--------------------------------------- Training ------------------------------------

NOTE: This section is for practising. Questions specified by Q are not a part of your coursework. You can check the comments to see what each cell does.

Importing necessary Libraries

In [1]:

```
import pandas as pd
import numpy as np
```

Python Warmup exercises

- Q1: Write a code that receives an optional text, print all letters in the text and store them in a list

In [2]:

```
output=[]
text1='''welcome to workshop 1'''
for i in text1:
    print(i)
    output.append(i)
print(output)
```

```
w
e
l
c
o
m
e
 
t
o
 
w
o
r
k
s
h
o
p
 
1
['w', 'e', 'l', 'c', 'o', 'm', 'e', ' ', 't', 'o', ' ', 'w', 'o', 'r', 'k',
's', 'h', 'o', 'p', ' ', '1']
```

- Q2: Write a code that receives an optional text, prints all words in the text and store them in a list

In [3]:
```
output=[]
text1='''welcome to workshop 1'''
for word in text1.split():
    print(word)
    output.append(word)
print(output)
```

```
welcome
to
workshop
1
['welcome', 'to', 'workshop', '1']
```

- Q3: Write Q2 answer as a function

In [4]:
```
text1='''welcome to workshop 1'''
def print_word(text):
    for word in text.split():
        print(word)
print_word(text1)
```

```
welcome
to
workshop
1
```

- Q4: Write a program to find out the average of a set of integers. The program prompts the user to enter the count of numbers

In [6]:
```
count = int(input("Enter the count of numbers: ")) # This gets the count of num
# puts it in an integer variable
i = 0
sum = 0
for i in range(count):
    x = int(input("Enter an integer: "))
    sum = sum + x
avg = sum/count
print("The average is: ", avg)
```

```
Enter the count of numbers: 1
Enter an integer: 1
The average is:  1.0
```

- Q5: Write a program to check whether the given number is even or not.

In [7]:

```python
# If we divide an even number by 2, the remainder will be zero
number = input("Enter a number: ")
x = int(number)%2
if x == 0:
    print("The number is Even.")
else:
    print("The number is Odd.")
```

```
Enter a number: 1
The number is Odd.
```

- Q6: Write a program to find the average of 4 numbers using while loop. The program prompts the user to enter the numbers

In [8]:

```python
count = 0
sum = 0.0
while(count<4):
    number = float(input("Enter a real number: "))
    count=count+1
    sum = sum+number
avg = sum/10
print("Average is :",avg)
```

```
Enter a real number: 1
Enter a real number: 1
Enter a real number: 1
Enter a real number: 1
Average is : 0.4
```

- Q7: Write a function to find the average of a list of integers

In [9]:

```python
def give_me_average(in_list):
    count=len(in_list)
    sum=0
    for i in in_list:
        sum=sum+i
    avg = sum/count
    return avg
a=[1,2,3,4]
print("Average is :",give_me_average(a))
```

```
Average is : 2.5
```

- Q8: Write a program to display all integers within the range 100-120 whose sum of digits is an even number and store them in a list

In [10]:

```
output=[]
for i in range(100,120):
    num = i
    sum = 0
    while(num!=0):
        digit = num%10
        sum = sum + digit
        num = num//10
    if(sum%2==0):
        output.append(i)
        print(i)
print(output)
```

```
101
103
105
107
109
110
112
114
116
118
[101, 103, 105, 107, 109, 110, 112, 114, 116, 118]
```

- Q9: Write a program to check whether the given integer is a multiple of both 5 and 7

In [12]:

```
number = int(input("Enter an integer: "))
if((number%5==0)and(number%7==0)):
    print(number, "is a multiple of both 5 and 7")
else:
    print(number, "is not a multiple of both 5 and 7")
```

```
Enter an integer: 1
1 is not a multiple of both 5 and 7
```

Pandas warmup exercises

```
• You can install Pandas on your system by using pip as follows:
pip install pandas

• You can import Pandas as follows:
import pandas

• Pandas is usually imported under the pd alias.
import pandas as pd

• You can check Pandas version as follows:
import pandas as pd
print(pd.__version__)
```

In [13]:
```
# You can use the DataFrame() method to create a Pandas DataFrame from a dictic
data = {"calories": [420, 380, 390], "duration": [50, 40, 45]}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
print('\n') # Addine an empty line to the end of the data frame
# ---
data = [[420, 380, 390],[50, 40, 45]]
df = pd.DataFrame(data)
print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45



     0    1    2
0  420  380  390
1   50   40   45
```

In [14]:
```
# You can use loc() or iloc() to return values, rows, or columns inside a data
# column name, but using iloc() you only need to specify the column index.
data = {"calories": [420, 380, 390], "duration": [50, 40, 45]}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
print("\n")
print(df.loc[0,'calories'])
print("\n")
print(df.loc[0,:])
print("\n")
print(df.iloc[:,0])
```

```
   calories  duration
0       420        50
1       380        40
2       390        45



420



calories    420
duration     50
Name: 0, dtype: int64



0    420
1    380
2    390
Name: calories, dtype: int64
```

In [15]:
```
# You can get the indices and columns names using index and column attributes a

data = {"calories": [420, 380, 390], "duration": [50, 40, 45]}
#load data into a DataFrame object:
df = pd.DataFrame(data)
index_names=list(df.index)
column_names=list(df.columns)
print(index_names,end='\n\n')
print(column_names)
```

```
[0, 1, 2]

['calories', 'duration']
```

In [16]:
```
# With the index and columns argument, you can name your own indexes and column
data = {
 "calories": [420, 380, 390],
 "duration": [50, 40, 45]
}
df = pd.DataFrame(data, index = ["day1", "day2", "day3"])
print(df)
```

```
      calories  duration
day1       420        50
day2       380        40
day3       390        45
```

In [17]:
```
# With the index argument, you can name your own indexes and columns
data =[[420, 380, 390],[50, 40, 45]]
df = pd.DataFrame(data, columns = ["c1", "c2", "c3"])
print(df)
```

```
    c1   c2   c3
0  420  380  390
1   50   40   45
```

Importing a dataset for this workshop

Importing the dataset directly from the source

In [18]:
```
# Installing a package which is needed to download the dataset from its online
# URL : https://archive.ics.uci.edu/dataset/2/adult
!pip3 install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in c:\users\hp\anaconda3\lib\site-pa
ckages (0.0.3)
```

In [19]:

```
# Downloading the dataset from the online source. The first two lines are given
from ucimlrepo import fetch_ucirepo
# fetch dataset
adult = fetch_ucirepo(id=2)

# Putting data in a pandas dataframe
X = adult.data.features
y = adult.data.targets
data=pd.concat([X,y],axis=1)

data
```

Out[19]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black
...	...	...	...	...	...	...	...	...	...
48837	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
48838	64	NaN	321403	HS-grad	9	Widowed	NaN	Other-relative	Black
48839	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
48840	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
48841	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

48842 rows × 15 columns

In [20]:
```
#printing data
data
```

Out[20]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black
...	...	...	...	...	...	...	...	...	...
48837	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
48838	64	NaN	321403	HS-grad	9	Widowed	NaN	Other-relative	Black
48839	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
48840	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
48841	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

48842 rows × 15 columns

Importing the dataset from a folder on your local disk

In [21]:
```
#importing the dataset as a Pandas DataFrame into Python if the dataset is stor
# You can download the dataset from the following URL: https://archive.ics.uci.
data = pd.read_csv('C:/Users/HP/Downloads/adult.csv') # Replace the current dat
```

Exploring the dataset

In [22]:

```
# Showing the first 5 rows of the dataset
data.head()
```

Out[22]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	s
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Ma
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Ma
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Ma
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Ma
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fema

In [23]:

```
# Finding the shape of the data
print(data.shape)
```

```
(48842, 15)
```

In [24]:

```
# Generate a dataset by randomly extracting 30000 rows (samples)
data_new = data.sample(n=30000, random_state = 48)
```

In [25]:

```
# Printing the new dataset
data_new
```

Out[25]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
8029	29	Private	216481	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White
45203	36	Private	280570	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White
27498	25	?	100903	Bachelors	13	Married-civ-spouse	?	Wife	White
48416	47	Private	145636	Assoc-voc	11	Married-civ-spouse	Handlers-cleaners	Husband	White
43230	33	Private	119422	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White
...	...	...	...	...	...	...	...	...	...
37544	20	Private	166371	HS-grad	9	Never-married	Craft-repair	Other-relative	White
20482	80	Private	202483	HS-grad	9	Married-spouse-absent	Adm-clerical	Not-in-family	White
39667	20	Private	175808	HS-grad	9	Never-married	Craft-repair	Own-child	White
40001	25	State-gov	31350	Some-college	10	Never-married	Other-service	Not-in-family	White
48523	19	Private	239057	HS-grad	9	Never-married	Craft-repair	Own-child	White

30000 rows × 15 columns

In [26]:

```
# The indices of different rows in the dataset are currently messy. This happen
# the dataset if you are unsure the indices are correct.
data_new.reset_index(drop=True, inplace=True)
```

In [27]:
```
# Checking if the indices are correct
data_new
```

Out[27]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	29	Private	216481	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White
1	36	Private	280570	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White
2	25	?	100903	Bachelors	13	Married-civ-spouse	?	Wife	White
3	47	Private	145636	Assoc-voc	11	Married-civ-spouse	Handlers-cleaners	Husband	White
4	33	Private	119422	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White
...	...	...	...	...	...	...	...	...	...
29995	20	Private	166371	HS-grad	9	Never-married	Craft-repair	Other-relative	White
29996	80	Private	202483	HS-grad	9	Married-spouse-absent	Adm-clerical	Not-in-family	White
29997	20	Private	175808	HS-grad	9	Never-married	Craft-repair	Own-child	White
29998	25	State-gov	31350	Some-college	10	Never-married	Other-service	Not-in-family	White
29999	19	Private	239057	HS-grad	9	Never-married	Craft-repair	Own-child	White

30000 rows × 15 columns

In [28]:
```
data
```

Out[28]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black
...	...	...	...	...	...	...	...	...	...
48837	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
48838	64	NaN	321403	HS-grad	9	Widowed	NaN	Other-relative	Black
48839	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
48840	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
48841	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

48842 rows × 15 columns

In [29]:

```
# Getting statistical information of the dataset for different columns (feature
data.describe(include="all")
```

Out[29]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	re
count	48842.000000	47879	4.884200e+04	48842	48842.000000	48842	47876	
unique	NaN	9	NaN	16	NaN	7	15	
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Prof-specialty	
freq	NaN	33906	NaN	15784	NaN	22379	6172	
mean	38.643585	NaN	1.896641e+05	NaN	10.078089	NaN	NaN	
std	13.710510	NaN	1.056040e+05	NaN	2.570973	NaN	NaN	
min	17.000000	NaN	1.228500e+04	NaN	1.000000	NaN	NaN	
25%	28.000000	NaN	1.175505e+05	NaN	9.000000	NaN	NaN	
50%	37.000000	NaN	1.781445e+05	NaN	10.000000	NaN	NaN	
75%	48.000000	NaN	2.376420e+05	NaN	12.000000	NaN	NaN	
max	90.000000	NaN	1.490400e+06	NaN	16.000000	NaN	NaN	

In [30]:

```
# Droping row with Null values (This topic will be discussed in detail in the n
data.dropna(inplace=True)
```

In [31]:
```
data
```

Out[31]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black
...	...	...	...	...	...	...	...	...	...
48836	33	Private	245211	Bachelors	13	Never-married	Prof-specialty	Own-child	White
48837	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
48839	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
48840	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
48841	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

47621 rows × 15 columns

```
NOTE: As you can see, because a few rows have been dropped, the index numbers
don't correspond to the actual row number (the number of rows is 47623 but the
last index is 48841). We need to reindex the data
```

In [32]:
```
# reindexing the original data
data.reset_index(drop=True, inplace=True)
```

In [33]:

```
data
```

Out[33]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black
...	...	...	...	...	...	...	...	...	...
47616	33	Private	245211	Bachelors	13	Never-married	Prof-specialty	Own-child	White
47617	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
47618	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
47619	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
47620	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

47621 rows × 15 columns

In [34]:
```
# Getting the number of Null values for different columns
data.isna().sum()
```

Out[34]:
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

In [35]:
```
# Showing the dataset information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47621 entries, 0 to 47620
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             47621 non-null  int64 
 1   workclass       47621 non-null  object
 2   fnlwgt          47621 non-null  int64 
 3   education       47621 non-null  object
 4   education-num   47621 non-null  int64 
 5   marital-status  47621 non-null  object
 6   occupation      47621 non-null  object
 7   relationship    47621 non-null  object
 8   race            47621 non-null  object
 9   sex             47621 non-null  object
 10  capital-gain    47621 non-null  int64 
 11  capital-loss    47621 non-null  int64 
 12  hours-per-week  47621 non-null  int64 
 13  native-country  47621 non-null  object
 14  income          47621 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.4+ MB
```

In [36]:
```
data.shape
```

Out[36]:
```
(47621, 15)
```

In [37]:
```
# Getting the count of different values in the column "education-num"
data['education-num'].value_counts()
```

Out[37]:
```
education-num
9     15444
10    10512
13     7881
14     2610
11     2034
7      1746
12     1566
6      1336
4       912
15      819
5       735
8       633
16      582
3       494
2       239
1        78
Name: count, dtype: int64
```

In [38]:
```
# Getting the count of different values in the column "education"
data['education'].value_counts()
```

Out[38]:
```
education
HS-grad         15444
Some-college    10512
Bachelors        7881
Masters          2610
Assoc-voc        2034
11th             1746
Assoc-acdm       1566
10th             1336
7th-8th           912
Prof-school       819
9th               735
12th              633
Doctorate         582
5th-6th           494
1st-4th           239
Preschool          78
Name: count, dtype: int64
```

In [39]:
```
# Dropping a column
data = data.drop(['fnlwgt'], axis=1)
```

In [40]:
```
data.shape
```

Out[40]:
```
(47621, 14)
```

In [41]:

```
# Getting the number of unique values of a column
data['education'].nunique()
```

Out[41]:

```
16
```

In [42]:

```
# Finding how many rows are related to either gender
data['sex'].value_counts()
```

Out[42]:

```
sex
Male      31937
Female    15684
Name: count, dtype: int64
```

In [43]:

```
# Calculating the average age of different genders in the dataset
data['age'].groupby([data['sex']]).mean()
```

Out[43]:

```
sex
Female    36.961043
Male      39.465542
Name: age, dtype: float64
```

In [44]:

```
# Getting the average age of different genders in the dataset broken down based
data['age'].groupby([data['sex'],data['education']]).mean()
```

Out[44]:

```
sex     education
Female  10th            36.520642
        11th            29.891410
        12th            29.716418
        1st-4th         46.508475
        5th-6th         45.346774
        7th-8th         51.088889
        9th             41.566038
        Assoc-acdm      36.402619
        Assoc-voc       38.000000
        Bachelors       35.747837
        Doctorate       45.557522
        HS-grad         38.862285
        Masters         42.520482
        Preschool       44.409091
        Prof-school     40.131783
        Some-college    33.778719
Male    10th            38.406667
        11th            33.479185
        12th            33.842593
        1st-4th         46.050000
        5th-6th         42.524324
        7th-8th         48.608443
        9th             40.556405
        Assoc-acdm      38.537173
        Assoc-voc       38.853881
        Bachelors       40.190319
        Doctorate       47.936034
        HS-grad         39.073626
        Masters         44.584270
        Preschool       41.160714
        Prof-school     46.017391
        Some-college    37.176643
Name: age, dtype: float64
```

In [45]:

```
# Getting the maximum age of different races in the dataset
data['age'].groupby([data['race']]).max()
```

Out[45]:

```
race
Amer-Indian-Eskimo    82
Asian-Pac-Islander    90
Black                 90
Other                 77
White                 90
Name: age, dtype: int64
```

In [46]:

```
# Extracting the age and education columns and creating a new DataFrame using t
a=data['age']
b=data['education']
new_data=pd.concat([a,b],axis=1)
```

In [47]:
```
data
```

Out[47]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female
...	...	...	...	...	...	...	...	...	...
47616	33	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Male
47617	39	Private	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female
47618	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male
47619	44	Private	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male
47620	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male

47621 rows × 14 columns

- Q10: Write a function that receives the dataset and replace Famle with F and Male with M (please try to write it yourself before checking the answer in the next cell)

In [48]:

```python
# Answer
def encode_sex(data):
    rows=data.shape[0]
    a=data.loc[:,'sex']
    for i in range(rows):
        if a[i]=="Male":
            data.loc[i,"sex"]="M"
        elif a[i]=="Female":
            data.loc[i,"sex"]="F"
    return data
```

In [49]:

```python
# Copying the data
data_copy=data.copy()
```

In [50]:

```python
# Applying the encode_sex function to the copied data
data_encoded=encode_sex(data_copy)
data_encoded.head()
```

Out[50]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	M	2174
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	M	0
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	M	0
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	M	0
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	F	0

In [51]:
```
data
```

Out[51]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female
...	...	...	...	...	...	...	...	...	...
47616	33	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Male
47617	39	Private	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female
47618	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male
47619	44	Private	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male
47620	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male

47621 rows × 14 columns

Tasks

TASK 1.1: Import the dataset from the URL we used in this workshop. Then generate a new dataset by randomly extracting 10000 samples. Reindex the generated dataset and remove NULL values. Name the new dataset 'task_dataset'

In [52]:
```
############# WRITE THE CODE IN THIS CELL ####################
# Downloading the dataset from the online source. The first two lines are given
from ucimlrepo import fetch_ucirepo
# fetch dataset
adult = fetch_ucirepo(id=2)

# Putting data in a pandas dataframe
X = adult.data.features
y = adult.data.targets
data=pd.concat([X,y],axis=1)

#Printing data
data
```

Out[52]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black
...	...	...	...	...	...	...	...	...	...
48837	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
48838	64	NaN	321403	HS-grad	9	Widowed	NaN	Other-relative	Black
48839	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
48840	44	Private	83891	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
48841	35	Self-emp-inc	182148	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

48842 rows × 15 columns

```
############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############

The adult dataset was fetched from the UCI Machine learning repository
ucimlrepo, and i loaded it into a pandas dataframe, and splitted to set all
the features columns as X and the target column (Y) as income. i concatenated
(X,y) back into a single dataframe and then stored into Data, which was
printed and shown above as containing 48842 rows and 15 columns.

i went further to generate new dataset by randomly extracting 10000 rows using
the sample function. The data was then renamed, stored into data_new and
printed. The randomly sampled data (data_new) was reindexed.

further, data preprocessing and cleaning was carried out by dropping all null
values in the dataset and the cleaned data stored as task_dataset
```

```
 (120 WORDS)
```

In [53]:

```
# Generate a dataset by randomly extracting 10000 rows (samples)
data_new = data.sample(n=10000, random_state = 48)

# Printing the new dataset
data_new
```

Out[53]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
8029	29	Private	216481	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White
45203	36	Private	280570	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White
27498	25	?	100903	Bachelors	13	Married-civ-spouse	?	Wife	White
48416	47	Private	145636	Assoc-voc	11	Married-civ-spouse	Handlers-cleaners	Husband	White
43230	33	Private	119422	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White
...	...	...	...	...	...	...	...	...	...
5637	19	Private	63363	Some-college	10	Never-married	Sales	Own-child	White
46277	53	Private	58535	HS-grad	9	Divorced	Sales	Not-in-family	White
26200	30	Private	342709	HS-grad	9	Married-civ-spouse	Handlers-cleaners	Husband	White
33519	41	Self-emp-not-inc	134724	Assoc-voc	11	Married-civ-spouse	Other-service	Wife	White
28113	21	Private	252253	Some-college	10	Never-married	Adm-clerical	Unmarried	Black

10000 rows × 15 columns

In [54]:
```
# Reindexing the indices and printing the outcome
data_new.reset_index(drop=True, inplace=True)

#printing outcome
data_new
```

Out[54]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	29	Private	216481	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	F
1	36	Private	280570	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	
2	25	?	100903	Bachelors	13	Married-civ-spouse	?	Wife	White	F
3	47	Private	145636	Assoc-voc	11	Married-civ-spouse	Handlers-cleaners	Husband	White	
4	33	Private	119422	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	
...	...	...	...	...	...	...	...	...	...	
9995	19	Private	63363	Some-college	10	Never-married	Sales	Own-child	White	F
9996	53	Private	58535	HS-grad	9	Divorced	Sales	Not-in-family	White	F
9997	30	Private	342709	HS-grad	9	Married-civ-spouse	Handlers-cleaners	Husband	White	
9998	41	Self-emp-not-inc	134724	Assoc-voc	11	Married-civ-spouse	Other-service	Wife	White	F
9999	21	Private	252253	Some-college	10	Never-married	Adm-clerical	Unmarried	Black	F

10000 rows × 15 columns

In [55]:
```
# Droping row with Null values and renaming
data_new.dropna(inplace=True)
```

In [56]:

```
# Renaming the dataset
task_dataset = data_new

#printing
task_dataset
```

Out[56]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	29	Private	216481	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	F
1	36	Private	280570	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	
2	25	?	100903	Bachelors	13	Married-civ-spouse	?	Wife	White	F
3	47	Private	145636	Assoc-voc	11	Married-civ-spouse	Handlers-cleaners	Husband	White	
4	33	Private	119422	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	
...	...	...	...	...	...	...	...	...	...	
9995	19	Private	63363	Some-college	10	Never-married	Sales	Own-child	White	F
9996	53	Private	58535	HS-grad	9	Divorced	Sales	Not-in-family	White	F
9997	30	Private	342709	HS-grad	9	Married-civ-spouse	Handlers-cleaners	Husband	White	
9998	41	Self-emp-not-inc	134724	Assoc-voc	11	Married-civ-spouse	Other-service	Wife	White	F
9999	21	Private	252253	Some-college	10	Never-married	Adm-clerical	Unmarried	Black	F

9765 rows × 15 columns

In [57]:

```
# Reindexing the indices and printing the outcome
task_dataset.reset_index(drop=True, inplace=True)

#printing outcome
task_dataset
```

Out[57]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	29	Private	216481	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	F
1	36	Private	280570	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	
2	25	?	100903	Bachelors	13	Married-civ-spouse	?	Wife	White	F
3	47	Private	145636	Assoc-voc	11	Married-civ-spouse	Handlers-cleaners	Husband	White	
4	33	Private	119422	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	
...	...	...	...	...	...	...	...	...	...	
9760	19	Private	63363	Some-college	10	Never-married	Sales	Own-child	White	F
9761	53	Private	58535	HS-grad	9	Divorced	Sales	Not-in-family	White	F
9762	30	Private	342709	HS-grad	9	Married-civ-spouse	Handlers-cleaners	Husband	White	
9763	41	Self-emp-not-inc	134724	Assoc-voc	11	Married-civ-spouse	Other-service	Wife	White	F
9764	21	Private	252253	Some-college	10	Never-married	Adm-clerical	Unmarried	Black	F

9765 rows × 15 columns

In [58]:
```
# Getting the number of Null values for different columns
task_dataset.isna().sum()
```

Out[58]:
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

TASK 1.2: Write a code to find how much contribution each sex and occupation category made to the capital-gain on average. Apply the code to the task_dataset and report the result (Hint: you need to use the groupby method)

In [59]:

```
############# WRITE THE CODE IN THIS CELL ####################
# Getting the average capital gain based on the contribution from each sex and
task_dataset['capital-gain'].groupby([task_dataset['sex'],task_dataset['occupat
```

Out[59]:

```
sex     occupation
Female  ?                     359.855556
        Adm-clerical          508.217678
        Craft-repair          273.969231
        Exec-managerial      1311.540984
        Farming-fishing      1049.523810
        Handlers-cleaners     199.136364
        Machine-op-inspct     290.217687
        Other-service         191.525394
        Priv-house-serv        74.425926
        Prof-specialty       1008.057269
        Protective-serv      5081.000000
        Sales                 276.654822
        Tech-support          382.803419
        Transport-moving      533.800000
Male    ?                     879.403509
        Adm-clerical          425.640751
        Armed-Forces            0.000000
        Craft-repair          906.637306
        Exec-managerial      2339.700234
        Farming-fishing       441.013746
        Handlers-cleaners     149.265957
        Machine-op-inspct     577.642082
        Other-service         155.411765
        Priv-house-serv       118.800000
        Prof-specialty       3918.730198
        Protective-serv       368.563158
        Sales                1129.695105
        Tech-support          419.494565
        Transport-moving      409.625000
Name: capital-gain, dtype: float64
```

```
############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############

The result of the average capital gain of each sex based on occupation shows
some economic disparities between the males and the females within same
occupation. Occupations such as Admin clerical, farming-fishing, Handlers-
cleaners, Protective serv and transport-moving, recorded higher average
contribution from the females to the capital gain, than the males. on the
average females in Admin Clerical made 12% more contribution than the males.
40% more in farming-fishing occupation, 14% more in Handlers-cleaners, 86%
more in Protective-serv occupation and 14% more in Transport-moving.
```

While in occupations such as Craft repair, Exec-managerial, Machine-op-inspct, Priv-house-serv, Prof-specialty, Sales and Tech-support, the males recorded higher average contribution to the capital gain than the females with 54%, 28%, 33%, 24%, 60%, 60% and 4% respectively

(121 WORDS)

TASK 1.3: Write a code to find the country with the highest number of people with a Bachelors degree. Apply the code to the task_dataset and report the result (Hint: you need to use the groupby method)

In [60]:

```
############# WRITE THE CODE IN THIS CELL ###################
# adjust column names based on the actual dataset
bachelors_task_dataset = task_dataset[task_dataset['education'] == 'Bachelors']

# Group by 'native-country' and count the number of people with a Bachelor's de
country_bachelors_count = bachelors_task_dataset.groupby('native-country').size

# Find the country with the highest count
highest_country = country_bachelors_count.idxmax()
highest_count = country_bachelors_count.max()

# Sort the counts in descending order
country_bachelors_count_sorted = country_bachelors_count.sort_values(ascending=

# Print the sorted counts
print(country_bachelors_count_sorted)

print(f"The country with the highest number of people with a Bachelors degree i
native-country
United-States         1456
?                       30
Philippines             15
England                  9
Japan                    8
Canada                   7
Mexico                   7
India                    7
Italy                    6
China                    6
South                    6
Puerto-Rico              6
Cuba                     5
Taiwan                   5
Germany                  5
Iran                     4
El-Salvador              4
Jamaica                  3
Vietnam                  3
Cambodia                 2
Peru                     2
Dominican-Republic       2
Thailand                 2
Ireland                  1
Guatemala                1
Greece                   1
Poland                   1
Ecuador                  1
Columbia                 1
dtype: int64
The country with the highest number of people with a Bachelors degree is United-States with 1456 people.
```

```
############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############

The codes above sought to find the country with the highest distribution of
individuals with a Bachelor's degree across different countries from the
task_dataset

This dataset is converted into a pandas DataFrame, task_dataset, which
represents the larger dataset containing various education levels and
countries of origin for different individuals. The dataset is then filtered to
select only the rows where the education column has the value 'Bachelors'.
This filtered dataset is stored in bachelors_task_dataset.

The code then grouped the filtered dataset by the native-country column, which
categorizes individuals based on their country of origin. The size() function
is used to count the number of individuals with a Bachelor's degree in each
country, resulting in a Series object, country_bachelors_count, indexed by
country name with counts as values.
```

To find the country with the highest number of individuals holding a Bachelor's degree, i used the idxmax() on country_bachelors_count. This function returns the country name of the first occurrence of the maximum value followed by the highest count of individuals with a Bachelor's degree which is obtained using max().

Finally, The counts of individuals with a Bachelor's degree by country were sorted in descending order using sort_values(ascending=False). This sorted Series, country_bachelors_count_sorted, simplifies the distribution across different countries, and the United States was identified as the country with the highest count of individuals with Bachelors degree.

(222 WORDS)

TASK 1.4: Write a code to receive two lists including five names and their respective ages and print 'Hello Name Age'

For example, if it received a list of two names ['Amin', 'Michael'] and respective ages [27,38], it would print 'Hello Amin 27', 'Hello Michael 38'. Each hello statement should be printed in a new line

In [61]:

```python
############# WRITE THE CODE IN THIS CELL ####################

# creating two lists containing names and ages
names = ['Chienyeze', 'Ada', 'Ogechukwu', 'Chioma', 'Emeka']
ages = [38, 33, 39, 34, 40]

# Iterating through the lists and printing 'Hello Name Age'
for name, age in zip(names, ages):
    print(f'Hello {name} {age}')
```

```
Hello Chienyeze 38
Hello Ada 33
Hello Ogechukwu 39
Hello Chioma 34
Hello Emeka 40
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############
The code received two lists including five names and their ages, and iterated through them using the zip method to print "Hello, 'Name', 'Age'"

TASK 1.5: Write a code to receive an optional text, capitalise all words in the text and print them

In [62]:

```
############# WRITE THE CODE IN THIS CELL ####################
#defining the optional text parameters for words capitalisation
def capitalise_words(text=None):
    if text is None:
        print("No text.")
    elif text.isalpha():
        capitalised_text = text.upper()
        print("Capitalised Words:", capitalised_text)
    else:
        print("numbers cannot be capitalised")

text = input("Enter optional text: ")
capitalise_words(text)
```

```
Enter optional text: chienyeze
Capitalised Words: CHIENYEZE
```

```
############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############

The code defines a function capitalise_words that takes an optional parameter
text and performs capitalisation (text.upper) operations and others based on
the content of text. if text is alpha, it capitalises it, if not alpha, it
return as 'numbers cannot be capitalised' and if no text input, it returns as
'No text'. Using the def, if, elif and else functions, this optional text code
was executed

(66 WORDS)
```

TASK 1.6: Write a function to split the task_dataset in half column-wise and swap the first half and the second half

In [63]:

```
############# WRITE THE CODE IN THIS CELL ###################
#splitting task_dataset into 2 halves and then swapping both halves
def split_column(task_dataset):
    #getthing the half
    half_value = int(len(task_dataset.columns) // 2)
    print(half_value)
    #splitting the halves
    print(task_dataset.iloc[:, half_value:].join(task_dataset.iloc[:, :half_val

split_column(task_dataset)
```

```
7
       relationship   race     sex  capital-gain  capital-loss  \
0              Wife  White  Female             0             0
1           Husband  White    Male             0             0
2              Wife  White  Female             0             0
3           Husband  White    Male             0             0
4           Husband  White    Male             0             0
...             ...    ...     ...           ...           ...
9760      Own-child  White  Female             0             0
9761  Not-in-family  White  Female             0             0
9762        Husband  White    Male             0             0
9763           Wife  White  Female          3103             0
9764      Unmarried  Black  Female             0             0

      hours-per-week native-country  income  age         workclass  fnlwgt  \
0                 40  United-States    >50K   29           Private  216481
1                 45  United-States  <=50K.   36           Private  280570
2                 25  United-States   <=50K   25                 ?  100903
3                 48  United-States   >50K.   47           Private  145636
4                 40  United-States  <=50K.   33           Private  119422
...              ...            ...     ...  ...               ...     ...
9760              30  United-States   <=50K   19           Private   63363
9761              40  United-States  <=50K.   53           Private   58535
9762              40  United-States   <=50K   30           Private  342709
9763              40  United-States   >50K.   41  Self-emp-not-inc  134724
9764              40  United-States   <=50K   21           Private  252253

         education  education-num      marital-status         occupation
0          Masters             14  Married-civ-spouse    Exec-managerial
1     Some-college             10  Married-civ-spouse       Craft-repair
2        Bachelors             13  Married-civ-spouse                  ?
3        Assoc-voc             11  Married-civ-spouse  Handlers-cleaners
4          HS-grad              9  Married-civ-spouse    Exec-managerial
...            ...            ...                 ...                ...
9760  Some-college             10       Never-married              Sales
9761       HS-grad              9            Divorced              Sales
9762       HS-grad              9  Married-civ-spouse  Handlers-cleaners
9763     Assoc-voc             11  Married-civ-spouse      Other-service
9764  Some-college             10       Never-married       Adm-clerical

[9765 rows x 15 columns]
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############

The swap column function was used to swap columns for the task_dataset after splitting it into 2 halfs columns-wise. the swapped halves are the concatenated along the column axis to form the final swap dataset (split_column). the function enable the rearrangement of column data as needed

(46 WORDS)

Task 1.7: Write a function that receives two numerical columns' names and compare their values for all rows. If the value of the first column is greater than the second column, the function should produce True, otherwise, it should produce False. The function should append an additional column to the dataset to store the results of the comparison for all rows. Apply the function to the "age" and "hours-per-week" columns in the task_dataset and print the result.

In [64]:

```
############# WRITE THE CODE IN THIS CELL###################
# Comparing 'age' and 'hours-per-week' columns
def compare_age_hours(col_a, col_b):
    task_dataset['age_vs_hours'] = col_a > col_b
    return task_dataset

# Calling the function to compare 'age' and 'hours-per-week' columns
result_task_dataset = compare_age_hours(task_dataset['age'], task_dataset['hour

# Printing the result
print(result_task_dataset)
```

```
      age         workclass  fnlwgt     education  education-num  \
0      29           Private  216481       Masters             14   
1      36           Private  280570  Some-college             10   
2      25                 ?  100903     Bachelors             13   
3      47           Private  145636     Assoc-voc             11   
4      33           Private  119422       HS-grad              9   
...   ...               ...     ...           ...            ...   
9760   19           Private   63363  Some-college             10   
9761   53           Private   58535       HS-grad              9   
9762   30           Private  342709       HS-grad              9   
9763   41  Self-emp-not-inc  134724     Assoc-voc             11   
9764   21           Private  252253  Some-college             10   

          marital-status         occupation   relationship   race     sex  \
0     Married-civ-spouse    Exec-managerial           Wife  White  Female   
1     Married-civ-spouse       Craft-repair        Husband  White    Male   
2     Married-civ-spouse                  ?           Wife  White  Female   
3     Married-civ-spouse  Handlers-cleaners        Husband  White    Male   
4     Married-civ-spouse    Exec-managerial        Husband  White    Male   
...                  ...                ...            ...    ...     ...   
9760       Never-married              Sales      Own-child  White  Female   
9761            Divorced              Sales  Not-in-family  White  Female   
9762  Married-civ-spouse  Handlers-cleaners        Husband  White    Male   
9763  Married-civ-spouse      Other-service           Wife  White  Female   
9764       Never-married       Adm-clerical      Unmarried  Black  Female   

      capital-gain  capital-loss  hours-per-week native-country  income  \
0                0             0              40  United-States    >50K   
1                0             0              45  United-States  <=50K.   
2                0             0              25  United-States   <=50K   
3                0             0              48  United-States   >50K.   
4                0             0              40  United-States  <=50K.   
...            ...           ...             ...            ...     ...   
9760             0             0              30  United-States   <=50K   
9761             0             0              40  United-States  <=50K.   
9762             0             0              40  United-States   <=50K   
9763          3103             0              40  United-States   >50K.   
9764             0             0              40  United-States   <=50K   

      age_vs_hours  
0            False  
1            False  
2            False  
3            False  
4            False  
...            ...  
9760         False  
9761          True  
9762         False  
9763          True  
9764         False  

[9765 rows x 16 columns]
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############

```
This code receives and compares two numerical columns values (age and hour)
across each row in the task_dataset. Here, the compare_age_hours function
takes a pandas DataFrame, task_dataset, as its input, and for each row in the
DataFrame, the function compares the value in the age column with the value in
the hours-per-week column. It then creates a new column, age_vs_hours, where
each entry is True if the individual's age is greater than the number of hours
they work per week, and False otherwise.

The result_task_dataset then takes the compare_age_hours(task_dataset) as its
input and print it out with the print(result_task_dataset) function

(100 WORDS)
```

Task 1.8: Write a function that returns the names of countries with maximum and minimum average ages in the task_dataset (Hint: you can use the numpy module)

In [66]:

```python
############# WRITE THE CODE IN THIS CELL###################
#Finding the countries with min and max average ages using Group by method
def find_extreme_average_age_countries():

    # Group by 'native-country' and calculate average age
    avg_age_by_country = task_dataset.groupby('native-country')['age'].mean()

    # Finding the country with maximum average age
    max_avg_age_country = avg_age_by_country.idxmax()

    # Find the country with minimum average age
    min_avg_age_country = avg_age_by_country.idxmin()

    return max_avg_age_country, min_avg_age_country

# Call the function and print the results
max_avg_country, min_avg_country = find_extreme_average_age_countries()
print("Country with maximum average age:", max_avg_country)
print("Country with minimum average age:", min_avg_country)
```

```
Country with maximum average age: Hungary
Country with minimum average age: Laos
```

```
############ WRITE EXPLANATIONS HERE (IF APPLICABLE) #############

The codes above seeks to analyze the task_dataset to determine which countries
have the maximum and minimum average age. respectively. The function
calculates the average (mean) age for each country represented in the dataset
by grouping the data by native-country and then applying the mean function to
the age column. This results in a series object, avg_age_by_country, indexed
by country with average ages as values.
```

The country with the maximum average age and minimum average age are then identified using the idxmax() and idxmin() method on the avg_age_by_country.This method returns the index (country name) associated with the maximum and minimum value respectively