In [11]:
```
#Import the necessary libraries

import pandas as pd
import matplotlib
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sns

#Import the dataset

x = pd.read_csv("Online_Retail.csv",encoding ='latin1')
x
```

Out[11]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/10 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/10 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/10 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/10 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/10 8:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/11 12:50	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/11 12:50	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/11 12:50	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/11 12:50	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/11 12:50	4.95	12680.0	France

541909 rows × 8 columns

In [12]:
```
 #Assign it to a variable called online_rt

online_rt=pd.read_csv("Online_Retail.csv", encoding ='latin1')
online_rt
#Note: if you receive a utf-8 decode error, set encoding = 'latin1' in pd.read_csv()
```

Out[12]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/10 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/10 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/10 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/10 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/10 8:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/11 12:50	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/11 12:50	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/11 12:50	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/11 12:50	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/11 12:50	4.95	12680.0	France

541909 rows × 8 columns

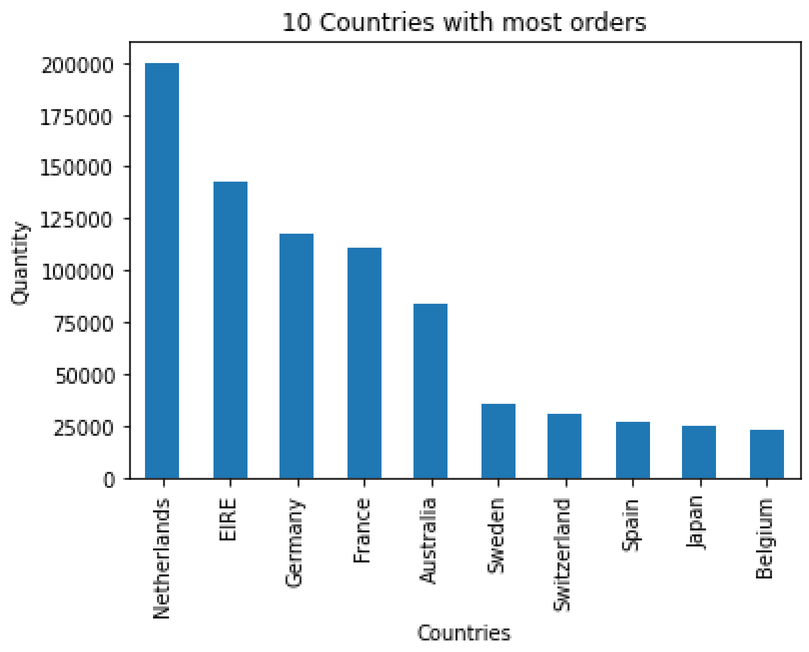
In [42]:
```
#Create a dataframe and extract only 'Country' column
y=pd.DataFrame(online_rt)
y
z=online_rt.loc[0:,('Country')]
z
```

Out[42]:
```
0         United Kingdom
1         United Kingdom
2         United Kingdom
3         United Kingdom
4         United Kingdom
               ...
541904            France
541905            France
541906            France
541907            France
541908            France
Name: Country, Length: 541909, dtype: object
```

In [43]:
```
#Find 'maximum' value of the 'Country' column
y=pd.DataFrame(online_rt)
y.max()
```

Out[43]:
```
Country    Unspecified
dtype: object
```

In [49]:
```
#Create a histogram with the 10 countries that have the most 'Quantity' ordered except UK
countries = online_rt.groupby('Country').sum()
# sort the value and get the first 10 after UK
countries = countries.sort_values(by = 'Quantity',ascending = False)[1:11]
# create the plot
countries['Quantity'].plot(kind='bar')
# Set the title and labels
plt.xlabel('Countries')
plt.ylabel('Quantity')
plt.title('10 Countries with most orders')
# show the plot
plt.show()
```



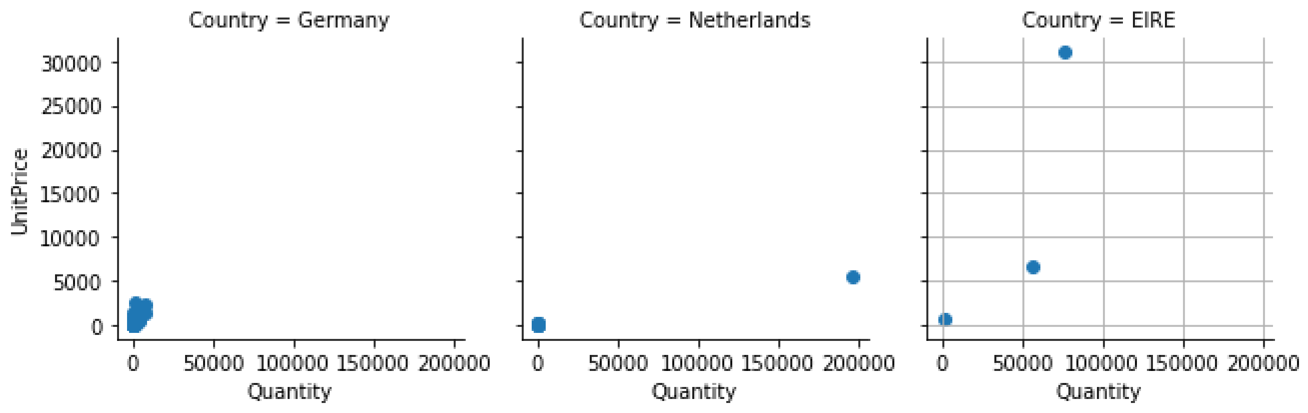
In [8]:
```
#step 5-Exclude negative Quantity entries
countries = online_rt.groupby('Country').sum()
countries
non_neg=countries[countries.Quantity>=0]
non_neg
top10= non_neg.sort_values(by = 'Quantity',ascending = False)[1:11]
top10
```

Out[8]:

	Quantity	UnitPrice	CustomerID
Country			
Australia	83653	4054.750	1.569300e+07
Austria	4827	1701.520	5.021102e+06
Bahrain	260	86.570	2.100270e+05
Belgium	23152	7540.130	2.571829e+07
Brazil	356	142.600	4.086080e+05
Canada	2763	910.580	2.615483e+06
Channel Islands	9479	3738.550	1.128522e+07
Cyprus	6317	3920.070	7.715880e+06
Czech Republic	592	88.150	3.834300e+05
Denmark	8188	1266.950	4.876734e+06
EIRE	142637	48447.190	1.103917e+08
European Community	497	294.050	9.215880e+05
Finland	10666	3786.850	8.699324e+06
France	110480	43031.990	1.076489e+08
Germany	117448	37666.000	1.200751e+08
Greece	1556	713.290	2.008584e+06
Hong Kong	4769	12241.500	0.000000e+00
Iceland	2458	481.210	2.247154e+06
Israel	4353	1079.040	3.164467e+06

	Quantity	UnitPrice	CustomerID
Country			
Italy	7999	3879.390	1.015666e+07
Japan	25218	814.860	4.567292e+06
Lebanon	386	242.440	5.743800e+05
Lithuania	652	99.440	5.366200e+05
Malta	944	666.010	2.158496e+06
Netherlands	200128	6492.550	3.419054e+07
Norway	19247	6529.060	1.350765e+07
Poland	3653	1422.270	4.341972e+06
Portugal	16180	13037.540	1.886480e+07
RSA	352	248.100	7.218680e+05
Saudi Arabia	75	24.110	1.256500e+05
Singapore	5234	25108.890	2.918376e+06
Spain	26824	12633.450	3.268929e+07
Sweden	35637	1806.830	6.790083e+06
Switzerland	30325	6813.690	2.377592e+07
USA	1034	644.980	3.672086e+06
United Arab Emirates	982	229.890	1.018952e+06
United Kingdom	4263829	2245715.474	5.626433e+09
Unspecified	3300	1204.010	3.348046e+06

In [13]:
```python
#Step-6Create a scatterplot with the Quantity per UnitPrice by CustomerID for the top 3 Countries (except UK)
#groupby CustomerID
customers = online_rt.groupby(['CustomerID','Country']).sum()

#outliers with negative price
customers=customers[customers.UnitPrice>0]

#value of the index and put in the column Country
customers['Country']=customers.index.get_level_values(1)

#top three countries
top_countries=['Netherlands','EIRE','Germany']

#select the top countries
customers=customers[customers['Country'].isin(top_countries)]

#scatter Graph
p=sns.FacetGrid(customers,col="Country")

p.map(plt.scatter,"Quantity","UnitPrice",alpha=1)

p.add_legend()

plt.grid(True)

plt.show()
```



In [14]:
```
#Step 7.1 Look at the first line of code in Step 6. And try to figure out if it leads to any kind of problem

#Step 7.1.1 Display the first few rows of that DataFrame.
customers = online_rt.groupby(['CustomerID','Country']).sum().head()
customers
```

Out[14]:

		Quantity	UnitPrice
CustomerID	**Country**		
12346.0	**United Kingdom**	0	2.08
12347.0	**Iceland**	2458	481.21
12348.0	**Finland**	2341	178.71
12349.0	**Italy**	631	605.10
12350.0	**Norway**	197	65.30

In [15]:
```
#Step 7.1.2 Think about what that piece of code does and display the dtype of UnitPrice
customers.UnitPrice.dtype
```

Out[15]:
```
dtype('float64')
```

In [21]:
```
#Step 7.1.3 Pull data from online_rtfor CustomerIDs 12346.0 and 12347.0

display(online_rt[online_rt.CustomerID == 12347.0].
sort_values(by='UnitPrice', ascending = False).head())

display(online_rt[online_rt.CustomerID == 12346.0].
sort_values(by='UnitPrice', ascending = False).head())
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
428966	573511	22423	REGENCY CAKESTAND 3 TIER	6	10/31/11 12:25	12.75	12347.0	Iceland
286637	562032	22423	REGENCY CAKESTAND 3 TIER	3	8/2/11 8:48	12.75	12347.0	Iceland
72267	542237	22423	REGENCY CAKESTAND 3 TIER	3	1/26/11 14:30	12.75	12347.0	Iceland
148300	549222	22423	REGENCY CAKESTAND 3 TIER	3	4/7/11 10:43	12.75	12347.0	Iceland
428967	573511	23173	REGENCY TEAPOT ROSES	2	10/31/11 12:25	9.95	12347.0	Iceland

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	1/18/11 10:01	1.04	12346.0	United Kingdom
61624	C541433	23166	MEDIUM CERAMIC TOP STORAGE JAR	-74215	1/18/11 10:17	1.04	12346.0	United Kingdom

In [15]:
```
#Step 7.2 Reinterpreting the initial problem.
#To reiterate the question that we were dealing with:
"""Create a scatterplot with the Quantity per UnitPrice by CustomerID for the top 3
Countries"""
"""The question is open to a set of different interpretations. We need to disambiguate.
We could do a single plot by looking at all the data from the top 3 countries. Or we
could do one plot per country. To keep things consistent with the rest of the exercise,
let's stick to the latter option. So that's settled.
But top 3 countries with respect to what? Two answers suggest themselves: Total
sales volume (i.e. total quantity sold) or total sales (i.e. revenue). This exercise goes for
sales volume, so let's stick to that."""

#Step 7.2.1 Find out the top 3 countries in terms of sales volume.

sales_volume = online_rt.groupby('Country').Quantity.sum().sort_values(ascending=False)
#We are excluding UK
top3 = sales_volume.index[1:4]
top3
```

Out[15]:
```
Index(['Netherlands', 'EIRE', 'Germany'], dtype='object', name='Country')
```

In [13]:
```
#Step 7.2.2
#Now that we have the top 3 countries, we can focus on the rest of the problem:
#Quantity per UnitPrice by CustomerID.
"""We need to unpack that.
by CustomerID" part is easy. That means we're going to be plotting one dot per
CustomerID's on our plot. In other words, we're going to be grouping by CustomerID."""
#Quantity per UnitPrice" is trickier. Here's what we know:
"""One axis will represent a Quantity assigned to a given customer. This is easy; we can
just plot the total Quantity for each customer. The other axis will represent a UnitPrice
assigned to a given customer. Remember a single customer can have any number of
orders with different prices, so summing up prices isn't quite helpful. Besides it's not
quite clear what we mean when we say "unit price per customer; it sounds like price of
the customer! A reasonable alternative is that we assign each customer the average
amount each has paid per item. So let's settle that question in that manner."""
#Step 7.3 Modify, select and plot data
#Step 7.3.1 Add a column to online_rt called Revenue calculate the revenue (Quantity * UnitPrice) from each sale

#We will use this later to figure out an average price per customer

online_rt['Revenue'] = online_rt.Quantity * online_rt.UnitPrice
online_rt.head()
```

Out[13]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Revenue
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/10 8:26	2.55	17850.0	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	12/1/10 8:26	3.39	17850.0	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/10 8:26	2.75	17850.0	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/10 8:26	3.39	17850.0	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/10 8:26	3.39	17850.0	United Kingdom	20.34

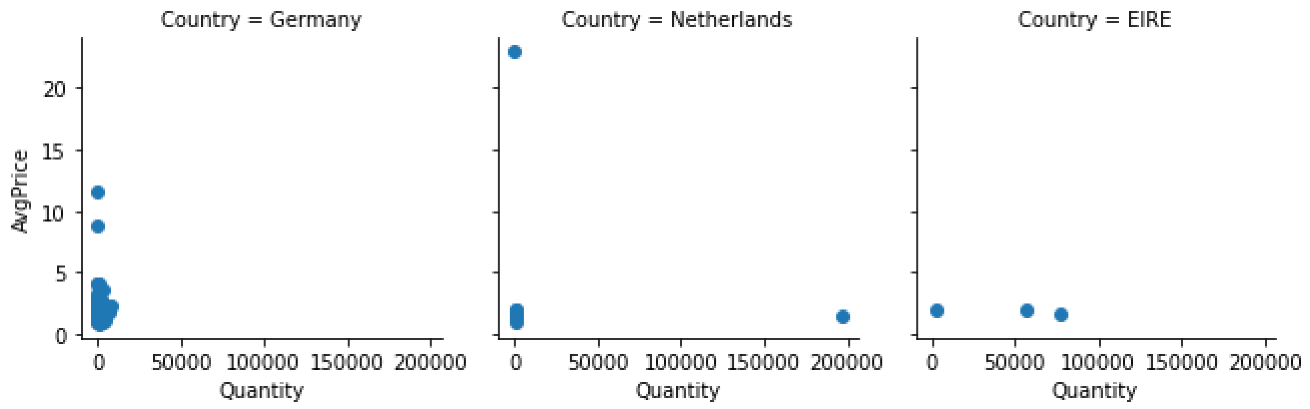
In [16]:
```
#Step 7.3.2 Group by CustomerID and Country and find out the average price (AvgPrice) each customer spends per unit.

grouped = online_rt[online_rt.Country.isin(top3)].groupby(['CustomerID','Country'])

pltable = grouped['Quantity','Revenue'].agg('sum')
pltable['AvgPrice'] = pltable.Revenue / pltable.Quantity

# get the value of the index and put in the column Country
pltable['Country'] = pltable.index.get_level_values(1)
pltable.head()
```

```
C:\Users\rishi\AppData\Local\Temp\ipykernel_7708\795876281.py:5: FutureWarning: Indexing with multiple keys (implicitly
converted to a tuple of keys) will be deprecated, use a list instead.
  pltable = grouped['Quantity','Revenue'].agg('sum')
```

Out[16]:

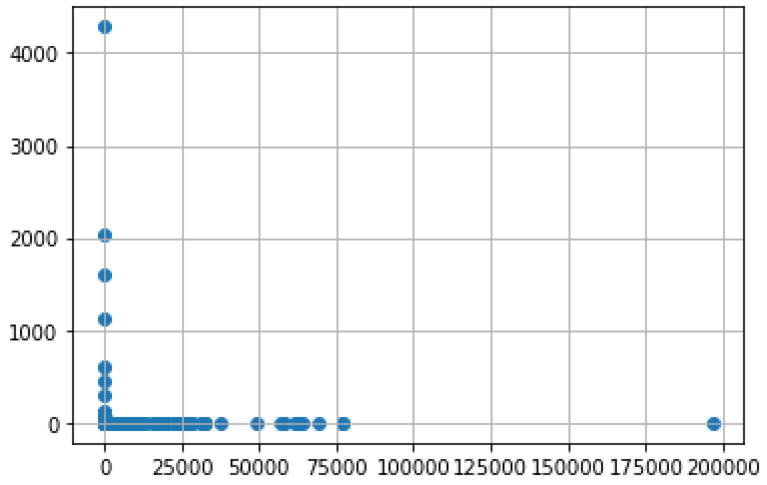
		Quantity	Revenue	AvgPrice	Country
CustomerID	**Country**				
12426.0	**Germany**	258	582.73	2.258643	Germany
12427.0	**Germany**	236	708.37	3.001568	Germany
12468.0	**Germany**	364	724.04	1.989121	Germany
12471.0	**Germany**	7965	18740.92	2.352909	Germany
12472.0	**Germany**	4020	6229.48	1.549622	Germany

In [17]:
```python
#Step 7.3.3 Plot
# creates the FaceGrid
g = sns.FacetGrid(pltable, col="Country")
# map over a make a scatterplot
g.map(plt.scatter,"Quantity","AvgPrice", alpha=1)
# adds legend
g.add_legend();
```

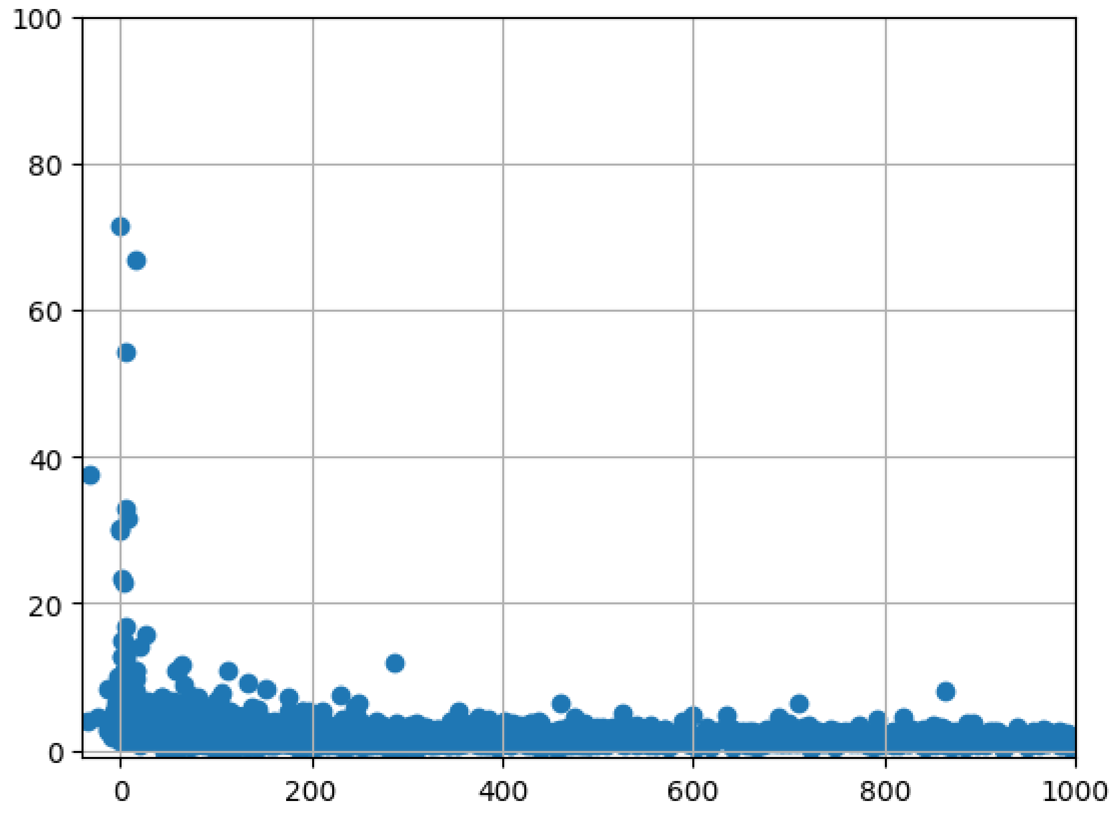


In [38]:
```
#Step 7.4 What to do now?

"""We aren't much better-off than what we started with. The data are still extremely
scattered around and don't seem quite informative."""
#But we shouldn't despair! There are two things to realize:
"""1) The data seem to be skewed towards the axes (e.g. we don't have any values where Quantity = 50000 and
AvgPrice = 5). So that might suggest a trend.
2) We have more data! We've only been looking at the data from 3 different countries and they are plotted on different gr
So:
    we should plot the data regardless of Country and hopefully see a less scattered
graph."""

#Step 7.4.1 Plot the data for each CustomerID on a single graph

grouped = online_rt.groupby(['CustomerID'])
pltable = grouped['Quantity','Revenue'].agg('sum')
pltable['AvgPrice'] = pltable.Revenue / pltable.Quantity

# map over a make a scatterplot
plt.scatter(pltable.Quantity, pltable.AvgPrice)
plt.grid(True)
plt.show()
```

```
C:\Users\rishi\AppData\Local\Temp\ipykernel_15176\3274060240.py:16: FutureWarning: Indexing with multiple keys (implici
tly converted to a tuple of keys) will be deprecated, use a list instead.
  pltable = grouped['Quantity','Revenue'].agg('sum')
```


4000
3000
2000
1000
0
0
25000
50000
75000
100000
125000
150000
175000
200000

In [19]:
```
#Step 7.4.2 Zoom in so we can see that curve more clearly

grouped = online_rt.groupby(['CustomerID','Country'])
pltable = grouped.agg({'Quantity': 'sum',
                         'Revenue': 'sum'})
pltable['AvgPrice'] = pltable.Revenue / pltable.Quantity

# map over a make a scatterplot
plt.scatter(pltable.Quantity, pltable.AvgPrice)

#Zooming in. (I'm starting the axes from a negative value so that
#the dots can be plotted in the graph completely.)
plt.xlim(-40,1000)
plt.ylim(-1,100)
plt.grid(True)

plt.show()
```

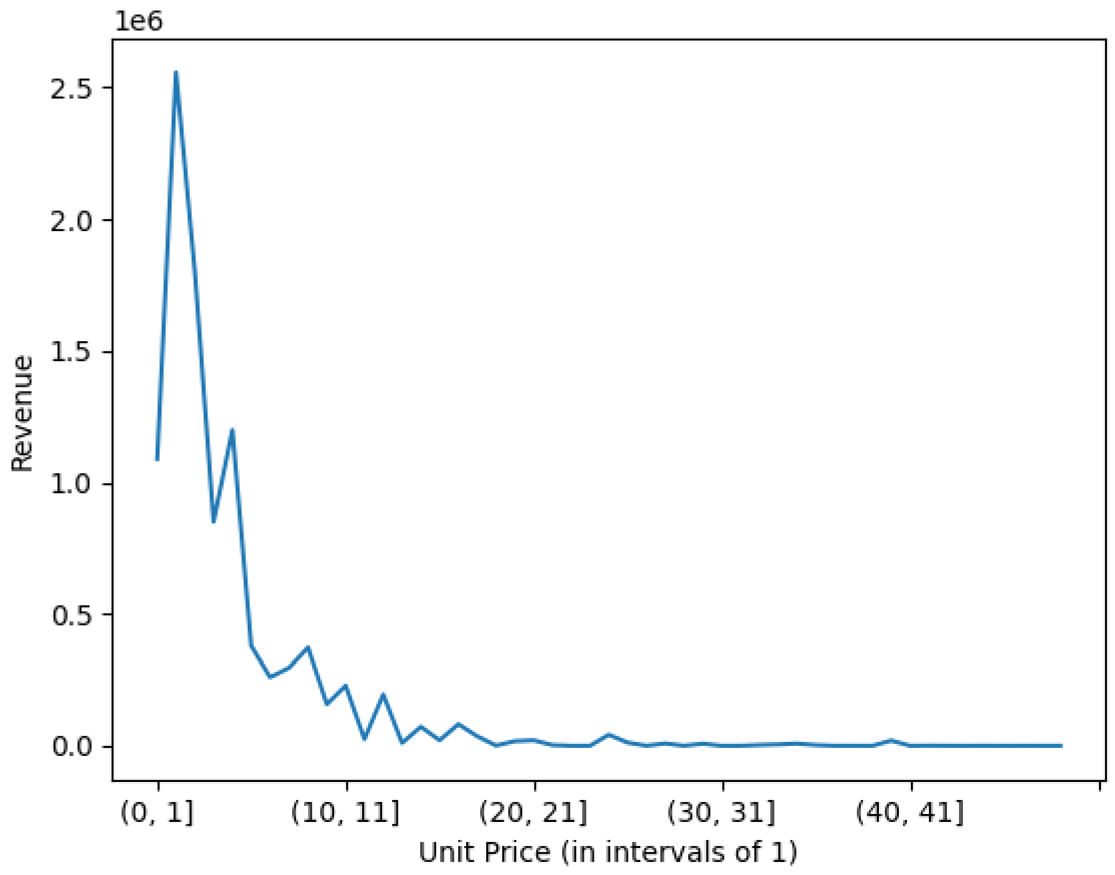


100
80
60
40
20
0
0
200
400
600
800
1000

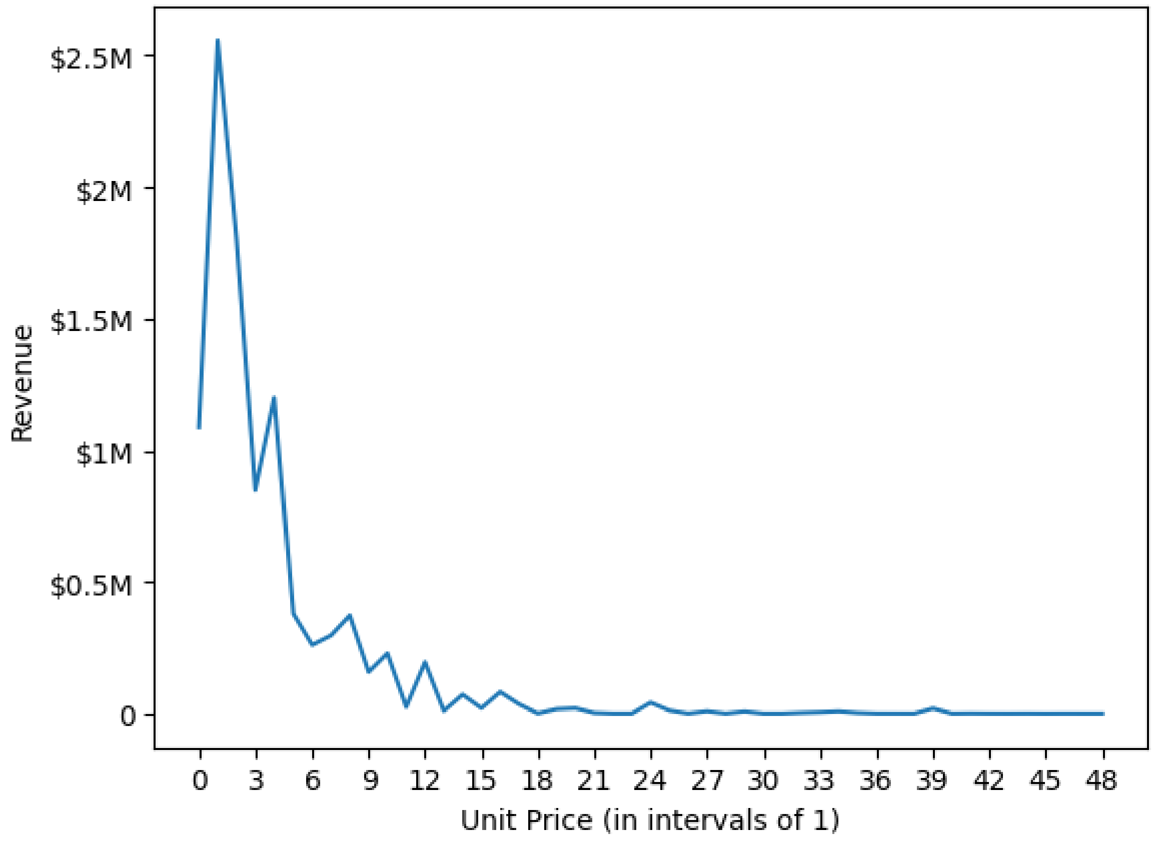
In [21]:
```
#8. Plot a line chart showing revenue (y) per UnitPrice (x).

"""Did Step 7 give us any insights about the data? Sure! As average price increases, the
quantity ordered decreases. But that's hardly surprising. It would be surprising if that
wasn't the case!
Nevertheless the rate of drop in quantity is so drastic, it makes me wonder how our
revenue changes with respect to item price. It would not be that surprising if it didn't
change that much. But it would be interesting to know whether most of our revenue
comes from expensive or inexpensive items, and what that relation looks like.
That is what we are going to do now."""

#8.1 Group UnitPrice by intervals of 1 for prices [0,50), and sum Quantity and Revenue
price_start = 0
price_end = 50
price_interval = 1

#Creating the buckets to collect the data accordingly
container = np.arange(price_start,price_end,price_interval)

#Select the data and sum
revenue_per_price = online_rt.groupby(pd.cut(online_rt.UnitPrice, container)).Revenue.sum()
revenue_per_price.head()
```

Out[21]:
```
UnitPrice
(0, 1]    1089068.414
(1, 2]    2557511.340
(2, 3]    1803381.940
(3, 4]     849919.340
(4, 5]    1199346.770
Name: Revenue, dtype: float64
```

```
In [22]: #8.3 Plot.

         revenue_per_price.plot()
         plt.xlabel('Unit Price (in intervals of '+str(price_interval)+')')
         plt.ylabel('Revenue')
         plt.show()
```



In [26]:
```
#8.4 Make it look nicer.
"""x-axis needs values.
y-axis isn't that easy to read; show in terms of millions"""

revenue_per_price.plot()

plt.xlabel('Unit Price (in intervals of '+str(price_interval)+')')
plt.ylabel('Revenue')

plt.xticks(np.arange(price_start,price_end,3),
           np.arange(price_start,price_end,3))
plt.yticks([0, 500000, 1000000, 1500000, 2000000, 2500000],
           ['0', '$0.5M', '$1M', '$1.5M', '$2M', '$2.5M'])
plt.show()
```



In [ ]:

In [ ]: