Midterm Skills Exam: Data Wrangling and Analysis

In this activity, you are expected to demonstrate skills learned from concluded modules. Specifically:

- Analyze data using tools such as numpy and pandas for data wrangling tasks;
- Visualize data using pandas and seaborn;
- Perform exploratory data analysis on a complex dataset.

Resources:

- Jupyter Lab / Notebook
- Dataset: https://archive-beta.ics.uci.edu/dataset/20/census+income

Submission Requirements:

- Perform data wrangling on the given dataset.
- Visualize the given dataset.
- Submit pdf of exploratory data analysis.
- Submit pdf of EDA presentation. Sample: https://aseandse.org/asean-dse-storyboard/

Data Wrangling

```
!pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2.0.3)
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2024.6.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0->ucimlrepo) (1.25.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlrep
```

```
# Import the dataset into your code
from ucimlrepo import fetch_ucirepo

# fetch dataset
census_income = fetch_ucirepo(id=20)

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

# metadata
print(census_income.metadata)

# variable information
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
```

```
# append together into a single dataframe

import pandas as pd
import numpy as np

# concatenate the two DataFrames
census_data = pd.concat([X, y], axis=1)
census_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-	Not-in-family

Next steps: Generate code with `census_data` View recommended plots

```
# display dataset X

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-	Not-in-family

Next steps: Generate code with `X` View recommended plots

```
# display dataset y

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

Next steps: Generate code with y | View recommended plots

```
# check datatypes of the dataframe

census_data.dtypes
```

```
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
```

```
# generate descriptive statistics

census_data.describe()
```

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000

```
# check the columns

census_data.columns
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
```

```
       'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
       'income'],
      dtype='object')
```

```
# check the info of dataframe

census_data.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
```

```
# check the unique values of the workclass

census_data['workclass'].unique()
```

```
array(['State-gov', 'Self-emp-not-inc', 'Private', 'Federal-gov',
       'Local-gov', '?', 'Self-emp-inc', 'Without-pay', 'Never-worked',
       nan], dtype=object)
```

```
# check the unique values of the workclass

census_data['native-country'].unique()
```

```
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)
```

```
# check the unique values of the occupation

census_data['occupation'].unique()
```

```
array(['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners',
       'Prof-specialty', 'Other-service', 'Sales', 'Craft-repair',
       'Transport-moving', 'Farming-fishing', 'Machine-op-inspct',
       'Tech-support', '?', 'Protective-serv', 'Armed-Forces',
       'Priv-house-serv', nan], dtype=object)
```

```
# make the missing values into others

census_data = census_data.replace('?', np.nan)
census_data = census_data.fillna('Others')
```

```
census_data['workclass'].unique()
```

```
array(['State-gov', 'Self-emp-not-inc', 'Private', 'Federal-gov',
       'Local-gov', 'Others', 'Self-emp-inc', 'Without-pay',
       'Never-worked'], dtype=object)
```

```
census_data = census_data.replace('?', np.nan)
census_data = census_data.fillna('Others')

census_data['native-country'].unique()
```

```
array(['United-States', 'Cuba', 'Jamaica', 'India', 'Others', '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'], dtype=object)
```

```
census_data = census_data.replace('?', np.nan)
census_data = census_data.fillna('Others')

census_data['occupation'].unique()
```

```
array(['Adm-clerical', 'Exec-managerial', 'Handlers-cleaners',
       'Prof-specialty', 'Other-service', 'Sales', 'Craft-repair',
       'Transport-moving', 'Farming-fishing', 'Machine-op-inspct',
       'Tech-support', 'Others', 'Protective-serv', 'Armed-Forces',
       'Priv-house-serv'], dtype=object)
```

```
# check properly if the missing values has been replaced properly

census_data.isnull().sum()
```

```
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
```

```
# drop the duplicates in the dataframe

census_data.drop_duplicates(inplace=True)
census_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-	Not-in-family

Next steps: Generate code with census_data View recommended plots

```
# check the number of duplicates if there are still

num_duplicates = census_data.duplicated().sum()
print(f"Number of duplicates: {num_duplicates}")
```

```
Number of duplicates: 0
```

```
# check unique values of the income

census_data['income'].unique()
```

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

```
# removing the . after the K

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

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

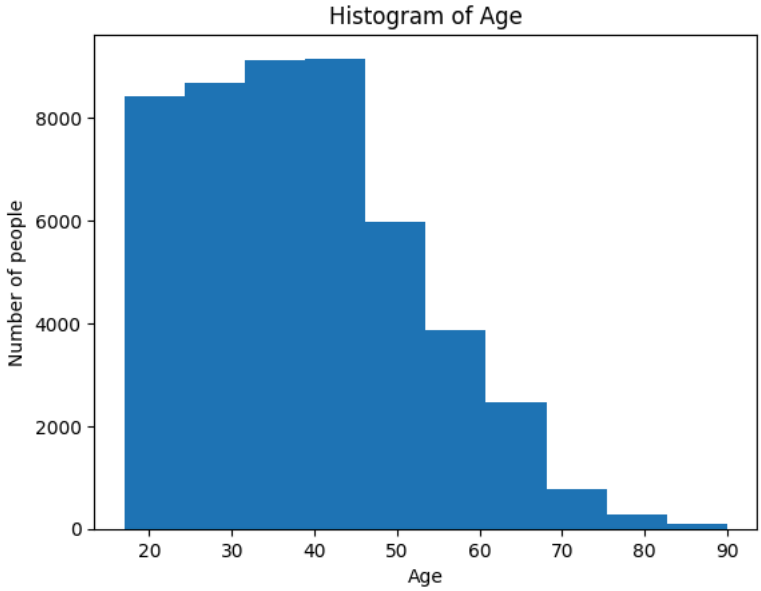
	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family
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-	Not-in-family

Next steps: Generate code with `data` View recommended plots

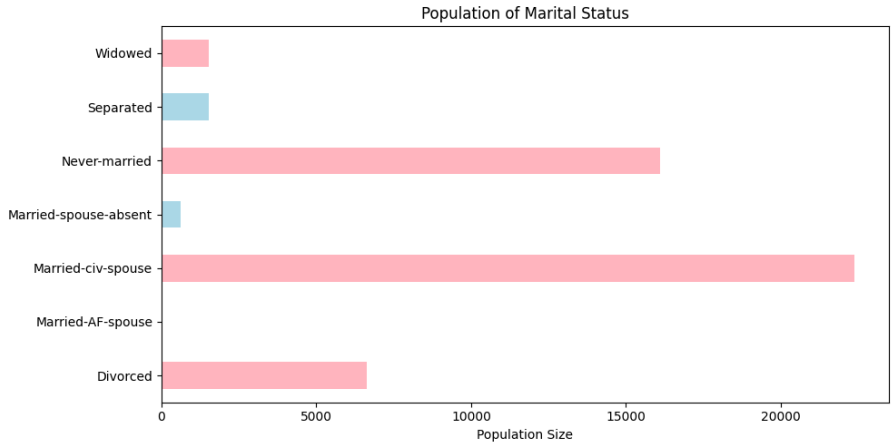
Data Analysis and Data Visualization:

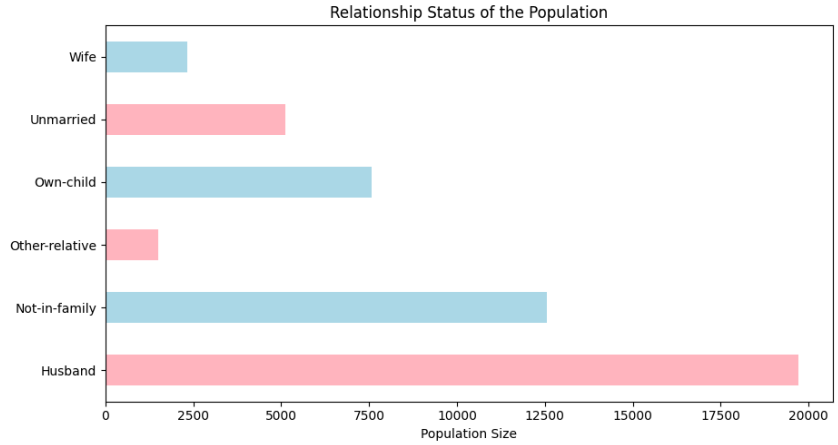
```
# create a histogram of age

import matplotlib.pyplot as plt

plt.hist(data["age"])

plt.xlabel("Age")
plt.ylabel("Number of people")

plt.title("Histogram of Age")
```

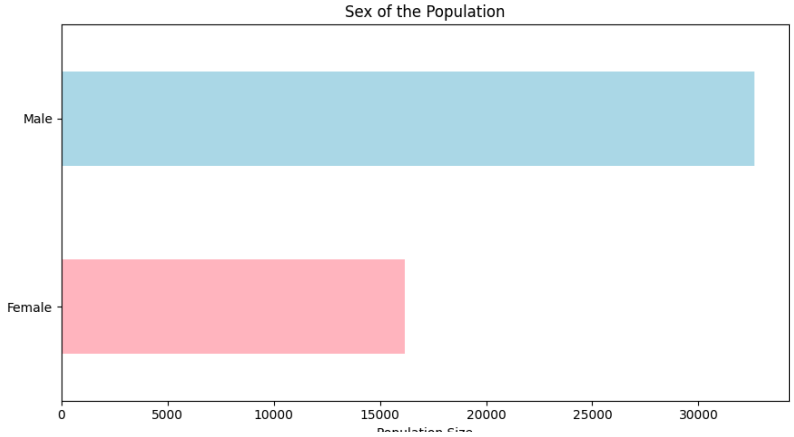
```
Text(0.5, 1.0, 'Histogram of Age')
```




```
# sub-plotting the marital-status, relationship, and sex

import matplotlib.pyplot as plt
import seaborn as sns

fig, ax = plt.subplots(3, figsize = [10,15])

data.groupby('marital-status').size().plot(kind='barh', ax = ax[0], color = ('lightpink','lightblue'))
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 = ('lightpink','lightblue'))
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 = ('lightpink','lightblue'))
ax[2].set_title('Sex of the Population')
ax[2].set_xlabel('Population Size')
ax[2].set_ylabel('')

fig.tight_layout()
```

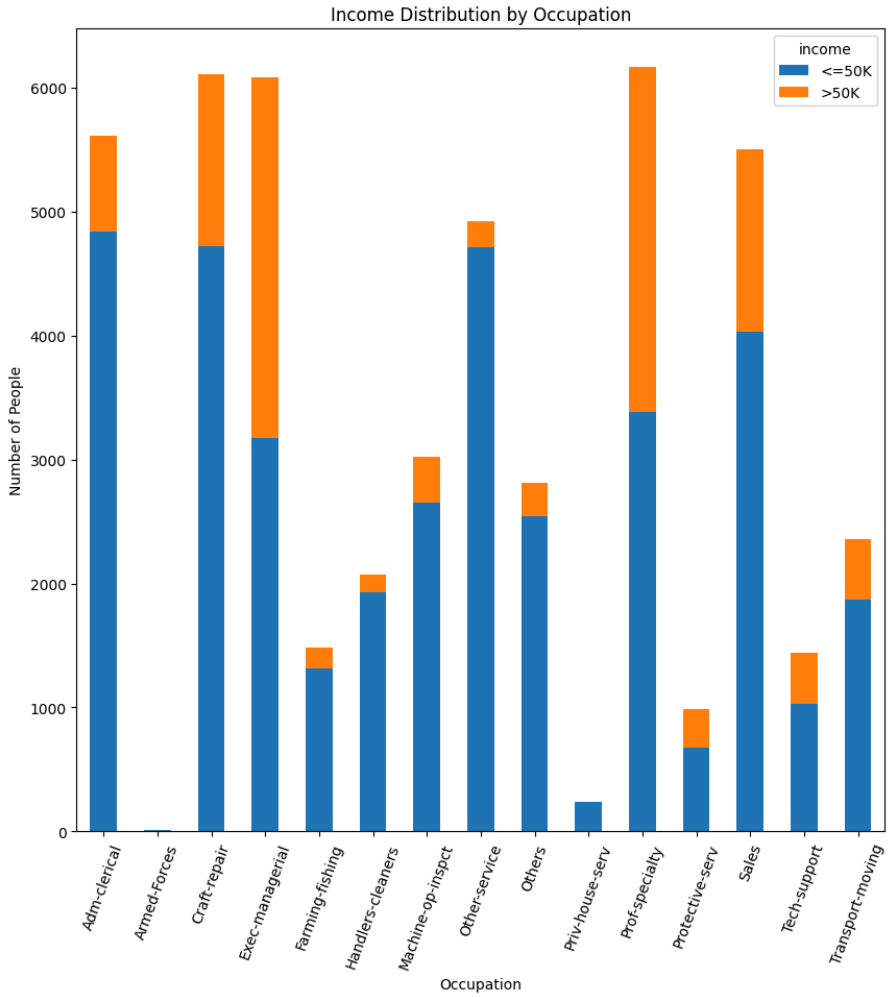

Population of Marital Status
Widowed
Separated
Never-married
Married-spouse-absent
Married-civ-spouse
Married-AF-spouse
Divorced
0
5000
10000
15000
20000
Population Size


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

```
# bar chart of income based on occupation

import matplotlib.pyplot as plt

grouped_data = data.groupby(['occupation', 'income']).size().unstack()

fig, ax = plt.subplots(figsize=(10, 10))
grouped_data.plot(kind='bar', stacked=True, ax=ax)

ax.set_title('Income Distribution by Occupation')
ax.set_xlabel('Occupation')
ax.set_ylabel('Number of People')

plt.xticks(rotation=70)
```

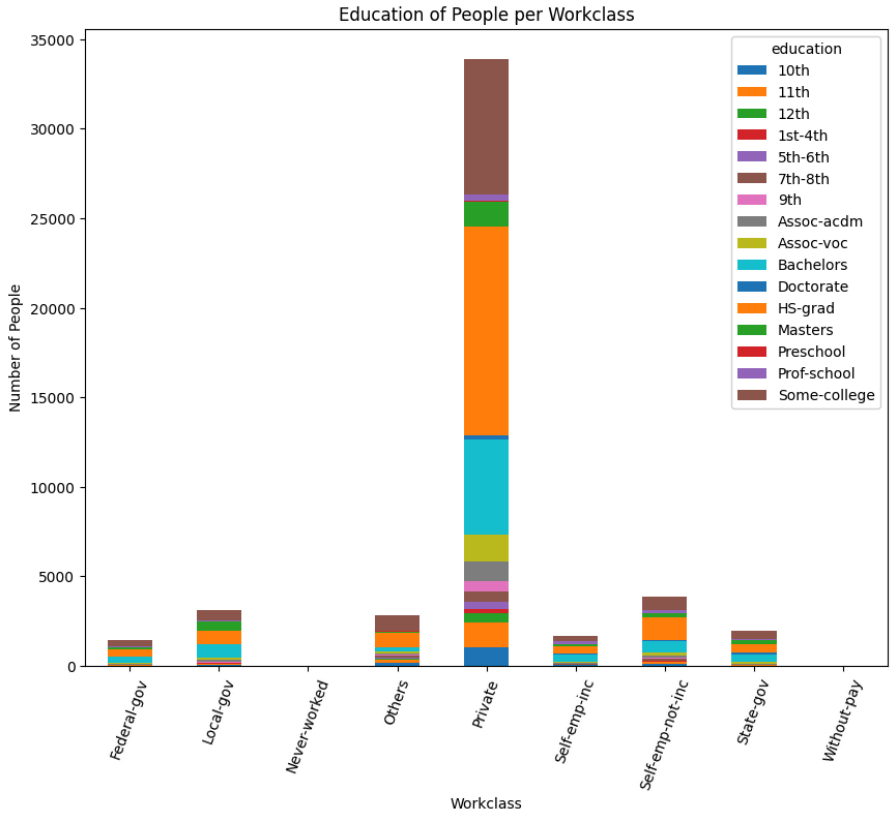
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 [Text(0, 0, 'Adm-clerical'),
  Text(1, 0, 'Armed-Forces'),
  Text(2, 0, 'Craft-repair'),
  Text(3, 0, 'Exec-managerial'),
  Text(4, 0, 'Farming-fishing'),
  Text(5, 0, 'Handlers-cleaners'),
  Text(6, 0, 'Machine-op-inspct'),
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




```
# plotting the education of people per workclass

import matplotlib.pyplot as plt

grouped_data = data.groupby(['workclass', 'education']).size().unstack()

fig, ax = plt.subplots(figsize=(10, 8))
grouped_data.plot(kind='bar', stacked=True, ax=ax)

ax.set_title('Education of People per Workclass')
```

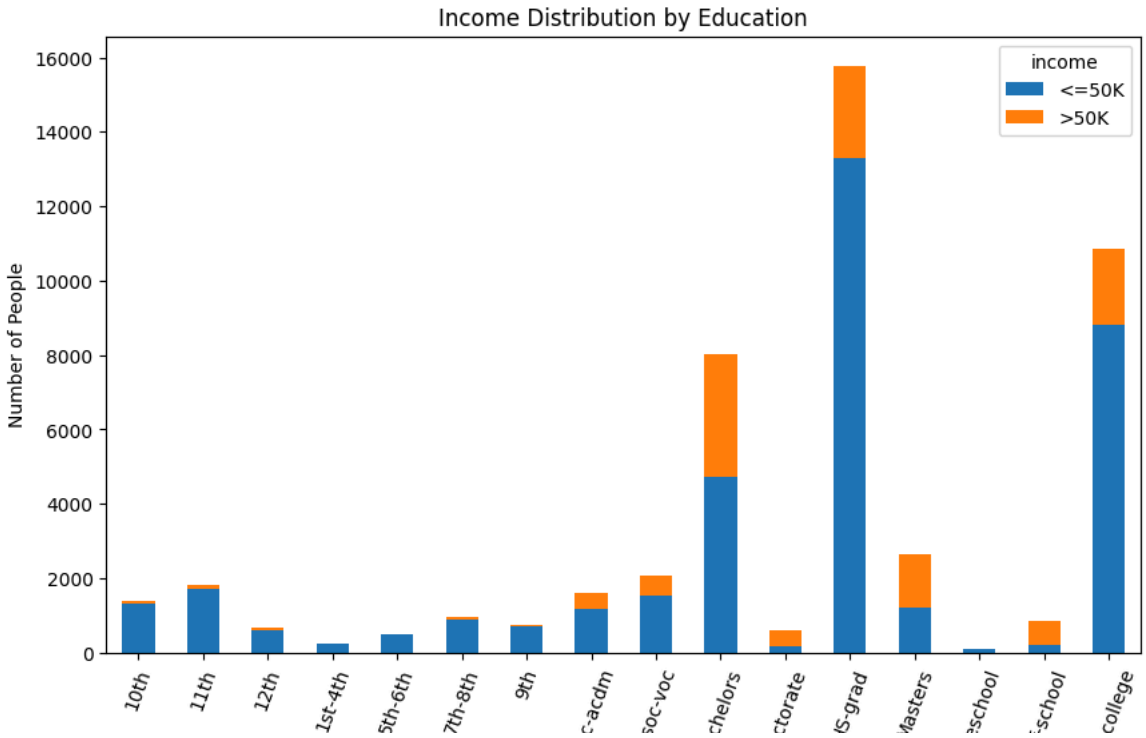
```
ax.set_xlabel('Workclass')
ax.set_ylabel('Number of People')

plt.xticks(rotation=70)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Federal-gov'),
  Text(1, 0, 'Local-gov'),
  Text(2, 0, 'Never-worked'),
  Text(3, 0, 'Others'),
  Text(4, 0, 'Private'),
  Text(5, 0, 'Self-emp-inc'),
  Text(6, 0, 'Self-emp-not-inc'),
  Text(7, 0, 'State-gov'),
  Text(8, 0, 'Without-pay')])
```




```
# plotting the income based on education using matplotlib

import matplotlib.pyplot as plt

grouped_data = data.groupby(['education', 'income']).size().unstack()

fig, ax = plt.subplots(figsize=(10, 6))
grouped_data.plot(kind='bar', stacked=True, ax=ax)

ax.set_title('Income Distribution by Education')
ax.set_xlabel('Education')
ax.set_ylabel('Number of People')

plt.xticks(rotation=70)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15]),
 [Text(0, 0, '10th'),
  Text(1, 0, '11th'),
  Text(2, 0, '12th'),
  Text(3, 0, '1st-4th'),
  Text(4, 0, '5th-6th'),
  Text(5, 0, '7th-8th'),
  Text(6, 0, '9th'),
  Text(7, 0, 'Assoc-acdm'),
  Text(8, 0, 'Assoc-voc'),
  Text(9, 0, 'Bachelors'),
  Text(10, 0, 'Doctorate'),
  Text(11, 0, 'HS-grad'),
  Text(12, 0, 'Masters'),
  Text(13, 0, 'Preschool'),
  Text(14, 0, 'Prof-school'),
  Text(15, 0, 'Some-college')])
```



Start coding or generate with AI.