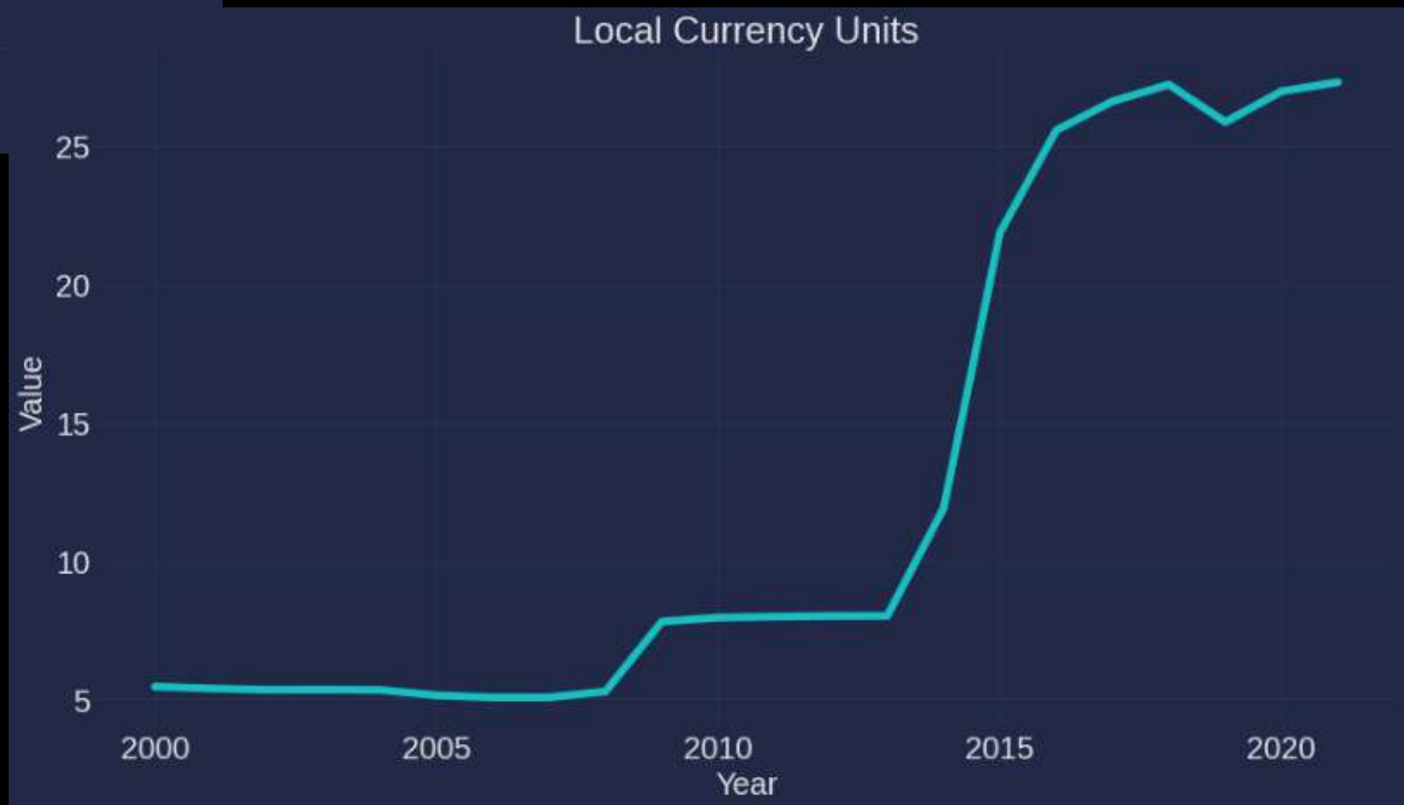
```
#Select only annual value SLC (Standard Local Currency)
UKR_ER_SLC = UKR_ER.loc[(UKR_ER['Months'] == 'Annual value') & (UKR_ER['Element Code'] == 'SLC')  ]
UKR_ER_SLC.head()
```

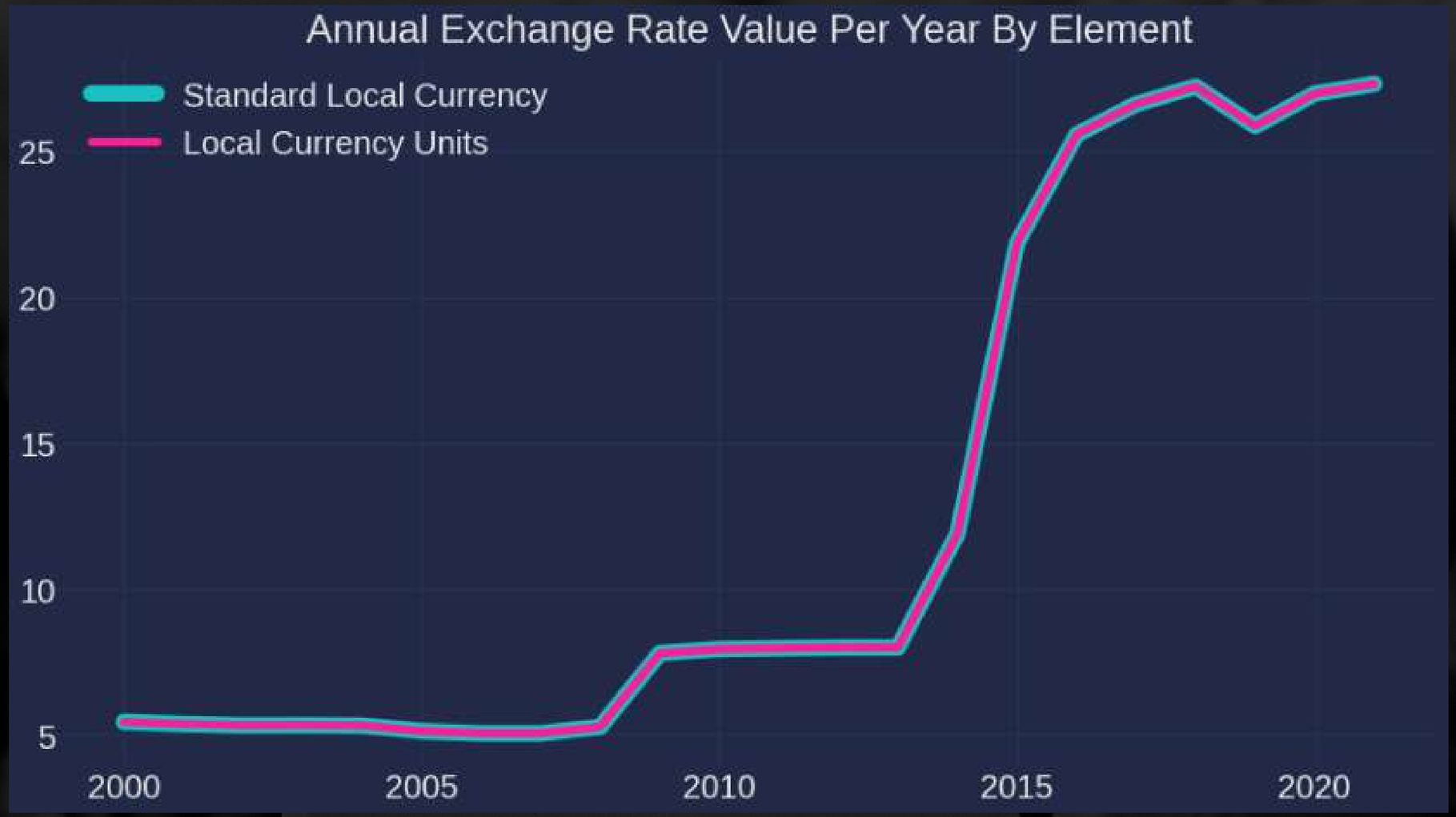
	StartDate	EndDate	Area	Element Code	Element	Currency	Year	Months	Value	Flag
357	2000-01-01	2000-12-31	Ukraine	SLC	Standard local currency units per USD	Hryvnia	2000	Annual value	5.440233	X
358	2001-01-01	2001-12-31	Ukraine	SLC	Standard local currency units per USD	Hryvnia	2001	Annual value	5.372158	X
359	2002-01-01	2002-12-31	Ukraine	SLC	Standard local currency units per USD	Hryvnia	2002	Annual value	5.326625	X
360	2003-01-01	2003-12-31	Ukraine	SLC	Standard local currency units per USD	Hryvnia	2003	Annual value	5.332688	X
361	2004-01-01	2004-12-31	Ukraine	SLC	Standard local currency units per USD	Hryvnia	2004	Annual value	5.319181	X

Select only annual value SLC (Standard Local Currency)


Standard Local Currency
25
20
15
10
5
Value
2000
2005
2010
2015
Year




Local Currency Units
25
20
15
10
5
Value
2000
2005
2010
2015
2020
Year



Ukraine experienced a **rapid increase** in the **exchange rate** from 2000 to 2020.

Ukraine's GDP Growth

Ukraine's GDP Growth Dataset

Load Data

```
GDP = pd.read_csv("https://raw.githubusercontent.com/laurentiaalyssa/Data/main/ukraine_gdp.csv", sep=';')
```

```
GDP.head()
```

	Year	GDP
0	2000	5.09
1	2001	8.08
2	2002	5.03
3	2003	9.05
4	2004	11.08

```
GDP.tail()
```

	Year	GDP
18	2018	3.05
19	2019	3.02
20	2020	-3.80
21	2021	3.04
22	2022	-35.00

Missing Value & Data Type

```
GDP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Year    23 non-null     int64  
 1   GDP     23 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 496.0 bytes
```

From the output above it can be seen that:

1. There is no missing value

2. There are 2 kinds of data types: integer which is owned by the Year column and float which is owned by GDP column

3. Each column already has a good name and data type

Checking Duplicate

```
duplicate_GDP = GDP[GDP.duplicated()]
duplicate_GDP
```

Year	GDP

Every records in the dataset are unique.

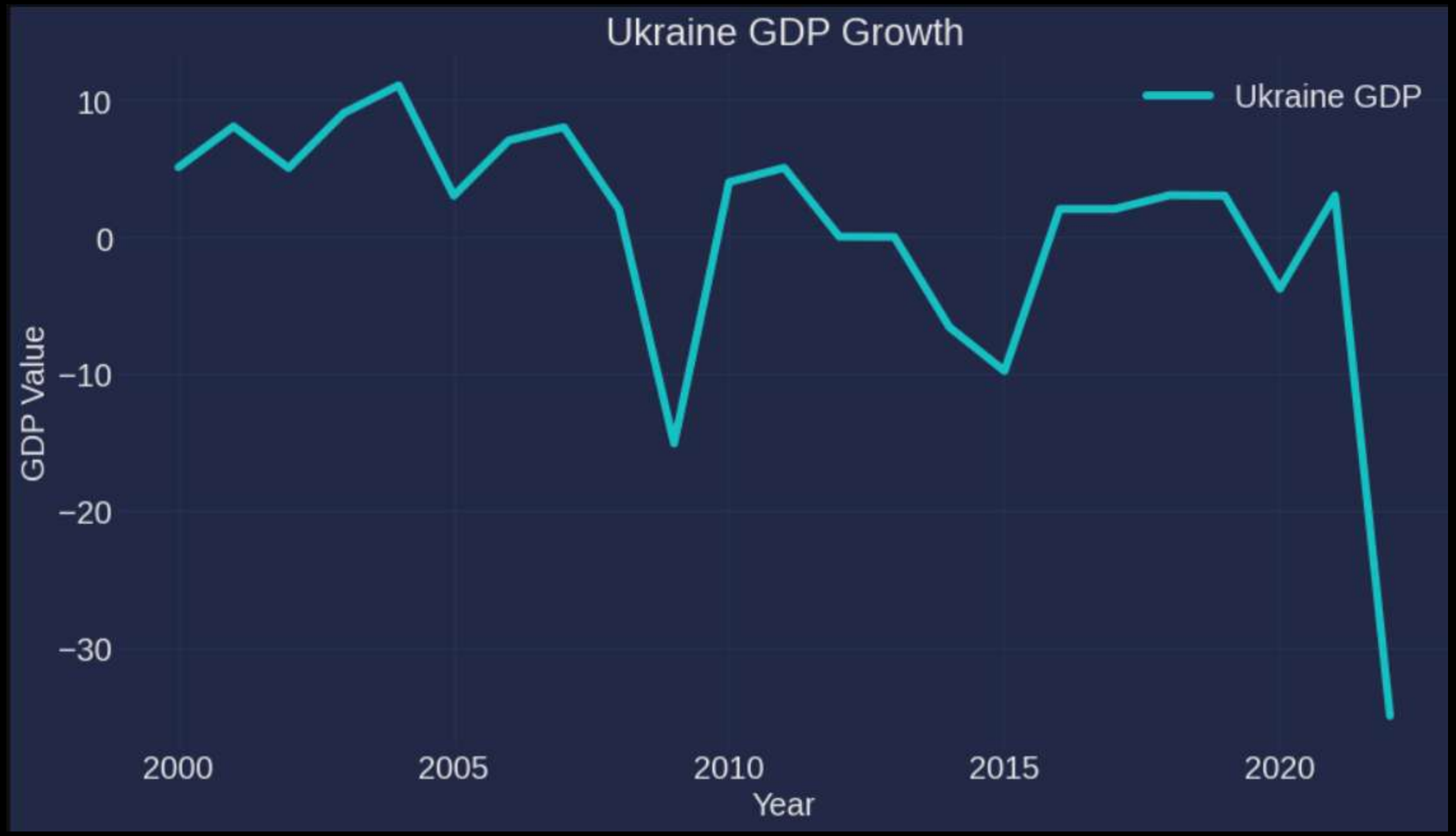
There is no duplication in Ukraine's GDP Growth dataset

Visualization

```
plt.plot(GDP["Year"],GDP["GDP"], label= "Ukraine GDP", linewidth=5)

plt.rcParams.update({'font.size': 20})
plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothie-dark.mplstyle')

plt.title("Ukraine GDP Growth")
plt.xlabel("Year", fontsize=20)
plt.ylabel("GDP Value")
plt.gcf().set_size_inches(15, 8)
plt.xticks(size = 20)
plt.yticks(size = 20)

plt.legend()
plt.show()
```

UKRAINE's GDP Growth

Since **2020** ukraine has experienced a **decline** in **GDP**



source: www.imf.org

Ukraine's GDP is **expected to drop** by **35%** in 2022, according IMF.

source: www.imf.org

EXPORT TRADE DATASET

Load Data

This export dataset includes the value of all nations' exports from 1988 through 2020.

```
1  data_export = pd.read_csv("https://raw.githubusercontent.com/laurentiaalyssa/Data/main/Exports%20By%20Country%20Product%20to%20World%20in%20US%24%20Thousand%201988-2020.csv", sep=";")
2  data_export.head(6)
```

	Reporter Name	Partner Name	Trade Flow	Product Group	Indicator	1988	1989	1990	1991	1992	...	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
0	Afghanistan	World	Export	All Products	Export (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	375850,935	428902,71	514972,983	570534,007	571404,967	596455,337	831926,7718	884504,4964	870488,5125	NaN
1	Albania	World	Export	All Products	Export (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	1948207,305	1967918,947	2331521,928	2430723,644	1929657,032	1962117,416	2302318,252	2875859,779	2719428,708	2415632,335
2	Algeria	World	Export	All Products	Export (US$ Thousand)	NaN	NaN	NaN	NaN	11136801,79	...	73436306,09	71865748,99	65998137,6	60387689,3	34795951,49	29992101,47	35191117,42	NaN	NaN	NaN
3	Andorra	World	Export	All Products	Export (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	114022,713	105949,644	98978,051	95157,70705	89501,93495	96912,95867	117965,6118	129448,151	NaN	NaN
4	Angola	World	Export	All Products	Export (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	66427390,22	70863076,41	67712526,55	58672369,19	33924937,48	28057499,52	34904881,11	42096736,31	34819240,57	NaN
5	Anguila	World	Export	All Products	Export (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Selecting Ukraine Export Data

Take just the rows with export values from Ukraine from 2000 to 2020.

```
data_export = data_export.drop(['Trade Flow', 'Partner Name', 'Indicator','Product Group','1988','1989','1990','1991','1992','1993','1994','1995','1996','1997','1998','1999'], axis=1)
data_export2 = data_export.copy()

data_export2 = data_export2[data_export2['Reporter Name'] == 'Ukraine']
data_export2.head()
```

	Reporter Name	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
193	Ukraine	14572550,14	16264734,34	17927432,23	23066845,56	32666131,94	34227973,96	38367609,3	49294389,7	66952306,46	...	68393034,21	68694495,45	63320468,79	53913302,43	38127039,56	36361032,46	43428391,06	47334679,85	50054402,43	49230799,58

Set x and y variable before doing visualization

```
arr = np.array([data_export2])
arr
index= [0]
new_arr = np.delete(arr, index)
new_arr
```

```
array(['14572550,14', '16264734,34', '17927432,23', '23066845,56',
       '32666131,94', '34227973,96', '38367609,3', '49294389,7',
       '66952306,46', '39695647,59', '51430285,58', '68393034,21',
       '68694495,45', '63320468,79', '53913302,43', '38127039,56',
       '36361032,46', '43428391,06', '47334679,85', '50054402,43',
       '49230799,58'], dtype=object)
```

- X variable (value)： Array of export value from 2000 to 2020
- Y variable (year_arr)： Array of year from 2000-2020

```
value = np.array([14572550.14, 16264734.34, 17927432.23, 23066845.56, 32666131.94,
                  34227973.96, 38367609.3, 49294389.7, 66952306.46, 39695647.59,
                  51430285.58, 68393034.21, 68694495.45, 63320468.79, 53913302.43,
                  38127039.56, 36361032.46, 43428391.06, 47334679.85, 50054402.43,
                  49230799.58])
value
```

```
array([14572550.14, 16264734.34, 17927432.23, 23066845.56, 32666131.94,
       34227973.96, 38367609.3 , 49294389.7 , 66952306.46, 39695647.59,
       51430285.58, 68393034.21, 68694495.45, 63320468.79, 53913302.43,
       38127039.56, 36361032.46, 43428391.06, 47334679.85, 50054402.43,
       49230799.58])
```

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

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

Visualization

```
plt.plot(year_arr,value,'-o')
plt.rcParams.update({'font.size': 20})
plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothie-dark.mplstyle')
plt.gcf().set_size_inches(15, 8)
plt.xlabel("Year")
plt.ylabel("Value")
plt.xticks(np.arange(2000,2021,1.0), rotation = 90)
plt.title("Export Trade in Ukraine")
plt.figure(figsize = (15,8))
plt.show()
```

IMPORT TRADE DATASET

Load Data

This import dataset includes the value of all nations' exports from 1988 through 2020.

```
1  data_import = pd.read_csv("https://raw.githubusercontent.com/laurentiaalyssa/Data/main/Imports%20By%20Country%20Product%20from%20World%20in%20US%24%20Thousand%201988-2020.csv", sep=";")
2  data_import.head(6)
```

	Reporter Name	Partner Name	Trade Flow	Product Group	Indicator	1988	1989	1990	1991	1992	...	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
0	Afghanistan	World	Import	All Products	Import (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	6390310,947	6204984,101	8554413,749	7697178,17	7722865,049	6534140,413	7792600,112	7406590,325	8568013,877	NaN
1	Albania	World	Import	All Products	Import (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	5395853,069	4879829,648	4880593,254	5229972,238	4320219,367	4669289,913	5282117,403	5941286,949	5908003,38	5410943,447
2	Algeria	World	Import	All Products	Import (US$ Thousand)	NaN	NaN	NaN	NaN	8647816,192	...	47219730,32	50369390,59	54909973,27	58618080,19	51803071,09	47090683,59	46053015,99	NaN	NaN	NaN
3	Andorra	World	Import	All Products	Import (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	1617690,977	1417733,795	1487194,105	1556048,524	1293752,873	1354095,937	1478292,965	1609394,249	NaN	NaN
4	Angola	World	Import	All Products	Import (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	20790996,04	28722984,06	26756062,99	28753499,33	21549256,53	14347710,5	15462313,48	16036479,58	14160145,46	NaN
5	Anguila	World	Import	All Products	Import (US$ Thousand)	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Selecting Ukraine Import Data

Take just the rows with import values from Ukraine from 2000 to 2020.

```
data_import = data_import.drop(['Trade Flow', 'Partner Name', 'Indicator','Product Group','1988','1989','1990','1991','1992','1993','1994','1995','1996','1997','1998','1999'], axis=1)
data_import2 = data_import.copy()

data_import2 = data_import2[data_import2['Reporter Name'] == 'Ukraine']
data_import2.head()
```

	Reporter Name	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
194	Ukraine	13956027,19	15775092,66	16975884,13	23020437,66	28996784,33	36121997,44	45021620,6	60600580,94	85448380,85	...	82607537,26	84656666,98	76986012,5	54381409,09	37516153,22	39249626,35	49439155,77	57187093,43	60799689,21	53674668,69

Set x and y variable before doing visualization

```
arr = np.array([data_import2])
arr
index= [0]
new_arr = np.delete(arr, index)
new_arr
```

```
array(['13956027,19', '15775092,66', '16975884,13', '23020437,66',
       '28996784,33', '36121997,44', '45021620,6', '60600580,94',
       '85448380,85', '45412943,89', '60737134,59', '82607537,26',
       '84656666,98', '76986012,5', '54381409,09', '37516153,22',
       '39249626,35', '49439155,77', '57187093,43', '60799689,21',
       '53674668,69'], dtype=object)
```

- X variable (import_value) : Array of export value from 2000 to 2020
- Y variable (import_year_arr) : Array of year from 2000-2020

```
import_value = np.array([13956027.19, 15775092.66, 16975884.13, 23020437.66, 28996784.33,
                         36121997.44, 45021620.6, 60600580.94, 85448380.85, 45412943.89,
                         60737134.59, 82607537.26, 84656666.98, 76986012.5, 54381409.09,
                         37516153.22, 39249626.35, 49439155.77, 57187093.43, 60799689.21,
                         53674668.69])
import_value
```

```
array([13956027.19, 15775092.66, 16975884.13, 23020437.66, 28996784.33,
       36121997.44, 45021620.6 , 60600580.94, 85448380.85, 45412943.89,
       60737134.59, 82607537.26, 84656666.98, 76986012.5 , 54381409.09,
       37516153.22, 39249626.35, 49439155.77, 57187093.43, 60799689.21,
       53674668.69])
```

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

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

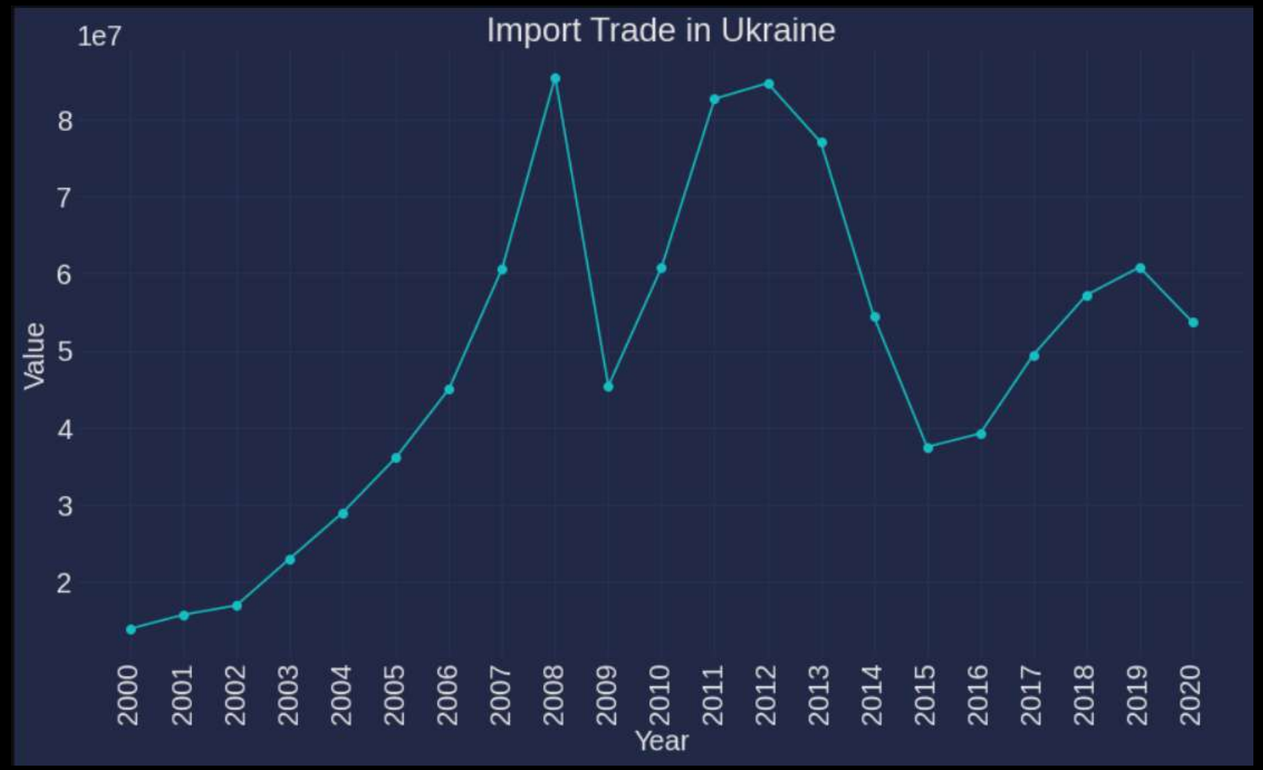
Visualization

```
plt.plot(import_year_arr, import_value,'-o')
plt.rcParams.update({'font.size': 20})
plt.style.use('https://github.com/dhaitz/matplotlib-stylesheets/raw/master/pitayasmoothie-dark.mplstyle')
plt.gcf().set_size_inches(15, 8)
plt.xlabel("Year")
plt.ylabel("Value")
plt.xticks(np.arange(2000,2021,1.0), rotation = 90)
plt.title("Import Trade in Ukraine")
plt.figure(figsize = (15,8))
plt.show()
```

UKRAINE's EXPORT & IMPORT TRADE

Export trade in Ukraine appears to be **less steady.** And it decreased again in 2020.

Import trade in Ukraine looks to be **less consistent as well**. Since 2020, it has been decreasing.



Causes and Impact of the decline in Ukraine' s export and import Trade

One of the main reasons for the decline in Ukraine's export and import trade is that Ukrainian ports on the Black Sea, like the key city of Odesa, are under blockade. As a result, imports and exports are hampered and cannot run properly.

The impacts of the war in Ukraine are being felt not only regionally, but around the world because of the region’s significant contribution to food and energy supplies. Ukraine is a grain and sunflower oil exporter that account 30% of world exports of grains and sunflower oil.

If it happens continuously, there will be a very severe food crisis in Ukraine. Apart from the war, local farmers also have difficulty planting because of the average fertilizers from Russia and Belarus, they both account for over 20% of world exports of fertilizers.

Aids for Ukraine Economic

As part of **Indonesia**, a country that is adamant about the principle of a **"Non-Aligned Movement"**, we would like to give solutions that focus in economy and **don't side with any parties**.

1. Negotiation

As a mediator between Ukraine and Russia, Indonesia can propose measures to increase diplomatic opportunities. Even if it is a truce that ends the war

SDG 16

How Indonesia, as G20 host, can be a mediator between Russia and Ukraine

This picture shows the main meeting room for the G20 finance ministers meeting in Jakarta on February 16, 2022, scheduled to take place on February 17-18. (AFP/Bay Ismoyo)

source: https://www.thejakartapost.com

President Jokowi: Indonesia Ready to Play Mediator Role between Russia and Ukraine

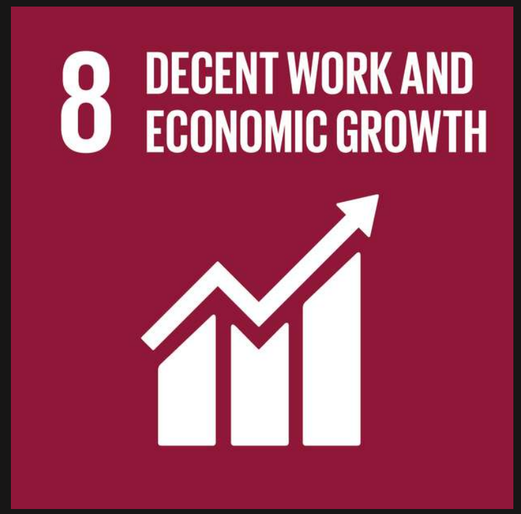
By Office of Assistant to Deputy Cabinet Secretary for State Documents & Translation Date 1 Juli 2022 Category: News Read: 530 Views

source: https://setkab.go.id

Solution for Finance Sector

2. Control the money supply in the economy

Controlling the money supply in the economy is an important monetary approach to control inflation. If the money supply decreases, demand for goods decreases, forcing prices to fall. Another approach for the government to control the money supply is to remove certain paper notes or coins from circulation. Furthermore, the government can control the circulation of money through higher interest rates and open market operations.



" Peace is a key component of sustainable development. "

Reference:

- https://unric.org/en/united-nations-and-the-european-union/
- https://www.thejakartapost.com/opinion/2022/04/10/how-indonesia-as-g20-host-can-be-a-mediator-between-russia-and-ukraine.html
- https://www.undp.org/war-ukraine?utm_source=EN&utm_medium=GSR&utm_content=US_UNDP_PaidSearch_Brand_English&utm_campaign=CENTRAL&c_src=CENTRAL&c_src2=GSR&gclid=Cj0KCQiA37KbBhDgARIsAIzce17dIg7vXwsysrqZO4VZj1q0IK4cHdYf0sKHXxRUkQU6wIIdsBNI0iMaAuFXEALw_wcB
- https://www.investopedia.com/ask/answers/111314/what-methods-can-government-use-control-inflation.asp
- https://news.un.org/pages/wp-content/uploads/2022/04/UN-GCRG-Brief-1.pdf
- https://www.jpmorgan.com/insights/research/gasoline-food-prices-rising
- https://www.weforum.org/agenda/2022/09/inflation-rising-food-energy-prices-economy
- https://www.investopedia.com/ask/answers/022415/how-does-inflation-affect-exchange-rate-between-two-nations.asp
- https://setkab.go.id/en/president-jokowi-indonesia-ready-to-play-mediator-role-between-russia-and-ukraine/