In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
```

```
C:\Users\User\AppData\Roaming\Python\Python310\site-packages\pandas\core\arrays\mask
ed.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (ver
sion '1.3.5' currently installed).
  from pandas.core import (
```

In [2]:
```
data = pd.read_csv('Walmart_dataset.csv', encoding = 'unicode_escape')
data.head()
```

Out[2]:

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

In [3]:
```
data.shape
```

Out[3]:
```
(541909, 8)
```

In [4]:
```
data.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
```

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
InvoiceNo          0
StockCode          0
Description      1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID     135080
Country            0
dtype: int64
```

In [6]:
```
data.isnull().sum().sum()
```

Out[6]:
```
136534
```

In [7]:
```
data_clean = data.copy()
```

In [8]:
```
data_clean
```

Out[8]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0
...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680.0
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12680.0
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680.0
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680.0
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680.0

541909 rows × 8 columns

In [9]:
```
data_clean['InvoiceDate'] = pd.to_datetime(data_clean['InvoiceDate'])
```

In [10]:
```
data_clean.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  datetime64[ns]
 5   UnitPrice    541909 non-null  float64       
 6   CustomerID   406829 non-null  float64       
 7   Country      541909 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

In [11]:
```
print('Min Date of Purchase :', data_clean['InvoiceDate'].min())
print()
print('Max Date of Purchase :', data_clean['InvoiceDate'].max())
```

```
Min Date of Purchase : 2010-12-01 08:26:00

Max Date of Purchase : 2011-12-09 12:50:00
```

In [12]:
```
data_clean.duplicated().sum()
```

Out[12]:
```
5268
```

In [13]:
```
data_clean = data_clean.drop_duplicates(ignore_index = True)
```

In [14]:
```
data_clean.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
data_clean['Monetary'] = data_clean['Quantity'] * data_clean['UnitPrice']
```

In [16]:
```
data_clean
```

Out[16]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0
...	...	...	...	...	...	...	...
536636	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0
536637	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0
536638	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0
536639	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0
536640	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0

536641 rows × 9 columns

In [17]:
```
data_clean = data_clean[['InvoiceNo', 'CustomerID', 'Monetary', 'InvoiceDate']]
```

In [18]:
```
data_clean
```

Out[18]:

	InvoiceNo	CustomerID	Monetary	InvoiceDate
0	536365	17850.0	15.30	2010-12-01 08:26:00
1	536365	17850.0	20.34	2010-12-01 08:26:00
2	536365	17850.0	22.00	2010-12-01 08:26:00
3	536365	17850.0	20.34	2010-12-01 08:26:00
4	536365	17850.0	20.34	2010-12-01 08:26:00
...	...	...	...	...
536636	581587	12680.0	10.20	2011-12-09 12:50:00
536637	581587	12680.0	12.60	2011-12-09 12:50:00
536638	581587	12680.0	16.60	2011-12-09 12:50:00
536639	581587	12680.0	16.60	2011-12-09 12:50:00
536640	581587	12680.0	14.85	2011-12-09 12:50:00

536641 rows × 4 columns

In [19]:
```
#data_clean['InvoiceDate'] = pd.to_datetime(data_clean['InvoiceDate']).dt.date
```

In [20]:
```
data_clean.head(2)
```

Out[20]:

	InvoiceNo	CustomerID	Monetary	InvoiceDate
0	536365	17850.0	15.30	2010-12-01 08:26:00
1	536365	17850.0	20.34	2010-12-01 08:26:00

In [21]:
```
data_clean = data_clean.rename(columns = {'InvoiceDate' : 'Date'})
```

In [22]:
```
data_clean.isnull().sum()/len(data_clean) * 100
```

Out[22]:
```
InvoiceNo      0.000000
CustomerID    25.163377
Monetary       0.000000
Date           0.000000
dtype: float64
```

In [23]:
```
data_clean = data_clean[~(data_clean.CustomerID.isnull())]
data_clean = data_clean[~(data_clean.Monetary < 0)]
data_clean = data_clean[(data_clean.Monetary > 0)]
```

In [24]:
```
data_clean.isnull().sum()
```

Out[24]:
```
InvoiceNo     0
CustomerID    0
Monetary      0
Date          0
dtype: int64
```

In [25]:
```
print('Min Date of Purchase :', data_clean['Date'].min())
print()
print('Max Date of Purchase :', data_clean['Date'].max())
```

```
Min Date of Purchase : 2010-12-01 08:26:00

Max Date of Purchase : 2011-12-09 12:50:00
```

In [26]:
```
import datetime
new_date = datetime.datetime(2011, 12, 11)
new_date
```

Out[26]:
```
datetime.datetime(2011, 12, 11, 0, 0)
```

Analysing the data by RFM (Recency, Frequency, Monetary) approch

In [65]:
```
RFM = data_clean.groupby('CustomerID').agg({'Date': lambda x : (new_date - x.max())
RFM.reset_index()
```

Out[65]:

	CustomerID	Date	InvoiceNo	Monetary
0	12346.0	326	1	77183.60
1	12347.0	3	182	4310.00
2	12348.0	76	31	1797.24
3	12349.0	19	73	1757.55
4	12350.0	311	17	334.40
...	...	...	...	...
4333	18280.0	278	10	180.60
4334	18281.0	181	7	80.82
4335	18282.0	8	12	178.05
4336	18283.0	4	721	2045.53
4337	18287.0	43	70	1837.28

4338 rows × 4 columns

In [68]:
```
RFM.rename(columns = {'Date' : 'Recency', 'InvoiceNo' : 'Frequency'}, inplace = Tru
RFM.reset_index()
```

Out[68]:

	CustomerID	Recency	Frequency	Monetary
0	12346.0	326	1	77183.60
1	12347.0	3	182	4310.00
2	12348.0	76	31	1797.24
3	12349.0	19	73	1757.55
4	12350.0	311	17	334.40
...	...	...	...	...
4333	18280.0	278	10	180.60
4334	18281.0	181	7	80.82
4335	18282.0	8	12	178.05
4336	18283.0	4	721	2045.53
4337	18287.0	43	70	1837.28

4338 rows × 4 columns

In [69]:
```
RFM.columns
```

Out[69]:
```
Index(['Recency', 'Frequency', 'Monetary'], dtype='object')
```

In [70]:
```
RFM.describe()
```

Out[70]:

	Recency	Frequency	Monetary
count	4338.000000	4338.000000	4338.000000
mean	93.059474	90.523744	2048.688081
std	100.012264	225.506968	8985.230220
min	1.000000	1.000000	3.750000
25%	18.000000	17.000000	306.482500
50%	51.000000	41.000000	668.570000
75%	142.750000	98.000000	1660.597500
max	374.000000	7676.000000	280206.020000

In [71]:
```
#Splitting data into 4 segment.
quantile = RFM.quantile(q = [0.25, 0.5, 0.75])
quantile = quantile.to_dict()
quantile
```

Out[71]:
```
{'Recency': {0.25: 18.0, 0.5: 51.0, 0.75: 142.75},
 'Frequency': {0.25: 17.0, 0.5: 41.0, 0.75: 98.0},
 'Monetary': {0.25: 306.48249999999996,
  0.5: 668.5700000000002,
  0.75: 1660.5974999999999}}
```

In [72]:
```
def recent(x, p, d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.5]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4
```

In [73]:
```
def freq_money(x, p, d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.5]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
```

In [74]:
```
RFM.columns
```

Out[74]:
```
Index(['Recency', 'Frequency', 'Monetary'], dtype='object')
```

In [75]:
```
RFM['R'] = RFM['Recency'].apply(recent, args = ('Recency', quantile))
RFM['F'] = RFM['Frequency'].apply(freq_money, args = ('Frequency', quantile))
RFM['M'] = RFM['Monetary'].apply(freq_money, args = ('Monetary', quantile))
```

In [76]:
```
RFM.reset_index()
```

Out[76]:

	CustomerID	Recency	Frequency	Monetary	R	F	M
0	12346.0	326	1	77183.60	4	4	1
1	12347.0	3	182	4310.00	1	1	1
2	12348.0	76	31	1797.24	3	3	1
3	12349.0	19	73	1757.55	2	2	1
4	12350.0	311	17	334.40	4	4	3
...	...	...	...	...	...	...	...
4333	18280.0	278	10	180.60	4	4	4
4334	18281.0	181	7	80.82	4	4	4
4335	18282.0	8	12	178.05	1	4	4
4336	18283.0	4	721	2045.53	1	1	1
4337	18287.0	43	70	1837.28	2	2	1

4338 rows × 7 columns

In [77]:
```
RFM['RMF'] = RFM[['R', 'F', 'M']].sum(axis = 1)
```

In [78]:
```
RFM['RMF'].value_counts()
```

Out[78]:
```
RMF
10    519
9     461
7     459
8     456
3     443
5     428
6     410
11    391
4     390
12    381
Name: count, dtype: int64
```

In [79]:
```
RFM['RMF'].nunique()
```

Out[79]:
```
10
```

In [86]:
```
customer_type = ['Platinum', 'Gold', 'Silver', 'Bronze']
score = pd.qcut(RFM.RMF, q = 4, labels = customer_type)
RFM['customer_type'] = score.values
RFM = RFM.reset_index()

RFM
```

Out[86]:

	CustomerID	Recency	Frequency	Monetary	R	F	M	RMF	customer_type
0	12346.0	326	1	77183.60	4	4	1	9	Silver
1	12347.0	3	182	4310.00	1	1	1	3	Platinum
2	12348.0	76	31	1797.24	3	3	1	7	Gold
3	12349.0	19	73	1757.55	2	2	1	5	Platinum
4	12350.0	311	17	334.40	4	4	3	11	Bronze
...	...	...	...	...	...	...	...	...	...
4333	18280.0	278	10	180.60	4	4	4	12	Bronze
4334	18281.0	181	7	80.82	4	4	4	12	Bronze
4335	18282.0	8	12	178.05	1	4	4	9	Silver
4336	18283.0	4	721	2045.53	1	1	1	3	Platinum
4337	18287.0	43	70	1837.28	2	2	1	5	Platinum

4338 rows × 9 columns

In [ ]:

In [87]:
```
RFM.CustomerID
```

Out[87]:
```
0       12346.0
1       12347.0
2       12348.0
3       12349.0
4       12350.0
         ...
4333    18280.0
4334    18281.0
4335    18282.0
4336    18283.0
4337    18287.0
Name: CustomerID, Length: 4338, dtype: float64
```

In [42]:
```
RFM.to_csv('Segmentation Report.csv')
```

In [88]:
```
RFM.columns
```

Out[88]:
```
Index(['CustomerID', 'Recency', 'Frequency', 'Monetary', 'R', 'F', 'M', 'RMF',
       'customer_type'],
      dtype='object')
```

In [89]:
```
final_report = RFM.groupby('customer_type')[['Recency', 'Frequency', 'Monetary', 'C
    'Recency' : 'sum', 'Frequency' : 'sum', 'Monetary' : 'sum', 'CustomerID' : 'nun
```

In [90]:
```
final_report
```

Out[90]:

	customer_type	Recency	Frequency	Monetary	CustomerID
0	Platinum	25914	284840	6624735.110	1261
1	Gold	85375	75541	1540958.652	1325
2	Silver	123655	23851	567863.412	980
3	Bronze	168748	8460	153651.720	772

In [93]:
```
RFM.customer_type.value_counts()
```

Out[93]:
```
customer_type
Gold        1325
Platinum    1261
Silver       980
Bronze       772
Name: count, dtype: int64
```

In [94]:
```
final_report = final_report.rename(columns = {'CustomerID': 'Number of Buyers'})
final_report['%ages'] = round(final_report['Number of Buyers']/final_report['Number
final_report
```

Out[94]:

	customer_type	Recency	Frequency	Monetary	Number of Buyers	%ages
0	Platinum	25914	284840	6624735.110	1261	29.07
1	Gold	85375	75541	1540958.652	1325	30.54
2	Silver	123655	23851	567863.412	980	22.59
3	Bronze	168748	8460	153651.720	772	17.80

In [95]:
```
!pip install squarify
```

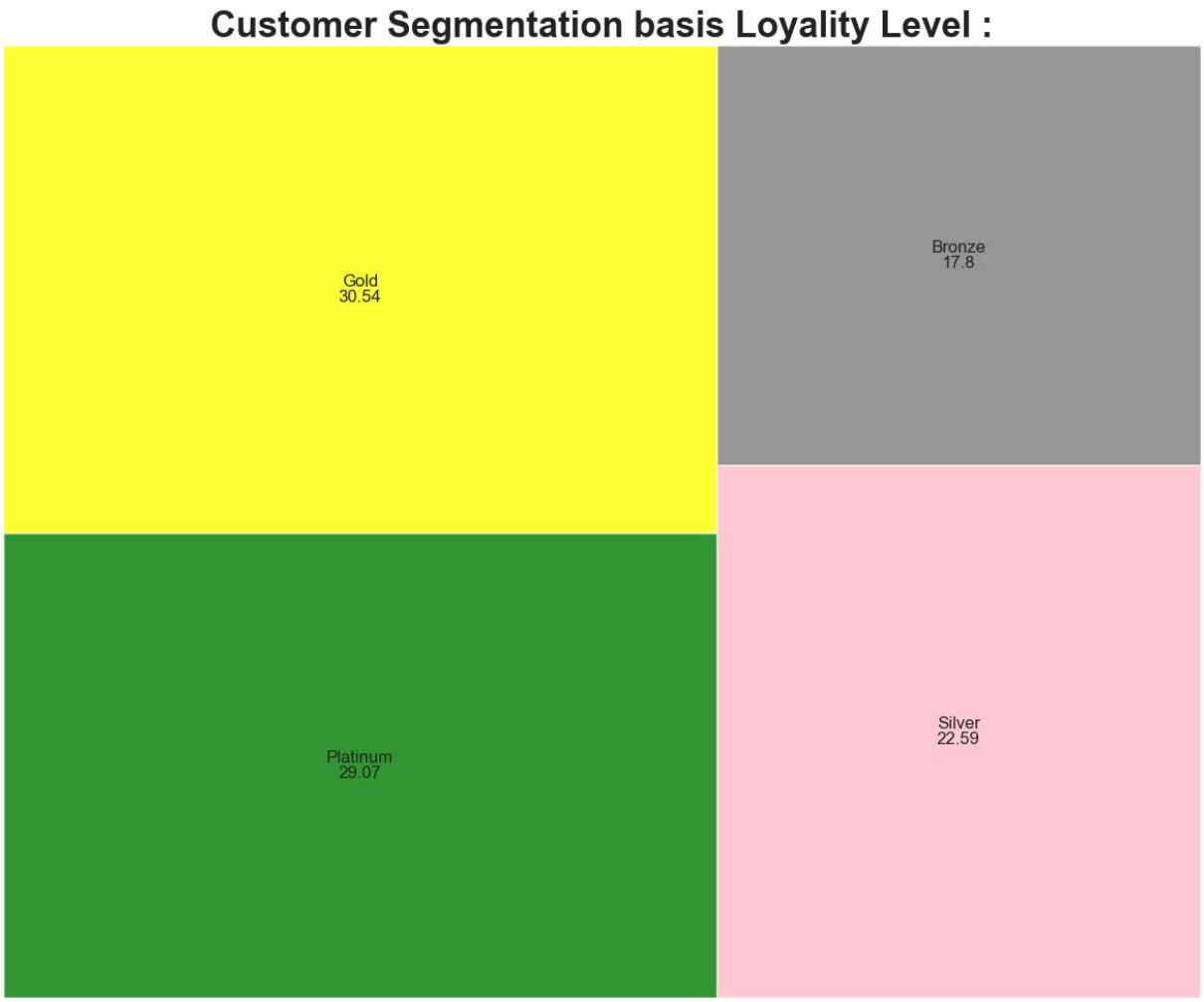
```
Defaulting to user installation because normal site-packages is not writeable
Collecting squarify
  Downloading squarify-0.4.4-py3-none-any.whl (4.1 kB)
Installing collected packages: squarify
Successfully installed squarify-0.4.4
```

In [96]:
```
import squarify
```

In [101…
```
fig = plt.gcf()
ax = fig.add_subplot()
fig.set_size_inches(15,12)
colors_dics = {'Platinum':'green','Gold':'yellow', 'Silver':'pink','Bronze':'grey'}

#squarify.plot(sizes = final_report1['NumBuyers'], color = colors_dics.values(),lab

squarify.plot(sizes=final_report['Number of Buyers'],color = colors_dics.values(),
              value=final_report['%ages'])
plt.title("Customer Segmentation basis Loyality Level :", fontsize=25, fontweight='
plt.axis('off')
plt.show()
```


Customer Segmentation basis Loyality Level :
Gold
30.54
Bronze
17.8
Platinum
29.07
Silver
22.59

In [ ]: