Reading data

In [5]:

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

In [6]:

```
df=pd.read_csv(r"C:\Users\DSK 8920444598\Downloads\adult_data.csv")
```

In [7]:

```
pd.set_option("display.max_columns",1000)
pd.set_option("display.max_rows",1000)
```

In [8]:

```
df
```

Out[8]:

	age	workclass	education	marrital status	occupation	sex	capital gain	capital loss	working hours per week	salary
0	39	State-gov	Bachelors	Never-married	Adm-clerical	Male	2174	0	40	<=50K
1	50	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Male	0	0	13	<=50K
2	38	Private	HS-grad	Divorced	Handlers-cleaners	Male	0	0	40	<=50K
3	53	Private	11th	Married-civ-spouse	Handlers-cleaners	Male	0	0	40	<=50K
4	28	Private	Bachelors	Married-civ-spouse	Prof-specialty	Female	0	0	40	<=50K
...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	Assoc-acdm	Married-civ-spouse	Tech-support	Female	0	0	38	<=50K
32557	40	Private	HS-grad	Married-civ-spouse	Machine-op-inspct	Male	0	0	40	>50K
32558	58	Private	HS-grad	Widowed	Adm-clerical	Female	0	0	40	<=50K
32559	22	Private	HS-grad	Never-married	Adm-clerical	Male	0	0	20	<=50K
32560	52	Self-emp-inc	HS-grad	Married-civ-spouse	Exec-managerial	Female	15024	0	40	>50K

32561 rows × 10 columns

Data Information

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   age                     32561 non-null  int64 
 1   workclass               32561 non-null  object
 2   education               32561 non-null  object
 3   marrital status         32561 non-null  object
 4   occupation              32561 non-null  object
 5   sex                     32561 non-null  object
 6   capital gain            32561 non-null  int64 
 7   capital loss            32561 non-null  int64 
 8   working hours per week  32561 non-null  int64 
 9   salary                  32561 non-null  object
dtypes: int64(4), object(6)
memory usage: 2.5+ MB
```

Data descriptive stats

In [11]:

```
df.describe(include='all').T
```

Out[11]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
age	32561.0	NaN	NaN	NaN	38.581647	13.640433	17.0	28.0	37.0	48.0	90.0
workclass	32561	9	Private	22696	NaN	NaN	NaN	NaN	NaN	NaN	NaN
education	32561	16	HS-grad	10501	NaN	NaN	NaN	NaN	NaN	NaN	NaN
marrital status	32561	7	Married-civ-spouse	14976	NaN	NaN	NaN	NaN	NaN	NaN	NaN
occupation	32561	15	Prof-specialty	4140	NaN	NaN	NaN	NaN	NaN	NaN	NaN
sex	32561	2	Male	21790	NaN	NaN	NaN	NaN	NaN	NaN	NaN
capital gain	32561.0	NaN	NaN	NaN	1077.648844	7385.292085	0.0	0.0	0.0	0.0	99999.0
capital loss	32561.0	NaN	NaN	NaN	87.30383	402.960219	0.0	0.0	0.0	0.0	4356.0
working hours per week	32561.0	NaN	NaN	NaN	40.437456	12.347429	1.0	40.0	40.0	45.0	99.0
salary	32561	2	<=50K	24720	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Drop nan columns

In [12]:

```
for col in df.columns:
    print(f'the percentage of null values of {col} is {round((df[col].isnull().sum()/df.shape[0]*100),2)}%')
```

```
the percentage of null values of age is 0.0%
the percentage of null values of workclass is 0.0%
the percentage of null values of education is 0.0%
the percentage of null values of marrital status is 0.0%
the percentage of null values of occupation is 0.0%
the percentage of null values of sex is 0.0%
the percentage of null values of capital gain is 0.0%
the percentage of null values of capital loss is 0.0%
the percentage of null values of working hours per week is 0.0%
the percentage of null values of salary is 0.0%
```

In [13]:

```
#But in the dataset we saw many ? values, so for tackling them we convert them first to null values
```

In [14]:

```
df[df == '?'] = np.nan
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   age                     32561 non-null  int64 
 1   workclass               30725 non-null  object
 2   education               32561 non-null  object
 3   marrital status         32561 non-null  object
 4   occupation              30718 non-null  object
 5   sex                     32561 non-null  object
 6   capital gain            32561 non-null  int64 
 7   capital loss            32561 non-null  int64 
 8   working hours per week  32561 non-null  int64 
 9   salary                  32561 non-null  object
dtypes: int64(4), object(6)
memory usage: 2.5+ MB
```

In [15]:

```
for col in df.columns:
    print(f'the percentage of null values of {col} is {round((df[col].isnull().sum()/df.shape[0]*100),2)}%')
```

```
the percentage of null values of age is 0.0%
the percentage of null values of workclass is 5.64%
the percentage of null values of education is 0.0%
the percentage of null values of marrital status is 0.0%
the percentage of null values of occupation is 5.66%
the percentage of null values of sex is 0.0%
the percentage of null values of capital gain is 0.0%
the percentage of null values of capital loss is 0.0%
the percentage of null values of working hours per week is 0.0%
the percentage of null values of salary is 0.0%
```

Data Visualization

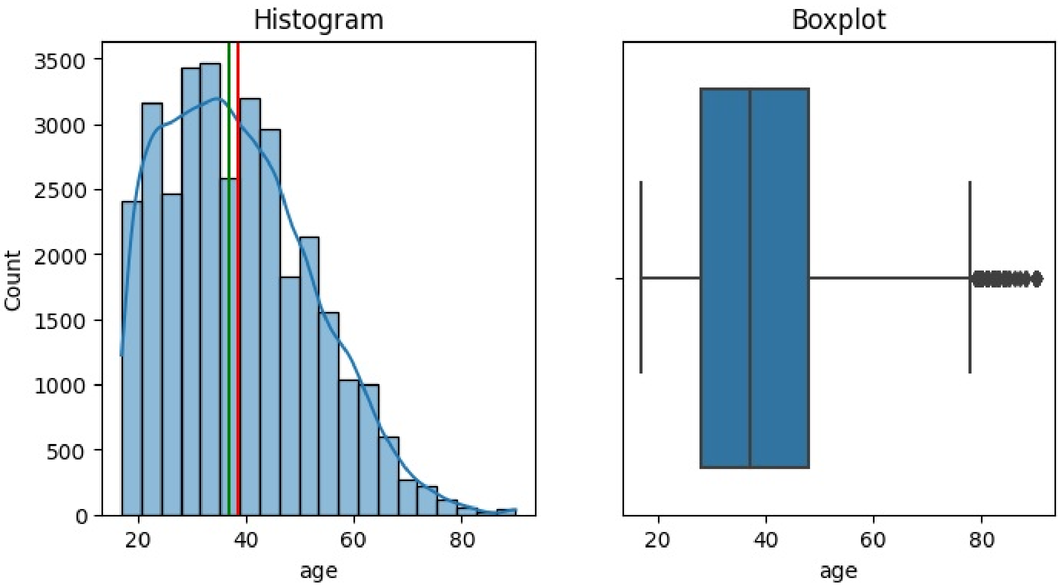
Univariate:

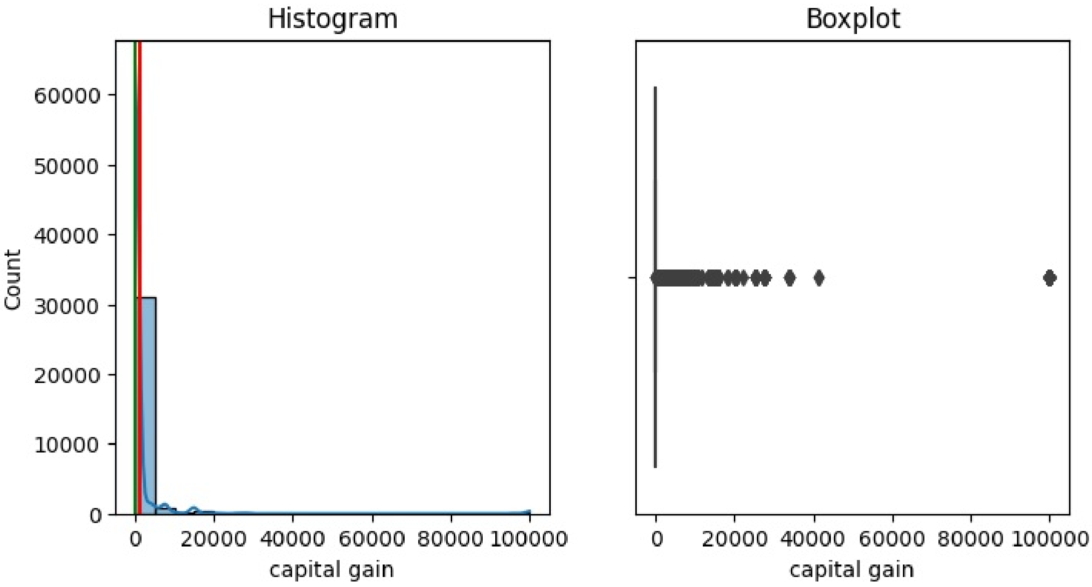
In [18]:

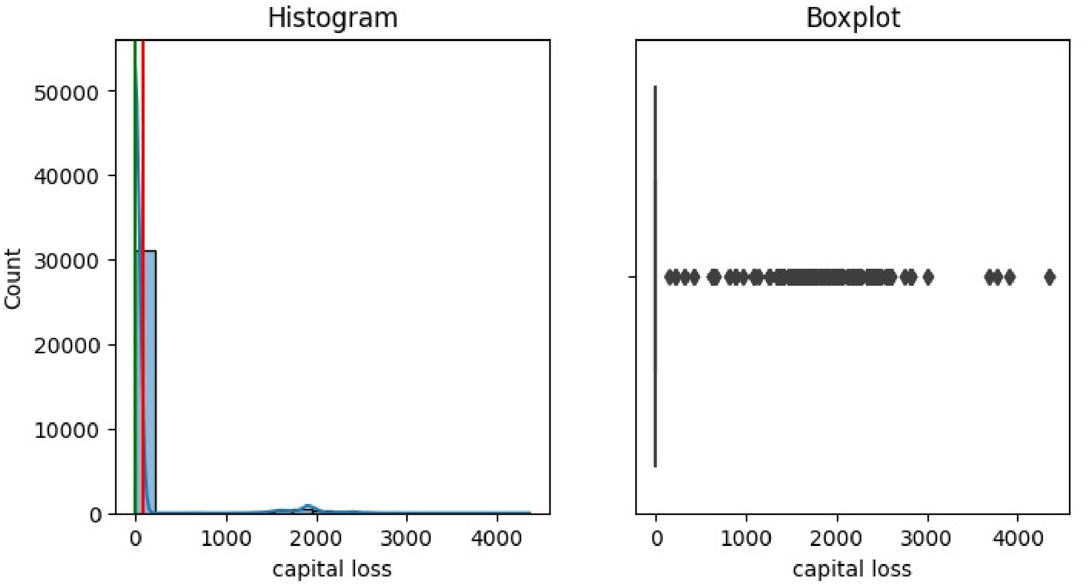
```
def uniplot(col):
    plt.figure(figsize=(8,4))
    plt.subplot(1,2,1)
    sns.histplot(x=df[col],bins=20,kde=True)
    plt.axvline(x=df[col].mean(),ymin=0,ymax=1,color="red")
    plt.axvline(x=df[col].median(),ymin=0,ymax=1,color="green")
    plt.title("Histogram")
    plt.subplot(1,2,2)
    sns.boxplot(x=df[col])
    plt.title("Boxplot")
    plt.show()
```

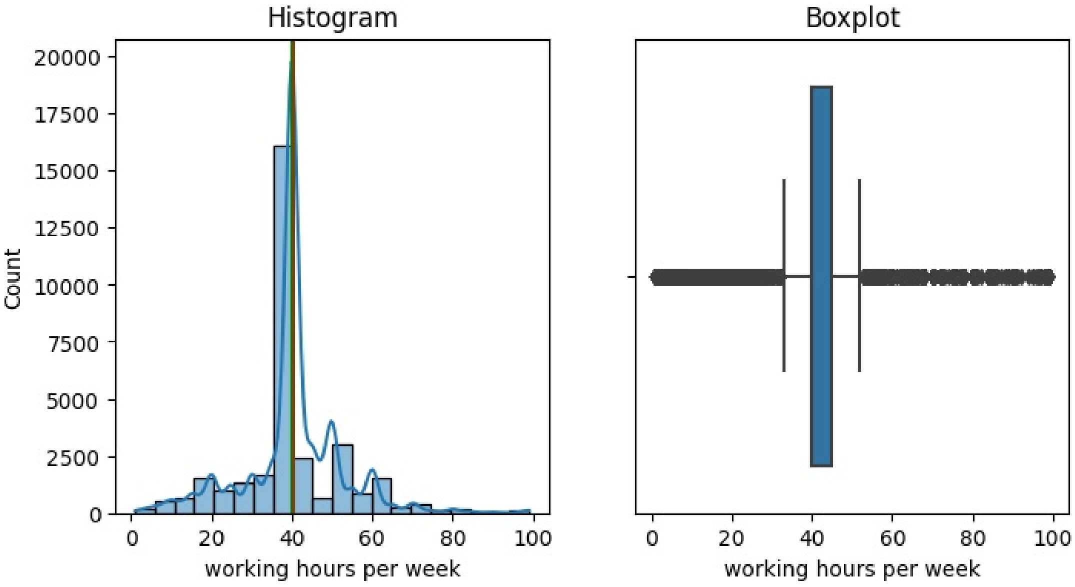
In [19]:

```
for col in df.select_dtypes(exclude='object'):
    uniplot(col)
```



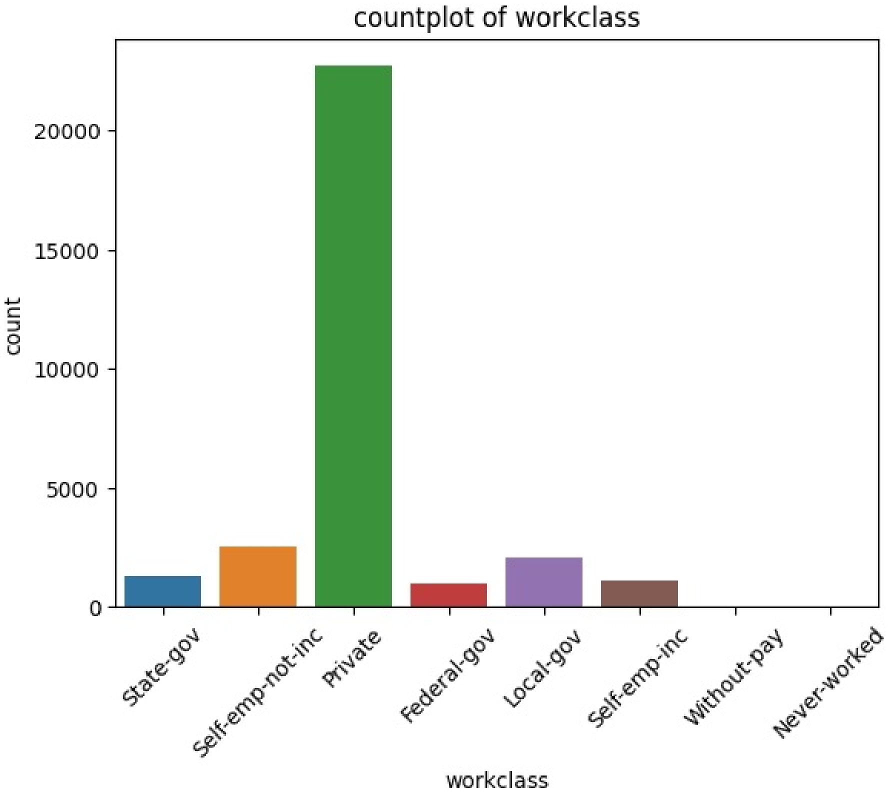


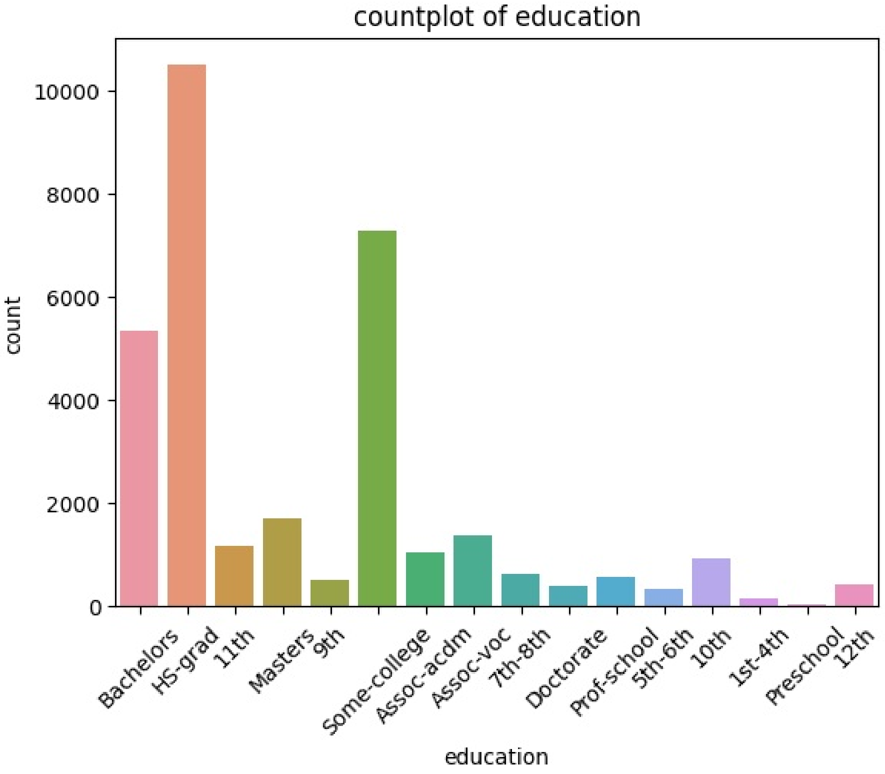


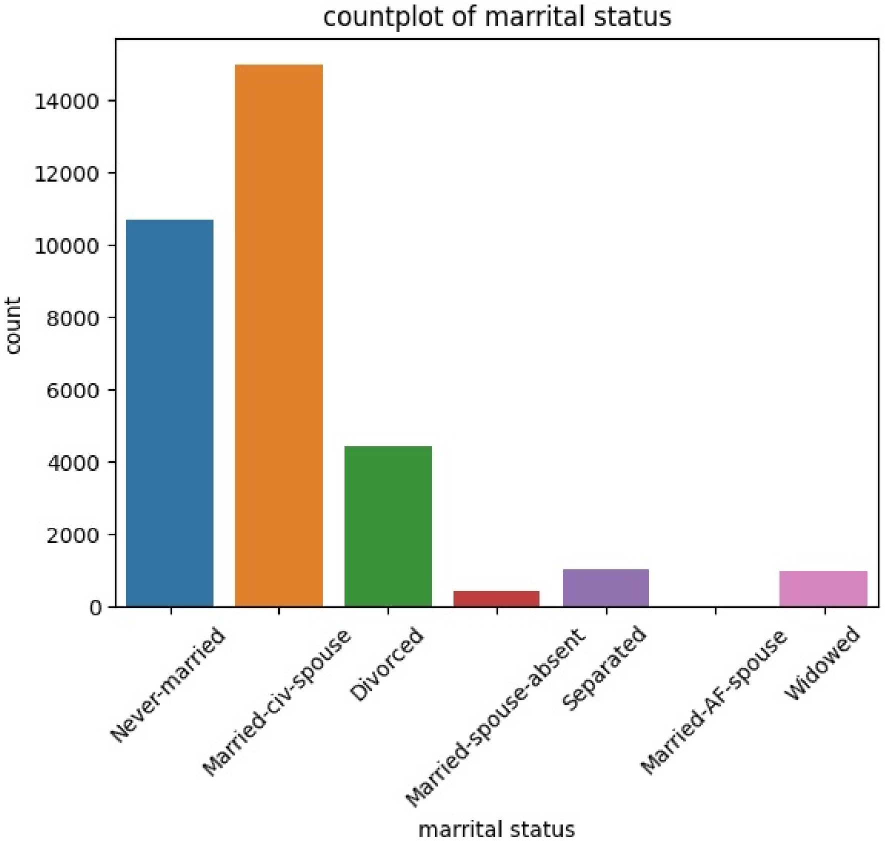


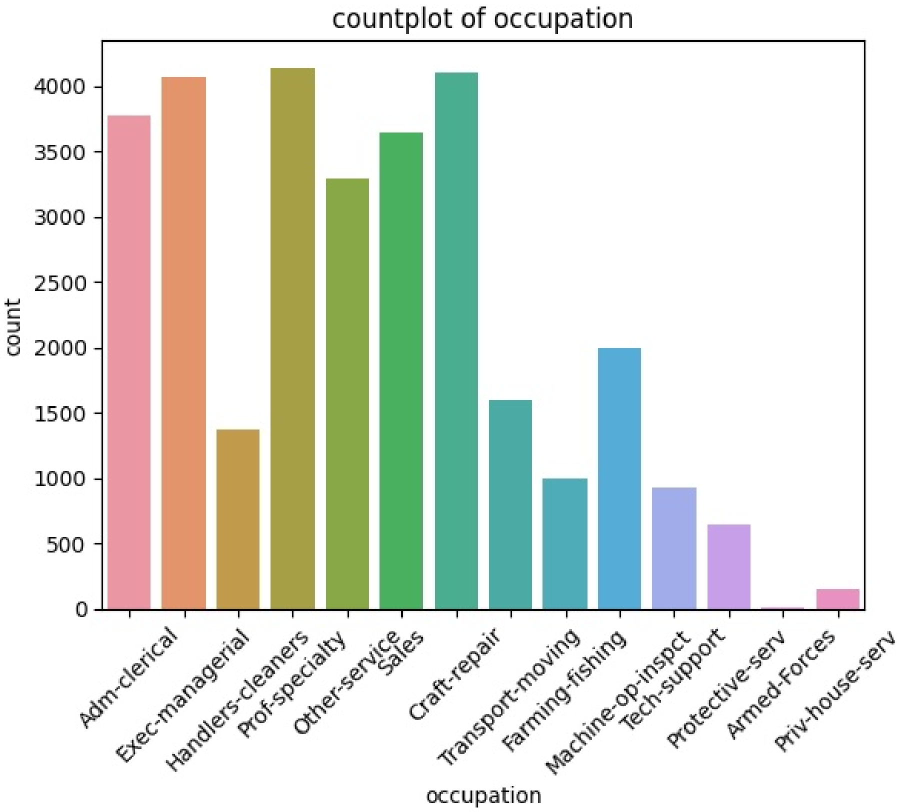
In [20]:

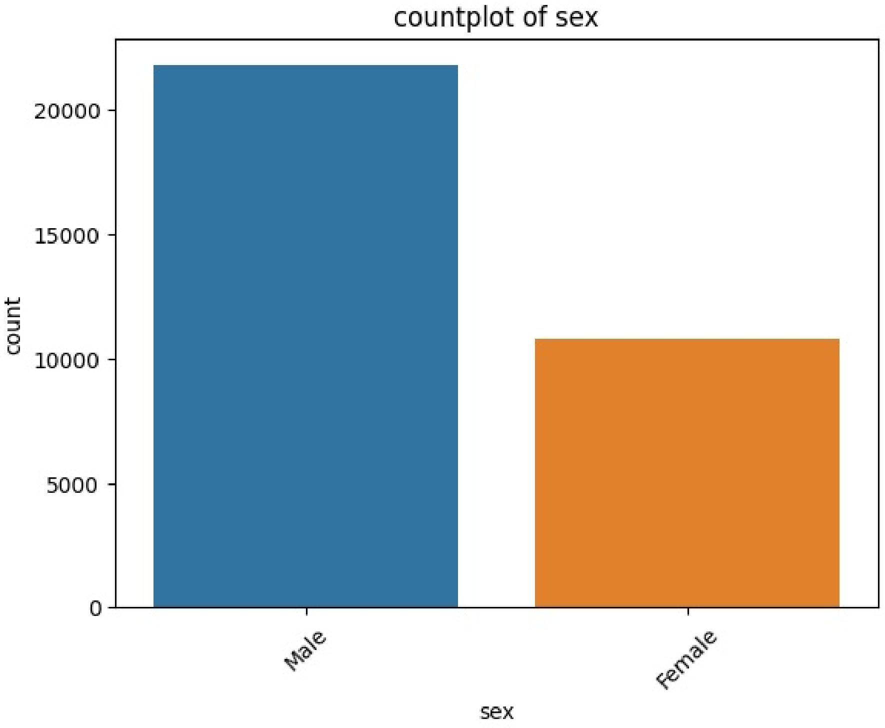
```
for col in df.select_dtypes(include='object'):
    sns.countplot(x=df[col])
    plt.title(f'countplot of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

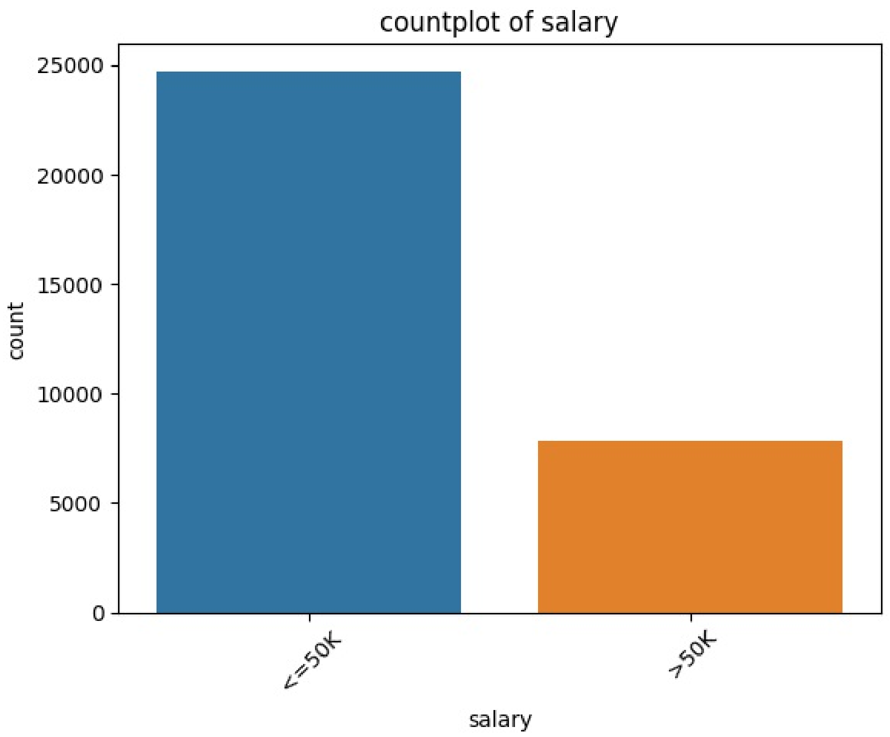



countplot of education
count
10000
8000
6000
4000
2000
0
Bachelors
HS-grad
11th
Masters
9th
Some-college
Assoc-acdm
Assoc-voc
7th-8th
Doctorate
Prof-school
5th-6th
10th
1st-4th
Preschool
12th
education


countplot of marrital status
count
14000
12000
10000
8000
6000
4000
2000
0
Never-married
Married-civ-spouse
Divorced
Married-spouse-absent
Separated
Married-AF-spouse
Widowed
marrital status


countplot of occupation
count
4000
3500
3000
2500
2000
1500
1000
500
0
Adm-clerical
Exec-managerial
Handlers-cleaners
Prof-specialty
Other-service
Sales
Craft-repair
Transport-moving
Farming-fishing
Machine-op-inspct
Tech-support
Protective-serv
Armed-Forces
Priv-house-serv
occupation

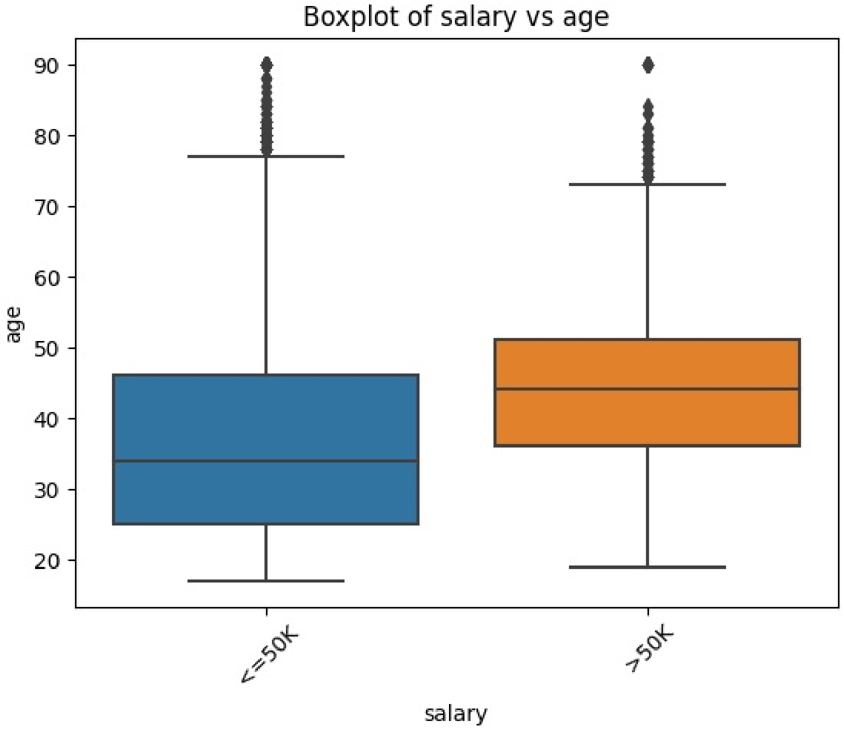

countplot of sex
count
20000
15000
10000
5000
0
Male
Female
sex

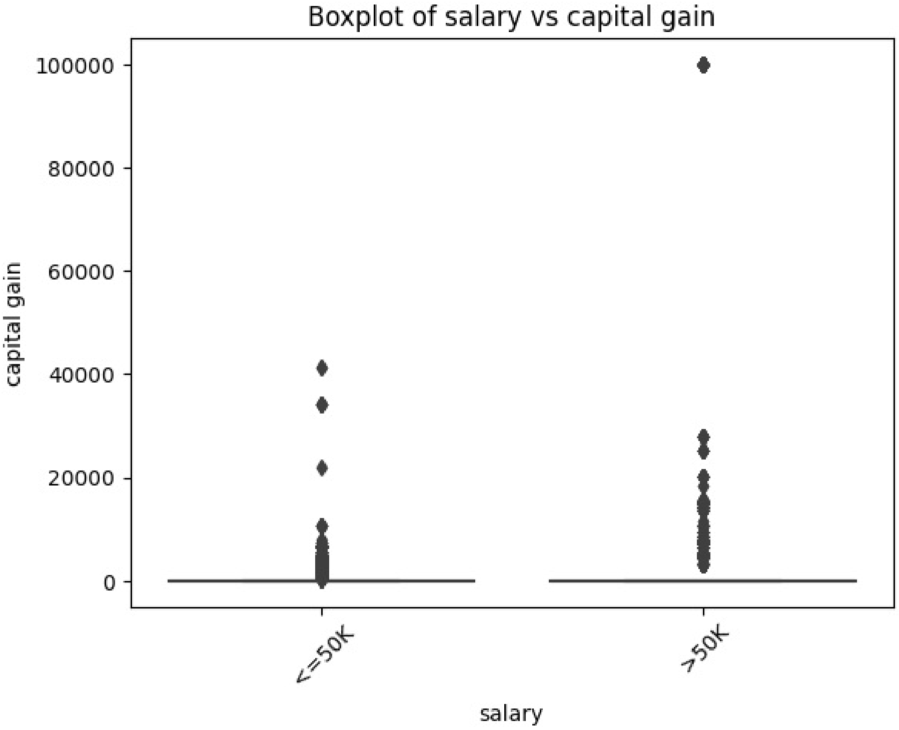


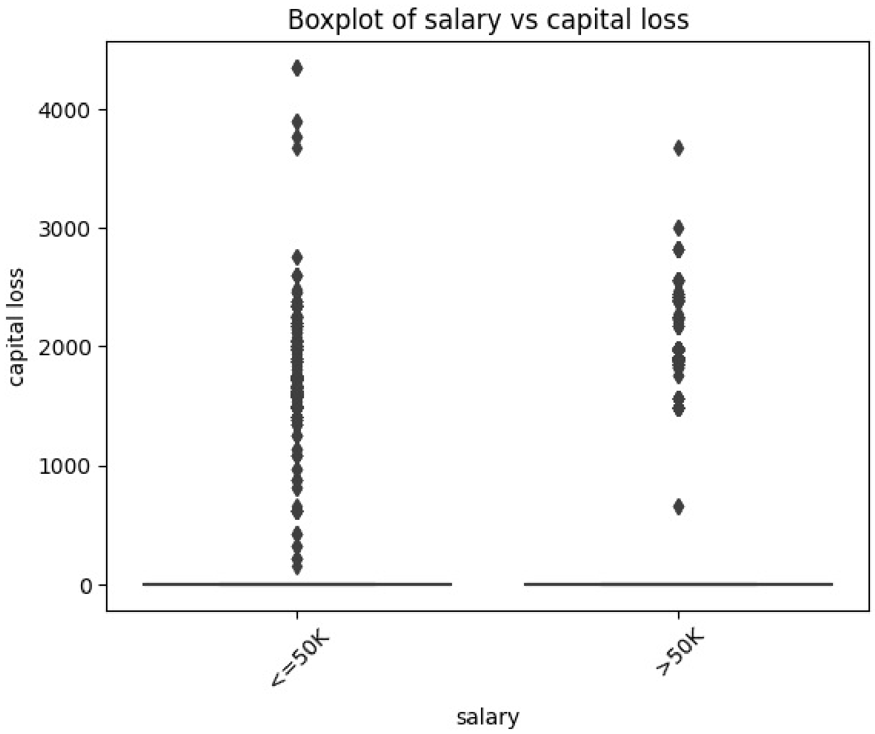
Bivariate:

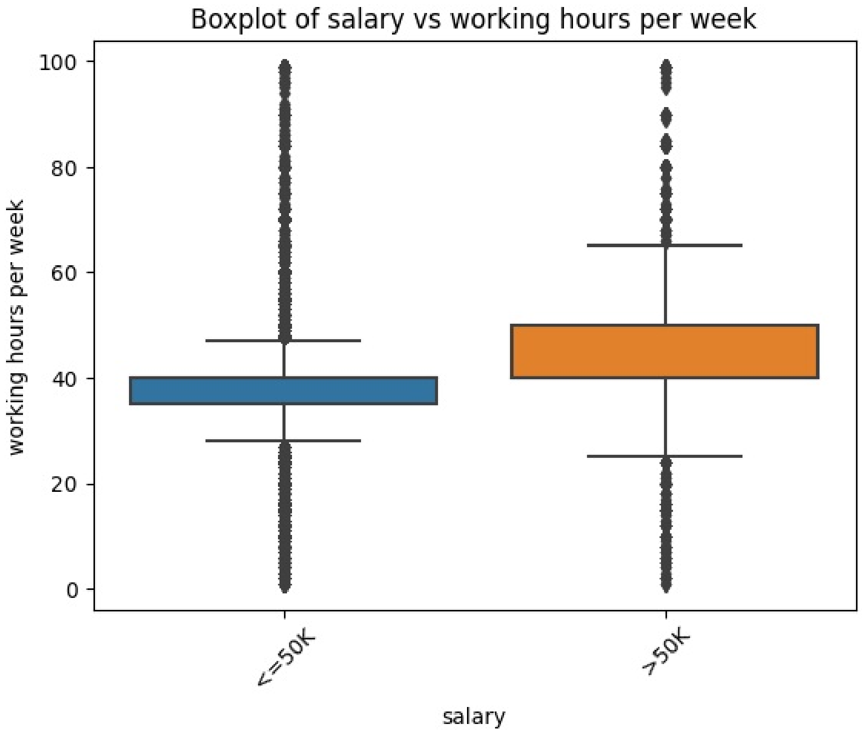
In [24]:

```
for col in df.select_dtypes(include='int64').columns:
    sns.boxplot(x=df['salary'],y=df[col])
    plt.title(f'Boxplot of salary vs {col}')
    plt.xticks(rotation=45)
    plt.show()
```


Boxplot of salary vs age
age
90
80
70
60
50
40
30
20
<=50K
>50K
salary


Boxplot of salary vs capital gain
capital gain
100000
80000
60000
40000
20000
0
<=50K
>50K
salary

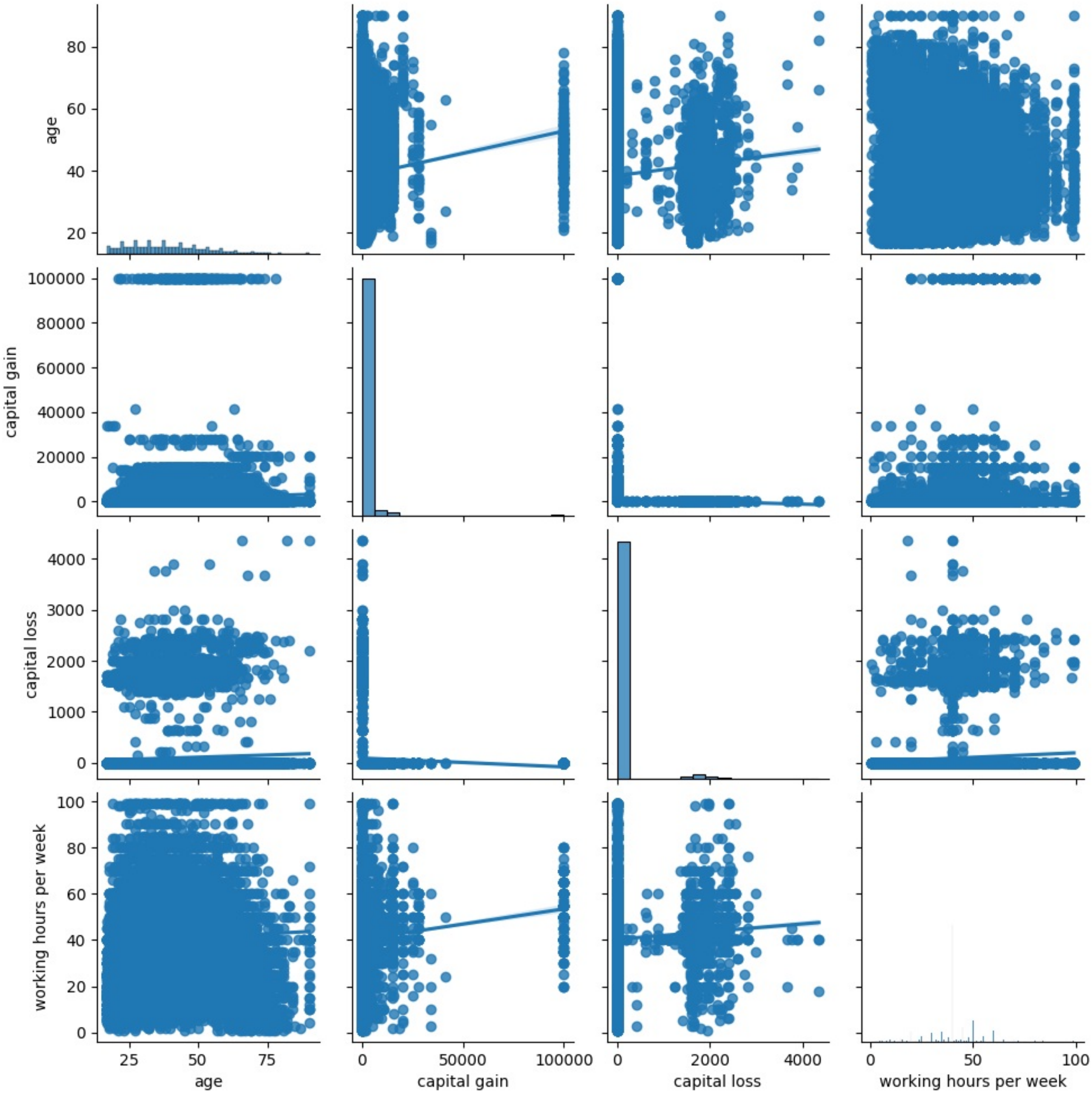




Multivariate:

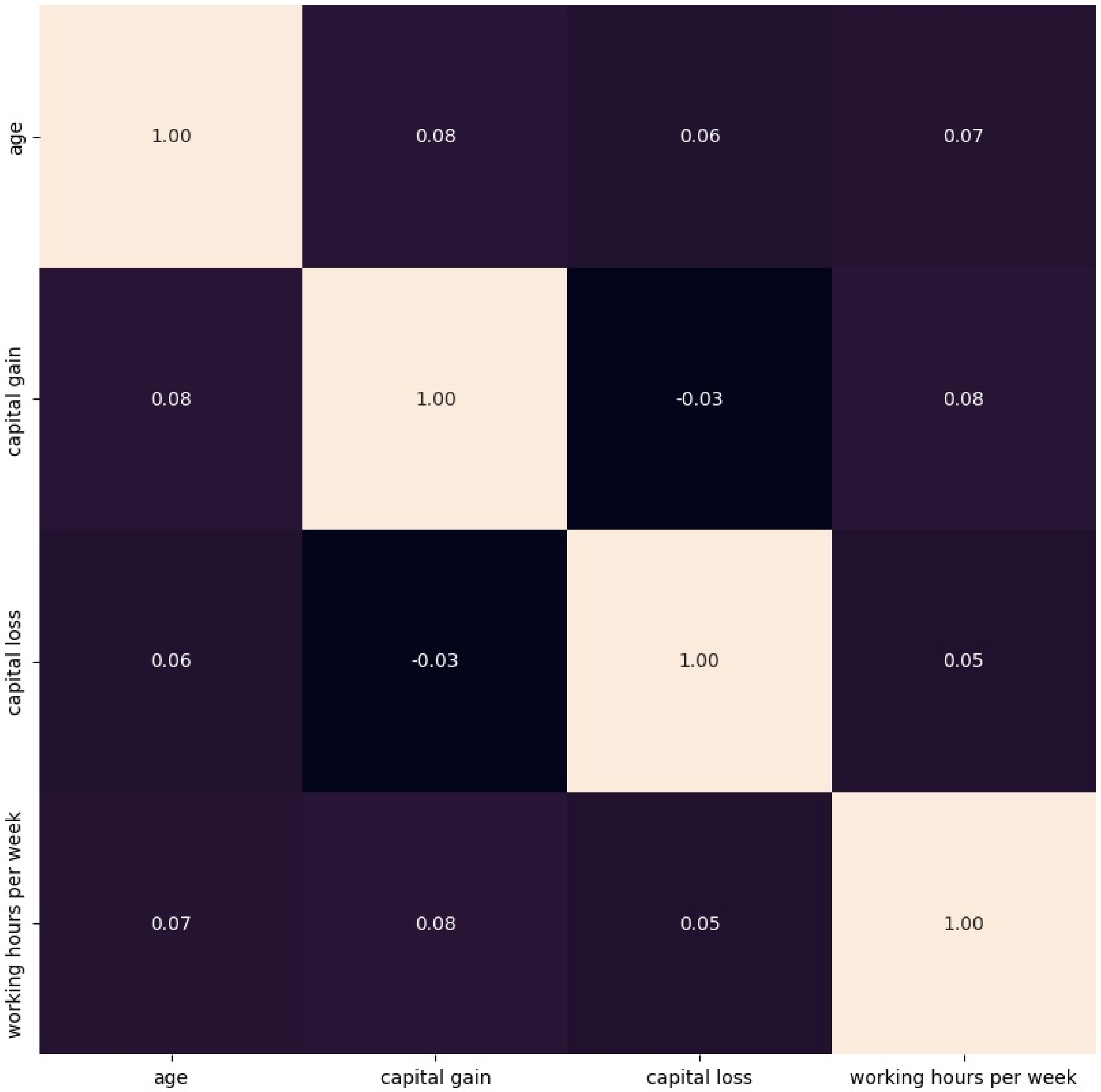
```
In [25]: sns.pairplot(df,kind='reg')
```

```
Out[25]: <seaborn.axisgrid.PairGrid at 0x22a23ee45e0>
```



```
In [28]: plt.figure(figsize=(10,10))
         sns.heatmap(df.corr(),annot=True,cbar=False,fmt='.2f')
```

```
Out[28]: <AxesSubplot:>
```



Imputation

```
In [29]: df.isnull().sum()
```

```
Out[29]: age                          0
         workclass                 1836
         education                    0
         marrital status              0
         occupation                1843
         sex                          0
         capital gain                 0
         capital loss                 0
         working hours per week       0
         salary                       0
         dtype: int64
```

```
In [30]: for col in df.select_dtypes(include='object'):
             df[col]=df[col].replace(np.nan,df[col].mode()[0])
```

```
In [31]: df.isnull().sum()
```

Out[31]:
```
age                       0
workclass                 0
education                 0
marrital status           0
occupation                0
sex                       0
capital gain              0
capital loss              0
working hours per week    0
salary                    0
dtype: int64
```

In [32]:
```
df
```

Out[32]:

	age	workclass	education	marrital status	occupation	sex	capital gain	capital loss	working hours per week	salary
0	39	State-gov	Bachelors	Never-married	Adm-clerical	Male	2174	0	40	<=50K
1	50	Self-emp-not-inc	Bachelors	Married-civ-spouse	Exec-managerial	Male	0	0	13	<=50K
2	38	Private	HS-grad	Divorced	Handlers-cleaners	Male	0	0	40	<=50K
3	53	Private	11th	Married-civ-spouse	Handlers-cleaners	Male	0	0	40	<=50K
4	28	Private	Bachelors	Married-civ-spouse	Prof-specialty	Female	0	0	40	<=50K
...	...	...	...	...	...	...	...	...	...	...
32556	27	Private	Assoc-acdm	Married-civ-spouse	Tech-support	Female	0	0	38	<=50K
32557	40	Private	HS-grad	Married-civ-spouse	Machine-op-inspct	Male	0	0	40	>50K
32558	58	Private	HS-grad	Widowed	Adm-clerical	Female	0	0	40	<=50K
32559	22	Private	HS-grad	Never-married	Adm-clerical	Male	0	0	20	<=50K
32560	52	Self-emp-inc	HS-grad	Married-civ-spouse	Exec-managerial	Female	15024	0	40	>50K

32561 rows × 10 columns

In [34]:
```
df['workclass'].values
```

Out[34]:
```
array(['State-gov', 'Self-emp-not-inc', 'Private', ..., 'Private',
       'Private', 'Self-emp-inc'], dtype=object)
```

Encoding

In [35]:
```
for col in df.select_dtypes(include='object'):
    df[col]=pd.Categorical(df[col]).codes
```

In [36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     32561 non-null  int64
 1   workclass               32561 non-null  int8
 2   education               32561 non-null  int8
 3   marrital status         32561 non-null  int8
 4   occupation              32561 non-null  int8
 5   sex                     32561 non-null  int8
 6   capital gain            32561 non-null  int64
 7   capital loss            32561 non-null  int64
 8   working hours per week  32561 non-null  int64
 9   salary                  32561 non-null  int8
dtypes: int64(4), int8(6)
memory usage: 1.2 MB
```

In [37]:
```
df
```

Out[37]:

	age	workclass	education	marrital status	occupation	sex	capital gain	capital loss	working hours per week	salary
0	39	6	9	4	0	1	2174	0	40	0
1	50	5	9	2	3	1	0	0	13	0
2	38	3	11	0	5	1	0	0	40	0
3	53	3	1	2	5	1	0	0	40	0
4	28	3	9	2	9	0	0	0	40	0
...	...	...	...	...	...	...	...	...	...	...
32556	27	3	7	2	12	0	0	0	38	0
32557	40	3	11	2	6	1	0	0	40	1
32558	58	3	11	6	0	0	0	0	40	0
32559	22	3	11	4	0	1	0	0	20	0
32560	52	4	11	2	3	0	15024	0	40	1

32561 rows × 10 columns

Outlier detection

In [38]:

```
def outliers(col):
    q1=np.quantile(df[col],.25)
    q3=np.quantile(df[col],.75)

    iqr=q3-q1

    lr=q1-(1.5*iqr)
    ur=q3+(1.5*iqr)
    lower_per=round(df[df[col]<lr][col].count()/df.shape[0],2)
    upper_per=round(df[df[col]>ur][col].count()/df.shape[0],2)

    return print(f'the outliers percentage of {col} for lower_range is {lower_per}, for upper_range is {upper_p
```

In [59]:

```
for i in df.select_dtypes(include='int64').columns:
    outliers(i)
```

```
the outliers percentage of age for lower_range is 0.0, for upper_range is 0.0
the outliers percentage of capital gain for lower_range is 0.0, for upper_range is 0.08
the outliers percentage of capital loss for lower_range is 0.0, for upper_range is 0.05
the outliers percentage of working hours per week for lower_range is 0.17, for upper_range is 0.11
```

In [60]:

```
for i in df.select_dtypes(include='int8').columns:
    outliers(i)
```

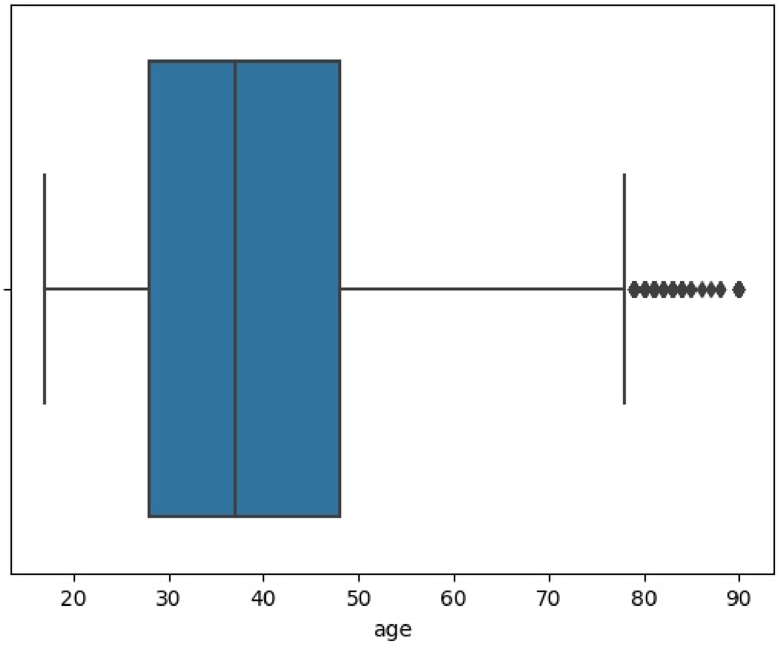
```
the outliers percentage of workclass for lower_range is 0.09, for upper_range is 0.15
the outliers percentage of education for lower_range is 0.09, for upper_range is 0.0
the outliers percentage of marrital status for lower_range is 0.0, for upper_range is 0.0
the outliers percentage of occupation for lower_range is 0.0, for upper_range is 0.0
the outliers percentage of sex for lower_range is 0.0, for upper_range is 0.0
the outliers percentage of salary for lower_range is 0.0, for upper_range is 0.24
```

In [62]:

```
sns.boxplot(x=df['age'])
```

Out[62]:

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



In [64]:
```
X=df.drop('salary',axis=1)
y=df.pop('salary')
```

In [65]:
```
X.head()
```

Out[65]:

	age	workclass	education	marrital status	occupation	sex	capital gain	capital loss	working hours per week
0	39	6	9	4	0	1	2174	0	40
1	50	5	9	2	3	1	0	0	13
2	38	3	11	0	5	1	0	0	40
3	53	3	1	2	5	1	0	0	40
4	28	3	9	2	9	0	0	0	40

In [66]:
```
y.head()
```

Out[66]:
```
0    0
1    0
2    0
3    0
4    0
Name: salary, dtype: int8
```

In [67]:
```
from sklearn.model_selection import train_test_split
```

In [68]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=15)
```

In [69]:
```
X_train.head()
```

Out[69]:

	age	workclass	education	marrital status	occupation	sex	capital gain	capital loss	working hours per week
15190	56	3	7	2	7	1	0	0	50
26120	29	3	15	4	11	1	0	0	80
15937	34	6	11	4	7	1	0	0	42
1744	43	5	15	2	2	1	0	0	60
29838	29	3	11	2	13	1	0	0	40

In [70]:

```
y_train.head()
```

Out[70]:

```
15190    0
26120    0
15937    0
1744     0
29838    0
Name: salary, dtype: int8
```

In [71]:

```
from sklearn.linear_model import LogisticRegression
```

In [72]:

```
lr = LogisticRegression()
```

In [74]:

```
lr.fit(X_train,y_train)
```

```
C:\Users\DSK 8920444598\AppData\Roaming\Python\Python310\site-packages\sklearn\linear_model\_logistic.py:444: C
onvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[74]:

```
▾ LogisticRegression
LogisticRegression()
```

In [75]:

```
lr.coef_
```

Out[75]:

```
array([[ 5.25703261e-03, -2.22023674e-01, -3.28494544e-02,
        -4.23546704e-01, -6.34266765e-03,  7.26911118e-02,
         3.32983239e-04,  7.63619061e-04,  1.21676127e-02]])
```

In [76]:

```
X_test
```

Out[76]:

	age	workclass	education	marrital status	occupation	sex	capital gain	capital loss	working hours per week
10125	47	5	11	2	0	0	0	0	35
11478	73	1	4	2	10	1	0	0	20
4224	18	3	15	4	7	0	0	0	25
6592	66	3	11	6	11	0	0	0	40
21910	39	3	2	0	0	0	0	0	40
...	...	...	...	...	...	...	...	...	...
4835	28	3	0	4	9	0	0	0	40
23660	17	3	0	4	2	1	594	0	30
10091	29	6	12	2	9	1	0	0	20
20819	30	3	9	4	9	1	13550	0	45
15198	32	3	0	0	2	1	0	0	40

10746 rows × 9 columns

In [77]:

```
lr.predict(X_test)
```

Out[77]:

```
array([0, 0, 0, ..., 0, 1, 0], dtype=int8)
```

In [78]:

```
lr.score(X_test,y_test)*100
```

Out[78]:

```
79.95533221663875
```

In [79]:

```
lr.score(X_train,y_train)*100
```

Out[79]:

```
79.08319963327986
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js