Exploratory Data Analysis practise

In [1]:
```
import pandas as pd
```

In [2]:
```
emp = pd.read_excel(r"C:\Users\Affan\OneDrive\Desktop\FSDS Course NIT\Prakash Si
```

In [3]:
```
emp
```

Out[3]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

In [4]:
```
emp.isnull().sum()
```

Out[4]:
```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

In [5]:
```
emp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       4 non-null      object
 3   Location  4 non-null      object
 4   Salary    6 non-null      object
 5   Exp       5 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

In [6]:
```
emp.describe()
```

Out[6]:

	Name	Domain	Age	Location	Salary	Exp
count	6	6	4	4	6	5
unique	6	6	4	4	6	5
top	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
freq	1	1	1	1	1	1

In [7]:
```
emp.isna()
```

Out[7]:

	Name	Domain	Age	Location	Salary	Exp
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	True	True	False	False
3	False	False	True	False	False	True
4	False	False	False	True	False	False
5	False	False	False	False	False	False

In [8]:
```
emp.head()
```

Out[8]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year

In [9]:
```
emp.tail()
```

Out[9]:

	Name	Domain	Age	Location	Salary	Exp
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

In [10]:
```
id(emp)
```

Out[10]:
```
1807361002672
```

In [11]:
```
len(emp)
```

Out[11]: 6

In [12]:
```
emp.shape
```

Out[12]: (6, 6)

In [13]:
```
emp.columns
```

Out[13]: Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')

In [14]:
```
len(emp.columns)
```

Out[14]: 6

In [15]:
```
emp['Name']
```

Out[15]:
```
0      Mike
1    Teddy^
2     Uma#r
3      Jane
4    Uttam*
5       Kim
Name: Name, dtype: object
```

In [16]:
```
emp['Domain']
```

Out[16]:
```
0     Datascience#$
1           Testing
2    Dataanalyst^^#
3       Ana^^lytics
4        Statistics
5               NLP
Name: Domain, dtype: object
```

In [17]:
```
emp[['Name','Domain','Age','Location','Salary','Exp']]
```

Out[17]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

data cleansing

In [18]:
```
emp['Name']
```

Out[18]:
```
0      Mike
1    Teddy^
2     Uma#r
3      Jane
4    Uttam*
5       Kim
Name: Name, dtype: object
```

In [19]:
```
emp['Name'] = emp['Name'].str.replace(r'\W','',regex=True) #W means we can parse
```

In [20]:
```
emp['Name']
```

Out[20]:
```
0     Mike
1    Teddy
2     Umar
3     Jane
4    Uttam
5      Kim
Name: Name, dtype: object
```

In [21]:
```
emp['Domain'] = emp['Domain'].str.replace(r'\W','',regex=True)
```

In [22]:
```
emp['Domain']
```

Out[22]:
```
0    Datascience
1        Testing
2    Dataanalyst
3      Analytics
4     Statistics
5            NLP
Name: Domain, dtype: object
```

In [23]:
```
emp['Age'] = emp['Age'].str.extract('(\d+)')
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Affan\AppData\Local\Temp\ipykernel_10220\1884116463.py:1: SyntaxWarning:
invalid escape sequence '\d'
  emp['Age'] = emp['Age'].str.extract('(\d+)')
```

In [24]:
```
emp['Age']
```

Out[24]:
```
0     34
1     45
2    NaN
3    NaN
4     67
5     55
Name: Age, dtype: object
```

In [25]:
```
emp['Location'] = emp['Location'].str.replace(r'\W','',regex=True)
```

In [26]:
```
emp['Location']
```

Out[26]:
```
0       Mumbai
1    Bangalore
2          NaN
3     Hyderbad
4          NaN
5        Delhi
Name: Location, dtype: object
```

In [27]:
```
emp['Salary'] = emp['Salary'].str.replace(r'\W','',regex=True)
```

In [28]:
```
emp['Salary']
```

Out[28]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: object
```

In [29]:
```
emp['Exp'] = emp['Exp'].str.extract('(\d+)')
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Affan\AppData\Local\Temp\ipykernel_10220\3836251810.py:1: SyntaxWarning:
invalid escape sequence '\d'
  emp['Exp'] = emp['Exp'].str.extract('(\d+)')
```

In [30]:
```
emp['Exp']
```

Out[30]:
```
0      2
1      3
2      4
3    NaN
4      5
5     10
Name: Exp, dtype: object
```

In [31]:
```
clean_data = emp.copy()
clean_data #raw to clean data
```

Out[31]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	NaN	NaN	15000	4
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

missing value data treatment

In [33]:
```
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       4 non-null      object
 3   Location  4 non-null      object
 4   Salary    6 non-null      object
 5   Exp       5 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

In [34]:
```
import numpy as np #to take care of multi dim array
```

In [35]:
```
#numpy and pndas is used for data cleaning
```

In [36]:
```
clean_data.head(1)
```

Out[36]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2

In [37]:
```
clean_data
```

Out[37]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	NaN	NaN	15000	4
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [38]:
```
clean_data['Age'] = clean_data['Age'].fillna(np.mean(pd.to_numeric(clean_data['A
```

In [39]:
```
clean_data['Age']
```

Out[39]:
```
0       34
1       45
2    50.25
3    50.25
4       67
5       55
Name: Age, dtype: object
```

In [40]:
```
emp
```

Out[40]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	NaN	NaN	15000	4
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [41]:
```
clean_data
```

Out[41]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50.25	NaN	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [42]:
```
clean_data['Exp'] = clean_data['Exp'].fillna(np.mean(pd.to_numeric(clean_data['E
```

In [45]:
```
clean_data['Exp'] #cleaning Nan value in exp
```

Out[45]:
```
0      2
1      3
2      4
3    4.8
4      5
5     10
Name: Exp, dtype: object
```

In [44]:
```
clean_data
```

Out[44]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50.25	NaN	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	4.8
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

missing value done on dta

In [46]:
```
clean_data['Location']=clean_data['Location'].fillna(clean_data['Location'].mode
```

In [47]:
```
clean_data['Location']
```

Out[47]:
```
0       Mumbai
1    Bangalore
2    Bangalore
3     Hyderbad
4    Bangalore
5        Delhi
Name: Location, dtype: object
```

-----to chnage dtype of attributes---------

In [59]:
```
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      category
 1   Domain    6 non-null      category
 2   Age       6 non-null      int32
 3   Location  6 non-null      category
 4   Salary    6 non-null      int32
 5   Exp       6 non-null      int32
dtypes: category(3), int32(3)
memory usage: 866.0 bytes
```

#dtypes - how to convert Name to category, domain also...age-number,location,cateogry,salary-no.....by defualt python gives objewct as defualt dtype

In [52]:
```
clean_data['Age'] = clean_data['Age'].astype(int)
clean_data['Exp'] = clean_data['Exp'].astype(int)
clean_data['Salary'] = clean_data['Salary'].astype(int)
```

In [53]:
```
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       6 non-null      int32
 3   Location  6 non-null      object
 4   Salary    6 non-null      int32
 5   Exp       6 non-null      int32
dtypes: int32(3), object(3)
memory usage: 348.0+ bytes
```

-----datatype is changed now----

In [56]:
```
clean_data['Name'] = clean_data['Name'].astype('category')
clean_data['Domain'] = clean_data['Domain'].astype('category')
clean_data['Location'] = clean_data['Location'].astype('category')
```

In [57]:
```
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype   
---  ------    --------------  -----   
 0   Name      6 non-null      category
 1   Domain    6 non-null      category
 2   Age       6 non-null      int32   
 3   Location  6 non-null      category
 4   Salary    6 non-null      int32   
 5   Exp       6 non-null      int32   
dtypes: category(3), int32(3)
memory usage: 866.0 bytes
```

WHAT HAVE WE DONE TILL NOW????? ---we have raw data (unclean,filled with noise char(*^%#)--regex,replace,extrac)---- -----numeric(mean) || category(mode) ----change inbuild dtype to respective dtype

In [60]:
```
clean_data
```

Out[60]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [61]:
```
clean_data.to_csv('clean_data.csv')
```

In [62]:
```
import os
os.getcwd()
```

Out[62]:
```
'C:\\Users\\Affan'
```

once data set is cleaned we visualize thenmon-25 deg, tues,35 deg ....(predictive model)

Var identification

In [64]:
```
clean_data.columns
```

Out[64]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [65]:
```
#for visulization we need matplotlib and seaborn
```

In [66]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [67]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [68]:
```
clean_data
```

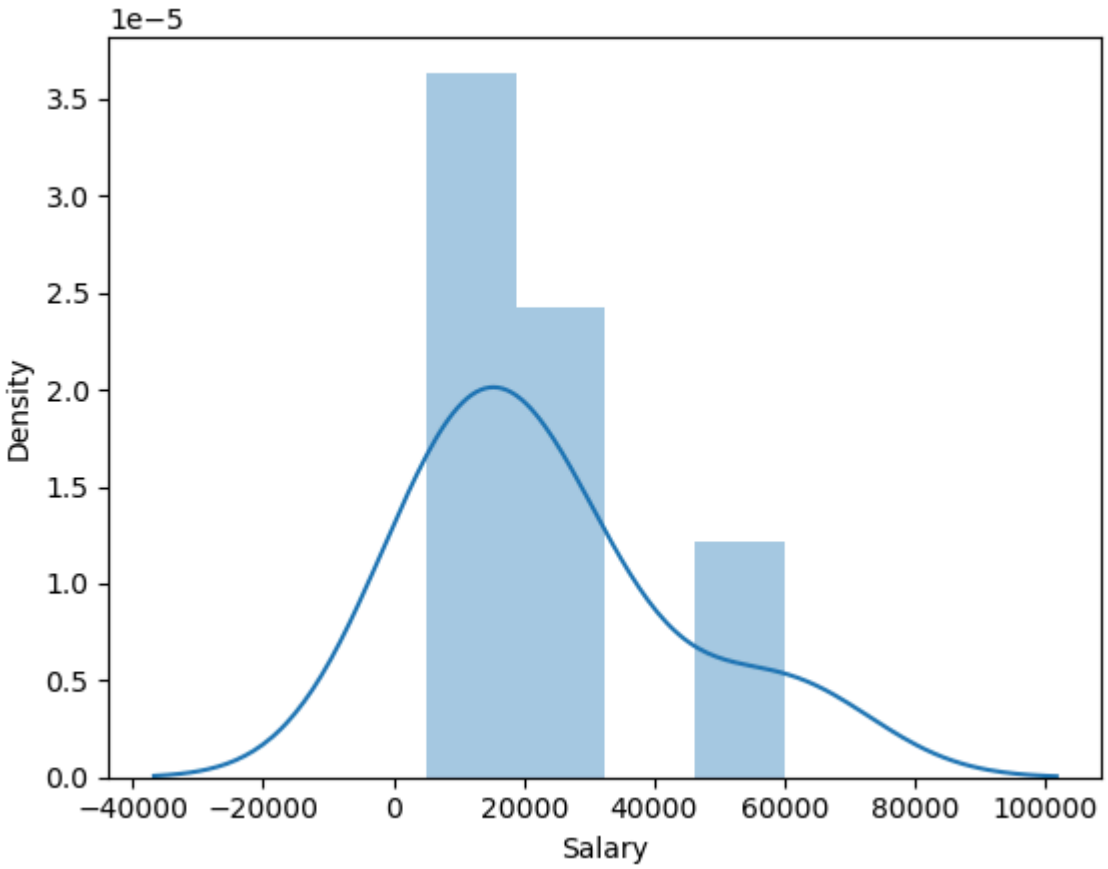
Out[68]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

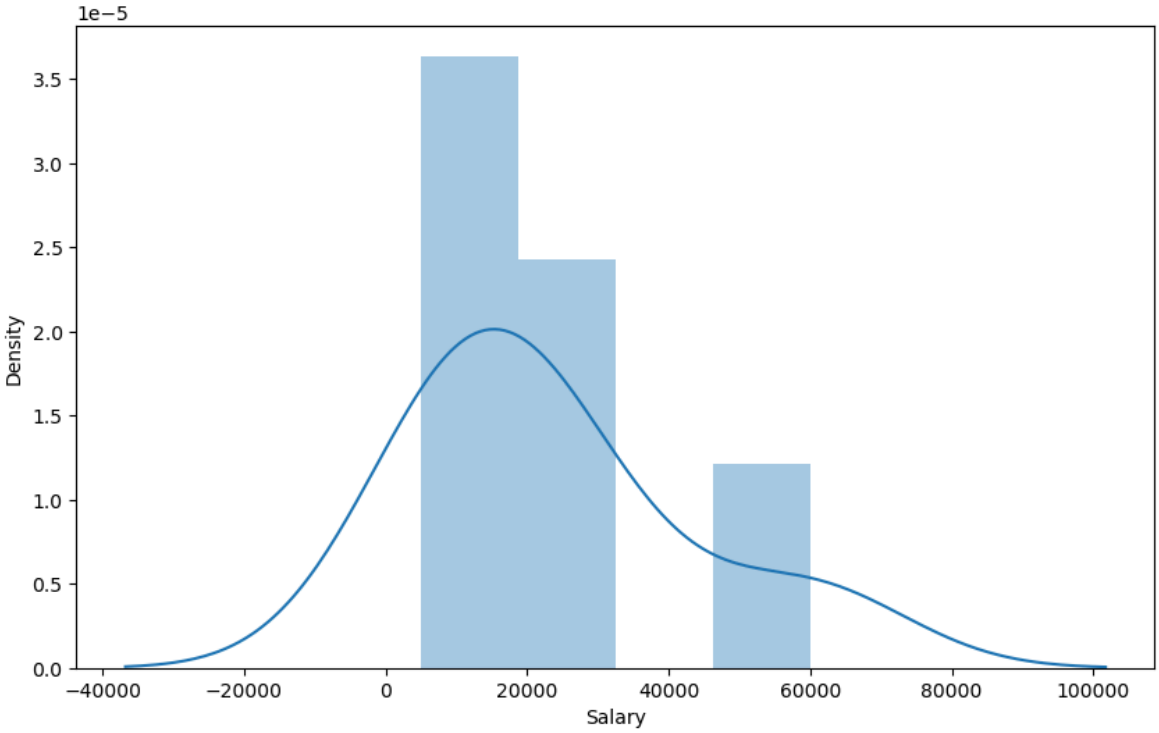
In [69]:
```
clean_data['Salary']
```

Out[69]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: int32
```

In [71]:
```
vis1 = sns.distplot(clean_data['Salary']) #univariate analysis coz only using 1
```

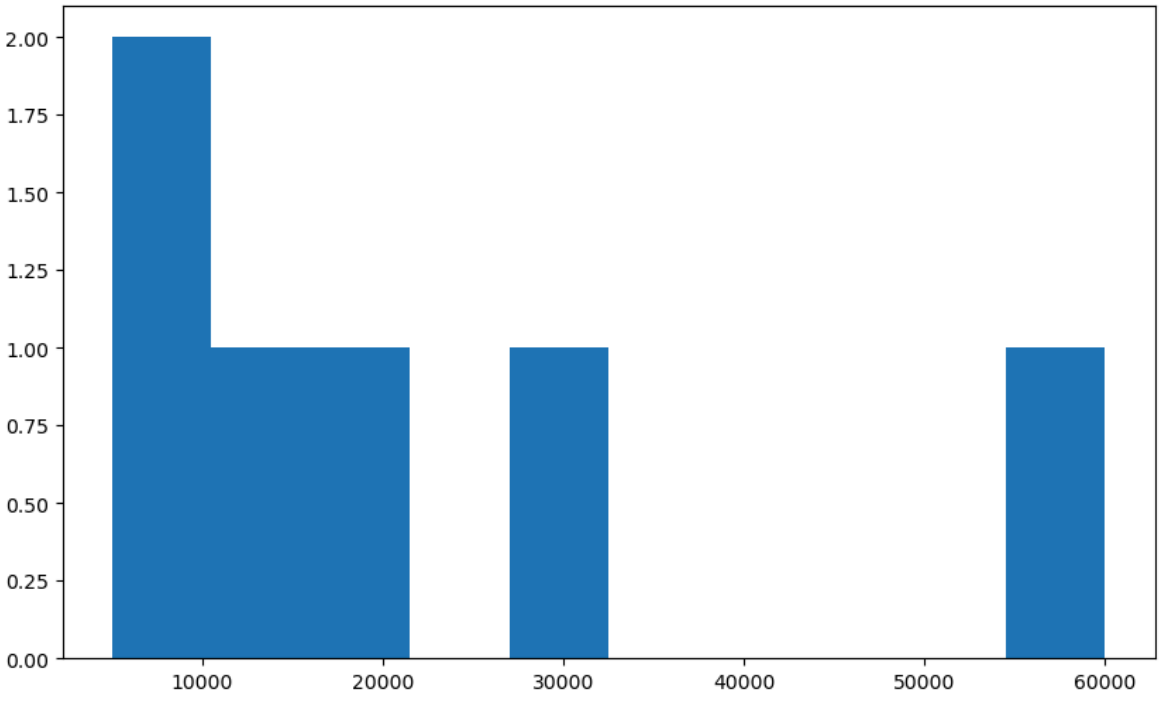


In [74]:
```
plt.rcParams['figure.figsize']=10,6
vis1 = sns.distplot(clean_data['Salary'])
```

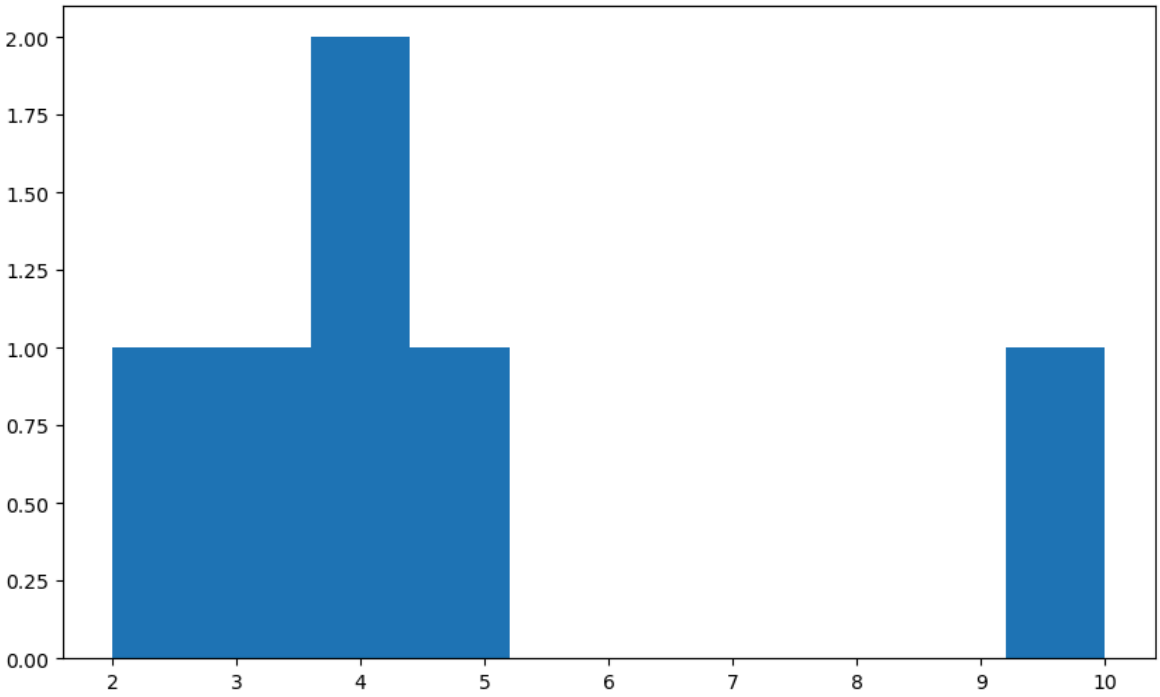


OUTLIER DETECTION --- how to detect outlier by visulization tecnique (dist lm plot)

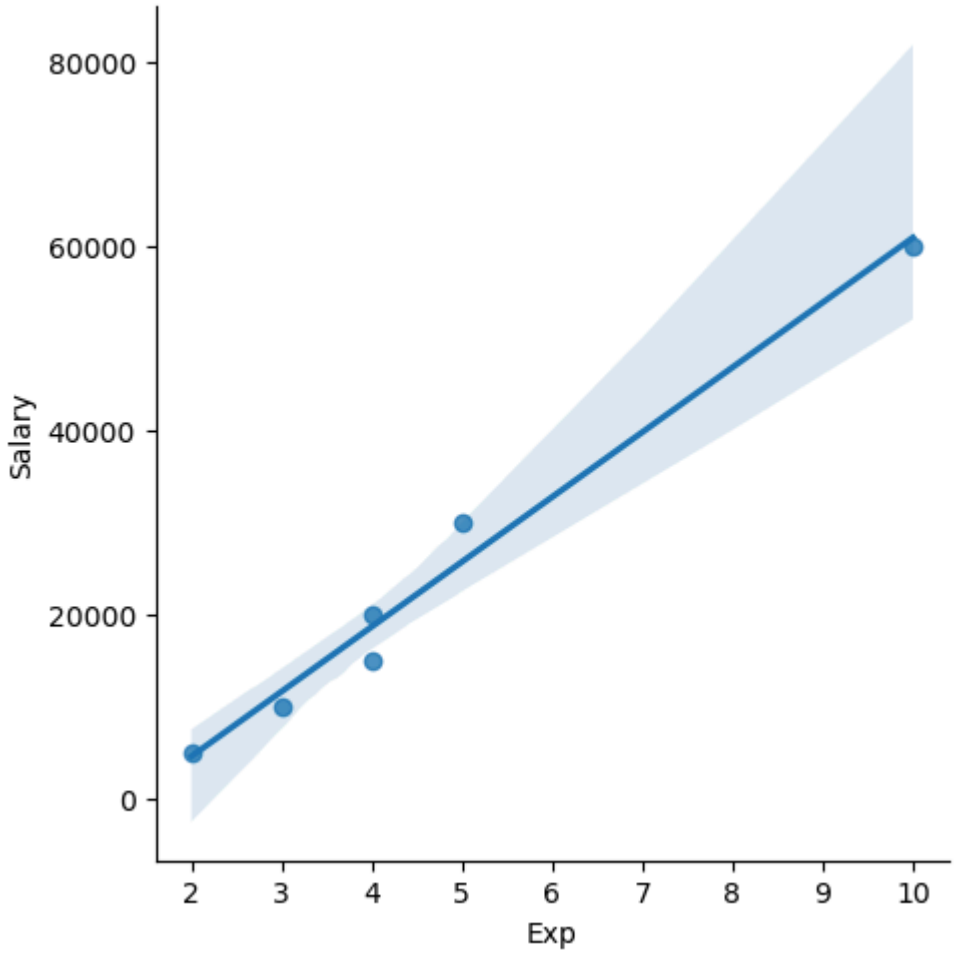
In [77]:
```
vis2 = plt.hist(clean_data['Salary']) #60000 ies an outlier here
```



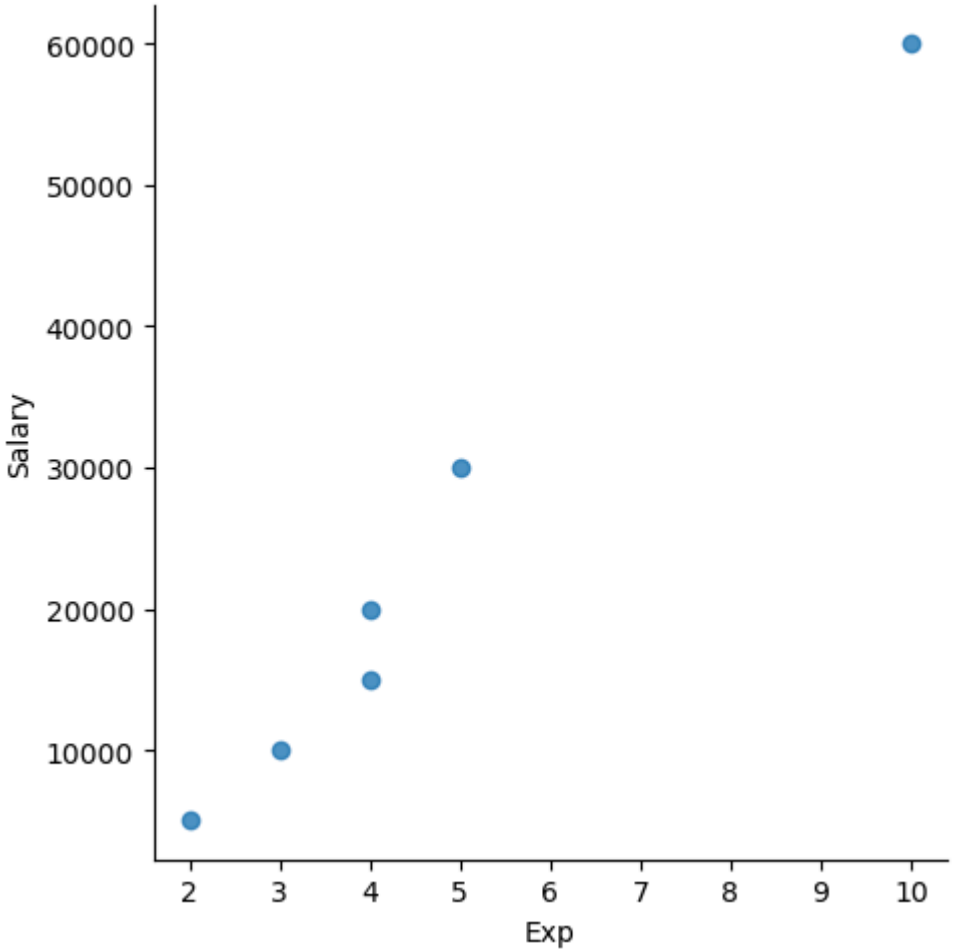
In [78]:
```
vis3 = plt.hist(clean_data['Exp'])
```



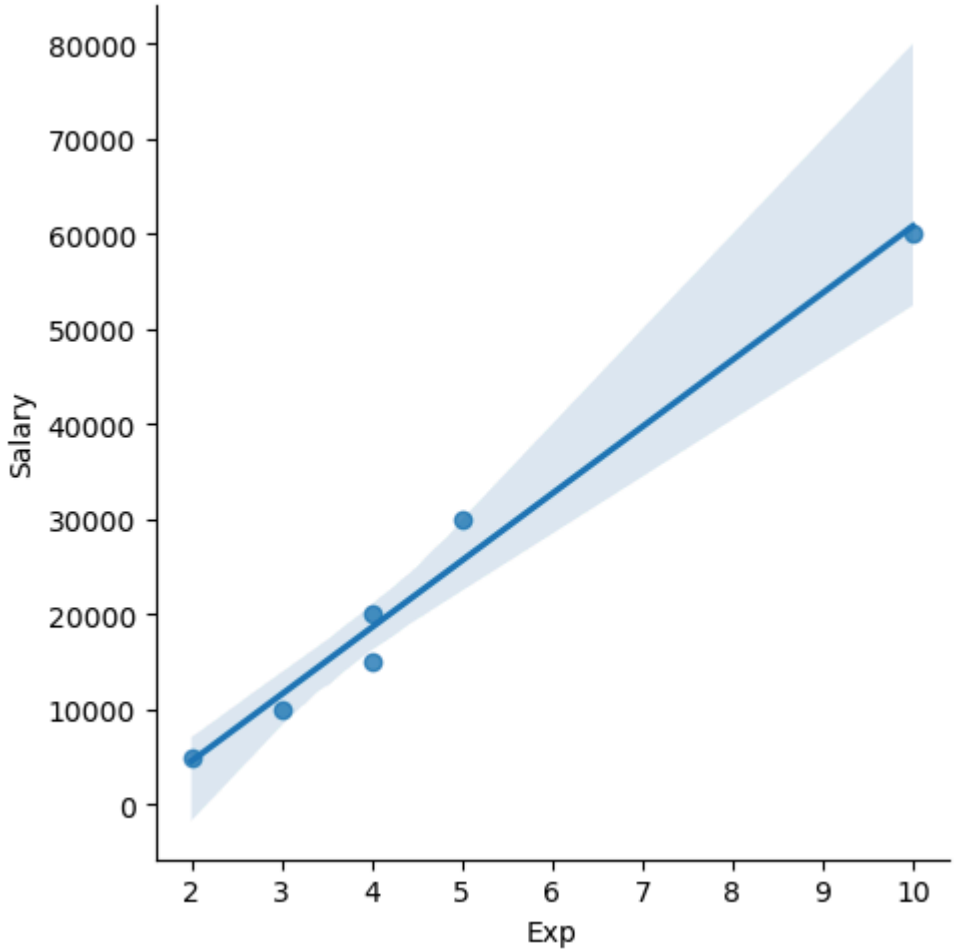
In [79]:
```
vis4 = sns.lmplot(data=clean_data,x='Exp',y='Salary')
```



In [80]:
```
vis5 = sns.lmplot(data=clean_data,x='Exp',y='Salary',fit_reg=False)
```



In [81]:
```
vis4 = sns.lmplot(data=clean_data,x='Exp',y='Salary',fit_reg=True)
```



heatmap can be used to visuliaze more than 2 varafter visuliazing the clean data,Var tranformation and creation---relevant, irrelevant data

In [82]:
```
clean_data
```

Out[82]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [83]:
```
clean_data[:]
```

Out[83]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [84]:
```
clean_data[2:]
```

Out[84]:

	Name	Domain	Age	Location	Salary	Exp
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [86]:
```
clean_data[2:3]
```

Out[86]:

	Name	Domain	Age	Location	Salary	Exp
2	Umar	Dataanalyst	50	Bangalore	15000	4

split the data into idependent var adn dependent var

In [91]:
```
x_iv = clean_data.drop(['Salary'],axis=1) #salary is dependent var so if we drop
```

In [88]:
```
x_iv
```

Out[88]:

	Name	Domain	Age	Location	Exp
0	Mike	Datascience	34	Mumbai	2
1	Teddy	Testing	45	Bangalore	3
2	Umar	Dataanalyst	50	Bangalore	4
3	Jane	Analytics	50	Hyderbad	4
4	Uttam	Statistics	67	Bangalore	5
5	Kim	NLP	55	Delhi	10

In [89]:
```
x_iv.columns
```

Out[89]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Exp'], dtype='object')
```

In [90]:
```
clean_data.columns
```

Out[90]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [92]:
```
y_dv= clean_data.drop(['Name', 'Domain', 'Age', 'Location', 'Exp'],axis=1)
y_dv
```

Out[92]:

	Salary
0	5000
1	10000
2	15000
3	20000
4	30000
5	60000

In [93]:
```
x_iv
```

Out[93]:

	Name	Domain	Age	Location	Exp
0	Mike	Datascience	34	Mumbai	2
1	Teddy	Testing	45	Bangalore	3
2	Umar	Dataanalyst	50	Bangalore	4
3	Jane	Analytics	50	Hyderbad	4
4	Uttam	Statistics	67	Bangalore	5
5	Kim	NLP	55	Delhi	10

In [94]:
```
y_dv
```

Out[94]:

	Salary
0	5000
1	10000
2	15000
3	20000
4	30000
5	60000

In [95]:
```
clean_data
```

Out[95]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [97]:
```
imputation = pd.get_dummies(clean_data,dtype=int)
imputation
```

Out[97]:

	Age	Salary	Exp	Name_Jane	Name_Kim	Name_Mike	Name_Teddy	Name_Umar
0	34	5000	2	0	0	1	0	0
1	45	10000	3	0	0	0	1	0
2	50	15000	4	0	0	0	0	1
3	50	20000	4	1	0	0	0	0
4	67	30000	5	0	0	0	0	0
5	55	60000	10	0	1	0	0	0

EDA Workshop Completed

In [ ]:
```
```