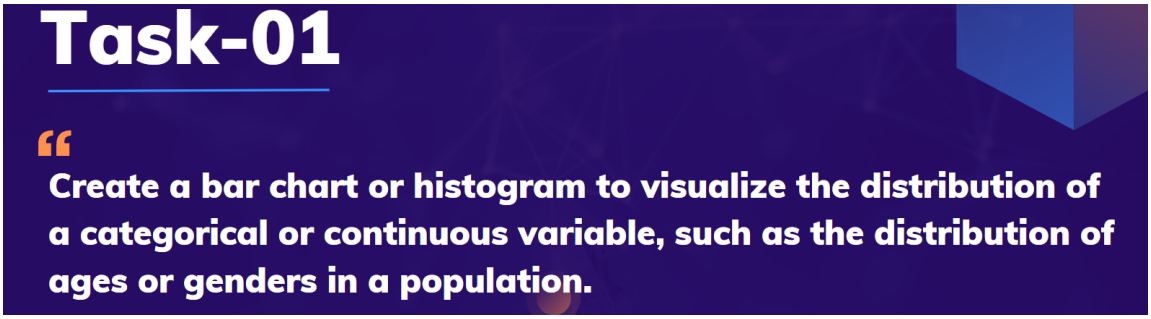
Data Science Internship

- Ankitha Sridhar



Objective: The objective of creating a bar chart or histogram is to visually represent and analyze the distribution of a specific variable, be it categorical or continuous, within a population, enabling insights into the data's characteristics and patterns.

In [44]:
```
#Importing packages

import pandas as pd
import matplotlib.pyplot as plt
```

In [33]:
```
# Excel Data stored as dataframe

data = pd.read_excel("D:/Prodigy/Task 1/worldbank data.xlsx")
```

In [42]:
```
# Displaying the first 5 records

data.head()
```

Out[42]:

	Country Name	Country Code	Region	IncomeGroup	Year	Birth rate, crude (per 1,000 people)	Death rate, crude (per 1,000 people)	Electric power consumption (kWh per capita)	GDP (USI
0	Afghanistan	AFG	South Asia	Low income	2018	NaN	NaN	NaN	1.936300e+1
1	Afghanistan	AFG	South Asia	Low income	2017	33.211	6.575	NaN	2.019180e+1
2	Afghanistan	AFG	South Asia	Low income	2016	33.981	6.742	NaN	1.936260e+1
3	Afghanistan	AFG	South Asia	Low income	2015	34.809	6.929	NaN	1.990710e+1
4	Afghanistan	AFG	South Asia	Low income	2014	35.706	7.141	NaN	2.048490e+1

In [43]:
```
# Displaying the last 5 records

data.tail()
```

Out[43]:

	Country Name	Country Code	Region	IncomeGroup	Year	Birth rate, crude (per 1,000 people)	Death rate, crude (per 1,000 people)	Electric power consumption (kWh per capita)	GDP
12444	Zimbabwe	ZWE	Sub-Saharan Africa	Low income	1964	47.770	13.083	NaN	1.21713
12445	Zimbabwe	ZWE	Sub-Saharan Africa	Low income	1963	47.876	13.419	NaN	1.15951
12446	Zimbabwe	ZWE	Sub-Saharan Africa	Low income	1962	47.950	13.762	NaN	1.11760
12447	Zimbabwe	ZWE	Sub-Saharan Africa	Low income	1961	47.988	14.104	NaN	1.09664
12448	Zimbabwe	ZWE	Sub-Saharan Africa	Low income	1960	47.996	14.441	NaN	1.05299

In [16]:
```
# gives the total number of records (rows) and attributes/fields (columns)

data.shape
```

Out[16]:
```
(12449, 15)
```

In [17]:
```
#total number of records is displayed
```

```
data.size
```

Out[17]:
```
186735
```

In [45]:
```
# Displaying all the column/attribute names

data.columns
```

Out[45]:
```
Index(['Country Name', 'Country Code', 'Region', 'IncomeGroup', 'Year',
       'Birth rate, crude (per 1,000 people)',
       'Death rate, crude (per 1,000 people)',
       'Electric power consumption (kWh per capita)', 'GDP (USD)',
       'GDP per capita (USD)',
       'Individuals using the Internet (% of population)',
       'Infant mortality rate (per 1,000 live births)',
       'Life expectancy at birth (years)',
       'Population density (people per sq. km of land area)',
       'Unemployment (% of total labor force) (modeled ILO estimate)'],
      dtype='object')
```

In [46]:
```
#Data type of each type of attribute will be described

data.dtypes
```

Out[46]:
```
Country Name                                                    object
Country Code                                                    object
Region                                                          object
IncomeGroup                                                     object
Year                                                             int64
Birth rate, crude (per 1,000 people)                           float64
Death rate, crude (per 1,000 people)                           float64
Electric power consumption (kWh per capita)                    float64
GDP (USD)                                                      float64
GDP per capita (USD)                                           float64
Individuals using the Internet (% of population)               float64
Infant mortality rate (per 1,000 live births)                  float64
Life expectancy at birth (years)                               float64
Population density (people per sq. km of land area)            float64
Unemployment (% of total labor force) (modeled ILO estimate)   float64
dtype: object
```

In [18]:
```
data.index
```

Out[18]:
```
RangeIndex(start=0, stop=12449, step=1)
```

In [19]:
```
# Summary statistics of the dataset

data.describe()
```

Out[19]:

	Year	Birth rate, crude (per 1,000 people)	Death rate, crude (per 1,000 people)	Electric power consumption (kWh per capita)	GDP (USD)	GDP per capita (USD)	Indiv usi Inter popu
count	12449.00000	11440.000000	11416.000000	5848.000000	9.578000e+03	9575.000000	5064.(
mean	1989.00000	28.643276	10.588539	3175.294686	1.700740e+11	8231.812259	23.3
std	17.03007	13.131893	5.489382	4467.139298	8.979866e+11	16173.539954	28.3
min	1960.00000	6.900000	1.127000	0.000000	8.824450e+06	34.790600	0.(
25%	1974.00000	16.600000	6.863750	390.385750	1.393010e+09	513.145500	0.5
50%	1989.00000	27.545500	9.200000	1541.895000	7.275305e+09	1852.810000	8.4
75%	2004.00000	40.881250	12.687000	4313.767500	4.857782e+10	7774.565000	41.2
max	2018.00000	58.227000	54.444000	54799.200000	2.050000e+13	189171.000000	100.(

In [39]:
```
# the output indicates that there is no duplicate values in the records
data.duplicated().sum()
```

Out[39]:
```
0
```

In [20]:
```
# Null and NaN values are dropped from the dataset

data1=data.dropna()
data1.head()
```

Out[20]:

	Country Name	Country Code	Region	IncomeGroup	Year	Birth rate, crude (per 1,000 people)	Death rate, crude (per 1,000 people)	Electric power consumption (kWh per capita)	GDP (USD)
63	Albania	ALB	Europe & Central Asia	Upper middle income	2014	12.259	7.219	2309.37	1.322820e+10
64	Albania	ALB	Europe & Central Asia	Upper middle income	2013	12.257	7.096	2533.25	1.277630e+10
65	Albania	ALB	Europe & Central Asia	Upper middle income	2012	12.197	6.996	2118.33	1.231980e+10
66	Albania	ALB	Europe & Central Asia	Upper middle income	2011	12.100	6.915	2205.70	1.289090e+10
67	Albania	ALB	Europe & Central Asia	Upper middle income	2010	12.001	6.841	1943.34	1.192700e+10

In [48]:
```
data1.shape
```

Out[48]:
```
(2775, 15)
```

In [49]:
```
#Verifying to check the presence of null values

data1.isna()
```

Out[49]:

	Country Name	Country Code	Region	IncomeGroup	Year	Birth rate, crude (per 1,000 people)	Death rate, crude (per 1,000 people)	Electric power consumption (kWh per capita)	GDP (USD)	ca (U
63	False	False	False	False	False	False	False	False	False	
64	False	False	False	False	False	False	False	False	False	
65	False	False	False	False	False	False	False	False	False	
66	False	False	False	False	False	False	False	False	False	
67	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
12410	False	False	False	False	False	False	False	False	False	
12411	False	False	False	False	False	False	False	False	False	
12412	False	False	False	False	False	False	False	False	False	
12413	False	False	False	False	False	False	False	False	False	
12414	False	False	False	False	False	False	False	False	False	

2775 rows × 15 columns

In [23]:
```
#sum() sums up the boolean values [true=1,false=0].

data1.isna().sum()

#Here we can see all columns' NaN values are dropped. There is no missing value in
```

Out[23]:
```
Country Name                                                    0
Country Code                                                    0
Region                                                          0
IncomeGroup                                                     0
Year                                                            0
Birth rate, crude (per 1,000 people)                            0
Death rate, crude (per 1,000 people)                            0
Electric power consumption (kWh per capita)                     0
GDP (USD)                                                       0
GDP per capita (USD)                                            0
Individuals using the Internet (% of population)                0
Infant mortality rate (per 1,000 live births)                   0
Life expectancy at birth (years)                                0
Population density (people per sq. km of land area)             0
Unemployment (% of total labor force) (modeled ILO estimate)    0
dtype: int64
```

In [24]:
```
data1.info()
```

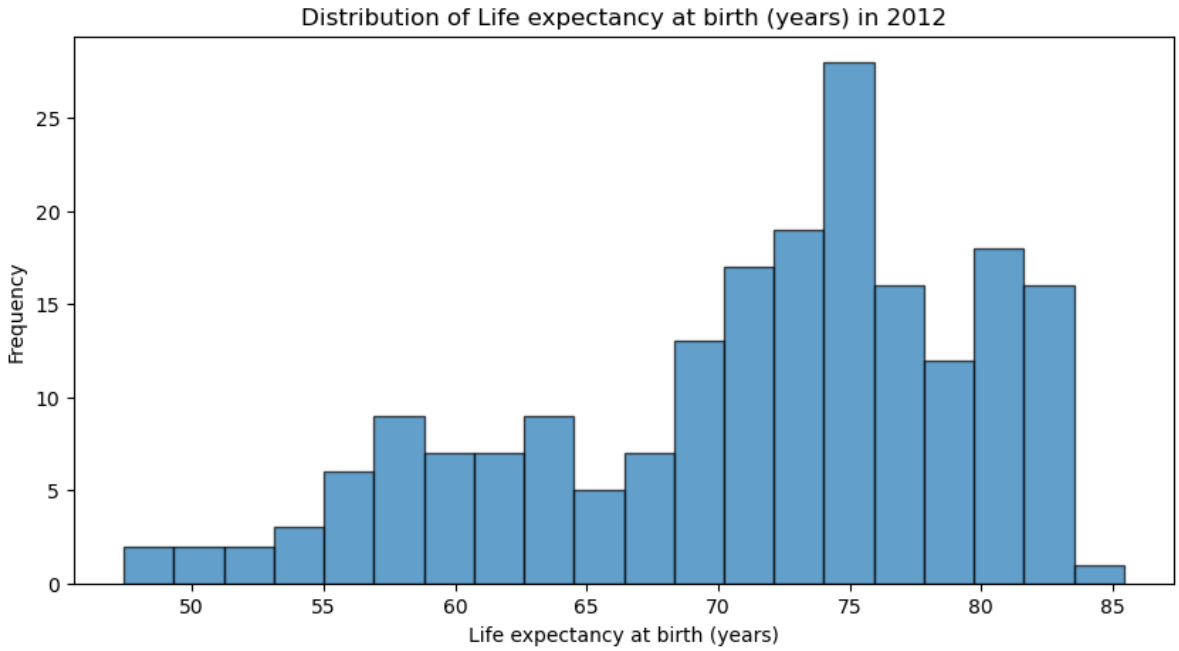
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2775 entries, 63 to 12414
Data columns (total 15 columns):
 #   Column                                                  Non-Null Count
Dtype
---  ------                                                  --------------
-----
 0   Country Name                                            2775 non-null
object
 1   Country Code                                            2775 non-null
object
 2   Region                                                  2775 non-null
object
 3   IncomeGroup                                             2775 non-null
object
 4   Year                                                    2775 non-null
int64
 5   Birth rate, crude (per 1,000 people)                    2775 non-null
float64
 6   Death rate, crude (per 1,000 people)                    2775 non-null
float64
 7   Electric power consumption (kWh per capita)             2775 non-null
float64
 8   GDP (USD)                                               2775 non-null
float64
 9   GDP per capita (USD)                                    2775 non-null
float64
 10  Individuals using the Internet (% of population)        2775 non-null
float64
 11  Infant mortality rate (per 1,000 live births)           2775 non-null
float64
 12  Life expectancy at birth (years)                        2775 non-null
float64
 13  Population density (people per sq. km of land area)     2775 non-null
float64
 14  Unemployment (% of total labor force) (modeled ILO estimate)  2775 non-null
float64
dtypes: float64(10), int64(1), object(4)
memory usage: 346.9+ KB
```

In [25]:
```
year = 2012
variable = 'Life expectancy at birth (years)'
```

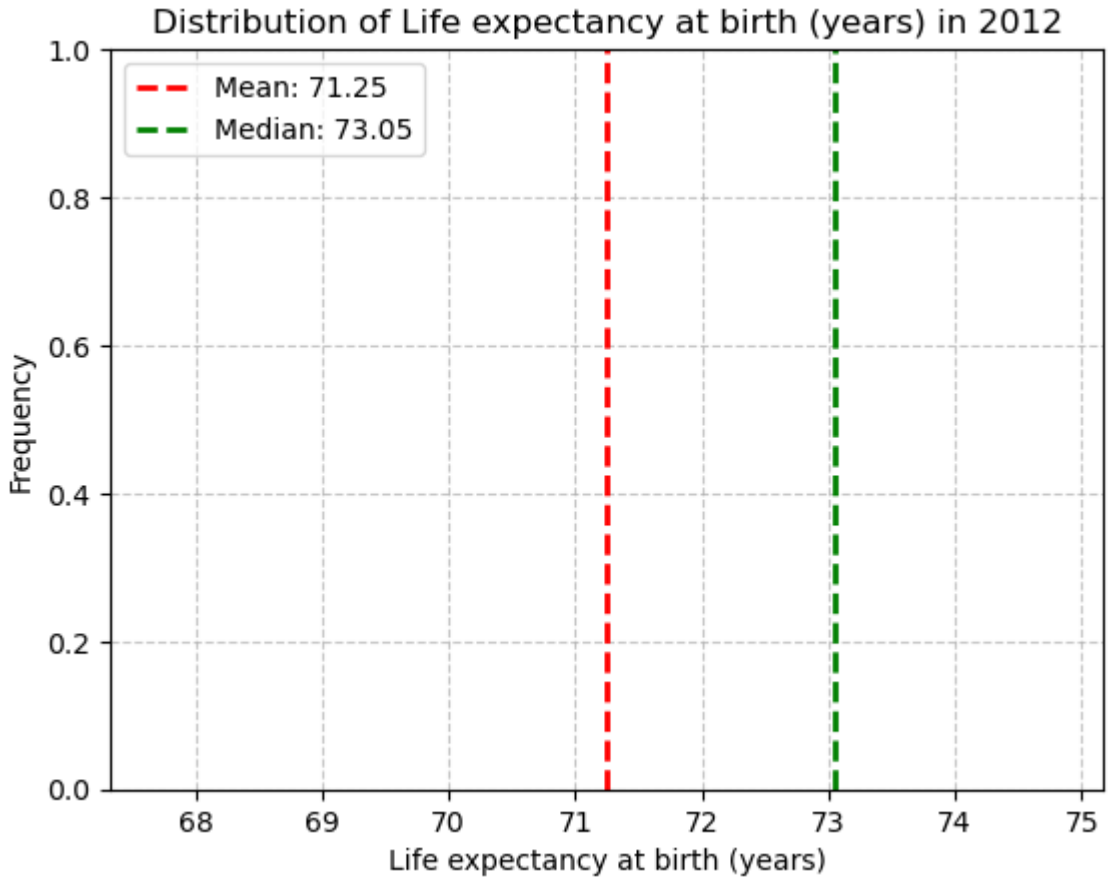
In [26]:
```
#Filter the data for a specific year
data_year = data[data['Year'] == year]
```

Creating a histogram

In [27]:
```
plt.figure(figsize=(10,5))
plt.hist(data_year[variable],bins=20, edgecolor='k', alpha=0.7)

plt.title(f'Distribution of {variable} in {year}')
plt.xlabel(variable)
plt.ylabel('Frequency')
plt.show()
```

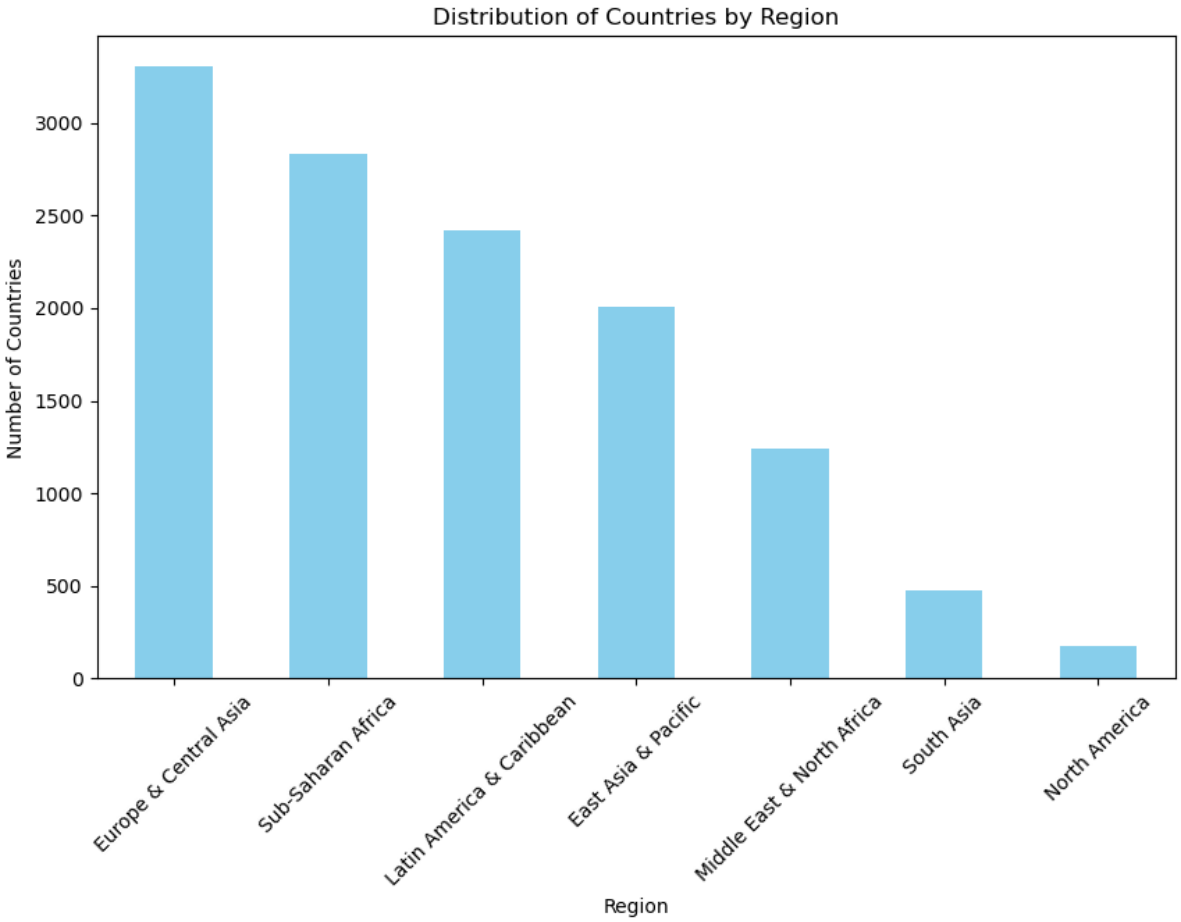


In [28]:
```
# Customize the plot further
plt.title(f'Distribution of {variable} in {year}')
plt.xlabel(variable)
plt.ylabel('Frequency')

plt.grid(True, linestyle='--', alpha=0.7)

# Add labels for mean and median
mean_value = data_year[variable].mean()
median_value = data_year[variable].median()
plt.axvline(mean_value, color='red', linestyle='dashed', linewidth=2, label=f'Mean:
plt.axvline(median_value, color='green', linestyle='dashed', linewidth=2, label=f'M
plt.legend()
plt.show()
```



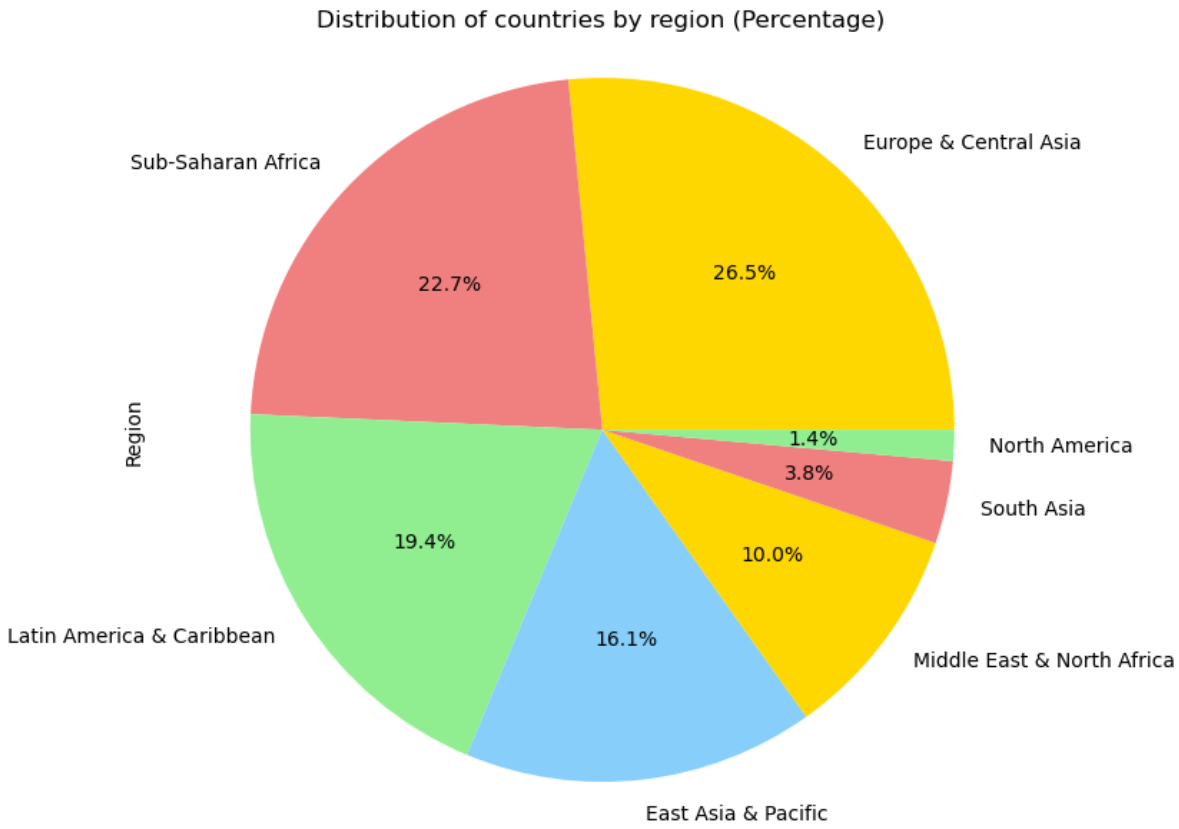
Bar Plot

In [29]:
```
region_counts = data['Region'].value_counts()

# Create a bar chart
plt.figure(figsize=(10, 6))
region_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Region')
plt.ylabel('Number of Countries')
plt.title('Distribution of Countries by Region')
plt.xticks(rotation=45)
plt.show()
```



Pie Plot

In [32]:
```
plt.figure(figsize=(15,6))
plt.subplot(1,2,2)
region_percentage = (region_counts / region_counts.sum())*100
colors = ['gold', 'lightcoral', 'lightgreen', 'lightskyblue']

region_percentage.plot(kind='pie', autopct='%1.1f%%', colors=colors)
plt.axis('equal')
plt.title("Distribution of countries by region (Percentage)")

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


Distribution of countries by region (Percentage)
Sub-Saharan Africa
Europe & Central Asia
22.7%
26.5%
Region
1.4%
North America
3.8%
South Asia
19.4%
10.0%
Latin America & Caribbean
16.1%
Middle East & North Africa
East Asia & Pacific

Thank you!