Test Case-1 : Check Outlier is exists or not ?

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

In [32]:
```
df = pd.read_csv("C://Users//Hp/Documents/Practice DataSets//Bank_Stock_Price_10Y.c
df.head()
```

Out[32]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-02-03	1980.0	2000.0	1965.0	1965.0	1691.382568	55407000
1	2014-02-04	1970.0	1980.0	1940.0	1970.0	1695.686035	83683500
2	2014-02-05	1980.0	1990.0	1965.0	1990.0	1712.901367	42715000
3	2014-02-06	1975.0	2030.0	1970.0	2030.0	1747.331299	63581000
4	2014-02-07	2050.0	2060.0	2035.0	2050.0	1764.546753	104825500

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

Out[5]:
```
(2483, 7)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2483 entries, 0 to 2482
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2483 non-null   object 
 1   Open       2483 non-null   float64
 2   High       2483 non-null   float64
 3   Low        2483 non-null   float64
 4   Close      2483 non-null   float64
 5   Adj Close  2483 non-null   float64
 6   Volume     2483 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 135.9+ KB
```

In [16]:
```
df.describe()
```

Out[16]:

	Height	Weight
count	10000.000000	10000.000000
mean	66.367560	161.440357
std	3.847528	32.108439
min	54.263133	64.700127
25%	63.505620	135.818051
50%	66.318070	161.212928
75%	69.174262	187.169525
max	78.998742	269.989699

In [1]:
```
import outlier_process
```

In [2]:
```
from outlier_process import about_library
```

In [3]:
```
about_library.info()
```

```
.........Welcome Outlier of Python Package.........

@ Module-1)

--> first check in your column data in exists outlier or not...
--> if yes then detect outlier in column data.
--> Ex :- import outlier_process
          from outlier_process import detection_process
          detection_process.detection.outlier_detect(df,column_name)

@ Module-2)

--> Now we remove outlier data in your column data using by trimming method.
--> Ex :- import outlier_process
          from outlier_process import clean_by_trimming
          new_variable = clean_by_trimming.clean1.trim_process(df,column_name)

@ Module-3)

--> Also we can do this change the outlier value to
upper limit or lower limit using by capping method
--> Ex :- import outlier_process
          from outlier_process import clean_by_capping
          new_variable = clean_by_capping.clean2.cap_process(df,column_name)

@ Recommendations :

--> In Statistic, the choice between trimming and capping methods depends on
various factors such as the nature of the data and the specific analysis being con
ducted.
--> just if founded outlier between 0 % to 1% lie so you can use trimming method
& if founded outlier above 1% lie so you can use capping method.
```

In [33]:
```
from outlier_process import detection_process
```

In [34]:
```
detection_process.detection.outlier_detect(df,"Open")
```

```
Outlier are not found for your selected column in this dataset.
```

In [48]:
```
print()
```

Test Case-2 : If Outlier is exists so you can you detect and which percent % (check by new database)

In [35]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [37]:
```
df1 = pd.read_csv("C://Users//Hp/Documents/Practice DataSets//weight-height.csv")
df1.head()
```

Out[37]:

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

In [38]:
```
df1.shape
```

Out[38]:
```
(10000, 3)
```

In [41]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Gender  10000 non-null  object 
 1   Height  10000 non-null  float64
 2   Weight  10000 non-null  float64
dtypes: float64(2), object(1)
memory usage: 234.5+ KB
```

In [42]:
```
df1.describe()
```

Out[42]:

	Height	Weight
count	10000.000000	10000.000000
mean	66.367560	161.440357
std	3.847528	32.108439
min	54.263133	64.700127
25%	63.505620	135.818051
50%	66.318070	161.212928
75%	69.174262	187.169525
max	78.998742	269.989699

In [43]:
```
import outlier_process
```

In [44]:
```
from outlier_process import about_library
```

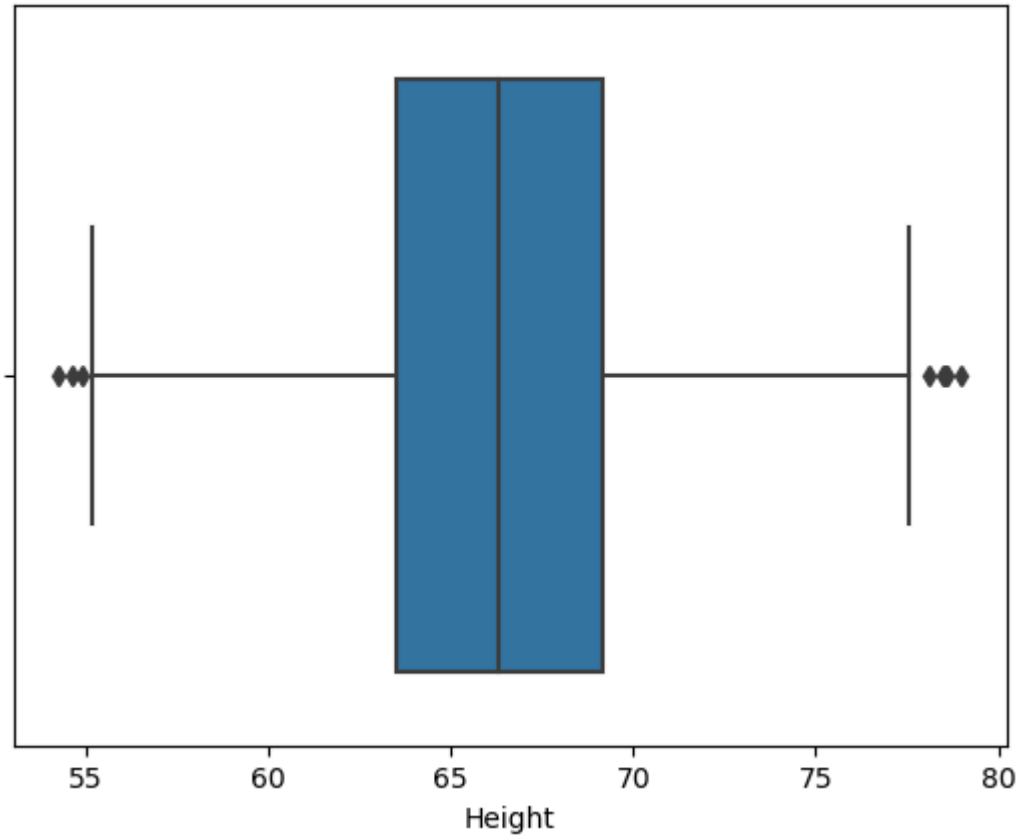
In [4]:
```
about_library.info()
```

```
.........Welcome Outlier of Python Package.........

@ Module-1)

--> first check in your column data in exists outlier or not...
--> if yes then detect outlier in column data.
--> Ex :- import outlier_process
          from outlier_process import detection_process
          detection_process.detection.outlier_detect(df,column_name)

@ Module-2)

--> Now we remove outlier data in your column data using by trimming method.
--> Ex :- import outlier_process
          from outlier_process import clean_by_trimming
          new_variable = clean_by_trimming.clean1.trim_process(df,column_name)

@ Module-3)

--> Also we can do this change the outlier value to
upper limit or lower limit using by capping method
--> Ex :- import outlier_process
          from outlier_process import clean_by_capping
          new_variable = clean_by_capping.clean2.cap_process(df,column_name)

@ Recommendations :

--> In Statistic, the choice between trimming and capping methods depends on
various factors such as the nature of the data and the specific analysis being con
ducted.
--> just if founded outlier between 0 % to 1% lie so you can use trimming method
& if founded outlier above 1% lie so you can use capping method.
```

In [46]:
```
from outlier_process import detection_process
```

In [47]:
```
detection_process.detection.outlier_detect(df1,"Height")
```

```
~> Outliers :  [78.0958674715774, 78.4620529193772, 78.9987423463896, 78.528210425
8694, 78.621373968548, 54.6168578301035, 54.8737275315254, 54.2631333250971]

~> Count of Outliers :  8

~> Percentage of Outlier in your selectd column :  0.08 %
```



In [49]:
```
print()
```

Test Case-3 : After finding the outlier, now remove it using the trimming method

In [50]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [51]:
```
df1 = pd.read_csv("C://Users//Hp/Documents/Practice DataSets//weight-height.csv")
df1.head()
```

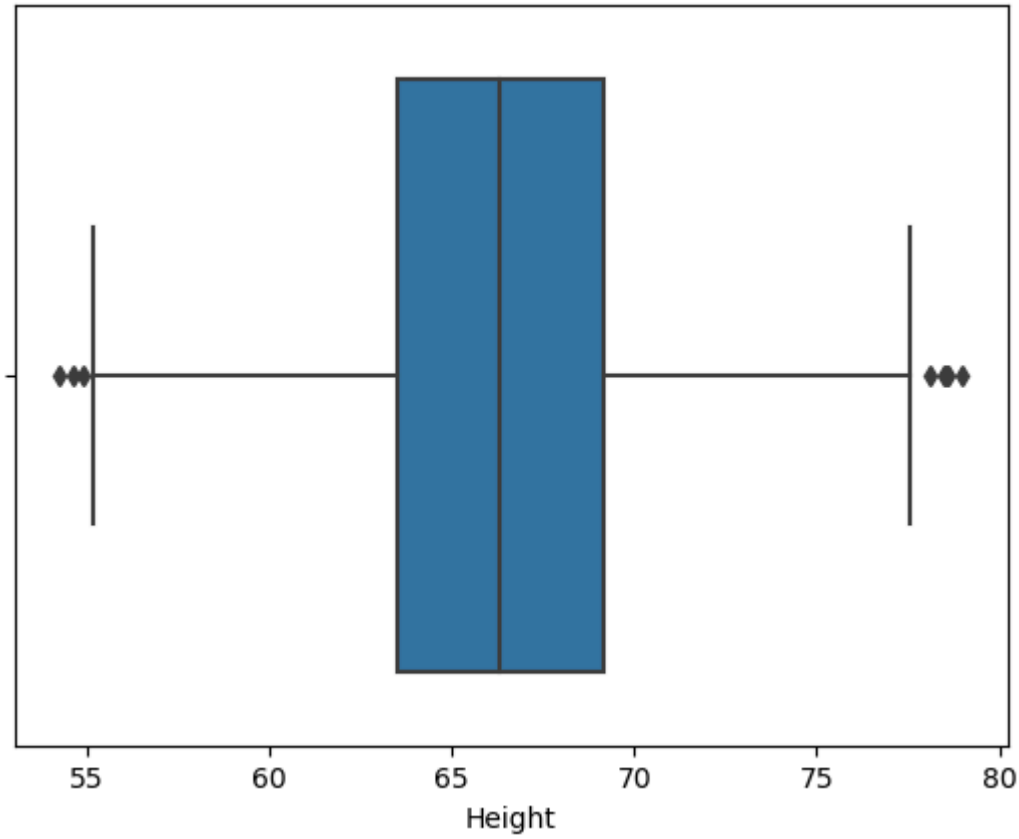
Out[51]:

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

In [52]:
```
from outlier_process import detection_process
```

In [53]:
```
detection_process.detection.outlier_detect(df1,"Height")
```

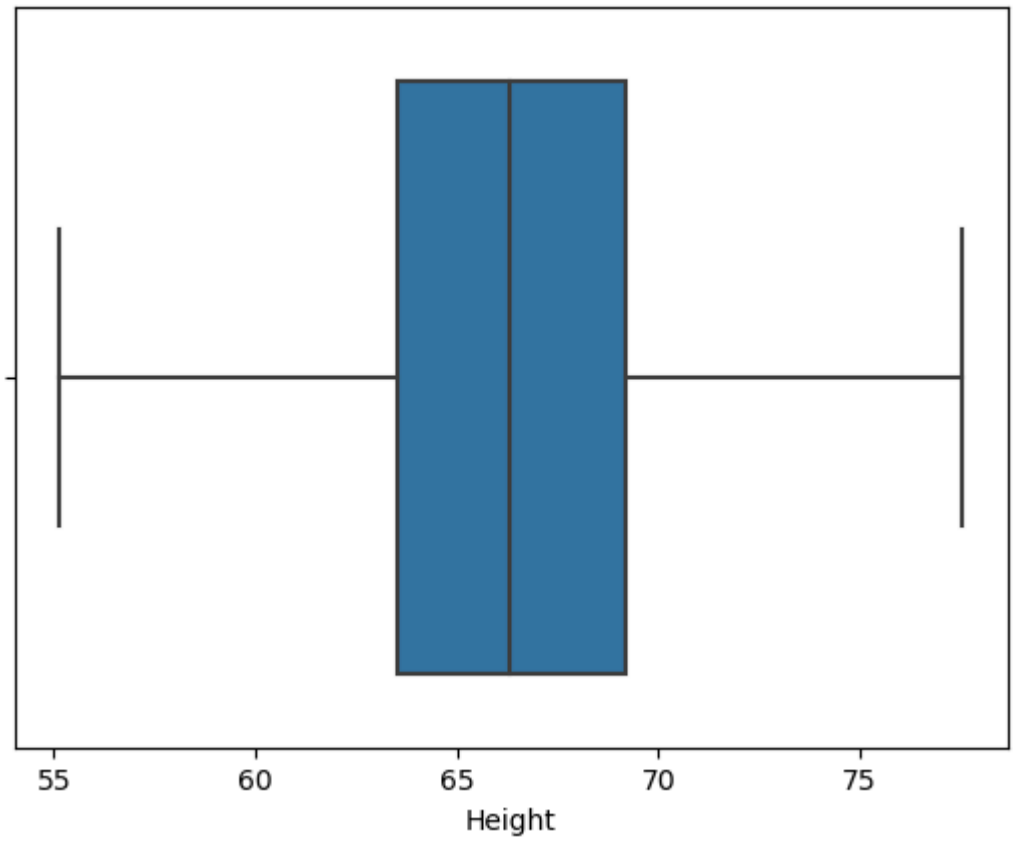
```
~> Outliers :  [78.0958674715774, 78.4620529193772, 78.9987423463896, 78.528210425
8694, 78.621373968548, 54.6168578301035, 54.8737275315254, 54.2631333250971]

~> Count of Outliers :  8

~> Percentage of Outlier in your selectd column :  0.08 %
```



In [54]:
```
from outlier_process import clean_by_trimming
```

In [56]:
```
t_df = clean_by_trimming.clean1.trim_process(df1,"Height")
```

```
~> Data before cleaning outliers within the data :  10000
~> Number of outliers within the data            :  8
~> Data after cleaning outliers within the data  :  9992
```



In [57]:
```
t_df    # remove data show at by new data frame
```

Out[57]:

	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801
...	...	...	...
9995	Female	66.172652	136.777454
9996	Female	67.067155	170.867906
9997	Female	63.867992	128.475319
9998	Female	69.034243	163.852461
9999	Female	61.944246	113.649103

9992 rows × 3 columns

In [59]:
```
print()
```

Test Case-4 : After finding the outlier, now clean it using the capping method (check by new database)

In [60]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [61]:
```
df2 = pd.read_csv("C://Users//Hp/Documents/Practice DataSets//jobs_in_data.csv")
df2.head()
```

Out[61]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_usd	employee_residence
0	2023	Data DevOps Engineer	Data Engineering	EUR	88000	95012	Germany
1	2023	Data Architect	Data Architecture and Modeling	USD	186000	186000	United States
2	2023	Data Architect	Data Architecture and Modeling	USD	81800	81800	United States
3	2023	Data Scientist	Data Science and Research	USD	212000	212000	United States
4	2023	Data Scientist	Data Science and Research	USD	93300	93300	United States

In [63]:
```
df.shape
```

Out[63]:
```
(2483, 7)
```

In [64]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2483 entries, 0 to 2482
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2483 non-null   object 
 1   Open       2483 non-null   float64
 2   High       2483 non-null   float64
 3   Low        2483 non-null   float64
 4   Close      2483 non-null   float64
 5   Adj Close  2483 non-null   float64
 6   Volume     2483 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 135.9+ KB
```

In [65]:
```
df.describe()
```

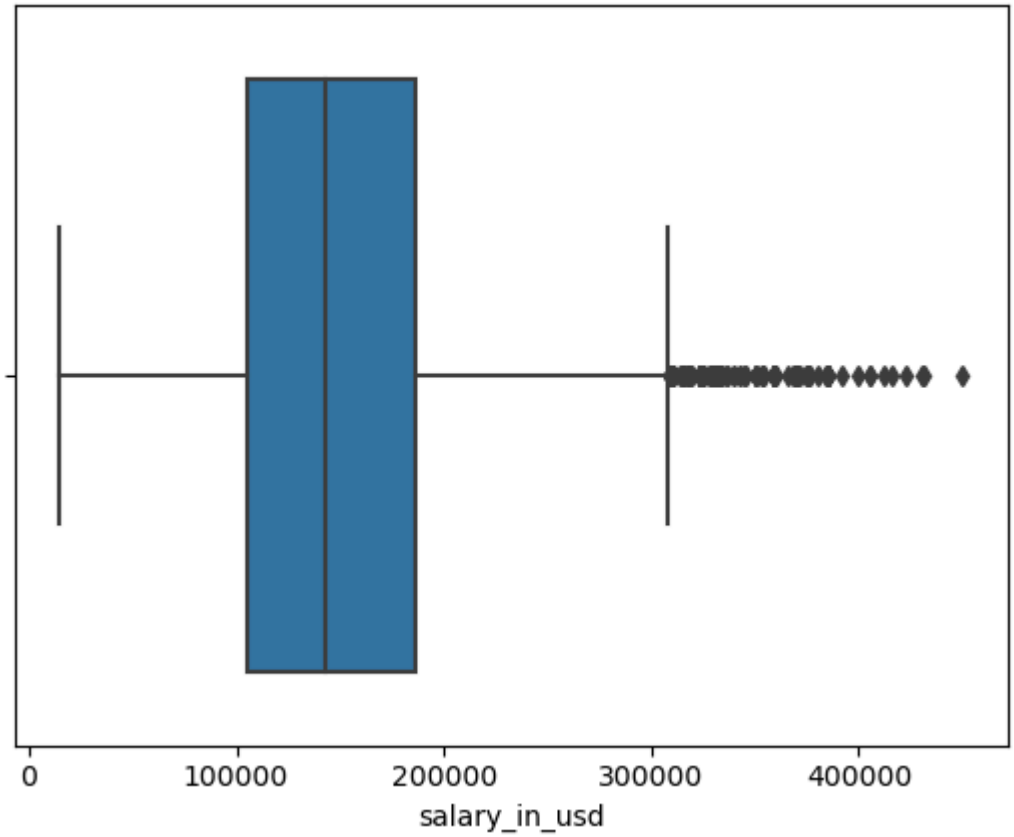
Out[65]:

	Open	High	Low	Close	Adj Close	Volume
count	2483.000000	2483.000000	2483.000000	2483.000000	2483.000000	2.483000e+03
mean	5219.973822	5265.847765	5173.628675	5219.887233	4886.148684	7.997496e+07
std	2223.156537	2240.113146	2206.459905	2223.903144	2276.934419	5.378122e+07
min	1970.000000	1980.000000	1940.000000	1965.000000	1691.382568	0.000000e+00
25%	2955.000000	2985.000000	2930.000000	2950.000000	2612.564454	5.153575e+07
50%	5170.000000	5235.000000	5120.000000	5180.000000	4736.543945	7.009800e+07
75%	6822.500000	6890.000000	6740.000000	6800.000000	6349.964111	9.651755e+07
max	9775.000000	9775.000000	9675.000000	9750.000000	9750.000000	1.062862e+09

In [67]:
```
from outlier_process import detection_process
```

In [68]:
```
detection_process.detection.outlier_detect(df2,"salary_in_usd")
```

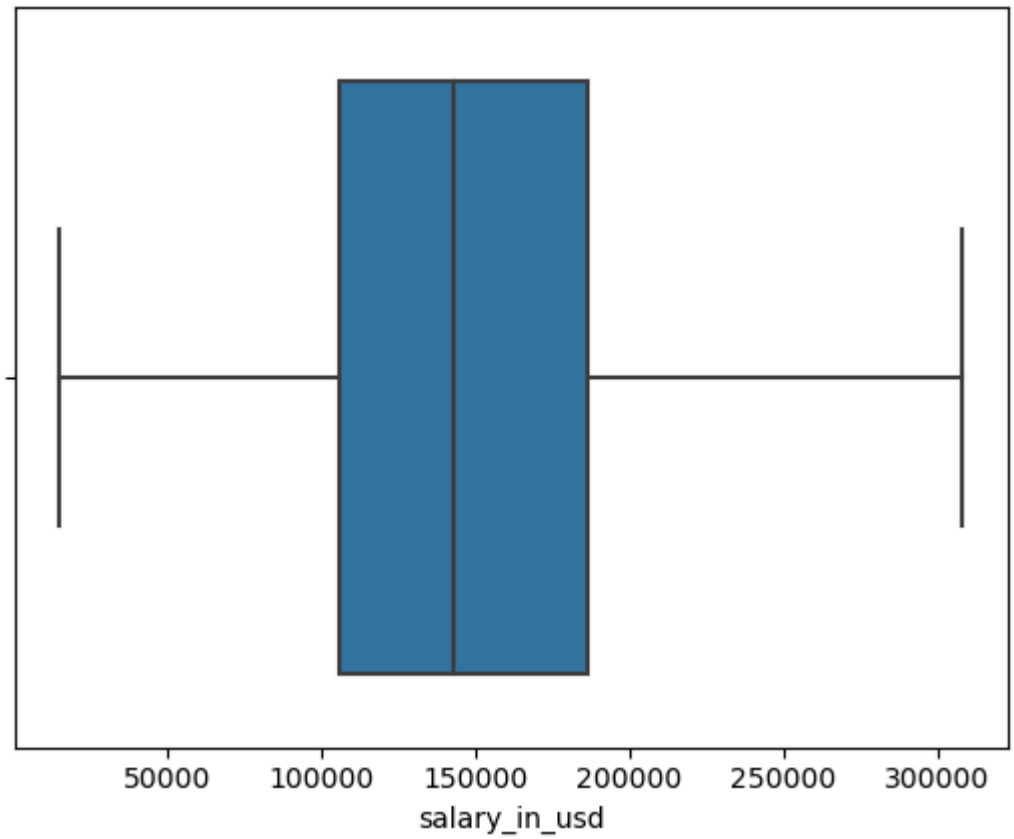
```
~> Outliers :  [319000, 331640, 336300, 323300, 324000, 370000, 333500, 354200, 32
8400, 329700, 310000, 372000, 324000, 370000, 309000, 324000, 331640, 369120, 3160
00, 405000, 353200, 350000, 385000, 385000, 310000, 385000, 310000, 369120, 36912
0, 360000, 330000, 324000, 370000, 316000, 330000, 329500, 340000, 309000, 385000,
330000, 328133, 309000, 375500, 309400, 385000, 330000, 329500, 323300, 324000, 30
9000, 370000, 359170, 359170, 323300, 370000, 330000, 325000, 331640, 323300, 3102
70, 350000, 315300, 359170, 310270, 323300, 309000, 323300, 330000, 324000, 38391
0, 315850, 323300, 392000, 330000, 359170, 310270, 359170, 331640, 331640, 331640,
331640, 323300, 359170, 316000, 365630, 315850, 309400, 331640, 323300, 330000, 31
0270, 331640, 331640, 359170, 310270, 331640, 359170, 330000, 309400, 331640, 3591
70, 370000, 345000, 370000, 309400, 374000, 370000, 331640, 309400, 399880, 31707
0, 333500, 311000, 329500, 328000, 323905, 336400, 370000, 370000, 430640, 342810,
309400, 342300, 318300, 309400, 329500, 353200, 317070, 376080, 340000, 310000, 31
0000, 370000, 323300, 310000, 375000, 318300, 385000, 370000, 314100, 350000, 3100
00, 310000, 310000, 430967, 310000, 375000, 350000, 315000, 345600, 324000, 40500
0, 380000, 450000, 416000, 325000, 423000, 412000]

~> Count of Outliers :  158

~> Percentage of Outlier in your selectd column :  1.69 %
```



In [69]:
```
from outlier_process import clean_by_capping
```

In [71]:
```
c_df = clean_by_capping.clean2.cap_process(df2,"salary_in_usd")
```

```
Data Before Outlier :  9355
Data After Outlier :  9355
Outlier =  0

founded outliers are existing upper boundry and lower boundry
```



In [72]:
```
c_df    # remove data show at by new data frame
```

Out[72]:

	work_year	job_title	job_category	salary_currency	salary	salary_in_usd	employee_residen
0	2023	Data DevOps Engineer	Data Engineering	EUR	88000	95012.0	Germa
1	2023	Data Architect	Data Architecture and Modeling	USD	186000	186000.0	United Stat
2	2023	Data Architect	Data Architecture and Modeling	USD	81800	81800.0	United Stat
3	2023	Data Scientist	Data Science and Research	USD	212000	212000.0	United Stat
4	2023	Data Scientist	Data Science and Research	USD	93300	93300.0	United Stat
...	...	...	...	...	...	...	
9350	2021	Data Specialist	Data Management and Strategy	USD	165000	165000.0	United Stat
9351	2020	Data Scientist	Data Science and Research	USD	412000	308257.5	United Stat
9352	2021	Principal Data Scientist	Data Science and Research	USD	151000	151000.0	United Stat
9353	2020	Data Scientist	Data Science and Research	USD	105000	105000.0	United Stat
9354	2020	Business Data Analyst	Data Analysis	USD	100000	100000.0	United Stat

9355 rows × 12 columns