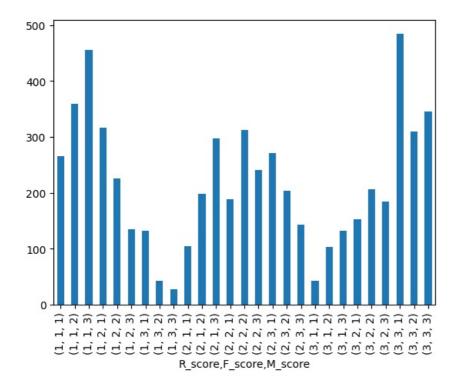
```
In [ ]: import matplotlib.pyplot as plt
             import os
             import pandas as pd
In []: df = pd.read csv(os.path.join("data", "online retail II preprocessed.csv")) # read the csv file
In [ ]: ### measure variables - recency, frequency, and monetary value
             rec = df.groupby(['Customer ID']).mean()['Recency'] # recency was calculated in preprocessing code
             ### code for calculating recency - from preprocessing code ###
             # df val is the dataset after dropping the nans or invalid (e.g. negative quantity) data
             present = dt.datetime(2012,1,1)
             df val['Recency'] = (present - df val.groupby(['Customer ID'])['InvoiceDate'].transform(max)).dt.days
             freq = df.groupby(['Customer ID']).count()['Invoice'] # frequency number of purchases since the first purchase
             # monetary value average spending per order
             df['Spending'] = df['Price']*df['Quantity']
             mon = df.groupby(['Customer ID']).mean()['Spending'] # monetary value average spending per order
             /var/folders/vg/9frsl 4j0xd91tddcqtwkgpr0000gn/T/ipykernel 23039/2013441190.py:2: FutureWarning: The default va
             lue of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to F
             alse. Either specify numeric_only or select only columns which should be valid for the function.
                rec = df.groupby(['Customer ID']).mean()['Recency'] # recency was calculated in preprocessing code
             /var/folders/vg/9frsl 4j0xd91tddcqtwkgpr0000gn/T/ipykernel 23039/2013441190.py:12: FutureWarning: The default v
             alue of numeric only in DataFrameGroupBy.mean is deprecated. In a future version, numeric only will default to
             False. Either specify numeric only or select only columns which should be valid for the function.
               mon = df.groupby(['Customer ID']).mean()['Spending'] # monetary value average spending per order
In [ ]: ### for each dimension, divide all customers into three groups evenly.
             seg = [1]*(len(rec)//3) + [2]*(len(rec) - len(rec)//3*2) + [3]*(len(rec)//3) # score: 1 for lowest, 2 for medium, 1 for lowest, 2 for medium, 2 for medium, 3 for lowest, 3 for lowest
             rec = rec.sort_values(ascending=False).to_frame()
             freq = freq.sort values().to frame("Frequency")
             mon = mon.sort_values().to_frame("MonetaryValue")
             rec['Recency score'] = seg
             freq['Freq_score'] = seg
             mon['MV score'] = seg
             df['R_score'] = [1]*len(df.index)
             df['F score'] = [1]*len(df.index)
             df['M score'] = [1]*len(df.index)
             df.loc[df['Customer ID'].isin(rec[rec['Recency score']==2].index),'R score'] = 2
             df.loc[df['Customer ID'].isin(rec[rec['Recency_score']==3].index),'R_score'] = 3
             df.loc[df['Customer ID'].isin(freq[freq['Freq score']==2].index),'F_score'] = 2
             df.loc[df['Customer ID'].isin(freq[freq['Freq_score']==3].index), 'F_score'] = 3
             df.loc[df['Customer ID'].isin(mon[mon['MV score']==2].index),'M score'] = 2
             df.loc[df['Customer ID'].isin(mon[mon['MV score']==3].index),'M score'] = 3
             rfm = df.groupby(['R_score', 'F_score', 'M_score'])
In []: pd.options.display.float format = '{:.2f}'.format
             print(rfm.nunique()['Customer ID'])
             print(rfm.nunique()['Customer ID']/rfm.nunique()['Customer ID'].sum()*100) # proportion
             %matplotlib inline
             rfm.nunique()['Customer ID'].plot.bar()
```

R_score	F_score	M_score	
1	1	1 2	266 359
		3	456
	2	1 2	317 225
		3	134
	3	1 2	132 43
		3	43 27
2	1	1	105
		2	198 297
	2	1	188
		2	313 241
	3	1	271
		2 3	204 143
3	1	1	43
		2	103
	2	3 1	132
	2	2	152 206
		3	184
	3	1	485
		2	309
		3	345
Name (iii	stomer ID	dtvne:	int64
Name: Cu R score	stomer ID F score		int64
		M_score 1	4.53
R_score	F_score	M_score 1 2	4.53 6.11
R_score	F_score 1	M_score 1 2 3	4.53 6.11 7.76
R_score	F_score	M_score 1 2 3	4.53 6.11 7.76 5.39
R_score	F_score 1	M_score 1 2 3 1 2 3	4.53 6.11 7.76
R_score	F_score 1	M_score 1 2 3 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25
R_score	F_score 1 2	M_score 1 2 3 1 2 3 1 2	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73
R_score 1	F_score 1 2 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46
R_score	F_score 1 2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37
R_score 1	F_score 1 2 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05
R_score 1	F_score 1 2 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20
R_score 1	F_score 1 2 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32
R_score 1	F_score 1 2 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61
R_score 1	F_score 1 2 3 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47
R_score 1	F_score 1 2 3 1 2 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 3	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47 2.43
R_score 1	F_score 1 2 3 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47
R_score 1	F_score 1 2 3 1 2 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47 2.43 0.73 1.75 2.25
R_score 1	F_score 1 2 3 1 2 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47 2.43 0.73 1.75 2.25 2.59
R_score 1	F_score 1 2 3 1 2 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47 2.43 0.73 1.75 2.25 2.59 3.50
R_score 1	F_score 1 2 3 1 2 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47 2.43 0.73 1.75 2.25 2.59 3.50 3.13 8.25
R_score 1	F_score 1 2 3 1 2 3 1 2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 3	4.53 6.11 7.76 5.39 3.83 2.28 2.25 0.73 0.46 1.79 3.37 5.05 3.20 5.32 4.10 4.61 3.47 2.43 0.73 1.75 2.25 2.59 3.50 3.13

Name: Customer ID, dtype: float64

Out[]: <Axes: xlabel='R_score,F_score,M_score'>



```
In []: print("number of countries in each group")
    print(rfm.nunique()['Country'])
    print("most frequent countries of the customers in each group")
    tmp = df.groupby('Customer ID').max().get(['R_score', 'F_score', 'M_score', 'Country'])
    print(tmp.groupby(['R_score', 'F_score', 'M_score'])['Country'].apply(pd.Series.mode))
```

```
number of countries in each group
R_score F_score M_score
                   1
                   2
                               14
                   3
                              17
                   1
         2
                               9
                   2
                               15
                   3
                               18
         3
                   1
                               3
                   2
                               3
                   3
                               9
2
                   1
                               4
         1
                   2
                               9
                   3
                               16
         2
                   1
                               3
                   2
                               18
                   3
                               20
         3
                   1
                               4
                   2
                               10
                   3
                               18
3
         1
                   1
                               2
                               7
                   2
                   3
                               10
         2
                   1
                               3
                   2
                               10
                   3
                               15
         3
                   1
                               6
                   2
                               11
```

Name: Country, dtype: int64

 $\hbox{most frequent countries of the customers in each group}\\$

R score	F_score	M score			
1	1	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
	2	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
	3	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
2	1	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
	2	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
	3	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
3	1	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
	2	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom
	3	1	0	United	Kingdom
		2	0	United	Kingdom
		3	0	United	Kingdom

Name: Country, dtype: object

```
In [ ]: print("which weekday were the purchases made most frequently?")
   tmp = df.groupby('Invoice').max().get(['R_score', 'F_score', 'M_score', 'Weekday'])
   print(tmp.groupby(['R_score', 'F_score', 'M_score'])['Weekday'].apply(pd.Series.mode))
```

```
R_score F_score M_score
                           1
                                           Tuesday
                                         Wednesday
                                    1
                                          Thursday
                           2
                                    0
                           3
                                    0
                                           Tuesday
                                    1
                                         Wednesday
                 2
                           1
                                    0
                                            Sunday
                           2
                                    0
                                          Thursday
                           3
                                    0
                                          Tuesday
                 3
                           1
                                    0
                                            Sunday
                                          Thursday
                           2
                                    0
                           3
                                    0
                                          Thursday
                                    1
                                         Wednesday
        2
                           1
                 1
                                    0
                                          Thursday
                           2
                                    0
                                          Thursday
                           3
                                    0
                                          Thursday
                 2
                           1
                                    0
                                          Thursday
                           2
                                    0
                                          Thursday
                           3
                                          Thursday
                                    0
                 3
                           1
                                    0
                                            Sunday
                           2
                                    0
                                          Thursday
                           3
                                    0
                                          Thursday
                           1
        3
                 1
                                    0
                                          Tuesday
                           2
                                    0
                                          Thursday
                           3
                                    0
                                         Wednesday
                 2
                           1
                                    0
                                            Sunday
                           2
                                    0
                                          Thursday
                           3
                                    0
                                          Thursday
                 3
                           1
                                    0
                                          Thursday
                           2
                                    0
                                          Thursday
                           3
                                    0
                                          Thursday
        Name: Weekday, dtype: object
In [ ]: print("contribution in aggrgated sales")
        contrib = rfm.sum()['Spending']
        print(contrib)
        print(contrib/contrib.sum()*100) # proportion
```

which weekday were the purchases made most frequently?

contrib.plot.bar()

1	1	1	44072.51
		2	91822.07
		3	297082.22
	2	1	128164.14
		2	209068.87
		3	360259.20
	3	1	190263.70
	9	2	176885.49
		3	274258.04
2	1	1	18088.95
2	1		
		2	58307.86
	_	3	320341.09
	2	1	100644.51
		2	315284.89
		3	616486.64
	3	1	625216.73
		2	721828.65
		3	959399.93
3	1	1	8162.32
3	1	2	30437.10
	_	3	291856.46
	2	1	85234.31
		2	205836.75
		3	503807.05
	3	1	1900280.51
		2	1543246.18
		3	7667092.99
Name: Sp	endina d	type: floa	
Hamer Sp	charing, a	cypci icou	
R score	Fscore		
R_score	F_score	M_score	
R_score 1	F_score 1	M_score	0.25
_	_	M_score 1 2	0.25 0.52
_	1	M_score 1 2 3	0.25 0.52 1.67
_	_	M_score 1 2 3	0.25 0.52 1.67 0.72
_	1	M_score 1 2 3	0.25 0.52 1.67
_	1	M_score 1 2 3	0.25 0.52 1.67 0.72
_	1	M_score 1 2 3 1	0.25 0.52 1.67 0.72 1.18
_	2	M_score 1 2 3 1 2 3	0.25 0.52 1.67 0.72 1.18 2.03
_	2	M_score 1 2 3 1 2 3 1 2	0.25 0.52 1.67 0.72 1.18 2.03 1.07
1	2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 3	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00
_	2	M_score 1 2 3 1 2 3 1 2 3 1 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55
1	2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10
1	2 3 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81
1	2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81
1	2 3 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78
1	1 2 3 1 2 2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47
1	2 3 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52
1	1 2 3 1 2 2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07
1	1 2 3 1 2 2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52
1	1 2 3 1 2 2	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 1 2 3 3 3 3	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07
2	1 2 3 1 2 3 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41
2	1 2 3 1 2 3 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41 0.05 0.17
2	1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41 0.05 0.17 1.64
2	1 2 3 1 2 3 3	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41 0.05 0.17 1.64 0.48
2	1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41 0.05 0.17 1.64 0.48 1.16
2	1 2 3 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	M_score 1 2 3 1 2	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41 0.05 0.17 1.64 0.48 1.16 2.84
2	1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	M_score 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41 0.05 0.17 1.64 0.48 1.16 2.84 10.71
2	1 2 3 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	M_score 1 2 3 1 2	0.25 0.52 1.67 0.72 1.18 2.03 1.07 1.00 1.55 0.10 0.33 1.81 0.57 1.78 3.47 3.52 4.07 5.41 0.05 0.17 1.64 0.48 1.16 2.84

contribution in aggregated sales R_score F_score M_score

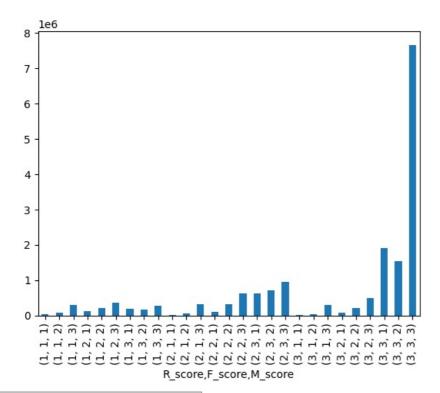
1

44072.51

/var/folders/vg/9frsl_4j0xd91tddcqtwkgpr0000gn/T/ipykernel_23039/2464572255.py:2: FutureWarning: The default va lue of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to Fa lse. Either specify numeric_only or select only columns which should be valid for the function. contrib = rfm.sum()['Spending']

Out[]: <Axes: xlabel='R_score,F_score,M_score'>

Name: Spending, dtype: float64



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js