DATA MERGING AND FILTERING PROJECT ON PYTHON

Tasks

- Data Merging and Basic Filtering
- Aggregation and Multiple Grouping
- Slicing and Advanced Filtering with .iloc and .loc
- Filtering using startswith, endswith, and contains
- Conditional Column Creation with np.where

Audience

Technical audience only

Data Merging and Basic Filtering

In [1]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving w3MartDB.tar to w3MartDB.tar
```

- Imported the data (a zipped folder)

In [2]:
```
tar_file_name = "w3MartDB.tar" #To store the tar file in the variable "tar_file_
```

In [3]:
```
import tarfile  #To import the moudle that handles tarfiles

#Then, I extracted the TAR file
with tarfile.open(tar_file_name) as tar: #Used the "with" statement to ensure th
  tar.extractall()
```

In [4]:
```
import os #To import os and viewed extracted files
os.listdir()
```

Out[4]:
```
['.config', 'w3MartDB.tar', 'w3MartDB', 'sample_data']
```

In [5]:
```
for f in os.listdir("w3MartDB"):  #Used "for loop" to view all csv files in targ
  print (f)
```

```
suppliers.csv
orders.csv
orderdetails.csv
shippers.csv
categories.csv
customers.csv
employees.csv
products.csv
```

```
In [6]: import pandas as pd
        import numpy as np  #To import Pandas and Numpy for data manipulation and analys
```

```
In [7]: extracted_file = os.listdir("w3MartDB")
        print ("extracted_file:", extracted_file)  #To store extracted files in the vari
```

```
extracted_file: ['suppliers.csv', 'orders.csv', 'orderdetails.csv', 'shippers.cs
v', 'categories.csv', 'customers.csv', 'employees.csv', 'products.csv']
```

```
In [8]: #To initialize an empty dictionary to store Dataframes
        dataframe = {}
```

```
In [9]: #To read each csv file into a Pandas Dataframe and stored in the dictionary
        for file in extracted_file:
          df_name = file.split(".")[0] #To use the filename without the extension (.csv)
          dataframe[df_name] = pd.read_csv(os.path.join("w3MartDB",file)) #read the CSV
```

```
In [10]: dataframe.keys() #To return the column labels of the dataframe
```

```
Out[10]: dict_keys(['suppliers', 'orders', 'orderdetails', 'shippers', 'categories', 'cu
stomers', 'employees', 'products'])
```

```
In [11]: dataframe["orders"] #To return the table in the file "orders"
```

Out[11]:

	orderid	customerid	employeeid	orderdate	shipperid
0	10248	90	5	1996-07-04	3
1	10249	81	6	1996-07-05	1
2	10250	34	4	1996-07-08	2
3	10251	84	3	1996-07-08	1
4	10252	76	4	1996-07-09	2
...	...	...	...	...	...
191	10439	51	6	1997-02-07	3
192	10440	71	4	1997-02-10	2
193	10441	55	3	1997-02-10	2
194	10442	20	3	1997-02-11	2
195	10443	66	8	1997-02-12	1

196 rows × 5 columns

```
In [12]: for name,df in dataframe.items():
           globals()[name] = df  #To access columns in dataframe as global variables
```

- **Merge the orders and orderdetails tables on the OrderID field to create a single DataFrame with detailed information about each order.**

```
In [13]: orderdetails #To view the table in orderdetails
```

Out[13]:

	orderdetailid	orderid	productid	quantity
0	1	10248	11	12
1	2	10248	42	10
2	3	10248	72	5
3	4	10249	14	9
4	5	10249	51	40
...	...	...	...	...
513	514	10442	11	30
514	515	10442	54	80
515	516	10442	66	60
516	517	10443	11	6
517	518	10443	28	12

518 rows × 4 columns

In [14]:
```
orders #To view the table in orders
```

Out[14]:

	orderid	customerid	employeeid	orderdate	shipperid
0	10248	90	5	1996-07-04	3
1	10249	81	6	1996-07-05	1
2	10250	34	4	1996-07-08	2
3	10251	84	3	1996-07-08	1
4	10252	76	4	1996-07-09	2
...	...	...	...	...	...
191	10439	51	6	1997-02-07	3
192	10440	71	4	1997-02-10	2
193	10441	55	3	1997-02-10	2
194	10442	20	3	1997-02-11	2
195	10443	66	8	1997-02-12	1

196 rows × 5 columns

In [15]:
```
#Joined "orderdetails" to "orders" table using inner join, stored the result in
Mart_Merge = pd.merge(orderdetails,orders[["orderid","customerid","orderdate"]],
```

In [ ]:
```
Mart_Merge #To view merged table
```

Out[ ]:

	orderdetailid	orderid	productid	quantity	customerid	orderdate
0	1	10248	11	12	90	1996-07-04
1	2	10248	42	10	90	1996-07-04
2	3	10248	72	5	90	1996-07-04
3	4	10249	14	9	81	1996-07-05
4	5	10249	51	40	81	1996-07-05
...	...	...	...	...	...	...
513	514	10442	11	30	20	1997-02-11
514	515	10442	54	80	20	1997-02-11
515	516	10442	66	60	20	1997-02-11
516	517	10443	11	6	66	1997-02-12
517	518	10443	28	12	66	1997-02-12

518 rows × 6 columns

Join the resulting DataFrame with the products and customers tables using ProductID and CustomerID respectively, to get detailed information about which customers ordered which products.

In [16]:
```
products.head() #To view the table "products"
```

Out[16]:

	productid	productname	supplierid	categoryid	unit	price
0	1	Chais	1	1	10 boxes x 20 bags	18.00
1	2	Chang	1	1	24 - 12 oz bottles	19.00
2	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00
3	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00
4	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

In [17]:
```
#To merge resulting dataframe to the the "products" table, specifying the necess

Mart_Merge = pd.merge(Mart_Merge,products[["productid","productname","unit","pri
```

In [18]:
```
Mart_Merge.head(2)
```

Out[18]:

	orderdetailid	orderid	productid	quantity	customerid	orderdate	productname	u
0	1	10248	11	12	90	1996-07-04	Queso Cabrales	1 p
1	2	10248	42	10	90	1996-07-04	Singaporean Hokkien Fried Mee	3 1 pk

In [19]:
```
customers #To view the "customers" table
```

Out[19]:

	customerid	customername	contactname	address	city	postalcode	cour
0	1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germ
1	2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Me
2	3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Me
3	4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	
4	5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Swe
...	...	...	...	...	...	...	
86	87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	90110	Finl
87	88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	08737-363	B
88	89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	l
89	90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finl
90	91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Pol

91 rows × 7 columns

In [20]:
```
#To join resulting dataframe to the "customers" table, specifying necessary colu

Mart_Merge = pd.merge(Mart_Merge[["orderdate","quantity","customerid","productna
```

In [21]:
```
Mart_Merge.head(2)
```

Out[21]:

	orderdate	quantity	customerid	productname	unit	price	supplierid	categoryid
0	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.0	5	4
1	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.0	20	5

```
In [22]: Mart_Merge.groupby("customername")["productname"].sum() #To return the products
```

Out[22]:

customername	productname
Ana Trujillo Emparedados y helados	GudbrandsdalsostOutback Lager
Antonio Moreno Taquería	Queso Cabrales
Around the Horn	Guaraná FantásticaRavioli AngeloKonbuValkoinen...
B's Beverages	Aniseed SyrupWimmers gute Semmelknödel
Berglunds snabbköp	Gula MalaccaRaclette CourdavaultVegie-spreadRö...
...	...
Wartian Herkku	Queso Manchego La PastoraInlagd SillIpoh Coffe...
Wellington Importadora	Perth PastiesOriginal Frankfurter grüne SoßeMi...
White Clover Markets	GeitostMozzarella di GiovanniChef Anton's Caju...
Wilman Kala	Queso CabralesSingaporean Hokkien Fried MeeMoz...
Wolski	Gorgonzola TelinoEscargots de Bourgogne

74 rows × 1 columns

dtype: object

- **Filter the merged DataFrame to include only orders placed in the year, 1996.**

```
In [23]: Mart_Merge = pd.DataFrame(Mart_Merge)
```

```
In [24]: Mart_Merge["orderdate"] = pd.to_datetime(Mart_Merge["orderdate"]) #To ensure the
```

```
In [25]: Mart_Merge = Mart_Merge[Mart_Merge["orderdate"].dt.year == 1996] #To include onl
```

```
In [26]: Mart_Merge #To ensure "year" filter has been effected
```

Out[26]:

	orderdate	quantity	customerid	productname	unit	price	supplierid	category
0	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.00	5	
1	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.00	20	
2	1996-07-04	5	90	Mozzarella di Giovanni	24 - 200 g pkgs.	34.80	14	
3	1996-07-05	9	81	Tofu	40 - 100 g pkgs.	23.25	6	
4	1996-07-05	40	81	Manjimup Dried Apples	50 - 300 g pkgs.	53.00	24	
...	...	...	...	...	...	...	...	
400	1996-12-30	120	71	Pâté chinois	24 boxes x 2 pies	24.00	25	
401	1996-12-31	60	83	Scottish Longbreads	10 boxes x 8 pieces	12.50	8	
402	1996-12-31	30	83	Fløtemysost	10 - 500 g pkgs.	21.50	15	
403	1996-12-31	35	83	Lakkalikööri	500 ml	18.00	23	
404	1996-12-31	14	83	Original Frankfurter grüne Soße	12 boxes	13.00	12	

405 rows × 9 columns

- **From the filtered data, identify the top 5 products by order volume in 1996.**

In [27]:
```
Mart_Merge.groupby("productname")["quantity"].sum().sort_values(ascending=False)
```

Out[27]:

productname	quantity
Gorgonzola Telino	444
Camembert Pierrot	370
Steeleye Stout	274
Chartreuse verte	266
Fløtemysost	261

dtype: int64

```
In [28]: Mart_Merge.groupby("productname")["quantity"].sum().sort_values(ascending=True).
```

Out[28]:

productname	quantity
Laughing Lumberjack Lager	5
Gustaf's Knäckebröd	6
Queso Manchego La Pastora	12
Røgede sild	15
Zaanse koeken	16

dtype: int64

```
In [29]: Mart_Merge.groupby("customername")["quantity"].sum().sort_values(ascending=False
```

Out[29]:

customername	quantity
Ernst Handel	837
QUICK-Stop	693
Save-a-lot Markets	567
Frankenversand	553
Hungry Owl All-Night Grocers	490

dtype: int64

Observations:

- The product in highest demand by customers is Gorgonzola Telino.
- Laughing Lumberjack Lager is the least performing product.

- Ernst Handel bought the highest qauntities, among customers.

Aggregation and Multiple Grouping

- **Merge the categories and suppliers tables with the previous DataFrame to include category and supplier details.**

In [30]:
```
categories  #To view the "categories" table
```

Out[30]:

	categoryid	categoryname	description
0	1	Beverages	Soft drinks, coffees, teas, beers, and ales
1	2	Condiments	Sweet and savory sauces, relishes, spreads, an...
2	3	Confections	Desserts, candies, and sweet breads
3	4	Dairy Products	Cheeses
4	5	Grains/Cereals	Breads, crackers, pasta, and cereal
5	6	Meat/Poultry	Prepared meats
6	7	Produce	Dried fruit and bean curd
7	8	Seafood	Seaweed and fish

In [31]:
```
Mart_Merge = pd.merge(Mart_Merge,categories[["categoryid","categoryname"]],on="c
#To merge the previous dataframe to the "categories" table
```

In [ ]:
```
Mart_Merge.head(2)
```

Out[ ]:

	orderdate	quantity	customerid	productname	unit	price	supplierid	categoryid
0	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.0	5	4
1	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.0	20	5

In [32]:
```
suppliers.head() #To view the details of the suppliers' table
```

Out[32]:

	supplierid	suppliername	contactname	address	city	postalcode	country
0	1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	Londona	EC1 4SD	UK
1	2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
2	3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA
3	4	Tokyo Traders	Yoshi Nagase	9-8 Sekimai Musashino-shi	Tokyo	100	Japan
4	5	Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saavedra	Calle del Rosal 4	Oviedo	33007	Spain

In [33]:
```
Mart_Merge = pd.merge(Mart_Merge,suppliers[["supplierid","suppliername"]],on="su
#To merge the previous dataframe to the "suppliers" table
```

In [34]:
```
Mart_Merge.head(2)
```

Out[34]:

	orderdate	quantity	customerid	productname	unit	price	supplierid	categoryid
0	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.0	5	4
1	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.0	20	5

- **Group by CategoryName and SupplierName, and calculate the following metrics for each group:**

Total Quantity sold

Total Revenue

In [35]:
```
Mart_Merge.groupby("categoryname")["quantity"].sum().sort_values(ascending=False
#To return the Total Quantity Sold by category
```

Out[35]:

	categoryname	quantity
0	Dairy Products	2086
1	Beverages	1842
2	Confections	1357
3	Seafood	1286
4	Condiments	962
5	Meat/Poultry	950
6	Grains/Cereals	549
7	Produce	549

In [36]:
```
Mart_Merge.groupby("suppliername")["quantity"].sum().sort_values(ascending=False
#To return the Total Quantity Sold by supplier
```

Out[36]:

	suppliername	quantity
0	Pavlova, Ltd.	857
1	Formaggi Fortini s.r.l.	756
2	Norske Meierier	607
3	Gai pâturage	601
4	Plutzer Lebensmittelgroßmärkte AG	565

In [37]:
```
Mart_Merge.groupby("categoryname").apply(lambda x: (x["quantity"] * x["price"]).
#To return the Total Revenue by product category
```

```
<ipython-input-37-a433eb0fddf7>:1: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  Mart_Merge.groupby("categoryname").apply(lambda x: (x["quantity"] * x["pric
e"]).sum()).sort_values(ascending=False).head(3)
```

Out[37]:

	0
categoryname	
Beverages	67349.0
Dairy Products	55781.0
Confections	39448.0

dtype: float64

In [38]:
```
Mart_Merge.groupby("suppliername").apply(lambda x: (x["quantity"] * x["price"]).
#To return the Total Revenue by supplier
```

```
<ipython-input-38-03c68da3f955>:1: DeprecationWarning: DataFrameGroupBy.apply ope
rated on the grouping columns. This behavior is deprecated, and in a future versi
on of pandas the grouping columns will be excluded from the operation. Either pas
s `include_groups=False` to exclude the groupings or explicitly select the groupi
ng columns after groupby to silence this warning.
  Mart_Merge.groupby("suppliername").apply(lambda x: (x["quantity"] * x["pric
e"]).sum()).sort_values(ascending=False).head(3)
```

Out[38]:

suppliername	0
Aux joyeux ecclésiastiques	41678.00
Pavlova, Ltd.	27273.50
Plutzer Lebensmittelgroßmärkte AG	27254.17

dtype: float64

In [39]:
```
Mart_Merge["totalrev"] = Mart_Merge["quantity"] * Mart_Merge["price"] #Created a
```

In [40]:
```
Mart_Merge.head()
```

Out[40]:

	orderdate	quantity	customerid	productname	unit	price	supplierid	categoryid
0	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.00	5	4
1	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.00	20	5
2	1996-07-04	5	90	Mozzarella di Giovanni	24 - 200 g pkgs.	34.80	14	4
3	1996-07-05	9	81	Tofu	40 - 100 g pkgs.	23.25	6	7
4	1996-07-05	40	81	Manjimup Dried Apples	50 - 300 g pkgs.	53.00	24	7

Observations:

- *Diary Products* have the highest quantity sold, compared to other product categories.
- *Pavlova, Ltd* is the supplier whose products recorded most sales for the company.
- *Beverages* generated the most revenue in 1996.

- *Aux joyeux* products generated most revenue, among other suppliers in 1996.

- **Using the previous DataFrame, slice the data using .loc to view all order records for the top product category identified above. Select only the columns: ProductName, CustomerName, OrderDate, Quantity, and TotalRevenue.**

In [41]:
```
Mart_Merge.set_index("categoryname",inplace=True) #To set the column "categoryna
```

Slicing and Advanced Filtering with .iloc and .loc

In [42]:
```
Mart_Merge.loc[["Beverages", "Dairy Products", "Confections"],  ["productname","
#To view order records for the top 3 Product Categories
```

Out[42]:

	productname	customername	orderdate	quantity	totalrev
categoryname					
Beverages	Chartreuse verte	Hanari Carnes	1996-07-10	42	756.0
Beverages	Guaraná Fantástica	Chop-suey Chinese	1996-07-11	15	67.5
Beverages	Chang	Richter Supermarkt	1996-07-12	20	380.0
Beverages	Chartreuse verte	HILARIÓN-Abastos	1996-07-16	6	108.0
Beverages	Chang	Ernst Handel	1996-07-17	50	950.0
...	...	...	...	...	...
Confections	NuNuCa Nuß-Nougat-Creme	Save-a-lot Markets	1996-12-25	7	98.0
Confections	Gumbär Gummibärchen	Save-a-lot Markets	1996-12-25	70	2186.1
Confections	Tarte au sucre	Hungry Coyote Import Store	1996-12-25	10	493.0
Confections	Sir Rodney's Scones	Princesa Isabel Vinhoss	1996-12-27	10	100.0
Confections	Scottish Longbreads	Vaffeljernet	1996-12-31	60	750.0

218 rows × 5 columns

Using .iloc to extract the first 10 rows from the data to focus on a sample for detailed analysis.

```
In [43]: Mart_Merge.iloc[:10]
```

Out[43]:

	orderdate	quantity	customerid	productname	unit	price	supplieric
categoryname							
Dairy Products	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.00	5
Grains/Cereals	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.00	20
Dairy Products	1996-07-04	5	90	Mozzarella di Giovanni	24 - 200 g pkgs.	34.80	14
Produce	1996-07-05	9	81	Tofu	40 - 100 g pkgs.	23.25	6
Produce	1996-07-05	40	81	Manjimup Dried Apples	50 - 300 g pkgs.	53.00	24
Seafood	1996-07-08	10	34	Jack's New England Clam Chowder	12 - 12 oz cans	9.65	19
Produce	1996-07-08	35	34	Manjimup Dried Apples	50 - 300 g pkgs.	53.00	24
Condiments	1996-07-08	15	34	Louisiana Fiery Hot Pepper Sauce	32 - 8 oz bottles	21.05	2
Grains/Cereals	1996-07-08	6	84	Gustaf's Knäckebröd	24 - 500 g pkgs.	21.00	9
Grains/Cereals	1996-07-08	15	84	Ravioli Angelo	24 - 250 g pkgs.	19.50	26

```
In [44]: Mart_Merge = Mart_Merge #To remove filter in dataframe
```

```
In [45]: Mart_Merge.reset_index(inplace=True) #To reset dataframe back to the default int
```

```
In [46]: Mart_Merge.head(2)
```

Out[46]:

	categoryname	orderdate	quantity	customerid	productname	unit	price	supplie
0	Dairy Products	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.0	
1	Grains/Cereals	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.0	

Filtering with Single and Multiple Conditions

Filter the data to include only orders where Quantity is greater than 50 and TotalRevenue exceeds $500.

In [47]:
```
Mart_Merge[(Mart_Merge['quantity'] > 50) & (Mart_Merge['totalrev'] > 500)]
#To show only orders with quantity above 50 and totalrev above 500
```

Out[47]:

	categoryname	orderdate	quantity	customerid	productname	unit	price	su
30	Condiments	1996-07-17	65	20	Chef Anton's Gumbo Mix	36 boxes	21.35	
43	Confections	1996-07-23	60	20	Pavlova	32 - 500 g boxes	17.45	
45	Seafood	1996-07-23	60	20	Nord-Ost Matjeshering	10 - 200 g glasses	25.89	
53	Dairy Products	1996-07-29	70	25	Raclette Courdavault	5 kg pkg.	55.00	
68	Seafood	1996-08-05	60	63	Boston Crab Meat	24 - 4 oz tins	18.40	
102	Beverages	1996-08-21	100	63	Steeleye Stout	24 - 12 oz bottles	18.00	
132	Beverages	1996-09-04	60	7	Chartreuse verte	750 cc per bottle	18.00	
183	Confections	1996-09-27	70	65	Tarte au sucre	48 pies	49.30	
197	Beverages	1996-10-08	70	71	Steeleye Stout	24 - 12 oz bottles	18.00	
200	Condiments	1996-10-08	80	71	Vegie-spread	15 - 625 g jars	43.90	
244	Meat/Poultry	1996-10-28	70	51	Alice Mutton	20 - 1 kg tins	39.00	
252	Dairy Products	1996-10-30	56	25	Gorgonzola Telino	12 - 100 g pkgs	12.50	
259	Condiments	1996-11-01	70	89	Northwoods Cranberry Sauce	12 - 12 oz jars	40.00	
260	Condiments	1996-11-04	70	63	Northwoods Cranberry Sauce	12 - 12 oz jars	40.00	
261	Confections	1996-11-04	80	63	Teatime Chocolate Biscuits	10 boxes x 12 pieces	9.20	
276	Condiments	1996-11-11	77	20	Gula Malacca	20 - 2 kg bags	19.45	

	categoryname	orderdate	quantity	customerid	productname	unit	price	su
295	Confections	1996-11-21	56	72	Pavlova	32 - 500 g boxes	17.45	
296	Dairy Products	1996-11-21	70	72	Gorgonzola Telino	12 - 100 g pkgs	12.50	
297	Dairy Products	1996-11-21	80	72	Camembert Pierrot	15 - 300 g rounds	34.00	
303	Beverages	1996-11-22	54	63	Chartreuse verte	750 cc per bottle	18.00	
304	Dairy Products	1996-11-22	55	63	Camembert Pierrot	15 - 300 g rounds	34.00	
332	Dairy Products	1996-12-04	70	62	Camembert Pierrot	15 - 300 g rounds	34.00	
334	Seafood	1996-12-05	80	37	Escargots de Bourgogne	24 pieces	13.25	
378	Dairy Products	1996-12-23	60	20	Gorgonzola Telino	12 - 100 g pkgs	12.50	
387	Confections	1996-12-25	70	71	Gumbär Gummibärchen	100 - 250 g bags	31.23	
392	Meat/Poultry	1996-12-26	70	35	Perth Pasties	48 pieces	32.80	
395	Dairy Products	1996-12-27	60	25	Fløtemysost	10 - 500 g pkgs.	21.50	
400	Meat/Poultry	1996-12-30	120	71	Pâté chinois	24 boxes x 2 pies	24.00	
401	Confections	1996-12-31	60	83	Scottish Longbreads	10 boxes x 8 pieces	12.50	

Observations:

- This output helps us to identify high-value orders
- We recorded 29 high-value orders

In [48]:
```
Mart_Merge[Mart_Merge['customername'].str.startswith('A') | Mart_Merge['customer
```

Out[48]:

	categoryname	orderdate	quantity	customerid	productname	unit	price
161	Dairy Products	1996-09-18	1	2	Gudbrandsdalsost	10 kg pkg.	36.00
162	Beverages	1996-09-18	5	2	Outback Lager	24 - 355 ml bottles	15.00
284	Beverages	1996-11-15	25	4	Guaraná Fantástica	12 - 355 ml cans	4.50
285	Grains/Cereals	1996-11-15	25	4	Ravioli Angelo	24 - 250 g pkgs.	19.50
313	Dairy Products	1996-11-27	24	3	Queso Cabrales	1 kg pkg.	21.00
357	Seafood	1996-12-16	20	4	Konbu	2 kg box	6.00
358	Confections	1996-12-16	15	4	Valkoinen suklaa	12 - 100 g bars	16.25
359	Grains/Cereals	1996-12-16	20	4	Gnocchi di nonna Alice	24 - 250 g pkgs.	38.00

Using the full customer list, filter for customers whose CustomerName ends with "son" to see if there's a pattern in purchasing behavior.

In [49]:
```
customers[customers["customername"].str.endswith("son")]
#No customername ends with son
```

Out[49]:

customerid	customername	contactname	address	city	postalcode	country

Filtering Using startswith, endswith, and contains

Filter the products table for products with ProductName containing the keyword "Organic" or starting with "Fresh" to understand the demand for specific product types.

In [50]:
```
products[products["productname"].str.contains("Organic")]
 #Uncle Bob's Organic Dried Pears contains keyword "Organic"
```

```
Out[50]:
```

	productid	productname	supplierid	categoryid	unit	price
6	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.0

```
In [51]: products[products["productname"].str.startswith("Fresh")]
         #No productname starts with Fresh
```

```
Out[51]:
```

productid	productname	supplierid	categoryid	unit	price

Observation:

Uncle Bob's Organic Dried Pears is not part of products in high demand.

Using existing DataFrame, create a new column OrderSize using np.where. Set OrderSize as:

'Large' if Quantity > 100.

'Medium' if 50 < Quantity <= 100.

'Small' if Quantity <= 50.

```
In [52]: Mart_Merge["OrderSize"] = np.where(Mart_Merge["quantity"] > 100, "Large", np.whe
         #To create a new column named "Ordersize" and classify orders based on quantity
```

```
In [53]: Mart_Merge
         #To view output of code above
```

Out[53]:

	categoryname	orderdate	quantity	customerid	productname	unit	price	supp
0	Dairy Products	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.00	
1	Grains/Cereals	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.00	
2	Dairy Products	1996-07-04	5	90	Mozzarella di Giovanni	24 - 200 g pkgs.	34.80	
3	Produce	1996-07-05	9	81	Tofu	40 - 100 g pkgs.	23.25	
4	Produce	1996-07-05	40	81	Manjimup Dried Apples	50 - 300 g pkgs.	53.00	
...	...	...	...	...	...	...	...	
400	Meat/Poultry	1996-12-30	120	71	Pâté chinois	24 boxes x 2 pies	24.00	
401	Confections	1996-12-31	60	83	Scottish Longbreads	10 boxes x 8 pieces	12.50	
402	Dairy Products	1996-12-31	30	83	Fløtemysost	10 - 500 g pkgs.	21.50	
403	Beverages	1996-12-31	35	83	Lakkalikööri	500 ml	18.00	
404	Condiments	1996-12-31	14	83	Original Frankfurter grüne Soße	12 boxes	13.00	

405 rows × 13 columns

Conditional Column Creation with np.where

Create a HighRevenue column where orders with TotalRevenue > $1,000 are marked as True, and others as False.

In [54]:
```
Mart_Merge["HighRevenue"] = np.where(Mart_Merge["totalrev"] > 1000, True, False)
#To create a new column called "High Revenue" and classify orders based on reven
```

In [55]:
```python
Mart_Merge
#To view output of code above
```

Out[55]:

	categoryname	orderdate	quantity	customerid	productname	unit	price	supp
0	Dairy Products	1996-07-04	12	90	Queso Cabrales	1 kg pkg.	21.00	
1	Grains/Cereals	1996-07-04	10	90	Singaporean Hokkien Fried Mee	32 - 1 kg pkgs.	14.00	
2	Dairy Products	1996-07-04	5	90	Mozzarella di Giovanni	24 - 200 g pkgs.	34.80	
3	Produce	1996-07-05	9	81	Tofu	40 - 100 g pkgs.	23.25	
4	Produce	1996-07-05	40	81	Manjimup Dried Apples	50 - 300 g pkgs.	53.00	
...	...	...	...	...	...	...	...	
400	Meat/Poultry	1996-12-30	120	71	Pâté chinois	24 boxes x 2 pies	24.00	
401	Confections	1996-12-31	60	83	Scottish Longbreads	10 boxes x 8 pieces	12.50	
402	Dairy Products	1996-12-31	30	83	Fløtemysost	10 - 500 g pkgs.	21.50	
403	Beverages	1996-12-31	35	83	Lakkalikööri	500 ml	18.00	
404	Condiments	1996-12-31	14	83	Original Frankfurter grüne Soße	12 boxes	13.00	

405 rows × 14 columns

Analyze the proportion of orders that are Large and HighRevenue.

In [56]:
```python
# Filter orders that are Large and HighRevenue
large_and_high_revenue = Mart_Merge[(Mart_Merge['OrderSize'] == 'Large') & (Mart
```

In [57]:
```python
large_and_high_revenue
#To show orders that are large and high revenue
```

Out[57]:

	categoryname	orderdate	quantity	customerid	productname	unit	price	supp
400	Meat/Poultry	1996-12-30	120	71	Pâté chinois	24 boxes x 2 pies	24.0	

In [58]:
```
proportion_large_high_revenue = len(large_and_high_revenue) / len(Mart_Merge)*10
# To calculate the proportion of large orders that are high revenue and compare
```

In [59]:
```
proportion_large_high_revenue
#To display the proportion
```

Out[59]:
```
0.24691358024691357
```

Observations:

- Only one order qualifies as both large and high-revenue
- The proportion of large and high revenue to total orders is very low.

Recommendations

- The business could prioritize strategies that encourage large orders (e.g., offering bulk discounts, volume incentives, or promotions targeting large-scale buyers).
- For products performing below expectations, we can reassess the positioning of the product in the market. Make sure it aligns with the target audience's needs, preferences, and expectations.
- We can develop a loyalty program that rewards top customers for their repeat business. Offer tiered benefits where the more they spend, the more rewards they receive (discounts, exclusive access, early product releases, etc.).