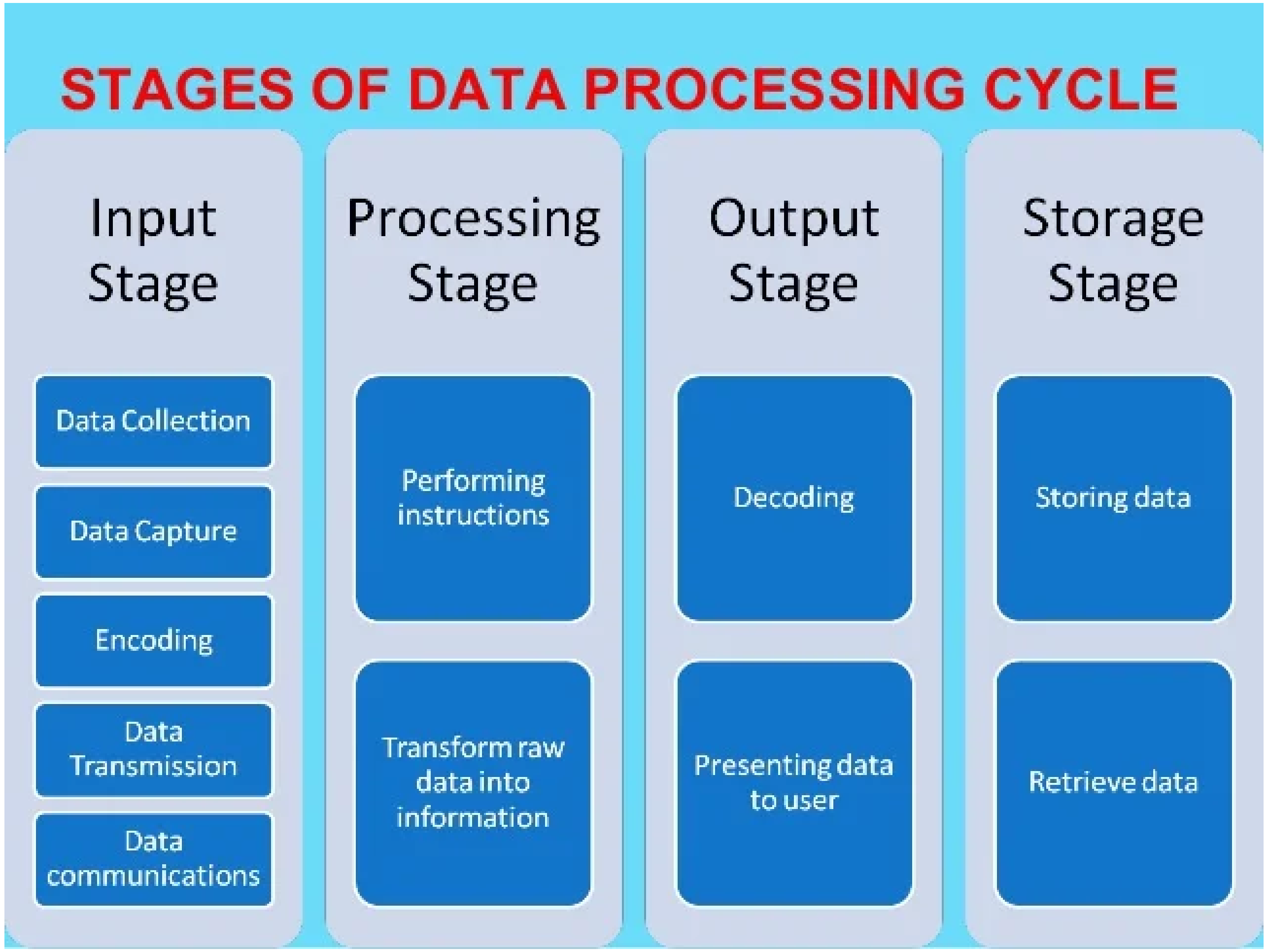
DATA PROCESSING:

- Data Processing is the task of converting data from a given form to a much more usable and desired form i.e. making it more meaningful and informative.
- Using Machine Learning algorithms, mathematical modeling, and statistical knowledge, this entire process can be automated.
- The output of this complete process can be in any desired form like graphs, videos, charts, tables, images, and many more, depending on the task we are performing and the requirements of the machine.
- This might seem to be simple but when it comes to massive organizations like Twitter, Facebook, Administrative bodies like Parliament, UNESCO, and health sector organizations, this entire process needs to be performed in a very structured manner
- Data processing is a crucial step in the machine learning (ML) pipeline, as it prepares the data for use in building and training ML models.
- The goal of data processing is to clean, transform, and prepare the data in a format that is suitable for modeling.
- Data processing refers to the entire process of collecting, transforming (i.e. cleaning, or putting the data into a usable state), and classifying data.

- Raw data is the data collected from various sources, in its original state.It is usually not in the most proper format for data analysis or modeling.
- Clean data is the data obtained after processing the raw data – i.e. it’s data that’s ready to be analyzed. It has been transformed into a usable format; incorrect, inconsistent, or missing data has (as much as possible) been corrected or removed
- Data processing refers to the entire process of collecting, transforming (i.e. cleaning, or putting the data into a usable state), and classifying data.

data processing flow chart:

data processing cycle:



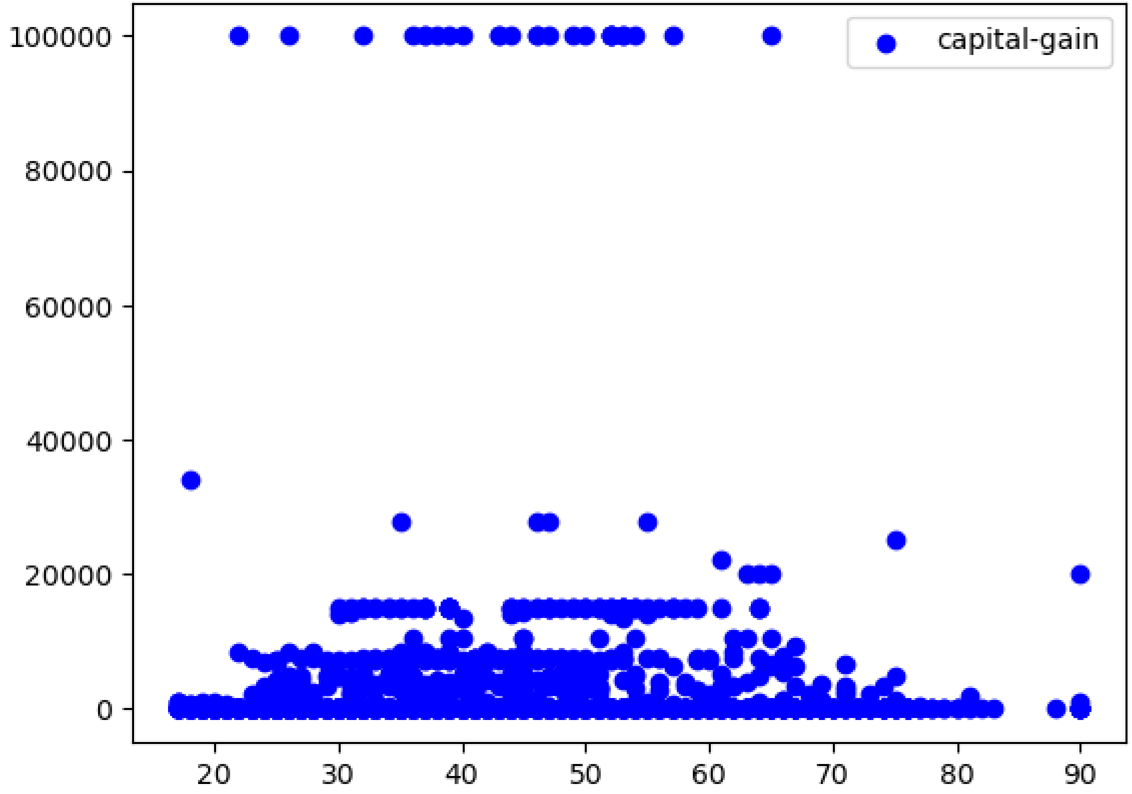
In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('Expense.csv')
df
```

Out[4]:

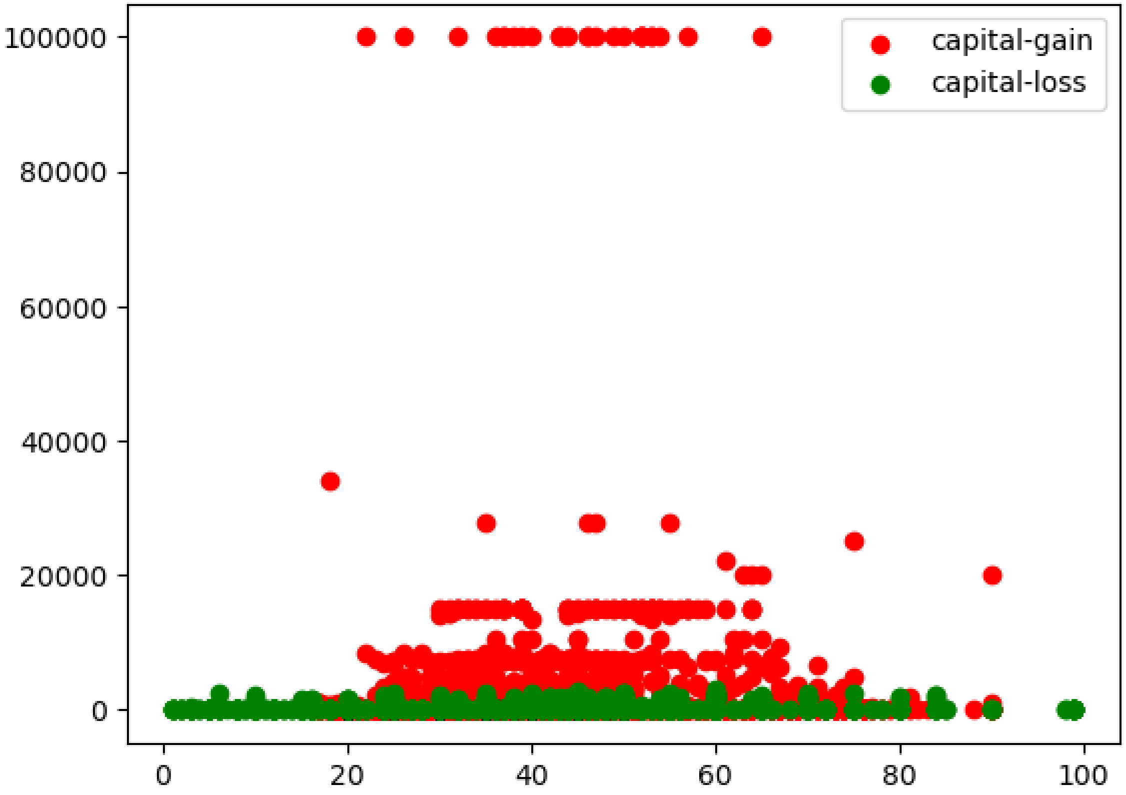
	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
0	39	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	15024	0	50	United-States	>50K
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K
2	50	Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	1902	65	United-States	>50K
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K
4	23	Local-gov	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Female	0	0	60	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	38	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	<=50K
4996	26	Private	Some-college	10	Never-married	Tech-support	Own-child	White	Female	0	0	40	United-States	<=50K
4997	20	Private	11th	7	Never-married	Transport-moving	Own-child	White	Male	0	0	60	United-States	<=50K
4998	24	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	60	Mexico	>50K
4999	40	Private	HS-grad	9	Divorced	Craft-repair	Not-in-family	White	Male	0	0	45	United-States	<=50K

5000 rows × 14 columns

In [5]:
```
# scatter plot
plt.scatter(x = 'age' ,y = 'capital-gain',label='capital-gain',data = df,color = 'b')
# add legend:
plt.legend()
# display the plot:
plt.show()
```



In [9]:
```
#  multiple scatter plot:
plt.scatter(x = 'age' ,y = 'capital-gain',label='capital-gain',data = df,color = 'r')
plt.scatter(x = 'hours-per-week' ,y = 'capital-loss',label='capital-loss',data = df,color = 'g')
# add legend:
plt.legend()
# display the plot:
plt.show()
```



In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('Expense.csv')
df
```

Out[2]:

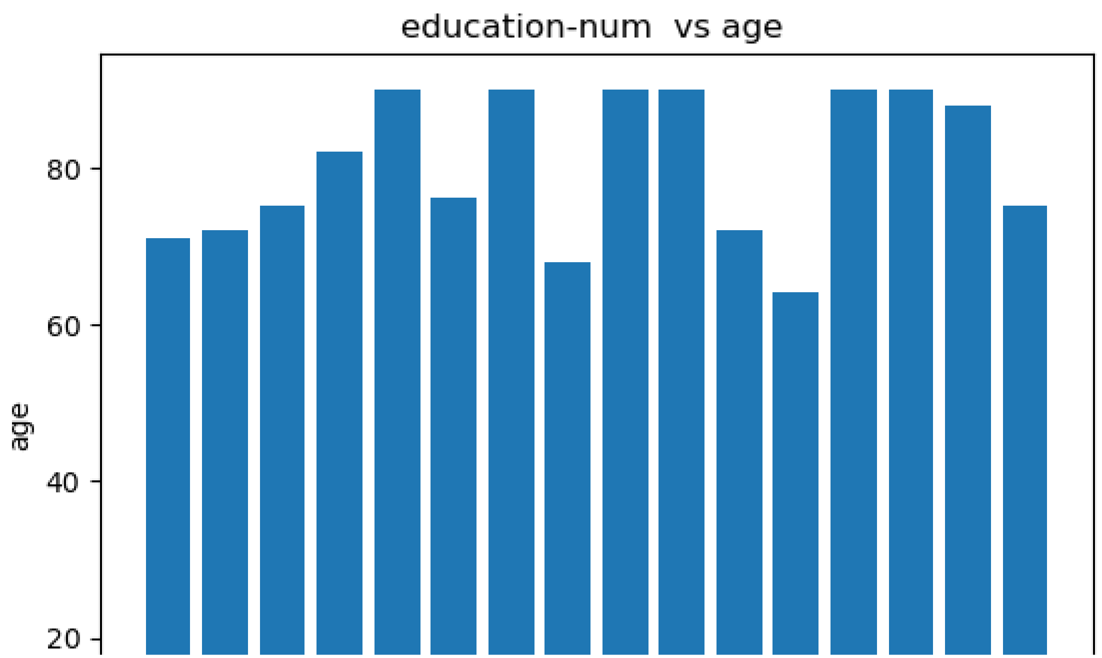
	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
0	39	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	15024	0	50	United-States	>50K
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K
2	50	Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	1902	65	United-States	>50K
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K
4	23	Local-gov	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Female	0	0	60	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	38	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	<=50K
4996	26	Private	Some-college	10	Never-married	Tech-support	Own-child	White	Female	0	0	40	United-States	<=50K
4997	20	Private	11th	7	Never-married	Transport-moving	Own-child	White	Male	0	0	60	United-States	<=50K
4998	24	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	60	Mexico	>50K
4999	40	Private	HS-grad	9	Divorced	Craft-repair	Not-in-family	White	Male	0	0	45	United-States	<=50K

5000 rows × 14 columns

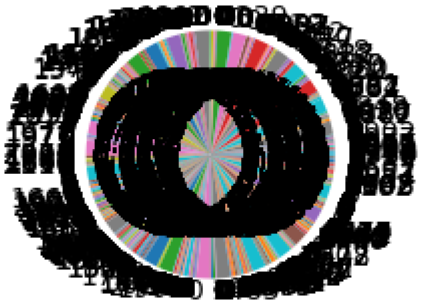
In [3]:
```
df.dtypes
```

Out[3]:
```
age                int64
workclass         object
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
Expense           object
dtype: object
```

In [4]:
```
# Bar chart with purpose against amount
plt.bar(df['education-num'], df['age'])
plt.title("education-num  vs age ")
# Setting the X and Y labels
plt.xlabel('education-num')
plt.ylabel('age')
# Adding the legends
plt.show()
```

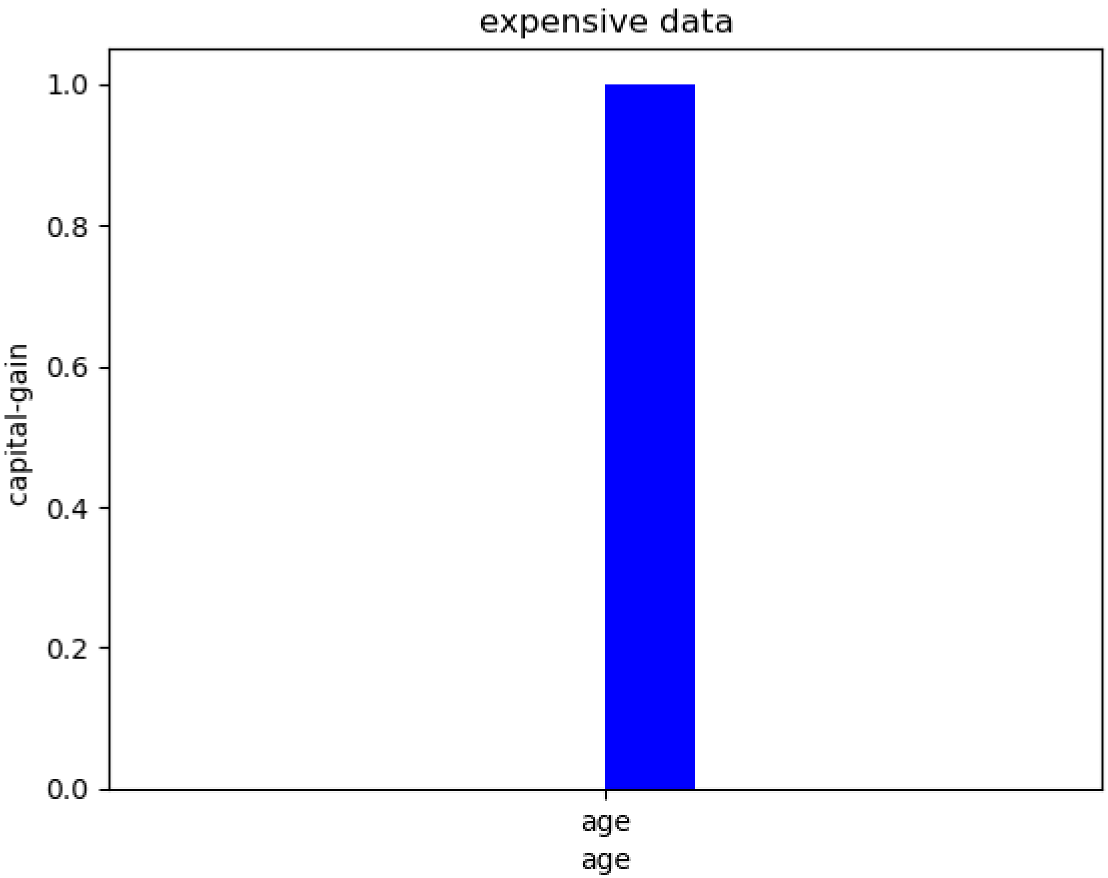


In [17]:
```
# pie chart :
plt.figure(figsize =(2,3))
plt.pie( x=df['capital-gain'],labels=df['capital-loss'] ,autopct = '%1.2f%%')
plt.show()
```

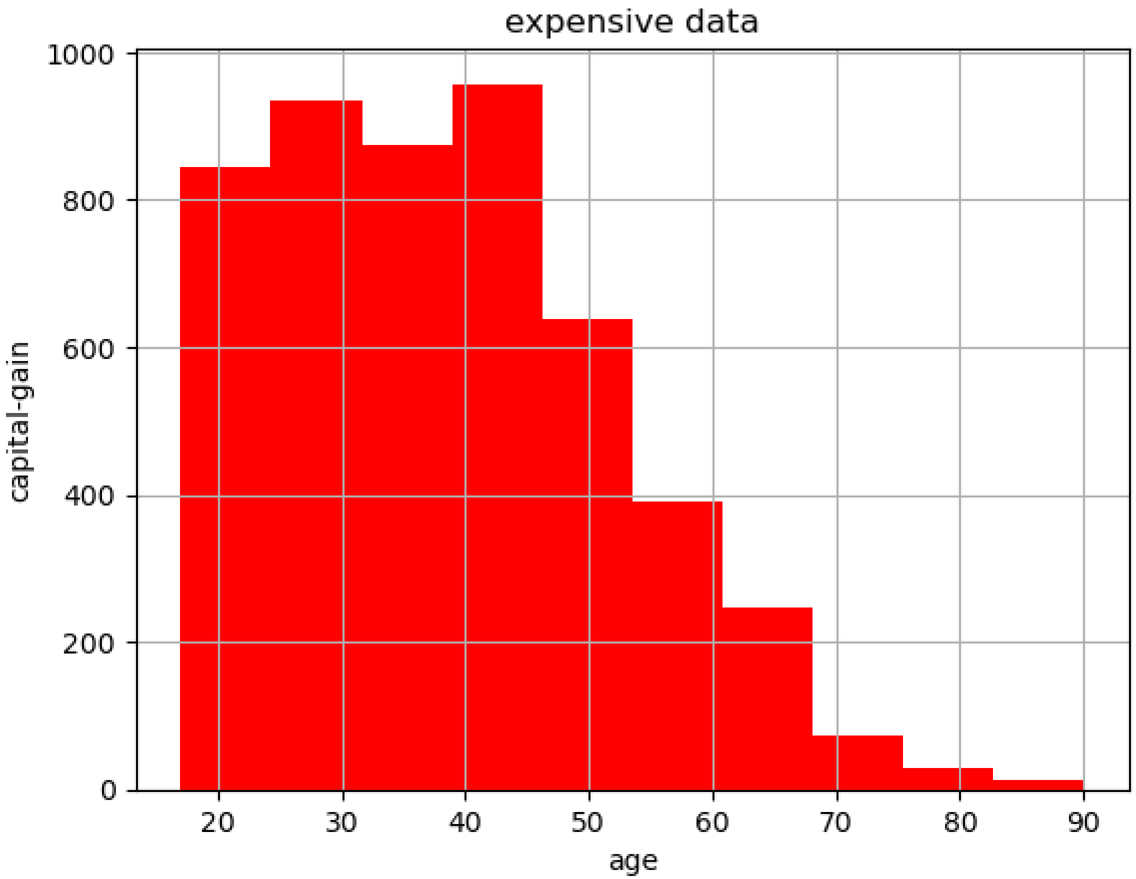


In [ ]:
```
# histogram
- always combined with numerical columns
- it posseses the distribution of particular column
```

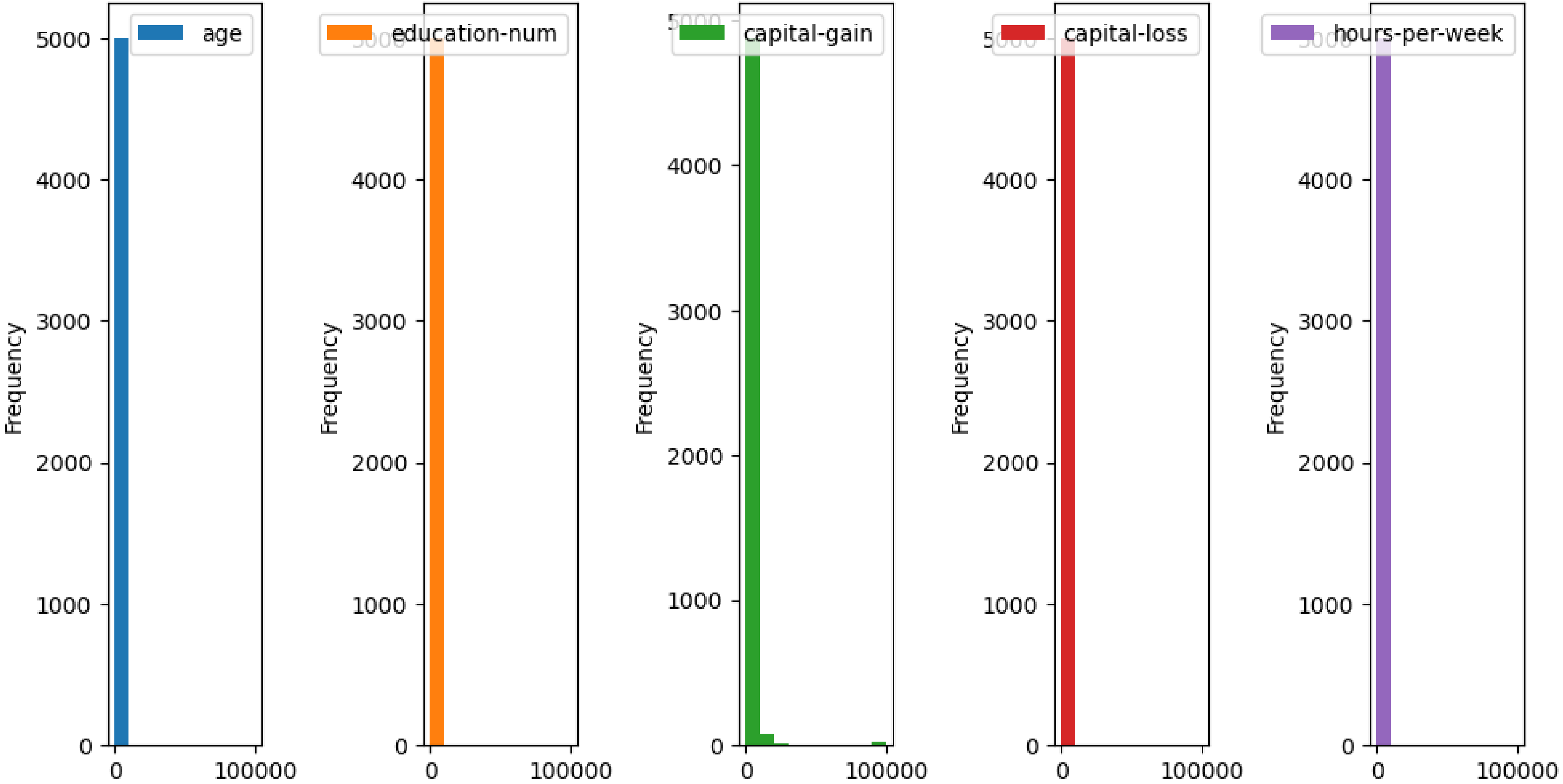
In [7]:
```
plt.hist(x =['age'],color ='b')
# add title:
plt.title("expensive data")
# add laebels:
plt.xlabel("age")
plt.ylabel("capital-gain ")
# display the plot
plt.show()
```



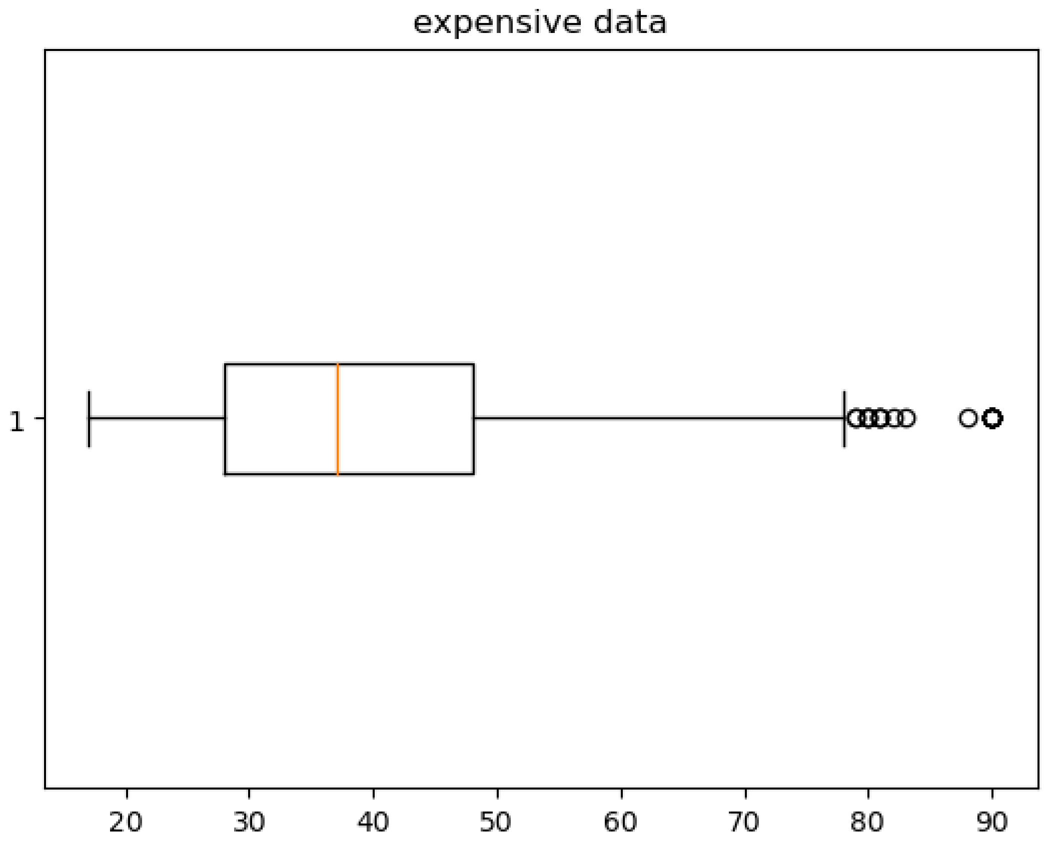
In [14]:
```
# plot the histogram with multiple bins add grid :
plt.hist(x = df["age"],color ='r',bins=10)
plt.title("expensive data")
plt.xlabel("age")
plt.ylabel("capital-gain")
# plot the grid:
plt.grid()
# display the plot:
plt.show()
```



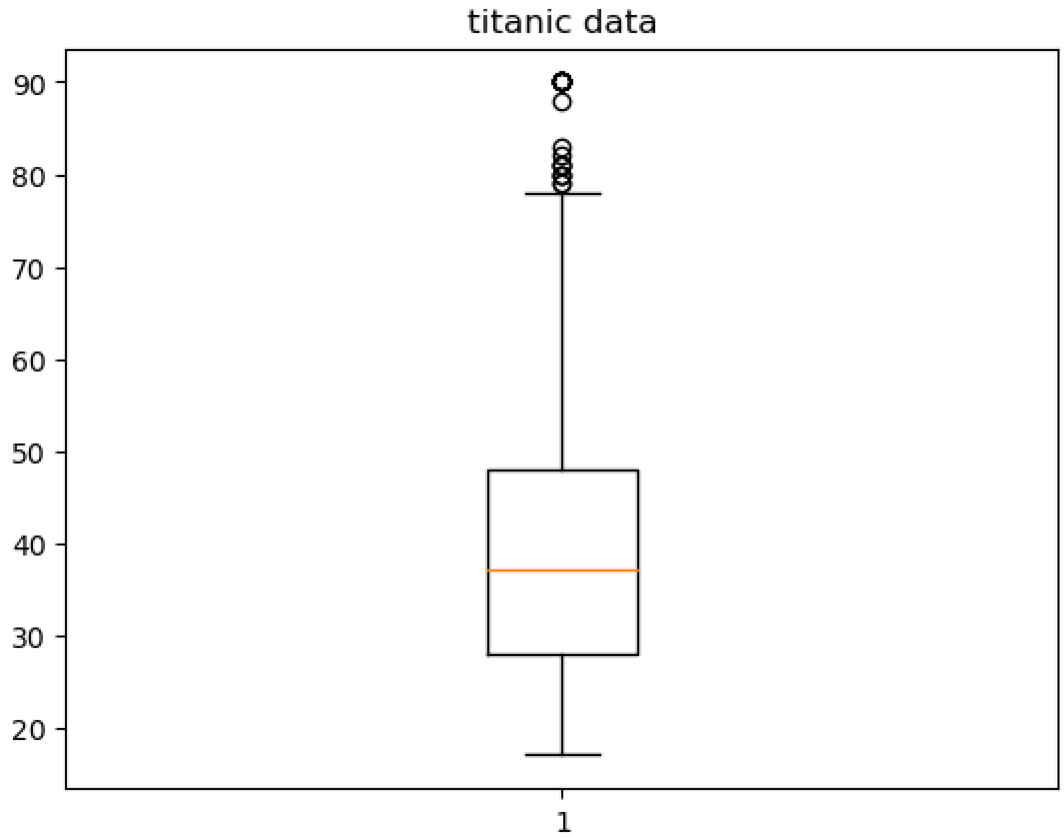
In [15]:
```
# plot multiple histogram for all the numerical column in my data set:
df.plot.hist(subplots = True , layout = (4,5),figsize=(10,20))
# rearrange the plot:
plt.tight_layout()
plt.show()
```



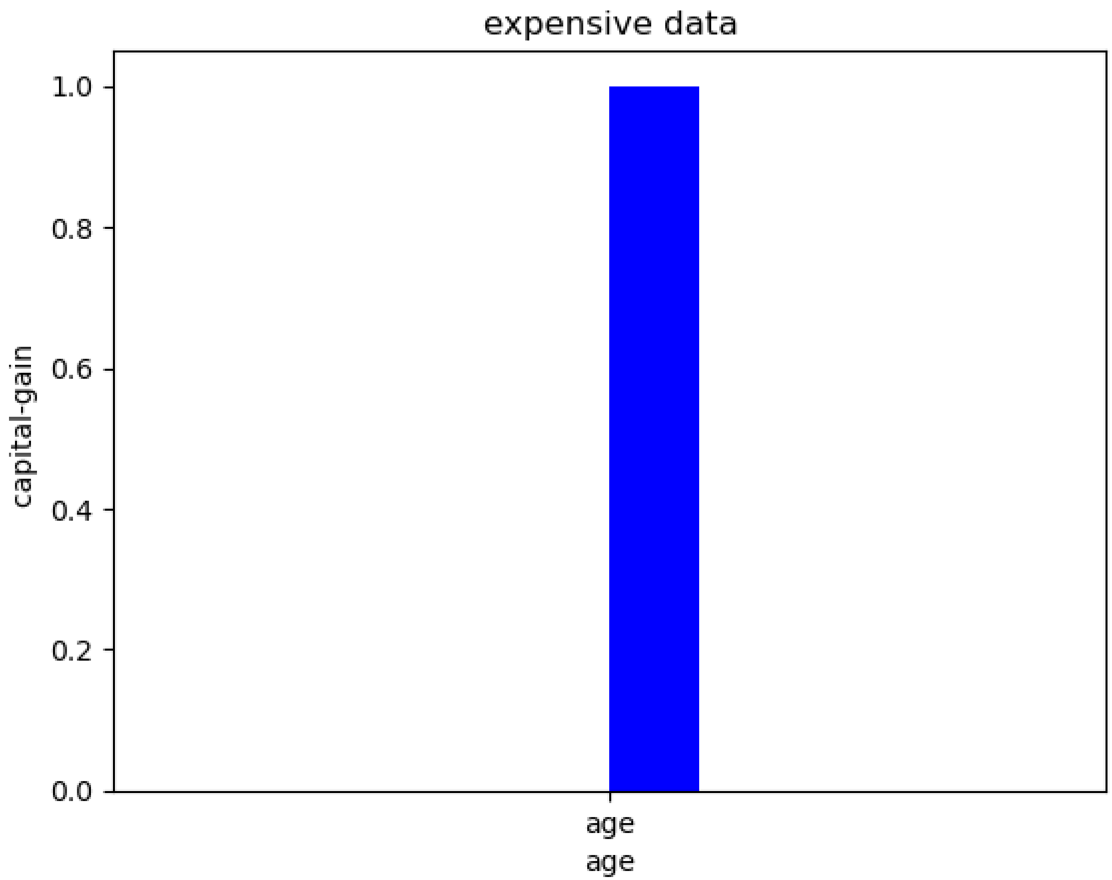
In [21]:
```
# vertical box plot:
plt.boxplot(x = df["age"],vert = False)
plt.title("expensive data")
plt.show()
```



In [23]:
```
# horizontal plot box plot:
plt.boxplot(x = df["age"])
# add a title:
plt.title("titanic data")
# display the plot:
plt.show()
```



In [24]:
```
plt.hist(x =['age'],color ='b')
# add title:
plt.title("expensive data")
# add laebels:
plt.xlabel("age")
plt.ylabel("capital-gain ")
# display the plot
plt.show()
```



In [25]:
```
df.dtypes
```

Out[25]:
```
age                int64
workclass         object
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
Expense           object
dtype: object
```

In [26]:
```
import scipy
from scipy.stats import variation
```

In [29]:
```
# mean:
# trimmed mean
scipy.stats.trim_mean(df['age'] ,proportiontocut = 0.20)
```

Out[29]:
```
37.443666666666665
```

In [31]:
```
# median:
df['age'].median()
```

Out[31]:
```
37.0
```

In [32]:
```
# the first quartile:
Q1=df['age'].quantile(0.85)
Q1
```

Out[32]:
```
54.0
```

In [33]:
```
# IQR:
v1=df['age'].quantile(0.25)
v2=df['age'].quantile(0.55)
IQR = v2-v1
IQR
```

Out[33]:
```
11.0
```

In [35]:
```

# Check for null values in a specific column
null_values = df['age'].isnull().sum()
null_values
```

Out[35]:
```
0
```

In [36]:
```
null = df.isnull()
null
```

Out[36]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4996	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4997	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4998	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4999	False	False	False	False	False	False	False	False	False	False	False	False	False	False

5000 rows × 14 columns

In [37]:
```
# Check for null values in all columns
null_values_all_columns = df.isnull().sum()
null_values_all_columns
```

Out[37]:
```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
Expense           0
dtype: int64
```

In [38]:
```
# Check for null values in all columns
null_values_all_columns = df.isnull().sum().sum()
null_values_all_columns
```

Out[38]:
```
0
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('Expense.csv')
df
```

Out[2]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
0	39	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	15024	0	50	United-States	>50K
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K
2	50	Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	1902	65	United-States	>50K
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K
4	23	Local-gov	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Female	0	0	60	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

In [3]:
```
df.dtypes
```

Out[3]:
```
age                 int64
workclass          object
education          object
education-num       int64
marital-status     object
occupation         object
relationship       object
race               object
sex                object
capital-gain        int64
capital-loss        int64
hours-per-week      int64
native-country     object
Expense            object
dtype: object
```

In [4]:
```
from sklearn.preprocessing import StandardScaler
print('Minimum   value Before Transformation :',df.age .min(),'\n'
     'maximium value Before Transformation:',df.age .max())
# Create an instance
standard_scale = StandardScaler()

# Fit the StandardScaler
# Transform the data

df['capital-gain'] = standard_scale.fit_transform(df[['age']])

print('Minimum   value After Transformation :',df['capital-gain'].min(),'\n',
     'maximium value After Transformation:',df['capital-gain'].max())
```

```
Minimum   value Before Transformation : 17
maximium value Before Transformation: 90
Minimum   value After Transformation : -1.5810851866976907
 maximium value After Transformation: 3.7485795080257773
```

In [5]:
```
# Importing MinMaxNormalization from sklearn
from sklearn.preprocessing import MinMaxScaler

# Create An Instance

min_max = MinMaxScaler()

# Fit and transform of weight column:
df['capital-gain'] = min_max.fit_transform(df[['age']])

#minimum and maximum of Normalization of weight:
df['capital-gain'].min(),df['capital-gain'].max()
```

Out[5]: (0.0, 1.0)

In [6]:
```
# summary statistics:
df.describe()
```

Out[6]:

	age	education-num	capital-gain	capital-loss	hours-per-week
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000
mean	38.656000	10.065000	0.296658	90.032800	40.566200
std	13.698292	2.558141	0.187648	404.168991	12.154191
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.150685	0.000000	40.000000
50%	37.000000	10.000000	0.273973	0.000000	40.000000
75%	48.000000	12.000000	0.424658	0.000000	45.000000
max	90.000000	16.000000	1.000000	3004.000000	99.000000

In [7]:
```
# summary stats for categorical column:
import pandas as pd
import numpy as np

from warnings import filterwarnings
filterwarnings('ignore')
df.describe(include = [np.object])
```

Out[7]:

	workclass	education	marital-status	occupation	relationship	race	sex	native-country	Expense
count	5000	5000	5000	5000	5000	5000	5000	5000	5000
unique	9	16	7	15	6	5	2	40	2
top	Private	HS-grad	Married-civ-spouse	Craft-repair	Husband	White	Male	United-States	<=50K
freq	3444	1602	2294	630	2026	4271	3374	4459	3776

In [8]:
```
df.isnull().sum()
```

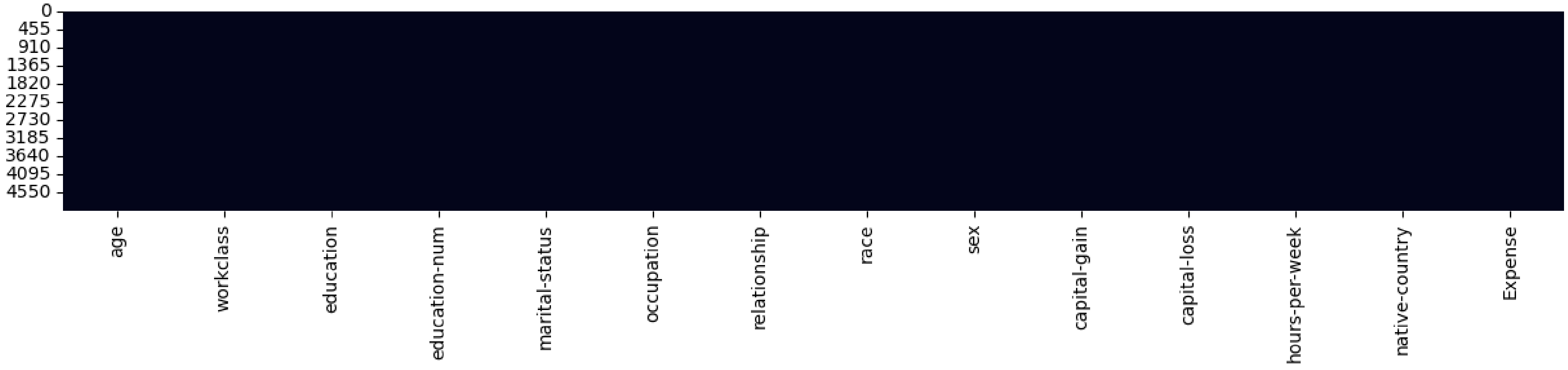
Out[8]:
```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
Expense           0
dtype: int64
```

In [9]:
```
# get the count of null  values:

missing_values = df.isnull().sum()

#check for missing values:

total = df.isnull().sum().sort_values(ascending = False)
# calculate the percentage of missing values:
percent= ((df.isnull().sum()/df.shape[0])*100)

# sort the values in descending order :
percent =  percent.sort_values(ascending = False)

# concate the total missing values:

missing_data = pd.concat([total,percent],axis =1, keys= ['total missing values','percentage of missing values'])


# add  the data types:

missing_data['data[Dtpes]']=df[missing_data.index].dtypes

# view thw missing values:
missing_data
```

Out[9]:

	total missing values	percentage of missing values	data[Dtpes]
age	0	0.0	int64
workclass	0	0.0	object
education	0	0.0	object
education-num	0	0.0	int64
marital-status	0	0.0	object
occupation	0	0.0	object
relationship	0	0.0	object
race	0	0.0	object
sex	0	0.0	object
capital-gain	0	0.0	float64
capital-loss	0	0.0	int64
hours-per-week	0	0.0	int64
native-country	0	0.0	object
Expense	0	0.0	object

In [10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# visualize the null values by means of heatmap:
# 1. set the figure size:
plt.rcParams['figure.figsize']  = [15,2]
# plot the heat map:
sns.heatmap(df.isnull(),cbar = False)

# display the heatmap:
plt.show()
```



In [11]:
```
df
```

Out[11]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
0	39	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.301370	0	50	United-States	>50K
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0.041096	0	40	United-States	<=50K
2	50	Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0.452055	1902	65	United-States	>50K
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0.287671	0	40	United-States	>50K
4	23	Local-gov	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Female	0.082192	0	60	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	38	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0.287671	0	40	United-States	<=50K
4996	26	Private	Some-college	10	Never-married	Tech-support	Own-child	White	Female	0.123288	0	40	United-States	<=50K
4997	20	Private	11th	7	Never-married	Transport-moving	Own-child	White	Male	0.041096	0	60	United-States	<=50K
4998	24	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male	0.095890	0	60	Mexico	>50K
4999	40	Private	HS-grad	9	Divorced	Craft-repair	Not-in-family	White	Male	0.315068	0	45	United-States	<=50K

5000 rows × 14 columns

In [15]:
```
df.isnull().sum()
```

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

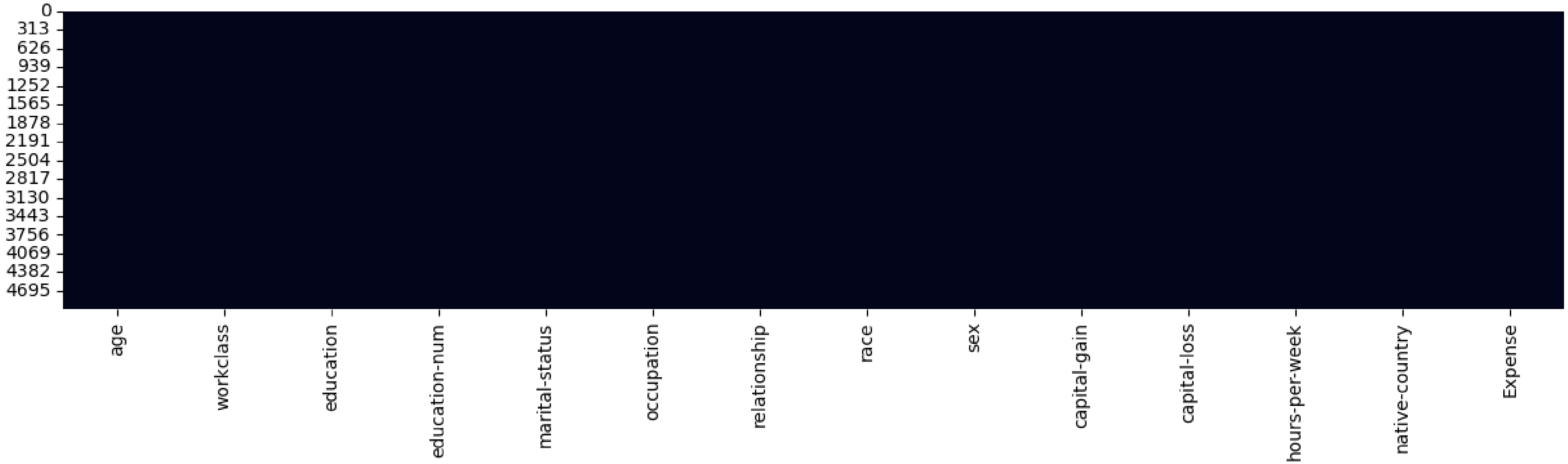
In [16]:
```
# summary stats for categorical column:
import pandas as pd
import numpy as np

from warnings import filterwarnings
filterwarnings('ignore')
df.describe(include = [np.object])
```

Out[16]:

	workclass	education	marital-status	occupation	relationship	race	sex	native-country	Expense
count	5000	5000	5000	5000	5000	5000	5000	5000	5000
unique	9	16	7	15	6	5	2	40	2
top	Private	HS-grad	Married-civ-spouse	Craft-repair	Husband	White	Male	United-States	<=50K
freq	3444	1602	2294	630	2026	4271	3374	4459	3776

In [17]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# visualize the null values by means of heatmap:
# 1. set the figure size:
plt.rcParams['figure.figsize']  = [15,3]
# plot the heat map:
sns.heatmap(df.isnull(),cbar = False)

# display the heatmap:
plt.show()
```



In [19]:
```
df.isnull().sum()
```

Out[19]:
```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
Expense           0
dtype: int64
```

In [20]:
```
df['age'].mean()
```

Out[20]:
```
38.656
```

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

Out[24]:
```
0
```

In [25]:
```
df['capital-gain'].isnull().sum()
```

Out[25]:
```
0
```

In [21]:
```
df['capital-gain'].median()
```

Out[21]:
```
0.27397260273972607
```

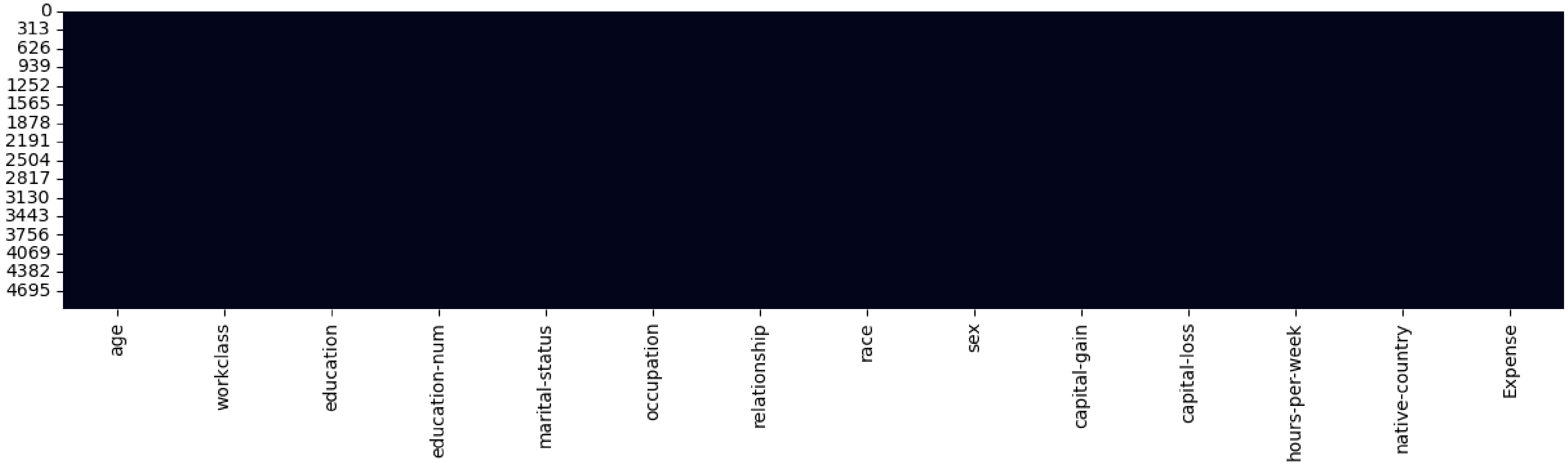
In [22]:
```
df['capital-loss'].std()
```

Out[22]:
```
404.16899118912085
```

In [26]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             5000 non-null   int64  
 1   workclass       5000 non-null   object 
 2   education       5000 non-null   object 
 3   education-num   5000 non-null   int64  
 4   marital-status  5000 non-null   object 
 5   occupation      5000 non-null   object 
 6   relationship    5000 non-null   object 
 7   race            5000 non-null   object 
 8   sex             5000 non-null   object 
 9   capital-gain    5000 non-null   float64
 10  capital-loss    5000 non-null   int64  
 11  hours-per-week  5000 non-null   int64  
 12  native-country  5000 non-null   object 
 13  Expense         5000 non-null   object 
dtypes: float64(1), int64(4), object(9)
memory usage: 547.0+ KB
```

In [27]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# visualize the null values by means of heatmap:
# 1. set the figure size:
plt.rcParams['figure.figsize']  = [15,3]
# plot the heat map:
sns.heatmap(df.isnull(),cbar = False)

# display the heatmap:
plt.show()
```



OUT LAYER DETECTION:

In [28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:
```
df= pd.read_csv('Expense.csv')
df.head()
```

Out[29]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Expense
0	39	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	15024	0	50	United-States	>50K
1	20	Private	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	40	United-States	<=50K
2	50	Private	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	1902	65	United-States	>50K
3	38	State-gov	HS-grad	9	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	40	United-States	>50K
4	23	Local-gov	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Female	0	0	60	United-States	<=50K

In [9]:
```
# filtering out numerical columns from the data set:
# method 1:

df_num = df.select_dtypes(include=['int64', 'float64'])
df_num
```

Out[9]:

	age	education-num	capital-gain	capital-loss	hours-per-week
0	39	13	15024	0	50
1	20	10	0	0	40
2	50	16	0	1902	65
3	38	9	0	0	40
4	23	13	0	0	60
...	...	...	...	...	...
4995	38	9	0	0	40
4996	26	10	0	0	40
4997	20	7	0	0	60
4998	24	9	0	0	60
4999	40	9	0	0	45

5000 rows × 5 columns

In [10]:
```
df_num.columns
```

Out[10]:
```
Index(['age', 'education-num', 'capital-gain', 'capital-loss',
       'hours-per-week'],
      dtype='object')
```

In [11]:
```
# filtering out numerical columns from the data set:
# method 2:
df_num = df_num.select_dtypes(include = 'number')
df_num
```

Out[11]:

	age	education-num	capital-gain	capital-loss	hours-per-week
0	39	13	15024	0	50
1	20	10	0	0	40
2	50	16	0	1902	65
3	38	9	0	0	40
4	23	13	0	0	60
...	...	...	...	...	...
4995	38	9	0	0	40
4996	26	10	0	0	40
4997	20	7	0	0	60
4998	24	9	0	0	60
4999	40	9	0	0	45

5000 rows × 5 columns

In [12]:
```
df_num.columns
```

Out[12]:
```
Index(['age', 'education-num', 'capital-gain', 'capital-loss',
       'hours-per-week'],
      dtype='object')
```

In [13]:
```
# filtering out numerical columns from the data set:
df_cat = df.select_dtypes(include = ['object', 'category'])
df_cat
```

Out[13]:

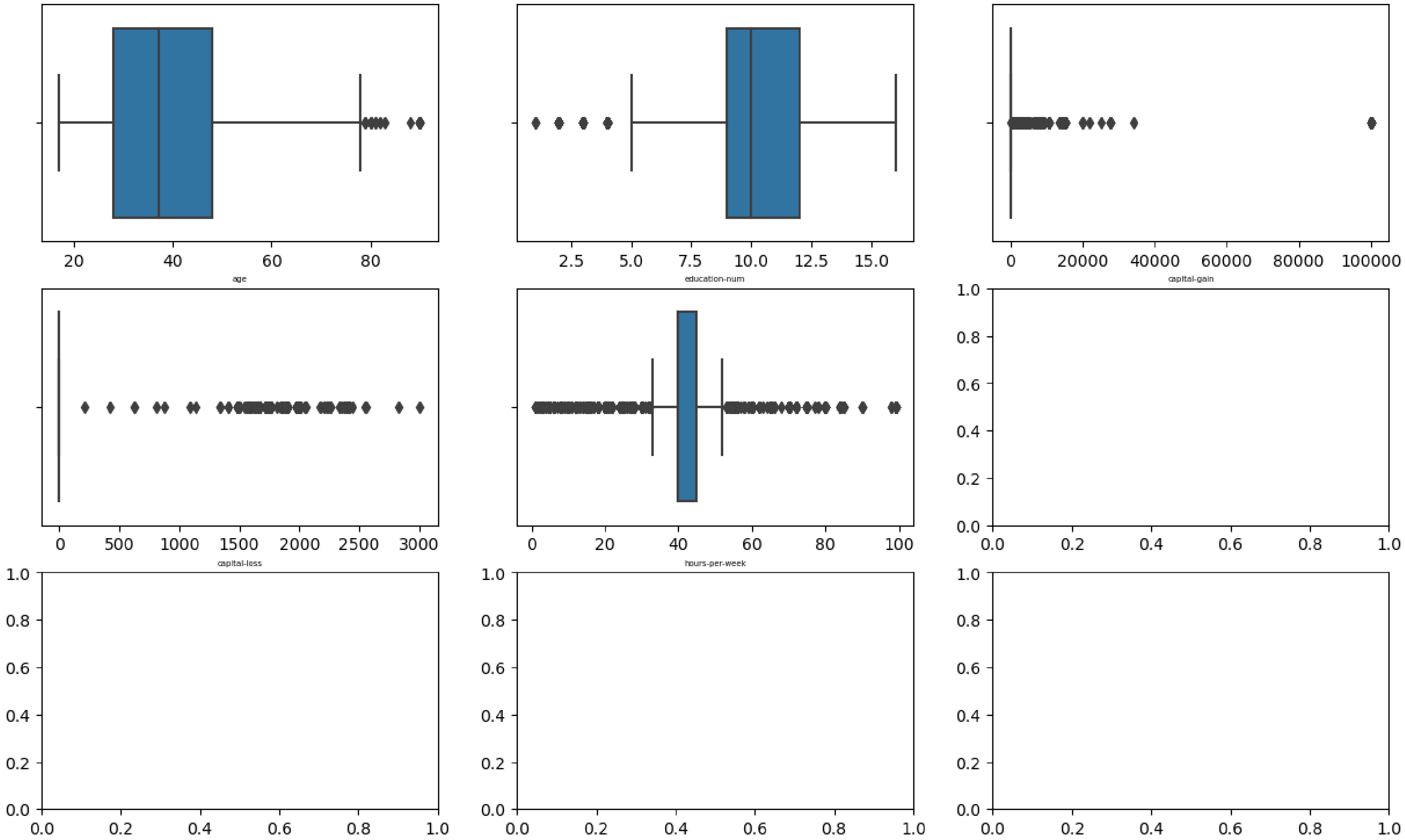
	workclass	education	marital-status	occupation	relationship	race	sex	native-country	Expense
0	Self-emp-inc	Bachelors	Married-civ-spouse	Exec-managerial	Husband	White	Male	United-States	>50K
1	Private	Some-college	Never-married	Other-service	Own-child	White	Male	United-States	<=50K
2	Private	Doctorate	Married-civ-spouse	Prof-specialty	Husband	White	Male	United-States	>50K
3	State-gov	HS-grad	Married-civ-spouse	Prof-specialty	Wife	White	Female	United-States	>50K
4	Local-gov	Bachelors	Never-married	Prof-specialty	Own-child	White	Female	United-States	<=50K
...	...	...	...	...	...	...	...	...	...
4995	Private	HS-grad	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	United-States	<=50K
4996	Private	Some-college	Never-married	Tech-support	Own-child	White	Female	United-States	<=50K
4997	Private	11th	Never-married	Transport-moving	Own-child	White	Male	United-States	<=50K
4998	Private	HS-grad	Married-civ-spouse	Craft-repair	Husband	White	Male	Mexico	>50K
4999	Private	HS-grad	Divorced	Craft-repair	Not-in-family	White	Male	United-States	<=50K

5000 rows × 9 columns

In [14]:
```
df_cat.columns
```

Out[14]:
```
Index(['workclass', 'education', 'marital-status', 'occupation',
       'relationship', 'race', 'sex', 'native-country', 'Expense'],
      dtype='object')
```

In [15]:
```
# to identify the outliers in numerical Columns:

# SUB PLOTS()
fig, ax = plt.subplots(3,3,figsize =(15,9))


for variable,subplot in zip(df_num.columns,ax.flatten()):
    z = sns.boxplot( x = df_num[variable], orient = 'h',whis = 1.5,ax = subplot)

    z.set_xlabel(variable,fontsize = 5)
```


age
education-num
capital-gain
capital-loss
hours-per-week

In [16]:
```
# Filling Out only Categorical column from your Dataset:
from warnings import filterwarnings
filterwarnings('ignore')
df_cat = df.select_dtypes(include = [np.object])
print(df_cat)
# Print heading names of columns contain only numbers
df_num.columns
```

```
          workclass     education       marital-status         occupation  \
0      Self-emp-inc     Bachelors   Married-civ-spouse    Exec-managerial
1           Private  Some-college        Never-married      Other-service
2           Private     Doctorate   Married-civ-spouse     Prof-specialty
3         State-gov       HS-grad   Married-civ-spouse     Prof-specialty
4         Local-gov     Bachelors        Never-married     Prof-specialty
...             ...           ...                  ...                ...
4995        Private       HS-grad   Married-civ-spouse  Machine-op-inspct
4996        Private  Some-college        Never-married       Tech-support
4997        Private          11th        Never-married   Transport-moving
4998        Private       HS-grad   Married-civ-spouse       Craft-repair
4999        Private       HS-grad             Divorced       Craft-repair

        relationship   race     sex native-country Expense
0            Husband  White    Male  United-States    >50K
1          Own-child  White    Male  United-States   <=50K
2            Husband  White    Male  United-States    >50K
3               Wife  White  Female  United-States    >50K
4          Own-child  White  Female  United-States   <=50K
...              ...    ...     ...            ...     ...
4995         Husband  White    Male  United-States   <=50K
4996       Own-child  White  Female  United-States   <=50K
4997       Own-child  White    Male  United-States   <=50K
4998         Husband  White    Male         Mexico    >50K
4999   Not-in-family  White    Male  United-States   <=50K

[5000 rows x 9 columns]
```

Out[16]:
```
Index(['age', 'education-num', 'capital-gain', 'capital-loss',
       'hours-per-week'],
      dtype='object')
```

In [39]:

```
# 1. based on IQR method:
Q1 = df_num.quantile(0.25)


Q3 = df_num.quantile(0.75)

# obtain the type;
IQR =Q3-Q1
IQR
```

Out[39]:

```
age                20.0
education-num       3.0
capital-gain        0.0
capital-loss        0.0
hours-per-week      5.0
dtype: float64
```

In [40]:

```
df_iqr = df[~((df_num<(Q1 - 1.5* IQR))|(df_num>(Q3+1.5*IQR))).any(axis = 1)]

df_iqr.shape
```

Out[40]:

```
(3032, 14)
```

In [41]:

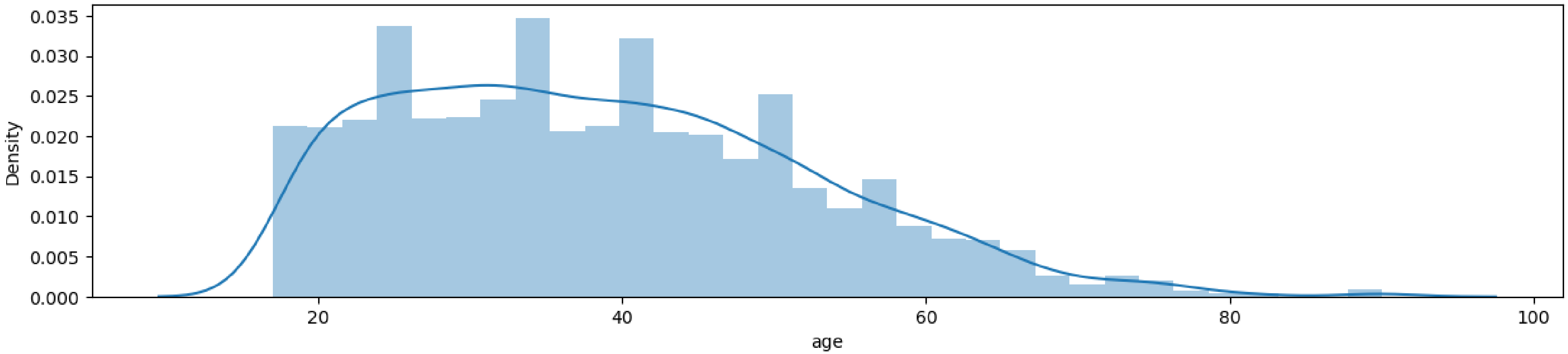
```
df.shape
```

Out[41]:

```
(5000, 14)
```

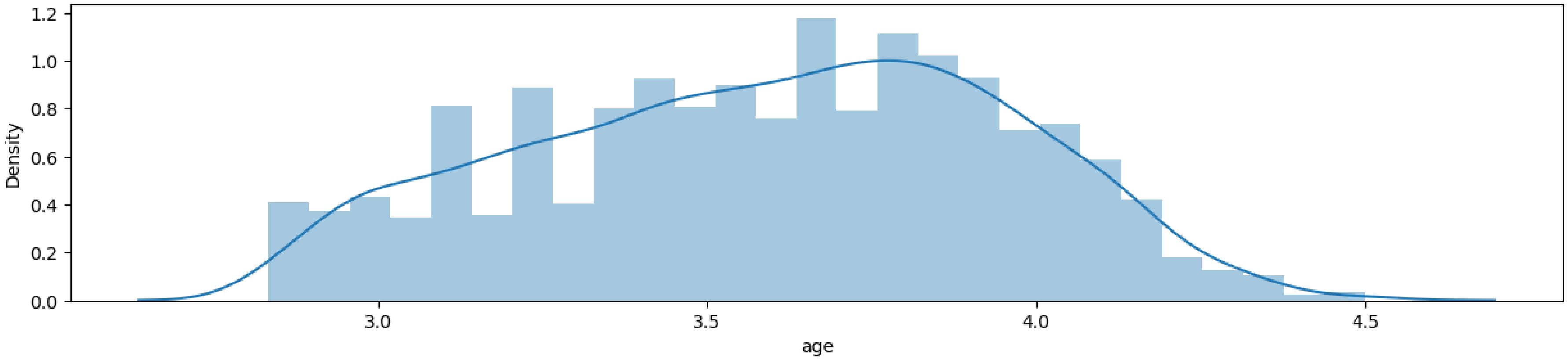
In [42]:
```
# Distribution of the age  column
sns.distplot(df['age'])
plt.ylabel('Density')
print('Skewness : ',df['capital-gain'].skew())
plt.show()
```

```
Skewness :  11.738141362159757
```

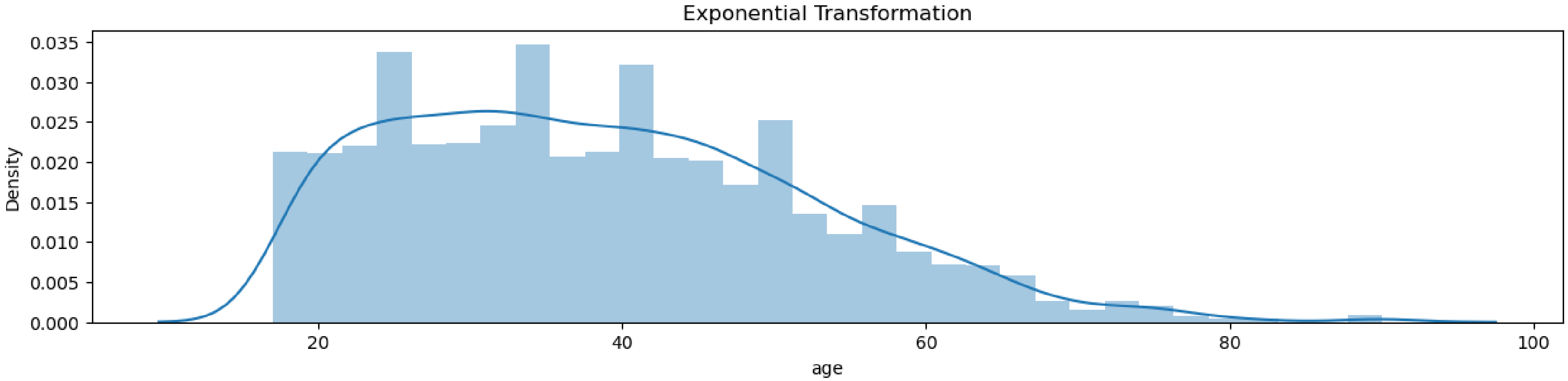


In [45]:
```
# Apply natural log transformation for age column:
log_displacement = np.log(df['age'])
print('Skewness after log Transformtion : ',log_displacement.skew())

sns.distplot(log_displacement)
plt.ylabel('Density')
plt.show()
```

```
Skewness after log Transformtion :  -0.12574343649471817
```



In [46]:
```
# Anti - log or Exponential Transformation
displacement = np.exp(log_displacement)
# plot the Distribution
sns.distplot(displacement)
plt.ylabel('Density')
plt.title('Exponential Transformation')
plt.show()
```



In [ ]: