```
import pandas as pd
```

```
pd.__version__
```

```
'2.2.2'
```

```
pip install --upgrade openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\bhavesh govind
bhork\anaconda3\lib\site-packages (3.1.5)
Requirement already satisfied: et-xmlfile in c:\users\bhavesh govind
bhork\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

```
emp = pd.read_excel(r'C:\Users\Bhavesh Govind Bhork\Downloads\
Rawdata.XLSX')
```

```
emp
```

	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+

```
id(emp)
```

```
1948209461248
```

```
emp.columns
```

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

```
emp.shape
```

```
(6, 6)
```

```
emp
```

	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+

```
emp.head()
```

```
     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
```

```
emp.tail()
```

```
     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+
```

```
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
```

```
emp
```

```
     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+
```

```
emp.isnull()
```

```
    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
```

```
emp.isna()
```

```
    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
```

```
emp.isnull().sum()
```

```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

```
emp['Name']
```

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

DATA CLEAING OR DATA CLEANSING

```
emp['Name']
```

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

```
emp['Name'] = emp['Name'].str.replace(r'\W','',regex=True)
```

```
emp['Name'] #cleaned Name atribute
```

```
0     Mike
1    Teddy
2     Umar
3     Jane
4    Uttam
```

```
5      Kim
Name: Name, dtype: object
```

```
emp['Domain']
```

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

```
emp['Domain'] = emp['Domain'].str.replace(r'\W','',regex=True)
#formula for cleaning
```

```
emp['Domain'] #cleaned domain
```

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

```
emp
```

	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+

```
emp['Age']
```

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

```
emp['Age'] = emp['Age'].str.replace(r'\W','',regex=True) #by this
irregular text removed but not words we need to remove text totally
```

```
emp['Age']
```

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

```
emp['Age']=emp['Age'].str.extract(r'(\d+)') #by this code we can
remove the text totally from our numerical data
```

```
emp['Age'] #sucesfully removed text data
```

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

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

```
emp['Location']
```

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

```
emp
```

```
    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+
```

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

```
emp['Salary']
```

```
0     5000
1    10000
2    15000
3    20000
4    30000
```

```
5    60000
Name: Salary, dtype: object
```

```
emp.head()
```

```
    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
```

```
emp['Exp']=emp['Exp'].str.extract(r'(\d+)') #by this code we can
remove the text totally from our numerical data
```

```
emp['Exp']
```

```
0      2
1      3
2      4
3    NaN
4      5
5     10
Name: Exp, dtype: object
```

```
emp
```

```
    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
```

```
clean_data =  emp.copy()
```

```
clean_data
```

```
    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
```

LETS APPLY EDA TECHNIQUE :-

```
clean_data
```

```
    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
```

```
clean_data.isnull().sum()
```

```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

```
clean_data['Age']
```

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

```
import numpy as np
```

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

```
clean_data['Age']
```

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

```
clean_data['Exp']=clean_data['Exp'].fillna(np.mean(pd.to_numeric(clean
_data['Exp'])))
```

```
clean_data['Exp']
```

```
0      2
1      3
2      4
3    4.8
```

```
4      5
5     10
Name: Exp, dtype: object
```

```
clean_data
```

```
    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
```

```
clean_data['Location']
```

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

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

```
clean_data['Location']
```

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

```
clean_data
```

```
    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
```

```
clean_data.isnull()
```

```
    Name  Domain    Age  Location  Salary    Exp
0  False   False  False     False   False  False
1  False   False  False     False   False  False
```

```
2  False   False  False     False   False  False
3  False   False  False     False   False  False
4  False   False  False     False   False  False
5  False   False  False     False   False  False
```

```
emp.info
```

```
<bound method DataFrame.info of     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>
```

```
clean_data.info()          #system bydefault capture it as a object
but we have to convert it as integer
```

```
<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
```

```
clean_data['Age'] = clean_data['Age'].astype(int)
```

```
clean_data.info()    #hence we sucessfully converted object into
integer
```

```
<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      object
 5   Exp       6 non-null      object
dtypes: int32(1), object(5)
memory usage: 396.0+ bytes
```

```
clean_data['Salary'] = clean_data['Salary'].astype(int)

clean_data.info()   #Here we converted salary into int 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      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      object
dtypes: int32(2), object(4)
memory usage: 372.0+ bytes
```

```
clean_data
```

	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	4.8
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

```
clean_data['Salary'] = clean_data['Salary'].astype(int)
clean_data['Exp']    = clean_data['Exp'].astype(int)

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
```

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

```
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
```

```
clean_data
```

	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

```
clean_data.to_csv('clean_data.csv')
```

```
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
Cell In[172], line 1
----> 1 clean_data.to_csv('clean_data.csv')

File c:\Users\Bhavesh Govind Bhork\anaconda3\Lib\site-packages\pandas\util\_decorators.py:333, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    327 if len(args) > num_allow_args:
    328     warnings.warn(
    329         msg.format(arguments=_format_argument_list(allow_args)),
    330         FutureWarning,
    331         stacklevel=find_stack_level(),
    332     )
--> 333 return func(*args, **kwargs)

File c:\Users\Bhavesh Govind Bhork\anaconda3\Lib\site-packages\pandas\core\generic.py:3967, in NDFrame.to_csv(self, path_or_buf, sep, na_rep, float_format, columns, header, index, index_label, mode,
```

```
encoding, compression, quoting, quotechar, lineterminator, chunksize,
date_format, doublequote, escapechar, decimal, errors,
storage_options)
   3956 df = self if isinstance(self, ABCDataFrame) else
self.to_frame()
   3958 formatter = DataFrameFormatter(
   3959     frame=df,
   3960     header=header,
   (...)   3964     decimal=decimal,
   3965 )
-> 3967 return DataFrameRenderer(formatter).to_csv(
   3968     path_or_buf,
   3969     lineterminator=lineterminator,
   3970     sep=sep,
   3971     encoding=encoding,
   3972     errors=errors,
   3973     compression=compression,
   3974     quoting=quoting,
   3975     columns=columns,
   3976     index_label=index_label,
   3977     mode=mode,
   3978     chunksize=chunksize,
   3979     quotechar=quotechar,
   3980     date_format=date_format,
   3981     doublequote=doublequote,
   3982     escapechar=escapechar,
   3983     storage_options=storage_options,
   3984 )

File c:\Users\Bhavesh Govind Bhork\anaconda3\Lib\site-packages\pandas\
io\formats\format.py:1014, in DataFrameRenderer.to_csv(self,
path_or_buf, encoding, sep, columns, index_label, mode, compression,
quoting, quotechar, lineterminator, chunksize, date_format,
doublequote, escapechar, errors, storage_options)
    993     created_buffer = False
    995 csv_formatter = CSVFormatter(
    996     path_or_buf=path_or_buf,
    997     lineterminator=lineterminator,
   (...)   1012     formatter=self.fmt,
   1013 )
-> 1014 csv_formatter.save()
   1016 if created_buffer:
   1017     assert isinstance(path_or_buf, StringIO)

File c:\Users\Bhavesh Govind Bhork\anaconda3\Lib\site-packages\pandas\
io\formats\csvs.py:251, in CSVFormatter.save(self)
    247 """
    248 Create the writer & save.
    249 """
```

```
    250 # apply compression and byte/text conversion
--> 251 with get_handle(
    252     self.filepath_or_buffer,
    253     self.mode,
    254     encoding=self.encoding,
    255     errors=self.errors,
    256     compression=self.compression,
    257     storage_options=self.storage_options,
    258 ) as handles:
    259     # Note: self.encoding is irrelevant here
    260     self.writer = csvlib.writer(
    261         handles.handle,
    262         lineterminator=self.lineterminator,
   (...)    267         quotechar=self.quotechar,
    268     )
    270     self._save()

File c:\Users\Bhavesh Govind Bhork\anaconda3\Lib\site-packages\pandas\
io\common.py:873, in get_handle(path_or_buf, mode, encoding,
compression, memory_map, is_text, errors, storage_options)
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary
mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

PermissionError: [Errno 13] Permission denied: 'clean_data.csv'
```

```
import os
os.getcwd()
```

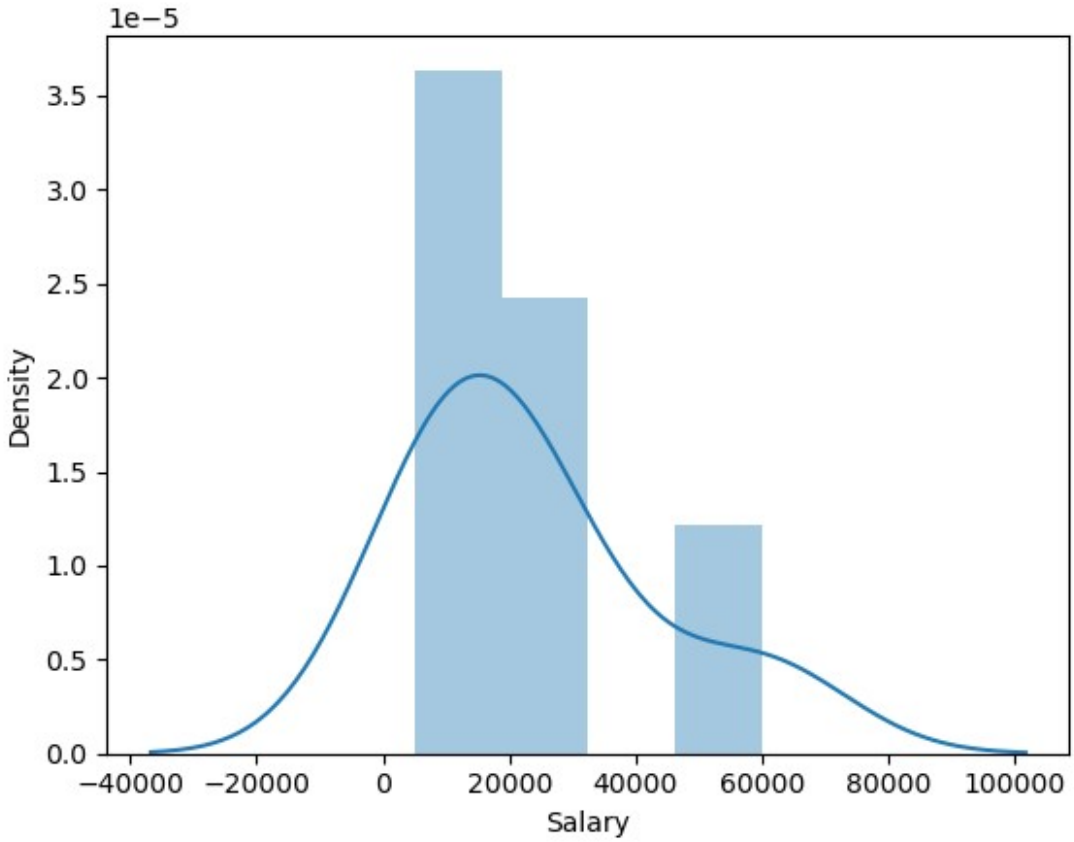
```
'c:\\Users\\Bhavesh Govind Bhork\\Downloads'
```

^ we cleaned the RAW DATA into CLEAN DATA and also imported that data into our system

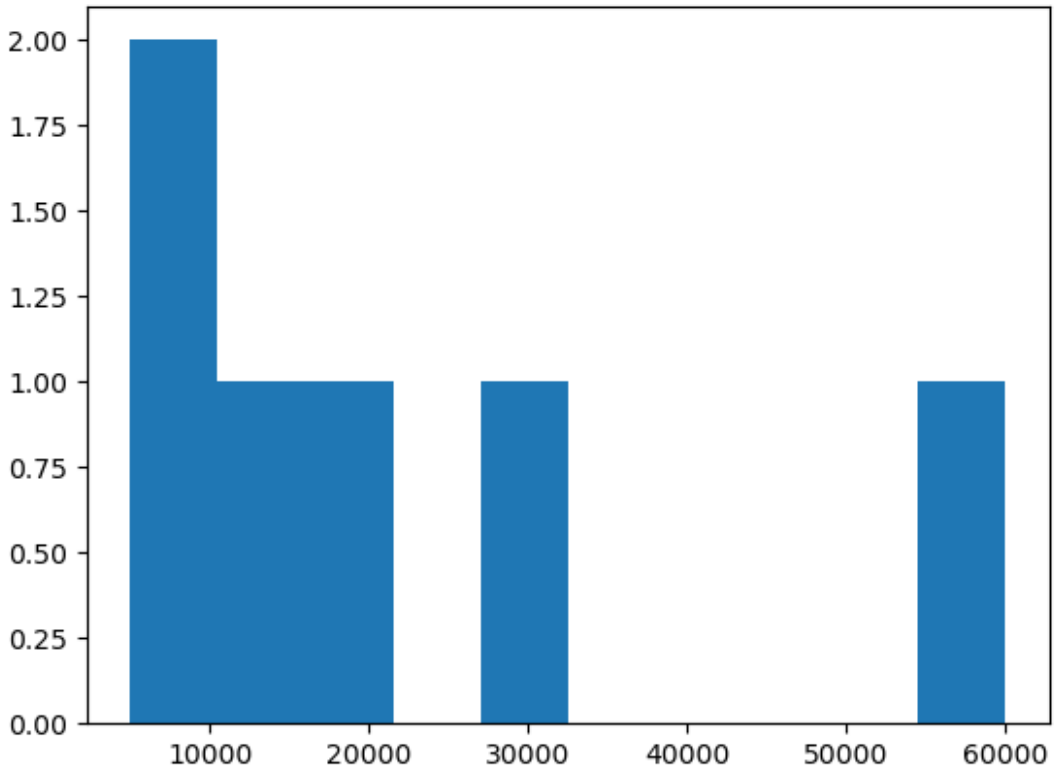
```
import warnings
warnings.filterwarnings('ignore')

clean_data['Salary']
```

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

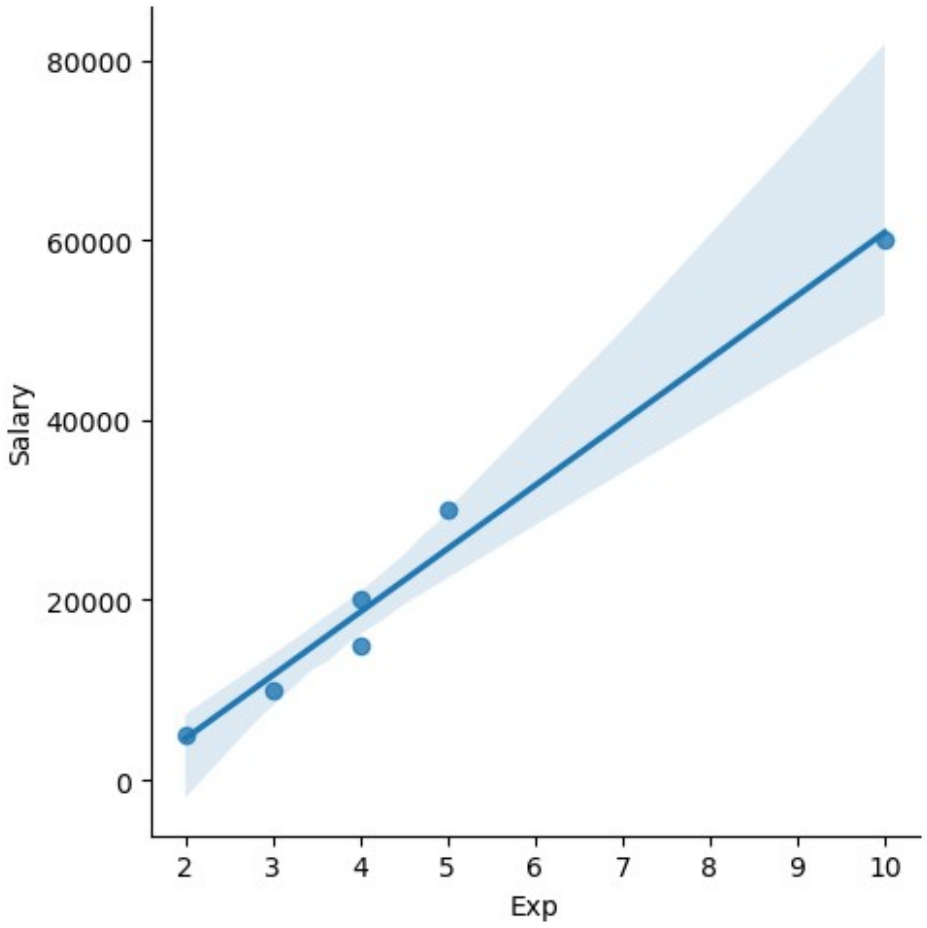
```
import seaborn as sns #we have to import library first to visualize
the data

visl = sns.distplot(clean_data['Salary'])
```



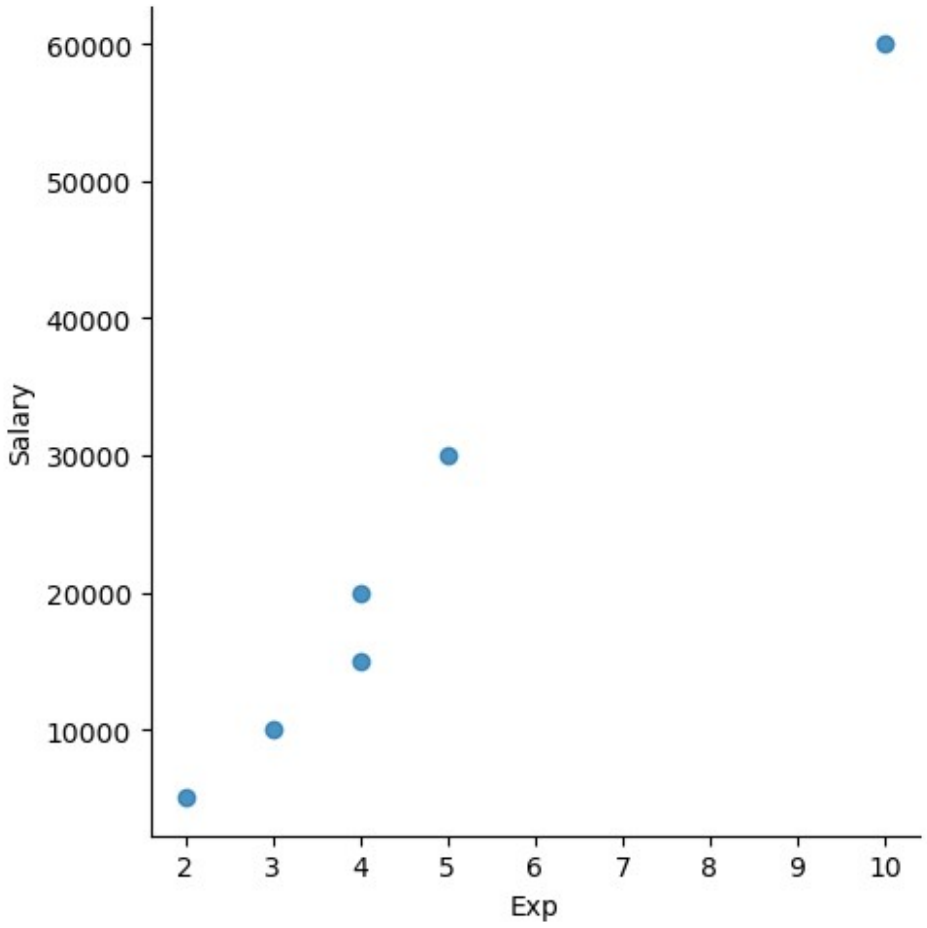

```
import matplotlib.pyplot as plt

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




```
vis4 = sns.lmplot(data=clean_data, x='Exp', y='Salary') #linear model
plot
```




```
vis5 = sns.lmplot(data=clean_data, x='Exp', y='Salary', fit_reg=False)
```




```
clean_data[:]
```

	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

```
x_4 = clean_data[['Name', 'Domain', 'Age', 'Location', 'Exp']]

x_4 #these all are independent vaariables
```

	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

```
y_dv = clean_data[['Salary']]

y_dv  #this is dependent variable
```

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

```
emp
```

```
    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
```

```
clean_data
```

```
    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
```

```
x_4
```

```
    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
```

```
y_dv
```

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

```
clean_data
```

```
    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
```

```
imputation = pd.get_dummies(clean_data, dtype=int)
```

```
imputation
```

```
   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

   Name_Uttam  Domain_Analytics  Domain_Dataanalyst
Domain_Datascience  \
0           0                 0                   0
1
1           0                 0                   0
0
2           0                 0                   1
0
3           0                 1                   0
0
4           1                 0                   0
0
5           0                 0                   0
0

   Domain_NLP  Domain_Statistics  Domain_Testing
Location_Bangalore  \
0           0                  0               0                    0

1           0                  0               1                    1
```

```
2           0                   0                0                   1

3           0                   0                0                   0

4           0                   1                0                   1

5           1                   0                0                   0
```

```
   Location_Delhi  Location_Hyderbad  Location_Mumbai
0               0                  0                1
1               0                  0                0
2               0                  0                0
3               0                  1                0
4               0                  0                0
5               1                  0                0
```

```
imputation.columns
```

```
Index(['Age', 'Salary', 'Exp', 'Name_Jane', 'Name_Kim', 'Name_Mike',
       'Name_Teddy', 'Name_Umar', 'Name_Uttam', 'Domain_Analytics',
       'Domain_Dataanalyst', 'Domain_Datascience', 'Domain_NLP',
       'Domain_Statistics', 'Domain_Testing', 'Location_Bangalore',
       'Location_Delhi', 'Location_Hyderbad', 'Location_Mumbai'],
      dtype='object')
```

```
len(imputation.columns)
```

```
19
```

in this project we have covered:-

. raw data with lot of regx, missing, uncleaned regex, clean

. filled missing numerical and category

. clean data set

. outlier treatment, univarient, bivarient, corelation

. split the data into x_4 and y_dv

. impute category data to numerical

. EDA done