IMPORTANT PLOT IN DATA ANALYSIS

DATA ANALYSIS

Seaborn is a data visualization library based on MATPLOTLIB. it provides a high-level interface for drawing attractive and informative statistical graphics.

1.Importance plot in data visualization every data science student should know this:

A) Count plot
B) Bar plot
C) Box plot
D) Violin plot
E) Strip plot
F) Swarm plot
G) Factor plot
H) Dis plot
I) Join plot
K) Pair plot
L) Rug plot
M) Heatmap
N) Scatter plot

In [ ]:

```
```

2.Importing all the required libraries

In [1]:

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

Import the data

In [2]:

```
mart = pd.read_excel(r"supermarket_sales.xlsx")
mart.shape
```

Out[2]:

```
(1000, 17)
```

In [3]:

```
mart.columns
```

Out[3]:

```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'G
ender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%',
'Total', 'Date',
       'Time', 'Payment', 'cogs', 'gross margin percentag
e', 'gross income',
       'Rating'],
      dtype='object')
```

In [4]:

```
mart.head()
```

Out[4]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	
0	692-92-5582	B	Mandalay	Member	Female	Food_and_beverages	5
1	351-62-0822	B	Mandalay	Member	Female	Fashion_accessories	1
2	529-56-3974	B	Mandalay	Member	Male	Electronic_accessories	2
3	299-46-1805	B	Mandalay	Member	Female	Sports_and_travel	9
4	319-50-3348	B	Mandalay	Normal	Female	Home_and_lifestyle	4

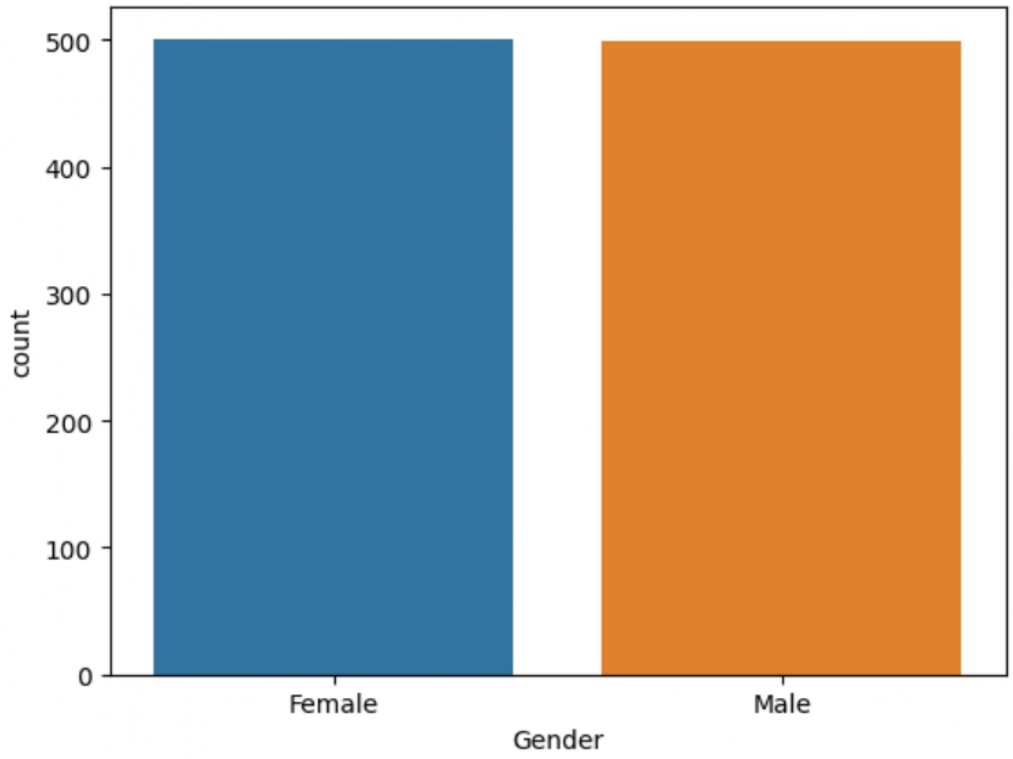
4.create a countplot

In [5]:

```
sns.countplot(x= "Gender",data = mart)
```

Out[5]:

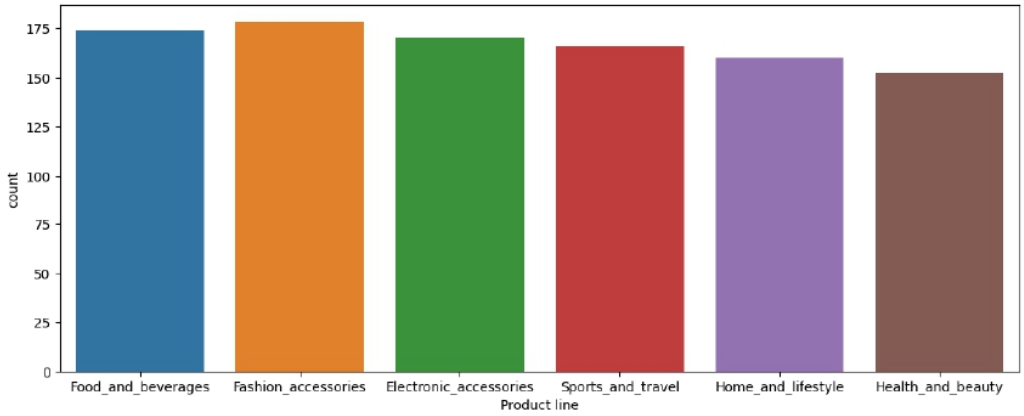
```
<AxesSubplot: xlabel='Gender', ylabel='count'>
```



5.create a basic COUNTPLOT to get number of trensaction for each of the product line and change the figure size

In [6]:

```
plt.figure(figsize = (13,5))
sns.countplot(x = "Product line",data = mart)
plt.show()
```



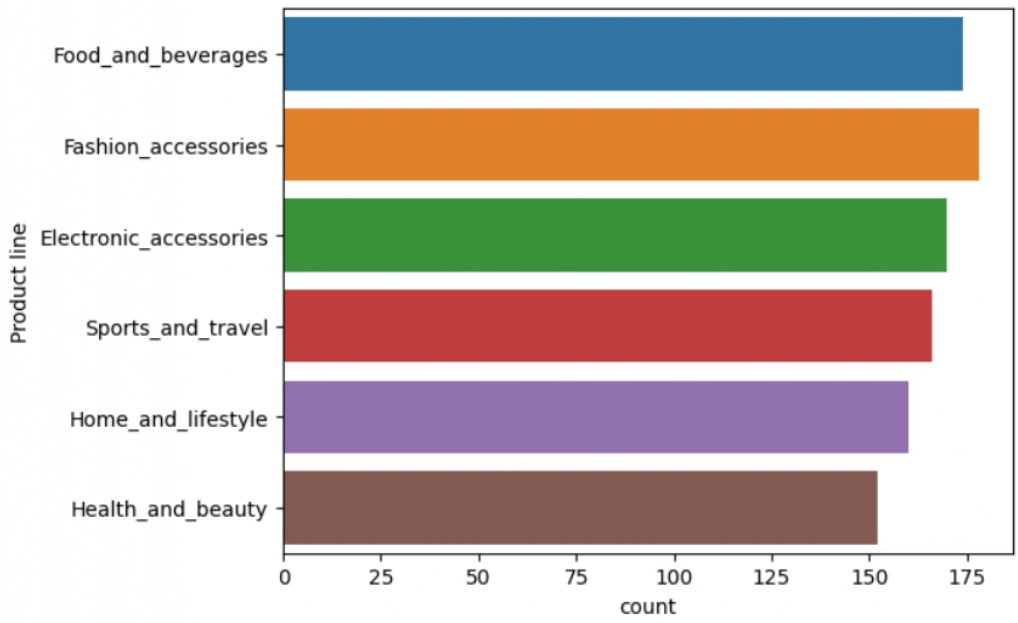
6.Make it horizontal bar plot

In [7]:

```
sns.countplot(y = "Product line",data = mart)
```

Out[7]:

```
<AxesSubplot: xlabel='count', ylabel='Product line'>
```



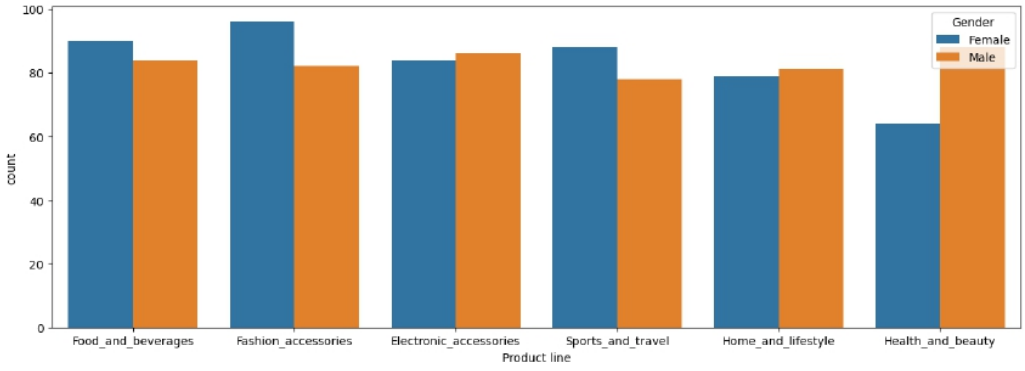
7.Add hue get the count on two catagories that is product line and gender

In [8]:

```
plt.figure(figsize = (15,5))
sns.countplot(x = "Product line",hue = "Gender",data = mart)
```

Out[8]:

```
<AxesSubplot: xlabel='Product line', ylabel='count'>
```



B)1.Bar plot

In [9]:

```
sns.get_dataset_names()
```

Out[9]:

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

2.Import file

In [10]:

```
titanic = pd.read_csv("train_and_test2.csv")
```

In [11]:

```
titanic.head()
```

Out[11]:

	Passengerid	Age	Fare	Sex	sibsp	zero	zero.1	zero.2	zero.3	z
0	1	22.0	7.2500	0	1	0	0	0	0	
1	2	38.0	71.2833	1	1	0	0	0	0	
2	3	26.0	7.9250	1	0	0	0	0	0	
3	4	35.0	53.1000	1	1	0	0	0	0	
4	5	35.0	8.0500	0	0	0	0	0	0	

5 rows × 28 columns

In [12]:

```
titanic.columns
```

Out[12]:

```
Index(['Passengerid', 'Age', 'Fare', 'Sex', 'sibsp', 'zer
o', 'zero.1',
       'zero.2', 'zero.3', 'zero.4', 'zero.5', 'zero.6',
'Parch', 'zero.7',
       'zero.8', 'zero.9', 'zero.10', 'zero.11', 'zero.1
2', 'zero.13',
       'zero.14', 'Pclass', 'zero.15', 'zero.16', 'Embarke
d', 'zero.17',
       'zero.18', '2urvived'],
      dtype='object')
```

In [13]:

```
titanic.index
```

Out[13]:

```
RangeIndex(start=0, stop=1309, step=1)
```

In [14]:

```
mart = pd.read_excel(r"supermarket_sales.xlsx")
```

In [15]:

```
mart.head()
```

Out[15]:

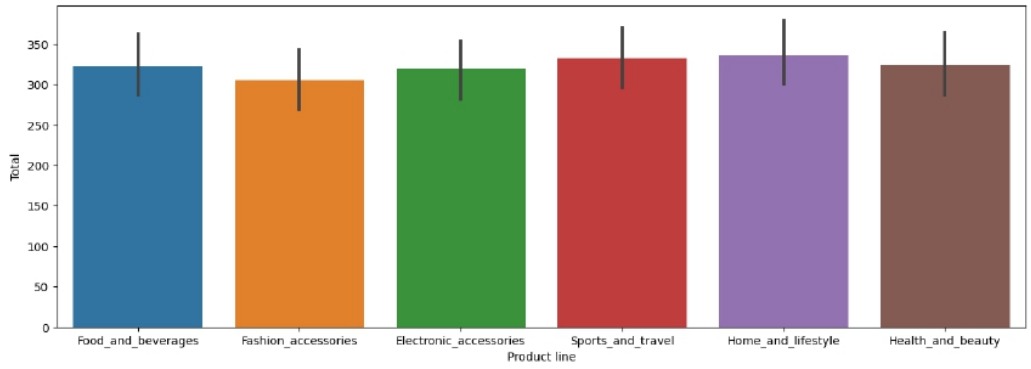
	Invoice ID	Branch	City	Customer type	Gender	Product line	
0	692-92-5582	B	Mandalay	Member	Female	Food_and_beverages	5
1	351-62-0822	B	Mandalay	Member	Female	Fashion_accessories	1
2	529-56-3974	B	Mandalay	Member	Male	Electronic_accessories	2
3	299-46-1805	B	Mandalay	Member	Female	Sports_and_travel	9
4	319-50-3348	B	Mandalay	Normal	Female	Home_and_lifestyle	4

In [16]:

```
plt.figure(figsize =(15,5))
sns.barplot(x = "Product line",y = "Total",data = mart)
```

Out[16]:

```
<AxesSubplot: xlabel='Product line', ylabel='Total'>
```



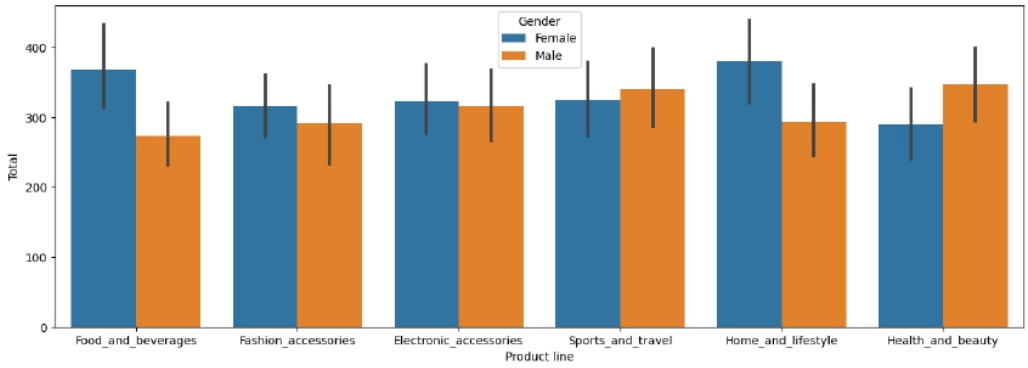
3.Add the HUE the barplot

In [17]:

```
plt.figure(figsize =(15,5))
sns.barplot(x = "Product line",y = "Total",hue ="Gender" ,data = mart)
```

Out[17]:

```
<AxesSubplot: xlabel='Product line', ylabel='Total'>
```



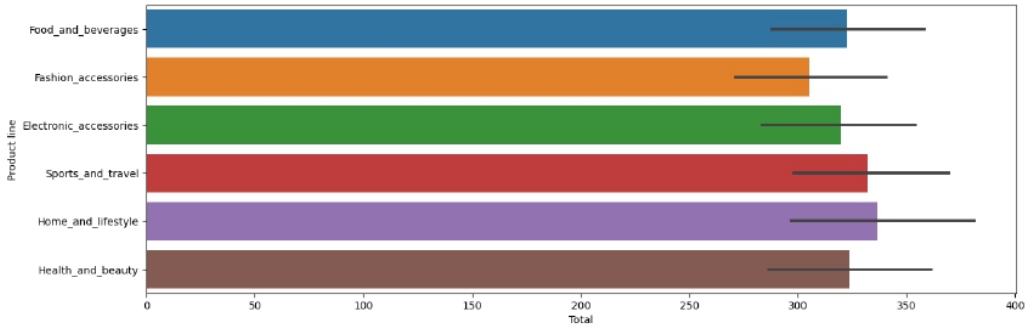
4.Make the barplot HORIZONTAL

In [18]:

```
plt.figure(figsize =(15,5))
sns.barplot(y = "Product line",x= "Total",data = mart)
```

Out[18]:

```
<AxesSubplot: xlabel='Total', ylabel='Product line'>
```



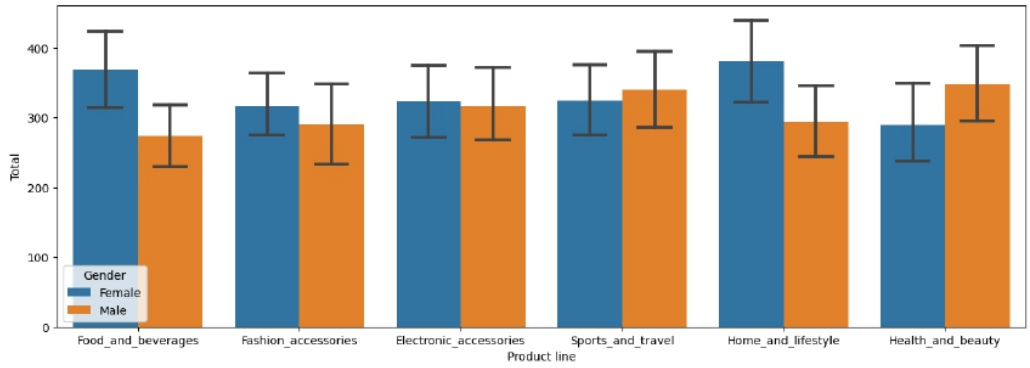
5.Ad cap on the ERROR BAR

In [19]:

```
plt.figure(figsize =(15,5))
sns.barplot(x = "Product line",y = "Total",hue ="Gender" ,capsize = 0.2,c
```

Out[19]:

```
<AxesSubplot: xlabel='Product line', ylabel='Total'>
```



C) BOXPLOT

In [ ]:

```

```

1.creat a basic BOX plot one NUMERIC veriable

In [20]:

```
mart = pd.read_excel(r"supermarket_sales.xlsx")
```

In [21]:

```
mart.head()
```

Out[21]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	
0	692-92-5582	B	Mandalay	Member	Female	Food_and_beverages	5
1	351-62-0822	B	Mandalay	Member	Female	Fashion_accessories	1
2	529-56-3974	B	Mandalay	Member	Male	Electronic_accessories	2
3	299-46-1805	B	Mandalay	Member	Female	Sports_and_travel	9
4	319-50-3348	B	Mandalay	Normal	Female	Home_and_lifestyle	4

In [22]:

```
mart.columns
```

Out[22]:

```
Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'G
ender',
       'Product line', 'Unit price', 'Quantity', 'Tax 5%',
'Total', 'Date',
       'Time', 'Payment', 'cogs', 'gross margin percentag
e', 'gross income',
       'Rating'],
      dtype='object')
```

In [23]:

```
mart.index
```

Out[23]:

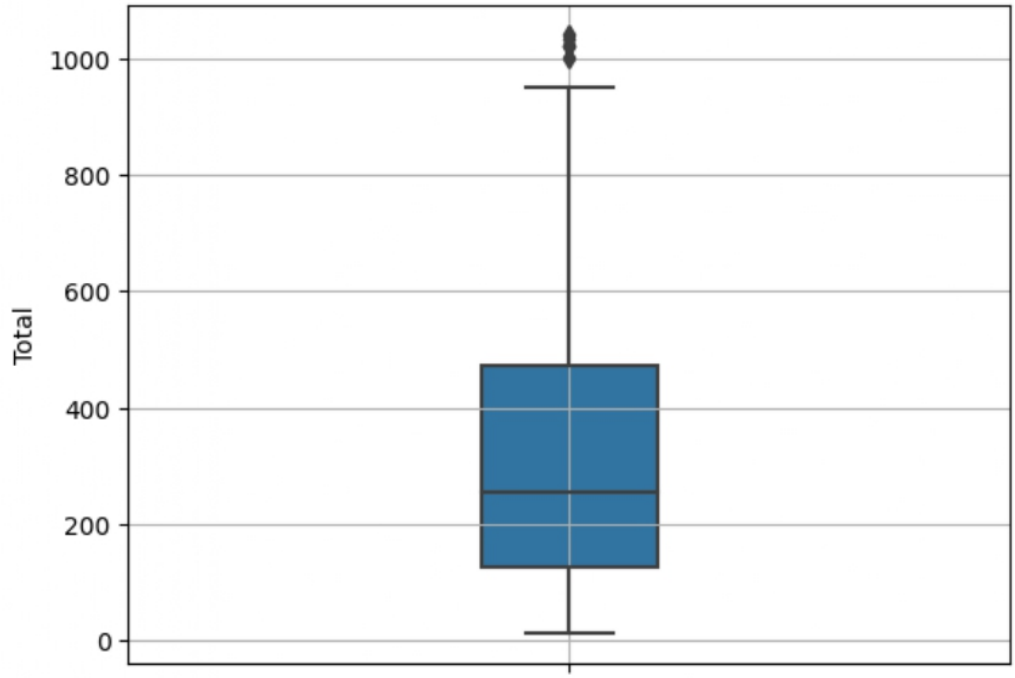
```
RangeIndex(start=0, stop=1000, step=1)
```

In [ ]:

```
```

In [24]:

```
sns.boxplot(y= "Total",data = mart,width = 0.2)
plt.grid()
```



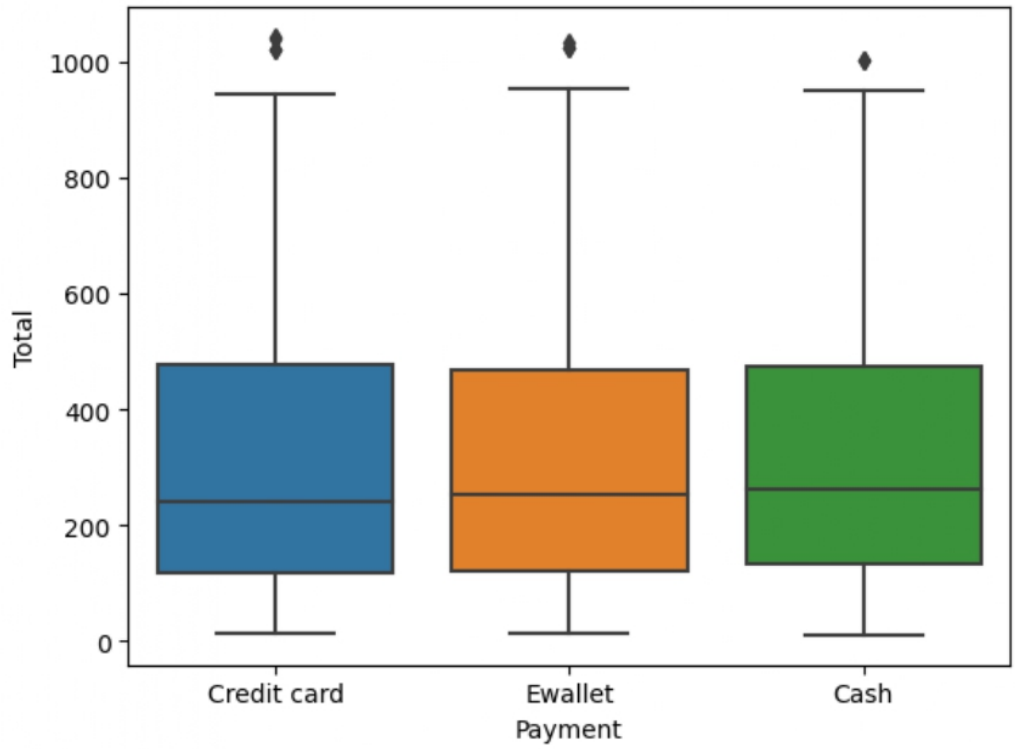
2.creat a basic BOX plot one NUMERIC veriable by CATEGORICAL vadiable

In [25]:

```
sns.boxplot(x = "Payment",y = "Total",data = mart)
```

Out[25]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```



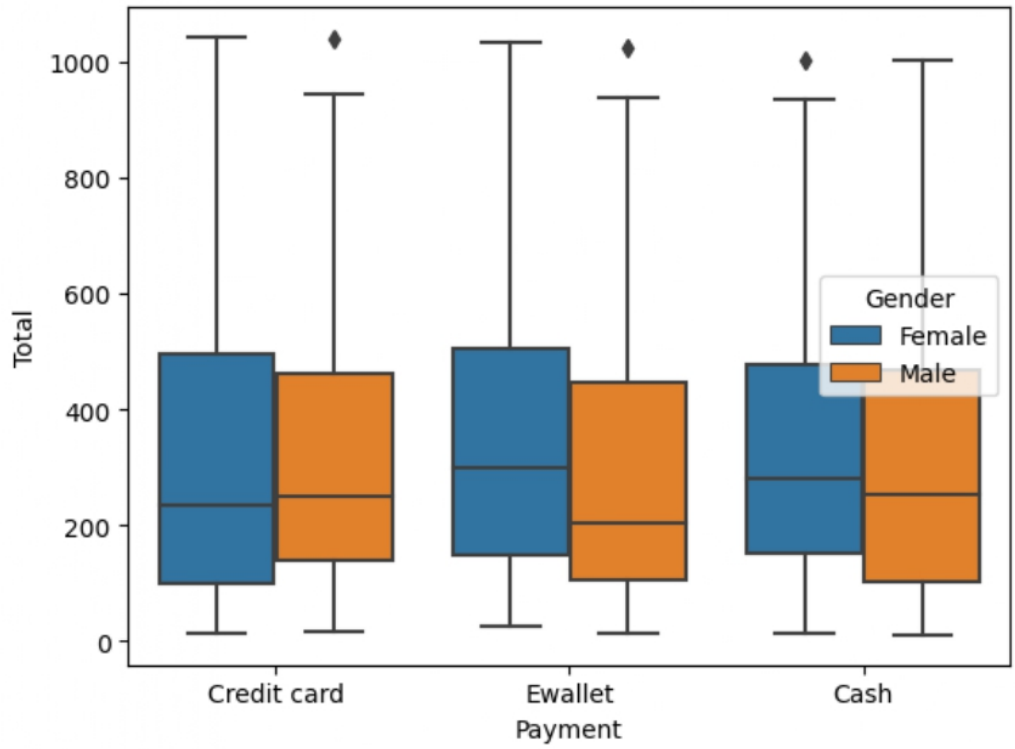
3.create a boxplot on one numeric variacle by two catogorical variable using HUE attribute

In [26]:

```
sns.boxplot(x = "Payment",y = "Total",hue = "Gender",data = mart)
```

Out[26]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```



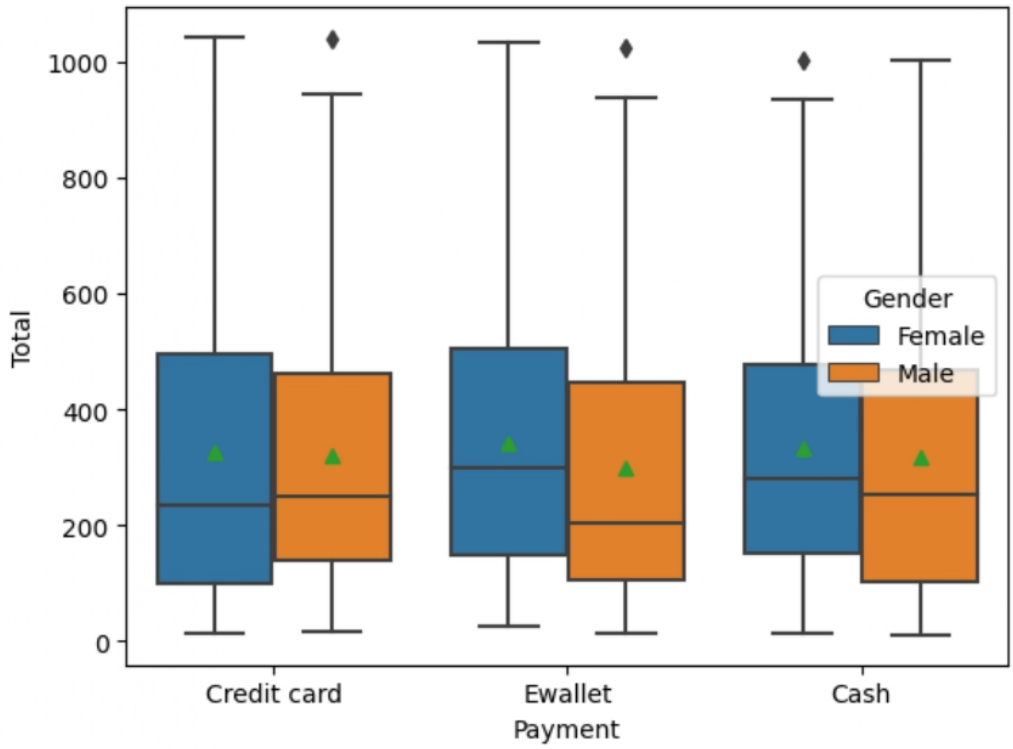
4.Add mean marker in the box plot using showmeans attribute and change its style using meanprops

In [27]:

```
sns.boxplot(x = "Payment",y = "Total",hue = "Gender",data = mart,showmear
```

Out[27]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```

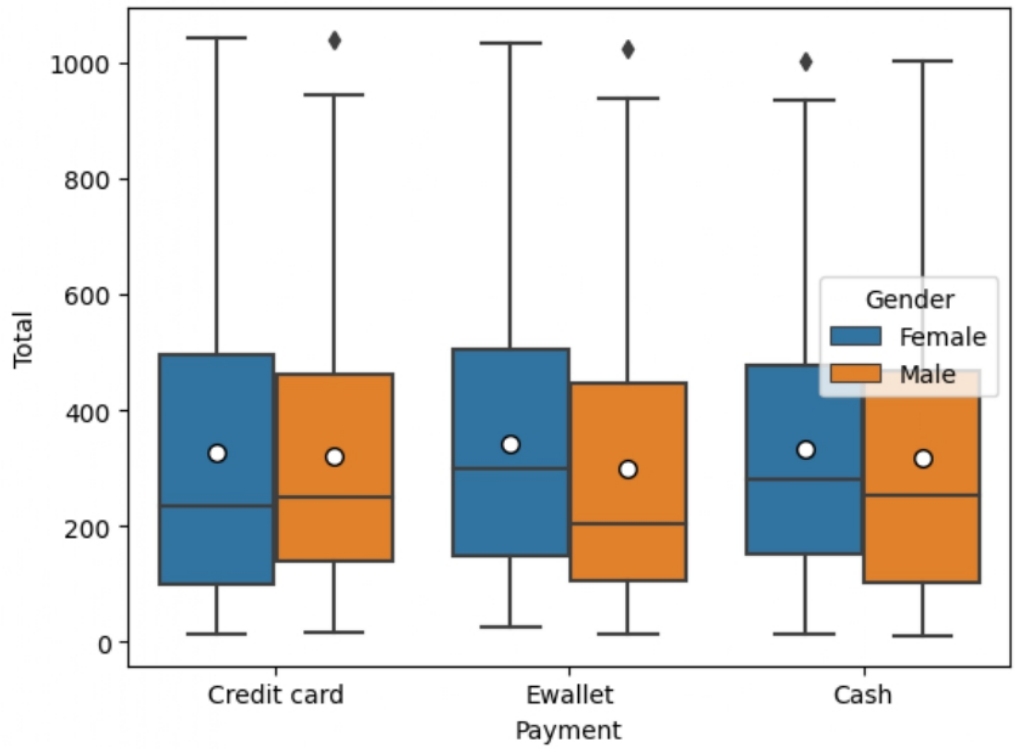


In [28]:

```
sns.boxplot(x = "Payment",y = "Total",hue = "Gender",data = mart,showmear
```

Out[28]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```



D) VIOLIN PLOT

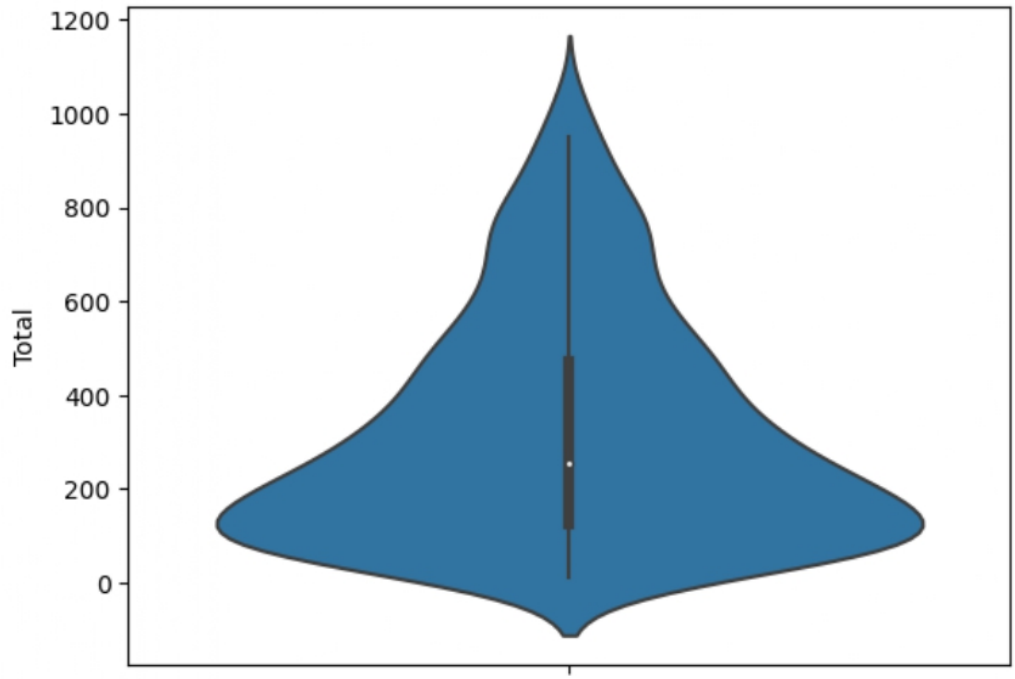
1.create a basic violinplot

In [29]:

```
sns.violinplot(y = "Total",data = mart)
```

Out[29]:

```
<AxesSubplot: ylabel='Total'>
```



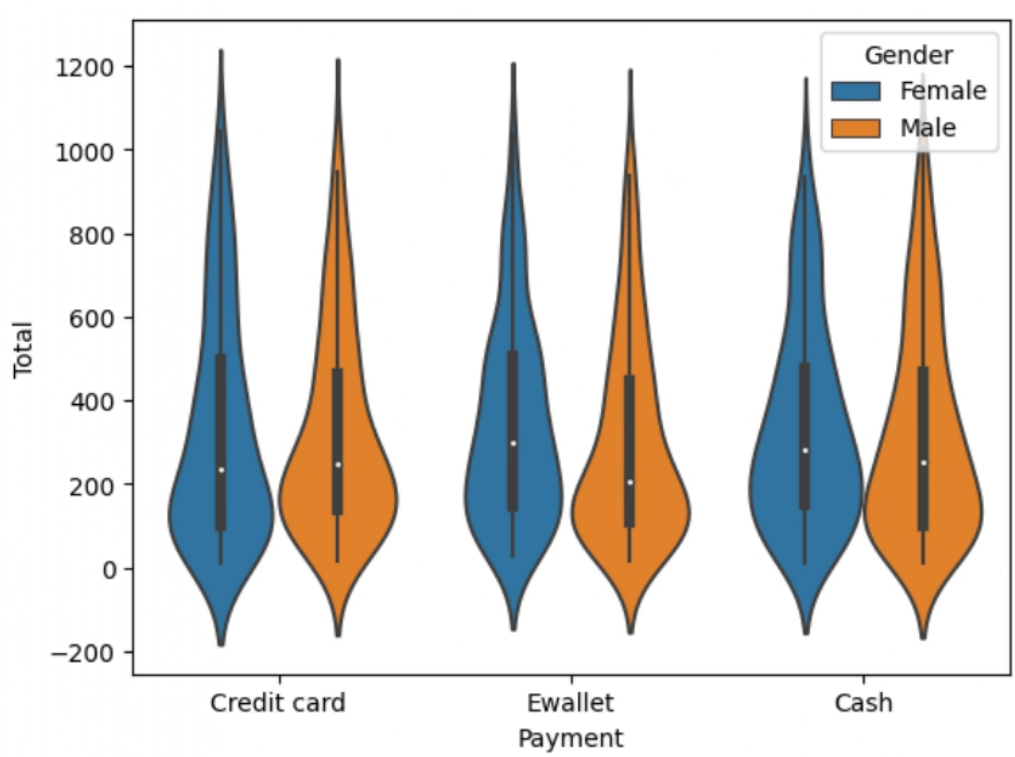
2.create a violelplot on two categorical and numeric variable and use split

In [30]:

```
sns.violinplot(x = "Payment",y ="Total",hue = "Gender",data = mart)
```

Out[30]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```

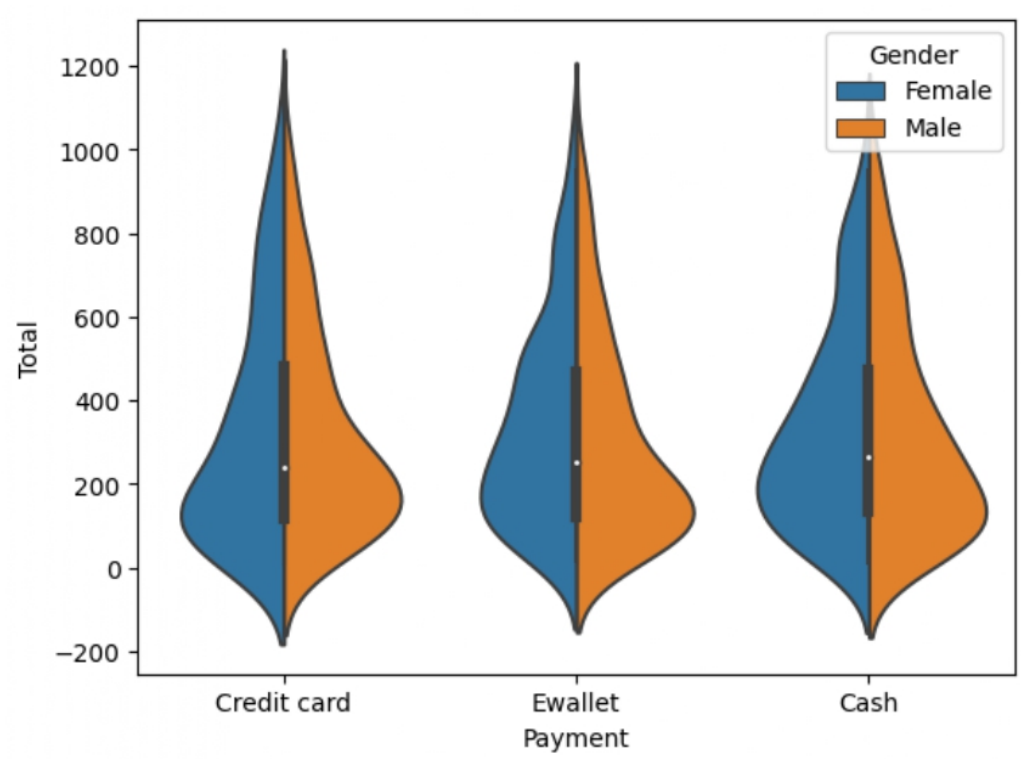


In [31]:

```
sns.violinplot(x = "Payment",y ="Total",hue = "Gender",data = mart,split
```

Out[31]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```



In [ ]:

```

```

E) Strip plot

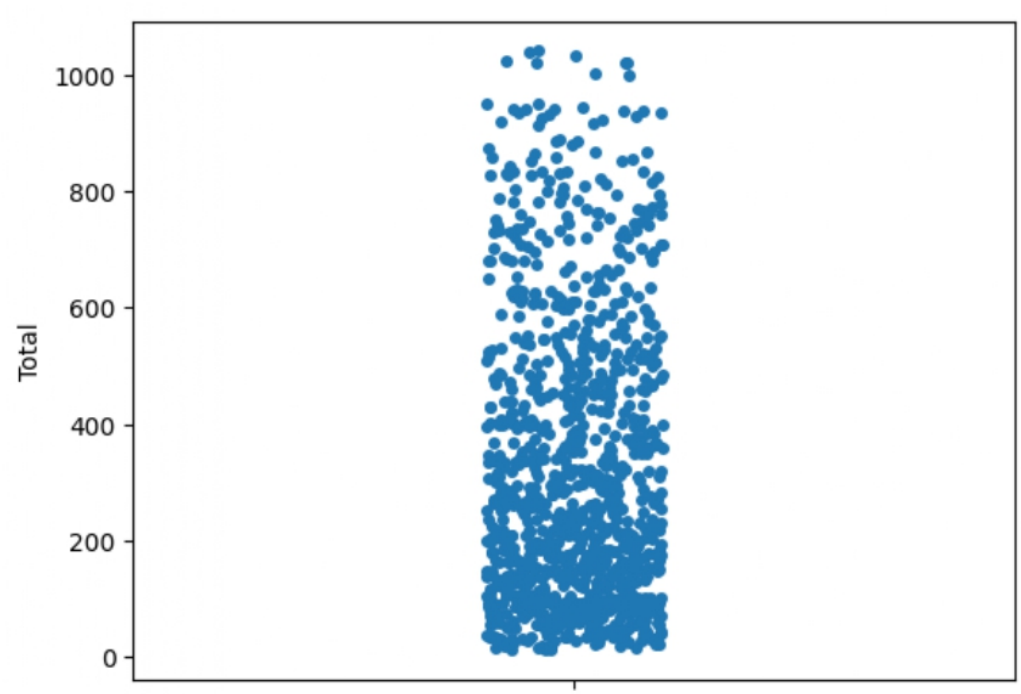
1.create a basic strip plot

In [32]:

```
sns.stripplot(y = "Total",data = mart)
```

Out[32]:

```
<AxesSubplot: ylabel='Total'>
```



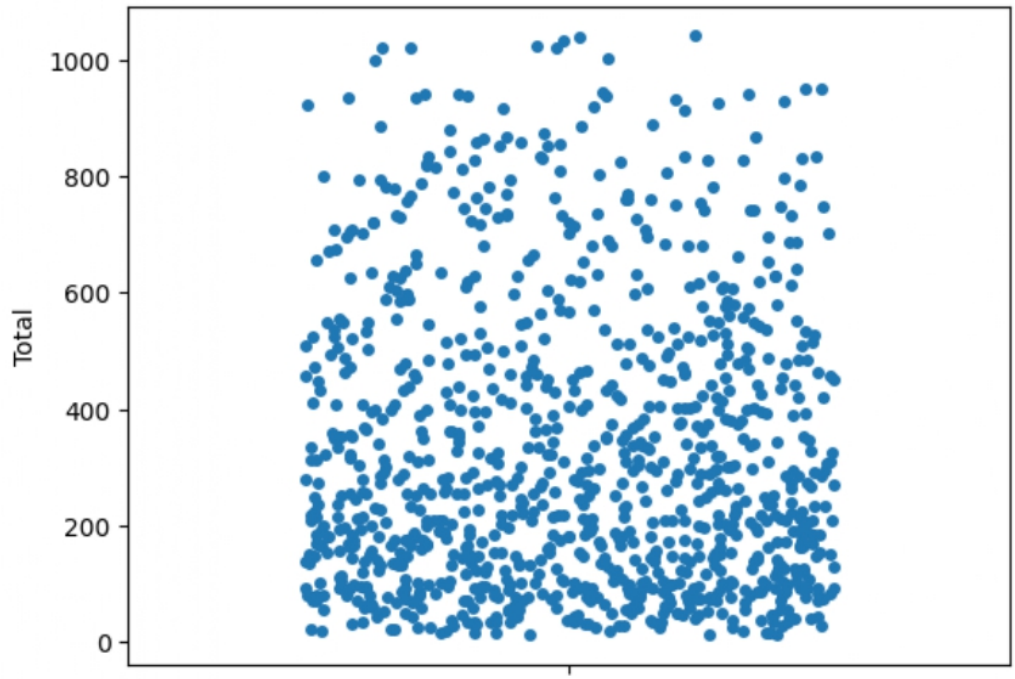
2.Expand markers in strip plot using jitter

In [33]:

```
sns.stripplot(y = "Total",data = mart,jitter =0.3)
```

Out[33]:

```
<AxesSubplot: ylabel='Total'>
```



In [34]:

```
# name = sns.load_dataset("titanic")
```

In [35]:

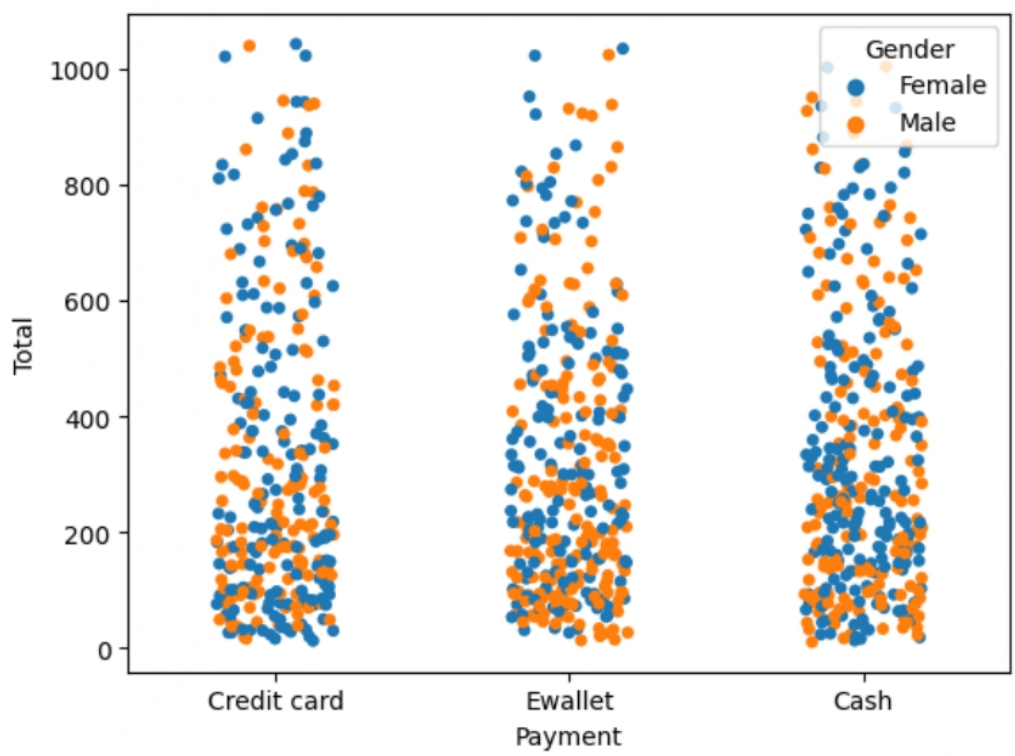
```
# name.head()
```

In [36]:

```
sns.stripplot(data = mart,y = 'Total',x = 'Payment',hue= 'Gender',jitter
```

Out[36]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```



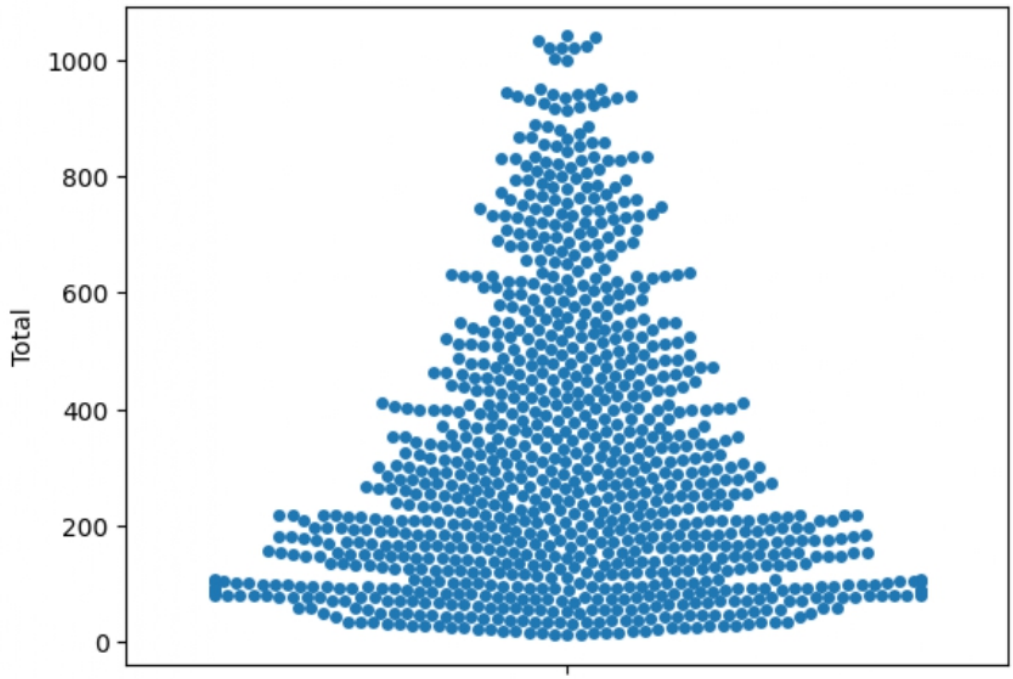
F) Swarm plot

In [37]:

```
sns.swarmplot(data = mart,y = "Total")
```

Out[37]:

```
<AxesSubplot: ylabel='Total'>
```



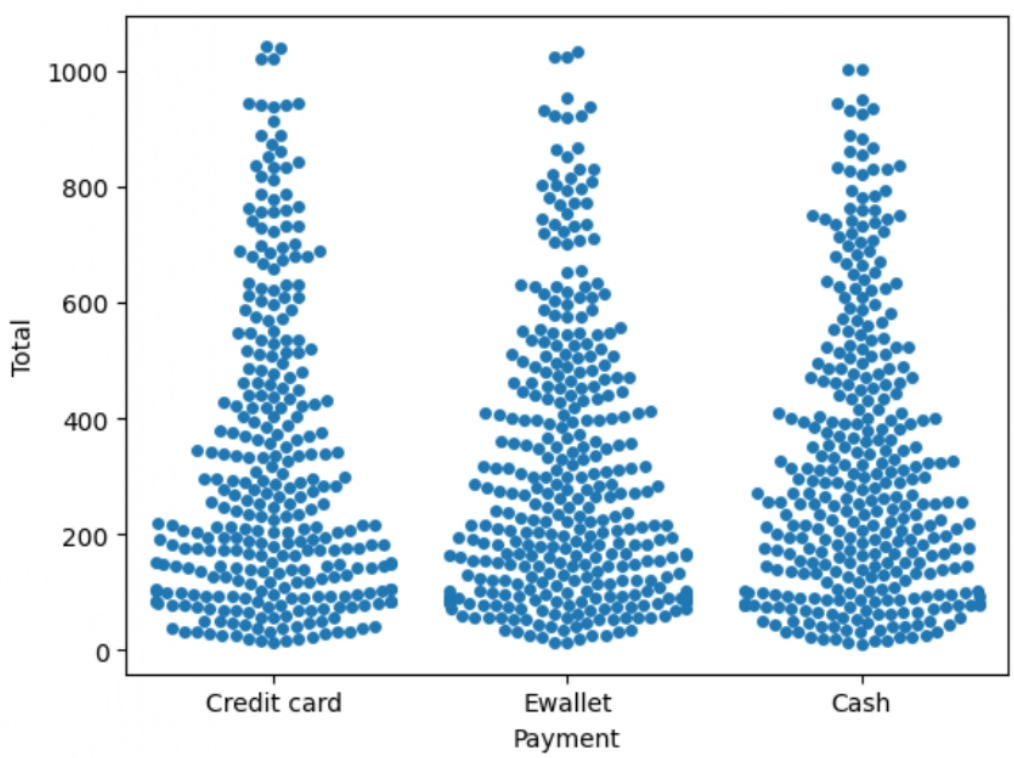
2.create swarm plot group by categories

In [38]:

```
sns.swarmplot(data = mart,y= "Total",x = "Payment")
```

Out[38]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```

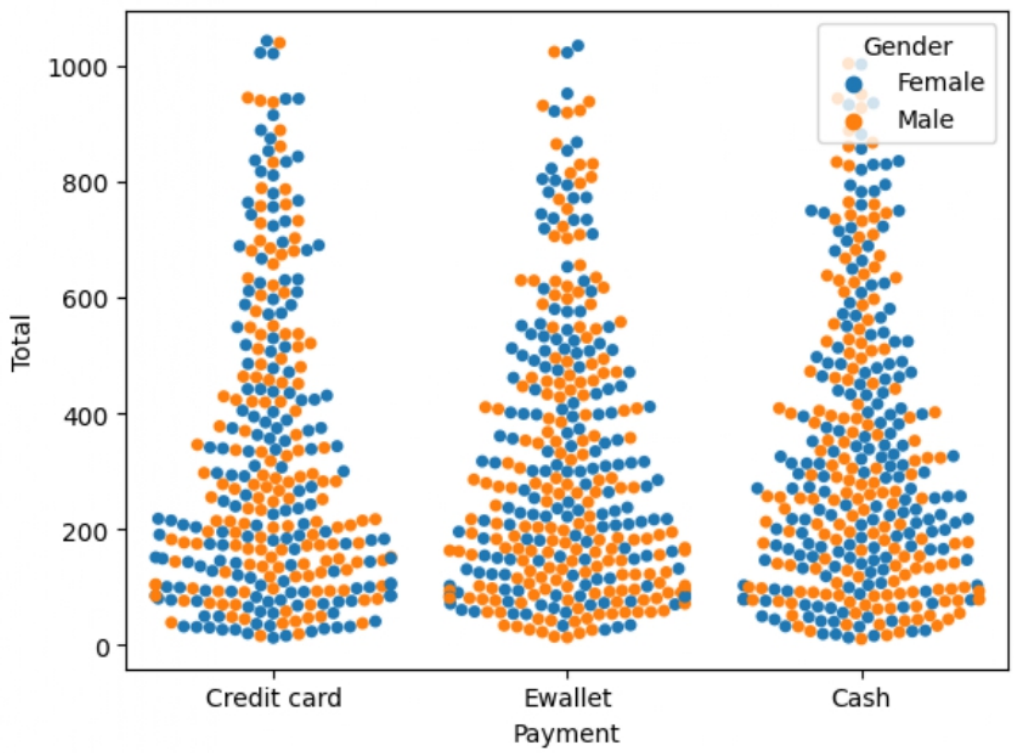


In [39]:

```
sns.swarmplot(data = mart,y= "Total",x = "Payment",hue  ="Gender")
```

Out[39]:

```
<AxesSubplot: xlabel='Payment', ylabel='Total'>
```



3.showing hues separatly on categorical axis

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

In [ ]:

```
```

DISTRIBUTION

G) Histogram plot

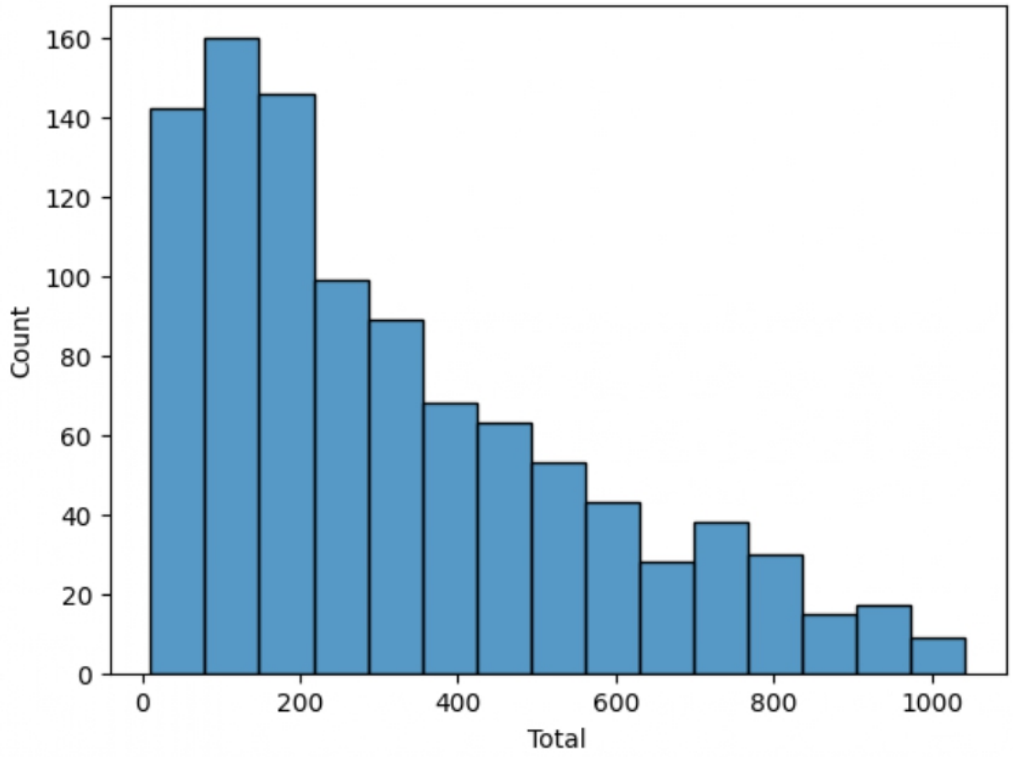
1.create a basic histogram plot

In [40]:

```
sns.histplot(data = mart,x = "Total")
```

Out[40]:

```
<AxesSubplot: xlabel='Total', ylabel='Count'>
```



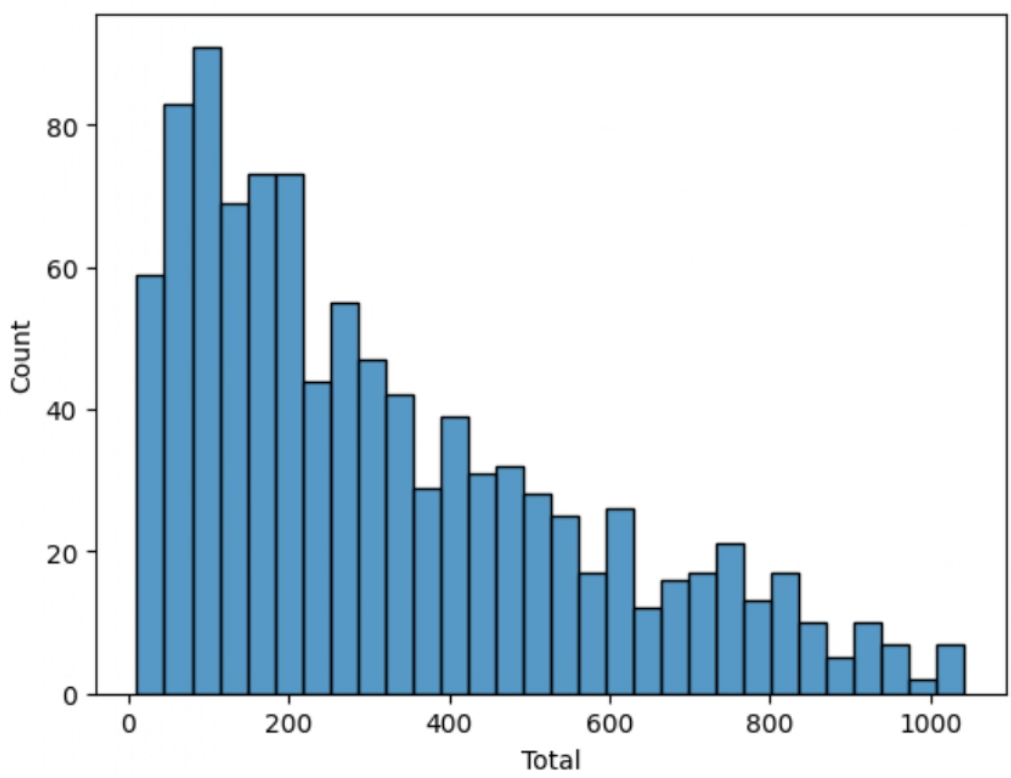
2.change number of bins and interval

In [41]:

```
sns.histplot(data = mart,x = "Total",bins = 30)
```

Out[41]:

```
<AxesSubplot: xlabel='Total', ylabel='Count'>
```



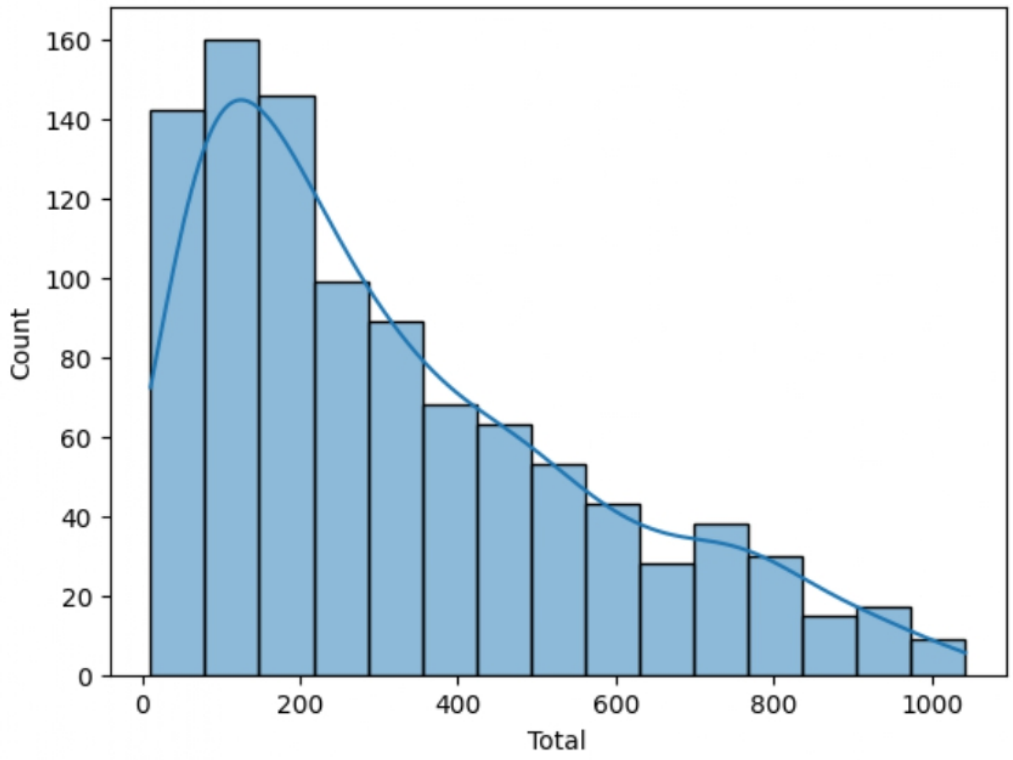
3.combine with KDE

In [42]:

```
sns.histplot(data = mart,x = "Total",kde = True)
```

Out[42]:

```
<AxesSubplot: xlabel='Total', ylabel='Count'>
```



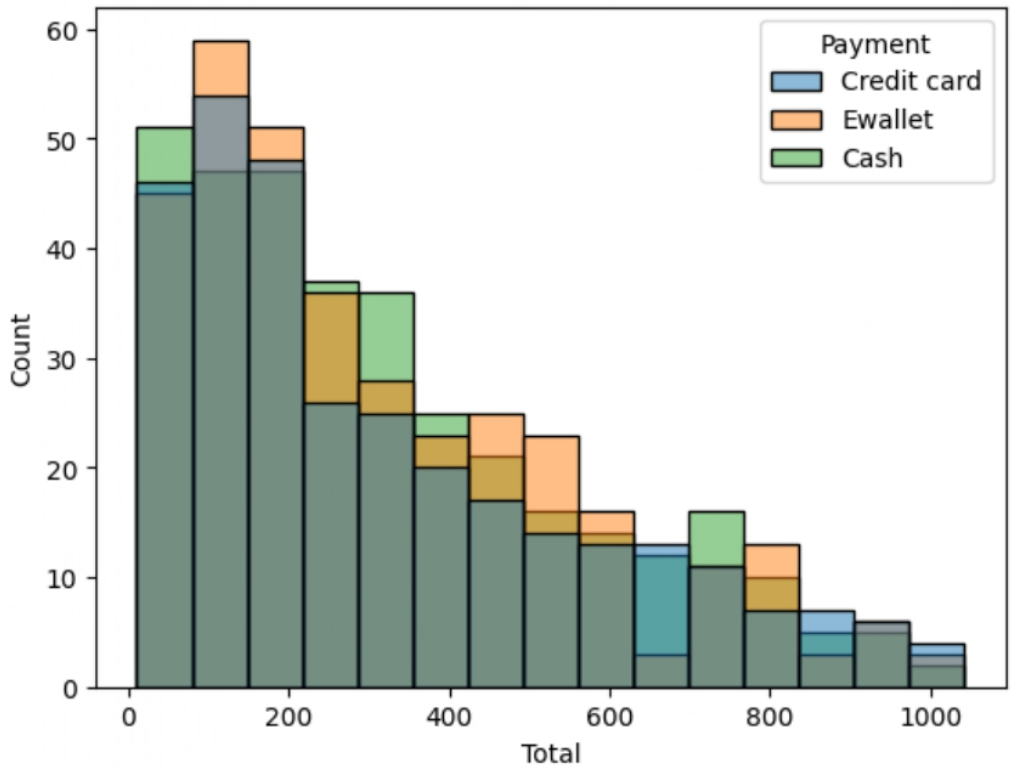
4.use a catagorical variable in HUE and STICK it MULTIPLE argument

In [43]:

```
sns.histplot(data = mart,x = "Total",hue = "Payment")
```

Out[43]:

```
<AxesSubplot: xlabel='Total', ylabel='Count'>
```

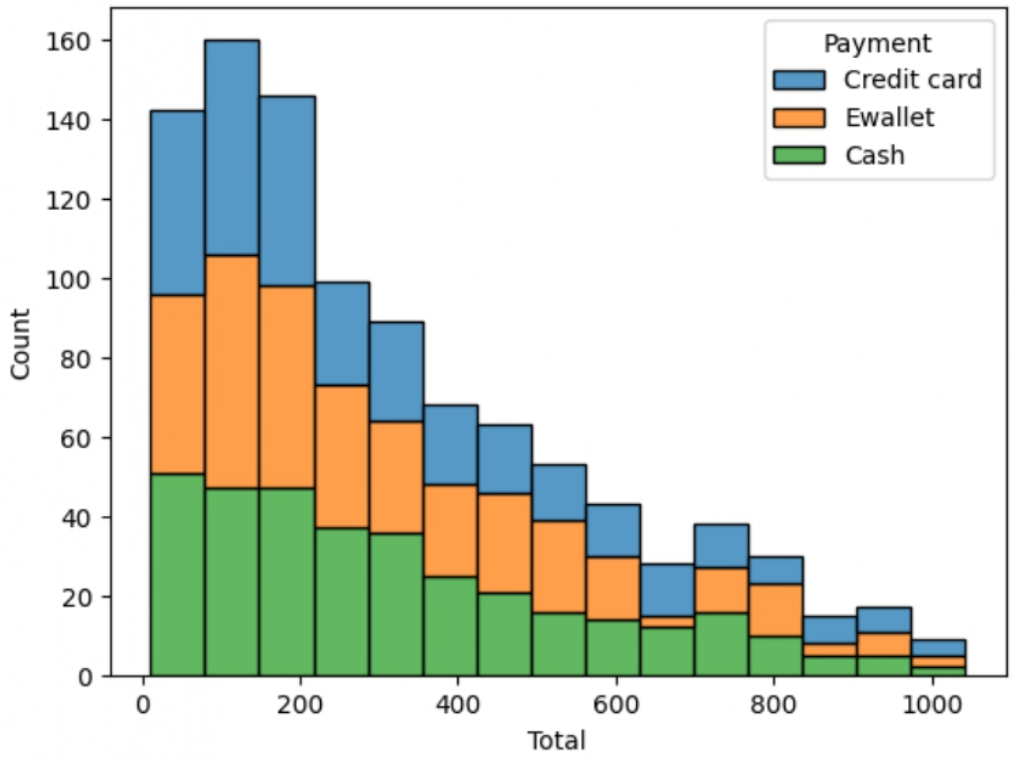


In [44]:

```
sns.histplot(data = mart,x = "Total",hue = "Payment",multiple = "stack")
```

Out[44]:

```
<AxesSubplot: xlabel='Total', ylabel='Count'>
```

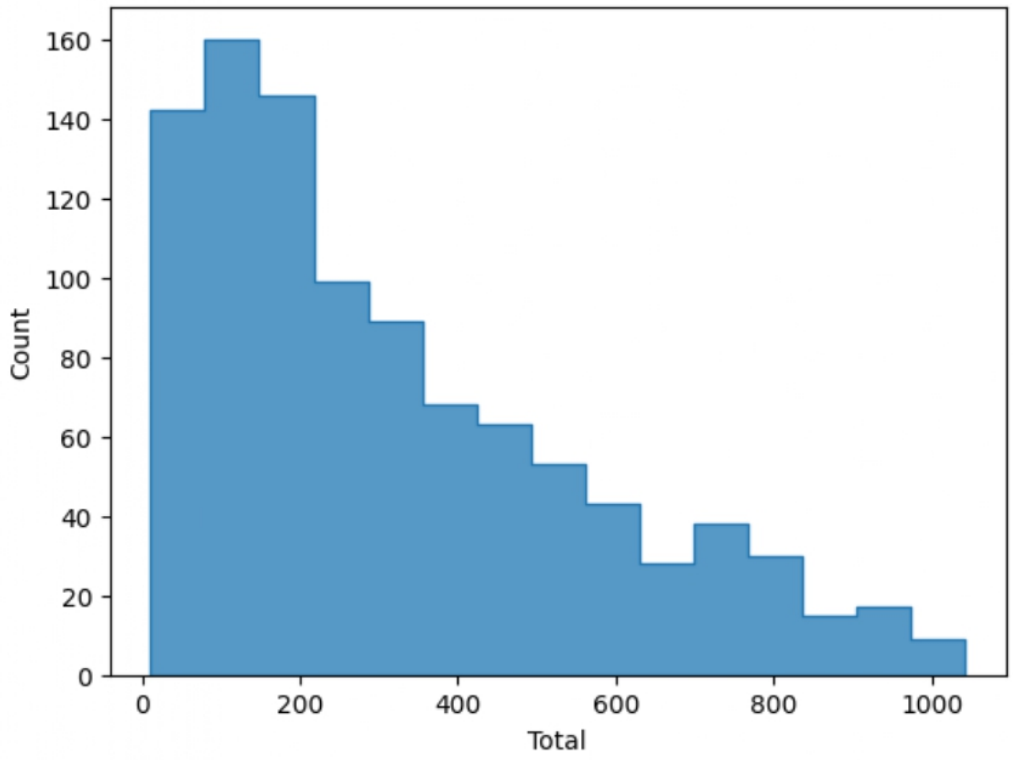


In [45]:

```
sns.histplot(data = mart,x = "Total",element = "step")
```

Out[45]:

```
<AxesSubplot: xlabel='Total', ylabel='Count'>
```

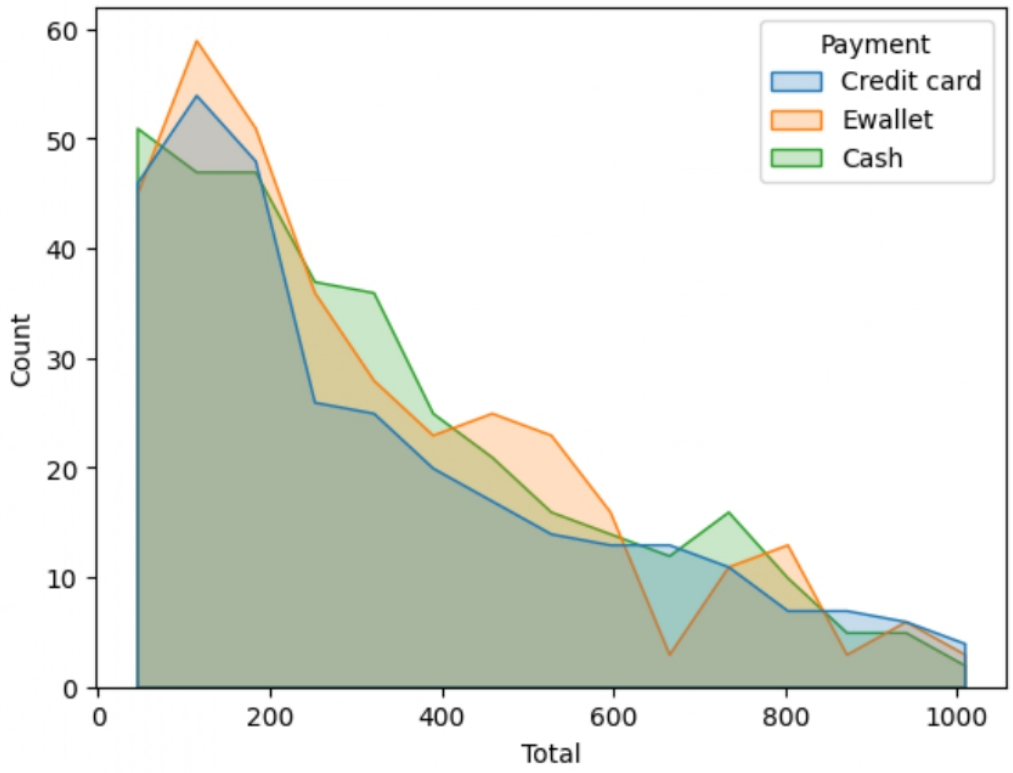


In [46]:

```
sns.histplot(data = mart,x = "Total",hue = "Payment",element = "poly")
```

Out[46]:

```
<AxesSubplot: xlabel='Total', ylabel='Count'>
```



H) KDEplot

In [47]:

```
# KDE means Kernel Density Estimation plot
```

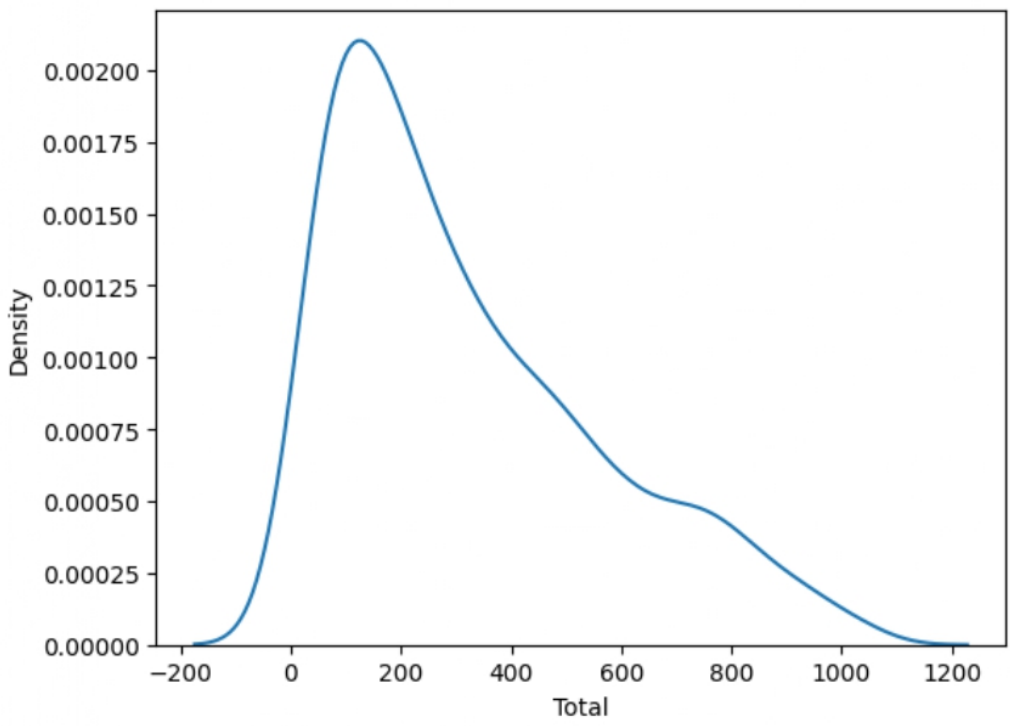
1.create a KDE plot for total column

In [48]:

```
sns.kdeplot(data = mart,x = "Total")
```

Out[48]:

```
<AxesSubplot: xlabel='Total', ylabel='Density'>
```



2.create a KDE plot for all the numeric variables in dataframe

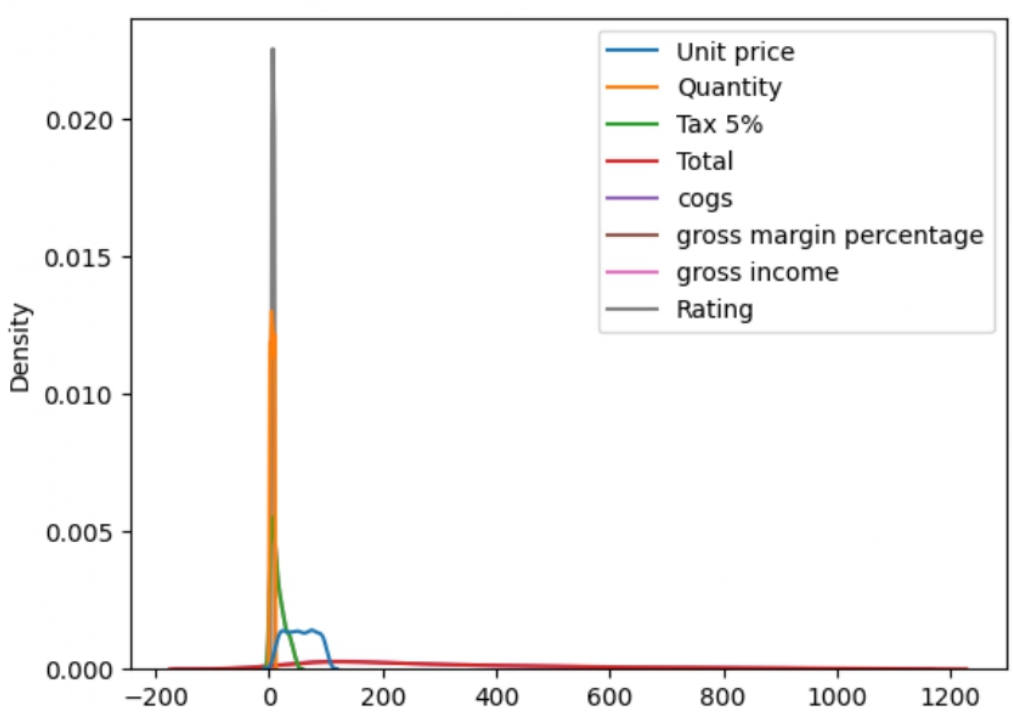
In [49]:

```
sns.kdeplot(data = mart)
```

```
C:\Users\joypa\AppData\Local\Temp\ipykernel_7164\124168617
3.py:1: UserWarning: Dataset has 0 variance; skipping dens
ity estimate. Pass `warn_singular=False` to disable this w
arning.
  sns.kdeplot(data = mart)
```

Out[49]:

```
<AxesSubplot: ylabel='Density'>
```



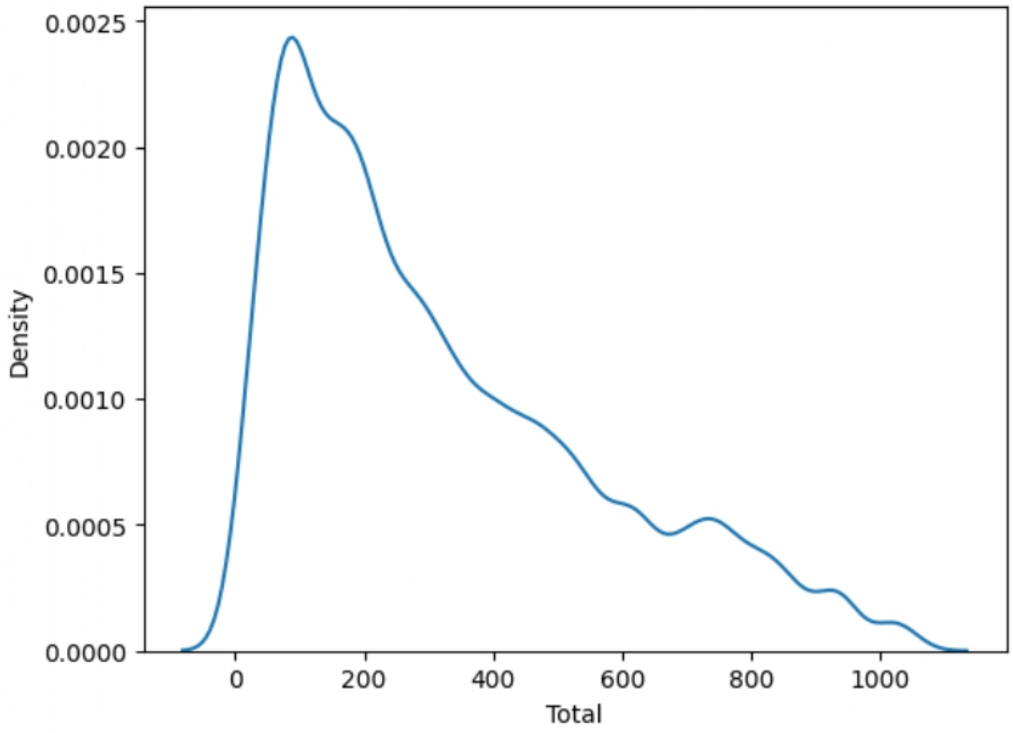
3.adjust the smoothing using bw_adjust

In [50]:

```
sns.kdeplot(data = mart,x = "Total",bw_adjust = 0.5)
```

Out[50]:

```
<AxesSubplot: xlabel='Total', ylabel='Density'>
```



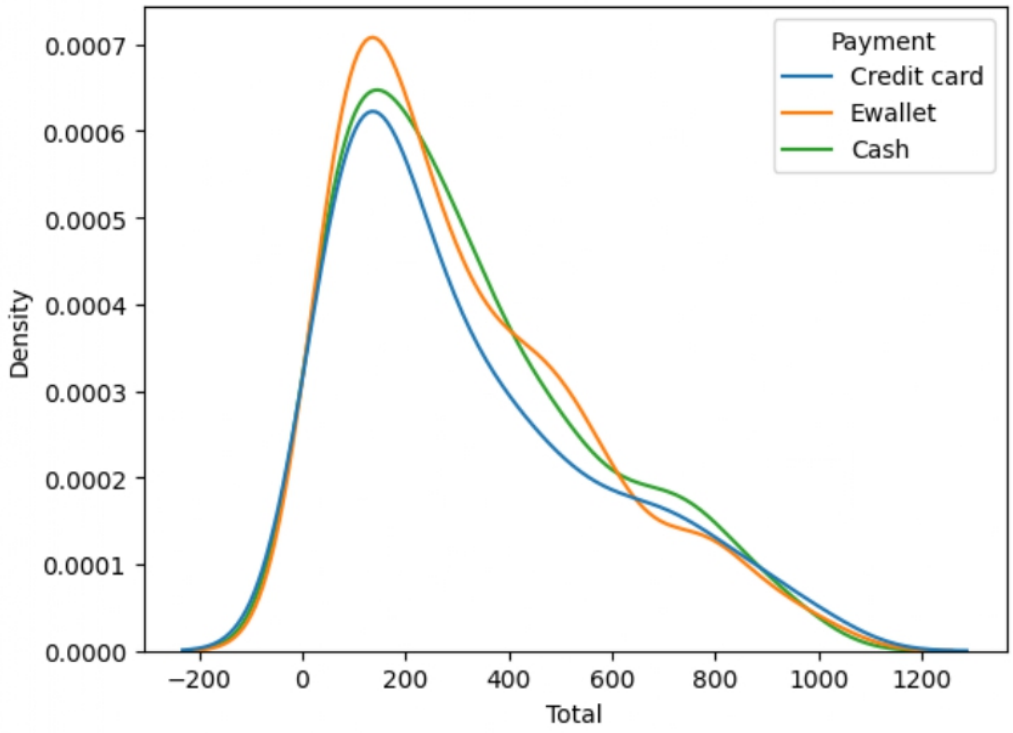
4.group the KDE on categorical variable

In [51]:

```
sns.kdeplot(data = mart,x = "Total",hue = "Payment")
```

Out[51]:

```
<AxesSubplot: xlabel='Total', ylabel='Density'>
```



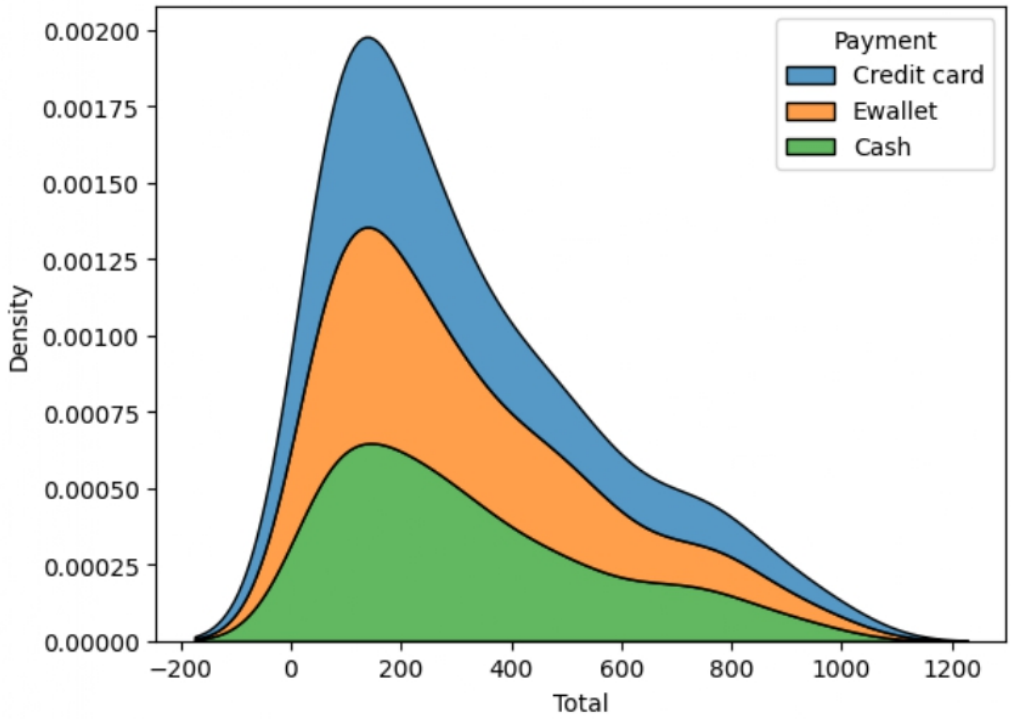
5.stack KDE on a categorical using MULTIPLE argument

In [52]:

```
sns.kdeplot(data = mart,x = "Total",hue = "Payment",multiple = "stack")
```

Out[52]:

```
<AxesSubplot: xlabel='Total', ylabel='Density'>
```



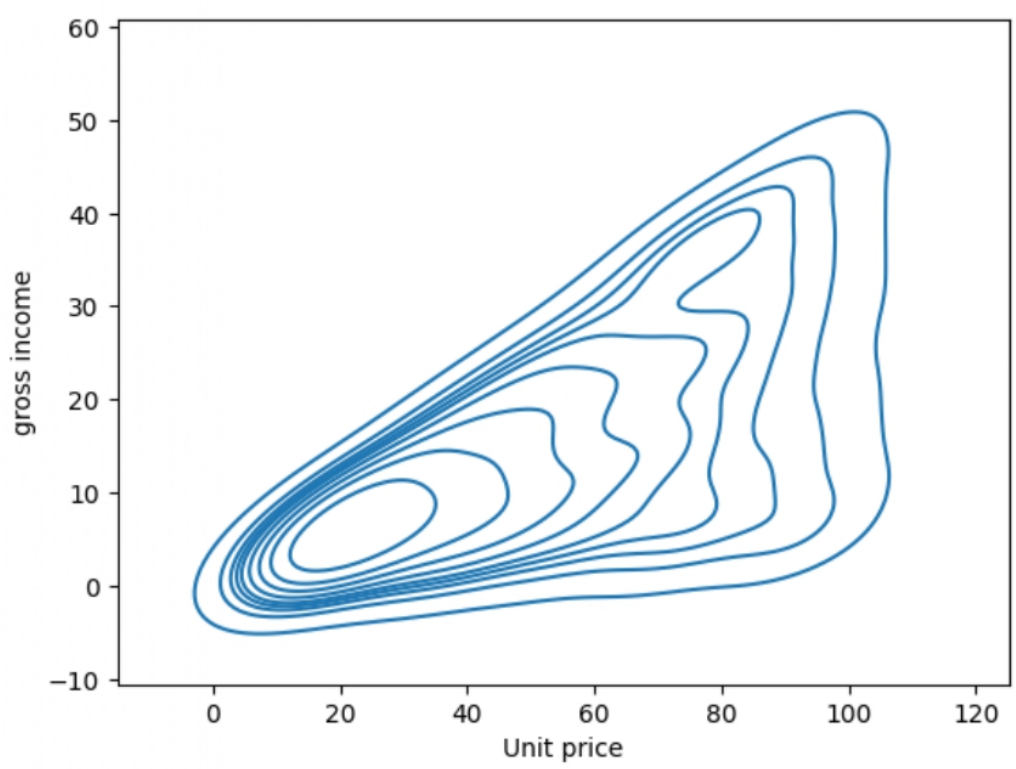
6.create a bivariate KDE

In [53]:

```
sns.kdeplot(data = mart,x = "Unit price",y = "gross income")
```

Out[53]:

```
<AxesSubplot: xlabel='Unit price', ylabel='gross income'>
```



In [ ]:

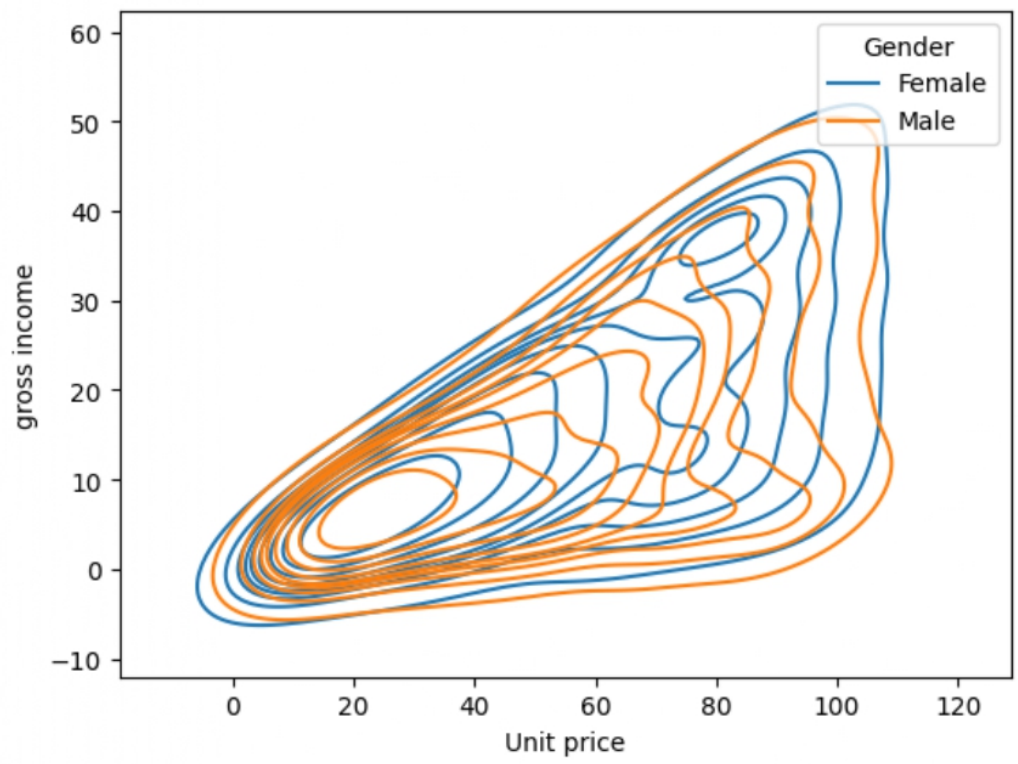
```

```

In [54]:

```
sns.kdeplot(data = mart,x = "Unit price",y = "gross income",hue = "Gender
```

Out[54]:

```
<AxesSubplot: xlabel='Unit price', ylabel='gross income'>
```

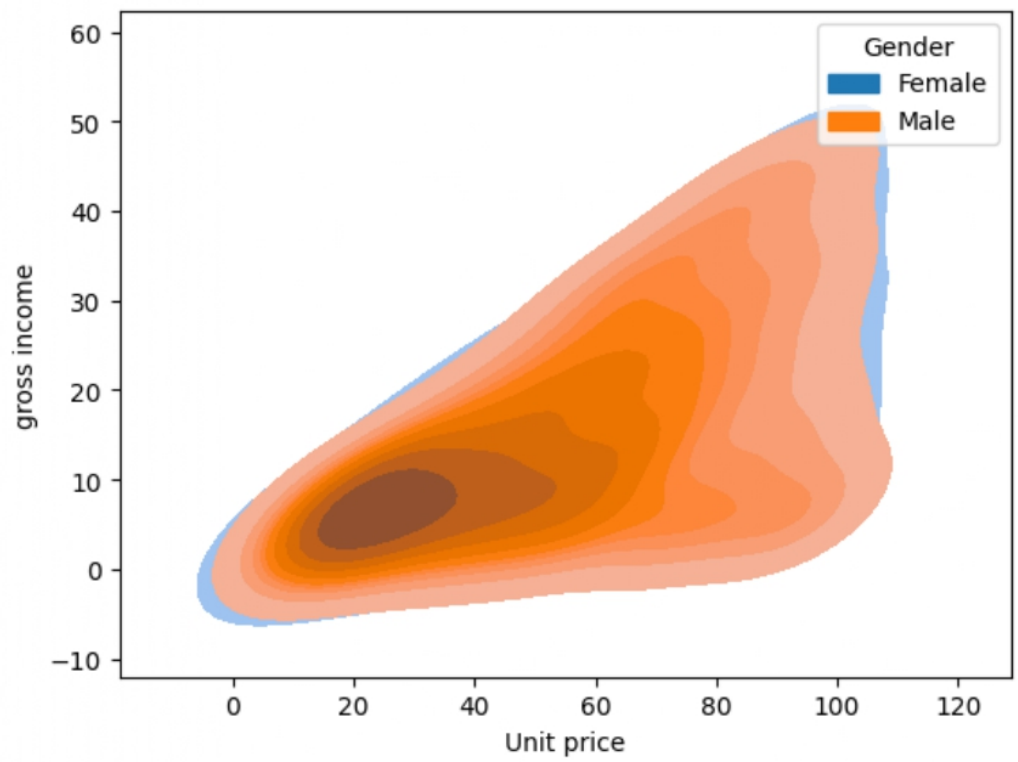


In [55]:

```
sns.kdeplot(data = mart,x = "Unit price",y = "gross income",hue = "Gender
```

Out[55]:

```
<AxesSubplot: xlabel='Unit price', ylabel='gross income'>
```



I)RUG plot

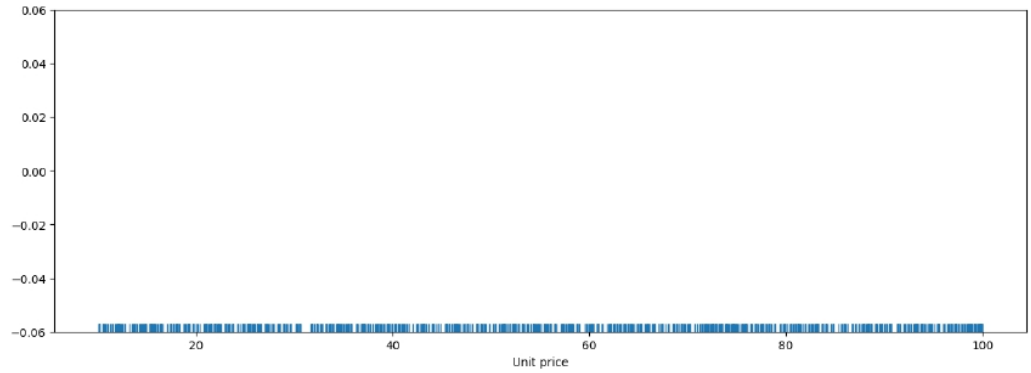
1.create a basic rug plot for one variable

In [56]:

```
plt.figure(figsize=  (15,5))
sns.rugplot(data = mart,x = "Unit price")
```

Out[56]:

```
<AxesSubplot: xlabel='Unit price'>
```



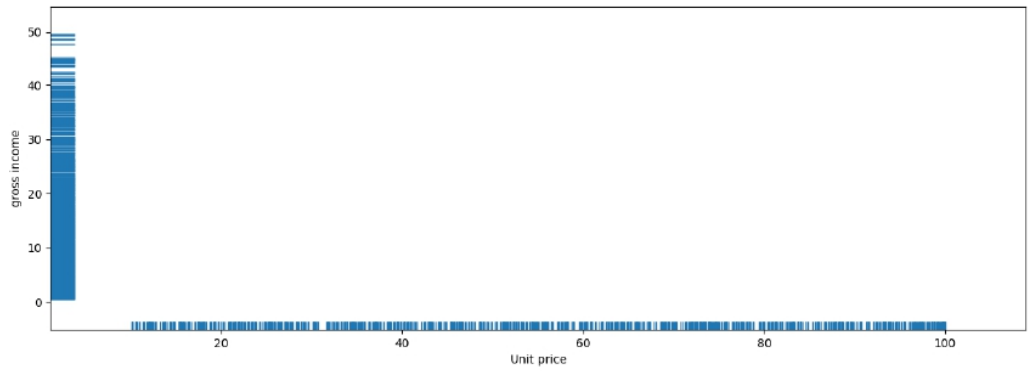
2. create a rug plot for two variable

In [57]:

```
plt.figure(figsize=  (15,5))
sns.rugplot(data = mart,x = "Unit price",y = "gross income")
```

Out[57]:

```
<AxesSubplot: xlabel='Unit price', ylabel='gross income'>
```



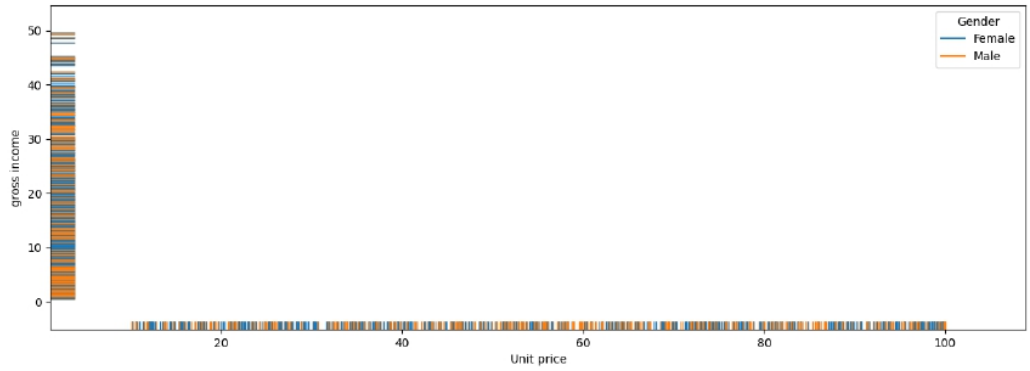
3. group it by a catagorical variable using HUE

In [58]:

```
plt.figure(figsize=  (15,5))
sns.rugplot(data = mart,x = "Unit price",y = "gross income",hue = "Gender
```

Out[58]:

```
<AxesSubplot: xlabel='Unit price', ylabel='gross income'>
```



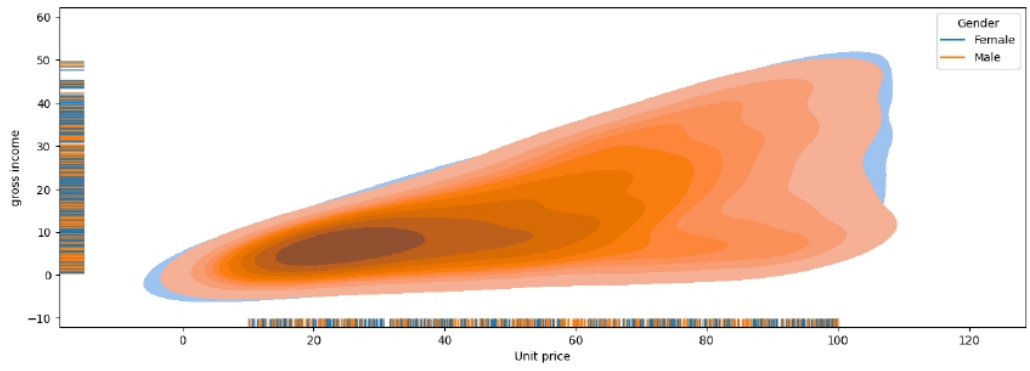
4. combine this with a KDE plot

In [59]:

```
plt.figure(figsize=  (15,5))
sns.kdeplot(data = mart,x = "Unit price",y = "gross income",hue = "Gender
sns.rugplot(data = mart,x = "Unit price",y = "gross income",hue = "Gender
```

Out[59]:

```
<AxesSubplot: xlabel='Unit price', ylabel='gross income'>
```



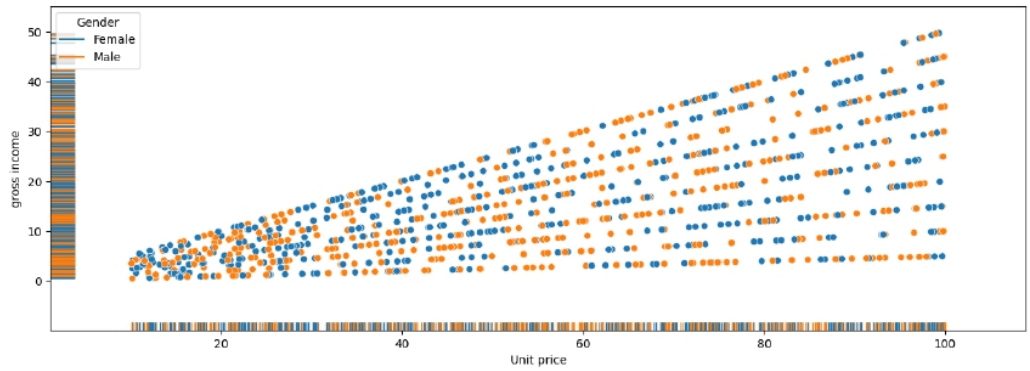
5.combine with scatter plot

In [60]:

```
plt.figure(figsize=  (15,5))
sns.scatterplot(data = mart,x = "Unit price",y = "gross income",hue = "Ge
sns.rugplot(data = mart,x = "Unit price",y = "gross income",hue = "Gender
```

Out[60]:

```
<AxesSubplot: xlabel='Unit price', ylabel='gross income'>
```



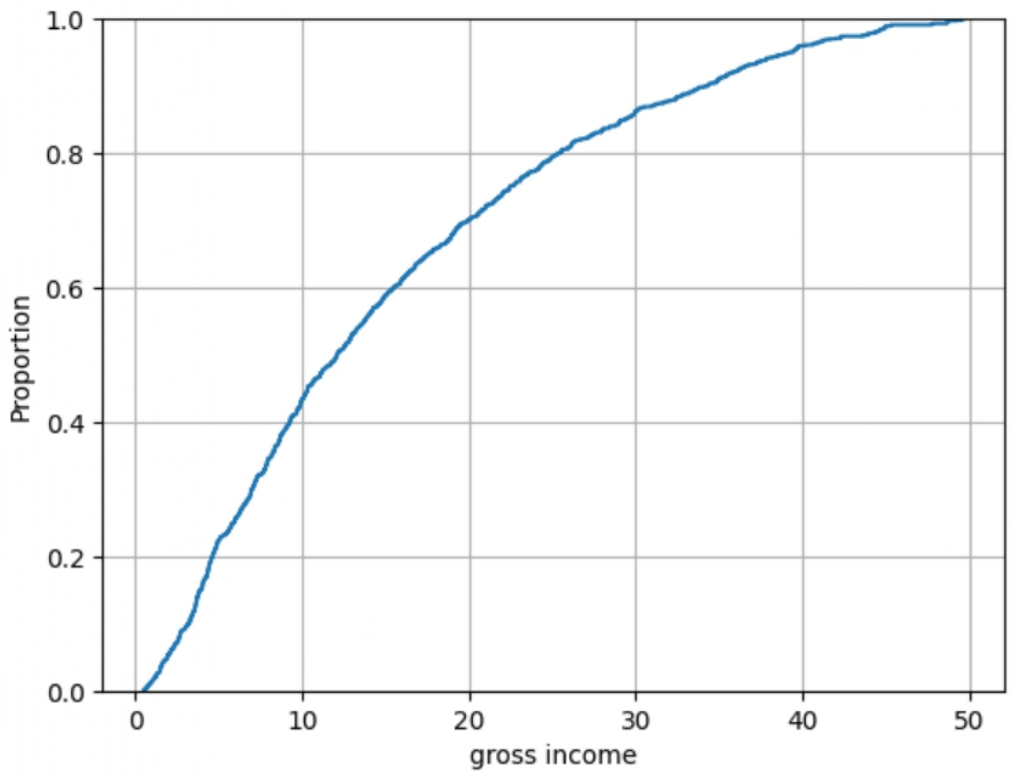
J) ECDF plot

In [61]:

```
# ECDF = Empirical Cumulative Distribution Function
```

In [62]:

```
sns.ecdfplot(data = mart,x = "gross income")
plt.grid()
```

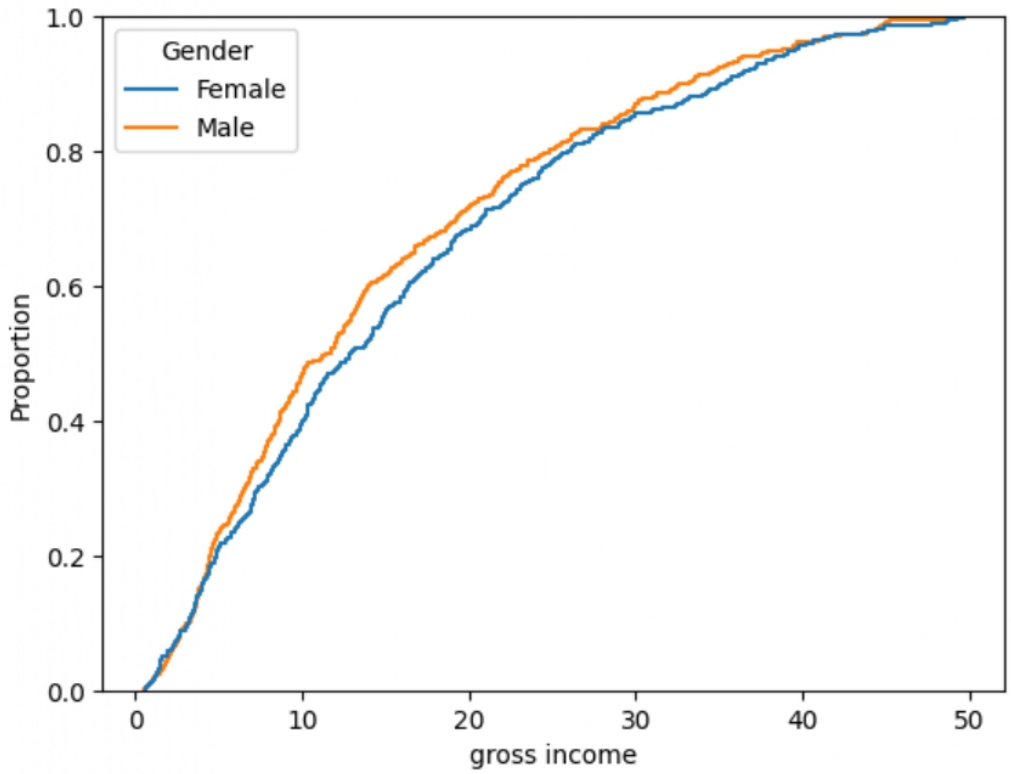


In [63]:

```
sns.ecdfplot(data = mart,x = "gross income",hue = "Gender",stat = "propor
```

Out[63]:

```
<AxesSubplot: xlabel='gross income', ylabel='Proportion'>
```

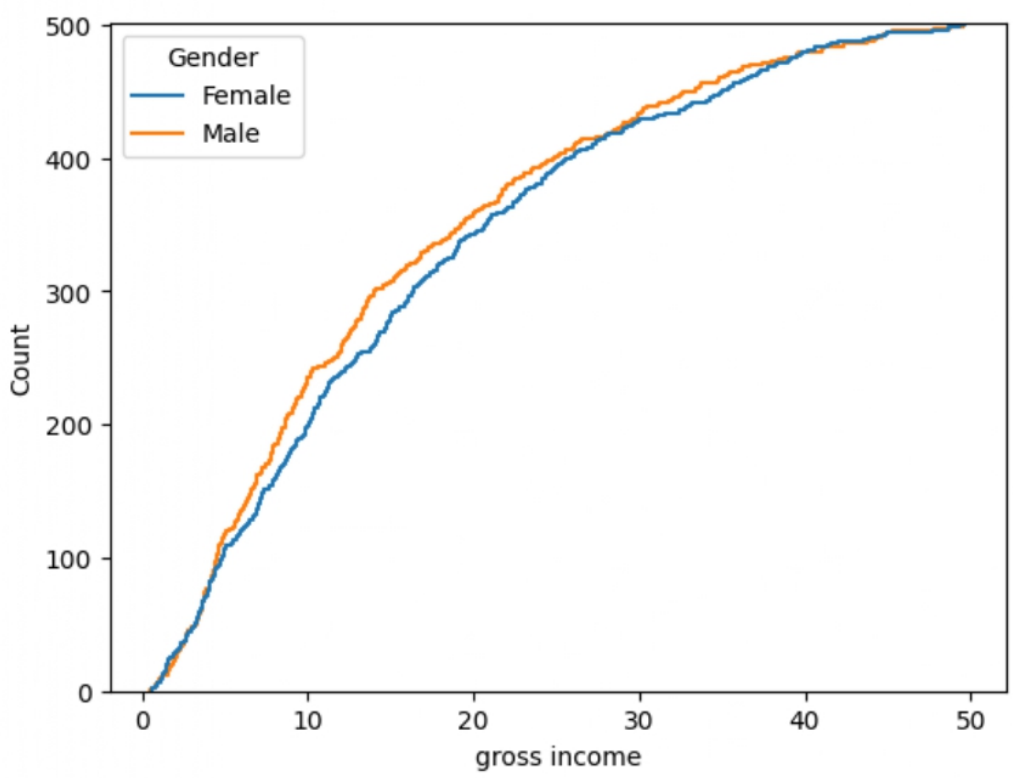


In [64]:

```
sns.ecdfplot(data = mart,x = "gross income",hue = "Gender",stat = "count"
```

Out[64]:

```
<AxesSubplot: xlabel='gross income', ylabel='Count'>
```



K) DIS plot

1. create a basic dis plot

In [65]:

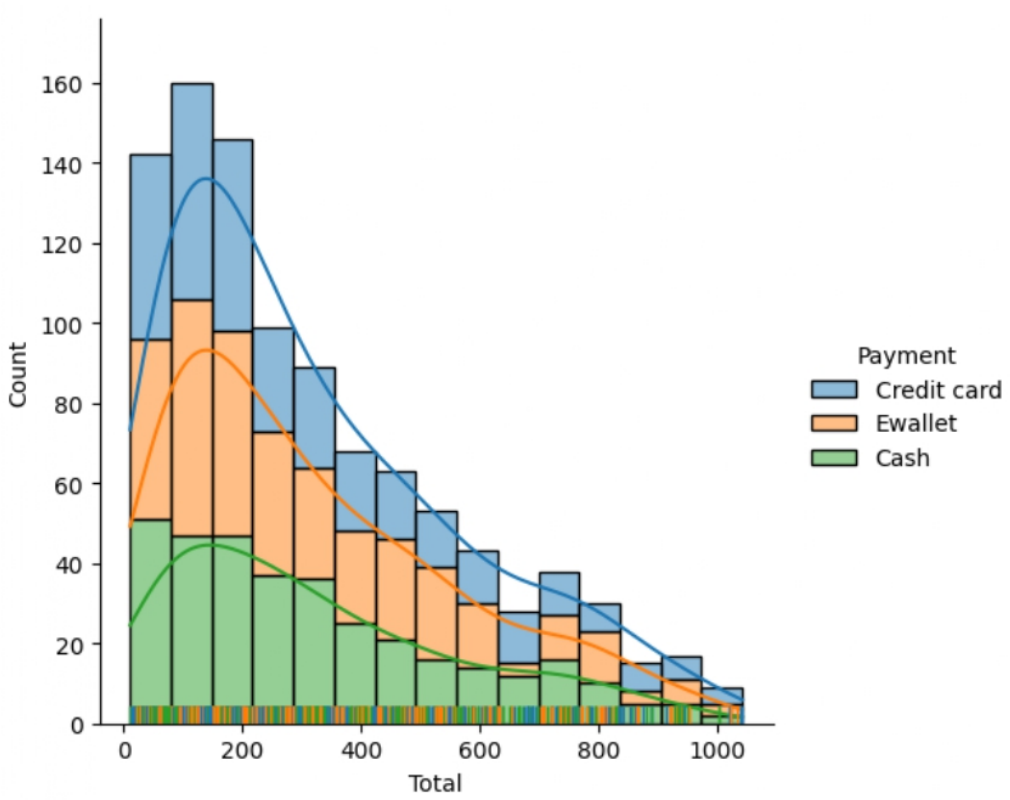
```
# DIS plot = Distribution plot
```

In [66]:

```
sns.displot(data = mart,x = "Total",kde = True,rug = True,hue = "Payment"
```

Out[66]:

```
<seaborn.axisgrid.FacetGrid at 0x2427e6f7280>
```



L) JOIN plot

In [67]:

```
irs= sns.load_dataset("iris")
```

In [68]:

```
irs.head()
```

Out[68]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

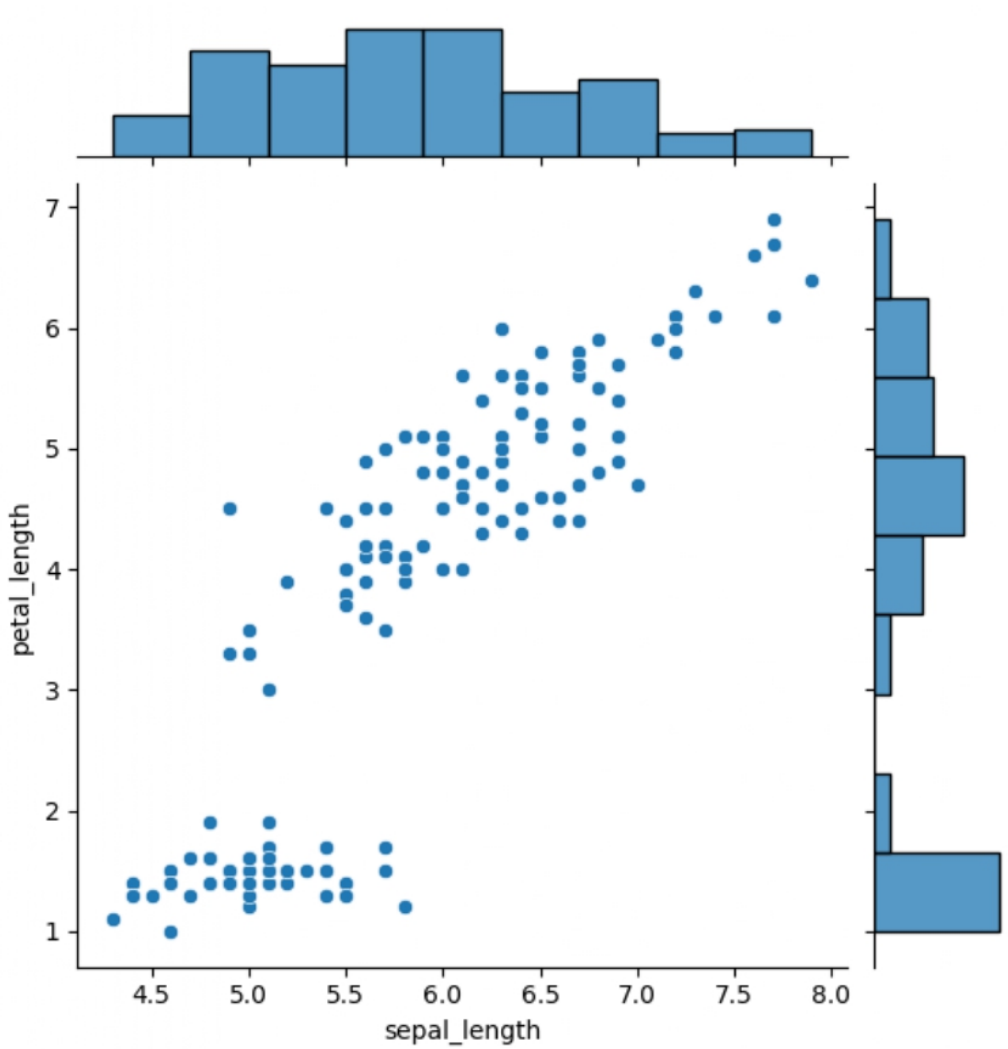
1.Draw a first basic JOIN plot

In [69]:

```
sns.jointplot(data = irs,x = "sepal_length",y = "petal_length")
```

Out[69]:

```
<seaborn.axisgrid.JointGrid at 0x2427e341090>
```

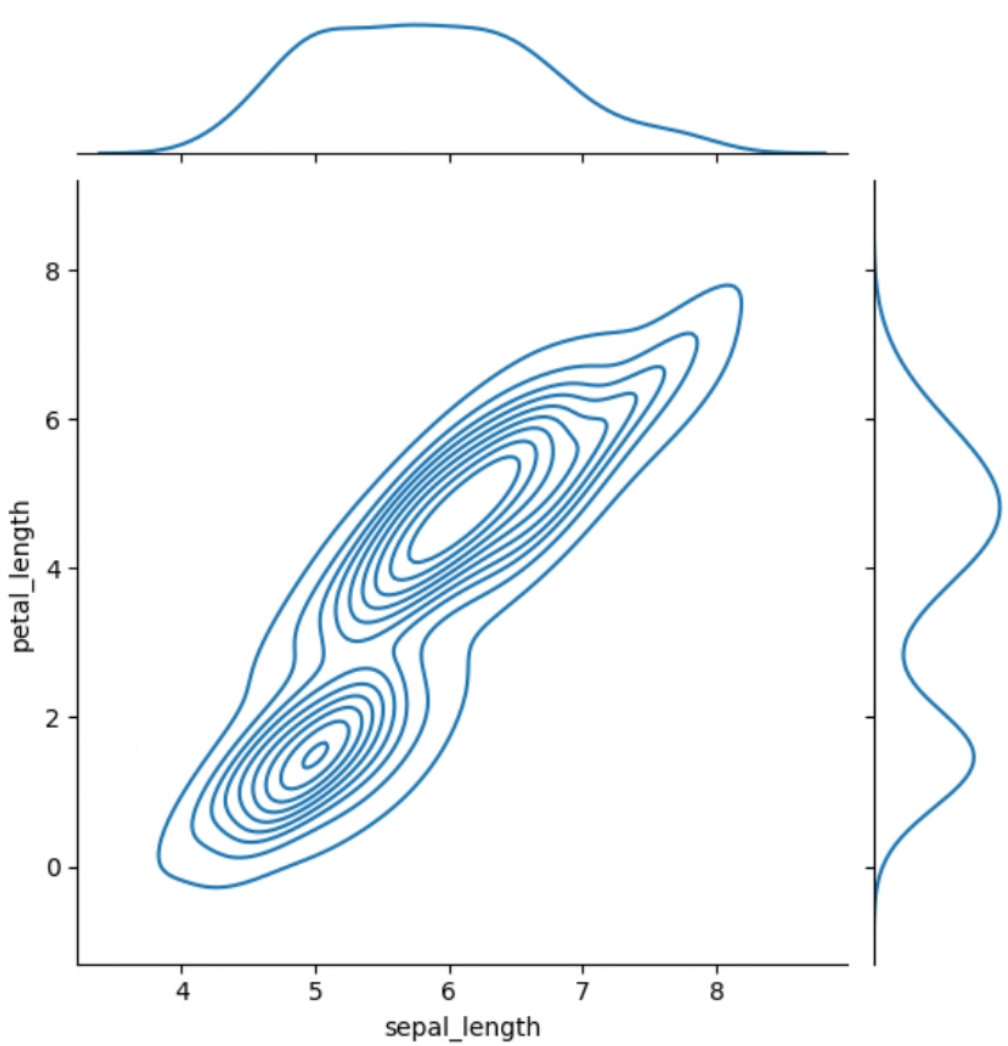


In [70]:

```
sns.jointplot(data = irs,x = "sepal_length",y = "petal_length"
              ,kind = "kde")
```

Out[70]:

```
<seaborn.axisgrid.JointGrid at 0x2427e9c1ea0>
```

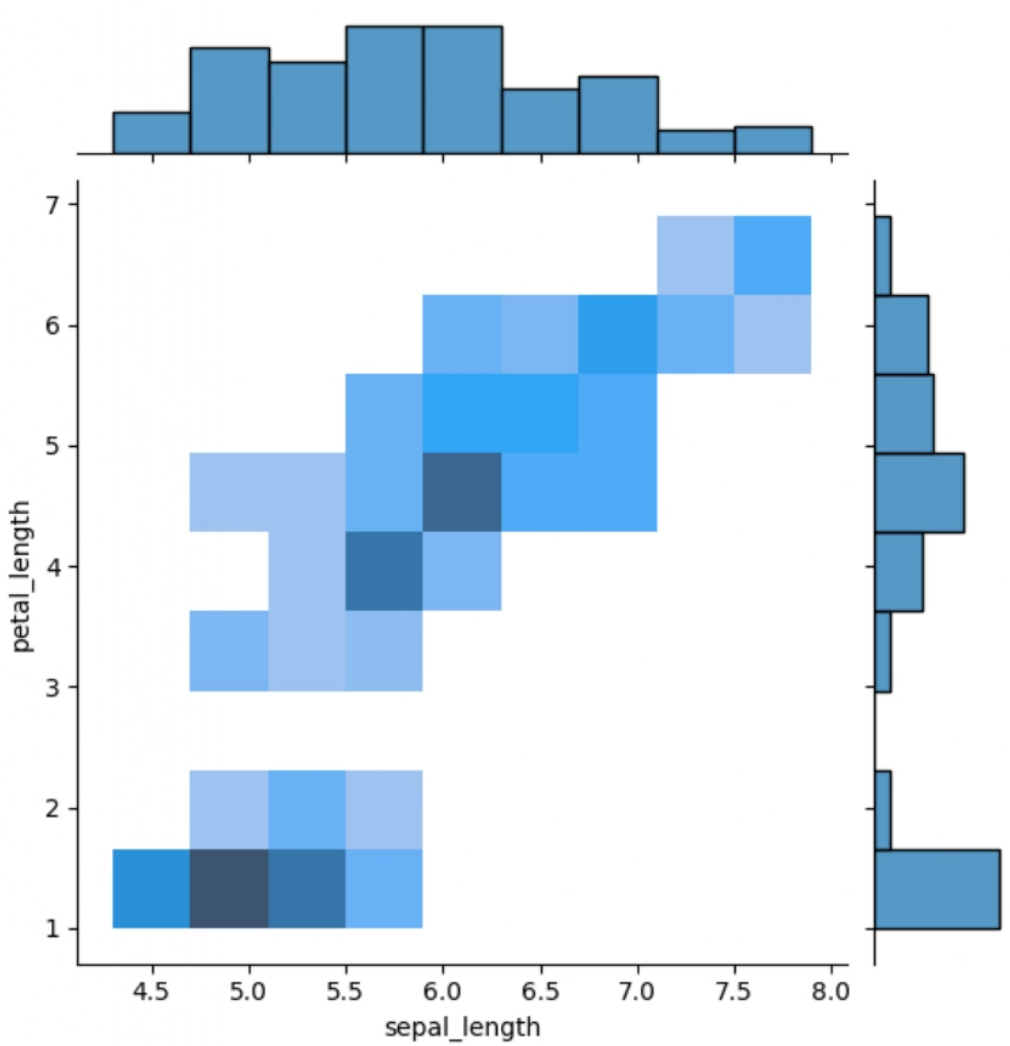


In [71]:

```
sns.jointplot(data = irs,x = "sepal_length",y = "petal_length"
              ,kind = "hist")
```

Out[71]:

```
<seaborn.axisgrid.JointGrid at 0x2427bf5dc90>
```

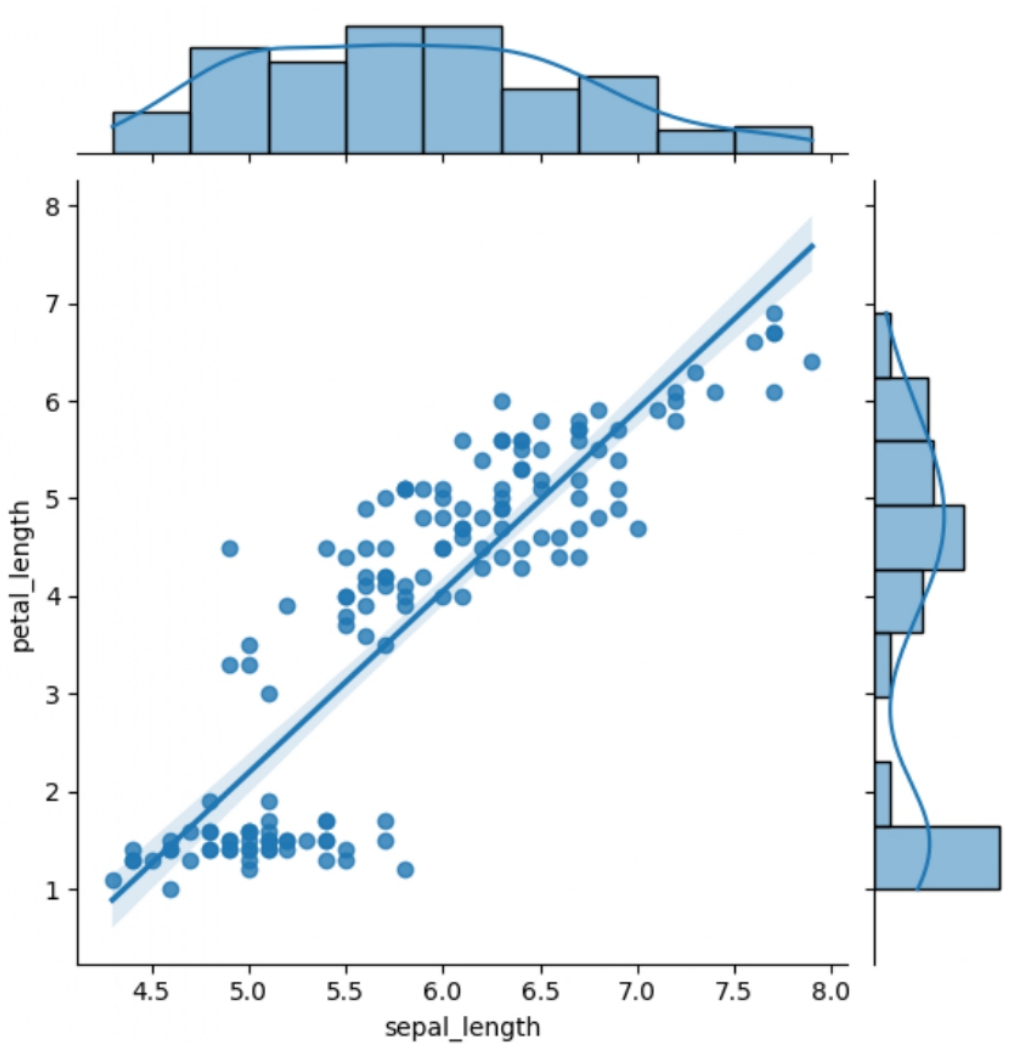


In [72]:

```
sns.jointplot(data = irs,x = "sepal_length",y = "petal_length"
              ,kind = "reg")
```

Out[72]:

```
<seaborn.axisgrid.JointGrid at 0x2427c0cca00>
```

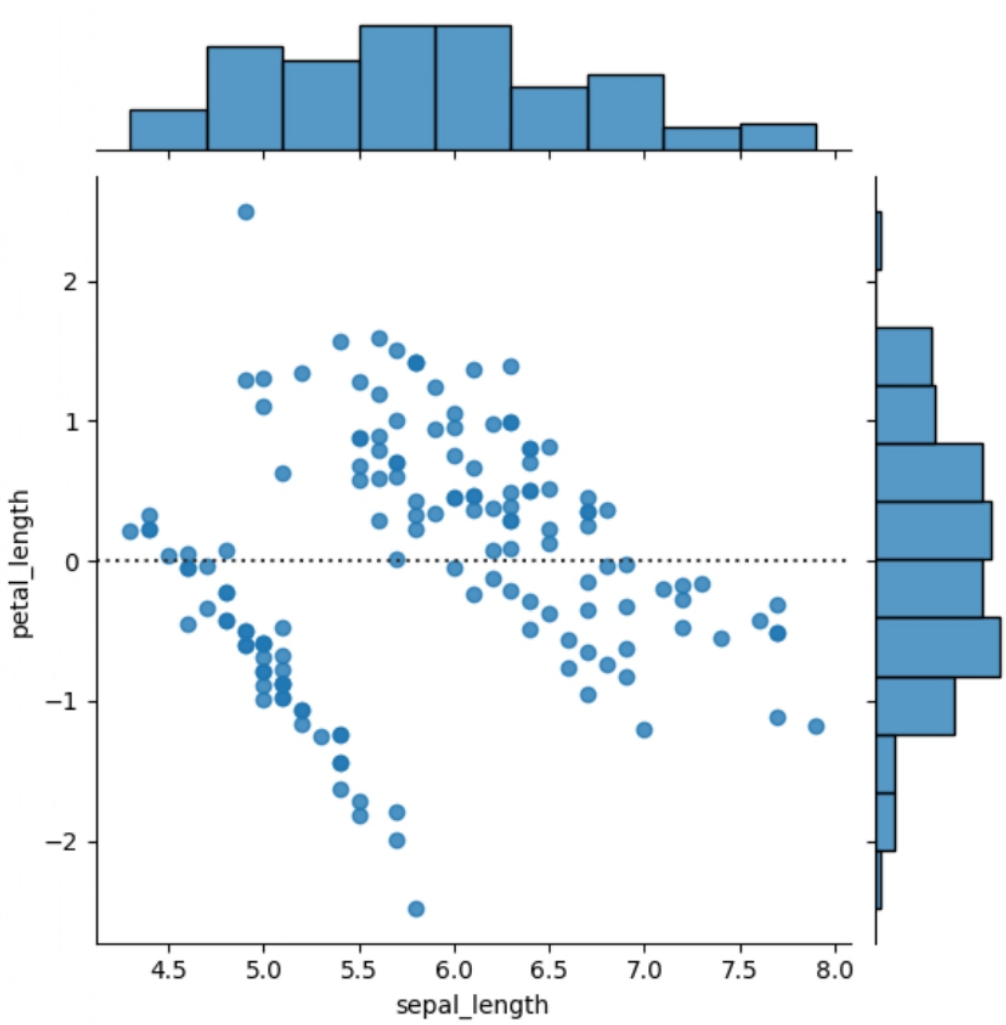


In [73]:

```
sns.jointplot(data = irs,x = "sepal_length",y = "petal_length"
              ,kind = "resid")
```

Out[73]:

```
<seaborn.axisgrid.JointGrid at 0x2427f84ad10>
```



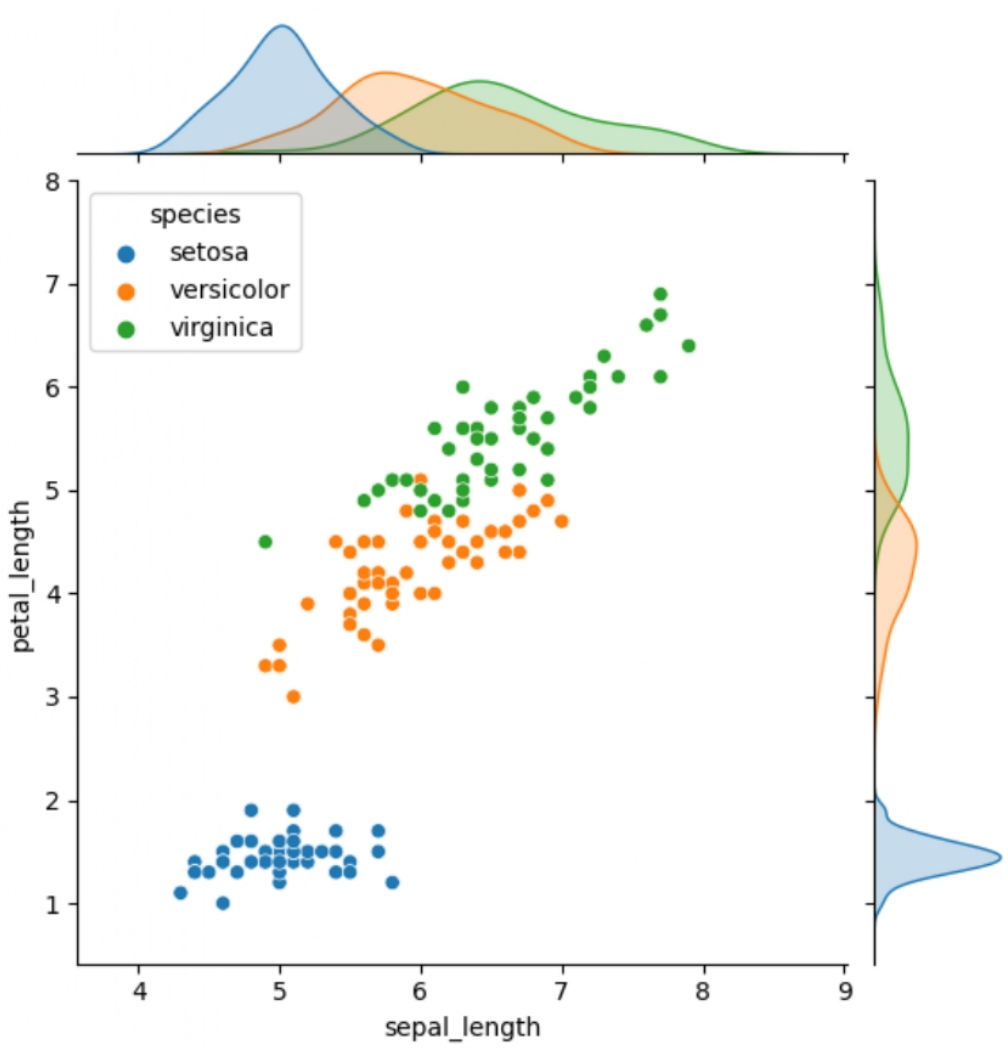
2. grouping on the based on catagorical value

In [74]:

```
sns.jointplot(data = irs,x = "sepal_length",y = "petal_length"
              ,hue = "species")
```

Out[74]:

```
<seaborn.axisgrid.JointGrid at 0x2427fdecd90>
```



M) PAIR plot

1. create a basic pair plot

In [75]:

```
irs.head()
```

Out[75]:

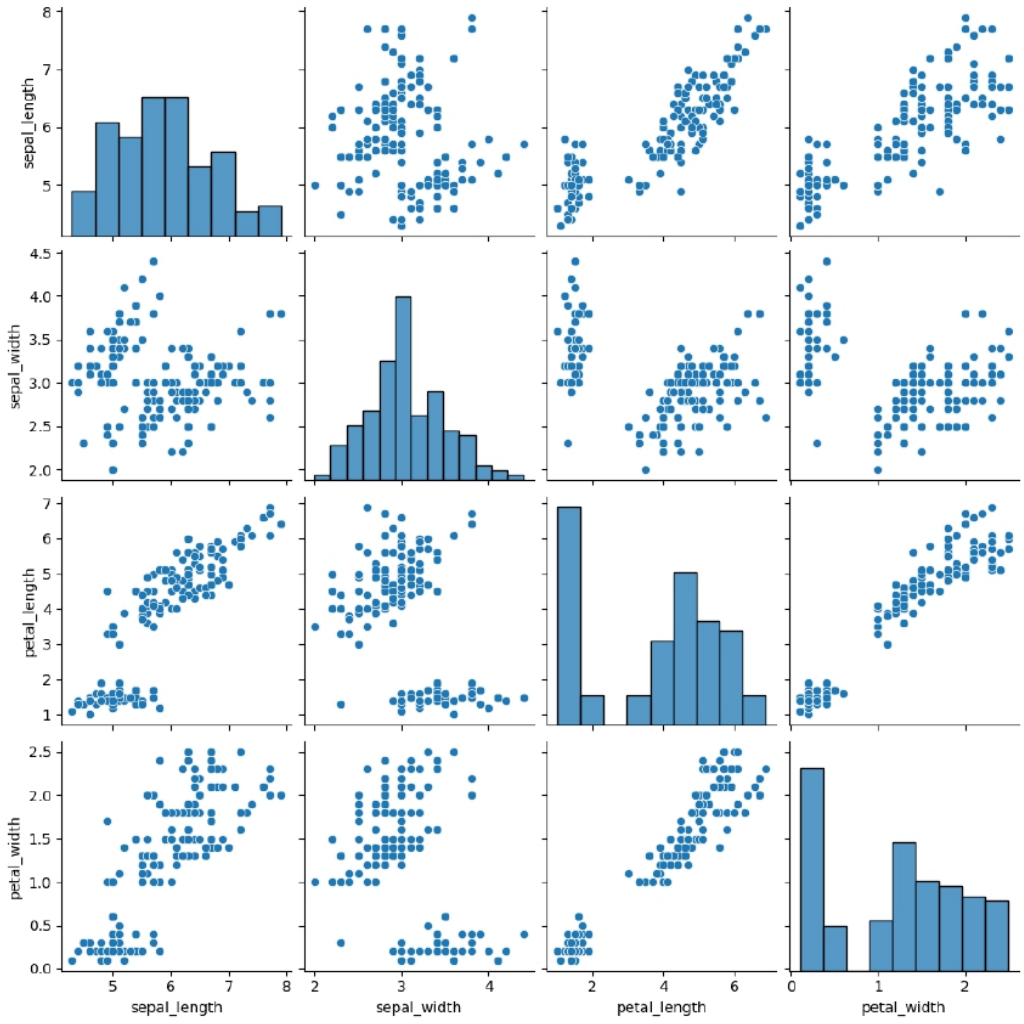
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [76]:

```
sns.pairplot(data = irs)
```

Out[76]:

```
<seaborn.axisgrid.PairGrid at 0x242001f2ce0>
```



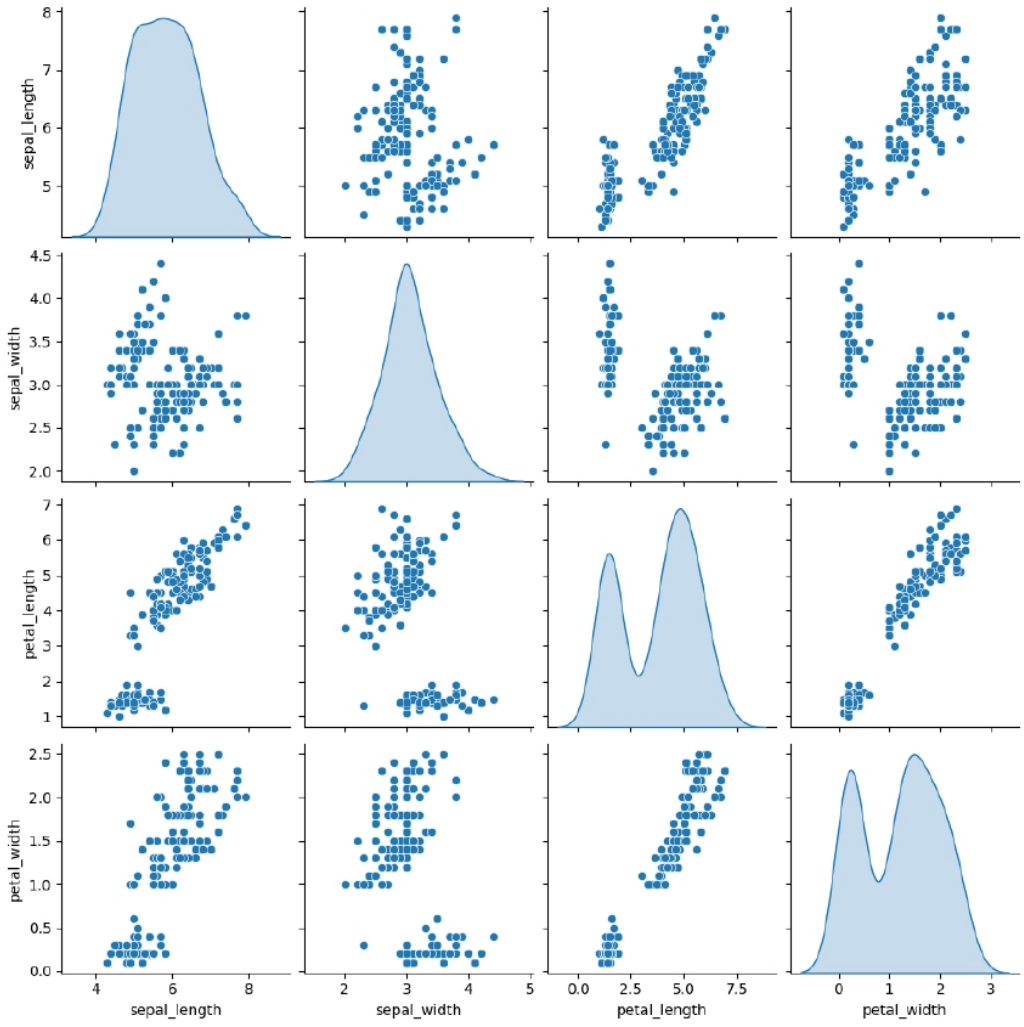
2.changing the diagonal plot KIND to KDE,HIST or None

In [77]:

```
sns.pairplot(data = irs,diag_kind = "kde")
```

Out[77]:

```
<seaborn.axisgrid.PairGrid at 0x24201fd9d50>
```



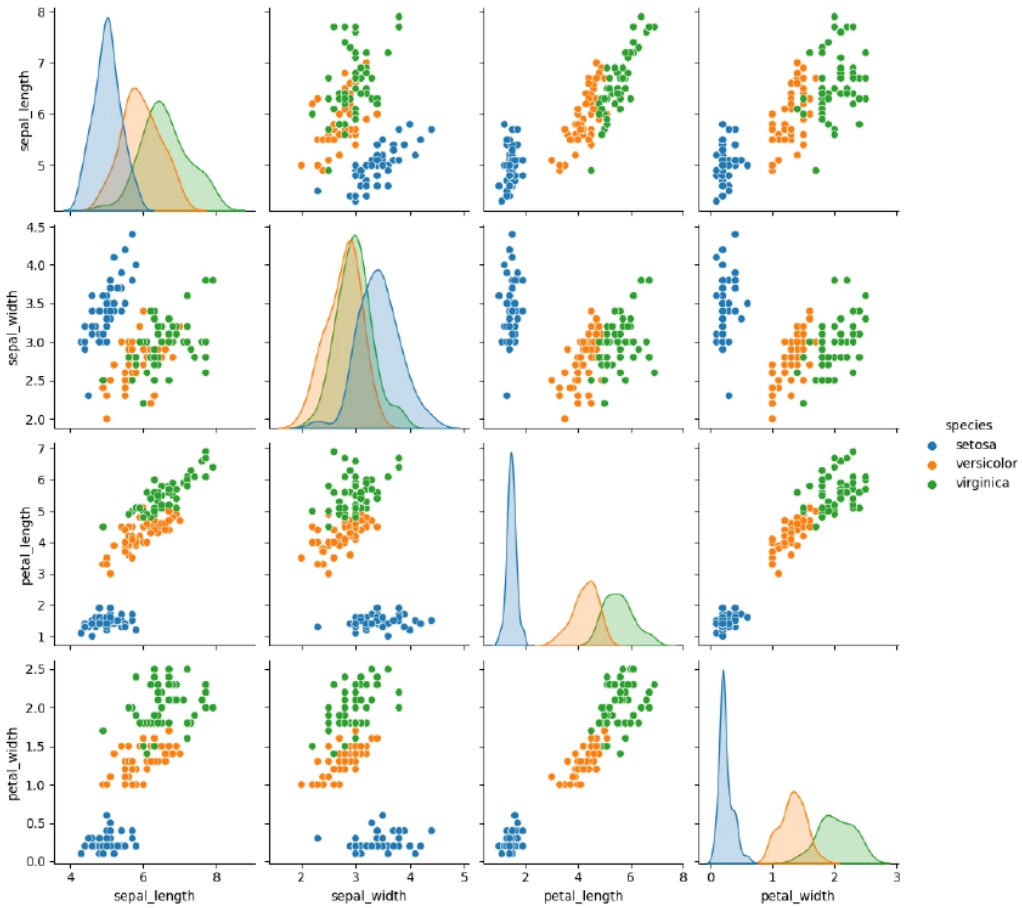
3. adding hue in the pair plot

In [78]:

```
sns.pairplot(data = irs,diag_kind = "kde",hue = 'species')
```

Out[78]:

```
<seaborn.axisgrid.PairGrid at 0x24202ec3eb0>
```



In [79]:

```
# Relational plot
# 1.scatter plot
# 2.line plot
# rel plot
```

N) Scatter plot

1. create a basic seaborn plot

In [80]:

```
m = pd.read_excel("mart_train_sample.xlsx")
```

In [81]:

```
m.head()
```

Out[81]:

	Item_ID	Item_W	Item_Type	Item_MRP	Outlet_ID	Outlet_Year	Ou
0	FDN27	9.437510	Snack Foods	195.313244	OUT046	1997	
1	FDW35	9.033762	Fruits and Vegetables	36.371449	OUT035	2003	
2	FDS28	20.884849	Frozen Foods	70.318167	OUT013	2008	
3	FDC09	19.976300	Snack Foods	121.723264	OUT018	2009	
4	FDX04	10.824812	Snack Foods	264.673302	OUT035	2004	

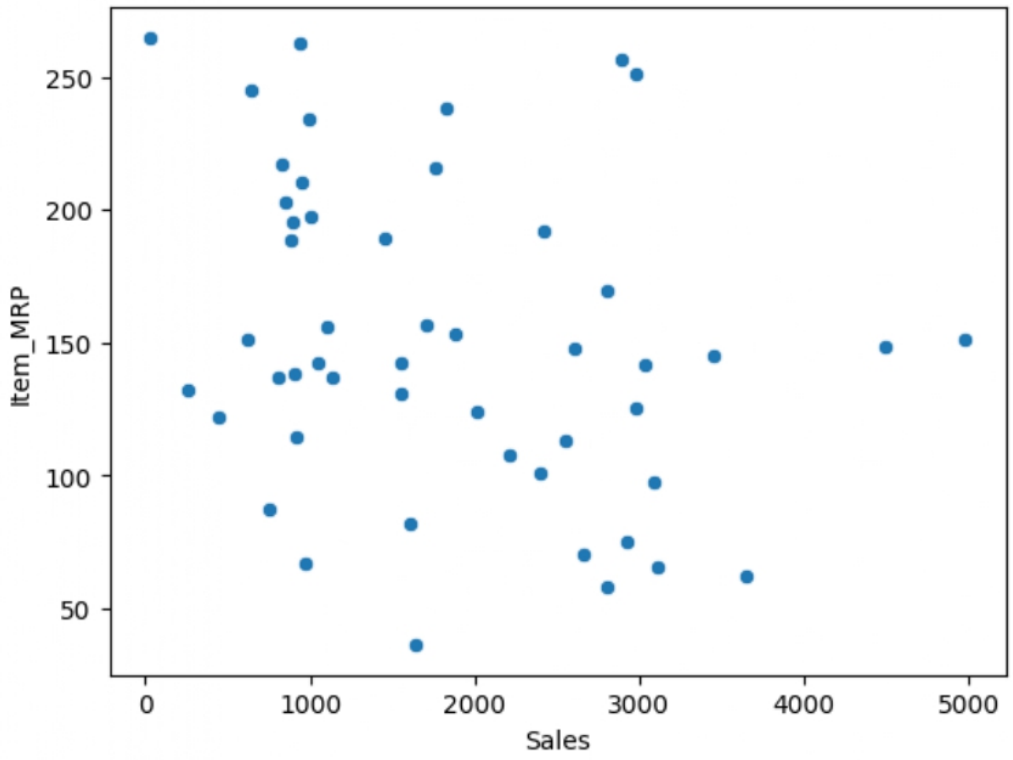
1.create a scatter plot

In [82]:

```
sns.scatterplot(data = m,x = "Sales",y = "Item_MRP")
```

Out[82]:

```
<AxesSubplot: xlabel='Sales', ylabel='Item_MRP'>
```



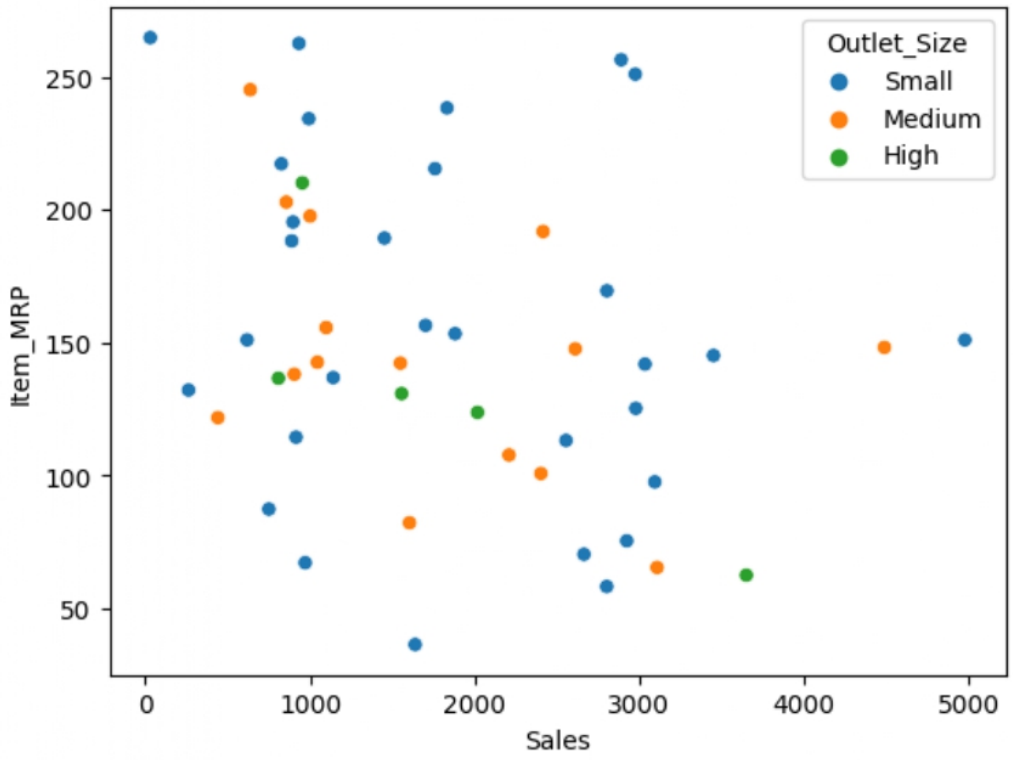
2. group basis on a catagorical variable using HUE

In [83]:

```
sns.scatterplot(data = m,x = "Sales",y = "Item_MRP",hue ="Outlet_Size" )
```

Out[83]:

```
<AxesSubplot: xlabel='Sales', ylabel='Item_MRP'>
```



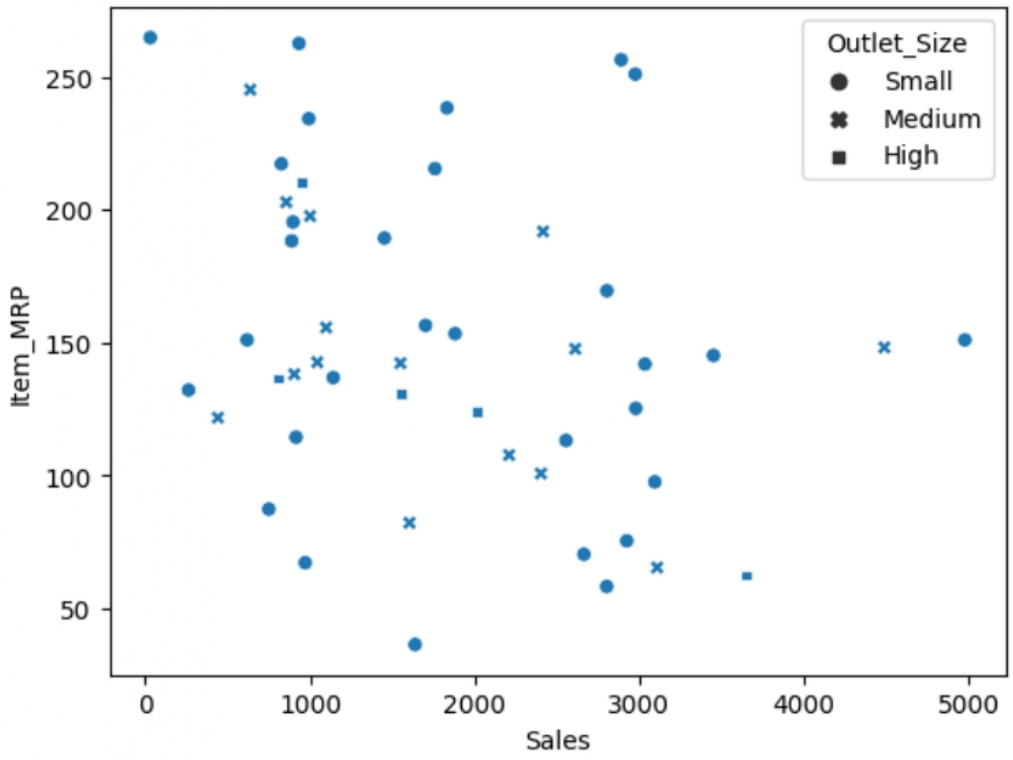
3. group basis on a catagorical variable using Style

In [84]:

```
sns.scatterplot(data = m,x = "Sales",y = "Item_MRP",style="Outlet_Size" )
```

Out[84]:

```
<AxesSubplot: xlabel='Sales', ylabel='Item_MRP'>
```



o) LINE plot

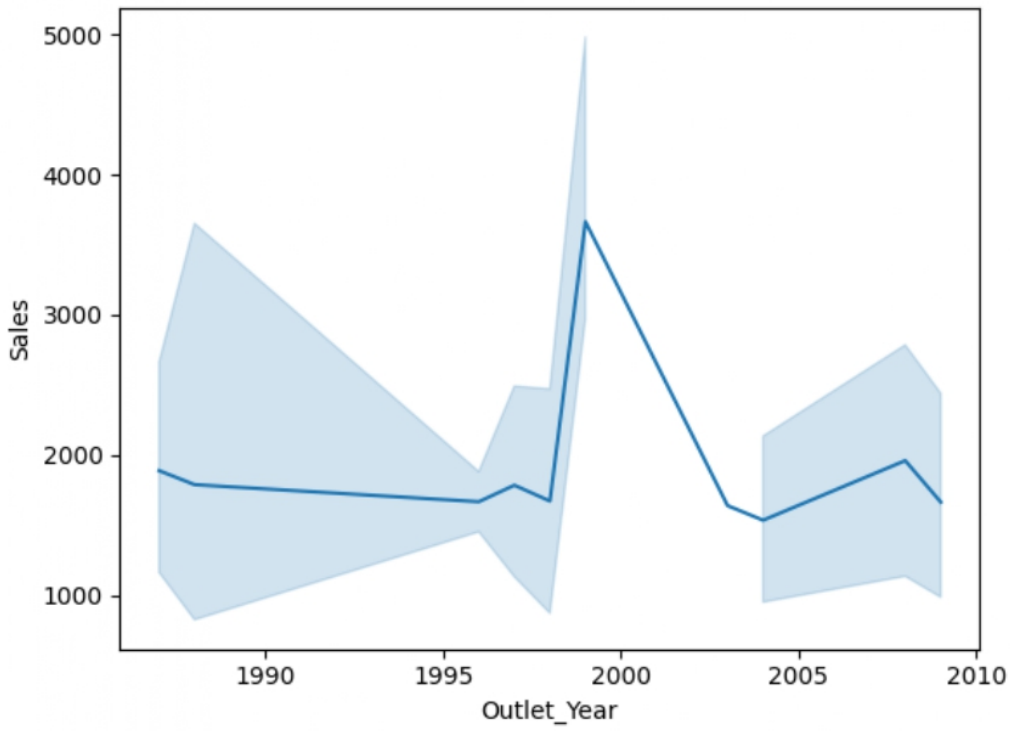
1.create a basic line plot

In [85]:

```
sns.lineplot(data = m,x = "Outlet_Year",y = "Sales")
```

Out[85]:

```
<AxesSubplot: xlabel='Outlet_Year', ylabel='Sales'>
```



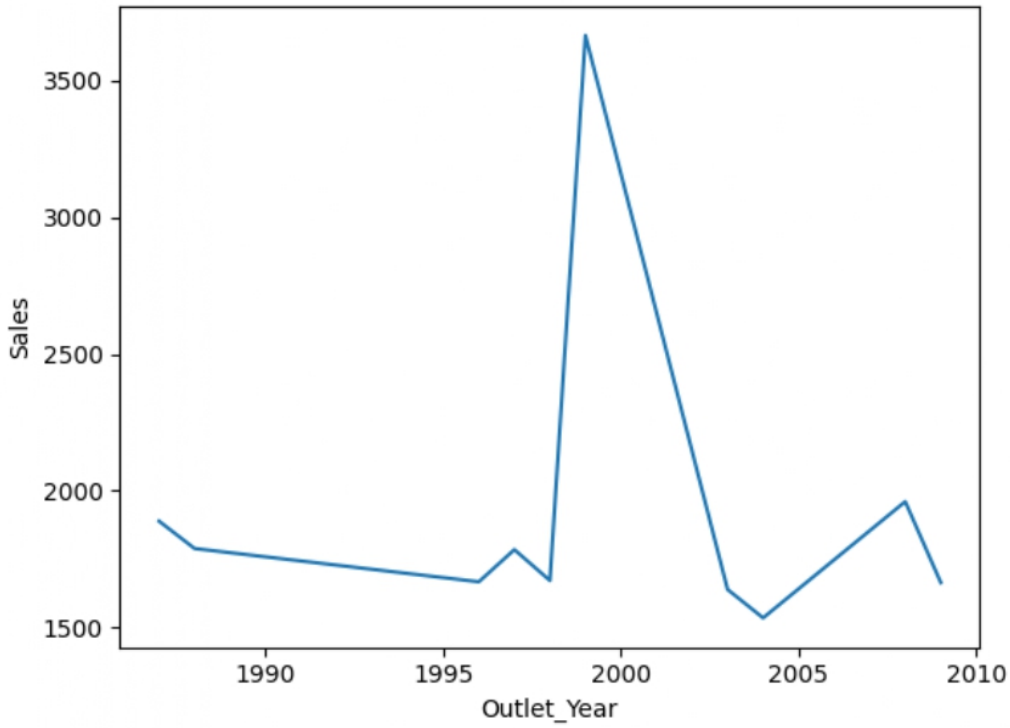
In [86]:

```
sns.lineplot(data = m,x = "Outlet_Year",y = "Sales",ci = None)
```

```
C:\Users\joypa\AppData\Local\Temp\ipykernel_7164\340264817
6.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for
the same effect.

  sns.lineplot(data = m,x = "Outlet_Year",y = "Sales",ci =
None)
```

Out[86]:

```
<AxesSubplot: xlabel='Outlet_Year', ylabel='Sales'>
```



In [87]:

```
m.columns
```

Out[87]:

```
Index(['Item_ID', 'Item_W', 'Item_Type', 'Item_MRP', 'Outl
et_ID',
       'Outlet_Year', 'Outlet_Size', 'Outlet_Location_Typ
e', 'Sales', 'Tier'],
      dtype='object')
```

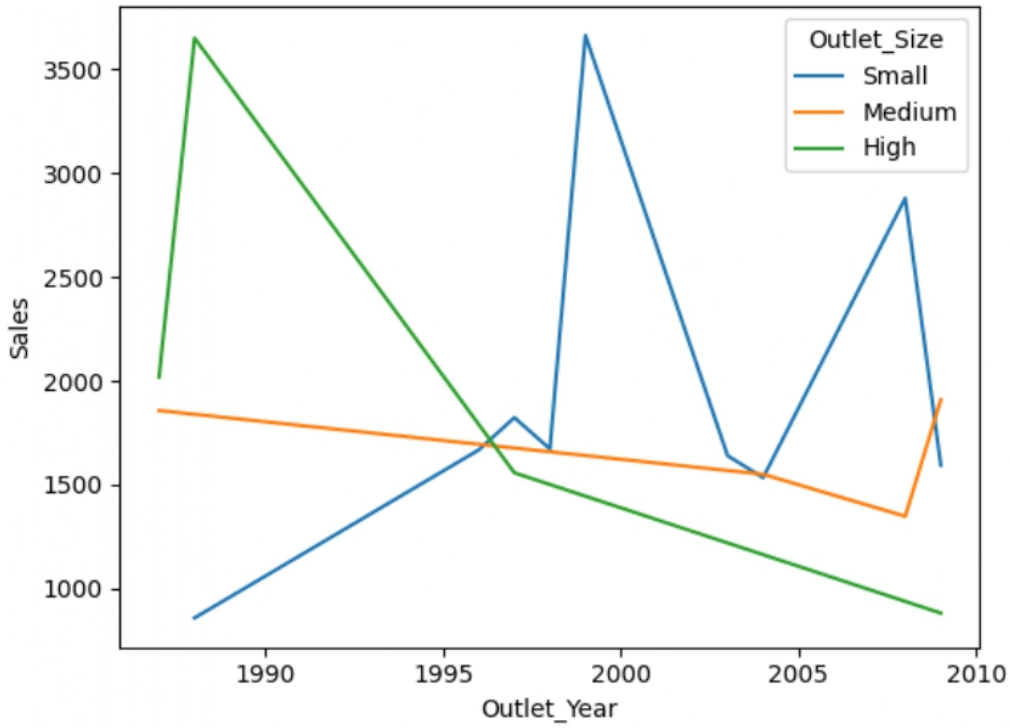
In [88]:

```
sns.lineplot(data = m,x = "Outlet_Year",y = "Sales",ci = None,hue = "Outl
```

```
C:\Users\joypa\AppData\Local\Temp\ipykernel_7164\97926062
9.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for
the same effect.

  sns.lineplot(data = m,x = "Outlet_Year",y = "Sales",ci =
None,hue = "Outlet_Size")
```

Out[88]:

```
<AxesSubplot: xlabel='Outlet_Year', ylabel='Sales'>
```



P) REL plot

In [89]:

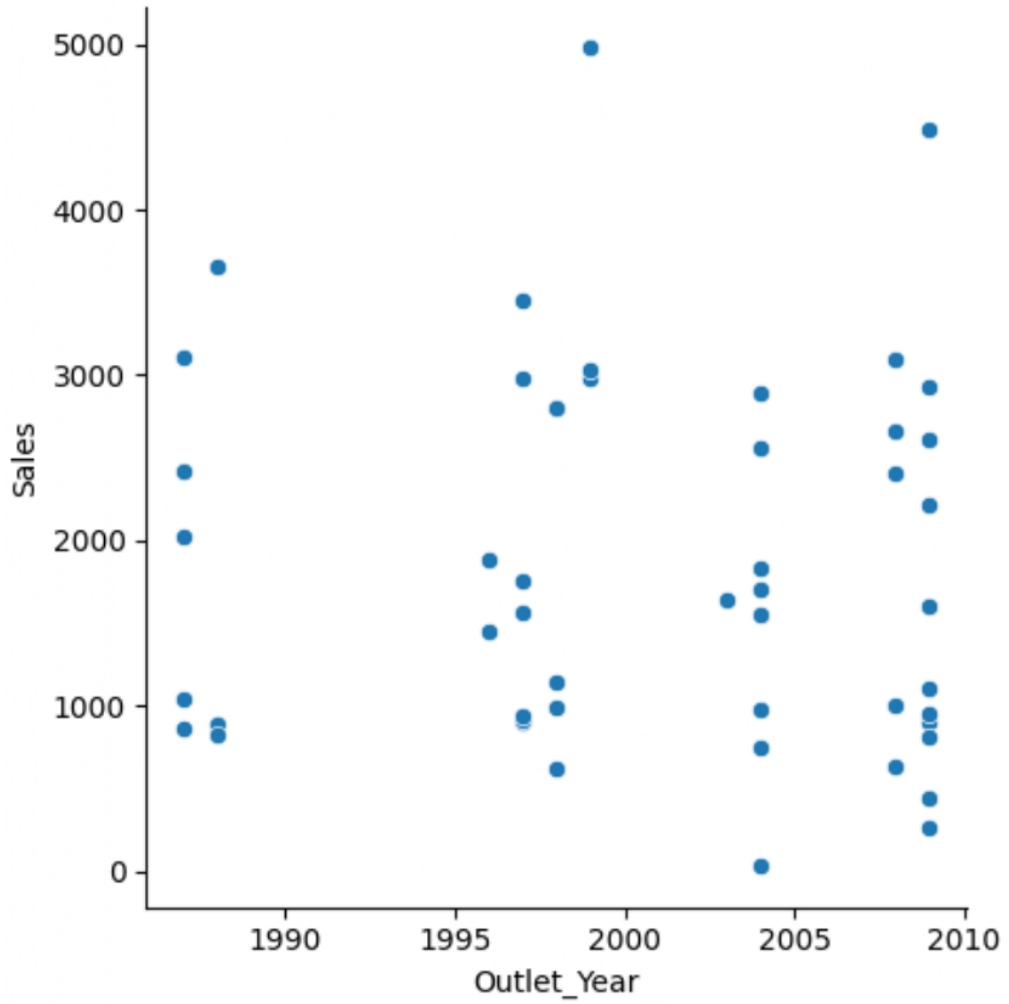
```
# REL plot  = Relational Plot
```

In [90]:

```
sns.relplot(data = m,x = "Outlet_Year",y = "Sales")
```

Out[90]:

```
<seaborn.axisgrid.FacetGrid at 0x242040075b0>
```

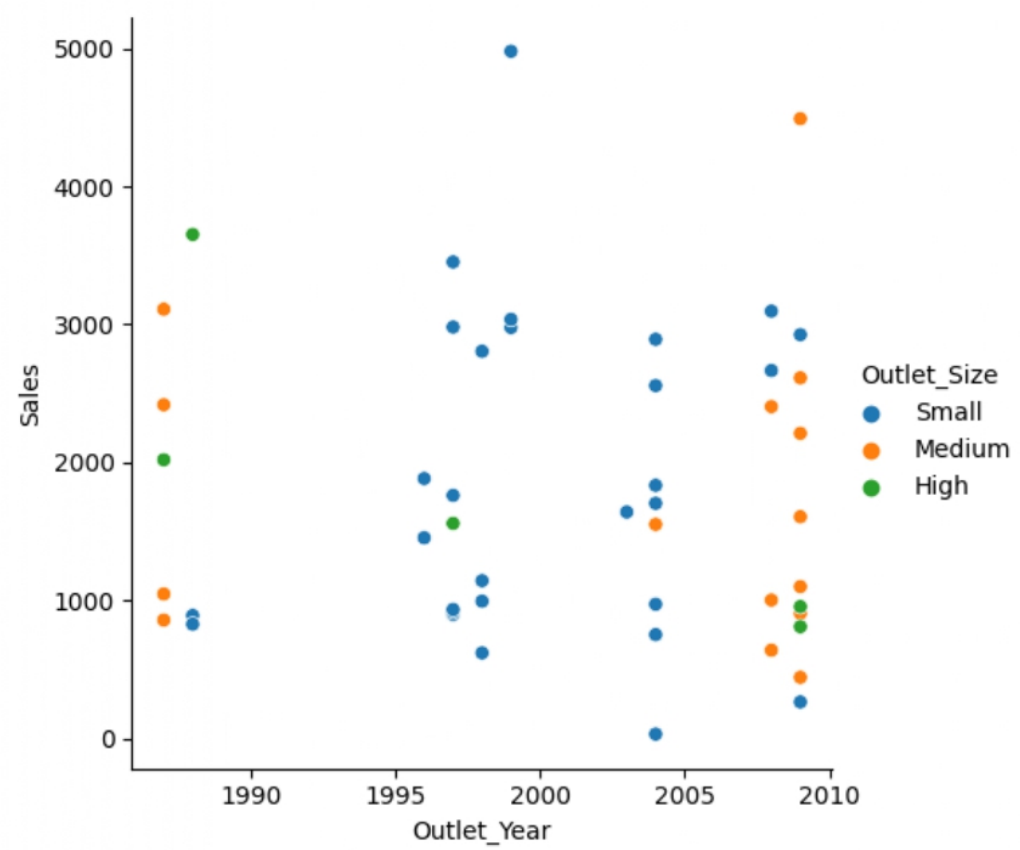


In [91]:

```
sns.relplot(data = m,x = "Outlet_Year",y = "Sales",hue = "Outlet_Size")
```

Out[91]:

```
<seaborn.axisgrid.FacetGrid at 0x24202dad9c0>
```

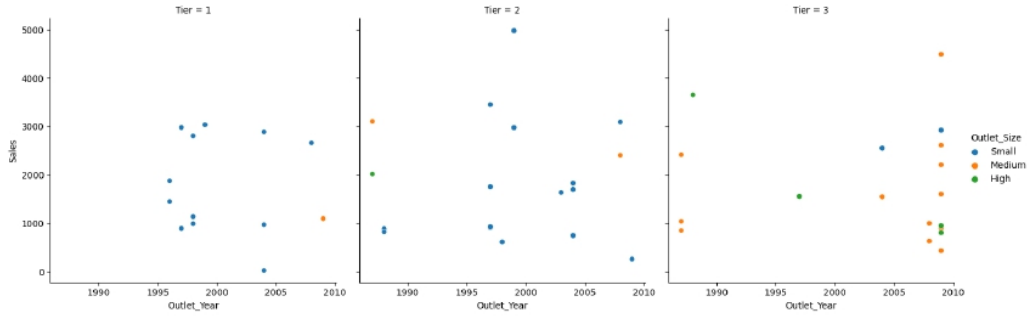


In [92]:

```
sns.relplot(data = m,x = "Outlet_Year",y = "Sales",hue = "Outlet_Size",cc
```

Out[92]:

```
<seaborn.axisgrid.FacetGrid at 0x24202c96800>
```



P) REG plot

In [93]:

```
# reg plot = Regression plot
```

In [94]:

```
tip = sns.load_dataset("tips")
```

In [95]:

```
tip.head()
```

Out[95]:

	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

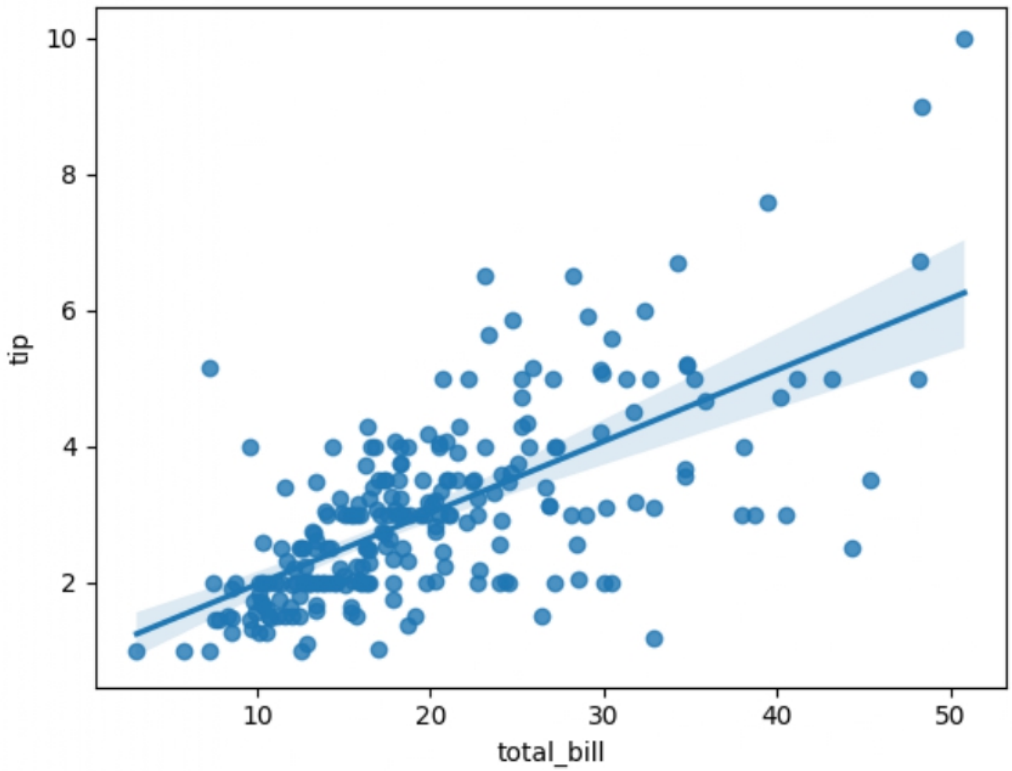
1. create a first ref plot\

In [96]:

```
sns.regplot(data  = tip,x = "total_bill",y = "tip")
```

Out[96]:

```
<AxesSubplot: xlabel='total_bill', ylabel='tip'>
```

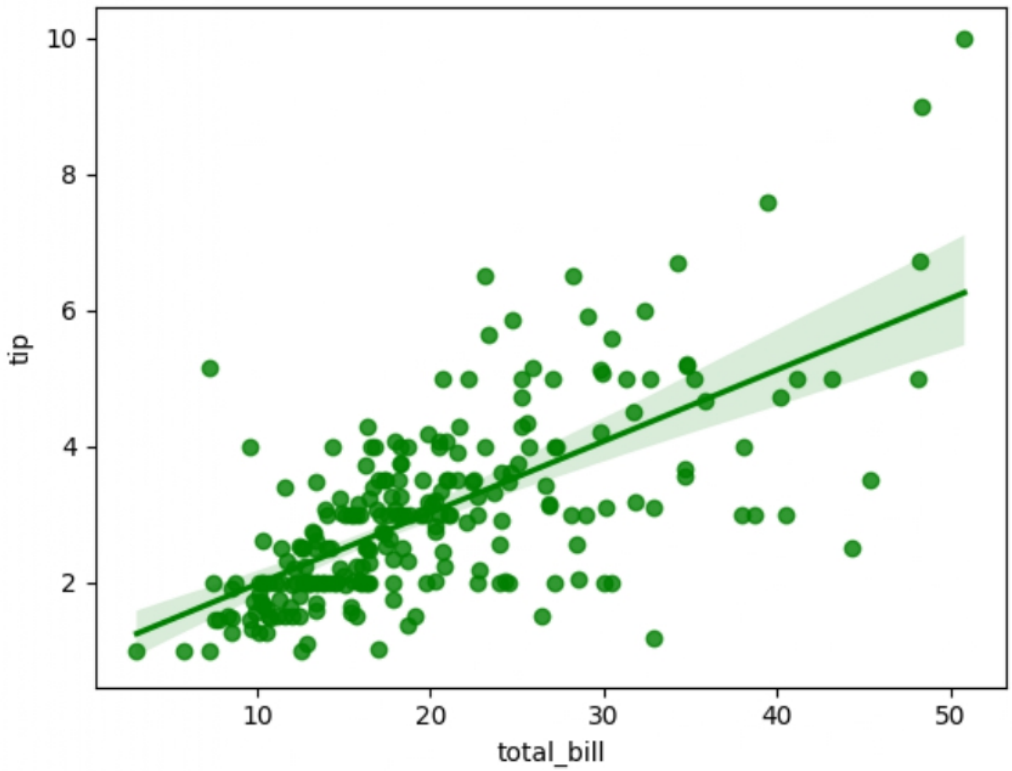


In [97]:

```
sns.regplot(data  = tip,x = "total_bill",y = "tip",color= "green")
```

Out[97]:

```
<AxesSubplot: xlabel='total_bill', ylabel='tip'>
```

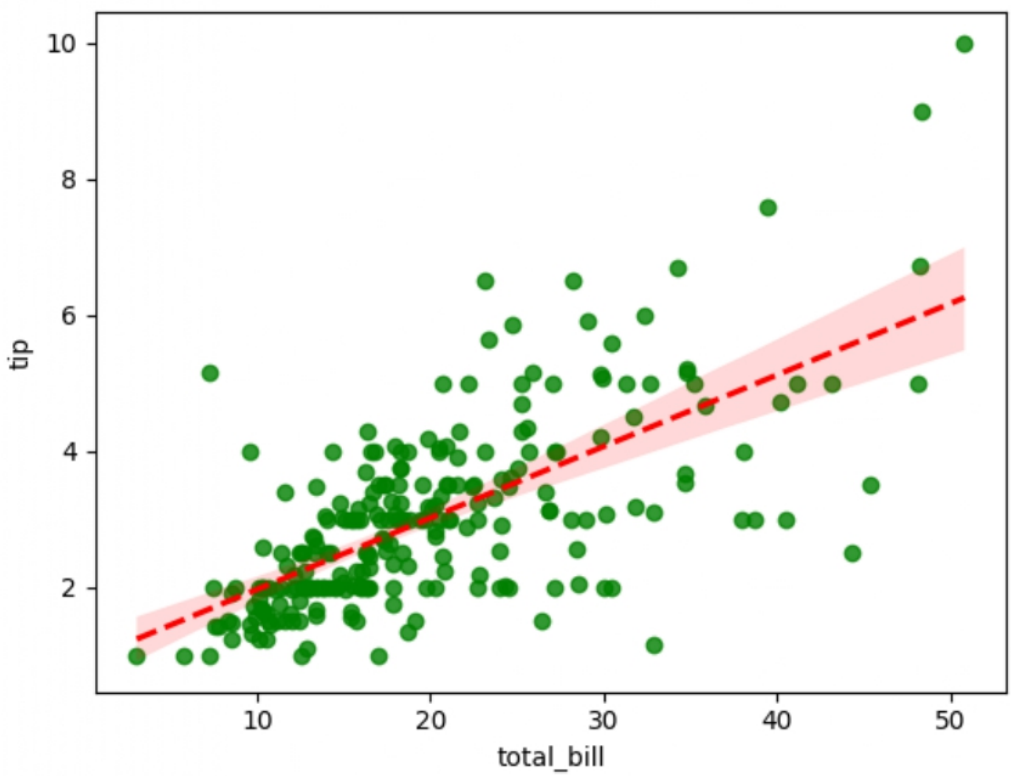


In [98]:

```
sns.regplot(data  = tip,x = "total_bill",y = "tip",color= "green"
           ,line_kws = {"color":"red","linestyle":"--"})
```

Out[98]:

```
<AxesSubplot: xlabel='total_bill', ylabel='tip'>
```



Q) HEATMAPS

In [99]:

```
# This plot only apply for numeric value not catagorical value
```

In [ ]:

```

```