WAY TO REINVENT YOURSELF

¶

Data Analysis & Visualization with Python

Project Solution

Analysis of US Citizens by Income Levels

Content

Introduction

Content

One of the most important components to any data science experiment that doesn't get as much importance as it should is **`Exploratory Data Analysis (EDA)`** . In short, EDA is **`"A first look at the data"`** . It is a critical step in analyzing the data from an experiment. It is used to understand and summarize the content of the dataset to ensure that the features which we feed to our machine learning algorithms are refined and we get valid, correctly interpreted results. In general, looking at a column of numbers or a whole spreadsheet and determining the important characteristics of the data can be very tedious and boring. Moreover, it is good practice to understand the problem statement and the data before you get your hands dirty, which in view, helps to gain a lot of insights. I will try to explain the concept using the Adult dataset/Census Income dataset available on the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Adult). The problem statement here is to predict whether the income exceeds 50k a year or not based on the census data.

Aim of the Project

Applying Exploratory Data Analysis (EDA) and preparing the data to implement the Machine Learning Algorithms;

1. Analyzing the characteristics of individuals according to income groups
2. Preparing data to create a model that will predict the income levels of people according to their characteristics (So the "salary" feature is the target feature)

Dataset Info

Content

The Census Income dataset has 32561 entries. Each entry contains the following information about an individual:

- **salary (target feature/label):** whether or not an individual makes more than $50,000 annually. (<= 50K, >50K)
- **age:** the age of an individual. (Integer greater than 0)
- **workclass:** a general term to represent the employment status of an individual. (Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked)
- **fnlwgt:** this is the number of people the census believes the entry represents. People with similar demographic characteristics should have similar weights. There is one important caveat to remember about this statement. That is that since the CPS sample is actually a collection of 51 state samples, each with its own probability of selection, the statement only applies within state.(Integer greater than 0)
- **education:** the highest level of education achieved by an individual. (Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.)
- **education-num:** the highest level of education achieved in numerical form. (Integer greater than 0)
- **marital-status:** marital status of an individual. Married-civ-spouse corresponds to a civilian spouse while Married-AF-spouse is a spouse in the Armed Forces. Married-spouse-absent includes married people living apart because either the husband or wife was employed and living at a considerable distance from home (Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse)
- **occupation:** the general type of occupation of an individual. (Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces)
- **relationship:** represents what this individual is relative to others. For example an individual could be a Husband. Each entry only has one relationship attribute. (Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried)
- **race:** Descriptions of an individual's race. (White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black)
- **sex:** the biological sex of the individual. (Male, female)
- **capital-gain:** capital gains for an individual. (Integer greater than or equal to 0)
- **capital-loss:** capital loss for an individual. (Integer greater than or equal to 0)
- **hours-per-week:** the hours an individual has reported to work per week. (continuous)
- **native-country:** country of origin for an individual (United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands)

How to Installing/Enabling Intellisense or Autocomplete in Jupyter Notebook

Installing jupyter_contrib_nbextensions (https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/install.html)

To install the current version from The Python Package Index (PyPI), which is a repository of software for the Python programming language, simply type:

!pip install jupyter_contrib_nbextensions

Alternatively, you can install directly from the current master branch of the repository:

!pip install https://github.com/ipython-contrib/jupyter_contrib_nbextensions/tarball/master (https://github.com/ipython-contrib/jupyter_contrib_nbextensions/tarball/master)

Enabling Intellisense or Autocomplete in Jupyter Notebook (https://botbark.com/2019/12/18/how-to-enable-intellisense-or-autocomplete-in-jupyter-notebook/)

Installing hinterland for jupyter without anaconda

`STEP 1:` `Open cmd prompt and run the following commands` 1) pip install jupyter_contrib_nbextensions
2) pip install jupyter_nbextensions_configurator
3) jupyter contrib nbextension install --user
4) jupyter nbextensions_configurator enable --user

`STEP 2:` `Open jupyter notebook`

- click on nbextensions tab
- unckeck disable configuration for nbextensions without explicit compatibility
- put a check on Hinterland

`Step 3:` `Open new python file and check autocomplete feature`

VIDEO SOURCE (https://www.youtube.com/watch?v=DKE8hED0fow)

Jupyter Notebook extensions set up

1. pip install jupyter_contrib_nbextensions
2. pip install jupyter_nbextensions_configurator
3. jupyter contrib nbextension install --user
4. jupyter nbextensions_configurator enable --user

Jupyter Notebook extensions set up for those that using Anaconda

1. conda install -c conda-forge jupyter_contrib_nbextensions
2. jupyter contrib nbextension install --user
3. jupyter nbextensions_configurator enable --user

Importing Related Libraries

Content

Once you've installed NumPy & Pandas you can import them as a library:

In [1025]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
warnings.warn("this will not show")

plt.rcParams["figure.figsize"] = (10, 6)

sns.set_style("whitegrid")
pd.set_option('display.float_format', lambda x: '%.3f' % x)

# Set it None to display all rows in the dataframe
# pd.set_option('display.max_rows', None)

# Set it to None to display all columns in the dataframe
pd.set_option('display.max_columns', None)
```

Reading the data from file

In [1026]:

```
df = pd.read_csv("adult_eda.csv")
```

In [ ]:

```
```

Recognizing and Understanding Data

Content

1.Try to understand what the data looks like

- Check the head, shape, data-types of the features.
- Check if there are some dublicate rows or not. If there are, then drop them.
- Check the statistical values of features.
- If needed, rename the columns' names for easy use.
- Basically check the missing values.

In [1027]:

```
# Your Code is Here
df.head()
```

Out[1027]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	39	State-gov	77516	Bachelors	13.000	Never-married	Adm-clerical	Not-in-family	White	
1	50	Self-emp-not-inc	83311	Bachelors	13.000	Married-civ-spouse	Exec-managerial	Husband	White	
2	38	Private	215646	HS-grad	9.000	Divorced	Handlers-cleaners	Not-in-family	White	
3	53	Private	234721	11th	7.000	Married-civ-spouse	Handlers-cleaners	Husband	Black	
4	28	Private	338409	Bachelors	13.000	Married-civ-spouse	Prof-specialty	Wife	Black	F

Desired Output:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13.000	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13.000	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9.000	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7.000	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13.000	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [1028]:

```
# Your Code is Here
df.shape
```

Out[1028]:

```
(32561, 15)
```

Desired Output: (32561, 15)

In [1029]:

```
# Your Code is Here
df.info()
```

```
<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   31759 non-null  float64
 5   marital-status  32561 non-null  object 
 6   occupation      32561 non-null  object 
 7   relationship    27493 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  salary          32561 non-null  object 
dtypes: float64(1), int64(5), object(9)
memory usage: 3.7+ MB
```

Desired Output: 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 31759 non-null float64 5 marital-status 32561 non-null object 6 occupation 32561 non-null object 7 relationship 27493 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 salary 32561 non-null

object dtypes: float64(1), int64(5), object(9) memory usage: 3.7+ MB

In [1030]:

```
# Check if the Dataset have any Duplicate

# Your Code is Here
df.duplicated().value_counts()
```

Out[1030]:

```
False    32537
True        24
dtype: int64
```

Desired Output: False 32537 True 24 dtype: int64

In [1031]:

```
# Drop Duplicates

# Your Code is Here
df.drop_duplicates(inplace=True)
```

In [1032]:

```
# Check the shape of the Dataset

# Your Code is Here

df.shape
```

Out[1032]:

```
(32537, 15)
```

Desired Output: (32537, 15)

In [1033]:

```
# Your Code is Here
df.describe().T
```

Out[1033]:

	count	mean	std	min	25%	50%	75%
age	32537.000	38.586	13.638	17.000	28.000	37.000	48.000
fnlwgt	32537.000	189780.849	105556.471	12285.000	117827.000	178356.000	236993.000
education-num	31735.000	10.084	2.575	1.000	9.000	10.000	12.000
capital-gain	32537.000	1078.444	7387.957	0.000	0.000	0.000	0.000
capital-loss	32537.000	87.368	403.102	0.000	0.000	0.000	0.000
hours-per-week	32537.000	40.440	12.347	1.000	40.000	40.000	45.000

Desired Output:

	count	mean	std	min	25%	50%	75%	max
age	32537.000	38.586	13.638	17.000	28.000	37.000	48.000	90.000
fnlwgt	32537.000	189780.849	105556.471	12285.000	117827.000	178356.000	236993.000	1484705.000
education-num	31735.000	10.084	2.575	1.000	9.000	10.000	12.000	16.000
capital-gain	32537.000	1078.444	7387.957	0.000	0.000	0.000	0.000	99999.000
capital-loss	32537.000	87.368	403.102	0.000	0.000	0.000	0.000	4356.000
hours-per-week	32537.000	40.440	12.347	1.000	40.000	40.000	45.000	99.000

Rename the features of;
"education-num", "marital-status", "capital-gain", "capital-loss", "hours-per-week", "native-country" as
"education_num", "marital_status", "capital_gain", "capital_loss", "hours_per_week", "native_country", respectively and permanently.

In [1034]:

```
# Your Code is Here
df.columns = df.columns.str.replace("-", "_")
```

In [1035]:

```
# Check the sum of Missing Values per column

# Your Code is Here
df.isnull().sum()
```

Out[1035]:

```
age                  0
workclass            0
fnlwgt               0
education            0
education_num      802
marital_status       0
occupation           0
relationship      5064
race                 0
sex                  0
capital_gain         0
capital_loss         0
hours_per_week       0
native_country       0
salary               0
dtype: int64
```

age 0 workclass 0 fnlwgt 0 education 0 education_num 802 marital_status 0 occupation 0 relationship 5064 race

0 gender 0 capital_gain 0 capital_loss 0 hours_per_week 0 native_country 0 salary 0 dtype: int64

In [1036]:

```
# Check the Percentage of Missing Values

# Your Code is Here
df.isnull().sum() / df.shape[0] * 100
```

Out[1036]:

```
age               0.000
workclass         0.000
fnlwgt            0.000
education         0.000
education_num     2.465
marital_status    0.000
occupation        0.000
relationship     15.564
race              0.000
sex               0.000
capital_gain      0.000
capital_loss      0.000
hours_per_week    0.000
native_country    0.000
salary            0.000
dtype: float64
```

Desired Output: age 0.000 workclass 0.000 fnlwgt 0.000 education 0.000 education_num 2.465 marital_status 0.000 occupation 0.000 relationship 15.564 race 0.000 gender 0.000 capital_gain 0.000 capital_loss 0.000 hours_per_week 0.000 native_country 0.000 salary 0.000 dtype: float64

2.Look at the value counts of columns that have object datatype and detect strange values apart from the NaN Values

In [1037]:

```
# Your Code is Here
df.select_dtypes(include=["object"]).columns
```

Out[1037]:

```
Index(['workclass', 'education', 'marital_status', 'occupation',
       'relationship', 'race', 'sex', 'native_country', 'salary'],
      dtype='object')
```

Desired Output: Index(['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation', 'relationship', 'race', 'gender', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'salary'],

dtype='object')

In [1038]:

```
# Your Code is Here
df.select_dtypes(include=["object"]).describe().T
```

Out[1038]:

	count	unique	top	freq
workclass	32537	9	Private	22673
education	32537	16	HS-grad	10494
marital_status	32537	7	Married-civ-spouse	14970
occupation	32537	15	Prof-specialty	4136
relationship	27473	5	Husband	13187
race	32537	5	White	27795
sex	32537	2	Male	21775
native_country	32537	42	United-States	29153
salary	32537	2	<=50K	24698

Desired Output:

	count	unique	top	freq
workclass	32537	9	Private	22673
education	32537	16	HS-grad	10494
marital_status	32537	7	Married-civ-spouse	14970
occupation	32537	15	Prof-specialty	4136
relationship	27473	5	Husband	13187
race	32537	5	White	27795
gender	32537	2	Male	21775
native_country	32537	42	United-States	29153
salary	32537	2	<=50K	24698

Assign the Columns (Features) of object data type as `"object_col"`

In [1039]:

```
# Your Code is Here
object_col = df.select_dtypes(include=["object"]).columns
object_col
```

Out[1039]:

```
Index(['workclass', 'education', 'marital_status', 'occupation',
       'relationship', 'race', 'sex', 'native_country', 'salary'],
      dtype='object')
```

Desired Output: Index(['workclass', 'education', 'marital_status', 'occupation', 'relationship', 'race', 'gender',

'native_country', 'salary'], dtype='object')

In [1040]:

```
for col in object_col:
    print(col)
    print("--"*8)
    print(df[col].value_counts(dropna=False))
    print("--"*20)
```

```
workclass
----------------
Private             22673
Self-emp-not-inc     2540
Local-gov            2093
?                    1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
----------------------------------------
education
----------------
HS-grad         10494
Some-college     7282
Bachelors        5353
Masters          1722
Assoc-voc        1382
11th             1175
Assoc-acdm       1067
10th              933
7th-8th           645
Prof-school       576
9th               514
12th              433
Doctorate         413
5th-6th           332
1st-4th           166
Preschool          50
Name: education, dtype: int64
----------------------------------------
marital_status
----------------
Married-civ-spouse       14970
Never-married            10667
Divorced                  4441
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: marital_status, dtype: int64
----------------------------------------
occupation
----------------
Prof-specialty       4136
Craft-repair         4094
Exec-managerial      4065
Adm-clerical         3768
Sales                3650
Other-service        3291
Machine-op-inspct    2000
?                    1843
Transport-moving     1597
Handlers-cleaners    1369
Farming-fishing       992
Tech-support          927
Protective-serv       649
Priv-house-serv       147
Armed-Forces            9
```

```
Name: occupation, dtype: int64
----------------------------------------
relationship
----------------
Husband           13187
Not-in-family      8292
NaN                5064
Unmarried          3445
Wife               1568
Other-relative      981
Name: relationship, dtype: int64
----------------------------------------
race
----------------
White                27795
Black                 3122
Asian-Pac-Islander    1038
Amer-Indian-Eskimo     311
Other                  271
Name: race, dtype: int64
----------------------------------------
sex
----------------
Male      21775
Female    10762
Name: sex, dtype: int64
----------------------------------------
native_country
----------------
United-States                 29153
Mexico                          639
?                               582
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Japan                            62
Guatemala                        62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
France                           29
Greece                           29
Ecuador                          28
Ireland                          24
Hong                             20
```

```
Cambodia                        19
Trinadad&Tobago                 19
Laos                            18
Thailand                        18
Yugoslavia                      16
Outlying-US(Guam-USVI-etc)      14
Honduras                        13
Hungary                         13
Scotland                        12
Holand-Netherlands               1
Name: native_country, dtype: int64
----------------------------------------
salary
----------------
<=50K    24698
>50K      7839
Name: salary, dtype: int64
----------------------------------------
```

Check if the Dataset has any Question Mark `"?"`

In [1041]:

```
# Your Code is Here

df.apply(lambda x: x.astype('str').str.contains('\?')).any()
```

Out[1041]:

```
age               False
workclass          True
fnlwgt            False
education         False
education_num     False
marital_status    False
occupation         True
relationship      False
race              False
sex               False
capital_gain      False
capital_loss      False
hours_per_week    False
native_country     True
salary            False
dtype: bool
```

Desired Output: age False workclass True fnlwgt False education False education_num False marital_status False occupation True relationship False race False gender False capital_gain False capital_loss False

hours_per_week False native_country True salary False dtype: bool

Univariate & Multivariate Analysis

Examine all features (first target feature("salary"), then numeric ones, lastly categoric ones) separetly from different aspects according to target feature.

to do list for numeric features:

1. Check the boxplot to see extreme values
2. Check the histplot/kdeplot to see distribution of feature
3. Check the statistical values
4. Check the boxplot and histplot/kdeplot by "salary" levels
5. Check the statistical values by "salary" levels
6. Write down the conclusions you draw from your analysis

to do list for categoric features:

1. Find the features which contains similar values, examine the similarities and analyze them together
2. Check the count/percentage of person in each categories and visualize it with a suitable plot
3. If need, decrease the number of categories by combining similar categories
4. Check the count of person in each "salary" levels by categories and visualize it with a suitable plot
5. Check the percentage distribution of person in each "salary" levels by categories and visualize it with suitable plot
6. Check the count of person in each categories by "salary" levels and visualize it with a suitable plot
7. Check the percentage distribution of person in each categories by "salary" levels and visualize it with suitable plot
8. Write down the conclusions you draw from your analysis

Note : Instruction/direction for each feature is available under the corresponding feature in detail, as well.

Salary (Target Feature)

Check the count of person in each "salary" levels and visualize it with a countplot

In [1042]:

```
# Your Code is Here
df.salary.value_counts()
```

Out[1042]:

```
<=50K    24698
>50K      7839
Name: salary, dtype: int64
```

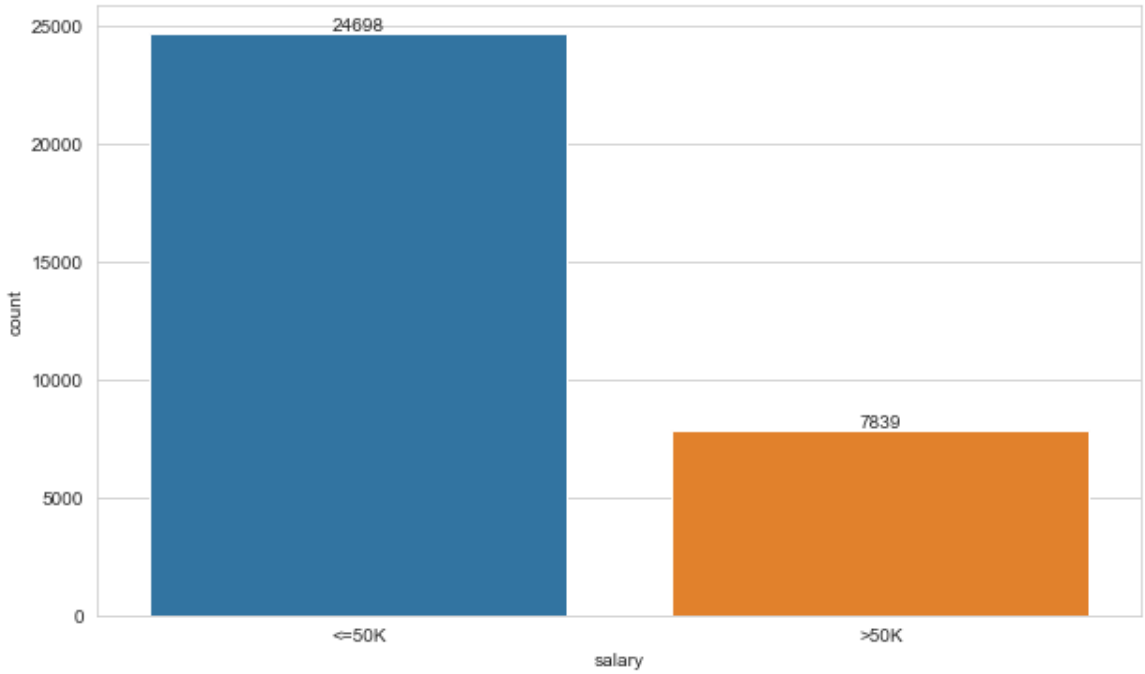
Desired Output: <=50K 24698 >50K 7839 Name: salary, dtype: int64

In [1043]:

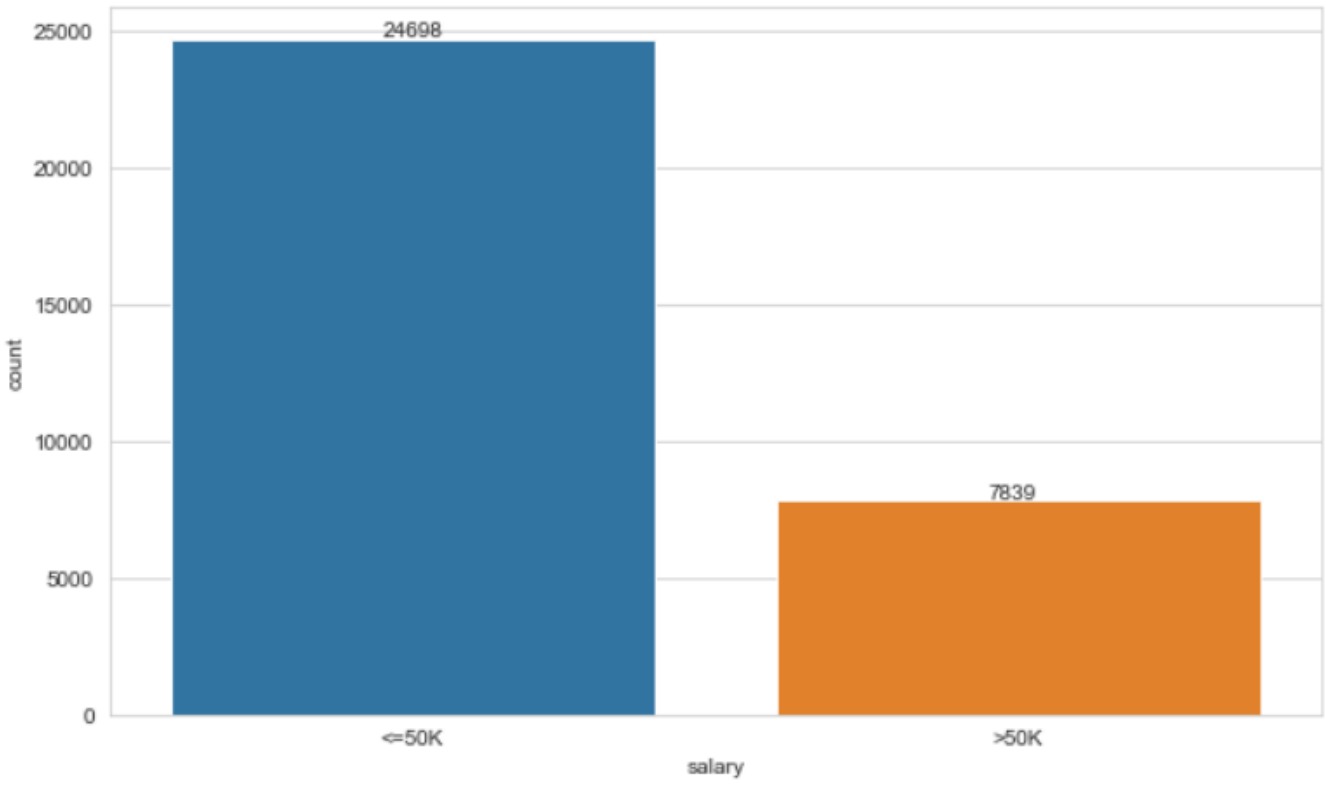
```
# Your Code is Here
g = sns.countplot(x='salary', data=df)
plt.bar_label(g.containers[0])
```

Out[1043]:

```
[Text(0, 0, '24698'), Text(0, 0, '7839')]
```



Desired Output:



Check the percentage of person in each "salary" levels and visualize it with a pieplot

In [1044]:

```
# Your Code is Here
df.salary.value_counts() / df.shape[0]
```

Out[1044]:

```
<=50K    0.759
>50K     0.241
Name: salary, dtype: float64
```

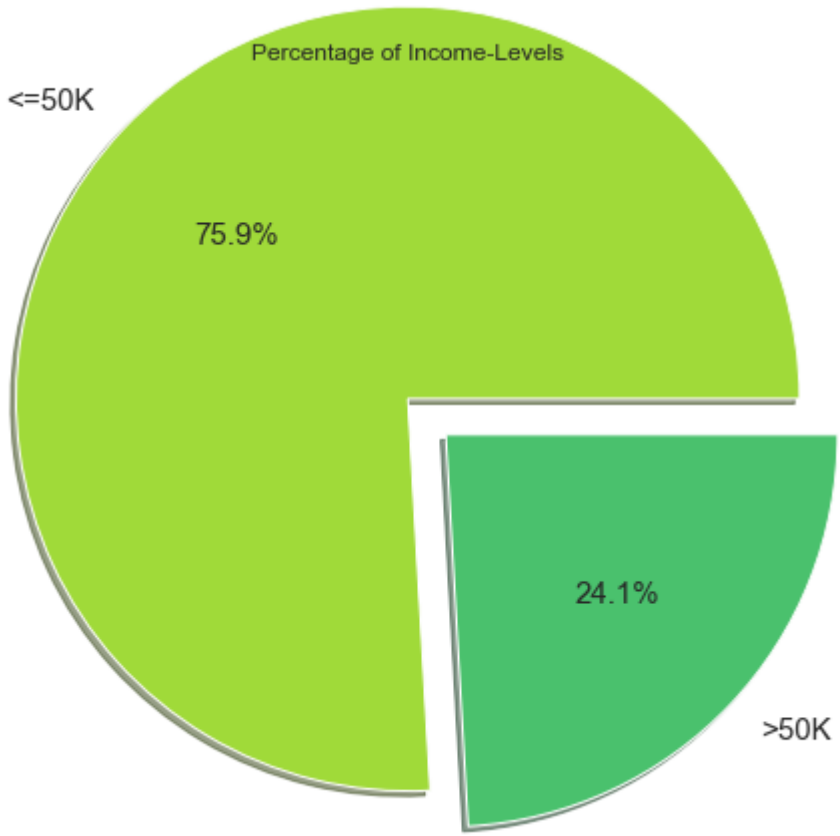
Desired Output: <=50K 0.759 >50K 0.241 Name: salary, dtype: float64

In [1045]:

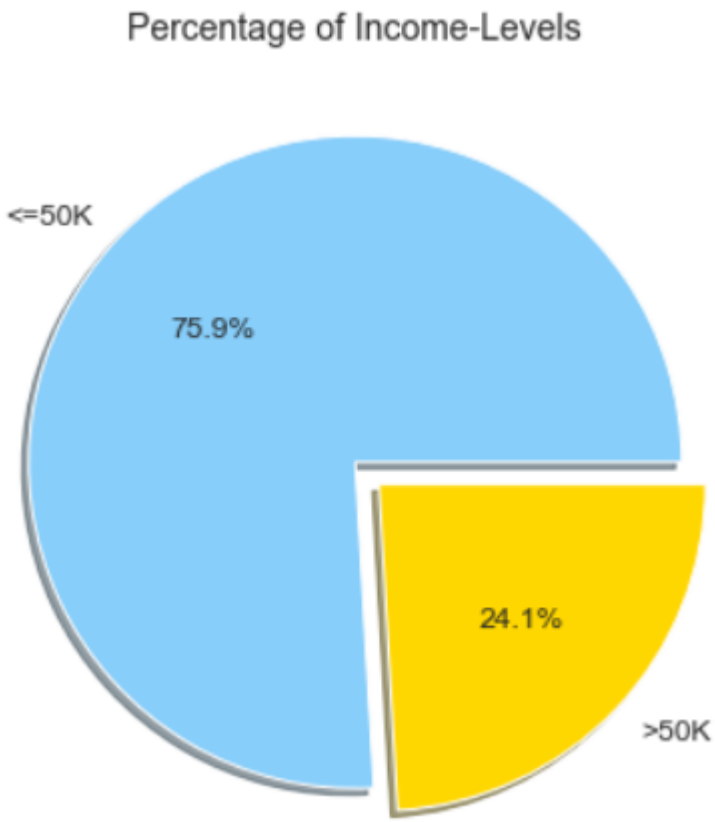
```
# Your Code is Here
viridis = sns.color_palette("viridis")
df.salary.value_counts().plot.pie(autopct='%0.1f%%', radius=1.5, labels=df.salar
y.value_counts().index, fontsize =15, shadow=True, explode=[0,0.2], colors=[viri
dis[-1],viridis[-2]])
plt.title('Percentage of Income-Levels')
```

Out[1045]:

```
Text(0.5, 1.0, 'Percentage of Income-Levels')
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : One out of every 4 people is paid over 50k, while 3 out of every 4 people are paid less than 50k

Numeric Features

age

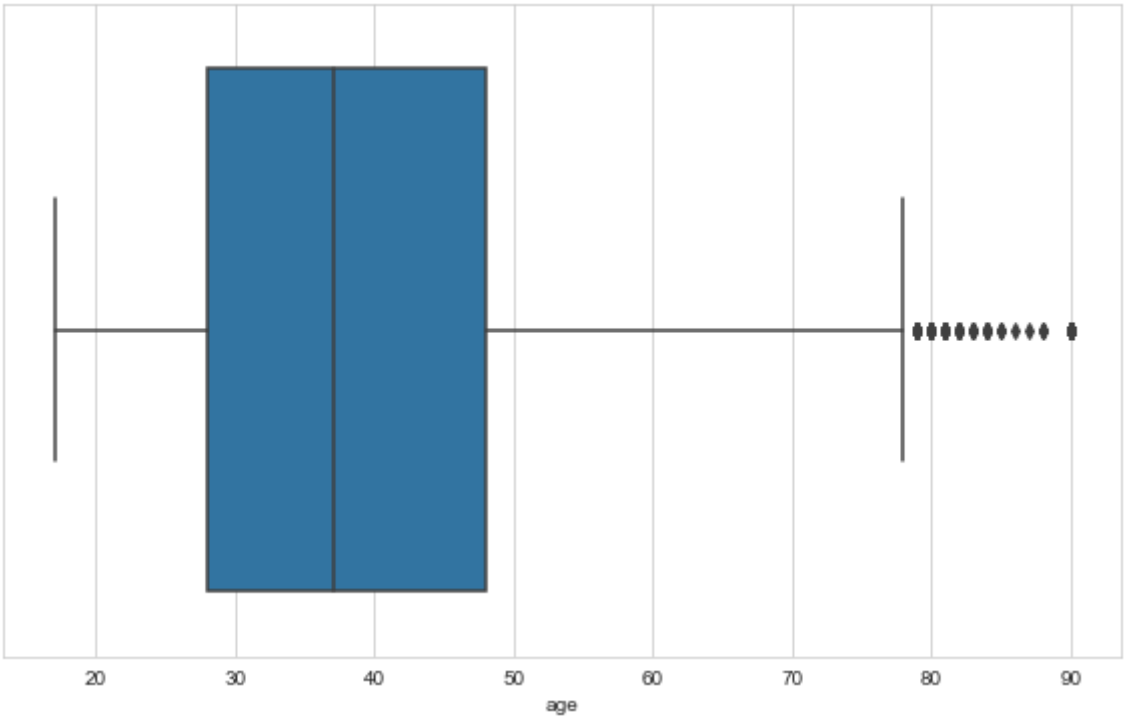
Check the boxplot to see extreme values

In [1046]:

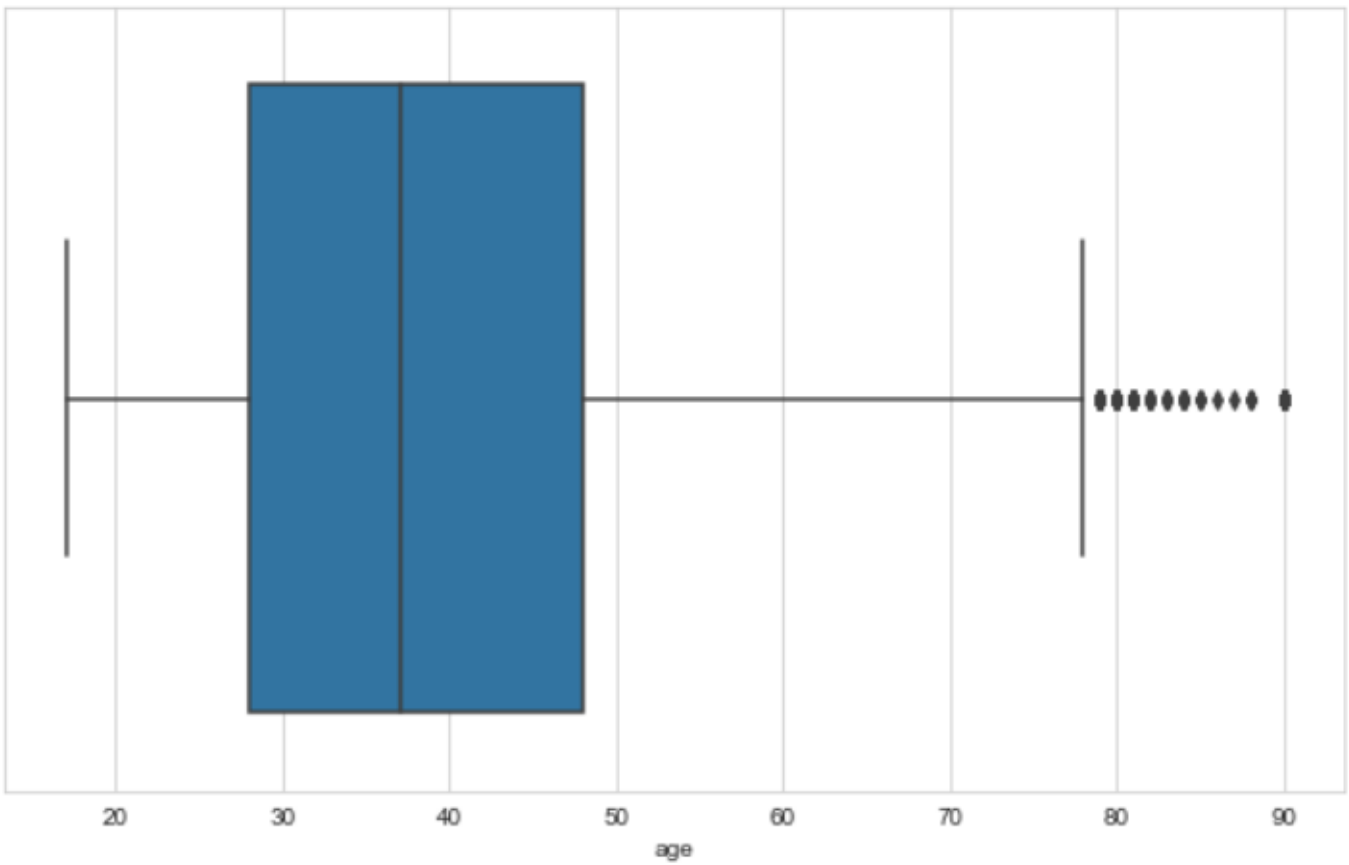
```
# Your Code is Here

sns.boxplot(x='age', data=df)
```

Out[1046]:

```
<AxesSubplot:xlabel='age'>
```



Desired Output:



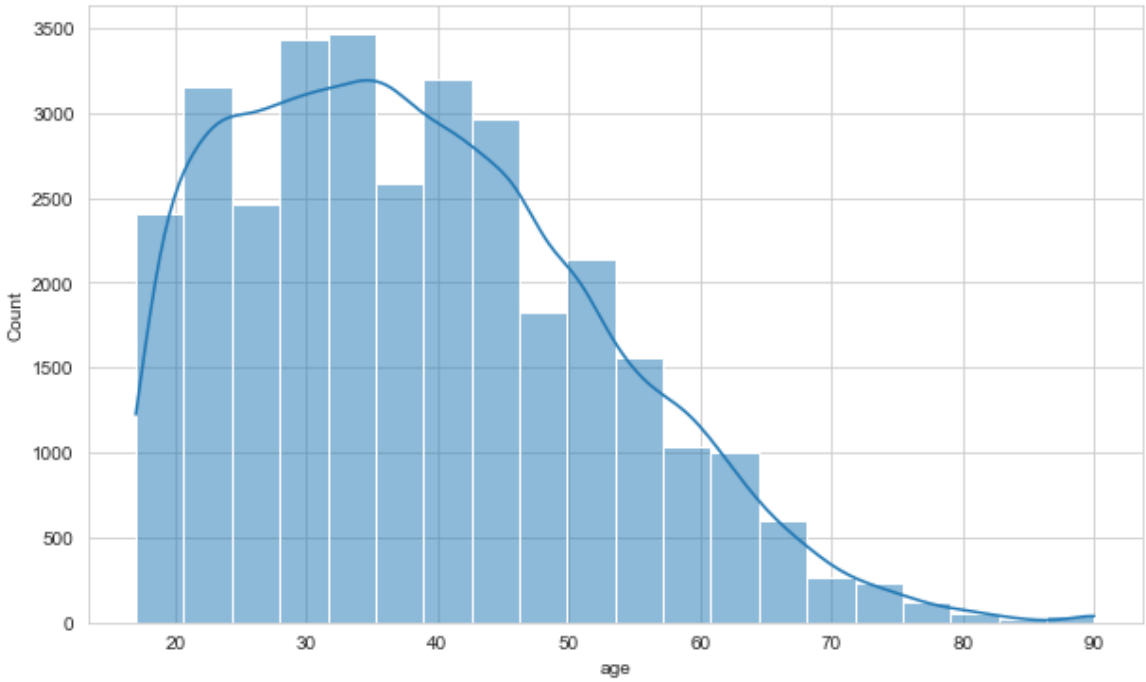
Check the histplot/kdeplot to see distribution of feature

In [1047]:

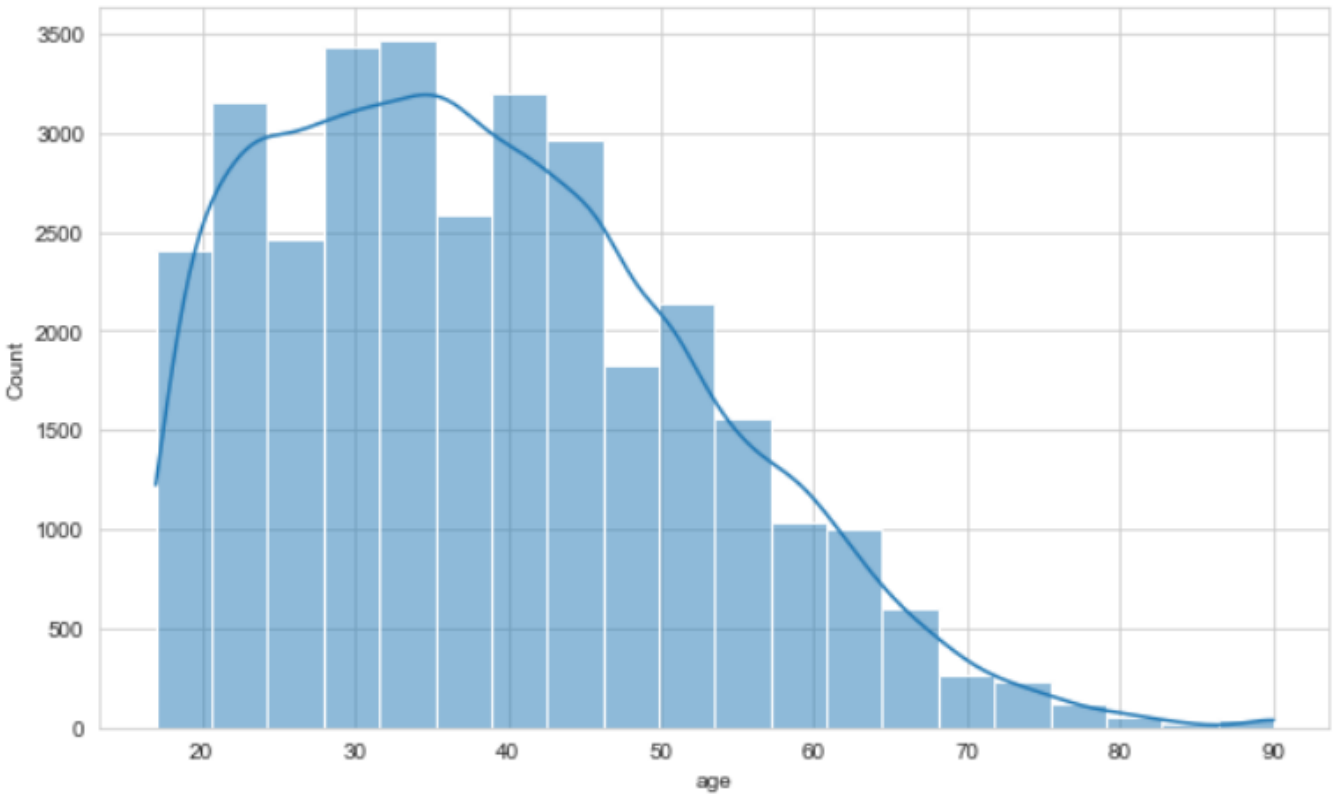
```
# Your Code is Here

sns.histplot(x='age',kde=True, data=df, bins=20)
```

Out[1047]:

```
<AxesSubplot:xlabel='age', ylabel='Count'>
```



Desired Output:



Check the statistical values

In [1048]:

```
# Your Code is Here

df.age.describe()
```

Out[1048]:

```
count   32537.000
mean       38.586
std        13.638
min        17.000
25%        28.000
50%        37.000
75%        48.000
max        90.000
Name: age, dtype: float64
```

Desired Output: count 32537.000 mean 38.586 std 13.638 min 17.000 25% 28.000 50% 37.000 75% 48.000 max 90.000 Name: age, dtype: float64

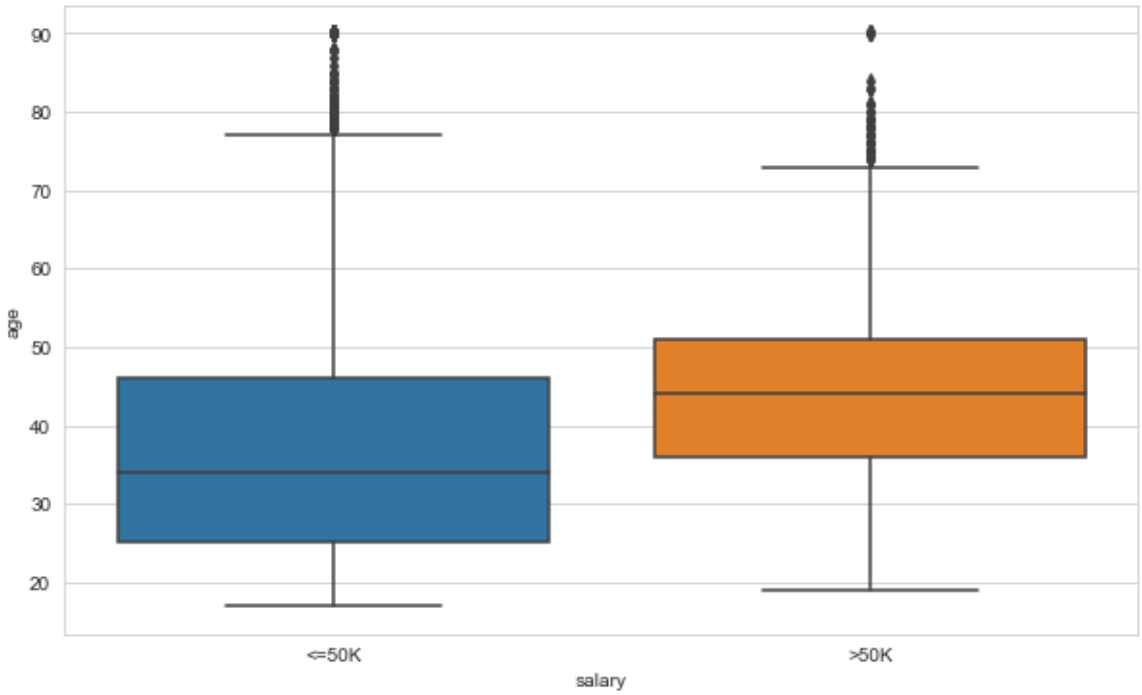
Check the boxplot and histplot/kdeplot by "salary" levels

In [1049]:

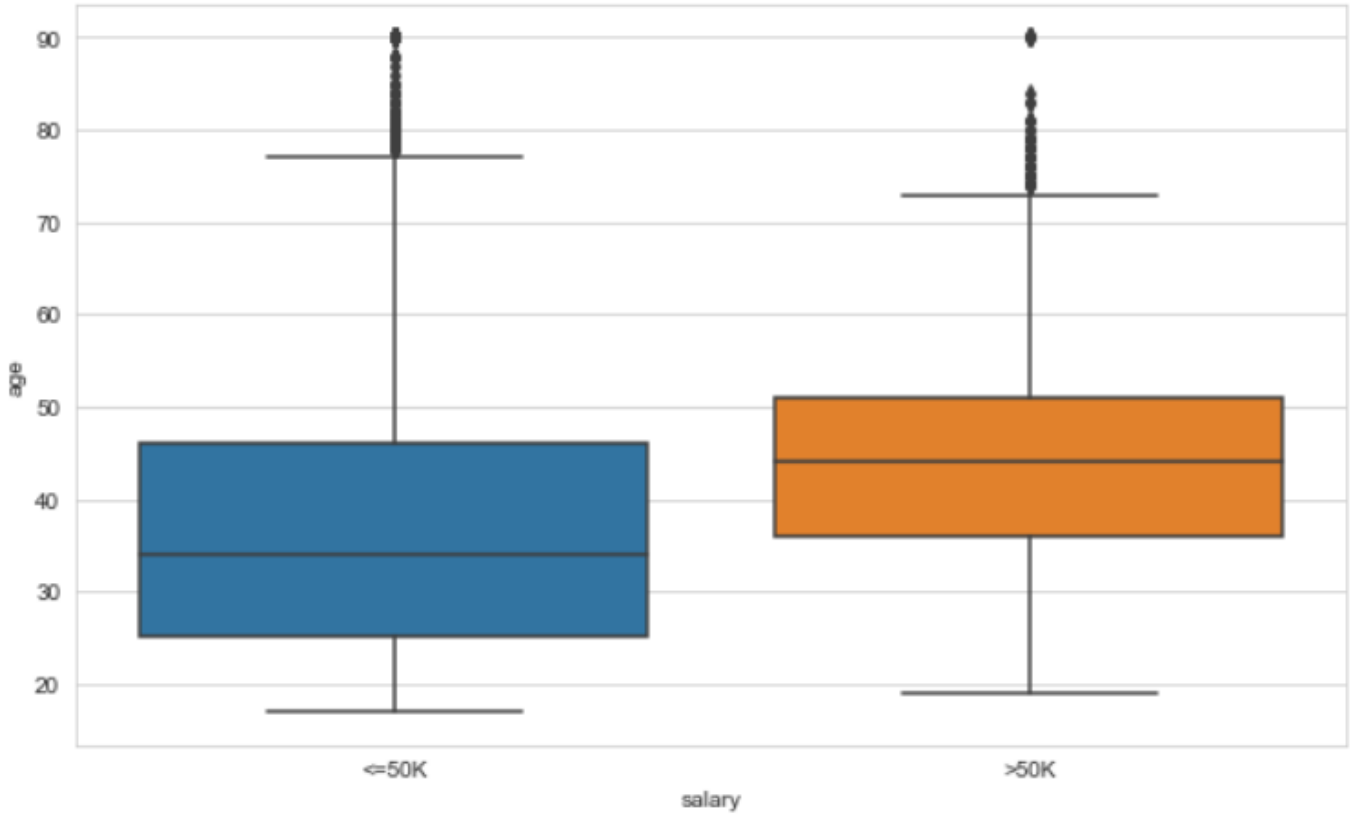
```
# Your Code is Here
sns.boxplot(x='salary', y='age', data=df)
```

Out[1049]:

```
<AxesSubplot:xlabel='salary', ylabel='age'>
```



Desired Output:

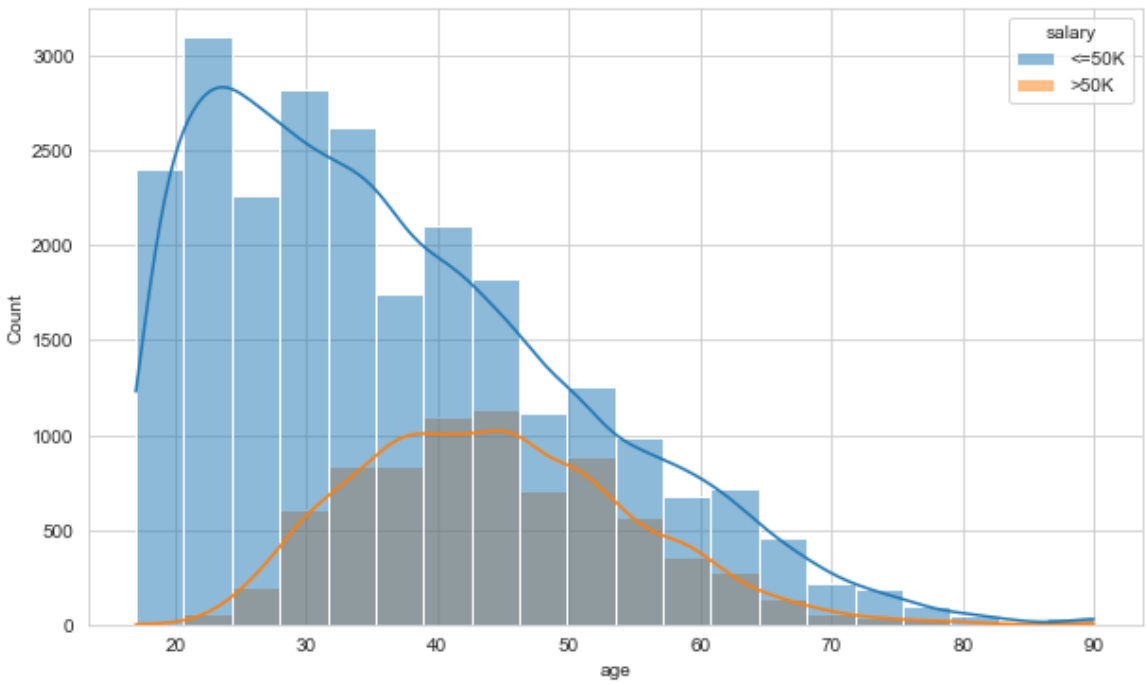


In [1050]:

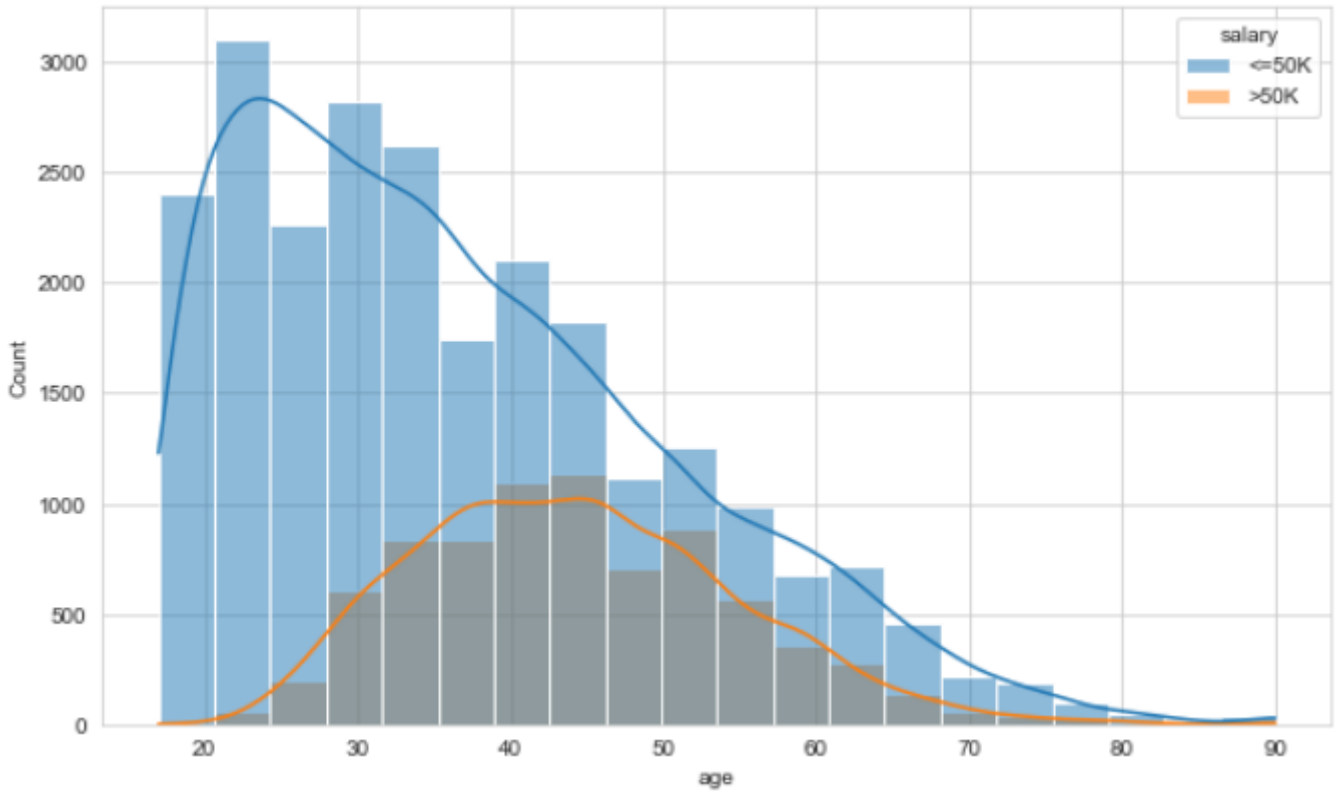
```
# Your Code is Here
sns.histplot(x='age', hue='salary',data=df,kde=True, bins =20)
```

Out[1050]:

```
<AxesSubplot:xlabel='age', ylabel='Count'>
```



Desired Output:

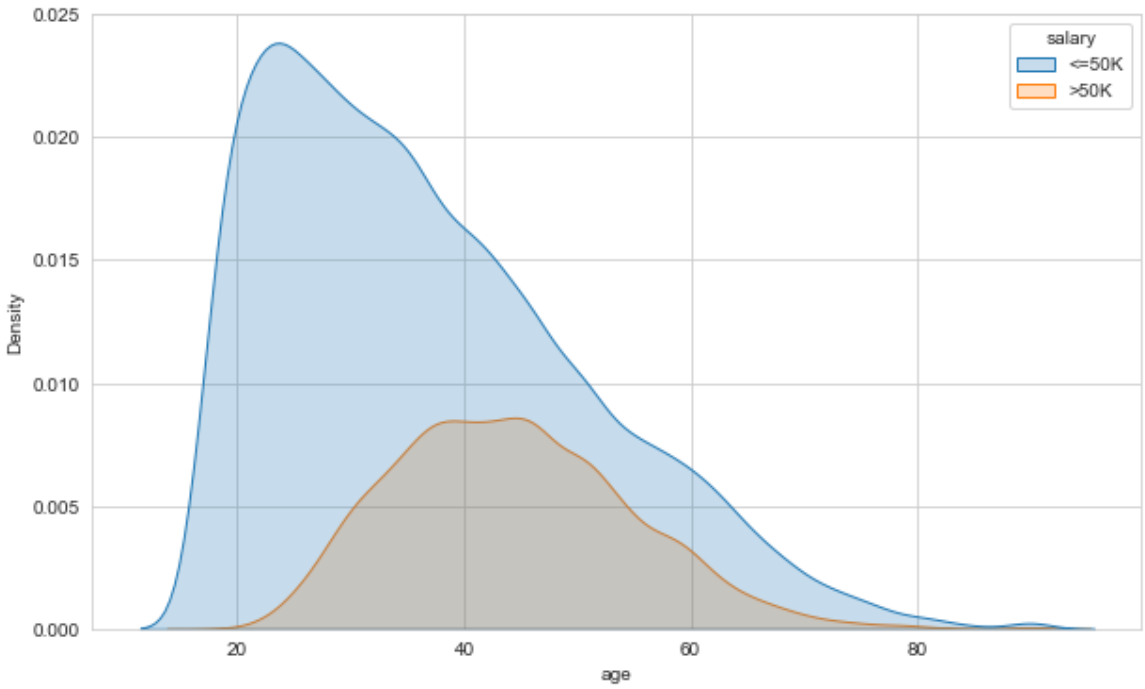


In [1051]:

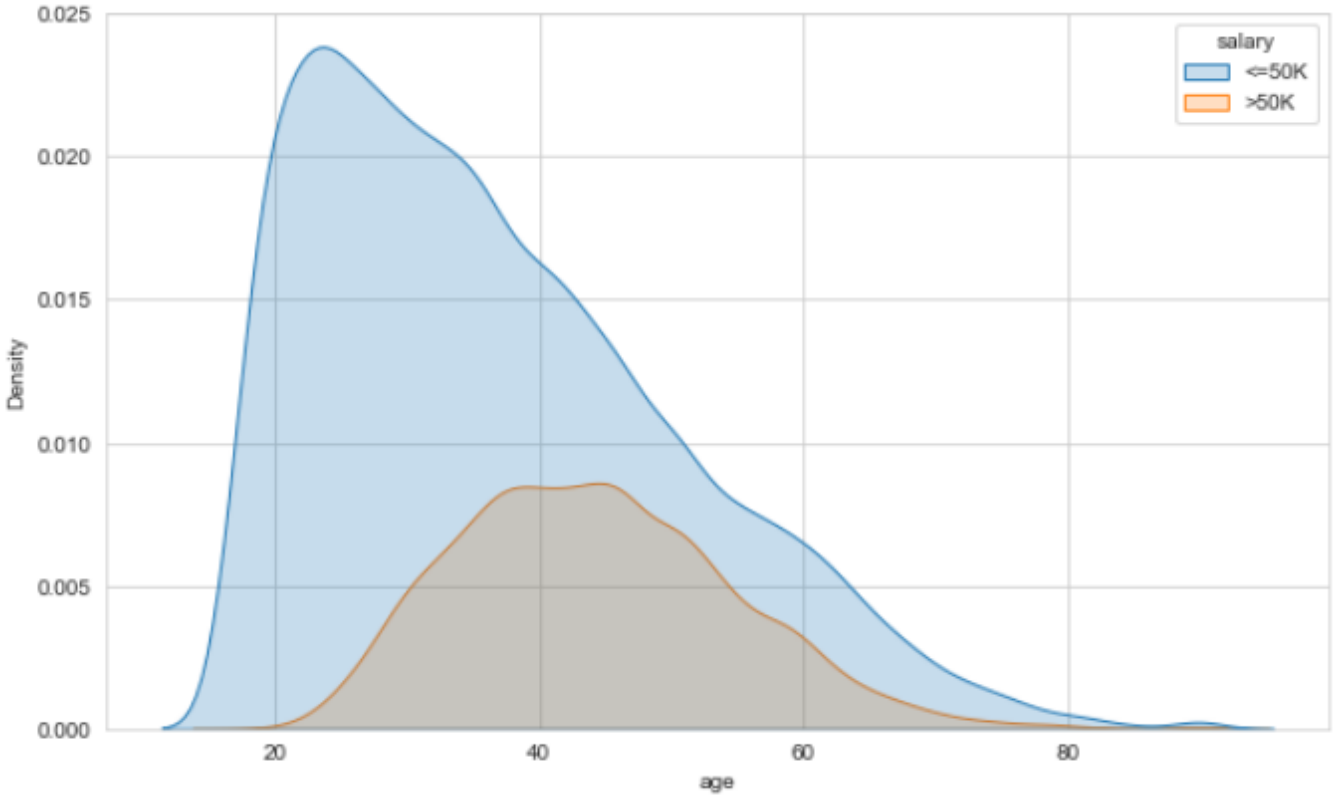
```
# Your Code is Here
sns.kdeplot(x='age',hue='salary',data=df,fill=True)
```

Out[1051]:

```
<AxesSubplot:xlabel='age', ylabel='Density'>
```



Desired Output:



Check the statistical values by "salary" levels

In [1052]:

```
# Your Code is Here
df.groupby('salary').age.describe()
```

Out[1052]:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	36.787	14.017	17.000	25.000	34.000	46.000	90.000
>50K	7839.000	44.251	10.520	19.000	36.000	44.000	51.000	90.000

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	36.787	14.017	17.000	25.000	34.000	46.000	90.000
>50K	7839.000	44.251	10.520	19.000	36.000	44.000	51.000	90.000

Write down the conclusions you draw from your analysis

Result :

The mean age of our data set is 38, the youngest is 17, and the oldest is 90 years old.
The average age of those who earn less than 50k salary is 36. 50 percent are between 25 and 46 years old.
The average age of those earning more than 50k salary is 44. 50 percent are between 36 and 51 years old.
It is observed that the distribution of those who earn less than 50k salary is right-skewed, while the distribution of those who earn more than 50k is close to the normal distribution.

fnlwgt

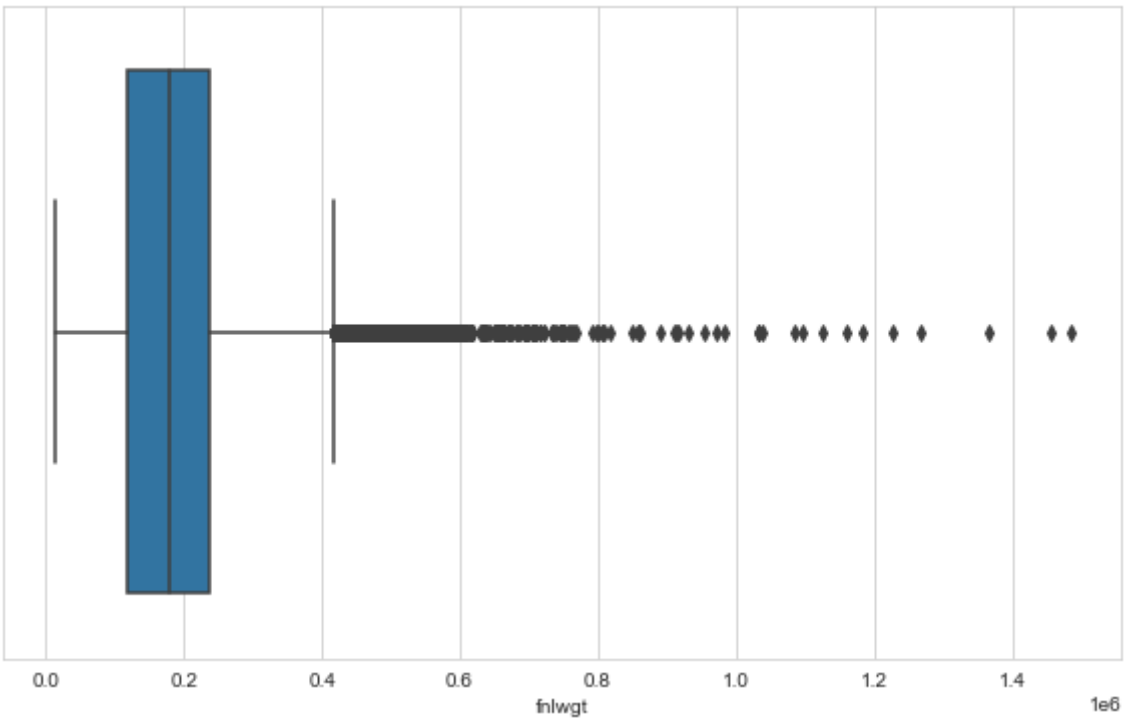
Check the boxplot to see extreme values

In [1053]:

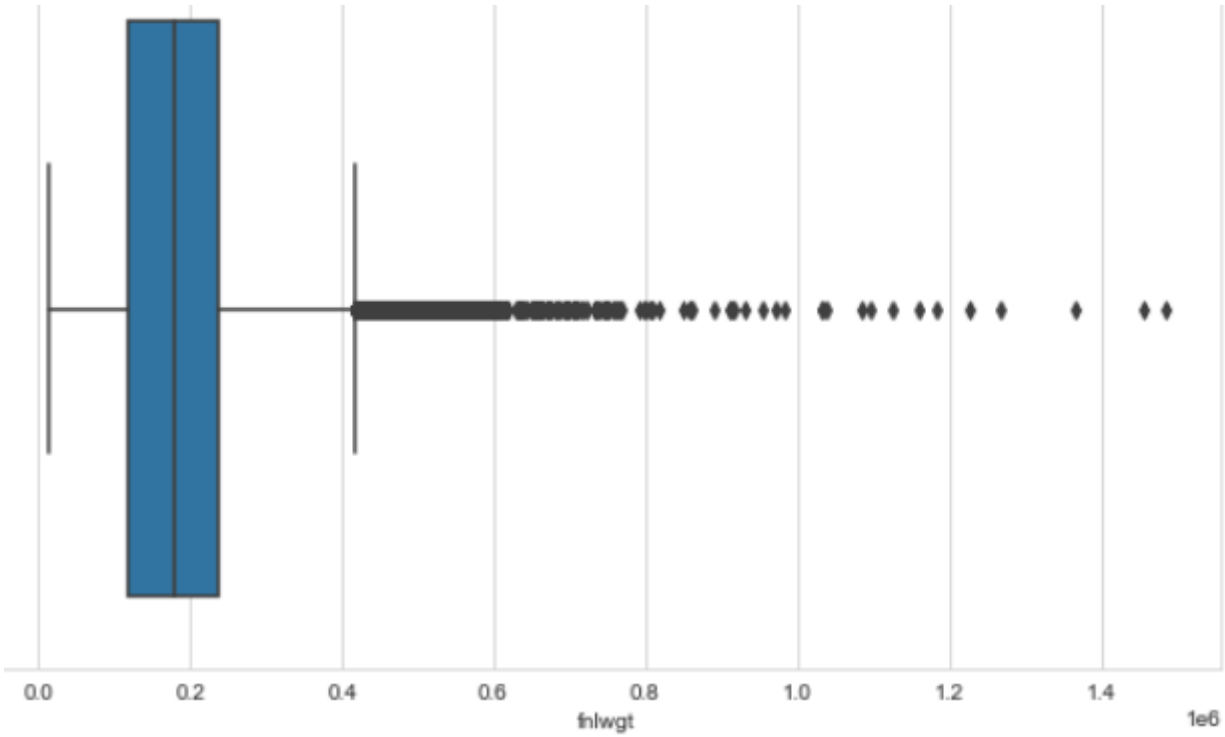
```
# Your Code is Here
sns.boxplot(x='fnlwgt', data=df)
```

Out[1053]:

```
<AxesSubplot:xlabel='fnlwgt'>
```



Desired Output:



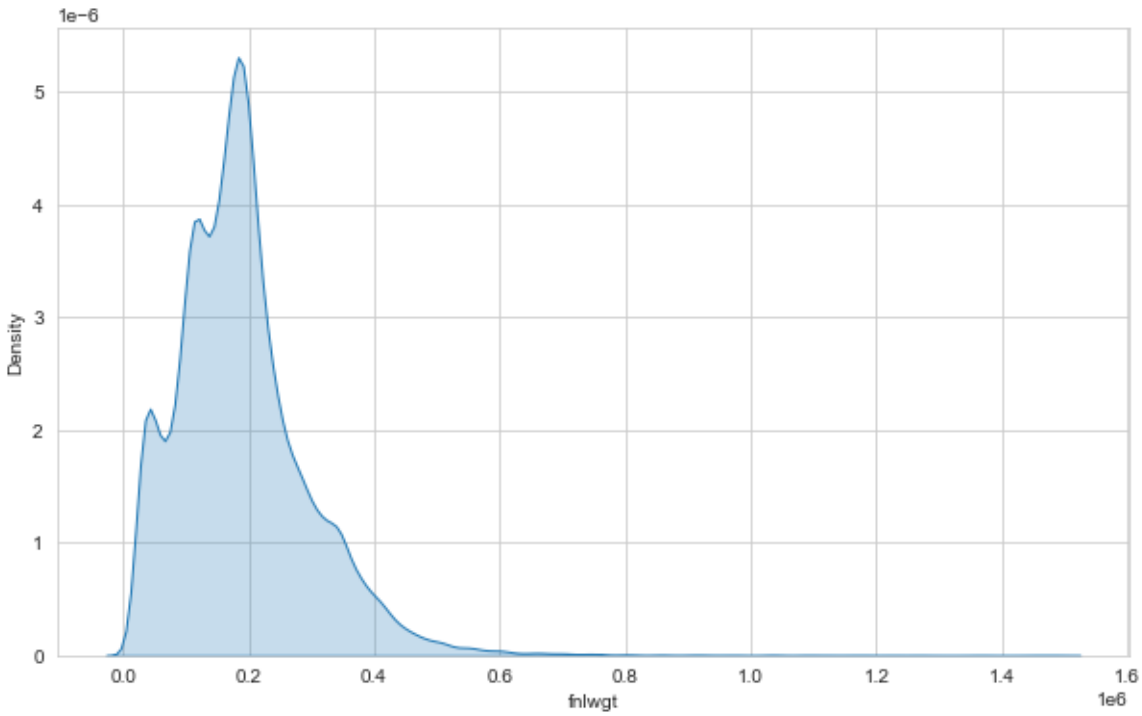
Check the histplot/kdeplot to see distribution of feature

In [1054]:

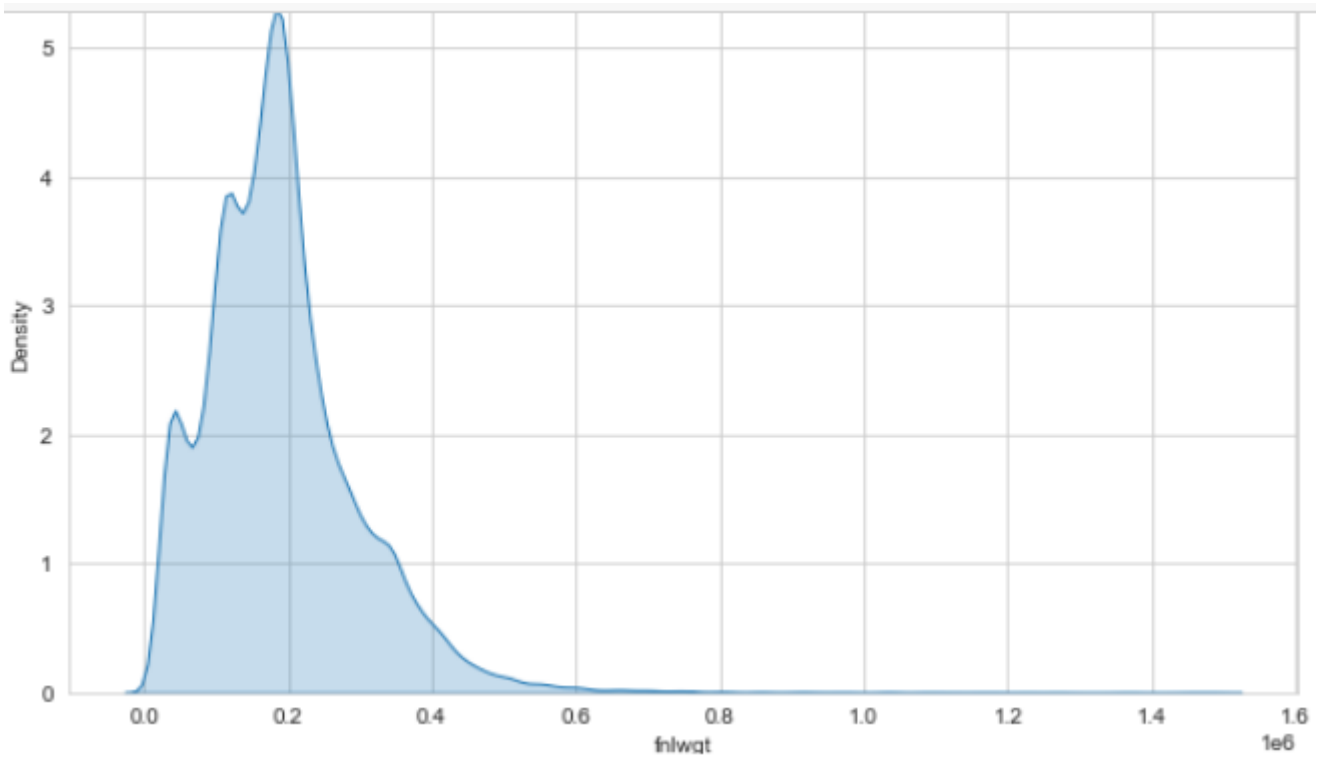
```
# Your Code is Here

sns.kdeplot(x='fnlwgt', data=df, fill=True)
```

Out[1054]:

```
<AxesSubplot:xlabel='fnlwgt', ylabel='Density'>
```



Desired Output:



Check the statistical values

In [1055]:

```
# Your Code is Here

df.fnlwgt.describe()
```

Out[1055]:

```
count     32537.000
mean     189780.849
std      105556.471
min       12285.000
25%      117827.000
50%      178356.000
75%      236993.000
max     1484705.000
Name: fnlwgt, dtype: float64
```

Desired Output: count 32537.000 mean 189780.849 std 105556.471 min 12285.000 25% 117827.000 50% 178356.000 75% 236993.000 max 1484705.000 Name: fnlwgt, dtype: float64

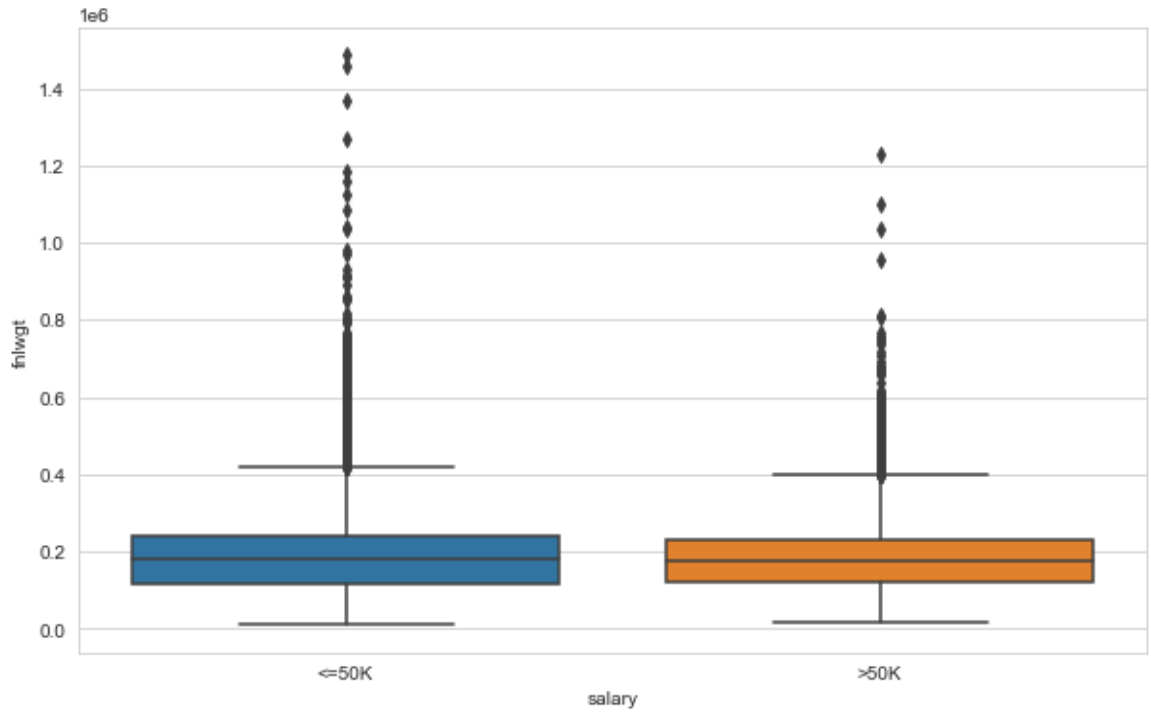
Check the boxplot and histplot/kdeplot by "salary" levels

In [1056]:

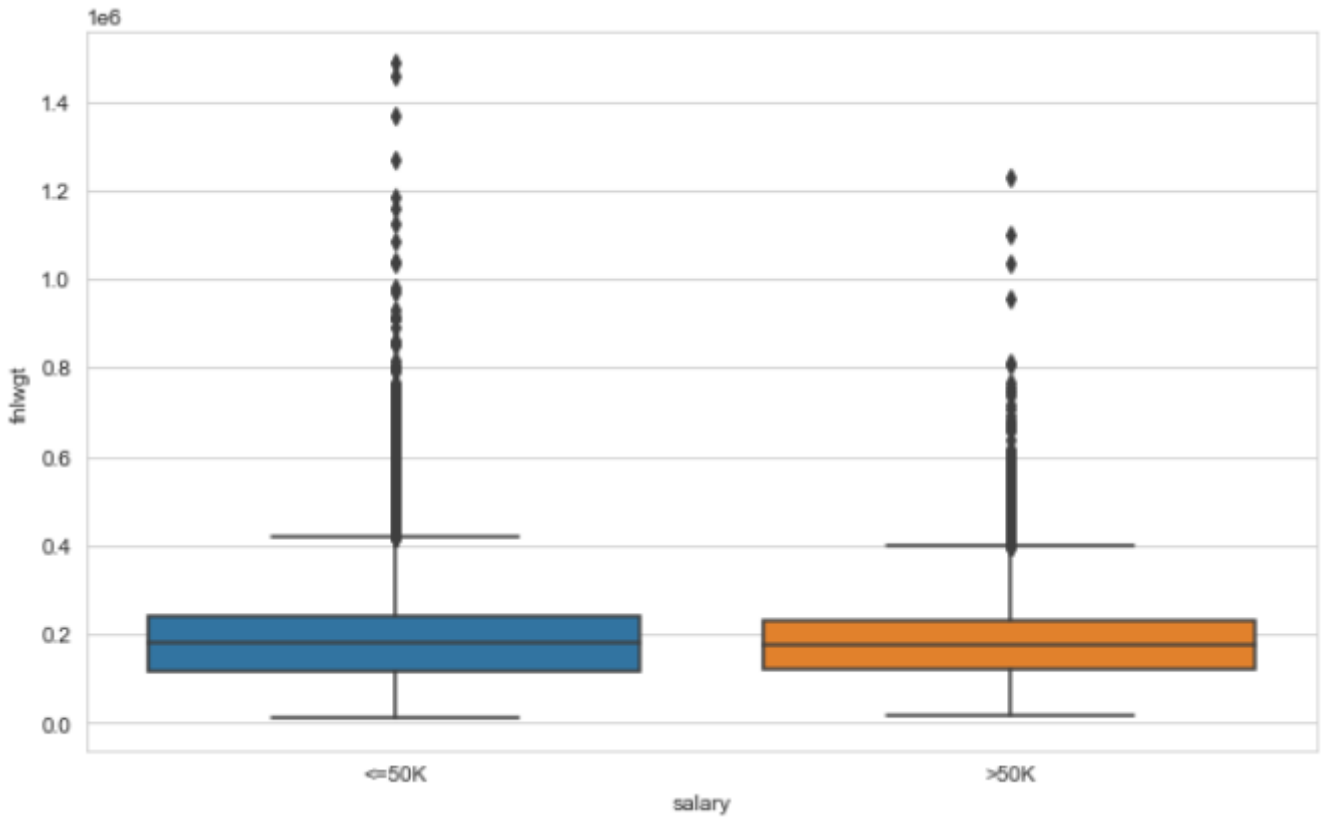
```
# Your Code is Here

sns.boxplot(x='salary', y='fnlwgt', data=df)
```

Out[1056]:

```
<AxesSubplot:xlabel='salary', ylabel='fnlwgt'>
```



Desired Output:

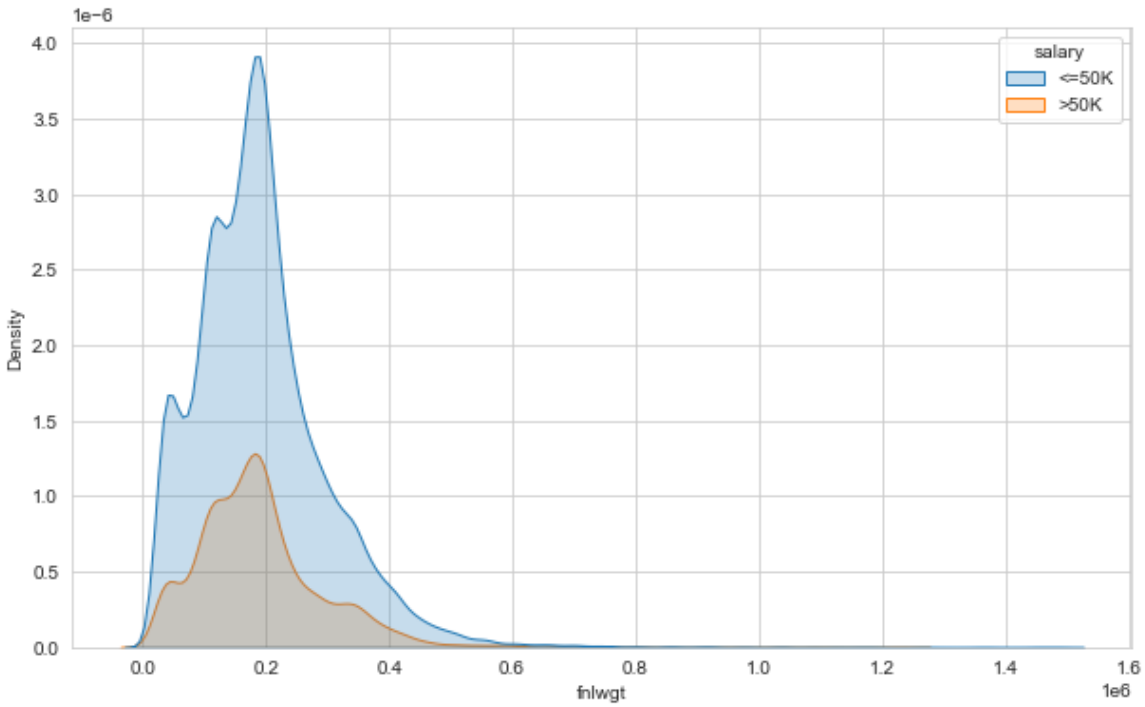


In [1057]:

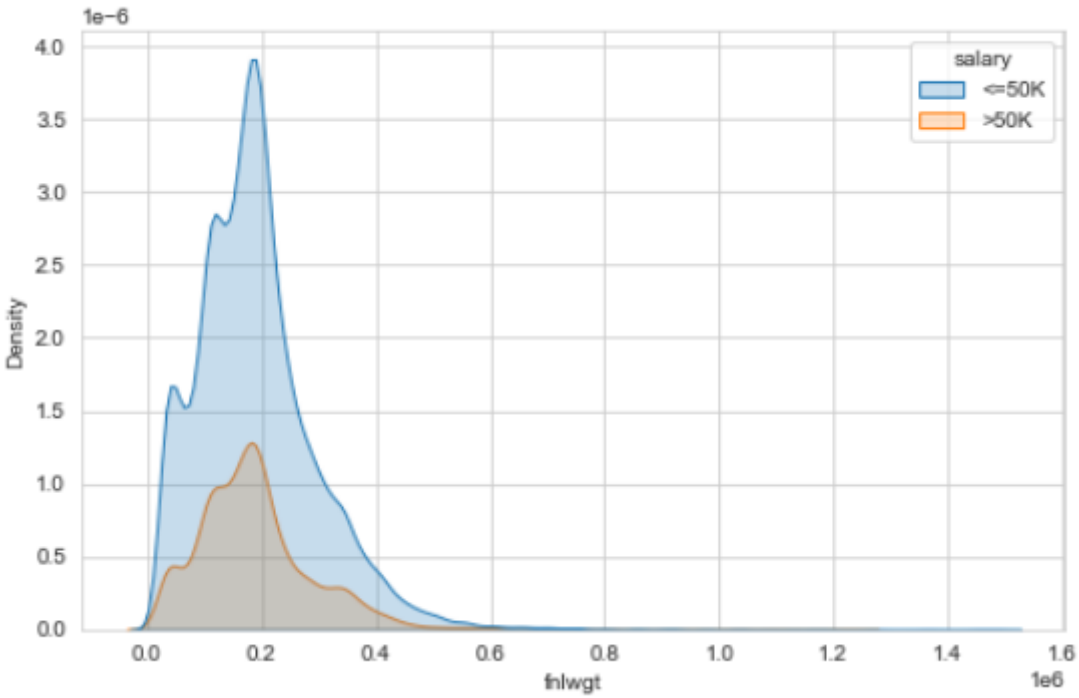
```
# Your Code is Here

sns.kdeplot(x='fnlwgt', fill=True, hue='salary', data=df)
```

Out[1057]:

```
<AxesSubplot:xlabel='fnlwgt', ylabel='Density'>
```



Desired Output:



Check the statistical values by "salary" levels

In [1058]:

```
# Your Code is Here

df.groupby('salary').fnlwgt.describe()
```

Out[1058]:

salary	count	mean	std	min	25%	50%	75%	
<=50K	24698.000	190345.927	106487.413	12285.000	117606.000	179465.000	238968.250	148
>50K	7839.000	188000.481	102554.464	14878.000	119100.000	176063.000	230969.000	122

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	190345.927	106487.413	12285.000	117606.000	179465.000	238968.250	1484705.000
>50K	7839.000	188000.481	102554.464	14878.000	119100.000	176063.000	230969.000	1226583.000

Write down the conclusions you draw from your analysis

Result : The fnlwgt variable has outliers.
fnlwgt's distribution is right-skewed.
When the distributions of fnlwgt areas below 50K and above 50K are analyzed, there are extreme values in both.
While the average of fnlwgt lower than 50K is 190,000, the average of fnlwgt higher than 50K is 188000.
The distribution of areas higher than 50K is close to the normal distribution.
The distribution of areas lower than 50K is right-skewed.

capital_gain

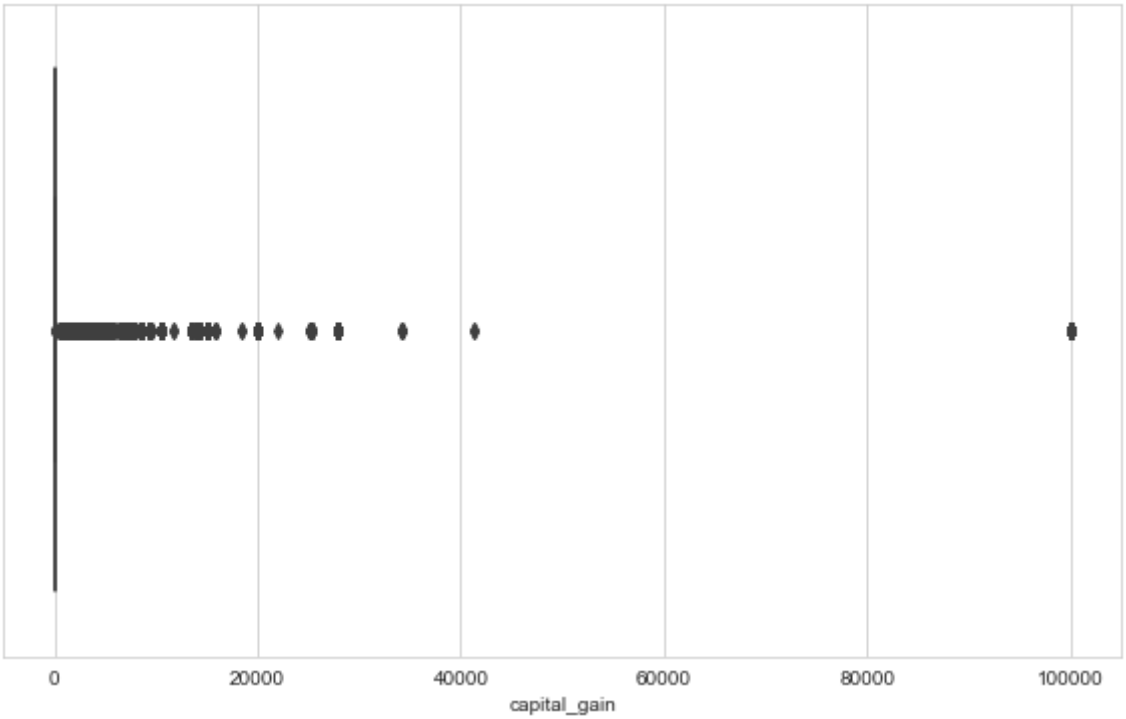
Check the boxplot to see extreme values

In [1059]:

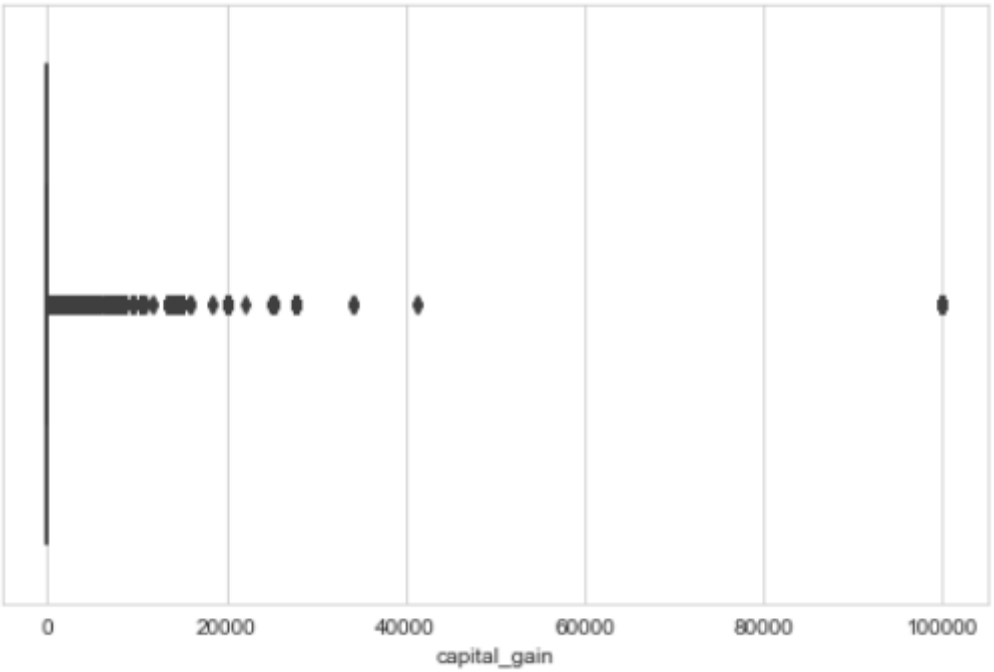
```
# Your Code is Here
sns.boxplot(x='capital_gain',data=df)
```

Out[1059]:

```
<AxesSubplot:xlabel='capital_gain'>
```



Desired Output:



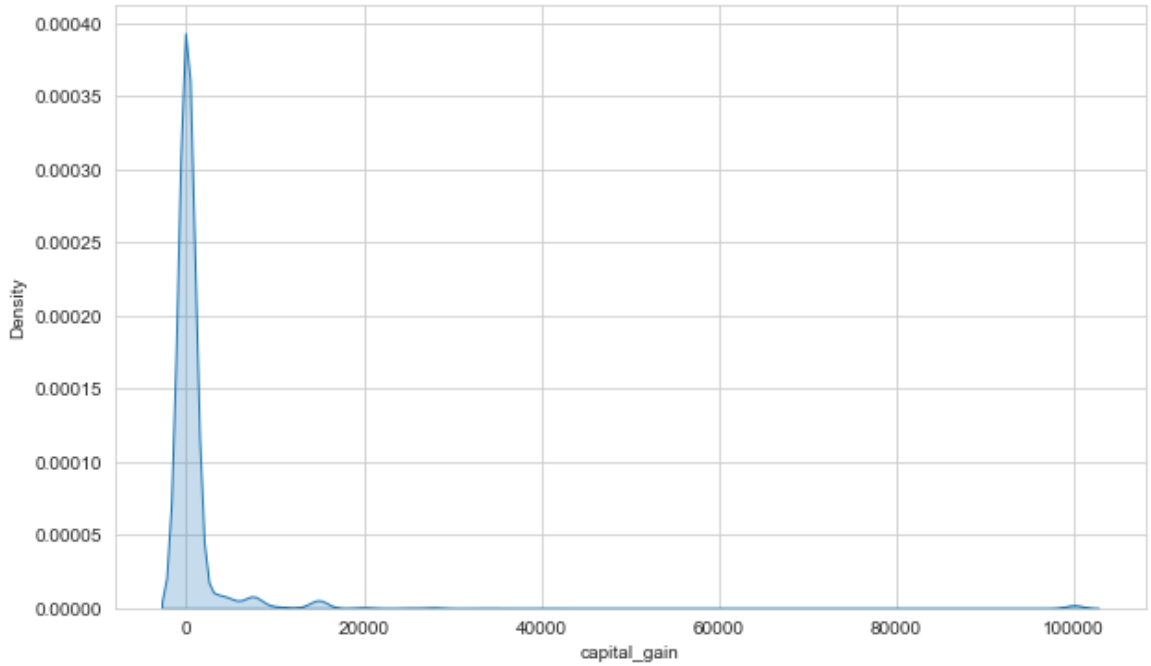
Check the histplot/kdeplot to see distribution of feature

In [1060]:

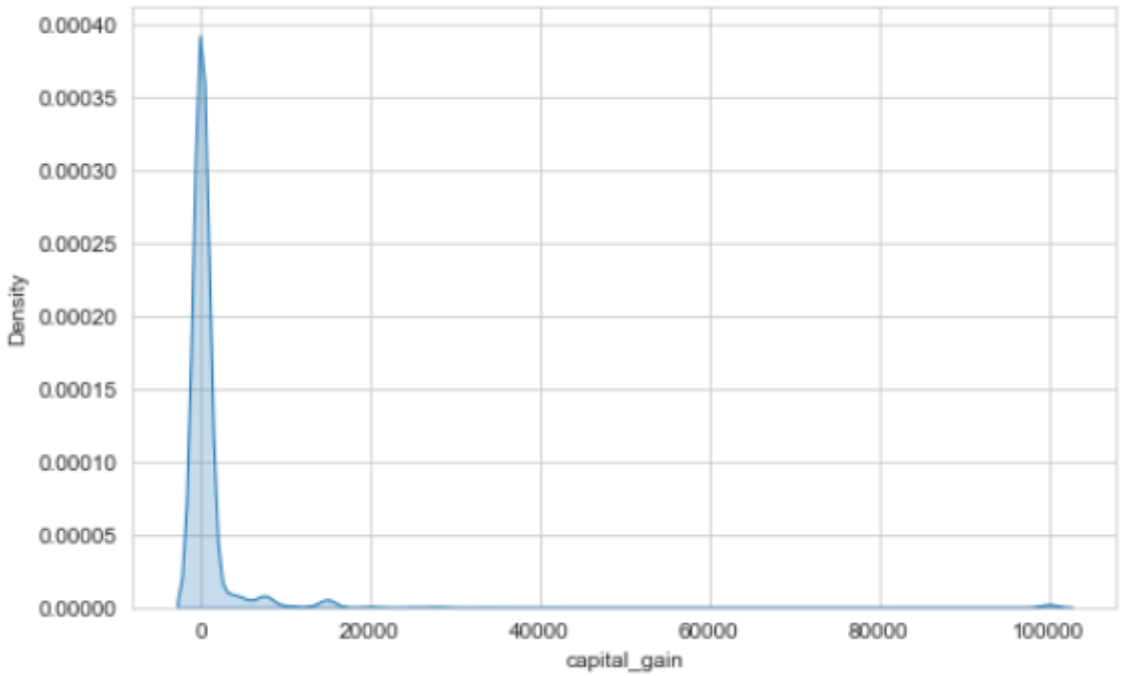
```
# Your Code is Here

sns.kdeplot(x='capital_gain', data=df, fill=True)
```

Out[1060]:

```
<AxesSubplot:xlabel='capital_gain', ylabel='Density'>
```



Desired Output:



Check the statistical values

In [1061]:

```
# Your Code is Here

df.capital_gain.describe()
```

Out[1061]:

```
count   32537.000
mean     1078.444
std      7387.957
min         0.000
25%         0.000
50%         0.000
75%         0.000
max     99999.000
Name: capital_gain, dtype: float64
```

Desired Output: count 32537.000 mean 1078.444 std 7387.957 min 0.000 25% 0.000 50% 0.000 75% 0.000 max 99999.000 Name: capital_gain, dtype: float64

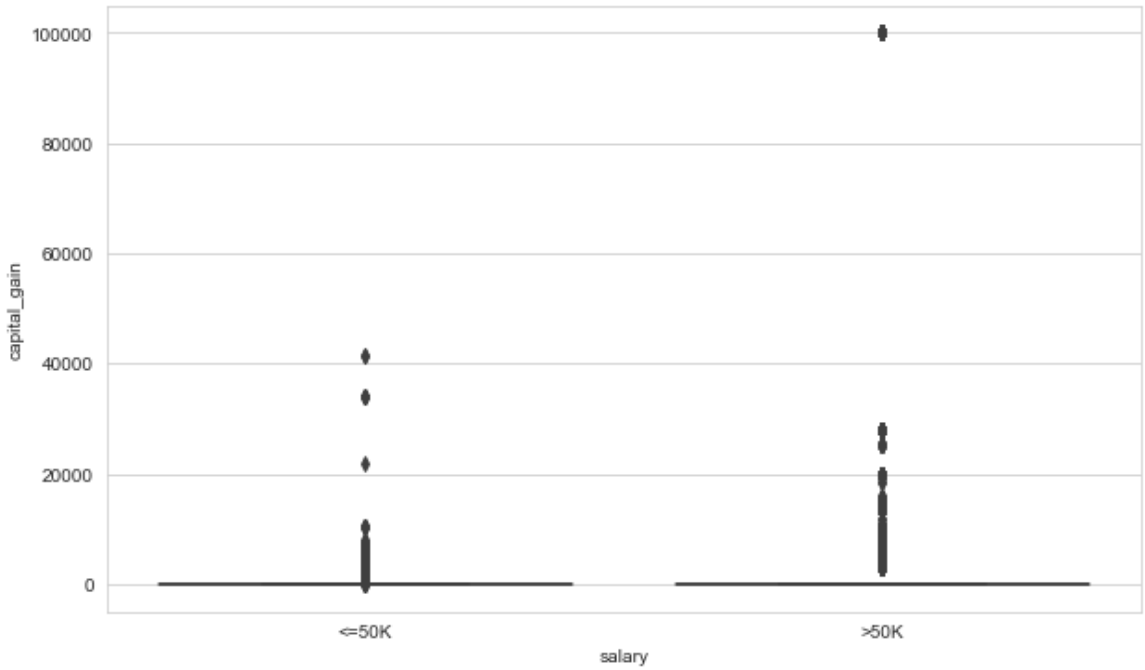
Check the boxplot and histplot/kdeplot by "salary" levels

In [1062]:

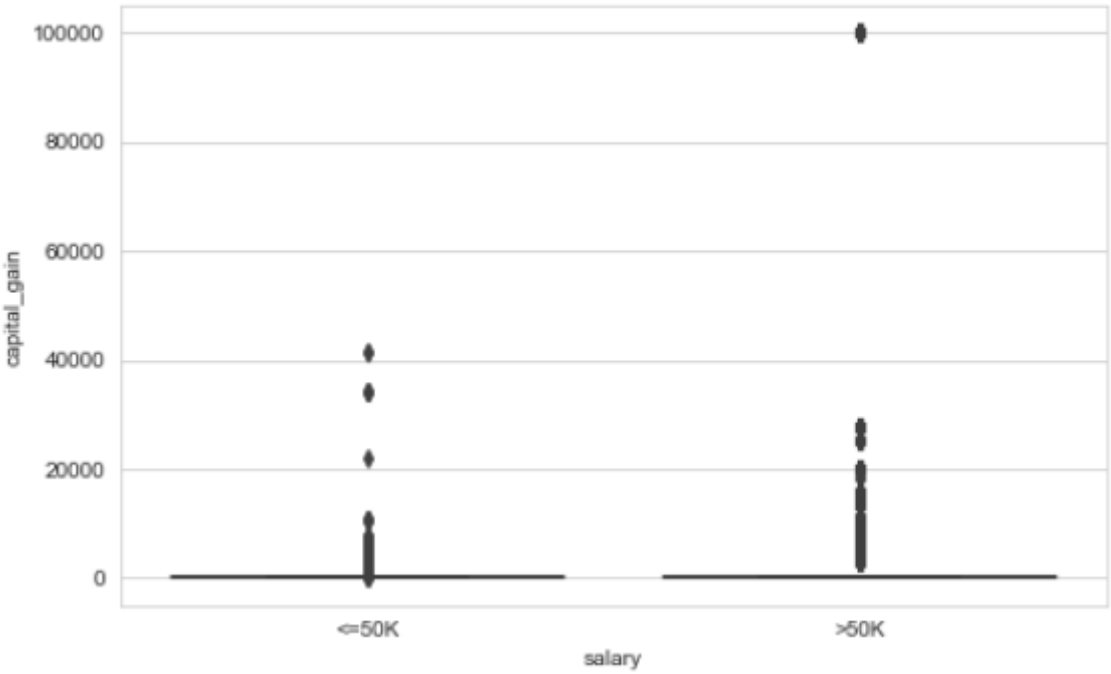
```
# Your Code is Here

sns.boxplot(x='salary', y='capital_gain', data=df)
```

Out[1062]:

```
<AxesSubplot:xlabel='salary', ylabel='capital_gain'>
```



Desired Output:

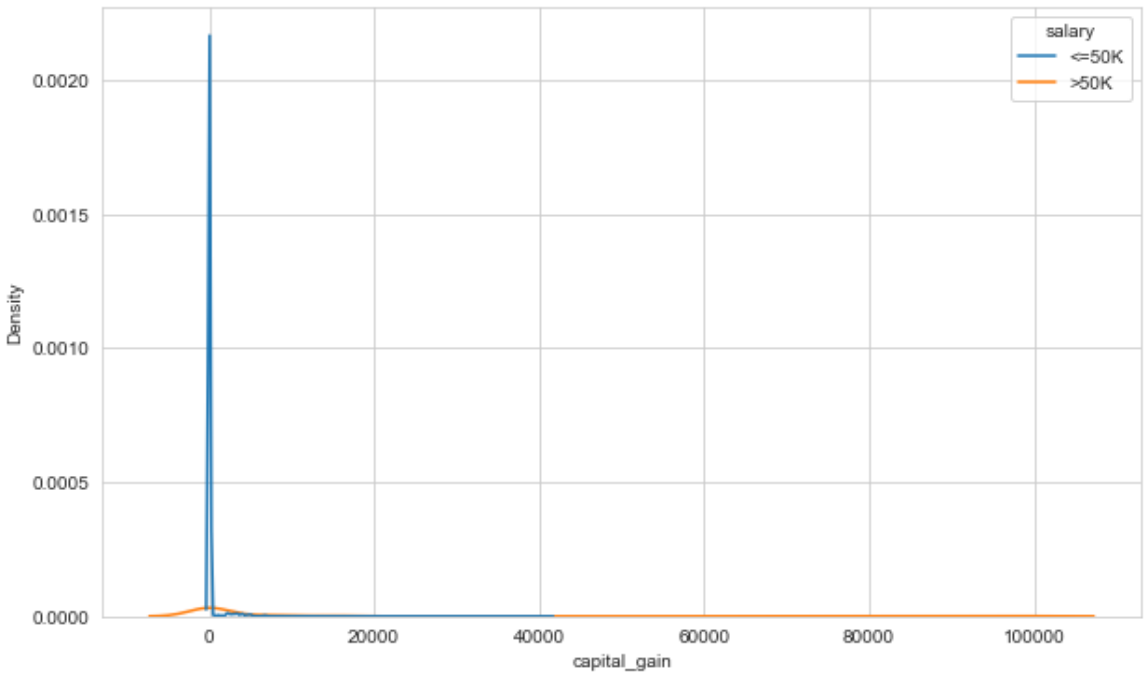


In [1063]:

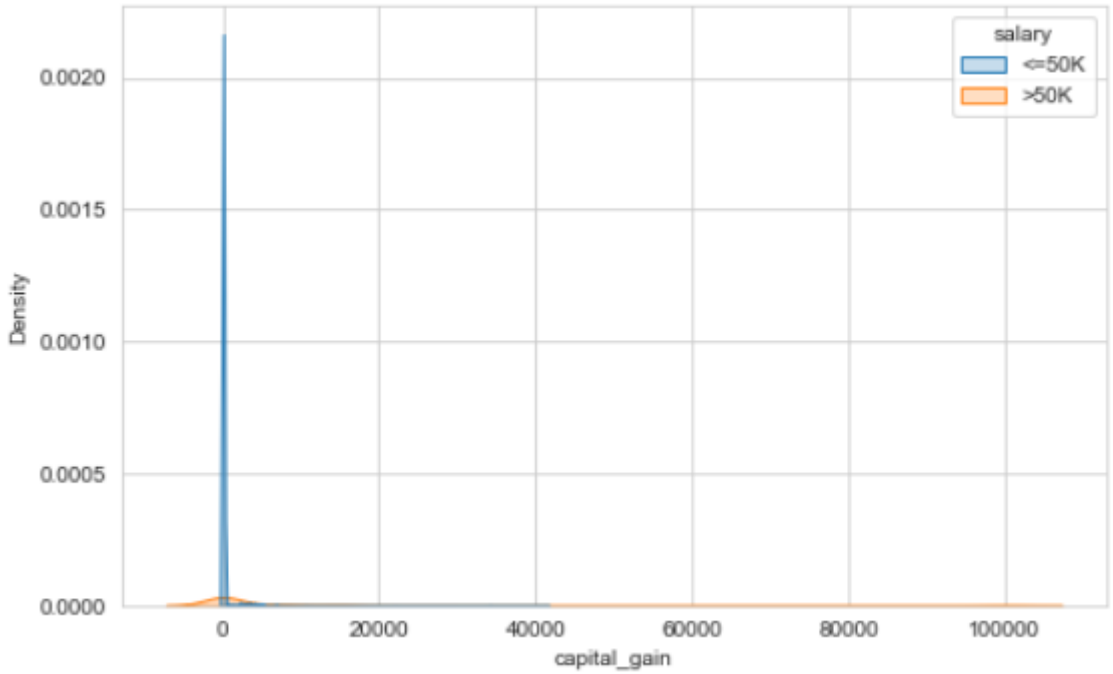
```
# Your Code is Here

sns.kdeplot(x='capital_gain', hue='salary', data=df)
```

Out[1063]:

```
<AxesSubplot:xlabel='capital_gain', ylabel='Density'>
```



Desired Output:



Check the statistical values by "salary" levels

In [1064]:

```
# Your Code is Here
df.groupby('salary').capital_gain.describe()
```

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	148.885	963.558	0.000	0.000	0.000	0.000	41310.000
>50K	7839.000	4007.165	14572.097	0.000	0.000	0.000	0.000	99999.000

Check the statistical values by "salary" levels for capital_gain not equal the zero

In [1065]:

```
# Your Code is Here

df[df.capital_gain != 0].groupby('salary').capital_gain.describe()
```

Out[1065]:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	1035.000	3552.813	3173.419	114.000	2202.000	3273.000	4101.000	41310.000
>50K	1677.000	18731.165	26778.676	3103.000	7298.000	7896.000	15024.000	99999.000

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	1035.000	3552.813	3173.419	114.000	2202.000	3273.000	4101.000	41310.000
>50K	1677.000	18731.165	26778.676	3103.000	7298.000	7896.000	15024.000	99999.000

Write down the conclusions you draw from your analysis

Result : The capital_gain variable has outlier values.
The capital_gain histogram is right-skewed.
Its average is 1078, 25 percent is 0. 75 percent is zero.
Its minimum is zero while its maximum is 99999.
When viewed at the Salary level, there are outliers in both parts and they are gathered around 0 in general.
The average of below 50K is 148. 25 percent is 0. 75 percent is zero.
The average of over 50K is 4007. 25 percent is 0. 75 percent is zero.
Under 50K earners with no capital_gain 0, average is 3552, 50 percent of them is between 2202 and 4101.
Earnings over 50K without a capital_gain 0, average is 18731, 50 percent of them is between 7298 and 15024.

capital_loss

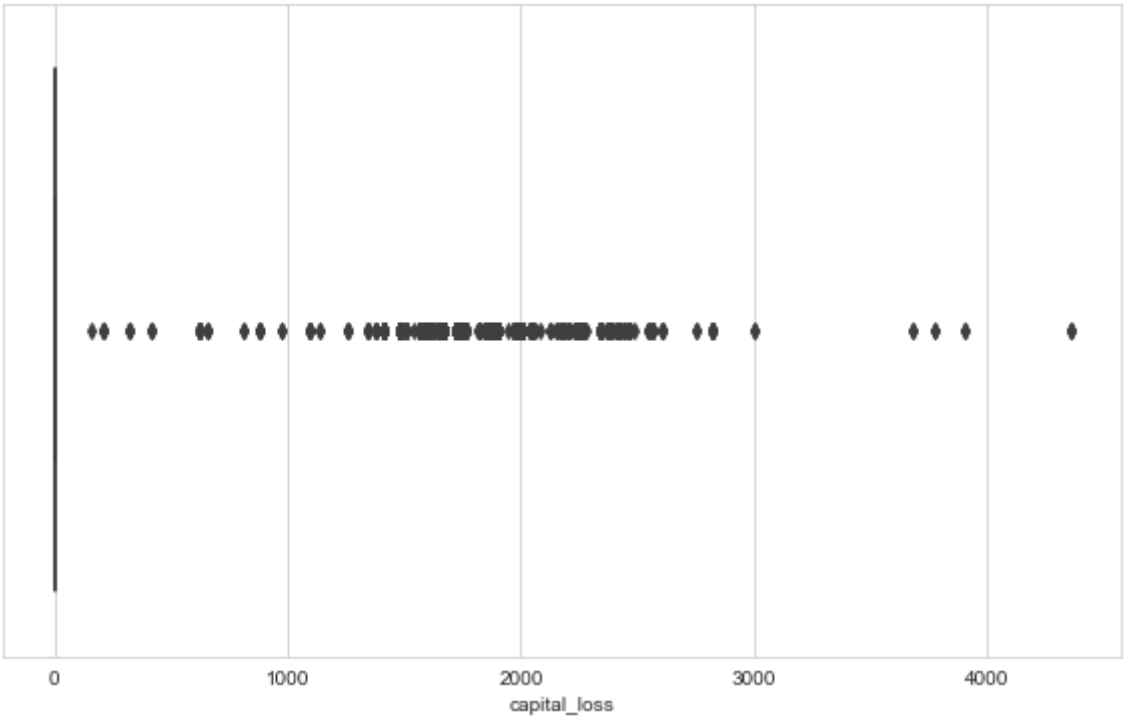
Check the boxplot to see extreme values

In [1066]:

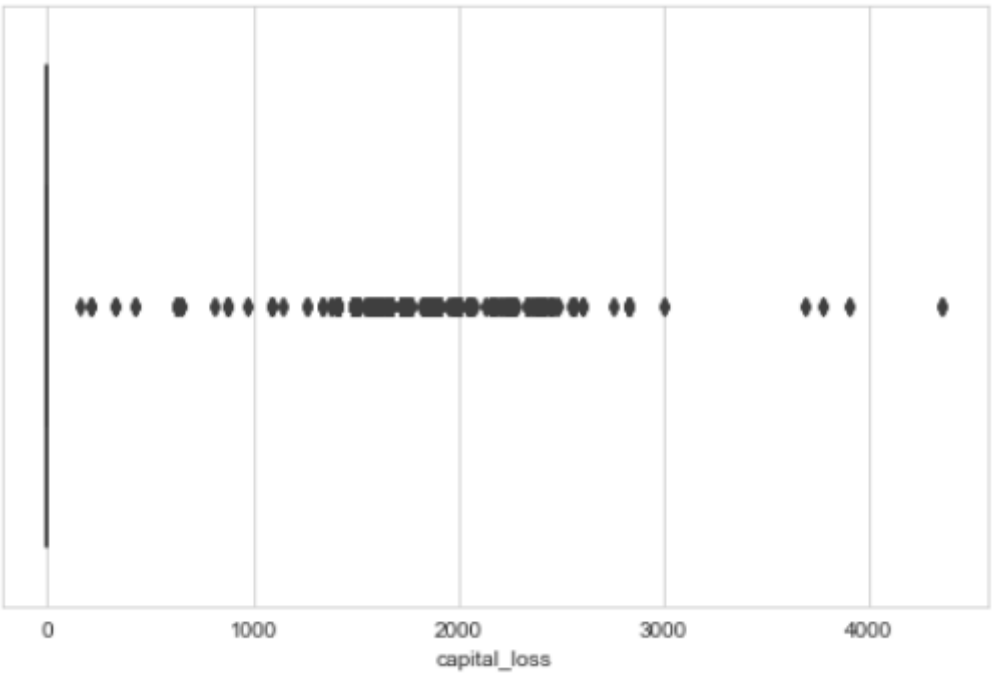
```
# Your Code is Here

sns.boxplot(x='capital_loss',data=df)
```

Out[1066]:

<AxesSubplot:xlabel='capital_loss'>



Desired Output:



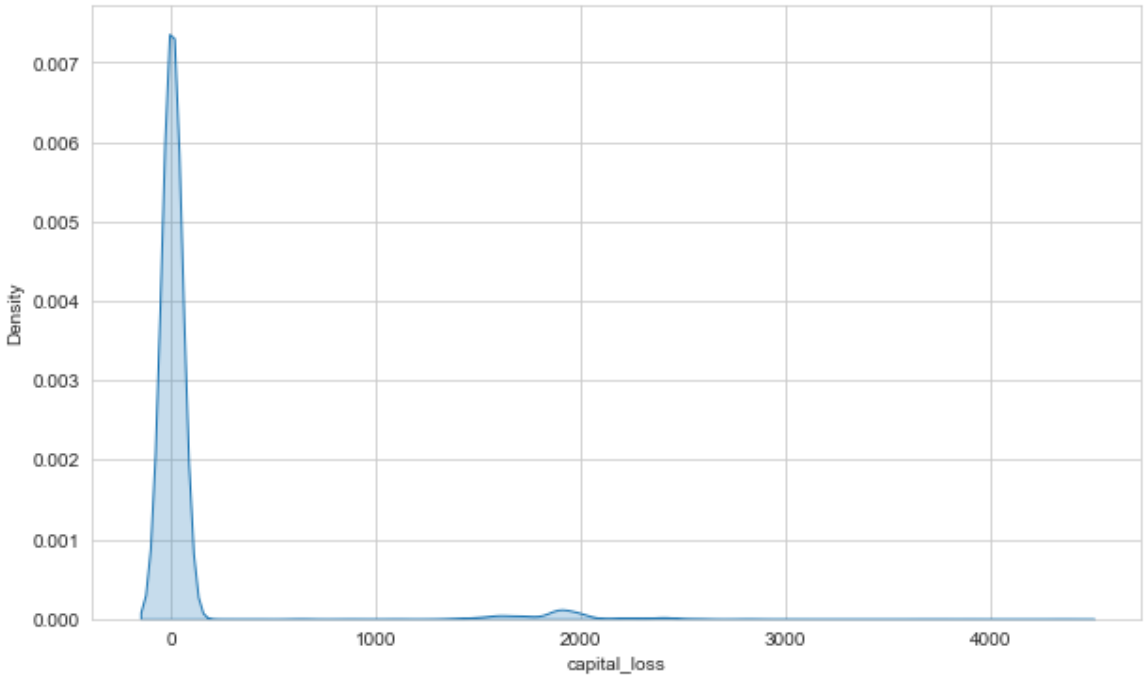
Check the histplot/kdeplot to see distribution of feature

In [1067]:

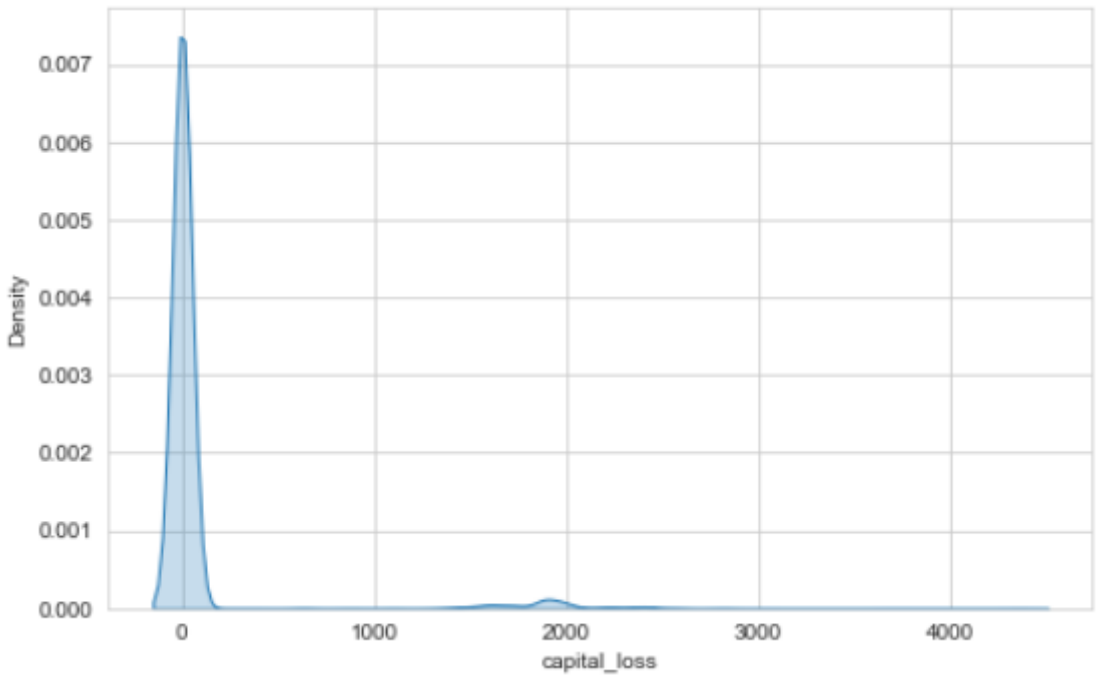
```
# Your Code is Here

sns.kdeplot(x='capital_loss', data=df, fill=True)
```

Out[1067]:

```
<AxesSubplot:xlabel='capital_loss', ylabel='Density'>
```



Desired Output:



Check the statistical values

In [1068]:

```
# Your Code is Here
df.capital_loss.describe()
```

Out[1068]:

```
count   32537.000
mean       87.368
std       403.102
min         0.000
25%         0.000
50%         0.000
75%         0.000
max      4356.000
Name: capital_loss, dtype: float64
```

Desired Output: count 32537.000 mean 87.368 std 403.102 min 0.000 25% 0.000 50% 0.000 75% 0.000 max 4356.000 Name: capital_loss, dtype: float64

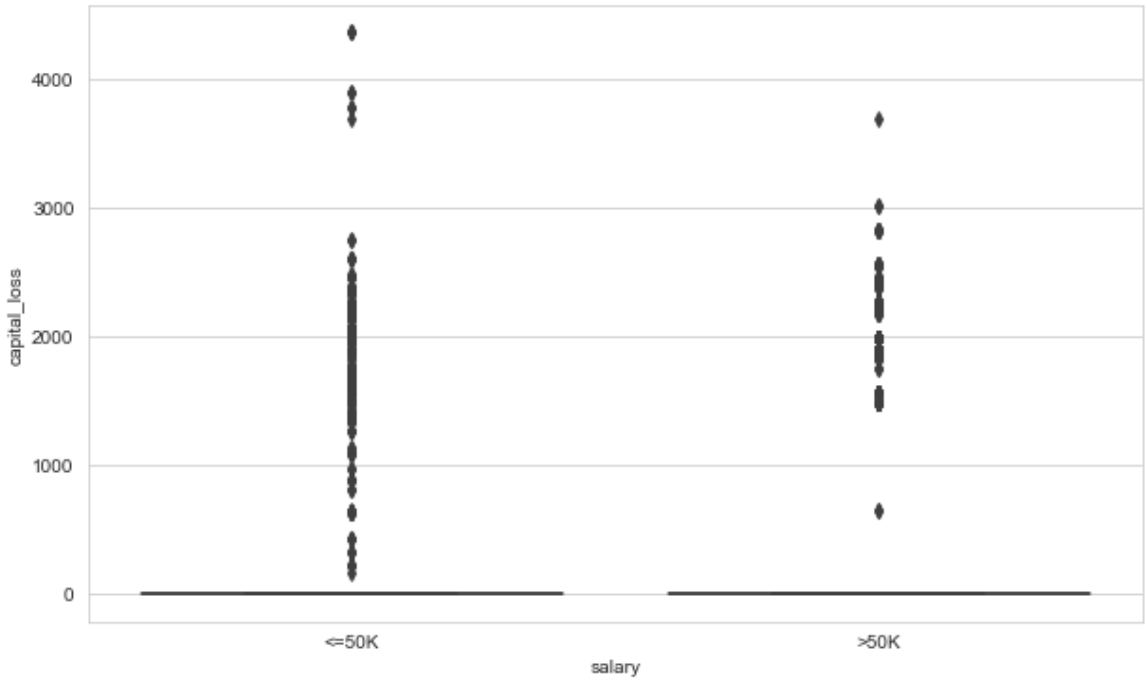
Check the boxplot and histplot/kdeplot by "salary" levels

In [1069]:

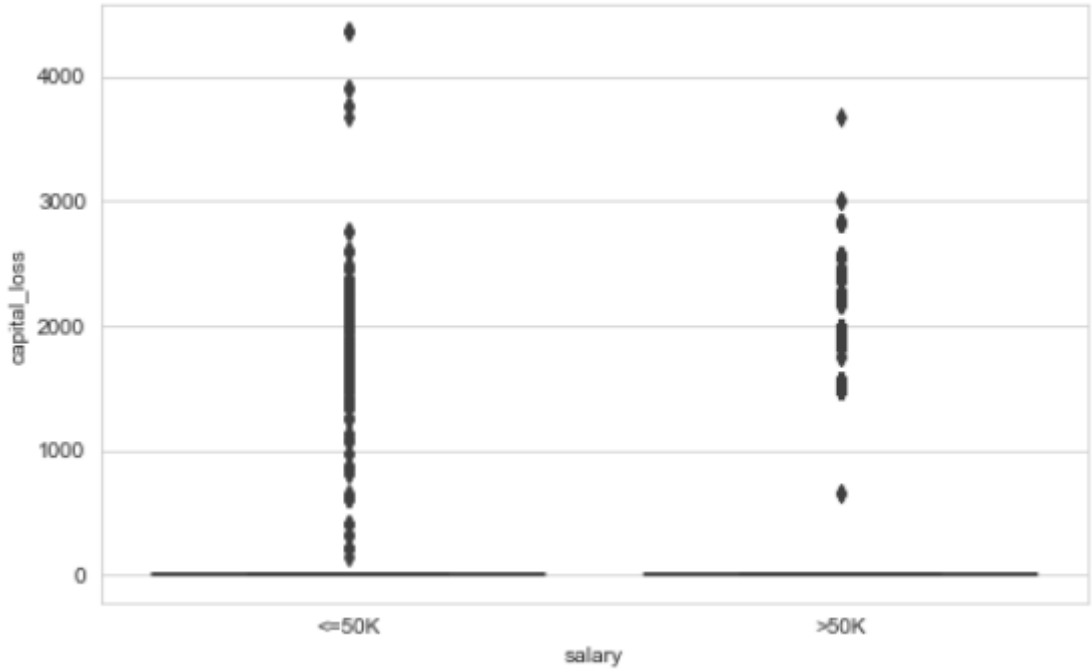
```
# Your Code is Here

sns.boxplot(x='salary',y='capital_loss', data=df)
```

Out[1069]:

```
<AxesSubplot:xlabel='salary', ylabel='capital_loss'>
```



Desired Output:

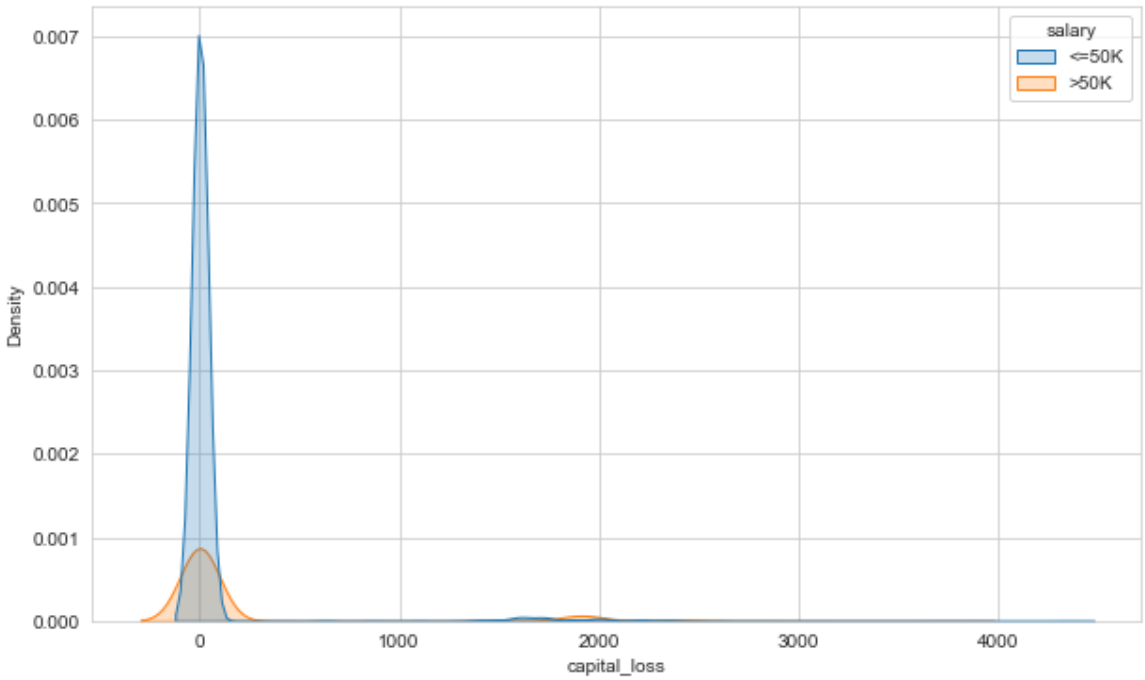


In [1070]:

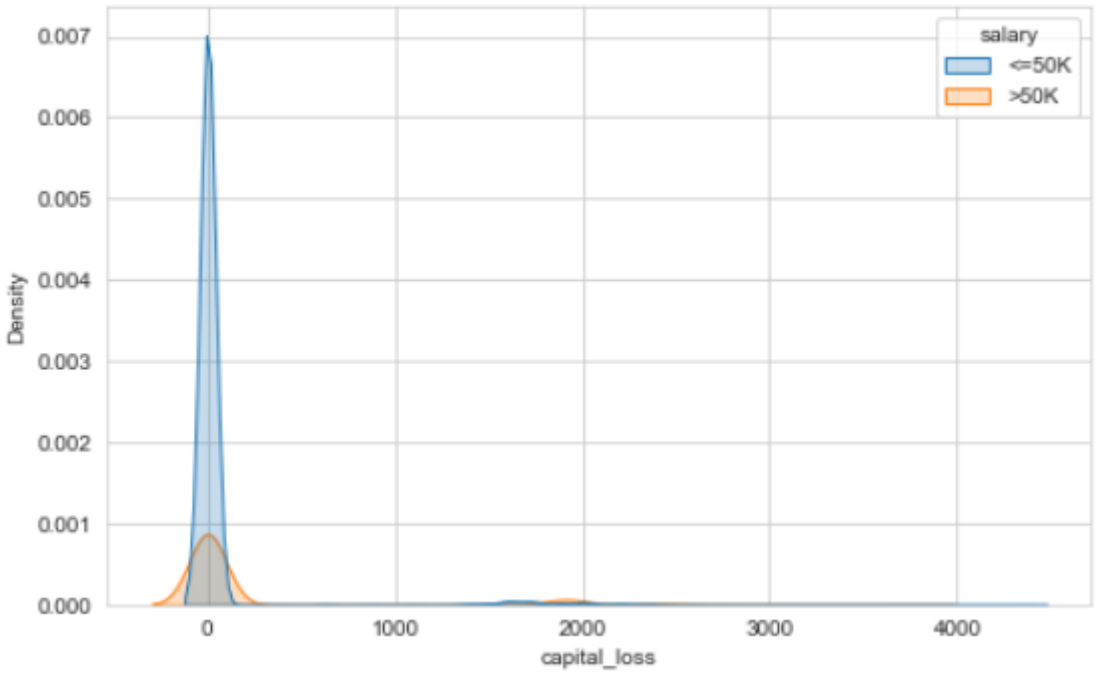
```
# Your Code is Here

sns.kdeplot(x='capital_loss', hue = 'salary', data=df, fill=True)
```

Out[1070]:

```
<AxesSubplot:xlabel='capital_loss', ylabel='Density'>
```



Desired Output:



Check the statistical values by "salary" levels

In [1071]:

```
# Your Code is Here

df.groupby('salary').capital_loss.describe()
```

Out[1071]:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	53.190	310.890	0.000	0.000	0.000	0.000	4356.000
>50K	7839.000	195.051	595.555	0.000	0.000	0.000	0.000	3683.000

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	53.190	310.890	0.000	0.000	0.000	0.000	4356.000
>50K	7839.000	195.051	595.555	0.000	0.000	0.000	0.000	3683.000

Check the statistical values by "salary" levels for capital_loss not equel the zero

In [1072]:

```
# Your Code is Here
df[df.capital_loss != 0].groupby('salary').capital_loss.describe()
```

Out[1072]:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	746.000	1760.983	438.906	155.000	1590.000	1721.000	1980.000	4356.000
>50K	773.000	1978.017	264.144	653.000	1887.000	1902.000	1977.000	3683.000

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	746.000	1760.983	438.906	155.000	1590.000	1721.000	1980.000	4356.000
>50K	773.000	1978.017	264.144	653.000	1887.000	1902.000	1977.000	3683.000

Write down the conclusions you draw from your analysis

Result : The capital_loss variable has outlier values.
The capital_loss histogram is right-skewed.
Its average is 87, 25 percent is 0. 75 percent is zero.
Its minimum is zero while its maximum is 4356.
When viewed at the Salary level, there are outliers in both parts and they are gathered around 0 in general.
The average of below 50K capital_loss is 53, 25 percent is 0. 75 percent is zero.
The average of over 50K capital_loss is 195. 25 percent is 0. 75 percent is zero.
The average of under 50K earners with 0 capital_loss is 1760, 50 percent of them is between 1590 and 1980.
The average of above 50K earners with 0 capital_loss is 1978, 50 percent of them is between 1887 and 1977.

hours_per_week

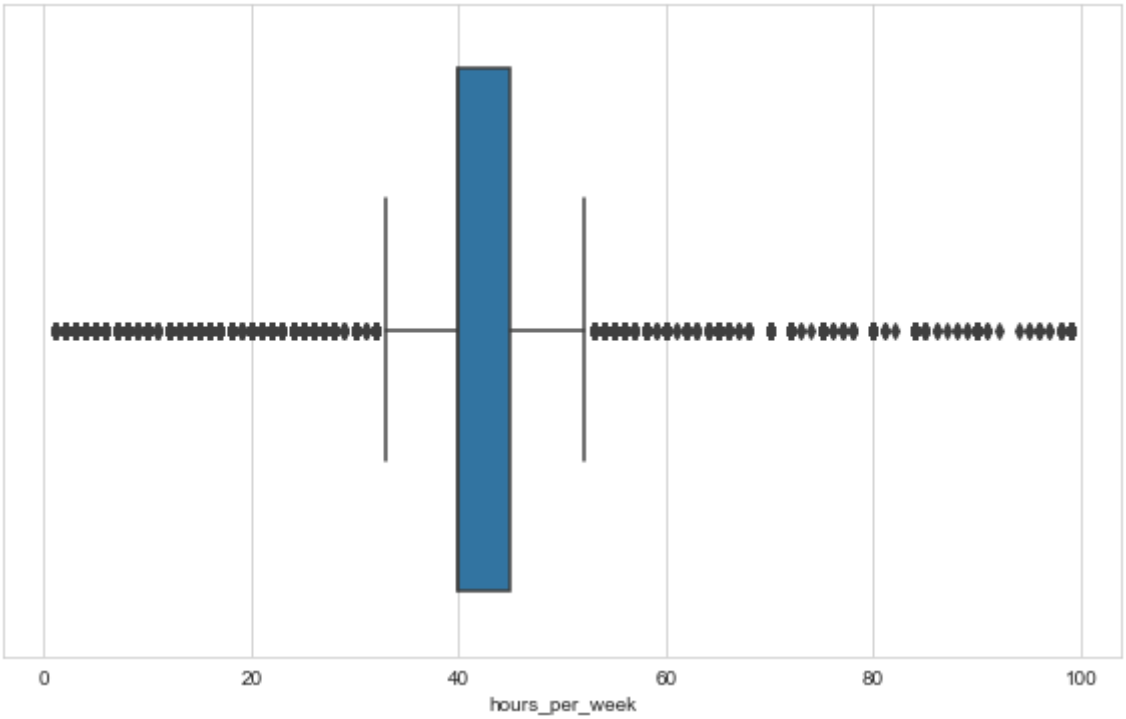
Check the boxplot to see extreme values

In [1073]:

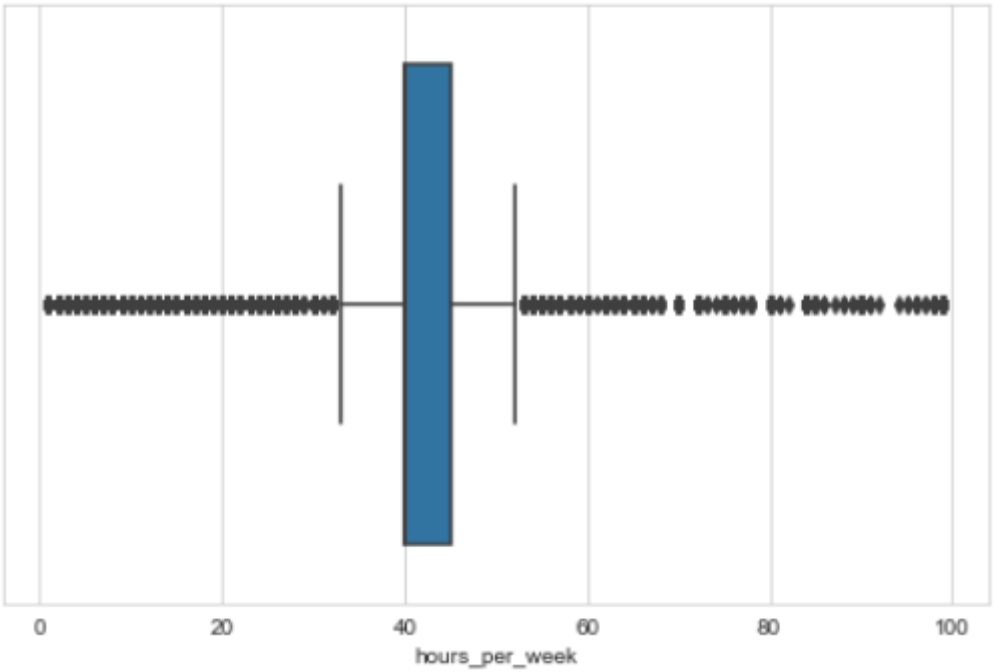
```
# Your Code is Here

sns.boxplot(x='hours_per_week',data=df)
```

Out[1073]:

<AxesSubplot:xlabel='hours_per_week'>



Desired Output:



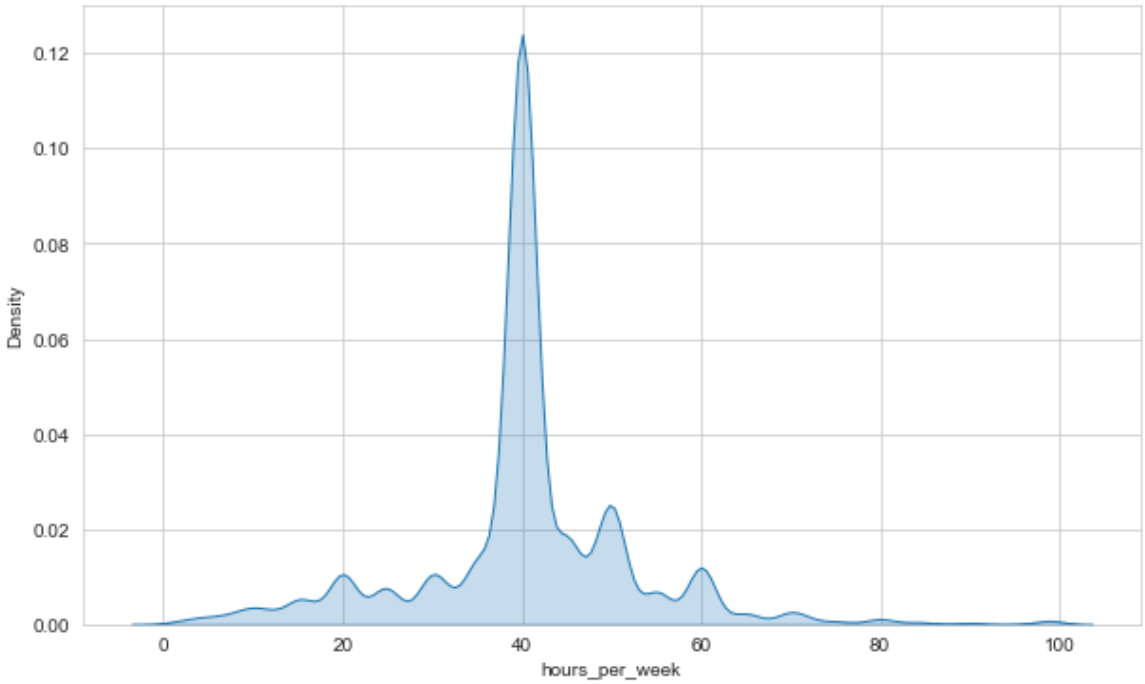
Check the histplot/kdeplot to see distribution of feature

In [1074]:

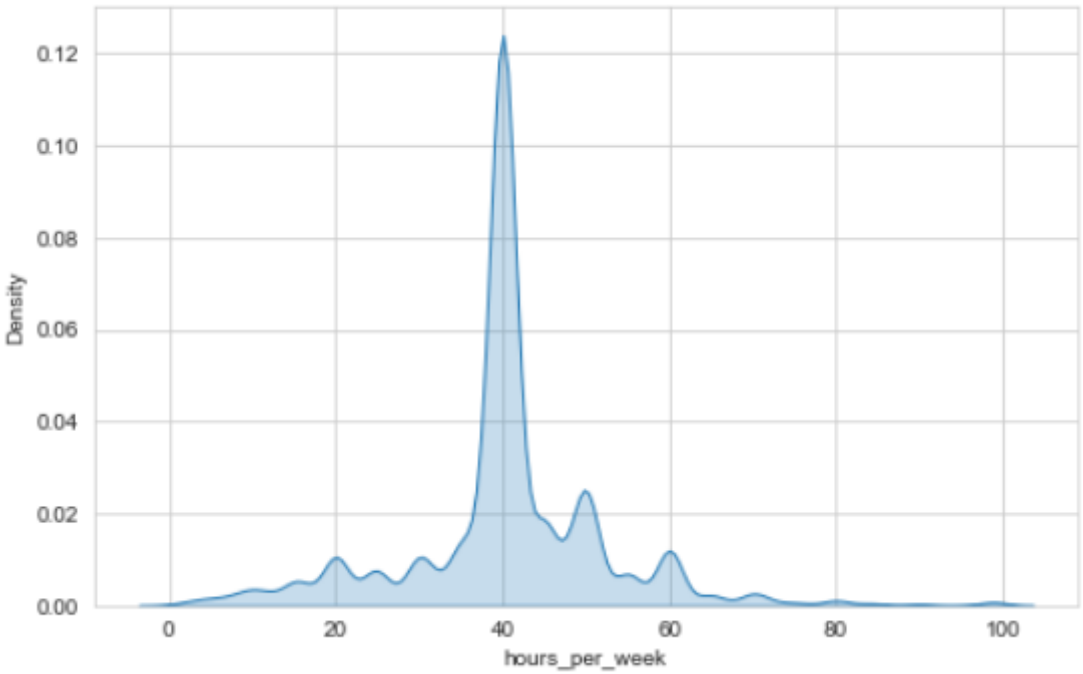
```
# Your Code is Here
sns.kdeplot(x='hours_per_week',data=df, fill=True)
```

Out[1074]:

```
<AxesSubplot:xlabel='hours_per_week', ylabel='Density'>
```



Desired Output:



Check the statistical values

In [1075]:

```
# Your Code is Here

df.hours_per_week.describe()
```

Out[1075]:

```
count   32537.000
mean       40.440
std        12.347
min         1.000
25%        40.000
50%        40.000
75%        45.000
max        99.000
Name: hours_per_week, dtype: float64
```

Desired Output: count 32537.000 mean 40.440 std 12.347 min 1.000 25% 40.000 50% 40.000 75% 45.000 max 99.000 Name: hours_per_week, dtype: float64

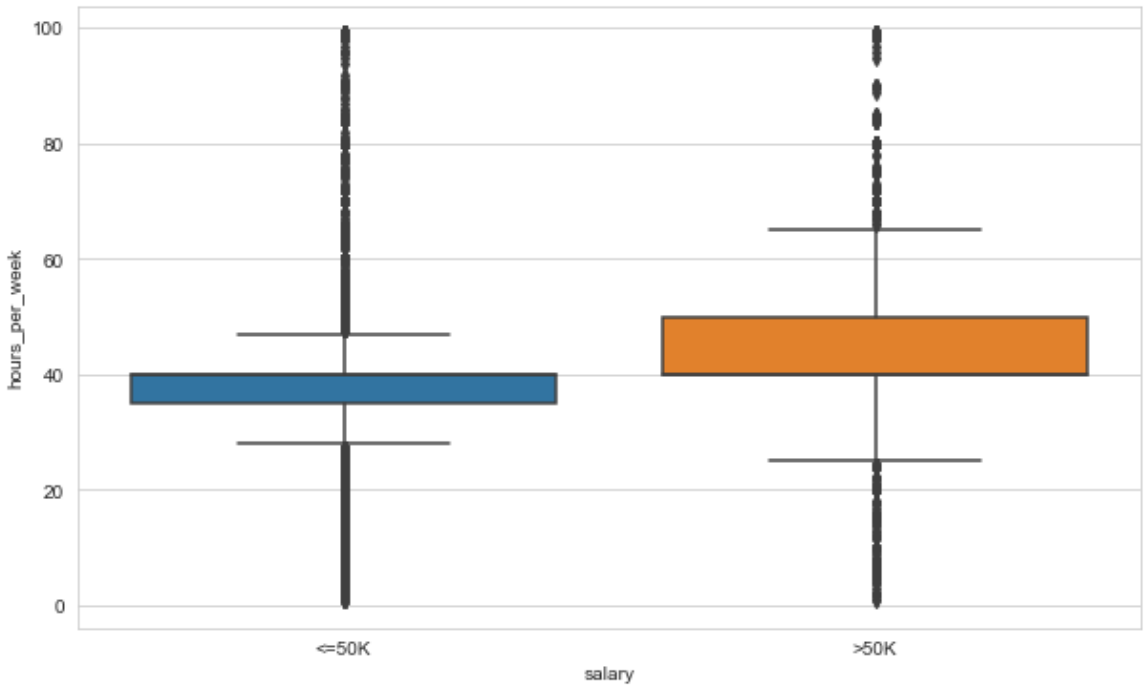
Check the boxplot and histplot/kdeplot by "salary" levels

In [1076]:

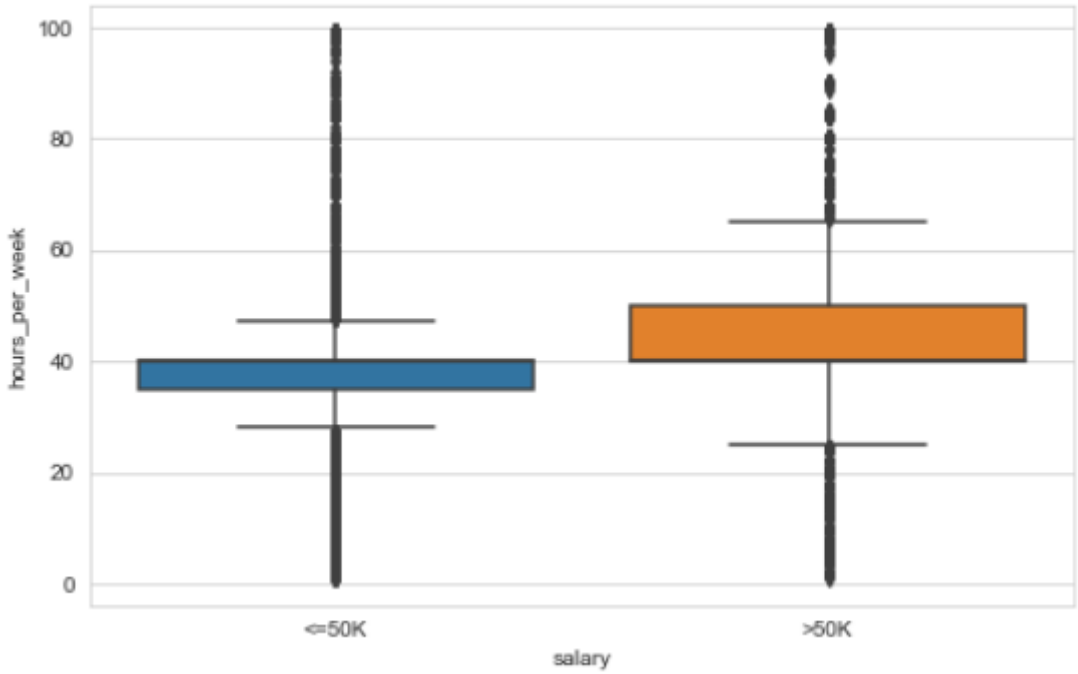
```
# Your Code is Here

sns.boxplot(x='salary', y='hours_per_week', data=df)
```

Out[1076]:

```
<AxesSubplot:xlabel='salary', ylabel='hours_per_week'>
```



Desired Output:

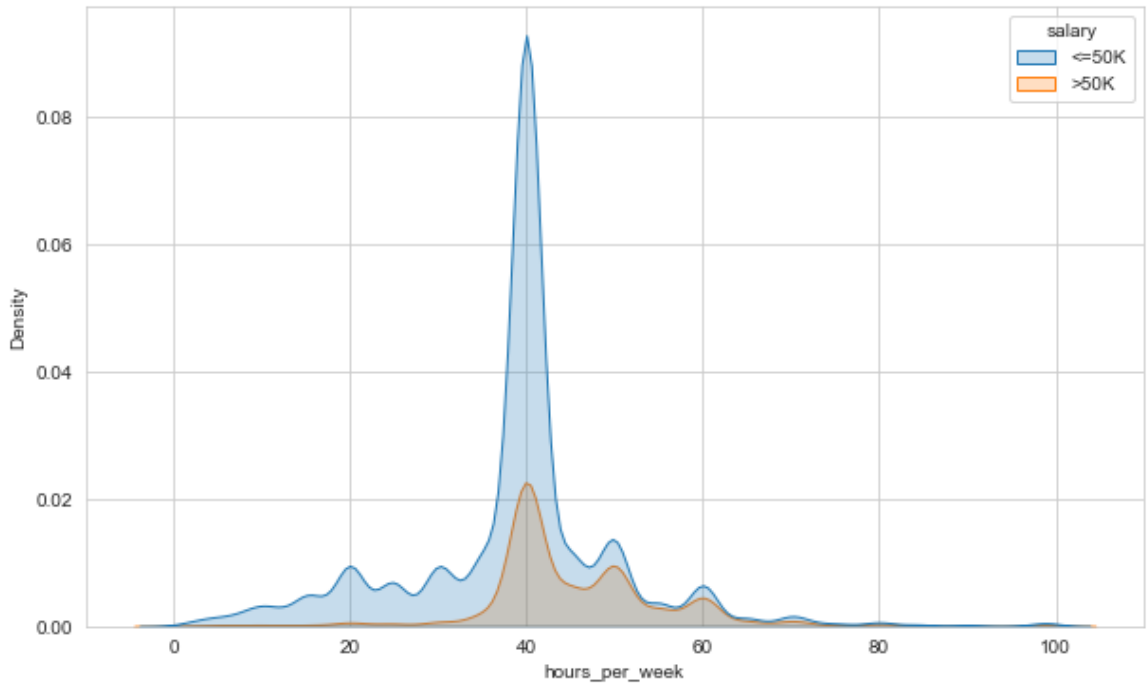


In [1077]:

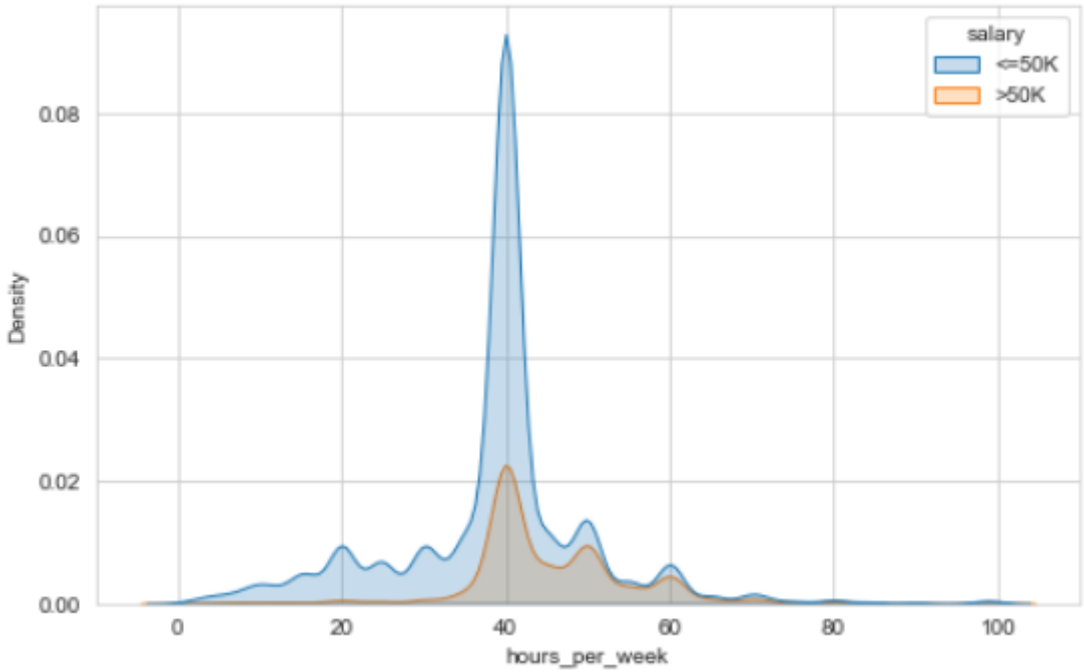
```
# Your Code is Here

sns.kdeplot(x='hours_per_week', hue='salary',data=df,fill=True)
```

Out[1077]:

```
<AxesSubplot:xlabel='hours_per_week', ylabel='Density'>
```



Desired Output:



Check the statistical values by "salary" levels

In [1078]:

```
# Your Code is Here

df.groupby('salary').hours_per_week.describe()
```

Out[1078]:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	38.843	12.318	1.000	35.000	40.000	40.000	99.000
>50K	7839.000	45.473	11.014	1.000	40.000	40.000	50.000	99.000

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	24698.000	38.843	12.318	1.000	35.000	40.000	40.000	99.000
>50K	7839.000	45.473	11.014	1.000	40.000	40.000	50.000	99.000

Write down the conclusions you draw from your analysis

Result : The hours_per_week variable has outlier values.
The hours_per_week histogram is close to the normal distribution. Its mean and median value is 40.
Its average is 40, 25 percent is 40. 75 percent is 45.
Its minimum is 1 while its maximum is 99.
At the salary level, there are outliers in both parts.
The average of below 50K is 38, 50 percent is between 35 and 40.
The average of over 50K is 45. The 50th percentile is between 40 and 50.
Those who earn over 50K generally work more than those who earn under 50K.

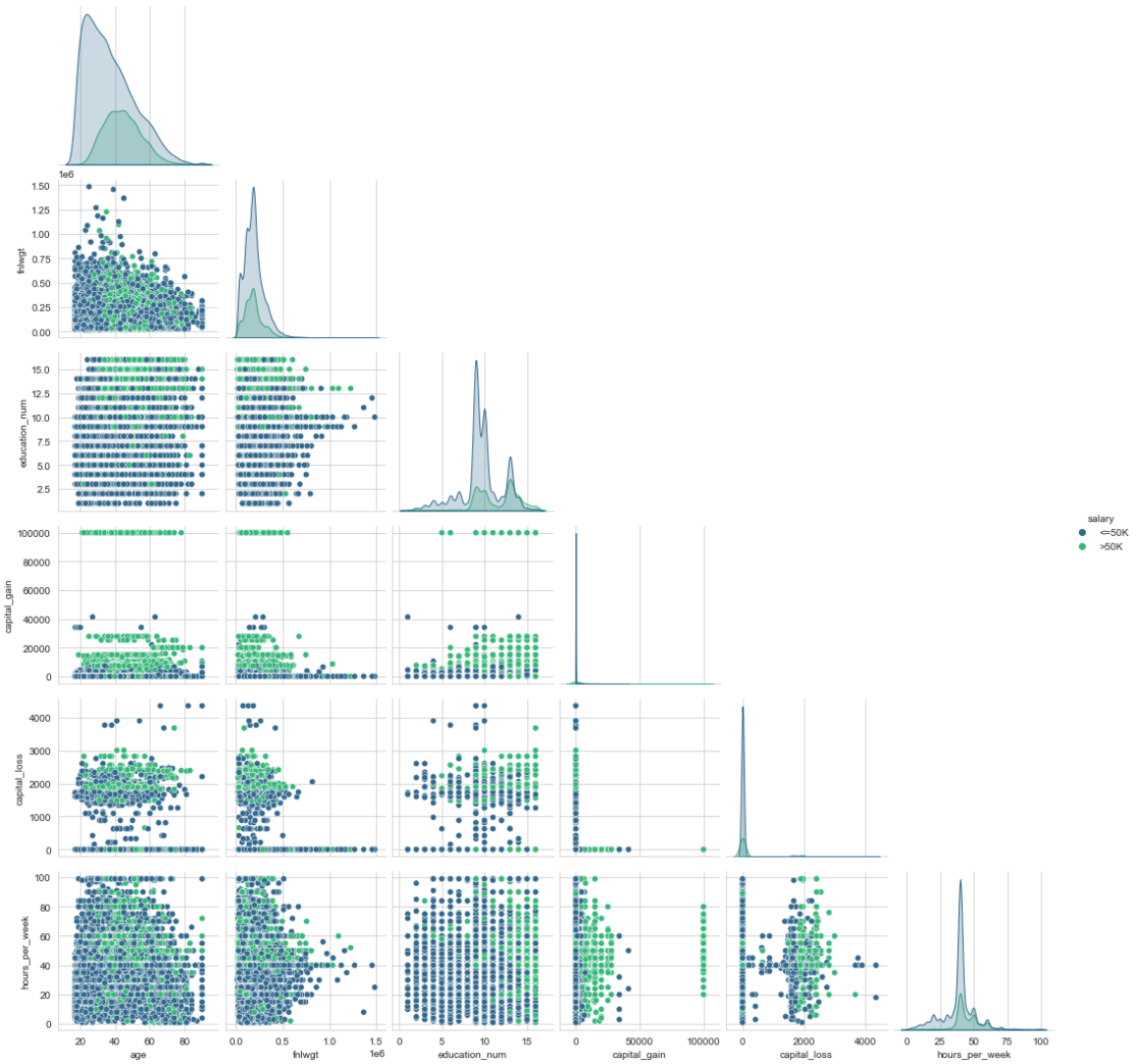
See the relationship between each numeric features by target feature (salary) in one plot basically

In [1079]:

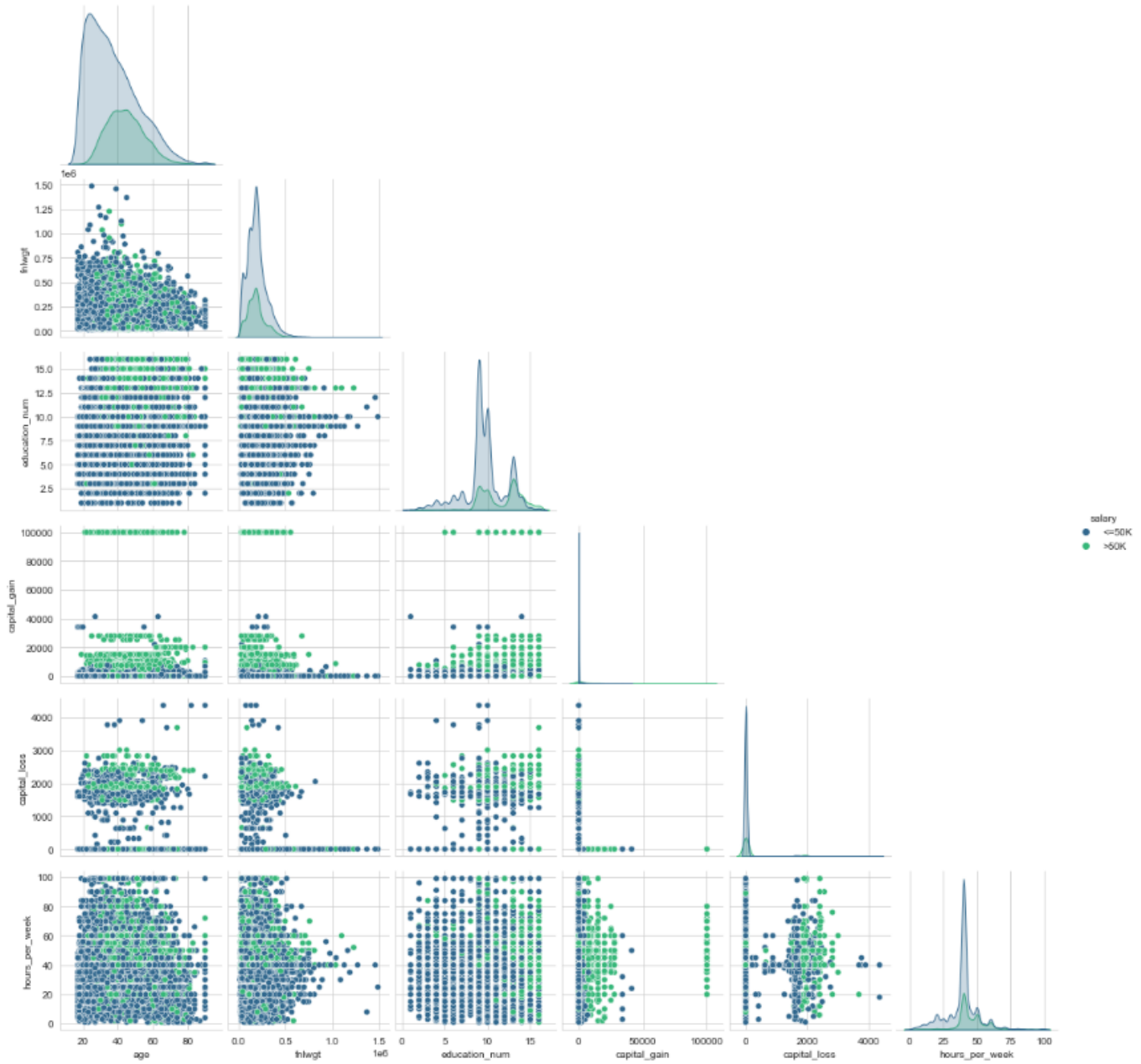
```
# Your Code is Here
sns.pairplot(data = df, hue='salary', palette='viridis', corner= True)
```

Out[1079]:

```
<seaborn.axisgrid.PairGrid at 0x7fd3a9c3eb20>
```



Desired Output:



Categorical Features

education & education_num

Detect the similarities between these features by comparing unique values

In [1080]:

```
# Your Code is Here

df.education.value_counts(dropna=False)
```

Out[1080]:

```
HS-grad         10494
Some-college     7282
Bachelors        5353
Masters          1722
Assoc-voc        1382
11th             1175
Assoc-acdm       1067
10th              933
7th-8th           645
Prof-school       576
9th               514
12th              433
Doctorate         413
5th-6th           332
1st-4th           166
Preschool          50
Name: education, dtype: int64
```

Desired Output: HS-grad 10494 Some-college 7282 Bachelors 5353 Masters 1722 Assoc-voc 1382 11th 1175 Assoc-acdm 1067 10th 933 7th-8th 645 Prof-school 576 9th 514 12th 433 Doctorate 413 5th-6th 332 1st-4th 166 Preschool 50 Name: education, dtype: int64

In [1081]:

```
# Your Code is Here
df.education_num.value_counts(dropna=False)
```

Out[1081]:

```
9.000     10208
10.000     7089
13.000     5245
14.000     1686
11.000     1343
7.000      1146
12.000     1044
6.000       916
NaN         802
4.000       630
15.000      559
5.000       503
8.000       424
16.000      405
3.000       329
2.000       159
1.000        49
Name: education_num, dtype: int64
```

Desired Output: 9.000 10208 10.000 7089 13.000 5245 14.000 1686 11.000 1343 7.000 1146 12.000 1044 6.000 916 NaN 802 4.000 630 15.000 559 5.000 503 8.000 424 16.000 405 3.000 329 2.000 159 1.000 49 Name:

education_num, dtype: int64

In [1082]:

```
# Your Code is Here
df.groupby('education').education_num.value_counts(dropna=False)
```

Out[1082]:

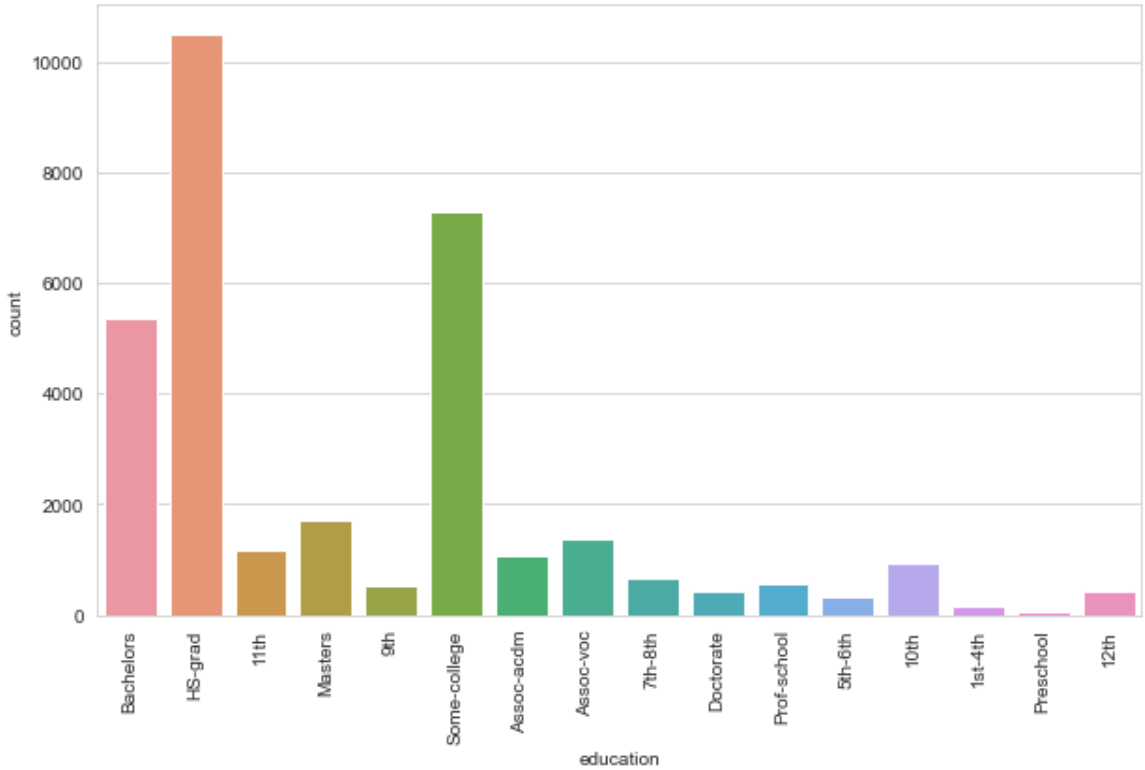
```
education     education_num
10th          6.000              916
              NaN                 17
11th          7.000             1146
              NaN                 29
12th          8.000              424
              NaN                   9
1st-4th       2.000              159
              NaN                   7
5th-6th       3.000              329
              NaN                   3
7th-8th       4.000              630
              NaN                 15
9th           5.000              503
              NaN                 11
Assoc-acdm    12.000            1044
              NaN                 23
Assoc-voc     11.000            1343
              NaN                 39
Bachelors     13.000            5245
              NaN                108
Doctorate     16.000             405
              NaN                   8
HS-grad       9.000            10208
              NaN                286
Masters       14.000            1686
              NaN                 36
Preschool     1.000                49
              NaN                   1
Prof-school   15.000             559
              NaN                 17
Some-college  10.000            7089
              NaN                193
Name: education_num, dtype: int64
```

Desired Output: education education_num 10th 6.000 916 NaN 17 11th 7.000 1146 NaN 29 12th 8.000 424 NaN 9 1st-4th 2.000 159 NaN 7 5th-6th 3.000 329 NaN 3 7th-8th 4.000 630 NaN 15 9th 5.000 503 NaN 11 Assoc-acdm 12.000 1044 NaN 23 Assoc-voc 11.000 1343 NaN 39 Bachelors 13.000 5245 NaN 108 Doctorate 16.000 405 NaN 8 HS-grad 9.000 10208 NaN 286 Masters 14.000 1686 NaN 36 Preschool 1.000 49 NaN 1 Prof-school 15.000 559 NaN 17 Some-college 10.000 7089 NaN 193 Name: education_num, dtype: int64

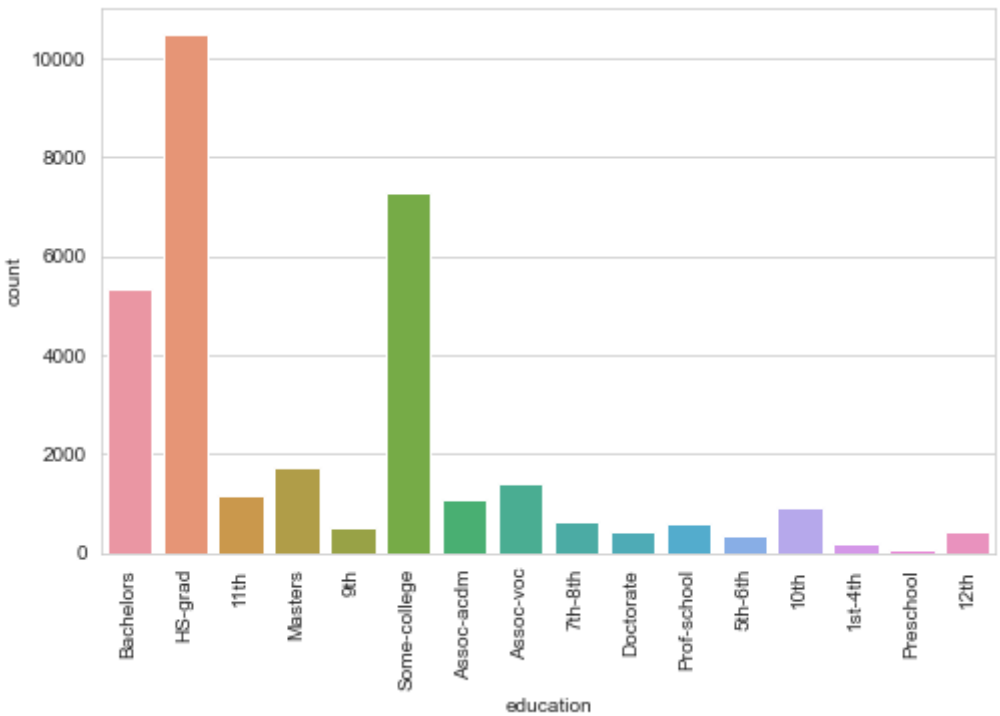
Visualize the count of person in each categories for these features (education, education_num) separately

In [1083]:

```
# Your Code is Here
sns.countplot(x='education', data=df)
plt.xticks(rotation = 90);
```



Desired Output:

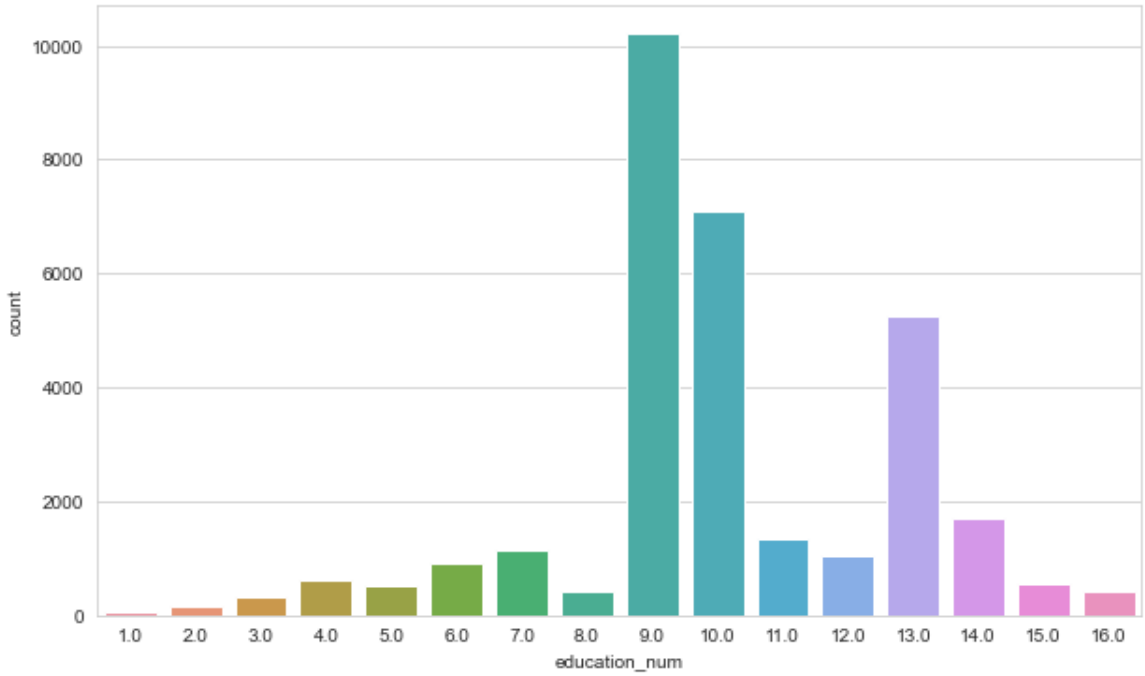


In [1084]:

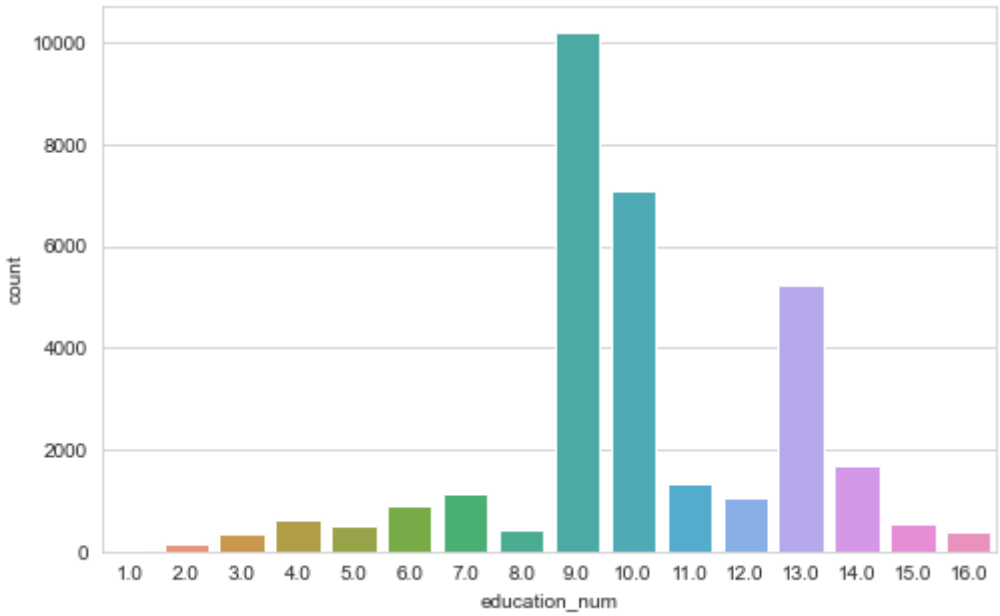
```
# Your Code is Here
sns.countplot(x='education_num', data=df)
```

Out[1084]:

```
<AxesSubplot:xlabel='education_num', ylabel='count'>
```



Desired Output:



Check the count of person in each "salary" levels by these features (education and education_num) separately and visualize them with countplot

In [1085]:

```
# Your Code is Here
df.groupby('education').salary.value_counts(dropna=False)
```

Out[1085]:

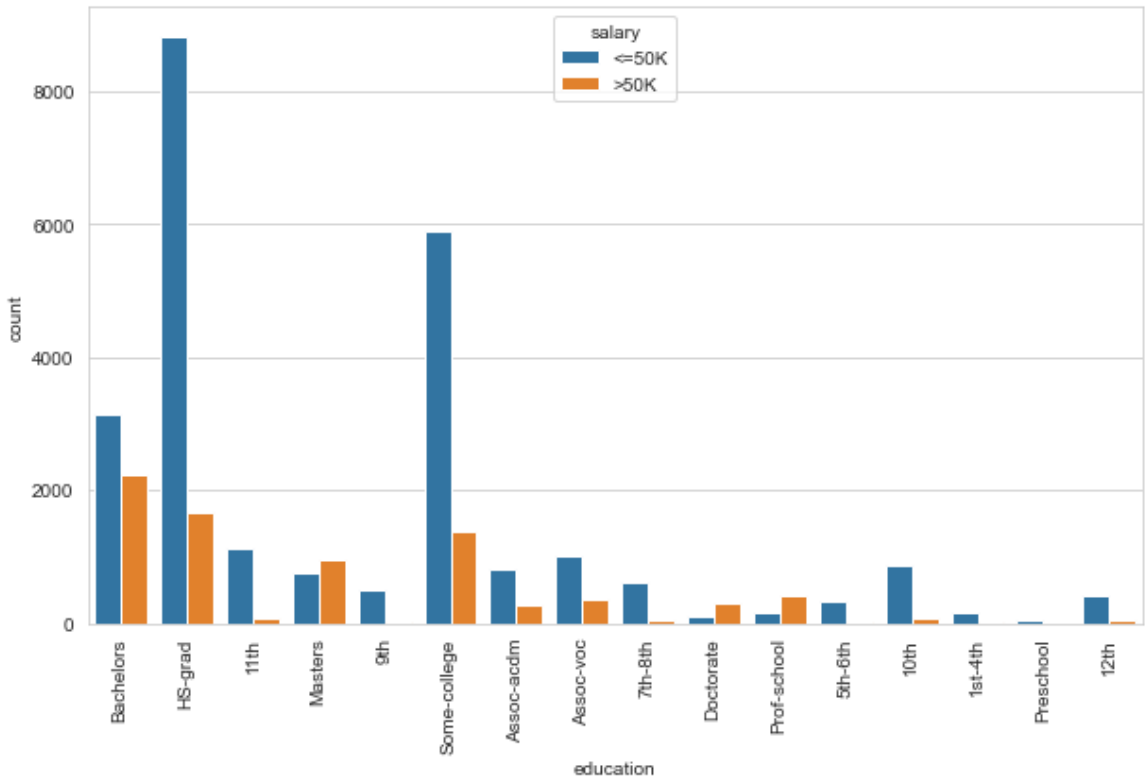
```
education     salary
10th          <=50K      871
              >50K        62
11th          <=50K     1115
              >50K        60
12th          <=50K      400
              >50K        33
1st-4th       <=50K      160
              >50K         6
5th-6th       <=50K      316
              >50K        16
7th-8th       <=50K      605
              >50K        40
9th           <=50K      487
              >50K        27
Assoc-acdm    <=50K      802
              >50K       265
Assoc-voc     <=50K     1021
              >50K       361
Bachelors     <=50K     3132
              >50K      2221
Doctorate     >50K       306
              <=50K      107
HS-grad       <=50K     8820
              >50K      1674
Masters       >50K       959
              <=50K      763
Preschool     <=50K       50
Prof-school   >50K       423
              <=50K      153
Some-college  <=50K     5896
              >50K      1386
Name: salary, dtype: int64
```

Desired Output: education salary 10th <=50K 871 >50K 62 11th <=50K 1115 >50K 60 12th <=50K 400 >50K 33 1st-4th <=50K 160 >50K 6 5th-6th <=50K 316 >50K 16 7th-8th <=50K 605 >50K 40 9th <=50K 487 >50K 27 Assoc-acdm <=50K 802 >50K 265 Assoc-voc <=50K 1021 >50K 361 Bachelors <=50K 3132 >50K 2221 Doctorate >50K 306 <=50K 107 HS-grad <=50K 8820 >50K 1674 Masters >50K 959 <=50K 763 Preschool

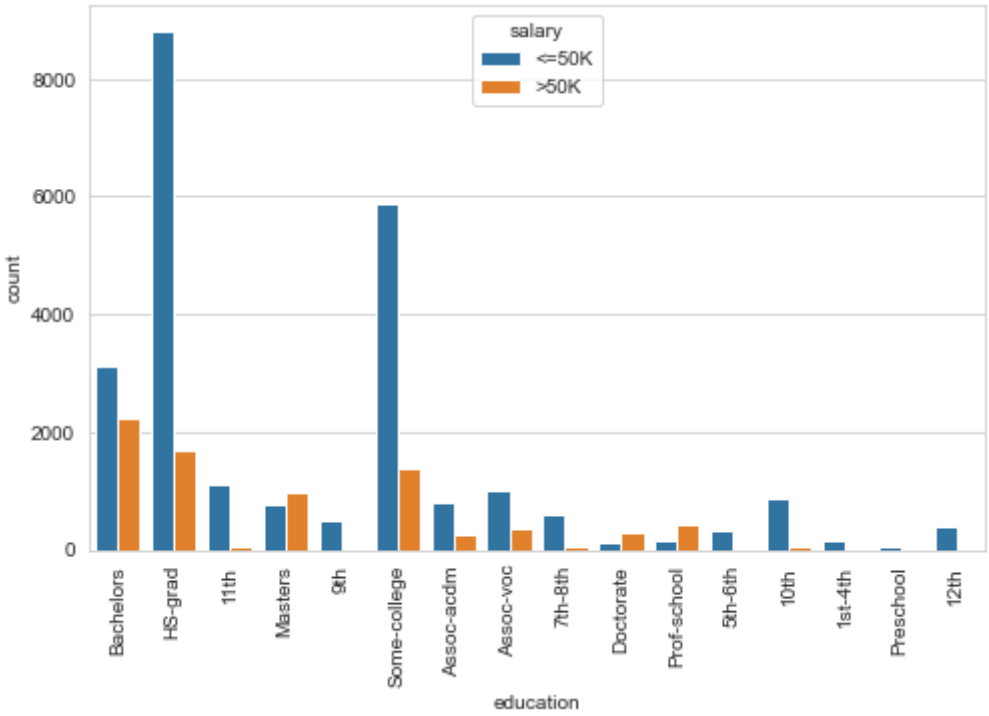
<=50K 50 Prof-school >50K 423 <=50K 153 Some-college <=50K 5896 >50K 1386 Name: salary, dtype: int64

In [1086]:

```
# Your Code is Here
sns.countplot(x='education', hue='salary', data=df)
plt.xticks(rotation = 90);
```



Desired Output:



In [1087]:

```
# Your Code is Here
df.groupby('education_num').salary.value_counts(dropna=False)
```

Out[1087]:

```
education_num  salary
1.000          <=50K       49
2.000          <=50K      153
               >50K         6
3.000          <=50K      313
               >50K        16
4.000          <=50K      592
               >50K        38
5.000          <=50K      477
               >50K        26
6.000          <=50K      854
               >50K        62
7.000          <=50K     1088
               >50K        58
8.000          <=50K      391
               >50K        33
9.000          <=50K     8579
               >50K      1629
10.000         <=50K     5746
               >50K      1343
11.000         <=50K      994
               >50K       349
12.000         <=50K      787
               >50K       257
13.000         <=50K     3078
               >50K      2167
14.000         >50K       935
               <=50K      751
15.000         >50K       410
               <=50K      149
16.000         >50K       302
               <=50K      103
Name: salary, dtype: int64
```

Desired Output: education_num salary 1.000 <=50K 49 2.000 <=50K 153 >50K 6 3.000 <=50K 313 >50K 16 4.000 <=50K 592 >50K 38 5.000 <=50K 477 >50K 26 6.000 <=50K 854 >50K 62 7.000 <=50K 1088 >50K 58 8.000 <=50K 391 >50K 33 9.000 <=50K 8579 >50K 1629 10.000 <=50K 5746 >50K 1343 11.000 <=50K 994 >50K 349 12.000 <=50K 787 >50K 257 13.000 <=50K 3078 >50K 2167 14.000 >50K 935 <=50K 751 15.000

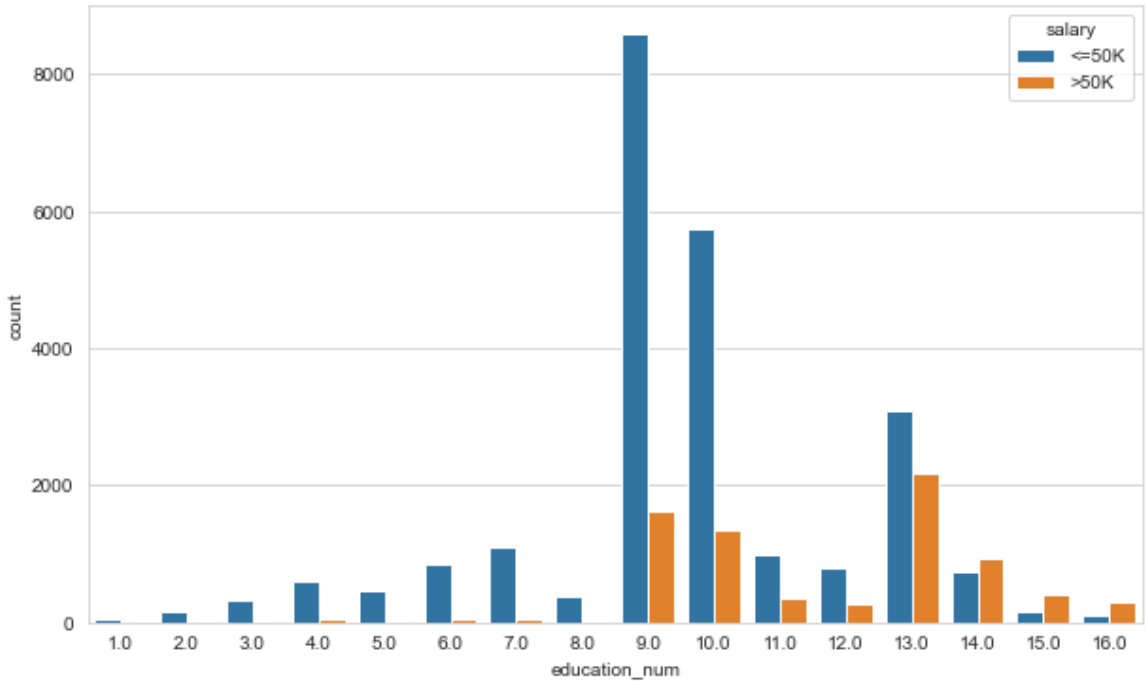
>50K 410 <=50K 149 16.000 >50K 302 <=50K 103 Name: salary, dtype: int64

In [1088]:

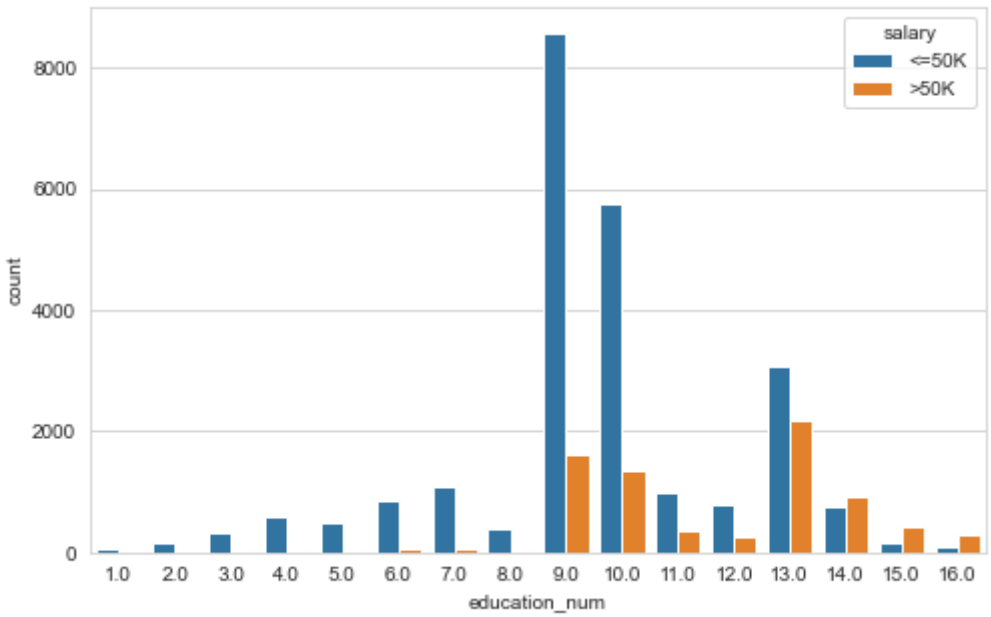
```
# Your Code is Here
sns.countplot(x='education_num',hue='salary',data=df)
```

Out[1088]:

```
<AxesSubplot:xlabel='education_num', ylabel='count'>
```



Desired Output:



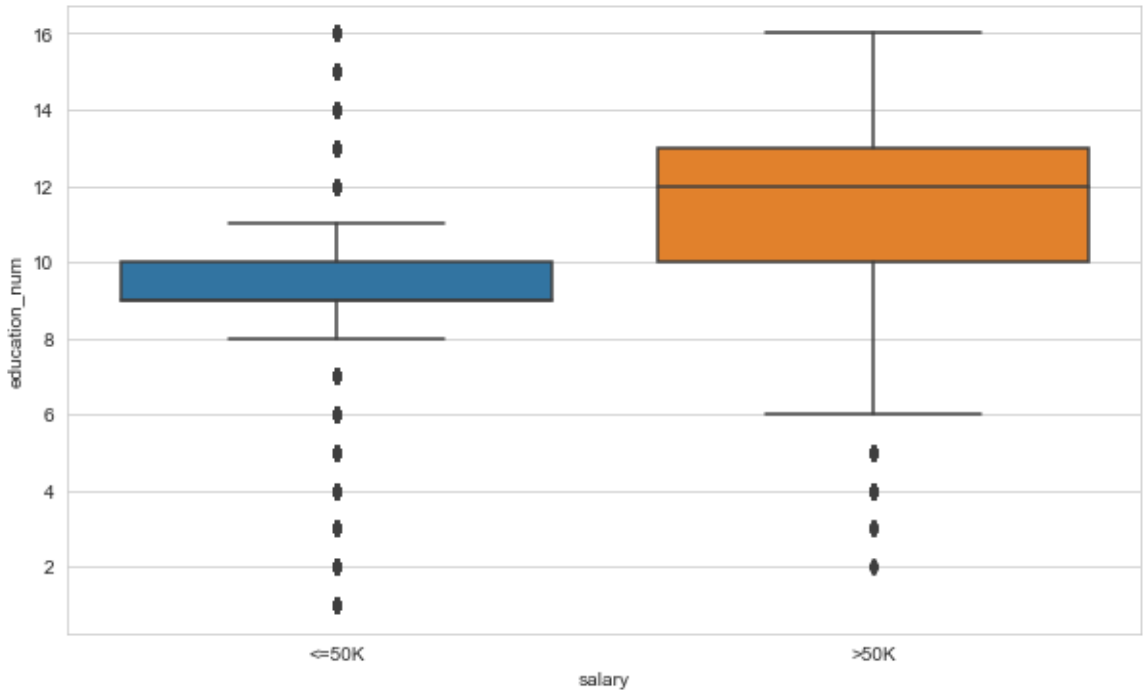
Visualize the boxplot of "education_num" feature by "salary" levels

In [1089]:

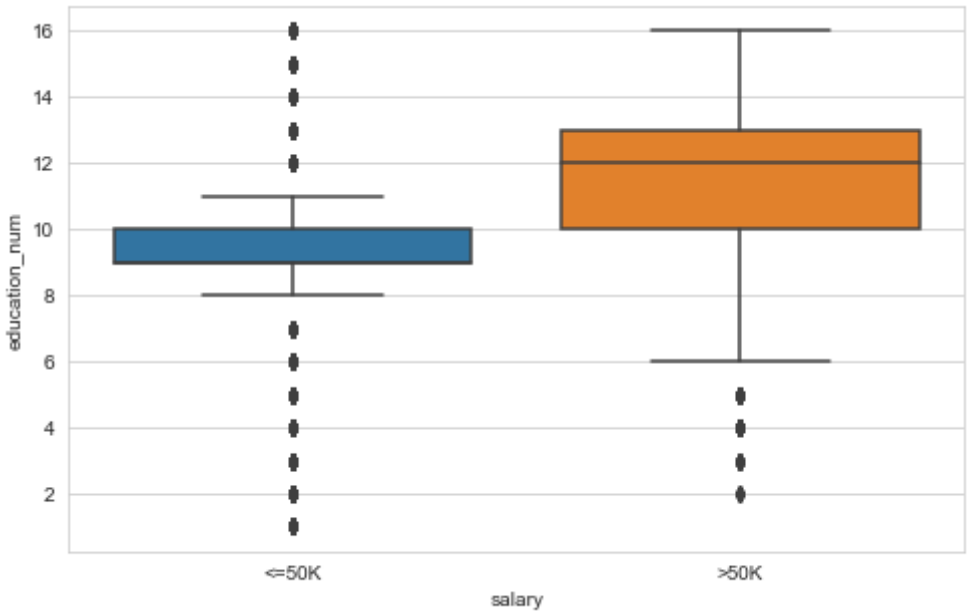
```
# Your Code is Here

sns.boxplot(x='salary',y='education_num', data=df)
```

Out[1089]:

<AxesSubplot:xlabel='salary', ylabel='education_num'>



Desired Output:



Decrease the number of categories in "education" feature as low, medium, and high level and create a new feature with this new categorical data.

In [1090]:

```
def mapping_education(x):
    if x in ["Preschool", "1st-4th", "5th-6th", "7th-8th", "9th", "10th", "11t
h", "12th"]:
        return "low_level_grade"
    elif x in ["HS-grad", "Some-college", "Assoc-voc", "Assoc-acdm"]:
        return "medium_level_grade"
    elif x in ["Bachelors", "Masters", "Prof-school", "Doctorate"]:
        return "high_level_grade"
```

In [1091]:

```
# Your Code is Here
df.education.apply(lambda x: mapping_education(x)).value_counts(dropna=False)
```

Out[1091]:

```
medium_level_grade    20225
high_level_grade       8064
low_level_grade        4248
Name: education, dtype: int64
```

Desired Output: medium_level_grade 20225 high_level_grade 8064 low_level_grade 4248 Name: education, dtype: int64

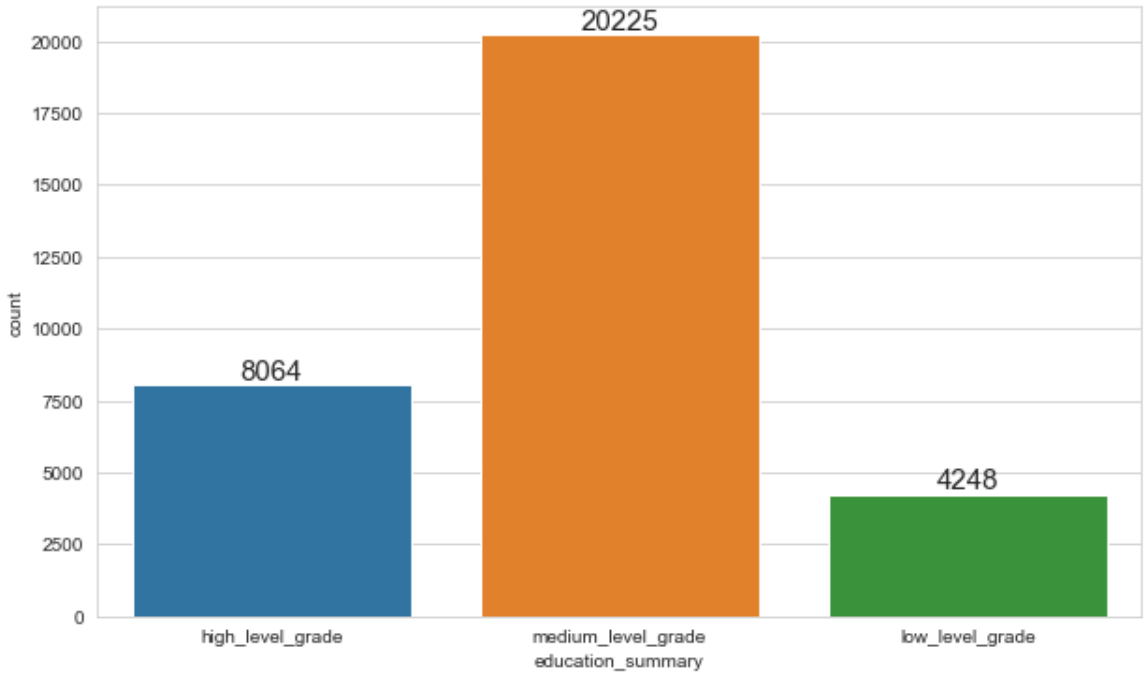
In [1092]:

```
# By using "mapping_education" def function above, create a new column named "ed
ucation_summary"

# Your Code is Here
df['education_summary'] = df.education.apply(lambda x: mapping_education(x))
```

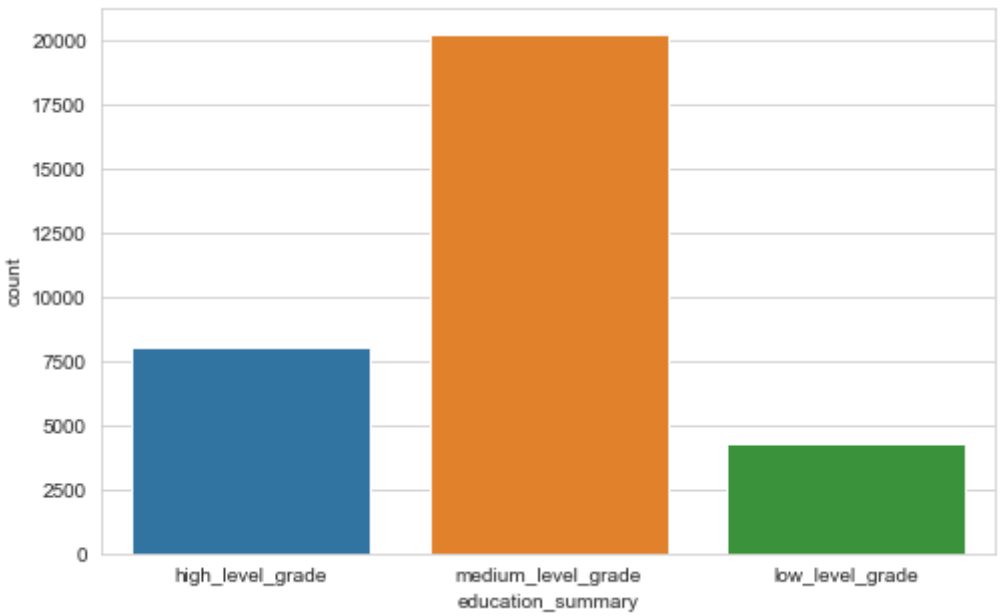
Visualize the count of person in each categories for these new education levels (high, medium, low)

In [1093]:

```
# Your Code is Here

g = sns.countplot(x='education_summary', data=df)
plt.bar_label(g.containers[0], fontsize= 15);
```



Desired Output:



Check the count of person in each "salary" levels by these new education levels(high, medium, low) and visualize it with countplot

In [1094]:

```
# Your Code is Here

df.groupby('education_summary').salary.value_counts()
```

Out[1094]:

```
education_summary   salary
high_level_grade    <=50K      4155
                    >50K       3909
low_level_grade     <=50K      4004
                    >50K        244
medium_level_grade  <=50K     16539
                    >50K       3686
Name: salary, dtype: int64
```

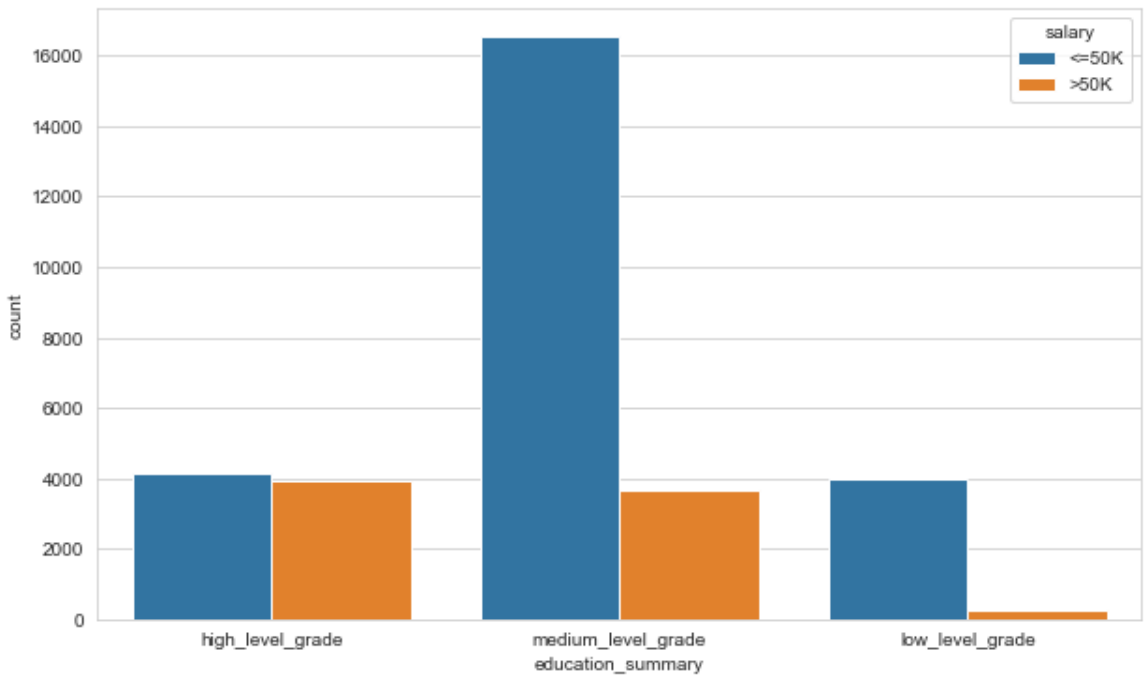
Desired Output: education_summary salary high_level_grade <=50K 4155 >50K 3909 low_level_grade <=50K 4004 >50K 244 medium_level_grade <=50K 16539 >50K 3686 Name: salary, dtype: int64

In [1095]:

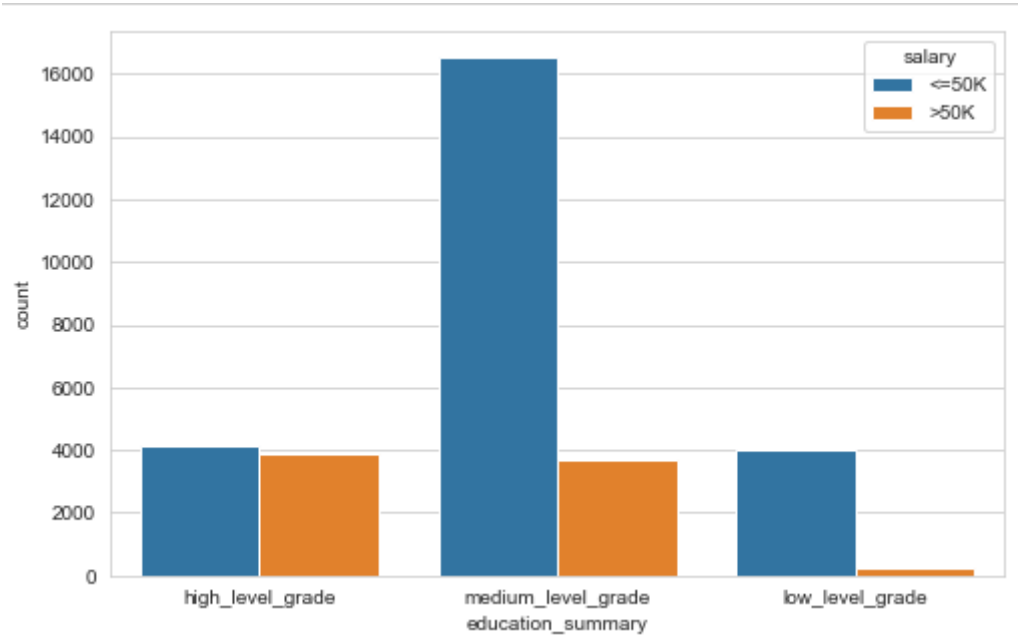
```
# Your Code is Here

sns.countplot(x='education_summary', hue='salary', data=df)
```

Out[1095]:

```
<AxesSubplot:xlabel='education_summary', ylabel='count'>
```



Desired Output:



Check the percentage distribution of person in each "salary" levels by each new education levels (high, medium, low) and visualize it with pie plot separately

In [1096]:

```
# Your Code is Here
df.groupby('education_summary').salary.value_counts(dropna=False) / df.groupby
('education_summary').salary.count()
```

Out[1096]:

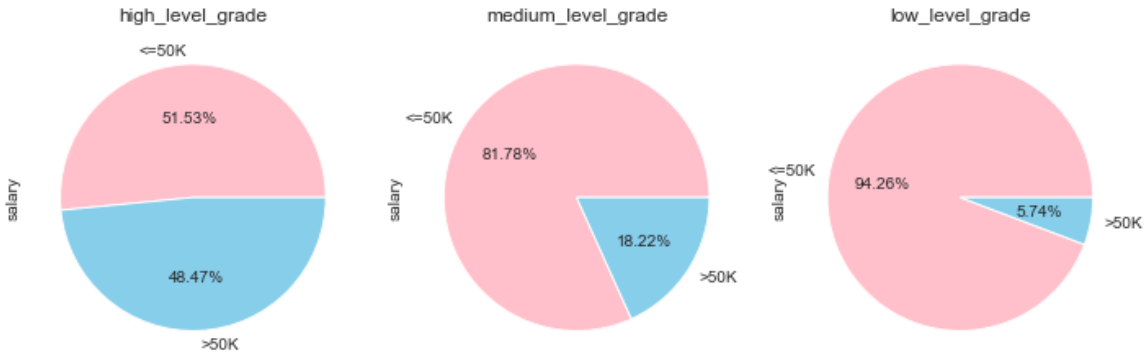
```
education_summary   salary
high_level_grade    <=50K    0.515
                    >50K     0.485
low_level_grade     <=50K    0.943
                    >50K     0.057
medium_level_grade  <=50K    0.818
                    >50K     0.182
Name: salary, dtype: float64
```

Desired Output: education_summary salary high_level_grade <=50K 0.515 >50K 0.485 low_level_grade <=50K

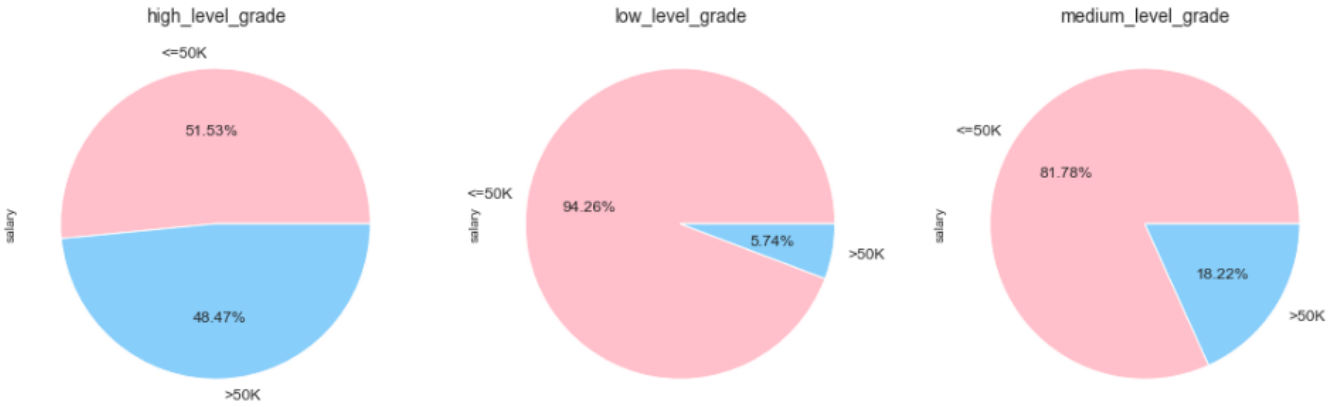
0.943 >50K 0.057 medium_level_grade <=50K 0.818 >50K 0.182 Name: salary, dtype: float64

In [1097]:

```
# Your Code is Here

for count,x in enumerate(df.education_summary.unique()):
    plt.subplot(1,3,count + 1)
    df[df['education_summary'] == x].salary.value_counts(dropna=False).plot.pie
(autopct='%0.2f%%',colors=['pink','skyblue'])
    plt.title(x)
plt.tight_layout()
```



Desired Output:



Check the count of person in each these new education levels(high, medium, low) by "salary" levels and visualize it with countplot

In [1098]:

```
# Your Code is Here
df.groupby('salary').education_summary.value_counts(dropna=False)
```

Out[1098]:

```
salary  education_summary
<=50K   medium_level_grade    16539
        high_level_grade       4155
        low_level_grade        4004
>50K    high_level_grade       3909
        medium_level_grade     3686
        low_level_grade         244
Name: education_summary, dtype: int64
```

Desired Output: salary education_summary <=50K medium_level_grade 16539 high_level_grade 4155

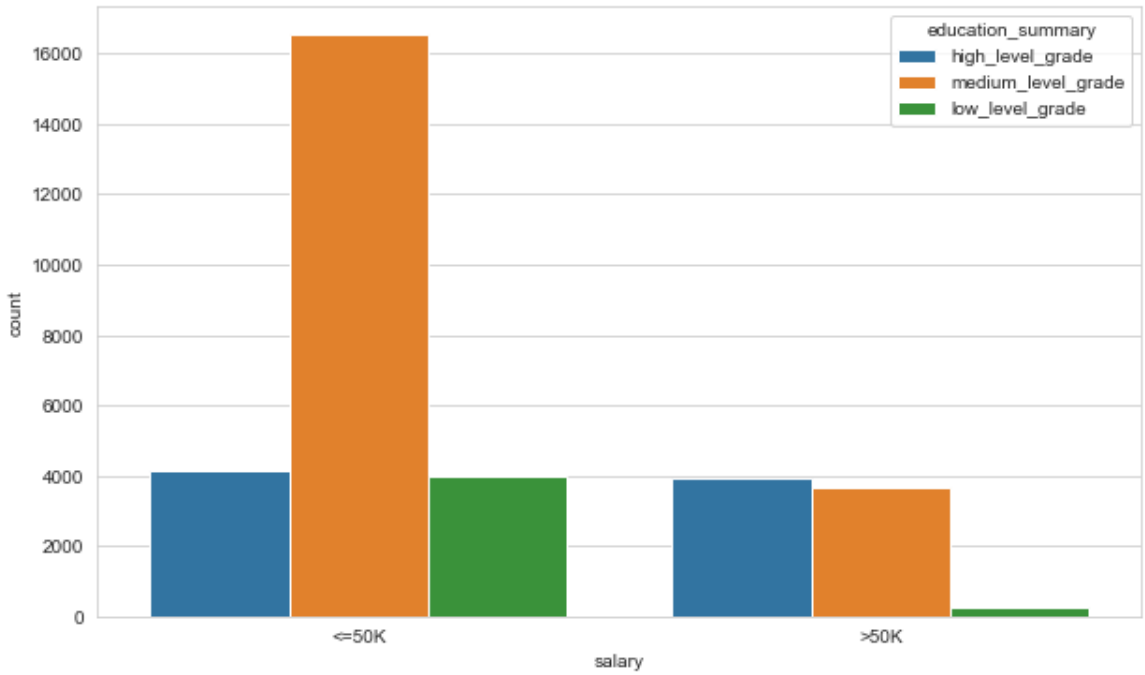
low_level_grade 4004 >50K high_level_grade 3909 medium_level_grade 3686 low_level_grade 244 Name: education_summary, dtype: int64

In [1099]:

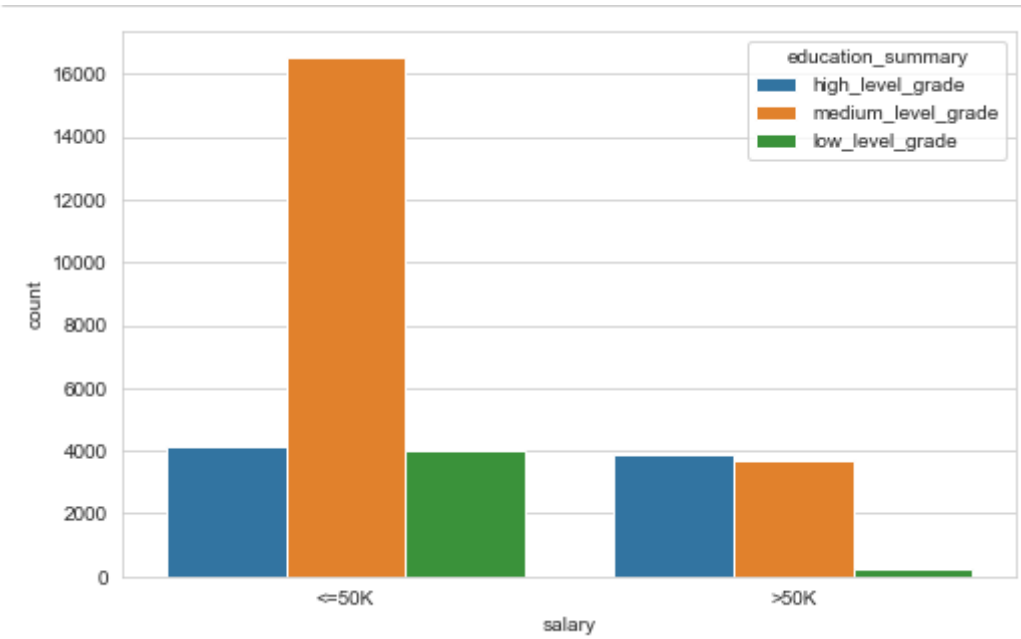
```
# Your Code is Here
sns.countplot(x='salary', hue='education_summary', data=df)
```

Out[1099]:

```
<AxesSubplot:xlabel='salary', ylabel='count'>
```



Desired Output:



Check the the percentage distribution of person in each these new education levels(high, medium, low) by "salary" levels and visualize it with pie plot separately

In [1100]:

```
# Your Code is Here
df.groupby('salary').education_summary.value_counts(dropna=False) / df.groupby
('salary').education_summary.count()
```

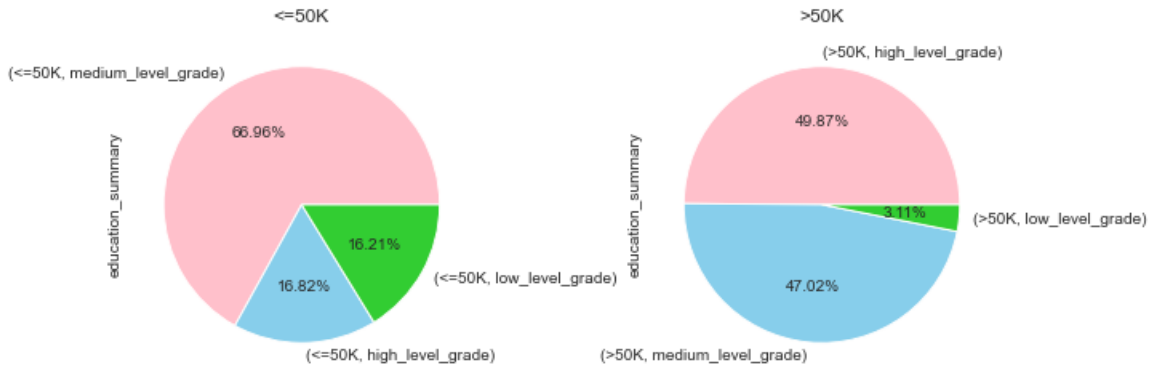
Out[1100]:

```
salary  education_summary
<=50K   medium_level_grade    0.670
        high_level_grade      0.168
        low_level_grade       0.162
>50K    high_level_grade      0.499
        medium_level_grade    0.470
        low_level_grade       0.031
Name: education_summary, dtype: float64
```

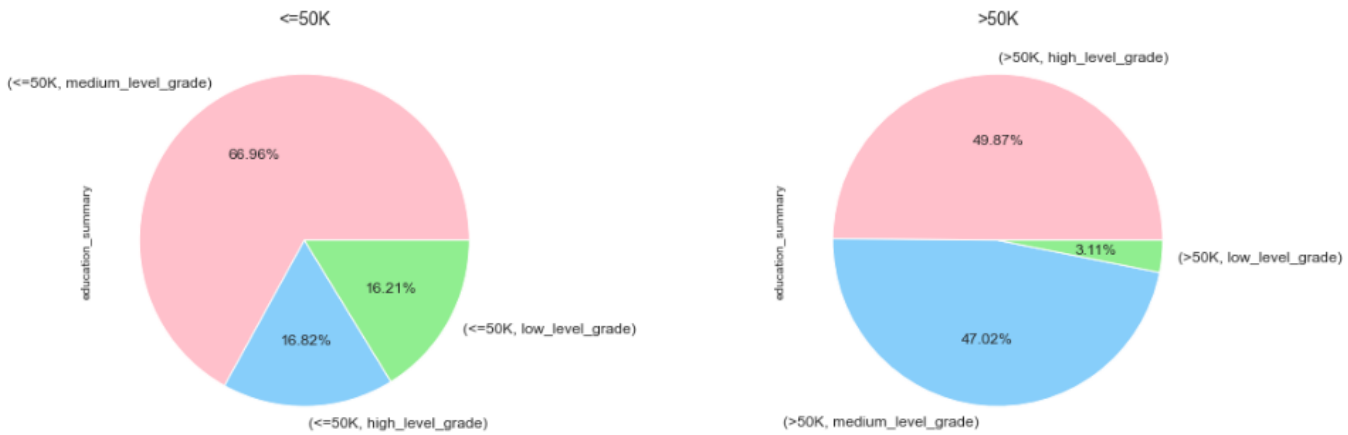
Desired Output: salary education_summary <=50K medium_level_grade 0.670 high_level_grade 0.168 low_level_grade 0.162 >50K high_level_grade 0.499 medium_level_grade 0.470 low_level_grade 0.031 Name: education_summary, dtype: float64

In [1101]:

```
# Your Code is Here
for count,x in enumerate(df.salary.unique()):
    plt.subplot(1,2,count + 1)
    df[df['salary'] == x].groupby('salary').education_summary.value_counts(dropn
a=False).plot.pie(autopct='%0.2f%%',colors=['pink','skyblue','limegreen'])
    plt.title(x)
plt.tight_layout()
```



Desired Output:



In [1102]:

```
# Your Code is Here
(df.groupby('salary').education_summary.value_counts(dropna=False) / df.groupby
('salary').education_summary.count()).to_frame().rename(columns={'education_summ
ary':'percentage'}).reset_index().sort_values(['salary','percentage'])
```

Out[1102]:

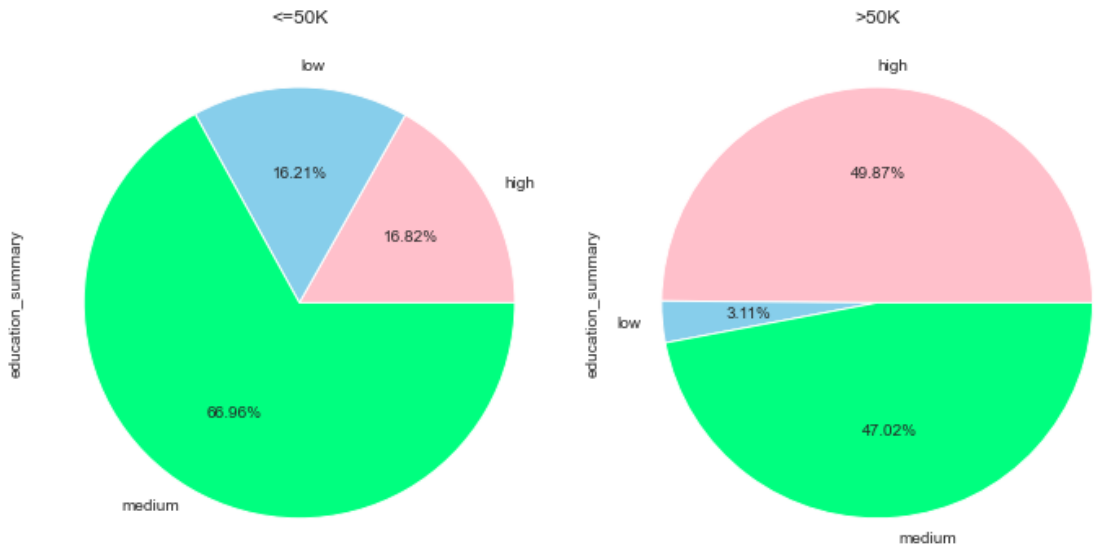
	salary	education_summary	percentage
2	<=50K	low_level_grade	0.162
1	<=50K	high_level_grade	0.168
0	<=50K	medium_level_grade	0.670
5	>50K	low_level_grade	0.031
4	>50K	medium_level_grade	0.470
3	>50K	high_level_grade	0.499

Desired Output:

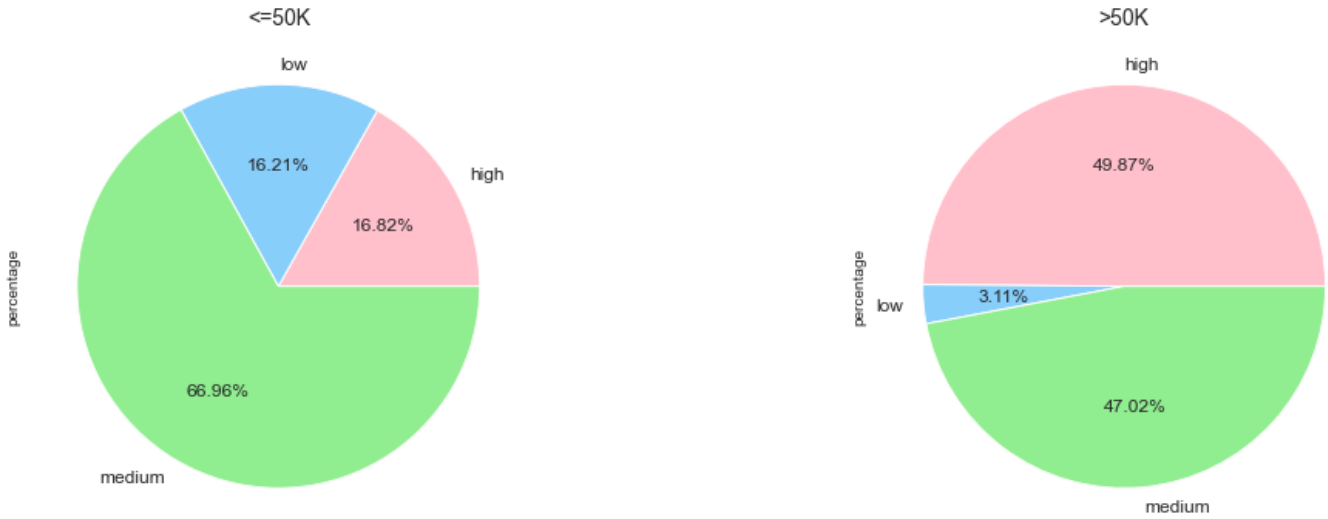
	salary	education_summary	percentage
1	<=50K	high_level_grade	0.168
2	<=50K	low_level_grade	0.162
0	<=50K	medium_level_grade	0.670
3	>50K	high_level_grade	0.499
5	>50K	low_level_grade	0.031
4	>50K	medium_level_grade	0.470

In [1103]:

```
# Your Code is Here
for count,x in enumerate(df.salary.unique()):
    plt.subplot(1,2,count + 1)
    df[df['salary'] == x].education_summary.map({'low_level_grade':'low','medium
_level_grade':'medium','high_level_grade':'high'}).value_counts(dropna=False).so
rt_index().plot.pie(autopct='%0.2f%%',colors=['pink','skyblue','springgreen'])
    plt.title(x)
plt.tight_layout()
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : When the education variable is examined in our data set, it is seen that most of them are high school graduates.
In our data set, medium level education seen most.
The rate of those who earns more than 50K in higher education is the highest. In low education, the rate of those who earns above 50K is the lowest.
One out of every two people with high education earns above 50K.
Most of those with low level education (94 percent) earn less than 50K salaries.
8 out of 10 people with intermediate education receive a salary of less than 50K.
Half of those who earns above 50K are those with higher education, and the other half are those with medium level grade.
In general, we can say that the higher the education level, the higher the salary.

marital_status & relationship

Detect the similarities between these features by comparing unique values

In [1104]:

```
# Your Code is Here
df.marital_status.value_counts(dropna=False)
```

Out[1104]:

```
Married-civ-spouse       14970
Never-married            10667
Divorced                  4441
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: marital_status, dtype: int64
```

Desired Output: Married-civ-spouse 14970 Never-married 10667 Divorced 4441 Separated 1025 Widowed 993 Married-spouse-absent 418 Married-AF-spouse 23 Name: marital_status, dtype: int64

In [1105]:

```
# Your Code is Here
df.relationship.value_counts(dropna=False)
```

Out[1105]:

```
Husband           13187
Not-in-family      8292
NaN                5064
Unmarried          3445
Wife               1568
Other-relative      981
Name: relationship, dtype: int64
```

Desired Output: Husband 13187 Not-in-family 8292 NaN 5064 Unmarried 3445 Wife 1568 Other-relative 981

Name: relationship, dtype: int64

In [1106]:

```
# Fill missing values with "Unknown" in the column of "relationship"

# Your Code is Here
df.relationship = df.relationship.fillna('Unknown')
```

In [1107]:

```
# Your Code is Here

df.groupby('relationship').marital_status.value_counts()
```

Out[1107]:

```
relationship    marital_status
Husband         Married-civ-spouse       13178
                Married-AF-spouse            9
Not-in-family   Never-married             4694
                Divorced                  2403
                Widowed                    547
                Separated                  420
                Married-spouse-absent      211
                Married-civ-spouse          17
Other-relative  Never-married              611
                Married-civ-spouse         124
                Divorced                   110
                Separated                   55
                Widowed                     48
                Married-spouse-absent       32
                Married-AF-spouse            1
Unknown         Never-married             4481
                Divorced                   328
                Separated                   99
                Married-civ-spouse          95
                Married-spouse-absent       45
                Widowed                     15
                Married-AF-spouse            1
Unmarried       Divorced                  1600
                Never-married              881
                Separated                  451
                Widowed                    383
                Married-spouse-absent      130
Wife            Married-civ-spouse        1556
                Married-AF-spouse           12
Name: marital_status, dtype: int64
```

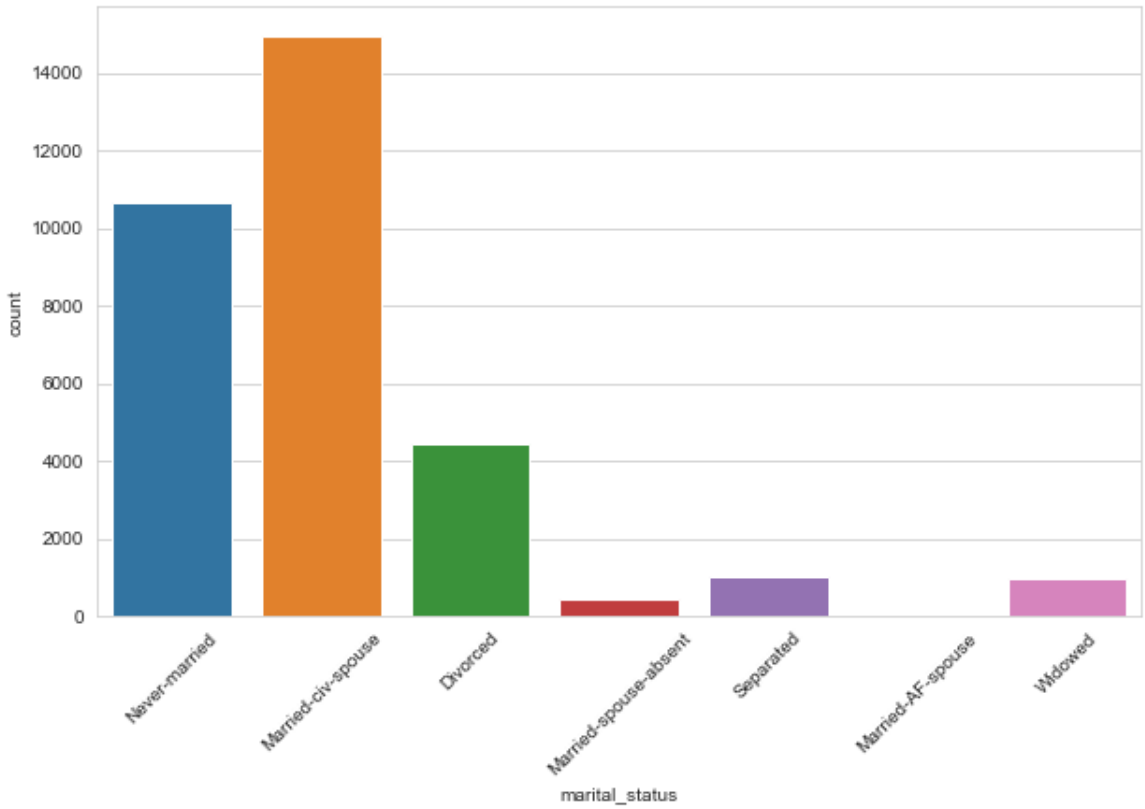
Desired Output: relationship marital_status Husband Married-civ-spouse 13178 Married-AF-spouse 9 Not-in-family Never-married 4694 Divorced 2403 Widowed 547 Separated 420 Married-spouse-absent 211 Married-civ-spouse 17 Other-relative Never-married 611 Married-civ-spouse 124 Divorced 110 Separated 55 Widowed 48 Married-spouse-absent 32 Married-AF-spouse 1 Unknown Never-married 4481 Divorced 328 Separated 99 Married-civ-spouse 95 Married-spouse-absent 45 Widowed 15 Married-AF-spouse 1 Unmarried Divorced 1600 Never-married 881 Separated 451 Widowed 383 Married-spouse-absent 130 Wife Married-civ-spouse 1556 Married-AF-spouse 12 Name: marital_status, dtype: int64

Assessment : When marital_status in our dataset is examined, it is seen that married-civ-spouse is the most common.
When the data set is examined, husband value seen most.

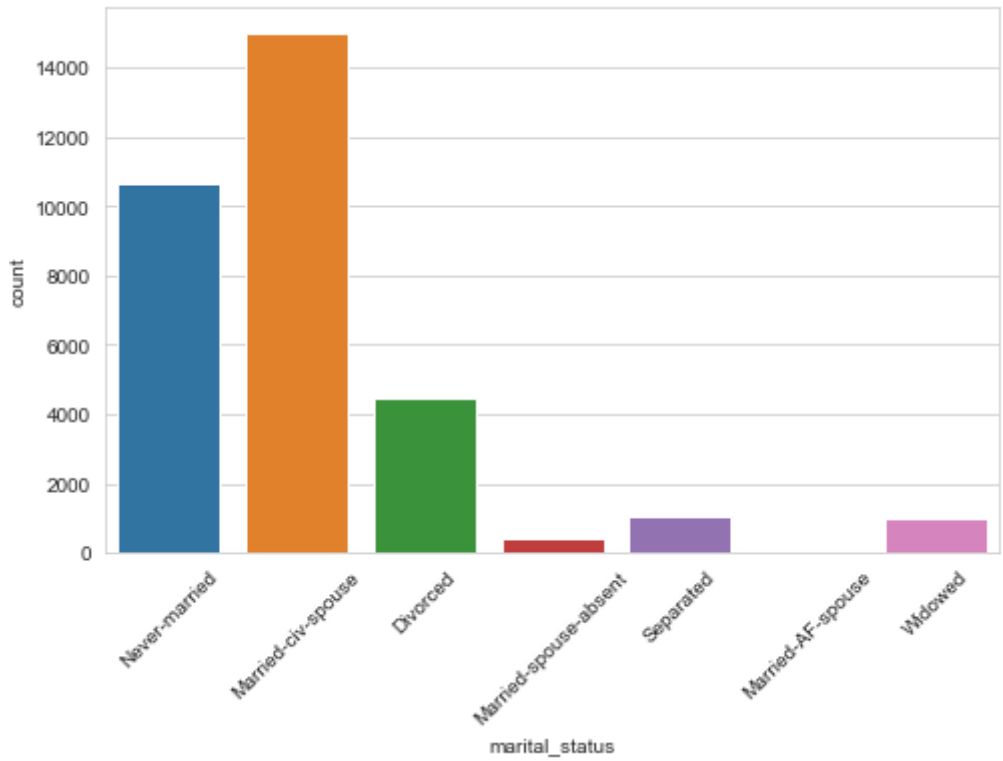
Visualize the count of person in each categories

In [1108]:

```
# Your Code is Here
sns.countplot(x='marital_status',data=df)
plt.xticks(rotation = 45);
```



Desired Output:



Check the count of person in each "salary" levels by categories and visualize it with countplot

In [1109]:

```
# Your Code is Here

df.groupby('marital_status').salary.value_counts(dropna=False)
```

Out[1109]:

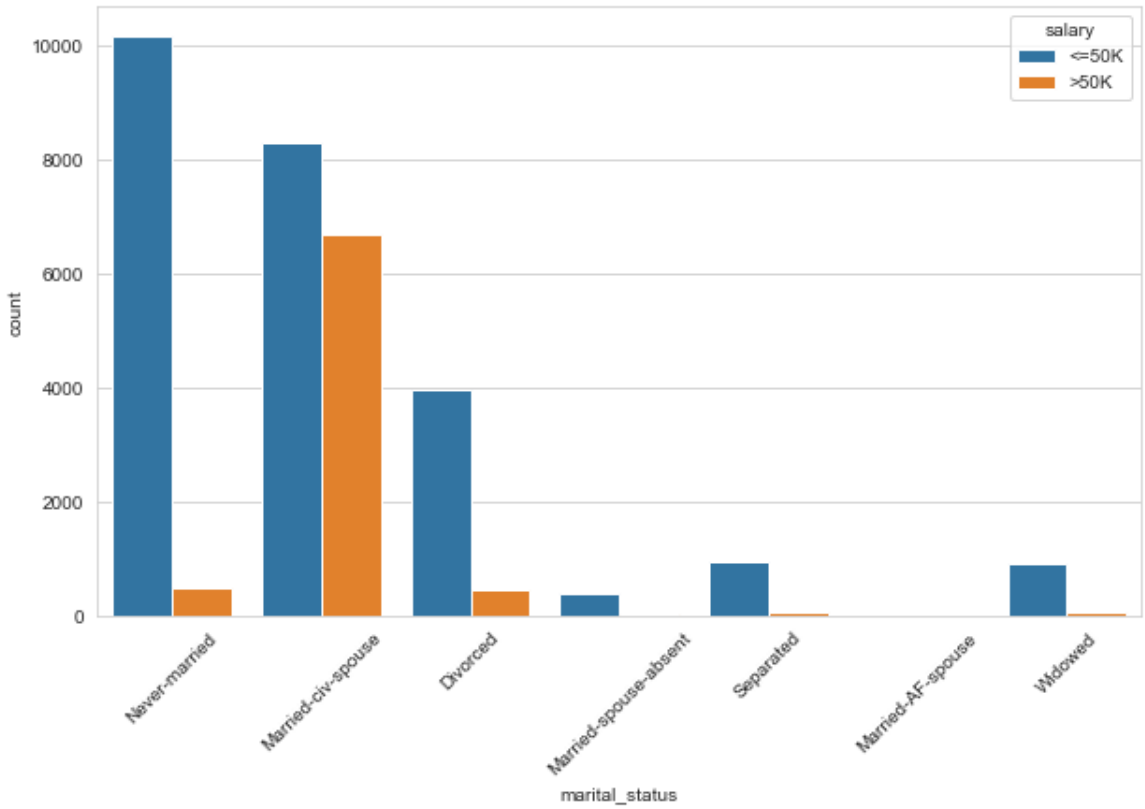
```
marital_status          salary
Divorced                <=50K      3978
                        >50K        463
Married-AF-spouse       <=50K        13
                        >50K         10
Married-civ-spouse      <=50K      8280
                        >50K       6690
Married-spouse-absent   <=50K       384
                        >50K         34
Never-married           <=50K     10176
                        >50K        491
Separated               <=50K       959
                        >50K         66
Widowed                 <=50K       908
                        >50K         85
Name: salary, dtype: int64
```

Desired Output: marital_status salary Divorced <=50K 3978 >50K 463 Married-AF-spouse <=50K 13 >50K 10 Married-civ-spouse <=50K 8280 >50K 6690 Married-spouse-absent <=50K 384 >50K 34 Never-married <=50K

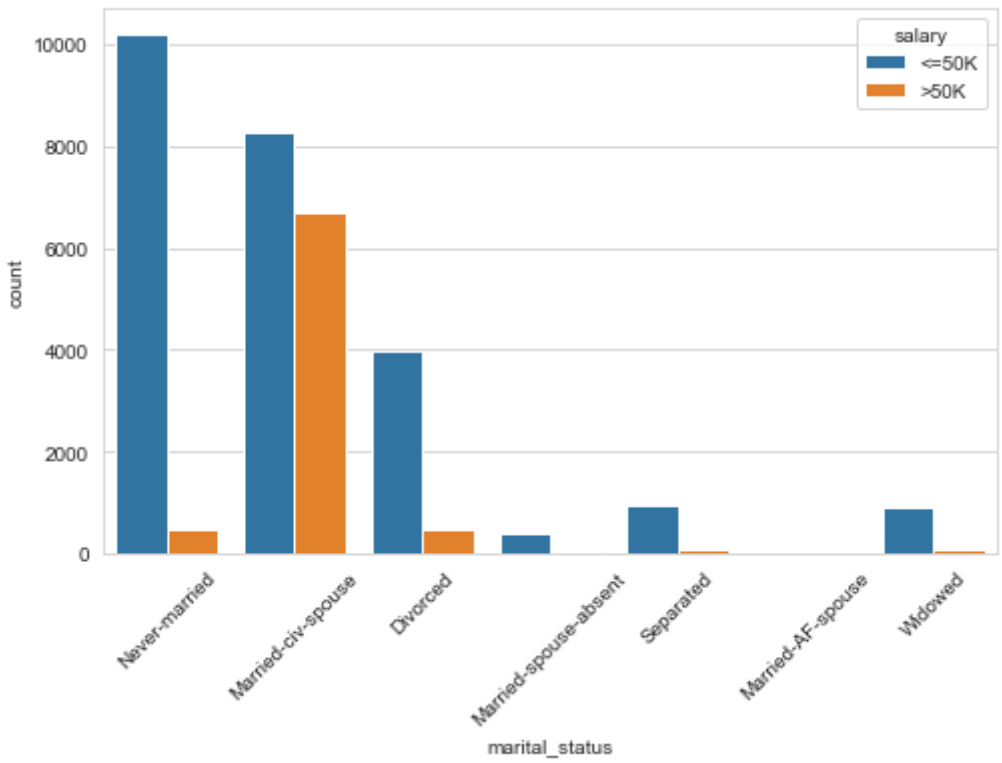
10176 >50K 491 Separated <=50K 959 >50K 66 Widowed <=50K 908 >50K 85 Name: salary, dtype: int64

In [1110]:

```
# Your Code is Here
sns.countplot(x='marital_status',data=df, hue='salary')
plt.xticks(rotation=45);
```



Desired Output:



Decrease the number of categories in "marital_status" feature as married, and unmarried and create a new feature with this new categorical data

In [1111]:

```
def mapping_marital_status(x):
    if x in ["Never-married", "Divorced", "Separated", "Widowed"]:
        return "unmarried"
    elif x in ["Married-civ-spouse", "Married-AF-spouse", "Married-spouse-absen
t"]:
        return "married"
```

In [1112]:

```
# Your Code is Here
df.marital_status.apply(lambda x: mapping_marital_status(x)).value_counts(dropna
=False)
```

Out[1112]:

```
unmarried    17126
married      15411
Name: marital_status, dtype: int64
```

Desired Output: unmarried 17126 married 15411 Name: marital_status, dtype: int64

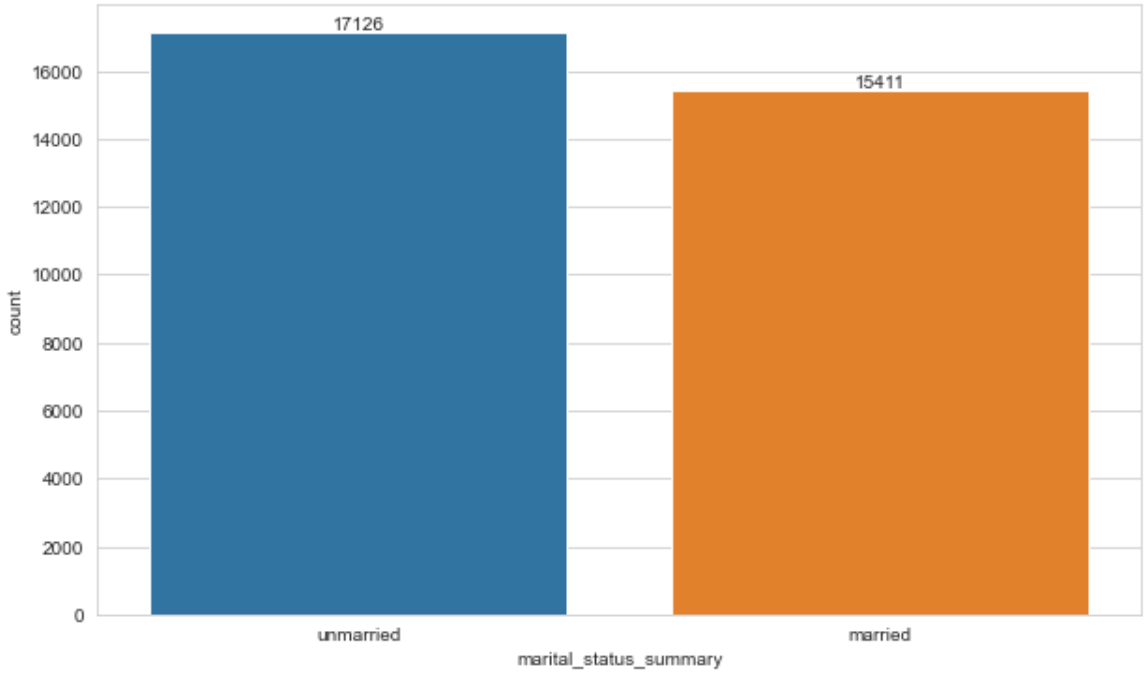
In [1113]:

```
# By using "mapping_marital_status" def function above, create a new column name
d "marital_status_summary"

# Your Code is Here
df['marital_status_summary'] = df.marital_status.apply(lambda x: mapping_marital
_status(x))
```

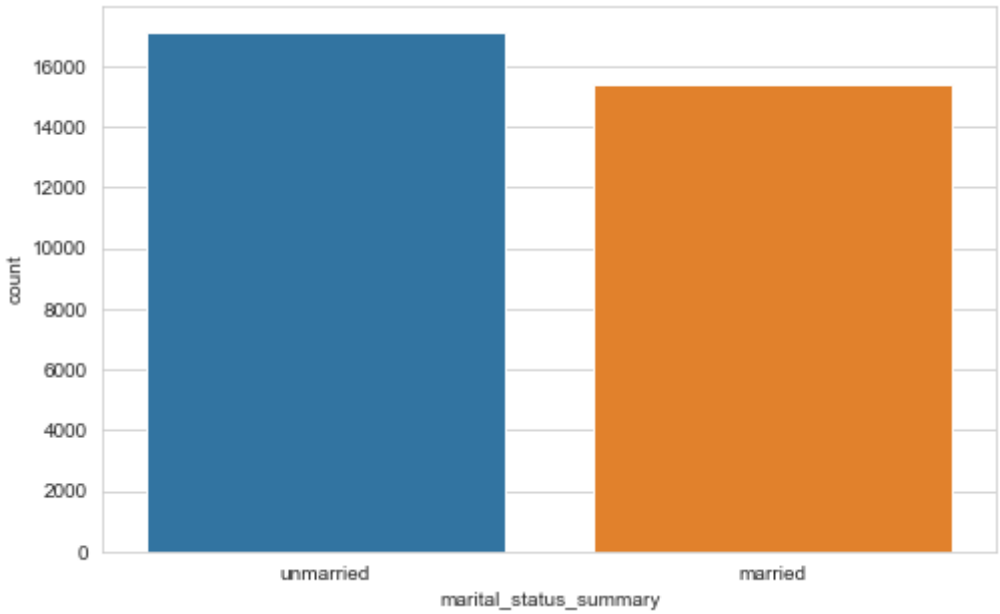
Visualize the count of person in each categories for these new marital status (married, unmarried)

In [1114]:

```
# Your Code is Here

g = sns.countplot(x='marital_status_summary', data=df)
plt.bar_label(g.containers[0]);
```



Desired Output:



Check the count of person in each "salary" levels by these new marital status (married, unmarried) and visualize it with countplot

In [1115]:

```
# Your Code is Here
df.groupby('marital_status_summary').salary.value_counts(dropna=False)
```

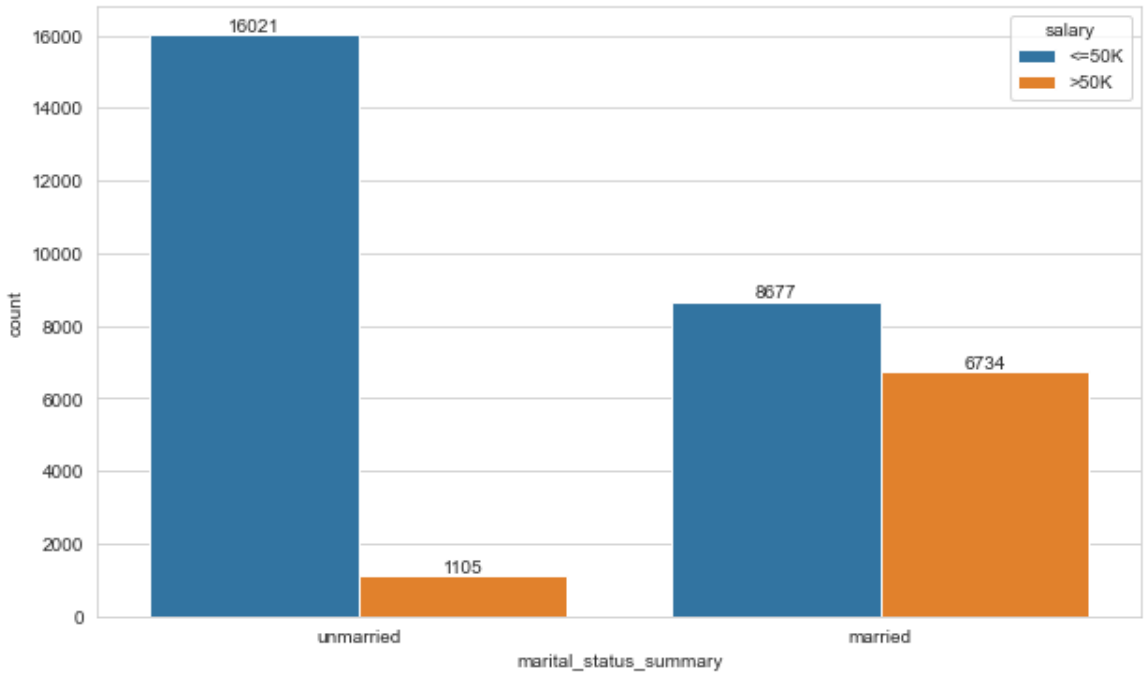
Out[1115]:

```
marital_status_summary  salary
married                 <=50K      8677
                        >50K       6734
unmarried               <=50K     16021
                        >50K       1105
Name: salary, dtype: int64
```

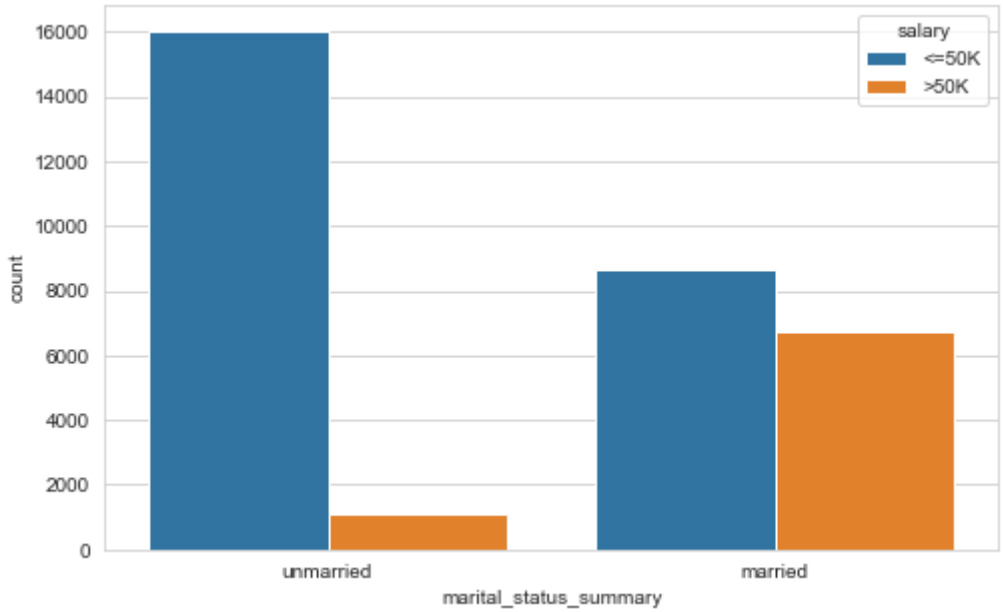
Desired Output: marital_status_summary salary married <=50K 8677 >50K 6734 unmarried <=50K 16021 >50K 1105 Name: salary, dtype: int64

In [1116]:

```
# Your Code is Here

g = sns.countplot(x='marital_status_summary', hue='salary',data=df)
plt.bar_label(g.containers[0]);
plt.bar_label(g.containers[1]);
```



Desired Output:



Check the percentage distribution of person in each "salary" levels by each new marital status (married, unmarried) and visualize it with pie plot separately

In [1117]:

```
# Your Code is Here

df.groupby('marital_status_summary').salary.value_counts(dropna=False, normalize
= True)
```

Out[1117]:

```
marital_status_summary  salary
married                 <=50K    0.563
                        >50K     0.437
unmarried               <=50K    0.935
                        >50K     0.065
Name: salary, dtype: float64
```

Desired Output: marital_status_summary salary married <=50K 0.563 >50K 0.437 unmarried <=50K 0.935 >50K

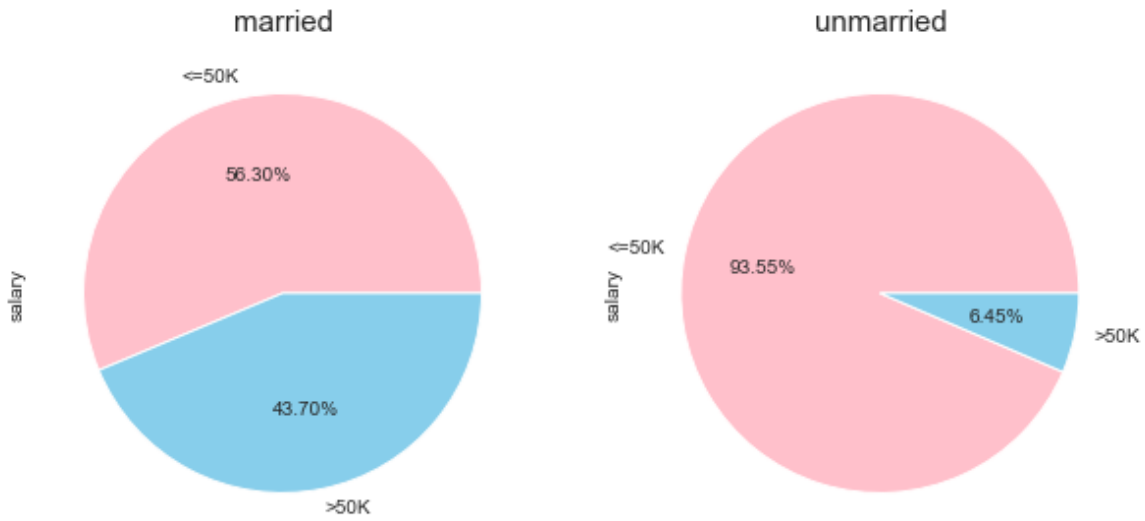
0.065 Name: salary, dtype: float64

In [1118]:

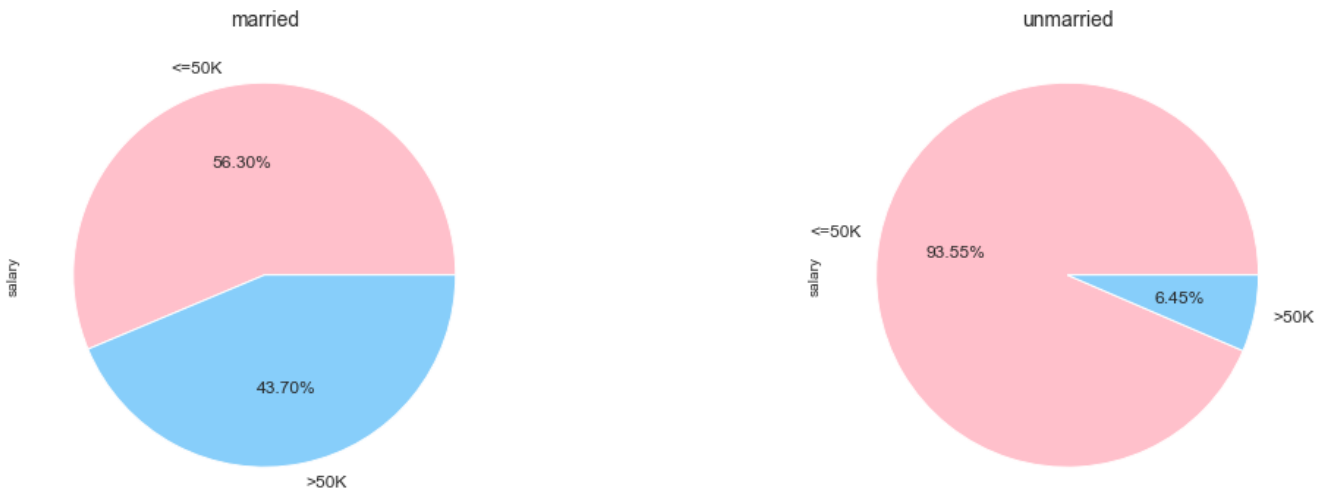
```
# Your Code is Here
plt.subplot(1,2,1)
df[df.marital_status_summary == 'married'].salary.value_counts().plot.pie(autopc
t='%0.2f%%',colors=['pink','skyblue'])
plt.title('married', fontsize=15)
plt.subplot(1,2,2)
df[df.marital_status_summary == 'unmarried'].salary.value_counts().plot.pie(auto
pct='%0.2f%%',colors=['pink','skyblue'])
plt.title('unmarried', fontsize=15)
```

Out[1118]:

```
Text(0.5, 1.0, 'unmarried')
```



Desired Output:



Check the count of person in each these new marital status (married, unmarried) by "salary" levels and visualize it with countplot

In [1119]:

```
# Your Code is Here
df.groupby('salary').marital_status_summary.value_counts(dropna=False)
```

Out[1119]:

```
salary  marital_status_summary
<=50K   unmarried                16021
        married                   8677
>50K    married                   6734
        unmarried                 1105
Name: marital_status_summary, dtype: int64
```

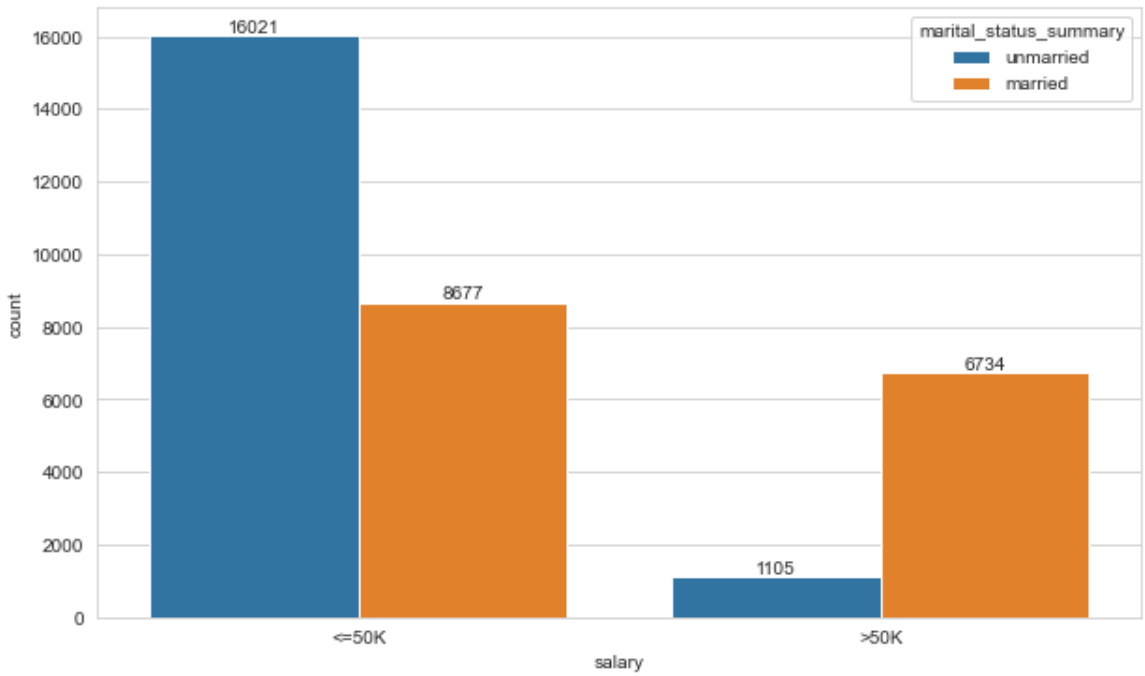
Desired Output: salary marital_status_summary <=50K unmarried 16021 married 8677 >50K married 6734 unmarried 1105 Name: marital_status_summary, dtype: int64

In [1120]:

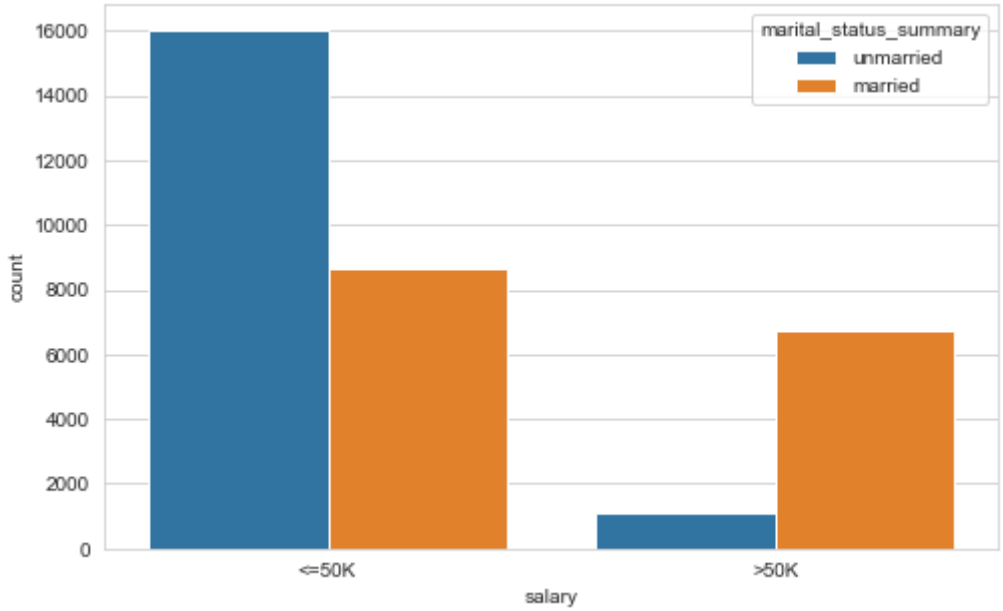
```
# Your Code is Here
g = sns.countplot(x='salary', hue='marital_status_summary', data=df)
plt.bar_label(g.containers[0])
plt.bar_label(g.containers[1])
```

Out[1120]:

```
[Text(0, 0, '8677'), Text(0, 0, '6734')]
```



Desired Output:



Check the the percentage distribution of person in each these new marital status (married, unmarried) by "salary" levels and visualize it with pie plot separately

In [1121]:

```
# Your Code is Here
df.groupby('salary').marital_status_summary.value_counts(dropna=False, normalize
= True)
```

Out[1121]:

```
salary  marital_status_summary
<=50K   unmarried                0.649
        married                  0.351
>50K    married                  0.859
        unmarried                0.141
Name: marital_status_summary, dtype: float64
```

Desired Output: salary marital_status_summary <=50K unmarried 0.649 married 0.351 >50K married 0.859

unmarried 0.141 Name: marital_status_summary, dtype: float64

In [1122]:

```
# Your Code is Here
df.groupby('salary').marital_status_summary.value_counts(dropna=False, normalize
= True).to_frame().rename(columns = {'marital_status_summary':'percentage'}).res
et_index().sort_values(by=['salary','marital_status_summary'])
```

Out[1122]:

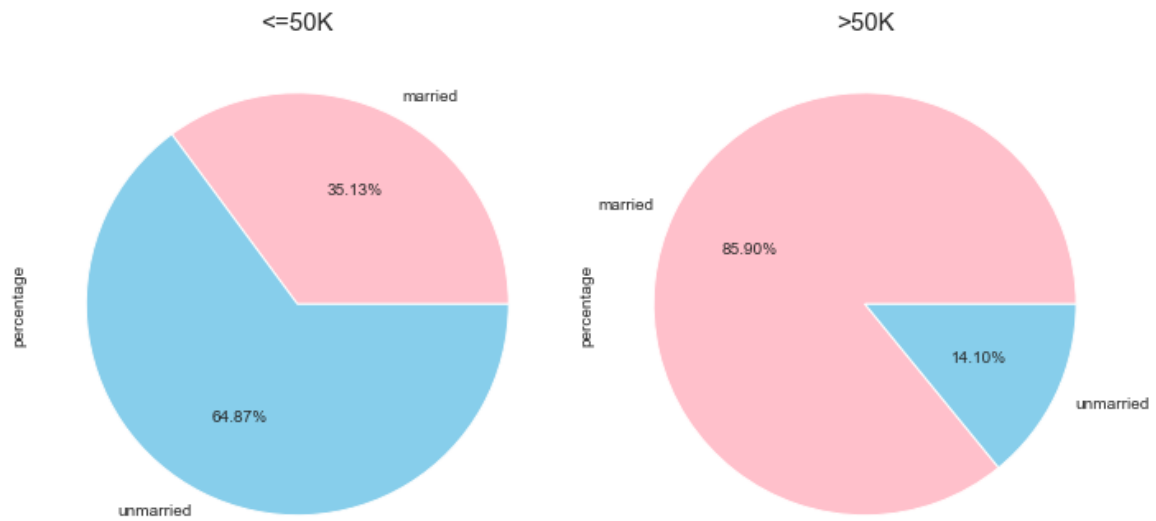
	salary	marital_status_summary	percentage
1	<=50K	married	0.351
0	<=50K	unmarried	0.649
2	>50K	married	0.859
3	>50K	unmarried	0.141

Desired Output:

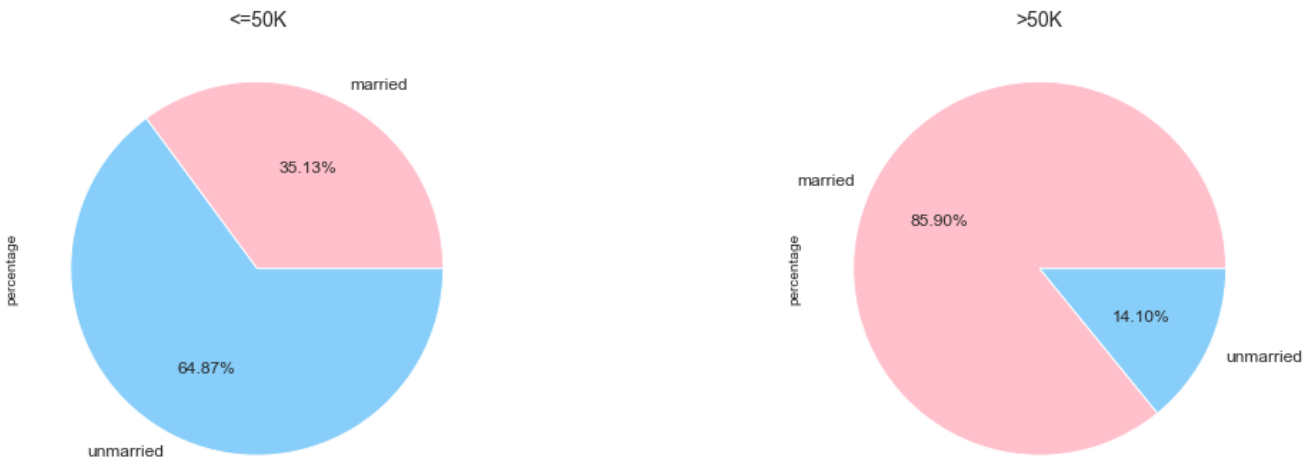
	salary	marital_status_summary	percentage
1	<=50K	married	0.351
0	<=50K	unmarried	0.649
2	>50K	married	0.859
3	>50K	unmarried	0.141

In [1123]:

```
# Your Code is Here
plt.subplot(1,2,1)
df[df.salary == '<=50K'].marital_status_summary.value_counts().sort_index().plo
t.pie(autopct='%0.2f%%',colors=['pink','skyblue'])
plt.title('<=50K', fontsize=15)
plt.ylabel('percentage')
plt.subplot(1,2,2)
df[df.salary == '>50K'].marital_status_summary.value_counts().sort_index().plot.
pie(autopct='%0.2f%%',colors=['pink','skyblue'])
plt.title('>50K', fontsize=15)
plt.ylabel('percentage')
plt.tight_layout()
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : half of our data set is married and the other half is unmarried.
One out of every two married couples earns over 50K.
The vast majority (93 percent) of unmarried people earn less than 50K.
The majority of those who earn over 50K are those who are married with 85 percent.

workclass

Check the count of person in each categories and visualize it with countplot

In [1124]:

```
# Your Code is Here
df.workclass.value_counts(dropna=False)
```

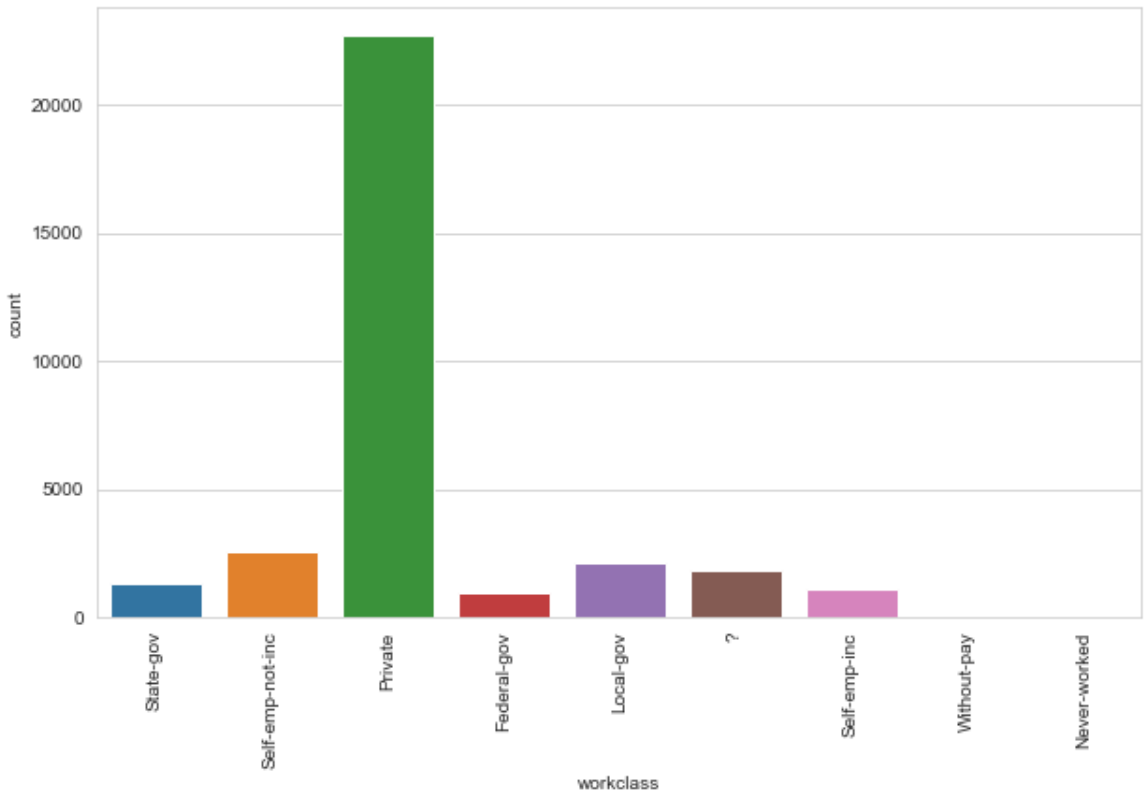
Out[1124]:

```
Private             22673
Self-emp-not-inc     2540
Local-gov            2093
?                    1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

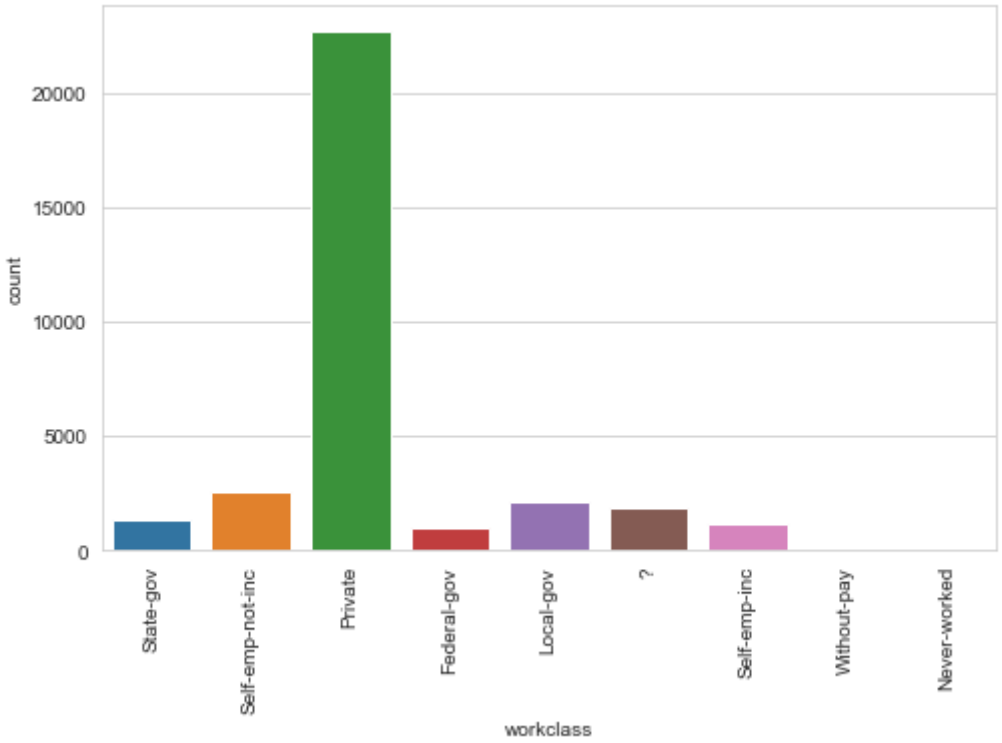
Desired Output: Private 22673 Self-emp-not-inc 2540 Local-gov 2093 ? 1836 State-gov 1298 Self-emp-inc 1116 Federal-gov 960 Without-pay 14 Never-worked 7 Name: workclass, dtype: int64

In [1125]:

```
# Your Code is Here
sns.countplot(x='workclass', data=df)
plt.xticks(rotation=90);
```



Desired Output:



Replace the value "?" to the value "Unknown"

In [1126]:

```
# Replace "?" values with "Unkown"

# Your Code is Here
df.workclass = df.workclass.replace({'?':'Unknown'})
```

Check the count of person in each "salary" levels by workclass groups and visualize it with countplot

In [1127]:

```
# Your Code is Here

df.groupby('workclass').salary.value_counts(dropna=False)
```

Out[1127]:

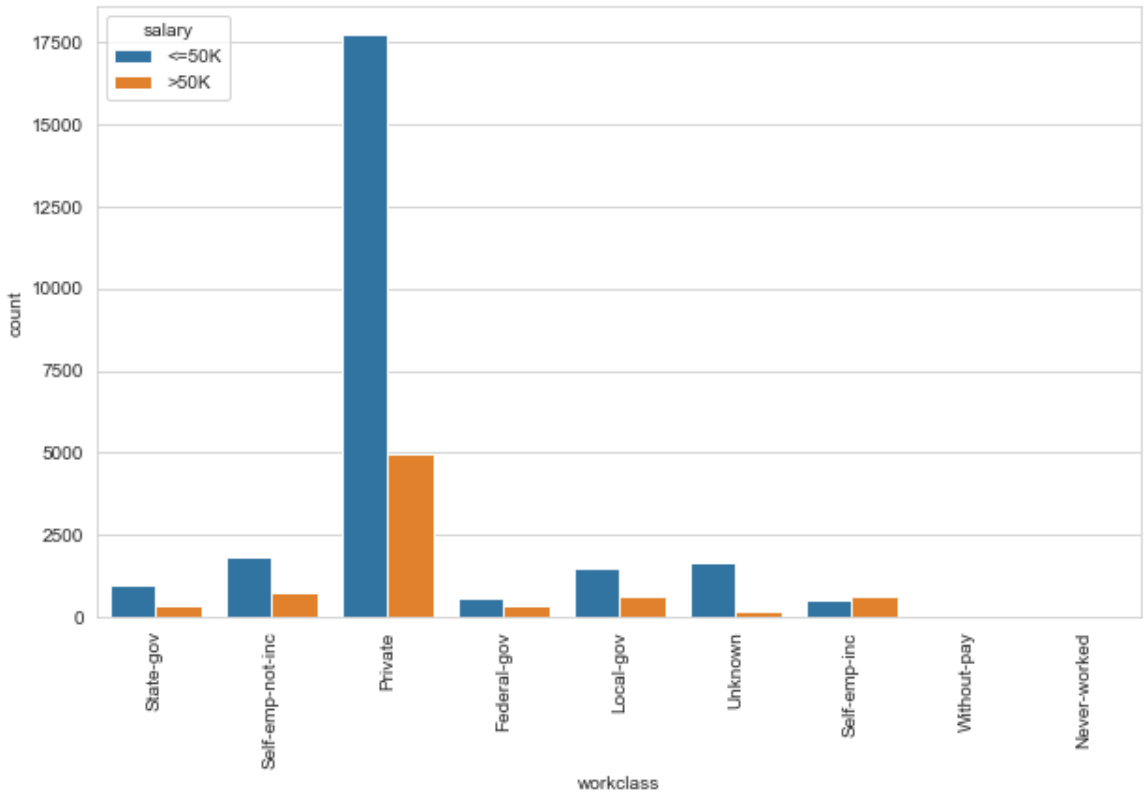
```
workclass          salary
Federal-gov        <=50K       589
                   >50K        371
Local-gov          <=50K      1476
                   >50K        617
Never-worked       <=50K         7
Private            <=50K     17712
                   >50K       4961
Self-emp-inc       >50K        622
                   <=50K       494
Self-emp-not-inc   <=50K      1816
                   >50K        724
State-gov          <=50K       945
                   >50K        353
Unknown            <=50K      1645
                   >50K        191
Without-pay        <=50K        14
Name: salary, dtype: int64
```

Desired Output: workclass salary Federal-gov <=50K 589 >50K 371 Local-gov <=50K 1476 >50K 617 Never-worked <=50K 7 Private <=50K 17712 >50K 4961 Self-emp-inc >50K 622 <=50K 494 Self-emp-not-inc <=50K 1816 >50K 724 State-gov <=50K 945 >50K 353 Unknown <=50K 1645 >50K 191 Without-pay <=50K 14 Name:

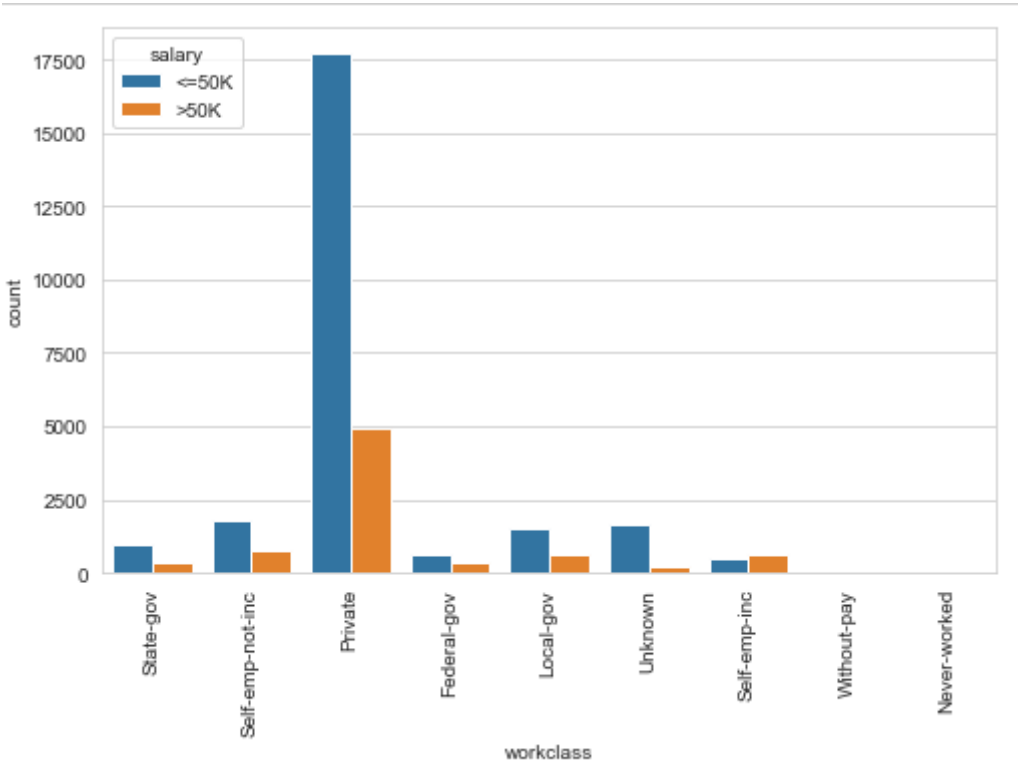
salary, dtype: int64

In [1128]:

```
# Your Code is Here
sns.countplot(x='workclass',data=df,hue='salary')
plt.xticks(rotation = 90);
```



Desired Output:



Check the percentage distribution of person in each "salary" levels by each workclass groups and visualize it with bar plot

In [1129]:

```
# Your Code is Here
df.groupby('workclass').salary.value_counts(dropna=False, normalize = True)
```

Out[1129]:

```
workclass          salary
Federal-gov        <=50K    0.614
                   >50K     0.386
Local-gov          <=50K    0.705
                   >50K     0.295
Never-worked       <=50K    1.000
Private            <=50K    0.781
                   >50K     0.219
Self-emp-inc       >50K     0.557
                   <=50K    0.443
Self-emp-not-inc   <=50K    0.715
                   >50K     0.285
State-gov          <=50K    0.728
                   >50K     0.272
Unknown            <=50K    0.896
                   >50K     0.104
Without-pay        <=50K    1.000
Name: salary, dtype: float64
```

Desired Output: workclass salary Federal-gov <=50K 0.614 >50K 0.386 Local-gov <=50K 0.705 >50K 0.295 Never-worked <=50K 1.000 Private <=50K 0.781 >50K 0.219 Self-emp-inc >50K 0.557 <=50K 0.443 Self-emp-not-inc <=50K 0.715 >50K 0.285 State-gov <=50K 0.728 >50K 0.272 Unknown <=50K 0.896 >50K 0.104

Without-pay <=50K 1.000 Name: salary, dtype: float64

In [1130]:

```
# Your Code is Here
df.groupby('workclass').salary.value_counts(dropna=False, normalize = True).to_f
rame().rename(columns={'salary':'percentage'}).reset_index()
```

Out[1130]:

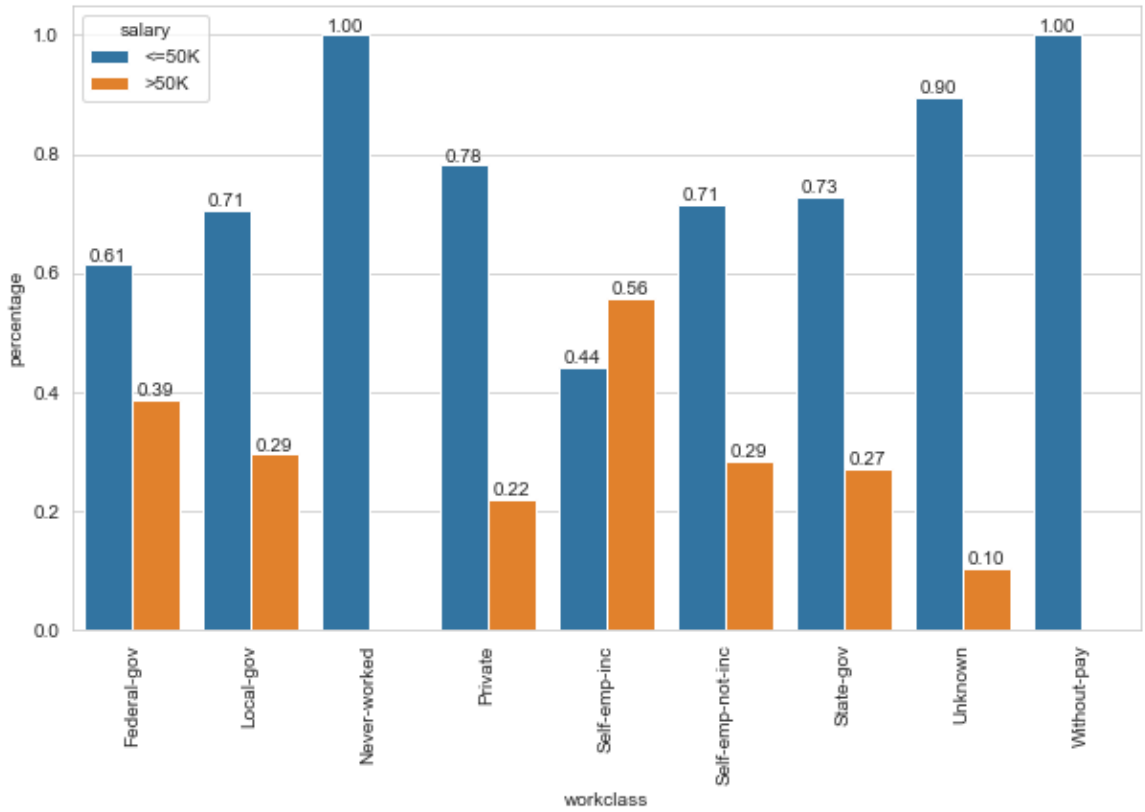
	workclass	salary	percentage
0	Federal-gov	<=50K	0.614
1	Federal-gov	>50K	0.386
2	Local-gov	<=50K	0.705
3	Local-gov	>50K	0.295
4	Never-worked	<=50K	1.000
5	Private	<=50K	0.781
6	Private	>50K	0.219
7	Self-emp-inc	>50K	0.557
8	Self-emp-inc	<=50K	0.443
9	Self-emp-not-inc	<=50K	0.715
10	Self-emp-not-inc	>50K	0.285
11	State-gov	<=50K	0.728
12	State-gov	>50K	0.272
13	Unknown	<=50K	0.896
14	Unknown	>50K	0.104
15	Without-pay	<=50K	1.000

Desired Output:

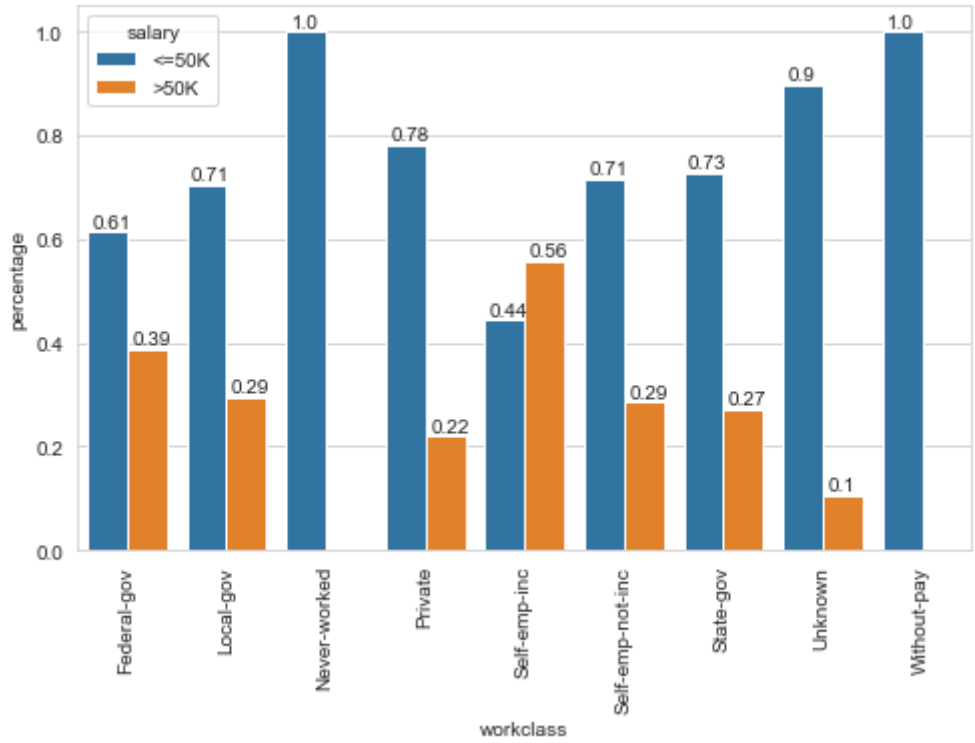
	workclass	salary	percentage
0	Federal-gov	<=50K	0.614
1	Federal-gov	>50K	0.386
2	Local-gov	<=50K	0.705
3	Local-gov	>50K	0.295
4	Never-worked	<=50K	1.000
5	Private	<=50K	0.781
6	Private	>50K	0.219
8	Self-emp-inc	<=50K	0.443
7	Self-emp-inc	>50K	0.557
9	Self-emp-not-inc	<=50K	0.715
10	Self-emp-not-inc	>50K	0.285
11	State-gov	<=50K	0.728
12	State-gov	>50K	0.272
13	Unknown	<=50K	0.896
14	Unknown	>50K	0.104
15	Without-pay	<=50K	1.000

In [1131]:

```
# Your Code is Here
df_temp = df.groupby('workclass').salary.value_counts(dropna=False, normalize =
True).to_frame().rename(columns={'salary':'percentage'}).reset_index()
g = sns.barplot(x='workclass', y='percentage', data=df_temp, hue='salary')
plt.bar_label(g.containers[0], fmt='%.2f')
plt.bar_label(g.containers[1], fmt='%.2f')
plt.xticks(rotation= 90);
```



Desired Output:



Check the count of person in each workclass groups by "salary" levels and visualize it with countplot

In [1132]:

```
# Your Code is Here
df.groupby('salary').workclass.value_counts()
```

Out[1132]:

```
salary  workclass
<=50K   Private             17712
        Self-emp-not-inc     1816
        Unknown              1645
        Local-gov            1476
        State-gov             945
        Federal-gov           589
        Self-emp-inc          494
        Without-pay            14
        Never-worked            7
>50K    Private             4961
        Self-emp-not-inc     724
        Self-emp-inc         622
        Local-gov            617
        Federal-gov          371
        State-gov            353
        Unknown              191
Name: workclass, dtype: int64
```

Desired Output: salary workclass <=50K Private 17712 Self-emp-not-inc 1816 Unknown 1645 Local-gov 1476 State-gov 945 Federal-gov 589 Self-emp-inc 494 Without-pay 14 Never-worked 7 >50K Private 4961 Self-emp-not-inc 724 Self-emp-inc 622 Local-gov 617 Federal-gov 371 State-gov 353 Unknown 191 Name: workclass,

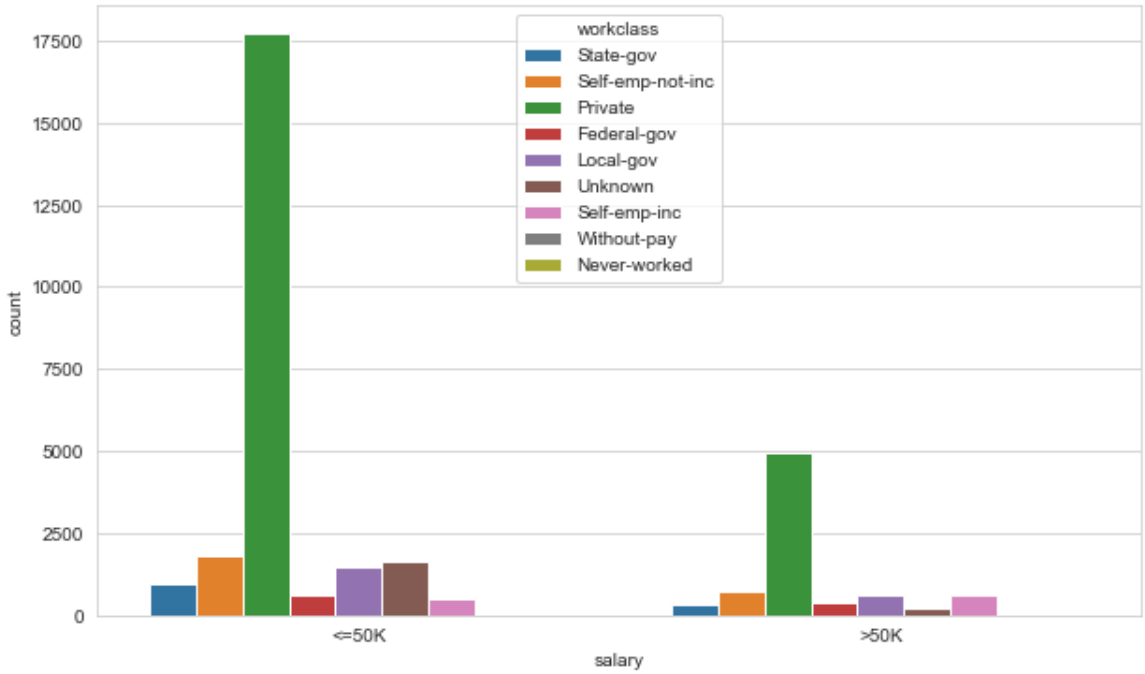
dtype: int64

In [1133]:

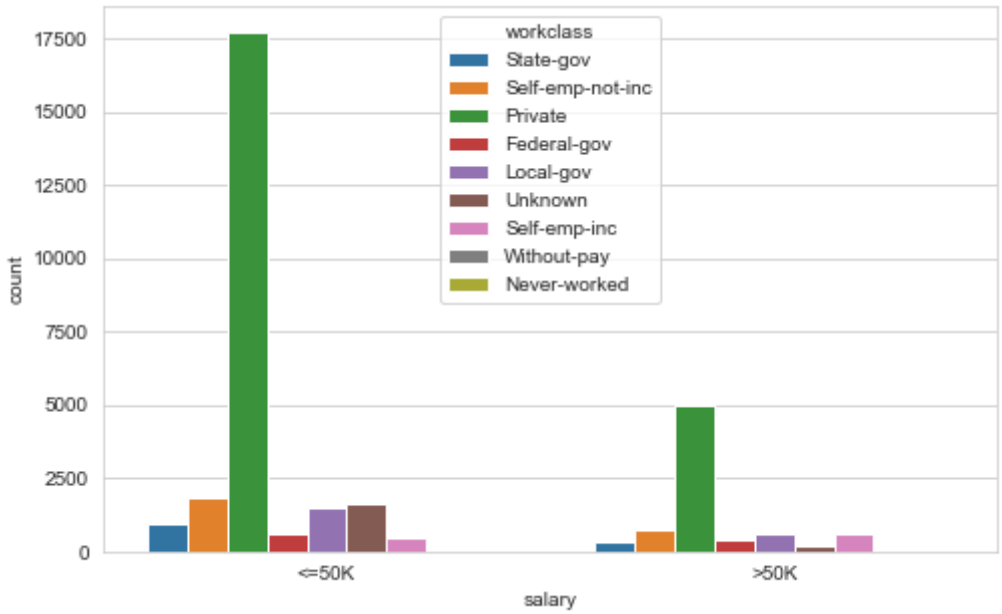
```
# Your Code is Here
sns.countplot(x='salary', hue='workclass',data=df)
```

Out[1133]:

```
<AxesSubplot:xlabel='salary', ylabel='count'>
```



Desired Output:



Check the the percentage distribution of person in each workclass groups by "salary" levels and visualize it with countplot

In [1134]:

```
# Your Code is Here
df.groupby('salary').workclass.value_counts(dropna=False, normalize=True)
```

Out[1134]:

```
salary  workclass
<=50K   Private             0.717
        Self-emp-not-inc    0.074
        Unknown             0.067
        Local-gov           0.060
        State-gov           0.038
        Federal-gov         0.024
        Self-emp-inc        0.020
        Without-pay         0.001
        Never-worked        0.000
>50K    Private             0.633
        Self-emp-not-inc    0.092
        Self-emp-inc        0.079
        Local-gov           0.079
        Federal-gov         0.047
        State-gov           0.045
        Unknown             0.024
Name: workclass, dtype: float64
```

Desired Output: salary workclass <=50K Private 0.717 Self-emp-not-inc 0.074 Unknown 0.067 Local-gov 0.060 State-gov 0.038 Federal-gov 0.024 Self-emp-inc 0.020 Without-pay 0.001 Never-worked 0.000 >50K Private 0.633 Self-emp-not-inc 0.092 Self-emp-inc 0.079 Local-gov 0.079 Federal-gov 0.047 State-gov 0.045 Unknown

0.024 Name: workclass, dtype: float64

In [1135]:

```
# Your Code is Here
df.groupby('salary').workclass.value_counts(dropna=False, normalize=True).rename
('percentage').reset_index().sort_values(['salary','workclass'])
```

Out[1135]:

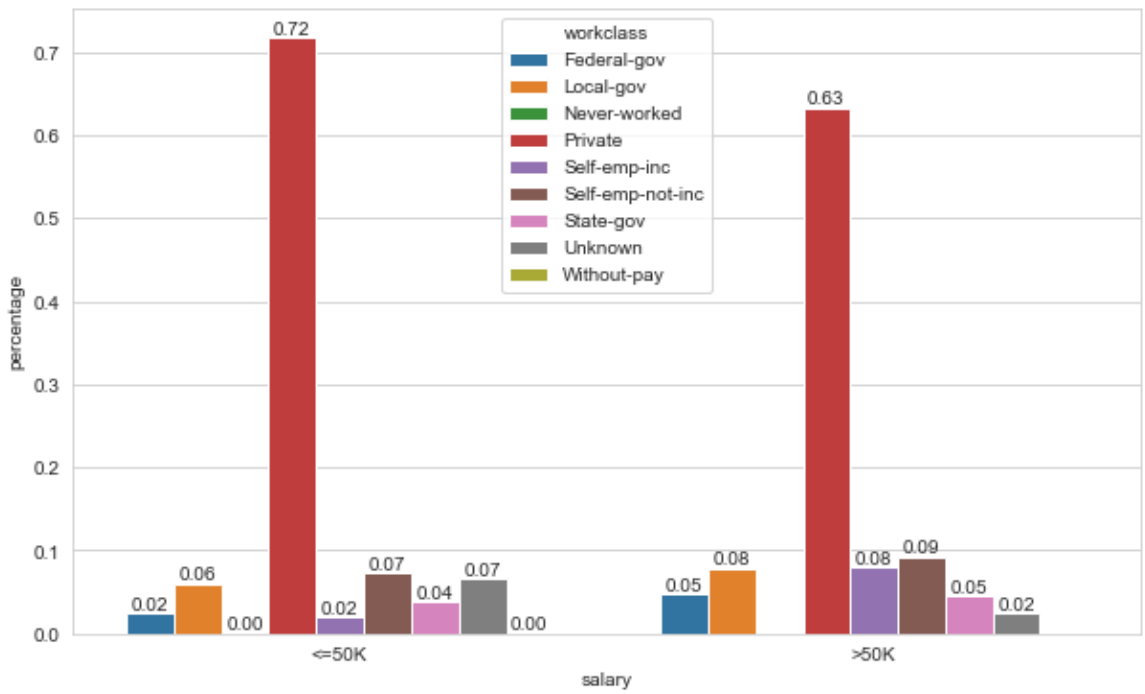
	salary	workclass	percentage
5	<=50K	Federal-gov	0.024
3	<=50K	Local-gov	0.060
8	<=50K	Never-worked	0.000
0	<=50K	Private	0.717
6	<=50K	Self-emp-inc	0.020
1	<=50K	Self-emp-not-inc	0.074
4	<=50K	State-gov	0.038
2	<=50K	Unknown	0.067
7	<=50K	Without-pay	0.001
13	>50K	Federal-gov	0.047
12	>50K	Local-gov	0.079
9	>50K	Private	0.633
11	>50K	Self-emp-inc	0.079
10	>50K	Self-emp-not-inc	0.092
14	>50K	State-gov	0.045
15	>50K	Unknown	0.024

Desired Output:

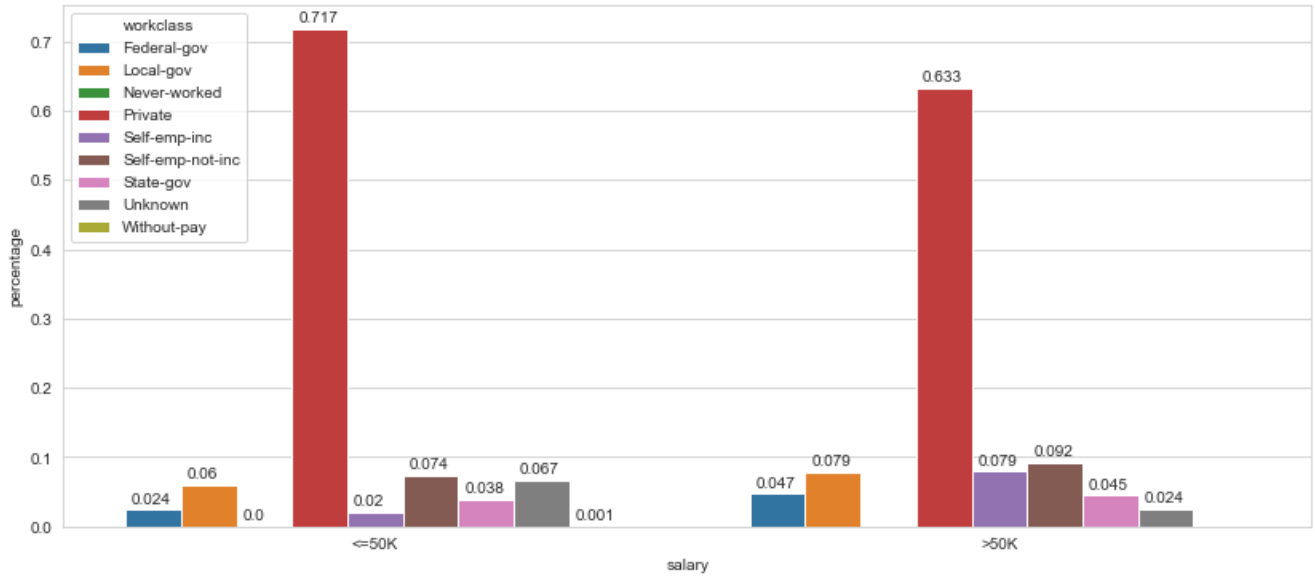
	salary	workclass	percentage
5	<=50K	Federal-gov	0.024
3	<=50K	Local-gov	0.060
8	<=50K	Never-worked	0.000
0	<=50K	Private	0.717
6	<=50K	Self-emp-inc	0.020
1	<=50K	Self-emp-not-inc	0.074
4	<=50K	State-gov	0.038
2	<=50K	Unknown	0.067
7	<=50K	Without-pay	0.001
13	>50K	Federal-gov	0.047
12	>50K	Local-gov	0.079
9	>50K	Private	0.633
11	>50K	Self-emp-inc	0.079
10	>50K	Self-emp-not-inc	0.092
14	>50K	State-gov	0.045
15	>50K	Unknown	0.024

In [1136]:

```
# Your Code is Here
df_temp = df.groupby('salary').workclass.value_counts(dropna=False, normalize=True).rename('percentage').reset_index().sort_values(['salary','workclass'])
g= sns.barplot(x='salary', y='percentage', hue='workclass', data=df_temp)
for x in g.containers:
    plt.bar_label(x, fmt='%.2f');
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : When the workclass variable is examined in our dataset, Private value seen most.
In the self-emp-inc workclass, the rate of those who earn more than 50K is the highest.
Those working at federal-gov earn proportionally more money than those working at local-gov and state-gov.
Private workclass constitutes 63 percent of those who earn over 50K. Private workclass constitutes 72 percent of those who earn less than 50K.

occupation

Check the count of person in each categories and visualize it with countplot

In [1137]:

```
# Your Code is Here
df.occupation.value_counts(dropna=False)
```

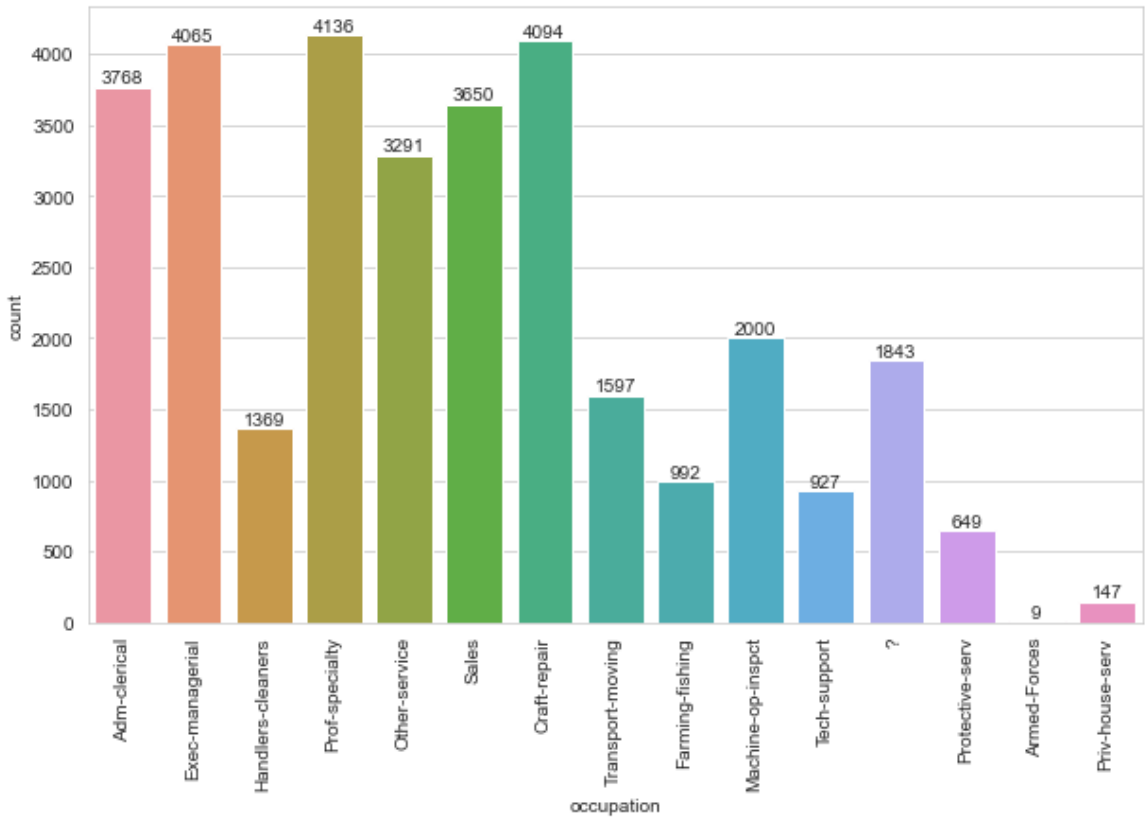
Out[1137]:

```
Prof-specialty       4136
Craft-repair         4094
Exec-managerial      4065
Adm-clerical         3768
Sales                3650
Other-service        3291
Machine-op-inspct    2000
?                    1843
Transport-moving     1597
Handlers-cleaners    1369
Farming-fishing       992
Tech-support          927
Protective-serv       649
Priv-house-serv       147
Armed-Forces            9
Name: occupation, dtype: int64
```

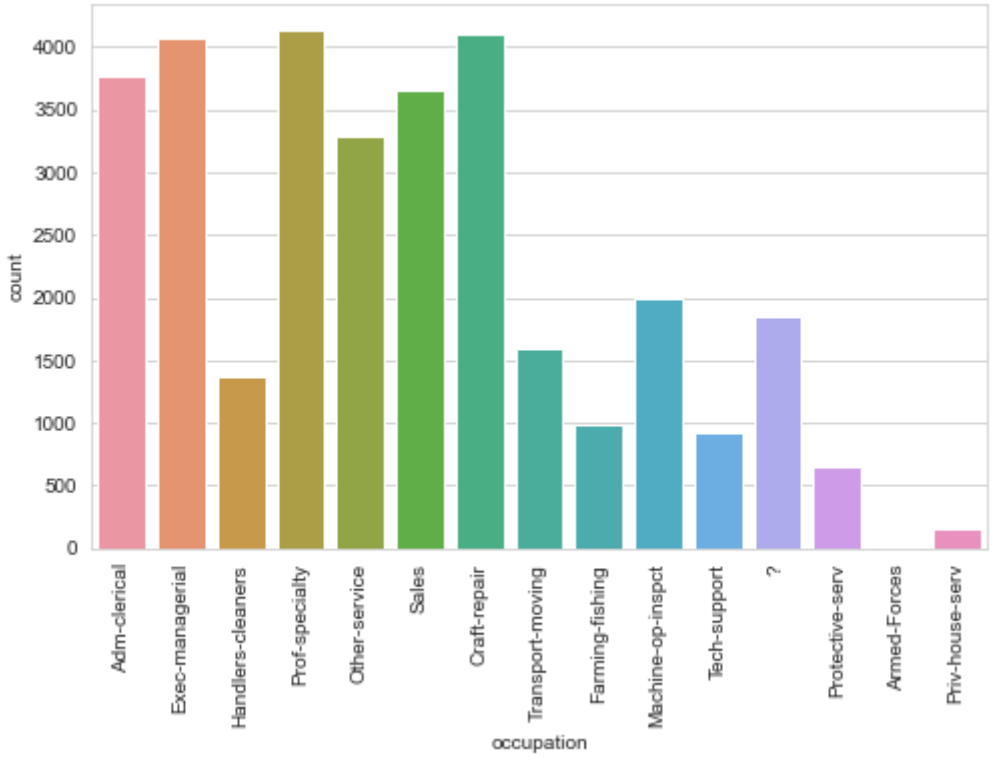
Desired Output: Prof-specialty 4136 Craft-repair 4094 Exec-managerial 4065 Adm-clerical 3768 Sales 3650 Other-service 3291 Machine-op-inspct 2000 ? 1843 Transport-moving 1597 Handlers-cleaners 1369 Farming-fishing 992 Tech-support 927 Protective-serv 649 Priv-house-serv 147 Armed-Forces 9 Name: occupation, dtype: int64

In [1138]:

```
# Your Code is Here
g = sns.countplot(x='occupation', data=df)
plt.xticks(rotation= 90)
plt.bar_label(g.containers[0]);
```



Desired Output:



Replace the value "?" to the value "Unknown"

In [1139]:

```
# Replace "?" values with "Unknown"

# Your Code is Here
df.occupation = df.occupation.replace({'?':'Unknown'})
```

Check the count of person in each "salary" levels by occupation groups and visualize it with countplot

In [1140]:

```
# Your Code is Here
df.groupby('occupation').salary.value_counts(dropna=False)
```

Out[1140]:

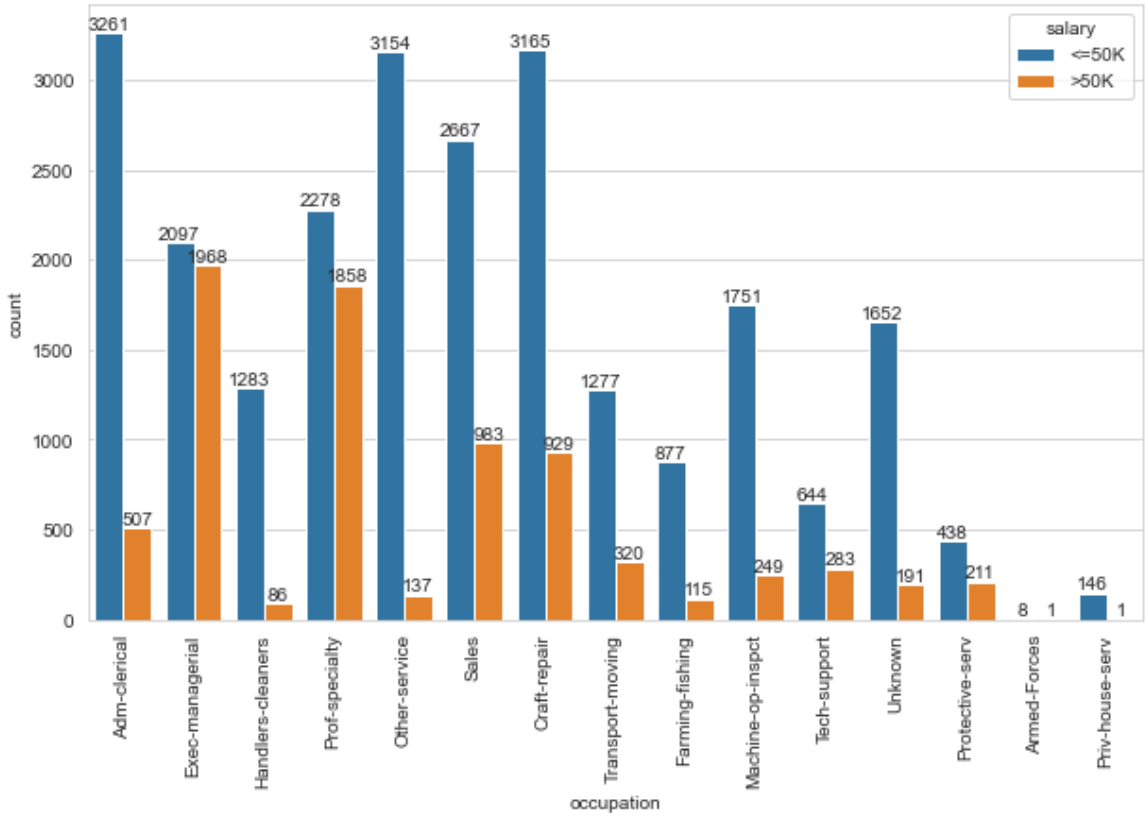
```
occupation         salary
Adm-clerical       <=50K     3261
                   >50K       507
Armed-Forces       <=50K        8
                   >50K         1
Craft-repair       <=50K     3165
                   >50K       929
Exec-managerial    <=50K     2097
                   >50K      1968
Farming-fishing    <=50K      877
                   >50K       115
Handlers-cleaners  <=50K     1283
                   >50K        86
Machine-op-inspct  <=50K     1751
                   >50K       249
Other-service      <=50K     3154
                   >50K       137
Priv-house-serv    <=50K      146
                   >50K         1
Prof-specialty     <=50K     2278
                   >50K      1858
Protective-serv    <=50K      438
                   >50K       211
Sales              <=50K     2667
                   >50K       983
Tech-support       <=50K      644
                   >50K       283
Transport-moving   <=50K     1277
                   >50K       320
Unknown            <=50K     1652
                   >50K       191
Name: salary, dtype: int64
```

Desired Output: occupation salary Adm-clerical <=50K 3261 >50K 507 Armed-Forces <=50K 8 >50K 1 Craft-repair <=50K 3165 >50K 929 Exec-managerial <=50K 2097 >50K 1968 Farming-fishing <=50K 877 >50K 115 Handlers-cleaners <=50K 1283 >50K 86 Machine-op-inspct <=50K 1751 >50K 249 Other-service <=50K 3154 >50K 137 Priv-house-serv <=50K 146 >50K 1 Prof-specialty <=50K 2278 >50K 1858 Protective-serv <=50K 438 >50K 211 Sales <=50K 2667 >50K 983 Tech-support <=50K 644 >50K 283 Transport-moving <=50K 1277 >50K

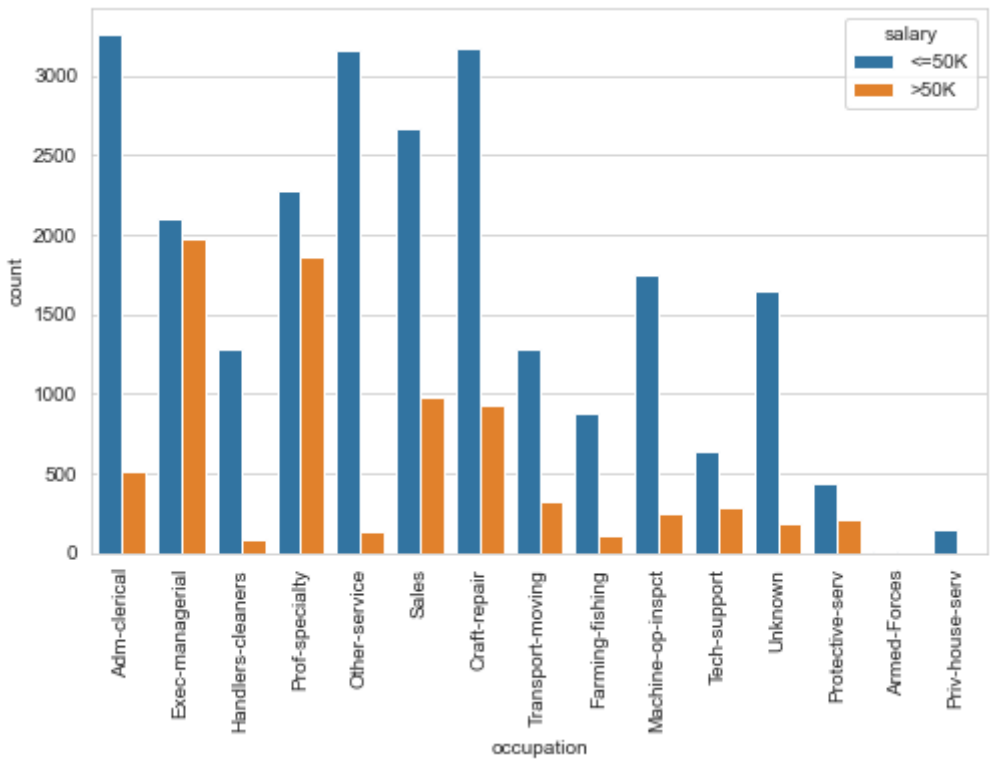
320 Unknown <=50K 1652 >50K 191 Name: salary, dtype: int64

In [1141]:

```
# Your Code is Here
g= sns.countplot(x='occupation', data=df,hue='salary')
plt.xticks(rotation= 90)
plt.bar_label(g.containers[0])
plt.bar_label(g.containers[1]);
```



Desired Output:



Check the percentage distribution of person in each "salary" levels by each occupation groups and visualize it with bar plot

In [1142]:

```
# Your Code is Here

df_temp = df.groupby('occupation').salary.value_counts(dropna=False, normalize = True)
df_temp
```

Out[1142]:

```
occupation         salary
Adm-clerical       <=50K    0.865
                   >50K     0.135
Armed-Forces       <=50K    0.889
                   >50K     0.111
Craft-repair       <=50K    0.773
                   >50K     0.227
Exec-managerial    <=50K    0.516
                   >50K     0.484
Farming-fishing    <=50K    0.884
                   >50K     0.116
Handlers-cleaners  <=50K    0.937
                   >50K     0.063
Machine-op-inspct  <=50K    0.875
                   >50K     0.124
Other-service      <=50K    0.958
                   >50K     0.042
Priv-house-serv    <=50K    0.993
                   >50K     0.007
Prof-specialty     <=50K    0.551
                   >50K     0.449
Protective-serv    <=50K    0.675
                   >50K     0.325
Sales              <=50K    0.731
                   >50K     0.269
Tech-support       <=50K    0.695
                   >50K     0.305
Transport-moving   <=50K    0.800
                   >50K     0.200
Unknown            <=50K    0.896
                   >50K     0.104
Name: salary, dtype: float64
```

Desired Output: occupation salary Adm-clerical <=50K 0.865 >50K 0.135 Armed-Forces <=50K 0.889 >50K 0.111 Craft-repair <=50K 0.773 >50K 0.227 Exec-managerial <=50K 0.516 >50K 0.484 Farming-fishing <=50K 0.884 >50K 0.116 Handlers-cleaners <=50K 0.937 >50K 0.063 Machine-op-inspct <=50K 0.875 >50K 0.124 Other-service <=50K 0.958 >50K 0.042 Priv-house-serv <=50K 0.993 >50K 0.007 Prof-specialty <=50K 0.551 >50K 0.449 Protective-serv <=50K 0.675 >50K 0.325 Sales <=50K 0.731 >50K 0.269 Tech-support <=50K 0.695 >50K 0.305 Transport-moving <=50K 0.800 >50K 0.200 Unknown <=50K 0.896 >50K 0.104 Name: salary, dtype:

float64

In [1143]:

```
# Your Code is Here
df_temp.rename('percentage').reset_index()
```

Out[1143]:

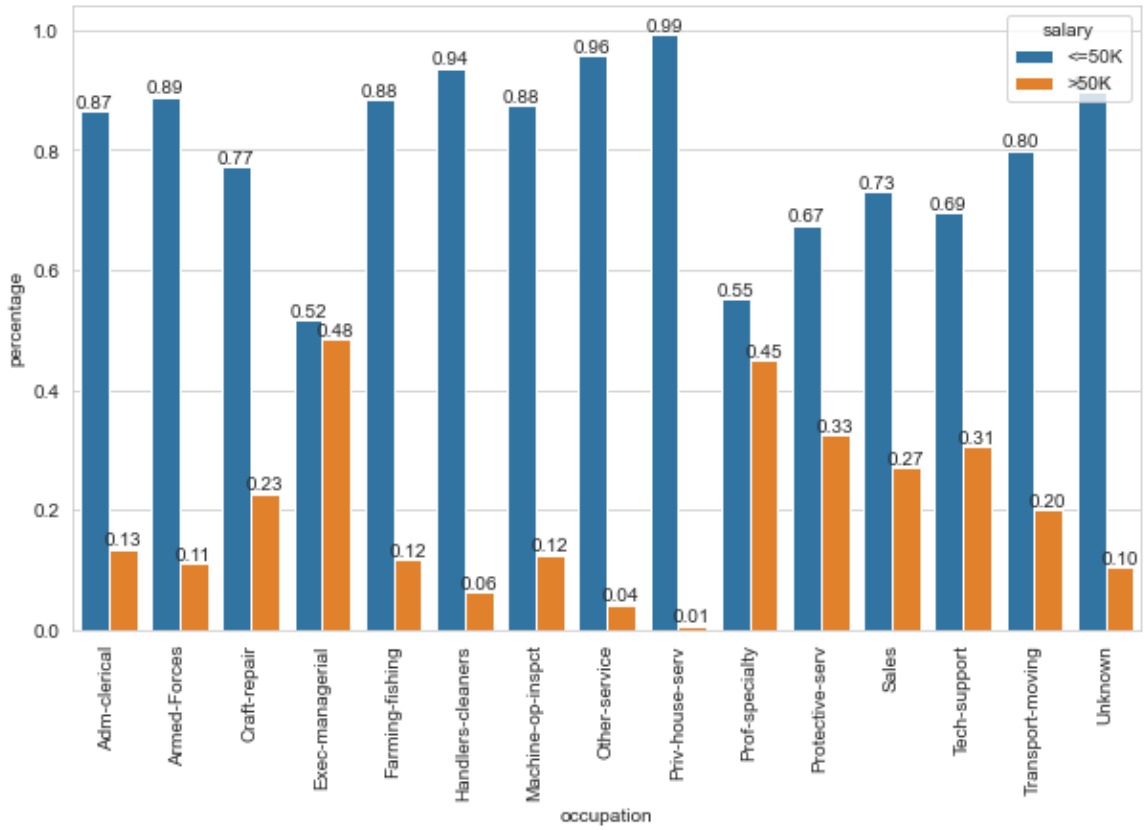
	occupation	salary	percentage
0	Adm-clerical	<=50K	0.865
1	Adm-clerical	>50K	0.135
2	Armed-Forces	<=50K	0.889
3	Armed-Forces	>50K	0.111
4	Craft-repair	<=50K	0.773
5	Craft-repair	>50K	0.227
6	Exec-managerial	<=50K	0.516
7	Exec-managerial	>50K	0.484
8	Farming-fishing	<=50K	0.884
9	Farming-fishing	>50K	0.116
10	Handlers-cleaners	<=50K	0.937
11	Handlers-cleaners	>50K	0.063
12	Machine-op-inspct	<=50K	0.875
13	Machine-op-inspct	>50K	0.124
14	Other-service	<=50K	0.958
15	Other-service	>50K	0.042
16	Priv-house-serv	<=50K	0.993
17	Priv-house-serv	>50K	0.007
18	Prof-specialty	<=50K	0.551
19	Prof-specialty	>50K	0.449
20	Protective-serv	<=50K	0.675
21	Protective-serv	>50K	0.325
22	Sales	<=50K	0.731
23	Sales	>50K	0.269
24	Tech-support	<=50K	0.695
25	Tech-support	>50K	0.305
26	Transport-moving	<=50K	0.800
27	Transport-moving	>50K	0.200
28	Unknown	<=50K	0.896
29	Unknown	>50K	0.104

Desired Output:

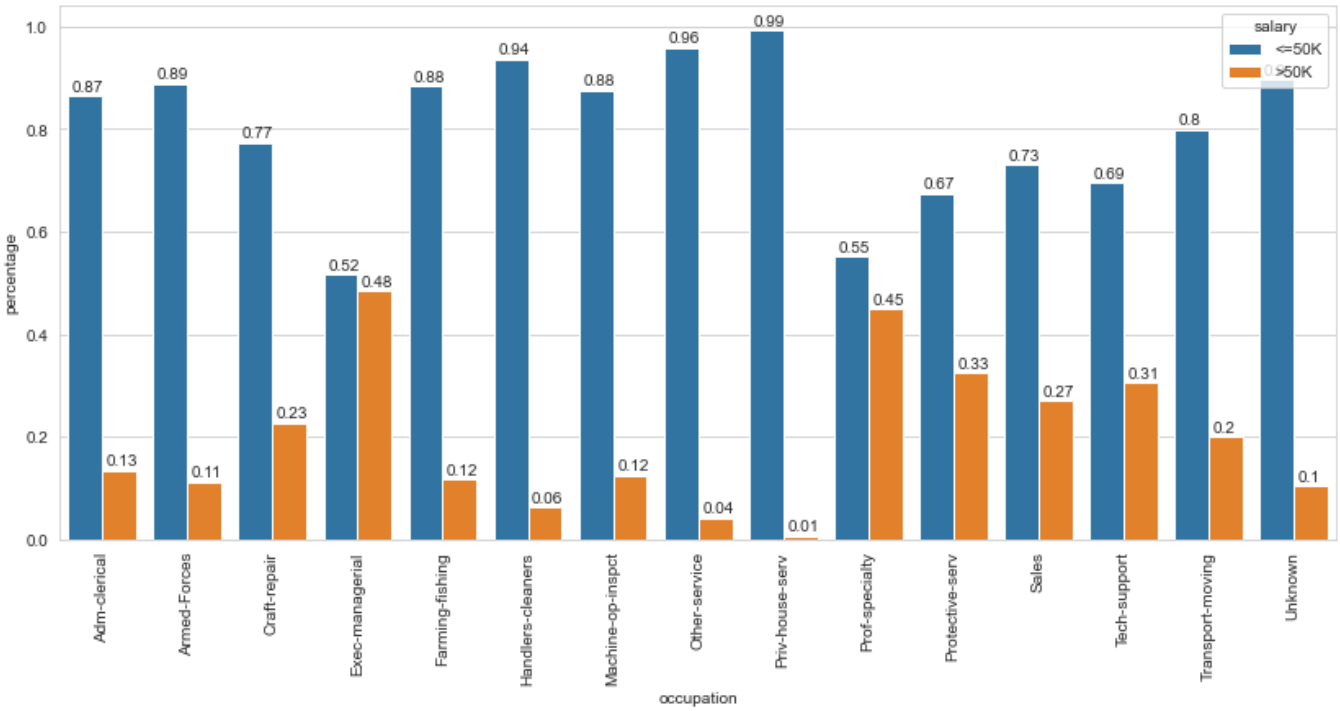
	occupation	salary	percentage
0	Adm-clerical	<=50K	0.865
1	Adm-clerical	>50K	0.135
2	Armed-Forces	<=50K	0.889
3	Armed-Forces	>50K	0.111
4	Craft-repair	<=50K	0.773
5	Craft-repair	>50K	0.227
6	Exec-managerial	<=50K	0.516
7	Exec-managerial	>50K	0.484
8	Farming-fishing	<=50K	0.884
9	Farming-fishing	>50K	0.116
10	Handlers-cleaners	<=50K	0.937
11	Handlers-cleaners	>50K	0.063
12	Machine-op-inspct	<=50K	0.875
13	Machine-op-inspct	>50K	0.124
14	Other-service	<=50K	0.958
15	Other-service	>50K	0.042
16	Priv-house-serv	<=50K	0.993
17	Priv-house-serv	>50K	0.007
18	Prof-specialty	<=50K	0.551
19	Prof-specialty	>50K	0.449
20	Protective-serv	<=50K	0.675
21	Protective-serv	>50K	0.325
22	Sales	<=50K	0.731
23	Sales	>50K	0.269
24	Tech-support	<=50K	0.695
25	Tech-support	>50K	0.305
26	Transport-moving	<=50K	0.800
27	Transport-moving	>50K	0.200
28	Unknown	<=50K	0.896
29	Unknown	>50K	0.104

In [1144]:

```
# Your Code is Here
g = sns.barplot(x='occupation', y='percentage', hue='salary', data=df_temp.renam
e('percentage').reset_index())
plt.xticks(rotation= 90)
for x in g.containers:
    plt.bar_label(x, fmt='%0.2f');
```



Desired Output:



Check the count of person in each occupation groups by "salary" levels and visualize it with countplot

In [1145]:

```
# Your Code is Here

df.groupby('salary').occupation.value_counts(dropna=False)
```

Out[1145]:

```
salary  occupation
<=50K   Adm-clerical         3261
        Craft-repair         3165
        Other-service        3154
        Sales                2667
        Prof-specialty       2278
        Exec-managerial      2097
        Machine-op-inspct    1751
        Unknown              1652
        Handlers-cleaners    1283
        Transport-moving     1277
        Farming-fishing       877
        Tech-support          644
        Protective-serv       438
        Priv-house-serv       146
        Armed-Forces            8
>50K    Exec-managerial      1968
        Prof-specialty       1858
        Sales                 983
        Craft-repair          929
        Adm-clerical          507
        Transport-moving      320
        Tech-support          283
        Machine-op-inspct     249
        Protective-serv       211
        Unknown               191
        Other-service         137
        Farming-fishing       115
        Handlers-cleaners      86
        Armed-Forces            1
        Priv-house-serv         1
Name: occupation, dtype: int64
```

Desired Output: salary occupation <=50K Adm-clerical 3261 Craft-repair 3165 Other-service 3154 Sales 2667 Prof-specialty 2278 Exec-managerial 2097 Machine-op-inspct 1751 Unknown 1652 Handlers-cleaners 1283 Transport-moving 1277 Farming-fishing 877 Tech-support 644 Protective-serv 438 Priv-house-serv 146 Armed-Forces 8 >50K Exec-managerial 1968 Prof-specialty 1858 Sales 983 Craft-repair 929 Adm-clerical 507 Transport-moving 320 Tech-support 283 Machine-op-inspct 249 Protective-serv 211 Unknown 191 Other-service 137 Farming-fishing 115 Handlers-cleaners 86 Armed-Forces 1 Priv-house-serv 1 Name: occupation,

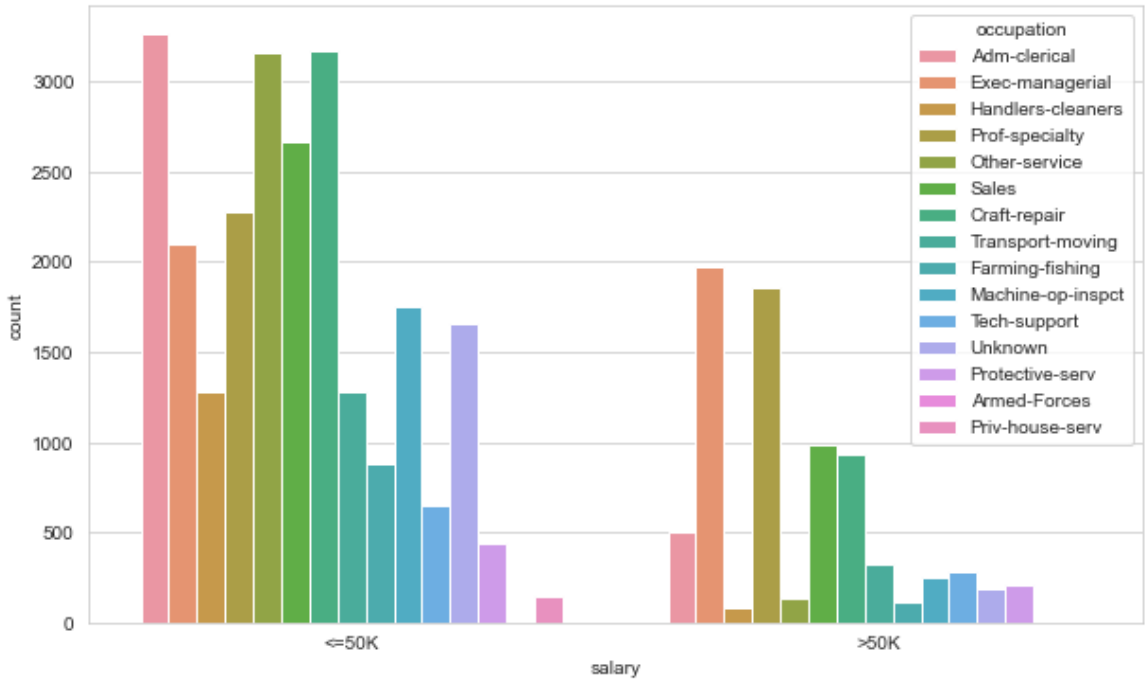
dtype: int64

In [1146]:

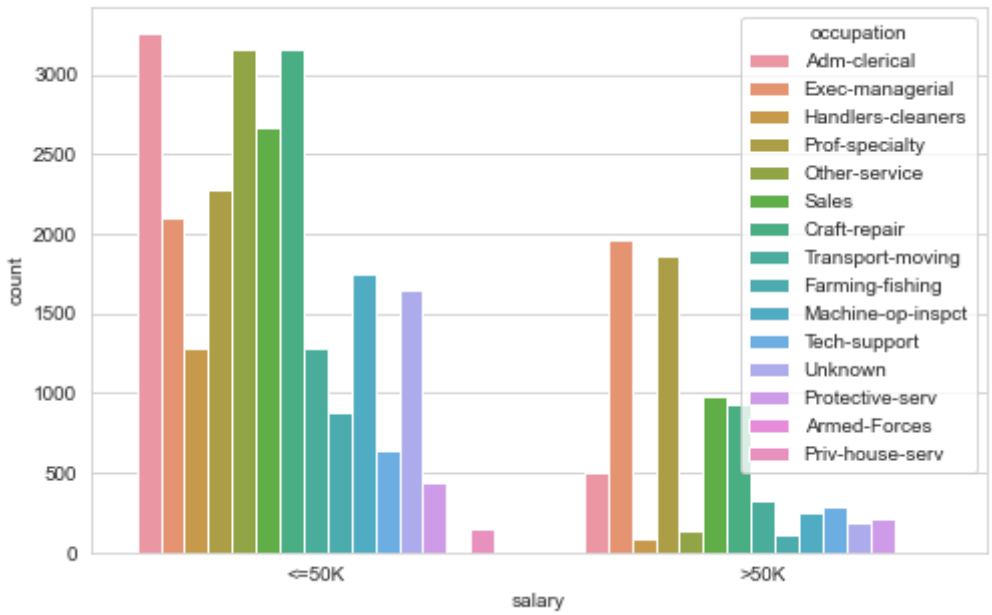
```
# Your Code is Here

sns.countplot(x='salary', hue='occupation', data=df)
```

Out[1146]:

```
<AxesSubplot:xlabel='salary', ylabel='count'>
```



Desired Output:



Check the the percentage distribution of person in each occupation groups by "salary" levels and visualize it with bar plot

In [1147]:

```
# Your Code is Here
df.groupby('salary').occupation.value_counts(dropna=False, normalize = True)
```

Out[1147]:

```
salary  occupation
<=50K   Adm-clerical        0.132
        Craft-repair        0.128
        Other-service       0.128
        Sales               0.108
        Prof-specialty      0.092
        Exec-managerial     0.085
        Machine-op-inspct   0.071
        Unknown             0.067
        Handlers-cleaners   0.052
        Transport-moving    0.052
        Farming-fishing     0.036
        Tech-support        0.026
        Protective-serv     0.018
        Priv-house-serv     0.006
        Armed-Forces        0.000
>50K    Exec-managerial     0.251
        Prof-specialty      0.237
        Sales               0.125
        Craft-repair        0.119
        Adm-clerical        0.065
        Transport-moving    0.041
        Tech-support        0.036
        Machine-op-inspct   0.032
        Protective-serv     0.027
        Unknown             0.024
        Other-service       0.017
        Farming-fishing     0.015
        Handlers-cleaners   0.011
        Armed-Forces        0.000
        Priv-house-serv     0.000
Name: occupation, dtype: float64
```

Desired Output: salary occupation <=50K Adm-clerical 0.132 Craft-repair 0.128 Other-service 0.128 Sales 0.108 Prof-specialty 0.092 Exec-managerial 0.085 Machine-op-inspct 0.071 Unknown 0.067 Handlers-cleaners 0.052 Transport-moving 0.052 Farming-fishing 0.036 Tech-support 0.026 Protective-serv 0.018 Priv-house-serv 0.006 Armed-Forces 0.000 >50K Exec-managerial 0.251 Prof-specialty 0.237 Sales 0.125 Craft-repair 0.119 Adm-clerical 0.065 Transport-moving 0.041 Tech-support 0.036 Machine-op-inspct 0.032 Protective-serv 0.027 Unknown 0.024 Other-service 0.017 Farming-fishing 0.015 Handlers-cleaners 0.011 Armed-Forces 0.000 Priv-

house-serv 0.000 Name: occupation, dtype: float64

In [1148]:

```
# Your Code is Here
df.groupby('salary').occupation.value_counts(dropna=False, normalize = True).ren
ame('percentage').reset_index()
```

Out[1148]:

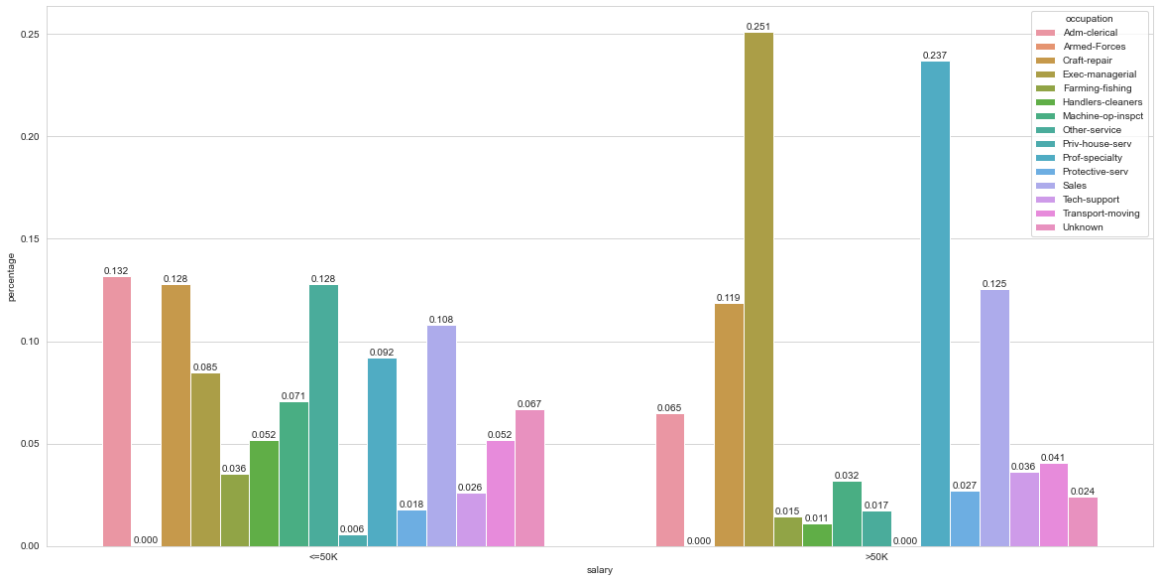
	salary	occupation	percentage
0	<=50K	Adm-clerical	0.132
1	<=50K	Craft-repair	0.128
2	<=50K	Other-service	0.128
3	<=50K	Sales	0.108
4	<=50K	Prof-specialty	0.092
5	<=50K	Exec-managerial	0.085
6	<=50K	Machine-op-inspct	0.071
7	<=50K	Unknown	0.067
8	<=50K	Handlers-cleaners	0.052
9	<=50K	Transport-moving	0.052
10	<=50K	Farming-fishing	0.036
11	<=50K	Tech-support	0.026
12	<=50K	Protective-serv	0.018
13	<=50K	Priv-house-serv	0.006
14	<=50K	Armed-Forces	0.000
15	>50K	Exec-managerial	0.251
16	>50K	Prof-specialty	0.237
17	>50K	Sales	0.125
18	>50K	Craft-repair	0.119
19	>50K	Adm-clerical	0.065
20	>50K	Transport-moving	0.041
21	>50K	Tech-support	0.036
22	>50K	Machine-op-inspct	0.032
23	>50K	Protective-serv	0.027
24	>50K	Unknown	0.024
25	>50K	Other-service	0.017
26	>50K	Farming-fishing	0.015
27	>50K	Handlers-cleaners	0.011
28	>50K	Armed-Forces	0.000
29	>50K	Priv-house-serv	0.000

Desired Output:

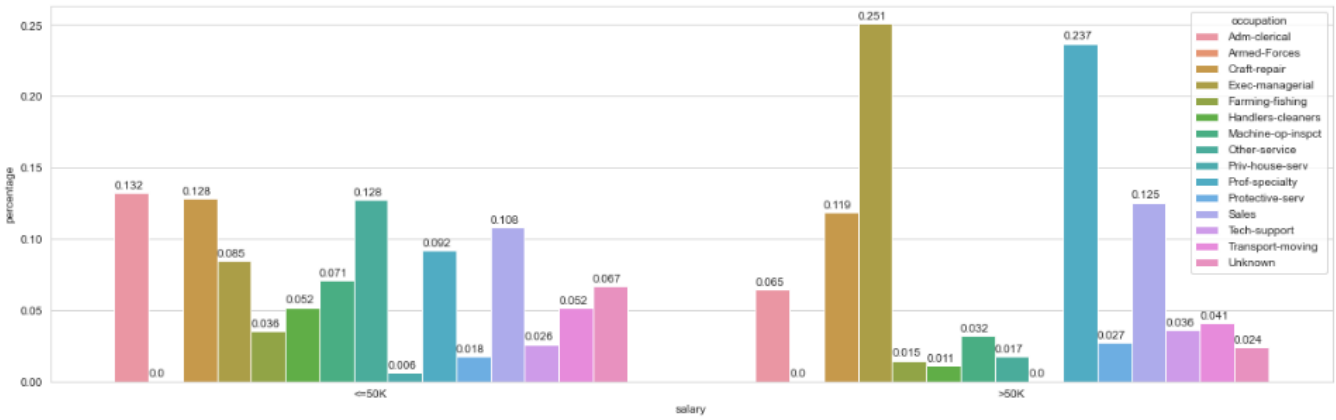
	salary	occupation	percentage
0	<=50K	Adm-clerical	0.132
14	<=50K	Armed-Forces	0.000
1	<=50K	Craft-repair	0.128
5	<=50K	Exec-managerial	0.085
10	<=50K	Farming-fishing	0.036
8	<=50K	Handlers-cleaners	0.052
6	<=50K	Machine-op-inspct	0.071
2	<=50K	Other-service	0.128
13	<=50K	Priv-house-serv	0.006
4	<=50K	Prof-specialty	0.092
12	<=50K	Protective-serv	0.018
3	<=50K	Sales	0.108
11	<=50K	Tech-support	0.026
9	<=50K	Transport-moving	0.052
7	<=50K	Unknown	0.067
19	>50K	Adm-clerical	0.065
28	>50K	Armed-Forces	0.000
18	>50K	Craft-repair	0.119
15	>50K	Exec-managerial	0.251
26	>50K	Farming-fishing	0.015
27	>50K	Handlers-cleaners	0.011
22	>50K	Machine-op-inspct	0.032
25	>50K	Other-service	0.017
29	>50K	Priv-house-serv	0.000
16	>50K	Prof-specialty	0.237
23	>50K	Protective-serv	0.027
17	>50K	Sales	0.125
21	>50K	Tech-support	0.036
20	>50K	Transport-moving	0.041
24	>50K	Unknown	0.024

In [1149]:

```
# Your Code is Here
plt.figure(figsize=(20,10))
df_temp = df.groupby('salary').occupation.value_counts(dropna=False, normalize =
True).rename('percentage').reset_index().sort_values(by=['salary','occupation'])
g=sns.barplot(x='salary',y='percentage', hue='occupation', data=df_temp)
for x in g.containers:
    plt.bar_label(x, fmt='%0.3f');
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : When the occupation feature is examined in the data set, Prof-specialty is seen the most.
When the Prof-specialty value is examined, the rate of those who earn over 50K is the highest compared to the others.(45 percent)
99 percent of Priv-house-serv is making under 50K.
Considering those earning less than 50K, Adm-clerical is the highest with 13 percent.
Examining those earning over 50K, Exec-managerial is the highest with 25 percent.

race

Check the count of person in each categories and visualize it with countplot

In [1150]:

```
# Your Code is Here
df.race.value_counts(dropna=False)
```

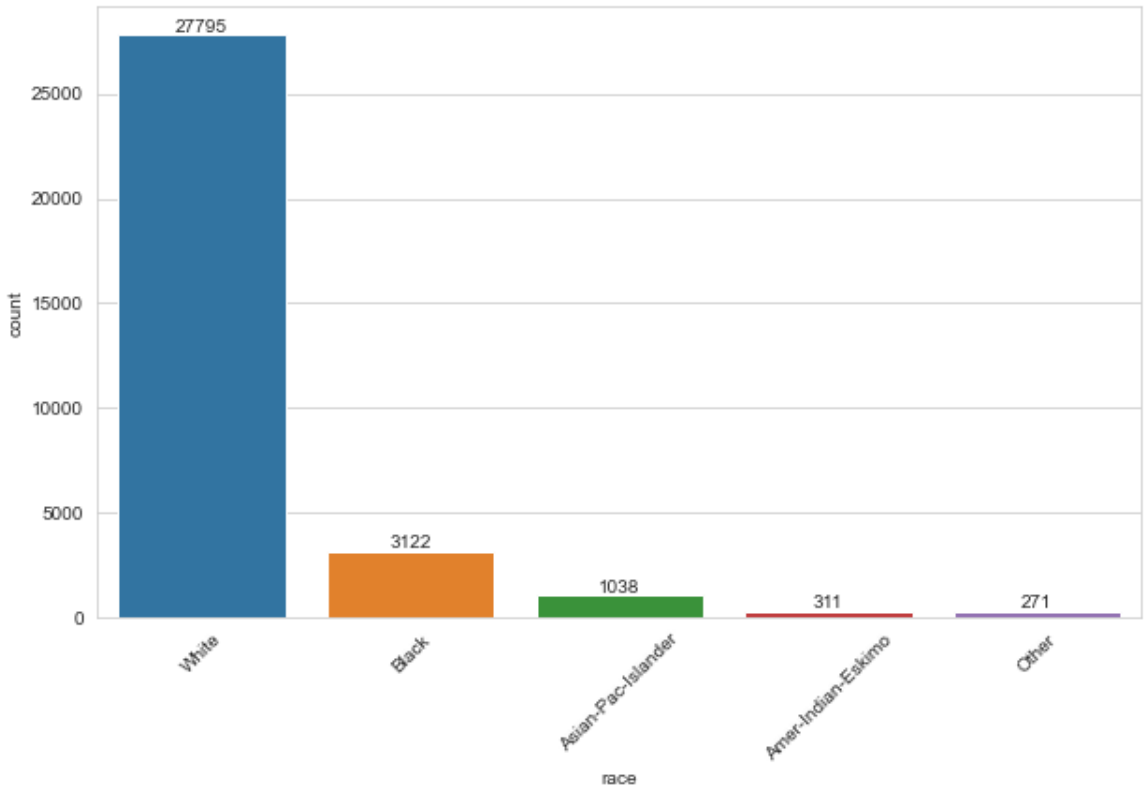
Out[1150]:

```
White                 27795
Black                  3122
Asian-Pac-Islander     1038
Amer-Indian-Eskimo      311
Other                   271
Name: race, dtype: int64
```

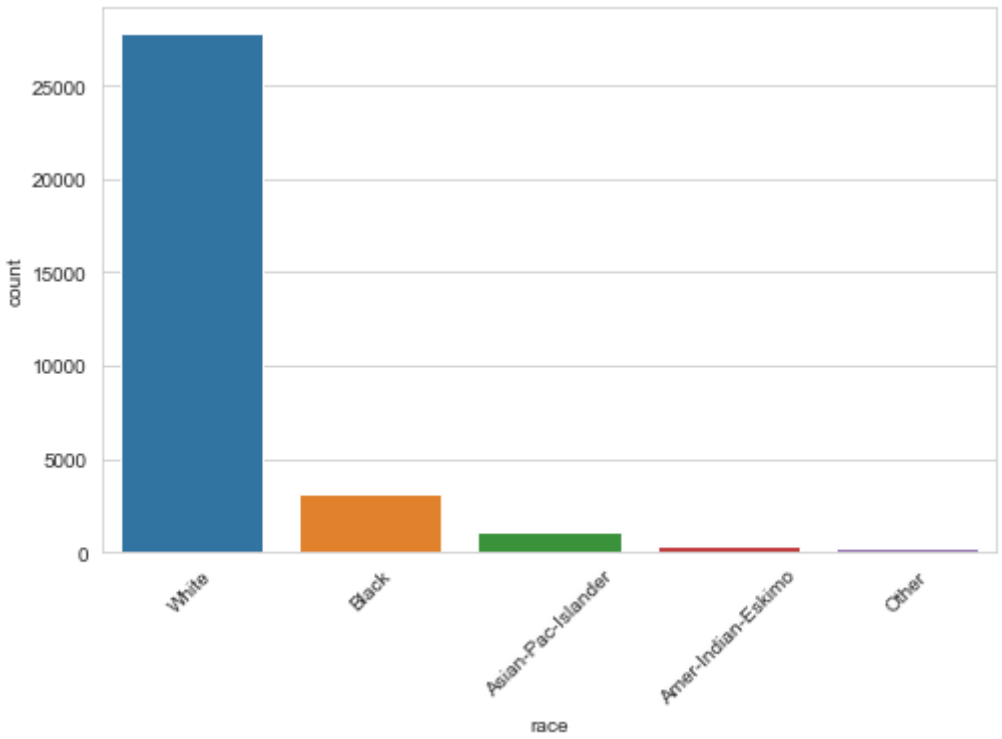
Desired Output: White 27795 Black 3122 Asian-Pac-Islander 1038 Amer-Indian-Eskimo 311 Other 271 Name: race, dtype: int64

In [1151]:

```
# Your Code is Here

g=sns.countplot(x='race', data=df)
plt.xticks(rotation=45)
plt.bar_label(g.containers[0]);
```



Desired Output:



Check the count of person in each "salary" levels by races and visualize it with countplot

In [1152]:

```
# Your Code is Here
df.groupby('race').salary.value_counts(dropna=False)
```

Out[1152]:

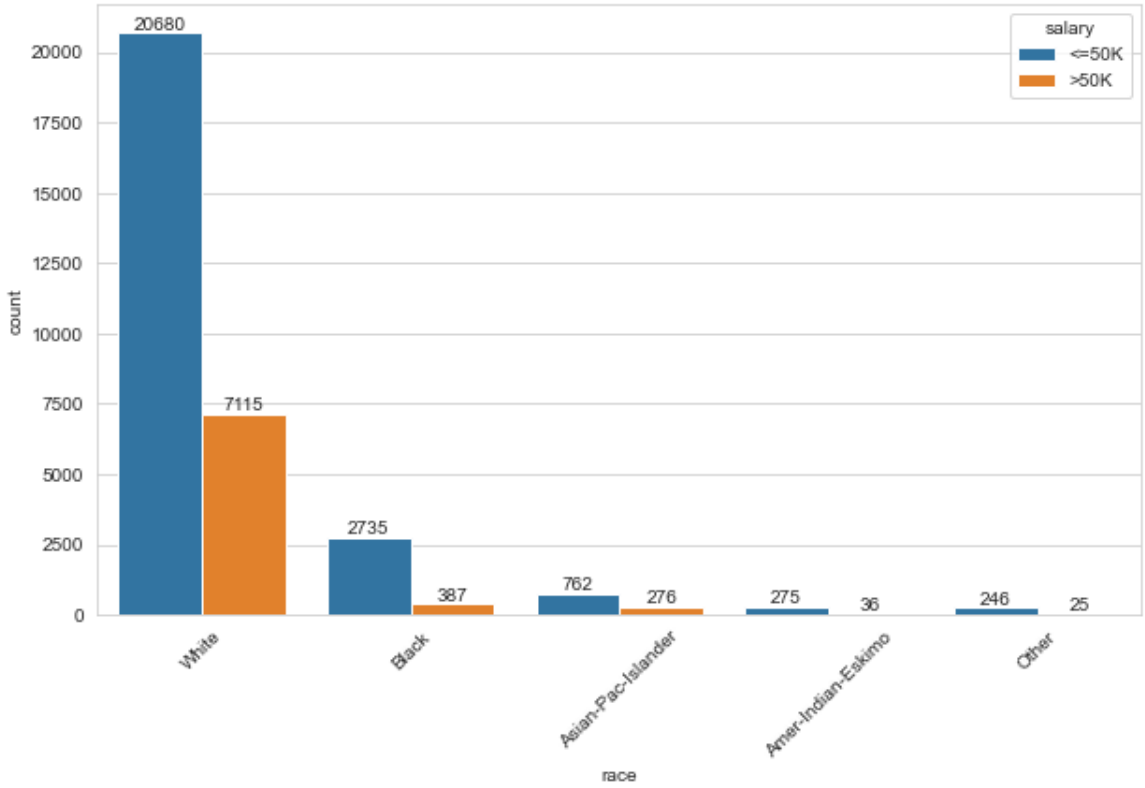
```
race                 salary
Amer-Indian-Eskimo   <=50K       275
                     >50K         36
Asian-Pac-Islander   <=50K       762
                     >50K        276
Black                <=50K      2735
                     >50K        387
Other                <=50K       246
                     >50K         25
White                <=50K     20680
                     >50K       7115
Name: salary, dtype: int64
```

Desired Output: race salary Amer-Indian-Eskimo <=50K 275 >50K 36 Asian-Pac-Islander <=50K 762 >50K 276 Black <=50K 2735 >50K 387 Other <=50K 246 >50K 25 White <=50K 20680 >50K 7115 Name: salary, dtype:

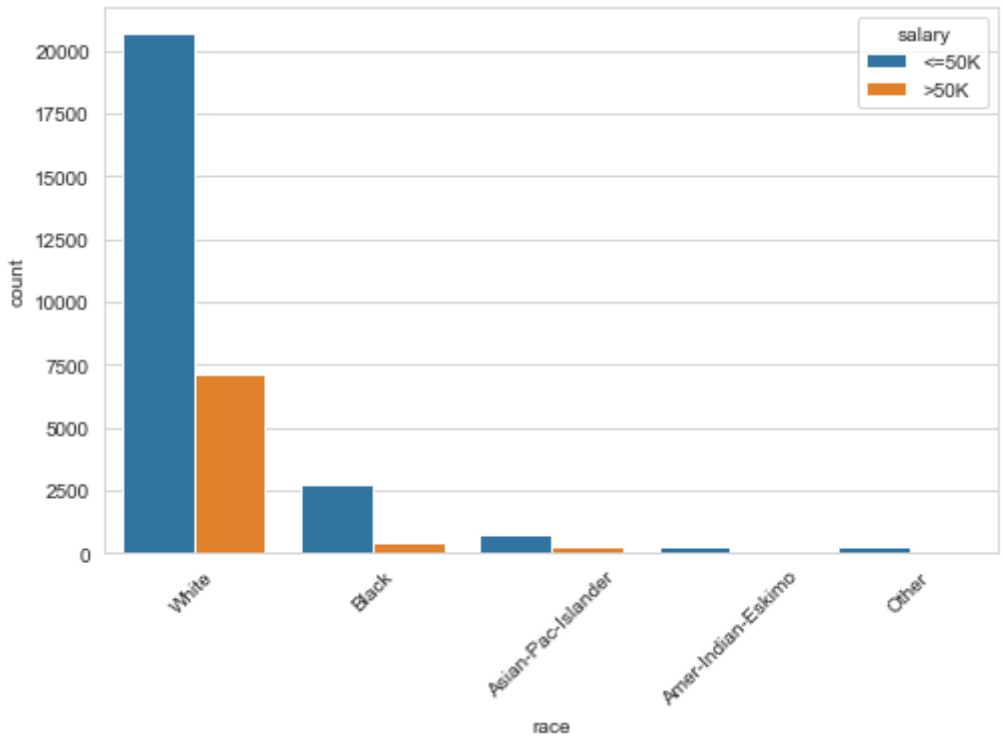
int64

In [1153]:

```
# Your Code is Here
g=sns.countplot(x='race', data=df, hue='salary')
plt.xticks(rotation=45)
plt.bar_label(g.containers[0])
plt.bar_label(g.containers[1]);
```



Desired Output:



Check the percentage distribution of person in each "salary" levels by each races and visualize it with pie plot

In [1154]:

```
# Your Code is Here
df.groupby('race').salary.value_counts(dropna=False, normalize=True)
```

Out[1154]:

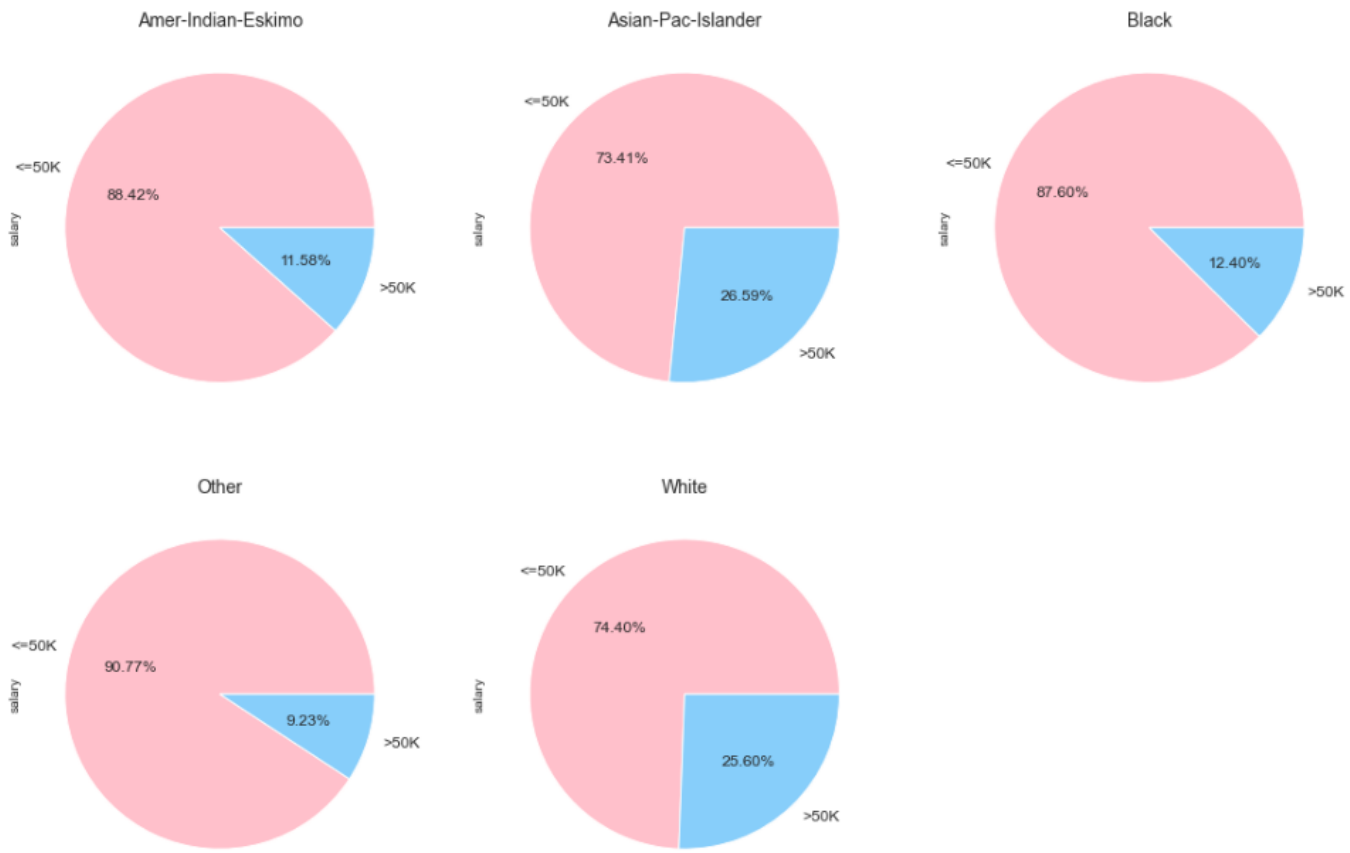
```
race                 salary
Amer-Indian-Eskimo   <=50K     0.884
                     >50K      0.116
Asian-Pac-Islander   <=50K     0.734
                     >50K      0.266
Black                <=50K     0.876
                     >50K      0.124
Other                <=50K     0.908
                     >50K      0.092
White                <=50K     0.744
                     >50K      0.256
Name: salary, dtype: float64
```

Desired Output: race salary Amer-Indian-Eskimo <=50K 0.884 >50K 0.116 Asian-Pac-Islander <=50K 0.734 >50K 0.266 Black <=50K 0.876 >50K 0.124 Other <=50K 0.908 >50K 0.092 White <=50K 0.744 >50K 0.256 Name: salary, dtype: float64

In [1155]:

```
# Your Code is Here
plt.figure(figsize=(20,10))
for count,x in enumerate(sorted(df.race.unique())):
    plt.subplot(2,3,count+ 1)
    df[df.race == x].salary.value_counts(dropna=False).plot.pie(autopct='%0.2f%
%', colors=['pink','deepskyblue'], radius=1, textprops={'fontsize': 14})
    plt.title(x, fontsize =20);
plt.tight_layout()
```

Desired Output:



Check the count of person in each races by "salary" levels and visualize it with countplot

In [1156]:

```
# Your Code is Here
df.groupby('salary').race.value_counts()
```

Out[1156]:

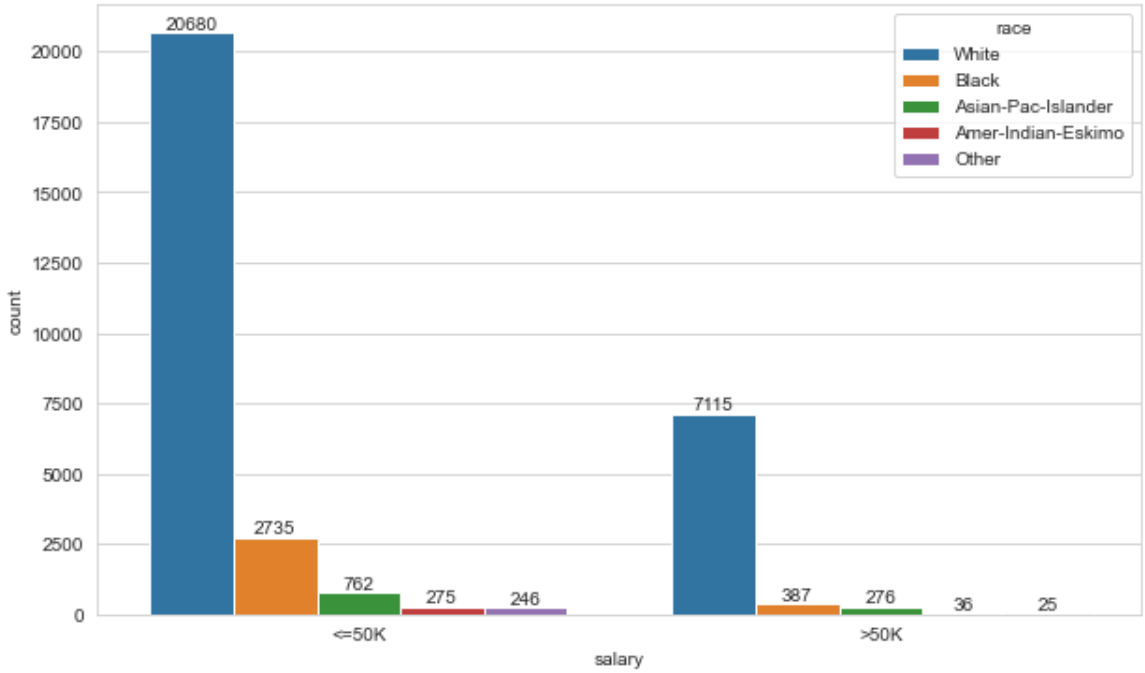
```
salary  race
<=50K   White                20680
        Black                 2735
        Asian-Pac-Islander     762
        Amer-Indian-Eskimo     275
        Other                  246
>50K    White                 7115
        Black                  387
        Asian-Pac-Islander     276
        Amer-Indian-Eskimo      36
        Other                   25
Name: race, dtype: int64
```

Desired Output: salary race <=50K White 20680 Black 2735 Asian-Pac-Islander 762 Amer-Indian-Eskimo 275 Other 246 >50K White 7115 Black 387 Asian-Pac-Islander 276 Amer-Indian-Eskimo 36 Other 25 Name: race,

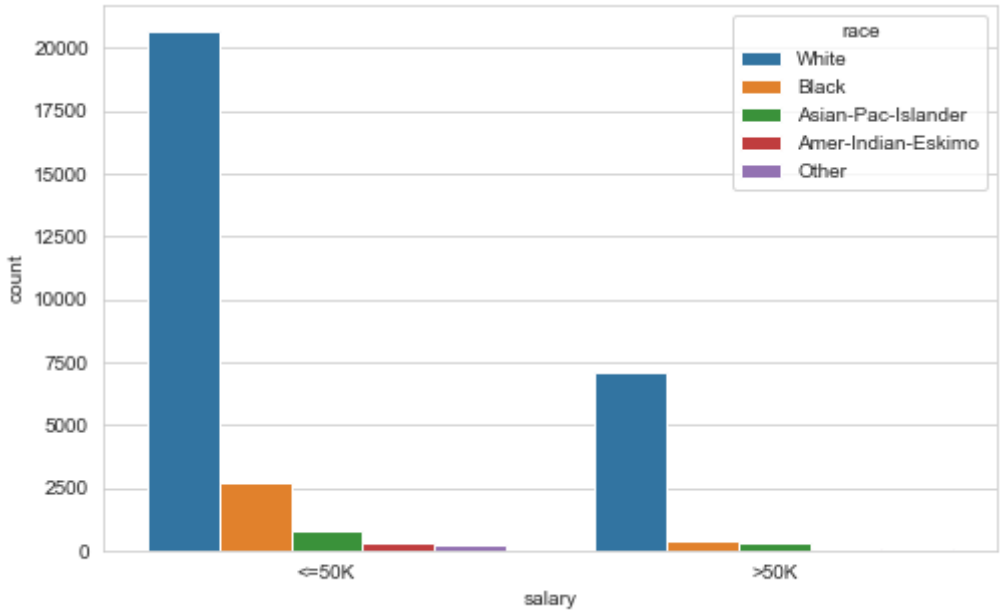
dtype: int64

In [1157]:

```
# Your Code is Here

g = sns.countplot(x='salary', hue='race', data=df)
for x in g.containers:
    plt.bar_label(x);
```



Desired Output:



Check the the percentage distribution of person in each races by "salary" levels and visualize it with bar plot

In [1158]:

```
# Your Code is Here

df.groupby('salary').race.value_counts(dropna=False, normalize=True)
```

Out[1158]:

```
salary  race
<=50K   White                 0.837
        Black                 0.111
        Asian-Pac-Islander    0.031
        Amer-Indian-Eskimo    0.011
        Other                 0.010
>50K    White                 0.908
        Black                 0.049
        Asian-Pac-Islander    0.035
        Amer-Indian-Eskimo    0.005
        Other                 0.003
Name: race, dtype: float64
```

Desired Output: salary race <=50K White 0.837 Black 0.111 Asian-Pac-Islander 0.031 Amer-Indian-Eskimo 0.011 Other 0.010 >50K White 0.908 Black 0.049 Asian-Pac-Islander 0.035 Amer-Indian-Eskimo 0.005 Other 0.003 Name: race, dtype: float64

In [1159]:

```
# Your Code is Here
df.groupby('salary').race.value_counts(dropna=False, normalize=True).rename('percentage').reset_index().sort_values(['salary','race'])
```

Out[1159]:

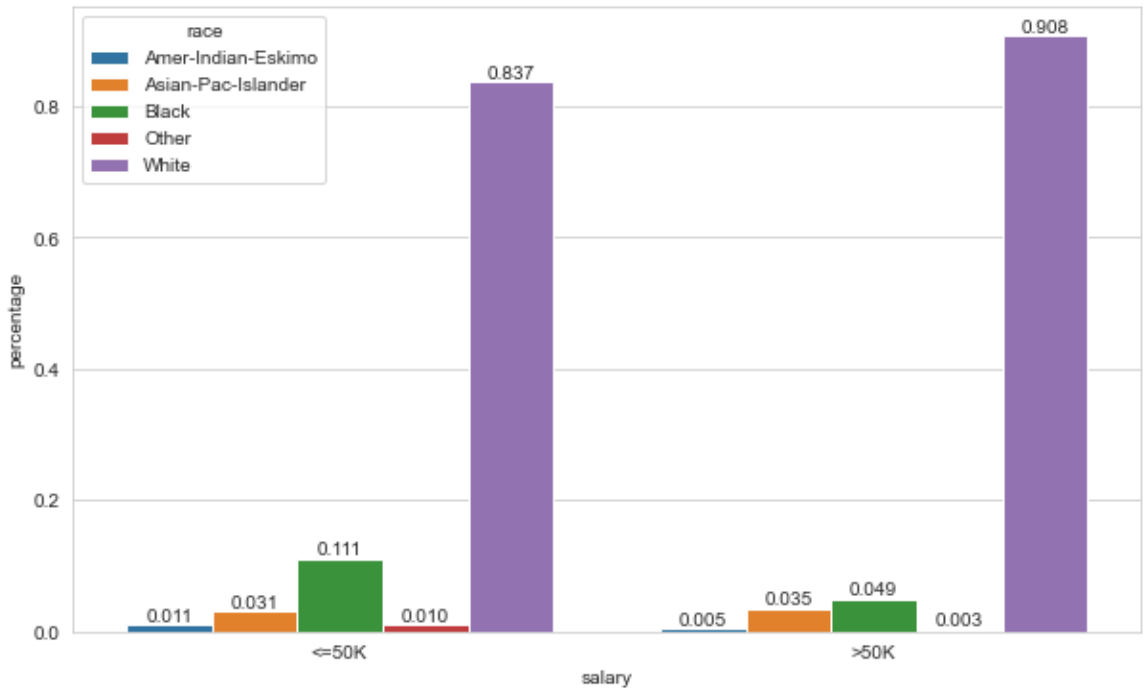
	salary	race	percentage
3	<=50K	Amer-Indian-Eskimo	0.011
2	<=50K	Asian-Pac-Islander	0.031
1	<=50K	Black	0.111
4	<=50K	Other	0.010
0	<=50K	White	0.837
8	>50K	Amer-Indian-Eskimo	0.005
7	>50K	Asian-Pac-Islander	0.035
6	>50K	Black	0.049
9	>50K	Other	0.003
5	>50K	White	0.908

Desired Output:

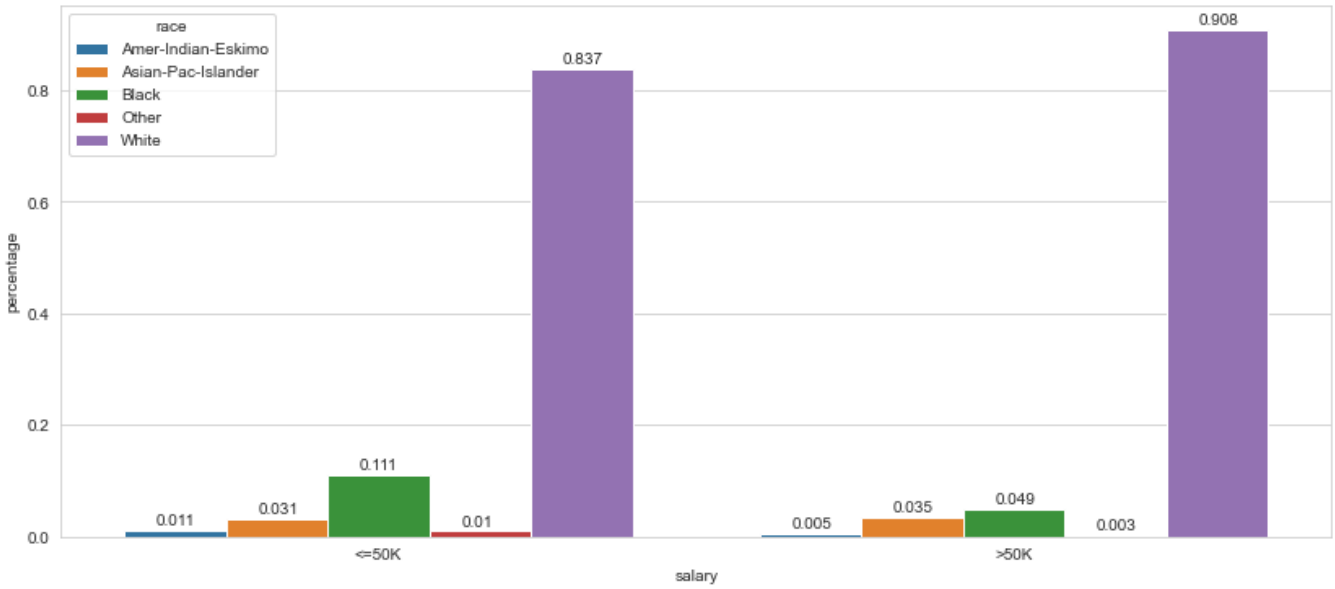
	salary	race	percentage
3	<=50K	Amer-Indian-Eskimo	0.011
2	<=50K	Asian-Pac-Islander	0.031
1	<=50K	Black	0.111
4	<=50K	Other	0.010
0	<=50K	White	0.837
8	>50K	Amer-Indian-Eskimo	0.005
7	>50K	Asian-Pac-Islander	0.035
6	>50K	Black	0.049
9	>50K	Other	0.003
5	>50K	White	0.908

In [1160]:

```
# Your Code is Here
df_temp = df.groupby('salary').race.value_counts(dropna=False, normalize=True).r
ename('percentage').reset_index().sort_values(['salary','race'])
g = sns.barplot(x='salary', y='percentage',hue='race', data=df_temp)
for x in g.containers:
    plt.bar_label(x, fmt='%0.3f');
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : When the race feature is examined in the data set, white value is seen the most.
The vast majority of Amer-Indian-Eskimos earn under 50K (88 percent).
One out of every 4 Whites and 1 out of every 4 Asian-Pac-Islanders earn over 50K.
Whites make up the majority of those who earn under 50K.
Whites make up the majority of those who earn over 50K.

gender

Check the count of person in each gender and visualize it with countplot

In [1161]:

```
# Your Code is Here

df.sex.value_counts()
```

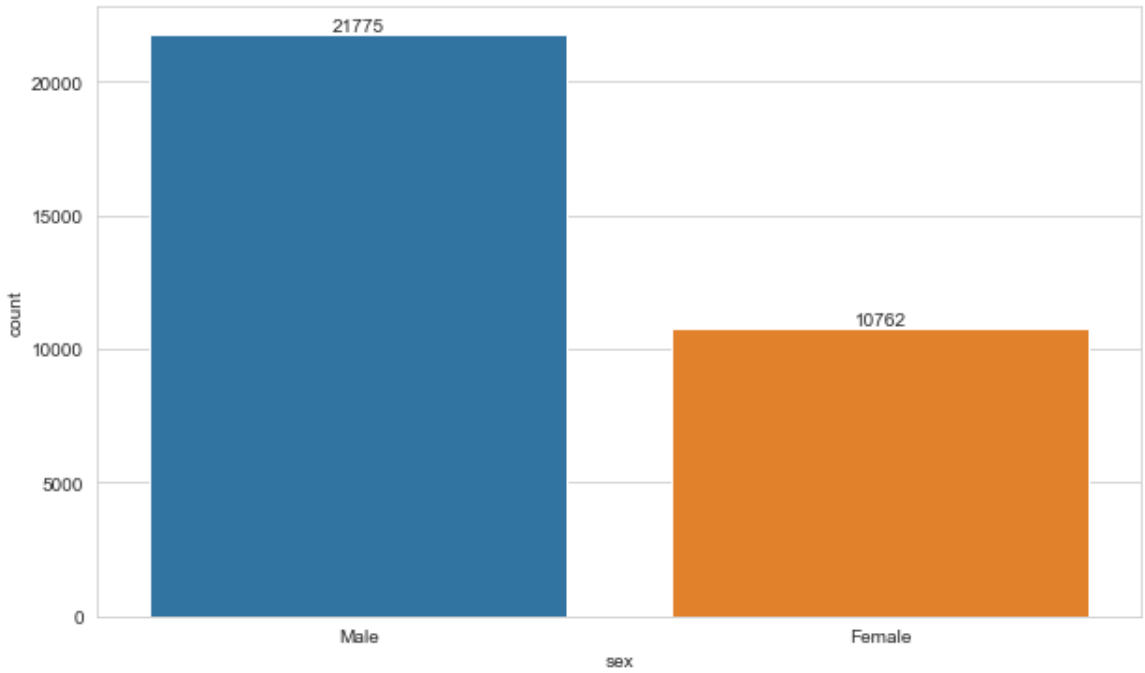
Out[1161]:

```
Male      21775
Female    10762
Name: sex, dtype: int64
```

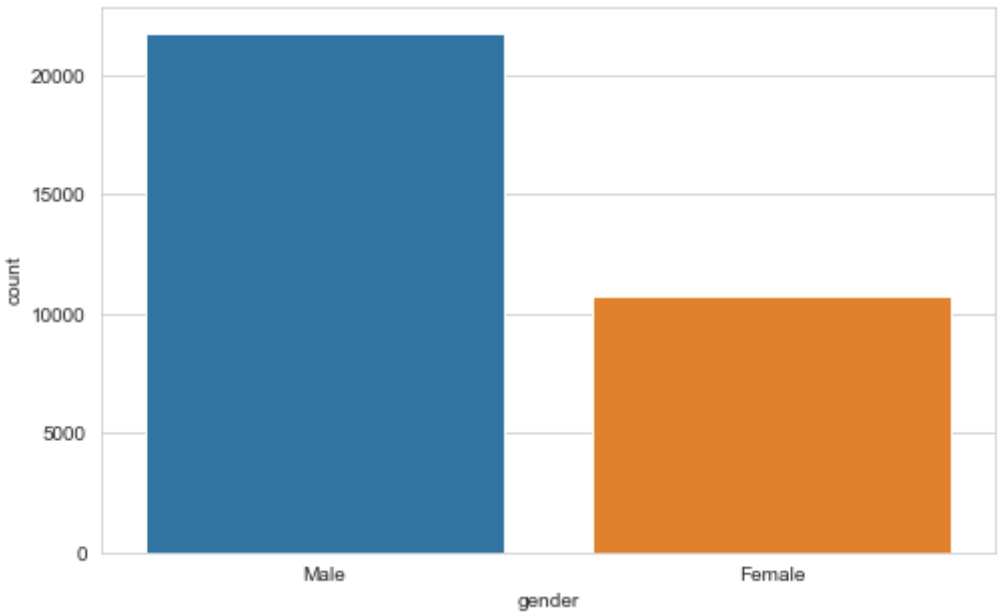
Desired Output: Male 21775 Female 10762 Name: gender, dtype: int64

In [1162]:

```
# Your Code is Here
g = sns.countplot(x='sex', data=df)
plt.bar_label(g.containers[0]);
```



Desired Output:



Check the count of person in each "salary" levels by gender and visualize it with countplot

In [1163]:

```
# Your Code is Here
df.groupby('sex').salary.value_counts(dropna=False)
```

Out[1163]:

```
sex     salary
Female  <=50K      9583
        >50K       1179
Male    <=50K     15115
        >50K       6660
Name: salary, dtype: int64
```

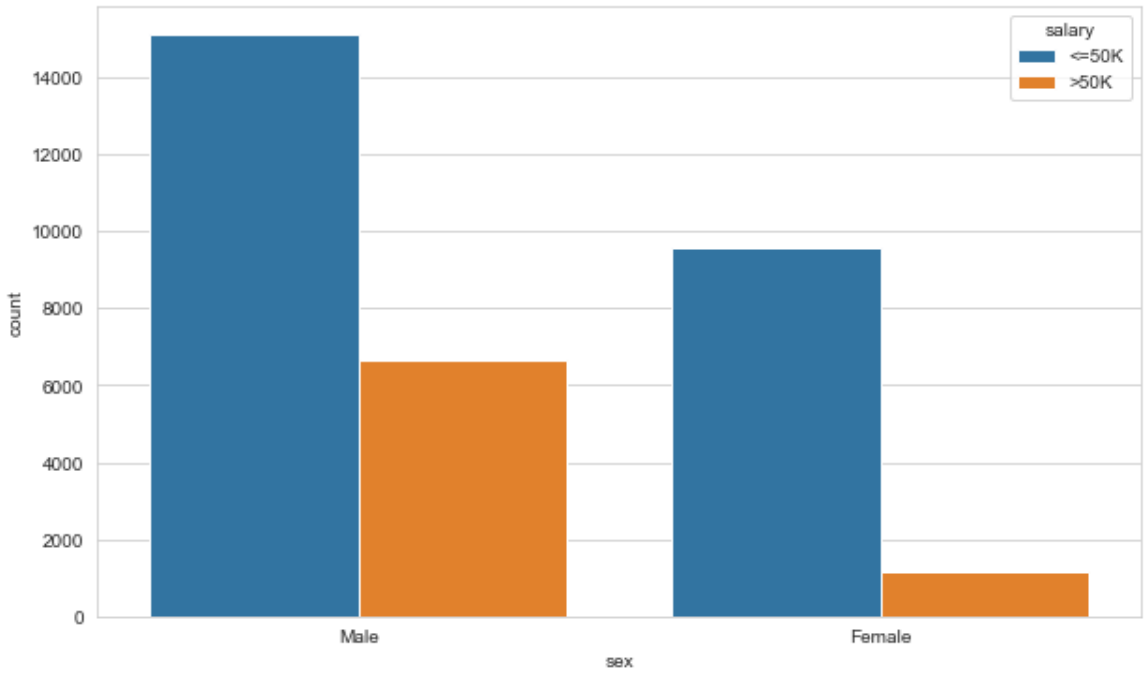
Desired Output: gender salary Female <=50K 9583 >50K 1179 Male <=50K 15115 >50K 6660 Name: salary, dtype: int64

In [1164]:

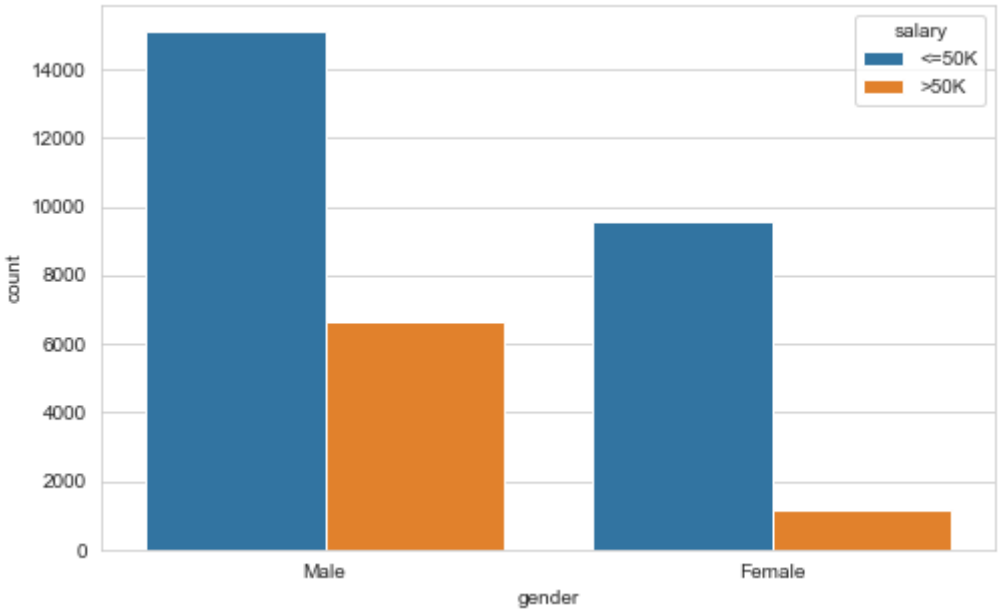
```
# Your Code is Here
sns.countplot(x='sex',hue='salary',data=df)
```

Out[1164]:

```
<AxesSubplot:xlabel='sex', ylabel='count'>
```



Desired Output:



Check the percentage distribution of person in each "salary" levels by each gender and visualize it with pie plot

In [1165]:

```
# Your Code is Here

df.groupby('sex').salary.value_counts(normalize=True)
```

Out[1165]:

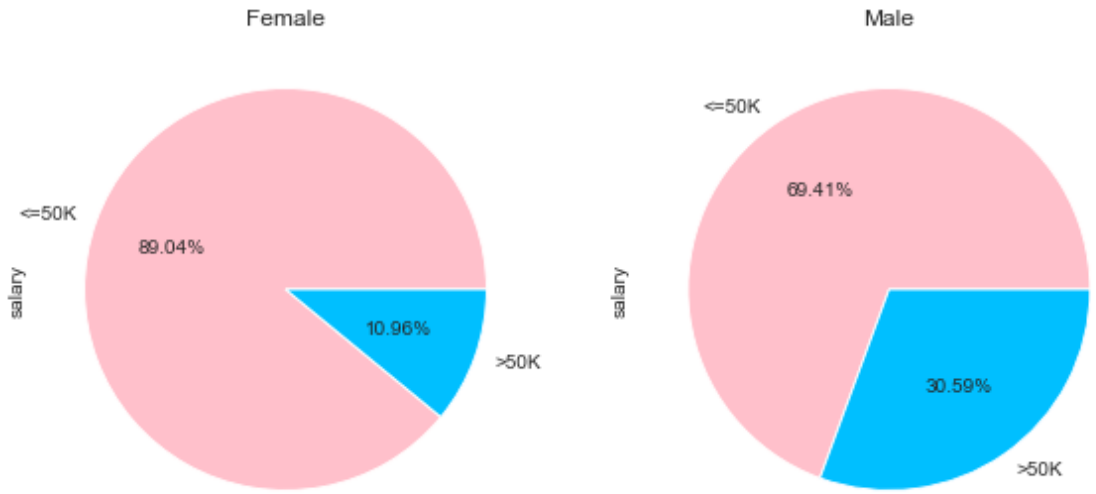
```
sex     salary
Female  <=50K    0.890
        >50K     0.110
Male    <=50K    0.694
        >50K     0.306
Name: salary, dtype: float64
```

Desired Output: gender salary Female <=50K 0.890 >50K 0.110 Male <=50K 0.694 >50K 0.306 Name: salary,

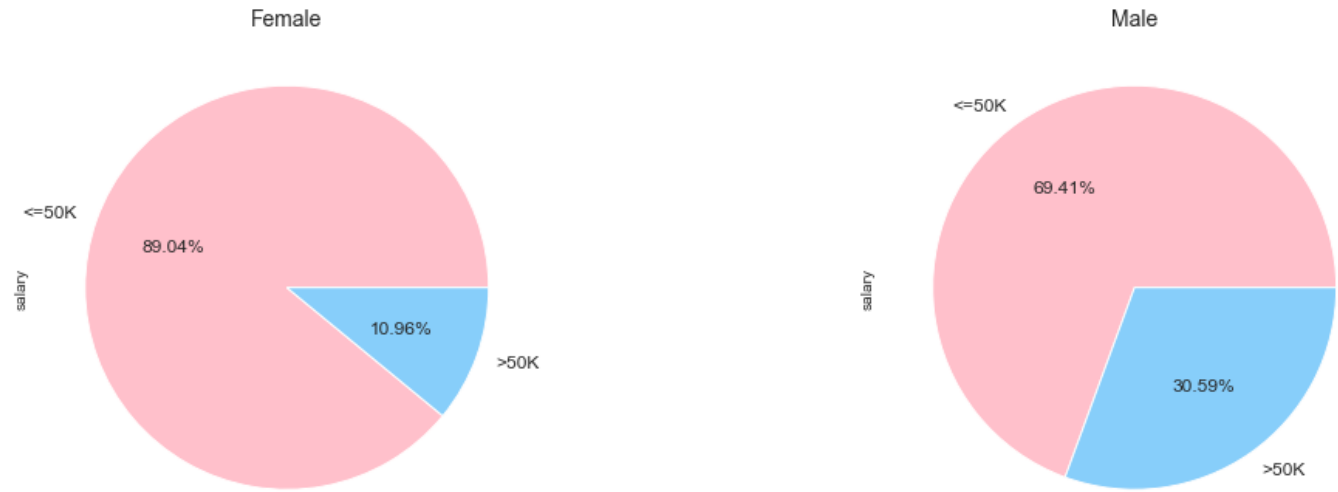
dtype: float64

In [1166]:

```
# Your Code is Here
for count,x in enumerate(sorted(df.sex.unique())):
    plt.subplot(1,2,count+1)
    df[df.sex == x].salary.value_counts().plot.pie(autopct='%0.2f%%', colors=['p
ink','deepskyblue'])
    plt.title(x);
```



Desired Output:



Check the count of person in each gender by "salary" levels and visualize it with countplot

In [1167]:

```
# Your Code is Here
df.groupby('salary').sex.value_counts()
```

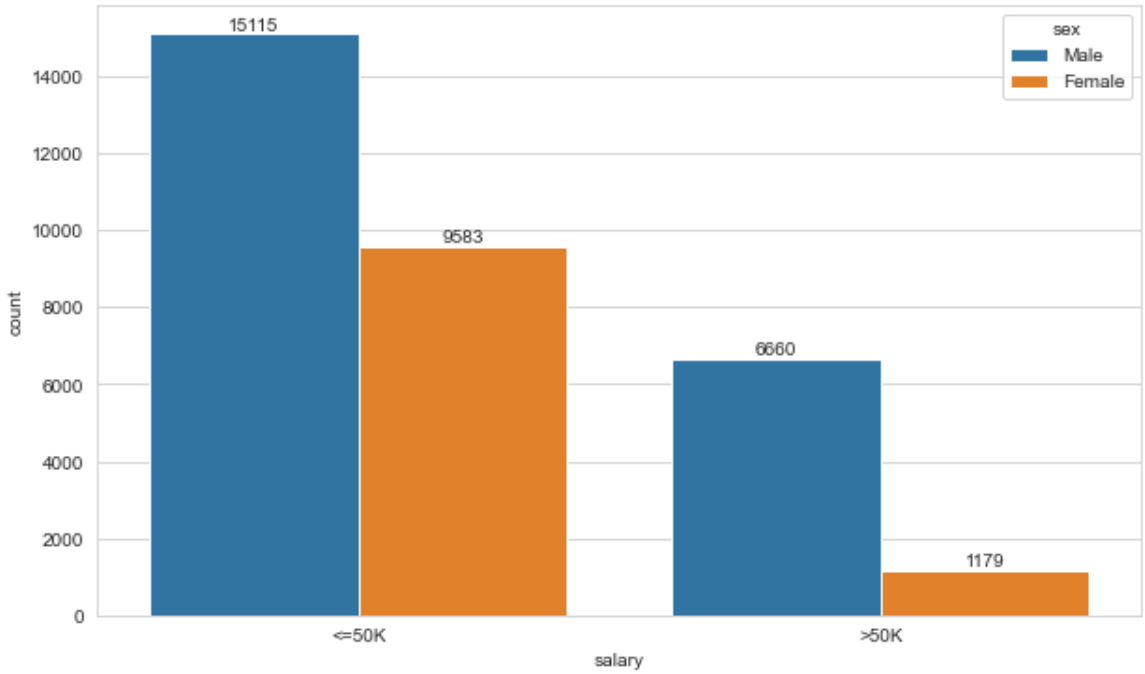
Out[1167]:

```
salary  sex
<=50K   Male      15115
        Female     9583
>50K    Male       6660
        Female     1179
Name: sex, dtype: int64
```

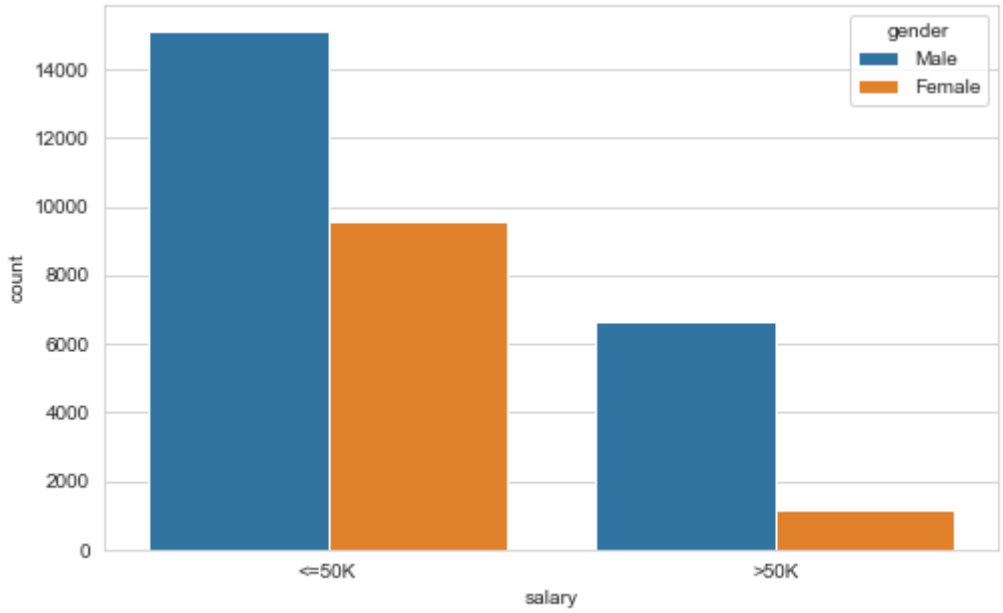
Desired Output: salary gender <=50K Male 15115 Female 9583 >50K Male 6660 Female 1179 Name: gender, dtype: int64

In [1168]:

```
# Your Code is Here
g = sns.countplot(x='salary', hue = 'sex', data=df)
plt.bar_label(g.containers[0])
plt.bar_label(g.containers[1]);
```



Desired Output:



Check the the percentage distribution of person in each gender by "salary" levels and visualize it with pie plot

In [1169]:

```
# Your Code is Here
df.groupby('salary').sex.value_counts(normalize=True)
```

Out[1169]:

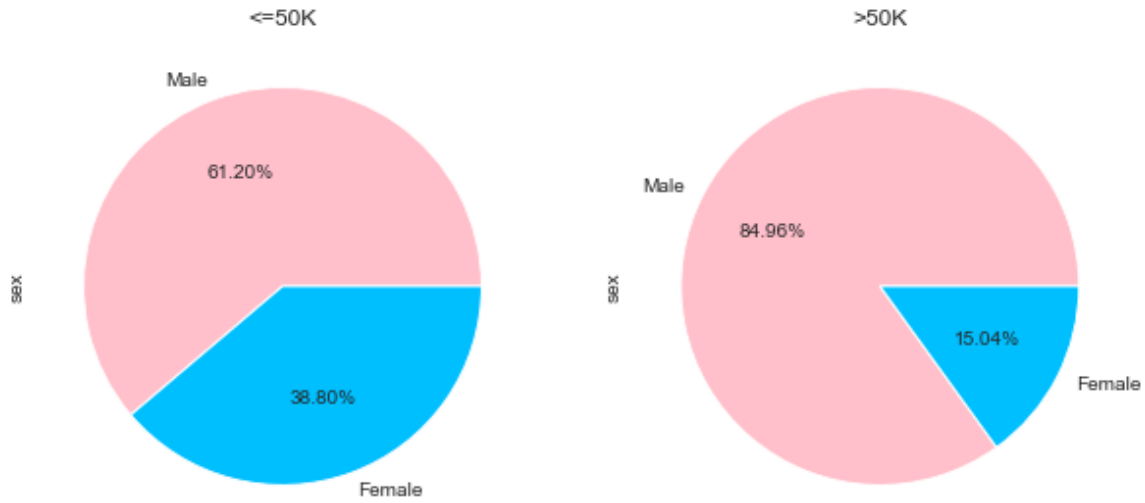
```
salary  sex
<=50K   Male      0.612
        Female    0.388
>50K    Male      0.850
        Female    0.150
Name: sex, dtype: float64
```

Desired Output: salary gender <=50K Male 0.612 Female 0.388 >50K Male 0.850 Female 0.150 Name: gender,

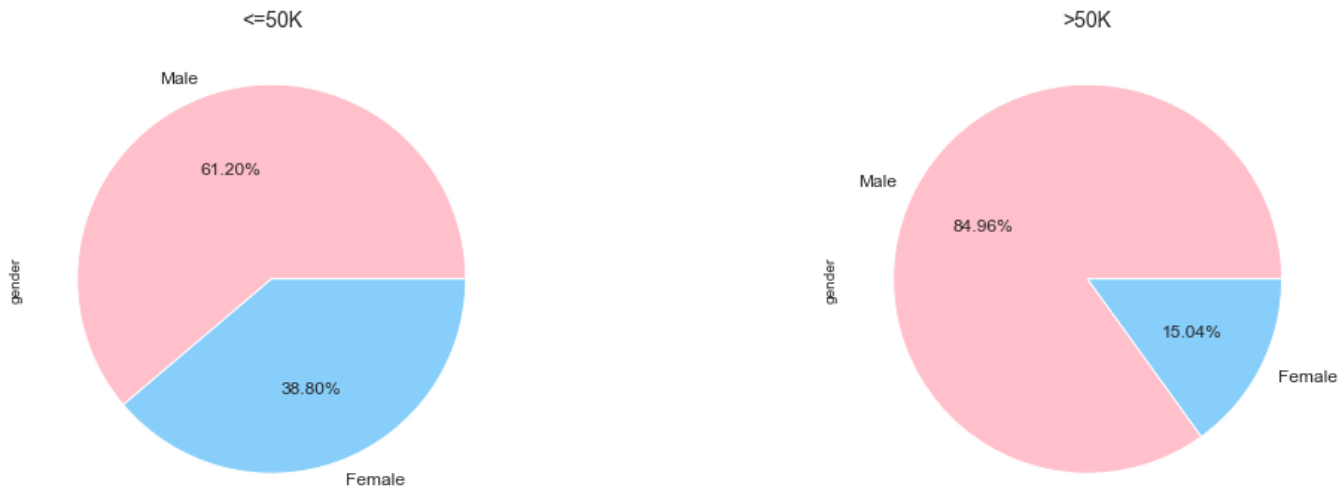
dtype: float64

In [1170]:

```
# Your Code is Here
for count,x in enumerate(sorted(df.salary.unique())):
    plt.subplot(1,2,count+1)
    df[df.salary == x].sex.value_counts().plot.pie(autopct='%0.2f%%', colors=['p
ink','deepskyblue'])
    plt.title(x);
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : When the gender feature is examined in the data set, the male value is seen the most.
9 out of 10 Females earn less than 50K. In Males, this ratio is 7 out of 10.
3 out of 10 Males earn more than 50K.
Males make up the majority of those who earn over 50K (85 percent).

native_country

Check the count of person in each categories and visualize it with countplot

In [1171]:

```
# Your Code is Here
df.native_country.value_counts(dropna=False)
```

Out[1171]:

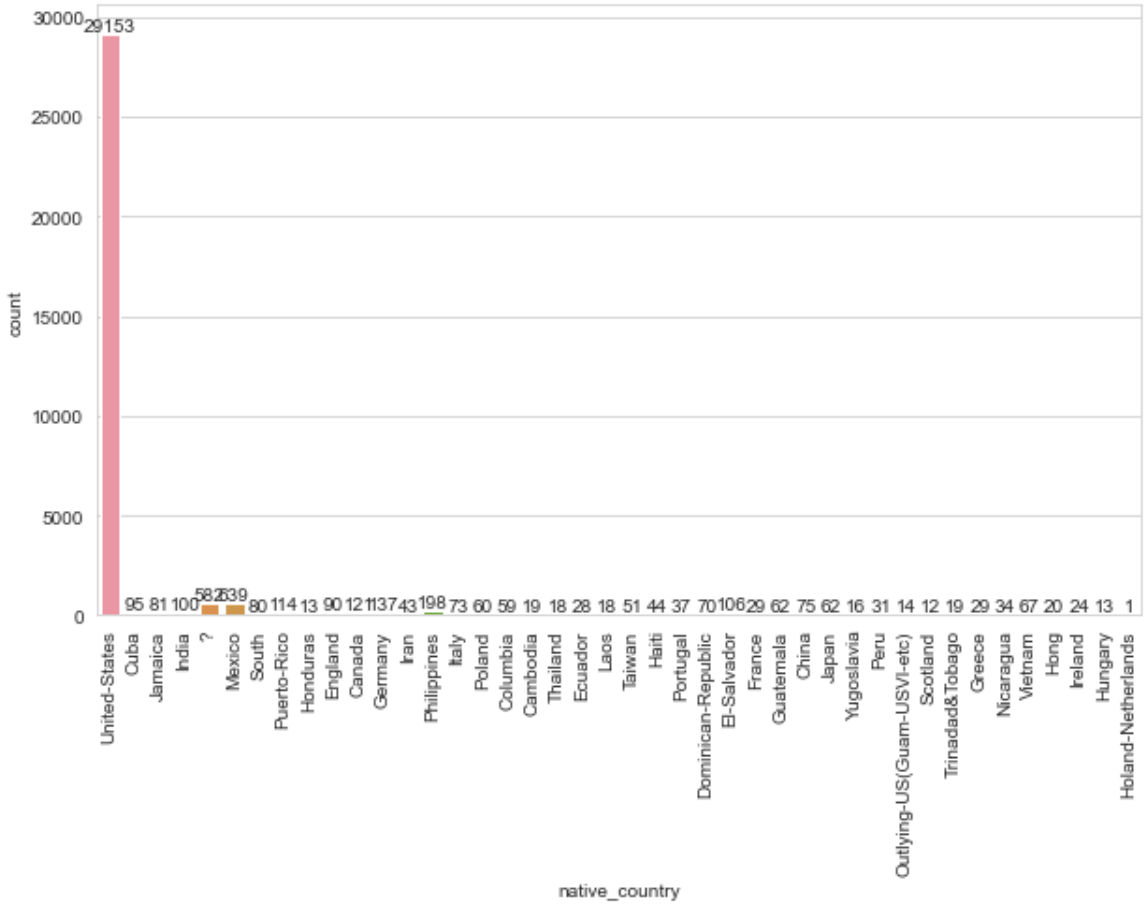
```
United-States                 29153
Mexico                          639
?                               582
Philippines                     198
Germany                         137
Canada                          121
Puerto-Rico                     114
El-Salvador                     106
India                           100
Cuba                             95
England                          90
Jamaica                          81
South                            80
China                            75
Italy                            73
Dominican-Republic               70
Vietnam                          67
Japan                            62
Guatemala                        62
Poland                           60
Columbia                         59
Taiwan                           51
Haiti                            44
Iran                             43
Portugal                         37
Nicaragua                        34
Peru                             31
France                           29
Greece                           29
Ecuador                          28
Ireland                          24
Hong                             20
Cambodia                         19
Trinadad&Tobago                  19
Laos                             18
Thailand                         18
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Honduras                         13
Hungary                          13
Scotland                         12
Holand-Netherlands                1
Name: native_country, dtype: int64
```

Desired Output: United-States 29153 Mexico 639 ? 582 Philippines 198 Germany 137 Canada 121 Puerto-Rico 114 El-Salvador 106 India 100 Cuba 95 England 90 Jamaica 81 South 80 China 75 Italy 73 Dominican-Republic 70 Vietnam 67 Japan 62 Guatemala 62 Poland 60 Columbia 59 Taiwan 51 Haiti 44 Iran 43 Portugal 37 Nicaragua 34 Peru 31 France 29 Greece 29 Ecuador 28 Ireland 24 Hong 20 Cambodia 19 Trinadad&Tobago 19 Laos 18 Thailand 18 Yugoslavia 16 Outlying-US(Guam-USVI-etc) 14 Honduras 13 Hungary 13 Scotland 12 Holand-

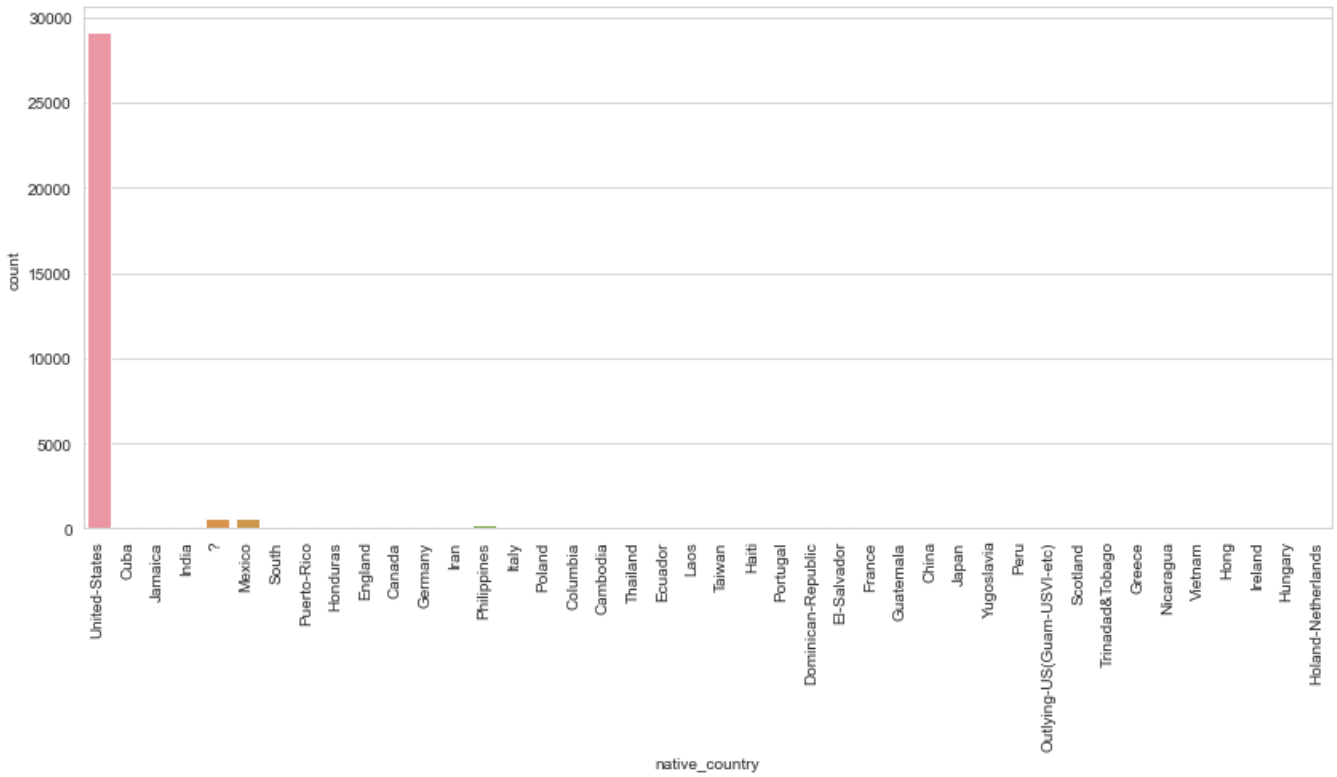
Netherlands 1 Name: native_country, dtype: int64

In [1172]:

```
# Your Code is Here
g = sns.countplot(x='native_country', data=df)
plt.xticks(rotation = 90)
plt.bar_label(g.containers[0]);
```



Desired Output:



Replace the value "?" to the value "Unknown"

In [1173]:

```
# Replace "?" values with "Unknown"

# Your Code is Here
df.native_country = df.native_country.replace({'?':'Unknown'})
```

Decrease the number of categories in "native_country" feature as US, and Others and create a new feature with this new categorical data

In [1174]:

```
def mapping_native_country(x):
    if x == "United-States":
        return "US"
    else:
        return "Others"
```

In [1175]:

```
# Your Code is Here
df.native_country.apply(lambda x: mapping_native_country(x)).value_counts()
```

Out[1175]:

```
US        29153
Others     3384
Name: native_country, dtype: int64
```

Desired Output: US 29153 Others 3384 Name: native_country, dtype: int64

In [1176]:

```
# By using "mapping_native_country" def function above, create a new column name
d "native_country_summary"

# Your Code is Here
df['native_country_summary'] = df.native_country.apply(lambda x: mapping_native_
country(x))
df['native_country_summary']
```

Out[1176]:

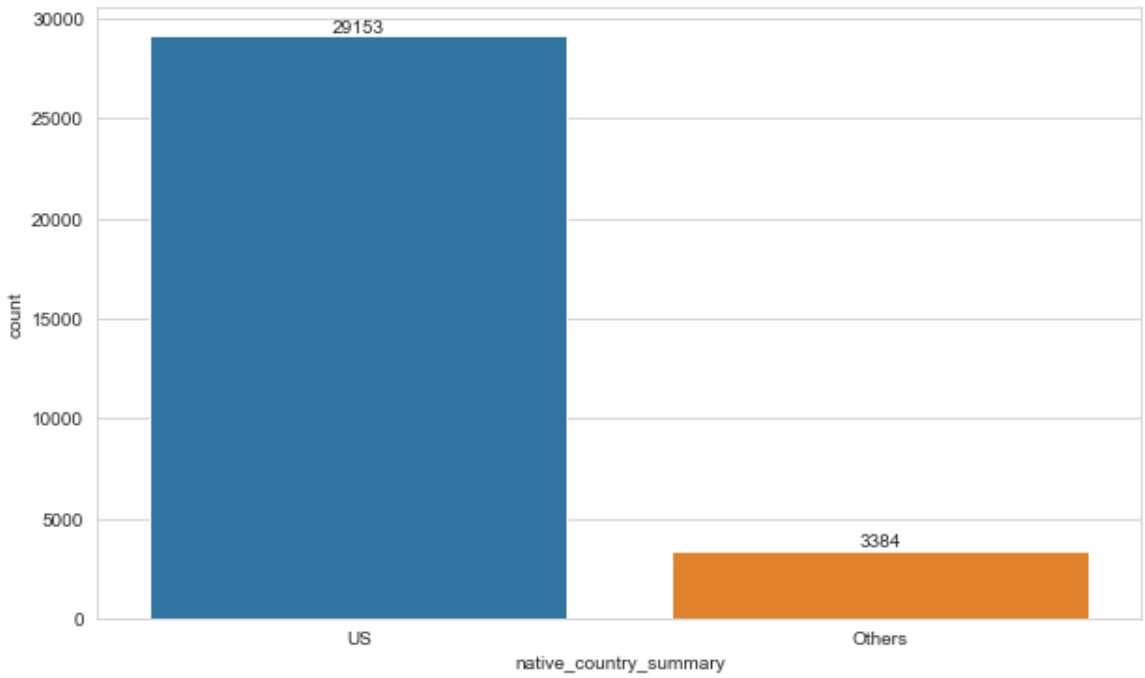
```
0            US
1            US
2            US
3            US
4        Others
          ...
32556        US
32557        US
32558        US
32559        US
32560        US
Name: native_country_summary, Length: 32537, dtype: object
```

Desired Output: 0 US 1 US 2 US 3 US 4 Others ... 32556 US 32557 US 32558 US 32559 US 32560 US Name: native_country_summary, Length: 32537, dtype: object

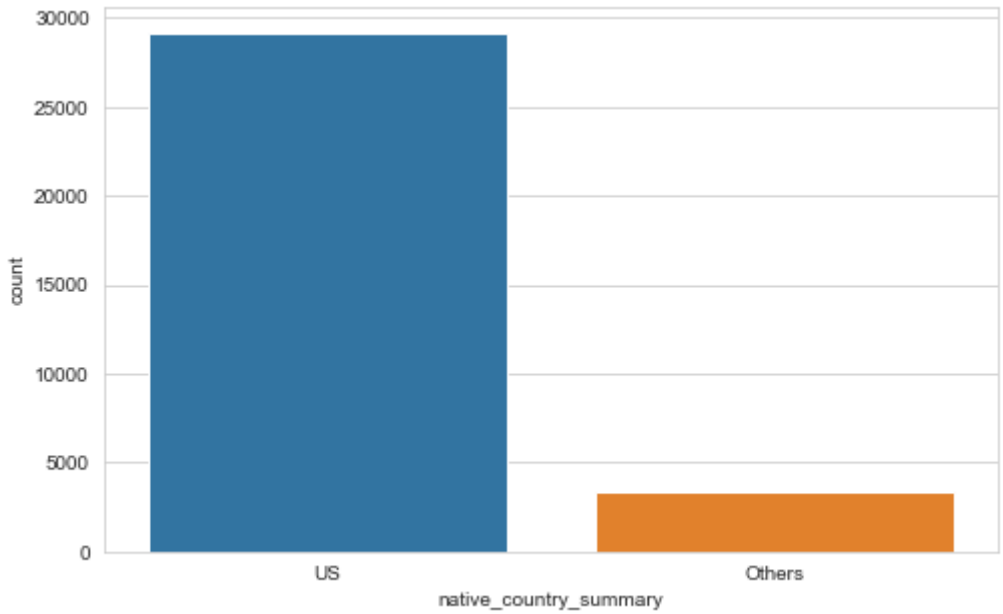
Visualize the count of person in each new categories (US, Others)

In [1177]:

```
# Your Code is Here
g= sns.countplot(x='native_country_summary', data=df)
plt.bar_label(g.containers[0]);
```



Desired Output:



Check the count of person in each "salary" levels by these new native countries (US, Others) and visualize it with countplot

In [1178]:

```
# Your Code is Here
df.groupby('native_country_summary').salary.value_counts()
```

Out[1178]:

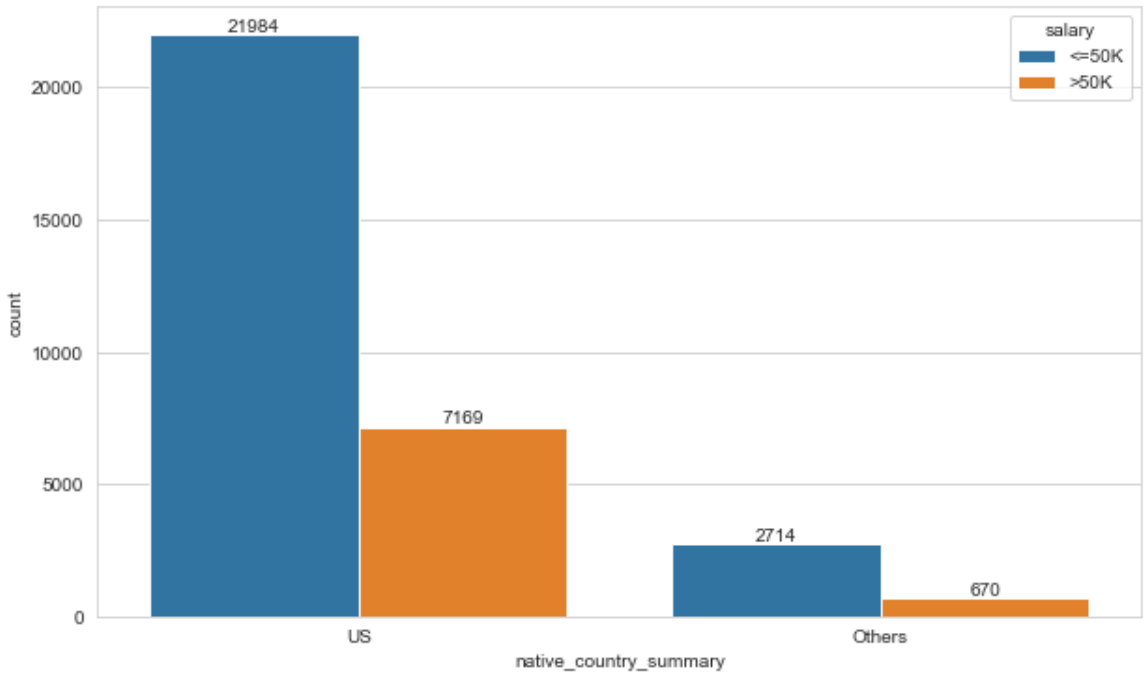
```
native_country_summary  salary
Others                  <=50K      2714
                        >50K        670
US                      <=50K     21984
                        >50K       7169
Name: salary, dtype: int64
```

Desired Output: native_country_summary salary Others <=50K 2714 >50K 670 US <=50K 21984 >50K 7169

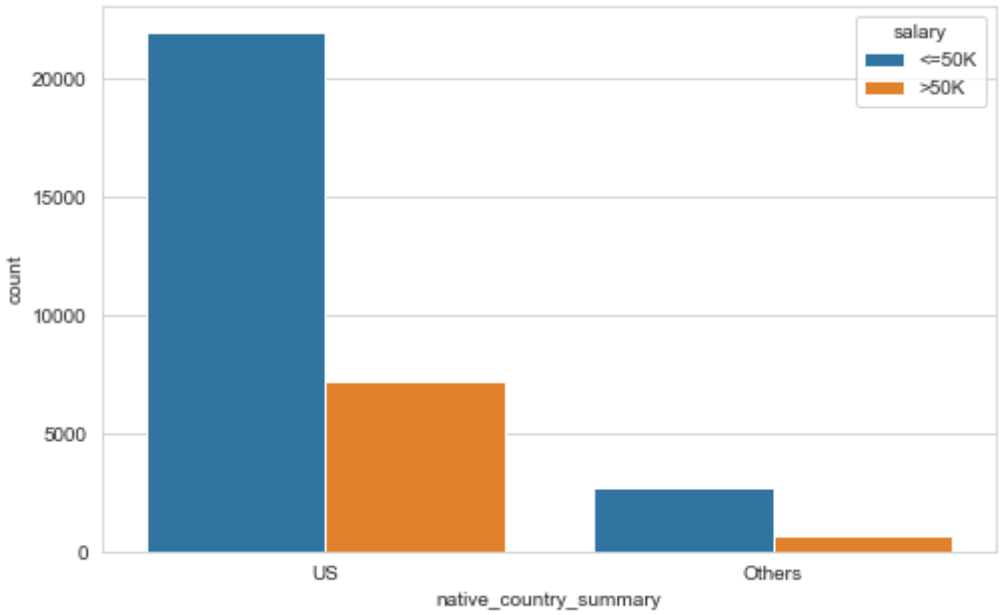
Name: salary, dtype: int64

In [1179]:

```
# Your Code is Here
g=sns.countplot(x='native_country_summary', data=df, hue='salary')
for x in g.containers:
    plt.bar_label(x);
```



Desired Output:



Check the percentage distribution of person in each "salary" levels by each new native countries (US, Others) and visualize it with pie plot separately

In [1180]:

```
# Your Code is Here
df.groupby('native_country_summary').salary.value_counts(normalize= True)
```

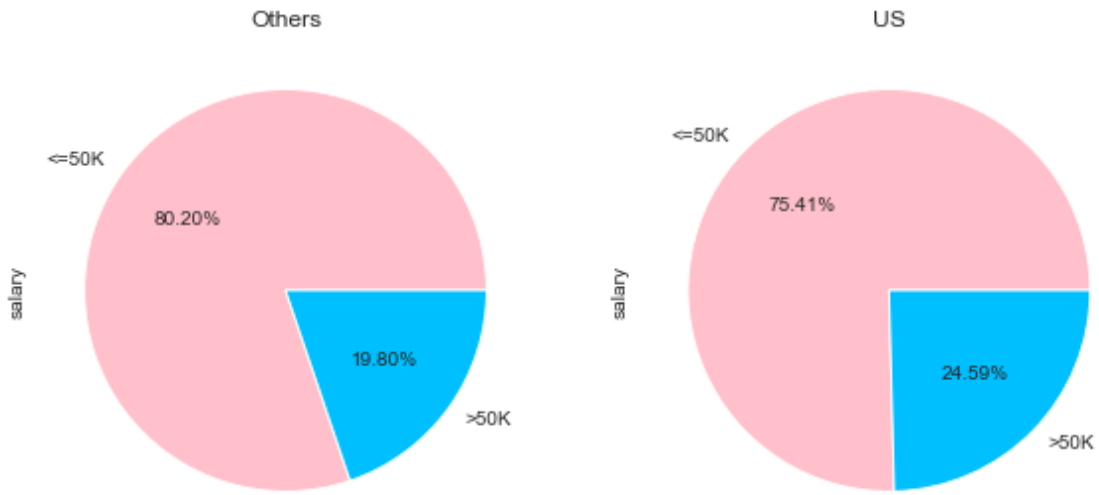
Out[1180]:

```
native_country_summary  salary
Others                  <=50K     0.802
                        >50K      0.198
US                      <=50K     0.754
                        >50K      0.246
Name: salary, dtype: float64
```

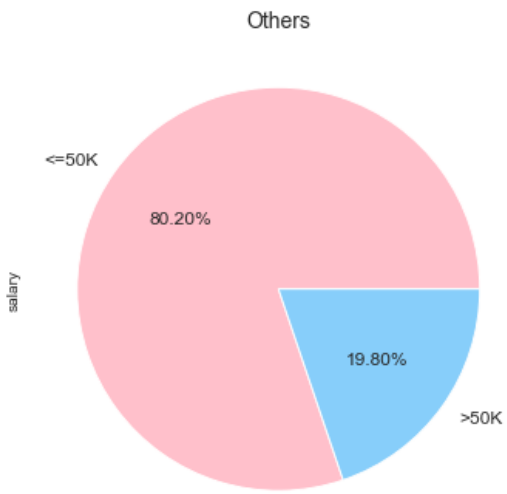
Desired Output: native_country_summary salary Others <=50K 0.802 >50K 0.198 US <=50K 0.754 >50K 0.246 Name: salary, dtype: float64

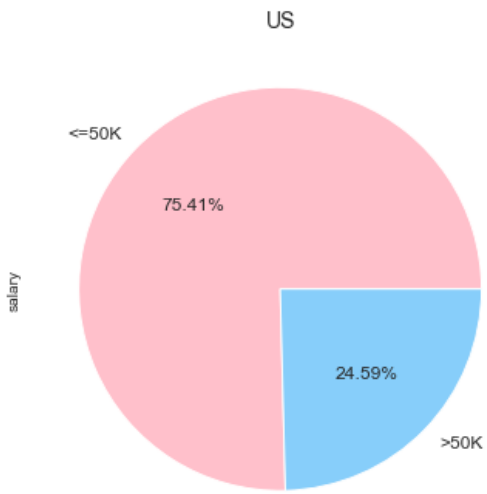
In [1181]:

```
# Your Code is Here
for count,x in enumerate(sorted(df.native_country_summary.unique())):
    plt.subplot(1,2,count+1)
    df[df.native_country_summary == x].salary.value_counts().plot.pie(autopct='%
0.2f%%', colors=['pink','deepskyblue'])
    plt.title(x);
```



Desired Output:





Check the count of person in each these new native countries (US, Others) by "salary" levels and visualize it with countplot

In [1182]:

```
# Your Code is Here
df.groupby('salary').native_country_summary.value_counts()
```

Out[1182]:

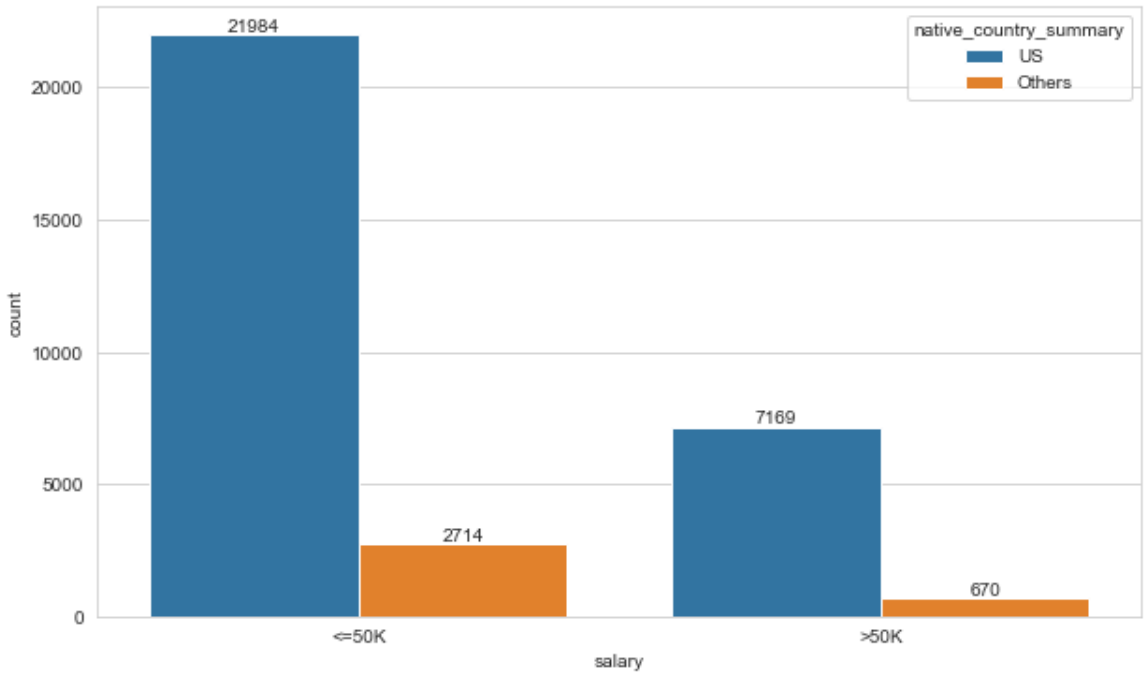
```
salary  native_country_summary
<=50K   US                       21984
        Others                    2714
>50K    US                       7169
        Others                    670
Name: native_country_summary, dtype: int64
```

Desired Output: salary native_country_summary <=50K US 21984 Others 2714 >50K US 7169 Others 670

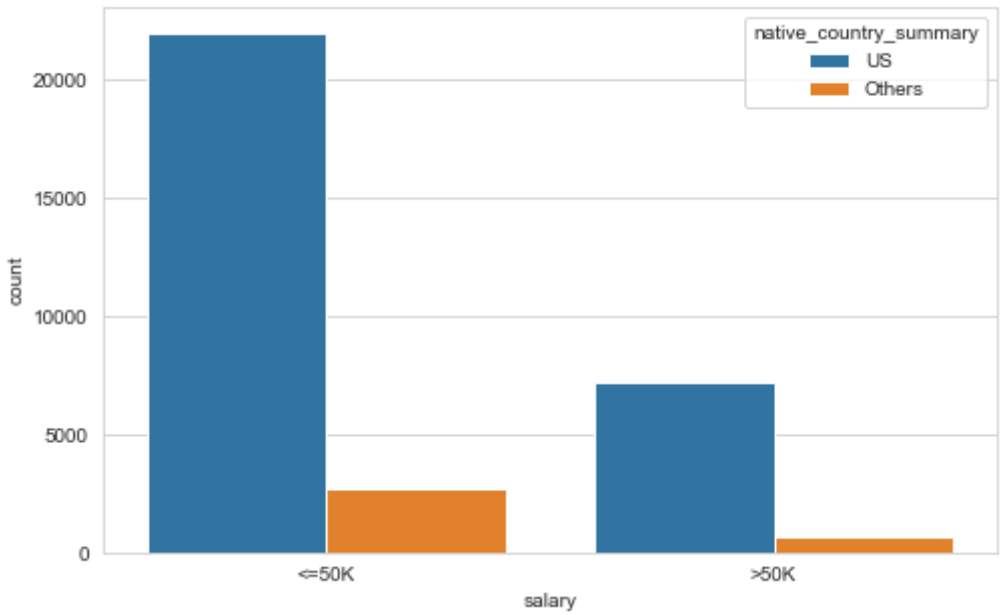
Name: native_country_summary, dtype: int64

In [1183]:

```
# Your Code is Here
g= sns.countplot(x='salary', hue='native_country_summary', data=df)
plt.bar_label(g.containers[0])
plt.bar_label(g.containers[1]);
```



Desired Output:



Check the the percentage distribution of person in each these new native countries (US, Others) by "salary" levels and visualize it with pie plot separately

In [1184]:

```
# Your Code is Here
df.groupby('salary').native_country_summary.value_counts(normalize= True)
```

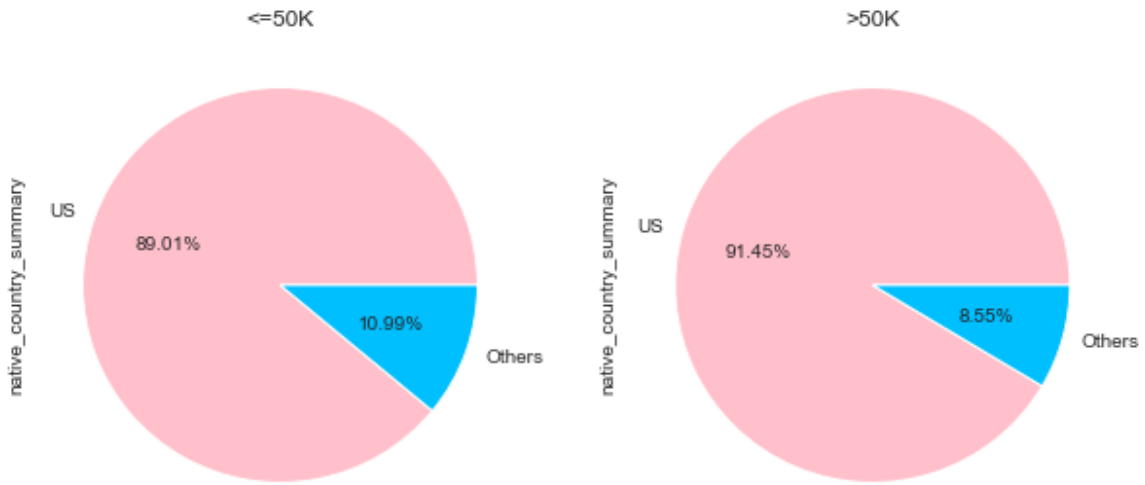
Out[1184]:

```
salary  native_country_summary
<=50K   US                        0.890
        Others                    0.110
>50K    US                        0.915
        Others                    0.085
Name: native_country_summary, dtype: float64
```

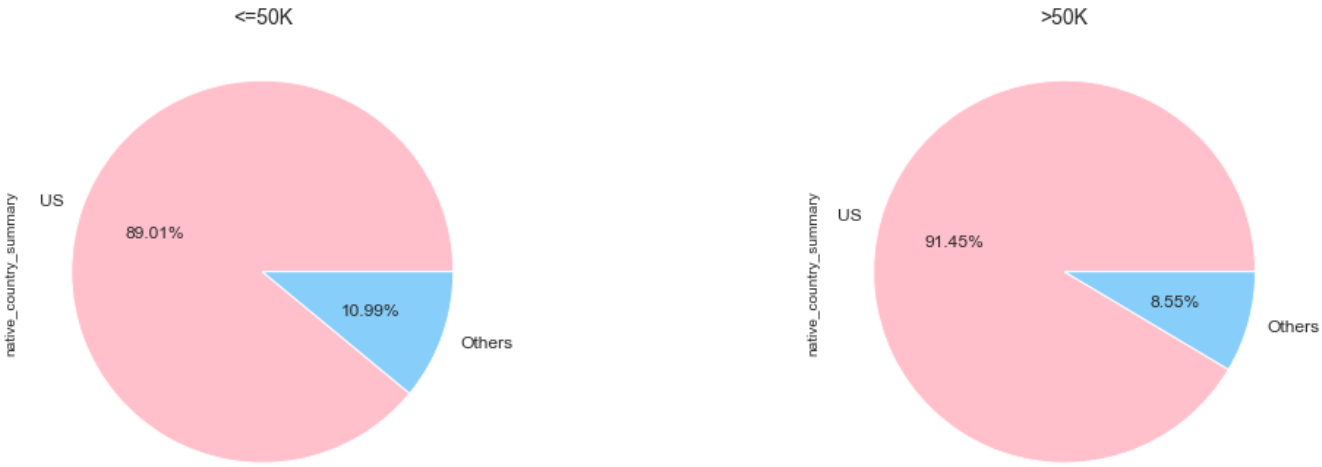
Desired Output: salary native_country_summary <=50K US 0.890 Others 0.110 >50K US 0.915 Others 0.085 Name: native_country_summary, dtype: float64

In [1185]:

```
# Your Code is Here
for count,x in enumerate(sorted(df.salary.unique())):
    plt.subplot(1,2,count+1)
    df[df.salary == x].native_country_summary.value_counts().plot.pie(autopct='%
0.2f%%', colors=['pink','deepskyblue'])
    plt.title(x);
```



Desired Output:



Write down the conclusions you draw from your analysis

Result : When the native_country feature is examined in the data set, US is seen the most.
The rate of earning over 50K in the US is higher than the others.
The US constitutes the majority of those earning less than 50K (89 percent).
The US makes up the majority of those who earn over 50K (91 percent).

Other Specific Analysis Questions

Content

1. What is the average age of males and females by income level?

In [1186]:

```
# Your Code is Here
df.groupby(['salary','sex']).age.mean()
```

Out[1186]:

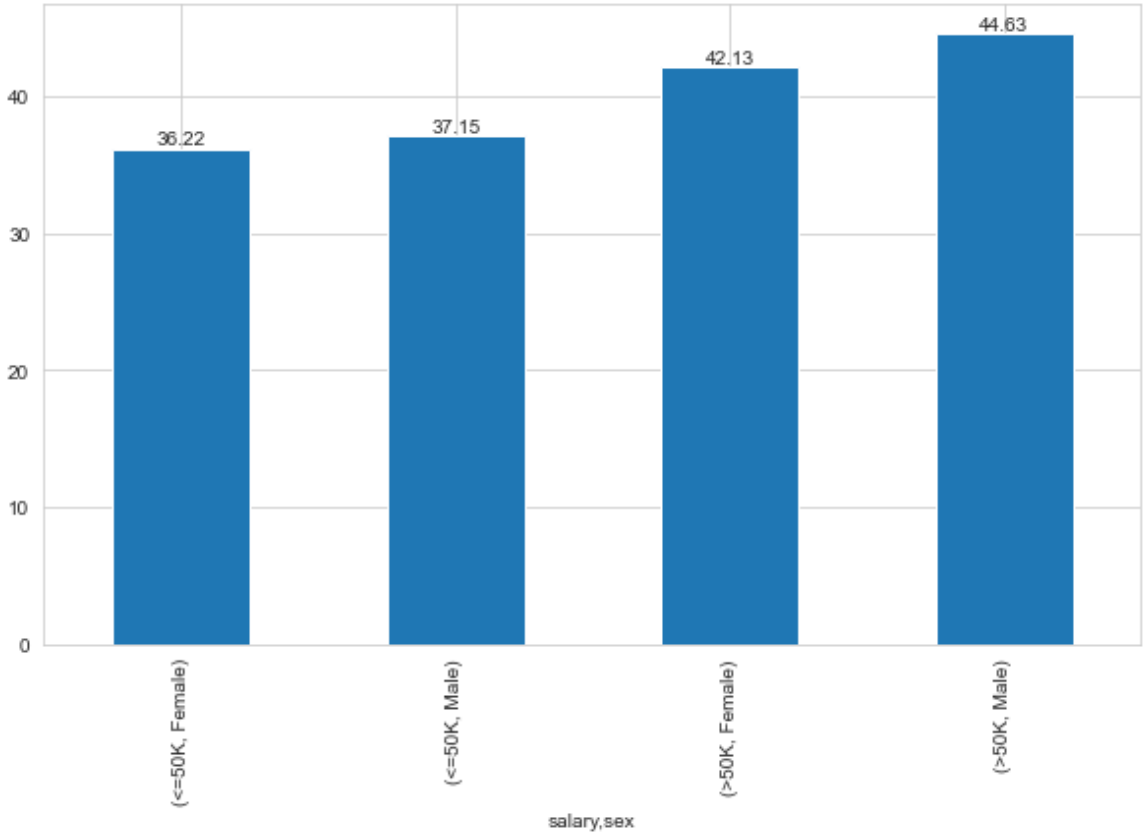
```
salary  sex
<=50K   Female   36.217
        Male     37.149
>50K    Female   42.126
        Male     44.627
Name: age, dtype: float64
```

Desired Output: salary gender <=50K Female 36.217 Male 37.149 >50K Female 42.126 Male 44.627 Name: age,

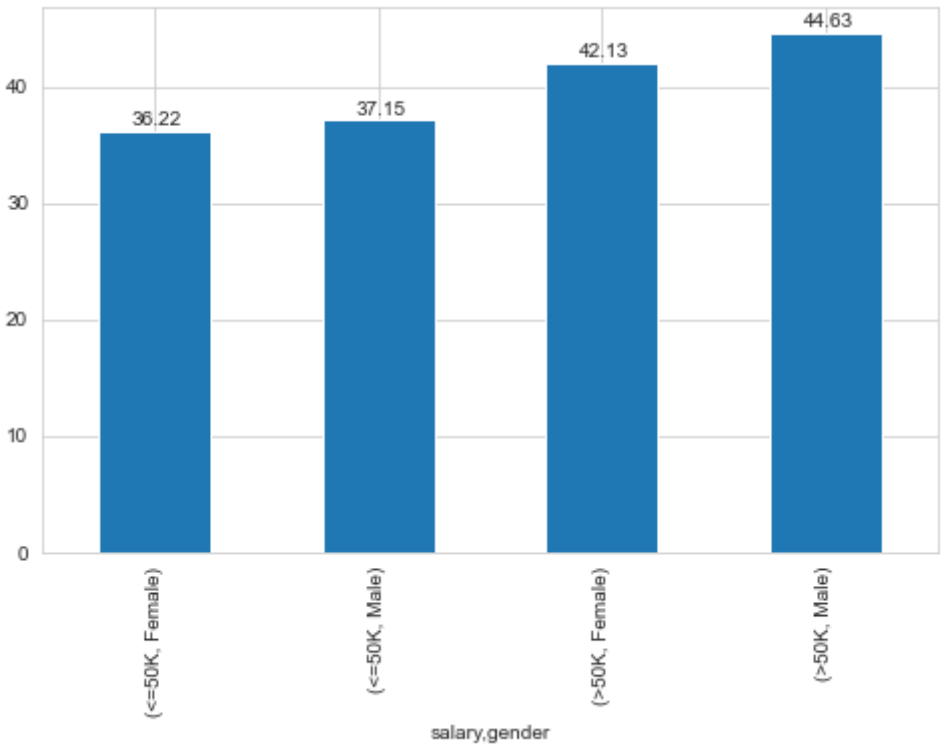
dtype: float64

In [1187]:

```
# Your Code is Here
g = df.groupby(['salary','sex']).age.mean().plot.bar()
plt.bar_label(g.containers[0], fmt='%0.2f');
```



Desired Output:



In [1188]:

```
# Your Code is Here
df.groupby(['salary','sex']).age.mean().reset_index()
```

Out[1188]:

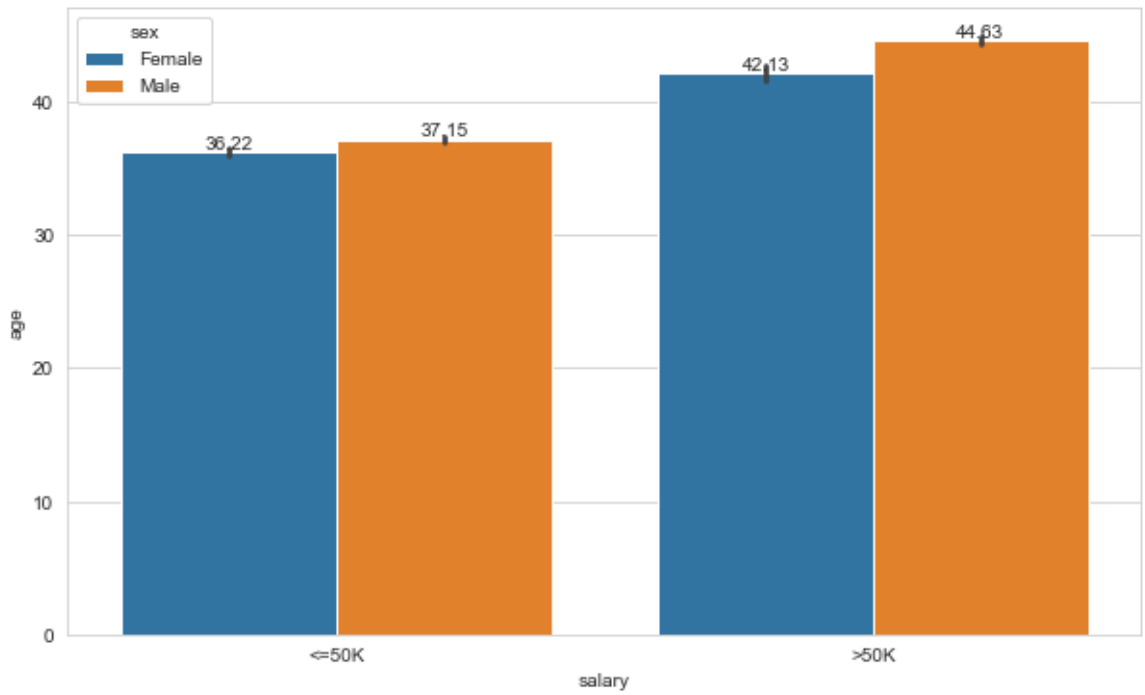
	salary	sex	age
0	<=50K	Female	36.217
1	<=50K	Male	37.149
2	>50K	Female	42.126
3	>50K	Male	44.627

Desired Output:

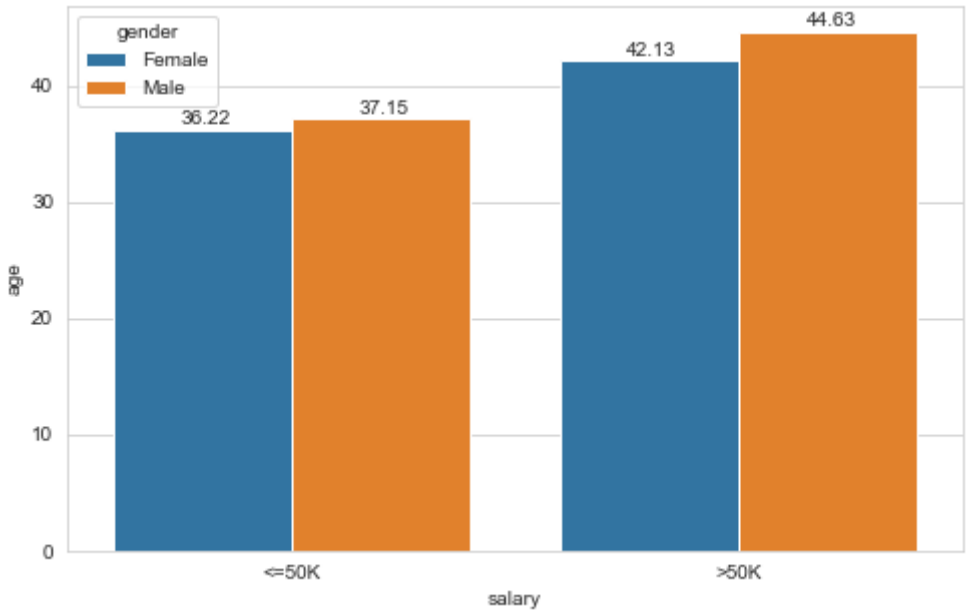
	salary	gender	age
0	<=50K	Female	36.217
1	<=50K	Male	37.149
2	>50K	Female	42.126
3	>50K	Male	44.627

In [1189]:

```
# Your Code is Here
g = sns.barplot(x='salary', y='age', hue='sex', data=df, hue_order=['Female','Male'])
for x in g.containers:
    plt.bar_label(x,fmt='%0.2f');
```



Desired Output:



2. What is the workclass percentages of Americans in high-level income group?

In [1190]:

```
# Your Code is Here
df[(df.native_country == 'United-States') & (df.salary == '>50K')].workclass.val
ue_counts(normalize=True) * 100
```

Out[1190]:

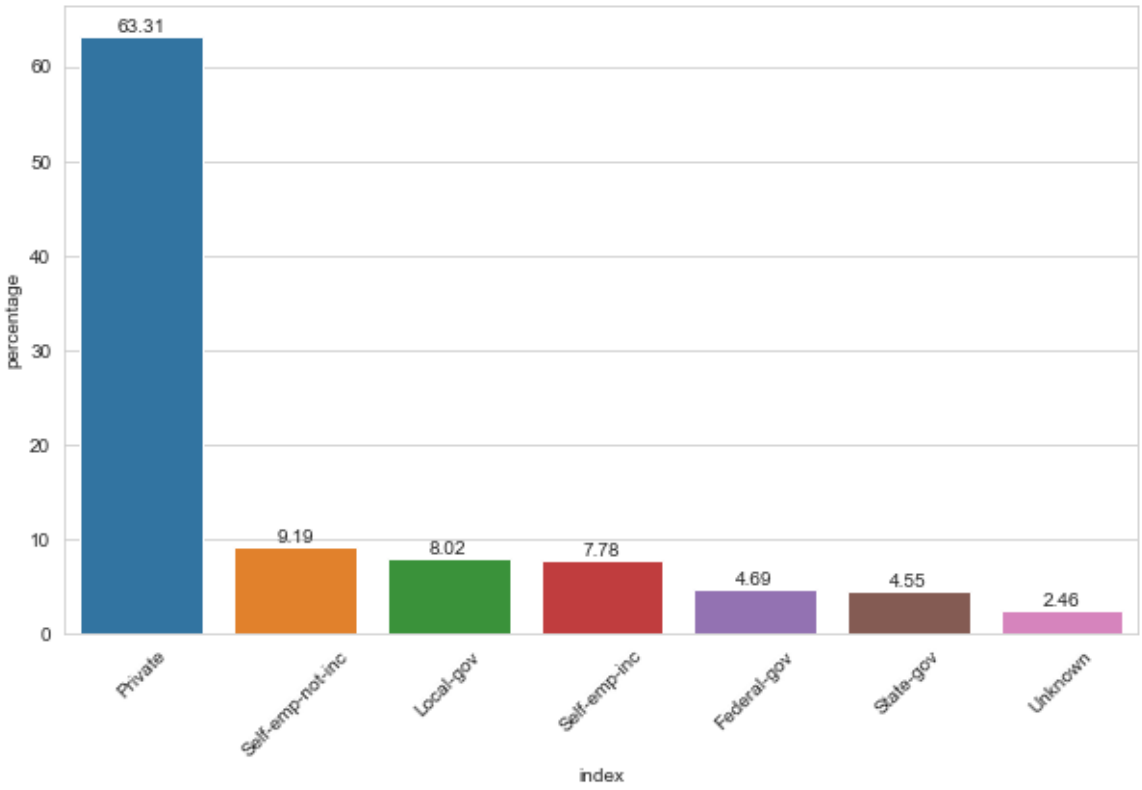
```
Private             63.314
Self-emp-not-inc     9.192
Local-gov            8.021
Self-emp-inc         7.784
Federal-gov          4.687
State-gov            4.547
Unknown              2.455
Name: workclass, dtype: float64
```

Desired Output: Private 63.314 Self-emp-not-inc 9.192 Local-gov 8.021 Self-emp-inc 7.784 Federal-gov 4.687

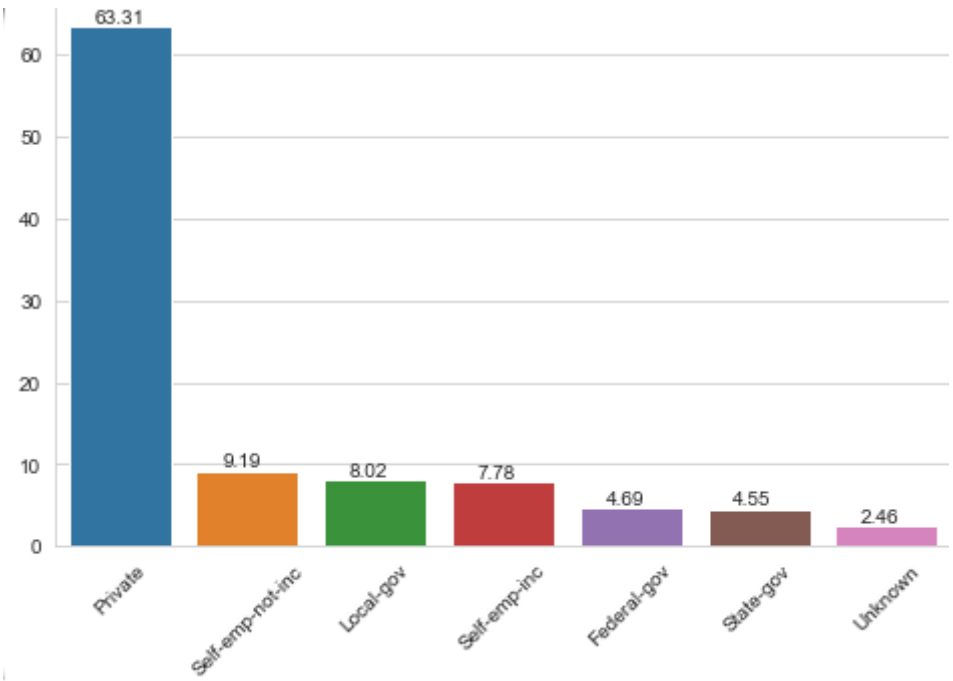
State-gov 4.547 Unknown 2.455 Name: workclass, dtype: float64

In [1191]:

```
# Your Code is Here
df_temp  = (df[(df.native_country == 'United-States') & (df.salary == '>50K')].w
orkclass.value_counts(normalize=True) * 100)
g = sns.barplot(x='index',y='percentage', data=df_temp.rename('percentage').rese
t_index())
plt.xticks(rotation = 45);
plt.bar_label(g.containers[0], fmt='%0.2f');
```



Desired Output:



3. What is the occupation percentages of Americans who work as "Private" workclass in high-level income group?

In [1192]:

```
# Your Code is Here
df[(df.salary == '>50K') & (df.native_country == 'United-States') & (df.workclas
s == 'Private')].occupation.value_counts(normalize = True) * 100
```

Out[1192]:

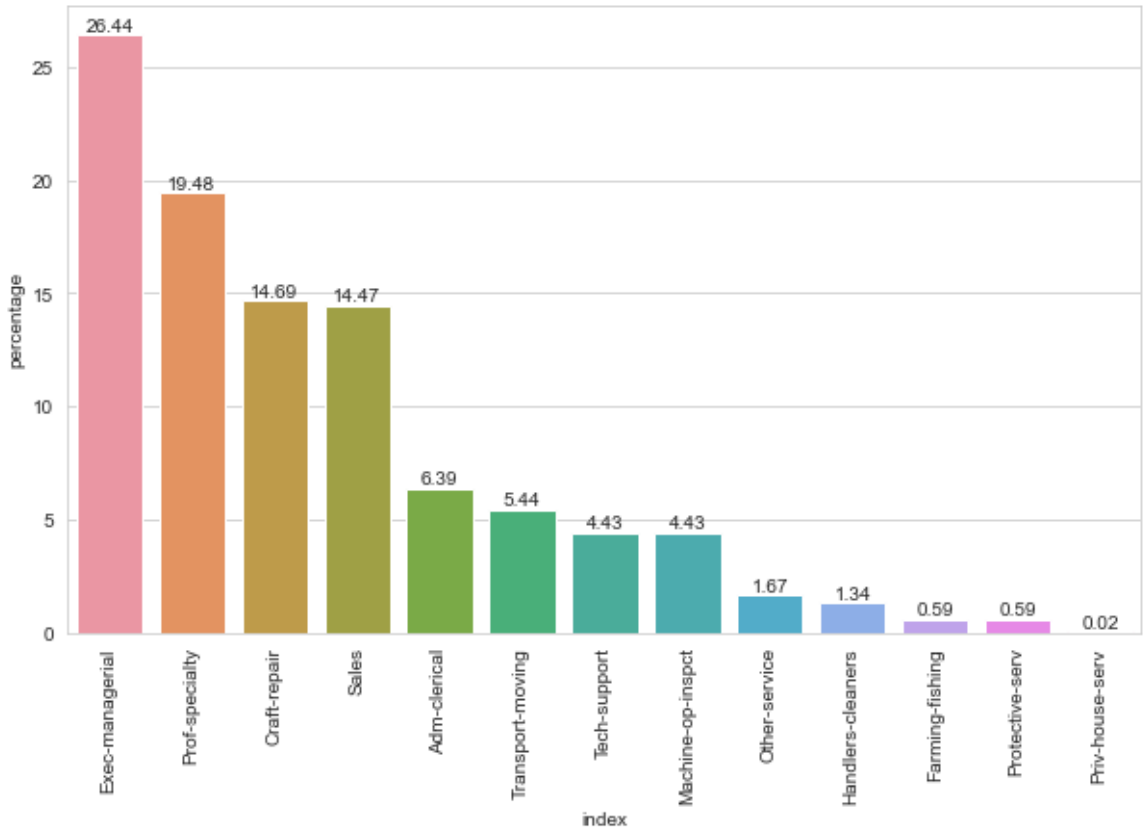
```
Exec-managerial     26.438
Prof-specialty      19.476
Craft-repair        14.695
Sales               14.475
Adm-clerical         6.389
Transport-moving     5.442
Tech-support         4.428
Machine-op-inspct    4.428
Other-service        1.674
Handlers-cleaners    1.344
Farming-fishing      0.595
Protective-serv      0.595
Priv-house-serv      0.022
Name: occupation, dtype: float64
```

Desired Output: Exec-managerial 26.438 Prof-specialty 19.476 Craft-repair 14.695 Sales 14.475 Adm-clerical 6.389 Transport-moving 5.442 Tech-support 4.428 Machine-op-inspct 4.428 Other-service 1.674 Handlers-cleaners 1.344 Farming-fishing 0.595 Protective-serv 0.595 Priv-house-serv 0.022 Name: occupation, dtype:

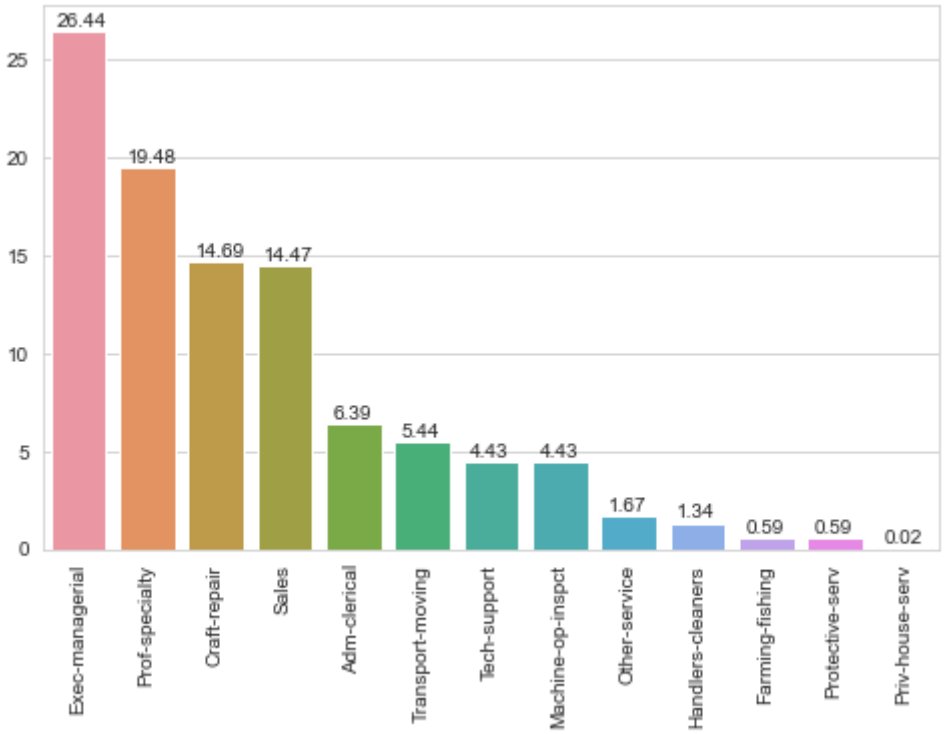
float64

In [1193]:

```
# Your Code is Here
df_temp = (df[(df.salary == '>50K') & (df.native_country == 'United-States') &
(df.workclass == 'Private')].occupation.value_counts(normalize = True) * 100).re
name('percentage').reset_index()
g = sns.barplot(x='index', y='percentage', data=df_temp)
plt.bar_label(g.containers[0], fmt='%0.2f')
plt.xticks(rotation=90);
```



Desired Output:



4. What is the education level percentages of Asian-Pac-Islander race group in high-level income group?

In [1194]:

```
# Your Code is Here
df[(df.salary == '>50K') & (df.race == 'Asian-Pac-Islander')].education.value_co
unts(normalize= True) * 100
```

Out[1194]:

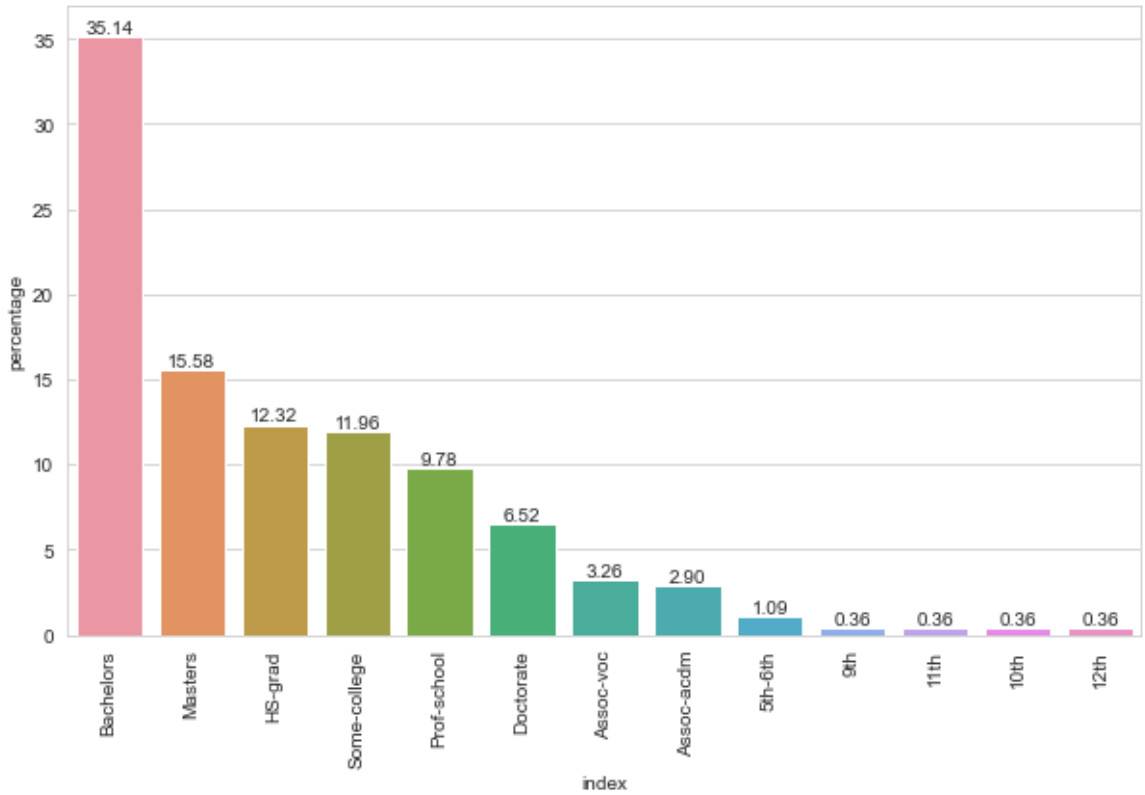
```
Bachelors       35.145
Masters         15.580
HS-grad         12.319
Some-college    11.957
Prof-school      9.783
Doctorate        6.522
Assoc-voc        3.261
Assoc-acdm       2.899
5th-6th          1.087
9th              0.362
11th             0.362
10th             0.362
12th             0.362
Name: education, dtype: float64
```

Desired Output: Bachelors 35.145 Masters 15.580 HS-grad 12.319 Some-college 11.957 Prof-school 9.783 Doctorate 6.522 Assoc-voc 3.261 Assoc-acdm 2.899 5th-6th 1.087 9th 0.362 11th 0.362 10th 0.362 12th 0.362

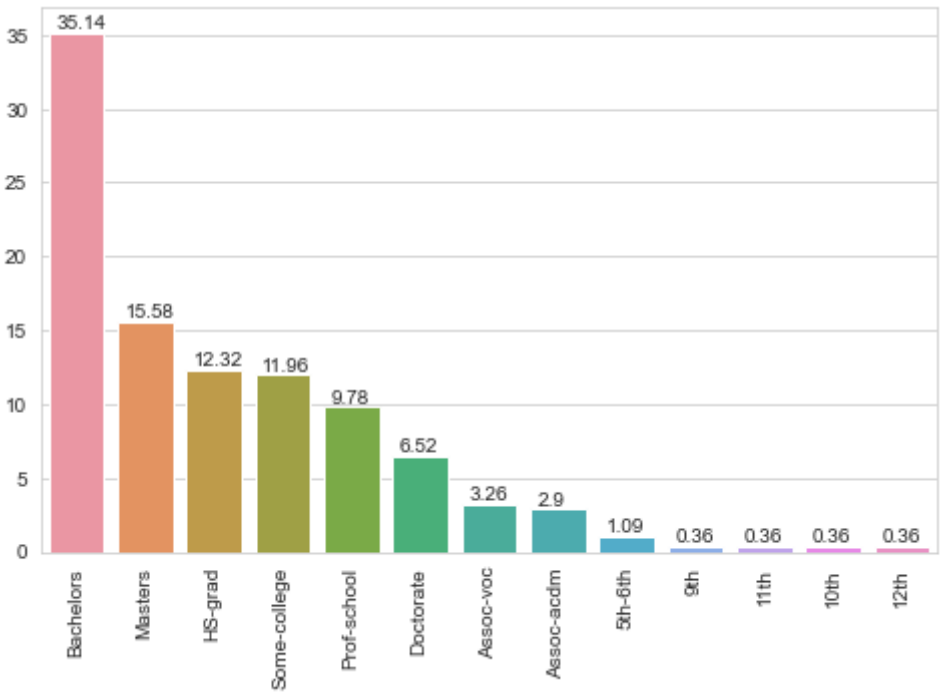
Name: education, dtype: float64

In [1195]:

```
# Your Code is Here
df_temp = (df[(df.salary == '>50K') & (df.race == 'Asian-Pac-Islander')].educati
on.value_counts(normalize= True) * 100).rename('percentage').reset_index()
g = sns.barplot(x='index', y='percentage', data=df_temp)
plt.bar_label(g.containers[0], fmt='%0.2f')
plt.xticks(rotation=90);
```



Desired Output:



5. What is the occupation percentages of Asian-Pac-Islander race group who has a Bachelors degree in high-level income group?

In [1196]:

```
# Your Code is Here
df[(df.salary == '>50K') & (df.race == 'Asian-Pac-Islander') & (df.education == 'Bachelors')].occupation.value_counts(dropna=False, normalize = True) * 100
```

Out[1196]:

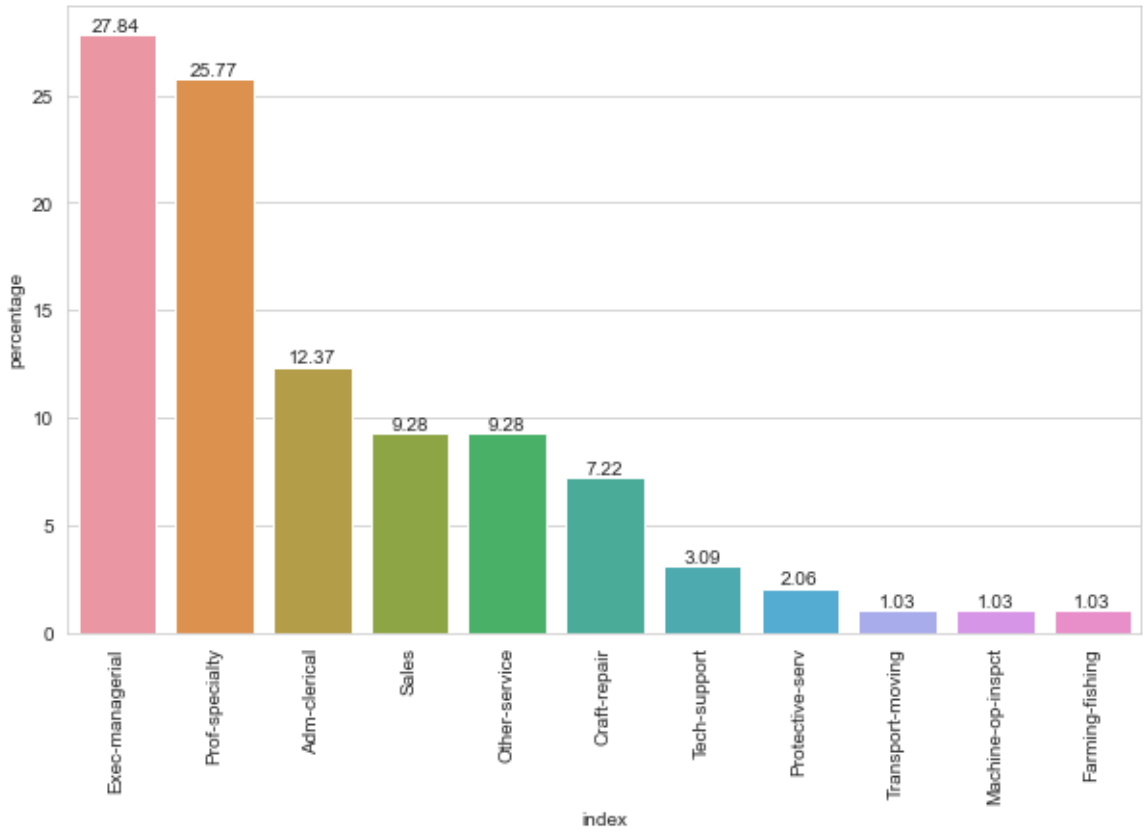
```
Exec-managerial     27.835
Prof-specialty      25.773
Adm-clerical        12.371
Sales                9.278
Other-service        9.278
Craft-repair         7.216
Tech-support         3.093
Protective-serv      2.062
Transport-moving     1.031
Machine-op-inspct    1.031
Farming-fishing      1.031
Name: occupation, dtype: float64
```

Desired Output: Exec-managerial 27.835 Prof-specialty 25.773 Adm-clerical 12.371 Sales 9.278 Other-service 9.278 Craft-repair 7.216 Tech-support 3.093 Protective-serv 2.062 Transport-moving 1.031 Machine-op-inspct

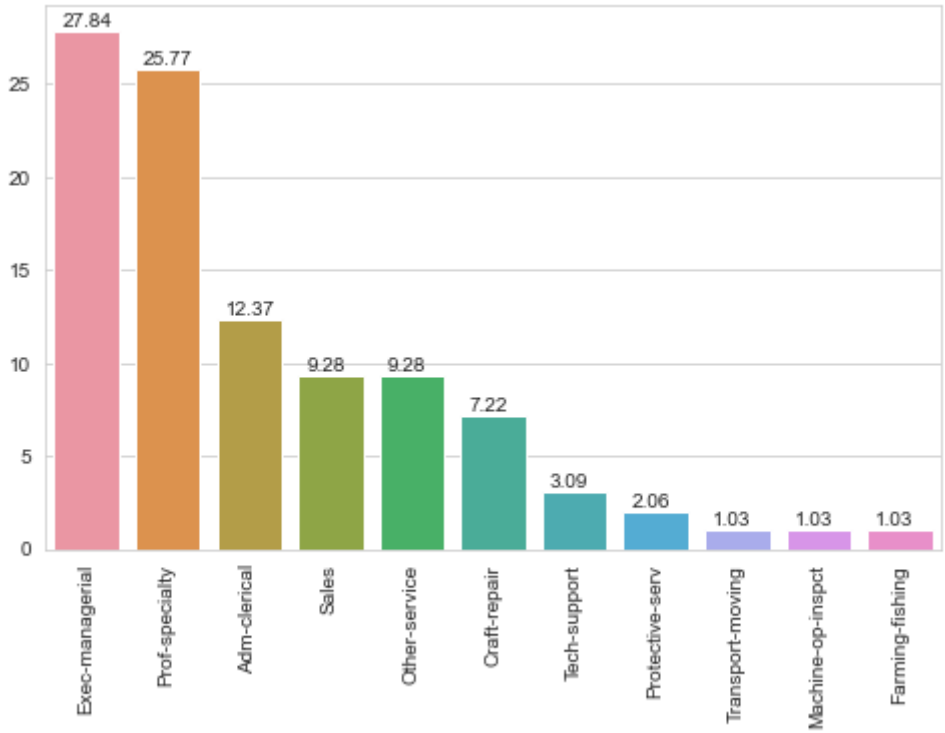
1.031 Farming-fishing 1.031 Name: occupation, dtype: float64

In [1197]:

```
# Your Code is Here
df_temp = (df[(df.salary == '>50K') & (df.race == 'Asian-Pac-Islander') & (df.ed
ucation == 'Bachelors')].occupation.value_counts(dropna=False, normalize = True)
* 100).rename('percentage').reset_index()
g = sns.barplot(x='index', y='percentage', data=df_temp)
plt.bar_label(g.containers[0], fmt='%0.2f')
plt.xticks(rotation=90);
```



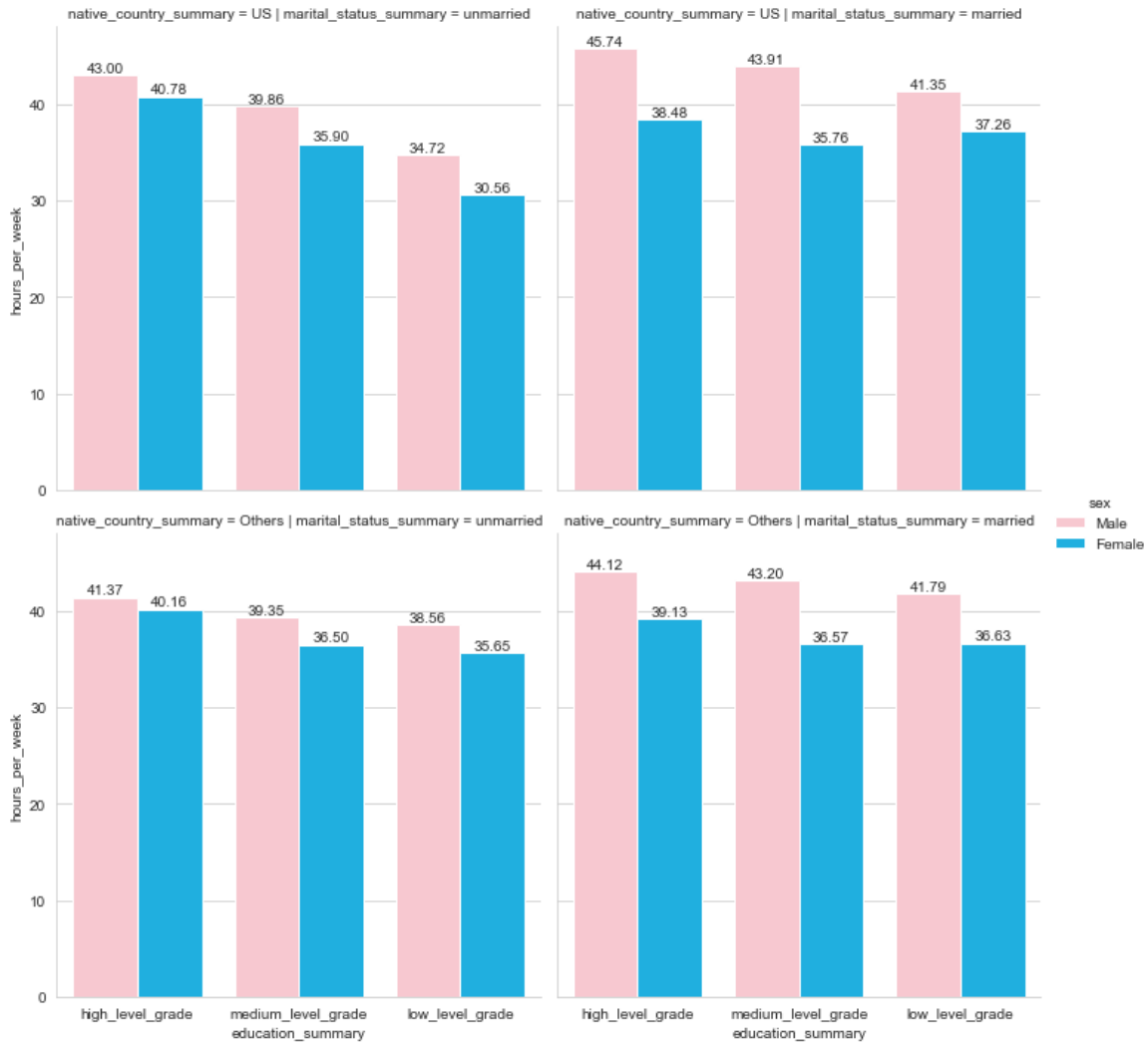
Desired Output:



6. What is the mean of working hours per week by gender for education level, workclass and marital status? Try to plot all required in one figure.

In [1198]:

```
# Your Code is Here
g = sns.catplot(row='native_country_summary', col='marital_status_summary', x=
'education_summary', y='hours_per_week', data=df, kind='bar', hue='sex', palette
=sns.color_palette(['pink','deepskyblue']), ci=None)
for ax in g.axes.ravel():
    for x in ax.containers:
        ax.bar_label(x, fmt='%0.2f');
```



Desired Output:

Dropping Similar & Unneccessary Features

Content

In [1199]:

```
# Your Code is Here
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32537 entries, 0 to 32560
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   age                     32537 non-null  int64  
 1   workclass               32537 non-null  object 
 2   fnlwgt                  32537 non-null  int64  
 3   education               32537 non-null  object 
 4   education_num           31735 non-null  float64
 5   marital_status          32537 non-null  object 
 6   occupation              32537 non-null  object 
 7   relationship            32537 non-null  object 
 8   race                    32537 non-null  object 
 9   sex                     32537 non-null  object 
 10  capital_gain            32537 non-null  int64  
 11  capital_loss            32537 non-null  int64  
 12  hours_per_week          32537 non-null  int64  
 13  native_country          32537 non-null  object 
 14  salary                  32537 non-null  object 
 15  education_summary       32537 non-null  object 
 16  marital_status_summary  32537 non-null  object 
 17  native_country_summary  32537 non-null  object 
dtypes: float64(1), int64(5), object(12)
memory usage: 5.7+ MB
```

Desired Output: Int64Index: 32537 entries, 0 to 32560 Data columns (total 18 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 age 32537 non-null int64 1 workclass 32537 non-null object 2 fnlwgt 32537 non-null int64 3 education 32537 non-null object 4 education_num 31735 non-null float64 5 marital_status 32537 non-null object 6 occupation 32537 non-null object 7 relationship 32537 non-null object 8 race 32537 non-null object 9 gender 32537 non-null object 10 capital_gain 32537 non-null int64 11 capital_loss 32537 non-null int64 12 hours_per_week 32537 non-null int64 13 native_country 32537 non-null object 14 salary 32537 non-null object 15 education_summary 32537 non-null object 16 marital_status_summary 32537 non-null object 17 native_country_summary 32537 non-null object dtypes: float64(1), int64(5), object(12) memory usage: 5.7+ MB

In [1200]:

```
# Drop the columns of "education", "education_num", "relationship", "marital_sta
tus", "native_country" permanently

# Your Code is Here
df = df.drop(["education", "education_num", "relationship", "marital_status", "n
ative_country"], axis = 1)
```

Handling with Missing Value

Content

Check the missing values for all features basically

In [1201]:

```
# Your Code is Here
df.isnull().sum()
```

Out[1201]:

```
age                        0
workclass                  0
fnlwgt                     0
occupation                 0
race                       0
sex                        0
capital_gain               0
capital_loss               0
hours_per_week             0
salary                     0
education_summary          0
marital_status_summary     0
native_country_summary     0
dtype: int64
```

Desired Output: age 0 workclass 0 fnlwgt 0 occupation 0 race 0 gender 0 capital_gain 0 capital_loss 0 hours_per_week 0 salary 0 education_summary 0 marital_status_summary 0 native_country_summary 0 dtype: int64

1. It seems that there is no missing value. But we know that "workclass", and "occupation" features have missing values as the "Unknown" string values. Examine these features in more detail.

2. Decide if drop these "Unknown" string values or not

In [1202]:

```
# Your Code is Here
df.workclass.value_counts()
```

Out[1202]:

```
Private             22673
Self-emp-not-inc     2540
Local-gov            2093
Unknown              1836
State-gov            1298
Self-emp-inc         1116
Federal-gov           960
Without-pay            14
Never-worked            7
Name: workclass, dtype: int64
```

Desired Output: Private 22673 Self-emp-not-inc 2540 Local-gov 2093 Unknown 1836 State-gov 1298 Self-emp-

inc 1116 Federal-gov 960 Without-pay 14 Never-worked 7 Name: workclass, dtype: int64

In [1203]:

```
# Your Code is Here
df.occupation.value_counts()
```

Out[1203]:

```
Prof-specialty       4136
Craft-repair         4094
Exec-managerial      4065
Adm-clerical         3768
Sales                3650
Other-service        3291
Machine-op-inspct    2000
Unknown              1843
Transport-moving     1597
Handlers-cleaners    1369
Farming-fishing       992
Tech-support          927
Protective-serv       649
Priv-house-serv       147
Armed-Forces            9
Name: occupation, dtype: int64
```

Desired Output: Prof-specialty 4136 Craft-repair 4094 Exec-managerial 4065 Adm-clerical 3768 Sales 3650 Other-service 3291 Machine-op-inspct 2000 Unknown 1843 Transport-moving 1597 Handlers-cleaners 1369 Farming-fishing 992 Tech-support 927 Protective-serv 649 Priv-house-serv 147 Armed-Forces 9 Name: occupation, dtype: int64

In [1204]:

```
# Your Code is Here
df[(df.occupation == 'Unknown') | (df.occupation == 'Never-worked')].workclass.v
alue_counts()
```

Out[1204]:

```
Unknown         1836
Never-worked       7
Name: workclass, dtype: int64
```

Desired Output: Unknown 1836 Never-worked 7 Name: workclass, dtype: int64

In [1205]:

```
# Replace "Unknown" values with NaN using numpy library

# Your Code is Here
df.occupation = df.occupation.replace({'Unknown':np.nan})
df.workclass = df.workclass.replace({'Unknown':np.nan})
```

In [1206]:

```
# Your Code is Here
df.isnull().sum()
```

Out[1206]:

```
age                          0
workclass                 1836
fnlwgt                       0
occupation                1843
race                         0
sex                          0
capital_gain                 0
capital_loss                 0
hours_per_week               0
salary                       0
education_summary            0
marital_status_summary       0
native_country_summary       0
dtype: int64
```

Desired Output: age 0 workclass 1836 fnlwgt 0 occupation 1843 race 0 gender 0 capital_gain 0 capital_loss 0 hours_per_week 0 salary 0 education_summary 0 marital_status_summary 0 native_country_summary 0 dtype: int64

In [1207]:

```
# Drop missing values in df permanently

# Your Code is Here
df.dropna(inplace=True)
```

In [1208]:

```
# Your Code is Here
df.isnull().sum()
```

Out[1208]:

```
age                          0
workclass                    0
fnlwgt                       0
occupation                   0
race                         0
sex                          0
capital_gain                 0
capital_loss                 0
hours_per_week               0
salary                       0
education_summary            0
marital_status_summary       0
native_country_summary       0
dtype: int64
```

Desired Output: age 0 workclass 0 fnlwgt 0 occupation 0 race 0 gender 0 capital_gain 0 capital_loss 0 hours_per_week 0 salary 0 education_summary 0 marital_status_summary 0 native_country_summary 0 dtype:

int64

In [1209]:

```
# Your Code is Here
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30694 entries, 0 to 32560
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   age                     30694 non-null  int64 
 1   workclass               30694 non-null  object
 2   fnlwgt                  30694 non-null  int64 
 3   occupation              30694 non-null  object
 4   race                    30694 non-null  object
 5   sex                     30694 non-null  object
 6   capital_gain            30694 non-null  int64 
 7   capital_loss            30694 non-null  int64 
 8   hours_per_week          30694 non-null  int64 
 9   salary                  30694 non-null  object
 10  education_summary       30694 non-null  object
 11  marital_status_summary  30694 non-null  object
 12  native_country_summary  30694 non-null  object
dtypes: int64(5), object(8)
memory usage: 3.3+ MB
```

Desired Output: Int64Index: 30694 entries, 0 to 32560 Data columns (total 13 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 age 30694 non-null int64 1 workclass 30694 non-null object 2 fnlwgt 30694 non-null int64 3 occupation 30694 non-null object 4 race 30694 non-null object 5 gender 30694 non-null object 6 capital_gain 30694 non-null int64 7 capital_loss 30694 non-null int64 8 hours_per_week 30694 non-null int64 9 salary 30694 non-null object 10 education_summary 30694 non-null object 11 marital_status_summary 30694 non-null object 12 native_country_summary 30694 non-null object dtypes: int64(5), object(8) memory usage: 3.3+ MB

Handling with Outliers

Boxplot and Histplot for all numeric features

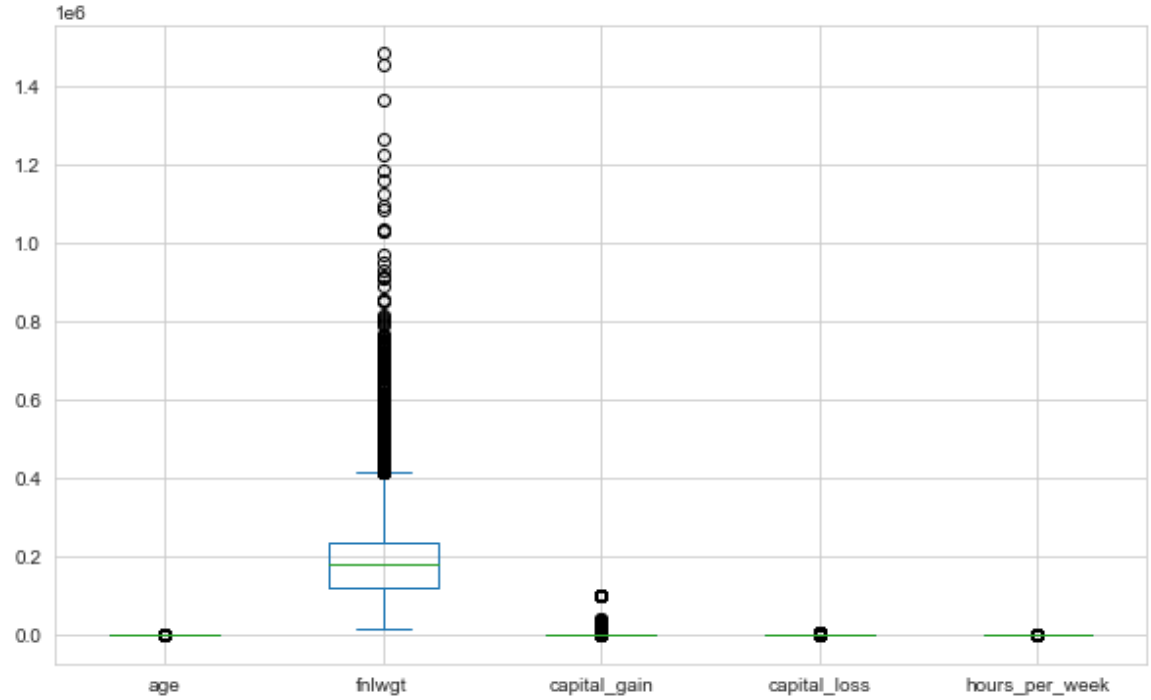
Plot boxplots for each numeric features at the same figure as subplots

In [1210]:

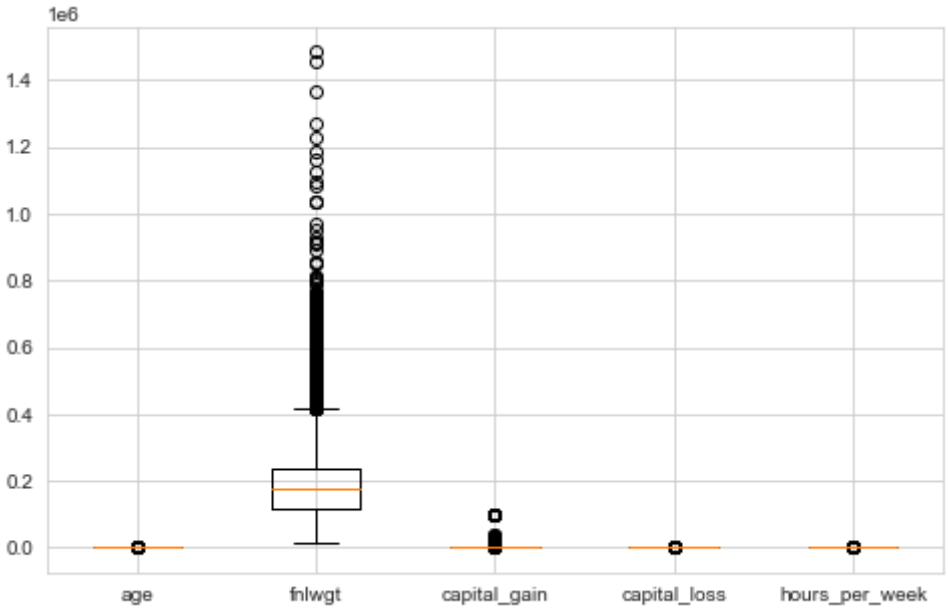
```
# Your Code is Here
df.plot.box()
```

Out[1210]:

<AxesSubplot:>

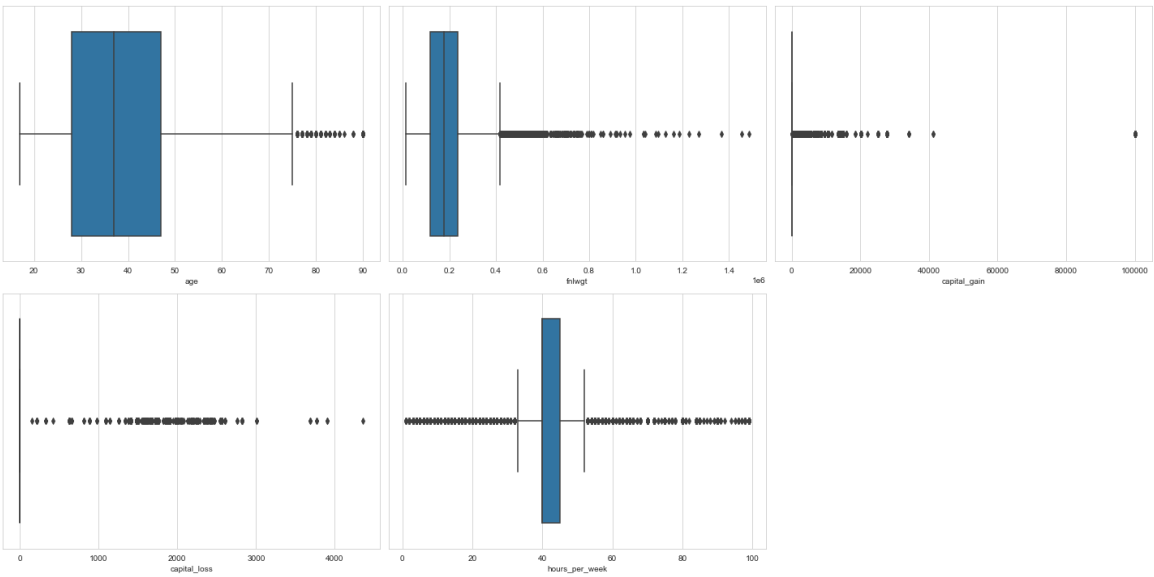


Desired Output:

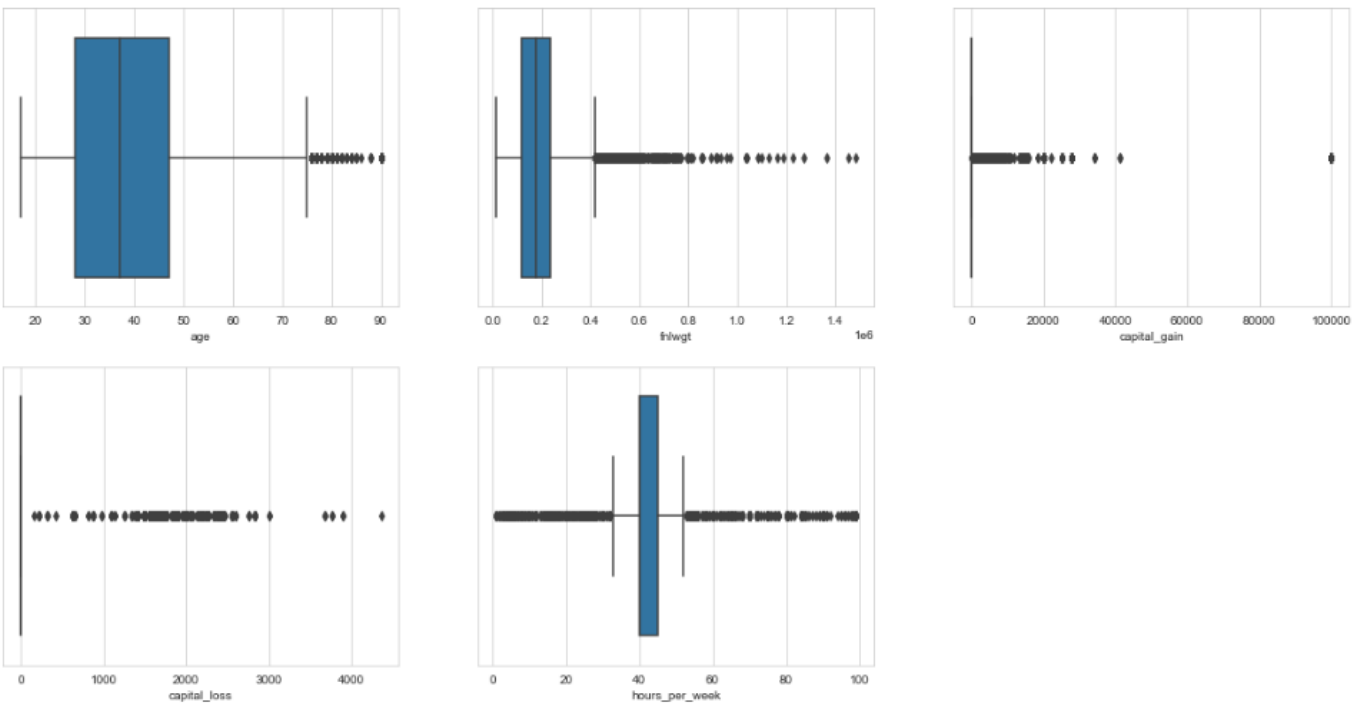


In [1211]:

```
# Your Code is Here
plt.figure(figsize=(20,10))
for count,x in enumerate(df.select_dtypes('number').columns.values):
    plt.subplot(2,3,count+ 1)
    sns.boxplot(x=x, data=df)
plt.tight_layout()
```



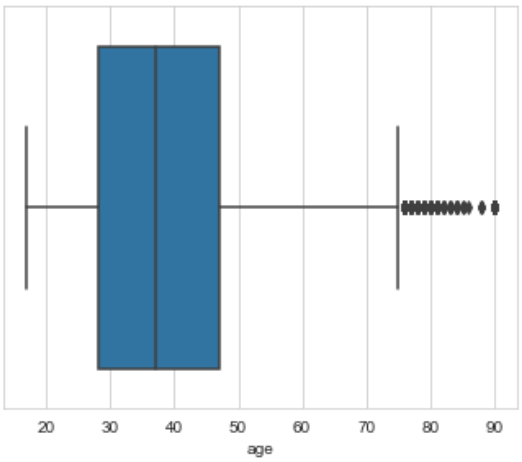
Desired Output:

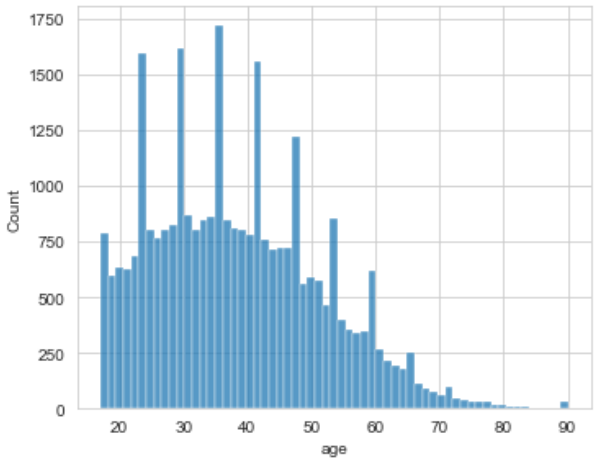


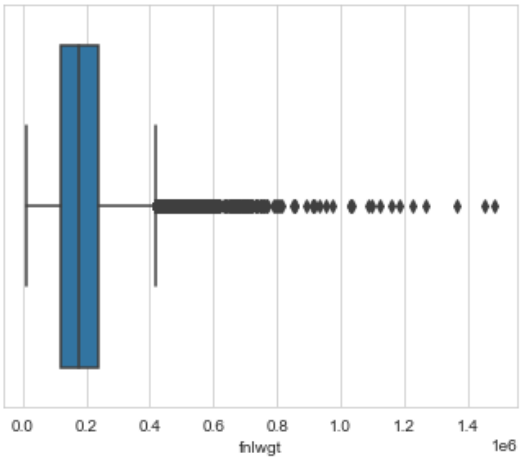
Plot both boxplots and histograms for each numeric features at the same figure as subplots

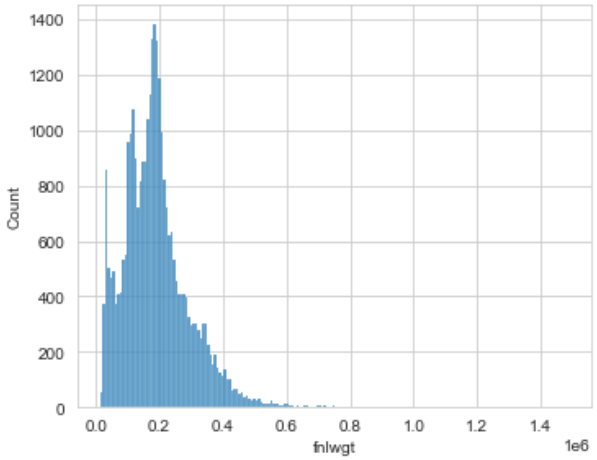
In [1212]:

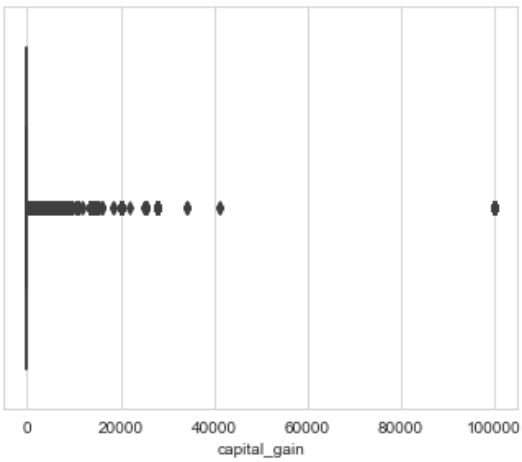
```
# Your Code is Here
# Your Code is Here
plt.figure(figsize=(10,20))
list1 = df.select_dtypes('number').columns.values
for count,x in enumerate(list1):
    plt.subplot(len(list1),2,(count +1)*2 - 1)
    sns.boxplot(x=x, data=df)
    plt.subplot(len(list1),2,(count +1)*2)
    sns.histplot(x=x, data=df)
plt.tight_layout()
```

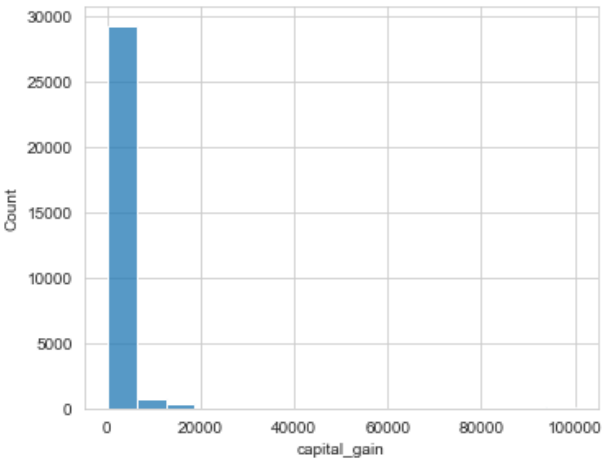

20
30
40
50
60
70
80
90
age

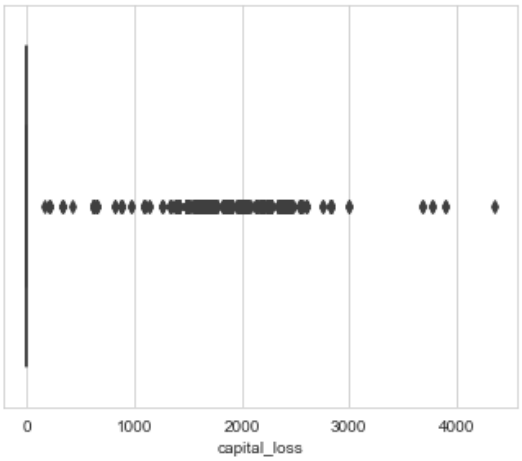

1750
1500
1250
1000
750
500
250
0
Count
20
30
40
50
60
70
80
90
age

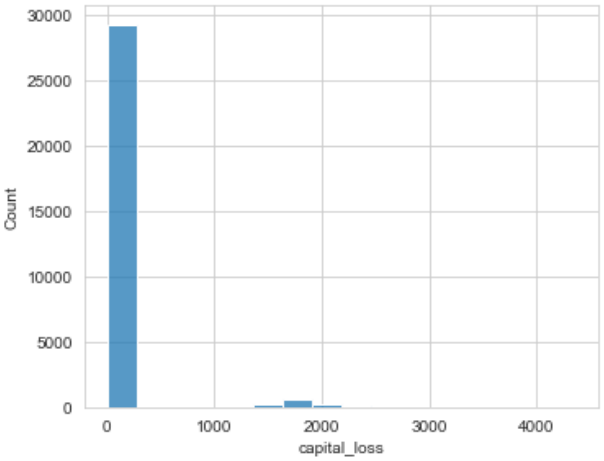

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
fnlwgt
1e6

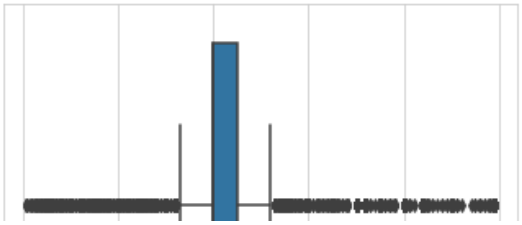

1400
1200
1000
800
600
400
200
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
fnlwgt
1e6

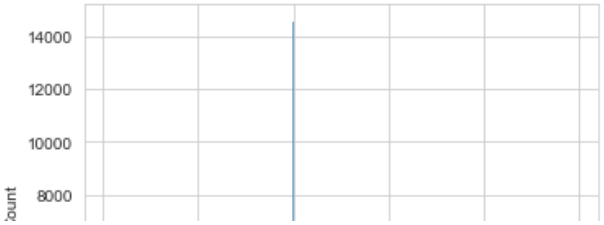

0
20000
40000
60000
80000
100000
capital_gain

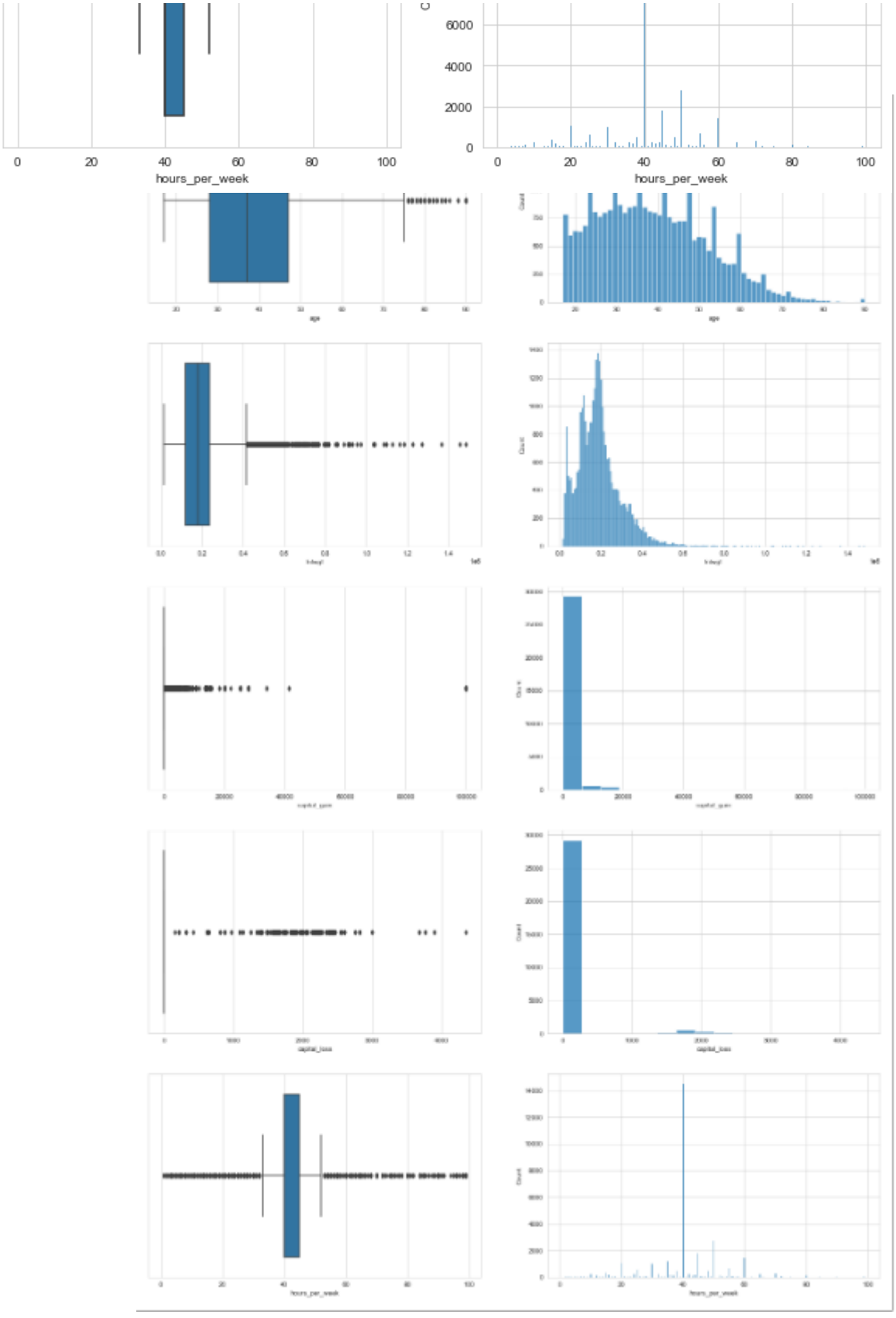

30000
25000
20000
15000
10000
5000
0
Count
0
20000
40000
60000
80000
100000
capital_gain


0
1000
2000
3000
4000
capital_loss


30000
25000
20000
15000
10000
5000
0
Count
0
1000
2000
3000
4000
capital_loss




14000
12000
10000
8000
Count



Check the statistical values for all numeric features

In [1213]:

```
df.describe().T
```

Out[1213]:

	count	mean	std	min	25%	50%	
age	30694.000	38.448	13.115	17.000	28.000	37.000	4
fnlwgt	30694.000	189848.229	105465.126	13769.000	117828.500	178513.500	23729
capital_gain	30694.000	1106.902	7500.730	0.000	0.000	0.000	
capital_loss	30694.000	88.980	405.808	0.000	0.000	0.000	
hours_per_week	30694.000	40.953	11.984	1.000	40.000	40.000	4

Desired Output:

	count	mean	std	min	25%	50%	75%	max
age	30694.000	38.448	13.115	17.000	28.000	37.000	47.000	90.000
fnlwgt	30694.000	189848.229	105465.126	13769.000	117828.500	178513.500	237293.750	1484705.000
capital_gain	30694.000	1106.902	7500.730	0.000	0.000	0.000	0.000	99999.000
capital_loss	30694.000	88.980	405.808	0.000	0.000	0.000	0.000	4356.000
hours_per_week	30694.000	40.953	11.984	1.000	40.000	40.000	45.000	99.000

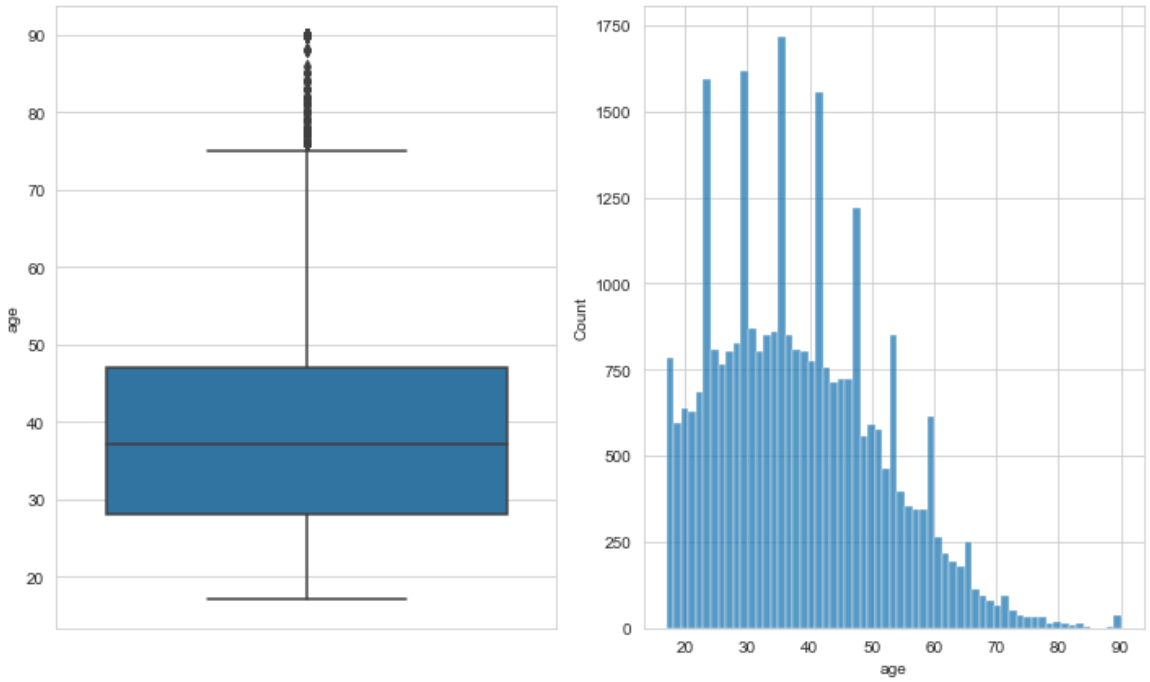
1. After analyzing all features, we have decided that we can't evaluate extreme values in "fnlwgt, capital_gain, capital_loss" features in the scope of outliers.

2. So let's examine "age and hours_per_week" features and detect extreme values which could be outliers by using IQR Rule.

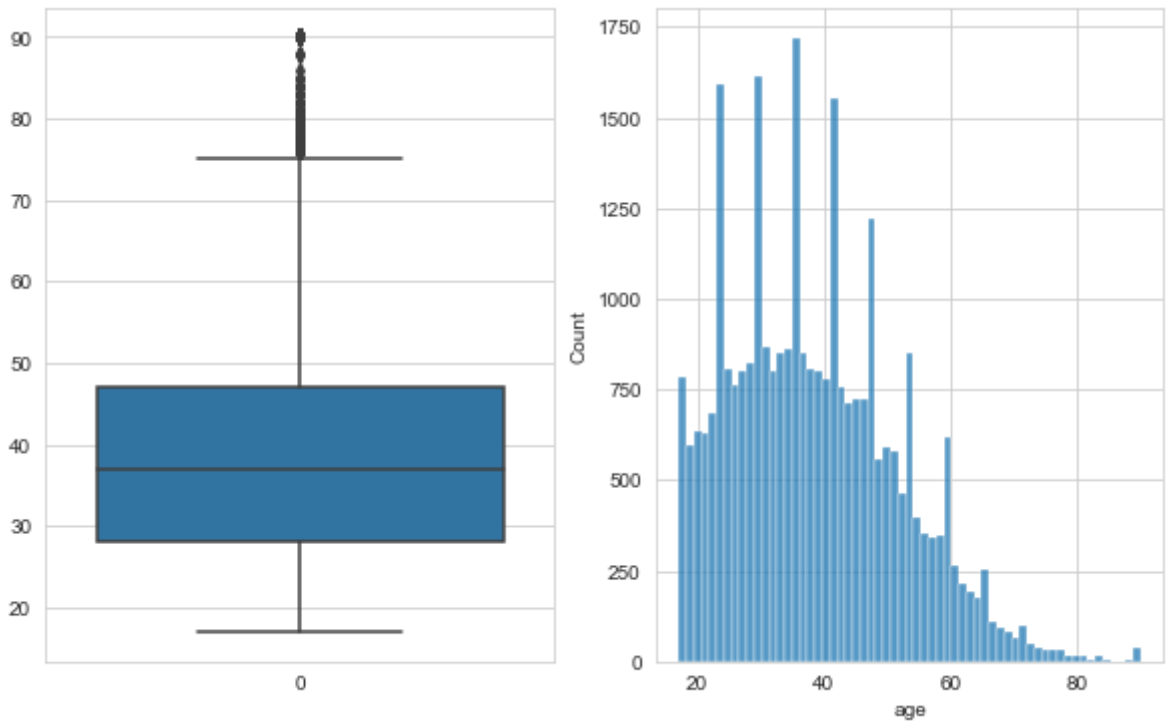
age

In [1214]:

```
# Your Code is Here
plt.subplot(1,2,1)
sns.boxplot(y='age', data=df)
plt.subplot(1,2,2)
sns.histplot(x='age', data=df)
plt.tight_layout()
```



Desired Output:



In [1215]:

```
# Find IQR defining quantile 0.25 for low level and 0.75 for high level

# Your Code is Here
from scipy.stats import stats
IQR = stats.iqr(df.age)
Q1 = np.percentile(df.age,25)
Q3 = np.percentile(df.age,75)
(Q1,Q3,IQR)
```

Out[1215]:

```
(28.0, 47.0, 19.0)
```

Desired Output: (28.0, 47.0, 19.0)

In [1216]:

```
# Find lower and upper limit using IQR

# Your Code is Here
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR
(lower_limit, upper_limit)
```

Out[1216]:

```
(-0.5, 75.5)
```

Desired Output: (-0.5, 75.5)

In [1217]:

```
# Your Code is Here
df[df.age > upper_limit].age.value_counts()
```

Out[1217]:

```
90    35
76    30
77    20
80    16
79    15
81    14
78    14
84     8
82     7
83     5
88     3
85     3
86     1
Name: age, dtype: int64
```

Desired Output: 90 35 76 30 77 20 80 16 79 15 81 14 78 14 84 8 82 7 83 5 88 3 85 3 86 1 Name: age, dtype:

int64

In [1218]:

```
# Define the observations whose age is greater than upper limit and sort these o
bservations by age in descending order

# Your Code is Here
df[df.age > upper_limit].sort_values(by='age', ascending=False)
```

Out[1218]:

	age	workclass	fnlwgt	occupation	race	sex	capital_gain	capital_loss	hours_
8806	90	Private	87372	Prof-specialty	White	Male	20051	0	
14159	90	Local-gov	187749	Adm-clerical	Asian-Pac-Islander	Male	0	0	
28463	90	Federal-gov	195433	Craft-repair	White	Male	0	0	
12975	90	Private	250832	Exec-managerial	White	Male	0	0	
11996	90	Private	40388	Exec-managerial	White	Male	0	0	
...	...	...	...	...	...	...	...	...	
15102	76	Local-gov	169133	Adm-clerical	White	Female	0	0	
19085	76	Private	125784	Exec-managerial	White	Male	0	0	
23515	76	Private	142535	Adm-clerical	White	Male	0	0	
8240	76	Self-emp-not-inc	225964	Sales	White	Male	0	0	
23354	76	Private	199949	Adm-clerical	White	Male	20051	0	

171 rows × 13 columns

Desired Output:

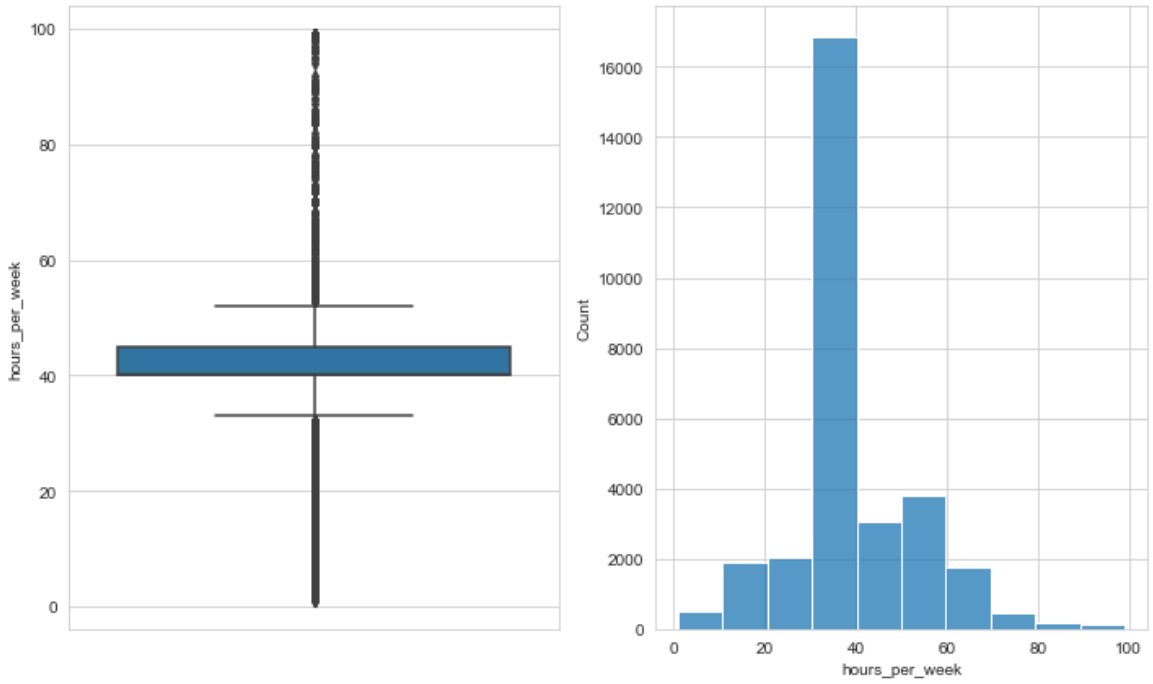
	age	workclass	fnlwgt	occupation	race	gender	capital_gain	capital_loss	hours_per_week	salary	education_summary	marital_status_summa
8806	90	Private	87372	Prof-specialty	White	Male	20051	0	72	>50K	high_level_grade	marri
14159	90	Local-gov	187749	Adm-clerical	Asian-Pac-Islander	Male	0	0	20	<=50K	medium_level_grade	marri
28463	90	Federal-gov	195433	Craft-repair	White	Male	0	0	30	<=50K	medium_level_grade	marri
12975	90	Private	250832	Exec-managerial	White	Male	0	0	40	<=50K	low_level_grade	marri
11996	90	Private	40388	Exec-managerial	White	Male	0	0	55	<=50K	high_level_grade	unmarri
...	...	...	...	...	...	...	...	...	...	...	...	
15102	76	Local-gov	169133	Adm-clerical	White	Female	0	0	30	<=50K	medium_level_grade	unmarri
19085	76	Private	125784	Exec-managerial	White	Male	0	0	40	<=50K	high_level_grade	marri
23515	76	Private	142535	Adm-clerical	White	Male	0	0	6	<=50K	medium_level_grade	unmarri
8240	76	Self-emp-not-inc	225964	Sales	White	Male	0	0	8	<=50K	medium_level_grade	unmarri
23354	76	Private	199949	Adm-clerical	White	Male	20051	0	50	>50K	high_level_grade	marri

171 rows × 13 columns

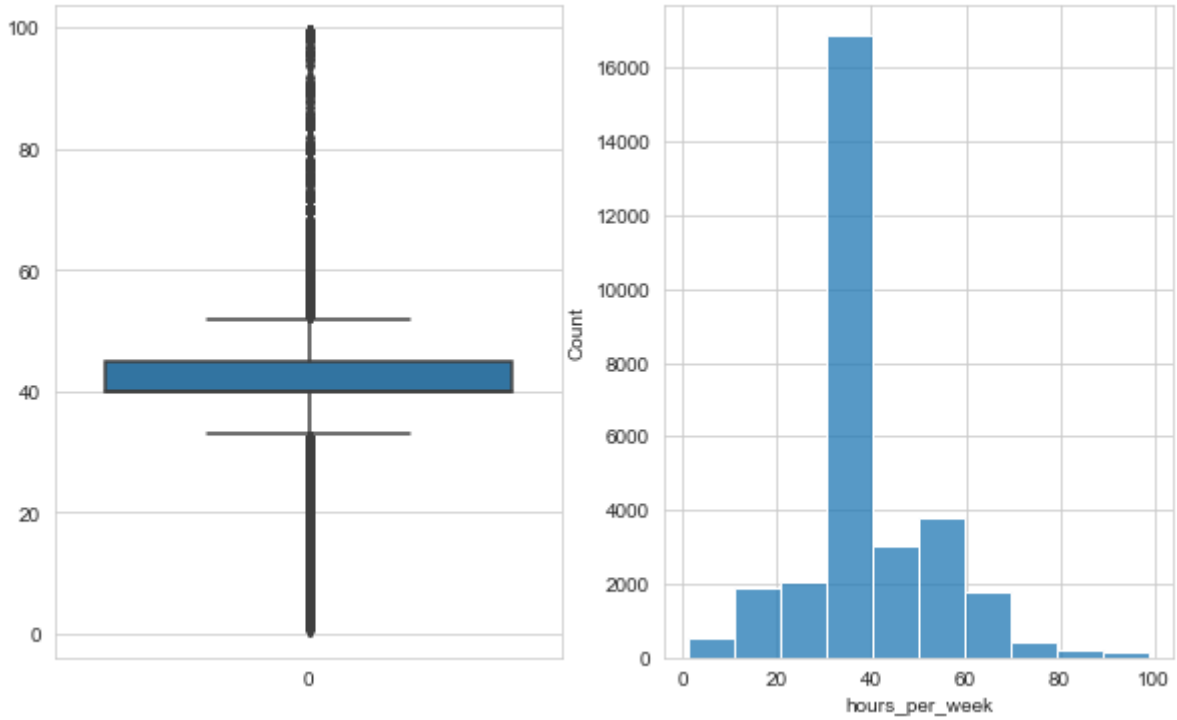
hours_per_week

In [1219]:

```
# Your Code is Here
plt.subplot(1,2,1)
sns.boxplot(y='hours_per_week', data=df)
plt.subplot(1,2,2)
sns.histplot(x='hours_per_week', data=df, bins= 10)
plt.tight_layout()
```



Desired Output:



In [1220]:

```
# Find IQR defining quantile 0.25 for low level and 0.75 for high level

# Your Code is Here
Q1 = np.percentile(df.hours_per_week,25)
Q3 = np.percentile(df.hours_per_week,75)
IQR = stats.iqr(df.hours_per_week)
(Q1,Q3,IQR)
```

Out[1220]:

```
(40.0, 45.0, 5.0)
```

Desired Output: (40.0, 45.0, 5.0)

In [1221]:

```
# Find the lower and upper limit using IQR

# Your Code is Here
lower_limit = Q1 - 1.5 * IQR
upper_limit = Q3 + 1.5 * IQR
(lower_limit, upper_limit)
```

Out[1221]:

```
(32.5, 52.5)
```

Desired Output: (32.5, 52.5)

In [1222]:

```
# Your Code is Here
df[df.hours_per_week > upper_limit].hours_per_week.value_counts().sort_index(asc
ending=False)
```

Out[1222]:

```
99      80
98      11
97       2
96       5
95       2
94       1
92       1
91       3
90      28
89       2
88       2
87       1
86       2
85      13
84      41
82       1
81       3
80     124
78       8
77       6
76       3
75      63
74       1
73       2
72      68
70     284
68      12
67       4
66      17
65     242
64      14
63      10
62      18
61       2
60    1441
59       5
58      27
57      17
56      91
55     683
54      39
53      23
Name: hours_per_week, dtype: int64
```

Desired Output: 99 80 98 11 97 2 96 5 95 2 94 1 92 1 91 3 90 28 89 2 88 2 87 1 86 2 85 13 84 41 82 1 81 3 80 124 78 8 77 6 76 3 75 63 74 1 73 2 72 68 70 284 68 12 67 4 66 17 65 242 64 14 63 10 62 18 61 2 60 1441 59 5

58 27 57 17 56 91 55 683 54 39 53 23 Name: hours_per_week, dtype: int64

In [1223]:

```
# Define the observations where  hours per week are greater than upper limit and
# sort these observations by hours per week in descending order

# Your Code is Here
df[df.hours_per_week > upper_limit].sort_values(by='hours_per_week', ascending=False)
```

Out[1223]:

	age	workclass	fnlwgt	occupation	race	sex	capital_gain	capital_loss	hours_p
25986	44	Local-gov	212665	Protective-serv	Black	Male	0	0	
19529	38	Private	66326	Transport-moving	White	Male	0	0	
8796	39	Private	70995	Transport-moving	White	Male	15024	0	
15534	29	Private	167716	Other-service	White	Female	0	0	
22313	26	Self-emp-not-inc	258306	Farming-fishing	White	Male	0	0	
...	...	...	...	...	...	...	...	...	
1138	27	Private	138705	Craft-repair	White	Male	0	0	
23244	58	Private	248739	Farming-fishing	White	Male	0	0	
28858	62	Private	123411	Transport-moving	White	Male	0	0	
18417	24	Private	117167	Other-service	White	Female	0	0	
31622	61	Self-emp-not-inc	268831	Sales	White	Male	0	0	

3402 rows × 13 columns

Desired Output:

	age	workclass	fnlwgt	occupation	race	gender	capital_gain	capital_loss	hours_per_week	salary	education_summary	marital_status_summary
25986	44	Local-gov	212665	Protective-serv	Black	Male	0	0	99	<=50K	medium_level_grade	marriec
19529	38	Private	66326	Transport-moving	White	Male	0	0	99	<=50K	medium_level_grade	marriec
8796	39	Private	70995	Transport-moving	White	Male	15024	0	99	>50K	high_level_grade	marriec
15534	29	Private	167716	Other-service	White	Female	0	0	99	<=50K	medium_level_grade	unmarriec
22313	26	Self-emp-not-inc	258306	Farming-fishing	White	Male	0	0	99	<=50K	low_level_grade	marriec
...	...	...	...	...	...	...	...	...	...	...	...	..
1138	27	Private	138705	Craft-repair	White	Male	0	0	53	<=50K	medium_level_grade	marriec
23244	58	Private	248739	Farming-fishing	White	Male	0	0	53	>50K	medium_level_grade	marriec
28858	62	Private	123411	Transport-moving	White	Male	0	0	53	<=50K	low_level_grade	marriec
18417	24	Private	117167	Other-service	White	Female	0	0	53	<=50K	medium_level_grade	unmarriec
31622	61	Self-emp-not-inc	268831	Sales	White	Male	0	0	53	<=50K	medium_level_grade	marriec

3402 rows × 13 columns

In [1224]:

```
# Your Code is Here
df[df.hours_per_week < lower_limit].hours_per_week.value_counts().sort_index()
```

Out[1224]:

```
1         8
2        15
3        24
4        28
5        39
6        40
7        20
8       103
9        17
10      223
11        9
12      143
13       19
14       28
15      350
16      182
17       27
18       64
19       14
20     1066
21       23
22       39
23       20
24      220
25      582
26       30
27       28
28       74
29        6
30     1009
31        5
32      239
Name: hours_per_week, dtype: int64
```

Desired Output: 1 8 2 15 3 24 4 28 5 39 6 40 7 20 8 103 9 17 10 223 11 9 12 143 13 19 14 28 15 350 16 182 17 27 18 64 19 14 20 1066 21 23 22 39 23 20 24 220 25 582 26 30 27 28 28 74 29 6 30 1009 31 5 32 239 Name: hours_per_week, dtype: int64

In [1225]:

```
# Your Code is Here
df[df.hours_per_week < lower_limit].groupby('salary').hours_per_week.describe()
```

Out[1225]:

	count	mean	std	min	25%	50%	75%	max
salary								
<=50K	4372.000	21.548	7.343	1.000	16.000	20.000	30.000	32.000
>50K	322.000	22.028	7.813	1.000	18.000	24.000	30.000	32.000

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	4372.000	21.548	7.343	1.000	16.000	20.000	30.000	32.000
>50K	322.000	22.028	7.813	1.000	18.000	24.000	30.000	32.000

In [1226]:

```
# Your Code is Here
df[df.hours_per_week < lower_limit].groupby('salary').age.describe()
```

Out[1226]:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	4372.000	34.052	17.730	17.000	20.000	26.000	45.000	90.000
>50K	322.000	49.484	13.894	22.000	39.000	48.000	60.000	90.000

Desired Output:

salary	count	mean	std	min	25%	50%	75%	max
<=50K	4372.000	34.052	17.730	17.000	20.000	26.000	45.000	90.000
>50K	322.000	49.484	13.894	22.000	39.000	48.000	60.000	90.000

Result : As we see, there are number of extreme values in both "age and hours_per_week" features. But how can we know if these extreme values are outliers or not? At this point, **domain knowledge** comes to the fore.

Domain Knowledge for this dataset:

1. In this dataset, all values are created according to the statements of individuals. So It can be some "data entries errors".
2. In addition, we have aimed to create an ML model with some restrictions as getting better performance from the ML model.
3. In this respect, our sample space ranges for some features are as follows.
    - **age : 17 to 80**
    - **hours_per_week : 7 to 70**
    - **if somebody's age is more than 60, he/she can't work more than 60 hours in a week**

Dropping rows according to the domain knownledge

In [1227]:

```
# Create a condition according to your domain knowledge on age stated above and
# sort the observations meeting this condition by age in ascending order

# Your Code is Here
df[~((df.age >= 17) & (df.age <= 80))].sort_values(by='age', ascending= False)
```

Out[1227]:

	age	workclass	fnlwgt	occupation	race	sex	capital_gain	capital_loss	hours_p
222	90	Private	51744	Other-service	Black	Male	0	2206	
18832	90	Private	115306	Exec-managerial	White	Female	0	0	
10545	90	Private	175491	Craft-repair	White	Male	9386	0	
11512	90	Private	87285	Other-service	White	Female	0	0	
11996	90	Private	40388	Exec-managerial	White	Male	0	0	
...	...	...	...	...	...	...	...	...	
19045	81	State-gov	132204	Other-service	White	Female	0	0	
6748	81	Private	122651	Sales	White	Male	0	0	
2906	81	Private	114670	Priv-house-serv	Black	Female	2062	0	
21501	81	Private	177408	Exec-managerial	White	Male	0	2377	
19495	81	Self-emp-inc	247232	Exec-managerial	White	Female	2936	0	

76 rows × 13 columns

Desired Output:

	age	workclass	fnlwgt	occupation	race	gender	capital_gain	capital_loss	hours_per_week	salary	education_summary	marital_status_summary
222	90	Private	51744	Other-service	Black	Male	0	2206	40	<=50K	medium_level_grade	unmarriec
18832	90	Private	115306	Exec-managerial	White	Female	0	0	40	<=50K	high_level_grade	unmarriec
10545	90	Private	175491	Craft-repair	White	Male	9386	0	50	>50K	medium_level_grade	marriec
11512	90	Private	87285	Other-service	White	Female	0	0	24	<=50K	medium_level_grade	unmarriec
11996	90	Private	40388	Exec-managerial	White	Male	0	0	55	<=50K	high_level_grade	unmarriec
...	...	...	...	...	...	...	...	...	...	...	...	..
19045	81	State-gov	132204	Other-service	White	Female	0	0	20	<=50K	low_level_grade	unmarriec
6748	81	Private	122651	Sales	White	Male	0	0	15	<=50K	medium_level_grade	marriec
2906	81	Private	114670	Priv-house-serv	Black	Female	2062	0	5	<=50K	low_level_grade	unmarriec
21501	81	Private	177408	Exec-managerial	White	Male	0	2377	26	>50K	medium_level_grade	marriec
19495	81	Self-emp-inc	247232	Exec-managerial	White	Female	2936	0	28	<=50K	low_level_grade	marriec

76 rows × 13 columns

In [1228]:

```
# Find the shape of the dataframe created by the condition defined above for age

# Your Code is Here

df[~((df.age >= 17) & (df.age <= 80))].sort_values(by='age', ascending= False).shape
```

Out[1228]:

```
(76, 13)
```

Desired Output: (76, 13)

In [1229]:

```
# Assign the indices of the rows defined in accordance with condition above for
age

# Your Code is Here

df[~((df.age >= 17) & (df.age <= 80))].sort_values(by='age', ascending= False).i
ndex
```

Out[1229]:

```
Int64Index([  222, 18832, 10545, 11512, 11996, 12975, 14159, 15892,
18277,
            18413, 18725, 19212,  8973, 19489, 19747, 20610, 22220,
24043,
            28463, 31030, 32277, 32367, 10210, 15356,  5370,  4070,
1040,
             6232,  1935,  2303,  5272,  6624,  2891,  5406,  8806,
1168,
            22895, 21835, 24027, 20463,  8381, 32459, 26731, 27795,
9471,
             6214, 14711, 11238,  7720, 15662,  7481, 24395, 23459,
19172,
            16302, 14756,  8431, 20421, 22481, 31855, 13696, 24280,
4834,
            29594, 28948, 12830,   918, 13295, 24560,  3537, 13928,
19045,
             6748,  2906, 21501, 19495],
           dtype='int64')
```

Desired Output: Int64Index([222, 18832, 10545, 11512, 11996, 12975, 14159, 15892, 18277, 18413, 18725, 19212, 8973, 19489, 19747, 20610, 22220, 24043, 28463, 31030, 32277, 32367, 10210, 15356, 5370, 4070, 1040, 6232, 1935, 2303, 5272, 6624, 2891, 5406, 8806, 1168, 22895, 21835, 24027, 20463, 8381, 32459, 26731, 27795, 9471, 6214, 14711, 11238, 7720, 15662, 7481, 24395, 23459, 19172, 16302, 14756, 8431, 20421, 22481, 31855, 13696, 24280, 4834, 29594, 28948, 12830, 918, 13295, 24560, 3537, 13928, 19045, 6748, 2906, 21501, 19495], dtype='int64')

In [1230]:

```
# Drop these indices defined above for age

# Your Code is Here
df.drop(df[~((df.age >= 17) & (df.age <= 80))].sort_values(by='age', ascending=
False).index, inplace= True)
```

In [1231]:

```
# Create a condition according to your domain knowledge on hours per week stated above and
# sort the observations meeting this condition by hours per week in descending order

# Your Code is Here
df[~((df.hours_per_week >= 7) & (df.hours_per_week <= 70))].sort_values(by='hours_per_week', ascending= False)
```

Out[1231]:

	age	workclass	fnlwgt	occupation	race	sex	capital_gain	capital_loss	hours_
22216	45	Private	54260	Craft-repair	White	Male	0	0	
5432	44	Private	83508	Prof-specialty	White	Female	2354	0	
19053	27	Private	40295	Transport-moving	White	Male	0	0	
19141	59	Private	106748	Other-service	White	Female	0	0	
19399	39	Self-emp-inc	163057	Craft-repair	White	Male	0	0	
...	...	...	...	...	...	...	...	...	
20909	77	Self-emp-not-inc	71676	Adm-clerical	White	Female	0	1944	
25078	74	Private	260669	Other-service	White	Female	0	0	
19750	23	Private	72887	Craft-repair	Asian-Pac-Islander	Male	0	0	
189	58	State-gov	109567	Prof-specialty	White	Male	0	0	
24284	57	Self-emp-not-inc	56480	Exec-managerial	White	Male	0	0	

621 rows × 13 columns

Desired Output:

	age	workclass	fnlwgt	occupation	race	gender	capital_gain	capital_loss	hours_per_week	salary	education_summary	marital_status_summa
22216	45	Private	54260	Craft-repair	White	Male	0	0	99	<=50K	medium_level_grade	unmarri
5432	44	Private	83508	Prof-specialty	White	Female	2354	0	99	<=50K	high_level_grade	unmarri
19053	27	Private	40295	Transport-moving	White	Male	0	0	99	<=50K	medium_level_grade	unmarri
19141	59	Private	106748	Other-service	White	Female	0	0	99	<=50K	low_level_grade	marri
19399	39	Self-emp-inc	163057	Craft-repair	White	Male	0	0	99	<=50K	medium_level_grade	unmarri
...	...	...	...	...	...	...	...	...	...	...	...	
20909	77	Self-emp-not-inc	71676	Adm-clerical	White	Female	0	1944	1	<=50K	medium_level_grade	unmarri
25078	74	Private	260669	Other-service	White	Female	0	0	1	<=50K	low_level_grade	unmarri
19750	23	Private	72887	Craft-repair	Asian-Pac-Islander	Male	0	0	1	<=50K	medium_level_grade	unmarri
189	58	State-gov	109567	Prof-specialty	White	Male	0	0	1	>50K	high_level_grade	marri
24284	57	Self-emp-not-inc	56480	Exec-managerial	White	Male	0	0	1	<=50K	medium_level_grade	marri

621 rows × 13 columns

In [1232]:

```
# Find the shape of the dataframe created by the condition defined above for hou
rs per week

# Your Code is Here

df[~((df.hours_per_week >= 7) & (df.hours_per_week <= 70))].sort_values(by='hour
s_per_week', ascending= False).shape
```

Out[1232]:

```
(621, 13)
```

Desired Output: (621, 13)

In [1233]:

```
# Assign the indices of the rows defined in accordance with condition above for
hours per week

# Your Code is Here
df[~((df.hours_per_week >= 7) & (df.hours_per_week <= 70))].sort_values(by='hour
s_per_week', ascending= False).index
```

Out[1233]:

```
Int64Index([22216,  5432, 19053, 19141, 19399, 19529, 19731, 19997,
20036,
            21056,
            ...
             6180, 29867,  1036, 11451, 22960, 20909, 25078, 19750,
189,
            24284],
           dtype='int64', length=621)
```

Desired Output: Int64Index([22216, 5432, 19053, 19141, 19399, 19529, 19731, 19997, 20036, 21056, ... 6180,

29867, 1036, 11451, 22960, 20909, 25078, 19750, 189, 24284], dtype='int64', length=621)

In [1234]:

```
# Drop these indices defined above for hours per week

# Your Code is Here
df.drop(df[~((df.hours_per_week >= 7) & (df.hours_per_week <= 70))].sort_values
(by='hours_per_week', ascending= False).index, inplace = True)
```

In [1235]:

```
# Create a condition according to your domain knowledge on both age and hours pe
r week stated above

# Your Code is Here
df[(df.age > 60) & (df.hours_per_week > 60)]
```

Out[1235]:

	age	workclass	fnlwgt	occupation	race	sex	capital_gain	capital_loss	hours_
1541	62	Private	162245	Prof-specialty	White	Male	0	1628	
2154	75	Private	101887	Priv-house-serv	White	Female	0	0	
2184	63	Self-emp-inc	54052	Sales	White	Male	0	0	
2665	70	Private	94692	Sales	White	Male	0	0	
3101	65	Self-emp-inc	81413	Farming-fishing	White	Male	0	2352	
5417	67	Private	197816	Sales	White	Male	0	1844	
6826	68	Private	284763	Transport-moving	White	Male	0	0	
8066	61	Self-emp-not-inc	133969	Sales	Asian-Pac-Islander	Male	0	0	
9646	62	Self-emp-not-inc	26911	Other-service	White	Female	0	0	
12624	63	Self-emp-inc	110890	Prof-specialty	White	Male	0	0	
16634	62	Self-emp-not-inc	115176	Farming-fishing	White	Male	0	0	
18367	70	Self-emp-not-inc	139889	Farming-fishing	White	Male	2653	0	
19584	64	Self-emp-not-inc	192695	Farming-fishing	White	Male	0	0	
20125	62	Private	252668	Prof-specialty	White	Male	0	0	
23399	63	Self-emp-not-inc	28612	Sales	White	Male	0	0	
23585	64	Private	212838	Sales	White	Male	0	0	
24903	61	Private	191417	Exec-managerial	Black	Male	0	0	
25910	66	Self-emp-inc	179951	Exec-managerial	White	Male	0	0	
26625	67	Self-emp-not-inc	152102	Farming-fishing	White	Male	0	0	
27721	61	Local-gov	28375	Prof-specialty	White	Female	0	0	
28294	65	Self-emp-inc	210381	Exec-managerial	White	Male	99999	0	
31342	62	Self-emp-not-inc	173631	Exec-managerial	White	Male	0	0	
32192	64	State-gov	104361	Adm-clerical	White	Female	0	0	

Desired Output:

	age	workclass	fnlwgt	occupation	race	gender	capital_gain	capital_loss	hours_per_week	salary	education_summary	marital_status_summa
1541	62	Private	162245	Prof-specialty	White	Male	0	1628	70	<=50K	high_level_grade	marri
2154	75	Private	101887	Priv-house-serv	White	Female	0	0	70	<=50K	low_level_grade	unmarri
2184	63	Self-emp-inc	54052	Sales	White	Male	0	0	68	>50K	high_level_grade	marri
2665	70	Private	94692	Sales	White	Male	0	0	70	>50K	high_level_grade	marri
3101	65	Self-emp-inc	81413	Farming-fishing	White	Male	0	2352	65	<=50K	medium_level_grade	marri
5417	67	Private	197816	Sales	White	Male	0	1844	70	<=50K	medium_level_grade	marri
6826	68	Private	284763	Transport-moving	White	Male	0	0	70	<=50K	low_level_grade	unmarri
8066	61	Self-emp-not-inc	133969	Sales	Asian-Pac-Islander	Male	0	0	63	<=50K	medium_level_grade	marri
9646	62	Self-emp-not-inc	26911	Other-service	White	Female	0	0	66	<=50K	low_level_grade	unmarri
12624	63	Self-emp-inc	110890	Prof-specialty	White	Male	0	0	70	>50K	high_level_grade	unmarri
16634	62	Self-emp-not-inc	115176	Farming-fishing	White	Male	0	0	65	<=50K	medium_level_grade	marri
18367	70	Self-emp-not-inc	139889	Farming-fishing	White	Male	2653	0	70	<=50K	medium_level_grade	marri
19584	64	Self-emp-not-inc	192695	Farming-fishing	White	Male	0	0	70	<=50K	low_level_grade	marri
20125	62	Private	252668	Prof-specialty	White	Male	0	0	70	<=50K	high_level_grade	unmarri
23399	63	Self-emp-not-inc	28612	Sales	White	Male	0	0	70	<=50K	medium_level_grade	unmarri
23585	64	Private	212838	Sales	White	Male	0	0	65	>50K	medium_level_grade	marri
24903	61	Private	191417	Exec-managerial	Black	Male	0	0	65	<=50K	low_level_grade	unmarri
25910	66	Self-emp-inc	179951	Exec-managerial	White	Male	0	0	70	<=50K	high_level_grade	marri
26625	67	Self-emp-not-inc	152102	Farming-fishing	White	Male	0	0	65	<=50K	medium_level_grade	unmarri
27721	61	Local-gov	28375	Prof-specialty	White	Female	0	0	70	<=50K	high_level_grade	unmarri
28294	65	Self-emp-inc	210381	Exec-managerial	White	Male	99999	0	65	>50K	high_level_grade	marri
31342	62	Self-emp-not-inc	173631	Exec-managerial	White	Male	0	0	70	<=50K	medium_level_grade	marri
32192	64	State-gov	104361	Adm-clerical	White	Female	0	0	65	<=50K	medium_level_grade	unmarri

In [1236]:

```
# Find the shape of the dataframe created by the condition defined above for both age and hours per week


# Your Code is Here
df[(df.age > 60) & (df.hours_per_week > 60)].shape
```

Out[1236]:

```
(23, 13)
```

Desired Output: (23, 13)

In [1237]:

```
# Assign the indices of the rows defined in accordance with condition above for
both age and hours per week

# Your Code is Here

df[(df.age > 60) & (df.hours_per_week > 60)].index
```

Out[1237]:

```
Int64Index([ 1541,  2154,  2184,  2665,  3101,  5417,  6826,  8066,
9646,
            12624, 16634, 18367, 19584, 20125, 23399, 23585, 24903,
25910,
            26625, 27721, 28294, 31342, 32192],
           dtype='int64')
```

Desired Output: Int64Index([1541, 2154, 2184, 2665, 3101, 5417, 6826, 8066, 9646, 12624, 16634, 18367, 19584, 20125, 23399, 23585, 24903, 25910, 26625, 27721, 28294, 31342, 32192], dtype='int64')

In [1238]:

```
# Drop these indices defined above for both age and hours per week

# Your Code is Here
df.drop(df[(df.age > 60) & (df.hours_per_week > 60)].index, inplace = True)
```

In [1239]:

```
# What is new shape of dataframe now

# Your Code is Here
df.shape
```

Out[1239]:

```
(29974, 13)
```

Desired Output: (29974, 13)

In [1240]:

```
# Reset the indices and take the head of DataFrame now

# Your Code is Here
df = df.reset_index(drop= True)
df.head()
```

Out[1240]:

	age	workclass	fnlwgt	occupation	race	sex	capital_gain	capital_loss	hours_per_w
0	39	State-gov	77516	Adm-clerical	White	Male	2174	0	
1	50	Self-emp-not-inc	83311	Exec-managerial	White	Male	0	0	
2	38	Private	215646	Handlers-cleaners	White	Male	0	0	
3	53	Private	234721	Handlers-cleaners	Black	Male	0	0	
4	28	Private	338409	Prof-specialty	Black	Female	0	0	

Desired Output:

	age	workclass	fnlwgt	occupation	race	gender	capital_gain	capital_loss	hours_per_week	salary	education_summary	marital_status_summary	na
0	39	State-gov	77516	Adm-clerical	White	Male	2174	0	40	<=50K	high_level_grade	unmarried	
1	50	Self-emp-not-inc	83311	Exec-managerial	White	Male	0	0	13	<=50K	high_level_grade	married	
2	38	Private	215646	Handlers-cleaners	White	Male	0	0	40	<=50K	medium_level_grade	unmarried	
3	53	Private	234721	Handlers-cleaners	Black	Male	0	0	40	<=50K	low_level_grade	married	
4	28	Private	338409	Prof-specialty	Black	Female	0	0	40	<=50K	high_level_grade	married	

Final Step to Make the Dataset Ready for ML Models

Content

1. Convert all features to numeric

Convert target feature (salary) to numeric (0 and 1) by using map function

In [1241]:

```
df_dropped = df.copy()
```

In [1242]:

```
# Your Code is Here
df.salary = df.salary.map({'<=50K':0,'>50K':1})
df.salary
```

Out[1242]:

```
0         0
1         0
2         0
3         0
4         0
         ..
29969     0
29970     1
29971     0
29972     0
29973     1
Name: salary, Length: 29974, dtype: int64
```

Desired Output: 0 0 1 0 2 0 3 0 4 0 .. 29969 0 29970 1 29971 0 29972 0 29973 1 Name: salary, Length: 29974, dtype: int64

In [1243]:

```
# Your Code is Here
df.salary.value_counts()
```

Out[1243]:

```
0    22524
1     7450
Name: salary, dtype: int64
```

Desired Output: 0 22524 1 7450 Name: salary, dtype: int64

Convert all features to numeric by using get_dummies function

In [1244]:

```
# Your Code is Here
df_dummy = pd.get_dummies(df, drop_first= True)
df_dummy
```

Out[1244]:

	age	fnlwgt	capital_gain	capital_loss	hours_per_week	salary	workclass_Local-gov	worl
0	39	77516	2174	0	40	0	0	
1	50	83311	0	0	13	0	0	
2	38	215646	0	0	40	0	0	
3	53	234721	0	0	40	0	0	
4	28	338409	0	0	40	0	0	
...	...	...	...	...	...	...	...	
29969	27	257302	0	0	38	0	0	
29970	40	154374	0	0	40	1	0	
29971	58	151910	0	0	40	0	0	
29972	22	201490	0	0	20	0	0	
29973	52	287927	15024	0	40	1	0	

29974 rows × 34 columns

Desired Output:

	age	fnlwgt	capital_gain	capital_loss	hours_per_week	salary	workclass_Local-gov	workclass_Private	workclass_Self-emp-inc	workclass_Self-emp-not-inc	workclass_St
0	39	77516	2174	0	40	0	0	0	0	0	
1	50	83311	0	0	13	0	0	0	0	1	
2	38	215646	0	0	40	0	0	1	0	0	
3	53	234721	0	0	40	0	0	1	0	0	
4	28	338409	0	0	40	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	
29969	27	257302	0	0	38	0	0	1	0	0	
29970	40	154374	0	0	40	1	0	1	0	0	
29971	58	151910	0	0	40	0	0	1	0	0	
29972	22	201490	0	0	20	0	0	1	0	0	
29973	52	287927	15024	0	40	1	0	0	1	0	

29974 rows × 34 columns

In [1245]:

```
# What's the shape of dataframe

# Your Code is Here
df.shape
```

Out[1245]:

```
(29974, 13)
```

Desired Output: (29974, 13)

In [1246]:

```
# What's the shape of dataframe created by dummy operation

# Your Code is Here
df_dummy.shape
```

Out[1246]:

```
(29974, 34)
```

Desired Output: (29974, 34)

2. Take a look at correlation between features by utilizing power of visualizing

In [1247]:

```
# Your Code is Here
df_dummy.corr()
```

Out[1247]:

	age	fnlwgt	capital_gain	capital_loss	hours_per_w
age	1.000	-0.074	0.082	0.059	0.
fnlwgt	-0.074	1.000	-0.002	-0.011	-0.
capital_gain	0.082	-0.002	1.000	-0.032	0.
capital_loss	0.059	-0.011	-0.032	1.000	0.
hours_per_week	0.119	-0.019	0.086	0.060	1.
salary	0.253	-0.010	0.222	0.152	0.
workclass_Local-gov	0.070	-0.004	-0.010	0.015	0.
workclass_Private	-0.211	0.045	-0.045	-0.039	-0.
workclass_Self-emp-inc	0.111	-0.025	0.093	0.031	0.
workclass_Self-emp-not-inc	0.146	-0.037	0.033	0.023	0.
workclass_State-gov	0.020	-0.012	-0.013	-0.002	-0.
workclass_Without-pay	0.016	-0.003	-0.002	-0.005	-0.
occupation_Armed-Forces	-0.011	0.004	-0.003	0.005	0.
occupation_Craft-repair	0.022	0.009	-0.023	0.000	0.
occupation_Exec-managerial	0.111	-0.020	0.061	0.048	0.
occupation_Farming-fishing	0.035	-0.031	-0.012	-0.013	0.
occupation_Handlers-cleaners	-0.104	0.030	-0.024	-0.023	-0.
occupation_Machine-op-inspct	-0.012	0.013	-0.027	-0.018	0.
occupation_Other-service	-0.094	-0.003	-0.042	-0.043	-0.
occupation_Priv-house-serv	0.011	0.008	-0.007	-0.011	-0.
occupation_Prof-specialty	0.065	-0.019	0.086	0.046	0.
occupation_Protective-serv	0.006	0.016	-0.008	-0.002	0.
occupation_Sales	-0.033	0.003	0.008	0.007	0.
occupation_Tech-support	-0.017	0.004	-0.010	0.004	-0.
occupation_Transport-moving	0.034	0.002	-0.019	-0.004	0.
race_Asian-Pac-Islander	-0.010	-0.049	0.010	0.004	-0.
race_Black	-0.015	0.118	-0.020	-0.024	-0.
race_Other	-0.032	0.006	-0.001	-0.008	-0.
race_White	0.028	-0.059	0.014	0.024	0.
sex_Male	0.083	0.028	0.047	0.048	0.
education_summary_low_level_grade	-0.010	0.042	-0.041	-0.033	-0.
education_summary_medium_level_grade	-0.079	-0.010	-0.084	-0.052	-0.
marital_status_summary_unmarried	-0.318	0.026	-0.084	-0.081	-0.

	age	fnlwgt	capital_gain	capital_loss	hours_per_w
native_country_summary_US	0.013	-0.077	0.006	0.010	0.

Desired Output:

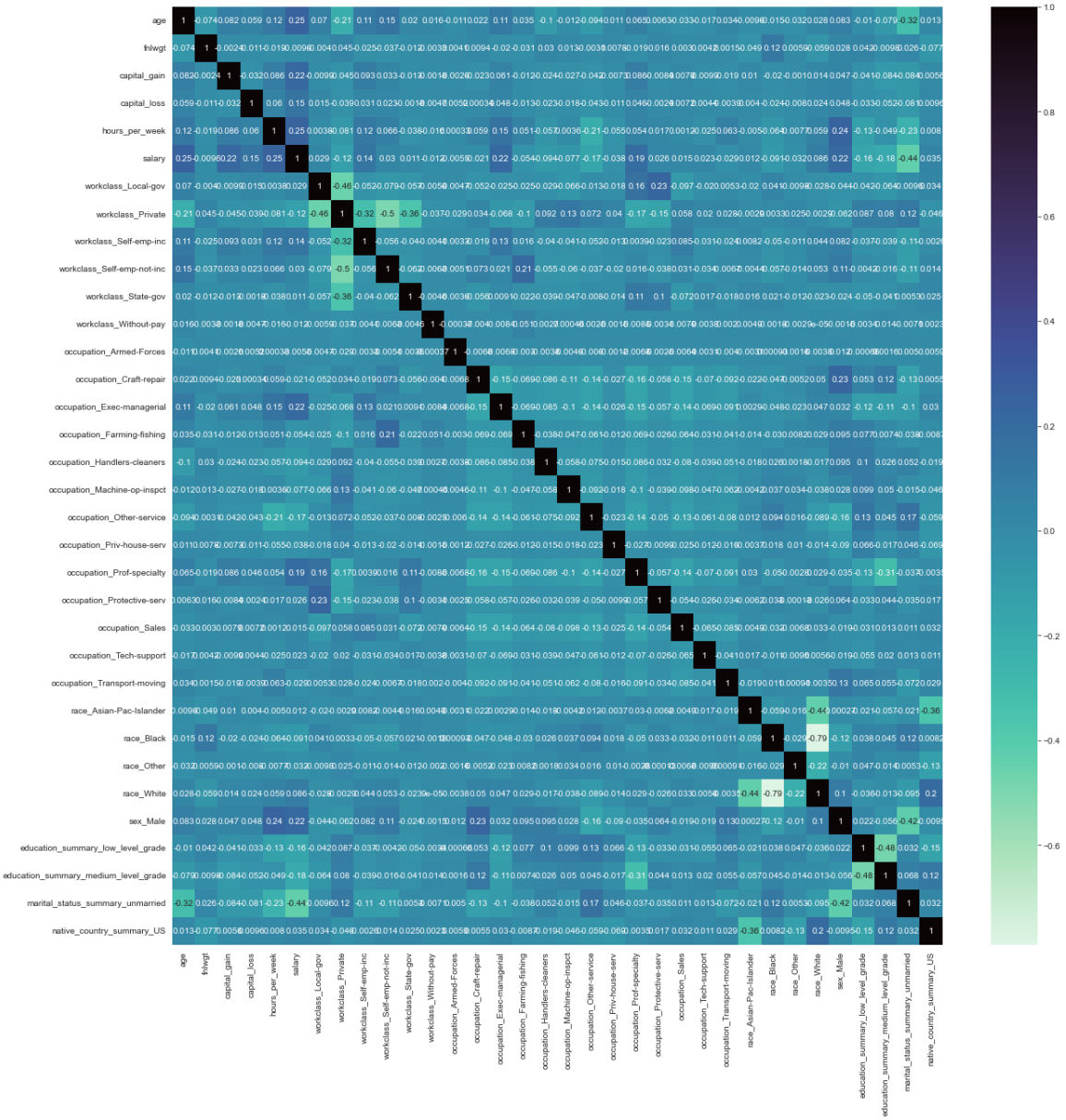
	age	fnlwgt	capital_gain	capital_loss	hours_per_week	salary	workclass_Local-gov	workclass_Private	workclass_ em
age	1.000	-0.074	0.082	0.059	0.119	0.253	0.070	-0.211	
fnlwgt	-0.074	1.000	-0.002	-0.011	-0.019	-0.010	-0.004	0.045	-(
capital_gain	0.082	-0.002	1.000	-0.032	0.086	0.222	-0.010	-0.045	(
capital_loss	0.059	-0.011	-0.032	1.000	0.060	0.152	0.015	-0.039	(
hours_per_week	0.119	-0.019	0.086	0.060	1.000	0.247	0.004	-0.081	(
salary	0.253	-0.010	0.222	0.152	0.247	1.000	0.029	-0.121	(
workclass_Local-gov	0.070	-0.004	-0.010	0.015	0.004	0.029	1.000	-0.461	-(
workclass_Private	-0.211	0.045	-0.045	-0.039	-0.081	-0.121	-0.461	1.000	-(
workclass_Self-emp-inc	0.111	-0.025	0.093	0.031	0.124	0.138	-0.052	-0.324	
workclass_Self-emp-not-inc	0.146	-0.037	0.033	0.023	0.066	0.030	-0.079	-0.497	-(
workclass_State-gov	0.020	-0.012	-0.013	-0.002	-0.038	0.011	-0.057	-0.358	-(
workclass_Without-pay	0.016	-0.003	-0.002	-0.005	-0.016	-0.012	-0.006	-0.037	-(
occupation_Armed-Forces	-0.011	0.004	-0.003	0.005	0.000	-0.006	-0.005	-0.029	-(
occupation_Craft-repair	0.022	0.009	-0.023	0.000	0.059	-0.021	-0.052	0.034	-(
occupation_Exec-managerial	0.111	-0.020	0.061	0.048	0.153	0.215	-0.025	-0.068	(
occupation_Farming-fishing	0.035	-0.031	-0.012	-0.013	0.051	-0.054	-0.025	-0.103	(
occupation_Handlers-cleaners	-0.104	0.030	-0.024	-0.023	-0.057	-0.094	-0.029	0.092	-(
occupation_Machine-op-inspct	-0.012	0.013	-0.027	-0.018	0.004	-0.077	-0.066	0.130	-(
occupation_Other-service	-0.094	-0.003	-0.042	-0.043	-0.205	-0.166	-0.013	0.072	-(
occupation_Priv-house-serv	0.011	0.008	-0.007	-0.011	-0.055	-0.038	-0.018	0.040	-(
occupation_Prof-specialty	0.065	-0.019	0.086	0.046	0.054	0.185	0.162	-0.167	(
occupation_Protective-serv	0.006	0.016	-0.008	-0.002	0.017	0.026	0.230	-0.150	-(
occupation_Sales	-0.033	0.003	0.008	0.007	0.001	0.015	-0.097	0.058	(
occupation_Tech-support	-0.017	0.004	-0.010	0.004	-0.025	0.023	-0.020	0.020	-(
occupation_Transport-moving	0.034	0.002	-0.019	-0.004	0.063	-0.029	0.005	0.028	-(
race_Asian-Pac-Islander	-0.010	-0.049	0.010	0.004	-0.005	0.012	-0.020	-0.003	(
race_Black	-0.015	0.118	-0.020	-0.024	-0.064	-0.091	0.041	0.003	-(
race_Other	-0.032	0.006	-0.001	-0.008	-0.008	-0.032	-0.010	0.025	-(
race_White	0.028	-0.059	0.014	0.024	0.059	0.086	-0.028	-0.003	(
gender_Male	0.083	0.028	0.047	0.048	0.239	0.216	-0.044	-0.062	(
education_summary_low_level_grade	-0.010	0.042	-0.041	-0.033	-0.134	-0.162	-0.042	0.087	-(
education_summary_medium_level_grade	-0.079	-0.010	-0.084	-0.052	-0.049	-0.183	-0.064	0.080	-(
marital_status_summary_unmarried	-0.318	0.026	-0.084	-0.081	-0.234	-0.438	-0.010	0.122	-(
native_country_summary_US	0.013	-0.077	0.006	0.010	0.008	0.035	0.034	-0.046	-(

In [1248]:

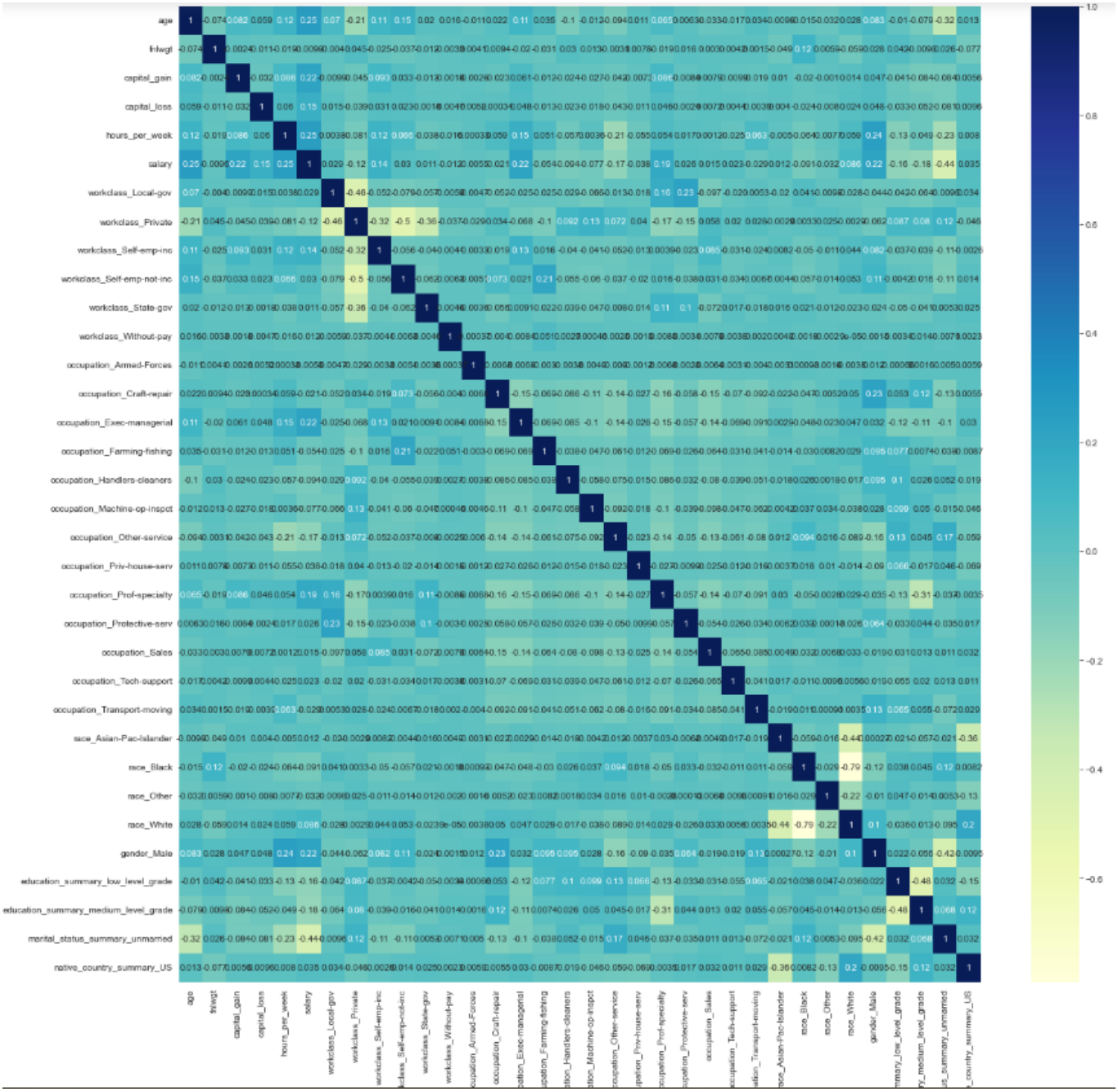
```
# Your Code is Here
plt.figure(figsize=(20,20))
sns.heatmap(df_dummy.corr(), cmap='mako_r', annot=True)
```

Out[1248]:

<AxesSubplot:>



Desired Output:

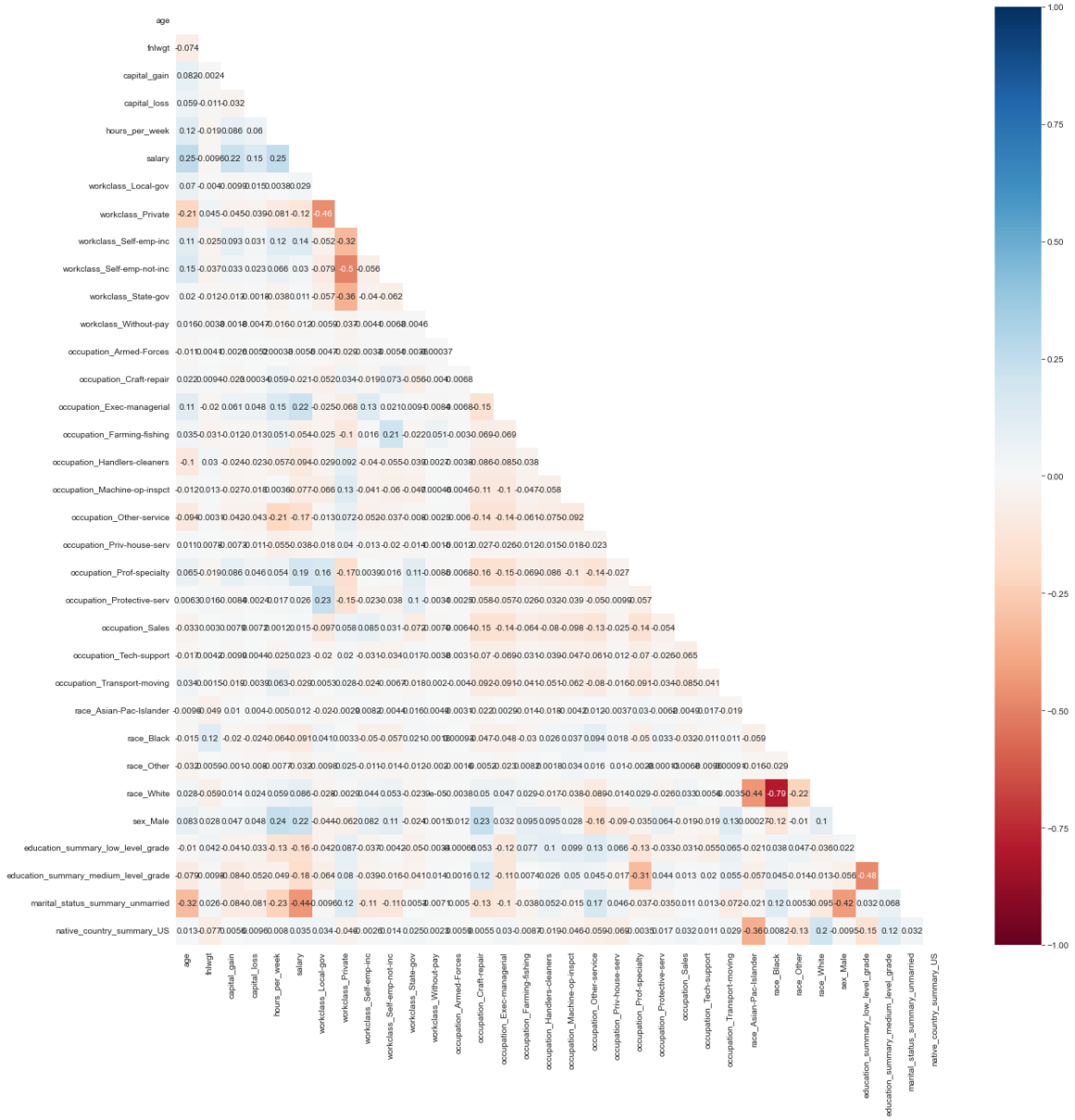


In [1249]:

```
plt.figure(figsize=(20,20))
mask = np.triu(df_dummy.corr())
sns.heatmap(df_dummy.corr(), annot=True, mask=mask, vmin=-1, vmax=1,cmap='RdBu')
```

Out[1249]:

<AxesSubplot:>



In [1250]:

```
# Your Code is Here
df_dummy.corr().salary.to_frame().drop('salary').sort_values(by='salary', ascend
ing=False)
```

Out[1250]:

	salary
age	0.253
hours_per_week	0.247
capital_gain	0.222
sex_Male	0.216
occupation_Exec-managerial	0.215
occupation_Prof-specialty	0.185
capital_loss	0.152
workclass_Self-emp-inc	0.138
race_White	0.086
native_country_summary_US	0.035
workclass_Self-emp-not-inc	0.030
workclass_Local-gov	0.029
occupation_Protective-serv	0.026
occupation_Tech-support	0.023
occupation_Sales	0.015
race_Asian-Pac-Islander	0.012
workclass_State-gov	0.011
occupation_Armed-Forces	-0.006
fnlwgt	-0.010
workclass_Without-pay	-0.012
occupation_Craft-repair	-0.021
occupation_Transport-moving	-0.029
race_Other	-0.032
occupation_Priv-house-serv	-0.038
occupation_Farming-fishing	-0.054
occupation_Machine-op-inspct	-0.077
race_Black	-0.091
occupation_Handlers-cleaners	-0.094
workclass_Private	-0.121
education_summary_low_level_grade	-0.162
occupation_Other-service	-0.166
education_summary_medium_level_grade	-0.183
marital_status_summary_unmarried	-0.438

Desired Output:

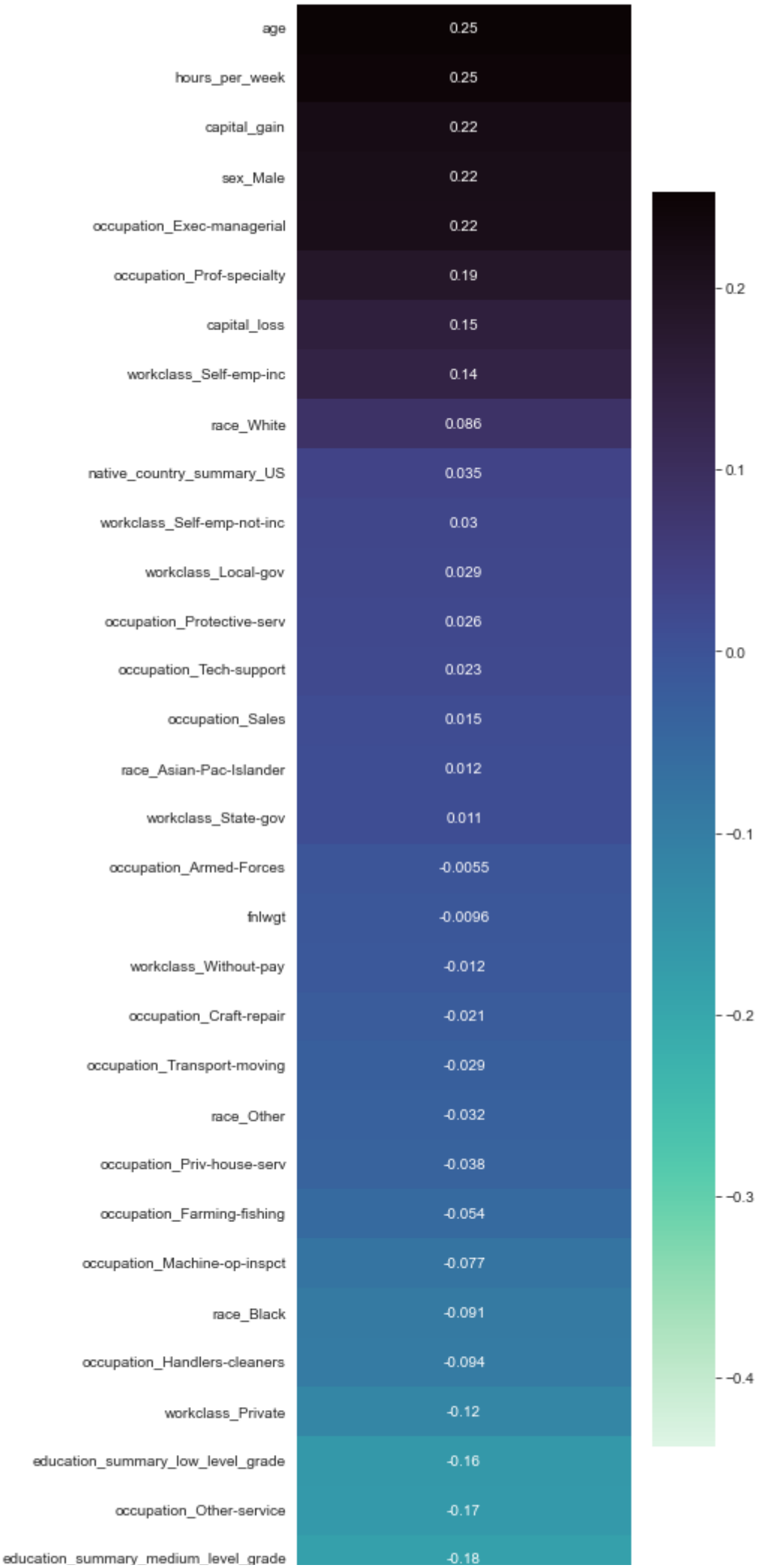
	salary
age	0.253
hours_per_week	0.247
capital_gain	0.222
gender_Male	0.216
occupation_Exec-managerial	0.215
occupation_Prof-specialty	0.185
capital_loss	0.152
workclass_Self-emp-inc	0.138
race_White	0.086
native_country_summary_US	0.035
workclass_Self-emp-not-inc	0.030
workclass_Local-gov	0.029
occupation_Protective-serv	0.026
occupation_Tech-support	0.023
occupation_Sales	0.015
race_Asian-Pac-Islander	0.012
workclass_State-gov	0.011
occupation_Armed-Forces	-0.006
fnlwgt	-0.010
workclass_Without-pay	-0.012
occupation_Craft-repair	-0.021
occupation_Transport-moving	-0.029
race_Other	-0.032
occupation_Priv-house-serv	-0.038
occupation_Farming-fishing	-0.054
occupation_Machine-op-inspct	-0.077
race_Black	-0.091
occupation_Handlers-cleaners	-0.094
workclass_Private	-0.121
education_summary_low_level_grade	-0.162
occupation_Other-service	-0.166
education_summary_medium_level_grade	-0.183
marital_status_summary_unmarried	-0.438

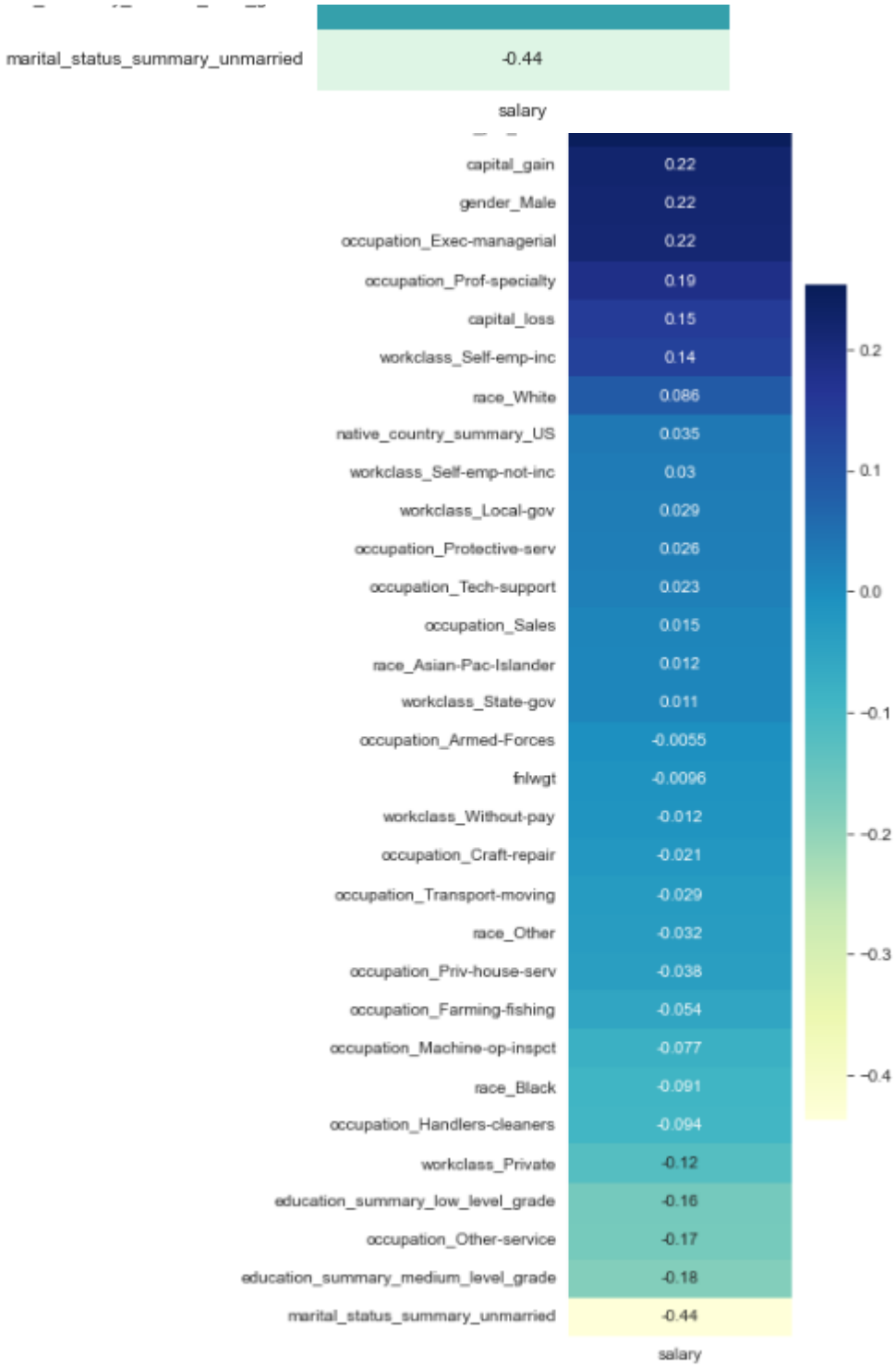
In [1251]:

```
plt.figure(figsize= (5,20))
sns.heatmap(df_dummy.corr().salary.to_frame().drop('salary').sort_values(by='sal
ary', ascending=False), cmap = 'mako_r', annot = True)
```

Out[1251]:

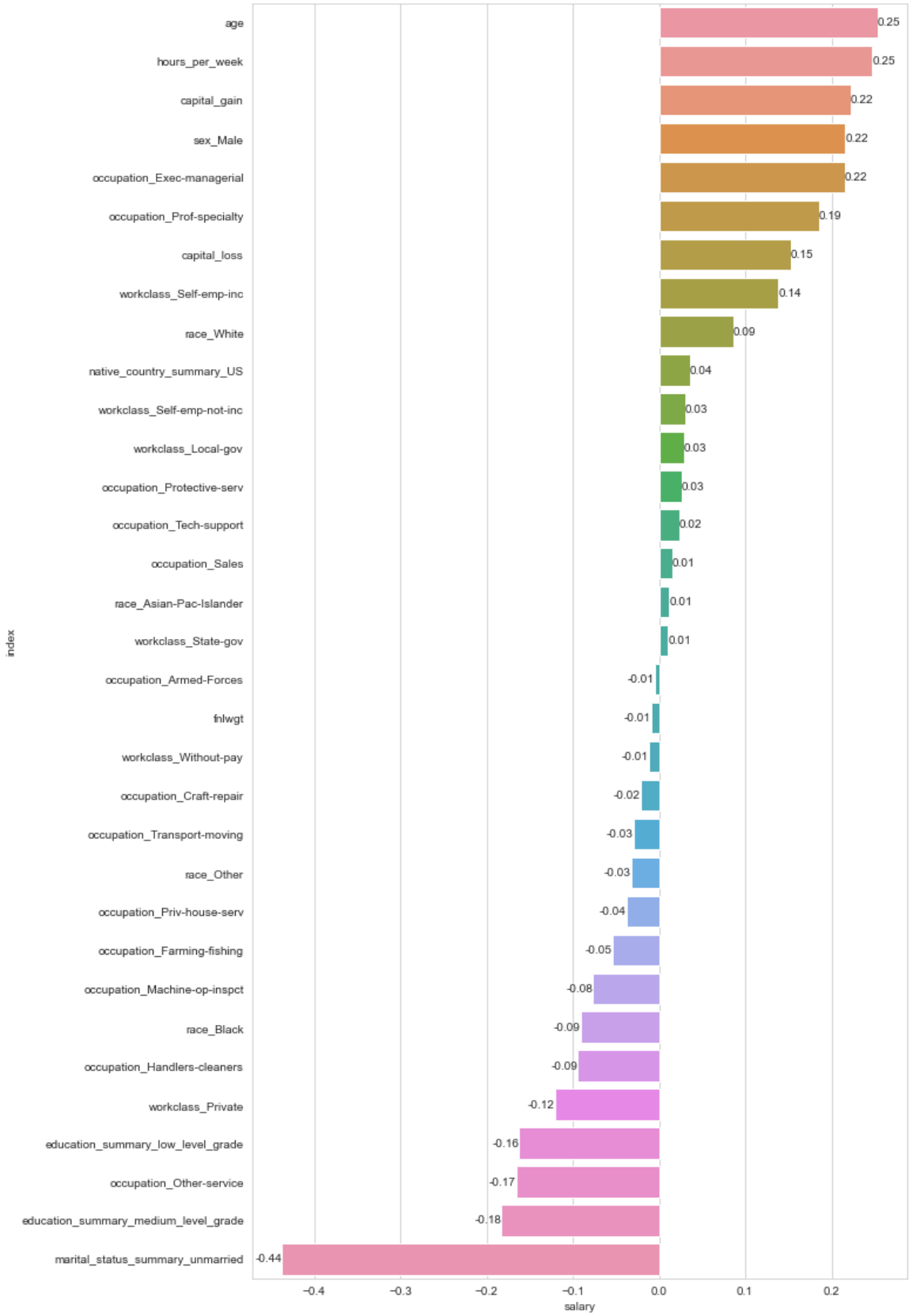
<AxesSubplot:>


age 0.25
hours_per_week 0.25
capital_gain 0.22
sex_Male 0.22
occupation_Exec-managerial 0.22
occupation_Prof-specialty 0.19
capital_loss 0.15
workclass_Self-emp-inc 0.14
race_White 0.086
native_country_summary_US 0.035
workclass_Self-emp-not-inc 0.03
workclass_Local-gov 0.029
occupation_Protective-serv 0.026
occupation_Tech-support 0.023
occupation_Sales 0.015
race_Asian-Pac-Islander 0.012
workclass_State-gov 0.011
occupation_Armed-Forces -0.0055
fnlwgt -0.0096
workclass_Without-pay -0.012
occupation_Craft-repair -0.021
occupation_Transport-moving -0.029
race_Other -0.032
occupation_Priv-house-serv -0.038
occupation_Farming-fishing -0.054
occupation_Machine-op-inspct -0.077
race_Black -0.091
occupation_Handlers-cleaners -0.094
workclass_Private -0.12
education_summary_low_level_grade -0.16
occupation_Other-service -0.17
education_summary_medium_level_grade -0.18
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4

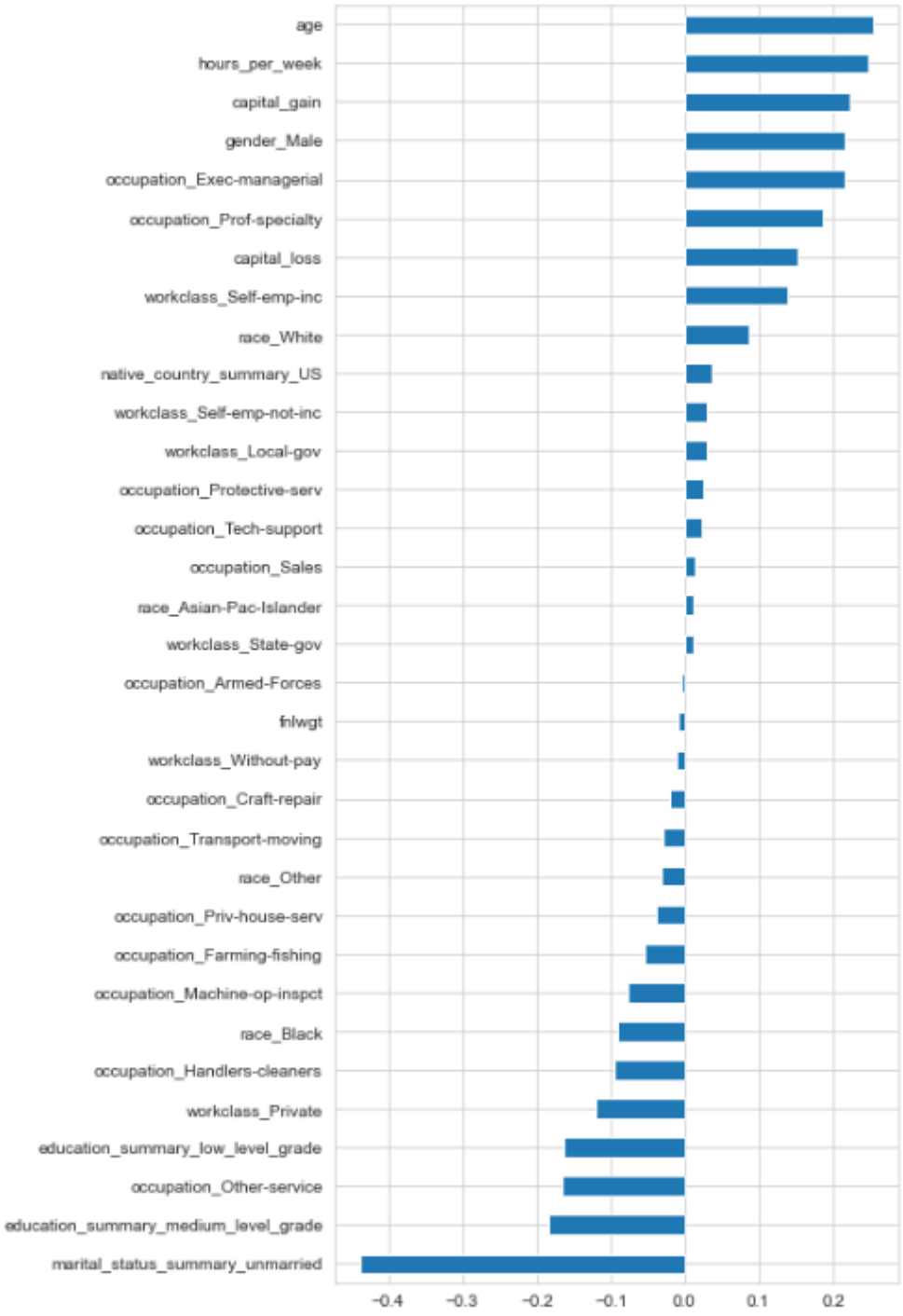

marital_status_summary_unmarried
-0.44
salary
capital_gain 0.22
gender_Male 0.22
occupation_Exec-managerial 0.22
occupation_Prof-specialty 0.19
capital_loss 0.15
workclass_Self-emp-inc 0.14
race_White 0.086
native_country_summary_US 0.035
workclass_Self-emp-not-inc 0.03
workclass_Local-gov 0.029
occupation_Protective-serv 0.026
occupation_Tech-support 0.023
occupation_Sales 0.015
race_Asian-Pac-Islander 0.012
workclass_State-gov 0.011
occupation_Armed-Forces -0.0055
fnlwgt -0.0096
workclass_Without-pay -0.012
occupation_Craft-repair -0.021
occupation_Transport-moving -0.029
race_Other -0.032
occupation_Priv-house-serv -0.038
occupation_Farming-fishing -0.054
occupation_Machine-op-inspct -0.077
race_Black -0.091
occupation_Handlers-cleaners -0.094
workclass_Private -0.12
education_summary_low_level_grade -0.16
occupation_Other-service -0.17
education_summary_medium_level_grade -0.18
marital_status_summary_unmarried -0.44
salary
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4

In [1252]:

```
# Your Code is Here
plt.figure(figsize=(10,20))
df_temp = df_dummy.corr().salary.to_frame().drop('salary').sort_values(by='salary', ascending = False).reset_index()
g = sns.barplot(y='index', x='salary', data=df_temp)
plt.bar_label(g.containers[0],fmt='%0.2f');
```



Desired Output:



Content

The End of the Project

WAY TO REINVENT YOURSELF