Exploratory Data Analysis & Data Cleaning, Feature Extraction

In this section, I checked variables and their properties. Try to get insight about the data enough to train a useful model. Then, I cleared outliers, handled missing data, and get dummy variables of categorical data.

You can find descriptions of the data from README.md.

Notebook Structure

- **Exploring Data**
  - **Continuous Variables Relationship with Income**
  - **Categorical Variables Relationship with Income**
- **Handling Missing Data**
  - **Explore Missing Data**
  - **Filling Null Values Which has Meaning**
- **Encoding Data**

In [1]:
```
from pathlib import Path
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from src.utils import html_table, categories_show

raw_data_dir = Path('../data/raw/')
processed_data_dir = Path('../data/processed/')
file_name = 'adult.csv'
file_path = raw_data_dir / file_name
```

Exploring Data

Our outcome variable is 'income'. We will try to estimate person has +50k income or not.

In [2]:
```
df = pd.read_csv(file_path)
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Out[2]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	W
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	W
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	B
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	W
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	W

In [3]:

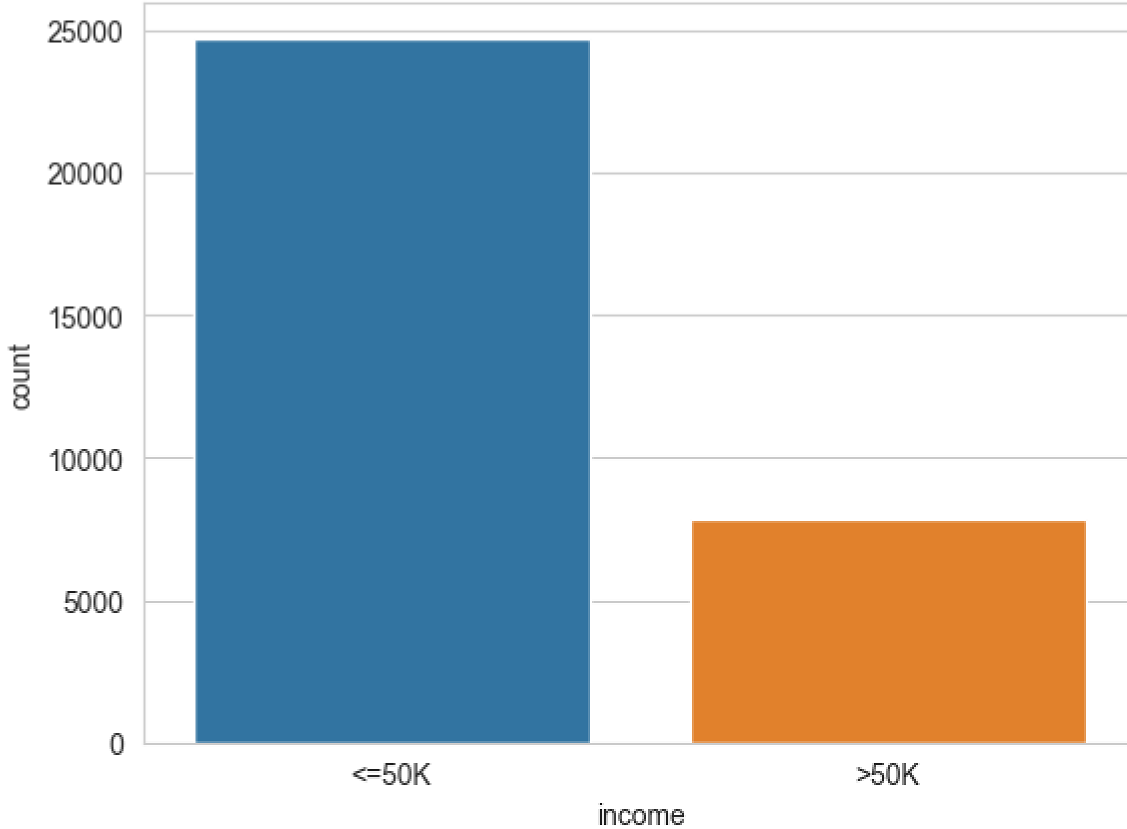
```
df.describe()
```

Out[3]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [4]:
```python
sns.countplot(data=df, x='income')
```

Out[4]: <AxesSubplot: xlabel='income', ylabel='count'>



Continuous Variables Relationship with Income

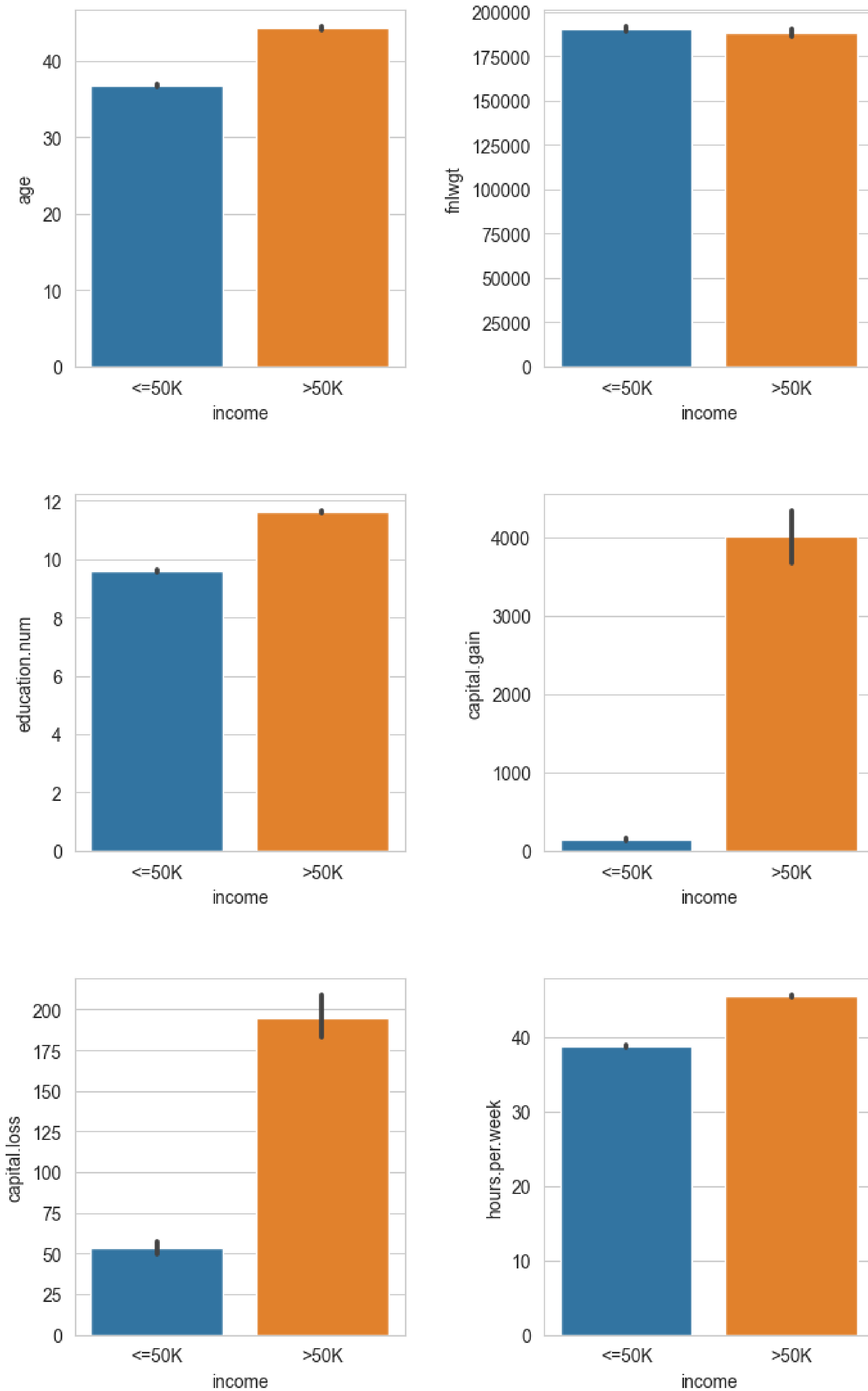
I inspect variables and check the most correlated continuous variables on the variable which will be predicted(DV). We can see below 4 continuous variables correlated with income has more than 0.2 r value so it seems like we can fit a suitable model even just by using them.

In [5]:
```python
df_corr = df.copy()
df_corr['income'] = pd.get_dummies(df_corr['income'], drop_first=True)
df_corr.corr(numeric_only=True).sort_values(by='income', ascending=False,
                                             key=lambda val: abs(val))
```

Out[5]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	i
income	0.234037	-0.009463	0.335154	0.223329	0.150526	0.229689	1.0
education.num	0.036527	-0.043195	1.000000	0.122630	0.079923	0.148123	0.3
age	1.000000	-0.076646	0.036527	0.077674	0.057775	0.068756	0.2
hours.per.week	0.068756	-0.018768	0.148123	0.078409	0.054256	1.000000	0.2
capital.gain	0.077674	0.000432	0.122630	1.000000	-0.031615	0.078409	0.2
capital.loss	0.057775	-0.010252	0.079923	-0.031615	1.000000	0.054256	0.1
fnlwgt	-0.076646	1.000000	-0.043195	0.000432	-0.010252	-0.018768	-0.0

In [6]:
```
cont_cols = df.select_dtypes(exclude='object').columns
fig, axs = plt.subplots(nrows=int((len(cont_cols) + 1) / 2), ncols=2,
                        figsize=(8, 2 * (int(len(cont_cols) + 1 / 2))))
fig.tight_layout(pad=5.0)
for i, col in enumerate(cont_cols):
    sns.barplot(data=df, x='income', y=col, ax=axs[i // 2][i % 2])
```



age
0
10
20
30
40
<=50K
>50K
income
fnlwgt
0
25000
50000
75000
100000
125000
150000
175000
200000
<=50K
>50K
income
education.num
0
2
4
6
8
10
12
<=50K
>50K
income
capital.gain
0
1000
2000
3000
4000
<=50K
>50K
income
capital.loss
0
25
50
75
100
125
150
175
200
<=50K
>50K
income
hours.per.week
0
10
20
30
40
<=50K
>50K
income

In [7]:
```python
cont_df = df.select_dtypes(exclude='object')
cont_df['income'] = df['income']
sns.pairplot(data=cont_df, hue='income')
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x24280331870>
```

Categorical Variables Relationship with Income

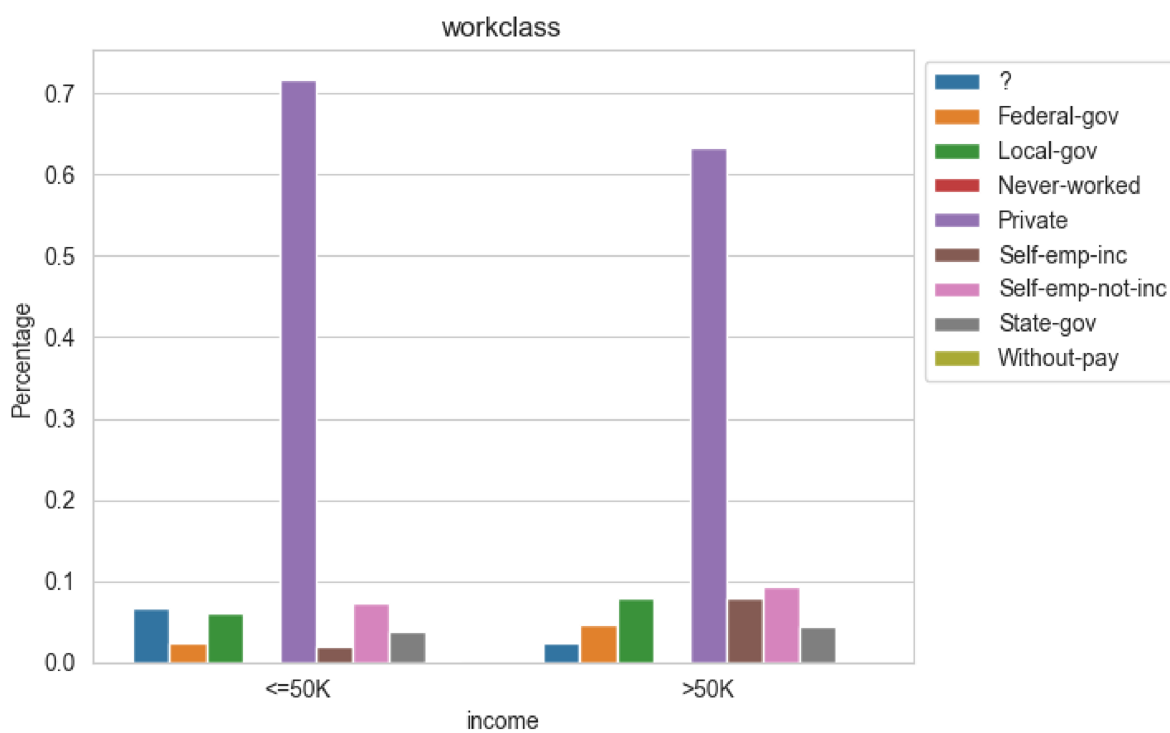
Because the data was unbalanced to see which variable how much contributed to each class I get and plot the percentage in this class.

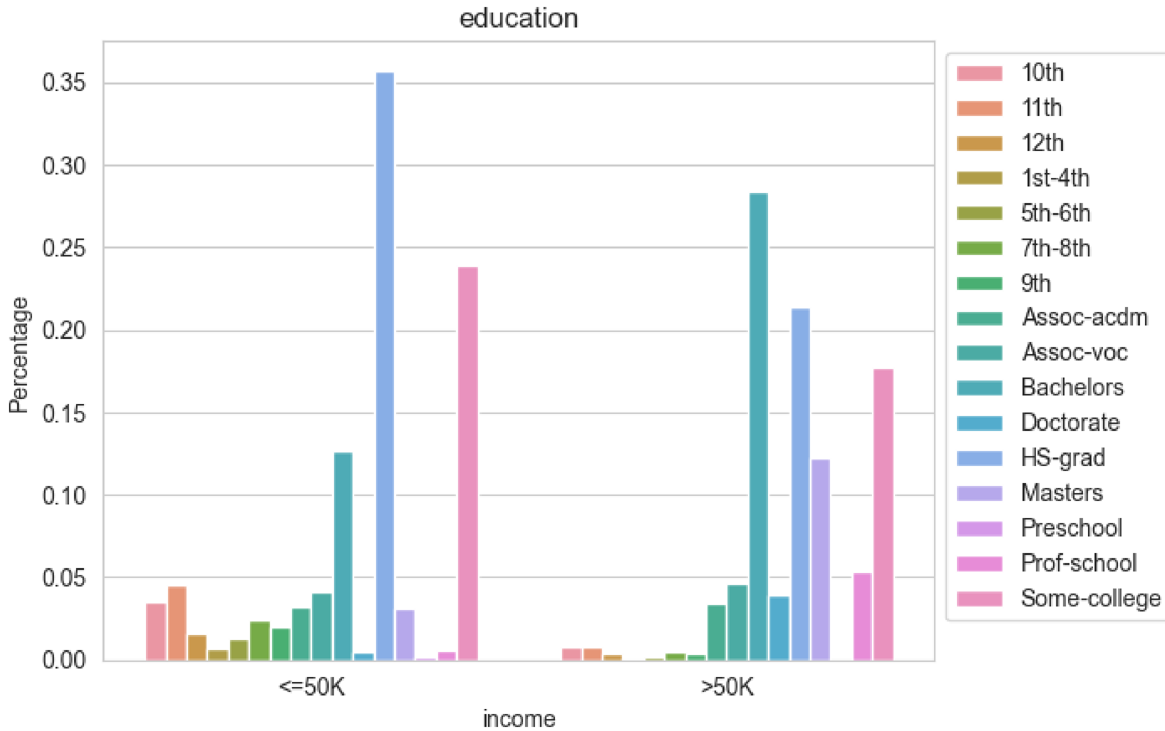
In [8]:
```python
obj_cols = list(df.select_dtypes('object').columns)
obj_cols.remove('income')
obj_cols.remove('native.country')
print(obj_cols)
print('Num of categorical cols: ', len(obj_cols))
```

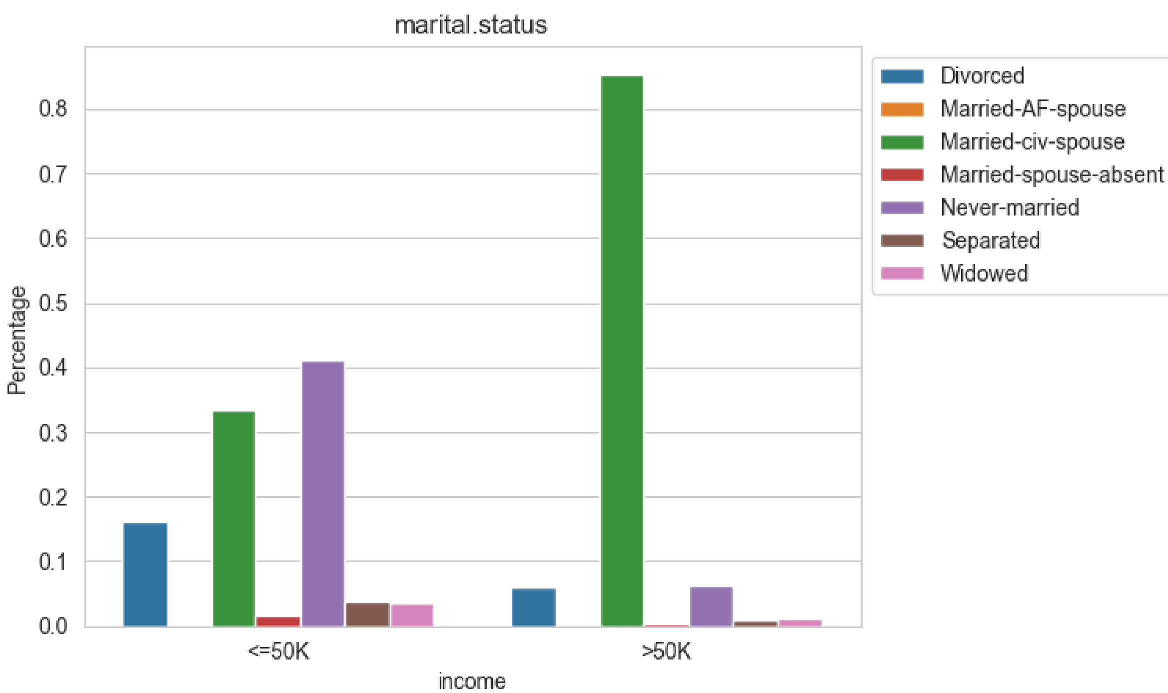
```
['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race',
'sex']
Num of categorical cols:  7
```

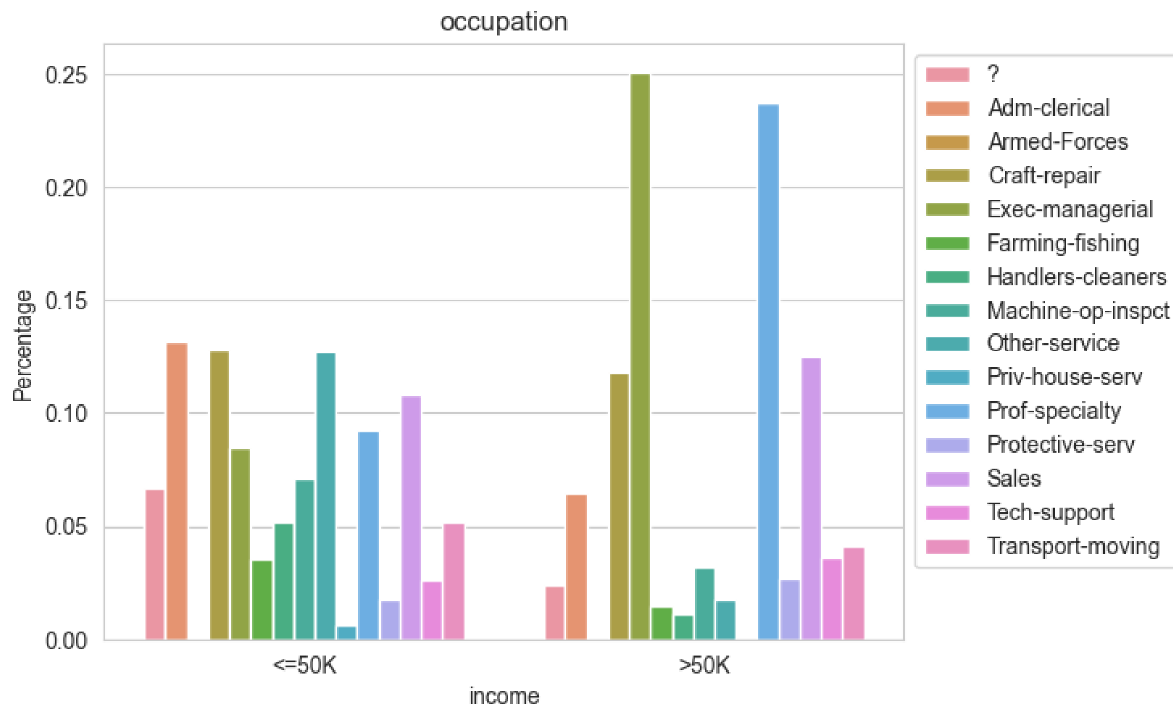
In [9]:
```python
figures = {}
```

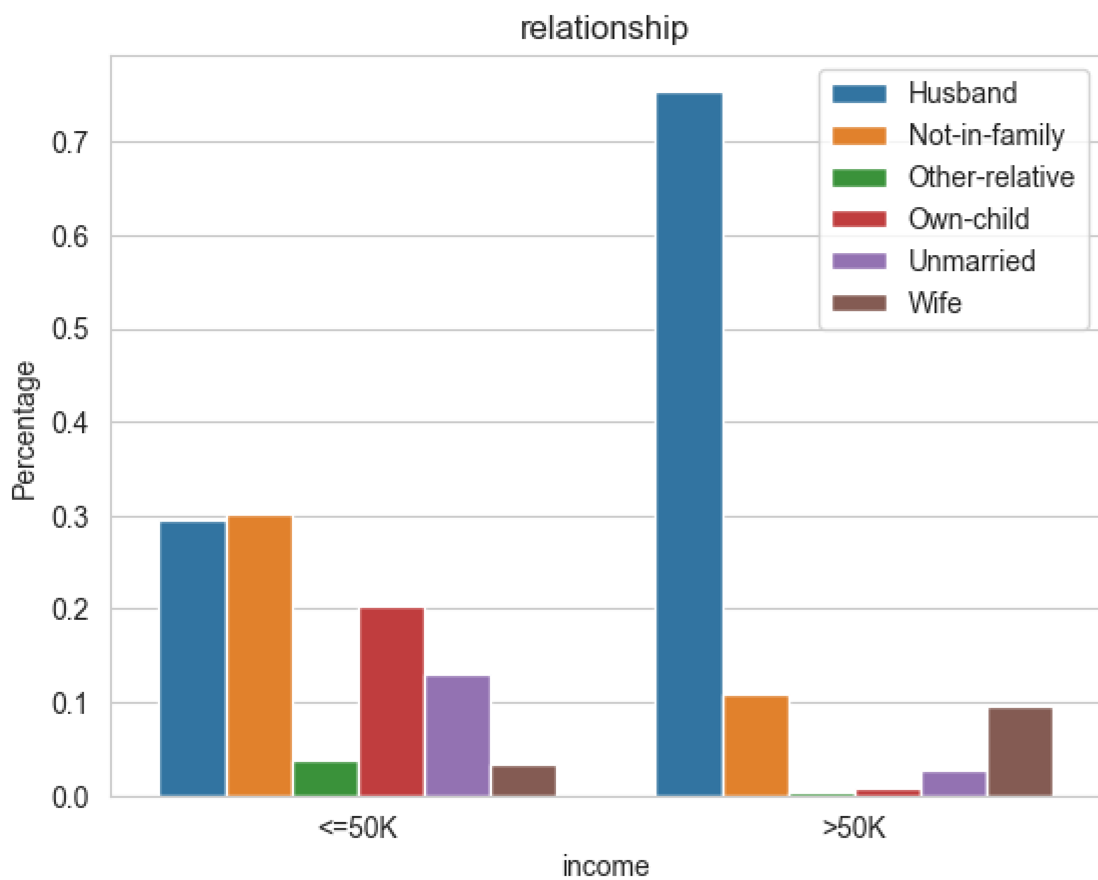
```python
for col in obj_cols:
    fig, ax = plt.subplots(dpi=100)
    figures[col] = {'fig': fig, 'ax': ax}

    count_incomes = df.groupby('income')[col].count()


    def apply_percentage(s):
        if s.iloc[0] == '<=50K':
            return s.count() / count_incomes['<=50K']
        return s.count() / count_incomes['>50K']


    percentage_incomes = df.groupby(['income', col])['income'].agg(
        apply_percentage).to_frame()
    percentage_incomes.columns = ['Percentage']
    index_names = tuple(zip(*percentage_incomes.index))
    percentage_incomes['income'] = index_names[0]
    percentage_incomes[col] = index_names[1]
    percentage_incomes.reset_index(drop=True, inplace=True)
    sns.barplot(data=percentage_incomes, x='income', y='Percentage', hue=col,
                ax=figures[col]['ax'])

    figures[col]['ax'].legend(loc='best', bbox_to_anchor=(1., 0., 0., 1.))
    figures[col]['ax'].set_title(col)
plt.show()
```

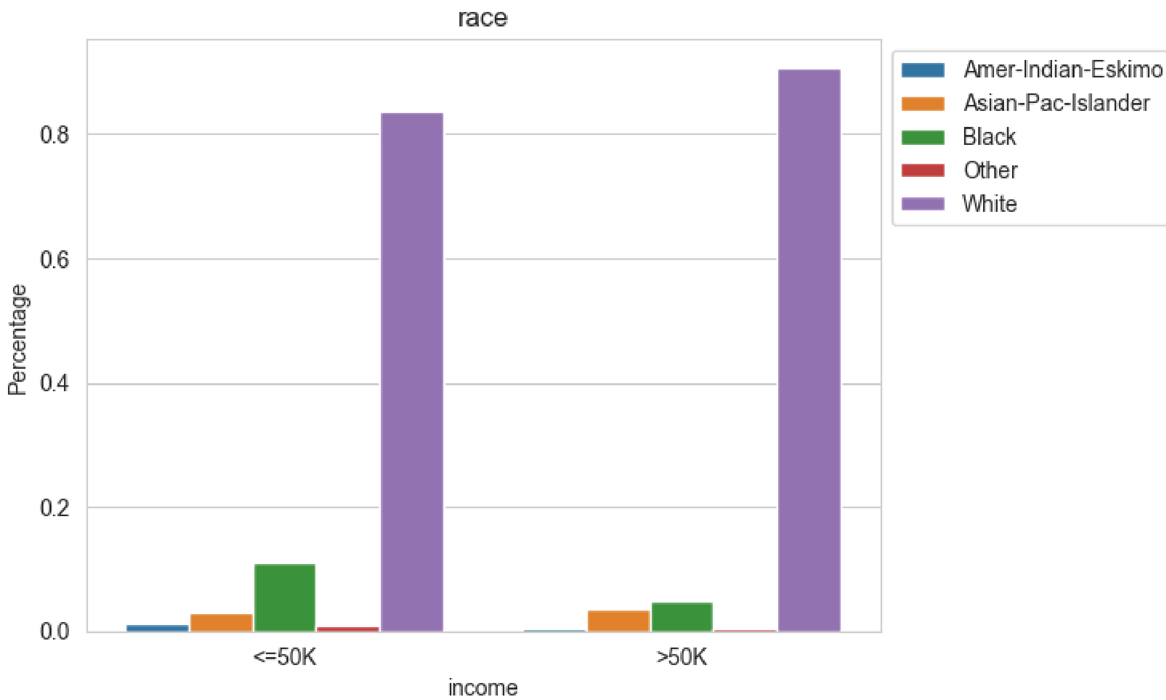


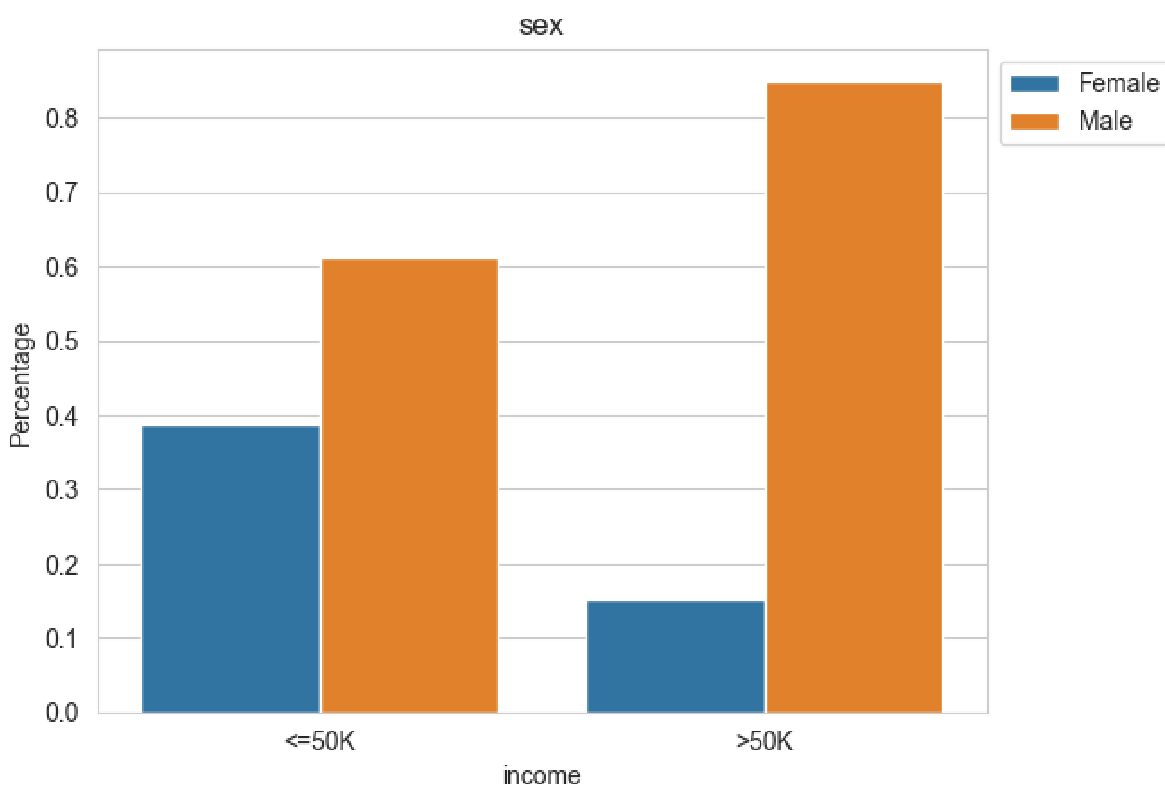

education
Percentage
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
<=50K
>50K
income
10th
11th
12th
1st-4th
5th-6th
7th-8th
9th
Assoc-acdm
Assoc-voc
Bachelors
Doctorate
HS-grad
Masters
Preschool
Prof-school
Some-college


marital.status
Percentage
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
<=50K
>50K
income
Divorced
Married-AF-spouse
Married-civ-spouse
Married-spouse-absent
Never-married
Separated
Widowed


occupation
Percentage
0.25
0.20
0.15
0.10
0.05
0.00
<=50K
>50K
income
?
Adm-clerical
Armed-Forces
Craft-repair
Exec-managerial
Farming-fishing
Handlers-cleaners
Machine-op-inspct
Other-service
Priv-house-serv
Prof-specialty
Protective-serv
Sales
Tech-support
Transport-moving


relationship
Percentage
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
<=50K
>50K
income
Husband
Not-in-family
Other-relative
Own-child
Unmarried
Wife





Handling Missing Data

Explore Missing Data

In [10]:
```
object_df = df.select_dtypes('object')
numeric_df = df.select_dtypes(exclude=['object'])
```

In [11]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[11]:
```
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 [12]:
```
(df.select_dtypes(exclude=['object']) == 0).sum().sort_values(ascending=False)
```

Out[12]:
```
capital.loss      31042
capital.gain      29849
age                   0
fnlwgt                0
education.num         0
hours.per.week        0
dtype: int64
```

In [13]:
```
categories_show(object_df)
```

	0
native.country	42
education	16
occupation	15
workclass	9
marital.status	7
relationship	6
race	5
sex	2
income	2

	0	1	2	3	4	5	6	7
native.country	United-States	?	Mexico	Greece	Vietnam	China	Taiwan	Indi
education	HS-grad	Some-college	7th-8th	10th	Doctorate	Prof-school	Bachelors	Masters
occupation	?	Exec-managerial	Machine-op-inspct	Prof-specialty	Other-service	Adm-clerical	Craft-repair	Transport-moving
workclass	?	Private	State-gov	Federal-gov	Self-emp-not-inc	Self-emp-inc	Local-gov	Without-pay
marital.status	Widowed	Divorced	Separated	Never-married	Married-civ-spouse	Married-spouse-absent	Married-AF-spouse	NaN
relationship	Not-in-family	Unmarried	Own-child	Other-relative	Husband	Wife	NaN	NaN
race	White	Black	Asian-Pac-Islander	Other	Amer-Indian-Eskimo	NaN	NaN	NaN
sex	Female	Male	NaN	NaN	NaN	NaN	NaN	NaN
income	<=50K	>50K	NaN	NaN	NaN	NaN	NaN	NaN

In [14]:
```
(df == '?').sum().sort_values(ascending=False)
```

Out[14]:
```
occupation        1843
workclass         1836
native.country     583
age                  0
fnlwgt               0
education            0
education.num        0
marital.status       0
relationship         0
race                 0
sex                  0
capital.gain         0
capital.loss         0
hours.per.week       0
income               0
dtype: int64
```

In [15]:
```
df = df.replace(to_replace='?', value=None)
```

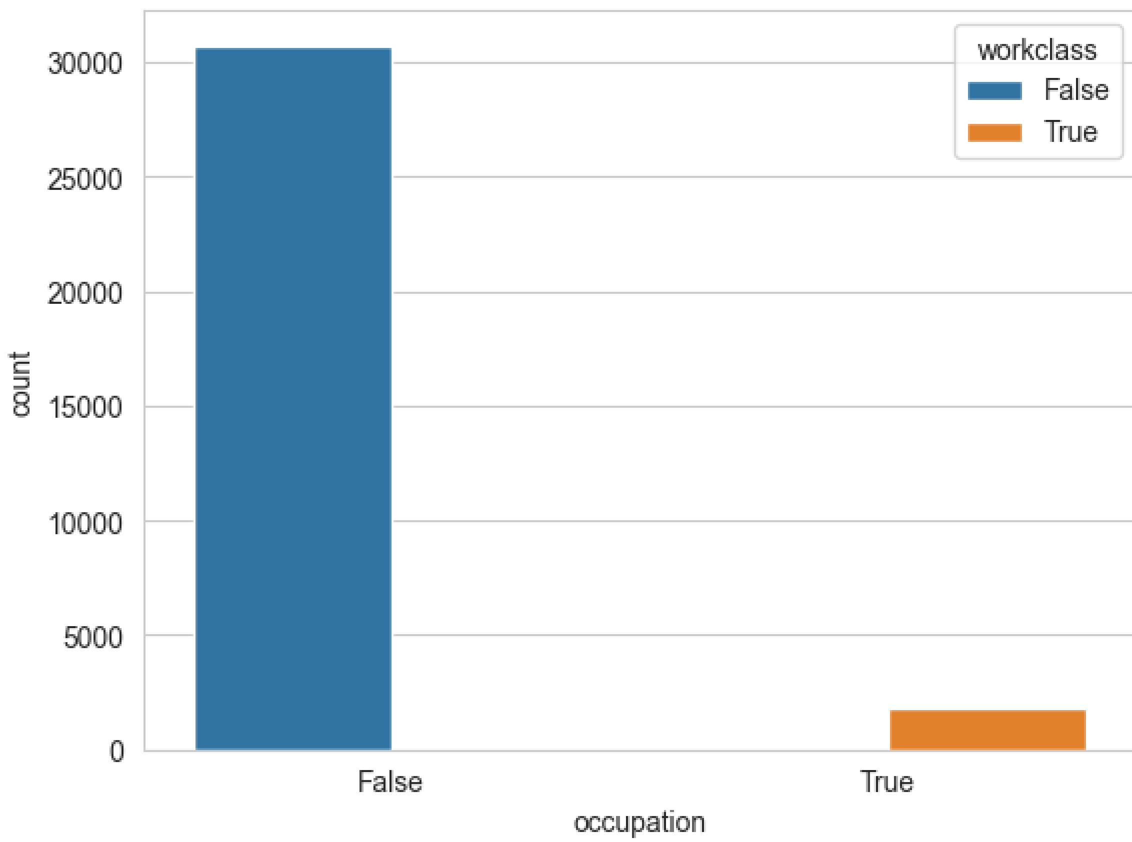
In [16]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[16]:
```
occupation        1843
workclass         1836
native.country     583
age                  0
fnlwgt               0
education            0
education.num        0
marital.status       0
relationship         0
race                 0
sex                  0
capital.gain         0
capital.loss         0
hours.per.week       0
income               0
dtype: int64
```

Filling Null Values Which has Meaning

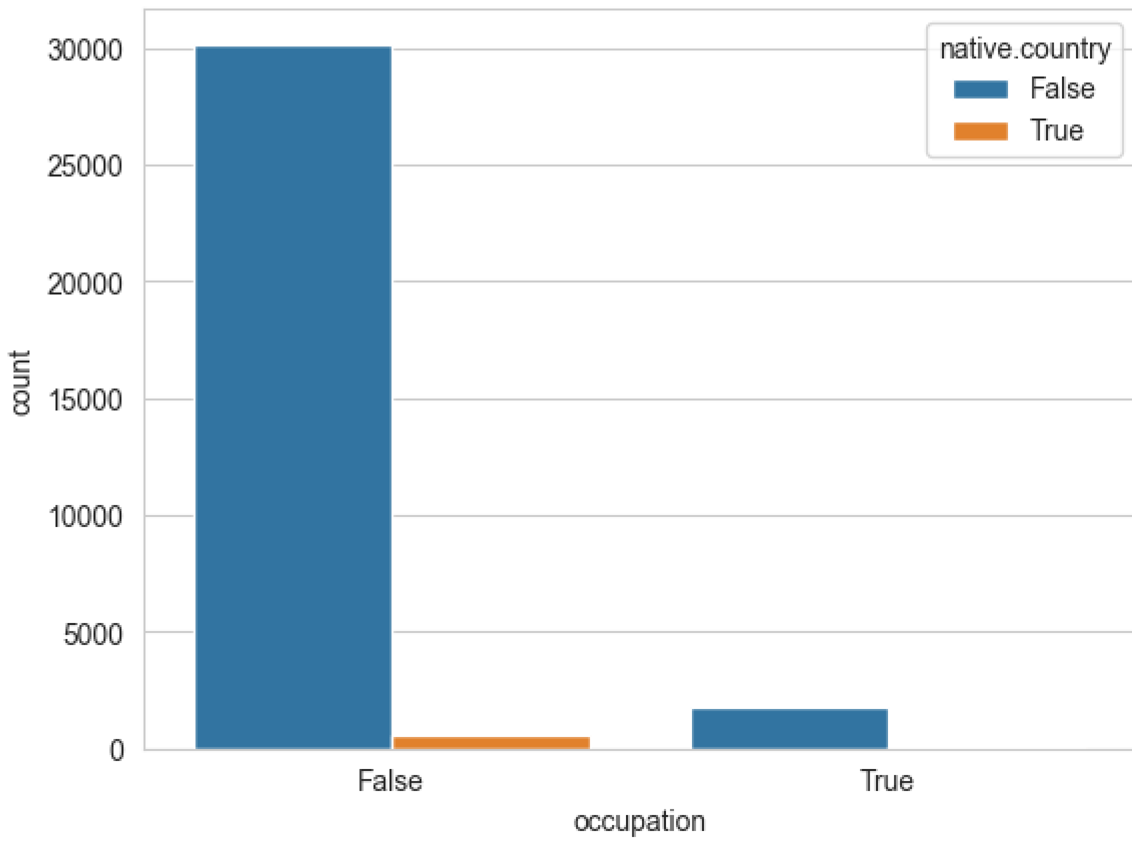
In [17]:
```
sns.countplot(data=df.isnull(), x='occupation', hue='workclass')
```

Out[17]: <AxesSubplot: xlabel='occupation', ylabel='count'>



In [18]:
```
sns.countplot(data=df.isnull(), x='occupation', hue='native.country')
```

Out[18]: <AxesSubplot: xlabel='occupation', ylabel='count'>



In [19]:
```
df[df['occupation'].isnull() & ~df['workclass'].isnull()]
```

Out[19]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationshi
8874	18	Never-worked	206359	10th	6	Never-married	None	Own-chil
13675	23	Never-worked	188535	7th-8th	4	Divorced	None	Not-in famil
17089	17	Never-worked	237272	10th	6	Never-married	None	Own-chil
21934	18	Never-worked	157131	11th	7	Never-married	None	Own-chil
24483	20	Never-worked	462294	Some-college	10	Never-married	None	Own-chil
32331	30	Never-worked	176673	HS-grad	9	Married-civ-spouse	None	Wif
32338	18	Never-worked	153663	Some-college	10	Never-married	None	Own-chil

In [20]:
```
df['occupation'].fillna('no-occupation', inplace=True)
df['workclass'].fillna('Never-worked', inplace=True)
```

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

Out[21]:
```
workclass           0
education           0
marital.status      0
occupation          0
relationship        0
race                0
sex                 0
native.country    583
income              0
dtype: int64
```

In [22]:
```
df[df['native.country'].isna()]
```

Out[22]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationshi
9	41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarrie
18	22	Private	119592	Assoc-acdm	12	Never-married	Handlers-cleaners	Not-in famil
65	60	Self-emp-inc	226355	Assoc-voc	11	Married-civ-spouse	Machine-op-inspct	Husban
86	39	Self-emp-not-inc	218490	Prof-school	15	Married-civ-spouse	Prof-specialty	Husban
87	43	Federal-gov	156996	Prof-school	15	Married-civ-spouse	Prof-specialty	Husban
...	...	...	...	...	...	...	...	.
32459	44	Self-emp-inc	71556	Masters	14	Married-civ-spouse	Sales	Husban
32476	58	Self-emp-inc	181974	Doctorate	16	Never-married	Prof-specialty	Not-in famil
32498	42	Self-emp-not-inc	217597	HS-grad	9	Divorced	Sales	Own-chil
32515	39	Private	107302	HS-grad	9	Married-civ-spouse	Prof-specialty	Husban
32528	81	Never-worked	120478	Assoc-voc	11	Divorced	no-occupation	Unmarrie

583 rows × 15 columns

In [23]:
```
df['native.country'].fillna('others', inplace=True)
```

In [24]:
```
df.select_dtypes('object').isnull().sum()
```

Out[24]:
```
workclass         0
education         0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
native.country    0
income            0
dtype: int64
```

In [25]:
```
categories_show(df.select_dtypes('object'))
```

	0
native.country	42
education	16
occupation	15
workclass	8
marital.status	7
relationship	6
race	5
sex	2
income	2

	0	1	2	3	4	5	6	
native.country	United-States	others	Mexico	Greece	Vietnam	China	Taiwan	Ind
education	HS-grad	Some-college	7th-8th	10th	Doctorate	Prof-school	Bachelors	Maste
occupation	no-occupation	Exec-managerial	Machine-op-inspct	Prof-specialty	Other-service	Adm-clerical	Craft-repair	Transpor movir
workclass	Never-worked	Private	State-gov	Federal-gov	Self-emp-not-inc	Self-emp-inc	Local-gov	Withou pa
marital.status	Widowed	Divorced	Separated	Never-married	Married-civ-spouse	Married-spouse-absent	Married-AF-spouse	Na
relationship	Not-in-family	Unmarried	Own-child	Other-relative	Husband	Wife	NaN	Na
race	White	Black	Asian-Pac-Islander	Other	Amer-Indian-Eskimo	NaN	NaN	Na
sex	Female	Male	NaN	NaN	NaN	NaN	NaN	Na
income	<=50K	>50K	NaN	NaN	NaN	NaN	NaN	Na

Encoding Data

In this section, I implemented one hot encoding and got dummy variables for categorical data.

In [26]:
```
df.reset_index(drop=True, inplace=True)
object_df = df.select_dtypes(include='object')
object_df.drop('income', axis=1, inplace=True)
# 1 represent income >50k and 0 represent income <= 50k
target_var = pd.get_dummies(df['income'], drop_first=True)
target_var.columns = ['income']
numeric_df = df.select_dtypes(exclude='object')
```

In [27]:
```
print(object_df.columns)
target_var.head()
```

```
Index(['workclass', 'education', 'marital.status', 'occupation',
       'relationship', 'race', 'sex', 'native.country'],
      dtype='object')
```

Out[27]:

	income
0	0
1	0
2	0
3	0
4	0

In [28]:

```
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder(handle_unknown='ignore', sparse=False)
enc.fit(object_df)
```

Out[28]:

```
▾ OneHotEncoder
OneHotEncoder(handle_unknown='ignore', sparse=False)
```

In [29]:

```
df_objects_dummies = enc.transform(object_df)
df_objects_dummies[0:5, :]
```

```
Out[29]: array([[0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1.,
                 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 1., 0., 0., 0.],
                [0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
                 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
                 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 1., 0., 0., 0.],
                [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
                 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 1., 0., 0., 0.],
                [0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 1., 0., 0., 0.],
                [0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
                 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
                 0., 1., 0., 0., 0.]])
```

```
In [30]: df_encoded = pd.concat(
             (numeric_df, pd.DataFrame(df_objects_dummies), target_var), axis=1)
         df_encoded.info()
         df_encoded.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Columns: 108 entries, age to income
dtypes: float64(101), int64(6), uint8(1)
memory usage: 26.6 MB
```

Out[30]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	0	1	2	3	...	9
0	90	77053	9	0	4356	40	0.0	0.0	1.0	0.0	...	0
1	82	132870	9	0	4356	18	0.0	0.0	0.0	1.0	...	0
2	66	186061	10	0	4356	40	0.0	0.0	1.0	0.0	...	0
3	54	140359	4	0	3900	40	0.0	0.0	0.0	1.0	...	0
4	41	264663	10	0	3900	40	0.0	0.0	0.0	1.0	...	0

5 rows × 108 columns

In [31]:

```
df_encoded.to_csv(processed_data_dir / file_name, index=False)
```

In [32]:

```
import joblib

joblib.dump(enc, '../models/featurebuild/0.1-onehotencoder.joblib')
```

Out[32]:

```
['../models/featurebuild/0.1-onehotencoder.joblib']
```