In [1]:
```
import pandas as pd
import mysql.connector
import datetime as dt
```

In [2]:
```
mydb=mysql.connector.connect(host="localhost",user="root",password="gigaberosql",database='classicmodels')
cursor=mydb.cursor()
```

In [3]:
```
customers=pd.read_sql("select * from customers",mydb)
employees=pd.read_sql("select * from employees",mydb)
offices=pd.read_sql("select * from offices",mydb)
orderdetails=pd.read_sql("select * from orderdetails",mydb)
orders=pd.read_sql("select * from orders",mydb)
payments=pd.read_sql("select * from payments",mydb)
productlines=pd.read_sql("select * from productlines",mydb)
products=pd.read_sql("select * from products",mydb)
```

Single Entity

Prepare a list of offices sorted by country, state, city.

In [7]:
```
offices.sort_values(["country","state","city"])
```

Out[7]:

	officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory	officeLocation
5	6	Sydney	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NSW	Australia	2010	APAC	[230, 16, 0, 0, 1, 1, 0, 0, 0, 109, 26, 219, 1...
3	4	Paris	+33 14 723 4404	43 Rue Jouffroy D'abbans	None	None	France	75017	EMEA	[230, 16, 0, 0, 1, 1, 0, 0, 0, 184, 145, 178, ...
4	5	Tokyo	+81 33 224 5000	4-1 Kioicho	None	Chiyoda-Ku	Japan	102-8578	Japan	[230, 16, 0, 0, 1, 1, 0, 0, 0, 78, 14, 159, 11...
6	7	London	+44 20 7877 2041	25 Old Broad Street	Level 7	None	UK	EC2N 1HN	EMEA	[230, 16, 0, 0, 1, 1, 0, 0, 0, 47, 247, 201, 8...
0	1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA	[230, 16, 0, 0, 1, 1, 0, 0, 0, 211, 218, 52, 1...
1	2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA	[230, 16, 0, 0, 1, 1, 0, 0, 0, 196, 11, 34, 82...
2	3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA	[230, 16, 0, 0, 1, 1, 0, 0, 0, 45, 234, 147, 2...

How many employees are there in the company?

In [11]:
```
len(employees.employeeNumber.unique())
```

Out[11]:
```
23
```

What is the total of payments received?

In [18]:
```
x=payments["amount"].sum()
"{:,}".format(round(x,2))
```

Out[18]:
```
'8,853,839.23'
```

List the product lines that contain 'Cars'.

In [32]:
```
list=productlines[productlines["productLine"].str.contains("Cars",case=False)]["productLine"].unique()
for i in list:
    print(i)
```

```
Classic Cars
Vintage Cars
```

Report total payments for October 28, 2004.

In [50]:
```
filtered=payments[(payments.paymentDate.dt.year==2004)&(payments.paymentDate.dt.month_name()=="October")&(payments.paymentDate.dt.day==28)]
sum=filtered["amount"].sum()
"{:,}".format(sum)
```

Out[50]:
```
'47,411.33'
```

In [53]:
```
"{:,}".format(payments[payments.paymentDate=="2004-10-28"]["amount"].sum())
```

Out[53]:
```
'47,411.33'
```

Report those payments greater than $100,000.

In [56]:
```
payments[payments["amount"]>100000].sort_values('amount')
```

Out[56]:

	checkNumber	paymentDate	amount	customerNumber
4	AE215433	2005-03-05	101244.59	124
192	KM172879	2003-12-26	105743.00	148
187	KI131716	2003-08-15	111654.40	124
151	ID10962	2004-12-31	116208.40	141
170	JE105477	2005-03-18	120166.58	141

List the products in each product line.

In [78]:
```
products.groupby("productLine")["productCode"].apply(" , ".join).reset_index()
```

Out[78]:

	productLine	productCode
0	Classic Cars	S10_1949 , S10_4757 , S10_4962 , S12_1099 , S1...
1	Motorcycles	S10_1678 , S10_2016 , S10_4698 , S12_2823 , S1...
2	Planes	S18_1662 , S18_2581 , S24_1785 , S24_2841 , S2...
3	Ships	S18_3029 , S24_2011 , S700_1138 , S700_1938 , ...
4	Trains	S18_3259 , S32_3207 , S50_1514
5	Trucks and Buses	S12_1666 , S12_4473 , S18_1097 , S18_2319 , S1...
6	Vintage Cars	S18_1342 , S18_1367 , S18_1749 , S18_2248 , S1...

How many products in each product line?

In [80]:
```
products.groupby("productLine")["productCode"].count().reset_index()
```

Out[80]:

	productLine	productCode
0	Classic Cars	38
1	Motorcycles	13
2	Planes	12
3	Ships	9
4	Trains	3
5	Trucks and Buses	11
6	Vintage Cars	24

What is the minimum payment received?

In [82]:
```
payments["amount"].min()
```

Out[82]:
```
615.45
```

List all payments greater than twice the average payment.

In [86]:
```
payments[payments.amount>2*payments.amount.mean()]
```

Out[86]:

	checkNumber	paymentDate	amount	customerNumber
4	AE215433	2005-03-05	101244.59	124
12	AL493079	2005-05-23	75020.13	323
20	BG255406	2004-08-28	85410.87	124
66	DJ15149	2003-11-03	85559.12	321
90	ET64396	2005-04-16	83598.04	124
119	GN228846	2003-12-03	85024.46	167
151	ID10962	2004-12-31	116208.40	141
158	IN446258	2005-03-25	65071.26	141
170	JE105477	2005-03-18	120166.58	141
187	KI131716	2003-08-15	111654.40	124
192	KM172879	2003-12-26	105743.00	148
212	MA765515	2004-12-15	82261.22	114
241	NQ865547	2004-03-15	80375.24	239

What is the average percentage markup of the MSRP on buyPrice?

In [91]:
```
(100*(products.MSRP-products.buyPrice)/products.buyPrice).mean()
```

Out[91]:
```
88.70239217134005
```

How many distinct products does ClassicModels sell?

In [93]:
```
products["productCode"].nunique()
```

Out[93]:
```
110
```

Report the name and city of customers who don't have sales representatives?

In [100]:
```
customers.isna().sum()
new_customers=customers[customers.salesRepEmployeeNumber.isna()]
new_customers[["contactFirstName","contactLastName","city"]].head(5)
```

Out[100]:

	contactFirstName	contactLastName	city
6	Zbyszek	Piestrzeniewicz	Warszawa
21	Isabel	de Castro	Lisboa
36	Brydey	Walker	Singapore
41	Horst	Kloss	Cunewalde
44	Alejandra	Camino	Madrid

-- What are the names of executives with VP or Manager in their title?

-- Use the CONCAT function to combine the employee's first name and last name into a single field for reporting.

In [109]:
```
new_employees=employees[employees.jobTitle.str.contains("vp|manager",case=False,regex=True)]
new_employees["concat"]=new_employees.firstName+" "+new_employees.lastName
new_employees["concat"]
```

```
C:\Users\berid\AppData\Local\Temp\ipykernel_12156\1222959855.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  new_employees["concat"]=new_employees.firstName+" "+new_employees.lastName
```

Out[109]:
```
1       Mary Patterson
2        Jeff Firrelli
3     William Patterson
4        Gerard Bondur
5          Anthony Bow
Name: concat, dtype: object
```

Which orders have a value greater than $5,000?

In [118]:
```
orderdetails[orderdetails.quantityOrdered*orderdetails.priceEach>5000]["orderNumber"].unique()
```

Out[118]:
```
array([10251, 10417, 10103, 10112, 10126, 10140, 10150, 10174, 10194,
       10206, 10215, 10228, 10245, 10258, 10280, 10291, 10304, 10312,
       10322, 10347, 10357, 10369, 10381, 10424, 10223, 10285, 10388,
       10399, 10402, 10120, 10134, 10145, 10180, 10188, 10201, 10210,
       10237, 10263, 10275, 10318, 10339, 10354, 10403, 10105, 10119,
       10143, 10167, 10197, 10209, 10222, 10234, 10325, 10359, 10400,
       10414, 10217, 10229, 10281, 10313, 10108, 10122, 10135, 10147,
       10159, 10211, 10266, 10276, 10300, 10310, 10320, 10329, 10341,
       10363, 10375, 10404, 10117, 10127, 10142, 10165, 10176, 10196,
       10208, 10220, 10231, 10247, 10260, 10272, 10282, 10293, 10306,
       10314, 10336, 10348, 10371, 10382, 10395, 10413, 10164, 10216,
       10370, 10121, 10224, 10153, 10230, 10358, 10372, 10393, 10170,
       10287, 10185, 10396, 10115, 10195, 10259, 10349, 10412, 10203,
       10377, 10405, 10181, 10240, 10321, 10331, 10356, 10406, 10410,
       10137, 10148, 10367, 10407, 10155, 10178, 10198, 10250, 10262,
       10274, 10284, 10296, 10307, 10338, 10351, 10360, 10110, 10149,
       10182, 10204, 10241, 10254, 10268, 10288, 10302, 10344, 10379,
       10420, 10109, 10161, 10192, 10212, 10301, 10141, 10151, 10184,
       10207, 10219, 10246, 10271, 10324, 10193, 10227, 10243, 10139,
       10162, 10205, 10214, 10244, 10289, 10421, 10123, 10366, 10419,
       10166, 10350, 10114, 10136, 10171, 10218, 10225, 10239, 10253,
       10278, 10305, 10342, 10383, 10238, 10252, 10265, 10330, 10378,
       10390, 10172, 10226, 10343, 10273, 10292, 10337, 10202, 10299,
       10319, 10168, 10389, 10129, 10177, 10221, 10232, 10248, 10425,
       10279, 10340, 10418, 10401, 10157], dtype=int64)
```

One to many relationships

Report the account representative for each customer.

In [122]:
```
customers.merge(employees,left_on="salesRepEmployeeNumber",right_on="employeeNumber")[["customerNumber","firstName","lastName"]]
```

Out[122]:

	customerNumber	firstName	lastName
0	103	Gerard	Hernandez
1	119	Gerard	Hernandez
2	141	Gerard	Hernandez
3	171	Gerard	Hernandez
4	209	Gerard	Hernandez
...	...	...	...
95	298	Martin	Gerard
96	344	Martin	Gerard
97	376	Martin	Gerard
98	458	Martin	Gerard
99	484	Martin	Gerard

100 rows × 3 columns

Report total payments for Atelier graphique.

In [126]:
```
merged=payments.merge(customers,on="customerNumber")
merged[merged.customerName=="Atelier graphique"]["amount"].sum()
```

Out[126]:
```
22314.36
```

In [137]:
```
x=customers[customers.customerName=="Atelier graphique"]["customerNumber"]
payments[payments.customerNumber==int(x)]["amount"].sum()
```

Out[137]:
```
22314.36
```

Report the total payments by date

In [11]:
```
payments.groupby("paymentDate")["amount"].sum().reset_index().sort_values("paymentDate")
```

Out[11]:

	paymentDate	amount
0	2003-01-16	10223.83
1	2003-01-28	10549.01
2	2003-01-30	5494.78
3	2003-02-16	50218.95
4	2003-02-20	53959.21
...	...	...
227	2005-05-20	29070.38
228	2005-05-23	75020.13
229	2005-05-25	30253.75
230	2005-06-03	12432.32
231	2005-06-09	46656.94

232 rows × 2 columns

Report the products that have not been sold

In [16]:
```
sold=orderdetails["productCode"].unique()
products[products["productCode"].isin(sold)==False]
```

Out[16]:

	productCode	productName	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	productLine
40	S18_3233	1985 Toyota Supra	1:18	Highway 66 Mini Classics	This model features soft rubber tires, working...	7733	57.01	107.57	Classic Cars

List the amount paid by each customer.

In [21]:
```
merged=customers.merge(payments,how="left",on="customerNumber")
merged["customerName"]=merged["customerName"].str.lower()
merged.groupby("customerName")["amount"].sum().reset_index().sort_values("customerName")
```

Out[21]:

	customerName	amount
0	alpha cognac	60483.36
1	american souvenirs inc	0.00
2	amica models & co.	82223.23
3	ang resellers	0.00
4	anna's decorations, ltd	137034.22
...	...	...
117	vida sport, ltd	108777.92
118	vitachrome inc.	72497.64
119	volvo model replicas, co	43680.65
120	warburg exchange	0.00
121	west coast collectables co.	43748.72

122 rows × 2 columns

How many orders have been placed by Herkku Gifts?

In [38]:
```
x=customers[customers.customerName.str.contains("herkku",case=False,regex=True)]["customerNumber"]
orders[orders["customerNumber"]==int(x)]["orderNumber"].nunique()
```

Out[38]:
```
3
```

Who are the employees in Boston?

In [46]:
```
x=offices[offices.city=="Boston"]["officeCode"]
employees[employees.officeCode.isin(x)]
```

Out[46]:

	employeeNumber	lastName	firstName	extension	email	reportsTo	jobTitle	officeCode
8	1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	1143.0	Sales Rep	2
9	1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	1143.0	Sales Rep	2

Report those payments greater than $100,000. Sort the report so the customer who made the highest payment appears first.

In [83]:
```
grouped=payments.groupby("customerNumber")["amount"].sum().reset_index()
filtered=grouped[grouped.amount>100000]
filtered.merge(customers,on="customerNumber")[["customerName","amount"]].sort_values("amount",ascending=False)
```

Out[83]:

	customerName	amount
5	Euro+ Shopping Channel	715738.98
3	Mini Gifts Distributors Ltd.	584188.24
0	Australian Collectors, Co.	180585.07
9	Muscle Machine Inc	177913.95
8	Dragon Souveniers, Ltd.	156251.03
18	Down Under Souveniers, Inc	154622.08
12	AV Stores, Co.	148410.09
13	Anna's Decorations, Ltd	137034.22
17	Corporate Gift Ideas Co.	132340.78
7	Saveley & Henriot, Co.	130305.35
14	Rovelli Gifts	127529.69
20	Reims Collectables	126983.19
1	La Rochelle Gifts	116949.68
21	Online Diecast Creations Co.	116449.29
24	Kelly's Gift Shop	114497.19
23	Corrida Auto Replicas, Ltd	112440.09
15	Vida Sport, Ltd	108777.92
4	Land of Toys Inc.	107639.94
6	Danish Wholesale Imports	107446.50
22	Tokyo Collectables, Ltd	105548.73
11	Handji Gifts& Co	105420.57
10	Technics Stores Inc.	104545.22
2	Baane Mini Imports	104224.79
19	Suominen Souveniers	103896.74
16	Mini Creations Ltd.	101872.52

List the value of 'On Hold' orders

In [100]:
```
merged=orders[orders.status.str.contains("on hold",case=False)].merge(orderdetails,on="orderNumber")
merged["value"]=merged.quantityOrdered*merged.priceEach
"{:,}".format(round(merged.value.sum()))+" $"
```

Out[100]:
```
'169,576 $'
```

Report the number of orders 'On Hold' for each customer

In [185]:
```
orders[orders.status=="On Hold"].groupby("customerNumber")["orderNumber"].nunique()
```

Out[185]:
```
customerNumber
144    1
328    1
362    1
450    1
Name: orderNumber, dtype: int64
```

Many to many relationship

List products sold by order date

In [13]:
```
orders.merge(orderdetails,on="orderNumber").merge(products,on="productCode")[["productName","orderNumber","orderDate"]]\
.sort_values(["orderDate","productName"])
```

Out[13]:

	productName	orderNumber	orderDate
25	1911 Ford Town Car	10100	2003-01-06
0	1917 Grand Touring Sedan	10100	2003-01-06
50	1932 Alfa Romeo 8C2300 Spider Sport	10100	2003-01-06
75	1936 Mercedes Benz 500k Roadster	10100	2003-01-06
128	1928 Mercedes-Benz SSK	10101	2003-01-09
...	...	...	...
715	1982 Camaro Z28	10424	2005-05-31
879	1992 Ferrari 360 Spider red	10425	2005-05-31
687	1996 Peterbilt 379 Stake Bed with Outrigger	10424	2005-05-31
798	1998 Chrysler Plymouth Prowler	10425	2005-05-31
1072	Diamond T620 Semi-Skirted Tanker	10425	2005-05-31

2996 rows × 3 columns

List the order dates in descending order for orders for the 1940 Ford Pickup Truck

In [27]:
```
x=products[products.productName.str.contains("1940 ford pickup truck",case=False)]["productCode"]
merged=orders.merge(orderdetails,on="orderNumber")
merged[merged.productCode.isin(x)][["productCode","orderNumber","orderDate"]].sort_values("orderNumber",ascending=False).head()
```

Out[27]:

	productCode	orderNumber	orderDate
2979	S18_1097	10424	2005-05-31
2870	S18_1097	10411	2005-05-01
2723	S18_1097	10391	2005-03-09
2631	S18_1097	10381	2005-02-17
2527	S18_1097	10370	2005-01-20

List the names of customers and their corresponding order number where a particular order from that customer has a value greater than $25,000?

In [45]:
```
merged=customers.merge(orders,on="customerNumber").merge(orderdetails,on="orderNumber")
merged["value"]=merged.quantityOrdered*merged.priceEach
grouped=merged.groupby(["orderNumber","customerName"],as_index=False)["value"].sum().reset_index()
#grouped["customerName"]=grouped["customerName"].str.lower()
grouped[grouped.value>25000].sort_values(["customerName","value"])
```

Out[45]:

	index	orderNumber	customerName	value
232	232	10332	AV Stores, Co.	47159.11
10	10	10110	AV Stores, Co.	48425.69
206	206	10306	AV Stores, Co.	52825.29
78	78	10178	Alpha Cognac	33818.34
193	193	10293	Amica Models & Co.	33924.24
...	...	...	...	...
125	125	10225	Vida Sport, Ltd	47375.92
187	187	10287	Vida Sport, Ltd	61402.00
224	224	10324	Vitachrome Inc.	44400.50
115	115	10215	West Coast Collectables Co.	36070.47
216	216	10316	giftsbymail.co.uk	46788.14

192 rows × 4 columns

List the names of products sold at less than 80% of the MSRP

In [29]:
```
merged=orderdetails.merge(products,on="productCode")[["productName","priceEach","MSRP"]]
results=merged[merged.priceEach<0.8*merged.MSRP]["productName"].unique()
pd.Series(results).head()
```

Out[29]:
```
0    1952 Alpine Renault 1300
1       1996 Moto Guzzi 1100i
2         1972 Alfa Romeo GTA
3           1957 Chevy Pickup
4             1993 Mazda RX-7
dtype: object
```

Reports those products that have been sold with a markup of 100% or more (i.e., the priceEach is at least twice the buyPrice)

In [36]:
```
merged=orderdetails.merge(products,on="productCode")[["productName","priceEach","buyPrice"]]
results=merged[merged.priceEach>2*merged.buyPrice]["productName"].unique()
pd.Series(results).head()
```

Out[36]:
```
0                 1952 Alpine Renault 1300
1    2003 Harley-Davidson Eagle Drag Bike
2                       1968 Ford Mustang
3                       2001 Ferrari Enzo
4                        2002 Suzuki XREO
dtype: object
```

List the products ordered on a Monday

In [50]:
```
merged=orders.merge(orderdetails,on="orderNumber")[["orderNumber","productCode","orderDate"]]
merged["dayName"]=merged.orderDate.dt.day_name()
merged[merged.orderDate.dt.day_name()=="Monday"]
```

Out[50]:

	orderNumber	productCode	orderDate	dayName
0	10100	S18_1749	2003-01-06	Monday
1	10100	S18_2248	2003-01-06	Monday
2	10100	S18_4409	2003-01-06	Monday
3	10100	S24_3969	2003-01-06	Monday
54	10106	S18_1662	2003-02-17	Monday
...	...	...	...	...
2972	10423	S18_2949	2005-05-30	Monday
2973	10423	S18_2957	2005-05-30	Monday
2974	10423	S18_3136	2005-05-30	Monday
2975	10423	S18_3320	2005-05-30	Monday
2976	10423	S24_4258	2005-05-30	Monday

407 rows × 4 columns

What is the quantity on hand for products listed on 'On Hold' orders?

In [58]:
```
x=orders[orders.status=="On Hold"]["orderNumber"]
results=orderdetails[orderdetails.orderNumber.isin(x)].merge(products,on="productCode")\
[["orderNumber","productCode","quantityInStock"]]
results.quantityInStock.sum()
```

Out[58]:
```
202811
```

Regular expressions

Find products containing the name 'Ford'

In [64]:
```
products[products.productName.str.contains("ford",case=False,regex=True)].head()
```

Out[64]:

	productCode	productName	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	productLine
6	S12_1099	1968 Ford Mustang	1:12	Autoart Studio Design	Hood, doors and trunk all open to reveal highl...	68	95.34	194.57	Classic Cars
12	S12_3891	1969 Ford Falcon	1:12	Second Gear Diecast	Turnable front wheels; steering function; deta...	1049	83.05	173.02	Classic Cars
16	S18_1097	1940 Ford Pickup Truck	1:18	Studio M Art Models	This model features soft rubber tires, working...	2613	58.33	116.67	Trucks and Buses
26	S18_2248	1911 Ford Town Car	1:18	Motor City Art Classics	Features opening hood, opening doors, opening ...	540	33.30	60.54	Vintage Cars
28	S18_2325	1932 Model A Ford J-Coupe	1:18	Autoart Studio Design	This model features grille-mounted chrome horn...	9354	58.48	127.13	Vintage Cars

List products ending in 'ship'

In [69]:
```
products[products.productName.str.lower().str.endswith("ship")]
```

Out[69]:

	productCode	productName	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	productLine
101	S700_2610	The USS Constitution Ship	1:700	Red Start Diecast	All wood with canvas sails. Measures 31 1/2 in...	7083	33.97	72.28	Ships

Report the number of customers in Denmark, Norway, and Sweden

In [77]:
```
customers[customers.country.str.strip().isin(["Denmark","Sweden","Norway"])]
```

Out[77]:

	customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	addressLine2	city	state	postalCode	country	salesRepEmployeeNumber	creditLimit	customerLocation
4	121	Baane Mini Imports	Bergulfsen	Jonas	07-98 9555	Erling Skakkes gate 78	None	Stavern	None	4110	Norway	1504.0	81700.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 18, 212, 149, 1...
11	144	Volvo Model Replicas, Co	Berglund	Christina	0921-12 3555	Berguvsv	None	Lule	None	S-958 22	Sweden	1504.0	53100.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 43, 120, 192, 4...
12	145	Danish Wholesale Imports	Petersen	Jytte	31 12 3555	Vinb	None	Kobenhavn	None	1734	Denmark	1401.0	83400.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 146, 32, 92, 1,...
19	167	Herkku Gifts	Oeztan	Veysel	+47 2267 3215	Brehmen St. 121	PR 334 Sentrum	Bergen	None	N 5804	Norway	1504.0	96800.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 196, 104, 12, 4...
42	227	Heintze Collectables	Ibsen	Palle	86 21 3555	Smagsloget 45	None	Aalborg	None	8200	Denmark	1401.0	120800.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 55, 111, 156, 2...
61	299	Norway Gifts By Mail, Co.	Klaeboe	Jan	+47 2212 1555	Drammensveien 126A	PB 211 Sentrum	Oslo	None	N 0106	Norway	1504.0	95100.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 155, 150, 179, ...
101	448	Scandinavian Gift Ideas	Larsson	Martha	0695-34 6555		None	GÃ¶teborg	None	S-844 67	Sweden	1504.0	116400.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 12, 2, 43, 135,...

What are the products with a product code in the range S700_1000 to S700_1499?

In [78]:
```
products[(products.productCode>"S700_1000")&(products.productCode<"S700_1499")]
```

Out[78]:

	productCode	productName	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	productLine
96	S700_1138	The Schooner Bluenose	1:700	Autoart Studio Design	All wood with canvas sails. Measures 31 1/2 in...	1897	34.0	66.67	Ships

Which customers have a digit in their name?

In [81]:
```
customers[customers.customerName.str.contains("[1-9]",regex=True)]
```

Out[81]:

	customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	addressLine2	city	state	postalCode	country	salesRepEmployeeNumber	creditLimit	customerLocation
35	205	Toys4GrownUps.com	Young	Julie	6265557265	78934 Hillside Dr.	None	Pasadena	CA	90003	USA	1166.0	90700.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 247, 144, 240, ...
84	362	Gifts4AllAges.com	Yoshido	Juri	6175559555	8616 Spinnaker Dr.	None	Boston	MA	51003	USA	1216.0	41900.0	[230, 16, 0, 0, 1, 1, 0, 0, 0, 185, 252, 135, ...

List the names of employees called Dianne or Diane.

In [83]:
```
employees[employees.firstName.str.contains("dianne|diane",regex=True,case=False)]
```

Out[83]:

	employeeNumber	lastName	firstName	extension	email	reportsTo	jobTitle	officeCode
0	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	NaN	President	1

List the products containing ship or boat in their product name

In [85]:
```
products[products.productName.str.contains("ship|boat",regex=True,case=False)]
```

Out[85]:

	productCode	productName	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	productLine
36	S18_3029	1999 Yamaha Speed Boat	1:18	Min Lin Diecast	Exact replica. Wood and Metal. Many extras inc...	4259	51.61	86.02	Ships
101	S700_2610	The USS Constitution Ship	1:700	Red Start Diecast	All wood with canvas sails. Measures 31 1/2 in...	7083	33.97	72.28	Ships

List the products with a product code beginning with S700.

In [88]:
```
products[products.productCode.str.startswith("S700")].head()
```

Out[88]:

	productCode	productName	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	productLine
96	S700_1138	The Schooner Bluenose	1:700	Autoart Studio Design	All wood with canvas sails. Measures 31 1/2 in...	1897	34.00	66.67	Ships
97	S700_1691	American Airlines: B767-300	1:700	Min Lin Diecast	Exact replia with official logos and insignias...	5841	51.15	91.34	Planes
98	S700_1938	The Mayflower	1:700	Studio M Art Models	Measures 31 1/2 inches Long x 25 1/2 inches Hi...	737	43.30	86.61	Ships
99	S700_2047	HMS Bounty	1:700	Unimax Art Galleries	Measures 30 inches Long x 27 1/2 inches High x...	3501	39.83	90.52	Ships
100	S700_2466	America West Airlines B757-200	1:700	Motor City Art Classics	Official logos and insignias. Working steering...	9653	68.80	99.72	Planes

List the names of employees called Larry or Barry.

In [89]:
```
employees[employees.firstName.str.contains("larry|barry",case=False)]
```

Out[89]:

	employeeNumber	lastName	firstName	extension	email	reportsTo	jobTitle	officeCode
15	1501	Bott	Larry	x2311	lbott@classicmodelcars.com	1102.0	Sales Rep	7
16	1504	Jones	Barry	x102	bjones@classicmodelcars.com	1102.0	Sales Rep	7

In [93]:
```
employees[employees.firstName.isin(["Larry","Barry"])]
```

Out[93]:

	employeeNumber	lastName	firstName	extension	email	reportsTo	jobTitle	officeCode
15	1501	Bott	Larry	x2311	lbott@classicmodelcars.com	1102.0	Sales Rep	7
16	1504	Jones	Barry	x102	bjones@classicmodelcars.com	1102.0	Sales Rep	7

List the names of employees with non-alphabetic characters in their names.

In [108]:
```
x=customers[customers.customerName.str.contains("[^a-zA-Z]",regex=True,case=False)]
```

List the vendors whose name ends in Diecast

In [110]:
```
products[products.productVendor.str.lower().str.endswith("diecast")].head()
```

Out[110]:

	productCode	productName	productScale	productVendor	productDescription	quantityInStock	buyPrice	MSRP	productLine
0	S10_1678	1969 Harley Davidson Ultimate Chopper	1:10	Min Lin Diecast	This replica features working kickstand, front...	7933	48.81	95.70	Motorcycles
3	S10_4698	2003 Harley-Davidson Eagle Drag Bike	1:10	Red Start Diecast	Model features, official Harley Davidson logos...	5582	91.02	193.66	Motorcycles
5	S10_4962	1962 LanciaA Delta 16V	1:10	Second Gear Diecast	Features include: Turnable front wheels; steer...	6791	103.42	147.74	Classic Cars
7	S12_1108	2001 Ferrari Enzo	1:12	Second Gear Diecast	Turnable front wheels; steering function; deta...	3619	95.59	207.80	Classic Cars
12	S12_3891	1969 Ford Falcon	1:12	Second Gear Diecast	Turnable front wheels; steering function; deta...	1049	83.05	173.02	Classic Cars

General queries

Who is at the top of the organization (i.e., reports to no one).

In [122]:
```
employees[employees.reportsTo.isna()].iloc[:,:3]
```

Out[122]:

	employeeNumber	lastName	firstName
0	1002	Murphy	Diane

Who reports to William Patterson?

In [127]:
```
x=employees[(employees.firstName=="William")&(employees.lastName=="Patterson")]["employeeNumber"]
employees[employees.reportsTo.isin(x)]
```

Out[127]:

	employeeNumber	lastName	firstName	extension	email	reportsTo	jobTitle	officeCode
17	1611	Fixter	Andy	x101	afixter@classicmodelcars.com	1088.0	Sales Rep	6
18	1612	Marsh	Peter	x102	pmarsh@classicmodelcars.com	1088.0	Sales Rep	6
19	1619	King	Tom	x103	tking@classicmodelcars.com	1088.0	Sales Rep	6

List all the products purchased by Herkku Gifts

In [140]:
```
x=customers[customers.customerName.str.contains("herkku",case=False)]["customerNumber"]
merged=orders.merge(orderdetails,on="orderNumber")
results=merged[merged.customerNumber==int(x)]["productCode"].unique()
pd.Series(results).head()
```

Out[140]:
```
0    S12_1099
1    S12_3380
2    S12_3990
3    S12_4675
4    S18_1129
dtype: object
```

Compute the commission for each sales representative, assuming the commission is 5% of the value of an order. Sort by employee last name and first name.

In [195]:
```
merged=employees.merge(customers,how="left",left_on="employeeNumber",right_on="salesRepEmployeeNumber")\
.merge(orders,on="customerNumber").merge(orderdetails,on="orderNumber")\
[["employeeNumber","firstName","lastName","quantityOrdered","priceEach"]]
merged["comission"]=merged.quantityOrdered*merged.priceEach*0.05
results=merged.groupby(["employeeNumber","firstName","lastName"],as_index=False)["comission"].sum().sort_values(["lastName","firstName"])
results["new_comission"]=results.comission.apply(lambda x :"$ {:,}".format(round(x,1)))
results
```

Out[195]:

	employeeNumber	firstName	lastName	comission	new_comission
6	1337	Loui	Bondur	28474.2875	$ 28,474.3
9	1501	Larry	Bott	36604.8395	$ 36,604.8
8	1401	Pamela	Castillo	43411.0275	$ 43,411.0
2	1188	Julie	Firrelli	19333.1600	$ 19,333.2
11	1611	Andy	Fixter	28129.1295	$ 28,129.1
14	1702	Martin	Gerard	19373.8735	$ 19,373.9
7	1370	Gerard	Hernandez	62928.8905	$ 62,928.9
0	1165	Leslie	Jennings	54076.5270	$ 54,076.5
10	1504	Barry	Jones	35242.6955	$ 35,242.7
12	1612	Peter	Marsh	29229.6880	$ 29,229.7
13	1621	Mami	Nishi	22855.5035	$ 22,855.5
3	1216	Steve	Patterson	25293.7710	$ 25,293.8
1	1166	Leslie	Thompson	17376.6515	$ 17,376.7
4	1286	Foon Yue	Tseng	24410.6335	$ 24,410.6
5	1323	George	Vanauf	33468.8525	$ 33,468.9

What is the difference in days between the most recent and oldest order date in the Orders file?

In [5]:
```
orders["orderDate"].max()-orders["orderDate"].min()
```

Out[5]:
```
Timedelta('876 days 00:00:00')
```

Compute the average time between order date and ship date for each customer ordered by the largest difference.

In [12]:
```
orders.groupby("customerNumber").apply(lambda x :(x["shippedDate"]-x["orderDate"]).mean())\
.reset_index(name="avg_diff").sort_values("avg_diff",ascending=False)
```

Out[12]:

	customerNumber	avg_diff
13	148	14 days 14:24:00
23	177	7 days 12:00:00
28	198	5 days 16:00:00
33	209	5 days 16:00:00
40	240	5 days 12:00:00
...	...	...
96	495	2 days 00:00:00
25	186	1 days 16:00:00
29	201	1 days 16:00:00
79	415	1 days 00:00:00
54	314	1 days 00:00:00

98 rows × 2 columns

What is the value of orders shipped in August 2004?

In [18]:
```
merged=orders.merge(orderdetails,on="orderNumber")[["shippedDate","quantityOrdered","priceEach"]]
merged["value"]=merged.quantityOrdered*merged.priceEach
merged[(merged.shippedDate.dt.year==2004)&(merged.shippedDate.dt.month==8)]["value"].sum()
```

Out[18]:
```
355964.29
```

Compute the total value ordered, total amount paid, and their difference for each customer for orders placed in 2004 and payments received in 2004

In [58]:
```
merged1=customers.merge(orders, how="left",on="customerNumber").merge(orderdetails,how="left",on="orderNumber")\
[["customerNumber","orderDate","quantityOrdered","priceEach"]]
merged1["value"]=merged1.quantityOrdered*merged1.priceEach
grouped1=merged1[merged1.orderDate.dt.year==2004].groupby("customerNumber")["value"].sum().reset_index()

merged2=customers.merge(payments,how="left",on="customerNumber")[["customerNumber","paymentDate","amount"]]
grouped2=merged2[merged2.paymentDate.dt.year==2004].groupby("customerNumber")["amount"].sum().reset_index()

final=grouped1.merge(grouped2,on="customerNumber")
final["difference"]=final.value-final.amount
import numpy as np
final["status"]=np.where(round(final.difference,1)<1,"No Debt","Debt")
final.sort_values("difference",ascending=False)
```

Out[58]:

	customerNumber	value	amount	difference	status
9	141	340830.87	293765.51	4.706536e+04	Debt
8	131	126792.53	85347.32	4.144521e+04	Debt
44	282	67642.09	35806.73	3.183536e+04	Debt
10	144	59019.88	36005.71	2.301417e+04	Debt
29	201	45443.54	37258.94	8.184600e+03	Debt
...	...	...	...	...	...
20	172	53170.38	53170.38	-7.275958e-12	No Debt
27	189	49898.27	49898.27	-7.275958e-12	No Debt
49	314	62253.85	62253.85	-7.275958e-12	No Debt
17	166	105420.57	105420.57	-1.455192e-11	No Debt
3	119	67426.01	67426.01	-1.455192e-11	No Debt

88 rows × 5 columns

List the employees who report to those employees who report to Diane Murphy. Use the CONCAT function to combine the employee's first name and last name into a single field for reporting.

In [74]:
```
x=employees[(employees.firstName=="Diane")&(employees.lastName=="Murphy")]["employeeNumber"]
y=employees[employees.reportsTo.isin(x)]["employeeNumber"]
results=employees[employees.reportsTo.isin(y)]
pd.Series(results["firstName"]+" "+results["lastName"]).reset_index(name="employee")
```

Out[74]:

	index	employee
0	3	William Patterson
1	4	Gerard Bondur
2	5	Anthony Bow
3	20	Mami Nishi

What is the percentage value of each product in inventory sorted by the highest percentage first

In [84]:
```
grouped=orderdetails.groupby("productCode")["priceEach"].mean()
merged=products.merge(grouped,how="left",on="productCode")[["productName","priceEach","quantityInStock"]]
merged["value"]=merged.quantityInStock*merged.priceEach
merged["percent"]=(merged.value/merged.value.sum()*100).round(1)
merged[["productName","percent"]].sort_values("percent",ascending=False)
```

Out[84]:

	productName	percent
1	1952 Alpine Renault 1300	2.9
9	2002 Suzuki XREO	2.6
24	1995 Honda Civic	2.5
39	1992 Ferrari 360 Spider red	2.5
44	1976 Ford Gran Torino	2.4
...	...	...
85	1997 BMW F650 ST	0.0
61	1960 BSA Gold Star DBD34	0.0
6	1968 Ford Mustang	0.0
109	Pont Yacht	0.0
40	1985 Toyota Supra	NaN

110 rows × 2 columns

What is the value of orders shipped in August 2004?

In [88]:
```
merged=orders.merge(orderdetails,on="orderNumber")
merged["value"]=merged.quantityOrdered*merged.priceEach
merged[(merged.shippedDate.dt.year==2004)&(merged.shippedDate.dt.month==8)]\
.groupby("orderNumber")["value"].sum().reset_index()
```

Out[88]:

	orderNumber	value
0	10276	51152.86
1	10277	2611.84
2	10278	33347.88
3	10279	20009.53
4	10280	48298.99
5	10281	39641.43
6	10282	47979.98
7	10283	37527.58
8	10284	32260.16
9	10285	43134.04

What is the ratio the value of payments made to orders received for each month of 2004

In [19]:
```
merged=orders.merge(orderdetails, on="orderNumber")[["orderDate","quantityOrdered","priceEach"]]
filtered=merged[merged.orderDate.dt.year==2004]
filtered["month"]=filtered.orderDate.dt.month_name()
filtered["value"]=filtered.quantityOrdered*filtered.priceEach
grouped1=filtered.groupby("month")["value"].sum().reset_index()

payments["month"]=payments.paymentDate.dt.month_name()
grouped2=payments[payments.paymentDate.dt.year==2004].groupby("month")["amount"].sum().reset_index()
result=grouped1.merge(grouped2,on="month")
result["ratio"]=(result.amount/result.value).round(2)
result.sort_values("ratio")
```

```
C:\Users\berid\AppData\Local\Temp\ipykernel_6764\1503616691.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered["month"]=filtered.orderDate.dt.month_name()
C:\Users\berid\AppData\Local\Temp\ipykernel_6764\1503616691.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered["value"]=filtered.quantityOrdered*filtered.priceEach
```

Out[19]:

	month	value	amount	ratio
3	February	289502.84	106652.01	0.37
10	October	500233.86	185103.43	0.37
6	June	343370.74	185842.86	0.54
4	January	292385.21	234152.13	0.80
8	May	248325.30	208524.42	0.84
5	July	325563.49	284191.48	0.87
9	November	979291.98	857187.30	0.88
1	August	419327.09	378094.30	0.90
0	April	187575.77	173245.96	0.92
11	September	283799.80	476445.53	1.68
7	March	217691.26	404603.21	1.86
2	December	428838.17	819285.62	1.91

What is the difference in the amount received for each month of 2004 compared to 2003?

In [40]:
```
filtered1=payments[payments.paymentDate.dt.year==2003]
filtered1["month"]=filtered1.paymentDate.dt.month_name()
grouped1=filtered1.groupby("month")["amount"].sum().reset_index()

filtered2=payments[payments.paymentDate.dt.year==2004]
filtered2["month"]=filtered2.paymentDate.dt.month_name()
grouped2=filtered2.groupby("month")["amount"].sum().reset_index()

result=grouped1.merge(grouped2,on="month")
result.rename(columns={"amount_x":"amount2003","amount_y":"amount2004"},inplace=True)
result["increase"]=((result.amount2004-result.amount2003)*100/result.amount2003).round(1)
result["increase"]=result["increase"].transform(lambda x: str(x)+" %")
result
```

```
C:\Users\berid\AppData\Local\Temp\ipykernel_6764\72323040.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered1["month"]=filtered1.paymentDate.dt.month_name()
C:\Users\berid\AppData\Local\Temp\ipykernel_6764\72323040.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered2["month"]=filtered2.paymentDate.dt.month_name()
```

Out[40]:

	month	amount2003	amount2004	increase
0	April	136313.92	173245.96	27.1 %
1	August	246204.86	378094.30	53.6 %
2	December	826637.64	819285.62	-0.9 %
3	February	144384.36	106652.01	-26.1 %
4	January	26267.62	234152.13	791.4 %
5	July	158247.00	284191.48	79.6 %
6	June	180218.98	185842.86	3.1 %
7	March	199704.48	404603.21	102.6 %
8	May	159881.97	208524.42	30.4 %
9	November	694292.68	857187.30	23.5 %
10	October	316857.96	185103.43	-41.6 %
11	September	161206.23	476445.53	195.6 %

find out the most popular products that were bought with productcode = S10_2016

In [53]:
```
x=orderdetails[orderdetails.productCode=="S10_2016"]["orderNumber"]
filtered=orderdetails[orderdetails.orderNumber.isin(x)]
filtered.groupby("productCode")["orderNumber"].count().reset_index().sort_values("orderNumber",ascending=False)\
.merge(products,on="productCode")[["productName","orderNumber"]].loc[1:].head(5)
```

Out[53]:

	productName	orderNumber
1	1936 Harley Davidson El Knucklehead	26
2	2003 Harley-Davidson Eagle Drag Bike	24
3	1997 BMW R 1100 S	21
4	1960 BSA Gold Star DBD34	20
5	1969 Harley Davidson Ultimate Chopper	20

ABC reporting: Compute the revenue generated by each customer based on their orders. Also, show each customer's revenue as a percentage of total revenue. Sort by customer name.

In [25]:
```
merged=customers.merge(orders,how="left",on="customerNumber").merge(orderdetails,how="left",on="orderNumber")
merged["value"]=merged.quantityOrdered*merged.priceEach
grouped=merged[["customerName","value"]].groupby("customerName")["value"].sum().reset_index()
grouped["percent"]=(grouped["value"]/grouped["value"].sum()*100).round(1)
#grouped["percent"]=grouped.percent.apply(lambda x:str(x)+" %")
values=[]
for i in grouped["percent"]:
    if i!=0:
        values.append(str(i)+" %")
    else:
        values.append(str("No Revenue"))
grouped["percent"]=values
grouped[["customerName","percent"]].sort_values("percent")
```

Out[25]:

	customerName	percent
20	Boards & Toys Co.	0.1 %
9	Atelier graphique	0.2 %
15	Auto-Moto Classics Inc.	0.2 %
71	Microscale Inc.	0.3 %
46	Frau da Collezione	0.3 %
...	...	...
88	Porto Imports Co.	No Revenue
89	Precious Collectables	No Revenue
91	Raanan Stores, Inc	No Revenue
53	Havel & Zbyszek Co	No Revenue
60	Kremlin Collectables, Co.	No Revenue

122 rows × 2 columns

Compute the profit generated by each customer based on their orders. Also, show each customer's profit as a percentage of total profit. Sort by profit descending.

In [10]:
```python
merged=customers.merge(orders,how="left",on="customerNumber").merge(orderdetails, how="left",on="orderNumber")\
.merge(products,how="left",on="productCode")[["customerName","quantityOrdered","priceEach","buyPrice"]]
merged['profit']=merged.quantityOrdered*(merged.priceEach-merged.buyPrice)
grouped=merged.groupby("customerName")["profit"].sum().reset_index()
grouped["percent"]=(100*grouped.profit/grouped.profit.sum()).round(2)
values=[]
for i in grouped["percent"]:
    if i==0:
        values.append("No Profit")
    else:
        values.append(str(i)+" %")
grouped["percent"]=values
grouped.sort_values("profit",ascending=False)
```

Out[10]:

	customerName	profit	percent
42	Euro+ Shopping Channel	326519.66	8.53 %
76	Mini Gifts Distributors Ltd.	236769.39	6.19 %
80	Muscle Machine Inc	72370.09	1.89 %
11	Australian Collectors, Co.	70311.07	1.84 %
63	La Rochelle Gifts	60875.30	1.59 %
...	...	...	...
45	Franken Gifts, Co	0.00	No Profit
53	Havel & Zbyszek Co	0.00	No Profit
70	Messner Shopping Network	0.00	No Profit
59	Kommission Auto	0.00	No Profit
0	ANG Resellers	0.00	No Profit

122 rows × 3 columns

Compute the revenue generated by each sales representative based on the orders from the customers they serve.

In [17]:
```python
merged=employees.merge(customers,left_on="employeeNumber",right_on="salesRepEmployeeNumber").merge(orders,on="customerNumber").merge(orderdetails,on="orderNumber")\
[["employeeNumber","firstName","lastName","quantityOrdered","priceEach"]]
merged["revenue"]=merged.quantityOrdered*merged.priceEach
grouped=merged.groupby(["employeeNumber","firstName","lastName"],as_index=False)["revenue"].sum().reset_index()
grouped["revenue"]=grouped.revenue.apply(lambda x:str("{:,}".format(round(x),0))+" $")
grouped
```

Out[17]:

	index	employeeNumber	firstName	lastName	revenue
0	0	1165	Leslie	Jennings	1,081,531 $
1	1	1166	Leslie	Thompson	347,533 $
2	2	1188	Julie	Firrelli	386,663 $
3	3	1216	Steve	Patterson	505,875 $
4	4	1286	Foon Yue	Tseng	488,213 $
5	5	1323	George	Vanauf	669,377 $
6	6	1337	Loui	Bondur	569,486 $
7	7	1370	Gerard	Hernandez	1,258,578 $
8	8	1401	Pamela	Castillo	868,221 $
9	9	1501	Larry	Bott	732,097 $
10	10	1504	Barry	Jones	704,854 $
11	11	1611	Andy	Fixter	562,583 $
12	12	1612	Peter	Marsh	584,594 $
13	13	1621	Mami	Nishi	457,110 $
14	14	1702	Martin	Gerard	387,477 $

Compute the profit generated by each sales representative based on the orders from the customers they serve. Sort by profit generated descending.

In [16]:
```python
merged=employees.merge(customers,left_on="employeeNumber",right_on="salesRepEmployeeNumber").merge(orders,on="customerNumber").merge(orderdetails,on="orderNumber").merge(products,on
[["employeeNumber","firstName","lastName","quantityOrdered","priceEach","buyPrice"]]
merged["profit"]=merged.quantityOrdered*(merged.priceEach-merged.buyPrice)
grouped=merged.groupby(["employeeNumber","firstName","lastName"],as_index=False)["profit"].sum().reset_index().sort_values("profit",ascending=False)
grouped["profit"]=grouped.profit.apply(lambda x:str("{:,}".format(round(x,0)))+" $")
grouped
```

Out[16]:

	index	employeeNumber	firstName	lastName	profit
7	7	1370	Gerard	Hernandez	504,645.0 $
0	0	1165	Leslie	Jennings	435,208.0 $
8	8	1401	Pamela	Castillo	340,728.0 $
9	9	1501	Larry	Bott	290,204.0 $
10	10	1504	Barry	Jones	276,659.0 $
5	5	1323	George	Vanauf	269,596.0 $
6	6	1337	Loui	Bondur	234,891.0 $
12	12	1612	Peter	Marsh	230,812.0 $
11	11	1611	Andy	Fixter	222,207.0 $
3	3	1216	Steve	Patterson	197,879.0 $
4	4	1286	Foon Yue	Tseng	194,840.0 $
13	13	1621	Mami	Nishi	181,182.0 $
14	14	1702	Martin	Gerard	156,879.0 $
2	2	1188	Julie	Firrelli	152,119.0 $
1	1	1166	Leslie	Thompson	138,031.0 $

Compute the revenue generated by each product, sorted by product name

In [23]:
```
merged=orderdetails.merge(products,how="right",on="productCode")[["productName","quantityOrdered","priceEach"]]
merged["revenue"]=merged["quantityOrdered"]*merged["priceEach"]
grouped=merged.groupby("productName")["revenue"].sum().reset_index()
grouped["revenue"]=grouped.revenue.apply(lambda x:str("{:,}".format(round(x)))+" $")
grouped.sort_values("productName")
```

Out[23]:

	productName	revenue
0	18th Century Vintage Horse Carriage	85,329 $
1	18th century schooner	112,427 $
2	1900s Vintage Bi-Plane	58,434 $
3	1900s Vintage Tri-Plane	68,276 $
4	1903 Ford Model A	111,529 $
...	...	...
105	The Mayflower	69,532 $
106	The Queen Mary	78,919 $
107	The Schooner Bluenose	56,455 $
108	The Titanic	84,992 $
109	The USS Constitution Ship	66,697 $

110 rows × 2 columns

Compute the profit generated by each product line, sorted by profit descending.

In [32]:
```
merged=products.merge(orderdetails,on="productCode")[["productLine","quantityOrdered","priceEach","buyPrice"]]
merged["profit"]=merged.quantityOrdered*(merged.priceEach-merged.buyPrice)
grouped=merged.groupby("productLine")["profit"].sum().reset_index().sort_values("profit",ascending=False)
grouped["profit"]=grouped.profit.apply(lambda x:str("{:,}".format(round(x)))+" $")
grouped
```

Out[32]:

	productLine	profit
0	Classic Cars	1,526,212 $
6	Vintage Cars	737,268 $
1	Motorcycles	469,255 $
5	Trucks and Buses	400,553 $
2	Planes	365,961 $
3	Ships	261,289 $
4	Trains	65,341 $

Same as Last Year (SALY) analysis: Compute the percentage of sales for each product for 2003 and 2004.

In [48]:
```
merged2003=products.merge(orderdetails,on="productCode").merge(orders,on="orderNumber")[["productName","quantityOrdered","priceEach","orderDate"]]
merged2003["sales"]=merged2003.quantityOrdered*merged2003.priceEach
grouped2003=merged2003[merged2003.orderDate.dt.year==2003].groupby("productName")["sales"].sum().reset_index()
grouped2003["percent"]=(100*grouped2003.sales/grouped2003.sales.sum()).round(2)

merged2004=products.merge(orderdetails,on="productCode").merge(orders,on="orderNumber")[["productName","quantityOrdered","priceEach","orderDate"]]
merged2004["sales"]=merged2004.quantityOrdered*merged2004.priceEach
grouped2004=merged2004[merged2004.orderDate.dt.year==2004].groupby("productName")["sales"].sum().reset_index()
grouped2004["percent"]=(100*grouped2004.sales/grouped2004.sales.sum()).round(2)

result=grouped2003.merge(grouped2004,on="productName")[["productName","percent_x","percent_y"]]
result.rename(columns={"percent_x":"percent_2003","percent_y":"percent_2004"},inplace=True)
result
```

Out[48]:

	productName	percent_2003	percent_2004
0	18th Century Vintage Horse Carriage	0.83	0.99
1	18th century schooner	1.22	1.23
2	1900s Vintage Bi-Plane	0.57	0.68
3	1900s Vintage Tri-Plane	0.64	0.69
4	1903 Ford Model A	1.18	1.24
...	...	...	...
104	The Mayflower	0.75	0.74
105	The Queen Mary	0.71	0.96
106	The Schooner Bluenose	0.52	0.67
107	The Titanic	0.88	0.94
108	The USS Constitution Ship	0.68	0.79

109 rows × 3 columns

Compute the ratio of payments for each customer for 2003 versus 2004.

In [26]:
```
merged2003=customers.merge(payments,on="customerNumber")[["customerNumber","customerName","paymentDate","amount"]]
filtered2003=merged[merged.paymentDate.dt.year==2003]
grouped2003=filtered2003.groupby(["customerNumber","customerName"],as_index=False)["amount"].sum().reset_index()

merged2004=customers.merge(payments,on="customerNumber")[["customerNumber","customerName","paymentDate","amount"]]
filtered2004=merged[merged.paymentDate.dt.year==2004]
grouped2004=filtered2004.groupby(["customerNumber","customerName"],as_index=False)["amount"].sum().reset_index()

result=customers.merge(grouped2003,how="left",on="customerName").merge(grouped2004,how="left",on="customerName")[["customerName","amount_x","amount_y"]].sort_values("customerName")
result.rename(columns={"amount_x":"amount2003","amount_y":"amount2004"},inplace=True)
result["ratio2003"]=(result.amount2003/(result.amount2003+result.amount2004)).round(2)
result["ratio2004"]=(result.amount2004/(result.amount2003+result.amount2004)).round(2)
result[["customerName","ratio2003","ratio2004"]]
```

Out[26]:

	customerName	ratio2003	ratio2004
44	ANG Resellers	NaN	NaN
29	AV Stores, Co.	0.33	0.67
47	Alpha Cognac	NaN	NaN
20	American Souvenirs Inc	NaN	NaN
49	Amica Models & Co.	NaN	NaN
...	...	...	...
27	Vitachrome Inc.	0.08	0.92
11	Volvo Model Replicas, Co	0.18	0.82
107	Warburg Exchange	NaN	NaN
112	West Coast Collectables Co.	0.18	0.82
46	giftsbymail.co.uk	NaN	NaN

122 rows × 3 columns

In [23]:
```
grouped2003
```

Out[23]:

	index	customerNumber	customerName	amount
0	0	103	Atelier graphique	14571.44
1	1	112	Signal Gift Stores	32641.98
2	2	114	Australian Collectors, Co.	53429.11
3	3	121	Baane Mini Imports	51710.33
4	4	124	Mini Gifts Distributors Ltd.	167783.08
...	...	...	...	...
68	68	486	Motor Mint Distributors Inc.	25833.14
69	69	487	Signal Collectibles Ltd.	29997.09
70	70	489	Double Decker Gift Stores, Ltd	22275.73
71	71	495	Diecast Collectables	59265.14
72	72	496	Kelly's Gift Shop	32077.44

73 rows × 4 columns

Find the products sold in 2003 but not 2004.

In [41]:
```
merged=orderdetails.merge(orders,on="orderNumber")[["productCode","orderDate"]]
sold_in_2004=merged[merged.orderDate.dt.year==2004]["productCode"].unique()
sold_in_2003=merged[merged.orderDate.dt.year==2003]["productCode"].unique()
for i in sold_in_2003:
    if i not in (sold_in_2004):
        print(i)
```

In [54]:
```
set(sold_in_2004).intersection(sold_in_2003) # common elements for both lists
set(sold_in_2004).symmetric_difference(sold_in_2003) # uncommon elements
```

Out[54]:
```
set()
```

Find the customers without payments in 2003.

In [61]:
```
x=payments[payments.paymentDate.dt.year==2003]["customerNumber"].unique()
customers[customers.customerNumber.isin(x)==False]["customerName"].head()
```

Out[61]:
```
3          La Rochelle Gifts
6         Havel & Zbyszek Co
16     Diecast Classics Inc.
18          Handji Gifts& Co
20    American Souvenirs Inc
Name: customerName, dtype: object
```

Correlated subqueries

Who reports to Mary Patterson?

In [18]:
```
x=employees[(employees.firstName=="Mary")&(employees.lastName=="Patterson")]["employeeNumber"]
result=employees[employees.reportsTo.isin(x)][["firstName","lastName"]]
result["employee"]=result.firstName+" "+result.lastName
result["employee"]
```

Out[18]:
```
3     William Patterson
4         Gerard Bondur
5           Anthony Bow
20           Mami Nishi
Name: employee, dtype: object
```

Which payments in any month and year are more than twice the average for that month and year (i.e. compare all payments in Oct 2004 with the average payment for Oct 2004)? Order the results by the date of the payment. You will need to use the date functions.

In [33]:
```
payments["year"]=payments["paymentDate"].dt.year
payments["month"]=payments["paymentDate"].dt.month_name()
grouped=payments.groupby(["year","month"])["amount"].mean().reset_index()
merged=payments.merge(grouped,on=["year","month"])
merged.rename(columns={"amount_x":"amount","amount_y":"avg_amount"},inplace=True)
merged[merged.amount>2*merged.avg_amount].sort_values("paymentDate")
```

Out[33]:

	checkNumber	paymentDate	amount	customerNumber	year	month	avg_amount
206	BI507030	2003-04-22	44380.15	148	2003	April	19473.417143
268	KI131716	2003-08-15	111654.40	124	2003	August	41034.143333
18	JN355280	2003-10-26	49539.37	141	2003	October	24373.689231
50	DJ15149	2003-11-03	85559.12	321	2003	November	36541.720000
73	GN228846	2003-12-03	85024.46	167	2003	December	41331.882000
78	KM172879	2003-12-26	105743.00	148	2003	December	41331.882000
250	PB951268	2004-02-13	36070.47	475	2004	February	17775.335000
107	NQ865547	2004-03-15	80375.24	239	2004	March	36782.110000
189	BG255406	2004-08-28	85410.87	124	2004	August	34372.209091
145	HE84936	2004-10-22	53116.99	256	2004	October	26443.347143
183	MA765515	2004-12-15	82261.22	114	2004	December	35621.113913
178	ID10962	2004-12-31	116208.40	141	2004	December	35621.113913
39	AE215433	2005-03-05	101244.59	124	2005	March	48158.511250
42	JE105477	2005-03-18	120166.58	141	2005	March	48158.511250
215	ET64396	2005-04-16	83598.04	124	2005	April	36779.544000
109	AL493079	2005-05-23	75020.13	323	2005	May	30249.881111

Report for each product, the percentage value of its stock on hand as a percentage of the stock on hand for product line to which it belongs. Order the report by product line and percentage value within product line descending. Show percentages with two decimal places.

In [51]:
```
products["percent_of_productLine"]=products.groupby(["productLine"])["quantityInStock"].transform(lambda x:round(100*x/x.sum(),2))
products[["productLine","productName","quantityInStock","percent_of_productLine"]].\
sort_values(["productLine","percent_of_productLine"],ascending=[True,False])
```

Out[51]:

	productLine	productName	quantityInStock	percent_of_productLine
24	Classic Cars	1995 Honda Civic	9772	4.46
75	Classic Cars	2002 Chevy Corvette	9446	4.31
11	Classic Cars	1968 Dodge Charger	9123	4.16
44	Classic Cars	1976 Ford Gran Torino	9127	4.16
20	Classic Cars	1965 Aston Martin DB5	9042	4.13
...	...	...	...	...
47	Vintage Cars	1941 Chevrolet Special Deluxe Cabriolet	2378	1.90
79	Vintage Cars	1936 Mercedes Benz 500k Roadster	2081	1.67
32	Vintage Cars	1928 Mercedes-Benz SSK	548	0.44
26	Vintage Cars	1911 Ford Town Car	540	0.43
90	Vintage Cars	1928 Ford Phaeton Deluxe	136	0.11

110 rows × 4 columns

For orders containing more than two products, report those products that constitute more than 50% of the value of the order

In [73]:
```
grouped=orderdetails.groupby("orderNumber")["productCode"].count().reset_index()
x=grouped[grouped.productCode>2]["orderNumber"].unique()
filtered=orderdetails[orderdetails.orderNumber.isin(x)]
filtered["value"]=filtered.quantityOrdered*filtered.priceEach
filtered["order_value"]=filtered.groupby("orderNumber")["value"].transform(sum)
result=filtered[filtered.value>0.5*filtered.order_value]
result
#products[products.productCode.isin(result.productCode)]["productName"] # the products itself
```

```
C:\Users\berid\AppData\Local\Temp\ipykernel_1332\1821722499.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered["value"]=filtered.quantityOrdered*filtered.priceEach
C:\Users\berid\AppData\Local\Temp\ipykernel_1332\1821722499.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy)
  filtered["order_value"]=filtered.groupby("orderNumber")["value"].transform(sum)
```

Out[73]:

	orderNumber	productCode	quantityOrdered	priceEach	orderLineNumber	value	order_value
1047	10166	S18_3140	43	136.59	2	5873.37	9977.85
2299	10335	S32_1268	44	77.05	1	3390.20	6466.44
2645	10199	S700_1691	48	81.29	2	3901.92	7678.25

In [ ]: