STATISTICS FOR DATA SCIENCE
CODE : UE19CS203

Project Title: STUDY AND EXPLORATORY DATA ANALYSIS OF THE TRANSACTIONS OF A UNITED KINGDOM BASED ONLINE STORE

Section : G

Team members:

- Sreenath Saikumar, SRN: PES2UG19CS406
- Suravarjhula Sasira, SRN: PES2UG19CS415
- Suhan B Revankar, SRN: PES2UG19CS412
- Vanga Bala Nava Sai Sri Nithya Reddy, SRN: PES2UG19CS446

1.Abstract:

With the absolute explosion of online shopping in the late 2000's and early 2010's, transaction based data became a treasure trove for many data scientists eagerly seeking out more and more methods to further capture, enlarge and diversify their customer base. Sites like Amazon, Flipkart, AliExpress etc. have been flourishing for the past few years to the point of rendering brick and mortar stores obsolete in many parts of the world.

This dataset was chosen with this aim in mind and the various analyses performed on it helps us to better understand market trends, customer segmentation and allows us to pinpoint methods to ensure better item sales down to the hour. The resulting conclusions help us to optimize and bring out the best that online sales have to offer.

2.Introduction:

With the COVID-19 Pandemic having struct the world in late 2019-early 2020 and forcing people to the confines of their homes to prevent further spread and ensure people's safety, many have turned to online stores as their haven for supplies amidst these uncertain times.

A survey entitled 'COVID-19 and E-commerce' examined how the pandemic had changed the way people use e-commerce and other digital solutions. Following the pandemic, more the half of the survey's respondents now shopped online more frequently and that consumers in emerging economies had made the greatest shift to online shopping. The dataset chosen further tries to drive home the fact that even before the pandemic, online sales were starting to pick up for a while now with many purchases coming in from customers located outside of the United Kingdom where the store in the dataset is based in.

Analysing customer behaviour when it comes to items in a certain price bracket or the decisions they make with respect to the quantity and even the colour of the items purchased is one of the aims of this data analysis project with more in depth explanations given in the paragraphs below.

3. Dataset:

The dataset chosen is called the 'E-commerce Data' dataset which can be described as a 'Transnational dataset which contains all the transactions occurring between 1/12/2010 and 09/12/2011 for a UK-based and registered **non-store** online retail. The company mainly sells unique all-occasion gifts and many customers are usually wholesalers.

The dataset can be found at the following sites:

- https://www.kaggle.com/carrie1/ecommerce-data
- https://archive.ics.uci.edu/ml/datasets/Online+Retail

It can be found and downloaded from Kaggle or the UCI Machine Learning Repository from the links given above.

```
df.head(10)
```

	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	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047	United Kingdom

The dataset chosen contains the following features:

1. InvoiceNo: Stores the invoice number for the item/items being purchased.
2. Stock Code: Stores the unique code that is given to each item in the inventory.
3. Description: Stores information about the products in the inventory.
4. Quantity: Indicates the quantity of the item being purchased.
5. InvoiceDate: Stores the date and time at which the item/ items were purchased.
6. UnitPrice: Stores the price of each item.
7. CustomerID: Stores the unique ID of each customer that makes a purchase on the online store.
8. Country: Stores information about the country from which the customer has made the purchase.

4.Preprocessing and Data Cleaning:

The first step was to identify the size of the uncleaned and unprocessed dataset.

```
df.shape
```

```
(541909, 8)
```

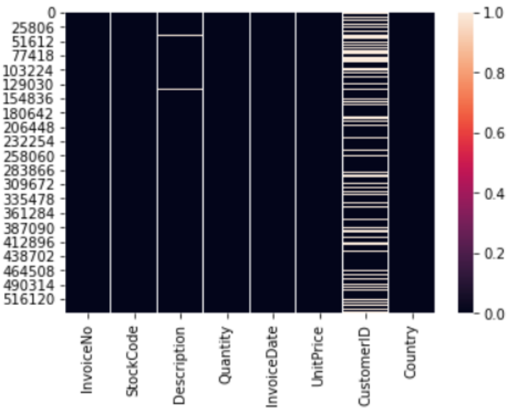
The next step was to identify whether there were any null/ missing values in any of the entries in the dataset. We also plot a heatmap to allow us to better visualize the missing values.

```
df.isnull().sum()
```

```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

```
sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0:
```



We notice that the CustomerID feature has a lot of missing values so we change all the missing values to represent unsold items. We also drop the missing values in the Description feature as we cannot assume anything about missing products. The heatmap once more shows us that we have removed all NULL/ missing values successfully.

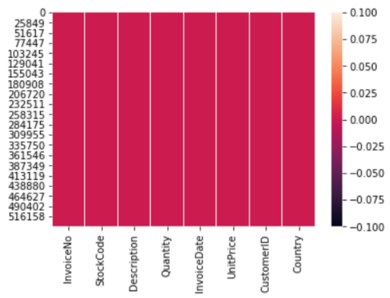
```
df['CustomerID'].fillna("Unsold",inplace=True)
```

```
df.dropna(subset=['Description'],inplace=True)
df.isnull().sum()
```

```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

```
sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5a
```



To further clean up the dataset, we remove any duplicate values and also any negative values in the UnitPrice feature to remove any store expenditures and also remove negative values int the Quantity feature.

```
df=df[df['Quantity']>0]
df=df[df['UnitPrice']>=0]
df.describe()
```

	Quantity	UnitPrice
count	525460.00000	525460.000000
mean	10.68106	3.918228
std	157.39957	36.073270
min	1.00000	0.000000
25%	1.00000	1.250000
50%	4.00000	2.080000
75%	11.00000	4.130000
max	80995.00000	13541.330000

```
print(df.duplicated().sum())
df.drop_duplicates(inplace = True)
```

```
5268
```

We then add a new feature called 'Cost' which stores the total price of that transaction and convert InvoiceDate to the Timestamp and split it into the Year_Month, Month ,Day and Hour features which are self-explanatory.

```
df.insert(loc=5,column='Year_Month',value=df['InvoiceDate'].map
(lambda x:100*x.year+x.month))
df.insert(loc=6,column='Month',value=df.InvoiceDate.dt.month)
df.insert(loc=7,column='Day',value=(df.InvoiceDate.dt.dayofwee
k)+1)
df.insert(loc=8,column='Hour',value=df.InvoiceDate.dt.hour)
```

```
df.head()
```

Date	Year_Month	Month	Day	Hour	UnitPrice	CustomerID	Country	Cost
12- 00	201012	12	3	8	2.55	17850	United Kingdom	15.30
12- 00	201012	12	3	8	3.39	17850	United Kingdom	20.34
12- 00	201012	12	3	8	2.75	17850	United Kingdom	22.00
12- 00	201012	12	3	8	3.39	17850	United Kingdom	20.34
12- 00	201012	12	3	8	3.39	17850	United Kingdom	20.34

Before Cleaning:

	A	B	C	D	E	F	G	H	I
1444	C536543	22355	CHARLOTTE BAG SUKI DESIGN	-2	########	0.85	17841	United Kingdom	
1445	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	########	2.51		United Kingdom	
1446	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	########	2.51		United Kingdom	
1447	536544	21786	POLKADOT RAIN HAT	4	########	0.85		United Kingdom	
1448	536544	21787	RAIN PONCHO RETROSPOT	2	########	1.66		United Kingdom	
1449	536544	21790	VINTAGE SNAP CARDS	9	########	1.66		United Kingdom	
1450	536544	21791	VINTAGE HEADS AND TAILS CARD GAME	2	########	2.51		United Kingdom	
1451	536544	21801	CHRISTMAS TREE DECORATION WITH BELL	10	########	0.43		United Kingdom	
1452	536544	21802	CHRISTMAS TREE HEART DECORATION	9	########	0.43		United Kingdom	
1453	536544	21803	CHRISTMAS TREE STAR DECORATION	11	########	0.43		United Kingdom	
1454	536544	21809	CHRISTMAS HANGING TREE WITH BELL	1	########	2.51		United Kingdom	
1455	536544	21810	CHRISTMAS HANGING STAR WITH BELL	3	########	2.51		United Kingdom	
1456	536544	21811	CHRISTMAS HANGING HEART WITH BELL	1	########	2.51		United Kingdom	
1457	536544	21821	GLITTER STAR GARLAND WITH BELLS	1	########	7.62		United Kingdom	
1458	536544	21822	GLITTER CHRISTMAS TREE WITH BELLS	1	########	4.21		United Kingdom	
1459	536544	21823	PAINTED METAL HEART WITH HOLLY BELL	2	########	2.98		United Kingdom	
1460	536544	21844	RED RETROSPOT MUG	2	########	5.91		United Kingdom	
1461	536544	21851	LILAC DIAMANTE PEN IN GIFT BOX	1	########	4.21		United Kingdom	
1462	536544	21870	I CAN ONLY PLEASE ONE PERSON MUG	1	########	3.36		United Kingdom	
1463	536544	21871	SAVE THE PLANET MUG	5	########	3.36		United Kingdom	
1464	536544	21874	GIN AND TONIC MUG	1	########	3.36		United Kingdom	
1465	536544	21879	HEARTS GIFT TAPE	1	########	1.66		United Kingdom	
1466	536544	21884	CAKES AND BOWS GIFT TAPE	1	########	1.66		United Kingdom	
1467	536544	21888	BINGO SET	1	########	7.62		United Kingdom	
1468	536544	21889	WOODEN BOX OF DOMINOES	2	########	2.51		United Kingdom	
1469	536544	21892	TRADITIONAL WOODEN CATCH CUP GAME	3	########	2.51		United Kingdom	
1470	536544	21894	POTTING SHED SEED ENVELOPES	1	########	2.51		United Kingdom	
1471	536544	21911	GARDEN METAL SIGN	1	########	3.36		United Kingdom	
1472	536544	21912	VINTAGE SNAKES & LADDERS	3	########	7.62		United Kingdom	

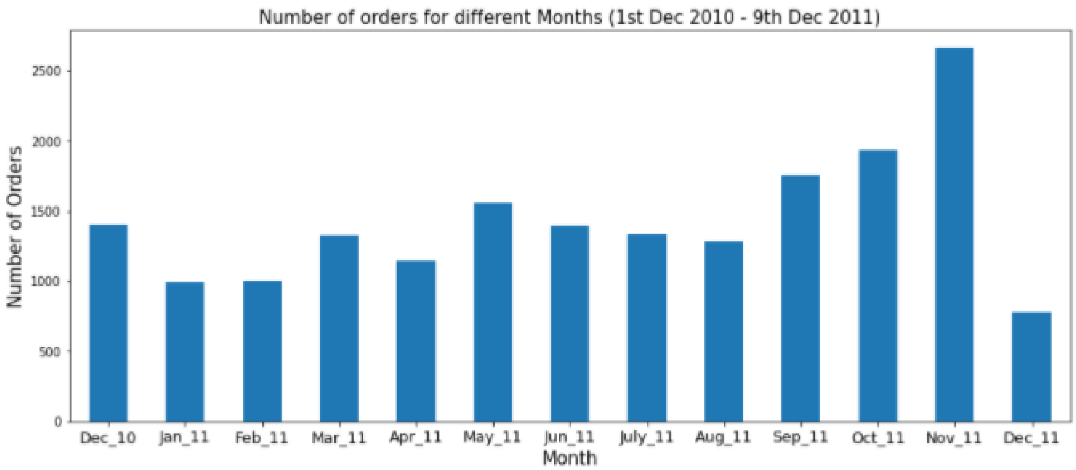
After Cleaning and Preprocessing:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1444	2046	536557	84508A	CAMOUFLAGE DESIGN TEDDY	1	########	201012	12	3	14	2.55	17841	United Kingdom	2.55
1445	2047	536557	22471	TV DINNER TRAY AIR HOSTESS	1	########	201012	12	3	14	4.95	17841	United Kingdom	4.95
1446	2048	536557	21935	SUKI SHOULDER BAG	4	########	201012	12	3	14	1.65	17841	United Kingdom	6.6
1447	2049	536557	21670	BLUE SPOT CERAMIC DRAWER KNOB	6	########	201012	12	3	14	1.25	17841	United Kingdom	7.5
1448	2050	536557	20668	DISCO BALL CHRISTMAS DECORATION	24	########	201012	12	3	14	0.12	17841	United Kingdom	2.88
1449	2051	536557	21672	WHITE SPOT RED CERAMIC DRAWER KNOB	1	########	201012	12	3	14	1.25	17841	United Kingdom	1.25
1450	2052	536557	22553	PLASTERS IN TIN SKULLS	1	########	201012	12	3	14	1.65	17841	United Kingdom	1.65
1451	2053	536557	22041	RECORD FRAME 7" SINGLE SIZE	4	########	201012	12	3	14	2.55	17841	United Kingdom	10.2
1452	2054	536557	20972	PINK CREAM FELT CRAFT TRINKET BOX	2	########	201012	12	3	14	1.25	17841	United Kingdom	2.5
1453	2055	536557	22568	FELTCRAFT CUSHION OWL	1	########	201012	12	3	14	3.75	17841	United Kingdom	3.75
1454	2056	536557	22570	FELTCRAFT CUSHION RABBIT	1	########	201012	12	3	14	3.75	17841	United Kingdom	3.75
1455	2057	536557	22730	ALARM CLOCK BAKELIKE IVORY	1	########	201012	12	3	14	3.75	17841	United Kingdom	3.75
1456	2058	536557	20749	ASSORTED COLOUR MINI CASES	1	########	201012	12	3	14	7.95	17841	United Kingdom	7.95
1457	2059	536557	22785	SQUARECUSHION COVER PINK UNION FLAG	1	########	201012	12	3	14	6.75	17841	United Kingdom	6.75
1458	2060	536557	22786	CUSHION COVER PINK UNION JACK	2	########	201012	12	3	14	5.95	17841	United Kingdom	11.9
1459	2061	536557	85064	CREAM SWEETHEART LETTER RACK	2	########	201012	12	3	14	5.45	17841	United Kingdom	10.9
1460	2062	536557	22212	FOUR HOOK WHITE LOVEBIRDS	1	########	201012	12	3	14	2.1	17841	United Kingdom	2.1
1461	2063	536557	21486	PINK HEART DOTS HOT WATER BOTTLE	2	########	201012	12	3	14	3.75	17841	United Kingdom	7.5
1462	2064	536557	22114	HOT WATER BOTTLE TEA AND SYMPATHY	2	########	201012	12	3	14	3.95	17841	United Kingdom	7.9
1463	2065	536557	21485	RETROSPOT HEART HOT WATER BOTTLE	1	########	201012	12	3	14	4.95	17841	United Kingdom	4.95
1464	2066	536557	84029E	RED WOOLLY HOTTIE WHITE HEART.	1	########	201012	12	3	14	3.75	17841	United Kingdom	3.75
1465	2067	536557	22678	FRENCH BLUE METAL DOOR SIGN 3	3	########	201012	12	3	14	1.25	17841	United Kingdom	3.75
1466	2068	536557	22686	FRENCH BLUE METAL DOOR SIGN No	1	########	201012	12	3	14	1.25	17841	United Kingdom	1.25
1467	2069	536557	22468	BABUSHKA LIGHTS STRING OF 10	1	########	201012	12	3	14	6.75	17841	United Kingdom	6.75
1468	2070	536557	85232B	SET OF 3 BABUSHKA STACKING TINS	1	########	201012	12	3	14	4.95	17841	United Kingdom	4.95
1469	2071	536557	21479	WHITE SKULL HOT WATER BOTTLE	5	########	201012	12	3	14	3.75	17841	United Kingdom	18.75
1470	2072	536557	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	2	########	201012	12	3	14	3.75	17841	United Kingdom	7.5
1471	2073	536557	22837	HOT WATER BOTTLE BABUSHKA	5	########	201012	12	3	14	4.65	17841	United Kingdom	23.25
1472	2074	536557	22112	CHOCOLATE HOT WATER BOTTLE	2	########	201012	12	3	14	4.95	17841	United Kingdom	9.9

5. Exploratory Data Analysis:

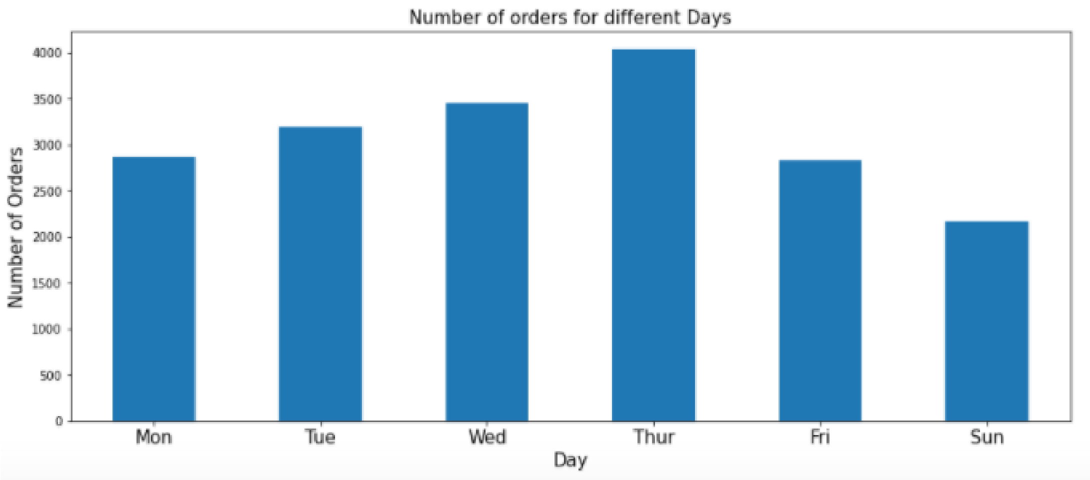
Shown below are the insights obtained from the EDA on the dataset along with suitable visualizations.

```
ax = df.groupby('InvoiceNo')['Year_Month'].unique().value_count
s().sort_index().plot(kind='bar',color=color[0],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 20
10 - 9th Dec 2011)',fontsize=15)
ax.set_xticklabels(('Dec_10','Jan_11','Feb_11','Mar_11','Apr_1
1','May_11','Jun_11','July_11','Aug_11','Sep_11','Oct_11','Nov_
11','Dec_11'), rotation='horizontal', fontsize=13)
plt.show()
```

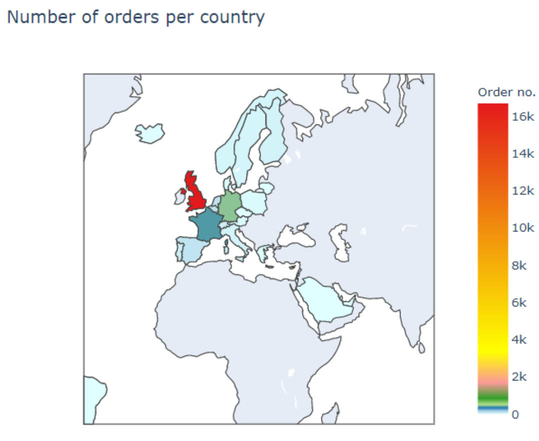


From the graph showing orders for various months, we can see that sales increase massively during the November of 2011 which is inline with the holiday season approaching in Western Countries with festivals like Christmas and Thanksgiving coming up.

```
ax = df.groupby('InvoiceNo')['Day'].unique().value_counts().sor
t_index().plot(kind='bar',color=color[0],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'), rota
tion='horizontal', fontsize=15)
plt.show()
```



By plotting the number of orders on different days, we are able to infer that the maximum sales happens on a Thursday with the online store not functioning on a Saturday.

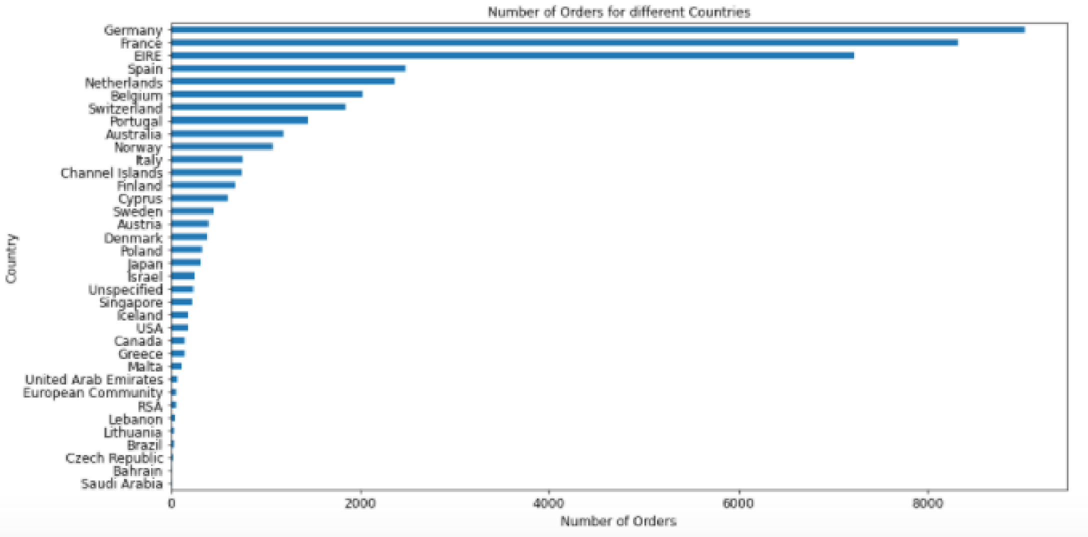



```
from plotly.offline import init_notebook_mode,iplot
temp = df[['CustomerID', 'InvoiceNo', 'Country']].groupby(['Custome
rID', 'InvoiceNo', 'Country']).count()
temp = temp.reset_index(drop = False)
countries = temp['Country'].value_counts()
print('No. of countries in the dataframe: {}'.format(len(countrie
s)))


data = dict(type='choropleth',
locations = countries.index,
locationmode = 'country names', z = countries,
text = countries.index, colorbar = {'title':'Order no.'},
colorscale=[[0, 'rgb(224,255,255)'],
              [0.01, 'rgb(166,206,227)'], [0.02, 'rgb(31,120,180)'],
              [0.03, 'rgb(178,223,138)'], [0.05, 'rgb(51,160,44)'],
              [0.10, 'rgb(251,154,153)'], [0.20, 'rgb(255,255,0)'],
              [1, 'rgb(227,26,28)']],
reversescale = False)
layout = dict(title='Number of orders per country',
geo = dict(showframe = True, projection={'type':'mercator'}))
choromap = go.Figure(data = [data], layout = layout)
iplot(choromap, validate=False)
```

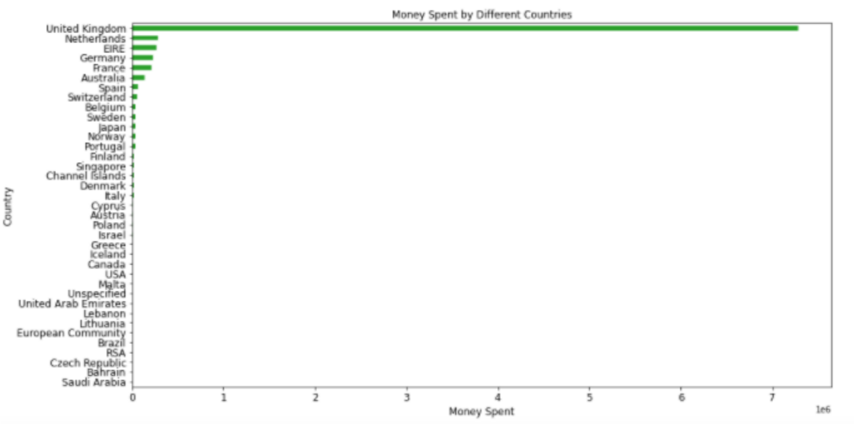
The above plot shows the number of unique orders per country with the United Kingdom dominating sales with Germany and France in the 2nd and 3rd positions.

```
del dfcountry['United Kingdom']
plt.subplots(figsize=(15,8))
dfcountry.plot(kind='barh', fontsize=12, color=color[0])
plt.xlabel('Number of Orders', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Number of Orders for different Countries', fontsize=12)
plt.show()
```



Here we plot the orders per country without UK to get a better view of the purchases made by other countries without the graph becoming unreadable because of the UK.

```
dfcountryrev=df.groupby('Country')['Cost'].sum().sort_values()
plt.subplots(figsize=(15,8))
dfcountryrev.plot(kind='barh', fontsize=12, color=color[2])
plt.xlabel('Money Spent', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Money Spent by Different Countries', fontsize=12)
plt.show()
```



Plotting the expenditures by various countries shows that the UK far and away dominates sales with Netherlands and EIRE surprisingly beating out Germany and France despite the latter 2 countries having more orders indicating average purchase price is higher.

```
df[df['Country']=='Germany']['Cost'].mean()
```

```
25.332712972194354
```

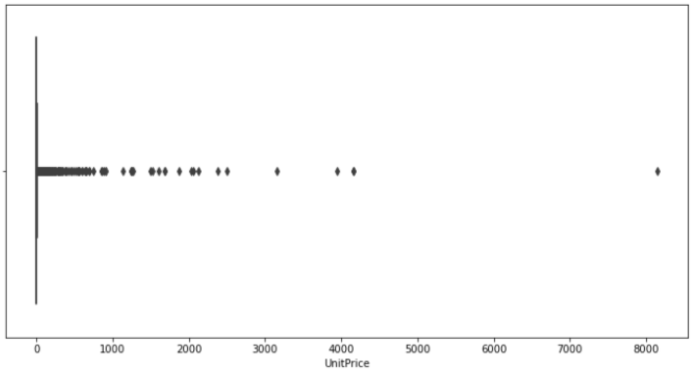
```
df[df['Country']=='Netherlands']['Cost'].mean()
```

```
120.7982818451216
```

```python
df['UnitPrice'].describe()
```

```
count    392732.000000
mean          3.125596
std          22.240725
min           0.000000
25%           1.250000
50%           1.950000
75%           3.750000
max        8142.750000
Name: UnitPrice, dtype: float64
```

```python
plt.subplots(figsize=(12,6))
sns.boxplot(df.UnitPrice)
plt.show()
```



Analysing UnitPrice indicates that most products are extremely inexpensive with many of them being in the 0-10 price range. Though the boxplot may not tell us much about the average price, it does indicate that there are outlier prices that are well into the thousands.

```python
maxdf=df.sort_values(by='UnitPrice',ascending=False)
maxdf.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Year_Month	Month	Day	Hour	UnitPrice
173382	551697	POST	POSTAGE	1	2011-05-03 13:46:00	201105	5	2	13	8142.75
422376	573080	M	Manual	1	2011-10-27 14:20:00	201110	10	4	14	4161.06
422351	573077	M	Manual	1	2011-10-27 14:13:00	201110	10	4	14	4161.06
406406	571751	M	Manual	1	2011-10-19 11:18:00	201110	10	3	11	3949.32
374542	569382	M	Manual	1	2011-10-03 16:44:00	201110	10	1	16	3155.95

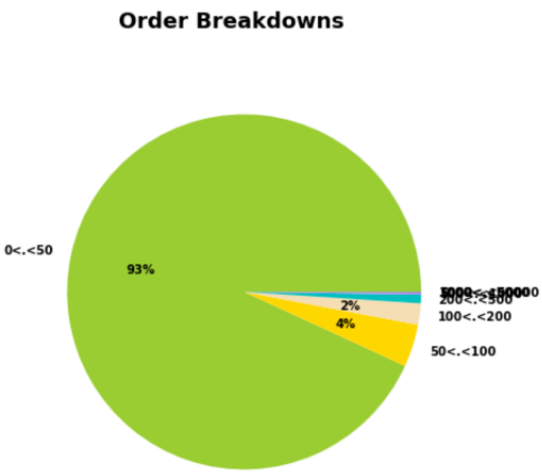
On sorting the dataset by descending order of UnitPrice, we see that the most expensive items are postage costs for shipping items and ‘manual’ costs which aren’t actual items so we try to ignore it.

```python
codes=df[df['StockCode'].str.contains('^[a-zA-Z]+',regex=True)]['StockCode'].unique()
maxdf=maxdf[~(maxdf['StockCode'].isin(codes))]
maxdf
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Year_Month	Month	Day	Hour	UnitPric
222682	556446	22502	PICNIC BASKET WICKER 60 PIECES	1	2011-06-10 15:33:00	201106	6	5	15	649.5
222680	556444	22502	PICNIC BASKET WICKER 60 PIECES	60	2011-06-10 15:28:00	201106	6	5	15	649.5
171178	551393	22656	VINTAGE BLUE KITCHEN CABINET	1	2011-04-28 12:22:00	201104	4	4	12	295.0
32484	539080	22655	VINTAGE RED KITCHEN CABINET	1	2010-12-16 08:41:00	201012	12	4	8	295.0
51636	540647	22655	VINTAGE RED KITCHEN CABINET	1	2011-01-10 14:57:00	201101	1	1	14	295.0

Upon eliminating the misc. transactions, we find out that the most expensive items are actually only around 650$ with the next item only costing 295$.

```python
dftemp=df[df['Cost']>0]
price_range = [0, 50, 100, 200, 500, 1000, 5000, 50000]
count_price = []
for i, price in enumerate(price_range):
    if i == 0: continue
    val = dftemp[(dftemp['Cost'] < price) &
                   (dftemp['Cost'] > price_range[i-1])]['Cost'].count()
    count_price.append(val)
plt.rc('font', weight='bold')
f, ax = plt.subplots(figsize=(11, 6))
colors = ['yellowgreen', 'gold', 'wheat', 'c', 'violet', 'royalblue','firebrick']
labels = [ '{}<.<{}'.format(price_range[i-1], s) for i,s in enumerate(price_range) if i !=
0]
sizes  = count_price
explode = [0.0 if sizes[i] < 100 else 0.0 for i in range(len(sizes))]
ax.pie(sizes, explode = explode, labels=labels, colors = colors,
       autopct = lambda x:'{:1.0f}%'.format(x) if x > 1 else '',
       shadow = False, startangle=0)
ax.axis('equal')
f.text(0.5, 1.01, "Order Breakdowns", ha='center', fontsize = 18);
```



Upon plotting the pie chart of the total cost of each transaction, we notice that actually 90% of all transactions are only in the 0-50$ range with an unnoticeable percentage of purchases actually even crossing the 100$ mark.

Earlier, we noticed that the minimum value for the UnitPrice feature was 0.0 which means the store also gives out items for free.

```
freeitems=df[df['UnitPrice']==0]
freeitems.head()
```

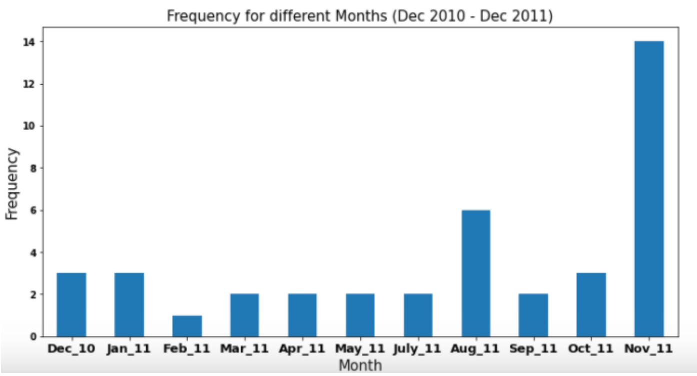
InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Year_Month	Month	Day	Hour	UnitPrice	CustomerID
537197	22841	ROUND CAKE TIN VINTAGE GREEN	1	2010-12-05 14:02:00	201012	12	7	14	0.0	12647
539263	22580	ADVENT CALENDAR GINGHAM SACK	4	2010-12-16 14:36:00	201012	12	4	14	0.0	16560
539722	22423	REGENCY CAKESTAND 3 TIER	10	2010-12-21 13:45:00	201012	12	2	13	0.0	14911
540372	22090	PAPER BUNTING RETROSPOT	24	2011-01-06 16:41:00	201101	1	4	16	0.0	13081
540372	22553	PLASTERS IN TIN SKULLS	24	2011-01-06 16:41:00	201101	1	4	16	0.0	13081

```
freeitems.Year_Month.value_counts().mean()
```

```
3.6363636363636362
```

We also notice that the average amount of free items given out per month is around 3.6 so rounded up to 4 items.

```
ax = freeitems.Year_Month.value_counts().sort_index().plot(kind
='bar',figsize=(12,6), color=color[0])
ax.set_xlabel('Month',fontsize=15)
ax.set_ylabel('Frequency',fontsize=15)
ax.set_title('Frequency for different Months (Dec 2010 - Dec 20
11)',fontsize=15)
ax.set_xticklabels(('Dec_10','Jan_11','Feb_11','Mar_11','Apr_1
1','May_11','July_11','Aug_11','Sep_11','Oct_11','Nov_11'), rot
ation='horizontal', fontsize=13)
plt.show()
```

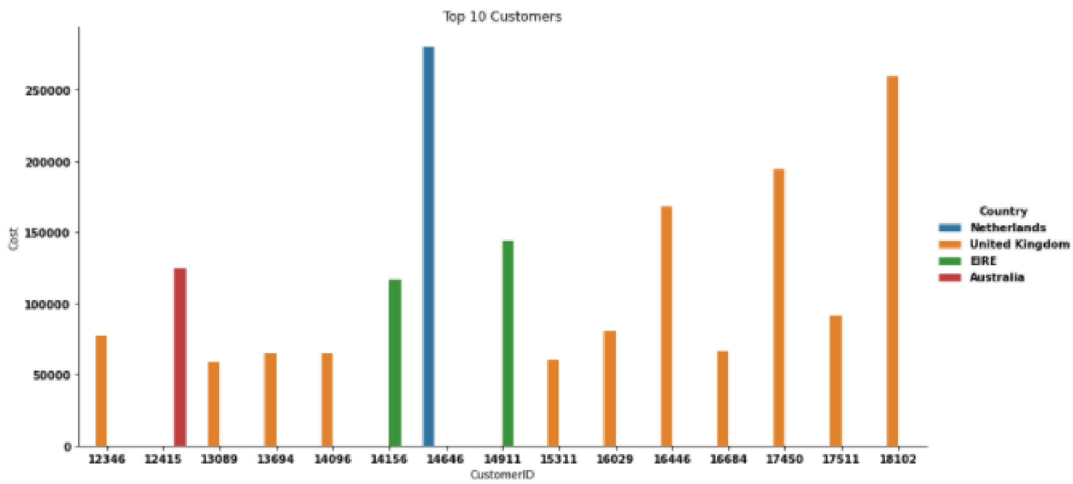


From the above graph, we notice that the number of free items given out in November 2011 is quite high which indicates that a festival/holiday promotional sale was held then due to the holiday season coming up.

```
spenddf=df.groupby(by=['CustomerID','Country'],as_index=False)['Cost'].sum().sort_values(b
y='Cost',ascending=False).iloc[0:15]
print(spenddf)
sns.catplot(data=spenddf,x='CustomerID',y='Cost',hue='Country',kind='bar',height=6,aspect=
2)
plt.title('Top 10 Customers')
```

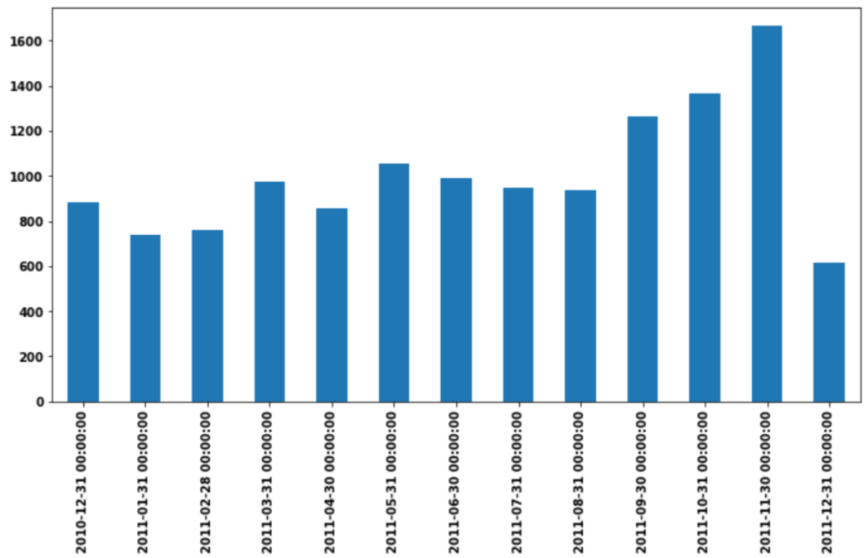
```
      CustomerID         Country       Cost
1698       14646     Netherlands  280206.02
4210       18102  United Kingdom  259657.30
3737       17450  United Kingdom  194390.79
3017       16446  United Kingdom  168472.50
1888       14911            EIRE  143711.17
57         12415       Australia  124914.53
1342       14156            EIRE  117210.08
3780       17511  United Kingdom   91062.38
2711       16029  United Kingdom   80850.84
0          12346  United Kingdom   77183.60
3185       16684  United Kingdom   66653.56
1298       14096  United Kingdom   65164.79
1005       13694  United Kingdom   65039.62
2185       15311  United Kingdom   60632.75
570        13089  United Kingdom   58762.08

Text(0.5, 1.0, 'Top 10 Customers')
```



Here, we plot the top **15** highest spending customers (not 10, error). We notice that while majority of the spenders are from the UK, the highest spender was actually from the Netherlands along with customers from the EIRE even joining the list.

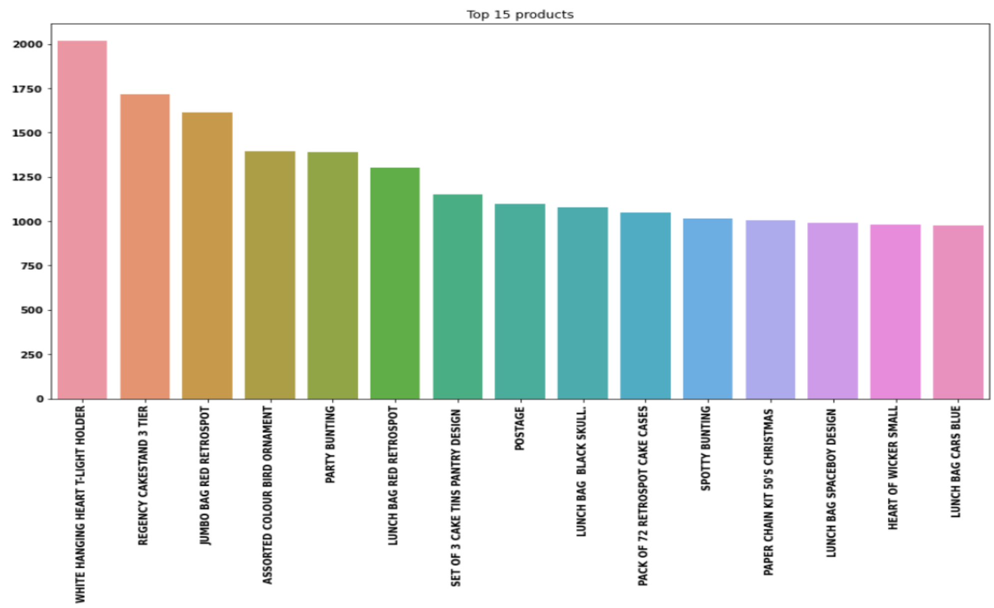
```
dfunique=df.set_index('InvoiceDate')['CustomerID'].resample('M').nunique().plot(kind='ba
r',figsize=(12,6), color=color[0])
ax.set_xlabel('Month',fontsize=15)
ax.set_ylabel('Number of customers',fontsize=15)
ax.set_title('Number of unique customers per month',fontsize=15)
plt.show()
```



Plotting the number of unique customers per month shows that there were substantially more customers ,again, in the month of November 2011, which coincides with the promotional sale that the store was having at the same time, proving that the marketing strategy behind the sale was a success.

```
y=df.Description.value_counts().sort_values(ascending=False).iloc[0:15]
plt.figure(figsize=(15,8))
sns.barplot(y=y.values,x=y.index)
plt.xticks(rotation=90)
plt.title("Top 15 products")
```

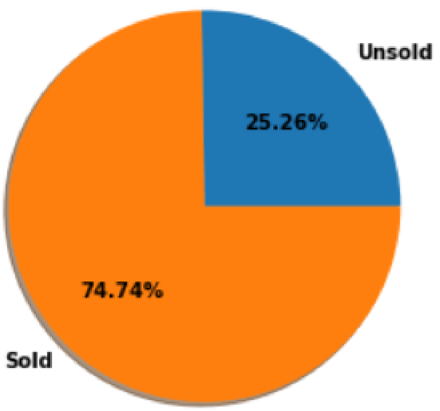
```
Text(0.5, 1.0, 'Top 15 products')
```



Above is the plot for the top 15 most popular products purchased off the online store. It is also noted that Postage is present in the graph.

```python
from wordcloud import WordCloud, STOPWORDS
item_words=''
stopwords=set(STOPWORDS)
for val in df.Description:
    val=str(val)
    tokens=val.split()
    for i in range(len(tokens)):
        tokens[i]=tokens[i].lower()
    item_words+=" ".join(tokens)+" "
wordcloud=WordCloud(width=800,height=800,background_color='white',stopwords=stopwords,min_
font_size=10,collocations=False).generate(item_words)
plt.figure(figsize=(8,8),facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
```

Also present in the EDA is a word cloud containing the most used keywords in the product description. We notice that people tend to purchase a lot of bags and items with hearts on then as well as many item sets. The words ‘red’ and ‘christmas’ also indicate the many shoppers tend to purchase Christmas presents and decorations from the store.




```python
unsoldno=df2[df2['CustomerID']=='Unsold'].shape[0]
totalitems=df2.shape[0]
percunsold=unsoldno/totalitems*100
print("No. of Unsold items=",unsoldno)
print("Total No. of items=",totalitems)
print("Percentage of Unsold items=",percunsold,'%')
fig1,ax1=plt.subplots()
ax1.pie([unsoldno,totalitems-unsoldno],labels=['Unsold','Sol
d'],autopct='%1.2f%%',shadow=True)
ax1.axis('equal')
plt.show()
```

```
No. of Unsold items= 132728
Total No. of items= 525460
Percentage of Unsold items= 25.25939177101968 %
```

We also count the percentage of unsold items left in the inventory which adds up to a whopping 132728 items but only accounts for 25% of the total item count.

```
codes=df[df['StockCode'].str.contains('^[a-zA-Z]+',regex=True)]
['StockCode'].unique()
print(codes)
for i in codes:
 print("{:<15} = {:<15}".format(i,df[df['StockCode']==i]['Descr
iption'].unique()[0]))
```

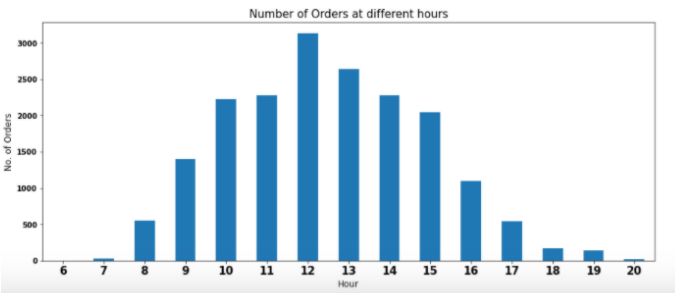
```
['POST' 'C2' 'M' 'BANK CHARGES' 'PADS' 'DOT']
POST            = POSTAGE
C2              = CARRIAGE
M               = Manual
BANK CHARGES    = Bank Charges
PADS            = PADS TO MATCH ALL CUSHIONS
DOT             = DOTCOM POSTAGE
```

Finally, we note the items having unusual stock codes which indicate the additional services that the store charges for.

6. Hypothesis Testing:

The first hypothesis test we perform is by taking a sample of entries from the dataset which correspond to purchases made by customers in the UK and perform a 2-tailed hypothesis test to check whether the sample mean purchase time is equal to the population mean i.e all the purchases from all over the world.

```
ax=df.groupby('InvoiceNo')['Hour'].unique().value_counts().iloc
[:-1].sort_index().plot(kind='bar',color=color[0],figsize=(15,
6))
ax.set_xlabel('Hour',fontsize=12)
ax.set_ylabel('No. of Orders',fontsize=12)
ax.set_title('Number of Orders at different hours',fontsize=15)
ax.set_xticklabels(range(6,21),rotation='horizontal',fontsize=1
5)
plt.show()
```



We notice that since the number of orders vs time graph for the population is more or less normally distributed, we need not perform any normalization.

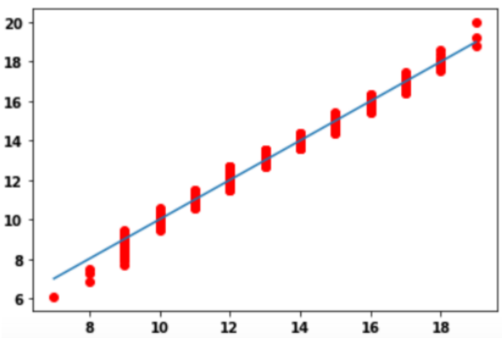
```
sample_df=df[df['Country']=='United Kingdom'].sample(n=500,rand
om_state=10)
sample_df.describe()
```

_Month	Month	Day	Hour	UnitPrice	Cost
000000	500.000000	500.000000	500.000000	500.000000	500.000000
01.020000	7.620000	3.664000	13.034000	3.409260	18.580120
12826	3.475595	1.928352	2.249214	9.284096	30.342029
12.000000	1.000000	1.000000	7.000000	0.120000	0.210000
04.000000	5.000000	2.000000	12.000000	1.040000	5.000000
07.000000	8.000000	4.000000	13.000000	1.950000	10.500000
10.000000	11.000000	5.000000	15.000000	3.750000	19.575000
12.000000	12.000000	7.000000	19.000000	195.000000	250.000000

A simple view of the sample taken

```
from scipy.stats import norm
def normality_check(data):
    pos =[]
    th_Q =[]
    data = np.sort(np.array(data))
    pos = [(i - 0.5)/len(data) for i in range(1, len(data)+1)]
    th_Q = [norm.ppf(i, np.mean(data), np.std(data, ddof = 1))
for i in pos]
    plt.plot(data, th_Q, 'ro', data, data)
    plt.show()

normality_check(sample_df["Hour"])
```



From the above function, we can confirm that the Hour feature is normally distributed.

```
from math import sqrt
time_mean=sample_df['Hour'].mean()
time_var=sample_df['Hour'].var()
time_sd=sqrt(time_var)
time_sampsize=sample_df['Hour'].shape[0]
print('Sample mean=',time_mean)
print('Population mean=',df['Hour'].mean())
print('Sample variance=',time_var)
print('Sample std deviation=',time_sd)
print('Sample size=',time_sampsize)
```

```
Sample mean= 13.034
Population mean= 12.721578582850391
Sample variance= 5.058961923847695
Sample std deviation= 2.249213623435465
Sample size= 500
```

We now perform a 2-tailed hypothesis test at a confidence level of 95%.

```
from scipy.stats import norm

def two_sided_hypo(sample_mean, pop_mean, std_dev, sample_size, alpha):
    actual_z = abs(norm.ppf(alpha/2))
    hypo_z = (sample_mean - pop_mean) / (std_dev/sqrt(sample_size))
    print('actual z value :', actual_z)
    print('hypothesis z value :', hypo_z, '\n')
    if hypo_z >= actual_z or hypo_z <= -(actual_z):
        return True
    else:
        return False


alpha = 0.05
sample_mean = time_mean
pop_mean = df['Hour'].mean()
sample_size =  time_sampsize
std_dev = time_sd

print('H0 : μ =', pop_mean)
print('H1 : μ !=', pop_mean)
print('alpha value is :', alpha, '\n')

reject = two_sided_hypo(sample_mean, pop_mean, std_dev, sample_size, alpha)
if reject:
    print('Reject NULL hypothesis')
else:
    print('Failed to reject NULL hypothesis')
```

```
H0 : μ = 12.721578582850391
H1 : μ != 12.721578582850391
alpha value is : 0.05

actual z value : 1.959963984540054
hypothesis z value : 3.105954539374992

Reject NULL hypothesis
```

The Z value returned is 3.106 which is greater than 1.96 therefore, we reject the Null Hypothesis that states that the sample mean purchase time is equal to the population mean purchase time.

For the 2nd Hypothesis test, we take a sample from countries other than the UK and perform a 1-tail test to check if the mean purchase time is less than a projected mean of 1200 hours(or 12pm).

```
sample2=df[df['Country']!='United Kingdom']
sample2.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	Year_Month	Month	Day	Hour	UnitPrice	CustomerID
26	536370	22728	ALARM CLOCK BAKELIKE PINK	24	2010-12-01 08:45:00	201012	12	3	8	3.75	12583
27	536370	22727	ALARM CLOCK BAKELIKE RED	24	2010-12-01 08:45:00	201012	12	3	8	3.75	12583
28	536370	22726	ALARM CLOCK BAKELIKE GREEN	12	2010-12-01 08:45:00	201012	12	3	8	3.75	12583
29	536370	21724	PANDA AND BUNNIES STICKER SHEET	12	2010-12-01 08:45:00	201012	12	3	8	0.85	12583
30	536370	21883	STARS GIFT TAPE	24	2010-12-01 08:45:00	201012	12	3	8	0.65	12583

```
def one_sided_hypo(sample_mean, pop_mean, std_dev, sample_size, alpha):
    actual_z = abs(norm.ppf(alpha))
    hypo_z = (sample_mean - pop_mean) / (std_dev/sqrt(sample_size))
    print('actual z value :', actual_z)
    print('hypothesis z value :', hypo_z, '\n')
    if hypo_z >= actual_z:
        return True
    else:
        return False

alpha = 0.05
sample_mean = sample2['Hour'].mean()
pop_mean = 12
sample_size =  sample2.shape[0]
std_dev = sqrt(sample2['Hour'].var())

print('H0 : μ <=', pop_mean)
print('H1 : μ >', pop_mean)
print('alpha value is :', alpha, '\n')

reject = one_sided_hypo(sample_mean, pop_mean, std_dev, sample_size, alpha)
if reject:
    print('Reject NULL hypothesis')
else:
    print('Failed to reject NULL hypothesis')
```

```
sample2.describe()
```

	Quantity	Year_Month	Month	Day	Hour	UnitPrice	Cost
count	43505.000000	43505.000000	43505.000000	43505.000000	43505.000000	43505.000000	43505.000000
mean	20.959637	201102.246133	7.335203	3.391403	12.062338	4.357557	36.827589
std	48.027529	21.143334	3.391330	1.670226	2.408614	43.193024	87.313143
min	1.000000	201012.000000	1.000000	1.000000	7.000000	0.000000	0.000000
25%	6.000000	201104.000000	5.000000	2.000000	10.000000	1.250000	13.200000
50%	12.000000	201107.000000	8.000000	3.000000	12.000000	1.950000	17.700000
75%	18.000000	201110.000000	10.000000	5.000000	14.000000	3.750000	30.000000
max	2400.000000	201112.000000	12.000000	7.000000	19.000000	4161.060000	4992.000000

```
H0 : μ <= 12
H1 : μ > 12
alpha value is : 0.05

actual z value : 1.6448536269514729
hypothesis z value : 5.398249774201496

Reject NULL hypothesis
```

The Z value here is far greater than the actual Z value, therefore the Null Hypothesis that states that the mean purchase time is less than or equal to 12 is rejected.

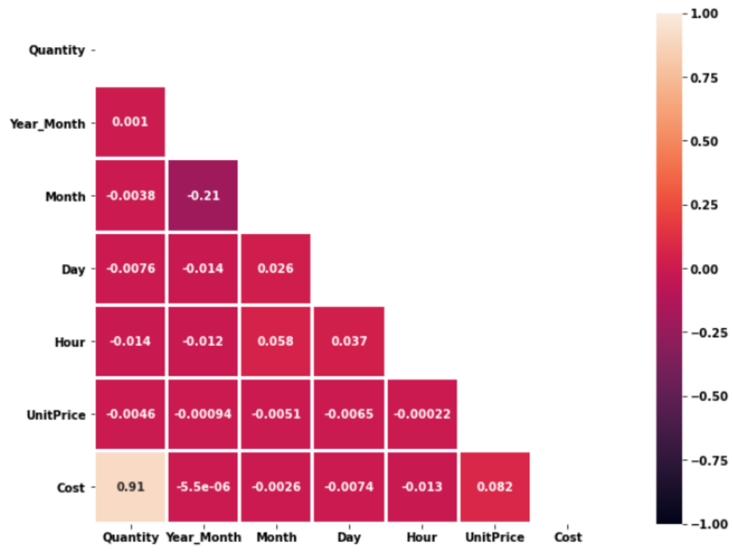
7. Correlation:

We also perform a few correlation tests to see how each feature depends on the other.

```
dftemp2=df.loc[(df['Quantity']!=0) & (df['UnitPrice']!=0)]
dfcorr=df.corr(method="pearson")
dfcorr
```

```
matrix=np.triu(dfcorr)
plt.figure(figsize=(16,8))
sns.heatmap(dfcorr,annot=True,square=True,mask=matrix,linewidths=2,vmin=-1,vmax=1)
```

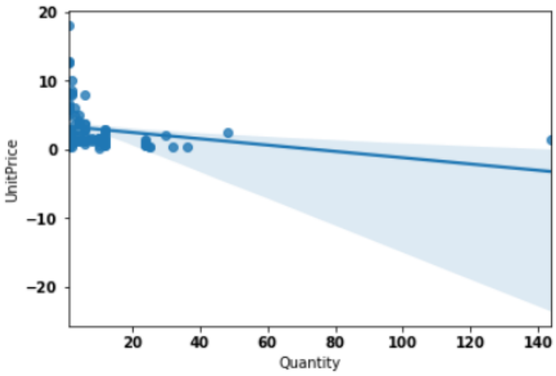
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5a6553d750>
```



The correlation matrix plotted above shows that the total cost of a transaction actually depends a lot more on the quantity of the item being purchased rather than the actual price of the single item.

```
sns.regplot(x='Quantity',y='UnitPrice',data=dfcross.sample(n=100))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5a633e8550>
```



We also show a regression plot taking a random sample from the dataset and observe that there is a very slight negative correlation between the quantity being purchased and the price of an item which seems to indicate that the cost of an individual item usually doesn’t play a big role in determining whether a customer will buy multiple items or not.

```
stat,p,dof,expected=scipy.stats.chi2_contingency(dfcross)
p
```

```
0.0
```

```
alpha=0.05
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (fail to reject H0)')
```

```
Dependent (reject H0)
```

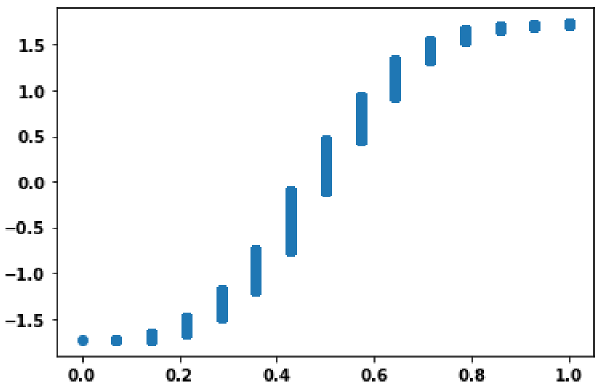
This conclusion is further reinforced by performing a chi- squared test on the dataset which proves that the 2 features, Quantity and UnitPrice, no matter how small the correlation may be, are actually dependent.

```python
p=[]
x=list(dftest['Hour'])
x=[i for i in x if i!=0]
x.sort()
l=len(x)
min1,max1=min(x),max(x)
norm1=lambda x:(x-min1)/(max1-min1)
x=[norm1(i) for i in x]
pi=lambda i,n:(i-0.5)/n
n=len(x)
p=[pi(i,n) for i in range(1,l+1)]
```

```python
Zscores=scipy.stats.zscore(p)
```

```python
plt.scatter(x,Zscores)
```

```
<matplotlib.collections.PathCollection at 0x7
```



As a small example of normalization, we take a data sample of purchases made in the UK(dftest) and normalize the purchase times using the min-max normalization technique.

We then compute the Z-scores and plot it to obtain a normal probability plot.

8. Results and Discussion:

From the above analyses, we can conclude that since the store is based in the UK, most of the purchases come from the UK with the neighbouring countries, France, Germany and Netherlands following it. We also reach the conclusion that having promotional sales and giving out free items helps attract new customers. We notice that a lot of sales also occur just before the holiday season begins in the West with maximum sales happening in the November of 2011 with peak sales occuring on Thursdays around 12-1pm. Christmas related items such as red items and bags also seem to be incredibly popular. Most products are generally affordable with very few items crossing the 100$ mark. In conclusion, the most effective way of attracting customers is to incentivize purchases with additional promotions which help market the store better bringing and attracting more new customers.