

```
In [1]: import pandas as pd
import mysql.connector as mysql
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
```

```
In [2]: db_connection_str = "mysql://root:252000@localhost/Big_mart_dataset"
db_connection = create_engine(db_connection_str)
```

```
In [3]: df = pd.read_sql('SELECT * FROM Big_mart', con=db_connection)
df.head(10)
```

Out[3]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8090	OU
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OU
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OU
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OU
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OU
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008	OU
6	FDO10	13.650	Regular	0.012741	Snack Foods	57.6588	OU
7	FDH17	16.200	Regular	0.016687	Frozen Foods	96.9726	OU
8	FDU28	19.200	Regular	0.094450	Frozen Foods	187.8210	OU
9	FDY07	11.800	Low Fat	0.000000	Fruits and Vegetables	45.5402	OU

## 1- what is the highest tier ( cluster ) performed a high sales?

```
In [4]: result1 = pd.read_sql('SELECT Outlet_Location_Type, sum(Item_Outlet_Sales) as Total
GROUP BY Outlet_Location_Type ORDER BY Total DESC LIMIT 1', con=db_connection)
result1
```

Out[4]:

	Outlet_Location_Type	Total
0	Tier 2	6.472314e+06

## 2- what is the top 5 products sales in each category in terms of fats

## categories?

```
In [5]: mydb = mysql.connect(
        host="localhost",
        user = "root",
        passwd = "252000",
        database="Big_mart_dataset"
    )

    mycursor = mydb.cursor()

    sql = "UPDATE Big_mart SET Item_Fat_Content = 'Low Fat' WHERE Item_Fat_Content IN ('Low', 'Reg', 'Reg_Low')"
    sql2 = "UPDATE Big_mart SET Item_Fat_Content = 'Regular' WHERE Item_Fat_Content = 'Reg'"

    mycursor.execute(sql)
    mycursor.execute(sql2)

    mydb.commit()
```

```
In [6]: result2 = pd.read_sql('(SELECT Item_Fat_Content, Item_Type, sum(Item_Outlet_Sales) as Total
    FROM Big_mart WHERE Item_Fat_Content = 'Low Fat' GROUP BY Item_Type ORDER BY Total)
    UNION (SELECT Item_Fat_Content, Item_Type, sum(Item_Outlet_Sales) as Total
    FROM Big_mart WHERE Item_Fat_Content = 'Regular' GROUP BY Item_Type ORDER BY Total)'
    result2
```

Out[6]:

	Item_Fat_Content	Item_Type	Total
0	Low Fat	Household	1.659037e+06
1	Low Fat	Snack Foods	1.317829e+06
2	Low Fat	Fruits and Vegetables	1.062656e+06
3	Low Fat	Health and Hygiene	8.351709e+05
4	Low Fat	Frozen Foods	7.772489e+05
5	Regular	Fruits and Vegetables	1.157322e+06
6	Regular	Snack Foods	8.762154e+05
7	Regular	Frozen Foods	7.169607e+05
8	Regular	Baking Goods	5.617694e+05
9	Regular	Canned	5.456750e+05

## 3- the relation between the outlet size and the sales?

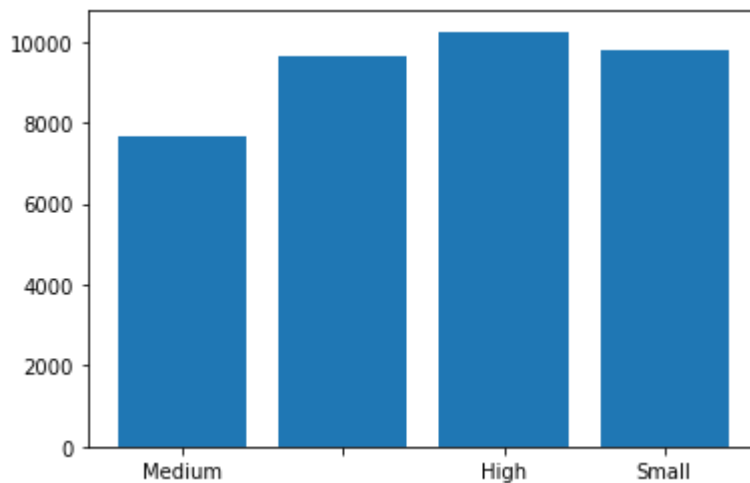
```
In [7]: result3 = pd.read_sql('SELECT Outlet_Size, Item_Outlet_Sales FROM Big_mart', con=
result3.head()
```

Out[7]:

	Outlet_Size	Item_Outlet_Sales
0	Medium	3735.140
1	Medium	443.423
2	Medium	2097.270
3		732.380
4	High	994.705

```
In [8]: plt.bar(result3.Outlet_Size,result3.Item_Outlet_Sales)
```

Out[8]: <BarContainer object of 7060 artists>



**As shown the Outlet\_Size "High" has the highest sales**

```
In [9]: result3.Outlet_Size.replace(['','Small','Medium','High'],[0,1,2,3],inplace=True)
result3.head()
```

Out[9]:

	Outlet_Size	Item_Outlet_Sales
0	2	3735.140
1	2	443.423
2	2	2097.270
3	0	732.380
4	3	994.705

```
In [10]: result3.corr()
```

```
Out[10]:
```

	Outlet_Size	Item_Outlet_Sales
Outlet_Size	1.000000	0.101286
Item_Outlet_Sales	0.101286	1.000000

**The relation between Outlet\_Size & Item\_Outlet\_Sales is not strong**

#### 4- Relation between item visibility and item sales?

```
In [11]: result4 = pd.read_sql('SELECT Item_Visibility, Item_Outlet_Sales FROM Big_mart',  
result4.head(10))
```

```
Out[11]:
```

	Item_Visibility	Item_Outlet_Sales
0	0.016047	3735.140
1	0.019278	443.423
2	0.016760	2097.270
3	0.000000	732.380
4	0.000000	994.705
5	0.000000	556.609
6	0.012741	343.553
7	0.016687	1076.600
8	0.094450	4710.540
9	0.000000	1516.030

```
In [12]: result4.corr()
```

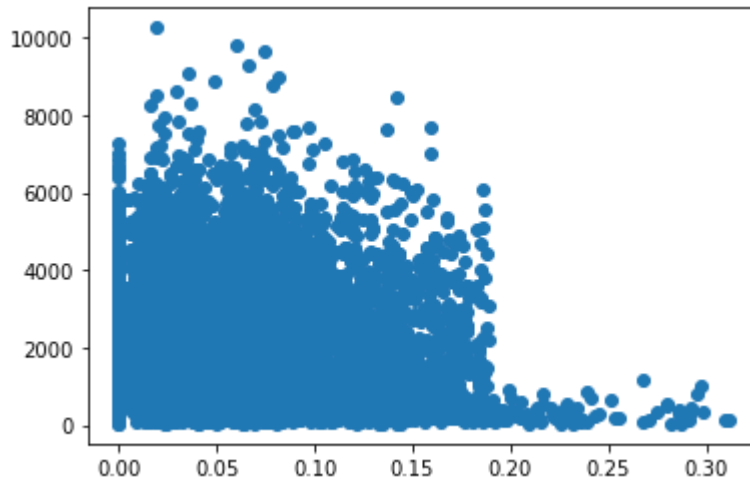
```
Out[12]:
```

	Item_Visibility	Item_Outlet_Sales
Item_Visibility	1.000000	-0.085334
Item_Outlet_Sales	-0.085334	1.000000

**The relation between Item\_Visibility & Item\_Outlet\_Sales is not strong**

```
In [13]: plt.scatter(result4.Item_Visibility,result4.Item_Outlet_Sales)
```

```
Out[13]: <matplotlib.collections.PathCollection at 0x1609b896c40>
```



## 5- Total sales for each product for each vendor?

```
In [14]: db_connection_str = "mysql://root:252000@localhost/classicmodels"
db_connection = create_engine(db_connection_str)
```

```
In [15]: df = pd.read_sql('''SELECT productVendor, productName, sum(amount) as Total FROM
    orders as o, customers as c, payments as pa WHERE pr.productCode = od.pro
    od.orderNumber = o.orderNumber AND o.customerNumber = c.customerNumber AN
    AND status = "Shipped" GROUP BY productName, productVendor ORDER BY produ
```

```
In [16]: df = df.set_index(df['productName'])
df.head(15)
```

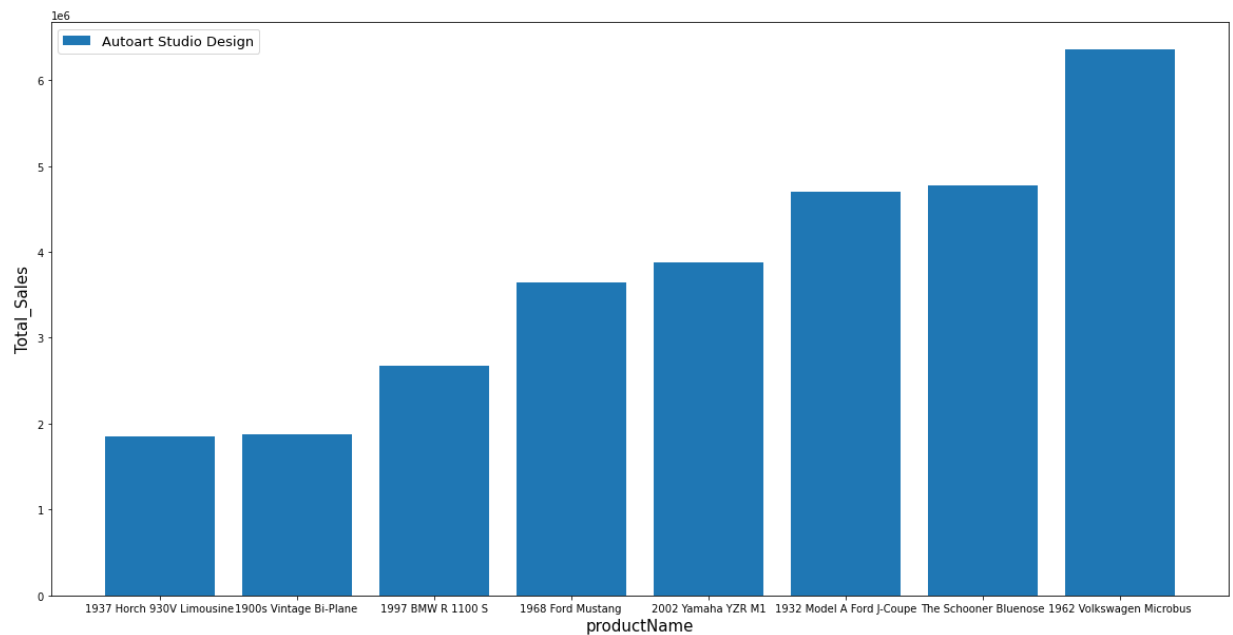
Out[16]:

	productVendor	productName	Total
productName			
1937 Horch 930V Limousine	Autoart Studio Design	1937 Horch 930V Limousine	1849456.67
1900s Vintage Bi-Plane	Autoart Studio Design	1900s Vintage Bi-Plane	1872298.25
1997 BMW R 1100 S	Autoart Studio Design	1997 BMW R 1100 S	2671616.54
1968 Ford Mustang	Autoart Studio Design	1968 Ford Mustang	3639265.70
2002 Yamaha YZR M1	Autoart Studio Design	2002 Yamaha YZR M1	3873140.49
1932 Model A Ford J-Coupe	Autoart Studio Design	1932 Model A Ford J-Coupe	4698226.35
The Schooner Bluenose	Autoart Studio Design	The Schooner Bluenose	4779581.62
1962 Volkswagen Microbus	Autoart Studio Design	1962 Volkswagen Microbus	6357436.57
1940 Ford Delivery Sedan	Carousel DieCast Legends	1940 Ford Delivery Sedan	3410857.15
The Titanic	Carousel DieCast Legends	The Titanic	3608636.16
1913 Ford Model T Speedster	Carousel DieCast Legends	1913 Ford Model T Speedster	3898100.84
1966 Shelby Cobra 427 S/C	Carousel DieCast Legends	1966 Shelby Cobra 427 S/C	4178900.25
18th century schooner	Carousel DieCast Legends	18th century schooner	4436503.98
1982 Camaro Z28	Carousel DieCast Legends	1982 Camaro Z28	4543620.81
1926 Ford Fire Engine	Carousel DieCast Legends	1926 Ford Fire Engine	5421393.12

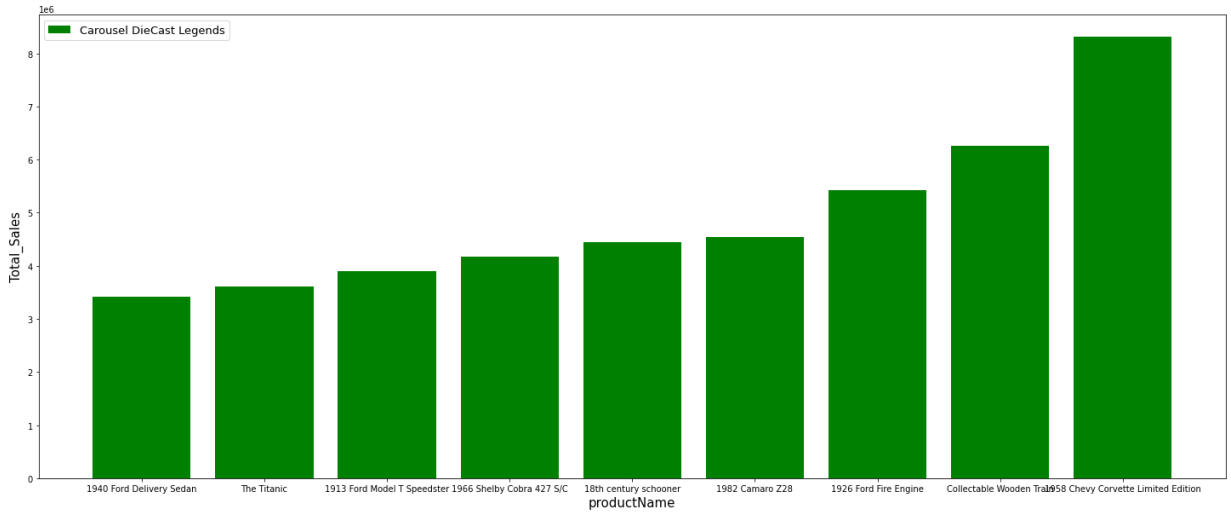
```
In [17]: labels = df.productVendor.unique()
labels
```

Out[17]: array(['Autoart Studio Design', 'Carousel DieCast Legends',  
'Classic Metal Creations', 'Exoto Designs', 'Gearbox Collectibles',  
'Highway 66 Mini Classics', 'Min Lin Diecast',  
'Motor City Art Classics', 'Red Start Diecast',  
'Second Gear Diecast', 'Studio M Art Models',  
'Unimax Art Galleries', 'Welly Diecast Productions'], dtype=object)

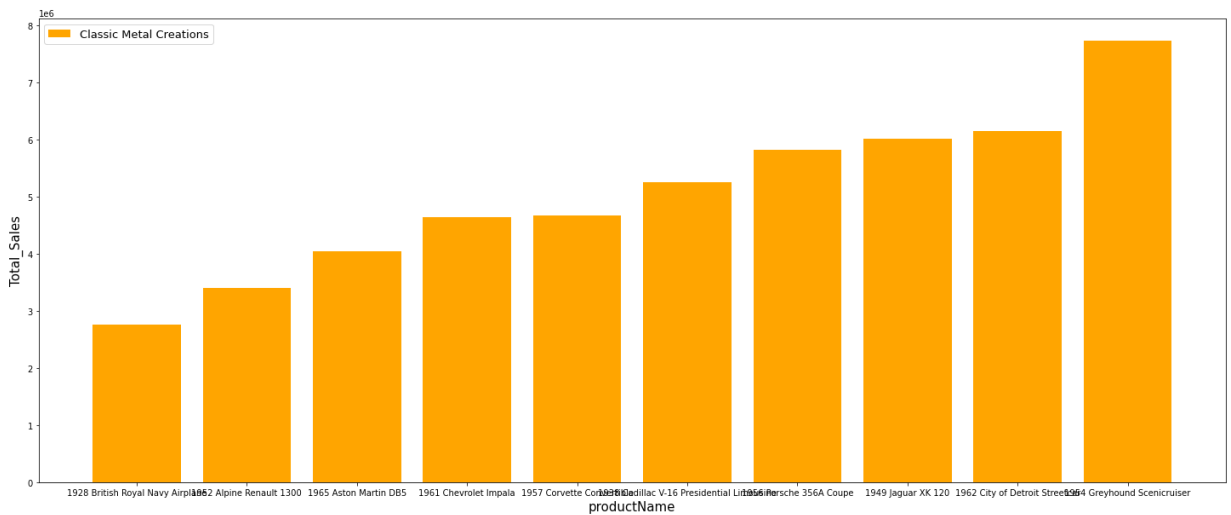
```
In [18]: plt.figure(figsize=(20,10))
plt.bar(df.index[df['productVendor']=='labels[0]'], df.Total[df['productVendor']=='labels[0]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



```
In [19]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[1]'], df.Total[df['productVendor']=='labels[1]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```

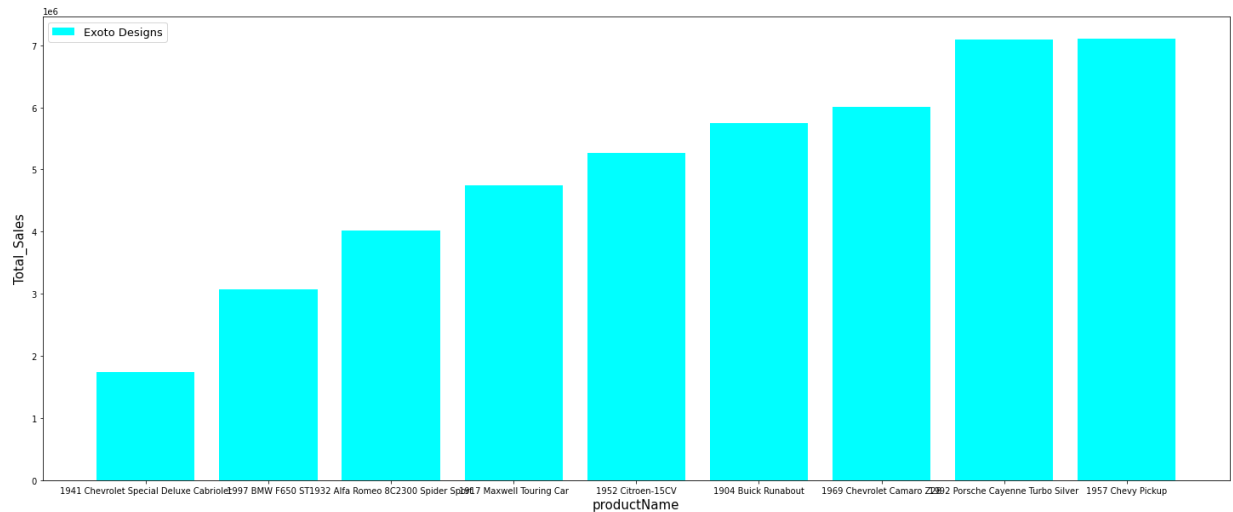


```
In [20]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[2]'], df.Total[df['productVendor']=='labels[2]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```

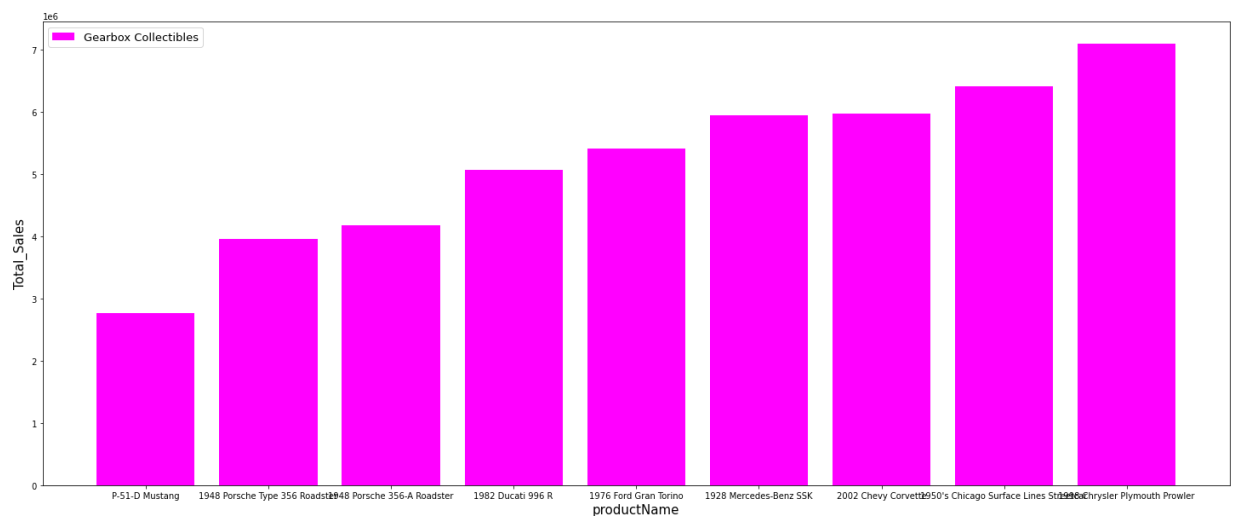




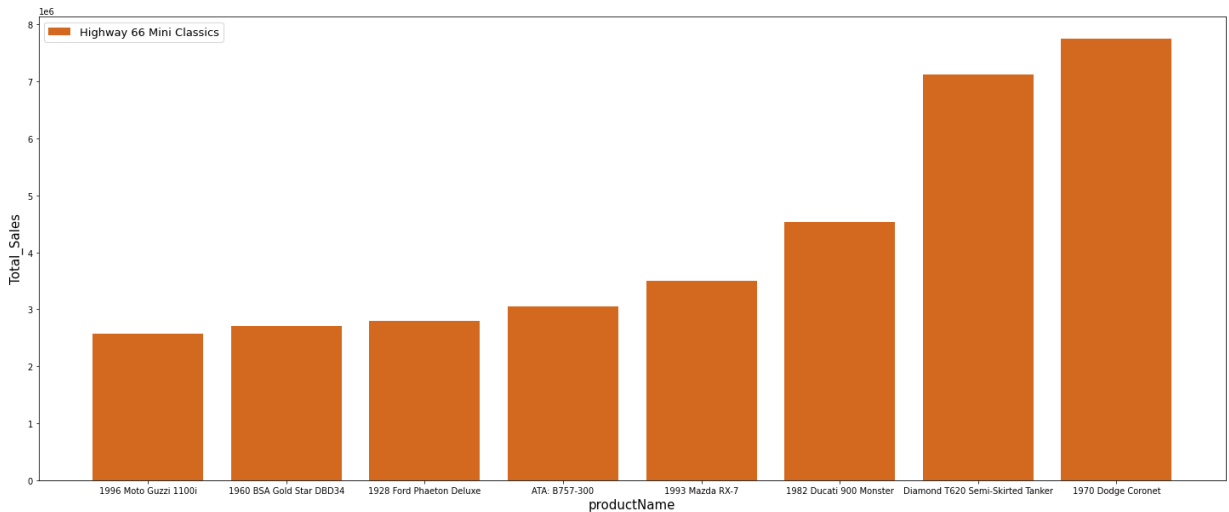
```
In [21]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='Exoto Designs'], df.Total[df['productVendor']=='Exoto Designs'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



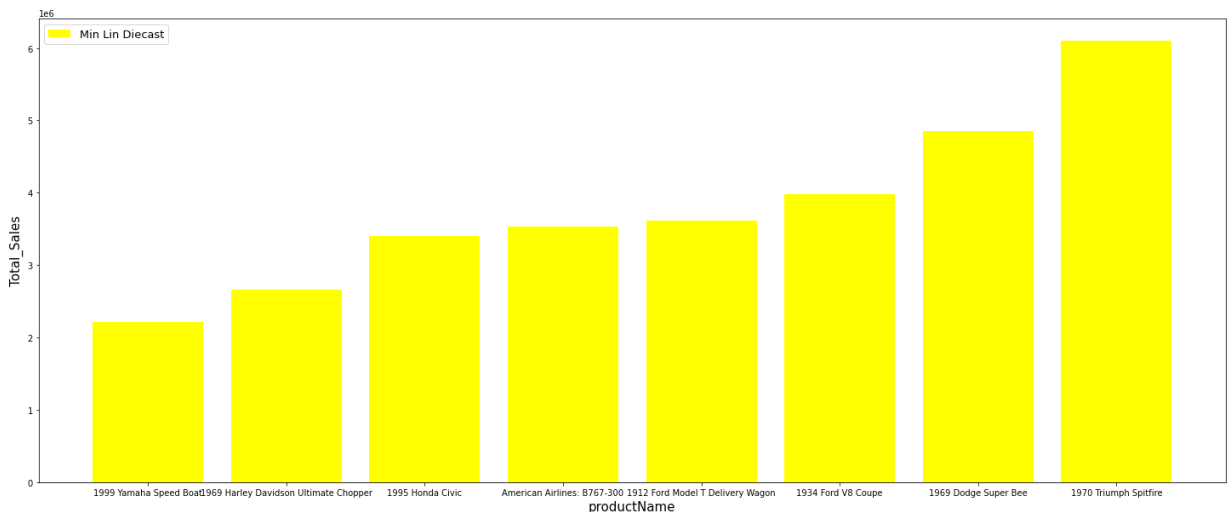
```
In [22]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='Gearbox Collectibles'], df.Total[df['productVendor']=='Gearbox Collectibles'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



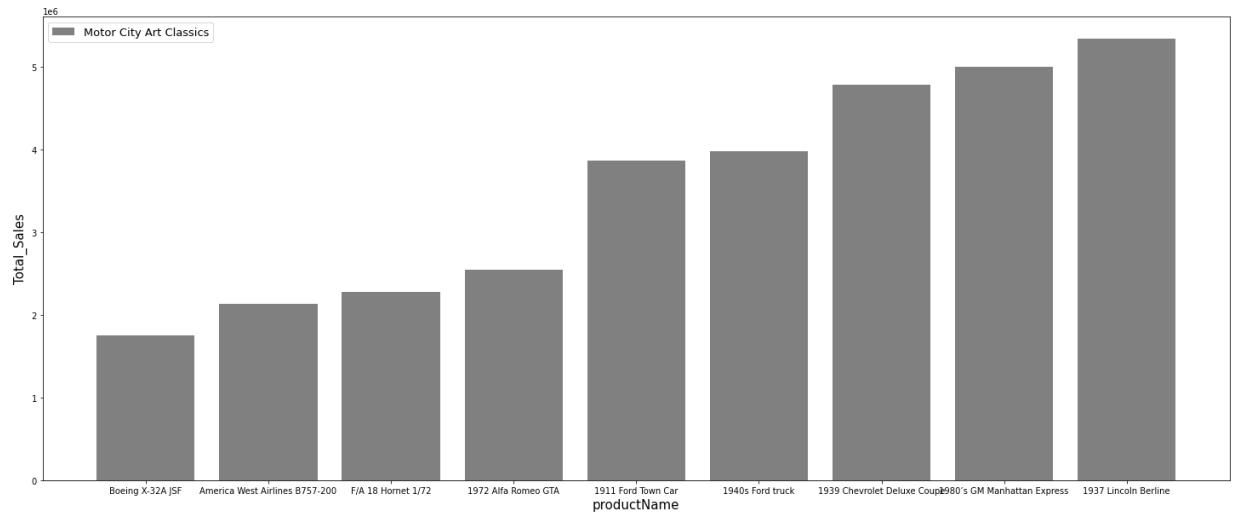
```
In [23]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[5]'], df.Total[df['productVendor']=='labels[5]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



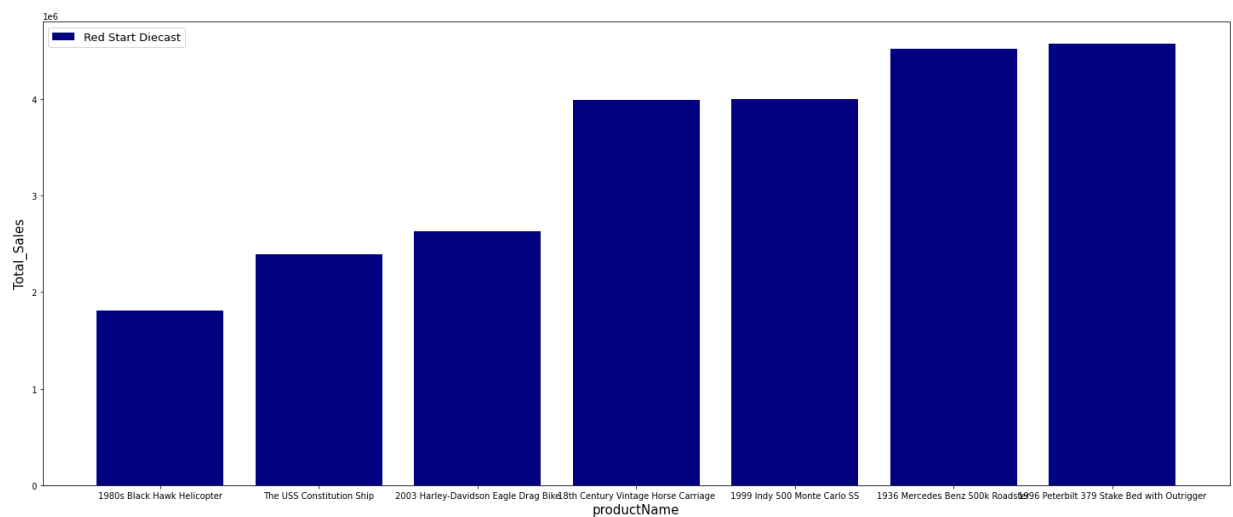
```
In [24]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[6]'], df.Total[df['productVendor']=='labels[6]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



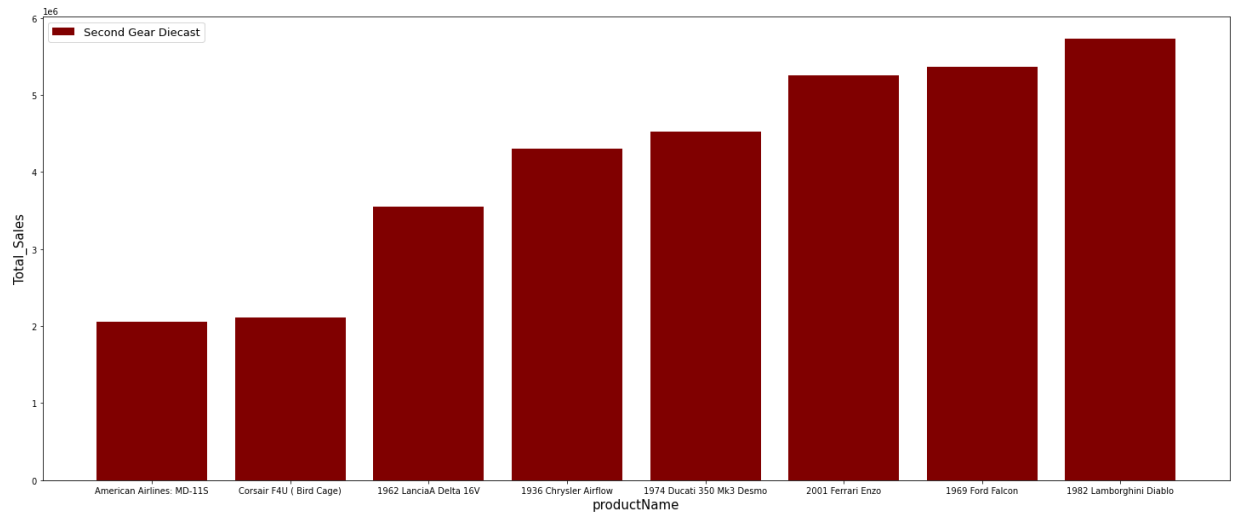
```
In [25]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[7]'], df.Total[df['productVendor']=='labels[7]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



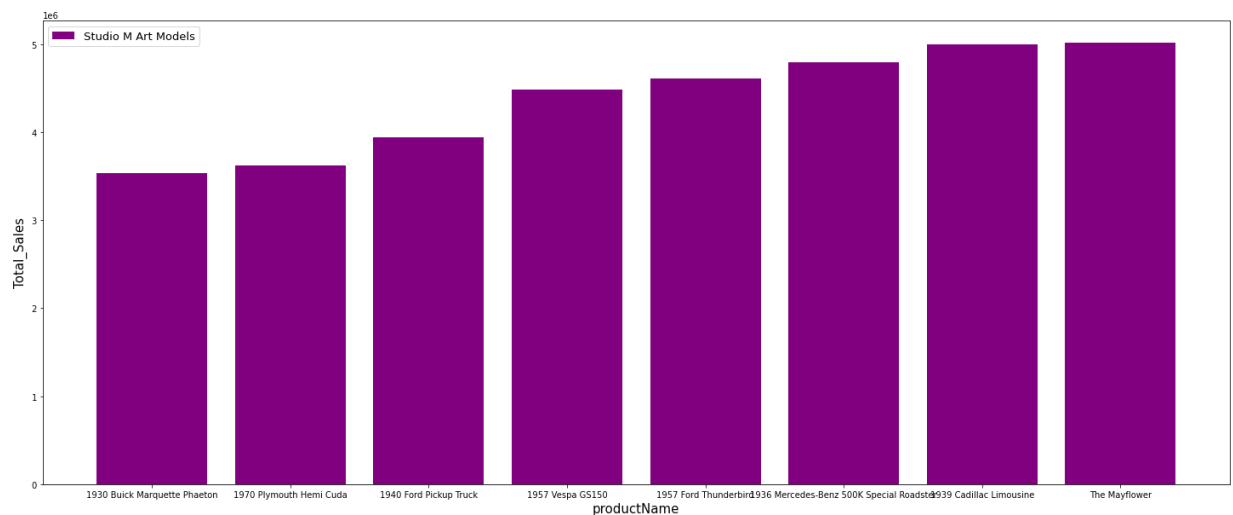
```
In [26]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[8]'], df.Total[df['productVendor']=='labels[8]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



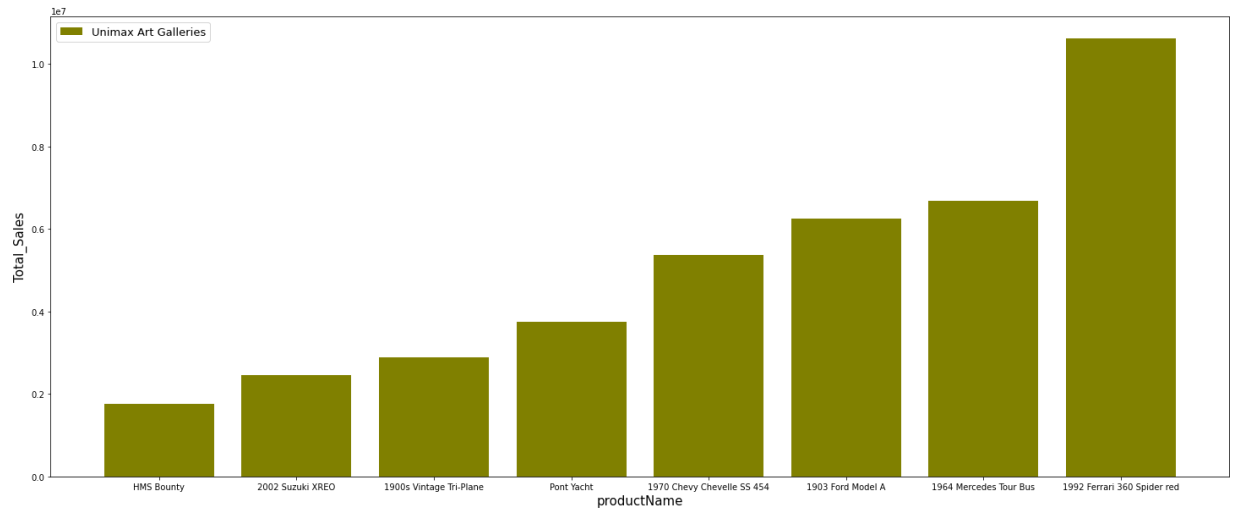
```
In [27]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[9]'], df.Total[df['productVendor']=='labels[9]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



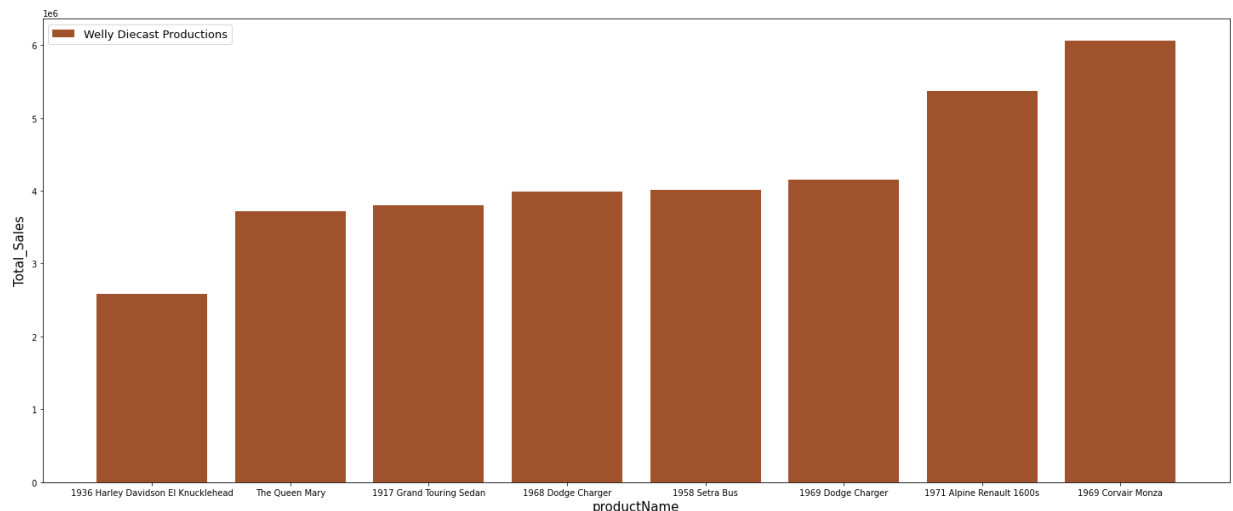
```
In [28]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[10]'], df.Total[df['productVendor']=='labels[10]'])
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



```
In [29]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[11]'], df.Total[df['productVendor']=='
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



```
In [30]: plt.figure(figsize=(25,10))
plt.bar(df.index[df['productVendor']=='labels[12]'], df.Total[df['productVendor']=='
plt.ylabel('Total_Sales', size=15)
plt.xlabel('productName', size=15)
plt.legend(fontsize=13)
plt.show()
```



```
In [31]: df.groupby('productVendor',as_index=False)['Total'].min()
```

Out[31]:

	productVendor	Total
0	Autoart Studio Design	1849456.67
1	Carousel DieCast Legends	3410857.15
2	Classic Metal Creations	2763814.78
3	Exoto Designs	1735078.51
4	Gearbox Collectibles	2758435.22
5	Highway 66 Mini Classics	2577976.24
6	Min Lin Diecast	2210532.38
7	Motor City Art Classics	1749369.58
8	Red Start Diecast	1806788.04
9	Second Gear Diecast	2053709.62
10	Studio M Art Models	3532461.58
11	Unimax Art Galleries	1769259.33
12	Welly Diecast Productions	2579014.80

```
In [32]: df.groupby('productVendor',as_index=False)['Total'].max()
```

Out[32]:

	productVendor	Total
0	Autoart Studio Design	6357436.57
1	Carousel DieCast Legends	8312033.60
2	Classic Metal Creations	7735217.35
3	Exoto Designs	7104392.05
4	Gearbox Collectibles	7091407.14
5	Highway 66 Mini Classics	7750886.69
6	Min Lin Diecast	6099675.92
7	Motor City Art Classics	5341201.32
8	Red Start Diecast	4574258.55
9	Second Gear Diecast	5728362.72
10	Studio M Art Models	5016145.06
11	Unimax Art Galleries	10620947.53
12	Welly Diecast Productions	6058769.33

In [ ]: