KSR DATAVIZON

Key to Sustainable Results

PYTHON MINI PROJECT

In [1]:
```
import pandas as pd
df1 = pd.read_csv("D:\\Data Engineering\\Python\\Pyhton_assignment_2\\TATA_TB1.csv",encoding='latin1')
df2 = pd.read_csv("D:\\Data Engineering\\Python\\Pyhton_assignment_2\\TATA_TB2.csv",encoding='latin1')
```

In [43]:
```
from IPython.display import display
```

In [44]:
```
#1) Write a query to calculate the total records in these two tables?
```

In [45]:
```
Totalrecords1 = df1.shape[0]
print("Total records present in TATA_TB1 is :",Totalrecords1)
Totalrecords2 = df2.shape[0]
print("Total records present in TATA_TB2 is :",Totalrecords2)
```

```
Total records present in TATA_TB1 is : 4117
Total records present in TATA_TB2 is : 8047
```

In [46]:
```
#2) Write a query to calculate the total unique count of customers?
```

In [47]:
```
totalcust = df1['CustomerName'].nunique()
print("The total count of customers is :",totalcust)
```

```
The total count of customers is : 792
```

In [48]:
```
##3) Write a query to fetch the latest order date and oldest order date?
```

In [49]:
```
latest_order_date = df1['OrderDate'].max()
oldest_order_date = df1['OrderDate'].min()
print("The latest order date is: ",latest_order_date)
print("The oldest order date is: ",oldest_order_date)
```

```
The latest order date is:  2014-12-31
The oldest order date is:  2011-01-01
```

In [50]:
```
#4) Write query to get unique years?
```

In [51]:
```
df1['OrderDate'] = pd.to_datetime(df1['OrderDate'],format='%Y-%m-%d') #converting the date to date dat
```

In [52]:
```
uniqueyears = df1['OrderDate'].apply(lambda x: x.year).unique()
print("The distinct years :",uniqueyears)
```

```
The distinct years : [2011 2012 2013 2014]
```

In [53]:
```
df1['year'] = df1['OrderDate'].dt.year # this will be used for extracting the year from the date as ne
```

In [54]:
```
#5) Write a query to get the no. of regions and display the region names?
```

In [55]:
```
regioncount = df1['Region'].nunique()
regionnames = df1['Region'].unique()
print("The count of regions is :",regioncount,"and regions are :",list(regionnames))
```

```
The count of regions is : 3 and regions are : ['North', 'Central', 'South']
```

```
In [56]: #6) Write a query to get the no. of countries and display the country names?
```

```
In [57]: countrycount = df1['Country'].nunique()
         countrynames = df1['Country'].unique()
         print("The count of regions is :",countrycount, "and countries are",countrynames)
```

```
The count of regions is : 15 and countries are ['Sweden' 'United Kingdom' 'France' 'Italy' 'Austria'
'Spain' 'Germany'
 'Netherlands' 'Denmark' 'Belgium' 'Norway' 'Portugal' 'Switzerland'
 'Ireland' 'Finland']
```

```
In [58]: #7) Write a query to get the no. of states and display the state names?
```

```
In [18]: statecount = df1['State'].nunique()
         statenames = df1['State'].unique()
         print("The count of states are :",statecount,"and the state names is :",statenames)
```

```
The count of states are : 127 and the state names is : ['Stockholm' 'England' 'Auvergne-Rhône-Alpes'
"Provence-Alpes-Côte d'Azur"
 'Languedoc-Roussillon-Midi-Pyrénées' 'Liguria' 'Vienna' 'Murcia'
 'Lower Saxony' 'South Holland' 'Västra Götaland' 'Hovedstaden'
 'Valenciana' 'South Denmark' 'Lombardy' 'Sicily' 'Ile-de-France'
 'North Rhine-Westphalia' 'Flemish Brabant' 'Tuscany' 'Emilia-Romagna'
 'Madrid' 'Oslo' 'Lisboa' 'Saxony' 'Andalusía' 'Catalonia'
 'Alsace-Champagne-Ardenne-Lorraine' 'Bavaria' 'Uppsala'
 'Nord-Pas-de-Calais-Picardie' 'Hesse' 'Overijssel' 'Basel-Stadt'
 'Bourgogne-Franche-Comté' 'Zürich' 'Dublin' 'Lazio' 'Namur'
 'North Holland' 'Berlin' 'Baden-Württemberg'
 'Aquitaine-Limousin-Poitou-Charentes' 'Uusimaa' 'Apulia' 'Saxony-Anhalt'
 'Rogaland' 'Sardinia' 'Drenthe' 'Mecklenburg-Vorpommern' 'North Brabant'
 'Umbria' 'Geneva' 'Veneto' 'Normandy' 'Scotland' 'Coimbra'
 'Castile and León' 'Gelderland' 'Hamburg' 'Brandenburg'
 'Pays de la Loire' 'Antwerp' 'Bremen' 'Thuringia' 'Porto' 'Utrecht'
 'Castile-La Mancha' 'Brittany' 'Campania' 'Cork' 'Groningen'
 'East Flanders' 'Ceuta' 'Halland' 'Navarra' 'Rhineland-Palatinate'
 'Limburg' 'Upper Austria' 'Schleswig-Holstein' 'Tyrol' 'Corsica'
 'Trentino-Alto Adige' 'Vaud' 'Piedmont' 'Calabria' 'Galicia' 'Buskerud'
 'Centre-Val de Loire' 'Styria' 'Abruzzi' 'Basque Country' 'Cantabria'
 'Asturias' 'Finland Proper' 'Central Jutland' 'Wales' 'Kymenlaakso'
 'Friesland' 'Braga' 'Aveiro' 'Marche' 'Saarland' 'Skåne'
 'Friuli-Venezia Giulia' 'Balearic Islands' 'Extremadura' 'Basilicata'
 'Hedmark' 'Hainaut' 'Melilla' 'Salzburg' 'Carinthia' 'Zeeland'
 'Hordaland' 'Lucerne' 'Liège' 'West Flanders' 'Bern' 'Brussels'
 'Södermanland' 'Zealand' 'Galway' 'Värmland' 'Vest-Agder' 'St. Gallen'
 'Setúbal']
```

```
In [19]: #8) Write a query to get the no. of cities and display the city names?
```

In [20]:
```python
citycount = df1['City'].nunique()
citynames = df1['City'].unique()
print("The count of cities are :",citycount,"and the city names are:",citynames)
```

```
The count of cities are : 999 and the city names are: ['Stockholm' 'Southport' 'Valence' 'Birming
ham' 'Echirolles'
 'La Seyne-sur-Mer' 'Toulouse' 'Genoa' 'Vienna' 'Murcia' 'Woking' 'Lohne'
 'Leicester' 'Sheffield' 'Dordrecht' 'Gothenburg' 'Langen' 'Copenhagen'
 'Gandia' 'Esbjerg' 'Sesto San Giovanni' 'Trapani' 'Villiers-sur-Marne'
 'Bielefeld' 'Leuven' 'Prato' 'Gela' 'Bologna' 'Menden' 'Maisons-Alfort'
 'Madrid' 'Oslo' 'Lisbon' 'Draguignan' 'Halle' 'Parma' 'Dresden' 'Seville'
 'Torrevieja' 'Barcelona' 'London' 'Reims' 'Rosenheim' 'Uppsala' 'Nice'
 'Boulogne-sur-Mer' 'La Crau' 'Siena' 'Frankfurt' 'Almelo' 'Basel'
 'Coslada' 'Marseille' 'Hanover' 'Elda' 'Hardenberg' 'Muret' 'Beaune'
 'Paris' 'Castrop-Rauxel' 'Milan' 'Zurich' 'Grosseto' 'Dublin' 'Rome'
 'Namur' 'Zaanstad' 'Bochum' 'Colmar' 'Farnborough' 'Berlin' 'Rimini'
 'Baden-Baden' 'Pforzheim' 'Coventry' 'Pessac' 'Helsinki' 'Bonn' 'Leipzig'
 'Tourcoing' 'Bari' 'Magdeburg' 'Noisy-le-Sec' 'Stavanger' 'Cagliari'
 'Marsala' 'Emmen' 'Augsburg' 'Stralsund' 'Carcassonne' 'Munich' 'Wigan'
 'Helmond' 'Castres' 'Foligno' 'Hamm' 'Troisdorf' 'Geneva'
 'Newcastle upon Tyne' 'Treviso' 'Le Havre' 'Edinburgh' 'Saint-Priest'
 'Lattes' 'Le Blanc-Mesnil' 'Essen' 'Coimbra' 'Ponferrada' 'Nacka' 'Crewe'
 'Duisburg' 'Montigny-le-Bretonneux' 'Apeldoorn' 'Brindisi' 'Hamburg'
```

In [21]:
```python
#09) Write a query to calculate the total count of products?
```

In [22]:
```python
prouctcount = df2['ProductName'].nunique()
print("The total count of products are :",prouctcount)
```

```
The total count of products are : 1810
```

In [23]:
```python
#10) Write a query to calculate total sales, total profit and total order quantity?
```

In [24]:
```python
totalsales = df2['Sales'].sum()
totalprofit = df2['Profit'].sum()
totalqty = df2['OrderQuantity'].sum()
print("The total sales is :",totalsales)
print("The total profit is :",totalprofit)
print("The totdal Orderqty is ",totalqty)
```

```
The total sales is : 2348482
The total profit is : 283240
The totdal Orderqty is  30354
```

In [25]:
```python
#11) Write a query to calculate the total sales amount,totla order quantity for each category.
#Display the category, total sales, and total order qty and order by total sales from highest to lowes
```

In [5]:
```python
df2.groupby(['Category'])[["Sales","OrderQuantity"]].sum()\
   .sort_values("Sales",ascending=False).style.background_gradient(cmap='Greens')
```

Out[5]:

Category	Sales	OrderQuantity
Technology	886015	5811
Office Supplies	823658	19902
Furniture	638809	4641

In [27]:
```python
#12) Write a query to calculate the total profit amount for each category.
      #Display the category, total profit, and total order qty and order by total profit from highest
```

In [12]:
```
df2.groupby(['Category'])[["Profit","OrderQuantity"]].sum()\
   .sort_values("Profit",ascending = False).style.background_gradient(cmap='BrBG')
```

Out[12]:

Category	Profit	OrderQuantity
Office Supplies	124952	19902
Technology	108554	5811
Furniture	49734	4641

In [29]:
```
#13) Write a query to fetch the subcategories where total sales are greater than 100000?
```

In [13]:
```
df2.groupby(['SubCategory'])[["Sales"]].sum().query('Sales > 100000')\
   .sort_values("Sales",ascending = False).style.background_gradient(cmap='GnBu')
```

Out[13]:

SubCategory	Sales
Bookcases	294396
Copiers	290081
Phones	282559
Storage	272489
Appliances	209900
Chairs	186698
Machines	182066
Accessories	131309
Art	127184

In [31]:
```
#14) Write a query to fetch the products where total profit is greater than 2500 and sort it based on
```

In [15]:
```
df2.groupby(["ProductName"])[["Profit"]].sum().query('Profit > 2500')\
   .sort_values("Profit",ascending = False).style.background_gradient(cmap='rainbow_r')
```

Out[15]:

ProductName	Profit
Nokia Smart Phone, Full Size	7583
Hoover Stove, Red	6139
Hamilton Beach Stove, Silver	5778
SAFCO Executive Leather Armchair, Black	4324
Safco Classic Bookcase, Metal	4183
Cisco Smart Phone, with Caller ID	4055
Brother Fax Machine, Laser	3918
Eldon Lockers, Industrial	3611
Cisco Smart Phone, Cordless	3388
Hamilton Beach Stove, Red	2738
Belkin Router, USB	2677
Nokia Smart Phone, Cordless	2633
Cuisinart Refrigerator, Black	2627
Eldon File Cart, Single Width	2539

```
In [33]: #15) Write a query to get the total sales and total profit for Office Supplies category?
```

```
In [34]: df2[df2["Category"] == "Office Supplies"][["Sales","Profit"]].sum()

Out[34]: Sales     823658
         Profit    124952
         dtype: int64
```

```
In [35]: #16) Write a query to get the total sales and total profit for Furniture category and
              #Tables, Bookcases sub-categories?
```

```
In [36]: df2[(df2["Category"] == "Furniture") & (df2["SubCategory"].isin(["Tables","Bookcases"]))][["Sales","Pr
         #noraml method

Out[36]: Sales     383874
         Profit     22924
         dtype: int64
```

```
In [37]: df2.query('Category == "Furniture" and SubCategory == ["Tables","Bookcases"]')[["Sales","Profit"]].sum
         #query method

Out[37]: Sales     383874
         Profit     22924
         dtype: int64
```

```
In [38]: #17)Write a query to get the total sales and total profit for Technology category
              #and the Accessories, Copiers, Phones sub-categories ?
```

```
In [66]: df2.loc[(df2["Category"] == "Technology") & (df2["SubCategory"].isin(["Accessories","Copiers","Phones"
         [["Sales","Profit"]].sum() #loc method

Out[66]: Sales     703949
         Profit     97236
         dtype: int64
```

```
In [40]: df2.query('Category == "Technology" and SubCategory == ["Accessories","Copiers","Phones"]')[["Sales","
         #query method

Out[40]: Sales     703949
         Profit     97236
         dtype: int64
```

```
In [41]: #18) Write a query to get total sales and total profit by Region, Segment and sort the sales from high
```

In [16]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["Region","Segment"])[["Sales","Profit"]].sum()\
  .sort_values('Sales',ascending=False).style.background_gradient(cmap='rainbow_r')
```

Out[16]:
```
                           Sales   Profit
Region    Segment
           Consumer       701892    82146
Central
           Corporate      396437    52115
 North     Consumer       267955    35686
 South     Consumer       266435    29615
Central  Home Office      216924    23375
 South     Corporate      177709    13647
           Corporate      163991    26872
 North
         Home Office       84033     8881
 South   Home Office       73106    10903
```

In [43]:
```
#19) Write a query to get total sales and total profit by Country, State and city and sort the sales f
```

In [18]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["Country","State","City"])[["Sales","Profit"]].sum
  .sort_values("Sales",ascending = False).style.background_gradient(cmap='rainbow_r')
```

Out[18]:
```
                                                              Sales   Profit
       Country                       State           City
United Kingdom                     England         London     69230    13931
       Germany                      Berlin         Berlin     52555     5942
       Austria                      Vienna         Vienna     51844    13207
         Spain                      Madrid         Madrid     44981    11129
        France                 Ile-de-France        Paris     42245     6680
         Italy                       Lazio           Rome     28330      191
         Spain                   Catalonia      Barcelona     27405     2246
       Germany                     Hamburg        Hamburg     23574     5858
        France  Provence-Alpes-Côte d'Azur      Marseille     21677     2889
                                  Piedmont          Turin     19829     1937
         Italy
```

In [45]:
```
#20)Write a query to get total sales and total orderqty by CustomerName and sort the sales from highes
```

```
In [20]: pd.merge(df1,df2,on="OrderID",how="inner").groupby(["CustomerName"])[["Sales","OrderQuantity"]].sum()\
           .sort_values("Sales",ascending = False).style.background_gradient(cmap='magma')
```

Out[20]:

CustomerName	Sales	OrderQuantity
Angie Massengill	16146	102
Lola Hughes	13191	127
Ashton Charles	13056	63
Isaac David	11271	110
Philip Newsom	10893	80
Joel Peters	10477	137
Bettie Lang	10466	108
Audrey Knowles	10363	96
Lilly Le Grand	9962	88
Elijah Sodeman	9689	101

```
In [47]: #21) Identify the top 5 products with the highest sales (by sales amount).
              #Show the product name, total sales, and total qty?
```

```
In [24]: pd.merge(df1,df2,on="OrderID",how="inner").groupby(["ProductName"])[["Sales","OrderQuantity"]].sum()\
           .sort_values("Sales",ascending = False).iloc[:5].style.background_gradient(cmap='rainbow_r')
```

Out[24]:

ProductName	Sales	OrderQuantity
Nokia Smart Phone, Full Size	30645	54
Hamilton Beach Stove, Silver	16890	33
Cisco Smart Phone, Cordless	14723	29
Novimex Executive Leather Armchair, Red	13898	44
Cisco Smart Phone, with Caller ID	13215	25

```
In [49]: #22) Write a query to get total sales by City having sales greater than 35000?
```

```
In [25]: pd.merge(df1,df2,on="OrderID",how="inner").groupby(["City"])[["Sales"]].sum().sort_values('Sales',asce
           .query('Sales > 25000').style.background_gradient(cmap='rainbow_r')
```

Out[25]:

City	Sales
London	69230
Berlin	52555
Vienna	51844
Madrid	44981
Paris	42245
Rome	28330
Barcelona	27405

```
In [ ]: #23) Write a query to get total sales by CustomerName having sales greater than 10000?
```

In [26]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["CustomerName"])[["Sales"]].sum()\
  .query('Sales > 10000').sort_values("Sales",ascending = False).style.background_gradient(cmap='rainb
```

Out[26]:

	Sales
CustomerName	
Angie Massengill	16146
Lola Hughes	13191
Ashton Charles	13056
Isaac David	11271
Philip Newsom	10893
Joel Peters	10477
Bettie Lang	10466
Audrey Knowles	10363

In [ ]:
```
#24)Write a query to get total sales and total profit by shipmode and sort the sales in ascending orde
```

In [27]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["ShipMode"])[["Sales","Profit"]].sum()\
  .sort_values(["Sales","Profit"],ascending = [False,True]).style.background_gradient(cmap='rainbow_r'
```

Out[27]:

	Sales	Profit
ShipMode		
Economy	1412777	178696
Economy Plus	483965	54336
Priority	320426	32639
Immediate	131314	17569

In [ ]:
```
#25) Write a query to get total sales for North and central region?
```

In [70]:
```
pd.merge(df1,df2,on="OrderID",how="inner").query('Region == ["North","Central"]')[["Sales"]].sum()
```

Out[70]:
```
Sales    1831232
dtype: int64
```

In [ ]:
```
#26) Write a query to get total sales and total profit for Italy and Spain countries?
```

In [71]:
```
pd.merge(df1,df2,on="OrderID",how="inner").query('Country == ["Italy","Spain"]')[["Sales","Profit"]].s
```

Out[71]:
```
Sales     502144
Profit     62869
dtype: int64
```

In [ ]:
```
#27) Write a query to get the total sales and  total profit for each year sort the sales from highest
```

In [87]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(['year'])[["Sales"]].sum().sort_values("Sales",asce
```

Out[87]:

year	Sales
2014	755030
2013	630224
2012	548880
2011	414348

In [ ]:
```
#28) Find the top 10 customers who spent the most across all transactions.
     #Display the customer name, total sales, and number of orders placed?
```

In [28]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["CustomerName"])[["Sales","OrderQuantity"]].sum()\
  .sort_values("Sales",ascending=False).iloc[:10].style.background_gradient(cmap='rainbow_r')
```

Out[28]:

CustomerName	Sales	OrderQuantity
Angie Massengill	16146	102
Lola Hughes	13191	127
Ashton Charles	13056	63
Isaac David	11271	110
Philip Newsom	10893	80
Joel Peters	10477	137
Bettie Lang	10466	108
Audrey Knowles	10363	96
Lilly Le Grand	9962	88
Elijah Sodeman	9689	101

In [ ]:
```
#29) Write a query to find which products are most preferred by customers based on the total sales.
     #Display customer name, favorite product(top 3 products per each customer) and total sales on tha
```

In [30]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["CustomerName","ProductName"])["Sales"].sum().rese
  .sort_values("Sales",ascending=False).assign(Rank = lambda x:x.groupby("CustomerName")["Sales"]\
  .rank(method="first",ascending=False)).query("Rank <= 3").sort_values(["CustomerName","Rank"])\
  .style.background_gradient(cmap='rainbow_r')
```

Out[30]:

	CustomerName	ProductName	Sales	Rank
3	Aaron Bootman	Brother Fax and Copier, Color	1543	1.000000
9	Aaron Bootman	HP Personal Copier, Digital	632	2.000000
12	Aaron Bootman	Konica Receipt Printer, Durable	245	3.000000
28	Aaron Cunningham	Eldon File Cart, Single Width	809	1.000000
30	Aaron Cunningham	Hon Rocking Chair, Red	392	2.000000
35	Aaron Cunningham	Tenex File Cart, Industrial	322	3.000000
37	Aaron Davey	Advantus Frame, Durable	973	1.000000
41	Aaron Davey	Smead File Cart, Industrial	321	2.000000
42	Aaron Davey	Stanley Canvas, Fluorescent	304	3.000000
44	Aaron Macrossan	Sanford Pencil Sharpener, Easy-Erase	81	1.000000
47	Abbie Perry	Harbour Creations Steel Folding Chair, Black	344	1.000000

In [ ]:
```
#30) Write a query to get 7th rank customer name based on total sales?
     #Display customer name, sales amount and rank.
```

In [38]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["CustomerName"])["Sales"].sum().reset_index()\
  .sort_values("Sales",ascending=False).assign(Rank = lambda x:x["Sales"].rank(method="first",ascendin
  .query("Rank == 7")
```

Out[38]:

	CustomerName	Sales	Rank
90	Bettie Lang	10466	7.0

In [ ]:
```
#31)Write query to get total sales and total profit in years 2011 and 2013 Display year wise total sal
```

In [61]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby("year")[["Sales"]].sum().query('year ==[2011,2013]'
```

Out[61]:

	Sales
year	
2011	414348
2013	630224

In [ ]:
```
#32)Write a query to get total sales, total profit and total order qty
     #by country, state, category and sub-category and sort it from highest to lowest based on sales c
```

In [31]:
```
pd.merge(df1,df2,on="OrderID",how="inner").groupby(["Country","State","Category","SubCategory"])\
  [["Sales","Profit","OrderQuantity"]].sum().sort_values("Sales",ascending=False).style.background_gra
```

Out[31]:

				Sales	Profit	OrderQuantity
Country	**State**	**Category**	**SubCategory**			
United Kingdom	**England**	**Furniture**	**Bookcases**	52576	12790	237
		Technology	**Phones**	52262	11124	198
			Copiers	49025	12408	214
		Office Supplies	**Storage**	42752	5987	569
		Technology	**Machines**	33920	7653	187
		Office Supplies	**Appliances**	31541	7589	105
France	**Ile-de-France**	**Technology**	**Copiers**	30879	1738	150
			Machines	26162	1372	154
			Phones	25955	3190	151
Germany	**North Rhine-Westphalia**	**Office Supplies**	**Storage**	24010	4551	275

In [ ]:
```
#33) write a function to get the region sales dynamically, if we pass any region, that region sales so
```

In [78]:
```
def regionsales(rgn):
    regin_sales = pd.merge(df1,df2,on="OrderID",how="inner").groupby(["Region"])[["Sales"]].sum()
    if rgn in regin_sales.index:
        return regin_sales.loc[rgn]
    else:
        return "Region not exists"
```

In [79]:
```
regionsales("South")
```

Out[79]:
```
Sales    517250
Name: South, dtype: int64
```

In [81]:
```
regionsales("North")
```

```
Sales    515979
Name: North, dtype: int64
```

In [82]:
```
regionsales("East")
```

Out[82]:
```
'Region not exists'
```

In [ ]:
```
#34) write a function to get the country sales,profit dynamically, if we pass any country, that countr
```

In [83]:
```
def country_sales_profit(cntry):
    ctry_s_p = pd.merge(df1,df2,on="OrderID",how="inner").groupby(["Country"])[["Sales","Profit"]].sum
    if cntry in ctry_s_p.index:
        return ctry_s_p.loc[cntry]
    else:
        return "Country doesn't exists"
```

In [84]:
```
country_sales_profit("Sweden")
```

Out[84]:
```
Sales     30490
Profit   -17524
Name: Sweden, dtype: int64
```

In [88]:
```
country_sales_profit("Ireland")
```

Out[88]:
```
Sales     15998
Profit    -6886
Name: Ireland, dtype: int64
```

In [89]:
```
country_sales_profit("Denmark")
```

Out[89]:
```
Sales     7763
Profit   -3608
Name: Denmark, dtype: int64
```

In [ ]:
```
#35) write a function to get the Category orderquantity dynamically, if we pass any Category, that Cat
```

In [95]:
```
def categoryordrqty(ctgryoqty):
    cate_order_qty = pd.merge(df1,df2,on="OrderID",how="inner").groupby(["Category"])[["OrderQuantity"
    if ctgryoqty in cate_order_qty.index:
        return cate_order_qty.loc[ctgryoqty]
    else:
        return "Category doesn't exists"
```

In [96]:
```
categoryordrqty("Office Supplies")
```

Out[96]:
```
OrderQuantity    19902
Name: Office Supplies, dtype: int64
```

In [97]:
```
categoryordrqty("papers")
```

Out[97]:
```
"Category doesn't exists"
```

In [99]:
```
categoryordrqty("Furniture")
```

Out[99]:
```
OrderQuantity    4641
Name: Furniture, dtype: int64
```

```
In [ ]: #36) write a function to get the City sales and profit dynamically, if we pass any city, that City sal
```

```
In [107]: def citysalesprofit(cti):
              city_sales_profit = pd.merge(df1,df2,on="OrderID",how="inner").groupby(["City"])[["Sales","Profit"
              if cti in city_sales_profit.index:
                  return city_sales_profit.loc[cti]
              else:
                  return "City deosn't exists"
```

```
In [108]: citysalesprofit("Toulouse")
```

```
Out[108]: Sales      11565
          Profit     -9675
          Name: Toulouse, dtype: int64
```

```
In [110]: citysalesprofit("Waterlooville")
```

```
Out[110]: Sales      623
          Profit     191
          Name: Waterlooville, dtype: int64
```