# crm-project

May 11, 2024

#CRM Analysis

CRM (customer relationship management) analytics comprises all of the programming that analyzes data about customers and presents it to an organization to help facilitate and streamline better business decisions.

It involves in the systematic interpretation of data related to interactions between and its customers through CRM analysis.

In this case study we try to evaluate customer behaviour,preferences and feedback to gain valuable insights into needs and expections.

[ ]:
```
''' Importing the required libraries for our analysis '''

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
''' reading the data file '''

df = pd.read_csv('Ecom_CRM_analysis.csv' , encoding= 'unicode_escape')
df
```

[ ]:

| | InvoiceNo | StockCode | Description | Quantity | \ |
|---|---|---|---|---|---|
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | |
| … | … | … | … | … | |
| 541904 | 581587 | 22613 | PACK OF 20 SPACEBOY NAPKINS | 12 | |
| 541905 | 581587 | 22899 | CHILDREN'S APRON DOLLY GIRL | 6 | |
| 541906 | 581587 | 23254 | CHILDRENS CUTLERY DOLLY GIRL | 4 | |
| 541907 | 581587 | 23255 | CHILDRENS CUTLERY CIRCUS PARADE | 4 | |
| 541908 | 581587 | 22138 | BAKING SET 9 PIECE RETROSPOT | 3 | |

| | InvoiceDate | UnitPrice | CustomerID | Country |
|---|---|---|---|---|
| 0 | 12/1/2010 8:26 | 2.55 | 17850.0 | United Kingdom |

```
1         12/1/2010 8:26       3.39     17850.0  United Kingdom
2         12/1/2010 8:26       2.75     17850.0  United Kingdom
3         12/1/2010 8:26       3.39     17850.0  United Kingdom
4         12/1/2010 8:26       3.39     17850.0  United Kingdom
…                    …          …          …              …
541904   12/9/2011 12:50       0.85     12680.0          France
541905   12/9/2011 12:50       2.10     12680.0          France
541906   12/9/2011 12:50       4.15     12680.0          France
541907   12/9/2011 12:50       4.15     12680.0          France
541908   12/9/2011 12:50       4.95     12680.0          France

[541909 rows x 8 columns]
```

[ ]:
```
''' getting the columns present in the datafile '''

df.columns
```

[ ]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

[ ]:
```
'Checking the shape number of rows and number of columns in dataframe'

df.shape
```

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

[ ]:
```
' checking the information related to columns like datatype and constraints '
df.info()
```

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

[ ]:
```
'Checking the aggregation values for numerical columns '

df.describe(include = np.number)
```

[ ]:
```
            Quantity      UnitPrice     CustomerID
count  541909.000000  541909.000000  406829.000000
mean        9.552250       4.611114   15287.690570
std       218.081158      96.759853    1713.600303
min    -80995.000000  -11062.060000   12346.000000
25%         1.000000       1.250000   13953.000000
50%         3.000000       2.080000   15152.000000
75%        10.000000       4.130000   16791.000000
max     80995.000000   38970.000000   18287.000000
```

[ ]:
```
'Checking the aggregation values for categorical columns '

df.describe(include = ['object'])
```

[ ]:
```
       InvoiceNo StockCode                          Description  \
count     541909    541909                               540455
unique     25900      4070                                 4223
top       573585    85123A  WHITE HANGING HEART T-LIGHT HOLDER
freq        1114      2313                                 2369

             InvoiceDate         Country
count             541909          541909
unique             23260              38
top     10/31/2011 14:41  United Kingdom
freq                1114          495478
```

[ ]:
```
' Checking for null values in the dataframe '

df.isna().sum()
```

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

[ ]:
```
' Checking the percentage of null values by each column w.r.t whole data  '

df.isnull().sum()/len(df)*100
```

[ ]:
```
InvoiceNo       0.000000
StockCode       0.000000
Description     0.268311
Quantity        0.000000
InvoiceDate     0.000000
UnitPrice       0.000000
CustomerID     24.926694
Country         0.000000
dtype: float64
```

[ ]:
```
' Filtering the null values for Customerid column '

df[df['CustomerID'].isna()]
```

[ ]:
```
        InvoiceNo StockCode                       Description  Quantity  \
622        536414     22139                               NaN        56
1443       536544     21773  DECORATIVE ROSE BATHROOM BOTTLE         1
1444       536544     21774  DECORATIVE CATS BATHROOM BOTTLE         2
1445       536544     21786               POLKADOT RAIN HAT          4
1446       536544     21787            RAIN PONCHO RETROSPOT         2
…             …        …                                  …        …
541536     581498    85099B          JUMBO BAG RED RETROSPOT         5
541537     581498    85099C   JUMBO  BAG BAROQUE BLACK WHITE         4
541538     581498     85150    LADIES & GENTLEMEN METAL SIGN         1
541539     581498     85174                S/4 CACTI CANDLES         1
541540     581498       DOT                   DOTCOM POSTAGE         1

             InvoiceDate  UnitPrice  CustomerID         Country
622     12/1/2010 11:52       0.00         NaN  United Kingdom
1443    12/1/2010 14:32       2.51         NaN  United Kingdom
1444    12/1/2010 14:32       2.51         NaN  United Kingdom
1445    12/1/2010 14:32       0.85         NaN  United Kingdom
1446    12/1/2010 14:32       1.66         NaN  United Kingdom
…                  …        …          …             …
541536  12/9/2011 10:26       4.13         NaN  United Kingdom
541537  12/9/2011 10:26       4.13         NaN  United Kingdom
541538  12/9/2011 10:26       4.96         NaN  United Kingdom
541539  12/9/2011 10:26      10.79         NaN  United Kingdom
541540  12/9/2011 10:26    1714.17         NaN  United Kingdom

[135080 rows x 8 columns]
```

[ ]:
```
'Country wise number of customers'

df[df['CustomerID'].isna()]['Country'].value_counts()
```

[ ]:
```
Country
United Kingdom    133600
EIRE                 711
Hong Kong            288
Unspecified          202
Switzerland          125
France                66
Israel                47
Portugal              39
Bahrain                2
Name: count, dtype: int64
```

[ ]:
```
' making a copy od dataframe '

df1 = df.copy()
```

[ ]:
```
' Checking for null values in customerid column '

df['CustomerID'].isna().sum()
```

[ ]:
```
135080
```

[ ]:
```
'calculating mode of customerid for each country'

df.groupby(['Country'])['CustomerID'].agg(pd.Series.mode)
```

[ ]:
```
Country
Australia               12415.0
Austria                 12360.0
Bahrain                 12355.0
Belgium                 12362.0
Brazil                  12769.0
Canada                  17444.0
Channel Islands         14936.0
Cyprus                  12359.0
Czech Republic          12781.0
Denmark                 12406.0
EIRE                    14911.0
European Community      15108.0
Finland                 12428.0
France                  12681.0
Germany                 12471.0
Greece                  12717.0
Hong Kong                    []
Iceland                 12347.0
Israel                  12688.0
Italy                   12584.0
```

```
Japan                   12753.0
Lebanon                 12764.0
Lithuania               15332.0
Malta                   17828.0
Netherlands             14646.0
Norway                  12433.0
Poland                  12779.0
Portugal                12757.0
RSA                     12446.0
Saudi Arabia            12565.0
Singapore               12744.0
Spain                   12540.0
Sweden                  17404.0
Switzerland             12451.0
USA                     12607.0
United Arab Emirates    12739.0
United Kingdom          17841.0
Unspecified             12743.0
Name: CustomerID, dtype: object
```

[ ]:
```
' Customer id null values for each country '

df[df['CustomerID'].isna()]['Country'].value_counts()
```

[ ]:
```
Country
United Kingdom    133600
EIRE                 711
Hong Kong            288
Unspecified          202
Switzerland          125
France                66
Israel                47
Portugal              39
Bahrain                2
Name: count, dtype: int64
```

[ ]:
```
df['CustomerID'].mode()
```

[ ]:
```
0    17841.0
Name: CustomerID, dtype: float64
```

[ ]:
```
'changing the datatype of invoice date to datetime'

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

[ ]:
```
' we got different dtype on changing'
```

```
df['InvoiceDate'].dtype
```

[ ]:
```
dtype('<M8[ns]')
```

[ ]:
```
' on checking both two datatypes are same '

np.dtype('<M8[ns]') == np.dtype('datetime64[ns]')
```

[ ]:
```
True
```

[ ]:
```
'Function to return mode of each country customerid  and fill it with null␣
 ↪values for customerid'

def fillna_with_mode(series):
    mode_values = series.mode()
    if not mode_values.empty:
        return series.fillna(mode_values.iloc[0])
    else:
        return series.fillna("Unknown")  # Provide a default value when mode is␣
 ↪not available

# Replace null values in 'customer id' column with mode for each country
df['CustomerID'] = df.groupby('Country')['CustomerID'].
 ↪transform(fillna_with_mode).reset_index(drop=True)
```

[ ]:
```
' Now null values for customerid is 0 '

df.isna().sum()
```

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

[ ]:
```
'replacing null values in description to unknown description '

df['Description'].fillna('Unknown Description',inplace = True)
```

[ ]:
```
'all null values are replaced with respective values'

df.isna().sum()
```

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

[ ]:
```
'calculating the revenue column'

df['revenue'] = df['Quantity'] * df['UnitPrice']
```

[ ]:
```
''' here we can see that most of negative revenue that is cost spend by company␣
 ↪for orders that are  returned or cancelled orders '''

df[df['revenue'] < 0]['Country'].value_counts()
```

[ ]:
```
Country
United Kingdom        7858
Germany                453
EIRE                   302
France                 149
USA                    112
Australia               74
Spain                   48
Italy                   45
Belgium                 38
Japan                   37
Switzerland             35
Portugal                18
Malta                   15
Norway                  14
Poland                  11
Sweden                  11
Finland                 10
Channel Islands         10
Denmark                  9
Cyprus                   8
Netherlands              8
Singapore                7
Czech Republic           5
Hong Kong                4
Austria                  3
Israel                   2
Saudi Arabia             1
```

```
Bahrain                 1
European Community      1
Greece                  1
Name: count, dtype: int64
```

[ ]:
```
'corelation between quantity and revenue'

df[['Quantity','revenue']].corr()
```

[ ]:
```
           Quantity   revenue
Quantity   1.000000  0.886681
revenue    0.886681  1.000000
```

the corelation between Quantity and revenue is fairly positive and increasing Quantity will increase revenue

[ ]:
```
'heatmap for quantity and revenue'

sns.heatmap(df[['Quantity','revenue']].corr(),annot = True)
```

[ ]:
```
<Axes: >
```

[ ]:
```
'corelation for unitprice and revenue '

df[['UnitPrice','revenue']].corr()
```

[ ]:
```
            UnitPrice   revenue
UnitPrice    1.000000 -0.162029
revenue     -0.162029  1.000000
```

the corelation factor is -0.16 which is negative that means increase in unit price wil decrease the revenue

[ ]:
```
'heatmap for revenue and unitprice corelation'

sns.heatmap(df[['UnitPrice','revenue']].corr(),annot = True)
```

[ ]: <Axes: >


[ ]:
```
'filtering the cancelled or returned orders'

df[df['Quantity']< 0]
```

[ ]:

| | InvoiceNo | StockCode | Description | Quantity | \ |
|---|---|---|---|---|---|
| 141 | C536379 | D | Discount | -1 | |
| 154 | C536383 | 35004C | SET OF 3 COLOURED FLYING DUCKS | -1 | |
| 235 | C536391 | 22556 | PLASTERS IN TIN CIRCUS PARADE | -12 | |
| 236 | C536391 | 21984 | PACK OF 12 PINK PAISLEY TISSUES | -24 | |
| 237 | C536391 | 21983 | PACK OF 12 BLUE PAISLEY TISSUES | -24 | |
| … | … | … | … | … | |
| 540449 | C581490 | 23144 | ZINC T-LIGHT HOLDER STARS SMALL | -11 | |
| 541541 | C581499 | M | Manual | -1 | |
| 541715 | C581568 | 21258 | VICTORIAN SEWING BOX LARGE | -5 | |
| 541716 | C581569 | 84978 | HANGING HEART JAR T-LIGHT HOLDER | -1 | |
| 541717 | C581569 | 20979 | 36 PENCILS TUBE RED RETROSPOT | -5 | |

| | InvoiceDate | UnitPrice | CustomerID | Country | revenue |
|---|---|---|---|---|---|
| 141 | 2010-12-01 09:41:00 | 27.50 | 14527.0 | United Kingdom | -27.50 |
| 154 | 2010-12-01 09:49:00 | 4.65 | 15311.0 | United Kingdom | -4.65 |
| 235 | 2010-12-01 10:24:00 | 1.65 | 17548.0 | United Kingdom | -19.80 |
| 236 | 2010-12-01 10:24:00 | 0.29 | 17548.0 | United Kingdom | -6.96 |
| 237 | 2010-12-01 10:24:00 | 0.29 | 17548.0 | United Kingdom | -6.96 |
| … | … | … | … | … | … |
| 540449 | 2011-12-09 09:57:00 | 0.83 | 14397.0 | United Kingdom | -9.13 |
| 541541 | 2011-12-09 10:28:00 | 224.69 | 15498.0 | United Kingdom | -224.69 |
| 541715 | 2011-12-09 11:57:00 | 10.95 | 15311.0 | United Kingdom | -54.75 |
| 541716 | 2011-12-09 11:58:00 | 1.25 | 17315.0 | United Kingdom | -1.25 |
| 541717 | 2011-12-09 11:58:00 | 1.25 | 17315.0 | United Kingdom | -6.25 |

[10624 rows x 9 columns]

[ ]:
```
# percentage of returned or cancelled products '''

len(df[df['Quantity']< 0])/len(df)*100


''' This the orders that are cancelled or returned percentage ~ 2% '''
```

[ ]:
```
' This the orders that are cancelled or returned percentage ~ 2% '
```

[ ]:
```
''' the negative and positive  revenue genereteaed by  product orders '''

a = df['revenue']
neg_rev = 0.0
pos_rev = 0.0
for i in a:
  if i < 0:
    neg_rev += i
  else:
    pos_rev += i
```

```
print(neg_rev,pos_rev, sep = ', ')
```

```
-918936.6099999961, 10666684.544004016
```

[ ]:
```
''' Total revenue generated by all orders '''

df['revenue'].sum()
```

[ ]:
```
9747747.933999998
```

[ ]:
```
10666684.544004016-918936.6099999961
```

[ ]:
```
9747747.93400402
```

The average return rate for ecommerce is typically 20% to 30%. Factors influencing this rate include product dissatisfaction, incorrect sizing, or discrepancies between the product and its online description.

Here the percentage is low and is good for a ecommerce platform for less return or cancelled rate.

[ ]:
```
'freebies given for customers from company side '

df[df['UnitPrice'] == 0]['CustomerID'].value_counts()
```

[ ]:
```
CustomerID
17841.0    2473
14911.0       4
13081.0       4
14646.0       4
13985.0       2
12415.0       2
16560.0       1
15804.0       1
16406.0       1
12444.0       1
12603.0       1
15602.0       1
13014.0       1
12431.0       1
12437.0       1
14110.0       1
18059.0       1
12446.0       1
12748.0       1
15107.0       1
16133.0       1
12647.0       1
```

```
15581.0       1
12507.0       1
16818.0       1
17667.0       1
12457.0       1
14410.0       1
13113.0       1
13239.0       1
17560.0       1
13256.0       1
Name: count, dtype: int64
```

[ ]:
```
# percentage of free products in dataset

a = len(df[df['UnitPrice']== 0])/len(df)*100

a
''' this is the percentage of free products given by the business  ~ 0.5% '''
```

[ ]:
```
' this is the percentage of free products given by the business  ~ 0.5% '
```

[ ]:
```
' extracting the month and year from invoicedate column '

df['Invoice_month'] = pd.DatetimeIndex(df['InvoiceDate']).month
df['Invoice_year'] = pd.DatetimeIndex(df['InvoiceDate']).year
```

[ ]:
```
df
```

[ ]:
```
       InvoiceNo StockCode                          Description  Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1         536365     71053                  WHITE METAL LANTERN         6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
…            …       …                                     …       …
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS        12
541905    581587     22899         CHILDREN'S APRON DOLLY GIRL          6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT          3

                InvoiceDate  UnitPrice CustomerID         Country  revenue  \
0       2010-12-01 08:26:00       2.55    17850.0  United Kingdom    15.30
1       2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
2       2010-12-01 08:26:00       2.75    17850.0  United Kingdom    22.00
3       2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
4       2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
```

```
…                     …        …        …               …       …
541904 2011-12-09 12:50:00       0.85    12680.0          France    10.20
541905 2011-12-09 12:50:00       2.10    12680.0          France    12.60
541906 2011-12-09 12:50:00       4.15    12680.0          France    16.60
541907 2011-12-09 12:50:00       4.15    12680.0          France    16.60
541908 2011-12-09 12:50:00       4.95    12680.0          France    14.85

        Invoice_month  Invoice_year
0                  12          2010
1                  12          2010
2                  12          2010
3                  12          2010
4                  12          2010
…                 …           …
541904             12          2011
541905             12          2011
541906             12          2011
541907             12          2011
541908             12          2011

[541909 rows x 11 columns]
```

[ ]:
```
'revenue generated by each month for each year '

df_year_rev = df.groupby(['Invoice_year','Invoice_month']).agg(revenue =␣
 ↪('revenue','sum')).reset_index()
```

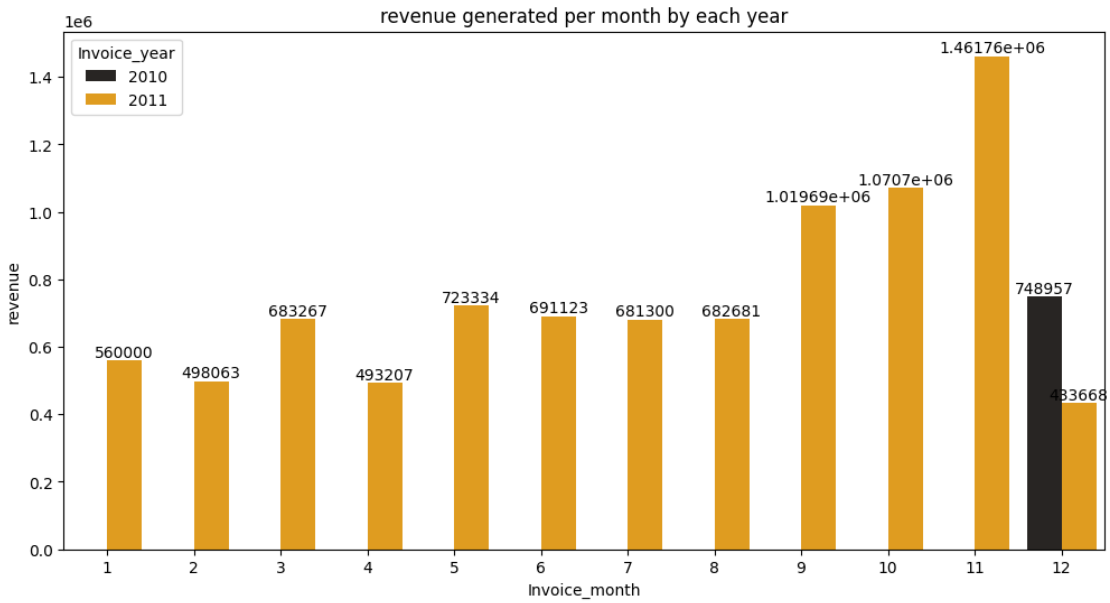
[ ]:
```
df_year_rev
```

[ ]:
```
    Invoice_year  Invoice_month      revenue
0           2010             12   748957.020
1           2011              1   560000.260
2           2011              2   498062.650
3           2011              3   683267.080
4           2011              4   493207.121
5           2011              5   723333.510
6           2011              6   691123.120
7           2011              7   681300.111
8           2011              8   682680.510
9           2011              9  1019687.622
10          2011             10  1070704.670
11          2011             11  1461756.250
12          2011             12   433668.010
```

[108]:
```
'plotting the revenue generated per month by each year'

plt.figure(figsize=(12,6))
```

```
plt.title('revenue generated per month by each year')

a = sns.barplot(data = df_year_rev,x = 'Invoice_month', y = 'revenue',hue =␣
 ↪'Invoice_year',color = 'orange')

for i in a.containers:
  a.bar_label(i,)

plt.show()
```

```
<ipython-input-108-4802b88a127f>:6: FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in
v0.14.0. Set `palette='dark:orange'` for the same effect.

  a = sns.barplot(data = df_year_rev,x = 'Invoice_month', y = 'revenue',hue =
'Invoice_year',color = 'orange')
```




In the above we plotted revenue generated by each month by each year.

months from January to August performed average in case of revenue generation.

From spetember on ward there is sudden spike in graph of revenue and continued this trend upto November.

we can se that November is the month of great revenue creation from the data and there is a sudden drop in December month.

We can obesrve 2 plots in December resembling different years – 2010 and 2011, The revenue in

2011 December has been decreased drastically when compared to 2010 december.

The months from september to November are great months for revenue or sales due to various reasons like discounts,offer days like big billion days etc.

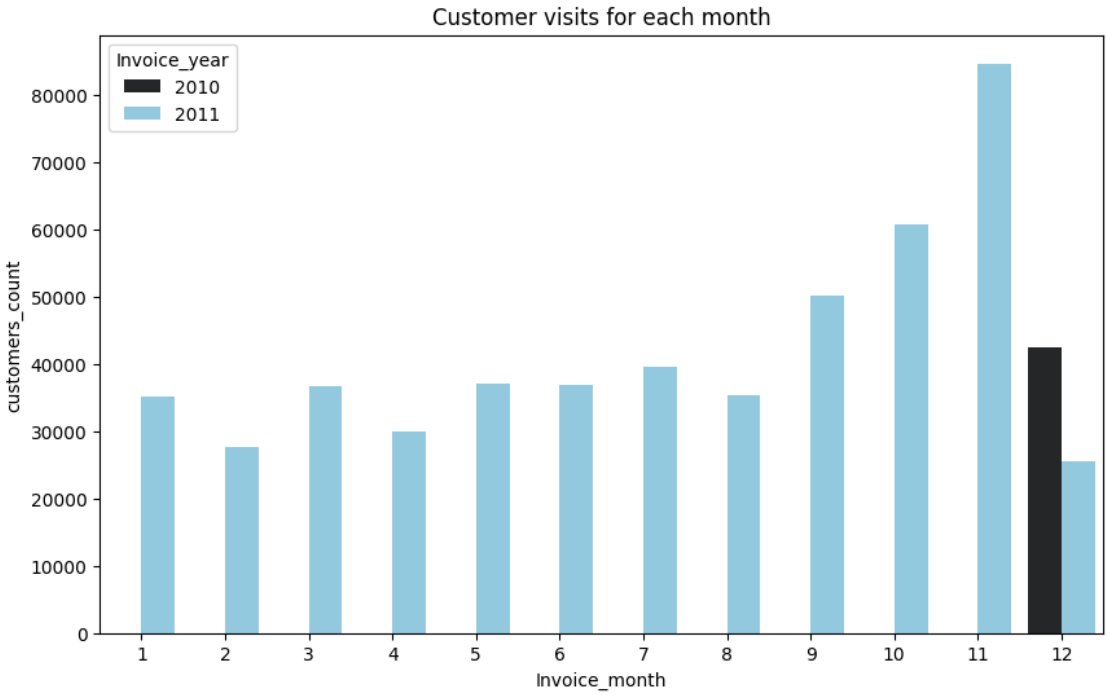
[ ]:
```
'Number of customer visits per each month by each year'

df_cust_count = df.groupby(['Invoice_year','Invoice_month']).
 ↪agg(customers_count = ('CustomerID','count')).reset_index()

df_cust_count
```

[ ]:

| | Invoice_year | Invoice_month | customers_count |
|---|---|---|---|
| 0 | 2010 | 12 | 42481 |
| 1 | 2011 | 1 | 35147 |
| 2 | 2011 | 2 | 27707 |
| 3 | 2011 | 3 | 36748 |
| 4 | 2011 | 4 | 29916 |
| 5 | 2011 | 5 | 37030 |
| 6 | 2011 | 6 | 36874 |
| 7 | 2011 | 7 | 39518 |
| 8 | 2011 | 8 | 35284 |
| 9 | 2011 | 9 | 50226 |
| 10 | 2011 | 10 | 60742 |
| 11 | 2011 | 11 | 84711 |
| 12 | 2011 | 12 | 25525 |

[109]:
```
'plotting the number of customers visits or interactions  per each month '

plt.figure(figsize = (10,6))
plt.title('Customer visits for each month')

a = sns.barplot(data = df_cust_count, x = 'Invoice_month',y =␣
 ↪'customers_count',hue = 'Invoice_year',color = 'skyblue')
plt.show()

a
```

```
<ipython-input-109-8bd4b848a84f>:6: FutureWarning:

Setting a gradient palette using color= is deprecated and will be removed in
v0.14.0. Set `palette='dark:skyblue'` for the same effect.

  a = sns.barplot(data = df_cust_count, x = 'Invoice_month',y =
'customers_count',hue = 'Invoice_year',color = 'skyblue')
```




[109]:
```
<Axes: title={'center': 'Customer visits for each month'},
xlabel='Invoice_month', ylabel='customers_count'>
```

In this plot we can see the customers interactions or visits for website per each month by each year.

The customer interaction with website is quite normal in the months like from January to August.

From september to November the customer interactions are very high which made these months generate more revenue in the previous section.

November is the month for higher visits of customers to website and also for more conversion.

the customers interaction with website is fluctuating in month of december for both years 2010 and 2011 . There is a decrease in visits by customer in month of december.

.

#RFM Analysis

The "RFM" in RFM analysis stands for recency, frequency and monetary value. RFM analysis is a way to use data based on existing customer behavior to predict how a new customer is likely to act in the future. An RFM model is built using three key factors:

**Recency** : how recently a customer has transacted with a brand

**Frequency** : how frequently they've engaged with a brand

**Monetary** : how much money they've spent on a brand's products and services

RFM analysis was born out of direct mail marketing, in particular a 1995 article by Tom Wansbeek

and Jan Roelf Bult titled “Optimal Selection for Direct Mail,” which was published in the journal Marketing Science. Their work helped confirm the Pareto Principle — the idea widely held among marketers that 80% of sales come from 20% of a brand’s customers.

[ ]:
```
' finding the recenecy date for each customer '

df_recency = df.groupby('CustomerID')['InvoiceDate'].max().reset_index()

recent_date = df['InvoiceDate'].max()

df_recency['recency'] = df_recency['InvoiceDate'].apply(lambda x : (recent_date␣
 ↪- x).days)
df_recency.sort_values(by = 'recency')
```

[ ]:
```
      CustomerID         InvoiceDate  recency
3925    17675.0 2011-12-08 18:03:00        0
2320    15484.0 2011-12-08 15:03:00        0
2527    15755.0 2011-12-08 18:59:00        0
1534    14422.0 2011-12-09 11:26:00        0
1674    14606.0 2011-12-08 19:28:00        0
…          …                   …       …
4140    17968.0 2010-12-01 12:23:00      373
4096    17908.0 2010-12-01 11:45:00      373
1046    13747.0 2010-12-01 10:37:00      373
4212    18074.0 2010-12-01 09:53:00      373
359     12791.0 2010-12-01 11:27:00      373

[4373 rows x 3 columns]
```

[ ]:
```
' finding the frequncy value for each customer '

df_frequency = df.groupby('CustomerID')['InvoiceDate'].count().reset_index()


df_frequency.rename(columns = {'InvoiceDate' : 'Frequency'},inplace = True)

df_frequency.sort_values(by = 'Frequency',ascending = False)
```

[ ]:
```
      CustomerID  Frequency
4042    17841.0     141583
1895    14911.0       6614
1300    14096.0       5128
330     12748.0       4642
1674    14606.0       2782
…          …        …
3351    16881.0          1
4207    18068.0          1
577     13099.0          1
```

```
127     12505.0          1
3435    16995.0          1

[4373 rows x 2 columns]
```

[ ]:
```
' Finding the monetory values for each customer '

df_monetory = df.groupby(['CustomerID'])['revenue'].sum().reset_index()

df_monetory


df_monetory.rename(columns ={'revenue' : 'monetory'},inplace = True)
df_monetory.sort_values(by = 'monetory',ascending = False)
```

[ ]:
```
      CustomerID    monetory
4042    17841.0  1460273.75
1703    14646.0   279489.02
4233    18102.0   256438.49
3758    17450.0   187482.17
1895    14911.0   145564.22
…         …         …
125     12503.0    -1126.00
3870    17603.0    -1165.30
1384    14213.0    -1192.20
2236    15369.0    -1592.49
3756    17448.0    -4287.63

[4373 rows x 2 columns]
```

[ ]:
```
'concating all three recency,frequency and monetory df with respect to␣
 ↪customerid '

df_rfm = pd.concat([df_recency,df_frequency,df_monetory],axis = 1,join =␣
 ↪'inner')
df_rfm
```

[ ]:
```
      CustomerID         InvoiceDate  recency CustomerID  Frequency CustomerID  \
0        12346.0 2011-01-18 10:17:00      325    12346.0          2    12346.0
1        12347.0 2011-12-07 15:52:00        1    12347.0        182    12347.0
2        12348.0 2011-09-25 13:13:00       74    12348.0         31    12348.0
3        12349.0 2011-11-21 09:51:00       18    12349.0         73    12349.0
4        12350.0 2011-02-02 16:01:00      309    12350.0         17    12350.0
…          …                   …       …         …         …         …
4368     18281.0 2011-06-12 10:53:00      180    18281.0          7    18281.0
4369     18282.0 2011-12-02 11:43:00        7    18282.0         13    18282.0
4370     18283.0 2011-12-06 12:02:00        3    18283.0        756    18283.0
```

```
4371    18287.0 2011-10-28 09:29:00       42    18287.0         70    18287.0
4372    Unknown 2011-11-14 13:27:00       24    Unknown        288    Unknown

      monetory
0         0.00
1      4310.00
2      1797.24
3      1757.55
4       334.40
…          …
4368     80.82
4369    176.60
4370   2094.88
4371   1837.28
4372  10117.04

[4373 rows x 7 columns]
```

[ ]:
```
df_rfm['Frequency'].value_counts().sort_values()
```

[ ]:
```
Frequency
288     1
721     1
372     1
321     1
671     1
       ..
7      72
12     72
10     74
6      78
1      79
Name: count, Length: 476, dtype: int64
```

[ ]:
```
df_rfm['Frequency'].max()
```

[ ]:
```
141583
```

[ ]:
```
df_rfm['Frequency'].min()
```

[ ]:
```
1
```

[ ]:
```
df_monetory.sort_values(by = 'monetory',ascending = False)
```

[ ]:
```
     CustomerID    monetory
4042    17841.0  1460273.75
1703    14646.0   279489.02
```

```
4233    18102.0   256438.49
3758    17450.0   187482.17
1895    14911.0   145564.22
…           …          …
125     12503.0    -1126.00
3870    17603.0    -1165.30
1384    14213.0    -1192.20
2236    15369.0    -1592.49
3756    17448.0    -4287.63

[4373 rows x 2 columns]
```

[ ]:
```
' Creating bins for each Recency,frequency and monetary values and creating␣
 ↪scores for each of RFM '

bin = [-1,50,100,150,200,250,300,400]
label = [7,6,5,4,3,2,1]

bin_f = [0,1000,2000,3000,4000,5000,20000,500000]
label_f = [1,2,3,4,5,6,7]

bin_m = [-5000,0,50000,100000,150000,250000,300000,400000,1500000]
label_m = [0,1,2,3,4,5,6,7]



df_rfm['recency_score'] = pd.cut(df_rfm['recency'],bins = bin,labels = label)
df_rfm['Frequency_score'] = pd.cut(df_rfm['Frequency'],bins = bin_f,labels =␣
 ↪label_f)
df_rfm['monetory_score'] = pd.cut(df_rfm['monetory'],bins = bin_m,labels =␣
 ↪label_m)
```

[ ]:
```
df_rfm
```

[ ]:
```
     CustomerID         InvoiceDate  recency CustomerID  Frequency CustomerID  \
0       12346.0 2011-01-18 10:17:00      325    12346.0          2    12346.0
1       12347.0 2011-12-07 15:52:00        1    12347.0        182    12347.0
2       12348.0 2011-09-25 13:13:00       74    12348.0         31    12348.0
3       12349.0 2011-11-21 09:51:00       18    12349.0         73    12349.0
4       12350.0 2011-02-02 16:01:00      309    12350.0         17    12350.0
…           …                  …      …         …        …        …
4368    18281.0 2011-06-12 10:53:00      180    18281.0          7    18281.0
4369    18282.0 2011-12-02 11:43:00        7    18282.0         13    18282.0
4370    18283.0 2011-12-06 12:02:00        3    18283.0        756    18283.0
4371    18287.0 2011-10-28 09:29:00       42    18287.0         70    18287.0
4372    Unknown 2011-11-14 13:27:00       24    Unknown        288    Unknown
```

```
      monetory recency_score Frequency_score monetory_score
0         0.00             1                1              0
1      4310.00             7                1              1
2      1797.24             6                1              1
3      1757.55             7                1              1
4       334.40             1                1              1
…          …             …                …              …
4368     80.82             4                1              1
4369    176.60             7                1              1
4370   2094.88             7                1              1
4371   1837.28             7                1              1
4372  10117.04             7                1              1

[4373 rows x 10 columns]
```

[ ]:
```
df_rfm[df_rfm['Frequency_score'] == 1]
```

[ ]:
```
     CustomerID         InvoiceDate  recency CustomerID  Frequency CustomerID  \
0       12346.0 2011-01-18 10:17:00      325    12346.0          2    12346.0
1       12347.0 2011-12-07 15:52:00        1    12347.0        182    12347.0
2       12348.0 2011-09-25 13:13:00       74    12348.0         31    12348.0
3       12349.0 2011-11-21 09:51:00       18    12349.0         73    12349.0
4       12350.0 2011-02-02 16:01:00      309    12350.0         17    12350.0
…           …                   …      …         …        …        …
4368    18281.0 2011-06-12 10:53:00      180    18281.0          7    18281.0
4369    18282.0 2011-12-02 11:43:00        7    18282.0         13    18282.0
4370    18283.0 2011-12-06 12:02:00        3    18283.0        756    18283.0
4371    18287.0 2011-10-28 09:29:00       42    18287.0         70    18287.0
4372    Unknown 2011-11-14 13:27:00       24    Unknown        288    Unknown

      monetory recency_score Frequency_score monetory_score
0         0.00             1                1              0
1      4310.00             7                1              1
2      1797.24             6                1              1
3      1757.55             7                1              1
4       334.40             1                1              1
…          …             …                …              …
4368     80.82             4                1              1
4369    176.60             7                1              1
4370   2094.88             7                1              1
4371   1837.28             7                1              1
4372  10117.04             7                1              1

[4351 rows x 10 columns]
```

Here we calculated the RFM score for each customer based on their values for recenecy,frequency and monetory calculated.

for instance, at index -1 ,customerid - [12347.0]

look at this customer where his recency score – 7 which implies he's last visit is very recent and by seeing the value for recency we get 1 day of recency score.

the frequency score for this customer is 1 which implies the number of visits is very less to the standard we implied.On seeing the frequency column it is 182 which is comparitvely less for a customer

the monetory score for this customer is also 1 which implies he spend les amount on purchasing items. On seeing through the monetory column its 4310.00 which is less compared to other customers.

In the above, we took an example of a customer and pulled his RFM score and understood how RFM analysis works for a customer in ecommerce.

[ ]:
```
''' country wise  revenue and customers count '''

df_con = df.groupby(['Country']).agg(
                            consumer_count = ('InvoiceDate','count'),
                            revenue = ('revenue','sum')).
 ↪sort_values(['revenue','consumer_count'],ascending = False).reset_index()
```

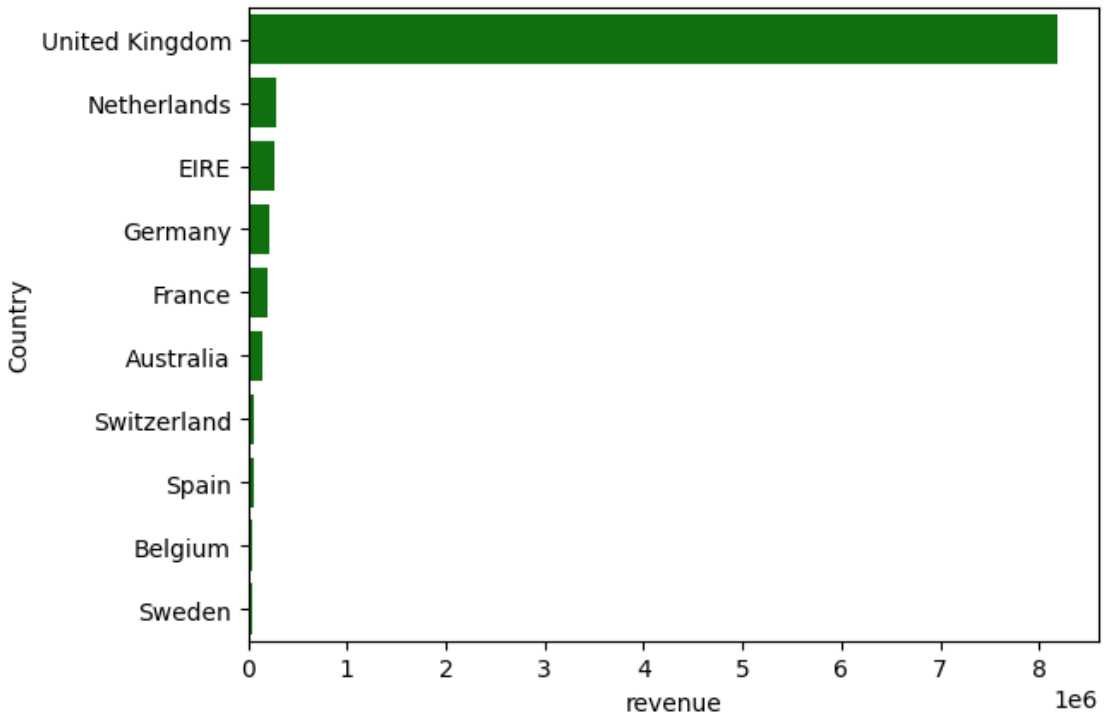
[ ]:
```
'Top 10 countries with high revenue generation'

a = df_con.sort_values(by='revenue',ascending = False)
a = a.head(10)
a
```

[ ]:

| | Country | consumer_count | revenue |
|---|---|---|---|
| 0 | United Kingdom | 495478 | 8187806.364 |
| 1 | Netherlands | 2371 | 284661.540 |
| 2 | EIRE | 8196 | 263276.820 |
| 3 | Germany | 9495 | 221698.210 |
| 4 | France | 8557 | 197403.900 |
| 5 | Australia | 1259 | 137077.270 |
| 6 | Switzerland | 2002 | 56385.350 |
| 7 | Spain | 2533 | 54774.580 |
| 8 | Belgium | 2069 | 40910.960 |
| 9 | Sweden | 462 | 36595.910 |

[ ]:
```
'Country wise revenue generated'

sns.barplot(data = a,y = 'Country',x = 'revenue',color = 'green')
```

[ ]:
```
<Axes: xlabel='revenue', ylabel='Country'>
```




The above plot shows the revenue generated by top 10 country.

we can see that unoited Kingdom has produced more revenue when compared to other countries.

Next to UK, Netherland is in second position for generatiing of more revenue or more sales.

All the countries have similar sales or revenue where as UK stands alone with more revenue generation.

[ ]:
```
'country wise number of customers'

b = df_con.sort_values(by ='consumer_count',ascending = False)
b = b.head(10)
b
```

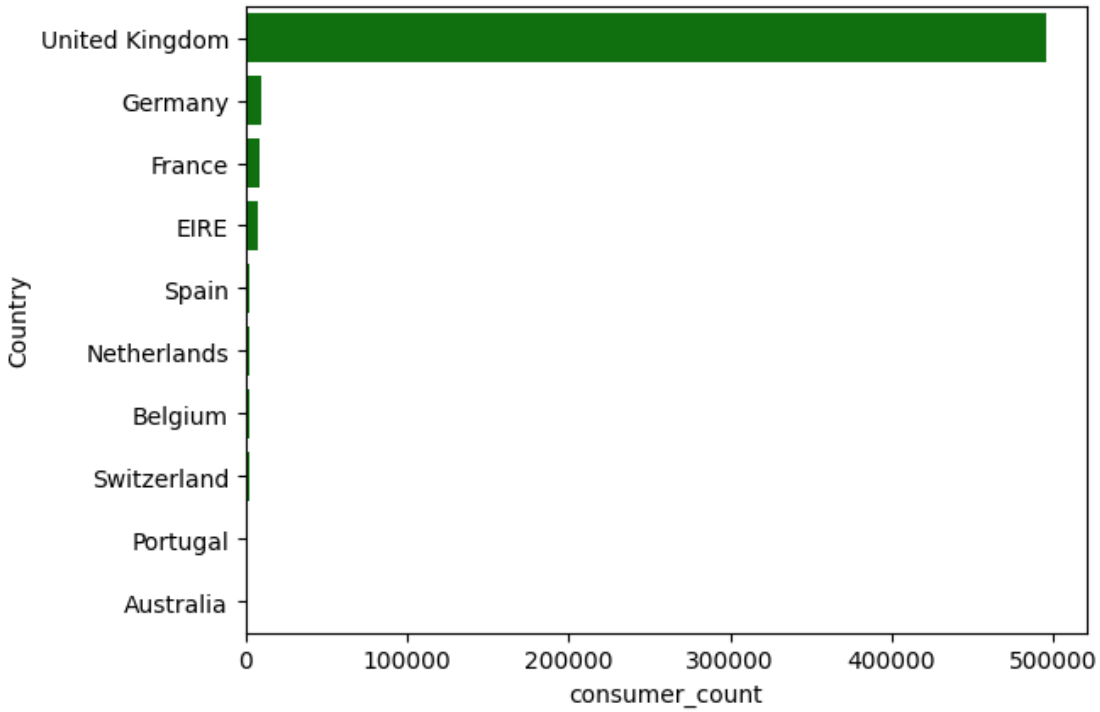
[ ]:

| | Country | consumer_count | revenue |
|---|---|---|---|
| 0 | United Kingdom | 495478 | 8187806.364 |
| 3 | Germany | 9495 | 221698.210 |
| 4 | France | 8557 | 197403.900 |
| 2 | EIRE | 8196 | 263276.820 |
| 7 | Spain | 2533 | 54774.580 |
| 1 | Netherlands | 2371 | 284661.540 |
| 8 | Belgium | 2069 | 40910.960 |
| 6 | Switzerland | 2002 | 56385.350 |
| 12 | Portugal | 1519 | 29367.020 |

```
5        Australia            1259    137077.270
```

[ ]:
```
sns.barplot(data = b,y = 'Country',x = 'consumer_count',color = 'green')
```

[ ]:
```
<Axes: xlabel='consumer_count', ylabel='Country'>
```




The above plot shows the number of customer visits for top 10 countries

clearly, th United Kingdom leads in the race with almost 500000 customer visits within the given period of time.

Germany and france follows the Uk with 2 and 3 position,

Due to high customer visists Uk is succeded in conversion of the customers which resulted in more sales or revenu generation.

[ ]:
```
''' Top 10 countries where return or cancelled products orders are high '''

df_ret = df[df['Quantity'] < 0]['Country'].value_counts().reset_index().head(10)
df_ret
```

[ ]:
```
          Country  count
0  United Kingdom   9192
1         Germany    453
2            EIRE    302
```

```
3          France    149
4             USA    112
5       Australia     74
6           Spain     48
7           Italy     45
8         Belgium     38
9           Japan     37
```

[ ]:
```
' Number of returned orders '

len(df[df['Quantity'] < 0])
```

[ ]:
```
10624
```

[ ]:
```
' percenatge of return orders for each country '

df_ret['return_percent'] = df_ret['count']/len(df[df['Quantity'] < 0])*100
```

[ ]:
```
df_ret
```

[ ]:
```
          Country  count  return_percent
0  United Kingdom   9192       86.521084
1         Germany    453        4.263931
2            EIRE    302        2.842620
3          France    149        1.402485
4             USA    112        1.054217
5       Australia     74        0.696536
6           Spain     48        0.451807
7           Italy     45        0.423569
8         Belgium     38        0.357681
9           Japan     37        0.348268
```

##Analysis of returned or cancelled order by each customer

[ ]:
```
' filetring the customers where the oreders are returne dor cancelled '

df[df['Quantity'] < 0].sort_values(by = 'Quantity',ascending = True)
```

[ ]:
```
       InvoiceNo StockCode                          Description  Quantity  \
540422   C581484     23843          PAPER CRAFT , LITTLE BIRDIE    -80995
61624    C541433     23166       MEDIUM CERAMIC TOP STORAGE JAR    -74215
225529    556690     23005         printing smudges/thrown away     -9600
225530    556691     23005         printing smudges/thrown away     -9600
4287     C536757     84347  ROTATING SILVER ANGELS T-LIGHT HLDR     -9360
…            …       …                                    …       …
240697   C558112     22796        PHOTO FRAME 3 CLASSIC HANGING        -1
240696   C558112     23091           ZINC HERB GARDEN CONTAINER        -1
```

```
240694   C558112     82486    WOOD S/3 CABINET ANT WHITE FINISH        -1
242447   C558347         S                              SAMPLES        -1
249284   C558897         M                               Manual        -1

                InvoiceDate  UnitPrice CustomerID         Country    revenue  \
540422 2011-12-09 09:27:00       2.08    16446.0  United Kingdom -168469.60
61624  2011-01-18 10:17:00       1.04    12346.0  United Kingdom  -77183.60
225529 2011-06-14 10:37:00       0.00    17841.0  United Kingdom      -0.00
225530 2011-06-14 10:37:00       0.00    17841.0  United Kingdom      -0.00
4287   2010-12-02 14:23:00       0.03    15838.0  United Kingdom    -280.80
…                      …          …          …               …         …
240697 2011-06-26 16:08:00       9.95    17114.0  United Kingdom      -9.95
240696 2011-06-26 16:08:00       6.25    17114.0  United Kingdom      -6.25
240694 2011-06-26 16:08:00       8.95    17114.0  United Kingdom      -8.95
242447 2011-06-28 14:47:00       9.90    17841.0  United Kingdom      -9.90
249284 2011-07-04 15:55:00     389.68    12619.0         Germany    -389.68

        Invoice_month  Invoice_year
540422             12          2011
61624               1          2011
225529              6          2011
225530              6          2011
4287               12          2010
…                 …           …
240697              6          2011
240696              6          2011
240694              6          2011
242447              6          2011
249284              7          2011

[10624 rows x 11 columns]
```

[ ]:
```
'Analysis for customer who cancelled orders'

df[df['CustomerID'] == 16446.0]
```

[ ]:
```
       InvoiceNo StockCode                  Description  Quantity  \
194354    553573     22980       PANTRY SCRUBBING BRUSH         1
194355    553573     22982          PANTRY PASTRY BRUSH         1
540421    581483     23843  PAPER CRAFT , LITTLE BIRDIE     80995
540422   C581484     23843  PAPER CRAFT , LITTLE BIRDIE    -80995

                InvoiceDate  UnitPrice CustomerID         Country    revenue  \
194354 2011-05-18 09:52:00       1.65    16446.0  United Kingdom       1.65
194355 2011-05-18 09:52:00       1.25    16446.0  United Kingdom       1.25
540421 2011-12-09 09:15:00       2.08    16446.0  United Kingdom  168469.60
540422 2011-12-09 09:27:00       2.08    16446.0  United Kingdom -168469.60
```

```
        Invoice_month  Invoice_year
194354              5          2011
194355              5          2011
540421             12          2011
540422             12          2011
```

Here we are considering the analysis for customerid – 16446

we can see this customer has placed orders of quantity 80995 for product " paper carft- little birdie" on date 2011-12-09 at 9:15 Am.

He cancelled order on same day at 9:27 AM which resembles this order is placed and returne don same date with very less time difference of approximately of 12 minutes.

There is a high chance of customer placed orderby mistake or he mis-typed the quantity.

There may be defect on both side with customer as well as from webiste side

The common reasons like user experience,product description, loading time taken by website, ping or lag by the payment or cart page where he might be added incorrectly product are some cancelled orders reasons.

[ ]:
```
df[(df['CustomerID'] == 17841.0) & (df['Quantity']< 0)]
```

[ ]:
```
       InvoiceNo StockCode                Description  Quantity  \
1441     C536543     22632  HAND WARMER RED RETROSPOT        -1
1442     C536543     22355  CHARLOTTE BAG SUKI DESIGN        -2
2406      536589     21777        Unknown Description       -10
4347      536764    84952C        Unknown Description       -38
6782     C536979     84685      BEACH HUT KEY CABINET        -1
…            …       …                          …        …
535333    581210     23395                      check       -26
535335    581212     22578                       lost     -1050
535336    581213     22576                      check       -30
536908    581226     23090                    missing      -338
538919    581422     23169                    smashed      -235

                InvoiceDate  UnitPrice CustomerID         Country  revenue  \
1441   2010-12-01 14:30:00       2.10    17841.0  United Kingdom    -2.10
1442   2010-12-01 14:30:00       0.85    17841.0  United Kingdom    -1.70
2406   2010-12-01 16:50:00       0.00    17841.0  United Kingdom    -0.00
4347   2010-12-02 14:42:00       0.00    17841.0  United Kingdom    -0.00
6782   2010-12-03 14:23:00       3.75    17841.0  United Kingdom    -3.75
…                      …          …          …               …        …
535333 2011-12-07 18:36:00       0.00    17841.0  United Kingdom    -0.00
535335 2011-12-07 18:38:00       0.00    17841.0  United Kingdom    -0.00
535336 2011-12-07 18:38:00       0.00    17841.0  United Kingdom    -0.00
536908 2011-12-08 09:56:00       0.00    17841.0  United Kingdom    -0.00
538919 2011-12-08 15:24:00       0.00    17841.0  United Kingdom    -0.00
```

```
        Invoice_month  Invoice_year
1441               12          2010
1442               12          2010
2406               12          2010
4347               12          2010
6782               12          2010
…                 …            …
535333             12          2011
535335             12          2011
535336             12          2011
536908             12          2011
538919             12          2011

[1795 rows x 11 columns]
```

[ ]:
```
'Orders placed by each customer'

df.groupby('CustomerID')['Quantity'].sum().sort_values(ascending = False).
 ↪reset_index()
```

[ ]:
```
     CustomerID  Quantity
0       17841.0    278161
1       14646.0    196719
2       14911.0     83488
3       12415.0     77242
4       17450.0     69029
…          …        …
4368    16252.0      -158
4369    16742.0      -189
4370    14213.0      -244
4371    15823.0      -283
4372    16546.0      -303

[4373 rows x 2 columns]
```

[ ]:
```
' Monthly customer visits '

df_month_cus = df.groupby('Invoice_month')['CustomerID'].count().reset_index()

df_month_cus.rename(columns = {'CustomerID' : 'count'},inplace = True)

df_month_cus
```

[ ]:
```
    Invoice_month  count
0               1  35147
1               2  27707
```

```
2              3  36748
3              4  29916
4              5  37030
5              6  36874
6              7  39518
7              8  35284
8              9  50226
9             10  60742
10            11  84711
11            12  68006
```

[111]:
```
''' The best months for customer enganment with business '''
plt.title('Monthly customer visits')
sns.lineplot(data = df_month_cus,x = 'Invoice_month', y = 'count',color =␣
↪'orange')
```

[111]:
```
<Axes: title={'center': 'Monthly customer visits'}, xlabel='Invoice_month',
ylabel='count'>
```


We can observe from the above plot that on average customers are more buying or interacting with

our business in the months from 6-11 which are generally from JUNE to NOVEMBER are the month with more business.

There are small drops in between JUNE and NOVEMBER but compared to other months its a good position

The first quarter or first four months from JANUARY - APRIL or MAY are dull in business where the customer interactions are low.

The peak of the business is in of 3 months SEPTEMBER,OCTOBER and NOVEMBER.

These low interactions with website is due to lack of discounts or sale days in the first quarter of the year,Accordingly care must be taken to improve these months or period to increase the customer interactions and increase conversion rates.

[ ]:
```
df['InvoiceTime'] =  pd.to_datetime(df['InvoiceDate']).dt.time
```

[ ]:
```
'parsing datetime column extracting hoy=ur values  '

df['InvoiceHour'] = df['InvoiceDate'].dt.hour
```

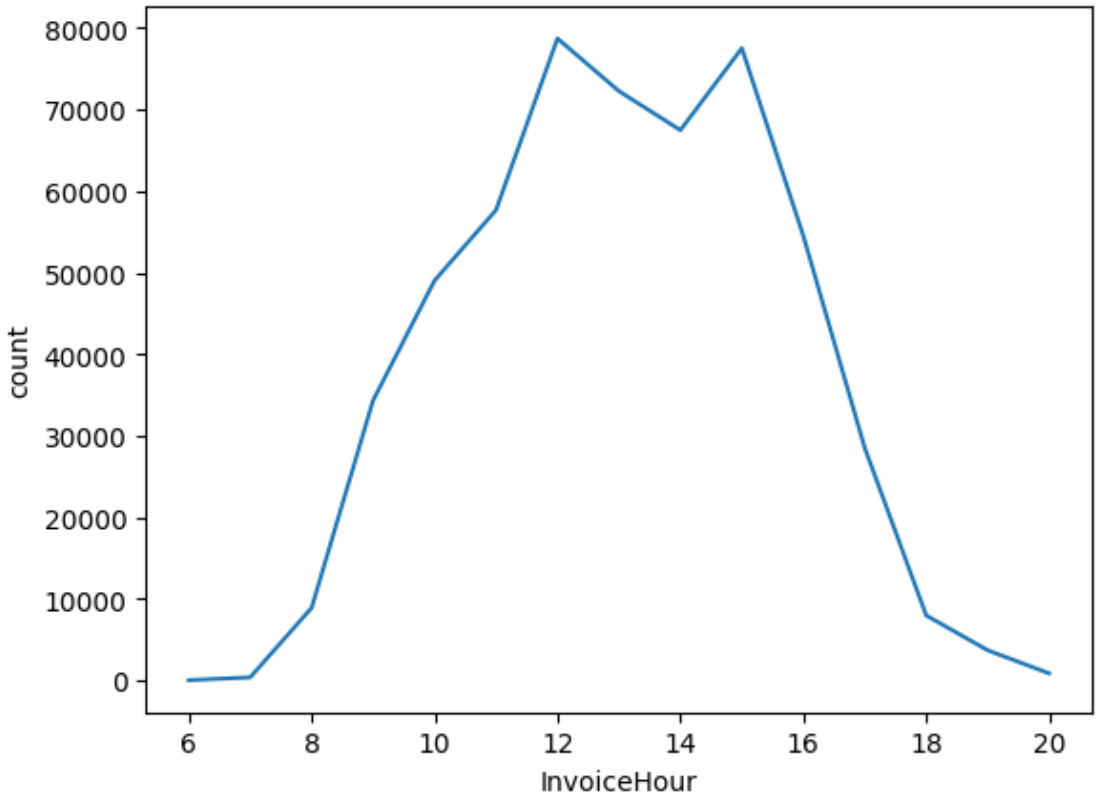
[ ]:
```
''' customers visits per each hour '''

df_hour = df.groupby(df['InvoiceHour'])['InvoiceTime'].count().reset_index()
df_hour.sort_values(by = 'InvoiceTime',ascending = False)
df_hour.rename(columns = {'InvoiceTime' : 'count'},inplace = True)
df_hour
```

[ ]:

| | InvoiceHour | count |
|---|---|---|
| 0 | 6 | 41 |
| 1 | 7 | 383 |
| 2 | 8 | 8909 |
| 3 | 9 | 34332 |
| 4 | 10 | 49037 |
| 5 | 11 | 57674 |
| 6 | 12 | 78709 |
| 7 | 13 | 72259 |
| 8 | 14 | 67471 |
| 9 | 15 | 77519 |
| 10 | 16 | 54516 |
| 11 | 17 | 28509 |
| 12 | 18 | 7974 |
| 13 | 19 | 3705 |
| 14 | 20 | 871 |

[ ]:
```
'Most customer visits per each hour in given data'

sns.lineplot(data = df_hour, x = 'InvoiceHour', y = 'count')
```

[ ]:
```
<Axes: xlabel='InvoiceHour', ylabel='count'>
```




here the plot details about number of visits of customers per each hour.

we cane see form plot that there is steady increase in customer visits from start in 8.00 AM and the business peaked between 10 AM and 6 PM.

there is a small drop in between from 1pm - 2pm and increase in traffic after 2 pm(afternoon)

there is slow decrease in customer visits after 6 PM and maximum time for decrease in customer interaction is after 8PM

[ ]:
```
''' revenue generated per each hour '''

df_hour_rev = df.groupby(['InvoiceHour'])['revenue'].sum().reset_index()

df_hour_rev
```

[ ]:

| | InvoiceHour | revenue |
|---|---|---|
| 0 | 6 | -497.350 |
| 1 | 7 | 31009.320 |
| 2 | 8 | 281840.860 |
| 3 | 9 | 766734.051 |
| 4 | 10 | 1329056.521 |
| 5 | 11 | 1147437.920 |

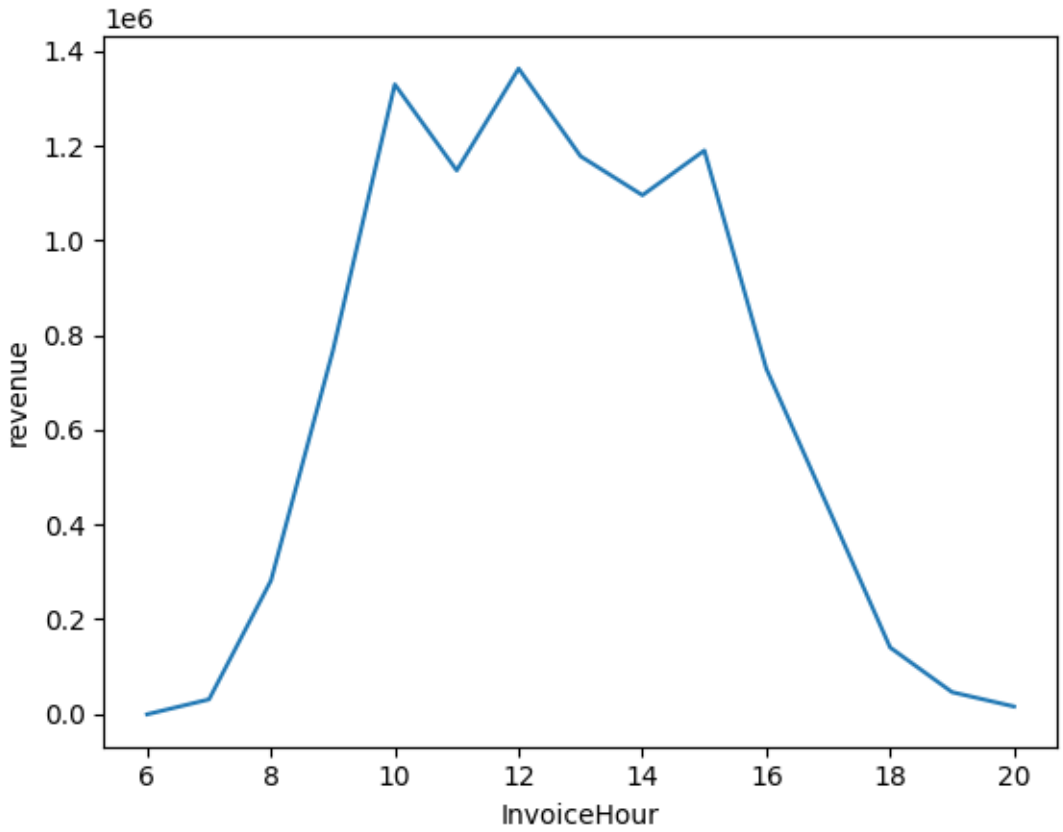
```
6              12  1362484.290
7              13  1177506.370
8              14  1095212.901
9              15  1189458.280
10             16   729140.820
11             17   435444.111
12             18   140574.480
13             19    46324.990
14             20    16020.370
```

[ ]:
```
df['InvoiceTime'].max()
```

[ ]:
```
datetime.time(20, 38)
```

[ ]:
```
' revenue generated per each hour '
sns.lineplot(df_hour_rev,x = 'InvoiceHour',y = 'revenue')
```

[ ]:
```
<Axes: xlabel='InvoiceHour', ylabel='revenue'>
```




The revenue generated per each hour is directly dependent on the number of customer visits.

here with increase in customer visits from 10 AM - 4 PM The revenue generated is also high.

the afternoon to evening time like from 11 AM to 5 PM is the range of period where the revenue generated is high and customer visits are more

[ ]:
```
df_ret = df[df['Quantity']<0]
df_ret
```

[ ]:

| | InvoiceNo | StockCode | Description | Quantity | \ |
|---|---|---|---|---|---|
| 141 | C536379 | D | Discount | -1 | |
| 154 | C536383 | 35004C | SET OF 3 COLOURED FLYING DUCKS | -1 | |
| 235 | C536391 | 22556 | PLASTERS IN TIN CIRCUS PARADE | -12 | |
| 236 | C536391 | 21984 | PACK OF 12 PINK PAISLEY TISSUES | -24 | |
| 237 | C536391 | 21983 | PACK OF 12 BLUE PAISLEY TISSUES | -24 | |
| … | … | … | … | … | |
| 540449 | C581490 | 23144 | ZINC T-LIGHT HOLDER STARS SMALL | -11 | |
| 541541 | C581499 | M | Manual | -1 | |
| 541715 | C581568 | 21258 | VICTORIAN SEWING BOX LARGE | -5 | |
| 541716 | C581569 | 84978 | HANGING HEART JAR T-LIGHT HOLDER | -1 | |
| 541717 | C581569 | 20979 | 36 PENCILS TUBE RED RETROSPOT | -5 | |

| | InvoiceDate | UnitPrice | CustomerID | Country | revenue | \ |
|---|---|---|---|---|---|---|
| 141 | 2010-12-01 09:41:00 | 27.50 | 14527.0 | United Kingdom | -27.50 | |
| 154 | 2010-12-01 09:49:00 | 4.65 | 15311.0 | United Kingdom | -4.65 | |
| 235 | 2010-12-01 10:24:00 | 1.65 | 17548.0 | United Kingdom | -19.80 | |
| 236 | 2010-12-01 10:24:00 | 0.29 | 17548.0 | United Kingdom | -6.96 | |
| 237 | 2010-12-01 10:24:00 | 0.29 | 17548.0 | United Kingdom | -6.96 | |
| … | … | … | … | … | … | |
| 540449 | 2011-12-09 09:57:00 | 0.83 | 14397.0 | United Kingdom | -9.13 | |
| 541541 | 2011-12-09 10:28:00 | 224.69 | 15498.0 | United Kingdom | -224.69 | |
| 541715 | 2011-12-09 11:57:00 | 10.95 | 15311.0 | United Kingdom | -54.75 | |
| 541716 | 2011-12-09 11:58:00 | 1.25 | 17315.0 | United Kingdom | -1.25 | |
| 541717 | 2011-12-09 11:58:00 | 1.25 | 17315.0 | United Kingdom | -6.25 | |

| | Invoice_month | Invoice_year | InvoiceTime | InvoiceHour |
|---|---|---|---|---|
| 141 | 12 | 2010 | 09:41:00 | 9 |
| 154 | 12 | 2010 | 09:49:00 | 9 |
| 235 | 12 | 2010 | 10:24:00 | 10 |
| 236 | 12 | 2010 | 10:24:00 | 10 |
| 237 | 12 | 2010 | 10:24:00 | 10 |
| … | … | … | … | … |
| 540449 | 12 | 2011 | 09:57:00 | 9 |
| 541541 | 12 | 2011 | 10:28:00 | 10 |
| 541715 | 12 | 2011 | 11:57:00 | 11 |
| 541716 | 12 | 2011 | 11:58:00 | 11 |
| 541717 | 12 | 2011 | 11:58:00 | 11 |

[10624 rows x 13 columns]

[ ]:
```
''' most return orders hourly basis '''

df_ret['InvoiceHour'] = df['InvoiceHour']
```

```
<ipython-input-79-f805852d187c>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_ret['InvoiceHour'] = df['InvoiceHour']
```

[ ]:
```
df_ret
```

[ ]:
```
       InvoiceNo StockCode                      Description  Quantity  \
141      C536379         D                         Discount        -1
154      C536383    35004C   SET OF 3 COLOURED  FLYING DUCKS        -1
235      C536391     22556    PLASTERS IN TIN CIRCUS PARADE        -12
236      C536391     21984  PACK OF 12 PINK PAISLEY TISSUES        -24
237      C536391     21983  PACK OF 12 BLUE PAISLEY TISSUES        -24
…          …       …                                  …       …
540449   C581490     23144   ZINC T-LIGHT HOLDER STARS SMALL       -11
541541   C581499         M                           Manual        -1
541715   C581568     21258       VICTORIAN SEWING BOX LARGE        -5
541716   C581569     84978  HANGING HEART JAR T-LIGHT HOLDER        -1
541717   C581569     20979    36 PENCILS TUBE RED RETROSPOT        -5

                InvoiceDate  UnitPrice CustomerID         Country  revenue  \
141    2010-12-01 09:41:00      27.50    14527.0  United Kingdom   -27.50
154    2010-12-01 09:49:00       4.65    15311.0  United Kingdom    -4.65
235    2010-12-01 10:24:00       1.65    17548.0  United Kingdom   -19.80
236    2010-12-01 10:24:00       0.29    17548.0  United Kingdom    -6.96
237    2010-12-01 10:24:00       0.29    17548.0  United Kingdom    -6.96
…                      …        …         …              …      …
540449 2011-12-09 09:57:00       0.83    14397.0  United Kingdom    -9.13
541541 2011-12-09 10:28:00     224.69    15498.0  United Kingdom  -224.69
541715 2011-12-09 11:57:00      10.95    15311.0  United Kingdom   -54.75
541716 2011-12-09 11:58:00       1.25    17315.0  United Kingdom    -1.25
541717 2011-12-09 11:58:00       1.25    17315.0  United Kingdom    -6.25

        Invoice_month  Invoice_year InvoiceTime  InvoiceHour
141                12          2010    09:41:00            9
154                12          2010    09:49:00            9
235                12          2010    10:24:00           10
236                12          2010    10:24:00           10
237                12          2010    10:24:00           10
…                 …           …          …           …
```

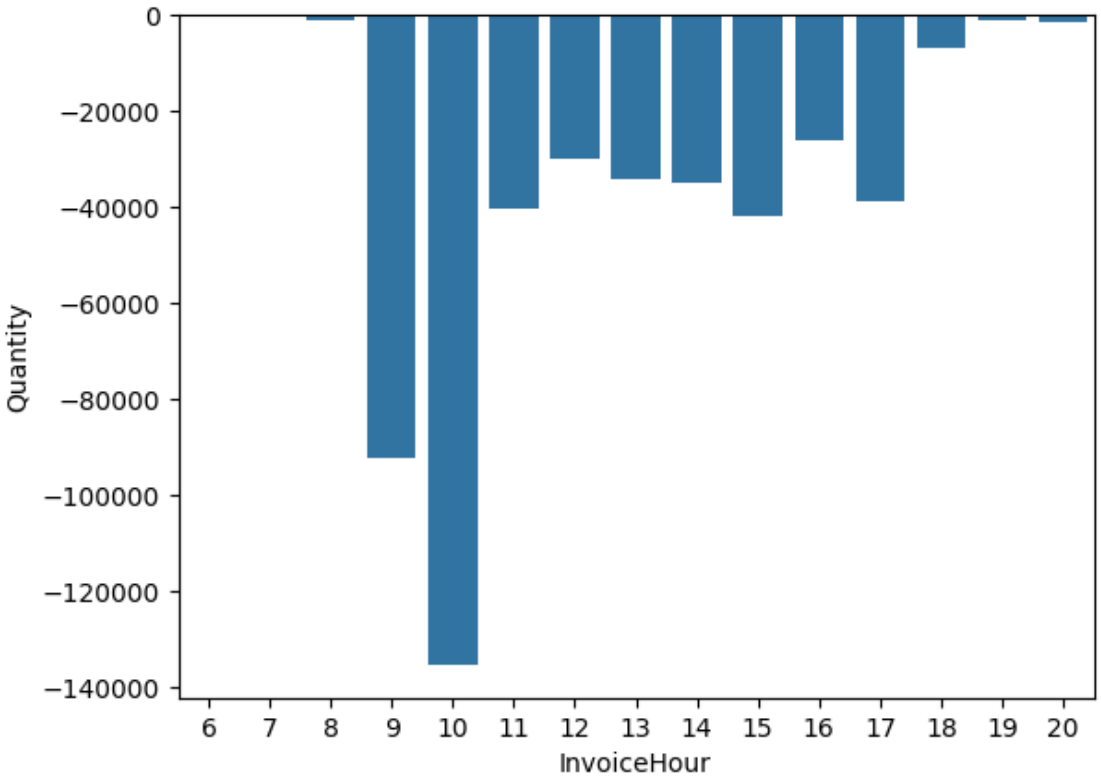
```
540449              12          2011    09:57:00           9
541541              12          2011    10:28:00          10
541715              12          2011    11:57:00          11
541716              12          2011    11:58:00          11
541717              12          2011    11:58:00          11

[10624 rows x 13 columns]
```

[ ]:
```
df_ret_h = df_ret.groupby('InvoiceHour')['Quantity'].sum().reset_index()
df_ret_h
```

[ ]:
```
    InvoiceHour  Quantity
0             6       -87
1             7        -9
2             8      -974
3             9    -92427
4            10   -135486
5            11    -40241
6            12    -29990
7            13    -34165
8            14    -35064
9            15    -41771
10           16    -26310
11           17    -38918
12           18     -6692
13           19     -1078
14           20     -1319
```

[ ]:
```
a = sns.barplot(data = df_ret_h,x = 'InvoiceHour', y = 'Quantity')
a
```

[ ]:
```
<Axes: xlabel='InvoiceHour', ylabel='Quantity'>
```




In the above plot we can see the orders that were cancelled or returned on hourly basis

We can observe from the graph that the time range between 9 AM - 11 AM is where the most orders are placed for cancellation or returns.

[ ]:
```
''' Analysis of data on Quarterly basis'''


bin_q = [1,4,7,10,13]
label_q = ['Q1','Q2','Q3','Q4']

df['Quarter'] = pd.cut(df['Invoice_month'],bins = bin_q,labels = label_q)
```

[ ]:
```
df[df['Quarter'] == 'Q2']
```

[ ]:

| | InvoiceNo | StockCode | Description | Quantity | \ |
|---|---|---|---|---|---|
| 171999 | 551515 | 21731 | RED TOADSTOOL LED NIGHT LIGHT | 12 | |
| 172000 | 551515 | 20749 | ASSORTED COLOUR MINI CASES | 12 | |
| 172001 | 551515 | 22729 | ALARM CLOCK BAKELIKE ORANGE | 8 | |
| 172002 | 551515 | 22998 | TRAVEL CARD WALLET KEEP CALM | 24 | |
| 172003 | 551515 | 22665 | RECIPE BOX BLUE SKETCHBOOK DESIGN | 24 | |
| … | … | … | … | … | |

```
285416    561903     21900                   KEY FOB , SHED        24
285417    561903     48187                DOORMAT NEW ENGLAND       2
285418    561903     85152   HAND OVER THE CHOCOLATE   SIGN        12
285419    561903     82600              NO SINGING METAL SIGN        12
285420    561903     21175       GIN + TONIC DIET METAL SIGN        12

                InvoiceDate  UnitPrice CustomerID         Country  revenue  \
171999 2011-05-01 10:51:00       1.65    15606.0  United Kingdom    19.80
172000 2011-05-01 10:51:00       6.35    15606.0  United Kingdom    76.20
172001 2011-05-01 10:51:00       3.75    15606.0  United Kingdom    30.00
172002 2011-05-01 10:51:00       0.42    15606.0  United Kingdom    10.08
172003 2011-05-01 10:51:00       2.55    15606.0  United Kingdom    61.20
…                     …        …        …              …      …
285416 2011-07-31 16:04:00       0.65    17162.0  United Kingdom    15.60
285417 2011-07-31 16:04:00       7.95    17162.0  United Kingdom    15.90
285418 2011-07-31 16:04:00       2.10    17162.0  United Kingdom    25.20
285419 2011-07-31 16:04:00       2.10    17162.0  United Kingdom    25.20
285420 2011-07-31 16:04:00       2.55    17162.0  United Kingdom    30.60

        Invoice_month  Invoice_year InvoiceTime  InvoiceHour Quarter
171999              5          2011    10:51:00           10      Q2
172000              5          2011    10:51:00           10      Q2
172001              5          2011    10:51:00           10      Q2
172002              5          2011    10:51:00           10      Q2
172003              5          2011    10:51:00           10      Q2
…               …           …          …           …      …
285416              7          2011    16:04:00           16      Q2
285417              7          2011    16:04:00           16      Q2
285418              7          2011    16:04:00           16      Q2
285419              7          2011    16:04:00           16      Q2
285420              7          2011    16:04:00           16      Q2

[113422 rows x 14 columns]
```

[ ]:
```
''' Quarter wise revenue '''

df.groupby('Quarter')['revenue'].sum().reset_index().sort_values(by =␣
 ↪'revenue',ascending = False)
```

[ ]:
```
  Quarter      revenue
2      Q3  2773072.802
3      Q4  2644381.280
1      Q2  2095756.741
0      Q1  1674536.851
```

Here the table above shows the Quarterly revenue generated on our platform. Q1 – 'January,February,March' Q2 – 'April,May,June' Q3 – 'July,August,September' Q4 – 'Octo-

ber,November,December'

We can see that Q3 has made more revenue generation or we can say more sales followed by Q4.

Q3 and Q4 are most profitable period for our paltform sales.

Q2 is average performed period in case of sales.

Care must be taken to improve sales and generate more revenu for Q1.

[ ]:
```
''' quarter wise customer visits '''

df.groupby('Quarter')['CustomerID'].count().reset_index().sort_values(by =␣
 ↪'CustomerID',ascending = False)
```

[ ]:

| | Quarter | CustomerID |
|---|---|---|
| 3 | Q4 | 152717 |
| 2 | Q3 | 146252 |
| 1 | Q2 | 113422 |
| 0 | Q1 | 94371 |

The above table shows details about customer visits on basis of Quarterly periods.

here in Q4, the customer visits were more then followed by Q3 and Q2.

Q2,Q4,Q3 are almost in a gardual increase in customer visists from Q2 - Q4 but the Q1 is much lower for customers visits which directly implies on less sales as well.

Although Q4 was leading in more customers but in case of Q3 sales was more prominent as we see in our above revenue table.

hence in Q3 there were more conversions than compared to Q4 where visits were more but conversions is less than of Q3.

[ ]:
```
df['InvoiceDay'] = df['InvoiceDate'].dt.day_name()
```

[ ]:
```
df
```

[ ]:

| | InvoiceNo | StockCode | Description | Quantity | \ |
|---|---|---|---|---|---|
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | |
| … | … | … | … | … | |
| 541904 | 581587 | 22613 | PACK OF 20 SPACEBOY NAPKINS | 12 | |
| 541905 | 581587 | 22899 | CHILDREN'S APRON DOLLY GIRL | 6 | |
| 541906 | 581587 | 23254 | CHILDRENS CUTLERY DOLLY GIRL | 4 | |
| 541907 | 581587 | 23255 | CHILDRENS CUTLERY CIRCUS PARADE | 4 | |
| 541908 | 581587 | 22138 | BAKING SET 9 PIECE RETROSPOT | 3 | |

| | InvoiceDate | UnitPrice | CustomerID | Country | revenue | \ |
|---|---|---|---|---|---|---|

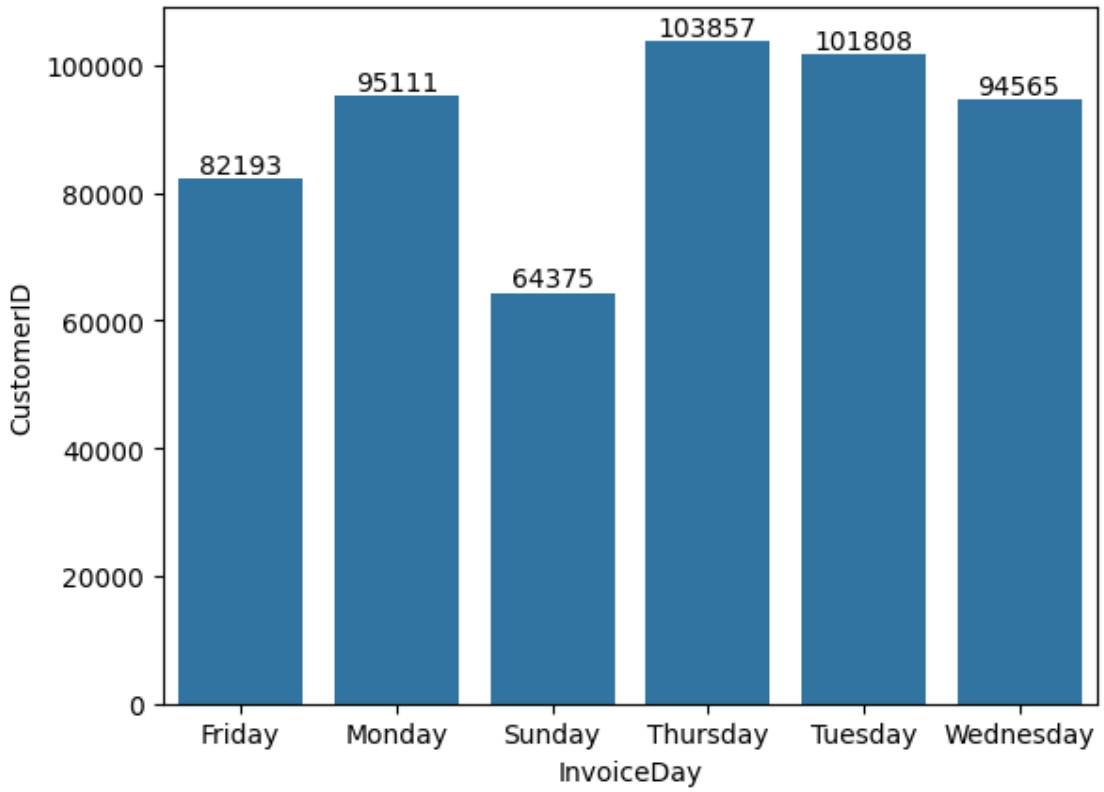
```
0      2010-12-01 08:26:00       2.55    17850.0  United Kingdom    15.30
1      2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
2      2010-12-01 08:26:00       2.75    17850.0  United Kingdom    22.00
3      2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
4      2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
…                      …          …         …               …      …
541904 2011-12-09 12:50:00       0.85    12680.0          France    10.20
541905 2011-12-09 12:50:00       2.10    12680.0          France    12.60
541906 2011-12-09 12:50:00       4.15    12680.0          France    16.60
541907 2011-12-09 12:50:00       4.15    12680.0          France    16.60
541908 2011-12-09 12:50:00       4.95    12680.0          France    14.85

        Invoice_month  Invoice_year InvoiceTime  InvoiceHour Quarter  \
0                  12          2010    08:26:00            8      Q4
1                  12          2010    08:26:00            8      Q4
2                  12          2010    08:26:00            8      Q4
3                  12          2010    08:26:00            8      Q4
4                  12          2010    08:26:00            8      Q4
…                 …            …          …           …      …
541904             12          2011    12:50:00           12      Q4
541905             12          2011    12:50:00           12      Q4
541906             12          2011    12:50:00           12      Q4
541907             12          2011    12:50:00           12      Q4
541908             12          2011    12:50:00           12      Q4

       InvoiceDay
0       Wednesday
1       Wednesday
2       Wednesday
3       Wednesday
4       Wednesday
…           …
541904     Friday
541905     Friday
541906     Friday
541907     Friday
541908     Friday

[541909 rows x 15 columns]
```

[ ]:
```
''' Best day in week to shop for customers '''

df_day_cus = df.groupby('InvoiceDay')['CustomerID'].count().reset_index()
df_day_cus
```

[ ]:
```
  InvoiceDay  CustomerID
0     Friday       82193
```

```
1      Monday       95111
2      Sunday       64375
3    Thursday      103857
4     Tuesday      101808
5   Wednesday       94565
```

[ ]:
```
a = sns.barplot(data = df_day_cus,x = 'InvoiceDay',y = 'CustomerID')

for i in a .containers:
  a.bar_label(i,)
```




The above graph representation shows the best day to shop for customers

Here we can see that Tuesday and Thursda are the days where most of shopping has been done.

While Friday and Sunday are least days preferable for shopping by customers.

From Monday to Thursday of 4 days are preferable by customer for more visits or shopping

where as from friday to Sunday the graph has been reduced which implies customers are more intreseted in shopping on weekdays.

New initiations or way must be found to make more transactions on weekends also.

[ ]:
```
'Average days for customer shopping '

a = df.groupby('CustomerID').apply(lambda df : (df.InvoiceDate - df.InvoiceDate.
 ↪shift(1)).dt.days).reset_index()

a1 = round(a.groupby('CustomerID')['InvoiceDate'].mean().reset_index().
 ↪sort_values(by = 'InvoiceDate'))

a1.fillna(0,inplace = True)
a1
```

[ ]:
```
     CustomerID  InvoiceDate
0       12346.0          0.0
2952    16343.0          0.0
2953    16344.0          0.0
2954    16345.0          0.0
1225    13992.0          0.0
…         …            …
4241    18113.0          0.0
4261    18141.0          0.0
4287    18174.0          0.0
4295    18184.0          0.0
4334    18233.0          0.0

[4373 rows x 2 columns]
```

[ ]:
```
a1['InvoiceDate'].value_counts().reset_index()
```

[ ]:
```
    InvoiceDate  count
0           0.0   1703
1           1.0    782
2           2.0    601
3           3.0    353
4           4.0    207
..          …     …
62         53.0      1
63         43.0      1
64         42.0      1
65         35.0      1
66        309.0      1

[67 rows x 2 columns]
```

[ ]:
```
''' Customers who do shopping on average after 30 days '''

a1[a1['InvoiceDate'] > 30]
```

[ ]:

| | CustomerID | InvoiceDate |
|---|---|---|
| 3497 | 17080.0 | 31.0 |
| 2445 | 15649.0 | 31.0 |
| 3139 | 16596.0 | 32.0 |
| 194 | 12586.0 | 32.0 |
| 3157 | 16620.0 | 32.0 |
| 435 | 12897.0 | 32.0 |
| 3074 | 16500.0 | 32.0 |
| 4365 | 18277.0 | 32.0 |
| 1972 | 15030.0 | 33.0 |
| 2111 | 15206.0 | 33.0 |
| 4223 | 18087.0 | 34.0 |
| 3670 | 17339.0 | 34.0 |
| 2340 | 15512.0 | 34.0 |
| 1571 | 14473.0 | 35.0 |
| 527 | 13029.0 | 36.0 |
| 2735 | 16048.0 | 36.0 |
| 1678 | 14616.0 | 36.0 |
| 2398 | 15587.0 | 38.0 |
| 3951 | 17707.0 | 38.0 |
| 3469 | 17044.0 | 39.0 |
| 3454 | 17025.0 | 39.0 |
| 4029 | 17820.0 | 42.0 |
| 3009 | 16414.0 | 43.0 |
| 2030 | 15101.0 | 44.0 |
| 3340 | 16861.0 | 44.0 |
| 1544 | 14437.0 | 44.0 |
| 893 | 13525.0 | 47.0 |
| 1338 | 14147.0 | 47.0 |
| 1445 | 14295.0 | 49.0 |
| 3386 | 16927.0 | 49.0 |
| 1262 | 14045.0 | 53.0 |
| 432 | 12891.0 | 54.0 |
| 379 | 12823.0 | 55.0 |
| 612 | 13145.0 | 55.0 |
| 3453 | 17024.0 | 56.0 |
| 1634 | 14548.0 | 56.0 |
| 4110 | 17929.0 | 59.0 |
| 3455 | 17026.0 | 61.0 |
| 3033 | 16446.0 | 68.0 |
| 1118 | 13848.0 | 68.0 |
| 4343 | 18246.0 | 70.0 |
| 3046 | 16462.0 | 73.0 |
| 1677 | 14609.0 | 74.0 |
| 2926 | 16308.0 | 74.0 |
| 440 | 12908.0 | 84.0 |
| 1610 | 14520.0 | 96.0 |

```
3879    17616.0         96.0
3039    16454.0         98.0
3096    16532.0        107.0
4175    18017.0        114.0
1945    14987.0        116.0
1860    14865.0        121.0
3457    17029.0        126.0
4362    18273.0        128.0
1799    14777.0        176.0
419     12875.0        219.0
4217    18080.0        223.0
4220    18084.0        284.0
555     13068.0        309.0
```

[ ]:
```
''' Filtering the data with orders that are purchased and has no returns '''

df_p = df[df['Quantity'] > 0]
df_p
```

[ ]:
```
       InvoiceNo StockCode                          Description  Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1         536365     71053                  WHITE METAL LANTERN         6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
…            …       …                                     …       …
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS        12
541905    581587     22899         CHILDREN'S APRON DOLLY GIRL          6
541906    581587     23254        CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587     22138        BAKING SET 9 PIECE RETROSPOT          3

                InvoiceDate  UnitPrice CustomerID         Country  revenue  \
0       2010-12-01 08:26:00       2.55    17850.0  United Kingdom    15.30
1       2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
2       2010-12-01 08:26:00       2.75    17850.0  United Kingdom    22.00
3       2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
4       2010-12-01 08:26:00       3.39    17850.0  United Kingdom    20.34
…                       …          …          …               …       …
541904 2011-12-09 12:50:00       0.85    12680.0          France    10.20
541905 2011-12-09 12:50:00       2.10    12680.0          France    12.60
541906 2011-12-09 12:50:00       4.15    12680.0          France    16.60
541907 2011-12-09 12:50:00       4.15    12680.0          France    16.60
541908 2011-12-09 12:50:00       4.95    12680.0          France    14.85

        Invoice_month  Invoice_year InvoiceTime  InvoiceHour Quarter  \
0                  12          2010    08:26:00            8      Q4
```

```
1                  12          2010    08:26:00            8       Q4
2                  12          2010    08:26:00            8       Q4
3                  12          2010    08:26:00            8       Q4
4                  12          2010    08:26:00            8       Q4
…                 …           …         …          …      …
541904             12          2011    12:50:00           12       Q4
541905             12          2011    12:50:00           12       Q4
541906             12          2011    12:50:00           12       Q4
541907             12          2011    12:50:00           12       Q4
541908             12          2011    12:50:00           12       Q4

       InvoiceDay
0       Wednesday
1       Wednesday
2       Wednesday
3       Wednesday
4       Wednesday
…           …
541904     Friday
541905     Friday
541906     Friday
541907     Friday
541908     Friday

[531285 rows x 15 columns]
```

[ ]:
```
''' getting the total revenue for each customer on latest shopping date '''

df_g = df_p.groupby(['CustomerID','InvoiceNo']).agg({'revenue' :␣
 ↪sum,'InvoiceDate' : max})
```

[ ]:
```
df_g
```

[ ]:
```
                        revenue         InvoiceDate
CustomerID InvoiceNo
12346.0    541431     77183.60 2011-01-18 10:01:00
12347.0    537626       711.79 2010-12-07 14:57:00
           542237       475.39 2011-01-26 14:30:00
           549222       636.25 2011-04-07 10:43:00
           556201       382.52 2011-06-09 13:01:00
…                         …                 …
Unknown    564087      2653.95 2011-08-23 09:38:00
           567353      2653.95 2011-09-19 16:14:00
           571652       423.42 2011-10-18 12:17:00
           573154       311.10 2011-10-28 08:20:00
           576234       326.10 2011-11-14 13:27:00
```

```
[20728 rows x 2 columns]
```

[ ]:
```
''' Creating the functions for required calculations of average,count,Customer␣
 ↪date range,purchase frequency '''

def meang(x):
  return x.mean()

def countg(x):
  return x.count()

def pur_dur(x):
  return(x.max() - x.min()).days

def avg_dur(x):
  return (x.max() - x.min()).days/x.count()


meang.__name__ = 'avg'
countg.__name__ = 'count'
pur_dur.__name__ = 'purchase_duration'
avg_dur.__name__ = 'purchase_frequency'
```

[ ]:
```
''' Aggregating the revenue and date columns for required analysis '''

dfg1 = df_g.reset_index().groupby('CustomerID').agg({
    'revenue' : [min,max,sum,meang,countg],
    'InvoiceDate' : [min,max,pur_dur,avg_dur]}
)
```

[ ]:
```
dfg1
```

[ ]:
```
             revenue                                                \
                 min       max       sum           avg count
CustomerID
12346.0     77183.60  77183.60  77183.60  77183.600000     1
12347.0       224.82   1294.32   4310.00    615.714286     7
12348.0       227.44    892.80   1797.24    449.310000     4
12349.0      1757.55   1757.55   1757.55   1757.550000     1
12350.0       334.40    334.40    334.40    334.400000     1
…                …         …         …            …    …
18281.0        80.82     80.82     80.82     80.820000     1
18282.0        77.84    100.21    178.05     89.025000     2
18283.0         1.95    313.65   2094.88    130.930000    16
18287.0        70.68   1001.32   1837.28    612.426667     3
Unknown       160.00   2653.95  15691.80   1426.527273    11
```

```
                    InvoiceDate                                          \
                            min                 max purchase_duration
CustomerID
12346.0     2011-01-18 10:01:00 2011-01-18 10:01:00                 0
12347.0     2010-12-07 14:57:00 2011-12-07 15:52:00               365
12348.0     2010-12-16 19:09:00 2011-09-25 13:13:00               282
12349.0     2011-11-21 09:51:00 2011-11-21 09:51:00                 0
12350.0     2011-02-02 16:01:00 2011-02-02 16:01:00                 0
…                           …                   …               …
18281.0     2011-06-12 10:53:00 2011-06-12 10:53:00                 0
18282.0     2011-08-05 13:35:00 2011-12-02 11:43:00               118
18283.0     2011-01-06 14:14:00 2011-12-06 12:02:00               333
18287.0     2011-05-22 10:39:00 2011-10-28 09:29:00               158
Unknown     2011-01-24 14:24:00 2011-11-14 13:27:00               293


            purchase_frequency
CustomerID
12346.0               0.000000
12347.0              52.142857
12348.0              70.500000
12349.0               0.000000
12350.0               0.000000
…                          …
18281.0               0.000000
18282.0              59.000000
18283.0              20.812500
18287.0              52.666667
Unknown              26.636364

[4340 rows x 9 columns]
```

[ ]:
```
''' Clearing the column banner with more clariifed column names '''

dfg1.columns = ['_'.join(col).lower() for col in dfg1.columns]
dfg1 = dfg1.reset_index()
dfg1
```

[ ]:
```
      CustomerID  revenue_min  revenue_max  revenue_sum   revenue_avg  \
0        12346.0     77183.60     77183.60     77183.60  77183.600000
1        12347.0       224.82      1294.32      4310.00    615.714286
2        12348.0       227.44       892.80      1797.24    449.310000
3        12349.0      1757.55      1757.55      1757.55   1757.550000
4        12350.0       334.40       334.40       334.40    334.400000
…          …            …            …            …            …
4335     18281.0        80.82        80.82        80.82     80.820000
4336     18282.0        77.84       100.21       178.05     89.025000
```

```
4337    18283.0         1.95       313.65      2094.88    130.930000
4338    18287.0        70.68      1001.32      1837.28    612.426667
4339    Unknown       160.00      2653.95     15691.80   1426.527273

      revenue_count     invoicedate_min     invoicedate_max  \
0                 1 2011-01-18 10:01:00 2011-01-18 10:01:00
1                 7 2010-12-07 14:57:00 2011-12-07 15:52:00
2                 4 2010-12-16 19:09:00 2011-09-25 13:13:00
3                 1 2011-11-21 09:51:00 2011-11-21 09:51:00
4                 1 2011-02-02 16:01:00 2011-02-02 16:01:00
…               …                   …                   …
4335              1 2011-06-12 10:53:00 2011-06-12 10:53:00
4336              2 2011-08-05 13:35:00 2011-12-02 11:43:00
4337             16 2011-01-06 14:14:00 2011-12-06 12:02:00
4338              3 2011-05-22 10:39:00 2011-10-28 09:29:00
4339             11 2011-01-24 14:24:00 2011-11-14 13:27:00

      invoicedate_purchase_duration  invoicedate_purchase_frequency
0                                 0                        0.000000
1                               365                       52.142857
2                               282                       70.500000
3                                 0                        0.000000
4                                 0                        0.000000
…                               …                          …
4335                              0                        0.000000
4336                            118                       59.000000
4337                            333                       20.812500
4338                            158                       52.666667
4339                            293                       26.636364

[4340 rows x 10 columns]
```

Exaplanation This dataset gives us an idea of the purchases each customer has made.

Let's have a look at CustomerID 12346 (first row). This customer made only one purchase on January 18,2011.

The second customer (12347) has made six purchases within December 7, 2010 and October 31, 2011. The timespan here is about 365 days. The average amount this customer spent on each order is 615. We also see from the record, that this customer made a purchase every 52.5 days.

[ ]:
```
'''
This plot shows the distributions of the number of purchases that the repeat␣
 ↪customers have made '''


a = dfg1['invoicedate_purchase_frequency'].hist(bins = 10,rwidth = 0.8,color =␣
 ↪'skyblue',figsize = (10,6))
```

```
a.set_xlabel = ('average number of days between purchase')
a.set_ylable = ('count')
plt.title('Number of days between purchases for repeated customers')

plt.show()
```


The above plot shows the Average days for a customer to return for shopping

here most of our data is concentrated between 0-20 which is a good sign but also some customers return for shopping more than 30 days

returning of customers are very prominent in our business as they will tell about our business model like rating or likelihood of shopping on our platform.

There are many ways for retruning customers to and making conversion from them.

Increasing the loyalty and improving the shopping experience for returning customers will improve business as these are ones who spread the their experience by means of ways to attract more customers to our business

[ ]:

Here in the above dataset gives the information about the Customers, orders purchased and returned,date and time of transactions,country,products and their details.

We have done the CRM analysis on given dataset and made some insights from the data. On doing the analysis there are some recommendations from my side as follows

#Recommendations:

1. The problem here arises with null values where their impact is more when the number is more for null vlues on data.
2. We can observe that country United Kingdom has been top in sales and revenue production but also we can see that the number of cancelled orders are also more in United Kingdom which shows us that cutomers may be palced wrongly order or may be not satisfied with roduct.
3. If customer placed by mistake there are some steps to follow if this actions repeats in frequent. Otherwise if customer id not satisfied with product thats where the challenge and downfall will increase in sales.
4. As ecommerce business is full of competition we should not let customer to slip away due to product related issues.
5. We must take care of products information that provided on website and also make sure what is customer requirment and product matching.
6. Customer loyalty and campaigns programs must be done to increase customer retention and also belief on our business.
7. Besides,We can see that months that attract more customers and sales are most in Quarter 3 and Quarter 4 which indicates us that most months in Q1 and Q2 are low in sales even in customers visits.
8. Customer segmentation and customer journey analysis has to be done in these months form January to August for the insights that missing in attracting more customers and increasing sales.
9. More marketing and sale days must be introduced in months where customer visits are less and make conversions more effective in these months as it is time where other competitiors move up by taking chance.
10. The preferred days for shopping has been from Monday to Thursday. Customers are preferred to shop mostly on weekdays, but there should be plan to be take forward for increasing the customer visits in weekends where most of the people are free and ads like email marketing,notifcations and other ways for contacting customers and introducing them for weekend special offers or delivery free campaign when orderd on weekends leads to more conversions on weekends which makes the business to run on all 7 days with maximum conversions.
11. the preferred time to shop is also from 10 AM to evening 6-7 PM which is good metric but it should also note that making the customer visits in the evening or night will increase the customer experinece ,as of payment gateways and other internet traffic is less that can make smooth experince for customers to shop. Making segementation of customers psychographically wil help us in solving this case as most of people are active on internet after 7 PM. Making use of this time will increase sales. MAking products available to customers that are dleivered within hours after ordering also oneway for more conversions.
12. We can see that average days for customer return is between 0 - 20 days in our data which is good metric to say that cutomers are satisfied with our website and products but turning them loyal is main benefit for business as they act as medium for attracting new customers

by sharing their reviews,comments and word of mouth which leads to increase in conversion rates.

13. These are some of recommendations from my side.

THANK YOU

[ ]:

.