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

In [2]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
df=pd.read_csv(path)
df
```

Out[2]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training
0	EZYV01	Asia	High School	N	N
1	EZYV02	Asia	Master's	Y	N
2	EZYV03	Asia	Bachelor's	N	Y
3	EZYV04	Asia	Bachelor's	N	N
4	EZYV05	Africa	Master's	Y	N
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	Y
25476	EZYV25477	Asia	High School	Y	N
25477	EZYV25478	Asia	Master's	Y	N
25478	EZYV25479	Asia	Master's	Y	Y
25479	EZYV25480	Asia	Bachelor's	Y	N

25480 rows × 12 columns

In [ ]:
```
#head and it will give top 5 rows give
```

In [3]:
```
# dataframe name:if
df.head()
```

Out[3]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
0	EZYV01	Asia	High School	N	N	
1	EZYV02	Asia	Master's	Y	N	
2	EZYV03	Asia	Bachelor's	N	Y	
3	EZYV04	Asia	Bachelor's	N	N	
4	EZYV05	Africa	Master's	Y	N	

In [ ]:
```
# tails
#last five rows it will give
```

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

Out[4]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

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

Out[5]:
```
(25480, 12)
```

In [6]:
```
print("the number of rows:",df.shape[0])
print("the number of columns:",df.shape[1])
```

```
the number of rows: 25480
the number of columns: 12
```

In [ ]:
```
# size
# how many values or indices are there provided by size
```

In [7]:
```
df.size
```

Out[7]:
```
305760
```

In [ ]:
```
#colunms
# gives the all colunms values
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experienc
e',
       'requires_job_training', 'no_of_employees', 'yr_of_estab',
       'region_of_employment', 'prevailing_wage', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [9]:
```
type(df)
```

Out[9]:
```
pandas.core.frame.DataFrame
```

In [10]:
```
type(df.columns)
```

Out[10]:
```
pandas.core.indexes.base.Index
```

In [ ]:
```
#dtypes
#data tpyes
```

In [11]:
```
df.dtypes

#object means categorical
#other then object numerical (int or float)
```

Out[11]:
```
case_id                  object
continent                object
education_of_employee    object
has_job_experience       object
requires_job_training    object
no_of_employees           int64
yr_of_estab               int64
region_of_employment     object
prevailing_wage         float64
unit_of_wage             object
full_time_position       object
case_status              object
dtype: object
```

In [12]:
```
type(df.dtypes)
```

Out[12]:
```
pandas.core.series.Series
```

In [ ]:
```
#task 1:extract the numerical columns and categorical column seperately by
```

In [13]:
```
# converts above one into dictionary
#key and values
#if for list

d1=dict(df.dtypes)
#for i in d1:
#    if d1[i]=='object':
#        print(i)
cat=[i for i in d1 if d1[i]=='object']
num=[i for i in d1 if d1[i]!='object']
```

In [14]:
```
cat
```

Out[14]:
```
['case_id',
 'continent',
 'education_of_employee',
 'has_job_experience',
 'requires_job_training',
 'region_of_employment',
 'unit_of_wage',
 'full_time_position',
 'case_status']
```

In [15]:
```
num
```

Out[15]:
```
['no_of_employees', 'yr_of_estab', 'prevailing_wage']
```

In [16]:
```
#categorical data available
df.select_dtypes(include='object')
```

Out[16]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 9 columns

In [17]:
```
df.select_dtypes(include='object').columns
```

Out[17]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experienc
e',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [18]:
```
df.select_dtypes(exclude='object').columns
```

Out[18]:
```
Index(['no_of_employees', 'yr_of_estab', 'prevailing_wage'], dtype='objec
t')
```

In [ ]:
```
#isnull

#if data has any missing values or null values
```

In [19]:
```
df.isnull()
#true means (yes)there is a null value
#false means (no) there is a no null value
```

Out[19]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
25475	False	False	False	False	False
25476	False	False	False	False	False
25477	False	False	False	False	False
25478	False	False	False	False	False
25479	False	False	False	False	False

25480 rows × 12 columns

In [ ]:
```
# when you open the excel sheet the data has empty
# which meansthat data is missed
# when you read that using pandas
# at that particular position it display as NULL
```

In [20]:
```
df.isnull().sum()
```

Out[20]:
```
case_id                 0
continent               0
education_of_employee   0
has_job_experience      0
requires_job_training   0
no_of_employees         0
yr_of_estab             0
region_of_employment    0
prevailing_wage         0
unit_of_wage            0
full_time_position      0
case_status             0
dtype: int64
```

In [ ]:
```
# drop duplicate values
```

In [21]:
```
df.drop_duplicates()
```

Out[21]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [ ]:
```
#info
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25480 entries, 0 to 25479
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   case_id                25480 non-null  object 
 1   continent              25480 non-null  object 
 2   education_of_employee  25480 non-null  object 
 3   has_job_experience     25480 non-null  object 
 4   requires_job_training  25480 non-null  object 
 5   no_of_employees        25480 non-null  int64  
 6   yr_of_estab            25480 non-null  int64  
 7   region_of_employment   25480 non-null  object 
 8   prevailing_wage        25480 non-null  float64
 9   unit_of_wage           25480 non-null  object 
 10  full_time_position     25480 non-null  object 
 11  case_status            25480 non-null  object 
dtypes: float64(1), int64(2), object(9)
memory usage: 2.3+ MB
```

In [23]:
```
len(df)
```

Out[23]:
```
25480
```

In [ ]:
```
# bound method

 -you need to keep brackets

# not callable

 -you need to remove the brackets

# attribute error

 -the method is not available

 -check the spelling mistake
```

In [ ]:
```
# we wnt to read some sample of data
# we know head will give top5
# we know tail will give last5
# if you want specific rows or columns
```

In [ ]:
```
# Take_loc_iloc
```

In [24]:
```
df.take([2,3,4])

# 2,3,4 are the colums or rows
# axis=1 reference as columns
# axis=0 reference as rows
# bt default axis=0 rows
```

Out[24]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
2	EZYV03	Asia	Bachelor's	N	Y	
3	EZYV04	Asia	Bachelor's	N	N	
4	EZYV05	Africa	Master's	Y	N	

In [25]:
```
df.take([2,5,7],axis=0)
# python index start with 0
```

Out[25]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
2	EZYV03	Asia	Bachelor's	N	Y	
5	EZYV06	Asia	Master's	Y	N	
7	EZYV08	North America	Bachelor's	Y	N	

In [26]:
```
df.take([100,200,300])
```

Out[26]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	r
100	EZYV101	Asia	Master's	Y	N	
200	EZYV201	Asia	Doctorate	Y	N	
300	EZYV301	Asia	Master's	Y	N	

In [27]:
```
df.take([100,200,300]).take([4,8,11],axis=1)
```

Out[27]:

	requires_job_training	prevailing_wage	case_status
100	N	28243.79	Certified
200	N	74441.11	Certified
300	N	101371.21	Certified

In [ ]:
```
# take does not take rows and columns at a time
```

In [ ]:
```
# iloc
```

In [ ]:
```
# df.iloc[<rows>,<columns>]
# df.iloc[<start:end>,<start:end>]
# rows=[]
# cols=[]
# df.iloc[rows,columns]
```

In [28]:
```
df.iloc[5:10]
```

Out[28]:

	has_job_experience	requires_job_training	no_of_employees	yr_of_estab	region_of_employme
5	Y	N	2339	2012	Sou
6	N	N	4985	1994	Sou
7	Y	N	3035	1924	We
8	N	N	4810	2012	Midwe
9	Y	N	2251	1995	Sou

In [29]:
```
df.iloc[4:10,5:11]
```

Out[29]:

	no_of_employees	yr_of_estab	region_of_employment	prevailing_wage	unit_of_wage	full_
4	1082	2005	South	149907.3900	Year	
5	2339	2012	South	78252.1400	Year	
6	4985	1994	South	53635.3900	Year	
7	3035	1924	West	418.2298	Hour	
8	4810	2012	Midwest	74362.1900	Year	
9	2251	1995	South	67514.7600	Year	

In [30]:
```
df.iloc[10:6,:]
```

Out[30]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_o

In [31]:
```
df.iloc[:,10:6]
```

Out[31]:

0
1
2
3
4
...
25475
25476
25477
25478
25479

25480 rows × 0 columns

In [32]:
```
df.iloc[[100,200,300]]
```

Out[32]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	r
100	EZYV101	Asia	Master's	Y	N	
200	EZYV201	Asia	Doctorate	Y	N	
300	EZYV301	Asia	Master's	Y	N	

In [33]:
```
df.iloc[5:10]# all the columns
df.iloc[5:10,3:6]# specific rows and specific columns
df.iloc[:,2:3]#all the rows
df.iloc[[100,200,300]]
df.iloc[[100,200,300],[7]]
```

Out[33]:

	region_of_employment
100	Northeast
200	West
300	Midwest

In [34]:
```
#only prevailing_wage
df.iloc[[100,200,300],[8]]

#no brackest:series
#barkests is there :data frame
```

Out[34]:

	prevailing_wage
100	28243.79
200	74441.11
300	101371.21

In [35]:
```
#only full time
df.iloc[[100,200,300],[10]]
# directly takes column index
```

Out[35]:

	full_time_position
100	Y
200	Y
300	Y

In [ ]:
```
#loc
```

In [36]:
```
df.loc[[100,200,300],['full_time_position']]
# it directly takes column name
```

Out[36]:

	full_time_position
100	Y
200	Y
300	Y

In [ ]:
```
# Reading a specific column
```

In [38]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)
visa_df
```

Out[38]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [39]:
```
visa_df['continent'] #series format
```

Out[39]:
```
0          Asia
1          Asia
2          Asia
3          Asia
4        Africa
          ...
25475      Asia
25476      Asia
25477      Asia
25478      Asia
25479      Asia
Name: continent, Length: 25480, dtype: object
```

In [40]:
```
visa_df[['continent']] #data frame
```

Out[40]:

	continent
0	Asia
1	Asia
2	Asia
3	Asia
4	Africa
...	...
25475	Asia
25476	Asia
25477	Asia
25478	Asia
25479	Asia

25480 rows × 1 columns

In [41]:
```
visa_df.continent #series
```

Out[41]:
```
0          Asia
1          Asia
2          Asia
3          Asia
4        Africa
          ...
25475      Asia
25476      Asia
25477      Asia
25478      Asia
25479      Asia
Name: continent, Length: 25480, dtype: object
```

In [ ]:
```
visa_df['continent'] #series
visa_df.continent  #series
visa_df[['continent']]# data frame
```

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

Out[42]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experienc
e',
       'requires_job_training', 'no_of_employees', 'yr_of_estab',
       'region_of_employment', 'prevailing_wage', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [43]:
```
cols=['continent','education_of_employee']
visa_df[cols]
```

Out[43]:

	continent	education_of_employee
0	Asia	High School
1	Asia	Master's
2	Asia	Bachelor's
3	Asia	Bachelor's
4	Africa	Master's
...	...	...
25475	Asia	Bachelor's
25476	Asia	High School
25477	Asia	Master's
25478	Asia	Master's
25479	Asia	Bachelor's

25480 rows × 2 columns

In [44]:
```
visa_df.values

# list all the samples
# list all the observation
# list of all the tuples
```

Out[44]:
```
array([['EZYV01', 'Asia', 'High School', ..., 'Hour', 'Y', 'Denied'],
       ['EZYV02', 'Asia', "Master's", ..., 'Year', 'Y', 'Certified'],
       ['EZYV03', 'Asia', "Bachelor's", ..., 'Year', 'Y', 'Denied'],
       ...,
       ['EZYV25478', 'Asia', "Master's", ..., 'Year', 'N', 'Certified'],
       ['EZYV25479', 'Asia', "Master's", ..., 'Year', 'Y', 'Certified'],
       ['EZYV25480', 'Asia', "Bachelor's", ..., 'Year', 'Y', 'Certifie
d']],
      dtype=object)
```

In [ ]:
```
# if i give list====df
# if i give df===list
```

In [ ]:
```
# continent:
```

In [45]:
```
l1=[1,2,3]
l2=['A','B','C']
L=[l1,l2]
L
pd.DataFrame(L)
```

Out[45]:

	0	1	2
0	1	2	3
1	A	B	C

In [47]:
```
cols=['continent',L]
pd.DataFrame(L,cols)
```

Out[47]:

	0	1	2
continent	1	2	3
[[1, 2, 3], [A, B, C]]	A	B	C

In [ ]:
```
# unique
```

In [48]:
```
#how many unique lables are there
visa_df['continent'].unique()
```

Out[48]:
```
array(['Asia', 'Africa', 'North America', 'Europe', 'South America',
       'Oceania'], dtype=object)
```

In [49]:
```
#python basic logics
l1=['A','A','B','C']
set(l1)
```

Out[49]:
```
{'A', 'B', 'C'}
```

In [50]:
```
visa_df['continent'].values
```

Out[50]:
```
array(['Asia', 'Asia', 'Asia', ..., 'Asia', 'Asia', 'Asia'], dtype=object)
```

In [51]:
```
#nunique
```

In [52]:
```
visa_df['continent'].nunique()
#number of unique elements
```

Out[52]:
```
6
```

In [ ]:
```
#> in the continent column only 7 elemsnts repeated
#['Asia', 'Africa', 'North America', 'Europe', 'South America',
# 'Oceania'], dtype=object)
# out of total observation how many asia observation are there
```

In [53]:
```
con=visa_df['continent']=='Asia'  #True  and False
visa_df[con]
```

Out[53]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
5	EZYV06	Asia	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

16861 rows × 12 columns

In [54]:
```
con=visa_df['continent']=='Asia'  #True  and False
len(visa_df[con])
```

Out[54]: 16861

In [55]:
```
con=visa_df['continent']=='Africa'  #True  and False
visa_df[con]
```

Out[55]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
4	EZYV05	Africa	Master's	Y	
18	EZYV19	Africa	Master's	Y	
74	EZYV75	Africa	Master's	Y	
194	EZYV195	Africa	Master's	Y	
242	EZYV243	Africa	Bachelor's	N	
...	...	...	...	...	
25385	EZYV25386	Africa	Doctorate	Y	
25408	EZYV25409	Africa	Master's	Y	
25443	EZYV25444	Africa	Bachelor's	N	
25446	EZYV25447	Africa	Master's	N	
25474	EZYV25475	Africa	Doctorate	N	

551 rows × 12 columns

In [56]:
```
con=visa_df['continent']=='Africa'  #True  and False
len(visa_df[con])
```

Out[56]:
```
551
```

In [57]:
```
unique_labels=visa_df['continent'].unique()
for i in unique_labels:
    con=visa_df['continent']==i
    print(i,":",len(visa_df[con]))
```

```
Asia : 16861
Africa : 551
North America : 3292
Europe : 3732
South America : 852
Oceania : 192
```

In [58]:
```
unique_labels=visa_df['continent'].unique()
count=[]
for i in unique_labels:
    con=visa_df['continent']==i
    count.append(len(visa_df[con]))
pd.DataFrame(zip(unique_labels,count),
             columns=['continent','count'])
```

Out[58]:

	continent	count
0	Asia	16861
1	Africa	551
2	North America	3292
3	Europe	3732
4	South America	852
5	Oceania	192

In [59]:
```
unique_labels=visa_df['continent'].unique()
count=[]
for i in unique_labels:
    con=visa_df['continent']==i
    count.append(len(visa_df[con]))
continent_df=pd.DataFrame(zip(unique_labels,count),
             columns=['continent','count'])
continent_df.to_csv('continent_df.csv',index=False)
```

In [ ]:
```
visa_df #total data frame
visa_df['continent'] #specific column
visa_df['continent']=='Asia' # specific label
#########################
```

In [ ]:
```
#value_counts
```

In [60]:
```
visa_df['continent'].value_counts()  #series
```

Out[60]:
```
continent
Asia             16861
Europe            3732
North America     3292
South America      852
Africa             551
Oceania            192
Name: count, dtype: int64
```

In [61]:
```
continent_vc=visa_df['continent'].value_counts()
continent_vc
```

Out[61]:
```
continent
Asia             16861
Europe            3732
North America     3292
South America      852
Africa             551
Oceania            192
Name: count, dtype: int64
```

In [ ]:
```
visa_df
visa_df['continent']
visa_df['continent'].unique()
visa_df['continent'].nunique()
visa_df['continent'].value_count()
```

In [62]:
```
continent_vc.keys()
```

Out[62]:
```
Index(['Asia', 'Europe', 'North America', 'South America', 'Africa',
       'Oceania'],
      dtype='object', name='continent')
```

In [63]:
```
continent_vc.values
```

Out[63]:
```
array([16861,  3732,  3292,   852,   551,   192], dtype=int64)
```

In [64]:
```
continent_vc=visa_df['continent'].value_counts()
l1=continent_vc.keys()
l2=continent_vc.values
pd.DataFrame(zip(l1,l2),columns=['continent','count'])
```

Out[64]:

	continent	count
0	Asia	16861
1	Europe	3732
2	North America	3292
3	South America	852
4	Africa	551
5	Oceania	192

In [65]:
```
continent_vc=visa_df['continent'].value_counts()
l1=continent_vc.keys()
l2=continent_vc.values
continent_vc_df=pd.DataFrame(zip(l1,l2),columns=['continent','count'])
continent_vc_df
```

Out[65]:

	continent	count
0	Asia	16861
1	Europe	3732
2	North America	3292
3	South America	852
4	Africa	551
5	Oceania	192

In [66]:
```
continent_vc
```

Out[66]:
```
continent
Asia             16861
Europe            3732
North America     3292
South America      852
Africa             551
Oceania            192
Name: count, dtype: int64
```

In [67]:
```
continent_df
```

Out[67]:

	continent	count
0	Asia	16861
1	Africa	551
2	North America	3292
3	Europe	3732
4	South America	852
5	Oceania	192

In [68]:
```
# Bar chart:
```

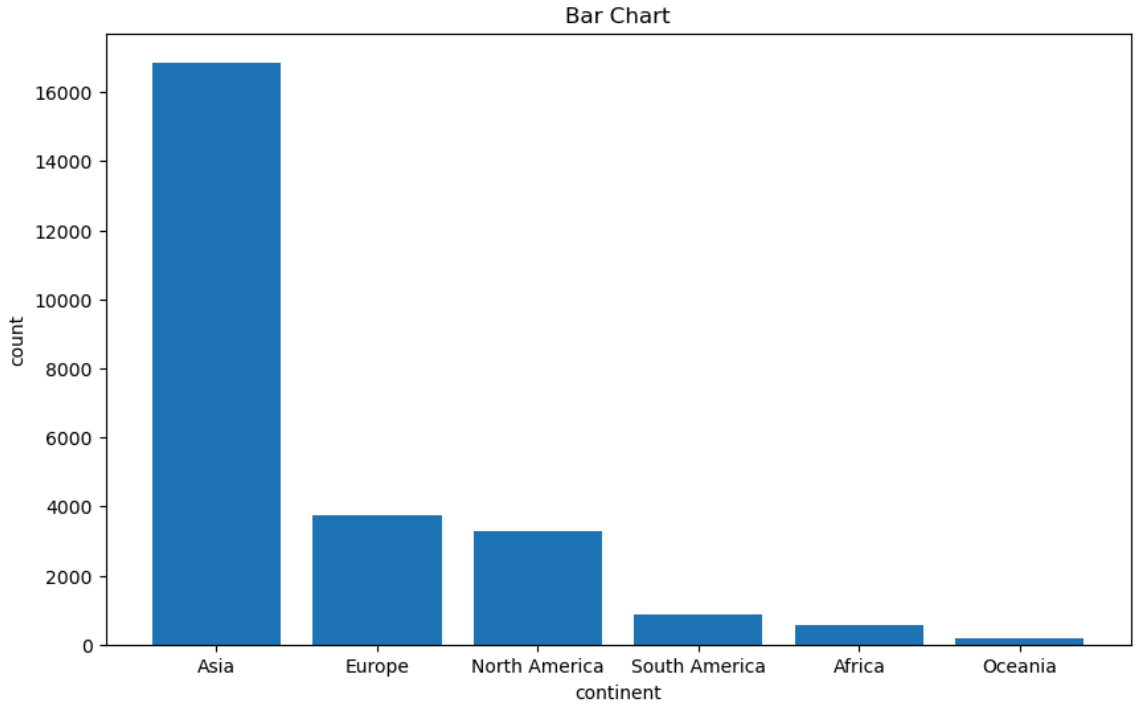
In [ ]:
```
-in order to draw bar chart
-we required one categorical colum
-we requried one  numerical colum
-package:matplotlib
-dataframe:continent_vc_df
```

In [69]:
```
#plt.bar(<cat>,<numer>,<data>)
continent_vc_df
```

Out[69]:

	continent	count
0	Asia	16861
1	Europe	3732
2	North America	3292
3	South America	852
4	Africa	551
5	Oceania	192

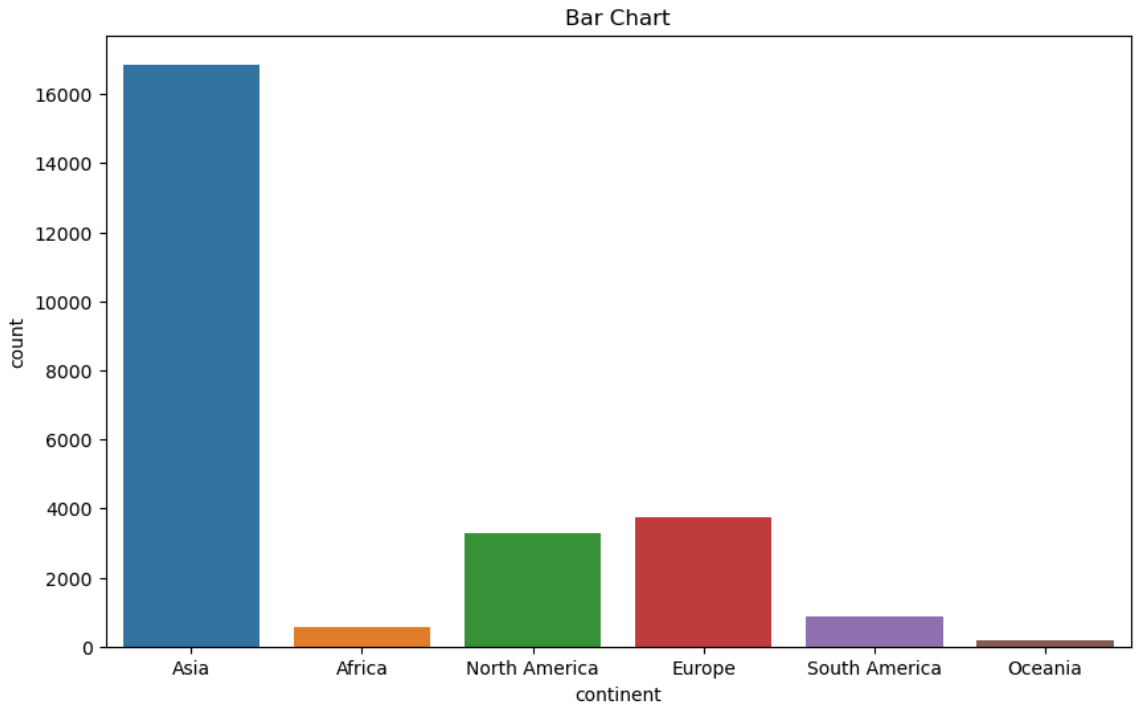
In [70]:
```
plt.figure(figsize=(10,6))  # increse the plot sixze
plt.bar('continent','count',
        data=continent_vc_df)
plt.xlabel('continent')  # x-axis name
plt.ylabel('count')      #y-axis name
plt.title("Bar Chart")#title of the chart
plt.savefig('continent_bar.jpg')
plt.show()
```



In [ ]:
```
-we read the data
-we read categprical colum
-we made frequency table by using value counts
-we plot the bar chart ussimg matplotlib
-but matplotlib requried 3 arguments


- x label:categorical column (width)
-y lable :numerical column (height)
-data:frequency table name
```

In [ ]:
```
# count plot:

- count plot can use bt seaborn package
- it requries only entire dataframe and categorical column
- entire dataframe name:visadf
-catrgorical column name:continent
-order;in which order
```

In [71]:
```
plt.figure(figsize=(10,6))
sns.countplot(data=visa_df,x='continent')
plt.xlabel('continent')  # x-axis name
plt.ylabel('count')      #y-axis name
plt.title("Bar Chart")
plt.savefig('continent_bar.jpg')
plt.show()
```



In [ ]:
```
#  prevailing wage
```

In [72]:
```
visa_df['prevailing_wage']
```

Out[72]:
```
0           592.2029
1         83425.6500
2        122996.8600
3         83434.0300
4        149907.3900
              ...
25475     77092.5700
25476    279174.7900
25477    146298.8500
25478     86154.7700
25479     70876.9100
Name: prevailing_wage, Length: 25480, dtype: float64
```

In [ ]:
```
# count
```

In [73]:
```
len(visa_df['prevailing_wage'])
```

Out[73]: 25480

In [74]:
```
visa_df['prevailing_wage'].count()
```

Out[74]: 25480

In [ ]:
```
# Mean
```

In [75]:
```
visa_df['prevailing_wage'].mean()  #pandas
```

Out[75]: 74455.81459209183

In [76]:
```
np.mean(visa_df['prevailing_wage'])
```

Out[76]: 74455.81459209183

In [ ]:
```
# Median
```

In [77]:
```
visa_df['prevailing_wage'].median()
```

Out[77]: 70308.20999999999

In [78]:
```
np.median(visa_df['prevailing_wage'])
```

Out[78]: 70308.20999999999

In [79]:
```
visa_df['prevailing_wage'].max()
```

Out[79]: 319210.27

In [80]:
```
np.max(visa_df['prevailing_wage'])
```

Out[80]: 319210.27

In [81]:
```
visa_df['prevailing_wage'].min()
```

Out[81]:
```
2.1367
```

In [82]:
```
np.min(visa_df['prevailing_wage'])
```

Out[82]:
```
2.1367
```

In [ ]:
```
#std
```

In [83]:
```
visa_df['prevailing_wage'].std()
```

Out[83]:
```
52815.94232687357
```

In [84]:
```
#all together

wage_count=round(visa_df['prevailing_wage'].count(),2)
wage_max=round(visa_df['prevailing_wage'].max(),2)
wage_min=round(visa_df['prevailing_wage'].min(),2)
wage_mean=round(visa_df['prevailing_wage'].mean(),2)
wage_median=round(visa_df['prevailing_wage'].median(),2)
wage_std=round(visa_df['prevailing_wage'].std(),2)

l=[wage_count,wage_max,wage_min,wage_mean,wage_median,wage_std]
cols=['prevailing_wage']
index=['count','max','min','mean','median','std']
pd.DataFrame(l,columns=cols,index=index)
```

Out[84]:

	prevailing_wage
count	25480.00
max	319210.27
min	2.14
mean	74455.81
median	70308.21
std	52815.94

In [ ]:
```
# Percentile_quantile

- percentile and quantile available in numpy

-np.percentile()
-column name
-percentile value between 0 to 100

-np.quantile()

-column name
-0 to 1
-in quantile 0.25 means 25percentile
```

In [85]:
```
np.percentile(visa_df['prevailing_wage'],25)
```

Out[85]: 34015.479999999996

In [86]:
```
np.quantile(visa_df['prevailing_wage'],0.25)
```

Out[86]: 34015.479999999996

In [ ]:
```
## what is the meaning of 25 percentile =34015.47

25 percentile of total data gas wage less than 34015.47
-find 25 percentage of total data=25*25480/100=6370
-6370 person wages has less than 34014
```

In [ ]:
```
# I need 25 percentage of data

25*25480/100

36370 person wages less than 34015
```

In [87]:
```
con=visa_df['prevailing_wage']<34014
len(visa_df[con])
```

Out[87]: 6370

In [ ]:
```
# can you valid for 50p data
```

In [88]:
```
np.percentile(visa_df['prevailing_wage'],50)
```

Out[88]: 70308.20999999999

In [89]:
```
con=visa_df['prevailing_wage']<17577
len(visa_df[con])
```

Out[89]: 4037

In [90]:
```
#all together:

wage_count=round(visa_df['prevailing_wage'].count(),2)
wage_max=round(visa_df['prevailing_wage'].max(),2)
wage_min=round(visa_df['prevailing_wage'].min(),2)
wage_mean=round(visa_df['prevailing_wage'].mean(),2)
wage_median=round(visa_df['prevailing_wage'].median(),2)
wage_std=round(visa_df['prevailing_wage'].std(),2)
wage_25=np.percentile(visa_df['prevailing_wage'],25)
wage_50=np.percentile(visa_df['prevailing_wage'],50)
wage_75=np.percentile(visa_df['prevailing_wage'],75)

l=[wage_count,wage_max,wage_min,wage_mean,wage_median,wage_std,wage_25,wage_
cols=['prevailing_wage']
index=['count','max','min','mean','median','std','25%','50%','75%']
pd.DataFrame(l,columns=cols,index=index)
```

Out[90]:

	prevailing_wage
count	25480.0000
max	319210.2700
min	2.1400
mean	74455.8100
median	70308.2100
std	52815.9400
25%	34015.4800
50%	70308.2100
75%	107735.5125

In [91]:
```
visa_df.describe()

# numerical columns
```

Out[91]:

	no_of_employees	yr_of_estab	prevailing_wage
count	25480.000000	25480.000000	25480.000000
mean	5667.043210	1979.409929	74455.814592
std	22877.928848	42.366929	52815.942327
min	-26.000000	1800.000000	2.136700
25%	1022.000000	1976.000000	34015.480000
50%	2109.000000	1997.000000	70308.210000
75%	3504.000000	2005.000000	107735.512500
max	602069.000000	2016.000000	319210.270000

In [92]:
```
cols=visa_df.select_dtypes(exclude='object').columns
l=[]
for i in cols:

    count=round(visa_df[i].count(),2)
    maxx=round(visa_df[i].max(),2)
    minn=round(visa_df[i].min(),2)
    mean=round(visa_df[i].mean(),2)
    median=round(visa_df[i].median(),2)
    std=round(visa_df[i].std(),2)
    p_25=np.percentile(visa_df[i],25)
    p_50=np.percentile(visa_df[i],50)
    p_75=np.percentile(visa_df[i],75)

    l.append([count,maxx,minn,mean,median,std,p_25,p_50,p_75])
print(l)
index=['count','max','min','mean','median','std','25%','50%','75%']
pd.DataFrame(zip(l[0],l[1],l[2]),columns=cols,index=index)
```

```
[[25480, 602069, -26, 5667.04, 2109.0, 22877.93, 1022.0, 2109.0, 3504.0],
[25480, 2016, 1800, 1979.41, 1997.0, 42.37, 1976.0, 1997.0, 2005.0], [2548
0, 319210.27, 2.14, 74455.81, 70308.21, 52815.94, 34015.479999999996, 7030
8.20999999999, 107735.51250000001]]
```

Out[92]:

	no_of_employees	yr_of_estab	prevailing_wage
count	25480.00	25480.00	25480.0000
max	602069.00	2016.00	319210.2700
min	-26.00	1800.00	2.1400
mean	5667.04	1979.41	74455.8100
median	2109.00	1997.00	70308.2100
std	22877.93	42.37	52815.9400
25%	1022.00	1976.00	34015.4800
50%	2109.00	1997.00	70308.2100
75%	3504.00	2005.00	107735.5125

In [ ]:
```
##steps in outlier analysis**
-
-step-1: find q1,q2,q3
    - np.percentile(column data,q)

-step-2:calculate the IQR
    -IQR=Q3-Q1

-step-3: calculate lower boundary and upper boundary
    -lb:Q1-1.5IQR

    -ub:Q3+1.5IQR

-Step-4: find the outerliersdf

    c1:column data <lb
    c2:column data >ub
    c:apply the main condition
    -main data(c)
```

In [93]:
```
visa_df.select_dtypes(exclude='object').columns
```

Out[93]:
```
Index(['no_of_employees', 'yr_of_estab', 'prevailing_wage'], dtype='objec
t')
```

In [94]:
```
visa_df['prevailing_wage']
```

Out[94]:
```
0           592.2029
1         83425.6500
2        122996.8600
3         83434.0300
4        149907.3900
              ...
25475     77092.5700
25476    279174.7900
25477    146298.8500
25478     86154.7700
25479     70876.9100
Name: prevailing_wage, Length: 25480, dtype: float64
```

In [95]:
```
Q1=np.percentile(visa_df['prevailing_wage'],25)
Q1
```

Out[95]:
```
34015.479999999996
```

In [96]:
```
Q2=np.percentile(visa_df['prevailing_wage'],50)
Q2
```

Out[96]:
```
70308.20999999999
```

In [97]:
```
Q3=np.percentile(visa_df['prevailing_wage'],75)
Q3
```

Out[97]:
```
107735.51250000001
```

In [98]:
```
#########step-1#################

Q1=np.percentile(visa_df['prevailing_wage'],25)
Q2=np.percentile(visa_df['prevailing_wage'],50)
Q3=np.percentile(visa_df['prevailing_wage'],75)

##############step-2##############

IQR=Q3-Q1


#########STEP-3#############
lb=Q1-1.5*IQR
ub=Q3+1.5*IQR

##########step-4####
c1=visa_df['prevailing_wage']<lb
c2=visa_df['prevailing_wage']>ub
con=c1|c2

############step-5############

outliers_df=visa_df[con]
outliers_df
```

Out[98]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
14	EZYV15	Asia	Master's	Y	
34	EZYV35	Asia	Master's	N	
130	EZYV131	South America	High School	N	
216	EZYV217	Asia	Master's	Y	
221	EZYV222	North America	Doctorate	Y	
...	...	...	...	...	
25191	EZYV25192	Asia	Master's	N	
25195	EZYV25196	North America	Master's	Y	
25468	EZYV25469	Asia	Bachelor's	N	
25469	EZYV25470	North America	Master's	Y	
25476	EZYV25477	Asia	High School	Y	

427 rows × 12 columns

In [99]:
```
Q1=np.percentile(visa_df['prevailing_wage'],25)
Q2=np.percentile(visa_df['prevailing_wage'],50)
Q3=np.percentile(visa_df['prevailing_wage'],75)

##############step-2##############

IQR=Q3-Q1


#########STEP-3##############
lb=Q1-1.5*IQR
ub=Q3+1.5*IQR

##########step-4#####
c1=visa_df['prevailing_wage']<lb
c2=visa_df['prevailing_wage']>ub
con=c1|c2

############step-5############

outliers_df=visa_df[con]
outliers_df

##############step-6######
c1=visa_df['prevailing_wage']>lb
c2=visa_df['prevailing_wage']<ub
con=c1&c2
non_outliers_df=visa_df[c1&c2]
non_outliers_df
```

0	EZYV01	Asia	High School	N
1	EZYV02	Asia	Master's	Y
2	EZYV03	Asia	Bachelor's	N
3	EZYV04	Asia	Bachelor's	N
4	EZYV05	Africa	Master's	Y
...	...	...	...	...
25474	EZYV25475	Africa	Doctorate	N
25475	EZYV25476	Asia	Bachelor's	Y
25477	EZYV25478	Asia	Master's	Y
25478	EZYV25479	Asia	Master's	Y
25479	EZYV25480	Asia	Bachelor's	Y

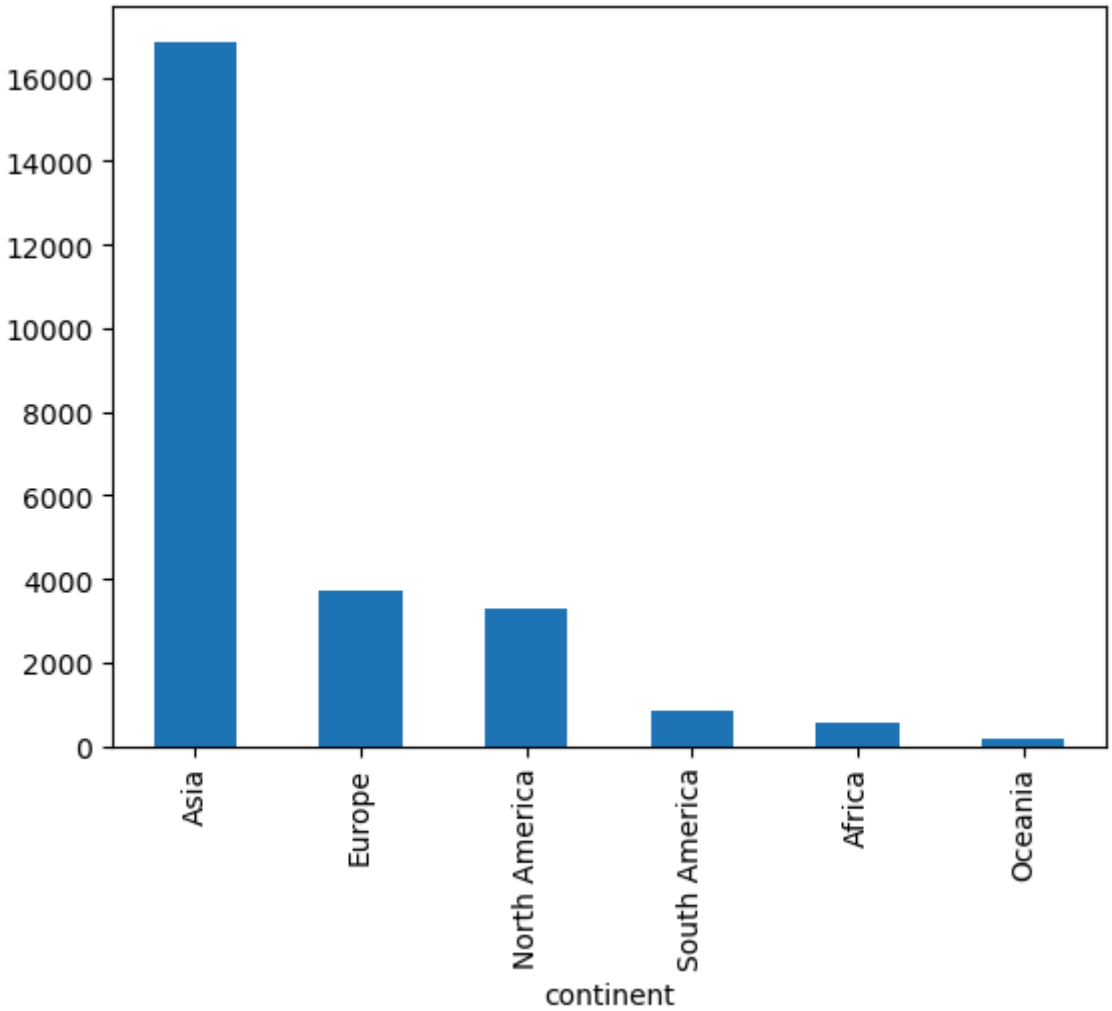
25053 rows × 12 columns

In [ ]:
```
# method-3:

-we created a frequency table:matplotlib

-we created bar chart using seaborn

-main dataframe

-column name

-by using value counts
```

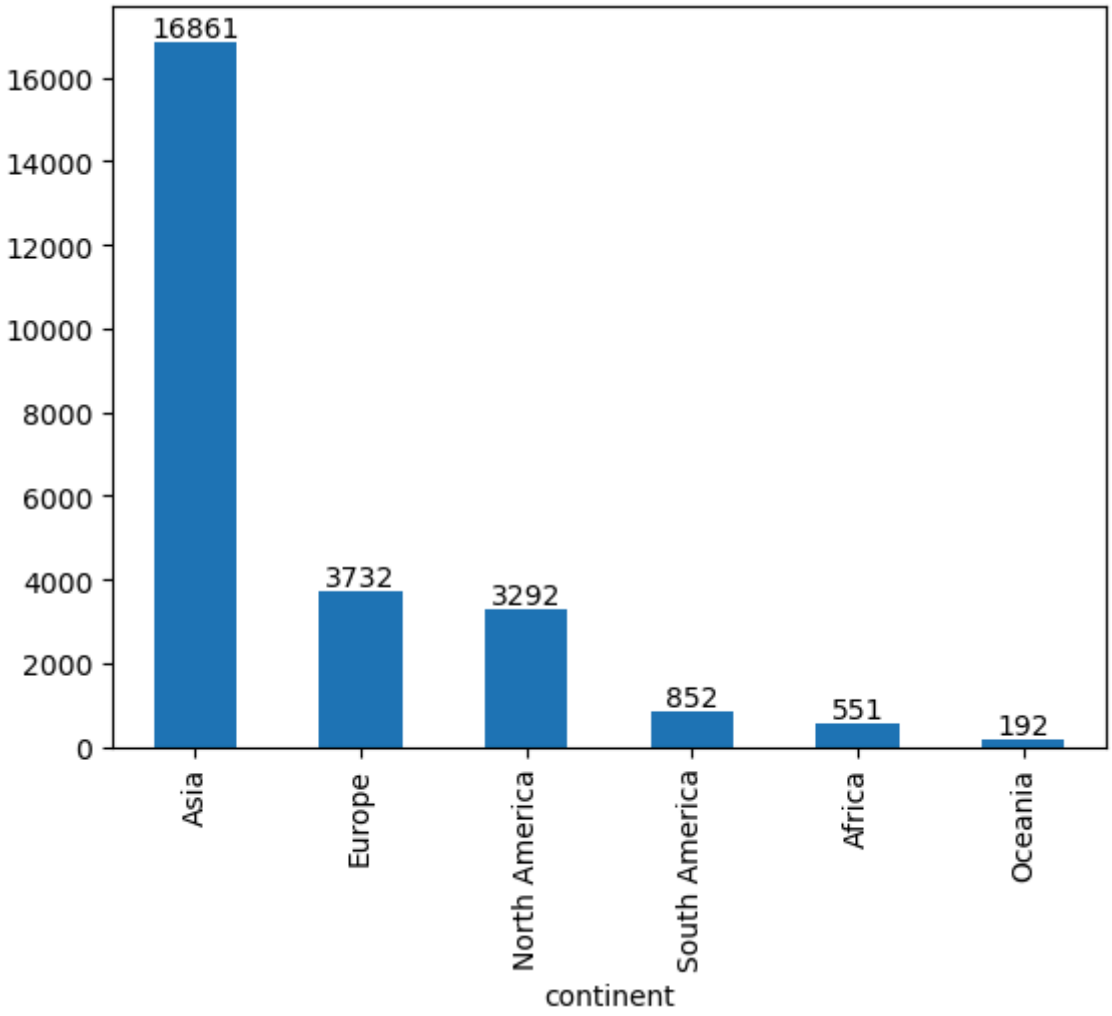
In [114]:
```
visa_df['continent'].value_counts()
```

Out[114]:
```
continent
Asia             16861
Europe            3732
North America     3292
South America      852
Africa             551
Oceania            192
Name: count, dtype: int64
```

In [115]:
```
values=visa_df['continent'].value_counts()
ax=values.plot(kind='bar')
```

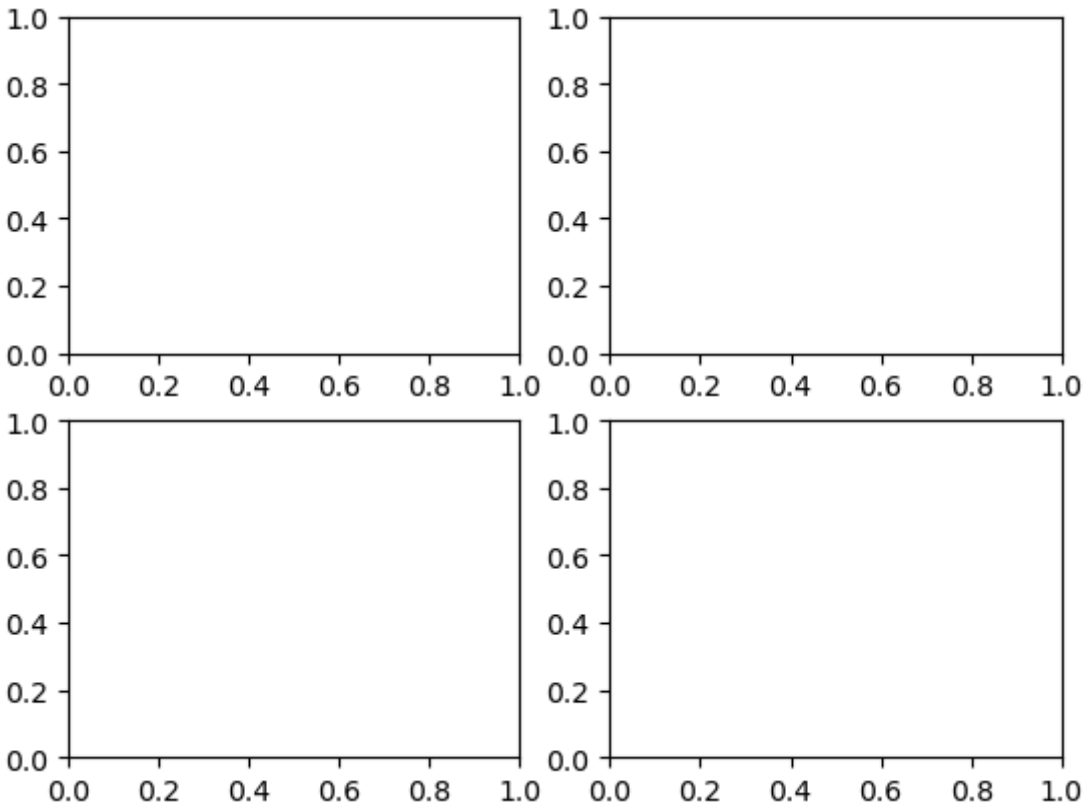


In [116]:
```
values=visa_df['continent'].value_counts()
ax=values.plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.show()
```



In [117]:
```
plt.subplot(2,2,1)
plt.subplot(2,2,2)
plt.subplot(2,2,3)
plt.subplot(2,2,4)
```

Out[117]:
```
<Axes: >
```



In [ ]:
```
# method 3 for bar garph:

values=visa_df['continent'].value_counts()
ax=values.plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.show()
```

In [ ]:
```
# relative frequency
```

In [118]:
```
visa_df['continent'].value_counts(normalize=True)
```

Out[118]:
```
continent
Asia             0.661735
Europe           0.146468
North America    0.129199
South America    0.033438
Africa           0.021625
Oceania          0.007535
Name: proportion, dtype: float64
```

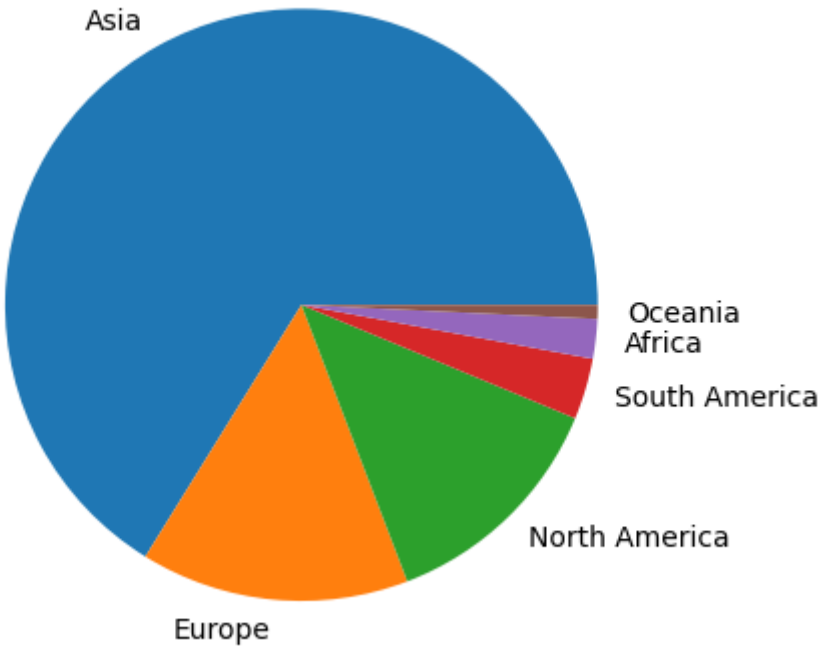
In [ ]:
```
# Pie Chart:
-pie chart will autometically converts value to percentage

-will take value count help with out normalize

- x is data in the form list

-lables also in the form of list
```

In [119]:
```
keys=visa_df['continent'].value_counts().keys()
values=visa_df['continent'].value_counts().values
values
```

Out[119]:
```
array([16861,  3732,  3292,   852,   551,   192], dtype=int64)
```

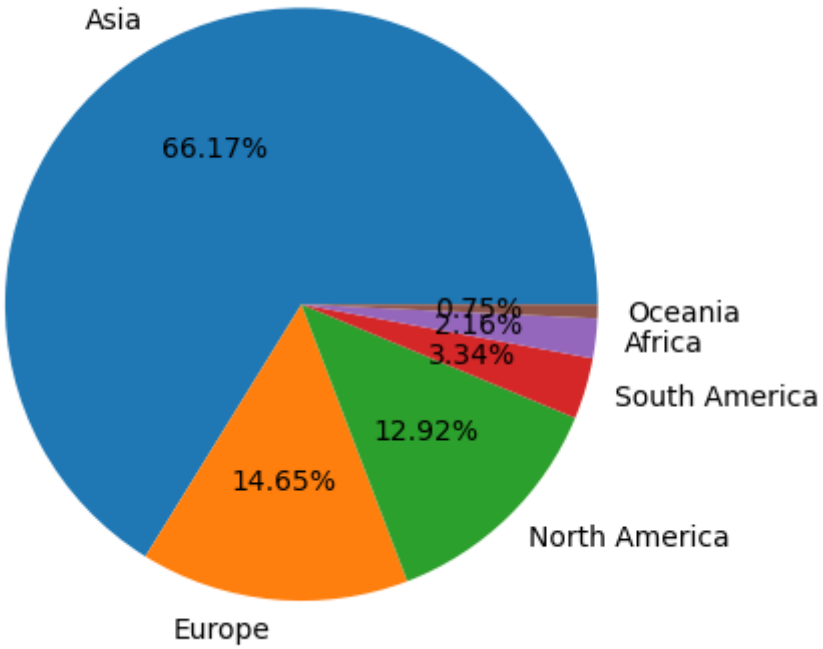
In [120]:
```
plt.pie(values,labels=keys)
```

Out[120]:
```
([<matplotlib.patches.Wedge at 0x2c560232250>,
  <matplotlib.patches.Wedge at 0x2c56023dad0>,
  <matplotlib.patches.Wedge at 0x2c56023c750>,
  <matplotlib.patches.Wedge at 0x2c56023c7d0>,
  <matplotlib.patches.Wedge at 0x2c56023e990>,
  <matplotlib.patches.Wedge at 0x2c56041bdd0>],
 [Text(-0.5351743362316361, 0.9610350825224997, 'Asia'),
  Text(-0.10373513115748138, -1.0950977228374372, 'Europe'),
  Text(0.7670026411947619, -0.7884839557024984, 'North America'),
  Text(1.0546117976794491, -0.3127202522947962, 'South America'),
  Text(1.0926986108246142, -0.12652962460213996, 'Africa'),
  Text(1.0996917916121562, -0.026037731484255974, 'Oceania')])
```

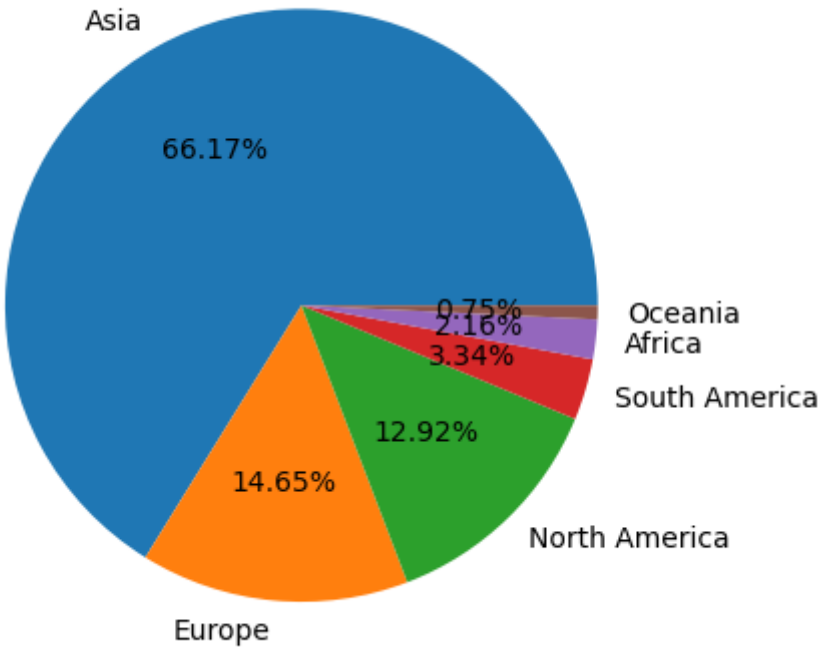


In [121]:
```
plt.pie(values,labels=keys,
        autopct="%0.2f%%")
```

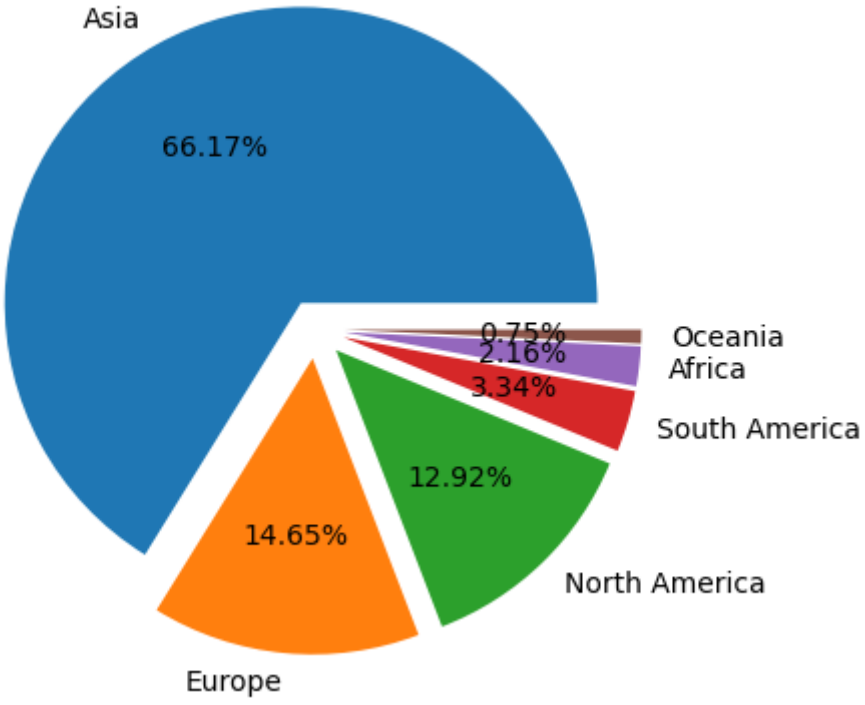
Out[121]:
```
([<matplotlib.patches.Wedge at 0x2c560454450>,
  <matplotlib.patches.Wedge at 0x2c560454e10>,
  <matplotlib.patches.Wedge at 0x2c560457890>,
  <matplotlib.patches.Wedge at 0x2c560449950>,
  <matplotlib.patches.Wedge at 0x2c56044b350>,
  <matplotlib.patches.Wedge at 0x2c560431310>],
 [Text(-0.5351743362316361, 0.9610350825224997, 'Asia'),
  Text(-0.10373513115748138, -1.0950977228374372, 'Europe'),
  Text(0.7670026411947619, -0.7884839557024984, 'North America'),
  Text(1.0546117976794491, -0.3127202522947962, 'South America'),
  Text(1.0926986108246142, -0.12652962460213996, 'Africa'),
  Text(1.0996917916121562, -0.026037731484255974, 'Oceania')],
 [Text(-0.2919132743081651, 0.5242009541031816, '66.17%'),
  Text(-0.05658279881317166, -0.597326030638602, '14.65%'),
  Text(0.4183650770153246, -0.4300821576559082, '12.92%'),
  Text(0.5752427987342449, -0.17057468306988885, '3.34%'),
  Text(0.5960174240861531, -0.0690161588738945, '2.16%'),
  Text(0.5998318863339034, -0.014202398991412348, '0.75%')])
```



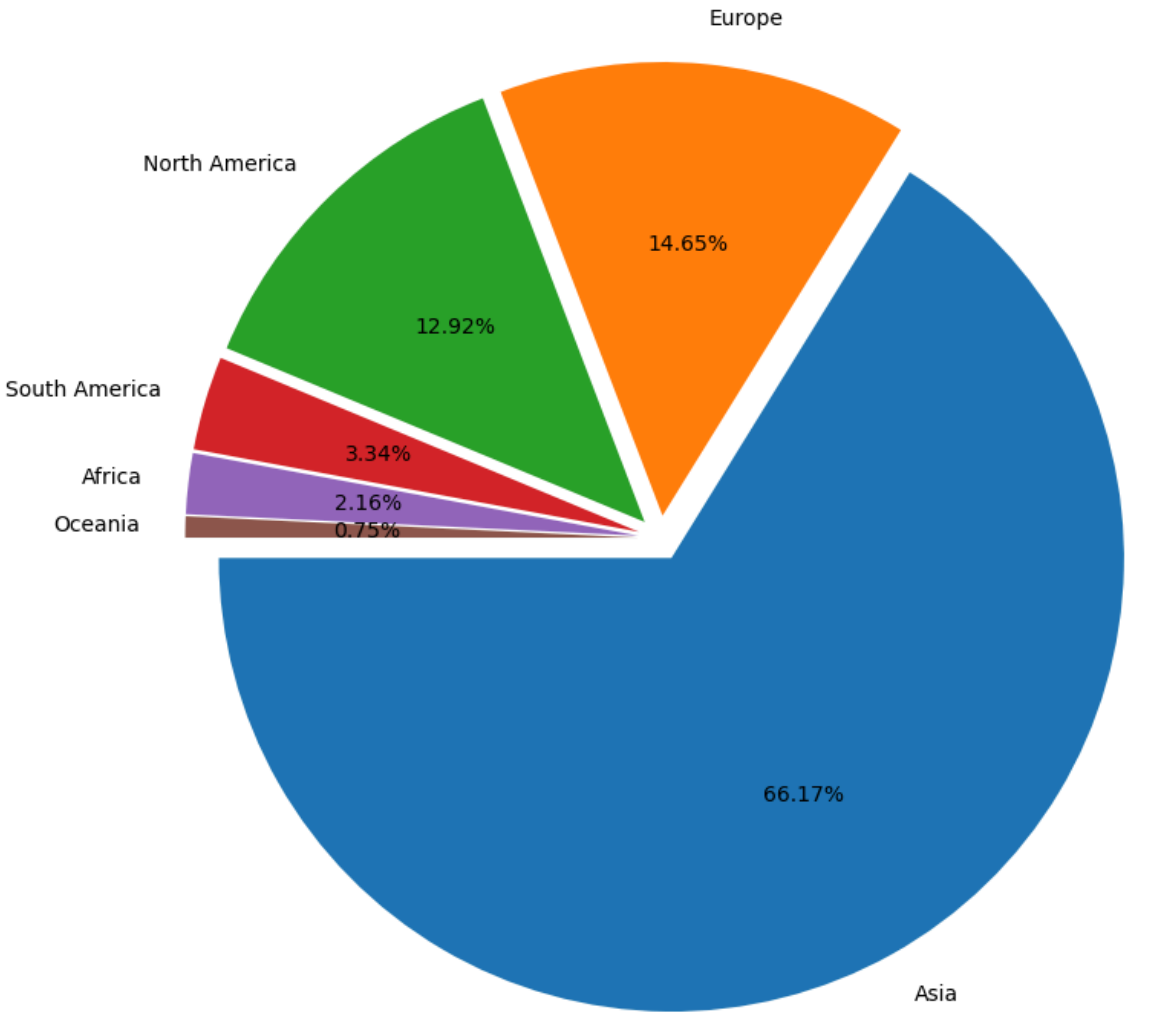
In [122]:
```
plt.pie(values,labels=keys,
        autopct="%0.2f%%")
plt.show()
```



In [123]:
```
plt.pie(values,labels=keys,
        autopct="%0.2f%%",
        explode=[0.1,0.1,0.1,0.1,0.1,0.1])
plt.show()
```



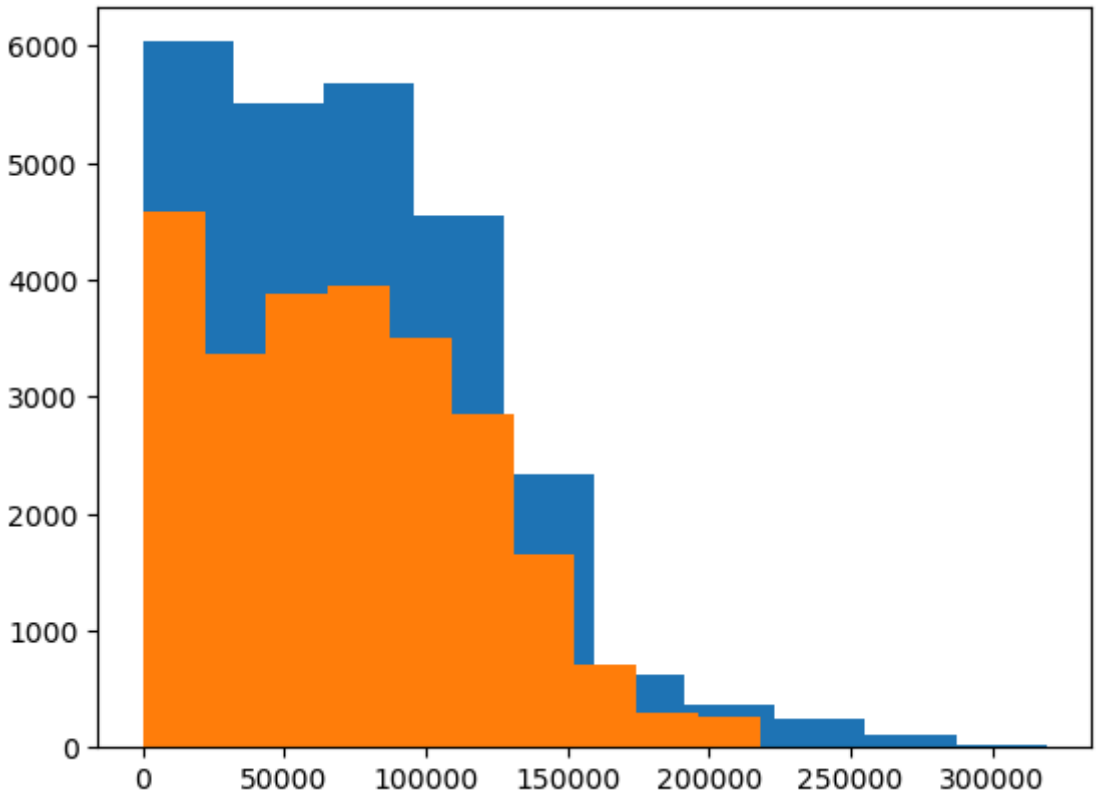
In [124]:
```
plt.pie(values,labels=keys,
        autopct="%0.2f%%",
        explode=[0.1,0.1,0.1,0.1,0.1,0.1],
        startangle=180,
        radius=2) #rotation
plt.show()
```



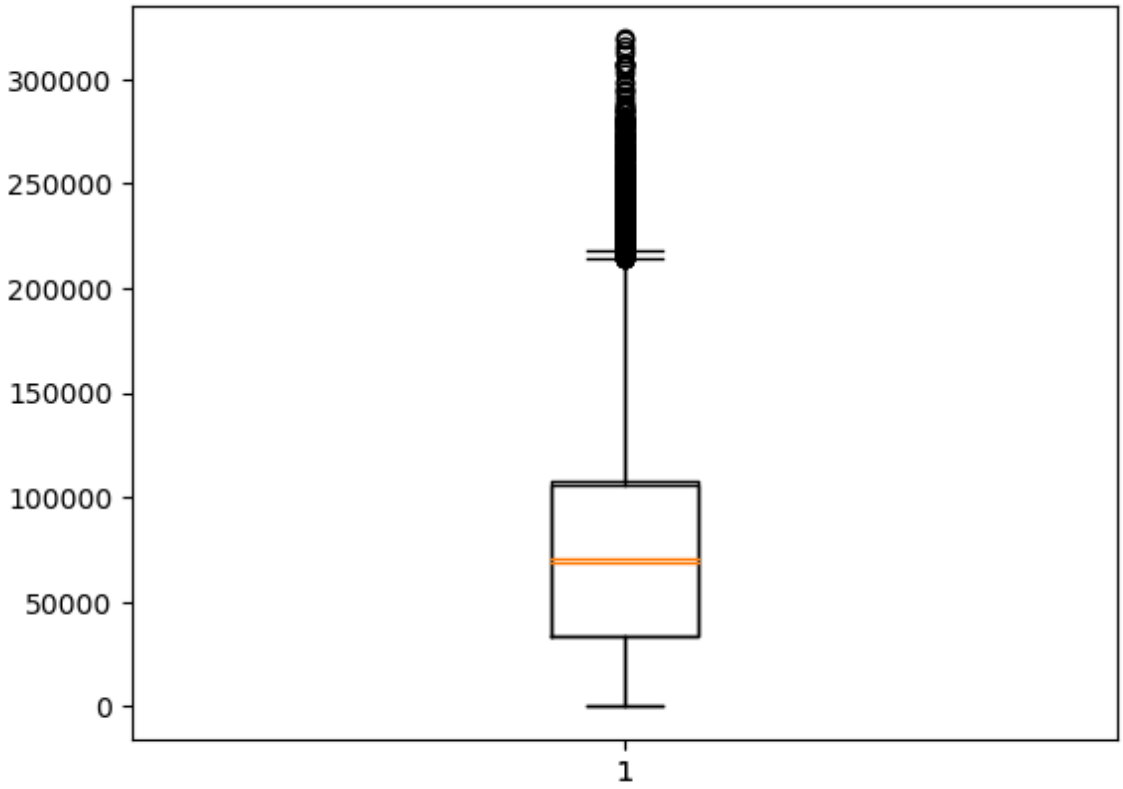
In [100]:
```
# compare original data with non outliers data

we plot histogram and box of the both
```

```
  Cell In[100], line 3
    we plot histogram and box of the both
       ^
SyntaxError: invalid syntax
```

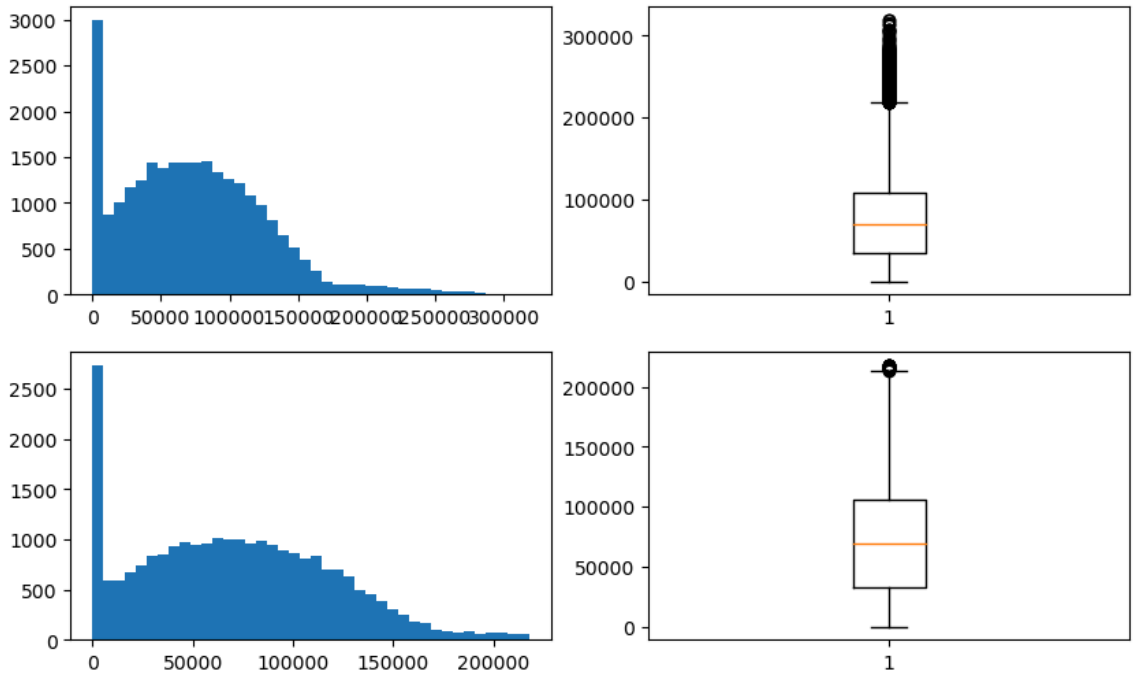
In [101]:
```
plt.hist(visa_df['prevailing_wage'])
plt.hist(non_outliers_df['prevailing_wage'])
plt.show()
```



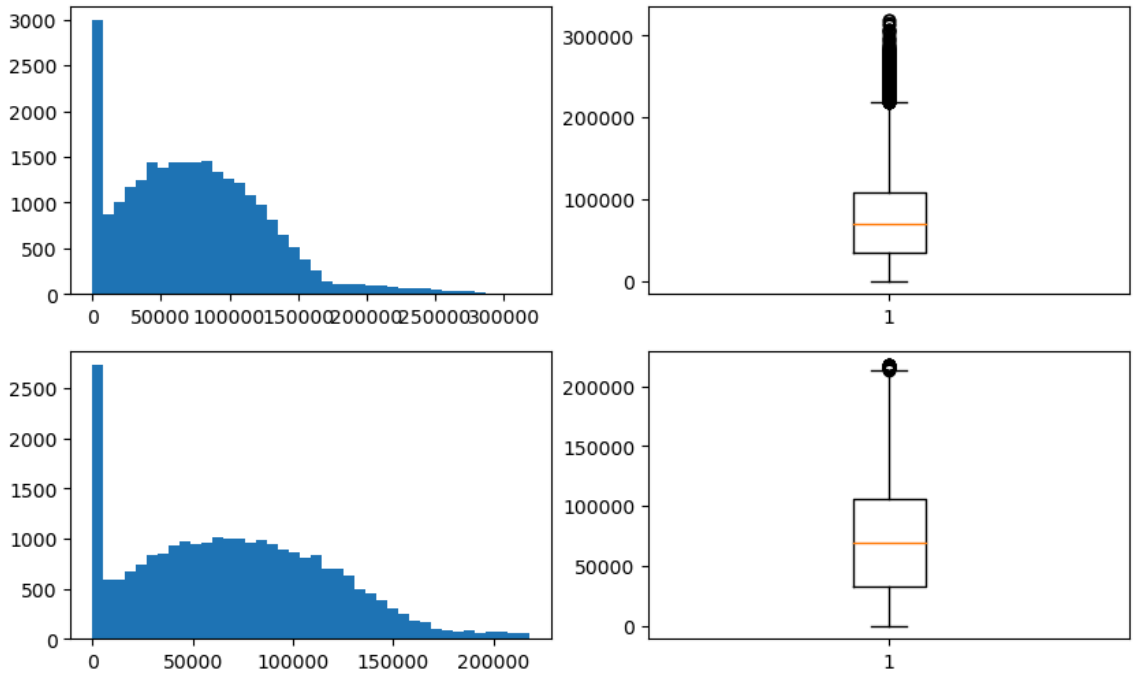
In [102]:
```
plt.boxplot(visa_df['prevailing_wage'])
plt.boxplot(non_outliers_df['prevailing_wage'])
plt.show()
```



In [103]:
```
plt.figure(figsize=(10,6))
plt.subplot(2,2,1)
plt.hist(visa_df['prevailing_wage'],bins=40)
plt.subplot(2,2,2)
plt.boxplot(visa_df['prevailing_wage'])
plt.subplot(2,2,3)
plt.hist(non_outliers_df['prevailing_wage'],bins=40)
plt.subplot(2,2,4)
plt.boxplot(non_outliers_df['prevailing_wage'])
plt.show()
```



In [104]:
```
plt.figure(figsize=(10,6))
plt.subplot(2,2,1).hist(visa_df['prevailing_wage'],bins=40)
plt.subplot(2,2,2).boxplot(visa_df['prevailing_wage'])
plt.subplot(2,2,3).hist(non_outliers_df['prevailing_wage'],bins=40)
plt.subplot(2,2,4).boxplot(non_outliers_df['prevailing_wage'])
plt.show()
```



In [ ]:
```
# how do deal the outliers

-drop the outliers

-we can drop the outliers if outliers percentage <2%
-but this is not recommended , we lost other columns data aslo

-inpute with Median values
- as we know that median does not afftected by outliers
-so it is good practise we can inpute outliers with median value

- cap with Q3 or Q1
-if outliers are present less than lower bound then fill with Q1
-if outliers are more than upper bound then fill with Q3
```

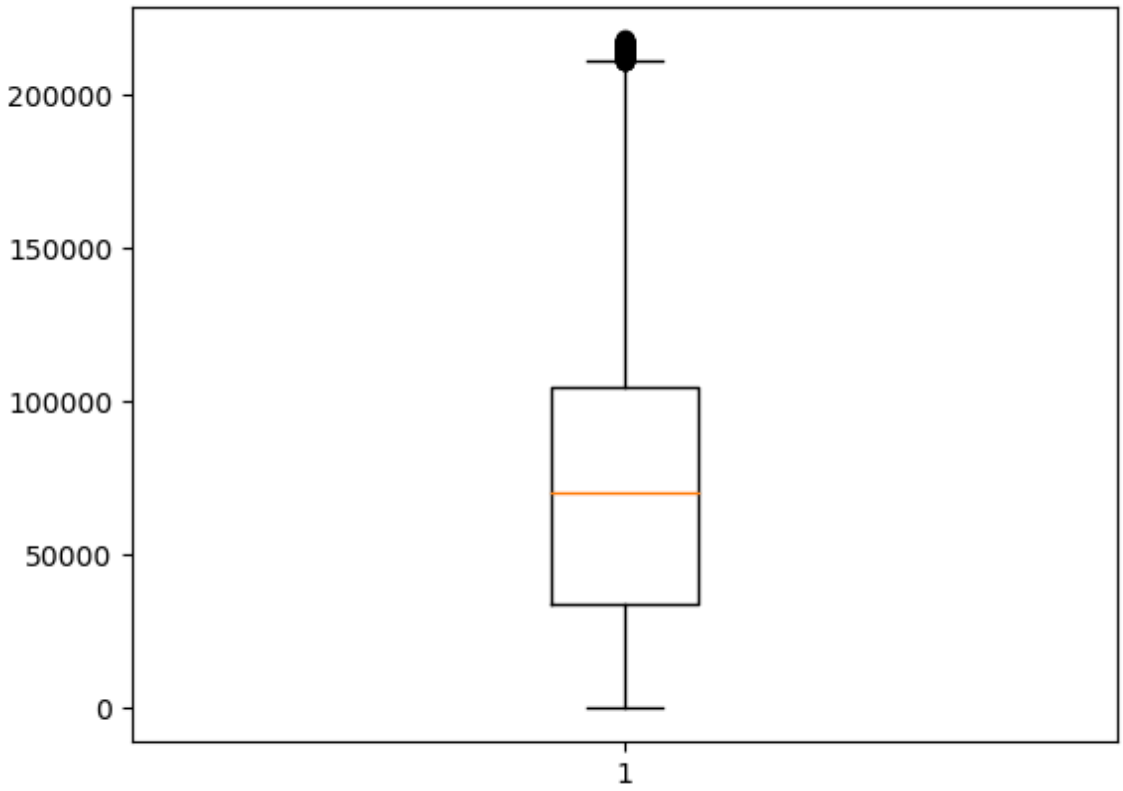
In [ ]:
```
#task read the each observation from prevailing -wage
# if the observation <lb or <ub: fill with median value
3
#else:keep as it is

# take empty list=[]
#median=visa_df['prevailing_wage'].median()
# for i in visa_df['prevailing_wage']:
#if i<lb or i>ub:
#   empty list.append(median)
#else:
#     emptylist.append(i)
```

In [105]:
```
Q1=np.percentile(visa_df['prevailing_wage'],25)
Q2=np.percentile(visa_df['prevailing_wage'],50)
Q3=np.percentile(visa_df['prevailing_wage'],75)
IQR=Q3-Q1
lb=Q1-1.5*IQR
ub=Q3+1.5*IQR

list=[]
median=visa_df['prevailing_wage'].median()
for i in visa_df['prevailing_wage']:
    if i<lb or i>ub:
        list.append(median)
    else:
         list.append(i)
len(list)
```

Out[105]:
```
25480
```

In [106]:
```
Q1=np.percentile(visa_df['prevailing_wage'],25)
Q2=np.percentile(visa_df['prevailing_wage'],50)
Q3=np.percentile(visa_df['prevailing_wage'],75)
IQR=Q3-Q1
lb=Q1-1.5*IQR
ub=Q3+1.5*IQR

list=[]
median=visa_df['prevailing_wage'].median()
for i in visa_df['prevailing_wage']:
    if i<lb or i>ub:
        list.append(median)
    else:
         list.append(i)
visa_df['prevailing_wage_new']=list
```

In [107]:
```
plt.boxplot(visa_df['prevailing_wage_new'])
plt.show()
```



In [108]:
```
# np.where;

-above replace one we use a traditional approach

-for loop, list ,if-else

-the same we get by using np.where method
```

```
  Cell In[108], line 3
    -above replace one we use a traditional approach
           ^
SyntaxError: invalid syntax
```

In [109]:
```
dict={'Name':['A','B','C','D'],
      'Num':[1,2,3,4]}
d=pd.DataFrame(dict)
d
```

Out[109]:

	Name	Num
0	A	1
1	B	2
2	C	3
3	D	4

In [110]:
```
# i want to replace with 100 num which has >2

# other wise keep same number

l=[]
for i in d['Num']:
    if i>2:
        l.append(100)
    else:
        l.append(i)

d['Num']=l
l
```

Out[110]:
```
[1, 2, 100, 100]
```

In [111]:
```
np.where(con,True,False)

-will take 3 arguments

-condition
con=d['Num']>2

-True value
t=100

-False value
f=d['Num']
```

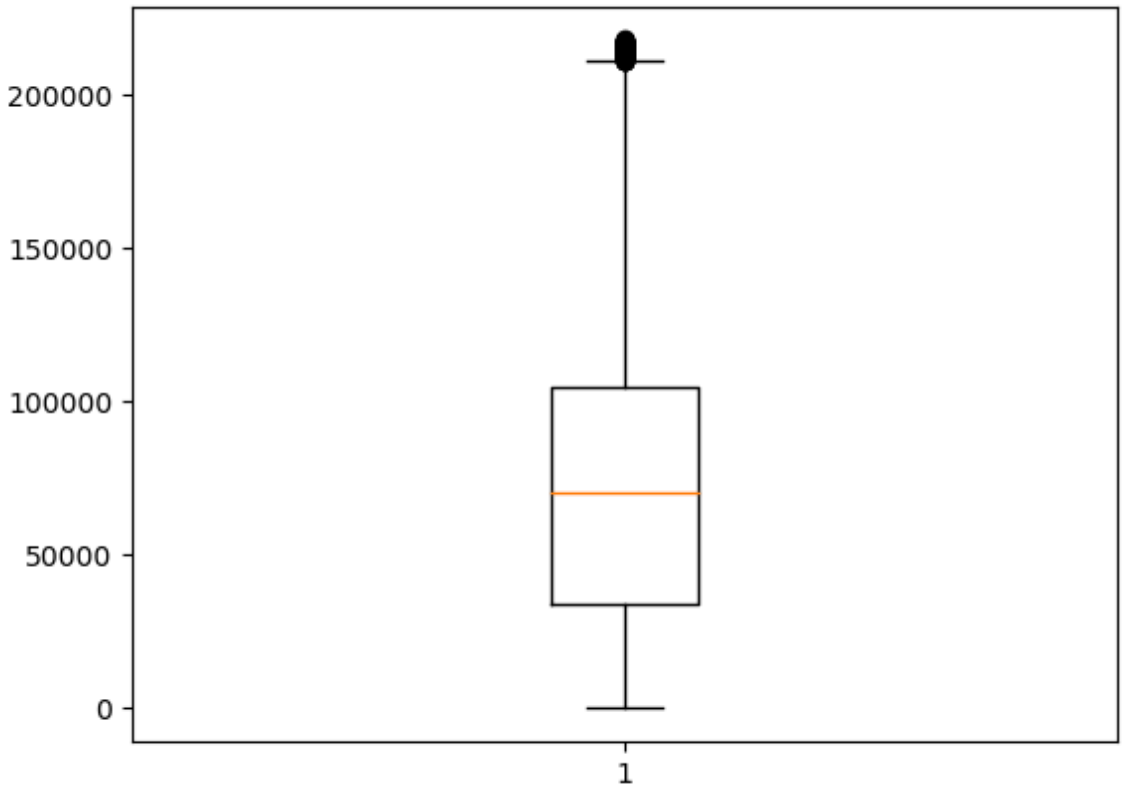
```
  Cell In[111], line 3
    -will take 3 arguments
          ^
SyntaxError: invalid syntax
```

In [112]:
```
l=np.where(d['Num']>2,100,d['Num'])

d['Num']=l
d
```

Out[112]:

	Name	Num
0	A	1
1	B	2
2	C	100
3	D	100

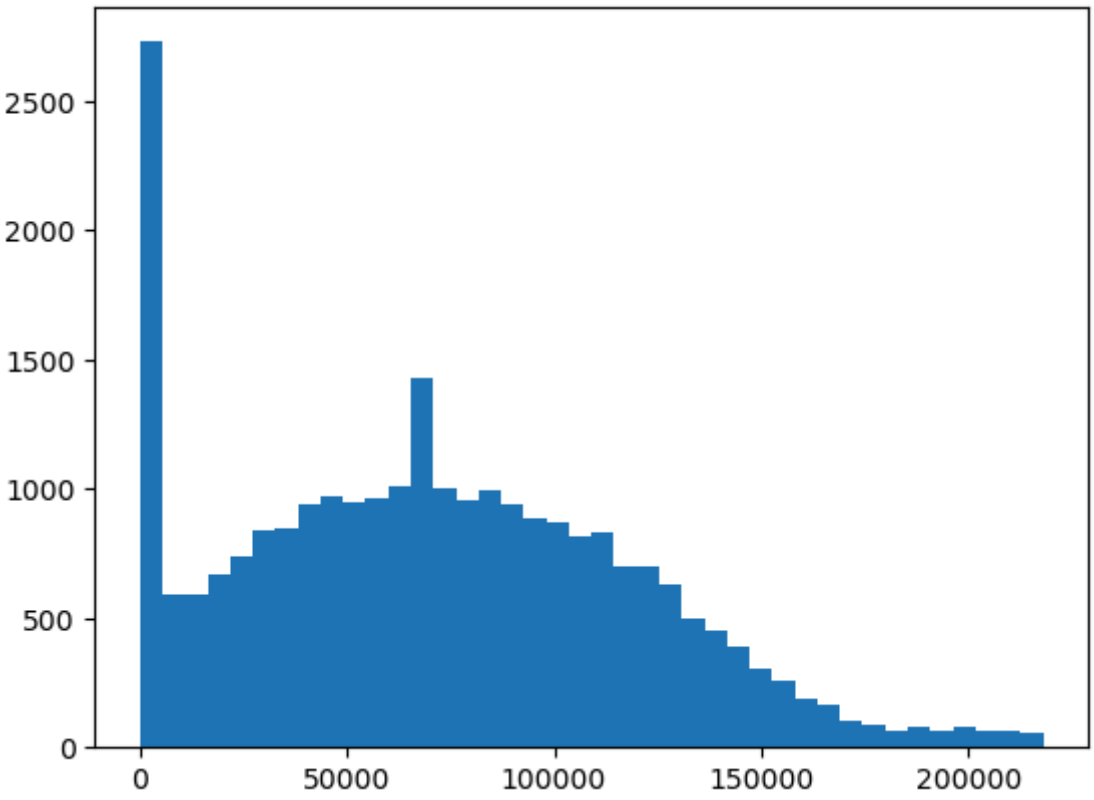
In [113]:
```
Q1=np.percentile(visa_df['prevailing_wage'],25)
Q2=np.percentile(visa_df['prevailing_wage'],50)
Q3=np.percentile(visa_df['prevailing_wage'],75)
IQR=Q3-Q1
lb=Q1-1.5*IQR
ub=Q3+1.5*IQR


median=visa_df['prevailing_wage'].median()
c1=visa_df['prevailing_wage']<lb
c2=visa_df['prevailing_wage']>ub
con=c1|c2
t=median
f=visa_df['prevailing_wage']
visa_df['prevailing_wage']=np.where(con,t,f)
plt.boxplot(visa_df['prevailing_wage'])
plt.show()
```



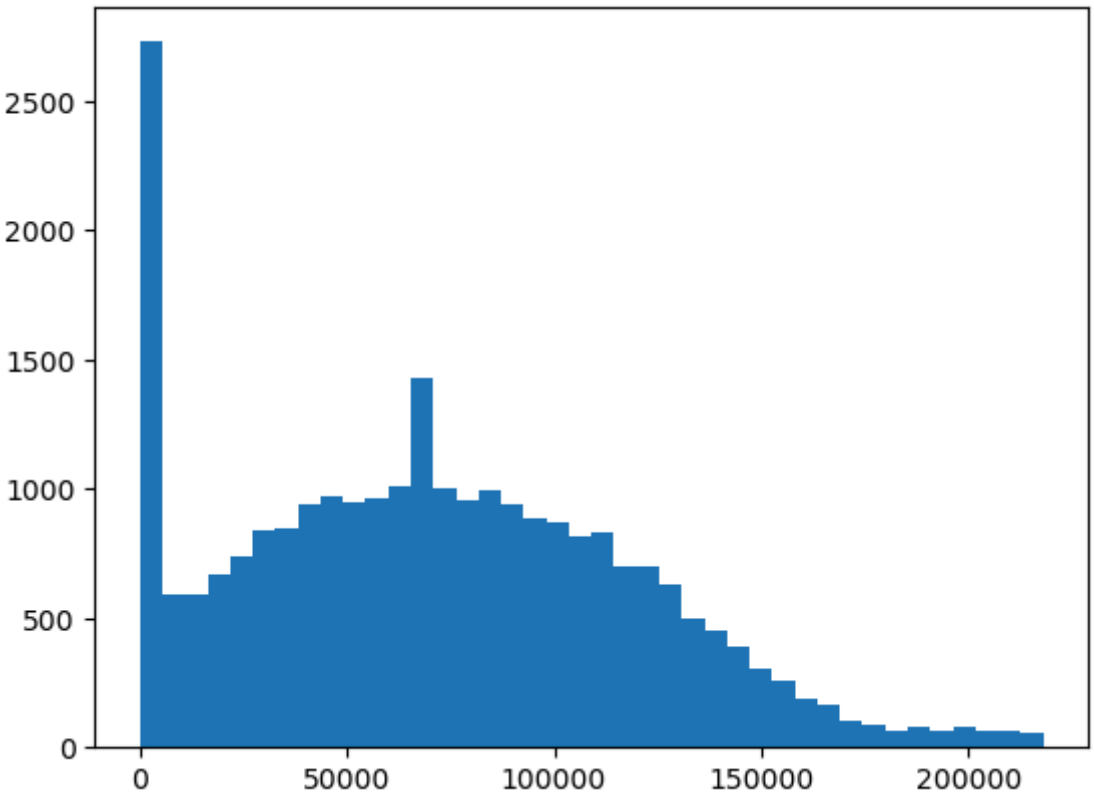
In [ ]:
```
#histogram:
```

In [125]:
```
plt.hist(visa_df['prevailing_wage'],bins=40)
```

Out[125]:
```
(array([2729.,  591.,  592.,  668.,  741.,  840.,  849.,  938.,  971.,
         950.,  962., 1007., 1425.,  999.,  954.,  991.,  941.,  885.,
         868.,  813.,  832.,  702.,  697.,  629.,  497.,  449.,  391.,
         306.,  255.,  187.,  163.,  101.,   85.,   68.,   79.,   63.,
          76.,   68.,   63.,   55.]),
 array([2.13670000e+00, 5.45444853e+03, 1.09067604e+04, 1.63590722e+04,
        2.18113840e+04, 2.72636959e+04, 3.27160077e+04, 3.81683195e+04,
        4.36206314e+04, 4.90729432e+04, 5.45252550e+04, 5.99775669e+04,
        6.54298787e+04, 7.08821905e+04, 7.63345024e+04, 8.17868142e+04,
        8.72391260e+04, 9.26914379e+04, 9.81437497e+04, 1.03596062e+05,
        1.09048373e+05, 1.14500685e+05, 1.19952997e+05, 1.25405309e+05,
        1.30857621e+05, 1.36309933e+05, 1.41762244e+05, 1.47214556e+05,
        1.52666868e+05, 1.58119180e+05, 1.63571492e+05, 1.69023804e+05,
        1.74476115e+05, 1.79928427e+05, 1.85380739e+05, 1.90833051e+05,
        1.96285363e+05, 2.01737675e+05, 2.07189986e+05, 2.12642298e+05,
        2.18094610e+05]),
 <BarContainer object of 40 artists>)
```



In [126]:
```
f,i,n=plt.hist(visa_df['prevailing_wage'],bins=40)
```



In [127]:
```
len(f),len(i),len(n)
```

Out[127]:
```
(40, 41, 40)
```

In [128]:
```
f
```

Out[128]:
```
array([2729.,  591.,  592.,  668.,  741.,  840.,  849.,  938.,  971.,
        950.,  962., 1007., 1425.,  999.,  954.,  991.,  941.,  885.,
        868.,  813.,  832.,  702.,  697.,  629.,  497.,  449.,  391.,
        306.,  255.,  187.,  163.,  101.,   85.,   68.,   79.,   63.,
         76.,   68.,   63.,   55.])
```

In [129]:
```
i
```

Out[129]:
```
array([2.13670000e+00, 5.45444853e+03, 1.09067604e+04, 1.63590722e+04,
       2.18113840e+04, 2.72636959e+04, 3.27160077e+04, 3.81683195e+04,
       4.36206314e+04, 4.90729432e+04, 5.45252550e+04, 5.99775669e+04,
       6.54298787e+04, 7.08821905e+04, 7.63345024e+04, 8.17868142e+04,
       8.72391260e+04, 9.26914379e+04, 9.81437497e+04, 1.03596062e+05,
       1.09048373e+05, 1.14500685e+05, 1.19952997e+05, 1.25405309e+05,
       1.30857621e+05, 1.36309933e+05, 1.41762244e+05, 1.47214556e+05,
       1.52666868e+05, 1.58119180e+05, 1.63571492e+05, 1.69023804e+05,
       1.74476115e+05, 1.79928427e+05, 1.85380739e+05, 1.90833051e+05,
       1.96285363e+05, 2.01737675e+05, 2.07189986e+05, 2.12642298e+05,
       2.18094610e+05])
```

In [130]:
```
between 2.13670000e+00, 7.98234003e+03
we have 2992 observation

between 7.98234003e+03, 1.59625434e+04,
we have 871 observation
```

```
  Cell In[130], line 1
    between 2.13670000e+00, 7.98234003e+03
            ^
SyntaxError: invalid syntax
```

In [131]:
```
l=2.13670000e+00
u=7.98234003e+03

c1=visa_df['prevailing_wage']>=l
c2=visa_df['prevailing_wage']<u
c=c1&c2
len(visa_df[c])
```

Out[131]: 2992

In [132]:
```
def frequency(l,u):
    c1=visa_df['prevailing_wage']>=l
    c2=visa_df['prevailing_wage']<u
    c=c1&c2
    print(len(visa_df[c]))
frequency(7.98234003e+03, 1.59625434e+04,)
```

```
871
```

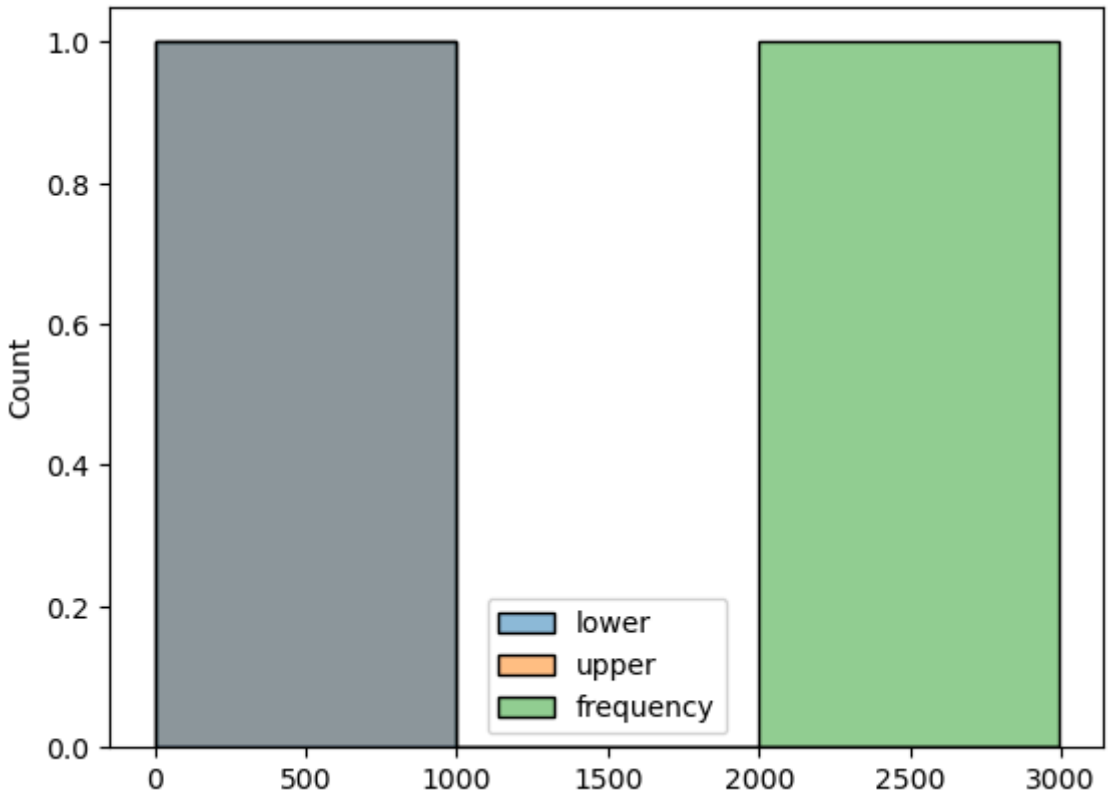
In [133]:
```
# task_1

#create a dataframe

#lower  upper frequency
#2.13   7.98  2992

l=[2.13]
u=[7.98]
f=[2992]
cols=['lower','upper','frequency']
task_1=pd.DataFrame(zip(l,u,f),columns=cols)
task_1
```

Out[133]:

	lower	upper	frequency
0	2.13	7.98	2992

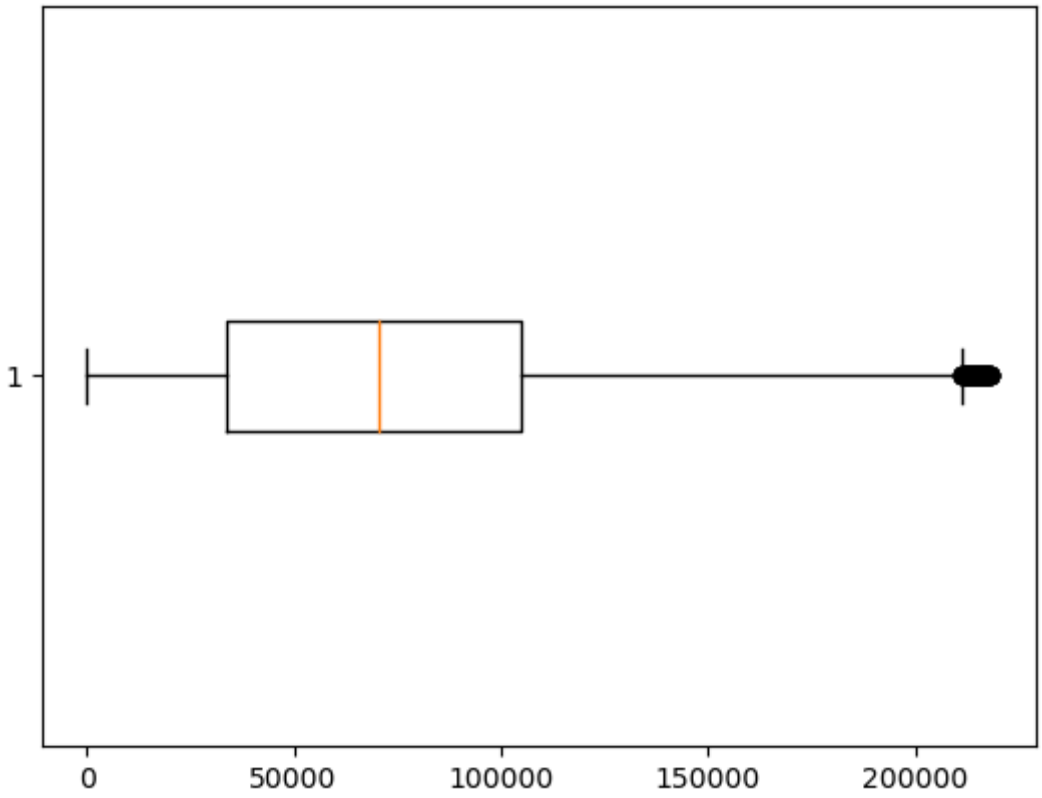
In [134]:
```
# task-2:
#in seaborn how to plot histogram
sns.histplot(task_1)
```

Out[134]:
```
<Axes: ylabel='Count'>
```



In [ ]:
```
# Box plot:

- boxplot is used to identify outliers
-in box plot we have

-Q1:25p value
-Q2:50p value
-Q3:75p value

-IQR:Q3-Q1
-mild outliers Q1-.5*IQR and Q3+1.5IQR
-huge outliers Q1-3IQR And Q3+3IQR


Q1-1.5IQR   Q1   median  Q3   Q3+1.5IQR
                       |-----:-----|
       o      |--------|     :     |--------|    o  o
                       |-----:-----|
     flier             <----------->           fliers
                            IQR
```

In [136]:
```
plt.boxplot(visa_df['prevailing_wage'],vert=False)
plt.show()

# black dots are outliers
# orange line is median
```



In [ ]:
```
# categorical vs categorical
```

In [ ]:
```
# continent
# case_status
# as we know that there are  25480 observation are there
# in that 16k are from asian applicants
# how many visa approved and how amny visa rejected
```

In [137]:
```
c1=visa_df['continent']=='Asia'
c2=visa_df['case_status']=='Certified'
c3=visa_df['case_status']=='Denied'

cert_con=c1&c2
den_con=c1&c3

certified_count=len(visa_df[cert_con])
denied_count=len(visa_df[den_con])
print(f"there are {certified_count} got certified from Asia")
print(f"there are {denied_count} got denied from Asia")
```

```
there are 11012 got certified from Asia
there are 5849 got denied from Asia
```

In [138]:
```
unique_labels=visa_df['continent'].unique()
certified_count=[]
denied_count=[]
for i in unique_labels:
    c1=visa_df['continent']==i
    c2=visa_df['case_status']=='Certified'
    c3=visa_df['case_status']=='Denied'

    cert_con=c1&c2
    den_con=c1&c3

    certified_count.append(len(visa_df[cert_con]))
    denied_count.append(len(visa_df[den_con]))
cols=['Continent','Certified','Denied']
d1=pd.DataFrame(zip(unique_labels,
                    certified_count,
                    denied_count),columns=cols)
d1
d1.set_index('Continent')
```

Out[138]:

Continent	Certified	Denied
Asia	11012	5849
Africa	397	154
North America	2037	1255
Europe	2957	775
South America	493	359
Oceania	122	70

In [139]:
```
d1.set_index('Continent')
```

Out[139]:

Continent	Certified	Denied
Asia	11012	5849
Africa	397	154
North America	2037	1255
Europe	2957	775
South America	493	359
Oceania	122	70

In [140]:
```
col1=[visa_df['case_status'],visa_df['education_of_employee']]
col2=visa_df['continent']
result1=pd.crosstab(col1,col2)
result1
```

Out[140]:

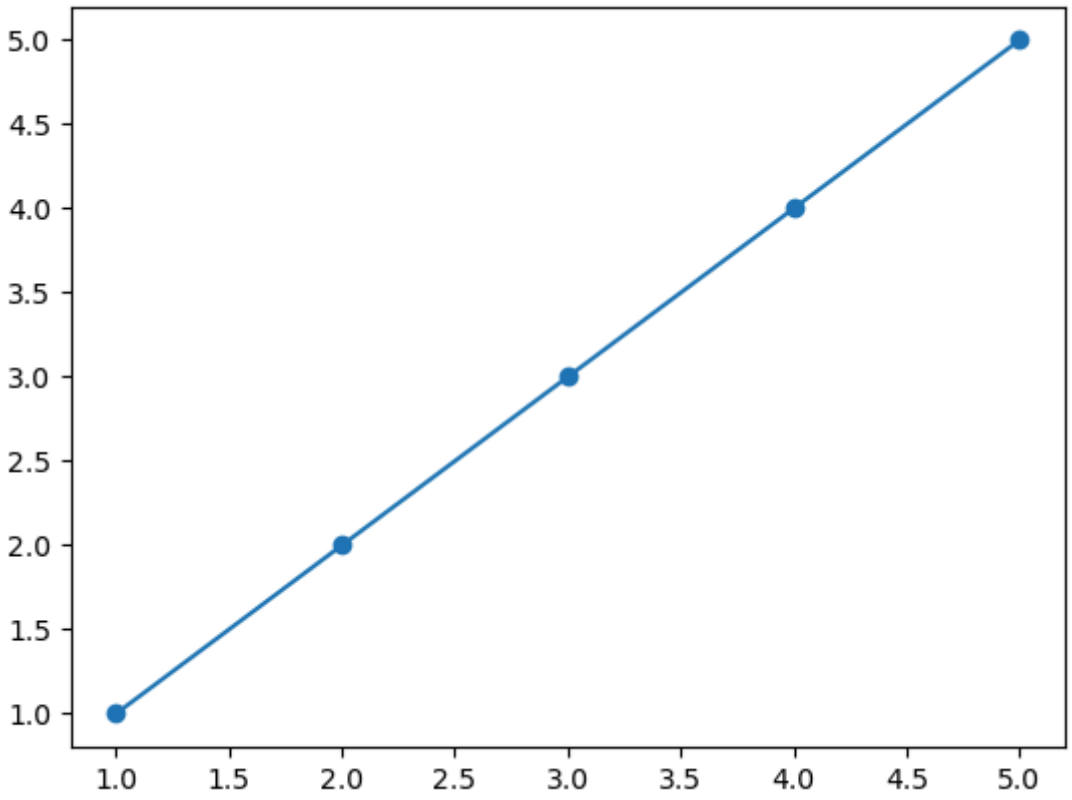
	continent	Africa	Asia	Europe	North America	Oceania	South America
case_status	education_of_employee						
Certified	Bachelor's	81	4407	1040	641	38	160
	Doctorate	43	780	788	207	19	75
	High School	23	676	162	210	19	74
	Master's	250	5149	967	979	46	184
Denied	Bachelor's	62	2761	259	584	28	173
	Doctorate	11	143	58	51	3	14
	High School	43	1614	328	191	17	63
	Master's	38	1331	130	429	22	109

In [ ]:
```
# numerical vs numerical
```

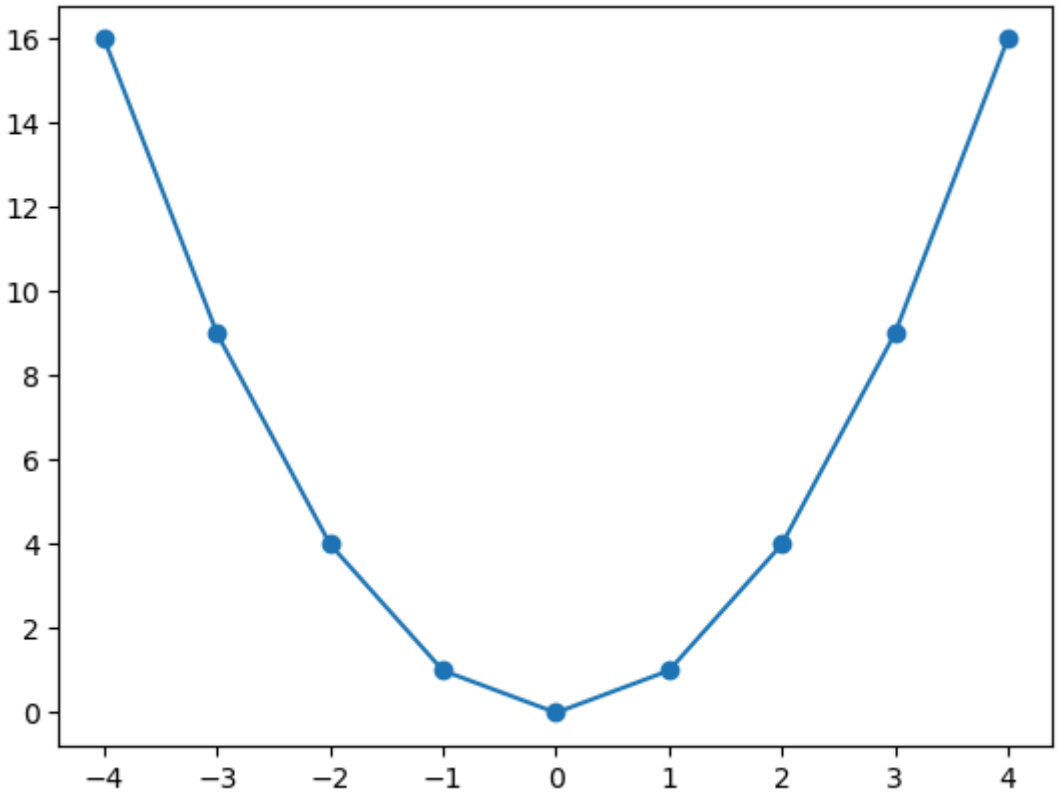
In [141]:
```
x=[1,2,3,4,5]
y=[1,2,3,4,5]
#(1,1),(2,2),(3,3),(4,4),(5,5)
```

In [ ]:
```
#plt.scatter
```

In [142]:
```
plt.scatter(x,y)
plt.plot(x,y)
plt.show()
```



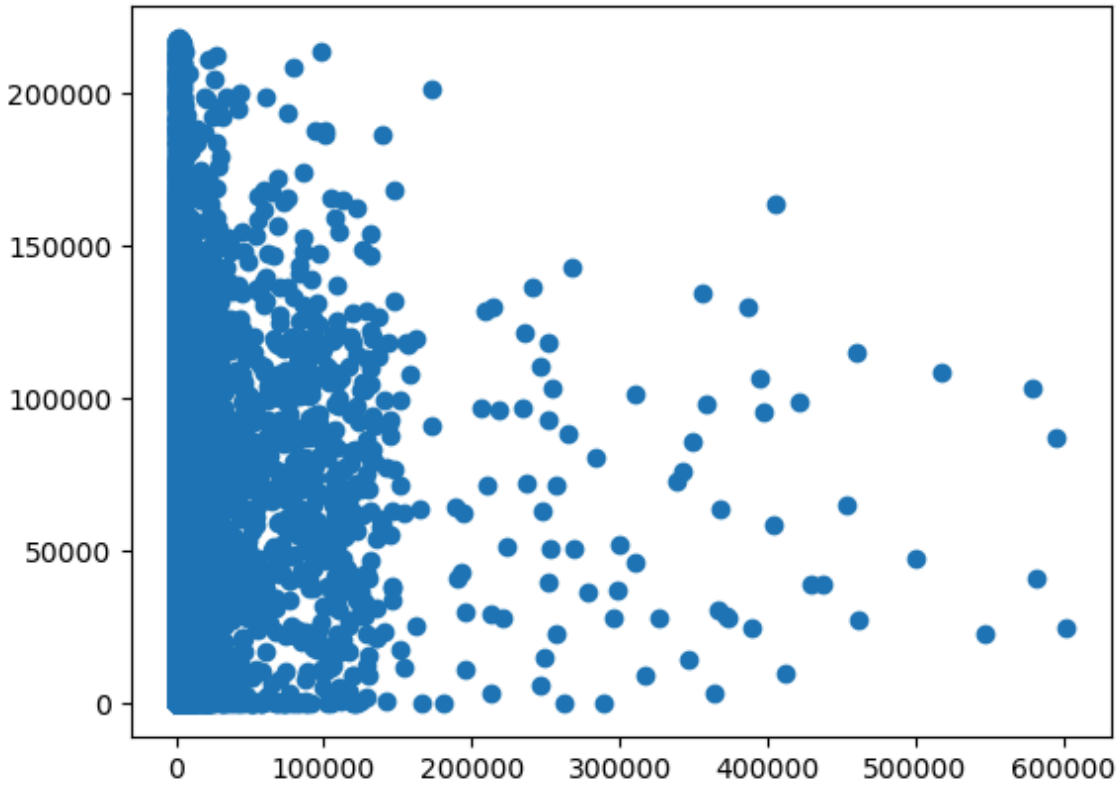
In [143]:
```
x=[i for i in range(-4,5)]
y=[i*i for i in x]
plt.scatter(x,y)
plt.plot(x,y)
plt.show()
```



In [144]:
```
# extracte only numerical columns

num_cols=visa_df.select_dtypes(exclude='object')
num_cols.columns
```

Out[144]:
```
Index(['no_of_employees', 'yr_of_estab', 'prevailing_wage',
       'prevailing_wage_new'],
      dtype='object')
```

In [145]:
```
col1=visa_df['no_of_employees']
col2=visa_df['prevailing_wage']
plt.scatter(col1,col2)
plt.show()  # no relation
```



In [ ]:
```
# co-relation pearson correlation coffecient

- r varies from -1 to 1
- -1 to 0 :negative relation
- o to 1:positive relation
- 0: no relation


- by formula it gives
-when you do this python
- it gives the matrix
-in visa data we have 3 numerical column are there
-python wii give a matrix w..r.t 3 numerical column
- the values in each field tells about then relation between the varibvles
```

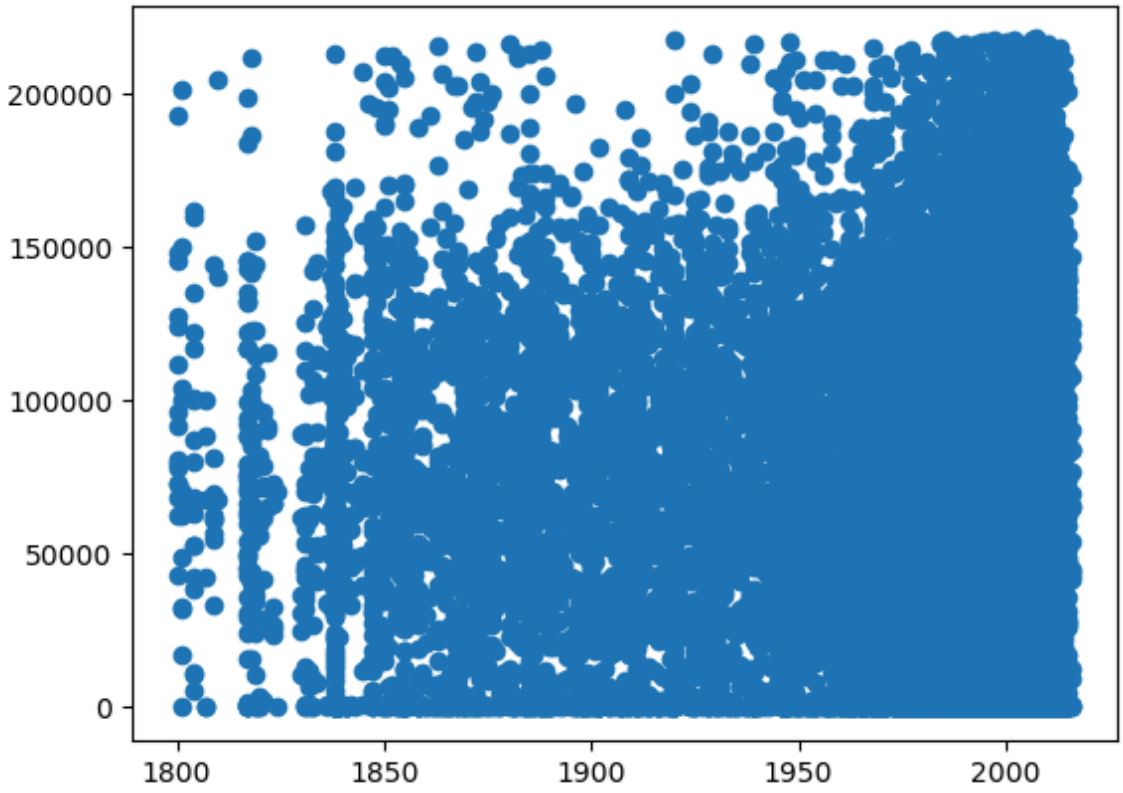
In [146]:
```
visa_df.corr(numeric_only=True)
```

Out[146]:

	no_of_employees	yr_of_estab	prevailing_wage	prevailing_wage_new
no_of_employees	1.000000	-0.017770	-0.006762	-0.006762
yr_of_estab	-0.017770	1.000000	0.015885	0.015885
prevailing_wage	-0.006762	0.015885	1.000000	1.000000
prevailing_wage_new	-0.006762	0.015885	1.000000	1.000000

In [ ]:
```
# check the scatter plot between yr-of_estab
# with prevailing_wage
# we are scattering the relation  is 0.012342
```

In [147]:
```
col1=visa_df['yr_of_estab']
col2=visa_df['prevailing_wage']
plt.scatter(col1,col2)
plt.show()
```



In [148]:
```
path=r"C:\Users\kasho\Downloads\winequality_red - winequality_red.csv"
wine=pd.read_csv(path)
wine
```

Out[148]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
3193	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75
3194	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3195	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71
3196	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3197	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66

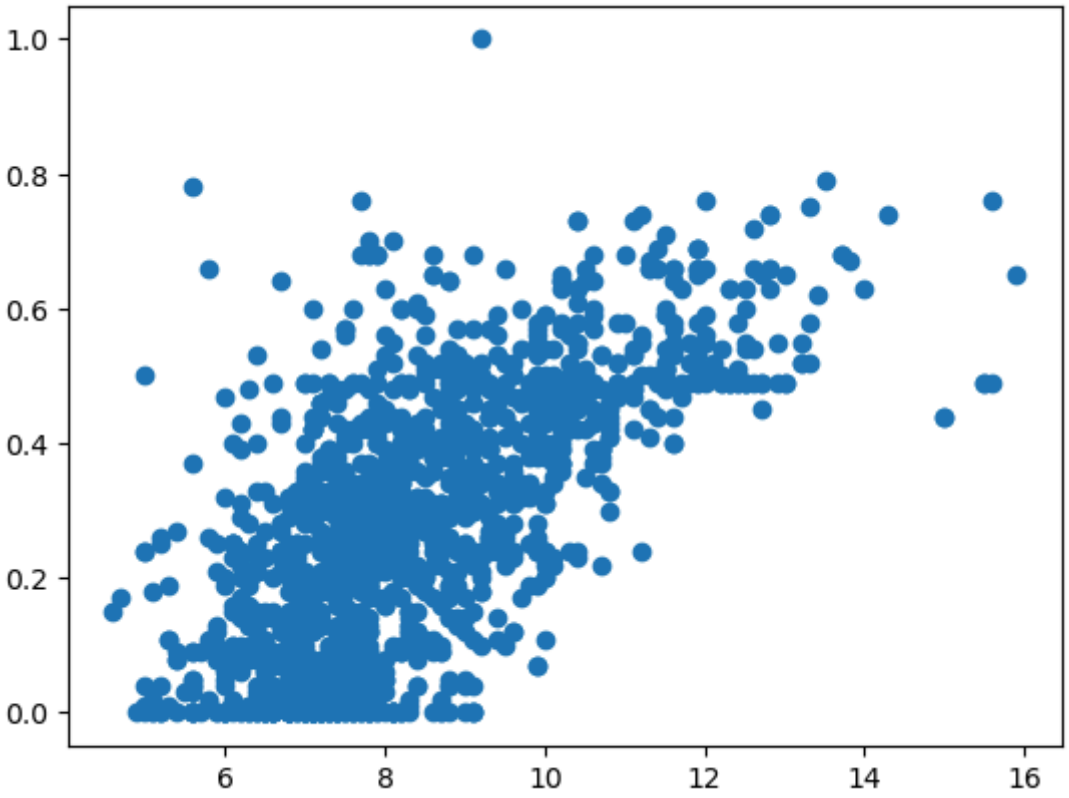
3198 rows × 12 columns

In [149]:
```
wine.columns
```

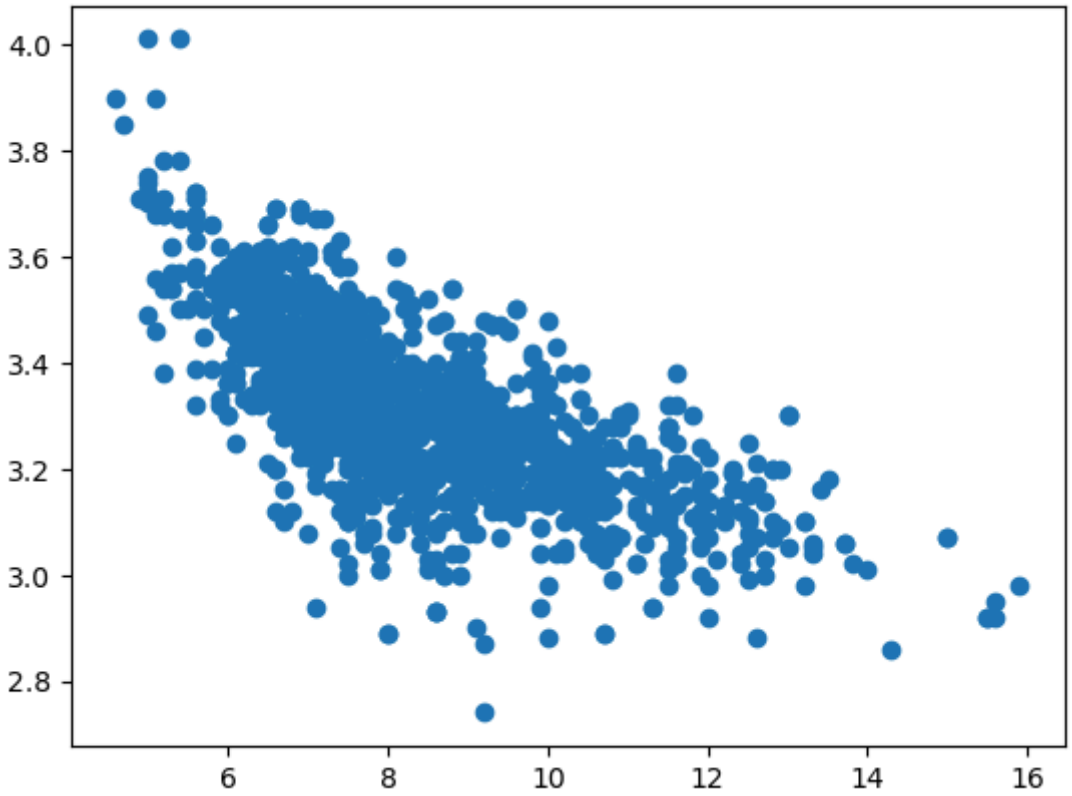
Out[149]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual suga
r',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'densit
y',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [ ]:
```
# fixed acidity and citric acid :o.67+ve
```

In [150]:
```
col1=wine['fixed acidity']
col2=wine['citric acid']
plt.scatter(col1,col2)
plt.show()
```



In [151]:
```
# fixed acidity and citric acid :o.67-ve
col1=wine['fixed acidity']
col2=wine['pH']
plt.scatter(col1,col2)
plt.show()
```



In [ ]:
```
# heat map

- heat map is useful to visulization of matrix

- it is under seaborn packages

- heat map will varies the values and gives the colors about the values
```

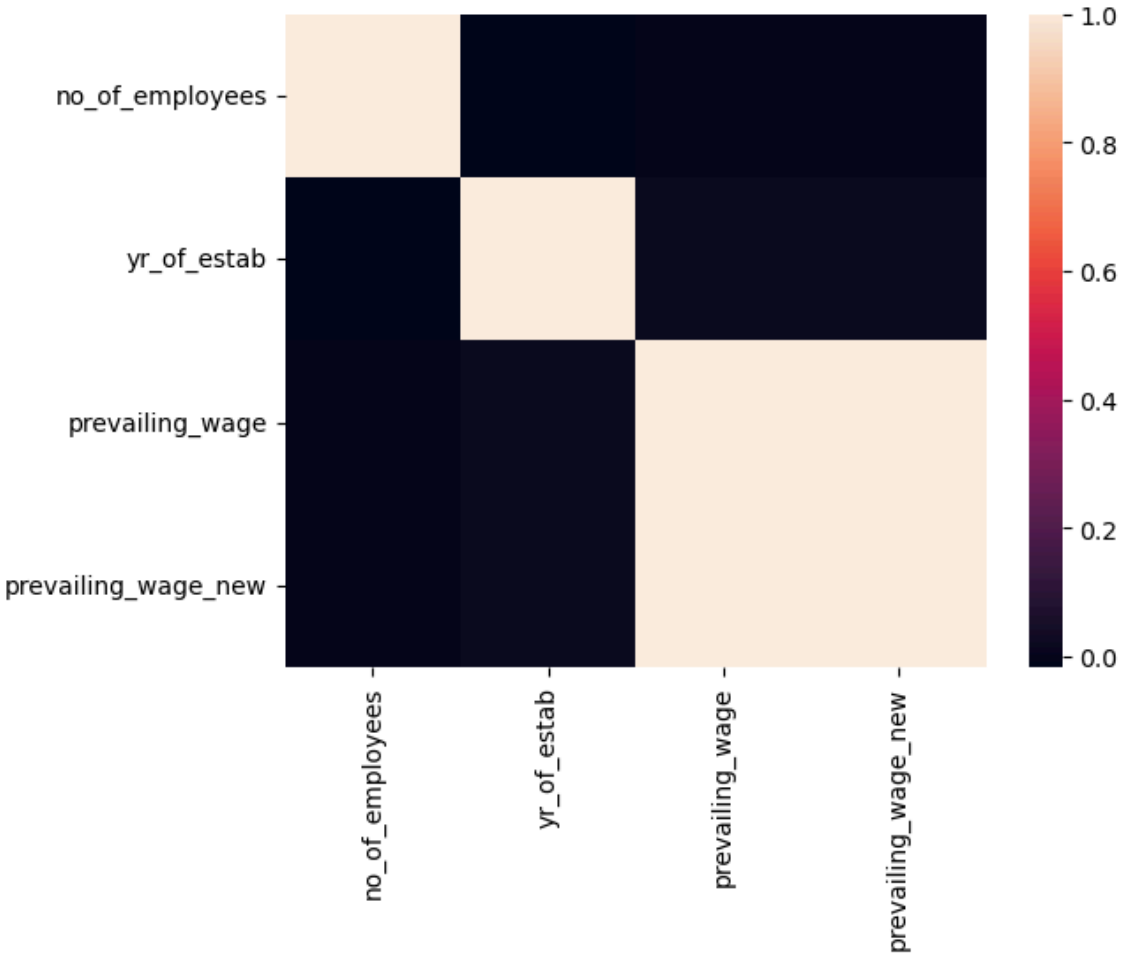
In [152]:
```
corr_wine=visa_df.corr(numeric_only=True)
corr_wine
```

Out[152]:

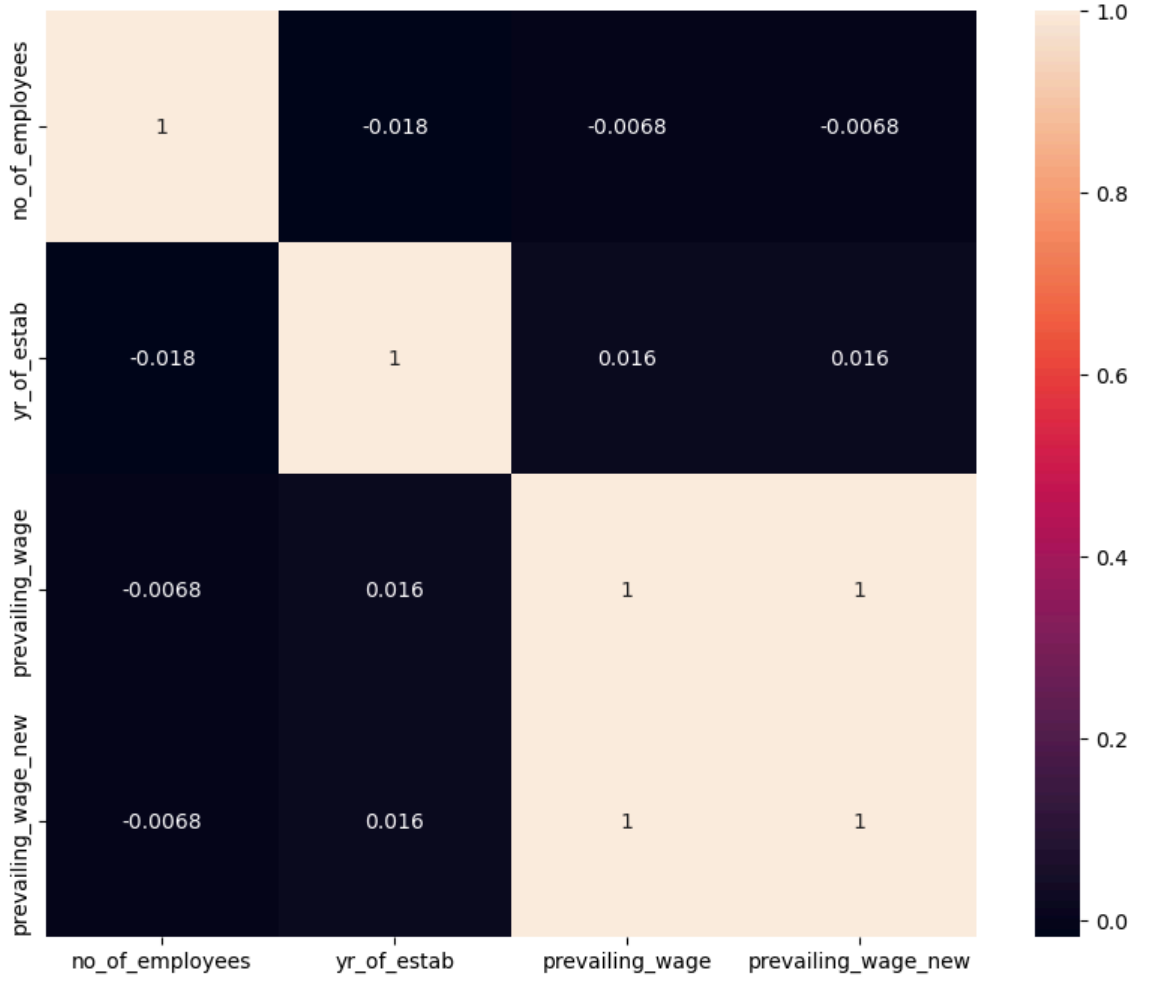
	no_of_employees	yr_of_estab	prevailing_wage	prevailing_wage_new
no_of_employees	1.000000	-0.017770	-0.006762	-0.006762
yr_of_estab	-0.017770	1.000000	0.015885	0.015885
prevailing_wage	-0.006762	0.015885	1.000000	1.000000
prevailing_wage_new	-0.006762	0.015885	1.000000	1.000000

In [153]:
```
sns.heatmap(corr_wine)
```

Out[153]: <Axes: >



In [154]:
```
plt.figure(figsize=(10,8))
sns.heatmap(corr_wine,
            annot=True)
plt.show()
```



In [ ]:
```
- in machine learing it is very important to convert categorical data to nu

- machine learing models develop by maths

-machine learing takes the input in the form of numbers only

-to convert that we have some encoding techniques

- Label Encoder
 -map
 -np.where
 -using sklearn package

- One hot encoder
 - using pandas package:pd.get_dummies
```

In [ ]:
```
# map method

-before applying map method first get the unique labels of the column
- for example case_status is a categorical column
- it has two unique labels are there
  -denied
 -certified


-create a dictionary key as labels, value as number
- d={'certified':0,'denied':1}
- this dictionary we need to map the case_status column
```

In [155]:
```
visa_df['case_status'].unique()
```

Out[155]:
```
array(['Denied', 'Certified'], dtype=object)
```

In [156]:
```
d={'Certified':0,'Denied':1}
visa_df['case_status']=visa_df['case_status'].map(d)
```

In [157]:
```
visa_df
```

Out[157]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 13 columns

In [158]:
```
labels=visa_df['continent'].unique()
```

In [159]:
```
d={}
labels=visa_df['continent'].unique()
for i in range(len(labels)):
    d[labels[i]]=i
visa_df['continent']=visa_df['continent'].map(d)
visa_df
```

Out[159]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	0	High School	N	
1	EZYV02	0	Master's	Y	
2	EZYV03	0	Bachelor's	N	
3	EZYV04	0	Bachelor's	N	
4	EZYV05	1	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	0	Bachelor's	Y	
25476	EZYV25477	0	High School	Y	
25477	EZYV25478	0	Master's	Y	
25478	EZYV25479	0	Master's	Y	
25479	EZYV25480	0	Bachelor's	Y	

25480 rows × 13 columns

In [160]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)
cat_cols=visa_df.select_dtypes(include='object').columns
d={}
for j in cat_cols[1:]:

    labels=visa_df[j].unique()  #j=column
    for i in range(len(labels)):#i=number
        d[labels[i]]=i
visa_df[j]=visa_df[j].map(d)
visa_df
```

Out[160]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [161]:
```
cat_cols=visa_df.select_dtypes(include='object').columns
cat_cols[1:]
```

Out[161]:
```
Index(['continent', 'education_of_employee', 'has_job_experience',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position'],
      dtype='object')
```

In [ ]:
```
# labelencoder

-labelencoder is package avaible in sklearn

- sickit learn heart of ml

- read the package

- save the package

- apply fit transform
```

In [162]:
```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
visa_df['case_status']=le.fit_transform(visa_df['case_status'])
visa_df
```

Out[162]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [163]:
```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
visa_df['continent']=le.fit_transform(visa_df['continent'])
visa_df
```

Out[163]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	1	High School	N	
1	EZYV02	1	Master's	Y	
2	EZYV03	1	Bachelor's	N	
3	EZYV04	1	Bachelor's	N	
4	EZYV05	0	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	1	Bachelor's	Y	
25476	EZYV25477	1	High School	Y	
25477	EZYV25478	1	Master's	Y	
25478	EZYV25479	1	Master's	Y	
25479	EZYV25480	1	Bachelor's	Y	

25480 rows × 12 columns

In [167]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)
cat_cols=visa_df.select_dtypes(include='object').columns
cat_cols

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
visa_df['i']=le.fit_transform(visa_df['i'])
visa_df
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3653, in In
dex.get_loc(self, key)
   3652 try:
-> 3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:

File ~\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:147, in pandas._
libs.index.IndexEngine.get_loc()

File ~\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:176, in pandas._
libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtab
le.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.hashtab
le.PyObjectHashTable.get_item()

KeyError: 'i'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
t)
Cell In[167], line 8
      6 from sklearn.preprocessing import LabelEncoder
      7 le=LabelEncoder()
----> 8 visa_df['i']=le.fit_transform(visa_df['i'])
      9 visa_df

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:3761, in DataFram
e.__getitem__(self, key)
   3759 if self.columns.nlevels > 1:
   3760     return self._getitem_multilevel(key)
-> 3761 indexer = self.columns.get_loc(key)
   3762 if is_integer(indexer):
   3763     indexer = [indexer]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3655, in In
dex.get_loc(self, key)
   3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:
-> 3655     raise KeyError(key) from err
   3656 except TypeError:
   3657     # If we have a listlike key, _check_indexing_error will raise
   3658     #  InvalidIndexError. Otherwise we fall through and re-raise
   3659     #  the TypeError.
   3660     self._check_indexing_error(key)

KeyError: 'i'
```

In [168]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)
cat_cols=visa_df.select_dtypes(include='object').columns
cat_cols
```

Out[168]:
```
Index(['case_id', 'continent', 'education_of_employee', 'has_job_experienc
e',
       'requires_job_training', 'region_of_employment', 'unit_of_wage',
       'full_time_position', 'case_status'],
      dtype='object')
```

In [169]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in  cat_cols:
    visa_df[i]=le.fit_transform(visa_df[i])
```

In [170]:
```
visa_df
```

Out[170]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training
0	0	1	2	0	0
1	1	1	3	1	0
2	2	1	0	0	1
3	3	1	0	0	0
4	4	0	3	1	0
...	...	...	...	...	...
25475	17204	1	0	1	1
25476	17205	1	2	1	0
25477	17206	1	3	1	0
25478	17207	1	3	1	1
25479	17209	1	0	1	0

25480 rows × 12 columns

In [171]:
```
le.inverse_transform(visa_df['continent'][:5])
```

Out[171]:
```
array(['Denied', 'Denied', 'Denied', 'Denied', 'Certified'], dtype=object)
```

In [172]:
```
visa_df['continent'][:5]
```

Out[172]:
```
0    1
1    1
2    1
3    1
4    0
Name: continent, dtype: int32
```

In [173]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)


from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
visa_df['continent']=le.fit_transform(visa_df['continent'])
```

In [174]:
```
le.inverse_transform(visa_df['continent'])
```

Out[174]:
```
array(['Asia', 'Asia', 'Asia', ..., 'Asia', 'Asia', 'Asia'], dtype=object)
```

In [ ]:
```
# np.where
```

In [ ]:
```
-np.where required 3 arguments

-condtion

-True

-False

-it is applicabale only for binary labels

-case status has only two lables certified and denied

-if case_status ===certified replace that as 0,otherwise 1
```

In [175]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)
```

In [176]:
```
con=visa_df['case_status']=='Certified'
visa_df['case_status']=np.where(con,0,1)
visa_df
```

Out[176]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 12 columns

In [ ]:
```
## one hot encoder:

- one hot encoder name sugguest at a time one will on and other will off

- for example case_statues has two labels
 -certified
 -denied

- when you apply one hot encoding on case _status ,it cretaes two more extra

-case_status _certified
-case_status_denied
```

In [ ]:
```
# advantages

- when you develop ml model it is very important the column should be indepe

- so here case_status creating two extra columns

- which are independent ,which means the row values at a time omly one colu
-columns are inpendent each other,\
-which means 90 degree phase shift
-which means perpendicular each other
-which means orthogonal each other
```

In [ ]:
```
# dis advantages

- the dis advanatages is if a column has 100 unique labels, 100 new column v

- the data willl become sparse ,which means huge
-the processing time is more
- the memory consumption is more
-#curse of dimensionality
```

In [ ]:
```
# pd.get_dummies
```

In [177]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)
#make sure drop the id column
visa_df.drop('case_id',axis=1,inplace=True)
# when you dont
pd.get_dummies(visa_df,dtype='int')
```

Out[177]:

	no_of_employees	yr_of_estab	prevailing_wage	continent_Africa	continent_Asia	cont
0	14513	2007	592.2029	0	1	
1	2412	2002	83425.6500	0	1	
2	44444	2008	122996.8600	0	1	
3	98	1897	83434.0300	0	1	
4	1082	2005	149907.3900	1	0	
...	...	...	...	...	...	
25475	2601	2008	77092.5700	0	1	
25476	3274	2006	279174.7900	0	1	
25477	1121	1910	146298.8500	0	1	
25478	1918	1887	86154.7700	0	1	
25479	3195	1960	70876.9100	0	1	

25480 rows × 30 columns

In [190]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
visa_df=pd.read_csv(path)
visa_df.head(3)
```

Out[190]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training	no_
0	EZYV01	Asia	High School	N	N	
1	EZYV02	Asia	Master's	Y	N	
2	EZYV03	Asia	Bachelor's	N	Y	

In [ ]:
```
# standardization ##

- satndardization means scaling the data into one scale

- we have different columns has different units so that the value will vary

- one column has very huge values

- another column has very less values

- so it is important to scale all type of units under one scale

- we have 2 producers

- standardization

- -Z-score:
    x-uniue\sigma

- the values range from -3 to 3
```

In [ ]:
```
- Normalization

 - min max scalar =x-xmin\xmax-xmin
```

In [ ]:
```
-# step:1 : take the prevailing wage column
-# z-score= x-mean\sigma
- # step-2: cacluate mean of prervailing wage
- # step-3: calculateb std of prervailing wage
- # step-4: nr: preaviling -mean
- # step-5: pwage_zscore=nr\dr
```

In [191]:
```
cols=visa_df.select_dtypes(include='object').columns
visa_df[cols]
```

Out[191]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 9 columns

In [192]:
```
pwage=visa_df['prevailing_wage']
pwage_mean=visa_df['prevailing_wage'].mean()
pwage_std=visa_df['prevailing_wage'].std()
Nr=pwage-pwage_mean
visa_df['prevailing_wage_z']=Nr/pwage_std
```

In [193]:
```
visa_df[['prevailing_wage','prevailing_wage_z']]
```

Out[193]:

	prevailing_wage	prevailing_wage_z
0	592.2029	-1.398510
1	83425.6500	0.169832
2	122996.8600	0.919060
3	83434.0300	0.169991
4	149907.3900	1.428576
...	...	...
25475	77092.5700	0.049923
25476	279174.7900	3.876083
25477	146298.8500	1.360253
25478	86154.7700	0.221504
25479	70876.9100	-0.067762

25480 rows × 2 columns

In [194]:
```
visa_df['prevailing_wage'].max(),visa_df['prevailing_wage_z'].max()
#99.7% dsata between -3 to 3
```

Out[194]: (319210.27, 4.634101837909902)

In [195]:
```
visa_df['prevailing_wage'].idxmax()

# in the prevailing_wage column the maximim value id is 21077
```

Out[195]: 21077

In [196]:
```
visa_df['prevailing_wage_z'].idxmax()
```

Out[196]: 21077

In [197]:
```
visa_df['prevailing_wage'].min(),visa_df['prevailing_wage_z'].min()
```

Out[197]: (2.1367, -1.4096818992891214)

In [198]:
```
visa_df['prevailing_wage'].idxmin()
```

Out[198]: 20575

In [199]:
```
visa_df['prevailing_wage_z'].idxmin()
```

Out[199]: 20575

In [200]:
```
#can you get only two rows values


visa_df.iloc[[21077,20575]]
```

Out[200]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
21077	EZYV21078	Asia	High School	N	
20575	EZYV20576	North America	Master's	N	

In [201]:
```
cols=['prevailing_wage','prevailing_wage_z']
ids=[2107,20575]
visa_df[['prevailing_wage','prevailing_wage_z']].iloc[[2107,20575]]
visa_df[cols].iloc[ids]
```

Out[201]:

	prevailing_wage	prevailing_wage_z
2107	56741.4400	-0.335398
20575	2.1367	-1.409682

In [ ]:
```
# generally will overwrite the column values
# becuase we want to  clean our data before we apply ml model
# if you create any extra column make column make sure drop same of non requ
```

In [202]:
```
# Satandard Scalar

#-read the package
#-save the package
#-apply fit trans

from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
visa_df['prevailing_wage_ss']=ss.fit_transform(visa_df[['prevailing_wage']]
```

In [203]:
```
cols=['prevailing_wage','prevailing_wage_z','prevailing_wage_ss']
visa_df[cols]
```

Out[203]:

	prevailing_wage	prevailing_wage_z	prevailing_wage_ss
0	592.2029	-1.398510	-1.398537
1	83425.6500	0.169832	0.169835
2	122996.8600	0.919060	0.919079
3	83434.0300	0.169991	0.169994
4	149907.3900	1.428576	1.428604
...	...	...	...
25475	77092.5700	0.049923	0.049924
25476	279174.7900	3.876083	3.876159
25477	146298.8500	1.360253	1.360280
25478	86154.7700	0.221504	0.221509
25479	70876.9100	-0.067762	-0.067763

25480 rows × 3 columns

In [ ]:
```
## Normalization ##

# x-x_max(x_max-x_min)

# step:1: read the pwage column
# step-2: find the min value of the pwage column
# step-3:find the max value of the pwage column
# step-4: dr =max_value-min_value
# step-5:nr\dr
```

In [204]:
```
visa_df
```

Out[204]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_traini
0	EZYV01	Asia	High School	N	
1	EZYV02	Asia	Master's	Y	
2	EZYV03	Asia	Bachelor's	N	
3	EZYV04	Asia	Bachelor's	N	
4	EZYV05	Africa	Master's	Y	
...	...	...	...	...	
25475	EZYV25476	Asia	Bachelor's	Y	
25476	EZYV25477	Asia	High School	Y	
25477	EZYV25478	Asia	Master's	Y	
25478	EZYV25479	Asia	Master's	Y	
25479	EZYV25480	Asia	Bachelor's	Y	

25480 rows × 14 columns

In [ ]:
```
# min max scalar method:
```

In [205]:
```
from sklearn.preprocessing import MinMaxScaler
mns=MinMaxScaler()
visa_df['prevailing_wage_mns']=ss.fit_transform(visa_df[['prevailing_wage']
```

In [206]:
```
cols=['prevailing_wage','prevailing_wage_mns']
visa_df[cols]
```

Out[206]:

	prevailing_wage	prevailing_wage_mns
0	592.2029	-1.398537
1	83425.6500	0.169835
2	122996.8600	0.919079
3	83434.0300	0.169994
4	149907.3900	1.428604
...	...	...
25475	77092.5700	0.049924
25476	279174.7900	3.876159
25477	146298.8500	1.360280
25478	86154.7700	0.221509
25479	70876.9100	-0.067763

25480 rows × 2 columns

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
path=r"C:\Users\kasho\OneDrive\Documents\Data Science\data files\Visadataset
df=pd.read_csv(path)
df
```

Out[4]:

	case_id	continent	education_of_employee	has_job_experience	requires_job_training
0	EZYV01	Asia	High School	N	N
1	EZYV02	Asia	Master's	Y	N
2	EZYV03	Asia	Bachelor's	N	Y
3	EZYV04	Asia	Bachelor's	N	N
4	EZYV05	Africa	Master's	Y	N
...	...	...	...	...	...
25475	EZYV25476	Asia	Bachelor's	Y	Y
25476	EZYV25477	Asia	High School	Y	N
25477	EZYV25478	Asia	Master's	Y	N
25478	EZYV25479	Asia	Master's	Y	Y
25479	EZYV25480	Asia	Bachelor's	Y	N

25480 rows × 12 columns

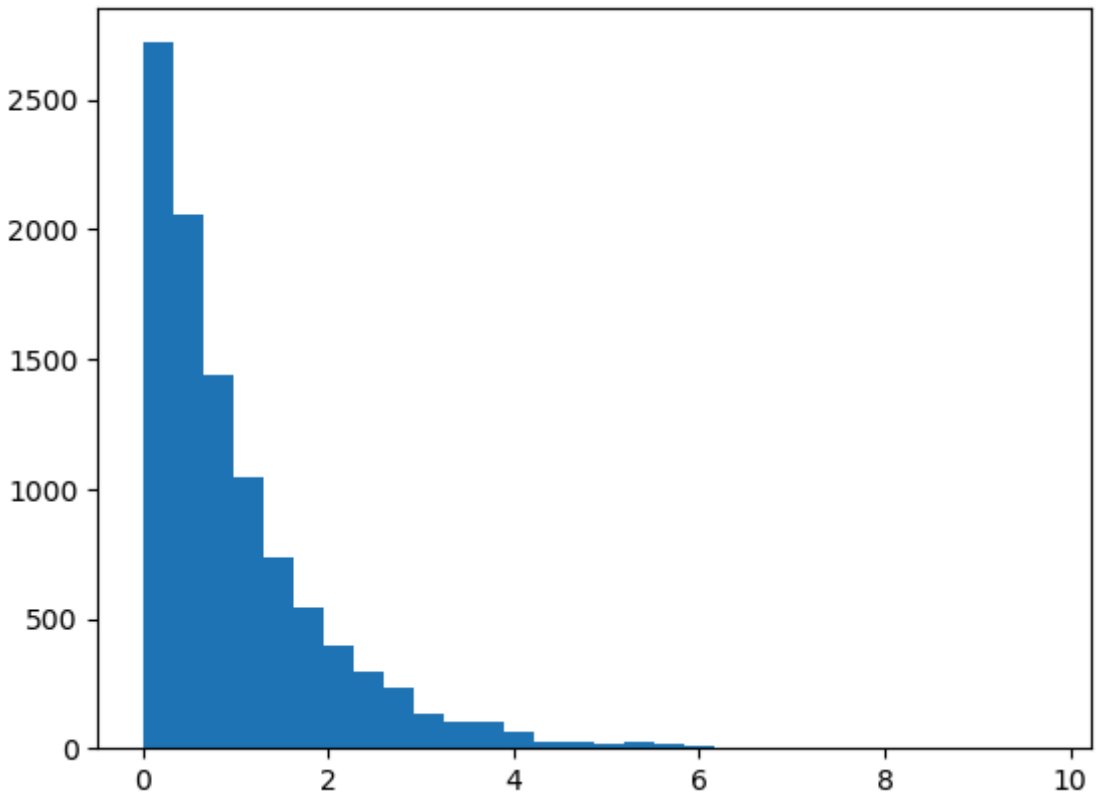
In [5]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [6]:
```
exp_data=np.random.exponential(size=10000)
exp_data[:10]
```

Out[6]:
```
array([0.59557919, 0.21743725, 0.0513954 , 0.58489996, 0.11544657,
       2.86593082, 2.75677147, 0.58587141, 0.19787968, 2.20027473])
```

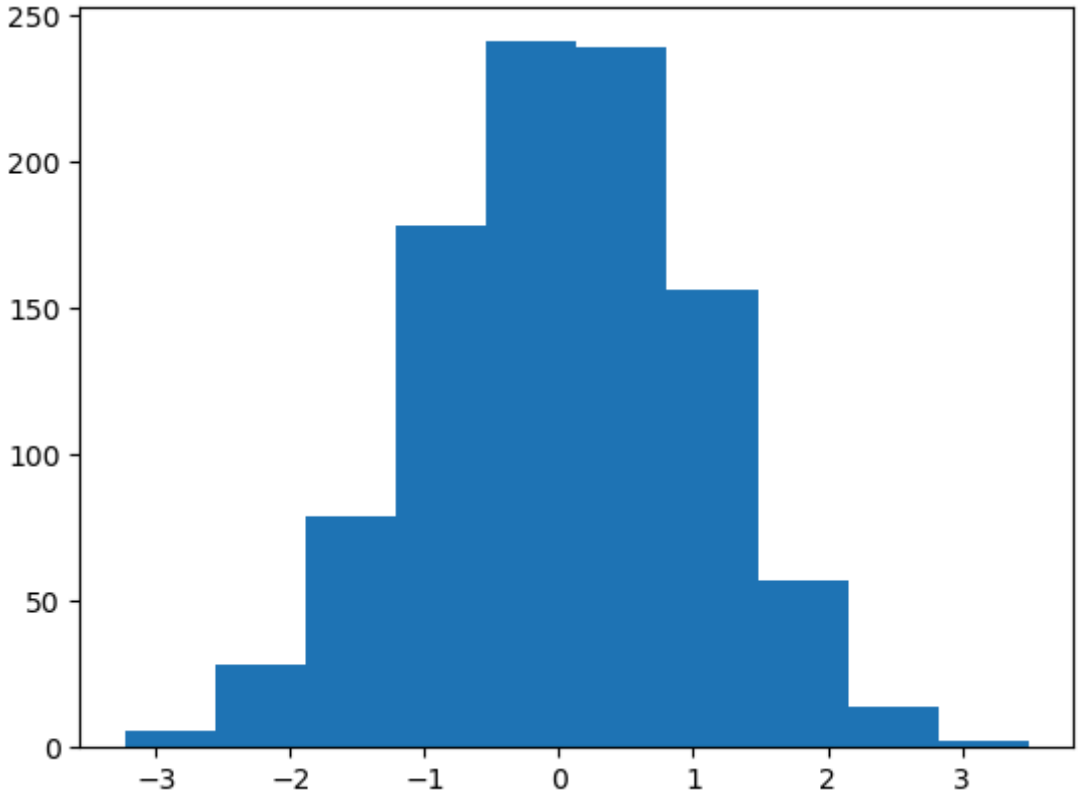
In [7]:
```
plt.hist(exp_data,bins=30,label='Exponential')
plt.lengend()
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[7], line 2
      1 plt.hist(exp_data,bins=30,label='Exponential')
----> 2 plt.lengend()
      3 plt.show()

AttributeError: module 'matplotlib.pyplot' has no attribute 'lengend'
```



In [8]:
```
norm_data=np.random.normal(size=1000)
plt.hist(norm_data)
```

Out[8]:
```
(array([  6.,  28.,  79., 178., 241., 239., 156.,  57.,  14.,   2.]),
 array([-3.23144467, -2.55864908, -1.88585348, -1.21305789, -0.5402623 ,
         0.1325333 ,  0.80532889,  1.47812448,  2.15092007,  2.82371567,
         3.49651126]),
 <BarContainer object of 10 artists>)
```



In [ ]:
```
# log transformation
np.log is used for transformation
```

In [9]:
```
x=2
import numpy as np
np.log(2)
```

Out[9]:
```
0.6931471805599453
```

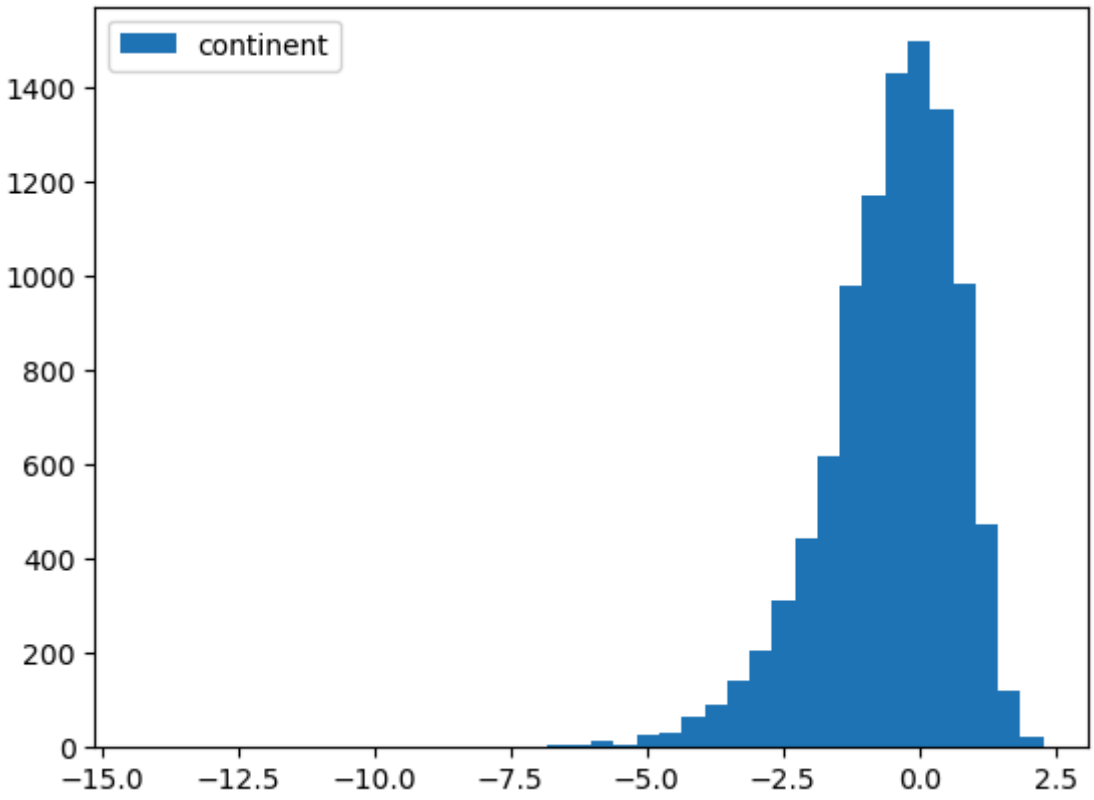
In [10]:
```
log_data=np.log(exp_data)
log_data[:10]
```

Out[10]:
```
array([-0.51822091, -1.52584498, -2.96820654, -0.53631445, -2.15894742,
        1.05289319,  1.01406024, -0.53465494, -1.62009612,  0.78858223])
```

In [11]:
```
exp_data[:10]
```

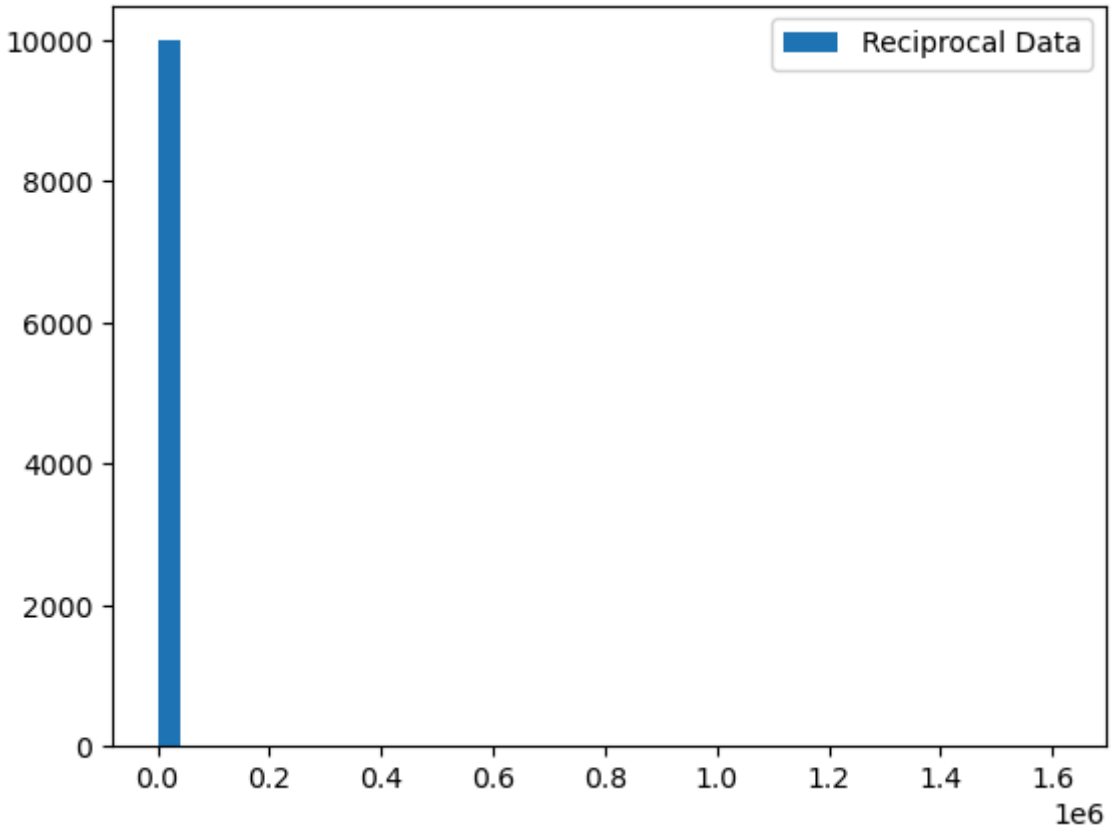
Out[11]:
```
array([0.59557919, 0.21743725, 0.0513954 , 0.58489996, 0.11544657,
       2.86593082, 2.75677147, 0.58587141, 0.19787968, 2.20027473])
```

In [12]:
```
plt.hist(log_data,bins=40,label='continent')
plt.legend()
plt.show()
```



In [ ]:
```
# reciporcal transformation
 reciprocial transformation fails when data has zero value
```

In [15]:
```
rec_data=np.reciprocal(exp_data)
plt.hist(rec_data,bins=40,label='Reciprocal Data')
plt.legend()
plt.show()
```



In [16]:
```
exp_data,rec_data
```

Out[16]:
```
(array([0.59557919, 0.21743725, 0.0513954 , ..., 0.36186399, 0.18173243,
        0.11902111]),
 array([ 1.67903783,  4.599028  , 19.45699295, ...,  2.7634692 ,
         5.50259536,  8.40187075]))
```

In [17]:
```
1/0.77
```

Out[17]:
```
1.2987012987012987
```

In [18]:
```
exp_data[:2]
```

Out[18]:
```
array([0.59557919, 0.21743725])
```

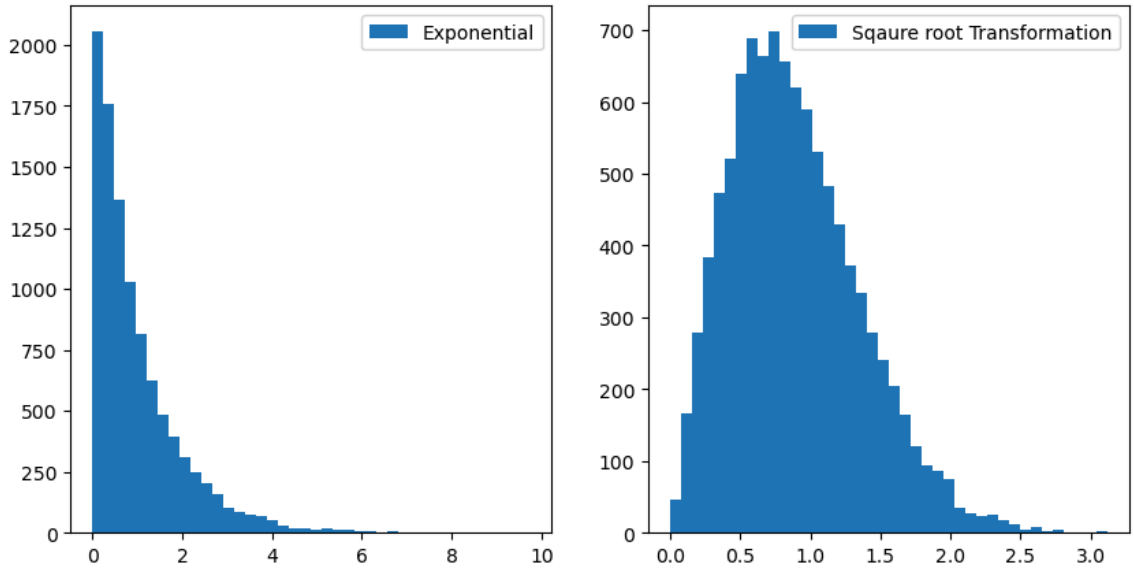
In [ ]:
```
#Square root transformation
```

In [19]:
```
print(25**2) # square,
print(25**(1/2)) # square root
print(np.sqrt(25))
```

```
625
5.0
5.0
```

In [20]:
```
sqrt_data=np.sqrt(exp_data)
```

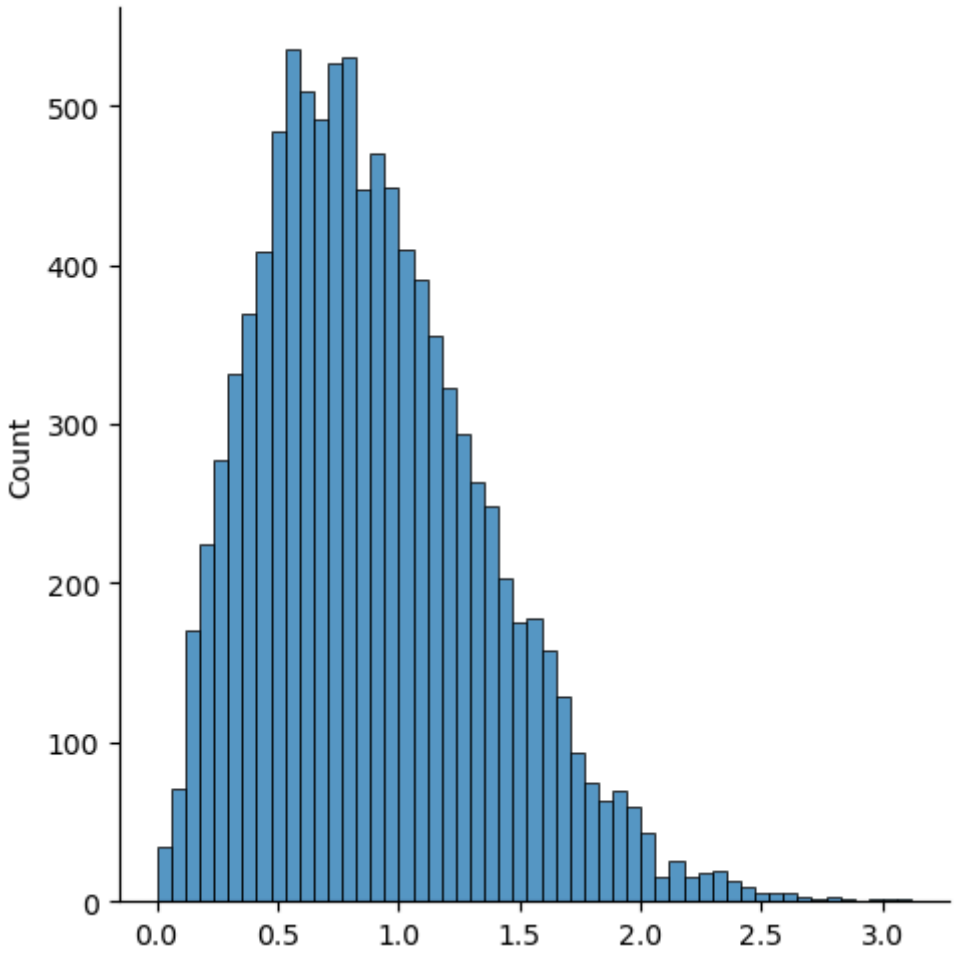
In [21]:
```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1).hist(exp_data,
bins=40,

label='Exponential')

plt.legend()
plt.subplot(1,2,2).hist(sqrt_data,
bins=40,

label='Sqaure root Transformation')

plt.legend()
plt.show()
```



In [22]:
```
import seaborn as sns
sns.displot(sqrt_data)
```

```
C:\Users\kasho\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWa
rning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x2296b0a4d50>
```



In [ ]:
```
#Power transformer
It is related to sklearn package
Package name: sklearn.preprocessing
Method name: Power Transformer
Inside Box-Cox , yeo-jhonson
```

In [23]:
```
exp_data
```

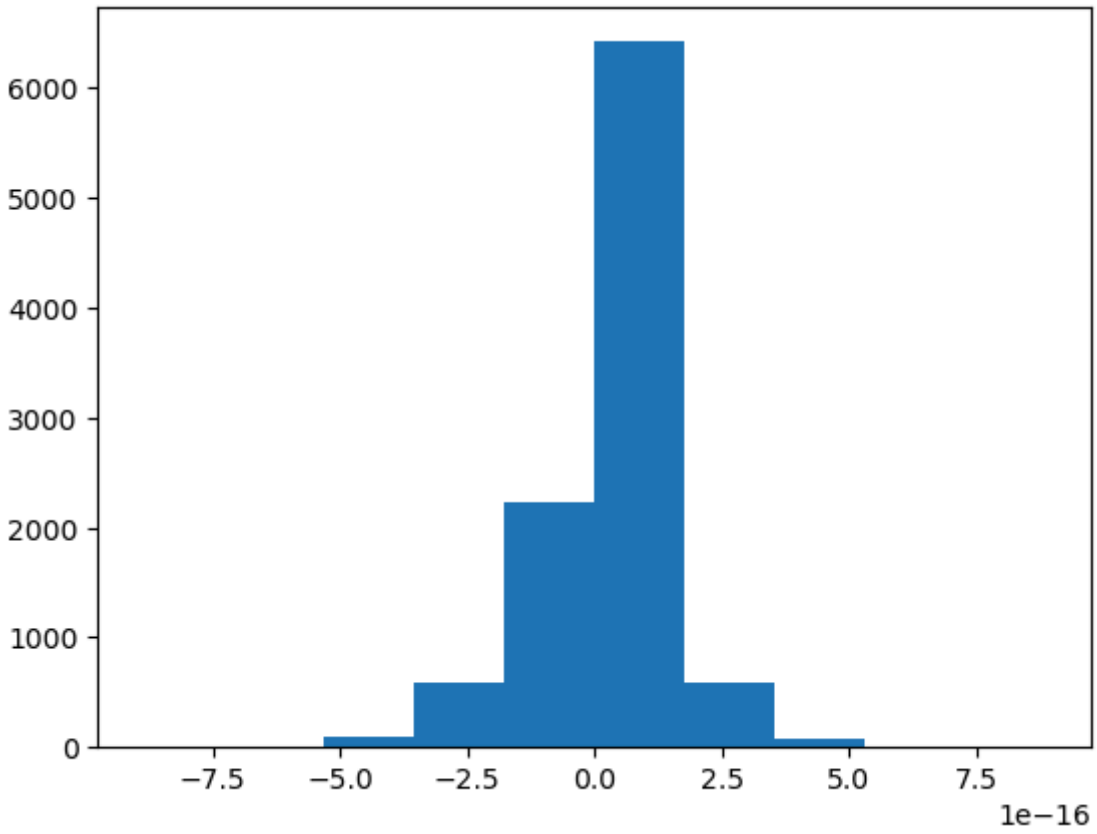
Out[23]:
```
array([0.59557919, 0.21743725, 0.0513954 , ..., 0.36186399, 0.18173243,
       0.11902111])
```

In [24]:
```
from sklearn.preprocessing import PowerTransformer
pt=PowerTransformer()
pt_data=pt.fit([exp_data]).transform([exp_data])
```

In [25]:
```
pt_data
```

Out[25]:
```
array([[ 1.11022302e-16,  5.55111512e-17, -9.71445147e-17, ...,
        -1.11022302e-16, -5.55111512e-17, -9.71445147e-17]])
```

In [26]:
```
plt.hist(pt_data[0])
plt.show()
```



In [27]:
```
import numpy as np
import pandas as pd
```

In [28]:
```
dict1={'Names':['Ramesh','Suresh',np.nan,'Mahesh'],
'Age':[31,32,33,np.nan],
'City':[np.nan,'Hyd','Mumbai','Chennai']}
```

In [29]:
```
data1=pd.DataFrame(dict1)
```

In [30]:
```
data1.isnull()
```

Out[30]:

	Names	Age	City
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

In [31]:
```
data1.isnull().sum()
# every column has one missing value is there
```

Out[31]:
```
Names    1
Age      1
City     1
dtype: int64
```

In [32]:
```
data1.isnull().sum()/len(data1)
```

Out[32]:
```
Names    0.25
Age      0.25
City     0.25
dtype: float64
```

In [33]:
```
data1.isnull().sum()*100/len(data1)
```

Out[33]:
```
Names    25.0
Age      25.0
City     25.0
dtype: float64
```

In [34]:
```
dict2={'Names':['Ramesh','Suresh',None,'Mahesh'],
'Age':[31,32,33,None],
'City':[None,'Hyd','Mumbai','Chennai']}
data2=pd.DataFrame(dict2)
data2
```

Out[34]:

	Names	Age	City
0	Ramesh	31.0	None
1	Suresh	32.0	Hyd
2	None	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [35]:
```
data2.isnull()
```

Out[35]:

	Names	Age	City
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

In [36]:
```
data2.isnull().sum()
```

Out[36]:
```
Names    1
Age      1
City     1
dtype: int64
```

In [37]:
```
dict3={'Names':['Ramesh','Suresh','Null','Mahesh'],
'Age':[31,32,33,'Null'],
'City':['Null','Hyd','Mumbai','Chennai']}
data3=pd.DataFrame(dict3)
data3
```

Out[37]:

	Names	Age	City
0	Ramesh	31	Null
1	Suresh	32	Hyd
2	Null	33	Mumbai
3	Mahesh	Null	Chennai

In [ ]:
```
#Method:1

Fill the missing values with random number
```

In [38]:
```
data1.fillna(40)
```

Out[38]:

	Names	Age	City
0	Ramesh	31.0	40
1	Suresh	32.0	Hyd
2	40	33.0	Mumbai
3	Mahesh	40.0	Chennai

In [ ]:
```
#Method:2
Fill the missing values with random number on specific column
```

In [39]:
```
data1['Names'].fillna('Sathish',inplace=True)
data1
```

Out[39]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	Sathish	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [40]:
```
#Create the data again
dict1={'Names':['Ramesh','Suresh',np.nan,'Mahesh'],
'Age':[31,32,33,np.nan],
'City':[np.nan,'Hyd','Mumbai','Chennai']}
data1=pd.DataFrame(dict1)
```

In [ ]:
```
#Method:3
bfill
ffill
pad
backfill
```

In [41]:
```
data1.fillna(method='backfill')
# Names index 2 has missed value
# it will replace by index 3 value
#Age index 3 has missed value
# we dont have index 4, so the value is NaN
# City index 0 has missed value
# it replace with index 1 value
```

Out[41]:

	Names	Age	City
0	Ramesh	31.0	Hyd
1	Suresh	32.0	Hyd
2	Mahesh	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [42]:
```
data1
```

Out[42]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	NaN	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [ ]:
```
bfill and backfill both are same
pad and ffill both are same
```

In [ ]:
```
#Method:4
Mean
Median
Mode
```

In [43]:
```
data1
```

Out[43]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	NaN	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [44]:
```
age_mean=data1['Age'].mean()
age_mean
```

Out[44]:
```
32.0
```

In [45]:
```
data1['Age'].fillna(age_mean)
```

Out[45]:
```
0    31.0
1    32.0
2    33.0
3    32.0
Name: Age, dtype: float64
```

In [46]:
```
# instead of providing a random number
# we are filling with mean of the data
age_median=data1['Age'].median()
age_median
data1['Age'].fillna(age_median)
```

Out[46]:
```
0    31.0
1    32.0
2    33.0
3    32.0
Name: Age, dtype: float64
```

In [ ]:
```
# Level1: Mean median mode
# Level2: bfill fill
# level3:
```

In [ ]:
```
#Method-5:KNN imputer

KNN: K nearest neighbours
in the KNN imputer instead of taking mean of all the values
will choose neigbours data
will take those mean only
```

In [47]:
```
from sklearn.impute import KNNImputer
knn=KNNImputer(n_neighbors=2)
knn.fit_transform(data1[['Age']])
```

Out[47]:
```
array([[31.],
       [32.],
       [33.],
       [32.]])
```

In [48]:
```
data1
```

Out[48]:

	Names	Age	City
0	Ramesh	31.0	NaN
1	Suresh	32.0	Hyd
2	NaN	33.0	Mumbai
3	Mahesh	NaN	Chennai

In [ ]:

In [ ]:

In [ ]:

In [ ]: