Raw data to clene data Conversion using python EDA

In [1]:
```
import pandas as pd # import the library
```

In [2]:
```
pd.__version__ #check the version
```

Out[2]:
```
'2.2.2'
```

In [3]:
```
emp=pd.read_excel(r'C:\Users\arati\Downloads\Rawdata.xlsx')
```

In [4]:
```
emp
```

Out[4]:

	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 [5]:
```
id(emp) #chech the id
```

Out[5]:
```
2272148549904
```

In [6]:
```
emp.columns #see the columns
```

Out[6]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [7]:
```
emp.shape #gives rows and columns
```

Out[7]:
```
(6, 6)
```

In [8]:
```
emp.head() #see top 5 rows
```

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() # see buttom 5 rows
```

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]:
```
emp.info() #see the information
```

```
<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 [11]:
```
emp
```

Out[11]:

	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 [12]:
```
emp.isnull() #chech whether there is any null value or not
```

Out[12]:

	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 [13]:
```
emp.isnull().sum() #calulate at which column how many null value we have
```

Out[13]:
```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

In [14]:
```
emp.columns #see the columns
```

Out[14]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [15]:
```
emp
```

Out[15]:

	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 CLEANING OR DATA CLEANSING

In [16]:
```
emp['Name'] #to read the name column
```

Out[16]:
```
0      Mike
1    Teddy^
2     Uma#r
3      Jane
4    Uttam*
5       Kim
Name: Name, dtype: object
```

In [17]:
```
#for remove these unwanted symbols
emp['Name'] = emp['Name'].str.replace(r'\W','',regex=True) #non Word Character
```

In [18]:
```
emp['Name'] #Check whether the number attribute change successfully or not
```

Out[18]:
```
0     Mike
1    Teddy
2     Umar
3     Jane
4    Uttam
5      Kim
Name: Name, dtype: object
```

In [19]:
```
#To remove unwanted characters
emp['Domain'] = emp['Domain'].str.replace(r'\W','',regex=True)
```

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

Out[20]:
```
0    Datascience
1        Testing
2    Dataanalyst
3      Analytics
4     Statistics
5            NLP
Name: Domain, dtype: object
```

In [21]:
```
emp #See the data
```

Out[21]:

	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	Umar	Dataanalyst	NaN	NaN	1$5%000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

In [22]:
```
#Check whether there is any unwanted symbols
emp['Age'] = emp['Age'].str.replace(r'\W','',regex=True)
```

In [23]:
```
emp['Age'] #Prints tha Age columns
```

Out[23]:
```
0    34years
1       45yr
2        NaN
3        NaN
4       67yr
5       55yr
Name: Age, dtype: object
```

In [24]:
```
emp['Age']=emp['Age'].str.extract(r'(\d+)') #for extract only numbers from the a
```

In [25]:
```
emp['Age'] #Prints the asg columns
```

Out[25]:
```
0     34
1     45
2    NaN
3    NaN
4     67
5     55
Name: Age, dtype: object
```

In [26]:
```
emp
```

Out[26]:

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

In [27]:
```
emp['Location']= emp['Location'].str.replace(r'\W','',regex=True) #Here we repla
```

In [28]:
```
emp['Location'] #disply the location columns
```

Out[28]:
```
0       Mumbai
1    Bangalore
2          NaN
3     Hyderbad
4          NaN
5        Delhi
Name: Location, dtype: object
```

In [29]:
```
emp
```

Out[29]:

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

In [30]:
```
emp['Salary']=emp['Salary'].str.replace(r'\W','',regex=True) #For replace the un
```

In [31]:
```
emp['Salary'] #prints the salary column
```

Out[31]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: object
```

In [32]:
```
emp
```

Out[32]:

	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> yrs
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5+ year
5	Kim	NLP	55	Delhi	60000	10+

In [33]:
```
emp['Exp']=emp['Exp'].str.extract(r'(\d+)') #For extract the integers
```

In [34]:
```
emp['Exp'] #prints Exp
```

Out[34]:
```
0      2
1      3
2      4
3    NaN
4      5
5     10
Name: Exp, dtype: object
```

In [35]:
```
emp
```

Out[35]:

	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 [36]:
```
clean_data = emp.copy() # Copy the dataset
```

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

Till now we have raw data we use regex to clean the data and removeed all noise char from the dataset

we can also work with sql

Using EDA technique

Missing value Treatment for numerical data

In [38]:
```
clean_data.isnull().sum()
```

Out[38]:
```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

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

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

In [40]:
```
import numpy as np #import numpy
```

In [41]:
```
#For numerical value  we use mean, median and mode strategy
clean_data['Age']= clean_data['Age'].fillna(np.mean(pd.to_numeric(clean_data['Ag
```

In [42]:
```
clean_data['Age'] #Displayes the cleane Age column
```

Out[42]:
```
0       34
1       45
2    50.25
3    50.25
4       67
5       55
Name: Age, dtype: object
```

In [43]:
```
clean_data['Exp']=clean_data['Exp'].fillna(np.mean(pd.to_numeric(clean_data['Exp
```

In [44]:
```
clean_data['Exp'] #prints it
```

Out[44]:
```
0      2
1      3
2      4
3    4.8
4      5
5     10
Name: Exp, dtype: object
```

In [45]:
```
clean_data['Location'].isnull().sum() #here we check rather there is any missing
```

Out[45]:
```
2
```

In [46]:
```
clean_data['Location'] #displayes the location column
```

Out[46]:
```
0       Mumbai
1    Bangalore
2          NaN
3     Hyderbad
4          NaN
5        Delhi
Name: Location, dtype: object
```

In [47]:
```
#for categorical value we use mode strategy or KNN strategy
clean_data['Location'] = clean_data['Location'].fillna(clean_data['Location'].mo
```

In [48]:
```
clean_data['Location'] # prints the location column
```

Out[48]:
```
0       Mumbai
1    Bangalore
2    Bangalore
3     Hyderbad
4    Bangalore
5        Delhi
Name: Location, dtype: object
```

In [49]:
```
clean_data #displayes the clean_data
```

Out[49]:

	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	Bangalore	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	4.8
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [50]:
```
clean_data.info() #see the informations
```

```
<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      object
 3   Location  6 non-null      object
 4   Salary    6 non-null      object
 5   Exp       6 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

In [51]:
```
#For change the type -- numerical
clean_data['Age'] = clean_data['Age'].astype(int) #Chenge flote or string type t
```

In [52]:
```
clean_data['Age']
```

Out[52]:
```
0    34
1    45
2    50
3    50
4    67
5    55
Name: Age, dtype: int32
```

In [53]:
```
clean_data['Salary']=clean_data['Salary'].astype(int) #change the type
```

In [54]:
```
clean_data['Salary']
```

Out[54]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: int32
```

In [55]:
```
clean_data['Exp']=clean_data['Exp'].astype(int) #change the type
```

In [56]:
```
clean_data['Exp']
```

Out[56]:
```
0     2
1     3
2     4
3     4
4     5
5    10
Name: Exp, dtype: int32
```

In [57]:
```
# change the type -- categorical
clean_data['Name'] = clean_data['Name'].astype('category')
clean_data['Domain'] = clean_data['Name'].astype('category')
clean_data['Location'] = clean_data['Name'].astype('category')
```

In [58]:
```
clean_data.info() #see the information
```

```
<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: 882.0 bytes
```

In [59]:
```
#We have to convert this file to csv to extract the dataframe
clean_data.to_csv('clean_data.csv') #change the file to csv
```

In [60]:
```
#get the location
import os
os.getcwd()
```

Out[60]:
```
'C:\\Users\\arati'
```

In [61]:
```
clean_data
```

Out[61]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Mike	34	Mike	5000	2
1	Teddy	Teddy	45	Teddy	10000	3
2	Umar	Umar	50	Umar	15000	4
3	Jane	Jane	50	Jane	20000	4
4	Uttam	Uttam	67	Uttam	30000	5
5	Kim	Kim	55	Kim	60000	10

EDA technique lets Apply

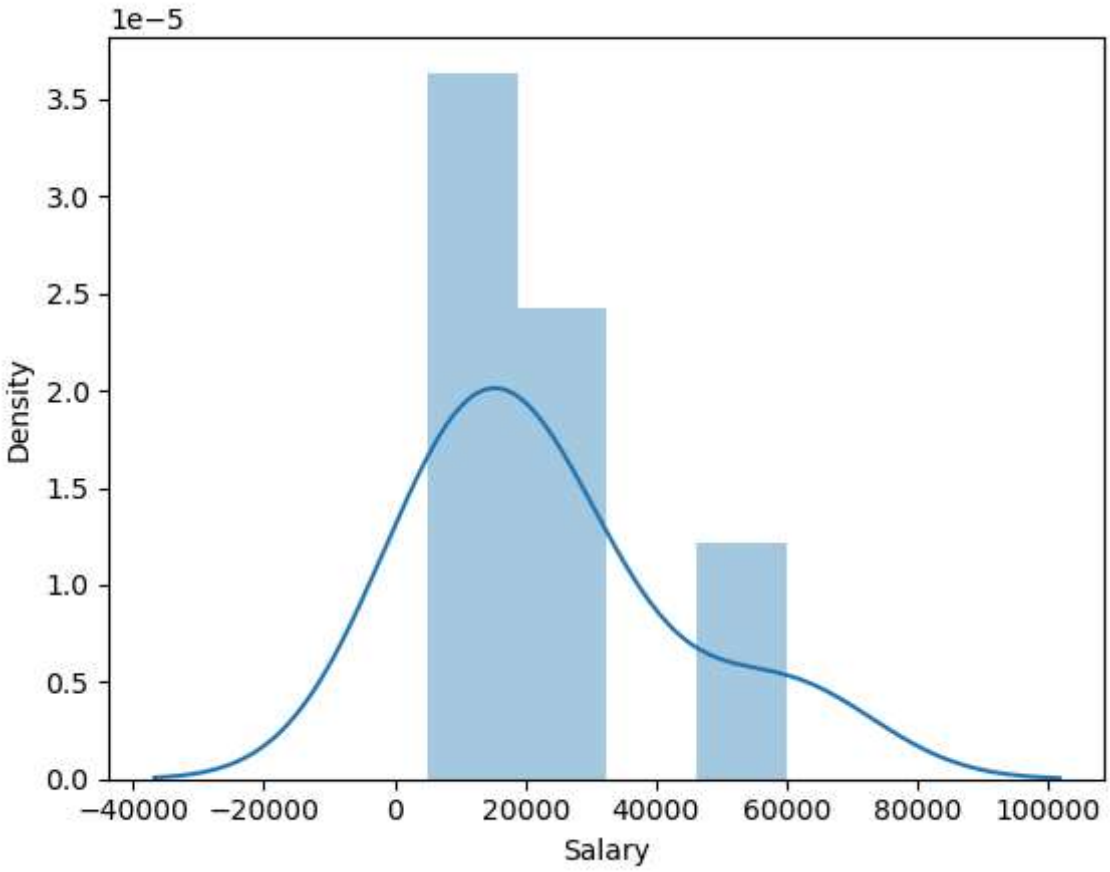
In [62]:
```
import matplotlib.pyplot as plt # For visualization
import seaborn as sns #for advance visualization
```

In [63]:
```
import warnings
warnings.filterwarnings('ignore') #to ignore all warnings
```

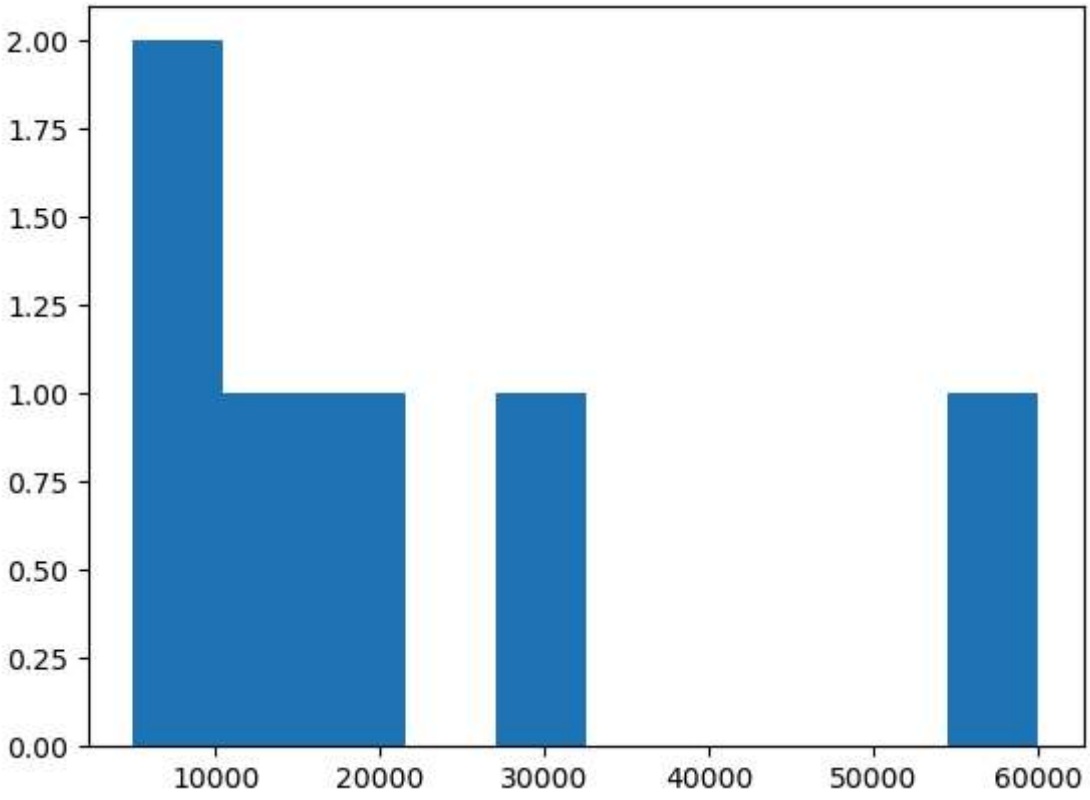
In [64]:
```
clean_data['Salary'] #display the salary column
```

Out[64]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: int32
```

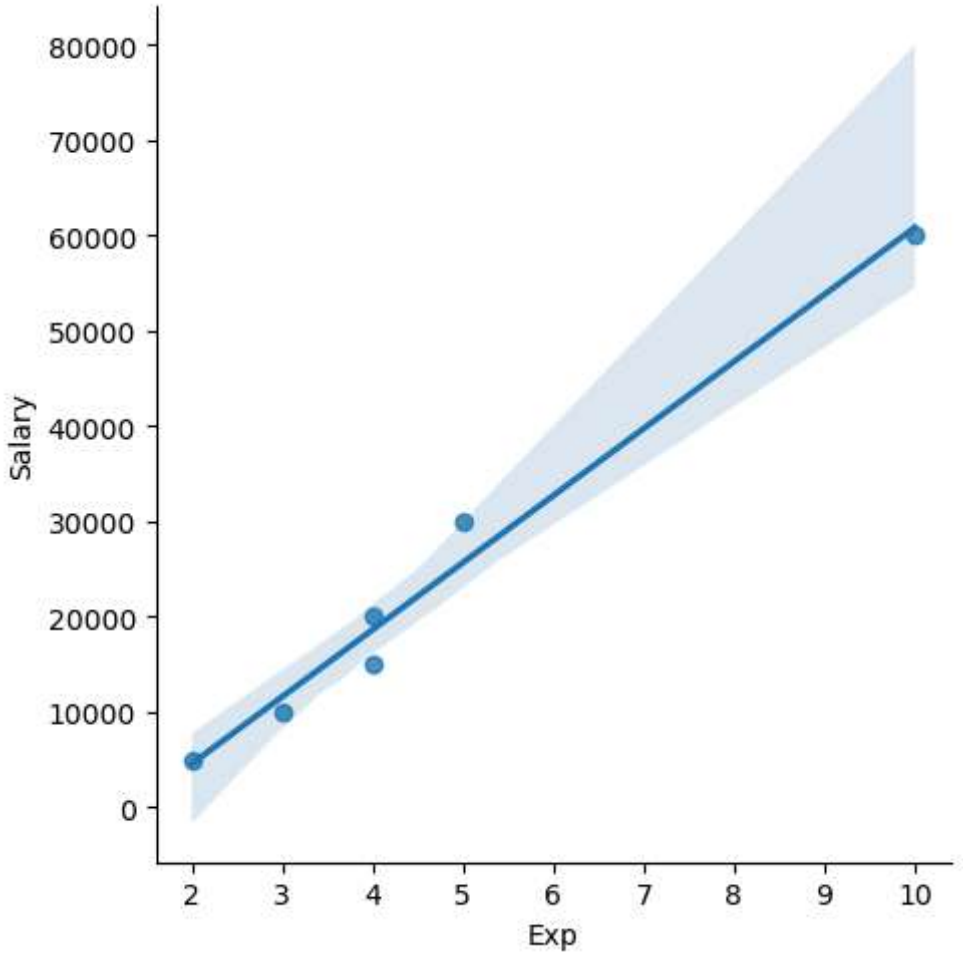
In [65]:
```
#univariate analysis--Plot the graph using one variable
vis1 = sns.distplot(clean_data['Salary'])
```



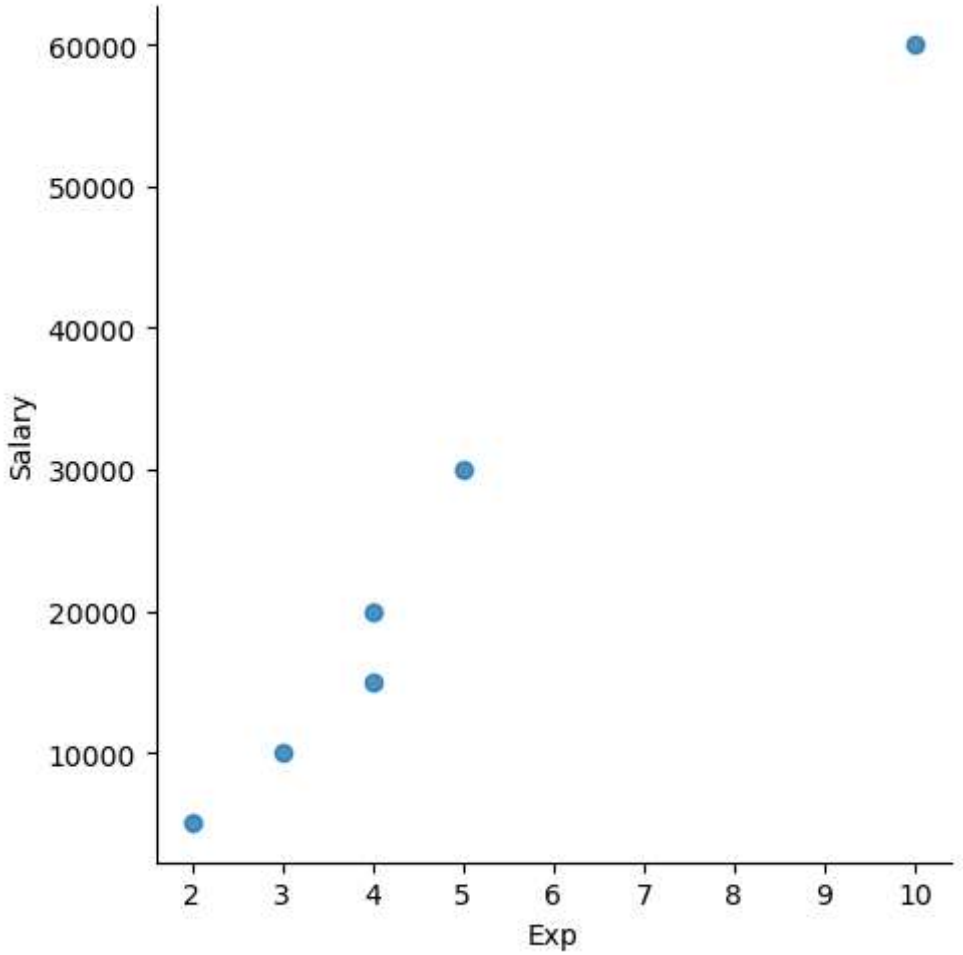
In [66]:
```
vis2 = plt.hist(clean_data['Salary'])
```



In [67]:
```
#Bivariate analysis
vis4 = sns.lmplot(clean_data,x ='Exp',y='Salary')
```



In [68]:
```
vis5 = sns.lmplot(data=clean_data,x = 'Exp', y='Salary', fit_reg = False) #remov
```



In [69]:
```
clean_data[:] #Slicing
```

Out[69]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Mike	34	Mike	5000	2
1	Teddy	Teddy	45	Teddy	10000	3
2	Umar	Umar	50	Umar	15000	4
3	Jane	Jane	50	Jane	20000	4
4	Uttam	Uttam	67	Uttam	30000	5
5	Kim	Kim	55	Kim	60000	10

In [70]:
```
clean_data[0:6:2]
```

Out[70]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Mike	34	Mike	5000	2
2	Umar	Umar	50	Umar	15000	4
4	Uttam	Uttam	67	Uttam	30000	5

In [71]:
```
clean_data[::-1]
```

Out[71]:

	Name	Domain	Age	Location	Salary	Exp
5	Kim	Kim	55	Kim	60000	10
4	Uttam	Uttam	67	Uttam	30000	5
3	Jane	Jane	50	Jane	20000	4
2	Umar	Umar	50	Umar	15000	4
1	Teddy	Teddy	45	Teddy	10000	3
0	Mike	Mike	34	Mike	5000	2

In [72]:
```
#Variable identification -- 2 type-- Dependent & Independent
clean_data.columns #see the columns
```

Out[72]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [73]:
```
X_iv = clean_data[['Name', 'Domain', 'Age', 'Location', 'Exp']] #Displayes the i
```

In [74]:
```
X_iv
```

Out[74]:

	Name	Domain	Age	Location	Exp
0	Mike	Mike	34	Mike	2
1	Teddy	Teddy	45	Teddy	3
2	Umar	Umar	50	Umar	4
3	Jane	Jane	50	Jane	4
4	Uttam	Uttam	67	Uttam	5
5	Kim	Kim	55	Kim	10

In [75]:
```
y_dv = clean_data[['Salary']] #dependent variables
```

In [76]:
```
y_dv
```

Out[76]:

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

In [77]:
```
emp
```

Out[77]:

	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 [78]:
```
clean_data
```

Out[78]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Mike	34	Mike	5000	2
1	Teddy	Teddy	45	Teddy	10000	3
2	Umar	Umar	50	Umar	15000	4
3	Jane	Jane	50	Jane	20000	4
4	Uttam	Uttam	67	Uttam	30000	5
5	Kim	Kim	55	Kim	60000	10

In [79]:
```
X_iv
```

Out[79]:

	Name	Domain	Age	Location	Exp
0	Mike	Mike	34	Mike	2
1	Teddy	Teddy	45	Teddy	3
2	Umar	Umar	50	Umar	4
3	Jane	Jane	50	Jane	4
4	Uttam	Uttam	67	Uttam	5
5	Kim	Kim	55	Kim	10

In [80]:
```
y_dv
```

Out[80]:

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

In [81]:
```
clean_data
```

Out[81]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Mike	34	Mike	5000	2
1	Teddy	Teddy	45	Teddy	10000	3
2	Umar	Umar	50	Umar	15000	4
3	Jane	Jane	50	Jane	20000	4
4	Uttam	Uttam	67	Uttam	30000	5
5	Kim	Kim	55	Kim	60000	10

In [82]:
```
#Variable creation and variable transformation
imputation = pd.get_dummies(clean_data) #Here we create the dummy variable for t
```

In [83]:
```
imputation  #display it
```

Out[83]:

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

6 rows × 21 columns

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

Out[84]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Mike	34	Mike	5000	2
1	Teddy	Teddy	45	Teddy	10000	3
2	Umar	Umar	50	Umar	15000	4
3	Jane	Jane	50	Jane	20000	4
4	Uttam	Uttam	67	Uttam	30000	5
5	Kim	Kim	55	Kim	60000	10

In [85]:
```
imputation #get the output as boolen type--true / false
```

Out[85]:

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

6 rows × 21 columns

In [86]:
```
#lets convert it as integer type
imputation=imputation.astype(int)
```

In [87]:
```
imputation
```

Out[87]:

	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

6 rows × 21 columns

In [88]:
```
clean_data
```

Out[88]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Mike	34	Mike	5000	2
1	Teddy	Teddy	45	Teddy	10000	3
2	Umar	Umar	50	Umar	15000	4
3	Jane	Jane	50	Jane	20000	4
4	Uttam	Uttam	67	Uttam	30000	5
5	Kim	Kim	55	Kim	60000	10

We have completed EDA part successfully

data set load

- remove numwanted this from data from every column
- treat missing value mean ana mode strategy
- outlier dectection
- univariate and bivariate analysis
- variable identification
- dummy vari

In [ ]: