About Dataset

An individual's annual income results from various factors. Intuitively, it is influenced by the individual's education level, age, gender, occupation, and etc.

The detailed description on the dataset can be found in the original UCI documentation

http://www.cs.toronto.edu/~delve/data/adult/adultDetail.html

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("adult.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband
4	18	?	103497	Some-college	10	Never-married	?	Own-child

In [4]:
```
df.tail()
```

Out[4]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relation
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husl
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unma
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	

In [5]:
```
df.shape
```

Out[5]:
```
(48842, 15)
```

In [6]:
```
df.shape[0]
```

Out[6]:
```
48842
```

In [7]:
```
df.shape[1]
```

Out[7]:
```
15
```

In [8]:
```
df.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        48842 non-null  object
 2   fnlwgt           48842 non-null  int64 
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64 
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           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   48842 non-null  object
 14  income           48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

Fetch Random Sample From the Dataset (50%)

In [9]:
```
df.sample(frac =0.50)
```

Out[9]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relations
43700	21	Private	442131	HS-grad	9	Never-married	Handlers-cleaners	Own-c
21438	49	Private	43910	HS-grad	9	Married-civ-spouse	Adm-clerical	\
10006	63	Private	156127	Some-college	10	Married-civ-spouse	Handlers-cleaners	Husb
347	59	Private	107318	7th-8th	4	Married-civ-spouse	Craft-repair	Husb
39143	37	Self-emp-not-inc	205359	Some-college	10	Married-civ-spouse	Exec-managerial	\
...	...	...	...	...	...	...	...	
8422	19	Local-gov	268722	Some-college	10	Never-married	Sales	Ot rela
4058	23	?	213004	Some-college	10	Never-married	?	Own-c
7437	41	Private	145441	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husb
5167	55	Self-emp-not-inc	183580	Masters	14	Divorced	Exec-managerial	Not fa
16027	44	Private	212894	10th	6	Married-civ-spouse	Machine-op-inspct	Husb

24421 rows × 15 columns

In [10]:
```
# if i write "random_state" to some integer, then it'll reproduce same sequence

df.sample(frac=0.50, random_state = 100)
```

Out[10]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relations
12393	37	Private	110331	Prof-school	15	Married-civ-spouse	Other-service	W
48701	23	Private	45834	Bachelors	13	Never-married	Exec-managerial	Not fa
17918	28	Private	89718	HS-grad	9	Never-married	Sales	Not fa
11352	30	Private	351770	9th	5	Divorced	Other-service	Unmar
36198	31	Private	164190	10th	6	Married-civ-spouse	Transport-moving	Husb
...	...	...	...	...	...	...	...	
48573	41	Private	318046	Some-college	10	Married-civ-spouse	Transport-moving	Husb
47252	41	Local-gov	33658	Some-college	10	Married-civ-spouse	Protective-serv	Husb
33142	69	Private	312653	Some-college	10	Married-civ-spouse	Sales	Husb
2965	21	?	334593	Some-college	10	Never-married	?	Not fa
32089	34	Private	186269	HS-grad	9	Divorced	Adm-clerical	Own-c

24421 rows × 15 columns

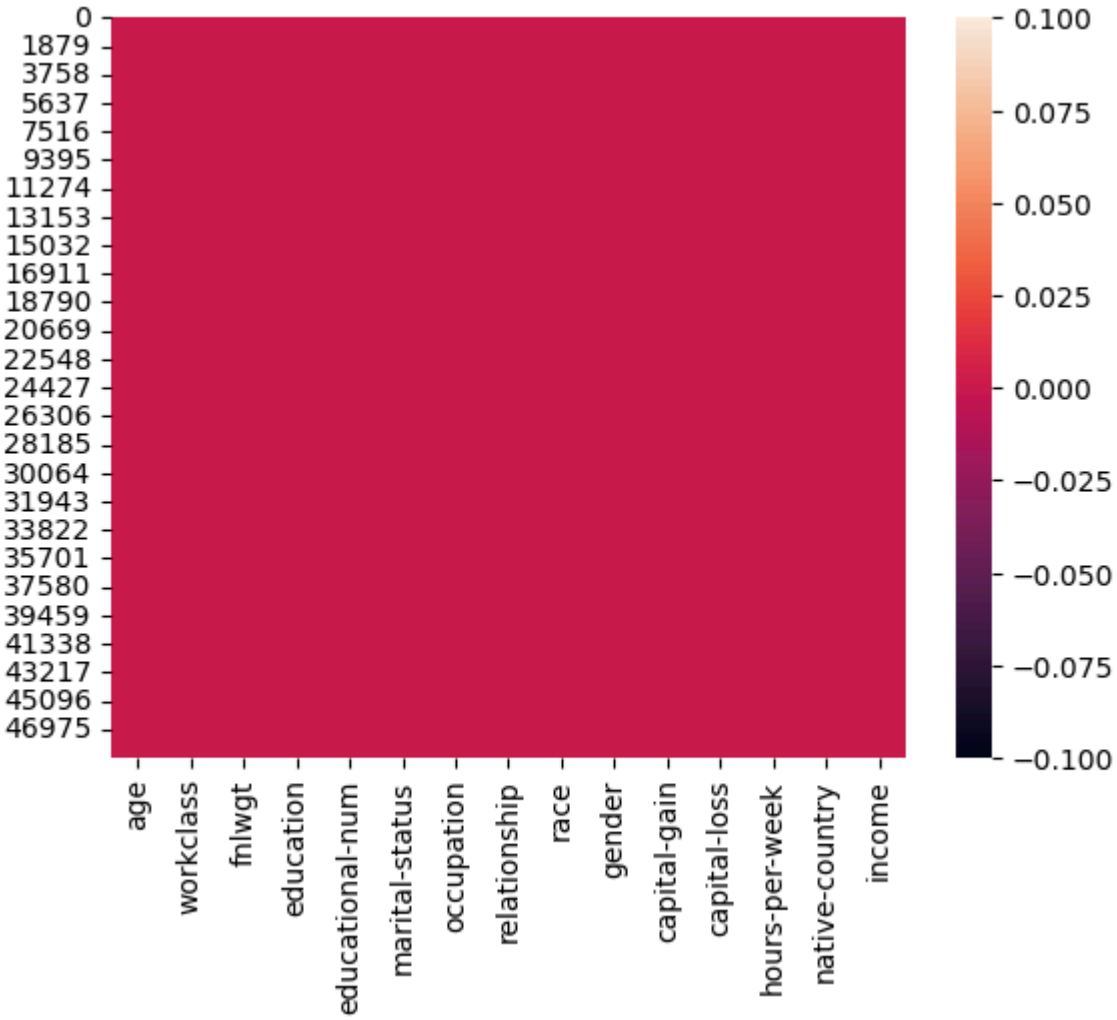
Check Null Values In The Dataset

In [11]:
```
df.isnull().sum(axis = 0)
```

Out[11]:
```
age               0
workclass         0
fnlwgt            0
education         0
educational-num   0
marital-status    0
occupation        0
relationship      0
race              0
gender            0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

In [12]:
```
sns.heatmap(df.isnull())
```

Out[12]: <Axes: >



Perform data cleaning [replacing '?' with NaN]

In [13]:
```
df.isin(['?']).sum()
```

Out[13]:
```
age                 0
workclass        2799
fnlwgt              0
education           0
educational-num     0
marital-status      0
occupation       2809
relationship        0
race                0
gender              0
capital-gain        0
capital-loss        0
hours-per-week      0
native-country    857
income              0
dtype: int64
```

In [14]:
```
# First we need to replace "?" with "NaN", then we can drop it with dropna metho

df['workclass'] = df['workclass'].replace('?',np.nan)
df['occupation'] = df['occupation'].replace('?',np.nan)
df['native-country'] = df['native-country'].replace('?',np.nan)
```

In [15]:
```
df.isin(['?']).sum()
```

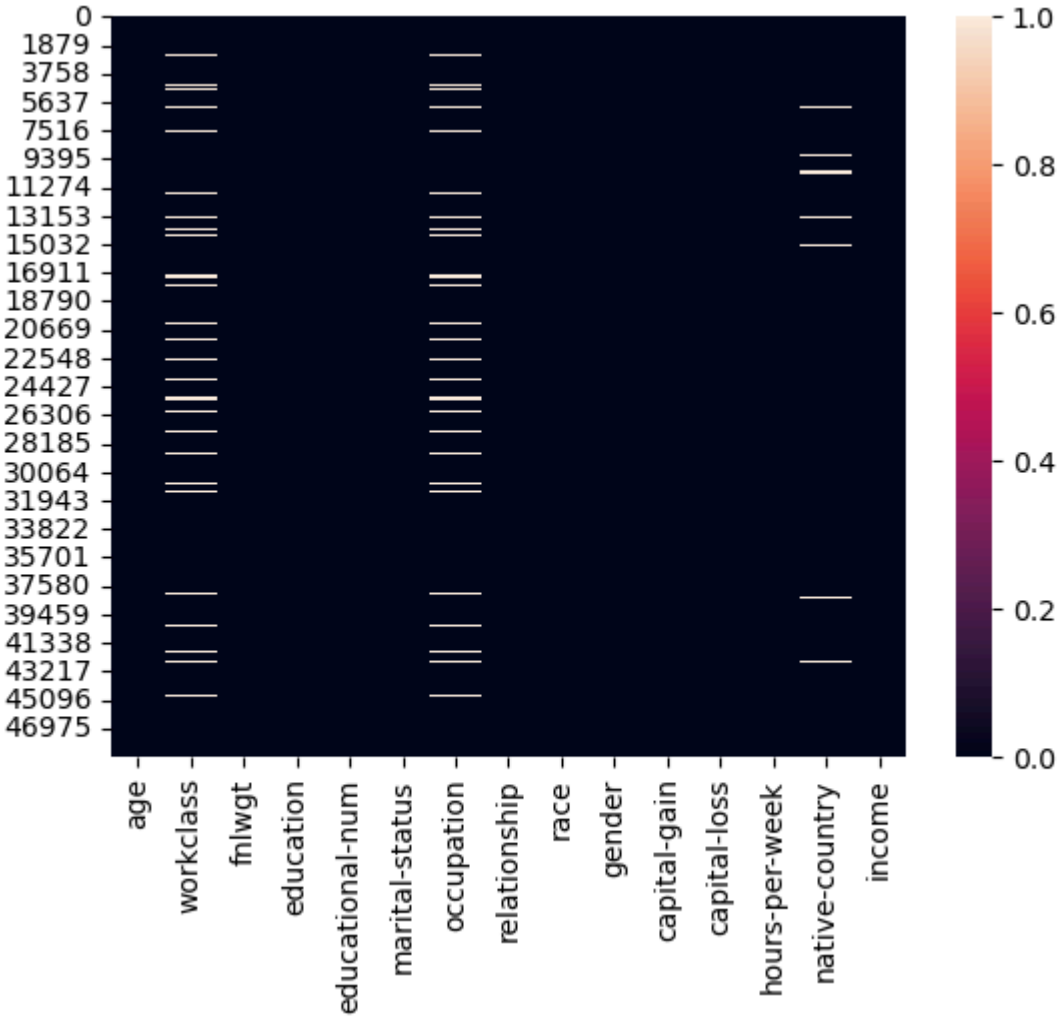
Out[15]:
```
age               0
workclass         0
fnlwgt            0
education         0
educational-num   0
marital-status    0
occupation        0
relationship      0
race              0
gender            0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

In [16]:
```
df.isnull().sum() # check for the null value
```

Out[16]:
```
age                0
workclass       2799
fnlwgt             0
education          0
educational-num    0
marital-status     0
occupation      2809
relationship       0
race               0
gender             0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country   857
income             0
dtype: int64
```

In [17]:
```
# Visualize the null values with heatmap
sns.heatmap(df.isnull())
```

Out[17]: <Axes: >



Drop all the Missing Values

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
age                 0
workclass        2799
fnlwgt              0
education           0
educational-num     0
marital-status      0
occupation       2809
relationship        0
race                0
gender              0
capital-gain        0
capital-loss        0
hours-per-week      0
native-country    857
income              0
dtype: int64
```

In [19]:
```
# Missing values in percentage
df.isnull().sum()*100 / len(df)
```

Out[19]:
```
age               0.000000
workclass         5.730724
fnlwgt            0.000000
education         0.000000
educational-num   0.000000
marital-status    0.000000
occupation        5.751198
relationship      0.000000
race              0.000000
gender            0.000000
capital-gain      0.000000
capital-loss      0.000000
hours-per-week    0.000000
native-country    1.754637
income            0.000000
dtype: float64
```

We can observe that 5% values are missing in "workclass", "occupation" and 1% in "native-country"

In [20]:
```
# Now drop the NaN values

df.dropna(how = 'any', inplace = True)
```

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
age               0
workclass         0
fnlwgt            0
education         0
educational-num   0
marital-status    0
occupation        0
relationship      0
race              0
gender            0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

In [22]:
```
df.shape
```

Out[22]:
```
(45222, 15)
```

Check For Duplicate Data and Drop Them

In [23]:
```
duplicate = df.duplicated().any()
```

In [24]:
```
duplicate
```

Out[24]:
```
np.True_
```

In [25]:
```
print("Do data have duplicated data::", duplicate)
```

```
Do data have duplicated data:: True
```

Drop duplicate data

In [26]:
```
df = df.drop_duplicates()
```

In [28]:
```
df.duplicated().any()
```

Out[28]:
```
np.False_
```

In [30]:
```
df.shape
```

Out[30]:
```
(45175, 15)
```

Get Overall Statistics About the Dataframe

In [32]:
```
df.describe()
```

Out[32]:

	age	fnlwgt	educational-num	capital-gain	capital-loss	hours-w
count	45175.000000	4.517500e+04	45175.000000	45175.000000	45175.000000	45175.000
mean	38.556170	1.897388e+05	10.119314	1102.576270	88.687593	40.942
std	13.215349	1.056524e+05	2.551740	7510.249876	405.156611	12.007
min	17.000000	1.349200e+04	1.000000	0.000000	0.000000	1.000
25%	28.000000	1.173925e+05	9.000000	0.000000	0.000000	40.000
50%	37.000000	1.783120e+05	10.000000	0.000000	0.000000	40.000
75%	47.000000	2.379030e+05	13.000000	0.000000	0.000000	45.000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000

In [34]:
```
df.describe(include = 'all') # to show complete stat for numerical and categoric
```

Out[34]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occu
count	45175.000000	45175	4.517500e+04	45175	45175.000000	45175	
unique	NaN	7	NaN	16	NaN	7	
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Craf
freq	NaN	33262	NaN	14770	NaN	21042	
mean	38.556170	NaN	1.897388e+05	NaN	10.119314	NaN	
std	13.215349	NaN	1.056524e+05	NaN	2.551740	NaN	
min	17.000000	NaN	1.349200e+04	NaN	1.000000	NaN	
25%	28.000000	NaN	1.173925e+05	NaN	9.000000	NaN	
50%	37.000000	NaN	1.783120e+05	NaN	10.000000	NaN	
75%	47.000000	NaN	2.379030e+05	NaN	13.000000	NaN	
max	90.000000	NaN	1.490400e+06	NaN	16.000000	NaN	

How many distinct categories are present in the education column?

In [35]:
```
df['education'].unique()
```

Out[35]:
```
array(['11th', 'HS-grad', 'Assoc-acdm', 'Some-college', '10th',
       'Prof-school', '7th-8th', 'Bachelors', 'Masters', '5th-6th',
       'Assoc-voc', '9th', 'Doctorate', '12th', '1st-4th', 'Preschool'],
      dtype=object)
```

In [39]:
```
df['educational-num'].unique()
```

Out[39]:
```
array([ 7,  9, 12, 10,  6, 15,  4, 13, 14,  3, 11,  5, 16,  8,  2,  1])
```

In [40]:
```
df.head()
```

Out[40]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family

Remove any columns from the DataFrame that are unnecessary for the intended analysis.

In [41]:
```
df = df.drop(['educational-num', 'capital-gain', 'capital-loss'], axis = 1)
```

In [42]:
```
df.columns
```

Out[42]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'marital-status',
       'occupation', 'relationship', 'race', 'gender', 'hours-per-week',
       'native-country', 'income'],
      dtype='object')
```

In [43]:
```
df.head()
```

Out[43]:

	age	workclass	fnlwgt	education	marital-status	occupation	relationship	race	gende
0	25	Private	226802	11th	Never-married	Machine-op-inspct	Own-child	Black	Mal
1	38	Private	89814	HS-grad	Married-civ-spouse	Farming-fishing	Husband	White	Mal
2	28	Local-gov	336951	Assoc-acdm	Married-civ-spouse	Protective-serv	Husband	White	Mal
3	44	Private	160323	Some-college	Married-civ-spouse	Machine-op-inspct	Husband	Black	Mal
5	34	Private	198693	10th	Never-married	Other-service	Not-in-family	White	Mal

Univariate Analysis

- In univariate analysis, we are taking one variable at a time and performing analysis.
- In univariate analysis, the data being analyzed contains only one variable.
- The main purpose of univariate analysis is to describe the data and find patterns exist with in it.

In [44]:
```
df.columns
```

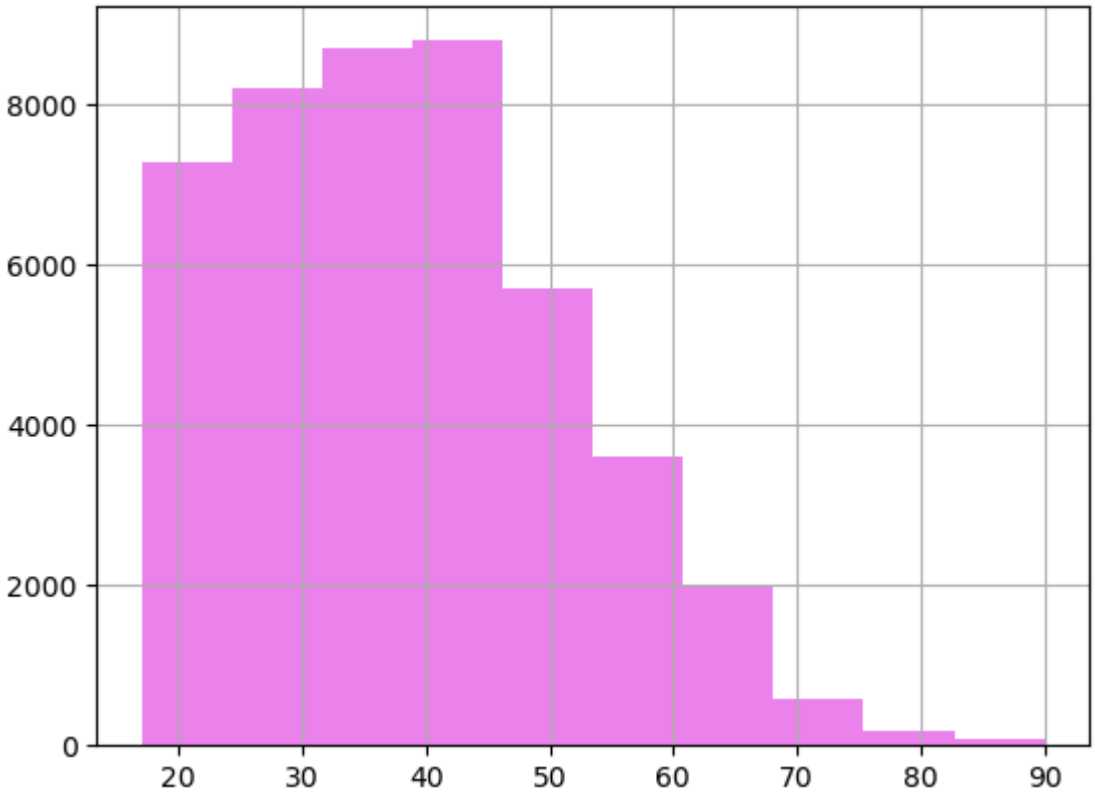
Out[44]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'marital-status',
       'occupation', 'relationship', 'race', 'gender', 'hours-per-week',
       'native-country', 'income'],
      dtype='object')
```

In [45]:
```
df['age'].describe()
```

Out[45]:
```
count    45175.000000
mean        38.556170
std         13.215349
min         17.000000
25%         28.000000
50%         37.000000
75%         47.000000
max         90.000000
Name: age, dtype: float64
```

In [52]:
```
df['age'].hist(color = 'violet')
```

Out[52]:
```
<Axes: >
```



- We can observe that most of the c=age group values from 17 to 50 years.

Determine how many people have ages ranging from 17 to 48, inclusive, by applying the between() method to the age column.

In [53]:
```
# Method 1

sum((df['age']>=17) & (df['age']<=48))
```

Out[53]: 34858

In [54]:
```
# Method 2 (using between method)

sum(df['age'].between(17,48))
```

Out[54]: 34858

- we can observe that we have total 34858 people lies between 17 to 48 age group

What is the frequency distribution of the categories in the Workclass column?

In [55]:
```
df.columns
```

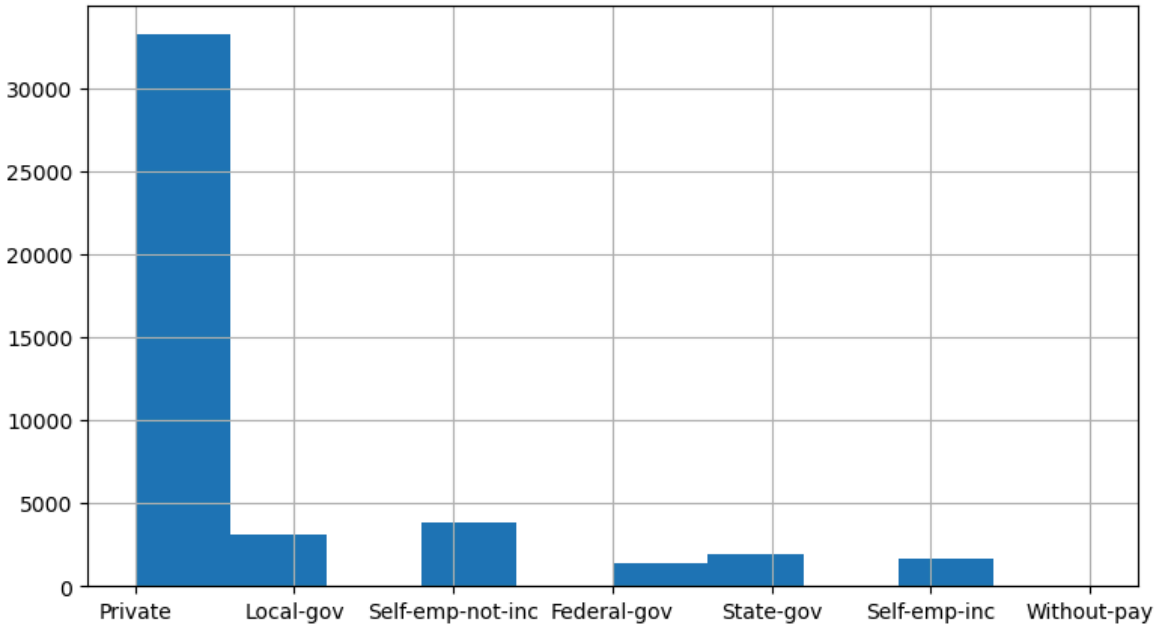
Out[55]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'marital-status',
       'occupation', 'relationship', 'race', 'gender', 'hours-per-week',
       'native-country', 'income'],
      dtype='object')
```

In [56]:
```
df['workclass'].describe()
```

Out[56]:
```
count       45175
unique          7
top       Private
freq        33262
Name: workclass, dtype: object
```

In [61]:
```
plt.figure(figsize = (9,5))
df['workclass'].hist()
```

Out[61]:
```
<Axes: >
```



- Most of the people belongs to the Private case

What is the total count of individuals holding either a Bachelor's or a Master's degree?

In [62]:
```
# Method 1

df.columns
```

Out[62]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'marital-status',
       'occupation', 'relationship', 'race', 'gender', 'hours-per-week',
       'native-country', 'income'],
      dtype='object')
```

In [63]:
```
f1 = df['education'] =="Bachelors"
f2 = df['education'] =="Masters"
```

In [64]:
```
len(df[f1 +f2])
```

Out[64]: 10072

In [66]:
```
len(df[f1 | f2])
```

Out[66]: 10072

In [67]:
```
(df[f1 | f2])
```

Out[67]:

	age	workclass	fnlwgt	education	marital-status	occupation	relationship	race
11	36	Federal-gov	212465	Bachelors	Married-civ-spouse	Adm-clerical	Husband	White
15	43	Private	346189	Masters	Married-civ-spouse	Exec-managerial	Husband	White
20	34	Private	107914	Bachelors	Married-civ-spouse	Tech-support	Husband	White
23	25	Private	220931	Bachelors	Never-married	Prof-specialty	Not-in-family	White
24	25	Private	205947	Bachelors	Married-civ-spouse	Prof-specialty	Husband	White
...	...	...	...	...	...	...	...	...
48817	34	Private	160216	Bachelors	Never-married	Exec-managerial	Not-in-family	White
48819	38	Private	139180	Bachelors	Divorced	Prof-specialty	Unmarried	Black
48825	31	Private	199655	Masters	Divorced	Other-service	Not-in-family	Other
48834	32	Private	116138	Masters	Never-married	Tech-support	Not-in-family	Asian-Pac-Islander
48835	53	Private	321865	Masters	Married-civ-spouse	Exec-managerial	Husband	White

10072 rows × 12 columns

In [68]:
```
# Method 2

sum(df['education'].isin(['Bachelors','Masters']))
```

Out[68]: 10072

Bivariate Analysis

- Bivariate analysis is a statistical technique used to examine the relationship between two variables, often visualized through simple methods such as scatterplots or boxplots.

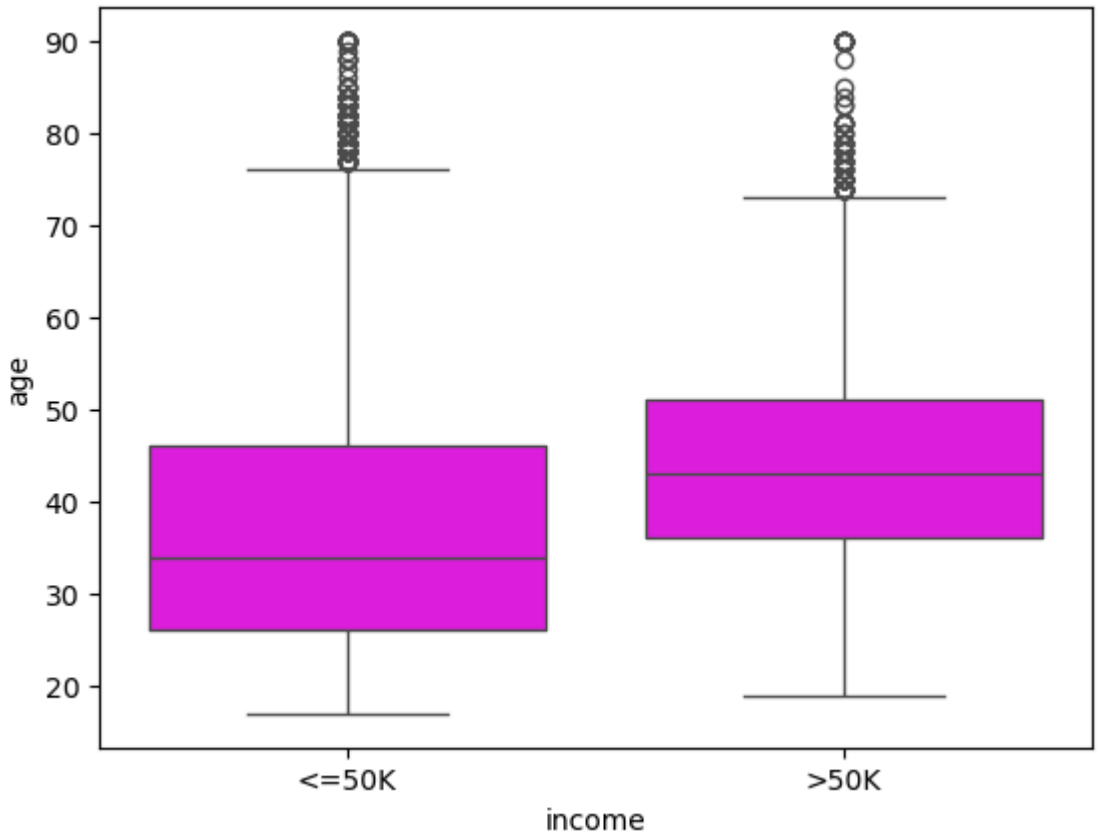
Analyse the income with respect to age

In [69]:
```
df.columns
```

Out[69]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'marital-status',
       'occupation', 'relationship', 'race', 'gender', 'hours-per-week',
       'native-country', 'income'],
      dtype='object')
```

In [71]:
```
sns.boxplot(x = 'income', y='age', data = df, color = 'magenta')
```

Out[71]:
```
<Axes: xlabel='income', ylabel='age'>
```



- From above graph we can see, most of people are younger and having income <=50K.
- People who are aged are having salary >50K

*** Thats how we can check relationship between two variables - Bivariate Analysis.**

Replace the Income Value ['<=50K', '>50k'] With 0 and 1

In [73]:
```
df['income'].value_counts()
```

Out[73]:
```
income
<=50K    33973
>50K     11202
Name: count, dtype: int64
```

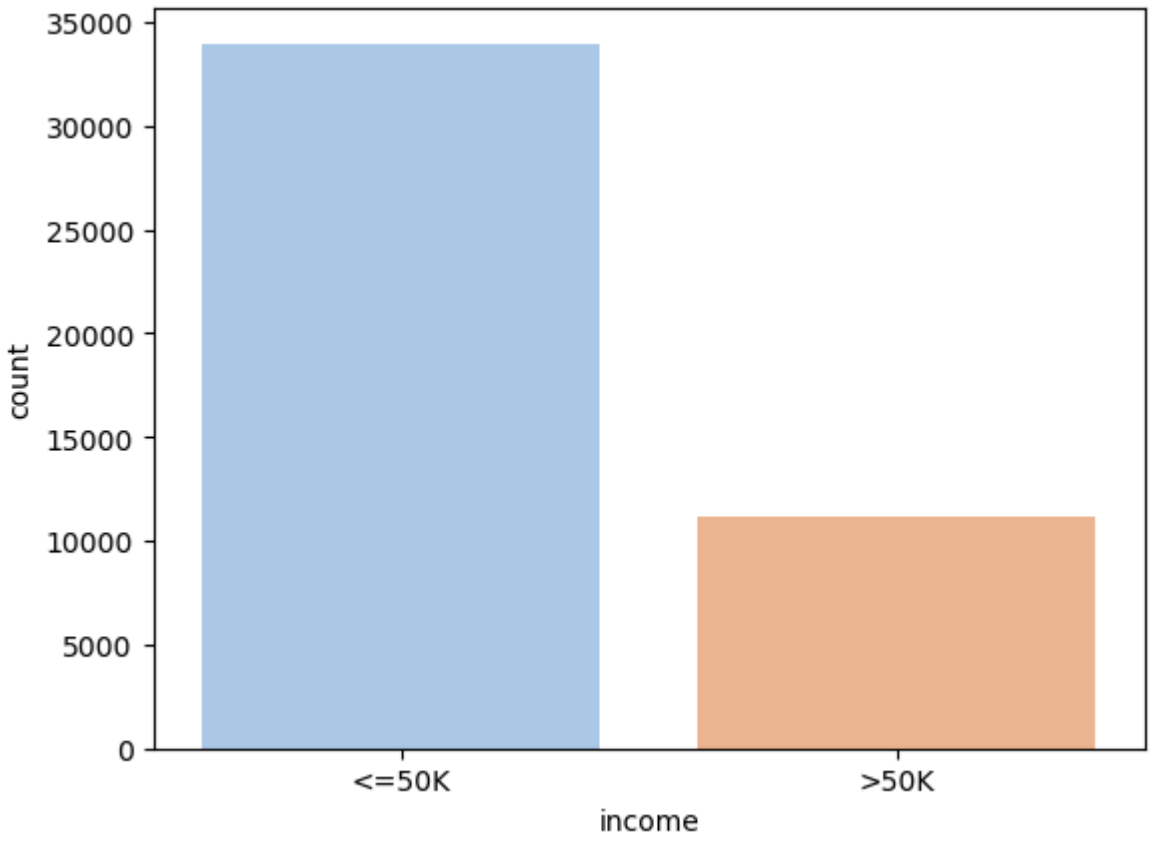
In [76]:
```
sns.countplot(x='income', data = df, palette='pastel', legend = False)
```

```
C:\Users\sanad\AppData\Local\Temp\ipykernel_7688\4122492235.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='income', data = df, palette='pastel', legend = False)
```

Out[76]:
```
<Axes: xlabel='income', ylabel='count'>
```



In [80]:
```
# Method 1
```

In [77]:
```
def income_data(inc):
    if inc == '<=50K':
        return 0
```

```
    else:
        return 1
```

In [78]:
```
df['enconded_salary'] = df['income'].apply(income_data)
```

In [79]:
```
df.head()
```

Out[79]:

	age	workclass	fnlwgt	education	marital-status	occupation	relationship	race	gende
0	25	Private	226802	11th	Never-married	Machine-op-inspct	Own-child	Black	Mal
1	38	Private	89814	HS-grad	Married-civ-spouse	Farming-fishing	Husband	White	Mal
2	28	Local-gov	336951	Assoc-acdm	Married-civ-spouse	Protective-serv	Husband	White	Mal
3	44	Private	160323	Some-college	Married-civ-spouse	Machine-op-inspct	Husband	Black	Mal
5	34	Private	198693	10th	Never-married	Other-service	Not-in-family	White	Mal

Method 2 - Use replace method,

- to_replace and values - to_replace=['<=50K','>50k']
- replace it with - value=[0,1]

In [82]:
```
df.replace(to_replace = ['<=50K','>50k'], value = [0,1], inplace =True)
```

In [84]:
```
df.head(1)
```

Out[84]:

	age	workclass	fnlwgt	education	marital-status	occupation	relationship	race	gender
0	25	Private	226802	11th	Never-married	Machine-op-inspct	Own-child	Black	Male

Which Workclass Getting The Highest Salary?

In [87]:
```
df.groupby('workclass')['enconded_salary'].mean().sort_values(ascending = False)
```

Out[87]:
```
workclass
Self-emp-inc        0.554407
Federal-gov         0.390469
Local-gov           0.295161
Self-emp-not-inc    0.279051
State-gov           0.267215
Private             0.217816
Without-pay         0.095238
Name: enconded_salary, dtype: float64
```

Who Has Better Chance To Get Salary greater than 50K Male or Female?

In [88]:
```
df.groupby('gender')['enconded_salary'].mean().sort_values(ascending = False)
```

Out[88]:
```
gender
Male      0.312609
Female    0.113692
Name: enconded_salary, dtype: float64
```