26-10-23

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
# DAta mining
df = pd.read_csv('emp.csv')
df.head(1)
```

Out[5]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus	Senior Management	Team
0	Douglas	Male	08/06/1993	12:42 pm	97308	6.945	True	Marketing

In [6]:
```
df.sample(5)
```

Out[6]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus	Senior Management	Team
247	Brenda	NaN	7/27/2010	11:07 pm	106115	3.742	True	Product
724	Andrea	Female	12/10/2001	6:40 am	37888	13.470	False	Engineering
361	Margaret	Female	05/05/2014	6:01 am	55044	4.078	False	Sales
290	Jeremy	Male	6/14/1988	6:20 pm	129460	13.657	True	NaN
776	Bobby	Male	9/30/2004	10:52 pm	79047	18.784	False	Human Resources

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
First Name           67
Gender              145
Start Date            0
Last Login Time       0
Salary                0
Bonus                 0
Senior Management    67
Team                 43
dtype: int64
```

In [8]:
```
df.isnull().all
```

Out[8]:
```
<bound method NDFrame._add_numeric_operations.<locals>.all of      First Name
Gender  Start Date  Last Login Time  Salary  Bonus  \
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
..          ...     ...         ...              ...     ...    ...
995       False    True       False            False   False  False
996       False   False       False            False   False  False
997       False   False       False            False   False  False
998       False   False       False            False   False  False
999       False   False       False            False   False  False

     Senior Management   Team
0                False  False
1                False   True
2                False  False
3                False  False
4                False  False
..                 ...    ...
995              False  False
996              False  False
997              False  False
998              False  False
999              False  False

[1000 rows x 8 columns]>
```

In [9]:
```
df.isnull().any()
```

Out[9]:
```
First Name           True
Gender               True
Start Date          False
Last Login Time     False
Salary              False
Bonus               False
Senior Management    True
Team                 True
dtype: bool
```

In [10]:
```
df['Salary'].isnull()
```

Out[10]:
```
0      False
1      False
2      False
3      False
4      False
       ...
995    False
996    False
997    False
998    False
999    False
Name: Salary, Length: 1000, dtype: bool
```

In [11]:
```
df['Salary'].isnull().all()
```

Out[11]:
```
False
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   First Name        933 non-null    object 
 1   Gender            855 non-null    object 
 2   Start Date        1000 non-null   object 
 3   Last Login Time   1000 non-null   object 
 4   Salary            1000 non-null   int64  
 5   Bonus             1000 non-null   float64
 6   Senior Management  933 non-null    object 
 7   Team              957 non-null    object 
dtypes: float64(1), int64(1), object(6)
memory usage: 62.6+ KB
```

In [13]:
```
df.describe()
```

Out[13]:

	Salary	Bonus
count	1000.000000	1000.000000
mean	90662.181000	10.207555
std	32923.693342	5.528481
min	35013.000000	1.015000
25%	62613.000000	5.401750
50%	90428.000000	9.838500
75%	118740.250000	14.838000
max	149908.000000	19.944000

In [14]:
```
# Data cleaning

df['First Name']=df['First Name'].fillna('NA')
df['First Name'].sample(25)
```

Out[14]:
```
13          Gary
741         Jane
347         Lori
152         Ruth
458       Albert
326      Jeffrey
456      Deborah
784    Stephanie
344        Scott
814       Rachel
928      Jeffrey
846      Stephen
376      Brandon
208     Jonathan
508        Scott
542       Amanda
278        Betty
320           NA
459      Charles
462        Craig
534       Gerald
989       Justin
623        Irene
686         Paul
762        Terry
Name: First Name, dtype: object
```

In [15]:
```
df['First Name']=df['First Name'].replace(to_replace='NA', value='Seemi')
df['First Name'].sample(25)
```

Out[15]:
```
54           Sara
673         Ralph
555          Anne
319    Jacqueline
190         Carol
654          Carl
226         Kathy
462         Craig
894         Betty
491      Nicholas
576       Michael
676         Annie
930         Nancy
918          Ryan
832         Keith
970         Alice
460          Tina
675         Diane
218       Gregory
971       Patrick
90         Janice
823         Seemi
336          Mark
640      Kathleen
306          Mark
Name: First Name, dtype: object
```

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   First Name        1000 non-null   object 
 1   Gender            855 non-null    object 
 2   Start Date        1000 non-null   object 
 3   Last Login Time   1000 non-null   object 
 4   Salary            1000 non-null   int64  
 5   Bonus             1000 non-null   float64
 6   Senior Management  933 non-null    object 
 7   Team              957 non-null    object 
dtypes: float64(1), int64(1), object(6)
memory usage: 62.6+ KB
```

In [17]:
```
df.dropna(inplace=True)
df
```

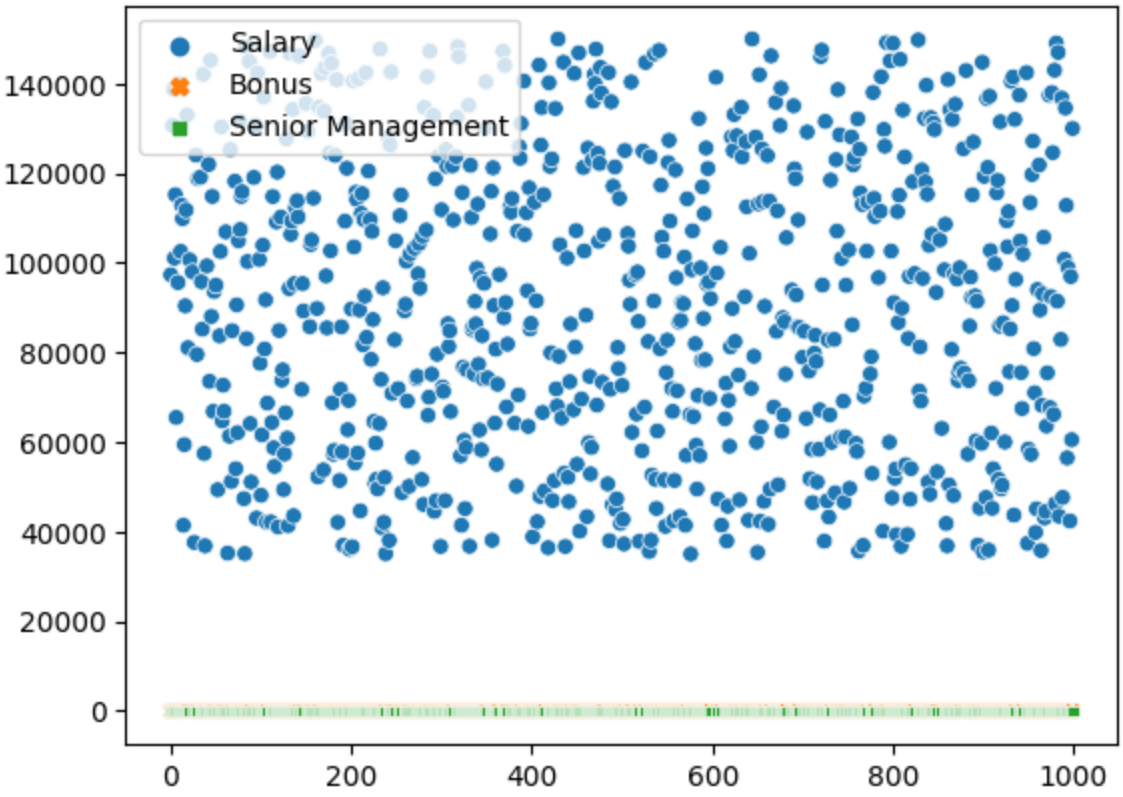
Out[17]:

	First Name	Gender	Start Date	Last Login Time	Salary	Bonus	Senior Management	Team
0	Douglas	Male	08/06/1993	12:42 pm	97308	6.945	True	Marketing
2	Maria	Female	4/23/1993	11:17 am	130590	11.858	False	Finance
3	Jerry	Male	03/04/2005	1:00 pm	138705	9.340	True	Finance
4	Larry	Male	1/24/1998	4:47 pm	101004	1.389	True	Client Services
5	Dennis	Male	4/18/1987	1:35 am	115163	10.125	False	Legal
...	...	...	...	...	...	...	...	...
994	George	Male	6/21/2013	5:47 pm	98874	4.479	True	Marketing
996	Phillip	Male	1/31/1984	6:30 am	42392	19.675	False	Finance
997	Russell	Male	5/20/2013	12:39 pm	96914	1.421	False	Product
998	Larry	Male	4/20/2013	4:45 pm	60500	11.985	False	Business Development
999	Albert	Male	5/15/2012	6:24 pm	129949	10.169	True	Sales

764 rows × 8 columns

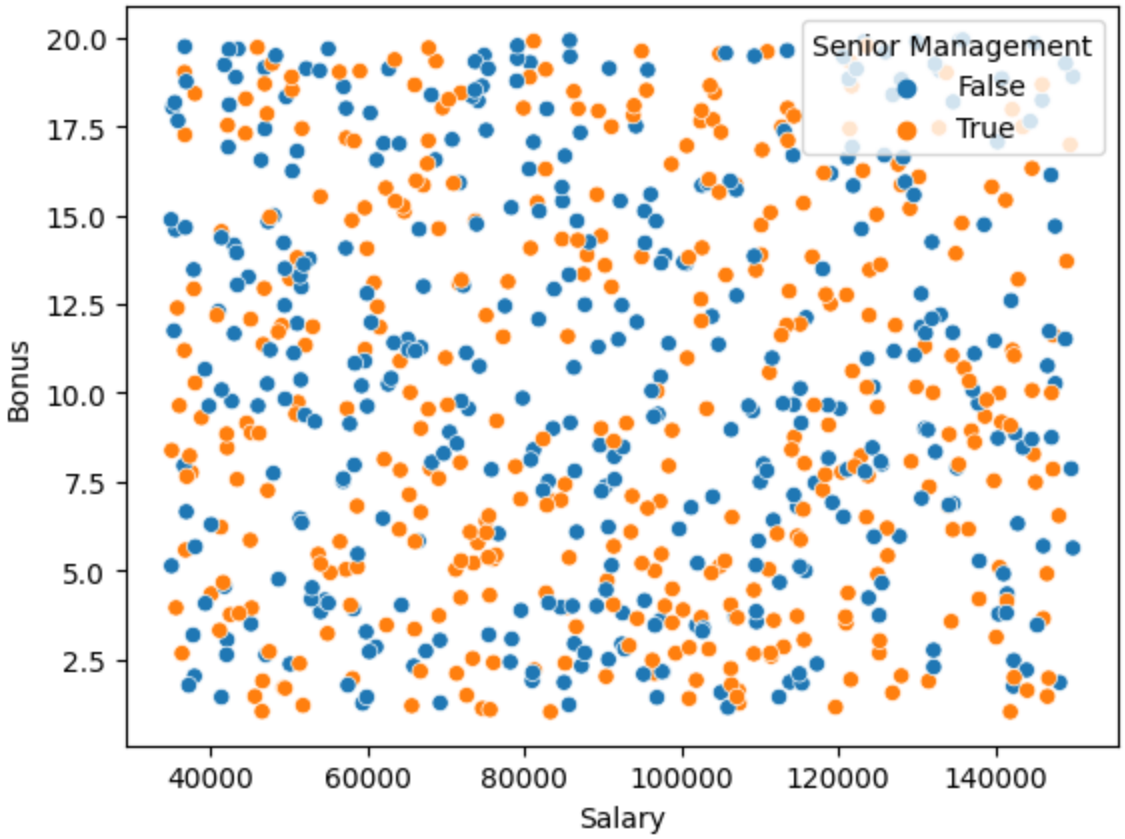
In [18]:
```
# sns.scatterplot(df['Salary'], df['Bonus%'])
sns.scatterplot(df)
```

Out[18]: <Axes: >



In [19]:
```
sns.scatterplot(x = 'Salary', y ='Bonus', data = df, hue='Senior Management')
```

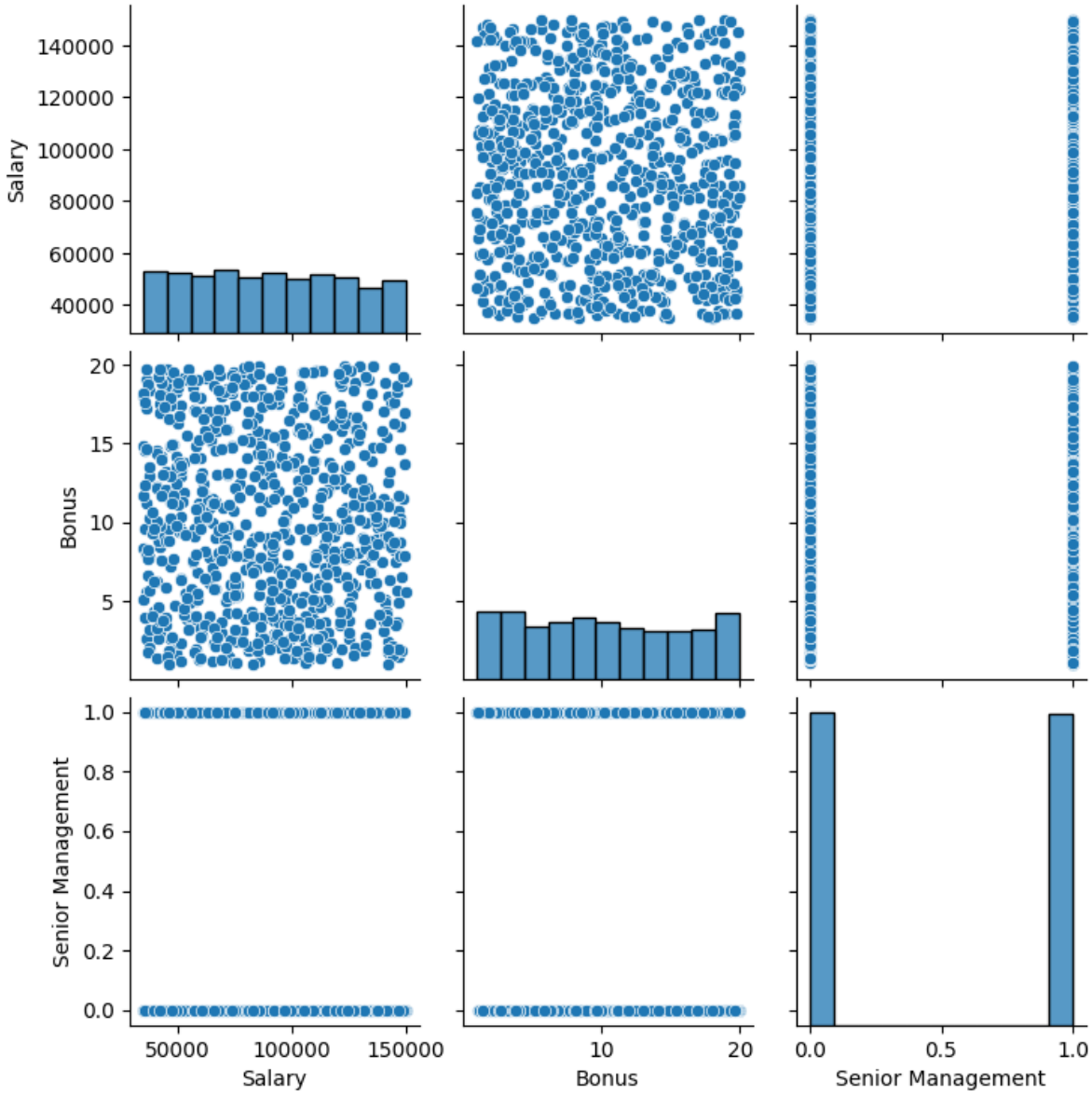
Out[19]:
```
<Axes: xlabel='Salary', ylabel='Bonus'>
```



In [20]:
```
sns.pairplot(df)
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\seaborn\_stats\counting.py:137: RuntimeWarning: Converting input from boo
l to <class 'numpy.uint8'> for compatibility.
  bin_edges = np.histogram_bin_edges(vals, bins, binrange, weight)
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\seaborn\_stats\counting.py:176: RuntimeWarning: Converting input from boo
l to <class 'numpy.uint8'> for compatibility.
  hist, edges = np.histogram(vals, **bin_kws, weights=weights, density=densit
y)
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x17fdbb03650>



In [21]:
```
import ydata_profiling
```

In [22]:
```
from ydata_profiling import ProfileReport
```

In [23]:
```
profile = ProfileReport(df, title='Profiling Report')
```

In [24]:
```
profile.to_file('report.html')
```

Summarize dataset: 100% 21/21 [00:02<00:00, 6.40it/s, Completed]

Generate report structure: 100% 1/1 [00:03<00:00, 3.54s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.71s/it]

Export report to file: 100% 1/1 [00:00<00:00, 46.92it/s]

In [25]:
```
profile.to_notebook_iframe()
```

Overview

Dataset statistics

Number of variables	8
Number of observations	764
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	53.7 KiB
Average record size in memory	72.0 B

Variable types

Text	1
Categorical	2
DateTime	2
Numeric	2
Boolean	1

Reproduction

Analysis	2023-10-31 09:58:58.040038

In [26]:
```
# x =df.drop(['First Name','Gender','Start Date','Last Login','Bonus','Senior M
x =df[['Salary']]
```

In [27]:
```
y=df['Bonus']
```

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y)
```

In [30]:
```
x_train.shape
```

Out[30]: (573, 1)

In [31]:
```
y_test.shape
```

Out[31]: (191,)

In [32]:
```
# Model Building

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

Out[32]:
```
▾ LinearRegression

LinearRegression()
```

In [33]:
```
# Training Model
model.fit(x_train, y_train)
```

Out[33]:
```
▾ LinearRegression

LinearRegression()
```

In [34]:
```
model.predict(x_test)
```

Out[34]:
```
array([ 9.76089795, 10.2514    , 10.19062668,  9.88817301,  9.92210149,
       10.39675423, 10.36344088, 10.03094175, 10.51299532, 10.616655  ,
       10.1790737 ,  9.99366848, 10.66814364,  9.90909717, 10.69533448,
       10.48547769, 10.16151355, 10.29817901,  9.76754908, 10.03956324,
       10.01084417,  9.73642715, 10.53545732, 10.50695932, 10.18338925,
       10.16904894, 10.20646638, 10.74821678, 10.44212997, 10.34645741,
       10.48797667,  9.78468633, 10.57778623, 10.70949217, 10.62628569,
       10.49660777, 10.82737683, 10.34923513, 10.55656411,  9.98536417,
       10.33820116,  9.79242356, 10.1461352 , 10.15896651,  9.90901067,
       10.77217817, 10.77274524, 10.53472685, 10.61221451, 10.5501052 ,
       10.54903833, 10.38520124, 10.56047597, 10.81383427, 10.62530532,
       10.0738954 , 10.70175494, 10.06822464, 10.27985955, 10.29304649,
       10.30426307, 10.71486498, 10.46477459,  9.81703853, 10.77542684,
       10.48764988, 10.58136169, 10.60951368, 10.00159794, 10.07007965,
       10.8221578 , 10.00732638,  9.90488735, 10.52341415, 10.82234042,
       10.19123221, 10.33419318, 10.68645348, 10.54262748, 10.14726935,
        9.8283224 , 10.3066371 , 10.81283468, 10.44604183, 10.61855807,
       10.00847014, 10.12419222, 10.28736611, 10.0955116 ,  9.77263355,
       10.00350101,  9.79320209, 10.0914075 ,  9.83579051, 10.75999083,
```

In [35]:
```
model.predict([[97308]])
```

```
C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packa
ges\sklearn\base.py:465: UserWarning: X does not have valid feature names, bu
t LinearRegression was fitted with feature names
  warnings.warn(
```

Out[35]:
```
array([10.24199037])
```

In [36]:
```
y_pred= model.predict(x_test)
y_pred
```

Out[36]:
```
array([ 9.76089795, 10.2514    , 10.19062668,  9.88817301,  9.92210149,
       10.39675423, 10.36344088, 10.03094175, 10.51299532, 10.616655  ,
       10.1790737 ,  9.99366848, 10.66814364,  9.90909717, 10.69533448,
       10.48547769, 10.16151355, 10.29817901,  9.76754908, 10.03956324,
       10.01084417,  9.73642715, 10.53545732, 10.50695932, 10.18338925,
       10.16904894, 10.20646638, 10.74821678, 10.44212997, 10.34645741,
       10.48797667,  9.78468633, 10.57778623, 10.70949217, 10.62628569,
       10.49660777, 10.82737683, 10.34923513, 10.55656411,  9.98536417,
       10.33820116,  9.79242356, 10.1461352 , 10.15896651,  9.90901067,
       10.77217817, 10.77274524, 10.53472685, 10.61221451, 10.5501052 ,
       10.54903833, 10.38520124, 10.56047597, 10.81383427, 10.62530532,
       10.0738954 , 10.70175494, 10.06822464, 10.27985955, 10.29304649,
       10.30426307, 10.71486498, 10.46477459,  9.81703853, 10.77542684,
       10.48764988, 10.58136169, 10.60951368, 10.00159794, 10.07007965,
       10.8221578 , 10.00732638,  9.90488735, 10.52341415, 10.82234042,
       10.19123221, 10.33419318, 10.68645348, 10.54262748, 10.14726935,
        9.8283224 , 10.3066371 , 10.81283468, 10.44604183, 10.61855807,
       10.00847014, 10.12419222, 10.28736611, 10.0955116 ,  9.77263355,
       10.00350101,  9.79320209, 10.0914075 ,  9.83579051, 10.75999083,
       10.45161649, 10.18881012, 10.52534606,  9.82387229, 10.15540065,
       10.10283553, 10.16746305, 10.3621145 , 10.15454523,  9.8287453 ,
       10.10677624, 10.55864018, 10.15391088,  9.7677317 , 10.04275425,
       10.05567206,  9.88790389, 10.00643251, 10.64820945,  9.85108235,
       10.40924913,  9.85678195,  9.95375205, 10.62103783, 10.46325597,
       10.2641448 ,  9.93101132, 10.69508458,  9.96323857, 10.83994863,
       10.60620734, 10.3052146 , 10.19771995,  9.81882627,  9.78016894,
       10.46884024, 10.59576928, 10.08079644,  9.83245533,  9.75366051,
       10.76639206, 10.39067017,  9.80844588, 10.2281018 , 10.6294767 ,
       10.29760232, 10.42419497, 10.39226567, 10.39859963, 10.35853903,
       10.53170885, 10.60318934,  9.98908381,  9.92511949, 10.31767107,
       10.14934543, 10.6019014 , 10.47665436, 10.45499011, 10.6832817 ,
       10.35204168, 10.21023408, 10.55090295,  9.97215802, 10.5980472 ,
       10.58042938,  9.98937215, 10.5435694 , 10.31758456, 10.23156193,
        9.89951454, 10.07809562, 10.70304288,  9.88285787, 10.03896733,
        9.85704146, 10.52210699,  9.78865587, 10.68536739, 10.68640543,
       10.66905673,  9.94659151, 10.14888408, 10.23143698, 10.82313817,
       10.78136673,  9.99222676, 10.07390502, 10.24847811, 10.16255159,
       10.48564108, 10.3143455 , 10.70148582, 10.68049437, 10.12471124,
        9.76527117])
```

In [37]:
```
# Slop(m)
model.coef_
```

Out[37]:
```
array([-9.61146802e-06])
```

In [38]:
```
# intercept
model.intercept_
```

Out[38]:
```
11.17726309851513
```

In [39]:
```
# y =mx+c
# x = 97308
# y =
# -8.56629539e-06*97308+10.926666573743816 =
```

In [40]:
```
# verify model
from sklearn.metrics import r2_score, mean_squared_error
r2_score(y_test, y_pred)
```

Out[40]: -0.015701038209921858

In [41]:
```
mean_squared_error(y_test, y_pred)
```

Out[41]: 33.78312178428922

In [42]:
```
model.predict(df[['Salary']])
```

```
       10.18227432, 10.64593153, 10.06511052, 10.08079644, 10.62430573,
        9.82138292, 10.31767107, 10.74821678, 10.11070733, 10.06822464,
       10.39226567, 10.12396154, 10.2877794 ,  9.80884956, 10.37623374,
       10.01961944, 10.12419222, 10.42356061, 10.14934543, 10.33820116,
       10.55656411, 10.68799132, 10.60318934, 10.68049437, 10.70109175,
       10.56132178,  9.7580049 , 10.46663921, 10.78475958, 10.2700078 ,
       10.77274524, 10.67669784, 10.84027542,  9.94879254, 10.81163324,
        9.96323857,  9.80704261, 10.49660777, 10.38160655, 10.16905855,
       10.48623699, 10.11534005, 10.07081012, 10.70922305, 10.31936269,
       10.3052146 , 10.21315596, 10.51278387, 10.69533448, 10.19746044,
       10.63396526, 10.1850232 , 10.46477459, 10.45958439, 10.24024108,
       10.17526756, 10.27046915, 10.6811864 , 10.15896651, 10.73436665,
        9.88346339, 10.1461352 ,  9.81703853, 10.5435694 , 10.50456606,
        9.7626184 , 10.45500933,  9.9004853 , 10.74919715, 10.03606467,
       10.72643719, 10.41236325,  9.98725763, 10.8244165 , 10.10283553,
       10.48276725, 10.4921961 , 10.7261969 ,  9.97371507, 10.0096908 ,
       10.34696682, 10.39577386, 10.3621145 , 10.53545732, 10.00847014,
       10.12471124,  9.98821878,  9.90488735,  9.75260325,  9.77410411,
       10.63012067, 10.7790792 , 10.06561993, 10.43999622, 10.59684577,
       10.74330532, 10.61254129, 10.44606106,  9.87857115, 10.82395515,
```

In [43]:
```
# Data Mining
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [44]:
```
df =pd.read_csv('car.csv')
df
```

Out[44]:

	User ID	Gender	Age	AnnualSalary	Purchased
0	385	Male	35	20000	0
1	681	Male	40	43500	0
2	353	Male	49	74000	0
3	895	Male	40	107500	1
4	661	Male	25	79000	0
...	...	...	...	...	...
995	863	Male	38	59000	0
996	800	Female	47	23500	0
997	407	Female	28	138500	1
998	299	Female	48	134000	1
999	687	Female	44	73500	0

1000 rows × 5 columns

In [45]:
```
df.sample(5)
```

Out[45]:

	User ID	Gender	Age	AnnualSalary	Purchased
333	293	Male	28	89000	0
627	632	Female	18	86000	0
923	707	Female	59	102500	1
949	731	Male	46	80500	0
176	378	Female	53	82000	1

In [46]:
```
# Data Cleaning
df.isnull().any()
```

Out[46]:
```
User ID         False
Gender          False
Age             False
AnnualSalary    False
Purchased       False
dtype: bool
```

In [47]:
```
df['Gender'].unique()
```

Out[47]:
```
array(['Male', 'Female'], dtype=object)
```

In [48]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   User ID       1000 non-null   int64
 1   Gender        1000 non-null   object
 2   Age           1000 non-null   int64
 3   AnnualSalary  1000 non-null   int64
 4   Purchased     1000 non-null   int64
dtypes: int64(4), object(1)
memory usage: 39.2+ KB
```

In [49]:
```
df.describe()
```

Out[49]:

	User ID	Age	AnnualSalary	Purchased
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	40.106000	72689.000000	0.402000
std	288.819436	10.707073	34488.341867	0.490547
min	1.000000	18.000000	15000.000000	0.000000
25%	250.750000	32.000000	46375.000000	0.000000
50%	500.500000	40.000000	72000.000000	0.000000
75%	750.250000	48.000000	90000.000000	1.000000
max	1000.000000	63.000000	152500.000000	1.000000

In [50]:
```
df.drop('User ID', axis='columns', inplace=True)
df
```

Out[50]:

	Gender	Age	AnnualSalary	Purchased
0	Male	35	20000	0
1	Male	40	43500	0
2	Male	49	74000	0
3	Male	40	107500	1
4	Male	25	79000	0
...	...	...	...	...
995	Male	38	59000	0
996	Female	47	23500	0
997	Female	28	138500	1
998	Female	48	134000	1
999	Female	44	73500	0

1000 rows × 4 columns

In [51]:
```
df['Gender'].replace({'Female':0, 'Male':1}, inplace=True)
df
```

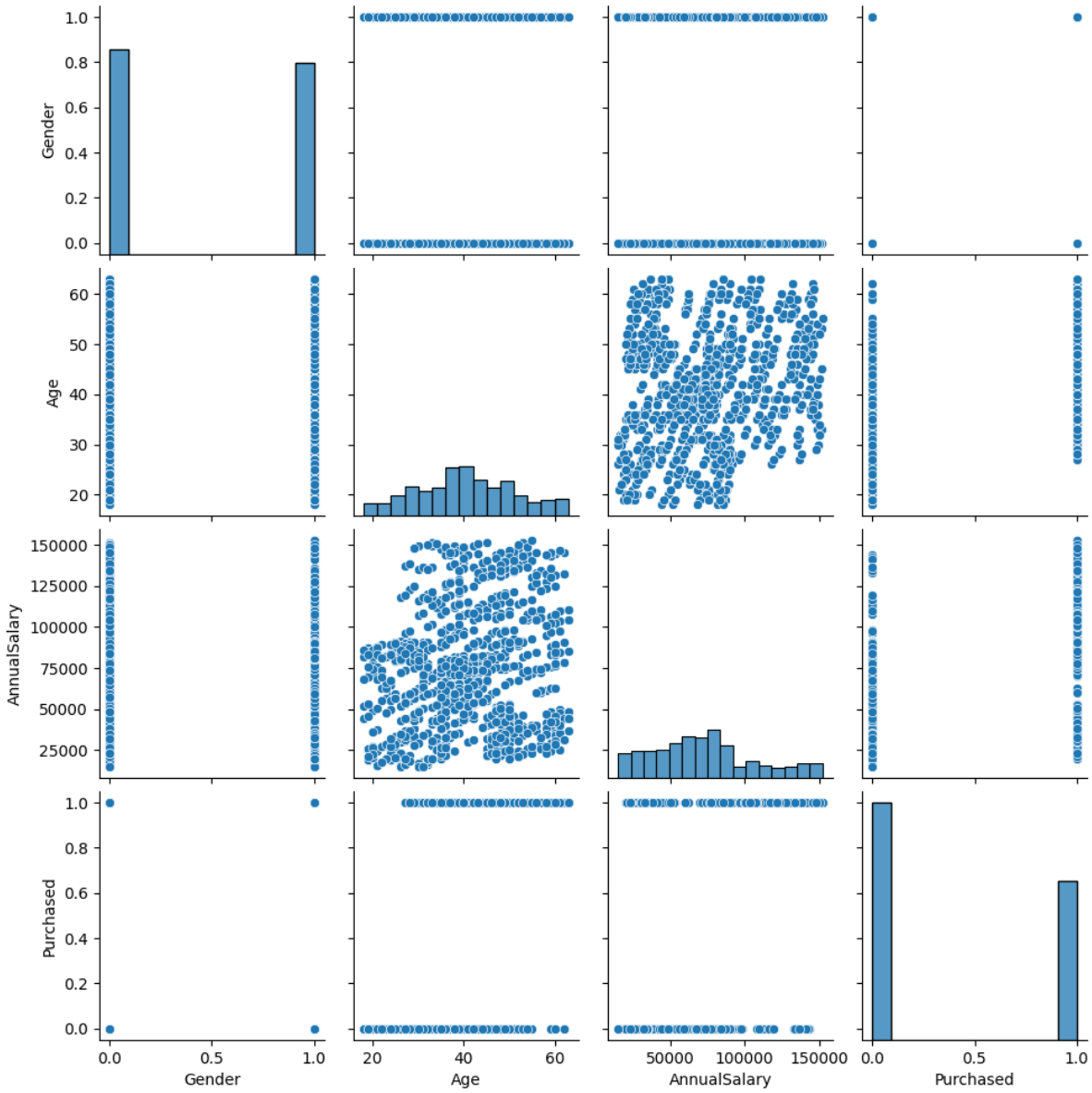
Out[51]:

	Gender	Age	AnnualSalary	Purchased
0	1	35	20000	0
1	1	40	43500	0
2	1	49	74000	0
3	1	40	107500	1
4	1	25	79000	0
...	...	...	...	...
995	1	38	59000	0
996	0	47	23500	0
997	0	28	138500	1
998	0	48	134000	1
999	0	44	73500	0

1000 rows × 4 columns

In [52]:
```
sns.pairplot(df)
```

Out[52]:
```
<seaborn.axisgrid.PairGrid at 0x17fe8ec3650>
```



In [53]:
```
df.sample()
```

Out[53]:

	Gender	Age	AnnualSalary	Purchased
875	1	40	71000	1

In [54]:
```
from sklearn.model_selection import train_test_split
```

In [55]:
```
x_test, x_train, y_test, y_train = train_test_split (df[['Gender', 'Age','Annua
```

In [56]:
```
from sklearn.linear_model import LogisticRegression
```

In [57]:
```
model =LogisticRegression()
model
```

Out[57]:
```
LogisticRegression
LogisticRegression()
```

In [58]:
```
model.fit(x_train, y_train)
```

Out[58]:
```
LogisticRegression
LogisticRegression()
```

In [59]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[59]:
```
array([0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1], dtype=int64)
```

In [60]:
```
model.score(x_test,y_test)
```

Out[60]:
```
0.372
```

In [61]:
```
model.score(x_test, y_test)
```

Out[61]:
```
0.372
```

In [62]:
```
model.coef_
```

Out[62]:
```
array([[-5.77313870e-04,  5.54036553e-03, -2.54997107e-06]])
```

In [63]:
```
model.intercept_
```

Out[63]:
```
array([-0.00079754])
```

In [64]:
```
x_train.sample()
```

Out[64]:

	Gender	Age	AnnualSalary
818	0	38	79500

In [65]:
```
def sig(a):
    import math
    return 1/(1+math.e**(-a))
```

```
b =0
c =45
d = 150000
round(pred_f(b,c,d)
```

In [66]:
```
def pred_f(b,c,d):
    m = -7.20706471e-04*b+-9.59322438e-03*c+3.4273703e-06*d+(-0.00117166)
    n =sig(m)
    return n
```

In [67]:
```
b =0
c =45
d = 150000
round(pred_f(b,c,d))
```

Out[67]:
```
1
```

In [68]:
```
b = 0
c = 49
d =880000
round(pred_f(b,c,d))
```

Out[68]:
```
1
```