upre.vinod,DS20SEP03,prabhasbablu123@gmail.com

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [4]:
```
df = pd.read_csv('D:/New folder/Ecommerce - UK Retailer.csv',encoding='iso-8859-1')
```

In [5]:
```
df.head()
```

Out[5]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

In [6]:
```
df.shape
```

Out[6]:
```
(541909, 8)
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

In [50]:
```
df.dtypes
```

Out[50]:
```
InvoiceNo       object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

In [8]:
```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
```

In [9]:
```
df['Date'] = df['InvoiceDate'].apply(lambda x: str(x).split(" "))
```

In [10]:
```
df.head()
```

Out[10]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0

In [11]:
```
df[['date','hour']] = pd.DataFrame(df['Date'].tolist(), index = df.index)
```

In [12]:
```
df.head()
```

Out[12]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0

In [13]:
```
df.dtypes
```

Out[13]:
```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
Date                   object
date                   object
hour                   object
dtype: object
```

In [14]:
```
df['date'] = pd.to_datetime(df['date'])
```

In [15]:
```
df.dtypes
```

Out[15]:
```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
Date                   object
date           datetime64[ns]
hour                   object
dtype: object
```

2. Check for missing values in all columns and replace them with the appropriate metric(Mean/Median/Mode)

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
Date                0
date                0
hour                0
dtype: int64
```

In [17]:
```
df.dropna(inplace=True)
```

we have null values in categorical columns that are 'Description' and 'CustomerID'.So we cant replace them with mean/median/mode,the customerID is more likely unique key its datatype is float,we have to change it to object but we cant replce them with anything

In [18]:
```
df.shape
```

Out[18]:
```
(406829, 11)
```

5. Add the columns - Month, Day and Hour for the invoice

In [19]:
```
df['Year'] = pd.DatetimeIndex(df['InvoiceDate']).year
df['Month'] = pd.DatetimeIndex(df['InvoiceDate']).month
df['Day'] = pd.DatetimeIndex(df['InvoiceDate']).dayofweek
```

In [ ]:
```
# df.insert(loc=5, column='month', value=df.invoice_date.dt.month)
# df.insert(loc=6, column='day', value=(df.invoice_date.dt.dayofweek))
# df.insert(loc=7, column='hour', value=df.invoice_date.dt.hour)
```

In [20]:
```
df.head()
```

Out[20]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	0

In [21]:
```
df['InvoiceNo'].nunique()
```

Out[21]: 22190

In [22]:
```
df['CustomerID'].nunique()
```

Out[22]: 4372

In [23]:
```
df['Month'].nunique()
```

Out[23]: 12

In [24]:
```
df['Country'].nunique()
```

Out[24]: 37

In [25]:
```
df[['Quantity','UnitPrice']].describe()
```

Out[25]:

	Quantity	UnitPrice
count	406829.000000	406829.000000
mean	12.061303	3.460471
std	248.693370	69.315162
min	-80995.000000	0.000000
25%	2.000000	1.250000
50%	5.000000	1.950000
75%	12.000000	3.750000
max	80995.000000	38970.000000

In [26]:
```
df1 = df.drop(['Date'],axis=1)
```

In [27]:
```
df1.shape
```

Out[27]:
```
(406829, 13)
```

3. Remove duplicate rows

In [28]:
```
df1.drop_duplicates(subset=None, keep="first", inplace=False)
```

Out[28]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Un Kingc
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Un Kingc
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
...	...	...	...	...	...	...	...	
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	Fra
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	Fra
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	Fra
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	Fra
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	Fra

401604 rows × 13 columns

4. Remove rows which have negative values in Quantity column

In [29]:
```
df1=df[df.Quantity>=0]
```

In [30]:
```
df1.shape
```

Out[30]:
```
(397924, 14)
```

6. How many orders made by the customers?

In [31]:
```
orders = df1.groupby('CustomerID')['InvoiceNo'].count().reset_index()
```

In [82]:
```
orders
```

Out[82]:

	CustomerID	InvoiceNo
0	12346.0	1
1	12347.0	182
2	12348.0	31
3	12349.0	73
4	12350.0	17
...	...	...
4334	18280.0	10
4335	18281.0	7
4336	18282.0	12
4337	18283.0	756
4338	18287.0	70

4339 rows × 2 columns

7. TOP 5 customers with higher number of orders

In [88]:
```
top_customers = df1.groupby('CustomerID')['InvoiceNo'].count().nlargest(10)
```

In [89]:
```
top_customers
```

Out[89]:
```
CustomerID
17841.0    7847
14911.0    5677
14096.0    5111
12748.0    4596
14606.0    2700
15311.0    2379
14646.0    2080
13089.0    1818
13263.0    1677
14298.0    1637
Name: InvoiceNo, dtype: int64
```

In [80]:
```
orders.sort_values(by = 'InvoiceNo',ascending=False).head()
```

Out[80]:

	CustomerID	InvoiceNo
4011	17841.0	7847
1880	14911.0	5677
1290	14096.0	5111
326	12748.0	4596
1662	14606.0	2700

8. How much money spent by the customers?

In [34]:
```
df1['revenue'] = df1['UnitPrice']*df1['Quantity']
```

```
<ipython-input-34-228b9eccf4d8>:1: 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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['revenue'] = df1['UnitPrice']*df1['Quantity']
```

In [35]:
```
money_spent = df1.groupby('CustomerID')['revenue'].sum()
money_spent
```

Out[35]:
```
CustomerID
12346.0    77183.60
12347.0     4310.00
12348.0     1797.24
12349.0     1757.55
12350.0      334.40
             ...
18280.0      180.60
18281.0       80.82
18282.0      178.05
18283.0     2094.88
18287.0     1837.28
Name: revenue, Length: 4339, dtype: float64
```

9. TOP 5 customers with highest money spent

In [36]:
```
money_spent.sort_values(ascending=False).head()
```

Out[36]:
```
CustomerID
14646.0    280206.02
18102.0    259657.30
17450.0    194550.79
16446.0    168472.50
14911.0    143825.06
Name: revenue, dtype: float64
```

10. How many orders per month?

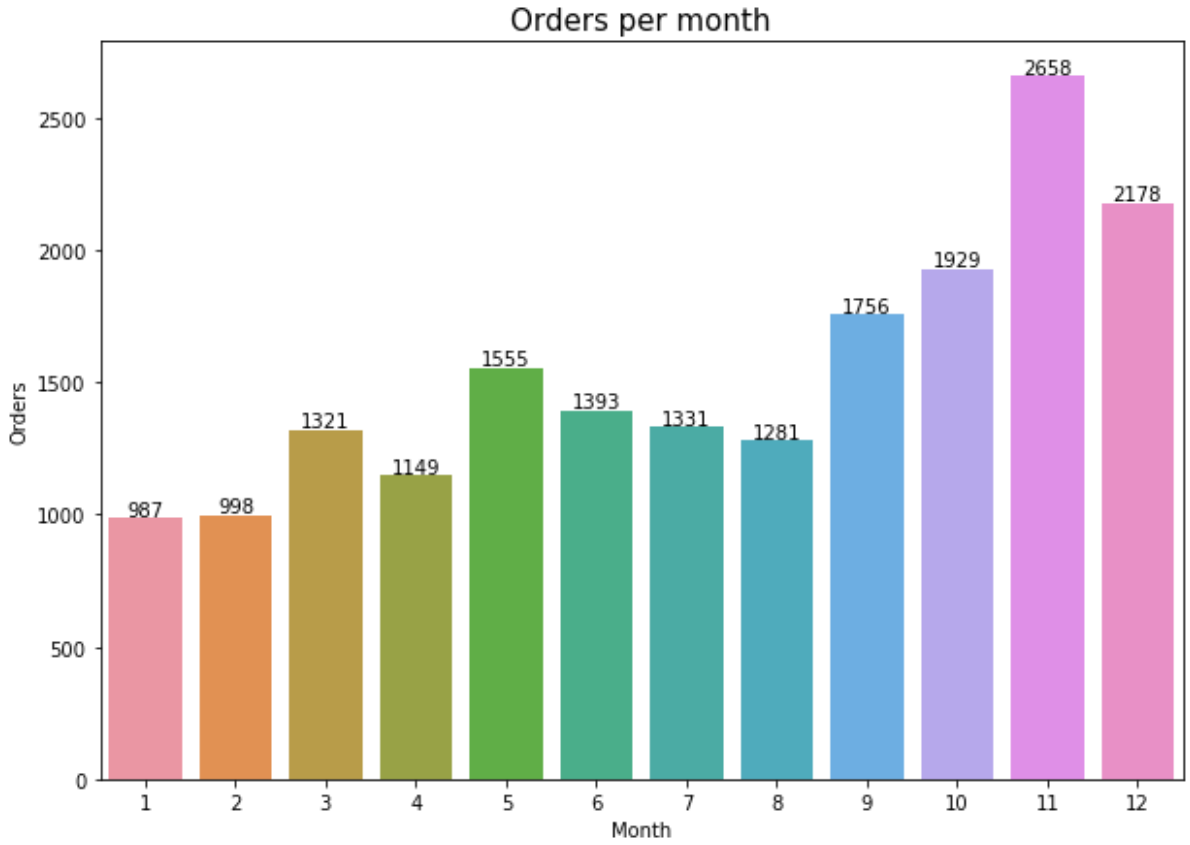
In [37]:
```
monthlywise_orders=df1.groupby('Month')['InvoiceNo'].nunique().reset_index()
```

In [38]:
```
monthlywise_orders
```

Out[38]:

	Month	InvoiceNo
0	1	987
1	2	998
2	3	1321
3	4	1149
4	5	1555
5	6	1393
6	7	1331
7	8	1281
8	9	1756
9	10	1929
10	11	2658
11	12	2178

In [37]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.barplot(x='Month',y='InvoiceNo',data=monthlywise_orders)
ax.set_title('Orders per month',fontsize=15)
ax.set_ylabel('Orders',fontsize=10)

for p in ax.patches:
    height = p.get_height()
    ax.text(x = p.get_x()+(p.get_width()/2),y = height+10,ha='center',s = '{:.
0f}'.format(height))
```



11. How many orders per day?

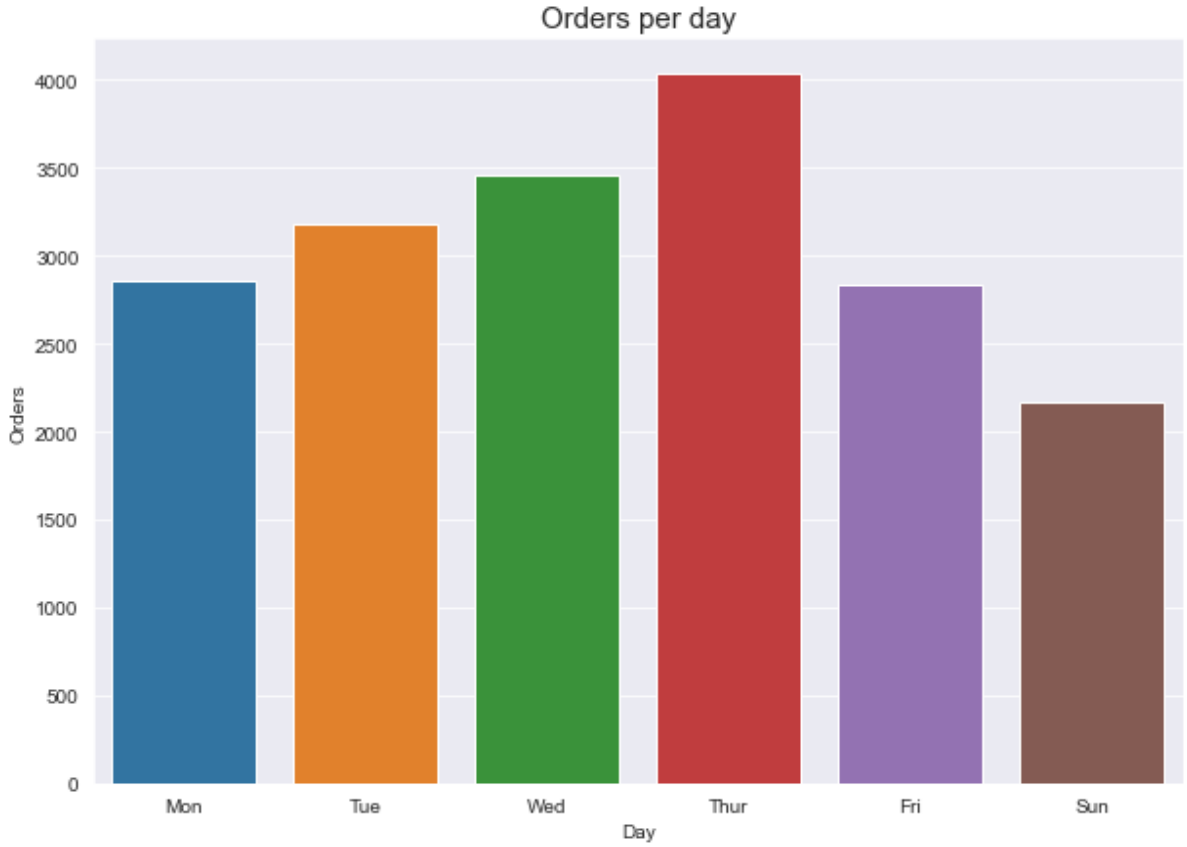
In [238]:
```
orderper_day = df1.groupby('Day')['InvoiceNo'].nunique().reset_index()
```

In [239]:
```
orderper_day
```

Out[239]:

	Day	InvoiceNo
0	0	2863
1	1	3185
2	2	3455
3	3	4033
4	4	2831
5	6	2169

In [244]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.barplot(x='Day',y='InvoiceNo',data=orderper_day)
ax.set_title('Orders per day',fontsize=15)
ax.set_ylabel('Orders',fontsize=10)
ax.set_xticklabels(('Mon','Tue','Wed','Thur','Fri','Sun'), rotation='horizontal', fontsize=10)
plt.show()
```



12. How many orders per hour?

In [101]:
```
df1.insert(loc=7, column='Hour', value=df1.InvoiceDate.dt.hour)
```

In [251]:
```
ordersper_Hour=df1.groupby('Hour')['InvoiceNo'].nunique().reset_index()
```

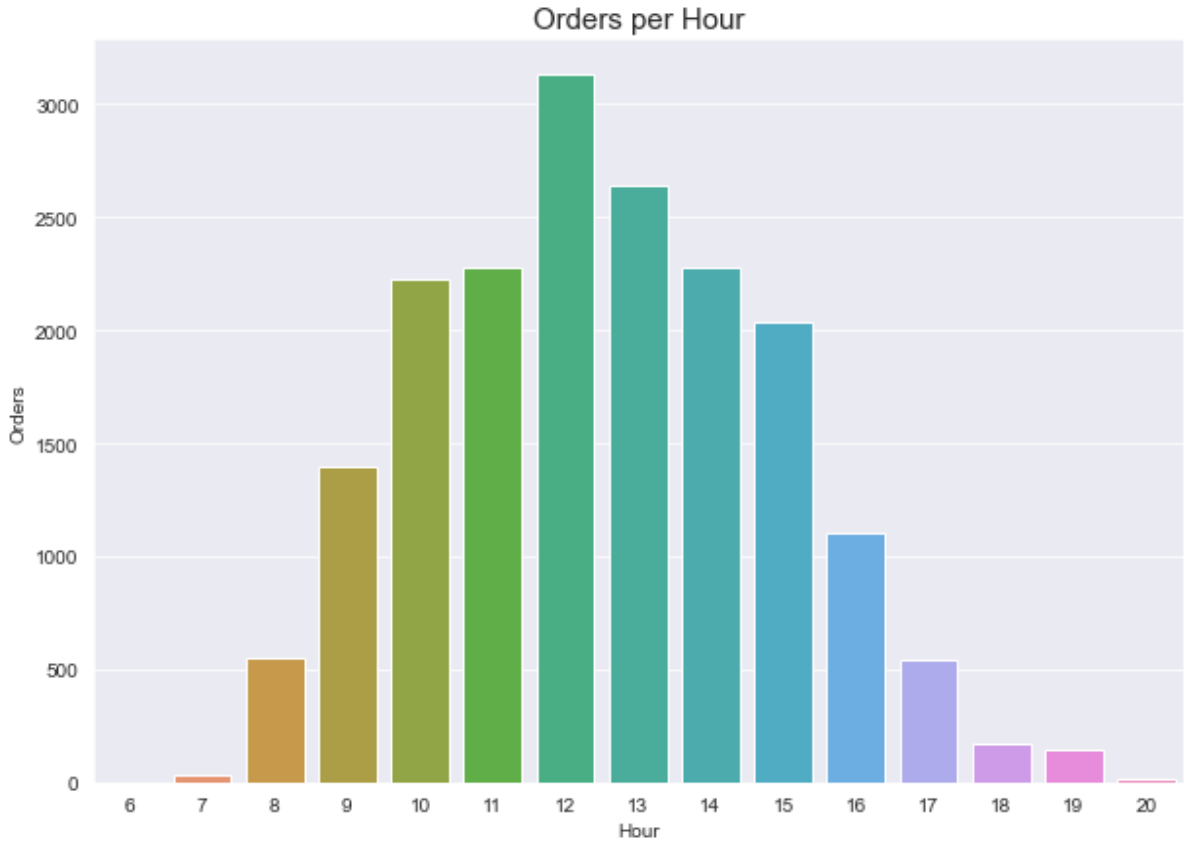
In [252]:
```
ordersper_Hour
```

Out[252]:

	Hour	InvoiceNo
0	6	1
1	7	29
2	8	555
3	9	1394
4	10	2226
5	11	2277
6	12	3130
7	13	2637
8	14	2275
9	15	2038
10	16	1100
11	17	544
12	18	169
13	19	144
14	20	18

In [253]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.barplot(x='Hour',y='InvoiceNo',data=ordersper_Hour)
ax.set_title('Orders per Hour',fontsize=15)
ax.set_ylabel('Orders',fontsize=10)
```

Out[253]: Text(0, 0.5, 'Orders')



13. How many orders for each country?

In [110]:
```
df.groupby('Country')['InvoiceNo'].count().sort_values(ascending=False)
```

Out[110]:
```
Country
United Kingdom          361878
Germany                   9495
France                    8491
EIRE                      7485
Spain                     2533
Netherlands               2371
Belgium                   2069
Switzerland               1877
Portugal                  1480
Australia                 1259
Norway                    1086
Italy                      803
Channel Islands            758
Finland                    695
Cyprus                     622
Sweden                     462
Austria                    401
Denmark                    389
Japan                      358
Poland                     341
USA                        291
Israel                     250
Unspecified                244
Singapore                  229
Iceland                    182
Canada                     151
Greece                     146
Malta                      127
United Arab Emirates        68
European Community          61
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              30
Bahrain                     17
Saudi Arabia                10
Name: InvoiceNo, dtype: int64
```

14. Orders trend across months

In [117]:
```
orders_trend=df1.groupby('Month')['InvoiceNo'].count().reset_index()
```

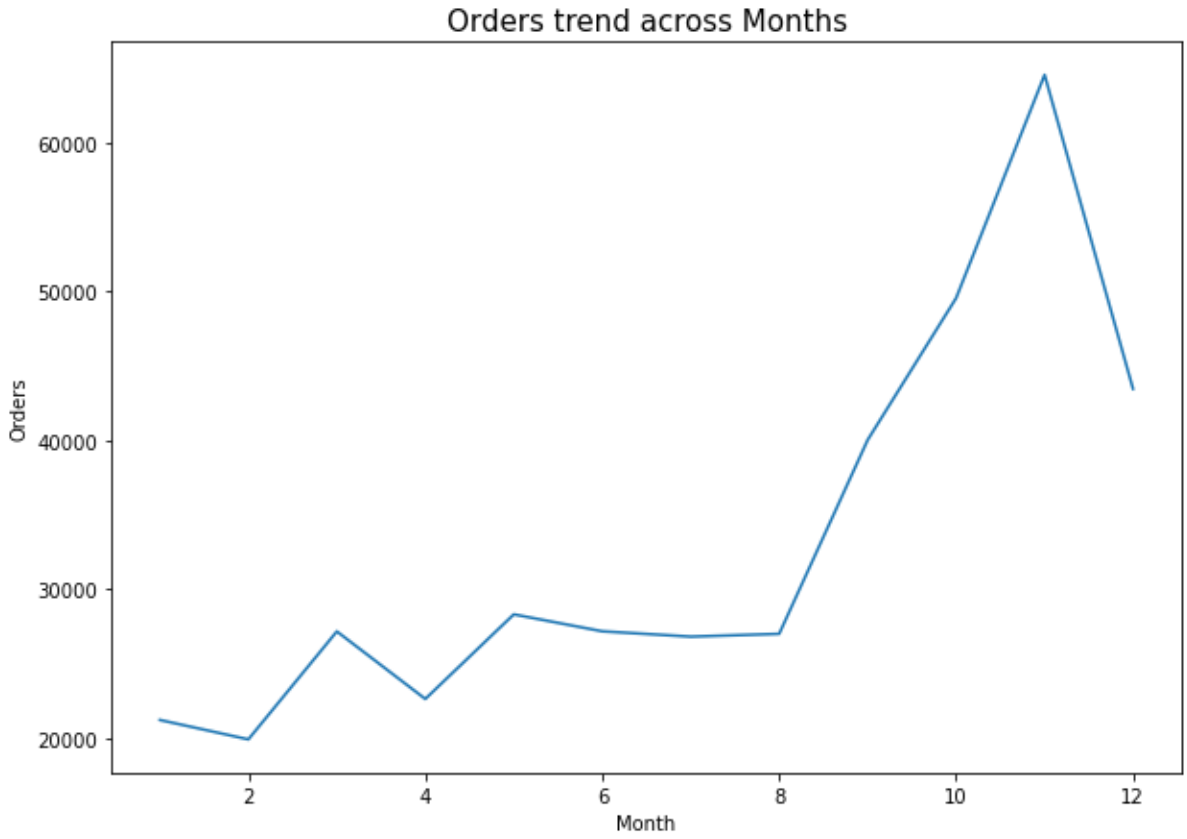
In [118]:
```
orders_trend
```

Out[118]:

	Month	InvoiceNo
0	1	21232
1	2	19928
2	3	27177
3	4	22644
4	5	28322
5	6	27185
6	7	26827
7	8	27013
8	9	40030
9	10	49557
10	11	64545
11	12	43464

In [121]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.lineplot(x ='Month',y='InvoiceNo',data=orders_trend,sort=False)
ax.set_title('Orders trend across Months',fontsize=15)
ax.set_ylabel('Orders',fontsize=10)
```

Out[121]: Text(0, 0.5, 'Orders')



15. How much money spent by each country?

In [124]:
```
money_spent_by_each_country = df1.groupby('Country')['revenue'].sum().sort_values(ascending=False)
```

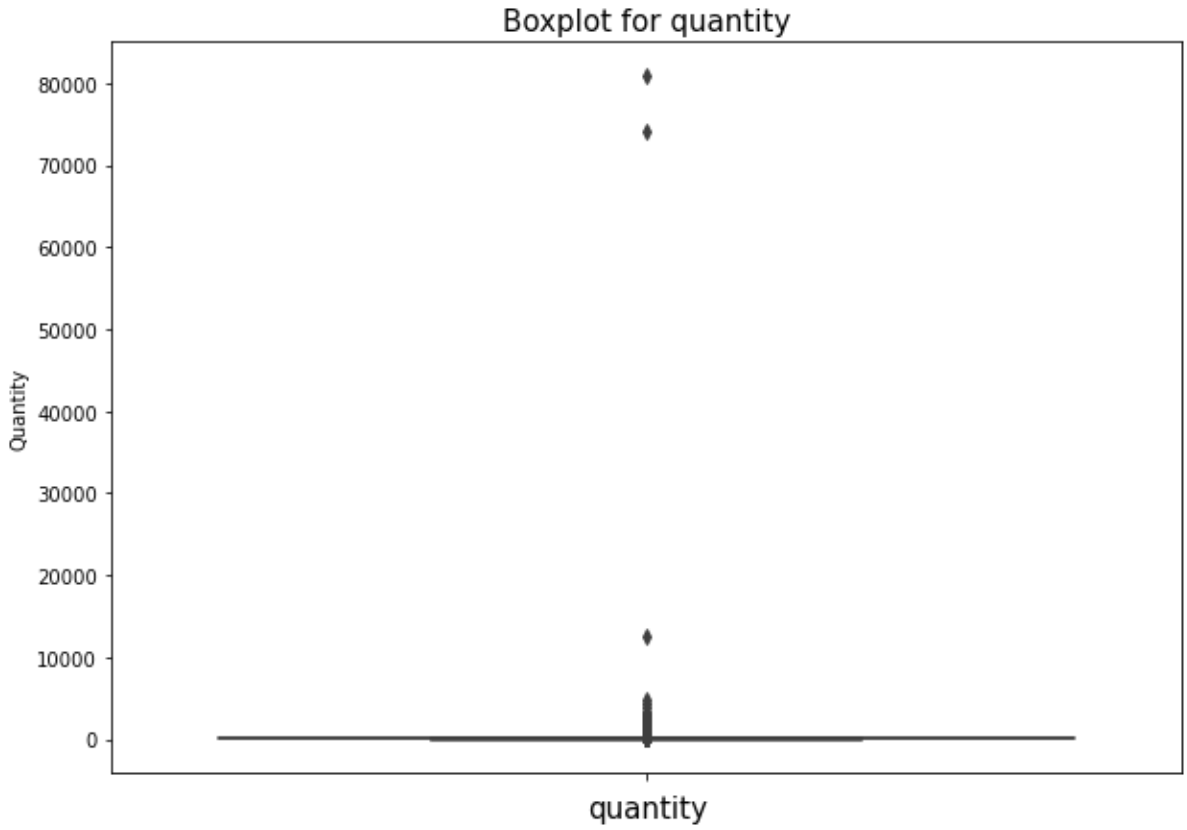
In [125]:
```
money_spent_by_each_country
```

Out[125]:
```
Country
United Kingdom          7.308392e+06
Netherlands             2.854463e+05
EIRE                    2.655459e+05
Germany                 2.288671e+05
France                  2.090240e+05
Australia               1.385213e+05
Spain                   6.157711e+04
Switzerland             5.644395e+04
Belgium                 4.119634e+04
Sweden                  3.837833e+04
Japan                   3.741637e+04
Norway                  3.616544e+04
Portugal                3.343989e+04
Finland                 2.254608e+04
Singapore               2.127929e+04
Channel Islands         2.045044e+04
Denmark                 1.895534e+04
Italy                   1.748324e+04
Cyprus                  1.359038e+04
Austria                 1.019868e+04
Poland                  7.334650e+03
Israel                  7.221690e+03
Greece                  4.760520e+03
Iceland                 4.310000e+03
Canada                  3.666380e+03
USA                     3.580390e+03
Malta                   2.725590e+03
Unspecified             2.667070e+03
United Arab Emirates    1.902280e+03
Lebanon                 1.693880e+03
Lithuania               1.661060e+03
European Community      1.300250e+03
Brazil                  1.143600e+03
RSA                     1.002310e+03
Czech Republic          8.267400e+02
Bahrain                 5.484000e+02
Saudi Arabia            1.459200e+02
Name: revenue, dtype: float64
```

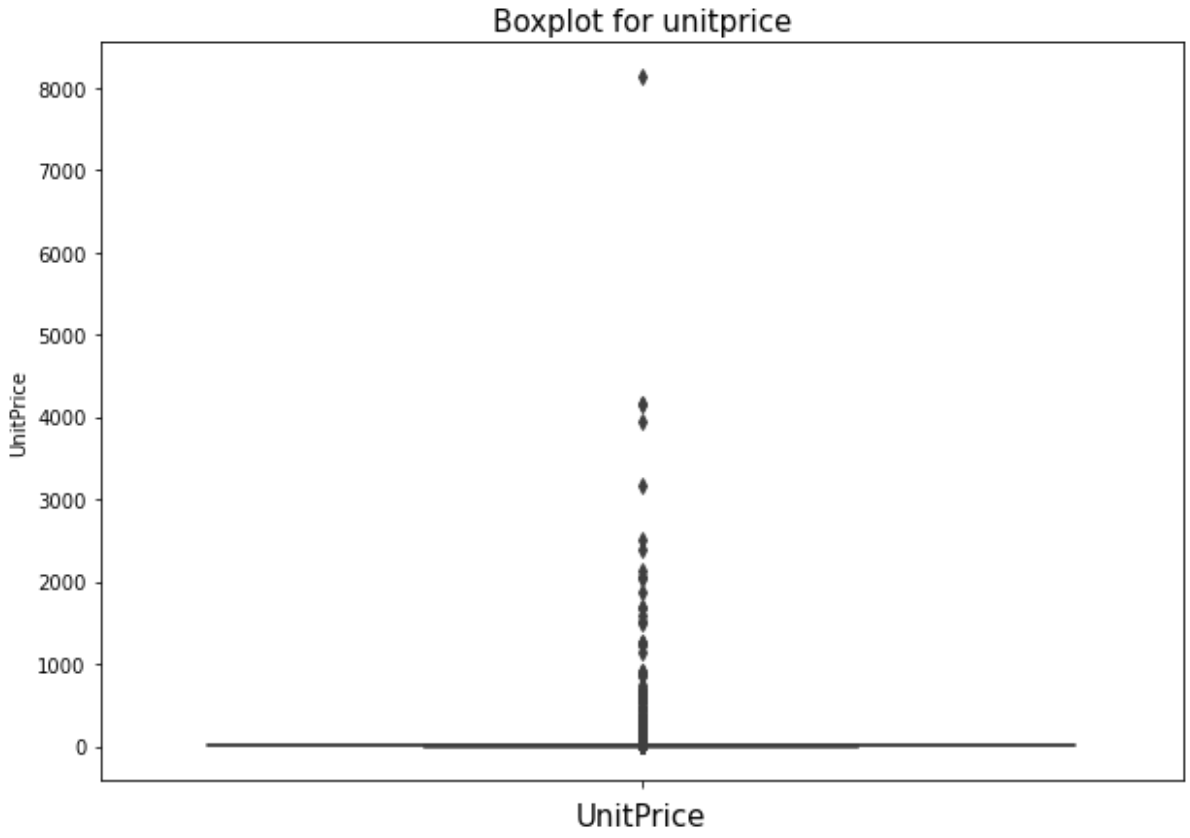
1. Perform Basic EDA

a. Boxplot – All Numeric Variables

In [131]:
```
fig,axes = plt.subplots(figsize=(10,7))
sns.boxplot('Quantity',data=df1,orient='v')
plt.xlabel('quantity',fontsize=15)
plt.title('Boxplot for quantity',fontsize=15)
plt.show()
```

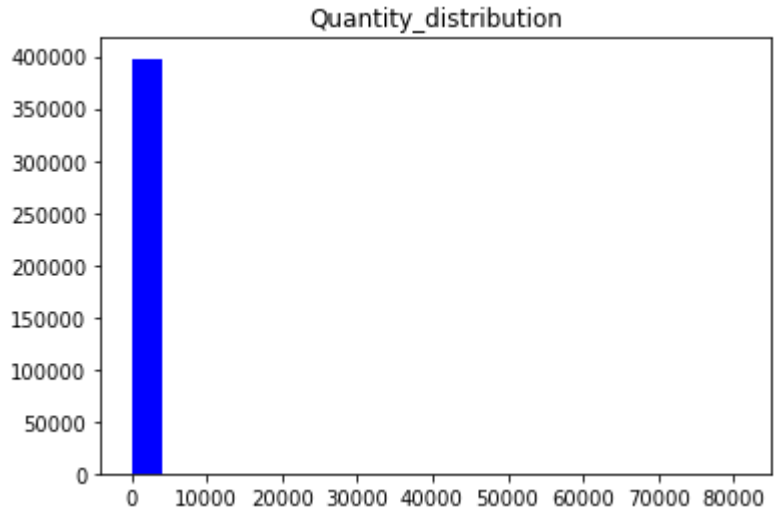


In [132]:
```
fig,axes = plt.subplots(figsize=(10,7))
sns.boxplot('UnitPrice',data=df1,orient='v')
plt.xlabel('UnitPrice',fontsize=15)
plt.title('Boxplot for unitprice',fontsize=15)
plt.show()
```

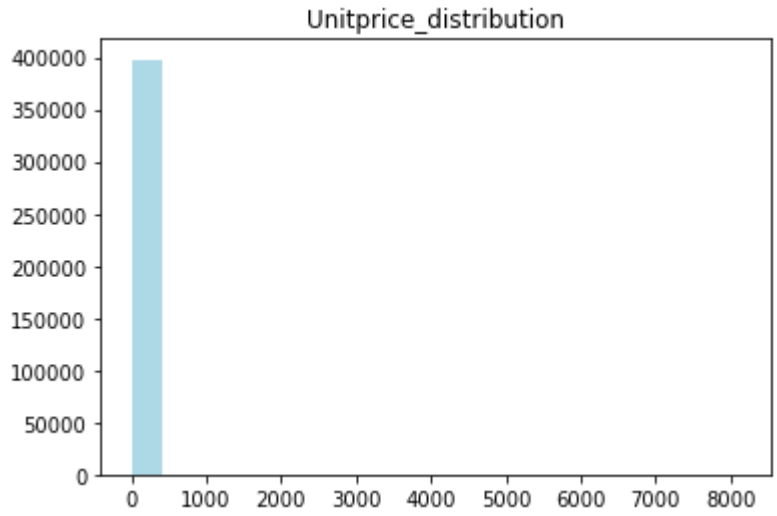


b. Histogram – All Numeric Variables

In [136]:
```
plt.hist(df1['Quantity'], bins= 20, color='blue', alpha=1)
plt.title('Quantity_distribution')
plt.show()
```



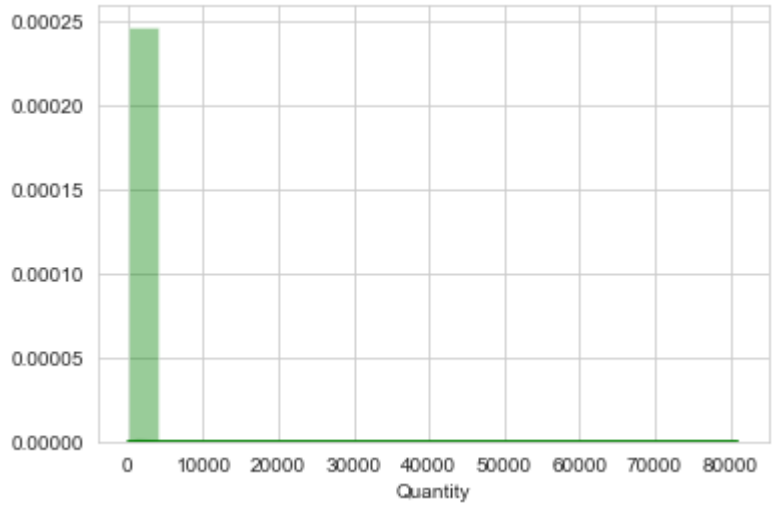
In [139]:
```
plt.hist(df1['UnitPrice'], bins= 20, color='lightblue', alpha=1)
plt.title('Unitprice_distribution')
plt.show()
```



c. Distribution Plot – All Numeric Variables

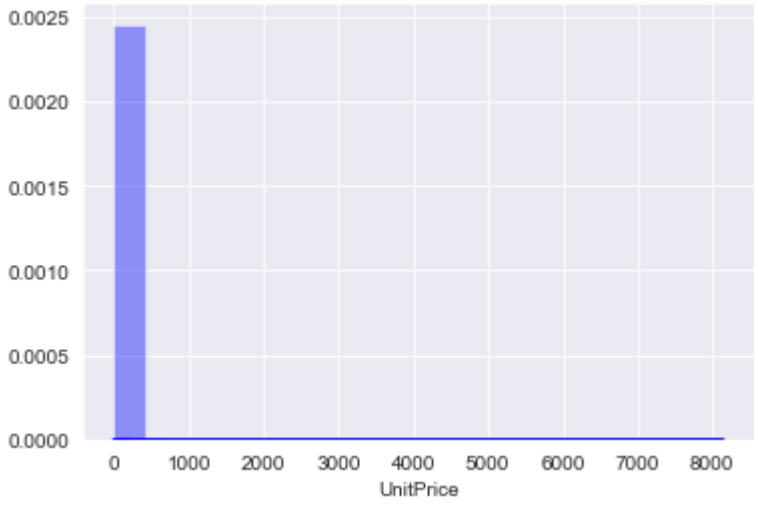
In [144]:
```
sns.set_style('whitegrid')
sns.distplot(df1['Quantity'] , color ='green', bins = 20)
```

Out[144]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25f3e9d68b0>
```



In [147]:
```
sns.set_style('darkgrid')
sns.distplot(df1['UnitPrice'] , color ='blue', bins = 20)
```

Out[147]: <matplotlib.axes._subplots.AxesSubplot at 0x25f3eb72490>



d. Aggregation for all numerical Columns

In [151]:
```
df1.aggregate({"Quantity":['sum', 'min','max','mean'],
               "UnitPrice":['min', 'sum','max','mean'],
               "revenue":['sum','max','min','mean']})
```

Out[151]:

	Quantity	UnitPrice	revenue
max	8.099500e+04	8.142750e+03	1.684696e+05
mean	1.302182e+01	3.116174e+00	2.239475e+01
min	1.000000e+00	0.000000e+00	0.000000e+00
sum	5.181696e+06	1.240001e+06	8.911408e+06

e. Unique Values across all columns

In [34]:
```

```

In [166]:
```
df1.nunique()
```

Out[166]:
```
InvoiceNo      18536
StockCode       3665
Description     3877
Quantity         302
InvoiceDate    17286
UnitPrice        441
CustomerID      4339
Hour              15
Country           37
date             305
hour             738
Year               2
Month             12
Day                6
revenue         2940
dtype: int64
```

f. Duplicate values across all columns

In [172]:
```
df1.duplicated(subset=None,keep='first').value_counts()
```

Out[172]:
```
False    392732
True       5192
dtype: int64
```

g. Correlation – Heatmap - All Numeric Variables

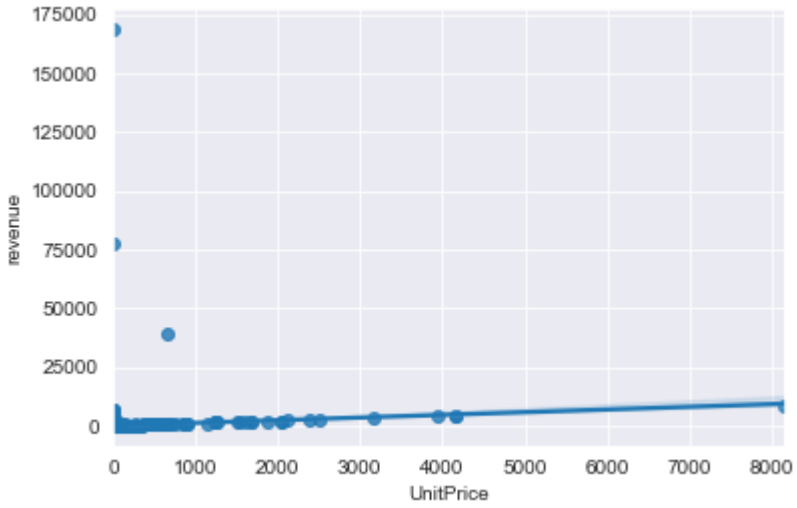
In [174]:
```
sns.heatmap(df1[['Quantity','UnitPrice','revenue']].corr(),xticklabels='auto',
yticklabels='auto',annot=True,fmt='.2g')
```

Out[174]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25f3e5f0880>
```

h. Regression Plot - All Numeric Variables

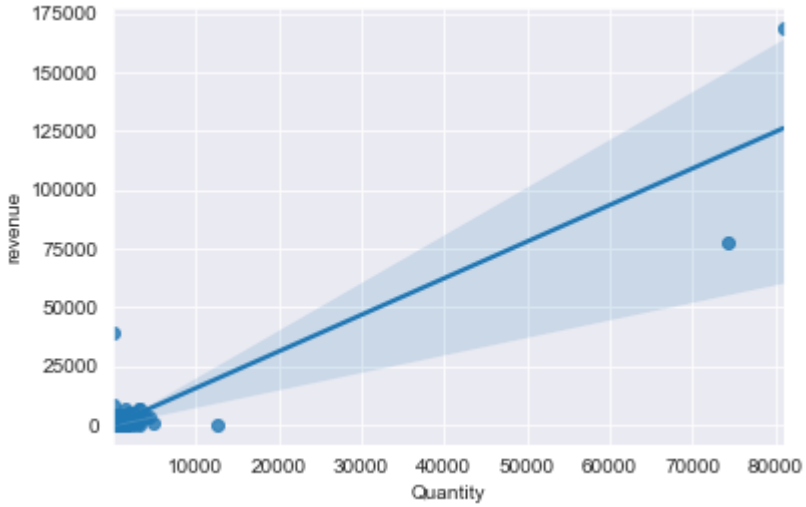
In [176]:
```
sns.regplot(x='UnitPrice',y='revenue',data=df1)
```

Out[176]: <matplotlib.axes._subplots.AxesSubplot at 0x25f3e3c85b0>



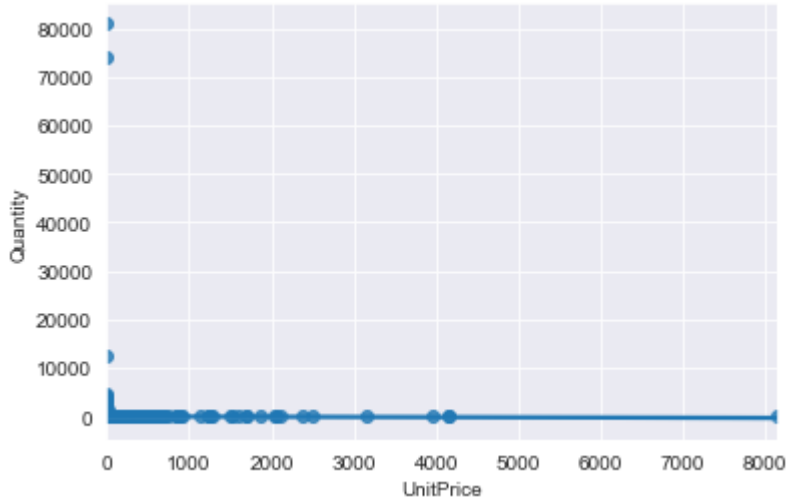
In [177]:
```
sns.regplot(x='Quantity',y='revenue',data=df1)
```

Out[177]: <matplotlib.axes._subplots.AxesSubplot at 0x25f3de568e0>



In [178]:
```
sns.regplot(x='UnitPrice',y='Quantity',data=df1)
```

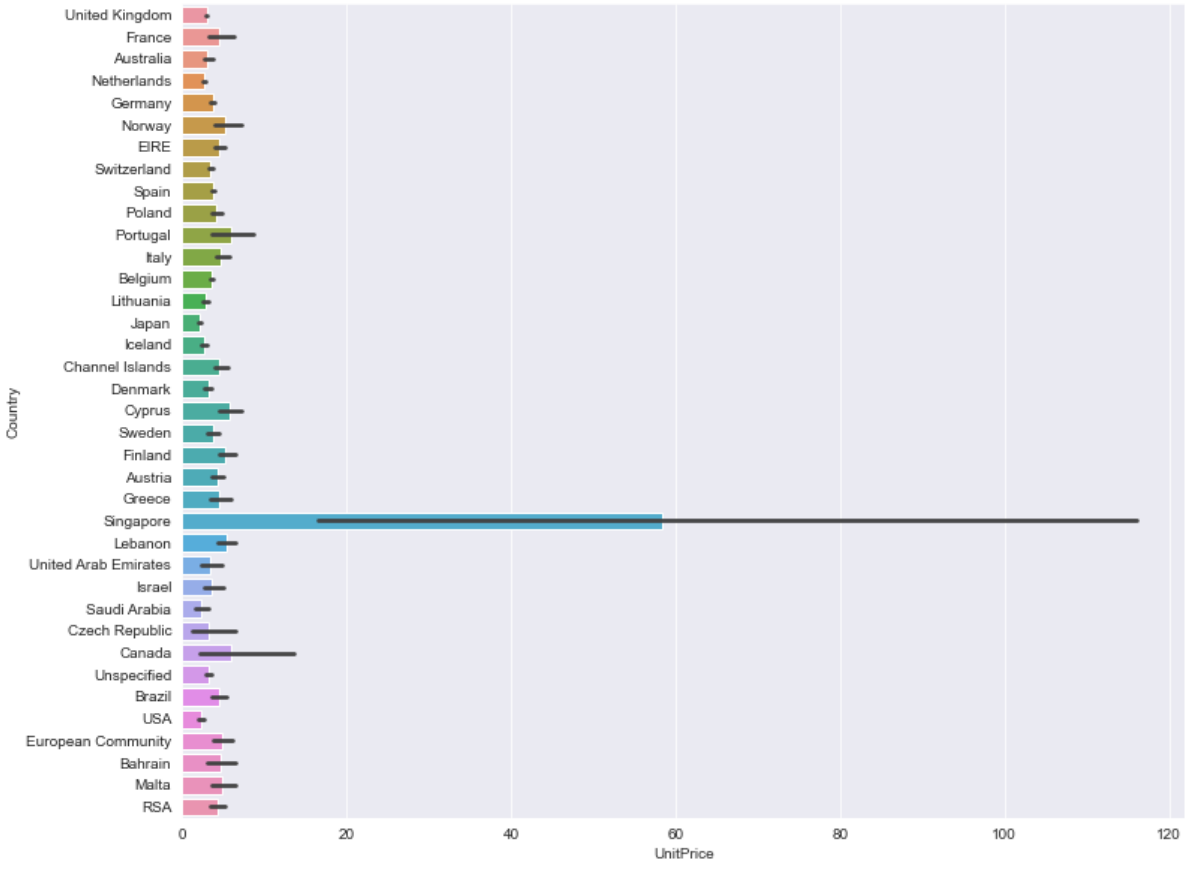
Out[178]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25f3de373d0>
```



i. Bar Plot – Every Categorical Variable vs every Numerical Variable

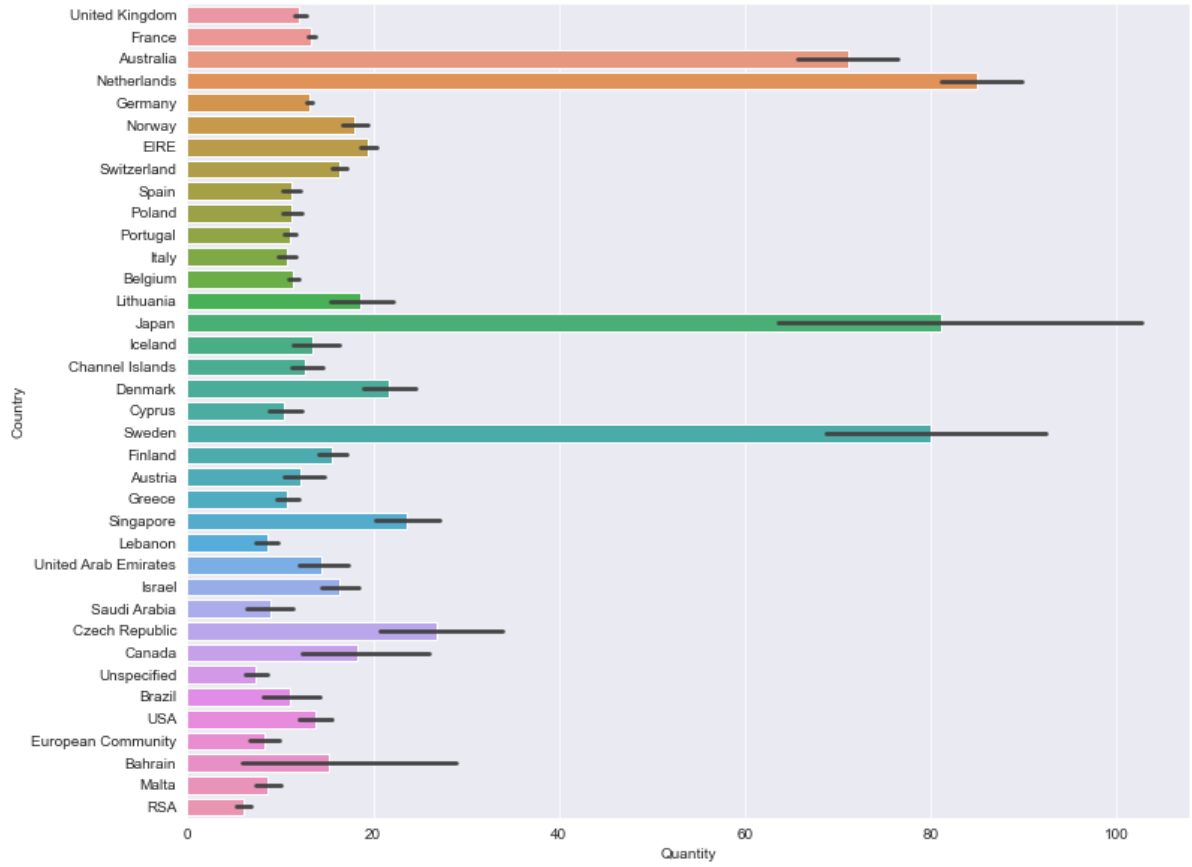
In [180]:
```
plt.figure(figsize=(12,10))
sns.barplot(x ='UnitPrice', y ='Country', data = df1)
```

Out[180]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25f3ea66730>
```



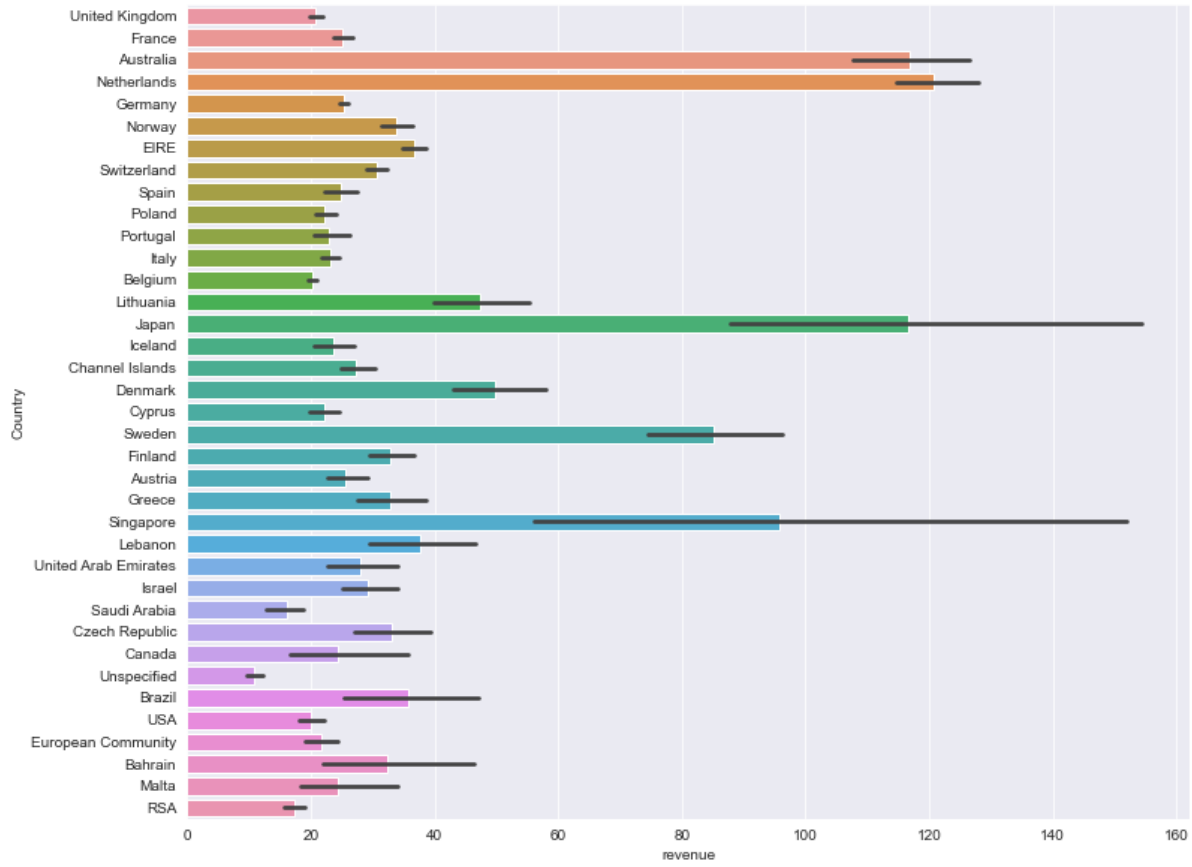
In [181]:
```
plt.figure(figsize=(12,10))
sns.barplot(x ='Quantity', y ='Country', data = df1)
```

Out[181]: <matplotlib.axes._subplots.AxesSubplot at 0x25f3bc1fb80>



In [182]:
```
plt.figure(figsize=(12,10))
sns.barplot(x ='revenue', y ='Country', data = df1)
```

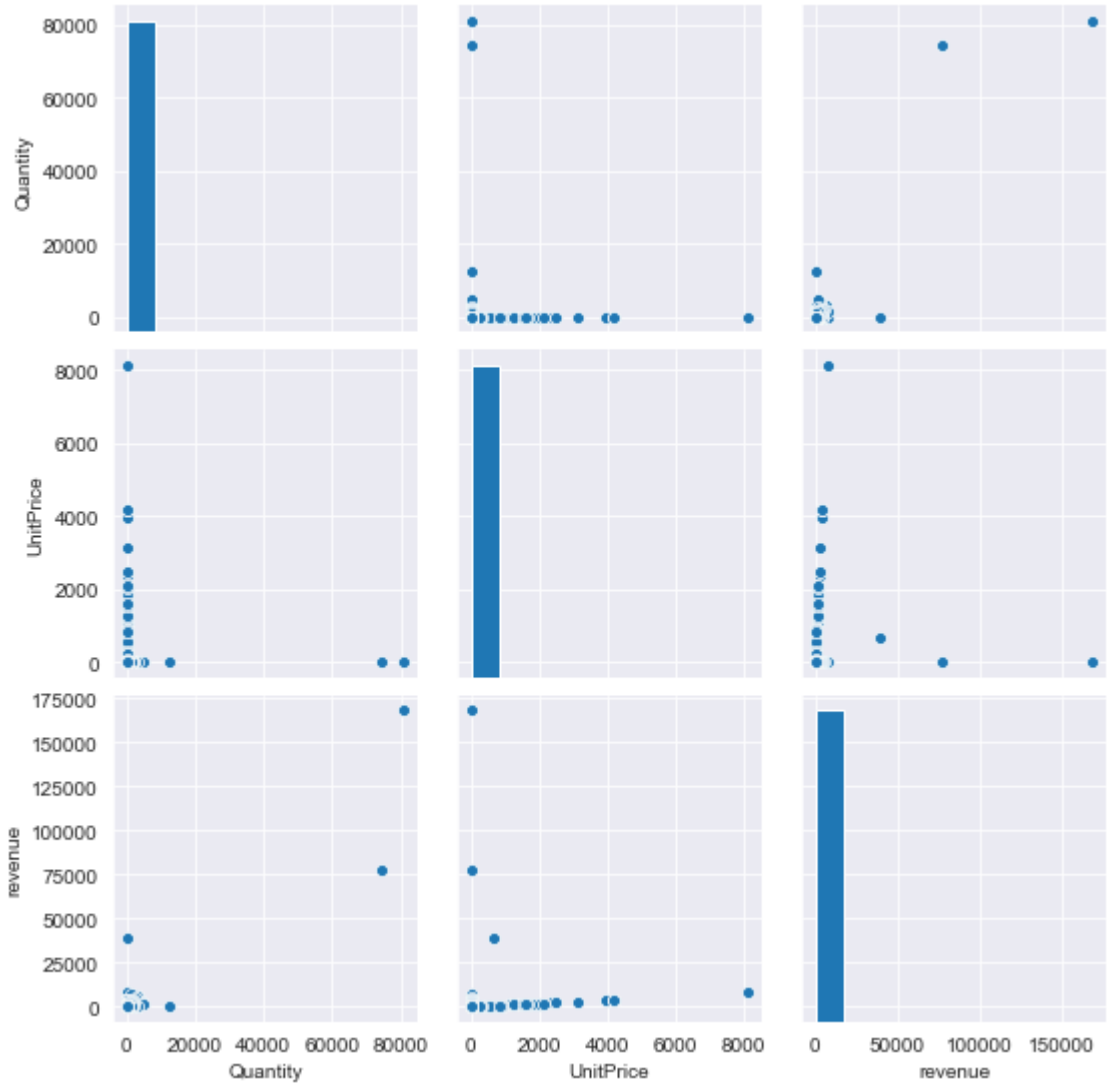
Out[182]: <matplotlib.axes._subplots.AxesSubplot at 0x25f3c8db580>



j. Pair plot - All Numeric Variables

In [183]:
```
sns.pairplot(data=df1[['Quantity','UnitPrice','revenue']])
```

Out[183]: <seaborn.axisgrid.PairGrid at 0x25f3da7c280>



k. Line chart to show the trend of data - All Numeric/Date Variables

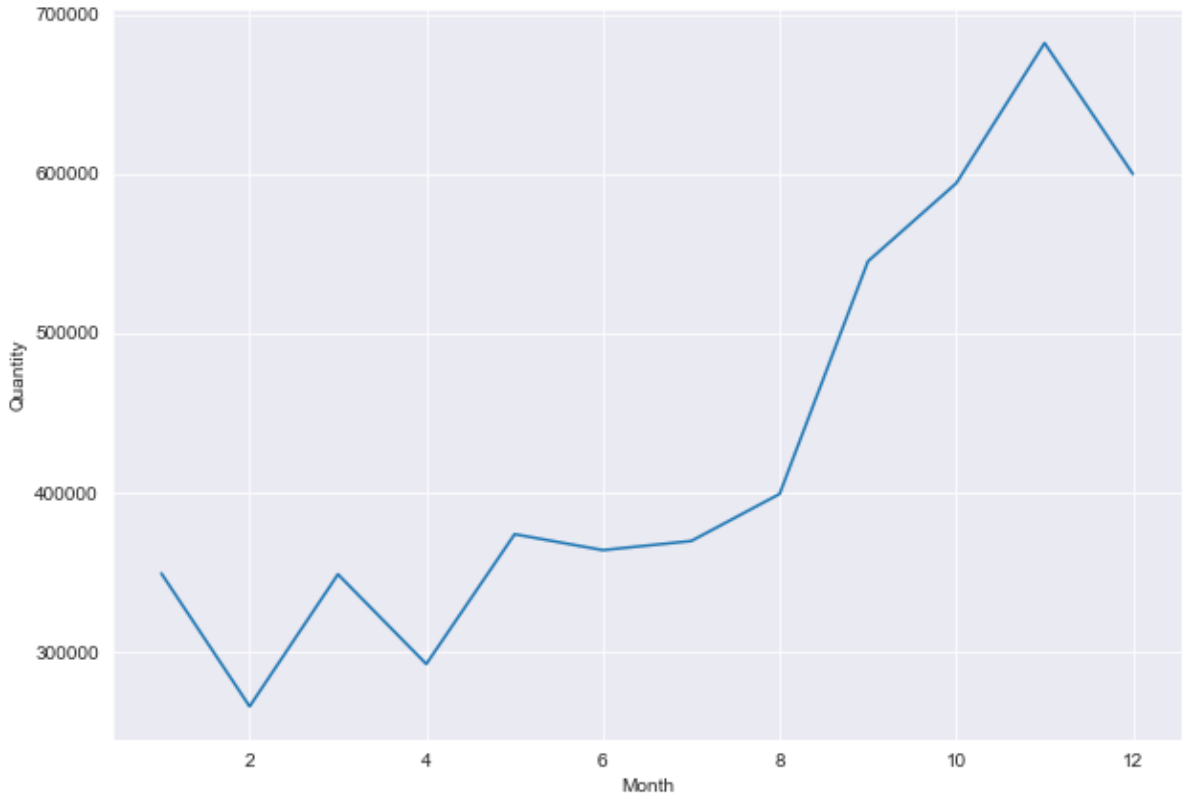
In [193]:
```
monthly_quantities = df1.groupby('Month')['Quantity'].sum().reset_index()
```

In [194]:
```
monthly_quantities
```

Out[194]:

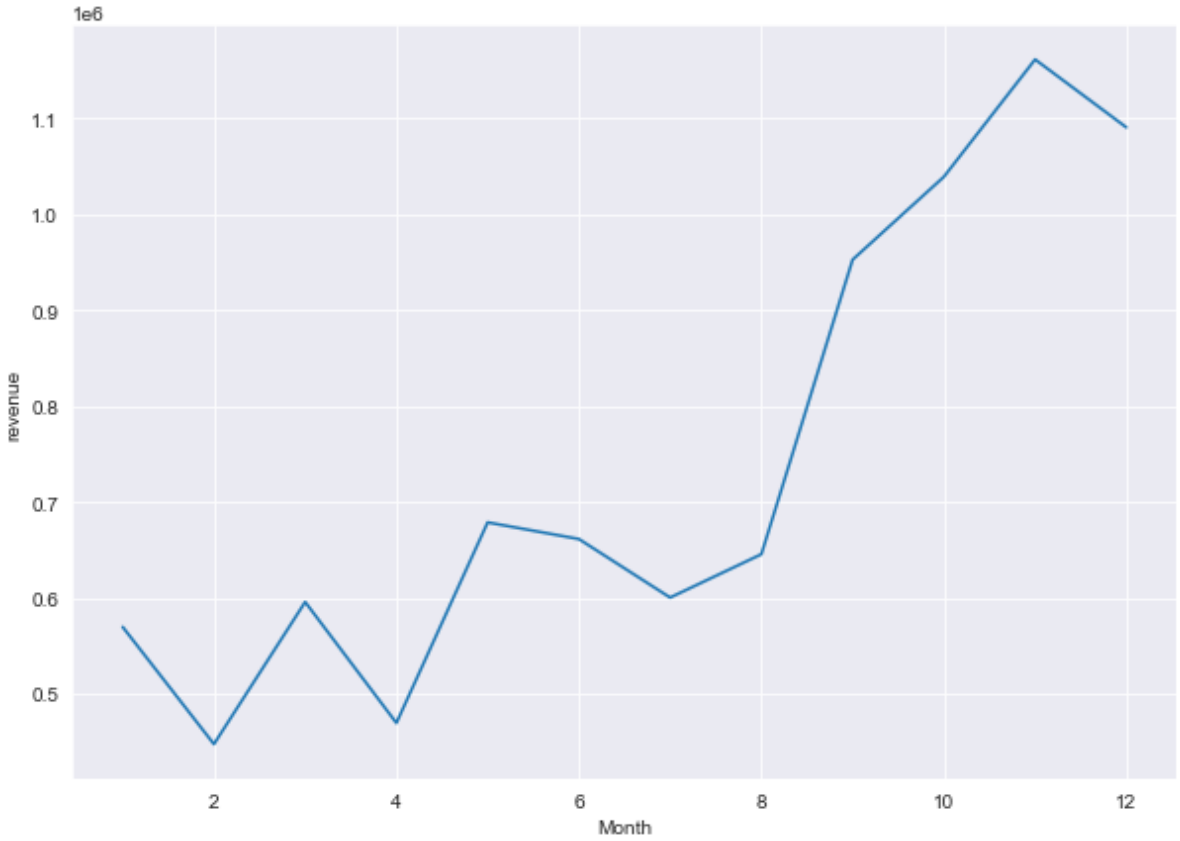
	Month	Quantity
0	1	349147
1	2	265638
2	3	348544
3	4	292225
4	5	373685
5	6	363699
6	7	369432
7	8	398938
8	9	544899
9	10	593908
10	11	681888
11	12	599693

In [196]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.lineplot(x='Month',y='Quantity',data=monthly_quantities,sort=False)
```



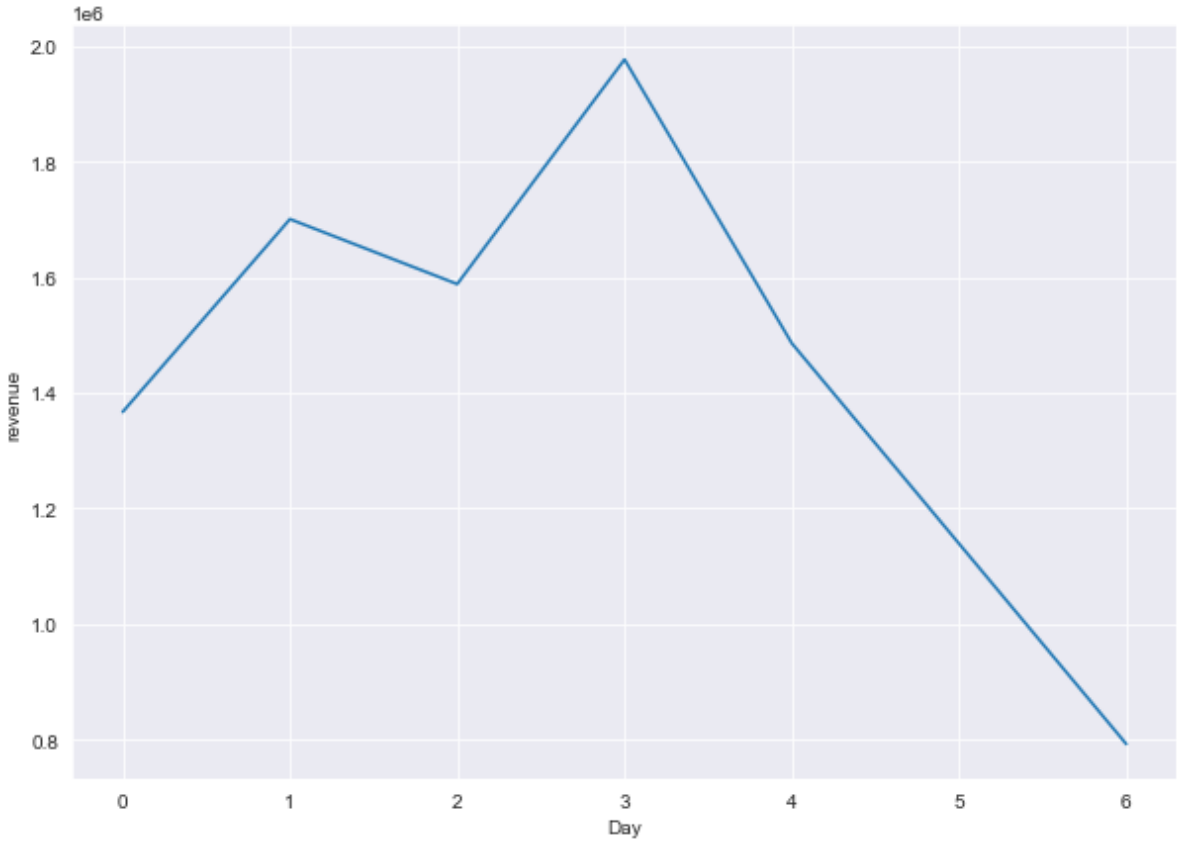
In [197]:
```
monthly_revenue = df1.groupby('Month')['revenue'].sum().reset_index()
```

In [199]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.lineplot(x ='Month',y='revenue',data=monthly_revenue,sort=False)
```



In [200]:
```
weekly_sales = df1.groupby('Day')['revenue'].sum().reset_index()
```

In [201]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.lineplot(x='Day',y='revenue',data=weekly_sales,sort=False)
```



I. Plot the skewness - All Numeric Variables

In [203]:
```
df1[['Quantity','UnitPrice','revenue']].skew(axis = 0, skipna = True)
```

Out[203]:
```
Quantity     403.319431
UnitPrice    204.042413
revenue      451.465538
dtype: float64
```

In [205]:
```
df1.skew(axis=0,skipna=True)
```

Out[205]:
```
InvoiceNo      -0.178563
Quantity      403.319431
UnitPrice     204.042413
CustomerID      0.025776
Hour            0.189037
Year           -3.504515
Month          -0.444842
Day             0.396235
revenue       451.465538
dtype: float64
```

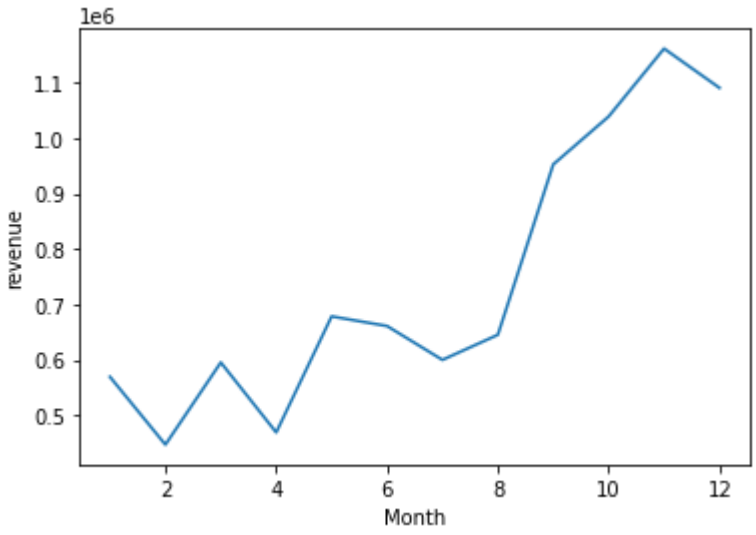
In [39]:
```
#revenue by month

data_revenue = df1.groupby('Month')['revenue'].sum().reset_index()
```

In [40]:
```
data_revenue
```

Out[40]:

	Month	revenue
0	1	569445.040
1	2	447137.350
2	3	595500.760
3	4	469200.361
4	5	678594.560
5	6	661213.690
6	7	600091.011
7	8	645343.900
8	9	952838.382
9	10	1039318.790
10	11	1161817.380
11	12	1090906.680

In [41]:
```
ax = sns.lineplot(x='Month',y='revenue',data=data_revenue,sort=False)
```



In [42]:
```
##monthly growth rate
data_revenue['MonthlyGrowth'] = data_revenue['revenue'].pct_change()
```

In [43]:
```
data_revenue
```

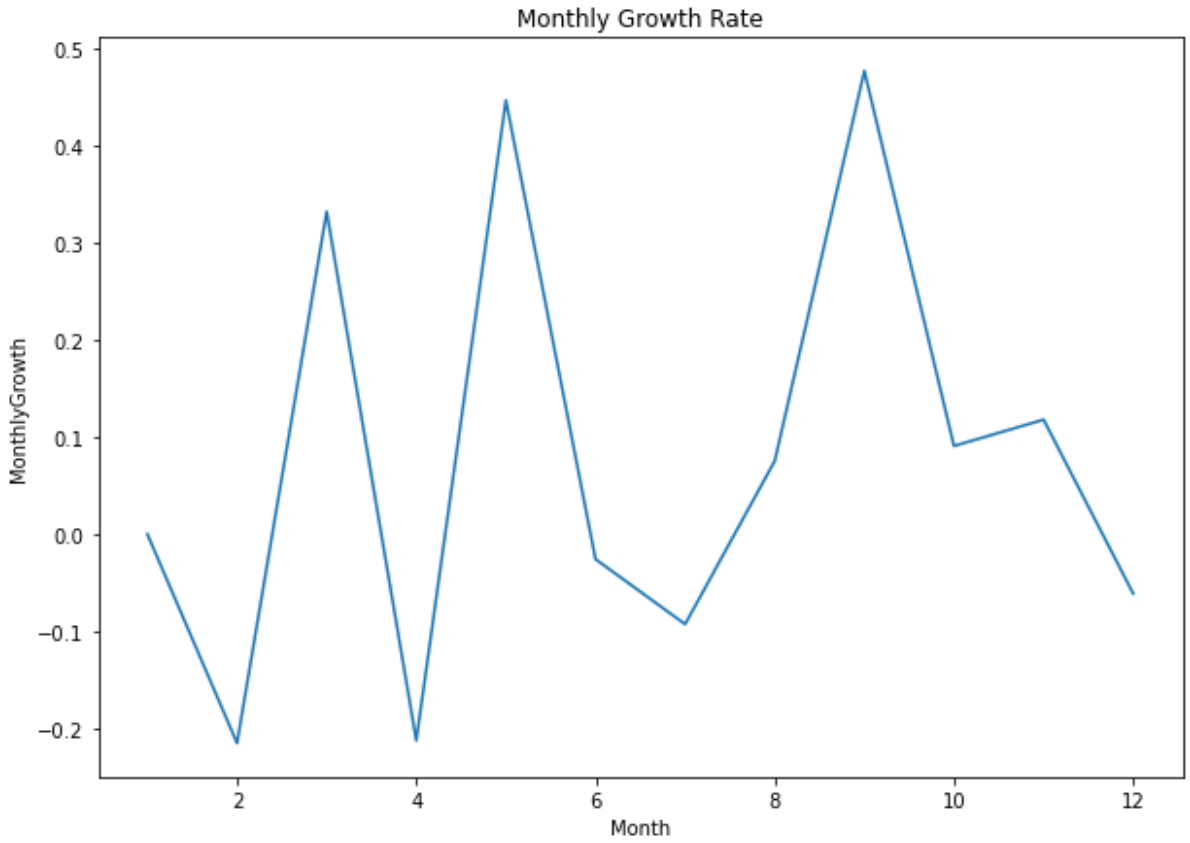
Out[43]:

	Month	revenue	MonthlyGrowth
0	1	569445.040	NaN
1	2	447137.350	-0.214784
2	3	595500.760	0.331807
3	4	469200.361	-0.212091
4	5	678594.560	0.446279
5	6	661213.690	-0.025613
6	7	600091.011	-0.092440
7	8	645343.900	0.075410
8	9	952838.382	0.476482
9	10	1039318.790	0.090761
10	11	1161817.380	0.117864
11	12	1090906.680	-0.061034

In [44]:
```
data_revenue['MonthlyGrowth'] = data_revenue['MonthlyGrowth'].fillna(0)
```

In [46]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.lineplot(x='Month',y='MonthlyGrowth',data=data_revenue,sort=False)
ax.set_title('Monthly Growth Rate')
```

Out[46]: Text(0.5, 1.0, 'Monthly Growth Rate')

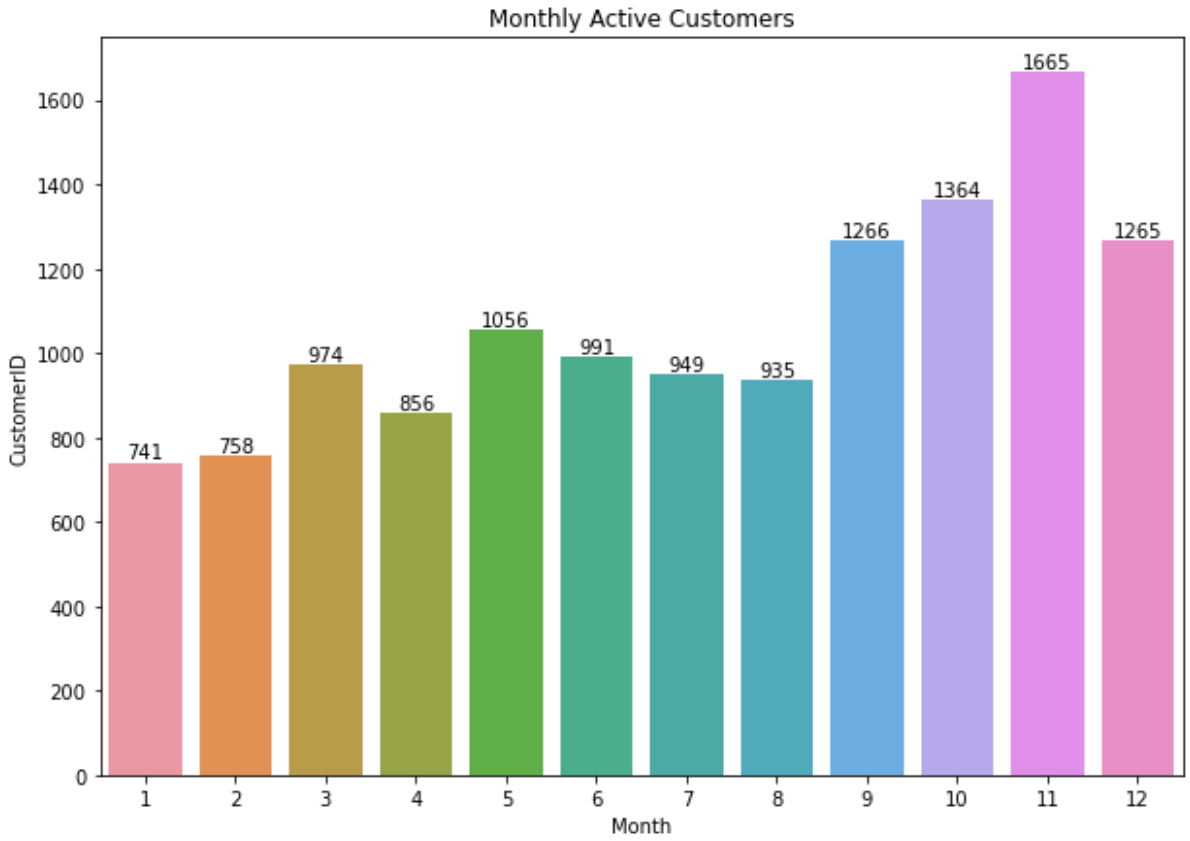


In [49]:
```
##monthly active customers
monthly_active = df1.groupby('Month')['CustomerID'].nunique().reset_index()
monthly_active
```

Out[49]:

	Month	CustomerID
0	1	741
1	2	758
2	3	974
3	4	856
4	5	1056
5	6	991
6	7	949
7	8	935
8	9	1266
9	10	1364
10	11	1665
11	12	1265

In [63]:
```
fig,axes = plt.subplots(figsize=(10,7))
ax = sns.barplot(x='Month',y='CustomerID',data=monthly_active)
ax.set_title('Monthly Active Customers')

for p in ax.patches:
    height = p.get_height()
    ax.text(x = p.get_x()+(p.get_width()/2),y = height+10,ha='center',s = '{:.
0f}'.format(height))
```



In [64]:
```
cust_minpur_date = df1.groupby('CustomerID').date.min().reset_index()
```

In [65]:
```
cust_minpur_date
```

Out[65]:

	CustomerID	date
0	12346.0	2011-01-18
1	12347.0	2010-12-07
2	12348.0	2010-12-16
3	12349.0	2011-11-21
4	12350.0	2011-02-02
...	...	...
4334	18280.0	2011-03-07
4335	18281.0	2011-06-12
4336	18282.0	2011-08-05
4337	18283.0	2011-01-06
4338	18287.0	2011-05-22

4339 rows × 2 columns

In [66]:
```
cust_minpur_date.columns=['CustomerID','Minpurchasedate']
```

In [67]:
```
cust_minpur_date['Minpurchasemonth'] = cust_minpur_date['Minpurchasedate'].map
(lambda date:date.month)
```

In [68]:
```
cust_minpur_date
```

Out[68]:

	CustomerID	Minpurchasedate	Minpurchasemonth
0	12346.0	2011-01-18	1
1	12347.0	2010-12-07	12
2	12348.0	2010-12-16	12
3	12349.0	2011-11-21	11
4	12350.0	2011-02-02	2
...	...	...	...
4334	18280.0	2011-03-07	3
4335	18281.0	2011-06-12	6
4336	18282.0	2011-08-05	8
4337	18283.0	2011-01-06	1
4338	18287.0	2011-05-22	5

4339 rows × 3 columns

In [69]:
```
df1 = pd.merge(df1,cust_minpur_date, on ='CustomerID')
```

In [70]:
```
df1
```

Out[70]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Coun
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Unit Kingd
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Unit Kingd
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Unit Kingd
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Unit Kingd
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Unit Kingd
...	...	...	...	...	...	...	...	
397919	581578	22993	SET OF 4 PANTRY JELLY MOULDS	12	2011-12-09 12:16:00	1.25	12713.0	Germa
397920	581578	22907	PACK OF 20 NAPKINS PANTRY DESIGN	12	2011-12-09 12:16:00	0.85	12713.0	Germa
397921	581578	22908	PACK OF 20 NAPKINS RED APPLES	12	2011-12-09 12:16:00	0.85	12713.0	Germa
397922	581578	23215	JINGLE BELL HEART ANTIQUE SILVER	12	2011-12-09 12:16:00	2.08	12713.0	Germa
397923	581578	22736	RIBBON REEL MAKING SNOWMEN	10	2011-12-09 12:16:00	1.65	12713.0	Germa

397924 rows × 17 columns

In [72]:
```
df1['usertype']='New'
df1.loc[df1['Month']>df1['Minpurchasemonth'],'usertype' ]= 'Existing'
```

In [73]:
```
df1
```

Out[73]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Coun
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Unit Kingd
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Unit Kingd
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Unit Kingd
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Unit Kingd
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Unit Kingd
...	...	...	...	...	...	...	...	
397919	581578	22993	SET OF 4 PANTRY JELLY MOULDS	12	2011-12-09 12:16:00	1.25	12713.0	Germa
397920	581578	22907	PACK OF 20 NAPKINS PANTRY DESIGN	12	2011-12-09 12:16:00	0.85	12713.0	Germa
397921	581578	22908	PACK OF 20 NAPKINS RED APPLES	12	2011-12-09 12:16:00	0.85	12713.0	Germa
397922	581578	23215	JINGLE BELL HEART ANTIQUE SILVER	12	2011-12-09 12:16:00	2.08	12713.0	Germa
397923	581578	22736	RIBBON REEL MAKING SNOWMEN	10	2011-12-09 12:16:00	1.65	12713.0	Germa

397924 rows × 18 columns

In [75]:
```
usertype_revenue = df1.groupby(['Month','usertype'])['revenue'].sum().reset_index()
```

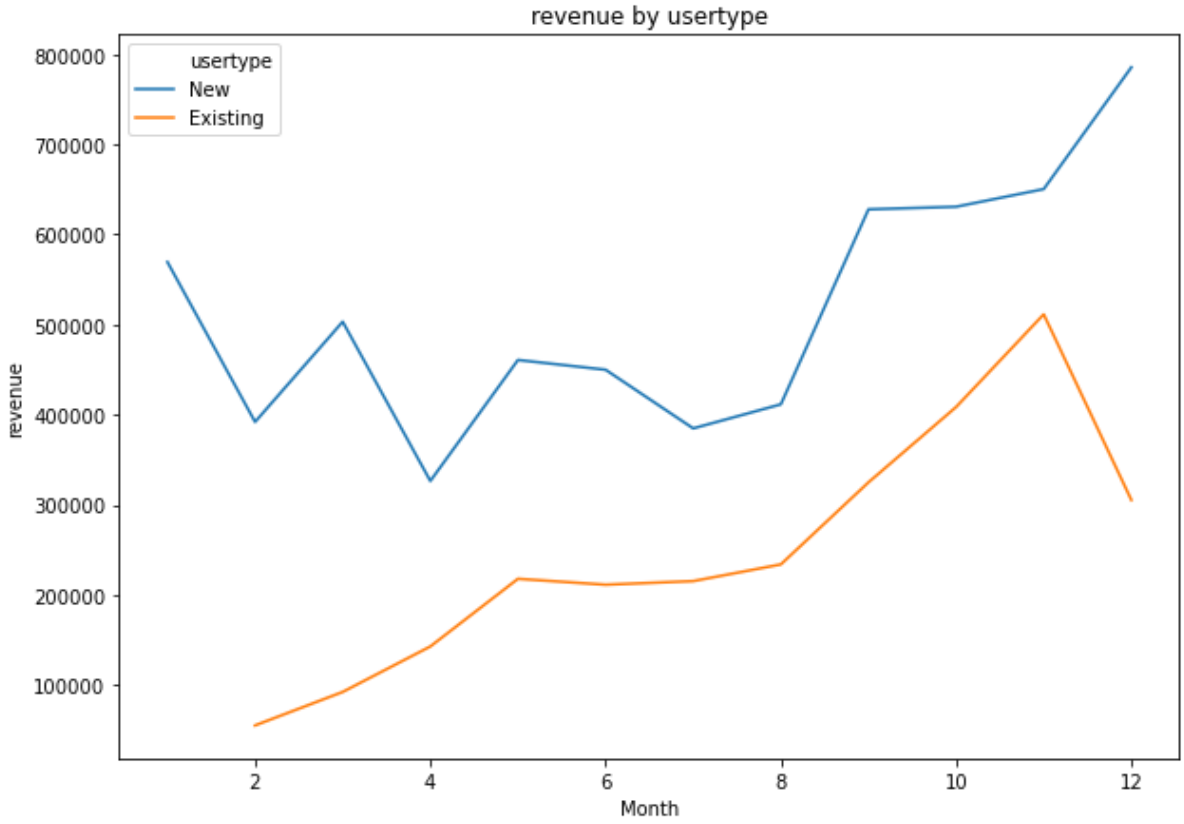
In [76]:
```
usertype_revenue
```

Out[76]:

	Month	usertype	revenue
0	1	New	569445.040
1	2	Existing	55149.910
2	2	New	391987.440
3	3	Existing	92311.410
4	3	New	503189.350
5	4	Existing	142781.210
6	4	New	326419.151
7	5	Existing	217863.280
8	5	New	460731.280
9	6	Existing	211308.130
10	6	New	449905.560
11	7	Existing	215298.340
12	7	New	384792.671
13	8	Existing	233838.580
14	8	New	411505.320
15	9	Existing	324920.841
16	9	New	627917.541
17	10	Existing	408516.420
18	10	New	630802.370
19	11	Existing	511421.670
20	11	New	650395.710
21	12	Existing	305330.200
22	12	New	785576.480

In [78]:
```
fig,axes = plt.subplots(figsize=(10,7))

ax = sns.lineplot(data=usertype_revenue,x='Month',y='revenue', hue='usertype')
ax.set_title('revenue by usertype')
```

Out[78]: Text(0.5, 1.0, 'revenue by usertype')



In [2]:
```
pip install xelatex
```

```
Note: you may need to restart the kernel to use updated packages.

ERROR: Could not find a version that satisfies the requirement xelatex (from
versions: none)
ERROR: No matching distribution found for xelatex
```