In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv("Ecommerce - UK Retailer.csv")
df.head()
```

Out[2]:

	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 [4]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[4]:
```
CustomerID     135080
Description      1454
InvoiceNo           0
StockCode           0
Quantity            0
InvoiceDate         0
UnitPrice           0
Country             0
dtype: int64
```

We see that there are some missing values for Customers ID and Description. The rows with any of these missing values will therefore be removed.

In [5]:
```
# check out the rows with missing values
df[df.isnull().any(axis=1)].head()
```

Out[5]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	12/1/2010 11:52	0.00	NaN	United Kingdom
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	12/1/2010 14:32	2.51	NaN	United Kingdom
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	12/1/2010 14:32	2.51	NaN	United Kingdom
1445	536544	21786	POLKADOT RAIN HAT	4	12/1/2010 14:32	0.85	NaN	United Kingdom
1446	536544	21787	RAIN PONCHO RETROSPOT	2	12/1/2010 14:32	1.66	NaN	United Kingdom

In [8]:
```
# change the invoice_date format - String to Timestamp format
df['InvoiceDate'] = pd.to_datetime(df.InvoiceDate, format='%m/%d/%Y %H:%M')
```

In [9]:
```
df['Description'] = df.Description.str.lower()
```

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
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Remove rows with missing values

In [11]:
```
# df_new without missing values
df_new = df.dropna()
```

In [12]:
```
# check missing values for each column
df_new.isnull().sum().sort_values(ascending=False)
```

Out[12]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

In [13]:
```
# change columns tyoe - String to Int type
df_new['CustomerID'] = df_new['CustomerID'].astype('int64')
df_new.head()
```

```
<ipython-input-13-c71e6e04ec74>: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)
  df_new['CustomerID'] = df_new['CustomerID'].astype('int64')
```

Out[13]:

	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	United Kingdom
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850	United Kingdom
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850	United Kingdom
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom

In [14]:
```
df_new.describe().round(2)
```

Out[14]:

	Quantity	UnitPrice	CustomerID
count	406829.00	406829.00	406829.00
mean	12.06	3.46	15287.69
std	248.69	69.32	1713.60
min	-80995.00	0.00	12346.00
25%	2.00	1.25	13953.00
50%	5.00	1.95	15152.00
75%	12.00	3.75	16791.00
max	80995.00	38970.00	18287.00

Remove Quantity with negative values

In [16]:
```
df_new = df_new[df_new.Quantity > 0]
df_new.describe().round(2)
```

Out[16]:

	Quantity	UnitPrice	CustomerID
count	397924.00	397924.00	397924.00
mean	13.02	3.12	15294.32
std	180.42	22.10	1713.17
min	1.00	0.00	12346.00
25%	2.00	1.25	13969.00
50%	6.00	1.95	15159.00
75%	12.00	3.75	16795.00
max	80995.00	8142.75	18287.00

Add the column - amount_spent

In [18]:
```
_new.loc['amount_spent'] = df_new['Quantity'] * df_new['UnitPrice']
rearrange all the columns for easy reference
_new = df_new[['InvoiceNo','InvoiceDate','StockCode','Description','Quantity','Un
```

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

In [19]:
```
df_new.insert(loc=2, column='year_month', value=df_new['InvoiceDate'].map(lambda
df_new.insert(loc=3, column='month', value=df_new.InvoiceDate.dt.month)
# +1 to make Monday=1.....until Sunday=7
df_new.insert(loc=4, column='day', value=(df_new.InvoiceDate.dt.dayofweek)+1)
df_new.insert(loc=5, column='hour', value=df_new.InvoiceDate.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
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Exploratory Data Analysis (EDA)

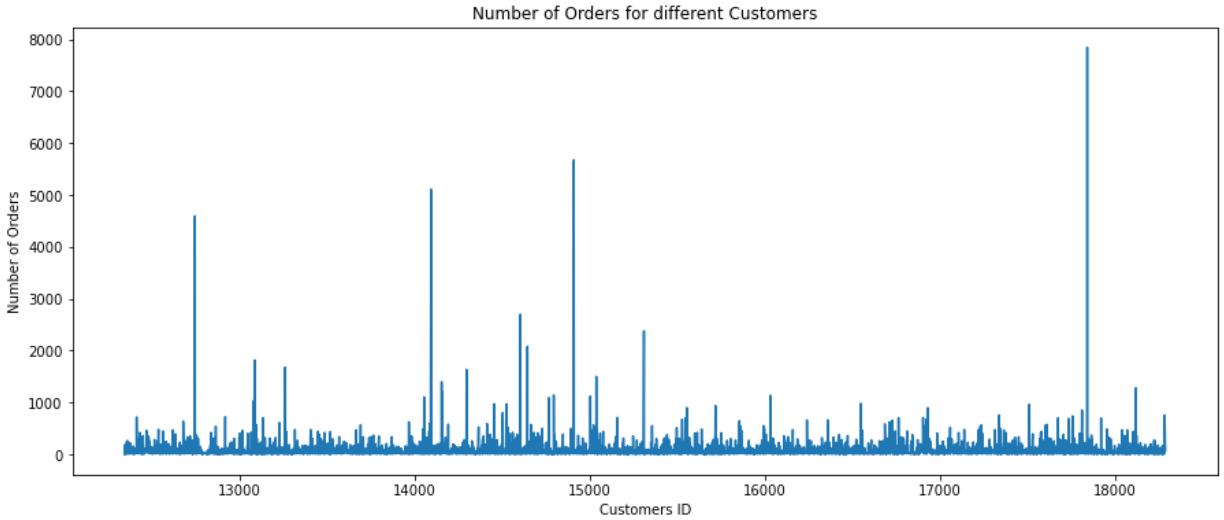
How many orders made by the customers?

In [21]:
```
df_new.groupby(by=['CustomerID','Country'], as_index=False)['InvoiceNo'].count().
```

Out[21]:

	CustomerID	Country	InvoiceNo
0	12346.0	United Kingdom	1
1	12347.0	Iceland	182
2	12348.0	Finland	31
3	12349.0	Italy	73
4	12350.0	Norway	17

In [25]:
```
orders = df_new.groupby(by=['CustomerID','Country'], as_index=False)['InvoiceNo']

import matplotlib.pyplot as plt
plt.subplots(figsize=(15,6))
plt.plot(orders.CustomerID, orders.InvoiceNo)
plt.xlabel('Customers ID')
plt.ylabel('Number of Orders')
plt.title('Number of Orders for different Customers')
plt.show()
```



Check TOP 5 most number of orders

In [27]:
```
print('The TOP 5 customers with most number of orders...')
orders.sort_values(by='InvoiceNo', ascending=False).head()
```

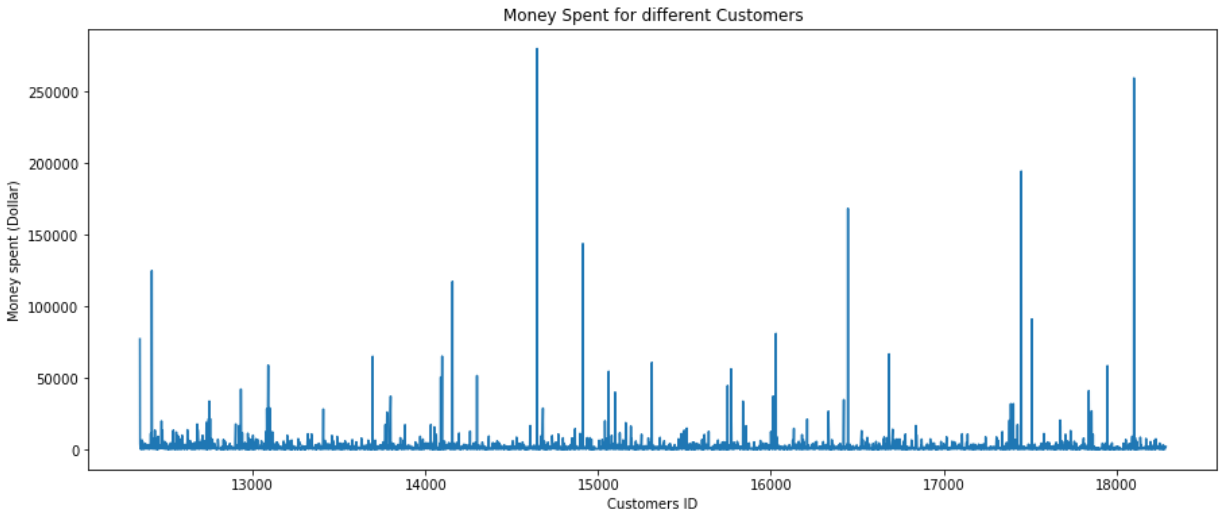
```
The TOP 5 customers with most number of orders...
```

Out[27]:

	CustomerID	Country	InvoiceNo
4019	17841.0	United Kingdom	7847
1888	14911.0	EIRE	5677
1298	14096.0	United Kingdom	5111
334	12748.0	United Kingdom	4596
1670	14606.0	United Kingdom	2700

How much money spent by the customers?

In [28]:
```
money_spent = df_new.groupby(by=['CustomerID','Country'], as_index=False)['amount

plt.subplots(figsize=(15,6))
plt.plot(money_spent.CustomerID, money_spent.amount_spent)
plt.xlabel('Customers ID')
plt.ylabel('Money spent (Dollar)')
plt.title('Money Spent for different Customers')
plt.show()
```



Check TOP 5 highest money spent

In [29]:
```
print('The TOP 5 customers with highest money spent...')
money_spent.sort_values(by='amount_spent', ascending=False).head()
```

```
The TOP 5 customers with highest money spent...
```

Out[29]:

	CustomerID	Country	amount_spent
1698	14646.0	Netherlands	280206.02
4210	18102.0	United Kingdom	259657.30
3737	17450.0	United Kingdom	194550.79
3017	16446.0	United Kingdom	168472.50
1888	14911.0	EIRE	143825.06

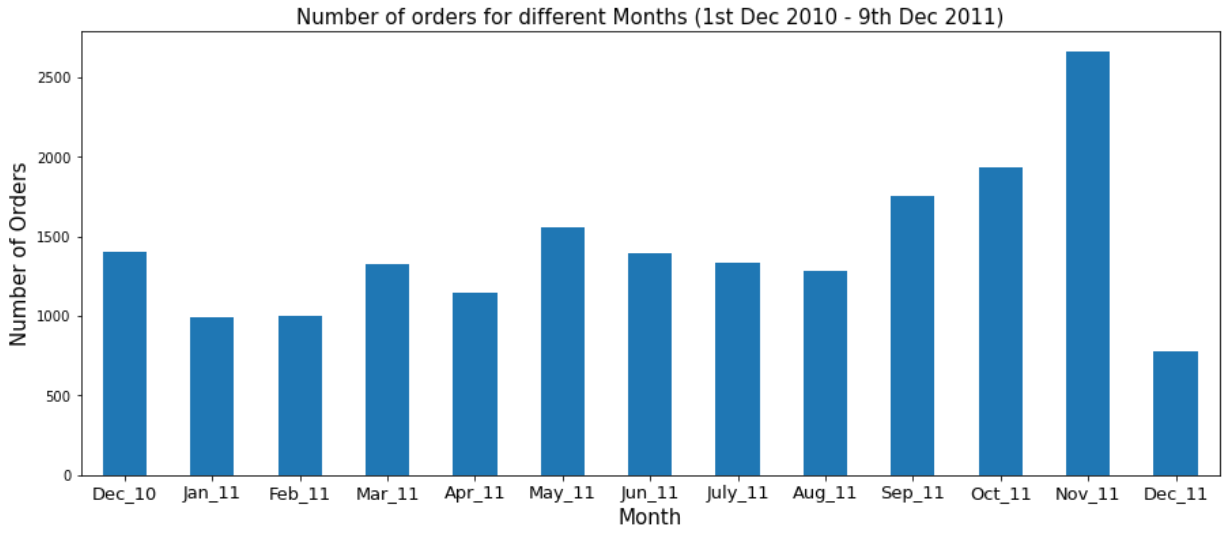
How many orders (per month)?

In [30]:
```
df_new.head()
```

Out[30]:

	InvoiceNo	InvoiceDate	year_month	month	day	hour	StockCode	Description	Quantity	UnitP
0	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	85123A	white hanging heart t-light holder	6.0	
1	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	71053	white metal lantern	6.0	
2	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	84406B	cream cupid hearts coat hanger	8.0	
3	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	84029G	knitted union flag hot water bottle	6.0	
4	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	84029E	red woolly hottie white heart.	6.0	

In [35]:
```
#import seaborn as sns

#sns.palplot(color)

ax = df_new.groupby('InvoiceNo')['year_month'].unique().value_counts().sort_index
ax.set_xlabel('Month',fontsize=15)
ax.set_ylabel('Number of Orders',fontsize=15)
ax.set_title('Number of orders for different Months (1st Dec 2010 - 9th Dec 2011)
ax.set_xticklabels(('Dec_10','Jan_11','Feb_11','Mar_11','Apr_11','May_11','Jun_11
plt.show()
```



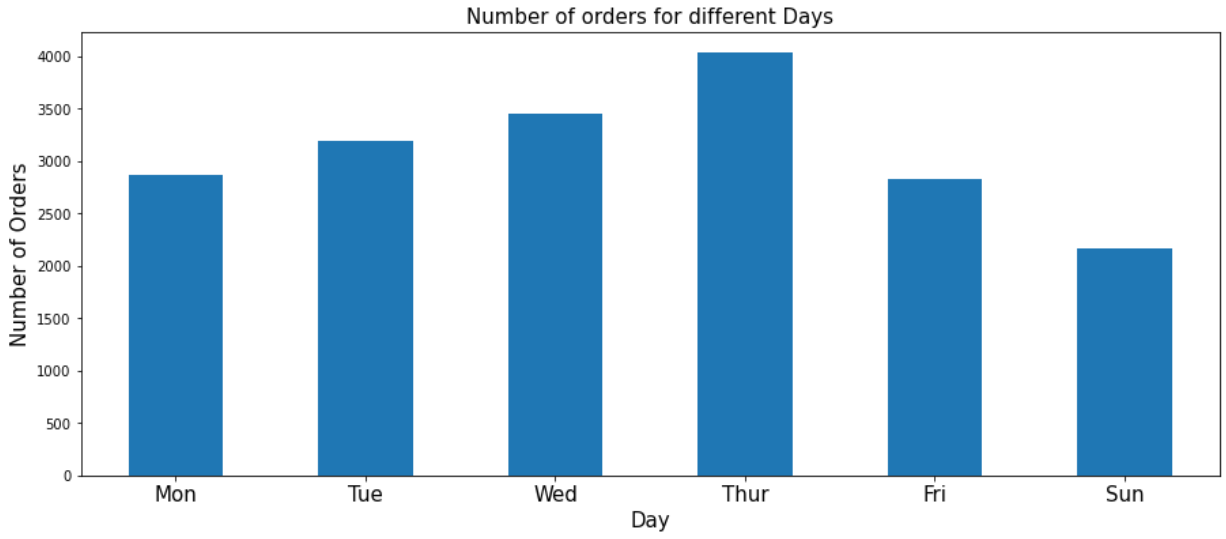
How many orders (per day)?

In [36]:
```
df_new.groupby('InvoiceNo')['day'].unique().value_counts().sort_index()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
able.map_locations()

TypeError: unhashable type: 'numpy.ndarray'


Exception ignored in: 'pandas._libs.index.IndexEngine._call_map_locations'
Traceback (most recent call last):
  File "pandas\_libs\hashtable_class_helper.pxi", line 4588, in pandas._libs.ha
shtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'numpy.ndarray'
```

Out[36]:
```
[1.0]    2863
[2.0]    3185
[3.0]    3455
[4.0]    4033
[5.0]    2831
[7.0]    2169
Name: day, dtype: int64
```

In [37]:
```
ax = df_new.groupby('InvoiceNo')['day'].unique().value_counts().sort_index().plot
ax.set_xlabel('Day',fontsize=15)
ax.set_ylabel('Number of Orders',fontsize=15)
ax.set_title('Number of orders for different Days',fontsize=15)
ax.set_xticklabels(('Mon','Tue','Wed','Thur','Fri','Sun'), rotation='horizontal',
plt.show()
```



How many orders (per hour)?

In [49]:
```
x=df_new.groupby('InvoiceNo')['hour'].unique().value_counts().iloc[:-1].sort_valu
x
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashT
able.map_locations()

TypeError: unhashable type: 'numpy.ndarray'


Exception ignored in: 'pandas._libs.index.IndexEngine._call_map_locations'
Traceback (most recent call last):
  File "pandas\_libs\hashtable_class_helper.pxi", line 4588, in pandas._libs.ha
shtable.PyObjectHashTable.map_locations
TypeError: unhashable type: 'numpy.ndarray'
```

Out[49]:
```
[11.0, 12.0]       1
[20.0]            18
[7.0]             29
[19.0]           144
[18.0]           169
[17.0]           544
[8.0]            555
[16.0]          1100
[9.0]           1394
[15.0]          2038
[10.0]          2226
[14.0]          2275
[11.0]          2276
[13.0]          2637
[12.0]          3129
Name: hour, dtype: int64
```

Discover patterns for Unit Price

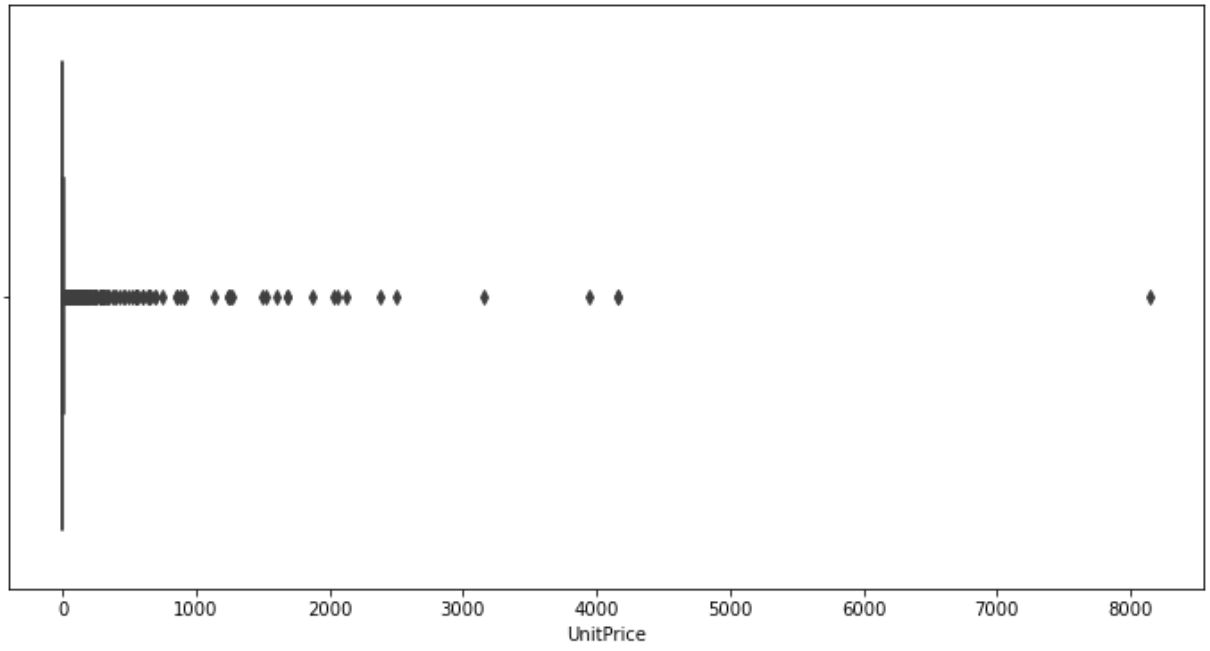
In [52]:
```
df_new.UnitPrice.describe()
```

Out[52]:
```
count    397924.000000
mean          3.116174
std          22.096788
min           0.000000
25%           1.250000
50%           1.950000
75%           3.750000
max        8142.750000
Name: UnitPrice, dtype: float64
```

We see that there are unit price = 0 (FREE items) .There are some free items given to customers from time to time.

In [55]:
```
# check the distribution of unit price
plt.subplots(figsize=(12,6))
sns.boxplot(df_new.UnitPrice)
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments witho
ut an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [56]:
```
df_free = df_new[df_new.UnitPrice == 0]
df_free.head()
```

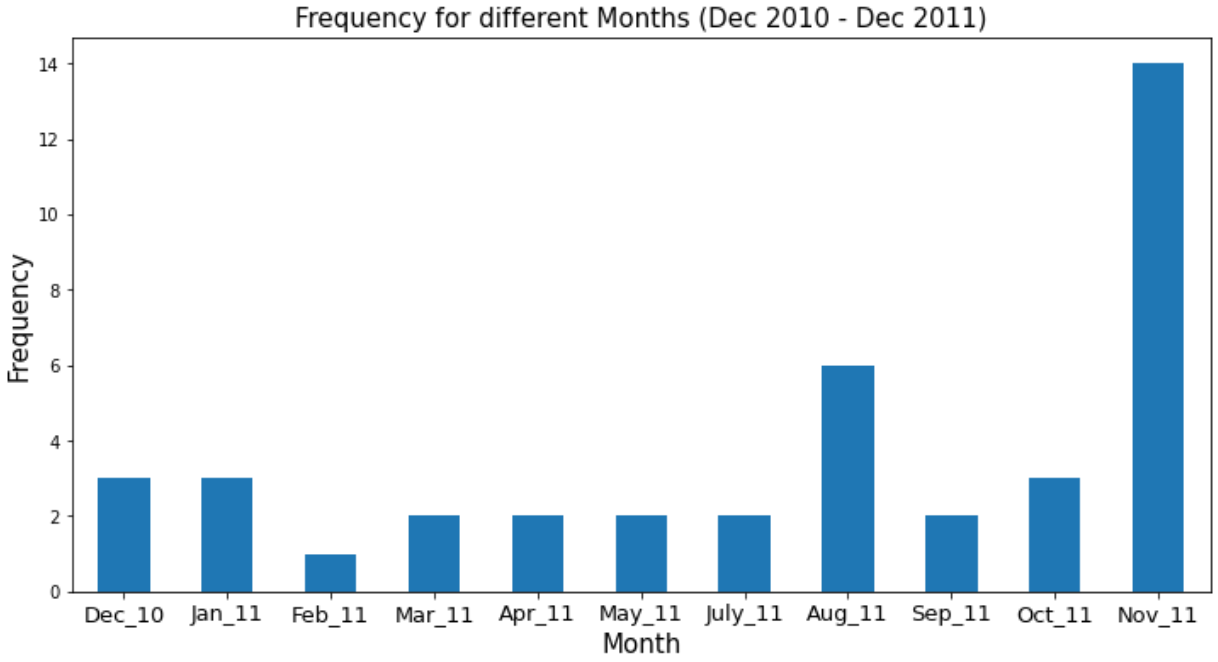
Out[56]:

	InvoiceNo	InvoiceDate	year_month	month	day	hour	StockCode	Description	Quantity
9302	537197	2010-12-05 14:02:00	201012.0	12.0	7.0	14.0	22841	round cake tin vintage green	1.0
33576	539263	2010-12-16 14:36:00	201012.0	12.0	4.0	14.0	22580	advent calendar gingham sack	4.0
40089	539722	2010-12-21 13:45:00	201012.0	12.0	2.0	13.0	22423	regency cakestand 3 tier	10.0
47068	540372	2011-01-06 16:41:00	201101.0	1.0	4.0	16.0	22090	paper bunting retrospot	24.0
47070	540372	2011-01-06 16:41:00	201101.0	1.0	4.0	16.0	22553	plasters in tin skulls	24.0

In [57]:
```
df_free.year_month.value_counts().sort_index()
```

Out[57]:
```
201012.0     3
201101.0     3
201102.0     1
201103.0     2
201104.0     2
201105.0     2
201107.0     2
201108.0     6
201109.0     2
201110.0     3
201111.0    14
Name: year_month, dtype: int64
```

In [59]:
```
ax = df_free.year_month.value_counts().sort_index().plot(kind='bar',figsize=(12,6
ax.set_xlabel('Month',fontsize=15)
ax.set_ylabel('Frequency',fontsize=15)
ax.set_title('Frequency for different Months (Dec 2010 - Dec 2011)',fontsize=15)
ax.set_xticklabels(('Dec_10','Jan_11','Feb_11','Mar_11','Apr_11','May_11','July_1
plt.show()
```



Not clear why there are FREE items given to certain customers,On average, the company gave out 2-4 times FREE items to customers each month (Except in June 2011)

Discover patterns for each Country

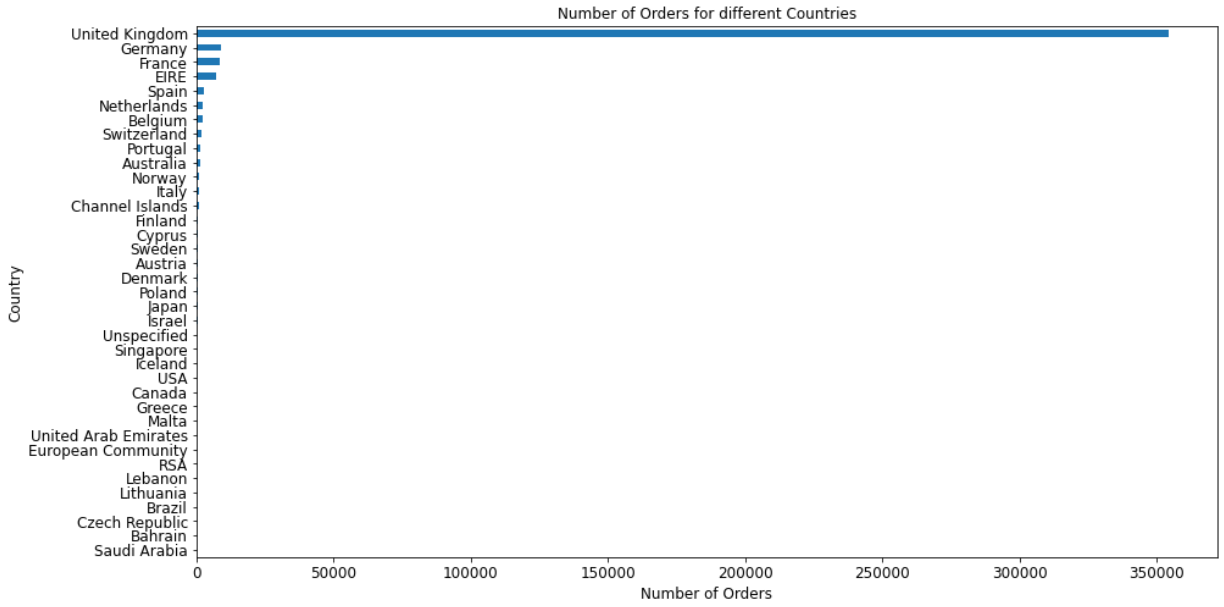
In [60]:
```
df_new.head()
```

Out[60]:

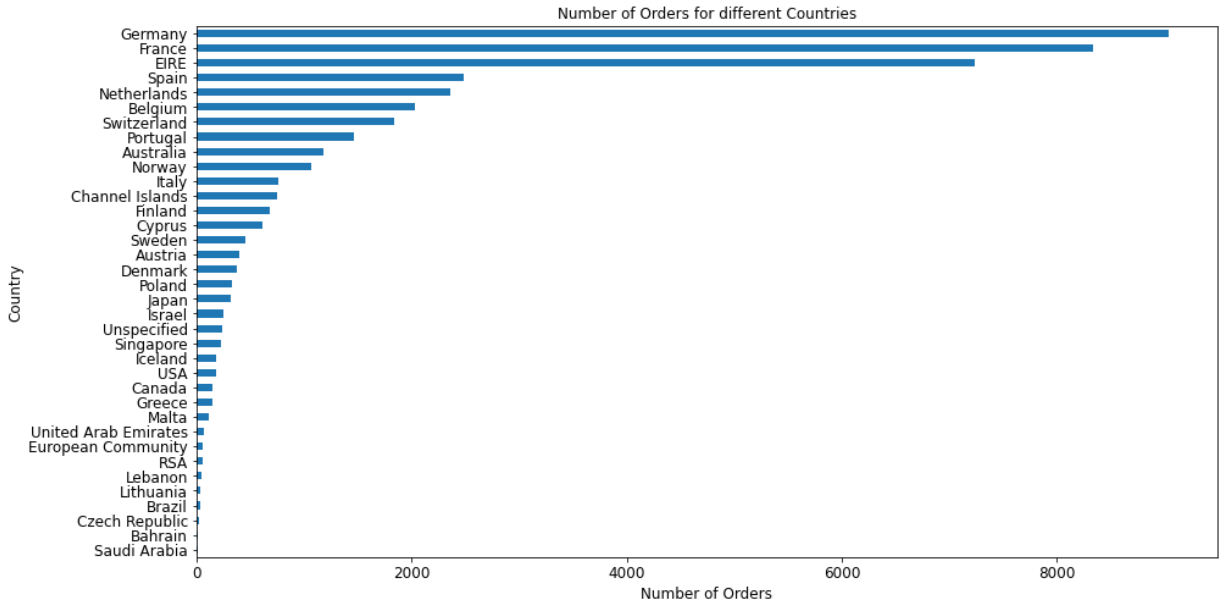
	InvoiceNo	InvoiceDate	year_month	month	day	hour	StockCode	Description	Quantity	UnitP
0	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	85123A	white hanging heart t-light holder	6.0	
1	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	71053	white metal lantern	6.0	
2	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	84406B	cream cupid hearts coat hanger	8.0	
3	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	84029G	knitted union flag hot water bottle	6.0	
4	536365	2010-12-01 08:26:00	201012.0	12.0	3.0	8.0	84029E	red woolly hottie white heart.	6.0	

How many orders for each country?

In [61]:
```
group_country_orders = df_new.groupby('Country')['InvoiceNo'].count().sort_values
# del group_country_orders['United Kingdom']

# plot number of unique customers in each country (with UK)
plt.subplots(figsize=(15,8))
group_country_orders.plot(kind='barh', fontsize=12)
plt.xlabel('Number of Orders', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Number of Orders for different Countries', fontsize=12)
plt.show()
```

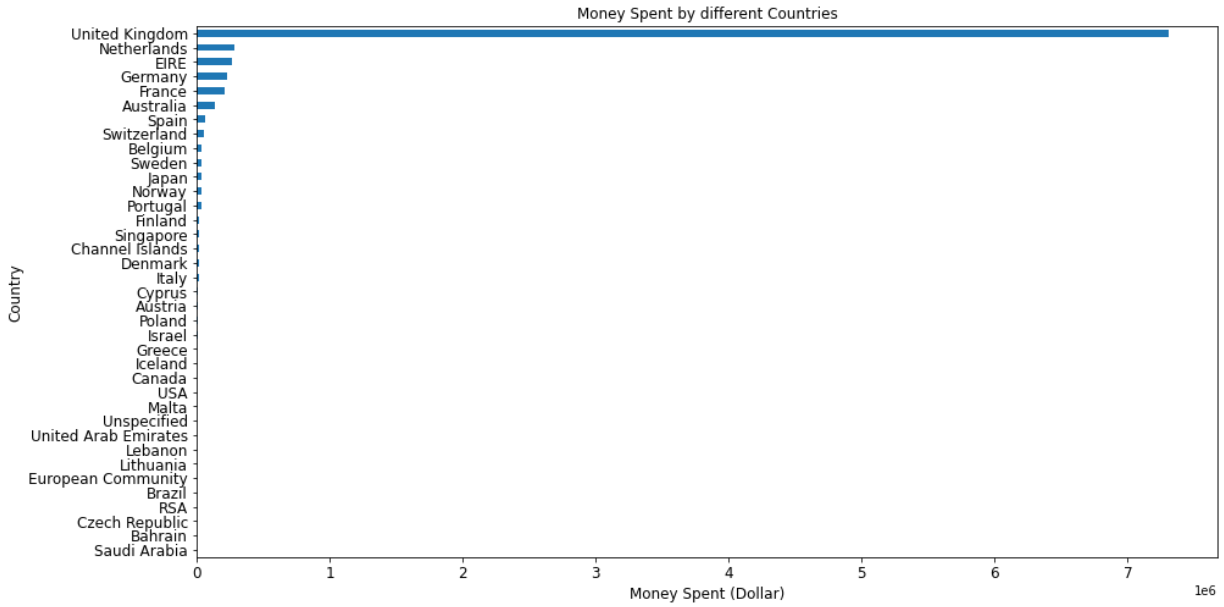


In [64]:
```
group_country_orders = df_new.groupby('Country')['InvoiceNo'].count().sort_values
del group_country_orders['United Kingdom']

# plot number of unique customers in each country (without UK)
plt.subplots(figsize=(15,8))
group_country_orders.plot(kind='barh', fontsize=12)
plt.xlabel('Number of Orders', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Number of Orders for different Countries', fontsize=12)
plt.show()
```

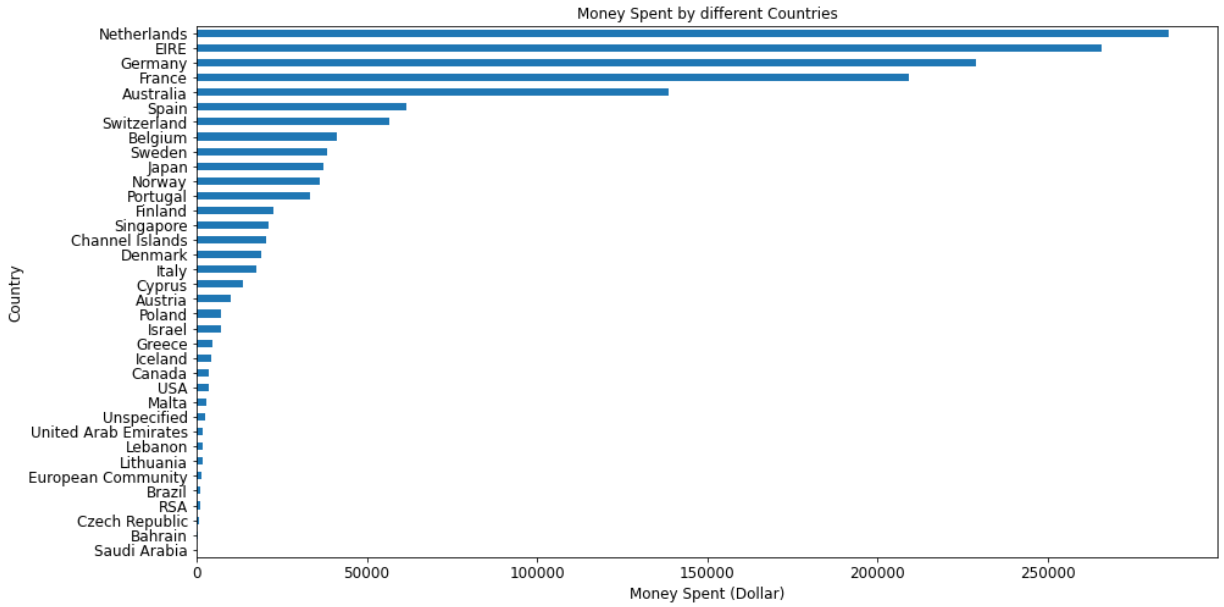


How much money spent by each country?

In [66]:
```
group_country_amount_spent = df_new.groupby('Country')['amount_spent'].sum().sort
# del group_country_orders['United Kingdom']

# plot total money spent by each country (with UK)
plt.subplots(figsize=(15,8))
group_country_amount_spent.plot(kind='barh', fontsize=12)
plt.xlabel('Money Spent (Dollar)', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Money Spent by different Countries', fontsize=12)
plt.show()
```



In [69]:
```
group_country_amount_spent = df_new.groupby('Country')['amount_spent'].sum().sort
del group_country_amount_spent['United Kingdom']

# plot total money spent by each country (without UK)
plt.subplots(figsize=(15,8))
group_country_amount_spent.plot(kind='barh', fontsize=12)
plt.xlabel('Money Spent (Dollar)', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Money Spent by different Countries', fontsize=12)
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: