Importing required libraries

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

Loading the DataSet

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

In [3]:
```
# ecom_data = pd.read_csv('data.csv', encoding = 'ISO-8859-1')
```

Number of rows and columns

In [4]:
```
df.shape
```

Out[4]: (541909, 8)

Access first five rows

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

Out[5]:

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

Access last five rows

In [6]:
```
df.tail()
```

Out[6]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Custor
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12

Displaying total columns from Dataset

In [7]:
```
df.columns
```

Out[7]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDat
e',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

Getting all columns one by one

In [8]:
```
for column in df.columns:
    print(column)
```

```
InvoiceNo
StockCode
Description
Quantity
InvoiceDate
UnitPrice
CustomerID
Country
```

Renaming columns names

In [9]:
```
d = {
        'InvoiceNo': 'invoice_num',
```

```
        'StockCode' : 'stock_code',
        'Description' : 'description',
        'Quantity' : 'quantity',
        'InvoiceDate' : 'invoice_date',
        'UnitPrice' : 'unit_price',
        'CustomerID' : 'cust_id',
        'Country' : 'country'
}
```

In [10]:
```
d
```

Out[10]:
```
{'InvoiceNo': 'invoice_num',
 'StockCode': 'stock_code',
 'Description': 'description',
 'Quantity': 'quantity',
 'InvoiceDate': 'invoice_date',
 'UnitPrice': 'unit_price',
 'CustomerID': 'cust_id',
 'Country': 'country'}
```

In [11]:
```
df.rename(columns = d, inplace = True)
```

After changing column names Checking new column names

In [12]:
```
df.columns
```

Out[12]:
```
Index(['invoice_num', 'stock_code', 'description', 'quantity', 'invoice_
date',
       'unit_price', 'cust_id', 'country'],
      dtype='object')
```

In [13]:
```
for i in df.columns:
    print(i)
```

```
invoice_num
stock_code
description
quantity
invoice_date
unit_price
cust_id
country
```

Lets check initial data

In [14]:
```
df.head()
```

Out[14]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0

Checking first five rows

In [15]:
```
df.head()
```

Out[15]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0

Data Cleaning

Checking column types

In [16]:
```
df.dtypes
```

Out[16]:
```
invoice_num      object
stock_code       object
description      object
quantity          int64
invoice_date     object
unit_price      float64
cust_id         float64
country          object
dtype: object
```

DataFrame information

In [17]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   invoice_num   541909 non-null  object 
 1   stock_code    541909 non-null  object 
 2   description   540455 non-null  object 
 3   quantity      541909 non-null  int64  
 4   invoice_date  541909 non-null  object 
 5   unit_price    541909 non-null  float64
 6   cust_id       406829 non-null  float64
 7   country       541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

Checking missing values for each column

In [18]:
```
df.isnull()
```

Out[18]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cus
0	False	False	False	False	False	False	F
1	False	False	False	False	False	False	F
2	False	False	False	False	False	False	F
3	False	False	False	False	False	False	F
4	False	False	False	False	False	False	F
...	...	...	...	...	...	...	
541904	False	False	False	False	False	False	F
541905	False	False	False	False	False	False	F
541906	False	False	False	False	False	False	F
541907	False	False	False	False	False	False	F
541908	False	False	False	False	False	False	F

541909 rows × 8 columns

Checkcing number of columns

In [19]:
```
len(df.columns)
```

Out[19]: 8

In [20]:
```
df.shape
```

Out[20]: (541909, 8)

Checking missing values count on each column

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
invoice_num          0
stock_code           0
description       1454
quantity             0
invoice_date         0
unit_price           0
cust_id         135080
country              0
dtype: int64
```

Checking missing values count on each column, applying sorting

In [22]:
```
df.isnull().sum().sort_values()
```

Out[22]:
```
invoice_num          0
stock_code           0
quantity             0
invoice_date         0
unit_price           0
country              0
description       1454
cust_id         135080
dtype: int64
```

In [23]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[23]:
```
cust_id         135080
description       1454
invoice_num          0
stock_code           0
quantity             0
invoice_date         0
unit_price           0
country              0
dtype: int64
```

Checking type of invoice_date column

In [24]:
```
df.dtypes
```

Out[24]:
```
invoice_num      object
stock_code       object
description      object
quantity          int64
invoice_date     object
unit_price      float64
cust_id         float64
country          object
dtype: object
```

Access intial data

In [25]:
```
df.head(2)
```

Out[25]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0

Converting invoice_date data type into datatime data type

In [26]:
```
df['invoice_date'] = pd.to_datetime(df.invoice_date, format='%m/%d/%Y %H:
```

Checking type of invoice_date

In [27]:
```
df.dtypes
```

Out[27]:
```
invoice_num             object
stock_code              object
description             object
quantity                 int64
invoice_date    datetime64[ns]
unit_price             float64
cust_id                float64
country                 object
dtype: object
```

In [28]:
```
df.head()
```

Out[28]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0

Let us check description column

In [29]:
```
df.description
```

Out[29]:
```
0          WHITE HANGING HEART T-LIGHT HOLDER
1                        WHITE METAL LANTERN
2             CREAM CUPID HEARTS COAT HANGER
3        KNITTED UNION FLAG HOT WATER BOTTLE
4             RED WOOLLY HOTTIE WHITE HEART.
                         ...
541904            PACK OF 20 SPACEBOY NAPKINS
541905          CHILDREN'S APRON DOLLY GIRL
541906          CHILDRENS CUTLERY DOLLY GIRL
541907       CHILDRENS CUTLERY CIRCUS PARADE
541908          BAKING SET 9 PIECE RETROSPOT
Name: description, Length: 541909, dtype: object
```

We need to call lower() method

In [30]:
```
df.description.str.lower()
```

Out[30]:
```
0          white hanging heart t-light holder
1                        white metal lantern
2             cream cupid hearts coat hanger
3        knitted union flag hot water bottle
4             red woolly hottie white heart.
                         ...
541904            pack of 20 spaceboy napkins
541905          children's apron dolly girl
541906          childrens cutlery dolly girl
541907       childrens cutlery circus parade
541908          baking set 9 piece retrospot
Name: description, Length: 541909, dtype: object
```

In [31]:
```
df.head(3)
```

Out[31]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0

In [32]:
```
df['description'] = df.description.str.lower()
```

In [33]:
```
df.head()
```

Out[33]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	white hanging heart t-light holder	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850.0
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850.0
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850.0

Missing values

Based on team meeting/client discussion we will need to perform accordingly

In [34]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[34]:
```
cust_id         135080
description       1454
invoice_num          0
stock_code           0
quantity             0
invoice_date         0
unit_price           0
country              0
dtype: int64
```

Dropping missing values

In [35]:
```
df_new = df.dropna()
```

After dropping missing values then again Checking missing values for each columns

In [36]:
```
df_new.isnull().sum()
```

Out[36]:
```
invoice_num     0
stock_code      0
description     0
quantity        0
invoice_date    0
unit_price      0
cust_id         0
country         0
dtype: int64
```

DataFrame information

In [37]:
```
df_new.info()
```

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

In [38]:
```
df_new.head()
```

Out[38]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	white hanging heart t-light holder	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850.0
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850.0
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850.0

Check type of cust_id data type

In [39]:
```
df_new.dtypes
```

Out[39]:
```
invoice_num              object
stock_code               object
description              object
quantity                  int64
invoice_date    datetime64[ns]
unit_price             float64
cust_id                float64
country                 object
dtype: object
```

Converting cust_id float type into integer type

In [40]:
```
df_new['cust_id']
```

Out[40]:
```
0         17850.0
1         17850.0
2         17850.0
3         17850.0
4         17850.0
           ...
541904    12680.0
541905    12680.0
541906    12680.0
541907    12680.0
541908    12680.0
Name: cust_id, Length: 406829, dtype: float64
```

Ignoring warnings in jupyter

In [41]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [42]:
```
df_new['cust_id'] = df_new['cust_id'].astype('int64')
```

Accessing first five rows

In [43]:
```
df_new.head()
```

Out[43]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	white hanging heart t-light holder	6	2010-12-01 08:26:00	2.55	17850
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850

New DataFrame information

In [44]:
```
df_new.info()
```

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

DataFrame description

In [45]:
```
df_new.describe()
```

Out[45]:

	quantity	invoice_date	unit_price	cust_id
count	406829.000000	406829	406829.000000	406829.000000
mean	12.061303	2011-07-10 16:30:57.879207424	3.460471	15287.690570
min	-80995.000000	2010-12-01 08:26:00	0.000000	12346.000000
25%	2.000000	2011-04-06 15:02:00	1.250000	13953.000000
50%	5.000000	2011-07-31 11:48:00	1.950000	15152.000000
75%	12.000000	2011-10-20 13:06:00	3.750000	16791.000000
max	80995.000000	2011-12-09 12:50:00	38970.000000	18287.000000
std	248.693370	NaN	69.315162	1713.600303

Rounding the values in DataFrame

In [46]:
```
df_new.describe().round(2)
```

Out[46]:

	quantity	invoice_date	unit_price	cust_id
count	406829.00	406829	406829.00	406829.00
mean	12.06	2011-07-10 16:30:57.879207424	3.46	15287.69
min	-80995.00	2010-12-01 08:26:00	0.00	12346.00
25%	2.00	2011-04-06 15:02:00	1.25	13953.00
50%	5.00	2011-07-31 11:48:00	1.95	15152.00
75%	12.00	2011-10-20 13:06:00	3.75	16791.00
max	80995.00	2011-12-09 12:50:00	38970.00	18287.00
std	248.69	NaN	69.32	1713.60

Let us do some analysis

Conclusion is: quantity column having negative values

So, we need to remove/delete negative values

Example to delete negative values from list object

In [47]:
```
values = [1, 2, 3, -4, -5, 6, 7]
```

In [48]:
```
for value in values:
    print(value)
```

```
1
2
3
-4
-5
6
7
```

In [49]:
```
for value in values:
    if value >= 0:
        print(value)
```

```
1
2
3
6
7
```

Remove negative values from quantity column

In [50]:
```
df_new.quantity > 0
```

Out[50]:
```
0          True
1          True
2          True
3          True
4          True
           ...
541904     True
541905     True
541906     True
541907     True
541908     True
Name: quantity, Length: 406829, dtype: bool
```

In [51]:
```
con = df_new.quantity > 0
```

In [52]:
```
df_new = df_new[con]
```

In [53]:
```
df_new.describe().round(2)
```

Out[53]:

	quantity	invoice_date	unit_price	cust_id
count	397924.00	397924	397924.00	397924.00
mean	13.02	2011-07-10 23:43:36.912475648	3.12	15294.32
min	1.00	2010-12-01 08:26:00	0.00	12346.00
25%	2.00	2011-04-07 11:12:00	1.25	13969.00
50%	6.00	2011-07-31 14:39:00	1.95	15159.00
75%	12.00	2011-10-20 14:33:00	3.75	16795.00
max	80995.00	2011-12-09 12:50:00	8142.75	18287.00
std	180.42	NaN	22.10	1713.17

Access initial Data

In [54]:
```
df_new.head()
```

Out[54]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	white hanging heart t-light holder	6	2010-12-01 08:26:00	2.55	17850
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850

Checking total number of rows and columns

In [55]:
```
df_new.shape
```

Out[55]:
```
(397924, 8)
```

Adding the column - amount_spent

In [56]:
```
df_new['amount_spent'] = df_new['quantity'] * df_new['unit_price']
```

In [57]:
```
df_new.head()
```

Out[57]:

	invoice_num	stock_code	description	quantity	invoice_date	unit_price	cust_id
0	536365	85123A	white hanging heart t-light holder	6	2010-12-01 08:26:00	2.55	17850
1	536365	71053	white metal lantern	6	2010-12-01 08:26:00	3.39	17850
2	536365	84406B	cream cupid hearts coat hanger	8	2010-12-01 08:26:00	2.75	17850
3	536365	84029G	knitted union flag hot water bottle	6	2010-12-01 08:26:00	3.39	17850
4	536365	84029E	red woolly hottie white heart.	6	2010-12-01 08:26:00	3.39	17850

Lets read the column names from DataFrame

In [58]:
```
for col in df_new.columns:
    print(col)
```

```
invoice_num
stock_code
description
quantity
invoice_date
unit_price
cust_id
country
amount_spent
```

Rearranging columns for more readability

In [59]:
```
col_order = ['invoice_num','invoice_date','stock_code','description','qua
```

In [60]:
```
df_new = df_new[col_order]
```

Access initial data

In [61]:
```
df_new.head()
```

Out[61]:

	invoice_num	invoice_date	stock_code	description	quantity	unit_price	amount_s
0	536365	2010-12-01 08:26:00	85123A	white hanging heart t-light holder	6	2.55	
1	536365	2010-12-01 08:26:00	71053	white metal lantern	6	3.39	
2	536365	2010-12-01 08:26:00	84406B	cream cupid hearts coat hanger	8	2.75	
3	536365	2010-12-01 08:26:00	84029G	knitted union flag hot water bottle	6	3.39	
4	536365	2010-12-01 08:26:00	84029E	red woolly hottie white heart.	6	3.39	

Number of rows and columns

In [62]:
```
df_new.shape
```

Out[62]: (397924, 9)

Let us do analysis on invoice_date column

number of columns in the dataset

In [63]:
```
len(df_new.columns)
```

Out[63]: 9

Accessing invoice_date column

Method - 1 to access column

In [64]:
```
df_new['invoice_date']
```

Out[64]:
```
0         2010-12-01 08:26:00
1         2010-12-01 08:26:00
2         2010-12-01 08:26:00
3         2010-12-01 08:26:00
4         2010-12-01 08:26:00
                  ...
541904    2011-12-09 12:50:00
541905    2011-12-09 12:50:00
541906    2011-12-09 12:50:00
541907    2011-12-09 12:50:00
541908    2011-12-09 12:50:00
Name: invoice_date, Length: 397924, dtype: datetime64[ns]
```

Method - 2 to access column

In [65]:
```
df_new.invoice_date
```

Out[65]:
```
0         2010-12-01 08:26:00
1         2010-12-01 08:26:00
2         2010-12-01 08:26:00
3         2010-12-01 08:26:00
4         2010-12-01 08:26:00
                  ...
541904    2011-12-09 12:50:00
541905    2011-12-09 12:50:00
541906    2011-12-09 12:50:00
541907    2011-12-09 12:50:00
541908    2011-12-09 12:50:00
Name: invoice_date, Length: 397924, dtype: datetime64[ns]
```

Accessing year value from invoice_date

In [66]:
```
df_new['invoice_date'].dt.year
```

Out[66]:
```
0         2010
1         2010
2         2010
3         2010
4         2010
          ...
541904    2011
541905    2011
541906    2011
541907    2011
541908    2011
Name: invoice_date, Length: 397924, dtype: int32
```

Accessing month value from invoice_date

In [67]:
```
df_new['invoice_date'].dt.month
```

Out[67]:
```
0          12
1          12
2          12
3          12
4          12
           ..
541904     12
541905     12
541906     12
541907     12
541908     12
Name: invoice_date, Length: 397924, dtype: int32
```

Access initial Data

In [68]:
```
df_new.head(2)
```

Out[68]:

	invoice_num	invoice_date	stock_code	description	quantity	unit_price	amount_s
0	536365	2010-12-01 08:26:00	85123A	white hanging heart t-light holder	6	2.55	
1	536365	2010-12-01 08:26:00	71053	white metal lantern	6	3.39	

Lets insert year_month colum in 2nd position

small calculation

In [69]:
```
y = 2010
m = 12
```

In [70]:
```
y_m = 100*2010 + 12
```

In [71]:
```
y_m
```

Out[71]:
```
201012
```

In [72]:
```
c1 = 'year_month'
```

In [73]:
```
v1 = df_new['invoice_date'].map(lambda col: 100*(col.year) + col.month)
```

In [74]:
```
df_new.insert(loc = 2, column = c1, value = v1)
```

In [75]:
```
df_new
```

Out[75]:

	invoice_num	invoice_date	year_month	stock_code	description	quantity	ur
0	536365	2010-12-01 08:26:00	201012	85123A	white hanging heart t-light holder	6	
1	536365	2010-12-01 08:26:00	201012	71053	white metal lantern	6	
2	536365	2010-12-01 08:26:00	201012	84406B	cream cupid hearts coat hanger	8	
3	536365	2010-12-01 08:26:00	201012	84029G	knitted union flag hot water bottle	6	
4	536365	2010-12-01 08:26:00	201012	84029E	red woolly hottie white heart.	6	
...	...	...	...	...	...	...	
541904	581587	2011-12-09 12:50:00	201112	22613	pack of 20 spaceboy napkins	12	
541905	581587	2011-12-09 12:50:00	201112	22899	children's apron dolly girl	6	
541906	581587	2011-12-09 12:50:00	201112	23254	childrens cutlery dolly girl	4	
541907	581587	2011-12-09 12:50:00	201112	23255	childrens cutlery circus parade	4	
541908	581587	2011-12-09 12:50:00	201112	22138	baking set 9 piece retrospot	3	

397924 rows × 10 columns

Access initial data

In [76]:
```
df_new.head()
```

Out[76]:

	invoice_num	invoice_date	year_month	stock_code	description	quantity	unit_pri
0	536365	2010-12-01 08:26:00	201012	85123A	white hanging heart t-light holder	6	2.
1	536365	2010-12-01 08:26:00	201012	71053	white metal lantern	6	3.
2	536365	2010-12-01 08:26:00	201012	84406B	cream cupid hearts coat hanger	8	2.
3	536365	2010-12-01 08:26:00	201012	84029G	knitted union flag hot water bottle	6	3.
4	536365	2010-12-01 08:26:00	201012	84029E	red woolly hottie white heart.	6	3.

Adding month column to the exisint DataFrame

In [77]:
```
c2 = 'month'
```

In [78]:
```
v2 = df_new.invoice_date.dt.month
```

In [79]:
```
df_new.insert(loc = 3, column = c2, value = v2)
```

In [80]:
```
df_new.head()
```

Out[80]:

	invoice_num	invoice_date	year_month	month	stock_code	description	quantity
0	536365	2010-12-01 08:26:00	201012	12	85123A	white hanging heart t-light holder	6
1	536365	2010-12-01 08:26:00	201012	12	71053	white metal lantern	6
2	536365	2010-12-01 08:26:00	201012	12	84406B	cream cupid hearts coat hanger	8
3	536365	2010-12-01 08:26:00	201012	12	84029G	knitted union flag hot water bottle	6
4	536365	2010-12-01 08:26:00	201012	12	84029E	red woolly hottie white heart.	6

Lets access invoice_date column

In [81]:
```
df_new.invoice_date
```

Out[81]:
```
0         2010-12-01 08:26:00
1         2010-12-01 08:26:00
2         2010-12-01 08:26:00
3         2010-12-01 08:26:00
4         2010-12-01 08:26:00
                  ...
541904    2011-12-09 12:50:00
541905    2011-12-09 12:50:00
541906    2011-12-09 12:50:00
541907    2011-12-09 12:50:00
541908    2011-12-09 12:50:00
Name: invoice_date, Length: 397924, dtype: datetime64[ns]
```

We can get day of the week

In [82]:
```
df_new.invoice_date.dt.dayofweek
```

Out[82]:
```
0         2
1         2
2         2
3         2
4         2
         ..
541904    4
541905    4
541906    4
541907    4
541908    4
Name: invoice_date, Length: 397924, dtype: int32
```

In pandas, the day formate starts from 0 to 6

Monday = 0 Tuesday = 1 .... Sunday = 6

Apply +1 to make Monday = 1.....until Sunday = 7

In [83]:
```
c3 = 'day'
```

In [84]:
```
v3 = (df_new.invoice_date.dt.dayofweek)+1
```

In [85]:
```
df_new.insert(loc = 4, column = c3, value = v3)
```

In [86]:
```
df_new.head()
```

Out[86]:

	invoice_num	invoice_date	year_month	month	day	stock_code	description	qua
0	536365	2010-12-01 08:26:00	201012	12	3	85123A	white hanging heart t-light holder	
1	536365	2010-12-01 08:26:00	201012	12	3	71053	white metal lantern	
2	536365	2010-12-01 08:26:00	201012	12	3	84406B	cream cupid hearts coat hanger	
3	536365	2010-12-01 08:26:00	201012	12	3	84029G	knitted union flag hot water bottle	
4	536365	2010-12-01 08:26:00	201012	12	3	84029E	red woolly hottie white heart.	

Adding hour column to existing DataFrame

In [87]:
```
df_new.invoice_date
```

Out[87]:
```
0         2010-12-01 08:26:00
1         2010-12-01 08:26:00
2         2010-12-01 08:26:00
3         2010-12-01 08:26:00
4         2010-12-01 08:26:00
                  ...
541904    2011-12-09 12:50:00
541905    2011-12-09 12:50:00
541906    2011-12-09 12:50:00
541907    2011-12-09 12:50:00
541908    2011-12-09 12:50:00
Name: invoice_date, Length: 397924, dtype: datetime64[ns]
```

In [88]:
```
# dir(df_new.invoice_date)
```

In [89]:
```
# dir(df_new.invoice_date.dt)
```

In [90]:
```
# df_new.invoice_date.dt.hour
```

In [91]:
```
c4 = "hour"
```

In [92]:
```
v4 = df_new.invoice_date.dt.hour
```

In [93]:
```
df_new.insert(loc = 5, column = c4, value = v4)
```

In [94]:
```
df_new.head()
```

Out[94]:

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	white hanging heart t-ligh holde
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	white meta lantern
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	crean cupi hearts coa hange
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	knitte union flag hot wate bottl
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	red wooll hottie white heart

Lets display all columns once

In [95]:
```
df_new.columns
```

Out[95]:
```
Index(['invoice_num', 'invoice_date', 'year_month', 'month', 'day', 'hou
r',
       'stock_code', 'description', 'quantity', 'unit_price', 'amount_sp
ent',
       'cust_id', 'country'],
      dtype='object')
```

In [96]:
```
for col in df_new.columns:
    print(col)
```

```
invoice_num
invoice_date
year_month
month
day
hour
stock_code
description
quantity
unit_price
amount_spent
cust_id
country
```

Exploratory Data Analysis (EDA)

In [97]:
```
df_new.groupby(by = ['cust_id','country'])['invoice_num'].count()
```

Out[97]:
```
cust_id  country
12346    United Kingdom      1
12347    Iceland           182
12348    Finland            31
12349    Italy              73
12350    Norway             17
                           ...
18280    United Kingdom     10
18281    United Kingdom      7
18282    United Kingdom     12
18283    United Kingdom    756
18287    United Kingdom     70
Name: invoice_num, Length: 4347, dtype: int64
```

In [98]:
```
df_new.groupby(by = ['cust_id','country'], as_index = False)['invoice_num
```

Out[98]:

	cust_id	country	invoice_num
0	12346	United Kingdom	1
1	12347	Iceland	182
2	12348	Finland	31
3	12349	Italy	73
4	12350	Norway	17
...	...	...	...
4342	18280	United Kingdom	10
4343	18281	United Kingdom	7
4344	18282	United Kingdom	12
4345	18283	United Kingdom	756
4346	18287	United Kingdom	70

4347 rows × 3 columns

Data Visaulization libraries

In [99]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [100…
```
orders = df_new.groupby(by=['cust_id','country'], as_index=False)['invoic
```

In [101…
```
orders
```

Out[101…

	cust_id	country	invoice_num
0	12346	United Kingdom	1
1	12347	Iceland	182
2	12348	Finland	31
3	12349	Italy	73
4	12350	Norway	17
...	...	...	...
4342	18280	United Kingdom	10
4343	18281	United Kingdom	7
4344	18282	United Kingdom	12
4345	18283	United Kingdom	756
4346	18287	United Kingdom	70

4347 rows × 3 columns

Check TOP 5 most number of orders

In [102…

```
orders.sort_values(by = 'invoice_num', ascending = False).head()
```

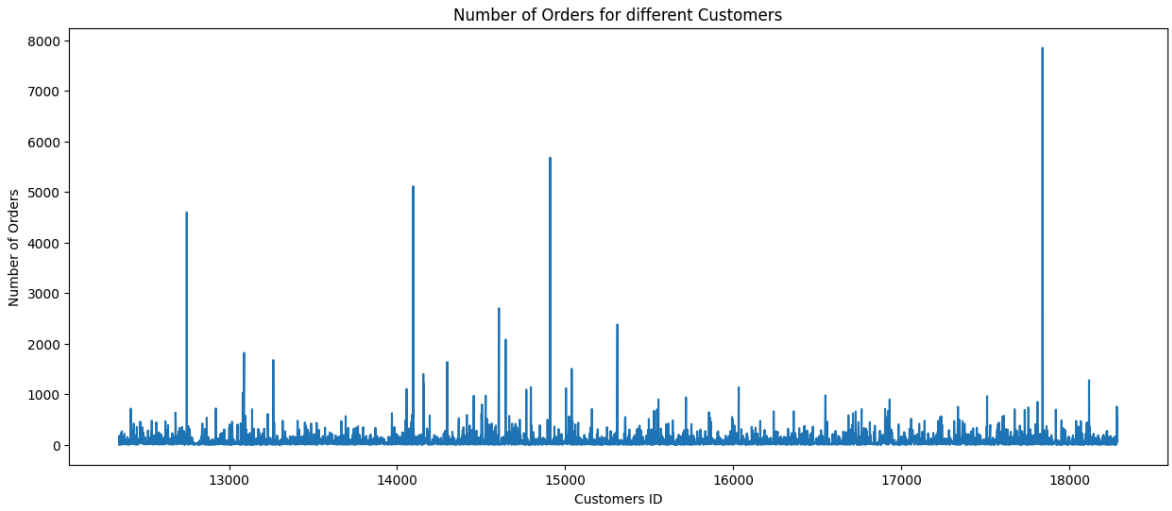
Out[102…

	cust_id	country	invoice_num
4019	17841	United Kingdom	7847
1888	14911	EIRE	5677
1298	14096	United Kingdom	5111
334	12748	United Kingdom	4596
1670	14606	United Kingdom	2700

Visualizing - Number of Orders for different Customers

In [103…

```
orders = df_new.groupby(by=['cust_id','country'], as_index=False)['invoic

plt.subplots(figsize=(15,6))

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()
```



How much money spent by each customers?

In [104…
```
df_new.groupby(by = ['cust_id', 'country'])['amount_spent'].sum()
```

Out[104…
```
cust_id  country
12346    United Kingdom    77183.60
12347    Iceland            4310.00
12348    Finland            1797.24
12349    Italy              1757.55
12350    Norway              334.40
                             ...
18280    United Kingdom      180.60
18281    United Kingdom       80.82
18282    United Kingdom      178.05
18283    United Kingdom     2094.88
18287    United Kingdom     1837.28
Name: amount_spent, Length: 4347, dtype: float64
```

In [105…
```
df_new.groupby(by = ['cust_id', 'country'], as_index = False)['amount_spe
```

Out[105...

	cust_id	country	amount_spent
0	12346	United Kingdom	77183.60
1	12347	Iceland	4310.00
2	12348	Finland	1797.24
3	12349	Italy	1757.55
4	12350	Norway	334.40
...	...	...	...
4342	18280	United Kingdom	180.60
4343	18281	United Kingdom	80.82
4344	18282	United Kingdom	178.05
4345	18283	United Kingdom	2094.88
4346	18287	United Kingdom	1837.28

4347 rows × 3 columns

In [106...
```
money_spent = df_new.groupby(by = ['cust_id', 'country'], as_index = Fals
```

In [107...
```
money_spent
```

Out[107...

	cust_id	country	amount_spent
0	12346	United Kingdom	77183.60
1	12347	Iceland	4310.00
2	12348	Finland	1797.24
3	12349	Italy	1757.55
4	12350	Norway	334.40
...	...	...	...
4342	18280	United Kingdom	180.60
4343	18281	United Kingdom	80.82
4344	18282	United Kingdom	178.05
4345	18283	United Kingdom	2094.88
4346	18287	United Kingdom	1837.28

4347 rows × 3 columns

Top FIVE customers who spend highest money

In [108...
```
money_spent.sort_values(by='amount_spent', ascending = False).head()
```

Out[108…

	cust_id	country	amount_spent
1698	14646	Netherlands	280206.02
4210	18102	United Kingdom	259657.30
3737	17450	United Kingdom	194550.79
3017	16446	United Kingdom	168472.50
1888	14911	EIRE	143825.06

Top TEN customers who spend highest money

In [109…

```
money_spent.sort_values(by='amount_spent', ascending = False).head(10)
```

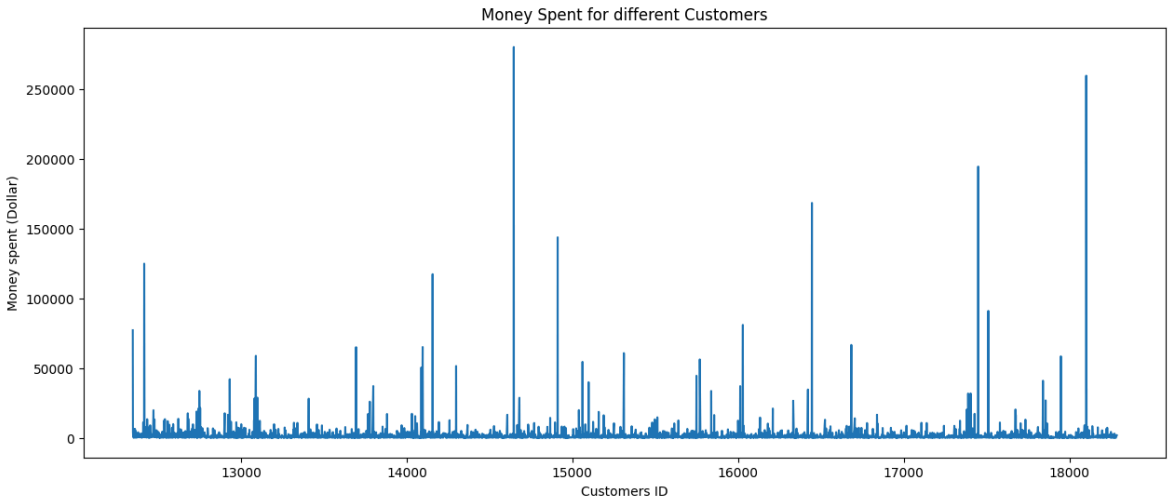
Out[109…

	cust_id	country	amount_spent
1698	14646	Netherlands	280206.02
4210	18102	United Kingdom	259657.30
3737	17450	United Kingdom	194550.79
3017	16446	United Kingdom	168472.50
1888	14911	EIRE	143825.06
57	12415	Australia	124914.53
1342	14156	EIRE	117379.63
3780	17511	United Kingdom	91062.38
2711	16029	United Kingdom	81024.84
0	12346	United Kingdom	77183.60

Visualizing - Money spent for different customers

In [110…

```
money_spent = df_new.groupby(by=['cust_id','country'], as_index=False)['a

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 [111…
```
df_new.head()
```

Out[111…

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	descriptio
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	whit hangin heart t-ligh holde
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	white meta lanter
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	crean cupi hearts coa hange
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	knitte union fla hot wate bottl
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	red wooll hottie whit heart

Number of order for different months

In [112…
```
color = sns.color_palette()
```

Initial Data

In [113…
```
df_new.head()
```

Out[113…

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	descriptio
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	whit hangin heart t-ligh holde
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	white meta lanter
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	crean cupi hearts coa hange
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	knitte union fla hot wate bottl
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	red wooll hottie whit heart

In [114…
```
df_new.groupby('invoice_num')['year_month'].unique()
```

Out[114…
```
invoice_num
536365    [201012]
536366    [201012]
536367    [201012]
536368    [201012]
536369    [201012]
             ...
581583    [201112]
581584    [201112]
581585    [201112]
581586    [201112]
581587    [201112]
Name: year_month, Length: 18536, dtype: object
```

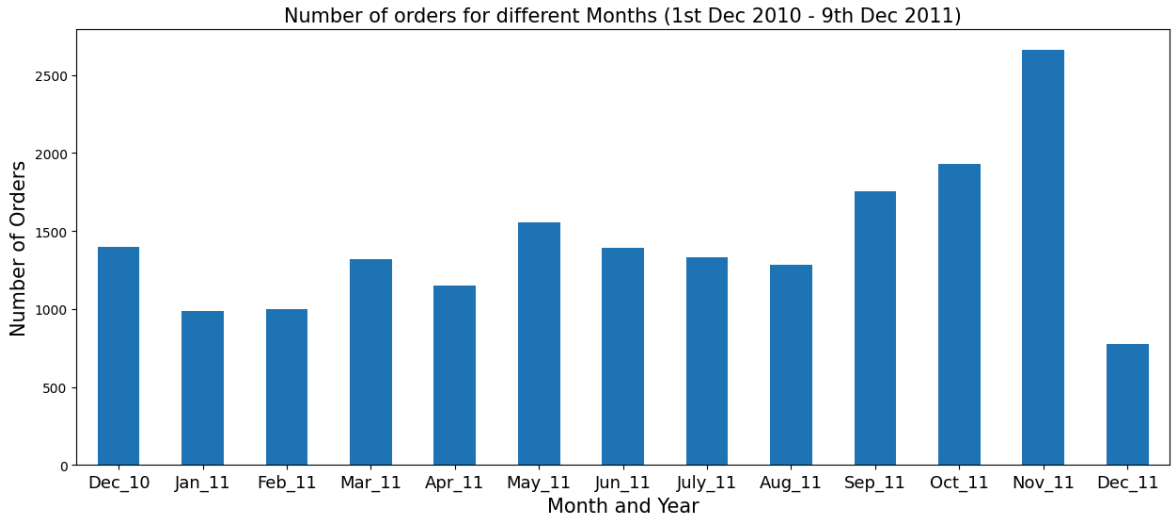
In [115…
```
df_new.groupby('invoice_num')['year_month'].unique().value_counts()
```

Out[115…
```
year_month
[201111]    2658
[201110]    1929
[201109]    1756
[201105]    1555
[201012]    1400
[201106]    1393
[201107]    1331
[201103]    1321
[201108]    1281
[201104]    1149
[201102]     998
[201101]     987
[201112]     778
Name: count, dtype: int64
```

In [116…
```
df_new.groupby('invoice_num')['year_month'].unique().value_counts().sort_
```

Out[116…
```
year_month
[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: count, dtype: int64
```

In [117…
```
ax = df_new.groupby('invoice_num')['year_month'].unique().value_counts().

ax.set_xlabel('Month and Year',fontsize=15)
ax.set_ylabel('Number of Orders',fontsize=15)
ax.set_title('Number of orders for different Months (1st Dec 2010 - 9th D

t = ('Dec_10','Jan_11','Feb_11','Mar_11','Apr_11','May_11','Jun_11','July

ax.set_xticklabels(t, rotation='horizontal', fontsize=13)

plt.show()
```



How many orders (per day)?

In [118…
```
df_new.groupby('invoice_num')['day'].unique()
```

Out[118…
```
invoice_num
536365    [3]
536366    [3]
536367    [3]
536368    [3]
536369    [3]
          ...
581583    [5]
581584    [5]
581585    [5]
581586    [5]
581587    [5]
Name: day, Length: 18536, dtype: object
```

In [119…
```
df_new.groupby('invoice_num')['day'].unique().value_counts()
```

Out[119…
```
day
[4]    4033
[3]    3455
[2]    3185
[1]    2863
[5]    2831
[7]    2169
Name: count, dtype: int64
```

In [120…
```
df_new.groupby('invoice_num')['day'].unique().value_counts().sort_index()
```

Out[120…
```
day
[1]    2863
[2]    3185
[3]    3455
[4]    4033
[5]    2831
[7]    2169
Name: count, dtype: int64
```

Day wise sales count/business

In [121…
```
df_new.head()
```

Out[121…

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	description
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	white hanging heart t-ligh holde
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	white meta lanter
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	crean cupi hearts coa hange
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	knitte union fla hot wate bottl
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	red wooll hottie whit heart

In [122…
```
df_new.groupby('invoice_num')['day'].unique()
```

Out[122…
```
invoice_num
536365    [3]
536366    [3]
536367    [3]
536368    [3]
536369    [3]
          ...
581583    [5]
581584    [5]
581585    [5]
581586    [5]
581587    [5]
Name: day, Length: 18536, dtype: object
```

In [123…
```
df_new.groupby('invoice_num')['day'].unique().value_counts()
```

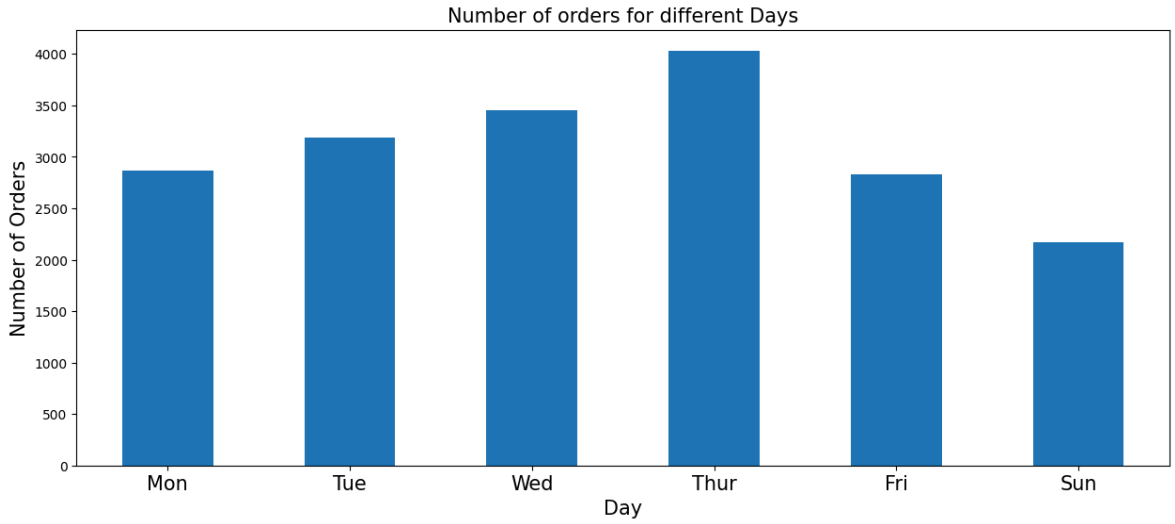
Out[123…
```
day
[4]    4033
[3]    3455
[2]    3185
[1]    2863
[5]    2831
[7]    2169
Name: count, dtype: int64
```

In [124…
```
df_new.groupby('invoice_num')['day'].unique().value_counts().sort_index()
```

Out[124…
```
day
[1]    2863
[2]    3185
[3]    3455
[4]    4033
[5]    2831
[7]    2169
Name: count, dtype: int64
```

Lets visualizat Day wise sales count/business

In [125…
```
ax = df_new.groupby('invoice_num')['day'].unique().value_counts().sort_in

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)
d = ('Mon','Tue','Wed','Thur','Fri','Sun')
ax.set_xticklabels(d, rotation='horizontal', fontsize=15)
plt.show()
```



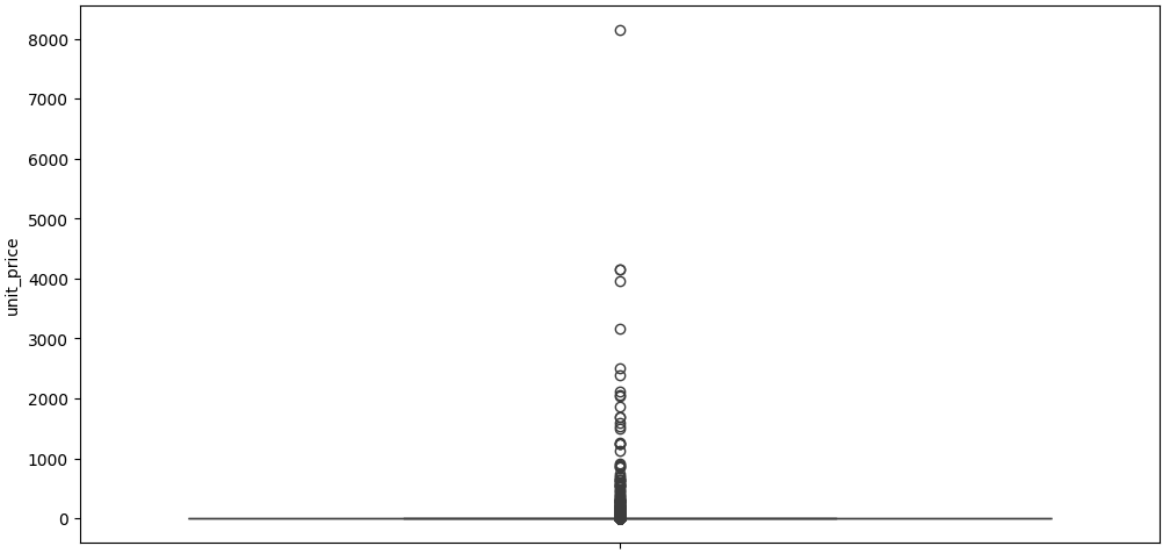
Discover patterns for Unit Price

In [126…
```
df_new.unit_price.describe()
```

Out[126…
```
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
```

Min value for product is zero, so there are some free products

In [127…
```
# check the distribution of unit price
plt.subplots(figsize = (12,6))

sns.boxplot(df_new.unit_price)
plt.show()
```



Filter only free products(cost = 0)

In [128…
```
df_free = df_new[df_new.unit_price == 0]
```

In [129…
```
len(df_free)
```

Out[129… 40

In [130…
```
df_free.year_month
```

Out[130…
```
9302      201012
33576     201012
40089     201012
47068     201101
47070     201101
56674     201101
86789     201102
130188    201103
139453    201103
145208    201104
157042    201104
187613    201105
198383    201105
279324    201107
282912    201107
285657    201108
298054    201108
314745    201108
314746    201108
314747    201108
314748    201108
358655    201109
361825    201109
379913    201110
395529    201110
420404    201110
436428    201111
436597    201111
436961    201111
439361    201111
446125    201111
446793    201111
446794    201111
454463    201111
454464    201111
479079    201111
479546    201111
480649    201111
485985    201111
502122    201111
Name: year_month, dtype: int64
```

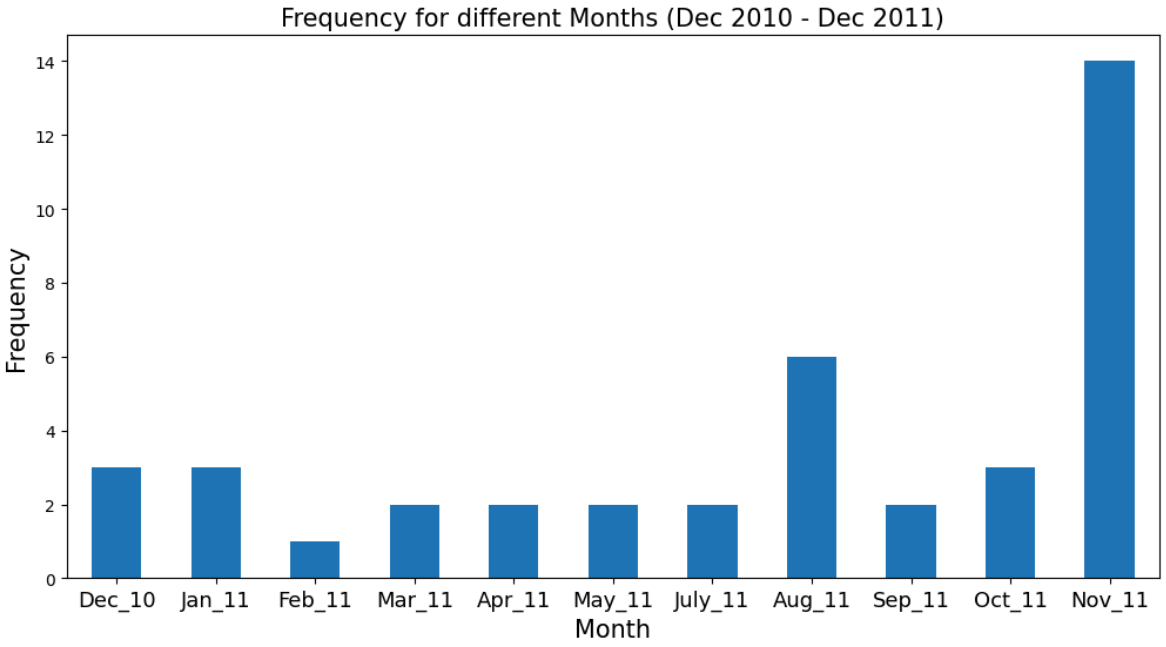
In [131…
```
df_free.year_month.value_counts()
```

Out[131…
```
year_month
201111    14
201108     6
201012     3
201101     3
201110     3
201103     2
201104     2
201105     2
201107     2
201109     2
201102     1
Name: count, dtype: int64
```

In [132…
```
df_free.year_month.value_counts().sort_index()
```

Out[132…
```
year_month
201012      3
201101      3
201102      1
201103      2
201104      2
201105      2
201107      2
201108      6
201109      2
201110      3
201111     14
Name: count, dtype: int64
```

In [133…
```
ax = df_free.year_month.value_counts().sort_index().plot(kind = 'bar',fig
ax.set_xlabel('Month',fontsize=15)
ax.set_ylabel('Frequency',fontsize=15)
ax.set_title('Frequency for different Months (Dec 2010 - Dec 2011)',fonts

m = ('Dec_10','Jan_11','Feb_11','Mar_11','Apr_11','May_11','July_11','Aug

ax.set_xticklabels(m, rotation='horizontal', fontsize=13)
plt.show()
```



How many orders for each country?

In [134…
```
df_new
```

Out[134…

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	desc
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	h hea
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	whit
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	hea
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	un ho
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	rec hotti
...	...	...	...	...	...	...	...	
541904	581587	2011-12-09 12:50:00	201112	12	5	12	22613	pac sp r
541905	581587	2011-12-09 12:50:00	201112	12	5	12	22899	ch apr
541906	581587	2011-12-09 12:50:00	201112	12	5	12	23254	ch cutle
541907	581587	2011-12-09 12:50:00	201112	12	5	12	23255	ch
541908	581587	2011-12-09 12:50:00	201112	12	5	12	22138	bak re

397924 rows × 13 columns

In [135…

```
df_new.groupby('country')['invoice_num'].count()
```

Out[135…

```
country
Australia                  1185
Austria                     398
Bahrain                      17
Belgium                    2031
Brazil                       32
Canada                      151
Channel Islands             748
Cyprus                      614
Czech Republic               25
Denmark                     380
EIRE                       7238
European Community           60
Finland                     685
France                     8342
Germany                    9042
Greece                      145
Iceland                     182
Israel                      248
Italy                       758
Japan                       321
Lebanon                      45
Lithuania                    35
Malta                       112
Netherlands                2363
Norway                     1072
Poland                      330
Portugal                   1462
RSA                          58
Saudi Arabia                  9
Singapore                   222
Spain                      2485
Sweden                      451
Switzerland                1842
USA                         179
United Arab Emirates         68
United Kingdom           354345
Unspecified                 244
Name: invoice_num, dtype: int64
```

In [136…

```
df_new.groupby('country')['invoice_num'].count().sort_values()
```

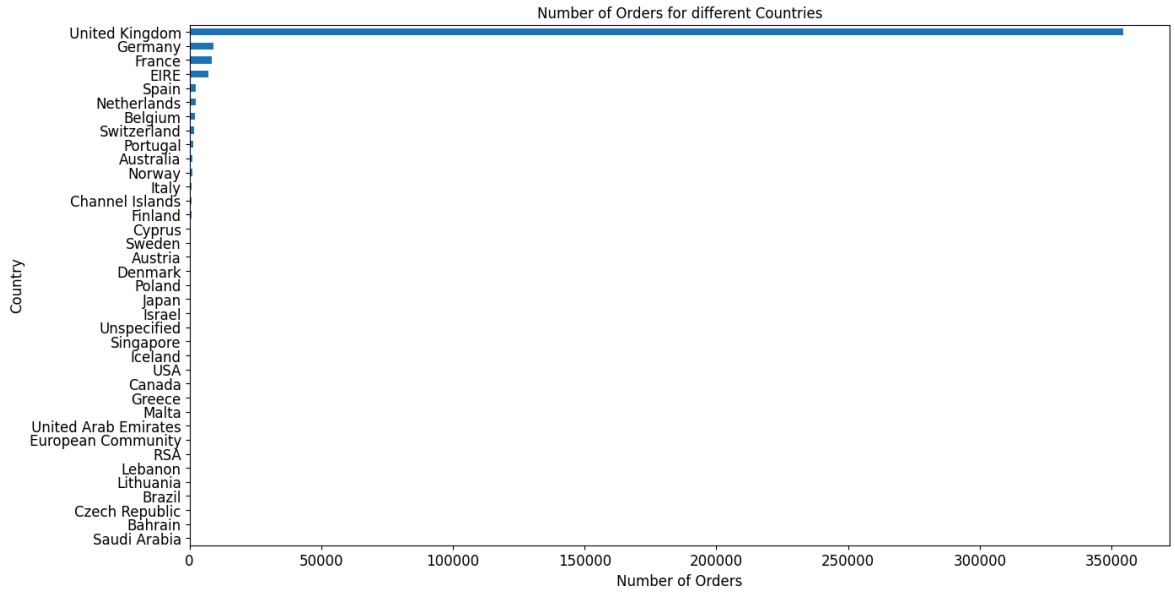
Out[136…

```
country
Saudi Arabia                 9
Bahrain                     17
Czech Republic              25
Brazil                      32
Lithuania                   35
Lebanon                     45
RSA                         58
European Community          60
United Arab Emirates        68
Malta                      112
Greece                     145
Canada                     151
USA                        179
Iceland                    182
Singapore                  222
Unspecified                244
Israel                     248
Japan                      321
Poland                     330
Denmark                    380
Austria                    398
Sweden                     451
Cyprus                     614
Finland                    685
Channel Islands            748
Italy                      758
Norway                    1072
Australia                 1185
Portugal                  1462
Switzerland               1842
Belgium                   2031
Netherlands               2363
Spain                     2485
EIRE                      7238
France                    8342
Germany                   9042
United Kingdom          354345
Name: invoice_num, dtype: int64
```

How many orders for each country?

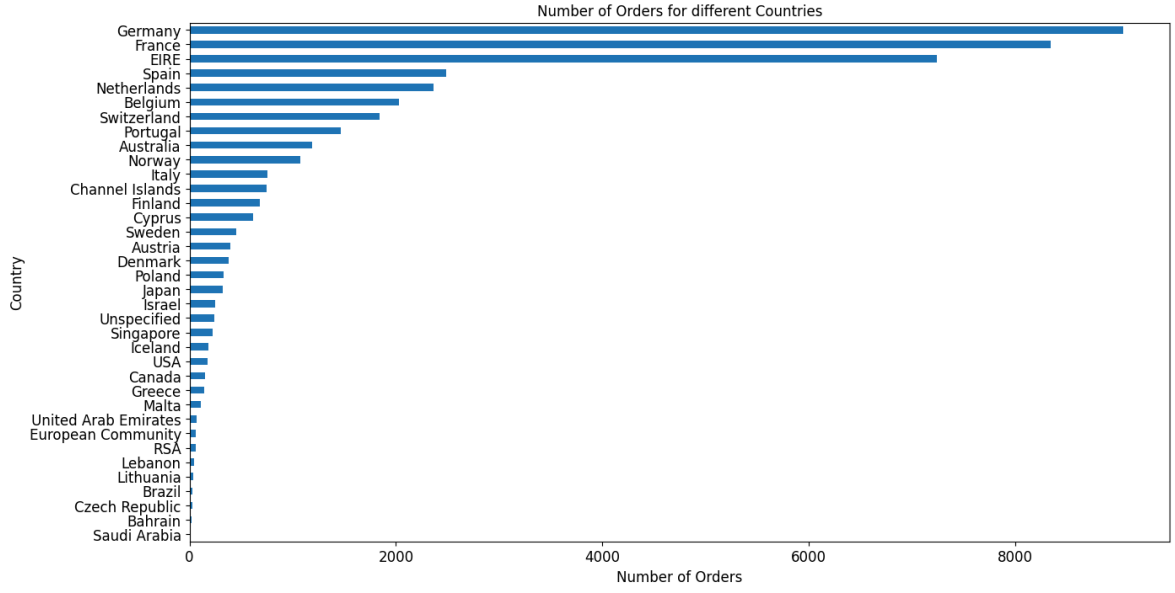
In [137…

```
group_country_orders = df_new.groupby('country')['invoice_num'].count().s
# 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, 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()
```



In [138…

```
group_country_orders = df_new.groupby('country')['invoice_num'].count().s
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, 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()
```



How much money spent by each country?

In [139…

```
df_new
```

Out[139…

	invoice_num	invoice_date	year_month	month	day	hour	stock_code	desc
0	536365	2010-12-01 08:26:00	201012	12	3	8	85123A	h hea
1	536365	2010-12-01 08:26:00	201012	12	3	8	71053	whit
2	536365	2010-12-01 08:26:00	201012	12	3	8	84406B	hea
3	536365	2010-12-01 08:26:00	201012	12	3	8	84029G	un ho
4	536365	2010-12-01 08:26:00	201012	12	3	8	84029E	rec hotti
...	...	...	...	...	...	...	...	
541904	581587	2011-12-09 12:50:00	201112	12	5	12	22613	pac sp r
541905	581587	2011-12-09 12:50:00	201112	12	5	12	22899	ch apr
541906	581587	2011-12-09 12:50:00	201112	12	5	12	23254	ch cutle
541907	581587	2011-12-09 12:50:00	201112	12	5	12	23255	ch
541908	581587	2011-12-09 12:50:00	201112	12	5	12	22138	bak re

397924 rows × 13 columns

In [140…

```
df_new.groupby('country')['amount_spent'].sum()
```

Out[140…

```
country
Australia                138521.310
Austria                   10198.680
Bahrain                     548.400
Belgium                   41196.340
Brazil                     1143.600
Canada                     3666.380
Channel Islands           20450.440
Cyprus                    13590.380
Czech Republic              826.740
Denmark                   18955.340
EIRE                     265545.900
European Community         1300.250
Finland                   22546.080
France                   209024.050
Germany                  228867.140
Greece                     4760.520
Iceland                    4310.000
Israel                     7221.690
Italy                     17483.240
Japan                     37416.370
Lebanon                    1693.880
Lithuania                  1661.060
Malta                      2725.590
Netherlands              285446.340
Norway                    36165.440
Poland                     7334.650
Portugal                  33439.890
RSA                        1002.310
Saudi Arabia                145.920
Singapore                 21279.290
Spain                     61577.110
Sweden                    38378.330
Switzerland               56443.950
USA                        3580.390
United Arab Emirates       1902.280
United Kingdom          7308391.554
Unspecified                2667.070
Name: amount_spent, dtype: float64
```

In [141…

```
df_new.groupby('country')['amount_spent'].sum().sort_values()
```

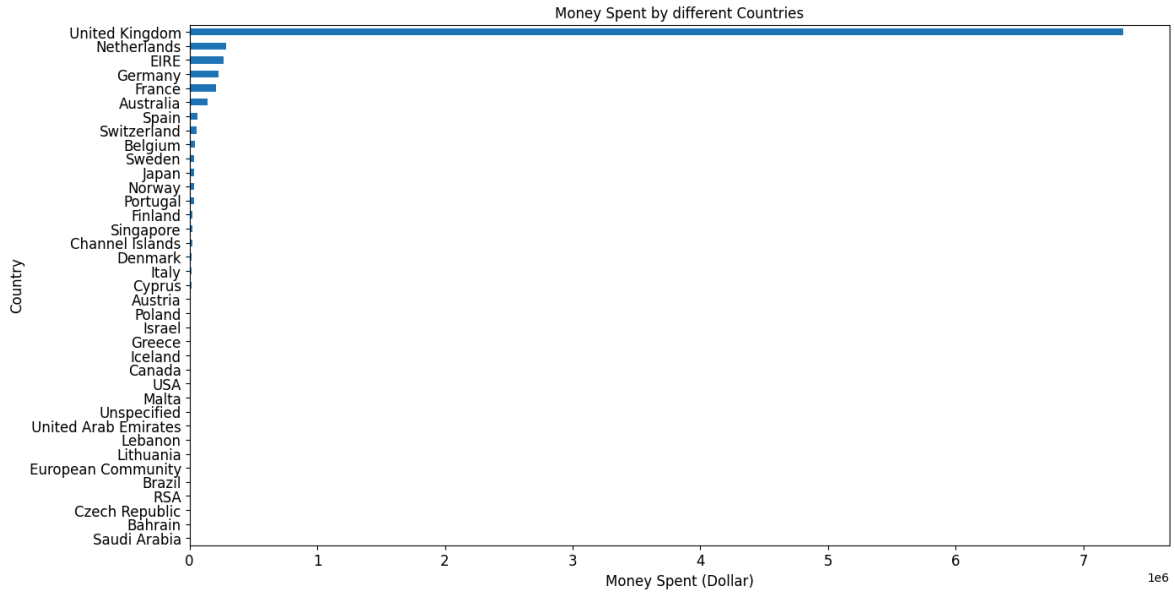
Out[141…

```
country
Saudi Arabia                 145.920
Bahrain                      548.400
Czech Republic               826.740
RSA                         1002.310
Brazil                      1143.600
European Community          1300.250
Lithuania                   1661.060
Lebanon                     1693.880
United Arab Emirates        1902.280
Unspecified                 2667.070
Malta                       2725.590
USA                         3580.390
Canada                      3666.380
Iceland                     4310.000
Greece                      4760.520
Israel                      7221.690
Poland                      7334.650
Austria                    10198.680
Cyprus                     13590.380
Italy                      17483.240
Denmark                    18955.340
Channel Islands            20450.440
Singapore                  21279.290
Finland                    22546.080
Portugal                   33439.890
Norway                     36165.440
Japan                      37416.370
Sweden                     38378.330
Belgium                    41196.340
Switzerland                56443.950
Spain                      61577.110
Australia                 138521.310
France                    209024.050
Germany                   228867.140
EIRE                      265545.900
Netherlands               285446.340
United Kingdom           7308391.554
Name: amount_spent, dtype: float64
```

How much money spent by each country?

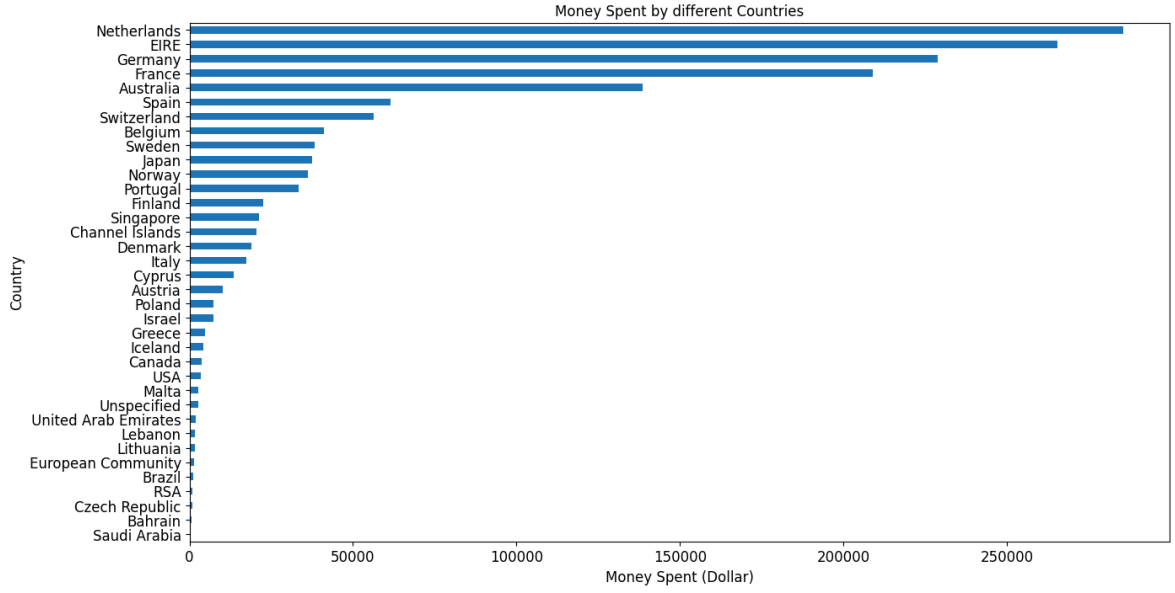
In [142…

```
group_country_amount_spent = df_new.groupby('country')['amount_spent'].su
# 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, color=color[0
plt.xlabel('Money Spent (Dollar)', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Money Spent by different Countries', fontsize=12)
plt.show()
```



In [143…

```
group_country_amount_spent = df_new.groupby('country')['amount_spent'].su
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, color=color[0
plt.xlabel('Money Spent (Dollar)', fontsize=12)
plt.ylabel('Country', fontsize=12)
plt.title('Money Spent by different Countries', fontsize=12)
plt.show()
```



This is called Data Analysis :)

In [ ]: