In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import thinkstats2
import thinkplot
```

Here we are reading the dataset in jupyter notebook. ISO-8859-1 here converts the only the invoice data.

In [2]:
```
df = pd.read_csv("data.csv",encoding = 'ISO-8859-1')
```

In [3]:
```
df.head()
```

Out[3]:

	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

InvoiceNo (invoice_num): A number assigned to each transaction

StockCode (stock_code): Product code

Description (description): Product name

Quantity (quantity): Number of products purchased for each transaction

InvoiceDate (invoice_date): Timestamp for each transaction

UnitPrice (unit_price): Product price per unit

CustomerID (cust_id): Unique identifier each customer

Country (country): Country name

In [4]:
```
df.rename(index=str, columns={'InvoiceNo': 'invoice_num',
                              'StockCode' : 'stock_code',
                              'Description' : 'description',
                              'Quantity' : 'quantity',
                              'InvoiceDate' : 'invoice_date',
                              'UnitPrice' : 'unit_price',
                              'CustomerID' : 'cust_id',
                              'Country' : 'country'}, inplace=True)
```

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

Out[5]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id	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.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 541909 entries, 0 to 541908
Data columns (total 8 columns):
invoice_num     541909 non-null object
stock_code      541909 non-null object
description     540455 non-null object
quantity        541909 non-null int64
invoice_date    541909 non-null object
unit_price      541909 non-null float64
cust_id         406829 non-null float64
country         541909 non-null object
dtypes: float64(2), int64(1), object(5)
memory usage: 37.2+ MB
```

In [7]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[7]:
```
cust_id         135080
description       1454
country              0
unit_price           0
invoice_date         0
quantity             0
stock_code           0
invoice_num          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 [8]:
```
df[df.isnull().any(axis=1)].head()
```

Out[8]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id	count
622	536414	22139	NaN	56	12/1/2010 11:52	0.00	NaN	Unit Kingdc
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	12/1/2010 14:32	2.51	NaN	Unit Kingdc
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	12/1/2010 14:32	2.51	NaN	Unit Kingdc
1445	536544	21786	POLKADOT RAIN HAT	4	12/1/2010 14:32	0.85	NaN	Unit Kingdc
1446	536544	21787	RAIN PONCHO RETROSPOT	2	12/1/2010 14:32	1.66	NaN	Unit Kingdc

In [9]:
```
df['invoice_date'] = pd.to_datetime(df.invoice_date, format='%m/%d/%Y %H:%M')
```

Here we convert the invoice into the format of month/date/year hours:minutes

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

Out[10]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id	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

In [11]:
```
df_new = df.dropna()
```

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

Out[12]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id	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

In [13]:
```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
invoice_num     406829 non-null object
stock_code      406829 non-null object
description     406829 non-null object
quantity        406829 non-null int64
invoice_date    406829 non-null datetime64[ns]
unit_price      406829 non-null float64
cust_id         406829 non-null float64
country         406829 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 27.9+ MB
```

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

Out[14]:

	quantity	unit_price	cust_id
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

By understanding the data in a more descriptive manner, we notice two things:

Quantity has negative values

Unit Price has zero values

In [15]:
```
order_canceled = df['invoice_num'].apply(lambda x:int('C' in x))
n1 = order_canceled.sum()
n2 = df.shape[0]
print('Number of orders canceled: {}/{} ({:.2f}%) '.format(n1, n2, n1/n2*100))
```

```
Number of orders canceled: 9288/541909 (1.71%)
```

Remove Quantity with negative values

In [16]:
```
df_new = df_new[df_new.quantity > 0]
```

In [17]:
```
df_new.describe().round(2)
```

Out[17]:

	quantity	unit_price	cust_id
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

To calculate the total money spent on each purchase, we simply multiply Quantity with Unit Price:

amount_spent = quantity * unit_price

In [18]:
```
df_new['amount_spent'] = df_new['quantity'] * df_new['unit_price']
```

In [19]:
```
# rearrange all the columns for easy reference
df_new = df_new[['invoice_num','invoice_date','stock_code','description','quan
tity','unit_price','amount_spent','cust_id','country']]
```

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

In [20]:
```
df_new.insert(loc=2, column='year_month', value=df_new['invoice_date'].map(lam
bda x: 100*x.year + x.month))
df_new.insert(loc=3, column='month', value=df_new.invoice_date.dt.month)
# +1 to make Monday=1.....until Sunday=7
df_new.insert(loc=4, column='day', value=(df_new.invoice_date.dt.dayofweek)+1)
df_new.insert(loc=5, column='hour', value=df_new.invoice_date.dt.hour)
```

Finally, we add a few columns that consist of the Year_Month, Month, Day and Hour for each transaction for analysis later. The final dataframe will look like this:

In [21]:
```
df_new.head()
```

Out[21]:

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description	quantity
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	WHITE METAL LANTERN	6
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	CREAM CUPID HEARTS COAT HANGER	8
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	RED WOOLLY HOTTIE WHITE HEART.	6

Exploratory Data Analysis (EDA)

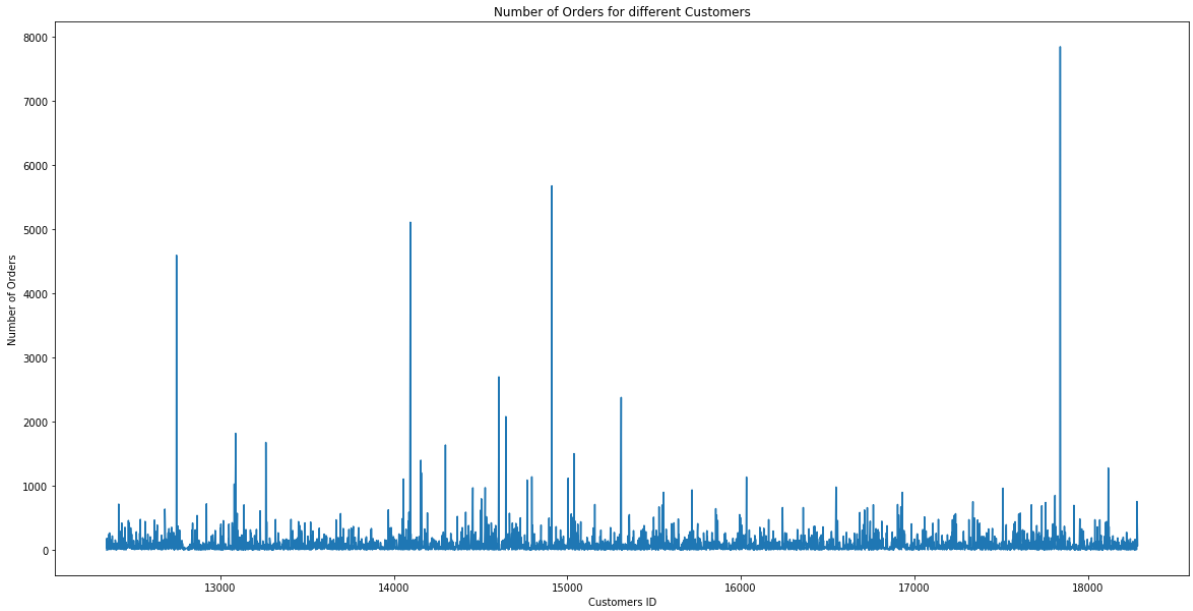
How many orders made by the customers?

In [22]:
```
df_new.groupby(by=['cust_id','country'], as_index=False)['invoice_num'].count().head()
```

Out[22]:

	cust_id	country	invoice_num
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 [23]:
```
orders = df_new.groupby(by=['cust_id','country'], as_index=False)['invoice_num'].count()

plt.subplots(figsize=(20,10))
plt.plot(orders.cust_id, orders.invoice_num)
plt.xlabel('Customers ID')
plt.ylabel('Number of Orders')
plt.title('Number of Orders for different Customers')
plt.show()
```



In [24]:
```
orders.invoice_num.max()
```

Out[24]: 7847

Check TOP 5 most number of orders

In [25]:
```
print('The TOP 5 customers with most number of orders...')
orders.sort_values(by='invoice_num', ascending=False).head()
```

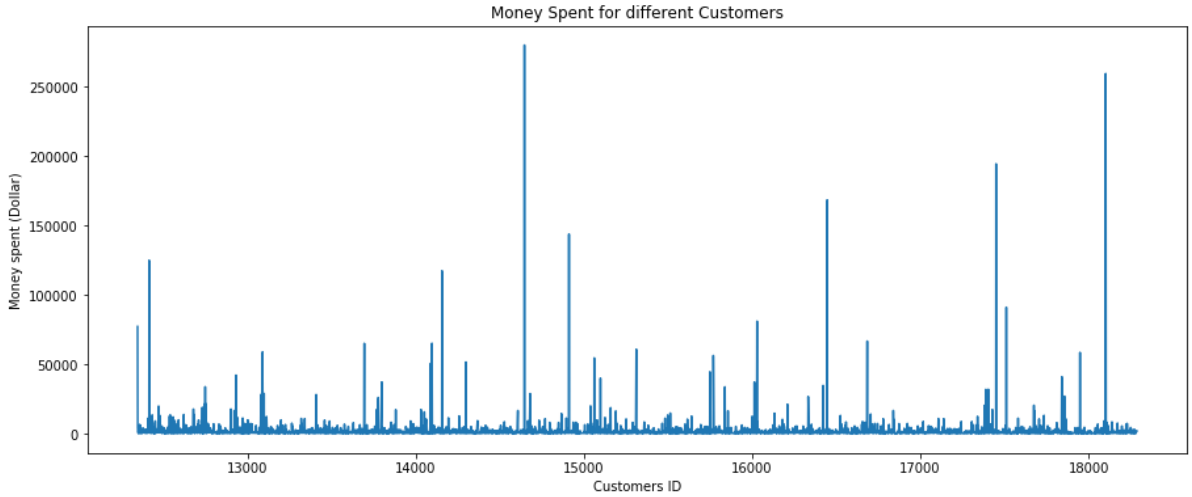
```
The TOP 5 customers with most number of orders...
```

Out[25]:

	cust_id	country	invoice_num
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 [26]:
```
money_spent = df_new.groupby(by=['cust_id','country'], as_index=False)['amount
_spent'].sum()

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



In [27]:
```
print('The TOP 5 customers with highest money spent...')
money_spent=money_spent.sort_values(by='amount_spent', ascending=False)
money_spent.head()
```

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

Out[27]:

	cust_id	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

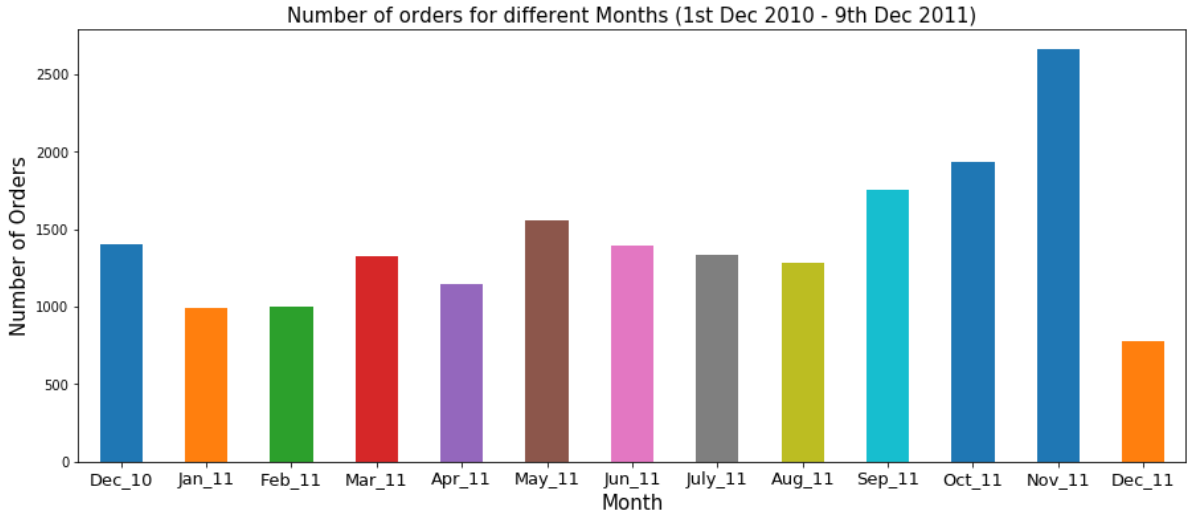
From the results we observe that most orders are made in the UK and customers from Netherlands spend the highest amount of money in their purchases.

How many orders (per month)?

In [28]:
```
df1=df_new.groupby('invoice_num')['year_month'].unique().value_counts().sort_index()
df1
```

Out[28]:
```
[201012]    1400
[201101]     987
[201102]     998
[201103]    1321
[201104]    1149
[201105]    1555
[201106]    1393
[201107]    1331
[201108]    1281
[201109]    1756
[201110]    1929
[201111]    2658
[201112]     778
Name: year_month, dtype: int64
```

In [29]:
```
ax = df1.plot('bar',figsize=(15,6))
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)',fontsize=15)
ax.set_xticklabels(('Dec_10','Jan_11','Feb_11','Mar_11','Apr_11','May_11','Jun_11','July_11','Aug_11','Sep_11','Oct_11','Nov_11','Dec_11'), rotation='horizontal', fontsize=13)
plt.show()
```



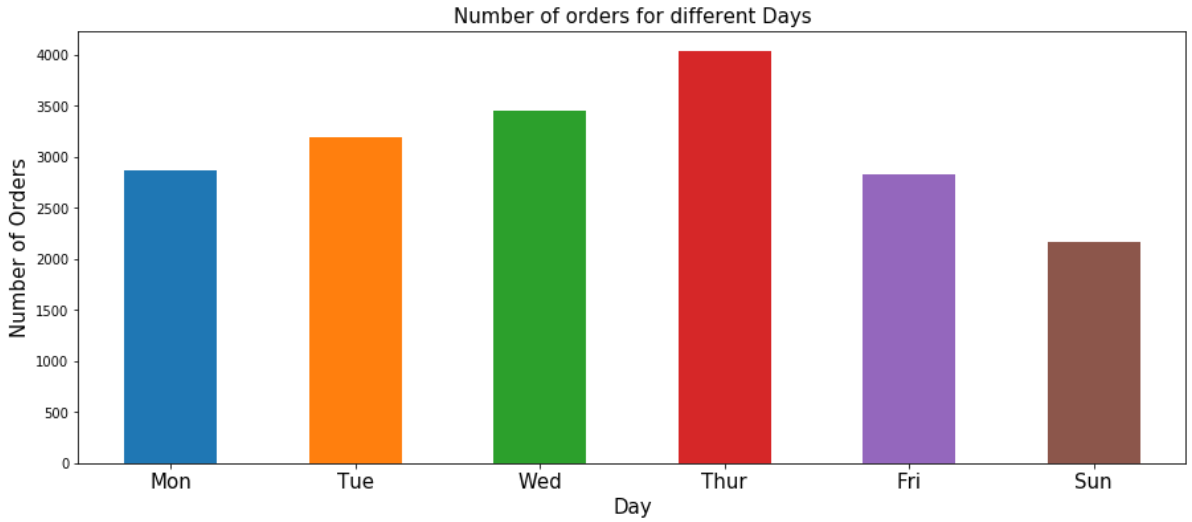
Overall, we consider that the company receives the highest number of orders in November 2011 since we do not have the full month of data for December 2011.

How many orders (per day)?

In [30]:
```
df2=df_new.groupby('invoice_num')['day'].unique().value_counts().sort_index()
df2
```

Out[30]:
```
[1]    2863
[2]    3185
[3]    3455
[4]    4033
[5]    2831
[7]    2169
Name: day, dtype: int64
```

In [31]:
```
ax = df2.plot('bar',figsize=(15,6))
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', fontsize=15)
plt.show()
```



Surprisingly, there are no transactions on Saturday throughout the whole period (1st Dec 2010–9th Dec 2011).We also spot a trend where the number of orders received by the company tends to increases from Monday to Thursday and decrease afterward.

How many orders (per hour)?

In [32]:
```
df3=df_new.groupby('invoice_num')['hour'].unique().value_counts().iloc[:-1].sort_index()
df3
```

Out[32]:
```
[6]        1
[7]       29
[8]      555
[9]     1394
[10]    2226
[11]    2276
[12]    3129
[13]    2637
[14]    2275
[15]    2038
[16]    1100
[17]     544
[18]     169
[19]     144
[20]      18
Name: hour, dtype: int64
```

In [33]:
```
ax = df3.plot('bar',figsize=(15,6))
ax.set_xlabel('Hour',fontsize=15)
ax.set_ylabel('Number of Orders',fontsize=15)
ax.set_title('Number of orders for different Hours',fontsize=15)
ax.set_xticklabels(range(6,21), rotation='horizontal', fontsize=15)
plt.show()
```

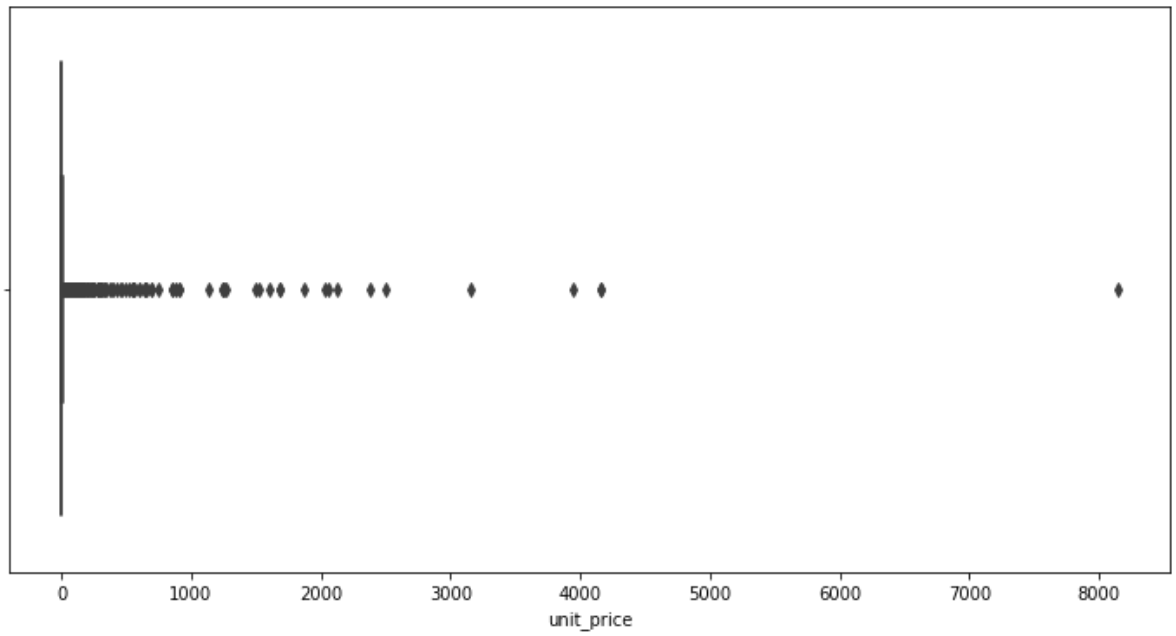
In terms of hours, there are no transactions after 8:00pm until the next day at 6:00am. Besides, we notice that the company receives the highest number of orders at 12:00pm. One of the reasons could be due to the fact that most customers make purchases during lunch hour between 12:00pm — 2:00pm.

Discover patterns for Unit Price

In [34]:
```
df_new.unit_price.describe()
```

Out[34]:
```
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: unit_price, dtype: float64
```

In [35]:
```
plt.subplots(figsize=(12,6))
sns.boxplot(df_new.unit_price)
plt.show()
```



We observe that 75% of the data has unit price of less than 3.75 dollars — which indicates most products are relatively cheap. Only minority of them has high prices per unit

Well… FREE items for purchase? YES, maybe…

In [36]:
```
df_free = df_new[df_new.unit_price == 0]
```

In [37]:
```
df_free.head()
```

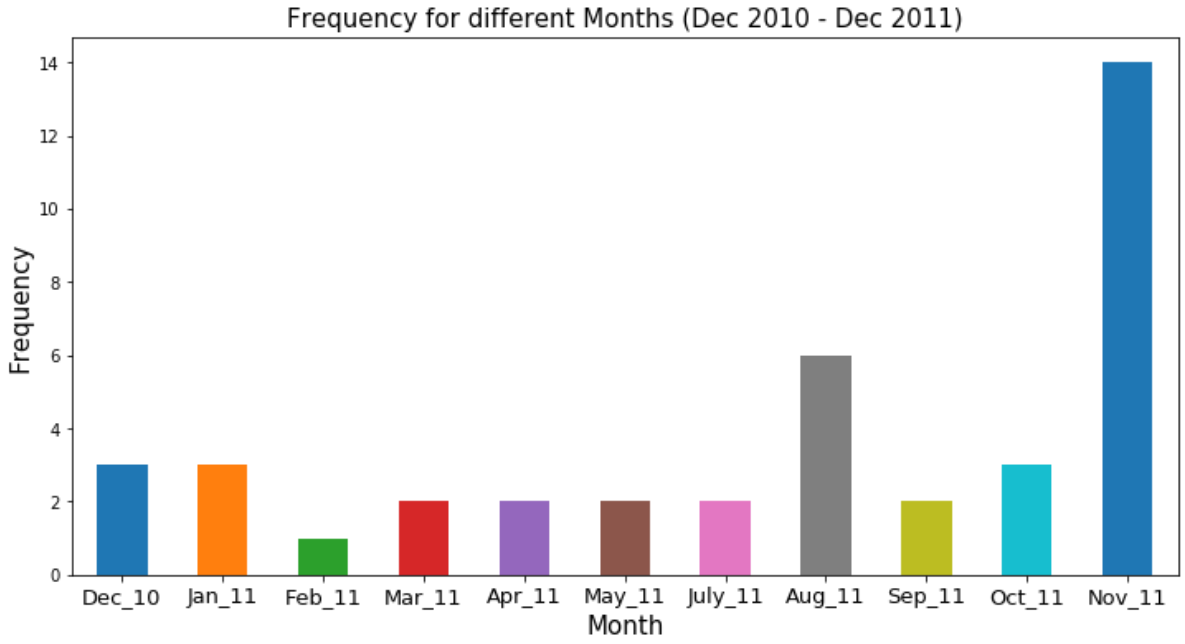
Out[37]:

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description	qu
9302	537197	2010-12-05 14:02:00	201012	12	7	14	22841	ROUND CAKE TIN VINTAGE GREEN	
33576	539263	2010-12-16 14:36:00	201012	12	4	14	22580	ADVENT CALENDAR GINGHAM SACK	
40089	539722	2010-12-21 13:45:00	201012	12	2	13	22423	REGENCY CAKESTAND 3 TIER	
47068	540372	2011-01-06 16:41:00	201101	1	4	16	22090	PAPER BUNTING RETROSPOT	
47070	540372	2011-01-06 16:41:00	201101	1	4	16	22553	PLASTERS IN TIN SKULLS	

In [38]:
```
df4=df_free.year_month.value_counts().sort_index()
df4
```

Out[38]:
```
201012     3
201101     3
201102     1
201103     2
201104     2
201105     2
201107     2
201108     6
201109     2
201110     3
201111    14
Name: year_month, dtype: int64
```

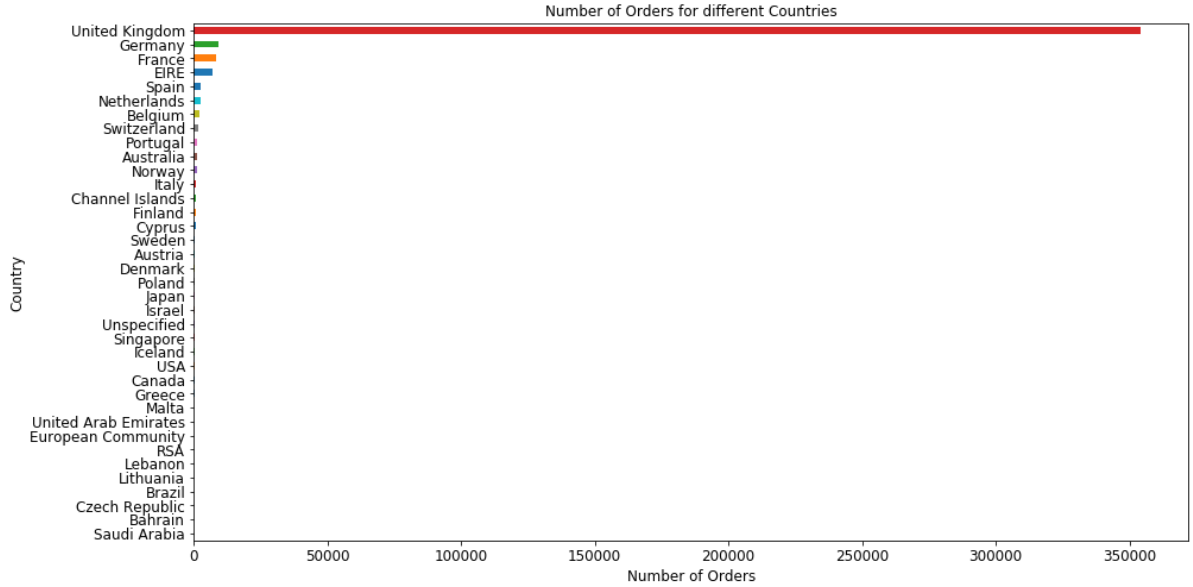
In [39]:
```
ax = df4.plot('bar',figsize=(12,6))
ax.set_xlabel('Month',fontsize=15)
ax.set_ylabel('Frequency',fontsize=15)
ax.set_title('Frequency for different Months (Dec 2010 - Dec 2011)',fontsize=15)
ax.set_xticklabels(('Dec_10','Jan_11','Feb_11','Mar_11','Apr_11','May_11','July_11','Aug_11','Sep_11','Oct_11','Nov_11'), rotation='horizontal', fontsize=13)
plt.show()
```



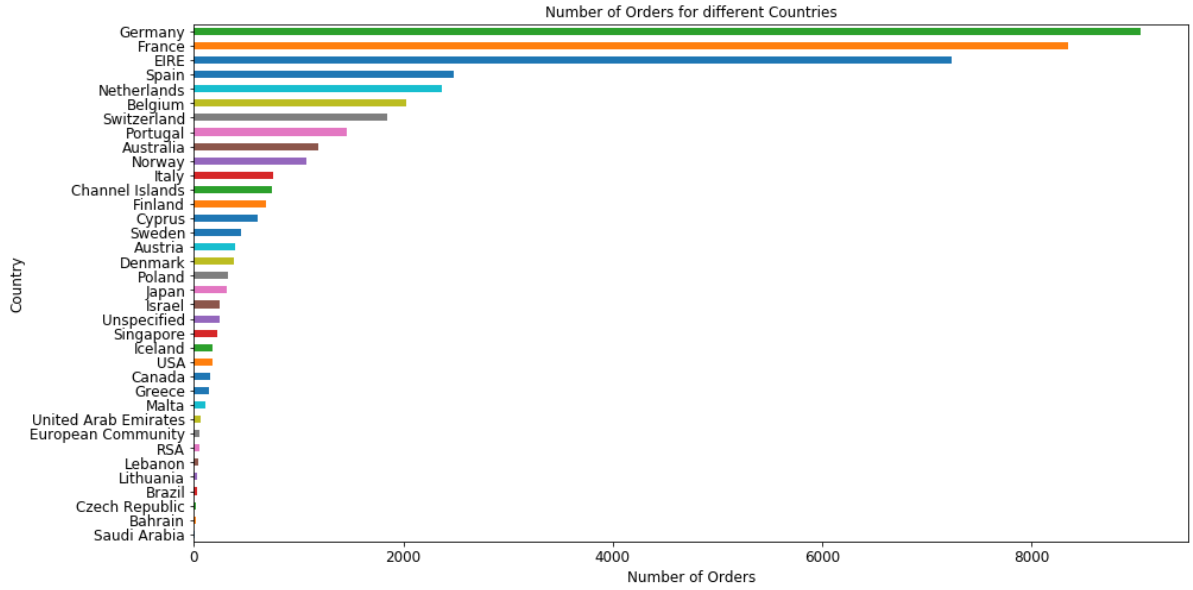
From the plot, the company tends to give out FREE items for purchases occasionally each month (except June 2011).

How many orders for each country?

In [40]:
```
group_country_orders = df_new.groupby('country')['invoice_num'].count().sort_values()

plt.subplots(figsize=(15,8))
group_country_orders.plot('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()
```



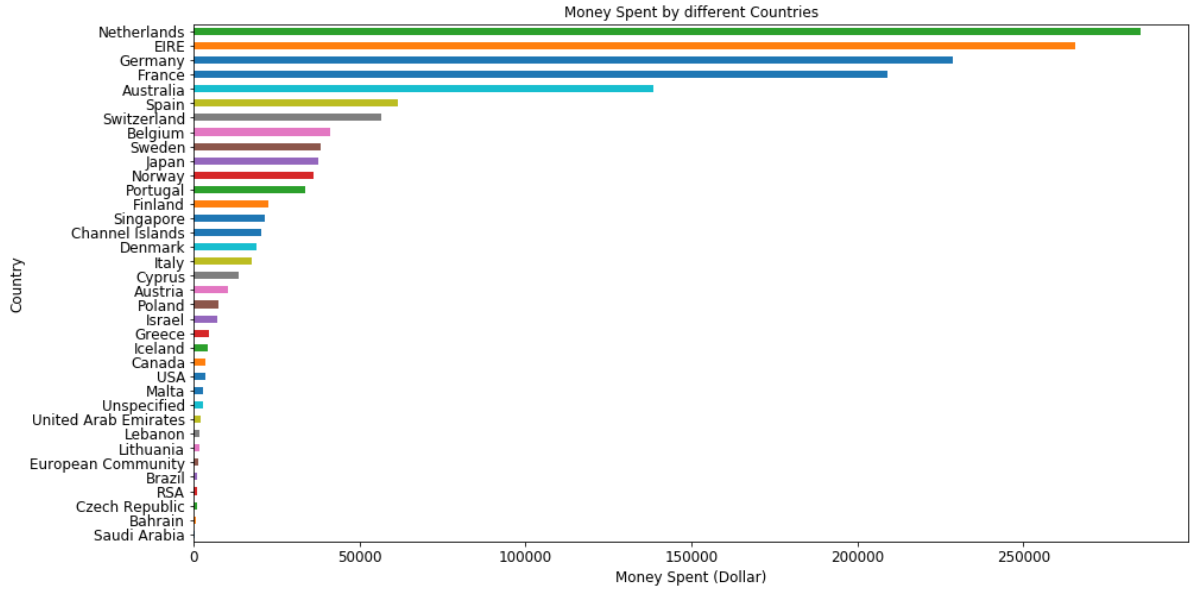
In [41]:
```
group_country_orders = df_new.groupby('country')['invoice_num'].count().sort_values()
del group_country_orders['United Kingdom']


plt.subplots(figsize=(15,8))
group_country_orders.plot('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()
```



As expected, the company receives the highest number of orders in the UK

How much money spent by each country?

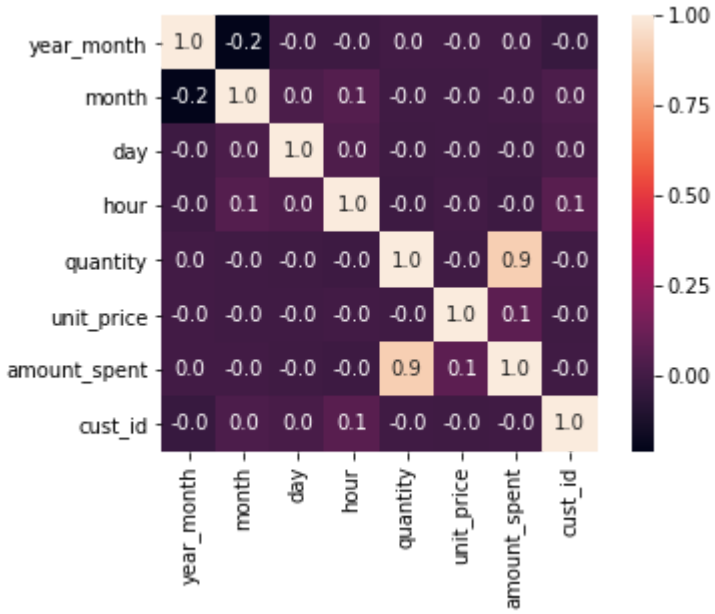
In [42]:
```
group_country_amount_spent = df_new.groupby('country')['amount_spent'].sum().sort_values()
del group_country_amount_spent['United Kingdom']


plt.subplots(figsize=(15,8))
group_country_amount_spent.plot('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()
```



As the company receives the highest number of orders from customers in the UK, it is natural to see that customers in the UK spend the most on their purchases.

In [43]:
```
sns.heatmap(df_new.corr(),square=True,annot=True,fmt='.1f')
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x1c99036f470>



Probability density function

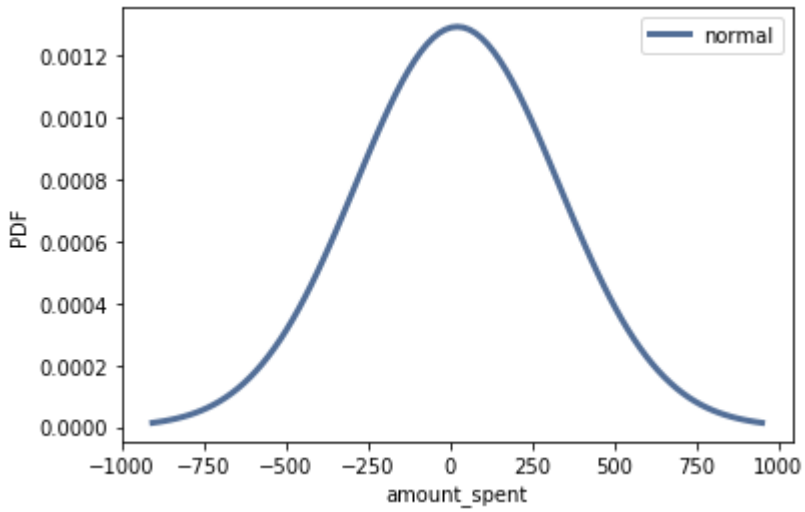
In [44]:
```
df_spent=df_new['amount_spent']
mean,std=df_spent.mean(),df_spent.std()
mean, std
```

Out[44]: (22.39474850474768, 309.05558838012377)

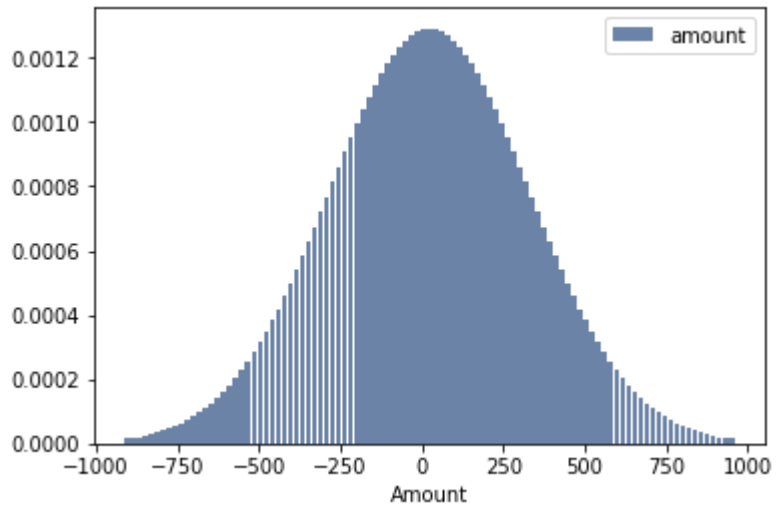
In [45]:
```
print("Skewness: %f" % df_new['amount_spent'].skew())
```

```
Skewness: 451.465538
```

In [46]:
```
pdf = thinkstats2.NormalPdf(mean, std)
thinkplot.Pdf(pdf, label='normal')
thinkplot.Config(xlabel='amount_spent', ylabel='PDF')
```



In [47]:
```
hist = thinkstats2.Hist(pdf, label='amount')
thinkplot.Hist(hist)
thinkplot.Config(xlabel='Amount', ylabeal='Count')
```



confidence interval

In [48]:
```
from sklearn.utils import resample
```

In [49]:
```
amount_spent=df_new[['amount_spent']]
amount_spent
```

Out[49]:

	amount_spent
0	15.30
1	20.34
2	22.00
3	20.34
4	20.34
5	15.30
6	25.50
7	11.10
8	11.10
9	54.08
10	12.60
11	12.60
12	30.00
13	9.90
14	25.50
15	14.85
16	19.90
17	17.85
18	17.85
19	31.80
20	31.80
21	25.50
22	14.85
23	14.85
24	14.85
25	17.85
26	90.00
27	90.00
28	45.00
29	10.20
...	...
541879	30.00
541880	15.00
541881	8.50
541882	10.08

	amount_spent
541883	10.50
541884	15.00
541885	10.20
541886	4.68
541887	15.00
541888	11.40
541889	23.40
541890	23.60
541891	30.00
541892	214.80
541893	70.80
541894	23.40
541895	19.80
541896	19.80
541897	15.00
541898	15.00
541899	15.00
541900	15.00
541901	15.60
541902	23.40
541903	16.60
541904	10.20
541905	12.60
541906	16.60
541907	16.60
541908	14.85

397924 rows × 1 columns

In [50]:
```
means = []

i=0

#confidence interval shrinks with more samples
while i<1000:
    means.append(resample(amount_spent, replace=True, n_samples=len(amount_spe
nt)).mean())
    i+=1
```

In [51]:
```
df_means = pd.DataFrame(means)
```

In [52]:
```
df_means
```

Out[52]:

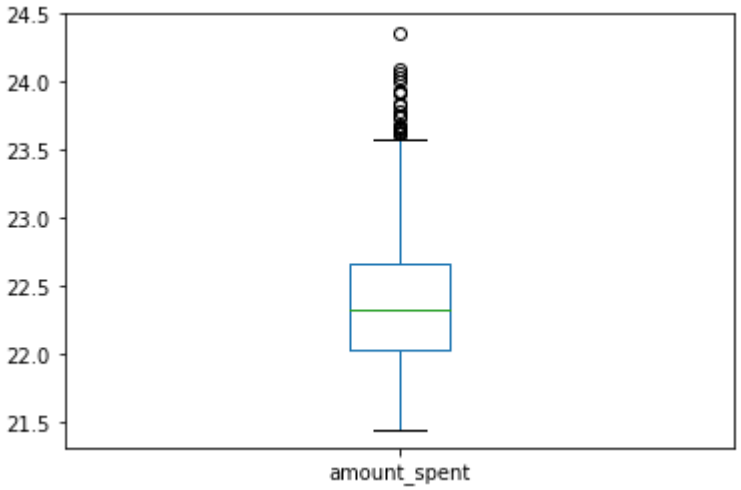
	amount_spent
0	22.497893
1	23.338441
2	22.477598
3	22.448443
4	21.490424
5	22.785318
6	22.705620
7	22.514314
8	22.194327
9	22.060410
10	23.049516
11	21.777683
12	22.623204
13	22.883964
14	21.935546
15	22.694739
16	23.056759
17	22.330766
18	22.398982
19	23.237875
20	21.688536
21	22.540117
22	22.640905
23	23.065799
24	21.979301
25	22.338486
26	22.637560
27	22.312480
28	22.382349
29	22.251829
...	...
970	21.945594
971	22.099651
972	22.536559
973	22.237589

	amount_spent
974	22.608421
975	22.006940
976	22.892985
977	21.749228
978	22.565904
979	23.020716
980	22.255714
981	22.597245
982	22.159107
983	22.037994
984	22.211438
985	23.307687
986	21.948878
987	22.215035
988	22.599847
989	21.815612
990	22.186319
991	22.214422
992	22.566989
993	23.222839
994	22.665348
995	21.737852
996	22.894691
997	23.080787
998	22.663671
999	22.006604

1000 rows × 1 columns

In [53]:
```
df_means.plot.box()
```

Out[53]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c999ba5518>
```



In [54]:
```
confidence_interval = df_means.quantile([0.05 - 0.025, 0.95 + 0.025])
```

In [55]:
```
confidence_interval
```

Out[55]:

	amount_spent
0.025	21.625012
0.975	23.518354

In [56]:
```
lower_interval = confidence_interval.iloc[0,0]
upper_interval = confidence_interval.iloc[1,0]

print(lower_interval, upper_interval)
```

```
21.625012036668185 23.518353892146372
```

In [57]:
```
amount_spent.mean()
```

Out[57]:
```
amount_spent    22.394749
dtype: float64
```

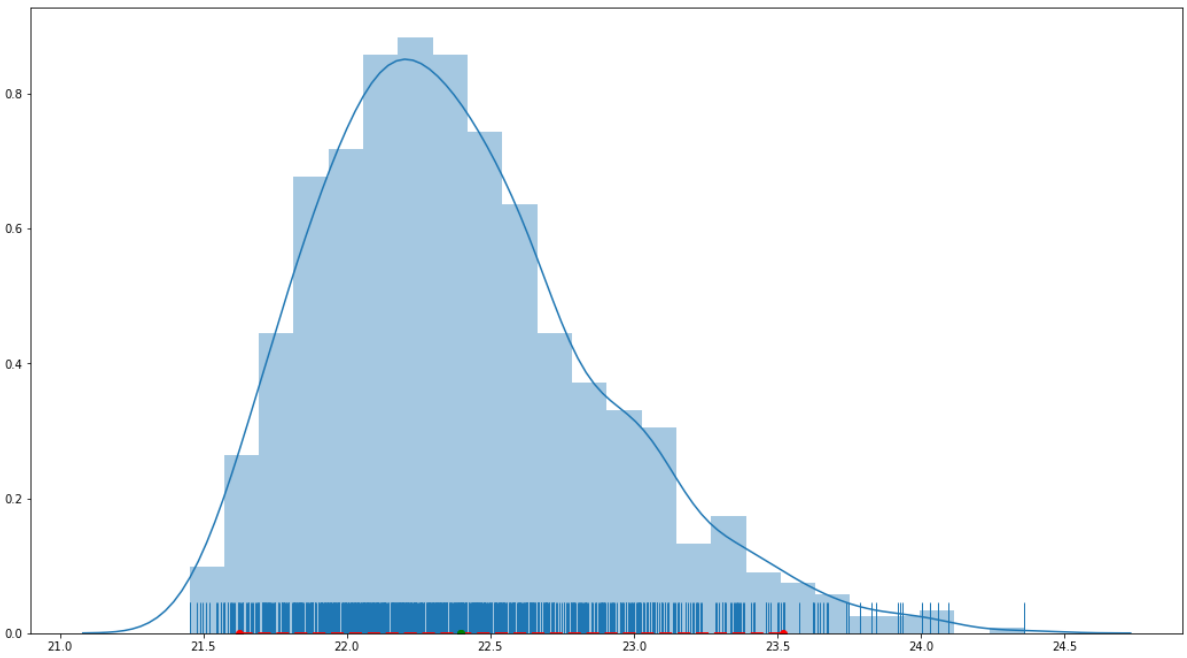
In [58]:
```
if amount_spent.mean().iloc[0] >= lower_interval and amount_spent.mean().iloc[
0] <= upper_interval:
    print('The true mean {} is between the confidence interval of {} and {}'.f
ormat(amount_spent.mean().iloc[0], confidence_interval.iloc[0,0], confidence_i
nterval.iloc[1,0]))
```

```
The true mean 22.39474850474768 is between the confidence interval of 21.6250
12036668185 and 23.518353892146372
```

In [59]:
```
fig, ax = plt.subplots(figsize=(18,10)) # figsize in inches
sns.distplot(df_means, rug=True)
#sns.boxplot(data=df_means, orient="h", notch=True)

#wierd
x1 = [lower_interval, upper_interval]
x2 = [0.0, 0.0]

ax.plot(x1, x2, 'red', linestyle='--', marker='o', lw=3)


y1 = [amount_spent.mean().iloc[0], amount_spent.mean().iloc[0]]
y2 = [0.0, 0.0]


ax.plot(y1, y2, 'green', linestyle='--', marker='o', lw=3)
```

```
C:\Users\Aarushi\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: Futur
eWarning: Using a non-tuple sequence for multidimensional indexing is depreca
ted; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be
interpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[59]:
```
[<matplotlib.lines.Line2D at 0x1c999c3ef28>]
```



How many means are within the confidence interval?

In [60]:
```
df_means['isInInterval'] = df_means['amount_spent'].apply(lambda x : x >= lowe
r_interval and x <= upper_interval)
```

In [61]:
```
df_means
```

Out[61]:

	amount_spent	isInInterval
0	22.497893	True
1	23.338441	True
2	22.477598	True
3	22.448443	True
4	21.490424	False
5	22.785318	True
6	22.705620	True
7	22.514314	True
8	22.194327	True
9	22.060410	True
10	23.049516	True
11	21.777683	True
12	22.623204	True
13	22.883964	True
14	21.935546	True
15	22.694739	True
16	23.056759	True
17	22.330766	True
18	22.398982	True
19	23.237875	True
20	21.688536	True
21	22.540117	True
22	22.640905	True
23	23.065799	True
24	21.979301	True
25	22.338486	True
26	22.637560	True
27	22.312480	True
28	22.382349	True
29	22.251829	True
...	...	...
970	21.945594	True
971	22.099651	True
972	22.536559	True
973	22.237589	True

	amount_spent	isInInterval
974	22.608421	True
975	22.006940	True
976	22.892985	True
977	21.749228	True
978	22.565904	True
979	23.020716	True
980	22.255714	True
981	22.597245	True
982	22.159107	True
983	22.037994	True
984	22.211438	True
985	23.307687	True
986	21.948878	True
987	22.215035	True
988	22.599847	True
989	21.815612	True
990	22.186319	True
991	22.214422	True
992	22.566989	True
993	23.222839	True
994	22.665348	True
995	21.737852	True
996	22.894691	True
997	23.080787	True
998	22.663671	True
999	22.006604	True

1000 rows × 2 columns

In [62]:
```
(sum(df_means['isInInterval']) / len(df_means))*100
```

Out[62]: 95.0

Now we add column of money_spent whether the customer is spending high,adequate or less money-

In [63]:
```
def amount(param):
    if param <=21:
        return 'less'
    elif param >21 and param <=23:
        return 'adequate'
    elif param >23:
        return 'more'

df_new['money_spent'] = df_new['amount_spent'].apply(amount)
```

In [64]:
```
df_new
```

Out[64]:

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description	c
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	WHITE HANGING HEART T-LIGHT HOLDER	
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	WHITE METAL LANTERN	
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	CREAM CUPID HEARTS COAT HANGER	
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	RED WOOLLY HOTTIE WHITE HEART.	
5	536365	2010-12-01 08:26:00	201012	12	3	8	22752	SET 7 BABUSHKA NESTING BOXES	
6	536365	2010-12-01 08:26:00	201012	12	3	8	21730	GLASS STAR FROSTED T-LIGHT HOLDER	
7	536366	2010-12-01 08:28:00	201012	12	3	8	22633	HAND WARMER UNION JACK	
8	536366	2010-12-01 08:28:00	201012	12	3	8	22632	HAND WARMER RED POLKA DOT	
9	536367	2010-12-01 08:34:00	201012	12	3	8	84879	ASSORTED COLOUR BIRD ORNAMENT	
10	536367	2010-12-01 08:34:00	201012	12	3	8	22745	POPPY'S PLAYHOUSE BEDROOM	
11	536367	2010-12-01 08:34:00	201012	12	3	8	22748	POPPY'S PLAYHOUSE KITCHEN	
12	536367	2010-12-01 08:34:00	201012	12	3	8	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	
13	536367	2010-12-01 08:34:00	201012	12	3	8	22310	IVORY KNITTED MUG COSY	

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description	c
14	536367	2010-12-01 08:34:00	201012	12	3	8	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	
15	536367	2010-12-01 08:34:00	201012	12	3	8	22623	BOX OF VINTAGE JIGSAW BLOCKS	
16	536367	2010-12-01 08:34:00	201012	12	3	8	22622	BOX OF VINTAGE ALPHABET BLOCKS	
17	536367	2010-12-01 08:34:00	201012	12	3	8	21754	HOME BUILDING BLOCK WORD	
18	536367	2010-12-01 08:34:00	201012	12	3	8	21755	LOVE BUILDING BLOCK WORD	
19	536367	2010-12-01 08:34:00	201012	12	3	8	21777	RECIPE BOX WITH METAL HEART	
20	536367	2010-12-01 08:34:00	201012	12	3	8	48187	DOORMAT NEW ENGLAND	
21	536368	2010-12-01 08:34:00	201012	12	3	8	22960	JAM MAKING SET WITH JARS	
22	536368	2010-12-01 08:34:00	201012	12	3	8	22913	RED COAT RACK PARIS FASHION	
23	536368	2010-12-01 08:34:00	201012	12	3	8	22912	YELLOW COAT RACK PARIS FASHION	
24	536368	2010-12-01 08:34:00	201012	12	3	8	22914	BLUE COAT RACK PARIS FASHION	
25	536369	2010-12-01 08:35:00	201012	12	3	8	21756	BATH BUILDING BLOCK WORD	
26	536370	2010-12-01 08:45:00	201012	12	3	8	22728	ALARM CLOCK BAKELIKE PINK	
27	536370	2010-12-01 08:45:00	201012	12	3	8	22727	ALARM CLOCK BAKELIKE RED	
28	536370	2010-12-01 08:45:00	201012	12	3	8	22726	ALARM CLOCK BAKELIKE GREEN	

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description	c
29	536370	2010-12-01 08:45:00	201012	12	3	8	21724	PANDA AND BUNNIES STICKER SHEET	
...	...	...	...	...	...	...	...	...	
541879	581585	2011-12-09 12:31:00	201112	12	5	12	22726	ALARM CLOCK BAKELIKE GREEN	
541880	581585	2011-12-09 12:31:00	201112	12	5	12	22727	ALARM CLOCK BAKELIKE RED	
541881	581585	2011-12-09 12:31:00	201112	12	5	12	16016	LARGE CHINESE STYLE SCISSOR	
541882	581585	2011-12-09 12:31:00	201112	12	5	12	21916	SET 12 RETRO WHITE CHALK STICKS	
541883	581585	2011-12-09 12:31:00	201112	12	5	12	84692	BOX OF 24 COCKTAIL PARASOLS	
541884	581585	2011-12-09 12:31:00	201112	12	5	12	84946	ANTIQUE SILVER T-LIGHT GLASS	
541885	581585	2011-12-09 12:31:00	201112	12	5	12	21684	SMALL MEDINA STAMPED METAL BOWL	
541886	581585	2011-12-09 12:31:00	201112	12	5	12	22398	MAGNETS PACK OF 4 SWALLOWS	
541887	581585	2011-12-09 12:31:00	201112	12	5	12	23328	SET 6 SCHOOL MILK BOTTLES IN CRATE	
541888	581585	2011-12-09 12:31:00	201112	12	5	12	23145	ZINC T-LIGHT HOLDER STAR LARGE	
541889	581585	2011-12-09 12:31:00	201112	12	5	12	22466	FAIRY TALE COTTAGE NIGHT LIGHT	
541890	581586	2011-12-09 12:49:00	201112	12	5	12	22061	LARGE CAKE STAND HANGING STRAWBERY	
541891	581586	2011-12-09 12:49:00	201112	12	5	12	23275	SET OF 3 HANGING OWLS OLLIE BEAK	

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description	c
541892	581586	2011-12-09 12:49:00	201112	12	5	12	21217	RED RETROSPOT ROUND CAKE TINS	
541893	581586	2011-12-09 12:49:00	201112	12	5	12	20685	DOORMAT RED RETROSPOT	
541894	581587	2011-12-09 12:50:00	201112	12	5	12	22631	CIRCUS PARADE LUNCH BOX	
541895	581587	2011-12-09 12:50:00	201112	12	5	12	22556	PLASTERS IN TIN CIRCUS PARADE	
541896	581587	2011-12-09 12:50:00	201112	12	5	12	22555	PLASTERS IN TIN STRONGMAN	
541897	581587	2011-12-09 12:50:00	201112	12	5	12	22728	ALARM CLOCK BAKELIKE PINK	
541898	581587	2011-12-09 12:50:00	201112	12	5	12	22727	ALARM CLOCK BAKELIKE RED	
541899	581587	2011-12-09 12:50:00	201112	12	5	12	22726	ALARM CLOCK BAKELIKE GREEN	
541900	581587	2011-12-09 12:50:00	201112	12	5	12	22730	ALARM CLOCK BAKELIKE IVORY	
541901	581587	2011-12-09 12:50:00	201112	12	5	12	22367	CHILDRENS APRON SPACEBOY DESIGN	
541902	581587	2011-12-09 12:50:00	201112	12	5	12	22629	SPACEBOY LUNCH BOX	
541903	581587	2011-12-09 12:50:00	201112	12	5	12	23256	CHILDRENS CUTLERY SPACEBOY	
541904	581587	2011-12-09 12:50:00	201112	12	5	12	22613	PACK OF 20 SPACEBOY NAPKINS	
541905	581587	2011-12-09 12:50:00	201112	12	5	12	22899	CHILDREN'S APRON DOLLY GIRL	
541906	581587	2011-12-09 12:50:00	201112	12	5	12	23254	CHILDRENS CUTLERY DOLLY GIRL	
541907	581587	2011-12-09 12:50:00	201112	12	5	12	23255	CHILDRENS CUTLERY CIRCUS PARADE	

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description	c
541908	581587	2011-12-09 12:50:00	201112	12	5	12	22138	BAKING SET 9 PIECE RETROSPOT	

397924 rows × 14 columns

In [65]:
```
df1=df_new[['amount_spent','money_spent']]
df1.head()
```

Out[65]:

	amount_spent	money_spent
0	15.30	less
1	20.34	less
2	22.00	adequate
3	20.34	less
4	20.34	less

In [70]:
```
count = df1.groupby(['money_spent']).count()
```

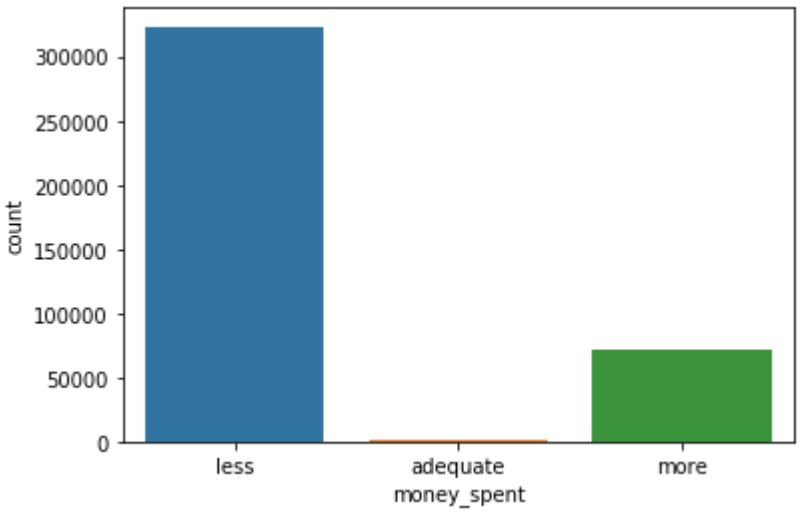
In [71]:
```
count
```

Out[71]:

	amount_spent
money_spent	
adequate	2676
less	322336
more	72912

In [73]:
```
sns.countplot(df_new['money_spent'])
```

Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x1c998499f28>



Chi Square test of independence

Our hypotheses will be:

H0:There is a relationship between 2 categorical variables(country and money_spent)

H1:There is no relationship between 2 categorical variables(country and money_spent)

In [74]:
```
contingency_table=pd.crosstab(df_new['country'],df_new["money_spent"])
print('contingency_table :-\n',contingency_table.head(3))
```

```
contingency_table :-
 money_spent  adequate  less  more
country
Australia           3   371   811
Austria             2   297    99
Bahrain             0     8     9
```

In [75]:
```
Observed_Values = contingency_table.head(3).values
print("Observed Values :-\n",Observed_Values)
```

```
Observed Values :-
 [[  3 371 811]
 [  2 297  99]
 [  0   8   9]]
```

In [76]:
```
import scipy.stats
b=scipy.stats.chi2_contingency(contingency_table.head(3))
Expected_Values = b[3]
print("Expected Values :-\n",Expected_Values)
```

```
Expected Values :-
 [[3.70312500e+00 5.00662500e+02 6.80634375e+02]
 [1.24375000e+00 1.68155000e+02 2.28601250e+02]
 [5.31250000e-02 7.18250000e+00 9.76437500e+00]]
```

In [77]:
```
no_of_rows=len(contingency_table.iloc[0:3,0])
no_of_columns=len(contingency_table.iloc[0,0:3])
degree=(no_of_rows-1)*(no_of_columns-1)
print("Degree of Freedom:-",degree)
```

```
Degree of Freedom:- 4
```

In [78]:
```
#Significance Level 5%
alpha=0.05
```

In [79]:
```
from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
print("chi-square statistic:-",chi_square_statistic)
```

```
chi-square statistic:- 133.04433414817888
```

In [80]:
```
critical_value=chi2.ppf(q=1-alpha,df=degree)
print('critical_value:',critical_value)
```

```
critical_value: 9.487729036781154
```

In [81]:
```
p_value=1-chi2.cdf(x=chi_square_statistic,df=degree)
print('p-value:',p_value)
```

```
p-value: 0.0
```

In [82]:
```
print('Significance level: ',alpha)
print('Degree of Freedom: ',degree)
print('chi-square statistic:',chi_square_statistic)
print('critical_value:',critical_value)
print('p-value:',p_value)
```

```
Significance level:  0.05
Degree of Freedom:  4
chi-square statistic: 133.04433414817888
critical_value: 9.487729036781154
p-value: 0.0
```

In [83]:
```
if chi_square_statistic>=critical_value:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables"
)

if p_value<=alpha:
    print("Reject H0,There is a relationship between 2 categorical variables")
else:
    print("Retain H0,There is no relationship between 2 categorical variables"
)
```

```
Reject H0,There is a relationship between 2 categorical variables
Reject H0,There is a relationship between 2 categorical variables
```

Bayes theorem

$$p(uk|high)=p(uk) *p(high|uk)/p(high)$$

Statement: What are the chance that the customers are from uk given that shopping website have selected the customer who spent the money -"more"?

In [160]:
```
class BayesTable(pd.DataFrame):
    def __init__(self, hypothesis, prior=1):
        columns = ['hypothesis', 'prior', 'likelihood', 'unnormalized', 'norma
lized','posterior']
        super().__init__(columns=columns)
        self.hypothesis = hypothesis
        self.prior = prior

    def multiply(self):
        self.unnormalized = self.prior * self.likelihood

    def normalize(self):

        self.posterior = self.unnormalized / self.normalized
```

In [149]:
```
high_prob=(df_new['money_spent']=='more').sum()/len(df_new['money_spent'])
high_prob
```

Out[149]: 0.18323096872769676

In [150]:
```
uk_prob=(df_new['country']=='United Kingdom').sum()/len(df_new['country'])
uk_prob
```

Out[150]: 0.8904841125441039

In [151]:
```
df1=df_new[df_new['money_spent']=='more']['country']
df2=(df1=='United Kingdom').sum()
highuk_prob=df2/(df_new['country']=='United Kingdom').sum()
highuk_prob
```

Out[151]: 0.16549125851923974

In [152]:
```
table = BayesTable(['uk'])
```

In [153]:
```
table
```

Out[153]:

	hypothesis	prior	likelihood	unnormalized	normalized	posterior
0	uk	1	NaN	NaN	NaN	NaN

In [154]:
```
table.prior=[uk_prob]
```

In [155]:
```
table.likelihood=[highuk_prob]
```

In [156]:
```
table.multiply()
table
```

Out[156]:

	hypothesis	prior	likelihood	unnormalized	normalized	posterior
0	uk	0.890484	0.165491	0.147367	NaN	NaN

In [157]:
```
table.normalized=[high_prob]
```

In [163]:
```
table.normalize()
```

Out[163]:
```
0    0.804271
Name: posterior, dtype: float64
```

In [164]:
```
table
```

Out[164]:

	hypothesis	prior	likelihood	unnormalized	normalized	posterior
0	uk	0.890484	0.165491	0.147367	0.183231	0.804271

So the probabilty of uk customers are 0.804

In [ ]:
```

```