Midterm Skills Exam: Data Wrangling and Analysis

Name: Calingo, Christian Lei S.

Section: CPE22S3

Course: Computational Thinking with Python

Course Code: CPE 311

Importing the dataset

```
pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-packages (0.0.6)
```

```
from ucimlrepo import fetch_ucirepo

census_income = fetch_ucirepo(id=20)

# data (as pandas dataframes)
X = census_income.data.features
y = census_income.data.targets

print(census_income.metadata)

print(census_income.variables)
```

```
{'uci_id': 20, 'name': 'Census Income', 'repository_url': 'https://archive.ics.uci.edu/dataset/20/census+income', 'data_url': 'https://a
              name     role         type      demographic  \
0              age  Feature      Integer              Age
1        workclass  Feature  Categorical           Income
2           fnlwgt  Feature      Integer             None
3        education  Feature  Categorical  Education Level
4    education-num  Feature      Integer  Education Level
5   marital-status  Feature  Categorical            Other
6       occupation  Feature  Categorical            Other
7     relationship  Feature  Categorical            Other
8             race  Feature  Categorical             Race
9              sex  Feature       Binary              Sex
10    capital-gain  Feature      Integer             None
11    capital-loss  Feature      Integer             None
12  hours-per-week  Feature      Integer             None
13  native-country  Feature  Categorical            Other
14          income   Target       Binary           Income

                                      description units missing_values
0                                             N/A  None             no
1   Private, Self-emp-not-inc, Self-emp-inc, Feder...  None            yes
2                                            None  None             no
3    Bachelors, Some-college, 11th, HS-grad, Prof-...  None             no
4                                            None  None             no
5   Married-civ-spouse, Divorced, Never-married, S...  None             no
6   Tech-support, Craft-repair, Other-service, Sal...  None            yes
7   Wife, Own-child, Husband, Not-in-family, Other...  None             no
8   White, Asian-Pac-Islander, Amer-Indian-Eskimo,...  None             no
9                                   Female, Male.  None             no
10                                           None  None             no
11                                           None  None             no
12                                           None  None             no
13  United-States, Cambodia, England, Puerto-Rico,...  None            yes
14                                  >50K, <=50K.  None             no
```

Displaying the datasets

```
X
```

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

```
y
```

	income
0	<=50K
1	<=50K
2	<=50K
3	<=50K
4	<=50K
...	...
48837	<=50K.
48838	<=50K.
48839	<=50K.
48840	<=50K.
48841	>50K.

48842 rows × 1 columns

Concatenating the 2 datasets

```
import pandas as pd
import numpy as np

data = pd.concat([X, y], axis = 1) #concatenating our data

data
```

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

Checking the data types of the column in the dataset

```
data.dtypes #checking their data types
```

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

Let's check the info() of our dataframe

```
data.info() #checking their info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             48842 non-null  int64
 1   workclass       47879 non-null  object
 2   fnlwgt          48842 non-null  int64
 3   education       48842 non-null  object
 4   education-num   48842 non-null  int64
 5   marital-status  48842 non-null  object
 6   occupation      47876 non-null  object
 7   relationship    48842 non-null  object
 8   race            48842 non-null  object
 9   sex             48842 non-null  object
 10  capital-gain    48842 non-null  int64
 11  capital-loss    48842 non-null  int64
 12  hours-per-week  48842 non-null  int64
 13  native-country  48568 non-null  object
 14  income          48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

Let's check the values in one of the columns

```
data['native-country'].unique() #checking for the unique values of country
```

```
array(['United-States', 'Cuba', 'Jamaica', 'India', '?', 'Mexico',
       'South', 'Puerto-Rico', 'Honduras', 'England', 'Canada', 'Germany',
       'Iran', 'Philippines', 'Italy', 'Poland', 'Columbia', 'Cambodia',
       'Thailand', 'Ecuador', 'Laos', 'Taiwan', 'Haiti', 'Portugal',
       'Dominican-Republic', 'El-Salvador', 'France', 'Guatemala',
       'China', 'Japan', 'Yugoslavia', 'Peru',
       'Outlying-US(Guam-USVI-etc)', 'Scotland', 'Trinadad&Tobago',
       'Greece', 'Nicaragua', 'Vietnam', 'Hong', 'Ireland', 'Hungary',
       'Holand-Netherlands', nan], dtype=object)
```

As you can see, we have a ? value for the native country, other columns might have ? values as well, we can change those values to Others so that other data will not be wasted

```
replace_nan = {"?" : "Others"}

data.replace(replace_nan, inplace = True) #changing the values of the unidentified
data.fillna('Others', inplace = True)
data
```

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

Now, we wil check if we have replaced it properly

```
data.isna().any()
```

```
age              False
workclass        False
fnlwgt           False
education        False
education-num    False
marital-status   False
occupation       False
relationship     False
race             False
sex              False
capital-gain     False
capital-loss     False
hours-per-week   False
native-country   False
income           False
dtype: bool
```

As you can see, each column returned False, therefore, we don't have NaN values anymore

---

Now, we are going to set the country as its index

```
data.rename(columns = {'native-country' : 'country'}, inplace = True)
data.set_index(data['country'],inplace = True) #setting the country as index
data.drop('country', axis = 1, inplace = True)
data
```

country	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationsh
United-States	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-fam
United-States	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husba
United-States	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-fam
United-States	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husba
Cuba	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	W
...	...	...	...	...	...	...	...	
United-States	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-fam
United-States	64	Others	321403	HS-grad	9	Widowed	Others	Other-relati
United-States	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husba
United-	44	Private	83891	Bachelors	13	Divorced	Adm-	Own-ch

At the income column, we can see that we have 4 unique values:

- (<=50K)
- (<= 50K.)
- (>50K)
- (>50K.)

We are going to remove the decimal after the 'K'

```
data['income'].unique() #checking for the unique values of income
```

```
array(['<=50K', '>50K', '<=50K.', '>50K.'], dtype=object)
```

```
row_change = {'<=50K.' : '<=50K', #changing the value because of the unecessary .
              '>50K.' : '>50K'}

data = data.replace({'income' : row_change})
data
```

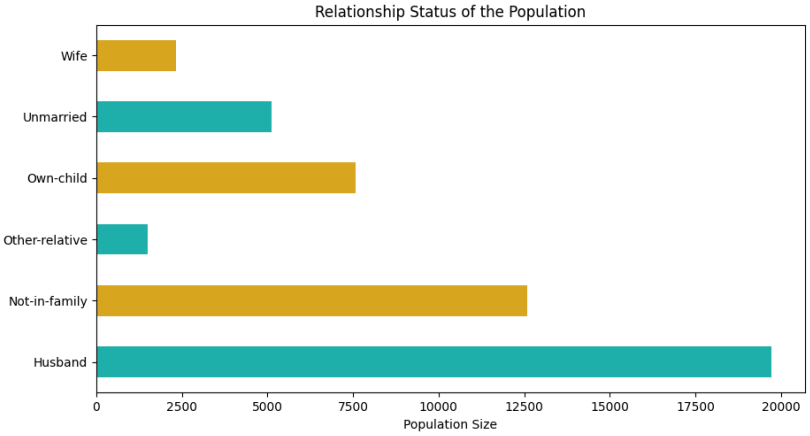
country	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationsh
United-States	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-fam
United-States	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husba
United-States	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-fam
United-States	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husba
Cuba	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	W
...	...	...	...	...	...	...	...	
United-States	39	Private	215419	Bachelors	13	Divorced	Prof-specialty	Not-in-fam
United-States	64	Others	321403	HS-grad	9	Widowed	Others	Other-relati
United-States	38	Private	374983	Bachelors	13	Married-civ-spouse	Prof-specialty	Husba
United-	44	Private	83891	Bachelors	13	Divorced	Adm-	Own-ch

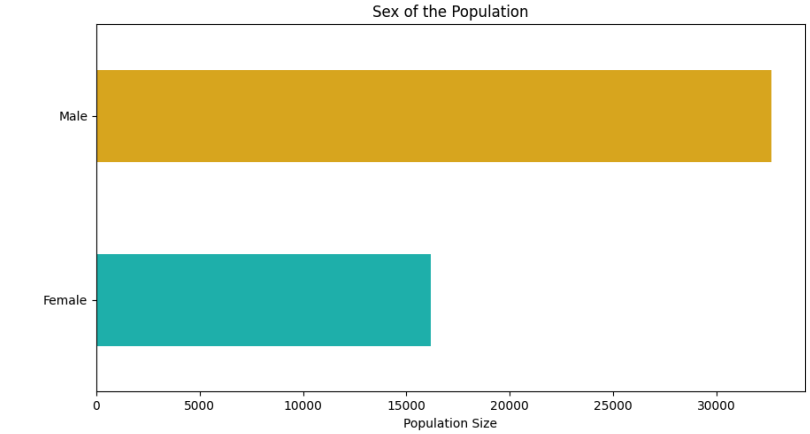
After cleaning the Data, we can now perform Data Analysis and Data Exploratory

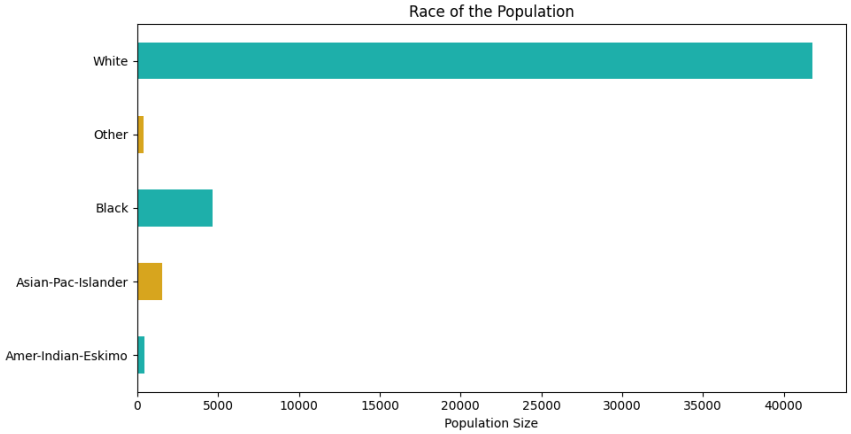
Let's plot the Marital Status, Relationship, Sex, Race, and Education of our Population to identify their quantity.

Since we won't be needing them on our Analysis, we will be dropping them except for the Education

```
import matplotlib.pyplot as plt #plotting the marital-status, relationship, sex, and race
import seaborn as sns

fig, ax = plt.subplots(4, figsize = [10,20])

data.groupby('marital-status').size().plot(kind='barh', ax = ax[0], color = ('lightseagreen','goldenrod')) #plotting the marital-status colu
ax[0].set_title('Population of Marital Status')
ax[0].set_xlabel('Population Size')
ax[0].set_ylabel('')

data.groupby('relationship').size().plot(kind='barh', ax = ax[1], color = ('lightseagreen','goldenrod')) #plotting the relationship column
ax[1].set_title('Relationship Status of the Population')
ax[1].set_xlabel('Population Size')
ax[1].set_ylabel('')

data.groupby('sex').size().plot(kind='barh', ax = ax[2], color = ('lightseagreen','goldenrod')) #plotting the sex column
ax[2].set_title('Sex of the Population')
ax[2].set_xlabel('Population Size')
ax[2].set_ylabel('')

data.groupby('race').size().plot(kind='barh', ax = ax[3], color = ('lightseagreen','goldenrod')) #plotting the race column
ax[3].set_title('Race of the Population')
ax[3].set_xlabel('Population Size')
ax[3].set_ylabel('')

fig.tight_layout()
```


Relationship Status of the Population
Wife
Unmarried
Own-child
Other-relative
Not-in-family
Husband
0
2500
5000
7500
10000
12500
15000
17500
20000
Population Size


Sex of the Population
Male
Female
0
5000
10000
15000
20000
25000
30000
Population Size

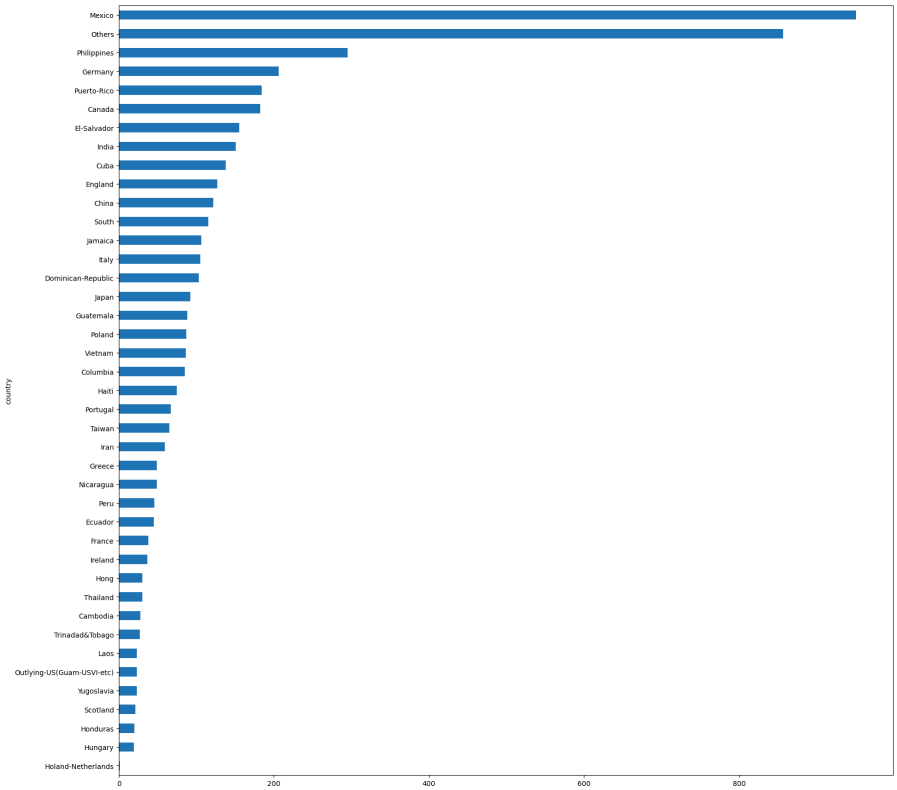

Race of the Population
White
Other
Black
Asian-Pac-Islander
Amer-Indian-Eskimo
0
5000
10000
15000
20000
25000
30000
35000
40000
Population Size

Let's now check the graph of our population. However, since United Scales outscales everyone, we are temporarily dropping it.

```
data_popu = data.groupby('country').size().sort_values(ascending = False)
data_popu = data_popu.head()
data_popu
```

```
country
United-States    43832
Mexico             951
Others             857
Philippines        295
Germany            206
dtype: int64
```

```
data_popu = data.reset_index()
data_popu = data.drop('United-States')
data_popu.groupby('country').size().sort_values(ascending = True).plot(kind = 'barh', figsize = (20,20))
fig.tight_layout()
```



Most of our Population comes from the United-States, followed by Mexico and Others which is unspecified countries

---

Let's now drop the unecesarry columns

```
data.drop(['marital-status', 'relationship', 'race', 'sex'], axis = 1, inplace = True) #dropping the unecessary column for the analysis
data
```

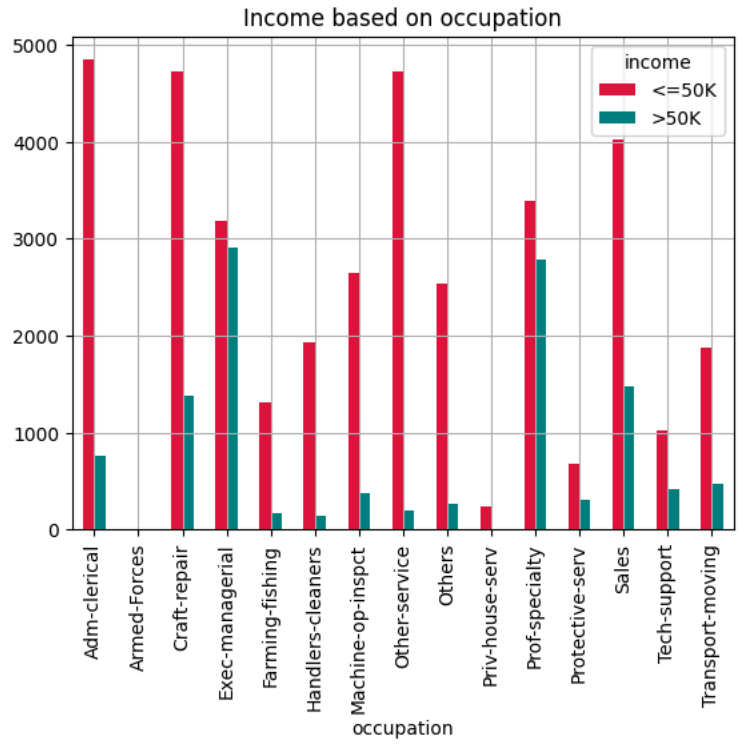
country	age	workclass	fnlwgt	education	education-num	occupation	capital-gain	capital-loss
United-States	39	State-gov	77516	Bachelors	13	Adm-clerical	2174	0
United-States	50	Self-emp-not-inc	83311	Bachelors	13	Exec-managerial	0	0
United-States	38	Private	215646	HS-grad	9	Handlers-cleaners	0	0
United-States	53	Private	234721	11th	7	Handlers-cleaners	0	0
Cuba	28	Private	338409	Bachelors	13	Prof-specialty	0	0
...	...	...	...	...	...	...	...	...
United-States	39	Private	215419	Bachelors	13	Prof-specialty	0	0

After dropping unecessary columns

We will check how many people earns above or below 50k according to their occupation

```
color_list = ['aqua', 'teal'] #list of colors

data1=data.groupby(['occupation','income']).size() #grouping our data
data1=data1.unstack() #putting it in a dataframe
data1.plot(kind='bar', grid = True, title ='Income based on occupation', color =('crimson','teal')) #plotting the income of our population w
```

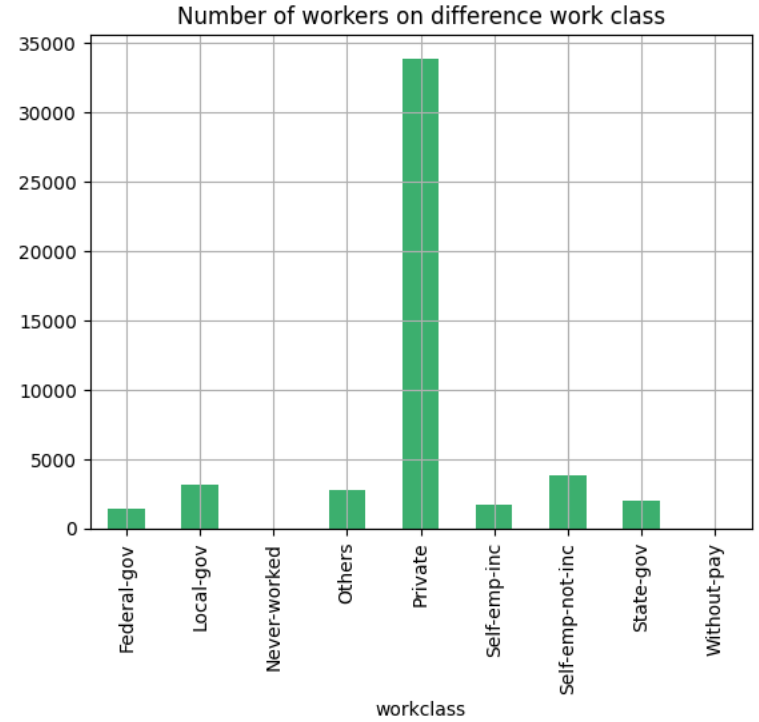
```
<Axes: title={'center': 'Income based on occupation'}, xlabel='occupation'>
```



Let's identify the work class of our data, let's see how many people does work on a certain work class

```
data2 = data.groupby(['workclass']).size() #grouping our population on their work class
data2.plot(kind = 'bar', grid = True, title = 'Number of workers on difference work class', color = 'mediumseagreen') #identifying the quant
                                                                                                                      #work class
```

```
<Axes: title={'center': 'Number of workers on difference work class'},
xlabel='workclass'>
```



Looking at our graph, we can see that there is a lot of people who does work in a Private Sector.

The "Never-Worked" and "Without-pay" have values, it's just that the Private sector outscales them making it look like it doesn't have value.
We can check the number of it by calling our data.

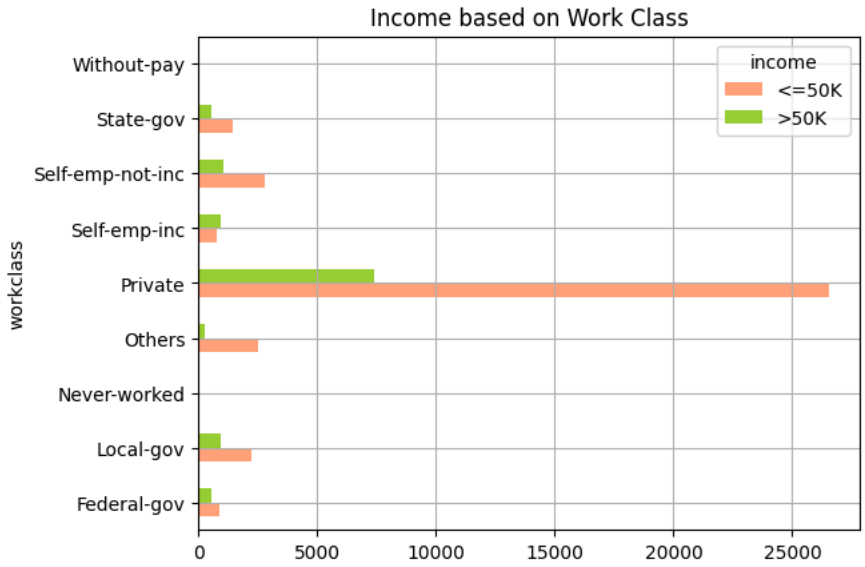
```
data2 #counts of the workclass
```

```
workclass
Federal-gov          1432
Local-gov            3136
Never-worked           10
Others               2799
Private             33906
Self-emp-inc         1695
Self-emp-not-inc     3862
State-gov            1981
Without-pay            21
dtype: int64
```

We can also check how many earn above or below 50,000 depending on which class they work for

```
data3 = data.groupby(['workclass','income']).size() #grouping the population according to their workclass and income
data3=data3.unstack() #putting it in a dataframe
data3.plot(kind = 'barh', grid = True, title = 'Income based on Work Class', color = ('lightsalmon', 'yellowgreen')) #graphing the income of
```

```
<Axes: title={'center': 'Income based on Work Class'}, ylabel='workclass'>
```




```
data3 #checking the values of data3
```

income	<=50K	>50K
workclass		
Federal-gov	871.0	561.0
Local-gov	2209.0	927.0
Never-worked	10.0	NaN
Others	2534.0	265.0
Private	26519.0	7387.0
Self-emp-inc	757.0	938.0
Self-emp-not-inc	2785.0	1077.0
State-gov	1451.0	530.0
Without-pay	19.0	2.0

Looking at our graph, we can see that majority of our population works in a private sector.

If we look at the bar of Self-employed-inc., those who earned above 50,000 is greater than for those who earn below it.

Let's identify the educational attainment for each occupation of our population

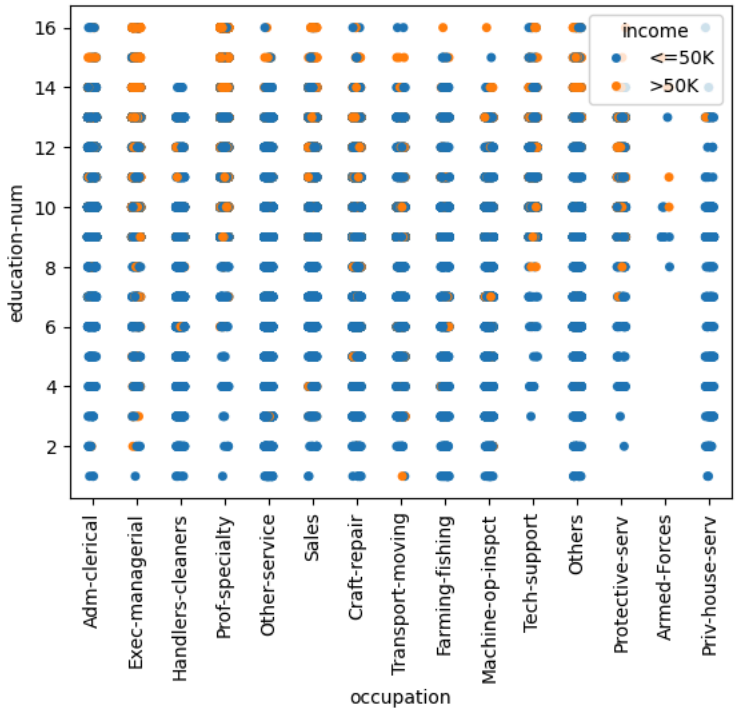
```
data_4 = data[['education-num', 'occupation']]
data_4.sort_values(by = 'education-num')
```

country	education-num	occupation
Mexico	1	Other-service
United-States	1	Exec-managerial
Philippines	1	Priv-house-serv
El-Salvador	1	Other-service
United-States	1	Adm-clerical
...	...	...
United-States	16	Prof-specialty
Mexico	16	Prof-specialty
United-States	16	Prof-specialty
United-States	16	Exec-managerial
United-States	16	Prof-specialty

48842 rows × 2 columns

```
sns.stripplot(
x='occupation',
y='education-num',
hue='income',
data=data
)
plt.xticks(rotation=90)
```

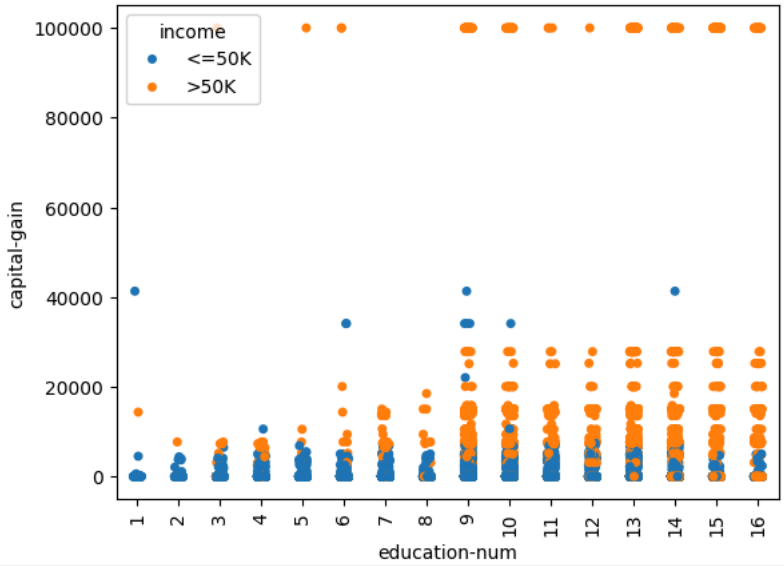
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14],
 [Text(0, 0, 'Adm-clerical'),
  Text(1, 0, 'Exec-managerial'),
  Text(2, 0, 'Handlers-cleaners'),
  Text(3, 0, 'Prof-specialty'),
  Text(4, 0, 'Other-service'),
  Text(5, 0, 'Sales'),
  Text(6, 0, 'Craft-repair'),
  Text(7, 0, 'Transport-moving'),
  Text(8, 0, 'Farming-fishing'),
  Text(9, 0, 'Machine-op-inspct'),
  Text(10, 0, 'Tech-support'),
  Text(11, 0, 'Others'),
  Text(12, 0, 'Protective-serv'),
  Text(13, 0, 'Armed-Forces'),
  Text(14, 0, 'Priv-house-serv')])
```



Let's try plotting the income of our population with accordance to their capital gain and their education attainment level

```
sns.stripplot( #plotting the the income of our population based on their educational attainment
x='education-num',
y='capital-gain',
hue='income',
data=data
)
plt.xticks(rotation=90)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15],
 [Text(0, 0, '1'),
  Text(1, 0, '2'),
  Text(2, 0, '3'),
  Text(3, 0, '4'),
  Text(4, 0, '5'),
  Text(5, 0, '6'),
  Text(6, 0, '7'),
  Text(7, 0, '8'),
  Text(8, 0, '9'),
  Text(9, 0, '10'),
  Text(10, 0, '11'),
  Text(11, 0, '12'),
  Text(12, 0, '13'),
  Text(13, 0, '14'),
  Text(14, 0, '15'),
  Text(15, 0, '16')])
/usr/local/lib/python3.10/dist-packages/IPython/core/events.py:89: UserWarning: Creating
  func(*args, **kwargs)
```



Here, we have the average capital-gain and capital-loss of each country

```
data_cap = data[['capital-gain', 'capital-loss', 'hours-per-week']]
data_cap = data_cap.groupby('country').mean()
data_cap.sort_values(by = 'country', ascending = True, inplace = True)
data_cap
```