Exp No:15 Experiment to understand the data preprocessing in Data science

In [1]:
```
import numpy as np
import pandas as pd
data = {
    'Name': ['John', 'Anna', 'Peter', 'Linda', 'James', 'Anna', None],
    'Age': [28, 22, np.nan, 32, 40, 22, 35],
    'City': ['New York', 'Paris', 'Berlin', 'New York', None, 'Paris', 'Berlin']
    'Salary': [50000, 54000, 58000, np.nan, 62000, 54000, 58000]
}
df=pd.DataFrame(data)
df.to_csv('emp.csv',index=False)
df=pd.read_csv('emp.csv')
df
```

Out[1]:

	Name	Age	City	Salary
0	John	28.0	New York	50000.0
1	Anna	22.0	Paris	54000.0
2	Peter	NaN	Berlin	58000.0
3	Linda	32.0	New York	NaN
4	James	40.0	NaN	62000.0
5	Anna	22.0	Paris	54000.0
6	NaN	35.0	Berlin	58000.0

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Name    6 non-null      object 
 1   Age     6 non-null      float64
 2   City    6 non-null      object 
 3   Salary  6 non-null      float64
dtypes: float64(2), object(2)
memory usage: 356.0+ bytes
```

In [4]:
```
df.City.mode()
```

Out[4]:
```
0      Berlin
1    New York
2       Paris
Name: City, dtype: object
```

In [5]:
```
df.City.mode()[0]
```

Out[5]:
```
'Berlin'
```

In [10]:
```
df.City.fillna(df.City.mode()[0],inplace=True)
df.Age.fillna(df.Age.median(),inplace=True)
```

```
df.Salary.fillna(round(df.Salary.mean()),inplace=True)
df
```

```
C:\Users\Kaviya\AppData\Local\Temp\ipykernel_21492\4129364907.py:2: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df.Age.fillna(df.Age.median(),inplace=True)
C:\Users\Kaviya\AppData\Local\Temp\ipykernel_21492\4129364907.py:3: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df.Salary.fillna(round(df.Salary.mean()),inplace=True)
```

Out[10]:

	Name	Age	City	Salary
0	John	28.0	New York	50000.0
1	Anna	22.0	Paris	54000.0
2	Peter	30.0	Berlin	58000.0
3	Linda	32.0	New York	56000.0
4	James	40.0	Berlin	62000.0
5	Anna	22.0	Paris	54000.0
6	NaN	35.0	Berlin	58000.0

In [11]:
```
pd.get_dummies(df.City)
```

Out[11]:

	Berlin	New York	Paris
0	False	True	False
1	False	False	True
2	True	False	False
3	False	True	False
4	True	False	False
5	False	False	True
6	True	False	False

In [15]:
```
updated_dataset=pd.concat([pd.get_dummies(df.City),df.iloc[:,[1,2,3]]],axis=1)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Name    6 non-null      object 
 1   Age     7 non-null      float64
 2   City    7 non-null      object 
 3   Salary  7 non-null      float64
dtypes: float64(2), object(2)
memory usage: 356.0+ bytes
```

In [16]:
```
updated_dataset
```

Out[16]:

	Berlin	New York	Paris	Age	City	Salary
0	False	True	False	28.0	New York	50000.0
1	False	False	True	22.0	Paris	54000.0
2	True	False	False	30.0	Berlin	58000.0
3	False	True	False	32.0	New York	56000.0
4	True	False	False	40.0	Berlin	62000.0
5	False	False	True	22.0	Paris	54000.0
6	True	False	False	35.0	Berlin	58000.0

EXPERIMENT-15

In [ ]:
```
Experiment to understand EDA-Quantitative and
Qualitative analysis.
```

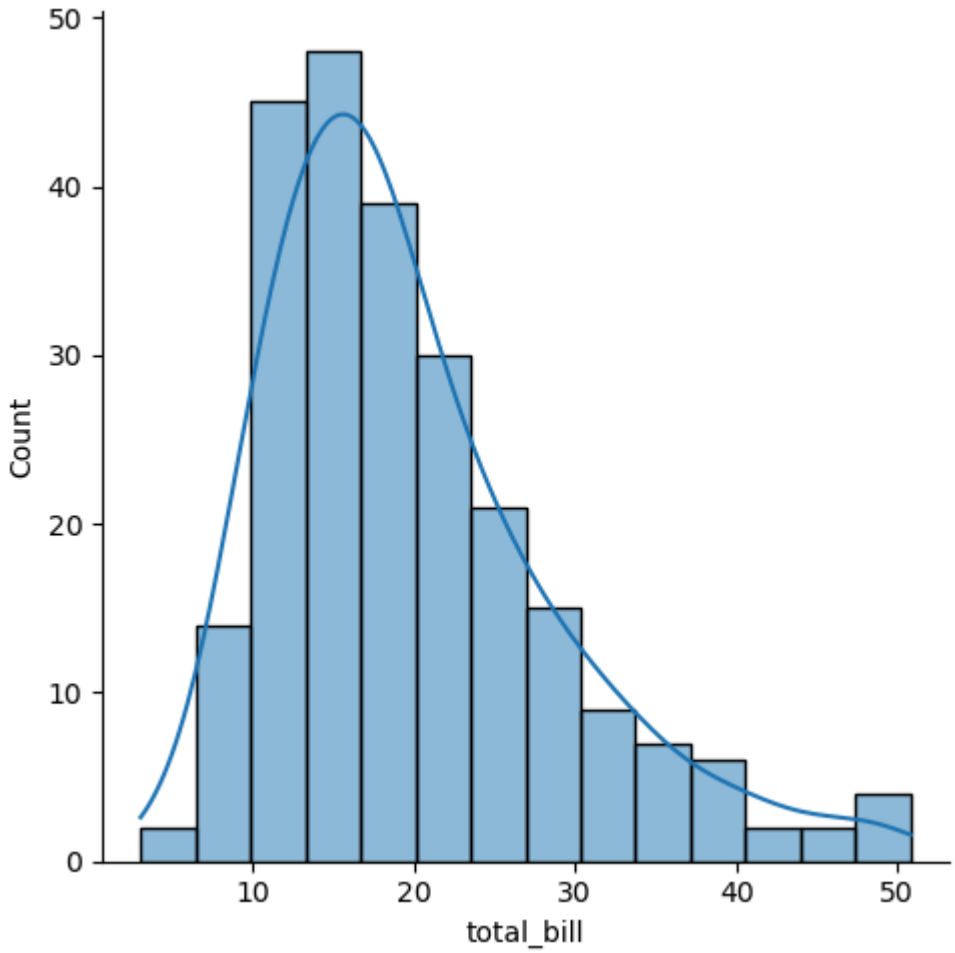
In [18]:
```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
tips=sns.load_dataset('tips')
tips.head()
```

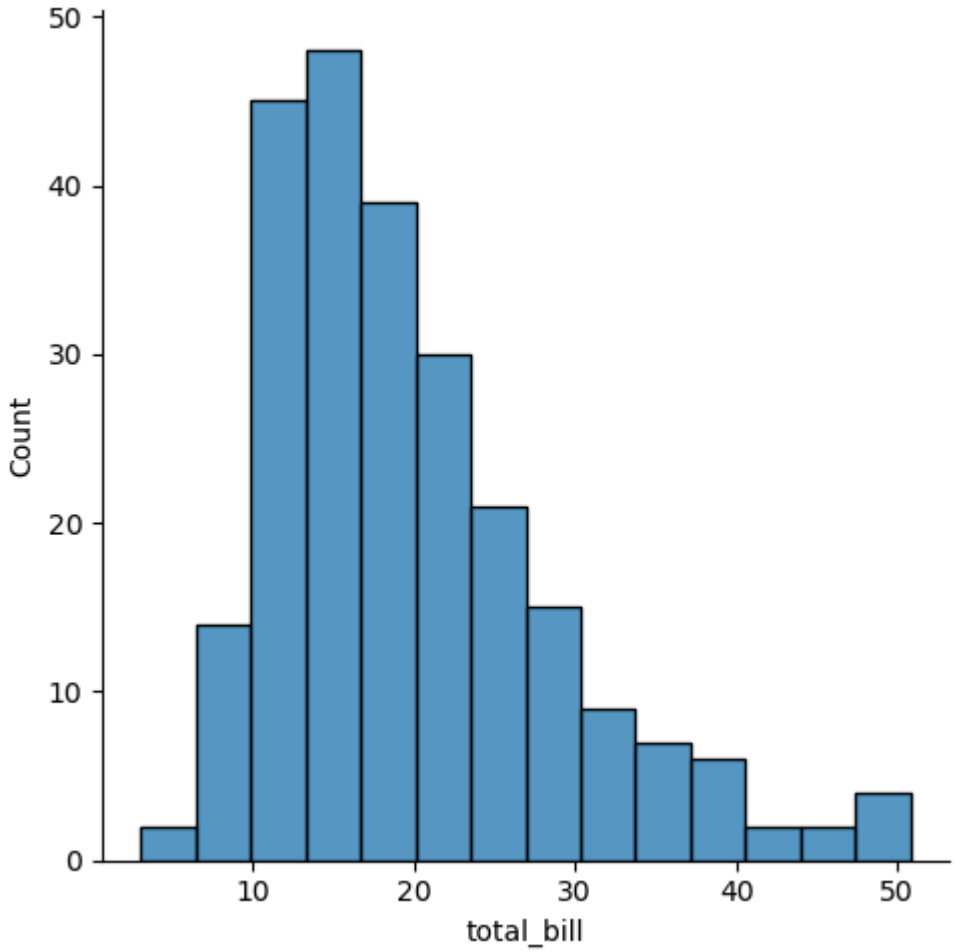
Out[18]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

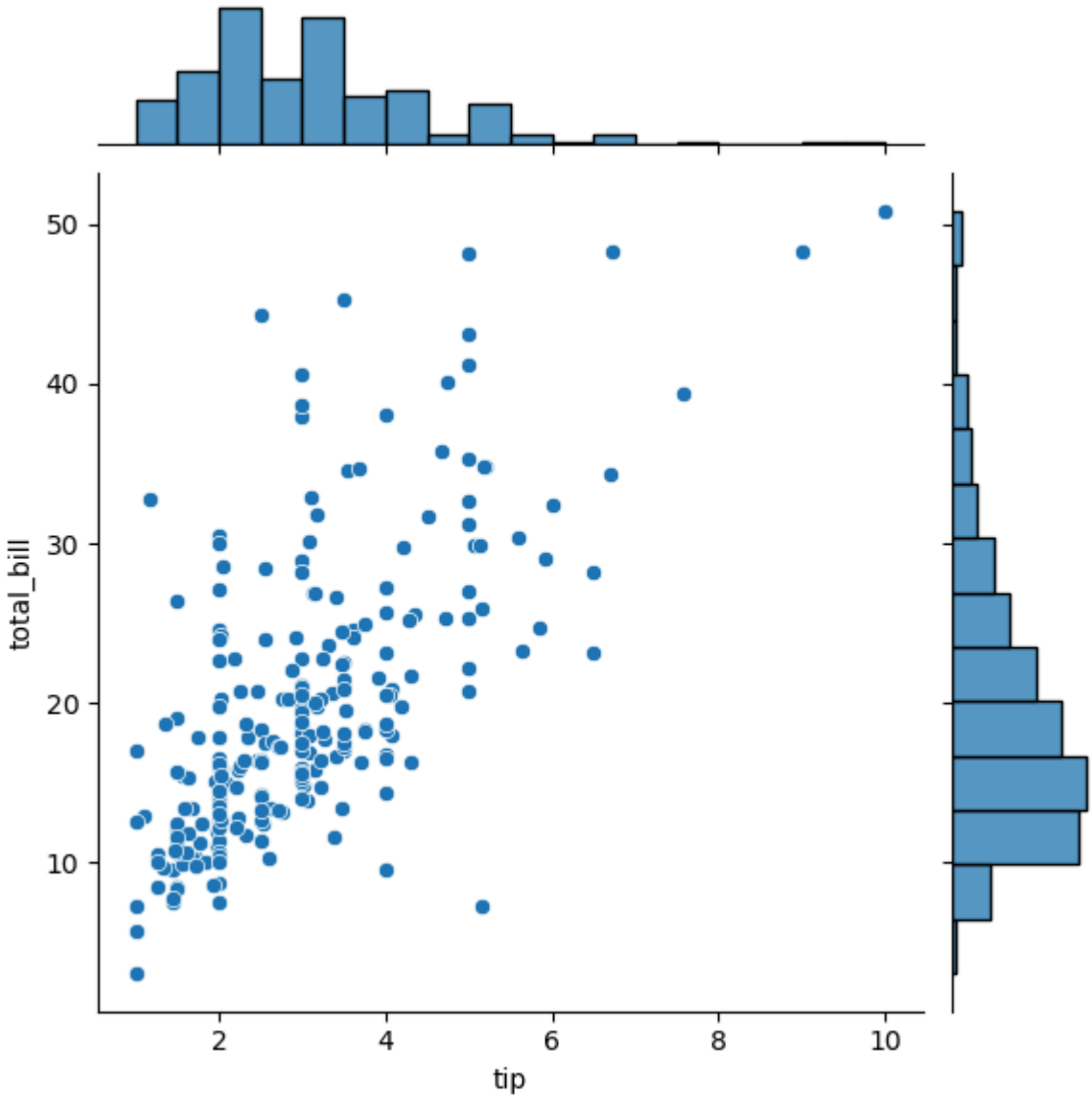
In [21]:
```
sns.displot(tips.total_bill,kde=True)
plt.show()
```



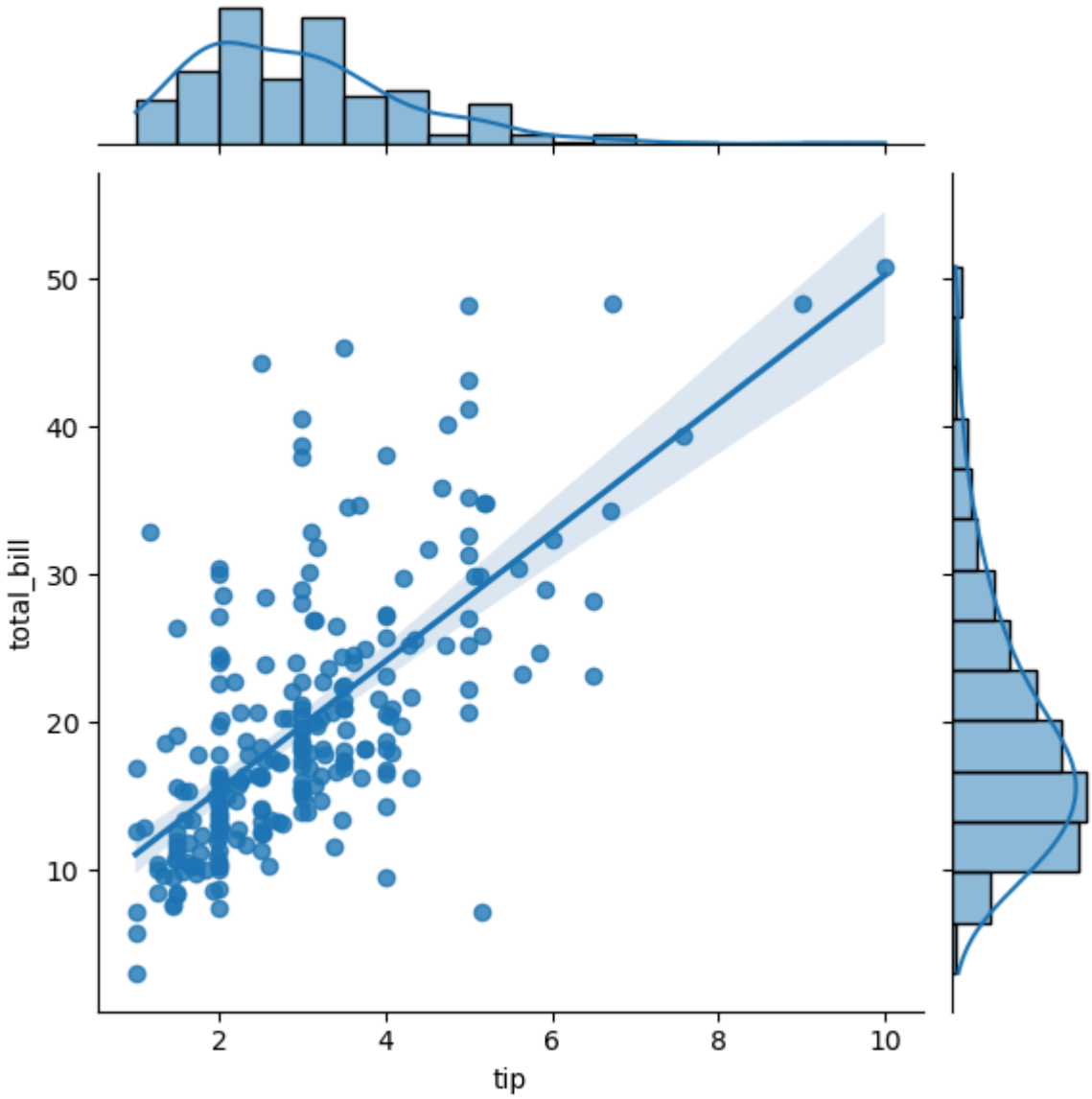
In [22]:
```
sns.displot(tips.total_bill,kde=False)
plt.show()
```



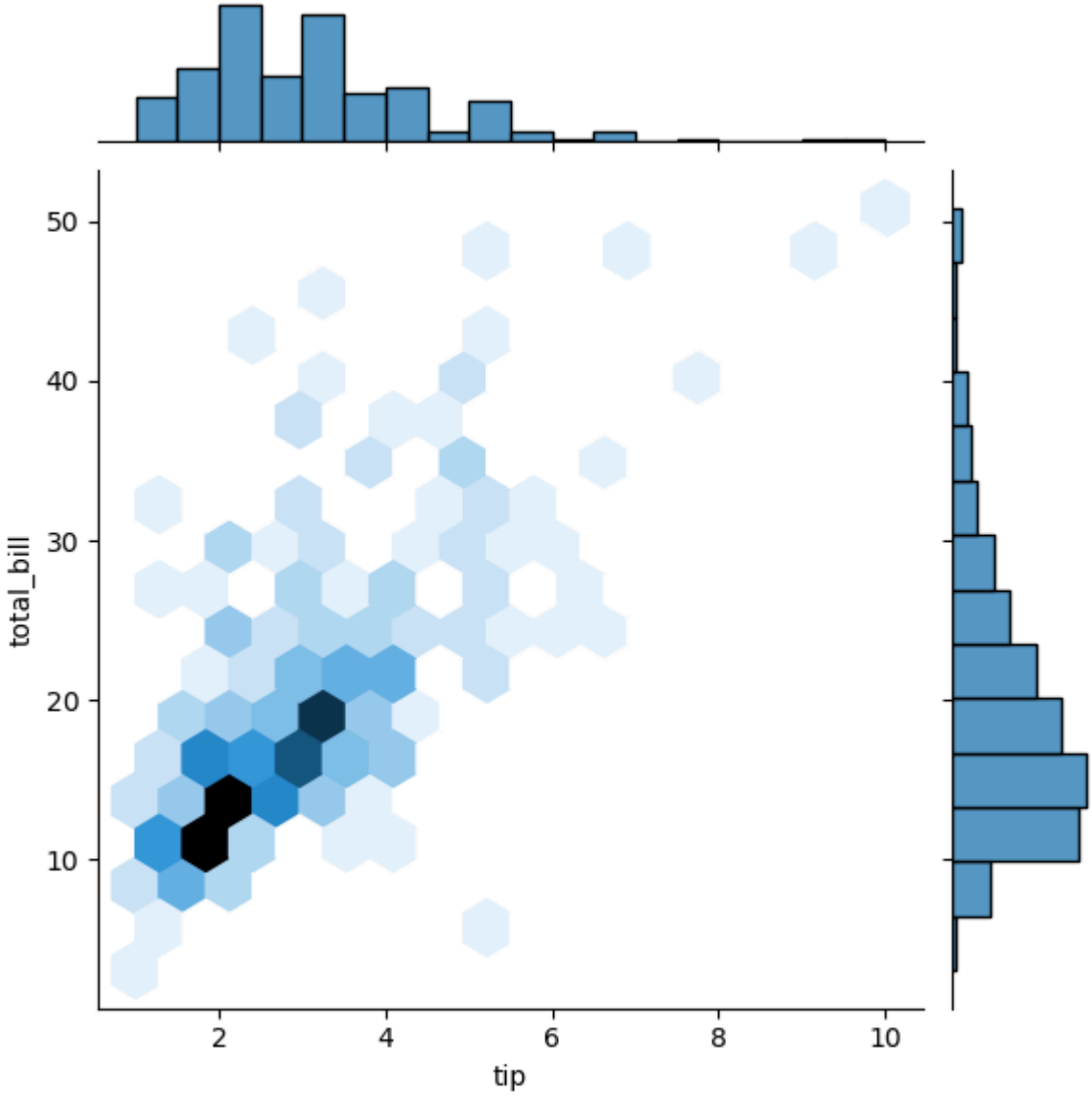
In [23]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill)
plt.show()
```



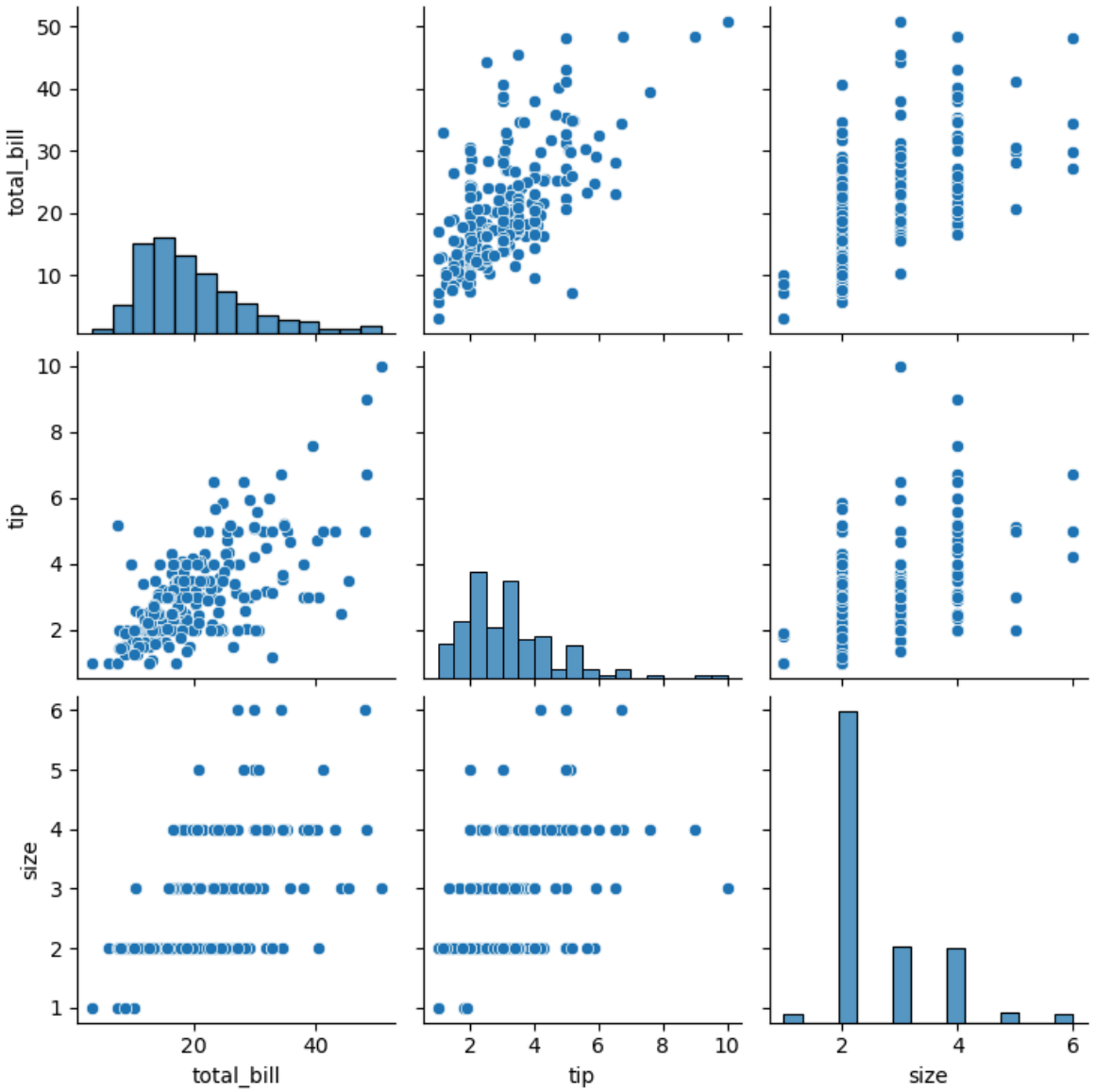
In [34]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
plt.show()
```



In [28]:
```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
plt.show()
```



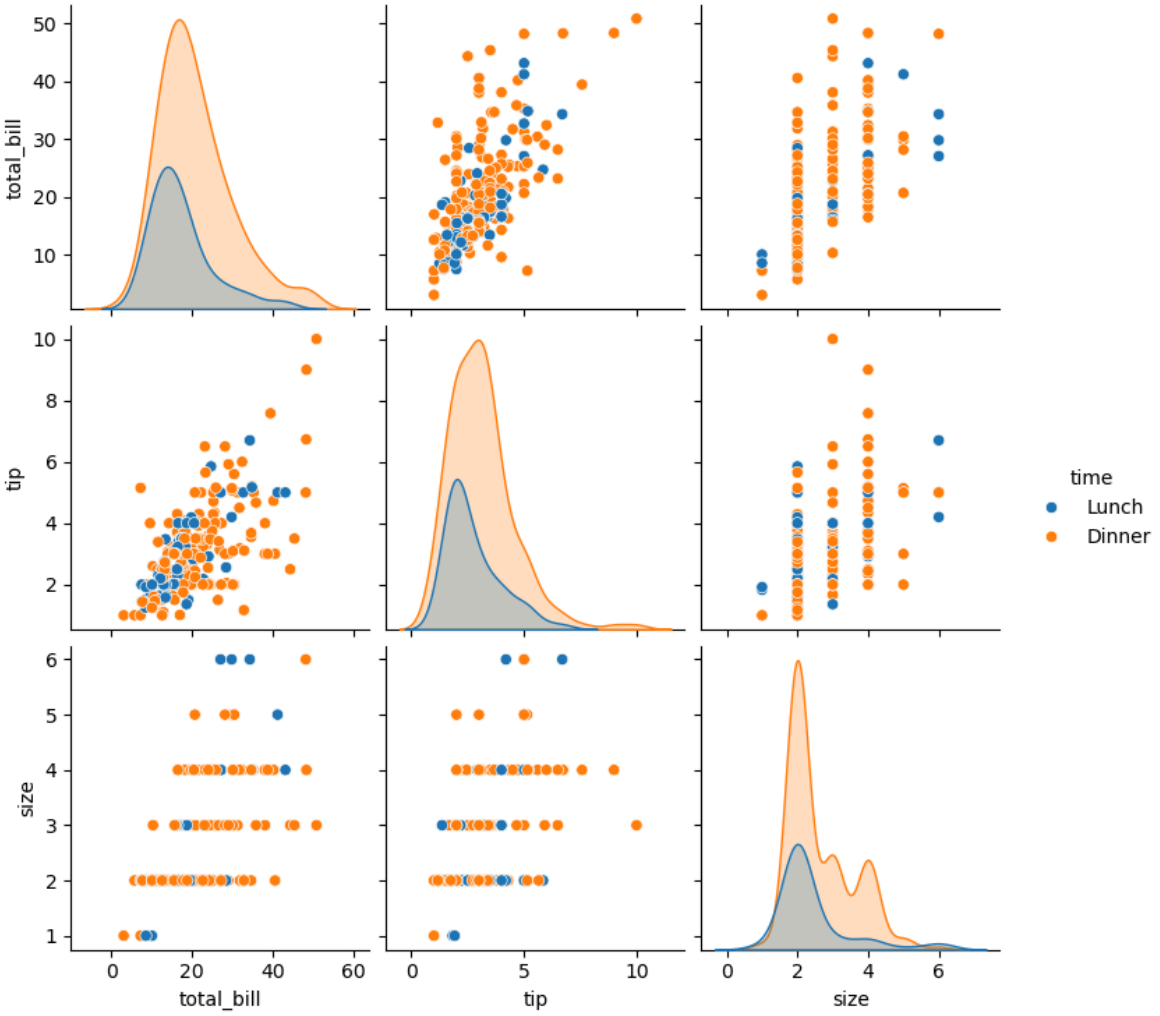
In [29]:
```
sns.pairplot(tips)
plt.show()
```



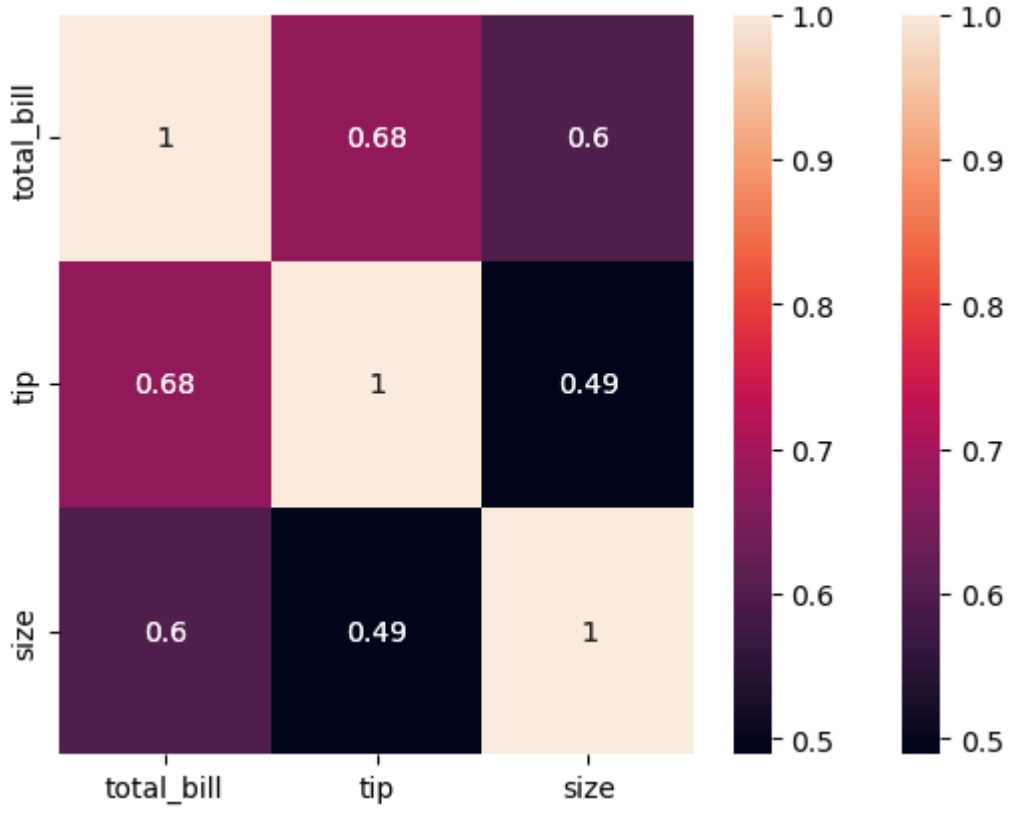
In [30]:
```
tips.time.value_counts()
```

Out[30]:
```
time
Dinner    176
Lunch      68
Name: count, dtype: int64
```

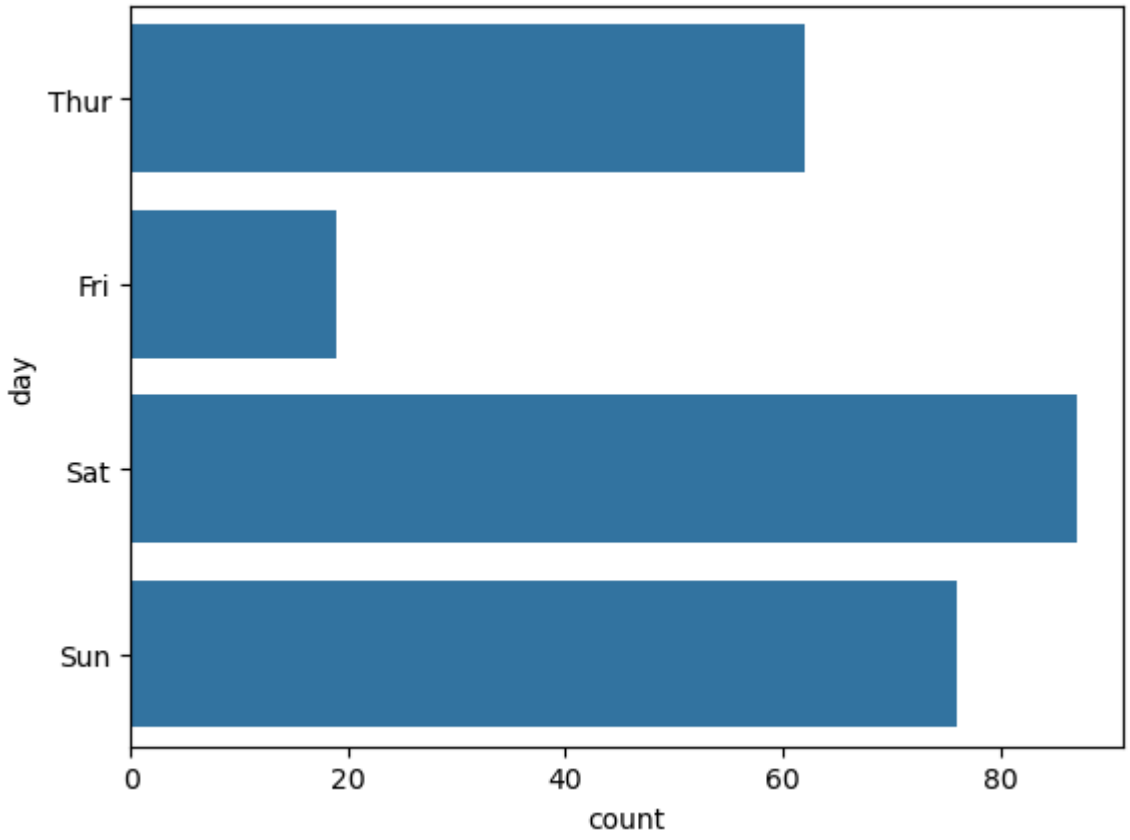
In [33]:
```
sns.pairplot(tips,hue='time')
plt.show()
```



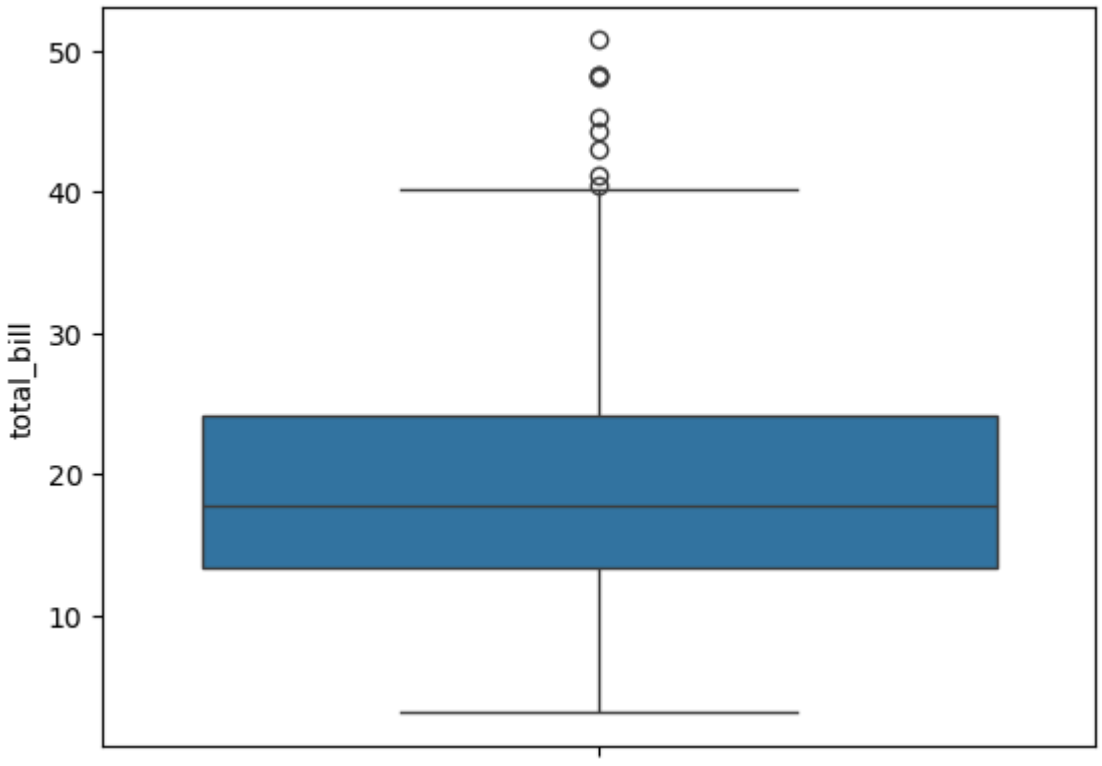
In [36]:
```
sns.heatmap(tips.corr(numeric_only=True),annot=True)
plt.show()
```



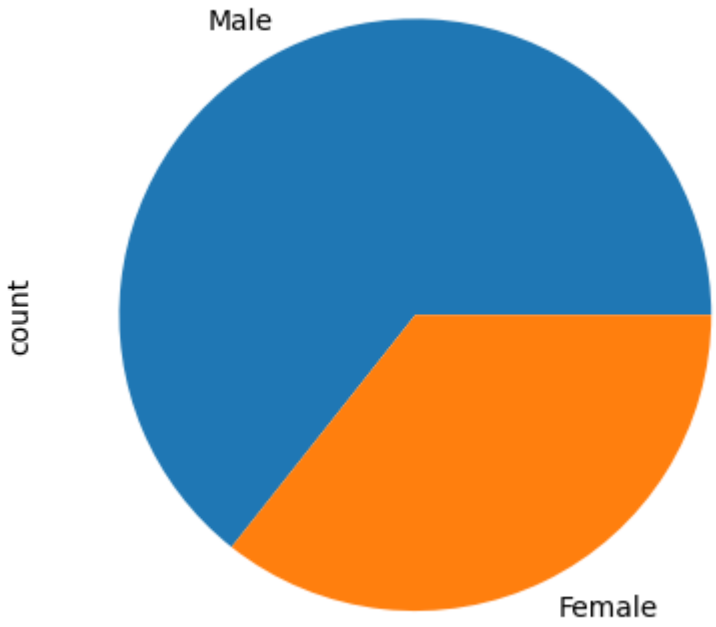
In [37]:
```
sns.countplot(tips.day)
plt.show()
```



In [38]:
```
sns.boxplot(tips.total_bill)
plt.show()
```



In [39]:
```
tips.sex.value_counts().plot(kind='pie')
plt.show()
```



EXPERIMENT-17

In [ ]:
```
Experiment to understand Linear Regression for a given
data set.
```

In [42]:
```
import numpy as np
import pandas as pd
data = {
    'EmployeeID': [1,2,3,4,5,6,7,8,9,10],
    'Name': ['John Smith','Priya Sharma','David Lee','Sarah Johnson','Karan Pate
             'Aisha Khan','Michael Brown','Meena Reddy','Rajesh Gupta','Emily Da
    'Gender': ['Male','Female','Male','Female','Male','Female','Male','Female','
    'Age': [28,32,45,29,38,26,50,35,31,40],
    'Department': ['IT','HR','Finance','IT','Marketing','HR','Management','Finan
    'Experience': [3,6,20,4,12,2,25,10,5,15],
    'Education': ['Bachelor','Master','Master','Bachelor','MBA','Bachelor','MBA'
    'Salary': [45000,58000,95000,52000,74000,40000,120000,83000,60000,88000]
}
df=pd.DataFrame(data)
df.to_csv('sal_data.csv',index=False)
df=pd.read_csv('sal_data.csv')
df
```

Out[42]:

	EmployeeID	Name	Gender	Age	Department	Experience	Education	Salary
0	1	John Smith	Male	28	IT	3	Bachelor	45000
1	2	Priya Sharma	Female	32	HR	6	Master	58000
2	3	David Lee	Male	45	Finance	20	Master	95000
3	4	Sarah Johnson	Female	29	IT	4	Bachelor	52000
4	5	Karan Patel	Male	38	Marketing	12	MBA	74000
5	6	Aisha Khan	Female	26	HR	2	Bachelor	40000
6	7	Michael Brown	Male	50	Management	25	MBA	120000
7	8	Meena Reddy	Female	35	Finance	10	Master	83000
8	9	Rajesh Gupta	Male	31	IT	5	Bachelor	60000
9	10	Emily Davis	Female	40	Marketing	15	MBA	88000

In [43]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   EmployeeID  10 non-null     int64 
 1   Name        10 non-null     object
 2   Gender      10 non-null     object
 3   Age         10 non-null     int64 
 4   Department  10 non-null     object
 5   Experience  10 non-null     int64 
 6   Education   10 non-null     object
 7   Salary      10 non-null     int64 
dtypes: int64(4), object(4)
memory usage: 772.0+ bytes
```

In [44]:
```
df.dropna(inplace=True)
```

In [45]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   EmployeeID  10 non-null     int64 
 1   Name        10 non-null     object
 2   Gender      10 non-null     object
 3   Age         10 non-null     int64 
 4   Department  10 non-null     object
 5   Experience  10 non-null     int64 
 6   Education   10 non-null     object
 7   Salary      10 non-null     int64 
dtypes: int64(4), object(4)
memory usage: 772.0+ bytes
```

In [46]:
```
df.describe()
```

Out[46]:

	EmployeeID	Age	Experience	Salary
count	10.00000	10.000000	10.000000	10.000000
mean	5.50000	35.400000	10.200000	71500.000000
std	3.02765	7.805981	7.771744	25176.046817
min	1.00000	26.000000	2.000000	40000.000000
25%	3.25000	29.500000	4.250000	53500.000000
50%	5.50000	33.500000	8.000000	67000.000000
75%	7.75000	39.500000	14.250000	86750.000000
max	10.00000	50.000000	25.000000	120000.000000

In [56]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
data = {
    'EmployeeID': [1,2,3,4,5,6,7,8,9,10],
    'Name': ['John Smith','Priya Sharma','David Lee','Sarah Johnson','Karan Pate
             'Aisha Khan','Michael Brown','Meena Reddy','Rajesh Gupta','Emily Da
    'Gender': ['Male','Female','Male','Female','Male','Female','Male','Female','
    'Age': [28,32,45,29,38,26,50,35,31,40],
    'Department': ['IT','HR','Finance','IT','Marketing','HR','Management','Finan
    'Experience': [3,6,20,4,12,2,25,10,5,15],
    'Education': ['Bachelor','Master','Master','Bachelor','MBA','Bachelor','MBA'
    'Salary': [45000,58000,95000,52000,74000,40000,120000,83000,60000,88000]
}
df = pd.DataFrame(data)
features = df[['Experience']].values
label = df[['Salary']].values
x_train, x_test, y_train, y_test = train_test_split(features, label, test_size=0
model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Training complete ")
print("\nTest Data (Experience):")
print(x_test)
```

```
print("\nPredicted Salaries:")
print(y_pred)
```

```
Training complete

Test Data (Experience):
[[5]
 [6]]

Predicted Salaries:
[[54108.37817064]
 [57326.67179093]]
```

In [57]:
```
model.score(x_train, y_train)
```

Out[57]: 0.9512880612893012

In [58]:
```
model.score(x_test, y_test)
```

Out[58]: -16.58228932867233

In [59]:
```
model.coef_
```

Out[59]: array([[3218.29362029]])

In [60]:
```
model.intercept_
```

Out[60]: array([38016.91006918])

Kmeans clustering

In [63]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
data = {
    'CustomerID': [1, 2, 3, 4, 5],
    'Gender': ['Male', 'Male', 'Female', 'Female', 'Female'],
    'Age': [19, 21, 20, 23, 31],
    'Annual Income (k$)': [15, 15, 16, 16, 17],
    'Spending Score (1-100)': [39, 81, 6, 77, 40]
}
df.to_csv('cust_data.csv',index=False)
df=pd.read_csv('cust_data.csv')
df.info()
```

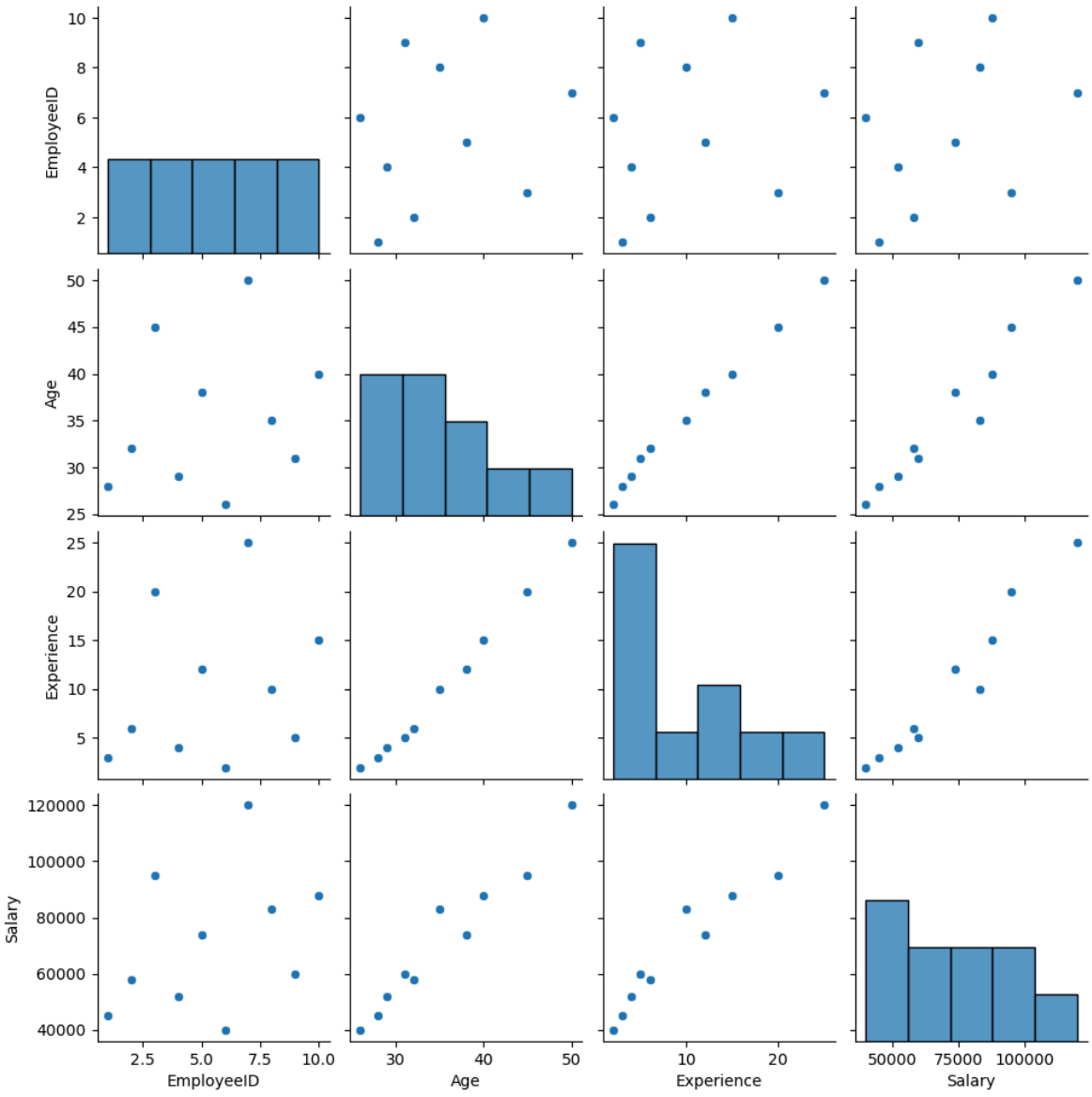
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   EmployeeID  10 non-null     int64
 1   Name        10 non-null     object
 2   Gender      10 non-null     object
 3   Age         10 non-null     int64
 4   Department  10 non-null     object
 5   Experience  10 non-null     int64
 6   Education   10 non-null     object
 7   Salary      10 non-null     int64
dtypes: int64(4), object(4)
memory usage: 772.0+ bytes
```

In [64]:
```
df.head()
```

Out[64]:

	EmployeeID	Name	Gender	Age	Department	Experience	Education	Salary
0	1	John Smith	Male	28	IT	3	Bachelor	45000
1	2	Priya Sharma	Female	32	HR	6	Master	58000
2	3	David Lee	Male	45	Finance	20	Master	95000
3	4	Sarah Johnson	Female	29	IT	4	Bachelor	52000
4	5	Karan Patel	Male	38	Marketing	12	MBA	74000

In [68]:
```
sns.pairplot(df)
plt.show()
```


EmployeeID
Age
Experience
Salary
EmployeeID
Age
Experience
Salary

In [ ]: