Employee Profile Data Analysis

This project focuses on analyzing employee profile data using Python's data manipulation and visualization libraries.

The dataset includes attributes such as Name, Domain, Age, Location, Salary, and Experience.

The primary goal is to clean, transform, and visualize the data to identify patterns and ensure usability for further analysis.

In [1]:
```
import numpy as np
import pandas as pd                    # importing libraries
```

In [2]:
```
emp = pd.read_excel(r"C:\Users\ratho\.ipynb_checkpoints\DATA\Rawdata.xlsx")
emp

# reading the raw data
# emp is identifire , we create to store data
```

Out[2]:

	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 [3]:
```
emp.shape                              # check dimention of data
```

Out[3]:
```
(6, 6)
```

In [4]:
```
len(emp)                               # check length of data
```

Out[4]:
```
6
```

In [5]:
```
emp.columns                            # fetching column names
```

Out[5]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [6]:
```
emp.info()                             # get informaation of data
```

```
<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: 416.0+ bytes
```

In [7]:
```
emp['Name']                    # fetch name column
```

Out[7]:
```
0      Mike
1    Teddy^
2     Uma#r
3      Jane
4    Uttam*
5       Kim
Name: Name, dtype: object
```

In [8]:
```
emp['Domain']                  # fetch domain column
```

Out[8]:
```
0     Datascience#$
1           Testing
2    Dataanalyst^^#
3       Ana^^lytics
4        Statistics
5               NLP
Name: Domain, dtype: object
```

In [9]:
```
emp['Age']                    # fetch Age column
```

Out[9]:
```
0    34 years
1      45' yr
2         NaN
3         NaN
4       67-yr
5        55yr
Name: Age, dtype: object
```

In [10]:
```
emp['Location']            # fetch Location column
```

Out[10]:
```
0       Mumbai
1    Bangalore
2          NaN
3     Hyderbad
4          NaN
5        Delhi
Name: Location, dtype: object
```

In [11]:
```
emp['Salary']               # fetch Salary column
```

Out[11]:
```
0     5^00#0
1    10%%000
2    1$5%000
3     2000^0
4     30000-
5    6000^$0
Name: Salary, dtype: object
```

In [12]:
```
emp['Exp']                    # fetch experience column
```

Out[12]:
```
0         2+
1         <3
2     4> yrs
3        NaN
4    5+ year
5        10+
Name: Exp, dtype: object
```

In [13]:
```
emp[['Name','Domain']]          # fetch Name and Domain column both
```

Out[13]:

	Name	Domain
0	Mike	Datascience#$
1	Teddy^	Testing
2	Uma#r	Dataanalyst^^#
3	Jane	Ana^^lytics
4	Uttam*	Statistics
5	Kim	NLP

In [14]:
```
emp[['Name','Domain','Age','Location']]     # fetch multiple column
```

Out[14]:

	Name	Domain	Age	Location
0	Mike	Datascience#$	34 years	Mumbai
1	Teddy^	Testing	45' yr	Bangalore
2	Uma#r	Dataanalyst^^#	NaN	NaN
3	Jane	Ana^^lytics	NaN	Hyderbad
4	Uttam*	Statistics	67-yr	NaN
5	Kim	NLP	55yr	Delhi

In [15]:
```
emp[['Name','Domain','Age','Location','Salary','Exp']]
```

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 cleansing

In [16]:
```
emp['Name']    # fetch name column ; hter we can see so many special char in names
```

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

In [17]:
```
emp['Name']= emp['Name'].str.replace(r'\W','')

# used regex xharacter to remove all special characters form string
```

```
C:\Users\ratho\AppData\Local\Temp\ipykernel_9280\1423654259.py:1: FutureWarning: The
default value of regex will change from True to False in a future version.
  emp['Name']= emp['Name'].str.replace(r'\W','')                    # used regex xhara
cter to remove all special characters form string
```

In [18]:
```
emp['Name']                    # column 'Name' is cleaned
```

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

In [19]:
```
emp['Domain']       # fetch domain column , here is also so many special char cpresent
```

Out[19]:
```
0     Datascience#$
1           Testing
2    Dataanalyst^^#
3       Ana^^lytics
4        Statistics
5               NLP
Name: Domain, dtype: object
```

In [20]:
```
emp['Domain']= emp['Domain'].str.replace(r'\W','')

# cleaning domain column by using regex wild character'\W'
```

```
C:\Users\ratho\AppData\Local\Temp\ipykernel_9280\2852824474.py:1: FutureWarning: The
default value of regex will change from True to False in a future version.
  emp['Domain']= emp['Domain'].str.replace(r'\W','')           # cleaning domain col
umn by using regex wild character'\W'
```

In [21]:
```
emp['Domain']           # domain column is cleaned
```

Out[21]:
```
0    Datascience
1        Testing
2    Dataanalyst
3      Analytics
4     Statistics
5            NLP
Name: Domain, dtype: object
```

In [22]:
```
emp['Age']              # fetch age column
```

Out[22]:
```
0    34 years
1      45' yr
2         NaN
3         NaN
4       67-yr
5        55yr
Name: Age, dtype: object
```

In [23]:
```
emp['Age'] = emp['Age'].str.replace(r'\W','')


# first we remove special char form it then extract no
# you can directly extract no also
```

```
C:\Users\ratho\AppData\Local\Temp\ipykernel_9280\3358378917.py:1: FutureWarning: The
default value of regex will change from True to False in a future version.
  emp['Age'] = emp['Age'].str.replace(r'\W','')
```

In [24]:
```
emp['Age']        #  using regex wild character  removed all special charcter
```

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

In [25]:
```
emp['Age'] = emp['Age'].str.extract('(\d+)')
emp

# extract only numbers from the age columns
```

Out[25]:

	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 [26]:
```
emp['Location']     # its already cleaned column
```

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','')

# remove all special character, using regex wil char
```

```
C:\Users\ratho\AppData\Local\Temp\ipykernel_9280\2371757234.py:1: FutureWarning: The
default value of regex will change from True to False in a future version.
  emp['Salary'] = emp['Salary'].str.replace(r'\W','')                    # remove all
special character, using regex wil char
```

In [28]:
```
emp                          # call emp data
```

Out[28]:

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

# extracted only no form the orignal column EXp
```

Out[29]:

	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 [30]:
```
clean_data = emp.copy()
clean_data

# we get copy of partial clean data
```

Out[30]:

	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 Treatment

In [31]:
```
clean_data.info()                    # fetching 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: 416.0+ bytes
```

In [32]:
```
clean_data.isnull()              # checking null values are present or not
```

Out[32]:

	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

First clear all the numerical data

In [33]:
```
clean_data.isnull().sum()           # fetching total null values present
```

Out[33]:
```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

In [34]:
```
clean_data['Age']                    # Age column has null values
```

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

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

# here we are filling the null values in Age column ,with mean of the age column
```

In [39]:
```
clean_data['Age']      # nun values are replaced by mean value in Age column
```

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

In [36]:
```
clean_data['Exp']             # Exp caolumn has null values
```

Out[36]:
```
0      2
1      3
2      4
3    NaN
4      5
5     10
Name: Exp, dtype: object
```

In [37]:
```
clean_data['Exp'] = clean_data['Exp'].fillna(np.mean(pd.to_numeric(clean_data['Exp'])))
clean_data['Exp']

# here we replaced nun vlaues in exp column with mean of exp column
```

Out[37]:
```
0      2
1      3
2      4
3    4.8
4      5
5     10
Name: Exp, dtype: object
```

In [40]:
```
clean_data

# calling clean_data  to see all numerical columns are cleaned
```

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	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

Now clear categorical data

In [41]:
```
clean_data                    # calling 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	4.8
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [44]:
```
clean_data['Location']       # fetch location column to see null values
```

Out[44]:
```
0       Mumbai
1    Bangalore
2          NaN
3     Hyderbad
4          NaN
5        Delhi
Name: Location, dtype: object
```

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

# we use mod because it is categorical column
```

Out[46]:

	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	Hyderbad	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

In [47]:
```
clean_data['Location'] = clean_data['Location'].fillna(clean_data['Location'].mode()[0
clean_data

# we use mod because it is categorical column
```

Out[47]:

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

Changing Dataypes

In [48]:
```
clean_data                    # calling clean_data
```

Out[48]:

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

In [49]:
```
clean_data.info()             # checking data types of all column
```

```
<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: 416.0+ bytes
```

name to category # domain to category # Age to number # location to category # salary to number # exp to number # by default python gives object # lets convert object to cat and int

Converting numerial data to int data type

In [50]:
```
# chnging the data type of Age column
clean_data['Age'] = clean_data['Age'].astype(int)

# changing the data types of Salary column
clean_data['Salary'] = clean_data['Salary'].astype(int)

# changing the data type of Exp column
clean_data['Exp'] = clean_data['Exp'].astype(int)
```

In [51]:
```
clean_data['Age']              # data type changed
```

Out[51]:
```
0    34
1    45
2    50
3    50
4    67
5    55
Name: Age, dtype: int32
```

In [52]:
```
clean_data['Salary']            # data type changed
```

Out[52]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: int32
```

In [53]:
```
clean_data['Exp']               # data type changed
```

Out[53]:
```
0     2
1     3
2     4
3     4
4     5
5    10
Name: Exp, dtype: int32
```

In [54]:
```
clean_data.info()           # getting info about the data set, check data types
```

```
<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: 344.0+ bytes
```

Convert catageorical data into categoricl datatype

In [56]:
```
# Changing data type of Name column to category
clean_data['Name'] = clean_data['Name'].astype('category')

# changing data type of Domain column to category
clean_data['Domain'] = clean_data['Domain'].astype('category')

# changing data type of Location column to category
clean_data['Location'] = clean_data['Location'].astype('category')
```

In [57]:
```
clean_data['Name']                # data type changed
```

Out[57]:
```
0     Mike
1    Teddy
2     Umar
3     Jane
4    Uttam
5      Kim
Name: Name, dtype: category
Categories (6, object): ['Jane', 'Kim', 'Mike', 'Teddy', 'Umar', 'Uttam']
```

In [58]:
```
clean_data['Domain']                # data type changed
```

Out[58]:
```
0    Datascience
1        Testing
2    Dataanalyst
3      Analytics
4     Statistics
5            NLP
Name: Domain, dtype: category
Categories (6, object): ['Analytics', 'Dataanalyst', 'Datascience', 'NLP', 'Statistic
s', 'Testing']
```

In [59]:
```
clean_data['Location']           # data type changed
```

Out[59]:
```
0       Mumbai
1    Bangalore
2     Hyderbad
3     Hyderbad
4     Hyderbad
5        Delhi
Name: Location, dtype: category
Categories (4, object): ['Bangalore', 'Delhi', 'Hyderbad', 'Mumbai']
```

In [60]:
```
clean_data.info()      # grting info about clean data and checking data type
```

```
<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: 862.0 bytes
```

In [61]:
```
clean_data.to_csv('EDA1_clean_data.csv')       # saving the cleaned data
```

In [63]:
```
import os              # fetching the location where the cleaned data is saved
os.getcwd()
```

Out[63]:
```
'C:\\Users\\ratho\\.ipynb_checkpoints'
```

In [65]:
```
clean_data.columns     # check columns
```

Out[65]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

In [96]:
```
clean_data            # calling clean data
```

Out[96]:

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

Data visualization

In [66]:
```
# for visualizarion
import matplotlib.pyplot as plt

# for advance vizulization
import seaborn as sns
```

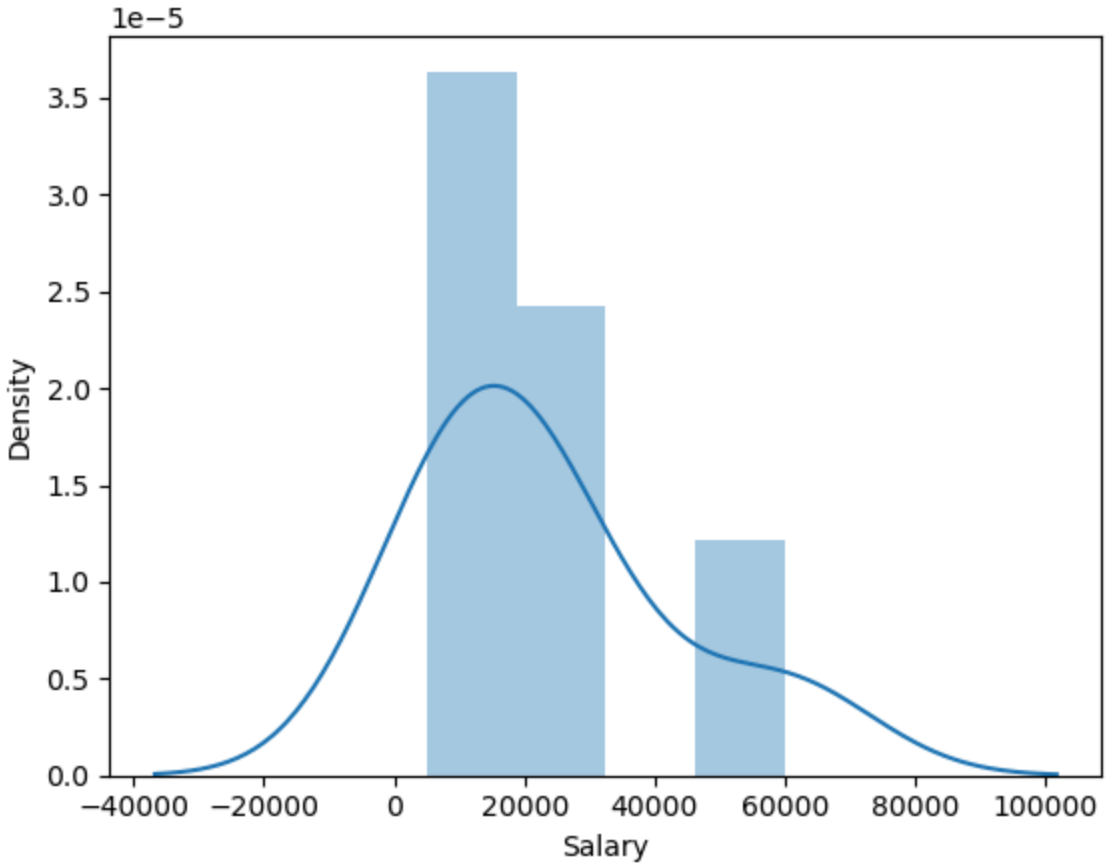
In [67]:
```
import warnings
warnings.filterwarnings('ignore')

# as system gets updated automaticaly , there may be possibility
```

```
# that it will generate error or warning sometime, to avoid those warnings,
# to avoid those warnigs , we are importing the library here
```

In [68]:
```
clean_data['Salary']      # call salary column
```

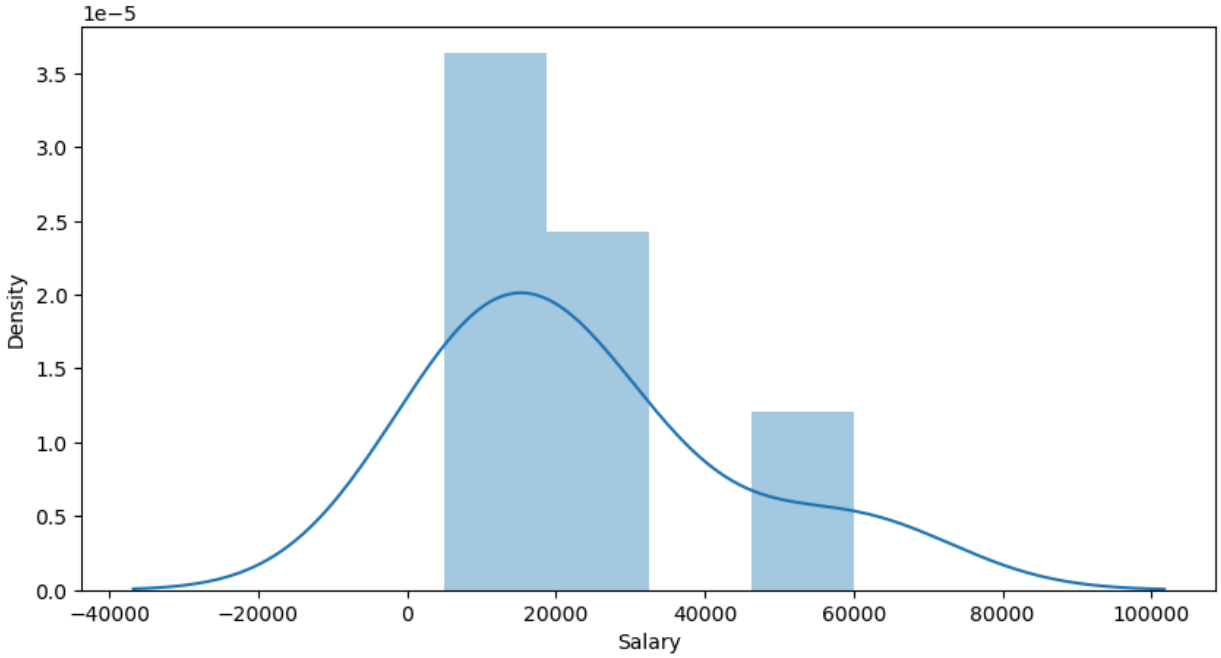
Out[68]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: int32
```

In [69]:
```
viz1 = sns.distplot(clean_data['Salary'])      # univerite analysis
```

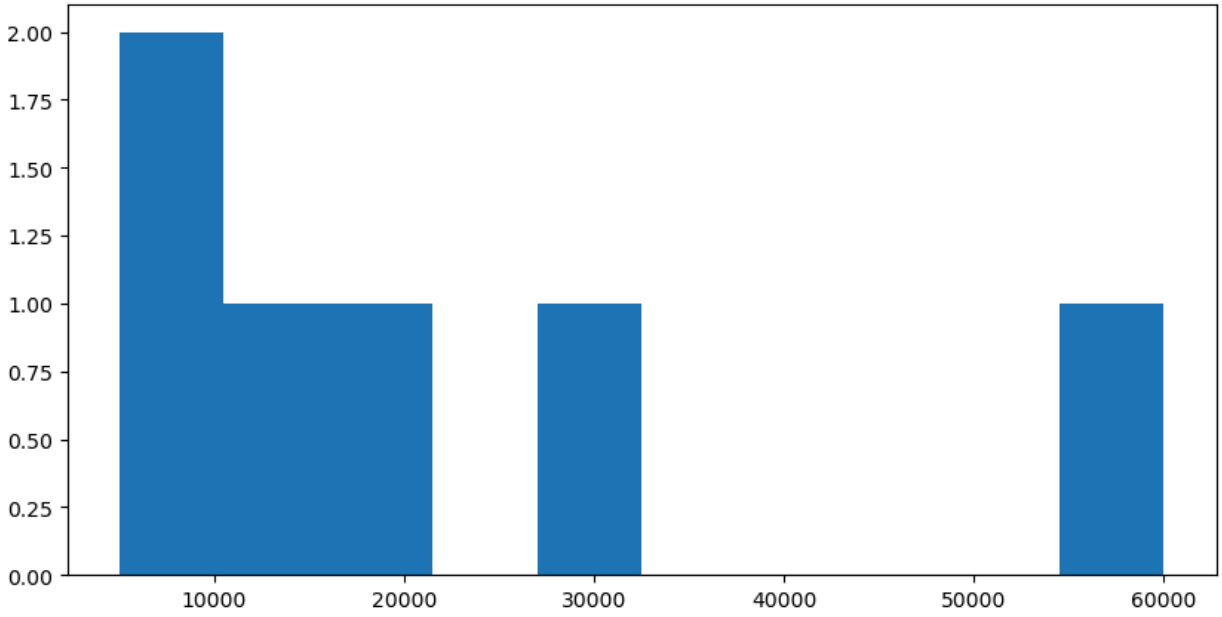


In [72]:
```
plt.rcParams['figure.figsize'] = 10 ,5     # changing size of visualization graph
```

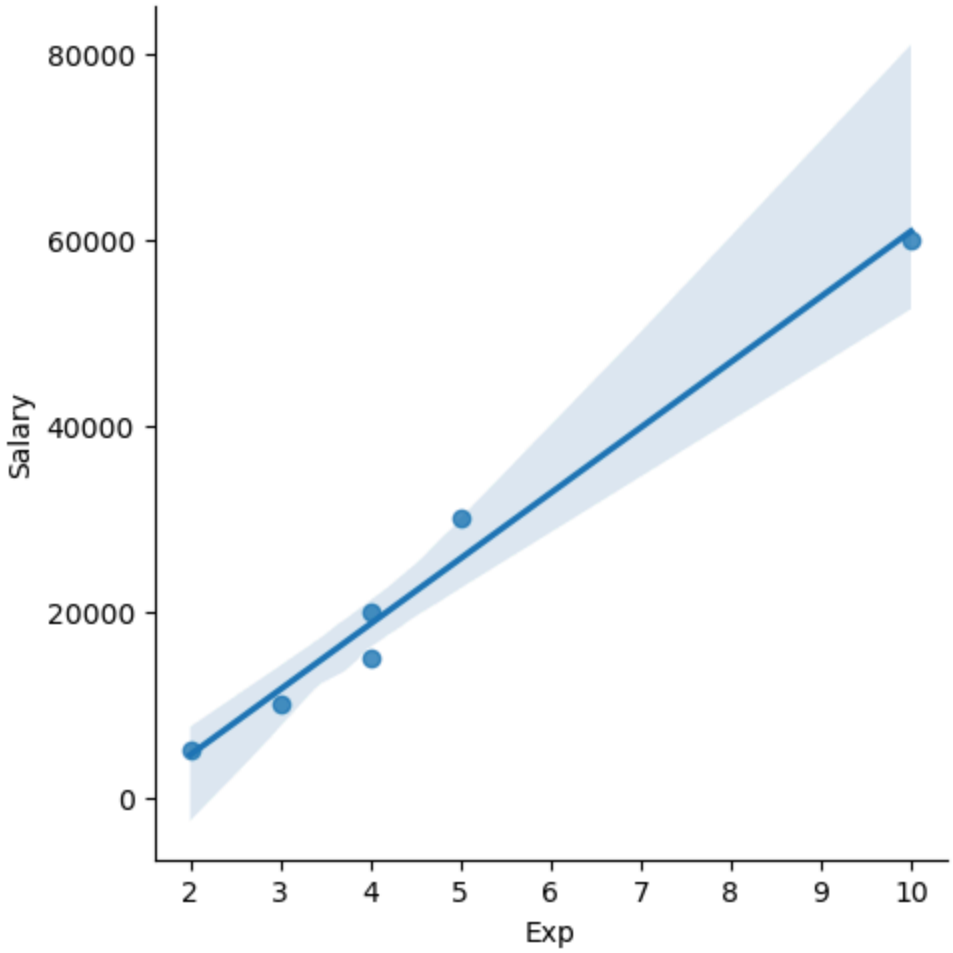
In [73]:
```
viz1 = sns.distplot(clean_data['Salary'])        # size changed
```



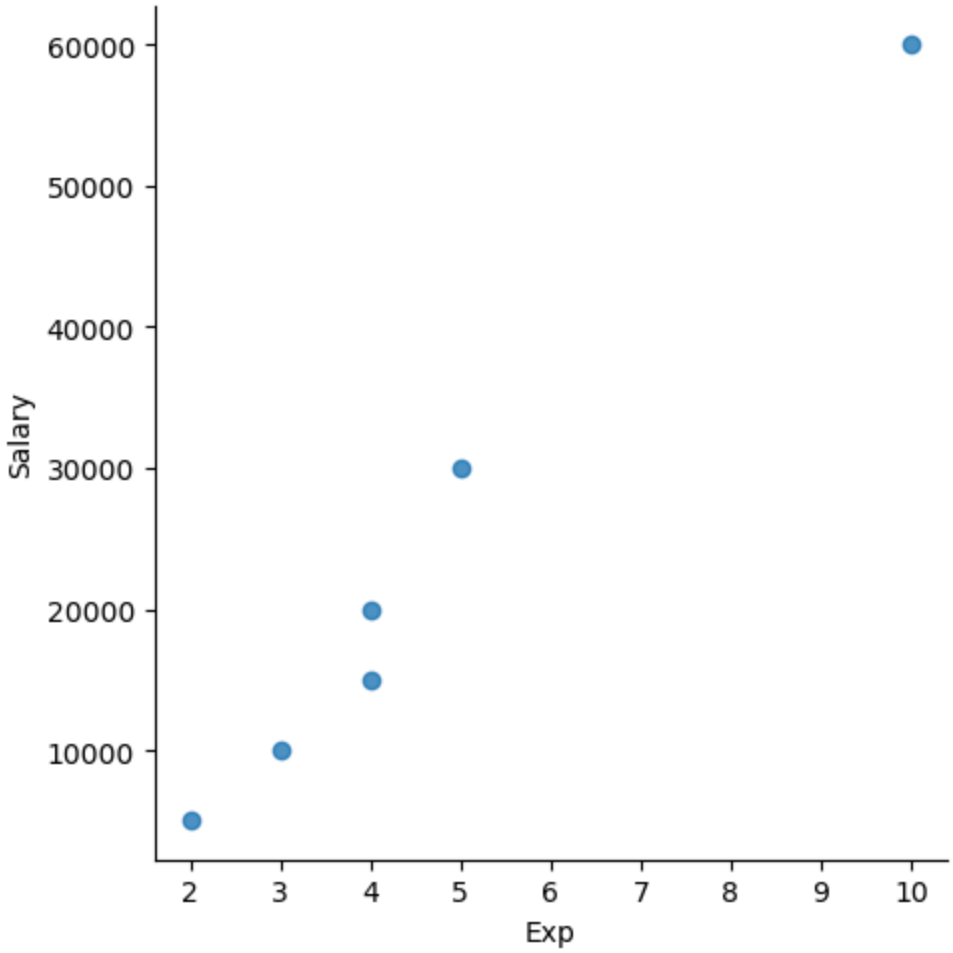
In [74]:
```
viz2 = plt.hist(clean_data['Salary'])       # Outlier detection using 1 variable
```



In [76]:
```
viz4 = sns.lmplot(data = clean_data, x = 'Exp',y = 'Salary')        # bivariate analys
```



In [77]:
```
viz4 = sns.lmplot(data = clean_data, x = 'Exp',y = 'Salary',fit_reg = False)

# without regression line
```



In [78]:
```
clean_data[:2]                    # slicing to fetch 2st 2 row
```

Out[78]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3

In [79]:
```
clean_data[2:]                    # sliced form 2 row till last
```

Out[79]:

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

In [80]:
```
clean_data[0:1]                   # fetch 0th row
```

Out[80]:

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

slplit the data set into dependent and independant n variable

In [82]:
```
clean_data                          # call 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	Hyderbad	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Hyderbad	30000	5
5	Kim	NLP	55	Delhi	60000	10

In [85]:
```
x_iv = clean_data.drop(['Salary'],axis = 1)
x_iv

#x_iv means x indepenedt vsriable,
# we want all indepent var form data , except salary
```

Out[85]:

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

In [87]:
```
x_iv.columns                # getting column names fom x_iv
```

Out[87]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Exp'], dtype='object')
```

In [89]:
```
y_dv =clean_data.drop(['Name', 'Domain', 'Age', 'Location', 'Exp'],axis = 1)
y_dv

# y_dv = y_dependent var
#we want dependent varible column, thats why we are droping remaining column
```

Out[89]:

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

In [90]:
```
clean_data                    # call original clean data
```

Out[90]:

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

In [91]:
```
x_iv                          # independent variable data
```

Out[91]:

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

In [93]:
```
y_dv                            # dependent variable data
```

Out[93]:

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

Imputatioin

In [95]:
```
imputation = pd.get_dummies(clean_data)    # final imputation
imputation
```

Out[95]:

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