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 [2]:
```
############# WRITE THE CODE IN THIS CELL ####################
import pandas as pd
import numpy as np

# importing our datasets from URL
# URL : https://archive.ics.uci.edu/dataset/2/adult
!pip3 install ucimlrepo


# Downloading the dataset from online source
from ucimlrepo import fetch_ucirepo
# generate 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)

#generating a new dataset,reindex, and eleminating null values
#Generating 10,000 dataset at random renamed
task_dataset = data.sample(n=10000, random_state = 45)

#Reindex the dataset
task_dataset.reset_index(drop=True, inplace=True)

#Dropping Null values with effect in the original data
task_dataset.dropna(inplace=True)

#Calling the newly generated dataset
task_dataset
```

```
Requirement already satisfied: ucimlrepo in c:\users\dessy\anaconda3\lib\site-pack
ages (0.0.3)
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race
0	51	Self-emp-not-inc	123011	HS-grad	9	Divorced	Craft-repair	Own-child	White
1	48	Private	148995	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White
2	53	Private	227475	Bachelors	13	Divorced	Sales	Not-in-family	White
3	37	Private	405723	1st-4th	2	Married-civ-spouse	Machine-op-inspct	Husband	White
4	36	Private	228652	Some-college	10	Divorced	Machine-op-inspct	Own-child	Other
...	...	...	...	...	...	...	...	...	...
9995	43	Private	163985	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White
9996	20	Private	194630	HS-grad	9	Married-civ-spouse	Adm-clerical	Husband	White
9997	64	Private	119506	HS-grad	9	Divorced	Other-service	Not-in-family	White
9998	52	Private	284329	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White
9999	33	Private	159888	11th	7	Married-civ-spouse	Craft-repair	Husband	White

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ Using the provided URL for the dataset, i was able to generate some codes and installation package needed First,i imported necessary packeges like pandas and numply, then to installing the ucimlrepo package, and proceeded to importing the dataset into the given code,and import it into pandas datadrame, afterwards i generated a new dataset with n=10,000, for the purpose of data analysis and manipulation we reindex the dataset,to have a clean and more accurate date we dropped the null values, thereby reducing the datset from 10,000 to 9741,and the dataset was renamed to task_dataset.

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 [3]:
```
############# WRITE THE CODE IN THIS CELL ####################
# Grouped by 'occupation' and 'sex', calculate the average capital-gain for each gr
task2  = task_dataset.groupby(['sex', 'occupation'])['capital-gain'].mean().reset_i
```

```
# view the outcome
task2
```

Out[3]:

	sex	occupation	capital-gain
0	Female	?	154.485549
1	Female	Adm-clerical	190.232859
2	Female	Craft-repair	122.929577
3	Female	Exec-managerial	1974.298630
4	Female	Farming-fishing	1118.612903
5	Female	Handlers-cleaners	123.000000
6	Female	Machine-op-inspct	120.167568
7	Female	Other-service	111.207721
8	Female	Priv-house-serv	600.000000
9	Female	Prof-specialty	1375.822176
10	Female	Protective-serv	103.821429
11	Female	Sales	271.000000
12	Female	Tech-support	567.382353
13	Female	Transport-moving	0.000000
14	Male	?	1234.255924
15	Male	Adm-clerical	636.471033
16	Male	Armed-Forces	0.000000
17	Male	Craft-repair	657.704225
18	Male	Exec-managerial	2288.728423
19	Male	Farming-fishing	296.575092
20	Male	Handlers-cleaners	244.650667
21	Male	Machine-op-inspct	334.439024
22	Male	Other-service	245.292035
23	Male	Priv-house-serv	118.800000
24	Male	Prof-specialty	3609.882965
25	Male	Protective-serv	233.114650
26	Male	Sales	1453.350140
27	Male	Tech-support	450.727273
28	Male	Transport-moving	437.317568

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ The above command was used t0 analyze the average capital gain across different occupations contributed by sex,The average capital gain varies across occupation categories for both female and male. we have 13 different occupations and the above analysis shows the male and female contribution,For instance Exec-managerial as an occupation have an average capital-gain of 4,263.02705, which female contribute 1974.298630 while the male contribute 2288.728423. We also noticed some occupations with 0 capital-gain Female : Transport-moving occupation has 0 capital-gain Male : Armed-Forces occupation has 0 capital-gain The reasons for a 0 average capital gain in some occupations categories can vary greatly and may be influenced by a variety of factors such as the

nature of the occupation, individual preferences, economic conditions, and regulatory environment. In essence the above analysis was done using the groupby function to filter the data based on the sex,occupation and capital-gain.

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 [4]:
```
############ WRITE THE CODE IN THIS CELL ####################

# Group by 'country' and 'education', count the number of people in each group
education_counts = task_dataset.groupby(['native-country', 'education']).size().res

# bachelors degree count in education
degree_counts = education_counts[education_counts['education'] == 'Bachelors']

# Find the country with the highest number of people with a Bachelors degree
highestcountry = degree_counts.loc[degree_counts['count'].idxmax()]

# Display the result
highestcountry
```

Out[4]:
```
native-country    United-States
education             Bachelors
count                      1404
Name: 266, dtype: object
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ United-states has the highest number of individuals with Bcahelors degree with about 1404 counts.

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 [5]:
```
#Highlight the list by names and ages
Names = ['Frank', 'Johnson','Khadijat','Yetunde','Shittu']
Age = [20,30,40,50,60]

#iterating around the list and print
for Name,Age in zip (Names,Age):
    print(f'Hello {Name } {Age}')
```

```
Hello Frank 20
Hello Johnson 30
Hello Khadijat 40
Hello Yetunde 50
Hello Shittu 60
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ The above code has been used to create a two lists , with the fisrt list as names and second as age respectivelty. Each line of output

contains a greeting message for a specific person, including their name and age. The loop iterates through each pair of name and age from the Names and Age lists, printing the appropriate greeting message.

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

In [6]:
```
############ WRITE THE CODE IN THIS CELL ####################
def capitalize_all_words(text=None):
    if text is not None:
        outcome = text.title()
        print(outcome)

capitalize_all_words('good morning khadijat,hope you had a great weekend?')
```

```
Good Morning Khadijat,Hope You Had A Great Weekend?
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ Explanation for Output: The function capitalises the first letter of each word in the input text, giving the following capitalised output: "Good" (The first letter of "good" is capitalised) "Morning" (the first letter is capitalised) "Khadijat" (The first letter of "khadijat" is capitalised). "Hope" (The first letter of "hope" is capitalised) "You" (The first letter of "you" is capitalised) "Had" (the first letter is capitalised) "A" (The first letter of "a" is capitalised) "Great" (The first letter of "great" is capitalised) "Weekend" (The first letter of "weekend" is capitalised) Each word in the input text is appropriately capitalised, and the entire capitalised text is printed as the output.

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 [47]:
```
############ WRITE THE CODE IN THIS CELL ####################
#Calculating the total columns the dataset and defining the data column-wise
def split_and_swap(task_dataset):
    column_wise = len(task_dataset.columns)
    midpoints=column_wise // 2

    column1 = task_dataset.iloc[:, :midpoints]
    column2 = task_dataset.iloc[:, midpoint:]


#Swap both halves
    swap = pd.concat([column2,column1],axis=1)

    print("\n SWAPPED OUTCOME")
    print(swap)
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ Explanation for the code: Column_wise = len(task_dataset.columns) This line computes the total number of columns in the dataset (task_dataset) and assigns it to the variable column-wise. midpoint = column_wise // 2: This line computes the column midpoint by dividing the total number of columns (column_wise) by two using integer division (//). This midpoint will be used to divide the columns into two sections. Column 1 = task_dataset.iloc[:,:midpoint]: This line uses integer-location indexing (iloc) to extract the first half of the columns from the dataset (task_dataset). It selects all rows (:) and columns from index 0 to the midpoint (midpoint) minus 1. Column 2 = task_dataset.iloc[:, midpoint:]: This line extracts the second half of the dataset's columns (task_dataset). It selects all rows (:) and columns from the centre (midpoint) to the end.

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 [48]:

```
############# WRITE THE CODE IN THIS CELL####################
def compare_columns (df, coll, col2):
    """
    Compare values of two numerical columns in a DataFrame.
    If value in coll is greater than value in col2, return True, otherwise False.
    Append the result as a new column to the DataFrame.
    """
# Perform comparison and create new column with the results
    df[' comparison_result'] = df[coll] > df [col2]

    return df
# Apply the function to the task dataset
result_dataset = compare_columns (task_dataset, 'age', 'hours-per-week')
# Print the result
print (result_dataset)
```

```
      age         workclass  fnlwgt     education  education-num  \
0      51  Self-emp-not-inc  123011       HS-grad              9
1      48           Private  148995       HS-grad              9
2      53           Private  227475     Bachelors             13
3      37           Private  405723       1st-4th              2
4      36           Private  228652  Some-college             10
...   ...               ...     ...           ...            ...
9995   43           Private  163985       HS-grad              9
9996   20           Private  194630       HS-grad              9
9997   64           Private  119506       HS-grad              9
9998   52           Private  284329  Some-college             10
9999   33           Private  159888          11th              7

          marital-status         occupation   relationship   race     sex  \
0               Divorced       Craft-repair      Own-child  White    Male
1                Widowed    Exec-managerial  Not-in-family  White    Male
2               Divorced              Sales  Not-in-family  White  Female
3     Married-civ-spouse  Machine-op-inspct        Husband  White    Male
4               Divorced  Machine-op-inspct      Own-child  Other    Male
...                  ...                ...            ...    ...     ...
9995  Married-civ-spouse       Craft-repair        Husband  White    Male
9996  Married-civ-spouse       Adm-clerical        Husband  White    Male
9997           Divorced      Other-service  Not-in-family  White  Female
9998  Married-civ-spouse       Craft-repair        Husband  White    Male
9999  Married-civ-spouse       Craft-repair        Husband  White    Male

      capital-gain  capital-loss  hours-per-week native-country  income  \
0                0             0              40  United-States  <=50K.
1            14084             0              45  United-States   >50K.
2                0             0              40  United-States  <=50K.
3                0             0              40         Mexico   <=50K
4                0             0              40         Mexico  <=50K.
...            ...           ...             ...            ...     ...
9995             0             0              40  United-States   <=50K
9996          3781             0              50  United-States   <=50K
9997             0             0              15  United-States   <=50K
9998             0             0              45  United-States    >50K
9999             0             0              40  United-States   <=50K

       comparison_result
0                   True
1                   True
2                   True
3                  False
4                  False
...                  ...
9995                True
9996               False
9997                True
9998                True
9999               False

[9741 rows x 16 columns]
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ This above code compares the values of two numerical columns in the task-dataset, with numerical columns ('age' and 'hours-per-week') and adds the result as a new column called 'comparison_result' to the DataFrame. Finally, it prints the DataFrame containing the comparison results. In essence The Row 0 of the Age = 51 while hours-per-week = 40 51 is > 40 Making the comparison_result for Row 0 = True, according to the comparison command.

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 [49]:
```
############# WRITE THE CODE IN THIS CELL####################
def countries_with_maximum_minimum_average_age(task_dataset):

    # Group the task_dataset by country and compute the average age for each one.
    average = task_dataset.groupby('native-country')['age'].mean()

    # Find the country that has the highest average age.
    MAX = average.idxmax()

    # Find the country with the lowest average age.
    MIN = average.idxmin()

    return MAX, MIN

# using the task_ dataset verify the function
MAX, MIN = countries_with_maximum_minimum_average_age(task_dataset)
print("Country with minimum average age:" , MIN)
print("Country with maximum average age:", MAX)
```

```
Country with minimum average age: Vietnam
Country with maximum average age: Outlying-US(Guam-USVI-etc)
```

############ WRITE EXPLANATIONS HERE (IF APPLICABLE) ############ The output shows that VIETMAN has the minimum avaerage age while Outlying-US has the maximum avaerage age