```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import pandas as pd
sal_data = pd.read_csv(r'/content/archive.zip')
sal_data.head()
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	High School	8	Urban	Manager	63	Male	84620.053665
1	PhD	11	Suburban	Director	59	Male	142591.255894
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404
3	High School	29	Rural	Director	45	Male	96834.671282
4	PhD	25	Urban	Analyst	26	Female	132157.786175

Next steps: Generate code with sal_data | View recommended plots | New interactive sheet

```
sal_data.shape
```

```
(1000, 7)
```

```
display(sal_data.columns)
```

```
Index(['Education', 'Experience', 'Location', 'Job_Title', 'Age', 'Gender',
       'Salary'],
      dtype='object')
```

```
sal_data.columns = ['Education', 'Experience', 'Location', 'Job_Title', 'Age', 'Gender', 'Salary']
sal_data1 = sal_data.copy() # Create a copy after renaming to avoid modifying the original DataFrame if needed later
```

```
sal_data.head()
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	High School	8	Urban	Manager	63	Male	84620.053665
1	PhD	11	Suburban	Director	59	Male	142591.255894
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404
3	High School	29	Rural	Director	45	Male	96834.671282
4	PhD	25	Urban	Analyst	26	Female	132157.786175

```
sal_data.dtypes
```

	0
Education	object
Experience	int64
Location	object
Job_Title	object
Age	int64
Gender	object
Salary	float64

dtype: object

```
sal_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Education   1000 non-null   object
 1   Experience  1000 non-null   int64
 2   Location    1000 non-null   object
 3   Job_Title   1000 non-null   object
 4   Age         1000 non-null   int64
 5   Gender      1000 non-null   object
 6   Salary      1000 non-null   float64
dtypes: float64(1), int64(2), object(4)
memory usage: 54.8+ KB
```

```
sal_data[sal_data.duplicated()]
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary

```
sal_data[sal_data.duplicated()].shape
```

```
(0, 7)
```

```
sal_data1 = sal_data.drop_duplicates(keep = 'first')
sal_data1.shape
```

```
(1000, 7)
```

```
sal_data1.isnull().sum()
```

	0
Education	0
Experience	0
Location	0
Job_Title	0
Age	0
Gender	0
Salary	0

dtype: int64

```
sal_data1.shape
```

```
(1000, 7)
```

```
sal_data1.head()
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	High School	8	Urban	Manager	63	Male	84620.053665
1	PhD	11	Suburban	Director	59	Male	142591.255894
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404
3	High School	29	Rural	Director	45	Male	96834.671282
4	PhD	25	Urban	Analyst	26	Female	132157.786175

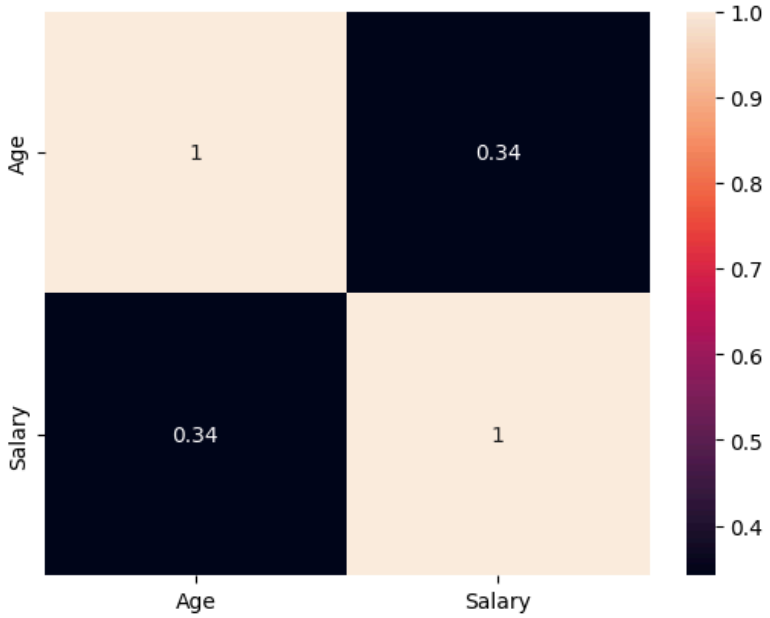
```
sal_data1.describe()
```

	Experience	Age	Salary
count	1000.000000	1000.000000	1000.000000
mean	14.771000	42.377000	105558.404239
std	8.341111	13.609412	28256.972075
min	1.000000	20.000000	33510.510669
25%	7.000000	30.000000	85032.141517
50%	15.000000	43.000000	104314.518315
75%	22.000000	55.000000	126804.047524
max	29.000000	64.000000	193016.602150

```
corr = sal_data1[['Age', 'Salary']].corr()
corr
```

```
import seaborn as sns
sns.heatmap(corr, annot = True)
```

```
<Axes: >
```



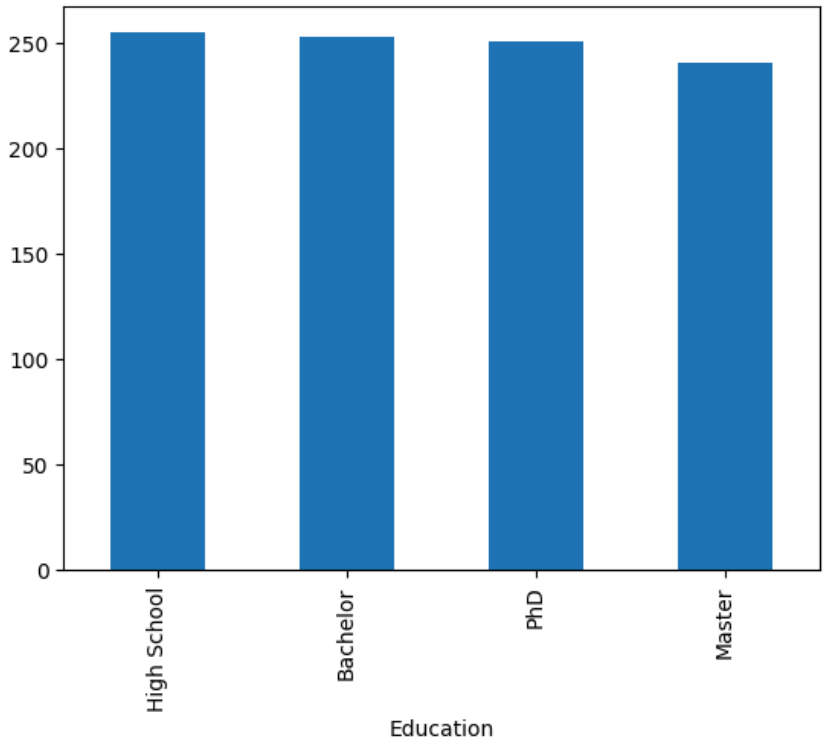

```
sal_data1['Education'].value_counts()
```

Education	count
High School	255
Bachelor	253
PhD	251
Master	241

dtype: int64

```
sal_data1['Education'].value_counts().plot(kind = 'bar')
```

```
<Axes: xlabel='Education'>
```




```
sal_data1['Job_Title'].value_counts()
```

Job_Title	count
27	33
60	33
58	30
59	30
62	30
21	30
24	29
20	28
41	28
63	28
44	27
45	25
49	25
54	25
61	24
25	24
23	24
42	24
31	23
48	23
26	22
57	22
50	22
36	22
37	21
64	21
56	21
52	20
40	20
43	19
39	19
22	19
29	19
34	19
53	18
33	18
35	18
47	16
28	15
46	15
51	15
38	15

30	14
55	14
32	13

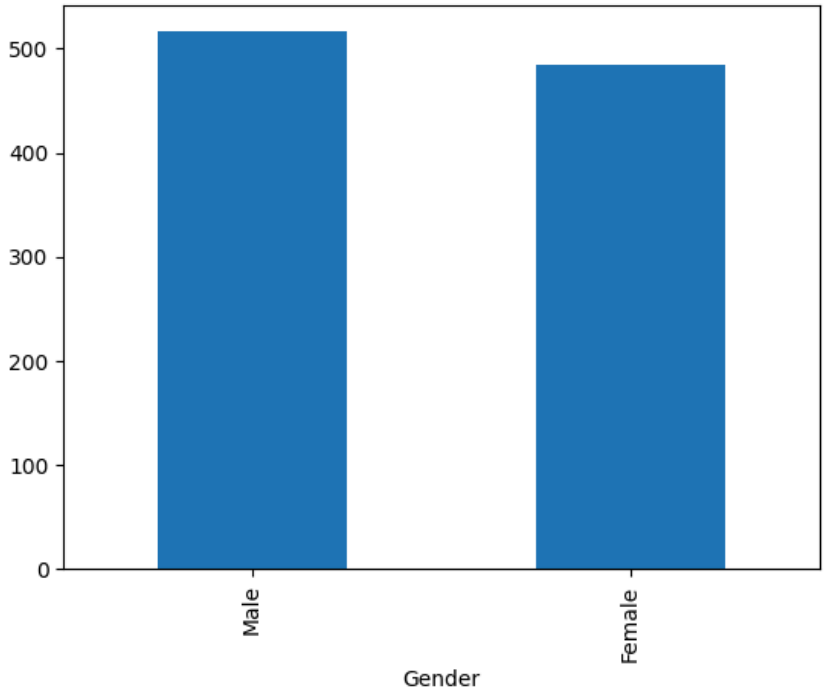
dtype: int64

```
sal_data1['Job_Title'].unique()
```

```
array([63, 59, 61, 45, 26, 27, 60, 49, 25, 58, 23, 43, 44, 37, 53, 34, 62,
       36, 21, 20, 35, 28, 40, 22, 50, 33, 31, 47, 64, 24, 57, 32, 48, 46,
       42, 51, 41, 56, 54, 30, 38, 29, 52, 39, 55])
```

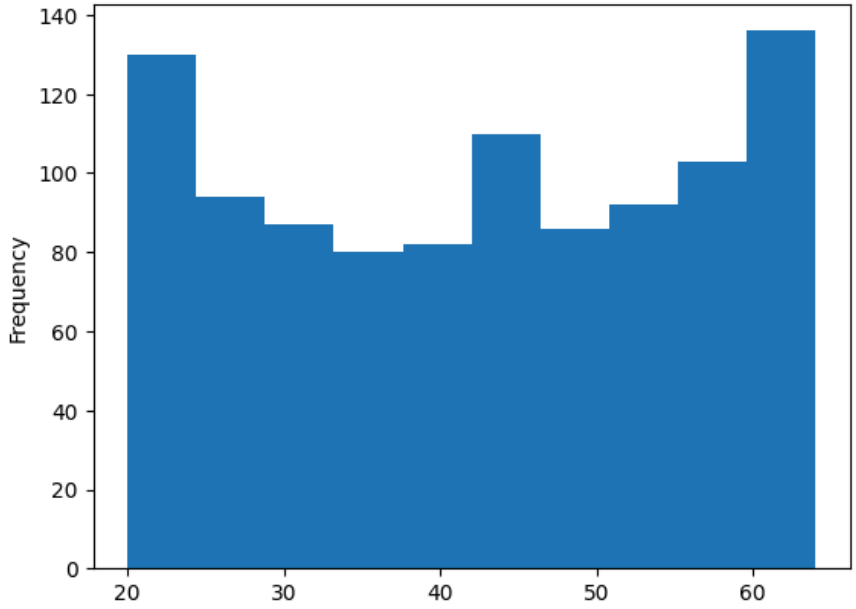
```
sal_data1['Gender'].value_counts().plot(kind = 'bar')
```

```
<Axes: xlabel='Gender'>
```



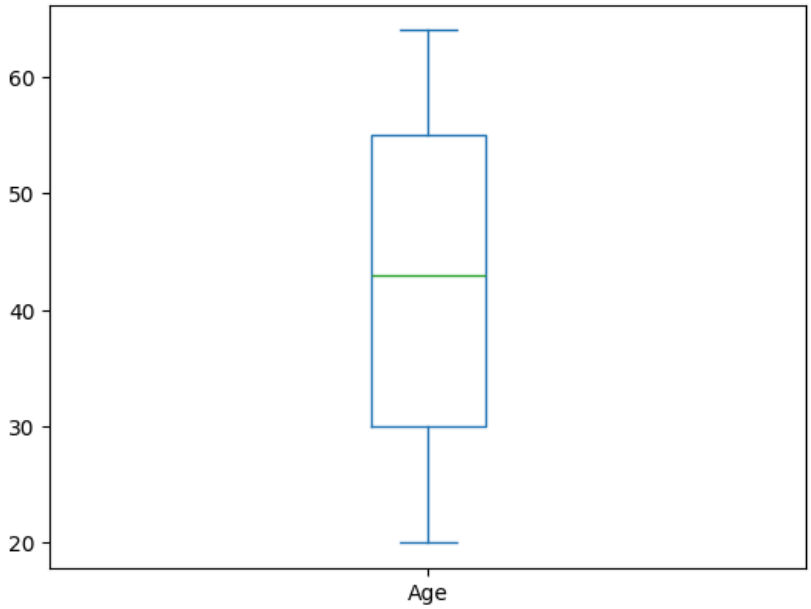

```
sal_data1['Age'].plot(kind = 'hist')
```

```
<Axes: ylabel='Frequency'>
```



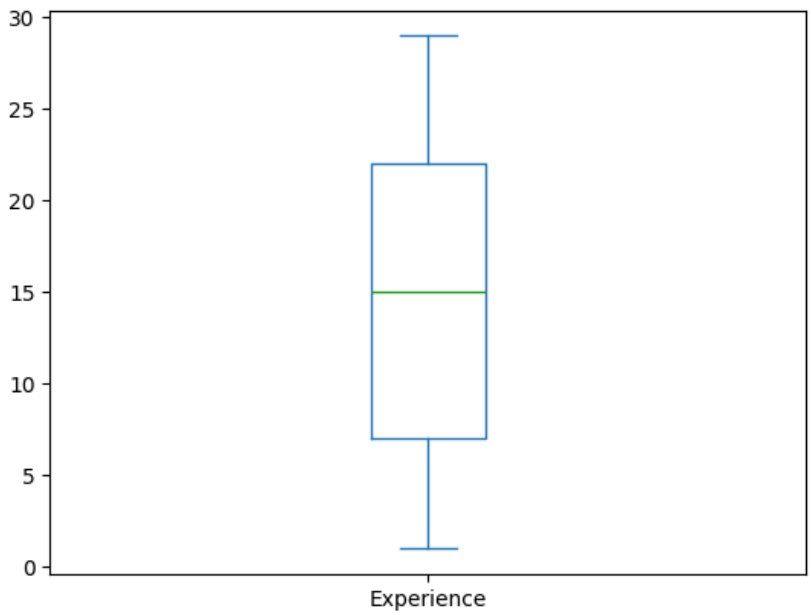
```
sal_data1.Age.plot(kind = 'box')
```

```
<Axes: >
```



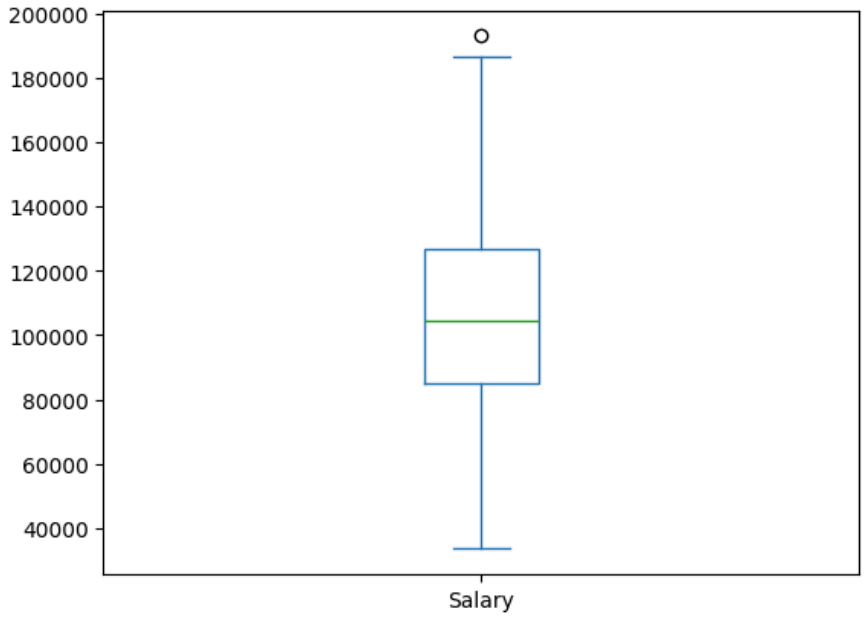

```
import pandas as pd

# Load the data
sal_data = pd.read_csv(r'/content/archive.zip')

# Create sal_data1 as a copy
sal_data1 = sal_data.copy()

# The original code to plot the box plot
sal_data1.Experience.plot(kind = 'box')
```

```
<Axes: >
```



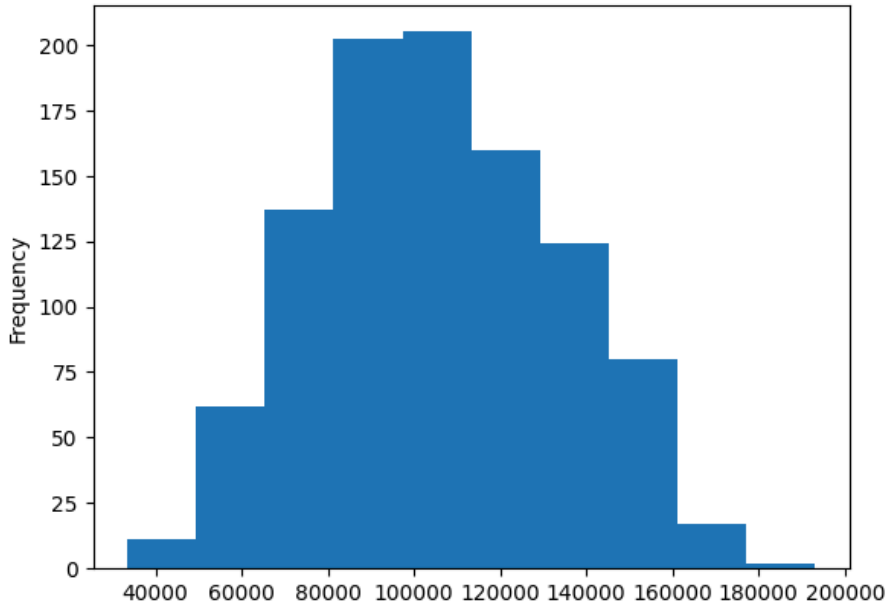

```
sal_data1.Salary.plot(kind = 'box')
```

```
<Axes: >
```




```
sal_data1.Salary.plot(kind = 'hist')
```

```
<Axes: ylabel='Frequency'>
```




```
sal_data1.head()
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	High School	8	Urban	Manager	63	Male	84620.053665
1	PhD	11	Suburban	Director	59	Male	142591.255894
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404
3	High School	29	Rural	Director	45	Male	96834.671282
4	PhD	25	Urban	Analyst	26	Female	132157.786175

Next steps: Generate code with sal_data1 View recommended plots New interactive sheet

```
from sklearn.preprocessing import LabelEncoder
LabelEncoder = LabelEncoder()
```

```
sal_data1['Gender_Encode'] = LabelEncoder.fit_transform(sal_data1['Gender'])
```

```
sal_data1['Education_Encode'] = LabelEncoder.fit_transform(sal_data1['Education'])
```

```
sal_data1['Job_Title_Encode'] = LabelEncoder.fit_transform(sal_data1['Job_Title'])
```

```
sal_data1.head()
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary	Gender_Encode	Education_Encode	Job_Tit
0	High School	8	Urban	Manager	63	Male	84620.053665	1	1	
1	PhD	11	Suburban	Director	59	Male	142591.255894	1	3	
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404	0	0	
3	High School	29	Rural	Director	45	Male	96834.671282	1	1	
4	PhD	25	Urban	Analyst	26	Female	132157.786175	0	3	

Next steps: Generate code with sal_data1 View recommended plots New interactive sheet

```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
```

```
sal_data1['Age_scaled'] = std_scaler.fit_transform(sal_data1[['Age']])
sal_data1['Experience_scaled'] = std_scaler.fit_transform(sal_data1[['Experience']])
```

```
sal_data1.head()
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary	Gender_Encode	Education_Encode	Job_Tit
0	High School	8	Urban	Manager	63	Male	84620.053665	1	1	
1	PhD	11	Suburban	Director	59	Male	142591.255894	1	3	
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404	0	0	
3	High School	29	Rural	Director	45	Male	96834.671282	1	1	
4	PhD	25	Urban	Analyst	26	Female	132157.786175	0	3	

Next steps: Generate code with sal_data1 View recommended plots New interactive sheet

```
x = sal_data1[['Age_scaled','Gender_Encode','Education_Encode','Job_Title_Encode','Experience_scaled']]
y = sal_data1['Salary']
```

```
x.head()
```

	Age_scaled	Gender_Encode	Education_Encode	Job_Title_Encode	Experience_scaled
0	1.516107	1	1	3	-0.812169
1	1.222045	1	3	1	-0.452324
2	1.369076	0	0	3	1.586793
3	0.192831	1	1	1	1.706741
4	-1.203961	0	3	0	1.226949

Next steps: Generate code with x View recommended plots New interactive sheet

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=42)
```

```
x_train.head()
```

	Age_scaled	Gender_Encode	Education_Encode	Job_Title_Encode	Experience_scaled
29	1.369076	0	1	0	-0.092480
535	-1.571537	1	1	1	-0.932117
695	-1.498022	0	2	1	-0.452324
557	1.295561	0	3	0	-1.291961
836	1.516107	0	0	3	0.267364

Next steps: Generate code with x_train | View recommended plots | New interactive sheet

```
x_train.shape, y_train.shape # 80%
```

```
((800, 5), (800,))
```

```
x_test.shape, y_test.shape # 20%
```

```
((200, 5), (200,))
```

```
from sklearn.linear_model import LinearRegression
Linear_regeression_model = LinearRegression()
```

```
x_train = x_train.fillna(x_train.mean())
# Remove rows where y_train is NaN
nan_mask = y_train.isna()
x_train = x_train[~nan_mask]
y_train = y_train[~nan_mask]
Linear_regeression_model.fit(x_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
y_pred_lr = Linear_regeression_model.predict(x_test)
y_pred_lr
```

```
array([ 78736.67935912,  84243.12350731, 110383.66811572,  69940.84563019,
        94586.35054785,  67240.90939562,  63279.38907311, 120665.77884558,
       122720.94914157, 108756.54613555, 142945.11287477, 136508.06311966,
       123755.6426505 ,  68130.13737156,  83608.78132127,  69695.60695722,
       133593.30548726,  83010.3104424 ,  97363.55135463,  81802.43877348,
       119808.37436459,  81587.60174159,  86788.5149634 , 144285.33731007,
       141065.04875129,  81586.77902726, 117495.61497352,  83000.72503826,
       142654.37141176, 129065.10651271, 108758.45851401,  83047.02009796,
       116566.68606339, 123015.2336011 , 117620.85130221, 128624.20161613,
        87487.83301233,  92561.32705004,  79505.30719417, 135987.69516139,
       138531.42555476, 108526.09692596,  98529.52359987, 106225.67366587,
        83530.67840067,  90904.59569845,  89726.85519365, 119127.43485428,
       117201.33051399, 123958.96626572,  86292.25139569,  73970.89910321,
        71267.11895051,  95108.91699561, 103973.47700517,  75989.05747908,
       107111.29693251,  92732.3223546 ,  96887.29560064,  83580.81737851,
       114652.39606139, 129654.78425711, 138165.0649482 ,  88973.28729893,
       131504.44673951, 113874.97509186, 104417.14826266, 127026.08582118,
        81047.76205341,  90611.65574596, 134726.91815371, 119578.74786933,
       127943.24647175, 146763.04959809,  99207.44190637,  81435.7772453 ,
       119700.96299421,  87178.71301071, 109579.42381639,  78450.30360695,
        81555.79388069, 110158.42296536, 107042.82549477,  97706.63028496,
        70317.09256251,  88056.67888573,  92576.11651341, 103162.93545532,
        81733.08290863, 140041.28515354,  64400.41076568, 133020.03219023,
        84704.03756056,  79088.5053452 , 111233.14959822, 125914.95045438,
       128303.31335495, 117313.96023473, 112784.93854586,  82483.37828379,
        79047.41434473, 102329.90449891,  72162.89549271,  86205.40141934,
       125391.56129229,  86805.4872822 , 103011.41188065, 106961.70137037,
```

```
       117978.44921896, 115821.07295475,  95162.59896997,  91147.36540498,
       114502.75442052,  88815.73695071,  94275.61556224, 141002.57364245,
       126510.40012944, 119423.38037652, 122940.9902327 , 103313.09888889,
        84041.4609548 ,  91960.9437927 , 118897.80835901, 123362.17208363,
       123456.97684607, 108380.29920323,  81495.50162728, 134350.68685545,
       133543.42135231,  83373.11241838,  75769.01638796, 128969.46340188,
       136357.89968609,  88001.59069155,  81067.23378331,  92338.53523206,
       118063.93905421,  83956.22596244,  82338.41890945, 114944.49766549,
       123304.06268564, 105997.4073117 , 119947.29133131, 143310.11334023,
       127036.49393965, 111227.94553899, 116940.73450622,  65894.65818675,
       105530.19953508, 145482.27906781, 119425.56323194, 104696.91373577,
        67253.51600357, 102122.19953877, 116814.67546321,  99132.36018959,
        82188.25547588,  83878.96139023, 134578.11486123, 143985.84879131,
        84622.39164345,  92636.67924378,  98632.55362535,  88689.66227363,
        86295.2725995 , 138016.26165572, 141525.12445615, 134941.75518561,
        82184.41155774, 120975.16932416, 140407.94668172, 130248.05107674,
       120833.75294634,  82927.82617689,  94047.61968503, 142944.27452638,
        85913.29981523,  90681.53340352,  97716.78356054,  94290.40502561,
        86744.40275912,  79714.10181844,  84804.88473062, 129505.18869496,
       116964.83889678,  94347.10820378, 122327.72989046,  97186.26232669,
       116337.89791651, 108020.2479541 ,  90409.99319345,  97765.53195262,
       125932.4906446 ,  93975.55917205, 108676.26035953,  65233.69656498,
        88679.25415516,  99640.9598883 , 138074.37105371,  73680.15764021])
```

```
df = pd.DataFrame({'y_Actual': y_test, 'y_Predicted': y_pred_lr})
df['Error'] = df['y_Actual'] - df['y_Predicted']
df['abs_error'] = abs(df['Error'])
df
```

	y_Actual	y_Predicted	Error	abs_error
521	86677.840109	78736.679359	7941.160750	7941.160750
737	56036.163010	84243.123507	-28206.960498	28206.960498
740	92226.871819	110383.668116	-18156.796296	18156.796296
660	100710.088052	69940.845630	30769.242422	30769.242422
411	91775.012832	94586.350548	-2811.337716	2811.337716
...	...	...	...	...
408	62915.445683	65233.696565	-2318.250882	2318.250882
332	92041.749991	88679.254155	3362.495836	3362.495836
208	85534.397486	99640.959888	-14106.562402	14106.562402
613	164373.967469	138074.371054	26299.596415	26299.596415
78	93375.479141	73680.157640	19695.321501	19695.321501

200 rows × 4 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
Mean_absolute_error = df['abs_error'].mean()
Mean_absolute_error
```

```
np.float64(15507.313542911043)
```

```
from sklearn.metrics import accuracy_score, r2_score
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
r2_score(y_test, y_pred_lr)
```

```
0.5538121948118675
```

```
print(f'Accuracy of the model = {round(r2_score(y_test, y_pred_lr),4)*100} %')
```

```
Accuracy of the model = 55.379999999999995 %
```

```
round(mean_squared_error(y_test, y_pred_lr),2)
```

```
364324191.67
```

```
print(f"Mean Absolute Error = {round(mean_squared_error(y_test, y_pred_lr),2)}")
```

```
Mean Absolute Error = 364324191.67
```

```
mse = round(mean_squared_error(y_test, y_pred_lr),2)
mse
```

```
364324191.67
```

```
print(f"Mean Sqaured Error = {round(mean_squared_error(y_test, y_pred_lr),2)}")
```

```
Mean Sqaured Error = 364324191.67
```

```
print('Root Mean Squared Error (RMSE) =', mse**(0.5))
```

```
Root Mean Squared Error (RMSE) = 19087.27826773634
```

```
Linear_regeression_model.coef_
```

```
array([-1021.30695884,   886.19113807, 16717.49942225,  1869.65600501,
        8719.93822176])
```

```
Linear_regeression_model.intercept_
```

```
np.float64(77061.4558527059)
```

```
sal_data1.head()
```

	Education	Experience	Location	Job_Title	Age	Gender	Salary	Gender_Encode	Education_Encode	Job_Tit
0	High School	8	Urban	Manager	63	Male	84620.053665	1	1	
1	PhD	11	Suburban	Director	59	Male	142591.255894	1	3	
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404	0	0	
3	High School	29	Rural	Director	45	Male	96834.671282	1	1	
4	PhD	25	Urban	Analyst	26	Female	132157.786175	0	3	

Next steps: Generate code with sal_data1 | View recommended plots | New interactive sheet

```
# Re-fit the scaler
std_scaler = StandardScaler()
sal_data1['Age_scaled'] = std_scaler.fit_transform(sal_data1[['Age']])
sal_data1['Experience_scaled'] = std_scaler.fit_transform(sal_data1[['Experience']])
```

```
Age1 = std_scaler.transform([[49]])
Gender = 0
Degree = 2
Job_Title = 22
Experience_years1 = std_scaler.transform([[15]])
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid featu
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid featu
  warnings.warn(
```

```
std_scaler.transform([[15]])
```