In [102…

```
# اول خطوه : نكتب اسم المكتبات
# Import packages

import pandas as PD
import numpy as NP
import matplotlib.pyplot as PLT
import seaborn as SNS

# Input data files are available in the "../input/" directory
import time, warnings
import datetime as DT

#Visualizations
import matplotlib.pyplot as PLT
from pandas.plotting import scatter_matrix
%matplotlib inline
import seaborn as SNS

warnings.filterwarnings("ignore")
```

In [103…

```
#Load the dataset

retail_DF = PD.read_csv('/kaggle/input/customer-segmentation/customer segmintation.csv',encoding="ISO-8859-1",dtype={'CustomerID': str,'InvoiceID': str})
retail_DF.head()
```

Out[103]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom

In [104…

```
# Information about the dataset
#طبعت المعلومات الموجودة في الداتا بيس

retail_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  object
 7   Country      541909 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 33.1+ MB
```

In [105…

```
# Is France on file?
retail_F = retail_DF[retail_DF['Country']=='France']

#check the shape
retail_F.shape
```

Out[105]:

```
(8557, 8)
```

In [106…

```
# Is USA on file?
retail_USA = retail_DF[retail_DF['Country']=='USA']

#check the shape
retail_USA.shape
```

Out[106]:

```
(291, 8)
```

In [107…

```
#Is Germany on file?
retail_G = retail_DF[retail_DF['Country']=='Germany']

#check the shape
retail_G.shape
```

Out[107]:

```
(9495, 8)
```

So I'll choose Germany to do the RFM analysis

Prepare the Data

تجهيز البيانات

In [108…

```
# Remove canceled orders
# نحذف الاوردر المتكنسله

retail_G = retail_G[retail_G['Quantity']>0]

#check the shape
retail_G.shape
```

Out[108]:

```
(9042, 8)
```

In [109…

```
#remove rows where customerID are NA
# نحذف العملاء الي ما لهم ID

retail_G.dropna(subset=['CustomerID'],how='all',inplace=True)

#check the shape
retail_G.shape
```

Out[109]:

```
(9042, 8)
```

In [110…

```
#restrict the data to one full year because it's better to use a metric per Months or Years in RFM

retail_G = retail_G[retail_G['InvoiceDate']>= "2010-1-12"]

#check the shape
retail_G.shape
```

Out[110]:

```
(4757, 8)
```

In [111…

```
print("Summary..")
print("-----------------")

#exploring the unique values of each attribute

print("Number of transactions: ", retail_G['InvoiceNo'].nunique())
print("Number of products bought: ",retail_G['StockCode'].nunique())
print("Number of customers:", retail_G['CustomerID'].nunique() )
print("Percentage of customers NA: ", round(retail_G['CustomerID'].isnull().sum() * 100 / len(retail_DF),2),"%" )
```

```
Summary..
-----------------
Number of transactions:  234
Number of products bought:  1248
Number of customers: 70
Percentage of customers NA:  0.0 %
```

Recency

How many days ago was the customer's last purchase

In [112…

```
#last date available in our dataset
#آخر تاريخ متاح في الداتا بيس المجوده عندنا

retail_G['InvoiceDate'].max()
```

Out[112]:

```
'9/8/2011 15:22'
```

In [113…

```
#The last date we have is 2011-8-09 so we will use it as reference
# اخر تاريخ موجود هو 2011-8-9 علشان كذا حنستخدمه كمرجع

now = DT.date(2011,8,9)
print(now)
```

```
2011-08-09
```

In [114…

```
#Create a new column called date which contains the date of invoice only
# حنسوي عمود نسميه Date و حيتضمن بس تاريخ الفاتوره

retail_G['date'] = PD.DatetimeIndex(retail_G['InvoiceDate']).date
```

In [115…

```
# Double check the data
# نرجع نتأكد مره ثانيه من البيانات

retail_G.head()
```

Out[115]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	date
105885	545295	22567	20 DOLLY PEGS RETROSPOT	20	3/1/2011 12:09	1.25	12709	Germany	2011-03-01
105886	545295	84029E	RED WOOLLY HOTTIE WHITE HEART.	10	3/1/2011 12:09	3.75	12709	Germany	2011-03-01
105887	545295	21485	RETROSPOT HEART HOT WATER BOTTLE	24	3/1/2011 12:09	4.25	12709	Germany	2011-03-01
105888	545295	22354	RETROSPOT PADDED SEAT CUSHION	20	3/1/2011 12:09	3.75	12709	Germany	2011-03-01
105889	545295	22333	RETROSPOT PARTY BAG + STICKER SET	30	3/1/2011 12:09	1.65	12709	Germany	2011-03-01

In [116…

```
# Group by customers and check last date of purshace
# حنطلع اخر عمليات شراء و مين هما العملاء الي طلبوا

recency_DF = retail_G.groupby(by='CustomerID', as_index=False)['date'].max()
recency_DF.columns = ['CustomerID','LastPurshaceDate']

recency_DF.head()
```

Out[116]:

	CustomerID	LastPurshaceDate
0	12426	2011-05-29
1	12468	2011-06-05
2	12471	2011-09-22
3	12472	2011-07-24
4	12473	2011-08-17

In [117…

```
# Calculate Recency

recency_DF['Recency'] = recency_DF['LastPurshaceDate'].apply(lambda x: (now - x).days)
```

In [118…

```
recency_DF.head()
```

Out[118]:

	CustomerID	LastPurshaceDate	Recency
0	12426	2011-05-29	72
1	12468	2011-06-05	65
2	12471	2011-09-22	-44
3	12472	2011-07-24	16
4	12473	2011-08-17	-8

Frequency

How many invoices are registered by the same customer

In [119…

```
# Drop duplicates
retail_G_copy = retail_G
retail_G_copy.drop_duplicates(subset=['InvoiceNo', 'CustomerID'], keep="first", inplace=True)

#calculate frequency of purchases
frequency_DF = retail_G_copy.groupby(by=['CustomerID'], as_index=False)['InvoiceNo'].count()
frequency_DF.columns = ['CustomerID','Frequency']

frequency_DF.head()
```

Out[119]:

	CustomerID	Frequency
0	12426	1
1	12468	1
2	12471	14
3	12472	3
4	12473	2

Monetary

How much money did the customer spent over time?

In [120…

```
# Create column total cost
retail_G['TotalCost'] = retail_G['Quantity'] * retail_G['UnitPrice']

monetary_DF = retail_G.groupby(by='CustomerID',as_index=False).agg({'TotalCost': 'sum'})
monetary_DF.columns = ['CustomerID','Monetary']

monetary_DF.head()
```

Out[120]:

	CustomerID	Monetary
0	12426	17.70
1	12468	13.20
2	12471	856.68
3	12472	45.30
4	12473	28.20

Create RFM Table

In [121…

```
# Merge recency dataframe with frequency dataframe
# تسوى دمج بين recency dataframe و frequency dataframe

temp_DF = recency_DF.merge(frequency_DF,on='CustomerID')
temp_DF.head()

# Merge with monetary dataframe to get a table with the 3 columns
rfm_DF = temp_DF.merge(monetary_DF,on='CustomerID')

# Use CustomerID as index
rfm_DF.set_index('CustomerID',inplace=True)

# Check the head
rfm_DF.head()
```

Out[121]:

	LastPurshaceDate	Recency	Frequency	Monetary
CustomerID				
12426	2011-05-29	72	1	17.70
12468	2011-06-05	65	1	13.20
12471	2011-09-22	-44	14	856.68
12472	2011-07-24	16	3	45.30
12473	2011-08-17	-8	2	28.20

RFM Table Correctness verification

In [122…

```
retail_G[retail_G['CustomerID']=='12426']
```

Out[122]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	date	TotalCost
207305	554985	20665	RED RETROSPOT PURSE	6	5/29/2011 12:26	2.95	12426	Germany	2011-05-29	17.7

In [123…

```
(now - DT.date(2011,5,29)).days == 72
```

Out[123]:

```
True
```

Customer segments with RFM Model

شرائح من العملاء مع RFM

We assign a score from 1 to 4 to Recency, Frequency and Monetary. Four is the best/highest value, and one is the lowest/worst value

حنقسم العملاء حسب درجاتهم من 1 الى 4. 4 تعتبر افضل و اعلى درجه, 1 تعتبر ادنى و اسوء درجه

RFM Quartiles

In [124…

```
quantiles = rfm_DF.quantile(q=[0.25,0.5,0.75])
quantiles
```

Out[124]:

	Recency	Frequency	Monetary
0.25	-41.00	1.0	25.53
0.50	-16.00	2.0	53.74
0.75	29.25	4.0	140.31

In [125…

```
quantiles.to_dict()
```

Out[125]:

```
{'Recency': {0.25: -41.0, 0.5: -16.0, 0.75: 29.25},
 'Frequency': {0.25: 1.0, 0.5: 2.0, 0.75: 4.0},
 'Monetary': {0.25: 25.53, 0.5: 53.74, 0.75: 140.31}}
```

- Creation of RFM Segments

Create two segmentation classes since, high recency is bad, while high frequency and monetary value is good

In [126…

```
# Arguments (x = value, p = recency, monetary_value, frequency, d = quartiles dict)
def RScore(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1

# Arguments (x = value, p = recency, monetary_value, frequency, k = quartiles dict)
def FMScore(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4


 #create rfm segmentation table
rfm_segmentation = rfm_DF
rfm_segmentation['R_Quartile'] = rfm_segmentation['Recency'].apply(RScore, args=('Recency',quantiles,))
rfm_segmentation['F_Quartile'] = rfm_segmentation['Frequency'].apply(FMScore, args=('Frequency',quantiles,))
rfm_segmentation['M_Quartile'] = rfm_segmentation['Monetary'].apply(FMScore, args=('Monetary',quantiles,))

rfm_segmentation.head()
```

Out[126]:

	LastPurshaceDate	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile
CustomerID							
12426	2011-05-29	72	1	17.70	1	1	1
12468	2011-06-05	65	1	13.20	1	1	1
12471	2011-09-22	-44	14	856.68	4	4	4
12472	2011-07-24	16	3	45.30	2	3	2
12473	2011-08-17	-8	2	28.20	2	2	2

Now that we have the score of each customer, we can represent our customer segmentation. First, we need to combine the scores R_Quartile, F_Quartile,M_Quartile together

بعد ما طلعنا درجه كل عميل حنجمع كل الدرجات مع بعض

In [127…

```
rfm_segmentation['RFMScore'] = rfm_segmentation.R_Quartile.map(str) \
                            + rfm_segmentation.F_Quartile.map(str) \
                            + rfm_segmentation.M_Quartile.map(str)
rfm_segmentation.head()
```

Out[127]:

	LastPurshaceDate	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile	RFMScore
CustomerID								
12426	2011-05-29	72	1	17.70	1	1	1	111
12468	2011-06-05	65	1	13.20	1	1	1	111
12471	2011-09-22	-44	14	856.68	4	4	4	444
12472	2011-07-24	16	3	45.30	2	3	2	232
12473	2011-08-17	-8	2	28.20	2	2	2	222

Let's see who are our Champions (best customers)

حنطلع و حنشوف افضل عميل الي تكون كل درجاته 4

In [128…

```
rfm_segmentation[rfm_segmentation['RFMScore']=='444'].sort_values('Monetary', ascending=False).head(10)
```

Out[128]:

	LastPurshaceDate	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile	RFMScore
CustomerID								
12471	2011-09-22	-44	14	856.68	4	4	4	444
12619	2011-09-23	-45	7	821.77	4	4	4	444
12621	2011-09-19	-41	13	467.90	4	4	4	444
12709	2011-09-29	-51	9	297.30	4	4	4	444
12720	2011-09-20	-42	14	264.78	4	4	4	444
12647	2011-09-22	-44	7	252.53	4	4	4	444

In [129…

```
# Total RFM score

rfm_segmentation['Total Score'] = rfm_segmentation['R_Quartile'] + rfm_segmentation['F_Quartile'] + \
rfm_segmentation['M_Quartile']

rfm_segmentation.head()
```

Out[129]:

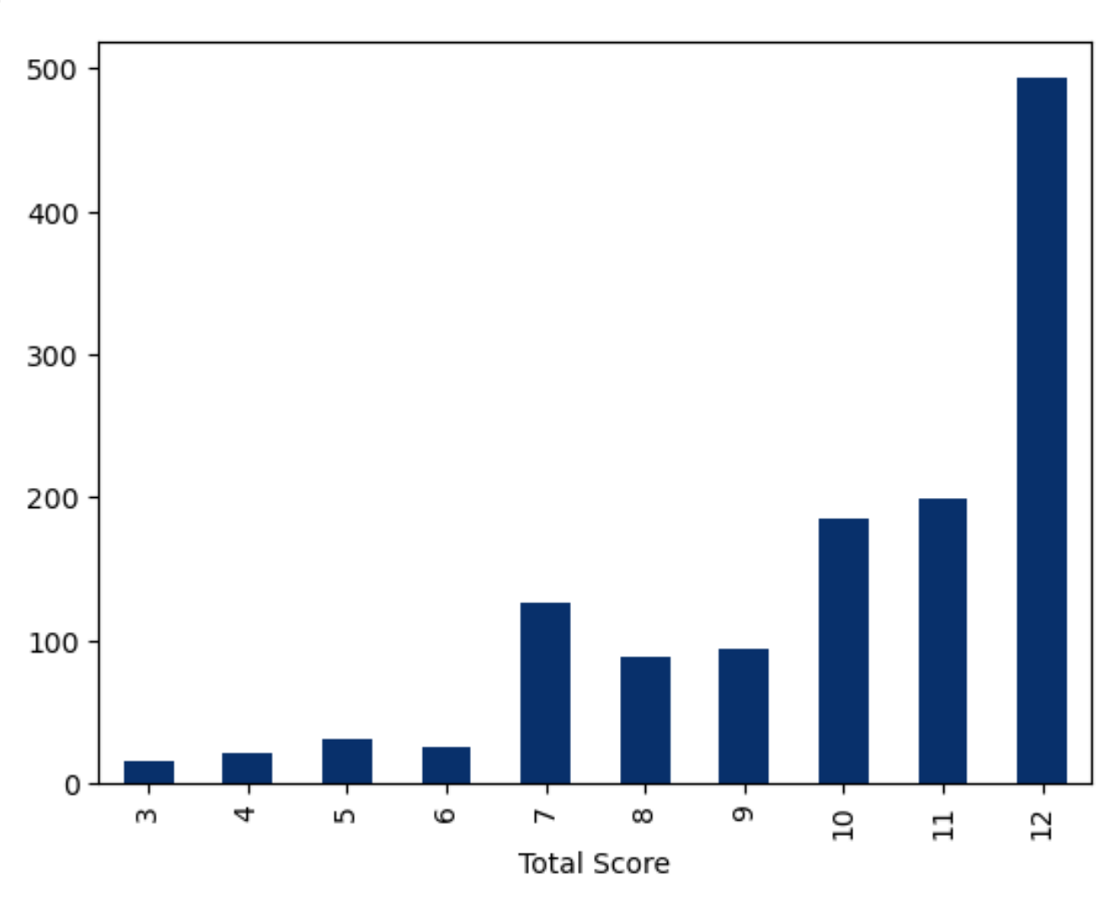
	LastPurshaceDate	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile	RFMScore	Total Score
CustomerID									
12426	2011-05-29	72	1	17.70	1	1	1	111	3
12468	2011-06-05	65	1	13.20	1	1	1	111	3
12471	2011-09-22	-44	14	856.68	4	4	4	444	12
12472	2011-07-24	16	3	45.30	2	3	2	232	7
12473	2011-08-17	-8	2	28.20	2	2	2	222	6

In [130…

```
rfm_segmentation.groupby('Total Score')['Monetary'].mean().plot(kind='bar', colormap='Blues_r')
```

Out[130]:

```
<Axes: xlabel='Total Score'>
```

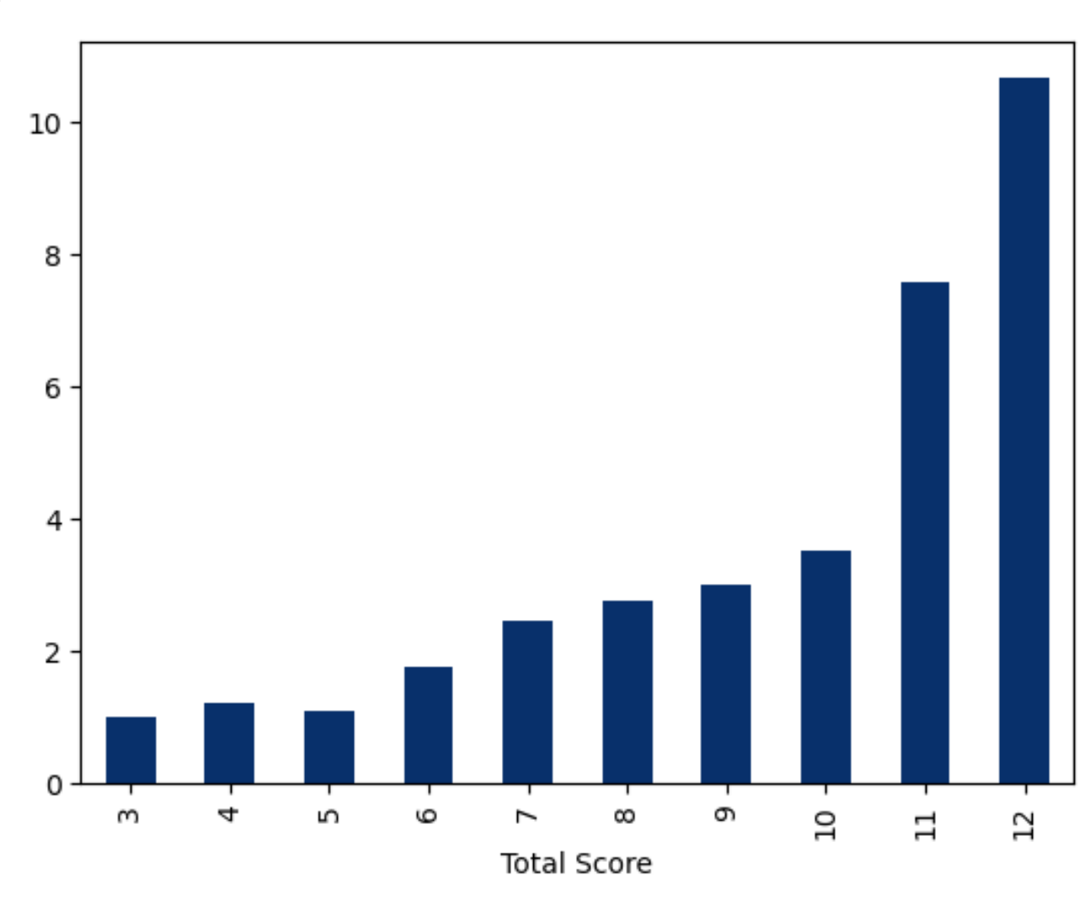


In [131…

```
rfm_segmentation.groupby('Total Score')['Frequency'].mean().plot(kind='bar', colormap='Blues_r')
```

Out[131]:

```
<Axes: xlabel='Total Score'>
```

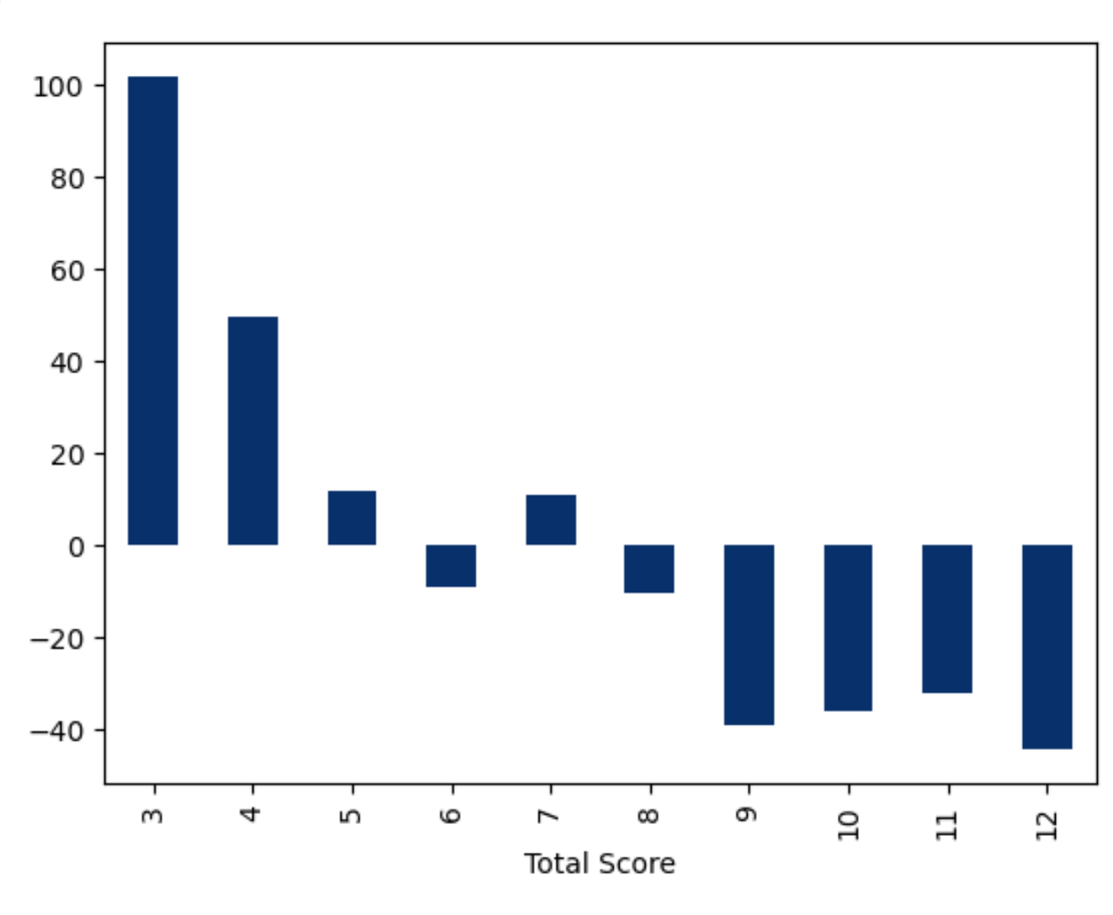


In [132…

```
rfm_segmentation.groupby('Total Score')['Recency'].mean().plot(kind='bar', colormap='Blues_r')
```

Out[132]:

```
<Axes: xlabel='Total Score'>
```



- Generating Segments Based on RFM Scores

In [133…

```
rfm_DF['Segment'] = rfm_DF['RFMScore']
rfm_DF.head()
```

Out[133]:

	LastPurshaceDate	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile	RFMScore	Total Score	Segment
CustomerID										
12426	2011-05-29	72	1	17.70	1	1	1	111	3	111
12468	2011-06-05	65	1	13.20	1	1	1	111	3	111
12471	2011-09-22	-44	14	856.68	4	4	4	444	12	444
12472	2011-07-24	16	3	45.30	2	3	2	232	7	232
12473	2011-08-17	-8	2	28.20	2	2	2	222	6	222

In [134…

```
seg_map = {
    r'[1-2][1-2]': 'hibernating',
    r'[1-2][3-4]': 'at_Risk',
    r'[1-2]5': 'cant_loose',
    r'3[1-2]': 'about_to_sleep',
    r'33': 'need_attention',
    r'[3-4][4-5]': 'loyal_customers',
    r'41': 'promising',
    r'51': 'new_customers',
    r'[4-5][2-3]': 'potential_loyalists',
    r'5[4-5]': 'champions'
}

rfm_DF['Segment'] = rfm_DF['Segment'].replace(seg_map, regex=True)
rfm_DF.reset_index(inplace=True)
rfm_DF.head()
```

Out[134]:

	CustomerID	LastPurshaceDate	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile	RFMScore	Total Score	Segment
0	12426	2011-05-29	72	1	17.70	1	1	1	111	3	hibernating1
1	12468	2011-06-05	65	1	13.20	1	1	1	111	3	hibernating1
2	12471	2011-09-22	-44	14	856.68	4	4	4	444	12	loyal_customers4
3	12472	2011-07-24	16	3	45.30	2	3	2	232	7	at_Risk2
4	12473	2011-08-17	-8	2	28.20	2	2	2	222	6	hibernating2

- How many customers do we have in each segment?

حنشوف كم عميل عندنا في كل Segment ؟

In [135…

```
print("Best Customers: ",len(rfm_segmentation[rfm_segmentation['RFMScore']=='444']))
print('Loyal Customers: ',len(rfm_segmentation[rfm_segmentation['F_Quartile']==4]))
print("Big Spenders: ",len(rfm_segmentation[rfm_segmentation['M_Quartile']==4]))
print('Almost Lost: ', len(rfm_segmentation[rfm_segmentation['RFMScore']=='244']))
print('Lost Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='144']))
print('Lost Cheap Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='111']))
```

```
Best Customers:  6
Loyal Customers:  15
Big Spenders:  18
Almost Lost:  0
Lost Customers:  0
Lost Cheap Customers:  7
```