In [2]:
```python
import pandas as pd
import numpy as np
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from yellowbrick.cluster import KElbowVisualizer
from numpy import math
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

1. Loading & Reading the Dataset

In [3]:
```python
df = pd.read_csv('C:\\Users\\PC\\Desktop\\DA PROJECT\\RFMT ANALYSIS\\online_retail_II_da

print('-'*50)
print('Data imported successfully!!!')
df.head(5).style.set_properties(**{"background-color": "#cd5c5c","color": "black", "bord
```

```
--------------------------------------------------
Data imported successfully!!!
```

Out[3]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.950000	13085.000000	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.750000	13085.000000	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.750000	13085.000000	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.100000	13085.000000	United Kingdom
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.250000	13085.000000	United Kingdom

2. Exploring information of the dataset

In [4]:
```python
pd.set_option('display.max_columns', None)
def data_overview(df, head=5):
    print('SHAPE OF DATASET'.center(125,'-'))
    print('Rows:{}'.format(df.shape[0]))
    print('Columns:{}'.format(df.shape[1]))
    print('HEAD OF DATASET'.center(125,'-'))
    print(df.head(head))
    print('DATA TYPE'.center(125, '-'))
    print(df.dtypes)
    print('MISSING VALUES'.center(125,'-'))
    print(df.isnull().sum()[df.isnull().sum()>0].sort_values(ascending = False))
    print('DUPLICATED VALUES'.center(125,'-'))
    print(df.duplicated().sum())
    print('STATISTICS OF DATA'.center(125,'-'))
    print(df.describe(include="all"))
    print("DATA INFO".center(125,'-'))
```

```
    print(df.info())
data_overview(df)
```

```
-------------------------------------------------------SHAPE OF DATASET-----------------
-------------------------------------
Rows:1067371
Columns:8
-------------------------------------------------------HEAD OF DATASET------------------
-------------------------------------
  Invoice StockCode                          Description  Quantity  \
0  489434     85048  15CM CHRISTMAS GLASS BALL 20 LIGHTS        12
1  489434    79323P                   PINK CHERRY LIGHTS        12
2  489434    79323W                  WHITE CHERRY LIGHTS        12
3  489434     22041         RECORD FRAME 7" SINGLE SIZE         48
4  489434     21232       STRAWBERRY CERAMIC TRINKET BOX        24

          InvoiceDate  Price  Customer ID         Country
0  2009-12-01 07:45:00   6.95      13085.0  United Kingdom
1  2009-12-01 07:45:00   6.75      13085.0  United Kingdom
2  2009-12-01 07:45:00   6.75      13085.0  United Kingdom
3  2009-12-01 07:45:00   2.10      13085.0  United Kingdom
4  2009-12-01 07:45:00   1.25      13085.0  United Kingdom
----------------------------------------------------------DATA TYPE---------------------
-------------------------------------
Invoice         object
StockCode       object
Description     object
Quantity         int64
InvoiceDate     object
Price          float64
Customer ID    float64
Country         object
dtype: object
--------------------------------------------------------MISSING VALUES------------------
-------------------------------------
Customer ID    243007
Description      4382
dtype: int64
------------------------------------------------------DUPLICATED VALUES-----------------
-------------------------------------
34335
------------------------------------------------------STATISTICS OF DATA----------------
-------------------------------------
        Invoice StockCode                         Description      Quantity  \
count   1067371   1067371                             1062989  1.067371e+06
unique    53628      5305                                5698           NaN
top      537434    85123A  WHITE HANGING HEART T-LIGHT HOLDER           NaN
freq       1350      5829                                5918           NaN
mean        NaN       NaN                                 NaN  9.938898e+00
std         NaN       NaN                                 NaN  1.727058e+02
min         NaN       NaN                                 NaN -8.099500e+04
25%         NaN       NaN                                 NaN  1.000000e+00
50%         NaN       NaN                                 NaN  3.000000e+00
75%         NaN       NaN                                 NaN  1.000000e+01
max         NaN       NaN                                 NaN  8.099500e+04

                InvoiceDate         Price    Customer ID         Country
count               1067371  1.067371e+06  824364.000000         1067371
unique                47635           NaN            NaN              43
top     2010-12-06 16:57:00           NaN            NaN  United Kingdom
freq                   1350           NaN            NaN          981330
mean                    NaN  4.649388e+00   15324.638504             NaN
std                     NaN  1.235531e+02    1697.464450             NaN
min                     NaN -5.359436e+04   12346.000000             NaN
25%                     NaN  1.250000e+00   13975.000000             NaN
50%                     NaN  2.100000e+00   15255.000000             NaN
```

```
75%                        NaN  4.150000e+00   16797.000000             NaN
max                        NaN  3.897000e+04   18287.000000             NaN
-----------------------------------------------------------DATA INFO---------------------
-------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1067371 entries, 0 to 1067370
Data columns (total 8 columns):
 #   Column       Non-Null Count    Dtype  
---  ------       --------------    -----  
 0   Invoice      1067371 non-null  object 
 1   StockCode    1067371 non-null  object 
 2   Description  1062989 non-null  object 
 3   Quantity     1067371 non-null  int64  
 4   InvoiceDate  1067371 non-null  object 
 5   Price        1067371 non-null  float64
 6   Customer ID  824364 non-null   float64
 7   Country      1067371 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 65.1+ MB
None
```

NOTES:

- The Dataset has Rows: 1067371 and Columns:8
- The Dataset has 3 types of columns: strings(5), integer(1), float(2)
- The Dataset has Missing values in Customer ID (243007) and Description (4382)
- Invoice starts with the 'c' needs to be cleaned as it is cancelled transaction
- The Dataset has duplicates
- Aslo check for negative value and outliers in Quantity and Price

3. Data Wrangling

In [5]:
```
df = df.rename(columns = {
    'Customer ID' : 'CustomerID'
})
```

In [6]:
```
# 3.1 Checking the data types:
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
df.dtypes
```

Out[6]:
```
Invoice                object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
Price                 float64
CustomerID            float64
Country                object
dtype: object
```

In [7]:
```
# 3.2 Dealing with missing values
print("Shape of data before removing NaN's CustomerID",df.shape)
df.dropna(subset="CustomerID",axis=0,inplace=True)
print("Shape of data after removing NaN's CustomerID",df.shape)
```

```
Shape of data before removing NaN's CustomerID (1067371, 8)
Shape of data after removing NaN's CustomerID (824364, 8)
```

In [8]:
```
print("Missing values in each column after cleaning customerID :\n", df.isnull().sum())
```

```
Missing values in each column after cleaning customerID :
```

```
 Invoice        0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
Price          0
CustomerID     0
Country        0
dtype: int64
```

In [9]:
```
# 3.3 Removing canceled products from invoice
df=df[~df['Invoice'].str.contains('C', na=False)]
df.shape
print("Dataset is free from cancelled products information")
```

```
Dataset is free from cancelled products information
```

In [10]:
```
# 3.4 Removing the duplicates
print('Number of duplicates before cleaning:', df.duplicated().sum())
df=df.drop_duplicates(keep='first')
print('Number of duplicates before cleaning:', df.duplicated().sum())
```

```
Number of duplicates before cleaning: 26125
Number of duplicates before cleaning: 0
```

In [11]:
```
# 3.5 Checking for negative values
print('Negative values in Quantity is:', (df['Quantity']<0).sum())
print('Negative values in Price is:', (df['Price']<0).sum())
```

```
Negative values in Quantity is: 0
Negative values in Price is: 0
```

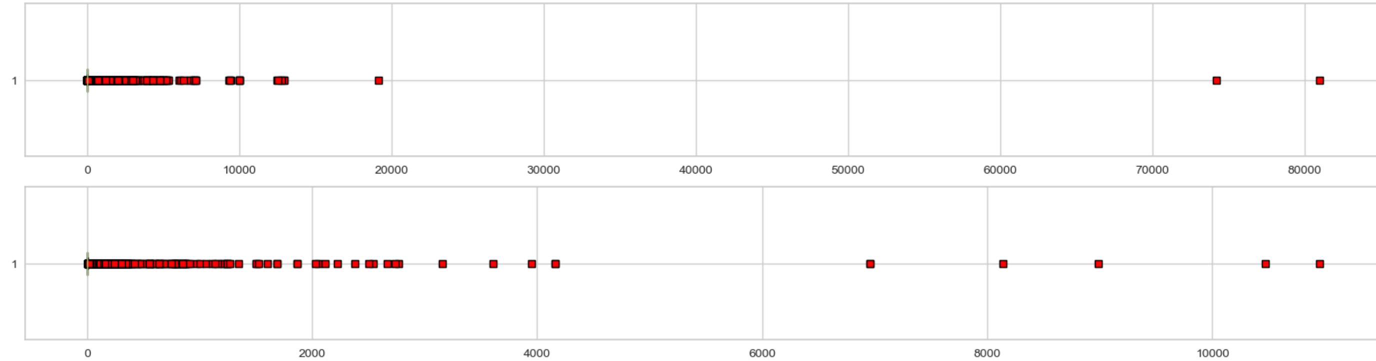
In [12]:
```
# 3.6 Cleaning outliers
def outliers_thresholds(dataframe, variable):
    quartile_1 = dataframe[variable].quantile(0.25)
    quartile_3 = dataframe[variable].quantile(0.75)
    interquantile_range = quartile_3 - quartile_1
    up_limit = quartile_3 + 1.5*interquantile_range
    low_limit = quartile_1 - 1.5*interquantile_range
    return up_limit, low_limit

def replace_with_thresholds(dataframe, variable):
    up_limit, low_limit = outliers_thresholds(dataframe, variable)
    dataframe.loc[(dataframe[variable]<low_limit), variable] = low_limit
    dataframe.loc[(dataframe[variable]>up_limit), variable] = up_limit

print(outliers_thresholds(df, 'Quantity'))
print(outliers_thresholds(df, 'Price'))
```

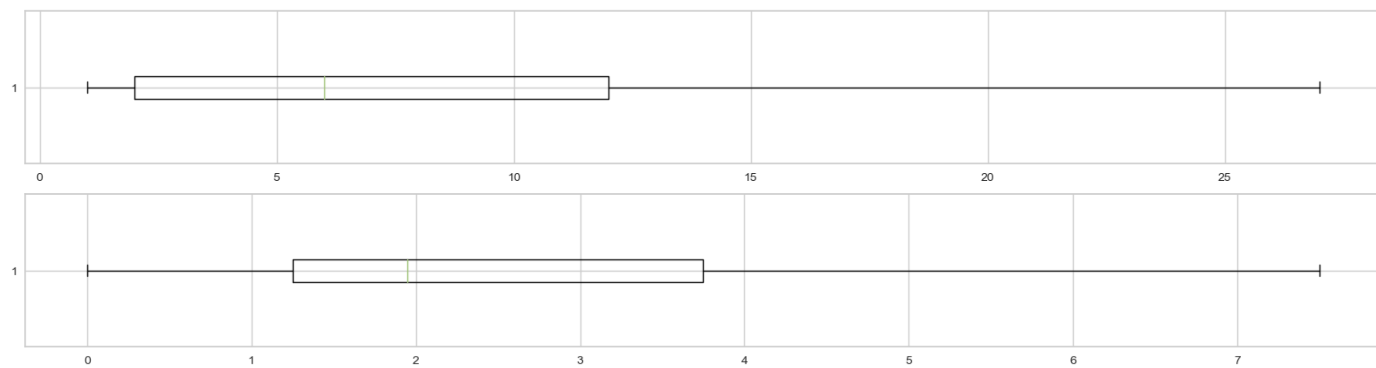
```
(27.0, -13.0)
(7.5, -2.5)
```

In [13]:
```
# Observing them before removing outliers.
fig, ax = plt.subplots(2,1, figsize = (20,5))
col_list = ["Quantity","Price"]
for i in range(0,2):
    ax[i].boxplot(df[col_list[i]],flierprops = dict(marker = "s", markerfacecolor = "red
plt.show()
```



In [14]:
```
# Apply the function to remove the outliers
replace_with_thresholds(df, 'Quantity')
replace_with_thresholds(df, 'Price')
```

In [15]:
```
# Observing them after removing outliers.
fig, ax = plt.subplots(2,1, figsize = (20,5))
for i in range(0,2):
    ax[i].boxplot(df[col_list[i]],flierprops = dict(marker = "s", markerfacecolor = "red
plt.show()
```



Data is clean now

4. EDA: Feature Engineer

In [16]:
```
#Creating new feature Revenue
df['Revenue'] = df['Quantity']*df['Price']
df
```

Out[16]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	CustomerID	Country	Revenue
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom	83.40
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom	81.00
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom	81.00
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	27	2009-12-01 07:45:00	2.10	13085.0	United Kingdom	56.70
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom	30.00

...	...	...	...	...	...	...	...	...	...
1067366	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France	12.60
1067367	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France	16.60
1067368	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France	16.60
1067369	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France	14.85
1067370	581587	POST	POSTAGE	1	2011-12-09 12:50:00	7.50	12680.0	France	7.50

779495 rows × 9 columns

In [17]:
```
print('Max Date:', df['InvoiceDate'].max())
print('Min Date:', df['InvoiceDate'].min())
```

```
Max Date: 2011-12-09 12:50:00
Min Date: 2009-12-01 07:45:00
```

In [18]:
```
# Set latest date is 2011-12-10 as the last invoice is on 2011-12-09
Latest_Date = dt.datetime(2011,12,10)
```

In [19]:
```
#Creating the RFM features with subsets of CustomerID
RFM = df.groupby(df['CustomerID']).agg({
    'InvoiceDate': lambda x: (Latest_Date - x.max()).days,
    'Invoice': lambda x: x.nunique(),
    'Revenue': lambda x: x.sum()})
RFM.dtypes
```

Out[19]:
```
InvoiceDate      int64
Invoice          int64
Revenue        float64
dtype: object
```

In [20]:
```
#Renaming column names to Recency, Frequency and Monetary
RFM.rename(columns = {
    'InvoiceDate' : 'Recency',
    'Invoice' : 'Frequency',
    'Revenue' : 'Monetary'}, inplace=True)

RFM.reset_index().head().style.set_properties(**{"background-color": "#cd5c5c","color":
```

Out[20]:

	CustomerID	Recency	Frequency	Monetary
0	12346.000000	325	12	400.940000
1	12347.000000	2	8	4473.220000
2	12348.000000	75	5	779.730000
3	12349.000000	18	4	3347.990000
4	12350.000000	310	1	301.900000

- The Fourth varibale of RFM, InterPurchase Time, is a measure of average time gap between total shopping trips by a customer.

- The Interpurchase Time is calcluted as fallows :
- T = L/F = (Tn - T1)/F
- Note: We consider only those customers who made purchase more than once.

In [21]:
```
RFM = RFM[RFM['Frequency']>1]
RFM.reset_index().head().style.set_properties(**{"background-color": "#cd5c5c","color":
```

Out[21]:

	CustomerID	Recency	Frequency	Monetary
0	12346.000000	325	12	400.940000
1	12347.000000	2	8	4473.220000
2	12348.000000	75	5	779.730000
3	12349.000000	18	4	3347.990000
4	12352.000000	36	10	1739.490000

In [22]:
```
Shopping_Cycle = df.groupby(df['CustomerID']).agg({'InvoiceDate': lambda x: (x.max() - x
```

In [23]:
```
RFM['Shopping_Cycle'] = Shopping_Cycle
RFM.reset_index().head().style.set_properties(**{"background-color": "#cd5c5c","color":
```

Out[23]:

	CustomerID	Recency	Frequency	Monetary	Shopping_Cycle
0	12346.000000	325	12	400.940000	400
1	12347.000000	2	8	4473.220000	402
2	12348.000000	75	5	779.730000	362
3	12349.000000	18	4	3347.990000	570
4	12352.000000	36	10	1739.490000	356

In [24]:
```
RFM['InterPurchase_Time'] = RFM['Shopping_Cycle'] // RFM['Frequency']
RFM.reset_index().head().style.set_properties(**{"background-color": "#cd5c5c","color":

RFMT = RFM[["Recency","Frequency","Monetary","InterPurchase_Time"]]
RFMT.reset_index().head().style.set_properties(**{"background-color": "#cd5c5c","color":
```

Out[24]:

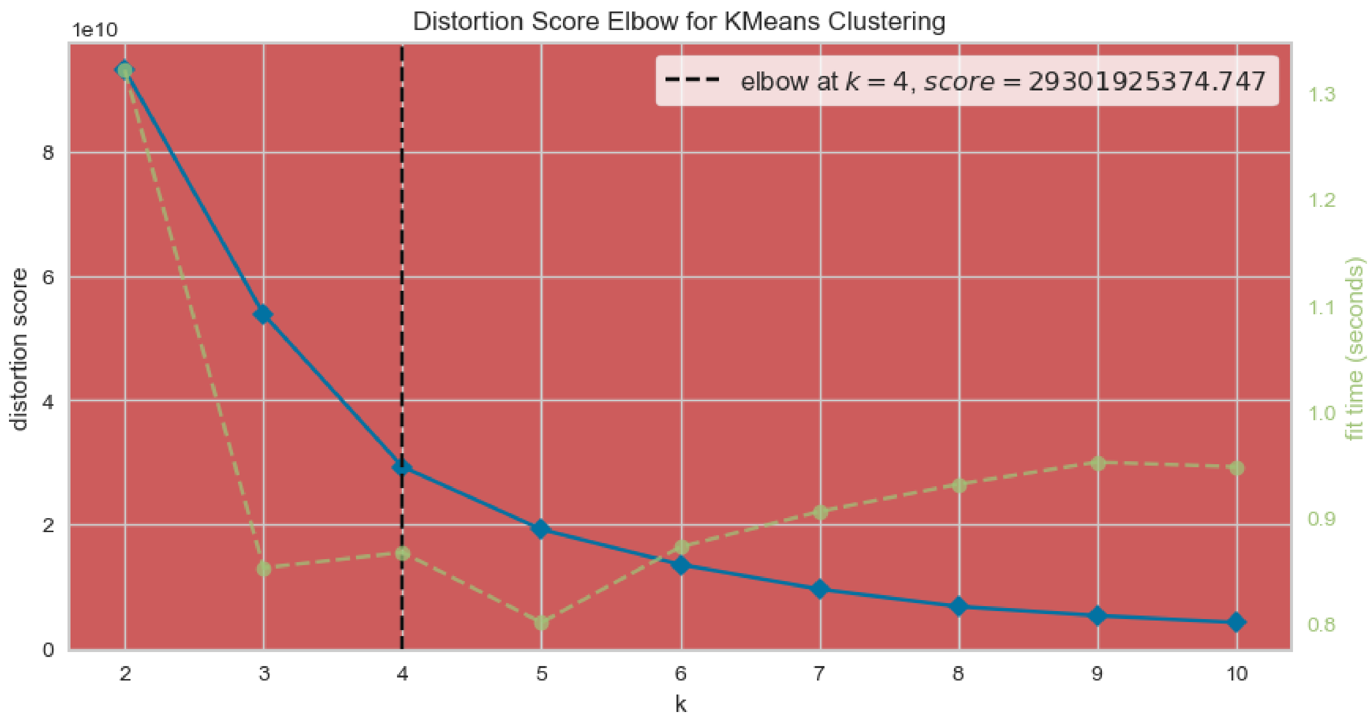
	CustomerID	Recency	Frequency	Monetary	InterPurchase_Time
0	12346.000000	325	12	400.940000	33
1	12347.000000	2	8	4473.220000	50
2	12348.000000	75	5	779.730000	72
3	12349.000000	18	4	3347.990000	142
4	12352.000000	36	10	1739.490000	35

RFMT Model is ready for segmentation

5. Modelling with KMeans Algorithm

In [25]:
```
# Finding initial K value using Elbow Method
plt.figure(figsize=(10,5))
```

```
ax = plt.axes()
ax.set_facecolor("#cd5c5c")
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(RFMT)
Elbow_M.show()
print("Therefore K = 4")
```




```
Therefore K = 4
```

In [26]:

```
#Fitting KMeans Model
kmeans = KMeans(n_clusters=4,max_iter=50)
kmeans.fit(RFMT)
```

Out[26]:

```
KMeans
KMeans(max_iter=50, n_clusters=4)
```

In [27]:

```
RFMT["Clusters"]=kmeans.labels_
RFMT.head().style.set_properties(**{"background-color": "#cd5c5c","color": "black", "bor
```

Out[27]:

	Recency	Frequency	Monetary	InterPurchase_Time	Clusters
CustomerID					
12346.000000	325	12	400.940000	33	0
12347.000000	2	8	4473.220000	50	0
12348.000000	75	5	779.730000	72	0
12349.000000	18	4	3347.990000	142	0
12352.000000	36	10	1739.490000	35	0

6. Model : Evaluation

In [28]:

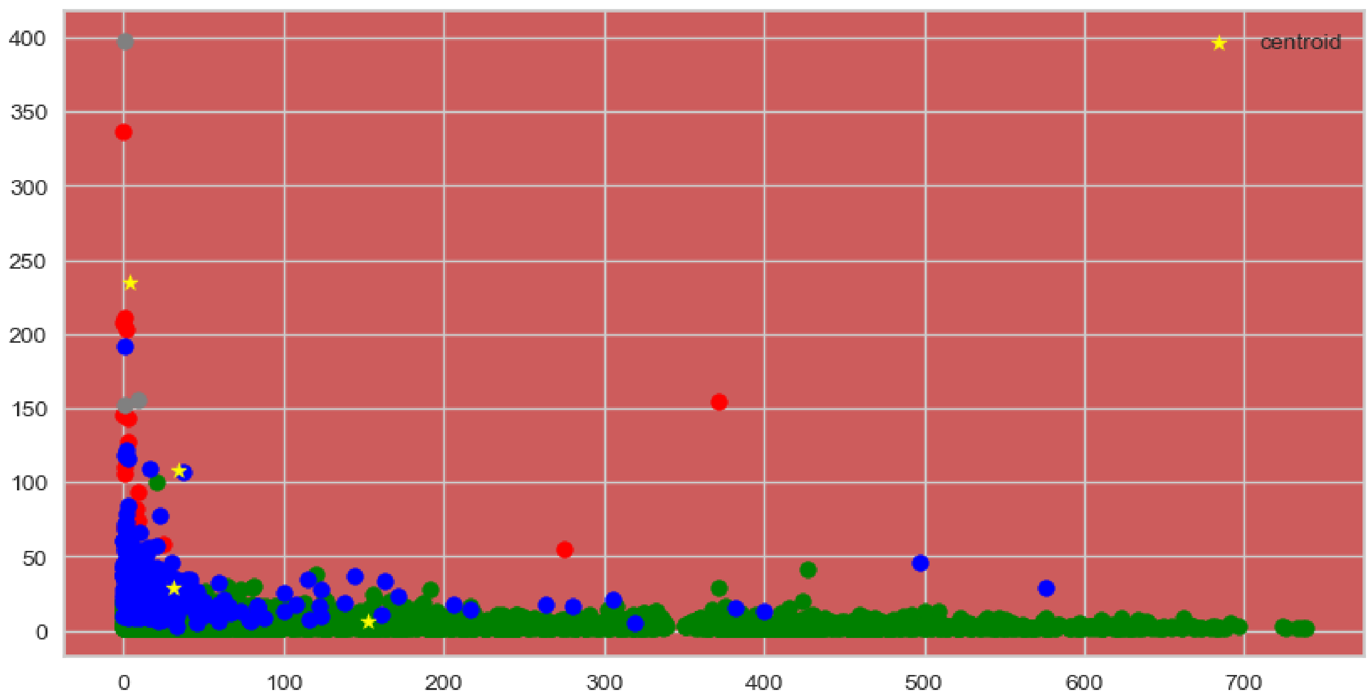
```
# how well the clusters are?:
#centriods
kmeans.cluster_centers_
```

Out[28]:
```
array([[1.52262942e+02, 5.86186571e+00, 1.55692095e+03, 7.25317786e+01],
       [3.47727273e+01, 1.08227273e+02, 5.11814165e+04, 8.68181818e+00],
       [3.66666667e+00, 2.35333333e+02, 1.70986510e+05, 3.00000000e+00],
       [3.07256098e+01, 2.86250000e+01, 1.13062049e+04, 3.04542683e+01]])
```

In [29]:
```python
# grouping the data in accordance with each cluster seperately
one = RFMT[RFMT["Clusters"]==0]
two = RFMT[RFMT["Clusters"]==1]
three = RFMT[RFMT["Clusters"]==2]
four = RFMT[RFMT["Clusters"]==3]

#Checking the quality of clustering in the data set
plt.figure(figsize=(10,5))
ax = plt.axes()
ax.set_facecolor("#cd5c5c")
plt.scatter(one["Recency"],one["Frequency"],color='green')
plt.scatter(two["Recency"],two["Frequency"],color='red')
plt.scatter(three["Recency"],three["Frequency"],color='grey')
plt.scatter(four["Recency"],four["Frequency"],color='blue')
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],color="yellow",mar
plt.legend()
plt.show
```

Out[29]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [30]:
```python
from sklearn.metrics import silhouette_score
print("Silhouette score :",silhouette_score(RFMT, kmeans.labels_, metric='euclidean'))
```

```
Silhouette score : 0.797397771198647
```